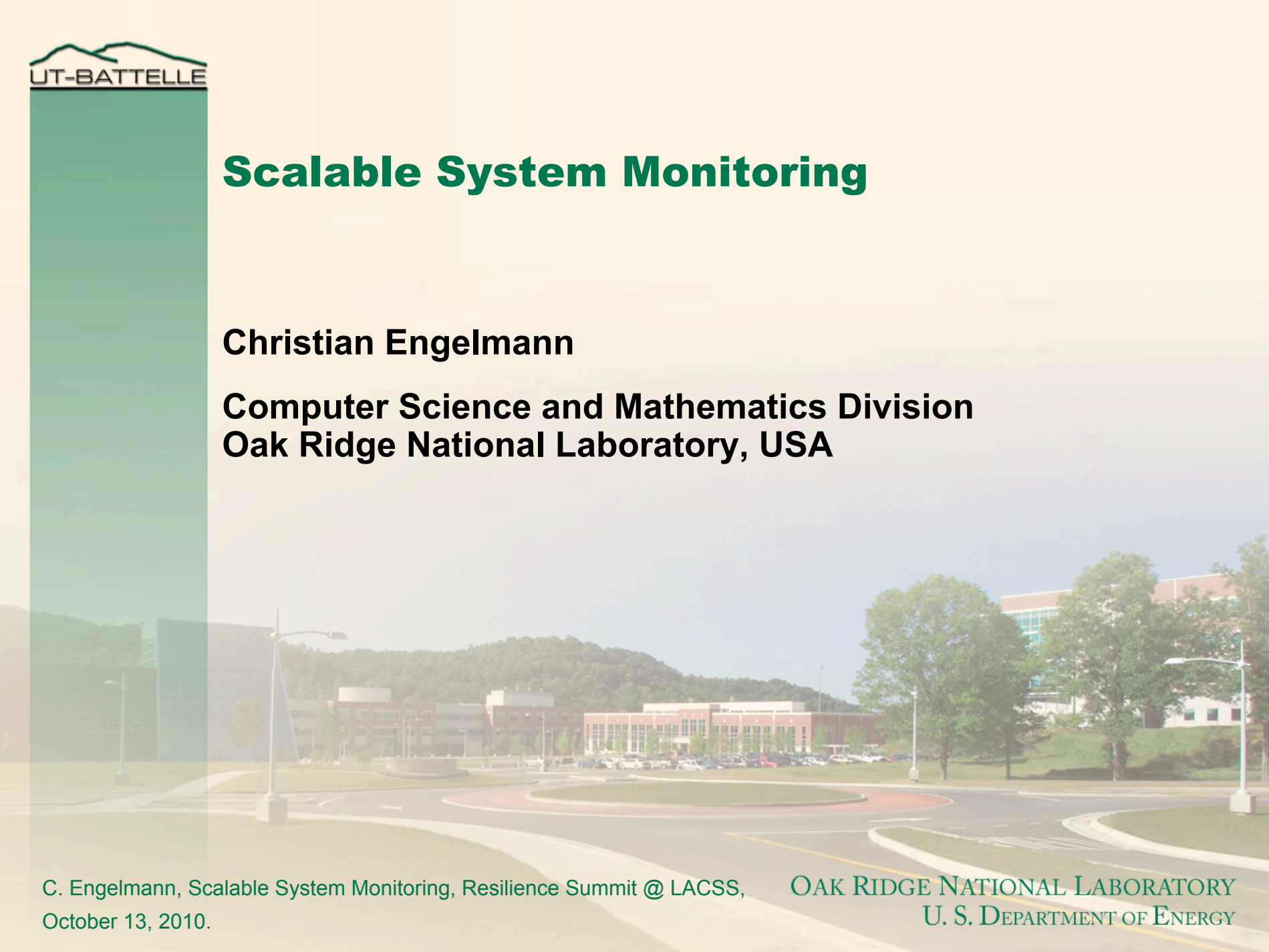
UT-BATTELLE

# Scalable System Monitoring

**Christian Engelmann**

**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


#3



OAK RIDGE National Laboratory

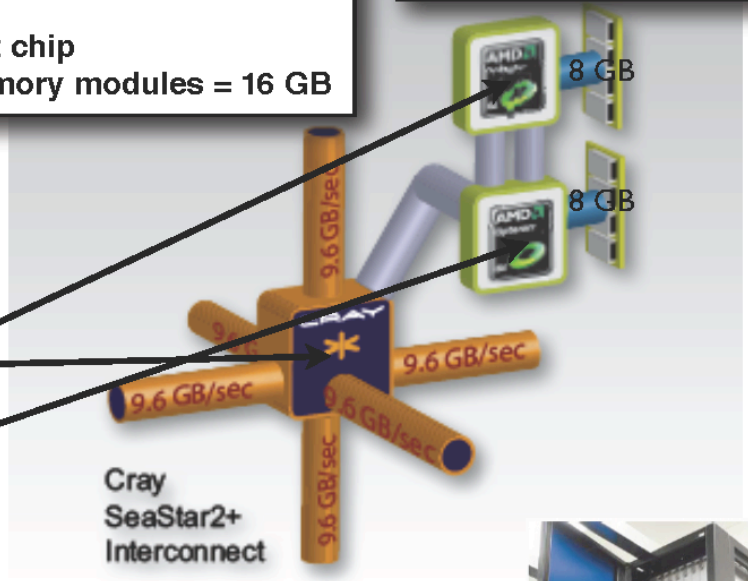
# #1: Jaguar at Oak Ridge National Laboratory

**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 |

# 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

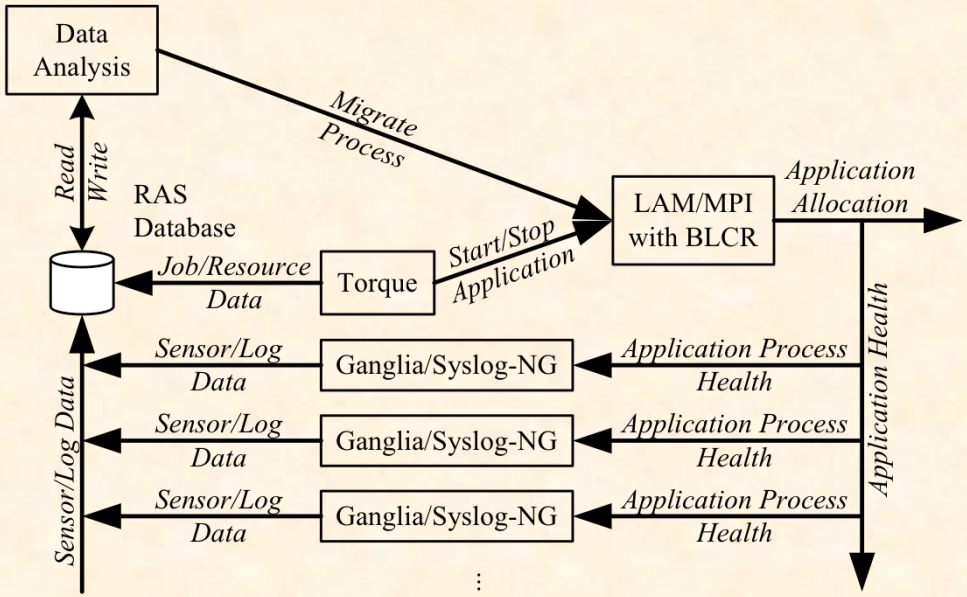
- Increased error rate requires more frequent checkpoint/ restart, thus lowering efficiency (application progress)
- Memory to I/O ratio improves due to less memory/node, but concurrency for coordination and scheduling increases significantly (up to 50x nodes, 444x cores)
- Current application-level 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

# Objectives

- Non-stop scientific high-performance computing
- Develop scalable system software technologies for next-generation petascale computing resources
- Address the computer science challenges for extreme-scale computing:
  - High-level RAS for enhanced productivity
  - Transparent system software support for resilience
  - Model causes and propagation of failures and errors
- As part of this greater effort, this work targets:
  - Scalable system monitoring to support health analysis and anomaly (failures, errors and indicators) reporting

# Our Initial Efforts: Proactive Fault Tolerance Framework

- Central MySQL database
- Environmental monitoring
  - OpenIPMI and Ganglia
- Event logging and analysis
  - Syslog forwarding
- Job & resource monitoring
  - Torque (epilogue/ prologue)
- Migration mechanism
  - Process-level with BLCR



# Our Initial Efforts: System Monitoring with Ganglia and Syslog

**Experiment #1:**

- 32-node Linux cluster
- 30 second interval
- 40 Ganglia metrics
- ≈20 GB of data in 27 days
- ≈33 MB/hour
- ≈1 MB/hour per node
- ≈275 kb/interval
- NOT SCALABLE

**Experiment #2:**

- 32-node Linux cluster
- 30 second interval
- 40 Ganglia metrics
- No measurable impact on NAS benchmarks

| Class C NPB on 32 nodes | CG | FT | LU |
|---|---|---|---|
| Average time in seconds | 264 | 235 | 261 |
| Average time under load in seconds | 264 | 236 | 260 |

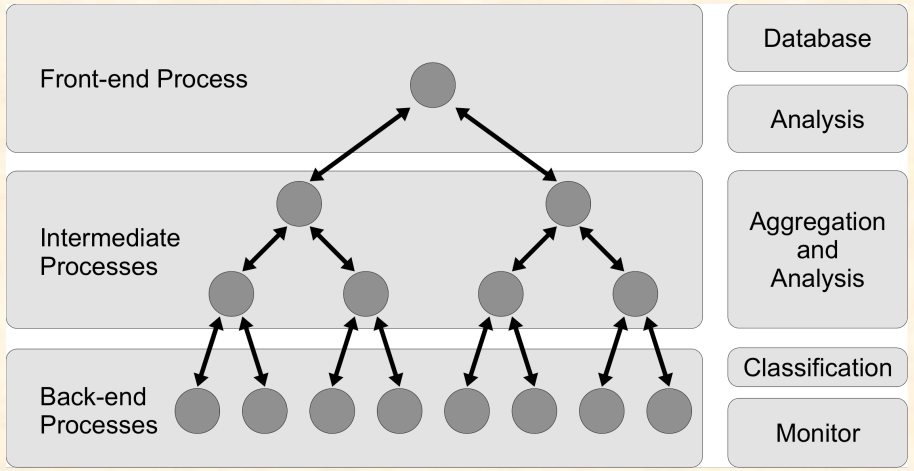
Table 2. NPB test results (averages over 10 runs)

# Other Related Work

- Ganglia and Nagios:
  - IT monitoring using network multicast or linear query
  - Support for hierarchical grouping of monitored nodes
  - Does not scale beyond 2,000 nodes due to data size
- HPC vendor RAS systems (e.g. Cray and IBM):
  - HPC system monitoring similar to Ganglia with scalability enhancements and SQL support
  - Scalability limits reached today with 10,000-100,000 nodes (operating system instances)
- Multicast Reduction Network (MRNet)
  - Generic tree-based over-lay network (TBON)
  - Recent work on aggregating node-local Ganglia data files in fan-in tree using group file operations

# Technical Approach

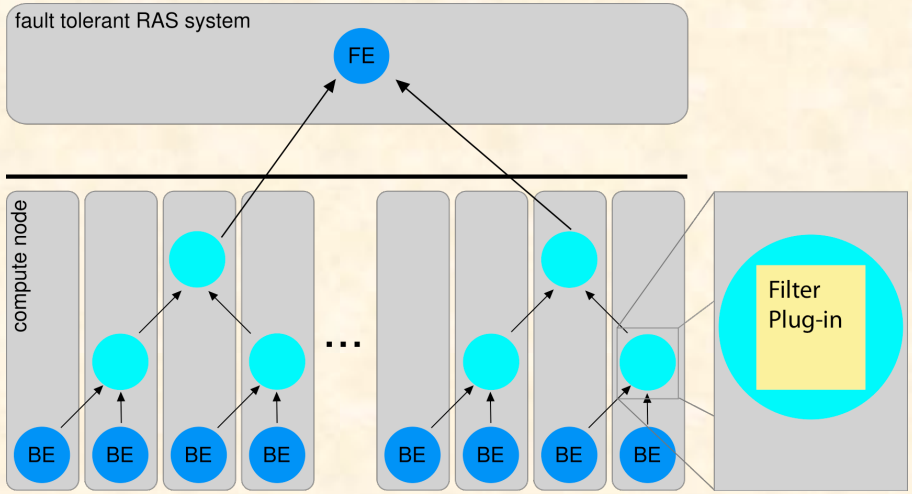
- Classify monitoring metrics locally before transmission to reduce monitoring data
- Leverage a TBON with its in-flight processing to reduce data further
  - Data aggregation using the fan-in tree
  - System configuration using the fan-out tree

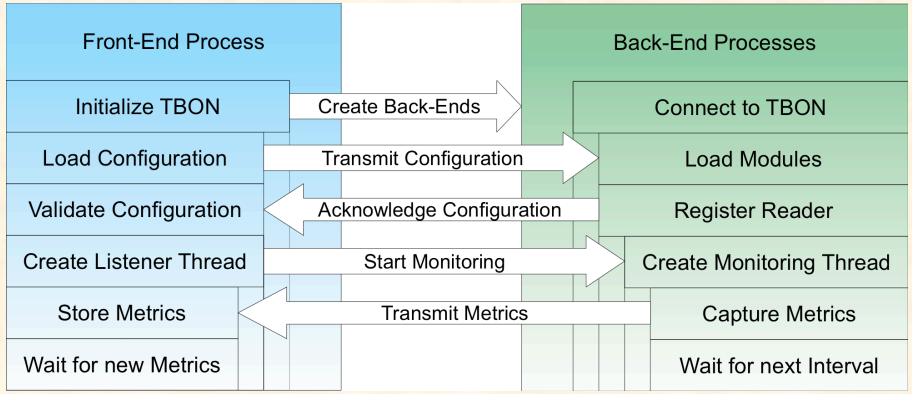



- Back-ends gather data
  - Sample and classify
- Intermediates reduce data
  - Group and compress
- Front-end stores data

# Implementation

- Deploy MRNet for the TBON
- Connect monitor processes at the MRNet back-ends
- Insert aggregation plugins at the MRNet intermediates
- Connect a database writer and configuration tool at the MRNet front-end
- Implemented in C++ using the Boost C++ libraries
- C++ interface to MySQL for the database writer



# Implementation Details: Front-End

- Located on the RAS management node (also often the head node running the job and resource management)
- Responsible for setting up the TBON, including the intermediates and back-ends
- Responsible for storing the received data in a MySQL database
- Instantiation: Boots up MRNet, configures the streams, and loads and configures needed modules
- Reconfiguration: Shutdown, reconfigure, and startup (runtime reconfiguration is currently in development)
- Only metric class updates are received and stored with the appropriate time stamp

# Implementation Details: Back-ends

- Located on the nodes to be monitored
- Gather metrics using the */proc* file system and and *libsensors* (Intelligent Platform Management Interface)
- Collection, classification, and transmission is performed in regular intervals for each metric
- Overlapping metric intervals and respective outgoing messages are aggregated
- One message per interval with metric IDs and classes
- Messages contain updates only


Figure 5. Message format for transferring metric updates from back-ends to intermediates

# Implementation Details: Intermediates

- Located on the same nodes as the back-ends, or on existing separated hierarchical RAS subsystems
- Configured for synchronous operation to process wave fronts of messages coming from back-ends
- Aggregation plug-ins (primitive forwarding filters) loaded at configuration time that simply attach incoming messages to each other
- In-flight processing, such as statistical analysis is in development

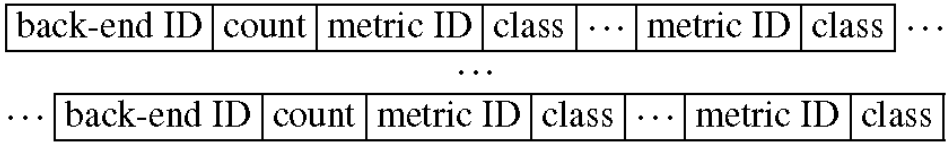


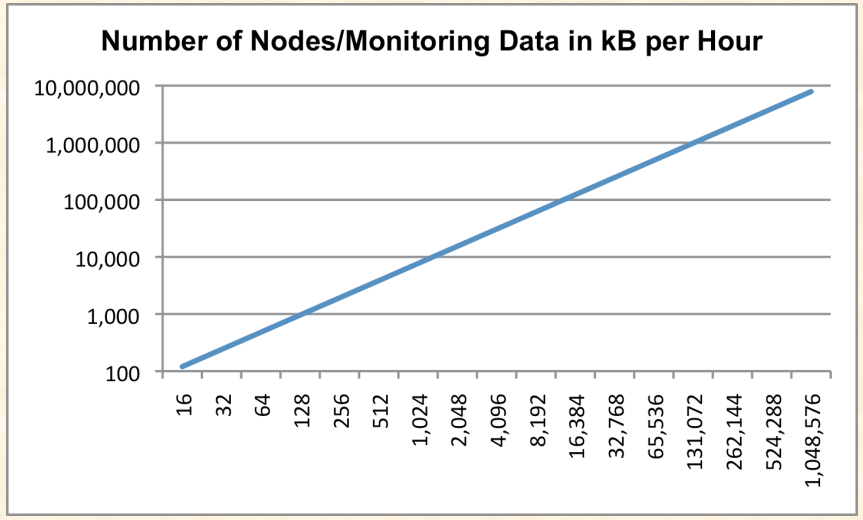

Figure 6. Message format for aggregating metric updates by the intermediates

# Experimental Results: Monitoring Data Accumulation

- 8-year old 32-node Intel-based Linux cluster
  - Perfect test environment due to failing hardware
- 18 monitoring metrics:
  - Processor heat and utilization, memory utilization, fan speeds, disk and network I/O rates, …
- 30 second sample interval for all metrics
- Collected data at the head node:
  - ≈ 1MB in 4 hours
  - ≈ 250kB/hour
  - ≈ 2kB/interval
  - ≈ 56x less data than collected by Ganglia

# Experimental Results: Data Accumulation Scaling Analysis

- Worst-case scenario:
  - Monitoring data amount scales with system size
- On 32 nodes, the measured accumulation rate was:
  - 64 B/interval per node
- The theoretical rate for 100,000 nodes is:
  - 6.1 MB/interval
  - 732 MB/hour
- The theoretical rate for 1,000,000 nodes is:
  - 61 MB/interval
  - 7.2 GB/hour




- The accumulation rate is:
  - Acceptable for off-line (post mortem) analysis
  - Too high for realistic real-time response (on-line) scenarios

# Experimental Results: Performance Impact on Applications

- The monitoring system performs communication and computation on nodes
- There is a potential for performance degradation
- Performed NAS Parallel Benchmark (NPB) suite runs data during collection
- No measurable overhead at this scale (32 nodes)
- No measurable overhead with Ganglia as well

| Class C NPB on 32 nodes | CG | FT | LU |
|---|---|---|---|
| Without monitoring | 264 | 235 | 260 |
| With monitoring | 264 | 235 | 260 |
| Overhead | 0% | 0% | 0% |

Table 1. NPB performance on the test cluster with/without the developed monitoring system (averages over 10 runs in seconds)

- 32-node test system scale is too small to impact application performance
- Ongoing work focuses on much larger-scale tests

# Conclusions and Future Work

- Developed a monitoring system that allows to trade-off accuracy in a tunable fashion to gain scalability without compromising fidelity
- The approach relies on classifying each metric and on aggregating messages in a fan-in tree fashion
- The prototype was able to reduce the amount of collected data by a factor of 56 in comparison to Ganglia
- A simple scaling study revealed that further data reduction is needed for monitoring extreme-scale systems
- Ongoing work focuses on in-flight statistical analysis and system log message aggregation and reduction
  - Histograms for metrics and pattern matching for syslog

UT-BATTELLE

# Questions?

**More details can be found at:**

S. Böhm, C. Engelmann, and S. L. Scott. *Aggregation of Real-Time System Monitoring Data for Analyzing Large-Scale Parallel and Distributed Computing Environments*. In Proceedings of HPCC 2010.