Extreme-Scale Resilience for Billion-Way of Parallelism

SC14 Workshop: ATIP Workshop on Japanese Research Toward Next-Generation Extreme Computing
11/17/2014

SC14 New Orleans, LA hpc matters.

Kento Sato
Lawrence Livermore National Laboratory

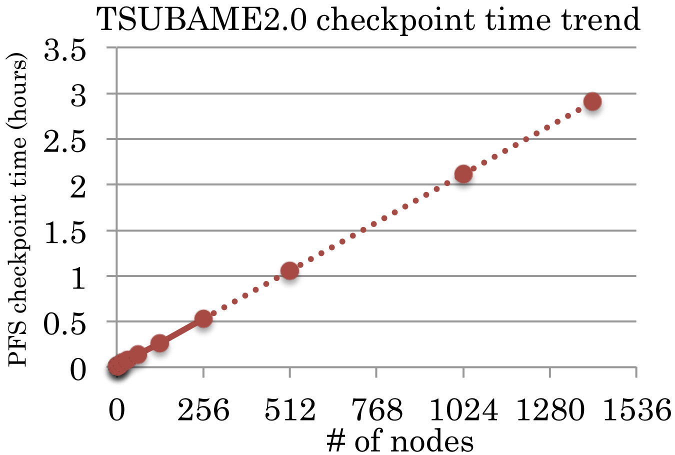
Lawrence Livermore National Laboratory

東京工業大学
Tokyo Institute of Technology

LLNL-PRES-664262
This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. Lawrence Livermore National Security, LLC

Failures on HPC systems

- System resiliency is critical for future extreme-scale computing
- MTBF of supercomputers
  - LLNL (Hera, Atlas & Coastal): 1.2 days[1]
  - Blue Waters: 8-12 hours[2]
  - Titan: 8-12 hours (<= a few failures/day[2])
- MTBF is shrinking
  - MTBF is projected to shrink to a few hours



- Checkpoint/Restart is a popular way for fault tolerance
- Simple checkpoint/restart may not work at extreme scale

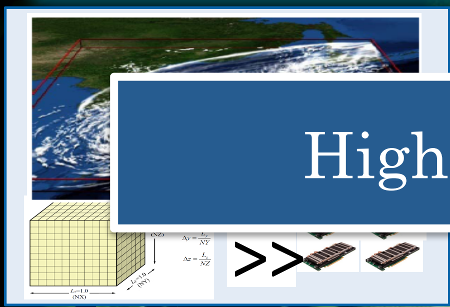
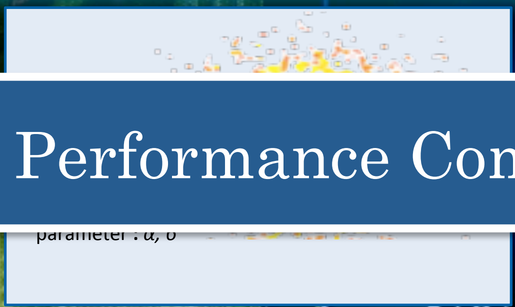
[1] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System," in Proceedings of the 2010 ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis, ser. SC '10. Washington, DC, USA: IEEE Computer Society, Nov. 2010, pp. 1–11. [Online]. Available: http://dx.doi.org/10.1109/SC.2010.18
[2] Yves Robert, "Fault_Tolerance Techniques for Computing at Scale", Keynote Talk, CCGrid2014
[3] Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of a Non-blocking Checkpointing System", SC12

Tokyo Tech. Billion-Way Resilience Project (2011-2015)

- PI: Satoshi Matsuoka
- Current collaborations:
  - ANL (Franck Cappello, FTI), LLNL (Bronis de Spinksi, SCR), ETH Zurich (Torsten Hoefler), RIKEN (Naoya Maruyama), U-Tokyo (Hideyuki Jitsumoto) …
- Objective: Scalable fault tolerance techniques for extreme scale system
  - API & Software: Encoding/Redundancy technique, Compression, Support for Many-core architecture
  - Architecture: Scalable Storage design, and Resilient network/interconnects
  - Analysis: Failure analysis, and Failure prediction
  - Modeling: Optimal checkpoint interval, Encoding/Redundancy, and I/O model

Argonne NATIONAL LABORATORY
Lawrence Livermore National Laboratory
理化学研究所 RIKEN
TOKYO TECH Pursuing Excellence
Inria INVENTEURS DU MONDE NUMÉRIQUE
ETH Zürich
東京大学 THE UNIVERSITY OF TOKYO

Tokyo Tech.
Billion-Way Resilience Project (2011-2015)

High Performance Computing Applications

API & software

- Encoding/Redundancy technique
- Checkpoint compression
- Asynchronous I/O APIs for checkpointing
- Support for Many-core architecture
- Resource manager & Scheduler for resilience

Model

- Models of checkpinting / I/O for optimal interval & performance prediction

Architecture

- Resilient storage design
- Resilient network design

Analysis

- Failure monitoring & analysis

Failure monitoring & analysis (2010 ~ present)

Failure history of TSUBAME2.0/2.5

Findings

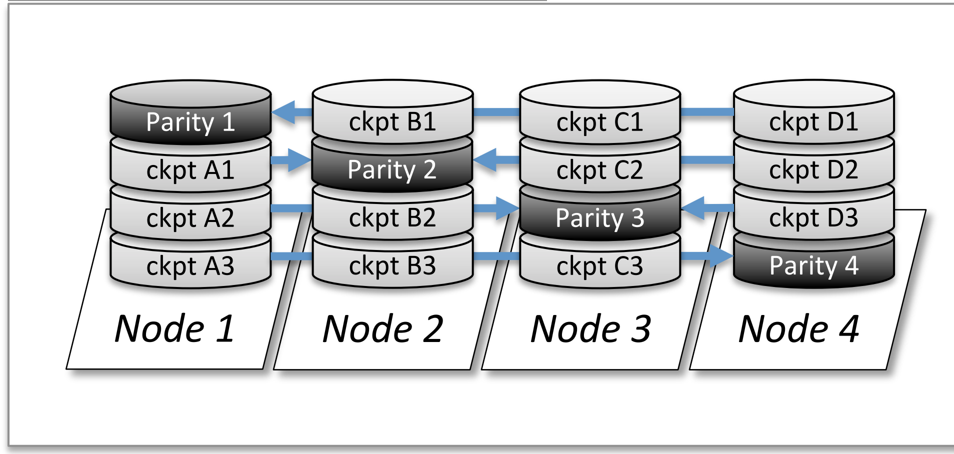
- Failures seasonal
  - Largely due to boot failures in peak-shift operations during summer to limit power, despite SW retries
  - Future SCs in Clouds need to cope with this
- GPU vs. CPUs
  - 19 CPU+memory fail-stop failures, 25 replacements, MTBF 118 years, 2.22^{18} FLOP/error
  - 53 GPU+memory ECC fail-stop failures, 57 replacements, MTBF 75 years, 1.61^{19} FLOP/error
  - GPU error rate x7 better / flop vs. CPU, proportional to performance difference per chip
- Failures are Largely Independent
  - Most of failures are a single node
  - Low # of InfiniBand and storage failures

松岡 聡, 佐藤 賢斗, 遠藤敏夫 "エクサスケールスパコンに向けた耐故障性の評価 --- TSUBAME2.0を例にして ---", IPSJ SIG Technical Reports 2013-HPC-141, Okinawa, Sep, 2013.

FTI: High Performance Fault Tolerance Interface

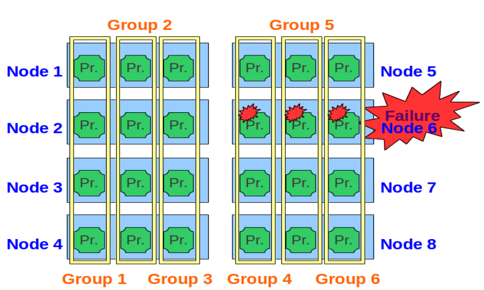
[SC11, EuroPar12 & Cluster12 (Leonardo Bautista-Gomez et al.)]

Diskless checkpointing

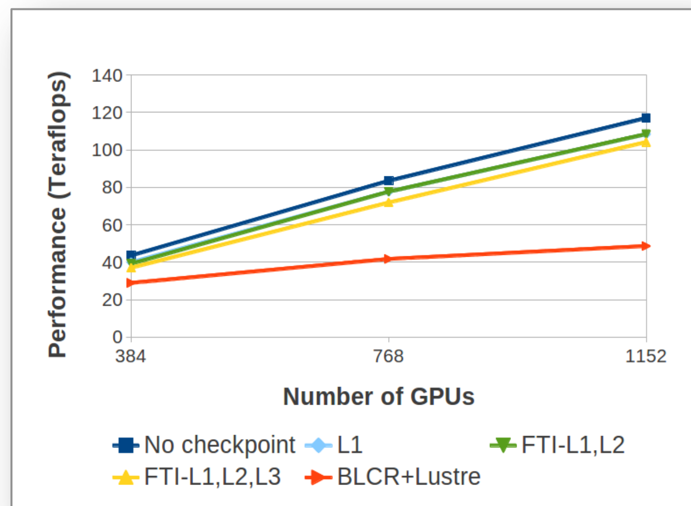


- Diskless checkpoint:
  - Create redundant data across local storages on compute nodes using a encoding technique such as Reed-solomon, XOR
    - Scalable by using distributed disks
  - Can restore lost checkpoints on a failure caused by small # of nodes like RAID-5

Diskless checkpoint runtime library using Reed-Solomon encoding

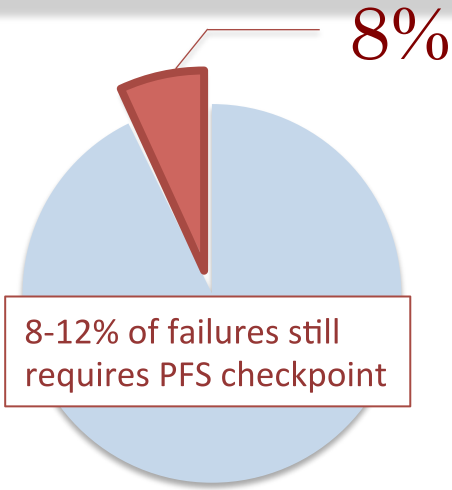


- FTI implements a scalable Reed-Solomon encoding algorithm by utilizing local storages such as SSD
- FTI analyzes the topology of the system and create encoding clusters that increase the resilience



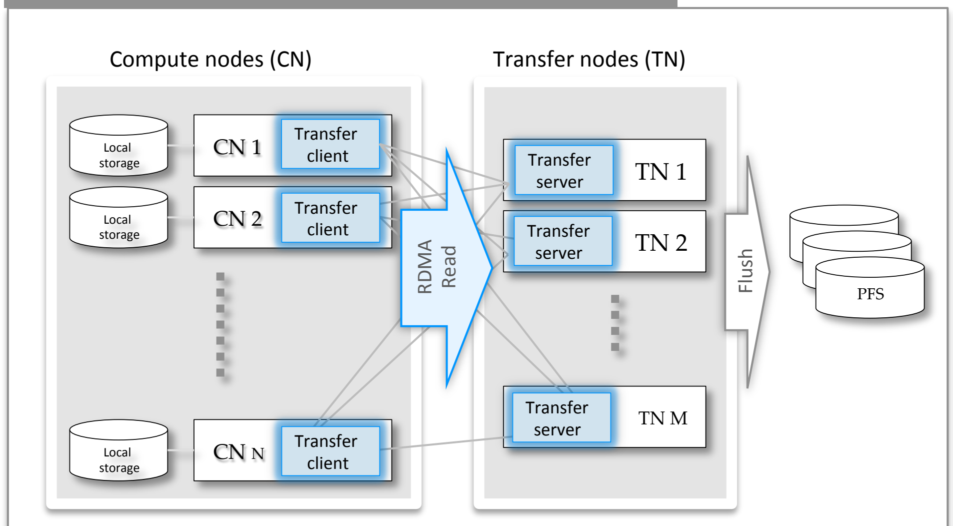
Design and Modeling of Async. Checkpointing
[SC12, Kento Sato et al.]

- **<u>Objective</u>**: Minimize checkpoint overhead to PFS
  - Minimize CPU usage, memory and network bandwidth
- **<u>Proposed method</u>**: Implementation and modeling Non-blocking checkpointing
  - Asynchronously write checkpoints to PFS through Staging nodes using RDMA
  - Determine the optimal checkpoint interval on the asynchronous checkpoint scheme

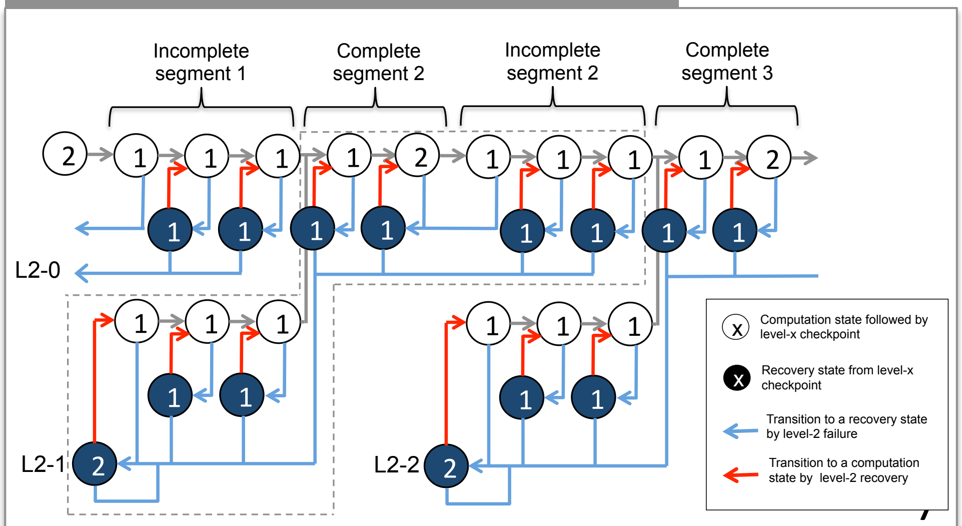


Failure analysis on TSUBAME2.0

Async. checkpointing system



Async. checkpointing model



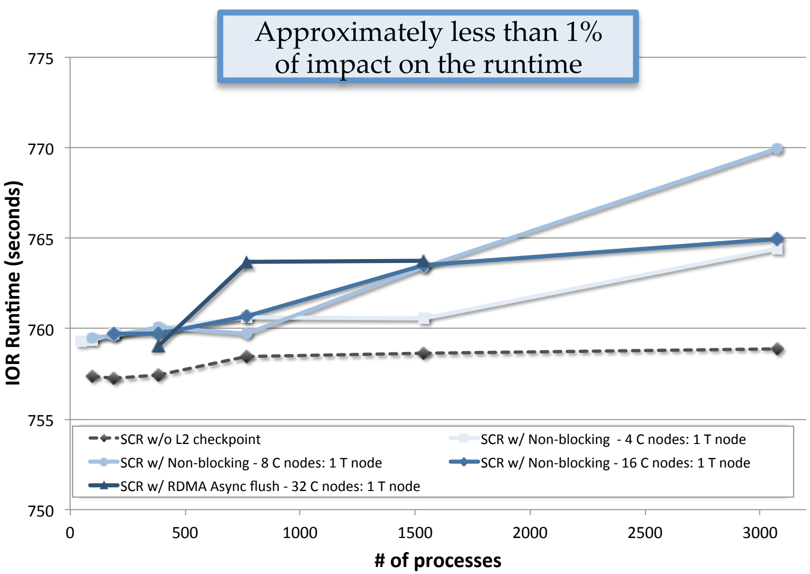
Design and Modeling of Async. Checkpointing

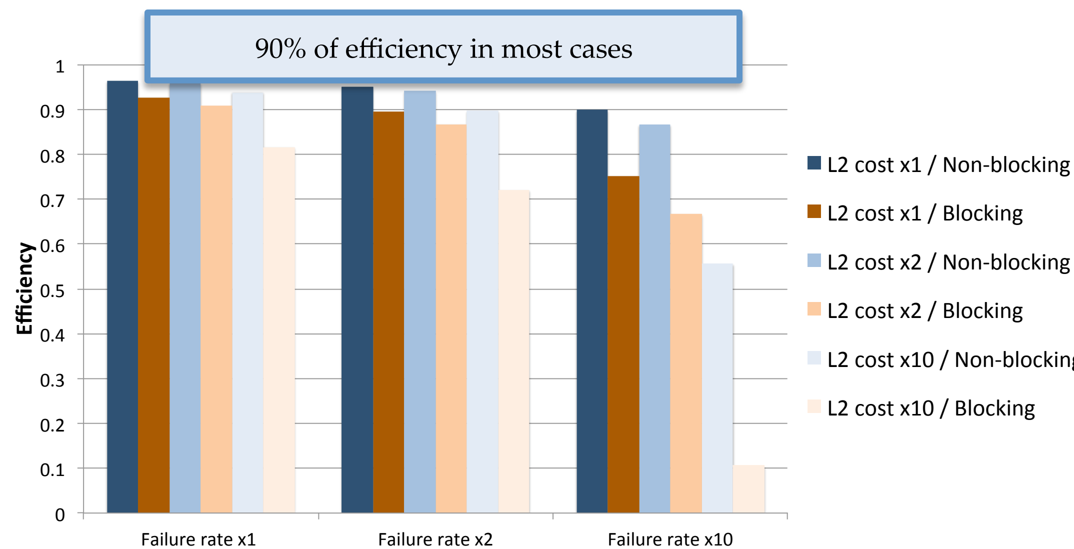
[SC12, Kento Sato et al.]

- **Experiment:**
  - Benchmark: CPU-bound micro benchmark
  - Method
    - Non-blocking: proposed method on two level checkpointing
      - L1: XOR checkpoint, L2: Proposed non-blocking checkpoint
    - Blocking: Existing two level checkpointing
      - L1: XOR checkpoint, L2: Blocking checkpoint

$$Efficiency = \frac{ideal\_time}{expected\_time}$$

- **Results**:
  - Asynchronous RDMA checkpoint: About 1 % of overhead with the proposed checkpointing
  - Optimal checkpoint interval: Achieved high efficiency even with increasing failure rates

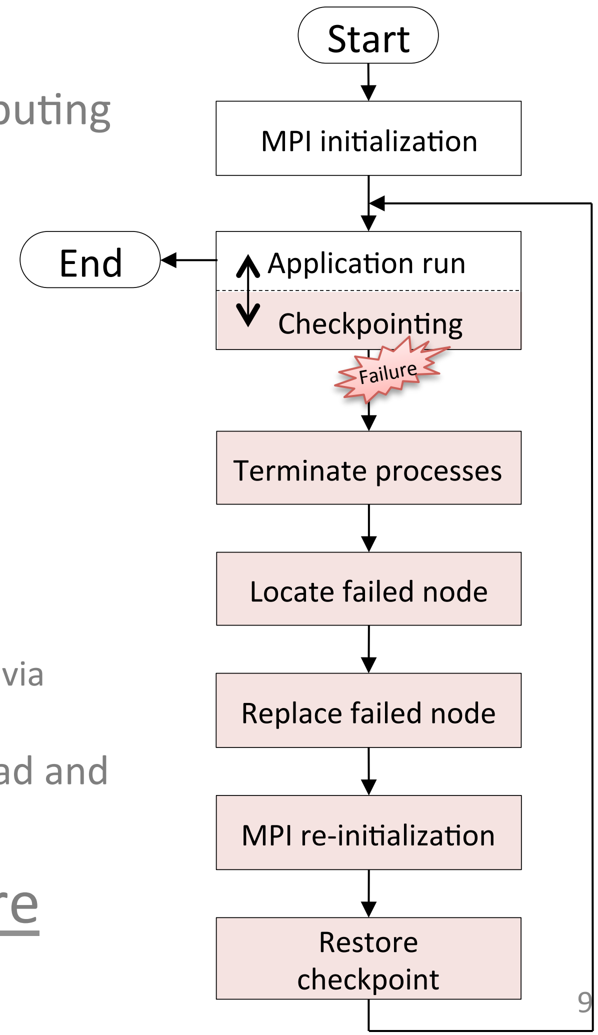




Requirement of fast and transparent recovery

- MPI
  - De-facto communication library enabling parallel computing
  - Standard MPI employs a fail-stop model
- When a failure occurs …
  - MPI terminates all processes
  - The user locate, replace failed nodes with spare nodes
  - Re-initialize MPI
  - Restore the last checkpoint
- Applications will use more time for recovery
  - Users manually locate and replace the failed nodes with spare nodes via `machinefile`
  - The manual recovery operations may introduce extra overhead and human errors

⇒ APIs for transparent, but fast recovery are critical

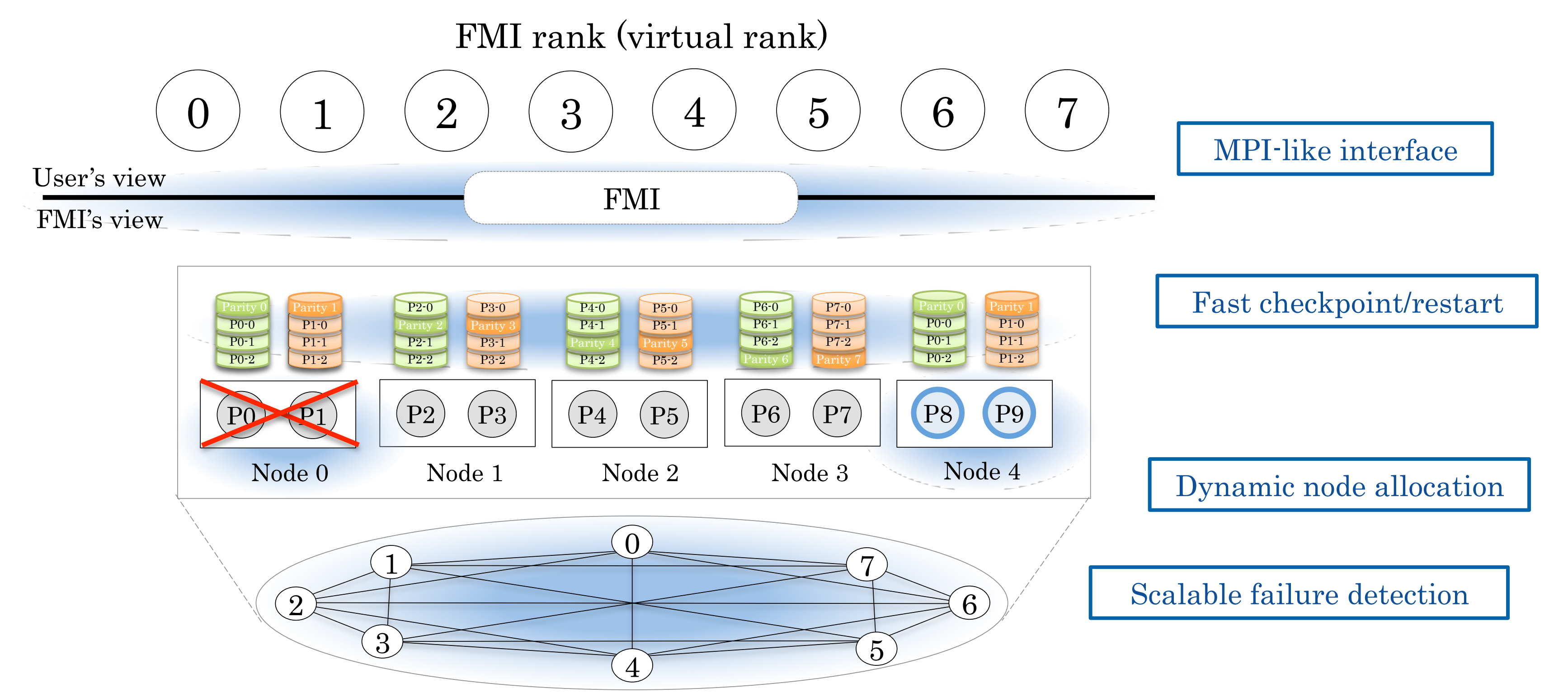


FMI: Fault Tolerant Messaging Interface

[IPDSP2014, Kento Sato et al.]

FMI for Fast and transparent recovery

FMI overview



- FMI is a survivable messaging interface providing MPI-like interface
  - Scalable failure detection ⇒ Overlay network
  - Dynamic node allocation ⇒ FMI ranks are virtualized
  - Fast in-memory checkpoint/restart ⇒ Diskless checkpoint/restart

FMI: Fault Tolerant Messaging Interface

[IPDSP2014, Kento Sato et al.]

Example code & Evaluation

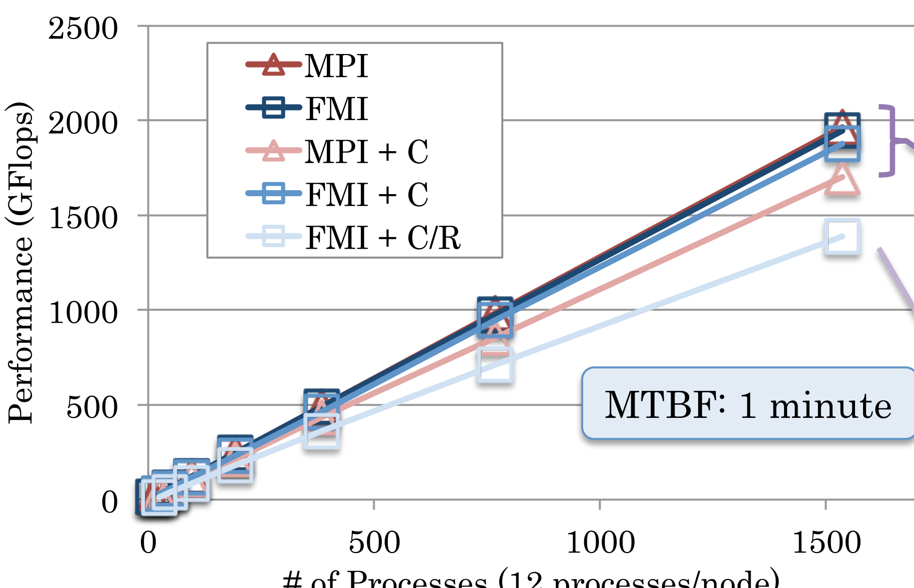
FMI example code

```
int main (int *argc, char *argv[]) {
  FMI_Init(&argc, &argv);
  FMI_Comm_rank(FMI_COMM_WORLD, &rank);
  /* Application’s initialization */
  while ((n = FMI_Loop(…)) < numloop) {
    /* Application’s program */
  }
  /* Application’s finalization */
  FMI_Finalize();
}
```

- FMI_Loop enables transparent recovery and roll-back on a failure
  - Periodically write a checkpoint
  - Restore the last checkpoint on a failure

P2P communication performance

	1-byte Latency	Bandwidth (8MB)
MPI	3.555 usec	3.227 GB/s
FMI	3.573 usec	3.211 GB/s



FMI directly writes checkpoints via memcpy, and can exploit the bandwidth

Even with the high failure rate, FMI incurs only a 28% overhead

Burst Buffers for Resilient Checkpoint/Restart

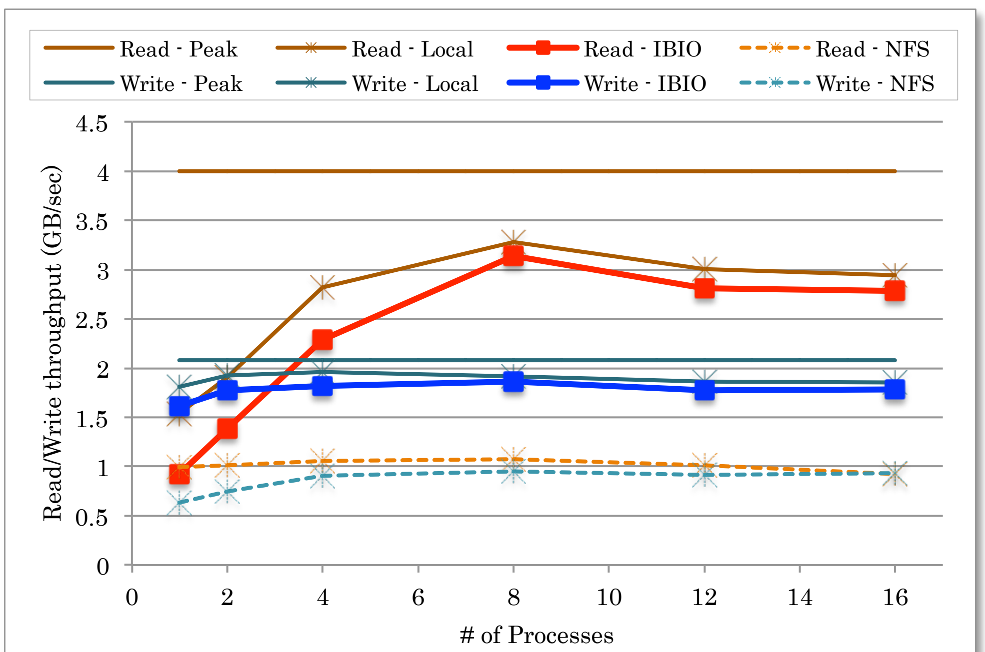
[CCGrid2014, Kento Sato et al.]

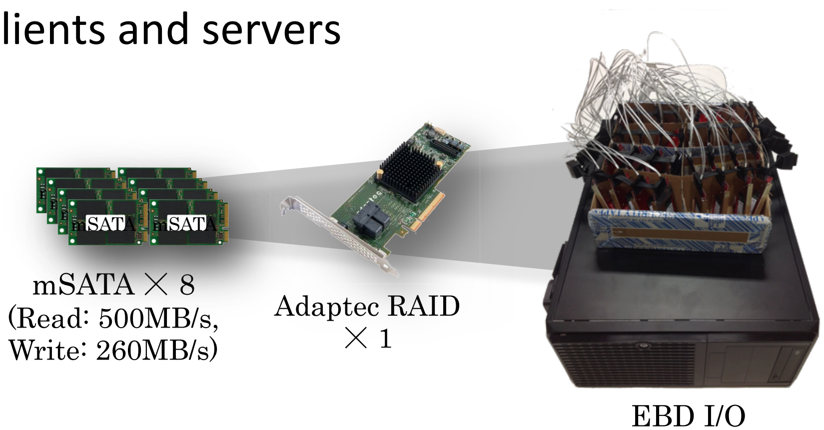
API
Modeling
Architecture

TSUBAME3.0 EBD Prototype mSATA High I/O BW, low power & cost

- Provide POSIX-like I/O interfaces
  - open, read, write and close
  - Client can open any files on any servers
- IBIO use ibverbs for communication between clients and servers
  - Exploit network bandwidth of infiniBand

```
open("hostname:/path/to/file", mode)
```





Node specification

CPU	Intel Core i7-3770K CPU (3.50GHz x 4 cores)
Memory	Cetus DDR3-1600 (16GB)
M/B	GIGABYTE GA-Z77X-UD5H
SSD	Crucial m4 msata 256GB CT256M4SSD3 (Peak read: 500MB/s, Peak write: 260MB/s)
SATA converter	KOUTECH IO-ASS110 mSATA to 2.5' SATA Device Converter with Metal Fram
RAID Card	Adaptec RAID 7805Q ASR-7805Q Single

Interconnect :Mellanox FDR HCA (Model No.: MCX354A-FCBT)

Burst Buffers for Resilient Checkpoint/Restart

[CCGrid2014, Kento Sato et al.]

Resilience modeling overview

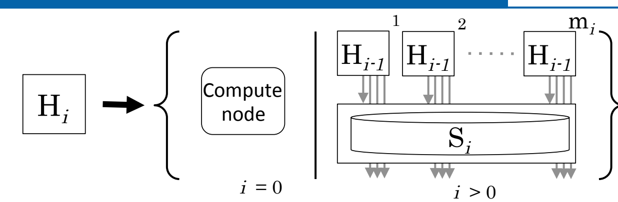
To find out the best checkpoint/restart strategy for systems with burst buffers, we model checkpointing strategies

C/R strategy model

$$O_i = \begin{cases} C_i + E_i & \text{(Sync.)} \\ I_i & \text{(Async.)} \end{cases} \qquad L_i = C_i + E_i$$

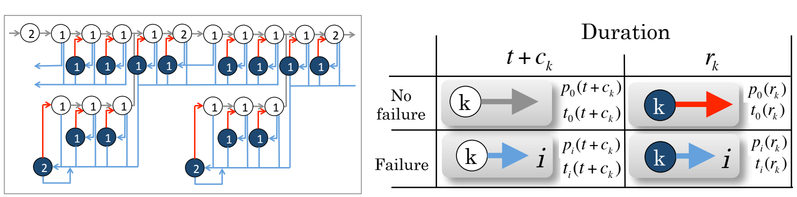
$$C_i \text{ or } R_i = \frac{\text{< C/R date size / node >} \times \text{<\# of C/R nodes per } S_i^* \text{ >}}{\text{< write perf. (} w_i \text{) > or <read perf. (} r_i \text{) >}}$$

Recursive structured storage model



Storage Model: $H_N \{m_1, m_2, \ldots, m_N\}$

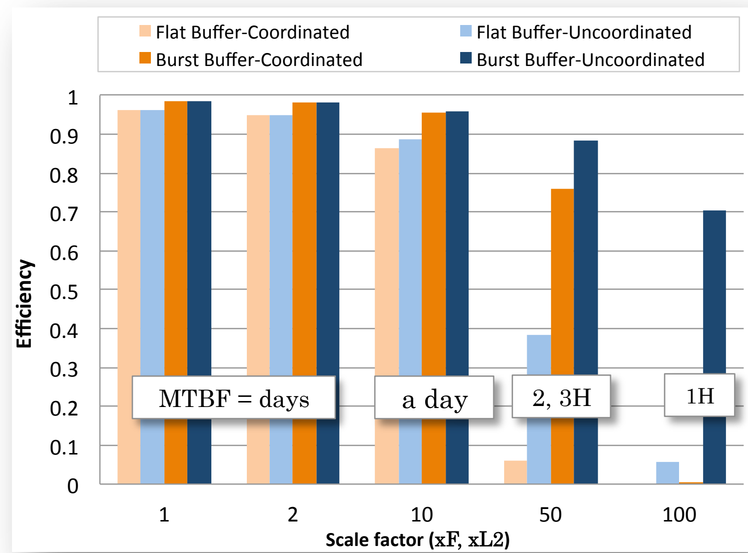
MLC model



t : Interval
c_c : c-level checkpoint time
r_c : c-level recovery time
λ_i : i-level checkpoint time

$$p_0(T) = e^{-\lambda T}$$
$$t_0(T) = T$$
$$p_i(T) = \frac{\lambda_i}{\lambda}(1 - e^{-\lambda T})$$
$$t_i(T) = \frac{1 - (\lambda T + 1) \cdot e^{-\lambda T}}{\lambda \cdot (1 - e^{-\lambda T})}$$

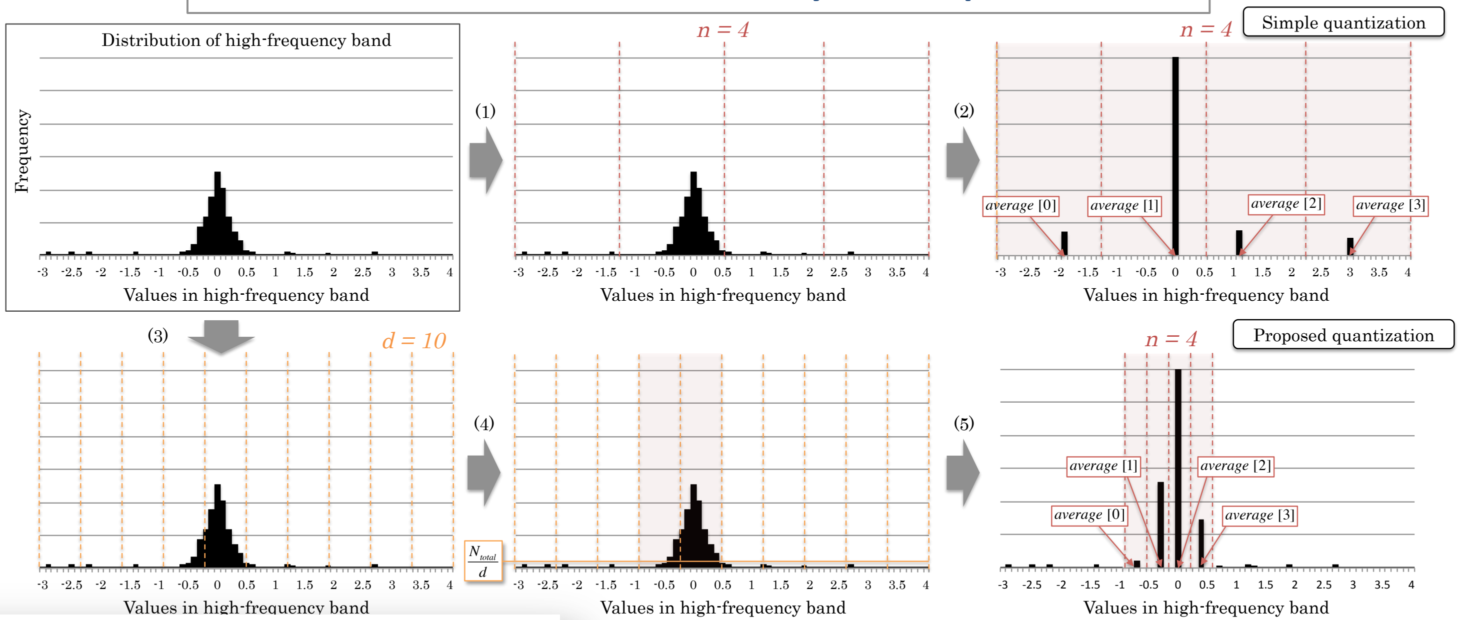
$p_0(T)$: No failure for T seconds
$t_0(T)$: Expected time when $p_0(T)$
$p_i(T)$: i - level failure for T seconds
$t_i(T)$: Expected time when $p_i(T)$

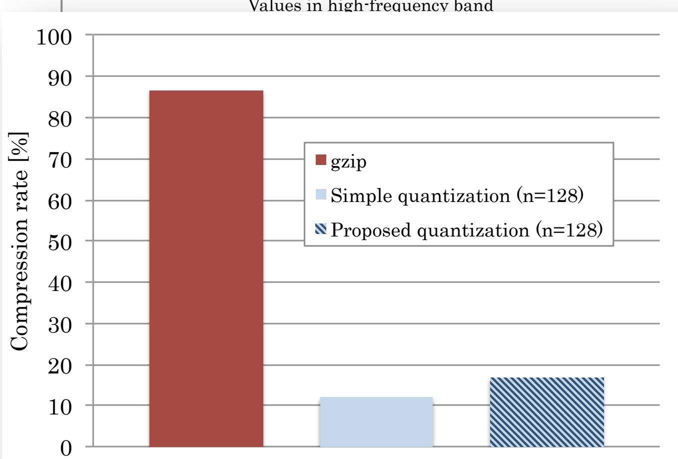


Lossy Floating-point compression

[Submitted to IPDPS2015, Naoto Sasaki et al.]

Haar wavlet-based lossy compression



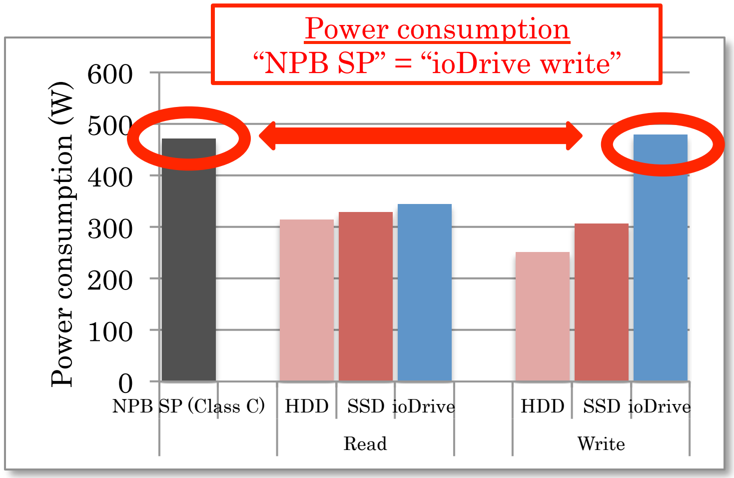


- Evaluation on a real application: NICAM
  - compression rate: 10%-15%
  - Errors: a few percent of errors
- More investigation is needed to deal with the errors

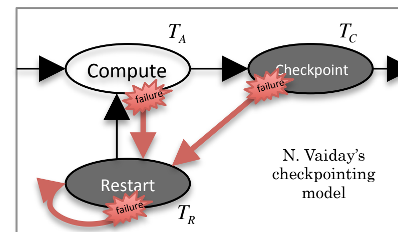
NVM Energy-Away C/R Optimization

[FTXS2013, Takafumi Saito et al.]

Energy optimization for C/R using DVFS



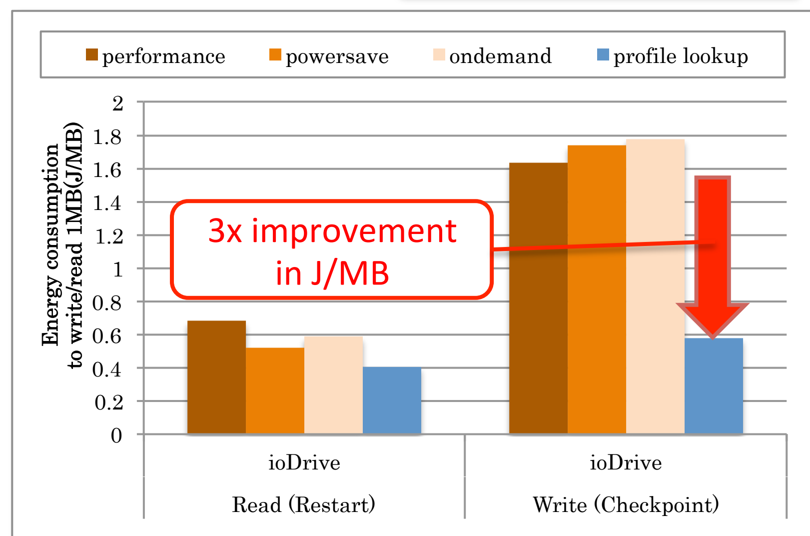
- ioDrive relies on CPU cores for
  - <u>*Grooming*</u>: a garbage collector that pre-erases unused blocks in background to accelerate future write operation
  - <u>*Wear leveling*</u>: a balanced write technique to extend the lifetime of a device
- When decreasing CPU frequency, I/O throughput of ioDrive is degraded like CPU



Energy-Away C/R Model

	Expected {run | I/O} time		Power
Compute	$T_{\bar{A}} = \lambda^{-1} e^{\lambda(T_C+T_R)}\left(e^{\lambda T_A}-1\right)$	⊗	W_A
Checkpoint	$T_{\bar{C}} = \lambda^{-1}\left(e^{\lambda T_C}-1\right)$	⊗	W_C
Restart	$T_{\bar{R}} = \lambda^{-1}\left(e^{\lambda T_C}-1\right)\left(e^{\lambda T_R}-1\right)$	⊗	W_R

$$J = T_{\bar{A}} \cdot W_A + T_{\bar{C}} \cdot W_C + T_{\bar{R}} \cdot W_R$$



Fail-in-Place Network Design

[SC14, Jens Domke et al.]

Failure Analysis of HPC Systems and Fail-in-Place Strategy

LANL Cluster 2 (97–05)

- Unknown configuration

Deimos (07–12)

- 728 nodes
- 108 IB switches
- ≈1,600 links

TSUBAME2.0/2.5 (10–?)

- 1,555 nodes (1,408 compute nodes)
- ≈500 IB switches
- ≈7,000 links

Software more reliable

High MTTR

≈1% annual failure rate

Repair/maintenance is expensive!

TABLE I. COMPARISON OF NETWORK-RELATED HARDWARE AND SOFTWARE FAILURES, MTBF/MTTR, AND ANNUAL FAILURE RATES

Fault Type	Deimos*	LANL Cluster 2	TSUBAME2.5
Percentages of network-related failures			
Software	13%	8%	1%
Hardware	87%	46%	99%
Unspecified		46%	
Percentages for hardware only			
NIC/HCA	59%	78%	1%
Link	27%	7%	93%
Switch	14%	15%	6%
Mean time between failure / mean time to repair			
NIC/HCA	X† / 10 min	10.2 d / 36 min	X / 5–72 h
Link	X / 24–48 h	97.2 d / 57.6 min	X / 5–72 h
Switch	X / 24–48 h	41.8 d / 77.2 min	X / 5–72 h
Annual failure rate			
NIC/HCA	1%	X	≫ 1%
Link	0.2%	X	0.9%‡
Switch	1.5%	X	1%

* Deimos' failure data is not publicly available

† Not enough data for accurate calculation

‡ Excludes first month, i.e., failures sorted out during acceptance testing

Fail-in-Place Strategy

- Replace only *critical* failures, and disable *non-critical* failed components
- Common in storage systems
- Applied when maintenance costs exceed maintenance benefits
- Example: IBM's Flipstone (uses RAID arrays; software disables failed HDD and migrates data)

Can we do fail-in-place in HPC networks?

SC14
Wed
2:00PM-2:30PM

Fail-in-Place Network Design
[SC14, Jens Domke et al.]

Architecture

Simulating Network Failures and Throughput Degradation

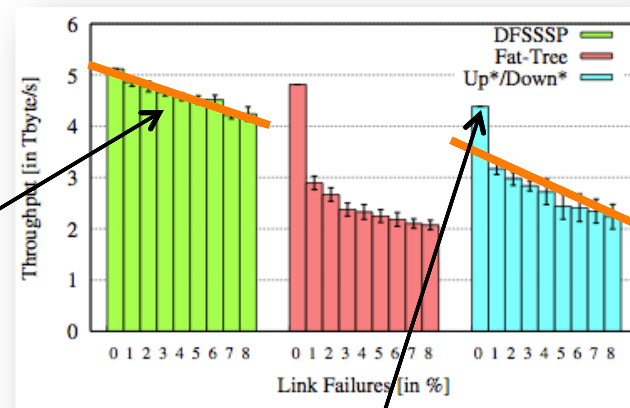
Routing is elementary component to enable fail-in-place networks

Tool-chain for checking fault tolerance of topology and routing algorithm

- Generate faulty topology based on artificial/real network topology
- Apply topology-[aware | agnostic] routing & check connectivity
- Flit-level simulation of InfiniBand hardware with uniform random injection or N-to-N exchange traffic

Simulated throughput degradation as a metric for network/routing reliability

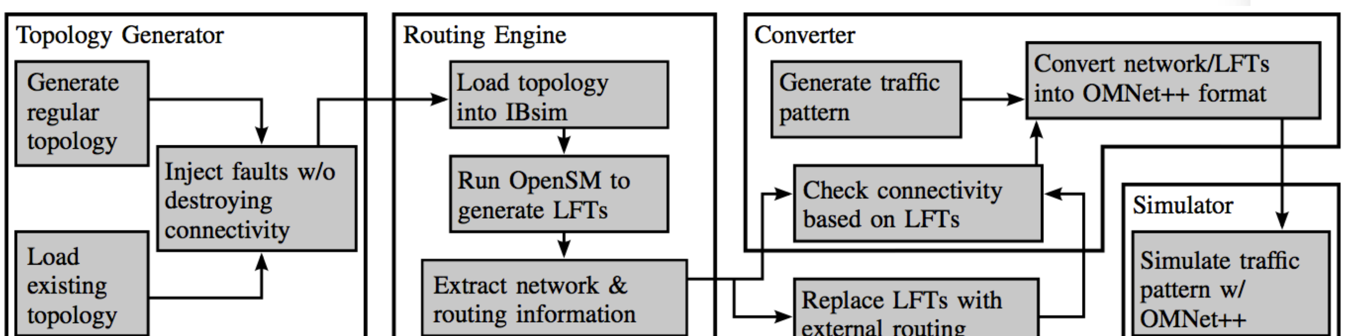
- For each % of switch/link failures do multiple runs (diff. seeds)
- Calculate throughput
- Linear regression



(b) 14-ary 3-tree with 2,156 HCAs

Slope (and R^2)

Intercept



Implications for a real HPC System and Conclusions

All investigated routing algorithms show limitations

- Fat-tree, UpDown, DOR, Torus2QoS
- MinHop, SSSP, DFSSSP, LASH

Topology-aware routings

- High throughput decrease possible with small failure percentage
- Fail to route highly damaged netw.
- Routes not always DL-free (DOR)

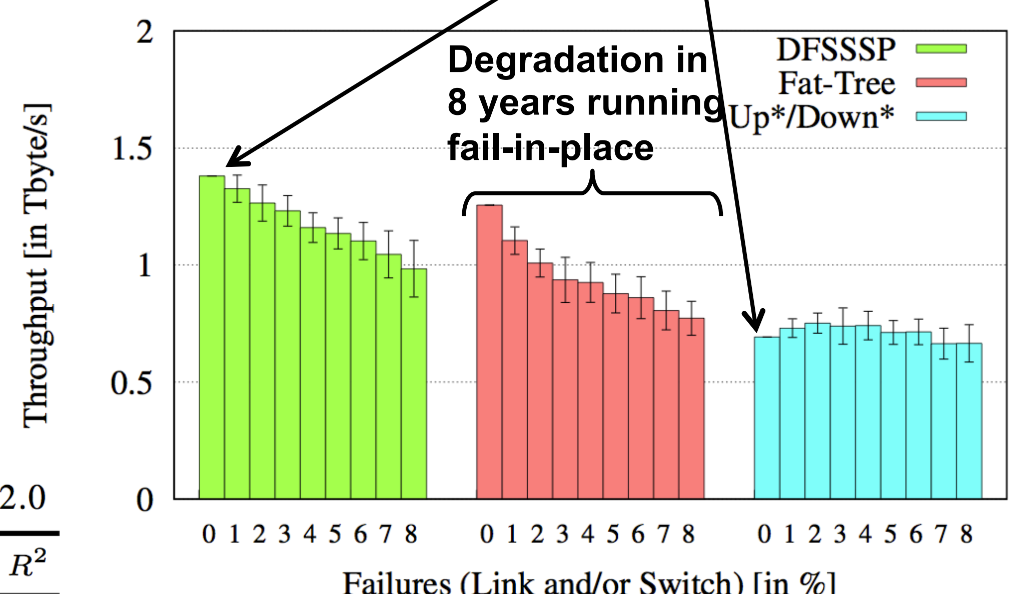
Topology-agnostic routing algorithms

- Ignore deadlocks (MinHop, SSSP)
- Deadlock-avoidance via VLs can be impossible for large scale netw.

TABLE II. INTERCEPT, SLOPE, AND R^2 FOR TSUBAME2.0

HPC system	Routing	Intercept [in Gbyte/s]	Slope	R^2
TSUBAME2.0	**DFSSSP**	1,393.40	-1.33	0.62
	Fat-Tree	1,187.19	-1.48	0.66
	Up/Down**	717.76	-0.08	0.01

Changing from Up*/Down* (default) to DFSSSP routing on TSUBAME2.5 improves the throughput by 2.1x for the fault- free network and increases TSUBAME's fail-in-place characteristics.



Fail-in-Place Network Design is possible! (but we have to improve the routing)

Tokyo Tech.
Billion-Way Resilience Project (2011-2015)

API software

NVCR: GPU C/R library
[HCW2011]

FP Compression
[Submitted to IPDPS2015]

FTI: Fault Tolerance Interface
[SC11, EuroPar12, Cluster12]

IBIO: Infiniband I/O
[CCGrid2014]

Async. C/R
[SC12]

FMI: Fault Tolerant Messaging Interface
[IPDPS2014]

API to resource manager
& scheduler

Model

Async. Model
[SC12]

NVM Energy Model
[FTXS2013]

Storage Model
[CCGrid2014]

NVM Durability model

Failure Prediction

Architecture

Burst buffer architecture
[CCGrid 2014]

Fault-in-Place Network Architecture
[SC14]

Analysis

Failure Monitoring
[IPSJ Tech Report]

Standardization of
failure log

Failure Analysis
w/ Machine Learning

Awards

SC11 Technical Paper Perfect Score Award
(Leonardo Batista Gomez, Seiji Tsuboi, Dimitri Komatitsch, Frank Cappello, Naoya Maruyama & Satoshi Matsuoka)

Inria

CCGrid2014 Best Paper Award
(Kento Sato, Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama & Satoshi Matsuoka)

Lawrence Livermore National Laboratory

API software

- N… R: GPU C/R library [HCW2011]
- FP Compression [Submitted to IPDPS20…
- **FTI**: Fault Tolerance Interface [SC11, EuroPar12, Cluster12]
- **IBIO**: Infiniband I/O [CCGrid2014]
- Asy… R
- **FMI**: Fault Tolerant Messaging Interface [IPDPS2014]
- … resource …ger & sched…

Model

- Async. Model [SC12]
- NVM Energy Model [FTXS2013]
- Storage Model [CCGrid2014]
- NVM Durability model
- Failure Prediction

Architecture

- Burst buffer architecture [CCGrid 2014]
- Fault-in-Place Network Architecture [SC14]

Analysis

- Failure Monitoring [IPSJ Tech Report]
- Standardization of failure log
- Failure Analysis w/ Machine Learning

SC14 Technical Paper

SC14 New Orleans, LA | hpc matters.

Wednesday 2:00PM-2:30PM
Room 388-89-90

Fail-in-Place Network Design: Interaction between Topology, Routing Algorithm and Failures

(Jens Domke, Torsten Hoefler, Satoshi Matsuoka)

ETH Zürich

FTI: Fault Tolerance Interface
[SC11, EuroPar12, Cluster12]

IBIO: Infiniband I/O
14]

Async. C/R
[SC12]

Async. Model
[SC12]

FMI: Fault Tolerant Messaging Interfac

to resource manager
& scheduler

NVM Energy Model
[FTXS2013]

Architecture

Burs architecture
[C Grid 2014]

Storage Model
[CCGrid2014]

Fault-in-Place Network Architecture
[SC14]

NVM Durability model

Failure Prediction

Analysis

Failure Monitoring
[IPSJ Tech Report]

Standardization of
failure log

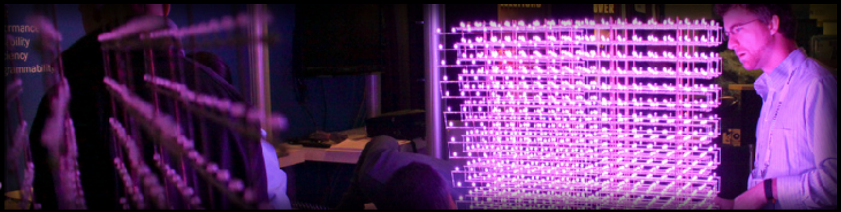
Failure Analysis
w/ Machine Learning

Selected Publications

SC14	J. Domke and T. Hoefler and S. Matsuoka, “Fail-in-Place Network Design: Interaction between Topology, Routing Algorithm and Failures”, IEEE/ACM International Conference on High Performance Computing, Networking, Storage and Analysis (SC14), New Orleans, LA, USA, 2014
CCGrid2014	Kento Sato, Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "A User-level InfiniBand-based File System and Checkpoint Strategy for Burst Buffers", In Proceedings of the 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2014), Chicago, USA, May, 2014.
IPDPS2014	Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "FMI: Fault Tolerant Messaging Interface for Fast and Transparent Recovery", In Proceedings of the International Conference on Parallel and Distributed Processing Symposium 2014 (IPDPS2014), Phoenix, USA, May, 2014.
FTXS2013	Takafumi Saito, Kento Sato, Hitoshi Sato and Satoshi Matsuoka, "Energy-aware I/O Optimization for Checkpoint and Restart on a NAND Flash Memory System", In the Workshop on Fault-Tolerance for HPC at Extreme Scale 2013 (FTXS2013) in conjunction with the International Symposium on High Performance Parallel and Distributed Computing (HPDC13), New York, USA, June, 2013.
IPDPS2013	Improving the computing efficiency of HPC systems using a combination of proactive and preventive checkpointing - Mohamed Slim Bouguerra, Ana Gainaru, Leonardo Bautista-Gomez, Franck Cappello, Naoya Maruyama, Satoshi Matsuoka, IEEE International Parallel & Distributed Processing Symposium 2013 (IPDPS'13), Boston, MA, USA. (Acceptance rate 21.0%)
SNA-MC13	SAMPSON Parallel Computation for Sensitivity Analysis of TEPCO's Fukushima Daiichi Nuclear Power Plant Accident - Marco Pellegrini, Leonardo Bautista-Gomez, Naoya Maruyama, Masanori Naitoh, Satoshi Matsuoka, Franck Cappello, Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2013 (SNA-MC'13), Paris, France.
SC12	Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of a Non-blocking Checkpointing System", In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis 2012 (SC12), Salt Lake, USA, Nov, 2012.
Cluster2012	Leonardo Bautista Gomez, Thomas Ropars, Naoya Maruyama, Franck Cappello, Satoshi Matsuoka. “Hierarchical Clustering Strategies for Fault Tolerance in Large Scale HPC Systems”, In Proc. of IEEE Cluster 2012, IEEE Press, Sep. 2012.
EuroPar2012	L. Bautista Gomez, B. Nicolae, N. Maruyama, F. Cappello, S. Matsuoka. “Scalable Reed-Solomon-based Reliable Local Storage for HPC Applications on IaaS Clouds”, In Proc. of International European Conference on Parallel and Distributed Computing (EuroPar 2012), Aug. 2012.
SC11	Leonardo Bautista, Naoya Maruyama, Dimitri Komatitsch, Tsuboi Seiji, Franck Cappello, Satoshi Matsuoka, Nakamura Takeshi. ”FTI: High performance Fault Tolerance Interface for hybrid systems”. In International Conference for High Performance Computing, Networking, Storage and Analysis (SC).Page 1-12.Nov. 2011.
IPDPSW2011	Nukada, A.; Takizawa, H.; Matsuoka, S., "NVCR: A Transparent Checkpoint-Restart Library for NVIDIA CUDA," Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW), 2011 IEEE International Symposium on , vol., no., pp.104,113, 16-20 May 2011

Question ?

SC14
New Orleans, LA | hpc matters.



Emerging Technologies Booth

Tuesday 5pm-6pm
Wednesday 5pm-6pm

Lawrence Livermore
National Laboratory

Lawrence Livermore
National Laboratory