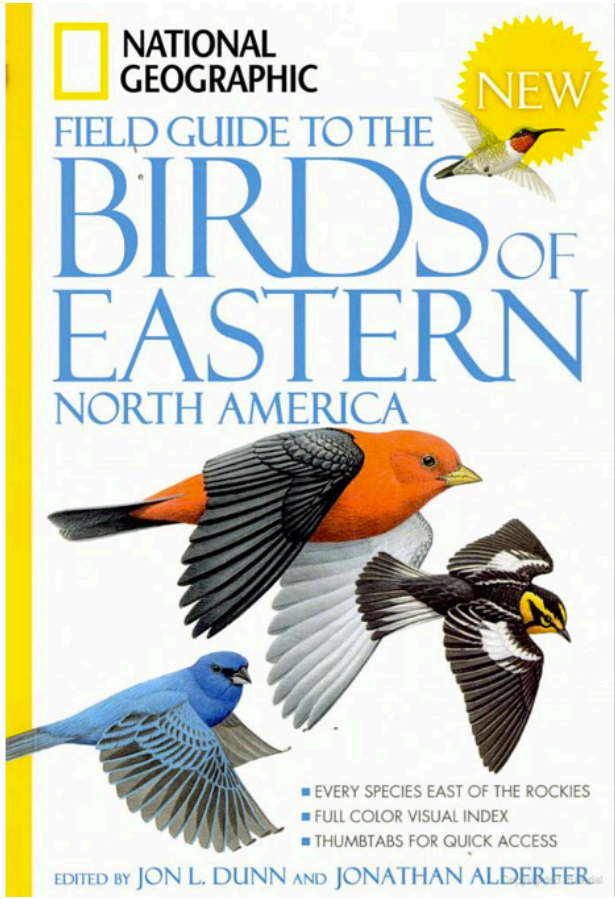
THE EXASCALE SOFTWARE CENTER

Pete Beckman

Director, Exascale Technology and Computing Institute (ETCi)

Jim Ahrens, Pavan Balaji, Pete Beckman, George Bosilca, Ron Brightwell,
Jason Budd, Shane Canon, Franck Cappello, Jonathan Carter,
Sunita Chandrasekaran, Barbara Chapman, Hank Childs, Bronis de Supinski,
Jim Demmel, Jack Dongarra, Sudip Dosanjh, Al Geist, Gary Grider,
Bill Gropp, Jeff Hammond, Paul Hargrove, Mike Heroux, Kamil Iskra,
Bill Kramer, Rob Latham, Rusty Lusk, Barney Maccabe, Al Malony,
Pat McCormick, John Mellor-Crummey, Ron Minnich, Terry Moore,
John Morrison, Jeff Nichols, Dan Quinlan, Terri Quinn, Rob Ross,
Martin Schultz, John Shalf, Sameer Shende, Galen Shipman, David Skinner,
Barry Smith, Marc Snir, Erich Strohmaier, Rick Stevens, Nathan Tallent,
Rajeev Thakur, Vinode Tipparaju, Jeff Vetter, Lee Ward, Andy White,
Kathy Yelick, Yili Zheng

IESP/ESC Preface



- Software Stacks in use
  - First initial draft (thanks Bernd)
  - We understand the space
  - Can now make it formal…
- ESC Application Inventory for exascale
  - Initial version collected from DOE
  - Time to widen and make formal…
- ESC Software R&D Plans
  - Initial draft presented today
  - Working on international dependencies
    - MOUs?
    - Funding plans?
    - Reliable milestones for joint work?

Context: Planning for Exascale

Cross-cutting Metrics, Objectives, Initiative Goals

Platforms	Cross-cutting Technologies	Co-Design Application Teams	Exascale Software
~2015/2016 ~2018/2019		•Materials •HEDP •Nuclear Energy •Chemistry •Fusion ….	

Goal: Ensure successful deployment of coordinated exascale software stack on Exascale Initiative platforms

Exascale Software Center

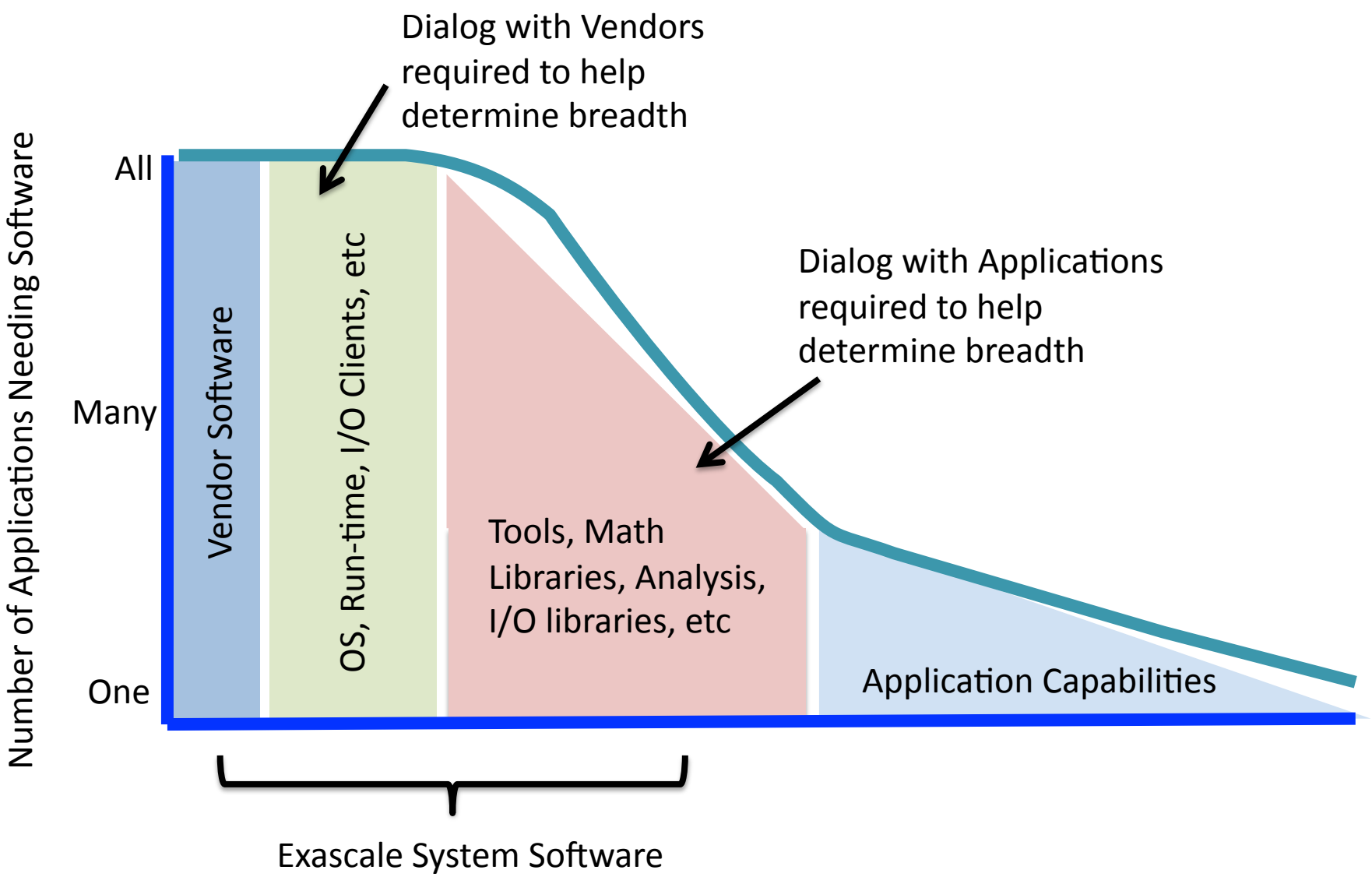
Responsible for success of software:

- Work with DOE mission applications, co-design centers, research community, platform partners to:
  - Identify required software capabilities
  - Identify gaps
- Design and develop open-source software to provide capability
  - Create new components
  - Evolve existing components
  - Includes maintainability, support, verification
- Ensure functionality, stability, and performance
- Partner with platform and software vendors to deploy software
- Coordinate outreach to the broader open source community
- Ensure milestones are met

Exascale Software Center

- Scope
  - Identify software capability gaps, research & develop solutions, test and support deployment
  - Deliver high quality system software for exascale platforms
    - Platforms in 2015/201 and 2018/2019
  - Increase the productivity and capability and reduce the risk of exascale deployments
- Cost: scale up to significant development teams
  - Applied R&D with distributed teams
  - Large, primarily centralized QA, integration, and verification team
- Schedule Overview
  - 2010 – Q1 2011: Planning and technical reviews
  - Spring 2011: Launch Exascale Software Center!
  - 2014, 2017: SW ready for integration for 2015, 2018 systems respectively

Breadth of Software



Depth of Software

Broad CS and Applied Math Research

Targeted R&D for Exascale Deployment

Prototypes, QA

Deployment

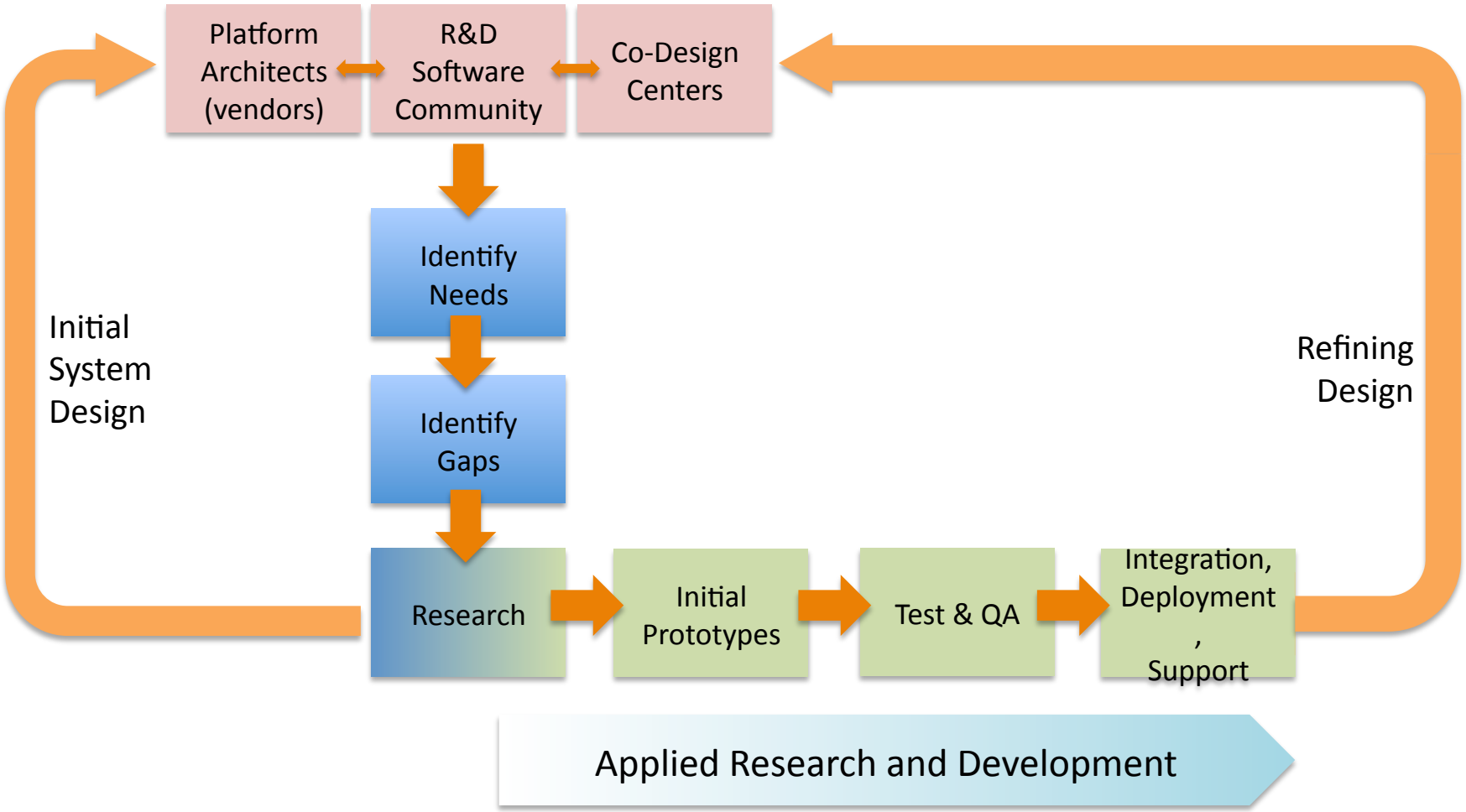
ExaPlatform Support

Commercialization / Productization

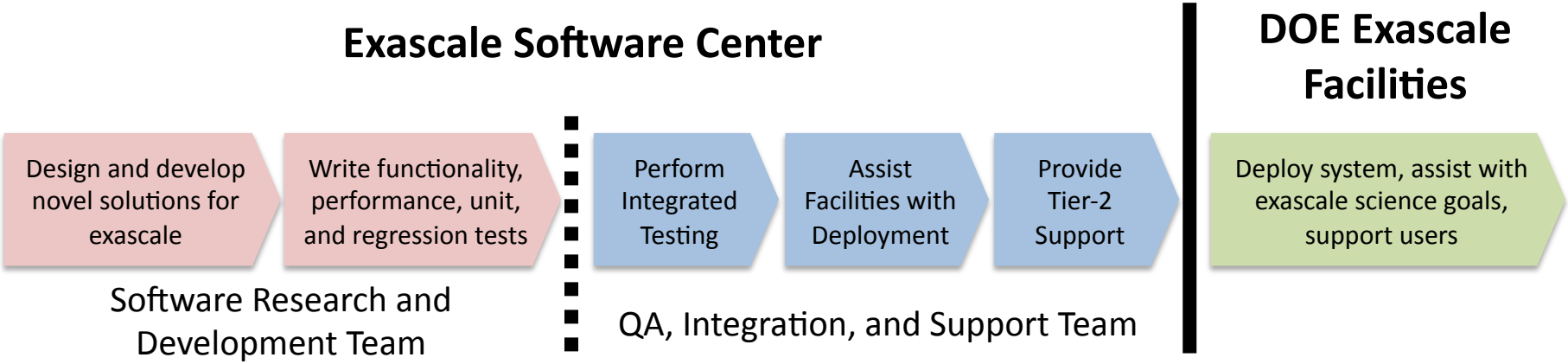
Exascale System Software

This process is heavily weighted to targeted development and applied to key applications and platforms

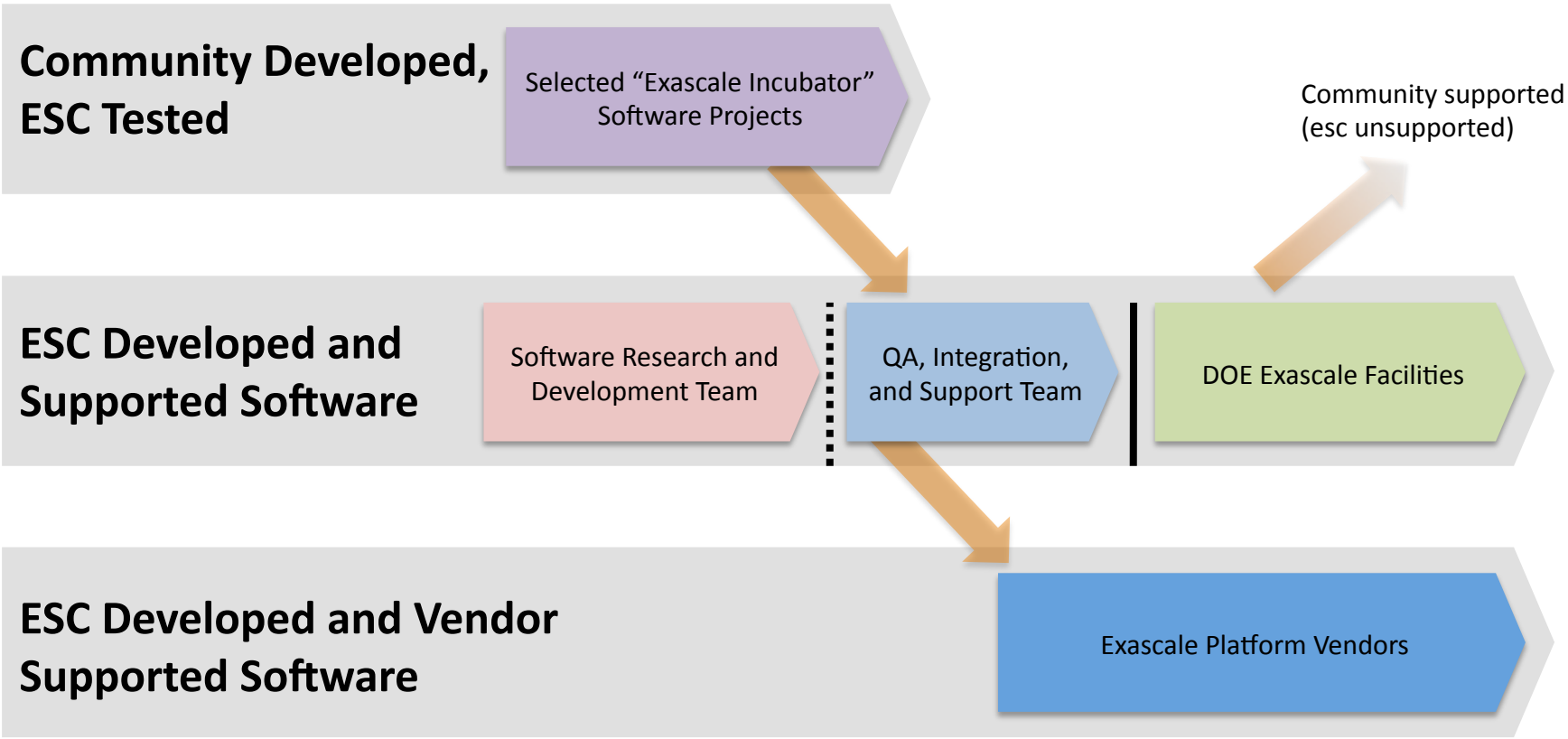
The Exascale Software Center




Esc
18


Exascale Software Center
DOE Exascale Facilities
Design and develop novel solutions for exascale
Write functionality, performance, unit, and regression tests
Perform Integrated Testing
Assist Facilities with Deployment
Provide Tier-2 Support
Deploy system, assist with exascale science goals, support users
Software Research and Development Team
QA, Integration, and Support Team


Esc
18


Community Developed, ESC Tested
Selected "Exascale Incubator" Software Projects
Community supported (esc unsupported)
ESC Developed and Supported Software
Software Research and Development Team
QA, Integration, and Support Team
DOE Exascale Facilities
ESC Developed and Vendor Supported Software
Exascale Platform Vendors

Process for identifying gaps and development targets

- Bottom up (Application Inventory):
  - Provides snapshot of what is currently in use to help frame discussion
  - Provides backdrop for discussion with vendors and applications
- Top down:
  - Application:
    - Science → Capability → Technology → Software
  - Architecture:
    - Design → Functionality → Software
  - Computer Science and Applied Math
    - Algorithm/ProgMod/Technique → Capability → Technology →Software

Current Planning…

- Operating Systems and Run-Time
  - Ron Brightwell
- Programming Models
  - Barbara Chapman
- Tools
  - Bronis R. de Supinski
- Math Libraries
  - Jack Dongarra
- Data and Analysis
  - Rob Ross
- Application Inventory
  - Al Geist

ESC Operating System and Runtime

Participants

ANL: Pete Beckman, Kamil Iskra

SNL: Ron Brightwell

ORNL: Barney Maccabe

Operating System and Runtime are Critical for Exascale

- Resource allocation and management are critical for any system
- Resource challenges
  - Power
  - Massive Intra-node parallelism
  - Memory hierarchy
  - Heterogeneity
  - Resiliency
- Application challenges
  - Expanded paradigms
    - Many-task, analysis-driven, data-driven
- Usage model
  - Space-shared, pipelined, data flow

OSR Components

- Linux integration (Iskra)
- Advanced OS functionality (Brightwell)
- Lightweight threading (Brightwell)
- I/O forwarding and global services (Iskra)
- OS development environments (Brightwell)

Linux Integration

- Motivation – Exascale OS challenges
  - Scalability, memory hierarchy management, heterogeneity, application complexity, power, hardware failures
- Approach
  - Use Linux
  - Asymmetric kernel
    - Replace time sharing with space sharing
    - Subset of cores dedicated to system services
    - Kernel need not scale across all cores
  - Memory management
    - Large memory pages
  - Scheduling
    - Eliminate preemption and migration
    - Reduce OS noise

Linux Integration (cont'd)

- Collaboration
  - Re-use techniques developed within ZeptoOS
  - ASCR research projects
  - BSC High-Performance Linux project
  - Flash HEDP Co-Design Center
  - Vendors
- Support and Integration
  - Base changes on long-term supported Linux distribution such as RHEL
  - Vendors provide first level of support
  - ESC responsible for second level for ESC components

Advanced OS Functionality

- Motivation
  - Need standard APIs for
    - Memory management
      - Stacked DRAM, on-package SDRAM, NVRAM
      - Hardware managed (cache), software managed (scratchpad)
      - Multiple NUMA domains, coherent, non-coherent
    - Power measurement and control
  - Resiliency within the OS itself
  - New application and system usage models
    - In-situ storage and analysis
    - Many-task model

Advanced OS (cont'd)

- Approach
  - Develop portable APIs for enhanced memory management
  - OS/Runtime interfaces for power management capabilities
  - Characterize OS reliability assumptions and explore alternative strategies
  - Develop mechanisms for functional partitioning of system and compute cores

Advanced OS (cont'd)

- Collaboration
  - ASCR research projects
  - ESC programming models and resilience activities
  - Vendors
- Support
  - ESC supported

Lightweight Threading

- Motivation
  - Exascale systems will have massive amounts of on-node parallelism
  - Lightweight cores need lightweight threads
  - Scheduling and locality challenges
- Approach
  - Develop Sandia Qthreads into well-defined API
  - Explore hardware support mechanisms for lightweight synchronization

I/O Forwarding and Global Services

- Motivation
  - I/O expected to be a key challenge
  - I/O spans multiple types of nodes, networks, and software layers
  - This component addresses need for a unified communication layer that supports efficient and consistent communication among compute and I/O nodes
- Approach
  - Based on existing IOFSL work
  - I/O offloading to system cores
  - Asynchronous I/O, handling I/O transfers in background
  - Burst buffer management of on-node storage

I/O Forwarding (cont'd)

- Collaboration
  - Leverage software developed in IOFSL project
  - NoLoSS project
  - Flash HEDP Co-Design Center
  - Vendors
- Support
  - Expect vendors will fork off reference baseline ESC code and provide first-level support
  - ESC responsible for second-level support

OS Development Environments

- Motivation
  - Access to test beds and prototype systems will be limited
  - Need environments to support exascale OS development and exploration
- Approach
  - Leverage Sandia SST simulation framework and virtualization technology

OS Development (cont'd)

- Collaboration
  - SST and Palacios researchers
- Support
  - ESC will support use of SST as a supported target
  - SST modules contributed back to SST and supported by ESC
  - ESC will work with Palacios team to integrate new capabilities

ESC-Software Stack

Programming Models

Participants:

- University of Houston: Barbara Chapman
- Rice University: John Mellor-Crummey
- UIUC: William Gropp
- ANL: Rajeev Thakur, Pavan Balaji, Ewing Lusk
- LBL (UC Berkeley): Kathy Yelick, Paul Hargrove
- ORNL: Al Geist

IESP meeting-April 2011

A Critical Matter

- MPI-everywhere model no longer viable
  - Hybrid MPI+OpenMP already in use
- Models and their implementation must take account of:
  - System scale, node parallelism, coherency domains
  - Power concerns, memory hierarchy, locality
  - Fault tolerance
  - Heterogeneity
  - Legacy code, interoperability

Approach

- Ensure availability of programming models most requested by applications developers
  - Make sure that they perform well on exascale systems
- Contribute to enhancement of programming models via participation in standardization / community efforts
- Enable interoperability between programming models
- Develop production-quality resilience model
- Collaborate with tools, OS partners for full solution
- Closely monitor other efforts (research, vendor models)
  - Respond as needed

R&D Topics

- Advanced Programming Models
  - OpenMP
  - CAF
  - UPC
- PLUM Programming Model Interoperability Infrastructure
- MPI
- Resilience and Fault Tolerance

OpenMP

Shared memory model instead of MPI everywhere

- Leader: Barbara Chapman

- Enables direct programming of threads and tasks
  - Reduces memory footprint via sharing of data
- Contribute to standardization of extensions that meet needs of exascale applications
  - Expression of affinity, locality; more powerful task construct
  - Support for heterogeneous nodes, hybrid programming
- Efficient implementation based on OpenUH compiler
- Interact with project partners to:
  - enable interoperability with other models
  - support resilience and fault tolerance
  - provide support for tools

PGAS Programming Models

Global address space within or across nodes

- Leaders: John Mellor-Crummey, Kathy Yelick

- Provides explicit user control of data layout and communication
- Coarray Fortran (CAF)
  - Rice University's CAF extensions (CAF 2.0) and CAF translator
  - Process teams, communication topology, support for overlapping computation and communication, additional synchronization mechanisms
  - Extensions for communication between teams, multithreading, parallel I/O, support for deeper memory hierarchies; Implicit support for offloading computation to heterogeneous cores
- Unified Parallel C (UPC)
  - Starting point is UPC 1.2 specification, Berkeley UPC compiler (BUPC)
  - Explicit asynchronous data movement for overlapping communication with computation, point-to-point synchronization
  - Non-blocking and subset collectives
- Interface to mechanisms that support fault tolerance and resilience; interoperability

Plum: Interoperability Infrastructure

- Leaders: Pavan Balaji and Paul Hargrove
- Problem statement
  - While there are many programming models available today, applications have to pick exactly one model to use
  - Primary reason: different models do not interoperate with each other (either because of semantic or resource management mismatches)
- Goal
  - Composable programming with MPI, OpenMP, UPC, and CAF
    - Can we have an infrastructure where a UPC application can use a PETSc library written with MPI?
    - Can CAF and OpenMP thread tasking models interoperate with each other on the limited computational resources on the node?

Plum Plans

- Plum will develop a runtime infrastructure for communication libraries and threading that will allow MPI, UPC, CAF and OpenMP to interoperate
- ESC will focus on (1) productizing this runtime infrastructure, (2) platform-specific hardening of the runtime, and (3) integration into DOE exascale codesign and other applications

- There are many research challenges that need to be addressed for the broader problem of interoperability including resource management issues, tasking model requirements, etc., which is expected to be separately funded

International Collaborations

- Advanced Programming Models
  - Collaboration with XcalableMP (XMP) effort at University of Tsukuba, Japan (Prof. Sato)
  - Many interactions with Barcelona Supercomputing Center (BSC), Spain on OpenMP language and runtime support
  - Collaboration with Chinese Academy of Sciences with regard to UPC and OpenMP programming models
  - Collaboration with Tsinghua University on deployment and testing of OpenMP
  - Interactions with TU Dresden and German Supercomputing Centre Juelich on compiler support for OpenMP tools

- Plum Programming Model Interoperability Infrastructure
  - Barcelona Supercomputing Center (BSC) - EU

MPI

- Leaders: Rajeev Thakur, Rusty Lusk, Bill Gropp
- ESC's application survey results
  - They intend to continue to primarily use MPI in the future, together with some shared-memory model within a node (MPI + X hybrid)
- All vendors also view MPI as a critical piece of software they need to provide on future HPC systems

What is needed in MPI for Exascale

- Performance scalability
- Reduced memory consumption at scale
- Better support for hybrid programming
- And all the good stuff from MPI-3 and future versions of the MPI standard

MPI-3

- Currently being defined by the MPI Forum
- Expected to be released late next year (2012)
- New Features
  - Better support for hybrid programming
  - New and improved RMA functionality
  - Fault tolerance
  - Nonblocking collectives
  - Sparse (memory optimized) collectives
  - Better support for parallel tools
  - Fortran 2008 bindings
  - Others

MPI in ESC

- ESC will develop a robust, tuned MPI-3 implementation for exascale based on MPICH
- Continue current collaboration model between MPICH group and vendors

- Note that the *research* needed to scale MPI and MPICH to exascale is expected to be performed in other projects supported by the DOE ASCR base program or other parts of the DOE exascale initiative, not in ESC
- ESC will deliver the hardened, production-quality implementation

MPICH Group's International Collaborations

- Europe
  - Guillaume Mercier, Univ of Bordeaux, France
    - Was a postdoc at Argonne and did some of the core work in MPICH2. We consider him a member of the MPICH team.
  - Jesper Larsson Träff, Univ of Vienna, Austria
    - Long-time collaborator. Several joint publications with Thakur, Gropp, Lusk, and others.
  - Franck Cappello, INRIA, France
    - Fault tolerance
- Japan
  - Yutaka Ishikawa, Univ of Tokyo
    - MPI optimizations for new 10PF K supercomputer
  - Satoshi Matsuoka, Tokyo Tech
    - Fault tolerance

MPICH Group's International Collaborations

- China
  - Pavan Balaji from Argonne visited China for 2 months (Nov 2010 – Jan 2011)
  - He has established active collaborations with
    - Yunquan Zhang, Mingyu Chen, and Guangming Tan, Chinese Academy of Sciences (CAS)
    - Yutong Lu, NUDT
    - Gaojin Wen, Shenzhen Inst. of Advanced Technologies
  - Joint papers submitted to ICPP and SC11
  - Argonne will host a summer student from China

Resilience and Fault Tolerance

- Leader: Al Geist
- Fault tolerance of ESC software components is critical for exascale
- Need coordinated effort from all parts of the software stack for fault detection, localization, notification, and recovery
- We will leverage the experience and results of the DOE CIFTS project

Creation of a Fault Model

(A programming model for resilience)

The key to making application and system components resilient lies in **creating a well-defined fault model** that identifies:

- What types of faults or performance degradations will be detected
- What and how the components will be notified
- What supported actions will be available to the components to recover or adapt to degradations of different types
- Quality of service and/or reliability needs of component

In collaboration with the International community, the ESC will develop:

- a holistic fault model
- prototype API, and
- fault-test environment

that allows the behavior of algorithms and mini-applications to be studied in the presence of faults defined in the resilience model.

International Collaborations in Fault Tolerance

- Europe
  - Franck Cappello, INRIA
- Japan
  - Satoshi Matsuoka, Tokyo Tech

ESC Tools

Bronis R. de Supinski, Allen Malony,
John Mellor-Crummey, Jeffrey S. Vetter

Wednesday, April 6, 2011

Our solutions will provide scalable insight for application issues

- Overall strategy
  - Build and extend existing tool community efforts
  - Provide components that support rapid prototyping to address exascale-specific and application-specific tool needs
  - Integrate to support common usage scenarios
    - Large-scale debugging
    - On-node correctness issues
    - Scalable performance analysis

Our flexible toolbox will support application-specific tools

- Wide range of tool component capabilities
  - Execution control
  - Asynchronous measurement
  - Synchronous measurement
  - Problem-focused measurement
  - Correctness introspection
  - Measurement data management
  - Analysis
  - Introspection APIs
  - Presentation

Our underlying infrastructure will solve critical exascale issues

- Coordination to ensure tool requirements are met
  - Across ESC team: Programming Models, I/O and OS
  - With vendors: particularly focused on architectural advances
- Infrastructure components
  - PAPI
    - Support for new processor architectures
    - Network and power counters
    - Programming model-centric metrics (i.e., MPIT)
  - Rose
  - Portable introspection and control
  - Scalable infrastructure
    - LaunchMon
    - MRNet: investigate merging with IOSFL
  - Novel architecture features and heterogeneous tool infrastructure
    - Static and dynamic/JIT instrumentation
    - Data collection

Our integrated correctness and performance tools will provide insight into root causes

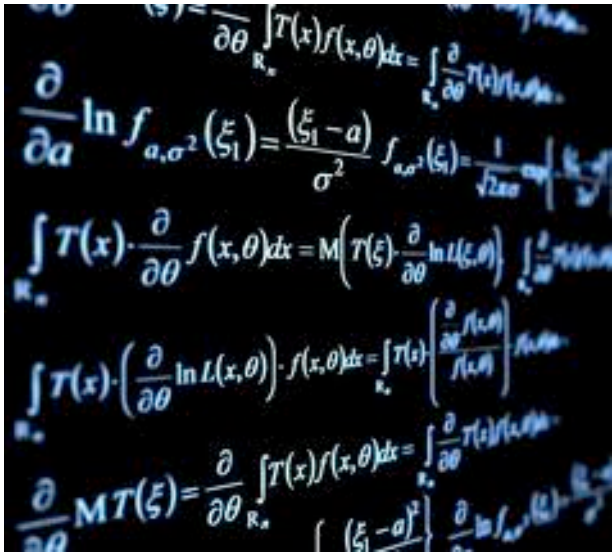
- Correctness tools
  - Identify relationships between behavior equivalence classes to guide root cause analysis
  - Detect programming model semantic errors
- Performance analysis tools
  - Text-based reports and GUI-based front-ends
  - Node-based and communication measurements
  - Attribute bottlenecks to underlying causes
    - SPMD programs
    - Communication
  - Scalable analysis and presentation

We will build on existing tool efforts

Tool/Technology	Components	Infrastructure	Integration
CBTF	F		
Dyninst	F	F	
StackWalker API		F	
SymTab API		F	
ProcControl API		F	
HPCToolkit	F	F	F
LaunchMon		F	
Marmot			T
Memcheck			T
mpiP			F
MRNet		F	
MUST	T		T
Novel Architecture Features		R	
Open|SpeedShop	T		F
PAPI		F	
P^NMPI	F		
Rose		F	
STAT			F
TAU	F		F
Valgrind	T	T	

Our approach facilitates tool community collaboration

- Expect component contributions from the entire HPC software tools community
  - Annually adopt at least one mature research solution
  - Provide framework for further research efforts
- Potential EESI contributions
  - Marmot/MUST for scalable MPI correctness checking
  - Scalasca: late sender analysis component?
  - Many others possible



Math Libraries & Frameworks

ANL, LLNL, SNL, UCB, & UTK

Barry Smith, Rob Falgout, Mike Heroux, Jim Demmel, & Jack Dongarra

Math Library *Software*

- We are relying on the algorithms for exascale systems to be develop in other programs, e.g.,
  - DOE Math/CS Institutes.
  - Exascale Co-Design center collaborations.
  - SciDAC.
  - External community efforts.
- This is not an algorithms research effort.
- This is a software delivery effort.

What Areas to Cover?

- **From Phil Colella's "Seven Dwarfs"**
  - Structured grids (including locally structured grids, e.g. Adaptive Mesh Refinement)
  - Unstructured grids
  - FFTs
  - Dense LA
  - Sparse LA
  - Monte Carlo
  - Particle/N-Body methods

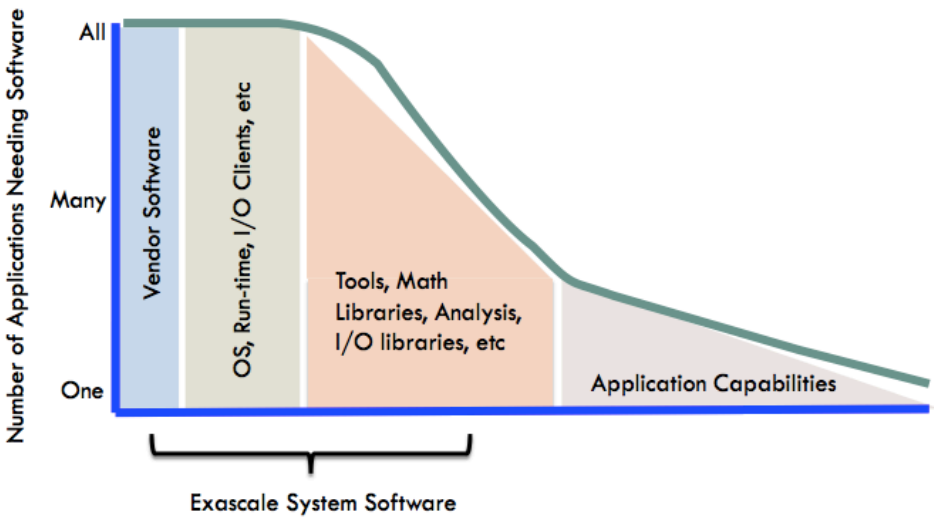
Survey Apps

1. Molecular dynamics (mat)
2. Electronic structure
3. Reactor analysis/CFD
4. Fuel design (mat)
5. Reprocessing (chm)
6. Repository optimizations
7. Molecular dynamics (bio)
8. Genome analysis
9. QMC
10. QCD
11. Astrophysics

- Structured grids
  - 3, 5, 6, 11
- Unstructured grids
  - 3, 4, 5, 6, 11
- FFTs
  - 1, 2, 3, 4, 7, 9
- Dense LA
  - 2, 3, 5
- Sparse LA
  - 2, 3, 4, 6, 8, 11
- Monte Carlo
  - 4, 9
- Particle/N-Body methods
  - 1, 7, 11

Prioritize the Areas to Cover

- First priority
  - Dense LA
    - Ax = b and Ax = λx
  - Sparse LA
    - Ax = b and Ax = λx
    - Direct and iterative
- Second priority
  - FFTs
  - Structured grids
  - Unstructured grids
- Third priority
  - Monte Carlo
  - Particle/N-Body methods



Funding levels and needs will determine what we work on.

- The more funding the more priorities can be developed.

Key Challenges at Exascale

- Levels of parallelism
  - O(100M and beyond)
- Hybrid architectures
  - Node composed of multiple multicore sockets + accelerators
- Bandwidth vs Arithmetic rate
  - Most approaches assume flops expensive
- Storage Capacity
  - Issue of weak scalability in future systems
- Fault occurrence; shared responsibility
  - Process failure recovery
- Power Management
  - API for fine grain management
- Language constraints
  - Fortran, C & MPI, Open-MP
- Autotuning
  - Systems complex and changing
- Bulk Sync Processing
  - Break fork join parallelism
- Lack of reproducibility; unnecessarily expensive (most of the time)
  - Can't guarantee bitwise results
- Need for effective scheduling of tasks

Potential Impact on Software Components

- Efficient libraries of numerical routines
- Agnostic of platforms
- Self adapting to the environment
- Libraries will be impacted by compilers, OS, runtime, prog env, etc
- Standards needed: FT, Power Management, Hybrid Programming, arch characteristics

How to Frame Evolutionary & Revolutionary

- For the 10 Petaflops systems will begin with the current base of software adapting for optimization.
- For the 100 Petaflops systems will replace key components with evolving research prototypes
- For the Exascale systems we envision a complete overhaul of the libraries

Numerical Libraries

- Uniquely exascale
  - Async methods
  - Hybrid and hierarchical based algorithms (eg linear algebra split across multi-core and gpu, self-adapting)
  - Algorithms that minimize communications and synchronization
  - Fault oblivious, Error tolerant software

- Exascale + trickle down
  - Autotuning based software
  - Standardization activities
  - Energy efficient algorithms
  - Mixed arithmetic

Adaptivity for architectural environment

- Objective is to provide a consistent library interface that remains the same for users independent of scale and processor heterogeneity, but which achieves good performance and efficiency by binding to different underlying code, depending on the configuration.
- Deal with the complexity of discovering and implementing the best algorithm for diverse and rapidly evolving architectures.
- Automating the process, both for the sake of productivity and for correctness.

Auto-Tuning

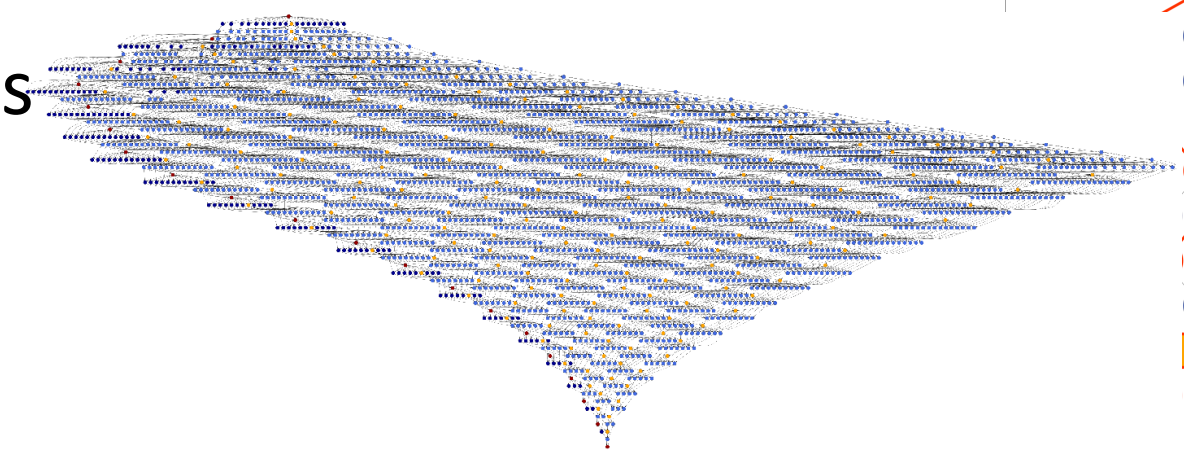
- Best algorithm implementation can depend strongly on the problem, computer architecture, compiler,…
- There are 2 main approaches
  - Model-driven optimization
    [Analytical models for various parameters;
    Heavily used in the compilers community;
    May not give optimal results]
  - Empirical optimization
    [Generate large number of code versions and runs them on a given platform to determine the best performing one;
    Effectiveness depends on the chosen parameters to optimize and the search heuristics used]
- Natural approach is to combine them in a hybrid approach
  [1st model-driven to limit the search space for a 2nd empirical part]
  [Another aspect is adaptivity – to treat cases where tuning can not be restricted to optimizations at design, installation, or compile time]

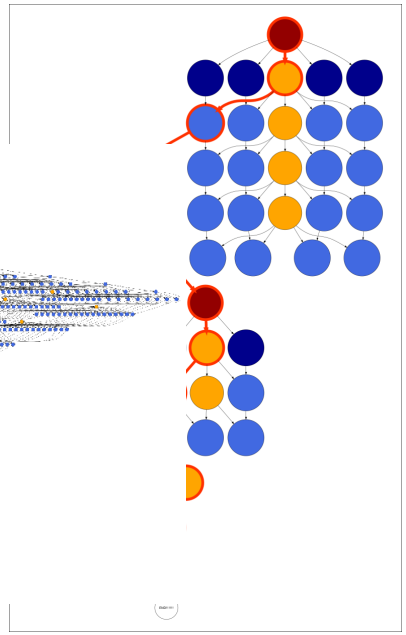
Scalability : Need Algorithms with Minimal Communication

- Communication (moving data) is increasingly bottleneck
- Goal: use algorithms that minimize communication & synchronization
  - Dense Direct
    - Incorporate known optimal algorithms for case of homogeneous machines, one-copy of data overall
    - As needed, add algorithms that communicate less by using more memory per processor (helps strong scaling)
    - As needed, extend to heterogeneous case
  - Sparse Direct
    - Fewer optimal algorithms known (Cholesky), add as needed
  - Sparse Iterative
    - Incorporate know optimal Krylov subspace methods
    - Target problem needs (eg bottom-solvers in multigrid, preconditioning)

Increasing the Level of Asynchronous Behavior

- DAG level description of methods
  - expressing parallelism explicitly in DAGs, so that scheduling tasks dynamically, support massive parallelism, and apply common optimization techniques to increase throughput.
- Scheduler needed
- Standards





Mixed Precision & Reproducibility

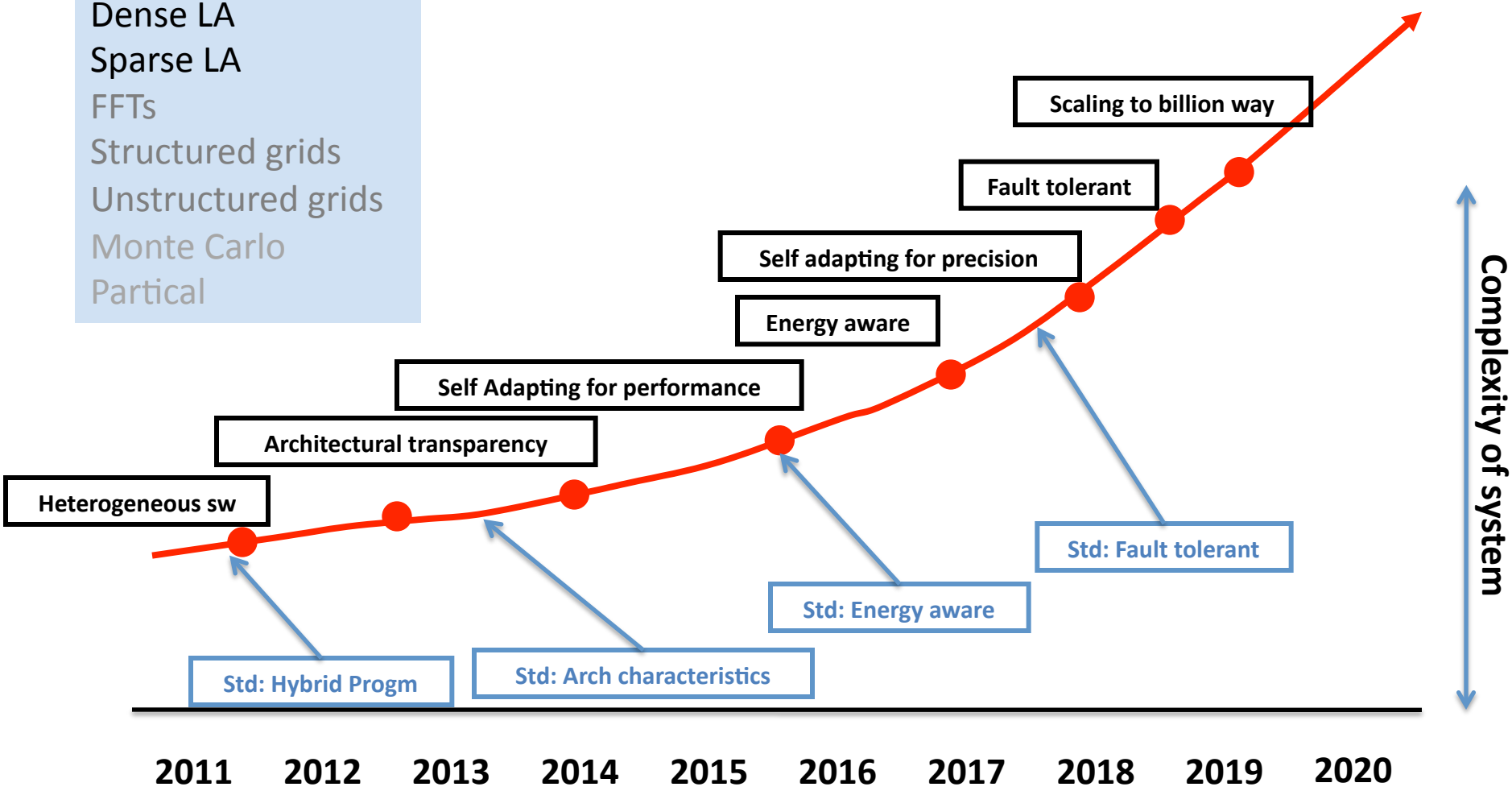
- Single Precision is 2X faster than Double Precision
  - With GP-GPUs 8x
- Power saving issues
- Reduced data motion
  - 32 bit data instead of 64 bit data
- Higher locality in cache
  - More data items in cache
- Want to provide way to guarantee reproducibilty
  - May increase the time to solution

ESC-EESI Collaborations

- Università di Roma, Salvatore Fillipone
  - OO Fortran Libraries
- ETH – Zurich, Peter Arbenz
  - Scalable Preconditioners.
- Barcelona group
  - StarSs (SMPSs)
- Bordeaux group
  - StarPU
- NAG Ltd
  - Numerical library

Numerical Libraries & Frameworks

Numerical Libraries
Dense LA
Sparse LA
FFTs
Structured grids
Unstructured grids
Monte Carlo
Partical



The Exascale Software Center: Data Storage and Analysis

Jim Ahrens (LANL)
Shane Canon (LBNL)
Hank Childs (LBNL)
Gary Grider (LANL)
Rob Ross (ANL)
Galen Shipman (ORNL)
Lee Ward (SNL)

What are the Exascale Data Challenges?

- **Bulk data movement**
  - Eliminating/minimizing data movement to external storage when possible
  - Eliminating unnecessary synchronization in movement to external storage
- **Efficient data triage and analysis**
  - Perform as much analysis/reduction as possible before writing data to external storage
  - In situ requires application to expose in-memory data structures, motivating standard APIs
  - Co-processing requires hooks into I/O middleware or storage software
- **Data management**
  - Very large numbers of files (i.e., namespace)
  - Metadata and provenance capture
  - End-to-end data integrity
- **Input data staging**
  - Pre-staging of input datasets onto storage in the system
  - Includes dynamic loading of libraries
- **Scalable, high productivity interfaces to data**
  - Libraries that provide high productivity interfaces to storage
  - Future interfaces and semantics of storage, coordination and synchronization

Evolutionary and Revolutionary Improvements

- Both evolutionary and revolutionary work contribute to solutions
  - Evolutionary research can result in new software products in existing technology areas (e.g., a new application interface)
  - Revolutionary research can result in new technologies that provide alternative paths to required capabilities (e.g., a new storage model)
- A plan for Exascale data that only leverages evolutionary R&D carries great risk
- We advocate two development tracks through the 2016 time frame
  - Evolutionary track technologies can be used in interim system deployment with appropriate enhancements, may be viable for Exascale
  - Revolutionary track develops new storage prototype, provides risk mitigation for Exascale systems
- Critical decision point in 2016 time frame used to choose most promising approaches to specific challenges

Some Assumptions

- Rotating media capacity and performance will continue on roughly the same curve as from the last five years
- Community will continue to depend on and invest in vendor-supplied parallel file systems and archive solutions, resulting in modest scalability improvements
- Solid-state storage will be integrated into compute system architectures in the interim system time frame
- Research efforts in related areas (e.g., ADIOS, Damsel, GLEAN) will be tracked and considered for integration into plan when appropriate

Critical Technologies and Candidate Software

Component Areas	Challenges Addressed	Technologies	Software Starting Points
Resiliency and Checkpoint	Bulk Data Movement (Fault Tolerance)	End-to-End Integrity In-System Checkpoint Scalable Fault Detection	SCR, PLFS
Libraries and Abstractions	High Productivity Interfaces Bulk Data Movement Input Data Staging Data Management	High-Level I/O Libraries File System Transformation Object Storage	HDF5, PLFS
Analysis Infrastructure	Data Triage and Analysis Bulk Data Movement High Productivity Interfaces	Many-core Algorithms Accerator-Based Algorithms In Situ Data Reduction In-Storage Data Processing	VTK
Low Level Storage	Bulk Data Movement Input Data Staging	Storage and Network Libraries Data Placement Algorithms Burst Buffer Management	IOFSL
Hierarchical Storage	Bulk Data Movement Input Data Staging Data Management	Archival Storage Multi-Tier Storage	
Metadata	Data Management Data Triage and Analysis	Provenance Capture Alternative Namespaces Scalable Directory Support	Giga+

Roadmap

- Resilience: Near-term focus areas
  - In-system checkpoint
  - Fault management
  - End-to-end integrity
- Libraries and Abstractions: Near-term focus areas
  - Scaling of the HDF5 library to enable exascale capability runs
  - Incorporation of PLFS file system transformations, indexing, and exploitation of in system storage into HDF5
- Analysis Infrastructure: Near-term focus areas
  - Shared memory parallelism for multi-core architectures
  - Data parallelism at extreme scale for GPU architectures
  - Native support for in situ analysis
  - Analysis integration into the storage system
- Low Level Storage: Near-term focus areas
  - Core storage infrastructure (with connections to OS R&D)
  - Data placement and autonomic storage
  - Burst buffer management (with connections to OS R&D)
- Hierarchical Storage: Near-term focus areas
  - Assessment of enterprise archival storage viability
  - Integration of in-system resources as tiers of storage prototype storage
  - Integration of enterprise storage into storage prototype
- Metadata: Near-term focus areas
  - Improved large directory support on parallel file systems
  - Automatic provenance capture
  - Search and alternative namespaces

Opportunity for International Collaboration

- One opportunity for continued international collaboration is in **portable I/O forwarding software**
- IOFSL project in the U.S. (ANL, LANL, ORNL, SNL) has been collaborating with Yutaka Ishikawa (Univ. of Tokyo) and Takashi Yasui (Hitachi)
  - IOFSL project provides basic I/O forwarding infrastructure (protocol, clients, server implementation)
  - Ishikawa/Yasui team implementing caching in IOFSL system
  - Target system is T2K machine (~950 compute nodes, HSFS file system)

Applications Inventory
History and Purpose

Also called "Co-Design Code Team Survey"

- A copy of the survey is in your IESP folder

Put together by the US Exascale Software Center

- Results collected over the past few months

Purpose

1. Find out what software the application teams are using now
2. Find out what software they believe they will need to achieve their goals for Exascale science
3. Identify the most important areas for co-design with the developers of a Exascale software stack

Applications Inventory

What the Survey Asked

Comprehensive Survey – over 7 pages of questions

Application overview

- Current size of codes, languages, methods
- Expectation for Exascale science

Parallel Programming

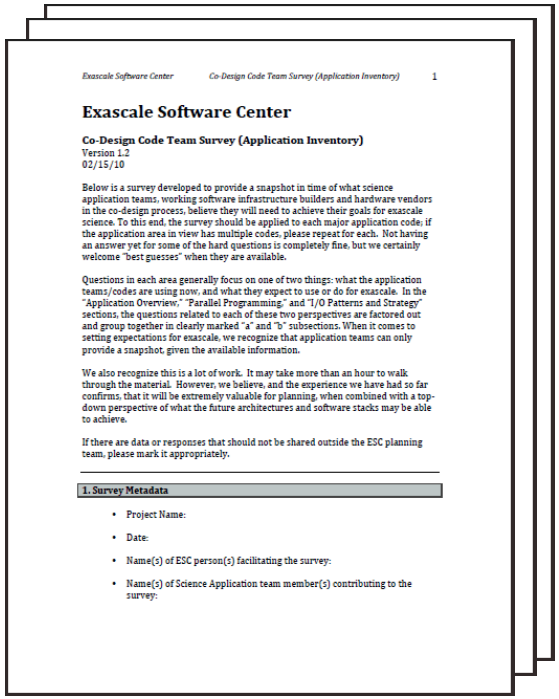
- Scalability, fault tolerance, memory needs,…
- Locality, data communication methods,…

I/O, Data Analysis, Visualization

- Data access patterns, file requirements,
- Output size, analysis needs, viz workflow,…

Tools

- Performance, Debugging, special tools needs?



Exascale Software Center *Co-Design Code Team Survey (Application Inventory)* 1

Exascale Software Center

Co-Design Code Team Survey (Application Inventory)
Version 1.2
02/15/10

Below is a survey developed to provide a snapshot in time of what science application teams, working software infrastructure builders and hardware vendors in the co-design process, believe they will need to achieve their goals for exascale science. To this end, the survey should be applied to each major application code. If the application area in view has multiple codes, please repeat for each. Not having an answer yet for some of the hard questions is completely fine, but we certainly welcome "best guesses" when they are available.

Questions in each area generally focus on one of two things: what the application teams/codes are using now, and what they expect to use or do for exascale. In the "Application Overview," "Parallel Programming," and "I/O Patterns and Strategy" sections, the questions related to each of these two perspectives are factored out and group together in clearly marked "a" and "b" subsections. When it comes to setting expectations for exascale, we recognize that application teams can only provide a snapshot, given the available information.

We also recognize this is a lot of work. It may take more than an hour to walk through the material. However, we believe, and the experience we have had so far confirms, that it will be extremely valuable for planning, when combined with a top-down perspective of what the future architectures and software stacks may be able to achieve.

If there are data or responses that should not be shared outside the ESC planning team, please mark it appropriately.

1. Survey Metadata

- Project Name:
- Date:
- Name(s) of ESC person(s) facilitating the survey:
- Name(s) of Science Application team member(s) contributing to the survey:

For all areas asked what are expected needs for Exascale

Applications Inventory
Who It was Sent to

Submitted to the 7 DOE Co-Design Centers (Thur. talks)

- Climate
- Chemistry
- Combustion
- Materials
- Fusion
- Nuclear Energy
- High Energy Physics

Results included details and Exascale expectations of over 40 HPC applications

Submitted to the DOE NNSA Code Teams

- National Security
- Device Design

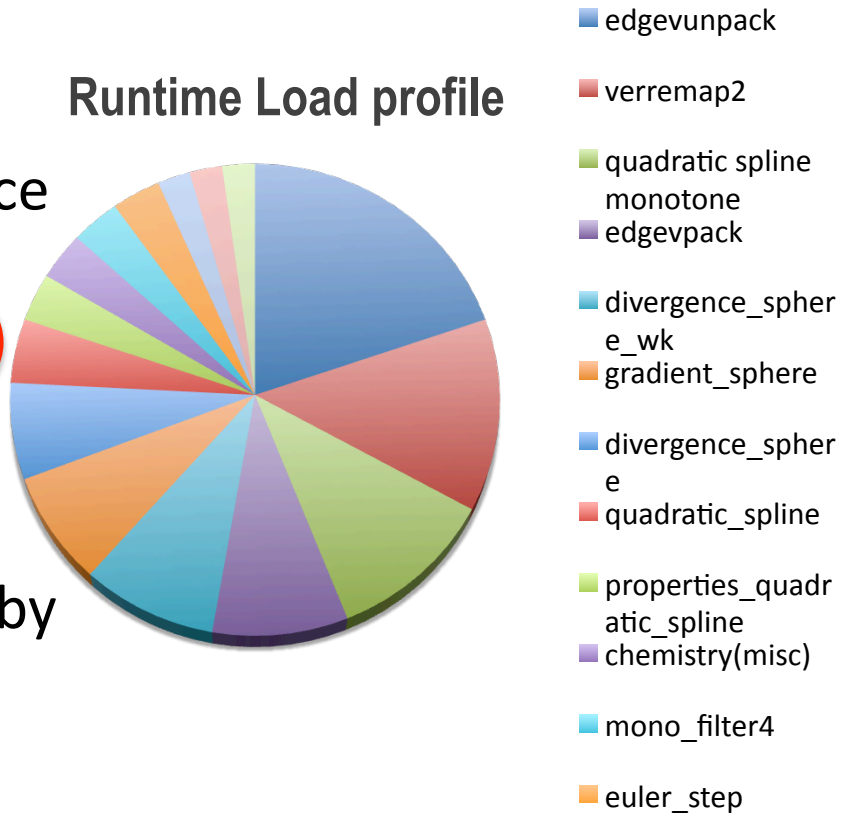
We would love for EU and Japan application teams to take survey

Climate Survey Results

- **Challenges Today**
  - Climate-specific analysis/viz tools don't scale
  - Model complexity and configuration management
  - Slow archiving and file system instability
  - Performance variability
- **Challenges at Exascale**
  - Making effective use of heterogeneous nodes
  - Balancing output with science goals
  - Machine limits and post-processing capabilities for increased data volumes at high spatial and temporal resolutions

Climate:
Learning about Methods and Bottlenecks

- **To better Performance:**
  - Node and PE level performance
  - Communication bound
  - Computational load is evenly distributed across many modules
- **To better Scaling:**
  - Scaling will be mostly limited by problem size
  - Currently can scale to 100K cores
  - Exascale-class cloud resolving problems should scale to 50M cores



Key Chemistry Challenges

- **Application computational kernels are typically compute-intensive rather than memory intensive (opposite climate)**
  - Memory bandwidth isn't always a limiting factor
  - Ability to exploit physical locality is critical
  - Support for hierarchical data decomposition is necessary
- **We are unprepared to deal with hard and soft error rates**
  - Challenge is to tie any soft error to a place in time so that computation can resume from a prior checkpoint
  - What about undetected errors?
- **Portable code**
  - Advances in code generation and transformation are critical

Key Combustion Challenges

- **Memory per Core is critical to problem size and performance**
- **Need better ways to program many-core/hybrid nodes**
- G**lobal synchronizations** required by elliptic solvers after each time-step
- **Hierarchical load balancing** (dynamic)
  - Different strategies for inter-node and intra-node
- **Data volume** will require 'online' analysis and/or data reduction
- **Fault tolerance**
  - "Tell me the difference between bugs and faults"
- **Complexity of future node architecture**s
  - No common abstract machine model for 'ease of use'

What the Fusion Codes need for Exascale

- Programming Models
  - Support for Fortran, C, and C++, MPI, OpenMP, CUDA and OpenCL
- Evaluation of new HW designs on DOE apps (simulators)
  - Identify hot-spots in apps (automated kernel extraction)
  - New compiler optimizations to leverage exascale features
  - Autotuning to support power and performance optimization
  - New forms of analysis for Exascale architecture specific issues (power)
- Development of Compact Apps to drive research
- Development of Skeleton Apps to drive simulators

And a Few Closing Thoughts From the Survey responses

- Applications are historically insensitive to OS, and most system software issues
- Apps require MPI, threads, for some asynchronous I/O, but nothing exotic is expected.
- Don't expect smart vendor compilers
- Do expect a programming model from the vendor that requires large user input and significant rewrite of application codes.

Applications Inventory
History and Purpose

Also called "Co-Design Code Team Survey"

- A copy of the survey is in your IESP folder

Put together by the US Exascale Software Center

- Results collected over the past few months

Purpose

1. Find out what software the application teams are using now
2. Find out what software they believe they will need to achieve their goals for Exascale science
3. Identify the most important areas for co-design with the developers of a Exascale software stack

Applications Inventory

What the Survey Asked

Comprehensive Survey – over 7 pages of questions

Application overview

- Current size of codes, languages, methods
- Expectation for Exascale science

Parallel Programming

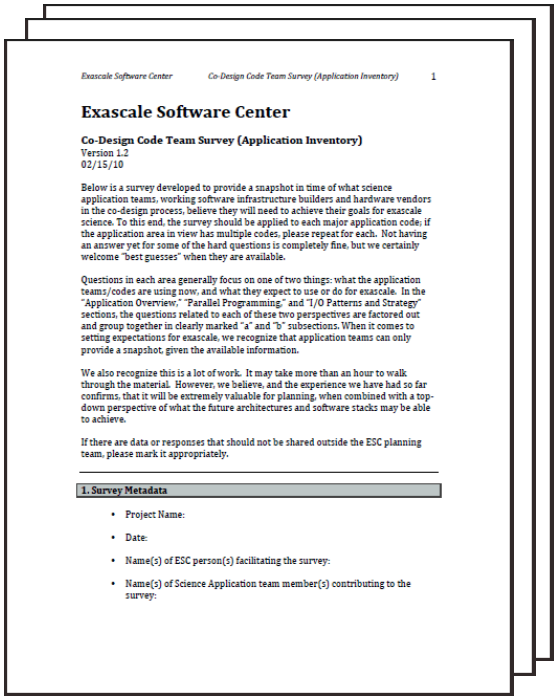
- Scalability, fault tolerance, memory needs,…
- Locality, data communication methods,…

I/O, Data Analysis, Visualization

- Data access patterns, file requirements,
- Output size, analysis needs, viz workflow,…

Tools

- Performance, Debugging, special tools needs?



Exascale Software Center *Co-Design Code Team Survey (Application Inventory)* 1

Exascale Software Center

Co-Design Code Team Survey (Application Inventory)
Version 1.2
02/15/10

Below is a survey developed to provide a snapshot in time of what science application teams, working software infrastructure builders and hardware vendors in the co-design process, believe they will need to achieve their goals for exascale science. To this end, the survey should be applied to each major application code; if the application area in view has multiple codes, please repeat for each. Not having an answer yet for some of the hard questions is completely fine, but we certainly welcome "best guesses" when they are available.

Questions in each area generally focus on one of two things: what the application team/codes are using now, and what they expect to use or do for exascale. In the "Application Overview," "Parallel Programming," and "I/O Patterns and Strategy" sections, the questions related to each of these two perspectives are factored out and group together in clearly marked "a" and "b" subsections. When it comes to setting expectations for exascale, we recognize that application teams can only provide a snapshot, given the available information.

We also recognize this is a lot of work. It may take more than an hour to walk through the material. However, we believe, and the experience we have had so far confirms, that it will be extremely valuable for planning, when combined with a top-down perspective of what the future architectures and software stacks may be able to achieve.

If there are data or responses that should not be shared outside the ESC planning team, please mark it appropriately.

1. Survey Metadata

- Project Name:
- Date:
- Name(s) of ESC person(s) facilitating the survey:
- Name(s) of Science Application team member(s) contributing to the survey:

For all areas asked what are expected needs for Exascale

Applications Inventory
Who It was Sent to

Submitted to the 7 DOE Co-Design Centers (Thur. talks)

- Climate
- Chemistry
- Combustion
- Materials
- Fusion
- Nuclear Energy
- High Energy Physics

Results included details and Exascale expectations of over 40 HPC applications

Submitted to the DOE NNSA Code Teams

- National Security
- Device Design

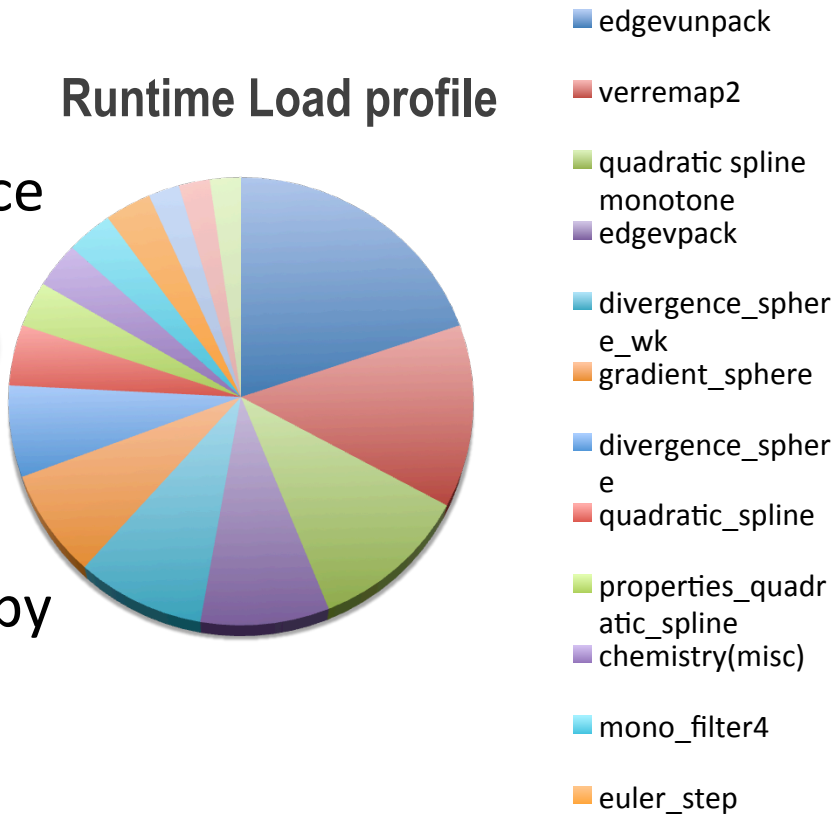
We would love for EU and Japan application teams to take survey

Climate Survey Results

- **Challenges Today**
  - Climate-specific analysis/viz tools don't scale
  - Model complexity and configuration management
  - Slow archiving and file system instability
  - Performance variability
- **Challenges at Exascale**
  - Making effective use of heterogeneous nodes
  - Balancing output with science goals
  - Machine limits and post-processing capabilities for increased data volumes at high spatial and temporal resolutions

Climate:
Learning about Methods and Bottlenecks

- **To better Performance:**
  - Node and PE level performance
  - Communication bound
  - Computational load is evenly distributed across many modules
- **To better Scaling:**
  - Scaling will be mostly limited by problem size
  - Currently can scale to 100K cores
  - Exascale-class cloud resolving problems should scale to 50M cores



Key Chemistry Challenges

- **Application computational kernels are typically compute-intensive rather than memory intensive (opposite climate)**
  - Memory bandwidth isn't always a limiting factor
  - Ability to exploit physical locality is critical
  - Support for hierarchical data decomposition is necessary
- **We are unprepared to deal with hard and soft error rates**
  - Challenge is to tie any soft error to a place in time so that computation can resume from a prior checkpoint
  - What about undetected errors?
- **Portable code**
  - Advances in code generation and transformation are critical

Key Combustion Challenges

- **Memory per Core is critical to problem size and performance**
- **Need better ways to program many-core/hybrid nodes**
- G**lobal synchronizations** required by elliptic solvers after each time-step
- **Hierarchical load balancing** (dynamic)
  - Different strategies for inter-node and intra-node
- **Data volume** will require 'online' analysis and/or data reduction

- **Fault tolerance**
  - "Tell me the difference between bugs and faults"
- **Complexity of future node architecture**s
  - No common abstract machine model for 'ease of use'

What the Fusion Codes need for Exascale

- Programming Models
  - Support for Fortran, C, and C++, MPI, OpenMP, CUDA and OpenCL
- Evaluation of new HW designs on DOE apps (simulators)
  - Identify hot-spots in apps (automated kernel extraction)
  - New compiler optimizations to leverage exascale features
  - Autotuning to support power and performance optimization
  - New forms of analysis for Exascale architecture specific issues (power)
- Development of Compact Apps to drive research
- Development of Skeleton Apps to drive simulators

And a Few Closing Thoughts From the Survey responses

- Applications are historically insensitive to OS, and most system software issues
- Apps require MPI, threads, for some asynchronous I/O, but nothing exotic is expected.
- Don't expect smart vendor compilers
- Do expect a programming model from the vendor that requires large user input and significant rewrite of application codes.