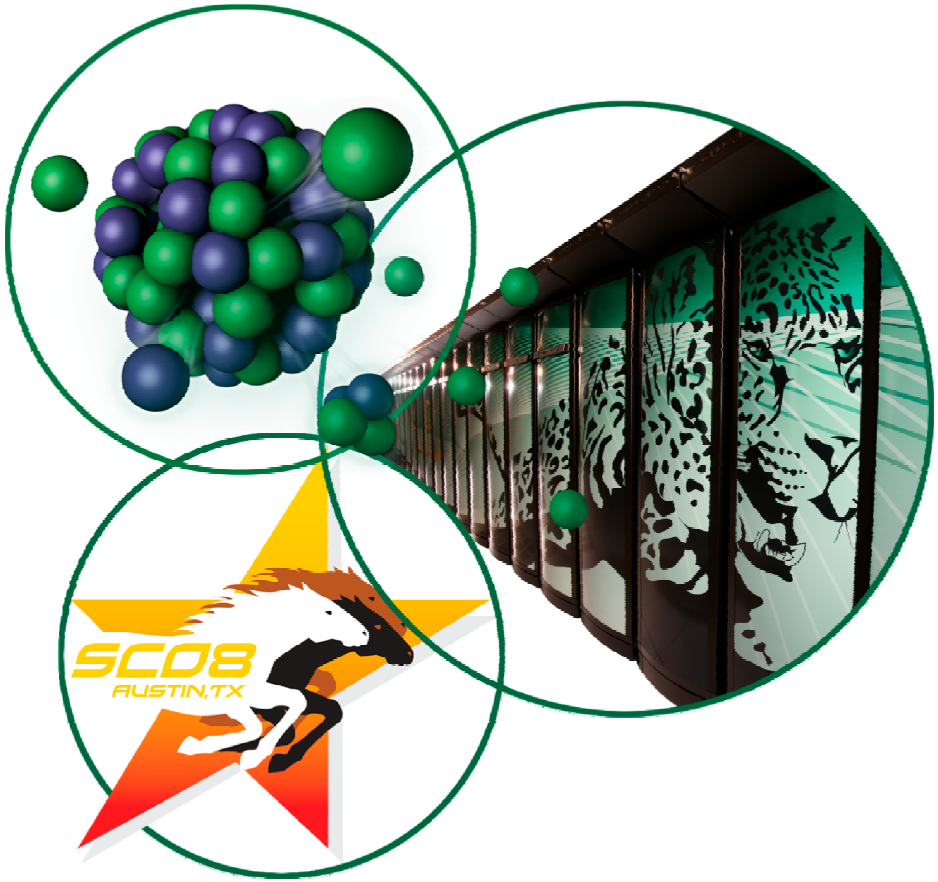
Reliability, Availability, and Serviceability (RAS) for High-Performance Computing (HPC)



Presented by

Stephen L. Scott
Christian Engelmann

Computer Science Research Group
Computer Science and Mathematics Division

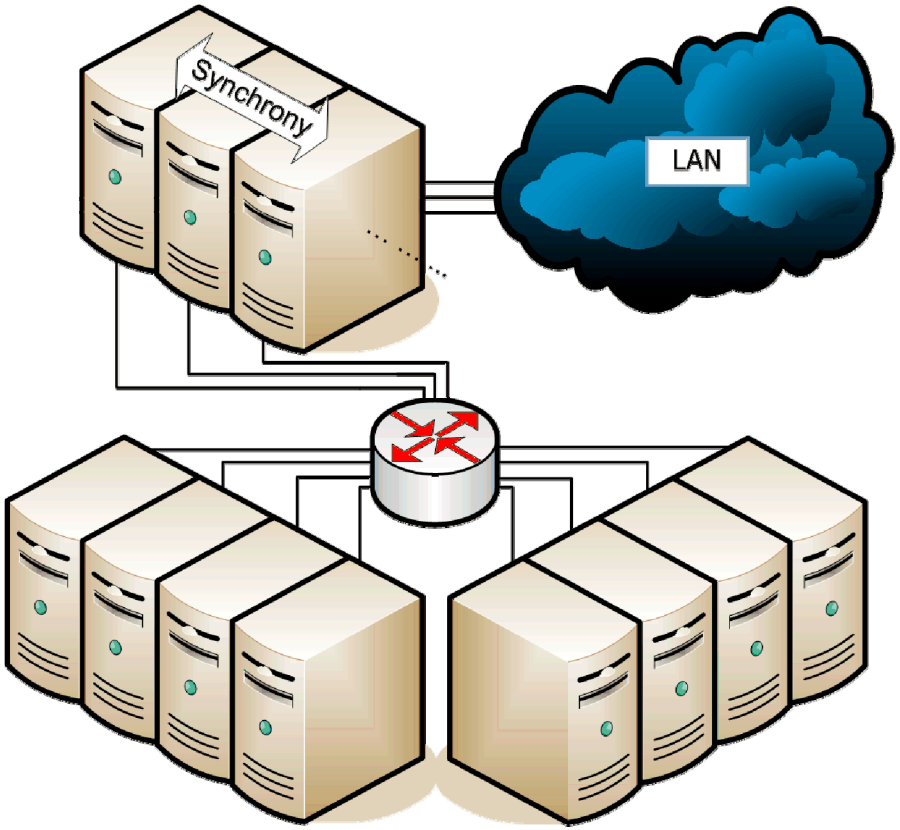
Oak Ridge National Laboratory

Research and development activities

- **Efficient redundancy strategies for HPC head/service nodes for high availability and high performance of critical services**
- **Reactive fault tolerance for HPC compute nodes utilizing the job pause approach and checkpoint placement adaptation**
- **Proactive fault tolerance using preemptive migration of computation away from compute nodes that are about to fail**
- **Reliability analysis for identifying pre-fault indicators, predicting failures, and modeling and monitoring reliability**
- **Holistic fault tolerance through combination of adaptive proactive and reactive fault tolerance mechanisms**

Symmetric active/active redundancy

Active/active head nodes

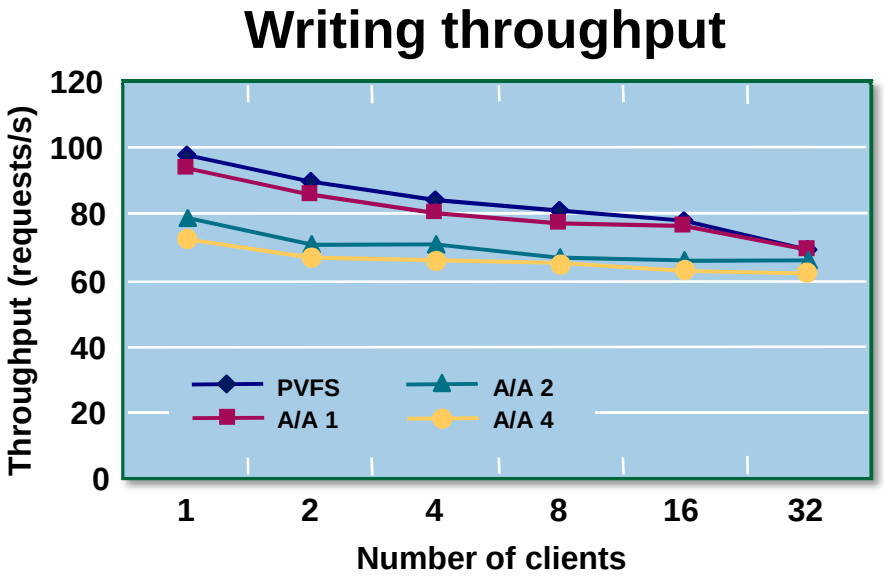


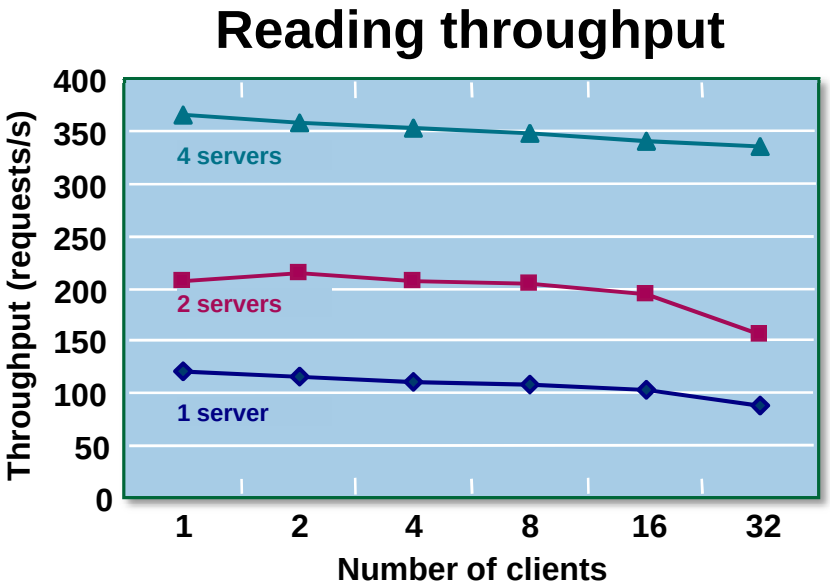
Compute nodes

- Many active head nodes
- Workload distribution
- Symmetric replication between head nodes
- Continuous service
- Always up to date
- No fail-over necessary
- No restore-over necessary
- Virtual synchrony model
- Complex algorithms

➡ **Prototypes for Torque and Parallel Virtual File System metadata server**

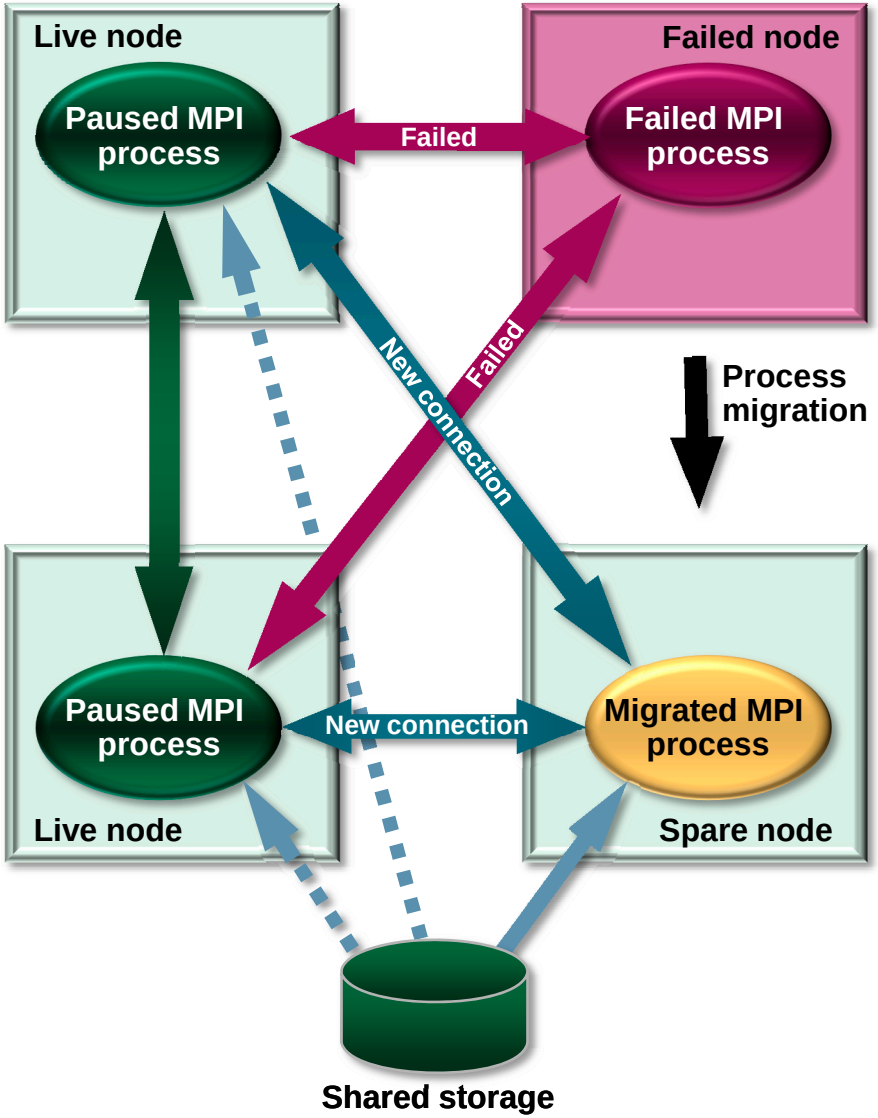
Symmetric active/active Parallel Virtual File System metadata server





Nodes	Availability	Est. annual downtime
1	98.58%	5d, 4h, 21m
2	99.97%	1h, 45m
3	99.9997%	1m, 30s

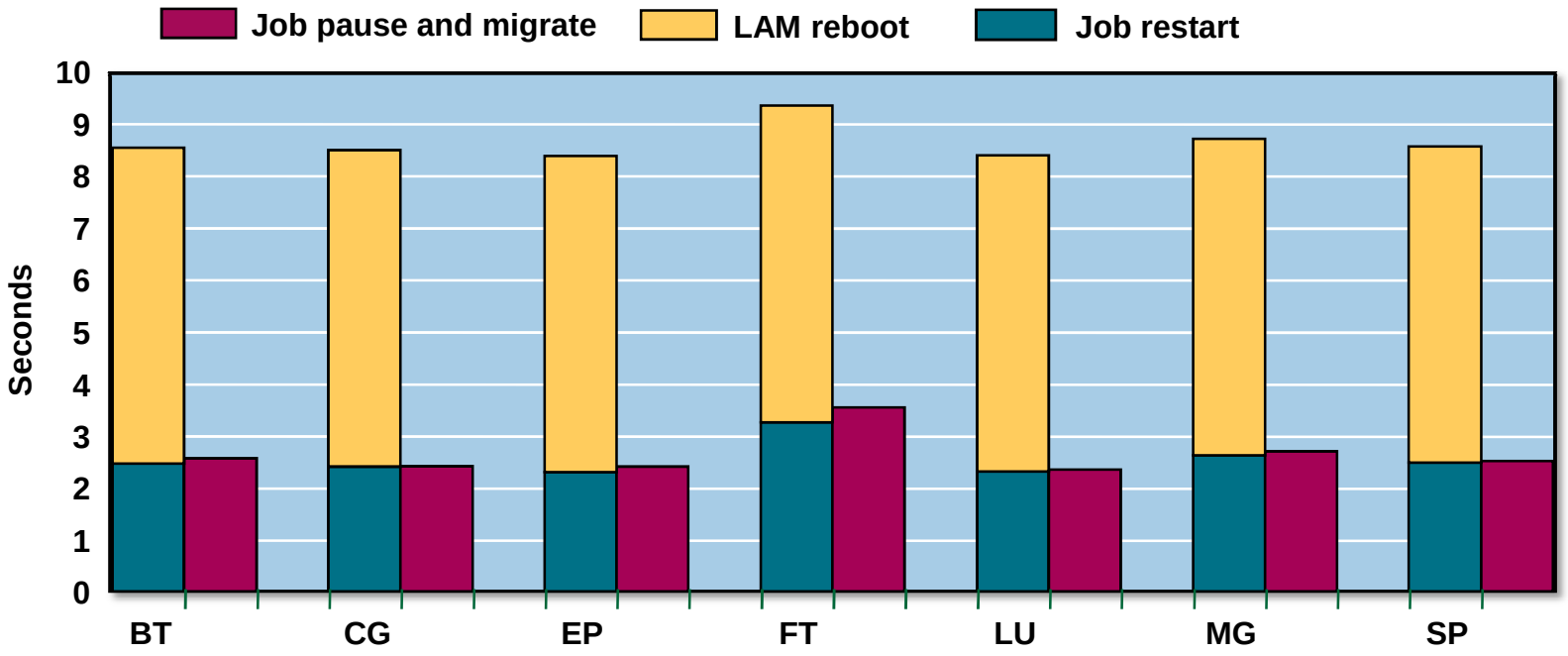
Reactive fault tolerance for HPC with LAM/MPI+BLCR job-pause mechanism



- Operational nodes: Pause
  - BLCR reuses existing processes
  - LAM/MPI reuses existing connections
  - Restore partial process state from checkpoint
- Failed nodes: Migrate
  - Restart process on new node from checkpoint
  - Reconnect with paused processes
- Scalable MPI membership management for low overhead

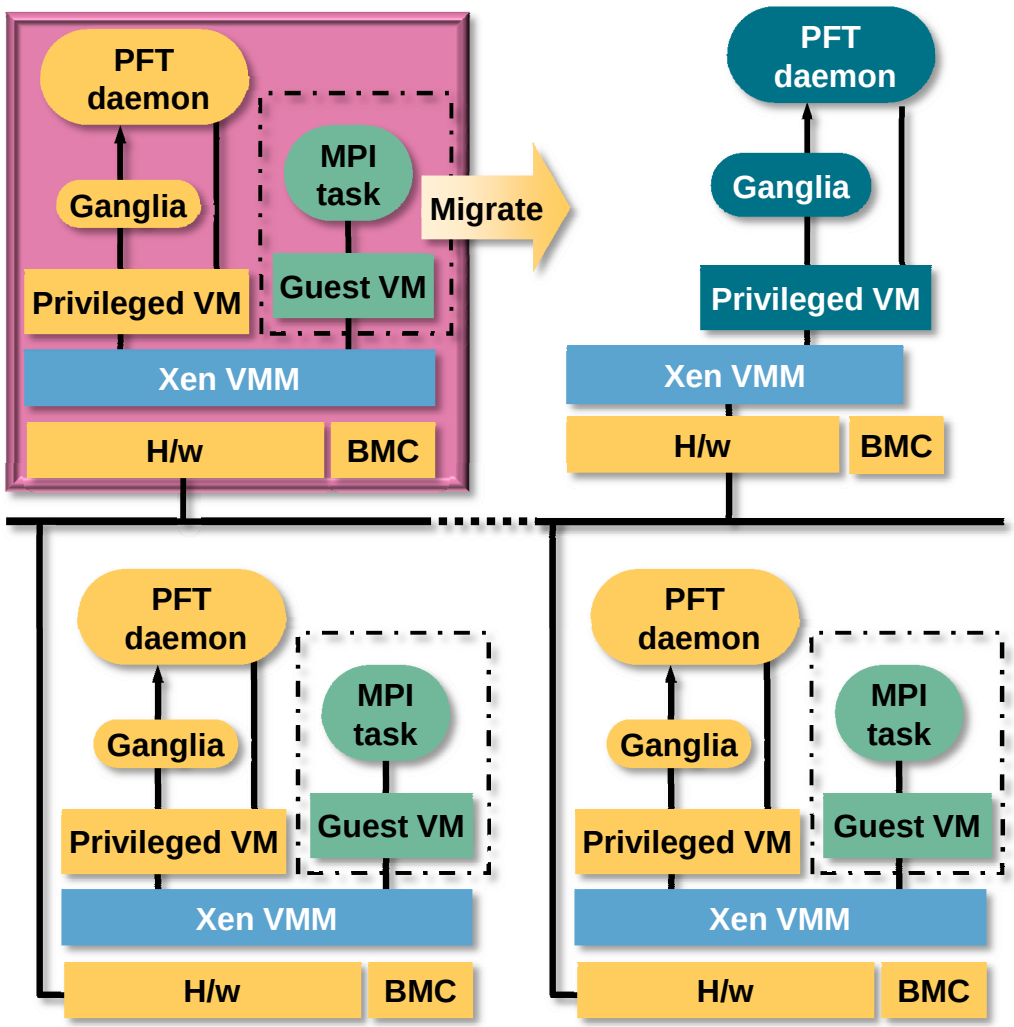
➡ **Efficient, transparent, and automatic failure recovery**

LAM/MPI+BLCR job pause performance



- 3.4% overhead over job restart, but
  - No LAM reboot overhead
  - Transparent continuation of execution
- No requeue penalty
- Less staging overhead

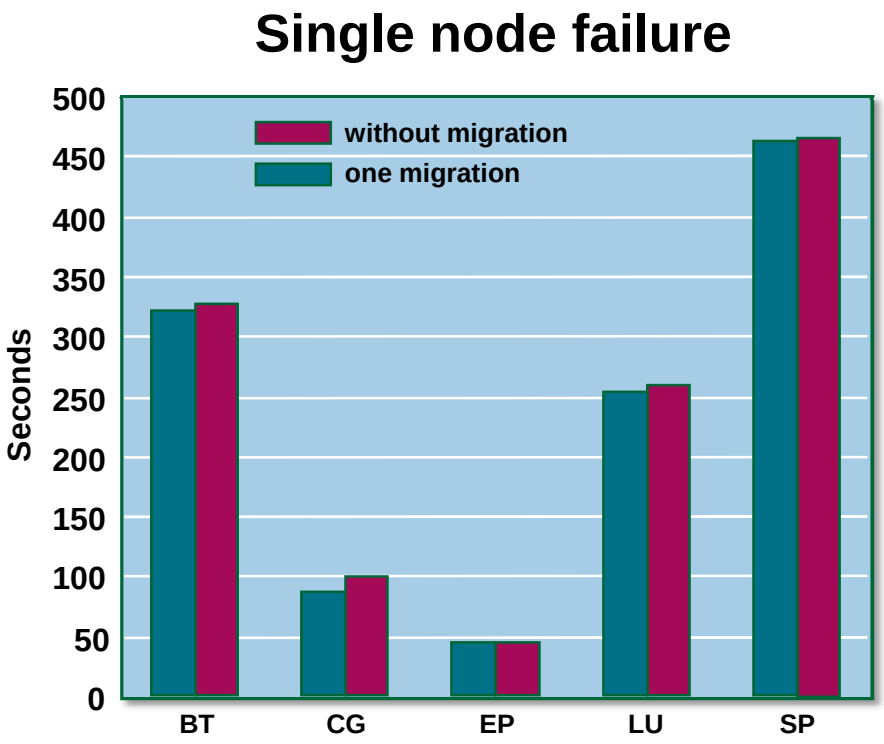
Proactive fault tolerance for HPC using Xen virtualization

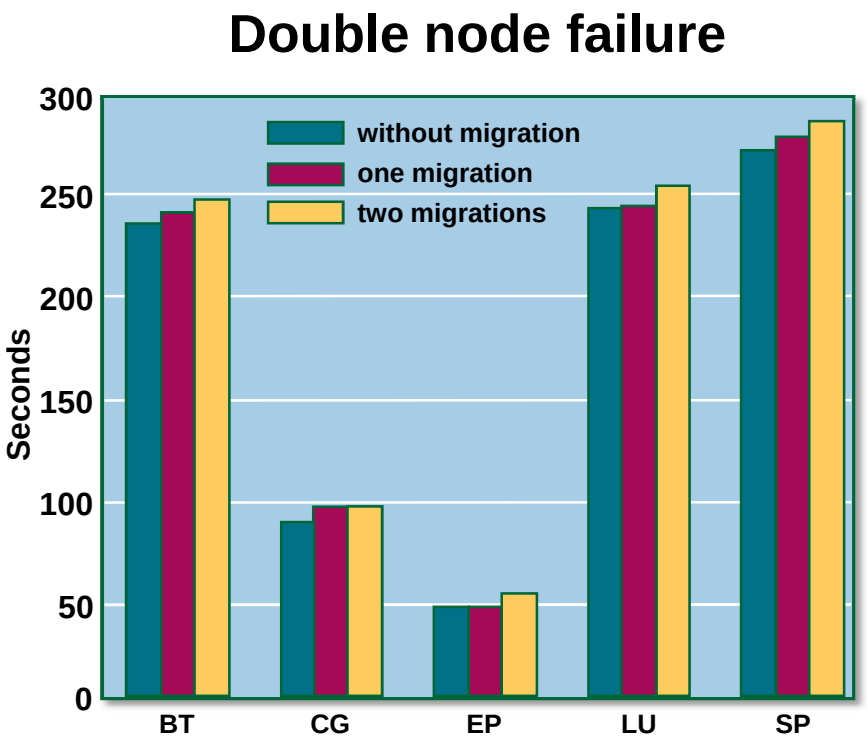


- Standby Xen host (spare node without guest VM)
- Deteriorating health
  - Migrate guest VM to spare node
- New host generates unsolicited ARP reply
  - Indicates that guest VM has moved
  - ARP tells peers to resend to new host

➡ **Novel fault-tolerance scheme that acts before a failure impacts a system**

VM migration performance impact

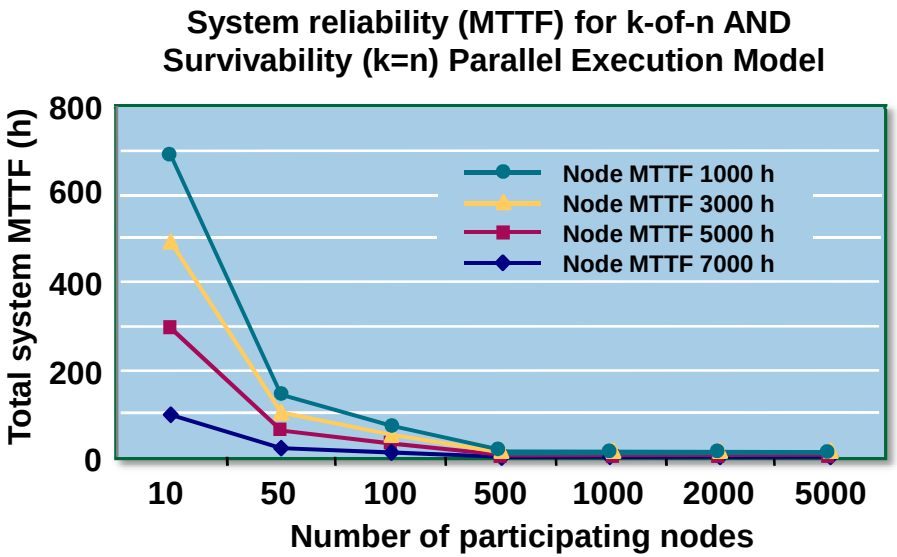


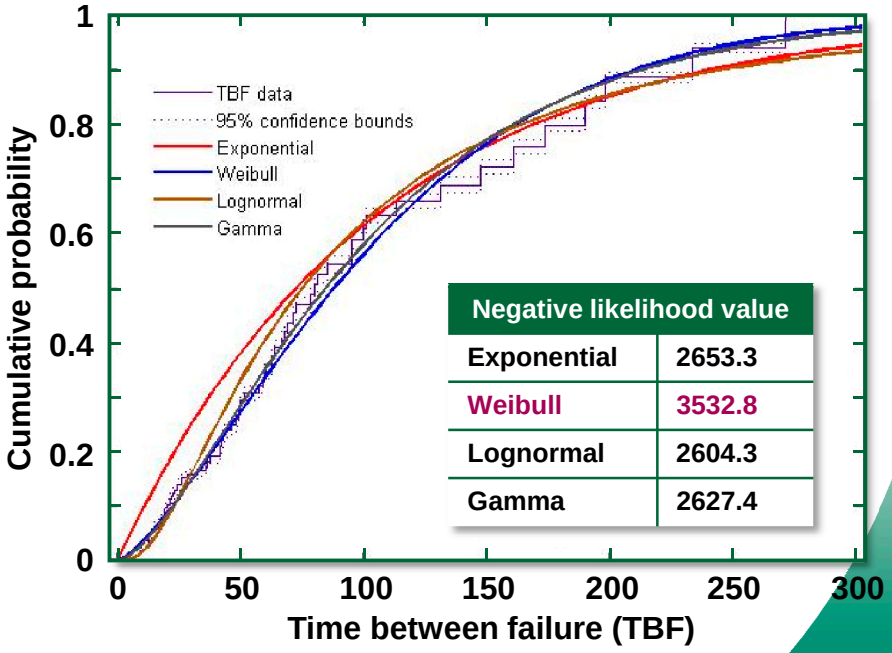


- **Single node failure: 0.5–5% additional cost over total wall clock time**
- **Double node failure: 2–8% additional cost over total wall clock time**

HPC reliability analysis and modeling

- Programming paradigm and system scale impact reliability
- Reliability analysis
- Estimate mean time to failure (MTTF)
- Obtain failure distribution: Exponential, Weibull, gamma, etc.
- Feedback into fault-tolerance schemes for adaptation

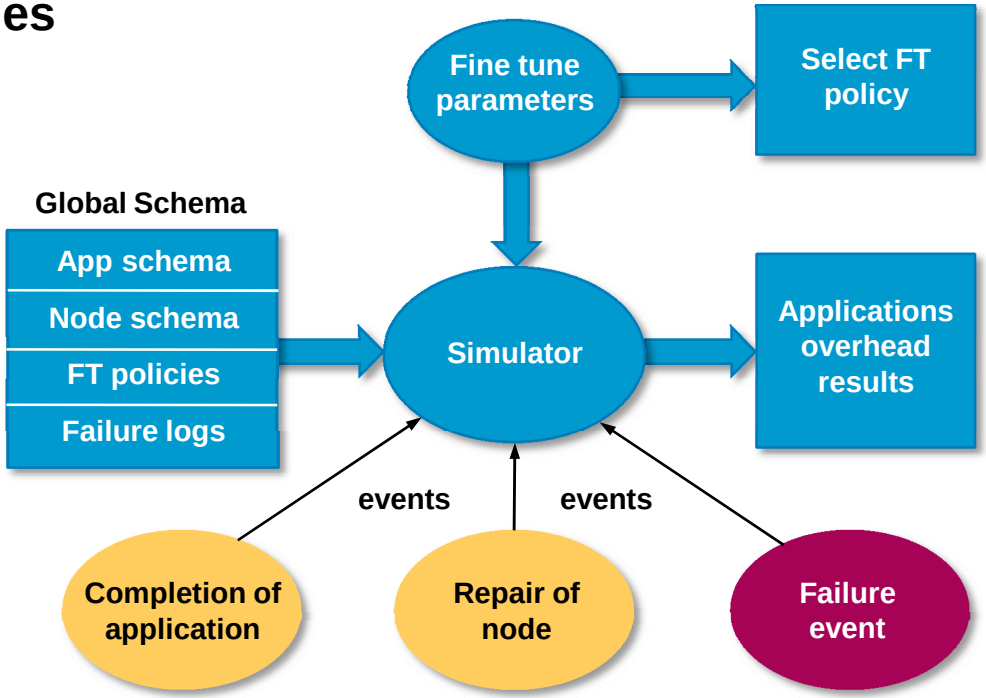




Negative likelihood value	
Exponential	2653.3
Weibull	3532.8
Lognormal	2604.3
Gamma	2627.4

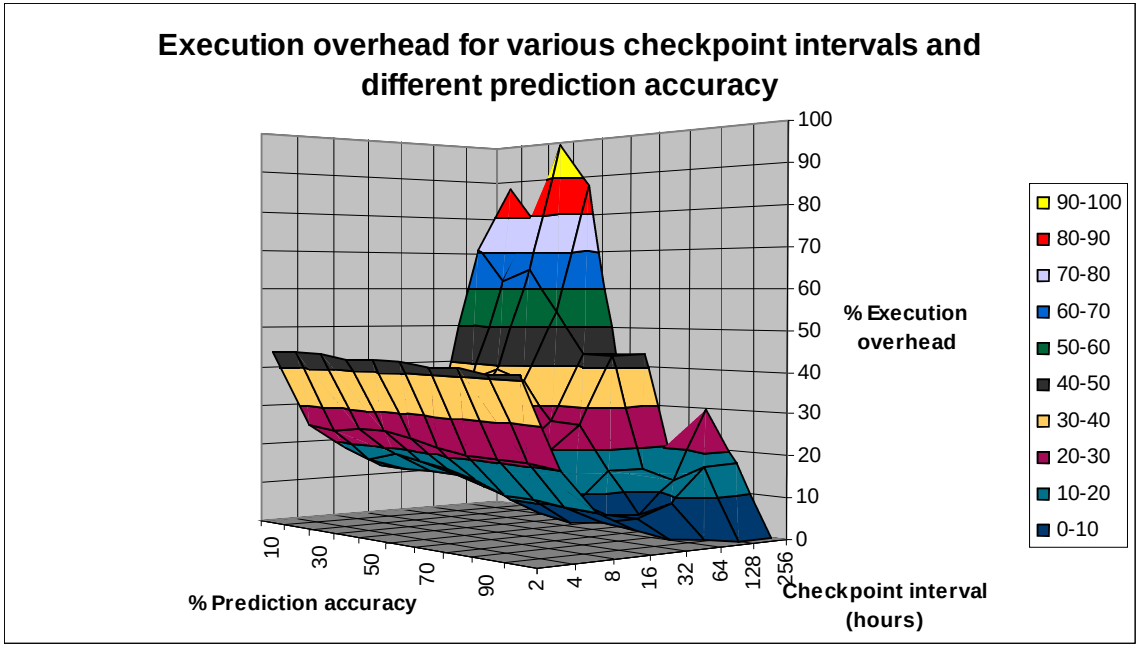
Simulation framework for HPC fault-tolerance policies

- 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



Oak Ridge National Laboratory

Combination of proactive and reactive fault tolerance: Simulation example



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

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

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

Contacts for HPC RAS research

Stephen L. Scott

Computer Science Research Group
Computer Science and Mathematics Division
(865) 574-3144
scottsl@ornl.ornl

Christian Engelmann

Computer Science Research Group
Computer Science and Mathematics Division
(865) 574-3132
engelmannc@ornl.ornl

www.fastos.org/ras