Resilience Challenges at the Exascale

Christian Engelmann

Computer Science and Mathematics Division
Oak Ridge National Laboratory

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

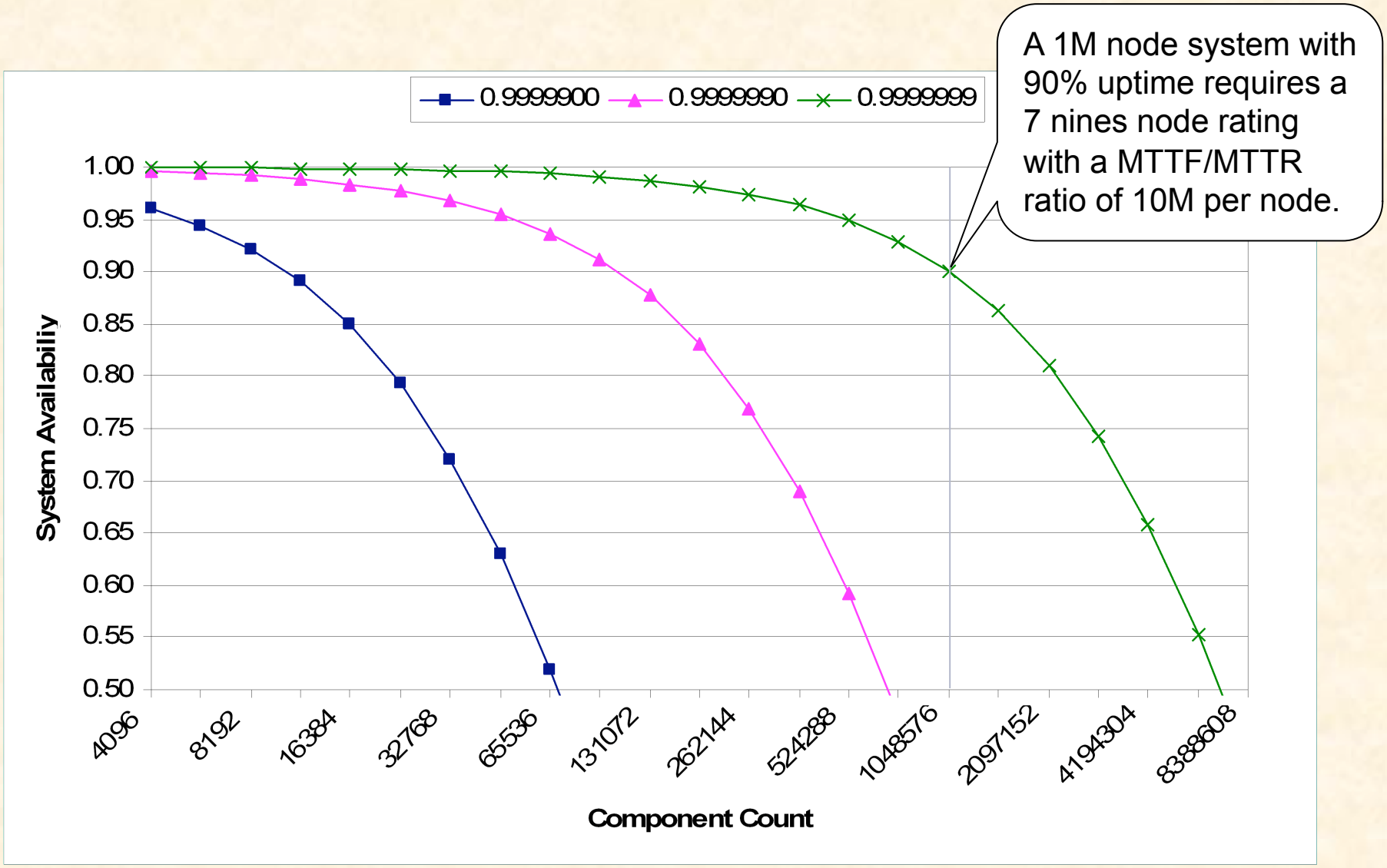
Factors Driving up the Error Rate

- Significant growth in component count (up to 50x nodes) results in respectively higher system 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 costs
- Heterogeneous architectures (CPU & GPU cores) add more complexity to fault detection and recovery

Risks of the Business as Usual Approach

- Increased error rate requires more frequent checkpoint/ restart, thus lowering efficiency (application progress)
- Current application-level checkpoint/restart to a parallel file system is becoming less efficient and soon obsolete
- Memory to I/O ratio (dump time) improves from 25 min to 8.3 min, but concurrency for coordination and I/O scheduling increases significantly (50x nodes, 444x cores)
- Missing strategy for silent data/code corruption will cause applications to produce erroneous results or just hang

System Availability with Checkpoint/Restart



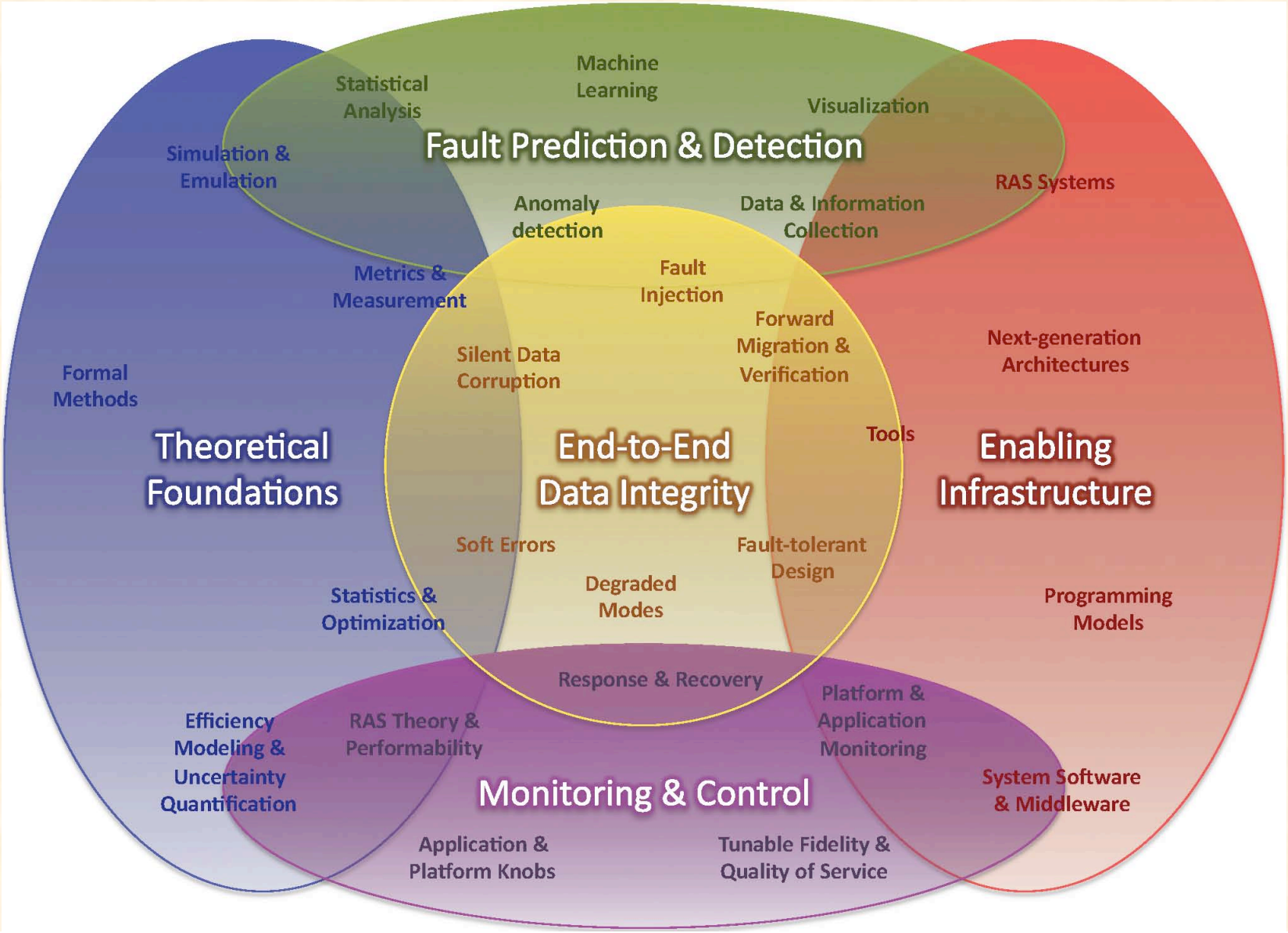
Existing HPC Resilience Technologies

- Checkpoint/restart (C/R)
  - SSD in Cray X/Y-MP (1982/88) and IBM 3090 (1985)
  - Networked disk storage in Intel Paragon XP/S (1992)
  - Local & networked disk storage in ASCI White (2000)
  - Networked disk storage in Cray XT and IBM BG (2000+)
- Application-level C/R dominates in practice
- System-level C/R
  - Libckpt (1995), CoCheck (1996), Condor (1997), BLCR(2003)
- Diskless C/R
  - Plank et al. (1997), Charm++/AMPI (2004), SCR (2009)
- Fault-tolerant message passing
  - PVM 3 (1993), Starfish MPI (1999), FT-MPI (2001), MPI-3 (?)

Existing HPC Resilience Technologies

- Message logging
  - Manetho (1992), Egida (1999), MPICH-V (2006)
- Algorithm-based fault tolerance (ABFT)
  - Huang et al. (1984), Chen et al. (2006), Ltaief et al. (2007)
- Proactive fault tolerance
  - Nagarajan et al. (2007), Wang et al. (2008)
- Log-based failure analysis and prediction
  - hPREFECT (2007), Sisyphus (2008)
- Soft-error resilience
  - Parity memory in Cray-1 (1977)
  - ECC memory in Cray X-MP (1982)
  - ECC for caches and registers in AMD Opteron (2007)

Key Areas for Future Research, Development, and Standards Work



Theoretical Foundations

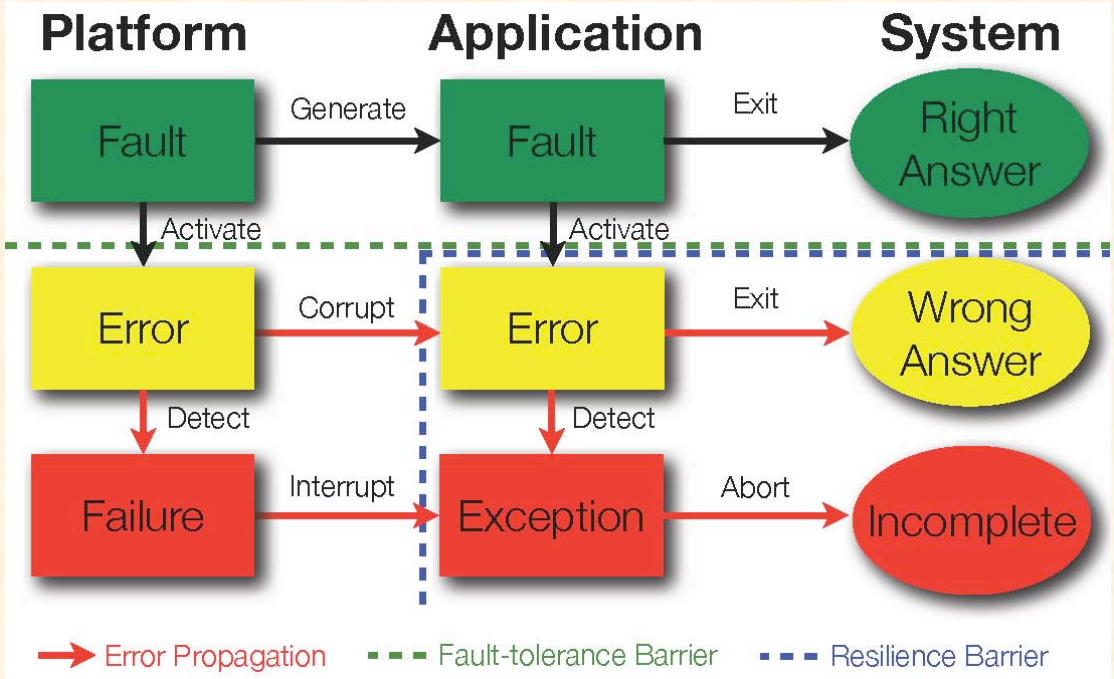
- Lord Kelvin: *"If you can't measure it, you can't improve it!"*
- Agreed upon definitions, metrics and methods
  - System vs. application MTTI, MTTR and availability/efficiency
- Dependability analysis
  - Fault injection studies using modeling and simulation
- Dependability benchmarking (robustness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

Conclusions

- Current resilience methods will be unpractical at exascale
- Alternatives need to be developed into practical solutions
- Agreed upon definitions, metrics and benchmarks are needed to measure improvement and to compare fairly
- Root causes and propagation are not well understood
  - No effective fault detection and prediction
- Resilience is needed across the entire software stack
  - System software, programming models, apps and tools
  - Communication/coordination between layers
- Faults and fault recovery will be continuous
- Tunable solutions to adjust resilience/performance/power

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***
- **Scientific Grand Challenges Workshop Series: *http://extremecomputing.labworks.org/***
- **International Exascale Software Project: *http://www.exascale.org/***

Questions?