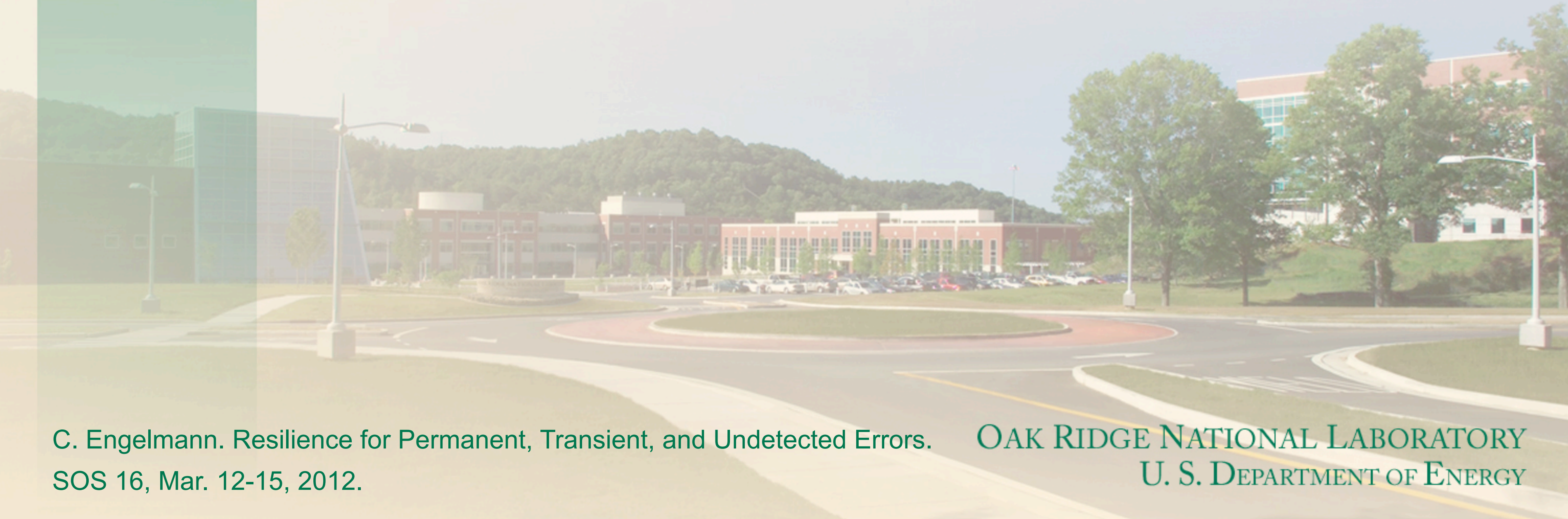
UT-BATTELLE

Runtime Systems and System Software Challenges:

Resilience for Permanent, Transient, and Undetected Errors

Christian Engelmann

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



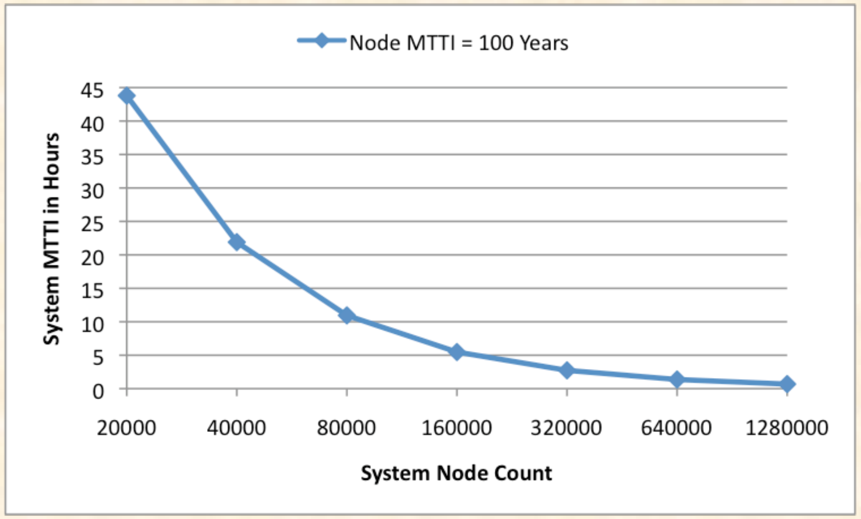
OAK RIDGE NATIONAL LABORATORY
U. S. DEPARTMENT OF ENERGY

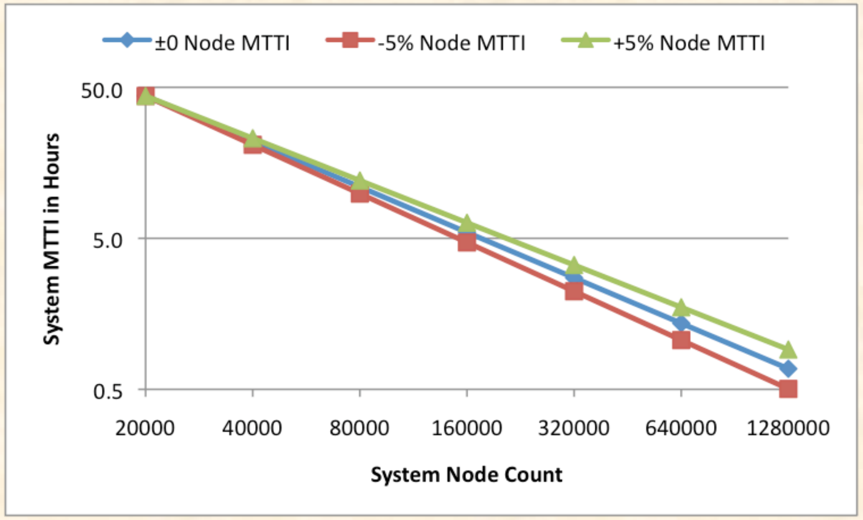
Exascale Resilience Challenges

- Significant growth in component count 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)
- Power management cycling decreases component lifetimes due to thermal and mechanical stresses
- Hardware fault detection and recovery is limited by power consumption requirements and production costs
- Heterogeneous architectures add more complexity to fault detection and recovery

The System MTTI Mystery

- **Theory: MTTI decreases as component count increases**
- **MTTI stayed roughly the same for the last 10 years**
  - **~40x more cores/nodes**
- **Node reliability has been improved accordingly**
  - **Removed disks and fans**
  - **Improved system software**
- **We are running out of room to improve node reliability**
  - **Node reliability may actually decrease**



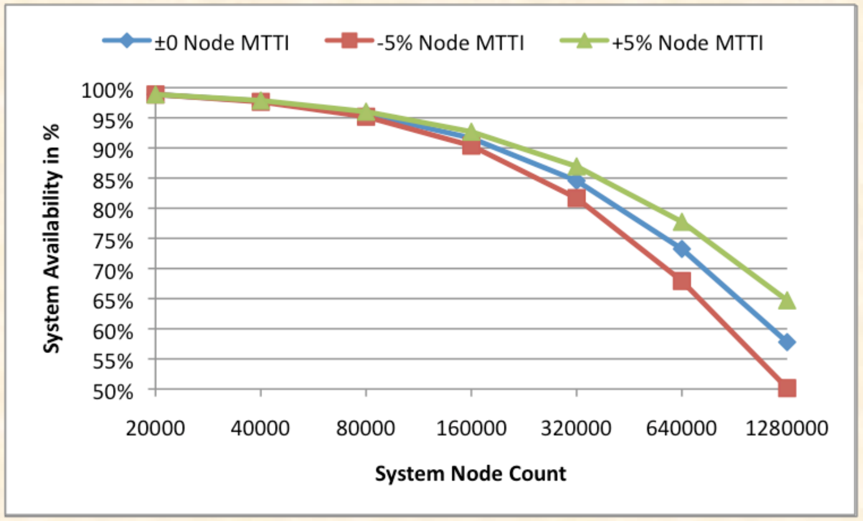


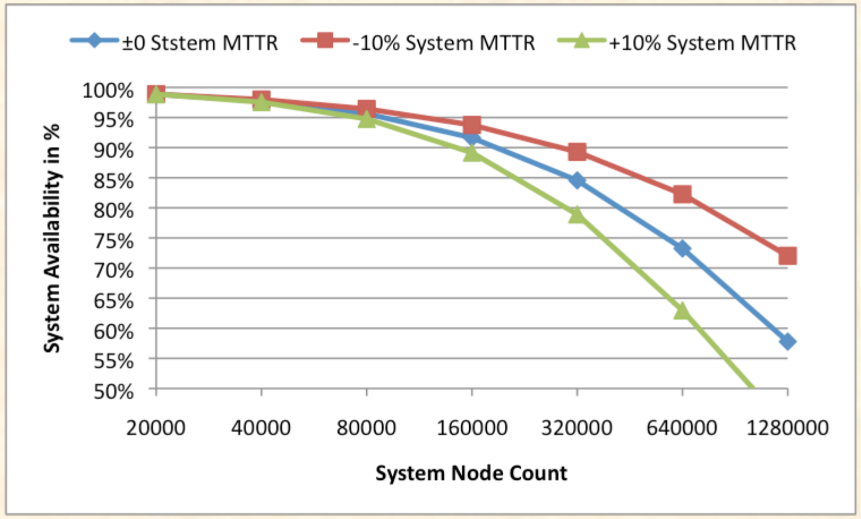
System MTTI is the Wrong Metric Anyway

- **Availability is the standard metric for uptime/progress**

$$A = \frac{MTTF}{MTTF + MTTR} = \frac{1}{1 + \frac{MTTR}{MTTF}}$$

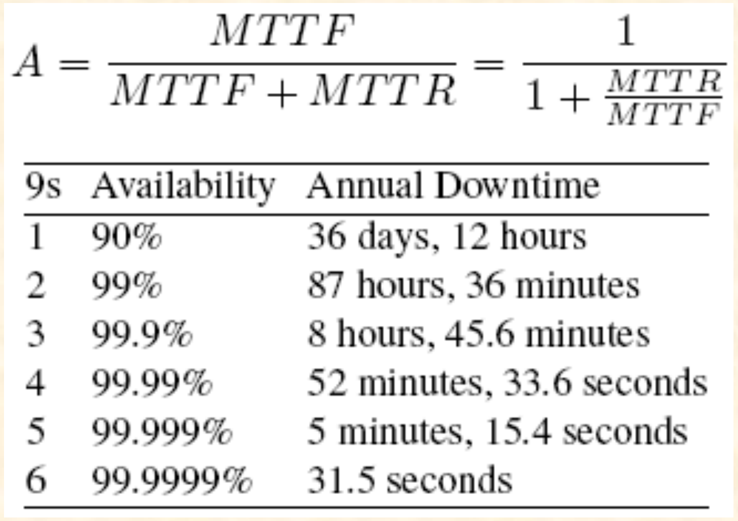
- **MTTR/MTTI ratio defines a system's availability**
  - There may be no room to improve node MTTI
  - There are certainly ways to improve system MTTR





The Node Availability Conundrum

- For 90% availability with 1M nodes, each none needs:
  - 7 nines without redundancy
  - 4 nines for DMR
  - 3 nines for dynamic DMR
  - 3 nines for TMR
  - 2 nines for dynamic TMR
- Can we achieve 7 nines w/:
  - Checkpoint/restart
  - Message logging
  - Algorithm-based fault tolerance
  - …?



$$A = \frac{MTTF}{MTTF + MTTR} = \frac{1}{1 + \frac{MTTR}{MTTF}}$$

9s	Availability	Annual Downtime
1	90%	36 days, 12 hours
2	99%	87 hours, 36 minutes
3	99.9%	8 hours, 45.6 minutes
4	99.99%	52 minutes, 33.6 seconds
5	99.999%	5 minutes, 15.4 seconds
6	99.9999%	31.5 seconds

Are we about to fail?

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 I/O coordination and scheduling increases significantly
- 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

HPC Resilience Solutions

- Advanced resilience solutions
  - System-level and incremental/differential checkpoint/restart
  - Checkpoint/restart to memory/SSDs in neighbor or I/O nodes
  - Uncoordinated checkpoint/restart with message logging
  - Fault tolerant MPI and algorithm-based fault tolerance
  - Proactive fault tolerance (migration-based fault avoidance)
  - Rejuvenation (reboot/refresh to clear latent errors)
  - Process and data-level redundancy (DMR and software ECC)
- Only system-level checkpoint/restart is used in production
- None of the other solutions are even close to production
- There is a huge gap between research and production
- There are no methods for comparing solutions

FAST-OS II: Hybrid Checkpointing with BLCR

- Hybrid checkpointing: 1 full and k incremental (part of BLCR distribution)
- Tracks and saves dirty memory pages
- Full: Saves all pages to file
- Incremental: Appends to checkpoint file
- Recovery: Scans file in reverse sequence

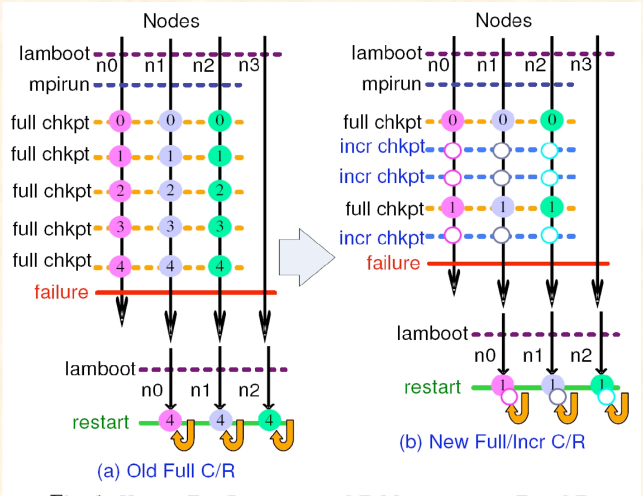
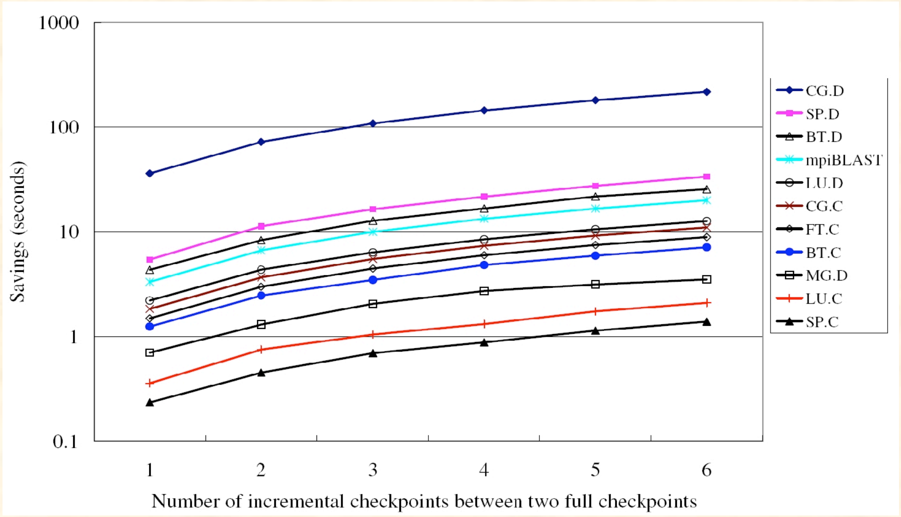
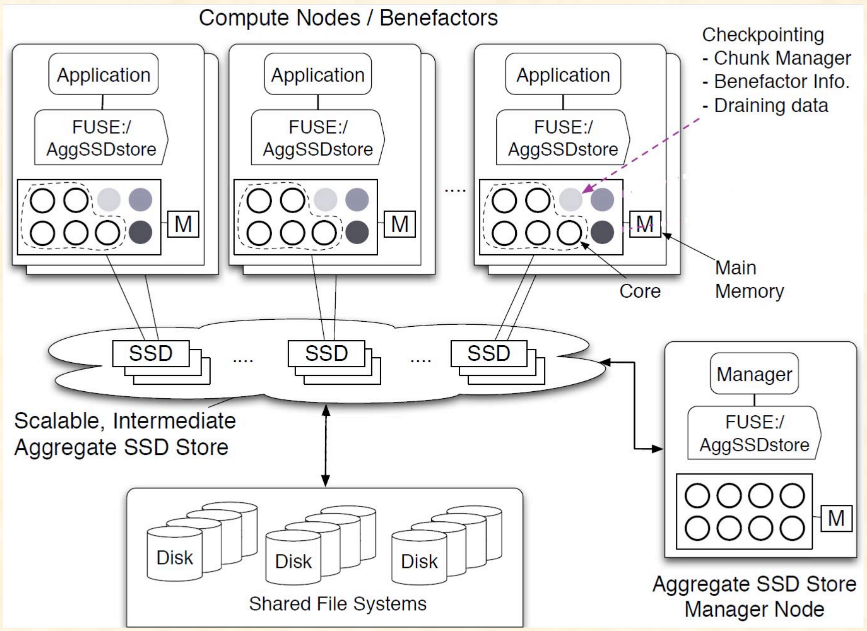


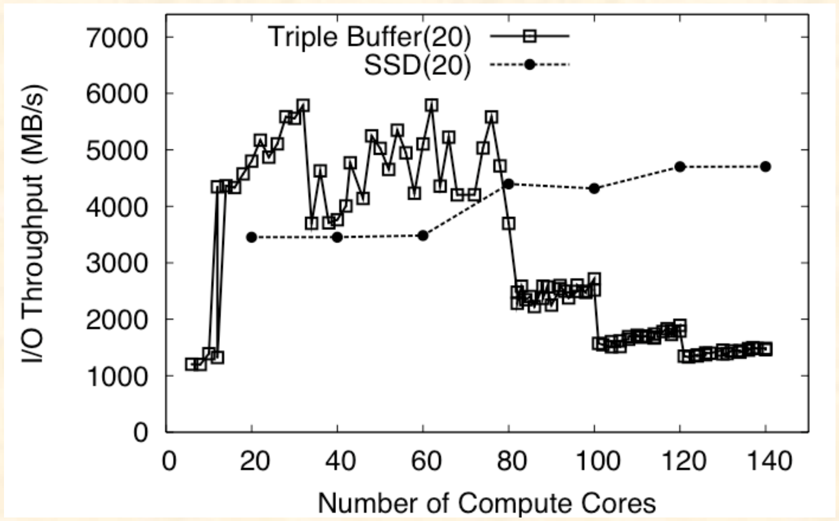
Fig. 1: Hybrid Full/Incremental C/R Mechanism vs. Full C/R



LDRD: Checkpoint Storage Architecture

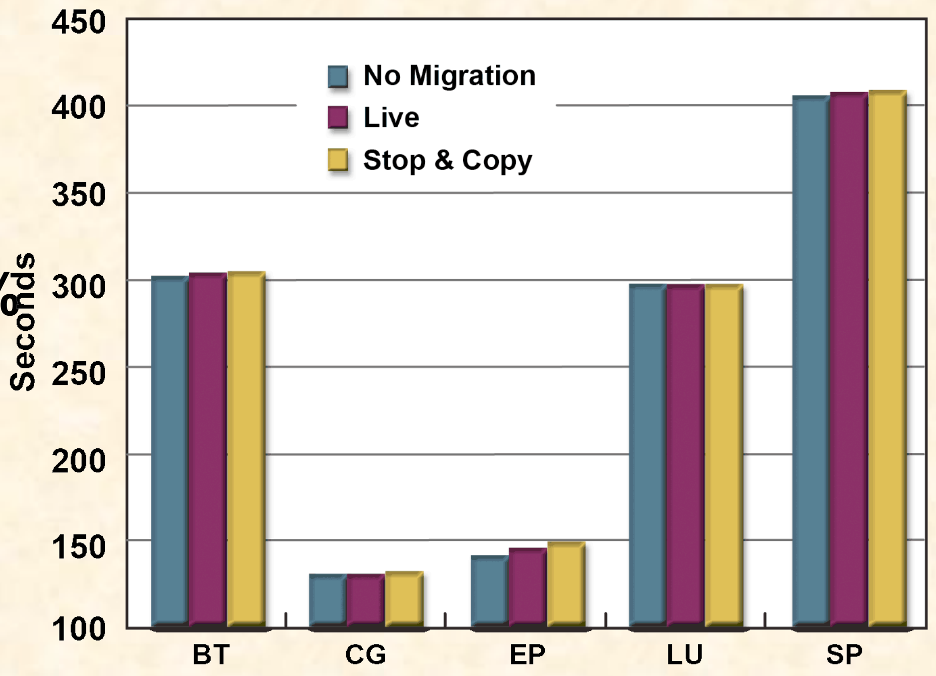
- Nodes offer available node-local SSD or memory space
- Manager process maintains metadata for each client
- FUSE client offers mount point w/o POSIX locking
- Data striped across nodes
- Tested on system with 160 cores using 140 as clients





FAST-OS II: Process-Level Migration w/ BLCR

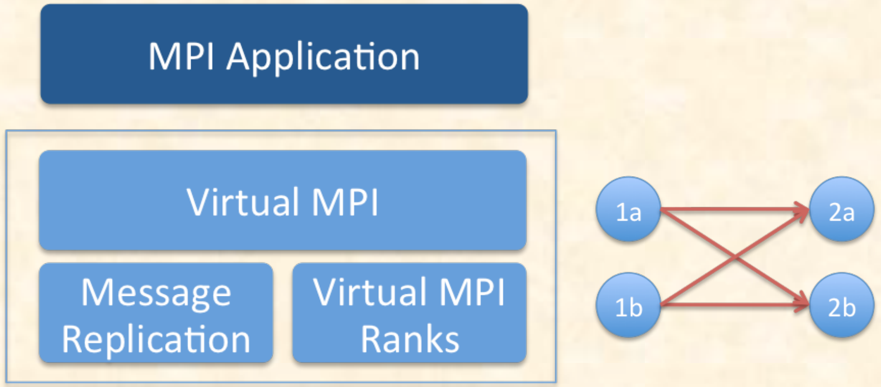
- Stop&copy + live migration (part of BLCR distribution)
- Single migration overhead
  - Stop & copy : 0.09-6.00%
  - Live : 0.08-2.98%
- Single migration duration
  - Stop & copy : 1.0-1.9s
  - Live : 2.6-6.5s
- Application downtime
  - Stop & copy > Live
- Node eviction time
  - Stop & copy < Live

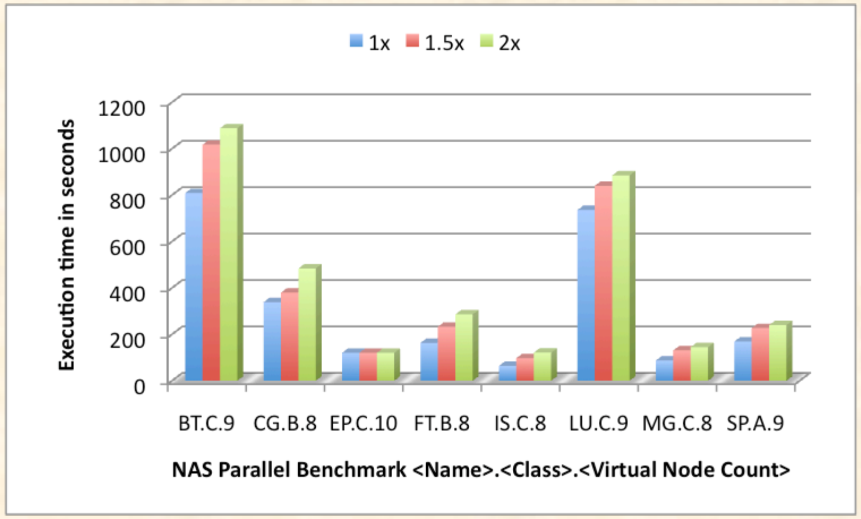


NPB runs on 16-node dual-core dual-processor Linux cluster at NCSU with AMD Opteron and Gigabit Ethernet

LDRD: Process-level Redundancy atop MPI

- **Transparent redundant execution of MPI apps. (MR-MPI, rMPI, redMPI)**
- **Interposition library between MPI and the app.**
- **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**





Hardware/Software Co-design for Resilience

- We can build an exascale hardware platform that never fails, but at which cost
  - We need to help vendors to avoid over-engineering and the corresponding cost explosion
- Hardware/software co-design tools for resilience
  - Define agreed upon definitions, metrics and methods
  - Evaluate the performance and power consumption cost of resilience solutions to provide vendors with choices
  - Evaluate the resilience properties of hardware and software
  - Coordinate the interfaces/responsibilities of individual layers
- There are currently no HPC resilience co-design tools

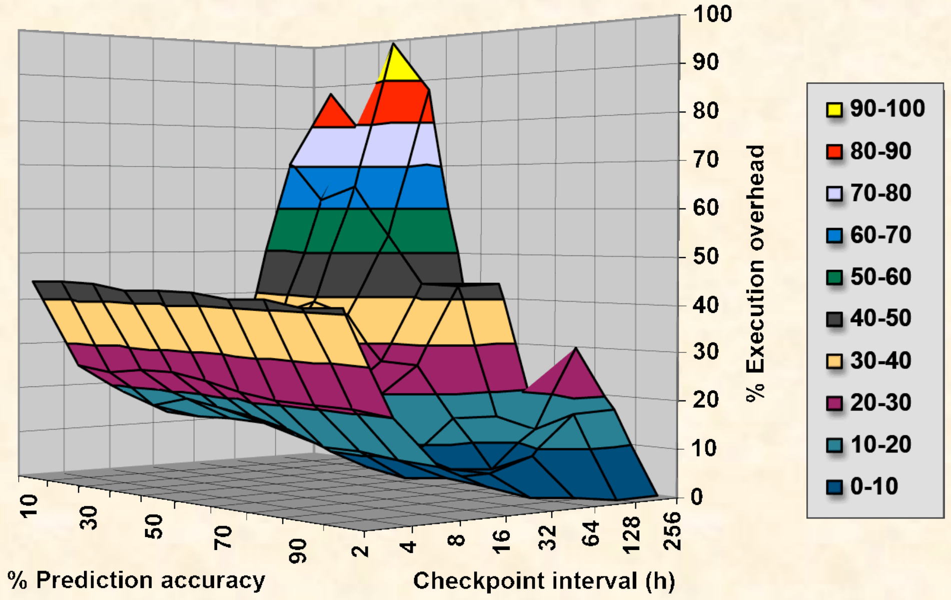
FAST-OS II: Combining Checkpoint/Restart with Prediction-based Migration

- 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

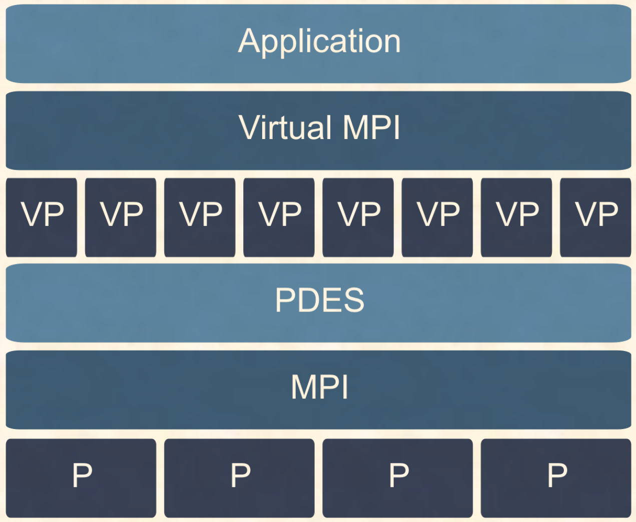
Simulation based on ASCI White logs (nodes 1-125 and 500-512)

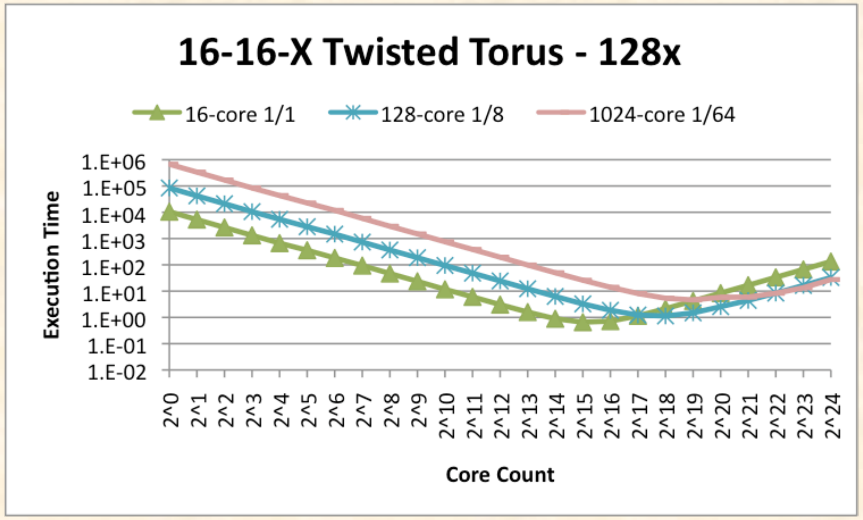
Execution overhead for various checkpoint intervals and different prediction accuracy



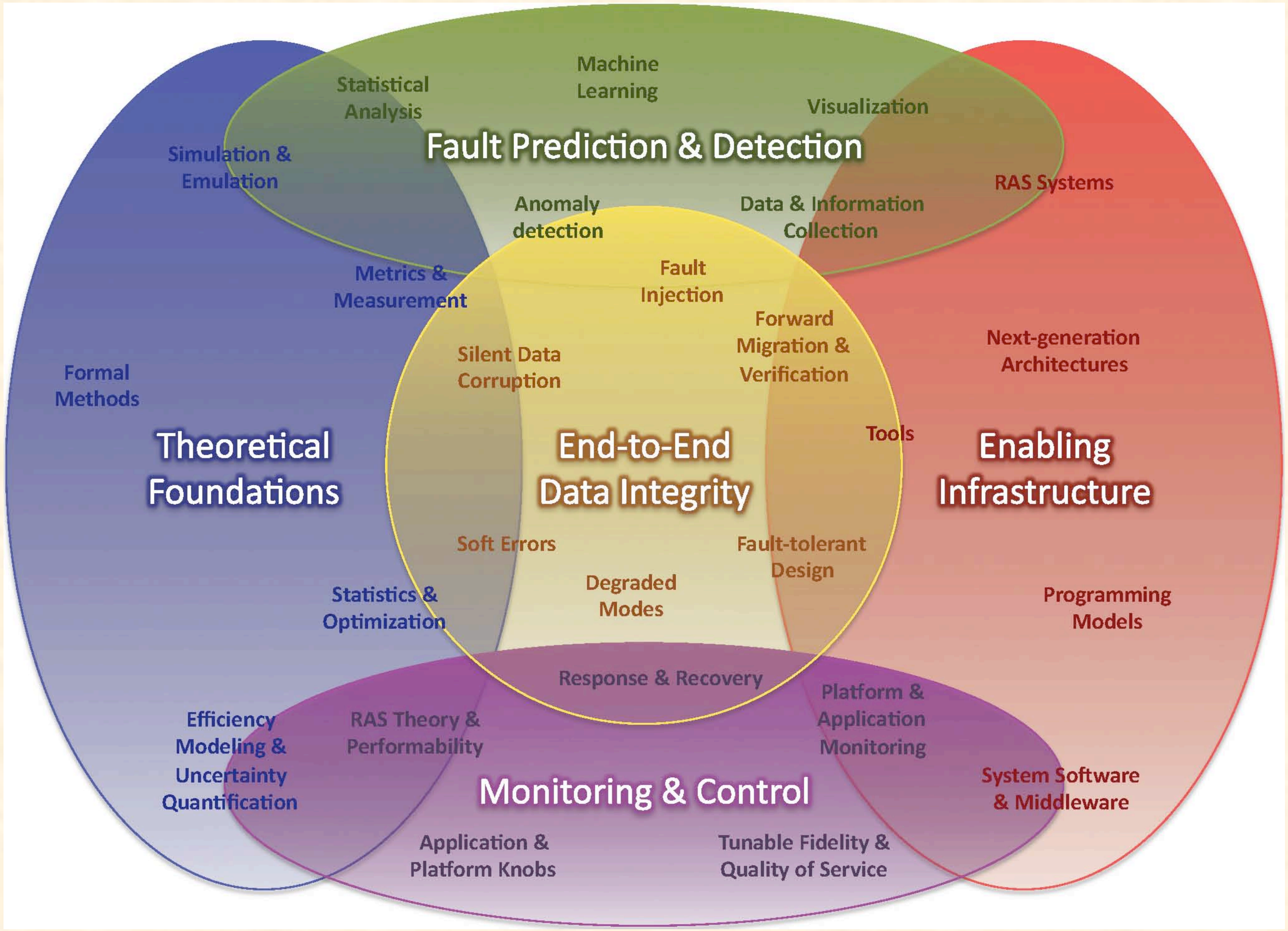
IAA/EASI: xSim – Extreme-Scale Simulator

- Parallel discrete event simulation (PDES) atop MPI
- Facilitates application-architecture co-design
  - Processor model
  - Network model
- Trades off simulation accuracy to gain scalability
- Scales to 100,000,000 ranks
- Ongoing work
  - Fault-tolerant MPI support
  - Fault injection capabilities





Key Areas for Future Resilience Research, Development and Standards Work


Machine Learning
Statistical Analysis
Visualization
Fault Prediction & Detection
Simulation & Emulation
RAS Systems
Anomaly detection
Data & Information Collection
Fault Injection
Metrics & Measurement
Forward Migration & Verification
Next-generation Architectures
Silent Data Corruption
Formal Methods
Tools
Theoretical Foundations
End-to-End Data Integrity
Enabling Infrastructure
Soft Errors
Fault-tolerant Design
Degraded Modes
Statistics & Optimization
Programming Models
Response & Recovery
Platform & Application Monitoring
Efficiency Modeling & Uncertainty Quantification
RAS Theory & Performability
System Software & Middleware
Monitoring & Control
Application & Platform Knobs
Tunable Fidelity & Quality of Service

Theoretical Foundations

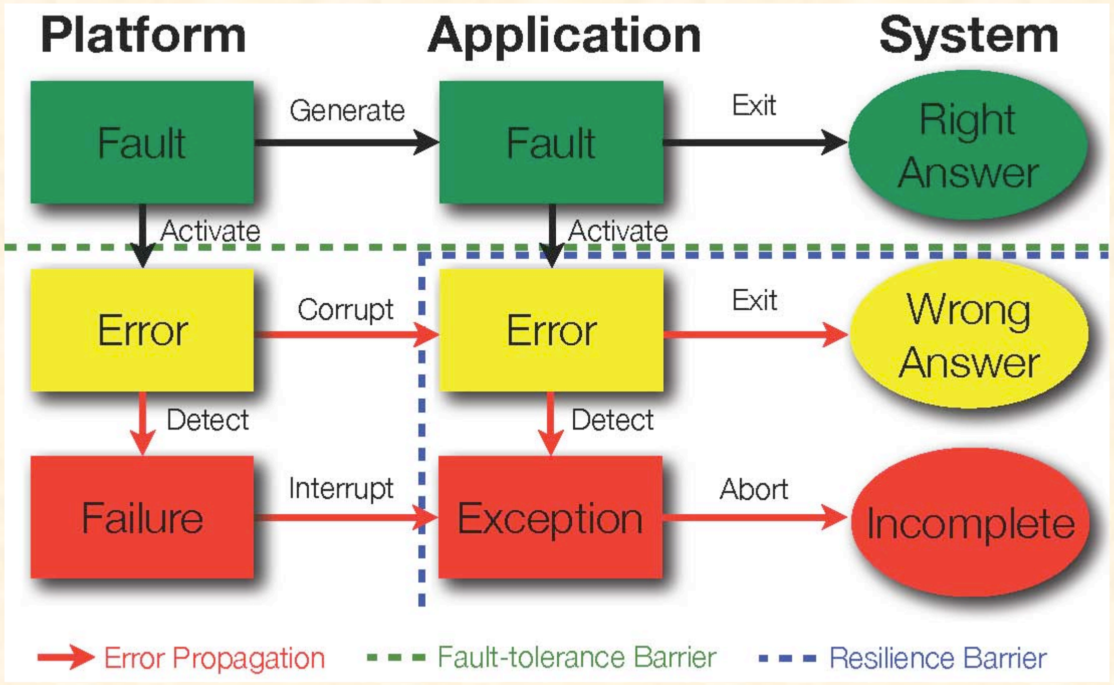
- Lord Kelvin: *"If you can't measure it, you can't improve it!"*
- Agreed upon definitions, metrics and methods
  - System vs. application MTTI, MTTR and availability/efficiency
- Dependability analysis
  - Fault injection studies using modeling and simulation
- Dependability benchmarking (robustness testing)
  - Fault injection studies using experimental evaluation
- Formal methods, statistics and uncertainty quantification

Enabling Infrastructure

- Programming models & libraries
  - Fault awareness and transparent fault tolerance
- System software
  - Reliable (hardened) system software (OS kernel, file systems)
- RAS systems and tools
  - System and application health monitoring
- Cooperation and coordination frameworks
  - Fault notification across software layers
  - Tunable resilience strategies
- Production solutions of existing resilience technologies
  - Enhanced recovery-oriented computing

Fault Prediction and Detection

- Statistical analysis
- Machine learning
- Anomaly detection
- Visualization
- Data & information collection



Monitoring and Control

- Non-intrusive, scalable monitoring and analysis
  - Decentralized/distributed scalable RAS systems
- Standards-based monitoring and control
  - Standardized metrics and application/system interfaces
- Tunable fidelity
  - Adjustable resilience/performance/power trade-off
  - Variety of resilience solutions to fit different needs
- Quality of service and performability
  - Measure-improve feedback loop at various granularities

End-to-End Data Integrity

- Confidence in getting the right answer and using correct data to make informed decisions
- Protection from undetected errors that corrupt data/code
  - Understanding root causes and error propagation
- Mitigation strategies against silent code/data corruption
  - Application-level checks
  - Self-checking code and selective software ECC
  - Redundant multi-threading and process pairs

Conclusions

- Faults and fault recovery will be continuous at exascale
- Co-design tools are required to understand the resilience, performance and power consumption trade-off
- Practical, tunable resilience solutions are needed to adjust the trade-off during design and at runtime
- Agreed upon definitions, metrics and benchmarks are needed to measure improvement and to compare fairly
- The occurrence and impact of undetected errors needs to be better understood to deploy mitigation techniques

Further References

- N. DeBardeleben, J. Laros, J. T. Daly, S. L. Scott, C. Engelmann, and B. Harrod. *High-End Computing Resilience: Analysis of Issues Facing the HEC Community and Path-Forward for Research and Development.* NSF whitepaper, 2009.
- C. Wang, F. Mueller, C. Engelmann, and S. L. Scott. *Hybrid Checkpointing for MPI Jobs in HPC Environments*. In Proceedings of ICPADS 2010.
- M. Li, S. S. Vazhkudai, A. R. Butt, F. Meng, X. Ma, Y. Kim, C. Engelmann, and G. Shipman. *Functional Partitioning to Optimize End-to-End Performance on Many-Core Architectures*. In Proceedings of SC 2010.
- C. Wang, F. Mueller, C. Engelmann, and S. L Scott. *Proactive Process-Level Live Migration and Back Migration in HPC Environments.* Journal of Parallel and Distributed Computing 72(2):254-267, 2012.
- C. Engelmann and S. Böhm. Redundant Execution of HPC Applications with MR-MPI. In Proceedings PDCN 2011.
- S. Böhm and C. Engelmann. *xSim: The Extreme-Scale Simulator*. In Proceedings of HPCS 2011.

UT-BATTELLE

Runtime Systems and System Software Challenges:

Resilience for Permanent, Transient, and Undetected Errors

Christian Engelmann

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