# Beyond Application-Level Checkpoint Restart – Advanced Software Approaches for Fault Resilience

**Christian Engelmann**

**Computer Science Research Group**
**Computer Science and Mathematics Division**
**Oak Ridge National Laboratory, USA**

# Extreme-Scale High-Performance Computing Systems for Computational Science

top500.org processor count:

- About three years ago the entire 500 list broke the million processor mark
- Now the top 7 add up to over a million

#1

# 4

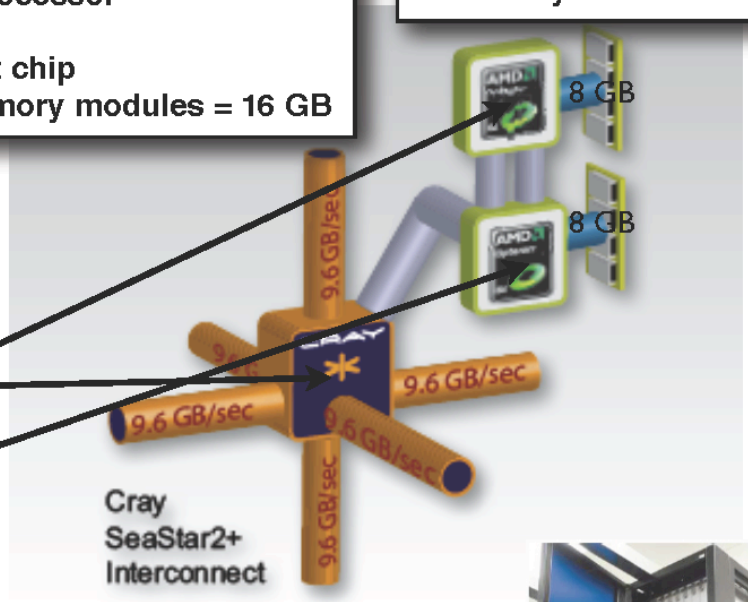
# #1: Jaguar at Oak Ridge National Laboratory

**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




**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

# Motivation

- Large-scale 1 PFlop/s systems are here
  - #1 ORNL Jaguar XT5: 1.759 PFlop/s, 224,162 cores
  - #2 NSCS Nebulae: 1.271 PFlop/s, 120,640 cores
  - #3 LANL Roadrunner: 1.042 PFlop/s, 122,400 cores
- Other large-scale systems exist
  - #4 NICS Kraken XT5: 0.831 PFlop/s, 98,928 cores
  - #5 Juelich JUGENE: 0.825 PFlop/s, 294,912 cores
  - #6 NASA Pleiades: 0.773 PFlop/s, 81,920 cores
- The trend is toward even larger-scale systems
  - End of processor frequency scaling ➔ Node/core scaling

# Proposed Exascale Initiative Road Map

| Systems | 2009 | 2011 | 2015 | 2018 |
|---|---|---|---|---|
| System peak | 2 Peta | 20 Peta | 100-200 Peta | 1 Exa |
| System memory | 0.3 PB | 1.6 PB | 5 PB | 10 PB |
| Node performance | 125 GF | 200GF | 200-400 GF | 1-10TF |
| Node memory BW | 25 GB/s | 40 GB/s | 100 GB/s | 200-400 GB/s |
| Node concurrency | 12 | 32 | O(100) | O(1000) |
| Interconnect BW | 1.5 GB/s | 22 GB/s | 25 GB/s | 50 GB/s |
| System size (nodes) | 18,700 | 100,000 | 500,000 | O(million) |
| Total concurrency | 225,000 | 3,200,000 | O(50,000,000) | O(billion) |
| Storage | 15 PB | 30 PB | 150 PB | 300 PB |
| IO | 0.2 TB/s | 2 TB/s | 10 TB/s | 20 TB/s |
| MTTI | days | days | days | O(1 day) |
| Power | 6 MW | ~10MW | ~10 MW | ~20 MW |

# My Exascale Resilience Scenario: MTTI Scales with Node Count

| Systems | 2009 | 2011 | 2015 | 2018 |
|---|---|---|---|---|
| System peak | 2 Peta | 20 Peta | 100-200 Peta | 1 Exa |
| System size (nodes) | | 5x | 5x | 2x |
| MTTI | 4 days | 19 h 4 min | 3 h 52 min | 1 h 56 min |

***Vendors are able to maintain current node MTTI***

# My Scary Scenario: Current MTTI of 1 Day

| Systems | 2009 | 2011 | 2015 | 2018 |
|---|---|---|---|---|
| System peak | 2 Peta | 20 Peta | 100-200 Peta | 1 Exa |
| System size (nodes) | | 5x | 5x | 2x |
| MTTI | 1 day | 4 h 48 min | 58 min | 29 min |

***Current system MTTI is actually lower***

# My Really Scary Scenario: Component MTTI drops 3% Each Year

| Systems | 2009 | 2011 | 2015 | 2018 |
|---|---|---|---|---|
| System peak | 2 Peta | 20 Peta | 100-200 Peta | 1 Exa |
| System size (nodes) | | 5x | 5x | 2x |
| MTTI | 1 day | 4 h 31 min | 48 min | 22 min |

***Vendors are not able to maintain current node MTTI***

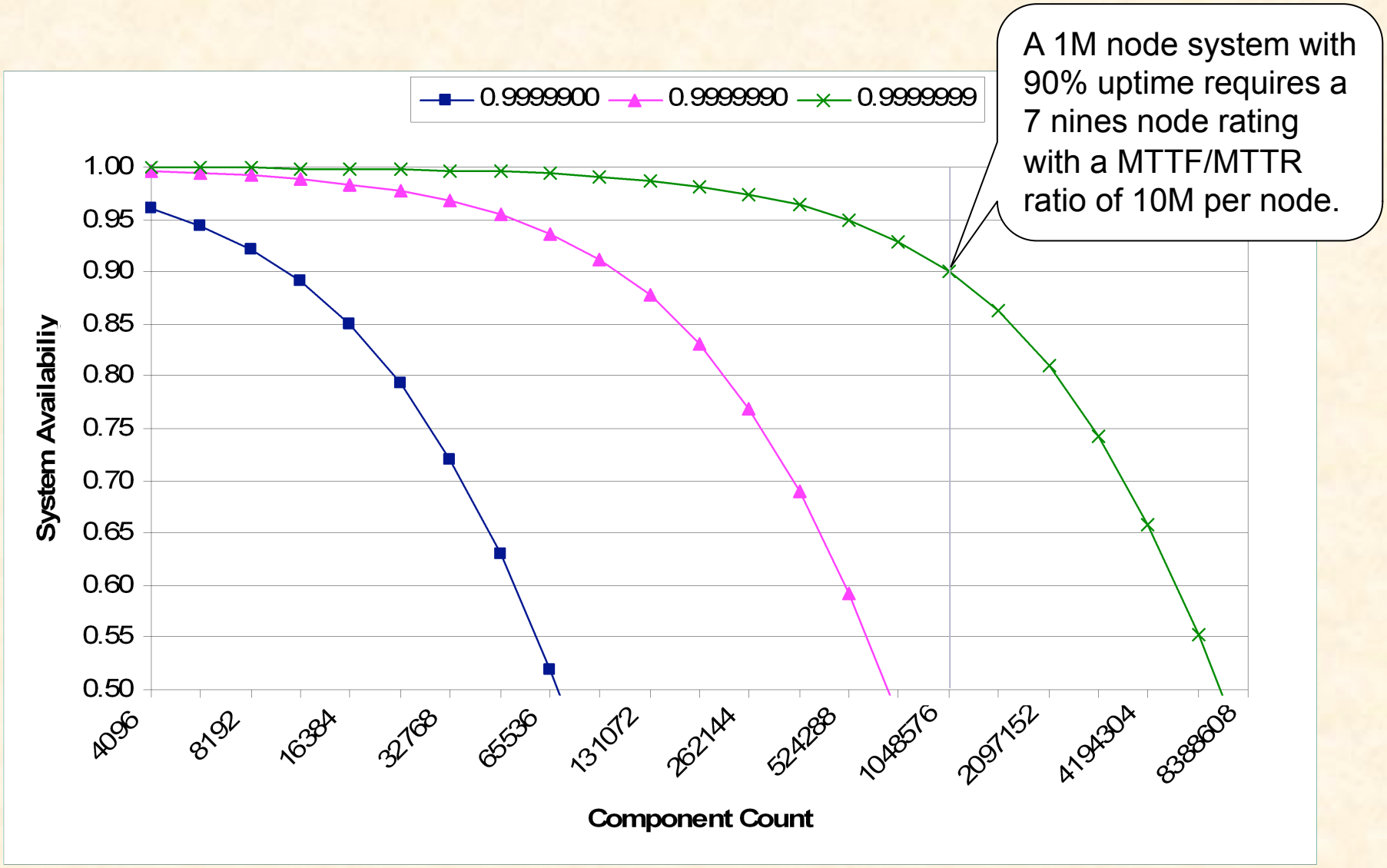
# Resilience Issues in Extreme-scale HPC

- Significant growth in component count (up to 50x nodes expected) results in correspondingly higher error rate
- Smaller circuit sizes and lower voltages increase soft error vulnerability (bit flips caused by thermal and voltage variations as well as radiation)
- Hardware fault detection and recovery is limited by power consumption requirements and production costs
- Heterogeneous architectures (CPU & GPU cores) add more complexity to fault detection and recovery
- Power management cycling decreases component lifetimes due to thermal and mechanical stresses

# Risks of the Business as Usual Approach

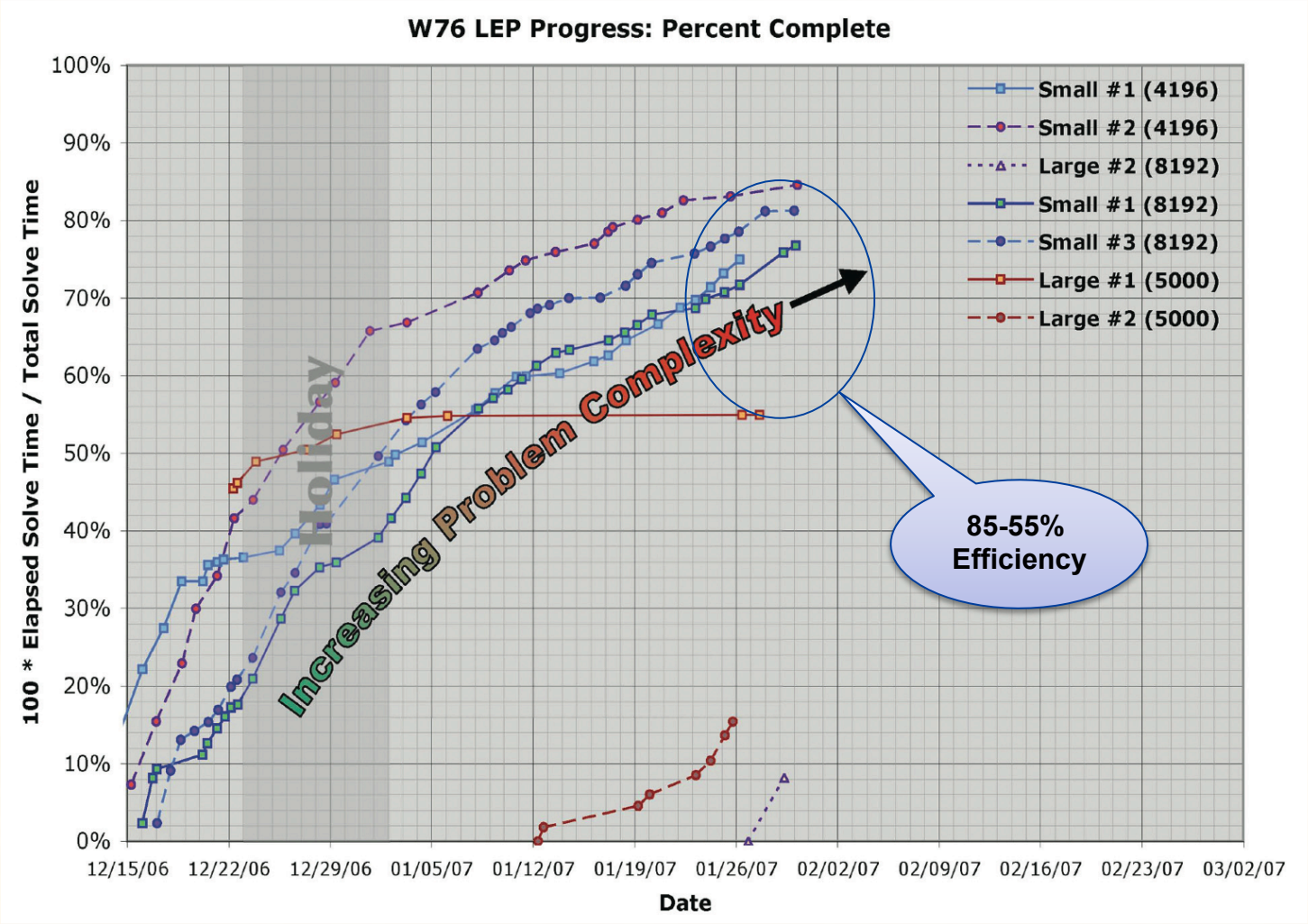
- Increased error rate requires more frequent checkpoint/ restart, thus lowering efficiency (application progress)
- Memory to I/O ratio improves due to less memory/node, but concurrency for coordination and scheduling increases significantly (up to 50x nodes, 444x cores)
- Current application-level checkpoint/restart to a parallel file system is becoming less efficient and soon obsolete
- Missing strategy for silent data/code corruption will cause applications to produce erroneous results or hang

# System Availability with Checkpoint/Restart



# 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.



# Advanced Software Approaches for Fault Resilience

- System software solutions
  - Transparent incremental checkpointing
  - Transparent checkpoint storage through virtualization
  - Transparent proactive fault tolerance using migration
  - Transparent redundant execution (DMR and TMR)
- Application solutions
  - Cellular algorithms: Naturally fault tolerant and scalable
  - Application-architecture co-design for performance/resilience
  - Interesting work by others: Fault-tolerant algorithms

# Incremental Checkpointing with BLCR

- Recent enhancement for Berkeley Lab Checkpoint/ Restart (BLCR)
- Track differences with dirty bit at PTE
- Hybrid: 1 full and *k* incremental checkpoints
- Available through BLCR distribution
- Expected integration into Open MPI+BLCR

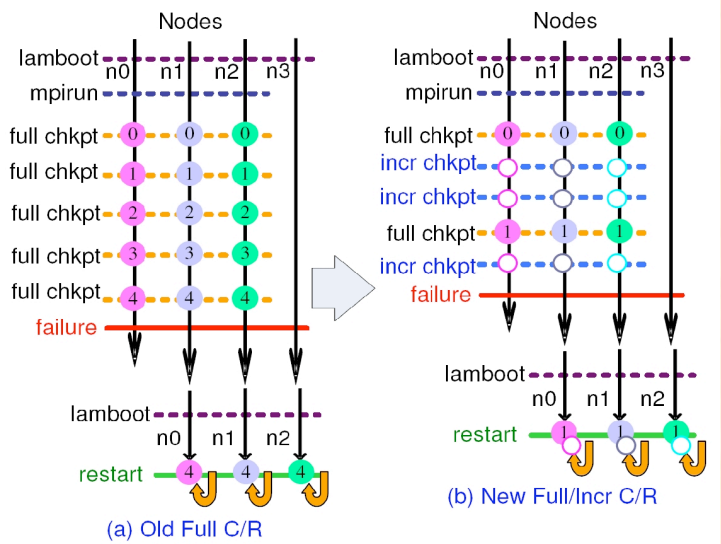
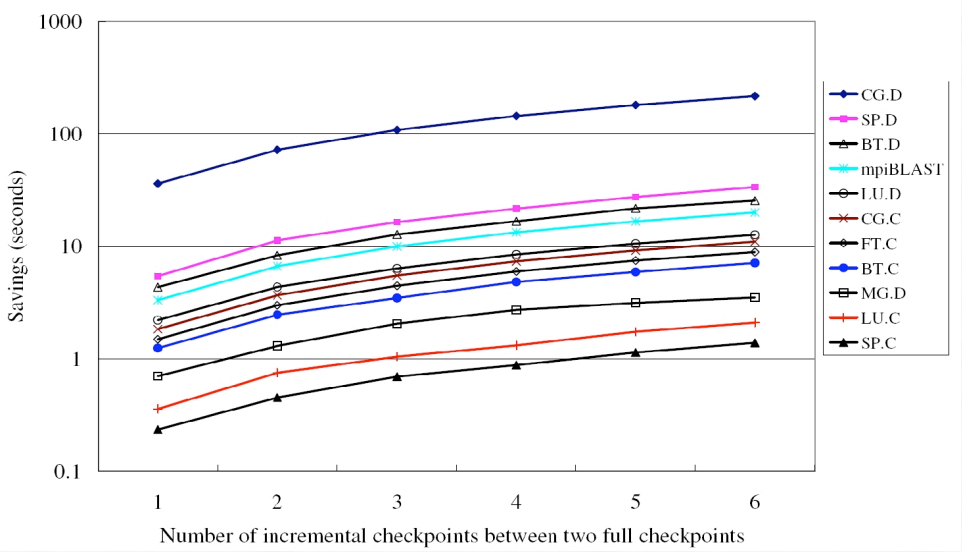



Fig. 1: Hybrid Full/Incremental C/R Mechanism vs. Full C/R

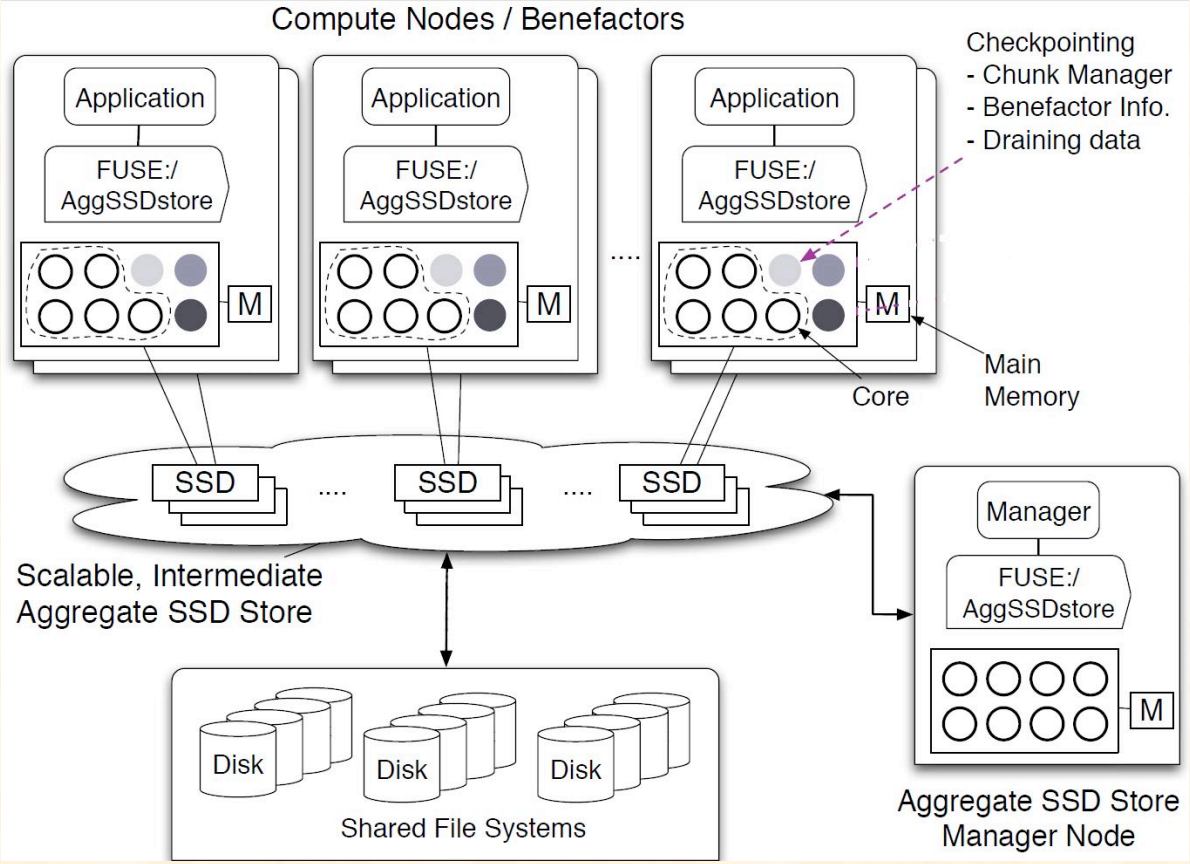


# Checkpoint Storage Virtualization

- Developed a checkpoint storage system that aggregates compute and service node resources (memory or SSD)
- Built a transparent file system interface (FUSE) to mount the aggregated checkpoint storage on compute nodes
- The distributed storage supports many scenarios:
  - Replicated compute-node checkpoint storage
  - Replicated compute-node checkpoint cache
  - Where each compute node has more memory or SSD
  - Where a special partition has more memory or SSD
  - Where the I/O nodes have more memory or SSD

# Checkpoint Storage Architecture

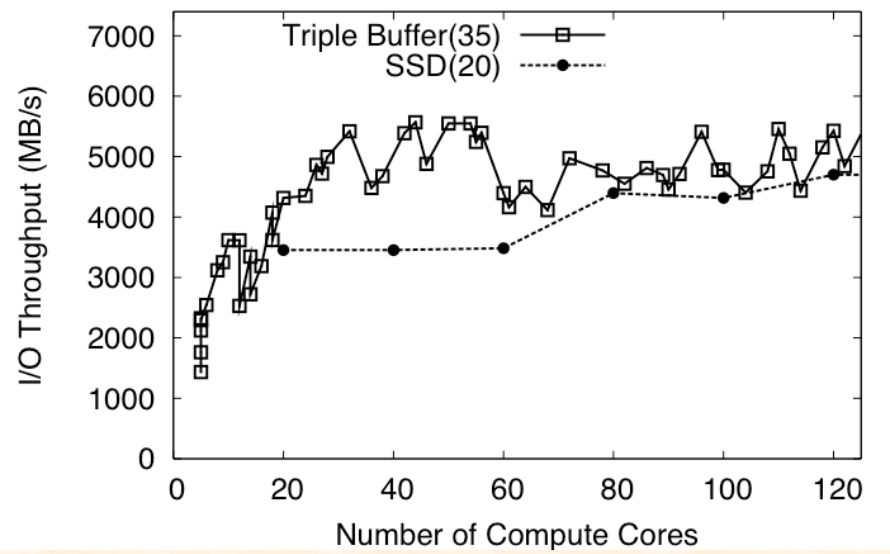
- Each contributing node runs a benefactor process that offers available SSD or memory space
- A manager maintains all metadata
- A FUSE client hides the client-manager-benefactor interaction from apps.

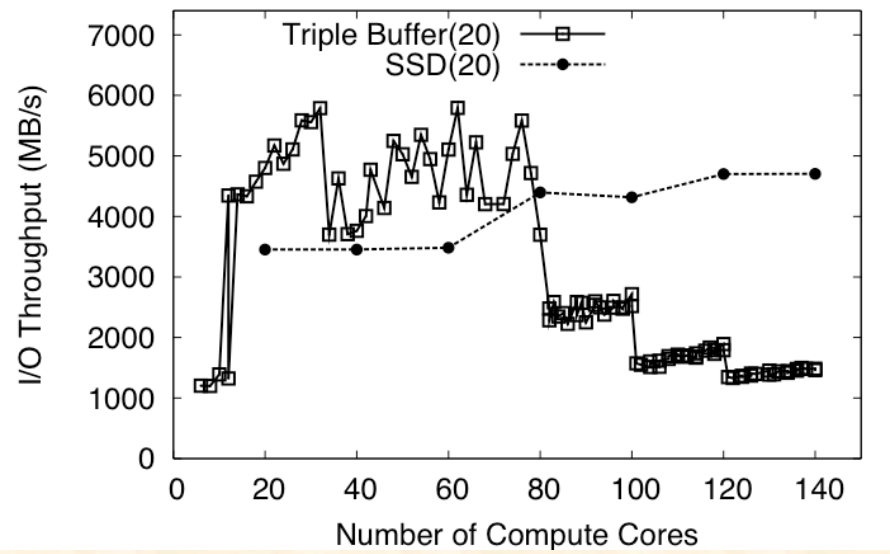



- Application writes to the mount point translate into striping of chunks across a stripe width of benefactors

# Checkpoint Storage Performance

- Tested the storage system on 160 cores with 140 clients
- Checkpoint was 250MB/node
- Emulated storage draining
- SSD(20): 32GB/node emulated SSD on 20 nodes
- Triple Buffer(35/20): 1GB/node memory on 35/20 nodes
- SSD(20) has 14.7% lower I/O throughput than Triple Buffer(35)
- Triple Buffer(20) fills up faster that it can be drained



# Reactive vs. Proactive Fault Tolerance

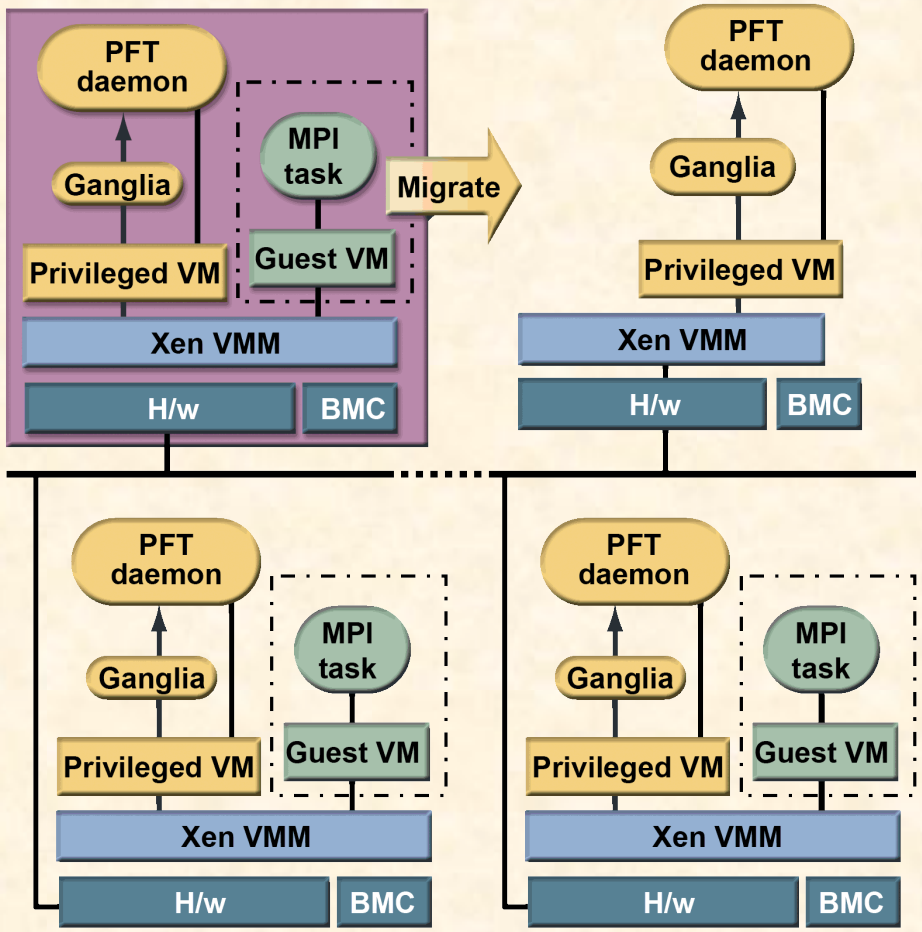
- Reactive fault tolerance
  - Keeps parallel applications alive through recovery from experienced failures
  - Employed mechanisms react to failures
  - Examples: Checkpoint/restart and message logging/replay
- Proactive fault tolerance
  - Keeps parallel applications alive by avoiding failures through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

# Proactive Fault Tolerance using Migration

- Relies on a feedback-loop control mechanism
  - Application health is constantly monitored and analyzed
  - Application is reallocated to improve to 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

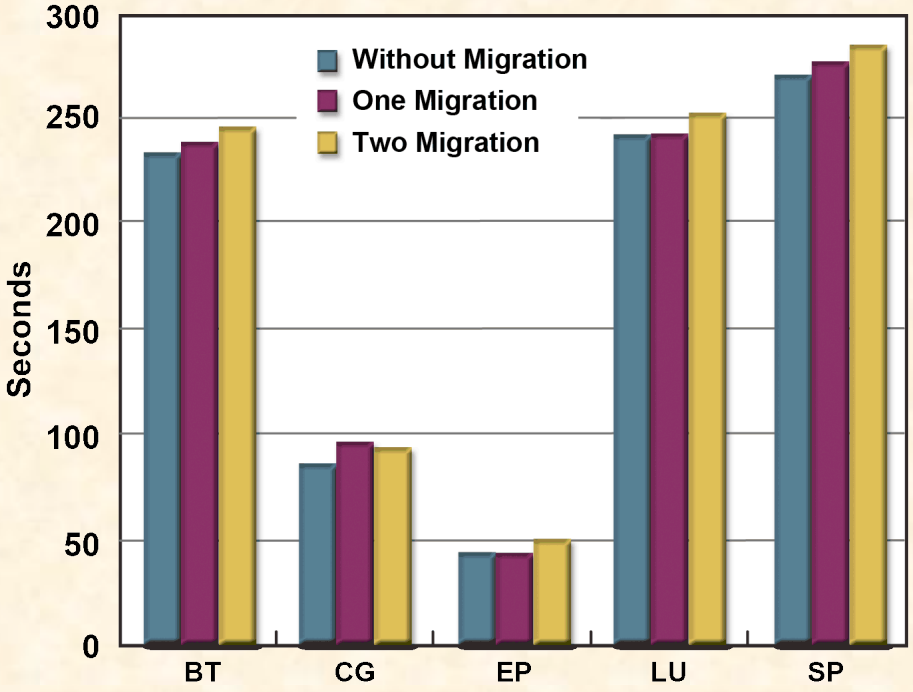
# VM-level Migration with Xen

- Type 1 system setup
  - Xen VMM on entire system
  - Host OS for management
  - Guest OS for computation
  - Spares without Guest OS
  - Monitoring in Host OS
  - Decentralized scheduler/ load balancer w/ Ganglia
- Deteriorating node health
  - Ganglia threshold trigger
  - Migrate guest OS to spare
  - Utilize Xen migration



# VM-level Migration Performance Impact

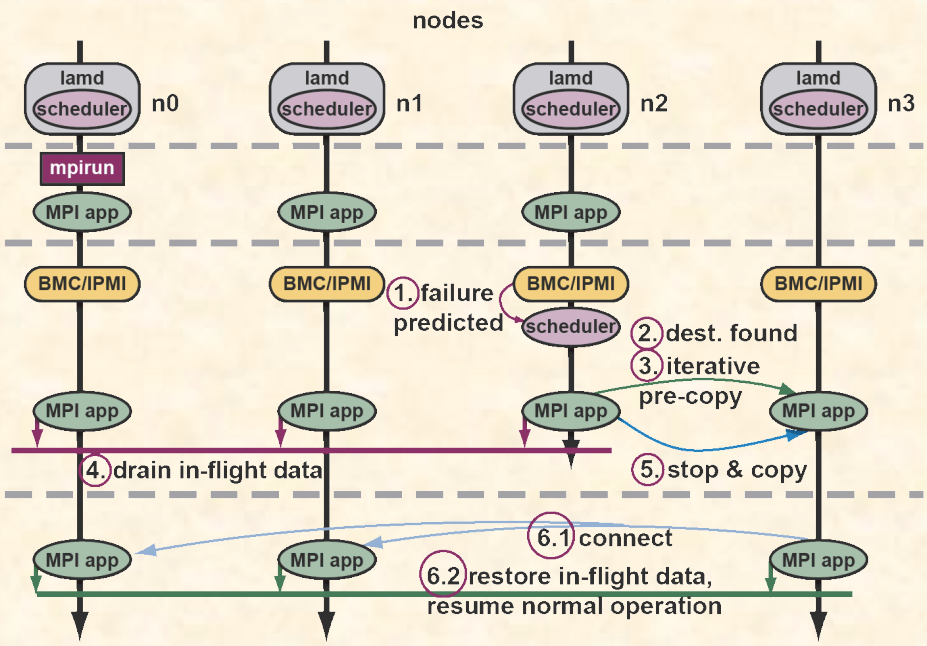
- Single migration overhead
  - Live : 0.5-5.0%
- Double migration overhead
  - Live : 2.0-8.0%
- Migration duration
  - Stop & copy : 13-14s
  - Live : 14-24s
- Application downtime
  - Stop & copy > Live




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

# Process-Level Migration with BLCR

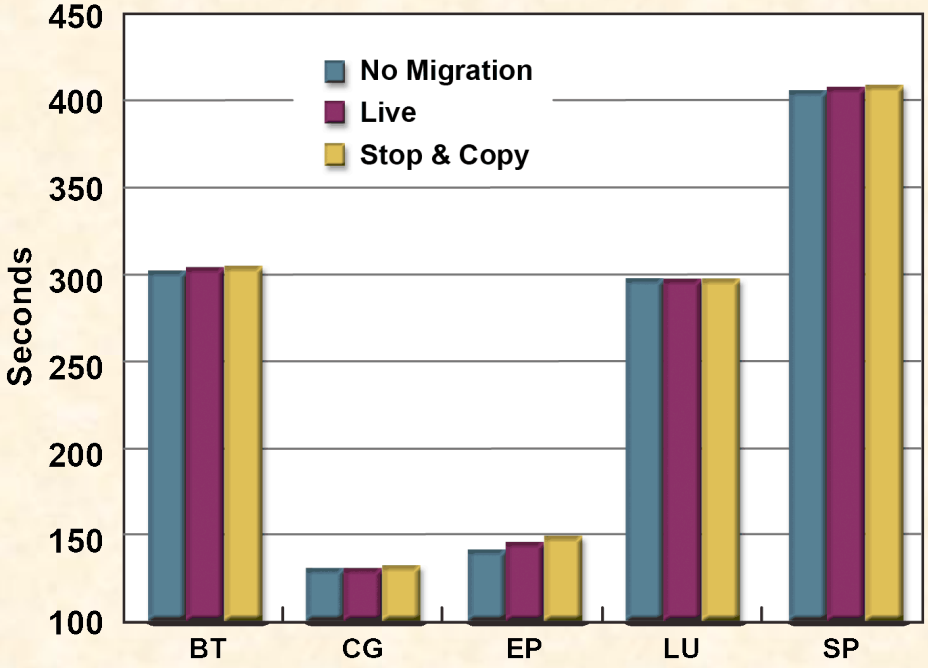
- LAM/MPI with Berkeley Lab Checkpoint/Restart (BLCR)
- Per-node health monitoring
- New decentralized scheduler/ load balancer in LAM
- New process migration facility in BLCR (stop&copy and live)
- Deteriorating node health
  - Simple threshold trigger
  - Migrate process to spare
- Available through BLCR distribution




- Expected integration into Open MPI+BLCR

# Process-Level Migration Performance Impact

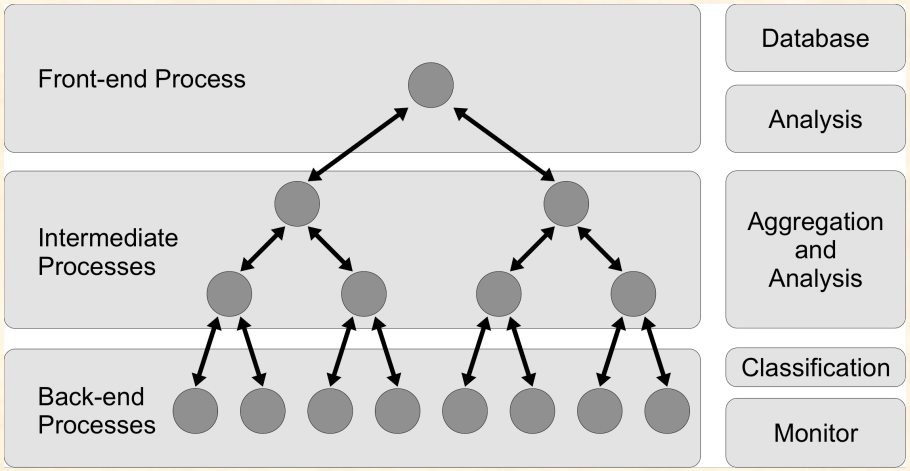
- Single migration overhead
  - Stop & copy : 0.09-6.00%
  - Live : 0.08-2.98%
- Single migration duration
  - Stop & copy : 1.0-1.9s
  - Live : 2.6-6.5s
- Application downtime
  - Stop & copy > Live
- Node eviction time
  - Stop & copy < Live




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

# MRNet-based System Monitoring

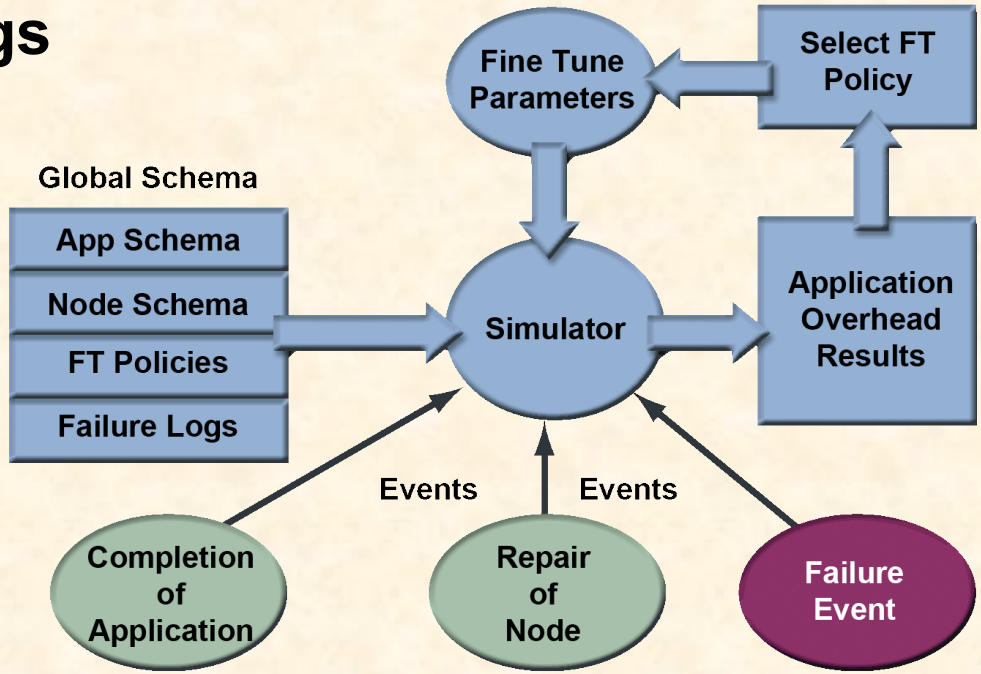
- Aggregation of metrics
- Tree-based overlay network
- Fan-in for metric data
- Fan-out for management
- Classification of data on back-end nodes
- In-flight processing on intermediate nodes
- Collection and storing on front-end node




- 1 MB of data in 4 hours
- ≈250 kB/hour
- ≈2 kb/interval
- ≈56x less than Ganglia

# Simulation of Fault Tolerance Policies

- DES with actual system logs
- Evaluation of policies
  - Reactive only
  - Proactive only
  - Reactive/proactive combination
- Evaluation of parameters
  - Checkpoint interval
  - Prediction accuracy
- Customizable simulation
  - # of active/spare nodes
  - Checkpoint and migration overheads



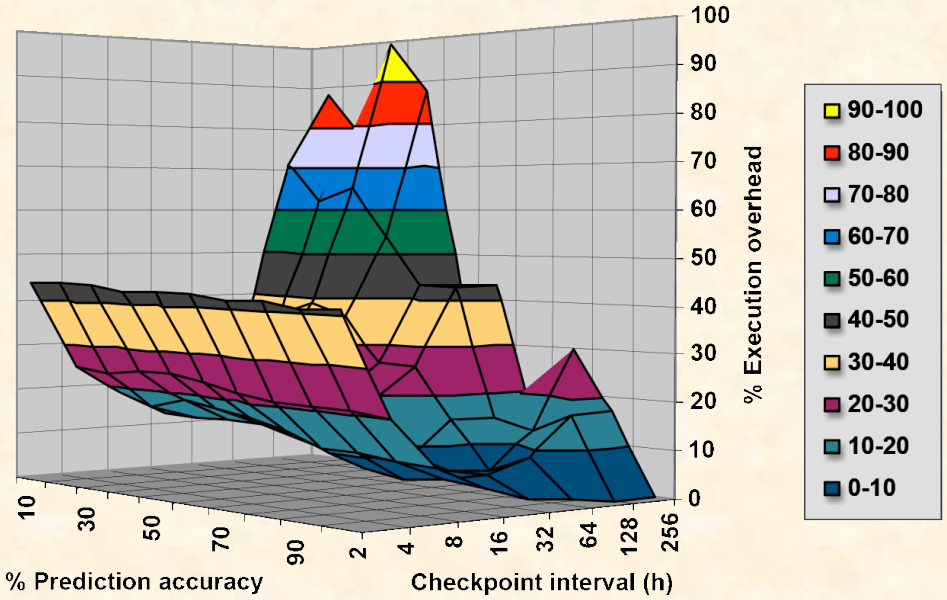
# Combining Proactive & Reactive Approaches

- Optimum for the given logs:
  - Prediction accuracy > 60%
  - Checkpoint interval 16-32h
- 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 |

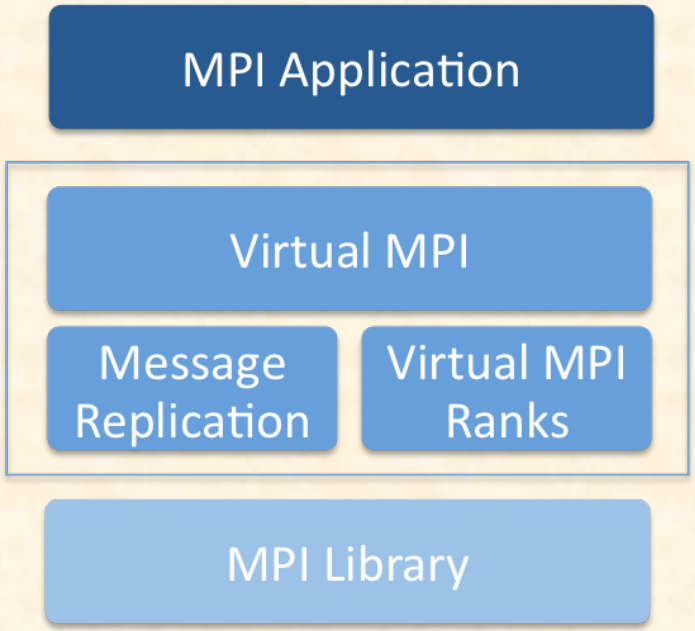
*Simulation based on ASCI White logs (nodes 1-125 and 500-512)*

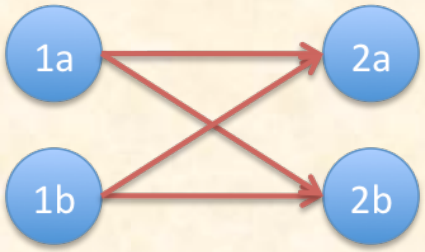
Execution overhead for various checkpoint intervals and different prediction accuracy



# Software Redundancy with MR-MPI

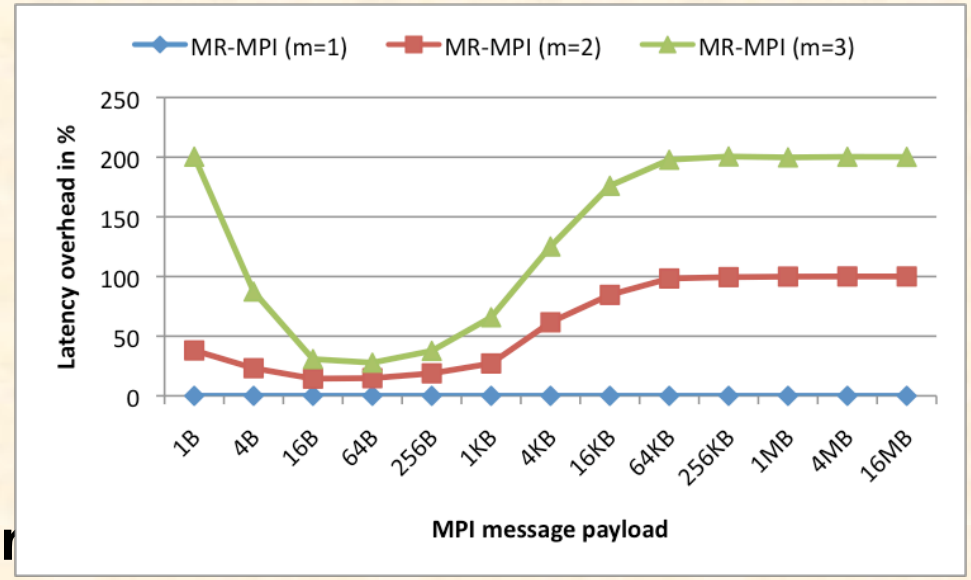
- Developed MR-MPI for transparent redundant execution of MPI applications
- Interposition library between MPI and the application
- App. runs with *r* * *m* ranks:
  - *r* ranks visible to the app.
  - *m* is the replication degree
- Fault model is fail-stop
- All messages are replicated
- Non-determinism (e.g. any source) is resolved with master/failover mechanism

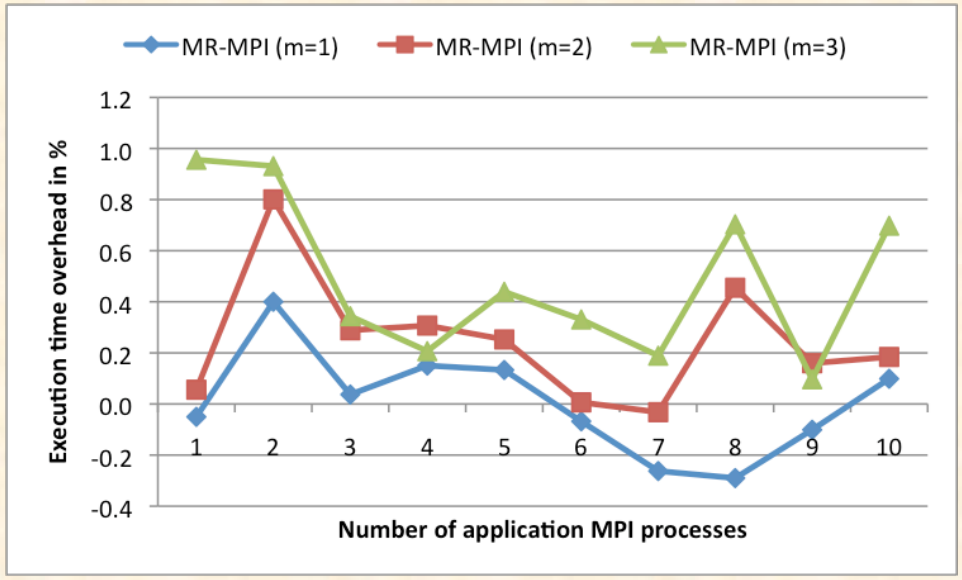


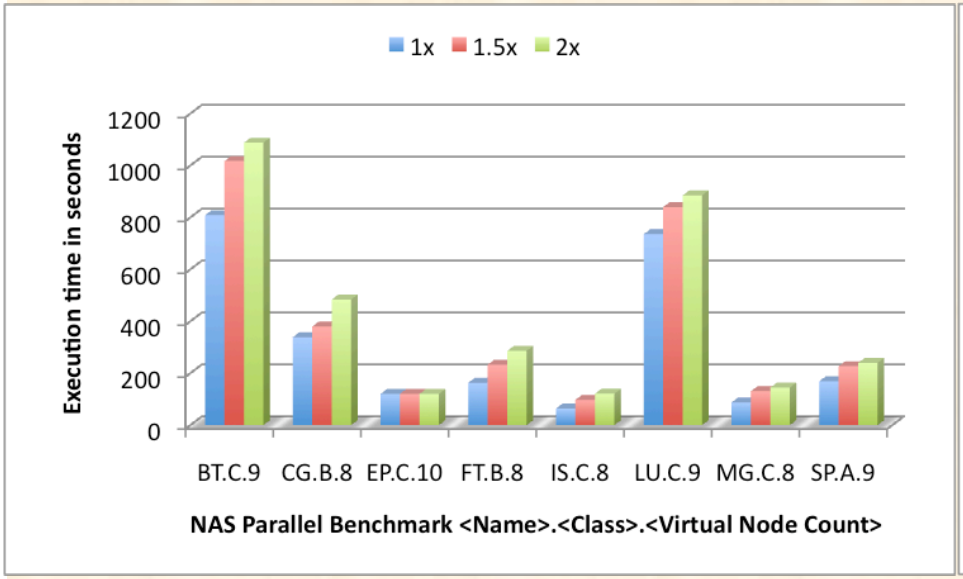
# MR-MPI Performance

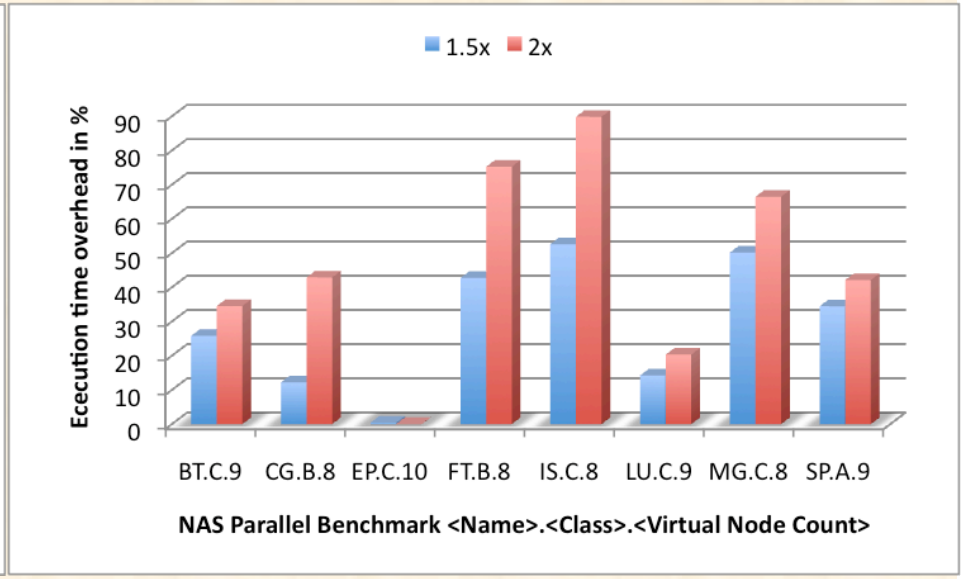
- Message replication ($O(m^m)$) influences performance
  - Up to *m*-times overhead for communication intensive applications
  - In contrast, 0% overhead for computation intensive applications
- Performance issue can be solved with alternatives for recovering lost messages
  - Sender- or receiver-side message caching
  - Group communication



# MR-MPI Performance: NAS Parallel Benchmarks

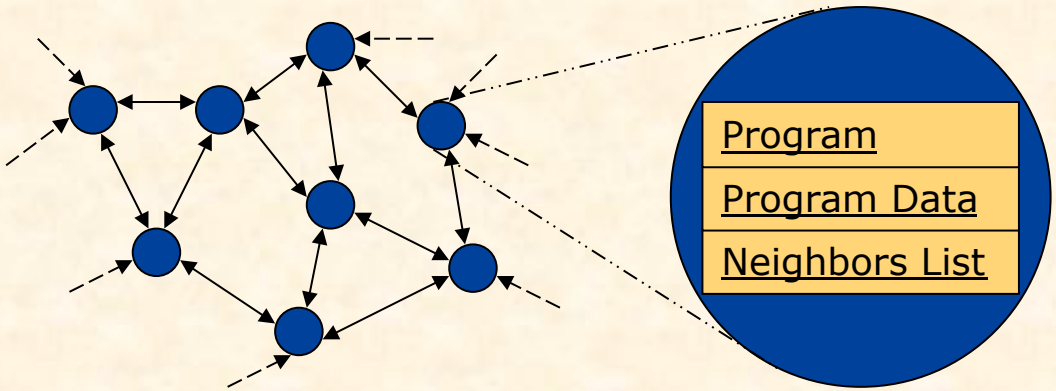



- 1x = non-redundant, 1.5x = 50% redundant, 2x = 100% = DMR
- Performance tests performed on XTORC cluster
- Up to 20 physical nodes with shared 100 Mbps network
- Each node: Intel Pentium 4 2GHz with 1GB RAM

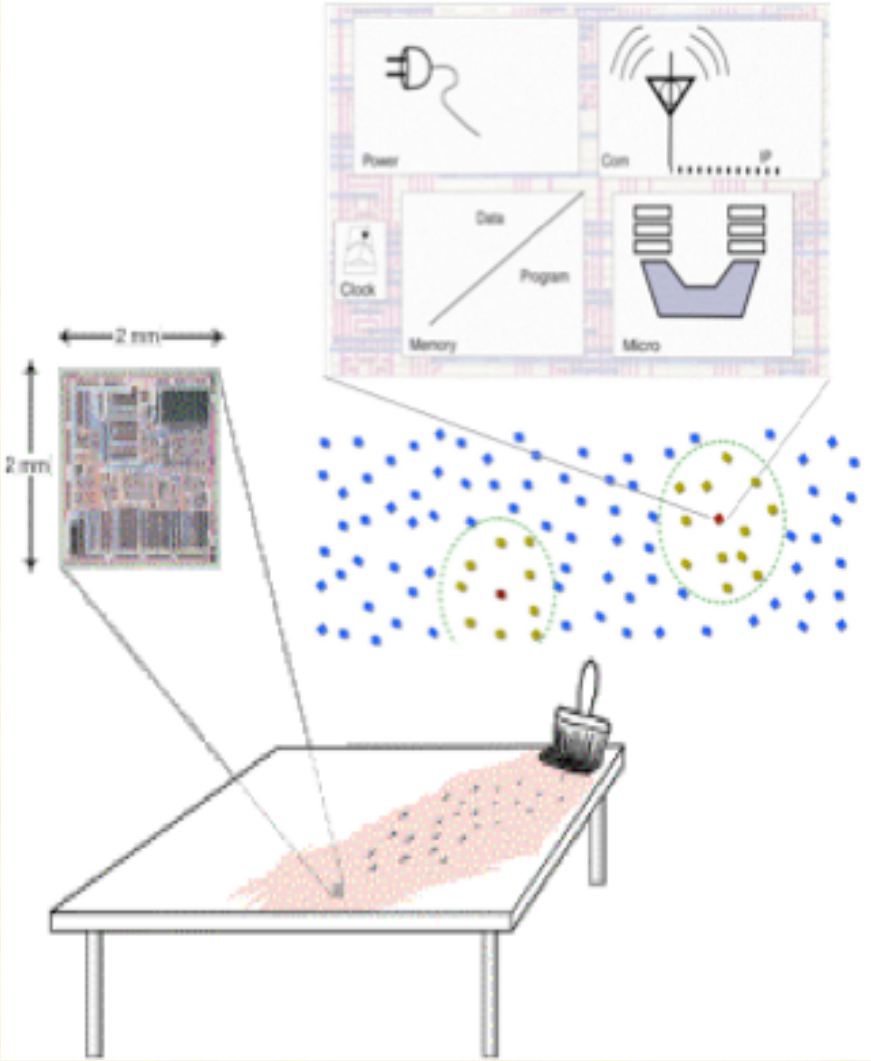
# Cellular Algorithms Theory

- Processes have only limited knowledge mostly about other processes in their neighborhood
- Application is composed of local algorithms
- Less inter-process dependencies, e.g, not everyone needs to know when a process dies (natural fault tolerance)
- Peer-to-peer communication promotes scalability
- MIT Media Lab. Research: Paintable Computing (~2000)



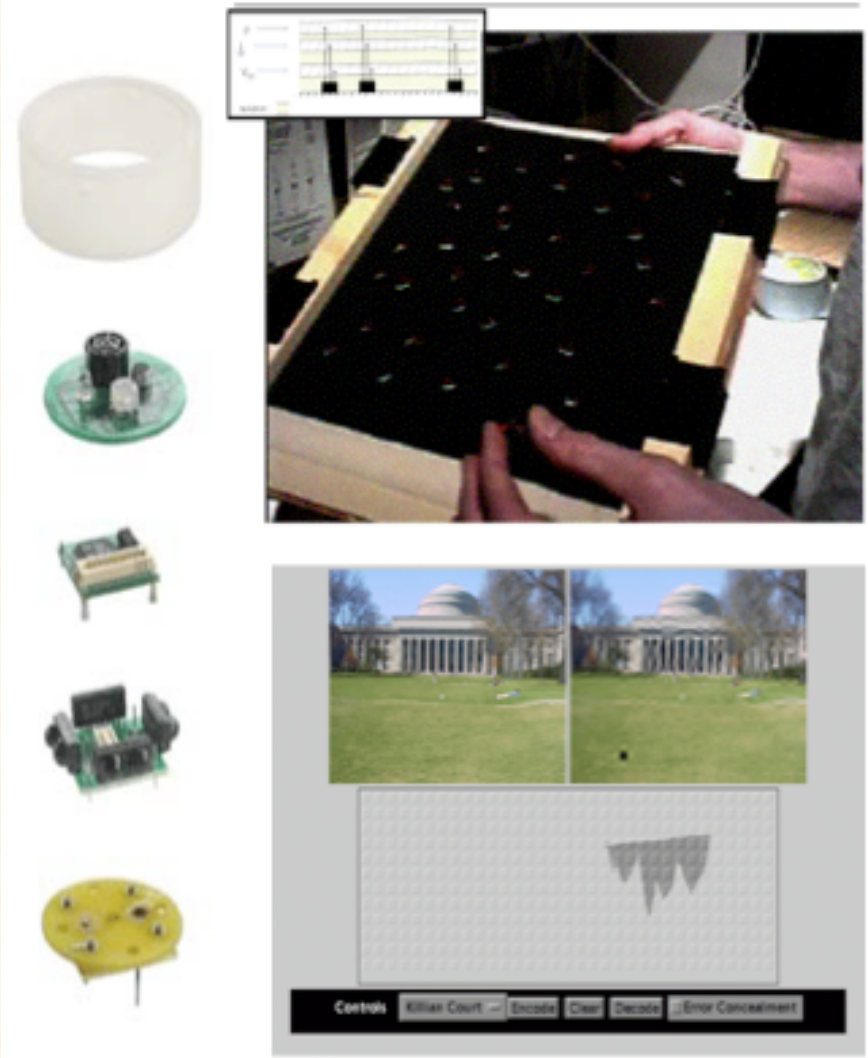
# MIT Research: Paintable Computing

- 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



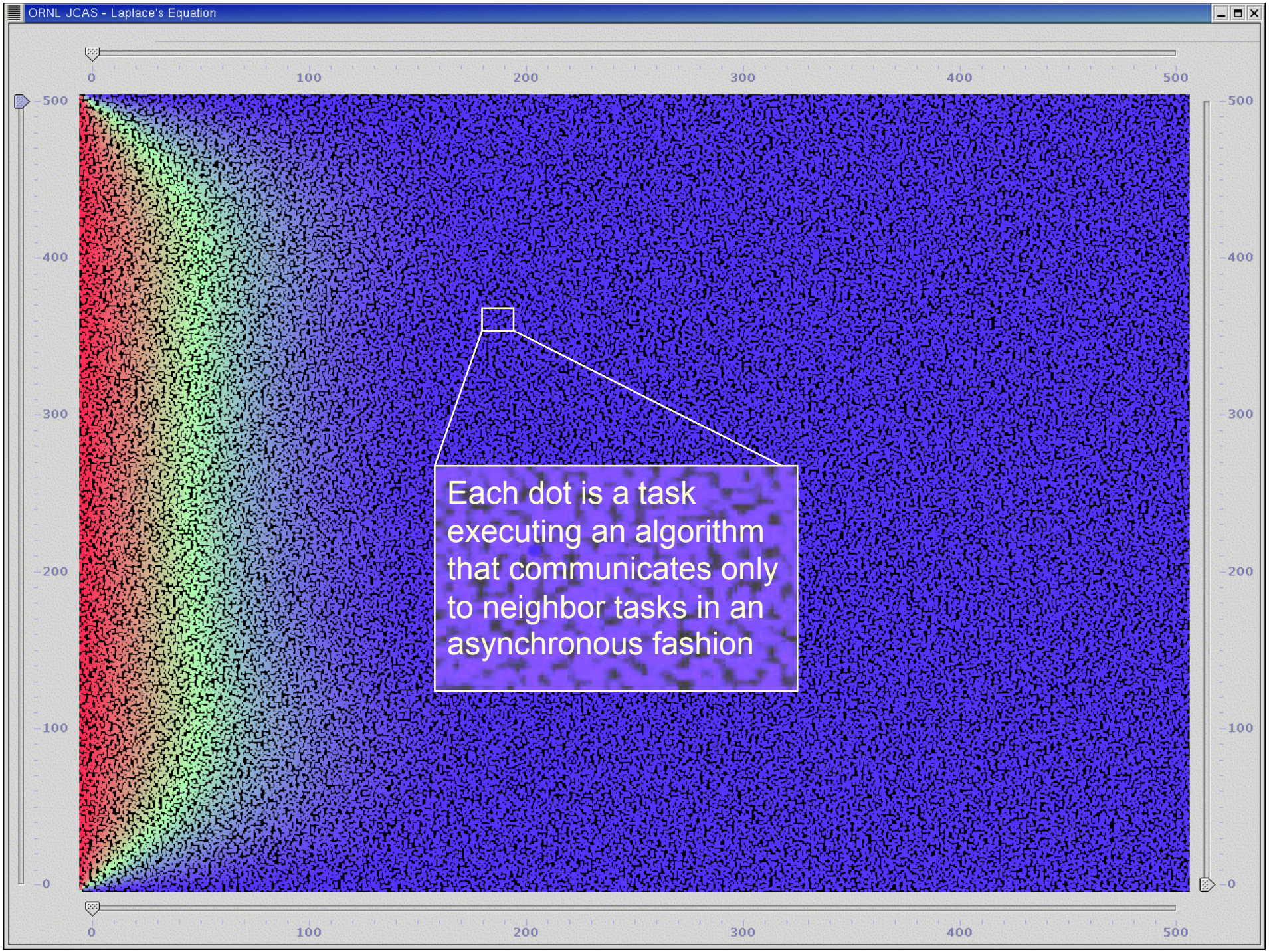
# MIT Research: Pushpin Computing

- 100 embedded nodes
- 1.25m x 1.25m pushpin board provides power
- Initial applications:
  - Distributed audio stream storage
  - Fault-tolerant holistic data (image) storage
- Ongoing research:
  - Sensor networks



# Java Cellular Architecture Simulator (JCAS)

- Developed in Java with native C/Fortran support (2002-04)
- Runs as standalone or distributed application
- Lightweight framework that 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 virtual time
- Primitive fault-tolerant MPI support
  - No collectives, no MPI 2


ORNL JCAS - Laplace's Equation
0
100
200
300
400
500
Each dot is a task executing an algorithm that communicates only to neighbor tasks in an asynchronous fashion

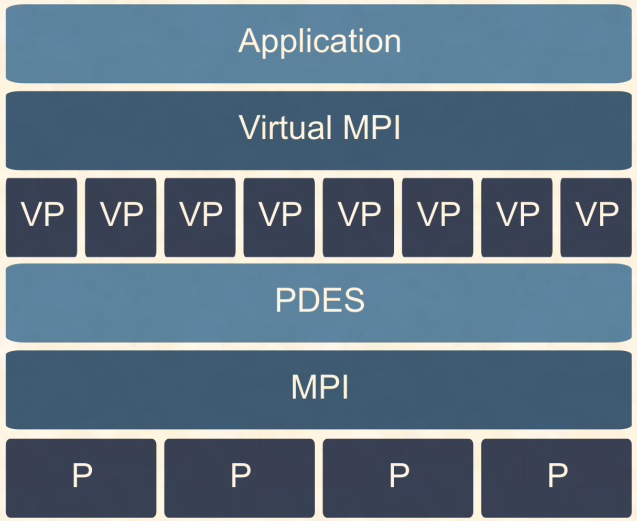
# Targeted Applications/Algorithms (Natural Fault Tolerance and Super-Scalable)

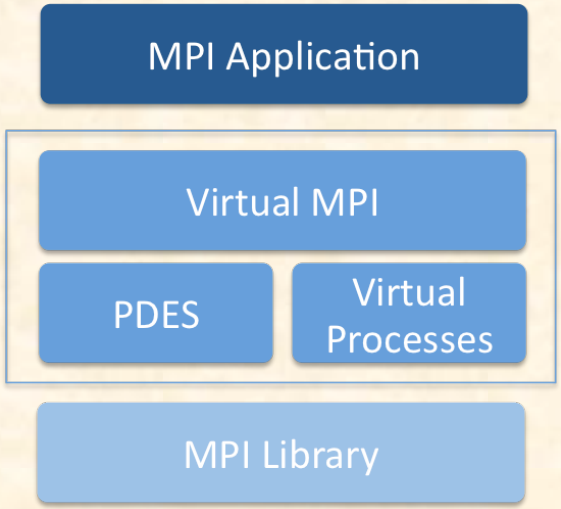
- Local information exchange algorithms:
  - Mesh-free chaotic relaxation (Laplace/Poisson)
  - Finite difference/element methods
  - Dynamic adaptive refinement at runtime
  - Asynchronous multi-grid methods
  - Monte Carlo method
  - Peer-to-peer diskless checkpointing
- Global information exchange algorithms:
  - Global peer-to-peer broadcasts of values
  - Global maximum/optimum search
- Applications:
  - Locally self-consistent multiple scattering (LSMS) method
  - Molecular dynamics simulation for computational biology

# Xsim – The Extreme-Scale Simulator

- Parallel discrete event simulator atop MPI (2010)
- Facilitate application-architecture co-design
  - Processor model
  - Network model
- Easy to use:
  - Replace the MPI header
  - Compile and link with xsim
  - Run the MPI program:

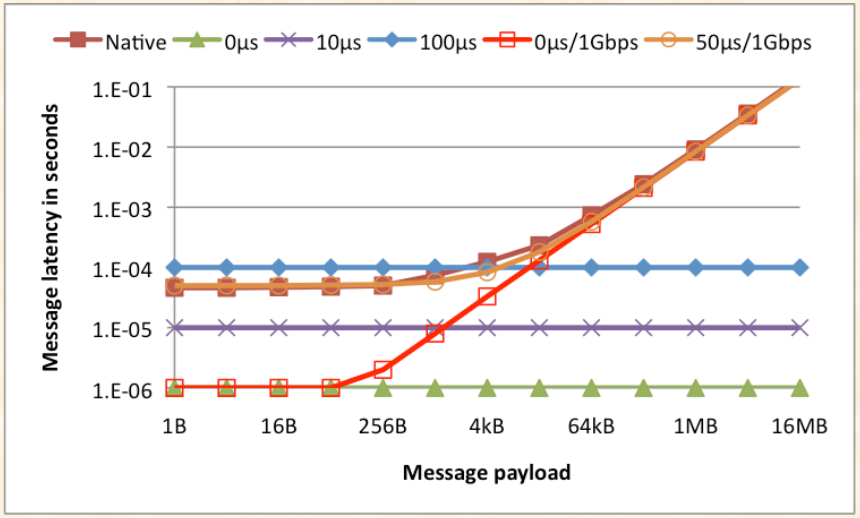
mpirun -np <np> <prog> -xsim-np <vp>

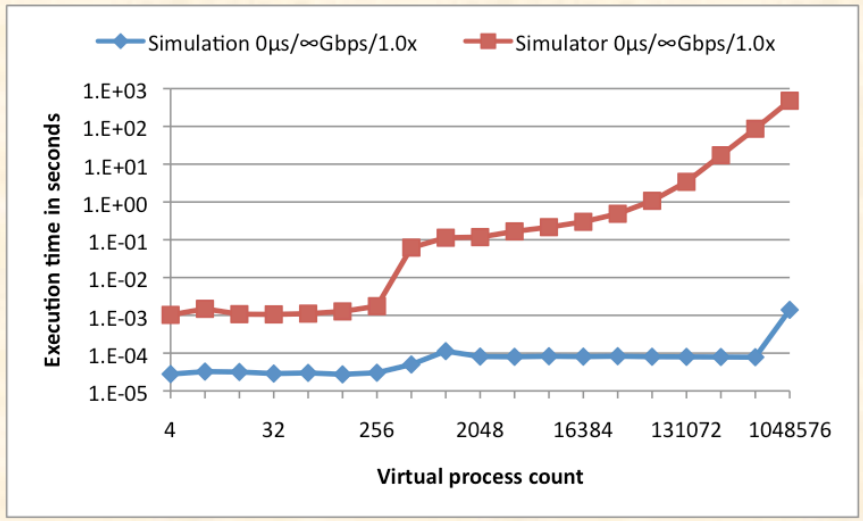


# Xsim Network and Processor Models

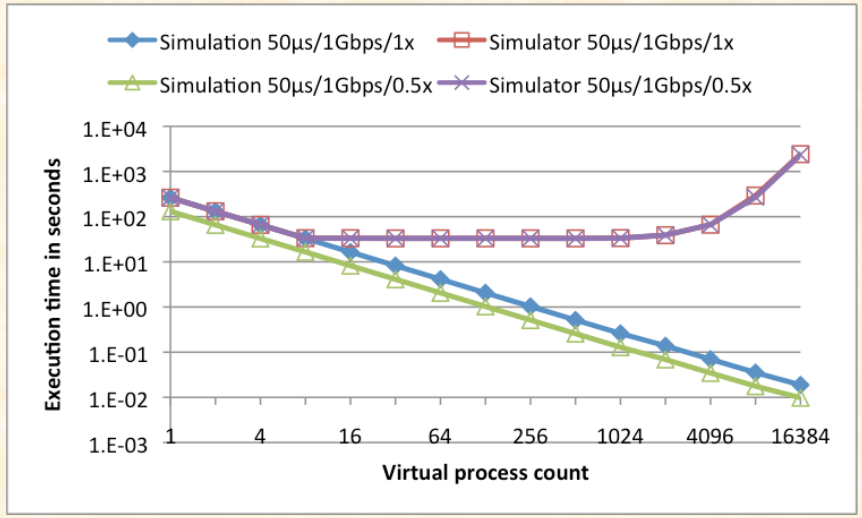
- Network model: latency, bandwidth and architecture
- Processor model: speed relative to actual execution
- Models can be set to 0|∞ to get baseline numbers
- Basic MPI `hello world` can be scaled up to 1M+ virtual MPI processes on just 4 real nodes

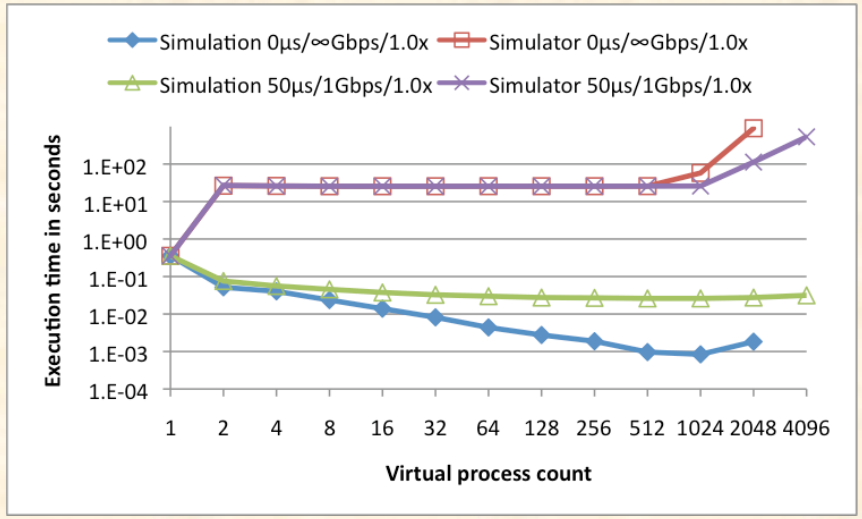


# Co-Design with Xsim: Evaluating Application Performance on Different Architectures

- Computation-intensive PI Monte Carlo solver
  - Perfect scaling
  - Processor models differ, thus exec. time differs
- Communication-intensive 1D Heat Eq. solver
  - Perfect scaling with 0µs/∞Gbps network model
  - Network models differ, thus exec. time differs
- Ongoing work in fault injection for app. testing



# Interesting Work by Others: Fault Tolerant Algorithms (1/3)

- **Numerical reconstruction for parabolic 3D heat solvers**
  - **Processes that didn't fail don't restart**
  - **Recover from checkpoint or initial solution**
  - **Forward implicit time integration on restarted process**
  - **Backward explicit time integration on neighbors**
- ***Hatem Ltaief, Edgar Gabriel, Marc Garbey: Fault tolerant algorithms for heat transfer problems. J. Parallel Distrib. Comput. 68(5): 663-677 (2008)***

# Interesting Work by Others: Fault Tolerant Algorithms (2/3)

- **Checkpoint-free fault tolerance for iterative methods**
  - **Extend matrix with checksum**
  - **Processes that didn't fail don't restart**
  - **Recover from checksum in other processes**
- **Checkpoint-Free Fault Tolerance for ScaLAPACK**
  - **Similar checksum matrix approach for matrix-matrix multiplication and others**
- ***Zizhong Chen, Jack Dongarra: Highly Scalable Self-Healing Algorithms for High Performance Scientific Computing. IEEE Trans. Computers 58(11): 1512-1524 (2009)***

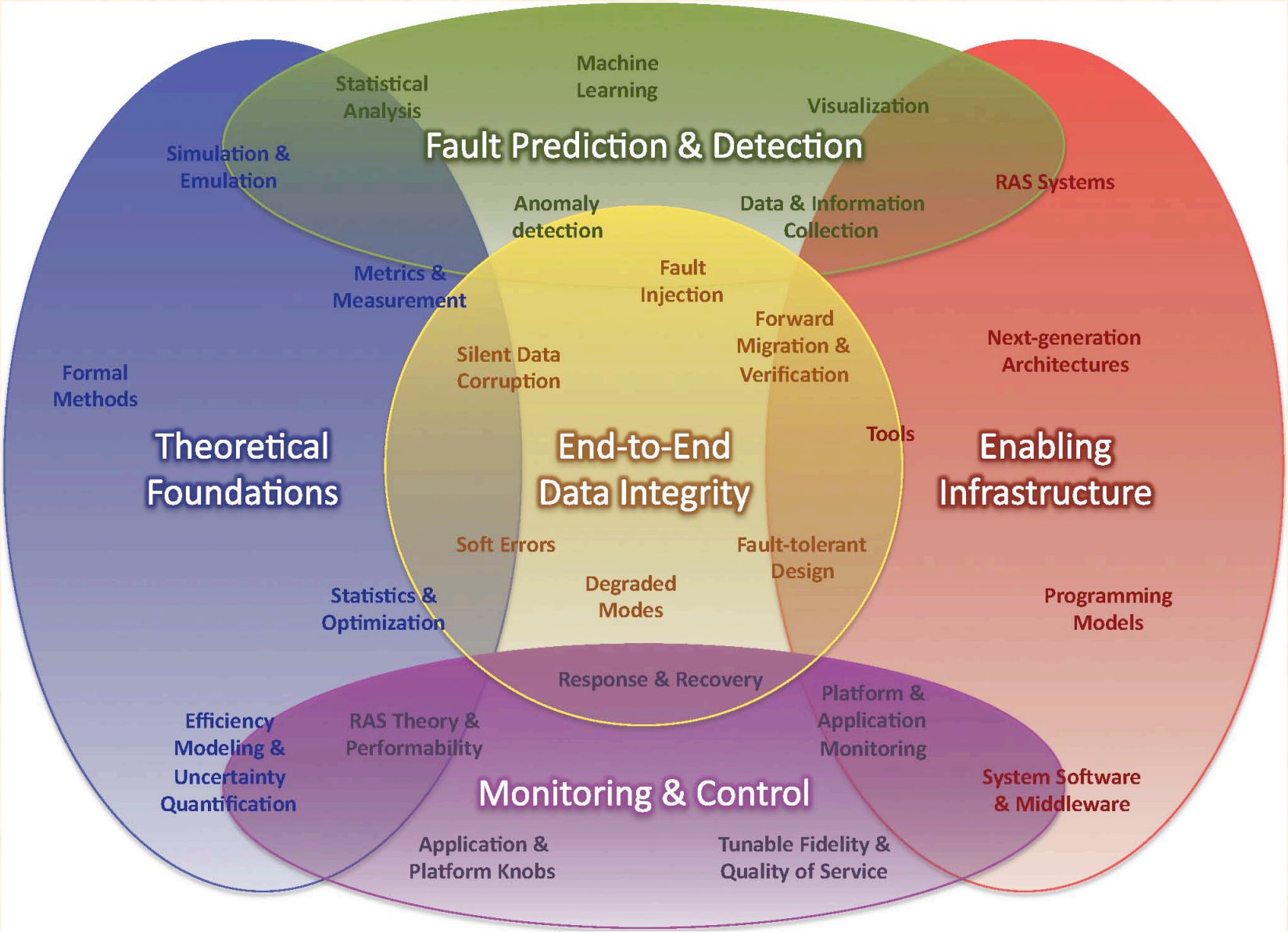
# Interesting Work by Others: Fault Tolerant Algorithms (3/3)

- **Fault-oblivious Monte Carlo methods to the rescue**
- **Mixed approach: Fast stochastic approximation and deterministic refinement algorithms for Matrix Inversion and Solving Systems of Linear Equations**
- ***Simon Branford, Cihan Sahin, Ashish Thandavan, Christian Weihrauch, Vassil N. Alexandrov, Ivan Tomov Dimov: Monte Carlo methods for matrix computations on the Grid. Future Generation Comp. Syst. 24(6): 605-612 (2008)***

# Other Past or Ongoing Work

- Fault-aware/-tolerant MPI
  - Fault detection and notification at scale
  - Reconfiguration of MPI runtime environment
  - Harness and FT-MPI
  - Open MPI Runtime Environment (ORTE)
  - Scalable Tools Communication Infrastructure (STCI)
- High availability for service nodes
  - Redundant parallel file system metadata server (PVFS)
  - Redundant job and resource manager (Torque)

# Key Areas for Future Research, Development, and Standards Work



# Acknowledgements


- Investigators at Oak Ridge National Laboratory (ORNL):
  - S. L. Scott, C. Engelmann, G. Vallée, T. Naughton, C. Wang, A. Tikotekar, G. Ostrouchov, S. Vazhkudai, Y. Kim, G. Shipman
- Investigators at Louisiana Tech University:
  - C. Leangsuksun, N. Naksinehaboon, R. Nassar, M. Paun
- Investigators at North Carolina State University:
  - F. Mueller, A. Nagarajan, J. Varma, X. Ma, F. Meng
- Investigators at Virginia Tech
  - M. Li, A. Butt
- Funding sources:
  - U.S. Department of Energy, Office of Science, FASTOS 2
  - U.S. National Science Foundation, CCF-0937827, CCF-0746832, CCF 0621470, CCF-0937690
  - ORNL LDRD and NCSU/ORNL Joint Appointment programs

# Workshop on Latest Advances in Scalable Algorithms for Large-Scale Systems (ScalA)

- @ SC'10: New Orleans, LA, USA, November 14-15, 2010
- Topics of interest include, but are not limited to:
  - Novel scientific algorithms that improve performance, scalability, resilience and power efficiency
  - Porting algorithms to many-core and heterogeneous architectures
  - Performance and resilience limitations of algorithms at scale
  - Crosscutting approaches (system software and applications)
  - Scientific algorithms that can exploit extreme concurrency
  - Naturally fault tolerant, self-healing or fault oblivious algorithms
  - Programming models & system software for scalability/resilience
- Chairs: V. Alexandrov (UoR), J. Dongarra (UT), A. Geist (ORNL)
- Paper deadline: 29 September, 2010
- URL: http://www.csm.ornl.gov/srt/conferences/Scala/2010

# Questions?