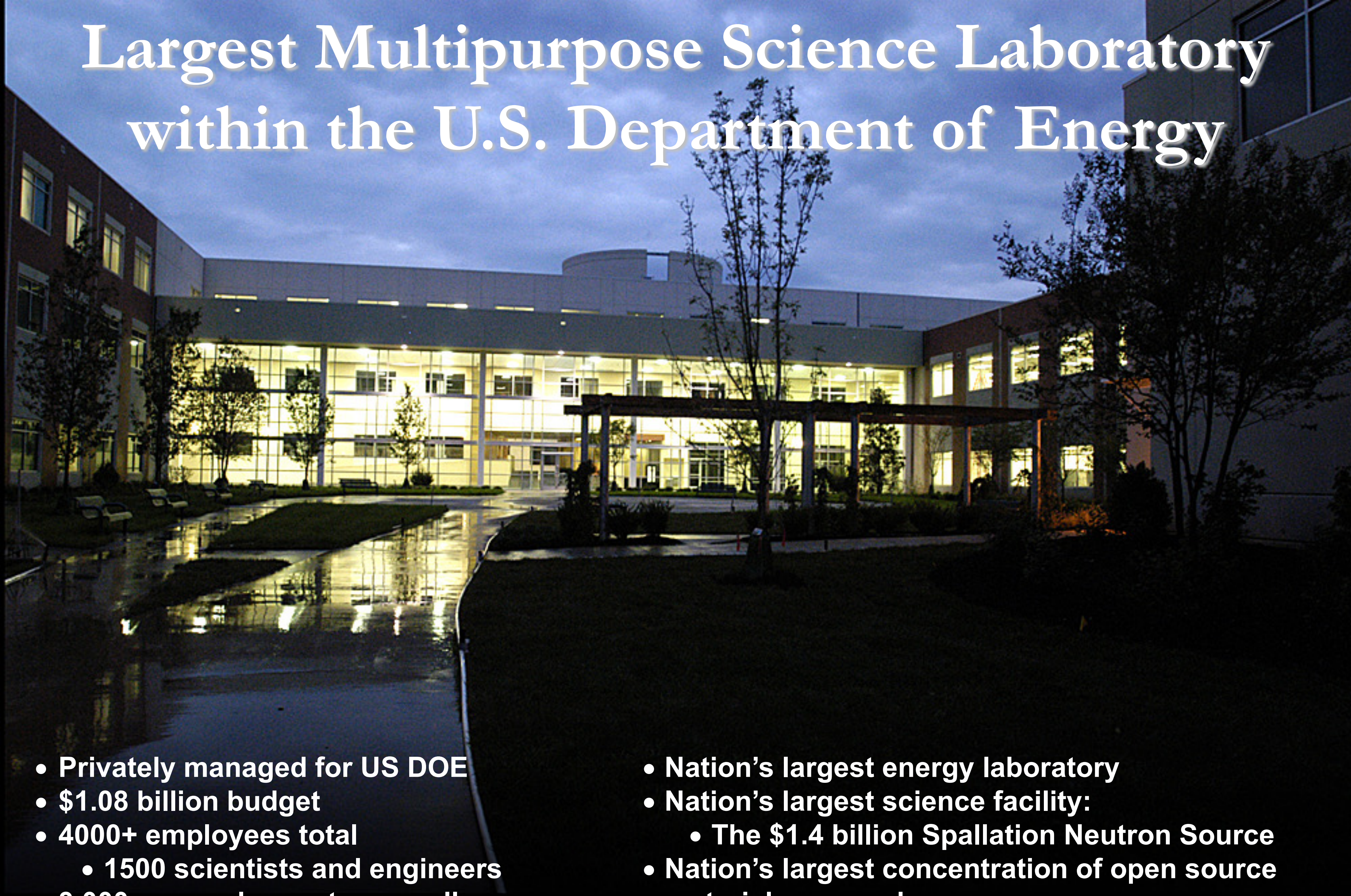
Advanced Fault Tolerance Solutions for High Performance Computing

Christian Engelmann

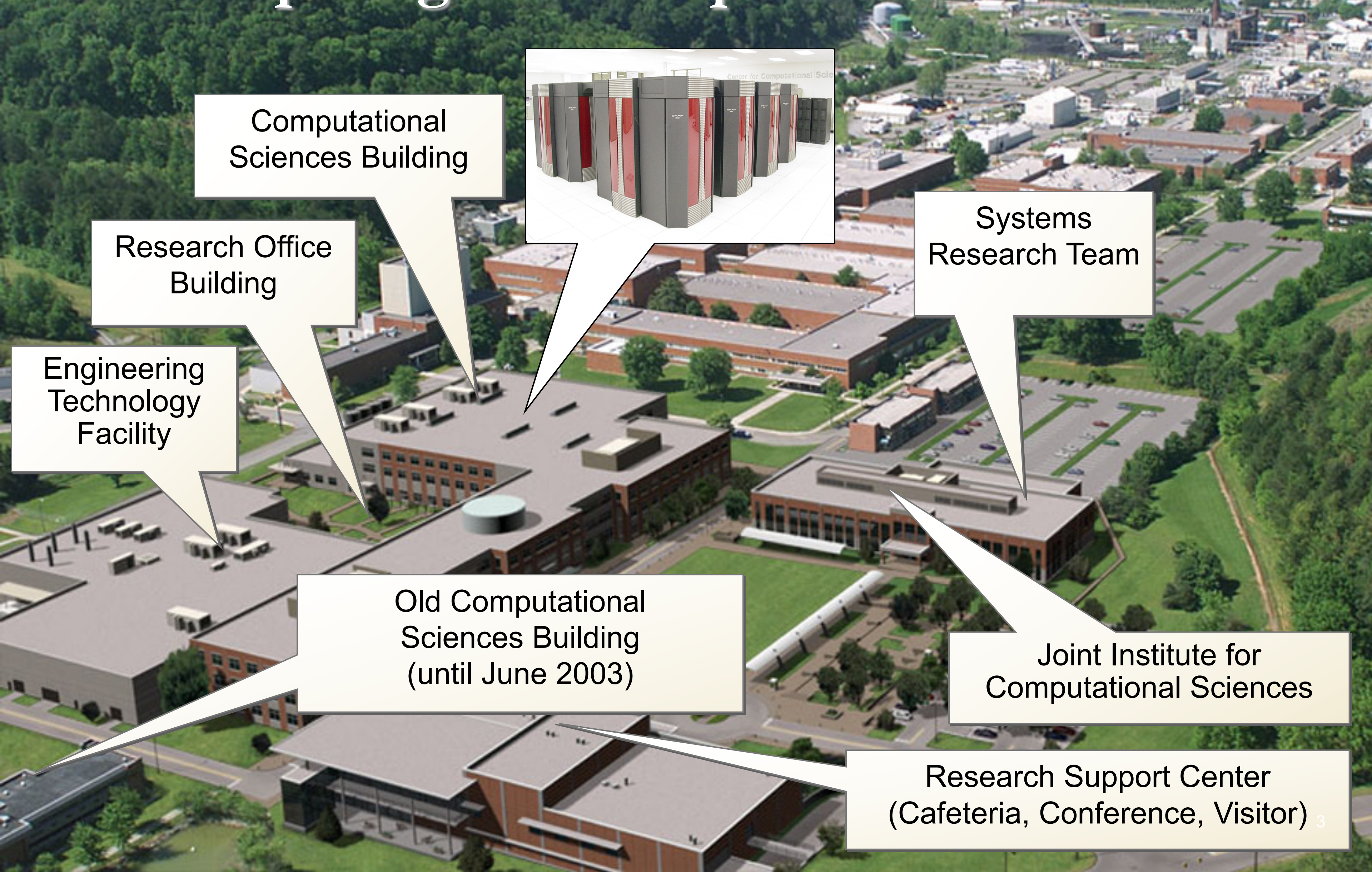
Oak Ridge National Laboratory, Oak Ridge, USA
The University of Reading, Reading, UK

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



- Privately managed for US DOE
- $1.08 billion budget
- 4000+ employees total
  - 1500 scientists and engineers
- 3,000 research guests annually
- 30,000 visitors each year
- Total land area 58mi^2 (150km^2)
- 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 open scientific computing facility

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



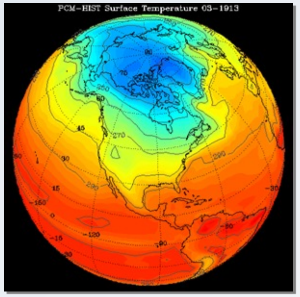
National Center for Computational Sciences

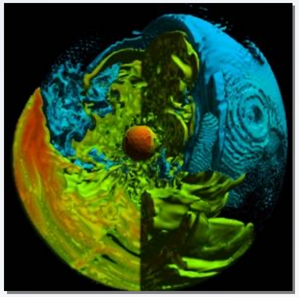
- **40,000 ft² (3700 m²) computer center:**
  - **36-in (~1m) raised floor, 18 ft (5.5 m) deck-to-deck**
  - **12 MW of power with 4,800 t of redundant cooling**
  - **High-ceiling area for visualization lab:**
    - **35 MPixel PowerWall, Access Grid, etc.**

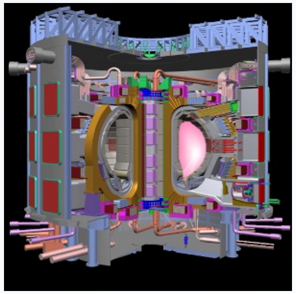
- **3 systems in the Top 500 List of Supercomputer Sites:**
  - **Jaguar: 7. Cray XT3, MPP with 11508 dual-core Processors ⇨ 119 TFlop**
  - **Jaguar: 41. IBM Blue Gene/P, MPP with 2048 quad-core Processors ⇨ 27 TFlop**
  - **Phoenix: 80. Cray X1E, Vector with 1014 Processors ⇨ 18 TFlop**

At Forefront in Scientific Computing and Simulation

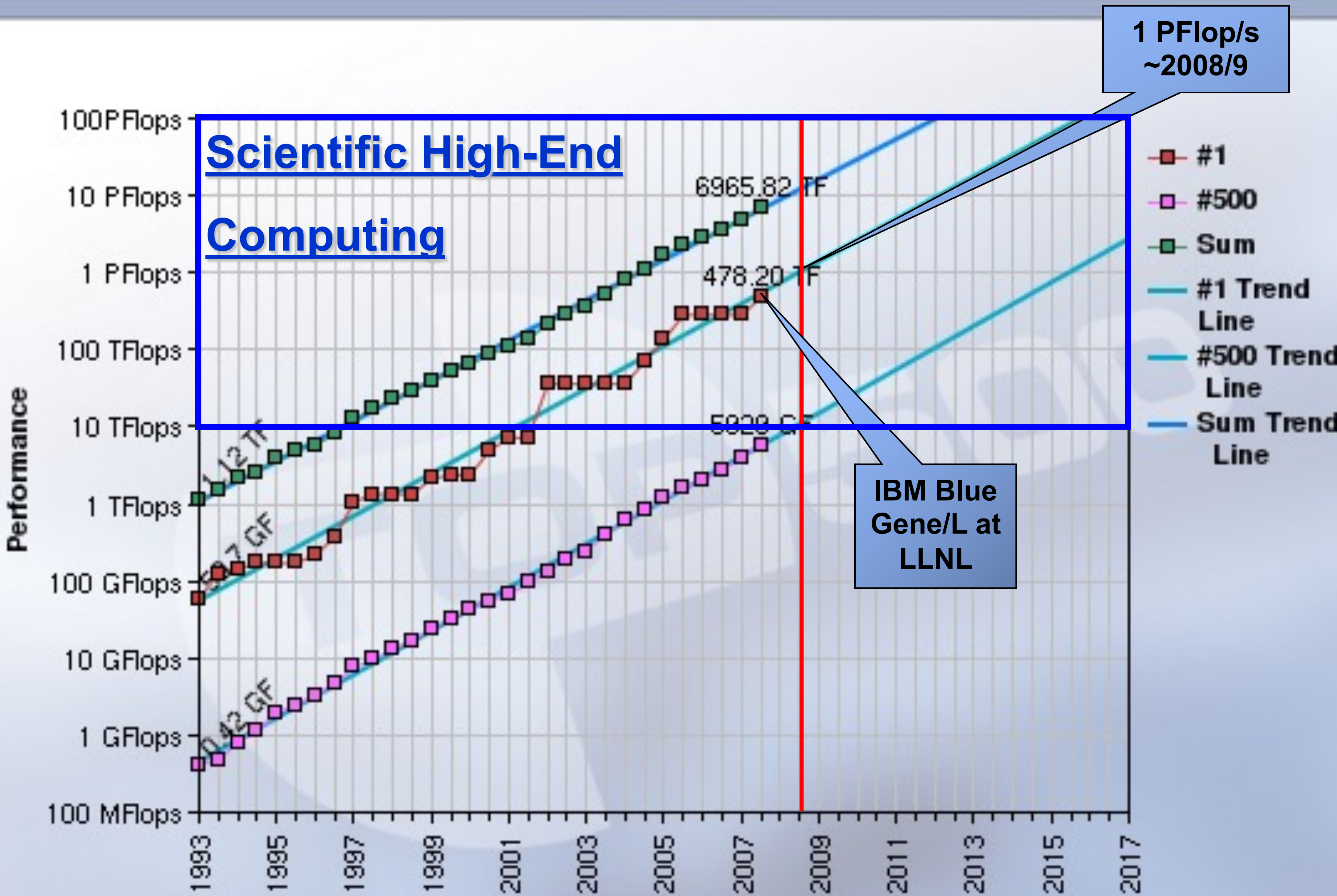
- Leading partnership in developing the National Leadership Computing Facility
  - Leadership-class scientific computing capability
  - 250 TFlop/s in 2008 (upgrade in progress)
  - 500 TFlop/s in 2008 (commitment made)
  - 1 PFlop/s in 2008/9 (commitment made)
- Attacking key computational challenges
  - Climate change
  - Nuclear astrophysics
  - Fusion energy
  - Materials sciences
  - Biology
- Providing access to computational resources through high-speed networking







Projected Performance Development



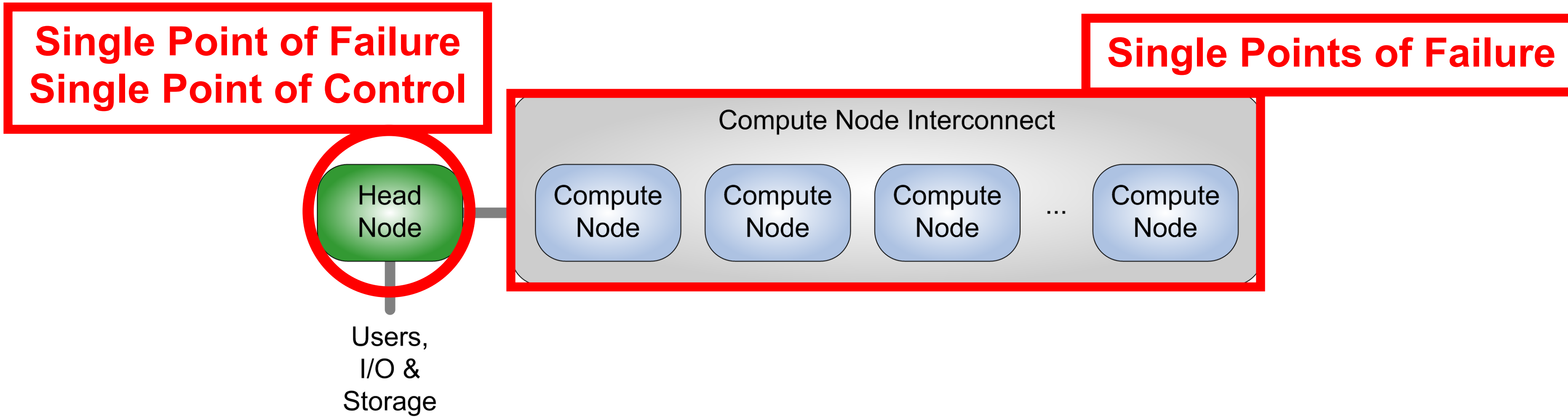
Talk Outline

- High performance computing system architectures
- Fault tolerance solutions for head & service nodes:
  - Active/standby with shared storage
  - Active/standby replication
  - Asymmetric active/active replication
  - Symmetric active/active replication
- Fault tolerance solutions for compute nodes:
  - Reactive: Checkpoint/restart and message logging
  - Proactive: Preemptive migration
  - Algorithmic approaches

Advanced Fault Tolerance Solutions for High Performance Computing

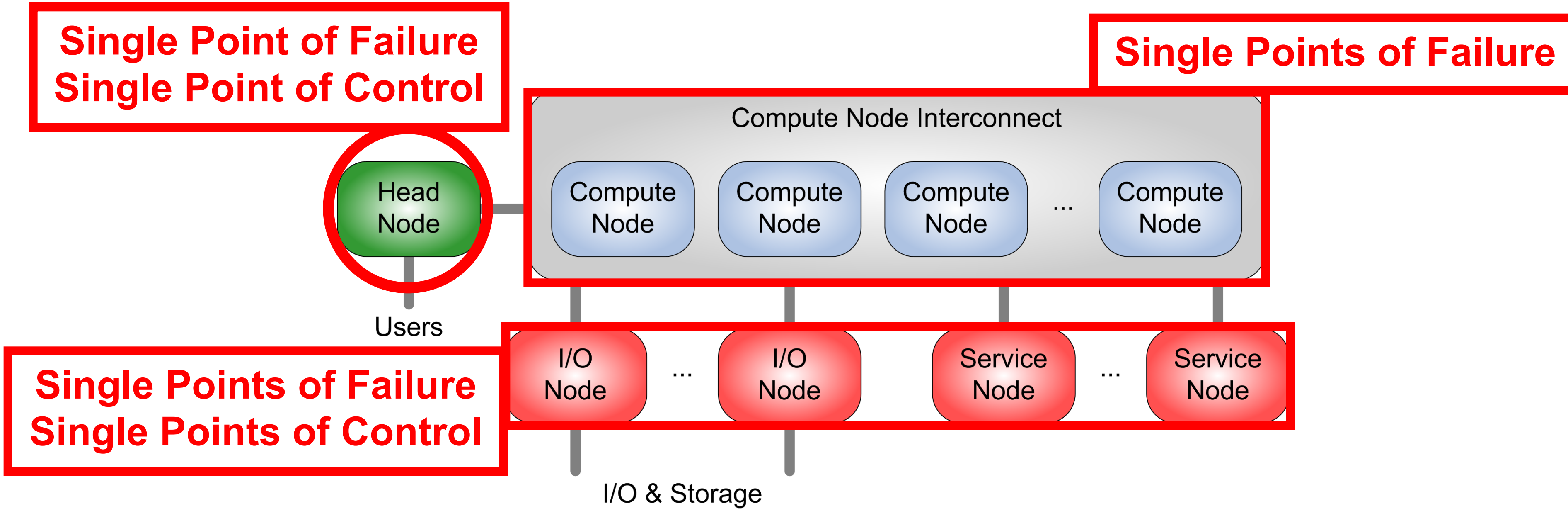
HPC System Architectures

Beowulf Cluster Computing Architecture



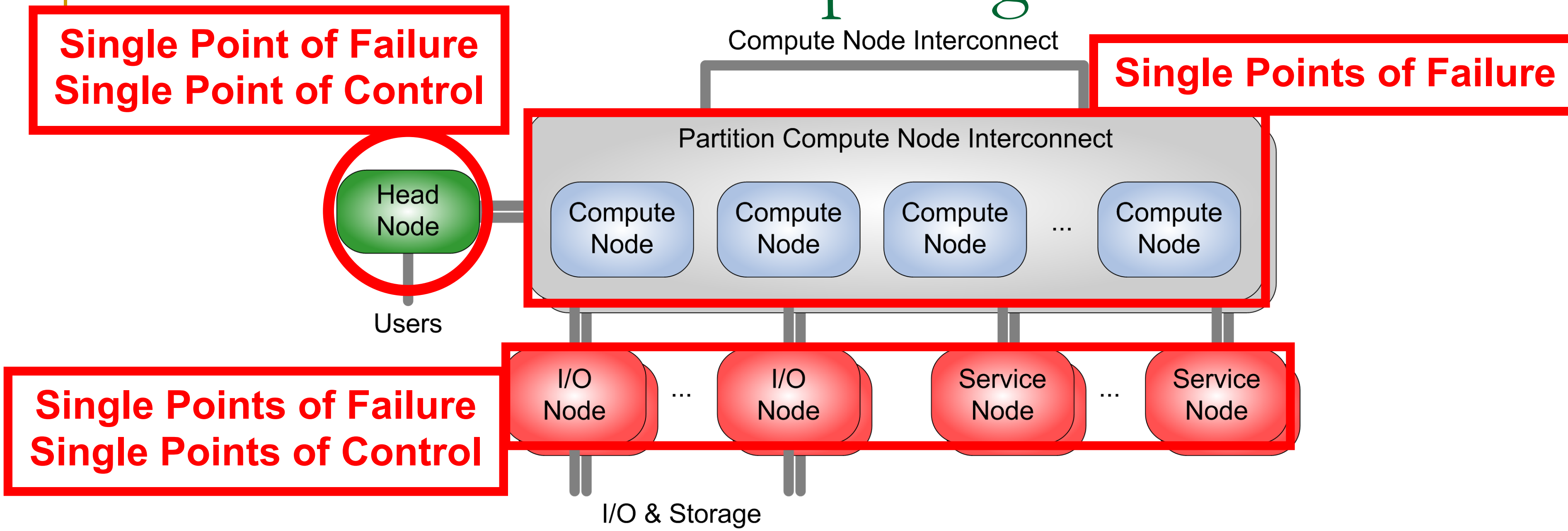
- Single head node manages entire HPC system
- System-wide services are provided by head node:
  - Job & resource management, networked file system, …
- Local services are provided by compute nodes
  - Message passing (MPI, PVM), …

Massively Parallel Computing Architecture



- Single head node and additional service nodes manage the entire HPC system
- System-wide services are provided by head node and are offloaded to service nodes, e.g., networked file system
- Local services are provided by service nodes and compute nodes, e.g., message passing

Partitioned MPP Computing Architecture



- Single head node manages entire HPC system
- Service nodes manage and support compute nodes belonging to their partitions

Typical Failure Causes in HPC Systems

- Overheating!!!
- Memory and network errors (bit flips)
- Hardware failures due to wear/age of:
  - Hard drives, memory modules, network cards, processors
- Software failures due to bugs in:
  - Operating system, middleware, applications

➔ Different scale requires different solutions:
  - ➔ Compute nodes (up to 150,000)
  - ➔ Front-end, service, and I/O nodes (1 to 150)

Availability Measured by the Nines

http://info.nccs.gov/resources - HPC system status at Oak Ridge National Laboratory

9's	Availability	Downtime/Year	Examples
1	90.0%	36 days, 12 hours	Personal Computers
2	99.0%	87 hours, 36 min	Entry Level Business
3	99.9%	8 hours, 45.6 min	ISPs, Mainstream Business
4	99.99%	52 min, 33.6 sec	Data Centers
5	99.999%	5 min, 15.4 sec	Banking, Medical
6	99.9999%	31.5 seconds	Military Defense

- Enterprise-class hardware + Stable Linux kernel = 5+
- Substandard hardware + Good high availability package = 2-3
- Today's supercomputers = 1-2
- My desktop = 1-2

Fault Tolerance & High Availability Goals

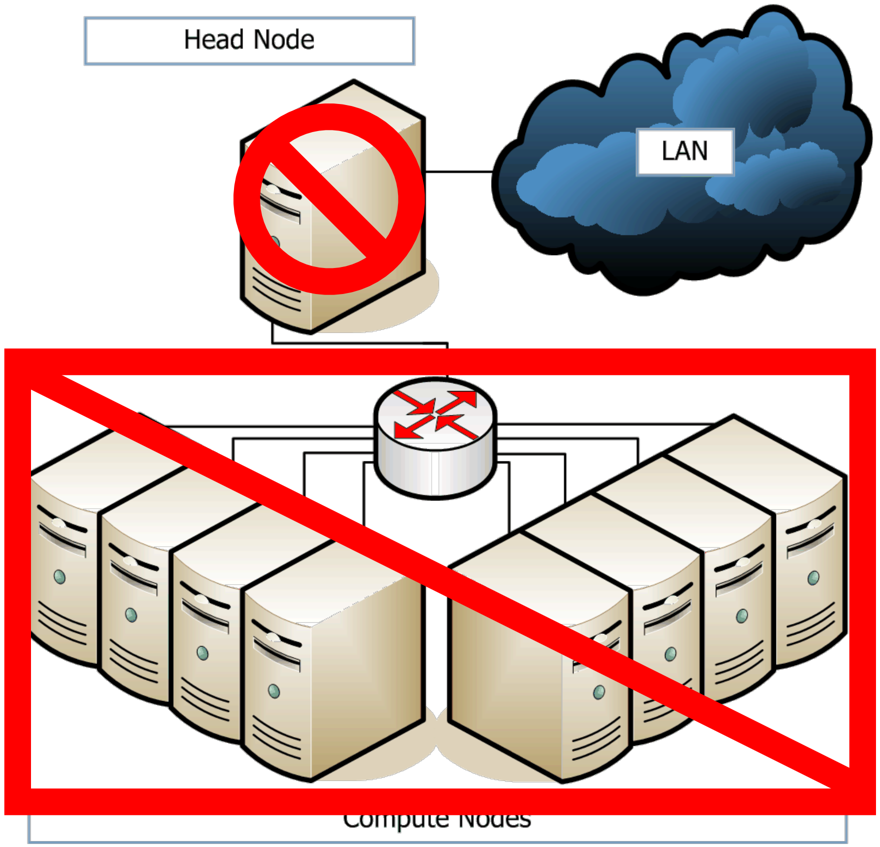
- Provide high-level Reliability, Availability, and Serviceability (RAS) capabilities
- Eliminate many of the numerous single-points of failure and control in HPC systems

- *Development of techniques to enable HPC systems to run computational jobs 24x7 without interruption*
- *Development of proof-of-concept implementations as blueprint for production-type RAS solutions*

Advanced Fault Tolerance Solutions for High Performance Computing

Head and Service Nodes

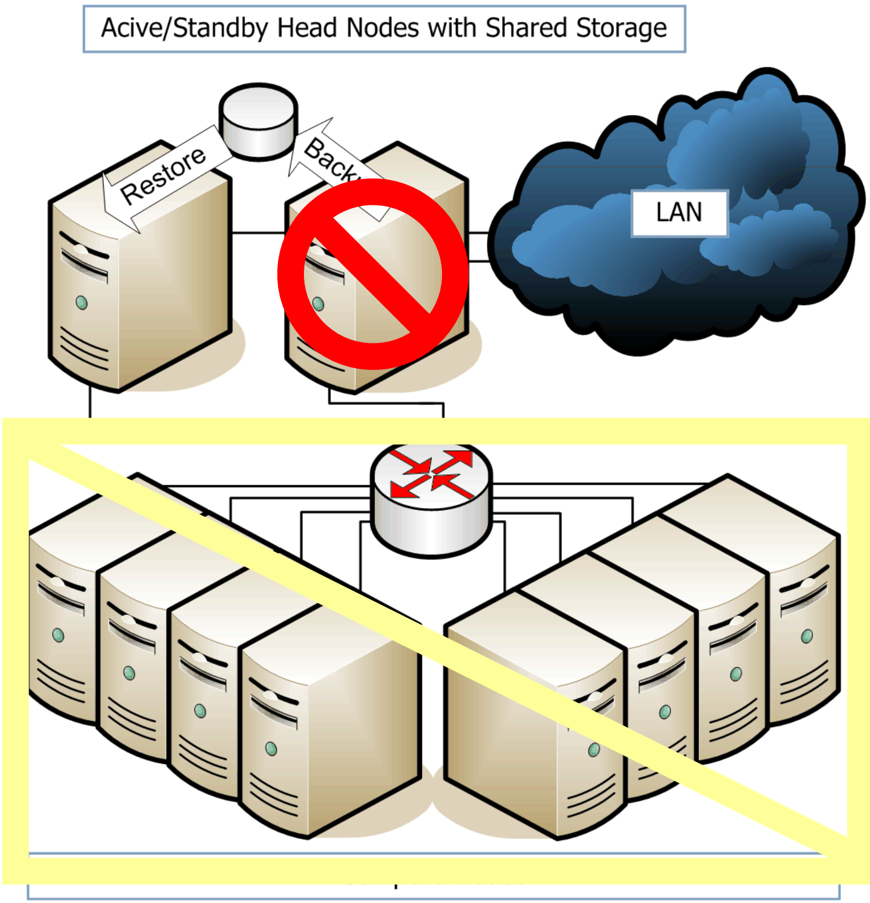
Single Head/Service Node Problem



- Single point of failure
- Compute nodes sit idle while head node is down
- A = MTTF / (MTTF + MTTR)
- MTTF depends on head node hardware/software quality
- MTTR depends on the time it takes to repair/replace node

➢ MTTR = 0 ➔ A = 1.00 (100%) **continuous availability**

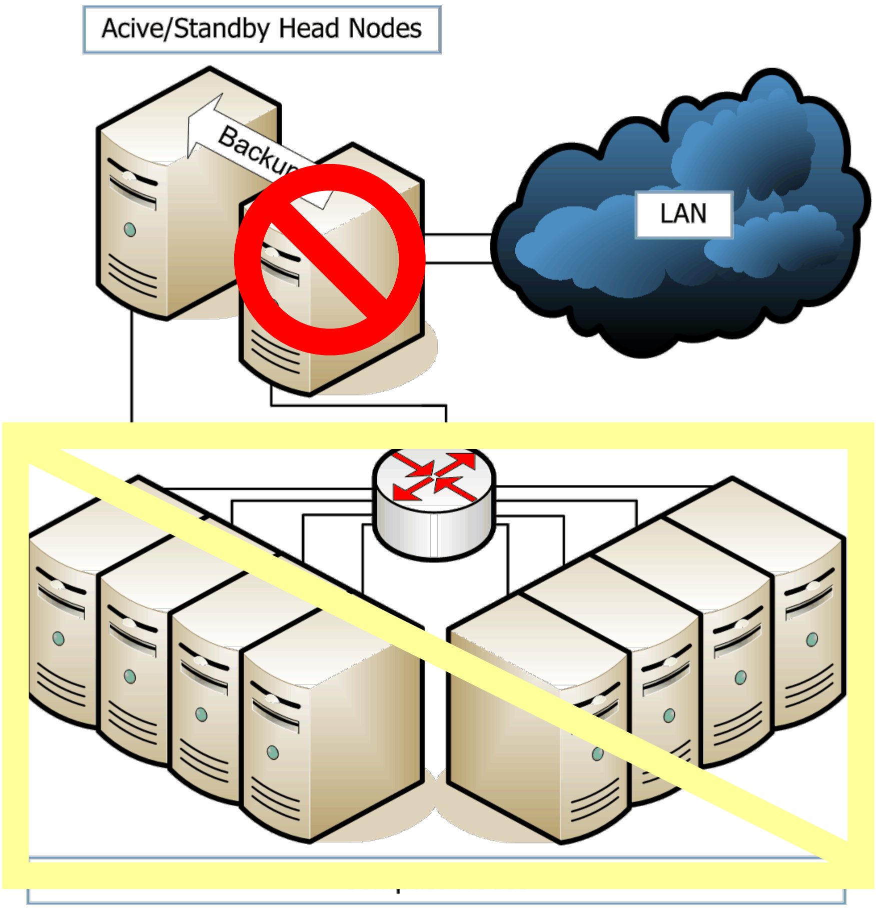
Active/Standby with Shared Storage



- Single active head node
- Backup to shared storage
- Simple checkpoint/restart
- Fail-over to standby node
- Possible corruption of backup state when failing during backup
- Introduction of a new single point of failure
- Correctness and availability are NOT ALWAYS guaranteed

➔ SLURM, metadata servers of PVFS and Lustre

Active/Standby Redundancy

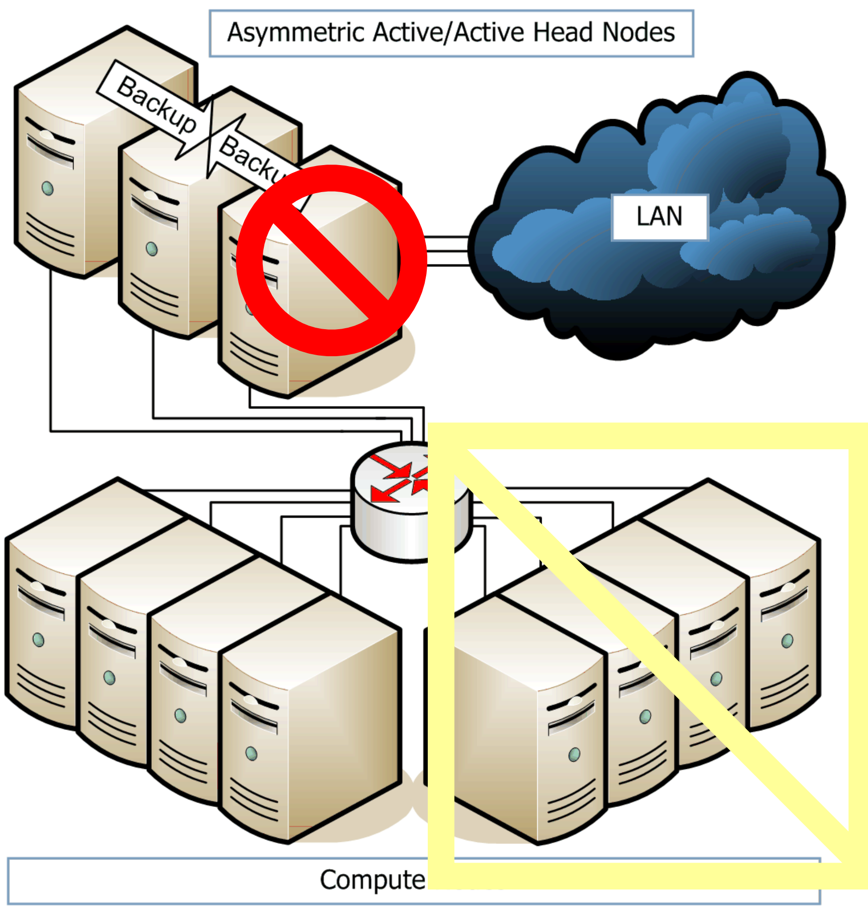


- Single active head node
- Backup to standby node
- Simple checkpoint/restart
- Fail-over to standby node
- Idle standby head node
- Rollback to backup
- Service interruption for fail-over and restore-over

➔ Torque on Cray XT

➔ HA-OSCAR prototype

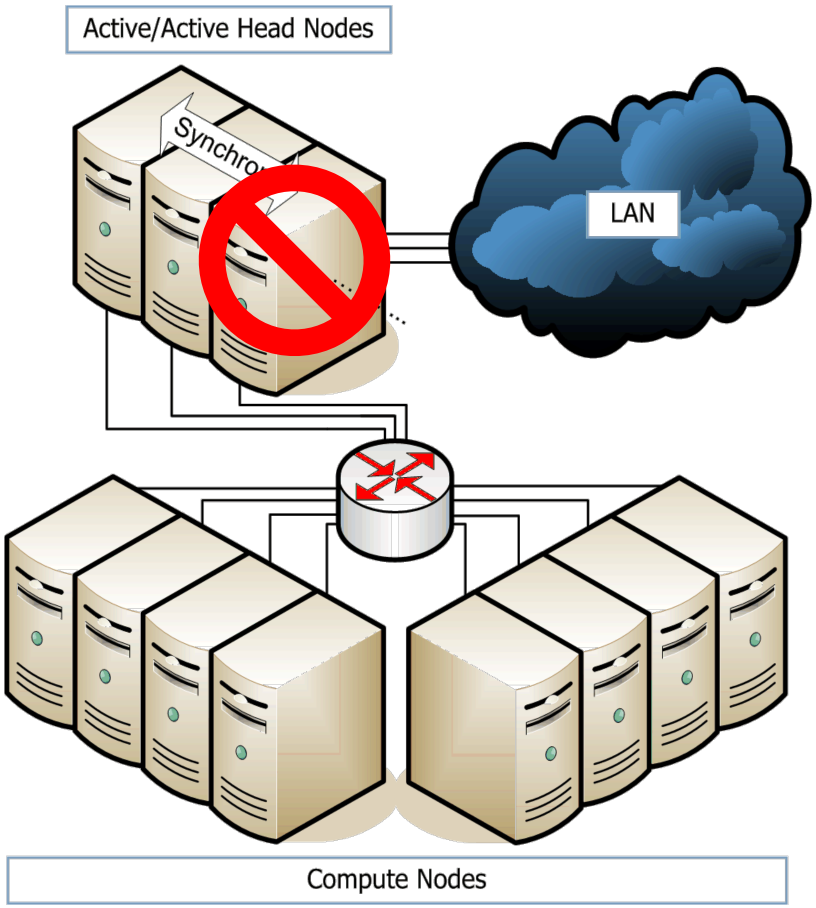
Asymmetric Active/Active Redundancy



- Many active head nodes
- Work load distribution
- Optional fail-over to standby head node(s) (*n+1* or *n+m*)
- No coordination between active head nodes
- Service interruption for fail-over and restore-over
- Loss of state w/o standby
- Limited use cases, such as high-throughput computing

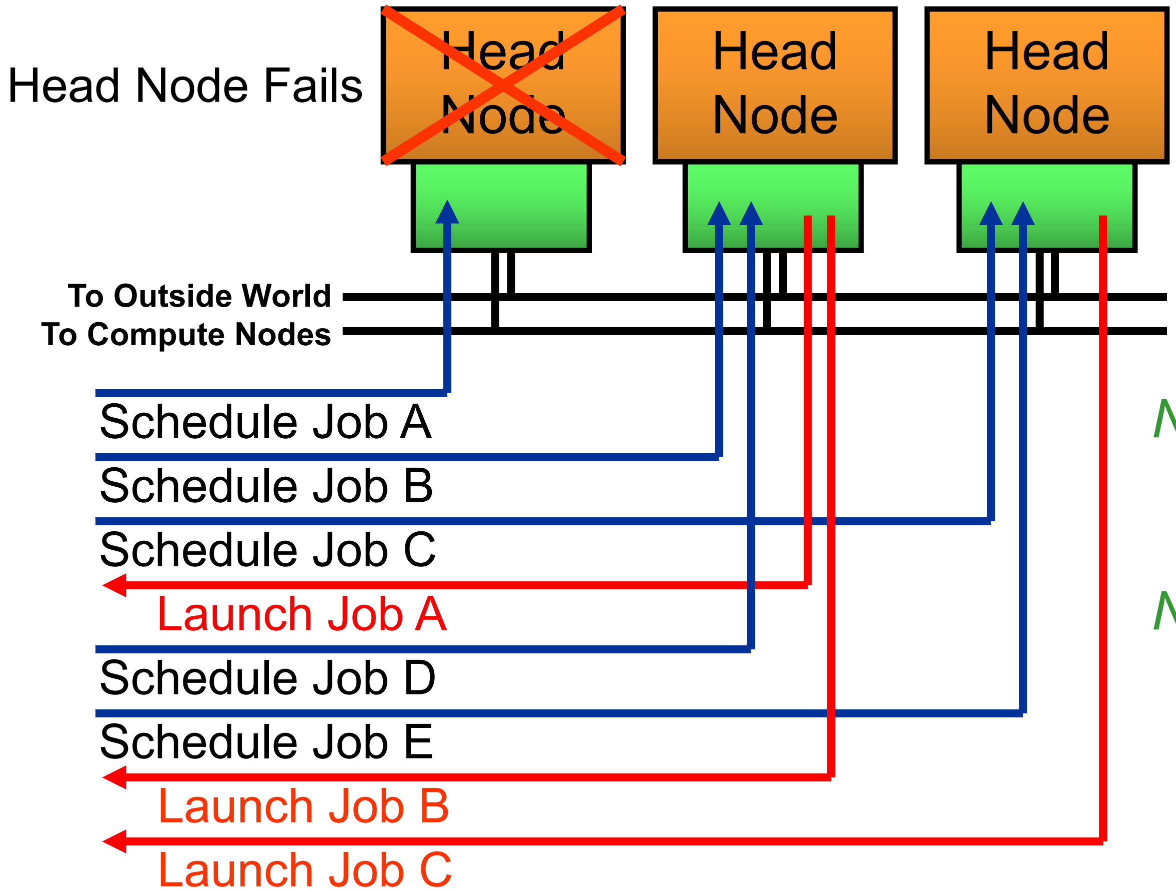
➔ Prototype based on HA-OSCAR

Symmetric Active/Active Redundancy



- Many active head nodes
- Work load distribution
- Symmetric replication between head nodes
- Continuous service
- Always up-to-date
- No fail-over, no restore-over
- Virtual synchrony model
- Complex algorithms
- <u>JOSHUA prototype for Torque</u>
- <u>PVFS metadata server</u>

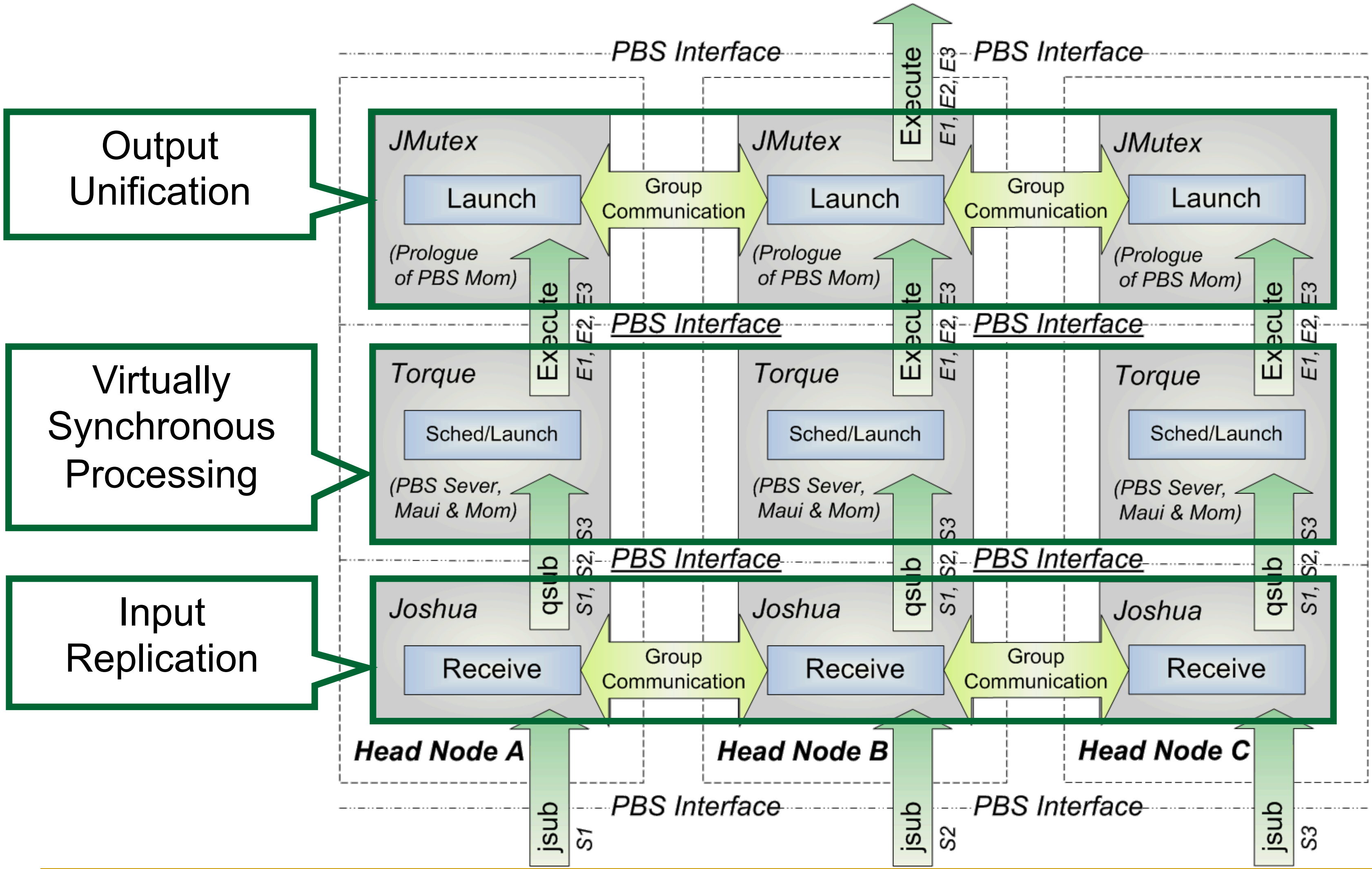
JOSHUA: Symmetric Active/Active Replication for PBS Torque



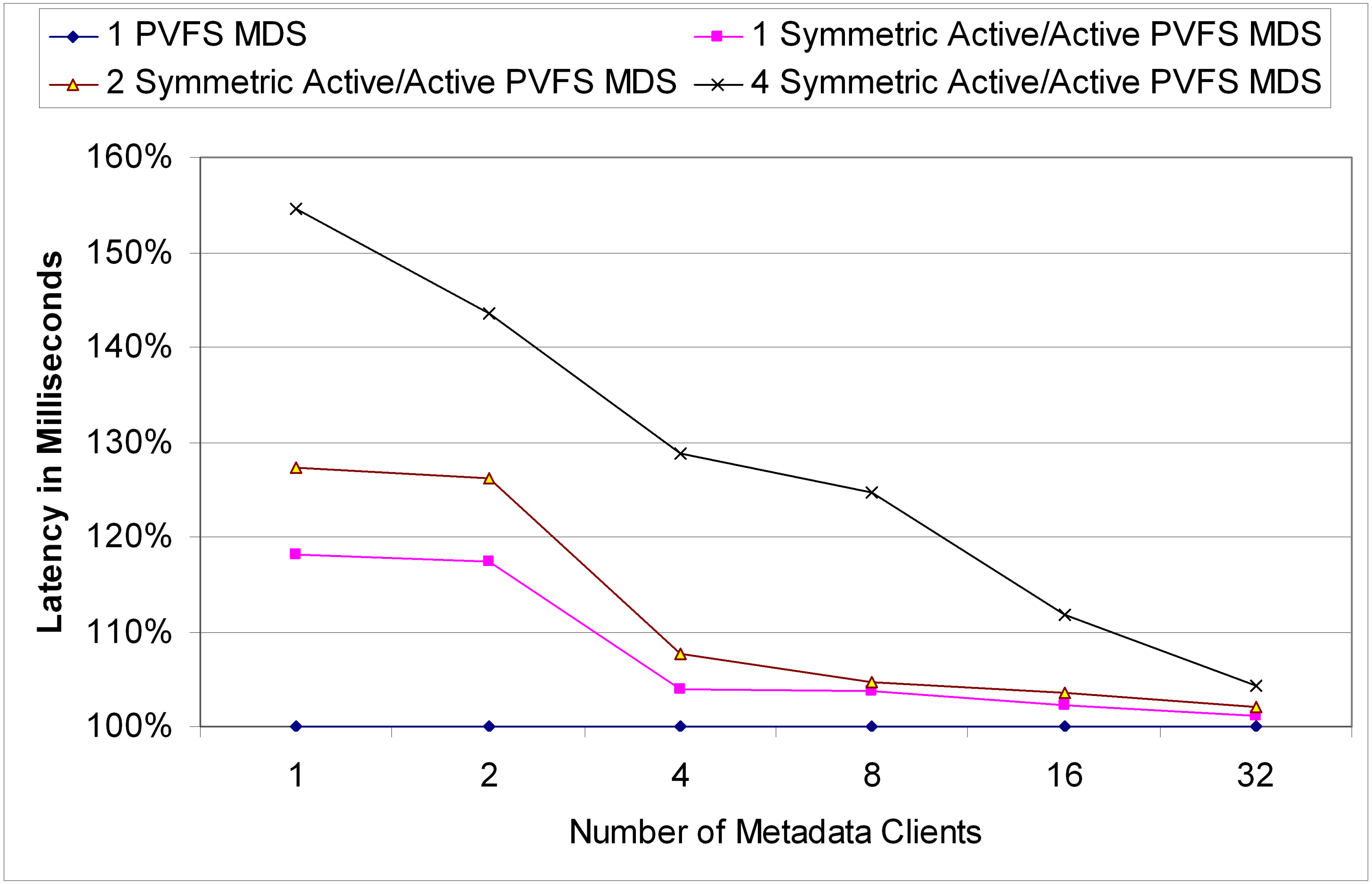
No Single Point of Failure

No Single Point of Control

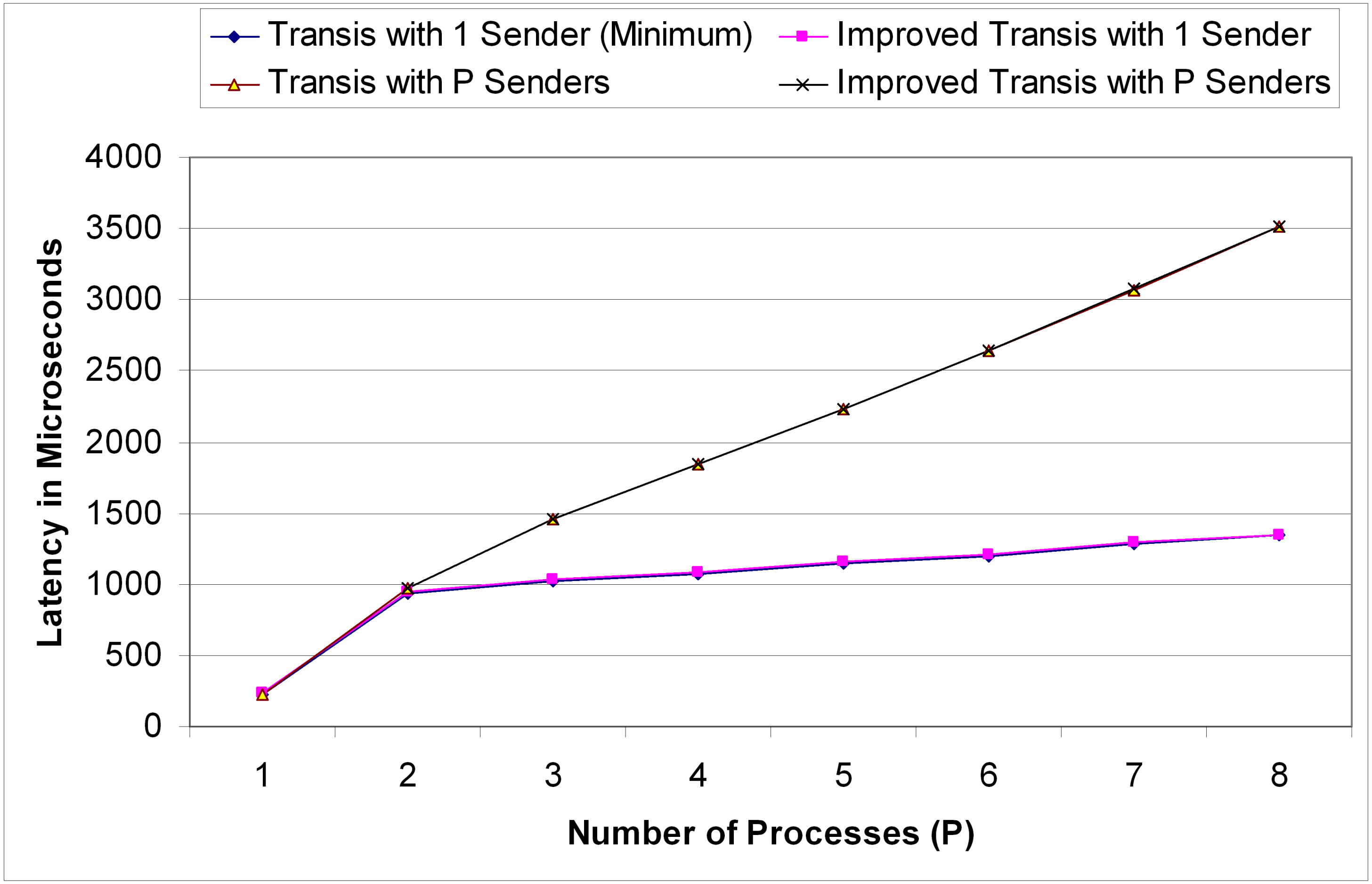
Symmetric Active/Active Replication



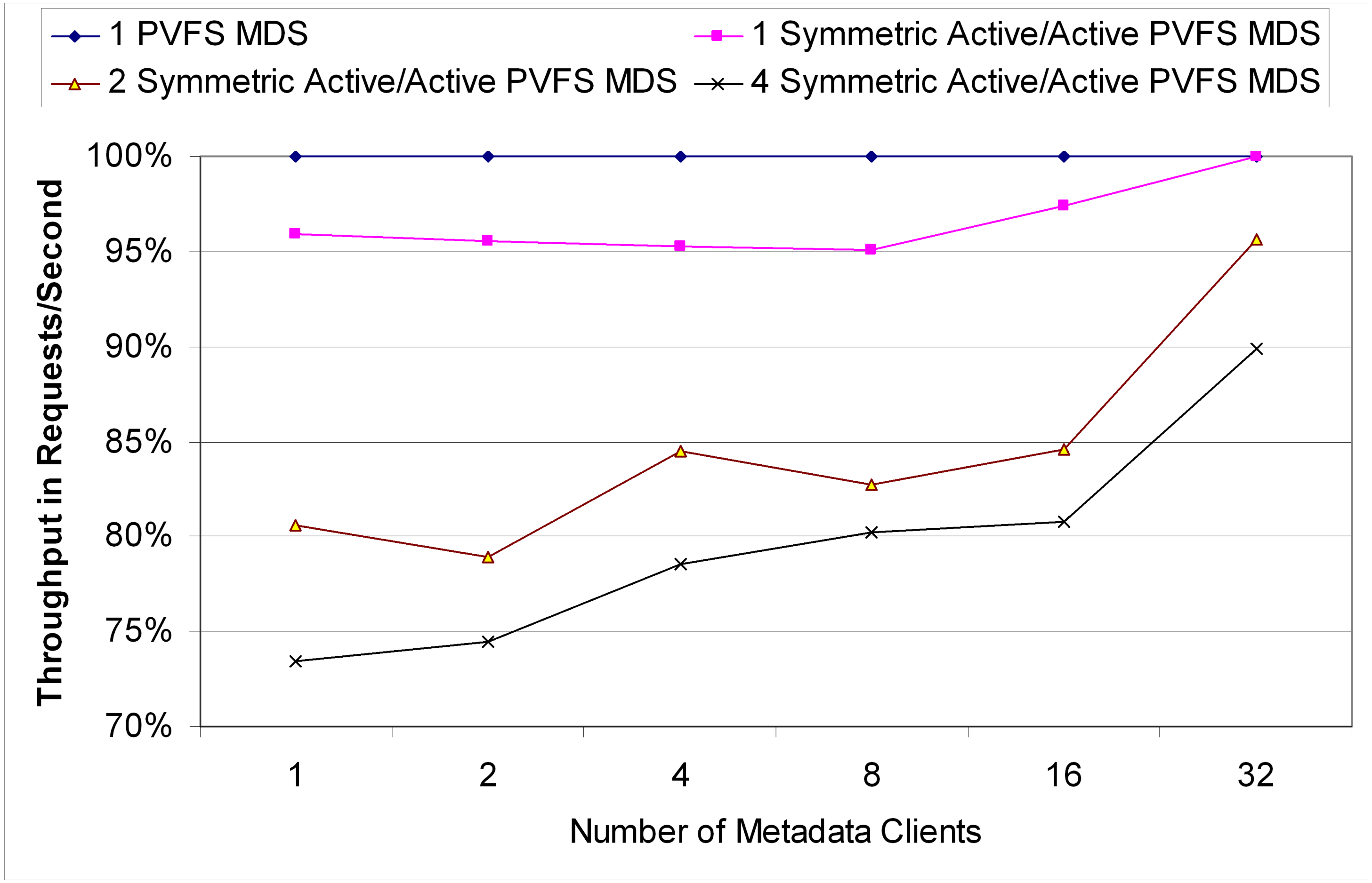
Symmetric Active/Active PVFS Metadata Service Latency



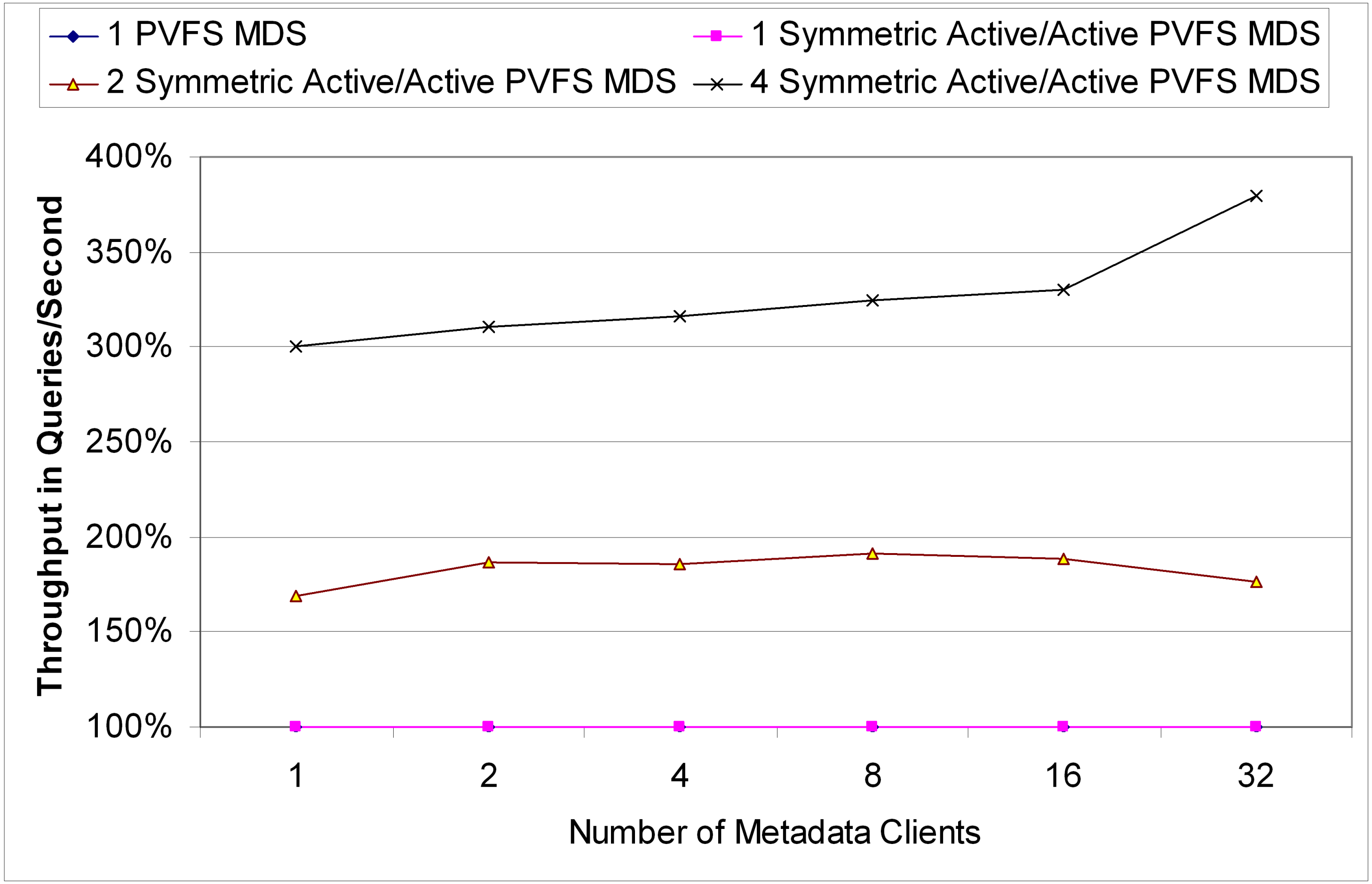
Total Message Order Latency



Symmetric Active/Active PVFS Metadata Service Write/Request Throughput



Symmetric Active/Active PVFS Metadata Service Read/Query Throughput



Symmetric Active/Active Availability

- $A_{component}$ = MTTF / (MTTF + MTTR)
- A_{system} = 1 - (1 - $A_{component}$) n
- T_{down} = 8760 hours * (1 – A)
- Single node MTTF: 5000 hours
- Single node MTTR: 72 hours

Nodes	Availability	Est. Annual Downtime
1	98.58%	5d 4h 21m
2	99.97%	1h 45m
3	99.9997%	1m 30s
4	99.999995%	1s

Single-site redundancy for 7 nines does not mask catastrophic events.

Advanced Fault Tolerance Solutions for High Performance Computing

Compute Nodes

Reactive vs. Proactive Fault Tolerance

- Reactive fault tolerance:
  - State saving during failure-free operation
  - State recovery **after failure**
  - Assured quality of service, but limited scalability
- Proactive fault tolerance:
  - System health monitoring and online reliability modeling
  - Failure anticipation and prevention through prediction and reconfiguration **before failure**
  - Highly scalable, but not all failures can be anticipated
- Ideal solution: Matching combination of both

Reactive Fault Tolerance Techniques (1/2)

- Checkpoint/restart:
  - Application state from all processors is saved regularly on stable storage, such as local disk or networked file system
  - On failure, application is restarted using saved state
  - Checkpoint always involves data movement (local/network)
  - Restart always involves a rollback, i.e., lost computation
  - Example: Berkeley Lab Checkpoint/Restart (Linux mod.)
  - May be used in combination with message logging to avoid rollback (see next slide)

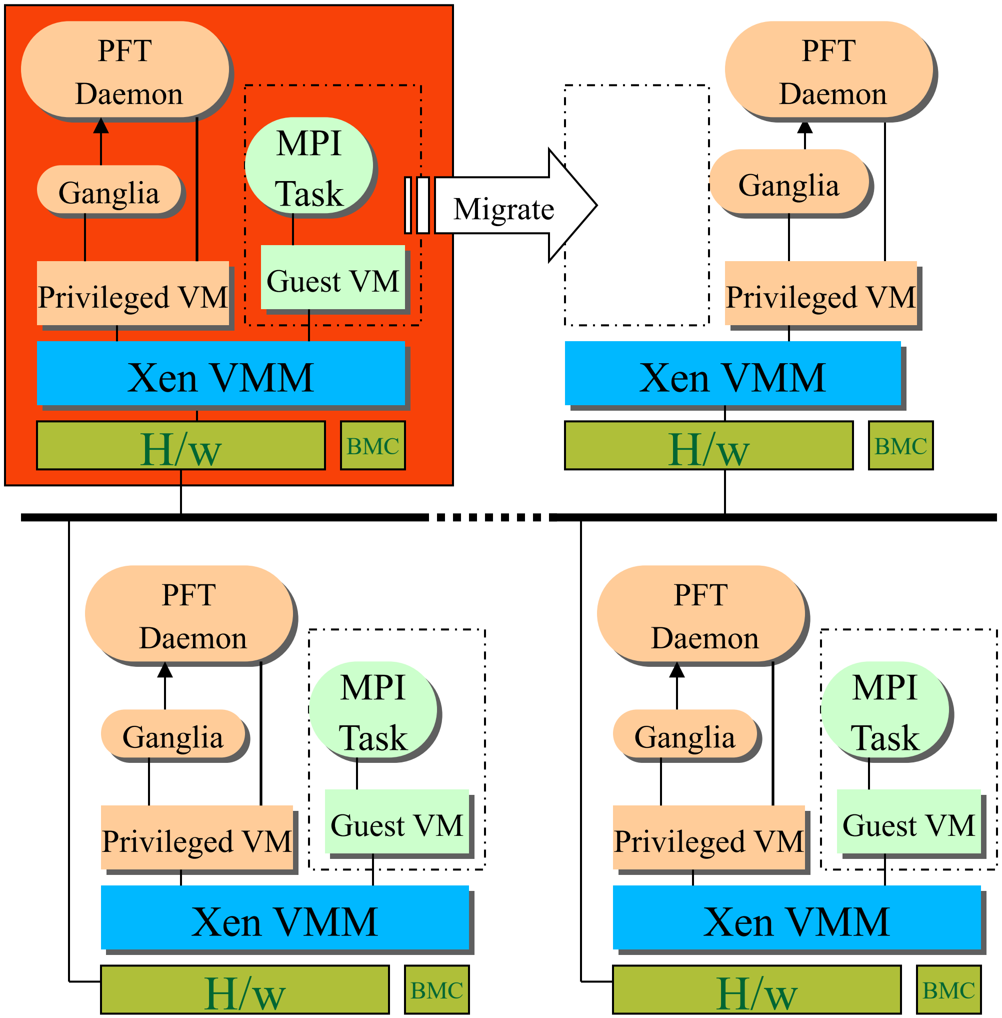
Reactive Fault Tolerance Techniques (2/2)

- Message logging:
  - All messages sent between application processes are logged to a central server
  - On failure, only the failed application part is restarted and replayed with saved messages
  - Doubles the number of messages
  - Message replay involves no rollback
  - Example: MPICH-VCL (MPI-based Chandy/Lamport alg.)
  - Combination with checkpoint/restart:
    - No rollback / shorter replay time, even higher overhead

Proactive Fault Tolerance Techniques

- Preemptive migration:
  - System health status is constantly monitored and evaluated
  - Monitoring data is processed by a filtering mechanism and/or an online reliability analysis
  - Pre-failure indicators are used to predict failures based on current system health status and historic information
  - Application parts (processes or virtual machines) are migrated away from compute nodes that are about to fail
  - Migration may be performed by stopping the application or live, while keeping the application running

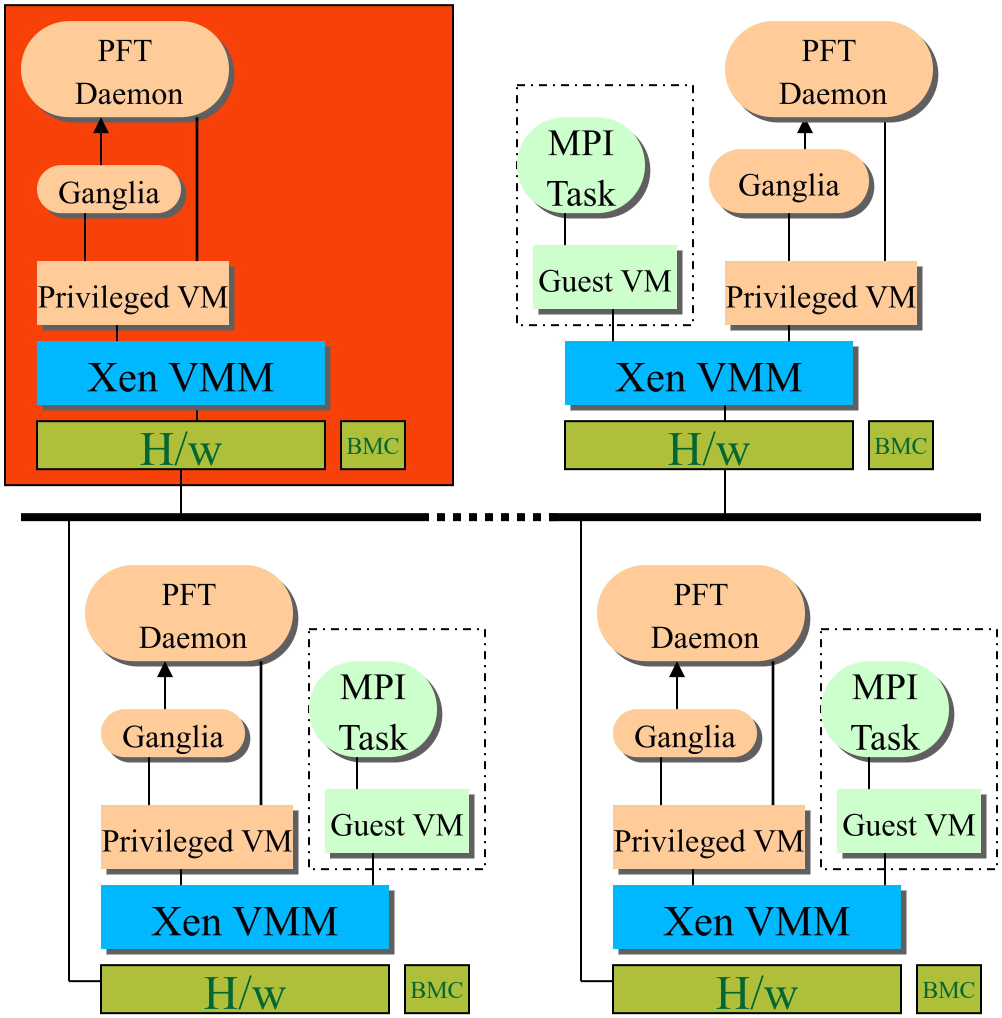
Preemptive Migration with Xen



BMC Baseboard Management Contoller

- Stand-by Xen host, no guest (spare node)
- Deteriorating health → migrate guest (w/ MPI app) to spare node

Preemptive Migration with Xen

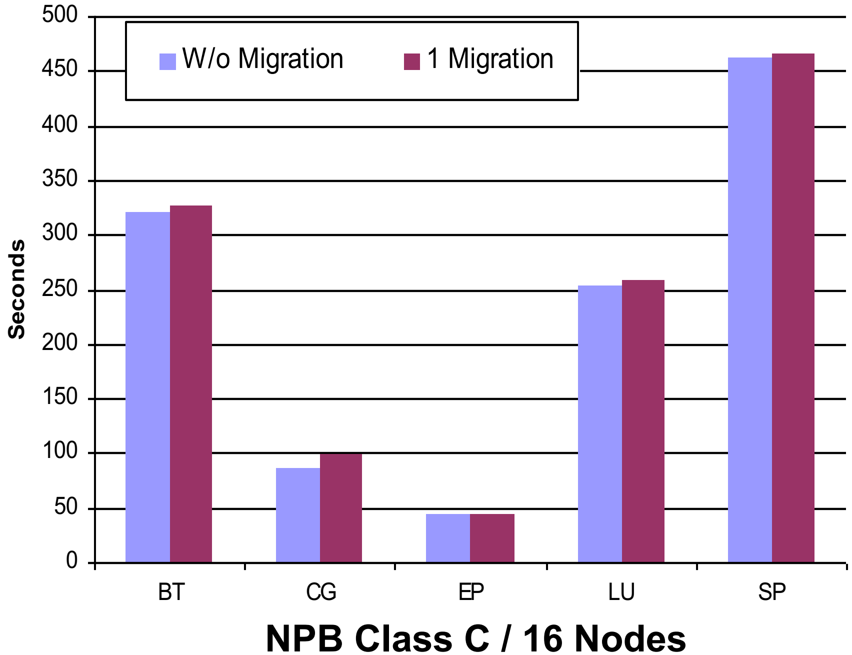


BMC Baseboard Management Contoller

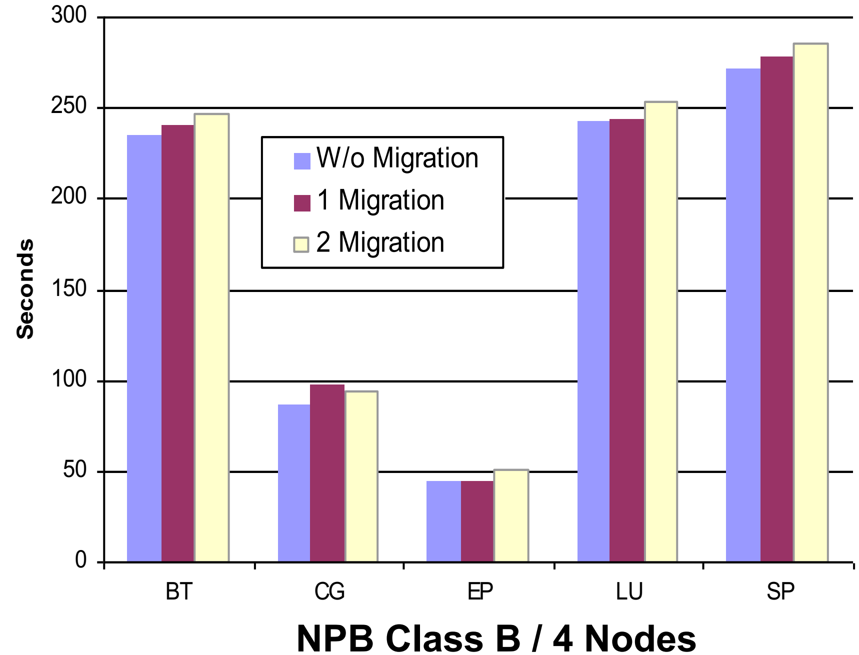
- Stand-by Xen host, no guest (spare node)
- Deteriorating health → migrate guest (w/ MPI app) to spare node
- Destination host generates unsolicited ARP reply
  - indicates Guest VM IP has moved to new location
  - ARP tells peers to resend packets to new host

Preemptive Migration Overhead

1. Single node failure



2. Double node failure

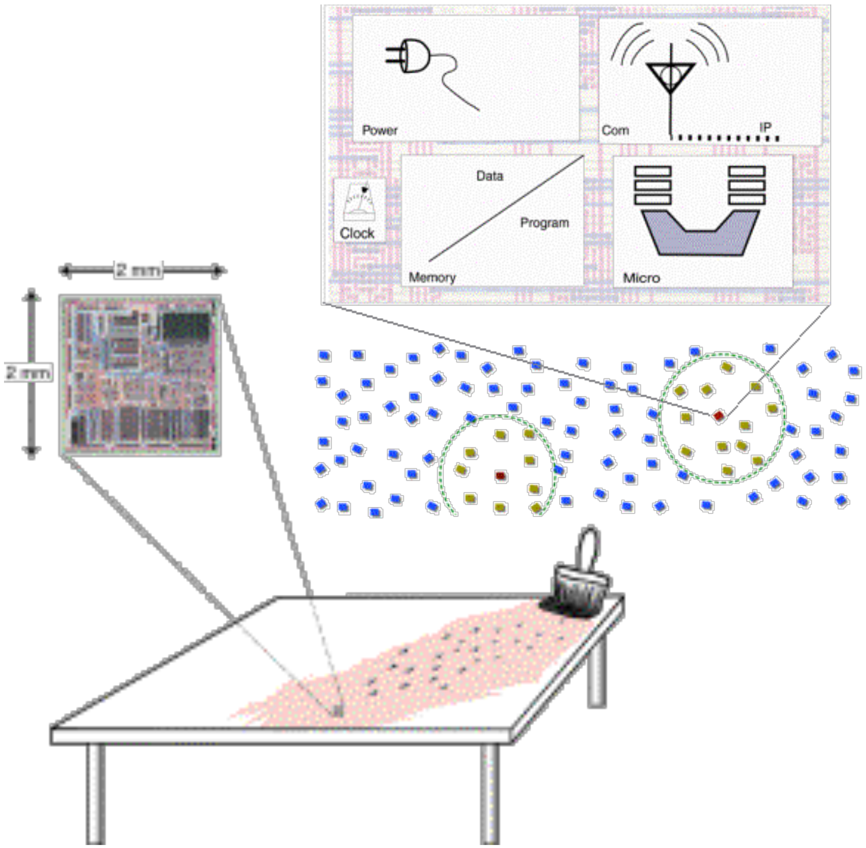


- Single node failure: 0.5-5% add'l cost over total wall clock time
- Double node failure: 2-8% add'l cost over total wall clock time

Algorithmic Fault Tolerance Approaches

- Naturally fault tolerant algorithms
  - Processes have only limited knowledge mostly about other processes in their neighborhood
  - Application is composed of local algorithms, where a failure has only a minor local impact
  - Examples: Chaotic relaxation, peer-to-peer communication
- Recovery & erasure codes
  - Reconstruction of lost information through algorithmic redundancy within the application
  - Rollback to consistent state through reverse computation

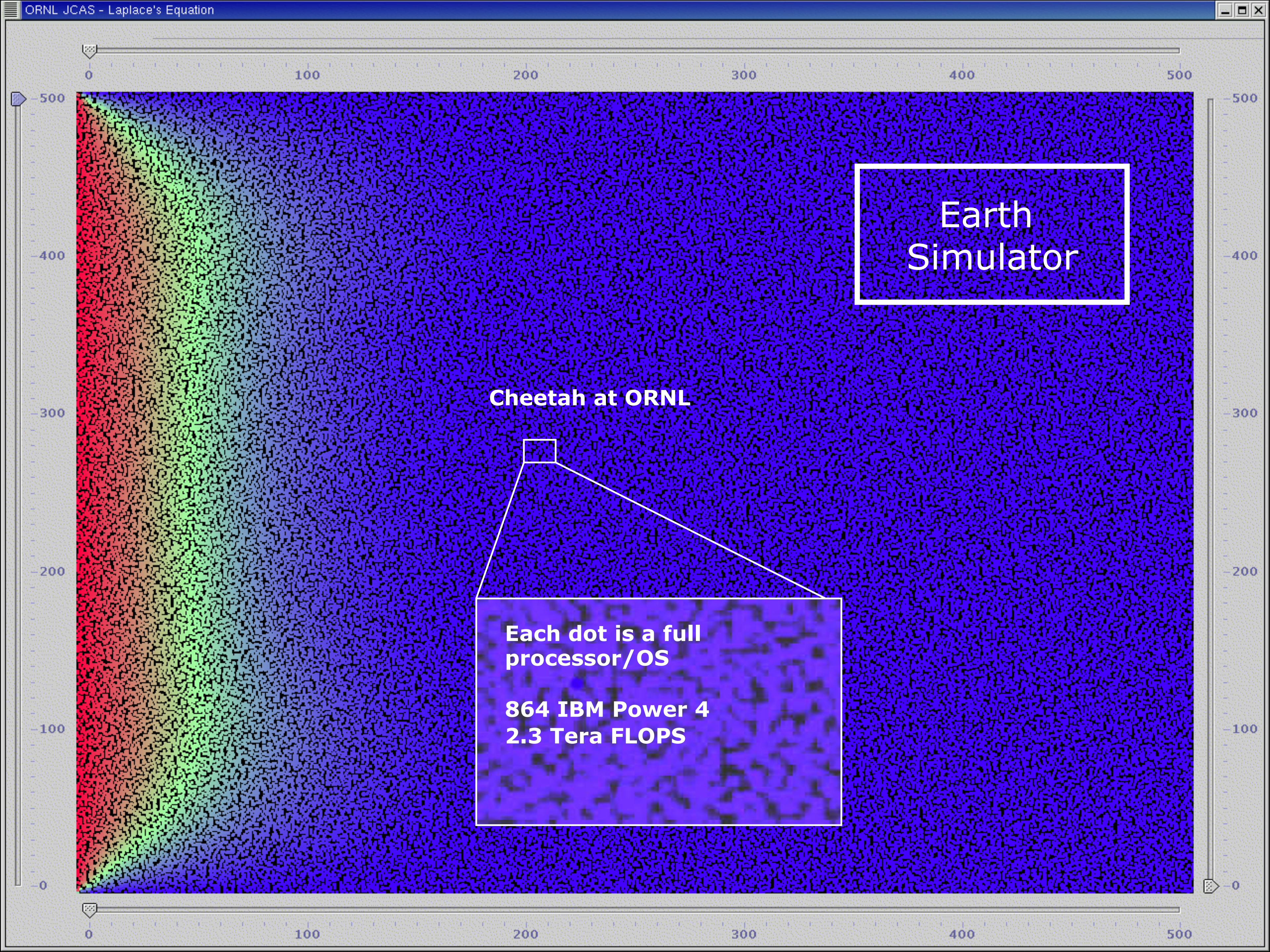
MIT Research: Paintable Computing



- In the future, embedded computers with a radio device will get as small as a paint pigment
- Supercomputers can be easily assembled by just painting a wall of embedded computers
- Applications are driven by cellular algorithms

Cellular Architecture (Smart Dust) Simulator

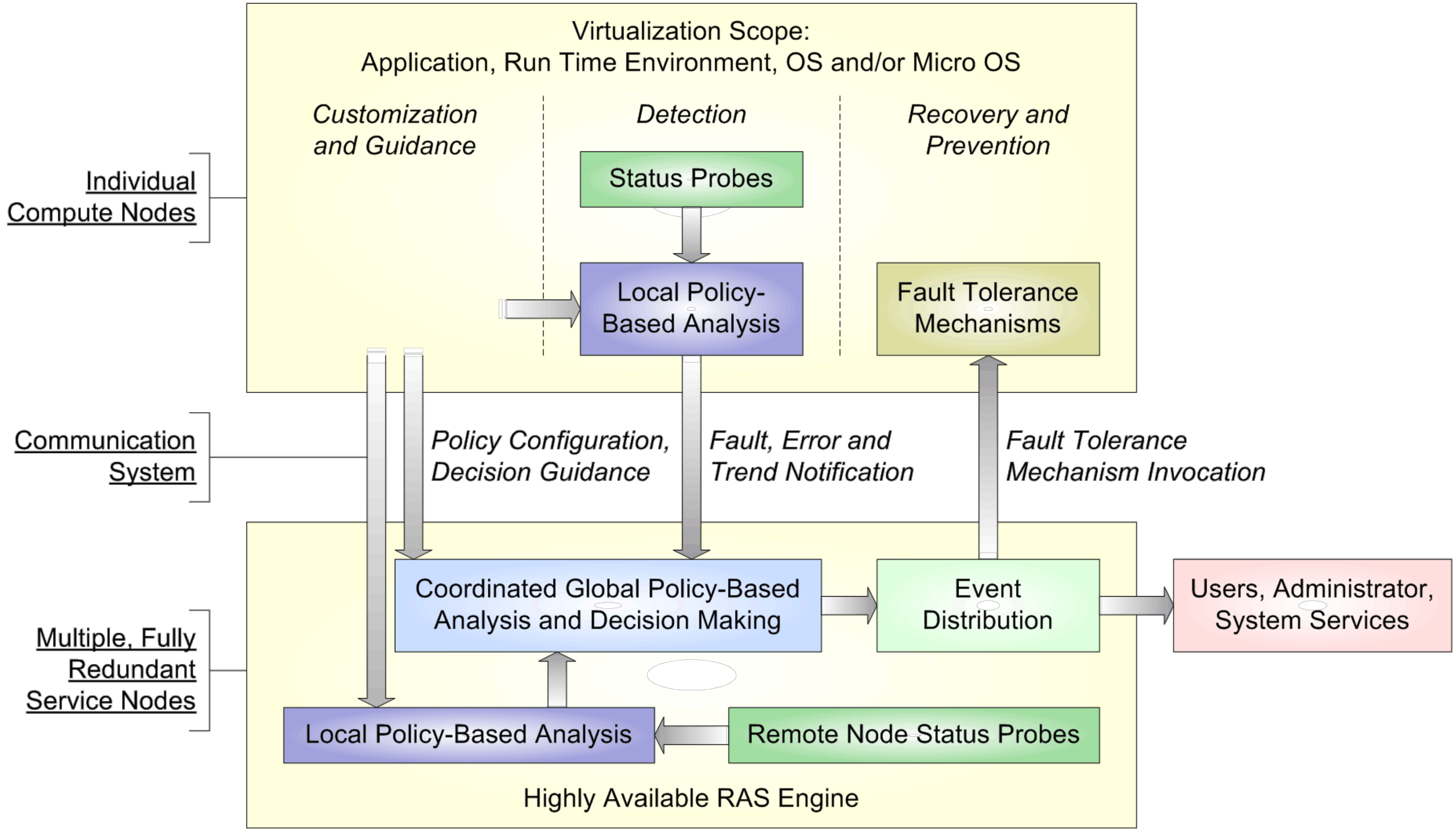
- Developed at ORNL in Java with native C and Fortran application support using JNI
- Runs as standalone or distributed application
- Lightweight framework simulates up to 1,000,000 lightweight processes on 9 real processors
- Standard and experimental networks:
  - Multi-dimensional mesh/torus
  - Nearest/Random neighbors
- Message driven simulation is not in real-time
- Primitive fault-tolerant MPI support


ORNL JCAS - Laplace's Equation
Earth Simulator
Cheetah at ORNL
Each dot is a full processor/OS
864 IBM Power 4
2.3 Tera FLOPS
0
100
200
300
400
500

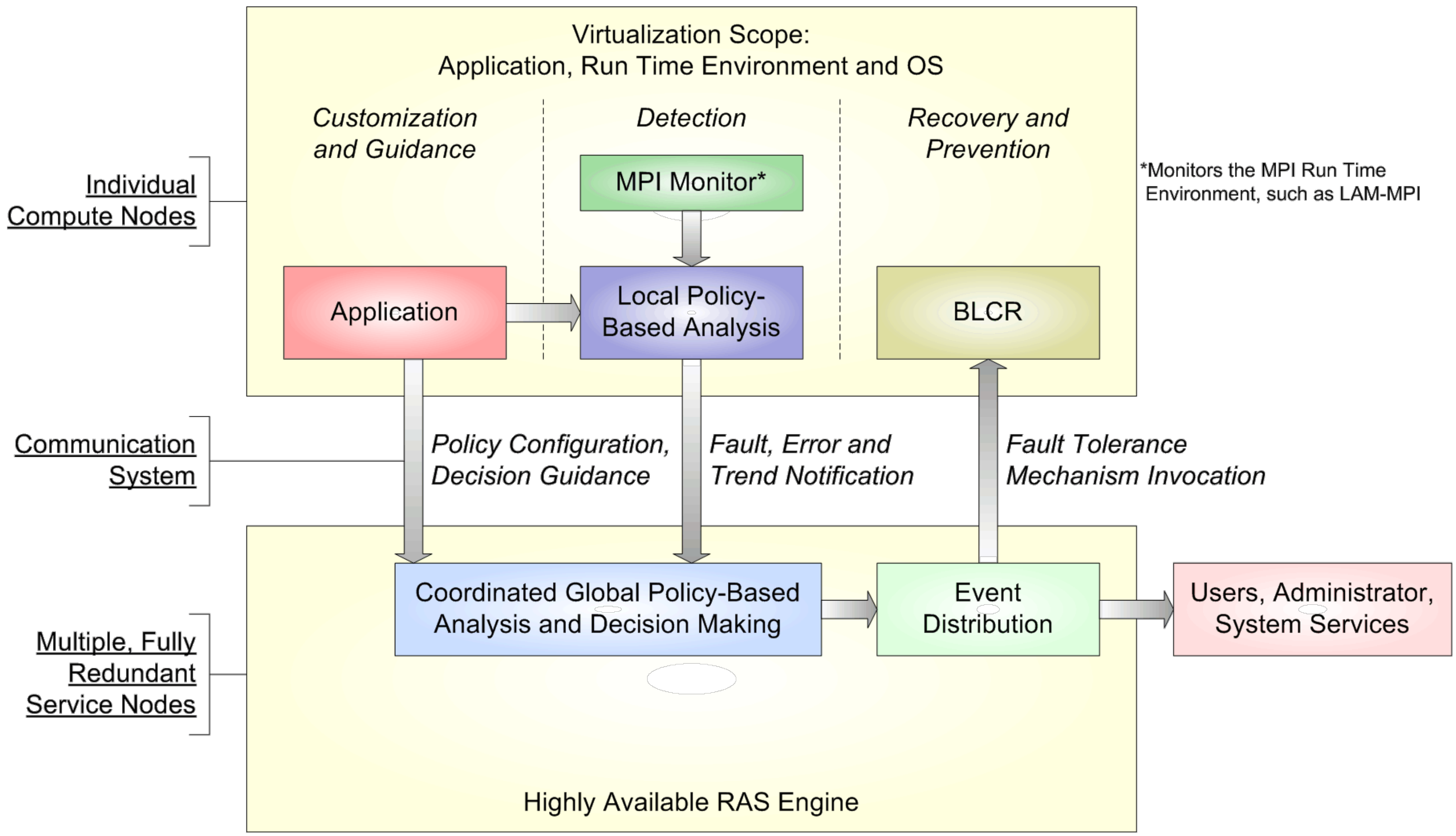
What's Next

- **Reliability analysis** for identifying pre-fault indicators, predicting failures, and modeling and monitoring individual system component reliability as well as overall system reliability
- **Proactive fault tolerance** technology based on preemptive migration of computation away from components that are about to fail using system-level virtualization in HPC environments
- **Reactive fault tolerance enhancements**, such as checkpoint interval and placement adaptation to actual and predicted system health threats, using system- and process-level virtualization
- **Holistic fault tolerance** through combination of adaptive proactive and reactive fault tolerance in conjunction with system health monitoring and reliability analysis

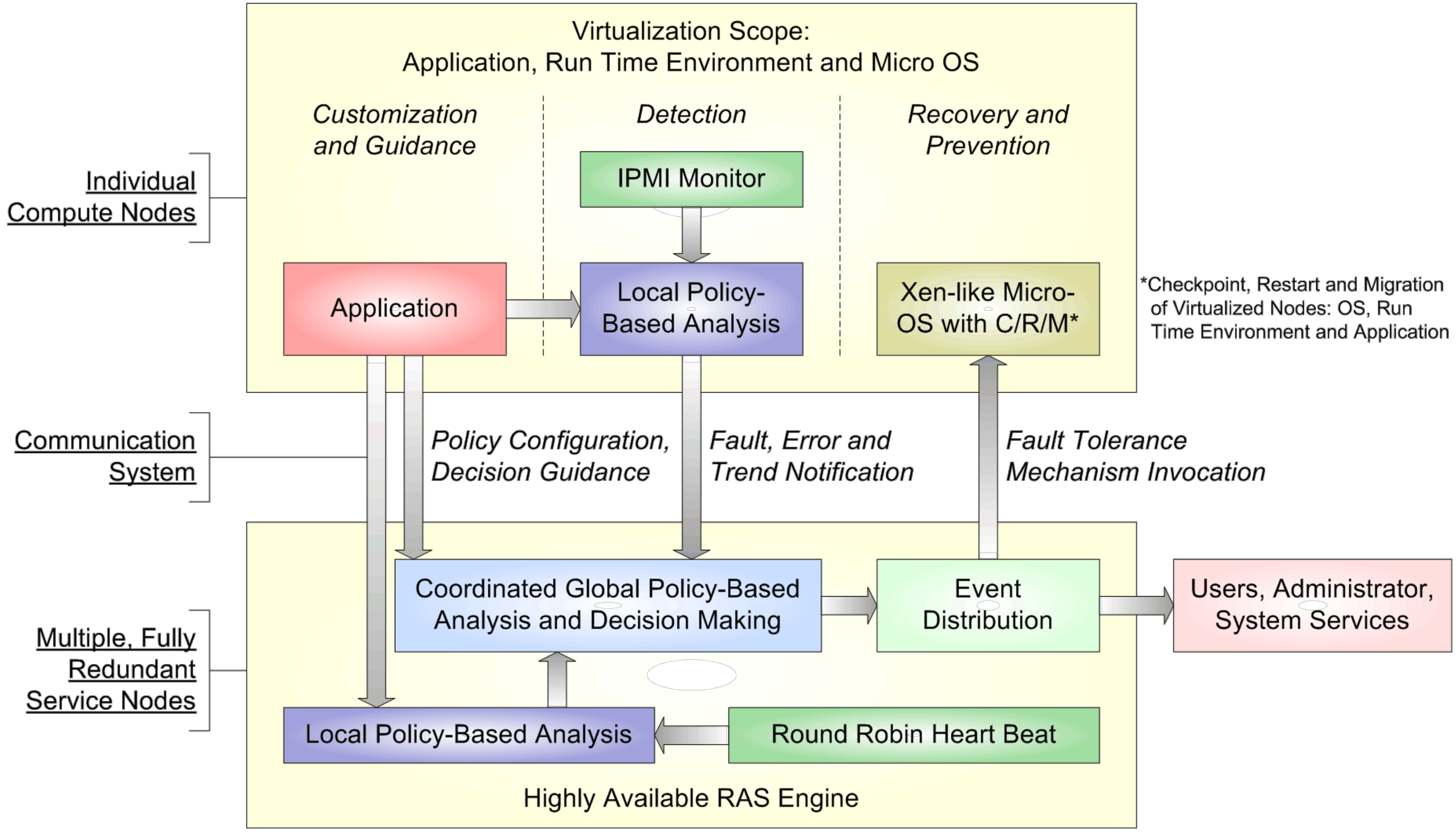
RAS Framework for Petascale HEC



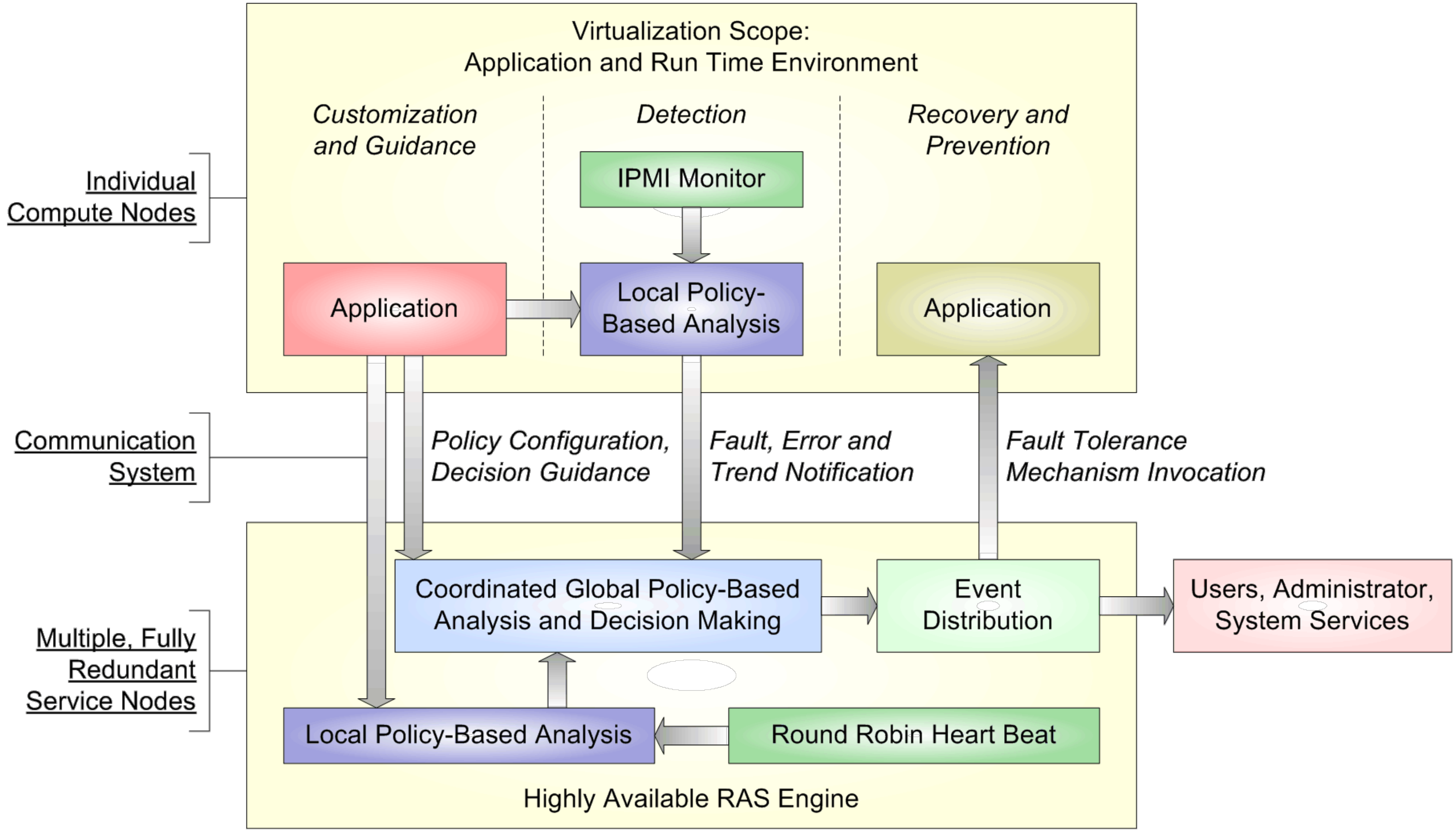
Example 1: Automatic Checkpoint/Restart



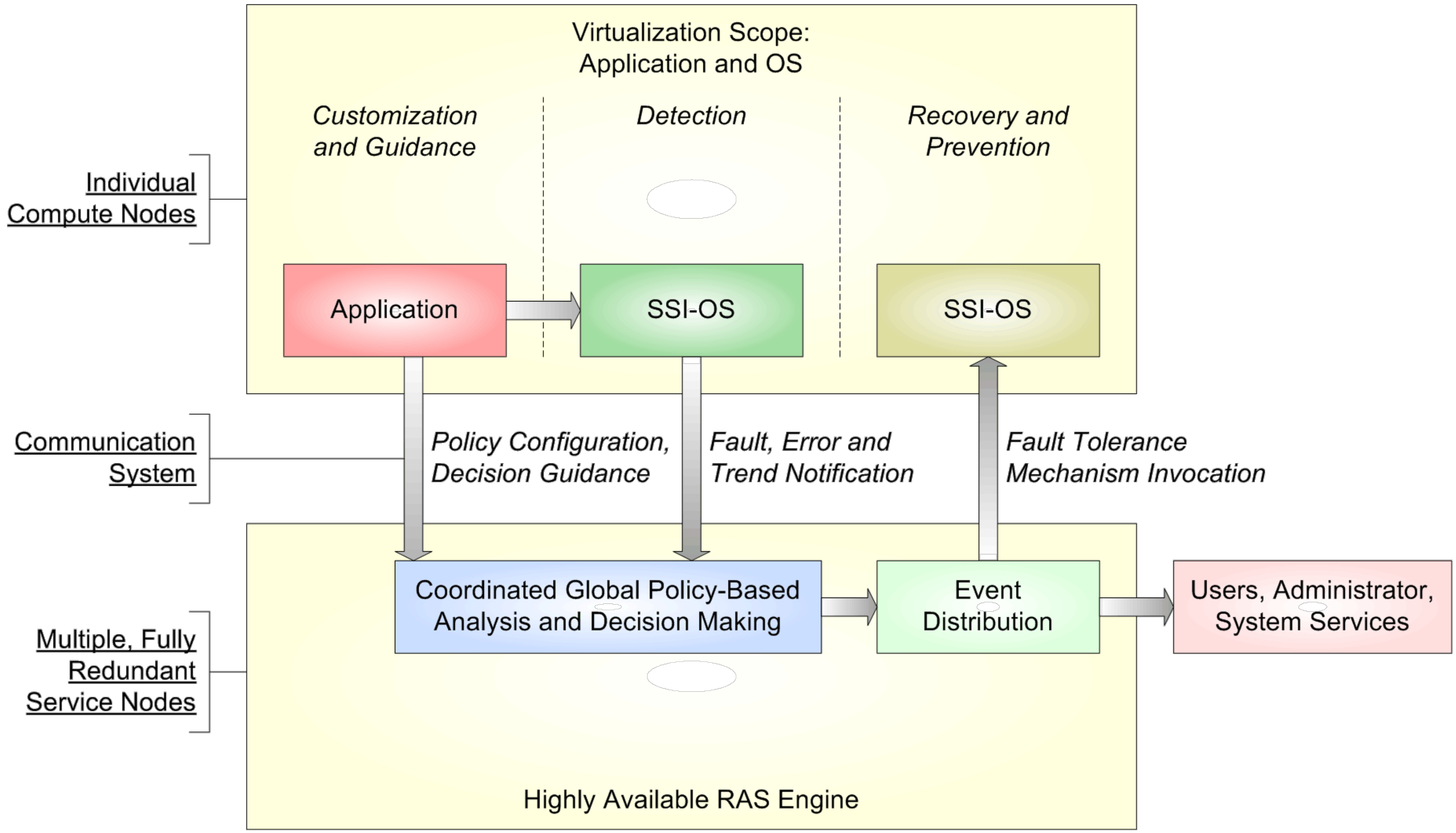
Example 2: Automatic Checkpoint/Restart/Migration



Example 3: Automatic Application-level Checkpoint/Restart



Example 4: OS-Supported Checkpoint/Restart/Migration



Summary and Conclusion

- Presented several traditional and advanced fault tolerance technologies for HPC
- Different scale requires different solutions:
  - Compute nodes
  - Front-end, service, and I/O nodes
- Scalable fault tolerance technologies are paramount to the success of large-scale HPC systems

Advanced Fault Tolerance Solutions for High Performance Computing

Christian Engelmann

Oak Ridge National Laboratory, Oak Ridge, USA
The University of Reading, Reading, UK