

HPC System Software Research at Oak Ridge National Laboratory

Dr. Christian Engelmann and Dr. Stephen L. Scott

**Computer Science and Mathematics Division
Oak Ridge National Laboratory**

Largest Multipurpose Science Laboratory within the U.S. Department of Energy

- Privately managed for US DOE
- \$1.4 billion budget
- 4600+ employees total
- 3,000 research guests annually
- 30,000 visitors each year
- Total land area 58mi² (150km²)

- Nation's largest energy laboratory
- Nation's largest science facility:
 - The \$1.4 billion Spallation Neutron Source
- Nation's largest concentration of open source materials research
- Nation's largest scientific computing facility

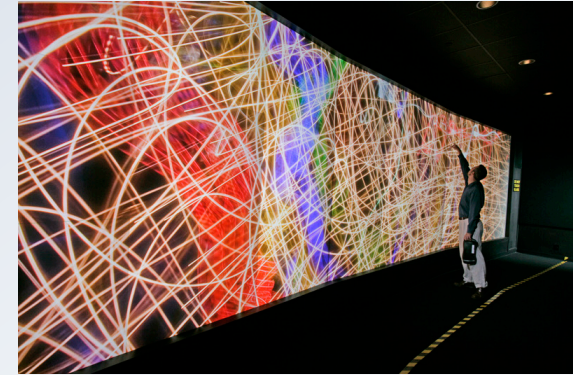
ORNL East Campus: Site of World Leading Computing and Computational Sciences



National Center for Computational Sciences

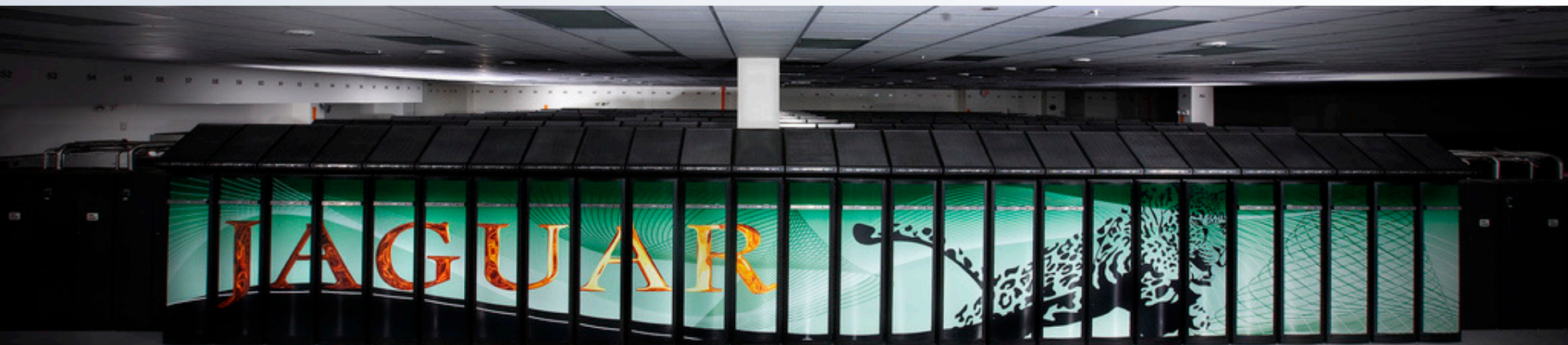
◆ 40,000 ft² (3700 m²) computer center:

- 36-in (~1 m) raised floor, 18 ft (5.5 m) deck-to-deck
- 36 MW of power with 6,600 t of redundant cooling
- High-ceiling area for visualization lab: 35 MPixel PowerWall



◆ 5 systems in the Top 500 List of Supercomputer Sites:

- 1. Jaguar XT5: Cray XT5, with 224,162 processor cores at 2,331 TFlop/s peak
- 3. Kraken: Cray XT5, with 98,928 processor cores at 1,028 TFlop/s peak
- 16. Jaguar XT4: Cray XT4, with 30,976 processor cores at 260 TFlop/s peak
- 30. Athena: Cray XT4, with 17,956 processor cores at 165 TFlop/s peak
- 370. Eugene: IBM BGP, with 8,192 processor cores at 28 TFlop/s peak



#1: Jaguar at Oak Ridge National Laboratory

Blade = 4 Nodes

8 processors
48 cores
4 interconnect chips
16 (4 GB) memory modules = 64 GB
6 voltage converters

Node = 2 Processors

6 cores per processor

1 interconnect chip

4 x (4 GB) memory modules = 16 GB

Processor = 6 Cores

2 memory modules

8 GB

8 GB

Cray
SeaStar2+
Interconnect

Cabinet = 24 Blades

1152 cores

96 interconnect chips

384 memory modules (1.5 TB)

144 voltage converters

+ power supply, liquid cooling, etc.

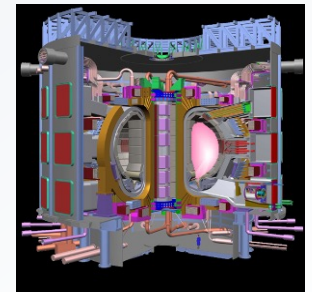
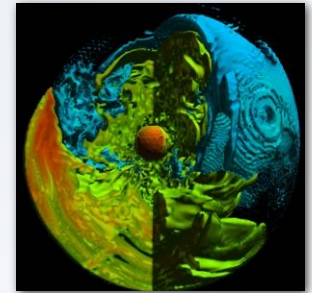
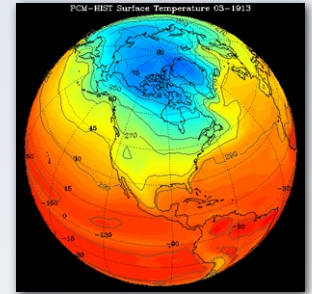
Power 480V, ~40,000 Watt per cabinet

Jaguar = 284 cabinets (XT5 and XT4), ~ 6.5 Megawatts

Cray XT5

At Forefront in Scientific Computing and Simulation

- Leading partnership in developing the National Leadership Computing Facility
 - Leadership-class scientific computing capability
 - Currently planning for 10-20 PFlop/s in 2012
 - On the path toward:
 - 100 PFlop/s in 2015 (10- 100 million cores)
 - 1,000 PFlop/s in 2018 (100-1,000 million cores)
- Attacking key computational challenges
 - Climate change
 - Nuclear astrophysics
 - Fusion energy
 - Materials sciences
 - Biology
- Providing access to computational resources through high-speed networking

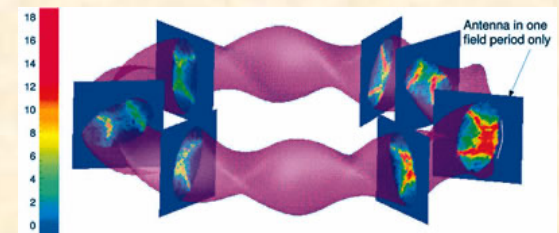
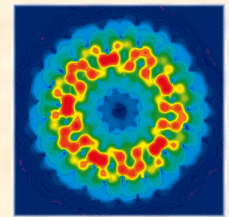


Computer Science and Mathematics Division

- Applied research focused on computational sciences, intelligent systems, and information technologies

- CSM Research Groups:

- Application Performance Tools
- Complex Systems
- Computational Astrophysics
- Computational Earth Sciences
- Computational Engineering and Energy Sciences
- Computational Materials Science
- Computational Mathematics
- **Computer Science Research (23 researchers & postdocs)**
- Future Technologies
- Statistics and Data Science



Computer Science Research Group Projects

- Parallel Virtual Machine (PVM)
- MPI Specification, FT-MPI and Open MPI
- Common Component Architecture (CCA)
- Open Source Cluster Application Resources (OSCAR)
- Scalable cluster tools (C3)
- Scalable Systems Software (SSS)
- Fault-tolerant metacomputing (HARNESS)
- High availability and resilience (RAS, FAST-OS 1 & 2)
- Super-scalable algorithms research
- Distributed file and storage systems (Freelader)



FT-MPI



System-level Virtualization Research

Christian Engelmann

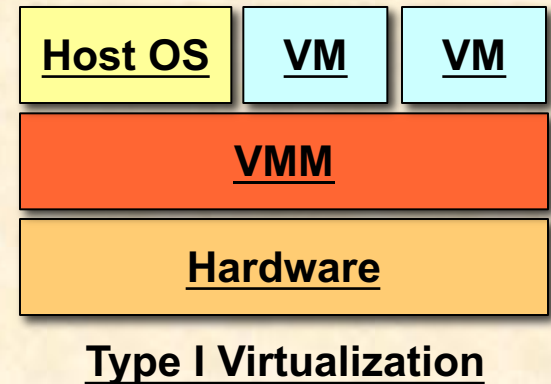
**Computer Science and Mathematics Division
Oak Ridge National Laboratory**

Research and Development Areas

- **Efficient hypervisors for limiting interferences with scientific applications in HPC systems**
- **Minimal host operating system for reduced system footprint of system-level virtualization solutions**
- **System management tools for supporting virtualized and standard HPC systems in disk-full and disk-less scenarios**
- **Performance characterization of scientific applications running in virtual machines**
- **Configurable virtual system environments for adaptation of HPC system properties to scientific application needs**

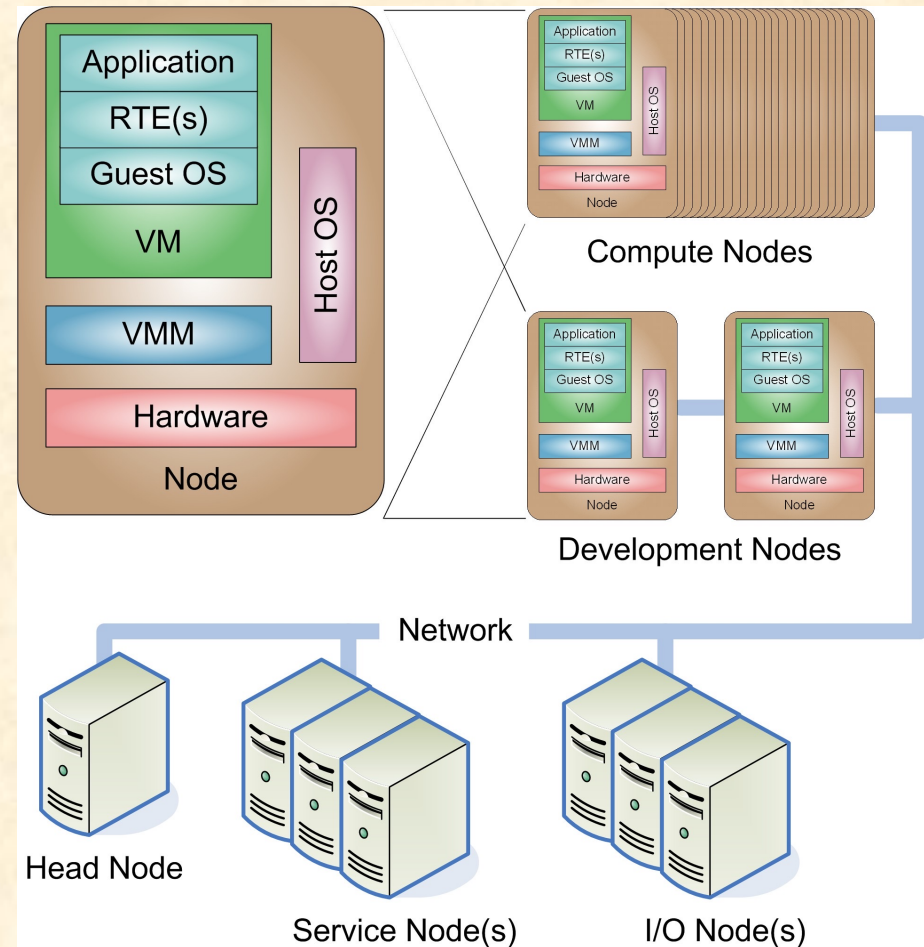
Virtualized System Environment (VSE)

- Hypervisors can provide a configurable ‘sandbox’ environment for system software and scientific application development and deployment
- System-level virtualization on development systems (desktops and small HPC systems) and production-type systems (large HPC systems) can provide:
 - Simplified application porting through virtualization
 - On-demand OS deployment on virtualized HPC systems
 - On-demand deployment of virtual testbeds isolated from the real systems and from each other via a hypervisor

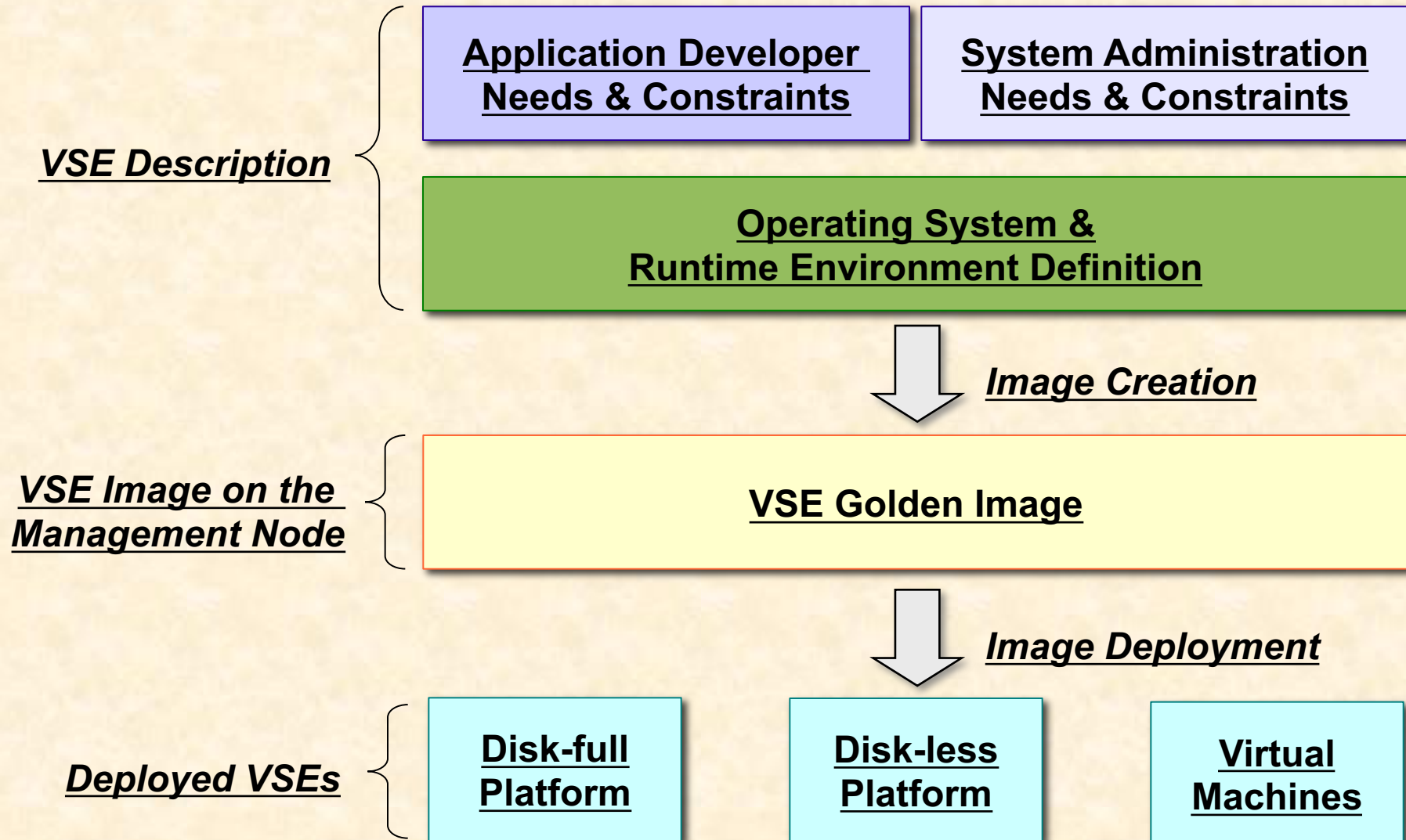


Virtual System Environment (VSE): System Architecture

- Hypervisor on development and compute nodes
- Virtual machines run the customized virtualized environment
- Customization is based on:
 - Application needs
 - System capabilities
 - Resource allocation

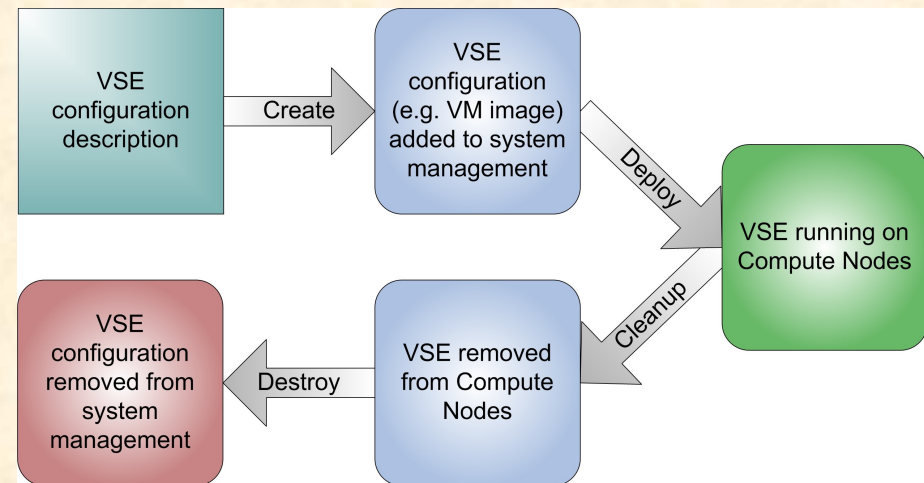


Virtual System Environment (VSE): Management



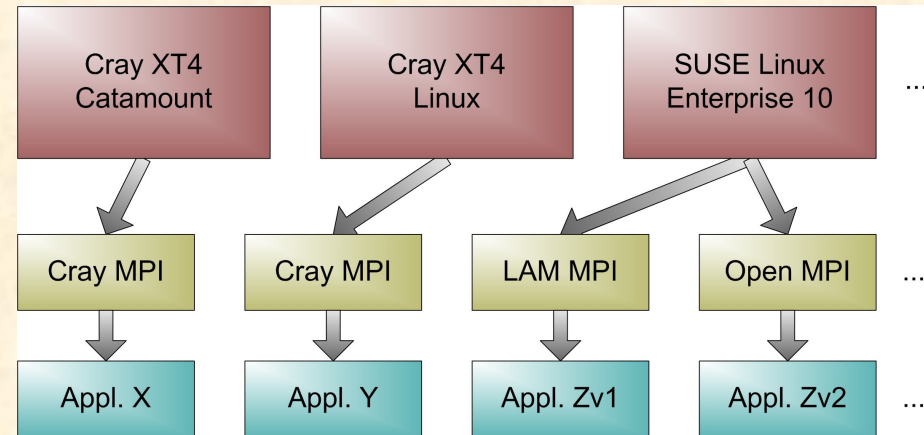
Virtual System Environment (VSE): Life Cycle

- **System management tools for VSE configuration:**
 - Description
 - Creation
 - Deployment
 - Cleanup
 - Destruction
- **Adaptation of existing VM management tools to HPC system resource management and software development tools**



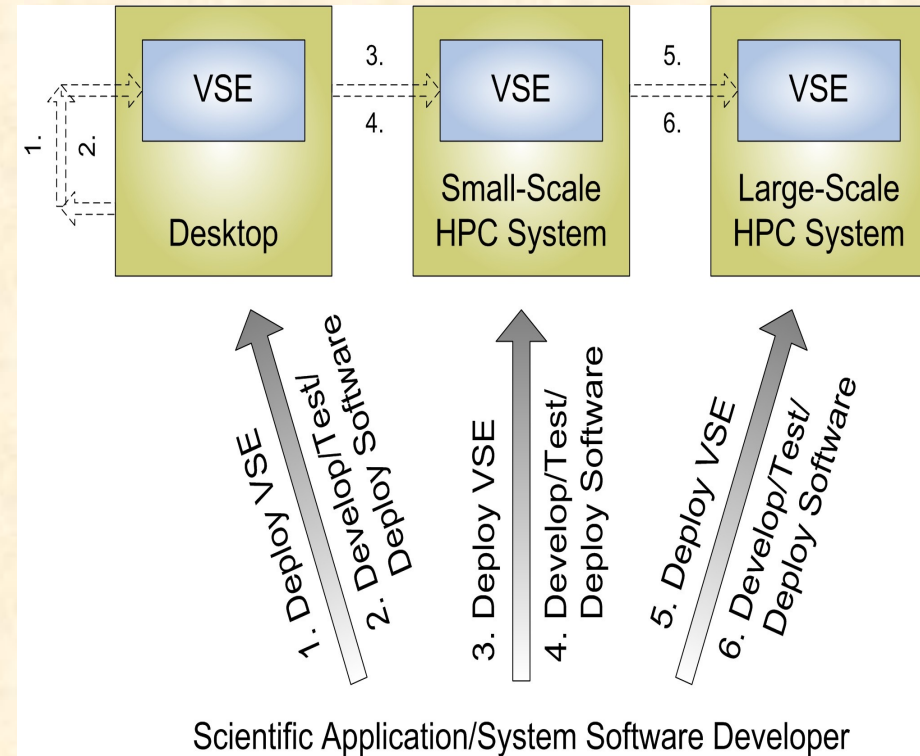
Virtual System Environment (VSE): Configuration Management

- Hierarchical configuration scheme enables users to:
 - Override
 - Remove
 - Addconfiguration options.
- Vendor and/or system operator configuration descriptions can be used as base configuration



Virtual System Environment (VSE): Use Case Scenarios

- Application and system software developers can deploy VSEs based on their actual needs to:
 - Desktops
 - Small-scale HPC systems
 - Large-scale HPC systemsfor software development and deployment activities.



A developer can work on his local desktop instead of logging into a remote HPC system development environment server!

OSCAR-V: System Management with Virtualization Support

- Extension of Open Source Cluster Application Resources (OSCAR) Linux cluster installation and management suite
- Includes system-level virtualization support:
 - Capability to switch between virtual and standard cluster computing environments
- Abstracts underlying virtualization solution:
 - Generic virtual machine management (V2M) layer
 - Capability to switch between different virtualization solution
- VSE configuration consists of a set of OSCAR packages
- Support for various Linux distributions: SUSE, RedHat, Debian, ...



Virtual Machine Management (V2M): Architecture

V2M
Virtual Machine Management
Command Line Interface

Applications
based on
libv3m

High-Level Interface
boot_vm, create_image_from_cdrom,
install_vm_with_oscar, migrate_vm,
pause_vm, unpause_vm

V3M
Front-end

Virtualization Abstraction

Qemu

Xen

VMWare

...

V3M
Back-ends

Ongoing Studies

- **Hypervisor for high-performance computing**
 - Low-profile virtual machine monitors (VMMs)
 - Modular VMMs for adaptation
 - Efficient I/O using VMM-bypass:
 - Isolation vs. performance
 - RDMA support
 - Optimizations for modern hardware features, such as IOMMU, Intel-VT, and AMD-V
- **Tiny domains**
 - Decrease the size of the host OS and VMs
 - Minimize overall system footprint

Proactive Fault Tolerance Using Preemptive Migration

Christian Engelmann

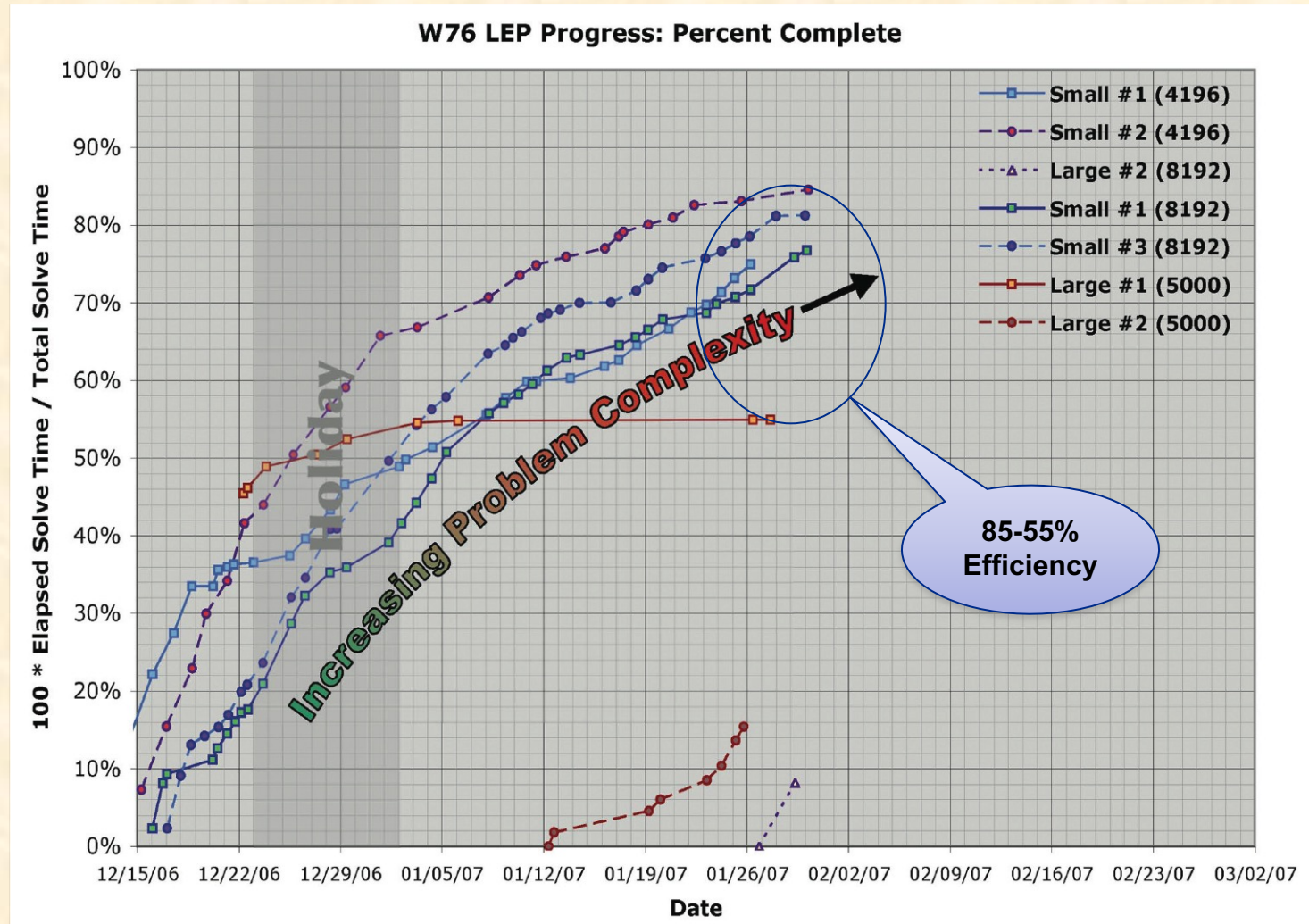
**Computer Science and Mathematics Division
Oak Ridge National Laboratory**

Motivation

- **Large-scale PFlop/s systems have arrived**
 - **#1 ORNL Jaguar XT5: 1.759 PFlop/s LINPACK, 224,162 cores**
 - **#2 LANL Roadrunner: 1.042 PFlop/s LINPACK, 122,400 cores**
- **Other large-scale systems exist**
 - **#3 NICS Kraken XT5: 0.831 PFlop/s LINPACK, 98,928 cores**
 - **#4 Juelich JUGENE: 0.825 PFlop/s LINPACK, 294,912 cores**
- **The trend is toward larger-scale systems**
 - **Exascale (1,000 PFlop/s) system with 100M-1B cores by 2018**
- **Significant increase in component count and complexity**
- **Expected matching increase in failure frequency**
- **Checkpoint/restart is becoming less and less efficient**

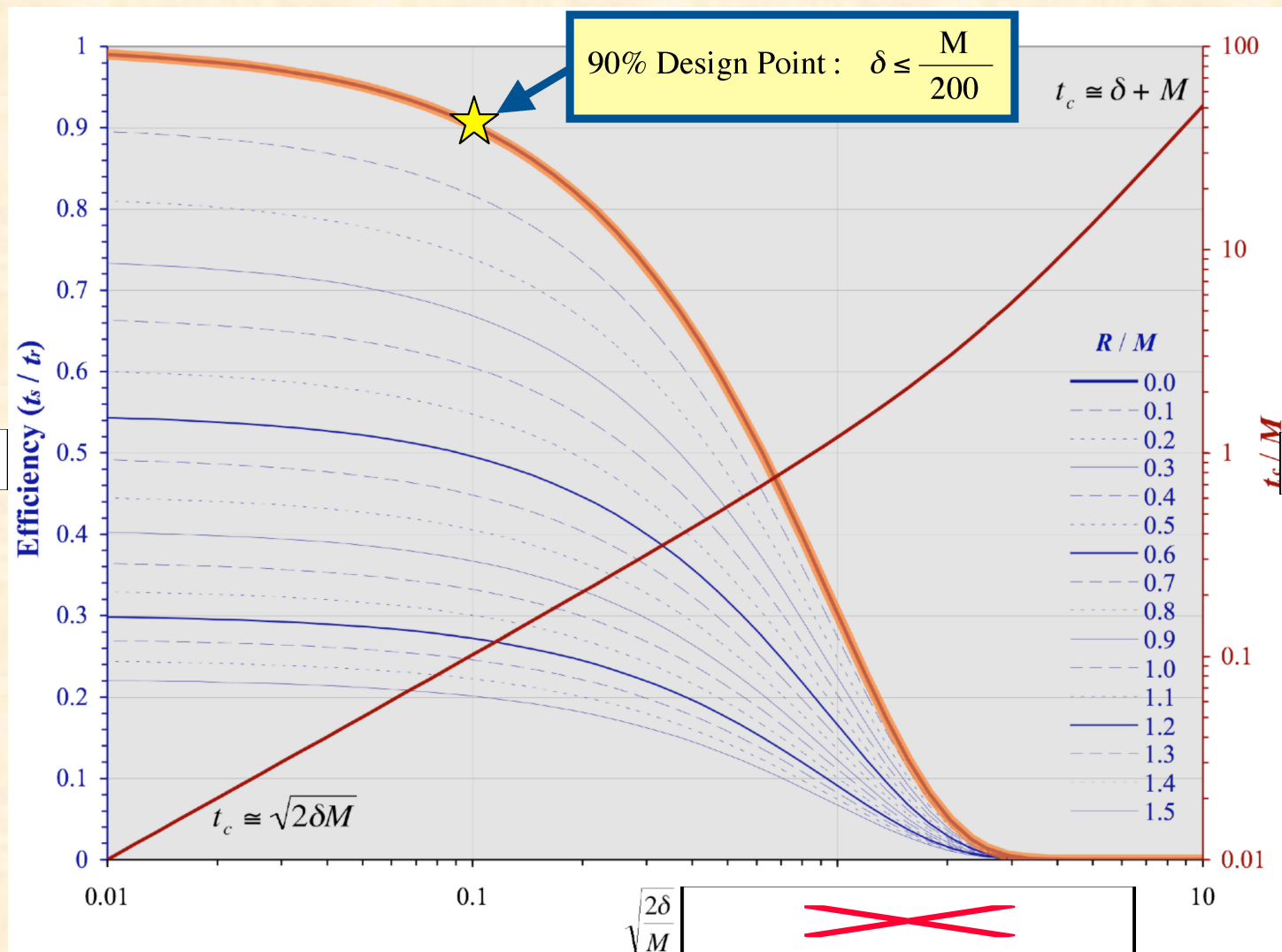
Parallel File System Checkpoint/Restart Efficiency Study (2006 @ LANL)

J. T. Daly. ADTSC Nuclear Weapons Highlights: Facilitating High-Throughput ASC Calculations.
Technical Report LALP-07-041, Los Alamos National Laboratory, June 2007.



Parallel File System Checkpoint/Restart Efficiency Model

J. T. Daly. Methodology and metrics for quantifying application throughput. In Proceedings of the Nuclear Explosives Code Developers Conference (NECDC) 2006, Los Alamos, NM, USA, Oct. 23-27, 2006.



Reactive vs. Proactive Fault Tolerance

- **Reactive fault tolerance**
 - Keeps parallel applications alive through recovery from experienced failures
 - Employed mechanisms react to failures
 - Examples: Checkpoint/restart, message logging/replay
- **Proactive fault tolerance**
 - Keeps parallel applications alive by avoiding failures through preventative measures
 - Employed mechanisms anticipate failures
 - Example: Preemptive migration

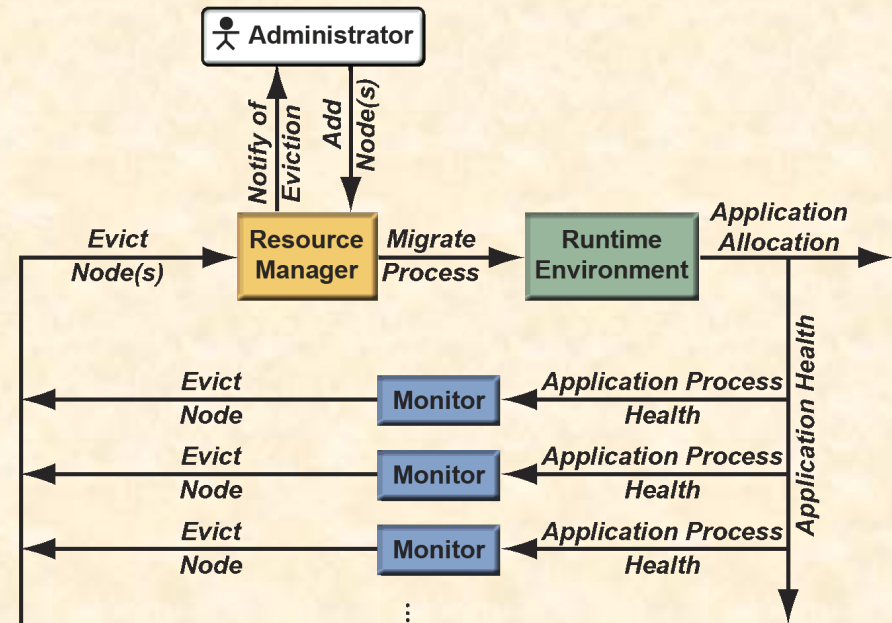
Proactive Fault Tolerance using Preemptive Migration

- Relies on a feedback-loop control mechanism
 - Application health is constantly monitored and analyzed
 - Application is reallocated to improve its health and avoid failures
 - Closed-loop control similar to dynamic load balancing
- Real-time control problem
 - Need to act in time to avoid imminent failures
- No 100% coverage
 - Not all failures can be anticipated, such as random bit flips



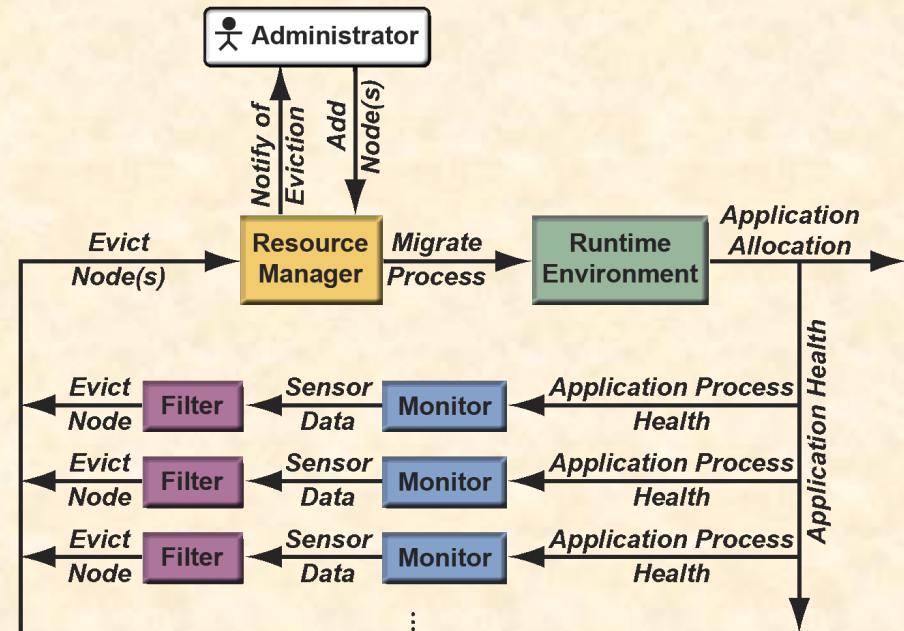
Type 1 Feedback-Loop Control Architecture

- Alert-driven coverage
 - Basic failures
- No evaluation of application health history or context
 - Prone to false positives
 - Prone to false negatives
 - Prone to miss real-time window
 - Prone to decrease application health through migration
 - No correlation of health context or history



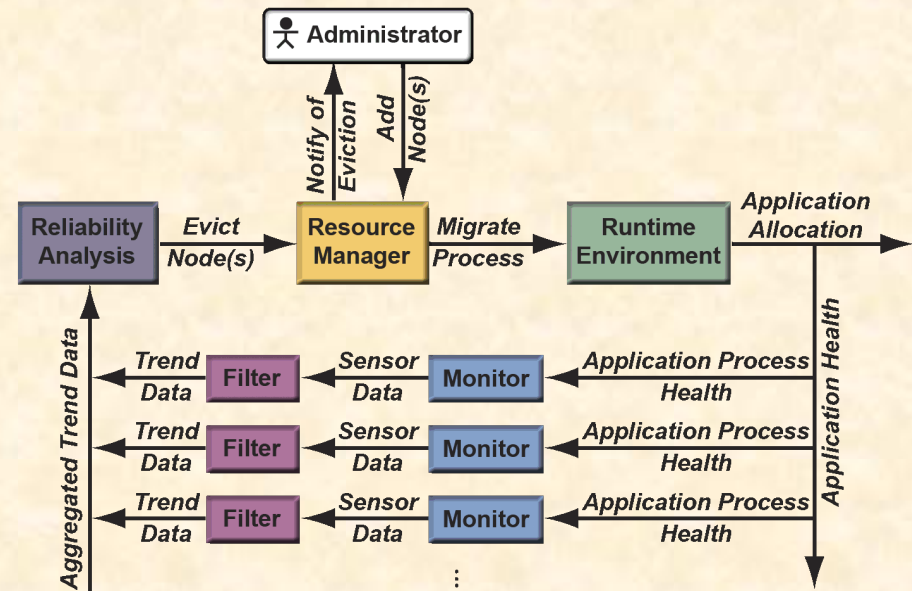
Type 2 Feedback-Loop Control Architecture

- Trend-driven coverage
 - Basic failures
 - Less false positives/negatives
- No evaluation of application reliability
 - Prone to miss real-time window
 - Prone to decrease application health through migration
 - No correlation of health context or history



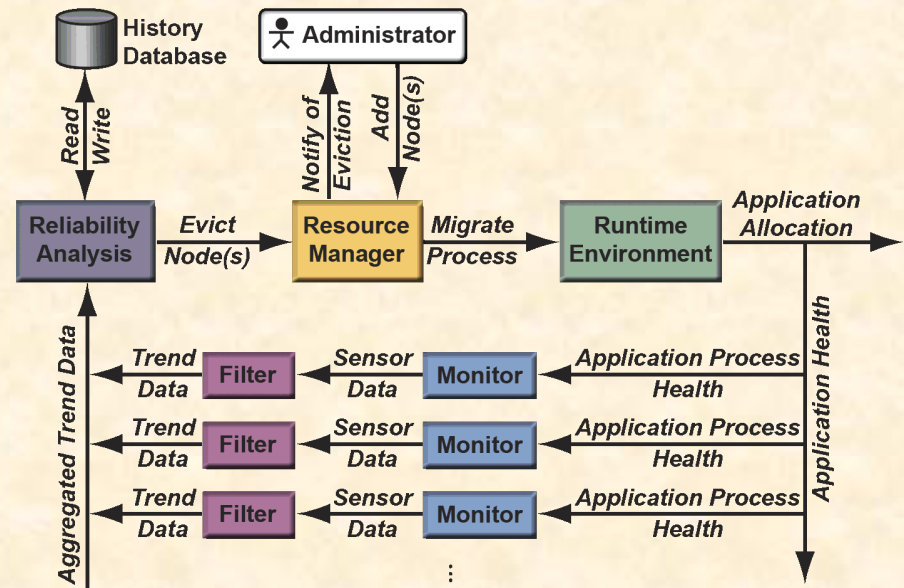
Type 3 Feedback-Loop Control Architecture

- Reliability-driven coverage
 - Basic and correlated failures
 - Less false positives/negatives
 - Able to maintain real-time window
 - Does not decrease application health through migration
 - Correlation of short-term health context and history
- No correlation of long-term health context or history
 - Unable to match system and application reliability patterns



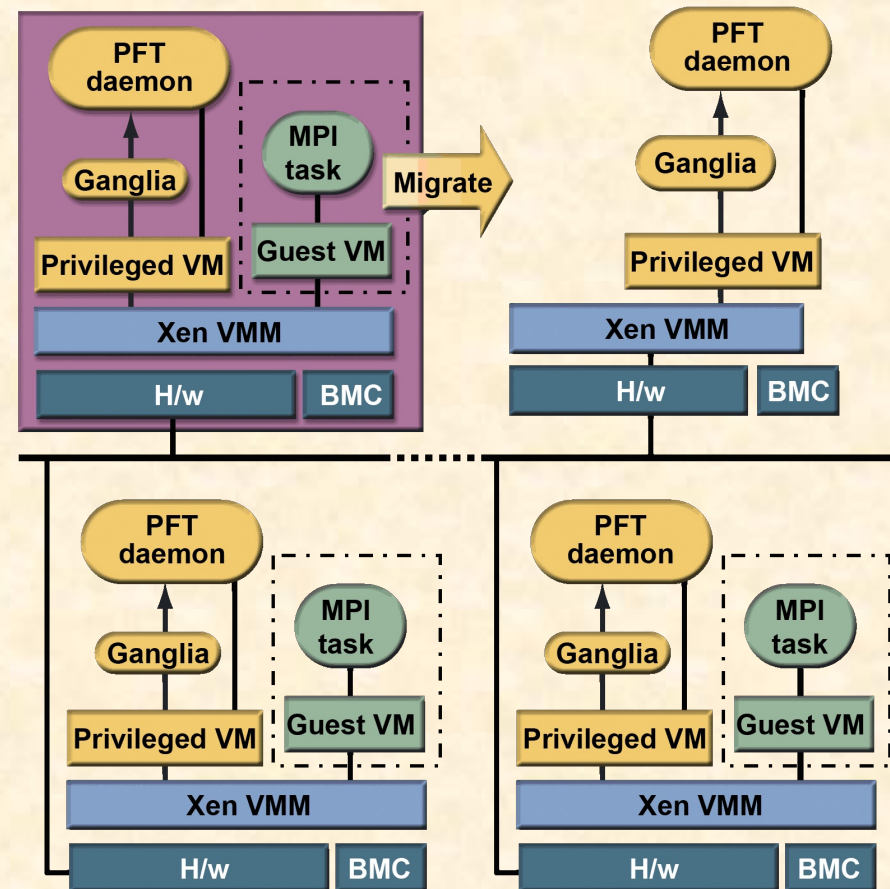
Type 4 Feedback-Loop Control Architecture

- Reliability-driven coverage of failures and anomalies
 - Basic and correlated failures, anomaly detection
 - Less prone to false positives
 - Less prone to false negatives
 - Able to maintain real-time window
 - Does not decrease application health through migration
 - Correlation of short and long-term health context & history



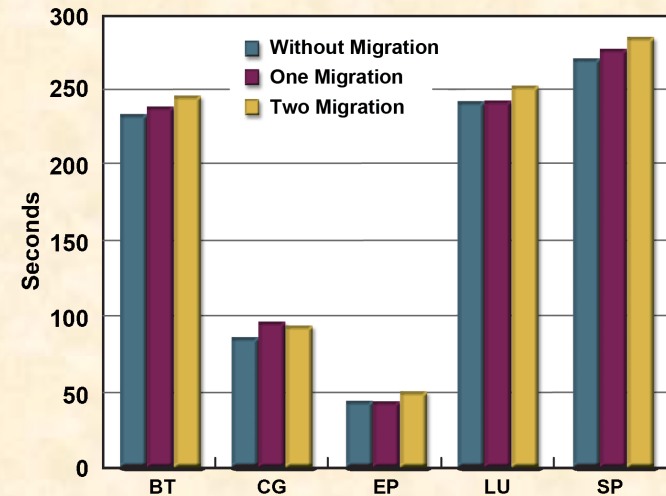
VM-level Preemptive Migration using Xen

- Type 1 system setup
 - Xen VMM on entire system
 - Host OS for management
 - Guest OS for computation
 - Spare nodes without Guest OS
 - System monitoring in Host OS
 - Decentralized scheduler/load balancer using Ganglia
- Deteriorating node health
 - Ganglia threshold trigger
 - Migrate guest OS to spare
 - Utilize Xen's migration facility



VM-level Migration Performance Impact

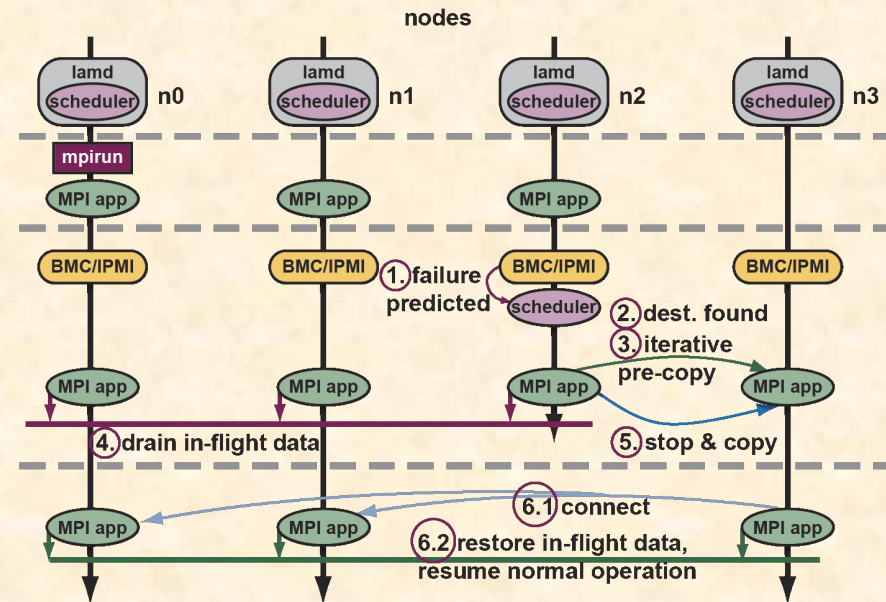
- **Single node migration**
 - 0.5-5% longer run time
- **Double node migration**
 - 2-8% longer run time
- **Migration duration**
 - Stop & copy : 13-14s
 - Live : 14-24s
- **Application downtime**
 - Stop & copy > Live



16-node Linux cluster at NCSU with dual core,
dual-processor AMD Opteron and Gigabit Ethernet

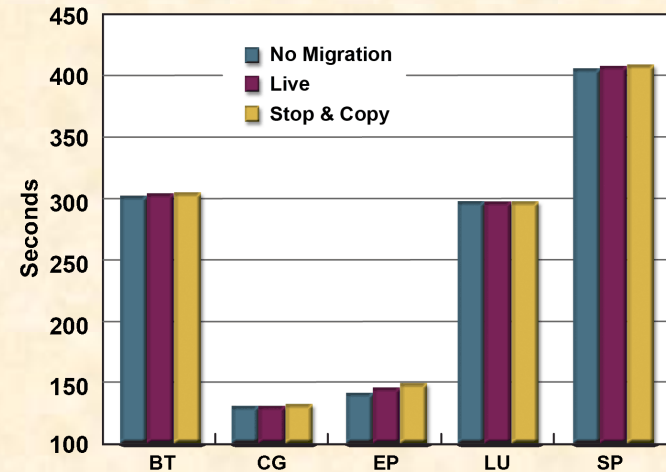
Process-Level Preemptive Migration w/ BLCR

- Type 1 system setup
 - LAM/MPI with Berkeley Lab Checkpoint/Restart (BLCR)
 - Per-node health monitoring
 - Baseboard management controller (BMC)
 - Intelligent platform management interface (IPMI)
 - New decentralized scheduler/load balancer in LAM
 - New process migration facility in BLCR (stop© and live)
- Deteriorating node health
 - Simple threshold trigger
 - Migrate process to spare



Process-Level Migration Performance Impact

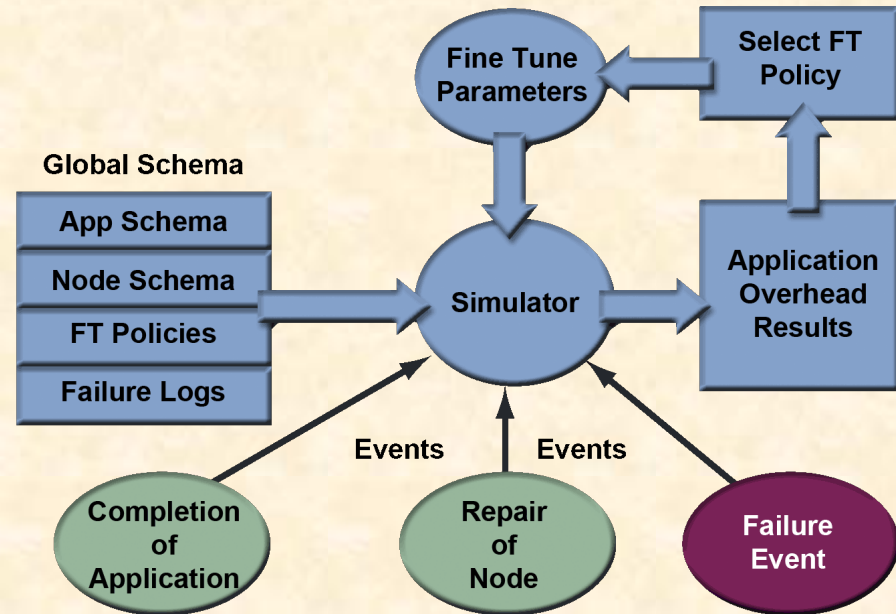
- **Single node migration overhead**
 - Stop & copy : 0.09-6 %
 - Live : 0.08-2.98%
- **Single node migration duration**
 - Stop & copy : 1.0-1.9s
 - Live : 2.6-6.5s
- **Application downtime**
 - Stop & copy > Live
- **Node eviction time**
 - Stop & copy < Live



16-node Linux cluster at NCSU with dual core, dual-processor AMD Opteron and Gigabit Ethernet

Simulation of Fault Tolerance Policies

- Evaluation of fault tolerance policies
 - Reactive only
 - Proactive only
 - Reactive/proactive combination
- Evaluation of fault tolerance parameters
 - Checkpoint interval
 - Prediction accuracy
- Event-based simulation framework using actual HPC system logs
- Customizable simulated environment
 - Number of active and spare nodes
 - Checkpoint and migration overheads



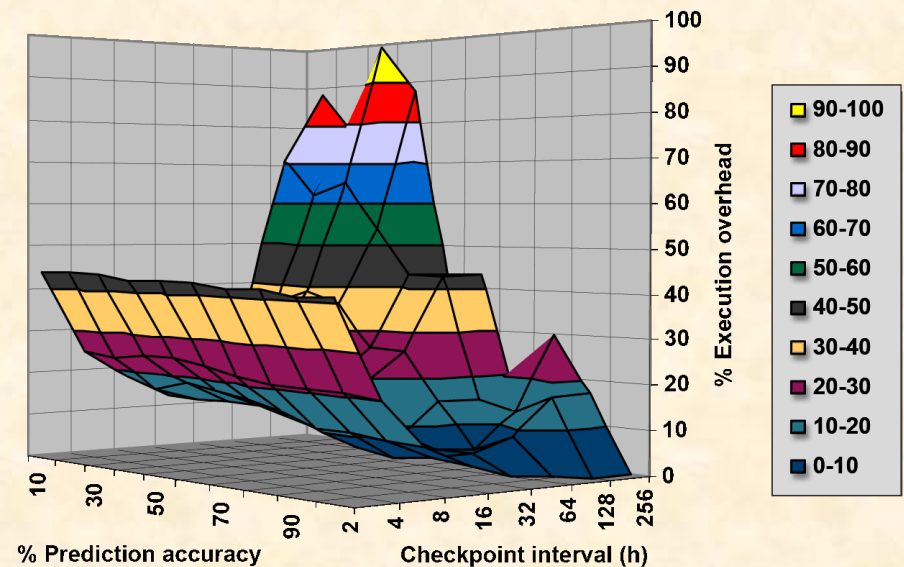
Combining Proactive & Reactive Approaches

- **Best: Prediction accuracy >60% and checkpoint interval 16-32h**
- **Better than only proactive or only reactive**
- **Results for higher accuracies and very low intervals are worse than only proactive or only reactive**

Number of processes	125
Active/Spare nodes	125/12
Checkpoint overhead	50min
Migration overhead	1 min

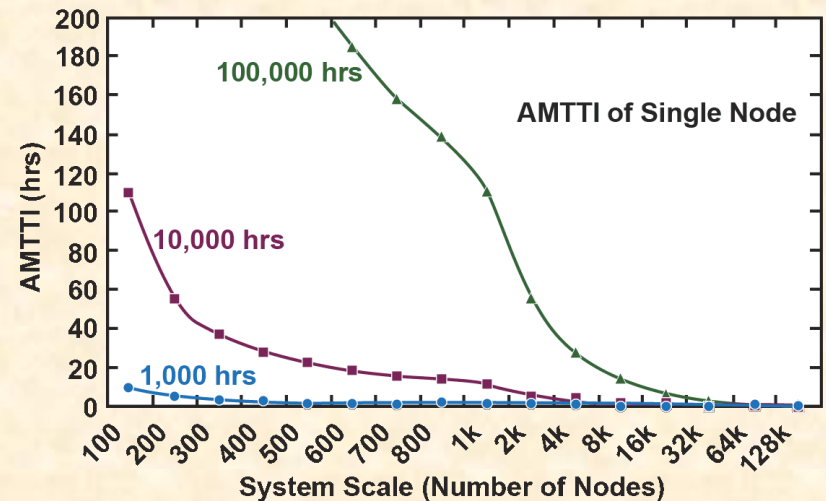
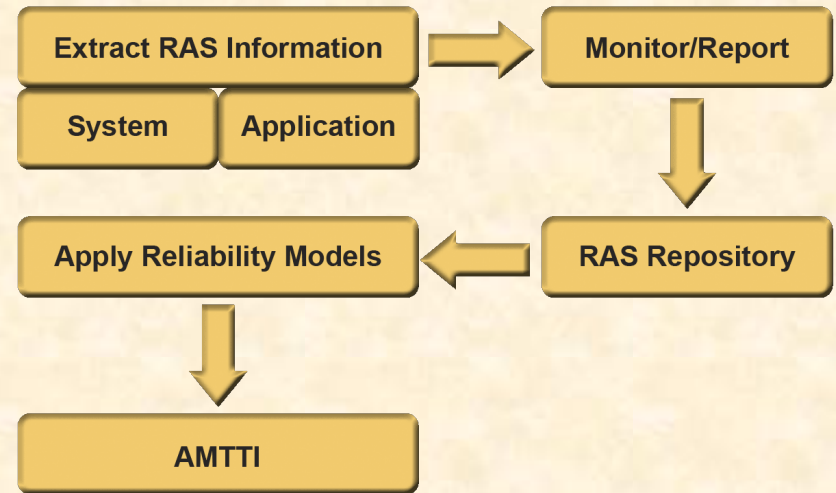
Simulation based on ASCI White system logs
(nodes 1-125 and 500-512)

Execution overhead for various checkpoint intervals and different prediction accuracy



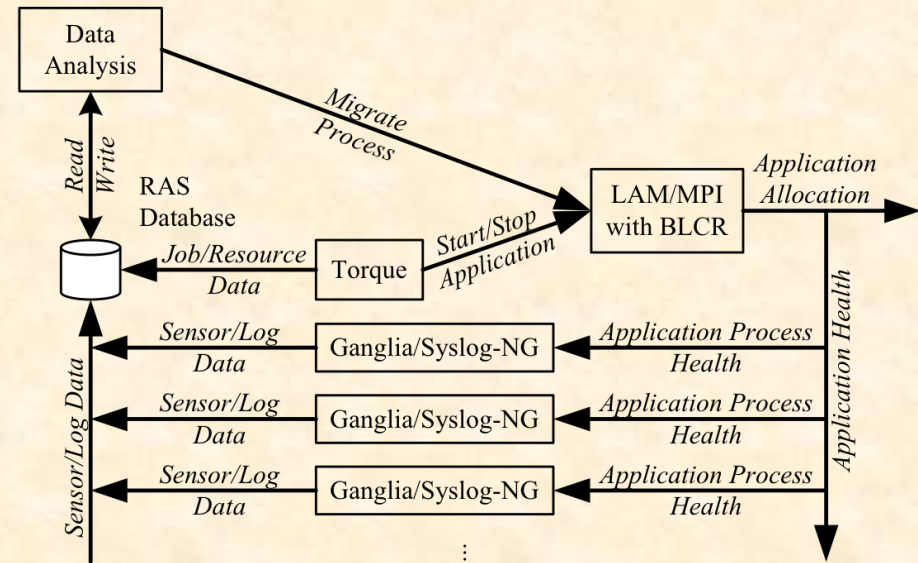
Research in Reliability Modeling

- **Type 3 system setup**
 - Monitoring of application and system health
 - Recording of application and system health monitoring data
 - Reliability analysis on recorded data
 - Application mean-time to interrupt (AMTTI) estimation
- **Type 4 system setup**
 - Additional recording of application interrupts
 - Reliability analysis on recent and historical data



Implementation of Type 1-4 RAS Framework

- Focus on proactive FT approach
- Central MySQL database for data logging and analysis
- Environmental monitoring
 - OpenIPMI and Ganglia
- Event logging and analysis
 - Syslog (node-local logging and forwarding to central server)
- Job and resource monitoring
 - Torque (epilogue/prologue)
- Migration mechanisms
 - Process-level using BLCR



Type 1-4 RAS Framework: Initial Results

- Deployed on XTORC @ ORNL
 - 64-node Intel-based Linux cluster
- MySQL, Gangila, Torque, Syslog, LAM-MPI+BLCR with migration
- Experiment #1:
 - Fully deployed on 64 nodes
 - 30 second data collection interval
 - Collection of 20 metrics resulted in over **20GB of data in 27 days** (~33MB/hour or ~275kb/interval)
 - Basic temperature threshold triggers for migration resulted in migration when covering up air intake holes
- Experiment #2:
 - Fully deployed on 32 nodes
 - Collection of 40 metrics
 - 30 second data collection interval
 - **No measurable impact** on NAS benchmarks (see Figure below)

Class C NPB on 32 nodes	CG	FT	LU
Average time in seconds	264	235	261
Average time under load in seconds	264	236	260

Table 2. NPB test results (averages over 10 runs)

Ongoing and Future Work

- **Research in scalable monitoring data aggregation/filtering**
 - Scalable, fault tolerant overlay reduction networks
 - In-flight monitoring data aggregation
 - Current MSc student (Sven Boehm)
- **Research in scalable monitoring data filtering**
 - Extend the current prototype with in-flight data filtering
 - Enhance filters with statistical analysis techniques
- **Research in scalable syslog data aggregation/filtering**
 - Extend the current prototype with log message aggregation
- **Integrate scalable monitoring prototype with proactive fault tolerance framework**

Challenges Ahead

- **Health monitoring**
 - Identifying deteriorating applications and OS conditions
 - Coverage of application failures: Bugs, resource exhaustion
- **Reliability analysis**
 - Performability analysis to provide extended coverage
- **Scalable data aggregation and processing**
 - Key to timeliness in the feedback control loop
- **Need for standardized metrics and interfaces**
 - System MTTF/MTTR **!=** Application MTTF/MTTR
 - System availability **!=** Application efficiency
 - Monitoring and logging is system/vendor dependent

Acknowledgements

- Investigators at Oak Ridge National Laboratory:
 - *Stephen L. Scott [Lead PI]*, Christian Engelmann, Geoffroy Vallée, Thomas Naughton, Anand Tikotekar, George Ostrouchov
- Investigators at Louisiana Tech University:
 - *Chokchai (Box) Leangsuksun [Lead PI]*, Nichamon Naksinehaboon, Raja Nassar, Mihaela Paun
- Investigators at North Carolina State University:
 - *Frank Mueller [Lead PI]*, Chao Wang, Arun Nagarajan, Jyothish Varma
- Funding sources:
 - U.S. Department of Energy, Office of Science, FASTOS Program



Institute for advanced Architectures and Algorithms (IAA): Simulation Efforts at ORNL

Christian Engelmann

**Computer Science and Mathematics Division
Oak Ridge National Laboratory**

Objectives

- **Simulation of system architectures at scale**
- **To investigate scalability, performance, and fault tolerance of algorithms at extreme scale**
- **ORNL's earlier work was already able to run up to 1,000,000 simulated processes (JCAS)**

Java Cellular Architecture Simulator (JCAS)

- **Developed at ORNL in Java**
- **Native C and Fortran application support using JNI**
- **Runs as standalone or distributed application**
- **Lightweight framework simulates up to 1,000,000 lightweight virtual processes on 9 real processors**
- **Standard and experimental network interconnects:**
 - Multi-dimensional mesh/torus
 - Nearest/Random neighbors
- **Message driven simulation without notion of time**
 - Not in real-time, no time-accurate discrete event simulation
- **Primitive fault-tolerant MPI support**
 - No collectives, no MPI 2

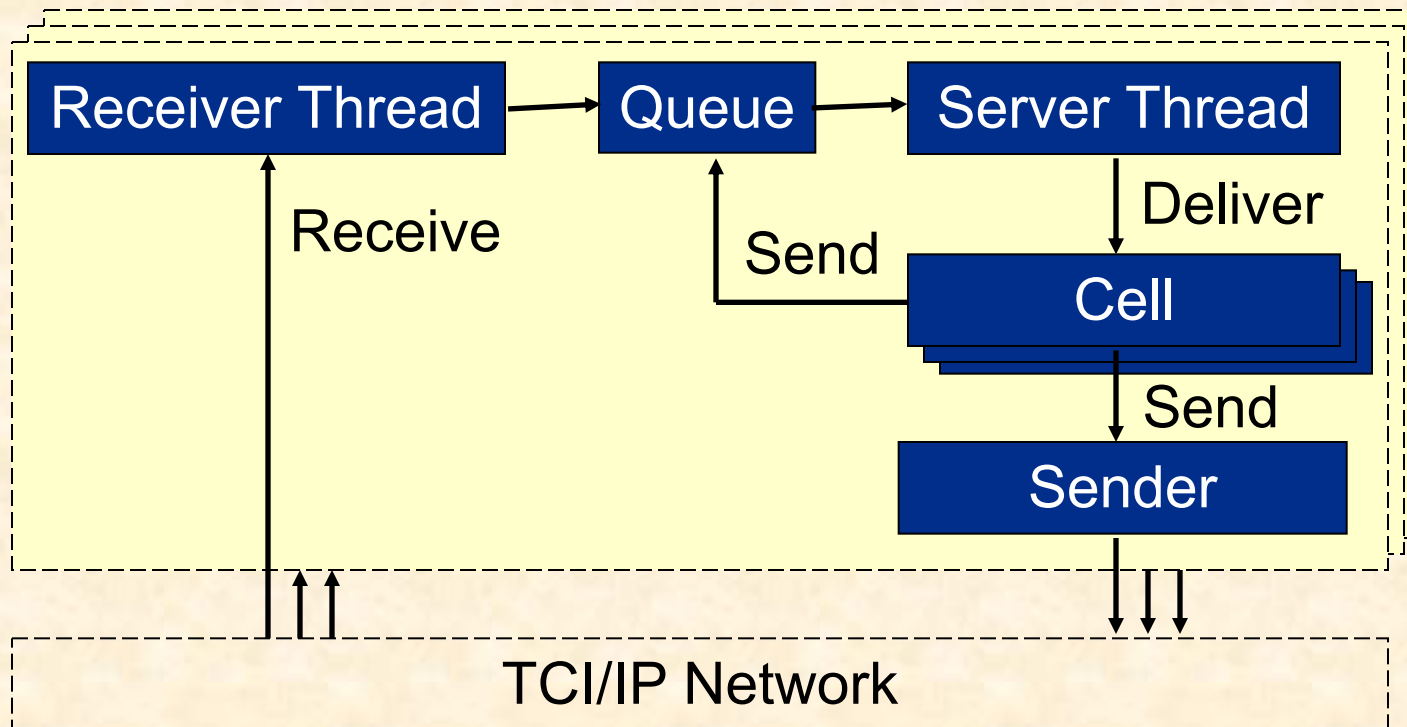
Technical Approach

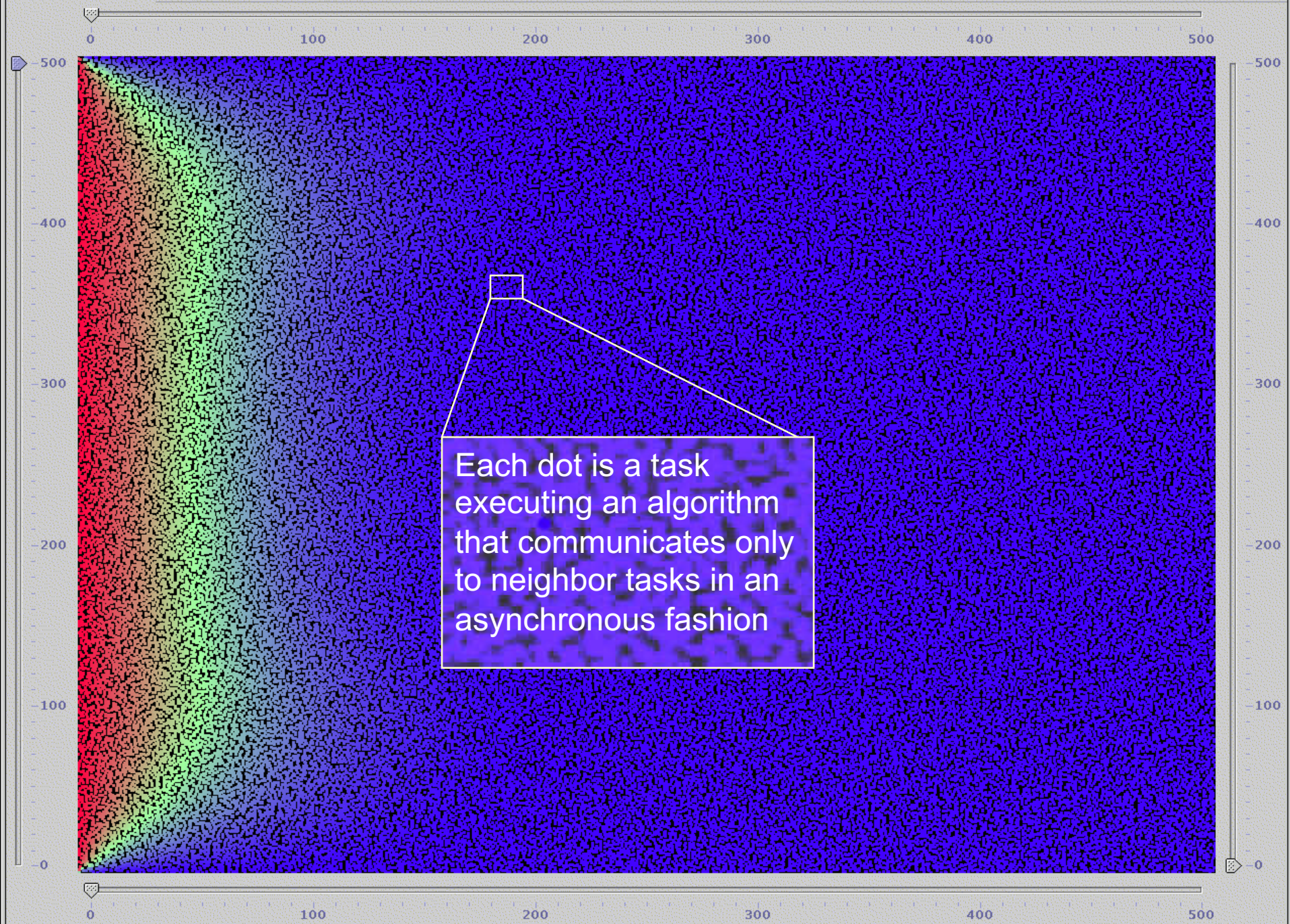
- Distributed set of discrete event simulators with node-local message queues
- Simulation of virtual MPI processes for parallel app.
- Virtual processes run on real hardware with virtual MPI
- No virtual process time
- Fault injection capability
- Interactive graphical user interface as front-end
- TCP servers as back-ends



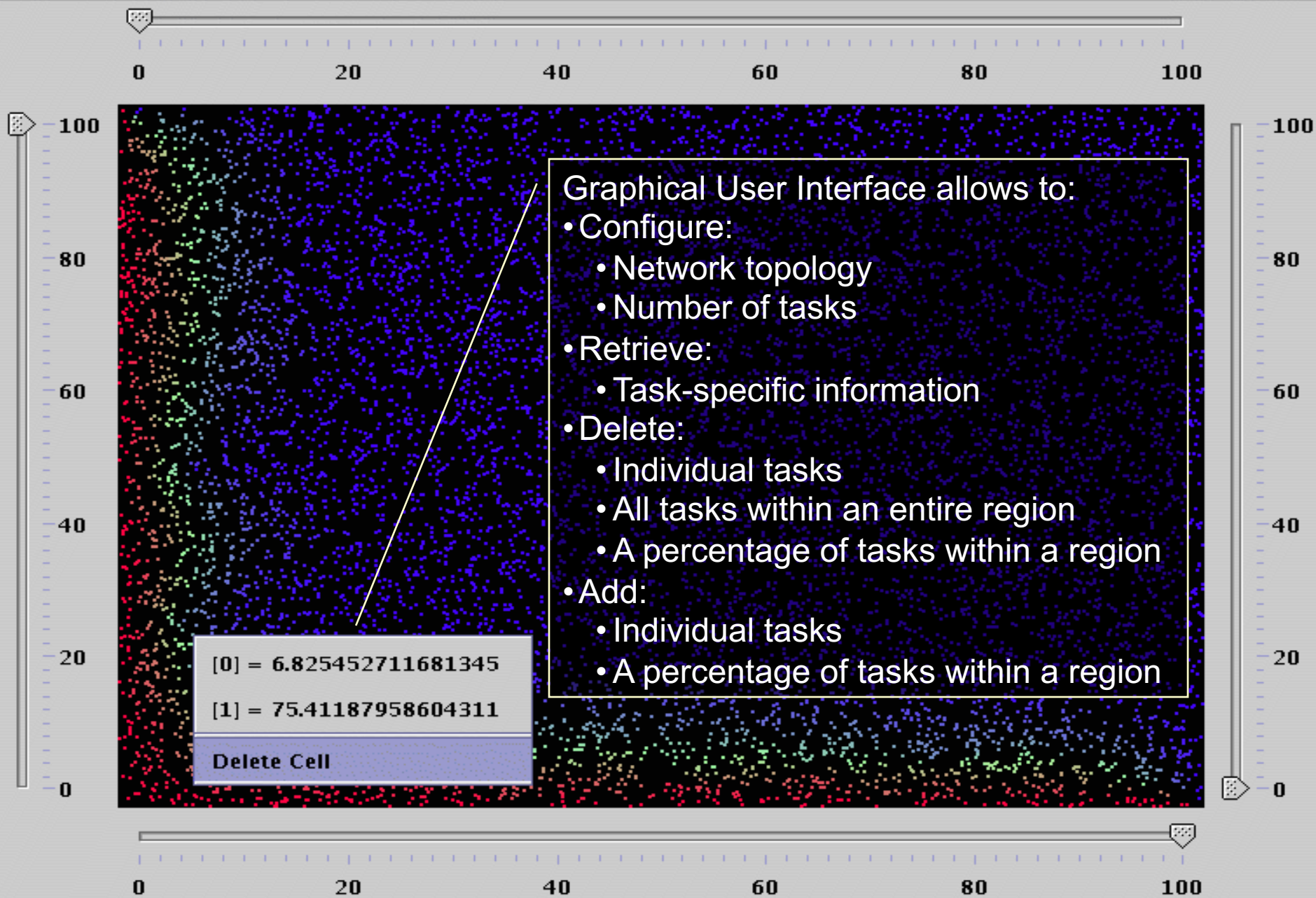
Implementation

- Every cell has own code, memory and neighbors list
- Server hosts cells and initiates the context switch
- Cells communicate asynchronously using messages





System Laplace (Java) Help



Graphical User Interface allows to:

- Configure:
 - Network topology
 - Number of tasks
- Retrieve:
 - Task-specific information
- Delete:
 - Individual tasks
 - All tasks within an entire region
 - A percentage of tasks within a region
- Add:
 - Individual tasks
 - A percentage of tasks within a region

[0] = 6.825452711681345

[1] = 75.41187958604311

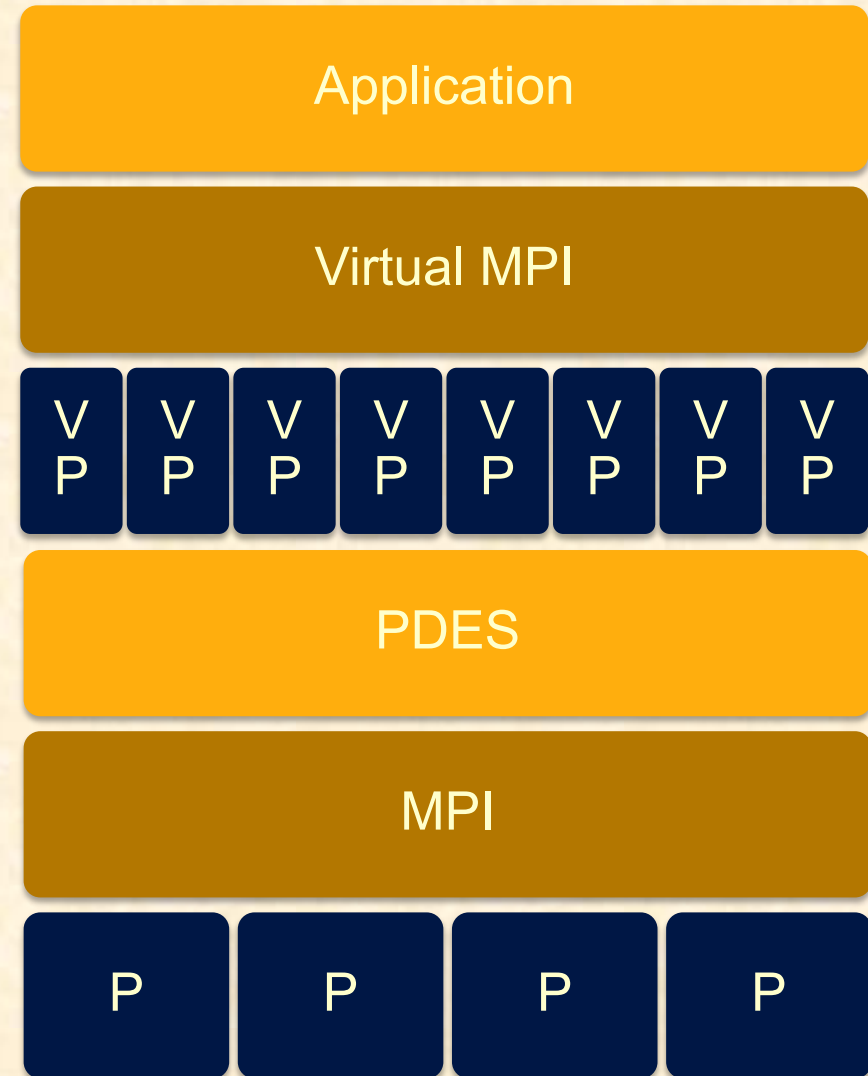
Delete Cell

IAA Simulation Efforts at ORNL

- **Investigate scalability, performance and fault tolerance of algorithms at extreme scale through simulation**
- **Extending the JCAS simulation capabilities**
 - **Simulating more processes (~10,000,000)**
 - **Running more complex and resource-hungry algorithms**
 - **Support for unmodified MPI applications**
- **Evaluation of algorithms at extreme scale**
 - **Notion of global virtual time and virtual process clocks**
 - **Accounting for resource usage, such as processor and network**
 - **Gathering of scalability, performance & fault tolerance metrics**
 - **Parameter studies at scale**

Technical Approach

- Parallel discrete event simulation (PDES) atop MPI
- Simulation of virtual MPI processes for parallel app.
- Virtual processes run on real hardware with virtual MPI
- Consistent virtual process clock from PDES
- Virtual process clock can be scaled by PDES via model
- Virtual interconnect latency is set by PDES via model



Ongoing and Future Work

- **Ported JCAS to C/C++ to improve scalability/performance**
- **Replaced TCP/IP with (native) MPI communication**
- **Replaced distributed set of DESs with PDES**
 - **Conservative synchronization only, need optimistic and time-warp synchronization**
- **Extend virtual MPI capabilities**
 - **Asynchronous, collectives, process control (spawn), ...**
- **Extend fault injection and notification mechanisms**
 - **Injection based on failure distributions and application state**
- **Add simulated machine model (for network)**
- **Gather scalability, performance & fault tolerance metrics**

Soft-Error Resilience for Future-Generation High-Performance Computing Systems

Christian Engelmann

**Computer Science and Mathematics Division
Oak Ridge National Laboratory**

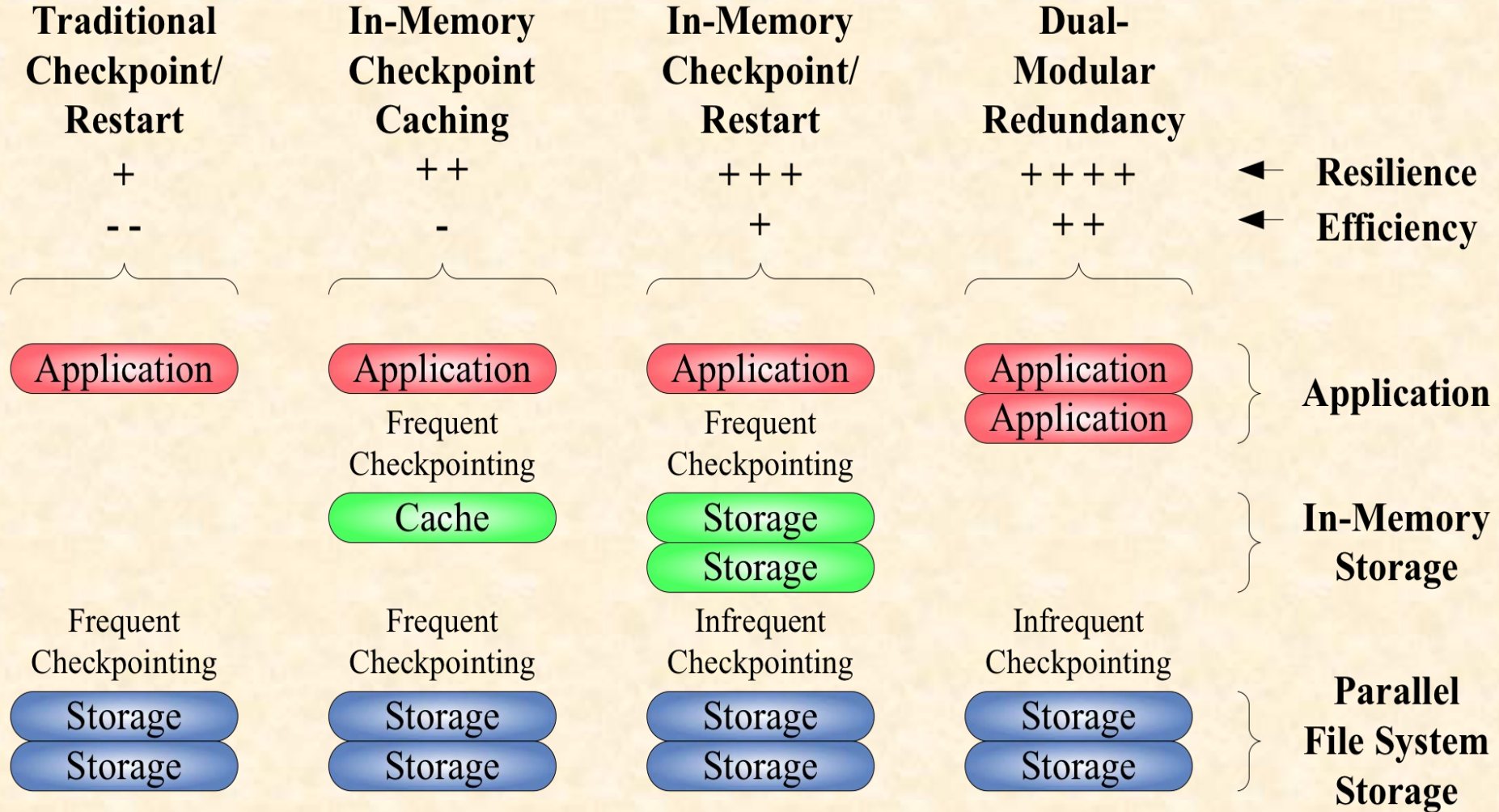
Motivation

- **Next-generation HPC systems will have**
 - More frequent failures in general
 - More frequent soft errors in particular
 - Less efficient parallel file system checkpoint/restart
- **Existing fault tolerance approaches and ongoing research efforts do not cover soft error resilience**
 - ECC double-bit errors require node/process restart
 - Silent data corruption remains undetected
- **Lack of soft error resilience strategy is preventing deployment of GPUs and FPGAs at scale**

Technical approach

- **Compute-node in-memory checkpoint caching**
 - Short-term solution
 - Improving parallel file system checkpoint/restart
- **Compute-node in-memory checkpoint/restart**
 - Near-term solution
 - Replacing parallel file system checkpoint/restart
- **Dual-modular redundancy (DMR)**
 - Long-term solution
 - Replacing rollback recovery schemes in HPC

Comparison of traditional and proposed technologies (1/2)



Improving System Availability with Modular Redundancy

- **Modular redundancy concepts have been around for a while**
 - E.g. aerospace and command & control systems
- **System availability is improved using redundant components**
- **Dual-modular redundancy (DMR) offers protection against hard errors and some soft errors**
- **Triple-modular redundancy (TMR) offers protection against hard and soft errors**
- **Dynamic dual- or triple-modular redundancy uses reboot or spare to reduce component MTTR**

$$A_{DMR} = 1 - (1 - A)^2$$

$$A_{TMR} = 1 - (1 - A)^3$$

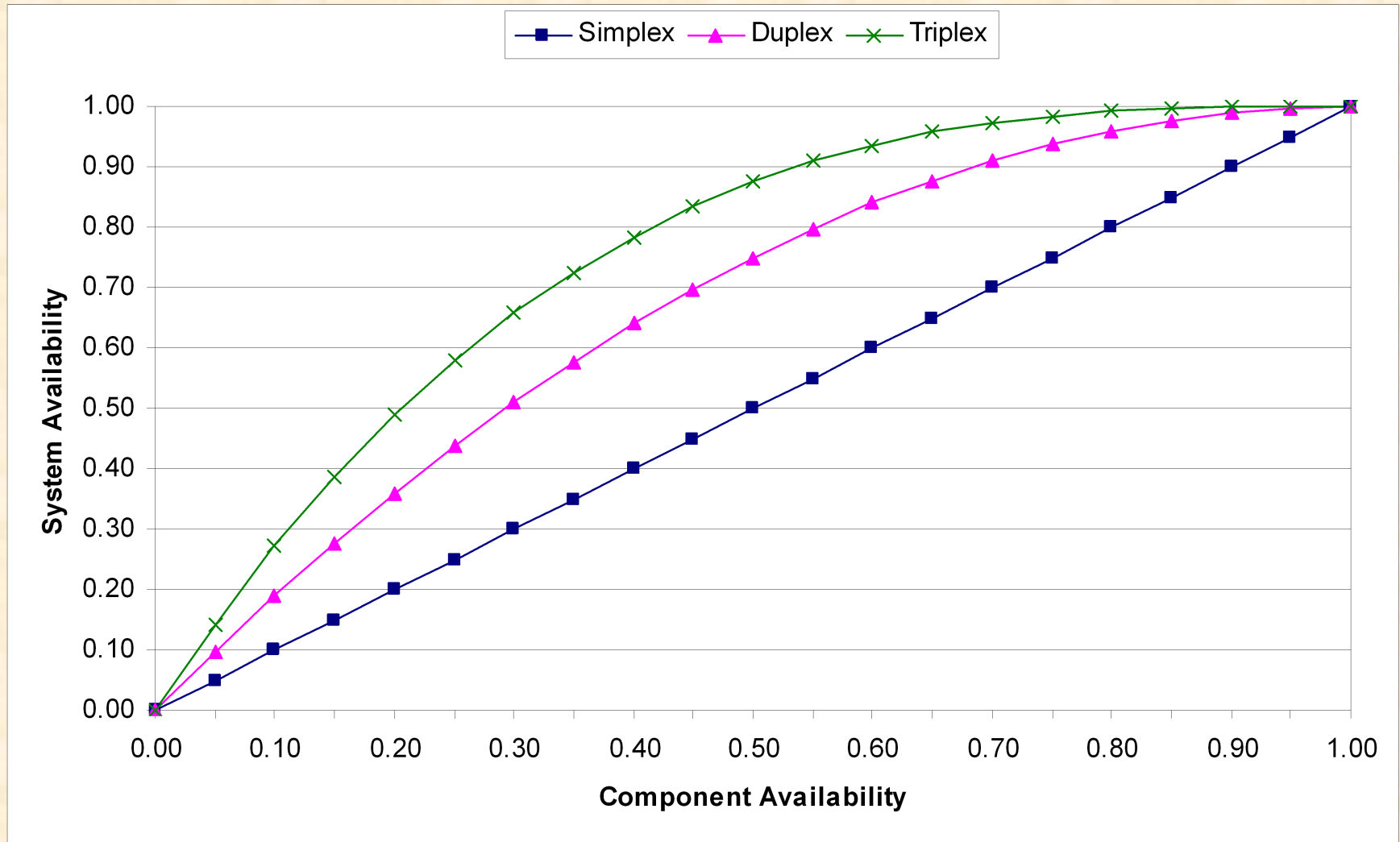
$$A_{DDMR} = 1 - (1 - A_1)(1 - A_2)$$

$$A_{DTMR} = 1 - (1 - A_1)(1 - A_2)^2$$

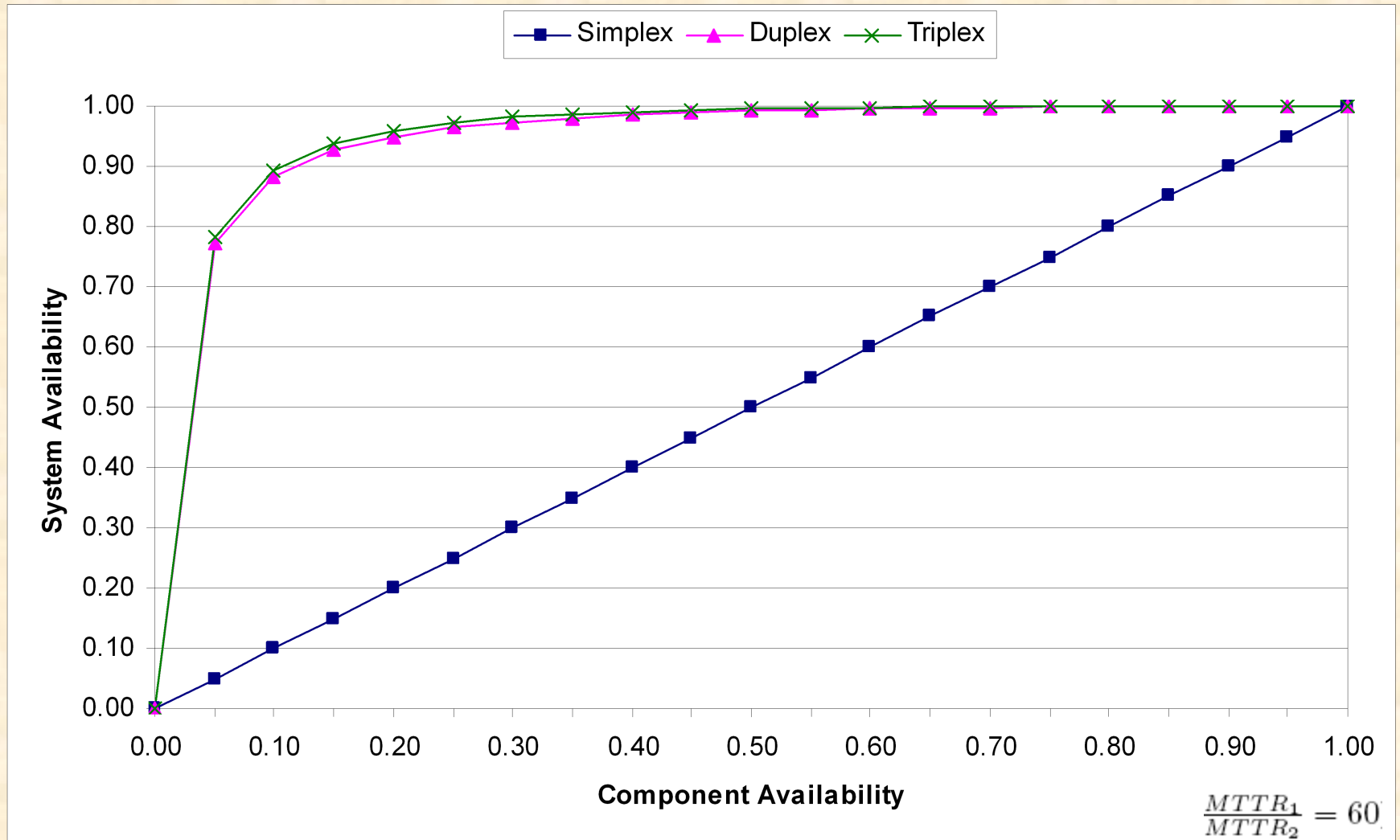
Improving Compute Node Availability with Modular Redundancy

- **Today's large-scale HPC systems have tens-to-hundreds of thousands of diskless compute nodes consisting of**
 - **processor(s), memory module(s) and a network interface**
- **Deploying modular redundancy for these systems would require to double or triple the number of compute nodes**
- **However, the network infrastructure is able to recover soft errors by retransmitting messages**
- **We only need to double or triple the number of processors and memory modules within each compute node**
- **A modular redundancy mechanism is needed for replication, error detection and error recovery in a massively parallel HPC system**

Compute Node Availability Improvement with Modular Redundancy



Compute Node Availability Improvement with Dynamic Modular Redundancy



Improving HPC System Availability with Compute-Node Modular Redundancy

- The availability of a modular redundant compute node is based on $2 \times / 3 \times$ parallel coupling
- The availability of a HPC system is based on $n \times$ serial coupling
- The availability of a compute-node modular redundant HPC system is based on $n \times$ serial of $2 \times / 3 \times$ parallel components
- Dynamic modular redundancy additionally reduces the MTTR of 1 (DMR) or 2 (TMR) components

$$A_{DMR} = [1 - (1 - A)^2]^n$$

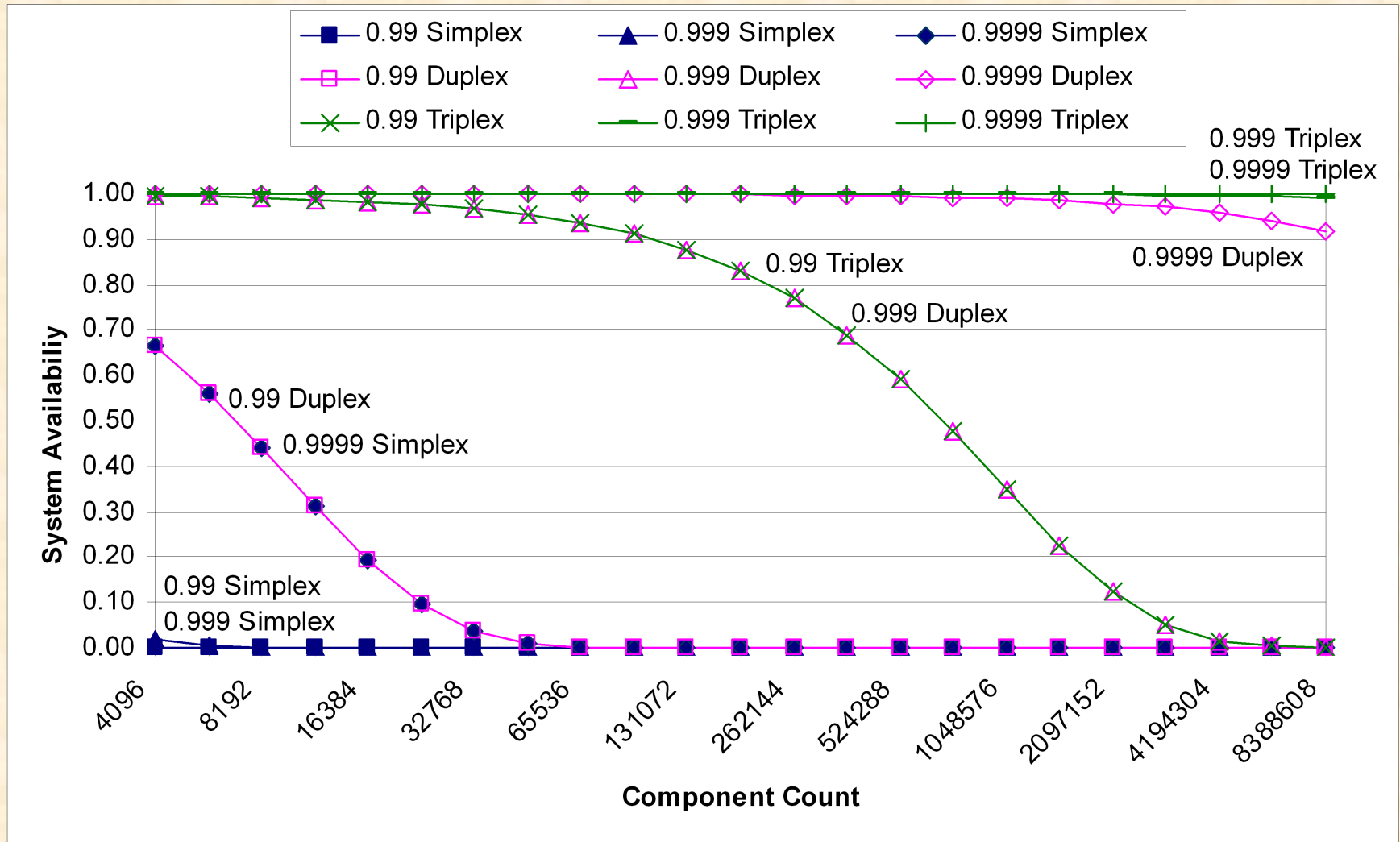
$$A_{TMR} = [1 - (1 - A)^3]^n$$

$$A_{DDMR} = [1 - (1 - A_1)(1 - A_2)]^n$$

$$A_{DTMR} = [1 - (1 - A_1)(1 - A_2)^2]^n$$

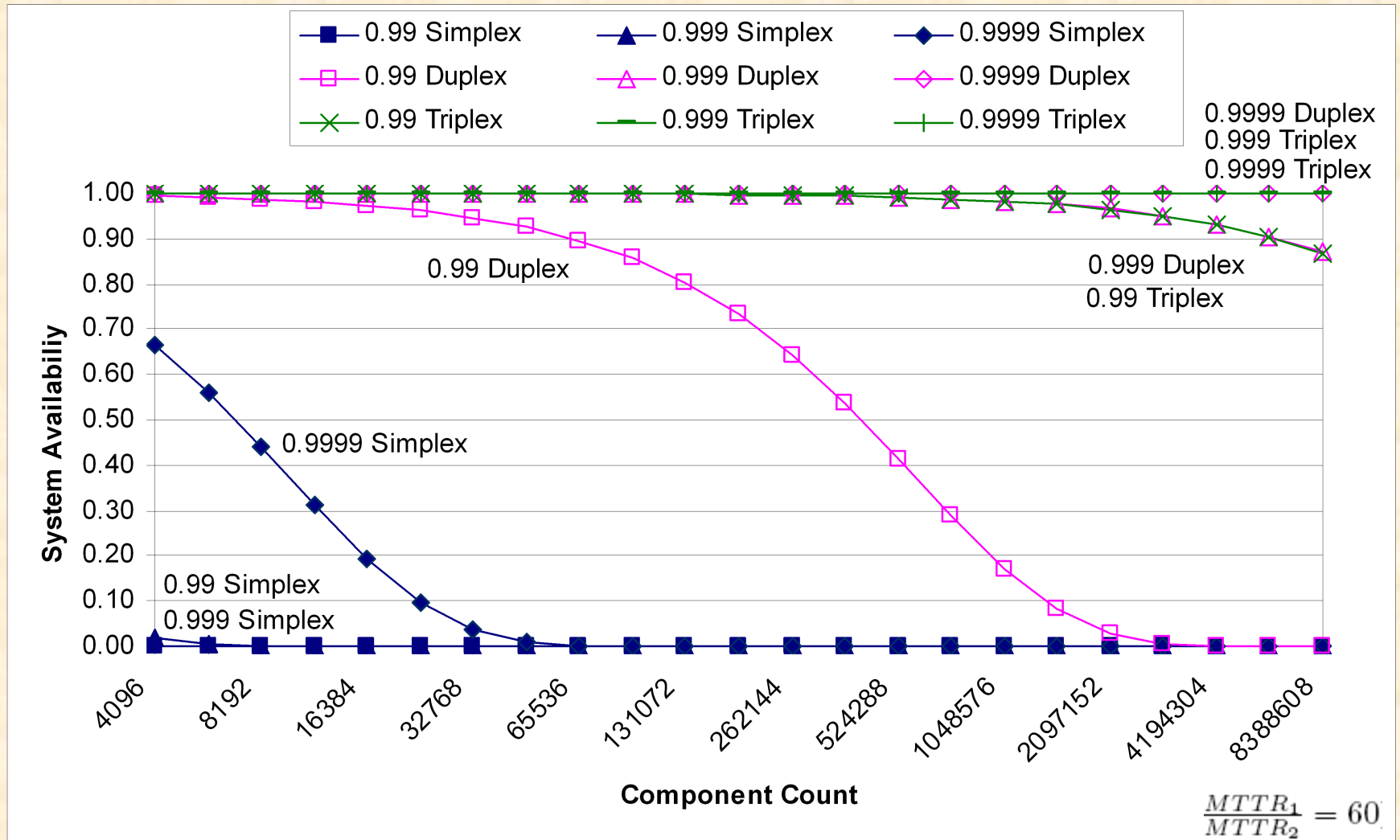
HPC System Availability Improvement with Modular Redundancy

(2, 3 and 4 Nines Compute Node Availability)



HPC System Availability Improvement with Dynamic Modular Redundancy

(2, 3 and 4 Nines Compute Node Availability)



Observations

- **DMR and TMR for compute nodes significantly increases compute node availability, which in turn dramatically increases HPC system availability**
- **DMR: Compute node MTTF can be 100-1,000 × less**
- **TMR: Compute node MTTF can be 1,000-10,000 × less**
- **DDMR and DTMR for compute nodes improve compute node availability even further, which in turn increases HPC system availability even more**
- **DDMR: Compute node MTTF can be 1,000-10,000 × less**
- **DTMR: Compute node MTTF can be 10,000-100,000 × less**

Financial Cost and Power Consumption

(Based on 2009 ORNL Jaguar Hardware and Market Prices)

Solution	Processor	Memory	Price	Power	
Traditional checkpoint/restart	1x AMD Opteron 2356	2x4GB Micron DDR2-800 ECC	\$ 500 +\$ 750 =\$1250	75W + 2W = 77W	Rollback Recovery
In-memory checkpoint caching	1x AMD Opteron 2356	4x4GB Micron DDR2-800 ECC	\$ 500 +\$1500 =\$2000	75W + 4W = 79W	
In-memory checkpoint/restart with new boards	1x AMD Opteron 2356	2x4GB Micron DDR2-800 ECC 4x4GB Kingston DDR2-800 ECC	\$ 500 +\$ 750 +\$ 600 >\$1700	75W + 2W + 4W = 81W	
DMR w/new boards & more racks	2x AMD Opteron 2356	4x4GB Kingston DDR2-800 ECC	\$1000 +\$ 600 >\$1600	150W + 4W =154W	Redundancy
TMR w/new boards & more racks	3x AMD Opteron 2356	6x4GB Kingston DDR2-800 ECC	\$1500 +\$ 900 >\$2400	225W + 6W >231W	

Financial Cost and Power Consumption

(Based on 2009 ORNL Jaguar Hardware and Market Prices)

Solution	Processor	Memory	Price	Power	
Traditional checkpoint/restart	1x AMD Opteron 2356	2x4GB Micron DDR2-800 ECC	=\$1250	= 77W	Rollback Recovery
In-memory checkpoint caching	1x AMD Opteron 2356	4x4GB Micron DDR2-800 ECC	=160%	=103%	
In-memory checkpoint/restart with new boards	1x AMD Opteron 2356	2x4GB Micron DDR2-800 ECC 4x4GB Kingston DDR2-800 ECC	=136%	=105%	
DMR w/new boards & more racks	2x AMD Opteron 2356	4x4GB Kingston DDR2-800 ECC	>128%	=200%	Redundancy
TMR w/new boards & more racks	3x AMD Opteron 2356	6x4GB Kingston DDR2-800 ECC	>192%	>300%	

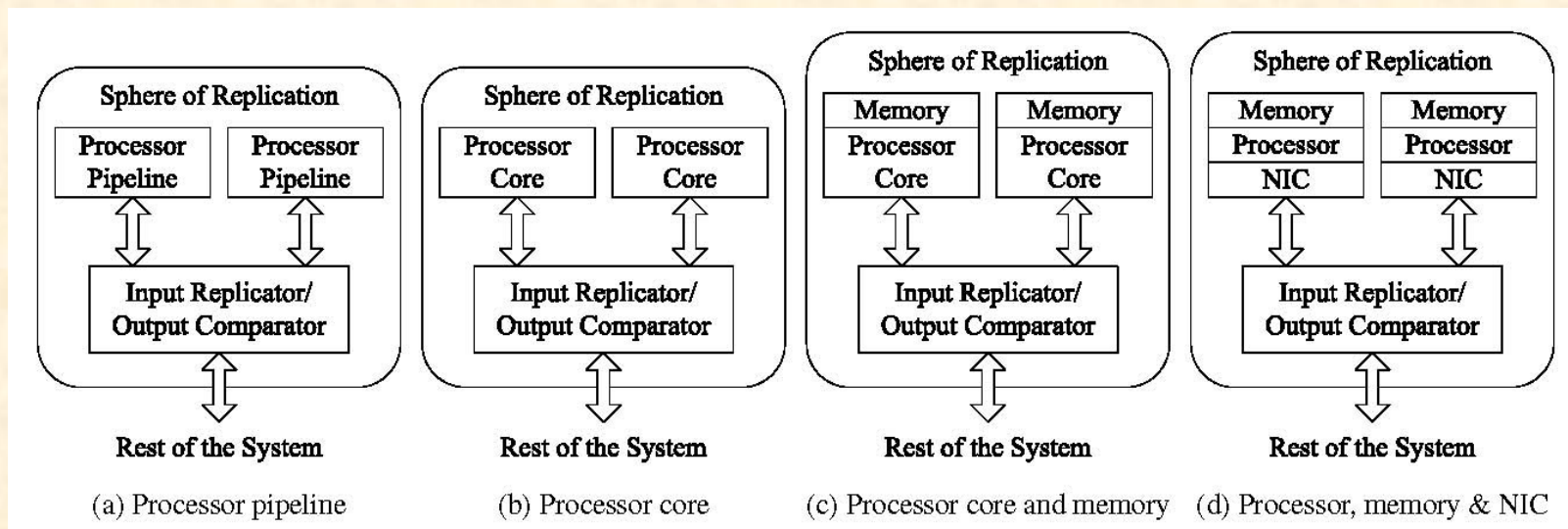
Not 2x/3x!

Availability Analysis Conclusions

- **DMR with 4-nine or TMR with 3-nine compute node rating provides enough system availability for HPC systems planned for the next 10 years with 1,000,000 compute nodes and beyond**
- **DDMR with 3-nine or DTMR with 2-nine single component rating provides enough overall system availability for future HPC systems**
- **The reduction of individual component reliability within a modular redundant system permits recovering the costs for using $2 \times$ or $3 \times$ the number of components**
- **This tunable cost vs. reliability/availability trade-off is the counter argument to the traditional view that modular redundancy comes at $2 \times$ or $3 \times$ costs**

Implementation as Sphere of Replication

- A logical boundary of redundancy for a replicated system
- Components within are protected via redundancy
- Components outside are not
- Data entering the sphere (input) is replicated
- Data leaving the sphere (output) is compared

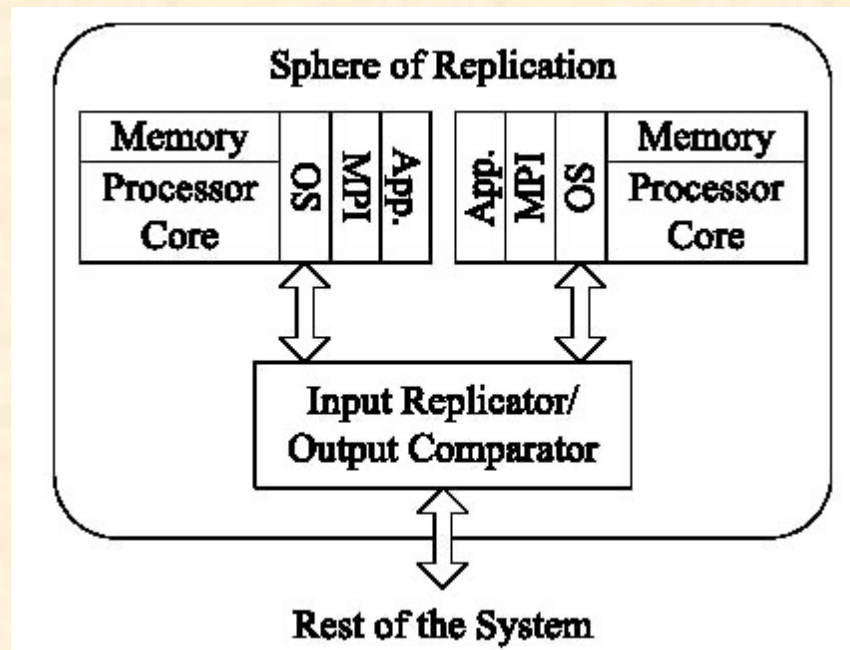


Existing non-HPC Solutions

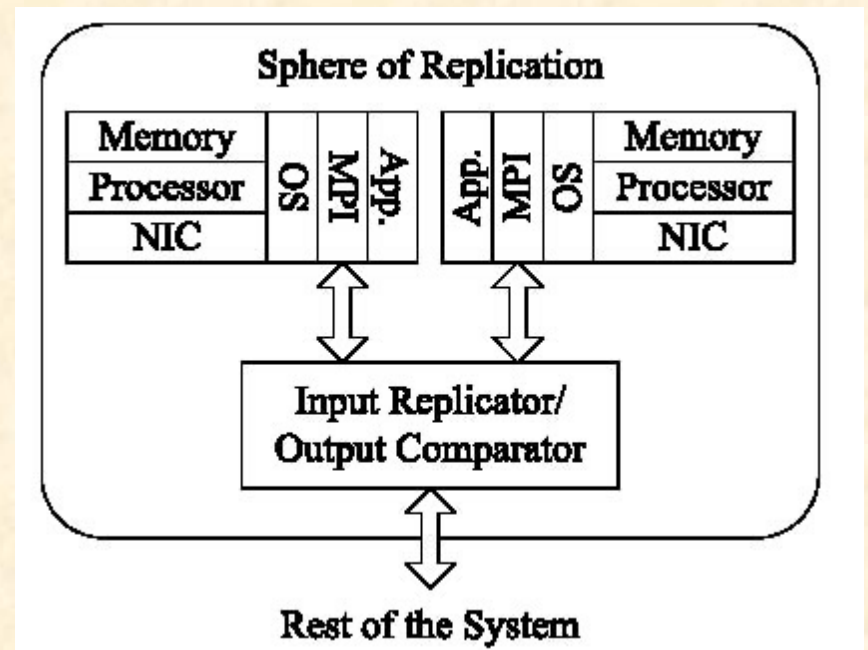
- **Thread-level redundancy**
 - On the same processor core using different pipelines
 - On the same processor using different cores
 - On different processors in the same system
 - *HP's NonStop Advanced Architecture (NSAA) (form. Tandem)*
- **Process-level redundancy**
 - On the same processor using different cores
 - On different processors in the same system
 - *HP's NonStop Himalaya*
- **System-level redundancy**
 - On split or separate systems
 - *Saturn V guidance computer, Boeing/Airbus navigation*

Proposed Solution for Modular Redundancy for Extreme-scale HPC

Process-level approach



System-level approach



Ongoing and Future Work

- **Design modular redundancy models and algorithms**
- **Implement a static modular redundancy prototype with system-level replication (MR-MPI)**
 - Interact with MPI fault tolerance approaches
 - Provide redundancy layer between application and MPI
 - Investigate moving the redundancy layer into MPI
- **Experiment with I/O & file system access scenarios**
 - Performance as well as failure modes, masking and recovery
- **Implement dynamic modular redundancy prototype with system-level replication (dMR-MPI)**
 - Utilize process cloning for hard- and soft-error recovery
- **Create trade-off models**
 - When does it make sense to use which type of redundancy?

HPC Computer Science Research Driver: The Road to Exa-Scale Computing

Systems	2009	2018	Difference Today & 2018
System peak	2 Pflop/s	1 Eflop/s	O(1000)
Power	6 MW	~20 MW	
System memory	0.3 PB	32 - 64 PB [.03 Bytes/Flop]	O(100)
Node performance	125 GF	1,2 or 15TF	O(10) - O(100)
Node memory BW	25 GB/s	2 - 4TB/s [.002 Bytes/Flop]	O(100)
Node concurrency	12	O(1k) or 10k	O(100) - O(1000)
Total Node Interconnect BW	3.5 GB/s	200-400GB/s (1:4 or 1:8 from memory BW)	O(100)
System size (nodes)	18,700	O(100,000) or O(1M)	O(10) - O(100)
Total concurrency	225,000	O(billion) [O(10) to O(100) for latency hiding]	O(10,000)
Storage	15 PB	500-1000 PB (> 10x system memory is min)	O(10) - O(100)
IO	0.2 TB	60 TB/s (how long to drain the machine)	O(100)
MTTI	days	O(1 day)	- O(10)

Collaboration Opportunities

- **Exa-scale computing research:**
 - U.S. Department of Energy calls for proposals are out (x-stack, advanced architectures, sci. data management)
 - G8 (Europe) calls for proposals are out
 - U.S. National Science Foundation calls will be out soon
- **Focus areas:**
 - Power and heat management (hardware & system software)
 - Billion-way parallelism (algorithms, math libraries, programming models, environments & frameworks)
 - Scalability (system software: management, I/O & file systems)
 - Fault tolerance and resilience (system software)
 - Information hiding to reduce complexity (system software, programming models, environments & frameworks)

Crosscutting efforts: resilience/power/performance/...

Questions?