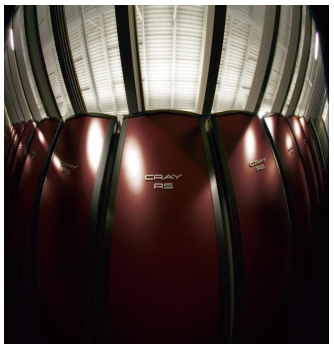
Exceptional service in the national interest

Sandia National Laboratories






Addressing the System Software Challenges for Converged Simulation and Analysis on Extreme-Scale Systems

Ron Brightwell, R&D Manager

Scalable System Software Department

U.S. DEPARTMENT OF ENERGY

NNSA National Nuclear Security Administration

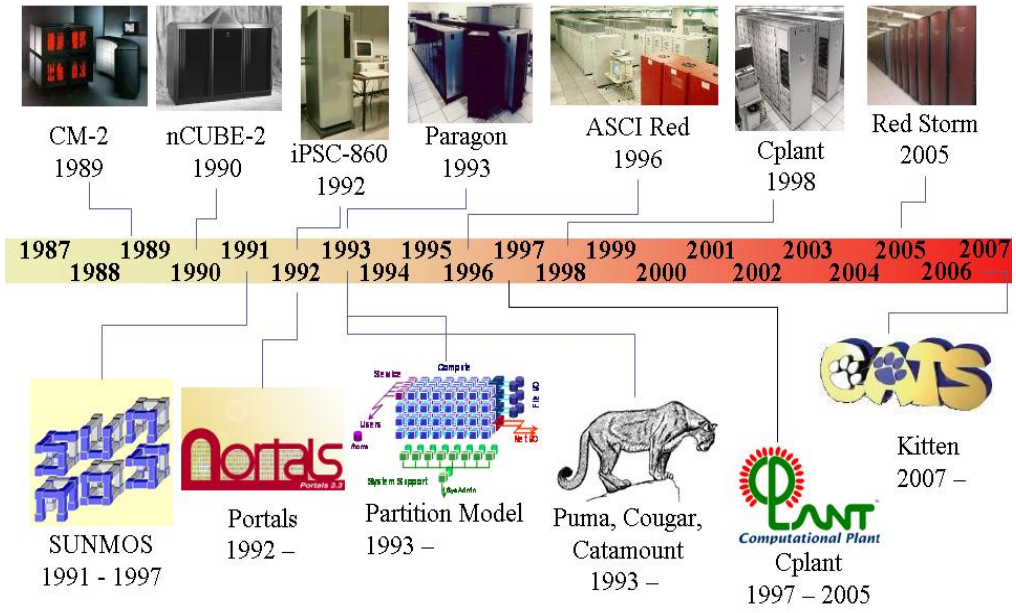
Sandia National Laboratories is a multi-program laboratory managed and operated by Sandia Corporation, a wholly owned subsidiary of Lockheed Martin Corporation, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

System Software@Sandia

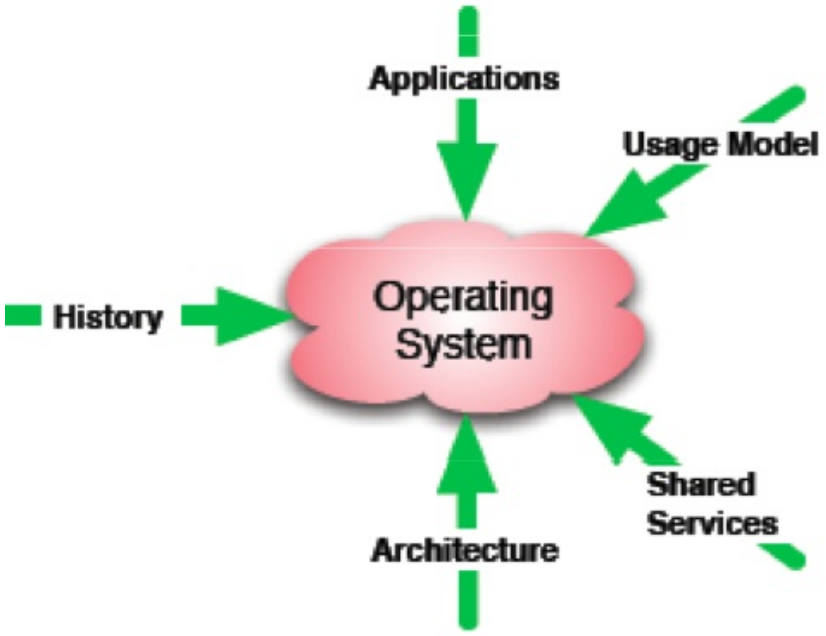
- Established the functional partition model for HPC systems
  - Tailor system software to function (compute, I/O, user services, etc.)
- Pioneered the research, development, and use of lightweight kernel operating systems for HPC
  - Only DOE lab to deploy OS-level software on large-scale production machines
  - Provided blueprint for IBM BlueGene OS
- Set the standard for scalable parallel runtime systems for HPC
  - Fast application launch on tens of thousands of processors
- Significant impact in the design and of scalable HPC interconnect APIs
  - Only DOE lab to deploy low-level interconnect API on large-scale production machines

AWARDS:

- **1998** Sandia Meritorious Achievement Award, TeraFLOP Computer Installation Team
- **2006** Sandia Meritorious Achievement Award, Red Storm Design, Development and Deployment Team
- **2006** NOVA Award Red Storm Design and Development Team
- **2009** R&D 100 Award for Catamount N-Way Lightweight Kernel
- **2010** Excellence in Technology Transfer Award, Federal Laboratory Consortium for Technology Transfer
- **2010** National Nuclear Security Administration Defense Programs Award of Excellence



Factors Influencing OS Design

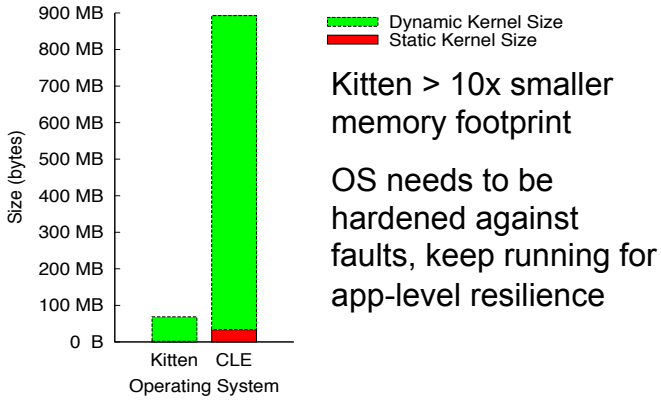


Sandia Lightweight kernels

- Applications
  - Small set of apps
- Programming model
  - MPI
- Architecture
  - Distributed memory
- Usage model
  - Space shared
- Shared services
  - Parallel file system
- History
  - None

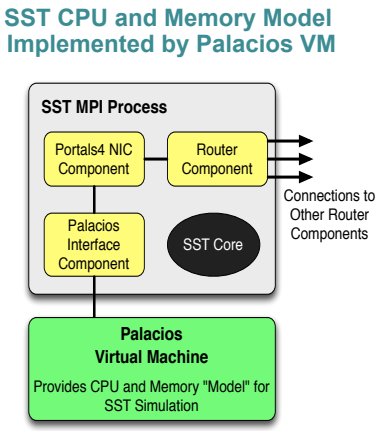
System Software Must Enable Co-Design

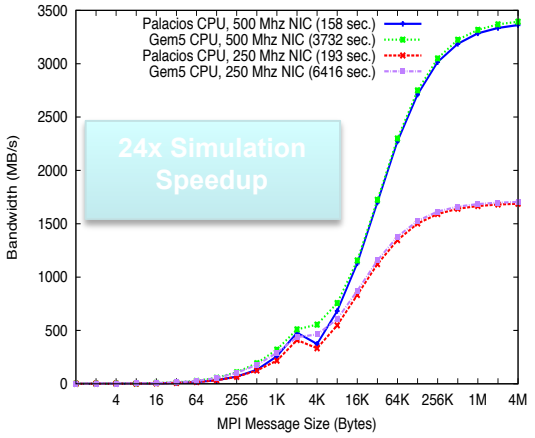
- Operating system and runtime (OS/R) software no longer an optimization layer between application and architecture
  - Sandia LWKs were optimized for MPPs
  - Evolving applications and architectures inhibit OS/R optimization
- Leveraging the agility of LWK and lightweight virtualization approach to support co-design
  - Motivations for mini-OS similar to mini-Apps – small and agile
  - Virtual machine enhances simulation capability and supports legacy application migration



Kitten > 10x smaller memory footprint

OS needs to be hardened against faults, keep running for app-level resilience





Extreme Scale Computing Grand Challenge

- Attempt to unify Sandia's physics and data analysis environments
  - How converge hardware?
  - How to converge system software?
- System software convergence based on building blocks approach
  - Kitten lightweight operating system
  - Portals network stack: networking stack with strong progress designed to support both MPI tagged matching and PGAS
  - Qthreads: User-level lightweight threading library with advanced synchronization

Portals 4 Network Stack

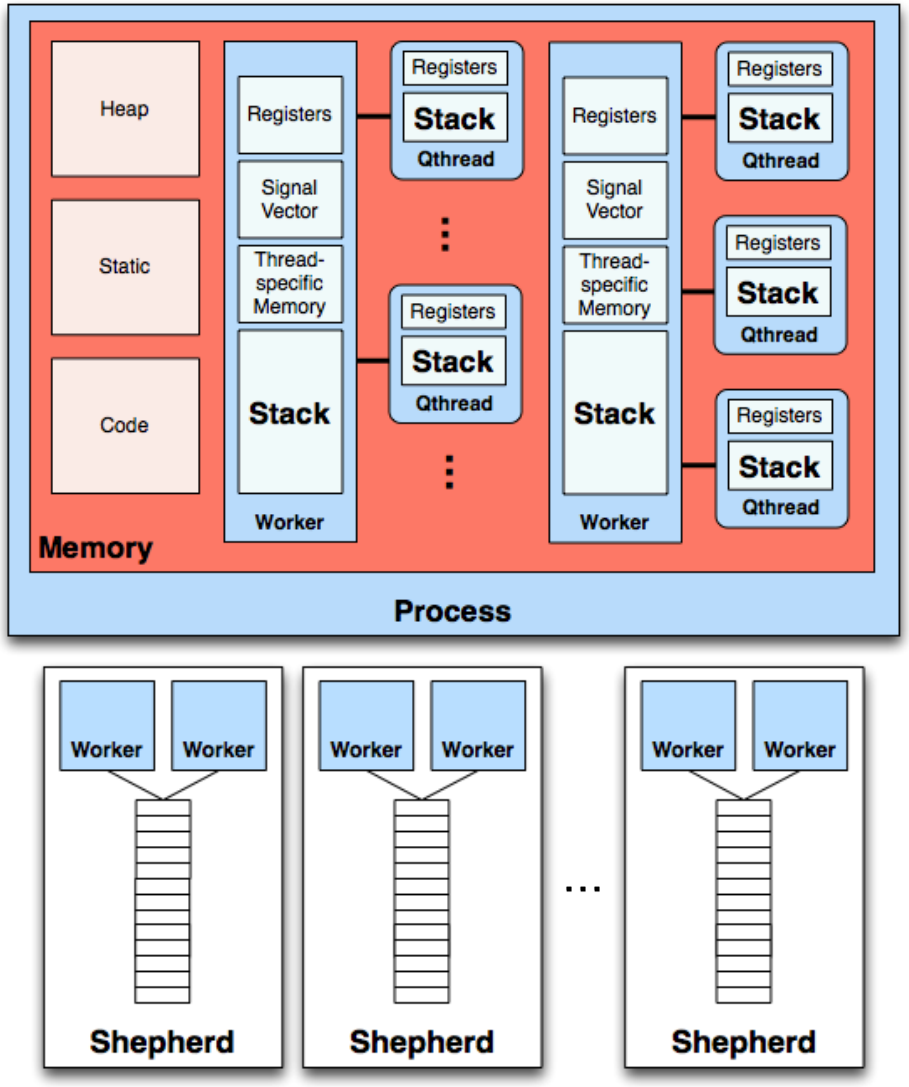
- Connectionless RDMA with matching
- Provides elementary building blocks for supporting higher-level protocols well
  - MPI, RPC, Lustre, etc.
- Allows structures to be placed in user-space, kernel-space, or NIC-space
- Receiver-managed offset allows for efficient and scalable buffering of MPI "unexpected" messages
- Supports multiple protocols within a process
  - Needed for compute nodes where everything is a message

XGC-Supported Parallel Programming Models

	Address Space	Parallel Loops	Asynchronous Tasks	Programming Interface	Execution Model
Qthreads	Shared	On-node	On-node	Library API	Work Queue
OpenMP	Shared	On-node	On-node	Compiler Directives	Fork/Join
MPI	Distributed	None	None	Library API	CSP
SPR	Distributed	On-node	Global	Library API	Work Queue
Chapel	Partitioned Global	Global	Global	Language	Global View

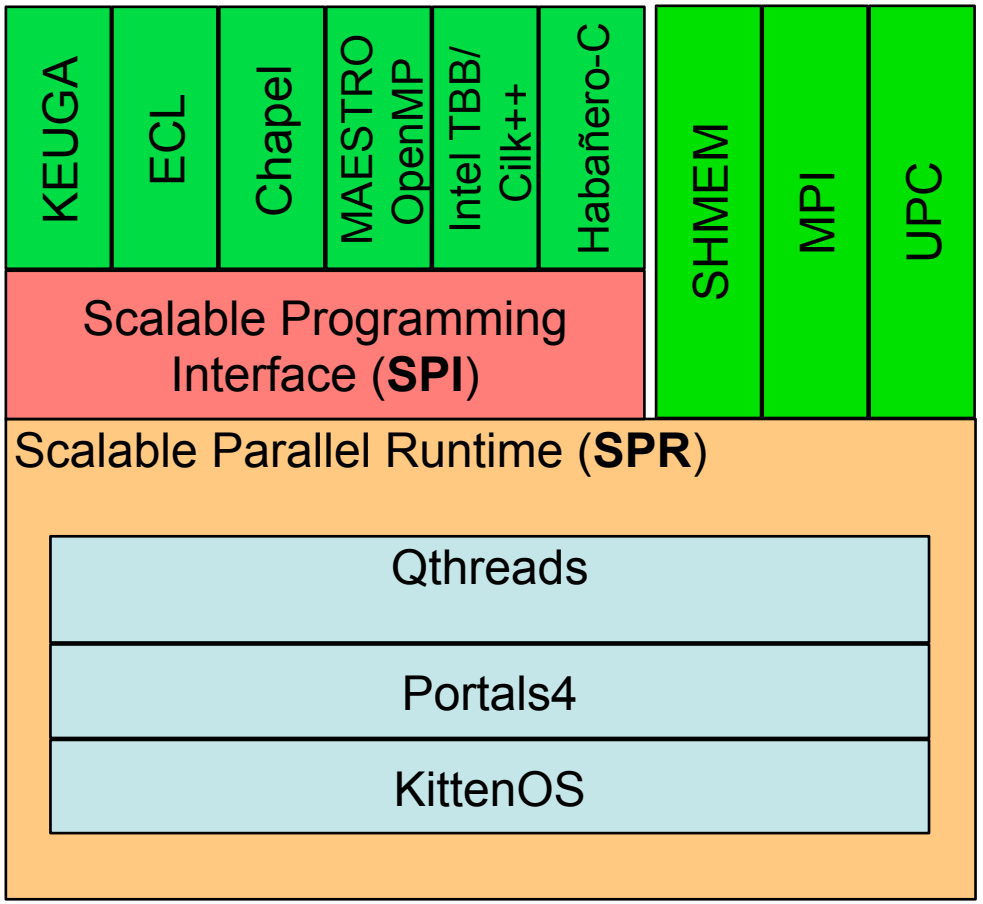
Qthreads Highlights

- Lightweight User-level Threading (tasking)
- Platform Portability
  - IA32/64, AMD64, PPC32/64, SparcV9+, SST, Tilera, ARM
  - Linux, BSD, Solaris, MacOSX, Cygwin
- Locality fundamental to model
  - "Shepherd" as thread-mobility domain
- Fine-grained synchronization
  - Full/Empty Bits (64-bit & 60-bit)
  - Mutexes
  - Atomic Operations (Integer incr, Float incr, & CAS)
- Locality-aware Cache-aware Workstealing Scheduler



Explore Revolutionary System Software to Preserve Evolutionary Application Software

- Our significant investment in the development and validation of the science and engineering application code base is a key driver for system software development
- A key goal for system software is to buffer the application code base from radical changes in the underlying hardware
- System Software Drivers:
  - Concurrency Management
  - Power Management
  - Resilience Performance



Open Questions

- How to support programming models such as MPI+X efficiently
  - Today, have two separate libraries with no interlocking
  - Need to co-schedule communication and work
  - How do we handle network's near-real-time needs with lightweight cooperatively scheduled threading models
- How do we transition existing codes from MPI to MPI plus a task-based lightweight threading model?
- How can Sandia's applications take advantage of parallel languages like Chapel or X10?

US DOE OS/Runtime Technical Council

- Summarize the OS/R-specific challenges
- Describe a model to integrate DOE-sponsored research with vendor products and support
- Assess the requirements of and impact on facilities, production support, tools, programming models, and hardware architecture
- Identify promising methods and novel approaches
- Write a report that can be referenced by FOA

Council Members

- Pete Beckman, ANL (co-chair)
- Ron Brightwell, SNL (co-chair)
- Bronis de Supinski, LLNL
- Maya Ghokale, LLNL
- Steven Hofmeyr, LBNL,
- Sriram Krishnamoorthy, PNNL
- Mike Lang, LANL
- Barney Maccabe, ORNL
- John Shalf, LBNL
- Marc Snir, ANL

Council Meetings

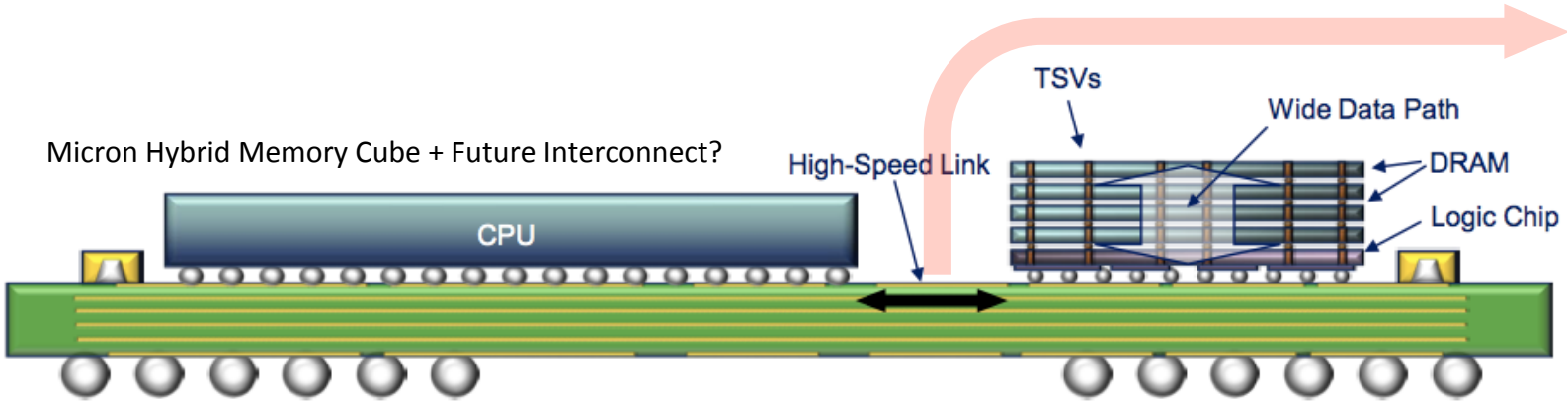
- March 21-22, 2012 – Washington, DC
- April 19, 2012 – Portland, OR (@ Exascale Planning Workshop)
- May 14-15, 2012 – Washington, DC
- June 11-12, 2012 – Washington, DC
- July 20-21, 2012 – Washington, DC (Vendor meeting)
- August 21, 2012 – VTC
- September 12-13, 2012 – Washington, DC & VTC
- October 3-4, 2012 – Washington, DC Workshop
- November 14, 2012 – Salt Lake City, Supercomputing 2012

Architecture Drivers for ExaOSR Software Changes (slide 1 of 2)

1

3D Memory & Integrated Interconnect

Micron Hybrid Memory Cube + Future Interconnect?



Jan 2012: “Early benchmarks show a memory cube blasting data 12 times faster than DDR3-1333 SDRAM while using only about 10 percent of the power."

2

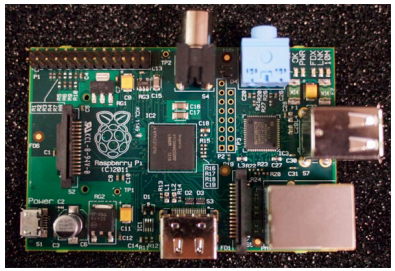
Parallelism

On-chip Parallelism Exploding: “The core is the Mhz”

- 2008: largest system had ~150K cores
- Today (2012)

LLNL BG/Q	1600K cores
RIKEN K	705K cores
Jülich BG/P	295K cores
ORNL XT5	224K cores
ANL BG/P	164K cores

IBM has successfully scaled the LAMMPS application to over 3 million MPI ranks on BG/Q



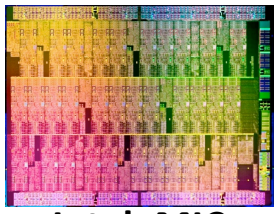
Raspberry Pi: $25

- 700MHz ARM11
- 4-core versions have been built

3

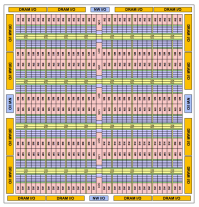
Power & Heterogeneity

Architecture Drivers for ExaOSR Software Changes (slide 2 of 2)



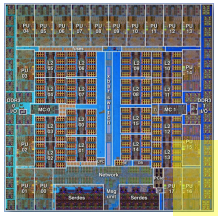
Intel: MIC

- Power-constrained Consistency
- Extreme specialization (hetero)
- Dark (dim) Silicon Management:



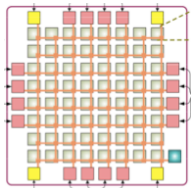
Dally: Echelon

Chien: 10x10



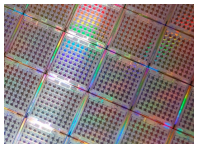
IBM: BG/Q

#18: unpowered



Godson T

Tilera

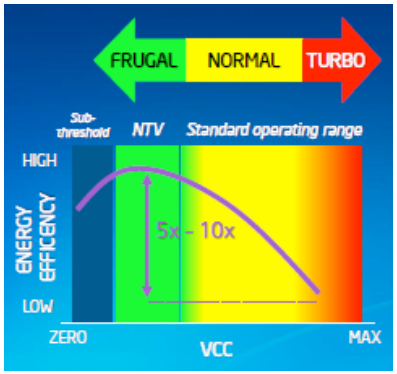


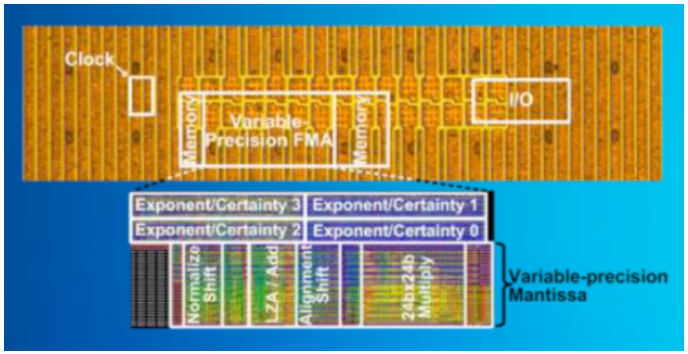
4

Variable Eff. & Precision

Advanced Chip Features:

- Near-threshold
- Variable Precision





In NTV range, 5 to 10 times more efficient

Intel demonstrated chip that can go from 3Mhz to 915Mhz

Key Observations for ExaOSR

- Massive Parallelism (exponential growth)
  - Dynamic parallelism and decomposition
  - Advanced run-time systems to manage tasks, dependencies, and messaging linked with scheduler
  - (with dynamic RTS, power and fault mgmt: "OS Noise" not an issue)
- Power as a managed system resource
  - Adjusting arithmetic precision, fault probability, directing power within global view at several levels
- Fault tolerance actively managed in software at many levels
  - Fault management with nodes and at global view
- Architecture organization (significant OS/R changes):
  - Heterogeneous cores, variable precision, specialized functional units
  - Deep memory hierarchies: 3D RAM, NVRAM on node
    - New models for deep memory hierarchy
    - Multi-level Parallelism within the node to hide latency
    - Memory logic

Other Challenges: Business/Social/Total Cost

- Preserving code base
- Vendor business models
- Sustainability/portability
- “Scale Down” important: from the extreme scale to the broader HPC marketplace
- Must address broad range of scientific domains
- DOE does not want an unsupported OS/R

The new ExaOSR will be a Global OS/R

Existing HPC Systems Have Focused on "Node" + ad hoc services/libraries

- Currently, the control systems (RAS) monitor system health
- Exascale systems need to *manage power/performance* and *respond to health*

Two Examples: Power and I/O Bandwidth

- Whole System
  - **Power**: Set budgets for each job/partition and file system; schedule jobs based on differentiated power demands
  - **I/O Bandwidth**: Orchestrate/arbitrate BW sharing across jobs, schedule jobs based on I/O mix to reduce contention
- Within Job/Partition
  - **Power**: Manage power across nodes within set budget; respond to system requests to dynamically adjust power consumption
  - **I/O Bandwidth**: Manage NVRAM as burst buffer to reduce I/O contention
- Node
  - **Power**: Manage functional units & dark silicon within nodes for best throughput for given power budget; respond to requests to dynamically adjust power consumption
  - **I/O Bandwidth**: Use compression when power and ops are available

Application OS/R Requirements: Feedback

- Support for:
  - I/O
  - Resilience and system health
  - Dynamic libraries
  - Debugging at scale and ease of use
  - In situ analytics and real-time visualization
  - Threads: creation, management, synchronization
- Desire to automate or be agnostic of power/energy and resilience
- Support new features (eg., non-blocking collectives, neighborhood collectives, ..)

Tool OS/R Requirements Overlap Those of Applications

- Bulk launch for scalability; mapping & affinity matter
- Low overhead way to cross protection domains
- Quality of service concerns for shared resources
- Can have extensive I/O requirements
  - Support for in-situ analysis is critical
- Need OS/R support to handle heterogeneity & scale
  - Synchronization for monitoring
- Need well defined APIs for information about key exascale challenges
  - Power and resilience
  - Asynchrony (API needs may be distinct)

Tool OS/R Requirements Extend Those of Applications

- Must launch with access to application processes
- Low overhead timers, counters & notifications
- Monitoring, access to protected resources
- Attribution mechanisms
  - Aggregation and differentiation
  - Process, resource and source code (including call stack) correspondence
  - Need HW support for shared activities?
- Measurement conversions?
- Multicast/reduction network (shared with OS/R)
- Less clear where tool ends and OS/R begins

*

Facilities

- System analysis
  - Log data: anonymization, mining, common formats
  - Per user, per job data incl. energy and errors
  - Scalable memory usage monitoring
  - Live real-time fault and RAS data
- Fault management
  - Offline and online system diagnostics
  - Ability to run single-node tests without impacting other jobs
  - Automatic handling of boot failures
  - Cope with node failure (e.g. through migration)
- Workflows
  - Non-traditional HPC workflows
    - (many small jobs e.g. bioinformatics)

Facilities (cont.)

- System-wide energy management
  - Power-capping, control and monitoring
  - Job scheduling dependent on power (e.g. peak vs off-peak)
- Performance
  - QoS I/O management
  - Topology-aware, improved job placement (e.g. using migration)
  - Fast launch time for huge jobs
  - Fast booting
- Maintenance
- Ease of upgrades: rolling upgrades, partial upgrades, rollback
- Partitioning
  - Multiple OS partitions (OS per job)
  - User-specialization, multiple software stacks
  - More OS functionality, e.g. syscall support

Vendor Input: Motivation

- It is not feasible for DOE to be the sole maintainer or developer of an exascale OS/R
- Want a common APIs to develop interoperating solutions across hardware, but we need vendor cooperation to achieve this.
- Draw from existing vendor experience and current research directions.
- Vendors were chosen from their participation in past government procurements and research programs, not all vendors contacted provided a representative or responded to all of the questions.
- To focus the responses of the vendors we provided them with guiding questions focusing on technical and the business model, and a strawman OS/R design.

Vendor Input: Intersection

Technical Issues

CRAY / IBM / INTEL / NVIDIA

- Entire software stack must be be tightly integrated
- Need to collect detailed information at all OS/R layers
- Need common APIs to pass information between OS/R layers.
- Need for auto tuning and auto placement but with the ability for fine-grained control if needed by runtime.
- Power: need common APIs & at different layers: Site, System, Node
- Need to support processor heterogeneity & deep memory hierarchies -- Good ideas, but no clear common path
- Reliability – Good ideas, but no clear common path

*

Vendor Input: Intersection

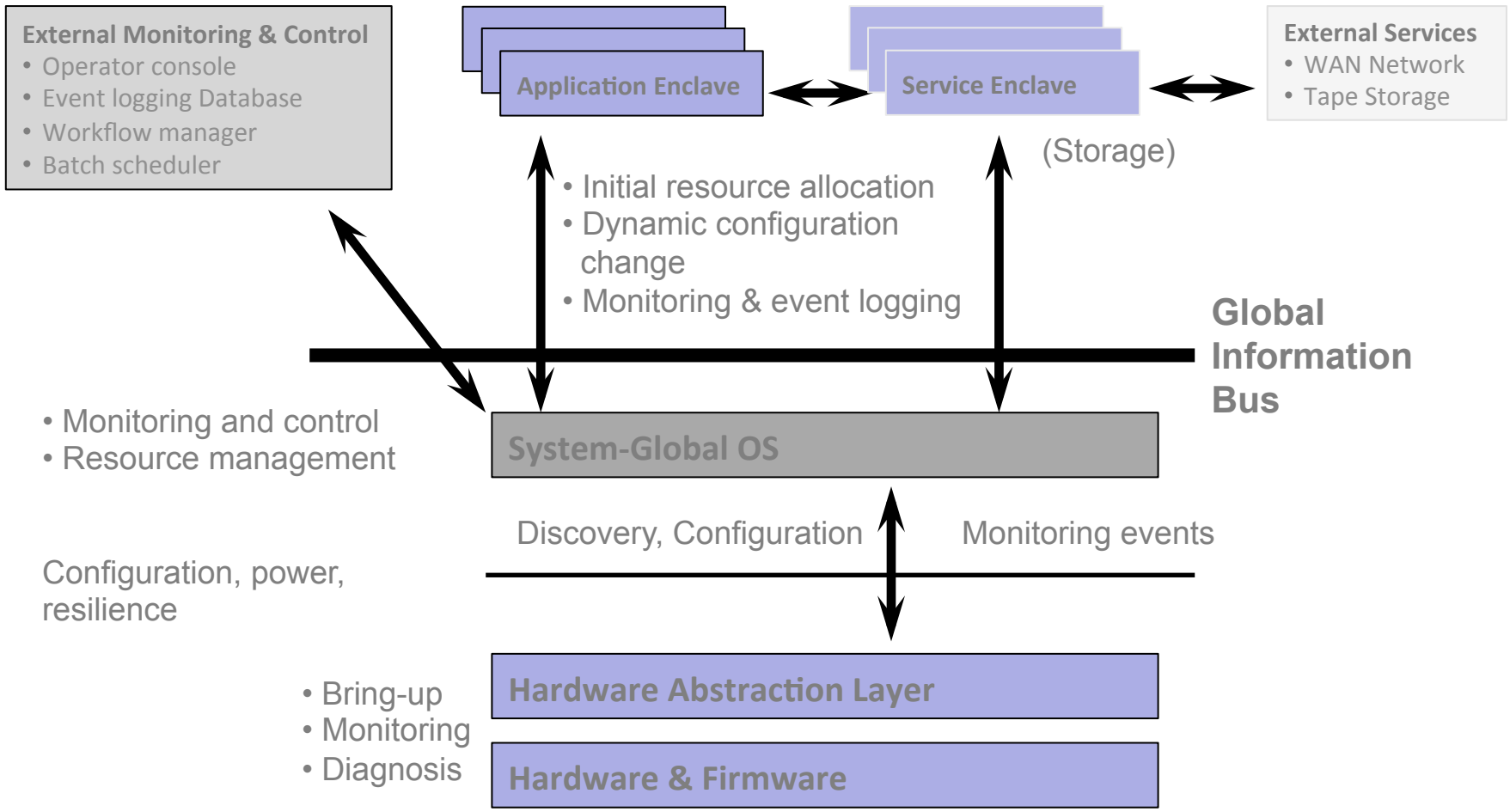
Business Engagement
CRAY / IBM / INTEL / NVIDIA

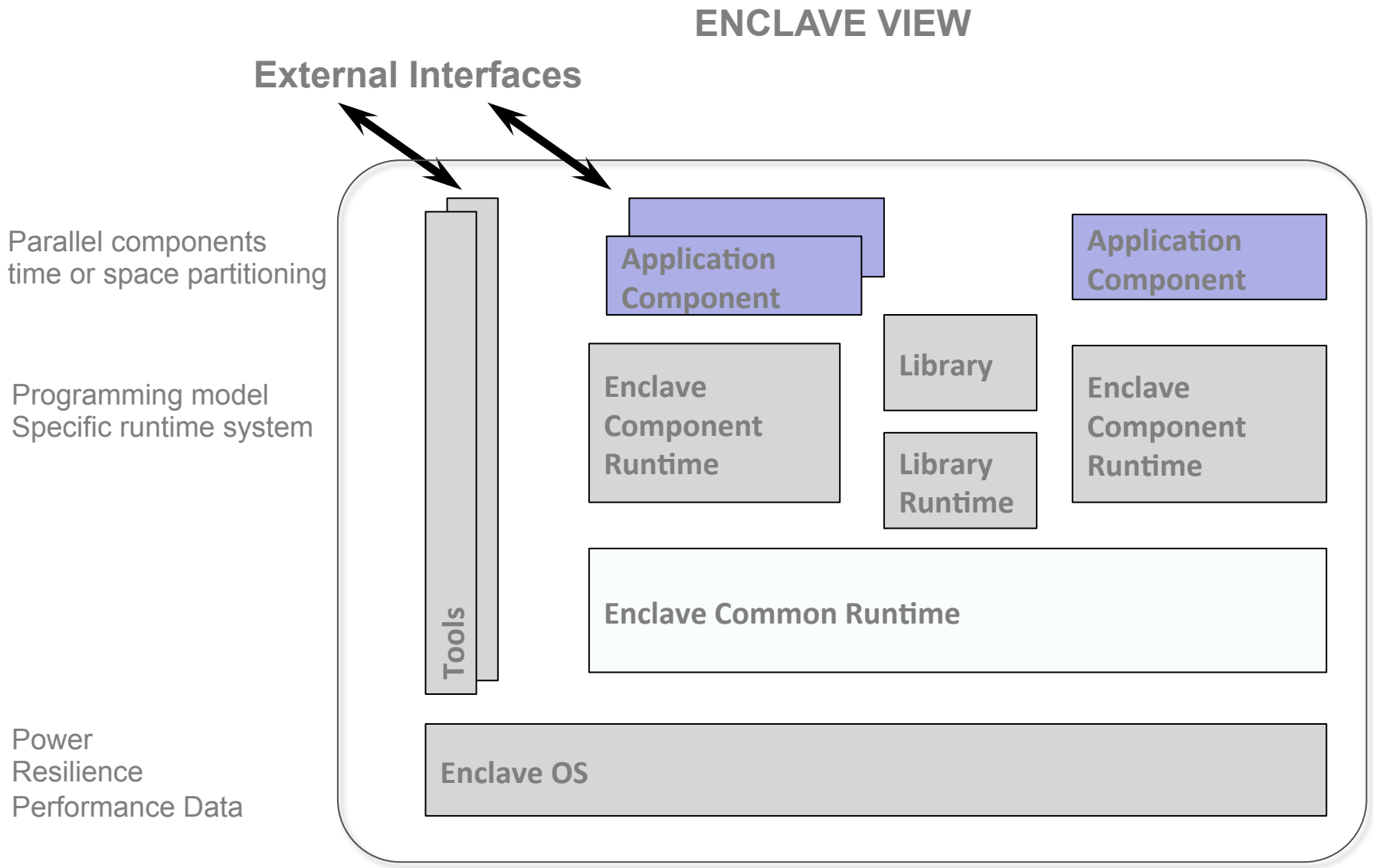
- Vendors have a track record of successfully integrating open source.
- Vendors prefer working with DOE by using open source
- Vendors require some proprietary software to differentiate solutions. Seek patents and other protective licensing
- In new *Complex Vendor Landscape* business models are diverse
- Collaborative software could be managed by a 3rd party to limit liability and provide long-term support
- Vendors must build on larger markets to survive

Exascale Software Architecture

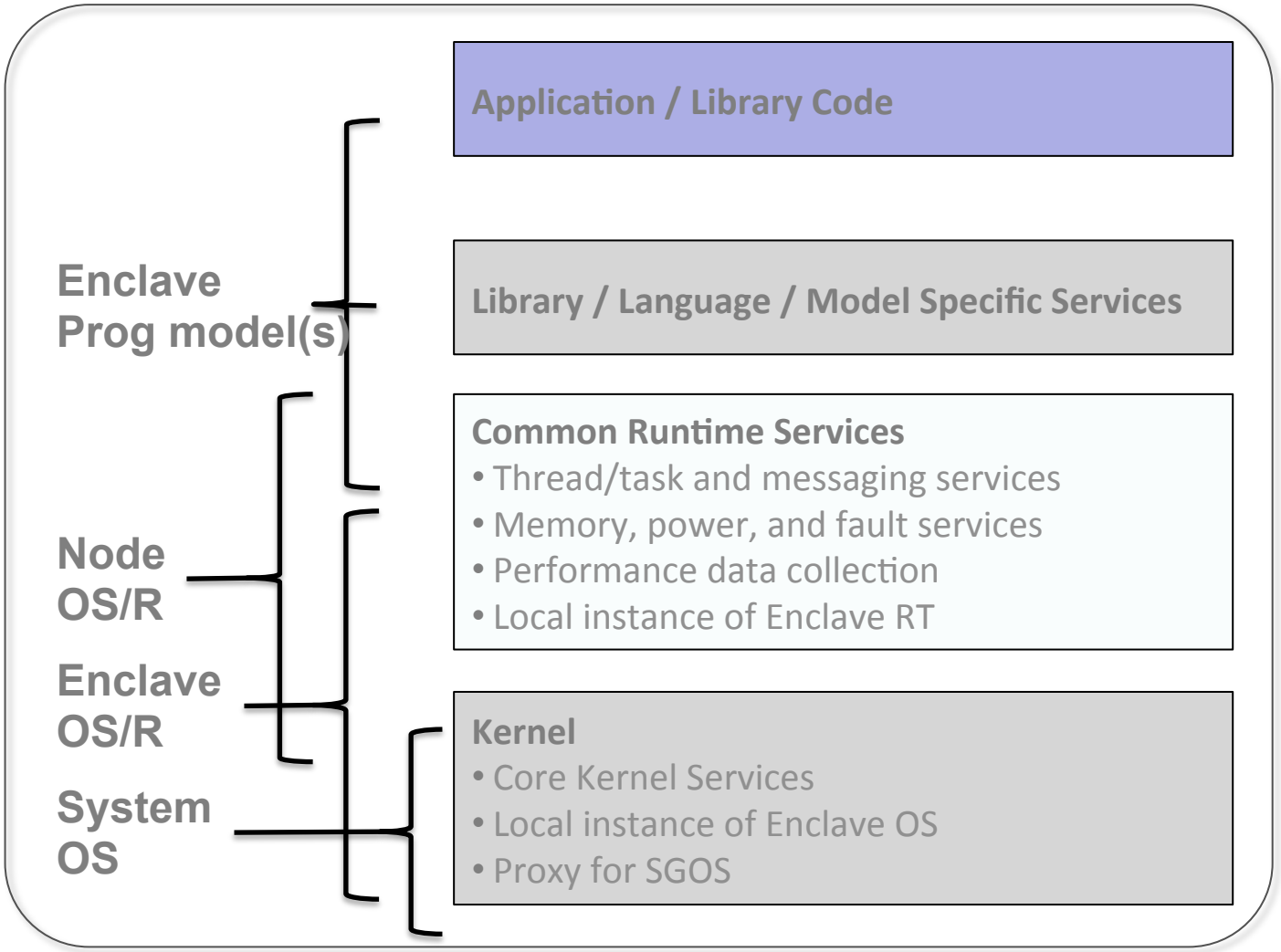
- Goal:
  - Create a common framework to describe software architecture and programming interfaces
  - Identify areas where design decisions have to be made and interfaces that are candidates for standardization
- Three levels of services:
  - Node (thread scheduling, memory management…)
  - *Enclave* (aka partition): Set of nodes dedicated to an application or a service such as I/O (user-space communication, error recovery…)
  - System-wide

System View


External Monitoring & Control
• Operator console
• Event logging Database
• Workflow manager
• Batch scheduler
Application Enclave
Service Enclave
External Services
• WAN Network
• Tape Storage
(Storage)
• Initial resource allocation
• Dynamic configuration change
• Monitoring & event logging
Global Information Bus
• Monitoring and control
• Resource management
System-Global OS
Discovery, Configuration
Monitoring events
Configuration, power, resilience
• Bring-up
• Monitoring
• Diagnosis
Hardware Abstraction Layer
Hardware & Firmware


ENCLAVE VIEW
External Interfaces
Parallel components
time or space partitioning
Programming model
Specific runtime system
Power
Resilience
Performance Data
Application Component
Application Component
Library
Enclave Component Runtime
Enclave Component Runtime
Library Runtime
Tools
Enclave Common Runtime
Enclave OS

NODE-LOCAL VIEW



Evolution

- Contain services to smallest possible container
  - E.g., enclave-level loader, enclave-level recovery
- Add services for energy management and recovery
- Add ability to negotiate interfaces (e.g., failure reporting)
- Standardize interfaces:
  - HAL -> OS
  - External -> System-wide OS
  - System-wide OS -> Enclave OS/R
  - Enclave OS/R -> application, library, language R/T

The Workshop
Position Papers (all online)

Area	Count
Architecture/Structure	13 (3)
Autonomic/Adaptation	9 (6)
Core Specialization	9 (4)
Fine Grained/Dynamic Tasks	5 (3)

Area	Count
Power	6 (3)
Resilience	7 (3)
Unified RTS	3
Global OS	6
Other	6 (3)

- Total of 80 submissions
- Out of Scope: 16

- https://collab.mcs.anl.gov/display/exaosr/Position+Papers

2002 OS/R Issues and Challenges

- Fault tolerance / resilience
- Programming models
- OS structure
- APIs
- Specific functionality
- Scalability
- Interactivity
- Future hardware
- Hardware support for Oses
- Application requirements
- Metrics
- Programmatic challenges
- Heterogeneity
- Degree of transparency
- Infrastructure support for multiple OS/Rs
- Vendor proprietary components
- Tools support/requirements
- Desktop integration
- Dynamic resource management
- Vendors
- Testbeds
- Adaptation
- Usage models
- Memory hierarchy
- Security
- Standards
- Portability
- Culture
- Non-traditional architectures
- Multiple management policies
- Mainstream technology overlap
- Support for introspection
- Interface to RAS
- Testing
- Application requirements
- Intellectual property
- Sustainability
- Energy/power

DOE LAB 13-02 FOA

Exascale Operating and Runtime Systems Program

- $7M of funding for OS/R research at DOE labs
- Focus areas
  - Power management
    - Adaptive power management to meet 20 MW goal
  - Support for dynamic programming environments
    - Manage billions of threads
  - Programmability and tuning support
    - Dynamic adaptation and debugging
  - Resilience
    - Predict, detect, contain, and recover from faults
  - Heterogeneity
    - Hierarchical process and memory systems
  - Memory management
    - Use of new memory technologies
  - Global optimization
    - Manage resources with a system-wide view

Exascale OS/R Focus is on Hardware

- Reliability/Resilience
- Power/Energy
- Heterogeneity
- Memory hierarchy
- Cores, cores, and more cores

- Risk
  - Hardware advancements and investments can provide orders of magnitude improvement
  - OS/R advancements can provide double-digit percentage improvement

Everything I Know About Resiliency I Learned in Kindergarten

- Clean up your own mess
  - Hardware is largely responsible for the increased need for resiliency, so the hardware community needs to (help) solve it
- Play fair
  - Hardware should do its part to enable low-overhead approaches
- Share everything
  - Need hardware-level interfaces for examining and recovering state
  - Shouldn't have to try to guess about whether a component is about to fail
- Don't take things that aren't yours
  - Need more protection mechanisms
- Say you're sorry when you hurt somebody
  - Be explicit when a hardware component has failed
- Put things back where you found them
  - There's some analogy to virtual addresses and memory here…
- Flush
  - Don't leave unwanted state lying around for software to clean up
- Take a nap every afternoon
  - Ok, maybe not everything….

What About Applications?

- Focus is on parallel (multi-core) programming model
  - Advanced runtime systems
  - Node-level resource allocation and management
  - Managing locality
  - Extracting parallelism
  - Introspective, adaptive capabilities
    - This is really hard
- Risk
  - Incremental approach (OpenMP) wins
    - Advanced runtime capabilities are overkill
  - No clear on-node parallel programming model winner
    - Difficult to optimize OS/R

Application Composition is Key

- Little attention focused on how applications are constructed
- Clunky interfaces like mmap, ptrace, python etc. for sharing data
- Tools stress OS functionality because of these legacy APIs and services
- Integrating simulation and analysis is important
  - Both from a marketing and technical perspective ☺
- Advanced workflows are driving capability
- Lots of use cases
  - Ensemble calculations for UQ
  - Multi-x simulations
  - In-situ analysis
  - Graph analytics
  - Performance and correctness tools
  - Burst buffers for I/O?
- Requirements are driven by applications
  - Not necessarily by parallel programming model
  - Insulated from hardware advancements

CTH Analysis using ParaView

Comparing In-Situ with In-Transit Analysis

Motivation for In-Transit

- Analysis code may not scale as well as HPC code
- Direct integration may be fragile (e.g., large binaries)
- "Fat" nodes may be available on Exascale architectures (e.g., burst-buffer nodes)

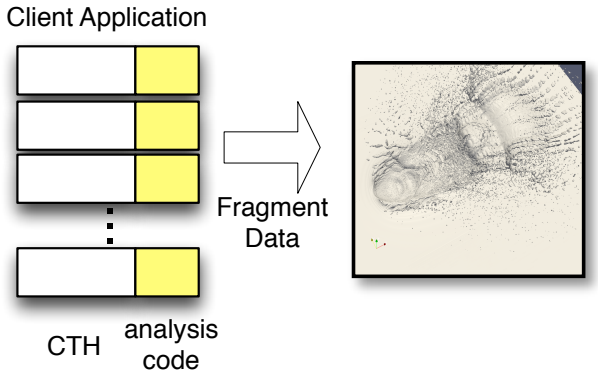
CTH fragment detection service

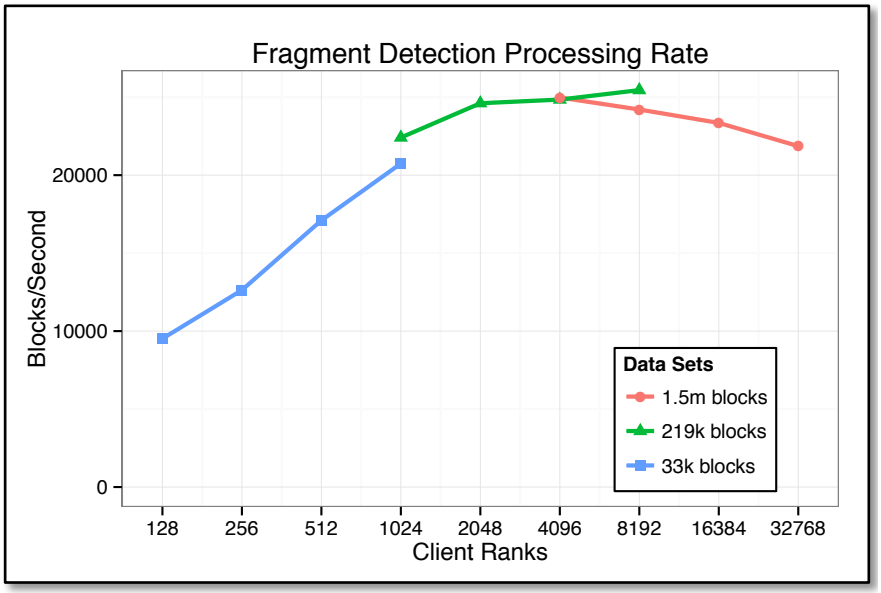
- Extra nodes provide in-line processing (overlap fragment detection with time step calculation)
- Only output results to storage (reduce I/O)
- Non-intrusive – Looks like in-situ (pvspy API)

Issues to Address

- Number of nodes for service
  - Based on memory requirements
  - Based on computational requirements
- Placement of nodes

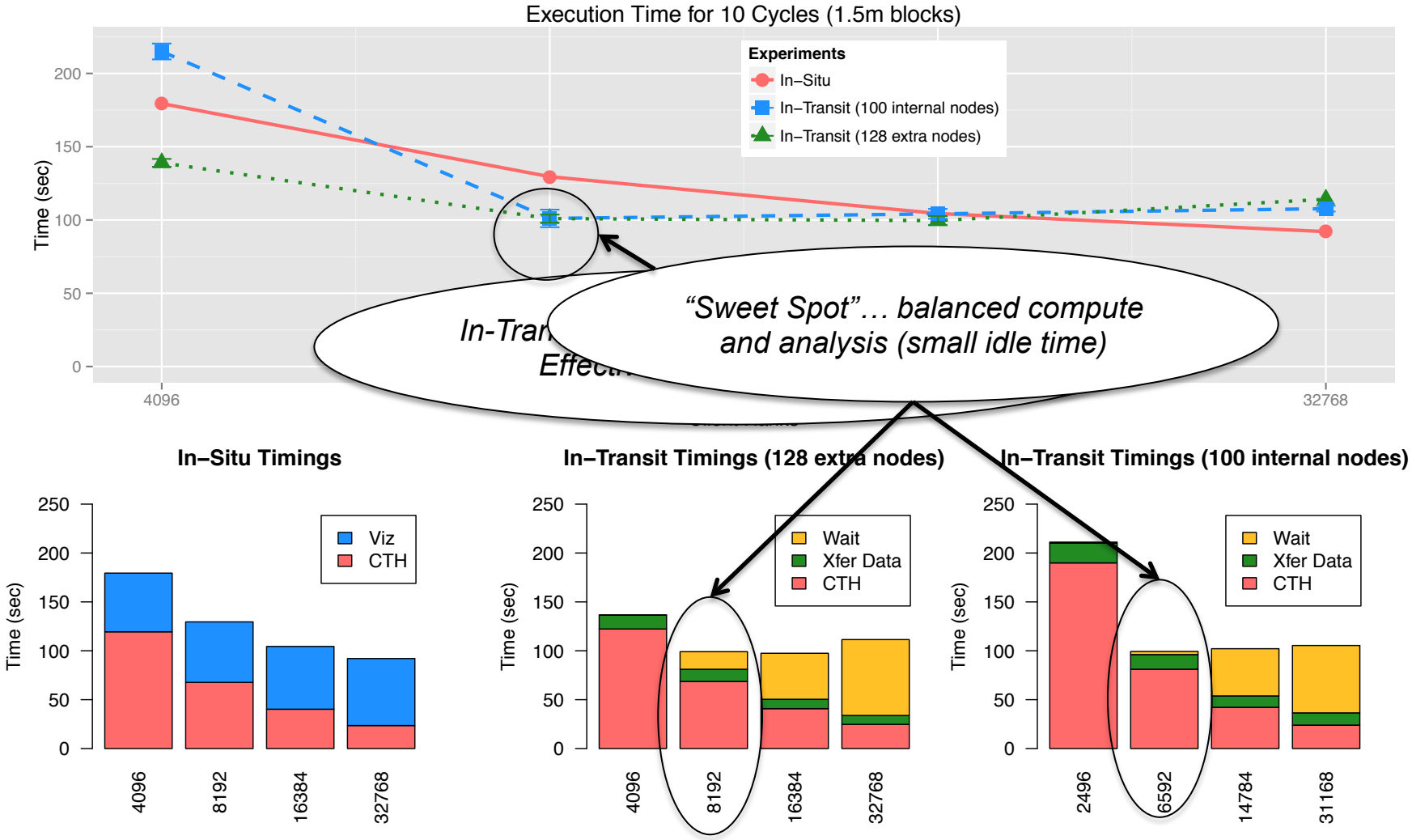
In-Situ Analysis





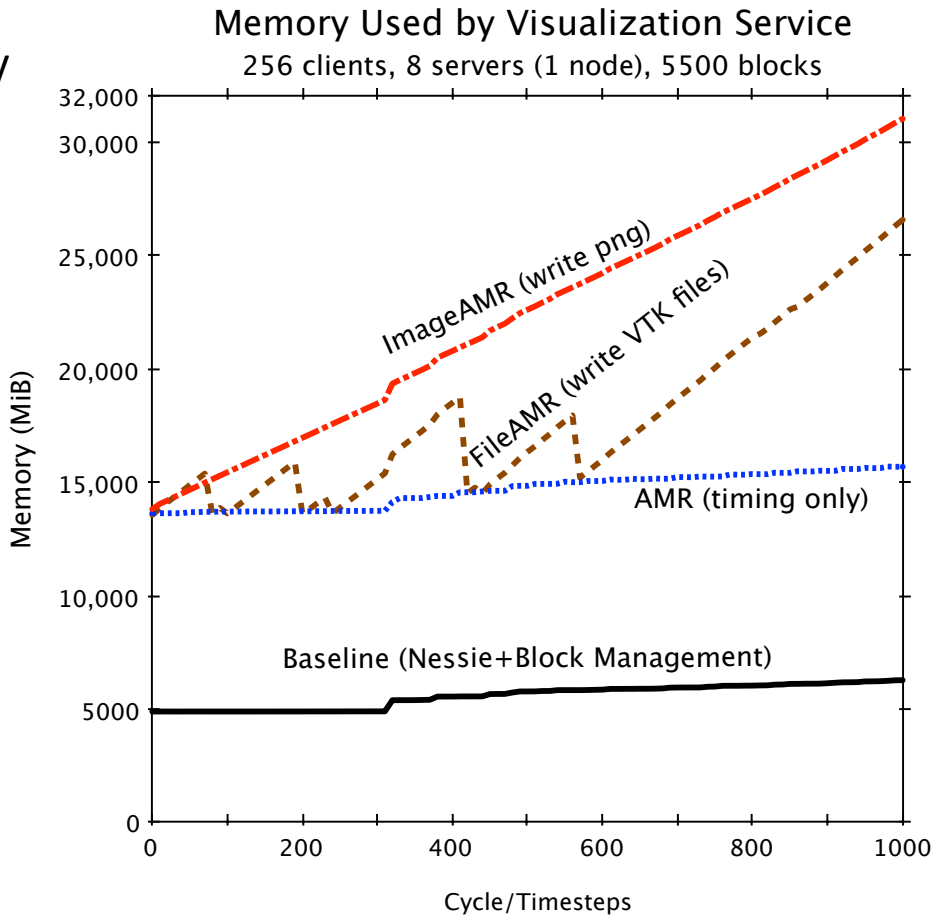
Strong Scaling Results

CTH Analysis 1.5m AMR Blocks



Memory Issues

- Analysis libs requires substantial memory
  - Offloading analysis allows for larger simulations
- Client binary size (static binaries)
  - CTH with ParaView Lib (360 MB)
  - CTH with In-Transit Lib (32 MB)
- Memory Requirements for In-Transit
  - One node can manage ~16K AMR blocks from CTH.
  - 4:1 – 16:1 ratio of compute nodes to service nodes (depending on problem size)
  - Current ParaView implementation has leak issues (being addressed)

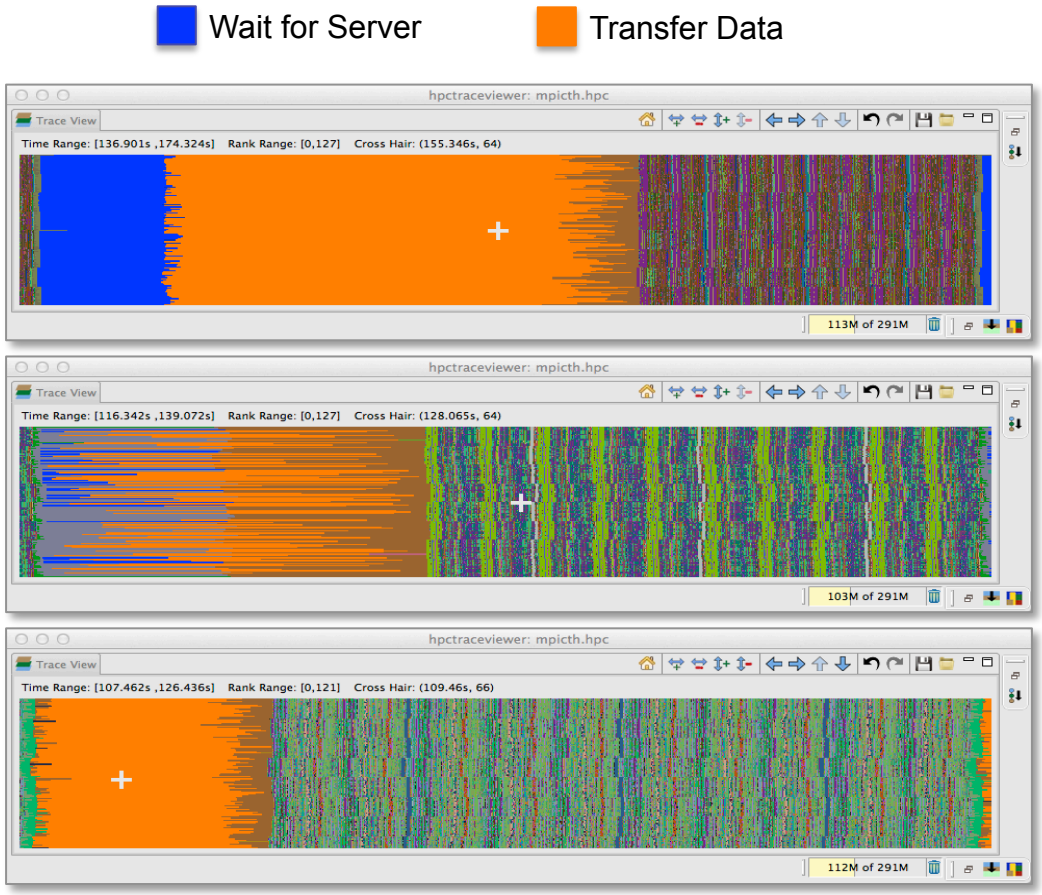


Load Balancing Issues

Reduce wait time on the client by adding cores to srvr

Ten Cycles of 128-core job (1 srvr)

- 2 server cores – 64:1
  - 10 cycles in 37 secs
  - Client idle waiting for server to complete (also affects transfers)
- 4 server cores – 32:1
  - 10 cycles in 23 secs
- 8 server cores – 16:1
  - 10 cycles in 19 secs
  - Less than 1% time waiting

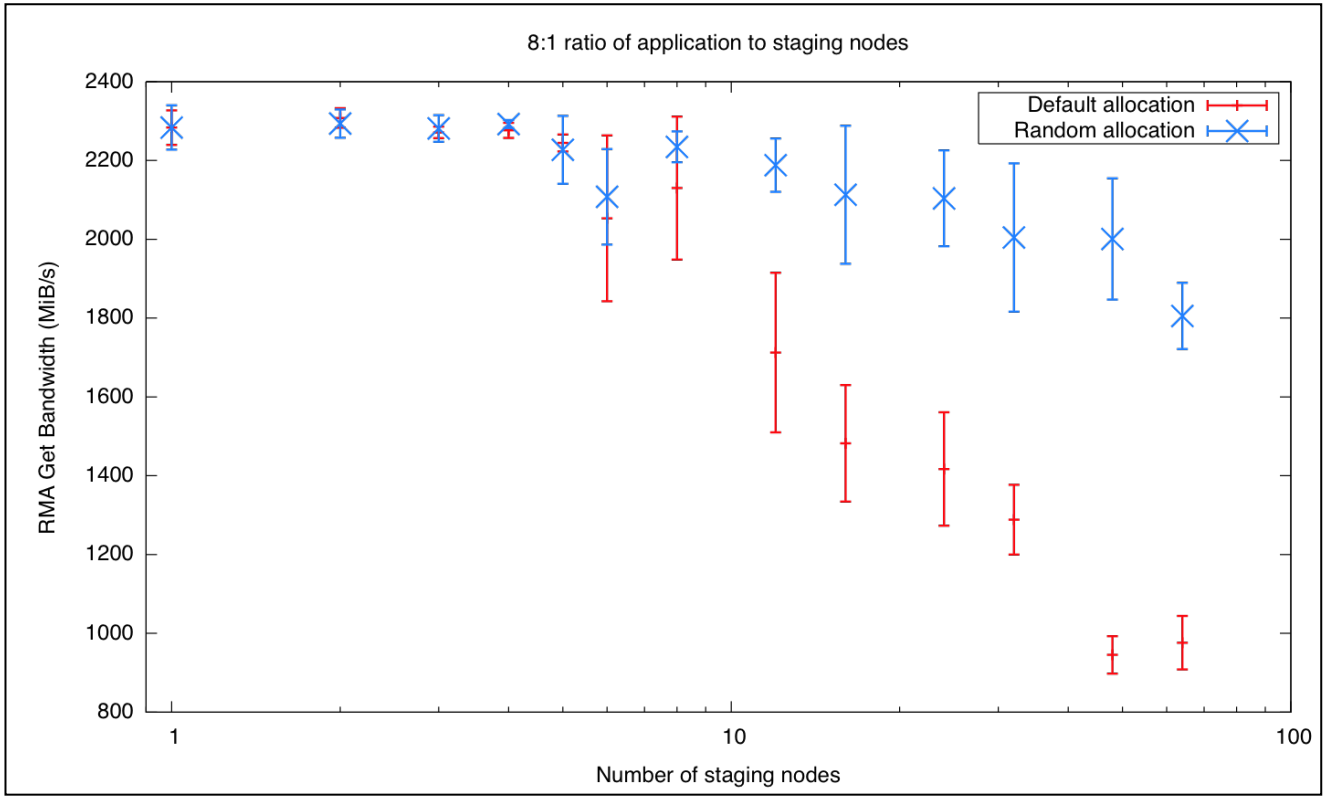


Adding cores can reduce analysis time, but increase memory requirements… it's a balancing act.

Impact of Placement on Performance

Work In-Progress…

- We know placement is important from previous study
- Goal is to place nodes within given allocation to avoid network contention
  - App-to-app (MPI), app-to-svc (NTTI), svc-to-svc (MPI), svc-to-storage (PFS)
  - Graph partitioning based on network topology and application network traffic



OS/R is Enabling Technology

- Need to support advanced run-time systems and approaches to resilience and energy, not necessarily provide solutions
- Follow BASF mantra
  - We don't make it, we make it possible
- OS/R should focus on providing capability, not just overcoming limitations of current hardware
- Application composition is the responsibility of the OS/R
  - Capability will be required regardless of underlying hardware or overlying parallel programming model

Acknowledgments

- Sandia
  - Ron Oldfield
  - Brian Barrett
  - Kyle Wheeler
- OS Technical Council
  - Pete Beckman, ANL
  - Marc Snir, ANL
  - Mike Lang, LANL