Applications Breakout Group:

"What CDVs want"

IESP Workshop

Maui, 18-19 October 2010

David Keyes, faciliator

Today's Charge to Apps Breakout

- "What do the apps teams need from the software and hardware sides?"
  - Simply stated, but complex in scope due to breadth of applications and breadth of demands
- Following the so-far productive tradition of Paris, Tsukuba, and Oxford, we structure our replies to the charge in terms of a "basis set" of concrete Co-Design Vehicles (CDV)

21 Candidate Co-Design Vehicles

- Caveat: Neither a complete nor an independent basis, but productive to consider
  - Interesting feature: 7 of the CDVs represented here today are Gordon Bell prize winners or finalists
- Discrete algorithms are underrepresented, so far
- Not necessarily representative of IESP collaborating country research priorities or proportions for exascale simulation-enabled or data-enabled computing

Applications Inventory - 21

- Magnetically Confined Fusion
  - Ethier, Princeton PPL
  - Guenter, Jenko & Heinzel, Max Planck Inst.
  - Koniges, LBNL
  - Nakashima, Kyoto University
- Molecular Dynamics
  - Zhong, Supercomputer Center, CAS
  - Swaminarayan, LANL
  - Streitz, LLNL
- Climate
  - Aloisio, Univ. of Salento & CMCC
- Combustion
  - Sankaran (Messer), ORNL
- Radio Astronomy
  - Cornwell and Humphreys, CSIRO
- Aerodynamics
  - Keyes, KAUST & Columbia
- Fluid Dynamics and Heat Transfer
  - Fischer, ANL
- Neutron Transport
  - Siegel, ANL
- Nuclear Fuel Assemblies
  - Berthou, EDF
- Aerodynamics and Combustion
  - Andre, CERFACS
- HEDP and Rad Hydro
  - Graziani, U Chicago
  - Messer, ORNL
- Electronic Structure
  - Scheffler, Blum, Heinzel, Fritz-Haber-Inst.
  - Eisenbach (Messer), ORNL
  - Harrison, ORNL

Candidate CDV characterization

- Programming Model
  - message-passing, shared memory, hybrid
- Current scaling performance and percentage of peak
  - high-water mark scalability on a node and between nodes
- Algorithm dependencies
  - PDEs, IEs, N-body, FFT, FMM, etc.
- Software library dependencies
  - ScaLAPACK, PETSc, Zoltan, VisIt, etc.
- Balance of hardware resources
  - Input/Output vs. Computation
  - Communication vs. Computation
  - Synchronization vs. Computation
- Scaling requirements of memory with flop/s
  - Strong (if relevant)
  - Weak

PI	Application	Programming Model	Scaling	Cores per node	% Peak	BW limited?	Algorithmic Dependencies	Libraries	I/O vs Comp	Comm vs. Comp.	Synch vs. Comp.	Memory vs. Flop/s
Ethier	Fusion, PIC	hybrid DD and particle dist.	no limit particle grid replicated	12	10%		PDEs: equil w MG	ParMetis PETSc Hypre	important: pre/post		frequent	log-linear
Guenter	Fusion, PIC	OpenMP/MPI hybrid GPUs under investigatio	260K no known limit	32	10%		FFTs PDE:3+2D PIC, Monte Carlo	BLAS ScaLAPACK	important: pre/post	latency crit. BW less crit.	frequent	log-linear
Koniges	Fusion, Gyro	MPI w/hierarchical splitt	no known limit	4 to 8	10%		PDE: 3+2D, struct collision, unstruct	FFTW UMFPACK BLAS ScaLAPACK	not limiting			log-linear
Nakashima	Fusion, PIC	OpenMP/MPI hybrid GPUs useful in phases bulk synch SPMD DD	10K vulnerable to imbalance	16	10%	yes	PDE: equil PIC	RNG		linear	per step	log-linear
Aloisio	Climate	OpenMP/MPI hybrid nested SPMD within MPMD DD with space-filling curves	86K	32			PDEs: evol PDEs: evol	FFTW PETSc ScaLAPACK Trilinos	important: pre/post	superlinear	per step	log-linear
Zhong	MD	DD and particle dist. GPUs	2K	8			N-body FFT FMM	FFTW	not limiting	superlinear	frequent	log-linear
Swaminarayar	MD	hybrid GPU	100K	48	18-36%	yes			not limiting			log-linear
Cornwell	Radio Astro	SPMD DD	1M threads no known limit	3	20-40% 5% GPU		FFT	BLAS GRAPHLAB	EB per day	can be limiting		log-linear
Keyes	Aerodynamics	partial hybrid bulk synch SPMD DD	no known limit	4	5-15%	yes	PDE: equil	ParMetis PETSc	not limiting	const w/weak	frequent	log-linear (min 1K vert/proc)
Fischer	Fluid/Heat Transfe	partial hybrid bulk synch SPMD DD MPI	260K no known limit	4		yes	PDE: equil	VisIt	important: pre/post	const w/weak	frequent	log-linear (min 5K vert/proc)
Siegel	Neutron Transport	domain/energy/angle decomp MPI			5-8%			PETSc ParMetis Triangle	important: pre/post			log-linear (min 1K vert/proc)
Graziani	HEDP with FLASH	bulk synch SPMD DD MPI	130K, incl I/O		8-15%		PDE: equil AMR	PARAMESH Chombo, PETS	important: pre/post output order of mag smaller than		frequent amenable to weaker	log-linear
Scheffler	Electronic Structu DFT	OpenMP/MPI hybrid	40K		10-15%		eigensolver	BLAS ScaLAPACK	important: pre/post		frequent	linear
Berthou	CFD for Nuclear R	SPMD DD	>10K				PDE: equil PDE: evol					log-linear
Andre	Gas Turbines	nested SPMD within MPI	100K					PALM	important: pre/post			log-linear
Sankaran/Che	Combustion	bulk synch SPMD DD limited hybrid	224K	6	5-10%	yes	unrolled integrator PDE: evol		periodic dump small input		per step	log-linear
Eisenbach	First Principle The	MC "walkers" outside of bulk synch SPMD DD	224K	6	75%			BLAS (ZGEMM ScaLAPACK	important: pre/post			log-linear (multiple atoms per wa
Messer	Astro Rad Hydro	bulk synch SPMD DD OpenMP for energy/angle	90K	6	5-15%	yes	PDE: equil FFT AMR pointwise	PETSc SILO	important: pre/post			log-linear
Streitz	Classical MD	bulk synch SPMD DD hybrid possible over physics	300K	4	30%		N-body FFT				per step	llinear

Dominant findings on models and scaling

- We are predominantly bulk synchronous, MPI, and either SPMD domain-decomposed, particle-decomposed, or other object-decomposed
- Electronic Structure codes, however, are dominantly *not* MPI, but global shared memory (e.g., GlobalArrays, Linda)
- Other models are Charm++ (NAMD) and distributed objects built on top of active messages and Pthreads
- Some codes have multiple phases with not necessarily compatible load-balanced decompositions
- We are typically already running hybrid MPI/OpenMP with small numbers of cores, up to 48
- We can typically weak-scale without serious loss of scaling out to the edge of today's machines (300K cores), and theoretically beyond

Dominant findings on resources

- Memory bandwidth
  - Majority of the CDVs are already BW limited in most phases with relatively few cores
- Flop/s per byte of storage
  - Generally linear or log-linear, in weak scaling
- I/O versus computation
  - With notable exceptions that are I/O intensive, the majority of our peta-apps need intensive I/O only at start-up and in dumping the output
  - We expect check-pointing to become much more intensive at the exascale, and perhaps limiting, even with users taking charge of check-pointing minimal state

Dominant findings on resources, cont.

- Communication versus computation
  - In weak scaling there is generally a constant ratio of near-neighbor communication to computation, represented by a minimum collections of vertices, cells, particles, etc., per processor
- Synchronization versus computation
  - We require frequent global collectives
    - At least once per timestep in evolutionary codes
    - Even more frequently when implicit linear algebra needs to be performed within each timestep

Core algorithms

- PDEs: equilibrium (implicit)
- PDEs: evolution (explicit)
- FFT
- FMM
- Particles pushing
- Adaptive mesh refinement
- Sparse and dense linear algebra
- ODE integrators

Minimal Library dependencies

- MPI, GlobalArrays, GasNet
- ParMetis
- BLAS, ScaLAPACK
- UMFPACK, SuperLU, MUMPS
- PETSc, hypre, Trilinos, SUNDIALS
- Chombo, SAMRAI
- FFTW
- GraphLib
- VisIt, VTK
- Triangle
- PALM
- SILO, ADIOS, HDF5
- BOOST

Priorities for exa-scaled enabling technologies (as of Oxford meeting, augmented today)

Development-related

- **Programming models**
- Compilers
- Debuggers
- Profilers
- Source-to-source transformations*
- Configuration and regression systems
- Unit testing software
- Simulators

Production-related

- **I/O systems**
- **Fault monitoring, application reporting, and recovery**
- Dynamic resource management
- Data management
- Visualization systems
- Frameworks
- Data miners
- Dynamic performance optimization
- Workflow controllers

Model-related

- **Uncertainty quantification**
- **Dynamic load balancing**
- Meshers
- Solvers / integrators
- Adaptivity systems
- Graphs and combinatorial algorithms
- DAG-based task scheduler
- Geometric modelers
- Discretizers
- Partitioners
- Random number generators
- Compression

* to provide some hardware independence for codes

Aggressive adopters wish list for HW, SW

- Programming models that optionally expose machine characteristics such as memory hierarchy, cores, etc.
- Memory related
  - Exposure of memory hierarchy
  - Hardware gather-scatter operations
  - Programmable caches (not just DMA engines: it would be nice to specify a memory access pattern or to mark regions of memory as write-rarely/read-often)
  - Access to DMA
- Compiler related
  - Single source across different swim lanes
  - Auto vectorization
  - Masked SIMD operations
- Power related
  - Ability to power on/off pieces of the hardware
  - Ability to hint on power patterns to hardware
- Tools
  - Tools for early adopters Vs. tools for masses
  - Early adopters will put up with a lot more
  - Tools for hot spot analysis and bottleneck identification
  - Hardware counters available to users

Reactions to discussion

- We did not consider the size of the coherence domain – a very interesting, important, and understudied issue
- We do not think that reproducibility is worth a large trade of power, but reproducible integers are good

Question to HW about “Swim Lanes”

- Exa = GigaHertz * MegaNode * KiloCore
- Replace homogeneous KiloCore with various inhomogeneous designs, including GPUs, custom accelerators, etc.
- Other?

Applications questions to *ourselves*

- Programming models and algorithms
- Fault tolerance
- Reproducibility
- Opportunities in big data

Programming model?

- Prior to possessing exascale hardware, apps groups can prepare themselves by exploring new programming models
  - on manycore and heterogeneous nodes
  - Giga to Tera on the desktop is the hard part of Peta to Exa in the big iron
- Attention to locality and reuse is valuable at all scales
  - will produce performance paybacks today *and* in the future
- Need to loosen up synchronization without violating causality
- Need to find the right trade between frequent small communications to allow fine-grained tasks to fire versus infrequent aggregated communications to avoid latency penalties
- New algorithms and data structures can be explored under the assumption that flop/s are cheap and moving data is expensive
- Bandwidth pressure can also be reduced by considering mixed-precision algorithms, using lower precision wherever possible
  - Can a wide variety of precisions, and dynamic precisions be conveniently expressed and implemented

Fault tolerance?

- Applications people reluctantly recognize that fault tolerance is a shared responsibility
  - It is too wasteful of I/O and processing cycles to handle faults purely automatically
- Different types of faults may be handled different ways, depending upon consequences evaluated by scientific impact
- Strategic, minimal workingset checkpoints can be orchestrated by application developers and users

Reproducibility?

- Applications people realize that bit-level reproducibility is unnecessarily expensive most of the time
- Though scientific outcomes must be run- and machine-independent, we have no illusions about bit-level reproducibility for individual pairs of executions with the same inputs
  - Since operands may be accessed in different orders, even floating point addition is not commutative in parallel and on inhomogeneous hardware platforms; this has been true for a long time
  - A new feature in the context of co-design, with an emphasis on low power (low voltage switching), is that lack of reproducibility may emerge for many *other* (hardware-based) reasons
  - If applications developers are tolerant of irreproducibility for their own reasons, e.g., for validation and verification through ensembles, then this has implications for considering less expensive, less reliable hardware

Big data?

- An exascale machine (once loaded up) is a 32 PB fast store, with lots of processors to graze over it
- There may be lots of interesting things to do with it that we don't do yet, e.g., in data mining in climate modeling, astrophysics, etc.
- Such applications can begin to be explored today in "miniature" on petascale computers with 300 TB

Issues for scaling up CDVs

- Weak-scale applications up to distributed memory limits
  - Proportional to number of nodes
- Strong scale applications beyond this
  - Proportional to cores per node/memory unit
- Scale the workflow, itself
  - Proportional to the number of instances (ensembles)
  - Integrated end-to-end simulation
- Co-design process is staged, with any of these three types of scaling valuable by themselves
- Big question: does the software for co-design factor? Or is all the inefficiency at the data copies at interfaces between the components after a while?

Limitations to be explored by CDVs

- Strong scaling algorithms
  - sufficient coarse grain parallelism but
  - expect load imbalance to be the main problem due to irregular task/data size
  - bulk synchronous algorithms on one million nodes do not tolerate load imbalance much worse than one part per million for a synchronous task
- Single-node performance
  - linear algebra kernels
    - typically come with autotuning
  - nonstandard linear algebra kernels
    - we need the autotuning tools, not just their output

Some major challenges that stay the same in peta to exa

- Poor scaling of collectives due to internode latency
- Poor performance of SpMV due to shared intranode bandwidth
- Poor scaling of coarse grids in hierarchical algorithms due to insufficient concurrency
- Lack of reproducibility due to floating point noncommutativity and algorithmic adaptivity (including autotuning) in efficient production mode
- Difficulty of understanding and controlling vertical memory transfers and coherency domains

The 21 would-be CDVs …

Candidate CDV: Full-f (Gyro)kinetic Particle-in-Cell for MFE, ICF, and IFE

Stephane Ethier, Princeton PPL

- Programming Model
  - SPMD: Domain decomposition, particle distribution, and on-node multi-threading
  - Message-passing for inter-node communication: particle motion and field solve.
  - Fine-grain multi-threading on node
- Current scaling performance and percentage of peak
  - Perfect particle weak scaling. Grid has less concurrency so some replication needed.
  - Gets about 10% of peak. Sensitive to memory latency due to gather-scatter.
- Algorithm dependencies
  - Multi-grid for field solve. Currently use Hypre for kinetic electron simulations
- Software library dependencies
  - PETSc (ParMetis, Hypre)
- Balances
  - Difficult to balance both particle work and grid work for toroidal geometry
  - Low ratio of communication vs. computation
  - Synchronization vs. Computation: frequent synchronization; careful load balancing essential
- Scaling requirements of memory with flop/s
  - Need to improve strong scaling performance of gather-scatter algorithm common to all PIC codes

Candidate CDV: Local and global Plasma Core Simulations of Fusion Devices

Sybille Guenter, Frank Jenko, Stefan Heinzel, MPI

- Programming Model
  - MPI/OpenMP hybrid
  - Shared memory implementation
  - GPU under investigation
- Current scaling performance and percentage of peak
  - Weak-scaling to distributed nodes till 260K cores on a BlueGene system
  - 10% of peak after tuned
- Algorithm dependencies
  - Ensemble of various CFD solvers for 5 dim grid, FFTs
  - Particle in cell approach, Monte Carlo codes in 5 dim phase space
- Software library dependencies
  - Blas, ScaLAPACK
- Balances
  - Extreme low latency for high communication requirements (high bandwidth less decisive)
  - FRequent synchronization
  - Efficient and strong I/O system for handling of large input/output data in the PB range
  - Pre- and post-processing: highly relevant
- Scaling requirements
  - In general weak scaling requirements
  - Multilevel of parallelism: Mixed mode possible to address core / node hierarchy

Candidate CDV: Eulerian Gyrokinetics (GYRO)
Alice Koniges, LBNL

- **Programming Model**
  - MPI with hierarchical splitting of communicators.
  - Linear collisionless advance requires no ALLTOALL.
  - Nonlinear advance requires ALLTOALL only on subgroups 1/16-1/128 of NCORES.
- **Current scaling performance and percentage of peak**
  - Weak-scales to distributed nodes without limit in principle.
  - 10% of peak on 2008 JOULE simulations.
- **Algorithm dependencies**
  - Structured 5-D grid (2-D velocity integration by Gauss-Legendre, spectral/FD spatial stencils)
  - Collision operator on unstructured grid using RBF collision stencil.
- **Software library dependencies**
  - UMFPACK (MUMPS optional), FFTW optional, BLAS/LAPACK.
- **Balances on previous single-core and vector architectures**
  - Communication vs. computation time constant for strong scaling of moderate-size problem.
- **Issues**
  - OpenMP and other multi-core optimizations untried.

Candidate CDV: PIC for Plasma Physics
Hiroshi Nakashima, Kyoto Univ.

- Programming Model
  - Bulk synchronous, SPMD by domain decomposition with a sophisticated load balancer.
  - MPI/OpenMP hybrid isn't tough and fairly efficient. For simple model (e.g. w/o MC collision), GPU will accelerate about a half of computation (particle push) but the other half (current scatter) will be tough.
- Current scaling performance and percentage of peak
  - Weak-scaling up to a few thousands MPI processes is promising. Needs a better load balancer for 10K or more processes.
  - About 10% of peak due to memory bandwidth limit in scanning particles.
- Algorithm dependencies
  - Fundamental PIC with charge conservation method (plus MC collision, Poisson solver, and/or particle/fluid hybrid).
- Software library dependencies
  - OhHelp load balancer (plus good random number generator, linear solver).
- Balances
  - I/O vs. Comp.: Snapshot per 10 or more time-steps looks efficient and scalable.
  - Comm. vs. Comp.: Nearly constant with weak scaling.
  - Synch. vs. Comp.: Needs an all-reduce synch. in each time-step for load balancing.
- Scaling requirements of memory with flop/s
  - Strong-scaling is simply tough and about 500-processes looks the upper limit.
  - Weak-scaling requires dreamingly constant byte/flop (or tough work to improve locality).

Candidate CDV: Climate Change
Giovanni Aloisio, Univ. of Salento & CMCC

- Programming Model
  - Hybrid programming model: shared memory (OpenMP) and distributed memory programming (MPI) model usually adopted.
  - MPMD by functional decomposition (for coupled models) and/or SPMD by domain decomposition (for each component model).
- Current scaling performance and percentage of peak
  - Weak-scaling up to 15K cores for O(10^7) regular grid points to simulate 100 years and 5 scenarios: 0.1 model months per wallclock day on 15K cores are required (33 years required to simulate 100 years)
  - Strong-scaling up to 86K cores by the the integration suitable dynamical cores, based on unstructured or block structured quasi-uniform grids
- Algorithm dependencies
  - PDEs (evolution operators and elliptic operators), Parallel Preconditioners, FFT, etc.
- Software library dependencies
  - ScalaPack, PETsc, Trillinos, FFTW, etc.
- Balances
  - I/O vs. Comp.: Initial and boundary conditions are read once and are resolution dependent
  - Comm. vs. Comp.: Variable ratio, increases with scaling.
  - Synch. vs. Comp.: process synchronization at each time step. Careful load balancing essential (space-filling curves partitioning is one of the promising efficient approaches, as it allows for the elimination of load imbalance in the computational grid due to land points).
- Scaling requirements of memory with flop/s
  - the memory required by the application scales with the increase of flops/s and memory optimization is needed.

Candidate CDV: Molecular Simulations (molecular chem, biochem, biophys, materials sci) Zhong Jin, CAS

- Programming Model
  - Domain decomposition, atom decomposition or force decomposition
  - MPI based, CHARM++ for dynamic load balancing
- Current scaling performance
  - Good app suitable for accelerating by using GPU
- Algorithm dependencies
  - N-body, FFT, FMM
- Software library dependencies
  - FFTW
- Balances
  - Communication vs. Computation: increases along with the increases of simulated system
  - Synchronization vs. Computation: frequent synchronization; careful load balancing essential
- Issues
  - Could try MPI + OpenMP for *ab initio* MD

Candidate CDV: CellMD
Sriram Swaminarayan, LANL

- Programming Model
  - Message passing between nodes, shared memory on node
  - Accelerated on cell processors
- Current scaling performance and percentage of peak
  - weak-scaling to 100000 cores on Roadrunner (~12000 nodes)
  - 36% of peak on Roadrunner
- Algorithm dependencies

- Software library dependencies
  - none
- Balances
  - Limited by memory bandwidth per core
- Scaling requirements of memory with flop/s

Candidate CDV: Synthesis Imaging (Square Kilometer Array Scale Imaging) Tim Cornwell, Ben Humphreys, CSIRO

- Programming Model
  - SPMD by domain decomposition. Long concurrent computation phase between barriers
  - Currently message passing. Possible benefits in shared memory
  - Amenable to accelerators, few "hot spots" however huge amount of ancillary/ supporting code
- Current scaling performance and percentage of peak
  - Strong-scaling to multi-core, sockets or nodes. Workload can be partitioned by spectral channel, w-plane to O(1,000,000) tasks, probably further
  - Efficient scaling on multi-core limited by memory bandwidth
  - Gets 20-40% of peak on CPU and ~5% on GPU (limited by memory bandwidth)
- Algorithm dependencies
  - FFT, BLAS (Primarily level 1 CAXPY, CDOTU_SUB)
- Software library dependencies
  - Astro specific – Coordinate systems, Image formats, etc.
- Balances
  - Cores vs Memory bandwidth: Today, hard to efficiently use more than 3 cores per socket
  - Input I/O vs Compute: Potentially Exabytes input per day for SKA
- Scaling requirements of memory with flop/s
  - Strong scaling: however reductions in per node memory footprint somewhat sub-linear
  - Weak scaling: with nearly constant memory per flop/s

Candidate CDV: Euler Aerodynamics
David Keyes, KAUST & Columbia Univ.

- Programming Model
  - Bulk synchronous, SPMD by domain decomposition
  - Limited hybrid (mainly message-passing, but flux-evaluation subroutines are easy to multicore, based on multicoloring the unstructured tet mesh)
- Current scaling performance and percentage of peak
  - Weak-scales to distributed nodes without limit in principle, but very limited multicore
  - Gets 5%-15% of peak when finely tuned (close to theoretical mem bandwidth limit)
- Algorithm dependencies
  - Newton-Krylov-Schwarz implicit unstructured grid control-volume code
- Software library dependencies
  - PETSc, ParMetis
- Balances
  - Input: needs to read mesh and dump solution once (10^7 vertices or more)
  - Communication vs. Compution: constant ratio in weak scaling
  - Synchronization vs. Computation: frequent synchronization; careful load balancing essential
- Scaling requirements of memory with flop/s
  - Strong scales only over limited ranges (down to about 10^3 vertices per proc)
  - Weak scales with nearly constant memory per flop/s

Candidate CDV: Nek5000 –CFD, Heat Transfer, MHD
P. Fischer, ANL

- Programming Model
  - Bulk synchronous, SPMD by domain decomposition
  - Limited hybrid (mainly message-passing, but flux-evaluation subroutines are easy to multicore, based on multicoloring the unstructured tet mesh)
- Current scaling performance and percentage of peak
  - Strong scaling: ~70 % efficiency on 131,000 cores with 7000 pts/core
    20% of peak on 262,000 cores of Juelich BG/P, 7 billion points
- Algorithm dependencies
  - Spectral element method – fast operator evaluation based on small dgemm kernels
  - Fast Poisson solve: p-multigrid + scalable AMG for coarse-grid solve
- Software library dependencies
  - Scalable MPI, VisIt for visualization
- Balances
  - Input: needs to read mesh and dump solution once (10^7 vertices or more)
  - Communication vs. Compution: constant ratio in weak scaling
  - Synchronization vs. Computation: frequent synchronization; careful load balancing essential
- Scaling requirements of memory with flop/s
  - Strong scales only over limited ranges (down to about 5000 vertices per proc at P > 10^5)
  - Require about 2-3 KB per point.

Candidate CDV: Multi-group Neutron Transport Andrew Siegel, ANL

- Programming Model
  - Full space-angle-energy decomposition with multigrid underdevelopment
  - Petascale machines are insufficient for added energy parallelism
  - No hybrid coding at this point, coefficient matrix vector application on each grid is suitable in a sense. Algebriac multigrid on multicore?
- Current scaling performance and percentage of peak
  - 71% strong scaling for typical execution with ability to improve to +85%
    - 95% spatial scaling on meshes with 10 million vertices at 2048 processors
    - 63% angular scaling for 1 dpp is used; 75% with 2 dpp (200 direction study)
  - 75% weak scaling in angle at full machine level of BGP-JSC and XT5-ORNL.
  - Gets 5%-8% of peak; unoptimized at this point (typical memory bandwidth problem)
- Algorithm dependencies
  - Multiple levels of (F)GMRES and CG with strong convergence dependence on group wise cross section data. Minimal of 4 levels of nested iteration (Power, GMRES, CG, CG)
- Software library dependencies
  - PETSc, MeTiS, Triangle
- Balances
  - I/O: needs to read mesh once and dump solution ($>>10^6$ vertices * $>10^2$ groups)
  - Communication vs. Computation: increases in weak angle scaling, not in space
  - Synchronization vs. Computation: frequent in space, order of magnitude less in angle

…

- Scaling requirements of memory with flops
  - Strong spatial scaling is purely a function of PETSc implementation where current machines require >1000 vertices per processor at lowest level. A multigrid preconditioner should improve per process performance but maintain the current constraints from memory usage.
  - Strong angle scaling has no real memory issues and is purely dependent upon “global reduce” efficiency on the machine (can be a function of the process layout: rack to rack; within rack, etc…).
  - Weak scaling of memory is effectively constant assuming the connection in space is maintained (i.e. energy and angle are fixed)

Candidate CDV: HEDP with FLASH
Carlo Graziani, U Chicago

- Programming Model
  - Bulk synchronous, SPMD by domain decomposition
  - Current implementation message-passing, experimentation underway to introduce hybrid model. Initial results are good.
- Current scaling performance and percentage of peak
  - Weak-scales all the way on Intrepid (production run currently up to > 130K cores, includes parallel I/O)
  - Gets 8%-15% of peak depending upon the specific application and code components being exercised.
- Algorithm dependencies
  - Block structured AMR, iterative solvers (explicit, semi-implicit and looking into implicit)
  - Heterogeneous collection of solvers (different type of solvers for different physics)
- Library dependencies
  - PARAMESH (current) / Chombo (future), maybe PetSc or Hypre in future
- Trade-offs
  - Communication vs. Computation: frequency of re-gridding dictates overall ratio which weak scales, most other communication localized, therefore strong scales.
  - Synchronization : Frequency synchronization to ensure correct behavior, the algorithms themselves are amenable to fuzzier synchronization if the option was available
  - Output data size roughly an order of magnitude less than memory footprint
- Scaling requirements of memory with flop/s
  - Weak scales with nearly constant memory per flop/s
  - Potential for improving the ratio

- Programming Model
  - Framework for exploration of layered approach: encapsulation and isolation
  - Higher fidelity physics dictates more variables

- Algorithm dependencies
  - Will not lend itself to a specialized machine because of different computational characteristics of different pieces of physics, may be representative of many different class of applications

- Trade-offs
  - AMR, critical for resolution and efficient use of computational resources
  - Highlights global communication patterns that are non-trivial to eliminate, and therefore may trigger completely new approach
  - Moving away from strict synchronization, how to ensure deterministic sequence of operations for dependability of results
  - Trade-offs between memory and repetitive computations
  - Exploration of alternative algorithms in the presence of soft errors.

Candidate CDV: Computational Materials Science / Electronic Structure Theory

M. Scheffler, V. Blum, Stefan Heinzel, Fritz-Haber-Inst.

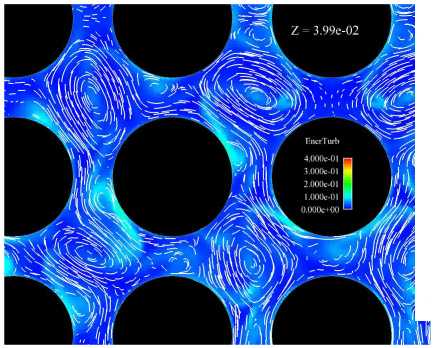
- Programming Model
  - MPI/OpenMP hybrid
  - Shared memory implementation
- Current scaling performance and percentage of peak
  - Weak-scaling to distributed nodes
  - 10%-15% of peak after tuned
- Algorithm dependencies
  - Efficient distribution of operations based on a real-space grid
  - Fast robust eigenvalue solution for generalized, non-sparse eigenvalue problems
- Software library dependencies
  - Blas, ScaLAPACK
- Balances
  - Extreme low latency for high communication requirements with high bandwidth
  - Efficient parallel "distribution" of independent sub-processes with regular but infrequent data synchronization between runs (structure search)
  - Frequent synchronization
  - Parallel post-processing of large amounts of output data (information collected during long molecular dynamics trajectories)
- Scaling requirements
  - Weak and scaling requirements

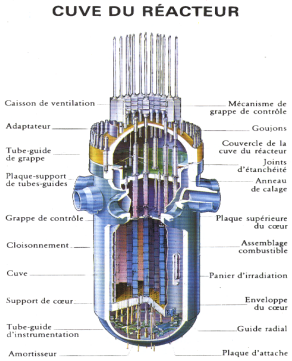
Candidate CDV
CFD Simulation for nuclear application
Jean Yves Berthou

<u>Long term objectives:</u> Mechanical and vibratory behaviour of the fuel assemblies inside a nuclear core vessel

<u>Existing codes:</u> Code_Saturne GPL Open Source , 500 K lines, http://www.code-saturne.org

- Single-phase laminar and **turbulent flows**: k-ε, k-ω SST, v2f, RSM, LES
- **Radiative** heat transfer (DOM, P-1)
- **Combustion** coal, gas, heavy fuel oil (EBU, pdf, LWP)
- **Electric** arc and Joule effect
- **Lagrangian** module for dispersed particle tracking
- **Atmospheric flows** (aka *Mercure_Saturne*)
- Specific **engineering module** for cooling towers
- **ALE** method for deformable meshes
- **Conjugate heat transfer** (SYRTHES & 1D)





<u>2020 target:</u> **3D LES** Full vessel (17x17x196 rods) unsteady approach, >50 billion cells, 1000000 cores during few weeks. Computations with smaller and smaller scales in larger and larger geometries for a better understanding of physical phenomena

⇒ A better optimisation of the production (margin benefits)

Candidate CDV
CFD Simulation for nuclear application (cont)

- Individual performance and scalability of component codes
- The Saturne_code (PRACE benchmark) is running on several tens thousand of cores with a mixed MPI/OpenMP architecture. Need less than 1 Go per core, is communication intensive.
- Software issue at long term:

Mesh generation, visualization

New numerical methods (stochastic, SPH, FV)

Scalability of linear solvers, hybrid solvers

Code optimisation: wall of the collective communications, load balancing

Adaptive methods (may benefit all of computation/visualisation/meshing)

Data redistribution, IO (if flat MPI-IO model OK, good, otherwise require new “standard” data models)

Fault tolerance

Machine independent code optimisation & performance

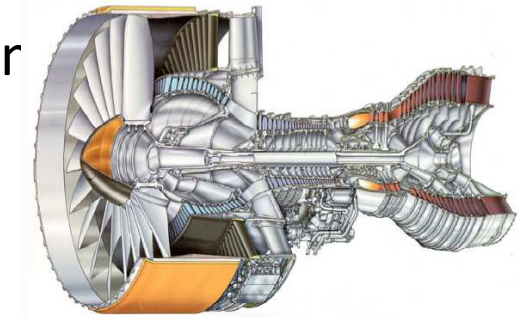
Candidate CDV
Combustion for Aeronautics
Jean-Claude André, CERFACS

Long term objectives: perform a coupled simulation of the whole engine of a gas turbine (multiphysics and multicomponents)

Existing codes: solvers for each component of the turbine already exist and run efficiently on parallel machines.

2020 target: fully coupled simulation

- Handle acoustics, mean flow, instabilities, non nominal regimes, ignition, quenching thanks to a fully coupled simulation with AVBP and elsA using PALM
- Run efficiently on typically 100 000 processors and split the machine between the two codes with maximum parallelization

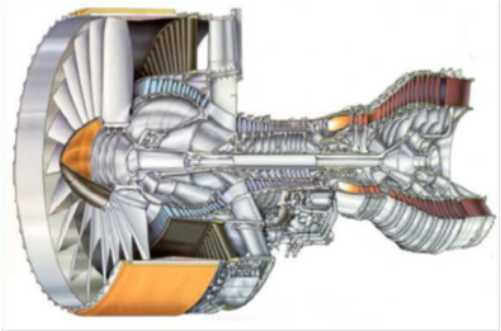


Candidate CDV
Combustion for Aeronautics (cont)

- <u>Individual performance and scalability of component codes</u>
  - The combustor code AVBP is working on non structured grids and scales very well (tests up top 16000 processors in production mode in 2010). Typical grids today are 10 to 330 Mcells run over 100 to 500 000 explicit time iterations
  - The aerodynamics code elsA uses structured grids and scales very well up to 4000 processors. It will be complemented by an unstructured version in 2011. Typical grids: 10 to 800 Mcells advanced with implicit time schemes

- <u>Overall performance of the multi-physics coupled system</u>
  - The coupling of these two codes in an efficient manner is the key question: this coupling will be based on PALM.
  - Optimization and parametric and statistical analysis will require to run these coupled simulations multiple times. Parallel simultaneous multiple coupled simulations is necessary .

- <u>Data management</u>
  - Exponential increase in Input/Output
  - Exponential increase in the amount of data to pre/post-process



Candidate CDV: Combustion (S3D)
Bronson Messer, ORNL

- Programming Model
  - Bulk synchronous, SPMD by domain decomposition
  - Limited hybrid (mainly message-passing, but recently OpenMP added for chemistry on-node)
- Current scaling performance and percentage of peak
  - Weak-scales to distributed nodes without limit in principle
  - Gets 5%-10% of peak when finely tuned
- Algorithm dependencies
  - Explicit compressible finite difference solver with higher-order RK integrator
  - Explicit rate solver for local chemsitry
- Software library dependencies
  - None, save for MPI (really!)
- Balances
  - Input: needs to read small input deck and dump solution at many (O(1000)) timesteps
  - Communication vs. Computation: constant ratio in weak scaling
  - Synchronization vs. Computation: frequent synchronization; careful load balancing essential
- Scaling requirements of memory with flop/s
  - Strong scales only over limited ranges
  - Weak scales with nearly constant memory per flop/s

First Principles Thermodynamics: WL-LSMS
Bronson Messer, ORNL

- Programming Model
  - Two major levels of parallelism:
    - Parallel Monte-Carlo walkers
    - Domain decomposition on the atomic level in real space
  - Reliance on linear algebra libraries for single node performance
- Current scaling performance and percentage of peak
  - Near perfect weak-scaling in the number of nodes
  - 75% of peak with optimized matrix multiply (close to LINPACK performance)
- Algorithm dependencies
  - Wang-Landau (WL) Monte-Carlo at the top level
  - Complex Matrix inversion at the Localy Selfconsistent Multiple Scattering (LSMS) level
- Software library dependencies
  - LAPACK, BLAS
- Balances
  - Input: needs to read starting potential (~50kB / atom and walker)
  - Communication vs. Computation: constant ratio in weak scaling
  - Synchronization vs. Computation: infrequent synchronization and global communication. Load balancing between WL walkers ensured by algorithm, LSMS load balance might be non ideal for some crystal structures
- Scaling requirements of memory with flop/s
  - Constant memory per atom and walker:
    - Strong scaling with single node LINPACK performance
    - Weak scaling with number of nodes

Candidate CDV:
Astrophysical Radiation Hydrodynamics (GenASiS)
Bronson Messer, ORNL

- Programming Model
  - Bulk synchronous, SPMD by domain decomposition
  - Hybrid, "home-brewed," physics-based preconditioner for transport
- Current scaling performance and percentage of peak
  - Weak-scales to distributed nodes to more than 90K MPI ranks
  - Gets 5%-15% of peak (5% for MHD only, 15% for additional microphysics)
- Algorithm dependencies
  - Newton-Krylov implicit solve for radiation, finite volume magnetohydro, structured AMR grid (currently cell-by-cell refinement)
- Software library dependencies
  - PETSc, SILO
- Balances
  - Input: needs to read IC and dumps solution at regular cadence (O(1000) times during a run of O(100,000) timesteps)
  - Communication vs. Computation: constant ratio in weak scaling
  - Synchronization vs. Computation: frequent synchronization; careful load balancing essential, must be dynamic for AMR
- Scaling requirements of memory with flop/s
  - Strong scales only over limited ranges (Amr cells represent smallest quantum)
  - Weak scales with nearly constant memory per flop/s (earlier versions of code had problems with this)

Candidate CDV: Classical Molecular Dynamics
Fred Streitz, LLNL

- Programming Model
  - MPI (also hybrid MPI/threads, depending on physics included)
  - SPMD with variety of domain decomposition schemes
- Current scaling performance and percentage of peak
  - weak scaling to 300,000 cores – no practical limit (30+% of peak)
  - strong scaling limited by communication bandwidth
- Algorithm dependencies
  - N-body
  - Possibly FFT
- Software library dependencies
  - none necessary
- Balances
  - Relatively light-weight state: restart (with optimized I/O) < 3% of runtime
  - Timestep is natural sync barrier
  - Communication or computation can dominate, depending on form of interaction potential
- Scaling requirements of memory with flop/s
  - Strong: decidedly sub-linear in limit
  - Weak: constant memory with flop/s

Candidate CDV: general purpose numerical framework (nuclear and solid state physics, chemistry, nanophotonics, fluid dynamics)

Programming Model
Global namespace/container distributed object multithreaded model (similar to Cilk and Charm++) with futures for hidy latency and managing dependencies and on top of an active message layer.
One MPI process per node; multithreading on a node via a parallel task queue (c.f., Intel TBB)
Standard C++ and pthreads

Current scaling performance and percentage of peak

Main kernels are long-thin matrix operations (e.g., (20*400)^T * (20,20)) for which we have hand optimized assembly.
Variable depending on application. Sequential performance maxes out at 40% of peak (~50% of time in the kernel that is running at ~75% of peak).
Multicore scaling good up to ~16 cores & need to switch to Intel TBB to improve upon this.
Parallel performance limited by both load balance and communication overhead (internode task stealing still only in prototype)
Excessive global synchronization

Algorithm dependencies
None

Software library dependencies
Pthreads, sequential BLAS, sequential linear algegbra, MPI that at least supports thread-serialized

Balances
One-sided access to global data permits dynamic load balancing
Object serialization permits dynamic data redistribution
Parallel computer flow control can be overwhelmed by volume of small messages (need aggregation by library or runtime).

Candidate CDV: NWChem general molecular electronic structure code

2009 Gordon Bell finalist, R&D 100 award

Programming Model
NUMA/PGAS using Global Arrays and MPI; SPMD with variety of task and data compositions
Limited multithreading to date (mostly via parallel BLAS) ... work in progress.

Current scaling performance and percentage of peak

Many-body elements strong scale to size of jaguar with circa 60% of peak speed for typical problem sizes.
Plane wave ab initio MD strong scales to 10+K processors at present (<1s per time step) ... goal is to extend this to 100+K processors (aiming for ~1ms per time step).Communication (?) is the bottleneck at the scaling limit.
Gaussian density functional theory weak scales well. Load balance and parallel linear algebra are scaling limits.

Algorithm dependencies
Parallel linear algebra, numerical quadrature, non-linear solve

Software library dependencies
Global Arrays, BLAS, parallel linear algebra (beyond scalapack)

Balances
One-sided access to global data permits dynamic load balancing
Many-body methods – ratio of comms to compute inversely proportional to square of problem size
Density functional methods – asymptotically ratio of comms to compute is a constant
Infrequent synchronization and collective global operations
One common performance problem is performance of remote accumulate operation