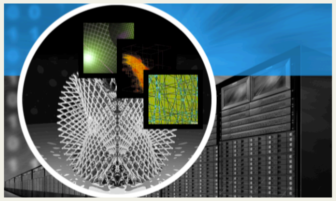


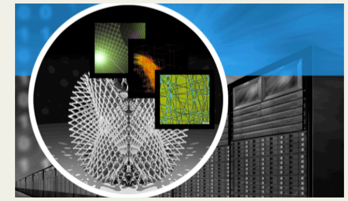
David L. Brown

Deputy Associate Director for Science & Technology
Computation Directorate, LLNL

Presentation to the Advanced Scientific Computing Advisory Committee
March 31, 2010

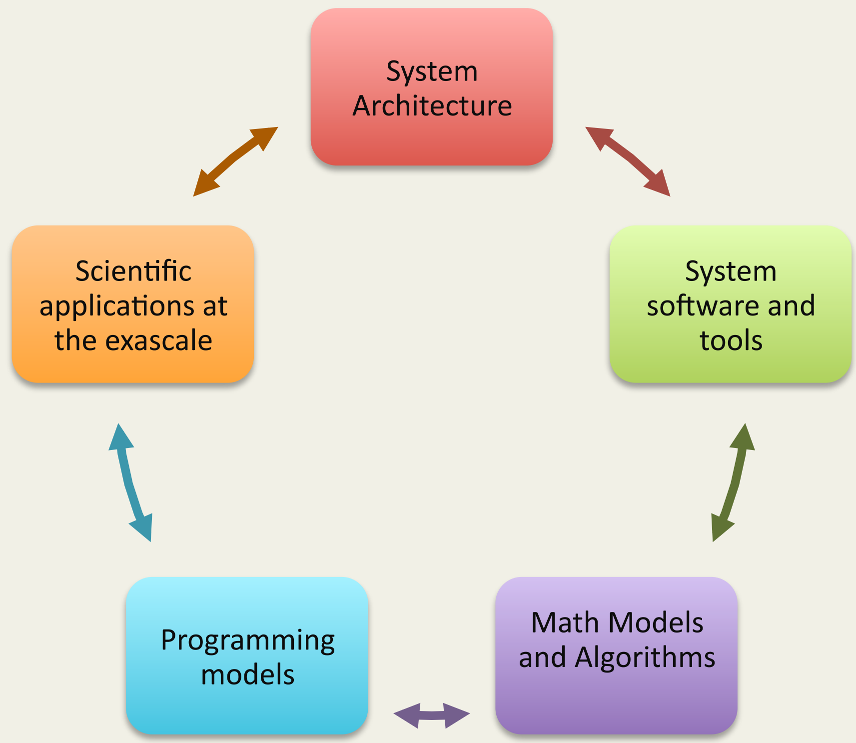
LLNL-PRES-426465

What is required to deliver exascale computational science by 2018?



Workshop Objectives:

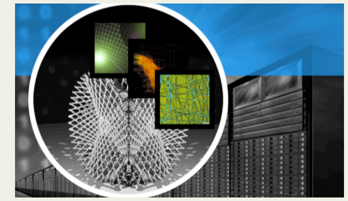
- **Outline the R&D needed for co-design of the exascale computational science environment**
- **Identify opportunities for "disruptive" computational approaches for future scientific discovery**
- **Produce a first cut at characteristics of hardware/software system roadmap that will meet science application needs over the next decade**
  - Initial systems 2015 @ 100-300TF
  - Exascale systems in 2018



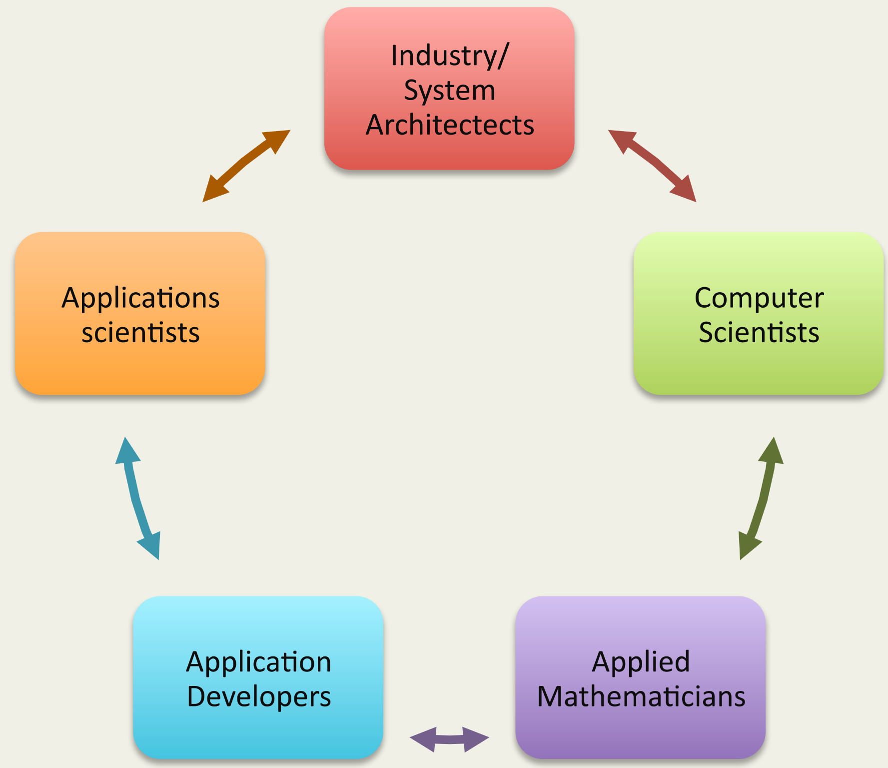
The cross-cutting workshop brought applied mathematicians and application developers into the discussion

Co-design

Co-design will be key for exascale scientific discovery by 2018

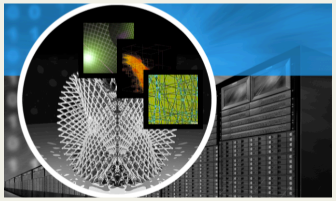


- ***Tightly-coupled multi-disciplinary partnerships will ensure delivery of science applications on exascale platforms***
- ***Transition to exascale will be as disruptive as transition from vector computing***
  - *New programming paradigms required*
  - *Emphasis on physics fidelity and UQ*
- ***Appropriate investments will be required***
  - *ASC spent only 20% of $$ on hardware*
  - *Significant investment in computer science and math research*
  - *Significant investment in re-design and re-write of applications codes*



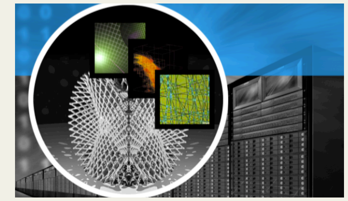
Co-design partnerships for exascale computing

The cross-cut workshop was a co-design "practice session"



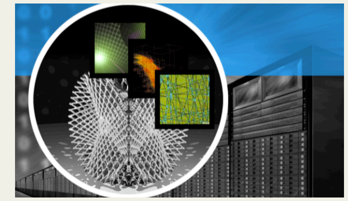
- **Brought disparate computational science research communities together to understand exascale challenges**
  - Applied mathematicians
  - Computer Scientists
  - Computer architects
  - Science application developers
  - Industry representatives
- **Breakout sessions successfully overcame communication barriers**
- **Participants left the meeting with a deeper understanding of each other's communities**
- **"Co-design could really work!"**

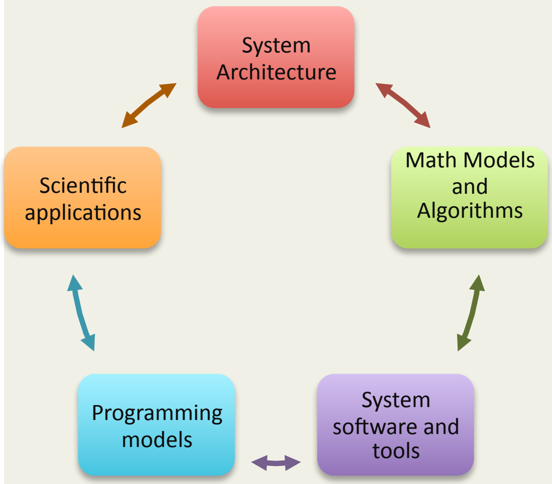
Who and where are the co-designers?
What organizational changes are needed?



- **Designers of extreme scale hardware must obtain a detailed understanding of the scientific challenges**
- **A multi-disciplinary computational science culture has blossomed over the past 15 years**
  - Advanced Simulation and Computing (**ASC**) program used vertically integrated code teams to successfully deliver 3D simulation capability for the Stockpile Stewardship program
  - **SciDAC** program taught computational scientists to "collaborate or die"
  - **CSGF** prepares young computational scientists for HPC scientific discovery
  - Many applications developers will not be skeptical this time about the need for drastic re-writes of critical simulation codes
- **Organizational changes will be essential to meet the 2018 target**
  - Co-location of co-designers ideal, but unlikely to be feasible
  - Vendor IP issues must be addressed
  - Early studies of how best to carry out co-design essential

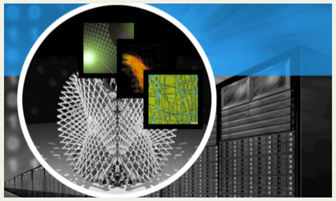
Breakout sessions addressed the three workshop themes





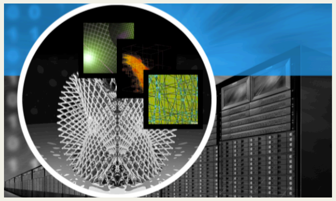
- **Theme 1: Math models and algorithms**
  - Impact of application needs and architectural developments on math models, algorithms and programming models
  - Impact of application, math model and algorithms needs on architectural development
- **Theme 2: System Software**
  - System software functionality required at exascale
  - What tasks traditionally handled by systems software will need to be addressed elsewhere, e.g. resiliency handling?
- **Theme 3: Programming models and environment**
  - What programming models and environments are needed?
  - Will programming models provide suitable abstractions and tools for applications/algorithm needs?

Six "math" areas were used to provide context for the discussions



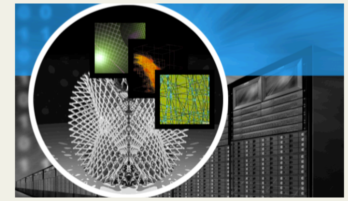
	Theme I	Theme II	Theme III
A. PDEs I	Tuesday pm		
C. PDEs II	Tuesday pm		
D. UQ/ Stochastic		Tuesday pm	
E. Discrete Math		Tuesday pm	
B. Data/ Visualization			Tuesday pm
F. Solvers/ Optimization			Tuesday pm

Six "math" areas were used to provide context for the discussions



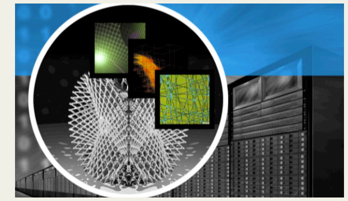
	Theme I	Theme II	Theme III
A. PDEs I		Wednesday am	
C. PDEs II		Wednesday am	
D. UQ/ Stochastic			Wednesday am
E. Discrete Math			Wednesday am
B. Data/ Visualization	Wednesday am		
F. Solvers/ Optimization	Wednesday am		

Six "math" areas were used to provide context for the discussions



	Theme I	Theme II	Theme III
A. PDEs I			Wednesday pm
C. PDEs II			Wednesday pm
D. UQ/ Stochastic	Wednesday pm		
E. Discrete Math	Wednesday pm		
B. Data/ Visualization		Wednesday pm	
F. Solvers/ Optimization		Wednesday pm	

Final sessions synthesized results for each theme

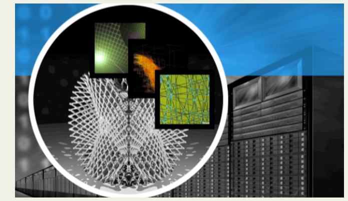


Thursday afternoon

	Theme I	Theme II	Theme III
A. PDEs I			
C. PDEs II	David Keyes		
D. UQ/ Stochastic		Pete Beckman	
E. Discrete Math			Jeff Vetter
B. Data/ Visualization			
F. Solvers/ Optimization			

FINDINGS

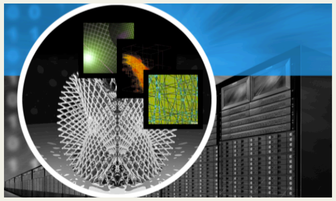
Exascale ≠ Petascale X 1000



- **Traditionally, PDE-based applications have expected 10x increase in resolution with each 1000x increase in compute capability, *but not this time:***
  - We won't have 1000x the memory available
  - The processors won't be 10x faster
  - Proportionally, we won't be able to move as much data on or off each processor
  - Introduction of massive parallelism at the node level is a significant new challenge (MPI is only part of the solution)

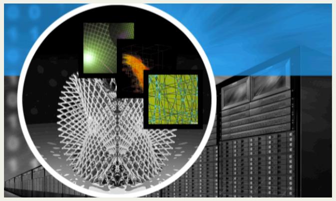
- **However, exascale computing is an opportunity for…**
  - ***More Fidelity:*** Incorporate more physics instead of increased resolution
  - ***Greater Understanding:*** Develop uncertainty quantification (UQ) to establish confidence levels in computed results and deliver predictive science

Uncertainty Quantification will permeate the exascale



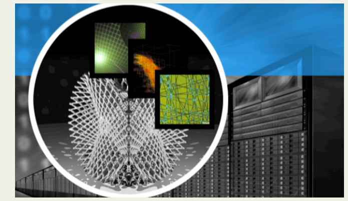
- *UQ is the end-to-end study of accuracy and reliability of scientific inferences*
- **Large ensemble calculations will have dynamic resource allocation requirements significantly different than traditional applications**
- **Traditional space-shared, batch-scheduled usage unlikely to be effective for UQ or new multi-physics codes**
- **Client/server model for UQ requires a different failure model (OK for clients to fail)**
- **Significant code redesign a likely requirement in general –** ***opportunity to embed UQ in exascale applications***

Understanding characteristics of PDE solvers important for co-design



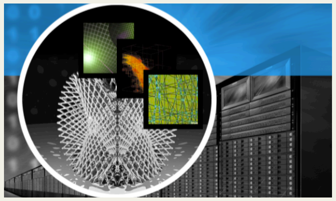
- **Domain decomposition with nearest-neighbor communication patterns**
- **Elliptic solvers: smaller, non-local communication patterns**
  - Frequent nearest-neighbor communications, less frequent messages to farthest neighbors
  - Frequent, Small global reductions (1000's/timestep)
- **Network performance needed:**
  - Low latency
  - High bandwidth
  - High message rate for point-to-point and collective communication operations
  - Highly desirable that physical topology of machine matches communication patterns (optimize performance by reducing network contention)

Memory management will be key in PDE applications



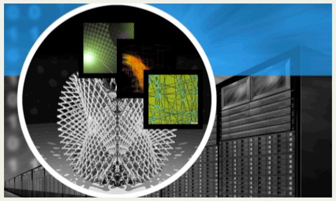
- **Memory hierarchy becomes deeper and more complex at exascale**
- **Inadequate tools and interfaces to hint, manage and control memory for run-time systems**
- **Fine-grain, node-level parallelism in PDE solvers could exploit a hierarchical two-level machine/programming model**
- **Want system software that can exploit spatial/temporal locality hints from application code**
- **Cache is energy-expensive – alternative fast local memory access approaches for performance and energy savings**
- **Low-cost thread create/destroy essential for performance**
  - Global namespace for threads as first-class objects
  - Research needed to find best threading model
  - Threads > processing elements might help hide memory latency
  - Adaptive nature of modern solvers requires a dynamic threading model/ system software support

More dynamic control of system resources will be required at exascale



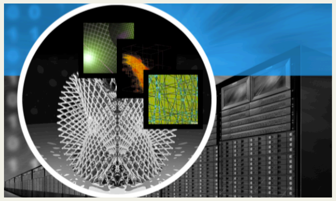
- **Adaptive run-time systems could address**
  - Dynamic load balancing
  - Dynamic power allocation, e.g. to network vs. cpu
  - Ability to reconfigure around faults
  - Dynamic resource requirements
- **Programming model support for dynamic control**
  - Dynamic load-balancing abstractions
  - Language support for dynamic control of resources
  - Methods to record control flows of execution and data to allow reverse computation in adjoint mehtods (UQ)

Applications must care more about fault tolerance and resilience



- Checkpoint-restart won't scale with current storage systems; use NVRAM instead?
- Co-design will be required to develop standard fault management API
- Application-specific fault recovery likely
- Consider local recovery from faults

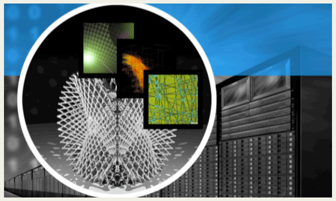
Discrete math applications will challenge exascale machines



- **Large amount of irregular data movement, few FLOPS**
- **Adaptive runtime will be important:**
  - Resource profile is typically dynamic
  - Dynamic load balancing on a node required
- **Energy efficiencies could be achieved with dynamic power allocation between FLOPS or data movement**
- **Need ability to efficiently handle irregular data**
- **Co-design opportunities:**
  - Discrete event simulators for exascale architectures
  - Use graph algorithms for task scheduling on nodes and across nodes

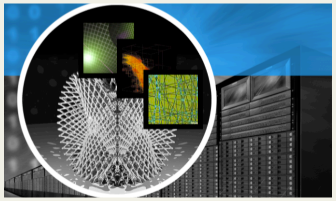
Graph theory
Integer programming
Combinatorial optimization

Familiar system support issues will be even more challenging at the exascale



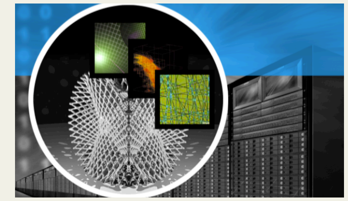
- **File system scalability and robustness will continue to be the weakest link at the exascale**
  - Codes must do more in-situ analysis needed to compensate for I/O limitations
- **Hierarchical debugging tools will be needed**
  - Sophisticated single-node debugger
  - Debugger for 10,000 nodes
  - Large-scale debugger a research challenge
- **Performance tools are not keeping up with largest machines**
  - Results in an understanding gap as we approach exascale
  - Existing performance tools don't address heterogeneous architectures
  - Research needed to develop vertically integrated performance analysis tool for exascale applications

Opportunity for compilers to support heterogeneous and multi-core processors



- **Leverage recent advances in compiler technology (e.g. ROSE)**
- **Compilers hide complexity of underlying instruction sets, some parallelism**
- **Optimizations:**
  - language keywords,
  - annotations
  - runtime adaptation
  - profile-guided optimization
- **New opportunities:**
  - power management
  - small memory capacities
  - resiliency
  - interoperability

Preliminary panel findings are grouped into three categories

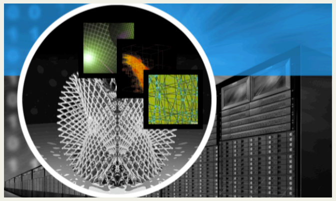


1. Algorithms R&D needed to support new architectures

2. R&D for Programming Models to support exascale computing

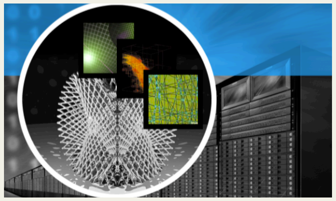
3. R&D for System Software at the exascale

Algorithms R&D



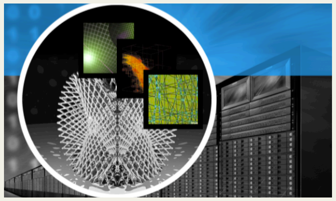
1. Re-cast critical applied math algorithms to reflect impact of anticipated macro architecture evolution
2. Adapt data analysis algorithms for exascale
3. Address numerical analysis questions associated with move from bulk-synchronous to multi-task approaches
4. Develop "mini-applications": essential elements of critical applications
5. Develop simulations of emerging architectures

Re-cast critical applied math algorithms



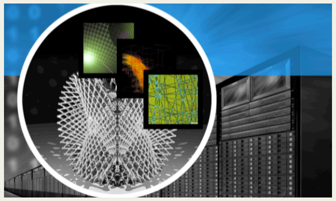
- **PDEs:**
  - New PDE discretizations reflecting shift from FLOP- to memory-constrained hardware
  - New algorithms with more compute, less communication
- **UQ:**
  - Opportunity to re-design codes with UQ built in
  - Move statistics inside loops
- **Solvers and optimization methods:**
  - Solvers with reduced global communication
  - Leverage low-latency on-chip all-gather
  - New sparse eigensolver formulations
  - FFTs
- **Novel algorithms:**
  - Reduced-precision arithmetic algorithms that store less, but maintain accuracy

Adapt data analysis algorithms to extreme scale environments



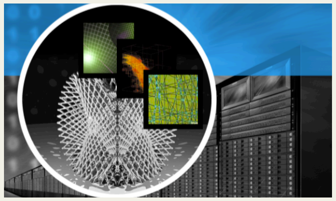
- **Leverage increased node-local NVRAM availability**
  - "Back to the future:" out-of-core approaches
- **Analysis algorithms for streaming data**
- **Leverage global address space**
- **Where is the best place to do analysis?**
  - In situ (part of simulation code)
  - Post processing on the exascale platform
  - Post processing on a dedicated analysis platform (but what about the I/O bottleneck?)
- **Research on development of common data structures or data access patterns to enable re-usable data analysis software**

Address numerical analysis issues associated with move away from bulk-synchronous programming model



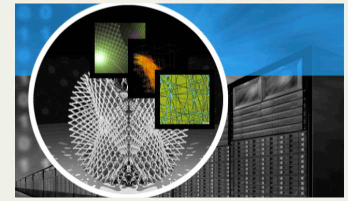
- **Accuracy, stability of multi-physics and multi-scale coupling**
- **High-order operator splitting methods**
- **Accuracy, stability of methods that apply operators more asynchronously**

Role of simulation as part of co-design



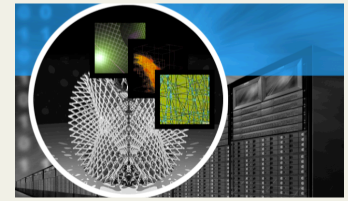
- **Develop "mini-applications" that capture essential elements of large scientific applications**
  - Hardware and system software engineers can understand critical performance issues
- **Develop simulation tools for emerging architectures**
  - Algorithm, application developers can understand code performance on a range of potential architectures

New programming models



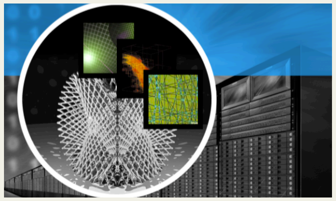
1. R&D new exascale programming paradigms (e.g. MPI+X)
2. Develop API's for dynamic resource management
3. Programming models that support memory management at the exascale
4. Scalable approaches for I/O
5. Interoperability tools to support transition to new environment
6. Language support for PE's at the exascale
7. PM support for latency management
8. PM support for fault tolerance/resilience
9. API's for power management

Investigate and develop new exascale programming paradigms



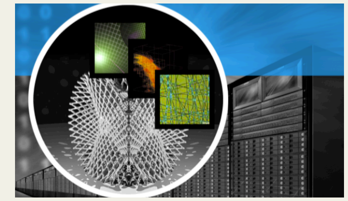
- **Hybrid programming models: MPI+X, with X=**
  - OpenMP
  - Pthreads
  - CUDA (GPUs)
  - Chapel, UPC, co-array Fortran
  - MPI
- **Effective abstractions that expose loop-level and data-level parallelism**
- **Improved abstract machine model**
- **Programming model support for multiple networks on same machine**
- ***New PM an opportunity to change how computational science is done:***
  - *Introduction of intrusive (but more efficient) UQ techniques*
  - *MPMD approach to multi-physics application codes*

Memory management, I/O, interoperability, language support



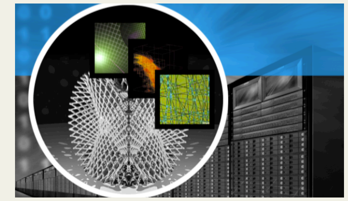
- **Memory management**
  - PGAS language support
  - Memory consistency models to support discrete algorithms
- **Interoperability**
  - Migration from old PMs/ languages will be gradual
  - Need support for interoperability between "old" and "new"
- **I/O**
  - Consider database approaches (object models) for I/O
  - PM support for data structure linearization
- **Language support**
  - Asynchronous algorithms
  - Uncertainty-carrying variables

Latency, Resilience, Power, New approaches



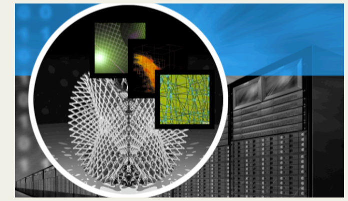
- **Latency management**
  - Need capability to overlap computing, analysis, communication, I/O
- **Resilience**
  - PM support for fault management
  - Fault-tolerant MPI collectives
  - API for checkpointing
- **Power**
  - Power-aware programming models
- **New approaches**
  - Message-driven PM's for scientific applications?
  - API to support execution through a DAG
  - PM's for in-situ data analysis

System Software



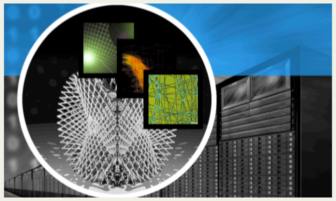
1. System software tools to support node-level parallelism
2. System support for dynamic resource allocation
3. System software support for memory access
4. Performance/Resource measurement and analysis tools
5. System tools to support fault management
6. System support for exascale I/O

Support for node-level parallelism



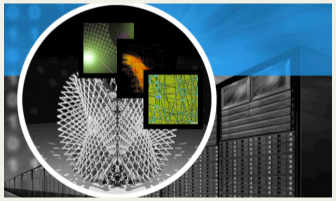
- Small light-weight messages
- Light-weight fine-grained and flexible synchronization
- Latency tolerance through high degree of threading
- System calls for node-level parallelism
- Low cost thread create/destroy
- Software control of on-chip data movement to enhance performance
- Fast all-reduce (for fast inner products)
- Tools to manage communication patterns
- Tools to support move away from bulk-synchronous parallelism
- Tools to support maintenance of local state when objects migrate between processors

System software support for memory access



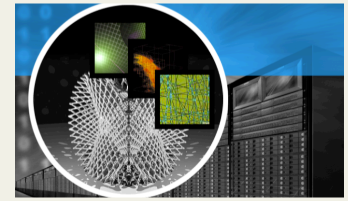
- Support for GAS to replace cache-coherence as as mechanism
- Research the use of GAS in partitioning of graphs
- Tools to manage memory hierarchies
- Ability to turn off memory hierarchy for accesses that cannot make good use of it
- Hooks for direct access to memory management
- Support to allow local memory to be configured in either scratchpad or cache mode
- System support for data provenance to support data analysis

Performance / Resource measurement and analysis tools for exascale



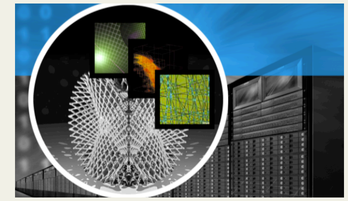
- **Apply data mining methods to help develop performance measurement tools**
- **Performance tools for heterogeneous environments**
- **New performance analysis tools, particularly for hybrid programs**
- **System calls to query relative costs of various operations**
  - Both static and dynamic information is needed
- **Runtime layer functionality to provide information about system state**

System tools for fault management

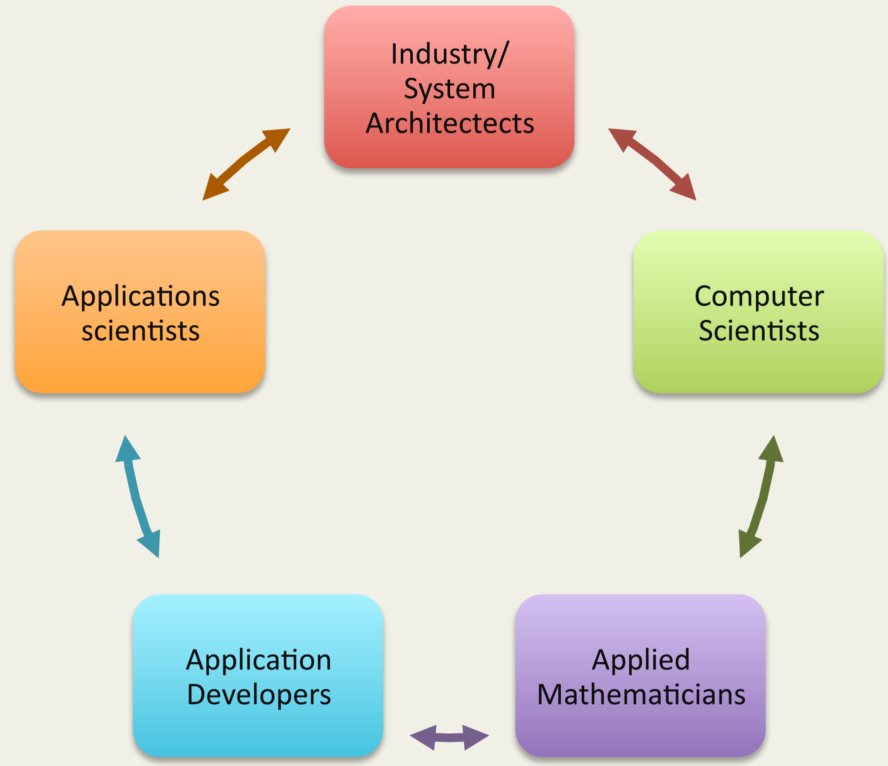


- **Tools to support fault tolerance management**
  - E.g. Fault notification API
- **Research in debugging at scale**
- **Research the fault-tolerance implications of UQ**
- **Develop a taxonomy of faults to support advanced fault handling**
- **Understand the role of system software in resilience**
- **If smaller system (e.g. 10%) is used for data analysis, observe that:**
  - Resilience will be less of a problem since it won't be possible to move large amounts of data off of the main compute platform

Co-design is essential for exascale scientific discovery by 2018



- ***Close multi-disciplinary partnerships will ensure delivery of science applications on exascale platforms***
- ***All partners must commit to significant changes in both hardware and software design***
  - *New programming paradigms required*
  - *Emphasis on physics fidelity and UQ*
- ***Appropriate investments will be required***
  - *ASC spent only 20% of $$ on hardware*
  - *Significant investment in computer science and math research*
  - *Significant investment in re-design and re-write of applications codes*



Co-design partnerships for exascale computing