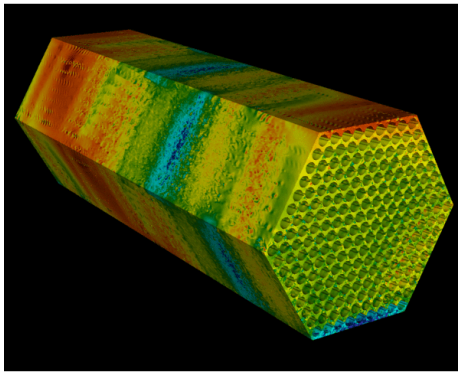
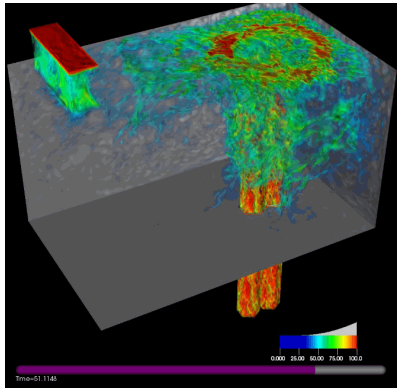
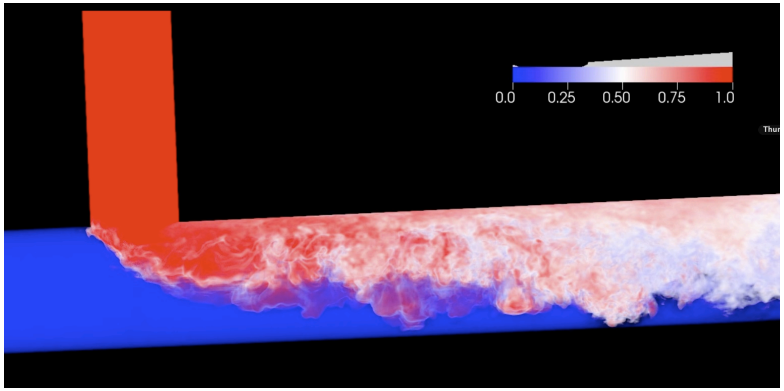




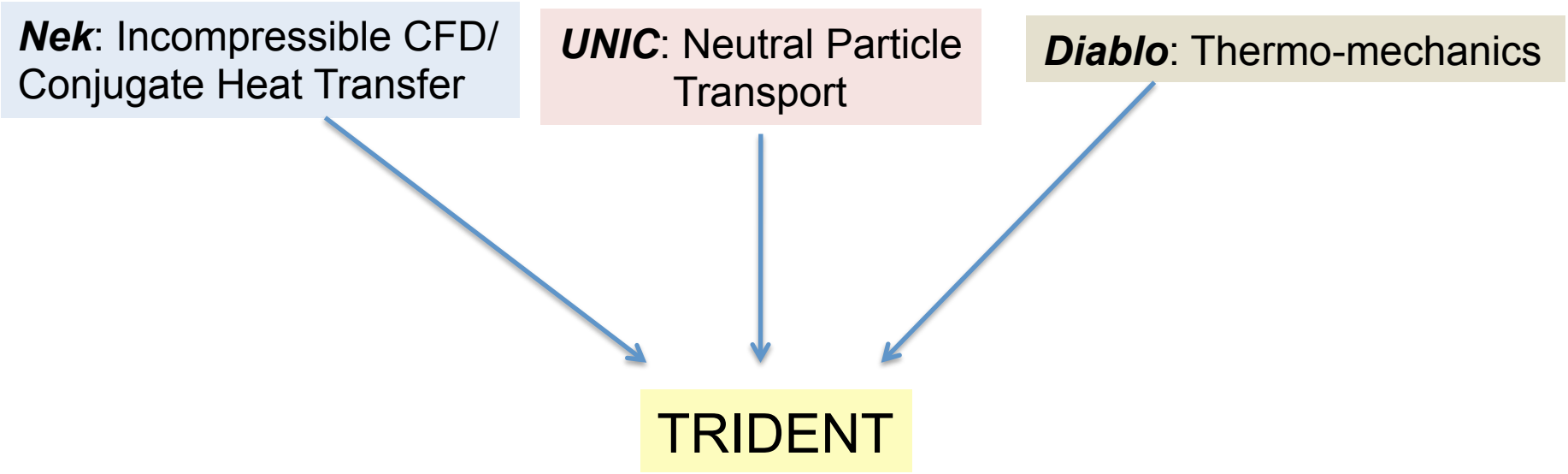


CESAR: Center for Exascale Simulation of Advanced Reactors

Initial perspective on priorities for supporting hardware, system software, and libraries

Andrew Siegel, ANL

CESAR - Reactor Core Physics



- All three building blocks: Nek, UNIC, Diablo, have had some success at petascale
- Success leans heavily on certain properties
  - -All three Fortran/C+MPI, no threading (good enough to run cores as MPI procs)
  - -For performance all three depend on
    - -sufficient memory per MPI process
    - -highly efficient MPI collectives
    - -for single proc performance, sufficient memory bandwidth
    - -optimized matrix vector products
    - -good tools to analyze performance (both use of mem hierarchy and MPI ops)

Hierarchy of issues

Push for what is needed to succeed at exascale along familiar path

↓

Employ same numerical approach with new algorithmic implementations (programming models, use of memory hierarchy, etc.)

↓

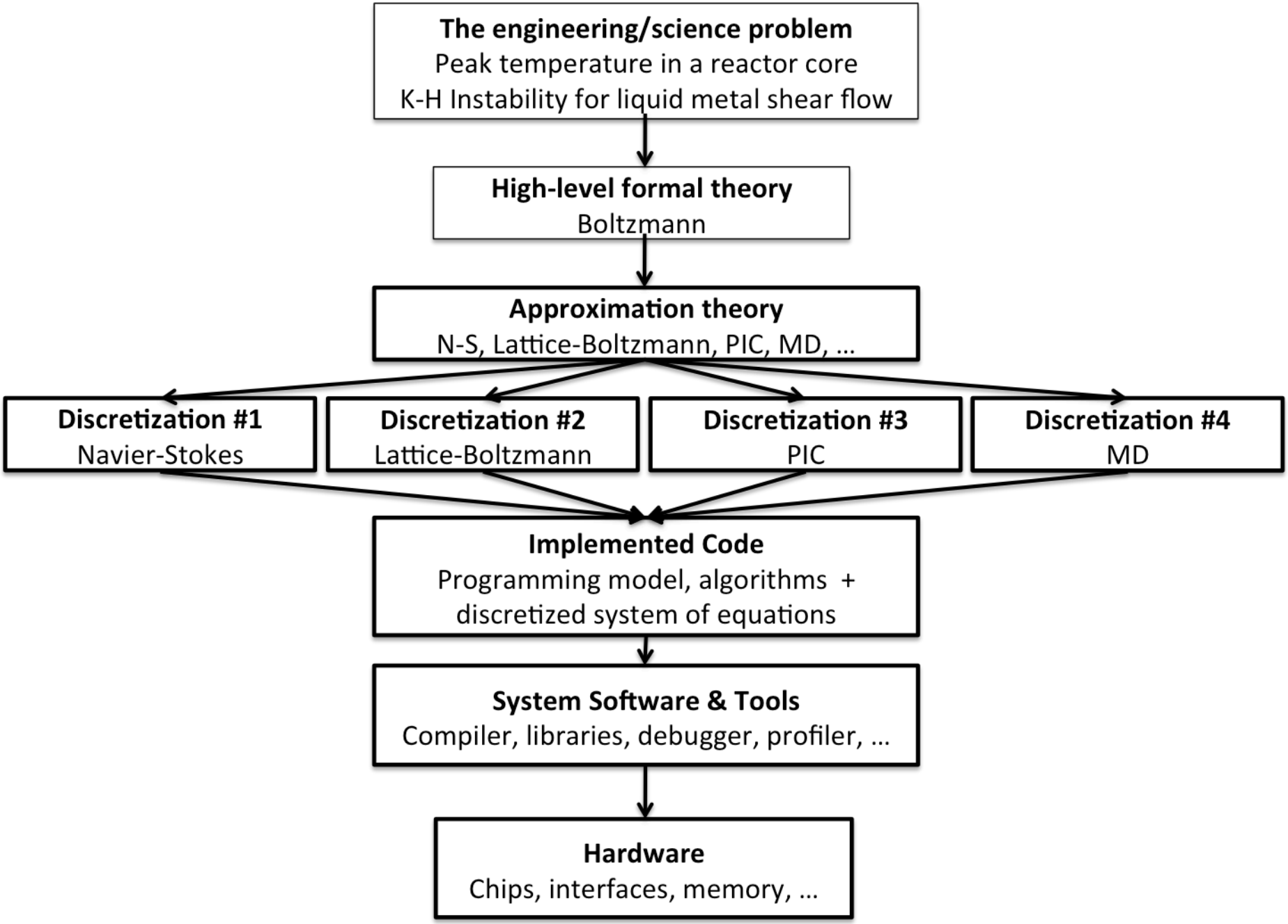
Adopt new discretizations and/or solvers for same basic mathematical formulation to better match new systems

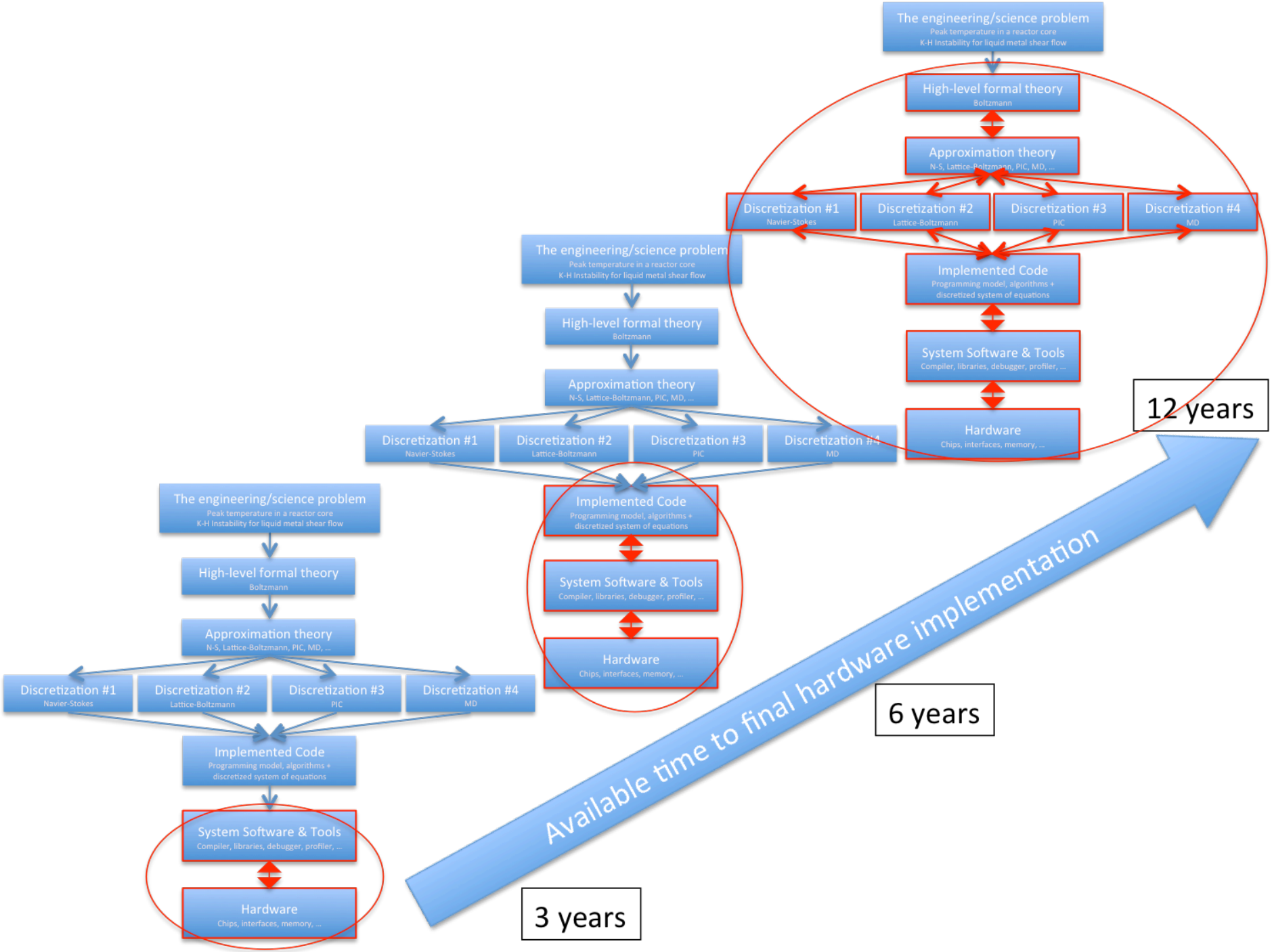
↓

New mathematical formulations: particle (Monte Carlo neutronics) or pseudo-particle (e.g. Lattice Boltzmann) methods

App inventory focus is here

Need to remain open to all four levels of code evolution as part of co-design


The engineering/science problem
Peak temperature in a reactor core
K-H Instability for liquid metal shear flow
High-level formal theory
Boltzmann
Approximation theory
N-S, Lattice-Boltzmann, PIC, MD, …
Discretization #1
Navier-Stokes
Discretization #2
Lattice-Boltzmann
Discretization #3
PIC
Discretization #4
MD
Implemented Code
Programming model, algorithms +
discretized system of equations
System Software & Tools
Compiler, libraries, debugger, profiler, …
Hardware
Chips, interfaces, memory, …


The engineering/science problem
Peak temperature in a reactor core
K-H Instability for liquid metal shear flow
High-level formal theory
Boltzmann
Approximation theory
N-S, Lattice-Boltzmann, PIC, MD, ...
Discretization #1
Navier-Stokes
Discretization #2
Lattice-Boltzmann
Discretization #3
PIC
Discretization #4
MD
Implemented Code
Programming model, algorithms + discretized system of equations
System Software & Tools
Compiler, libraries, debugger, profiler, ...
Hardware
Chips, interfaces, memory, ...
12 years
6 years
3 years
Available time to final hardware implementation

Discretizations/Codes

Code	Method	Scalability	Bottlenecks	Plans
Nek500	Large Eddy Simulation	• 294K BG/P • 80% parallel efficiency (weak)	• Good scaling assuming adequate memory/node • Log P effect due to algebraic MG	• clearly define memory per node requirements • P = 10^9 realizable assuming n=10^{12}
UNIC	Second order SN	• 294K BG/P • 222K XT5 • 75% parallel efficiency (weak)	• Load imbalance across angles • Adequate memory/ core	• Predictions of work • Memory saving algorithms • Decomposition by energy groups
UNIC	3-D Method of Characteristics	•Hundreds of procs	•Brute force domain decomposition very inefficient	• Creative approaches to DD, especially for hybrid architectures
UNIC	Monte Carlo	•Hundreds of procs	•Domain decomposition for huge tallies of spatial distributions	•Explore feasibility of computing random particle tracks efficeintly on accelerators; creative approaches to dynamic load blanacing

Highest priority issues

- Sufficient memory per MPI process for O[10^3-10^4] unknowns per MPI process desired but not likely. Need to rethink ways to exploit node parallelism.

- Optimized fast matrix-vector products for dense, short vectors (DGEMM)

- Robust tools for on-node performance analysis, particularly new concepts in appropriate tools for future architectures

Highest Priority Issues, cont.

- Ability to efficiently handle
  - random execution paths – hard to vectorize?
  - very large random access material data files
  - branch-heavy codes minimal branch prediction

- Reproducibility: critical for acceptance in nuclear energy community, particularly important for Monte Carlo neutronics

- Equivalent of very fast, high-volume nearest neighbor data exchanges

Highest Priority Issues, cont.

- Backward-compatible programming model, incremental migration (MPI+X) or at least way to re-use code infrastructure
- Good peta-scaling depends on relatively high byte to flop ratio. Minimum number of points per MPI process to achieve scalability
- Fast global MPI operations (allreduce, gather)
- Sparse eigen-problem – power iteration, preconditioned CG via PETSc
- Sparse matrix vector memory bandwidth limited

Highest Priority Issues

- Support for evolution of exascale infrastructure (MOAB, PETSc,…)
- Performance analysis tools for new programming models/hardware critical
- Hundreds of petabytes of data produced
  - Ability to produce, move and visualize critical for insight, particularly heat transfer/turbulent mixing

Additional Information from Application Survey (adapted from Tom Peterka, input from P. Fischer, R. Ferencz, and M. Smith)

Memory

	Trident	Nek5000	Diablo	UNIC
Memory layout of data structures	MOAB data on a mesh; future may need other data structures for fast transients not computed by component codes	Matrix-vector products in contiguous memory, cache-friendly	Unstructured FEM, homogeneous and highly vectorized (128-way)	Matrices, boundary conditions, material data, in cache-friendly blocks; MOC trajectory data
Memory requirements	Little in addition to component codes, but sum of components is substantial if components run on same node	24 PB total, divided by number of nodes	Up to 32 GB per node	Max. available, 2 GB per node currently
B/W limited	NA	Yes	Yes	Yes
Usage patterns	NA	Little replication, much WORM	Vectorized, little replication or WORM	Some replication, some WORM

NA = not answered; WORM = write once, read many

I/O

	Trident	Nek5000	Diablo	UNIC
In-memory data model	Regular unstructured	Globally regular unstructured, locally regular structured	Regular unstructured	Regular unstructured
Output data format	MPI-IO, HDF5	Custom, MPI-IO	HDF5	HDF5
Number of output files	Configurable, preferred 1 file total	Configurable, preferred 1 file total	1 file per process	1 file total
Output size	100s PB future	240 PB future	20 PB future	100s PB future
Input data	80-100 B per grid point (TBs)	40-80 B per grid point (TBs)	Small, few MB	GBs of material cross-section data
Fault tolerance	Will continue to checkpoint	Will continue to checkpoint	Will continue to checkpoint	Checkpointing in the future
Checkpoint restart memory	3%	3%	10-100%	100%
Out of core	Perhaps future	No plans	No, perhaps future	No plans
I/O benchmarks	Simple ones	No	No	No

Data Analysis & Visualization

	Trident	Nek5000	Diablo	UNIC
Current approach	VisIt, parallel postprocessing	Vist, parallel postprocessing Volume rendering, isosurfaces, movies	VisIt, serial postprocessing Isosurfaces, cutting planes	VisIt, serial postprocessing Power distribution along cross sections
Future approach	In situ	In situ	Same use model mapped to larger datasets	Unsure
Memory for in situ analysis	50%	50%	Few %	Very little
Analysis data capture frequency	Every 100 time steps	Every 10 time steps, more frequent in situ	Every 10 time steps	Every 10 time steps

Performance Analysis

	Trident	Nek5000	Diablo	UNIC
Tools used	HPM, TAU, self instr. OpenSpeedshop in future	PAPI, self instrumentation	TAU, VAMPIR	PAPI, self instrumentation
Tools shortcomings	Seeing cache behavior and ability to control it	Easy FLOP count	Culling output to extract useful information	Need tools to improve file I/O
Analytical model	No	Yes, accurate to 25%	No	No
Vectorization	No floating point ops in coupling so far	Yes, dense matrix-matrix products	Compiler-generated	Compiler generated
Scale to # cores	1000	262,000	10,000s	262,000
Performance barriers	Scalability of data movement	Cache, memory b/w	Nonlinear equation solving	Too early to say;
Scaling barriers	Joint decomposition of different physics	Internode latency	Nonlinear equation solving	Too early to say
Autotuning	No	Yes in past; plan to use ADLB in future	No, perhaps in future	No, perhaps in future

Software / Hardware

	Trident	Nek5000	Diablo	UNIC
Current platforms	Any MPI, C compatible	Any MPI-compatible	AIX, linux-based CHAOS (fat memory)	Any MPI-compatible except Sun or Win
Libraries	MPI, MOAB	MPI	MPI, Hypre, PETSc, looking at MOAB	MPI, PETSc, MeTiS, HDF5, MOAB in future
Advanced Arch. Features	Node-local storage	Hardware collectives	Hardware collectives, SIMD	Hardware collectives, transactional memory