IESP
INTERNATIONAL EXASCALE SOFTWARE PROJECT

PRESENTATIONS AND WHITEPAPERS

MEETING 1

APRIL 7-8, 2009 SANTA FE, NEW MEXICO

MEETING 2

JUNE 28-29, 2009 PARIS, FRANCE

MEETING 1

APRIL 7-8, 2009 SANTA FE, NEW MEXICO

PRESENTATIONS

Improving HPC Software: Welcome
Pete Beckman (Argonne National Laboratory/University of Chicago) and Jack Dongarra (University of Tennessee/Oak Ridge National Laboratory)

e-Infrastructure in FP7: HPC related aspects
Catherine Riviére, GENCI, France

Development of an Over Petascale Computer in Japan
Satoshi Matsuoka, GSIC Center, Tokyo Institute of Technology/National Institute of Informatics

International Exascale Software Program
Abani Patra, NSF Office of Cyberinfrastructure

Improving HPC Software: Overview
Pete Beckman (Argonne National Laboratory/University of Chicago) and Jack Dongarra (University of Tennessee/Oak Ridge National Laboratory)

Thou Shalt Specialize or Commoditize? The Japanese Situation Towards Peta and Exascale
Satoshi Matsuoka, GSIC Center, Tokyo Institute of Technology/National Institute of Informatics

Technology and Architectures for Future Large-Scale Computing Systems
Rick Stevens, Argonne National Laboratory and University of Chicago.

Computational Science and HPC Software-Development in Europe
Thomas Lippert and Bernd Mohr,Forschungszentrum Jülich, JSC and Gauss Centre for Supercomputing e.V.

Slides from the panel:Software Barriers to HPC, Today and Tomorrow
Panel participants:Al Gara, Jean-Yves Berthou, Mitsuhisa Sato, Peggy Williams, Vivek Sarkar, Ann Trefethen

Science Drivers, Current HPC Software Development, and Platform Deployment Plans for the USA Horst Simon,Lawrence Berkeley National Laboratory and UC Berkeley

MEETING 1

APRIL 7-8, 2009 SANTA FE, NEW MEXICO

WHITEPAPERS

Musings on the Path Toward Exascale
Robert Lucas - ISI/USC

BSC Vision Towards Exascale
Mateo Valero, BSC

Software Challenges of Extreme Scale Computing
Michael Heroux - Sandia National Laboratory

Software and Exascale Computing
Bill Camp - Intel Corporation

Application Analysis and Porting in the PRACE Project
Peter Michielse - Netherlands National Computing Facilities Foundation (NCF)

The Application Perspective - Seeking Productivity and Performance
David Barkai - Intel Corporation

EDF white paper
J.Y. Berthou and J.F. Hamelin - EDF R&D

The Biggest Need: A New Model of Computation
Thomas Sterling - Louisiana State University

NSF IESP Whitepaper
Abani Patra, Rob Pennington, Ed Seidel - Office of Cyberinfrastructure, National Science Foundation

A Proposal for a Capability Centers Consortium
Bill Gropp, Mark Snir - NCSA and the University of Illinois at Urbana-Champaign

Slouching Towards Exascale
Rusty Lusk, Argonne National Laboratory

A Collaboration and Commercialization Model for Exascale Software Research
Mark Seager and Brent Gorda, Lawrence Livermore National Laboratory

The Case for A Hierarchal System Model for Linux Clusters
Mark Seager and Brent Gorda, Lawrence Livermore National Laboratory

PDE-based applications and solvers at extreme scale
David Keyes, Columbia University & SciDAC TOPS project

Developing a high performance computing/numericalanalysis roadmap
Ann Trefethen, Nick Higham, Ian Duff, and Peter Coveney

MEETING 2

JUNE 28-29, 2009 PARIS, FRANCE

PRESENTATIONS AND WHITEPAPERS

Improving HPC Software

Pete Beckman & Jack Dongarra

IESP the Need

- The largest scale systems are becoming more complex, with designs supported by large consortium
  - The software community has responded slowly
- Significant architectural changes arriving
  - Software must dramatically change
- Our ad hoc community coordinates poorly, both with other software components and with the vendors
  - Computational science could achieve more with improved development and coordination

Where We Are Today:

We are not prepared for the changes coming

- Hardware features are uncoordinated with software development
  - (power mgmt, multicore tools, math libraries, advanced memory models, etc)
- Only basic acceptance test software is delivered with platform
  - UPC, HPCToolkit, Optimized libraries, PAPI, can be *YEARS* late
- Vendors often "snapshot" key Open Source components and then deliver a stale code branch
  - Counterexample: A model that works – MPICH for BG/P
- Community codes unprepared for sea change in architectures
- Coordination via SOW/contract is poor and only involves 2 parties
- No global evaluation of key missing components

The IESP Workshops:

- Goal: Improve the world's simulation and modeling capability by improving the coordination and development of the HPC software environment.
  - Build a plan for how the international community can join together to improve software available for high-end systems over the next 2 to 10 years.
- The DOE (SC, NNSA), NSF, and EU have committed their support for the workshops.
- This is the first workshop in the series of three.

International Community Effort

- We believe this needs to be international collaboration for various reasons including:
  - The scale of investment
  - The need for international input on requirements
  - Europeans, Asians, and others are working on their own software that should be part of a larger vision for HPC.
- The process must be totally open

Executive Committee:

Co-Chair: Jack Dongarra, Univ, of Tennessee / ORNL, US
Co-Chair: Pete Beckman, Argonne National Laboratory, US
Franck Cappello, INRIA, FR
Thomas Lippert, Jülich Supercomputing Centre, DE
Satoshi Matsuoka, Tokyo Institute of Technology, JP
Paul Messina, Argonne National Laboratory, US

A Plan Could Include:

- Work with vendors to create the HPC equivalent to the ITRS (Int'l Tech Roadmap for Semiconductors)
    - Get community working on software before machine becomes available
- Community proposed unified roadmap for exascale software
- Identify missing components for future architectures and a plan to address them
- Develop models for working more closely with vendors
  - (support, acceptance tests, target features)
- Identify key application areas to drive development
- Community software development models
- Funding and organizational models

Achievable Outcomes

- Improve the capability of computational science
- Build and strengthen international collaborations and leadership; deliver more capable, productive HPC systems
- Build and improve R&D program developing new programming models and tools addressing extreme scale
- Open source HPC development guided by roadmap with better coordination and fewer missing components
- Joint programs in education and training for the next generation of computational scientists.
- Vendor engagement and coordination for more capable software supporting exascale science

Workshops and Report

- 3 workshops over the next year
  - 1: Santa Fe, April 7-8
  - 2: Paris France, June 28-29
  - 3: Japan in the early Fall
- Broad engagement by the community
- Initial reports in summer 2009
- Final report for first year at SC09
- Planning for *IMMEDIATE* payoff
  - Could begin ramping up next year

www.exascale.org

Home | Meetings | Documents | Community | Help

Main Page

Page | Discussion | View source | History

Workshop Information

- Workshop Location
- Workshop Agenda (draft)
- Executive Committee
- Organizing Committee
- Background Material

The mission of the **International Exascale Software Project (IESP)** is to lay the foundation for exascale computing by mobilizing the global open source software community to combine and coordinate their collective efforts far more efficiently and effectively than ever before. The IESP will hold a series of three workshops to organize and structure this community wide effort. The first, invitation-only workshop will occur on April 7th and 8th in Sante Fe, New Mexico, US, with people arriving in time for a reception on April 6th. Attendees will include members from industry, academia, and government, with expertise in a range of critical areas.

Goals for the first meeting include the following:

- Assess the short-term, medium-term and long-term needs of applications for peta/exascale systems
- Explore how laboratories, universities, and vendors can work together on coordinated HPC software
- Understand existing R&D plans addressing new programming models and tools addressing extreme scale, multicore, heterogeneity and performance
- Start development of a roadmap for software on extreme-scale systems

Attendance at the workshop is by invitation only. Additional details on registration will be coming soon.

IESP

- *Plan to build an international partnership that joins together* ***industry, the HPC community, and production HPC facilities*** *in a collective effort to design, coordinate, and integrate software for leadership-class machines.*
- Build an international plan for developing the next generation open source software for scientific high-performance computing

Engagement in the Following Activities

- *Build international collaborations in the areas of high-performance computing software and applications.*
- *Development of open source systems software, I/O, data management, visualization, and libraries of all forms targeting tera/peta/exascale computing platforms,*
- *R&D of new programming models and tools addressing extreme scale, multicore, heterogeneity and performance,*
- *Cooperation in large-scale systems deployments for attacking global challenges,*
- *Joint programs in education and training for the next generation of computational scientists.*
- *Vendor engagement to coordinate on how to deal with anticipated scale.*

Goals for this the workshop include

- Assess the short-term, medium-term and long-term needs of applications for peta/exascale systems
- Explore how laboratories, universities, and vendors can work together on coordinated HPC software
- Understand existing R&D plans addressing new programming models and tools addressing extreme scale, multicore, heterogeneity and performance
- Start development of a roadmap for software on extreme-scale systems

Topics

- Purpose of the workshop series, desired outcome (international Research, Development, & Deployment efforts for open source system software and tools for exascale computers)
- Identify key technical areas on which to focus, e.g., file systems, message-passing and multi-threading sw, fundamental numerical sw, system management tools, debuggers, ...
- Begin to identify which groups would like to tackle what areas and which funding sources might support the work
- Begin to develop the open source model, cooperation and collaboration modes, project organization
- Goals for next two workshops, i.e., focus of their agendas

Plan

- Day 1
  - Overviews of architecture trends
  - Current status of HPC systems and SW models
  - Science Drivers in US, EU, and Japan
  - Panel on SW Barriers for HPC, today and tomorrow
    - Three evolutionary SW items
    - Three revolutionary SW items
    - What are the community interaction models to address both evolutionary and revolutionary themes?

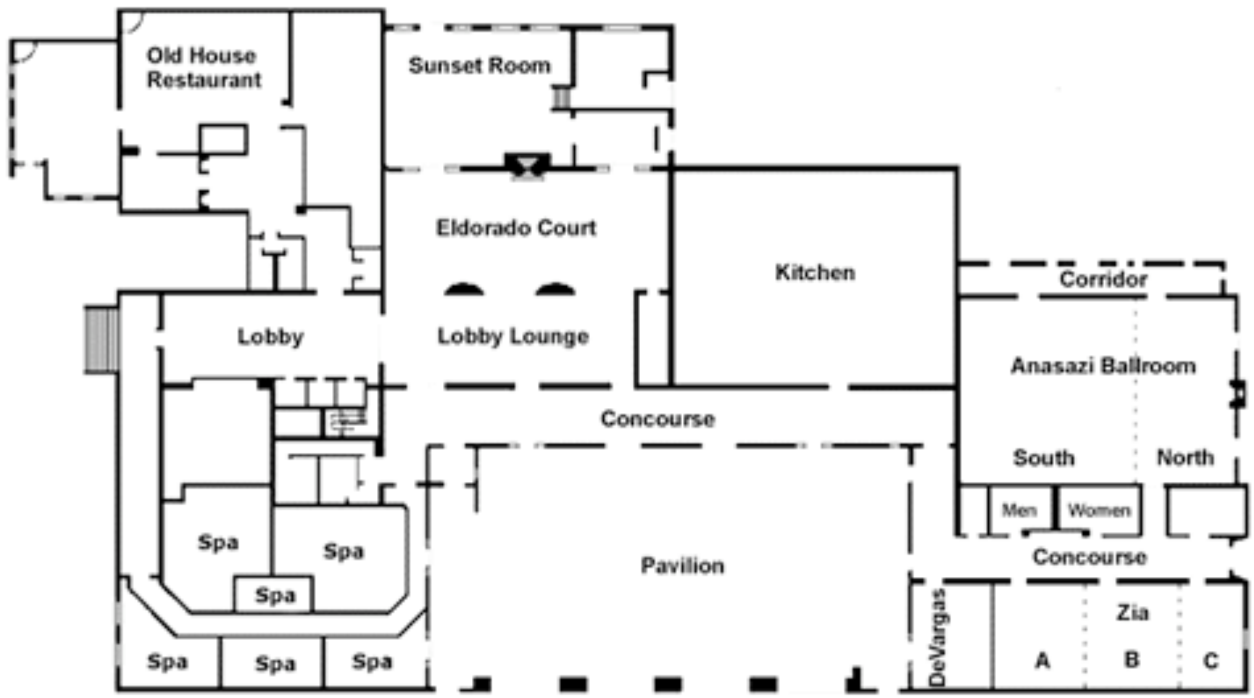
Plan Day 2

- Breakout 1: Technical Roadmap Discussion: What is feasible? What are the top challenges?
- Breakout 2: Collaboration model and funding: How can we work together?

- Goals and agenda for next workshop

Follow on Meetings

- Refine the ideas that emerged from the earlier meetings.
- Incorporate new ideas into the plan.
- Expose the IESP to a wider group of people.
- We would like to get buy in from as many people as possible. Some may not be able to attend the earlier meetings.


Old House Restaurant
Sunset Room
Eldorado Court
Kitchen
Corridor
Lobby
Lobby Lounge
Anasazi Ballroom
Concourse
South
North
Men
Women
Spa
Spa
Pavilion
Concourse
Spa
Zia
DeVargas
A
B
C
Spa
Spa
Spa

e-Infrastructure in FP7:
HPC related aspects

Mme Catherine Riviére
(on behalf of DG INFSO/F03)

IESP Workshop
Santa Fe, 7-8 April 2009

European Commission
Information Society and Media

e-infrastructure

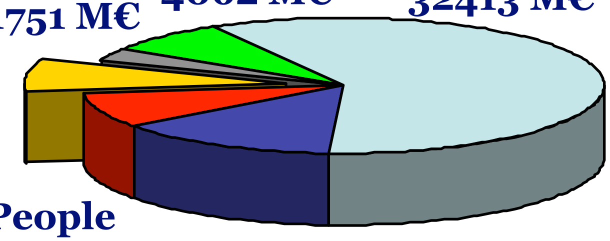
Main contents:

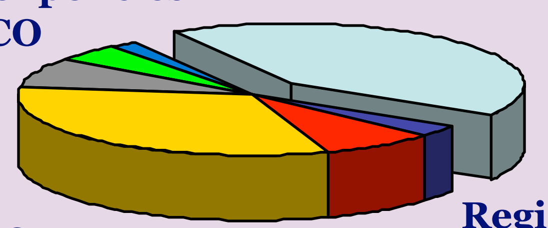
- e-Infrastructure: the mission
- Framework Programme 7
- Main flagship projects
  - GÉANT
  - EGEE
  - DEISA & PRACE
  - … and scientific data repositories
- FP7 ‘Capacities’: RI Call 7 topics

e-Infrastructure: the mission!

e-Infrastructure refers to the creation of a new research environment in which all European researchers have shared access to unique or distributed scientific facilities (including data, instruments, computing and communications), regardless of their type and location in the world.

Framework Programme 7: 2007 to 2013

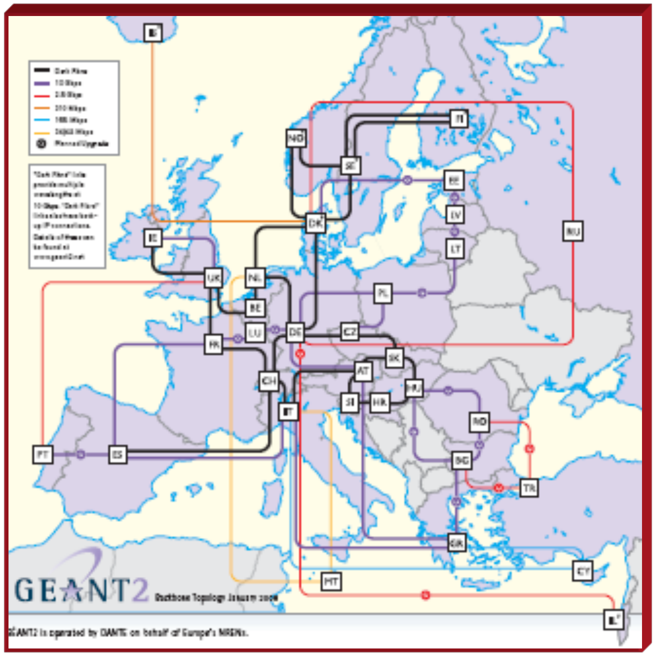




~55B€

GÉANT: connecting Europe

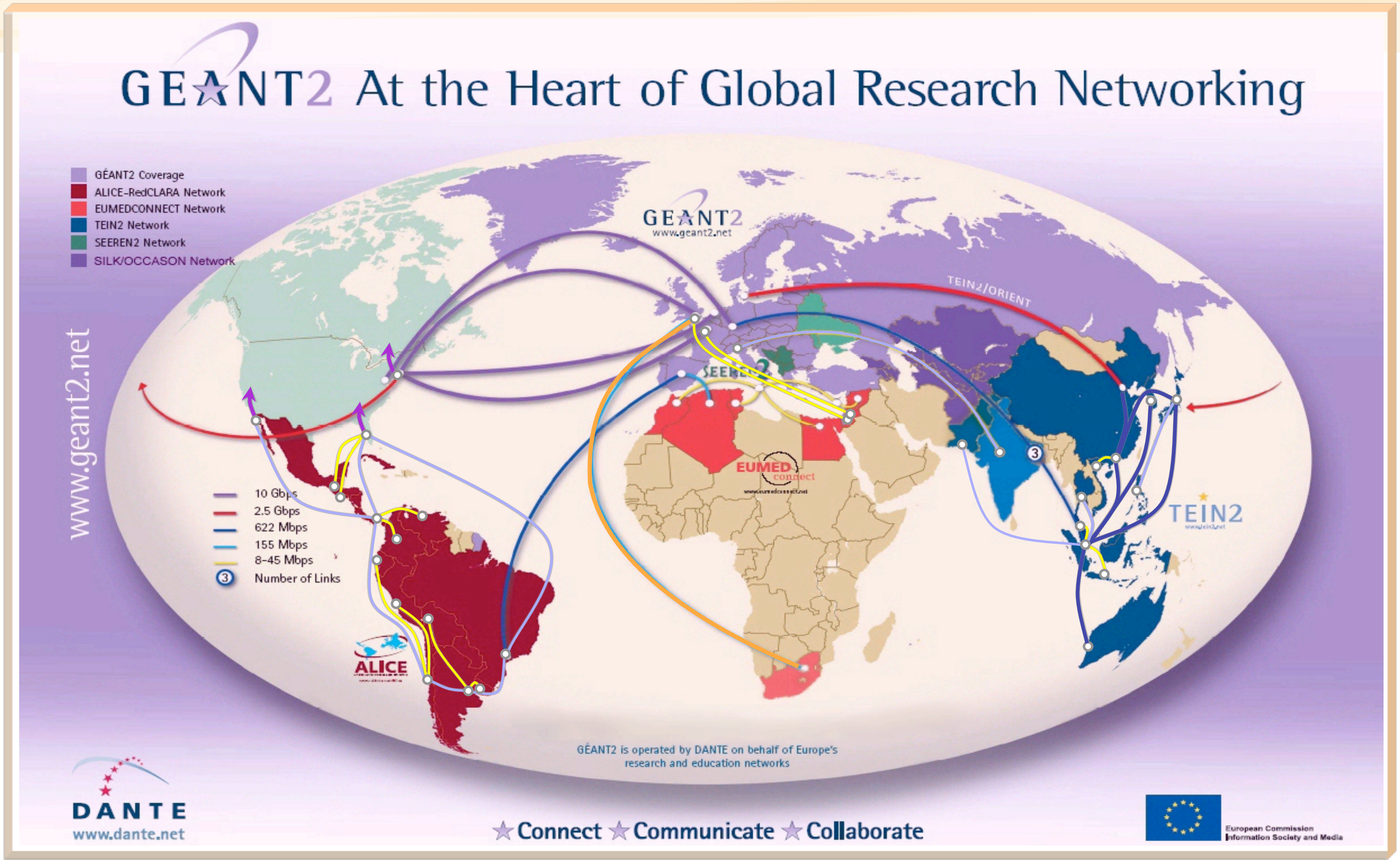
- Pan-European coverage
  (40+ countries /3900 universities / 30+ million students)
- Hybrid architecture:
  - connectivity at 10 Gb/s
    (aggregated traffic)
  - dark fiber wavelengths
    (demanding communities)



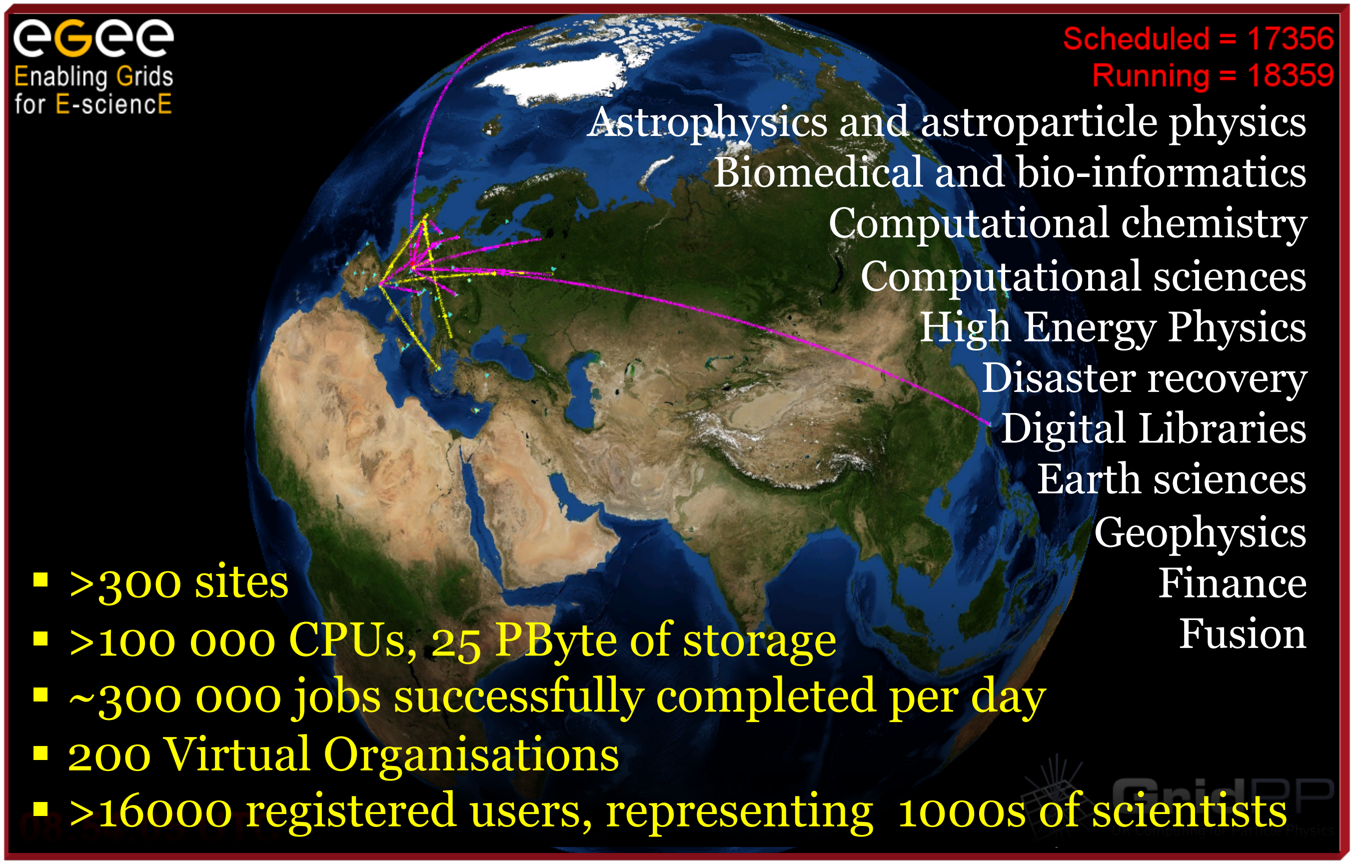
European Commission
Information Society and Media

e-infrastructure

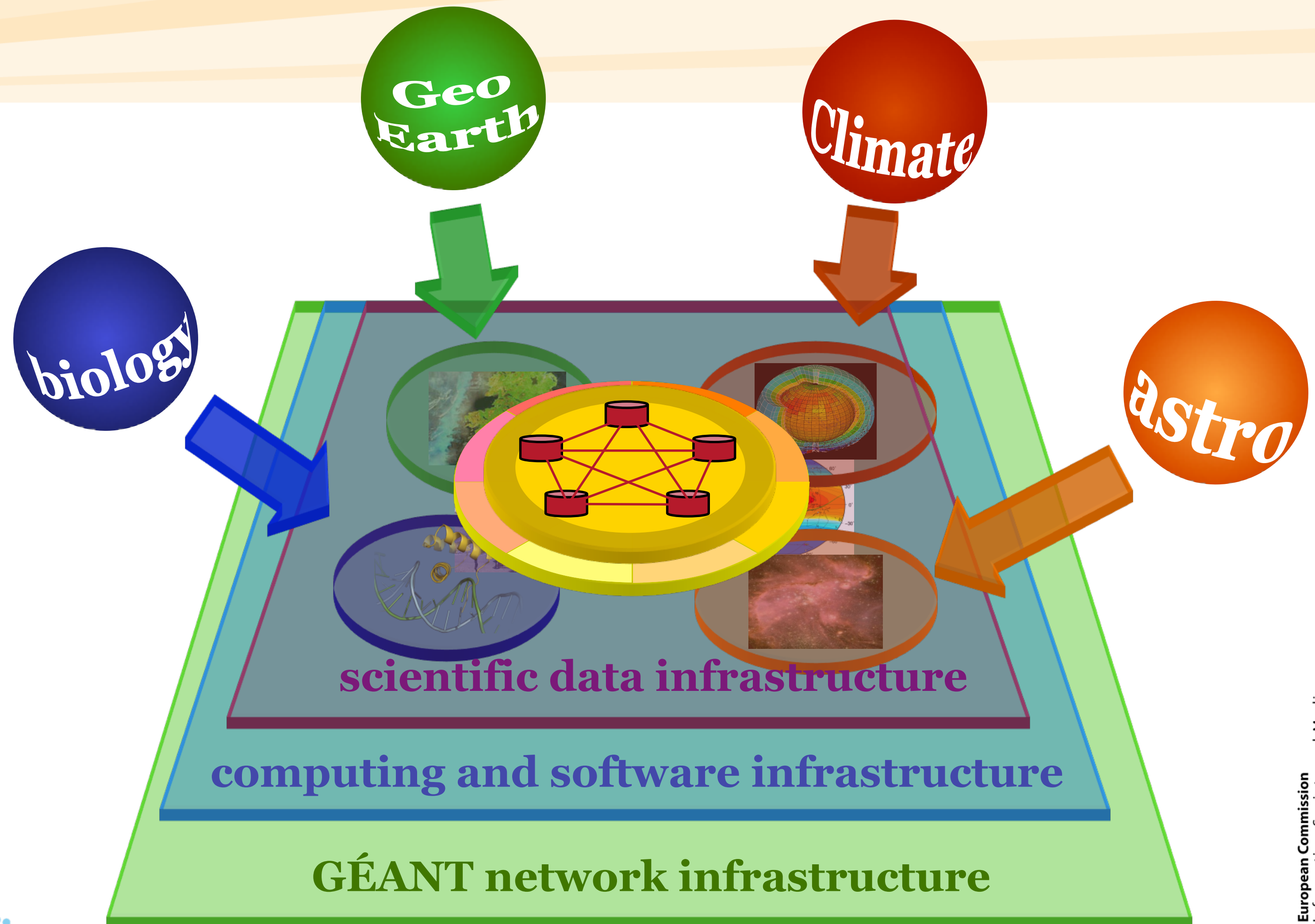
Global dimension of GÉANT



EGEE: Tackling Global Challenges



Scientific Data Infrastructure



e-infrastructure

European Commission
Information Society and Media

DEISA: 'virtual' HPC services

Distributed
European
Infrastructure for
Supercomputing
Applications

- 12 sites in 7 countries connected at 10 Gb/s
- Over 22,000 CPUs with an aggregated peak performance of close to 1 Peta flops
- Running larger parallel applications in individual sites
- Enabling workflow applications with grid technologies (UNICORE)
- Providing a global data management service
- Extreme Computing Initiative

PRACE: the preparatory phase

18 European countries signed the PRACE MoU !!

e-infrastructure

Image courtesy of the PRACE partnership

European Commission
Information Society and Media

Draft WP2010 topics:
RI Call 7: Open 30.07.09; close 24.11.09

Commission proposal

- **INFRA-2010-1.2.1: Distributed computing infrastructure (DCI)**
- **INFRA-2010-1.2.2: Simulation software and services**
- **INFRA-2010-1.2.3: Virtual Research Communities**
- **INFRA-2010-2.3.1: First implementation phase of the European HPC service**
- **INFRA-2010-3.3: Coordination actions, conferences and studies supporting policy development, incl. international cooperation**

TOTAL Indicative budget: 115 Million Euro

further information

www.cordis.europa.eu/fp7/ict/e-infrastructure/

Konstantinos.Glinos@ec.europa.eu

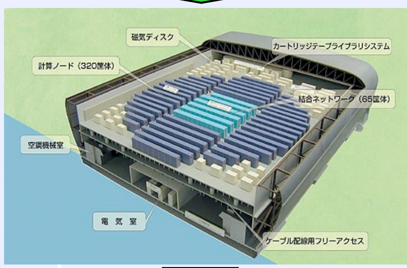
Contents

- simulations for predictions (example)
- science and technology policy in Japan
- project of the next generation supercomputer
- grand challenges in applications
- collaborations with private sectors
- concluding remarks

Contribution to the IPCC by the Earth Simulator

Global warming projections
by climate modeling groups
<under the MEXT* research project>

Earth Simulator



Some of major outcomes

- **Highest resolution coupled model**
  → ***"Very likely" Attribution (stronger confirmation)***
- **Super-high resolution Global Atmospheric model** → ***Projection of increased strength of Typhoons & Hurricanes (new finding)***
- ***Earth system model***
  → ***Carbon cycle feedback causing additional warming (new finding)***

Nobel Peace Prize

IPCC

Fourth Assessment Report (2007)

Synthesis Rep.

Working Group Ⅲ
(Mitigation)

Working Group Ⅱ
(IAV)*

Working Group Ⅰ
(Physical Science Basis)

Sound Scientific Basis for:
Bali Roadmap
(Climate Change Conference in 2007)

(IAV = Impact, Adaptation and Vulnerability)*

Outline of the 3rd S&T Basic Plan

1. Fundamental Concept

- Recent situation revolving around S&T
- Basic stance toward the 3rd plan
- Fundamental ideas and policy goals
- Total gov'tal R&D investment: \25 trillion ($200 billion)

2. Strategic Priority Setting in S&T

- Promotion of basic researches
- Prioritization of R&D for policy-oriented subjects
  Primary prioritized areas; Life science, IT, Environmental sciences, Nano-tech. & materials
  Secondary prioritized areas; Energy, MONODZUKURI tech., Infrastructure, Frontier (outer space & oceans)
- Promotion strategy for the prioritized areas

3. S&T system reforms

- Fostering S&T personnel and providing opportunities
- Progress in science and leading to innovation
- **Upgrading infrastructures for S&T promotion**
- Strategic commitment on international S&T activities

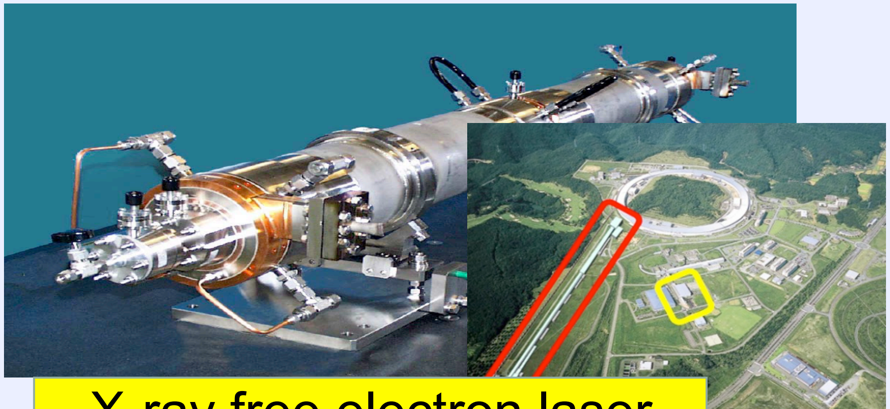
4. Public Confidence and Engagement

- Responsible actions regarding ethical, legal and social issues
- Reinforcement of accountability and public relations of S&T activities
- Promotion of public understanding of S&T
- Facilitation of public engagement with S&T-related issues

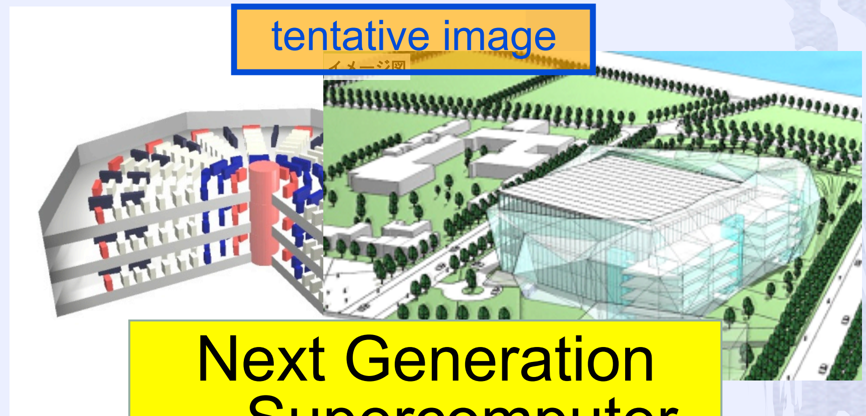
5. Missions of the CSTP

- More efficient and effective management of governmental R&D
- Break of institutional or operational bottle necks
- Follow-up of the Plan and promotion of progress in S&T

Key Technologies of National Importance



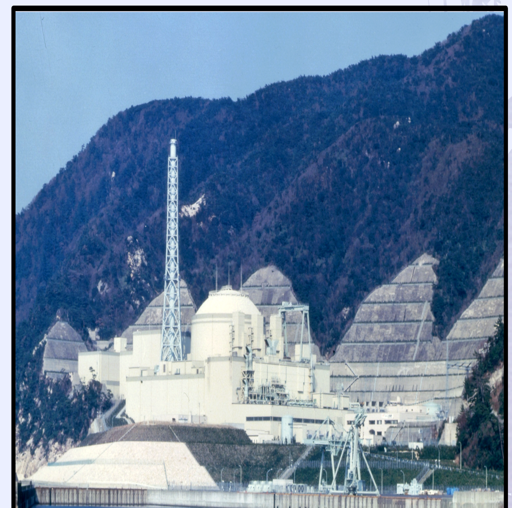
X-ray free electron laser



Next Generation Supercomputer

Space transport system

Ocean & earth exploration system



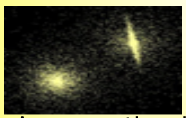
Fast breeder reactor technology

: projects RIKEN is conducting

Six Goals of the Third Science and Technology Basic Plan (FY2006-FY2010)

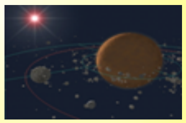
<Goal 1>
Discovery & Creation of Knowledge toward the future

Milky Way formation process



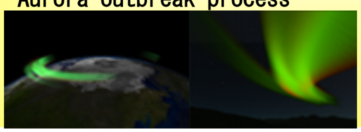
by RIKEN

Planet formation process



by RIKEN

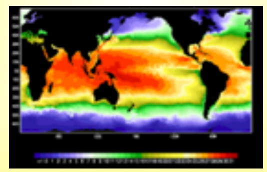
Aurora outbreak process



by JAMSTEC

< Goal 3 >
Sustainable Development
- Consistent with Economy and Environment -

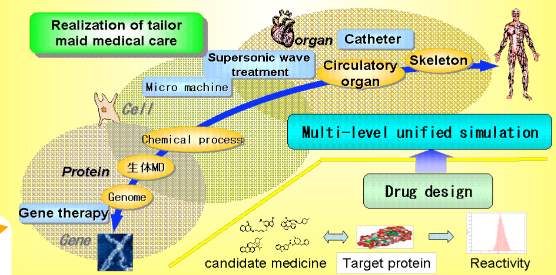
An influence prediction of El Nino phenomenon



by JAMSTEC

< Goal 5 >
Good Health over Lifetime

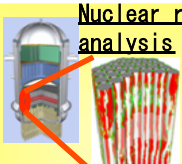
Multi-level unified simulation



by Univ. of Tokyo and RIKEN

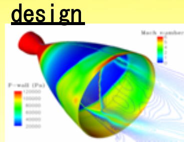
Development and Application of Advanced High-performance Supercomputer

Nuclear reactor analysis



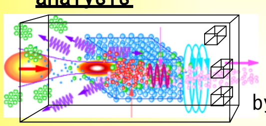
by JAEA

Rocket engine design



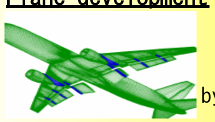
by JAXA

Laser reaction analysis



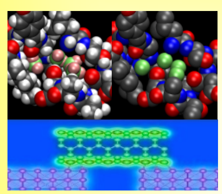
by JAEA

Plane development



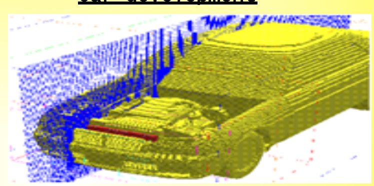
by JAXA

Nano technology



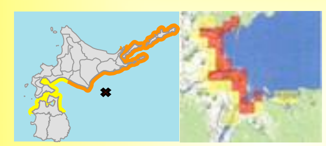
by IMS

Car development



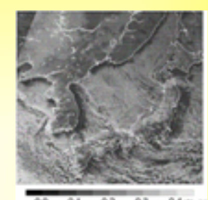
by NISSAN

Tsunami damage prediction



by Tohoku Univ.

Clouds analysis



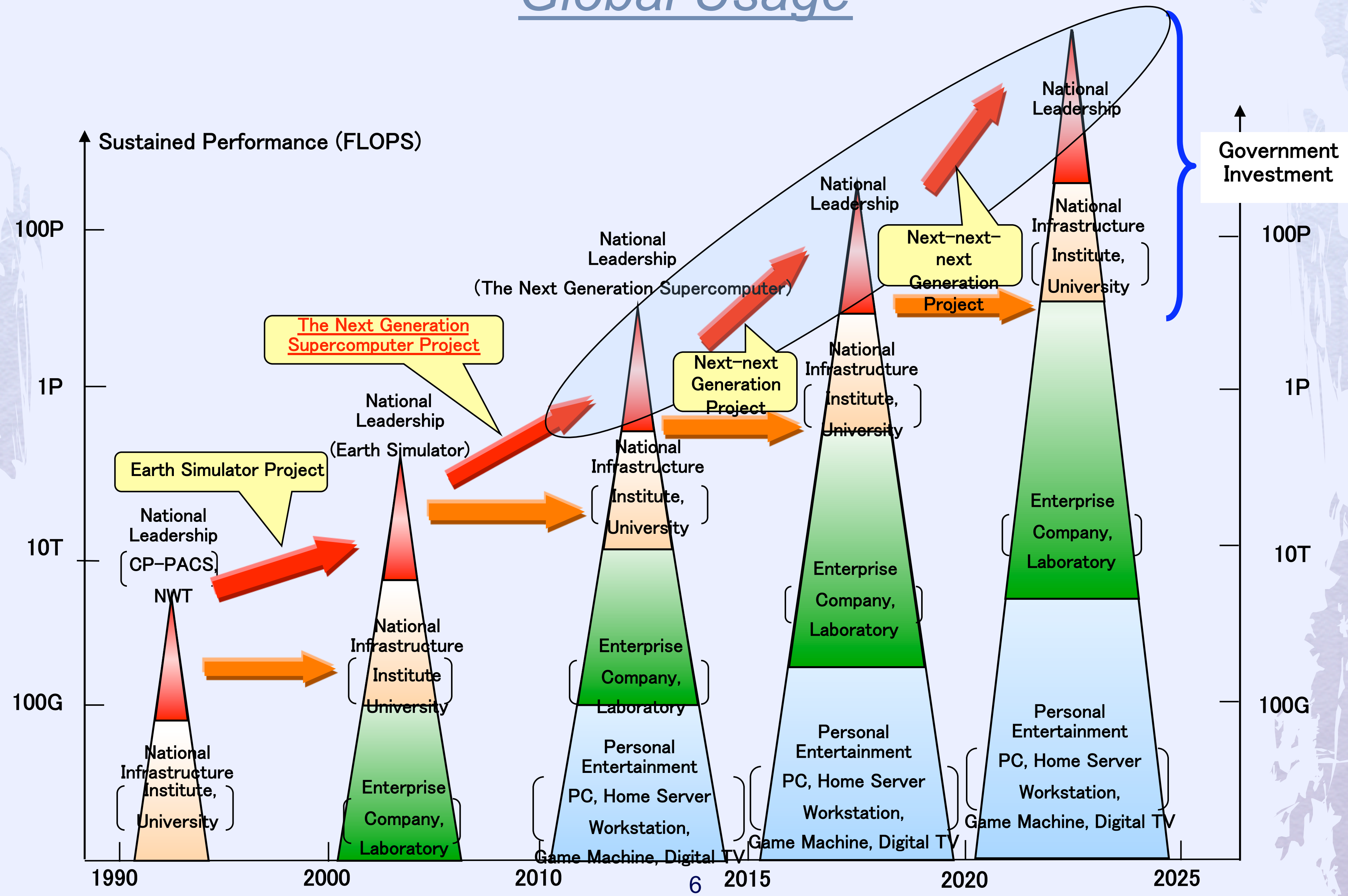
by MRI

< Goal 2 >
Breakthroughs in Advanced Science and Technology

< Goal 4 >
Innovator Japan
- Strength in Economy & Industry -

< Goal 6 >
Safe and secure Nation

Expansion of Highest Computer for Global Usage

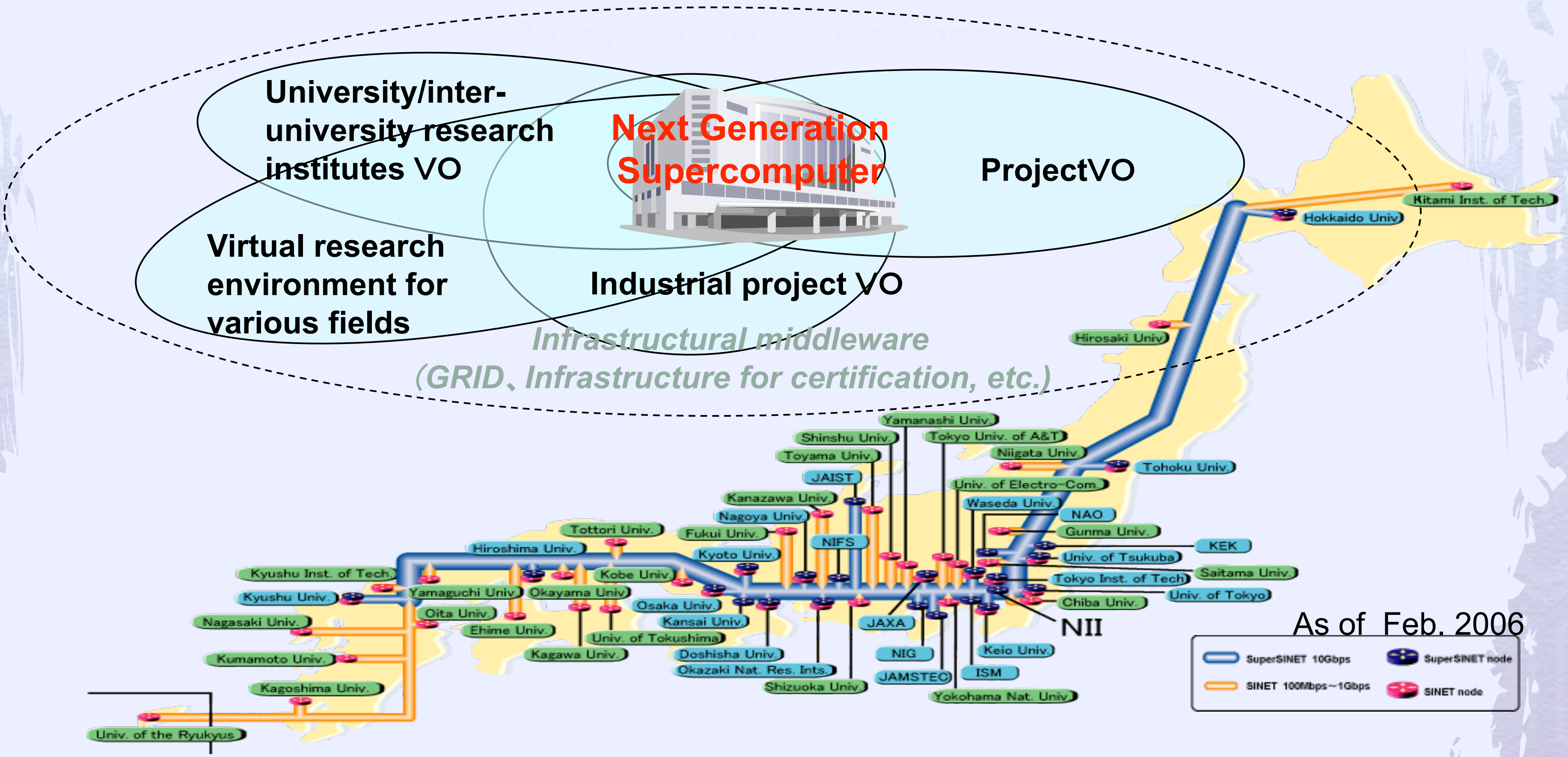


CACST: Center for Advanced Computational Science and Technology (tentative name)

- Computer science and Computational science
- Both researchers will gather and expect to develop new research fields and methodologies
- Currently, we are designing the center and operation policy of the supercomputer
  - The users will be chosen by a new committee independent from RIKEN to pick up valuable subjects

The Location of the Next Generation Supercomputer Center

Relations with Other Supercomputer Centers



Cyber Science Infrastructure Plan

Proposed by National Institute of Informatics (NII)

(Note)V O :Virtual Organization

Development & Application of Next-Generation Supercomputer Project by MEXT

FY2006: 3,547Million yen / FY2007: *7,736*Million yen	
FY2006～FY2012 (total budget expected）about 110billion yen	

1. Purpose of policy

Development and implementation of the world's most advanced and high-performance Next-Generation Supercomputer, and to develop and disseminate its usage technologies, as one of Japan's "Key Technologies of National Importance" (National Infrastructure).

2. Expected effects

As an important tool for simulation, supercomputing needs to be developed further. This project aims to bring the Next-Generation Supercomputer to completion in 2012.

In order to maintain world-leading position in variety of areas, the following academic-industrial collaboration activities will be conducted under the initiative of MEXT.

(1) Development and implementation of the world's most advanced high-performance Next-Generation supercomputer

(2) Development and dissemination of software that makes optimum use of the supercomputer

(3) Establishment of the world's most advanced and highest standard supercomputing Center of Excellence, which includes the Next-Generation Supercomputer

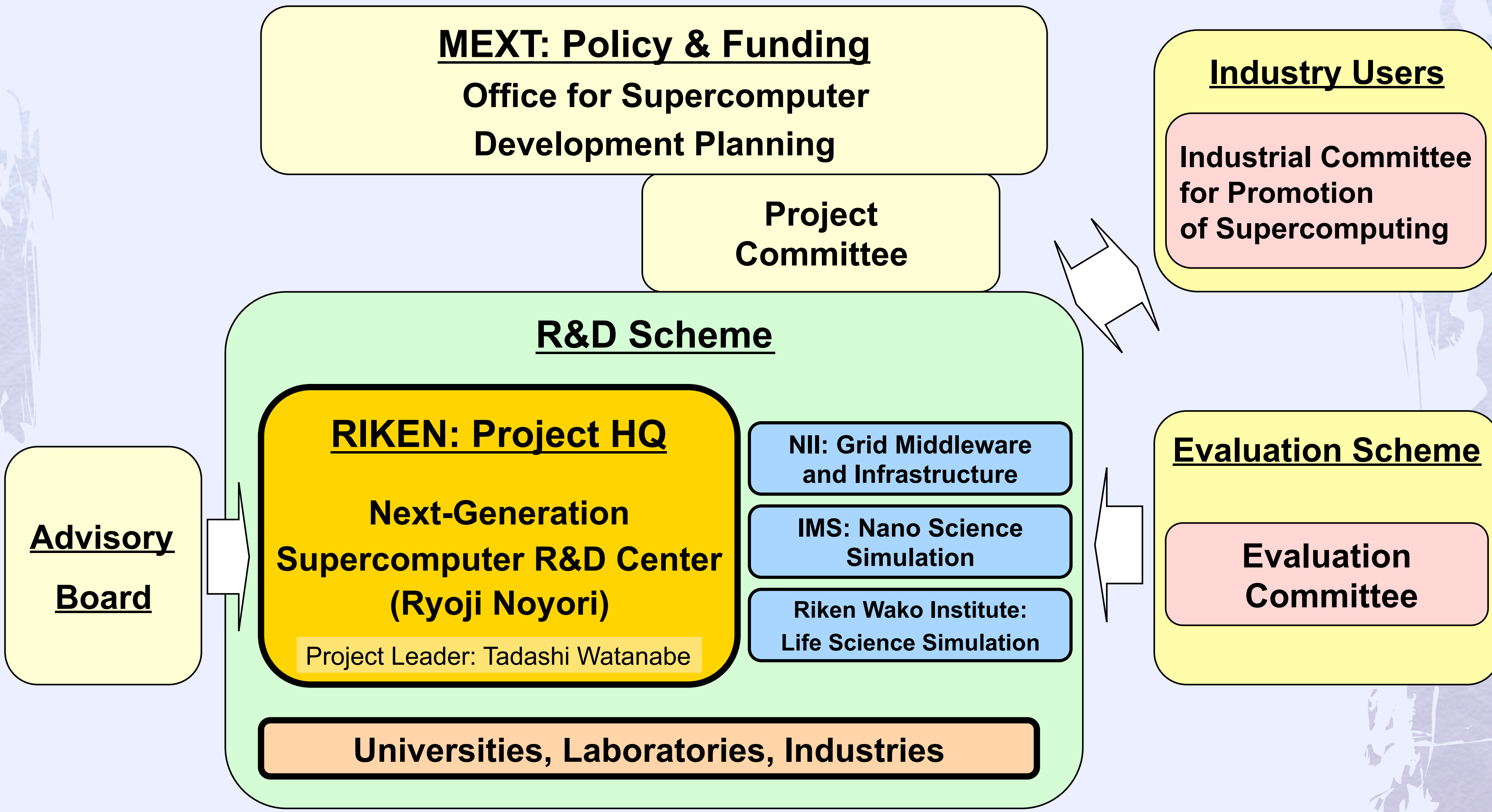
3. Project Framework

- Integrated development of computer and software
- Establishment of nationwide academic-industrial collaborative structure, with RIKEN as the project headquarters
- A new law has been introduced for the framework of usage and administration

Goals of the Next Generation Supercomputer Project

1. Development and installation of the most advanced high performance supercomputer system
2. Development and wide use of application software to utilize the supercomputer to the maximum extent
3. Provision of flexible computing environment by sharing the next generation supercomputer through connection with other supercomputers located at universities and research institutes
4. Establishment of “Advanced Computational Science and Technology Center (tentative name)”

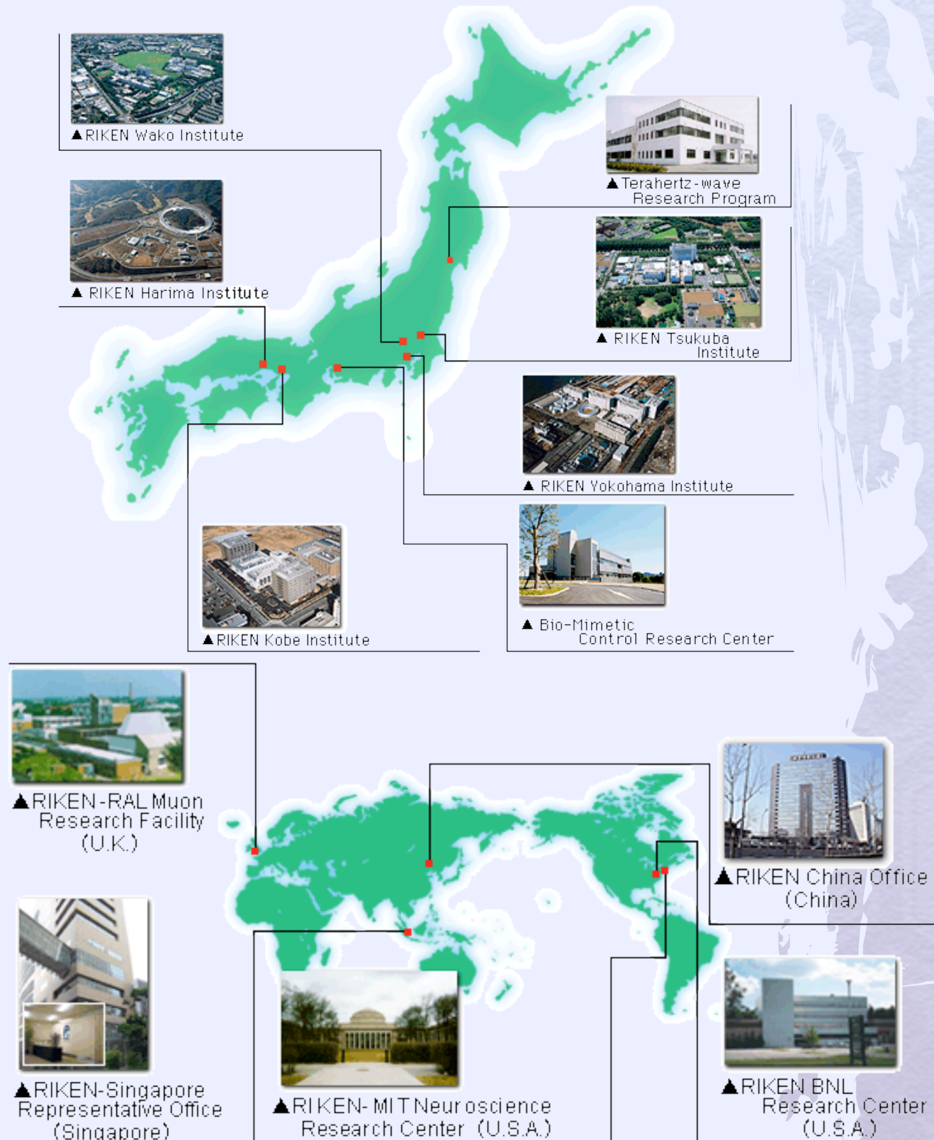
Project Organizations



(Note) NII: National Institute of Informatics, IMS: Institute for Molecular Science

RIKEN and Advanced Center for Computing and Communication

- RIKEN
  - comprehensive research in science and technology (excluding only humanities and social sciences)
  - physics, chemistry, medical science, biology, and engineering extending from basic research to practical application
  - 7 campus in Japan, 5 outside Japan
  - about 3000 researchers
  - an Independent Administrative Institution under the Ministry of Education, Culture, Sports, Science & Technology (MEXT) from 2003
- Advance Center for Computing & Communication
  - Providing RIKEN researchers with computer resources and network services
  - Operating RSCC(RIKEN Super Combined Cluster)



Policy and Outline of A Next Generation Supercomputer Project

Purpose of policy:

development, installation and application of an advanced high performance supercomputer system, as one of Japan's "Key Technologies of National Importance"

Total Budget:

about 115 billion Yen (0.7 billion Euros)

100% national funds

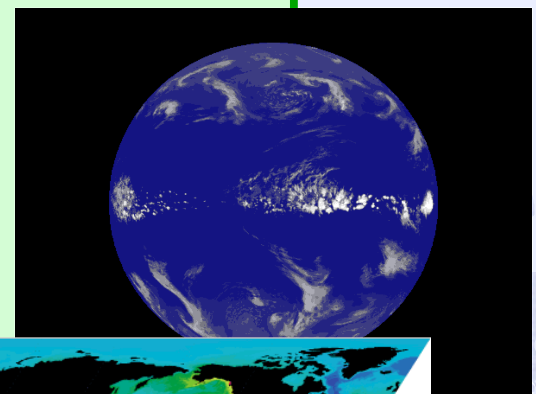
Period of Project:

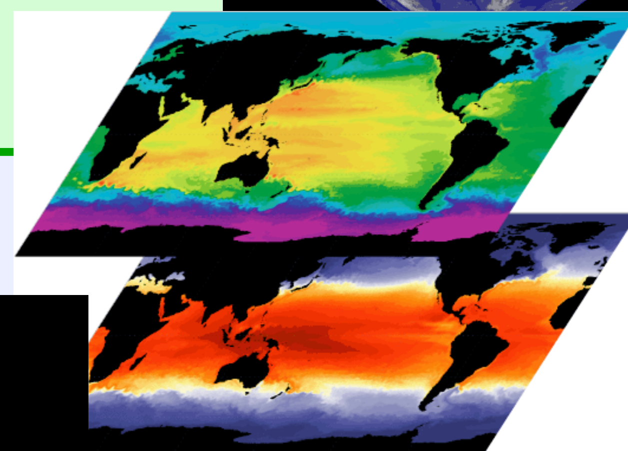
FY2006 – FY2012

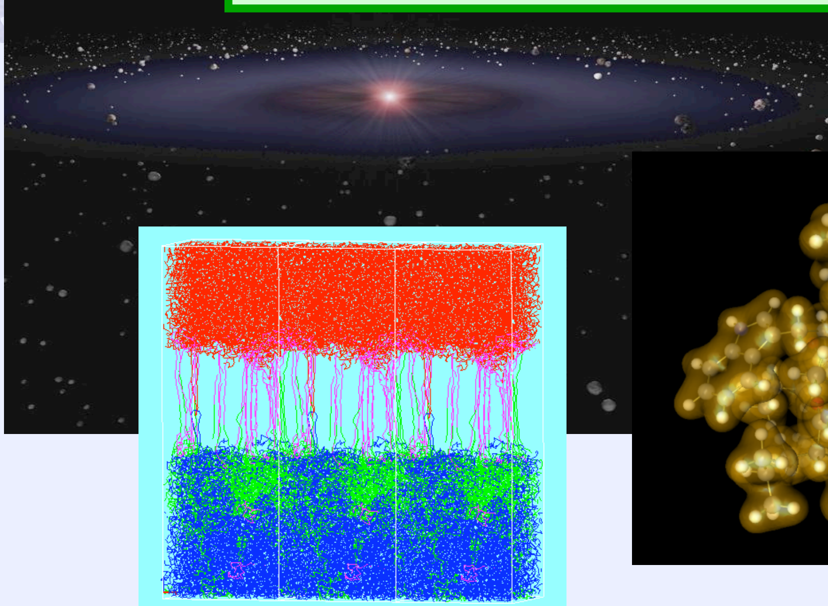
Applications Selected for Basic Designs of Hardware

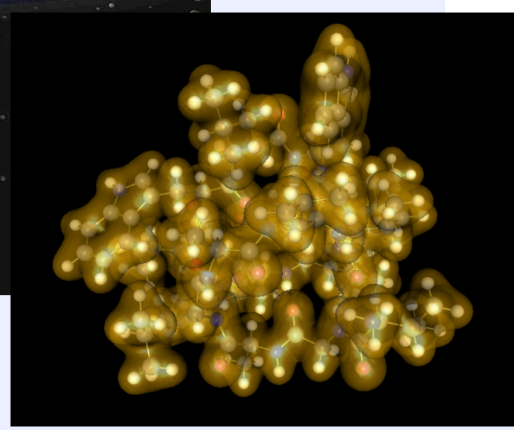
21 application codes have been selected for basic designs of hardware:

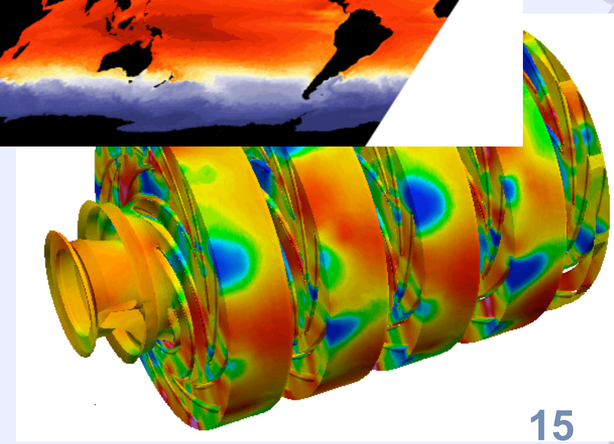
- **6 from nano sciences**
- **6 from life sciences**
- **3 from environment and disaster protection**
- **4 from engineering**
- **2 from physics and astronomy**









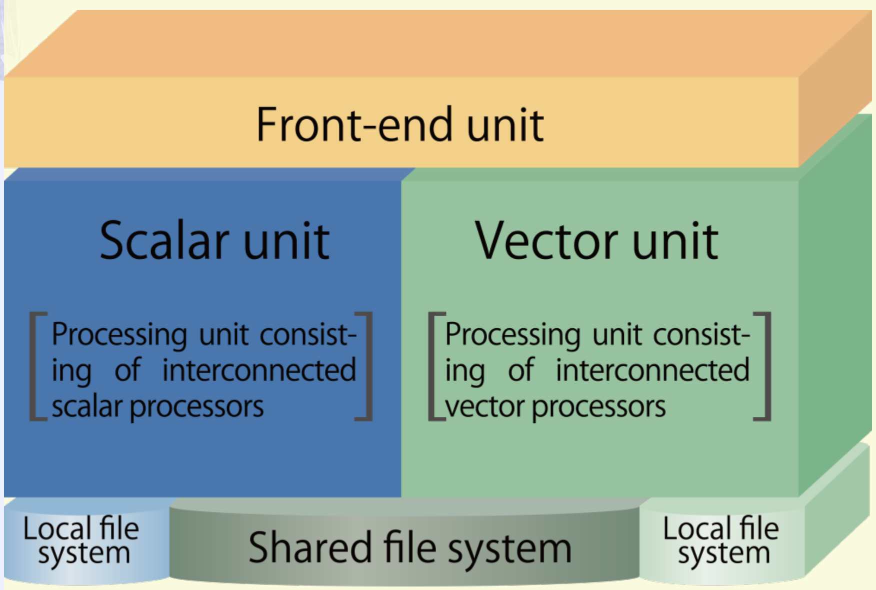


The Next-Generation Supercomputer project

The Next-Generation Supercomputer project started in 2006 which is being carried out by RIKEN, with partners in industry, universities, and the government, under an initiative by MEXT (the Ministry of Education, Culture, Sports, Science and Technology).

Due to be ready in 2012, the peta-scale computing by the new supercomputer will ensure that Japan continues to lead the world in science and technology, academic research, industry, and medicine.

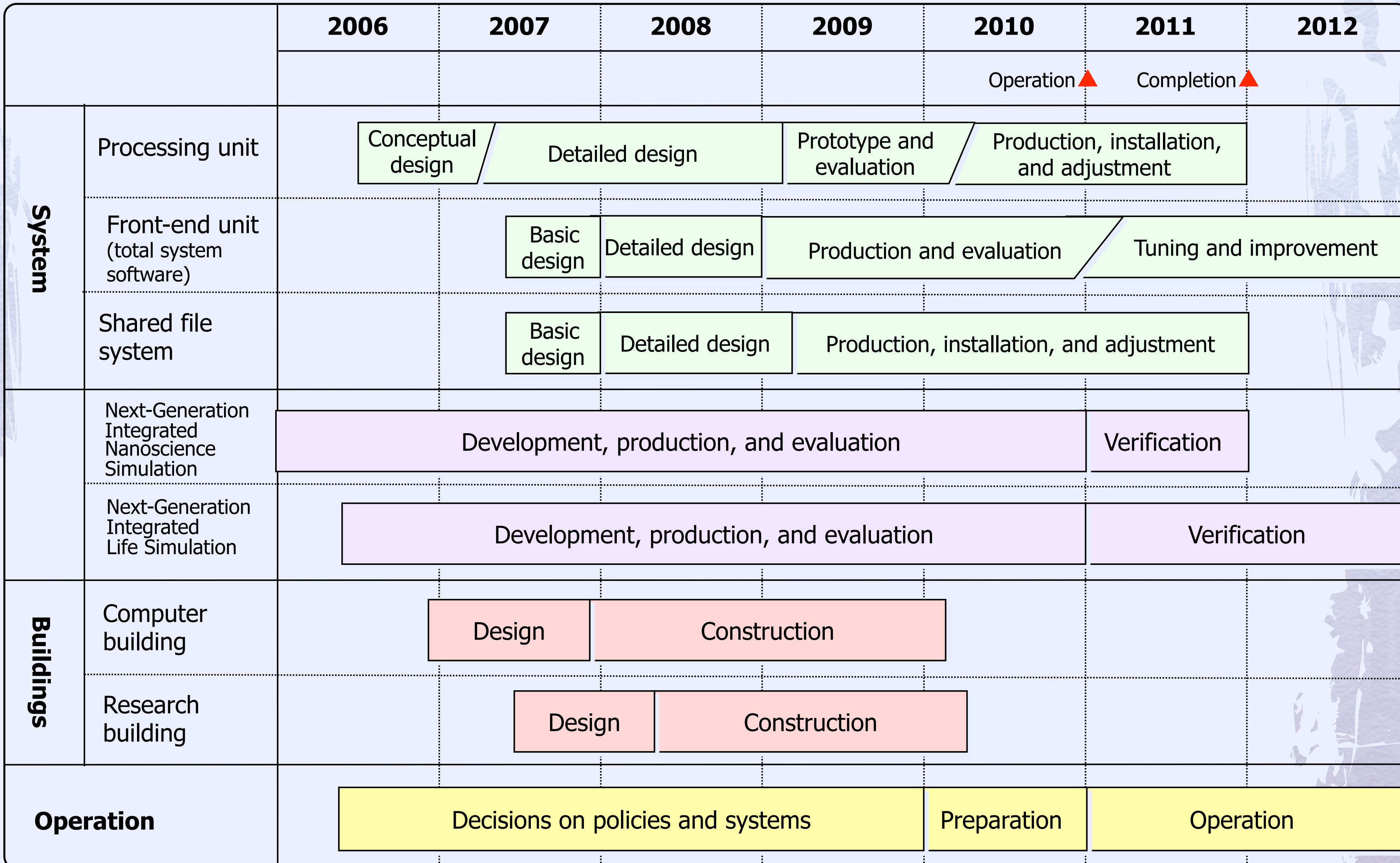
[System configuration]



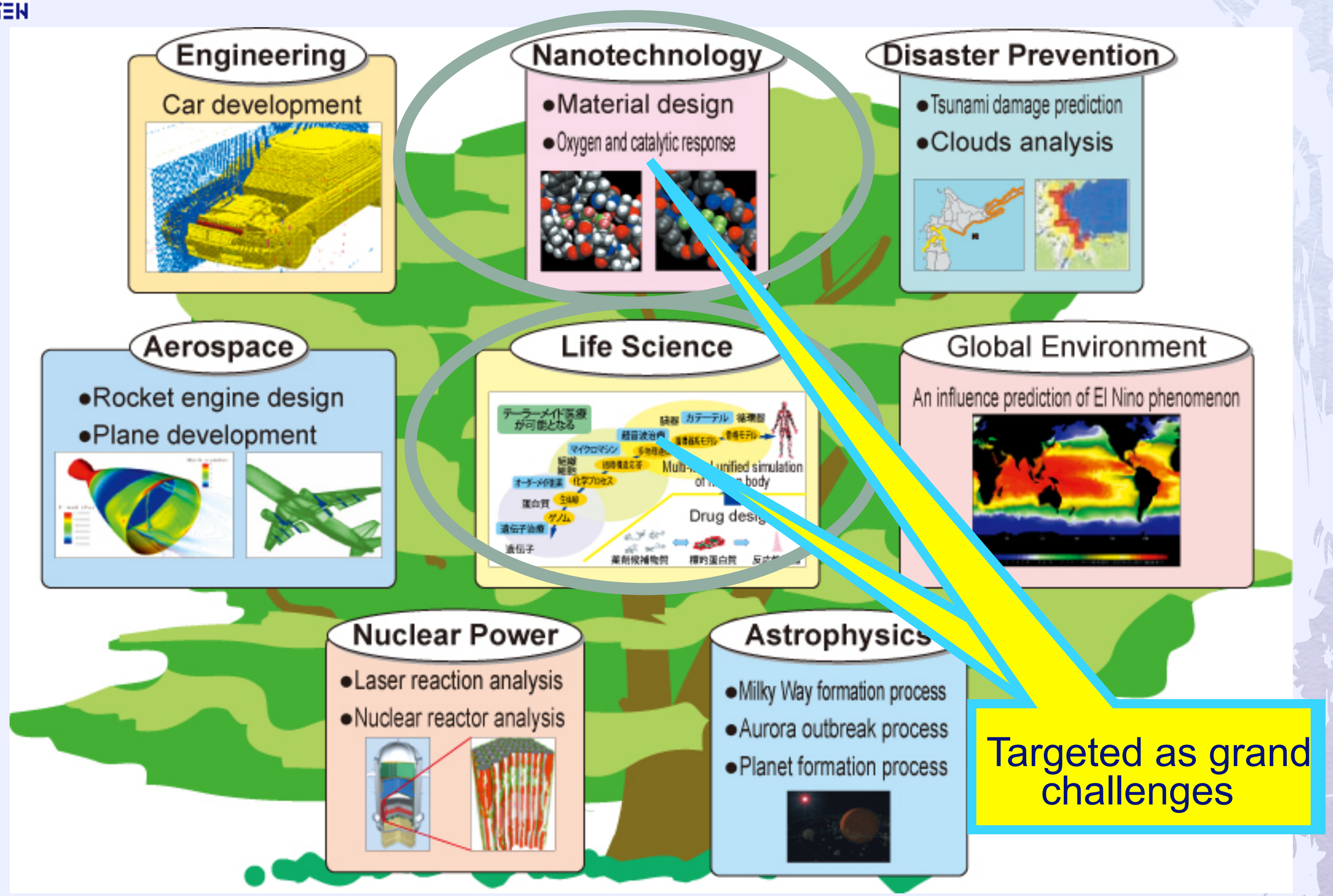
The Next-Generation Supercomputer will be hybrid general-purpose supercomputer that provides the optimum computing environment for a wide range of simulations.

- Calculations will be performed in processing units that are suitable for the particular simulation.
- Parallel processing in a hybrid configuration of scalar and vector units will make larger and more complex simulations possible.

Schedule of Project



Major Applications of Next Generation Supercomputer



Task Forces to Develop the Grand Challenges Application Codes

Nano Science

Conducting Institute: Institute for Molecular Science (IMS)

Budget for 2008 Fiscal Year: 5.6 Million US Dollars

Contributing Institutes and Universities: 6

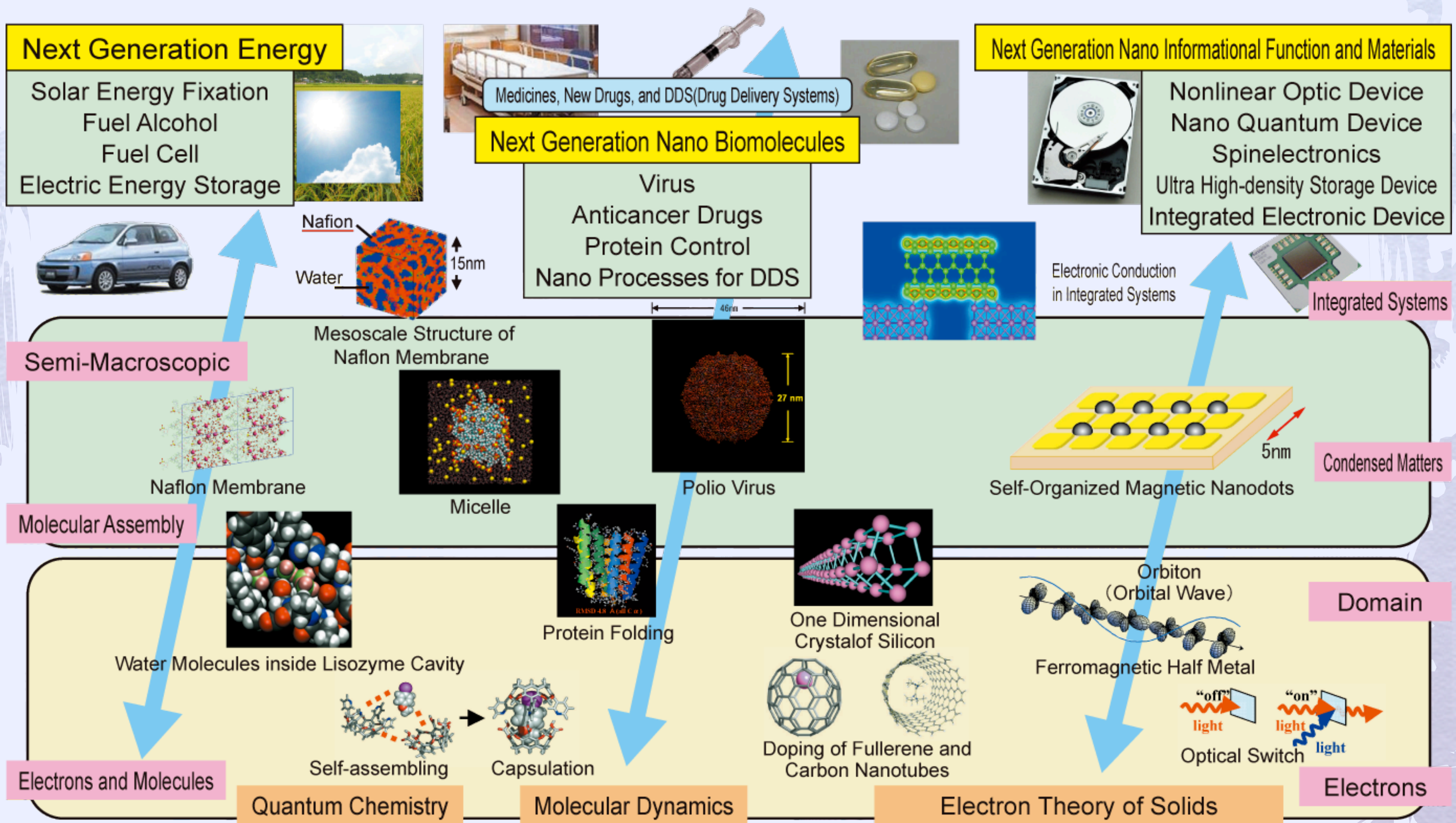
Life Science

Conducting Institute: RIKEN

Budget for 2008 Fiscal Year: 14.4 Million US Dollars

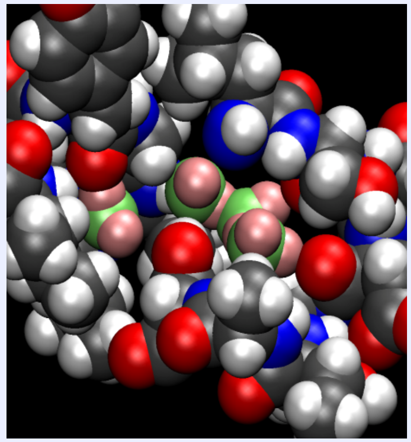
Contributing Institutes and Universities: 14

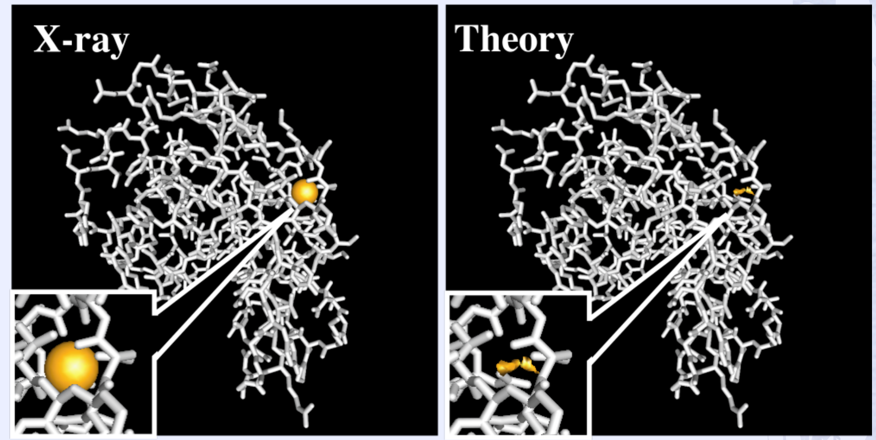
Basic Concept for Simulations in Nano-Science



Molecular Recognition of Proteins Reproduced by 3D-RISM

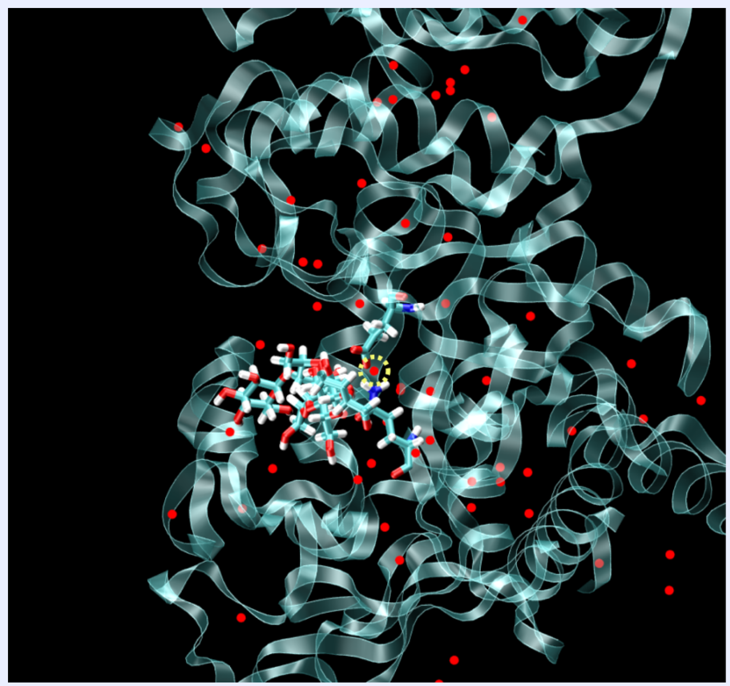
Water molecules and Ions recognized by a protein

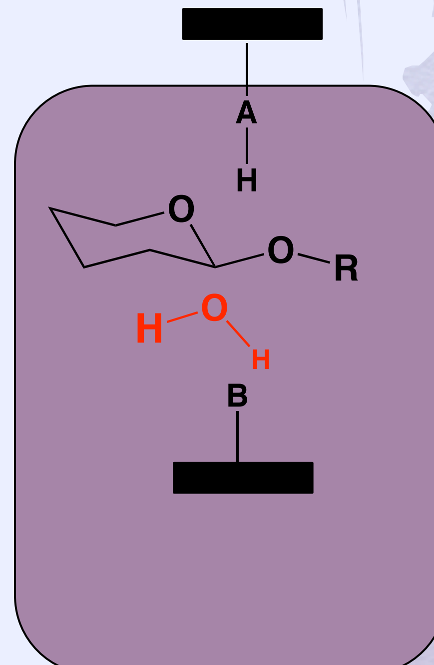




Cellulase

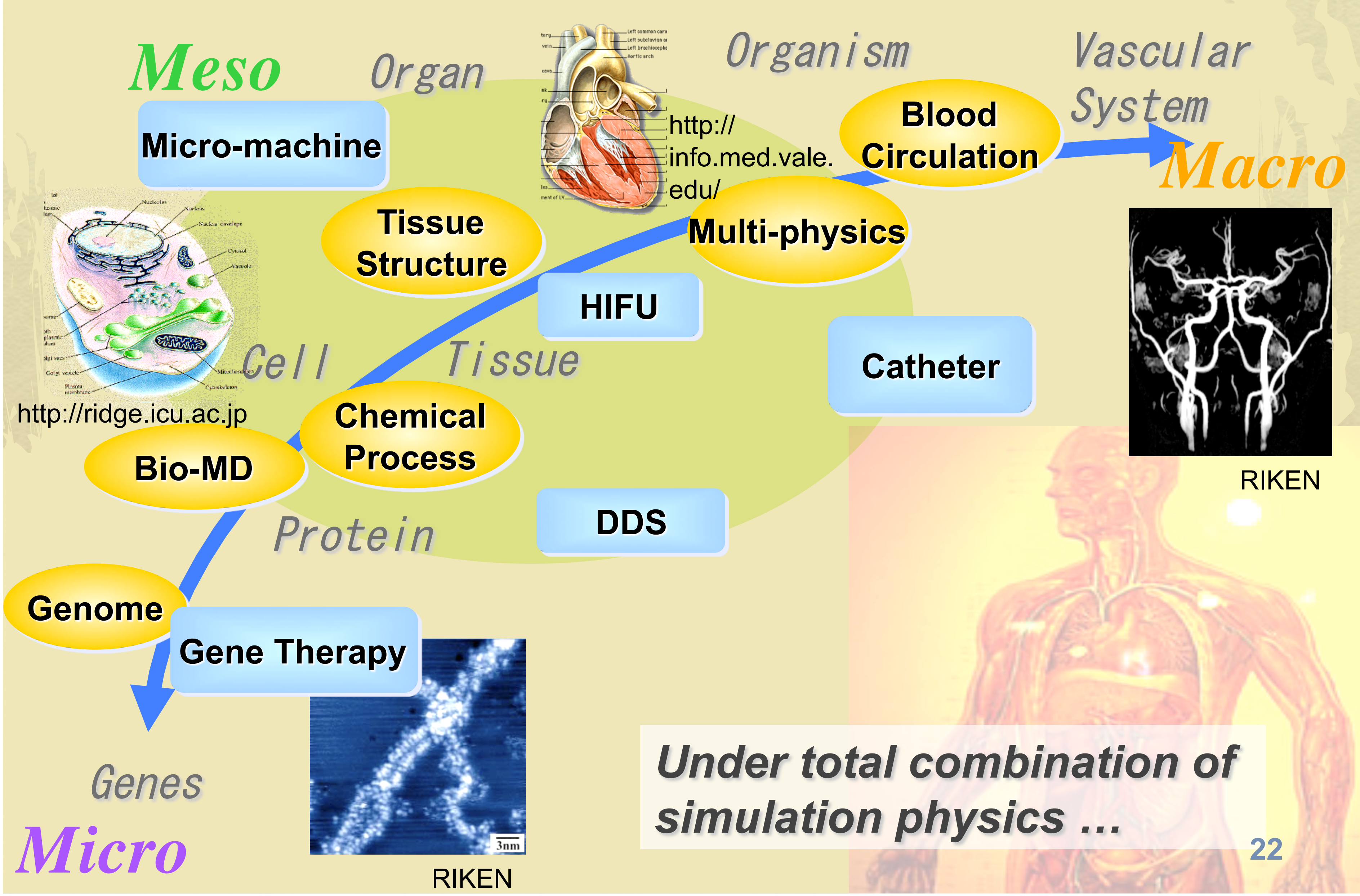
A water molecule recognized by enzyme-cellulose complex.



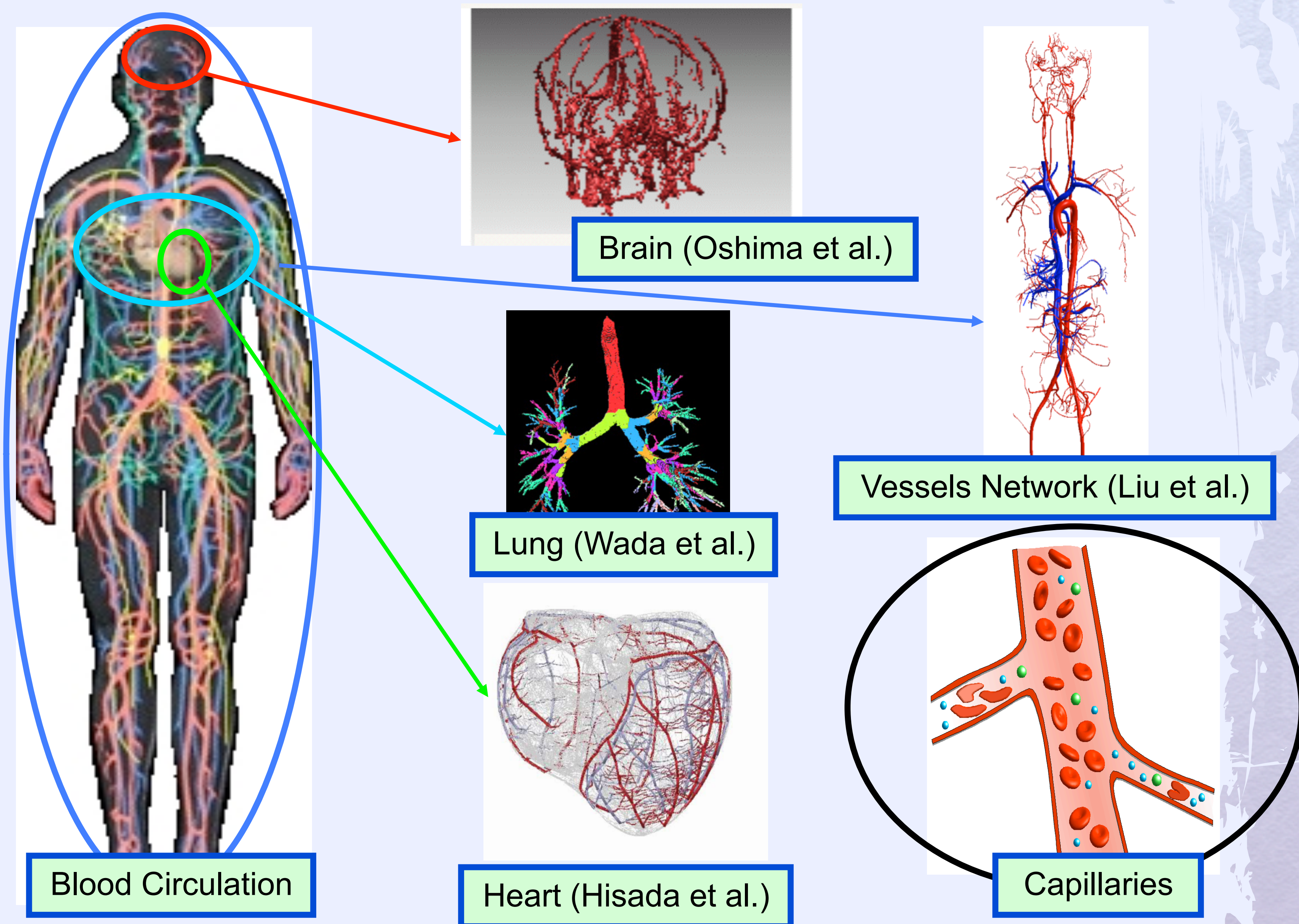


Courtesy of IMS

Basic Concept for Simulations in Life Sciences



Simulation for Circulation System



Promotion Program of Supercomputers for Industries

Industrial Committee for Super Computing Promotion

- established in 2006
- participated by more than 170 companies from various industries
- activities: simulation of engines, analyses of car body, material and polymer simulation, weather simulation

Recent Activities of ICSCP

seminar and expo to industrial researchers about usefulness of simulations

6 seminars a year for member industries

Promotive Activities by Public Computer Centers for Industrial Use of HPC

MEXT is conducting a project to stimulate use of public high-tech facilities for industrial R&D

Earth Simulator and computer centers of major universities provide their computer resources for industries

test use for free and productive use for charged

Computer Center of Tokyo University

Fields of about 40 applicants from industries

- drug design
- semi conductors
- aerodynamics
- functional materials
- banking system
- fuel cell
- noise control of cars and bullet train
- catalyst
- internet search engine
- audio interpretation

Concluding Remarks

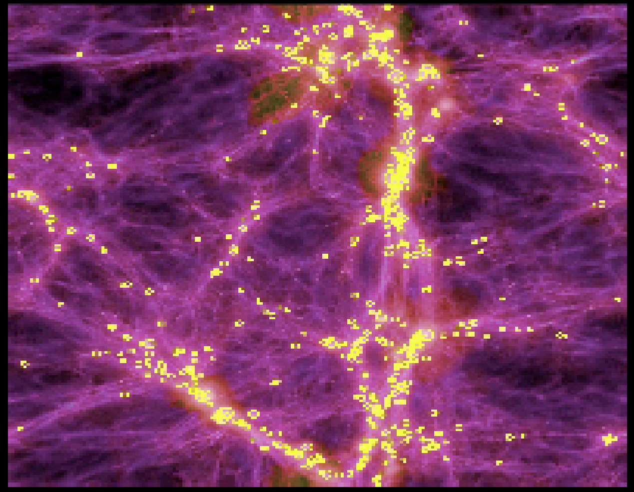
- The next generation supercomputer project aims at:
  - to keep cutting-edge computer technologies inside Japan
  - to prompt application software developing activities
  - to rear young scientists for HPC fields
- Consequently, we expect:
  - to maintain competitiveness in worldwide HPC technology
  - to innovate R&D in industries by computational science
  - to create new IT businesses such as SaaS (service as a software)
- Then, we accomplish:
  - to reinforce science infrastructure in Japan
  - to retain high economic activities

International Exascale Software Program

Abani K. Patra
Office of Cyberinfrastructure,
National Science Foundation

Drivers

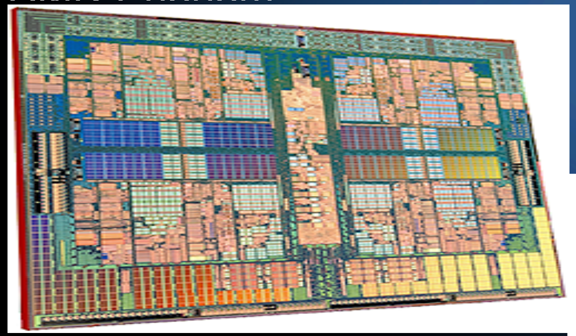
- Advances in most branches of science and engineering are critically dependent on increasingly complex multi-scale, multi-physics, data driven computations and analysis.
- Complexity of Systems
  - Moore’s Law and Beyond -- Multicore, manycore, ...
  - Heterogeneous machines
  - Data Intensive Scalable Computing
  - Workflows, Grids, Clouds ...
- All this complexity dealt with by software and tools!
- Support for which is ad hoc, disjoint and spread across divisions, directorates and agencies!



First Cosmological simulations to include black hole physics by Di Matteo et. al. at Carnegie Mellon funded by OCI and MPS/AST.

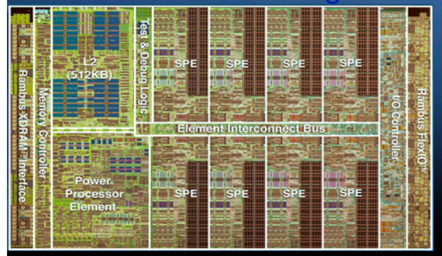
Optimal siting of oil exploration platform estimated by using simulation and optimization tools to maximize product

AMD Phenom

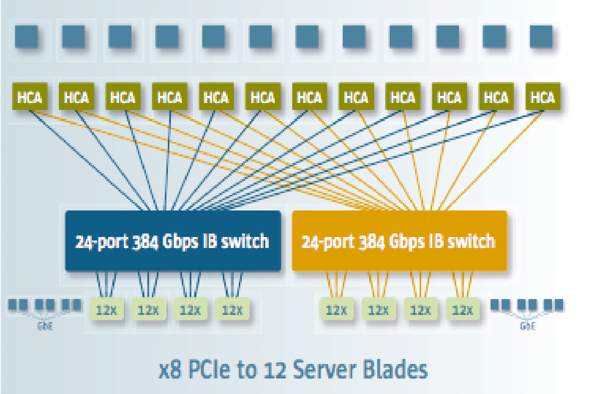


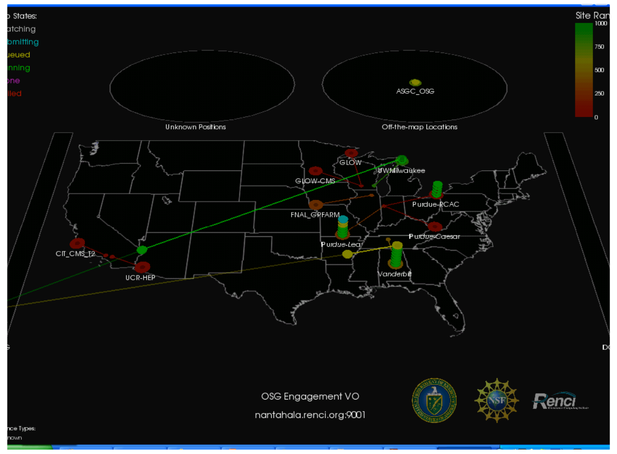
http://www.amd.com/us-en/assets/content_type/

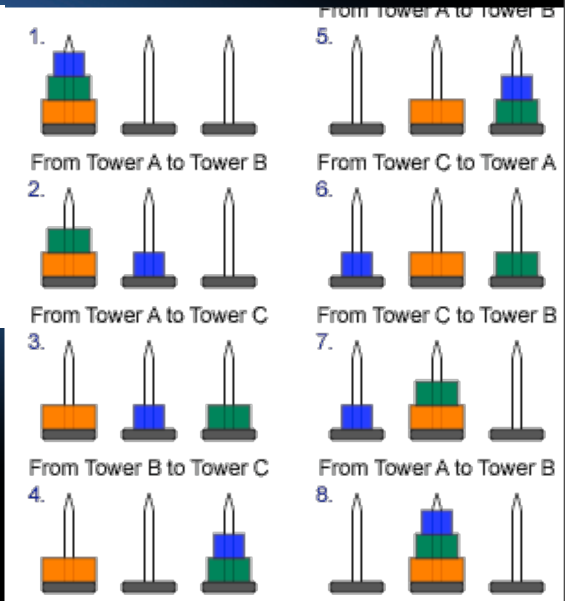
IBM Cell Broadband Engine



http://domino.research.ibm.com/comm/research.nsf/pages/r.arch.innovation.html?ope n







How?

- NSF/OCI engaged in actualizing "CI Vision..." -- Atkins et. al.
- Computational Science -- the unifying theme across many threads that lead to successful use of computational hardware in the discovery and innovation process -- support for which is ad hoc, disjoint and spread across divisions and directorates
- Advisory Committee on Cyberinfrastructure (ACCI) has formed sub-committees -- "Task Forces" to deal with multiple aspects -- HPC, Grand Challenges, Software, Campus Bridging, Data, LWD

“CI Task Force”

- “opportune time to carefully investigate alternate mechanisms and methodologies for ensuring that the research, development and sustenance of the nation’s software and tool infrastructure is well positioned to help our scientists with a competitive advantage and not a disadvantage.”

- The charge to the group comprises of the following:

  - Characterize and estimate the magnitude and scope of need

  - Develop initiatives and programs to promote future growth, development and sustainability of the software and tool infrastructure needed for transformative research and innovation leading to industrial competitiveness and knowledge leadership.

  - Analyze institutional and other barriers at NSF to promoting and supporting such an infrastructure.

Abani K. Patra
apatra@nsf.gov

Questions?

- What are the new applications that are emerging or likely to emerge in the coming decade?
- How can NSF best stimulate development of exascale software applications?
- How can useful software that has been developed as part of the exascale effort be sustained beyond the development period?
- What systems software will be required? Distributed systems support, programming environments, runtime support, data-management user tools?

Abani K. Patra
apatra@nsf.gov

Questions?

- What application support environments will be needed? Application packages, numeric and non-numeric library packages, problem-solving environments?
- How can NSF aid or catalyze developments that make it possible to use the same tools, including compilers, debuggers and performance tools, on system scales all the way down to the typical researcher's laptop or desktop?
- What education and training actions should be considered to prepare researchers, students and educators for future cyberinfrastructure?

Abani K. Patra
apatra@nsf.gov

Improving HPC Software

Pete Beckman & Jack Dongarra

IESP Workshop #1

Santa Fe, New Mexico: April 7 & 8

- Version 1.0 started by Ken Kennedy in 2006…
- Effort was re-launched in 2008
- Initial planning meeting at SC08
- This meeting sponsored by DOE & NSF in coordination with EU and Japanese
  - 68 attendees
- Subsequent meetings will be held and sponsored by Europe (end of June) and Japan (October)
- Workshop reports will focus plans and identify issues
- PIES?

Agenda

- Today
  - Goals and HPC Software Status
  - Science drivers and HPC plans: Japan
  - Architectural trends for HPC
  - Science drivers and HPC plans: Europe
  - Software Barriers for HPC, today and tomorrow
  - Science drivers and HPC plans: Europe: USA
- Tomorrow
  - Breakout groups:
    - Tech Roadmap, Collaboration / Coordination models

IESP Goal

Improve the world's simulation and modeling capability by improving the coordination and development of the HPC software environment

Workshops:

Build an international plan for developing the next generation open source software for scientific high-performance computing

Then Do It…

Components / Workshop Charge

- Outline of what a plan would include, and possible outcomes
- Assess the software needs for peta/exascale computation
- Explore how to develop a community architecture roadmap
- Gather and analyze existing R&D plans for addressing extreme scale; what is missing?
- Identify key technical areas to be included in plan
- Begin development of a roadmap for peta/exascale software
- Identify R&D models that enable laboratories, universities, and vendors to co-develop coordinated open source HPC software
- Examine funding and governance models that support international development

A Running Start: www.exascale.org

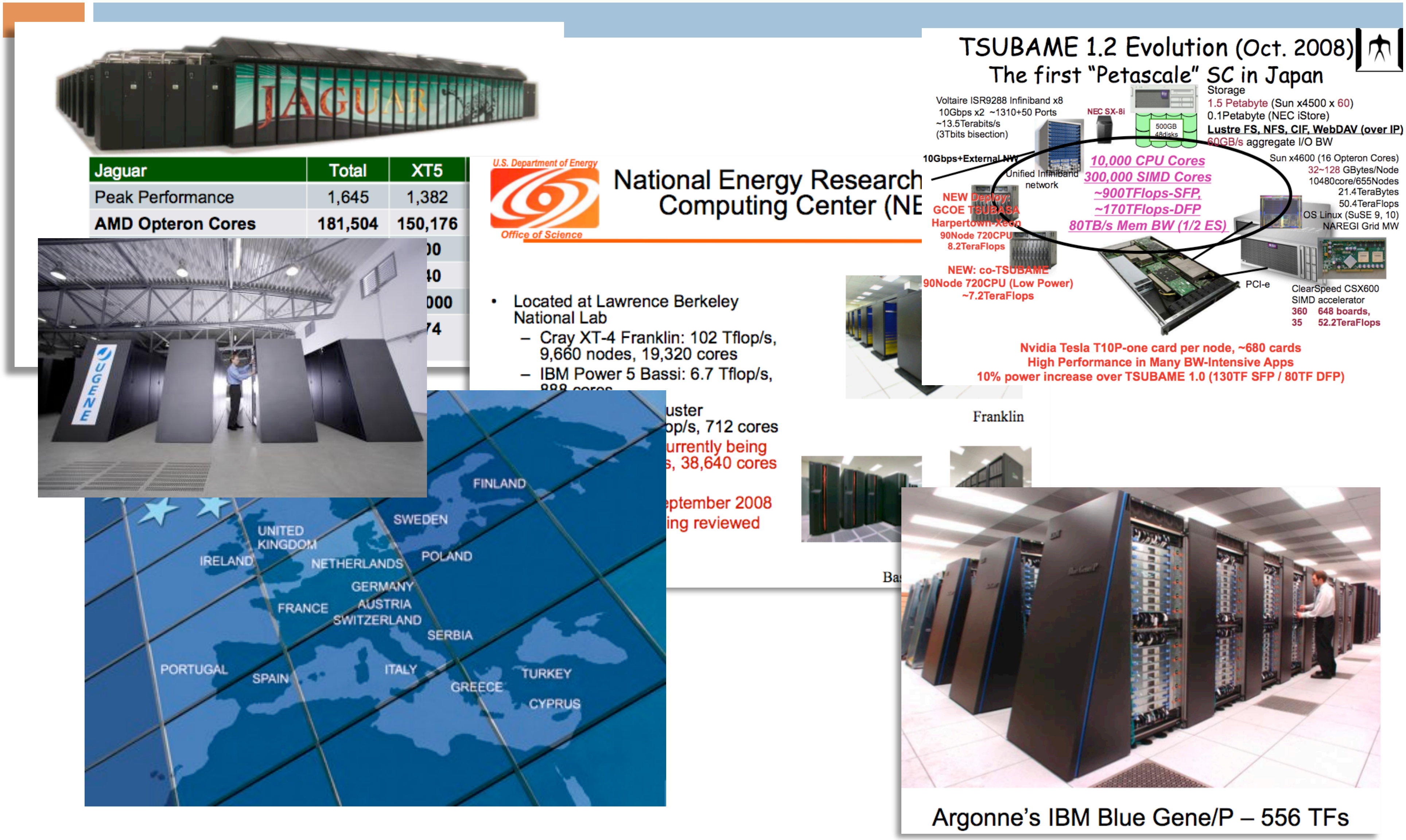
White papers

- *Musings on the Path Toward Exascale*, Robert Lucas – ISI/USC
- *BSC Vision Towards Exascale*, Mateo Valero, BSC
- *Software Challenges of Extreme Scale Computing*, Michael Heroux – Sandia National Laboratory
- *Software and Exascale Computing*, Bill Camp – Intel Corporation
- *Application Analysis and Porting in the PRACE Project*, Peter Michielse – Netherlands National Computing Facilities Foundation (NCF)
- *The Application Perspective – Seeking Productivity and Performance*, David Barkai – Intel Corporation
- *EDF white paper*, J.Y. Berthou and J.F. Hamelin – EDF R&D
- *The Biggest Need: A New Model of Computation*, Thomas Sterling – Louisiana State University
- *NSF IESP Whitepaper*, Abani Patra, Rob Pennington, Ed Seidel – Office of Cyberinfrastructure, National Science Foundation
- *A Proposal for a Capability Centers Consortium*, Bill Gropp, Mark Snir – NCSA and the University of Illinois at Urbana-Champaign
- *Slouching Towards Exascale*, Rusty Lusk, Argonne National Laboratory
- *A Collaboration and Commercialization Model for Eascale Software Research*, Mark Seager and Brent Gorda, Lawrence Livermore National Laboratory
- *The Case for A Hierarchal System Model for Linux Clusters*, Mark Seager and Brent Gorda, Lawrence Livermore National Laboratory
- *IESP Whitepaper: PDE-based applications and solvers at extreme scale*, DavidKeyes, Columbia University & SciDAC TOPS project

Outline: HPC Software

- Current State: HPC Software
- Background: Activities in Europe and Japan
- The Changing Architecture
- The IESP Workshops
- Roadmap and Outcomes

The Open Source Community Provides Most of the World's HPC Software



The Community is Diverse and Robust

- In the last 10 years, galvanization of Open Source development dramatically improved software

A very small sample:

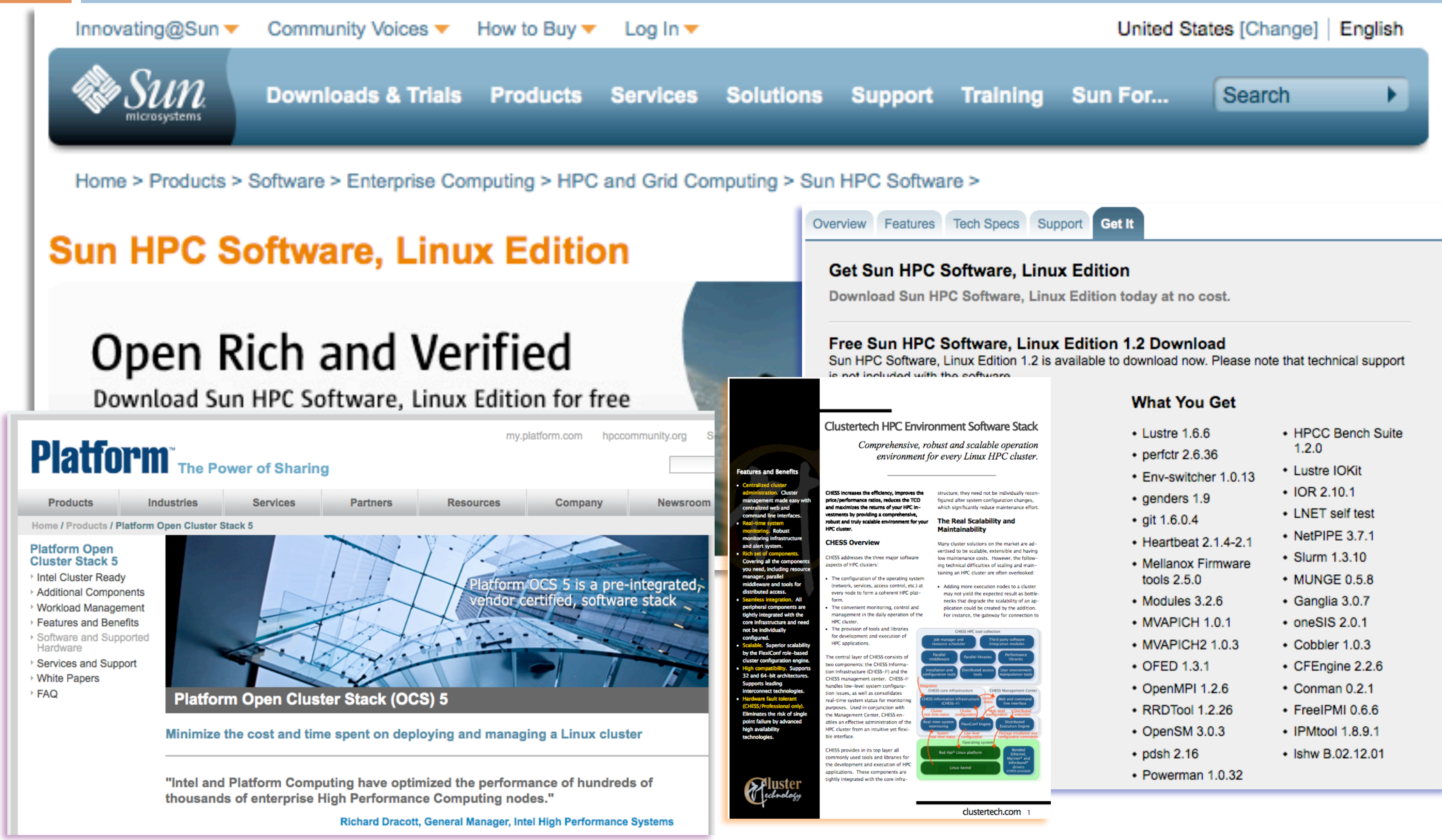
- Linux Operating System, libc
- Python, Perl
- PAPI, TAU, Kojak
- dCache
- UPC
- MPICH, OpenMPI
- ScaLAPACK
- JuBE
- VisIt
- GASNet, ARMCI/GA
- CFEngine, bconfig
- Ganglia
- SLURM, Cobalt
- Dyninst
- Torque/Moab, OpenPBS
- Charm++
- pNetCDF, HDF5
- GridFTP
- FFTW
- PVFS

A Long History of Collaboration & Sharing

The massive archive site WSMR-SIMTEL20.ARMY.MIL at White Sands Missile Range, New Mexico, USA, which is home to more than 2 gigabytes of files for many computer systems, including MSDOS, Unix, VMS and some mainframes, will be shut down by its operators as of September 20, 1993. Unless a new home is found for the archives, this major archive site will vanish.

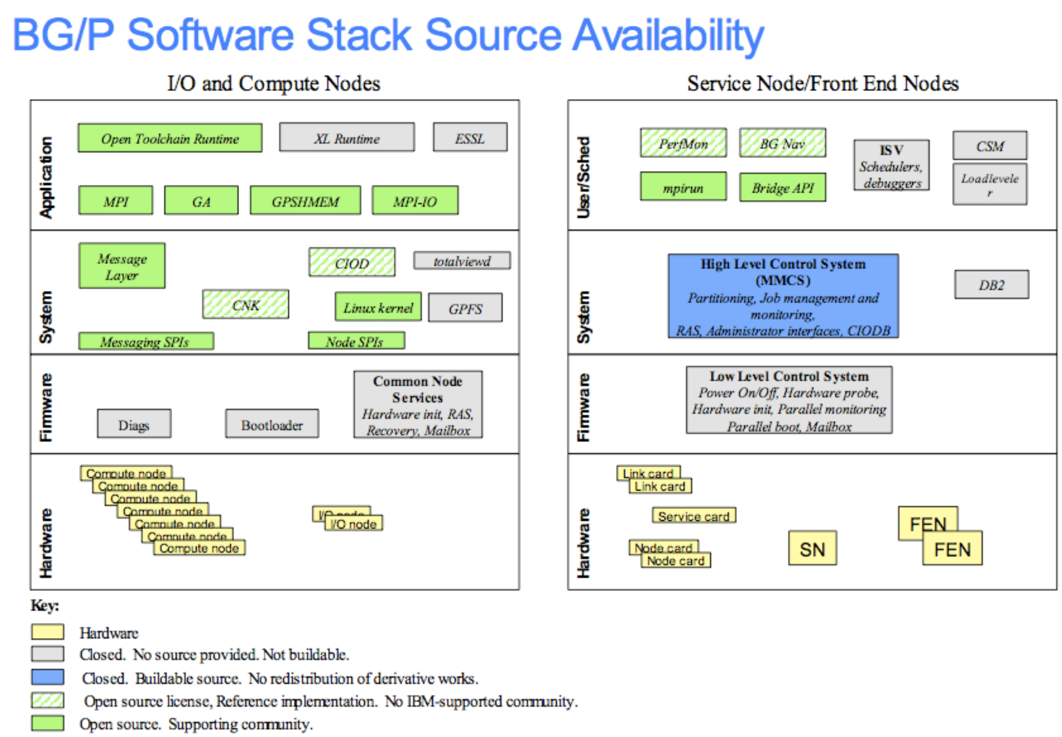
The Result….

Open Source HPC Software Stacks for Small Linux Clusters are Everywhere



Just Buy It?
Scalability Thins the Market

- For some markets, a closed source business model continues to work well
  - Single-node optimized math libraries & compilers
  - Debuggers for small clusters
  - Some queuing systems, parallel file systems, HSMs
  - Small cluster applications: Fluent, CFD++, etc

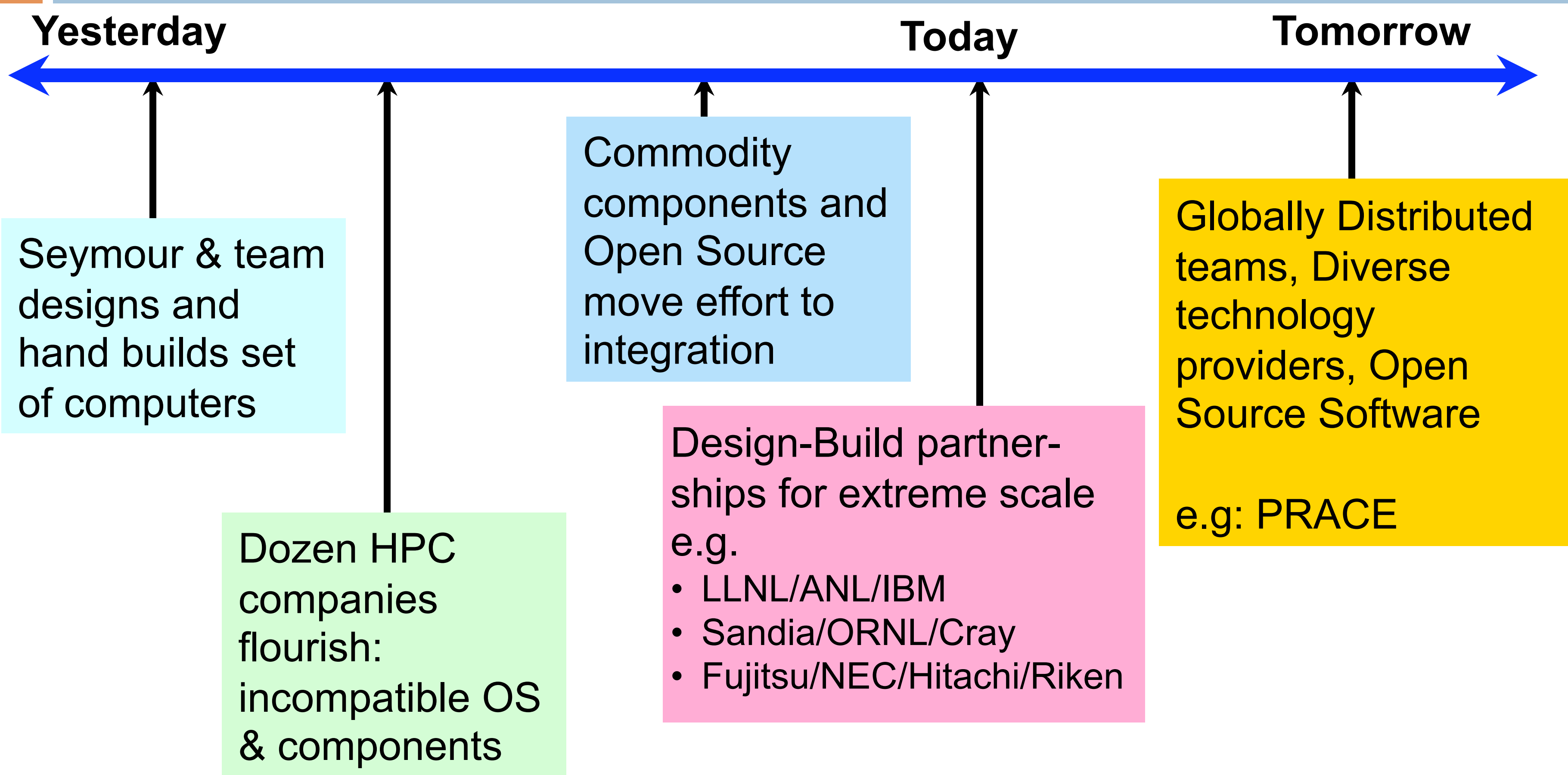


Why Seek to Improve This?

- The largest scale systems are becoming more complex, with designs supported by large consortium
  - The software community has responded slowly
- Significant architectural changes arriving
  - Software must dramatically change
- Our ad hoc community coordinates poorly, both with other software components and with the vendors
  - Computational science could achieve more with improved development and coordination

Extreme-Scale Platform Design:

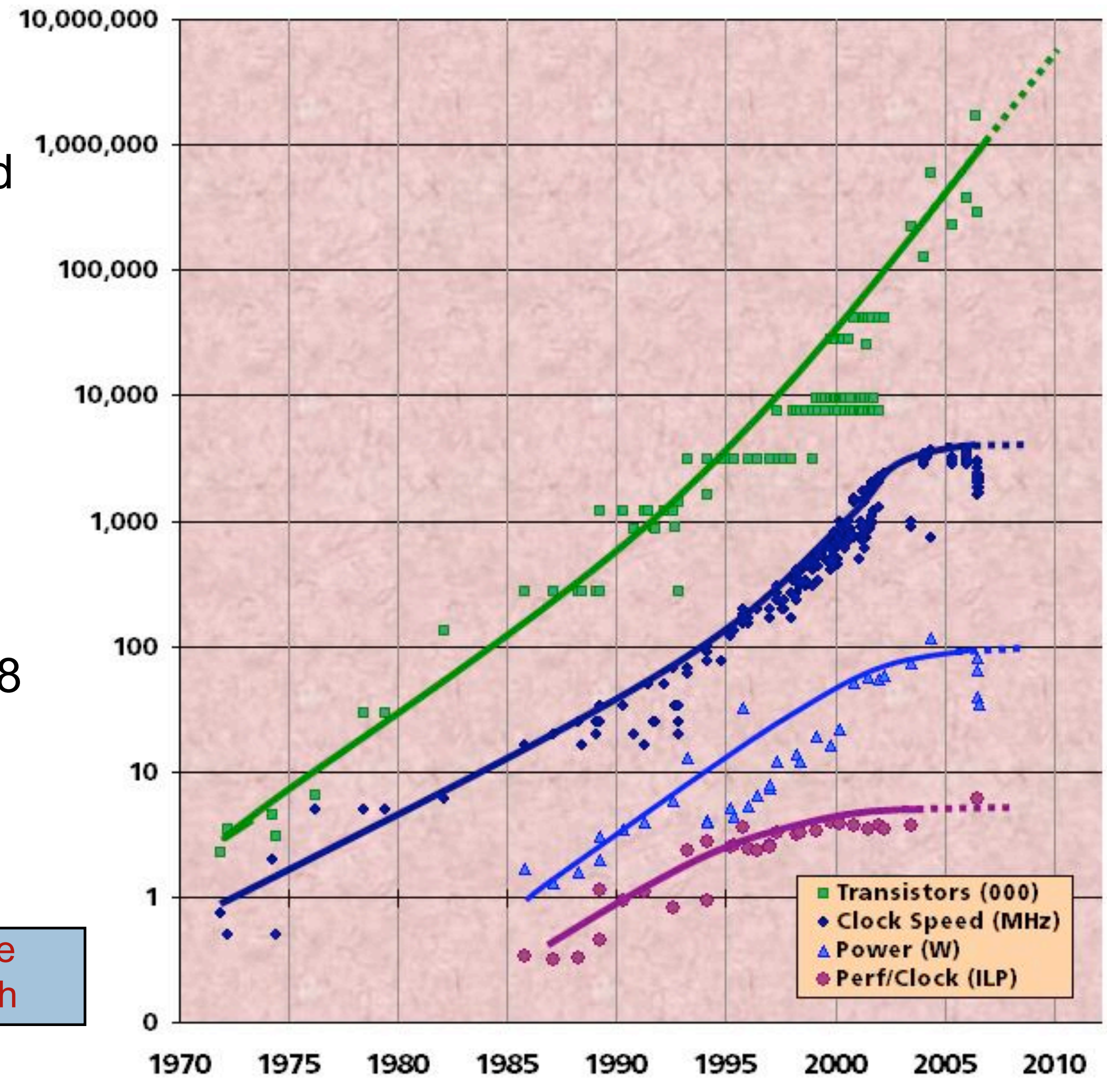
Industrial revolution and globalization has arrived

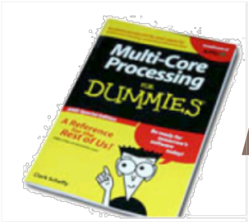


Traditional Sources of Performance Improvement are Flat-Lining (2004)

- New Constraints
  - 15 years of *exponential* clock rate growth has ended
- Moore's Law reinterpreted:
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: parallelism doubles every 18 months *instead* of clock frequency!

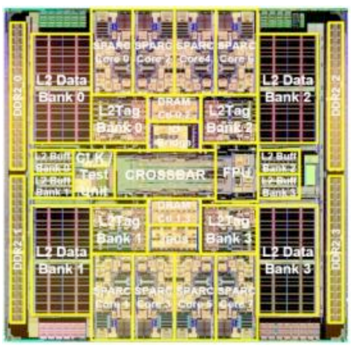
Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith





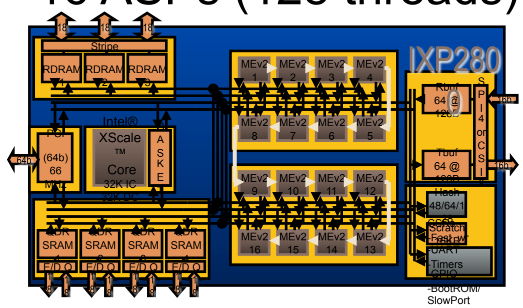
Multicore comes in a wide variety

- Multiple parallel general-purpose processors (GPPs)
- Multiple application-specific processors (ASPs)

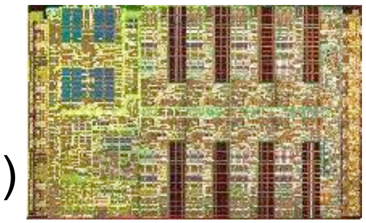


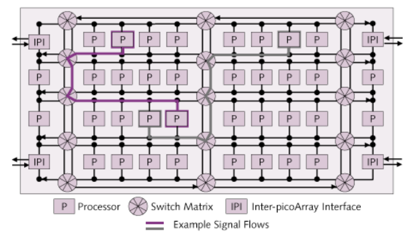
Sun Niagara
8 GPP cores (32 threads)

Intel Network Processor
1 GPP Core
16 ASPs (128 threads)



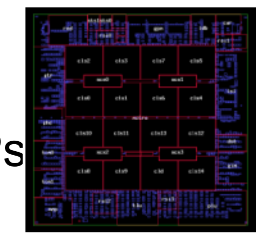
IBM Cell
1 GPP (2 threads)
8 ASPs

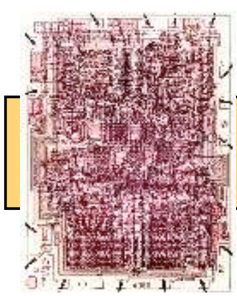




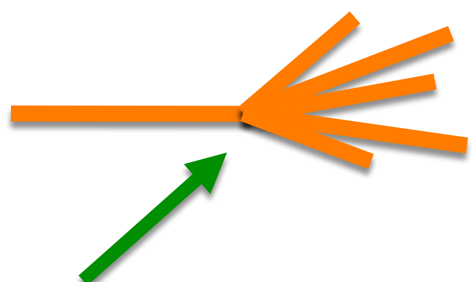
Picochip DSP
1 GPP core
248 ASPs

Cisco CRS-1
188 Tensilica GPPs





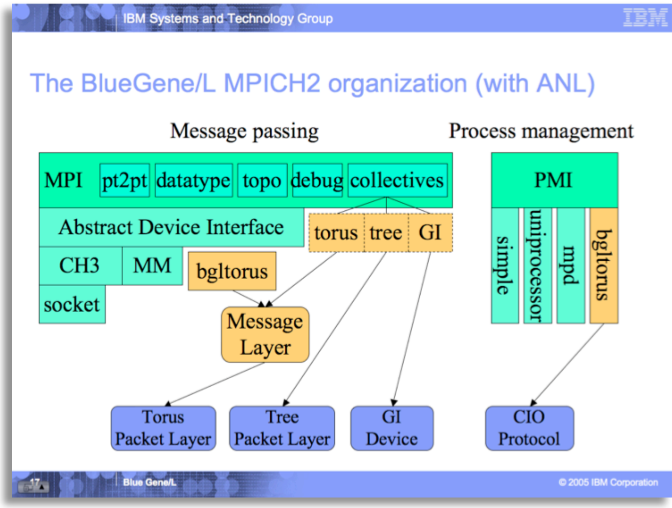
Intel 4004 (1971):
4-bit processor,
2312 transistors,
~100 KIPS,
10 micron PMOS,
11 mm^2 chip

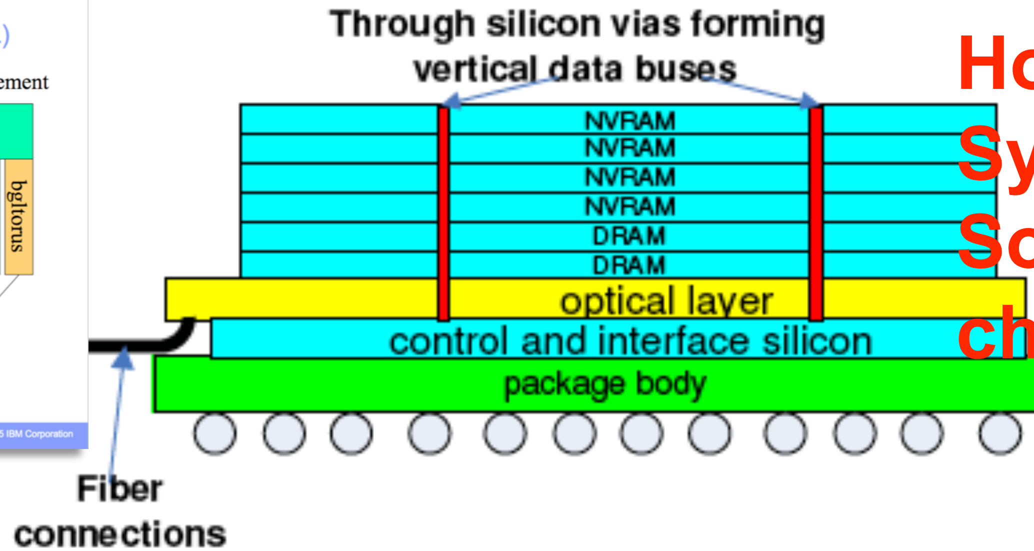


You Are Here

"The Processor is the new Transistor" [Rowen]

3D Packaging: Changing Paradigms





How will System Software change?

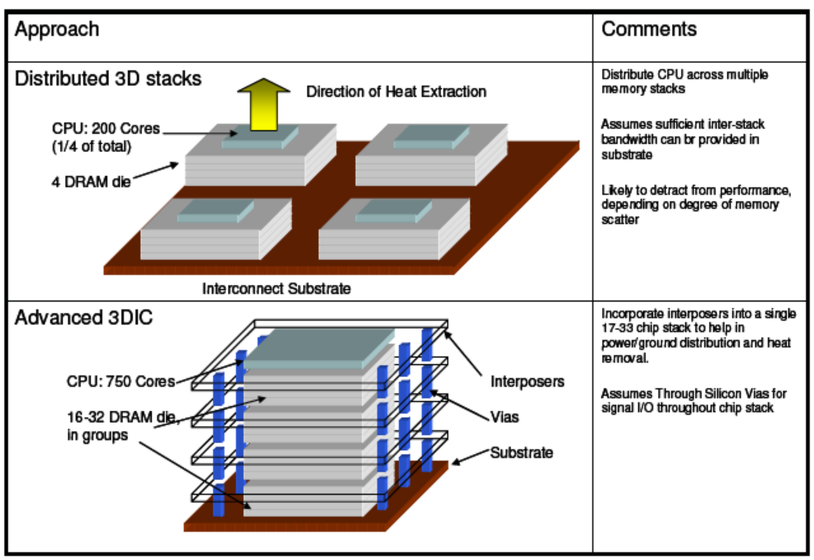


Figure 7.5: Potential directions for 3D packaging (A).

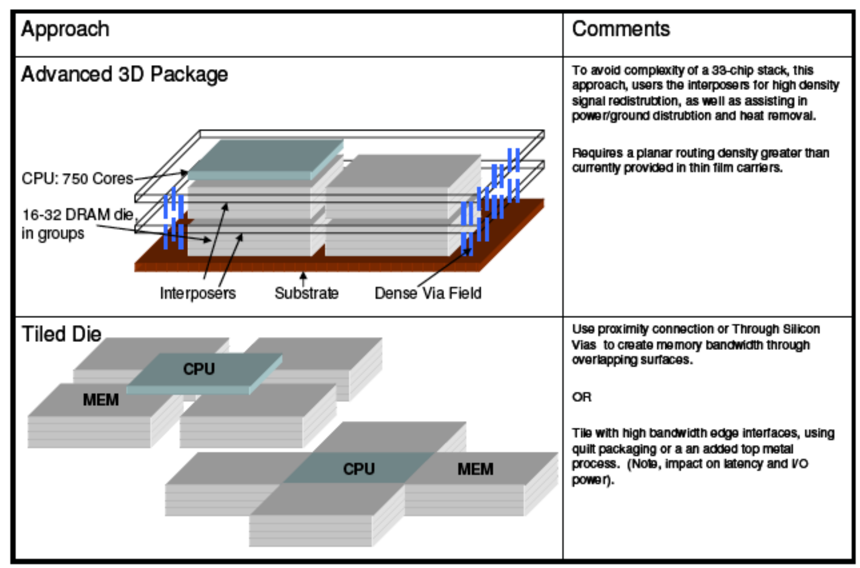
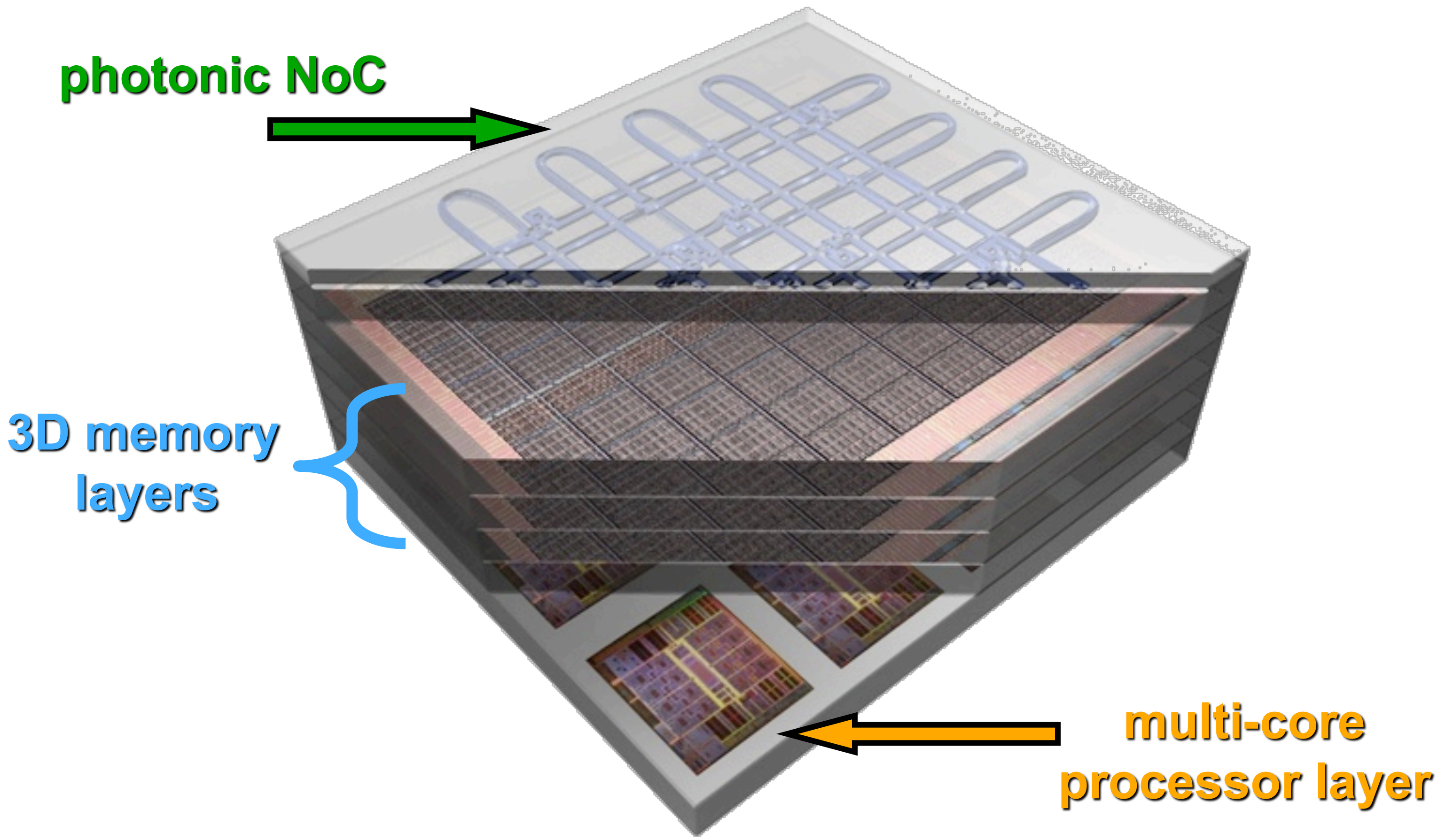


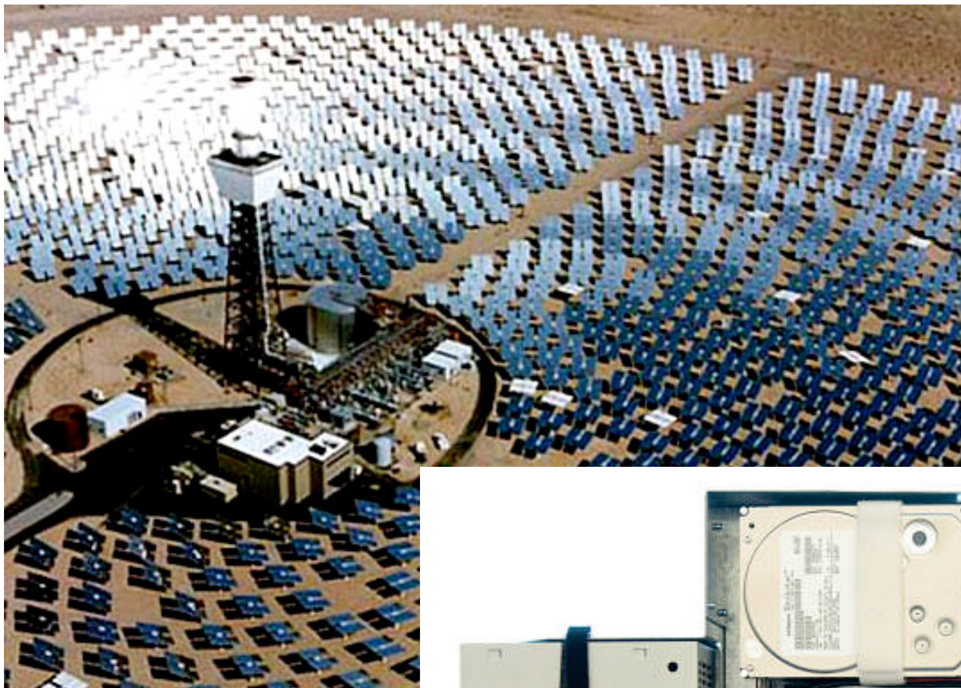
Figure 7.6: Potential directions for 3D packaging (B).

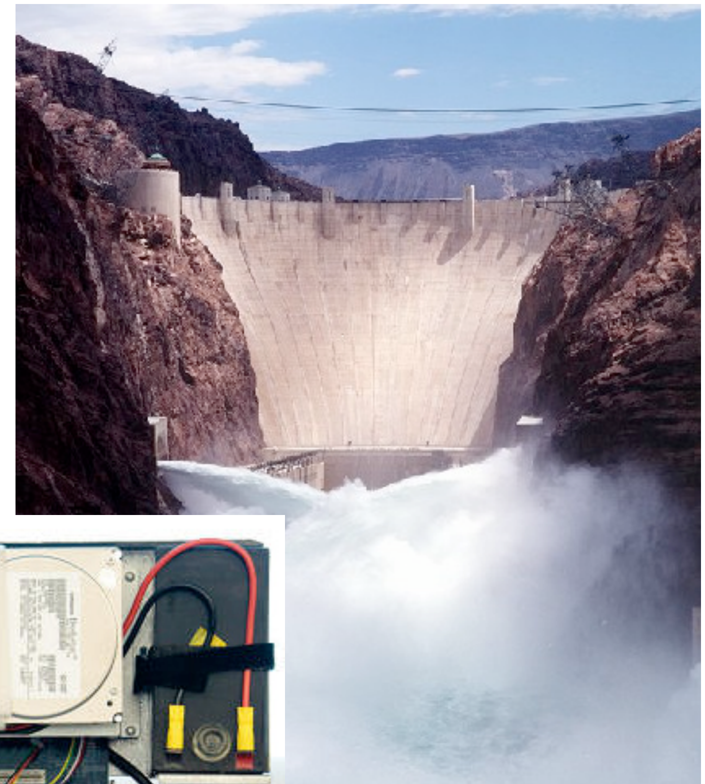
Vision of Photonic NoC Integration



Courtesy: Keren Bergman, Columbia

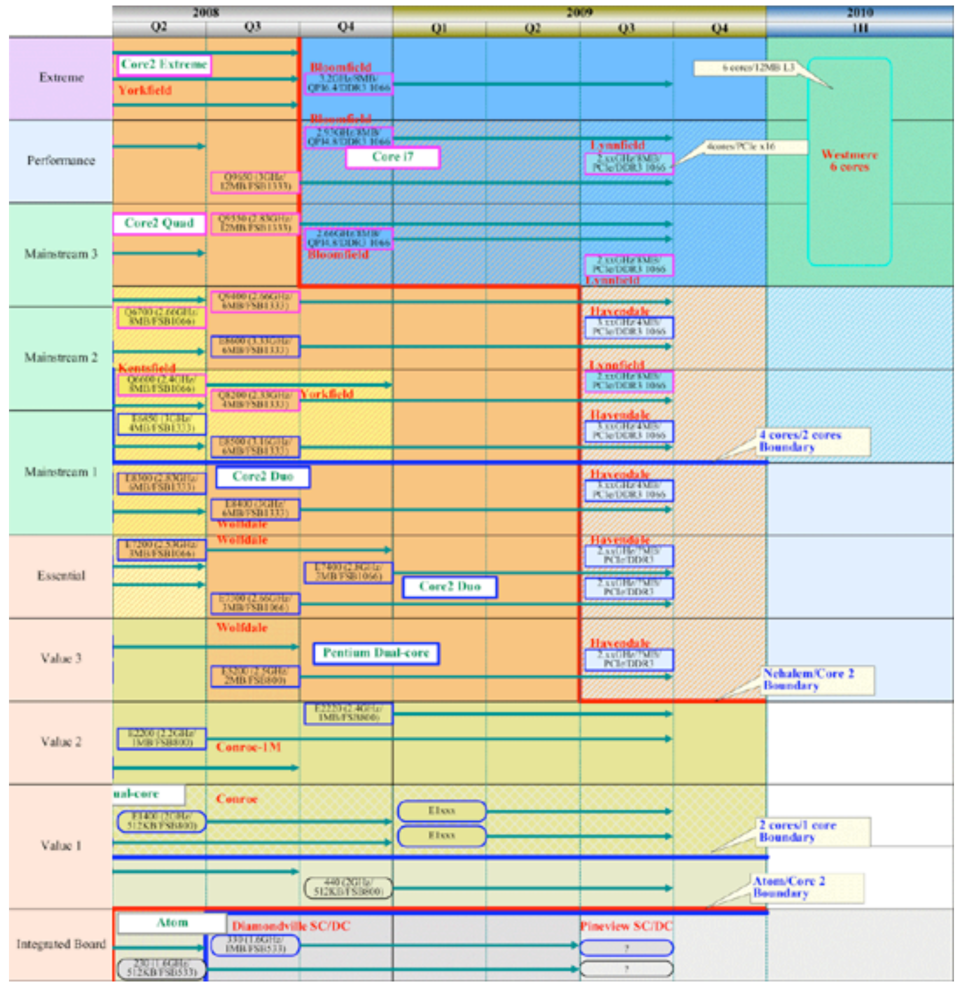
Power and Programming Models





To Build a SW Roadmap & Plan:

- What do we use today?
- What we need tomorrow?
- How we can fill in the gaps?



Inventory Exercise…

- Broadly divide software and functionality into hierarchical categories:
  - I/O Storage
  - Math Libraries
  - Performance Tools
  - Etc.
- Where it is run…
  - Service node, I/O nodes, compute nodes, login nodes, etc

Example Snippets: ORNL XT3

Software Requirements for HPC

Site: ORNL
System: Cray XT3
Submitted 8/25/06
Contact: Jeff Vetter, vetter@ornl.gov

Note that this list is best viewed as a da
(menu Data>>Filter>>Autofilter), and Pi

Site	System	Node Type	L1 Category	L2 Type	L3 Function	Package	Provider
ORNL	Cray XT3	All	App Support	Library	I/O & Storage	HDF5_PAR	NCSA
ORNL	Cray XT3	All	App Support	Library	I/O & Storage	HDF5_SERIAL	NCSA
ORNL	Cray XT3	All	App Support	Library	I/O & Storage	netCDF	UCAR/Unidata
ORNL	Cray XT3	All	App Support	Library	I/O & Storage	netCDF, parallel	ANL
ORNL	Cray XT3	Compute	App Support	Library	Math	PetSC	ANL
ORNL	Cray XT3	Compute	App Support	Library	Math	Aztec	Sandia
ORNL	Cray XT3	Compute	App Support	Library	Math	BLAS	AMD
ORNL	Cray XT3	Compute	App Support	Library	Math	FFTPack	Netlib
ORNL	Cray XT3	Compute	App Support	Library	Math	FFTW	MIT
ORNL	Cray XT3	Compute	App Support	Library	Math	LAPACK	AMD
ORNL	Cray XT3	Compute	App Support	Library	Math	MUMPS	CERFACS

LLNL BG/P

Site	System	Node Type	L1 Category	L2 Type	L3 Function	Package	Provider
LLNL	BG/P	Service	Prog Env	Tool	Infrastructure	LaunchMON	LLNL (Open Source)
LLNL	BG/P	Service	Prog Env	Tool	Infrastructure	MRNet	University of Wisconsin
LLNL	BG/P	Service	Prog Env	Tool	Infrastructure	DynInst	University of Wisconsin
LLNL	BG/P	Service	Prog Env	Tool	Infrastructure	StackWalker	University of Wisconsin
LLNL	BG/P	Service	Prog Env	Tool	Infrastructure	secure VNC	Vaporware
LLNL	BG/P	Service	Prog Env	Tool	GUI	Tool Gear	LLNL(Open Source)
LLNL	BG/P	Service	Prog Env	Tool	GUI	tcl/tk	Open Source
LLNL	BG/P	Service	Prog Env	Tool	GUI	X11	Open Source
LLNL	BG/P	Service	Prog Env	Tool	GUI	Qt	TrollTech (Open Source)
LLNL	BG/P	Compute	Prog Env	Tool	Performance Analysis	Tau	Paratools/Univ. of Oregon (
LLNL	BG/P	Compute	Prog Env	Tool	Performance Analysis	HPM	Processor Vendor and Linu
LLNL	BG/P	Compute	Prog Env	Tool	Performance Analysis	PAPI	UTK(Open Source)
LLNL	BG/P	Compute	Prog Env	Tool	Performance Analysis	OTF	Paratools (Open Source)
LLNL	BG/P	Service	Prog Env	Tool	Performance Analysis	Vampir/VampirServ	Dresden Univ
LLNL	BG/P	Service	Prog Env	Tool	Performance Analysis	VampirTrace	Dresden Univ
LLNL	BG/P	Service	Prog Env	Tool	Tool version selection	dotkit	LLNL(Open Source)
LLNL	BG/P	Service	Prog Env	Tool	Editor	emacs	Open Source
LLNL	BG/P	Service	Prog Env	Tool	Editor	vim	Open Source
LLNL	BG/P	Compute	Prog Env	Tool	Performance Analysis	mpiP	LLNL/ORNL(Open Source)
LLNL	BG/P	Service	Prog Env	Tool	Source Code Control	svn	Open Source
LLNL	BG/P	Service	Prog Env	Tool	Source Code Control	cvs	Open Source
LLNL	BG/P	Service	Prog Env	Tool	Source Code Control	git	Open Source

LLNL Viz/Analysis:

Package	Provider	Support	Criticality
VisIt	LLNL(Open Source)	LLNL	1
OpenGL	Open Source	Community	1
EnSight	CEI	Licensing	2
ImageMagick	Open Source	Open Source	2
Tecplot	Tecplot, Inc.	Licensing	2
IDL	ITT Visual Informations Systems	Licensing	2
gnuplot	Open Source	Open Source	2
POV-Ray	Open Source	Community	2
RasMol	Open Source	Community	2
vmd	UIUC(Open Source)	UIUC	2
ParaView	Open Source	Community	2
NCAR	NCAR(Open Source)	NCAR	3
mplayer	Open Source	Community	3
Blockbuster	LLNL(Open Source)	LLNL	3
GIMP	Open Source	Community	3
xxdiff/tkdiff/meld	Open Source	Community	3

Where We Are Today:

We are not prepared for the changes coming

- Hardware features are uncoordinated with software development
  - (power mgmt, multicore tools, math libraries, advanced memory models, etc)
- Only basic acceptance test software is delivered with platform
  - UPC, HPCToolkit, Optimized libraries, PAPI, can be *YEARS* late
- Vendors often “snapshot” key Open Source components and then deliver a stale code branch
  - Counterexample: A model that works – MPICH for BG/P
- Community codes unprepared for sea change in architectures
- Coordination via SOW/contract is poor and only involves 2 parties
- No global evaluation of key missing components

International Community Effort

- International collaboration is required because:
  - The scale of investment
  - The need for international input on requirements
  - Computational science projects are international
  - Europeans, Japanese, and Americans are each working on portions of the software
- The process must be open

Executive Committee:

Co-Chair: Jack Dongarra, Univ, of Tennesse / ORNL, US
Co-Chair: Pete Beckman, Argonne National Laboratory, US
Franck Cappello, INRIA, FR
Thomas Lippert, Jülich Supercomputing Centre, DE
Satoshi Matsuoka, Tokyo Institute of Technology, JP
Paul Messina, Argonne National Laboratory, US

An Example Development Community

The Apache Software Foundation

Meritocracy in Action.

The Apache Software Foundation provides support for the Apache community of open-source software projects. The **Apache projects** are characterized by a collaborative, consensus based development process, an open and pragmatic software license, and a desire to create high quality software that leads the way in its field.

We consider ourselves not simply a group of projects sharing a server, but rather a *community of developers and users*.

This page will give you everything you always wanted to know about the foundation but were afraid to ask. The difference between membership and committership, who decides what, how elections take ... what's the philosophy behind ... nt. Come and see behind the

Project	Sponsor	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
Bluesky	Incubator	2008-01-12	416	True	month	False	True	2008-07-22	224	0,0,8	7	0	True	False	True	True	True	False	False
Cassandra	Incubator	2009-01-01	61	True	month	False	True	2009-01-02	60	1,1,-	9	0	True	True	True	True	False	False	False
Click	Incubator	2008-07-21	225	False	group-3	True	True	2009-02-22	9	2,3,4	7	0	True	True	True	True	True	True	True
Composer	Incubator	2007-11-17	472	False	group-3	True	True	2008-10-09	145	0,0,1	-	0	True	False	True	True	True	False	False
Droids	HC, Lucene	2008-10-09	145	False	group-2	True	True	2008-10-23	131	0,0,3	4	0	True	True	True	True	True	False	False
Empire-db	Incubator	2008-07-08	238	False	group-1	True	True	2009-01-05	57	1,1,4	8	1	True	True	True	True	True	True	True
ESME	Incubator	2008-12-02	91	True	group-3	True	True	2008-12-05	88	0,1,-	10	0	True	True	True	True	True	False	False
Etch	Incubator	2008-09-02	182	False	group-3	True	True	2008-12-08	85	0,3,4	8	0	True	True	True	True	True	False	False
Hama	Incubator	2008-05-20	287	False	group-3	True	True	2008-11-18	105	0,1,3	6	0	True	True	True	True	True	False	False
Imperius	Incubator	2007-11-10	479	False	group-1	True	True	2009-02-05	26	1,1,2	9	1	True	True	True	True	True	False	False
JSecurity	Incubator	2008-05-20	287	False	group-1	True	True	2008-09-28	156	0,0,2	10	0	True	True	True	True	True	False	False
JSPWiki	Incubator	2007-09-17	533	False	group-1	True	True	2008-09-28	156	0,0,1	12	1	True	True	True	True	True	False	False
Kato	? not known		?	True	group-3	True	False		?		9	0	True	True	True	True	True	False	False
Log4php	Logging Services	2004-01-31	1858	False	group-3	True	True	2007-07-18	594	0,0,0	6	0	True	True	True	True	True	False	False
Lokahi	Incubator	2006-03-01	1098	False	group-2	True	True	2006-11-28	826	0,0,0	5	0	True	True	True	True	True	False	False
Lucene.Net	Lucene	2006-03-15	1084	False	group-1	True	True	2006-11-11	843	0,0,0	4	1	True	True	True	True	True	False	False
Olio	Incubator	2008-09-29	155	False	group-1	True	True	2009-02-05	26	1,3,5	12	0	True	True	True	True	True	True	False
OpenWebBeans	Incubator	2008-10-26	128	False	group-3	True	True	2009-01-18	44	1,7,7	6	2	True	True	True	True	True	True	True
PDFBox	Incubator	2008-02-07	390	False	group-2	True	True	2009-01-27	35	3,3,3	8	2	True	True	True	True	True	False	False
PhotArk	Incubator	2008-08-19	196	False	group-2	True	True	2008-10-26	128	0,0,4	6	0	True	True	True	True	True	False	False
Pivot	Incubator	2009-01-26	36	True	group-2	True	True	2009-02-20	11	1,-,-	3	0	False	True	True	True	True	False	False
RAT	Incubator	2008-01-06	422	False	group-3	True	True	2009-02-22	9	3,3,3	8	0	True	True	True	True	True	False	False
RCF	MyFaces	2007-04-06	697	False	group-1	True	True	2008-04-16	321	0,0,0	17	0	True	True	True	True	False	False	False
River	Incubator	2006-12-26	798	False	group-3	True	True	2008-11-10	113	0,1,1	19	0	True	True	True	True	True	True	True
Sanselan	Incubator	2007-09-09	541	False	group-1	True	True	2008-09-28	156	0,0,4	6	0	True	True	True	True	True	True	True

... n incorporated in the United

... ects by supplying hardware,

- create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit
- provide a means for individual volunteers to be sheltered from legal suits directed at the

Apache Projects

- HTTP Server
- Abdera
- ActiveMQ
- Ant
- APR
- Archiva
- Beehive
- Camel
- Cayenne
- Cocoon
- Commons
- Continuum
- CouchDB
- CXF
- DB
- Directory
- Excalibur
- Felix
- Forrest
- Geronimo
- Gump
- Hadoop
- Harmony
- HiveMind
- HttpComponents
- iBATIS

Foundation

- FAQ
- Licenses
- News
- Public Record
- Sponsorship
- Donations
- Thanks
- Contact

Foundation Proje

- Conferences
- Infrastructure
- JCP
- Legal Affairs
- Security
- Travel Assistar

How it works

- Introduction
- Meritocracy
- Structure
- Roles
- Collaboration
- Infrastructure
- Incubator
- Other entities

Apache Foundation

- Create a foundation for open, collaborative software development projects by supplying hardware, communication, and business infrastructure
- Incubator projects can become Apache projects
- 800 “committers”
- The ASF Infrastructure is mostly composed of the following services:
  - the web serving environment (web sites and wikis)
  - the code repositories
  - the mail management environment
  - the issue/ bug tracking
  - the distribution mirroring system

A Plan Could Include:

- Work with vendors to create the HPC equivalent to the ITRS (Int'l Tech Roadmap for Semiconductors)
  - Get community working on software before machine becomes available
- Community proposed unified roadmap for exascale software
- Identify missing components for future architectures and a plan to address them
- Develop models for working more closely with vendors
  - (support, acceptance tests, target features)
- Identify key application areas to drive development
- Community software development models
- Funding and organizational models (Apache, etc)

Achievable Outcomes

- Improve the capability of computational science
- Build and strengthen international collaborations and leadership; deliver more capable, productive HPC systems
- Build and improve R&D program developing new programming models and tools addressing extreme scale
- Strategic plan for HPC research
- Open source HPC development guided by roadmap with better coordination and fewer missing components
- Joint programs in education and training for the next generation of computational scientists.
- Vendor engagement and coordination for more capable software supporting exascale science

Possible Models (from loose to tight collaboration)

- Identify needs, focus Int'l R&D attention on missing components
- Coordinate features, delivery schedule, interoperability, and improvements across international R&D teams
- IESP community recommends funding for key areas
- Provide forums for vendors and community to work together on roadmaps
- Fund R&D and subsequent deployment of key components
- Fund collaborative relationship with vendors and co-develop components
- Test, integrate, and support internationally developed software components
- Build integrated software that can pass acceptance tests on extreme platforms

Future Workshops and Report

- 3 workshops over the next year
  - 1: Santa Fe, April 7-8
  - 2: France, June 28-29
  - 3: Japan in the early Fall
- Broad engagement by the community
- Initial reports in summer 2009
- Final report for first year at SC09
- Planning for *IMMEDIATE* payoff
  - Could begin initial components of plan this year

www.exascale.org

10[18] INTERNATIONAL EXASCALE SOFTWARE PROJECT

HOME MEETINGS DOCUMENTS COMMUNITY

MAIN PAGE

PAGE DISCUSSION VIEW SOURCE HISTORY

The mission of the **International Exascale Software Project (IESP)** is to lay the foundation for exascale computing by mobilizing the global open source software community to combine and coordinate their collective efforts far more efficiently and effectively than ever before. The IESP will hold a series of three workshops to organize and structure this community wide effort. The first, invitation-only workshop will occur on April 7th and 8th in Sante Fe, New Mexico, US, with people arriving in time for a reception on April 6th. Attendees will include members from industry, academia, and government, with expertise in a range of critical areas.

WORKSHOP INFORMATION

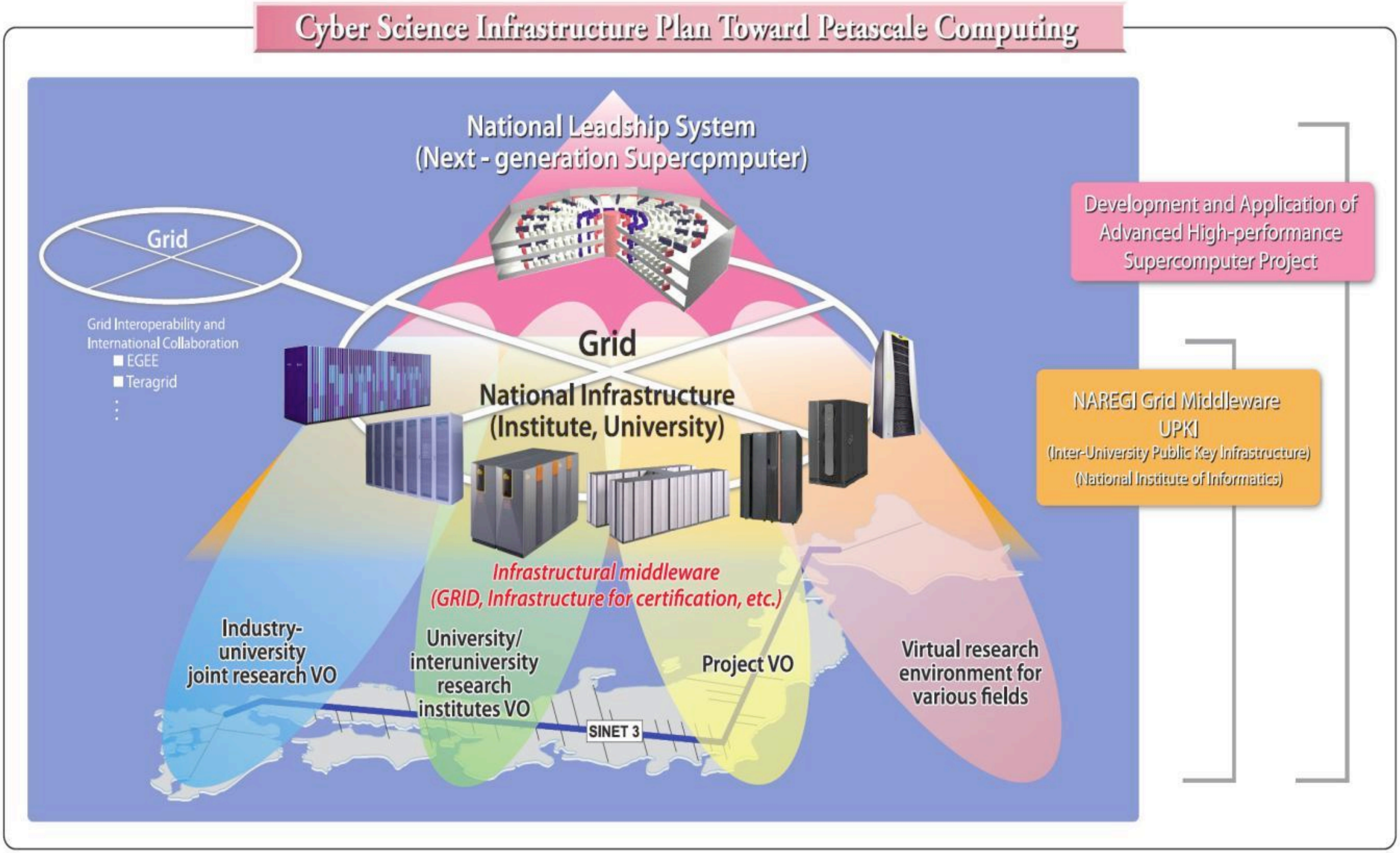
- Workshop Arrangements
- Workshop Agenda
- Workshop Charge
- Executive Committee
- Organizing Committee
- Workshop Attendees
- Whitepapers
- Background Material

Thou Shalt Specialize or Commoditize? The Japanese Situation Towards Peta and Exascale

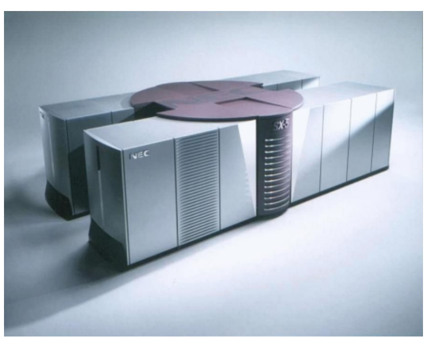
Satoshi Matsuoka, Prof., Dr. Sci.
GSIC Center, Tokyo Institute of Technology / National Institute of Informatics

DoE IESP Workshop @ Santa Fe, NM, USA Apr. 6-8, 2009

The Ideal: Hiearchy of Deployments



Vector Machines- NEC SX



- ACOS/SX-1
- SX-2 (1983): Bipolar, 4-wide, 1.3GFlops, single CPU
- SX-3: (1989): Bipolar, 8-wide, 22GFLOPS(2x4CPUs)
- SX-4 (1994): CMOS 8-wide 64GF/node (2GF x 32CPUs)
- SX-5 (1998): 16wide, 128GF/node (8GF x 16 CPUs)
- SX-6 (2001): 8-wide x 2 clock, 64GF/node (8x8CPUs), core of ES
- SX-7 (2002): 8-wide x 2 clock, 282GF/node (9GF x 32 CPUs)
- SX-8 (2004): 8-wide 2Ghz vector, 128GF/node (16GF x 8CPUs)
- SX-9 (2007): 8-wide x 4 3.2 Ghz 102GF/CPU, 1.6TF/node, 128GB/s inter-node BW

Glory Days of Vectors…just 12 years ago

Cray	71	
NEC	40	
Fujitsu	33	(#2)
Hitachi	9	
CM-2	7	
Total	160	
x86 (Meiko)	3	

“Cretaceous
”

2008/06/24 16:24:27ics ▸ Sublist Generator

TOP500 Sublist Generator

R_{max} and R_{peak} values are in GFlops. For more details about other fields, check the TOP500 description.

40 entries found.

Top500 June 1996, 40 NEC SXs

Rank	Site	System	Cores	R_{max}	R_{peak}
10	HWW/Universitaet Stuttgart Germany	SX-4/32 NEC	32	66.53	64
11	NEC Fuchu Plant Japan	SX-4/32 NEC	32	66.53	64
24	Japan Marine Science and Technology Japan	SX-4/20 NEC	20	42.4	40
25	National Research Institute for Metals Japan	SX-4/20 NEC	20	42.4	40
26	Toyota Central Research & Development Japan	SX-4/20 NEC	20	42.4	40
30	National Aerospace Laboratory (NLR) Netherlands	SX-4/16 NEC	16	34.42	32
31	National Cardiovascular Center Japan	SX-4/16 NEC	16	34.42	32
41	Swiss Scientific Computing Center (CSCS) Switzerland	SX-4/12 NEC	12	25.8	24
49	Atmospheric Environment Service (AES) Canada	SX-3/44R NEC	4	23.2	25.6
50	Tohoku University Japan	SX-3/44R NEC	4	23.2	25.6
55	Atmospheric Environment Service (AES) Canada	SX-3/44 NEC	4	20	22
59	Institute for Molecular Science Japan	SX-3/34R NEC	3	17.4	19.2
60	ATR Optical Communication Lab Japan	SX-4/8 NEC	8	17.2	16
61	Atmospheric Environment Service (AES) Canada	SX-4/8 NEC	8	17.2	16
62	Danish Meteorological Institute Denmark	SX-4/8 NEC	8	17.2	16
63	National Geographic Agency Japan	SX-4/8 NEC	8	17.2	16
111	Veritas DGC United States	SX-4/6 NEC	6	12.9	12
136	German Aerospace Laboratory (DLR) Germany	SX-3/24R NEC	2	11.6	12.8
137	National Institute for Fusion Science Japan	SX-3/24R NEC	2	11.6	12.8

Japan had ~30% performance share as a country

Rise of the Commodity Clusters: "The Scenario"

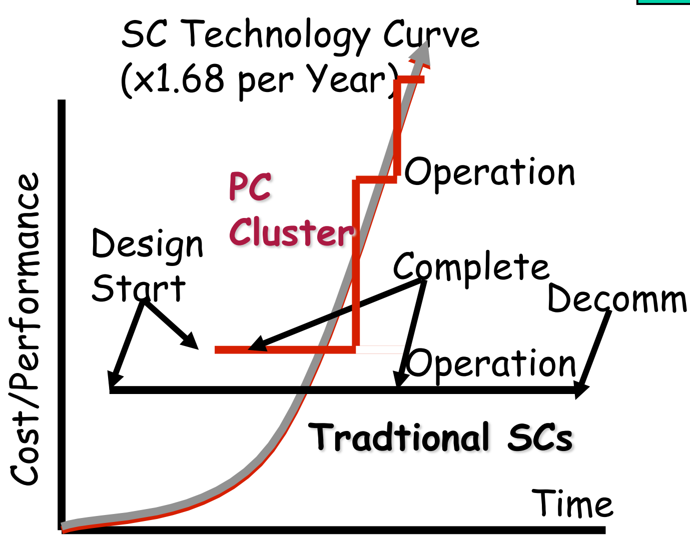
High Performance Commodity Computing
-High Performance x86 CPUs
-Fast Commodity Interconnect
-Cluster Software

High-Performance x86 CPUs

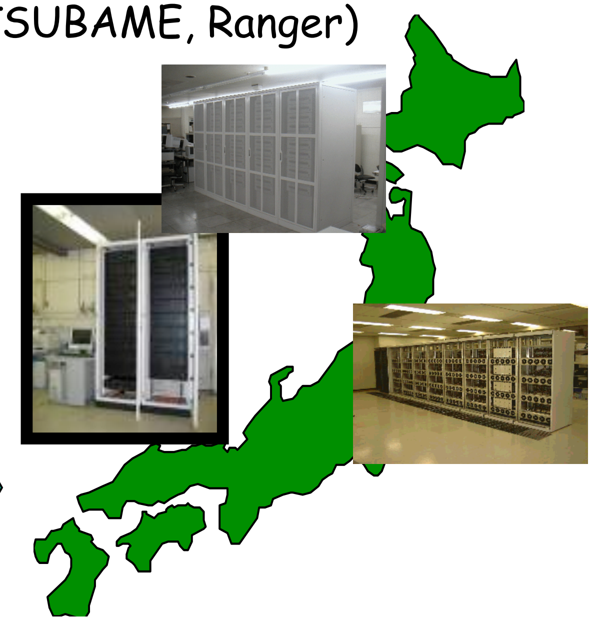
Myrinet, Infiniband, etc.

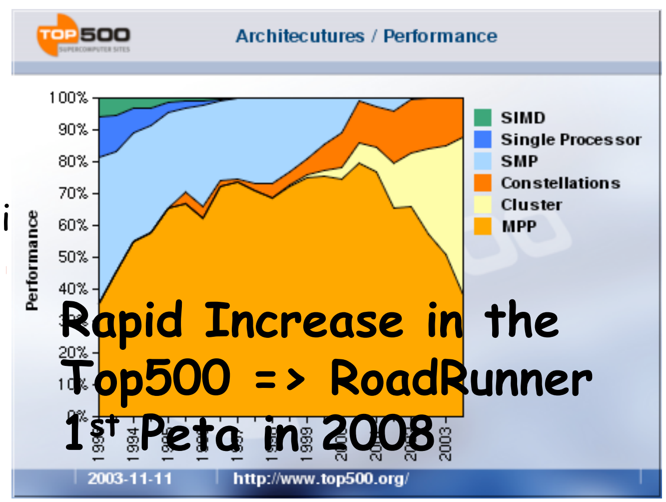
Rise and spread of Commodity Clusters and increase in their size

Real-time tracking of technology curve



Widespread Use of Clusters: Small to very large (e.g. TSUBAME, Ranger)





Rapid Increase in the Top500 => RoadRunner
1st Peta in 2008

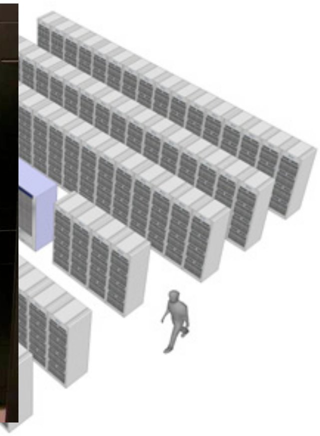
The First Beowulf – Wiglaf (1993~4) (NASA/CalTech)

- 16 processors
- Intel 80486 100 MHz
- VESA Local bus
- 256 Mbytes memory
- 6.4 Gbytes of disk
- Dual 10 base-T Ethernet
- 72 Mflops sustained, on real PPM code
- $40K
- Did not even come close to Top500

Early PC Clusters & Top500

- The 1st WS Cluster ranked: June 1997 (The 9th Top500)
  - UC Berkeley NOW, 344th (10.14 GFlops)
- The 1st PC Cluster ranked: June 1999 (The 13th Top500)
  - Univ. Bonn、Parnass2 Cluster, 362nd (29.5 GFlops)
- The 1st US PC Cluster: June 2000 (The 15th Top500)
  - NCSA (Windows) NT Supercluster, 207th (62 GFlops)
- The 1st Teraflop Cluster: Nov. 2002 (The 19th Top500)
  - LLNL MCR Linux Networx Linux Cluster Xeon 2.4 GHz – Quadrics, 5th (5694 Gflops)

And this went to Petascale, Despite all the Skepticism

- TACC Ranger
  - The largest x86 Linux Cluster ~50,000 x86 cores
  - 4th (326 TFlops) June 2008 (The 30th Top500)
- RR: the first #1 "commodity" cluster on Top500 June 2008(The 30th Top500)
  - The first Petaflop machine
  - The first #1 machine to use IB
  - The first #1 Linux machine
  - The first #1 "heterogeneous" SC (Cell and Opteron)

From Computonik Shock to Apollomodity Shock

Computnik article here

- 2002 The Japanese Earth Simulator New York Times "Computonik Shock"
  - Top500 #1 @ 35.8 TeraFlops
- 2004-5 US BG/L >100Tera like the Geminis
- 2008 US Roadrunner hitting Peta like Apollo 11 "commodity prevails"

The world's fastest computer uses chips based on one from Sony's PlayStation 3.

wrong," said Michael R. Anastasio, a physicist who is director of the Los Alamos National Laboratory. "This gives us a window into a whole new way of computing. We can look at phenomena we have never seen before."

percomputer created by I.B.M. reclaimed the speed record for the United States. The Japanese challenge, however, led Congress and the Bush administration to reinvest in high-performance computing.

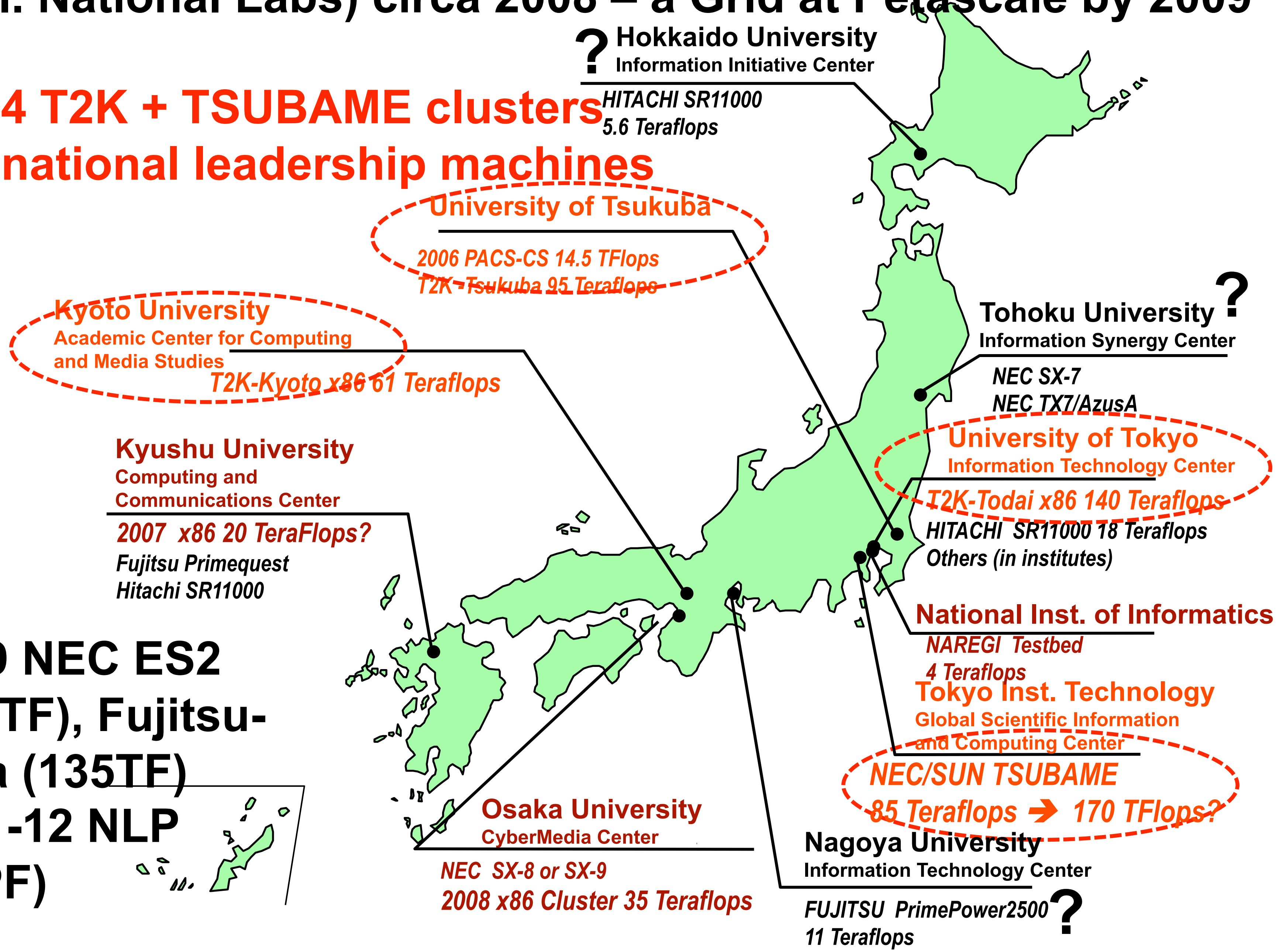
"It's a sign that we are maintaining our position," said Peter J. Ungaro, chief executive of Cray, a maker of supercomputers. He noted, however, that "the real competitiveness is based on the discoveries that are based on the machines."

Having surpassed the petaflop barrier, I.B.M. is already looking toward the next generation of supercomputing. "You do these

"One small step for RR, one giant leap for supercomputing"

Japan's 9 Major University Computer Centers (excl. National Labs) circa 2008 – a Grid at Petascale by 2009

The 4 T2K + TSUBAME clusters are national leadership machines



2009 NEC ES2 (131TF), Fujitsu-Jaxa (135TF)
2011-12 NLP (10PF)

The TSUBAME 1.0 @ Tokyo Tech. Spring 2006-- ~80 Teraflops Peak

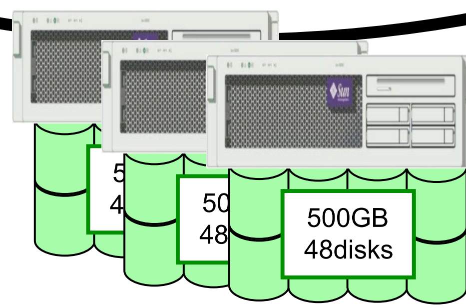
Unified IB network

Voltaire ISR9288 Infiniband 10Gbps x2 (DDR next ver.)
~1310+50 Ports
~13.5Terabits/s (3Tbits bisection)

10Gbps+External Network

“Fastest Supercomputer in Asia” 7th on the 27th Top500@38.18TF

Sun Galaxy 4 (Opteron Dual core 8-socket)
10480core/655Nodes
21.4Terabytes
50.4TeraFlops
OS Linux (SuSE 9, 10)
NAREGI Grid MW



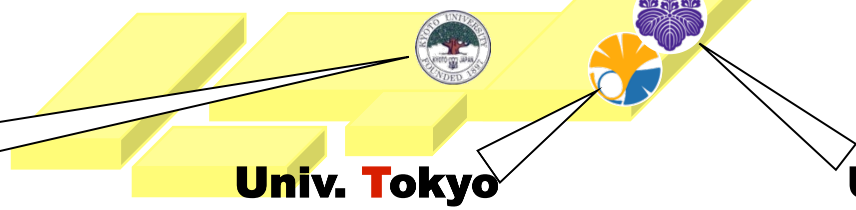
Storage
1.0 Petabyte (Sun “Thumper”)
0.1Petabyte (NEC iStore)
Lustre FS, NFS, WebDAV (over IP)
50GB/s aggregate I/O BW

ClearSpeed CSX600
SIMD accelerator
360 boards,
35TeraFlops(Current))

T2K Open Supercomputer Alliance

- Primary aiming at design of common specification of new supercomputers.
- Now extending to collaborative work on research, education, grid operation, ..., for inter-disciplinary computational (& computer) science.

- *Open* hardware architecture with commodity devices & technologies.
- *Open* software stack with open-source middleware & tools.
- *Open* to user's needs not only in FP & HPC field but also INT world.



Kyoto Univ.
416 nodes (61.2TF) / 13TB
Linpack Result:
Rpeak = 61.2TF (416 nodes)
Rmax = 50.5TF

Univ. Tokyo
952 nodes (140.1TF) / 31TB
Linpack Result:
Rpeak = 113.1TF (512+256 nodes)
Rmax = 83.0TF

Univ. Tsukuba
648 nodes (95.4TF) / 20TB
Linpack Result:
Rpeak = 92.0TF (625 nodes)
Rmax = 76.5TF

From Glory Days to Near Extinction…in 10+ years

- Japan as a country now only has 4% share --- now beaten by France
- Big Iron Vector & SMP SC now "niche"
- Clusters too small for cost, vendor inexperiences

"Cretaceous"

2008/06/24 16:24:27ics ▸ Sublist Generator

TOP500 Sublist Generator

R_{max} and R_{peak} values are in GFlops. For more details about other fields, check the TOP500 description.

40 entries found.

Top500 June 1996, 40 NEC SXs

Rank	Site	System	Cores	R_{max}	R_{peak}
10	HWW/Universitaet Stuttgart Germany	SX-4/32 NEC	32	66.53	64
11	NEC Fuchu Plant Japan	SX-4/32 NEC	32	66.53	64
24	Japan Marine Science and Technology Japan	SX-4/20 NEC	20	42.4	40
25	National Research Institute for Metals Japan	SX-4/20 NEC	20	42.4	40
26	Toyota Central Research & Development Japan	SX-4/20 NEC	20	42.4	40
30	National Aerospace Laboratory (NLR) Netherlands	SX-4/16 NEC	16	34.42	32
31	National Cardiovascular Center Japan	SX-4/16 NEC	16	34.42	32
41	Swiss Scientific Computing Center (CSCS) Switzerland	SX-4/12 NEC	12	25.8	24
49	Atmospheric Environment Service (AES) Canada	SX-3/44R NEC	4	23.2	25.6
50	Tohoku University Japan	SX-3/44R NEC	4	23.2	25.6
55	Atmospheric Environment Service (AES) Canada	SX-3/44 NEC	4	20	22
59	Institute for Molecular Science Japan	SX-3/34R NEC	3	17.4	19.2
60	ATR Optical Communication Lab Japan	SX-4/8 NEC	8	17.2	16
61	Atmospheric Environment Service (AES) Canada	SX-4/8 NEC	8	17.2	16
62	Danish Meteorological Institute Denmark	SX-4/8 NEC	8	17.2	16
63	National Geographic Agency Japan	SX-4/8 NEC	8	17.2	16
111	Veritas DGC United States	SX-4/6 NEC	6	12.9	12
136	German Aerospace Laboratory (DLR) Germany	SX-3/24R NEC	2	11.6	12.8
137	National Institute for Fusion Science Japan	SX-3/24R NEC	2	11.6	12.8

"Paleogene"

(Top500 Jun 2008 --- just 1 SX)

2008/06/24 16:29:29ics ▸ Sublist Generator

TOP500 Sublist Generator

Top500 Nov 2007, 2 NEC SXs

R_{max} and R_{peak} values are in GFlops. For more details about other fields, check the TOP500 description.

2 entries found.

Rank	Site	System	Cores	R_{max}	R_{peak}
30	The Earth Simulator Center Japan	Earth-Simulator NEC	5120	35.86	40.96
200	HWW/Universitaet Stuttgart Germany	SX8/576M72 NEC	576	8.92	9.22

In Response: Japanese Petascales

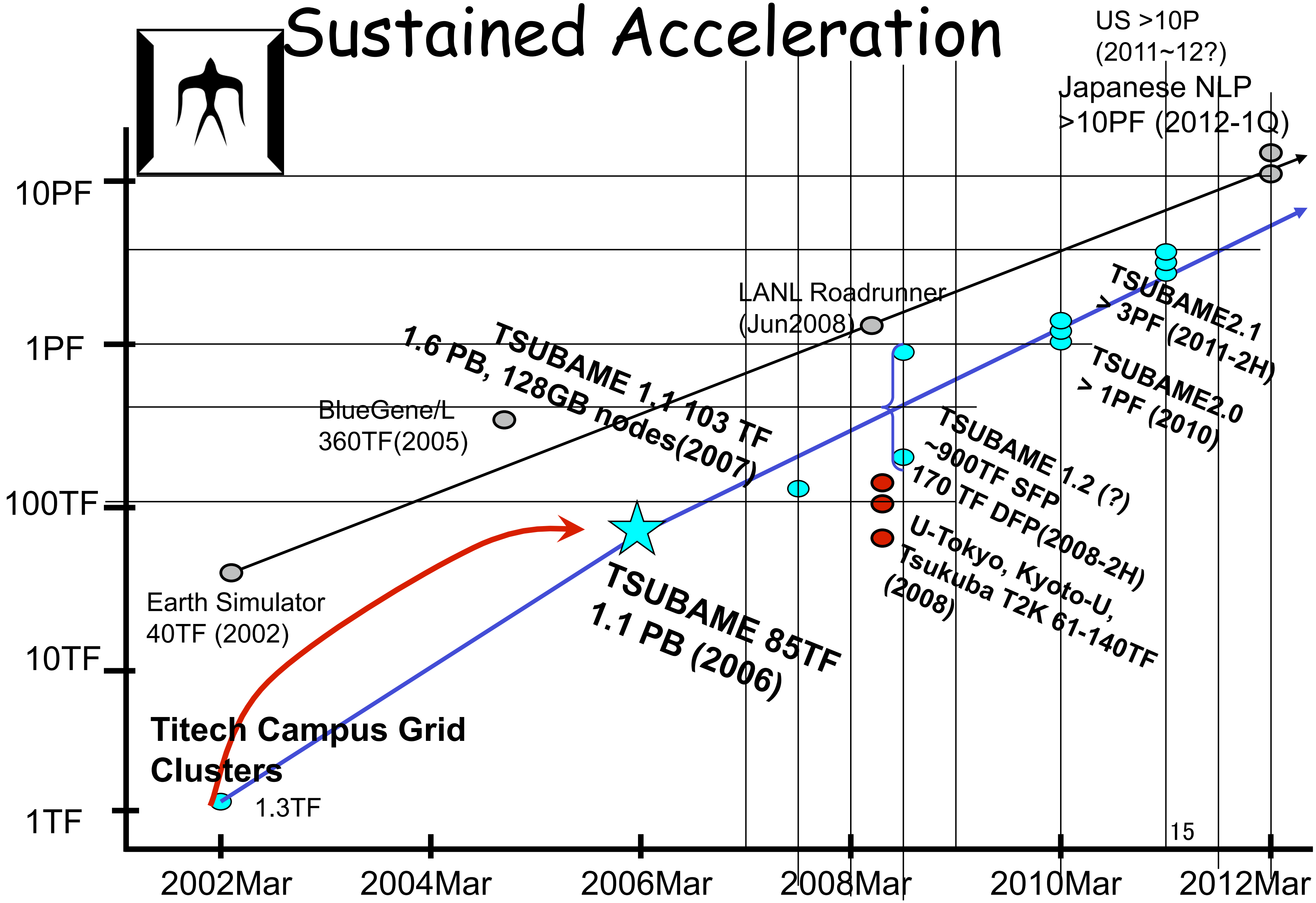
- The Next Leadership Petascale machine
  - > 10 Petaflops
  - Specialized
    - ½ NEC Vector, ½ Fujitsu SPARC derivative
    - Huge, Expensive ($1 billl)

Vs.

- Commodity efforts e.g. TSUBAME 2.0, T2K follow ons
  - The ES vs. TSUBAME 1.0 battle
  - The ES2 & Jaxa Fujitsu vs. T2K&TSUBAME 1.2
  - NLP vs. Gen 2 T2K& TSUBAME2.0?

TSUBAME Upgrades Towards Petaflops

Sustained Acceleration



Biggest Problem is Power…

Machine	CPU Cores	Watts	Peak GFLOPS	Peak MFLOPS/ Watt	Watts/ CPU Core	Ratio c.f. TSUBAME
TSUBAME(Opteron)	10480	800,000	50,400	63.00	76.34	
TSUBAME2006 (w/360CSs)	11,200	810,000	79,430	98.06	72.32	
TSUBAME2007 (w/648CSs)	11,776	820,000	102,200	124.63	69.63	1.00
Earth Simulator	5120	6,000,000	40,000	6.67	1171.88	0.05
ASCI Purple (LLNL)	12240	6,000,000	77,824	12.97	490.20	0.10
AIST Supercluster (Opteron)	3188	522,240	14400	27.57	163.81	0.22
LLNL BG/L (rack)	2048	25,000	5734.4	229.38	12.21	1.84
Next Gen BG/P (rack)	4096	30,000	16384	546.13	7.32	4.38
TSUBAME 2.0 (2010Q3/4)	160,000	810,000	1,024,000	1264.20	5.06	10.14

TSUBAME 2.0 x24 improvement in 4.5 years…? ➔ ~ x1000 over 10 years

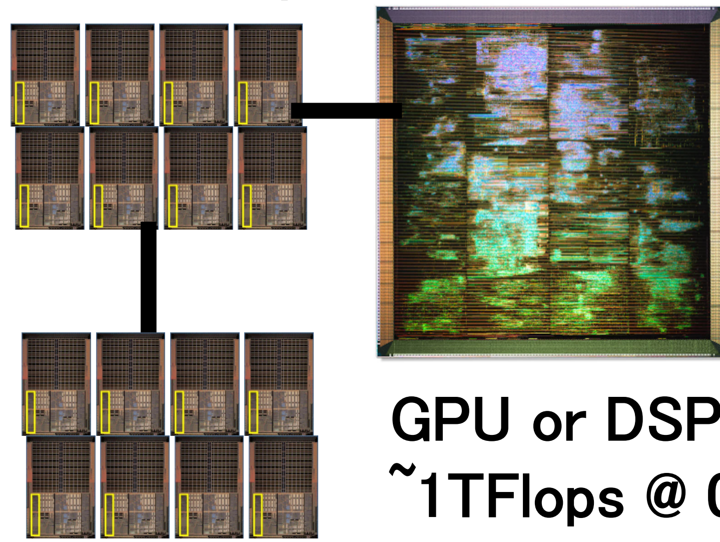
Circa 2004 My Prediction for a Petaflops Machine in 2004 (as TSUBAME was being designed)

"Future AV Vector-Parallel"

IBM/Toshiba/SONY Cell (Chip Vector) + SMT/CMT 256GF-1TFlops (@0.065-0.03)

OR

"Future PC Vector-Graphics"



GPU or DSP, ~1TFlops @ 0.045

Multicore SMT/CMT, ~50Gflops@0.065 μ

100CPUs/Rack => A 100 Teraflops per Rack

GPUs as Commodity Vector Engines---True Rebirth of Vectors

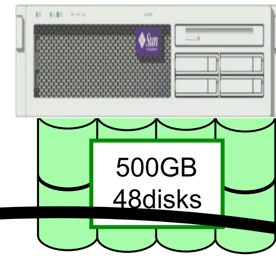
- E.g., NVIDIA Tesla, AMD Firestream
  - High Peak Performance 1TFlops
    - Good for tightly coupled code e.g. Nbody
  - High Memory bandwidth (>100GB/s)
    - Good for sparse codes e.g. CFD
    - High 3DFFT performance (>100 GFlops) due primarily to memory bandwidth
  - **Looks very much like classic vector machines**
    - Many many registers, small cache, abundunt multithread ~= long vectors
  - Restrictions: Limited non-stream memory access, PCI-express overhead, etc.

How do we exploit them given vector computing experiences?

TSUBAME 1.2 Evolution (Oct. 2008)
World's first GPU Accelerated SC

Voltaire ISR9288 Infiniband x8
10Gbps x2 ~1310+50 Ports
~13.5Terabits/s
(3Tbits bisection)

NEC SX-8i



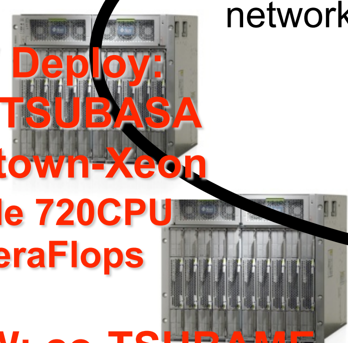
Storage
1.5 Petabyte (Sun x4500 x 60)
0.1Petabyte (NEC iStore)
Lustre FS, NFS, CIF, WebDAV (over IP)
60GB/s aggregate I/O BW

10Gbps+External NW

Unified Infiniband network

10,000 CPU Cores
300,000 SIMD Cores
~900TFlops-SFP,
~170TFlops-DFP
80TB/s Mem BW (x2 ES)

Sun x4600 (16 Opteron Cores)
32~128 GBytes/Node
10480core/655Nodes
21.4TeraBytes
50.4TeraFlops
OS Linux (SuSE 9, 10)
NAREGI Grid MW



NEW Deploy:
GCOE TSUBASA
Harpertown-Xeon
90Node 720CPU
8.2TeraFlops

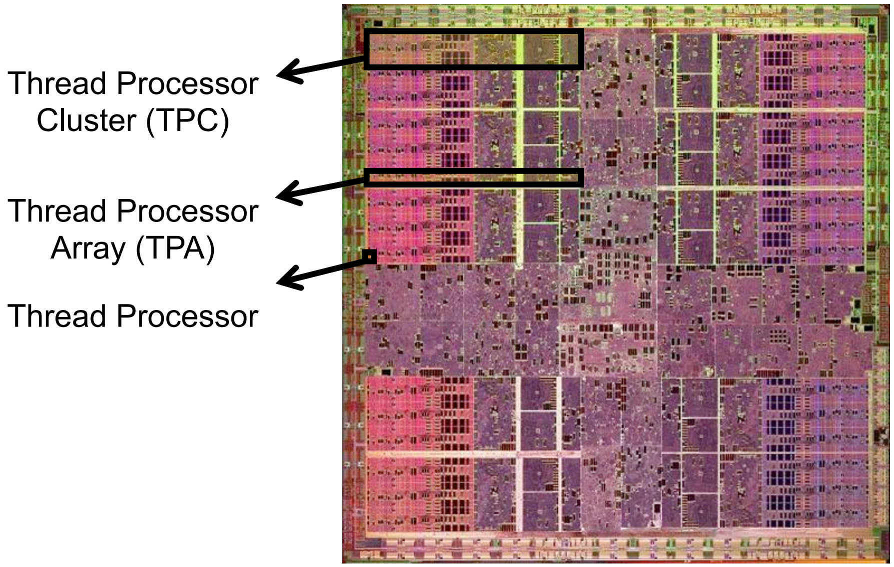
NEW: co-TSUBAME
90Node 720CPU (Low Power)
~7.2TeraFlops

PCI-e

ClearSpeed CSX600
SIMD accelerator
360 648 boards,
35 52.2TeraFlops

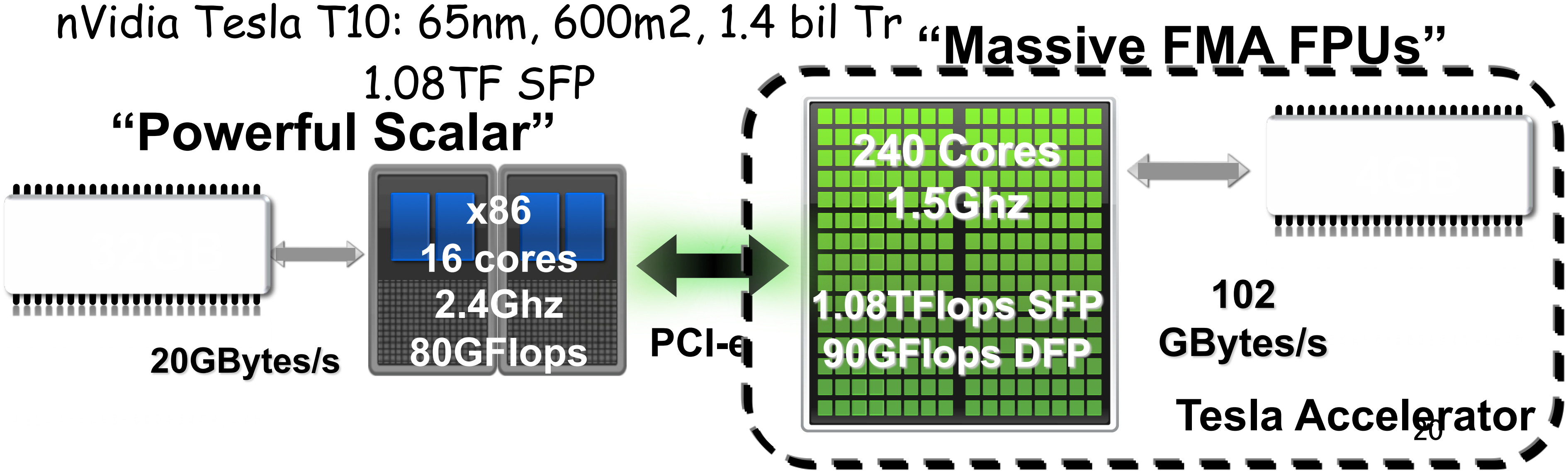
Nvidia Tesla T10P-one card per node, ~680 cards
High Performance in Many BW-Intensive Apps
10% power increase over TSUBAME 1.0 (130TF SFP / 80TF DFP)

But wait, we now have this in commodity…the GPUs (Tesla, FireStream, Larrabee, ClearSpeed)



65~55nm(2008)
=> 15 nm (2016)
x20 transitors (30 bil)
20TF FMA SFP
10TF FMA DFP

nVidia Tesla T10: 65nm, 600m2, 1.4 bil Tr
1.08TF SFP



680 Unit Tesla Installation…
While TSUBAME in Production Service (!)

Historical 10 year Parallel---Commodity x86 Clusters vs. GPU Clusters

- The 1st HPC GPU Cluster-2004 Stony Brook-U
  - Zhe Fan et. al. "GPU Cluster for High Performance Computing", SC2004
  - 32-node Xeon 2.4Ghz + nVidia GeForce FX5800
- The 1st HPC GPU Cluster-2008 TSUBAME 1.2

	X86 Cluster	GPU Cluster
1st Cluster	1993-4 (Wigraf@NASA)	2004 (Stony Brook)
1st Top500	1999 (Bonne Parnass2)	2008 (TSUBAME)
1st Tera/Peta	2004 (LLNL MCR)	??? 2010-2011
1st Peta/Exa	2008 (Ranger (1/2))	??? 2014-16

Extrapolating to Exascale

- 100MW power capacity => 1TFlop / 100W
- nVidia Tesla 10p@55nm is 1TFlop SFP/ ~170W incl. 4GB memory circa 2008…
  - 1-2TF DFP @ 100W w/8-16GB in 2012-13@22nm
- 10KW/m^2 power density => 10,000m^2
  - Save cooling energy via ambient cooling, PUE < 1.2
  - Various power optimization innovations
- Network design to stay within 25% of system power and cost
  - 10TF Nodes => 100,000 nodes, hard to build full fattree, bisection BW will suffer greatly

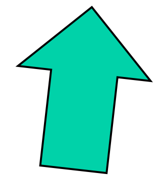
In fact we can build a Exaflop Cloud SC in 2013(!)

- @Tokyo---One of the Largest IDC in the World (in Toyosu, Tokyo... Built in 2003)
- Can fit a 10PF easy, 1 Exaflop in 2013
- On top of a 55KV/6GW Substation
- 150m diameter, 140,000 m2 IDC floorspace (x40 ES)
- 70+70 MW = 140MW power
- Can fit both Google/MS IDC or Any DOE center
- Remember interconnect cost 25% at most
- And can run Linux, Cloud/Grid interfaces, and HPC languages for accelerated & hybrid programming…(and tools that Jeff mentioned, which are all important)
- **Merger of "SC Centers" & "Cloud"**

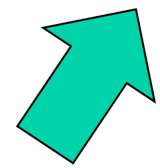
Future Architecture Trends and their Applications/Algorithms

The “n^2 (component density) vs. n (I/O BW) problem “

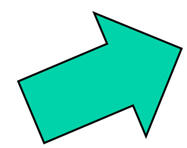
- Very Dense computation
  - Vector/SIMD/Multithreading arch.
  - Power consumption the issue



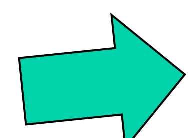
- Good absolute local memory BW
  - 1TB/s per chip soon, fast/opto signaling, 3-D packaging
  - but deepening memory hierarchy



- Relatively poor node I/O channel and NW BW
  - (only) 40Gb/100Gb soon, long distance signaling hard
  - There might be breakthrus, (e.g. planar laser diode emisson), but…

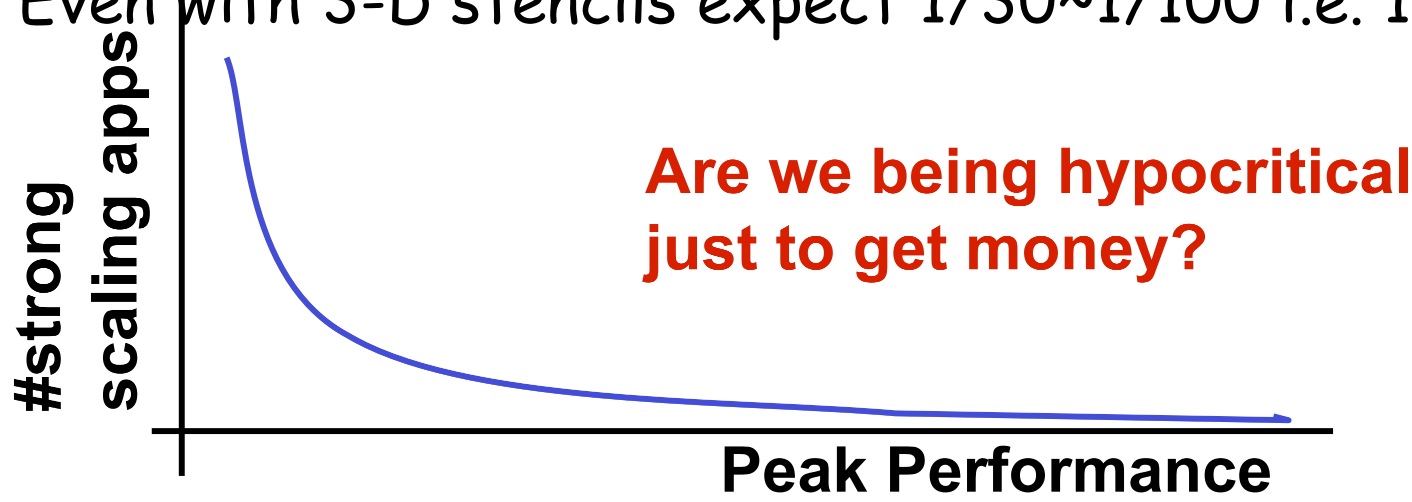


- Very poor Disk Storage BW
  - SSDs are just boosts, no exception to the laws of physics



Can we make petaflops scale to exa in "non-capacity capability app"?

- <u>Capability --- latency matters, strong scaling</u>
- ➔ requiring 1~10KB 1us messages to be efficient ➔ computation loop less than 1 us.
  => Can only tolerate 1/1000 fluctuation i.e. both loop and communicaiton will be 1ns, c.f. strong scaling code on a petaflops machine
  => Even with 3-D stencils expect 1/30~1/100 i.e. 10-30ns

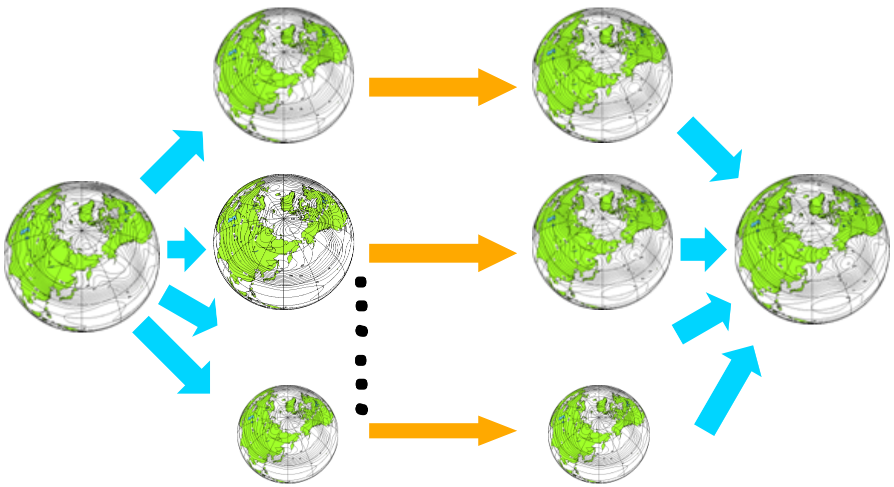


A Typical "Weak Scaling Capability App" - Capacity App in Disguise -

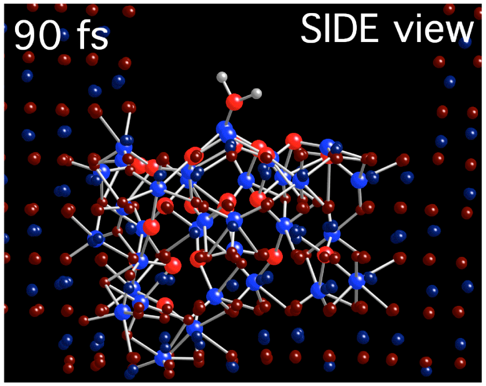
```
Initialize;
Loop until computation gets done {
  MPI_AllScatter();
  Do work within node for seconds,
  minutes, hours…;
  MPI_AllGather();
}
Finalize;
```

--- And is grossly inefficient compared to say simple workstealing parameter-sweep esp. if load is unbalanced

So the world will mostly go ensemble
--- capability at core, capacity at large ---



Barotropic S-Model Ensemble climate simulation



QM/MM Molecular Simulation

"How are we to judge sciences, in that using <u>100,000 cores in a single MPI app</u> has more scientific significance than <u>100,000 single-threaded app</u>, as they both require system scalability in the design?"

BTW, MW may need to scale better for "capacity" e.g. BQ systems

DOE SC Applications Overview
(following slides courtesy John Shalf @ LBL NERSC)

NAME	Discipline	Problem/Method	Structure
MADCAP	Cosmology	CMB Analysis	Dense Matrix
FVCAM	Climate Modeling	AGCM	3D Grid
CACTUS	Astrophysics	General Relativity	3D Grid
LBMHD	Plasma Physics	MHD	2D/3D Lattice
GTC	Magnetic Fusion	Vlasov-Poisson	Particle in Cell
PARATEC	Material Science	DFT	Fourier/Grid
SuperLU	Multi-Discipline	LU Factorization	Sparse Matrix
PMEMD	Life Sciences	Molecular Dynamics	Particle

Latency Bound vs. Bandwidth Bound?

- How large does a message have to be in order to saturate a dedicated circuit on the interconnect?
  - $N^{1/2}$ from the early days of vector computing
  - Bandwidth Delay Product in TCP

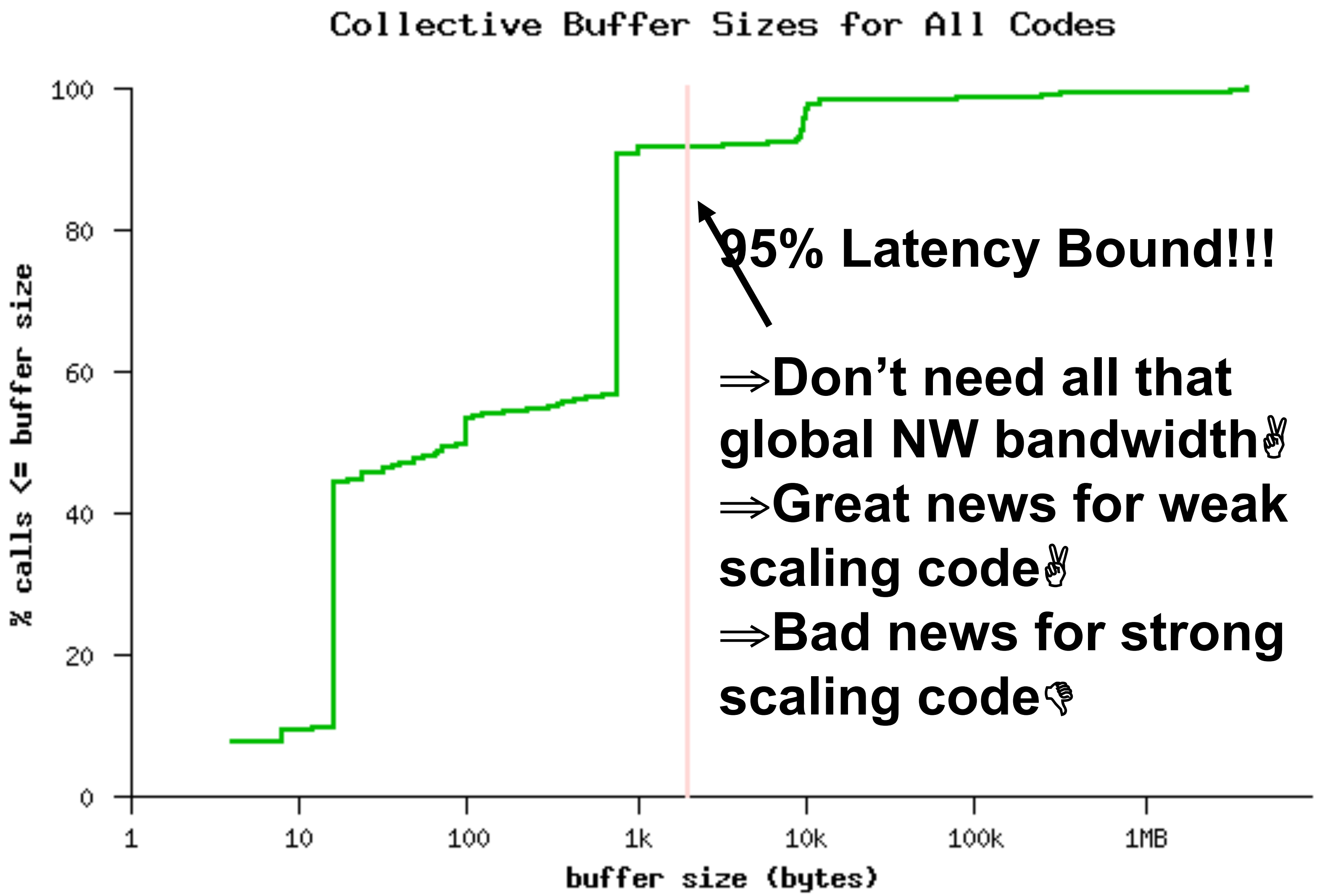
System	Technology	MPI Latency	Peak Bandwidth	Bandwidth Delay Product
SGI Altix	Numalink-4	1.1us	1.9GB/s	2KB
Cray X1	Cray Custom	7.3us	6.3GB/s	46KB
NEC ES	NEC Custom	5.6us	1.5GB/s	8.4KB
Myrinet Cluster	Myrinet 2000	5.7us	500MB/s	2.8KB
Cray XD1	RapidArray/IB4x	1.7us	2GB/s	3.4KB

- Bandwidth Bound if msg size > Bandwidth*Delay
- Latency Bound if msg size < Bandwidth*Delay
  - Except if pipelined *(unlikely with MPI due to overhead)*

- W/HW DMA a few 100ns but not much more

(Original slide courtesy John Shalf @ LBL)

Collective Buffer Sizes
- demise of metacomputing -

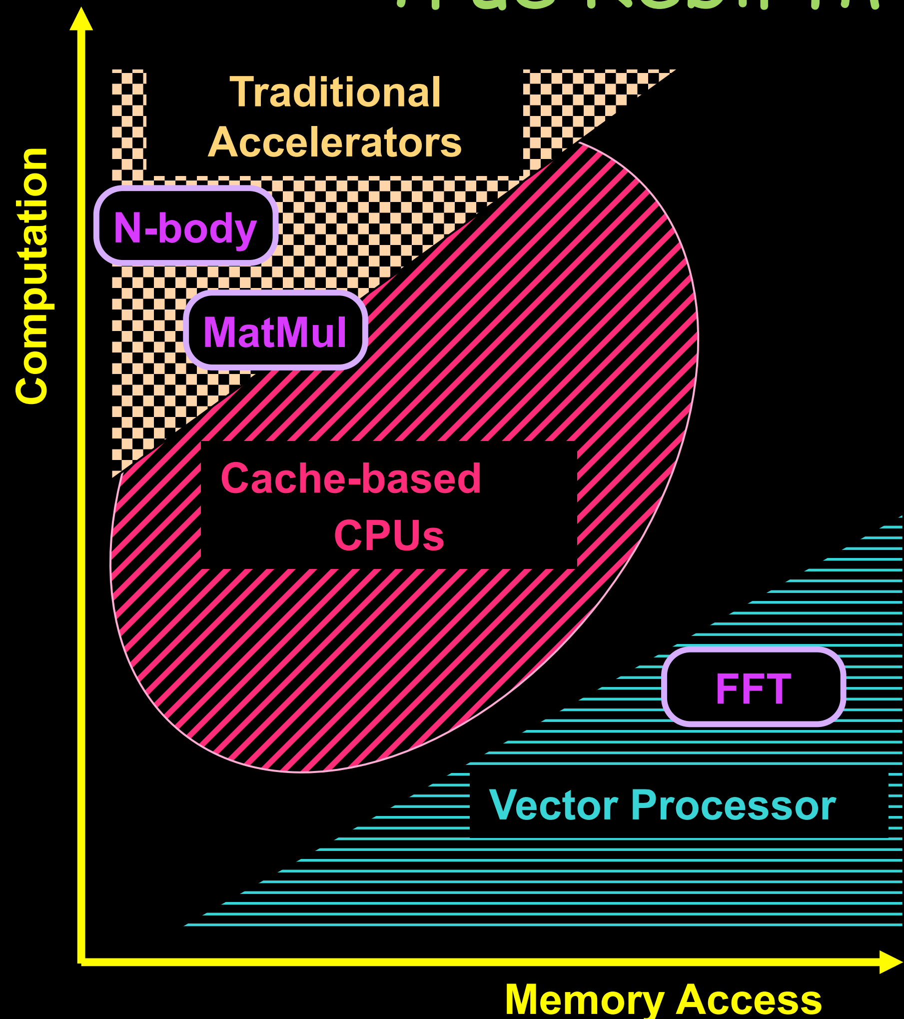


95% Latency Bound!!!

⇒Don’t need all that global NW bandwidth✌
⇒Great news for weak scaling code✌
⇒Bad news for strong scaling code👎

(Original slide courtesy John Shalf @ LBL)

GPUs as Commodity Vector Engines--- True Rebirth of Vectors



Unlike the conventional accelerators, GPUs have high memory bandwidth.

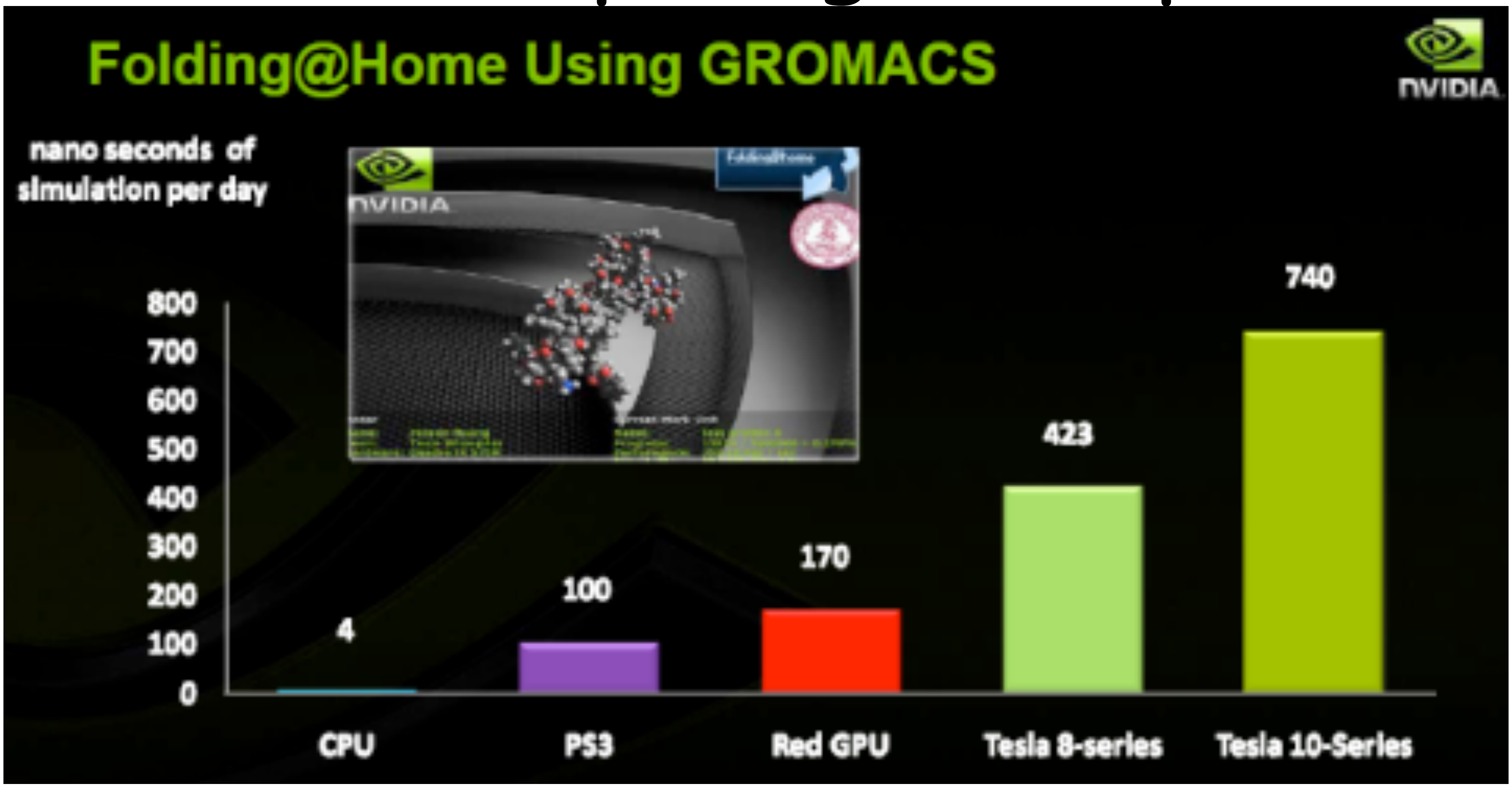
Since latest high-end GPUs support double precision, GPUs also work as commodity vector processors.

The target application area for GPUs is very wide.

Restrictions: Limited non-stream memory access, PCI-express overhead, etc.

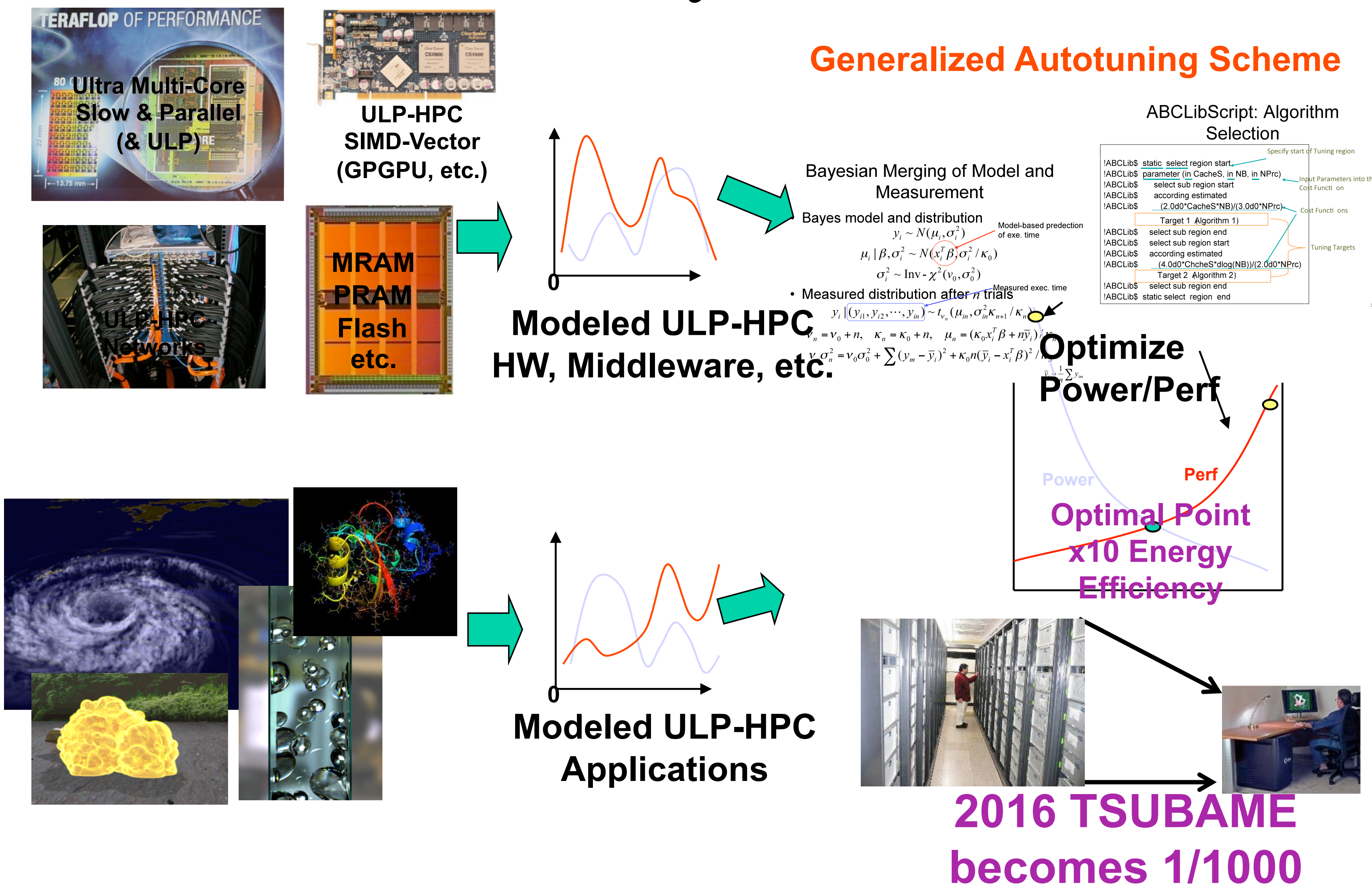
➔ How do we utilize them easily?

Dense Computing Example on



Fantastic but obvious speedups

The new JST-CREST "Ultra Low Power (ULP)-HPC" Project 2007-2012

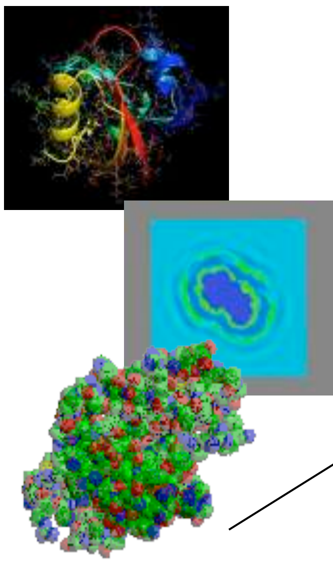


Microsoft TCI HPC-GPGPU Project

(work w/MS Research)

Research Focus

Advanced Bioinformatics/ Proteomics



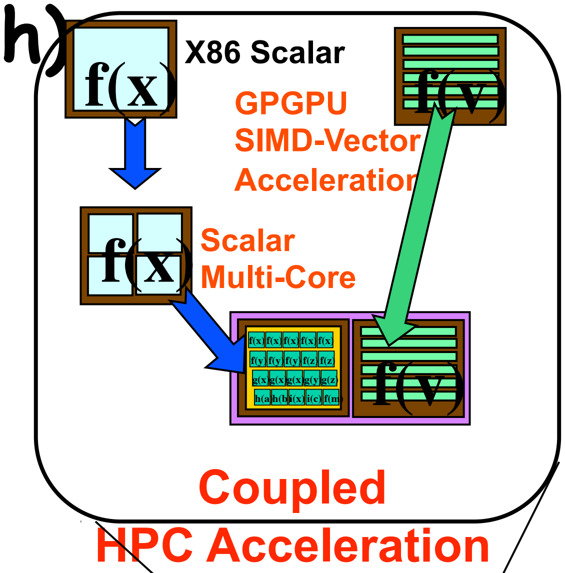
Need x1000 acceleration over standard PCs

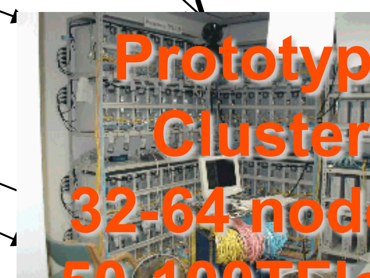
Bioinformatics Acceleration e.g., 3-D All-to-All Protein Docking

GPGPU-CPU

Hybrid Massively Parallel "Adaptive" Solvers + GPGPU FFT and other Acceleration Kernels

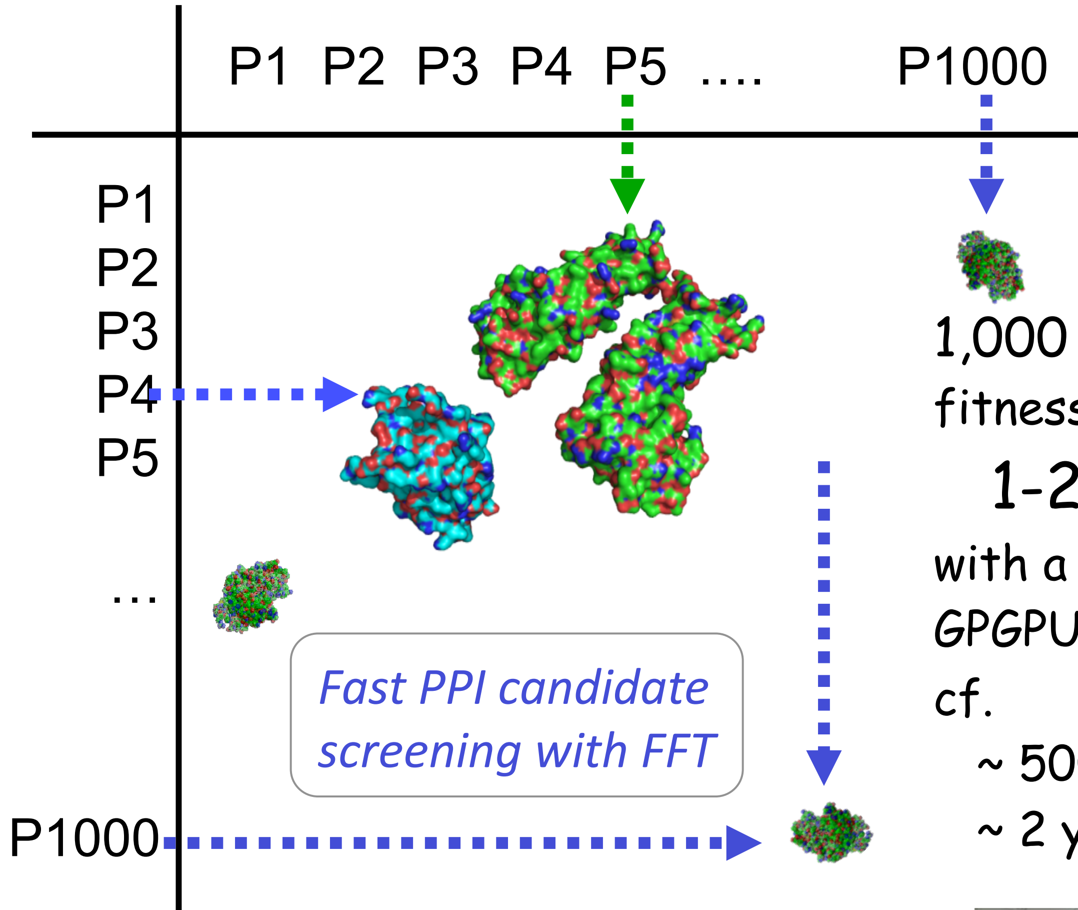
- Improving GPGPU Programmability w/ Library/Languages e.g. MS Accelerator
- High Dependability w/ large-scale GPGPU Cluster
- Model-based GPGPU-CPU Load Balancing





Towards Next Gen Petascale Personal Clusters and Desksides

All-to-all 3-D Protein Docking Challenge



1,000 x1,000 all-to-all docking fitness evaluation will take only

1-2 months (15 deg. pitch)

with a 32-node HPC-GPGPU cluster (128 GPGPUs).

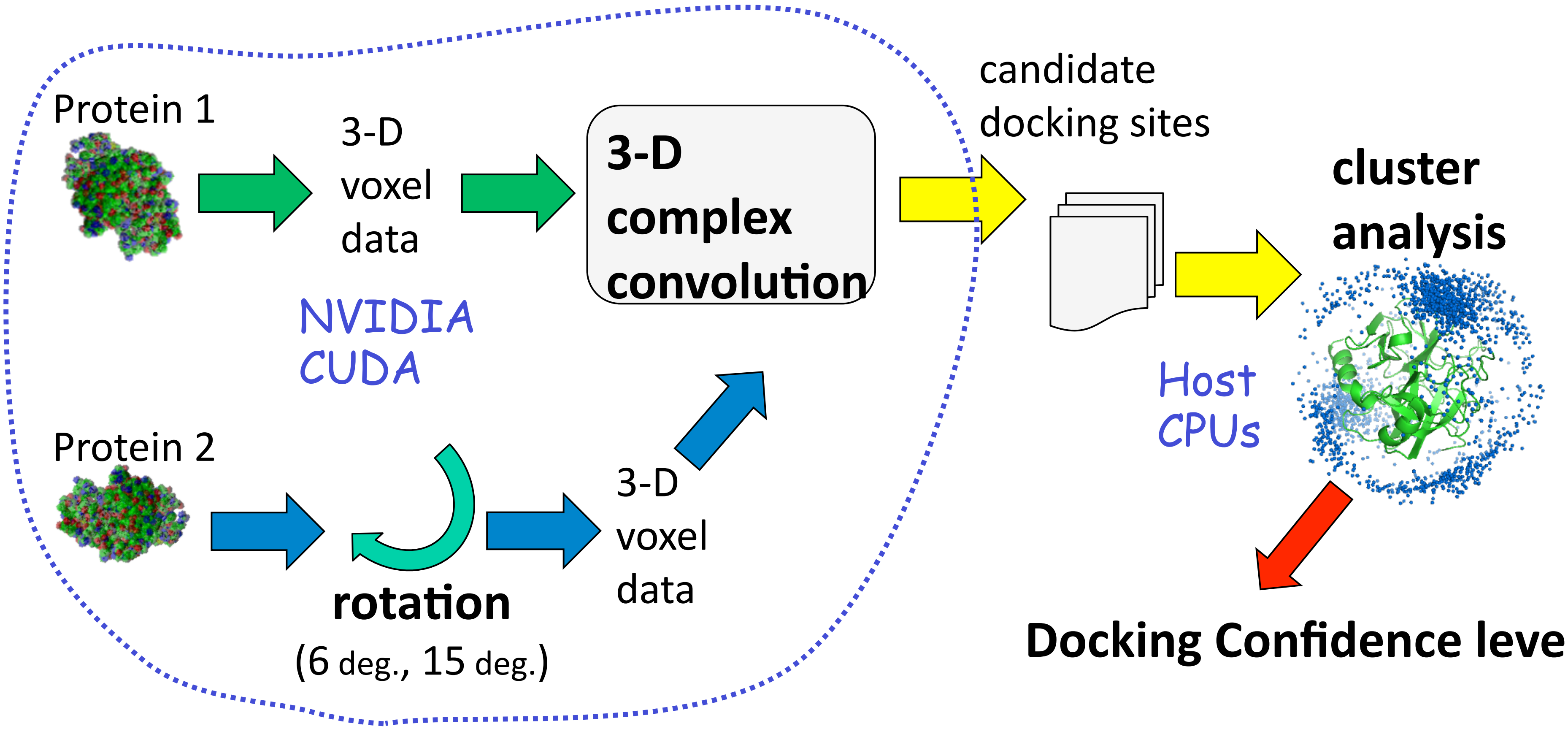
cf.

~ 500 years with single CPU (sus. 1+GF)

~ 2 years with 1-rack BlueGene/L

Blue Protein system
CBRC, AIST
(4 rack, 8192 nodes)

Calculation Flow for 3-D AA docking



Calculation for **a single protein-protein pair**: **~= 200 Tera ops.**

3-D complex convolution $O(N^3 \log N)$, *typically N = 256*

x

Possible rotations *R = 54,000 (6 deg. pitch)*

200 Exa Ops for 1000 × 1000

Bandwidth Intensive 3D-FFT for GPUs @ Tokyo Tech. [Nukada et. al., SC08]

Our 3-D FFT algorithm consists of the following two algorithms

to maximize the memory bandwidth.

(1) optimized 1-D FFTs for dimension X,

(2) *multi-row FFT* for dimension Y & Z.

The multi-row FFT computes multiple 1-D FFTs simultaneously.

Used for vector machines which provide high memory bandwidth.

Bandwidth Intensive Approach

Our 3-D FFT algorithm consists of the following two algorithms

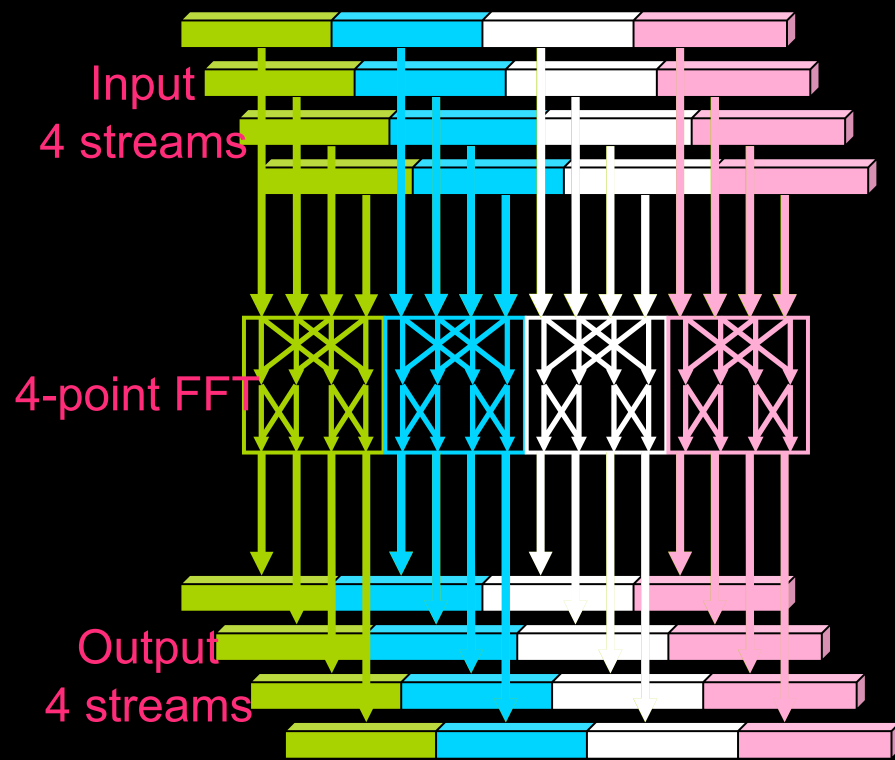
to maximize the memory bandwidth.

(1) optimized 1-D FFTs for dimension X,

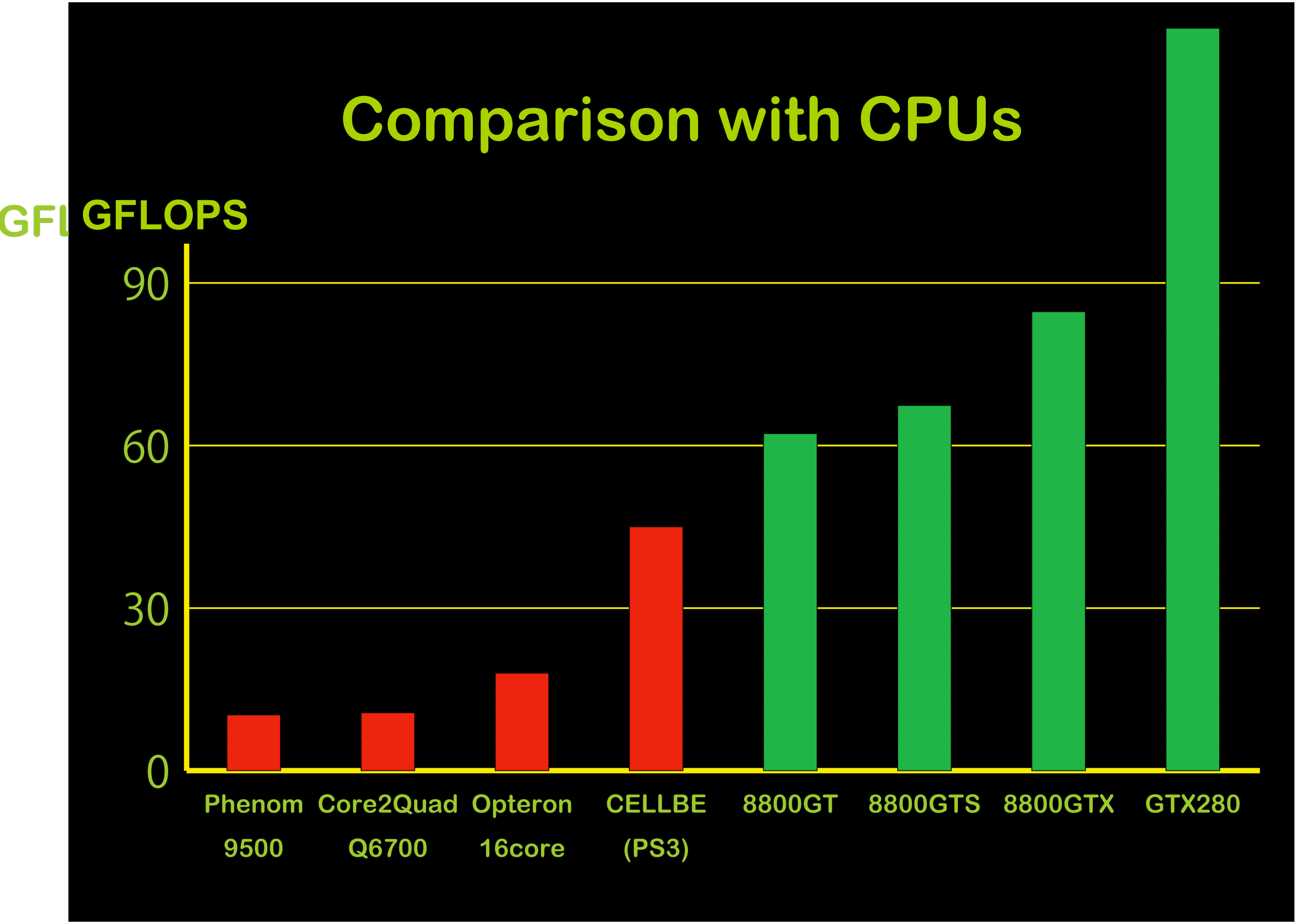
(2) *multi-row FFT* for dimension Y & Z.

The multi-row FFT computes multiple 1-D FFTs simultaneously.

Adapted from vector algorithms, assuming high memory bandwidth.

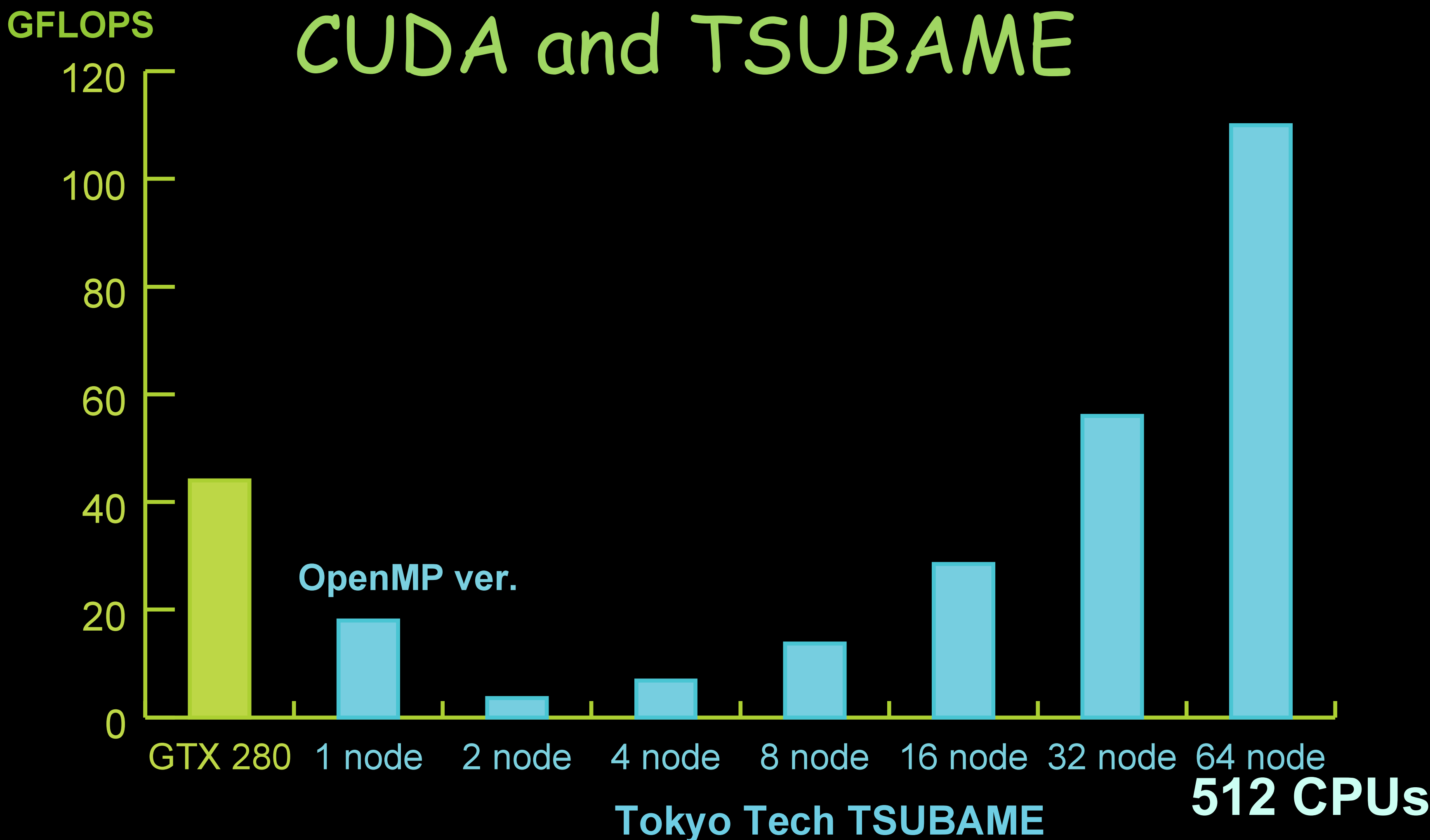


This algorithm accesses multiple streams, but each of them is successive.
Since each thread compute independent set of small FFT, thousands of registers are required
Solution: for 256-point FFT, use two- pass 16-point FFT kernels.



Comparison with CPUs
GFL
GFLOPS
90
60
30
0
Phenom 9500
Core2Quad Q6700
Opteron 16core
CELLBE (PS3)
8800GT
8800GTS
8800GTX
GTX280

Performance of DP 3-D FFT on CUDA and TSUBAME



MPI version is used for computation with multiple nodes

Performance including Data Transfer

The Worst Case:

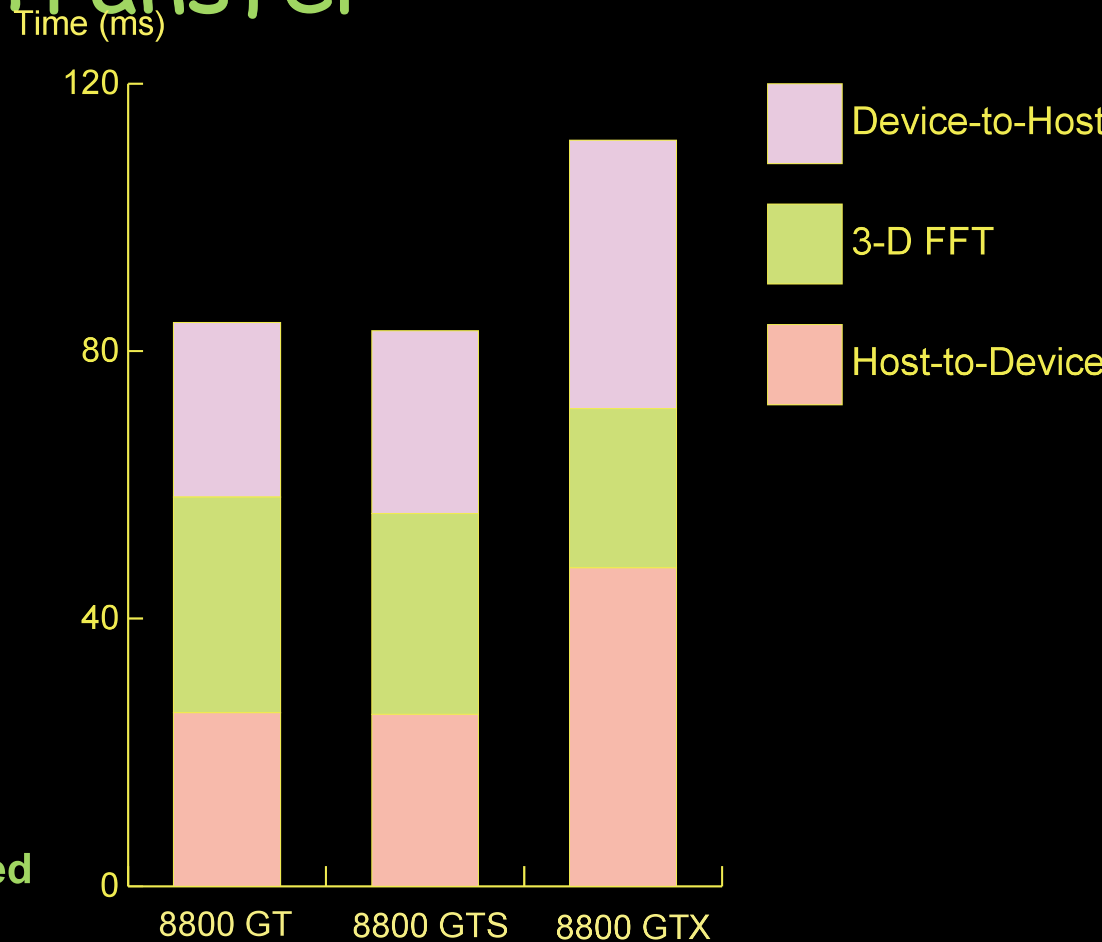
GPU computes only FFT, and CPU computes all the others.

Ex) simply replacing CPU library by GPU

Ex) data come from I/O devices

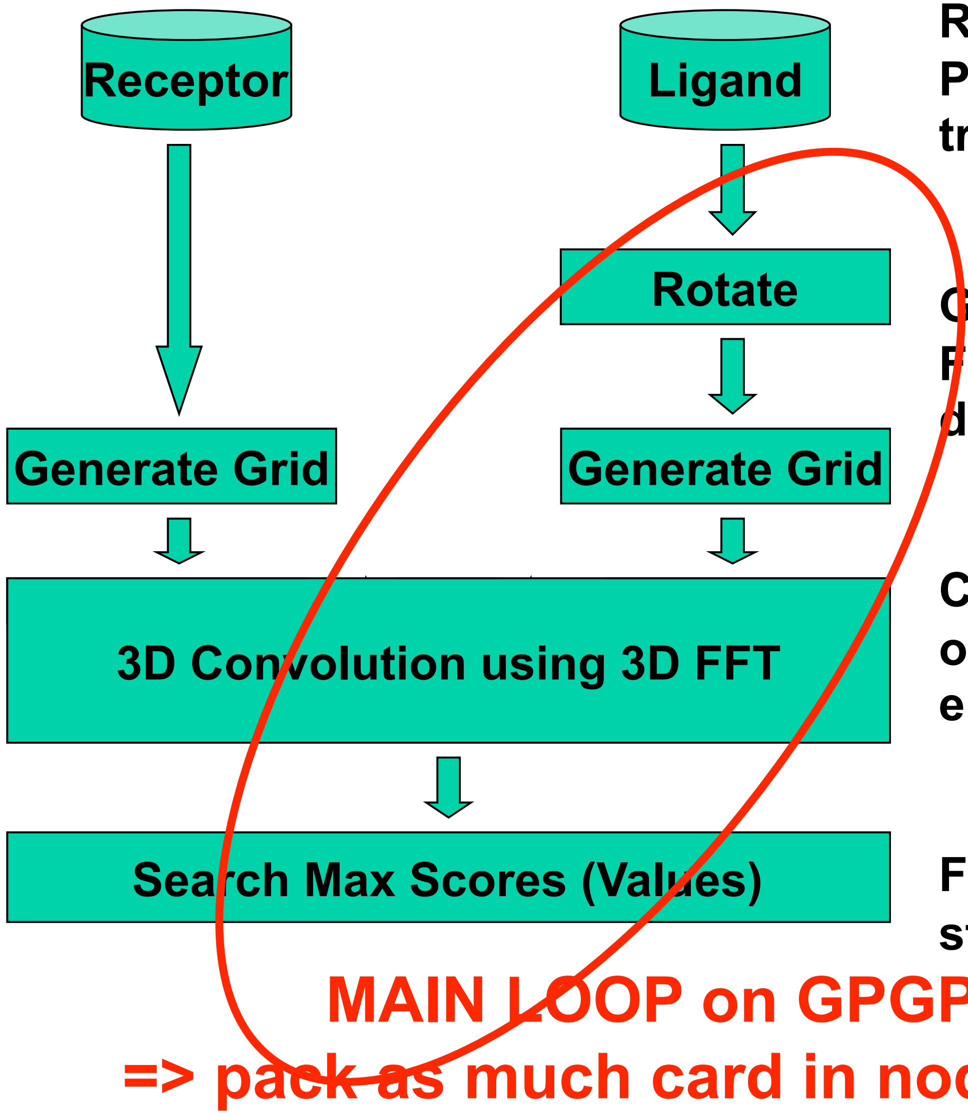
We have to transfer data between host and device using PCI-Express bus.

In the best case, the host CPU is used only to control GPUs.



only support PCI-e 1.1

Main Flow Chart



Read geometry data of atoms of Proteins from PDB, preprocess, and transfer both to the GPGPU card.

Generate 3D Grid data on the card. For ligands rotate by 6 or 15 degrees increments

Conduct two forward transforms, one backward transform, and an element-wise multiplication.

Find the best docking position with statistical clustering

Heavily Acclerated Prototype Cluster System Configuration

- 32 compute nodes
- 128 8800GTS GPGPUs
- one head node.
- Gigabit Ethernet network
- Three 40U rack cabinets.
- Windows Compute Cluster Server 2003 SP1, planned 2008 migration
- Visual Studio 2005 SP1
- nVidia CUDA 2.x

Performance Estimation for 3D PPD

Single Node

	Power (W)	Peak (GFLOPS)	3D-FFT (GFLOPS)	Docking (GFLOPS)	Nodes per 40 U rack
Blue Gene/L	20	5.6	-	1.8	1024
TSUBAME	1000 (est.)	76.8 (DP)	18.8 (DP)	26.7 (DP)	10
8800 GTS *4	570	1664	256	207	10~13

System Total ! Only CPUs for TSUBAME. DP=double precision.

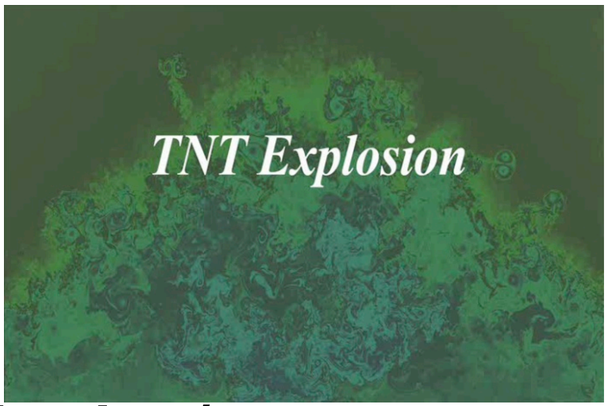
	# of nodes	Power (kW)	Peak (TFLOPS)	Docking (TFLOPS)	MFLOPS/W
Blue Gene/L (Blue Protein @ AIST)	4096 (4racks)	80	22.9	7.0	87.5
TSUBAME	655 (~70 racks)	~700	50.3 (DP)	17.5 (DP)	25
8800 GTS	32 (3racks)	18	53.2	6.5	361

BG/P vs. GTX280 would be even better for GPUs

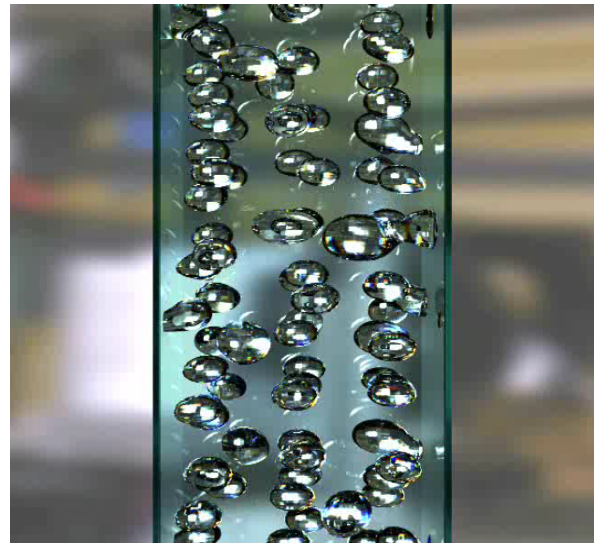
Accelerating CFD on GPUs

(Prof. Takayuki Aoki, Tokyo Tech.)

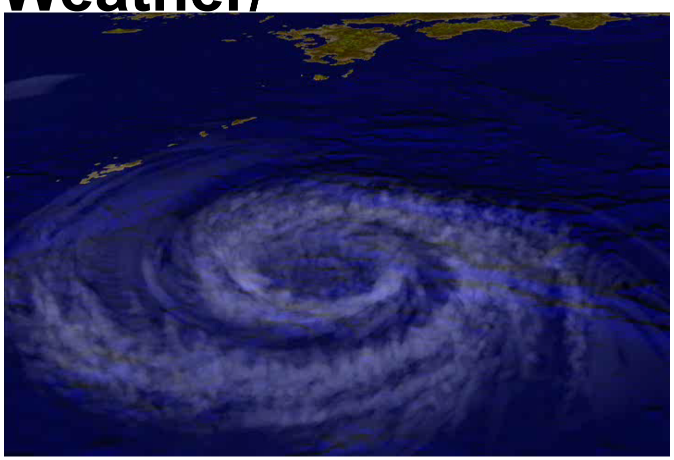
Safety



Nuclear



Weather/



Civil Engineering

Animations Courtesy Prof. Takayuki Aoki @ Tokyo Tech.

Rayleigh-Taylor Instability

Heavy fluid lays on light fluid and unstable.

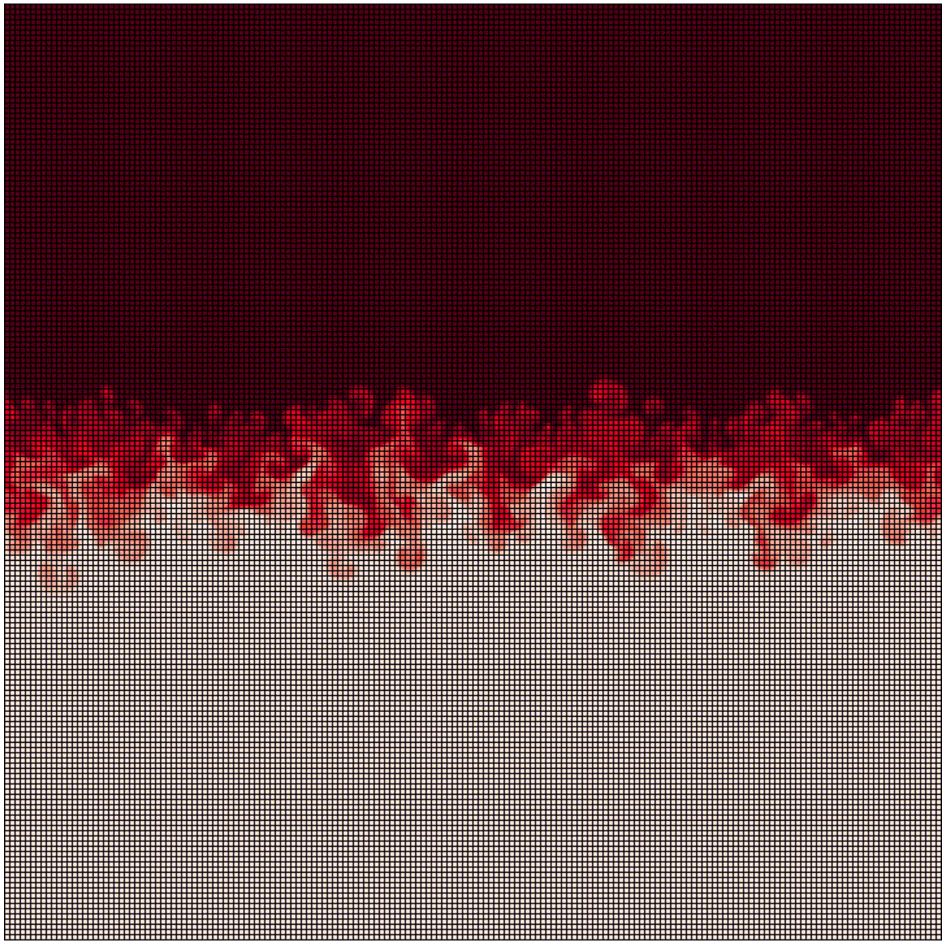
Euler equation:

$$\frac{\partial \boldsymbol{Q}}{\partial t} + \frac{\partial \boldsymbol{E}}{\partial x} + \frac{\partial \boldsymbol{F}}{\partial y} = 0$$

$$\boldsymbol{Q} = \begin{bmatrix} \rho \\ \rho u \\ \rho v \\ e \end{bmatrix} \quad \boldsymbol{E} = \begin{bmatrix} \rho u \\ \rho u^2 + p \\ \rho u v \\ eu + pu \end{bmatrix} \quad \boldsymbol{F} = \begin{bmatrix} \rho v \\ \rho u v \\ \rho v^2 + p \\ ev + pv \end{bmatrix}$$

×90

512 x 512



88 GFLOPS using GTX280

Phase Separation

Phase transition dynamics is described by the Phase Field Model.

Cahn-Hilliard equation:

$$\frac{\partial \psi}{\partial t} = L\nabla^2\left(\frac{\partial H}{\partial \psi} - C\nabla^2\psi\right)$$

H **: free energy**

$$\frac{\partial H}{\partial \psi} = v\psi - u\psi^3$$

Discretization:

$$\frac{\partial^4 \psi}{\partial x^4} = \frac{\psi_{i+2,j} - 4\psi_{i+1,j} + 6\psi_{i,j} - 4\psi_{i-1,j} + \psi_{i-2,j}}{\Delta x^4}$$

$$\frac{\partial^4 \psi}{\partial x^2 \partial y^2} = \big(\psi_{i+1,j+1} - 2\psi_{i,j+1} + \psi_{i-1,j+1}$$

$$- 2\psi_{i+1,j} + 4\psi_{i,j} - 2\psi_{i-1,j}$$

$$+ \psi_{i+1,j-1} - 2\psi_{i,j-1} + \psi_{i-1,j-1} \big)$$

3-D Computation of Phase Separation

Mixture of Oil and Water:

158 GFLOPS using GTX280

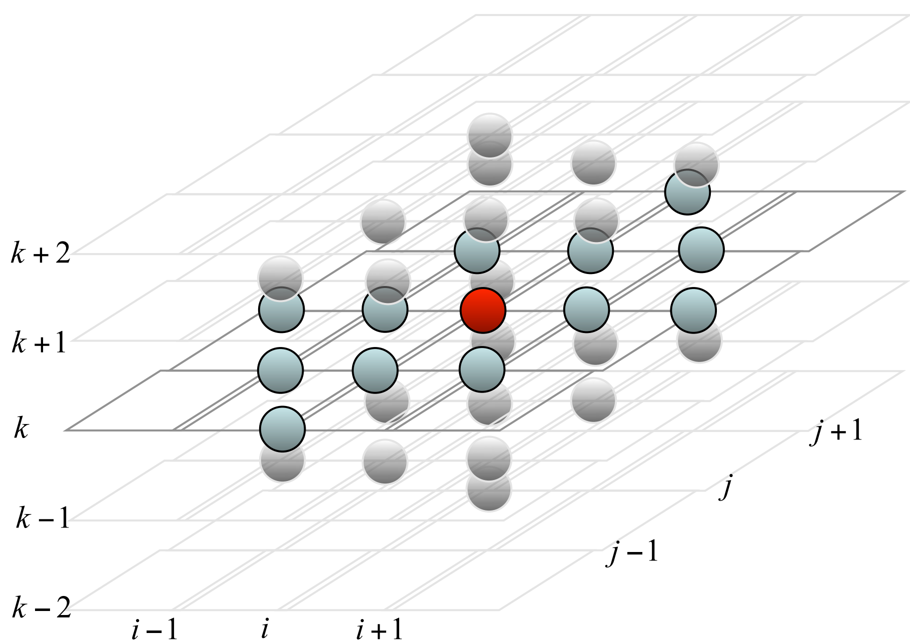
×160

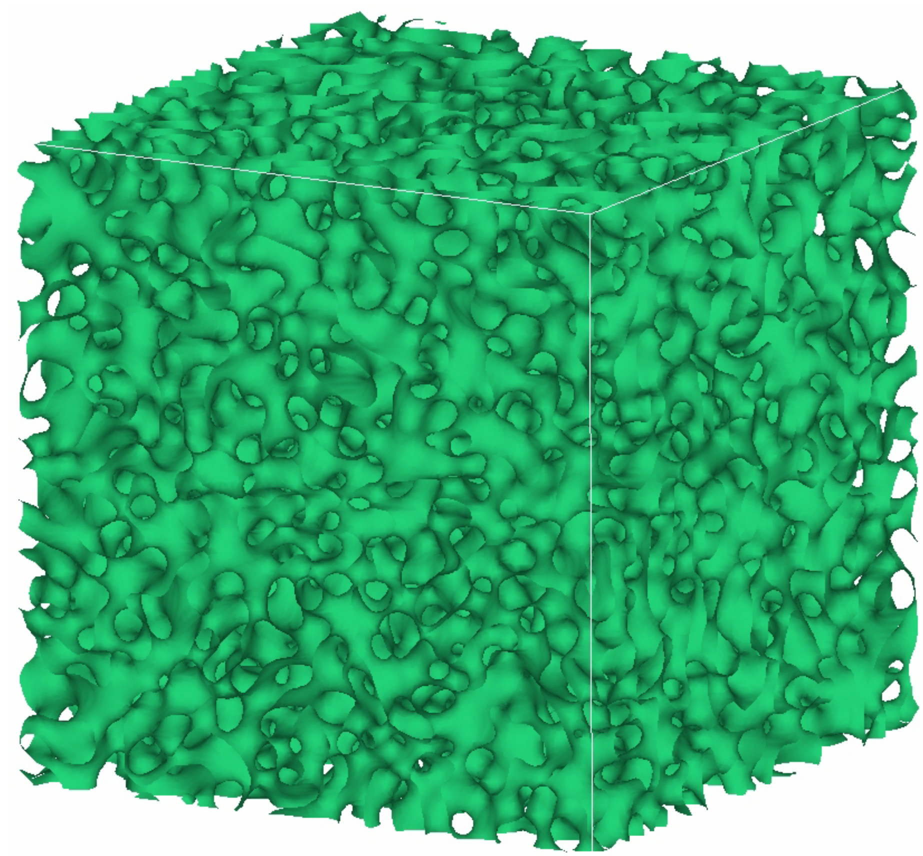
256 x 256 x 256

Used register number = 46

※ nvcc option –maxrregcount 32

for G80, 92





Real-time Tsunami Simulation

Collaboration with ADPC (Asian Disaster Preparedness Center)

Early Warning System:

Sensor Data Extrapolation → Real-time CFD

high accuracy

Shallow-Water Eq.

Conservative Form:

Assuming hydrostatic balance in the vertical direction,

3D → 2D equation

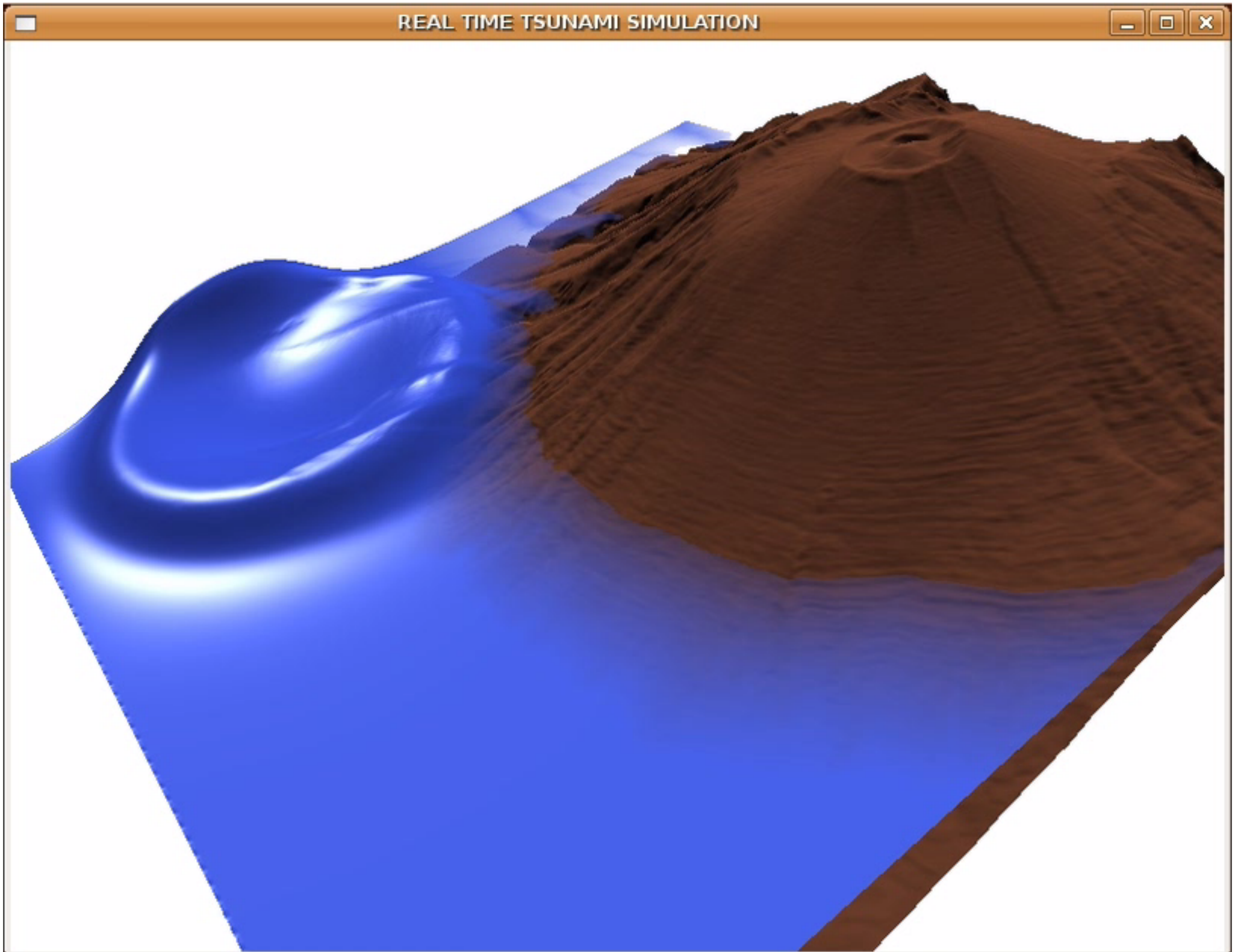
$$\frac{\partial h}{\partial t} + \frac{\partial hu}{\partial x} + \frac{\partial hv}{\partial y} = 0$$

$$\frac{\partial hu}{\partial t} + \frac{\partial}{\partial x}\left(hu^2 + \frac{1}{2}gh^2\right) + \frac{\partial huv}{\partial y} = -gh\frac{\partial z}{\partial x}$$

$$\frac{\partial hv}{\partial t} + \frac{\partial huv}{\partial x} + \frac{\partial}{\partial y}\left(hv^2 + \frac{1}{2}gh^2\right) = -gh\frac{\partial z}{\partial y}$$

Numerical Methods of Tsunami Simulation

- ■ **2-dimensional Problem : Directional-Splitting Fractional Method**
- ■ **Point Value Comp. : Characteristic-based Method using Multi-moment Interpolation**
- ■ **Integral Value Comp. : Conservative Semi-Lagrangian CIP + IDO**
- ■ **Run-up to dry area: thin water layer and artificial viscosities**


REAL TIME TSUNAMI SIMULATION

GPU Performance

Speed Comparison

x-direction : y-direction = 10 : 7

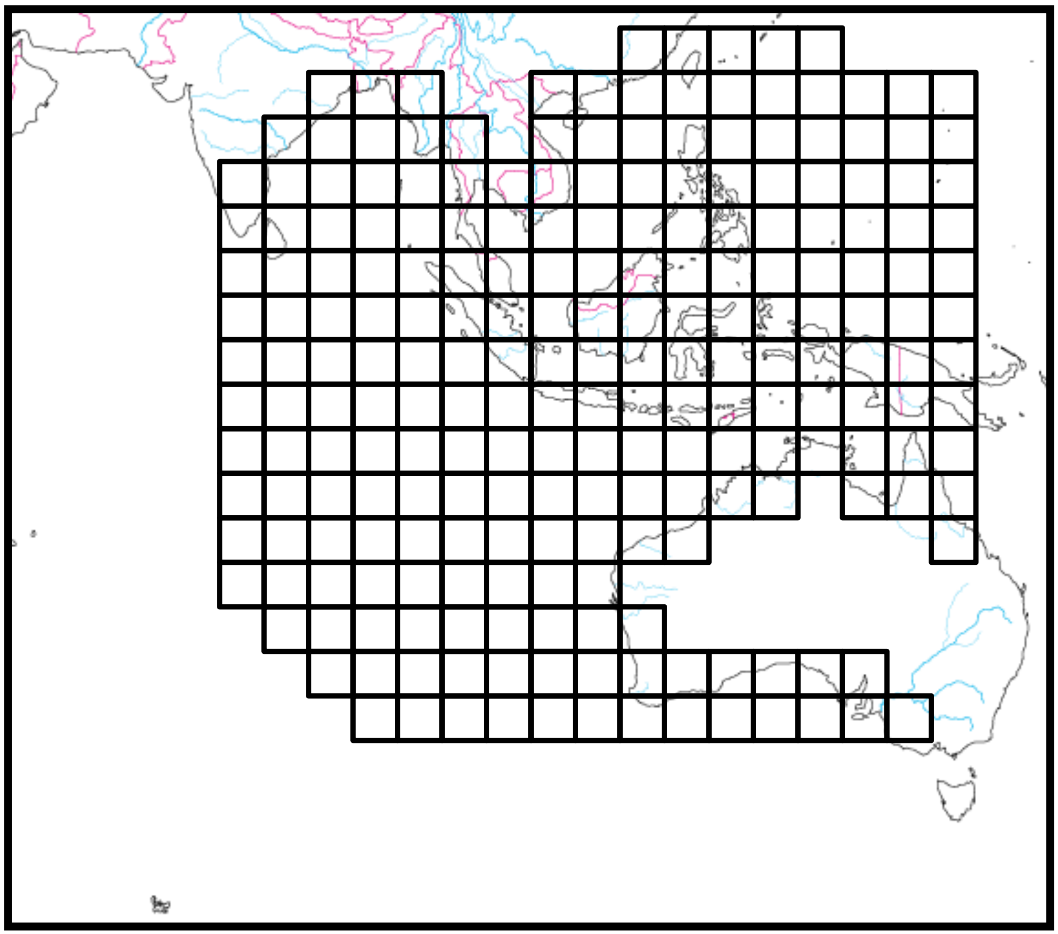
Current Speed-up

GPU : CPU = 62 : 1

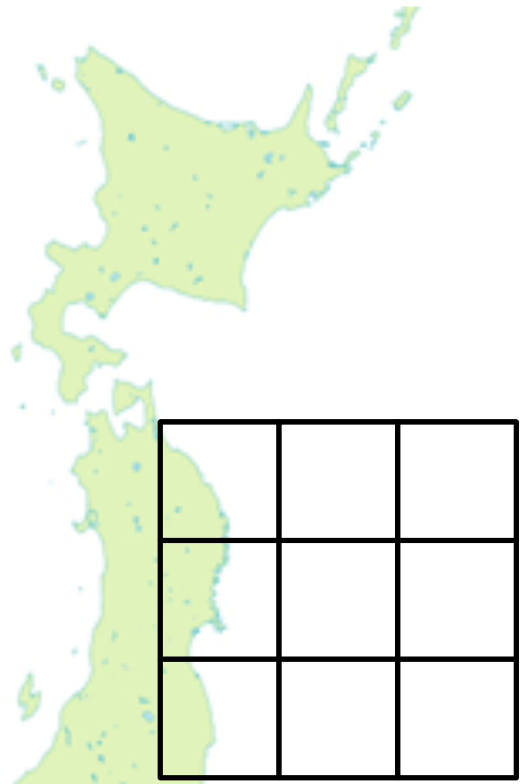
GPU – GeForce GTX280 (sp = 240, clock 1.3Ghz)

CPU – Xeon 2.4GHz 6MB Cache Memory

World-Wide Real-Time Tsunami Simulator



241 GPU 5000km×5000km（500m mesh）



9 GPU 600km×600km （100m mesh）

Multi-GPU: Riken Himeno Benchmark

(Prof. Takayuki Aoki and Akira Nukada, Tokyo Tech)

RIKEN Himeno CFD Benchmark

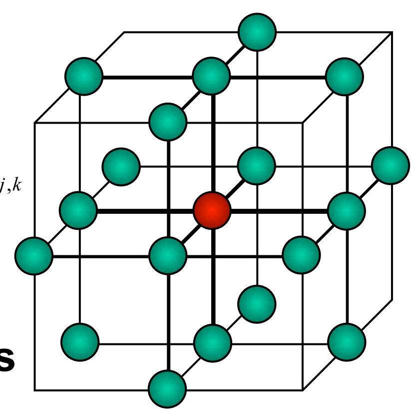
Poisson Equation: $\nabla \cdot (\nabla p) = \rho$

(Generalized coordinate)

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} + \frac{\partial^2 p}{\partial z^2} + \alpha \frac{\partial^2 p}{\partial xy} + \beta \frac{\partial^2 p}{\partial xz} + \gamma \frac{\partial^2 p}{\partial yz} = \rho$$

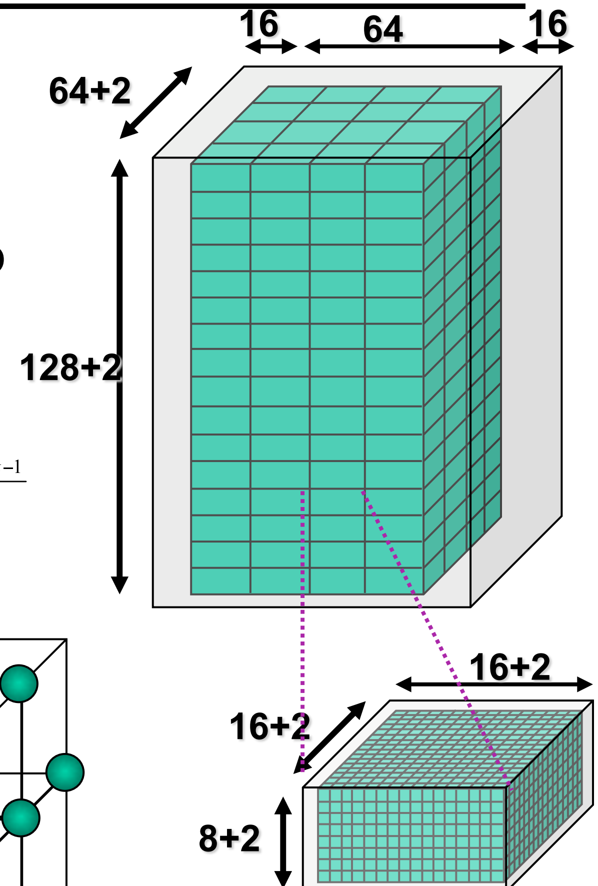
Discretized Form:

$$\frac{p_{i+1,j,k} - 2p_{i,j,k} + p_{i-1,j,k}}{\Delta x^2} + \frac{p_{i,j+1,k} - 2p_{i,j,k} + p_{i,j-1,k}}{\Delta y^2} + \frac{p_{i,j,k+1} - 2p_{i,j,k} + p_{i,j,k-1}}{\Delta y^2}$$
$$+ \alpha \frac{p_{i+1,j+1,k} - p_{i-1,j+1,k} - p_{i+1,j-1,k} + p_{i-1,j-1,k}}{4\Delta x \Delta y}$$
$$+ \beta \frac{p_{i+1,j,k+1} - p_{i-1,j,k+1} - p_{i+1,j-1,k} + p_{i-1,j,k-1}}{4\Delta x \Delta z}$$
$$+ \gamma \frac{p_{i,j+1,k+1} - p_{i,j+1,k-1} - p_{i,j-1,k-1} + p_{i,j-1,k-1}}{4\Delta y \Delta z} = \rho_{i,j,k}$$



18 neighbor point access

Himeno for CUDA



1 block = 16x16x8 compute region

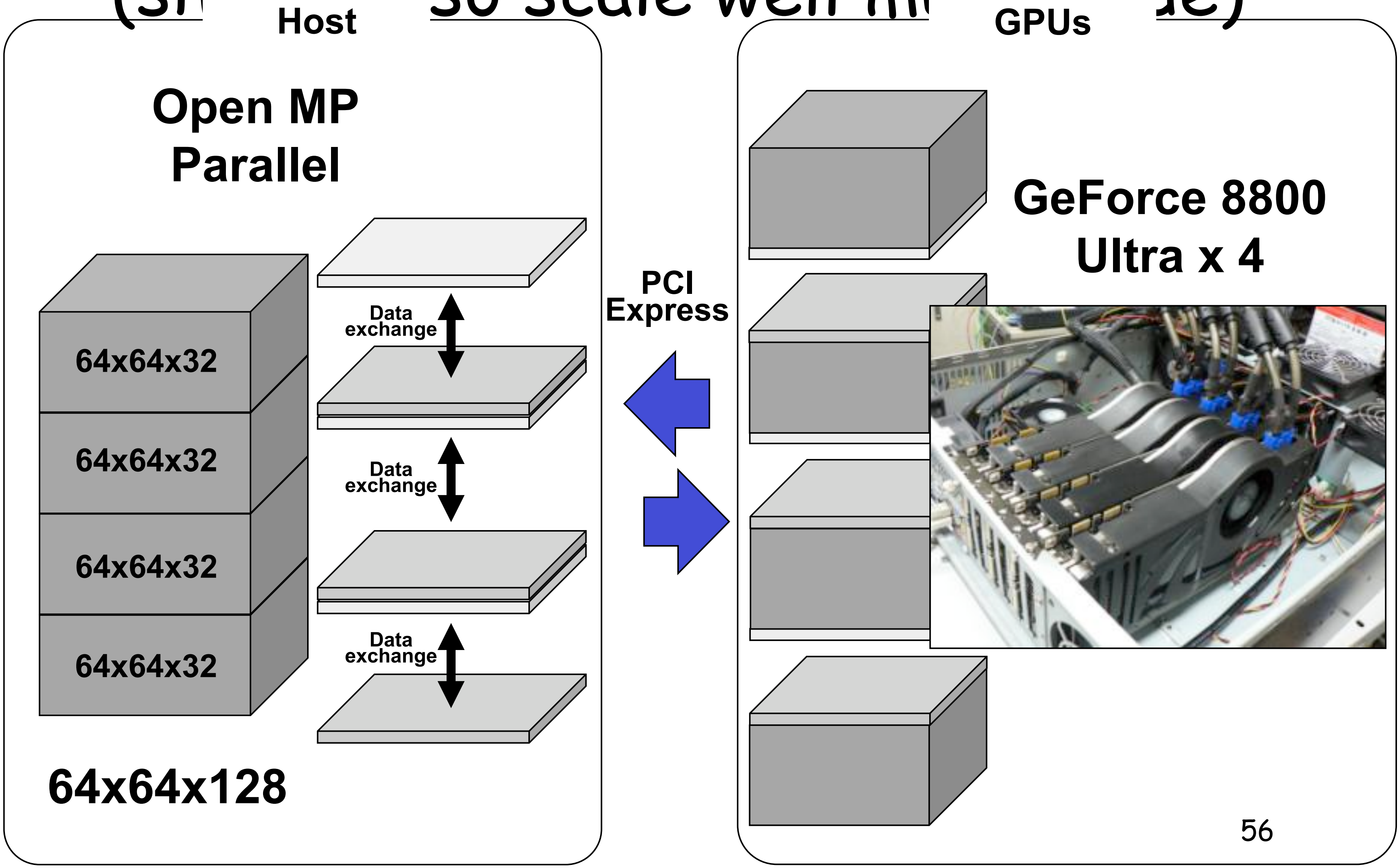
Block has 256 thread

Total 256 blocks = 65536 threads

Block shared mem =16kB

Boundary region used for transfer

4 GPU node parallelization (should also scale well multi-node)



4GPU Parallel Performance

S Model [65x65x129]

1 GPU (no data transfer) 30.6 GFLOPS (0.269sec)

2 GPU (16kB transfer) 42.5 GFLOPS (0.193sec)

4 GPU (32kB transfer) 51.9 GFLOPS (0.158sec)

x53.1 acceleration

0.976 GFLOPS (8.431sec)

Reference

M Model [129x129x257]

1 GPU (no data transfer) 29.4 GFLOPS (2.32

2 GPU (66kB transfer) 53.7 GFLOPS (1.275sec)

4 GPU (131kB transfer) 83.6 GFLOPS (0.819sec)

L Model [257x257x512]

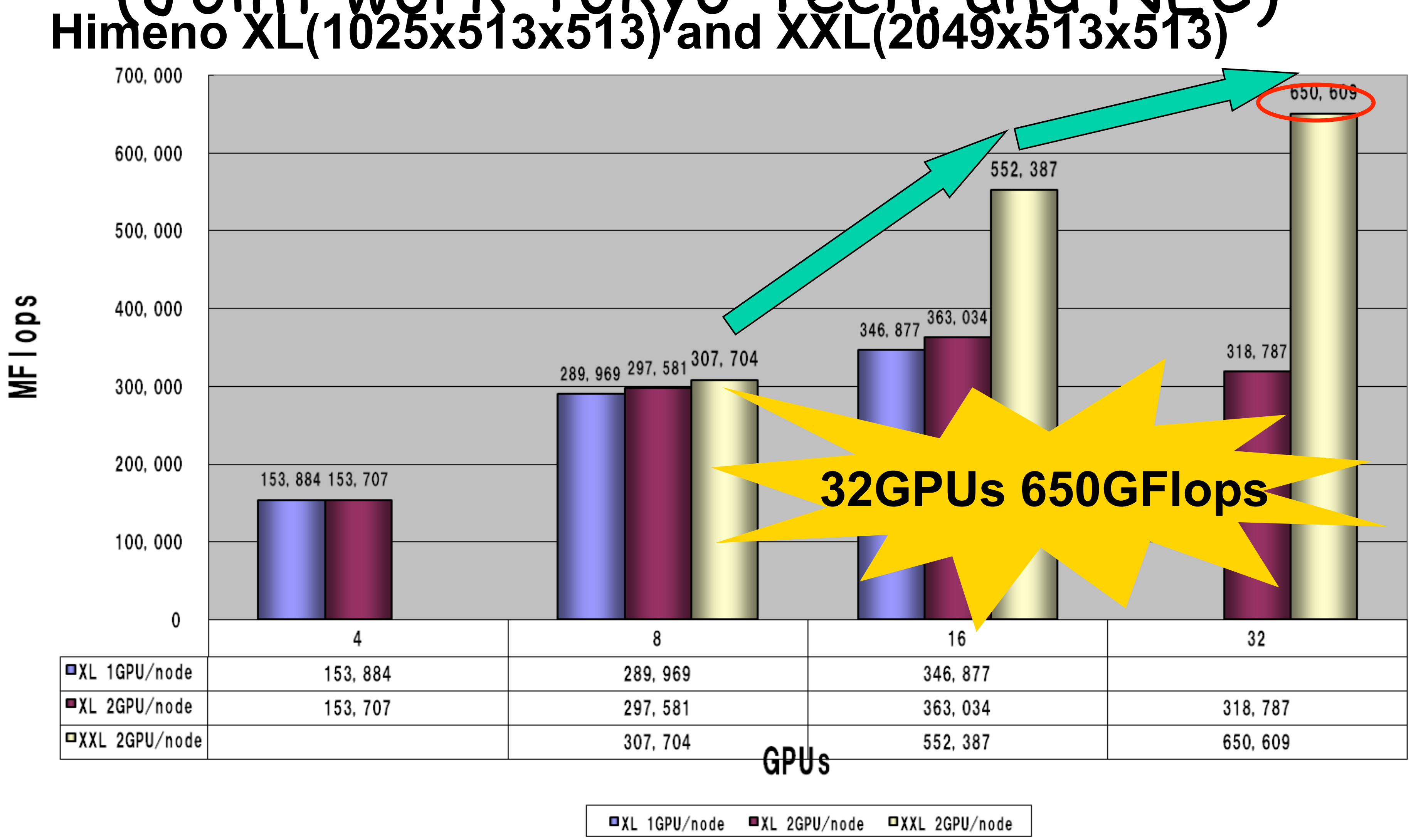
1 GPU (no data transfer) ....

2 GPU (262kB transfer) ....

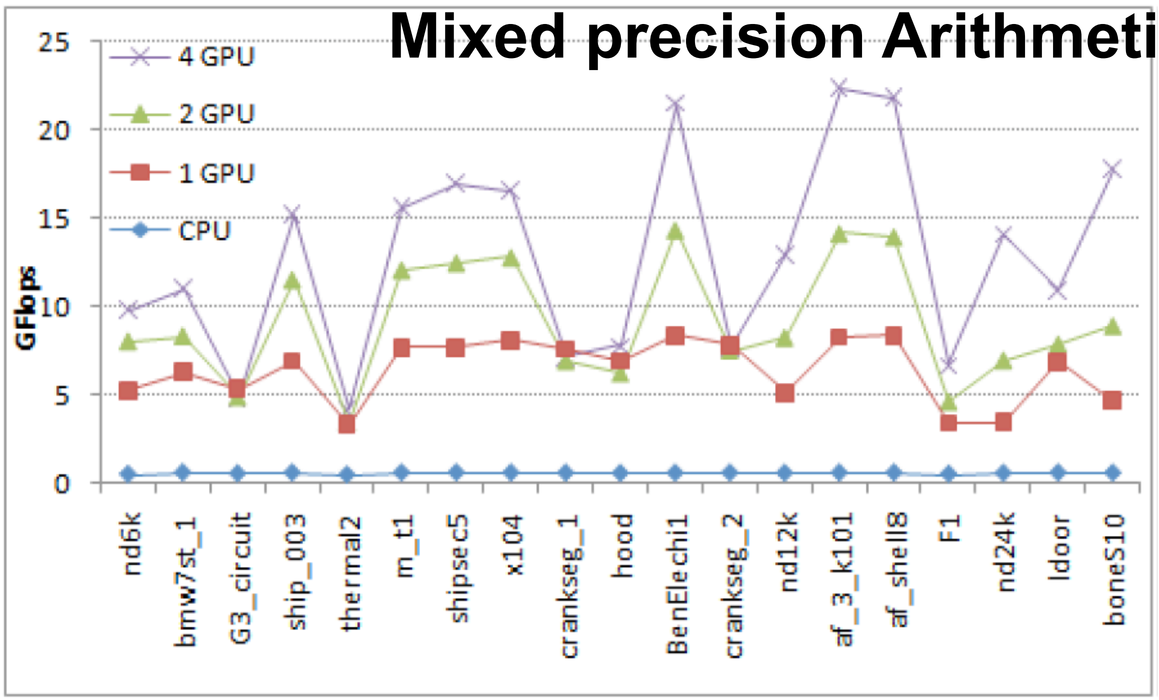
4 GPU (524kB transfer) 93.6 GFLOPS (5.974sec)

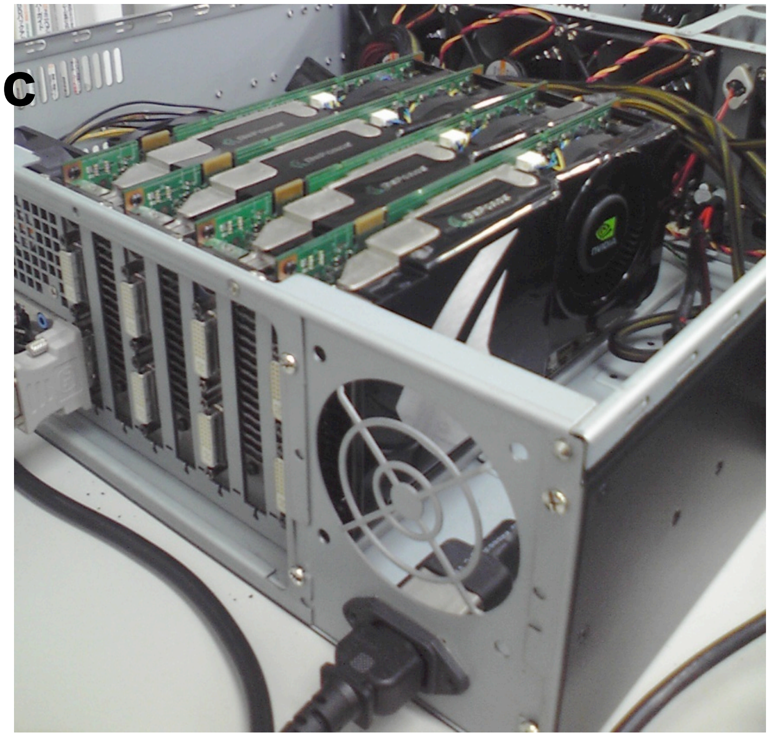
C.f. NEC SX-8 6 CPU (96GF Peak) 38.3GFLOPS Size XL

Multi-Node Himeno on TSUBAME (Joint work Tokyo Tech. and NEC)



Multi-GPU Parallel Sparse CG Solver [Cevahir et. Al. ICCS09]





14.5GFlops 4 GPUs (nVidia 8800 GTS) vs.
0.54GFlops 4 Core CPU (Phenom 2.5Ghz, DDR2-800)
Double Precision FP using mixed precision technique
(Sparse Matrix collection from UFlorida (size 1,440 to 1,585,478)

Portfolio of Tokyo Tech. GPU Computing Base Technologies for HPC & eScience

- TSUBAME 1.2 (680 Teslas) & 2.0
- Kernels (FFT, Dense/Sparse Matrix)
- Parallel Algorithms (Large FFT, LINPACK, CG)

- Task & Resource Mgmt (Heterogeneity, Scheduling, BQ Scheduling, etc.)
- Fault Tolerance (ECC, redundant computation, GPU checkpointing)
- Languages (OpenMP on GPU, Accelerator, MP)
- GPU Low Power computing (power modeling, measurement, optimization)

GPU Leadership from research to deployment(!)

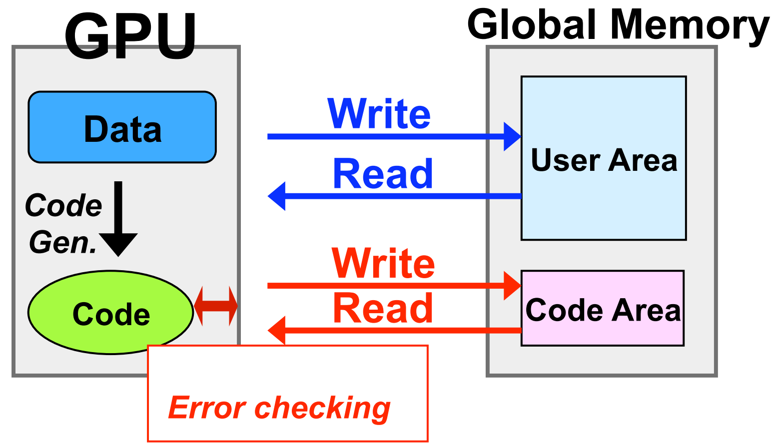
Software-Based ECC for GPUs (N. Maruyama)

Possible Collab. w/MSR Vivian Sewelsen and HPC Cluster

GPU computing reliabiliy

No ECCs on GPUs yet => Bit flip errors?

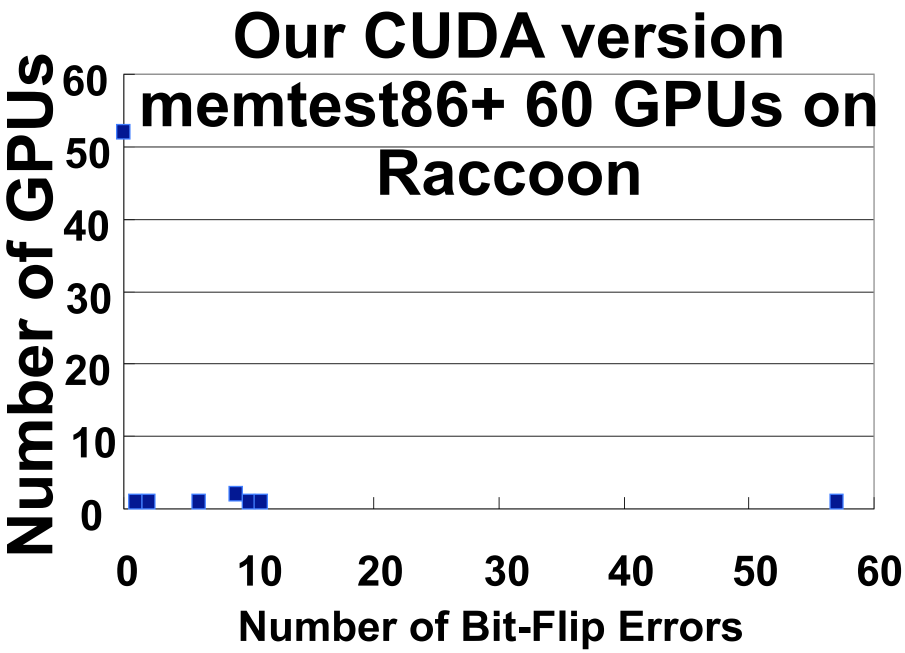
Large scale cluste quite problematic

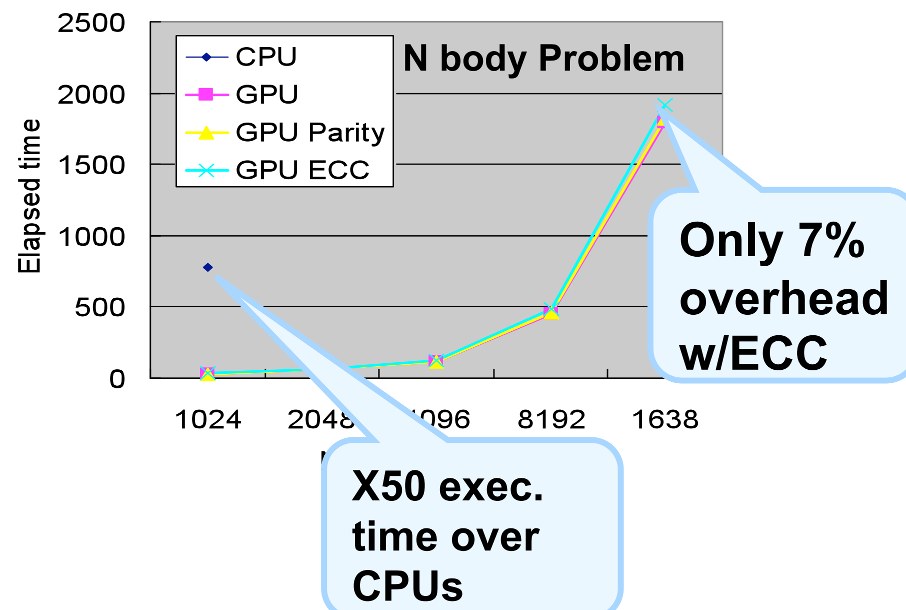


Software-based ECC on GPUs

Read : Read ECC data and check

Write : Generate ECC and store alongside data





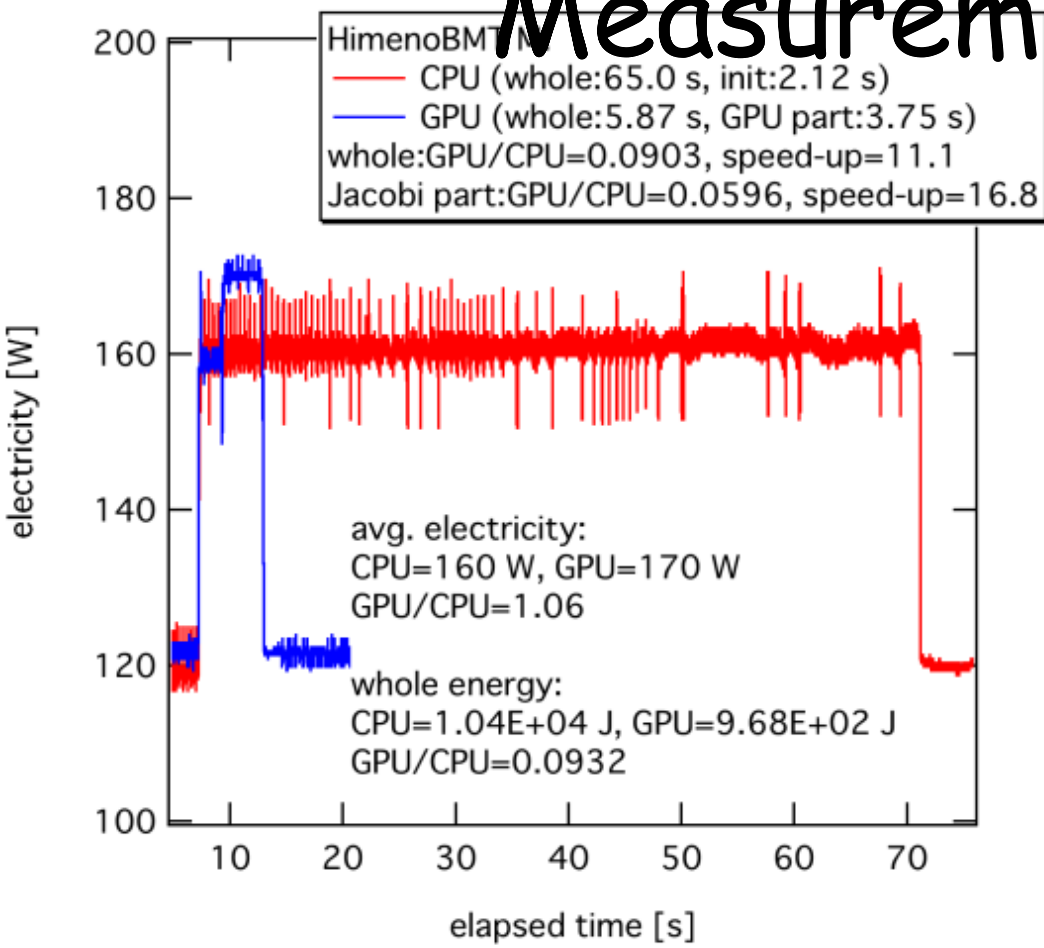
Power Efficiency in 3-D FFD

GPU	Computation	Idle	Power	GFLOPS	GFLOPS/W
RIVA128		126 W	140 W	10.3	0.074
8800 GT	On GPU	180 W	215 W	62.2	
8800 GTS	On GPU	196 W	238 W	67.2	
8800 GTX	On GPU	224 W	290 W	84.4	

CUDA GPUs have four times higher power efficiency than CPU in high-performance FFT.

RIVA128 is an old, low-power GPU,
to measure pure power consumption
of host system (CPU, chipset, memory).
The interface is legacy PCI.

Himeno Size M Power Measurement



Jacobi

- **＊CPU**
  - **Av. Power 160W**
  - **Exec. Time：62.88s**
- **＊GPU**
  - **Av. Power：170W**
  - **Exec. Time：3.75s**

GPU/CPU

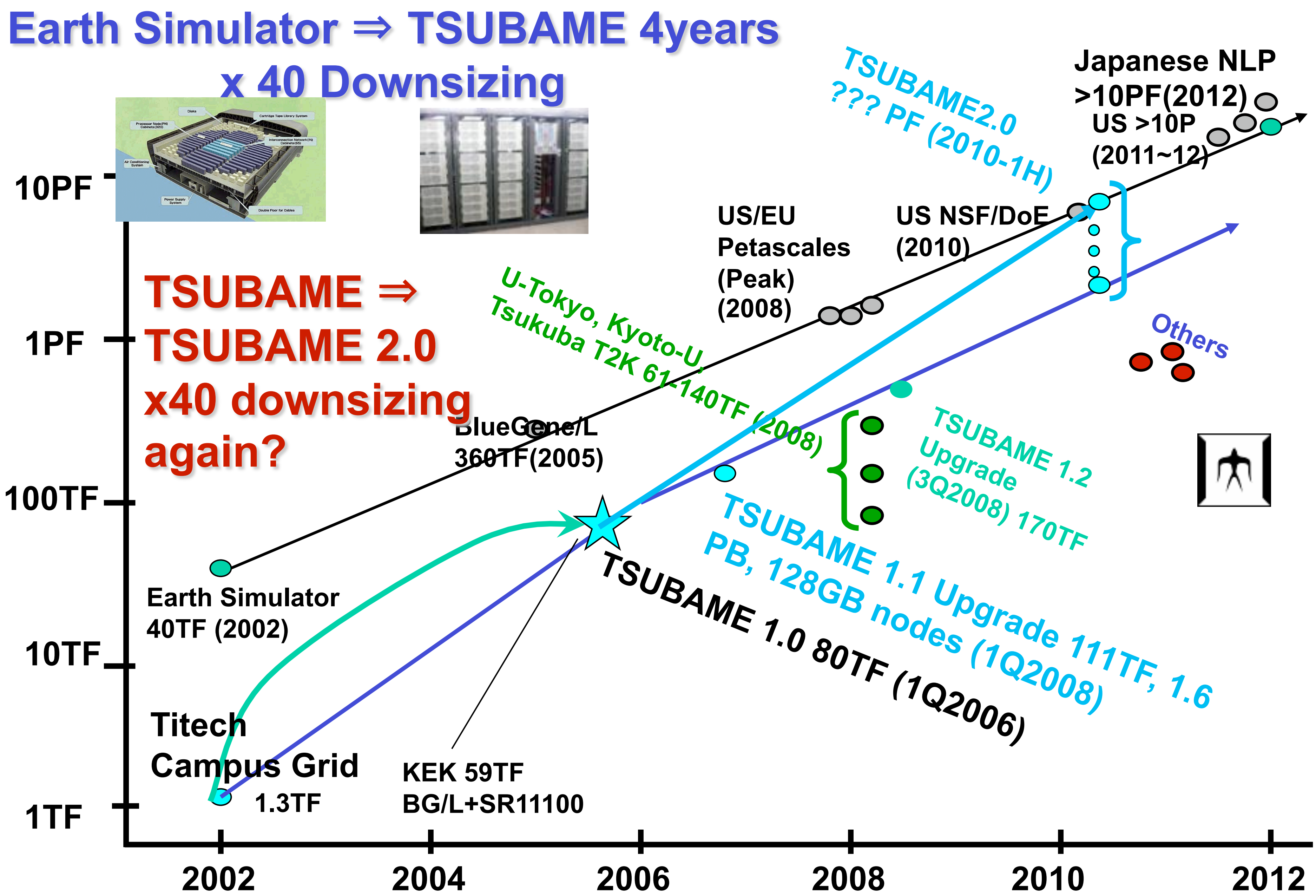
- **Power：106%**
- **Energy:1/15.8 ＝6.33%**

For extreme memory intensive CFD app GPU uses only 6% of CPU energy

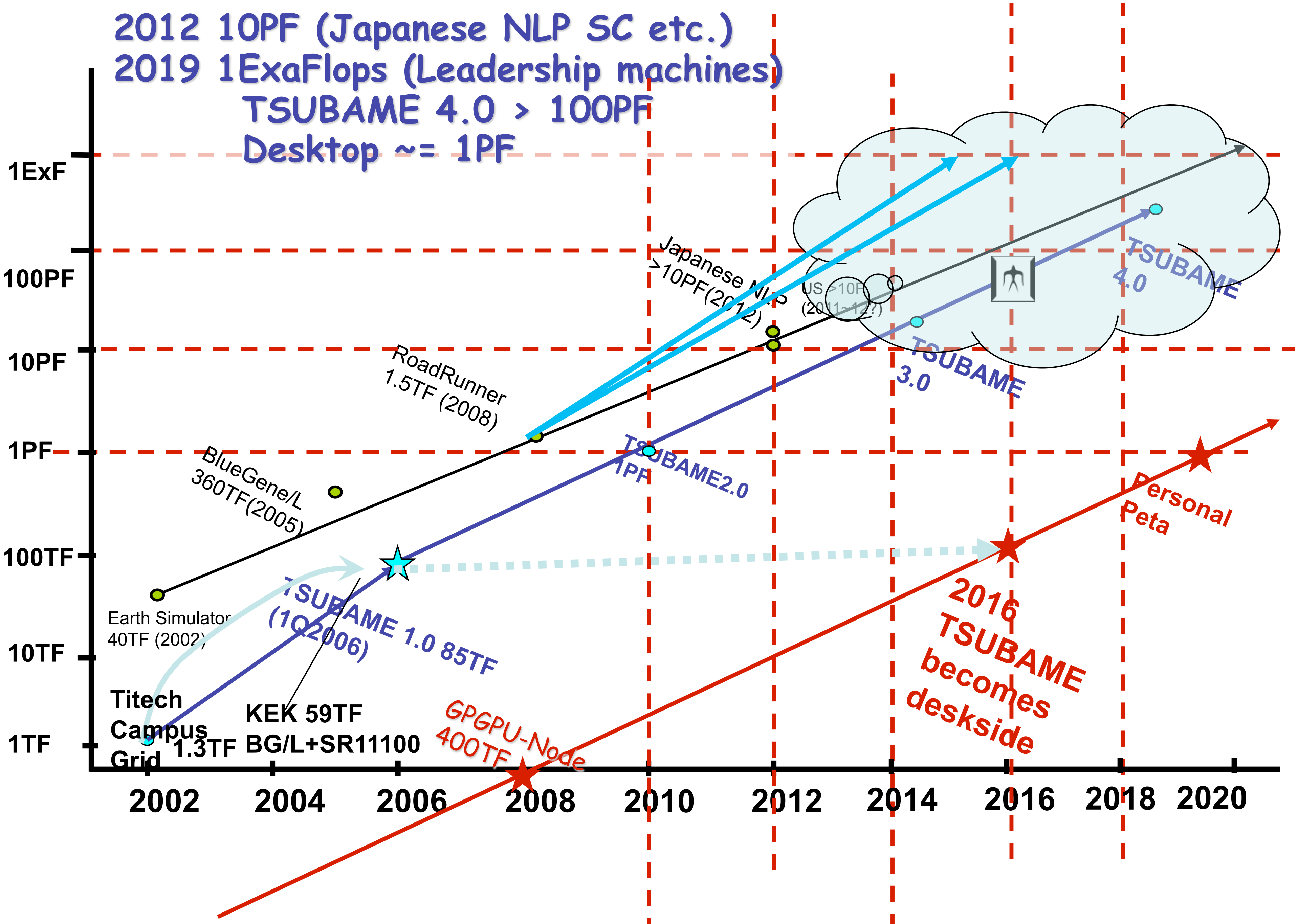
- 1. We can build an exascale system---with all its problems
- 2. Capacity apps disguised as capability will result in significant loss of efficiency
- 3. The entire machine can be more or less used for large jobs but will have room left over for capability
- 4. One would need ecosystem and growth model to improve app to be more capability oriented as problem scales
- 5. Next generation Cloud and SC centers will converge, with low cost HPC networking and commodity acceleration

"It is the Will"

Towards TSUBAME 2.0

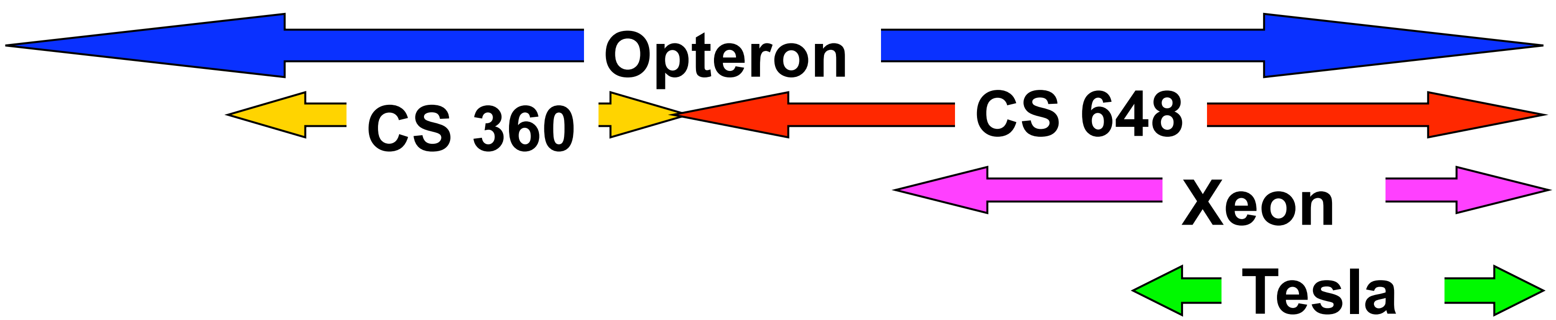


Road to Exascale



TSUBAME in Top500 Ranking

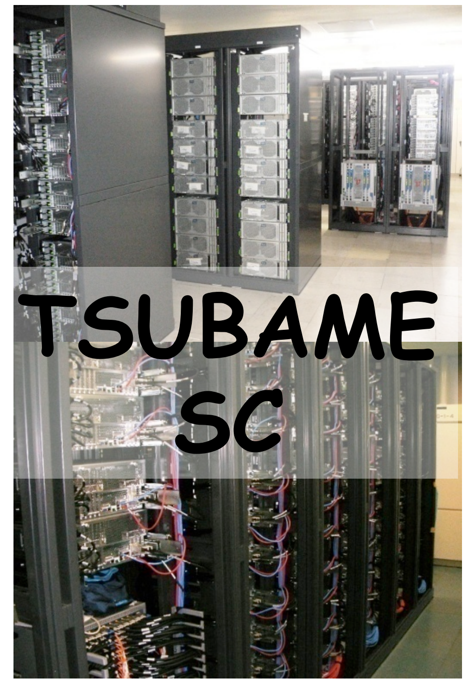
	Jun 06	Nov 06	Jun 07	Nov 07	Jun 08	Nov 08	Jun09
Rmax (Tflops)	38.18	47.38	48.88 [HPDC 2008]	56.43	67.70	77.48	???
Rank	7	9	14	16	24	29	???



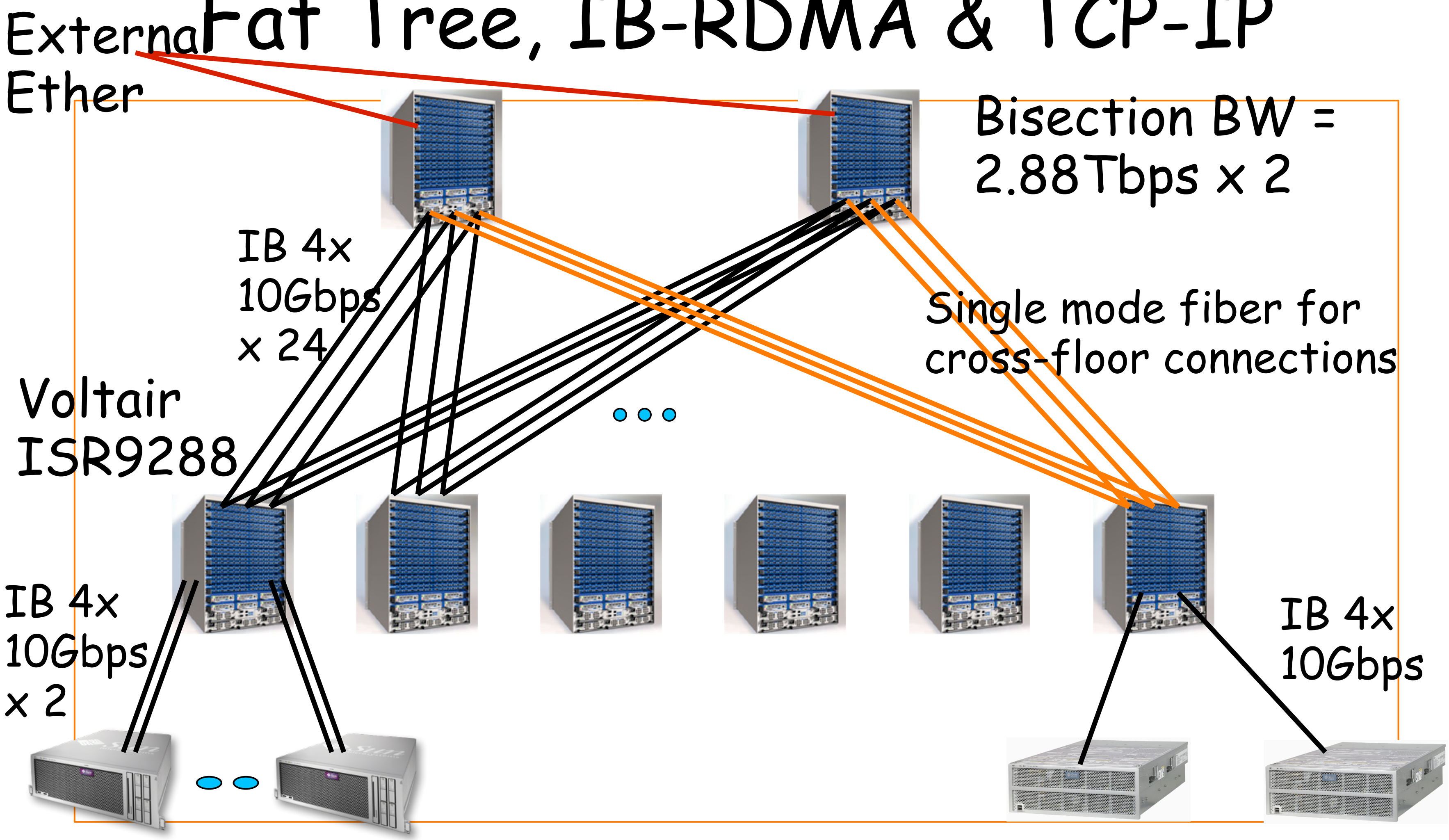
- Continuous improvement for 6 times
- The 2nd fastest heterogeneous supercomputer in the world (No.1 is RoadRunner)

IDC Servers and Cluster SC--- the differences (or are there?)

- The Same---the nodes
  - Processors (x86)
  - Memory (DDR DRAM)
  - I/O (PCI-e)
  - OS (Linux/Windows), MW
- Differences
  - Network (IB vs GbE) (< 10% of machine cost vs. 10-25%)
  - Parallel Storage
  - Power Density
  - Parallel SW Stack: (MPI, OpenMP, BQ, ...)
  - Operations as a SC
- Accelerators?



TSUBAME Network: ~1400 port Fat Tree, IB-RDMA & TCP-IP



X4600 x 120nodes (240 ports) per switch
=> 600 + 55 nodes, 1310 ports, 13.5Tbps

X4500 x 60nodes (60 ports)
=>60ports 600Gbps

Incompressible CFD Application

Incompressible Navior-Stokes Equation

$$\nabla \cdot \boldsymbol{u} = 0$$

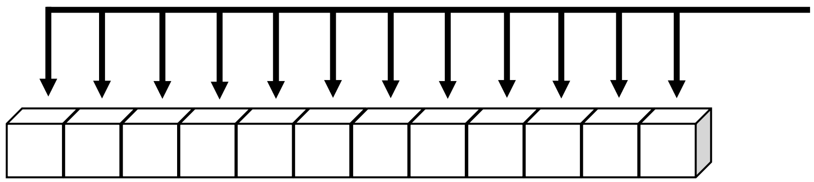
$$\frac{\partial \boldsymbol{u}}{\partial t} + \boldsymbol{u} \cdot \nabla \boldsymbol{u} = -\frac{1}{\rho} \nabla p + \nu \Delta \boldsymbol{u}$$

Poisson equation

$$\Delta p^{n+1} = \frac{\nabla \cdot \boldsymbol{u}^{n+1}}{\Delta t}$$

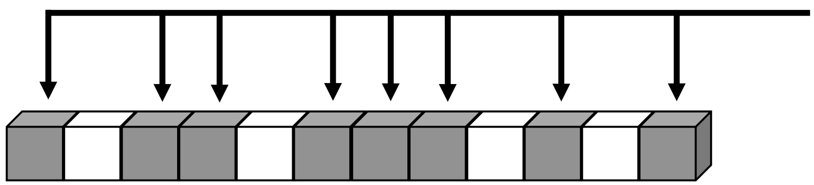
- ■ **Advection Term：** **High-accurate FDM** (Suitable for GPU)
- ■ **Diffusion Term：** **2nd order Center FDM** (easy)
- ■ **Velocity Divergence：** **Staggered FDM** (easy)
- ■ **Poisson equation：** **Red & Black MG** (hard)
- ■ **Pressure Gradient：** **Staggered FDM** (easy)

Types of Memory Access



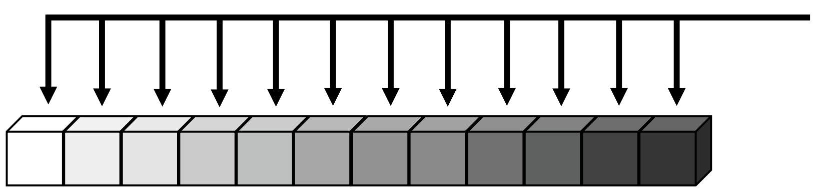
Continuous Access
FDM (Finite Difference)

good



Random (Indirect) Access
FEM (Finite Element)
A[i] = A[IP[i]] + A[IP[i-1]];

So-So



Data Dependency

A[i] = A[i-1] + A[i-2]*C;

Not Good

Poisson Equation solved by

MG(Multi Grid), Red & Black method

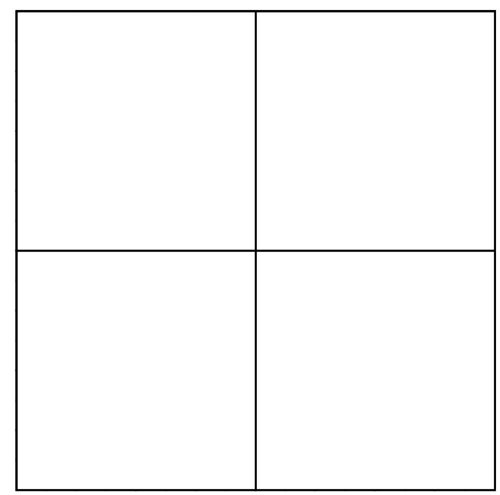
■ **Algorithm Acceleration**

Point Jacobi —×4 ～ 5→ **SOR** —×100→ **MG-SOR**

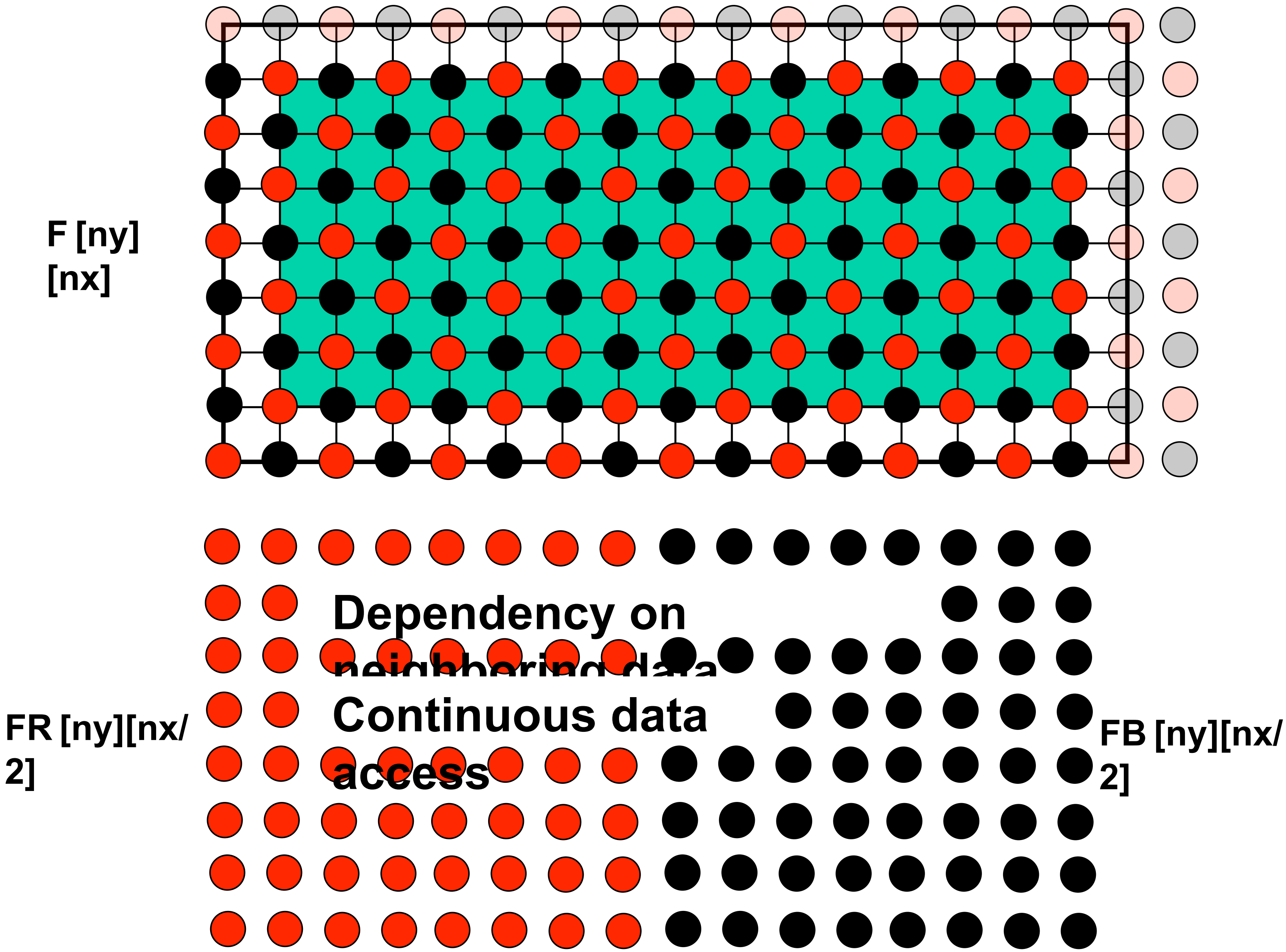
■ **Hardware Acceleration :**

GPU (CUDA) **×50?**

$$\frac{f_{i+1,j} - 2f_{i,j} + f_{i-1,j}}{\Delta x^2} + \frac{f_{i,j+1} - 2f_{i,j} + f_{i,j-1}}{\Delta y^2} = \rho_{i,j}$$



Red & Black method



Two-Stream Instability in Plasma Physics

Vlasov-Poisson Equation:

$$\frac{\partial f}{\partial t} + v\frac{\partial f}{\partial x} - \frac{eE}{m_e}\frac{\partial f}{\partial v} = 0 \qquad \frac{\partial^2 \phi}{\partial x^2} = \frac{e(n_e - n_i)}{\varepsilon_0}$$

$$\left(E = -\frac{\partial \phi}{\partial x}, \quad n_e = \int f\, dv \right)$$

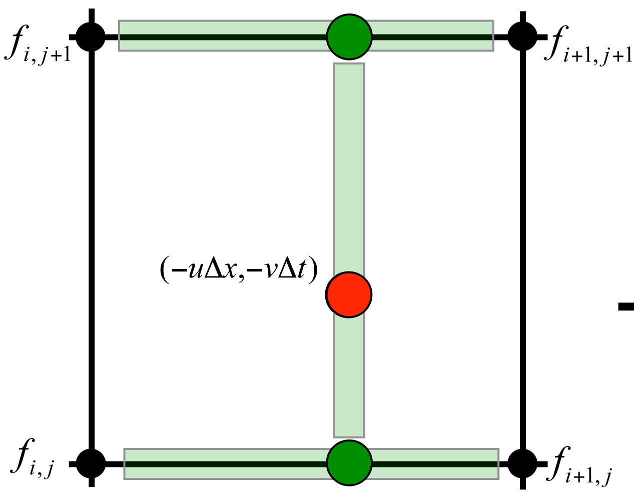
f **: electron distribution function**

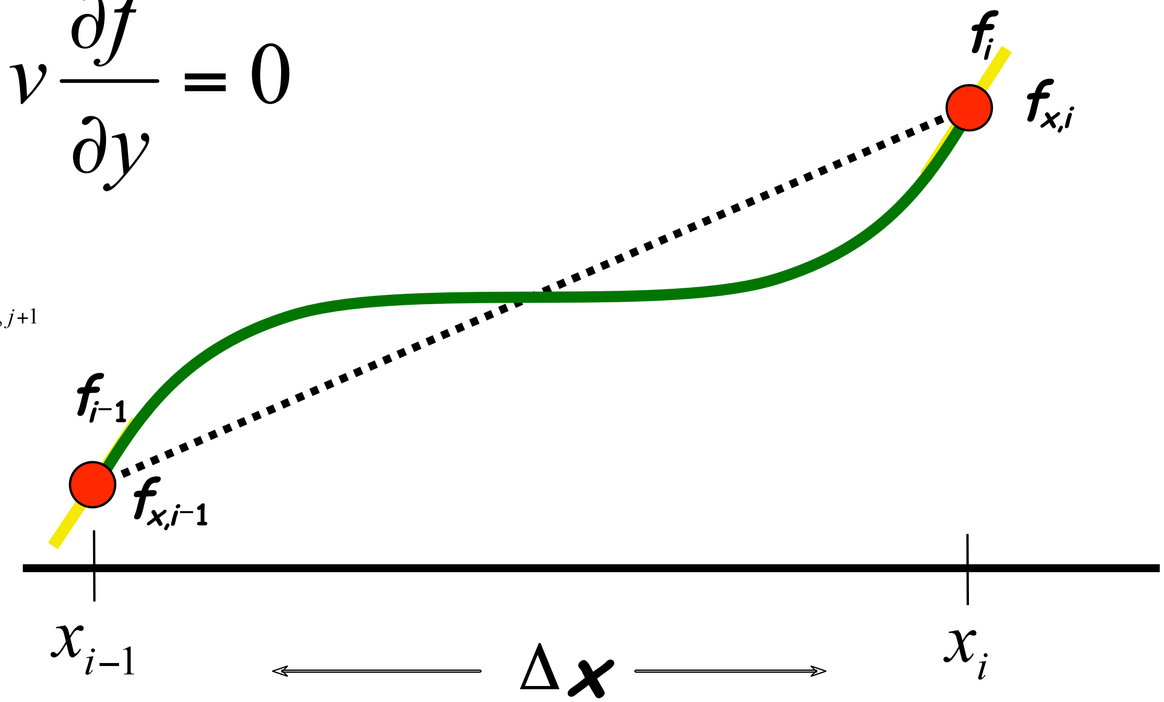
n **: electron number density**

CIP Method for 2-dimensional Advection

Equation

$$\frac{\partial f}{\partial t} + u\frac{\partial f}{\partial x} + v\frac{\partial f}{\partial y} = 0$$

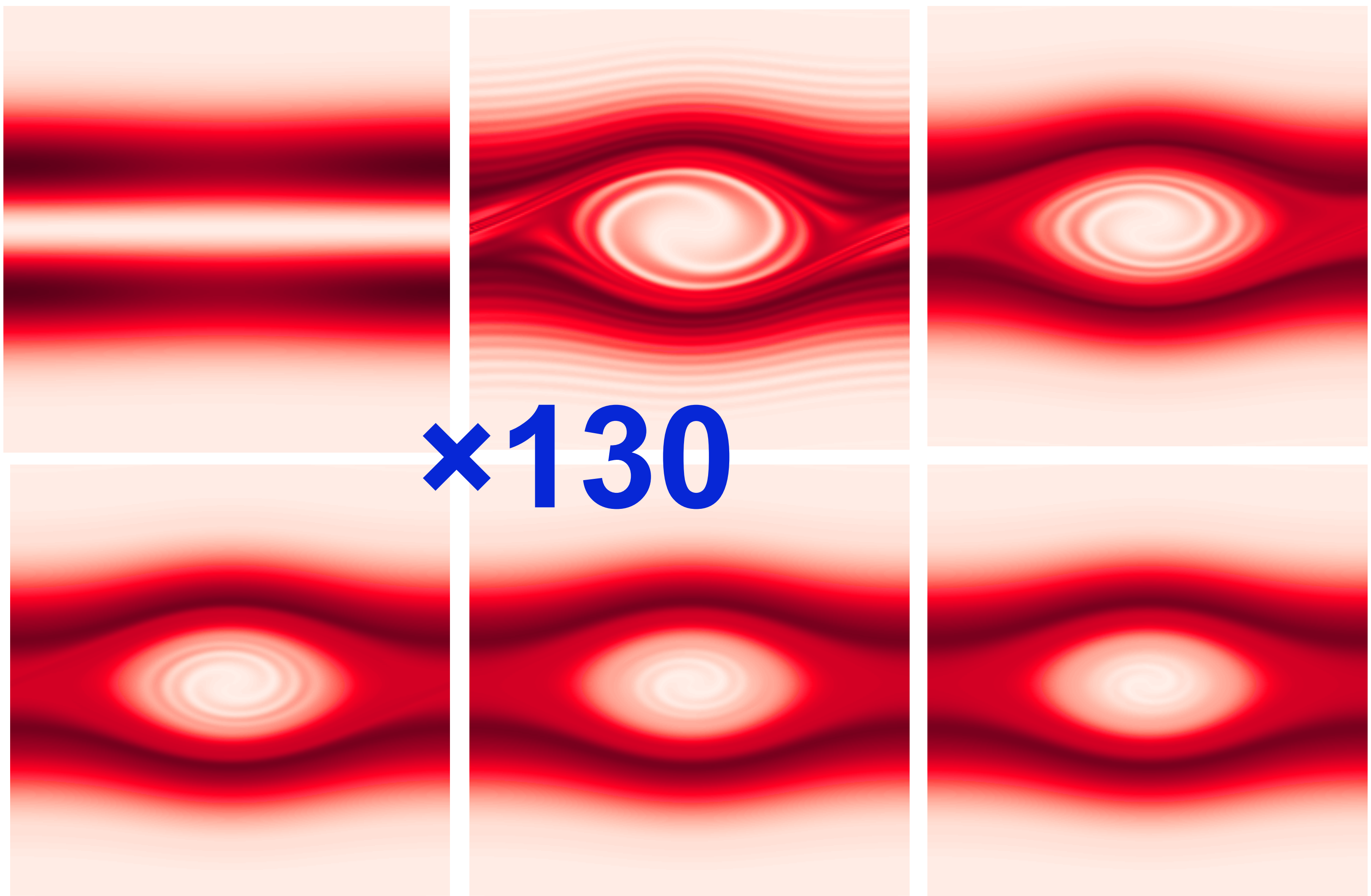




$$f_i^{n+1} = F_{CIP}(-u\Delta x) = a\xi^3 + b\xi^2 + f_{x,i}\xi + f_i$$

$$a = \frac{1}{\Delta x^2}\left(f_{x,i} + f_{x,i-1}\right) - \frac{2}{\Delta x}\left(f_i - f_{i-1}\right), \quad b = \frac{1}{\Delta x}\left(2f_{x,i} + f_{x,i-1}\right) - \frac{3}{\Delta x^2}\left(f_i - f_{i-1}\right)$$

120 GFLOPS using 8800GTS

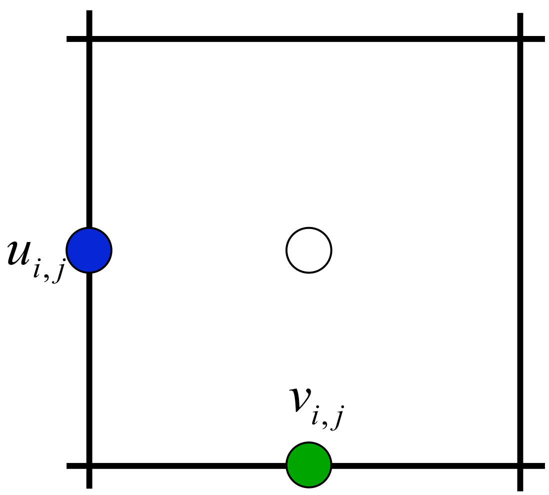


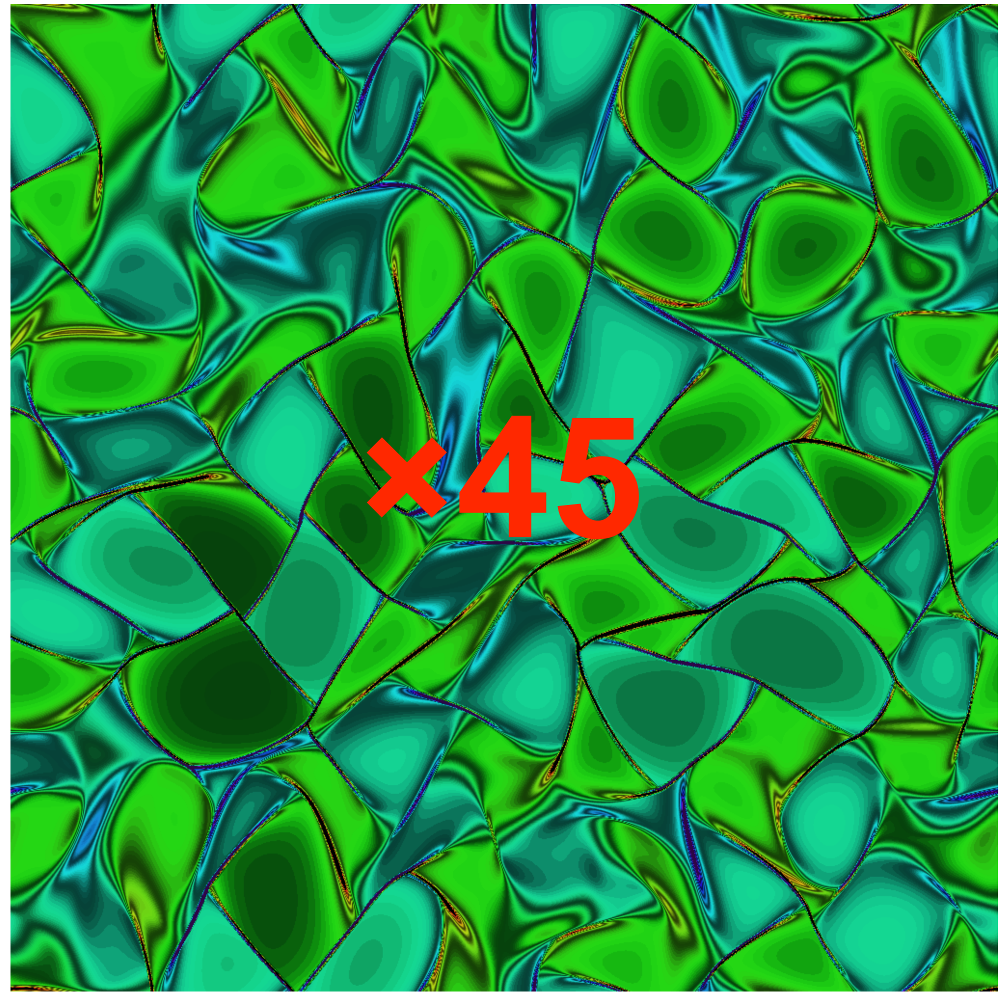
Two-dimensional Burgers Equation

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = \kappa\left(\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right)$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = \kappa\left(\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right)$$

GeForce 8800 GTS
40 GFLOPS





1024×1024

velocity u at the *v-point* $\quad u_s = \dfrac{u_{i,j} + u_{i+1,j} + u_{i,j-1} + u_{i+1,j-1}}{4}$

77

Homogeneous Isotopic Turbulence

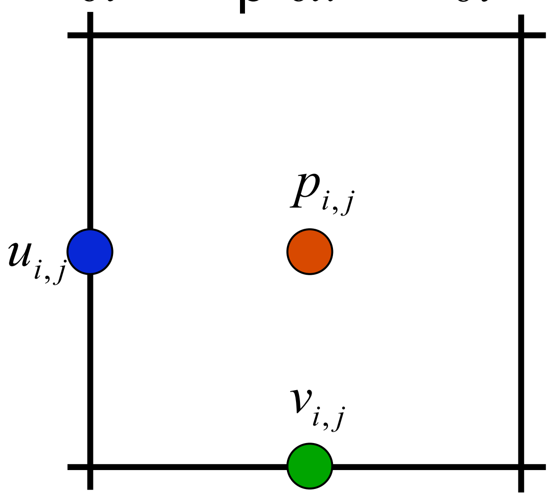
Burgers equation

Poisson equation

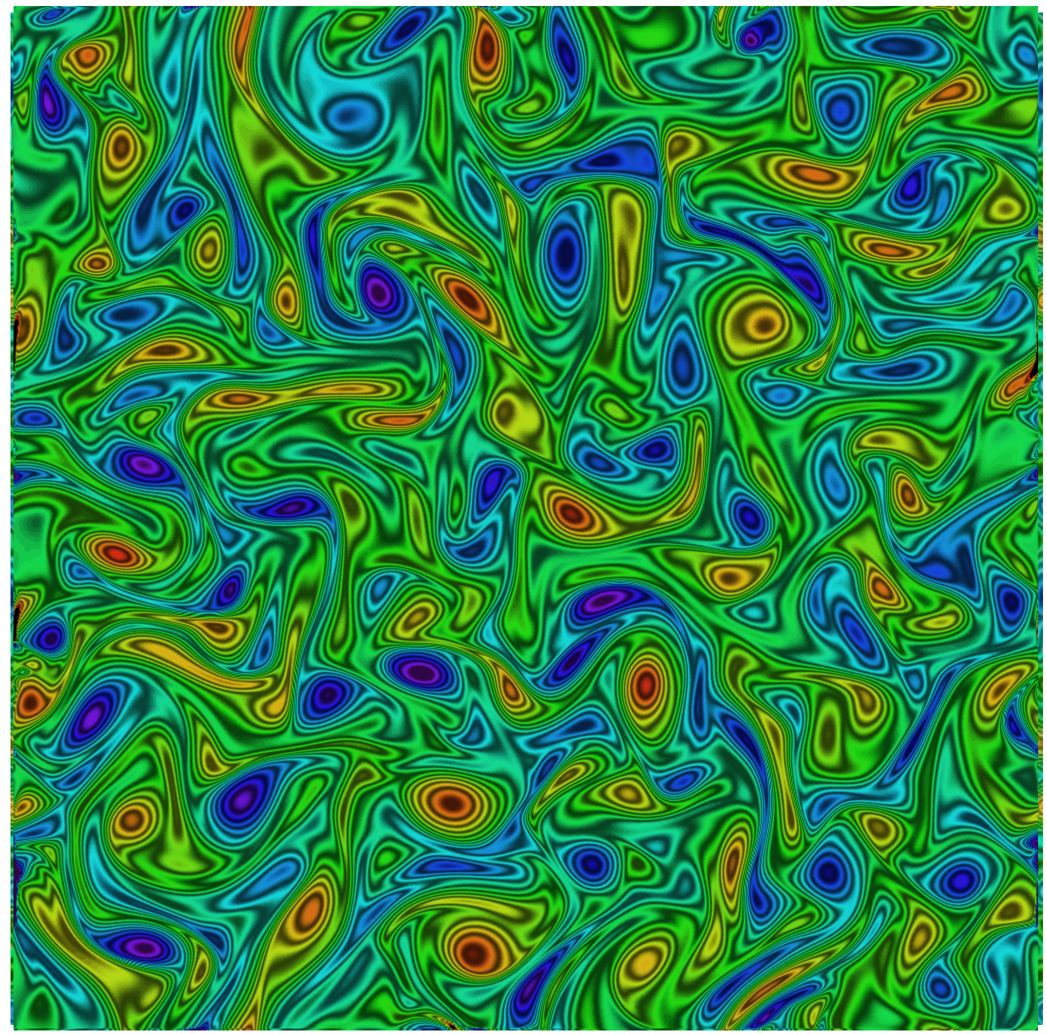
$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \frac{1}{\Delta t}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right)$$

Correction

$$\frac{\partial u}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial x} \qquad \frac{\partial v}{\partial t} = -\frac{1}{\rho}\frac{\partial p}{\partial y}$$

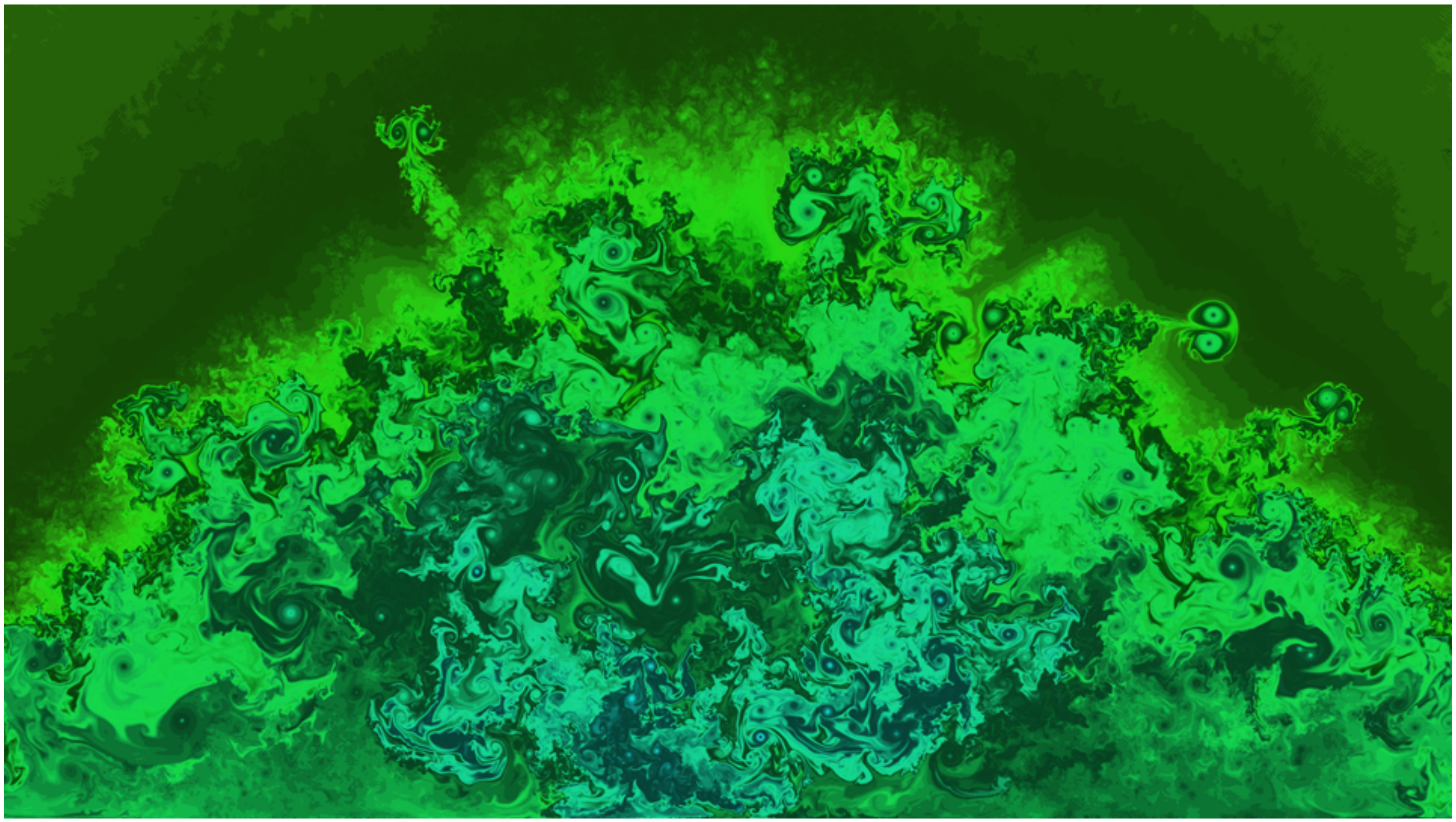


1024×1024



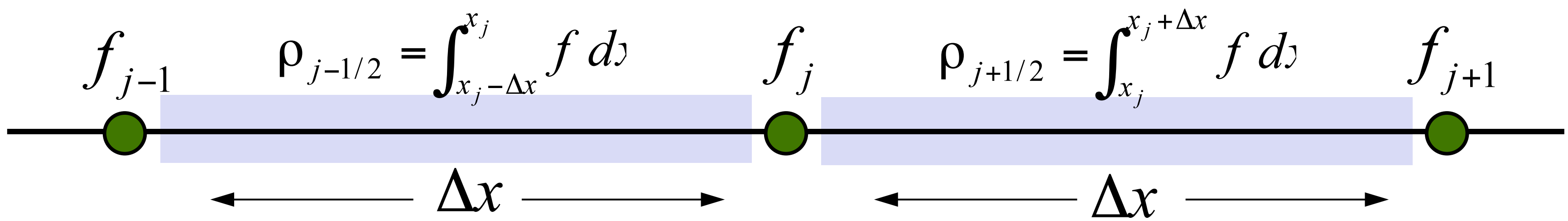
Compressible CFD Application

High-accurate numerical Scheme becomes very important.



Numerical Scheme IDO-CF

Y. Imai, T. Aoki and K. Takizawa, J. Comp. Phys., Vol. 227, Issue 4, 2263-2285 (2008)



$$F(x) = ax^4 + bx^3 + cx^2 + dx + f_j$$

Four matching conditions :

$$\int_{x_j-\Delta x}^{x_j} F(x)dx = \rho_{j-1/2} \quad F(-\Delta x) = f_{j-1} \quad \int_{x_j}^{x_j+\Delta x} F(x)dx = \rho_{j+1/2} \quad F(\Delta x) = f_{j+1} \ ,$$

Unknown coefficients :

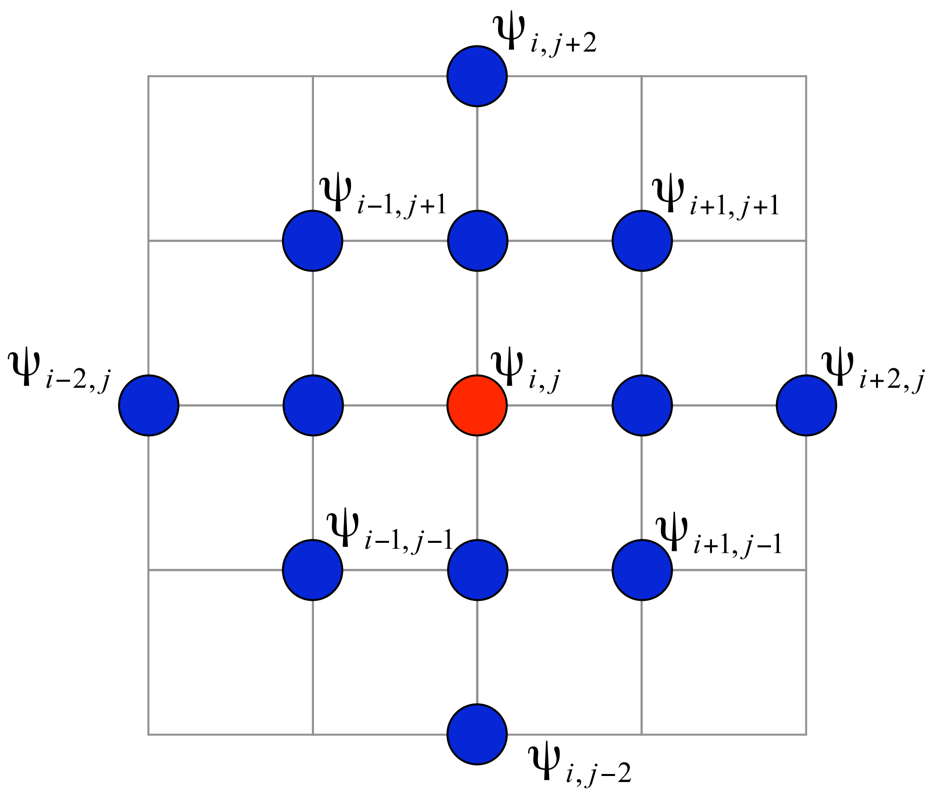
$$c = \frac{5}{4}\frac{3\rho_{j+1/2} + 3\rho_{j-1/2} - 6f_j\Delta x}{\Delta x^3} - \frac{3}{4}\frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta x^2} \qquad d = 2\frac{\rho_{j+1/2} - \rho_{j-1/2}}{\Delta x^2} - \frac{f_{j+1} - f_{j-1}}{2\Delta x}$$

$$\frac{\partial}{\partial x}F(0) = 2\frac{\rho_{j+1/2} - \rho_{j-1/2}}{\Delta x^2} - \frac{f_{j+1} - f_{j-1}}{2\Delta x}$$

$$\frac{\partial^2}{\partial x^2}F(0) = \frac{5}{2}\left(\frac{3\rho_{j+1/2} + 3\rho_{j-1/2} - 6f_j\Delta x}{\Delta x^3}\right) - \frac{3}{2}\left(\frac{f_{j+1} - 2f_j + f_{j-1}}{\Delta x^2}\right)$$

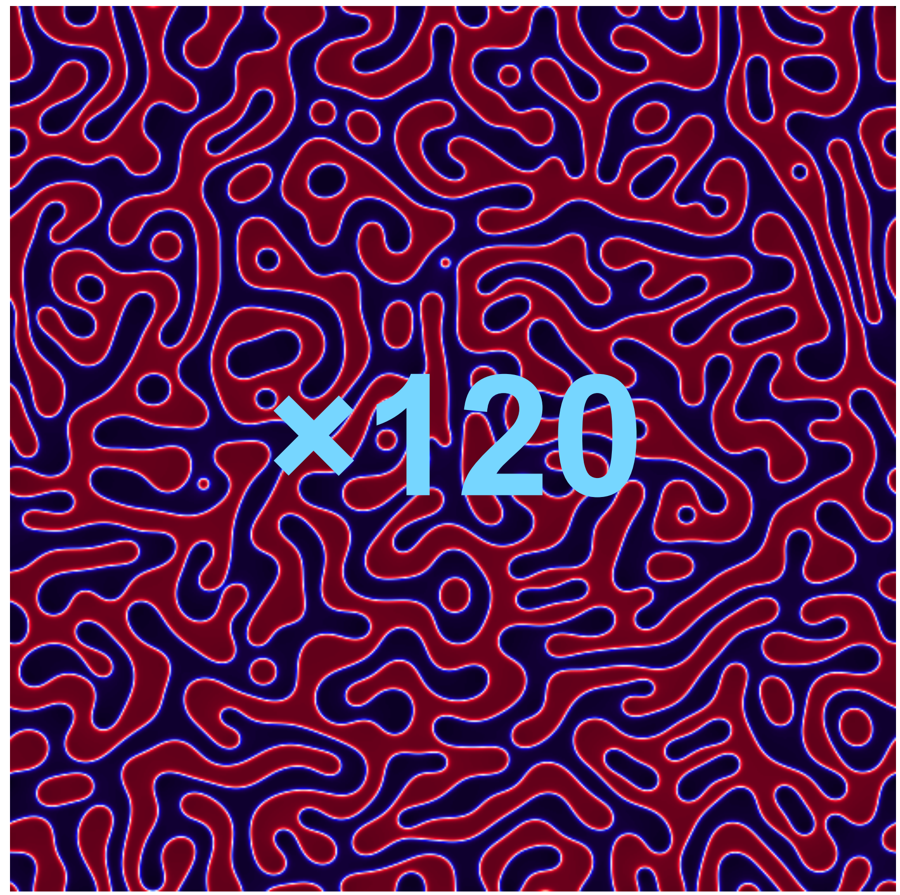
2-D Computation of Phase Separation

Mixture of Oil and Water:



114 GFLOPS using GTX280

512 x 512



Thou Shalt Specialize or Not

- HPC Architectures RIP…
  - Custom CPUs: Too many to be told…
  - Accelerators: Vector options on CM-5, Meiko-CS, Alliant FX, Grape… (RIP)
  - Very small ecosystem---no scale of economy
  - Arcane programming environment
  - Quick catchup by the "killer micros"
  - Not many code ported for fear of deprecation
- Specialized HPC Architectures Liveth(!)
  - NEC SX, Fujitsu PrimeXXX, SciCortex
  - Are they like birds evolved from Dinosaurs?

FUJITSU THE POSSIBILITIES ARE INFINITE

"Why HPC Architecture Must be Custom Built in the Exascal Era"

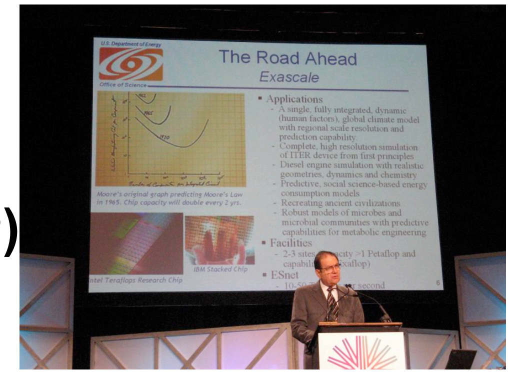
2007-11-28

Slides Coutesy of Hisa Ando
(Former) Senieor Architect
Fujitu Ltd.

(Abridged and Translated by Satoshi Matsuoka)

Exaflop HPC Energy Consumption

- In SC07, Ray Orbach "Exascale by 2016"
- Energy Consumption at 90nm
  - FPU: ～500pJ/DPFOP
    - (GRAPE-DR: 65W/256GFlops=250pJ/DP FOP)
  - General Purpose CPU：20nJ/Cycle
    - 4FOP/Cycle => x10 power over FPU



- 1 Exa Flops power requirement
  - Circa 2006-7: 90nm technology: 500pJ x 10^{18} = 500×10^{6} W
  - Circa 2016- (conservatively) suppose 22nm technology
    - Gate capacitance 22/90 = x 0.24, Vcc 0.8V/1V = x 0.8
    - Power $\propto CV^2 = 0.24 \times 0.8^2$ = x 0.15
    - 1 ExaFlop FPU array：0.15 x 500pJ x 10^{18} = 75MW(!)

FUJITSU
THE POSSIBILITIES ARE INFINITE

Why Special Architecture for Exascale?

- **Total System Power ~= 1.5GW～2GW (!?)**
  - Extrapolate to gen. purpose CPU: 75MWx10 = 750MW
  - Memory, power delivery loss, cooling, I/O and storage… incur additional x2~x3 overhead
  - > $100 million in Utility Bill(!)
- **Save Power, save power, and save power:**
  - Objective: 1/30 power reduction
    - Energy reduction of FPUs---low power design
    - SIMD-parallel control of massive FMA FPUs + Powerful scalar processor---beat Amdahl's law

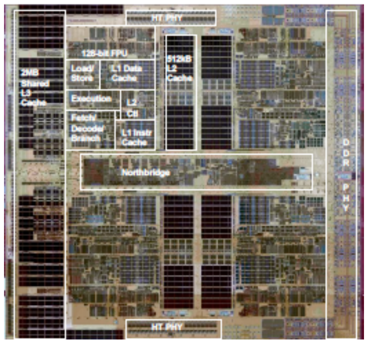
Tokyo Electric Co. Sodegaura PP 3.6GW

- **Claim (by Ando) such a processor cannot be general-purpose（= for Commercial Apps）**
- **I.e., Exascale machine must be (made of) special-purpose HPC architecture**

FUJITSU

THE POSSIBILITIES ARE INFINITE

Special Purpose Processor for Exascale circa 2016



Server Processor

65nm technology
AMD 4 Core Opteron
Chip 283mm^2
Core 26mm^2

22nm technology
32 Cores
Chip 283mm^2
Core 3.5mm^2

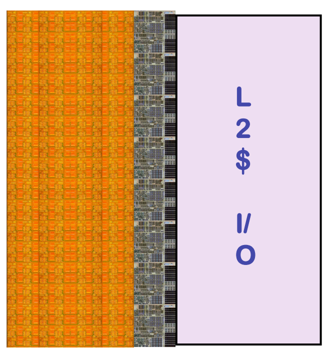
128Fopx5GHz
= 640GFlops

Grape-DR
90nm technology
512FM+FA

22nm technology
25(FM＋FA)s/mm^2

Suppose：
16FMA/mm^2

Custom HPC processor



8 Core (28mm^2)
＋2048FMA (128mm^2)
＋16MB L2$/LM (35mm^2)
＋I/O（60mm^2)

4096Fopx5GHz
= 20TFlops

22nm technology
Chip ~250mm^2

- Equivalent to CACM
- "Acceleration Again" Key to Supercomputing

5 Articles, 4 from Tokyo Tech on GPUs

Technology and Architectures for Future Large-Scale Computing Systems

Rick Stevens

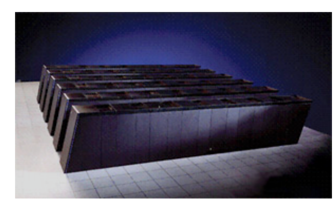
Argonne National Laboratory

The University of Chicago

U.S. Department of Energy
Office of Science

ASCR High Performance and Leadership Computing Facilities

- NERSC
  - 104 teraflop Cray XT4 with approximately 9,600 dual core processors; **will upgrade to approximately 360 teraflops with quad core in Summer, 2008**
  - 6.7 teraflop IBM Power 5 (Bassi) with 888 processors, 3.5 terabytes aggregate memory
  - *3.1 teraflop LinuxNetworx Opteron cluster (Jacquard) with 712 processors, 2.1 terabytes aggregate memory*
- LCF at Oak Ridge
  - 263 teraflop Cray XT4 (Jaguar) with 7,832 quad core 2.1 GHz AMD Opteron processor nodes, 46 terabytes aggregate memory
  - *18.5 teraflop Cray X1E (Phoenix) with 1,024 multi-streaming vector processors*
  - **Delivery of 1 Petaflop Cray Baker in 2008**
- Argonne LCF
  - *5.7 teraflop IBM Blue Gene/L (BGL) with 2,048 PPC processors*
  - 100 teraflop IBM Blue Gene/P began operations April 1, 2008
  - **446 teraflop IBM Blue Gene/P upgrade accepted in March, 2008.**



IBM Blue Gene/P – 556 TFs @ Argonne

160K cores, 80 TB RAM, 10 PB disk

Cray XT5 at ORNL > 1 Pflop/s in November 2008

Jaguar	Total	XT5	XT4
Peak Performance	1,645	1,382	263
AMD Opteron Cores	**181,504**	**150,176**	**31,328**
System Memory (TB)	**362**	**300**	**62**
Disk Bandwidth (GB/s)	**284**	**240**	**44**
Disk Space (TB)	**10,750**	**10,000**	**750**
Interconnect Bandwidth (TB/s)	**532**	**374**	**157**

The systems will be combined after acceptance of the new XT5 upgrade. Each system will be linked to the file system through 4x-DDR Infiniband

Traditional Sources of Performance Improvement are Flat-Lining (2004)

- New Constraints
  - 15 years of *exponential* clock rate growth has ended
- Moore's Law reinterpreted:
  - How do we use all of those transistors to keep performance increasing at historical rates?
  - Industry Response: #cores per chip doubles every 18 months *instead* of clock frequency!

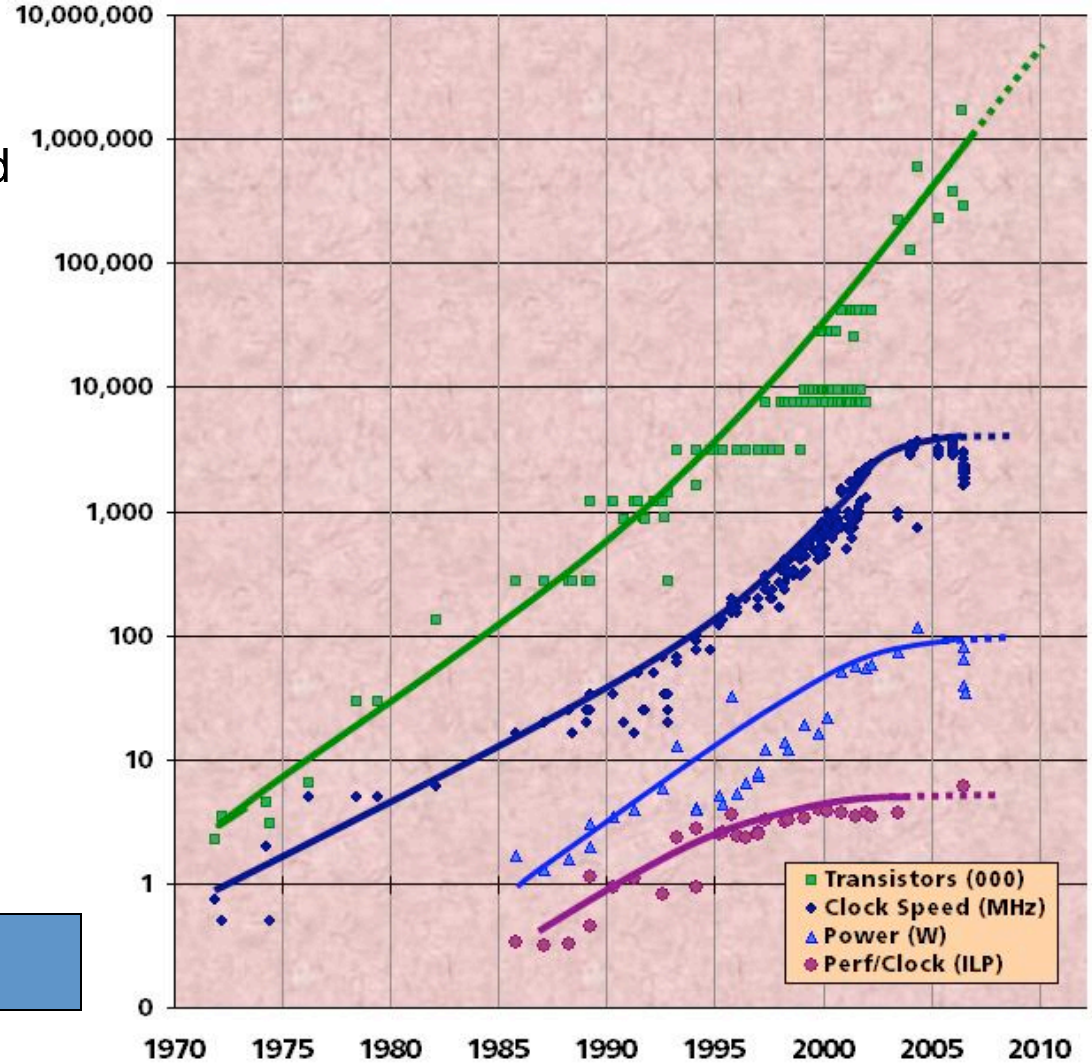
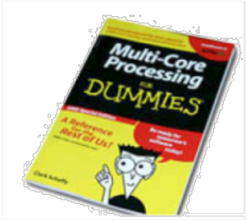
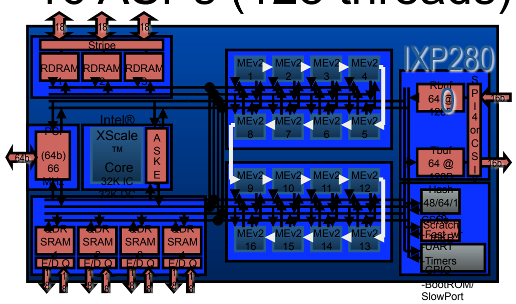


Figure courtesy of Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

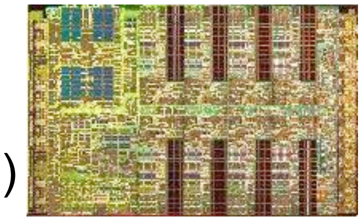


Multicore comes in a wide variety

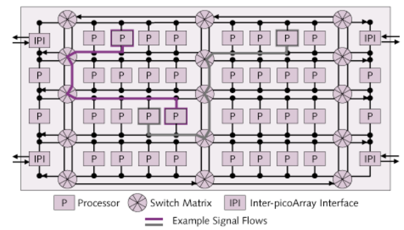
– Multiple parallel general-purpose processors (GPPs)
– Multiple application-specific processors (ASPs)



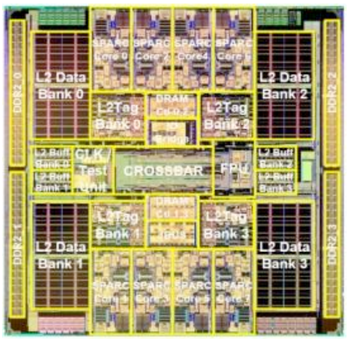
Intel Network Processor
1 GPP Core
16 ASPs (128 threads)



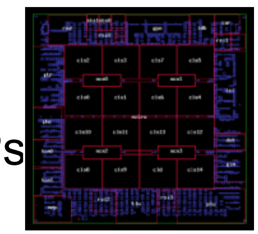
IBM Cell
1 GPP (2 threads)
8 ASPs



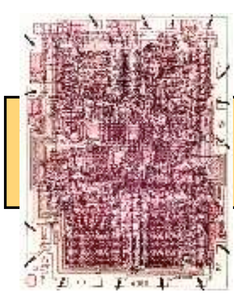
Picochip DSP
1 GPP core
248 ASPs



Sun Niagara
8 GPP cores (32 threads)



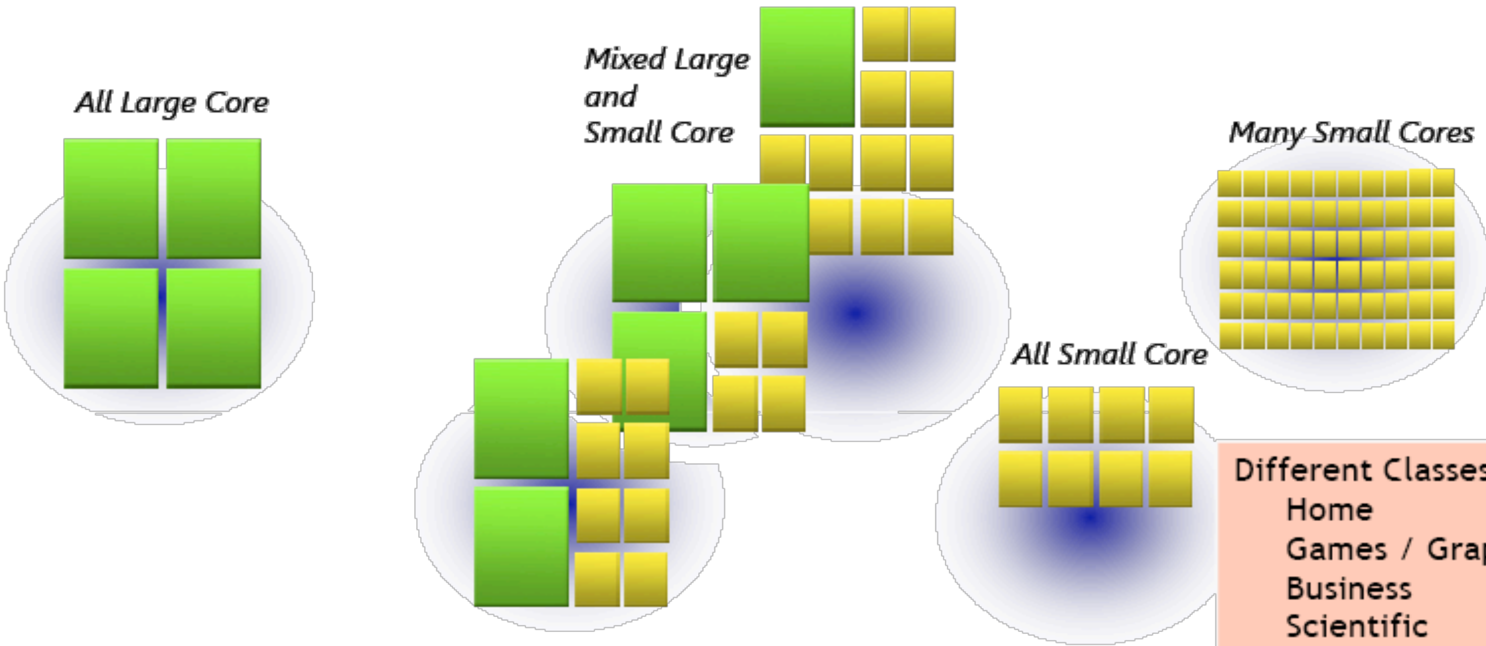
Cisco CRS-1
188 Tensilica GPPs



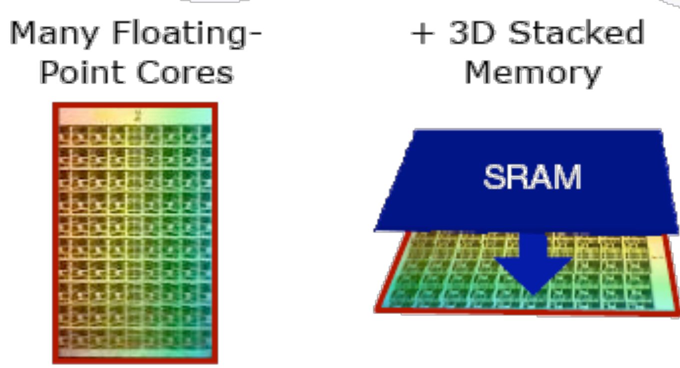
Intel 4004 (1971):
4-bit processor,
2312 transistors,
~100 KIPS,
10 micron PMOS,
11 mm² chip

“The Processor is the new Transistor” [Rowen]

What’s Next?



Different Classes of Chips
- Home
- Games / Graphics
- Business
- Scientific



The question is not whether this will happen but whether we are ready

Source: Jack Dongarra, ISC 2008

Outline of the Situation

- Million core systems and beyond are on the horizon
- Today labs and universities have general purpose systems with 10k-200K cores (BGL@ LLNL 200K, BGP@Argonne 160K, XT5@ORNL 150K cores)
- By 2012 there will be more systems deployed in the 200K-1M core range
- By 2020 there will be systems with perhaps 100M cores

- Personal systems with > 1000 cores within 5 years (I have over > 150 64bit cores in my office now) plus 240 GPU cores
- Personal systems with requirement for 1M threads is not too far fetched (GPUs for example)

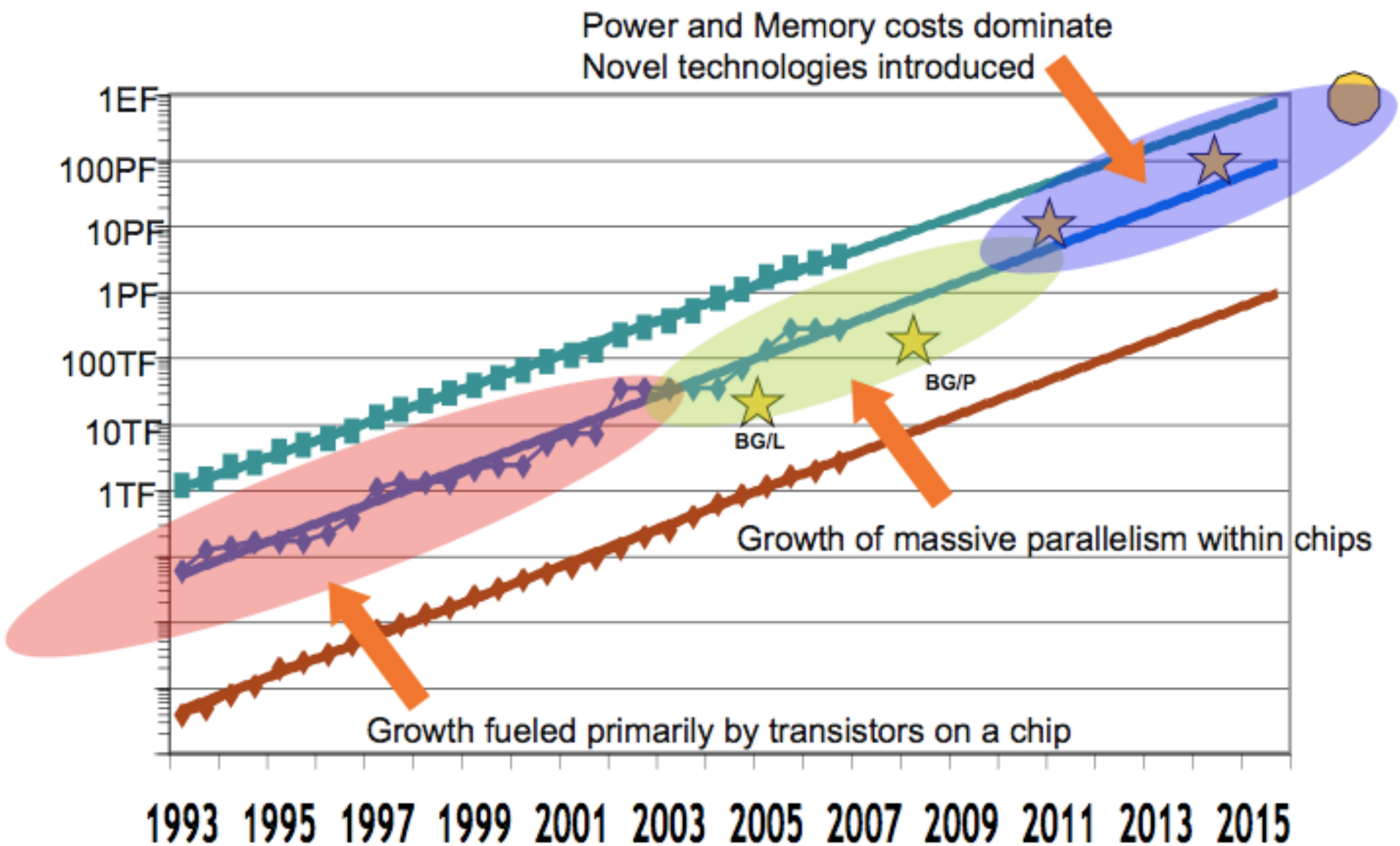
E3 Advanced Architectures - Findings

- Exascale systems are likely feasible by 2017±2
- 10-100 Million processing elements (mini-cores) with chips as dense as 1,000 cores per socket, clock rates will grow slowly
- 3D chip packaging likely
- Large-scale optics based interconnects
- 10-100 PB of aggregate memory
- > 10,000's of I/O channels to 10-100 Exabytes of secondary storage, disk bandwidth to storage ratios not optimal for HPC use
- Hardware and software based fault management
- Simulation and multiple point designs will be required to advance our understanding of the design space
- Achievable performance per watt will likely be the primary metric of progress

Top Technical Challenges

- Power Consumption
  - Proc/mem, I/O, optical, memory, delivery
- Chip-to-Chip Interface Scaling
  - pin/wire count $\Rightarrow$ 3D packaging
- Package-to-Package Interfaces (optics?)
  - Signaling rate, density, cost
- Fault Tolerance
  - FIT rates and Fault Management
  - Reliability of irregular logic, design practice
- Cost Pressure in Optics and Memory
  - CPUs will be smaller fraction of cost

Looking out to Exascale
Concurrency will be Doubling every 18 months



Systems Scaling Projections

Begin Full System Delivery (Yr)	2004	2007	2012	2015	2019
Design Parameters	**BG/L**	**BG/P**	**25PF**	**300PF**	**1200PF**
Cores / Node	2	4	8-24	32-64-128	96-128-500
Clock Speed (GHz)	0.7	0.85	1.6-4.1	2.3-4.8	2.8-6.0
Flops / Clock / Core	4	4	8-32	8-32	16-64
Nodes / Rack	1024	1024	100-1024	256-1024	256-1024
Racks / Full System Config	64	72	128-350	128-400	256-400
MB RAM/core	256	512	1024-4096	1024-4096	1024-4096
Total Power	2.5MW	4.8MW	8MW-20MW	20MW-50MW	30MW-80MW
Flops / Node (GF)	5.6	14	128-640	640-2000	2000-6000
Flops / Rack (TF)	5.7	14	200-400	400-1200	1600-4800
LB Concurrency	5.E+05	1.E+06	1M-2M	10M-100M	400M-1000M
Full System					
Total Cores (Millions)	0.13	0.3	.3M-1.2M	1M-10M	4M-30M
Total RAM (TB)	33.6	151	2,000-4,400	3,000-10,000	5,000-25,000
Total Racks	64	72	128-350	128-400	256-400
Peak Flops System (PF)	**0.37**	**1**	**25**	**300**	**1200**

ITRS Roadmap for Logic Devices

	Units	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Feature Size	**nm**	**90**	**78**	**68**	**59**	**52**	**45**	**40**	**36**	**32**	**28**	**25**	**22**	**20**	**18**	**16**	**14**
Logic Area	relative	1.00	0.80	0.63	0.51	0.39	0.32	0.25	0.20	0.16	0.12	0.10	0.08	0.06	0.05	0.04	0.03
SRAM Area	relative	1.00	0.78	0.61	0.48	0.38	0.29	0.23	0.18	0.14	0.11	0.09	0.07	0.06	0.04	0.03	0.03
50/50 Area	relative	1.00	0.79	0.62	0.49	0.38	0.30	0.24	0.19	0.15	0.12	0.09	0.07	0.06	0.05	0.04	0.03
High Performance Devices																	
Delay	ps	0.87	0.74	0.64	0.54	0.51	0.40	0.34	0.29	0.25	0.21	0.18	0.15	0.13	0.11	0.10	0.08
Average Device Capacitance	relative	1.00	0.87	0.76	0.66	0.58	0.50	0.44	0.40	0.36	0.31	0.28	0.24	0.22	0.20	0.18	0.16
Circuit speedup: 1/delay	relative	1.00	1.18	1.36	1.61	1.71	2.18	2.56	3.00	3.48	4.14	4.83	5.80	6.69	7.91	8.70	10.88
ITRS Max Clock	relative	1.00	1.30	1.78	2.11	2.38	2.90	3.39	3.86	4.42	5.45	6.42	7.63	8.75	10.22	12.00	14.05
Vdd	volts	1.10	1.10	1.10	1.00	1.00	1.00	1.00	0.90	0.90	0.90	0.80	0.80	0.70	0.70	0.70	0.70
Vdd/Vt	ratio	5.64	6.55	6.67	6.10	4.22	6.62	6.85	6.08	5.39	5.49	4.82	4.10	3.50	3.48	3.41	3.37
Power Density @ Circuit Speedup	relative	1.00	1.29	1.65	1.77	2.13	2.95	3.95	4.19	5.52	7.41	7.54	10.19	10.17	13.91	17.12	23.47
Power Density @ Max Clock	relative	1.00	1.43	2.17	2.31	2.97	3.93	5.23	5.39	7.00	9.75	10.01	13.39	13.30	17.98	23.61	30.33
Energy/Operation	relative	1.000	0.867	0.756	0.542	0.478	0.413	0.367	0.268	0.238	0.208	0.147	0.129	0.090	0.081	0.072	0.063
Low Operating Power Devices																	
Delay	ps	1.52	1.33	1.17	1.03	0.90	0.79	0.79	0.61	0.53	0.47	0.41	0.36	0.32	0.28	0.24	0.21
Circuit speedup: 1/delay	relative	0.57	0.65	0.74	0.84	0.97	1.10	1.10	1.43	1.64	1.85	2.12	2.42	2.72	3.11	3.63	4.14
Vdd	volts	0.90	0.90	0.80	0.80	0.80	0.70	0.70	0.70	0.60	0.60	0.60	0.50	0.50	0.50	0.50	0.50
Vdd/Vt	ratio	3.13	2.97	2.81	2.95	2.90	3.10	3.00	3.03	2.33	2.40	2.39	2.10	2.09	2.07	2.06	2.03
Power Density @ Circuit Speedup	relative	0.38	0.48	0.48	0.59	0.77	0.73	0.83	1.21	1.16	1.47	1.86	1.66	2.11	2.79	3.64	4.56
Energy/Operation	relative	0.669	0.580	0.460	0.347	0.306	0.202	0.180	0.162	0.106	0.093	0.083	0.051	0.046	0.041	0.037	0.032
Note: units of "relative" represent values normalized to those of the 2005 high performance technology																	

Figure 6.1: ITRS roadmap logic device projections

Darpa Exascale Study

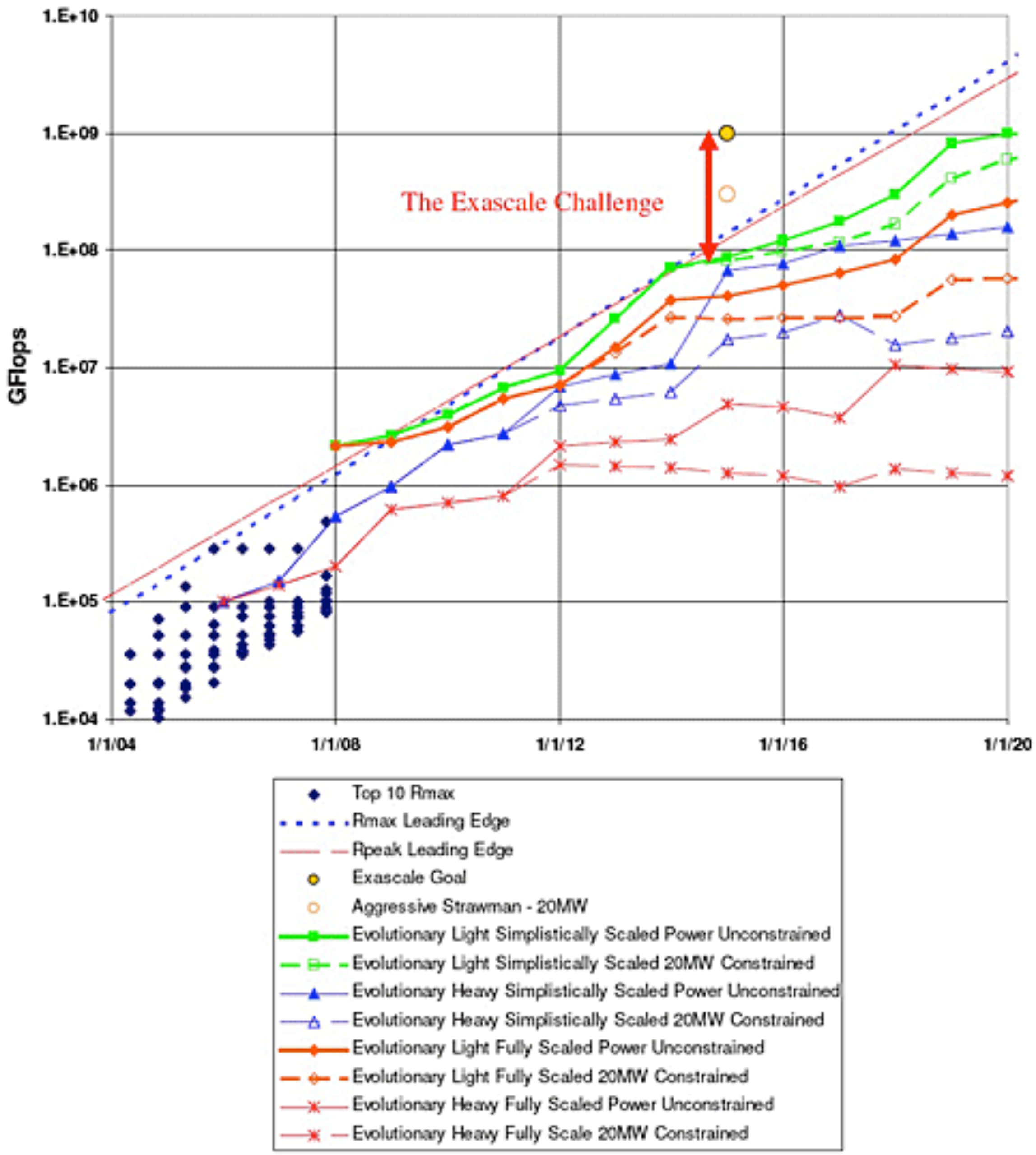
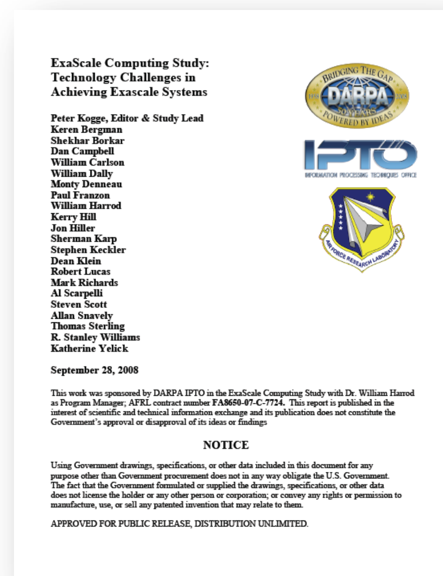


Figure 8.1: Exascale goals - Linpack.

Concluded that it will be a Major challenge to get to Sustained Exaops performance Levels by 2020



ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems

Peter Kogge, Editor & Study Lead
Keren Bergman
Shekhar Borkar
Dan Campbell
William Carlson
William Dally
Monty Denneau
Paul Franzon
William Harrod
Kerry Hill
Jon Hiller
Sherman Karp
Stephen Keckler
Dean Klein
Robert Lucas
Mark Richards
Al Scarpelli
Steven Scott
Allan Snavely
Thomas Sterling
R. Stanley Williams
Katherine Yelick

September 28, 2008

This work was sponsored by DARPA IPTO in the ExaScale Computing Study with Dr. William Harrod as Program Manager; AFRL contract number **FA8650-07-C-7724.** This report is published in the interest of scientific and technical information exchange and its publication does not constitute the Government's approval or disapproval of its ideas or findings

NOTICE

Using Government drawings, specifications, or other data included in this document for any purpose other than Government procurement does not in any way obligate the U.S. Government. The fact that the Government formulated or supplied the drawings, specifications, or other data does not license the holder or any other person or corporation; or convey any rights or permission to manufacture, use, or sell any patented invention that may relate to them.

APPROVED FOR PUBLIC RELEASE, DISTRIBUTION UNLIMITED.

Total System Concurrency

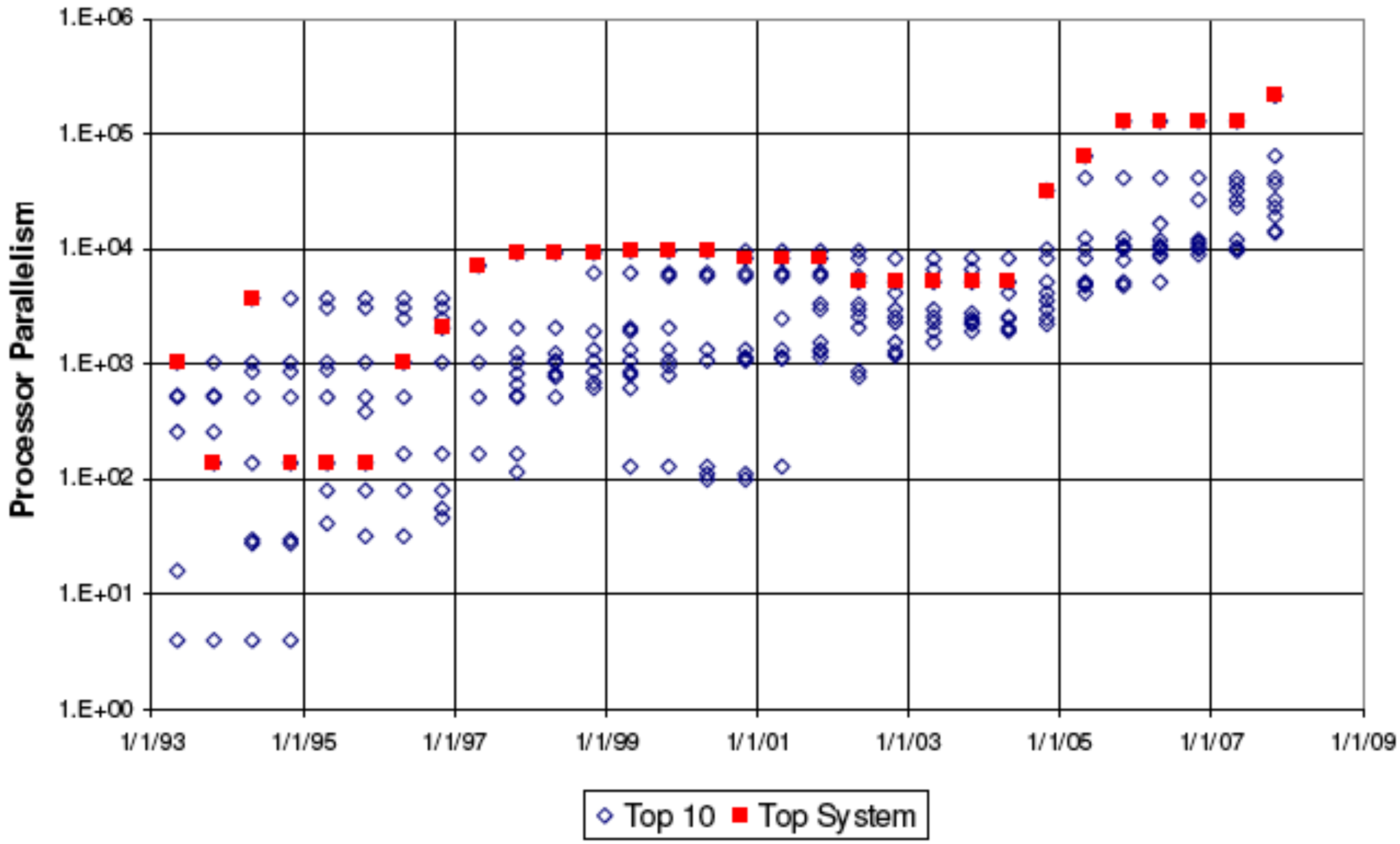


Figure 4.13: Processor parallelism in the Top 10 supercomputers.

Thread Level Concurrency

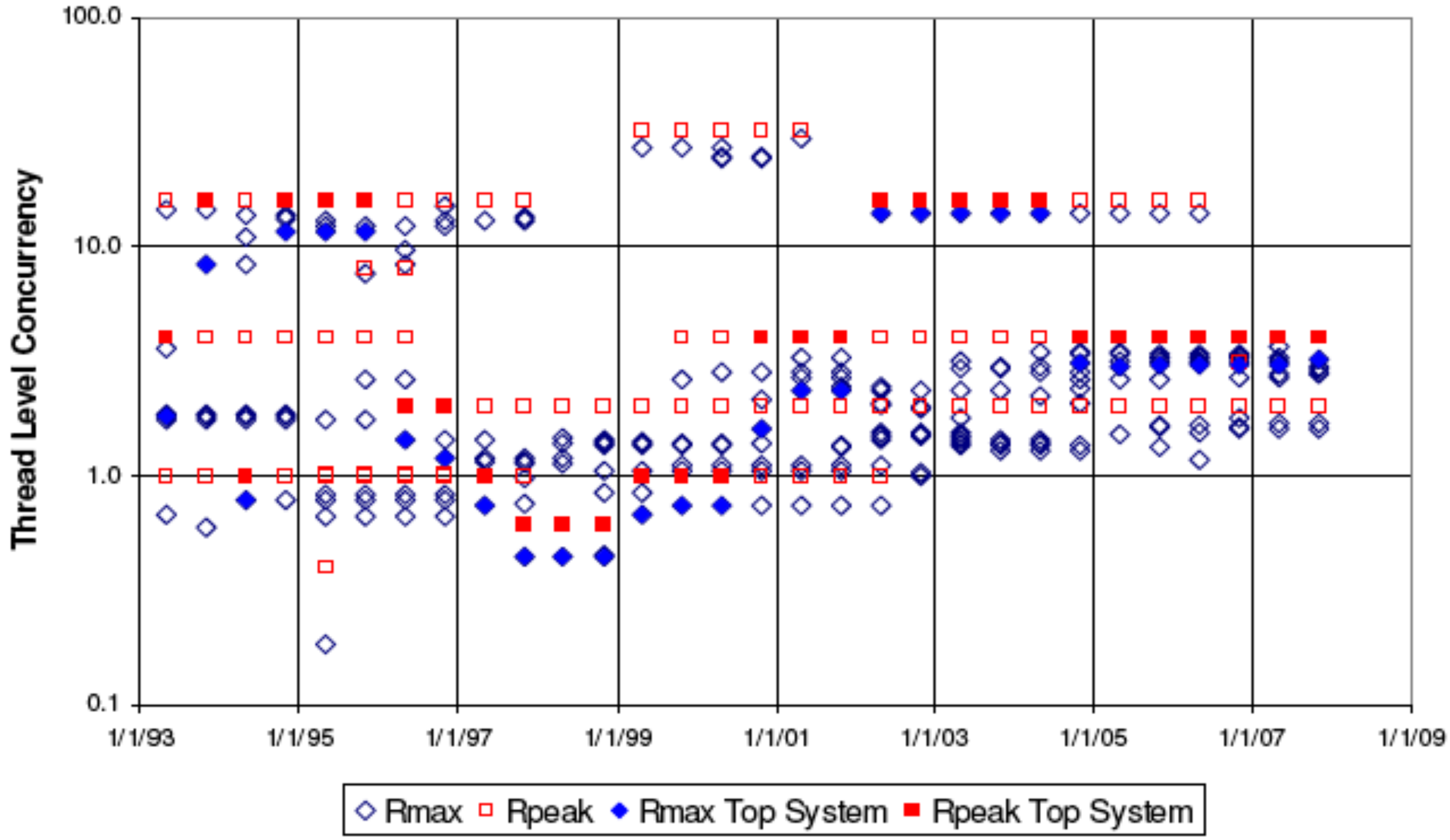


Figure 4.15: Thread level concurrency in the Top 10 supercomputers.

Parallelism and Locality Trends

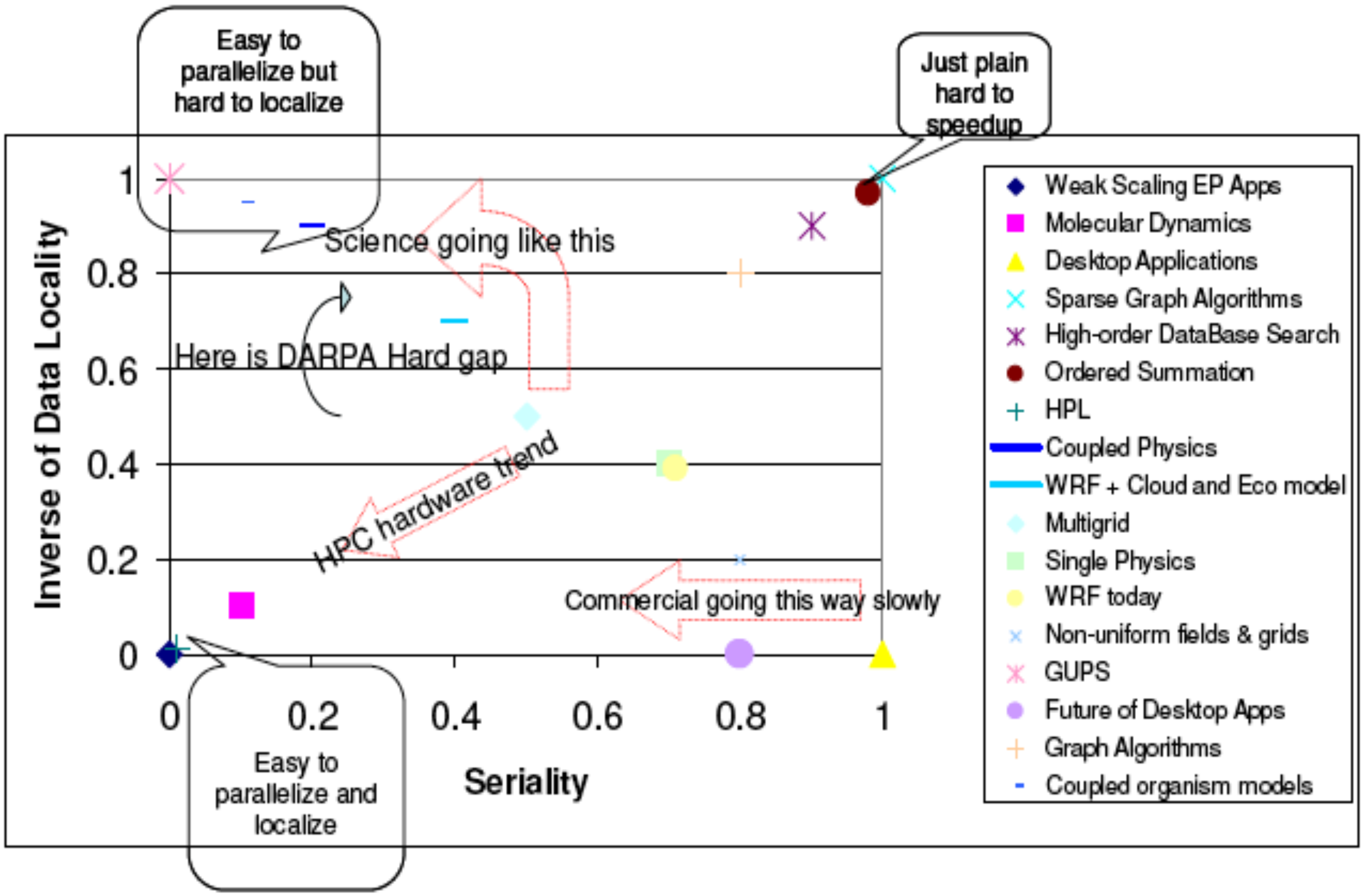


Figure 5.16: Future scaling trends

Applications Assumptions

	Departmental Class		Data Center Class	
	Range	"Sweet Spot"	Range	"Sweet Spot"
Memory Footprint				
System Memory	O(100TB) to O(1PB)	500 TB	O(1PB) to O(1EB)	50 PB
Scratch Storage	O(1PB) to O(100PB)	10 PB	O(100PB) to O(100EB)	2 EB
Archival Storage	>O(100PB) to O(100PB)	100 PB	>O(100EB)	100 EB
Communications Footprint				
Local Memory Bandwidth and Latency	Expect low spatial locality			
Global Memory Bisection Bandwidth	O(50TB/S) to O(1PB/s)	1PB/s	O(10PB/s) to O(1EB/s)	200PB/s
Global Memory Latency	Expect limited locality			
Storage Bandwidth	Will grow at faster rate than system peak performance or system memory growth			

Table 5.1: Summary applications characteristics.

Power Constrained Clock Rate

Clock = Power_Density/ (Capacitance_per_device * Transistor_Density * V^2_{dd})

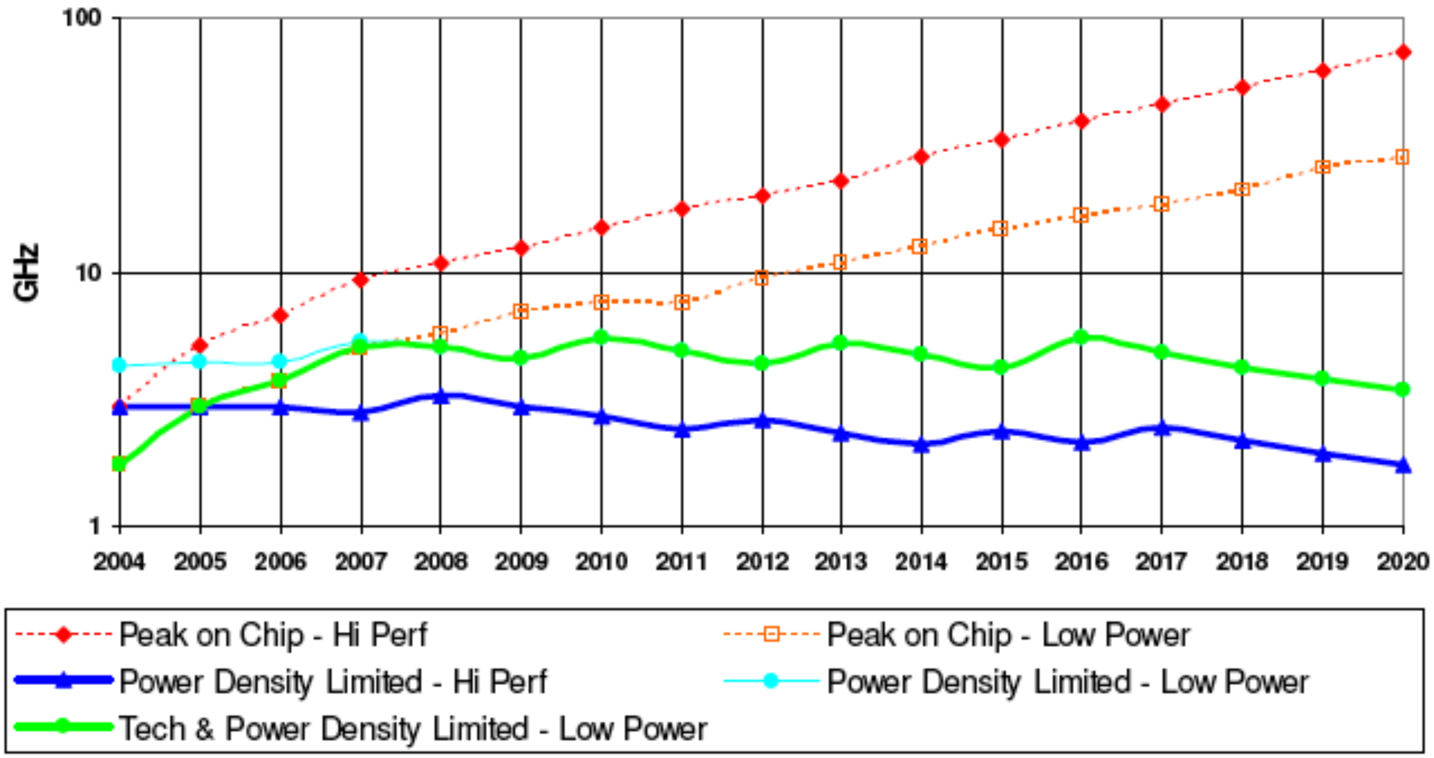


Figure 6.3: Power-constrained clock rate

Gflops per Watt
(0.1 ⇒ 100)

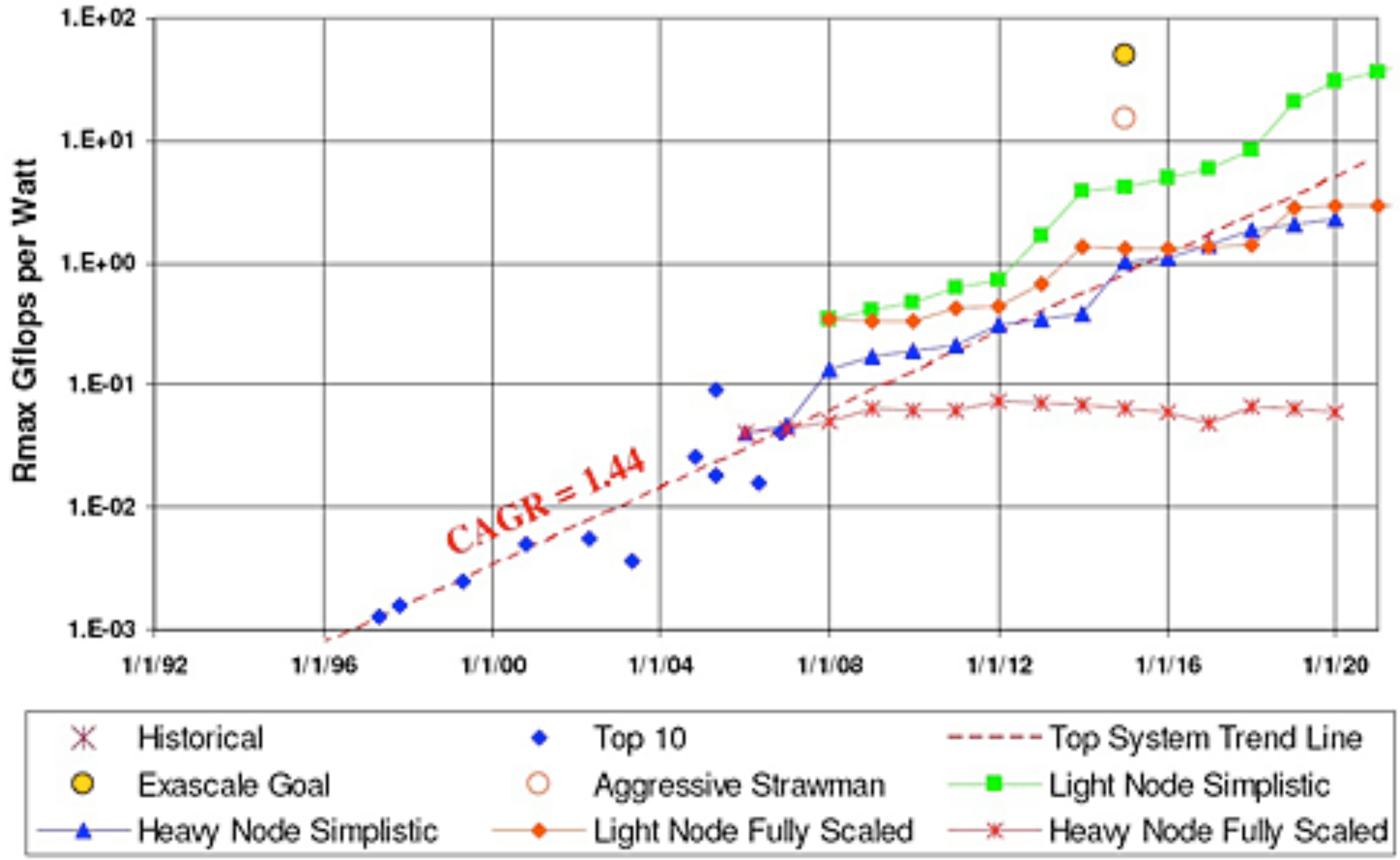


Figure 8.3: The power challenge for an Exaflops Linpack.

Power and FPUs to Reach Exaops

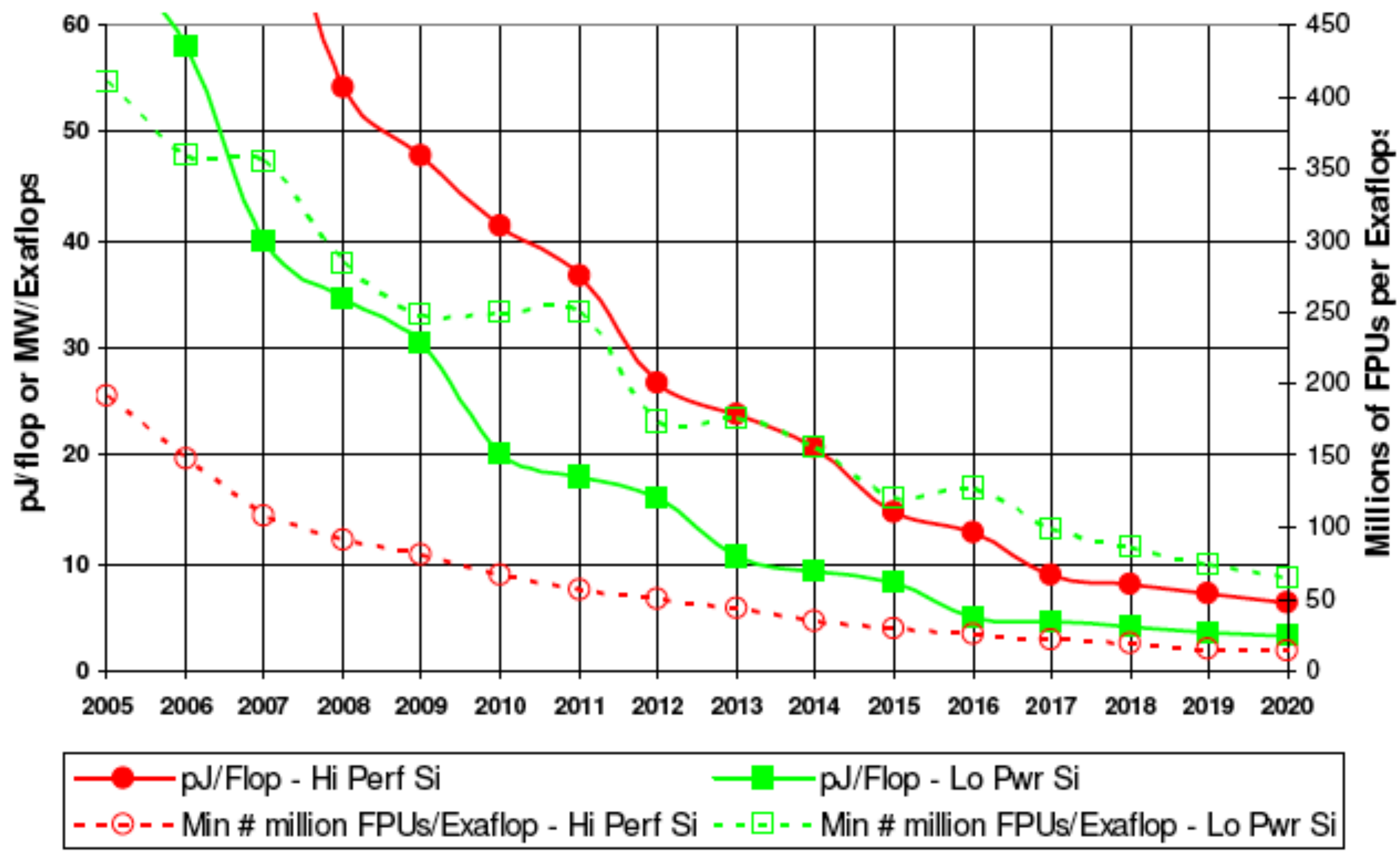


Figure 7.1: Projections to reach an Exaflop per second.

Interconnect Technology Roadmap

Technology	Density (wires/mm)	Power (pJ/bit)	Technology Readiness
Long-range on-chip copper	250	18 fJ/bit-mm	Demonstrated
Chip-to-chip copper	8	2 pJ/bit. Includes CDR	Demonstrated. Potential for scaling to 1 pJ/bit
Routed interconnect in 2015	n/a	2 pJ/bit router or non-blocking circuit switch 1 pJ/bit	roughly the same for packet
Optical State of Art (multi-mode)	10	9 pJ/bit. NOT including CDR	Demonstrated.
Optical (Single mode) in 2010	300	7.5 pJ/bit SOI waveguides PCB-embedded waveguide does not exist	Assumes lithographed
Optical (Single mode) in 2015	300	1.5 pJ/bit	At early research stage
Optical Routing		Add 0.1 pJ/bit (2010) for each switch	
Optical - temperature control			TEC cooler demonstrated
CNT bundles	1250	6 fJ/bit-mm	Undemonstrated

Table 6.8: Summary interconnect technology roadmap.

Heat Removal Approaches

Approach	Thermal Performance	Comments
Copper Heat Spreader	Thermal conductivity = 400 W/(m.K)	
Diamond	Thermal conductivity = 1000 - 2000 W/(m.K)	Expensive
Heat Pipe	Effective conductivity = 1400 W/(m.K)	Very effective
Thermal Grease	Thermal conductivity = 0.7 - 3 W/(m.K)	
Thermal vias with 10% fill factor	Effective Conductivity = 17 W/(m.K)	
Thermal Electric Coolers	Limited to less than 10 W/cm^2 and Consumes Power	
Carbon Nanotubes	Excellent	Early work only

Table 6.9: Internal heat removal approaches.

High-End Packaging Options

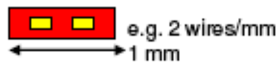
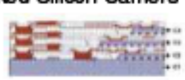
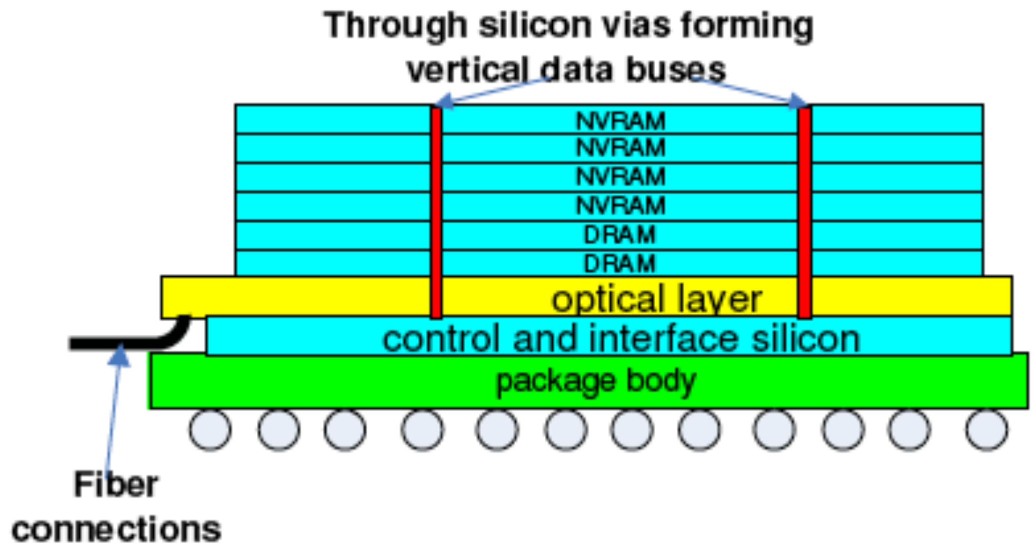
Approach	**Wires/mm or sq.cm**	**Bandwidth/mm** Routable signal pairs per mm per layer * # layers * bit-rate per signal pair	**Comments**
			
Laminate (Ball Grid Array) 	20 wires/mm/layer (2-4 signal layers) ~2,000 max total pin count	6 pairs/mm @ 30 Gbps = 180 – 720 Gbps/mm (1-4 signal layers) Package I/O: 500 pairs = 15 Tbps	1 mil line/trace presents practical limit. 1 mm BGA ball pitch
Silicon Carrier 	50 wires/mm/layer (2 signal layers) Has to be packaged for I/O	12 pair/mm @ 30 Gbps = 360 – 720 Gbps/mm (1-2 signal layers)	2 signal layers is practical limit.
3DIC Stack  	~10-40 wires/mm vertically around edge	Total: 100 – 200 pair @ 10 Gbps → 0.5 - 2 Tbps (assumes memory)	Limited interconnect performance
3D IC with Through Silicon Vias 	In excess of 10,000 vias per sq.mm.	In excess of 100,000 Tbps/sq.cm. Really determined by floorplan issues	Chip stack limited to 4-8 chips, depending on thermal and other issues
Stacked Packages 	1/mm on periphery	25 pairs total @ 10 Gbps → 250 Gbps	Not very applicable to high performance systems
Stacked Silicon Carriers 	Vertical connections @ 20 um pitch → 250,000 / sq.cm	62500 pairs @ 30 Gbps → 1900 Tbps / sq.cm	Limited by thermal and coplanarity issues.
Stacked Silicon Carriers 	Vertical connections @ 100 um pitch → 10,000 / sq.cm	2500 pairs @ 30 Gbps → 75 Tbps / sq.cm	Early demonstration only. Air cooled to < 117 W total.

Figure 6.36: Representative current and future high-end level 1 packaging.

3D Packaging Examples



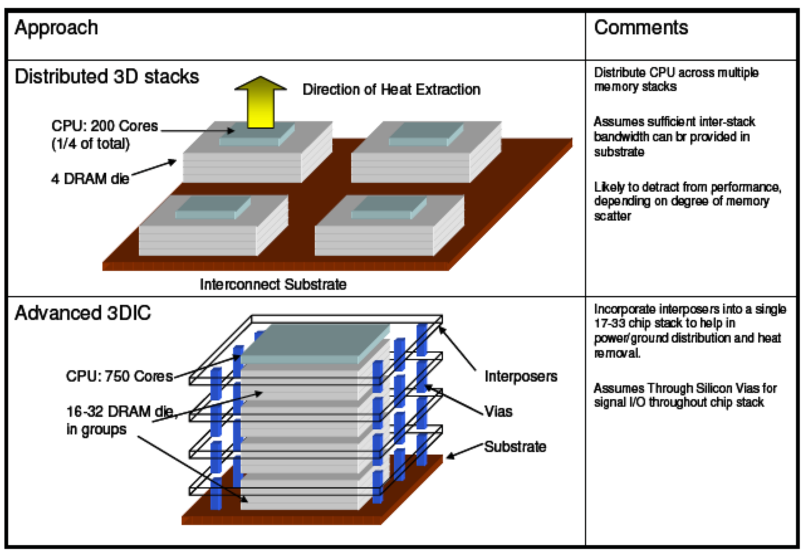


Figure 7.5: Potential directions for 3D packaging (A).

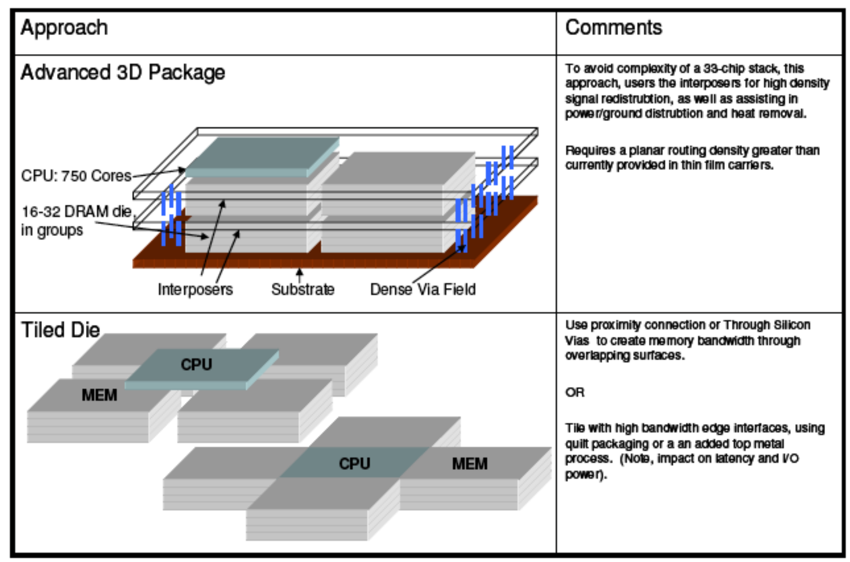


Figure 7.6: Potential directions for 3D packaging (B).

Secondary Storage Projections (Scratch at 25x , Archive at 200x)

	PB of Main Memory				
	0.006	0.5	3.6	50	300
Scratch Storage					
Capacity (EB)	1.2E-04	0.01	0.15	2	18
Drive Count	1.0E+01	8.3E+02	1.3E+04	1.7E+05	1.5E+06
Power (KW)	9.4E-02	7.8E+00	1.2E+02	1.6E+03	1.4E+04
Checkpoint Time (sec)	1.2E+03	1.2E+03	5.8E+02	6.0E+02	4.0E+02
Checkpoint BW (TB/s)	5.0E-03	4.2E-01	6.3E+00	8.3E+01	7.5E+02
Archival Storage					
Capacity (EB)	0.0012	0.1	7.2	100	600
Drive Count	1.0E+02	8.3E+03	6.0E+05	8.3E+06	5.0E+07
Power (KW)	9.4E-01	7.8E+01	5.6E+03	7.8E+04	4.7E+05

Table 7.1: Non-memory storage projections for Exascale systems.

Disk Characteristics

Year	Class	Capacity (GB)	RPM	B/W (Gb/s)	Idle Power(W)	Active Power (W)
2007	Consumer	1000	7200	1.03	9.30	9.40
2010	Consumer	3000	7200	1.80	9.30	9.40
2014	Consumer	12000	7200	4.00	9.30	9.40
2007	Enterprise	300	15000	1.20	13.70	18.80
2010	Enterprise	1200	15000	2.00	13.70	18.80
2014	Enterprise	5000	15000	4.00	13.70	18.80
2007	Handheld	60	3600	0.19	0.50	1.00
2010	Handheld	200	4200	0.38	0.70	1.20
2014	Handheld	800	8400	0.88	1.20	1.70

Table 6.6: Projected disk characteristics.

	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020
Chip Level Predictions															
Relative Max Power per Microprocessor	1.00	1.05	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10	1.10
Cores per Microprocessor	2.00	2.52	4.00	5.04	6.36	8.01	10.09	12.71	16.02	20.18	25.43	32.04	40.37	50.85	64.07
Flops per cycle per Core	2.00	2.00	4.00	4.00	4.00	4.00	4.00	4.00	4.00	8.00	8.00	8.00	8.00	8.00	8.00
Flops per cycle per Microprocessor	4.00	5.05	16.00	20.17	25.44	32.04	40.37	50.85	64.07	161.43	203.40	256.29	322.92	406.81	512.57
Power Constrained Clock Rate	1.00	0.94	1.10	0.99	0.90	0.81	0.88	0.79	0.71	0.80	0.72	0.83	0.73	0.65	0.59
Relative Rpeak per Microprocessor	1.00	1.19	4.39	4.98	5.76	6.48	8.88	9.99	11.42	32.38	36.79	52.86	58.73	66.08	75.51
Actual Rpeak per Microprocessor	9.60	11.44	42.15	47.82	55.26	62.16	85.27	95.93	109.64	310.82	353.20	507.46	563.85	634.33	724.94
ITRS Commodity Memory Capacity Growth	1.00	1.00	1.00	2.00	2.00	2.00	4.00	4.00	4.00	8.00	8.00	8.00	16.00	16.00	16.00
Required Memory Chip Count Growth	1.00	1.19	4.39	2.49	2.88	3.24	2.22	2.50	2.86	4.05	4.60	6.61	3.67	4.13	4.72
Relative Growth in BW per Memory Chip	1.00	1.00	1.00	2.00	2.00	2.00	4.00	4.00	4.00	8.00	8.00	8.00	16.00	16.00	16.00
BW Scaled Relative Memory System Power	1.00	1.19	4.39	4.98	5.76	6.48	8.88	9.99	11.42	32.38	36.79	52.86	58.73	66.08	75.51
Socket Level Predictions ("Socket" = Processor + Memory + Router)															
BW Scaled Relative per Socket Router Power	1.00	1.19	4.39	4.98	5.76	6.48	8.88	9.99	11.42	32.38	36.79	52.86	58.73	66.08	75.51
Simplistically Scaled per Socket Power	1.00	1.05	1.34	1.18	1.21	1.24	1.16	1.18	1.21	1.31	1.35	1.52	1.28	1.31	1.36
Fully Scaled Relative per Socket Power	1.00	1.12	3.53	3.15	3.71	4.28	4.88	5.63	6.67	20.77	25.15	44.44	35.32	42.11	51.64
Simplistically Scaled Relative Rpeak/Watt	1.00	1.14	3.29	4.21	4.74	5.21	7.66	8.45	9.43	24.75	27.20	34.88	45.98	50.26	55.42
Fully Scaled Relative Rpeak/Watt	1.00	1.06	1.24	1.58	1.55	1.51	1.82	1.77	1.71	1.56	1.46	1.19	1.66	1.57	1.46
Simplistically Scaled Rpeak/Watt	0.04	0.05	0.13	0.17	0.19	0.21	0.31	0.34	0.38	1.00	1.10	1.41	1.86	2.04	2.25
Fully Scaled Rpeak/Watt	0.04	0.04	0.05	0.06	0.06	0.06	0.07	0.07	0.07	0.06	0.06	0.05	0.07	0.06	0.06
Board and Rack Level Concurrency Predictions															
Maximum Sockets per Board	4	4	4	4	8	8	8	8	8	16	16	16	16	16	16
Maximum Boards per Rack	24	24	24	24	32	32	32	32	32	32	32	32	32	32	32
Maximum Sockets per Rack	96	96	96	96	256	256	256	256	256	512	512	512	512	512	512
Maximum Cores per Board	8	10	16	20	51	64	81	102	128	323	407	513	646	814	1025
Maximum Cores per Rack	192	242	384	484	1628	2050	2584	3254	4101	10331	13018	16402	20667	26036	32804
Maximum Flops per cycle per Board	16	20	64	81	204	256	323	407	513	2583	3254	4101	5167	6509	8201
Maximum Flops per cycle per Rack	384	484	1536	1936	6513	8201	10336	13018	16402	82650	104142	131218	165336	208285	262436
Board and Rack Level Power Predictions															
Max Relative Power per Rack	1	1	1	2	2	2	4	4	4	8	8	8	16	16	16
Simplistic Power-Limited Sockets/Rack	96	92	72	96	158	155	256	256	256	512	512	507	512	512	512
Fully Scaled Power-Limited Sockets/Rack	96	86	27	61	52	45	79	68	58	37	31	17	43	36	30
Simplistically Scaled Relative Rpeak per Rack	96	109	316	478	911	1001	2274	2558	2924	16577	18838	26788	30072	33831	38664
Fully Scaled Relative Rpeak per Rack	96	102	119	304	298	291	699	681	658	1197	1123	914	2554	2410	2246
System Predictions: Power Unconstrained, Gradual Increase in Affordable Racks to Max of 600															
Max Affordable Racks per System	155	200	250	300	350	400	450	500	550	600	600	600	600	600	600
Max Cores per System	29760	48441	9.6E+04	1.5E+05	5.7E+05	8.2E+05	1.2E+06	1.6E+06	2.3E+06	6.2E+06	7.8E+06	9.8E+06	1.2E+07	1.6E+07	2.0E+07
Max Flops per cycle per System	59520	96882	3.8E+05	5.8E+05	2.3E+06	3.3E+06	4.7E+06	6.5E+06	9.0E+06	5.0E+07	6.2E+07	7.9E+07	9.9E+07	1.2E+08	1.6E+08
Simplistically Scaled System Rpeak (GF)	1.0E+05	1.5E+05	5.4E+05	9.8E+05	2.2E+06	2.7E+06	7.0E+06	8.7E+06	1.1E+07	6.8E+07	7.7E+07	1.1E+08	1.2E+08	1.4E+08	1.6E+08
Fully Scaled System Rpeak (GF)	1.0E+05	1.4E+05	2.0E+05	6.2E+05	7.1E+05	7.9E+05	2.1E+06	2.3E+06	2.5E+06	4.9E+06	4.6E+06	3.7E+06	1.0E+07	9.9E+06	9.2E+06
System Power (MW)	2.5	3.2	4.0	9.7	11.3	12.9	29.0	32.3	35.5	77.4	77.4	77.4	154.8	154.8	154.8
System Predictions: Power Constrained to 20 MW															
Maximum Racks	155	200	250	300	350	400	310	310	310	155	155	155	78	78	78
Simplistically Scaled System Rpeak (GF)	1.E+05	1.E+05	5.E+05	1.E+06	2.E+06	3.E+06	5.E+06	5.E+06	6.E+06	2.E+07	2.E+07	3.E+07	2.E+07	2.E+07	2.E+07
Fully Scaled System Rpeak (GF)	1.E+05	1.E+05	2.E+05	6.E+05	7.E+05	8.E+05	1.E+06	1.E+06	1.E+06	1.E+06	1.E+06	1.E+06	1.E+06	1.E+06	1.E+06

Figure 7.10: Heavy node strawman projections.

Aggressive Strawman

Level	What	Perf	Power	RAM
FPU	FPU, regs,. Instruction-memory	1.5 Gflops	30mW	
Core	4FPUs, L1	6 Gflops	141mW	
Processor Chip	742 Cores, L2/L3, Interconnect	4.5 Tflops	214W	
Node	Processor Chip, DRAM	4.5Tflops	230W	16GB
Group	12 Processor Chips, routers	54Tflops	3.5KW	192GB
rack	32 Groups	1.7 Pflops	116KW	6.1 TB
System	583 racks	1 Eflops	67.7MW	3.6PB

Table 7.3: Summary characteristics of aggressively designed strawman architecture.

	Exascale System Class				
Characteristic	Exaflops Data Center	20 MW Data Center	Department	Embedded A	Embedded B
Top-Level Attributes					
Peak Flops (PF)	9.97E+02	303	1.71E+00	4.45E-03	1.08E-03
Cache Storage (GB)	3.72E+04	11,297	6.38E+01	1.66E-01	4.03E-02
DRAM Storage (PB)	3.58E+00	1	6.14E-03	1.60E-05	1.60E-05
Disk Storage (PB)	3.58E+03	1,087	6.14E+00	0.00E+00	0.00E+00
Total Power (KW)	6.77E+04	20,079	116.06	0.290	0.153
Normalized Attributes					
GFlops/watt	14.73	14.73	14.73	15.37	7.07
Bytes/Flop	3.59E-03	3.59E-03	3.59E-03	3.59E-03	1.48E-02
Disk Bytes/DRAM Bytes	1.00E+03	1.00E+03	1.00E+03	0	0
Total Concurrency (Ops/Cycle)	6.64E+08	2.02E+08	1.14E+06	2968	720
Component Count					
Cores	1.66E+08	50,432,256	2.85E+05	742	180
Microprocessor Chips	223,872	67,968	384	1	1
Router Chips	223,872	67,968	384	0	0
DRAM Chips	3,581,952	1,087,488	6,144	16	16
Total Chips	4,029,696	1,223,424	6,912	17	17
Total Disk Drives	298,496	90,624	512	0	0
Total Nodes	223,872	67,968	384	1	1
Total Groups	18,656	5,664	32	0	0
Total racks	583	177	1	0	0
Connections					
Chip Signal Contacts	8.45E+08	2.57E+08	1.45E+06	2,752	2,752
Board connections	1.86E+08	5.65E+07	3.19E+05	0	0
Inter-rack Channels	2.35E+06	7.14E+05	8,064	0	0

Table 7.10: Exascale class system characteristics derived from aggressive design.

Systems Scaling Projections

Begin Full System Delivery (Yr)	2004	2007	2012	2015	2019
Design Parameters	**BG/L**	**BG/P**	**25PF**	**300PF**	**1200PF**
Cores / Node	2	4	8-24	32-64-128	96-128-500
Clock Speed (GHz)	0.7	0.85	1.6-4.1	2.3-4.8	2.8-6.0
Flops / Clock / Core	4	4	8-32	8-32	16-64
Nodes / Rack	1024	1024	100-1024	256-1024	256-1024
Racks / Full System Config	64	72	128-350	128-400	256-400
MB RAM/core	256	512	1024-4096	1024-4096	1024-4096
Total Power	2.5MW	4.8MW	8MW-20MW	20MW-50MW	30MW-80MW
Flops / Node (GF)	5.6	14	128-640	640-2000	2000-6000
Flops / Rack (TF)	5.7	14	200-400	400-1200	1600-4800
LB Concurrency	5.E+05	1.E+06	1M-2M	10M-100M	400M-1000M
Full System					
Total Cores (Millions)	0.13	0.3	.3M-1.2M	1M-10M	4M-30M
Total RAM (TB)	33.6	151	2,000-4,400	3,000-10,000	5,000-25,000
Total Racks	64	72	128-350	128-400	256-400
Peak Flops System (PF)	**0.37**	**1**	**25**	**300**	**1200**

The Bottom Line

- Levels of concurrency ($10^6 \Rightarrow 10^9$)
- Clock rate of Core (1-4 GHz ⇒ 1-4 GHz)
- RAM per Core (1-2GB now to 1-4GB)
- Total Number of cores (200K ⇒ 100M)
- Number of cores per node (8 ⇒ 64-512)
- Aggressive Fault Management in HW and SW
- I/O channels ($>10^3 \Rightarrow 10^5$)
- Power Consumption (10MW ⇒ 40MW-150MW)
- Programming Model (MPI ⇒ MPI + X)

Parallel Programming Models: Twenty Years and Counting

- In large-scale scientific computing today essentially all codes are message passing based. Additionally many will use some form of multithreading on SMP or multicore nodes.
- Multicore is challenging programming models but there has not yet emerged a dominate model to augment message passing
- There is a need to identify new hierarchical programming models that will be stable over long term and can support the concurrency doubling pressure

Quasi Mainstream Programming Models

- C, Fortran, C++ and MPI
- OpenMP, pthreads
- (CUDA, RapidMind, Cn) → OpenCL
- PGAS (UPC, CAF, Titanium)
- HPCS Languages (Chapel, Fortress, X10)
- HPC Research Languages and Runtime
- HLL (Parallel Matlab, Grid Mathematica, etc.)

Chip Count Trends

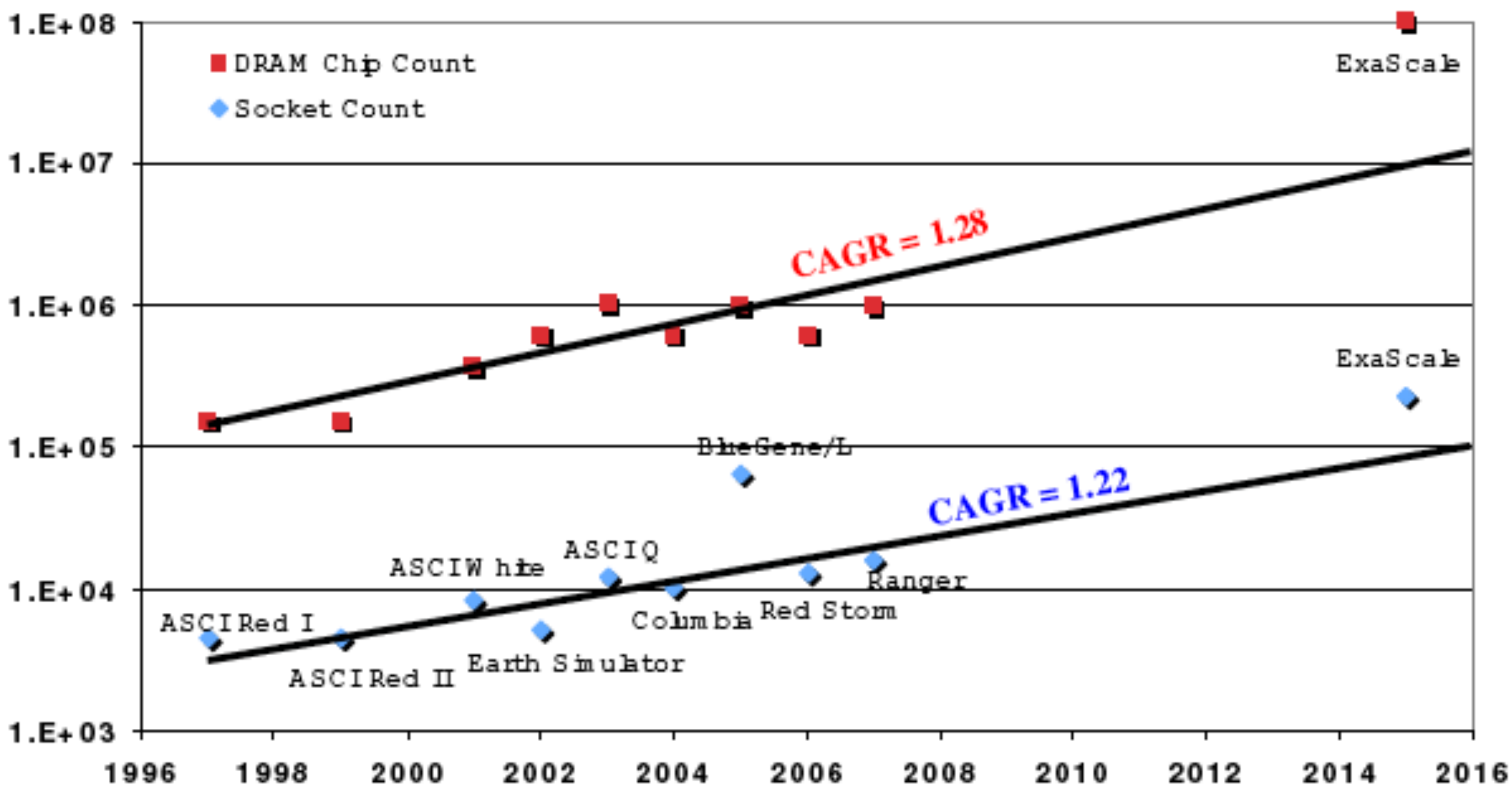


Figure 6.37: Estimated chip counts in recent HPC systems.

Computational Science and HPC Software-Development in Europe

Thomas Lippert / Bernd Mohr
Forschungszentrum Jülich, JSC
and Gauss Centre for Supercomputing e.V.

1st Workshop of the International Exascale Software Project (IESP), Santa Fé, April 7-8, 2009


2009-04-07 CEST 15:14:28
IBM
JUGENE

Thanks to

- Jean-Yves Berthou (EDF)
- Michel Marechal (ESF, Lincei Initiative, CSEC)
- Achim Bachem and all friends from PRACE
- Catherine Riviere (GENCI, PRACE)
- Peter Michielse (PRACE WP6)
- Herbert Huber (PRACE-STRATOS)
- Wolfgang Nagel (Gauß Alliance)
- Stefan Heinzel (DEISA)
- Kimmo Koski (HET, COSI-HPC)
- Wanda Andreoni et al. (CECAM)
- Martyn Guest (Editor of HET/HPC-EUR–Scientific Case)

  and many others

Mitglied der Helmholtz-Gemeinschaft

JÜLICH FORSCHUNGSZENTRUM

Science Fields & Drivers in Europe

From
- HET Scientific Case
- PRACE Initiative
- Linceï Initiative Report

HET: Scientific Case White Paper

Area	Application	Science Challenges & Potential Outcomes
Weather, Climatology and Earth Sciences	**Climate change**	Quantify uncertainties on the degree of warming and the likely impacts by increasing the capability and complexity of 'whole earth system' models that represent the scenarios for our future climate (**IPCC**).
	Oceanography	Build the most efficient modelling and prediction systems to study, understand and predict ocean properties and variations at all scales, and develop **economically relevant applications** to inform policy
	Meteorology, Hydrology	Predict weather and flood events with **high socio-economic and environmental impact** within a few days. Understand and predict the quality of air at the earth's surface; development of advanced real-time forecasting systems for early enough warning and practical mitigation in the case of pollution crisis.
	Earth Sciences	Challenges span a range of disciplines and have scientific and social implications, such as the **mitigation of seismic hazards**, treaty verification for nuclear weapons, and increased discovery of economically recoverable petroleum resources and monitoring of waste disposal. Increased computing capability will make it possible to address the issues of resolution, complexity, duration, confidence and certainty.
Astrophysics, HEP and Plasma Physics	**Astrophysics**	Deal with systems and structures which span a large range of different length and time scales; almost always non-linear coupled systems differential equations have to be integrated, in 3 spatial dimensions and explicitly in time, with rather complex material functions as input. Grand challenges range from formation of stars and planets to questions concerning the **evolution of the Universe** as a whole. Evaluate the huge mount of data expected from future space experiments such as the European Planck Surveyor satellite.
	Element. Part. Physics	Quantum field theories like QCD (quantum chromodynamics) are the topic of intense theoretical and experimental research by a large and truly international community involving large European centers like **GSI** and **CERN**. This research promises a much deeper understanding of the standard model as well as nuclear forces, but is also to discover yet unknown physics beyond the standard model.
	Plasma physics	The science and technology challenge raised by the construction of the magnetic confinement fusion reactor **ITER** calls for a major theory and modelling activity. Both the success of the experiment and its safety rely on such simulators. The quest to realize thermonuclear fusion by magnetically confining a high temperature plasma poses computationally most challenging problems of nonlinear physics.
Materials Science, Chemistry and Nanoscience	**Understanding Complex Materials**	The determination of electronic and transport properties is central to many devices in the electronic industry and hence to progress in the understanding of technologically relevant materials. Simulations of nucleation, growth, self-assembly and polymerization for design and performance of many diverse materials e.g., rubbers, paints, fuels, detergents, functional organic materials, cosmetics and food. Multiscale descriptions of the mechanical properties of materials to determine the relation between process, conditions of use and composition e.g., in **nuclear energy production**. Such simulations are central to the prediction of the lifetime of high performance materials in energy technology.
	Understanding Complex Chemistry	**Catalysis** is a major challenge in the chemistry of complex materials, with many applications in industrial chemistry. The knowledge of atmospheric chemistry is crucial for environmental prediction and protection (clean air). Improving the knowledge of chemical processing would improve the durability of chemicals. Supra molecular assemblies open new possibilities for the extraction of heavy elements from spent nuclear fuels. In biochemistry, a vast number of reactions in the human body are not understood in any detail. A key step for **clean fuels** of the future requires the realistic treatment of supported catalytic nanoparticles.
	Nanoscience	The advance of faster information processing or the development of **new generations of processors** requires the shrinking of devices, which leads inevitably towards nanoelectronics. Moreover, many new devices, such as nanomotors can be envisioned, which will require simulation of mechanical properties at the nanolevel. Composite high performance materials in the fields e.g. adhesion and coatings will require an atomistic based description of nanorheology, nanofluidics and nanotribology.

HET: Scientific Case White Paper II

JÜLICH FORSCHUNGSZENTRUM

Area	Application	Science Challenges & Potential Outcomes
Life sciences	Systems Biology	The use of increasingly sophisticated models to represent the entire behaviour of cells, tissues, and organs, or to evaluate degradation routes predicting the final excretion product of any drug. **In silico cell**.
	Chromatine Dynamics	The organization of DNA in nucleosomes largely modifies the accessibility of transcription factors recognition sites playing then a key role in the regulation of gene function. The understanding of nucleosome dynamics will be crucial to understand the mechanism of **gene regulation**.
	Large Scale Protein Dyn.	The study of large conformational changes in proteins. Major challenges appear in the simulation of protein missfolding, unfolding and refolding (understanding of prion-originated pathologies).
	Protein association and aggregation	One of the greatest challenges is the simulation of crowded **"not in the cell" protein environments**. To be able to represent "in silico" the formation of the different protein complexes associated with a signalling pathway opens the door to a better understanding of cellular function and to the generation of new drugs.
	Supramolecular Systems	The correct representation of protein machines is still out of range of European groups using current simulation protocols and computers. The challenge will be to analyze systematically how several of these machines work e.g., ribosome, topoisomerases, polymerases.
	Medicine	Genome sequencing, massive genotyping studies are providing massive volumes of information e.g. the simulation of the determinants triggering the development of **multigenic-based diseases** and the prediction of secondary effects related to bad metabolism of drugs in certain segments of population.
Engineering	Helicopter Simulation	The European helicopter industry has a strong tradition of innovation in technology and design. Computational Fluid Dynamics (CFD) based simulations of aerodynamics, aeroacoustics and coupling with dynamics of rotorcraft play a central role and will have to be improved further in the design loop.
	Biomedical Flows	Biomedical fluid mechanics can improve healthcare in many areas, with intensive research efforts in the field of the human circulatory system, the artificial heart or **heart valve prostheses**, the respiratory system with nose flow and the upper and lower airways, and the human balance system.
	Gas Turbines & Internal Combustion Engines	Scientific challenges in gas turbines or piston engines are numerous. First, a large range of physical scales should be considered from fast chemical reaction characteristics (reaction zone thicknesses of about tens of millimetres, 10^{-6} s), pressure wave propagation up to burner scales (tens of cm, 10^{-2} s) or system scales.
	Forest Fires	The development of reliable numerical tools able to model and predict **fire evolution** is critically important in terms of safety and protection fire fighting and could help in real time disaster management.
	Green Aircraft	ACARE 2020 provides the politically agreed targets for an acceptable maximum impact of air traffic on people and environment, while allowing the constantly increasing amount of air travel. The goals deal with a reduction of exhaust gas and noise. Air traffic will increase by a factor of 3, accidents are expected to go down by 80%. Passenger expense should drop (50%) and flights become largely weather independent. The "**Green Aircraft**" is the answer of the airframe as well as engine manufacturing industry.
	Virtual Power Plant	Safe production of high quality and **cost effective energy** is one of the major concerns of Utilities. Several challenges must be faced, amongst which are extending the lifespan of power plants to 60 years, guaranteeing the optimum fuel use and better managing waste.

PRACE: Support of Science Communities

European Organisations and Research Communities	
EFDA	**The European Fusion Development Agreement foresees a huge demand for HPC including tier-0. It is interested in cooperation with PRACE regarding benchmarking and code-scaling and provides the HPC-related requirements for Fusion community.**
EMBL-EBI	**The Euro Bioinformatics Institute within the European Molecular Biology Laboratory foresees huge demands for HPC resources in the future and is interested in investigating access policies to European tier-0 systems for life scientists.**
ENES	**The European Network for Earth System Modeling has contributed to the scientific case for HPC in Europe and will continue to promote the involvement of the European climate modelling community in PACE. ENES involvement includes porting of applications on prototype systems of PACE and defining of facility requirements.**
ESA	**ESA is the European Space Agency. The Space and in particular Earth Observation communities have very demanding HPC applications. ESA is pleased to collaborate with PRACE on specific applications.**
ESF	**The European Science Foundation is interested to contribute to PRACE, in particular to peer-review process dissemination activities and computer technologies beyond 2010.**
MOLSIMU	**MOLSIMU, a COST action on Molecular Simulations to Nanoscale Experiments, is offering its support for PRACE by porting their major applications to the prototype systems installed by PACE**
Psi-k Network	**The Psi-k network is the European Umbrella Network for Electronic Structure Calculations. Several groups within Psi-k are interested to port their ab-initio codes like CPMD, VASP, SIESTA, CASTEP, ABINIT, and Wien 2k on the prototype systema of PRACE.**

PRACE: Support of Research Infrastructures

DEISA	EU-Project	**DEISA currently deploys and operates the European Supercomputing Grid infrastructure to enable capability computing across remote computing platforms and data repositories at a continental scale.**
HPC-Europa	EU-Project	**HPC-Europa is a pan-European Research Infrastructure on HPC providing HPC access and scientific support to researchers in challenging computational activities.** **HPC-Europa expresses its interest in cooperating in the areas of access technologies and integrated advanced computational services.**
OMII-Europe	EU-Project	**OMII-Europe is the interoperability project in Europe providing open standards based interoperability components on top of the four major Grid middleware systems in the world.**
EGI	EU-Project Prop.	**The consortium of EGI aims at establishing a sustainable Grid infrastructure in Europe, coordinating national Grid initiatives.**

Linceï Initiative (2007-2009)

JÜLICH
FORSCHUNGSZENTRUM

EUROPEAN SCIENCE FOUNDATION
SETTING SCIENCE AGENDAS FOR EUROPE

FORWARD LOOK

EUROPEAN COMPUTATIONAL SCIENCE: THE "LINCEI INITIATIVE": FROM COMPUTERS TO SCIENTIFIC EXCELLENCE

Computational sciences and computer simulations in particular, are playing an ever growing role in fundamental and applied sciences. The aim of this Forward Look is to develop a vision on how computational sciences will evolve in the coming 10 to 20 years. Based on a scenario of how this field will evolve and on the needs of the scientific community, a strategy will be presented aimed at structuring software and hardware support and development at the European level.

http://ccp2007.ulb.ac.be/FL-Lincei.pdf

Linceï Initiative: Steering Committee

Doctor Vassilis Pontikis, *Chair*,
Commissariat à l'Énergie Atomique, Saclay , Gif-sur-Yvette, France
Professor Carmen N. Afonso, PESC rapporteur,
Consejo Superior de Investigaciones Cientifica, Instituto de Optica, Madrid, Spain
Professor Isabel Ambar, LESC rapporteur,
Directora Instituto de Oceanografia Faculdade de Ciências da Universidade de Lisboa
Professor Kenneth Badcock,
Dept. of Engineering, The University of Liverpool, Liverpool, United Kingdom
Professor Giovanni Ciccotti,
Dept. of Physics, Università "La Sapienza", Roma, Italy
Professor Peter H. Dederichs,
Institut für Festkörperforschung, Jülich Research Centre, Jülich, Germany
Doctor Paul Durham,
Daresbury Laboratory, Warrington, United Kingdom
Professor Franco Antonio Gianturco,
Dept. of Chemistry, Università "La Sapienza", Roma, Italy
Professor Volker Heine,
Cavendish Laboratory (TCM), Cambridge University, Cambridge, United Kingdom
Professor Ralf Klessen,
Institute für Astrophysik, Zentrum für Astronomie der Universität Heidelberg, Heidelberg, Germany
Professor Peter Nielaba,
Lehrstuhl für Theoretische Physik, Fachbereich Physik, Universität Konstanz, Konstanz, Germany
Doctor Simone Meloni, Scientific Secretary,
Consorzio per le Applicazioni del Supercalcolo per Università e Ricerca - CASPUR, Roma, Italy

Linceï Initiative: Six Fields Addressed

JÜLICH FORSCHUNGSZENTRUM

Astrophysics

- Institut für Theoretische Astrophysik, Heidelberg (DE), Dec. 1st-2nd 2006

Fluid Dynamics

- Daresbury Lab., Warrington (UK), Nov. 29th-30th 2006

Meteorology and Climatology

- Swiss Supercomputing Centre, Manno (CH), Jan. 27th 2007

Life sciences

- Chilworth Manor, Southampton (UK), Nov. 19th-21st 2006

Material Science and Nanotechnology

- Jülich Research Centre, Jülich (DE), Nov. 13th-14th 2006

Quantum Molecular Sciences

- Accademia dei Lincei, Rome, Nov. 25th-26th 2006

- State of infrastructure for scientific computing
- Needs in relation to future challenges, in 10-20 year timeframe

Some EU Scientific and Engineering Codes

(From Lincei Forward Look Report (for the ESF))

Name	Scientific Area	Brief Description	Licensing	Users
ABINIT	Condensed Matter	DFT+PW+Pseudopotentials	Free	~1000
ESPResSo	Condensed Matter	coarse grained off-lattice	Free	~20 groups
VASP	Condensed Matter	DFT+PW+Pseudopotentials	Licensed	800 li
CP2K	Condensed Matter	DFT-(gausssian+PW)+classical	Free	~100
CPMD	Condensed Matter	DFT+PW+Pseudopotentials	Licensed, free acad.	>1000
Wien2K	Condensed Matter	Full-electrons Augmented PW	Licensed	~1100
Quantum Espresso	Condensed Matter	DFT+PW+Pseudopotentials	Free	~700
Code_Aster	Engineering	Mechanical and thermal analysis	Free	300 (EDF) 22k downl.
Code_Saturne	Fluid Dynamics +heat transfer+combustion	Incompressible+expandable	Free	80 (EDF) + 25 groups
OpenFOAM	Fluid Dynamics +Structural Mechanics	Finite volume on unstructured grid	Free, fee for support	~2000
Salome	framework for multiphysics	Used in engineering	Free	50 (EDF) + 21 groups
COSMO-Model	Climatology, Meteorology and scientific research	Operational Weather forecasting	Special agreement	7 Centres 80 groups

Importance Hierarchy

IDEAS	>>	CODES	>>	HARDWARE
Science/algorithms		*(software)*		*(computer)*

Comments from FL-Linceï-Report

- Current [application] software is very complex
- Typical size is 400000 lines of code and 2500 routines/classes
- Large number of variables pass through the code in obscure data flow
- Few strictly object oriented (OpenFOAM, C++, CP2K, FORTRAN95
- Will be confronted with a software sustainability crisis
- Will be very difficult to adapt most existing complex codes to the coming massively parallel computers
- Structure of many of the codes strongly dependent on the parallel programming paradigm adopted in the early stage of the development
- Current shift from hundreds to tens of thousands of CPUs will require a change in the parallelization scheme
- Very difficult to implement in such very complex community codes

FL-Linceï-Report

Findings

- Bottleneck is the support to software, effort mainly focused on Hardware
- Less support is given to the writing, maintenance and dissemination of sc. codes
- Scientific computer programs do not comply with best practices in programming
- Successful efforts in all the technical areas required to support scientific computing: hardware, system and application software

Recommendations

- National science funding agencies in Europe must undertake a coordinated and sustained effort in scientific software development
- Set up a Computational Sciences Expert Committee (CSEC) attached to ESF which would speak for the whole community of computational sciences.
- Its purpose would be to start setting up a durable plan for European cooperation in each of the fields of science using computers

Example 1: MD for Radiation Hard Materials

Ian J. Bush, Ilian T. Todorov, CCSRC Daresbury, UK

DL-POLY3 classical molecular dynamics

First time on more than 1000 processors

Radiation damage in a fluoritized Zirconium pyrochlore

100 keV recoil of one Uranium atom after alpha decay

15 million particles, supercell very large

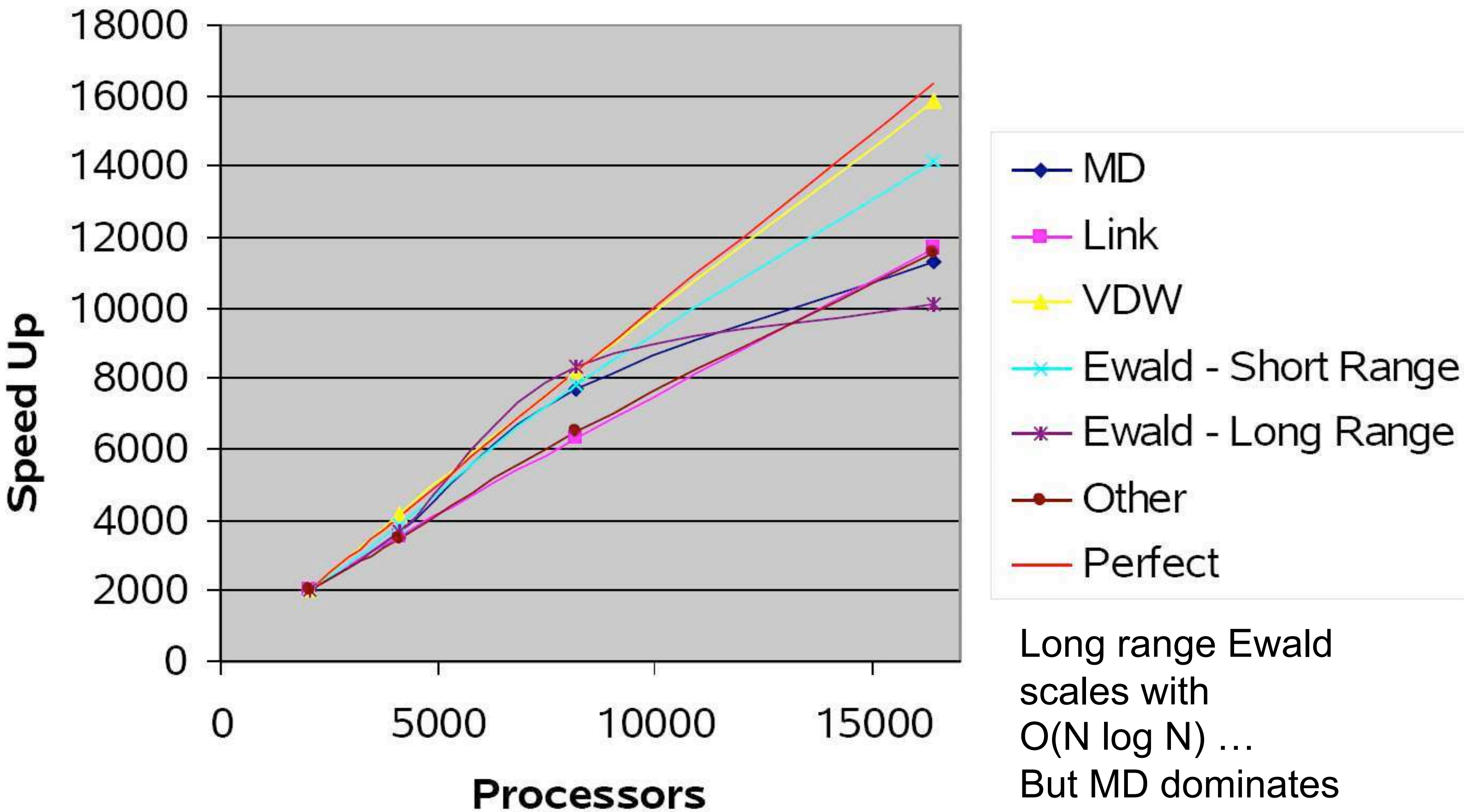
Forces: short range, van der Waals, Coulomb

Smooth particle-mesh Ewald algorithm → FFT

Implementation on BGL

Scaling DL_Poly3

Substantial improvements by performance analysis tool Scalasca

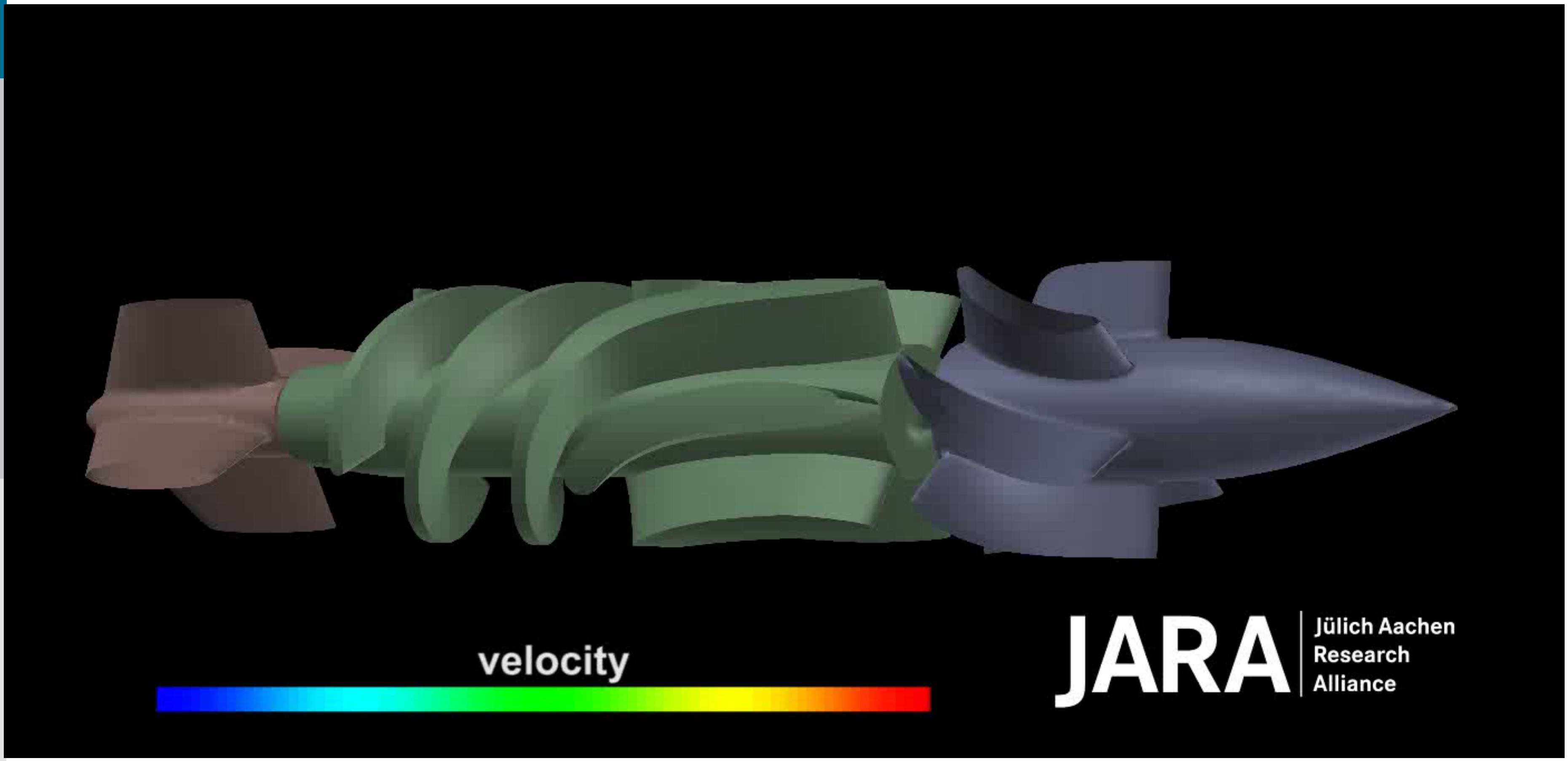


Long range Ewald scales with O(N log N) …
But MD dominates

Example 2: Engineering – Biomedical Flows

JÜLICH FORSCHUNGSZENTRUM

Simulation of Blood Flow in a Ventricular Assist Device
Marek Behr, RWTH Aachen



Code + Analysis tools → large Improvements

XNS CFD solver

- 3D space-time simulation of MicroMed DeBakey axial blood pump
- 4 million elements
- Partitioning by Metis graph patitioning package
- Incompressible Navier-Stokes Eq.
- FEM, GMRES, 3 time steps, 4 Newton-Raphson iterations

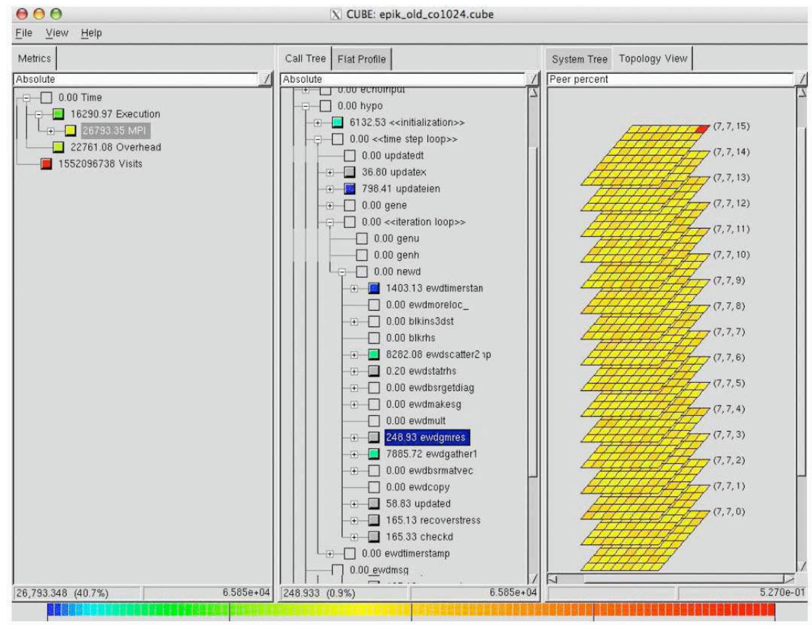
Analysis by SCALASCA package

(Bernd Mohr, Felixn Wolf)

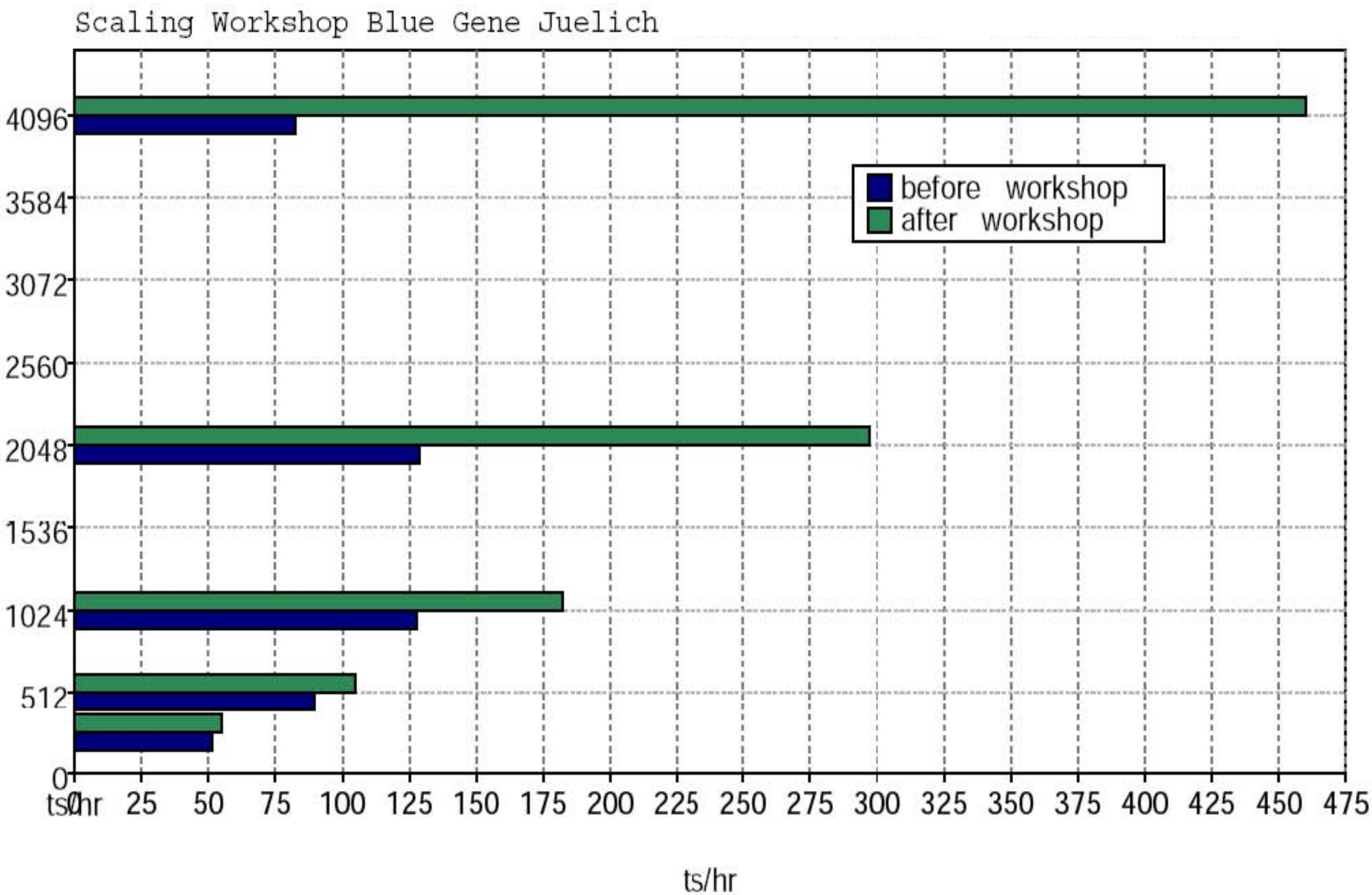
- Too many MPI_Sendrecv with zero-byte transfers
- Good speedup to 4096 processors

Still: strong load imbalance in GMRES

- process with highest rank overloaded



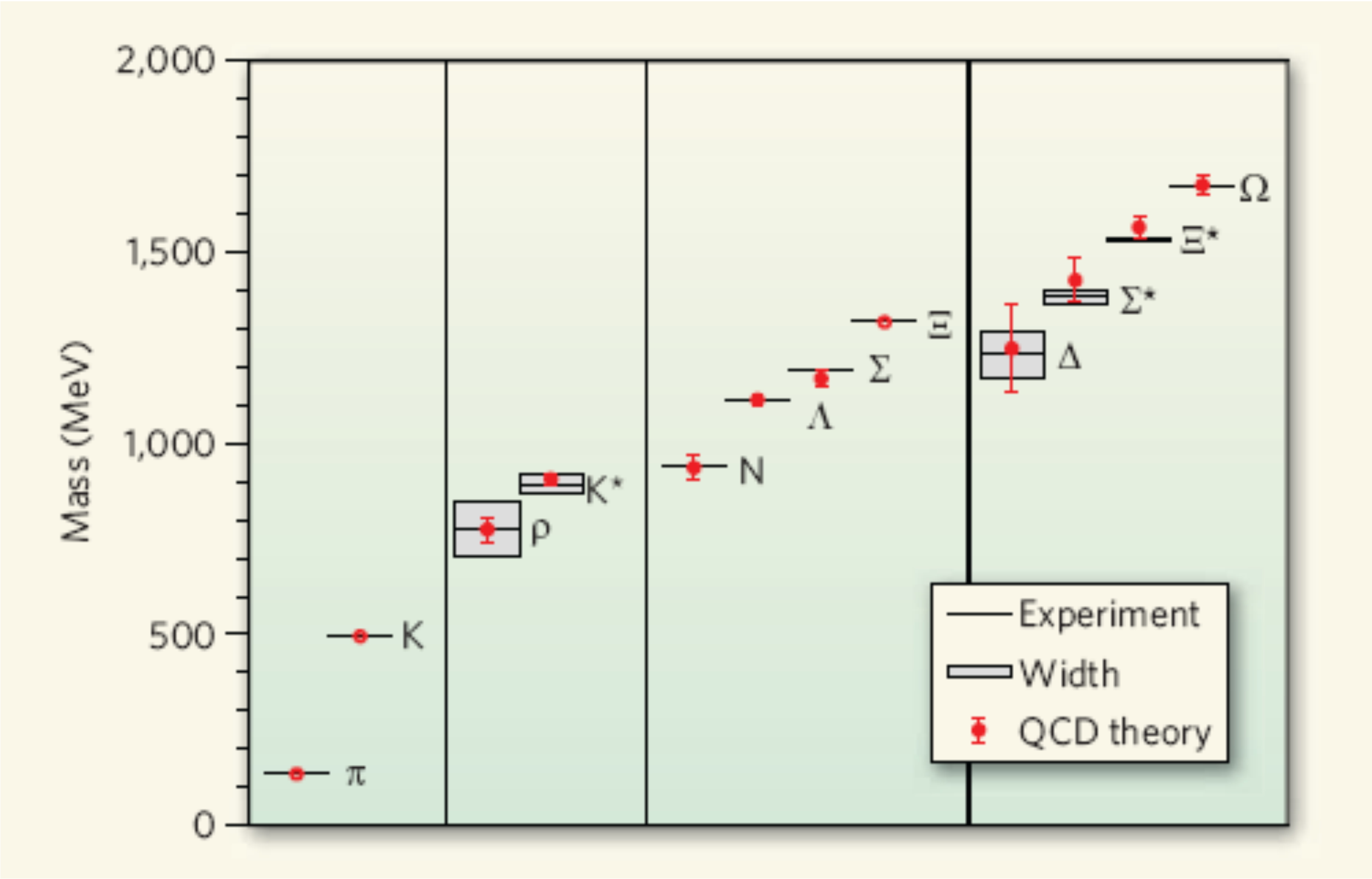
After Improvements

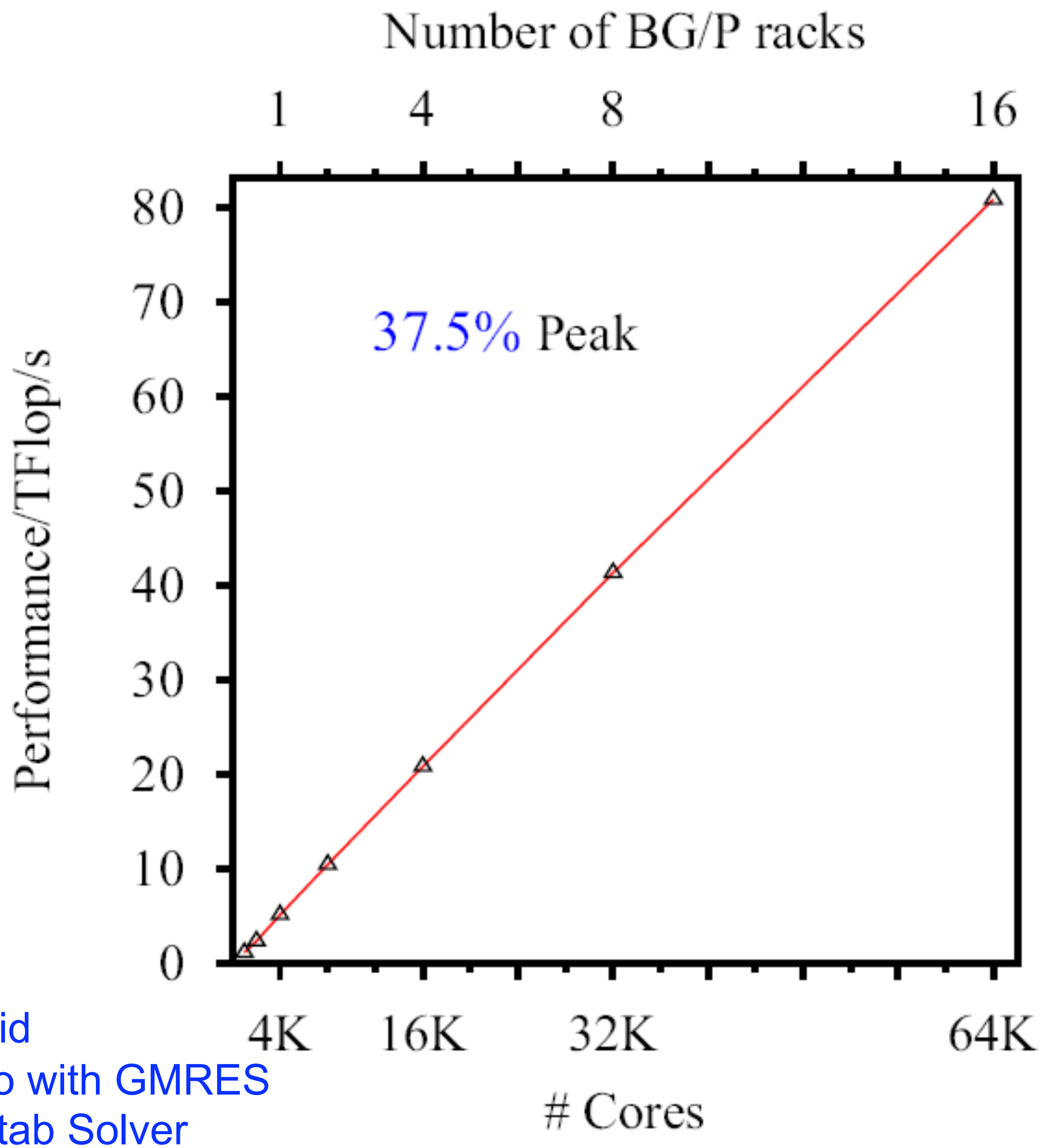


Example 3: Theoretical Particle Physics

Fodor et al. 2008: Validation of Quantum Chromodynamics

Among 10 SCIENCE-breakthroughs of 2008





Improvements through low-level programming

Code: Hybrid Monte Carlo with GMRES and BiCGStab Solver

JÜLICH FORSCHUNGSZENTRUM

HPC Software & Benchmark Codes

Disclaimer:

List of software is a selection and not comprehensive

HPC Software Challenges

- **Extreme scalability**
  - Exascale: number of cores beyond any reasonable, manageable limit
- **Extreme complexity**
  - Machine architecture gets more complicated instead of becoming simpler (KISS!)
- **Little to not existing fault-tolerance in existing base software**
  - e.g. MPI, OpenMP, schedulers, …
- **Rapid grows in system size /change in HW architecture**
  - SW developers cannot keep pace

EU HPC Software: Programming Models

MPI

- Open MPI European partners HLRS, INRIA, Univ. Jena, Univ. Chemnitz, TU-Dresden, BULL
- MPICH-V fault tolerance, MPI Madeleine (INRIA)
- HLRS, Bull, NEC, (ZIH, JSC) participating in MPI-3

OpenMP

- EU ARB members: EPCC, RWTH, (BSC?)
- BSC Mercurium compiler framework

Pragma-based task parallelism

- SuperScalar (BSC)
  - Subject in future EU proposals (EU ITEA2 H4H, FP7 FET EXACT)
- HPMM (INRIA/CAPS)

Parallel Object-oriented

- Kaapi (C++) + PROACTIVE (Java) (INRIA), PM2 (LaBRI, INRIA)

EU HPC Software: Numerical Applications

Numerical Middleware

- superLU (INRIA), MUMPS (ENSEEIHT)
- Scilab (Digiteo)

Benchmarks

- DEISA
  - 14 full applications
  - HPCC
- PRACE
  - 20 full applications
  - Various low-level
- EPCC micro benchmarks

Application Software Benchmarks: DEISA

Astrophysics: GADGET, RAMSES

CFD and combustion: Fenfloss

Earth sciences and climate research: ECHAM5, IFS, NEMO

Life sciences and informatics: NAMD, IQCS

Materials science: CPMD, QuantumESPRESSO

Plasma physics: GENE, PEPC

Quantum chromodynamics: BQCD, SU3_AHiggs

DEISA benchmark represents major EU HPC applications

Application Software Benchmarks: PRACE

(see White Paper by Peter Michielse)

JÜLICH FORSCHUNGSZENTRUM

Application	Application area
QCD	Particle physics
VASP	Computational chemistry, condensed matter physics
NAMD	Computational chemistry life sciences
CPMD	Computational chemistry, condensed matter physics
Code_Saturne	Computational fluid dynamics
GADGET	Astronomy and cosmology
TORB	Plasma physics
ECHAM5	Atmospheric modelling
NEMO	Ocean modelling

Application (to be considered)	Application area
AVBP	Computational fluid dynamics
CP2K	Computational chemistry, condensed matter physics
GROMACS	Computational chemistry
HELIUM	Computational physics
SMMP	Life sciences
TRIPOLI4	Computational engineering
PEPC	Plasma physics
RAMSES	Astronomy and cosmology
CACTUS	Astronomy and cosmology
NS3D	Computational fluid dynamics

Porting Codes

Application	MPP-BG	MPP-Cray	SMP-TN-x86	SMP-FN-pwr6	SMP-FN+Cell	SMP-TN+vector
QCD	Done	Done		Done		
VASP	Done			Done	Stopped	Yet to start
NAMD	Done	Done		Done	Yet to start	
CPMD	Done			Done	Done	Yet to start
Code_Saturne	Done	Done		Done	Stopped	Done
GADGET	Done		Done	Done		
TORB	Done			Done	Yet to start	
ECHAM5	Stopped	Done	In progress	Done		Yet to start
NEMO	Done	Done		Done		In progress
CP2K	Done	Done		Done		
GROMACS	Done	Done		Done		
NS3D		Yet to start	Done	Yet to start		Done
AVBP	Yet to start		Done	Done		
HELIUM	In progress	Done		Done		
TRIPOLI_4	Yet to start		Done			
PEPC	Done	Done		Done		
GPAW	Done	Done		Done		
ALYA					Done	
SIESTA					Done	
BSIT					Done	

Table 4: Summary on porting efforts for benchmark codes and prototype architectures.

EU HPC Software: Tools I

System / cluster tools

- Benchmarking: JuBe (JSC)
- Resource allocation: OAR (INRIA)
- System monitoring: LLview (JSC)
- Cluster middleware: ParaStation (ParaStation-Consortium: ParTec, JSC, Karlsruhe, Heidelberg, Wuppertal)

Grid Middleware

- UNICORE (UNICORE forum, JSC, …)
- GLite (CERN, LHC)
- dCache (DESY)
- DIET: grid RPC system (INRIA, CNRS, LIP/ENS Lyon, …)

EU HPC Software: Tools II

Programming tools

- Debugging: DDT (Allinea)
- MPI debugging: Marmot (ZIH –TU-Dresden / HLRS)

Performance

- OPT (Allinea)
- Paraver/Dimemas (BSC)
- KOJAK/Scalasca (JSC)
- Vampir (ZIH-TU-Dresden)
- Periscope (TU Munich)
- SlowSpotter/ThreadSpotter (Acumem)

European tool integration projects

- EU ITEA2 ParMA project (17 partners, FR, DE, ES, UK)
- German BMBF SILC

Existing Working Collaborations

- **MPI standardization and Open MPI project**
- **OpenMP standardization**
- **Global Grid Community**
- **Example: Performance tools community**
  - Voluntary US participation in EU APART WG (1998-2004)
  - Common Dagstuhl seminars (2002, 2005, 2007, 2010)
  - CScADS workshops (2007, 2008, 2009)
  - Collaborating collaboration projects
    - POINT (UO, ICL, NCSA, PSC)
    - VI-HPS (RWTH, ZIH, JSC, ICL)
  - New: DOE ASCR funding for non-U.S. partners!
    - PRIMA (UO, JSC): 2009-2012
    - “PTP++” (IBM, LANL, ORNL, JSC, Monarch): 2009-2012

Collaboration and Funding

Lessons learned

- Collaboration projects need
  - Strong leadership + Funding
  - Examples of failures: PTOOLS, OSPAT, ….
- Bottom-up, technology-driven, friendship approaches work much better than top-down, politically-driven, mandated ones
- Top-down provides funding
- Need combined approach: bottom-up meets top-down and long-term commitments of funding agencies

Proposal

- Local (US, EU, Asian) funding programs need to allow to fund additional global partners
- New global funding for networking (coordination, dissemination, synchronization efforts)

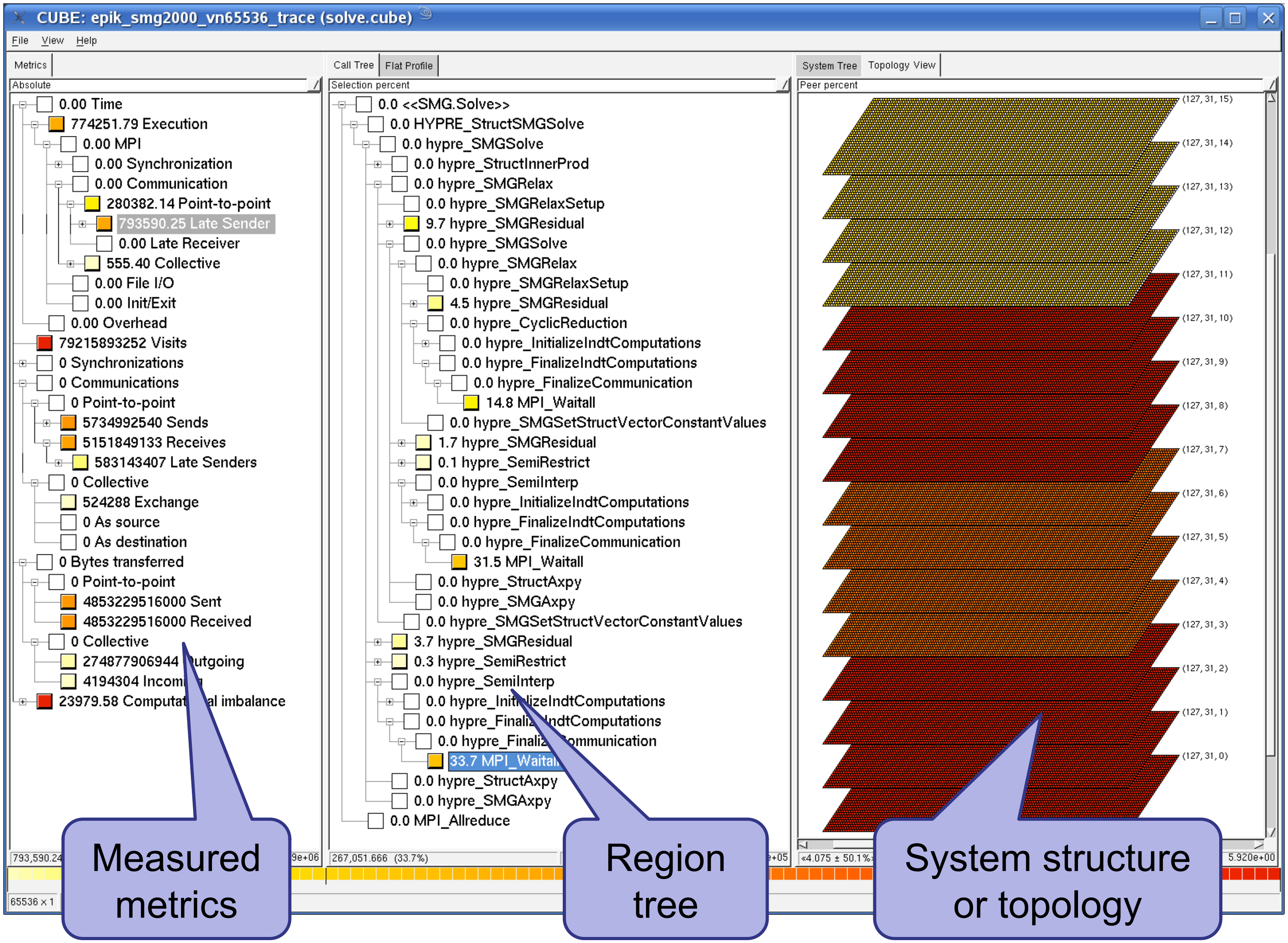
Example EU

scalasca

JÜLICH FORSCHUNGSZENTRUM

- **Scalable Analysis of Large Scale Applications**
- **Follow-up project to well-known KOJAK project**
- **Installed on many leadership class systems (EU, US)**
- **Successfully used on 65536 cores**
- **Integration with TAU and Vampir toolsets**
- **Approach**
  - **Instrument** C, C++, and Fortran parallel applications
    - Based on MPI, OpenMP, SHMEM, or hybrid
  - **Collect** event traces (or callpath profiles)
  - **Search** trace for event patterns representing inefficiencies **in parallel**
  - **Categorize and rank** inefficiencies found
- **http://www.scalasca.org/**

Trace analysis SMG2000@64k



Example EU

ParaStation Cluster Middleware

ParaStation V5:

- Multi-core aware cluster operating and management software
- Open source → GPL licensed
- ParaStation Consortium: ParTec, Forschungszentrum Jülich, Universities of Karlsruhe, Heidelberg, Wuppertal
- Deamon based
- MPI-2
- Grid Monitor (full awareness of complete cluster status)
- IB, Ethernet, Myrinet, just everything

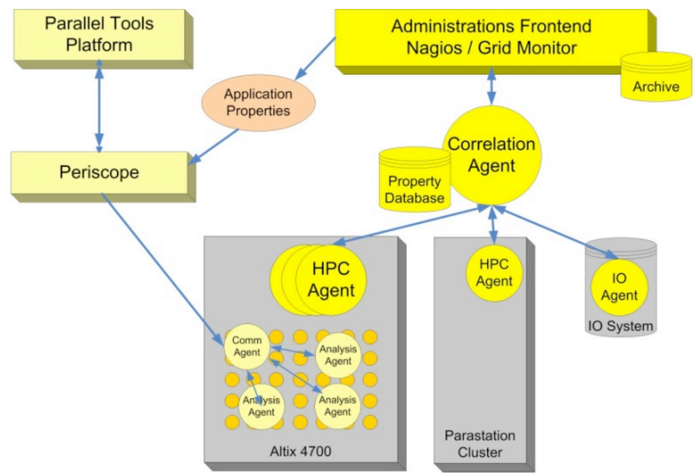
ParaStation Research
(Projects funded by Federal Ministry of Education & Research)

D-Grid 2, (2007-2010)

German Grid initiative
(funded by BMBF)

ISAR project (2008 – 2011)
Integrated system and application analysis for massive parallel computer

Members:
Uni Munich, Leibniz Compute Center (LRZ),
Compute Center Garching (Max-Planck), ParTec, IBM



ee-Clust project (2008 – 2011)
Energy efficient cluster computing

Members:
Uni Heidelberg, TU Dresden,
Research Centre Julich, ParTec

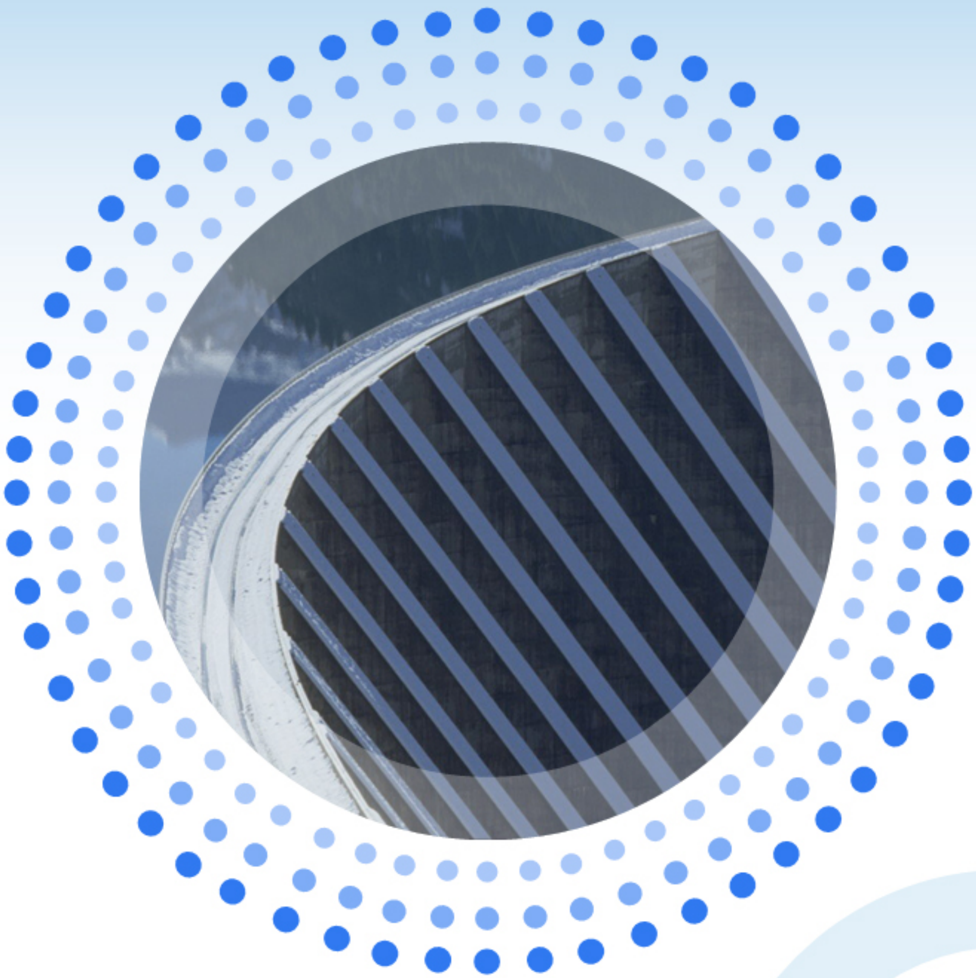
- Goals
- Scalable cluster OS
- Fighting OS-jitter

JÜLICH FORSCHUNGSZENTRUM

Plans for Exascale Activities and Initiatives in Europe

1. EESI (International HPC Software Coordination and Development)
2. COSI-HPC Proposal (HPC-Software - Coordination)
3. Lincei Initative (Comp. Science)
4. CECAM (Comp. Science)
5. PRACE (ESFRI-Infrastructure)

1.



Establishing the

European Exascale Software Initiative

Contribution by

Jean-Yves Berthou, EDF R&D

Context: International Exascale Software Project

SC'08 (nov. 2008) : DOE/NSF/DOD launched the International Exascale Software Project (IESP)

Plan to build an international partnership that joins together ***industry, the HPC community (CS and Apps), and production HPC facilities*** *in a collective effort to design, coordinate, and integrate software for leadership-class machines.*

Specifically, engagement in the following activities should be started:

- *Build international collaborations in the areas of high-performance computing software and applications.*
- *Development of open source systems software, I/O, data management, visualization, and libraries of all forms targeting tera/peta/exascale computing platforms,*
- *Research and development of new programming models and tools addressing extreme scale, multicore, heterogeneity and performance,*
- *Cooperation in large-scale systems deployments for attacking global challenges,*
- *Joint programs in education and training for the next generation of computational scientists.*
- *Vendor engagement to coordinate on how to deal with anticipated scale."*

European Exascale Software Initiative (EESI)

Main goals

- Building and promoting European position inside the IESP initiative
- Identifying Grand Challenge applications, from academia and industry, with a strong economical, societal and/or environmental impact that will benefit of Petaflop capacities in 2010 and Exaflops in 2020
- Identify critical software issues for Peta-ExaScale systems
- Building a European/US/Japan program in education and training for the next generation of computational scientists
- Output : Proposition of a strategic research action agenda for Peta-Exascale Software and Grand Challenge applications at the European level coordinated with US and Japanese agendas

European Exascale Software Initiative (EESI)
Preparatory phase Project Proposal – 12 months

Establish a European position inside the IESP initiative

- Promote and represent the European position
- Influence on decisions and actions
- Synchronize European agenda with other international agenda

Contribute to the International dialogs between US and Europe and Japan and Europe and be a bridge between some EU organizations including the European commission and IESP

Identify main HPC European actors both at end users level and at academic level

Define and implement the organization and governance rules of EESI

Identify main European HPC existing or planned projects

Built a first European and international vision of the on-coming HPC challenges and work to achieve

European Exascale Software Initiative (EESI)
Preparatory phase Project Proposal – 12 months

Submitted to ICT 2009.9.1 International cooperation a) Support to Information Society policy dialogues and strengthening of international cooperation

Partners	Country	Contact		Title
EDF (leader)	France	Jean-Yves	Berthou	EDF R&D Information Technology Program Manager
		Jean-François	Hamelin	EDF R&D Information System Director
GENCI, FR	France	Catherine	Rivière	Chairman and CEO of GENCI
		Virgine	Mahdi	
INRIA	France	Frank	Cappello	Director of the joint INRIA/NCSA laboratory
EPSRC	UK	Jane	Nicholson	High End Computing & E-Science Program Manager
Forschungszentrum Jülich GmbH	Germany	Thomas	Lippert	Director of Institute for Advanced Simulation, Head of Jülich Supercomputing Centre
		Bernard	Mohr	
BSC	Spain	Mateo	Valero	Director of BSC
		Sergi	Girona	Operations Director BSC
NCF	Netherlands	Patrick	Aerts	Director of NCF
		Peter	Michielse	Deputy Director of NCF
Arrtic	France	Thierry	Bidot	

European Exascale Software Initiative (EESI)
Preparatory phase Project Proposal – 12 months

Supporting partners

US

IESP, Executive Director, J. Dongarra
U. Urbana-Champaign, Deputy Director for Research, B. Gropp
U. Urbana-Champaign, Professor, M. Snir

Japan

Tokyo Institute of Technology, Professor & Director Research Infrastructures Division GSIC, Satoshi Matsuoka

Europe

PRACE, Current Chairman of the Initiative Management Board, Jane Nicholson
European Science Fondation, the Physics and Engineering Sciences Unit, Science Officer, Dr Thibaut Lery
European Network for Earth System modelling, Chairman of the Scientific Board, S. Joussaume
TERATEC, Chairman, C. Saguez
ORAP, Chairman of the Scientific Council, JC André
Daresbury Lab., Acting Director CS & E dpt., R. Blake
CERFACS, Director, JC André

Industry/Editor

TOTAL, Scientific Director, JF Minster
SNECMA, Vice President Engineering & Technology, P. Thouraud
NAG, Chief Tech Officer/Vice President HPC Business, M. Dewar/A. Jones

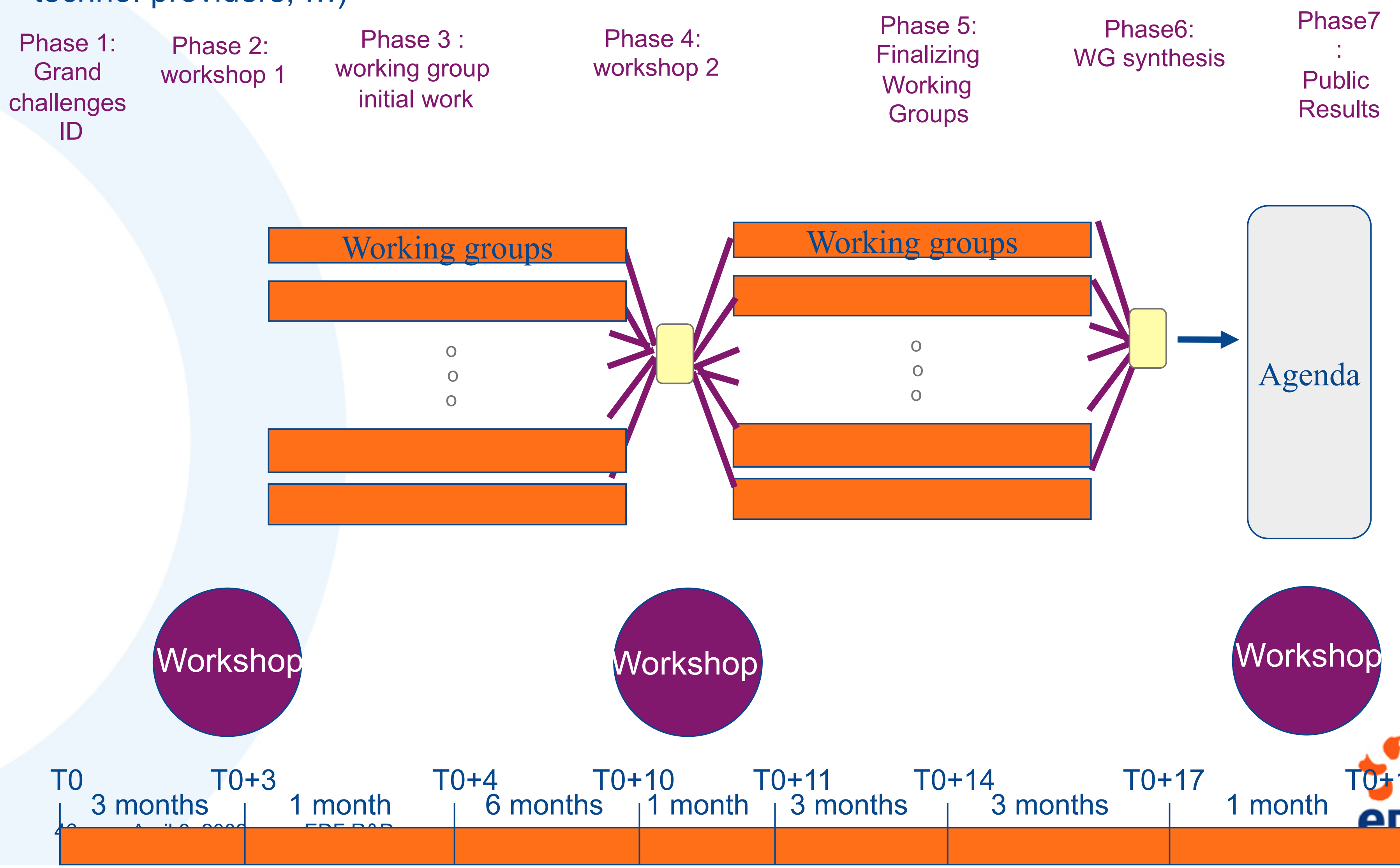
European Exascale Software Initiative (EESI)
Implementation phase (draft)

Building a research agenda and directions for future

- Identifying Grand Challenge applications, from academia and industry, with a strong economical, societal and/or environmental impact that will benefit of Petaflop capacities in 2010 and Exaflop around 2020
- Identify critical software issues for Peta-ExaScale systems
- Building a EU/US/Japan program in education and training for the next generation of computational scientists
- Proposition of a strategic research action agenda for Peta-Exascale Software and Grand Challenge applications at the European level coordinated with US and Japan agendas

European Exascale Software Initiative (EESI)
Implementation phase – 18 months (draft working program)

Input from EESI **Preparatory Phase** : identification of keyplayers (End user communities, techno. providers, …)

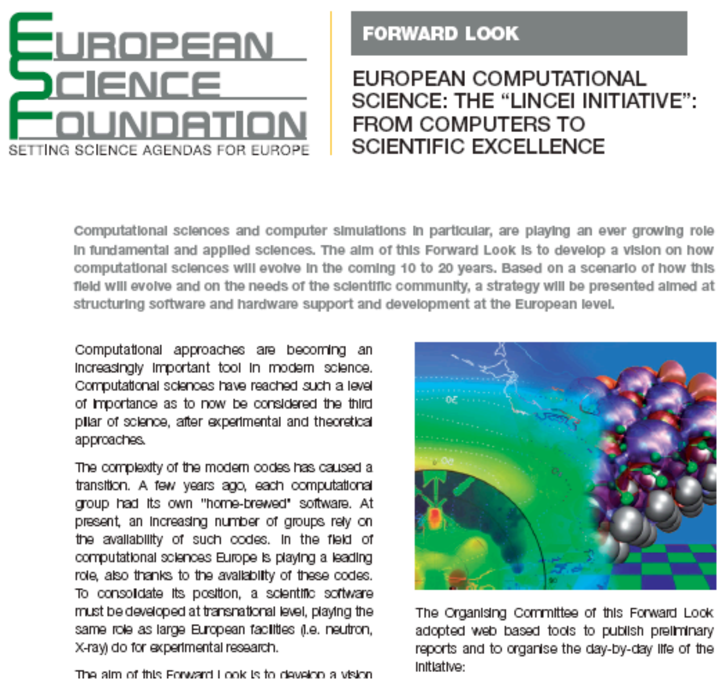


2. COSI-HPC
(proposal, lead by CSC-Finland)

- The Coordination for Software Initiatives in HPC (COSI-HPC) project is designed to promote key elements in an innovation and service ecosystem around the future European Petascale computing research infrastructure (RI).
- Set of actions aimed at coordinating activities in the area of software engineering and software services for large-scale computing, targeting the planned European Petascale facilities as well as future Exascale systems.
- Coordination of existing and future research and industry initiatives such as PRACE, DEISA, PROSPECT, and STRATOS
    - Analysis of HPC software activities in Europe
    - Building up a software community for HPC
    - Address future software challenges

3. Linceï Initiative

Contribution by Michel Marechal



EUROPEAN SCIENCE FOUNDATION
SETTING SCIENCE AGENDAS FOR EUROPE

FORWARD LOOK

EUROPEAN COMPUTATIONAL SCIENCE: THE "LINCEI INITIATIVE": FROM COMPUTERS TO SCIENTIFIC EXCELLENCE

Computational sciences and computer simulations in particular, are playing an ever growing role in fundamental and applied sciences. The aim of this Forward Look is to develop a vision on how computational sciences will evolve in the coming 10 to 20 years. Based on a scenario of how this field will evolve and on the needs of the scientific community, a strategy will be presented aimed at structuring software and hardware support and development at the European level.

Computational approaches are becoming an increasingly important tool in modern science. Computational sciences have reached such a level of importance as to now be considered the third pillar of science, after experimental and theoretical approaches.

The complexity of the modern codes has caused a transition. A few years ago, each computational group had its own "home-brewed" software. At present, an increasing number of groups rely on the availability of such codes. In the field of computational sciences Europe is playing a leading role, also thanks to the availability of these codes. To consolidate its position, a scientific software must be developed at transnational level, playing the same role as large European facilities (i.e. neutron, X-ray) do for experimental research.

The aim of this Forward Look is to develop a vision

The Organising Committee of this Forward Look adopted web based tools to publish preliminary reports and to organise the day-by-day life of the initiative:

ESF: European Science Foundation does coordinate National Research funding organizations in Europe

80 members in 30 countries
http://www.esf.org

A **Forward Look** has been set up by ESF Panel of 12 high level computational scientists has produced a report

http://ccp2007.ulb.ac.be/FL-Lincei.pdf

Recommendations (I)

- National science funding agencies in Europe undertake a coordinated and sustained effort in scientific software development, including documentation, updating, maintenance and dissemination.
- This necessarily implies the means for training and cooperation.
- **Restructure and federate**, within an European-scale infrastructure, existing and expanded activities on scientific software and other forms of cooperation and dissemination in Europe through **European Computational Collaborations specific to each scientific area**.
- This would be guided by active research scientists and deliver the infrastructural services to the working scientists.

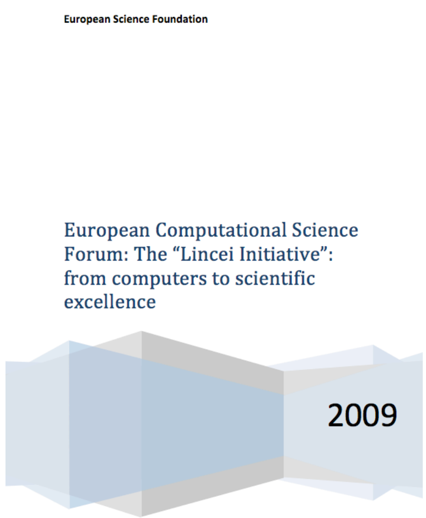
One such example:

- CECAM upgrade
- (multinode, multi-disciplines

CECAM is organizing code developpers in condensed matter

http://www.cecam.org/

- **Recommendations (II)**
- To achieve those goals, it is proposed to set up a **Computational Sciences Expert Committee (CSEC) attached to ESF which would speak for the whole community of computational sciences**.
- Its purpose would be to start setting up a durable plan for European cooperation in each of the fields of science using computers.
- It would address the policy issues involved, and work with national and European organisations to optimize the development of scientific computing in Europe.



ESF is now considering establishing CSEC

4.

cecam — Centre Européen de Calcul Atomique et Moléculaire

EPFL — ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Scientific software development: a new CECAM initiative

On March 30-31, 09, the director (Wanda Andreoni) and vice-president (Paul Durham) of CECAM convened a meeting at CECAM Headquarters in Lausanne of a group of scientists with the aim of reflecting upon the possible role CECAM could play in enhancing European scientific software development and support

- Alessandro Curioni *(IBM Research Zurich)*
- Stefano de Gironcoli *(SISSA, Trieste)*
- Mauro Ferrario *(University of Modena)*
- Xavier Gonze *(University of Louvain)*
- Christian Holm *(University of Stuttgart)*
- Wim Klopper *(University of Karlsruhe)*
- Mike Payne *(University of Cambridge)*
- Bill Smith *(Daresbury Laboratory)*
- Godehard Sutmann *(Research Centre Jülich)*
- Doros Theodorou *(University of Athens)*
- Other scientists will be invited to join the group.

CECAM (Centre Europeen de Calcul Atomique et Moleculaire) is a European organization devoted to the promotion of fundamental research on advanced computational methods and to their application to important problems in frontier areas of science and technology

Towards the High-End HPC Service for European Science

Thomas Lippert, PRACE Project Coordination@FZ-Jülich

CAPACITIES

e-infrastructure

Computational science infrastructure in Europe

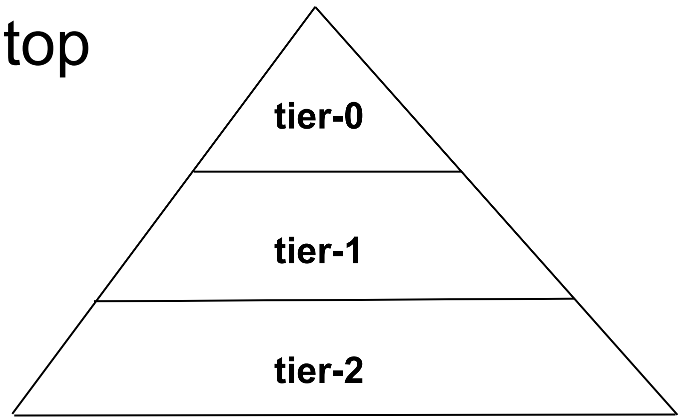
The European Roadmap for Research Infrastructures is the first comprehensive definition at the European level

Research Infrastructures are one of the crucial pillars of the European Research Area

A European HPC service:
- **Horizontal**
- **attractive for research communities**
- **supporting industrial development**

ESFRI Vision for a European HPC service

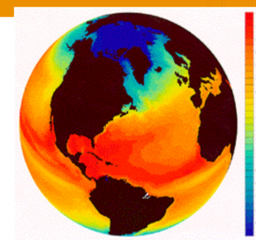
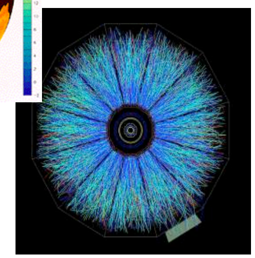
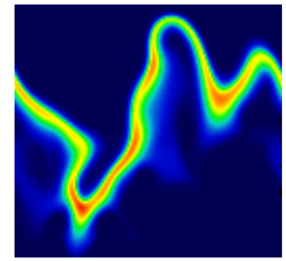
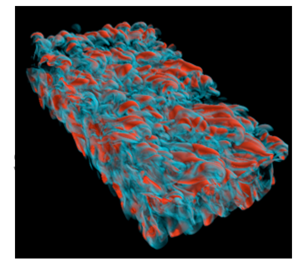
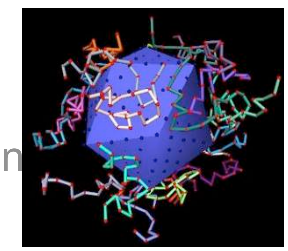
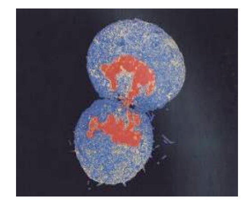
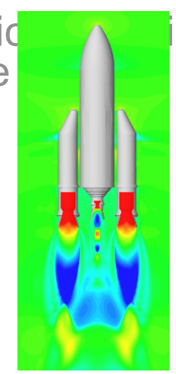
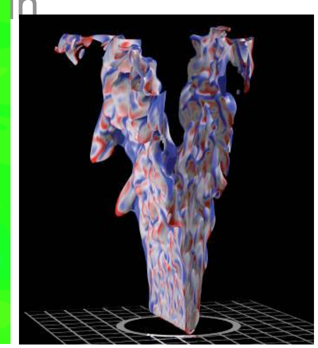
- Need European HPC-facilities at top of an HPC provisioning pyramid
  - Tier-0: 3-5 European Centres
  - Tier-1: National Centres
  - Tier-2: Regional/University Centres



- Part of the Creation of a European HPC ecosystem
  - HPC service providers on all tiers
  - Grid Infrastructures
  - Scientific and industrial communities
  - The European HPC industry

- **Renewal every 2-3 years**
  Construction cost 200 – 400 Mio. €
  Annual running cost 100 – 200 Mio.€

HET: The Scientific Case

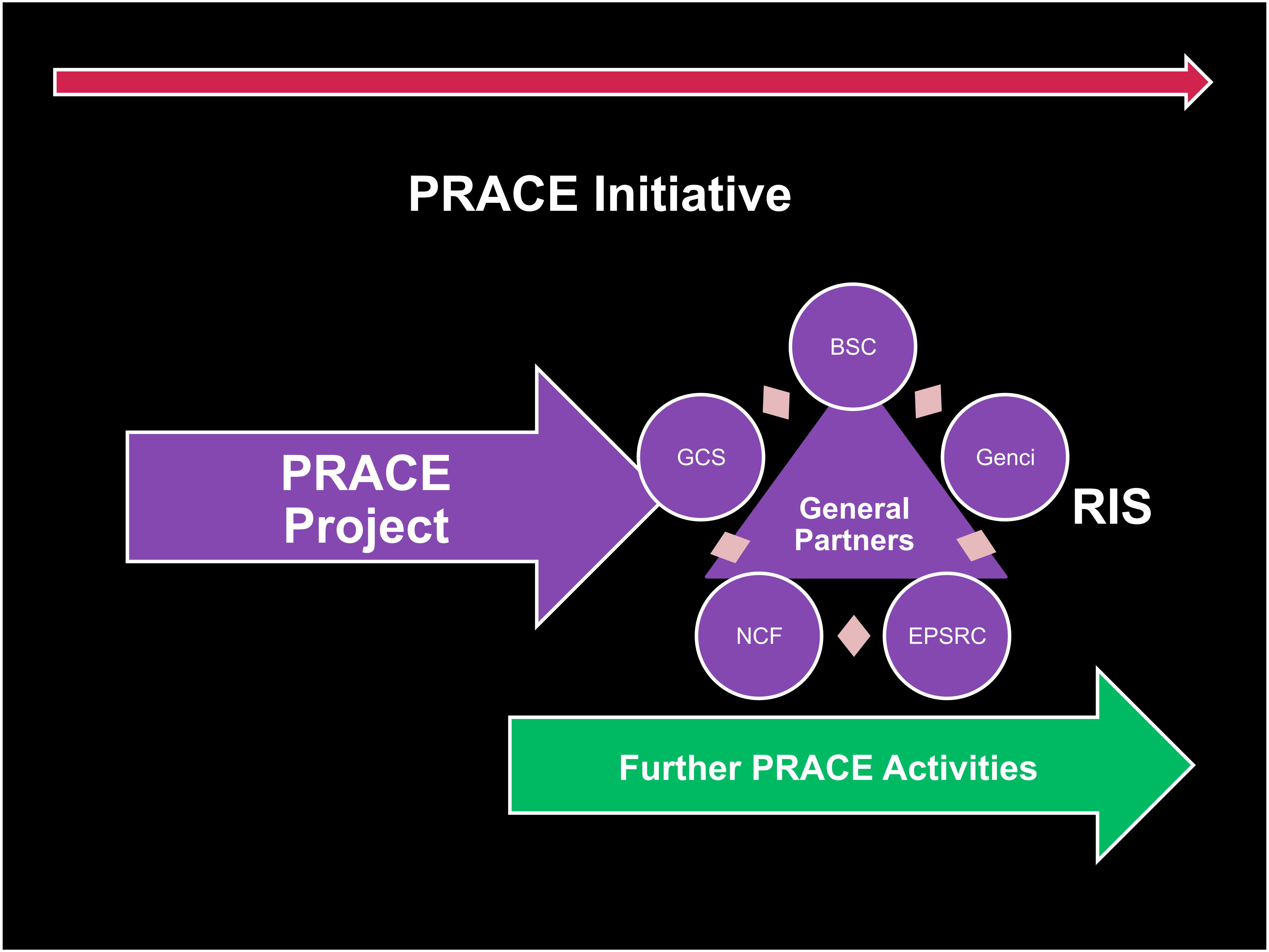
- Weather, Climatology, Earth Science
  - degree of warming, scenarios for our future climate.
  - understand and predict ocean properties and variations
  - weather and flood events
- Astrophysics, Elementary particle physics, Plasma physics
  - systems, structures which span a large range of different length and time
  - quantum field theories like QCD →LHC, FAIR
  - ITER
- Material Science, Chemistry, Nanoscience
  - understanding complex materials, complex chemistry, nan
  - the determination of electronic and transport properties
- Life Science
  - system biology, chromatin dynamics, large scale protein dynamic in association and aggregation, supramolecular systems, medicine
- Engineering
  - complex helicopter simulation, biomedical flows, gas turbines and internal combustion engines, forest fires, green aircraft

PRACE – Initiative

New Partners - since May 2008

First Industry Seminar attendees

TOTAL
PORSCHE
ArcelorMittal
cea
Alcatel·Lucent
CGGVERITAS
REPSOL
EADS
eDF
BAE SYSTEMS
GM
allinea
SCALE TO NEW HEIGHTS
SOCIETE GENERALE
CS
FUGRO
CERFACS
CAPS
MEDIT
EDS
HUTCHINSON
WORLDWIDE
NOVARTIS
BNP PARIBAS
Rabobank
ACUMEN solutions
Renater
PHILIPS
IBERDROLA
Schering-Plough
T-Systems
AkzoNobel
Tomorrow's Answers Today
AIRBUS
SCANIA
nag
Eni
Snecma
Groupe SAFRAN
NUMTECH
L'ATMODÉLISATION
mitrionics
NXP
founded by Philips


PRACE Initiative
PRACE Project
BSC
GCS
Genci
General Partners
RIS
NCF
EPSRC
Further PRACE Activities

PRACE Project

- PRACE is horizontal ESFRI project
  - Mission to serve the scientific communities at large
  - Need to cooperate with communities
- **Software for the Multi-Petaflop/s age**
  - Only few of today's applications are scalable to hundred-thousand CPU-cores
  - PRACE seeks to gain knowledge in Petascaling to educate and support its future users
  - **An additional European effort is needed – international cooperation should be sought for Exascale challenges**
- Exascale data services for scientific communities
  - Support efforts to agree on community standards for storing, annotating and retrieving their data, provide reliable data services

PRACE Project

- Prepare the contracts to establish the PRACE permanent Research Infrastructure as a single Legal Entity in 2010 including governance, funding, procurement, and usage strategies.

- Perform the technical work to prepare operation of the Tier-0 systems in 2009/2010 including deployment and benchmarking of prototypes for Petaflop/s systems and porting, optimising, Peta-scaling of applications

WP6	Software enabling for Petaflop/s systems (RTD)	**Prepare key applications to use the future Petaflop/s systems efficiently; capture requirements for WP7 and WP8 and create a benchmark suite.**	EPSRC
WP7	Petaflop/s Systems for 2009/2010 (RTD)	Identify potential Petaflop/s systems for PACE that can be installed in 2009/10 with prototypes deployed by WP5. Prepare the procurement process including acceptance criteria.	GENCI
WP8	Future Petaflop/s computer technologies beyond 2010 (RTD)	**Start a permanent process to identify technologies for future multi-Petaflop/s systems of the RI and work with hardware and software vendors to influence the direction they are taking. Establish PRACE as a leader in HPC technology.**	Gauss Centre

WP6: Software Enabling for Petaflop/s Systems

- Create an application benchmark suite
- Capture application requirements for Petascale systems
- Port, optimise and scale selected applications
- Evaluate application development environments of the prototypes

PRACE WP8: STRATOS

- STRATOS is a deliverable of PRACE WP8: Create sustained platform for technology watch and development for PRACE
- Hardware
  - Identifying and developing components of future multi-Petaflop/s hardware
- Software
  - Plans for Exascale software development within STRATOS

Areas of Contribution to IESP

European Science and Engineering Communities

- Coordination with science drivers
- Identify application codes and enabling HPC software

Performance Tools

Programming Tools

Benchmark Codes

MPI, OpenMPI standardization

Scalable Cluster OS

Grid/Cloud-Integration-Middleware

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara
Jean-Yves Berthou
Mitsuhisa Sato
Peggy Williams
Vivek Sarkar
Ann Trefethen

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara

Jean-Yves Berthou

Mitsuhisa Sato

Peggy Williams

Vivek Sarkar

Ann Trefethen

Evolutionary Software Areas for Exascale:

• Extend current program models through single node threading of messaging. Eliminate “per task” scaling terms in messaging layer to allow for higher “flat scaling”.

• Allow for mixed programming models to coexist. We need a bridge to new programming models that is not an all or nothing proposition.

• Enhance job flow to enable many concurrent capability scale jobs. (similar to the emerging approach at LLNL) This is likely to be a common early usage model for Exascale.

• Open source can be a very good thing for vendors and end users but we need to find a way share the responsibility and …… risk.

• Educate young people in parallel programming.

Revolutionary Software Areas for Exascale:

- Storage class memory is coming: Technology will offer 1000x less latency but there are many other dimensions to this. Need to think through the possible directions to use this technology.

- Systems are transitioning to being power optimized. Application developers are still focused on performance optimization regardless of power. In a world where there is a power budget, software should play a role in optimizing performance through optimization of Perf/ Watt. (with total power being a hard facility constraint)

- Reliability: This is not an issue of what will we do when systems can not be made reliable. This issue is making the best trade-offs between hardware, system software and fault tolerant applications.

Actions we can take

• Value of storage class memory: Need to have the HPC community united in articulating the value proposition associated with storage class memory. The critical break points in terms of bandwidth, density, cost and latency need to be understood to help guide the technology development.

• Power : This is somewhat a mindset change. Applications will eventually need to think of their computing resource as a total energy budget and they need to optimize within this. Fortunately much of performance tuning also drives toward energy efficiency… but not always. Tools and reports that detail the energy usage need to be accessible to users.

• Reliability: The realistic adoption, cost and risk of fault tolerant algorithms must be assessed and these should be traded off against hardware cost and risk. The systems can not move in a direction that “might” be acceptable from a reliability perspective. This makes a software solution very difficult.

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara

Jean-Yves Berthou

Mitsuhisa Sato

Peggy Williams

Vivek Sarkar

Ann Trefethen

Three important software areas where *evolution* is required to improve existing open source software for extreme scale

1.1 compilers/performance analysis tools for achieving mono-processor high performance, specially with accelerators (Larrabe, GPU, Cell, …)
Goal : more than 30% of the peak performance

1.2 Efficient, "easy to use", portable and fault tolerant implementation of Libraries, Languages/compilers for mixed parallelism : MPI/OpenMP/"cuda/Open CL like" languages
Goal: one million cores (heterogeneous, hierarchical and massively parallel)

1.3a Algorithm/solvers and data structures adapted to heterogeneous/hybrid, multilevel and hierarchical massively parallel machines.
Example: Dealing with non-structured irregular meshes for CFD computation on GPU
Goal:

- ⇒ No global communication involving the complete system(avoiding MPI_ALL-REDUCE, MPI_BARRIER,… on 1 million threads)
- ⇒ exhibiting different kind of parallelism (MPP, SIMD, …)
- ⇒ enabling fault tolerance techniques implementation
- ⇒ enabling efficient IO (data restructuring?)

1.3b Open Source scientific libraries sharing a single generic interface, targeting one million cores (heterogeneous & hierarchical)
Target: PETSc, SuperLU, ScaLAPACK, HyPre, MUMPS, PaStiX, …

Three important software areas where *revolution* is required to achieve scaling

2.1 Parallel visualization and remote/collaborative post-treatment tools

2.2 Parallel meshing, automatic hexahedral meshing, mesh healing, CAD healing for meshing and dynamic mesh refinement, hierarchical meshes (AMR like)
=> Dealing with $x10^{10}$ cells mesh before 2015 ($x10^{12}$ in 2020?)

2.3 Unified multiphysic/multiscale Simulation Framework and associated services, adapted to massively computing
⇒ mutualizing within a single platform pre and post-processing, calculation distribution and supervision, code coupling tools etc.
⇒ standardize integration of multiple solvers ("standard" for interoperability of scientific software components)
⇒ standardize data exchange (common data model for mesh and fields) and associated services (mesh projection, data interpolation, ..)

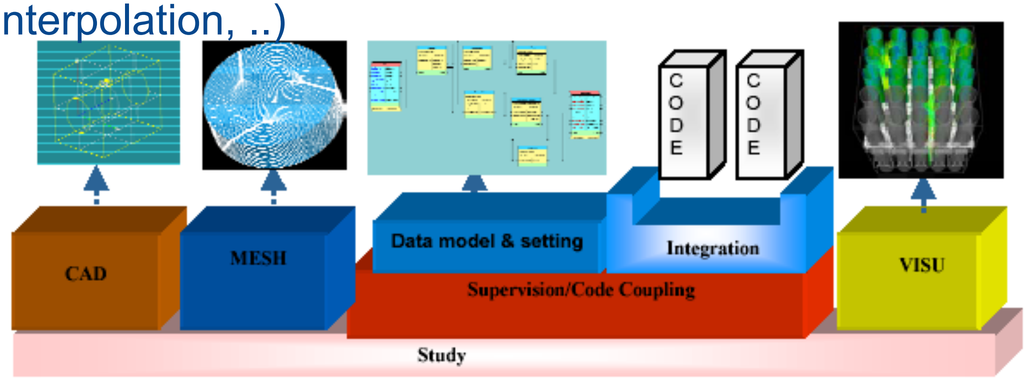


Figure 10. The Salome platform, www.platform-salome.org

How do we get it done to develop community-supported open source software to address these 6 areas, what is needed?

Some issues related to Open Source

Developing Open Source software, some conditions that may help to succeed and to keep going:

- one active leader and the recognition by key players
- A roadmap and a validated business model (at least for the leader)
- An ecosystem of partners for the software development, diffusion and associated services (installation, deployment, maintenance, specific developments)

Using Open Source software, some issues to be aware of: how they are supported, deployed, visibility of the roadmap, associated risks (as an example, moving from Qt3 to Qt4 cost 400 days of development to the SALOME project).

Suggestion:

- Identification of existing HPC Open Source software (cf.P. Beckman list)
- Promotion of an international HPC source forge for Open Source software diffusion?

How do we get it done to develop community-supported open source software to address these 6 areas, what is needed?

Need for International Task Forces on:

1. Parallel visualization tool. The community should focus on a small number of tools. VISIT and Paraview seems good candidates
2. Remote and collaborative post-treatment tools
3. Meshing tools. Need for an international joint effort between academic, commercial companies and end users
4. Common data model and associated libraries.Providing an international standard model for mesh and fields exchange and services(localization, projection, interpolation, arithmetic operations, …)
5. Supervising and code coupling tool. Unifying the software developments often driven by end users communities (climate, energy, …)
6. Uncertainties Quantification. Uncertainty analysis framework, Uncertainties referential (methodologies and tools, Open Turns)
7. Algorithm/solvers and data structures, solver interface
8. Fault tolerance. Need a joint effort involving OS, compilers, middleware/libraries, numerical solvers/algorithm researchers and engineer communities

Research policy:

- Identifying existing projects and research actions, roadmaps, cost
  - ⇒ Need for a consolidated international roadmap for HPC software
- Identifying grand challenge applications as driving forces
  - ⇒ Need for an International End User Forum structured around large communities (Climate, Health, Energy, Transport, Defense, …)

Identifying funding schemes: US/EU(or national)/Japan co-funding, single country funding with third parties participation?

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara

Jean-Yves Berthou

Mitsuhisa Sato

Peggy Williams

Vivek Sarkar

Ann Trefethen

3 important software areas where *evolution* is required

- To improve existing open source software for extreme scale
- I would propose "standard" development effort for making a steps for next evolution to exa-scale

- 3 areas
  - Programming language/interface for distributed memory
    - We should make "standard" for state-of-the-art programming languages
    - PGAS and remote memory interface
    - Global views such as Chapel and HPF
  - Fault tolerant model and APIs
    - Problems are in reality more than 10,000 cores (100TF)
    - Application people want some "standard" solutions in reality
    - e.g. MPI 3 effort is going on …
  - File I/O model for large scale systems
    - Data becomes more and more important.
    - We need "standard" model for I/O and file systems in hundreds thousands nodes
    - e.g. MPI IO, Grid distributed file system …

3 important software areas where *revolution* is required

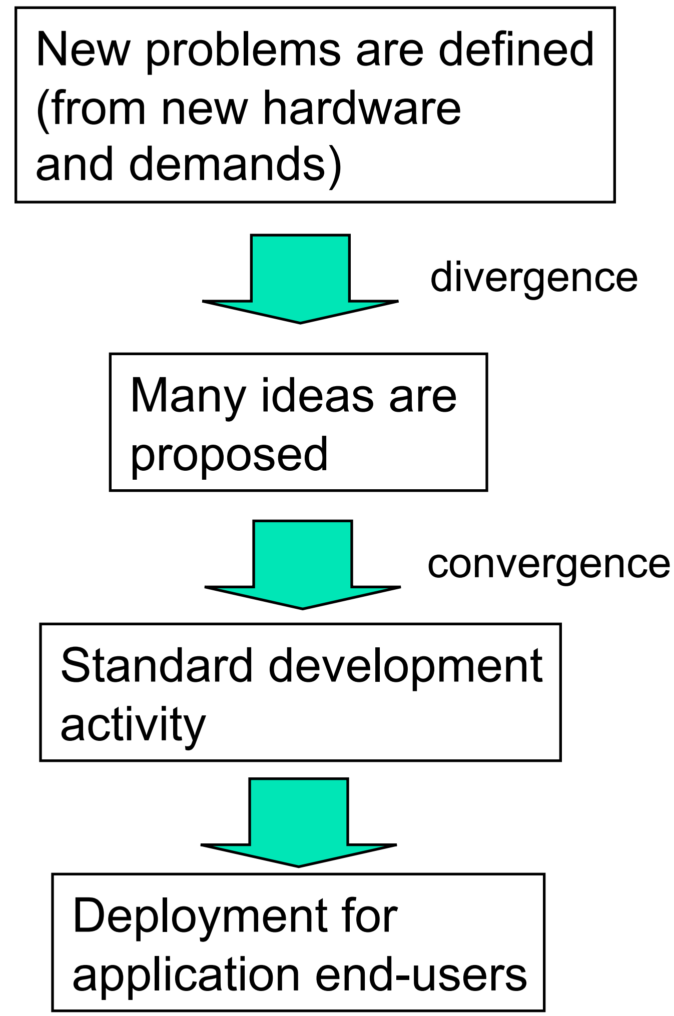
- To achieve scaling … (for exa-scale)
- I would propose software supports for platforms from "weak-scaling" to "strong scaling"
  - Exascale machine != embarrassingly parallel machine!
  - Complexity from arithmetic unit, cores, SMP nodes, network to systems.
- 3 area
  - Unified programming model for a high performance node such as multicore, many cores, accelerator (GPGPU, FPGA, …)
    - Data localities, scheduling, …
  - Programming model for compos-able and scalable software
    - Module programming in parallel software
    - High level programming lang. such as functional prog., dataflow prog., tele-scope lang.
    - For multi-physics simulations, …
  - Fault tolerant / dependability in exa-scale systems
    - Model, Cost, Programming, Algorithms, …
    - FT will still be important and hard problems.

How do we get it done?

- We should promote standard development effort of APIs between several levels and components of existing software (for "evolution")
  - For end-users, education, …
  - For development of higher-level software technologies
  - Improvement of technologies by defining clear APIs
  - e.g. MPI, OpenMP, …

- We should encourage the exchange ideas (for "revolution")
  - Diversity is important

A software research/development process model



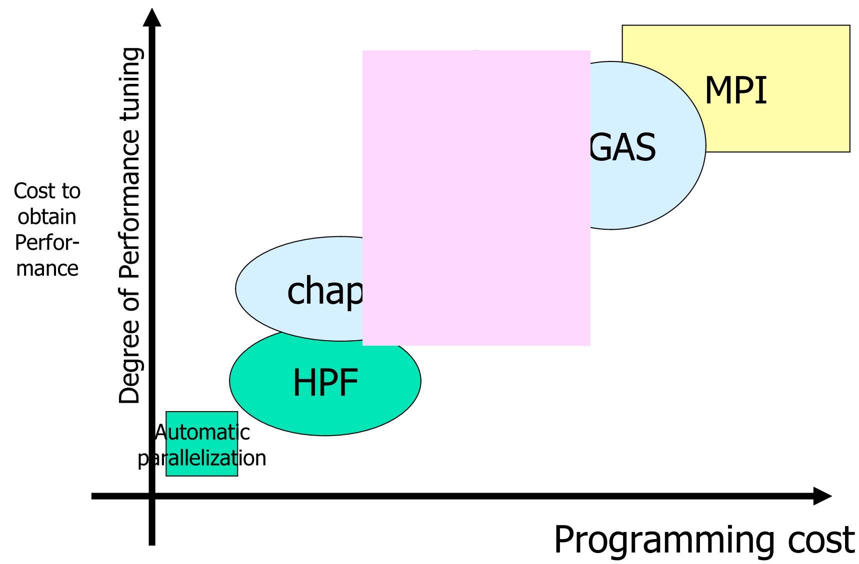
Proposal for "Parallel Programming Languages" area

- Many parallel programming languages have been proposed, but …
  - Many people still use MPI …
- OpenMP is now "standard" for programming multi-cores
- What about distributed memory programming?

- NOTE: Restrict us parallel extension of existing languages (C/F95) for end-users.
  - NOT HPCS languages and Java-based.

- How about PGAS (UPC and CAF)?
  - Local view parallel programming
  - Already standard?

- Global view parallel programming
  - We should learn from HPF history
  - Locality, efficient communication …
  - Any way to Combine to local view programming?



XcalableMP

http://www.xcalablemp.org

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara

Jean-Yves Berthou

Mitsuhisa Sato

Peggy Williams

Vivek Sarkar

Ann Trefethen

Where is evolution required?

- MPI
  - It's portable
  - It's ubiquitous
  - It isn't going away

- Thread Packages
  - Stable, portable, general user-level
  - Enable easier implementation of OpenMP
  - Allow oversubscription of HW threads

- Operating System
  - Extremely lightweight with global functionality (memory management, communication, etc.)
  - Heavily multithreaded locally for latency tolerance

Where is revolution required?

- Software to enable reliable systems built with unreliable parts
  - Infrastructure to enable application resiliency
  - Programming Models
  - System Software
  - APIs
- Finding and Expressing parallelism
  - User perspective (how to code it)
  - Compiler perspective (how to render what the user has expressed)
  - Make extremely fine-grain, massive, μthreading practical and effective
  - Exploit heterogeneous concurrency (computation, communication, I/O)
- Programming Tools
  - Intelligently collect data
  - Provide space efficient format for data storage
  - Collapse, reduce, filter data

How do we get it done?

- Define the overall architecture
  - Can we converge on a common architecture?
  - Establish well-defined interfaces between SW layers
  - Dedicated architects throughout the effort
- Establish a community for key projects
  - Dedicated maintainers
  - Research + Industrial partnerships with funding for both
  - User community participation
- Avoid "Design by Committee"
  - HPF, Ada are examples to avoid
  - Respected leaders make the tough calls

How do we get it done?

- Focus on the full SW life-cycle, not just the initial development
  - Test and integration
  - Maintenance
  - Management of the rate of change

- Provide a common exascale test and integration platform
  - All components tested at scale on a reference platform
  - Strong focus on:
    - Mainline testing
    - Error-path testing
    - Edge-condition/interface testing

- Resolve Differentiation Needs vs. Commonality Needs
  - Hardware has been commoditizing over time
  - Can common SW provide opportunities for vendor differentiation?

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara

Jean-Yves Berthou

Mitsuhisa Sato

Peggy Williams

Vivek Sarkar

Ann Trefethen

Acknowledgments

- DARPA Exascale Software Study team members
- Habanero Multicore Software Research project at Rice
  - Sponsors and donors: AMD, BHP Billiton, DARPA, IBM, Intel, Microsoft, NSF, NVIDIA, Sun
- CScADS project (Rice, ANL, UCB, UTK, UWM)
- IBM PERCS & X10 team members

This work was supported in part by the National Science Foundation under the HECURA program, award number CCF-0833166. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect those of the National Science Foundation.

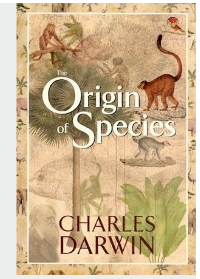
Context for ExaScale Software Study (in progress)

- Characteristics of Extreme Scale systems:
  - Massive multi-core (~ 1000 cores/chip)
  - Performance driven by parallelism, constrained by energy
  - Three system classes --- Exascale Data Center, Petascale Departmental, Terascale Embedded
- Key Software Challenges:
  - ***Concurrency***
  - ***Energy***
  - ***Resilience***
- Software stack:
  - Application frameworks & Tools
  - Programming models and languages
  - Libraries
  - Compilers
  - Runtimes for scheduling, memory management, communication, performance monitoring, power management, resilience, storage (including metadata access)
  - Operating & Storage System – persistence support

Extreme Scale software need long-term research that goes beyond industry efforts in cloud computing and manycore accelerators

Software-hardware co-design will be critical to the success of future Exascale systems

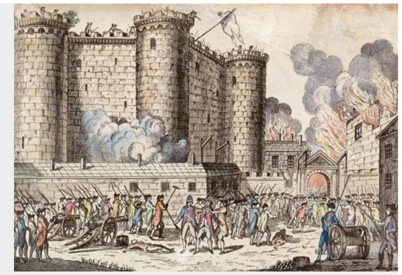
Three Software areas where Evolution is Necessary



1. ***Performance Analysis Tools***
   - Extensions for multithreaded code
   - Extensions for calling contexts
   - Progress under way in SciDAC centers such as CScADS & PERI
2. ***Node Compilers***
   - Adjust and adapt to proliferation of new multicore processors
   - Extend auto-tuning techniques with online & offline learning
   - DARPA AACE program will provide a major boost to this area
3. ***MPI + Dynamic Parallelism***
   - MPI Communicators are founded on fixed process structures
   - Process structures will need to change dynamically to address needs of emerging HPC applications (adaptive/unstructured grids, coupled models) and architectures (manycore)

RICE

Three Software areas where Revolution is Required



1. ***Fine-grained Asynchronous Parallelism***
   - Weak scaling and bulk-synchronous parallelism will not deliver billion-way concurrency needed in Exascale systems
   - Instead require unified abstractions of asynchrony and concurrency for multi-core & cluster parallelism
     - Subsumes threads, shared memory, message-passing, active messages, …
2. ***Locality Models***
   - Data movement will be major contributor to energy consumption in Exascale systems
   - Need locality models that enable programmer, compiler, and runtime to manage data movements across multiple levels of memory hierarchy
3. ***Software-Hardware co-design for Exascale systems***
   - IESP effort should identify software interfaces that are critical bottlenecks, and drive vendors to provide hardware support for software-hardware co-design of these interfaces
   - Examples to follow

Example Opportunities for Software-Hardware Co-Design

- Dynamic parallelism with fine-grained tasks (async, spawn, …)
  - Hardware support for scheduling data structures
- Distribution and co-location of tasks and data (places, locales, …)
  - Hardware support for virtual-to-physical translation and inter-place data transfers
- Collective and point-to-point synchronization with dynamic parallelism (barriers, phasers, …)
  - Hardware support for intra-node & inter-node synchronization and communication
- Producer-consumer parallelism (single-assignment vars, futures, …)
  - Hardware support for full-empty bits
- Isolation and mutual exclusion
  - Transactions, fine-grained locks
- Data parallelism
  - Vectors, SIMD, SIMT

Candidate items for Software-Hardware Interface

- Memory hierarchy configurations
  - Cache sizes & geometries, hardware vs. software cache coherence
  - Register file sizes and data widths
- Memory access patterns
  - Address ranges that should bypass cache
  - Address ranges that require hardware coherence
  - Address ranges for which coherence will be managed by software
  - Address ranges with values that are guaranteed to be read-only (immutable) for certain application phases
- Network bandwidth partitioning for different forms of data movement and communication
  - PGAS, RDMA, Message passing, Stream processing, …
- Other network reconfigurability parameters
  - Topology, Packet size, …
- Power management
  - Frequency scaling, Voltage scaling, …
- Performance profiling
  - Lightweight profiling, Identification of events to be counted and sampled, …
- Resilience
  - Identification of threads with lower resilience requirements e.g., for which software can perform error detection and recovery

RICE

From Powerpoint to Action

- Directed research needed for all 6 topics (and more)
  - Revolutionary areas --- let a thousand flowers bloom
    - Users will vote with their feet (and noses)
  - Evolutionary areas --- opportunities for consolidation starting with performance tools
- Application drivers
  - Application stakeholders should contribute sample applications and/or SSCA's --- requires effort, but will pay great dividends
- Platform drivers
  - Platform stakeholders should contribute to development, testing and integration for their platform --- requires effort, but will pay great dividends
- Coordination
  - Follow best practices of successful open source projects --- open development, continuous integration, continuous testing, customer focus, community involvement, meritocratic leadership, …
  - Open source participation in selected areas can be strategic to vendors too
    - For example, see IBM Systems Journal special issue on Open Source Software, Volume 44, Number 2, June 2005 for open source experiences by a range of IBM project
  - Software-hardware co-design – don't let software play second fiddle to hardware!

Software Barriers for HPC

Moderator

Pete Beckman

Presenters

Al Gara

Jean-Yves Berthou

Mitsuhisa Sato

Peggy Williams

Vivek Sarkar

Ann Trefethen

UK Roadmap activity

Leveraging work in the US and Europe together with UK specific workshops and discussions groups have lead to barriers for software development that fall into five themes

1. Cultural Issues
   - some people won't share...
2. Applications and Algorithms
   - Need to bring application and algorithm development closer
   - Need new algorithms for new architectures
3. Software Challenges
   - Engineering, portability, programming models, ...
4. Sustainability
   - Need better models for sustainability not only for UK efforts but those that we depend on!
5. Knowledge base
   - It would be good to know who is doing what and where
   - We need to train more people with this cross cutting set of skills.

http://www.oerc.ox.ac.uk/research/hpc-na♪

Evolution x 3

- Communication libraries
  - Cleverer
- Numerical and visualisation algorithms and libraries and tools {need both evolution and revolution}
- Integration of systems of models across scales and the like are increasingly important – need to evolve support for this – error propagation.
- Best practice software engineering....

Revolution x 3

- Portability
  - Architecture dependent code-generation
  - Dynamic adaptation
  - Check out on one platform check in on another
- Programmability
  - Develop systems that let us drive the machine with the hood down – better abstractions
- Dependability
  - On this scale things will fail – but it shouldn't mean they're broken
- Validation
  - Garbage generated in milliseconds is still garbage

- What can we learn from our formal methods colleagues?

Playing together

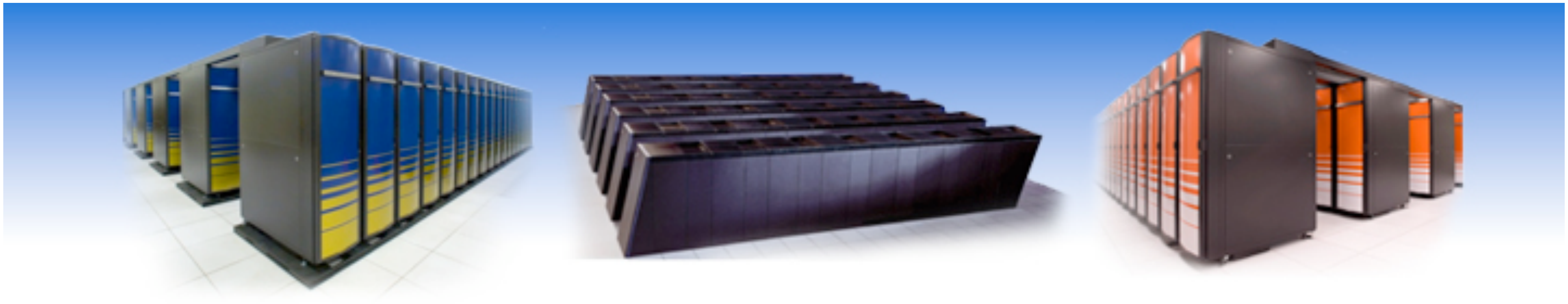
- Collaborative development of a roadmap for exascale software – several such already underway at the national level
- We need better coordination at the international programme level including mechanisms for collaboratively funded research and development
- Integration of applications, numerical and system software – silos of activity will not achieve our aims – US is better at than UK at this.
- Better models for sustainability
  - Community support?
  - Industry take-up
  - Need to ensure exascale efforts are not for the few
- Shared knowledge base required.

Playing together

- Success stories include BLAS, LAPACK, MPI, GPNL (what is that library called), PetSC
  - Good requirements capture, careful design, well engineered, well supported, used by many
- Support models:
  - Community support with funding agency investments
  - Vendor supported due to user requirements (eg MPI)
  - Industry support through direct licensing (library that Rolls Royce using), through integration into products, Matlab, NAG, ...
- Failures
  - Too many to mention – badly designed and/or engineered, no industry leverage.. Etc..
  - Created for a single audience or application area (CCPs)
  - Support model has relied on continuing investment from research councils (much grid software)
  - Tied to a particular architecture (CMSSL)

Playing together

- Ongoing activity – apace development site
  http://apace.myexperiment.org/



Science Drivers, Current HPC Software Development, and Platform Deployment Plans for the USA

Horst Simon
Lawrence Berkeley National Laboratory and UC Berkeley

IESP Workshop, Santa Fe, NM
April 7, 2009

IESP INTERNATIONAL EXASCALE SOFTWARE PROJECT

U.S. DEPARTMENT OF ENERGY
Office of Science

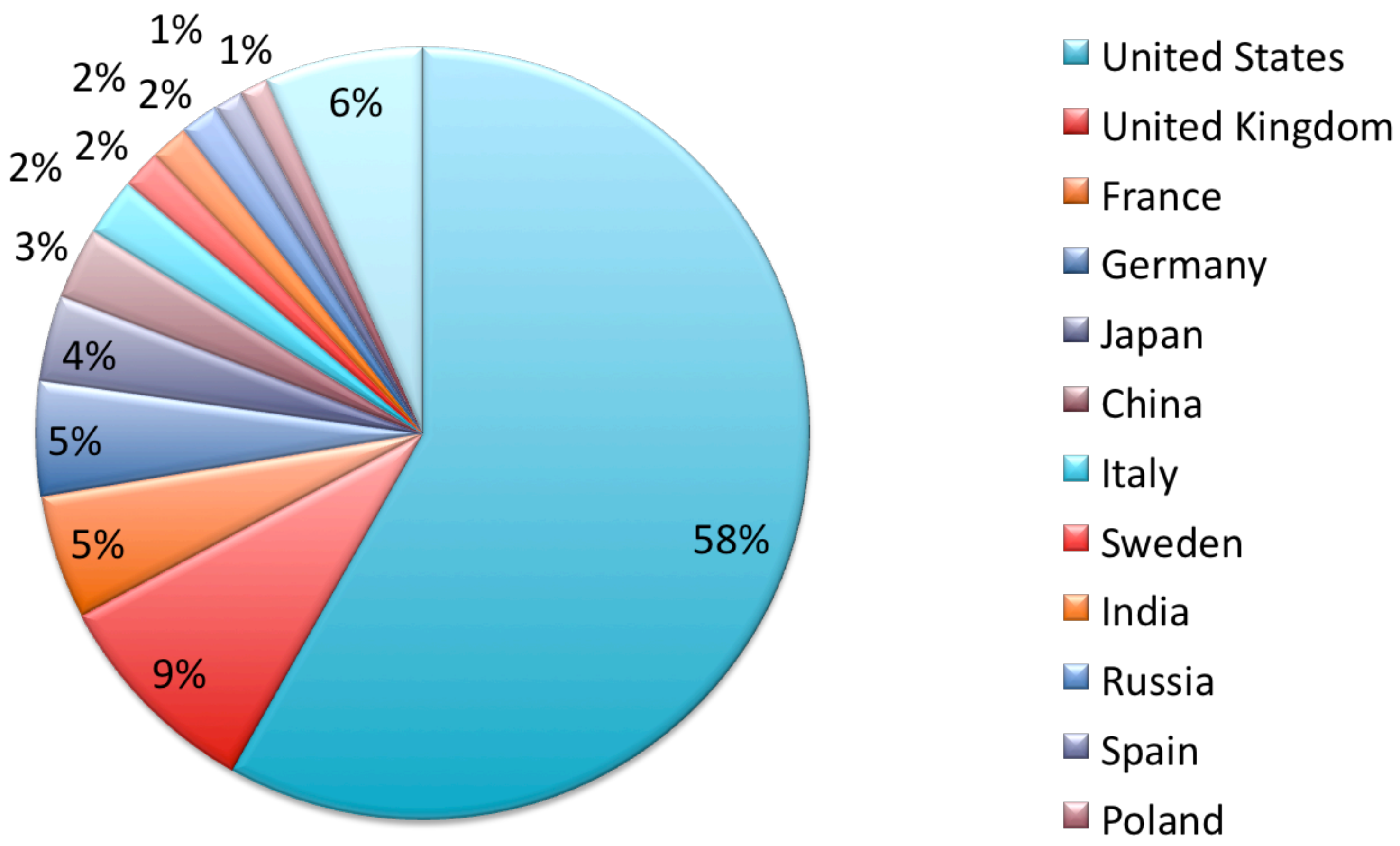
Acknowledgements

This presentation is a collection of slides contributed by Pete Beckman, Bill Gropp, Matt Leininger, Paul Messina, Abani Patra, Rob Pennington, Mark Seager, Ed Seidel, Rick Stevens, Michael Strayer, TOP500 team, Thomas Zacharia … and probably many others.

Overview

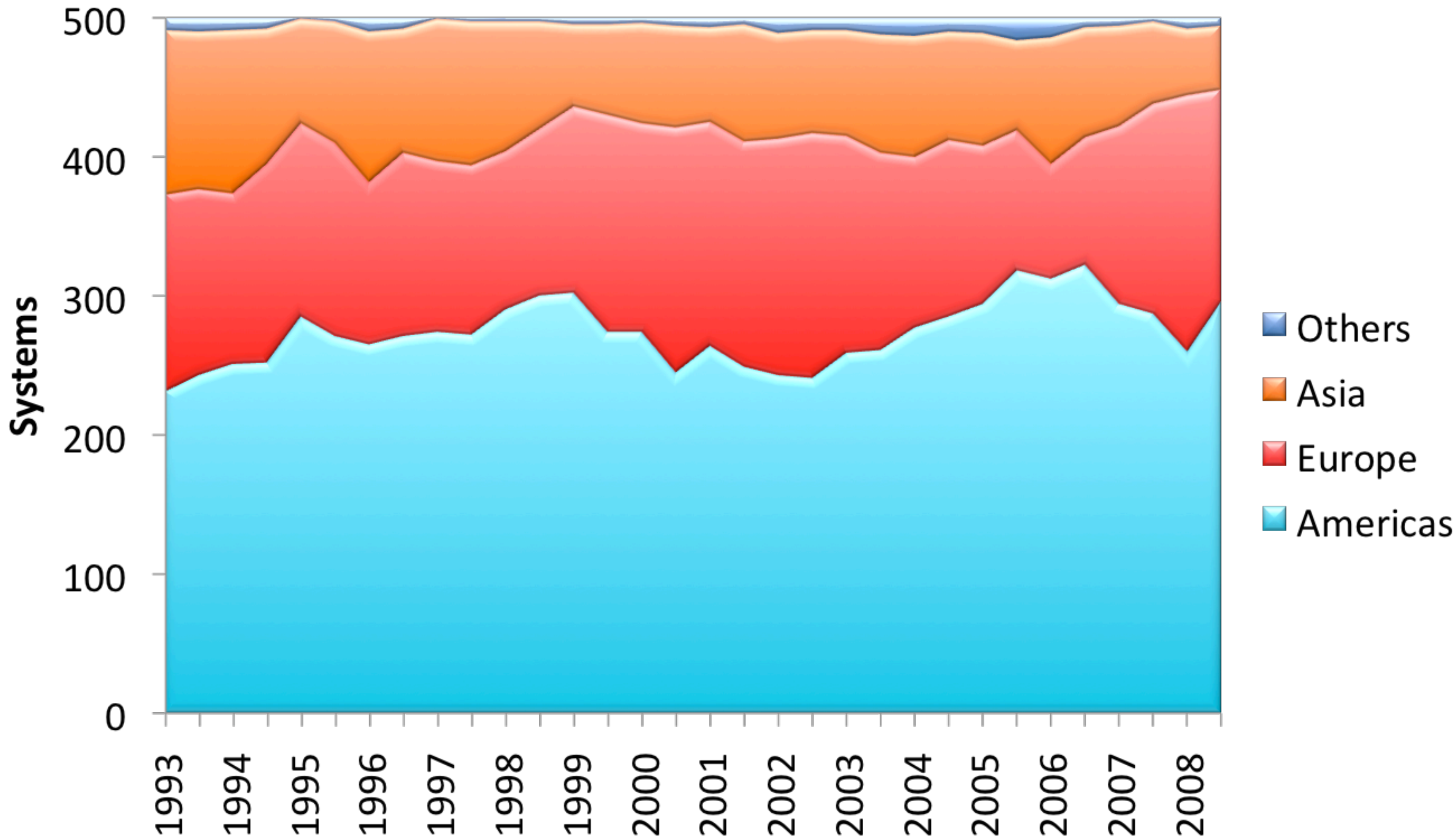
- State of HPC in the US
- Application Drivers
- Platforms Plans
- Software Development

Countries / System Share



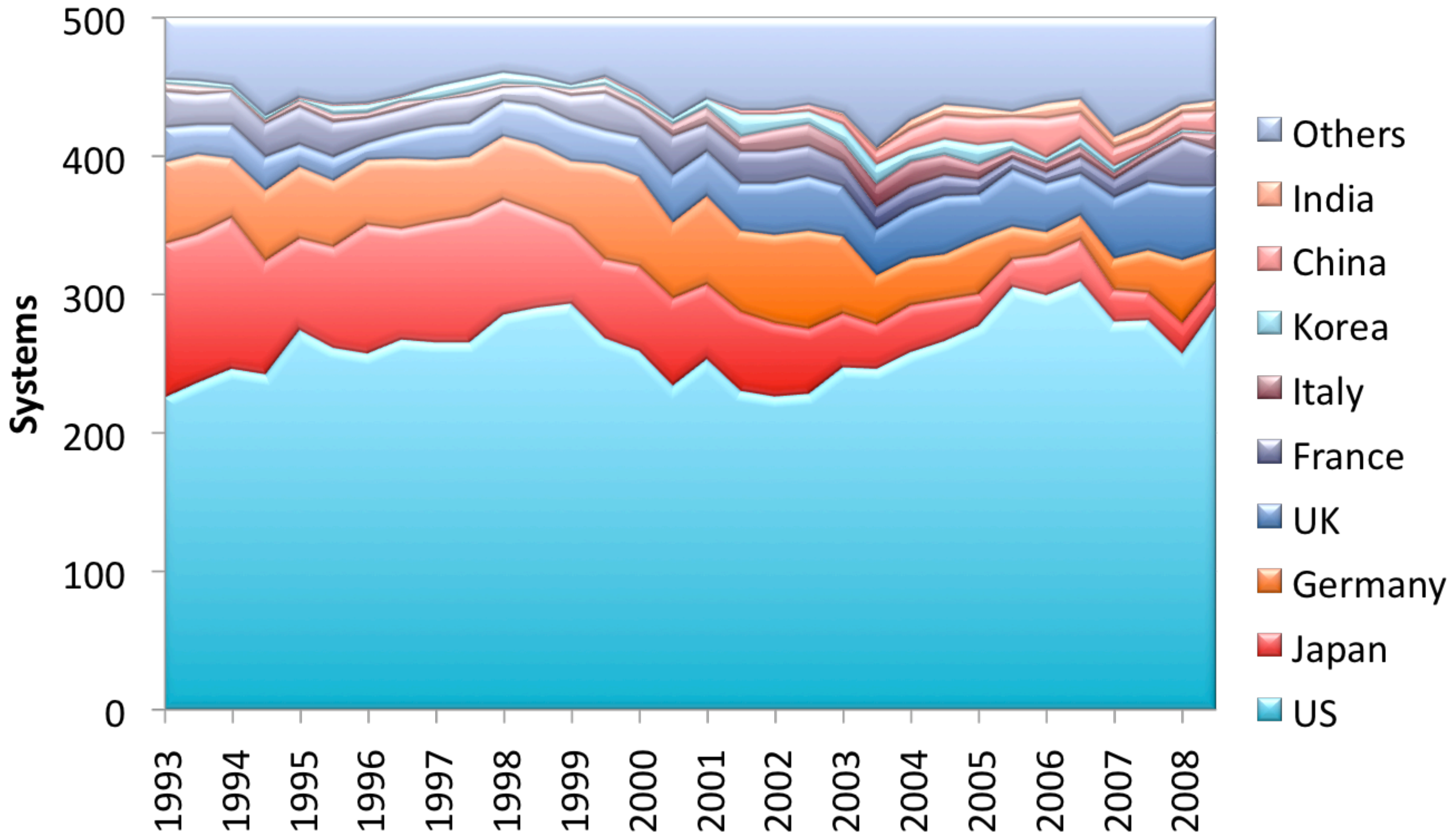
TOP500 SUPERCOMPUTER SITES

Continents



TOP500 SUPERCOMPUTER SITES

Countries



500
400
300
200
100
0
Systems
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
Others
India
China
Korea
Italy
France
UK
Germany
Japan
US

Roadrunner Breaks the Pflop/s Barrier

- **1,026 Tflop/s on LINPACK reported on June 9, 2008**
- **6,948 dual core Opteron + 12,960 cell BE**
- **80 TByte of memory**
- **IBM built, installed at LANL**

Cray XT5 at ORNL -- 1 Pflop/s in November 2008

Jaguar	Total	XT5	XT4
Peak Performance	1,645	1,382	263
AMD Opteron Cores	**181,504**	**150,176**	**31,328**
System Memory (TB)	**362**	**300**	**62**
Disk Bandwidth (GB/s)	**284**	**240**	**44**
Disk Space (TB)	**10,750**	**10,000**	**750**
Interconnect Bandwidth (TB/s)	**532**	**374**	**157**

The systems will be combined after acceptance of the new XT5 upgrade. Each system will be linked to the file system through 4x-DDR Infiniband

NITRD Agency Budgets (FY09 Request)

Agency		High End Computing Infrastructure & Applications (HEC I&A)	High End Computing Research & Development (HEC R&D)	Cyber Security & Information Assurance (CSIA)	Human-Computer Interaction & Information Management (HCI &IM)	Large Scale Networking (LSN)	High Confidence Software & Systems (HCSS)	Social, Economic, & Workforce Implications of IT (SEW)	Software Design & Productivity (SDP)	Total [1]
NSF	2008 Estimate	257.4	78.6	68.1	234.8	82.6	56.6	98.6	54.8	931.5
	2009 Request	**298.4**	**91.5**	**87.6**	**266.5**	**95.8**	**67.6**	**112.0**	**70.8**	**1,090.3**
DARPA			92.0	124.4	205.3	109.0				530.7
			142.6	**106.8**	**184.9**	**135.9**				**570.2**
OSD and DoD Service research orgs. [2]		247.6	18.1	38.6	109.6	136.1	25.6		6.7	582.3
		249.6	**15.6**	**40.7**	**92.9**	**114.1**	**26.9**		**7.8**	**547.5**
NIH		159.4	76.4	1.1	182.7	68.1	7.7	10.8	4.6	510.7
		159.4	**76.3**	**1.1**	**181.7**	**68.0**	**7.7**	**10.8**	**4.6**	**509.6**
DOE/SC/NE/FE [3]		282.0	73.1			47.6		5.0		407.6
		334.6	**73.1**			**52.2**		**5.0**		**465.0**
NSA			93.5	15.5		2.9	25.2			137.1
			72.6	**17.8**		**1.8**	**27.2**			**119.3**
NASA		59.4		0.3	6.5	1.3	4.8			72.3
		60.1		**0.2**	**5.5**	**0.7**	**4.3**			**70.7**
NIST		10.7	2.4	20.8	11.8	5.8	4.9		5.6	62.0
		10.7	**2.4**	**25.8**	**11.8**	**5.8**	**4.9**		**5.6**	**67.0**
AHRQ					39.8	5.0				44.8
					39.8	**5.0**				**44.8**
DOE/NNSA		8.4	14.3			1.3		4.3		28.3
		8.2	**15.7**			**0.9**		**4.7**		**29.5**
NOAA		15.9	1.9		0.5	2.9			1.6	22.8
		18.0	**1.9**		**0.5**	**2.9**				**23.3**
EPA		3.3			3.0					6.3
		3.3			**3.0**					**6.3**
NARA					4.5					4.5
					4.5					**4.5**
TOTAL (2008 Estimate) [1]		1,044.1	450.4	268.7	798.5	462.4	124.8	118.7	73.3	3,341
TOTAL (2009 Request) [1]		**1,142.4**	**491.8**	**279.8**	**791.2**	**483.0**	**138.5**	**132.6**	**88.7**	**3,548**

Annual HPC Investment in the US (FY09)

- **High End Computing Infrastructure and Applications $1,142 M**
- **High End Computing R&D $492 M**

32nd List: The TOP10

Rank	Site	Manufacturer	Computer	Country	Cores	Rmax [Tflops]	Power [MW]
1	DOE/NNSA/LANL	IBM	Roadrunner - BladeCenter QS22/LS21	USA	129600	1105.0	2.48
2	Oak Ridge National Laboratory	Cray Inc.	Jaguar - Cray XT5 QC 2.3 GHz	USA	150152	1059.0	6.95
3	NASA/Ames Research Center/NAS	SGI	Pleiades - SGI Altix ICE 8200EX	USA	51200	487.0	2.09
4	DOE/NNSA/LLNL	IBM	eServer Blue Gene Solution	USA	212992	478.2	2.32
5	Argonne National Laboratory	IBM	Blue Gene/P Solution	USA	163840	450.3	1.26
6	Texas Advanced Computing Center/ Univ. of Texas	Sun	Ranger - SunBlade x6420	USA	62976	433.2	2.0
7	NERSC/LBNL	Cray Inc.	Franklin - Cray XT4	USA	38642	266.3	1.15
8	Oak Ridge National Laboratory	Cray Inc.	Jaguar - Cray XT4	USA	30976	205.0	1.58
9	NNSA/Sandia National Laboratories	Cray Inc.	Red Storm - XT3/4	USA	38208	204.2	2.5
10	Shanghai Supercomputer Center	Dawning	Dawning 5000A, Windows HPC 2008	China	30720	180.6	

Focus of this Presentation

- DOE – SC
- DOE – NNSA
- NSF

Overview

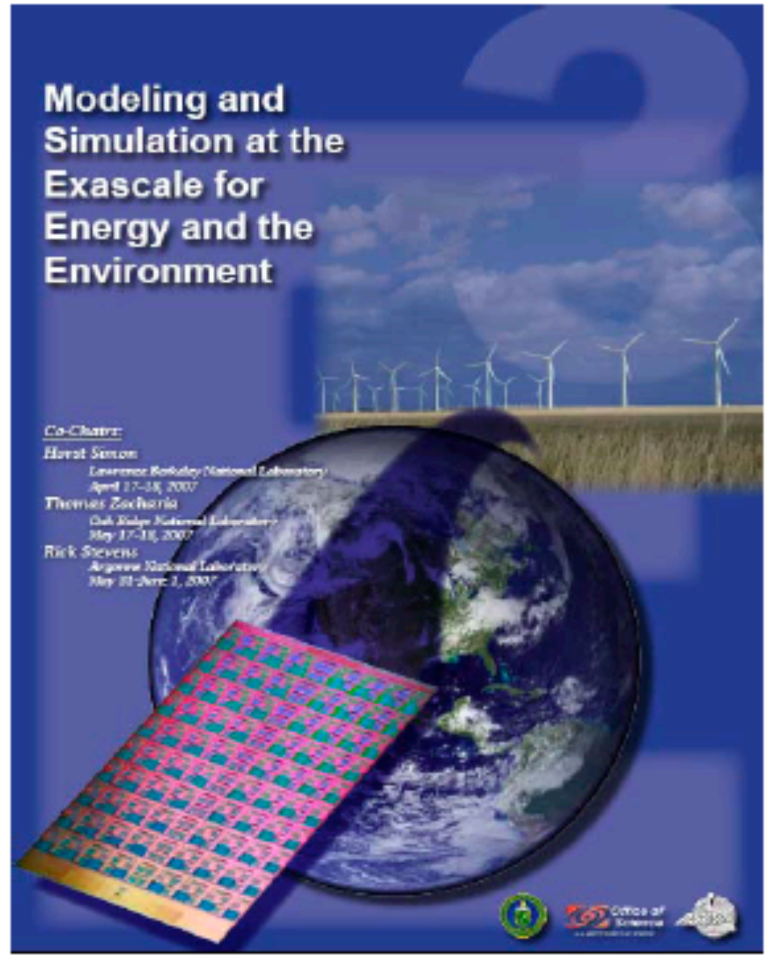
- State of HPC in the US
- Application Drivers
- Platforms Plans
- Software Development

Preparing for Extreme Scale Computing

Three Town Hall Meetings held April-June, 2007

Climate, Combustion, Fusion, Fission
Solar, Biology, Socioeconomic Modeling and Astrophysics

Mathematics, Computer Science
Algorithms, Software infrastructure and Cyberinfrastructure



Integrated program- investments in hardware and software research and development

Tight coupling to a selected set of science communities and the associated applied mathematics R&D.

Break Out Groups (applications)

B1. Improve our understanding of complex biogeochemical (C, N, P, etc.) cycles that underpin global ecosystems functions and control the sustainability of life on Earth.

B2. Develop and optimize new pathways for renewable energy production and development of long-term secure nuclear energy sources, through computational nanoscience and physics-based engineering models.

B3. Enhance our understanding of the roles and functions carried out by microbial life on Earth, and adapt these capabilities for human use, through bioinformatics and computational biology.

B6. Develop integrated modeling environments that couple the wealth of observational data and complex models to economic, energy, and resource models that incorporate the human dynamic into large-scale global change analysis.

B9. Develop a “cosmic simulator” capability that integrates increasingly complex astrophysical measurements with simulations of the growth and evolution of structure in the universe, linking the known laws of microphysics to the macro world. Develop large-scale, special-purpose computing devices and innovative algorithm development to achieve this goal.

B10. Manufacturing

Break Out Groups (technology)

B4. Develop tools and methods to protect the distributed information technology infrastructure: ensuring network security, preventing disruption of our communications infrastructure, and defending distributed systems against attacks.

B5. Drive innovation at the frontiers of computer architecture and information technology, preparing the way for ubiquitous adoption of parallel computing, power-efficient systems, and the software and architectures needed for a decade of increased capabilities. Accelerate the development of special-purpose devices that have the potential to change the simulation paradigm for certain science disciplines.

B7. Advance mathematical and algorithmic foundations to support scientific computing in emerging disciplines such as molecular self- assembly, systems biology, behavior of complex systems, agent-based modeling and evolutionary and adaptive computing.

B8. Integrate large, complex, and possibly distributed software systems with components derived from multiple applications domains and with distributed data gathering and analysis tools.

Scientific Challenge Workshop Series (2008 – 2009)

- **Climate, Nov. 2008**
- **Astrophysics, HEP, Experimental Particle Physics, HE Theoretical Physics, Dec. 2008**
- **Nuclear Physics, Jan. 2009**
- **Fusion Energy, March 2009**
- **Nuclear Energy, May 2009**
- **Combustion, Nanoscience, Chemistry, August 2009**
- **Biology, Sept. 2009**
- **NNSA and SC Mission, Sept/Oct. 2009**

Scientific Challenge Workshop Series (2008 – 2009)

- **Series of workshops organized as follow up by DOE-SC (Paul Messina):**
  - **To identify grand challenge scientific problems in [research area] that can exploit computing at extreme scales to bring about dramatic progress toward their resolution.**
  - **The goals of the workshops are to**
    - **identify grand challenge scientific problems […] that could be aided by computing at the extreme scale over the next decade;**
    - **identify associated specifics of how and why new high performance computing capability will address issues at the frontiers of […]; and**
    - **provide a forum for exchange of ideas among application scientists, computer scientists, and applied mathematicians to maximize the use of extreme scale computing for enabling advances and discovery in […].**

Priority Research Direction

Scientific and computational challenges

Brief overview of the underlying scientific and computational challenges

Summary of research direction

What will you do to address the challenges?

Potential scientific impact

What new scientific discoveries will result?

What new methods and techniques will be developed?

Potential impact on SCIENCE DOMAIN

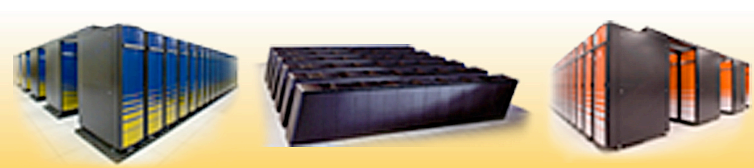
How will this impact key open issues in SCIENCE DOMAIN?

What's the timescale in which that impact may be felt?

PRDs for Climate Model Development and Integrated Assessment

(from Warren Washington's presentation to BERAC)

- How do the carbon, methane, and nitrogen cycles interact with climate change?
- How will local and regional water, ice, and clouds change with global warming?
- How will the distribution of weather events, particularly extreme events, that determine regional climate change with global warming?
- What are the future sea level and ocean circulation changes?



PRDs for Algorithms and Computational Environment

(from Washington's presentation to BERAC)

- **Develop numerical algorithms to efficiently use upcoming petascale and exascale architectures**
- **Form international consortium for parallel input/output, metadata, analysis, and modeling tools for regional and decadal multimodel ensembles**
- **Develop multicore and deep memory languages to support parallel software infrastructure**
- **Train scientists in the use of high-performance computers.**

Cosmic Structure Formation Probes of the Dark Universe

Scientific and computational challenges

Understand cosmic structure to enable the use the universe as a probe of fundamental physics

Perform cosmological hydrodynamical simulations with the dynamic range necessary to interpret future experiments

Summary of research direction

Develop precise predictions of structure formation from the Hubble Volume to the scale of the Solar System

Develop spatially and temporally adaptive codes, algorithms, and workflows for simulations and data on extreme-scale architectures.

Potential scientific impact

Determine the equation of state of dark energy and distinguish between dark energy and modifications of General Relativity

Measure the masses and interactions of dark matter

Measure the sum of the neutrino masses

Probe the fields responsible for primordial fluctuations

Potential impact on High Energy Physics

Revolutionize High Energy Physics by discovering and measuring physics beyond the standard model inaccessible to accelerators.

10 years



The Software Dimension
Consensus view of Astrophysics Simulation and Data Panels

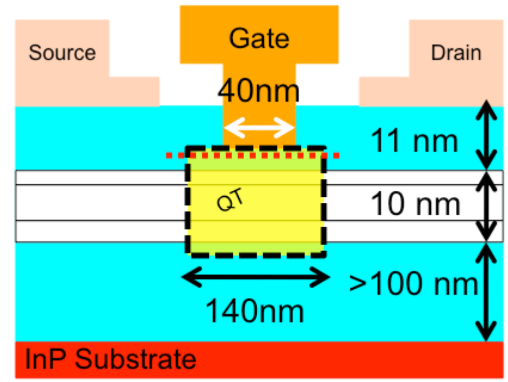
- Identify and support development of low-level modules and libraries, isolating architectural complexity (e.g., MPI, FFT)
- Identify and support development of open-source community application codes, but not to the exclusion of other promising efforts
- Promote development of data models and language for interoperable data analysis (observation <=> simulation)

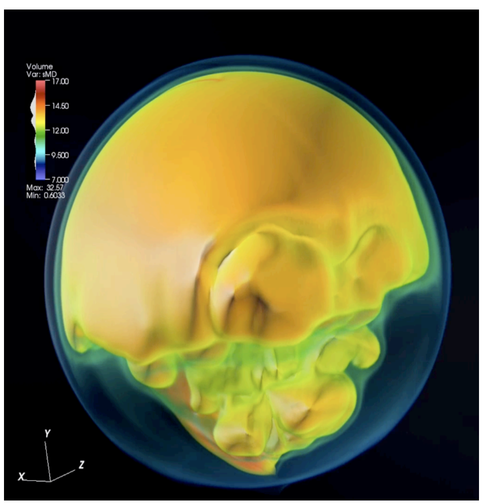
Selected PRDs identified by NP workshop

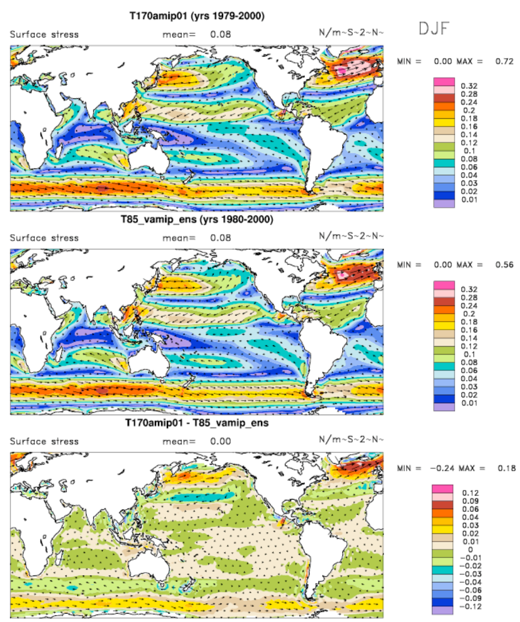
- Physics of extreme neutron-rich nuclei and matter
- Microscopic description of nuclear fission
- Early universe
- Stellar evolution
- Stellar explosions and their remnants

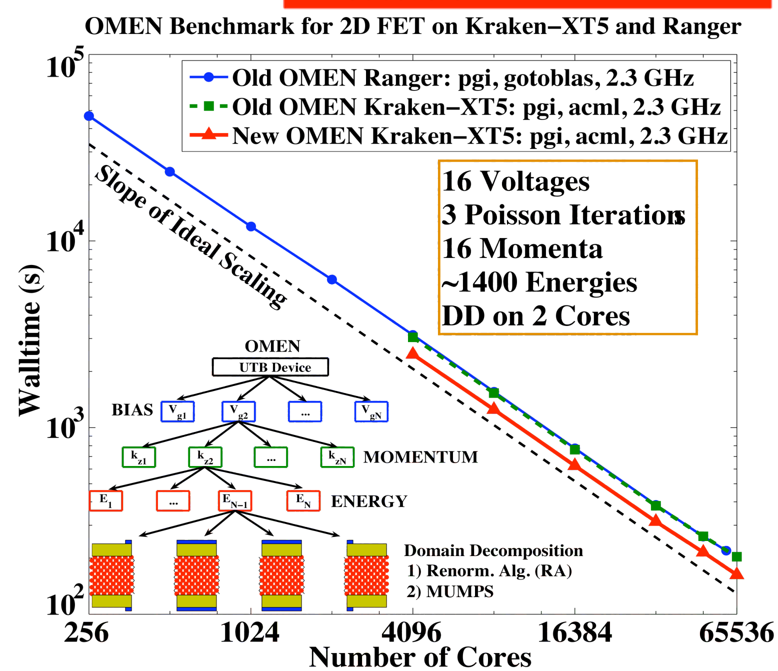
PetaApps Solicitation:NSF 07-559, 08-592

- Applications ranged over, climate change, earthquake dynamics and structural response, nanoscale transistor models, supernovae simulations, high Reynolds number turbulent flows, quantum chromodynamics …





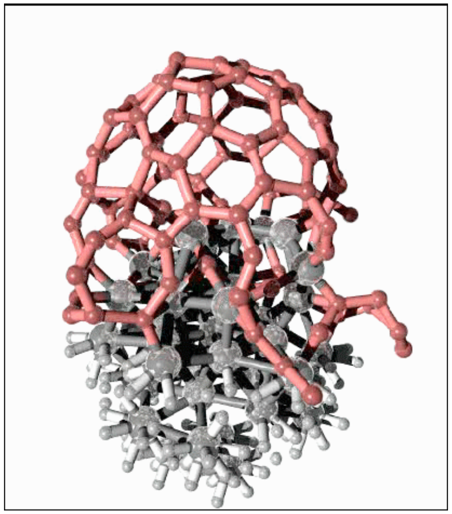


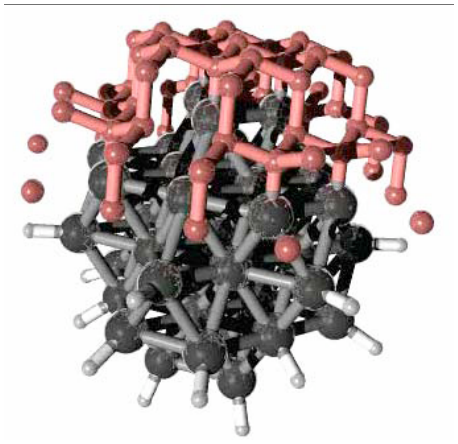


PetaApps Solicitation:NSF 07-559, 08-592

Solicitation sought proposals that

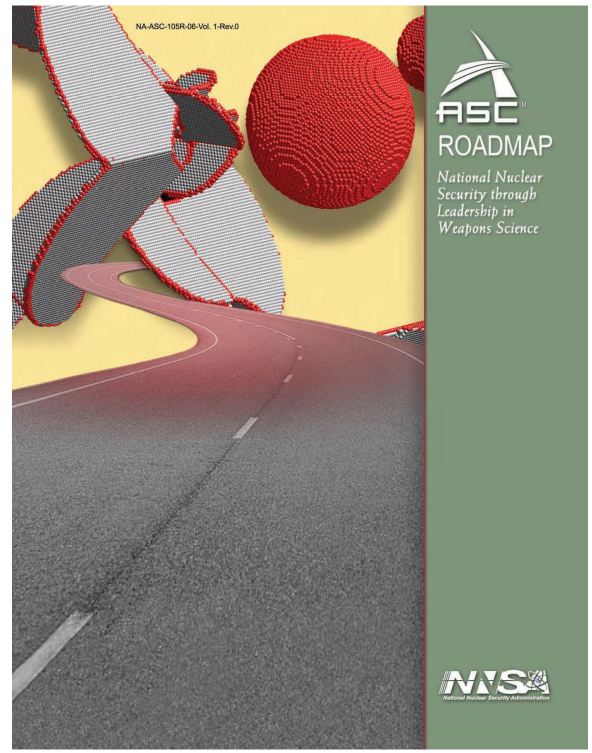
- develop the future simulation, optimization and analysis tools that can use emerging petascale computing to advance the frontiers of scientific and engineering research;
- have a high likelihood of enabling future transformative research;
- 133 distinct project proposals received;
- 18 awards ~$26M (50% funding from OCI, 50% from CISE, ENG, MPS)
- ~$30M investment planned for FY09-10





NNSA Advanced Simulation and Computing (ASC) Strategy Goals

- **Address national security simulations needs;**
- **Establish a validated *predictive capability* for key physical phenomena;**
- **Quantify and aggregate uncertainties in simulation tools;**
- **Provide mission-responsive computational environments.**



IESP INTERNATIONAL EXASCALE SOFTWARE PROJECT

U.S. DEPARTMENT OF ENERGY
Office of Science

Overview

- State of HPC in the US
- Application Drivers
- Platforms Plans
- Software Development

Oak Ridge's Cray XT5
Breaks the Petaflop Barrier

Jaguar	Total	XT5	XT4
Peak Performance	1,645	1,382	263
AMD Opteron Cores	**181,504**	**150,176**	**31,328**
System Memory (TB)	**362**	**300**	**62**
Disk Bandwidth (GB/s)	**284**	**240**	**44**
Disk Space (TB)	**10,750**	**10,000**	**750**
Interconnect Bandwidth (TB/s)	**532**	**374**	**157**

Argonne's IBM Blue Gene/P – 556 TFs

National Energy Research Scientific Computing Center (NERSC)

- Located at Lawrence Berkeley National Lab
  - Cray XT4 Franklin upgraded to 350 Tflop/s
  - Data facility with up to 50PBytes capacity

- NERSC-6 Project
  - RFP issued in September 2008
  - Installation 2009

Franklin

ESnet
40 Gbps Core

June–November, 2008

www.es.net

ESnet news

Princeton Gets a 6,400 Percent Increase in Bandwidth With ESnet Upgrades

ESnet finished improving its Internet connections to several institutions on Princeton University's Forrestal Campus, including the Princeton Plasma Physics Lab (PPPL), the High Energy Physics (HEP) Group within the Physics Department at Princeton University, and the National Oceanic and Atmospheric Administration's Geophysical Fluid Dynamics Laboratory (GFDL).

Now researchers around the globe can access data from these science facilities with increasing speeds and scalability, helping enable international collaborations on bandwidth-intensive applications and experiments.

"This is a great achievement," says Steve Cotter, head of ESnet. "With the availability of cutting-edge instruments and supercomputers, scientists around the world are collaborating to carry out large experiments that produce tremendous amounts of data. This upgrade links Princeton's physics researchers to that data through our robust and reliable network, ESnet4, via point-to-point dedicated circuits and IP services at multiple gigabit per second speeds."

The Princeton network upgrade took approximately five months to complete, and involved running fiber optic cabling underground from the Forrestal Campus outside Princeton, New Jersey, along Route 1 to South Brunswick, then to Philadelphia, where it is transported across the ESnet infrastructure to ESnet's main point of presence in McLean, Va.

On the Princeton campus, the PPPL's Internet connection is now operating at 10 gigabit speeds, 10 billion bits per second, and, significantly continued on page 2

PPPL (shown here) and GFDL are both located on Princeton University's Forrestal Campus.

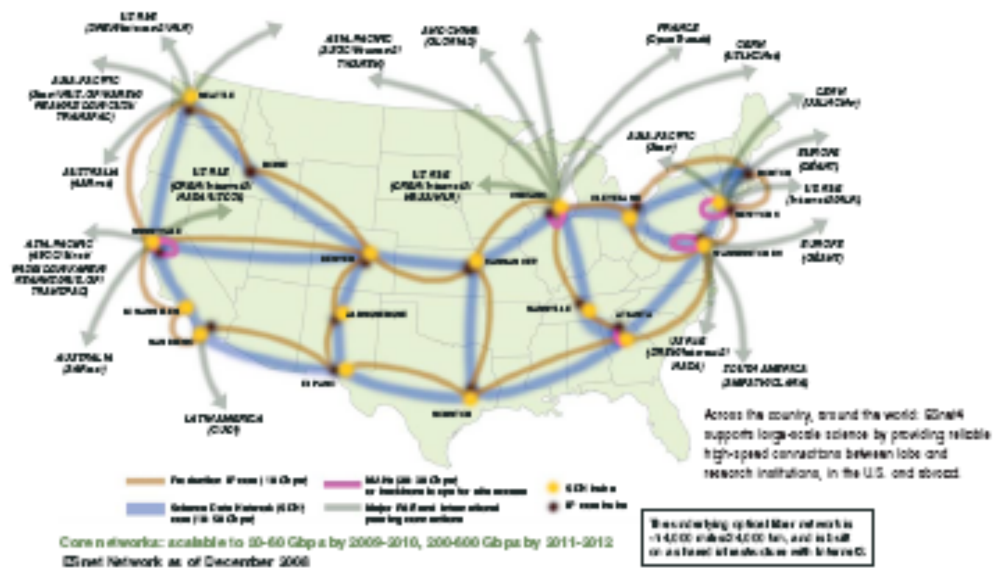
ESnet4 Provides Critical Link for U.S. Researchers Accessing LHC Data

Approaching the speed of light, millions of protons will collide per second when the Large Hadron Collider (LHC) comes online next year. The experiment will generate more data than the international scientific community has ever tried to manage. Scientists suspect the outcome of these "subatomic smashups" will provide valuable insights into the origins of matter and dark energy in the Universe.

As thousands of researchers across the globe anxiously await the results of this experiment, getting the massive amounts of data to them is no insignificant task. Fortunately, network engineers at the U.S. Department of Energy's (DOE) Energy Sciences Network (ESnet) foresaw this data challenge years ago and developed ESnet4, a new large-scale science data transport network with enough bandwidth to transport multiple streams of 10 gigabits of information per second — the equivalent of transmitting 500 hours of digital music per second for each 10 gigabit line.

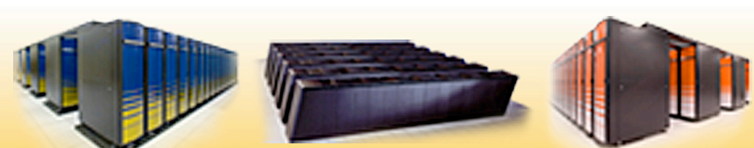
The LHC, which straddles the Swiss and French borders on the outskirts of Geneva, will be the first experiment to fully utilize the advanced capabilities of this network, which connects DOE national laboratories to researchers across the country and collaborators worldwide.

"ESnet4 is one of the most robust scientific data networks in existence," says Steve Cotter, Department Head for ESnet. "The science environment of today is very different from that of a few years ago. ESnet4 provides the high-speed, extremely reliable connectivity between labs and U.S. and international research institutions required to support the inherently collaborative, global nature of modern large-scale science." continued on page 3



Leader in Networks for Science

- OSCARS
- PerfSONAR
- DanteInternet2CanarieESnet

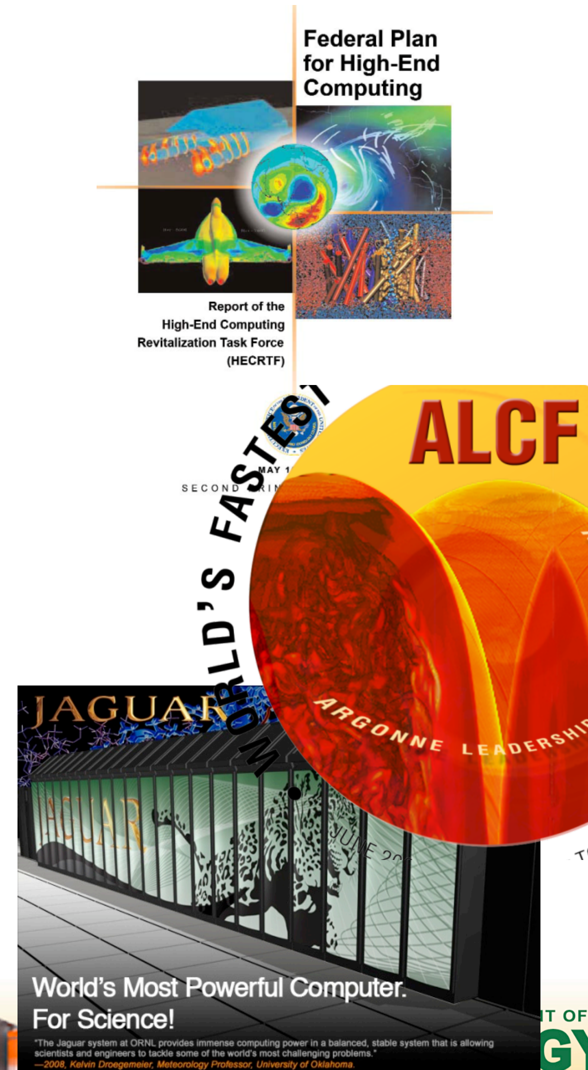


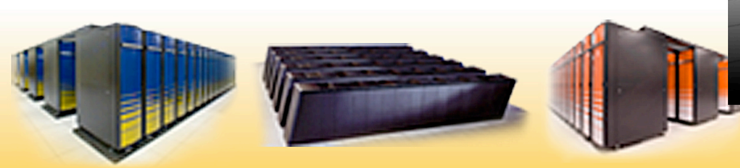
INTERNATIONAL EXASCALE SOFTWARE PROJECT

U.S. DEPARTMENT OF ENERGY
Office of Science

ASCR Facilities Strategy

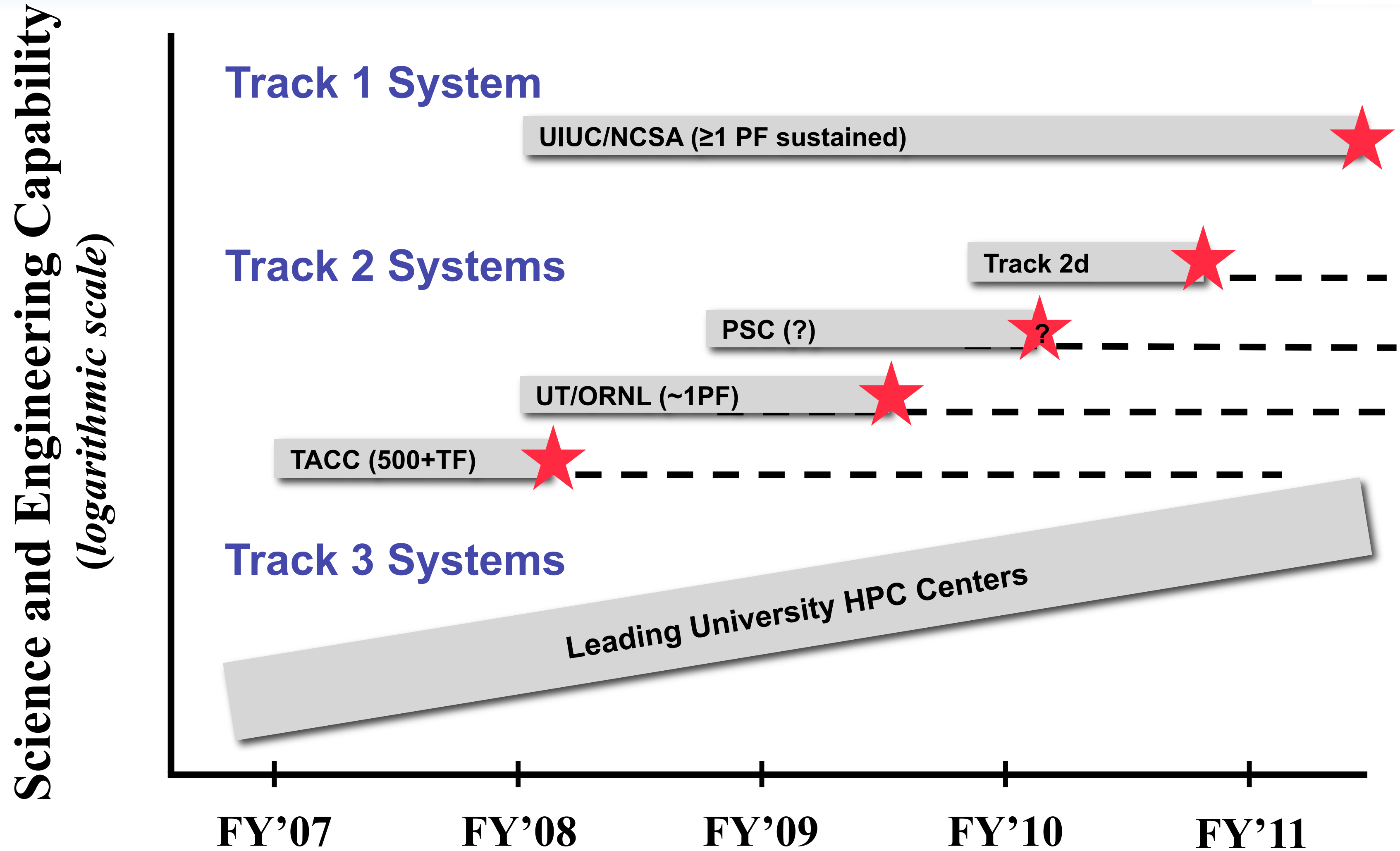
- **Providing the Tools** – High-End Computing
  - High-Performance Production Computing - National Energy Research Scientific Computing Center **(NERSC)** at Lawrence Berkeley National Laboratory
    - Delivers high-end capacity computing to entire DOE SC research community
  - Leadership-Class Computing – **Leadership Computing Centers at Argonne National Laboratory and Oak Ridge National Laboratory**
    - Delivers highest computational capability to national and international researchers through peer-reviewed **Innovative and Novel Computational Impact on Theory and Computation (INCITE)** program (80% of resources)
- **Investing in the Future** - Research and Evaluation Prototypes
- **Linking it all together** – Energy Sciences Network (ESnet)





Office of Science

NSF's Strategy for High-end Computing



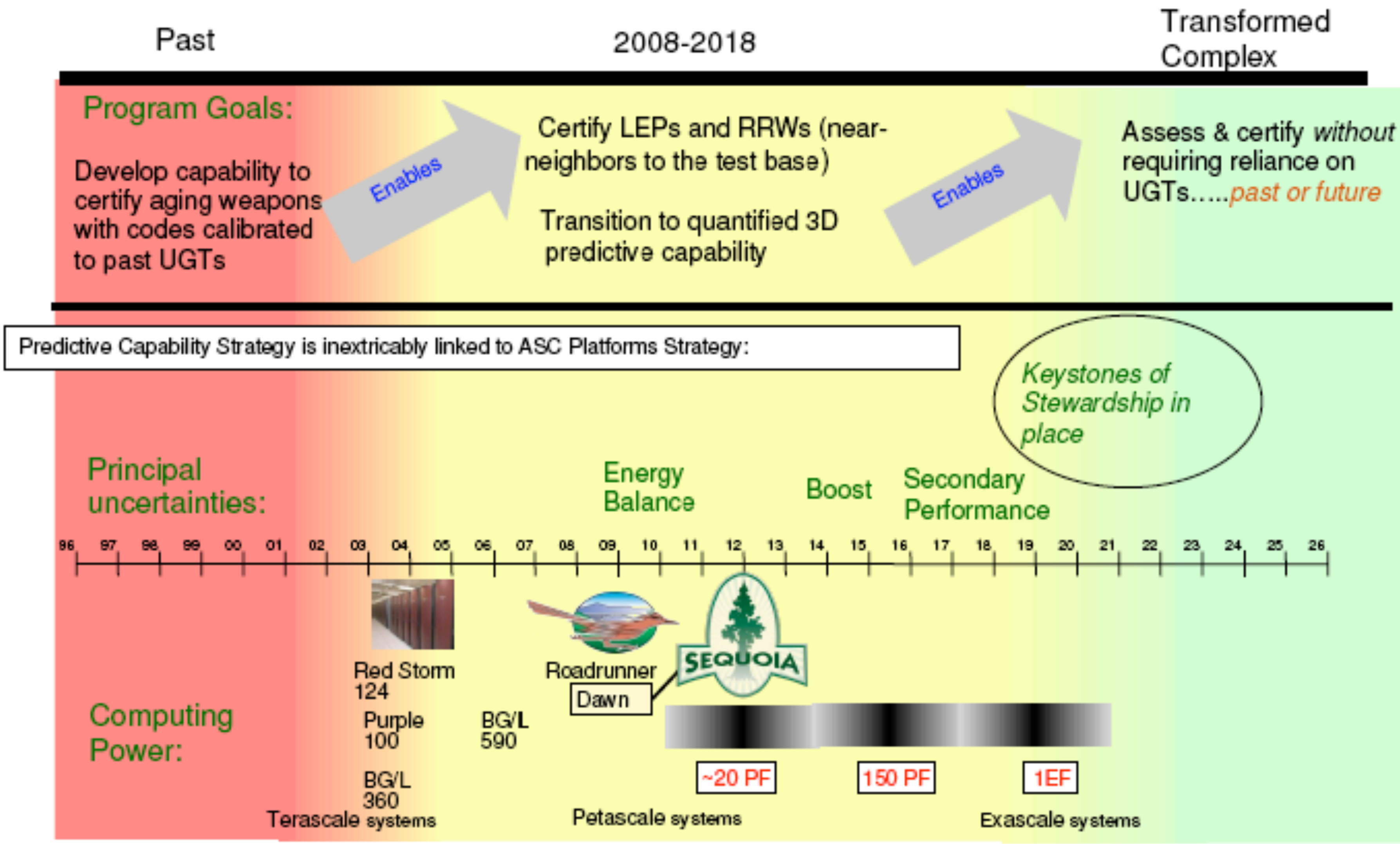
NSF's Track 2 Computing Systems

	TACC	UT-ORNL	PSC
System Attribute	**Ranger**	**Kraken**	**?**
Status	Installed	Installed	
Vendor	Sun	Cray	
Processor	AMD	Intel	
Peak Performance (TF)	504	~1000	
Number Cores/Chip	4	?	
Number Processor Cores	62,976	~80,000	
Amount Memory (TB)	123	~100	
Amount Disk Storage (TB)	194		
External Bandwidth (Gbps)	10		

Blue Waters Computing System at NCSA

System Attribute	Abe	Blue Waters
Vendor	Dell	IBM
Processor	Intel Xeon 5300	IBM Power7
Peak Performance (PF)	0.090	
Sustained Performance (PF)	0.005	≥1
Number Cores/Chip	4	multicore
Number Processor Cores	9,600	>200,000
Amount Memory (TB)	14.4	>800
Amount Disk Storage (TB)	100	>10,000
Amount of Archival Storage (PB)	5	>500
External Bandwidth (Gbps)	40	>100

ASC Continues with roadmap to exascale

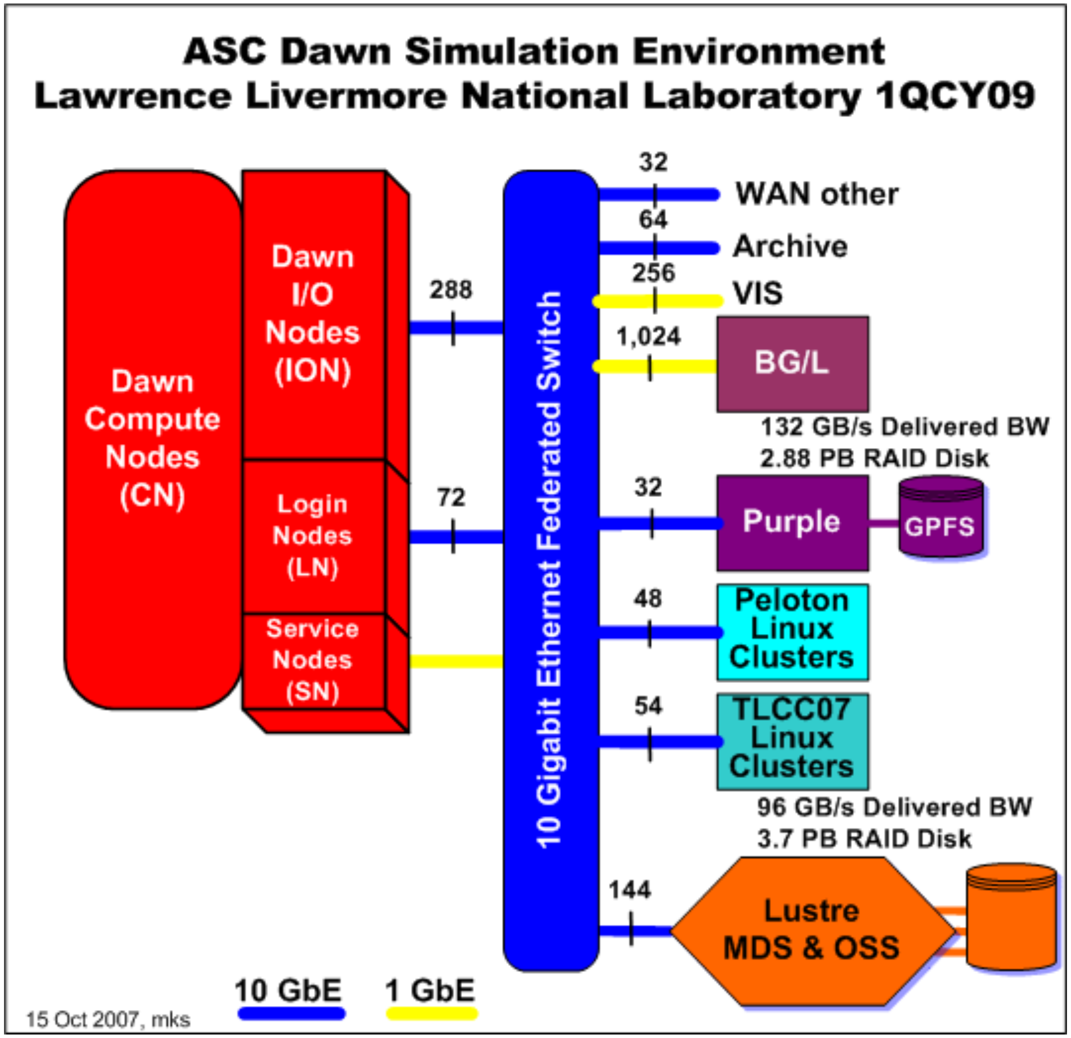


ASC RoadRunner and Sequoia is the dawn of the petascale era for predictive weapons science

Sequoia Strategy

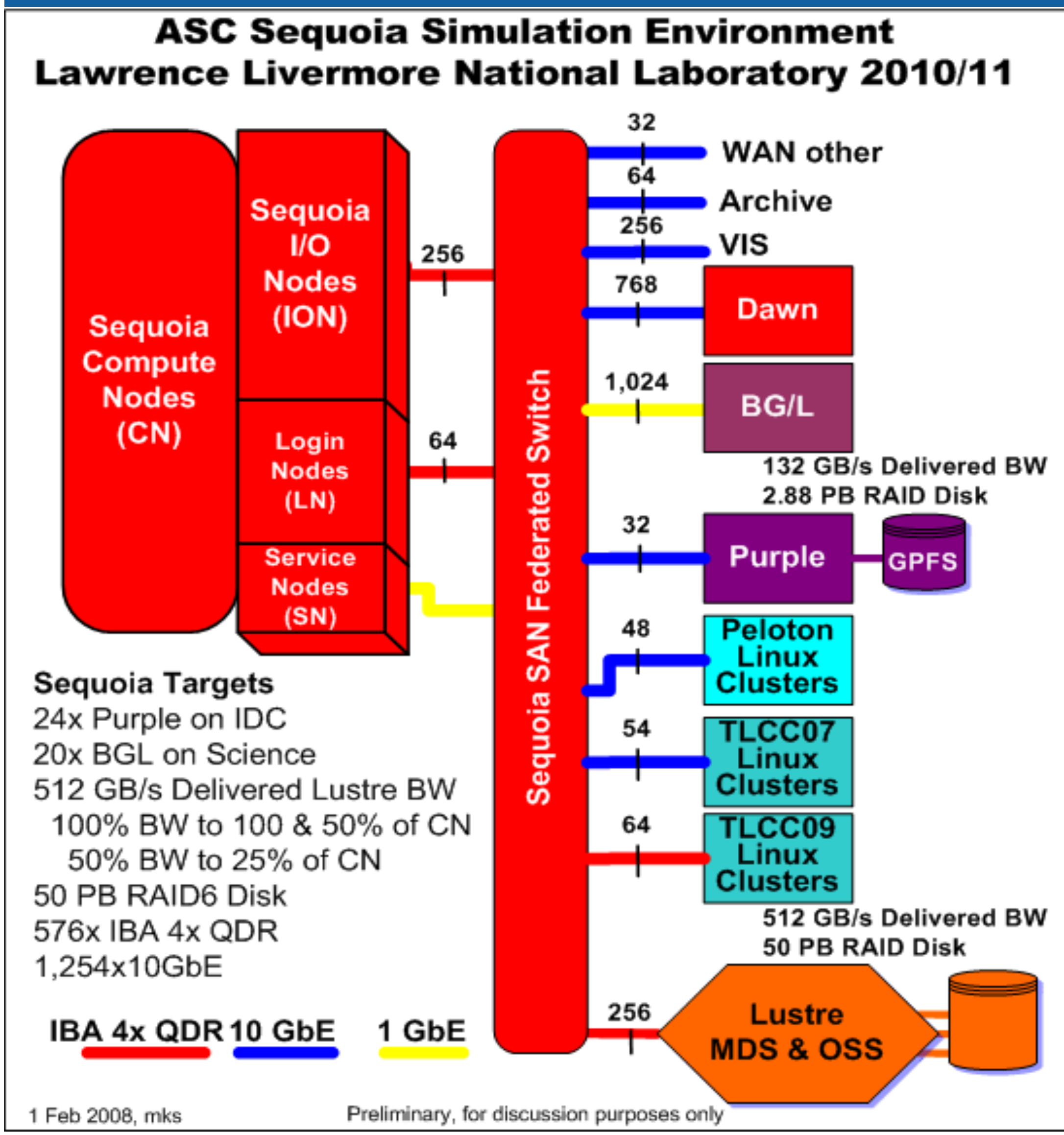
- Two major deliverables
  - Petascale Scaling “Dawn” Platform in 2009
  - Petascale “Sequoia” Platform in 2011
- Lessons learned from previous capability and capacity procurements
  - Leverage best-of-breed for platform, file system, SAN and storage
  - Major Sequoia procurement is for long term platform partnership
  - Three R&D partnerships to incentivize bidders to stretch goals
  - Risk reduction built into overall strategy from day-one
- Drive procurement with single peak mandatory
  - Target Peak+Sustained on marquee benchmarks
  - Timescale, budget, technical details as target requirements
  - Include TCO factors such as power

To Minimize Risk, Dawn Deployment Extends the Existing Purple and BG/L Integrated Simulation Environment



- ASC Dawn is the initial delivery system for Sequoia
- Code development platform and scaling for Sequoia
- 0.5 petaFLOP/s peak for ASC production usage
- Target production 2009-2014
- Dawn Component Scaling
  - Memory B:F = 0.3
  - Mem BW B:F = 1.0
  - Link BW B:F = 2.0
  - Min Bisect B:F = 0.001
  - SAN GB/s:PF/s = 384
  - F is peak FLOP/s

Sequoia Target Architecture in Integrated Simulation Environment Enables a Diverse Production Workload



- Diverse usage models drive platform and simulation environment requirements
  - Will be 2D ultra-res and 3D high-res Quantification of Uncertainty engine
  - 3D Science capability for known unknowns and unknown unknowns
- Peak of 14 petaFLOP/s with option for 20 petaFLOP/s
- Target production 2011-2016
- Sequoia Component Scaling
  - Memory B:F = 0.08
  - Mem BW B:F = 0.2
  - Link BW B:F = 0.1
  - Min Bisect B:F = 0.03
  - SAN BW GB/:PF/s = 25.6
  - F is peak FLOP/s

Overview

- State of HPC in the US
- Application Drivers
- Platforms Plans
- **Software Development**

Delivering the Software Foundation

Software Developed under ASCR Funding

Programming Models

Active Harmony
ARMCI
ATLAS
Berkeley UPC Compiler
Charm++
Fountain
FT-MPI
Global Arrays
Kepler
MVAPICH
OPEN-MPI
OpenUH
PVM

Development/ Performance Tools

BABEL
Berkeley Lab Checkpoint Restart (BLCR)
Dyninst API
Fast Bit
Goanna
HPCtoolkit
Jumpshot
KOJAK
MPIP
MRNet
Net PIPE
OpenAnalysis
PAPI
ROSE
ScalaTrace
STAT
TAO
TAU
Hpcviewer

Math Libraries

ACTS COLLECTION
ADIC
Hypre
ITAPS Software Suite
LAPACK
Mesquite
MPICH2
OpenAD
OPT++
PETSc
ROMIO
ScaLAPACK
Sparskit-CCA
Trilinos

System Software

Cluster Command & Control
High-Availability OSCAR HA-OSCAR
LWK-Sandia
PVFS
ZeptoOS

Collaboration

enote

Visualization /Data Analytics

BeSTMan
Parallel netCDF
Virtual Data Tool Kit

Miscellaneous

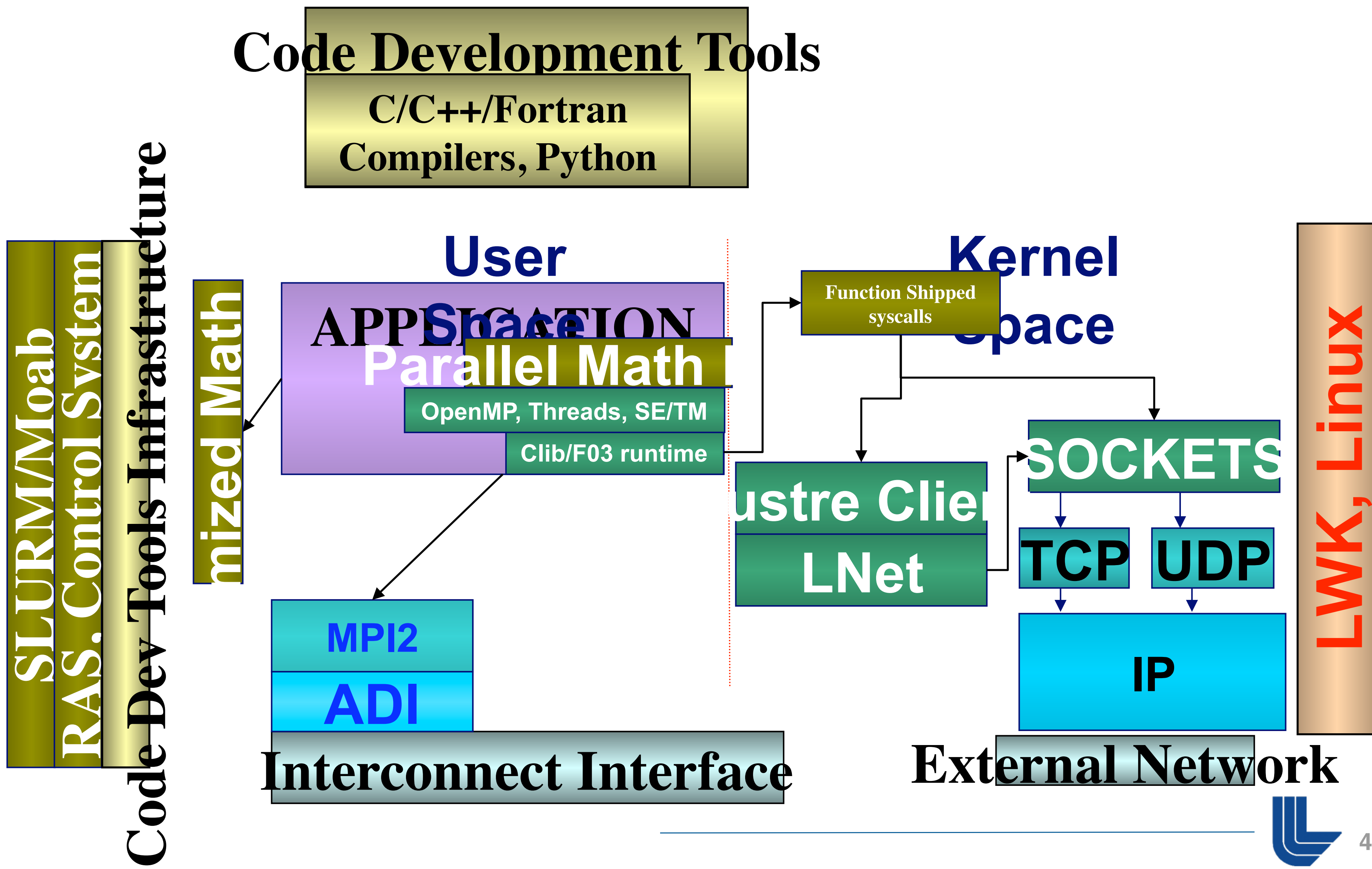
Libmonitor

Blue Waters Computing System Software Issues (collaboration w. IBM)

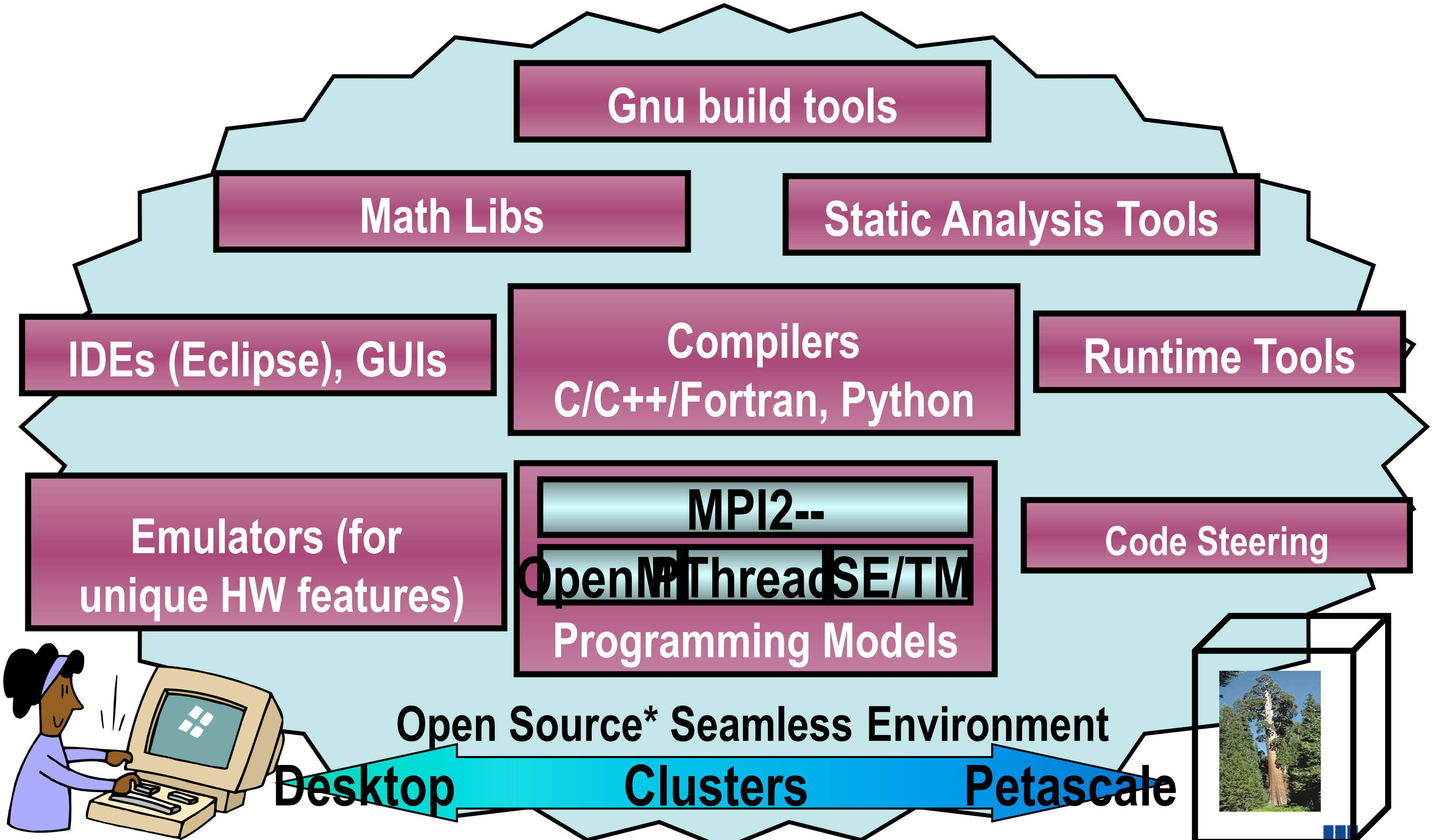
- **System software**
  - Scalable, jitter-free OS (AIX or Linux)
  - Integrated System Console
- **Software development environment and tools**
  - Programming
    - New models: MPI/OpenMP, UPC, CAF, GSM
    - Efficient compilers: C/C++, Fortran, UPC, CAF
    - Scalable debugger
    - Optimized libraries
    - Frameworks (e.g., Cactus)
  - Performance tools
  - Workflow management
- **Reliability**
  - Virtualization

Integrated via Eclipse

Sequoia Distributed Software Stack Targets Familiar Environment for Easy Applications Port



Consistent Software Development Tools for Livermore Model from Desktop and Linux Clusters to Sequoia

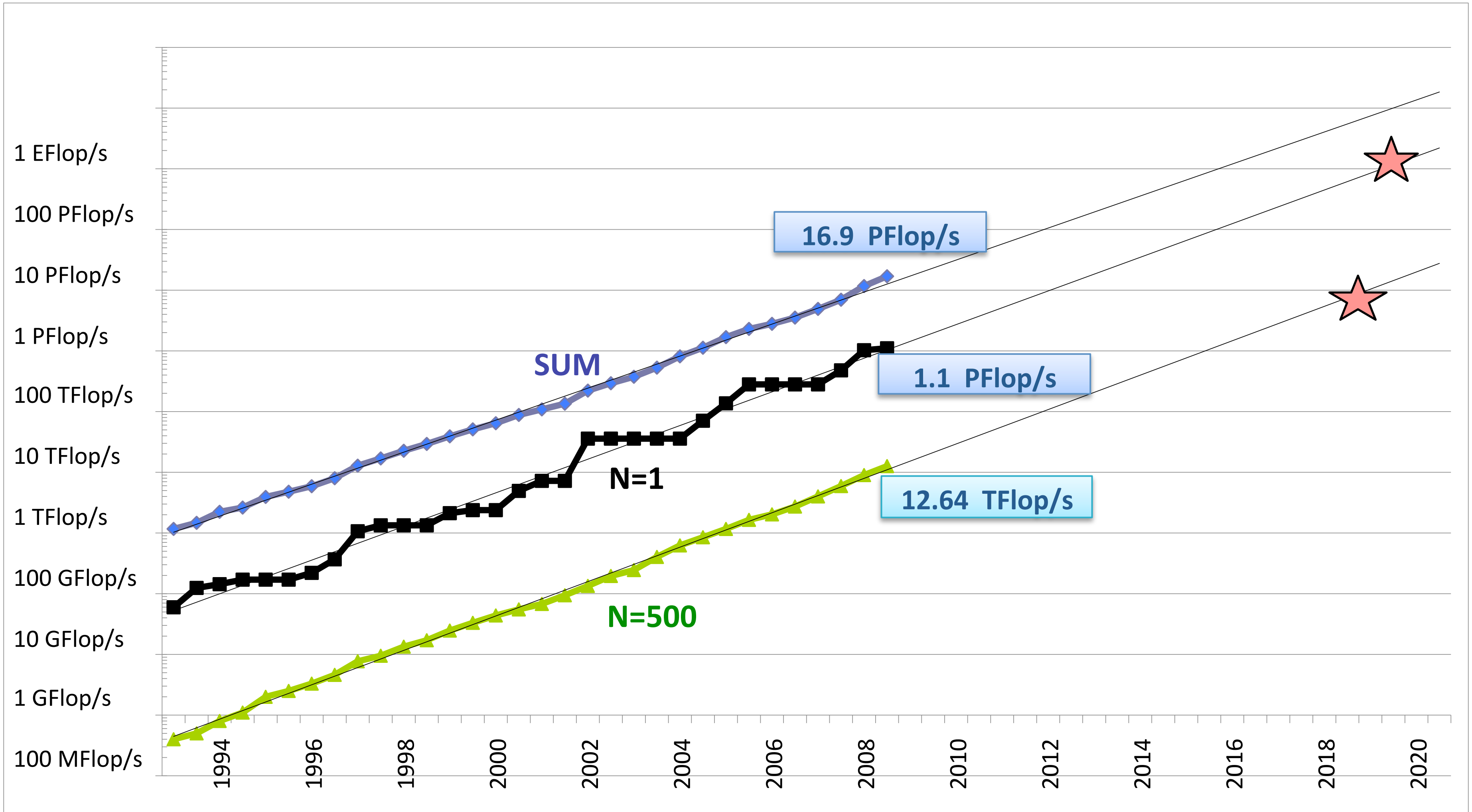


*Vendor, ISV components are negotiable

Observation #1

There is no coherent Petascale software plan across different platforms and different agencies

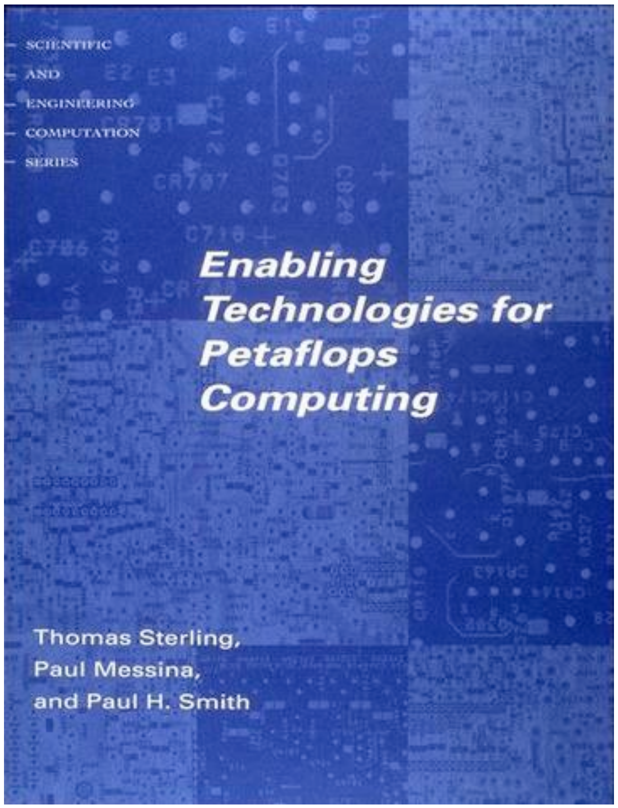
Performance Development Projection



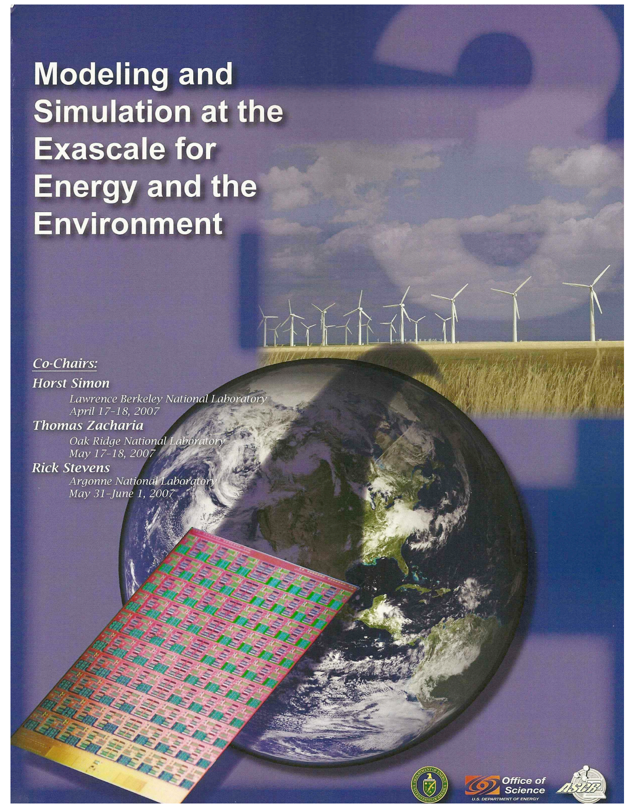
TOP500 SUPERCOMPUTER SITES

Petaflops to Exaflops

1995 “Building a computer 10 times larger than all the networked computing capability in the USA”

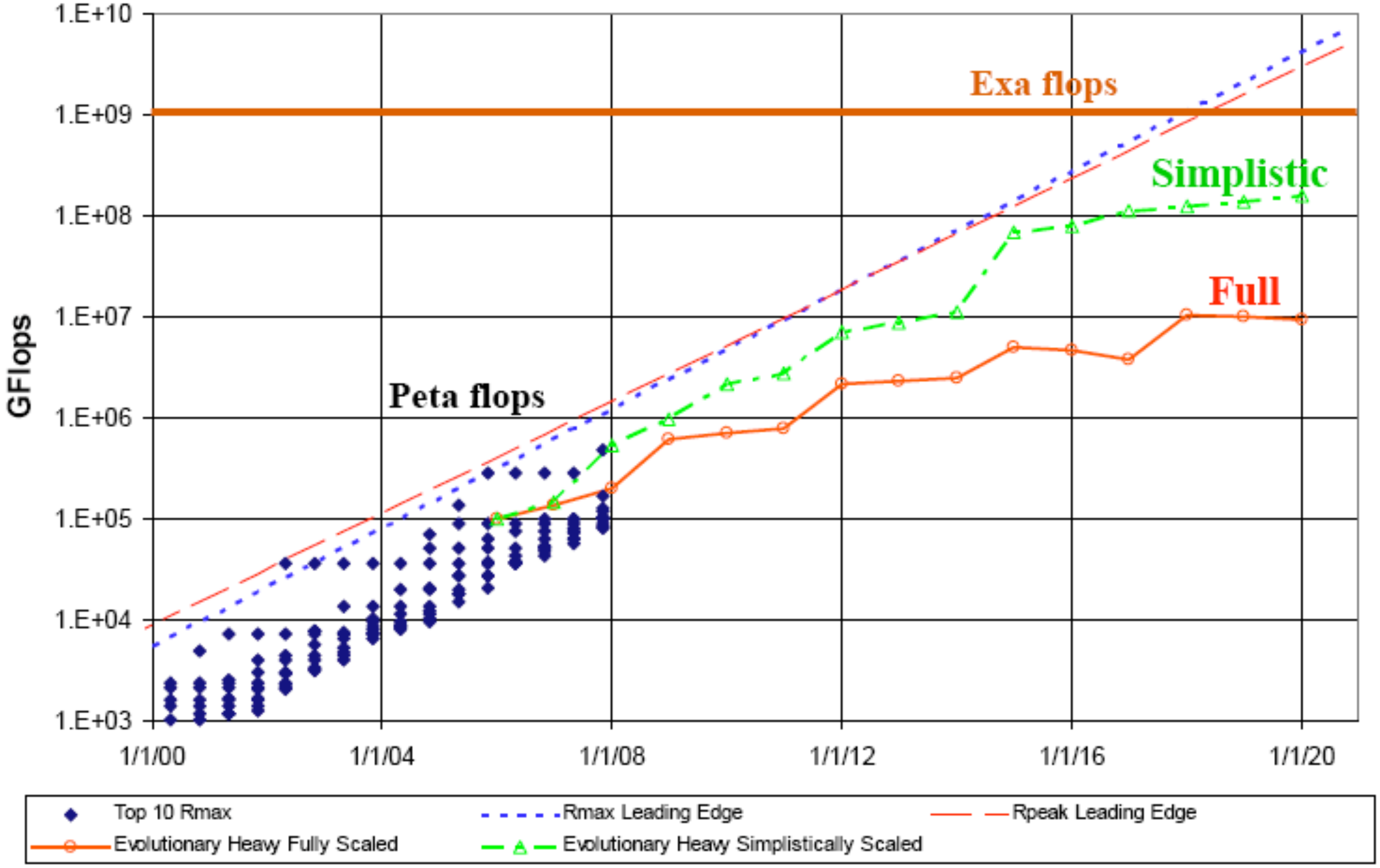


2007 “range of applications that would be materially transformed by the availability of exascale systems”



www.er.doe.gov/ASCR/ProgramDocuments/TownHall.pdf

DARPA Exascale Study:
We won't reach Exaflops with this approach

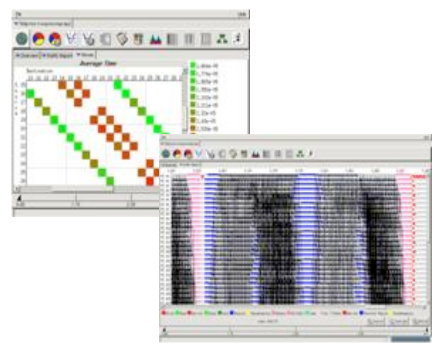


From Peter Kogge, DARPA Exascale Study

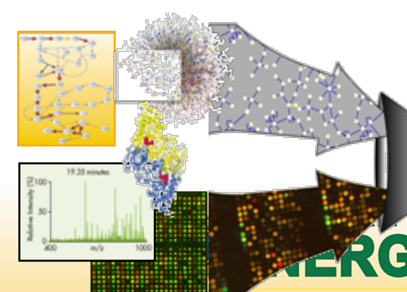
Exascale Townhall: Software – Findings

"Effective use of exascale systems will require fundamental changes in how we develop and validate simulation codes for these systems and how we manage and extract knowledge from the massive amount of data produced."

- **Exascale computer architectures necessitate radical changes to the software used to operate them and the science applications. The change is as disruptive as the shift from vector to distributed memory supercomputers 15 years ago.**
- **Message passing coupled with sequential programming languages will be inadequate for architectures based on many-core chips.**



- **Present code development, correctness, and performance analysis tools can't scale up to millions of threads.**
- **Checkpointing will be inadequate for fault tolerance at the exascale.**
- **Fundamental changes are necessary to manage and extract knowledge from the tsunami of data created by exascale applications.**



Exascale Townhall: Software - Challenges

Improve scientists' and administrators' productivity

- Creation of development and formal verification tools integrated with exascale programming models

Improve the robustness and reliability of the system and the applications.

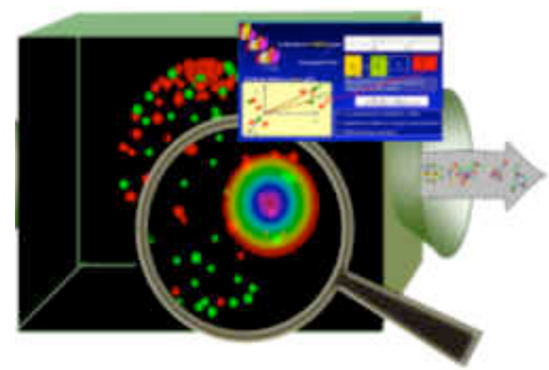
- New fault tolerance paradigms will need to be developed and integrated into both existing and new applications

Integrate knowledge discovery into the entire software life-cycle

- Application development tools, runtime steering, post-analysis, and visualization

Develop new approaches to handling the entire data life-cycle of exascale simulations

- Seamlessly integration into the scientist's workflow
- Automatically capture provenance
- Develop effective formats for storing scientific data



IESP INTERNATIONAL EXASCALE SOFTWARE PROJECT

RTMENT OF ENERGY Office of Science

Observation #2

- **Software environment evolved naturally from Terascale to Petascale**
  - same system architecture
  - only ~10X increase in parallelism
- **Software environment must change fundamentally in the transition from Petascale to Exascale**
  - different node architecture
  - massive parallelism (~1000X increase)



Two important questions about IESP

- **Evolution or revolution?**
- **Program or project?**

… to be discussed at the reception

Robert F. Lucas
Computational Sciences Division
Information Sciences Institute
University of Southern California
rflucas@isi.edu

Musings on the Path Forward to Exascale

In the hey-day of supercomputing, when Cray Research was the darling of Wall Street, scientists and engineers in both the public sector, Universities and National Laboratories, as well as those in industry used the same computer systems. Sometimes, they used the same codes, such as NASTRAN, which was initially developed by NASA GSFC and later distributed World-wide by independent software vendors (ISVs), like today's MSC Software. In other fields, such as nuclear weapons design, for which there is no commercial market, government-funded scientists could still leverage the same rich software ecosystem of operating systems, compilers, numerical libraries, and debuggers as was available to their colleagues in industry.

The advent of distributed memory, message-passing systems dramatically changed the above status quo. The codes that consumed the most supercomputer cycle time were often highly specialized to the Cray architecture, and it was not practical, if even feasible to port them. In Labs and academe, a new generation of capability codes was developed, often from scratch, i.e., designed form the beginning to exploit the new systems. This also required the development of a new software ecosystem with numerical libraries, debuggers, etc. Passing the burden of orchestrating data distribution and communication to the user (i.e., MPI) at least allowed most us to continue to use standard languages and compilers on individual processing nodes.

While they were much slower to do so, industrial users have now also adopted distributed memory systems. More and more of today's mainstream commercial software exploits thread level and even message-passing concurrency, though scaling of these codes is usually quite limited. Thus, an automaker with thousands of CPUs will not launch a handful of large capability computations, designed to explore some novel design, but rather will launch a large ensemble of modest jobs (~32 processors), each evaluating a small perturbation in their design space.

The divergence of public and industrial use of supercomputers had a deleterious impact for all involved. The market for high end systems stopped growing, and many system vendors left the market. Many users found that they could lower the cost of computation over the course of the last decade, but could not increase the scale and fidelity of those same computations [ref. Vince Scarafino, Ford Motor Company].

As we look forward to Exascale, there are reasons to believe that we will face a transformation similar to that experienced in the early 1990s, when distributed memory stopped being a curiosity, and went mainstream. The rate at which users and their tools must expose additional concurrency is actually increasing, and by the dawn of the

Exscale era could exceed 10^9. Meanwhile, the ratio of Bytes to Flops could drop by orders of magnitude as DRAM sees the end of its Moore's Law growth earlier than logic circuits. This in turn will almost certainly lead to a new memory hierarchy as technology like Flash fills the void. Heterogeneous systems like RoadRunner may become prevalent.

An obvious question that arises is how do we learn from our past, and manage this next transformation so that it is not as disruptive as the last one? The thesis of this white paper is that we need to do so in multiple ways. First, we must evolve our systems and software, whenever possible, in a manner that is predictable by users and developers. There is more value in application software today than there is computing systems. There are applications in use today that are forty or more years old (e.g., NASTRAN), and these applications can be expected to grow over the course of the next decade. The developers of these codes must be provided with a path to the future that allows them to incrementally add new features and anticipate changes in computing systems. Note, the transition from scalar to vector circa 1980 was evolutionary for most developers.

Secondly, we must remember that computer systems exist to solve problems for their human users. Thus Exascale systems must be co-designed with the applications that they will ultimately run. Building message passing systems was expedient for the system architects, punting to application developers the hard problems of distributing and coordinating the computation. As the level of concurrency approaches 10^9, this will no longer be feasible. We will not be able to tolerate unnecessary overheads in communication and synchronization, lest Amdahl fractions preclude users from making practical use of such systems. This author believes we should start an Exascale system program with the scientific and engineering challenges it will be expected to solve. In such a design, we must consider existing software to be an important boundary condition.

Finally, there is concern that the reliability of systems will begin to decline as we approach Exascale. The scale of the systems and the number of components involved is increasing. Worse, as VLSI geometries continue to shrink, the long-term reliability of integrated circuits will decrease and they may become increasing vulnerable to transient failures. Unfortunately, in mainstream science and engineering, the programming model has always been that the system is reliable, and simple measures like checkpoint/restart would be adequate for the rare exceptions. I firmly believe that every attempt must be made by computer hardware and system software to continue to isolate software developers and end users from any reduction in component reliability. Otherwise, we will poison the ecosystem and can expect to see fewer and fewer users of capability systems.

.

BSC vision Towards Exascale

Jesús Labarta, Eduard Ayguade and Mateo Valero,
Barcelona Supercomputing Center – Centro Nacional de Supercomputación
(BSC–CNS) and Universitat Politècnica de Catalunya
Nexus II Building, C. Jordi Girona 29, 08034–Barcelona, Spain.
{ jesus.labarta, eduard.ayguade, mateo.valero}@bsc.es.

What is scalability? We need to reach a view of our systems, where looking at them from different distances still lets us have a self similar view, like looking at the earth form the moon, a satellite, a plane, the top a mountain or standing on the ground. We need unified views of our computing systems, where **granularity** is the main difference between the levels we may focus at.

The current experience represents a single snapshot of different techniques at various granularity levels. We use dataflow ideas in out of order processor design, decoupling between logical and physical address space at the virtual memory level, synchronous algorithms at cluster level, … . We should look at all good ideas, developments and practices form the past and apply them in a broad scalable way, where granularity is the only difference between levels.

Power, variance, resource (energy budget, processing power, storage, and communication) sharing and management, and global complexity are important challenges. Memory structure is a key issue, seen both form the point of view of the model offered to the programmer and the actual hardware structure and support mechanisms. Overlap between communication and computation and in general better tolerance to latency is important to avoid the over dimensioning of communication infrastructures that current practice seems to favor.

Facing these challenges, **asynchronism and decoupling** different conceptual levels is the key to tackle a universe huge in scale and characterized by variance. We consider that the programming model is Alexander's sword to break the Gordian knot of multicore and exascale systems. A proper **programming model** is the key interface that will allow the separation at a coarse granularity level between the concerns of users and those of system designers in the same way ISAs did allow such separation and progress in the past. The only issue is that we still have for forge this sword and we will require strong interaction of all levels to do it.

We need programming models that help decouple the way programs are written and executed. At the programmer interface, we should be able to write ideas left to right, top to bottom in a clean and concise way. Runtime should be able to execute them out of order (right to left, top to bottom,…) in the way that the utilization of the resources and thus global efficiency is optimized.

Ideally a single programming model with hierarchical capabilities should cover the whole dynamic range from the single node to the exascale system. It is nevertheless foreseeable that mixed approaches will be used in the near future, with a different model being used for the cluster, node and accelerator/device level. In this situation, an issue that will have strong impact on the programmability of our systems and the final performance is the "compatibility" between the models at the different levels. Different models have/promote different parallelization and synchronization structures. Very often in the past, mixed models gave discouraging results due to a mismatch between the parallelization structure at the coarse and fine grain level. Considering that properly parallelizing an application (and we are really facing Amdahl's law) is a global issue, both programming model designers and application writers need to put special care in ensuring that the interactions between the fine and coarse level result in positive interference.

We consider that a clean specification of what are the inputs/outputs/accesses of a computational block (**task model**) is a proper boundary between a programmer who has a good knowledge of the algorithmic interactions and the execution engine. The runtime should be responsible of the scheduling issues: progressing as fast as possible along the critical path; knowing which functional units (cores) are more appropriate for each task; deciding where to issue task to maximize locality and minimize bandwidth requirements. Mechanisms for the programmer to provide hints and additional information to the runtime will be useful, but not a requirement.

We need to **decouple memory** as a logical address space to name objects from memory as container to keep the values. Matching objects to the actual containers available should be handled dynamically by the run time, in a much more flexible than what is today done. We are used to a single level of such mapping being handled by hardware (caches,…) and we will need to consider a hierarchical approach, where at coarse levels of granularity this functionality is handled by the runtime.

Many of these ideas come from dataflow, yes, and they should be extensively used in our execution engines. We do need syntactical ways to provide a **smooth transition** form current practices to facilitate the adoption of such techniques by the huge community of programmers who are scientist, but not computer scientist.

At the application level we do **need to restructure our codes** to clearly reflect the actual access patterns. Many current applications have accesses to key global structures deeply buried into the call tree. This is not only bad for the future exascaling of the code, it is also bad for today's maintenance and development of new functionalities. We envisage that such application cleaning process will have to be undertaken by application developers in their way to exascaling. This will have to be done while including in the code the asynchrony and means to determine dependences between computations. What would be important is to ensure that this is an **only once effort**, leading to applications that can survive for some decades and can be upgraded and rapidly ported to the foreseeable explosion of hardware platforms. This should be feasible in a **portable,**

modular and incremental way, possibly tuning some low level task description to specific accelerator hardware but leaving the program structure and code unmodified.

At the application level it will also be important to work on new **algorithms that are more asynchronous in nature**. It is hard to imagine programs with tens of millions of threads synchronizing globally at fine granularity that will run efficiently and insensitive to variance or noise. It will be necessary to study where the balance stands in terms of computational complexity of an algorithm and the level of asynchronism that it has.

Load balancing is a key issue to achieve performance at high scale, which is frequently underestimated. We tend to believe that our applications are more balanced than they really are if their actual execution is measured in fine detail. Very often in current practice we blame the communication subsystem when the real cause of the problems comes from load imbalances or serializations. MPI, like a perfect gas, fills whatever space you give it. We should look more at what and how we compute and a bit less at how much time we spend in MPI. **Dynamic load balancing** techniques will have to be used to solve the issue, irrespective of whether it is caused by the application itself, or originates from variance in the devices or system software or from the shared usage of resources. In the same way that having to continuously use force to enforce power is not having real power these dynamic techniques should be always there, but only enforced when needed.

Malleablility of applications is a feature that will be a requirement mostly arising as a requirement of shared utilization of systems and the attempts to optimize the global throughput of systems and quality of service/SLAs. Malleability, as the ability of an application to change its parallel structure (change resources used) is a feature that will have to be enabled/facilitated by programming models, although application developers will have to follow a few methodological guidelines. The same techniques developed for load balancing above will be needed in the runtime to achieve malleability. The only difference is that in this case, the decisions will have to be coordinated with the OS schedulers at the different levels (kernel threads, processes, jobs).

Fault Tolerance will certainly be a relevant issue as it will not be possible to ensure functional operations of all the components of a system for the execution time of applications. The failures may happen at different granularity levels (individual functional units or cores, whole address spaces…) Techniques to tolerate these faults will be needed. Depending on the granularity may be implemented in software or hardware support may be required. Task based programming models as advocated above in conjunction with transactional memory functionalities seem to provide a fair basis to approach the issue. Faults if properly handled, recovered and isolated will result in dynamic availability of resources, thus linking back with the load balance issue described above.

Understanding the performance of our programs will be of great importance. We have the feeling that performance tools and analysis practices are a bit in their infancy. Today we essentially measure some aspects of system performance and report very global aggregates that generally convey little information about the details, and unfortunately, it is in the details where a lot of the performance of these systems will be gained or lost. There is a need (and potential) for much more statistical processing of our data, use of analysis techniques from other areas (i.e. signal processing, clustering) more extensive use of models in order to actually provide insight to the analyst.

At BSC we have been working on the StarSs programming model, which we believe addresses in a clean way many of the above stated considerations. Initial implementations of the run time for SMP, Cell, GPUs are available. It can be integrated with other models at large cluster scale (i.e. MPI) and still propagate to such an outer level many of the benefits of the dataflow execution. Also further implementations of the basic model at coarser granularity levels are being explored. We do believe that ideas from the model can on one side guide and on the other highly benefit form architectural support, especially in the memory subsystem design area. We are also involved in performance tools, job scheduling, applications… We would like to contribute with our vision and ongoing efforts to this holistic Exascale initiative.

Software Challenges for Extreme Scale Computing: Going from Petascale to Exascale Systems

Michael A. Heroux, Sandia National Laboratories

1. Introduction

Preparing applications for a transition from petascale to exascale systems will require a very large investment in several areas of software research and development. The introduction of manycore nodes, the abundance of parallelism, an increase in system faults (including soft errors) and a complicated, multi-component software environment are some of the most challenging issues we face. In this paper we address four topics we believe to be most the challenging issues and therefore the greatest opportunities for making effective next-generation scalable applications.

2. Parallel Programming Transformation

The first and foremost barrier to optimal use of extreme scale computers is the required transformation of parallel programming strategies. There is mounting evidence that optimal parallel applications for scalable manycore computer systems will rely on MPI for inter-node parallelism, but will need to introduce large-volume functional parallelism and SIMD vectorization. Vectorization is the job of the compiler, with a little help from the programmer via pragmas and directives. The real issue is that presently there is no obvious parallel programming model for implementing the middle layer of parallelism. Current standards such as OpenMP, Pthreads and UPC are not designed for manycore nodes. CUDA, RapidMind and related products target manycore but are proprietary. OpenCL is an emerging standard but is not really a user-oriented interface, and will likely not provide optimal performance (e.g., in comparison to CUDA on GPUs).

However, even without an emerging programming model for manycore, there is a vast amount of work required to prepare existing applications for manycore nodes. Two major tasks are (i) reducing bandwidth requirements as much as possible, primarily by introducing the use of mixed precision, storing data in 32-bit arrays wherever possible, and (ii) rewriting low-level kernels as stateless functions with large enough granularity to keep a SIMD core busy, and small enough that there is a large volume of simultaneous function calls to execute.

Application developers can immediately begin refactoring software in anticipation of manycore nodes, but a manycore programming model will need to emerge in the near future.

3. Beyond the Forward Problem

In many areas of science and engineering, solving a single problem with given input conditions, the *forward problem*, is sufficiently challenging, and higher forward problem fidelity is the highest priority for scalable computing. However, as the fidelity of the forward problem becomes sufficiently good, it becomes possible and imperative to study parameter sensitivities, quantify uncertainties and automatically compute an optimal solution over a range of parameter values.

All of these advanced modeling and simulation techniques quickly increase problem size and parallelism—often by orders of magnitude—and large problems can easily exceed the computing capacity of our largest systems. The simplest of these approaches are "black box" in nature and do not require a true peta/exascale system (instead requiring a cluster of tera/petascale systems). However, more advanced methods (often called embedded methods) rely on a tightly coupled aggregation of forward problems and require a true peta/exascale system. The challenge with embedded methods is that they require the transformation of an application into a "subroutine" because embedded methods need to call the forward solve as a function. Most applications were not designed with this mindset, so this transformation will be challenging.

4. A Fault-resilient Application Environment

If hardware fault predictions are accurate, exascale systems will have very high fault rates and will in fact be in a constant state of decay. "All nodes up and running," our current sense of a well-functioning scalable system, will not be feasible. Instead we will always have a portion of the machine that is dead, a portion that is dying and perhaps producing faulty results, another that is coming back to life and a final, hopefully large, portion that is computing fast and accurate results.

Our current hardware and software environments are not well prepared for this kind of "stable" system. In fact, the only reliable, portable resilience mechanism we have is checkpoint-restart. Although there have been many research efforts in fault tolerance, much of this work has been focused on a single layer in the hardware and software stack, without sufficient consideration of the whole set of requirements. One of the biggest needs we have in resilient computing research is an increased effort to include the full vertical scope of the software and hardware stack into our design discussions. Furthermore we need a full-featured environment to probe the system, make decisions based on system state and recover from system faults, both hard and soft. Without a dramatic improvement in this environment, we face the very real risk that application developers will reject exascale systems in favor of smaller, more reliable systems that provide a better overall throughput.

Regardless of how unreliable a system is, from an application developer's perspective there has to be some way to perform reliable computations. This does not mean that every computation must be reliable, but that certain, perhaps higher cost, computations and their input and resulting data are highly reliable. Without this kind of capability, it becomes extremely difficult to provide any kind of verifiable result. An application needs the ability to declare certain ranges of data as highly reliable. Furthermore, it needs to know that certain computations have completed correctly or, if not, have the ability to react to faulty or interrupted computations. If the runtime environment can provide these two features, we can develop algorithms that will be reliable on exascale systems.

5. Hierarchical Software Engineering and Development

The CSE software community, by most accounts, has been slow to adopt formal software engineering practices. Although a lot of high quality software has been developed without formal practices, the demands of collaborative development, multi-code environments and large collective teams require more attention to the benefits that formal practices can provide.

Typically, single-physics CSE application and library software efforts naturally involve a small team of researchers who work closely with each other on a daily basis. However, advanced CSE projects require a coordinated effort of dozens or more researchers who, although contributing to a larger effort, continue to work in small teams on their portion of the project. The Trilinos project, as one example of a "project of projects," has used a kind of "federalist" approach to addressing these competing realities. We have formally defined a "package" to be a collection of related functionality developed by a small team with certain rights and responsibilities in the larger Trilinos framework.

This basic approach has enabled a great deal of local autonomy in decision-making, allowing us to tolerate and appreciate a variety software research and development styles, and team cultures. We can handle modest redundancy in software functionality and adapt to change in many ways. At the same time, this approach also provides a global interaction that promotes a variety of desirable outcomes: (i) cross-fertilization of ideas, techniques and tools across package teams, (ii) adoption of "best practices" from one package across other packages, (iii) fostering of trust among disparate groups (iv) software modularity that is naturally enforce by package and team boundaries and (v) well-defined interfaces between packages for interoperability.

One important factor that improves the effectiveness of the Trilinos architecture is the constant focus on improving software engineering practices and processes. The philosophy we promote is that we spend time on improving software engineering so that we can spend less time on software development and maintenance and more time on science and engineering. This emphasis has two major impacts on our efforts: (i) better software engineering in the project makes for better software so that package teams are willing to use each other's software and (ii) discussions of incompatibilities in practices and processes across packages can focus on the goal of determining best practices and not decay into expressions of personal preference that can be contentious and counter-productive.

The net result of this approach to software research and development is a large and growing collection of inter-related tools where Trilinos as a whole has an identity but, even more importantly, each package has its own identity within its community of interest. It is worth noting that this kind of approach is also operative within the TOPS-2 SciDAC project. The climate community uses the CCSM in a similar way, but we are unfamiliar with its internal dynamics.

We believe an international effort to coordinate the efforts of many groups can benefit from the kind of model the Trilinos project is using. This type of approach will allow individual teams to simultaneously continue with their current efforts, practices and culture while at the same time start contributing to a larger whole.

6. Conclusion

There are many challenges facing application development in the transition from petascale to exascale. We believe the four issues above have the highest priority and, if addressed, will greatly improve exascale computing capabilities.

Software and Exascale Computing

Bill Camp
Intel Corporation

Disclaimer: The views expressed herein are solely those of the author as a member of the scientific community and do not claim to represent those of Intel Corporation in any way.

There is really only one software issue facing us in developing a robust exascale computational economy: scalability. Because of scalability concerns, virtually none of today's applications is ready for exa-ops performance. We have increased system-level computing power about a factor of 1000 every decade for several decades now; and we have had to grow systems to do so. Since Moore's Law is increasing device capability at less than half that amount per decade, we have inexorably invested more money in ever larger systems. In 1997, the largest systems in the world achieved terascale performance with fewer than 10,000 processors; and none of them were multi-core. In 2007, the largest systems in the world achieved petascale performance but had more than 10 times as many processors in doing so. We anticipate that exascale systems will have around a million processors and that those processors will be MPPs themselves—having O(1000) cores. Thus an exa-ops system will have around a billion virtual or real cores.

Scalability faces us in numerous disguises:

Scalability of

1. programmability, debug-ability, and optimization
2. interpretability
3. reliability
4. performance
5. the energy cost of software

Programmability, debug-ability, and optimization:

I have little to say about programmability except to note that there is no single magic-bullet solution to this issue. As noted above, for reasons finding their roots in the physics of CMOS semiconductors, any exascale application in the 2018—20 timeframe will involve $O(10^9)$ threads. No human being can program, debug or optimize directly this many threads. At the same time, no new programming paradigms are credible at this point: it looks like we will use a combination of distributed memory methods (gets & puts, message passing, and incoherent global-address space methods) across the ensemble of processors possibly combined with shared memory methods on-processor. High-level languages may allow us to express that parallelism more effectively—or they may continue to just get in the way of successful parallelism. On the positive side of the ledger, I am convinced that for data-parallel applications, we can use the same kind of automation that has proven successful in areas like geometry and meshing: in data parallel applications, create primitives and extend, replicate, map them onto complex graphical representations to cover the domain of interest. In task-parallel applications, we can use self-similar and hierachical approaches familiar from statistical physics: utilize self-organization combined with automated hierarchy of control to manage complex work queues.

Interpretability:

I have even less to say about interpretability. We are already facing a gap between our ability to generate data and our ability to make sense out of it. Just as terascale applications led ultimately to petabytes of data and petascale applications are starting to generate exabytes of data, exascale applications will generate yoddabytes of data. We will struggle to make interpretation of that much data easy or even doable. Visualization is an obvious but less than desirable and incomplete solution. The human visual cortex can deal with about a gigabyte at a time. So, we will have $O(10^{12})$ times as much data as we can visualize effectively in a single image. And that assumes that we find a way to deal with the storage and computing problems implied by such an approach. Effective interpretation of such data sets will require advances in cognitive software to turn data into information and information into knowledge and knowledge into insight.

Reliability:

This is an area that properly speaking spans the worlds of hardware and software. Until now, we have separated software reliability from hardware reliability. The former has been the domain of software architecture, software engineering, and mathematics; while the latter has been an integral (some would say not integral enough) part of system architecture and design. At the exascale we can no longer afford that separation. Hardware designers are struggling with how to make systems a thousand times more reliable per bit-operation to keep us at the same level we are at in today's best systems. This is compounded by the fact that energy concerns are driving us inevitably to sub-threshold logic. At the same time, the only reason to do exascale computing is to address ever more complex issues. This will require ever more complex software. Software complexity is the number one cause of unreliability in computation today—well exceeding even hardware's worst efforts! So, we can anticipate that without a radical change in how we handle software resiliency and reliability, we are going to be worse off—much worse off than we are today. One idea is that we build a much higher level of local check-pointing capability into our software and hardware systems. For example, using raided non-volatile memory, we could checkpoint state very often by moving copies of needed application state to nearest neighbor nodes in the system several times a minute perhaps several times a second. Since non-volatile memory is only drawing power when it is in use, this would have minimal energy implications. Dynamically, we can pretty effectively protect correctness of state but correctness of logic poses special challenges. State can be protected at about a 10% energy overhead. Logic correctness requires more invasive approaches with some degree of redundancy that could well exceed the 10% overheads that we have learned to tolerate for state—current R&D focuses on residue checking and redundant multi-threading. However, these have significant energy overheads; and, due to the energy issues discussed below, we are going to be more limited than we should like in protecting logic paths. This will require some degree of cooperation between software and hardware—perhaps identifying at compile time certain critical regions which need stronger correctness guarantees. In any case a serious problem that I believe must be overcome is posed by the brittleness of today's algorithms and applications. We are already generating terabytes to petabytes of new state per second. At exascale we will be generating exabytes of state each second; and a single wrong bit can vitiate the entire calculation. For many scientific calculations we should be able to gracefully tolerate amny kinds of bit errors, indeed the loss of many kinds of local resources. For example, in simulating materials, loss of a processor should not cause inherent failure of the simulation. Think of real materials that are full of defects and faults. We know that we will get for most macroscopic and many microscopic properties the same result for quite different distributions of those defects. Why should we not be able to take advantage of that in our simulations?

Performance:

To a large extent, performance is bounded by the product of the effective speed of the local processor and the communications efficiency of the interconnect fabric. The speed of the processor is largely determined by the ability to issue and retire instructions which in turn is governed by pipeline efficiency and memory system overhead, latency, and bandwidth. Normally, we are used to thinking that communications efficiency is dominant at scale; and that probably remains true. However, due to energy concerns, the efficiency of the processor itself bears special watching: we clearly cannot afford the powerful out-of-order cores supporting both prefetch and speculative execution that characterize today's processors.

From a software point of view, scalability is limited by load imbalance, algorithmic serial complexity and parallel efficiency, communications overhead due to the communications hardware, but also overhead due to the communications software architecture and implementation. One should not dismiss the effect of the programming paradigm and its hardware implementation. If we insist on a cache-coherent shared memory programming environment, we should understand the cost of implementing such an environment in terms of coherency traffic, synchronization overhead, and memory sub-system conflicts.

Load imbalance will arise from vagaries of the applications but also will occur due to loss of self-synchronization caused by the run-time system, the resource manager, and the operating system. Communications overhead must be diminished by aggressive overlap of communications and computation. At 1 billion threads, if we wish to achieve significant parallel efficiency, we need to keep serial fraction and communications overhead extremely small. If we assume that communications overhead is negligible, Amdahl's Law tells us that the serial fraction must be much less than 10^{-9}. For many weak-scaling problems this may well be achievable. To make sure that communications overhead is also negligible, we must have $\alpha . \omega_{co}$ be much less than unity, where α is the ratio of computational speed to communications speed and ω_{co} is the ratio of non-overlapped communications workload in bytes to computational workload in flops. α is determined by the architecture and is limited by cost and especially by physics. ω_{co} is determined by the computational problem, the code architecture and the algorithmic approach. Unfortunately, physics will prevent us from achieving the kind of balance we wish for in α. We are left to compensate for that in software.

William J. Camp, Ph.D.
Chief Supercomputing Architect
Supercomputing Architecture and Planning
Intel Corporation
505 301 5598
william.j.camp@intel.com

Application Analysis and Porting in the PRACE Project

Peter Michielse
Netherlands National Computing Facilities Foundation (NCF)
The Netherlands
Email: michielse@nwo.nl

1 Introduction

PRACE, the Partnership for Advanced Computing in Europe[1], aims to set up a European HPC ecosystem to facilitate scientific research, with sustainable access to Tier-0 HPC systems, including system management and extensive application support. In order to become successful PRACE will need to understand (among others) the software requirements for future Petaflop/s systems. PRACE has identified the key scientific and technical categories of applications, through a survey of most major European HPC systems and the applications that exploit these, carried out in early 2008. Final goals in this part of the PRACE project are the construction of a benchmark suite, to be used both within the current PRACE project and beyond, when actual Tier-0 systems will be purchased. Other goals include insight in the optimisation and scalability issues with the selected applications, and applicability of synthetic benchmarks and performance analysis tools.

2 Methodology within PRACE

Each benchmark application will be worked on under the responsibility of a so-called Benchmark Code Owner (BCO). The BCO is a person who in most cases belongs to the staff of one of the PRACE partners. The BCO will steer the actual porting, petascaling and optimisation, such that the benchmark code will run on each of the designated hardware architectures for the underlying application. This includes the scheduling of work among the contributing PRACE partners to the benchmark code, and communication with the application owners on all aspects of the application: source code, dataset, output, run scripts, etc. In particular, actual results will first be communicated to the application owner, and through the public status of the deliverable report also to hardware or software vendors, and the rest of the HPC community.

As said, the BCO and his or her coworkers are not only responsible for porting the code to the actual platforms, but also for optimisation and scaling efforts. At this point in time in the PRACE project, porting has been done, and initial proposals and estimates of effort with respect to optimisation and scalability have been formulated by the BCOs.

3 Application Porting to Prototypes

PRACE conducted several surveys among both users of the top national HPC facilities in the PRACE countries, as well as among system administrators of these facilities, in order to establish a representative set of application areas and individual applications. These cover currently the most relevant usage of the national systems in Europe. As a result a list of core applications and a list of possible extensions was created. These are contained in tables 1 and 2. As many applications as possible of the core list should be worked upon in the PRACE project, both to serve in a benchmark suite and to investigate optimisation and scalability aspects.

[1] PRACE has been funded in part by the European Community under INFRA-2007-2.2.2.1 - Preparatory phase for 'Computer and Data Treatment' research infrastructures in the 2006 ESFRI Roadmap under Grant No INFSO-RI-211528. Website: www.prace-project.eu.

Application name	Application area
QCD	Particle physics
VASP	Computational chemistry, condensed matter physics
NAMD	Computational chemistry, life sciences
CPMD	Computational chemistry, condensed matter physics
Code_Saturne	Computational fluid dynamics
GADGET	Astronomy and cosmology
TORB	Plasma physics
ECHAM5	Atmospheric modelling
NEMO	Ocean modelling

Table 1: The proposed list of core applications.

Application name	Application area
AVBP	Computational fluid dynamics
CP2K	Computational chemistry, condensed matter physics
GROMACS	Computational chemistry
HELIUM	Computational physics
SMMP	Life sciences
TRIPOLI4	Computational engineering
PEPC	Plasma physics
RAMSES	Astronomy and cosmology
CACTUS	Astronomy and cosmology
NS3D	Computational fluid dynamics

Table 2: Possible extensions to the core list of applications.

Another consideration has been the actual choice of promising architectures, to be assessed in the PRACE project. For the work on applications, this set of architectures (which are production or near-production systems) has been identified by PRACE in May 2008, and deployed as prototype systems to different partner sites (see table 3). Also, for each of the applications, we have selected BCOs who combine knowledge of the particular application, expertise with certain hardware platforms and access to prototype architectures. For most applications, both from the core list as well from the extended list, this has been successful. Contributors to a benchmark code typically qualify if they satisfy at least one, and preferably two or even three of these aspects.

Architecture type	Actual system	Location
MPP-BG	IBM BlueGene/P	FZJ, Germany
MPP-Cray	Cray XT5	CSC, Finland
SMP-FatNode-pwr6	IBM p575 Power6	NCF/SARA, Netherlands
SMP-ThinNode-x86	Bull – Intel Xeon/Nehalem cluster	FZJ, Germany and CEA, France
SMP-ThinNode+Vector	NEC SX-9 + x86 …	HLRS, Germany
SMP-FatNode+Cell	IBM Power6 with Cell	BSC, Spain

Table 3: Actual prototype architectures in PRACE.

Table 4 shows that all applications from the core list are usable as benchmark codes, on at least 3 target prototype architectures, complemented with 3 applications from the non-core list: CP2K, GROMACS and NS3D. These are the first 12 rows of table 4. SMMP, RAMSES and CACTUS have disappeared from the extended list, as it turned out to be that there was no PRACE partner that could volunteer as BCO. Instead, GPAW (computational chemistry), ALYA (computational mechanics and fluid dynamics), SIESTA (computational chemistry, molecular dynamics) and BSIT (computational geophysics) have joined the application set, mainly to make sure that enough coverage of the SMP-FatNode+Cell platform could be guaranteed. An additional advantage of this is that two other application areas are introduced: computational mechanics and computational geophysics. Each BCO and its contributors have started the work on the benchmark codes and hardware architectures.

Table 4 also shows the current porting status of the applications to the prototype architectures. Green colors denote successful porting, yellow means that porting is in progress, and orange means that porting has not started yet or stopped for the moment because of practical reasons (mostly lack of human resources to do the work).

Application	MPP-BG	MPP-Cray	SMP-TN-x86	SMP-FN-pwr6	SMP-FN+Cell	SMP-TN+vector
QCD	Done	Done		Done		
VASP	Done			Done	Stopped	Yet to start
NAMD	Done	Done		Done	Yet to start	
CPMD	Done			Done	Done	Yet to start
Code_Saturne	Done	Done		Done	Stopped	Done
GADGET	Done		Done	Done		
TORB	Done			Done	Yet to start	
ECHAM5	Stopped	Done	In progress	Done		Yet to start
NEMO	Done	Done		Done		In progress
CP2K	Done	Done		Done		
GROMACS	Done	Done		Done		
NS3D		Yet to start	Done	Yet to start		Done
AVBP	Yet to start		Done	Done		
HELIUM	In progress	Done		Done		
TRIPOLI_4	Yet to start		Done			
PEPC	Done	Done		Done		
GPAW	Done	Done		Done		
ALYA					Done	
SIESTA					Done	
BSIT					Done	

Table 4: Summary on porting efforts for benchmark codes and prototype architectures.

4 Scalability expectations

Apart from porting efforts to the prototype architectures, initial insight in the potential for scaling to petascale systems (and single-CPU optimization) has been obtained. Table 5[2] contains the scalability potential of each of the benchmark codes, including an estimate on the amount of effort in person months (PM). We have defined scalability to be in the range none via low, medium to high and have assumed one core to deliver a minimum of 10 GFlop/s peak performance. The color codes mean:

None (red): No speed-up above 2500 cores;
Low (orange): Speed-up obtained up to 5000 cores;
Medium (yellow): Speed-up obtained up to 10000 cores;
High (green): Speed-up obtained for more than 100000 cores.

[2] Not all cells in table 5 have been filled yet, as initial analysis after porting is currently work in progress.

Speed-up at a certain number of cores is defined as still improving execution time when comparing the execution time on that number of cores to the execution time on half the number of cores.

From table 5, the following initial observations can be made:

- Within the set of computational chemistry codes (VASP, NAMD, CPMD, CP2K, GROMACS, GPAW) the potential varies from low to high. At first sight, this may seem surprising, as they all cover broadly the same application area, although individual codes may use different approaches. It will make sense to investigate how low scaling codes may benefit from algorithms and implementations used in highly scalable codes;
- The amount of effort estimated to improve scalability to medium or high seems to be reasonable: on average around 4 to 5 person months. This will be carried forward in remaining PRACE work.

Benchmark code	Expected scalability	Estimated effort	Comments and areas of attention
QCD	high	0-1 person months	
VASP	high		Depends on FFT and BLAS implementations
NAMD	medium-high	8-10 person months	Investigate master-slave (3 pm), investigate shared memory (7 pm)
CPMD	high	2 person months	Well parallelised already, some tuning needed
Code_Saturne	medium	3 person months	Preprocessing stage and IO
GADGET	medium-high	2 person months	Investigate potential OpenMP constructs and MPI implementation
TORB	high	3-5 person months	Adapt code internals (up to now 999 processes is max.)
ECHAM5	low-medium	2-8 person months	OpenMP optimisation, data output mechanism
NEMO	low	3 person months	Domain decomposition load imbalance, solver implementation, MPI
CP2K	low	5 person months	Load imbalance needs to be solved
GROMACS	medium	8 person months	Optimise communication patterns
NS3D	low-medium	1-6 person months	Very platform dependent - MPI AlltoAll implementation
AVBP	medium-high	2 person months	Focus on MPI implementation (AllReduce area)
HELIUM	medium	3-4 person months	Focus on MPI implementation (synchronisation constructs)
TRIPOLI_4	high	6 person months	Independent particles, Monte-Carlo approach, IO to be modified
PEPC	high	1 person month	Data structure to be investigated
GPAW	medium-high	3-6 person months	Implement SCALAPACK usage, parallelise over electronic states
ALYA	medium-high	2 person months	Explicit solver ok, implicit solver requires effort, IO to be modified
SIESTA	medium	2-3 person months	Focus on MPI implementation
BSIT	high	1 person month	Embarassingly parallel, need to consider queue management system

Table 5: Expected scalability potential and estimated effort for benchmark codes.

5 Future Work in PRACE, Relation to IESP and Acknowledgements

As has been mentioned before, porting the applications to the target prototype architectures is work-in-progress. Already a significant part of the sparse matrix has been filled. This work will continue to complete the sparse matrix on applications and prototype architectures.

Another aspect is the fact that already ported applications will enter the stadium of petascaling and optimisation. BCOs will remain responsible for the coordination of optimisation and petascaling aspects.

With respect to the future final benchmark suite for PRACE, there is the issue of usage and licensing of the application codes. This will need to be resolved with the code developers.

With respect to IESP, it seems to make sense to exchange experience and progress on many of the applications, since these are used globally and possibly already improved by US and/or Japanese efforts. Further, alignment of the efforts in PRACE on application scalability with efforts in the USA and Japan, maybe including software developers and hardware vendors, is important.

This white paper is based on the PRACE project's deliverable "Report on available Performance Analysis and Benchmark Tools, Representative Benchmark", dated November 28, 2008. Many people from the project partners have contributed to this public document.

The Application Perspective - Seeking Productivity *and* Performance -

David Barkai

Abstract—In this note we propose two projects: (1) Creating a hierarchical programming model from current models, and (2) Extracting application primitives from the "13 dwarfs". The first topic addresses the need for a unified and manageable framework for very large scale concurrent execution. This is the productivity part - less complexity will drive better mapping of algorithms to architecture; which will also contributes to better performance. The second topic focuses mostly on the processor and the node with the aim of laying the groundwork for software and silicon optimized kernels. While it is understood that applications primitives are outside the scope of IESP, the motivation for introducing it here is that it is a companion issue and that increasing the efficiency of each processor provides high return for science - at all levels of system size.

Index Terms—programming model, manycore, multicore, clusters, applications, HPC, application primitives

1 Setting the Stage

While the "moonshot" goal in front of us is preparing for systems with peak exaflops, we must not lose sight of the fact the all this is done so science can accomplish more through computations. To this end it is best to take the application perspective, and look for ways to help the scientist or application developer get more out of a given very large system. In this note we suggest to take on the two "P's" - Productivity and Performance (leaving out the third "P" - for Power; though with higher efficiency, another way of saying 'performance', a given computation gets done as fast on a smaller system - and consumes less power).

There is a fortunate synergy now between the need to address programmability on petascale and exascale systems and these three drivers that are now central to the future of high-performance computing (HPC):

- Almost universal adoption of clusters as a 'standard' architecture.
- Manycore processor chips in our future.
- Emergence of heterogeneous computing on or near the processor chip.

The synergy derives from the fact that a standard model that fits the above also suggests a hierarchical view of the system; a view that offers hope for a more manageable approach to dealing with the very high level of concurrency, of order $10^7 - 10^8$, required for a full use of an exascale system in circa 2018.

The ideas presented here are also influenced by the work commonly recognized now as the "view from Berkeley" [1], both with regard to extracting a cohesive programming model and in providing a framework for addressing performance through a set of application primitives.

2 The case for a consistent and layered model

The advent of multicore in all of our platforms presents an opportunity, and motivation, to take a fresh look at our programming model. Looking ahead we have a 3-layer architecture from the user's perspective: the chip - with multiple cores, caches, and, potentially, attached accelerators; the node - multiple processor chips sharing memory; and the system of nodes governed by its distributed memory.

Today we have, essentially, two approaches to parallelizing applications: one for shared-memory systems (OpenMP, for example), the other for distributed memory systems (where MPI is the most popular tool). Multicore on the chip adds another layer, but also impacts the application's choice of algorithm in that the way to increase the performance from one generation to the next is only through finer parallelism as the number of cores on the chip increases, whence preference for algorithms that scale better.

The time is right for a community-wide initiative that will include the application writers, the software providers, and the hardware vendors, with the goal to define a programming model that will be integrated, consistent, and seamless across the three architectural layers, scalable from the node to the petascale and beyond, and allow for application driven expression of concurrency that will extend to dataflow and multitasking, as well as parallel computations.

The discussion is framed with a strong emphasis on the application's perspective, as we believe this will lead the application designer taking more responsibility to map the implementation to the system, resulting in higher productivity, and allowing the system and tools

- *David Barkai is with Intel Corporation, HPC division of the Digital Enterprise Group, Hillsboro, Oregon 97124*
email: david.barkai@intel.com

Revised April 17, 2009

software to do a better job in mapping the hardware. In short, we will be closer to a desired balanced between scientists' productivity and a reasonable performance relative to theoretical peak.

The desired programming model should comprehend partitioning details at a finer level than just assigning processes and threads to cores. It should allow visibility to on-chip or socket-attached interactions. The convergence to a single architecture makes it a good time for the HPC community to take a fresh look at the programming model when designing new implementations of numerical and data-intensive applications. A typical cluster is made up of high-volume off-the-shelf components for processors, memory, boards, interconnect, storage, file systems, etc. This is not central to the discussion here, but for the fact that it provides greater motivation for a 'standard' programming model.

There are two other challenges that large system users have been struggling with and that have not been resolved yet:

- Scaling of applications effectively as they increase in complexity, use higher resolution with larger datasets, and run on an ever-increasing number of processors and cores is, so far, a rare occurrence.
- Productivity - both in terms of the programmer's time, and in terms of output from the compute system is still a panacea.

A holistic, integrated and consistent programming model, constructed and presented from the application writer's perspective might help us move forward with regard to the two challenges above.

3 What might the model look like

This is an abbreviated version of a longer discussion, and, therefore, statements may seem too blunt. My apologies to the reader.

Discussions of programming models almost always turns to languages for expressing parallelism and tools to support parallel programming. We are skeptical that any new language will gain a wide acceptance, and believe the best course of action is to build on the tools that current applications are most invested in. That would be MPI used by Fortran and C/C++.

The need for hierarchical model, to better map the application to the underlying architecture and for better manageability of concurrency, led to various experimentations in "hybrid" implementations - combining MPI with OpenMP or other shared memory schemes. These met with varying degrees of success (see [2], [3], [4], [5]). It is stipulated that the use of OpenMP would not have been required if we had 'layered-MPI' to define such a hierarchy to help manage the decomposition of the application.

A layered model is also necessary in order to have any hope of managing the level of concurrency that will be in the 100's of millions in the future exascale system.

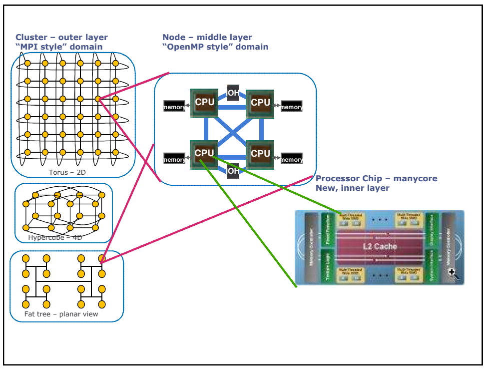


Fig. 1. Hierarchical view of the model

That said, we propose considering a hierarchical model (conceptually drawn in Figure 1), that can be built upon the following guiding principles:

- **Do no harm.** The expression of parallelism according to a new or modified model should not invalidate the huge investment put into existing codes. This principle forces us to look at extensions to, or evolution of, MPI. Given the much lesser use of shared memory models it seems more natural to build an integrated model from MPI.
- **Balance productivity and performance** - as expressed by the Berkeley team [1]. For productivity the model is to present the application view, be expressed in terms comprehended by a high level language and in terms relevant to the scientist and engineer. Let the compiler system (see below) deal with the details - which also vary from one system to another. And for performance's sake give the programmer the tools to associate computations with data, and to specify flow and communication patterns.
- **The application writer knows best** about how the application works and there will be no automatic parallelization any time soon. This has two important implications: (1) The programming model has to have 'hooks' into all the architectural layers and components. (2) The application writer can do a better job partitioning the data and computation than the compiler or middleware. Let the tools be there to offer the help the system software will need.
- **Integrated, layered model.** It would have a set of one or more MPI ranks per node, each may be split into a set of MPI processes, preferably optimized for shared memory, and allowing for each of those to further split into a set of 'fibers' to be executed on the same socket. It is this last, lowest, layer that can be used to interact with special-function units or an attached accelerator.
- **Extensible.** For large systems it may well be useful to allow for some kind of system-level partitioning, in addition to the layers described above. This will divide the highest level into regions of MPI environments working in tandem.

- **Coherency.** The implementation of the model has to be adaptable to various degrees and regimes of coherency. These may be dictated by the system in use, or be a choice to be managed by the user.
- **Robust runtime compiler system.** When the system is a cluster, compilers and runtime libraries that are local-node-aware are not optimum. A complement to any program such as the one outlined here has to drive a considerably more runtime-robust compiler system. A system that will pick up the allocated resources (the cluster or a part of it) and execute to it. This may include, for example, MPI operations that might be presented as directives or pragmas. This will allow skipping them when the job runs within a single node.

The benefits of the vision expressed above are fairly obvious: Common and comprehensive basis for applications design. Potential, and expected, higher performance due to the integration of the support for distributed, shared, and on-chip operations.

The next two sections deal with aspects of performance at the node level, likely to be dealt with in other forums.

4 Application primitives - key to performance

The topic raised here is not specific to exascale systems, but very relevant to scaling and delivered performance for real applications. We will have processor chips with billions of transistors, and looking out towards the 2018 timeframe we can ask how best to use them.

The model we propose to follow is that of signal and image processing. Compact application primitives were identified such that great performance improvement was achieved with a combination of special function silicon and libraries. Despite the greater complexity, diversity, and dependence on bandwidth and latency of data access, can such methods not be applied to HPC?

At the very least, this is worth investigating. A starting point can be the first seven of the "13 dwarfs" taxonomy defined in the "View from Berkeley" [1], as they are the ones corresponding to numerical simulations. To remind the reader, these seven are dense and sparse linear algebra, spectral and N-body methods, structured and unstructured grids, and Monte Carlo. The problem is that the broad brush definition of the application categories is not actionable as it stands. To be able to act on the taxonomy it would be most useful to identify:

1) The algorithms that are the most important (sparse or N-body, for example, may employ a number of different algorithms and methods).
2) The relative weight of the category/algorithm within the general (high end?) scientific workload.

Setting aside the tasks above, for now, we can assess the problem with another source. The NAS Parallel Benchmarks (NPB) [6] are composed of several common computational procedures. They are sure to feature in several of 13-dwarfs categories. A nice feature of NPB is that it reports the MOPS (millions of operations per second) score, which for the numerical tests we discuss here is, essentially, the rate of floating point calculations. This allows us to measure the "efficiency" of the benchmarks compared to the ideal case where all data access can be hidden or overlapped. To make a point five are chosen: Multigrid (MG), Conjugate Gradient (CG), FT (FFT), LU (Lower-Upper decomposition), and BT (Block Tridiagonal). These were run, using the NPB 3.3 version, on 8 cores (a 2-socket node) of Intel's recently launched microprocessor, which has far superior memory bandwidth compared to previous generations of x86 architecture. Even with these very competitive times the calculated efficiency, listed below, ranges from just over 4% to under 20%, averaging less than 12%.

MG	CG	FT	LU	BT
11.1%	4.5%	11.8%	12.5%	19.3%

These findings are not a great revelation to the HPC community, but it gives us an idea of where to start looking for improvements. We must not overlook the fact that the performance efficiencies given above, due to data access and communication between processes, are prior to any effects of the network. These measurement were done on a single (shared-memory) node.

5 Can we do better? - sample ideas

Of course, the easy answer is to say "increase memory bandwidth and cache size and lower latency", when the code is data access bound (as is true for most codes), and "give us more floating point functional units" when the code is compute-bound, as is for dense matrix operations, for example. The latter is relatively easy, but not particularly impactful. The former is hard. We suggest, instead, to go back to the image/signal/graphics processing analogy and look for ways to optimize kernels, or what we might call "numerical operators" - though we don't forget the real challenge is in data access. Here are some partial, tentative, and somewhat random ideas.

Consider the computations derived from a stencil representation after discretization. Figure 2 shows a simple 6-point stencil.

To compute a given grid point we need a set of values which are not consecutive in memory. A cache line is loaded for one or two useful values. But if we computed along the index that is stored consecutively (say, the "i" index) then all the values brought in with the cacheline will be needed for computing the following grid points. The programmer or the compiler can direct the order of stepping through the grid. But there is no guarantee that the needed cachelines will not be replaced. Will it make sense to define a "stencil operator" as a macro instruction, allowing for parametrized number of stencil

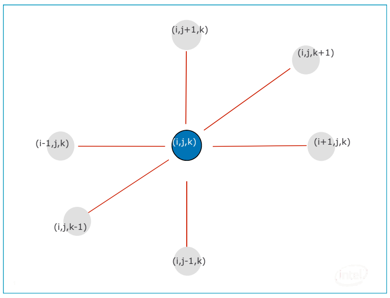


Fig. 2. A simple 3D stencil

points, and setting aside a buffer where these loaded 'vectors' can be kept? - this will result with a streaming of, say, 8 or 16 sets of computations before the buffer will have to be re-loaded. This is not necessarily practical, but illustrates how software and hardware can collaborate to provide higher performance for a useful and common sequence of operations.

Conjugate Gradient scores very low on efficiency mostly due to repeated passes through all the data with relatively small amount of computations done at each such pass. Here there might be a simple programming/algorithmic remedy. "Unroll" each iteration to perform 2 or 4 iterations through simple substitutions. We have done so in the past when the array did not fit in memory. We can do it now because the processors got so much faster. It is expected that this procedure will increase the efficiency by 2-4 times. Can a compiler be taught to unroll a loop in this iterative manner?

FFT performance is dominated by long distance communications, clearly noticeable even within a node (see FT above). The communication patterns are well structured and deterministic, though. Are there look-ahead ideas that, with some specially reserved buffer space, can reduce the shuffling of data, thus dramatically improve performance?

These are just sample random thoughts. A more structured approach is needed to provide cost-benefit analysis of where to place our efforts. An example of drilling in from the applications, to the common kernels, and to potential hardware support is the question of how best to approach the need of Gather/Scatter for HPC in a cache-based architecture.

Clearly, this short note does not do justice to the topic. More work is needed.

6 Conclusion: Proposed Actions

The desired outcome of this discussion note is to get the community at large to engage in creating a cluster-based holistic, integrated, backward compatible, application-based programming model. Whether the ideas and directions suggested here are followed is far less important than the getting together of all stakeholders to address the need for such "standard" programming model.

This is a call to the community - applications writers, software providers, and hardware vendors - to come together to define and implement a 3-layer (cluster, node, chip) programming model that:

- Extend MPI to allow layered, hierarchical, framework to express parallelism on a very large cluster. A single specification that defines the convention for the integrated model, and possibly adds directives that, for example, allow compilers to generate the calls to MPI routines.
- Adds mechanism for expressing interactions among cores within the processor chip. Allow extensibility to attached accelerators (OpenCL?).

The goals above are broad and directional in nature. A possible start can be to test the approach outlined here using a (crude?) prototype of the model on a couple of simple applications that span a cluster.

Just as important is setting goals for achieving higher performance out of each of the nodes that make up the total system. This, too, requires the HPC community to work with the Industry to -

- First, define and prioritize encapsulated computational kernels.
- Second, work jointly to come up with creative ways to combine software techniques, hardware capabilities, and architectural features that will enable a significantly higher efficiency of scientific codes.

The ideas presented here are far from even a proof of concept. Their intent is to encourage the community to create a more complete and more consistent framework for coding on our future HPC systems.

Acknowledgment

The author wishes to acknowledge and thank several Intel Corporation colleagues for helpful discussions and constructive feedback: Henry Gabb, Tim Mattson, Andrey Naraikin, and Rob van der Wijngaart.

References

[1] K. Asanovic et. al., *The Landscape of Parallel Computing Research: A View from Berkeley*, University of California at Berkeley, Technical Report No. UCB/EECS-2006-183, 2006. http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html
[2] Y. He, C. Ding, "Hybrid OpenMP and MPI Programming and Tuning", Lawrence Berkeley National Laboratory, 2004.
[3] M. Su, I. El-Kady, D. A. Bader, S-Y. Lin, "A Novel FDTD Application Featuring OpenMP-MPI Hybrid Parallelism", University of New Mexico and Sandia National Laboratory, 2004. http://ieeexplore.ieee.org/ielx5/9250/29349/01327945.pdf
[4] E. Lusk, A. Chan, "Early Experiments with the OpenMP/MPI Hybrid Programming Model", Argonne National Laboratory and University of Chicago, 2006.
[5] H. Gabb, "Hybrid Parallelism: where's the benefit?", LCI Conference on High Performance Clustered Computing, 2008. [contact henry.gabb@intel.com]
[6] NAS Parallel Benchmarks: http://www.nas.nasa.gov/Resources/Software/npb.html

EDF White Paper
IESP Workshop, 6-8 of April 2009, Santa Fe, NM-USA

JY Berthou and JF Hamelin and Etienne de Rocquigny
EDF R&D, 1, Av du Général de Gaulle, BP 408, 92141 Clamart Cedex, France
E-mail: jy.berthou@edf.fr, jean-francois.hamelin@edf.fr, etienne.derocquigny@edf.fr

As an industrial user with very high stakes in the operation and maintenance of complex systems like nuclear power plants, EDF has been engaged into simulation for many years. We have decided to design our own codes in order to capitalize precious knowledge on our fleet of nuclear reactors, and shorten the time to put this knowledge at work for the many engineering challenges that we have to meet. Software in the millions of lines have been written and explain why we feel very much concerned by the future requirements for Exaflops machines. We have already established the value of running our codes on 100 Tflops / 30 000 cores computers which yield a much better understanding of operating margins and in turn allow for a better optimisation of our power plants, increased safety and performance, lower environmental impact and costs and extended lifetime of assets. We have also recognized that some of our key industrial processes like waterflow within our nuclear cores or production optimisation under uncertain future are still out of reach of Petaflop grade technology and will require major changes in the way we write, validate, run and use simulation codes.

We therefore feel that Exaflops software should not only be thought as a way of tackling daunting research problems but should also take into account the sometimes equally daunting requirements that stem from an industrial usage perspective: this includes both the capacity to model very complex, possibly coupled phenomena over extended spatial and time scales, mixed with capacities like uncertainty quantification or data assimilation that are key to industrial acceptance. Our contribution to this IESP workshop is not that of software specialists but of fairly active users already engaged in the evolution of existing software for Petaflop/100 k cores machines. We will contribute the issues and problems that we are already facing at this first level, and that must find solutions for the future. We do feel that, whatever the hard changes that will probably have to be made on various software aspects, the group should not loose sight that continuity paths have also to be found in order to make those big changes acceptable and profitable to many. The context of simulation at EDF is detailed in [Hame].

[Hame] *"Jean-François Hamelin and Jean-Yves Berthou,* ***Getting ready for petaflop capacities and beyond: a utility perspective****, 2008 J. Phys.: Conf. Ser. 125 012001, July 2008"*

1 Major software barriers as seen by an industrial user of HPC and propositions for an international collaboration

One of the major difficulty will be to manage massively parallel systems, composed of approximately millions of heterogeneous cores that will appear at the end of this decade. The challenge is particularly severe for multi-physics, multi-scale simulation platforms that will have to combine massively parallel software components developed independently from each others. Another difficult issue is to deal with legacy codes, which are constantly evolving and have to stay in the forefront of their disciplines. This will require new compilers, libraries, middleware, programming environments, languages, as well as new numerical methods, code architectures, mesh generation tool, visualization tool:

We identified below what we think are priority research themes that could benefit of an international collaboration.

1.1 Programming massively parallel computers

Possible joint efforts:

- **Languages/compilers/performance** analysis tools for achieving **mono-processor** high performance, specially with accelerators (Larrabe, GPU, Cell, …)
  Goal : achieve more than 30% of the peak performance

- Efficient, "easy to use", portable and fault tolerant implementation of **Libraries/Languages/compilers** for mixed parallelism : MPI/OpenMP/"cuda like" language
  Goal: one million cores (heterogeneous, hierarchical and massively parallel)

- **Algorithm/solvers** and **data structures** adapted to heterogeneous/hybrid, multilevel and hierarchical massively parallel machines.
  Example: dealing with non-structured irregular meshes for CFD computation on GPU
  Goals:
    - No global communication involving the complete system(avoiding MPI_ALL-REDUCE, MPI_BARRIER,… on 1 million of threads)
    - exhibiting different type of parallelism (MPP, SIMD, …)
    - enabling fault tolerance techniques implementation
    - enabling efficient IO (data restructuring?)

1.2 A single generic interface for High Performance Solvers

Possible joint efforts. Defining and developing a single generic interface for High Performance Solvers

Computational scientists have developed over the past 20 years numerous[Dong] scientific libraries and solvers (direct, iterative and eigenvalue), ScaLAPACK, PETSc, HyPre, TRILINOS to cite some of them, which all have their own interface. This multiplicity of interfaces makes difficult and costly their integration and maintenance in end-user Scientific Application. It also makes tricky for a given community to test them and find the most appropriate for a given purpose. Both solver and code developers would greatly benefit of a **single generic interface for High Performance Solvers**. Moreover, coming with interfaces to freely available libraries, the sources of the codes are available. This is of great importance for industrial software stability in time. In order to be compatible with the external libraries, the necessary periodic efforts are only done once by the Interface's development team and not many times by each client software using, for example, PETSc or HyPre separately.

A similar project called Numerical Platon[NP] is developed by the French Atomic Energy Commission (CEA) . It provides an interface to a set of parallel linear equation solvers for high-performance computers that may be used in industrial software written in various programming languages (C, C++, FORTRAN, Python…). This tool was developed as part of considerable efforts by the CEA Nuclear Energy Division in the past years to promote massively parallel software and on-shelf parallel tools (public and in-house solvers, essentially PETSc, SuperLU and HyPre) to help develop new generation simulation codes.

Moreover, at EDF R&D, collaborations are currently underway to improve the direct solvers MUMPS[Mump] and PaStiX[Past] (Out-of-core, parallelization of the analyse step, null space basis computing) and their hybrid overlays (A2S2 and HIPS). These sparse parallel solvers are natural candidates to join such a product.

[Dong] J.Dongarra. *Freely available software for linear algebra on the web (sept 2006).* http://www.netlib.org/utk/people/JackDongarra/la-sw.html.
[Mump] Mumps' web page. http://www.mumps.enseeiht/fr.
[NP] B.Secher, M.Belliard & C.Calvin. *Numerical Platon: a unified linear equation solver interface by CEA for solving open foe scientific applications*. Nuclear Enginneering and design, vol. 239-1, pp87-95 (2009).
[Past] PaStiX's web page. http://pastix.gforge.inria.fr/files/README-txt.html.

1.3 Stochastic HPC computing for uncertainty and risk quantification

Numerical modeling of increasing complexity are developing in order to better characterize the underlying factors : multi-physics, multi-scale or complex portfolios all imply increasing computing power. Probabilistic quantification of the associated risks and uncertainties amounts to an additional technological challenge as one needs to multiply at a large scale these already-costly unit simulations in a framework that becomes stochastic. This also alters the way the computer power is invested in the sense that massive distribution becomes necessary; to best value decision-support computing power, one needs to re-work the compromise between the sophistication of best-estimate models and meshes and the stochastic exploration. On this rapidly evolving domain, two kinds of challenges may be highlighted: those related to the development of stochastic methods, and those related the associated computer science implications.

Probabilistic quantification of the risks and uncertainties affecting a best-estimate model has generated a whole domain of applied science, linking probabilistic, numerical analysis as well as physics and decision-theory [Rocq]. Beyond the traditional Monte-Carlo sampling whose history is closely linked to that of computing itself with Von Neumann's ENIAC pioneering applications, a number of uncertainty propagation and probabilistic simulation algorithms have been developed, such as accelerated sampling (importance sampling, particulate methods etc.), reliability techniques (FORM-SORM etc.), stochastic developments (e.g. chaos polynomials) and response surface techniques, yet still wanting for further development particularly regarding the challenges of low probability estimates for irregular response or high input dimension for sensitivity analysis/importance ranking or high-volatility time series.

Beyond uncertainty propagation or risk computation, even tougher challenges come with the need for inverse probabilistic techniques as the observable data to calibrate model variability generally comes on parameters different the model inputs, so that the identification of the extent of uncertainty affecting its input parameters requires the use of inverse techniques coupled with stochastic simulation. Closely related is the need for a general coupling between stochastic optimization and simulation in order to strike robust design or operational management strategies, with challenging mathematical implications that are only partially solved under existing Expectation-Maximization or stochastic dynamic programming algorithms (typically limited to close to Gaussian/linear behavior). Bayesian settings are also bound to develop to better incorporate expert knowledge in a solid decision-theory foundation.

Beyond the development of the methods itself there are key implications on the way HPC is structured and used: challenges involves striking an advanced compromise between parallel & distributed stochastic computing. While standard Monte-Carlo sampling leads to straightforward massive distribution as the runs are all fully independent, the other kinds of stochastic computing algorithms do need back-and-forth links between the various runs involved in exploring the stochastic space. For instance, past developments of uncertainty propagation such as adaptive importance sampling schemes have been designed with very limited link to the issue of computer implementation, ending up with purely-sequential formulations that fail to fully optimize the avenues offered by distributed computing to minimise the overall computing time. Adding the fact that parallel computing may be necessary to run a single simulation of complex underlying best-estimate models, optimizing the overall stochastic program becomes an insufficiently-researched domain.

[Rocq] *de Rocquigny E., Devictor N., Tarantola ed. (2008),* Uncertainty in industrial practice – A guide to Quantitative Uncertainty Management*, John Wiley & Sons*

1.4 Unified Simulation Framework and associated services

Advancing individual solvers performance is not enough to bring high performance simulation to the end-user. Each community needs a much broader set of tools in order to conduct industrial studies: CAD, mesh generation, data setting tools, computational scheme editing aids, visualization, etc.

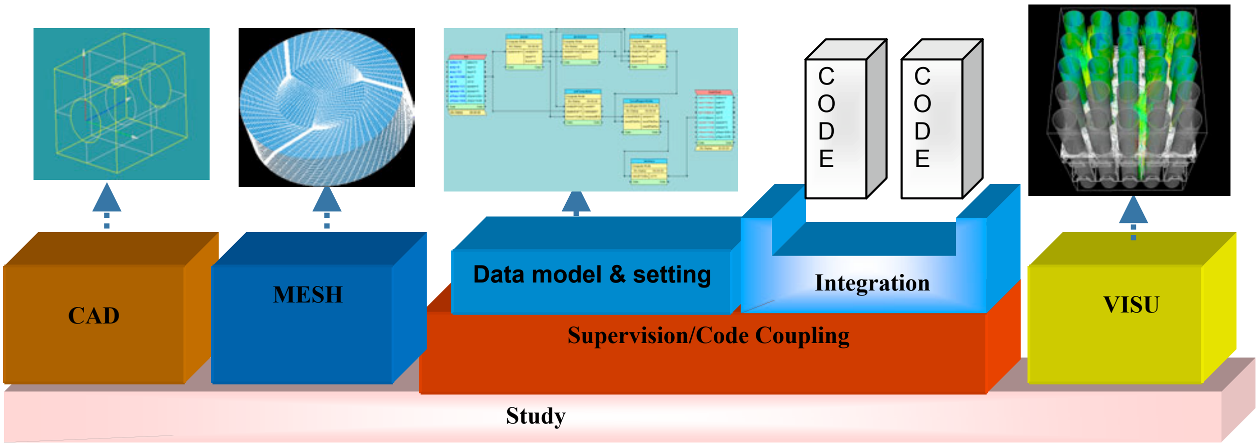


Figure 10. The Salome platform, www.platform-salome.org

In the early 2000 EDF, together with CEA and other industrial and academic partners, started the development of an integrated toolbox Salome www.platform-salome.org [Ribe,Berg], with the following aims:

- reduce the cost of complex simulation platforms by mutualizing a set of common tools: pre and post-processing, calculation distribution and supervision etc.
- boost performance through easy integration of multiple solvers for muti-physics studies (via a common data model).

If Salome has been proved to be well adapted for sequential and moderately parallel simulations it has to evolve in order to support **massively parallel computing**.

Possible joint efforts. Building a Unified Simulation Framework and associated services adapted to massively parallel simulation:

- Common data model : designing a common data model and associated libraries for mesh and field exchange adapted to massively parallel computing would enable interoperability and the coupling of independent parallel scientific softwares. High level operations on simulation data, such as mesh projection, data interpolation, could be implemented on top of this model.

- Meshing. In 2007 it took to the EDF CFD team several months to produce the 10^8 cells mesh for the simulation of part of a fuel assembly with the CFD code Saturne, compare to "only" 1 month of calculation needed on 8000 BG/L processors. Generating $x10^{10}$ cells mesh as targeted in 2015, requires future meshing tools to provide parallel meshing, automatic hexahedral meshing, mesh healing, CAD healing for meshing and dynamic mesh refinement.
  As an example are future works identified by a CFD Saturne code:
  - Re-evaluate if tetrahedra are really that bad
  - Our extended neighborhood gradient reconstruction scheme should reduce impact of non-orthogonality
  - Having mesh refinement algorithms would help, even if we don't do AMR right away
  - Some octree-based techniques lead to fully hexahedral meshes:
    - · conforming using stencils and smoothing

    - non-conforming with hanging nodes, using building-cube type method (also used by several codes, such as the Gerris Flow solver), combined with cut cells or immersed boundary
    - · At first, re-meshing on a low-quality, easily generated background would avoid issues with CAD interpretation and allow to easily define the local cell target size
  - • Using hierachical techniques would also make multi-resolution visualization possible
    - · We have been luckier with visualization than with meshing, but tools and formats have their limits

- Parallel visualisation tools. Considering the volume of data that will be produced by Petaflop and Exaflop computers, end users are needed adapted parallel visualisation tools and specific clusters to post-treat their simulation results. The international scientific community would benefit in focusing their research efforts in few software. VISIT and Paraview seem two good candidates.

- Remote and collaborative post-treatment: the sheer volume of data produced by Petaflopic/Exaflopic calculations, storage and network limitations, and multi-sites teams make it necessary to further advance R&D on remote and collaborative multi-user visualisation, parallel and distributed file systems.

- Supervising and code coupling tool, coupling schemes : EDF and CEA have engaged in 2006 the development of YACS, a new generation of supervisor, intended to handle parallel multi-physics coupling scheme through a portable parallel extension to CORBA named PACO++[13] developed by INRIA. Similar works are handle in US, based on different middlewares. Implementing tightly coupled scheme, involving scientific applications developed by separated teams with such generic tools is a particularly difficult challenge. A joint collaboration on code coupling tool architecture principle, middleware for massively parallel coupled simulations seems indispensable.

  The coupling using an external tool such as YACS is as less intrusive in the legacy codes as possible. On the other hand, we share the advanced coupling algorithms for all multi-physic simulation in a dedicated algorithmic box in the SALOME platform. From an algorithmic point of view, the existing couplings are mainly explicit and semi-implicit (fixed point algorithm). Current work are performed to implement Newton-like algorithms.

[Ribe] Ribes A and Caremoli C 2007 SALOME platform component model for numerical simulation *COMPSAC* july, Beijing, China
[Berg] Bergeaud V and Tajchman M 2007 Application of the SALOME software architecture to nuclear reactor research *SCS Spring Simulation Multiconference on High Performance Computing Symposium*, Norfolk, USA

The Biggest Need: A New Model of Computation

Thomas Sterling
Louisiana State University
March 30, 2009

HPC is experiencing a phase change driven by technology advancements and constraints. For the first time in more than a decade conventional software practices for programming and managing system resources are no longer sufficient to address the challenges for achieving the high end scalability for a wide range of applications. Further, past strategies for system hardware architecture no longer utilize the emerging technologies to their fullest. Both are reflected by the emergence of heterogeneous multicore components and the systems that use them. Power is restricting clock rate, design complexity has been exhausted as a path of future performance gains, and within a decade parallelism on the order of billion-way concurrency will be required. Historically, major disparities between enabling technologies and the methods of their use have driven computing through an evolutionary event of punctuated equilibrium requiring simultaneous changes in architecture, programming models, and system software to achieve a new balance for efficiency and continued progress in performance gain. Sequential, vector, SIMD-array, systolic, dataflow, multithreaded, and most recently communicating sequential processes represent distinct phases in HPC, each a different model of computation. Currently, a new such model is required to redress the challenges imposed by the need for multicore.

A model of computation is not a programming model, architecture, operating system, or some form of virtual machine. Instead it is a strategy or discipline that specifies referents, their interrelationships, and the actions that can be performed on them. In so doing, a model of computation governs the semantics of state objects, function, parallel flow control, and distributed interactions. While it provides an image of an entire parallel computer, not just any single core, it leaves unbound policies of implementation technology, structure, and mechanisms. Yet, it influences the decisions for co-design of programming languages, compilers, runtime software, operating system, and even hardware architecture.

The goal of a new model of parallel computation for future Exascale computing is to serve as a discipline to govern future scalable system architecture, programming methods, and runtime techniques as semiconductor technology proceeds to nanoscale feature size. Such a new model has to innately hide latency both system wide and to main memory. It has to exploit parallelism in a diversity of forms and granularity. To this end it has to provide a framework for efficient fine grain synchronization and scheduling, enabling optimized runtime adaptive resource management and task scheduling for dynamic load balancing. Perhaps for the first time, the model of computation must extend farther to support full virtualization for fault tolerance and power management.

Then what would a new model of computation look like, even as it replaces the venerable message-passing model? While no definitive specification can be given without substantial research in collaboration with the international community, there are a number of attributes that may prove imperative if it is to serve computing down to the nanoscale and up to the Exascale within the next decade. Perhaps most fundamental is to replace static processes assigned one on one to fixed processor cores with a new relationship between tasks and computing resources. One possibility is the basic work-queue model where each physical resource acts as a server to process a stream of task specifiers that work on relatively local program state. Instead of waiting for some remote access, the resource terminates the current task and begins a new one. Thus, the work-queue model decouples virtual tasks from physical processing resources to significantly increase resource utilization, at least in cases with sufficient parallelism. Complementing the work-queue model is the need to adopt a message-driven model, replacing conventional message passing. Message-driven computation allows work to be moved to the data when this is optimal, rather than always requiring that data be constantly moved to a fixed location of work. This is particularly well suited to dynamic graph problems such as adaptive mesh refinement and informatics. Such algorithms are heavily reliant on meta-data to describe the data structures. Extremes in parallelism will be required with future systems and meta-data used with message-driven computing may expose a diversity of parallelism forms and sizes, at least in comparison with conventional global barrier based techniques. Instead asynchronous methods need to be incorporated in the global flow control for adaptive management of resources. The elimination of global barriers allowing more flexible flow control such as data flow methods can be achieved with the powerful futures construct. To harness hundreds of millions of cores in to a single system requires a model capable of unifying all components and this requires a single system-wide name space. PGAS has been pursued and serves well for some problems. But when dynamic migration of first class objects is required; something more is needed so that data objects can move in physical space without changing their virtual names. Such an active global address space still excludes full cache consistency but enables lightweight access to remote data without overly constraining the distribution of that data. The parallel flow control state at the global level must be more flexible than simply the state of fixed allocation SPMD processes. A more powerful means of parallel control state based on the distributions of “continuations” is needed to decouple flow control from fixed physical resources allowing migration of control such as when traversing a dynamic directed graph. A new model will support a self-aware property that adjust system configuration and application rollback for fault tolerance and active power management.

The development of a new model of computation will free future application programming from the deadly embrace of MPI + Clusters/MPPs where no progress can be made because each is required to serve the other. It will permit a new co-design cycle of all levels of the system software and hardware delivering new programming models that will greatly simplify the programmer responsibility, dramatically improve efficiency, and exploit orders of magnitude more parallelism intrinsic to at least some algorithms.

NSF IESP Whitepaper

Abani Patra, Rob Pennington, Ed Seidel

Office of Cyberinfrastructure

National Science Foundation

Within the context of the "NSF's Vision for Cyberinfrastructure for 21st Century Discovery" document, NSF is developing a comprehensive program for supporting the national cyberinfrastructure (CI) for science and engineering, including major HPC facilities, grids, networks, software, data, and virtual organizations. NSF clearly cannot do this alone, and therefore must pursue global partnerships with other organizations and agencies.

NSF reaches deeply into every campus in the US, covers all the sciences and engineering areas, and in terms of cyberinfrastructure which includes HPC, is very broad. NSF researchers will clearly benefit from a stronger software program, improved support for complex applications and strengthened integration with campuses. Students and postdocs will benefit from training in software engineering, use of advanced CI, and socio-technical activities that are critical to success in many complex research activities.

The computational community is already dealing with several major challenges at petascale, including new hardware using manycore, massive scaling, system software, file systems, applications software, debuggers, applications development, programming environments, machine rooms, cooling and power costs.

Exascale challenges will drive innovation in many CI related areas. Developments in cyberinfrastructure to support scientific and engineering research will need to be integrated across the following major topics:

- Software: A major software grand challenge program responsive to emerging architectures needs to be developed, involving national and international efforts.
- Applications: NSF funded researchers have strength and breadth in the community that will use exascale facilities. New research challenges will further broaden the application coverage.
- Hardware: R&D activities in hardware design that are responsive to the most challenging application needs.

Questions for consideration:

The NSF cyberinfrastructure vision document provides the current high level framework for cyberinstructure strategy. The requirements for cyberinfrastructure are evolving rapidly and, as a result, new questions arise in planning for future cyberinfrastructure. As part of the process of understanding these requirements, we welcome discussion and input on a wide range of questions including the following.

- How will present & emerging applications use exascale systems?

- What are the new applications that are emerging or likely to emerge in the coming decade?
    - Are they new application domains, new modeling modalities, multimodal modeling, dynamic/on-line integration computation and measurements?
    - How will technology advances drive the advancement of applications capabilities (technology-push)?
- How can NSF best stimulate development of exascale software applications?
- How can application needs drive the design of hardware platforms, system software, and applications software development environments?
- How will new architectures aid or impede successful reformulation of problems for parallel solution approaches?
- How can useful software that has been developed as part of the exascale effort be sustained beyond the development period?
- What systems software will be required? Distributed systems support, programming environments, runtime support, data-management user tools?
- In what ways will fault tolerance need to be considered by the applications developers? By the system software developers?
- What application support environments will be needed? Application packages, numeric and non-numeric library packages, problem-solving environments?
- How can NSF aid seamless portability of applications across different hardware and software platforms as they all evolve?
- How can NSF aid or catalyze developments that make it possible to provide the same user experience and where possible use the same tools, including compilers, debuggers and performance tools, on system scales all the way down to the typical researcher's laptop or desktop?
- How can the community of science and engineering researchers who will use exascale systems be best supported in a rapidly changing environment?
- How feasible is the development of generally applicable software that will enable efficient translation of problems to programs? What priority should be given to pursuing this approach?
- What education and training actions should be considered to prepare researchers, students and educators for future cyberinfrastructure?

A Proposal for a Capability Centers Consortium

Bill Gropp & Marc Snir

NCSA. University of Illinois at Urbana-Champaign

22/03/2009

Introduction

Our proposal is motivated by the following observations:

A. Most, if not all of the users of the top NSF Supercomputing centers are using resources at more than one center; in particular, smaller systems and regional centers will be used for developing codes for the larger capability systems – in particular, Blue Waters. Users will be greatly advantaged if the various platforms they use have compatible application development environments and execution environments: same interfaces, languages, libraries and tools, and similar procedures.

B. No vendor provides a complete solution to the needs of the HPC community; any large platform will deploy a variety of libraries and tools that were developed by national labs or academic researchers. The development, porting, tuning, and maintenance of such software packages require collaborations with a variety of partners.

C. The desire for compatibility across platforms often lead application teams to seek the "lowest common denominator" – to use only basic languages, libraries and tools that are guaranteed to be available and well supported on all platforms. New approaches with the potential to increase the productivity of the application programmers are not adopted because of this well known vicious circle: application programmers are reluctant to use software that is not well-supported on most platforms; and platform providers are reluctant to support software that is not used by a large number of applications.

Our goal is to create mechanisms that

- Facilitate the sharing of expertise and information about user needs, system operation and HPC software among the top supercomputing centers.
- Facilitate the sharing of expertise and information about the use of large HPC systems among the users of top supercomputing platforms.
- Facilitate collaborations between these centers.

- Encourage the deployment of common software on all major HPC platforms used by scientists; in particular, encourage the deployment of new languages, libraries and tools.

This initiative is synergistic with other extant initiatives:

- XD: Our proposed initiative is (a) focused at the very high-end of the performance pyramid; and (b) is not aimed, like XD, at developing a specific s/w infrastructure, but at sharing information and collaborating in the deployment of any s/w that has can be common to many capability platforms.

- Exascale s/w initiative: Our proposal is aimed at creating strong interactions between the current, petascale centers. Such interactions are essential in providing a transition from new research products to actual deployment and utilization on available systems. Our initiative will provide a receptive environment for the technologies emerging from excascale s/w research.

- PRACE: PRACE provides a common meeting point for the top HPC centers in the EU. Our initiative can have a similar role in the US and can establish a strong collaboration with PRACE.

Potential Activities

Information Sharing

Information sharing can occur through

- Periodic phone conferences
- Periodic workshops, possibly focused on topics of common interest, such as parallel file systems
- Shared social networking tools (wiki, discussion groups, mailing lists, etc.)

Different mechanisms may be used for different groups – with an emphasis on regular interactions for the centers and on social networking tools for the users.

Information Aggregation

Shared information can be made more useful by collecting it in a common format and aggregating it. Possible examples include

- An inventory of used open source software (Pete Beckman, ANL has started this activity)
- An integrated directory of people: a list of contacts at the various centers for various subjects.
- An integrated directory of documentation and educational materials

- Aggregated statistics on system utilization, types of applications run, etc.: centers will agree to a core of consistent metrics
- Aggregated customer surveys: centers will agree to a core of common questions in their surveys, so as to enable aggregation of the results
- Aggregated bug reports: for vendor/platform related bugs, this will probably need to be done on a per platform basis and possibly kept confidential; for open source software, the information should be public. Centers will need to agree to consistent ontologies.

Collaborations

Collaborations can reduce duplicate work, and increase efficiencies in the various centers. Such collaborations may include

- Shared development of tools (e.g., application performance tools or system monitoring tools). The development is likely to occur in one place, but early interaction with other potential users will increase the odds that tools are portable and satisfy the needs of a broader community
- Shared testing: the development of good regression test suites is expensive; the sharing of general test suites, as well as tests focused on specific issues (such as OS jitter) can greatly benefit the centers
- Collaboration in the deployment of new software
- Collaborations in the evaluation of various tools and environments
- Collaborations in the development of education material and user guides

Standardization

Different centers have platforms form different vendors, with different software environments and different users; extensive homogenization of these environments is neither possible nor desirable. On the other hand, the differences between platforms are often spurious. Discussions between the centers could lead to agreements on a minimum common s/w stack for petascale/exascale platforms, either through support for the same tools and libraries, or through the provision of compatible profiles.

Issues

Funding

The consortium will need core funding for meetings and for support activities. It will be important that activities at the centers be funded from a budget managed by the consortium, to ensure that commitments are met.

Organization

The consortium needs a management model that ensures that decisions can be reached in a timely manner, while providing buy-in from the involved centers. This would probably involve a small executive committee coupled with a board representing all participating groups.

Technical Issues

1. Do we specify a particular version or range of versions (at least as a default)? What do we do if some version has a security hole and needs a quick fix (what is our contract with our users about stability of the choices)?

2. Do we specify a base version and allow extensions? If so, how do we make the base strong enough so that many/most users can and will choose to stay within that base level? Should there be more than one? E.g., there could be a standards-compliant level (POSIX) and an enhanced level (GNU+POSIX). Since the software stack will continue to evolve quite rapidly, easy composability – the ability to add components developed by other groups -- might be more important than a standard core. To achieve this, it may be more important to specify standard interfaces for extensions – rather than detailed core functionality.

3. How do we track changes and evolution of software/standards? Should we have a shared repository (containing version information, header files, if appropriate, and source files, if appropriate).

4. How do we test compliance (more gently, how can sites quickly assess whether their environment conforms to the spec)?

5. How do we make sure that users adopt? What is the process for user buy-in? How do we assess success in getting users to work within the base set(s)? Good social networking tools that facilitate the sharing of experience by users may be an important component of the solution to this problem.

Inventory

We list below software components that are relevant to the proposed consortium and issues raised by those components:

1. Compilers, linkers

    a. Primarily provided by vendors or GNU

b. Key issue is which languages (and which versions) are available – E.g., Fortran 2003, C99, etc. This could be a significant problem, as some vendors are slow to conform to current standards.

c. Issues for users are often prosaic ones such as common command line arguments, particularly for include and library paths.

d. A major issue is extensions to the languages – GCC implements many extensions that are often exploited in user code. Can we standardize on these extensions or on GCC?

e. Linking is also a problem – AIX has a very different approach to shared and dynamic libraries than most other Unix implementations (Mac OS/X also has a different -- though less so -- view).

f. A key approach should be to specify interfaces, not particular products (e.g., MPI 2.2, not MPICH2 or OpenMPI).

2. Build tools (make, configure, etc.)

   a. Make in various forms is provided by the vendor and by GNU.

   b. Configure dominates, but there are other tools such as cmake. Configure doesn't handle cross compilation environments well and most autoconf scripts (e.g., user programs in configure's language) are not correct with respect to cross-compilation – this may be a significant issue for some HPC platforms.

   c. Again, we have the problem of extensive GNU extensions to make – can we standardize on a subset, standardize on GNU tools everywhere, or something else.

3. Debuggers

   a. Primarily provided by vendors or GNU.

   b. Far less standardization; few truly parallel systems (such as Totalview); the consortium must avoid picking a solution here.

   c. An example of a place where we may want to specify a base level but allow (and even encourage) sites to innovate here.

4. Performance tools

   a. Low level tools/APIs such as PAPI.

i. It would be good to standardize on (something like) PAPI, or provide a per-node (instead of per-process) version that could be used as a kernel module instead of as a source code patch.

b. Command line, single thread or single process tools (e.g., gprof). Eliminate variations in output format, input commands, etc.

c. Parallel performance tools

i. Aggregate tools (mpiP, fpmpi)

ii. Trace-based tools (Tau, VAMPIR -- now Intel Trace Analyzer, Jumpshot, Scalasca, ...)

d. Extension of the above tools to OpenMP, UPC, CAF, …

5. Visualization and Data Analytics

a. Another example where a base set + site-specific extensions is necessary.

6. Parallel File Systems. Are full POSIX semantics (which can impact both performance and stability of the file system) required for all files? There are efforts to define more scalable POSIX APIs for file system metadata (e.g., to more efficiently handle directories with tens of thousands of files); what role can the consortium play in developing these enhancements? Can we provide better tests and diagnostics to ensure that parallel file systems provide efficient support for user parallel I/O needs?

7. IDEs

a. Can we standardize on an IDE, such as Eclipse? How do we handle versions (there is a lack of stability with many of these tools)?

b. Standard plugins, for

i. Each language

ii. Each parallel programming model/extension (e.g., MPI + C++, OpenMP + Fortran, MPI + OpenMP + C)

iii. Debuggers

iv. Job control (mpiexec, batch job submission, job status)

v. Performance debugging/analysis

vi. Remote use

8. Parallelism

   a. Which versions of MPI and OpenMP?

   b. UPC and CAF. Which versions of these?

   c. Interoperability of models – e.g., can you mix MPI and UPC routines in the same program?

   d. Parallel I/O

      i. Are POSIX semantics supported?

      ii. Are consistent semantics supported (e.g., PVFS, but not NFS v3)

      iii. Do we encourage single file per job instead of one file / core (with all of the support tools)? Are there common tools for managing collections of files?

9. Running codes

   a. Standardize on mpiexec (part of MPI since MPI 2.0).

   b. Standardize on OpenMP environment variables.

   c. Standardize on basic batch commands (see the DOE SciDAC project on system software that created a component-based framework for job management into which 3rd party components could be included).

10. Libraries

   a. Standard, stable libraries (e.g., BLAS, ScaLAPACK, LAPACK)

      i. These should be in a standard place and be optimized for performance.

      ii. Need a way to define this list in concert with users.

   b. Libraries from research groups (e.g., PETSc , SPRNG, WSMP, FFTW)

      i. These libraries are not stable; they change over time (though slowly, because they have a user community).

      ii. These cannot be tuned independently from their development group – a collaboration process will be required to both harden the code (portability, error reporting, coverage testing) and tuning for different platforms.

iii. User support (bug reports) and training could be standardized; “level 1” support could be provided by each site.

11. Frameworks from research groups (e.g., Cactus)

    a. Same issues as libraries from research groups – all work must be done collaboratively.

12. Software environments

    a. Standard set of scripting languages (which versions?).

    b. Common way to select software versions (e.g., module, softenv).

    c. Predefined personalities (e.g., a GNU personality for AIX).

13. Batch schedulers and resource managers

    a. Standard interfaces for workflow engines.

    b. Minimal common functionality.

14. Monitoring and error handling

    a. Interchangeable event descriptors (a definition of a minimal amount of information contained in an event descriptor).

Slouching Towards Exascale

Ewing Lusk
Mathematics and Computer Science Division
Argonne National Laboratory

Introduction

Let us speculate about how we will program exascale machines. Some believe that the current "standard" of MPI plus a venerable sequential language (Fortran, C, or C++) will become as abruptly obsolete as the vector Fortran compilers of the 1970s. While it is exciting to contemplate an *ab initio* redesign of the HPC software infrastructure, experience tells us that large-scale software (and HPC software is now very large scale) requires a migration path that consists of incremental steps during which only some parts change at a time. Indeed, as scalability forced vectorization to give way to message passing, Fortran changed a little but was not replaced by Ada.

Where We Are Now

We are about to take another major step, but not a cataclysmic one. We now have robust, portable, and effective standard languages for programming a von Neumann machine with a single program counter and a single address space. Thanks to MPI, we have a robust, portable, and effective standard for communication and synchronization among such machines, What we lack is a robust, portable, and effective standard for parallel programming (multiple program counters) within a single address space. (Neither OpenMP nor POSIX pthreads provide features needed for an approach effective for HPC.)

MPI, admittedly cumbersome for some straightforward tasks, has become the universal mechanism for expressing parallelism among multiple address spaces, for several reasons. Designed through a completely open process, it included the concerns of multiple stakeholders from the beginning. This process resulted in a definition that was portable to a wide class of machines and with a certain degree of performance transparency that encourages the development of high-performance, scalable libraries and applications. MPI's design favored the development of portable libraries over end-application programs, and in this it has been successful. Its specification includes language interoperability and other features that enable it to fit into the HPC ecosystem with existing tools. These properties are worth reviewing because we must be sure that what we add to our programming environment be not *worse* than MPI.

The Next Step

The next step we are about to take is forced upon us by physics, so it is pointless to resist. Because of power and heat dissipation requirements, multicore chips are already with us. Whatever shape exascale computers ultimately take, we will be programming machines with less memory per processing core than we are now. This reality will force most (not all) applications to augment their existing programming model to include parallelism within an address space together with their current MPI-based parallelism across multiple address spaces.

This “hybrid” style of programming is already being used by applications in many areas as they migrate toward petascale. The current most common shared-memory approach is OpenMP. Although high-performance programming is difficult with OpenMP because of its lack of locality control, OpenMP+MPI is virtually the only approach being widely used, for several reasons: (1) OpenMP is available on a wide variety of machines; (2) both Fortran and C are supported; and (3) the OpenMP and MPI standards make explicit commitments to each other that provide clear semantics for various levels of thread safety in hybrid programs. The fact that OpenMPI+MPI represents an incremental step for most applications (the overall MPI structure of the application can be maintained while the MPI processes are internally parallelized with OpenMP threads) is an important factor in encouraging applications to move to a hybrid model.

But OpenMP, at least as currently defined and implemented, is unlikely to be the final answer for shared-memory parallel programming. In addition to the lack of locality control, most implementations are restricted to single-node parallelism, where the hardware provides the shared memory and synchronization mechanisms. Applications are already finding the need for larger memories to be associated with their MPI processes than are hosted on the single nodes of petascale machines. Therefore it may be useful to consider the PGAS languages (UPC, Co-Array Fortran, and Titanium), which offer a shared-memory model with a distinction between local and shared memory, thus providing locality control and performance transparency.

What the PGAS languages lack so far is clear semantics for interaction with MPI and implementations to match. One can imagine a million-thread computation organized as 10,000 UPC or CAF address spaces with 100 threads each, communicating via MPI, which strains the scalability of neither model. Again, this would be an incremental change for an existing MPI application.

Libraries

In discussing approaches to parallel programming, one often forgets that not all programmers require the same features from their programming models. Let us define a *library* as a collection of functions that are usable in multiple applications. Writers of such libraries need access to performance and (except for certain vendor-specific libraries) portability. To obtain these features, they are willing to give up a certain degree of ease of use. Application writers, on the other hand, wish to focus on their science and would rather not cope with some of the details required for scalability and performance. For them, the easier it is to develop applications, the better they can produce computational science results.

We are most familiar with the dichotomy between application and library in the case of mathematical software, since the mathematics is the same for so many applications. But there also exist libraries that are specialized to certain families of algorithms rather than areas of application. For example, researchers have expressed interest in sophisticated load-balancing libraries that can hide all of the MPI communication from an application code, simultaneously providing scalability while simplifying the application logic.

What We Need to Do

Four actions would make progress toward programming exascale machines.

- *Eschew ritualized denigration of MPI.* It is a robust definition, with robust implementations, of a critical component of future programming systems, namely the transfer of data among separate address spaces. Support continued research into areas of MPI that need it. The MPI-3 Forum is at work on extending the standard.
- *Recognize the need for a shared-memory programming model.* What current applications and libraries alike will embrace is a programming system for parallelism within an address space. Such a system needs to be comparable with MPI in portability and performance transparency. It need not be scalable to the ultimate levels, but should not be restricted to running on a single node. Clear semantics for interoperability with MPI are required. This is a critical research topic; multiple solutions should be pursued at this point. PGAS languages show promise, but semantics for interoperability with MPI are not yet there.
- *Understand the difference between end applications and libraries*. While some applications will use hybrid systems consisting of explicit management of parallelism within an address space together with MPI, other applications may be able to rely on libraries, some of them specialized to single algorithms or domains.
- *Don't abandon the HPCS language ideas*. While separate, vendor-sponsored development of multiple "high-productivity" languages has not attracted much attention from application programmers yet, the HPCS languages (Chapel, X10, and Fortress) have introduced a number of important ideas. An open, multi-agency program with a clearly defined research focus could ultimately bear significant fruit.

Conclusion

This has been necessarily a simplified speculation on programming models for exascale machines. In particular, it has largely ignored the issue of GPUs (although they often come with their own shared address space and thus require a shared-memory programming model) and has focused on hierarchies having depth of only two. Even within these simplifications, however, many challenges and exciting research opportunities exist on the path to exascale.

A Collaboration and Commercialization Model for Eascale Software Research

Mark Seager and Brent Gorda,
Lawrence Livermore National Laboratory
March 24, 2009
Version 3

Motivation

In the US, recent software research and development for petascale systems has been performed by two main entities: US Government funded R&D collaborations (both at Universities and at Government Labs) and Industry efforts at products. With few notable exceptions, there has been little diffusion of technology from the R&D collaborations to industrial efforts and little feedback from the industrial efforts to the US government funded R&D efforts. However, the broader community has found value in some of the R&D efforts and would like to see continued support. For the most part, support is voluntary by the development groups because the funding was only for the R&D, not ongoing support. On the other hand, industry efforts end up being funded for specific platforms and are generally proprietary and suffer from the lack of overall effort due limited private and public investment. Understanding these lessons from petascale efforts is essential for forming a coherent strategy going forward to exascale. Clearly, a different research and development and commercialization model is desired going forward.

Proposed Model

Many US Government funded R&D collaborations produce useful results and lessons learned that are available to the HPC community for a variety of platforms. There is also much duplication of effort within various HPC vendor organizations in the name of differentiation and specialization. Both of these approaches are inefficient because they don't effectively leverage each other. The basic R&D efforts don't feed into commercial development models and overall requirements from customers fielding systems are not being fed back into the R&D efforts.

To overcome this and align forces toward the Petascale, we propose a new Open Source Collaborative R&D model with commercialization paths. This leverages the "best of breed" development models from DOE Office of Science (DOE/SC) petascale research efforts that are typically Open Source, Community development based. It also leverages the NNSA Advanced Computing and Simulation (ASC) PathForward (now FastForward) program where HPC provider product roadmaps are accelerated and provide a clear commercialization strategy.

Figure 1 depicts the proposed model graphically. In this model for software development for exascale systems, we retain the flexibility of R&D efforts to experiment, push the boundary and to be allowed to fail. The fruits of these efforts (in the blue STAR figure) are handed off as harvestable results (e.g., code, algorithms, models or techniques) and as "lessons learned." These are harvested by a new class of efforts labeled as Development and Engineering (D&E) collaborations in the Orange Box. These D&E

ASC PathForward-like efforts include a commercialization path should the results be successful. These products are then delivered and supported on various HPC systems by the providers of these commercial technologies (e.g., system software by system vendors and ISV products such as code development tools). The key difference is the management and funding model for these efforts. Rather than separate independent efforts in R&D, D&E, Products and Support, we propose they be linked. Funding agencies for the D&E collaborations (E.g., ASC and DARPA) should participate as contributors in the R&D efforts (e.g., DOE SC and NSF). That is, the R&D organizations should continue to lead the R&D portions, but include contributions from organizations that focus on the D&E collaborations. Likewise, R&D organizations should contribute to the D&E funding planning and execution in the D&E efforts. As vendor partners contribute to the D&E collaborations, natural commercialization strategies will emerge. Vendor partners should also be included, when appropriate, in the R&D collaborations.

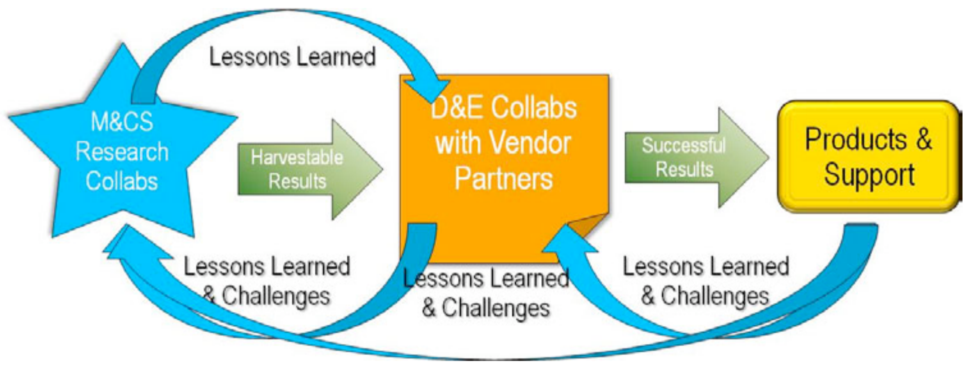


Figure 1: A new software development model for exascale systems couples basic R&D with commercial effort so leverage the best of both worlds.

In all cases, linkages between stages should be valued as part of the project selection process in order to incentivize the migration of technology from R&D to D&E and ultimately into products and services. Naturally some R&D proposals could be formed without D&E collaboration paths, but may be selected for funding based on the strength of the technical merits. In other words, the model should be flexible, but encourage and incentivize technology migration.

A side effect of this strategy is that at every stage of migrating technology from left to right in Figure 1, there is a corresponding opportunity to shape the agenda of upstream events by migrating challenges, requirements and “Lessons learned” in counter flow direction (right to left in Figure 1).

There is a large gap between what has been developed for current 100s of teraFLOP/s Linux clusters and 1-20 petaFLOP/s systems that have been delivered or are on the horizon. The larger system comes with huge requirements in terms of scalable systems software and file systems; Reliability Availability and Serviceability (RAS); programming models and application resiliency. It is important that the community consider multiple passes through the process depicted in Figure 1 be attempted before fielding exascale systems in 2018 and beyond.

MAIN PRINCIPLES

1. Coordinate strategy between R&D->D&E and D&E->P&S. With migration path towards commercialization.
2. Keep current focus areas and funding agents for R&D, D&E and P&S as they currently are and add stake holders from next stage in the process.
3. Keep the model flexible as possible to encourage development and competition.
4. Multiple iterations required to get to exascale.

The Case for A Hierarchal System Model for Linux Clusters

Mark Seager and Brent Gorda,
Lawrence Livermore National Laboratory
April 6, 2009
Version 2

Motivation

The computer industry today is no longer driven, as it was in the 40s, 50s and 60s, by High-performance computing requirements. Rather, HPC systems, especially Leadership class systems, sit on top of a pyramid investment mode. Figure 1 shows a representative pyramid investment model for systems hardware. At the base of the pyramid is the huge investment (order 10s of Billions of US Dollars per year) in semiconductor fabrication and process technologies. These costs, which are approximately doubling with every generation, are funded from investments multiple markets: enterprise, desktops, games, embedded and specialized devices. Over and above these base technology investments are investments for critical technology elements such as microprocessor, chipsets and memory ASIC components. Investments for these components are spread across the same markets as the base semiconductor processes investments. These second tier investments are approximately half the size of the lower level of the pyramid. The next technology investment layer up, tier 3, is more focused on scalable computing systems such as those needed for HPC and other markets. These tier 3 technology elements include networking (SAN, WAN and LAN), interconnects and large scalable SMP designs. Above these is tier 4 are relatively small investments necessary to build very large, scalable systems high-end or Leadership class systems. Primary among these are the specialized network designs of vertically integrated systems, etc.

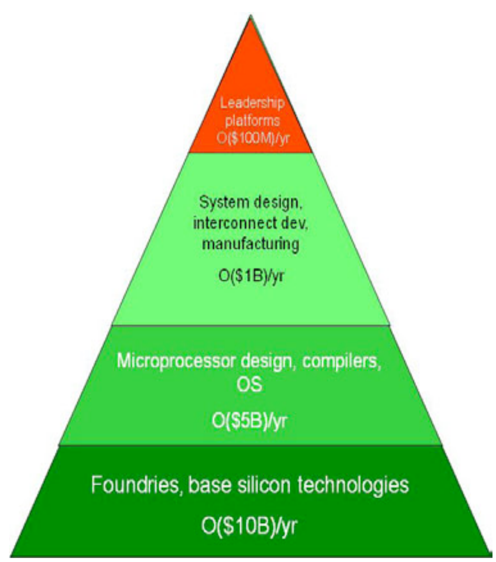


Figure 1: Leadership-class HPC systems sit on top of a $15B+ pyramid of investment.

The Hierarchal Systems Model of Petascale Systems

Since the mid-1990s Linux clusters and proprietary, vertically integrated systems (PVIS) have leveraged the above hardware and software pyramid investment model. The gap between the scalability of COTS Linux clusters and PVIS systems have diminished in the intervening years and now form a major fraction of the TOP500 list. However, with recent development in PVIS, such as IBM BlueGene and Cray XT4, the scalability of PVIS has again vastly outstripped basic Linux clusters. By looking at lessons learned in the march to petascale PVIS, we have learned that one must focus on three things: **scalability** of hardware, **scalability** of system software and infrastructure and applications **scalability**. Key observations on hardware and system software scalability coming out of the BlueGene experience are: 1) keep the highly replicated hardware and software components as simple possible and still get the job done (known as KISS, or Keep It System, Stupid); 2) applying a "factor and simplify" design methodology[1] leads to a hierarchal system model for both hardware and software; 3) the runtime environment (including the OS and system services) felt by applications must extremely low noise. These design principles lead to an extremely simple (small parts count) compute node implementation with MTBF measured in the field of about 3 millennia. On the software side the highly replicated unit is the light weight kernel (LWK). Due to the simplicity of the compute node architecture all external (but not interconnect) I/O and other OS functionality is function shipped to IO nodes (ION) with an external SAN interface. This creates a hierarchal system model where there is a large number of CN and a reasonable number of ION (about the same size as a small to medium size Linux cluster). If we now add a few Login nodes where users login interact with the system (e.g., code development, batch job management and visualization) and a few Service Nodes for the RAS infrastructure and scalable system administration, then we have the basis for a fully hierarchal system infrastructure. For example, job launch and debugger daemons can be migrated off the compute nodes (and thereby reduce the system noise and improve software reliability by keeping the CN LWK environment simple as possible) to the ION.

[1] The "factor and simplify" design methodology takes a seemingly impossible problem (e.g., scaling Linux OS to 65,536 way parallelism for BlueGene/L) and breaks it into two problems; one of which is easy to solve and the other is merely difficult (e.g., a light weight kernel on the 65,536 compute nodes (the difficult piece) and function ship to Linux on 1,024 IO Nodes (the solved piece)).

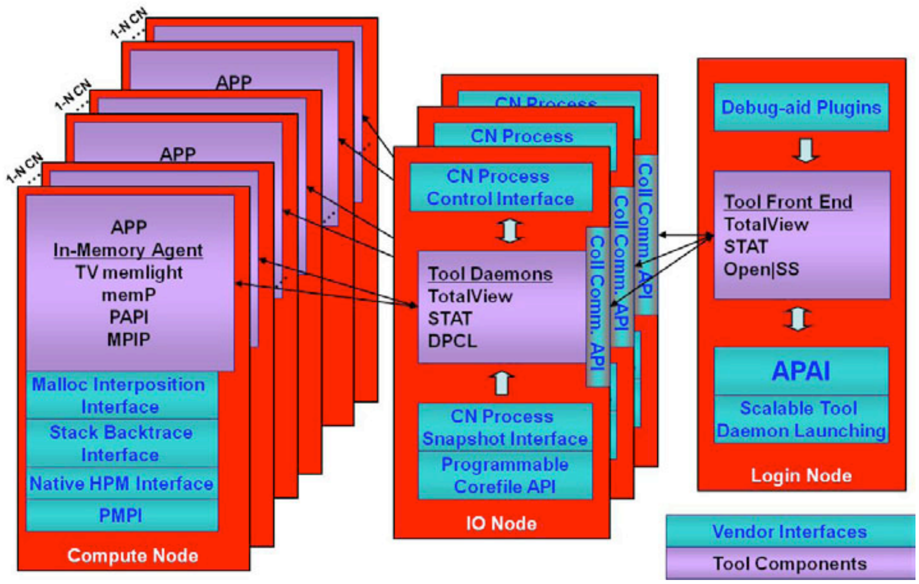


Figure 2: Hierarchal PVIS system model showing how the next generation of scalable tools can naturally use this hierarchal hardware and software infrastructure.

This offers the opportunity for a hierarchal system infrastructure that associates a set of ION with CN at job launch time and job launch is also hierarchal in the sense that a user submits the job and the batch system, running on a SN launches job steps (parallel jobs) that start by launching daemons on behalf of the user on the ION and then those daemons launch the job (and later manipulate it for the debugger and other performance analysis tools) onto the CN LWK. This "factor and simplify" approach also serendipitously provides a fan out infrastructure for tools and other system services. This fan out infrastructure approach provides unique opportunities for scalability. For example, debuggers when setting memory watch points or conditional memory watch points require processing every time each MPI task touches a page in memory containing the target memory address as most implementations use page table (or similar) mechanism to trap memory references with little performance impact on memory operations on watched pages. However upon this hardware page trap, the debugger must then determine if the memory address referenced in the page is the one being monitored or not and check to see if the condition is met, if there is one. This processing typically is today serialized back to the debugger process running on the Login node and interacting with the user. This is not scalable. With the hierarchal infrastructure, the debugger daemon running on each of the ION can process all of the page faults from MPI tasks on the CN under its dominion. This process runs in parallel across all the ION. As the job grows, so does the number of ION associated with it and the method describe is thereby scalable.

The Scalability Dilemma for Exascale Systems Has at Least Two Horns

Although significant research needs to be done on system scalability for Exascale systems, it is clear that a hierarchal system model, possibly with multiple levels in the hierarchy, is at least an intermediate step or starting point for research activities. The second horn of the Exascale systems scalability dilemma is that if PVIS systems drift too far away from where Linux clusters are, then the pyramid investment model in Figure 1 breaks down. It breaks down because more and more specialized technology will have to be developed for the PVIS and less and less leverage is obtained from lower levels in the pyramid. Thus we need to keep Linux clusters scaling up to petascale and beyond as we push PVIS systems technology to the Exascale. Thus the march to Exascale must be two pronged: scale PVIS to Exascale and COTS Linux clusters to petascale and beyond.

Current Flat Linux Cluster System Model

To understand the gaps here for Linux clusters it is instructive to review the current state of the art in Linux cluster design and deployment methodology.

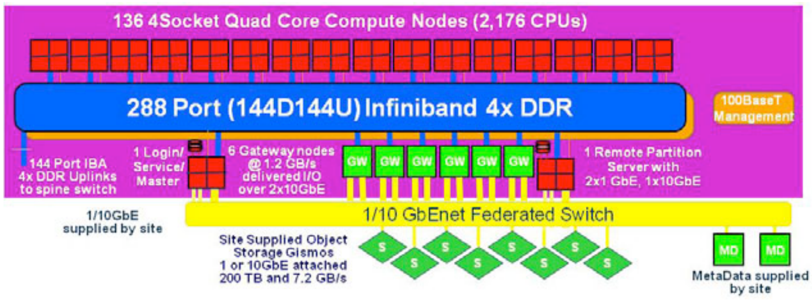


Figure 3: Linux Clusters of various sizes can be economically built from a Scalable Unit concept.

Recent advances in design indicate that multiple Linux clusters can be more economically built, integrated and operated by adopting a Scalable Unit (SU) design methodology. These and other Linux cluster designs in common use today essentially present a "flat system" model. SU are small aggregates of nodes that contain all the essential elements and node types necessary to build Linux clusters of various sizes: even vary large ones. In Figure 3, a SU design based on the 288 port IBA 4x DDR switch is depicted. The preponderance of nodes are CN as these are where the user MPI based applications run. The remaining nodes perform systems (and Login) functions and hence are kept minimal. In this SU design, we have the minimum of Login Nodes (LN at 1) and Remote Partition Servers (RPS at 1) and a few gateway nodes (GW at 4) necessary to provide sufficient IO bandwidth for applications running on the cluster over a SAN to the Lustre (or other) global (accessible multiple Linux clusters), parallel (supporting parallel IO within a cluster) file system. When building Linux clusters of various sizes the system functions also grow linearly and scale appropriately. For example, the RPS remote boots all the diskless nodes in the cluster (CN and GW) and serve up root and swap partitions for each node. Since this function is replicated independently in each SU these services scale with system size. For large clusters all one needs to add is a way to configure multiple RPS nodes in parallel from a single management workstation attached to the cluster over a management Ethernet.

Proposed Model

From the above discussion, we notice that a slight tweak on the Linux cluster "flat system model" based on SU design point can yield a hierarchal system model and offer the potential to scale Linux clusters to 10K-100K nodes. It turns out, from the hardware side, only a slight shift is necessary:

1. Design and build compute node as simple as possible (KISS)
2. Use gateway nodes as ION
3. Use RPS nodes as cluster of service nodes

The with recent advances in microprocessor design (e.g., including memory controllers and memory buses directly on the processor) and the tendency of the industry to aggregate more function onto the processor with time, it is possible to envision a very simple node design and a path to get there quickly.

On the software side a moderate shift is necessary to bridge the gap:

1. Light weight (low noise) Kernel
2. Function shipping interface to ION
3. All system services off of ION, only minimal job launch on CN
4. Debugging and process manipulation interface on ION to CN processes
5. Distributed RAS DB and infrastructure

Filling this gap will require significant effort by the Linux community. However, there has been a lot of research and development out of DOE SciDAC (e.g., FASTOS effort) that can be harvested. In addition, many vendors have indicated a willingness to commercialize such a model for the community.

This would be a good example of how we can change the industry by utilizing the R&D➔D&E➔Commercialization mechanisms described in a companion white paper titled "A Collaboration and Commercialization Model for Exascale Software Research."

MAIN PRINCIPLES

1. HPC pyramid investment model requires we pull up the rest of the pyramid while pushing to exascale or the model breaks down.
2. Hierarchal systems model developed for petascale systems is a good starting point, with possibly more than one level in the hierarchy, for exascale systems research
3. The current "Flat" Linux cluster systems model can be turned into a hierarchal systems model and scale up to 10K to 100K nodes.
4. A change to both hardware (simpler compute nodes) and software are required.
5. We can mine existing petascale systems efforts and combine it with readily available commercialization paths.

IESP Whitepaper: PDE-based applications and solvers at extreme scale

David Keyes
Columbia University & SciDAC TOPS project

The thirst for extreme floating-point processing rates is unquenchable in the foreseeable future, being driven by the need for: (1) better resolving the full ranges of length or time scales in multiscale phenomena, (2) accommodating physical effects with greater fidelity, (3) allowing the model degrees of freedom in all relevant dimensions, (4) better isolating artificial boundary conditions in PDE models and better approaching realistic levels of dilution in particle models, (5) optimizing or controlling physical scenarios (by solving inverse problems) once they are adequately resolved by forward models, (6) quantifying uncertainty, and (7) improving statistical estimates. As applications stretch to take full advantage of extreme architectures, however, the computational complexity of some algorithms, such as Courant-stability-limited explicit solvers as well as some linear and nonlinear solvers, grows superlinearly in memory size, making it impossible to weak scale, even though memory capacity would seem to allow it. Extreme scales put a premium on finding "optimal" algorithms, whose complexity is at worst log-linear in problem size; any suboptimal component will ultimately dominate the execution profile. In fact, to justify the acquisition and operating costs of exascale hardware, one needs to be concerned not only with complexity exponents, but also with the coefficients in front of the power laws, which can vary considerably from one formulation to another. *The availability of high capability architecture makes algorithms more, not less, important.*

Fortunately, algorithms such as linear solvers have kept pace with extreme scales, and optimal versions are known for many PDE-based formulations of driving applications. Therefore modelers who can cast their simulations in terms of these formulations (*e.g.*, sequences of Poisson solves to build up a preconditioner for a multicomponent system of more general type) may weak scale to 10^5 processor cores today, on a massively parallel computer with a log-diameter network. The logarithm, if it does not also arise from other causes, is a consequence of the global reduction operations that are present in Newton, Krylov, and other algorithms and *ultimately* degrades the marginal effectiveness of additional processor-memory elements if the synchronization stranglehold is not deferred by reducing its frequency. Furthermore, the marginal effectiveness of additional processors dividing the bandwidth of a memory shared among many processors may be nearly zero in many sparse algorithmic kernels.

As a further threat to effective use of extreme scale hardware, we note that progressive, mathematically beneficial trends in algorithms, such as increased use of unstructured meshes and adaptive discretizations that yield more accuracy per degree of freedom stored or flop performed at the expense of increased indirection, more conditionals, or more integer operations per flop, inveigh against the uniformity and predictability that are required to obtain maximum use of the floating point hardware. Traditional performance metrics focusing on floating point rates *only* in highly unbalanced hardware have long ceased, in general, to be reliable guides to the merits of a numerical computation. Instead, performance optimizers should hunt for each successive bottleneck – whether bandwidth, latency, number of integer load/store units, or whatever – and ask what algorithmic alternative could relieve it by exploiting unused capacity in some other hardware resource.

Solvers are just one of many algorithms that must scale. Tools for managing meshes, fields, and particles, *e.g.,* their generation, partitioning, adaptation, interpolation, and for constructing of the discrete equations from the underlying models must all be scalable, as well, or Amdahl's Law will impose a limit to scalability that is asymptotically independent of process granularity. The algorithmic techniques required to support simulations of interest at extreme scales include CAD-to-mesh geometric adaptivity, solution-based adaptivity, mesh partitioning, discretizations of virtually all types (with attention to advanced high-order discretizations), contact-detection algorithms, optimal implicit solvers, stiff method-of-lines integrators, kinetic and particle methods, unconstrained and constrained optimization (for parameter identification, control, design, etc.), sensitivity analysis (statistics- and derivatives-based), and uncertainty quantification. Extreme-scale simulation represents an opportunity for developers of the enabling technologies in applied mathematics and computer science to demonstrate a paradigmatic shift that they have envisioned for years as completely new application codes are written. The connective and control code and the majority of the means of interchange of data between code components will have to be rewritten together with algorithmic kernels take advantage of modern software practices and high-performance architectures. Virtually all large-scale data structures in existing codes will have to be replaced with distributed versions. In simulations at extreme scales, no data structure whose size scales with the system can be relegated to just one processor-memory element or replicated on each. As the software infrastructure is rebuilt, due attention can be given to extensibility, reusability, object orientation, componentization, portability, performance portability and tuning, code self-description and self-monitoring, and the construction of multi-layered interfaces that enforce correct usage.

Beyond these improvements that are occasioned by extreme scales (though valuable at any scale) the synchronizations that are built into most codes as matters of convenience in programming model must be drastically reduced. New algorithms and new programming models must be found that postpone synchronizations as long as possible. One class of trade-offs that is well developed requires more memory and more nearest-neighbor communication, which in turn allow many relaxation sweeps or Krylov steps to be conducted per synchronization. Another class of trade-offs hierarchically decomposes an implicit solve that involves all degrees of freedom globally into a set of infrequently communicating local implicit solves, with frequent synchronization within the local basins only. Such algorithms are known and are in some nonlinear problems actually demonstrably faster than their globally synchronizing counterparts, though they might in general be expected to be slower. However, full exploitation of asynchronous algorithms requires programming scientific applications much like operating systems, with different priorities assigned to different tasks, depending upon whether they are on or off the critical path, and with data-driven associative communication between them. The SPMD bulk synchronous model that is so convenient to understanding large-scale simulations will have to yield to far more general constructs that are less reproducible and likely far more difficult to verify for correctness and to predict for performance.

Prof. A. E. Trefethen, University of Oxford
Prof. N. J. Higham, University of Manchester
Prof. I. S. Duff, Rutherford Appleton Laboratory
Prof. P. V. Coveney, University College London

Developing a high performance computing/numerical analysis roadmap

Overview

A Roadmap Activity in the UK has leveraged US and European efforts for identifying the challenges and barriers in the development of high-performance computing algorithms and software. The activity has identified the Grand Challenge to provide:

- Algorithms and software that application developers can reuse in the form of high-quality, high performance, sustained software components, libraries and modules
- a community environment that allows the sharing of software, communication of interdisciplinary knowledge, and the development of appropriate skills.

Through a series of workshops and discussions with UK HPC application groups and numerical analysts five areas of challenge have emerged.

HPC-NA Roadmap Themes

Cultural

a. Identify potential community players
b. Develop models of community sharing
c. Provide community activities, workshops, training, virtual meeting spaces.
d. Engage internationally

Applications and Algorithms

a. Identify exemplar applications
    i. Develop baseline models for communication and benchmarking
b. Develop map of algorithms across application domain
    i. Indentify impact of specific algorithm development across discipline groups
    ii. Speed dating
    iii. Take mapping of dwarfs on capability computing
c. Develop map of developments internationally
    i. Collect information about ongoing related activities
    ii. Discuss with international funding agencies plans

Software

a. Abstractions (in collaboration with CS)
b. Code generation and adaptive software systems
c. Guidance on best practice for software engineering development
d. Develop frameworks and tools for application developers
e. Languages = take note of the DOE funded activities.

Sustainability

a. Develop models for sustainable software
    i. Long term funding
    ii. Industrial translation
    iii. Open community support
    iv. Other

Prof. A. E. Trefethen, University of Oxford
Prof. N. J. Higham, University of Manchester
Prof. I. S. Duff, Rutherford Appleton Laboratory
Prof. P. V. Coveney, University College London

b. Creation of MSC and other postgraduate training

Knowledge Base

a. Develop mechanisms for collecting information on existing software and dissemination
b. Develop mechanism for continuing community input
c. Education and training –
    i. Optimization for example
    ii. Software engineering
    iii. Provide computational science internships
    iv. Bid for short courses or summer schools

The activity is continuing in the UK to put more measurable priorities on the components in the evolving roadmap. Details can be found at http://www.oerc.ox.ac.uk/research/hpc-na.

Performance at Exascale
Bernd Mohr (Jülich Supercomputing Centre) and Matthias S. Mueller (Wolfgang E. Nagel Center for Information Services and HPC)

Resource Management
Barney McCabe (ORNL) and Hugo Falter (ParTec)

Programmability Issues
Vivek Sarkar (Rice U.), Jesus Labarta (UPC), Mitsuhisa Sato (U. of Tsukuba), Barbara Chapman (U. of Houston)

Models of Computation – Enabling Exascale
Thomas Sterling, Louisiana State University

Major Computer Science Challenges at Exascale
Al Geist (ORNL) and Robert Lucas (ISI)

Co-design of Architectures and Algorithms
Al Geist (ORNL) and Sudip Dosanjh (SNL)

IESP Exascale Challenge: Resilience and Fault Tolerance
Al Geist (ORNL) and Franck Cappello (INRIA)

Performance at Exascale

Bernd Mohr
Jülich Supercomputing Centre
b.mohr@fz-juelich.de

Matthias S. Mueller, Wolfgang E. Nagel
Center for Information Services and HPC
{matthias.mueller,wolfgang.nagel}@tu-dresden.de

Introduction

Exascale systems will consist of complex configurations with a huge number of potentially heterogeneous components. Deep software hierarchies of large, complex software components will be required to make use of such systems. While the software layers are designed to be transparent, they are typically not transparent with respect to performance. This *performance intransparency* will result in escalation of unforeseen problems to higher layers, including the application. This is not a really new problem, but certain properties of an exascale system significantly increase its severity and significance.

- At this scale, there always will be failing components in the system with a large impact on performance. A "real-world" application will never run on the exact same configuration twice.
- Load balancing issues limit the success even on moderately parallel systems, and the challenge of locality will become another severe issue which has to be addressed by appropriate mechanisms and tools.
- Dynamic power management, e.g., at hardware level inside a CPU, will result in performance variability between cores and across different runs. The alternative to run at lower speed without dynamic power adjustments may not be an option in the future.
- The unknown expectation of the application performance at exascale will make it difficult to detect a performance problem if it is escalated undetected to the application level.
- The ever growing higher integration of components into a single chip and the use of more and more hardware accelerators makes it more difficult to monitor application performance and move performance data out of the system unless special hardware support will be integrated into future systems.

Altogether this will require a integrated and collaborative approach to handle performance issues and correctly detect and analyze performance problems.

Performance Analysis

A large number of approaches for performance analysis exist that have successfully applied at small and medium scale. The large amount of performance data may seem to impede the use at exascale. However, this is not the case as long as features like memory size and I/O capabilities scale with compute power. An instrumented application is nothing but an application with modified demands on the system executing it. This makes current approaches for performance analysis still feasible in the future as long as all involved software components are parallel and scalable. In addition to increased scalability techniques like automatic analysis, advanced filtering techniques, on-line monitoring, clustering and analysis as well as data mining will be of increased importance. A combination of various techniques will have to be applied. The following considerations are key for a successful approach to performance at exascale:

- Failover or more general the operation with failed components should be performance neutral.
- An exascale system has to be capable to monitor the performance of components, not just the functionality.
- Hardware and software components need to provide sufficient performance details for analysis if a performance problem unexpectedly escalates to higher levels.
- Metrics beyond FLOPs need to be developed to identify and quantify performance problems, to measure the sustained performance and the gap to the attainable peak performance.
- Programming models should be designed with performance analysis in mind. Part of that could be a (standardized) hidden control mechanism in the runtime system that will be able to dynamically control – in time and space – the generation of performance data if requested.
- Performance analysis in the presence of "noise" requires inclusion of appropriate statistical descriptions.
- Performance analysis needs to incorporate techniques from the areas of signal processing and data mining.

DRAFT

Resource Management

Barney McCabe (ORNL) and Hugo Falter (ParTec)

A ***scalable application*** is an application whose performance scales with the size of the computing system. To be scalable an application must make effective use of additional resources, i.e., the application must demonstrate a performance improvement that is proportional to an increase in resources. This improvement can be demonstrated by reducing the time to completion for a fixed size problem (strong scaling) or by increasing the size of the problem that can be completed in the same amount of time (weak scaling). Alternately, a scalable application can be characterized as an application whose performance is constrained by the availability of one or more resources, i.e., a scalable application is a ***resource constrained application***. Ultimately, application scalability is based to the ability of the application to manage the resources provided by the computing system.

By presenting an abstraction of a computing system, ***programming models*** emphasize the management of some resources while de-emphasizing others. Successful HPC programming models emphasize the management of the resources that are most likely to constrain the scalability of an application, while de-emphasizing the management of other resources. For example, explicit message passing models, like MPI, have been very successful in HPC because they abstract the details of inter-node communication, but emphasize the management of distributed of memory by requiring that applications encode explicit message exchanges to access remote memory.

Approaches to resource management can be categorized in two dimensions: static/dynamic and explicit/implicit. Static resource management decisions are made before execution, while dynamic decisions are made during execution. Dynamic decisions typically incur some overhead (additional use of resources) during execution but they can incorporate information about the dynamic behavior of the program. Explicit resource management decisions are written into the code for the application, while implicit decisions are implemented in the translation or runtime system. Programming models emphasize the management of some resources over others by choosing which resources require explicit management by the application developer and which can be delegated to implicit management by the underlying runtime system.

Table 1. Approaches to Resource Management

	Static	**Dynamic**
Explicit	Algorithms	Zoltan load balancing
Implicit	Register allocation by a compiler	Demand-paged virtual memory

The tradeoffs between static and dynamic approaches in resource management are relatively straightforward to evaluate. Dynamic approaches can be justified when the overhead needed to monitor resource usage and to adjust the management of these resources results in an overall improvement in application performance. These justifications are typically complicated by the fact that the costs and benefits are highly application dependent and the fact that the overhead may require a resource that is different from the resource used to measure performance improvement, e.g., the overhead uses memory and performance is measured in time to completion.

Evaluating the tradeoffs between explicit and implicit approaches is rarely straightforward. Implicit resource management decisions remove much of the burden for making resource management decisions from the programmer (moving this complexity to the runtime system) and may enhance application portability, because details regarding resources of the target platform do not need to be encoded in the application. However, because implicit approaches seek to hide the true nature of the resource, there is a chance that application developers will unknowingly use the resource in an inappropriate fashion. A simple example of this comes when programmers fail to maintain temporal locality in their data access, yielding poor virtual memory or cache performance when the existence of these mechanisms is not explicit in the programming model.

No implicit resource management strategy is ideal for all applications. There is a significant chance that any implicit resource management decision will adversely affect the scalability of an important application. In most cases, the critical resource management decisions are limited to a small portion of the application and most of the application code does not need to include explicit resource management decisions. For this reason, it is important that implementations of programming models provide programmers with the tools needed to "opt out" of the implicit management decisions as needed. As an example, compilers for procedural programming languages provide implicit management of the registers available on a CPU. Using profiling tools, application programmer can identify performance critical parts of their code and, if needed, hand code specific subroutines in assembly code, opting out of the implicit management of CPU registers provided by the compiler. Providing mechanisms to opt out of dynamic, implicit resource management decisions is typically more difficult. In the past, this has been addressed by providing hints and callbacks. Hints allow the programmer to provide explicit advice to the runtime system in advance. The runtime system uses the hints provided by the programmer to guide its management of the resources. Callbacks allow programmers to register handlers that implement explicit resource management strategies.

For the past two decades, high performance computing (HPC) has focused on increases in processing resources; although, there is general recognition that balanced increases in other resources (e.g., memory, storage, and inter-processor communication) may critically impact the ability of an application to take advantage of increases in processor resources. As we enter a time in which processor cycles are ubiquitous, the processor is unlikely to be the resource which critically constrains the performance of an application. As such, we, as a community should take this opportunity to re-consider the tools and approaches available to application developers to support them in the management of resources for scalable applications.

Programmability Issues

Vivek Sarkar (Rice U.), Jesus Labarta (UPC), Mitsuhisa Sato (U. of Tsukuba),
Barbara Chapman (U. of Houston)

Programming models are central to our effort to address the exascale challenge. They are the key interface that will allow the separation of the programmers' concerns from those of system designers, potentially at different levels of granularity. Any such model must meet the extensive needs of application developers and be supported by the entire software stack. The programming and execution model interfaces are key to allowing programmers to focus on their algorithms while providing the mechanisms that will enable the compilers and run times to infer the information they need to optimize, automatically and dynamically, the use of system resources (cores, memory, bandwidth, power). Considerable research is needed to define and implement the programming and execution models for such systems. Whereas evolutionary approaches may best support the migration of existing application software, revolutionary models may be best suited to providing extreme-scale performance for new applications on emerging architectures. Both approaches should be explored.

Desirable properties of exascale programming models include the following:

- They should provide highest levels of **performance**. Most HPC programs are written for performance. Moreover, exascale programming languages should be **performance-aware**: they should provide an adequate abstraction of high performance parallel hardware platforms to enable the exploitation of their features, and some means to tune performance. The failure of automatically parallelizing compilers and HPF was caused not only by technical immaturity but also by a lack of an interface in the programming language for performance improvement. When the programmer finds a performance bug, he or she should have some mean to improve performance by modifying the program. The model should provide the necessary interfaces to allow tools (especially performance tools) to obtain information on the application's execution behavior.
- **Expressivity** is a key requirement. Exascale programming languages should provide a model and an interface to express the parallelism in programs. In functional programming languages and "old" dataflow languages, parallelism is implicit since the model of computation itself exploits the parallelism. In imperative languages, new constructs and mechanisms should be introduced to express the parallelism. From the application points of view, task parallelism must be able to support coupled multi-physics simulations at several levels for exascale systems. Applications will need to express massive amounts of potentially fine-grain parallelism, of asynchrony and locality. Dynamic application behavior will need to be supported. It should be possible to express hierarchical parallelism within the application. Latency hiding needs to be facilitated.
- They should enable **composability**. Composability is essential to support productive programming on exascale systems. Libraries and object-oriented approach help accomplish this in conventional sequential programming, but they

don't always work in parallel programming. For example, it is difficult to use parallel libraries with current OpenMP. Parallel object-oriented programming is sometimes useful, but has some problems.

- They should support **fault tolerance** and **error handling**. Fault tolerance is one of the most difficult issues faced on exascale systems. If faults are exposed to programmers, then some programming language support will be required to handle them. It must moreover be possible for an application to respond to faults and program errors gracefully rather than simply crashing.
- They need to support **massively parallel I/O**. An abstraction of I/O, including the file system, may help programmers handle the huge amounts of data that will have to be read and written.

Approaches to programming exascale systems should take the following into account:

- There is a need to provide a **smooth transition path** from existing practices and codes to future approaches. **Programming environments** will be needed that support this transition, as well as all phases of application development and tuning on exascale architectures under new and enhanced programming models.
- Approaches should provide **portability (functional and performance)** across platforms such that the porting effort can be amortized over the foreseeable variation of systems to appear from now till the exaflop era and beyond.
- **Incremental** parallelization/tuning of applications is a desirable property closely related to the above two issues.
- Initial approaches should address the **device, node and system level** programming. Proposals for hybrid programming should ensure clean interaction between the different levels and ensure that the synchronization semantics and scheduling decisions at one level do not imply restrictions on other levels.

Topics for detailed study include:

- **Address space structure**. Identify abstract levels of a structure that is simple enough for use by a programmer to express objects/ideas yet allows the run time flexibility regarding its mapping to the potentially varied physical structure.
- Flexible **work generation** (parallelism/task specification) **and synchronization structures** beyond pure fork-join approaches in order to support flexible parallelism and **high levels of asynchrony**. Ideas from data flow or functional programming may be revisited and smoothly integrated into current practices.
- **Latency tolerance**, being able to specify required data accesses with large lookaheads such that implementations (compiler or run time) can anticipate the required data transfers and schedule them appropriately.
- The issue of **hierarchy and heterogeneity**, providing mechanisms for **modular** designs with interchangeable implementations of tasks.
- **Separation of functionality and performance**, providing mechanisms for the programmer to provide hints that may help satisfy performance or power requirements, but are not required to provide functionality of the algorithms.

- **Malleability,** the ability of applications to dynamically adapt to the available resources which may vary during a job run. Programming and execution models should support/promote malleable programming practices by separating (virtualizing) the algorithmic structure of a program from the resources where it is executed.
- **Error handling and fault tolerance.** Providing the appropriate hooks for resilient applications.
- **Application development environments** that facilitate the migration of current codes and/or the development of new ones from scratch.

The evolutionary path aims to adapt existing programming models to needs of exascale computing, and facilitate task of creating and tuning potentially hybrid application codes. This could include work to enhance MPI, OpenMP, CUDA/OpenCL or other approaches to programming accelerators and SIMD units, as well as work to improve their interoperability. It might also include more effort to deploy the PGAS languages and ensure that they may interoperate with other programming interfaces. A revolutionary path might be based upon HPCS languages or might be a completely new path. It might be worthwhile to revisit old parallel programming models and languages to obtain new insights from the past, as is being done in the architecture community. Functional programming models used to programming the dataflow machines, such as Id, SISAL, ... could be interesting to evaluate. HPF was a great effort to develop a standard parallel programming language and is also worthy of re-examination. It is important to take their experience of failure into account for better future developments.

In such an open field, it is advisable to pursue a few alternatives and ensure there is sufficient sharing of experiences as well as **comparative studies** between them. These should be in terms of complexity/readability of the code and programming effort as well as performance (both actual measurements on common platforms as well as predictions for different potential targets). Although these types of studies are often difficult to perform, special efforts should be devoted to that. **Common sets of algorithms** should be used for evaluation by all the proposed models.

Finally, we should promote efforts to develop standard APIs between several levels and components of existing software in the IESP community. For programmers and end-users, candidates for standardization will include:

- PGAS languages (UPC and CAF, …)
- Global views models such as Chapel and HPF

For the system developer, the candidates are:

- One-sided communication APIs
- Fault tolerant model and APIs
- API for I/O on massively parallel system
- API for accelerators
- Performance profile API and data format such as OTF
- API for thread scheduler

The standard development effort is a key to "evaluation" which develops the community.
It will be the basis for the next "revolution" of rich diversity for exascale computing.

Models of Computation – Enabling Exascale

Thomas Sterling
Louisiana State University
May 17, 2009

The derivation of new systems' software and tools for high performance computing environments at Exascale will demand realignment and adjustment of functionality and capability of software components to exploit the new opportunities and address the new challenges of future system architectures, which themselves will be created in response to advancing hardware technologies. The evolution of digital device technology, the most dramatic in the history of human technology, has catalyzed a sequence of architecture classes over the last six decades, each optimized to the specific properties of their respective emergent technology phase. Programming models and representative languages followed to best exploit the performance capability of the system hardware during each phase. Algorithms were devised to reflect the computational needs of the applications while constrained to the semantic constructs of the available APIs. This reactionary strategy is being replayed as HPC once again experiences a phase-change with the advent of heterogeneous multicore for ultra-high performance computing. However, this empirical random-walk methodology is time consuming, error prone, and costly due to its intrinsic lack of guiding principles to facilitate co-design of all system layers simultaneously. Such comprehensive principles comprise a paradigm or *model of computation* to which all layers comply and contribute to achieve overall system optimal behavior with respect to critical objective functions. Can we get ahead of the game to leapfrog the tedium of catch-up? Or putting it another way, can a model of computation be derived that will enable the development of Exascale computer systems through the co-design of its comprising system software (and architecture) layers? A brief discussion of the nature and characteristics of models of computation (alternatively, "execution models") is offered to contribute to the current community discussions on proceeding toward the realization of Exascale computing by the end of the next decade.

Prior HPC phase-changes included the:

- original sequential instruction operation,
- sequential instruction issue,
- vector,
- array,
- systolic (for SPDs), and
- the most recent communicating sequential processes (CSP).

Others such as dataflow and reduction models did not achieve commercial status although interesting experiments were performed. The multiple-thread/shared-memory model is concurrent with CSP for limited scale systems.

The current HPC phase-change is apparent by the forced deployment of heterogeneous multicore components to maintain the continued peak performance progression consistent with Moore’s Law and the underlying exponential growth in semiconductor device density. However, these structures are reactive to the combined pressures of power consumption, processor design complexity, and efficiency factors. They do not reflect a clear understanding of an underlying innovative execution model by which this combination of resources can be effectively employed for future applications. It is a subject of controversy as to whether incremental extensions to current methodologies (e.g., MPI) may serve this purpose. Four factors of the new phase suggest that incrementalism is a false hope even if it does adequately serve over the next three to five years with diminishing efficiency and scalability. These factors include:

1. > 1000X scalability gain with respect to current best levels
2. Power efficiency > 50 Gigaflops per watt,
3. Non-stop operation in the presence of single point failures, and
4. Support for efficient dynamic graph processing

Together these factors challenge conventional practices to:

a) Solve the multicore programming problem,
b) Reduce the ever increasing memory wall,
c) Expose and exploit billion-way parallelism,
d) Incorporate innate latency mitigation and hiding methods,
e) Reduce average energy per operation by two orders of magnitude,
f) Integrate memory-oriented operations for meta-data parallel computing,
g) Achieve fault tolerance through support at all levels,
h) Embrace dynamic adaptive resource management for runtime efficiency, load balancing, and reconfiguration (resiliency),
i) Exhibit a global address space,
j) Greatly increase efficiency of parallel control such as elimination of global barriers, lightweight task creation and context switching, and dynamic task migration, and
k) Permit heterogeneous cores to be optimally scheduled.

Other requirements may prevail as well but these are sufficient to demonstrate the inadequacy of common methods which over the prior decade and a half have resorted to static mapping of coarse grained parallelism to physical processes, avoiding latency rather than hiding it (noting some pre-fetch methods), overly constraining flow control by simplistic global barriers, forcing a distributed memory mind set, and forcing programmers to explicitly manage allocation of resources to data and tasks. No one layer of the system is sufficient to address any of these but multiple layers engaged synergistically implementing new strategies may do so. The model of computation provides the template for the patterns of execution to be accomplished each layer working in tandem with the others.

A model of computation describes how an abstract computation evolves on a physical machine successively altering the intermediate state of both to converge on a final solution. It defines the name spaces, the control semantics, the memory consistency model, the forms of parallelism that it may exploit, and potentially other attributes as well. It may define policy interfaces or invariants without specifying the actual specific policies themselves in order to provide flexibility in system implementation and application. Such policies might include scheduling methods and priorities, name space management, and means of achieving compound atomic operations for example.

There are multiple key consequences of adapting a model of computation to a new class of system hardware technologies. One is the verification through its existence and mapping of functionality requirements to hardware mechanisms that full and complex calculations can be performed on expected hardware designs. A second is that such a model simplifies overall system design. Without it, each layer of a system must be developed (assuming complete system design) in terms of every other layer; a order n-squared process. Adopting a model of computation only requires that each layer be defined in terms of its contributing functionality to realizing the shared model; basically an order n process. Even with iteration for convergent refinements and optimization, an execution model can greatly simplify the design process. A third value is that it does permit early experimentation with early algorithm and application kernels through the likely existence of a low-level application programming interface and test environments. While unlikely to provide absolute performance numbers, it will yield insight in to the utility of the control semantics of the model, and therefore future systems that employ it as a basis for hardware and software system design. And forth, such a model as has happened before, facilitates sharing and cooperation across disciplines and institutions.

How does a model of computation directly contribute to design concepts and decisions for the many combined layers of the system? Some examples, in no way comprehensive, are suggested:

- Application interface layer – the execution model defines the basic data organization, name space (shared or distributed), distributed communication semantics, and parallelism form and control. All these relate to the API and programming models that may be employed in constructing applications and libraries.

- Compiler layer – the model of computation combined with the system processors' ISAs and the previously defined API syntax determines the responsibilities in translation and analysis that is to be performed by the compiler. This includes invocation of runtime system functions and operating system service calls. The compiler will provide software implementation of software support mechanisms.
- Runtime system layer – A major effect of the model of computation is its definition of the functionality of the runtime system. This software is likely to grow in importance for new systems and will be heavily influenced by the model of computation determining how and what information about the runtime state will be exploited to manage tasks and resources. The runtime system will provide dynamic control, scheduling, allocation, and some synchronization of concurrent activities according to the underlying execution model.
- Operating system layer – the model of computation will determine what support it requires from the lower level system some of which will be provided by operating system services that must be provided.
- Architecture layer – For efficiency and scalability, the model of computation will require certain time critical mechanisms to be implemented at least in part in the hardware architecture to minimize overhead. Other architecture requirements driven by the model of computation include how to perform virtual to physical address translation, guaranteed compound atomic functions on data, and efficient communications.

Towards the establishment of the next generation model of computation, research is required to understand the driving requirements and to devise alternative solutions that will enable computing systems and methods for Exascale in the next decade. The above discussion has considered the general strategy and approach as well as the basic challenges that will guide the derivation of such a model of computation.

Whitepaper on the

Major Computer Science Challenges at Exascale[1]

February 2009

Al Geist, ORNL and Robert Lucas, ISI

Exascale systems will provide an unprecedented opportunity for science, one that will make it possible to use computation not only as a critical tool along with theory and experiment in understanding the behavior of the fundamental components of nature but also for critical advances for the nation's energy needs and security. To create exascale systems and software that will enable DOE to meet the science goals critical to the nation's energy, ecological sustainability, and global security, we must focus on major architecture, software, algorithm, and data challenges, and build on newly emerging programming environments. Only with this new infrastructure will applications be able to scale up to the required levels of parallelism and integrate technologies into complex coupled systems for real-world multidisciplinary modeling and simulation. Achieving this goal will likely involve a shift from current static approaches for application development and execution to a combination of new software tools, algorithms, and dynamically adaptive methods. Additionally, we must bring together new developments in system software, data management, analysis, and visualization to allow disparate data sources (both simulation and real-world) to be managed in order to guide research and to directly advance science. Achieving this vision will require fostering long-term, sustained, communitywide activity in evolving code suites. Large-scale applications, like large-scale computers themselves, require the support of multiple specialists within a single community. Indeed, the community of computer vendors, application scientists, and computer scientists, together with the hardware and software they both develop and use, form an integrated, interdependent ecosystem.

Several recent studies and workshops [1-10] have identified the high level problems facing the HPC community as it moves towards exascale over the next decade. This paper compiles and organizes the major software challenges into four categories:

- Problems caused by the growing scale and complexity of computer architectures
- Problems caused by the growing complexity of science applications, including the longstanding problems with debugging and tuning large applications at scale
- Problems are caused by the huge increase in the data produced and consumed by peta and exascale systems.
- Problems of software sustainability such as hardening and long-term support of popular software packages, education of the next generation of HPC specialists, and training the existing users about advanced techniques and tools.

These workshops pointed out that it is critical that work begin today if the DOE's scientific computing community is to be able to exploit exascale systems when the technology to create them matures in the coming decade.

1. Challenges due to scale and complexity of system

[1] Work in process. Based on an analysis of the computer science challenges from the DOE Exascale studies.

For most of the past five decades, the growing computational power of supercomputers has come primarily from a doubling of clock frequency every 18 months. In the last two decades, this has been augmented by an increase in the number of processors. Over this time period, the clock rate increased by six orders of magnitude, while the number of processors increased by three orders of magnitude. Due to constraints on heat and the power requirements of today's microprocessors, the last frequency doubling occurred about five years ago and has remained effectively constant ever since. Vendors have shifted to putting multiple processors (cores) on a chip; first two, then four, then eight. The number of cores per chip is expected to continue to increase exponentially over the next decade. Today's supercomputer vendors see the only way to continue increasing the computational power of their systems is through increasing the number of processors and hence the scale and complexity of their systems. In the last five years supercomputer architectures have gone from 1000 processors to 100,000 processors and the next generation systems are going to have over a million processors. The rate of growth of parallelism is in fact accelerating, and will likely exceed one hundred million when exascale systems appear. Some estimates even predict that the need for multiple threads to cover main memory and communication latency means that scientific codes will contain billions of threads.

The change of shifting from using faster processors to using multi-core processors is as disruptive to scientific software as the shift from vector to distributed memory supercomputers fifteen years ago. That change required complete restructuring of scientific application codes, which took years of effort. Some application communities still haven't transitioned to even a thousand-way parallelism. The shift to multi-core exascale systems will require applications to exploit million-way parallelism and overcome significant reductions in the bandwidth and volume of memory available to each CPU. This "scalability challenge" driven by the exponential increase in the amount of parallelism in the system affects all aspects of the use of high performance computing. It makes all the existing problems harder, such as getting performance from the applications, managing the system, debugging, etc.. It also creates new challenges such as fault tolerance, the need for new programming models, and verification of results.

There is another looming shift in the complexity of the node architectures that will be as big a challenge to software development as the exponential growth in processors. This is the potential shift to heterogeneous node architectures. Today most supercomputers are of huge scale but they are homogeneous. Over the next decade it is expected that the multi-core processors will include several different types of cores on each node, for example, a computation accelerator, a graphics processor, a communication processor, an IO processor, etc. An early example of a heterogeneous system is the Roadrunner supercomputer at LANL.

The major challenges caused by the increasing scale and complexity HPC systems are cross cutting of the entire software stack. The software challenges include the rapid increase in parallelism, the memory wall, system heterogeneity and fault tolerance. For each of these challenges computer science research is needed across the entire stack not just at one level. For example, making an application fault tolerant is not sufficient if the system software is not also fault tolerant. Making the system software fault tolerant is not sufficient if the data can be corrupted by faults in the data management software. To be able to use these systems to solve the nation's problems, DOE, as the pioneer in HPC, must improve all parts of the software stack and influence the architecture design to meet the scientific needs. The challenges impact both the developers and users of the system software, the applications, the runtime, communication, IO, and data management, including analyzing the results.

1.1 Increasing Parallelism

The increase of system concurrency from hundreds of thousands to hundreds of millions will be a tremendous challenge for system software to manage and for applications to get good performance at this level of parallelism. Almost all of today's large-scale applications use the message-passing programming model (MPI) together with traditional sequential languages (C, Fortran, C++), but new architectures with many cores per chip and parallelism in the millions are expected to make this programming model more problematic and less productive in the future. Thus new approaches are needed. For example, a hybrid programming model such as MPI with some global view techniques such as Unified Parallel C (UPC) or Co-Array Fortran (CAF). In order to facilitate the utilization of the extreme scale resources, new programming models and High Productivity Computer Systems (HPCS) languages must be explored.

1.2 Memory Wall

The memory wall traditionally refers to the challenge that the bandwidth and latency to memory continues to grow at a slower rate than the processor power. The transition from frequency-based scaling to core-based scaling will make the memory wall both higher and broader. It is higher in that bandwidth and latency continue to get worse as memory gets farther away from CPU operations (at least in terms of clocks). The memory wall is going to get broader in that the overall memory capacity per core must decrease. It will be harder and harder to maintain the desired byte-to-flop ratio—in absolute capacity (flops/s per byte) and bandwidth terms (flops per byte). Hence, applications will have to be redesigned to make better user of limited memory. Additionally, applications will have to deal with increasing hierarchies of memory (and indeed storage). There are now often five levels of direct access memory (register sets, three levels of cache, and main memory). In the future there may be more levels and more (or less) sharing of these levels within a shared memory node, as well as a new level of persistent FLASH to augment the DRAM main memory.

1.3 Influencing Architecture Design

DOE scientists have been pioneering users of high-end systems for over five decades. While the systems themselves are usually manufactured and deployed by computer system vendors, architecture research conducted by DOE scientists, often in collaboration with the vendors, allows DOE to develop the specifications for the systems. To maximize the utility of the computer hardware, DOE computer scientists often contribute everything from system software to programming environments and debugging tools. Recent examples abound, including BlueGene/L, Red Storm, and Roadrunner. As we look forward to exascale, high-end systems will become increasingly specialized, and DOE scientists will have to take an even more active role in designing of both the software and the hardware of such systems to assure their utility for the scientific problems that face the nation.

1.4 Heterogeneity

Heterogeneity exists at many different levels in modern supercomputers. The systems have several different node types: compute, IO, login; several different operating systems; and several different interconnection networks: RAS network, command network, one or more communication networks. Despite this heterogeneity, these systems are usually considered homogeneous because the fundamental compute node is homogeneous and replicated tens of thousands of times across the system. A heterogeneous system is one where there are regions of different compute nodes across the system. For example the proposed Japan 10 PF system is designed to be a mix of three different types of architectures: vector, cluster, and specialized (GRAPE). Another form of heterogeneous system is where the compute nodes are heterogeneous. An example is the "Roadrunner" system where each compute node has a traditional AMD multi-core processor plus an IBM Cell processor, originally designed for the Sony Playstation. The major chip vendors have all started exploring creating heterogeneous multi-core chips that combine light-weight, high compute density processor units (e.g., GPUs) and traditional computational units (CPUs) in order to increase the computational power on a single chip. It is expected that over the next decade most supercomputers will be constructed using such heterogeneous multi-core processors.

Heterogeneity is also appearing at the system level, as computer centers adopt a "crop rotation" model, whereby systems are partially updated on a regular basis. A recent example occurred at the ORNL

Leadership Computing Facility, when two generations of Cray XT systems were simultaneously deployed.

Heterogeneity is a radical shift from today's environment. System management, job scheduling, efficient resource utilization, and load balancing all become much more complex. Today's code development assumes a homogenous run-time environment, with parallelization being done manually by each code developer. At the scale where applications need to make use of millions of heterogeneous processes, discovering the opportunities for parallelization becomes much more difficult and requires a set of tools that can automate the parallelization of the trivially parallelizable segments of code, and aid the application developer in finding less obvious opportunities. This task is even more daunting when considering future heterogeneous multi-core architectures, since the parallelization algorithms have to take into account the different types of processors and the interactions between them. Compiler research will be needed to understand how to exploit heterogeneous hardware, automating as much of this as possible and providing code-restructuring assistance where automation is not possible.

1.4 Fault Tolerance

Modern PCs may run for weeks without rebooting and more data servers are expected to run for years. However, because of their scale and complexity, today's supercomputers run for only a few days before rebooting. Exascale systems will be even more complex and have millions of processors in them. The major challenge in fault tolerance is that faults in extreme scale systems will be continuous rather than an exceptional event. This requires a major shift from today's software infrastructure. Every part of the exascale software ecosystem has to be able to cope with frequent faults; otherwise applications will not be able to run to completion. The system software must be designed to detect and adapt to frequent failure of hardware and software components. On today's supercomputers every failure, even ones that get reconfigured around, kills the application running on the affected resources. These applications have to be restarted from the beginning or from their last checkpoint. The checkpoint/restart technique will not be an effective way to utilize exascale systems, because checkpointing stresses the I/O system and restarting kills 999,999 running tasks because 1 fails in a million task application. With the potential that exascale systems will be having constant failures somewhere across the system, application software isn't going to be able to rely on checkpointing to cope with faults. A new fault will occur before the application could be restarted, causing the application to get stuck in a state of constantly being restarted. For exascale systems, new fault tolerance paradigms will need to be developed and integrated into both existing and new applications.

To complicate matters even more, the GPU accelerators that are being considered for heterogeneous systems often do not have any error checking on the processors or in their memories. This is because there is no market force to require error checking since a few incorrect pixels on the frame of an animation is not noticeable. But if GPUs become common in peta and exascale systems then undetected errors from GPUs or other sources could dramatically increase the rate of faults in large systems.

Research in the reliability and robustness of exascale systems for running large simulations is critical to the effective use of these systems. New paradigms must be developed for handling faults within both the system software and user applications. Equally important are new approaches for integrating detection algorithms in both the hardware and software and new techniques to help simulations adapt to faults.

2. Challenges due to complexity of applications

As computational capabilities have grown, so have the resolution and complexity of the simulation models. The large simulation codes today incorporate multidiscipline, multi-physics, multiple time scale and multiple solution methods. They have taken years to develop by teams of programmers and scientists and can include millions of lines of code. As we make the leap to exascale computation the impact on the cost to update, recode, and incorporate more advanced models into the simulations can be an order of magnitude higher than the cost of the supercomputer hardware. In order to contain these costs, the exascale software ecosystem must support more efficient program development that addresses the following application challenges:

- Scaling limitations of present algorithms
- Innovative algorithms for multi-core, heterogeneous nodes
- Software strategies to mitigate high memory latencies
- Hierarchical algorithms to deal with BW across the memory hierarchy
- Need for automated fault tolerance, performance analysis, and verification
- More complex multi-physics requires large memory per node
- Model coupling for more realistic physical processes
- Dynamic memory access patterns of data intensive applications
- Scalable IO for mining of experimental and simulation data

The user requirements are heavily shaped by the length of the life cycle of the applications. HPC applications have both long development cycles and long periods during which the application is in "production." An important aspect of this life cycle is that code is always in development -- even production code. Thus, the users require assurances of stable support for a programming model, including the development tools that enable its use. Further, "new" applications are almost never entirely new—they almost always take some existing code base to provide key underlying physics or mathematics functionality from an existing application. As a result, users are not open to tools that only target "new" applications or require significant changes to the established workflow of the application team.

Applications are becoming much more multifaceted as teams include a variety of languages, libraries, programming models, data structures, and algorithms in a single application. In fact, application teams are listing scalable tools for debugging, memory correctness, thread correctness, and multimode performance analysis as key factors in their productivity.

Today's tools are limited in scope, capability, and scalability. The overhead associated with current measurement techniques is too intrusive at the petascale and may skew analysis so much as to render any analysis ineffective. Therefore, we need to develop scalable and less intrusive methods of collecting performance data, develop knowledge discovery methods for extracting key performance features, and provide assistance in feeding the results of these analyses back to the code transformation.

2.1 Improving Programmability

Exascale computer architectures will require radical changes to the software used to operate them and the applications that run on them.

New ways of specifying computations: Scientists must be freed from the details of managing data movement among memory systems and synchronizing access to shared memory among threads of control. They will need languages and libraries, in some cases discipline- or even application-specific, which specify results to be obtained with less attention to the details of the computation than is currently necessary. Implementation of such libraries and languages will require lower-level programming models and tools that permit execution on a wide range of hardware and exploit the capabilities of exascale architectures.

Portability: Libraries are the typical software test beds where new programming models and execution models are proved out and this will continue to be the case. Numerical and communication libraries provide a fast vehicle for getting the new concepts into use by the application developers. MPI is the portable programming model today. Any new programming model that is created must be, at a minimum, be as portable across the key HPC systems, clusters, and development platforms of the time to be adopted by software developers. Tools to assist in the code transformations to new models and new algorithms are going to be critical in transitioning the millions of lines of code to a new programming model.

Huge code size: The increasing prevalence of coupled multi-disciplinary codes has combined with the long life cycle of scientific applications and the use of third party libraries to make codes larger and more complex. As a result, tools must handle larger executables. Tool developers are already seeing demand for tools to handle codes of several hundred mega-bytes to giga-bytes of executables. In addition, the rise of component based programming is resulting in applications that have hundreds if not thousands of shared libraries. So tools must be developed that handle both huge binary files as well as large numbers of files.

The memory challenge states that the memory per core in petascale architectures is going to decrease. Developers need tools that will help them understand the scaling behavior of memory allocations and usage as well as detect correct memory semantics. With limited node memory, tools that monitor how much memory is being used in a parallel job over time would also be useful. To be applied at extreme scale, all of these tools must have little overhead, a criteria that many existing memory correctness tools fail to meet.

Tools throughout software life cycle: The tool needs vary with the life cycle stage. Initial code developers need full featured debuggers and performance analysis tools and are willing to work with tools with relatively high overheads, such as some memory correctness tools. Similar functionality is also needed for code being maintained. In addition, support for version tracking, code coverage and regression testing (both correctness and performance) are useful at this stage. Supporting code ready to run at large scale requires yet different tools. Lightweight debugging functionality is essential at these scales, as are low overhead mechanisms for performance profiling and analysis. Codes in production need workflow tools to interact with applications and large scale systems. Finally, tools to support fault tolerance, with a focus on data integrity, are expected to become even more important during this life cycle stage as faults become continuous.

2.2 Building New Applications

The vision for the next decade is to have a totally integrated approach to how applications are built, modified, updated, and used in other applications. In such a development environment the tools will interoperate with each other and assist the scientists in writing, debugging, tuning, and maintaining their codes. This will be facilitated through:

Rapid, modular construction of new applications from existing suites of interoperable components. Scientific software components with well-defined interfaces have the potential to greatly increase code reuse, thus shortening development times and increasing software reliability.

Coupling of multiple applications into ever-larger applications through automated workflows. Single large runs remain an important class of large-scale computations, but many applications need parameter studies consisting of large numbers of coordinated sets of runs, each perhaps consistent of a pipeline of computation and analyses. High-level, standard languages for coordinating such families of executions will enable scientists to focus on science rather than "run management."

Debugging Tools. An integral part of application development includes verifying that code runs as expected. Current debuggers are not able to handle even a few thousand tasks much less the 100,000 tasks on today's supercomputers. Application developers for today's large systems fall back to the very inefficient method of debugging—dumping user inserted debug code to output files. With the vast increase of process count going to exascale systems, searching manually for a single anomalous process among the millions of running processes and threads is not tenable.

Application teams need tools for managing application builds and configurations, mixed language support, dynamic linking, program configurations, remote access, compiler infrastructures for application-specific analysis and transformations, and integrated development environments. Application teams specifically request lightweight tools to diagnose memory, threading, and message passing errors that are easy to use and scale from the desktop system to the petaflop platform. Furthermore, the architectures and system software must make the necessary performance and reliability information available to these tools so that they can perform root-cause analysis with greater accuracy.

For performance and correctness tools the availability of scalable tools is particularly critical. These tools require a scalable infrastructure to provide tool communication, data management, binary manipulation of application executables, execution management for batch schedulers and operating systems, and a variety of other capabilities. Tool infrastructures must be efficient, modular, fault tolerant, and flexible.

2.3 Execution Environment

Managing a system with a million processors and faults occurring almost continuously produces new challenges for system software. Efficient scheduling and resource management become significantly harder with a dynamically changing configuration as does upgrading and monitoring. The acceleration in scale puts additional pressure on the scaling of all system software components. In particular, OS scaling has been a historical challenge at each change in scale. Several performance issues are anticipated to become of increasing importance. Perhaps at the top of the list is load balancing. Tools are needed to detect load balance problems and to assist the dynamic load balancing of applications.

In order to guide research, and to directly advance science there must be a more flexible and dynamic resource management capability throughout the computing environment to allow computing, analysis, visualization, and live data to be integrated simultaneously during a simulation. While workflows provide a nice execution interface for the scientist, they will need to evolve to meet the needs of the growing complexity of applications.

a) **Semantic awareness in workflows**: Workflows need intelligence to identify which actors can be linked technically, highlight mismatch of units between actors, enable better control over parameter sweeps, and learn from previous workflows.
b) **Optimization of workflows:** Currently workflows are driven by the need to optimize the scientist's time. In the future they may also need to consider other options such as optimizing power, CPU cycles and data transmission time by dynamically scheduling on appropriate systems. This will require that the workflows be aware of the underlying hardware.
c) **End-to-end software environments to support collaborative data analysis:** As advances in mathematics and computer science make the analysis of larger and more complex data sets feasible, it is also necessary to bring these advances together in an environment that supports the end-to-end process of data analysis from the initial data to the final results. This environment should include support for workflows, provenance, and storage of data.
d) **Incorporation of policies:** Workflows will also need to incorporate any privacy and security policies that may dictate how and what data can be analyzed.

2.4 Validation and Verification

The scale and complexity of the science problems enabled by exascale systems require new techniques for making sure that the calculations are done correctly. It will be increasingly important to validate that new extreme scale algorithms are solving the right problem and to verify that the answer produced is correct and not corrupted by numerical stability or numerical errors from transient non-fatal faults.

The difficulties in drawing scientifically-meaningful conclusions from vast volumes of data is reflected in the greater need for code validation, uncertainty quantification, and the analysis of data across ensembles of simulations. Often, the quality of the data, and the variation in the data, add to the challenges resulting from the massive size of the data, thus increasing the need for robust algorithms that are not sensitive to the settings of parameters.

3. Challenges due to increased data

The data challenges include dealing with the volume, different formats, transfer rates, analysis, and visualization of massive (potentially distributed) data sets. Exascale applications running on as many as a million processors are likely to generate data at a rate of several terabytes per second (even assuming only a few megabytes per processor). It is not practical to store raw data generated at such a rate. Dynamic reduction of the data by summarization, subset selection, and more sophisticated dynamic pattern identification methods will be necessary to reduce the volume of data. And the reduced data volume will have to be stored at the same rate as it is generated, in order for the exascale computation to progress without interruption.

This requirement presents new challenges of orchestrating data movement from the supercomputer to the local and remote storage systems. Data distribution will have to be integrated into the data generation phase. Managing the dataflow using well-coordinated workflow engines will be required as part of the software infrastructure that runs the simulations.

The issue of large-scale data movement will become more acute as very large datasets or subsets are shared by large scientific communities. This situation will require large volumes of data to be replicated or moved between production and analysis machines, often across the wide area. While networking technology is greatly improving with the introduction of optical connectivity, the transmission of large volumes of data will inevitably encounter transient failures, and automatic recovery tools will be necessary.

Another fundamental requirement is the automatic allocation, use, and release of storage space. Replicated data cannot be left in storage devices unchecked, or storage systems will fill and become clogged. A new paradigm of attaching a lifetime to replicated datasets, and the automatic management of data whose lifetime expires, will be essential.

3.1 Parallel File Systems

Parallel file systems such as Lustre and PVFS2, and I/O software stacks including MPI-IO and high-level I/O libraries (e.g. HDF5, Parallel netCDF) are in extensive use in HPC by a wide variety of applications. Current deployments typically use vendor file systems and enterprise hardware and are providing adequate storage performance, capacity, and reliability for current systems. For the next decade the key challenges to Parallel File Systems are scaling, performance, and fault tolerance. Overall, we need storage systems at HPC centers that provide scalable bandwidth and tolerate non-catastrophic failures without data loss.

3.2 Data Management

Scientists are facing the burden of managing the data generated by large-scale simulations and experiments. They need to deal with multiple steps of moving the data between software modules, extracting subsets of the data, summarizing the data, generating images or movies, and moving the data to archival storage. Such tasks are extremely time consuming, and require expertise that is irrelevant to the scientist, such as transfer protocols, security mechanisms, and idiosyncrasies of archival systems.

In order to support exascale data generation, data storage will fundamentally change. Users will need tools that manage the movement of data automatically across a storage hierarchy. Data that is used often will be moved to highly parallel dynamic storage, while archived data will reside in powered down storage or passive storage devices. Furthermore, algorithms to automatically track and remove unused data from the dynamic storage will be essential to minimize storage costs. Collections of datasets will be organized as directories. Such abstraction will fundamentally change the way the I/O is expressed by applications and will involve a storage management layer that maps datasets into physical devices without effecting the applications.

Keeping track of the data generated is already a daunting task. The meaning of the data, referred to as metadata, requires precise annotation of how the data was generated, and the scientific interpretation of each data item. Furthermore, many scientific datasets are generated from other datasets, or perhaps a combination of datasets. This requires the capability of tracking the history, or provenance, of the data. Today, such tools are provided in ad hoc manner; some metadata is collected in various forms of notebooks, some in databases, and some embedded as headers of files. In the exascale regime the automation of this task is essential because of the sheer volume of the data and the accelerated rate of their production. Standard metadata models and tools will have to be developed, as well as tools to automatically capture the metadata as the datasets are generated. Furthermore, the data models need to support standard ontology for each scientific domain and allow for dynamic evolution of such standards.

3.3 Turning Data into Scientific Discoveries

One of the challenges in contemporary science is the process of discovering knowledge and testing hypotheses in the presence of a growing deluge of data. A recurring theme in this document—that existing methods will not scale to meet the challenges of exascale systems and data—holds true in the area of knowledge discovery. Existing approaches for knowledge discovery do not scale to the exascale. Failure to address the issues of knowledge discovery in the exascale ecosystem will have a profound and adverse impact on all science programs.

A number of different, yet complementary, approaches to address these problems will require exploration:

- Ability to visualize and analyze results at coarse and fine resolutions depending to support the natural investigatory process that relies on context/focus interaction;
- Better visual data analysis algorithms for characterizing and presenting uncertainty;
- Integration of visual data presentation and data analysis techniques (e.g., clustering, classification, statistical analysis and representation) to aid in accelerating knowledge discovery;
- Greater emphasis on the human-computer interface to increase the efficacy of visual presentation motifs and interactive knowledge discovery interaction models;
- Context-centric interfaces to simplify use of complex software infrastructure;
- Rethinking design and implementation of fundamental knowledge discovery algorithms and software infrastructure to be capable of effectively leveraging exascale platforms.

As the size of simulation, observational, and experimental datasets grow into the petascale range, many of the existing technologies do not scale to be practical for both on-line and off-line data analysis and knowledge discovery processes. These additional challenges need to take advantage of acceleration, parallel processing, and smart navigation, summarization, and manipulations of the massive datasets. New methods for achieving better efficiency of searching, processing, exploring, and displaying information are needed. Finally, scalable and flexible data formats for storing, processing, provenance, and sharing results of data analysis are required.

3.4 Efficient Searching

Searching for key pieces of information in data is becoming challenging due to several factors. With data reaching the petabyte scale, there is a need for better indexing technology to support multiple tasks such as database search, sub-graph extraction, and text searches with ranking. The data being searched is also becoming more complex, with simple row/column tables being replaced by graphs, data with associated uncertainty, collections of data such as a sequence of interactions in a graph, and so on. Users are also making more complex queries and may require an estimate of the time it would take to obtain an answer to the query. To address these issues, we need advances in several different areas including, but not restricted to, indexing, sampling, query estimation, and approximate query answering. The characteristics of modern datasets, as well as the hardware on which the analysis software is executed, suggest the need to re-think existing algorithms or develop new ones due to:

***Scalability of the analysis techniques**:* We need advances in both mathematical algorithms and computer science issues to ensure that our analysis techniques will scale with both the size of the data and the number of processors available to run the analysis algorithms. This would require new parallel algorithms, scalable data structures, techniques for re-organizing the data to be more suitable for multiple processors, automatic compiler-driven parallelization, etc. Since data maybe inherently distributed and streamlined, algorithms need to be adapted to these physical properties of the data.

***Modifying algorithms for new architectures**:* With the paradigm shift from single processors to multi-core architectures, GPUs, and FPGAs, we need research to determine how scientific data analysis tasks can be re-designed to be highly multi-threaded to take advantage of these architectures. In particular, I/O bottlenecks often encountered by data intensive applications can be circumvented with in-memory data operations and specialized indexing techniques such as Quaterrnary Triangular Mesh (for geoprocessing), and space filling curves (for increasing locality in multidimensional spaces). The new architectures can be particularly well suited for some of the newer types of data. For example, algorithms for fast quantile and frequency estimation in data streams can benefit from the use of GPUs. Likewise, significant amount of processing tasks may be accelerated using FPGAs, e.g., kernel computations, key statistics, pattern recognition using templates etc.

Analysis within storage: An approach to minimizing the time taken to move data from storage to where it is being analyzed is to analyze the data where it is in storage. This is referred to as Active Storage. Research is needed to understand the data structures necessary for such analysis and the approaches including programming models, software libraries used to embed analysis functions within storage, and the storage infrastructure enhancements necessary to make this possible.

Exploiting modern programming models and constructs: MapReduce, Bigtable, and have been successfully used in various applications on several different architectures for the analysis of large datasets. However, it is unclear if such programming models can be directly used in the context of scientific data. Research is needed to determine how such models can be extended to implement scientific data analysis algorithms and meet the requirements of fault tolerance and scalability, while supporting the fine granularity and frequent synchronization needs of scientific applications.

4. Software Sustainability

Creating an exascale software ecosystem entails more than just solving the technical challenges. It includes educating scientists on how to use the solutions, both new tools and new approaches, and demonstrating why using these solutions is to their advantage. It includes making sure that the solutions are hardened to production quality so that they can be integrated into the software suites of the nation's supercomputer centers. It includes making pieces available as they are completed, rather than waiting until everything is done. And it includes helping users integrate these pieces into existing codes so that science teams can benefit in the near term and build up trust in the solutions being provided for the exascale software ecosystem.

Sustaining and hardening software to production quality: Academic and laboratory researchers and developers rarely possess either the software engineering skills or the desire to transition research ideas to production code, with concomitant support. The pathway from research prototype to a software tool that is widely available, production quality and actively supported is not clear. In most cases, the funding researchers receive is targeted toward specific research goals, and not necessarily to provide tool porting, testing, documentation, standardization, or user support. A new model of software tool support is needed if we are to address current and future needs.

Engagement with applications and domain experts: All too often, software tools are developed in the absence of detailed understanding of the user and application needs. Conversely, users are often unaware of the technical difficulties underlying tool design and support. Bridging this gap with a collaborative software development and extension process, where promising ideas are identified and tested early, then enhanced and supported across the application development and support cycle, would ameliorate the expectations gap.
User training: Software development tools can be very flexible and powerful in their own right. The developers of these tools should make it a priority to train the user community on tool capabilities and usage. Furthermore, usability should be a major requirement included in any funding focusing on transition to production software.
Education and workforce: As is the case in other areas of HPC and computer science, there is a specific need to educate new students and workers in order to ensure a sufficiently large and capable workforce.

6. References

1. Final report from Exascale townhall meetings- Breakout Group Seven "Software Challenges". June 2007

2. Workshop on Software Development Tools for Petascale Computing final report. August 2007

3. Workshop on Visual Analysis and Data Exploration at Extreme Scale final report, October 2007,

4. Scalable Systems Software Summary Report ASCR PI meeting, April 2008

5. Data Management and Analysis Summary Report ASCR PI meeting, April 2008

6. Workshop on Mathematics for Analysis of Petascale Data final report, June 2008

7. Whitepaper "The Scientific Data Analysis Process at the Petascale" Editors: Chandrika Kamath, Arie Shoshani, August 2008

8. Workshop on CS/Math Institutes and High Risk/High Payoff Technologies for Applications preliminary report, October 2008

9. DARPA "ExaScale Computing Study: Technology Challenges in Achieving Exascale Systems", Kogge, et.al. (September 2008) http://www.cse.nd.edu/Reports/2008/TR-2008-13.pdf

10. DARPA "Exascale Software study", Sarkar, et.al., (In preparation) http://www.lbl.gov/CS/html/SC08ExascalePowerWorkshop/Sarkar-SC08-Exascale-Workshop-v2.pdf

Towards Exascale File I/O

Yutaka Ishikawa
University of Tokyo, Japan

2009/05/21

Background & Overview

- Existing libraries and systems
  - High Level I/O Library
    - Parallel HDF5
  - Collective I/O
    - MPI-IO
  - Global File System
    - Lustre, PVFS, GPFS, …
- Existing file systems
  - Global file system only
    - Most systems
  - Staging
    - Files are copied to a local disk of each compute node before execution, and then dirty files are copied to the global file system after execution.

      e.g., Earth simulator, Riken Super Combined Cluster, PACS-CS@Univ. of Tsukuba

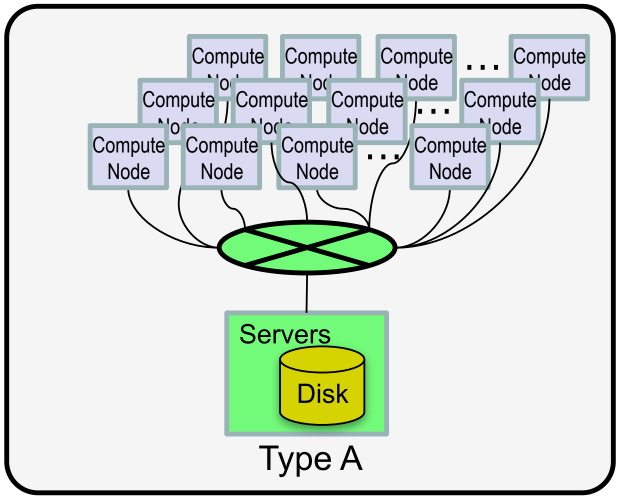
- Environment to develop exascale file systems
- Challenges towards exascale systems
  1. File system configuration
  2. Exascale file access technologies
  3. Exascale data access technologies
  4. Exascale Layered implementation
- Collaboration Scenarios
- Milestone

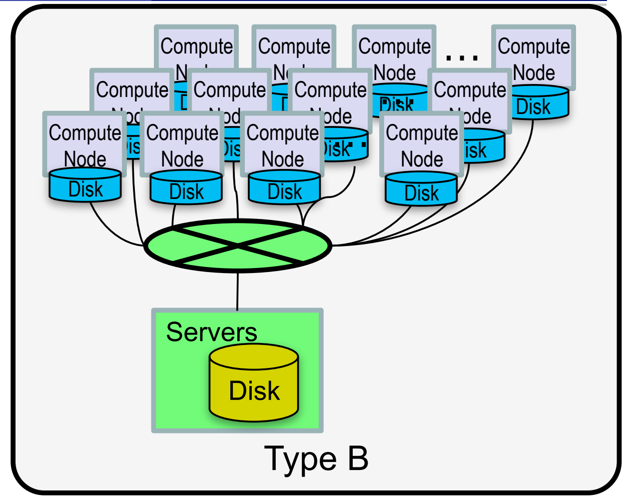
Environment to develop exascale file systems

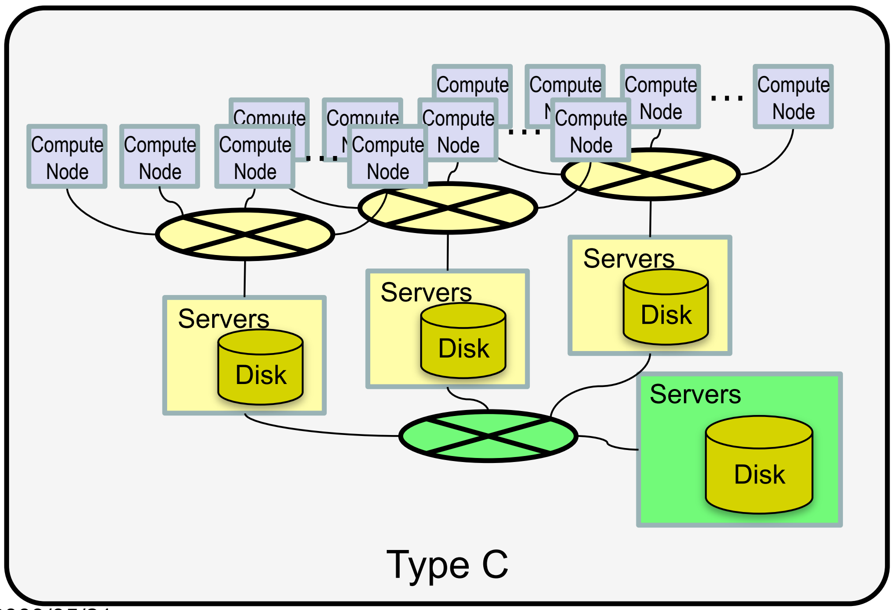
- Benchmarks and use cases
  - To understand file system performance and reveal the weakness of the file system
  - Involvement of application developers' skill
    - We have to discuss with application developers to understand the application characteristics
    - We have to cooperate with application developers to achieve better file I/O performance
  - Is the following consortium still working ?
    - Parallel I/O Benchmarking Consortium
      http://www-unix.mcs.anl.gov/pio-benchmark
- Tools
  - File I/O access tracer
    - To understand the application I/O characteristics
- Experimental Equipments
  - 1 K to 10 K nodes
  - The developed code can be deployed to compute nodes and file servers
    - Kernel modification

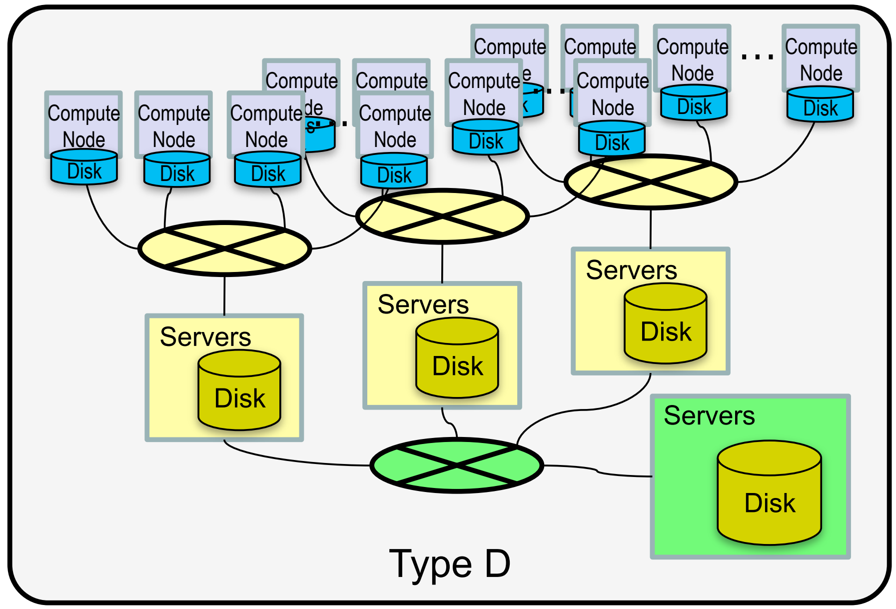
Research Topics: File system configurations

	Global file system	Local disk on each node	Disk for group of nodes
Type A	✓		
Type B	✓	✓	
Type C	✓		✓
Type D	✓	✓	✓









Research Topics: Exascale file access technologies

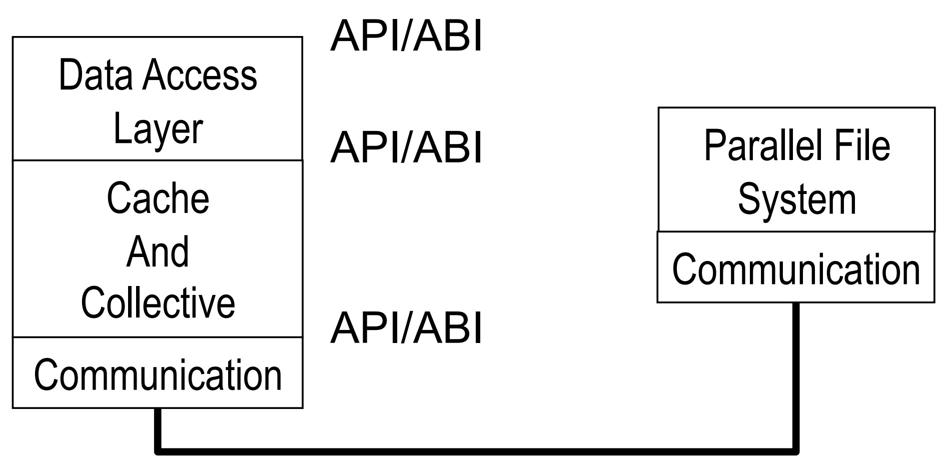
- Type A (Global file system only)
  - This configuration may be not applied
- Type B (Global file system + Local disk)
  - Local disk will be SSD.
  - Research topics
    - Both the file and meta-data cache mechanisms in each node
    - File staging
    - If two networks for both computing and file access are installed, some optimization mechanisms utilizing both networks are also research topics
- Type C (Global file system + Group file system)
  - Each group file system provides the file access service to the member nodes of its group
  - Research topics
    - Efficient file staging
    - The group file system as file cache
    - The file service mechanism to some groups if an application runs over those groups
- Type D (Global file system + Group file system + Local disk)
  - The combination of Types B and C

Research Topics: Exascale data access technologies

- Most application developers use the read/write file I/O system calls (, at least in Japan)
- If the parallel HDF-5 is enough capability to describe exascale applications, the following research topics are candidates:
  - Efficient implementation of parallel HDF-5 for exascale parallel file system
    - Optimization over cores in each node
  - Application domain specific libraries on top of parallel HDF-5
- If the parallel HDF-5 is not enough capability,
  - Design of extended API and the implementation data structure is redesigned
- Deployment Issues
  - Portable efficient implementation
  - Tutorials for the application developers

Research Topics: Layered Implementation for collaboration

- Data Access Layer
  - Parallel HDF and others
- Cache and Collective Layer
  - Approaches
    - Memory-mapped parallel file I/O
    - Distributed Shared Memory
    - …
- Communication Layer
  - Accessing parallel file system
- Parallel File System



Collaboration Scenarios

1. Almost no collaboration
   – Joint workshops are held
2. Loosely coupled collaboration
   – Benchmarks are defined
3. Collaboration with Standardization
   – Network protocol is defined
   – Client-side API/ABI are defined
      • New Parallel File I/O
      • Highly abstracted Parallel File I/O in addition of HDF5 ?
4. Tightly Coupled collboration
   – Developing the single File I/O software stack

Milestone

	2009	2010	2011	2012	2013	2014	2015	2016
Benchmarks		V1.0						
development								
Supercomputer centers@Japan			Univ. of Tokyo	Kyoto Univ.	Univ. of Tsukuba	Univ. of Tokyo		

IESP Exascale Challenge:

Co-design of Architectures and Algorithms

Al Geist (ORNL) and Sudip Dosanjh (SNL)

Historically, huge supercomputers were built and delivered with little or no software on them. The application developers were left with heroic efforts to get there simulations to run efficiently on these systems. In order to improve the effectiveness of peta and exascale systems we need to have a paradigm shift where architectures and algorithms are co-designed.

There is a large gap between the peak performance of supercomputers and the actual performance realized by today's algorithms. This architecture-algorithm performance gap will get even wider with the increase in computing power being driven by a rapid escalation in the number of cores incorporated into a single chip rather than increases in clock rate. The transition from massively parallel architectures to multi-core architectures will be as profound and challenging as the change from vector architectures to massively parallel computers that occurred in the early 1990's that enabled our Nation and the U.S. Department of Energy to break the teraflop barrier. To effectively bridge this architecture-algorithm gap and use the next generation of computers, we must solve a host of architectural challenges in hardware and software.

Hardware challenges:

- Moore's Law still holds, but clock speed is constrained by power and cooling limits
- Processors are shifting to multi/many core with attendant hierarchical parallelism
- Compute nodes with hardware accelerators create the additional complexity of heterogeneous architectures
- Processor cost is increasingly driven by pins and packaging, which means the memory wall is growing in proportion to the number of cores on a processor socket
- Supercomputer architectures must be designed with an understanding of the applications they are intended to run
- A supercomputer architecture that performs well on full scale real applications cannot be built from only commodity components.

Software challenges:

- Scaling limitations of present algorithms
- Hierarchical algorithms to deal with bandwidth across the memory hierarchy
- Software strategies to mitigate high memory latencies
- More complex multi-physics requires large memory per node
- Need for automated fault tolerance, performance analysis, and verification
- Innovative algorithms for multi-core, heterogeneous nodes

Promoting the integrated co-design of architectures and algorithms represents a fundamental shift from simply procuring and operating large scale systems. A key way to lower the risk of adopting novel architectures and technologies is to demonstrate through paper studies, system simulation, and hardware prototypes the performance benefit of these technologies. The vision is that both the hardware designers and the software designers will compromise based on what the other group can do in a given timeframe. The evolution of the architecture and algorithms then becomes more aligned, which helps close the performance gap. Deploying small prototype systems will facilitate application, algorithm and system software development, prove the technology to industry, and lower the risk of adoption of advanced architectures. The metrics for success will be measured through changes to product roadmaps, and integration or adoption of co-designed technologies into next generation supercomputer systems.

IESP Exascale Challenge:
Resilience and Fault Tolerance

Al Geist (ORNL) and Franck Cappello (INRIA)

Research in the reliability and robustness of exascale systems for running large simulations is critical to the effective use of these systems. New paradigms must be developed for handling faults within both the system software and user applications. Hardware support may also be investigated to reduce the fault tolerance overhead. Equally important are new approaches for integrating detection algorithms in both the hardware and software and new techniques to help simulations adapt or be indifferent to faults. One essential element toward these objectives is a better understanding of HPC usage scenarios, applications needs, fundamental origin of the algorithm sensibility to faults, and failures root causes.

What users want is resilience in the execution of their applications. They want to be able to submit a long-running job and have it run to completion in a timely manner. However, because of their scale and complexity, today's supercomputers typically have faults somewhere in the system every day and run for only a few days before the number of faults require rebooting. While supercomputers can often reconfigure around faults, every fault kills the application running on the affected resources. Historically these applications have to be restarted from the beginning or from their last checkpoint, but the checkpoint/restart technique is already losing its effectiveness on petascale systems and will not be viable on exascale systems because of the rate of failures and time required to write out checkpoints. A new fault will occur before the application could be restarted, causing the application to get stuck in a state of constantly being restarted.

Exascale systems will have millions of processors in them and some projections say they will have a billion threads of execution. The major challenge in resilience is that faults in extreme scale systems will be continuous rather than an exceptional event. This requires a major shift from today's software infrastructure. Every layer of the exascale software ecosystem has to be able to cope with frequent faults; otherwise applications will not be able to run to completion. The system software must be designed to detect and adapt to frequent failure of hardware and software components. With the potential that exascale systems will be having constant failures somewhere across the system, application software isn't going to be able to rely on current checkpointing techniques to cope with faults. For exascale systems, new paradigms to tolerate hardware and software faults will need to be developed and integrated into both existing and new applications.

Silent errors are the ìmonster in the closetî for exascale systems. Silent errors are simply faults that occur that never get detected. They can be transient as in the case when a bit or logic gate gets flipped spontaneously. Transient flipping of bits happens continuously in the memory of the largest systems in the world, but ECC memory automatically detects and corrects these faults. Silent errors arise when any part of the memory is not ECC or data paths are not protected, or when multiple memory faults cancel each other out preventing ECC from detecting the faults. Silent errors are not limited to transient affects, for example, an undetected hardware failure is a silent error. Often they are only discovered when the application running on this hardware: gives the wrong answer, fails to complete, or completes much more slowly than usual. By then it is too late for the application to recover. Silent errors are not limited to hardware faults. There have been several cases where software or firmware code has had bugs in it that only manifest in rare cases, for example, router-chip software that changes the bits in one message out of every billion. The key characteristic of silent errors is that they are undetected; therefore, there is no opportunity for an application to adapt or recover from the fault. If the rate of silent errors is too high, then a user must worry that the results of his simulation are correct. This gets back to resilience and correctness of their algorithms and application in the face of faults. Designing mechanisms to tolerate silent errors depend on a better comprehension of these errors especially when they hit the hardware. Very few results are available about the quantitative evaluation of their likelihood at large scale during the application executions.

Exascale systems will need to have much more hardware fault detection built into the architecture and software fault detection built into the software stack in order to reduce the rate of silent errors. Once detected, there is still much work to do, including coordination between different layers of the software stack, deciding on a plan for recovery, reconfiguration, adaptation, and recovery of the application.

When faults become continuous, there will be a critical need for fault oblivious algorithms, and applications that can run-through faults. Very little is known today about how to create such applications except for in the simplest cases that are nearly embarrassingly parallel. The challenge does not rest just with the application developer, the system software also needs to be completely rethought to allow it to cope with a continuous stream of faults and being in a constant state reconfiguration of the system. Much research and paradigm shifts must occur.

In addition to progress in application, system and hardware, there is a need for experimental environments being able to stress and compare different fault tolerance approaches and techniques in a scientific way. Large scale testbeds are essential in the observation and understanding of complex phenomena. Software environments capable of reproducing usage and fault scenarios are also needed to test and debug new resilience concepts at large scale, before putting them in production.

Consistent Application Performance at Exascale

William Kramer and David Skinner

June 21, 2009

This whitepaper sets out to examine the future of application performance consistency on exascale parallel computing systems. By performance consistency we mean the regularity of wall clock times to complete a fixed amount of application progress. In particular we do not address consistency of results from applications. Correctness of results is an important topic as well and will be treated separately.

The design of high performance computers concentrates on increasing computational performance for applications. Performance is often measured on an optimally configured, dedicated or near dedicated system to show the best case in performance. In real environments, resources are seldom dedicated to a single task and systems run multiple tasks that may negatively influence each other. It is this more complex production context that is arguably more important in setting user expectations of application performance. Managers of large HPC resources likewise depend on consistent delivery application performance in production for allocation shared use of the resource by multiple science teams.

Large scale systems running in production mode are particularly prone to performance fluctuation. By their nature they involve a large number of components servicing a varied workload. Resource contention which results in performance degradation can be caused by underprovisioned interconnects, topology mismatches, congestion aware messaging, assignment of memory, systems software layers, system management event timing (daemons running at particular times aka “system jitter”), bugs in configurations, software and hardware and system management and configurations. Keeping all of these impediments in check so that users observe consistent performance is challenging at the terascale and petascale. It is therefore crucial that we consider how larger machines and larger applications can be avoid the pitfalls encountered with today’s machines.

What level of consistency is reasonable to expect for Exascale? Inconsistency of parallel applications has implications for how much useful work can be produced by Exascale systems. Performance inconsistency is caused by many factors but on well managed HPC systems, the simple causes of inconsistency (multiple jobs running within a shared memory processor, simple configuration mistakes, etc.) are not the primary causes of inconsistency. Factors leading to changes in performance occur over multiple time scales and originate both from within systems, within applications and from external sources. As a result, variability in runtime performance is strongly tied to the hardware and software architecture. On today’s Terascale system, it has been shown that high degrees of consistency (CoV < 1%) are regularly achievable for most workloads (Kramer W. T., 2008).

The performance impact of inconsistency can be quite large, becoming the dominant impediment to parallel scaling in some cases. Consistency, or really the lack of it, will play an even larger role for the effectiveness of Exascale architectures unless proactive steps are taken to address it. Inconsistency can result from a myriad of causes including the hardware architecture, the.

Understanding the parallel scaling factors leading to performance inconsistency needs to be a chief concern of the design and use Exascale systems. Since the majority of testing and performance analysis is done on test systems much smaller than production machines, it is common to encounter variability induced performance loss at scale that goes unseen on smaller machines.

The variability of performance is as important as availability and mean time between failures to users to be able to accomplish their goals. For example, the user's productivity is impacted at least as much when performance varies by a factor of two, as when a system's availability is only ½ of the expected time. In both cases, the amount of work done is only half of what is expected of the system.

Multiple sources show inconsistency in runtimes leads to many negative impacts [(Figueira and Berman 1966), (Worley and Levesque 2004), (Zhang, Sivasubramaniam, Moreira, & Franke, 2001)] all of which make a HPC, and future Exascale systems have less value. The first impact is less overall work done by the system. Runtime inconsistency is inherently bad for performance since variations in runtime proceed upward from some best case runtime, i.e., variation is seldom toward better than optimal performance. The longer a task takes, the more time it takes to get usable results for analysis. Since some applications have a strict order of processing steps (i.e. in climate studies, year 1 has to be simulated before year 2 can start), they cannot directly overcome this slowdown via, say, increased parallelism. Inconsistency can also introduce wider error margins for non-deterministic applications, leading to more difficulty verifying results.

Inconsistency decreases the efficiency of HPC parallel computers since cycles are lost to both job failure and complex job scheduling to mitigate the lack of consistency [(Srinivasan, et al. 2002), (Lee, et al. 2004)]. Jobs fail through incorrect estimation of the batch queue requirements. System scheduling becomes less effective because users must be overly conservative in requesting batch time. Most scheduling software relies on user-provided run estimates, or times assigned by default values, to schedule work effectively. When a cautious user over estimates runtime, the job scheduler operates on poor information and results in inefficient scheduling selections on systems. These all contribute to the loss of user productivity and decreased system impact.

Consistency is influenced by a number of factors.

- System configuration and management errors and bugs (Kramer and Ryan 2003) – at Exascale, with orders of magnitude more components, there will be increased likelihood that such artifacts are introduced.
- Hardware architectural features – including the network topology, size of computational nodes, hardware collective features (from vectors to distributed collectives), automated error recovery and hardware consistency features (e.g. global cocks). (Skinner and Kramer October 6-8, 2005) At Exascale, the trade-offs of many more cores within an SMP/node or a much broader network, will greatly influence the consistency of systems.
- Software architectural features – including message passing collectives, the OS foot print (micro kernels, Light Weight OS, full OS), synchronization primitives, automated error recovery and service provisioning. (Kramer and Ryan May 2003) The software layers, being less integrated than HW design and having to be limited by hardware features by prove the most challenging area to control inconsistency at the Exascale.

- Application Implementations – including in-effective use of resources, static workload allocation, I/O and application specific check pointing. At the Exascale, in order to deal with the system challenges of resiliency, parallelism and performance, applications will have more responsibility for dynamic workload reassignment, adaptive behaviors (AMR) and recovery. This will lead to even more challenges for consistency unless there are well planned interactions between the system components and the applications.
- Resource Management – including scheduling applications that compete for resources, prioritization, Quality of Service, and coordination of services. Often this type of consistency challenge is the result of insufficient information for the scheduling agents and insufficient methods for applications to express their needs. At the Exascale, power management will increase the need for dynamic interactions competing different needs. For example, the Exascale facility may wish to control power costs, or the system may do power control automatically, without taking into account the consistency needs of the applications.

The challenge is how to maintain this level of consistency at the Exascale. To date, once inconsistency is identified, it is possible, albeit not always easy, to restore consistency by making changes to parameters, fixing bugs and adjusting configurations and so on. It is not clear this will be the case at Exascale unless consistency is a holistic design parameter. Key issues for assuring consistency at the Exascale include

- Architectural and system design criteria that reflects consistency requirements
- Testing for consistency at scale
- Well studied solutions and trade-offs for consistency
- Consistency metrics for Exascale systems
- Understanding system architectural influences that can be explicitly linked to consistency
- Resource management that is too narrowly defined
- Ineffective methods to express performance and consistency needs up and down the software hierarchy

In order for Exascale systems to exhibit the consistency that is required to make the applications and systems productive, new understanding of the causes and solutions to inconsistency are needed, along with new ways of measuring the impact of design, implementation and operational choices have on consistency. In order for applications to mitigate the effects that make systems inconsistent, new mechanisms for expressing consistency requirement and applications reactions are also required.

Figueira, S. M., & Berman, F. (1966). Modeling the Effects of Contention on the Performance of Heterogeneous Applications. *Proceedings of the High Performance Distributed Computing (HPDC '96)*, (p. 392).

Kramer, W. T. (2008). *PERCU: A Holistic Method for Evaluating High Performance Computing Systems.* University of California at Berkeley, Department of Electrical Engineering and Computer Science. Berkeley, CA: University of California.

Kramer, W., & Ryan, C. (May 2003). *Performance Variability of Highly Parallel Architectures.* Berkeley, CA: Lawrence Berkeley National Laboratory.

Kramer, W., & Ryan, C. (2003). Performance Variability on Highly Parallel Architectures. *International Conference on Computational Science 2003.* Melbourne Australia and St. Petersburg Russia.

Lee, C. B., Schwartzman, Y., Hardy, J., & Snavely, A. (2004). Are user runtime estimates inherently inaccurate? *10th Workshop on Job Scheduling Strategies for Parallel Processing.* New York, NY.

Skinner, D., & Kramer, W. (October 6-8, 2005). Understanding the Causes of Performance Variability in HPC Workloads. *2005 IEEE International Symposium on Workload Characterization (IISWC-2005).* Austin, TX.

Srinivasan, S., Kettimuthu, R., Subrarnani, V., & Sadayappan, P. (2002). Characterization of Backfilling Strategies for Parallel Job Scheduling. *nternational Conference on Parallel Processing Workshops (ICPPW'02)*, (p. 514).

Ujfalussy, B., Wang, X., Zhang, X., Nicholson, D. M., Shelton, W. A., Stocks, G. M., et al. (November, 1998). High performance first principles method for complex magnetic properties. *Proceedings of the ACM/IEEE SC98 Conference*. Orlando, FL: IEEE Computer Society, Los Alamitos, CA 90720-1264.

Worley, P., & Levesque, J. (2004). The Performance Evolution of the Parallel Ocean Program on the Cray X1. *Proceedings of the 46th Cray User Group Conference.*

Zhang, Y., Sivasubramaniam, A., Moreira, J., & Franke, H. (2001). Impact of Workload and System Parameters on Next Generation Cluster Scheduling Mechanisms. *IEEE Transactions on Parallel and Distributed Systems , 12* (9), 967-985.

An Exascale Approach to Software and Hardware Design

William Kramer and David Skinner

June 21, 2009

The demands of Exascale require a complete rethinking of the software and hardware development process that has become the ad hoc standard in HPC. For the past 10-15 years, horizontal layers software and hardware design and development have been the de facto standard of creating HPC software, in part due to the influences of funding methods, research incentives, software methods, the Open Source movement and commercial outsourcing and specialization. This Horizontal Design approach leads to the development of discrete components in the SW stack and independent hardware components – all developed with different methods, differing requirements and quality. Unlike past generations of system software (from the earliest OSs through to the original community source movement with Unix) and hardware, where some degree of top to bottom Vertical Design existed, the last 10-15 years have been dominated by plug and play componentization that are focused on horizontal functionality and portability.

The horizontal design approach has notable successes like the Linux kernel, MPI and a variety of job schedulers. It also has many challenges that are inhibiting progress and making even Petascale systems challenging to fully exploit. As many who field Terascale clusters know, every cluster is now unique with different horizontal components (often in name, at least in version). Currently at the Tera and Petascale level there is only one company that produces a system software stack which is vertically designed from top to bottom and one other company that is providing a scaled, vertically tested stack that has specifically designed components added to the horizontal components.

To reach Petascale, the HPC community has mitigated many of the issues in the horizontal design method such as relying on vendors to do vertical testing and integration, standing up extra test bed resources for integration testing and error correction, taking excessive time from the few large scale production systems to do integration, testing, diagnosis and correction, or living with inefficient and error prone systems. The current horizontal design method presents a number of insurmountable challenges to reach Exascale. Yet economics and cost effectiveness will not let us return to the days of completely proprietary vertical methods. Nor can one organization alone, be it government or private, afford to pioneer Exascale and make it a success.

There is some hope! What is needed is to change the horizontal approach of developing essentially isolated SW components that have narrow view of their function and role in the system. Instead, the community must organize the hardware and software development activities with component cross cutting principles. The cross cutting principles define the requirements, function, interfaces, integrations and performance needs for each horizontal component. Instead of thinking of integration as the final step in defining and developing and Exascale system, it will be the first step.

The cross cutting requirements for the vertical design approach were identified to first order at the first IESP workshop. They include: Resilience (reliability & fault tolerance);

Performance; Programmability; Computational model; I/O; Consistency and verification; Resource Management; and Power Management/Total Cost of Ownership.

There are limited proofs of existence that has the hint the vertical design approach yields an effective and long lived, yet flexible solution to this conundrum. Some examples include

- The DOE SciDAC program. SciDAC introduced the concept of software application development teams and software infrastructure teams that are linked in an iterative approach to developing applications that relay on increasing more effective software infrastructure
- Scientific "framework" development for large scale experiments and long lasting infrastructure. High Energy Physics regularly uses a formal process of vertical architecture definition, software development and testing often incorporating thousands of funded and unfunded contributors. The processes here are notable for progressive demonstrations of integrated progress milestones (e.g. Data Challenges) and timely delivery for equipment that is being co-developed.
- The methods used to produce community based SW such as the High Performance Storage System which follows formal methods and shares development across multiple organizations.
- Commercial OS development methods such as those that exist in IBM and Cray.
- Formal testing methods that are used in verification and validation of network protocol change proposals.

One factor motivating a renewed emphasis on vertical integration is the dominance of software failures as the causative factors in large scale system availability. Failure at scale of system software such as filesystems, batch schedulers, and even authentication mechanisms such as LDAP is a major problem for HPC resource managers. In many cases vendors leverage software which works well at smaller scales but place too much reliance on the ability of the software to integrate seamlessly at all levels. Some studies indicate that on large systems, across vendors and architectures, SW accounts for the majority of sytem wide failures on HPC systems.

User experience can also suffer when insufficient attention is paid to end to end software functionality. The usability of tools such as debuggers and performance profilers can diminish significantly when they are used beyond the scale that software vendors are capable of performing testing. Improving usability and thus the value of the HPC resource to science can be improved by a more goal oriented vertical approach, one that builds in expectations of usability at scale.

The vertical approach is not at odds with previous approaches to HPC software development. A tightly coupled vertical design can still produce software which is of lateral use, however attention to vertical integration diminishes risks of software failure encountered when relying upon "off the shelf" generic software. Vertical integration does not replace these software components but improves them for HPC.

In summary, Exascale will not be achievable without a tightly coupled vertical design, design and integration process. The methods that got the HPC to early Petascale will not stretch to Exascale. The vertical approach does not diminish community contributions, flexibility or openness, but rather makes the investments people and organizations make more likely to have impact.