A User-level InfiniBand-based File System and Checkpoint Strategy for Burst Buffers

Kento Sato[†1], Kathryn Mohror[†2], Adam Moody[†2], Todd Gamblin[†2], Bronis R. de Supinski[†2], Naoya Maruyama[†3] and Satoshi Matsuoka[†1]

†1 Tokyo Institute of Technology
†2 Lawrence Livermore National Laboratory
†3 RIKEN Advanced institute for Computational Science

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52- 07NA27344. LLNL-PRES-654744-DRAFT

Failures on HPC systems

- Exponential growth in computational power
  - Enables finer grained simulations with shorter period time
- Overall failure rate increase accordingly because of the increasing system size
- 191 failures out of 5-million node-hours
  - A production application of Laser-plasma interaction code (pF3D)
  - Hera, Atlas and Coastal clusters @LLNL

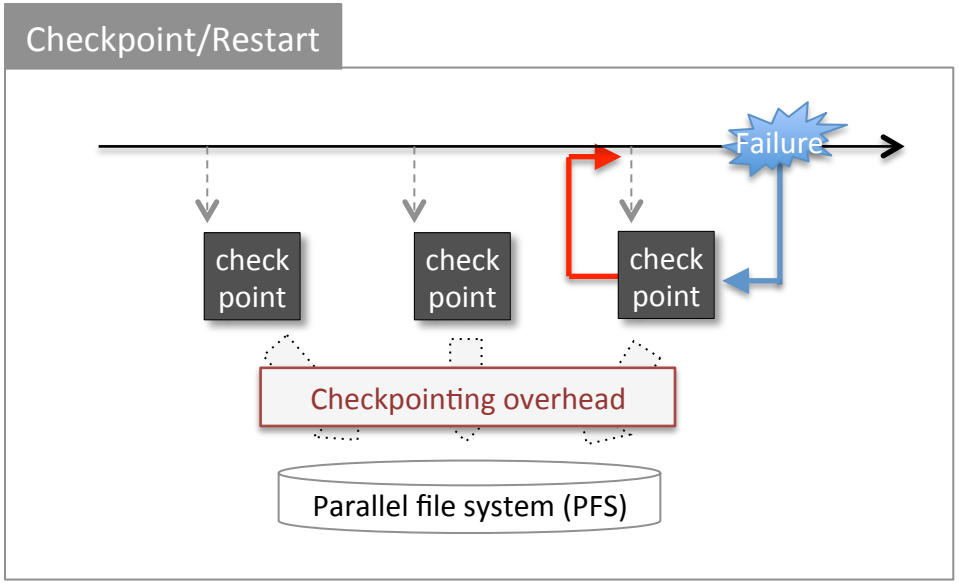
Estimated MTBF (w/o hardware reliability improvement per component in future)

	1,000 nodes	10,000 nodes	100,000 nodes
MTBF	1.2 days (Measured)	2.9 hours (Estimation)	17 minutes (Estimation)

- Will be difficult for applications to continuously run for a long time without fault tolerance at extreme scale

Checkpoint/Restart (Software-Lv.)

- Idea of Checkpoint/Restart
  - Checkpoint
    - Periodically save snapshots of an application state to PFS
  - Restart
    - On a failure, restart the execution from the latest checkpoint

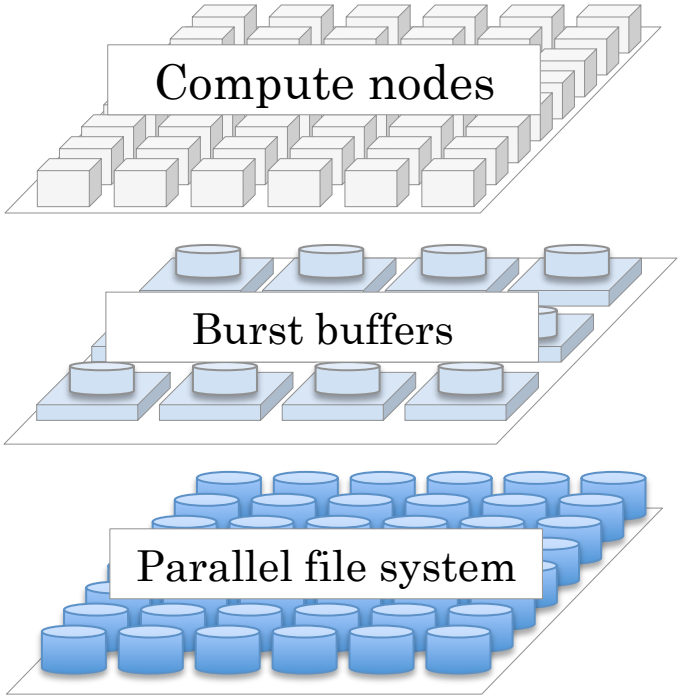


- Improved Checkpoint/Restart
  - Multi-level checkpointing [1]
  - Asynchronous checkpointing [2]
  - In-memory diskless checkpointing [3]
- We found that software-level approaches may be limited in increasing resiliency at extreme scale

[1] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System (SC 10)
[2] 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
[3] 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", IPDPS2014

Storage architectures

- We consider architecture-level approaches
- Burst buffer
  - A new tier in storage hierarchies
  - Absorb bursty I/O requests from applications
  - Fill performance gap between node-local storage and PFSs in both latency and bandwidth
- If you write checkpoints to burst buffers,
  - Faster checkpoint/restart time than PFS
  - More reliable than storing on compute nodes



- However,…
  - Adding burst buffer nodes may increase total system size, and failure rates accordingly
    - It's not clear if burst buffers improve overall system efficiency
  - Because burst buffers also connect to networks, the burst buffers may still be a bottleneck

[4] Doraimani, Shyamala and Iamnitchi, Adriana, "File Grouping for Scientific Data Management: Lessons from Experimenting with Real Traces", HPDC '08

Goal and Contributions

- **Goal**:
  - Develop an interface to exploit bandwidth of burst buffers
  - Explore effectiveness of burst buffers
  - Find out the best C/R strategy on burst buffers

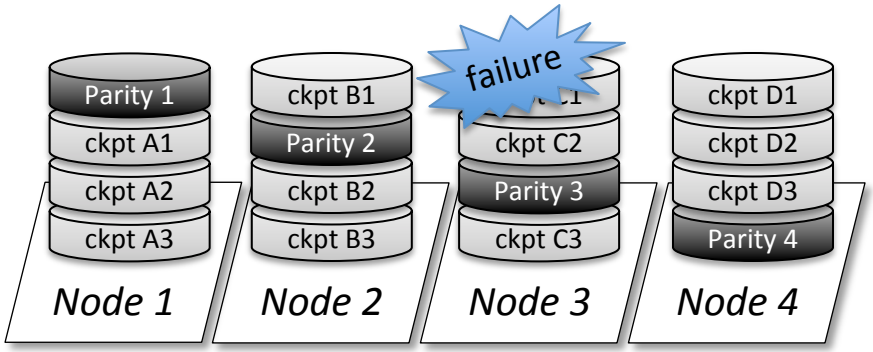
- **Contributions**:
  - Development of IBIO exploiting bandwidth to burst buffers
  - A model to evaluate system resiliency given a C/R strategy and storage configuration
  - Our experimental results show a direction to build resilient systems for extreme scale computing

Outlines

- Introduction
- Checkpoint strategies
- Storage designs
- IBIO: InfiniBand-based I/O interface
- Modeling
- Experiments
- Conclusion

Diskless checkpoint/restart (C/R)

- Diskless C/R
  - Create redundant data across local storages on compute nodes using a encoding technique such as XOR
  - Can restore lost checkpoints on a failure caused by small # of nodes like RAID-5

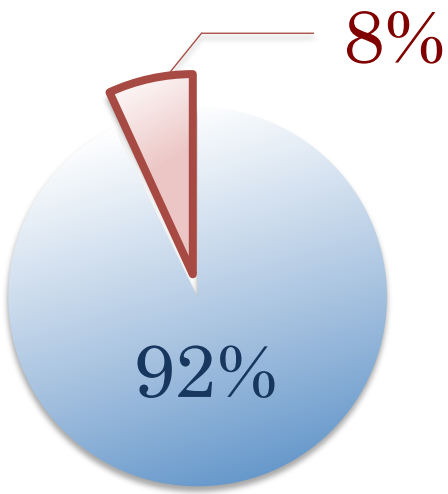


XOR encoding example

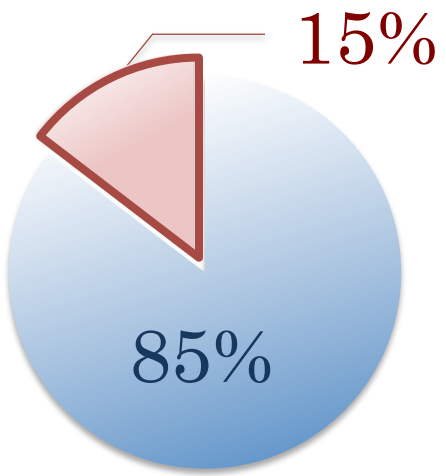
- Most of failures comes from one node, or can recover from XOR checkpoint
  - e.g. 1) TSUBAME2.0: 92% failures
  - e.g. 2) LLNL clusters: 85% failures

Rest of failures still require a checkpoint on a reliable PFS

Diskless checkpoint is promising approach

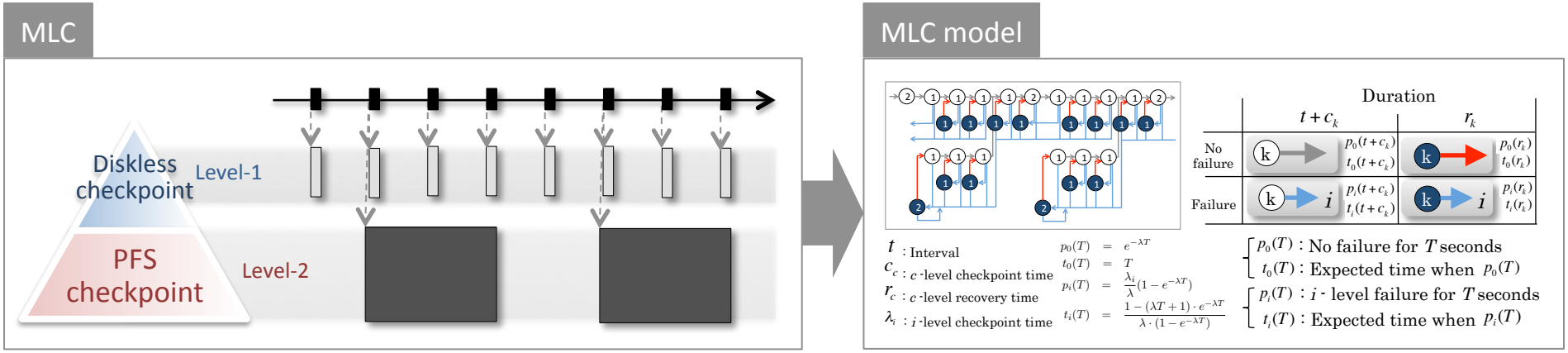


Failure analysis on TSUBAME2.0

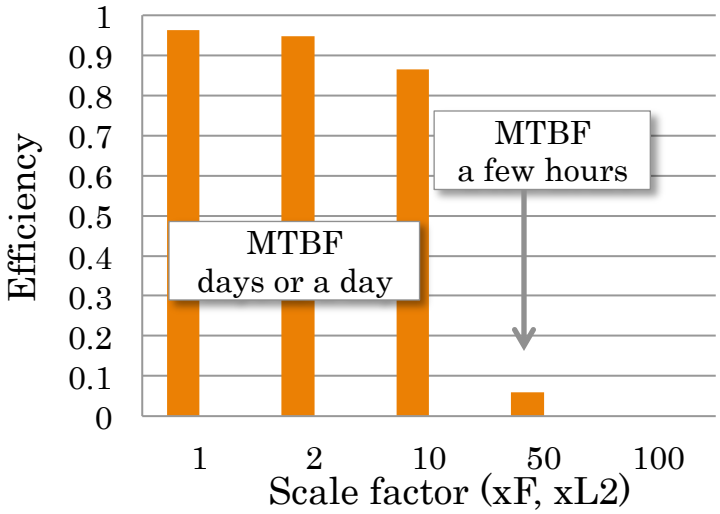


Failure analysis on LLNL clusters

Multi-level Checkpoint/Restart (MLC/R) [1,2]

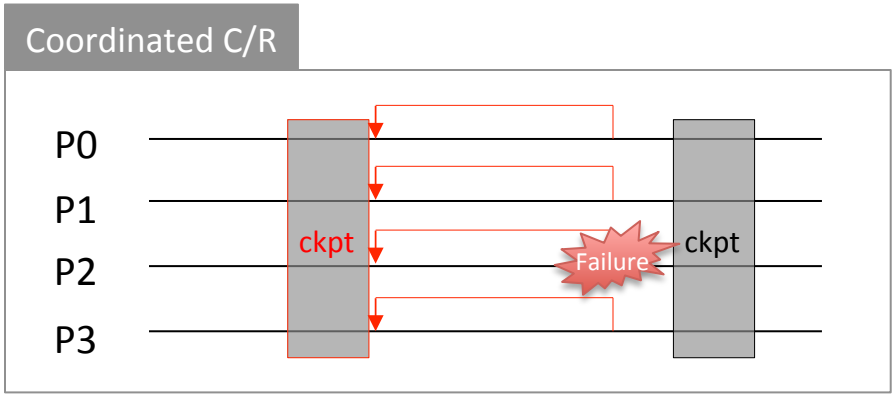


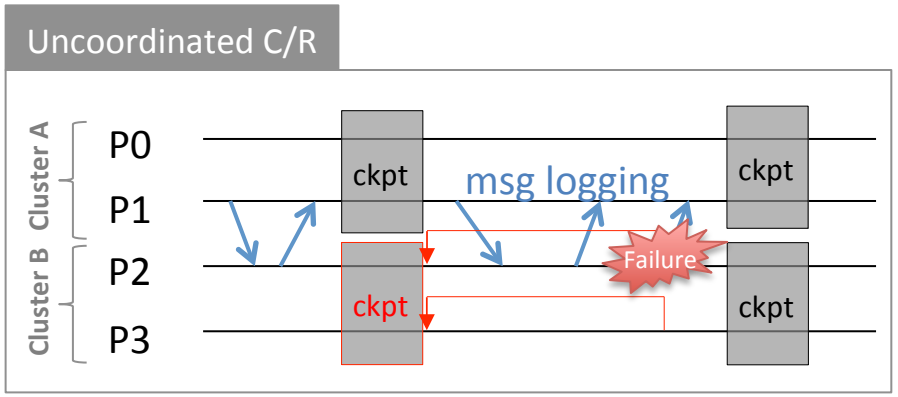
- MLC hierarchically use storage levels
  - Diskless checkpoint: Frequent for one node for a few node failure
  - PFS checkpoint: Less frequent and asynchronous for multi-node failure
- Our evaluation showed system efficiency drops to less than 10% when MTBF is a few hours



[1] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System (SC 10)
[2] 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

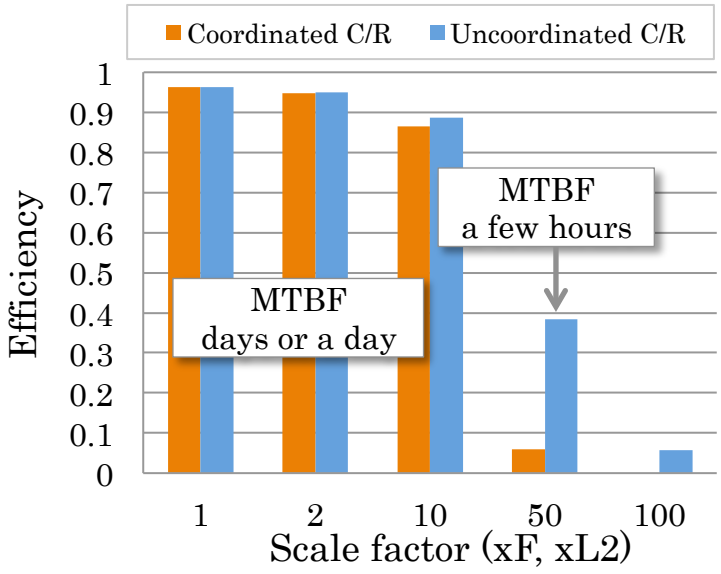
Uncoordinated C/R + MLC





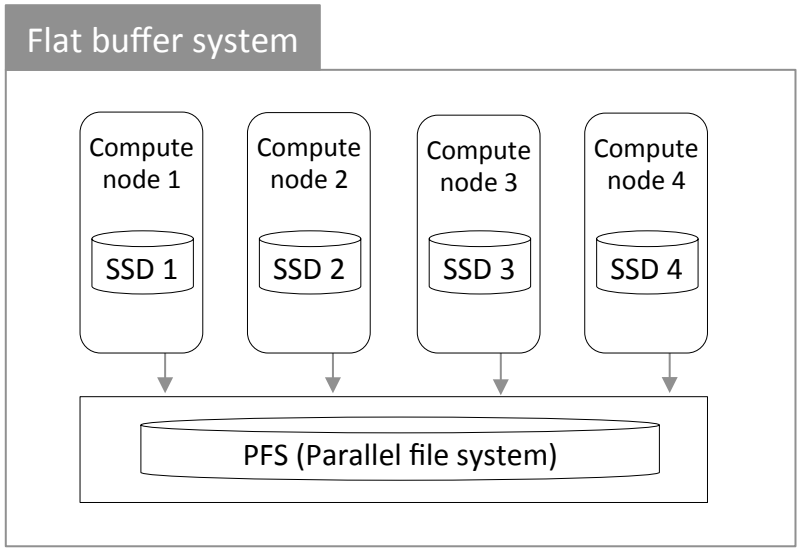
- Coordinated C/R
  - All processes globally synchronize before taking checkpoints and restart on a failure
  - Restart overhead
- Uncoordinated C/R
  - Create clusters, and log messages exchanged between clusters
  - Message logging overhead is incurred, but rolling-back only a cluster can restart the execution on a failure

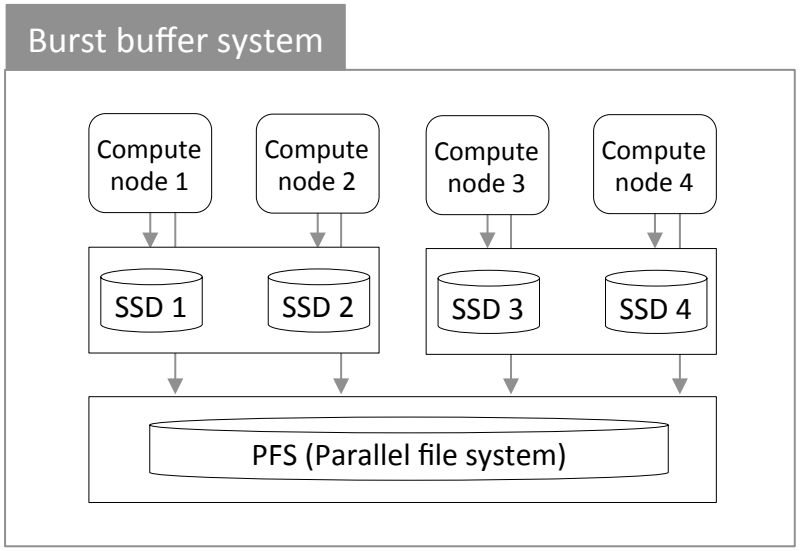
⇒ MLC + Uncoordinated C/R (Software-level) approaches may be limited at extreme scale



Storage designs

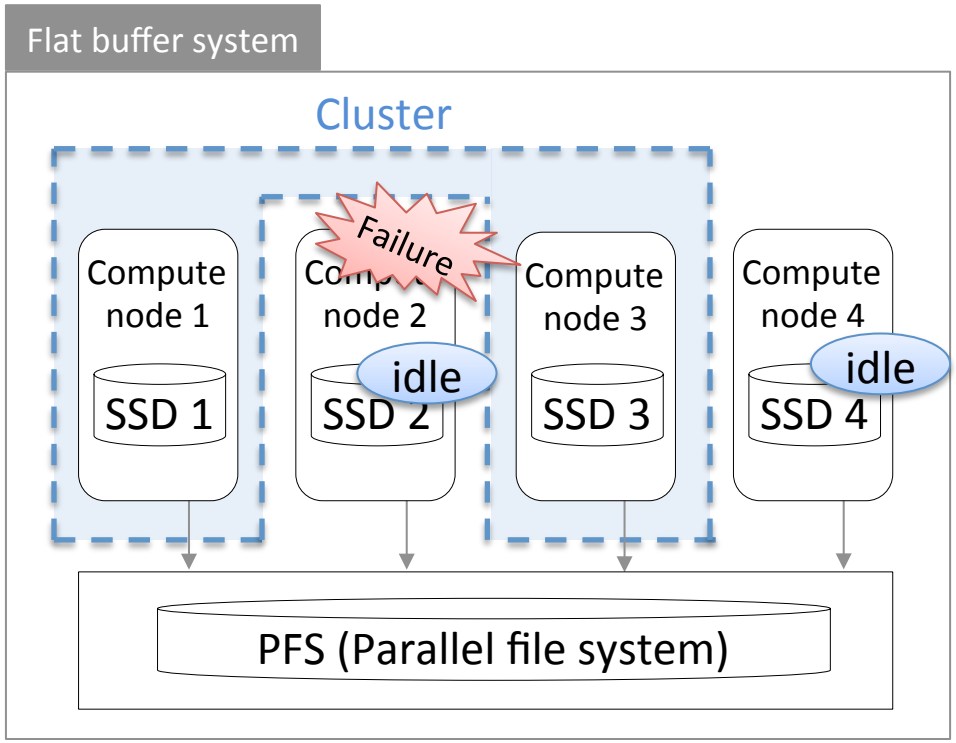
- Addition to the software-level approaches, we also explore two architecture-level approaches
  - Flat buffer system:
    - Current storage system
  - Burst buffer system:
    - Separated buffer space





Flat Buffer Systems

- Design concept
  - Each compute node has its dedicated node-local storage
  - Scalable with increasing number of compute nodes



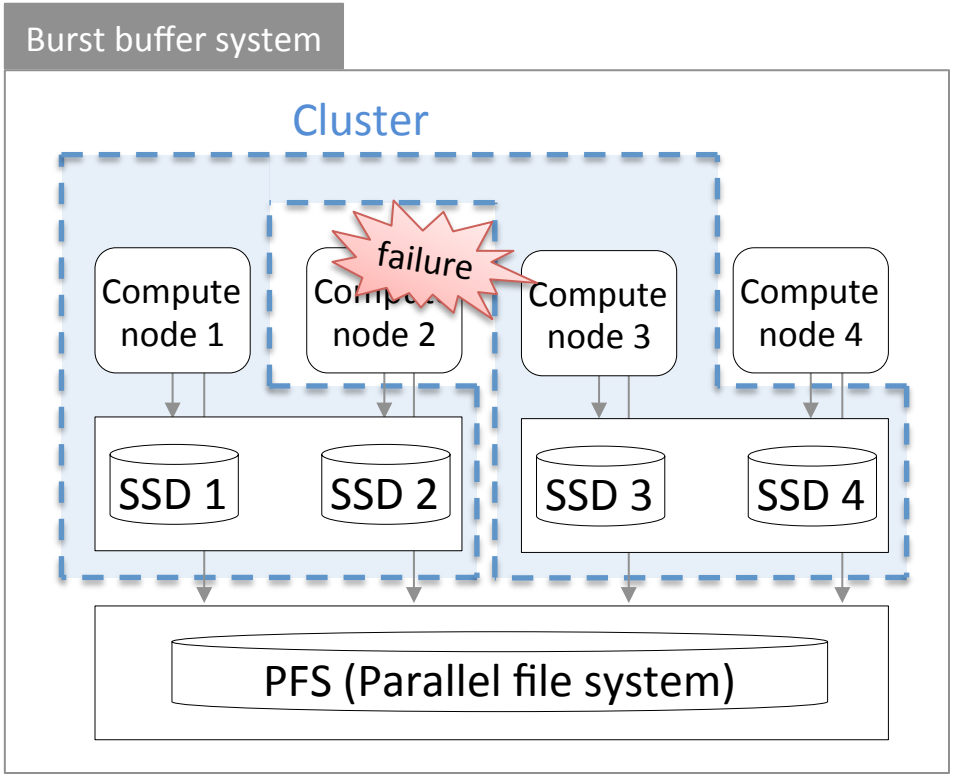
- This design has drawbacks:
  1. Unreliable checkpoint storage

     e.g.) If compute node 2 fails, a checkpoint on SSD 2 will be lost because SSD 2 is physically attached to the failed compute node 2
  2. Inefficient utilization of storage resources on uncoordinated checkpointing

     e.g.) If compute node 1 & 3 are in a same cluster, and restart from a failure, the bandwidth of SSD 2 & 4 will not be utilized

Burst Buffer Systems

- Design concept
  - A burst buffer is a storage space to bridge the gap in latency and bandwidth between node-local storage and the PFS
  - Shared by a subset of compute nodes

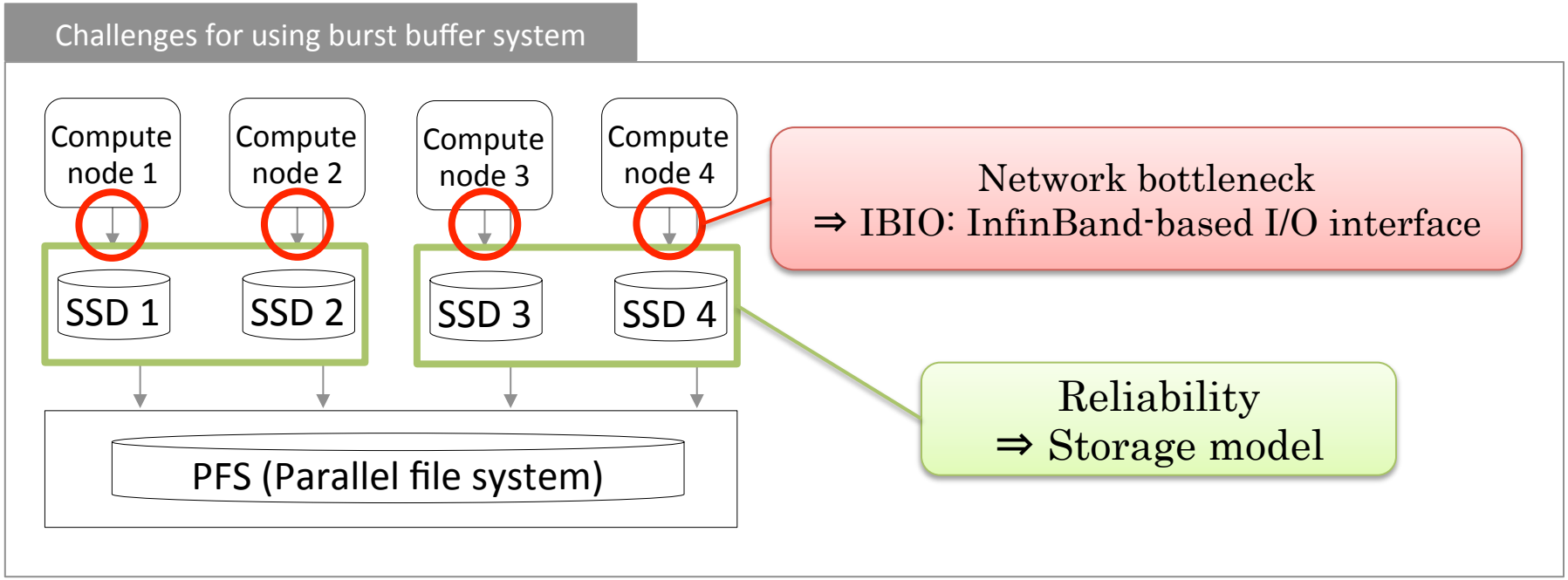


- Although additional nodes are required, several advantages
  1. More Reliable because burst buffers are located on a smaller # of nodes

     e.g.) Even if compute node 2 fails, a checkpoint of compute node 2 is accessible from the other compute node 1
  2. Efficient utilization of storage resources on uncoordinated checkpointing

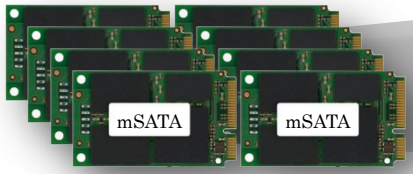
     e.g.) if compute node 1 and 3 are in a same cluster, and both restart from a failure, the processes can utilize all SSD bandwidth unlike a flat buffer system

Challenges for using burst buffers

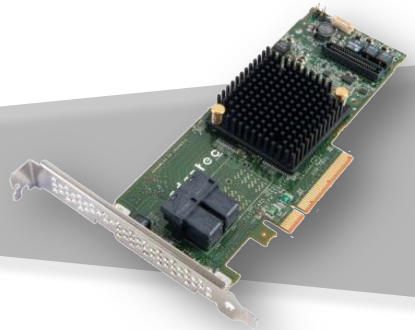


- Exploiting storage bandwidth of burst buffers
  - Burst buffers are connected to networks, networks can be bottleneck
- Analyzing reliability of systems with burst buffers
  - Adding burst buffer nodes increase total system size
  - System efficiency may decrease due to Increased overall failure by added burst buffers

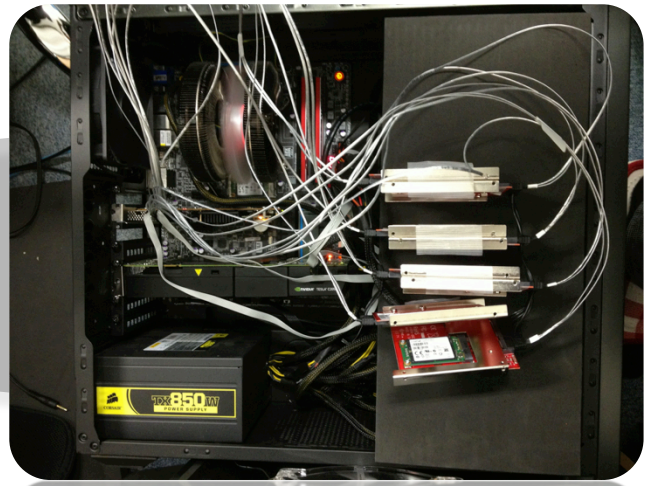
Burst buffer prototype
multi-mSATA High I/O BW & cost

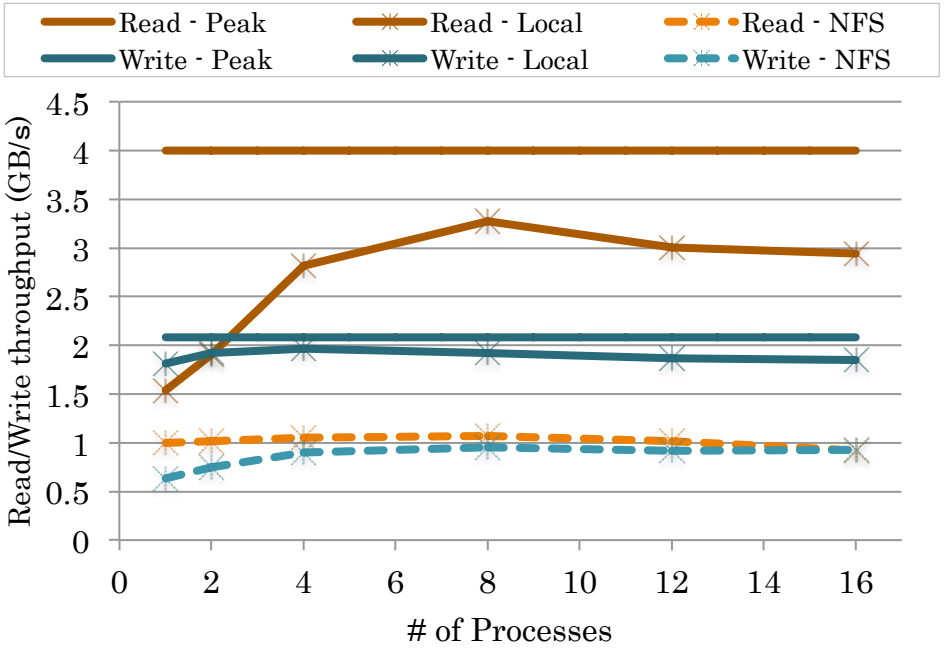


mSATA × 8
(Read: 500MB/s, Write: 260MB/s)



Adaptec RAID × 1





Node specification

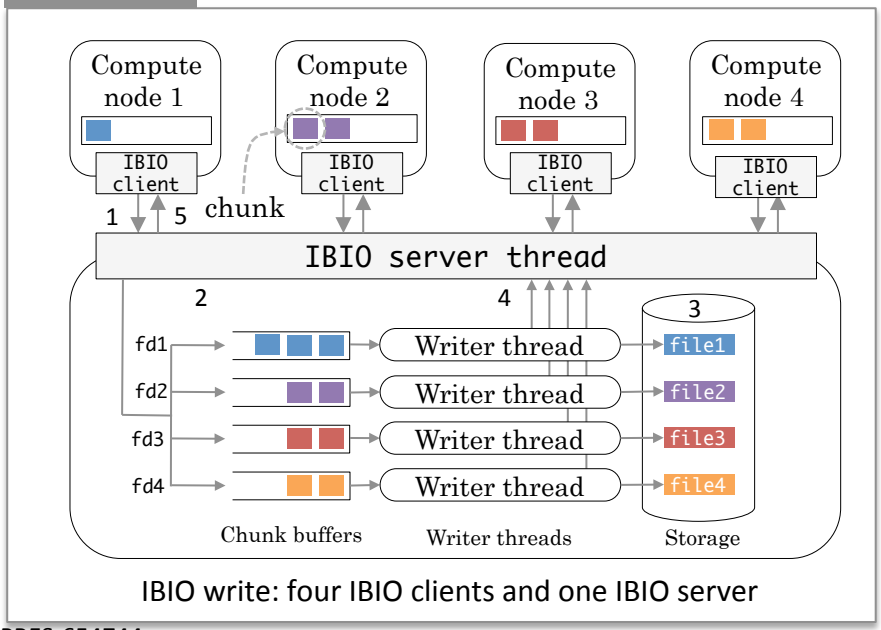
CPU	Intel Core i7-3770K CPU (3.50GHz x 4 cores)
Memory	Cetus DDR3-1600 (16GB)
M/B	GIGABYTE GA-Z77X-UD5H
SSD	Crucial m4 msata 256GB CT256M4SSD3 (Peak read: 500MB/s, Peak write: 260MB/s)
SATA converter	KOUTECH IO-ASS110 mSATA to 2.5' SATA Device Converter with Metal Fram
RAID Card	Adaptec RAID 7805Q ASR-7805Q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

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

IBIO: InfiniBand-based I/O interface

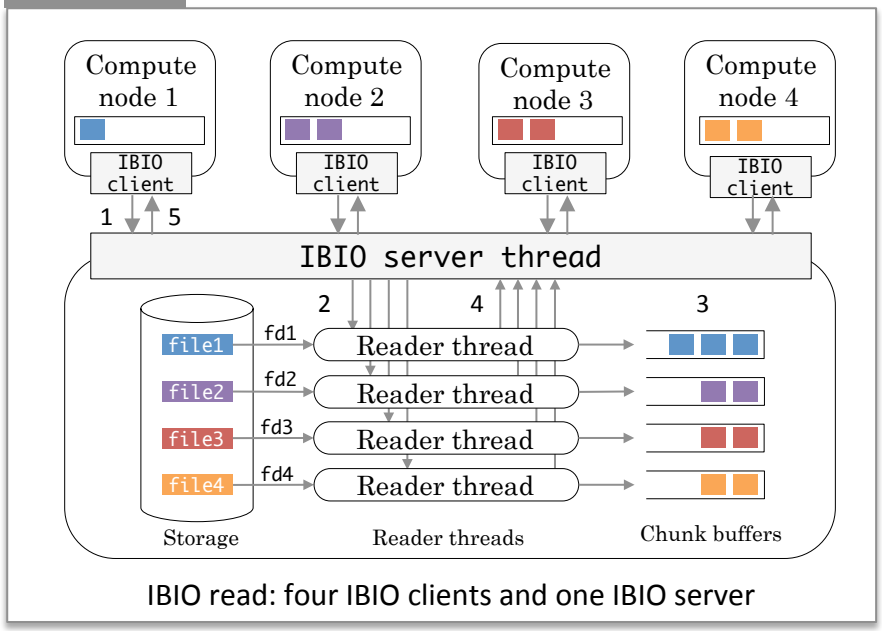
- Provide POSIX I/O interfaces
  - open, read, write and close
  - Client can open any files on any servers
    - open("hostname:/path/to/file", mode)
- IBIO use ibverbs for communication between clients and servers
  - Exploit network bandwidth of infiniBand

IBIO write



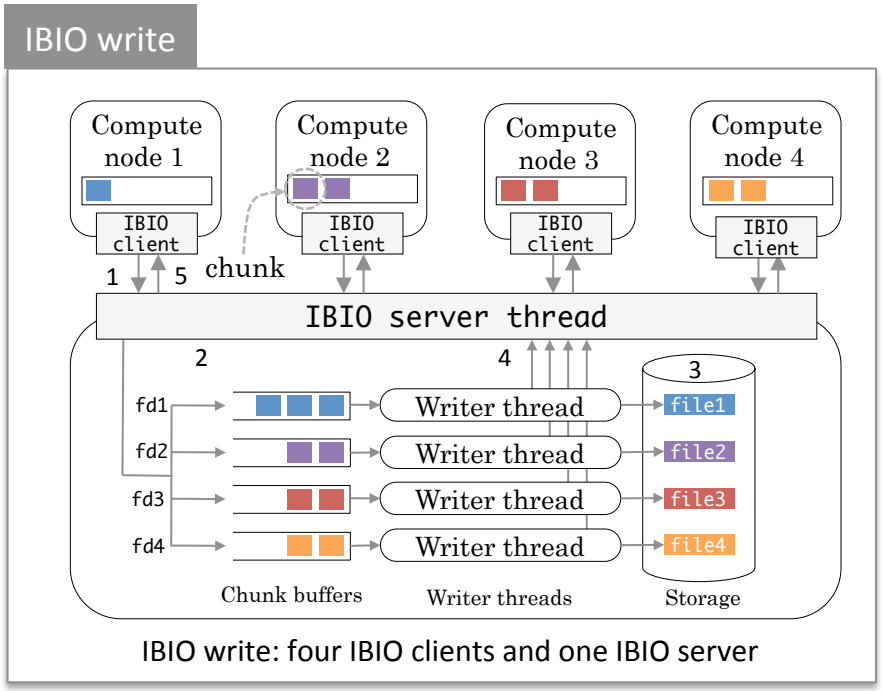
IBIO write: four IBIO clients and one IBIO server

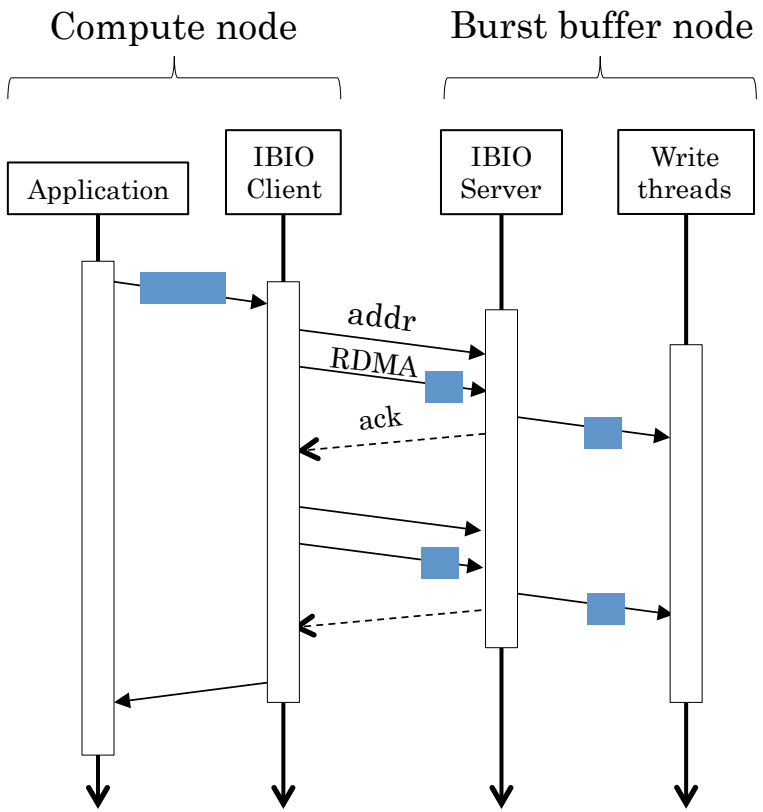
IBIO read



IBIO read: four IBIO clients and one IBIO server

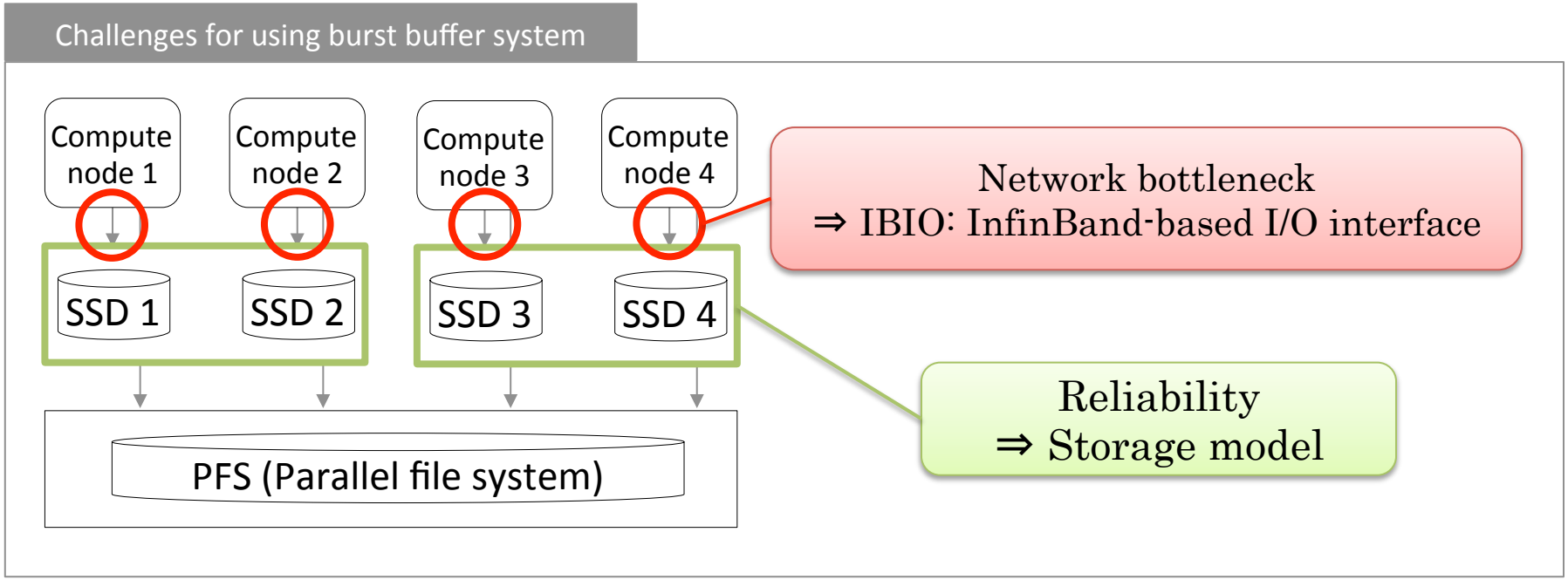
IBIO write/read





- IBIO write
  1. Application call IBIO client function with data to write
  2. IBIO client divides the data into chunks, then send the address to IBIO server for RDMA
  3. IBIO server issues RDMA read to the address, and reply ack
  4. Continues until all chunks are sent, and return to application
  5. Writer threads asynchronously write received data to storage
- IBIO read
  - Reads chunks by reader threads and send to clients in the same way as IBIO write by using RDMA

Challenges for using burst buffers



- Exploiting storage bandwidth of burst buffers
  - Burst buffers are connected to networks, networks can be bottleneck
- Analyzing reliability of systems with burst buffers
  - Adding burst buffer nodes increase total system size
  - System efficiency may decrease due to Increased overall failure by added burst buffers

Modeling overview

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

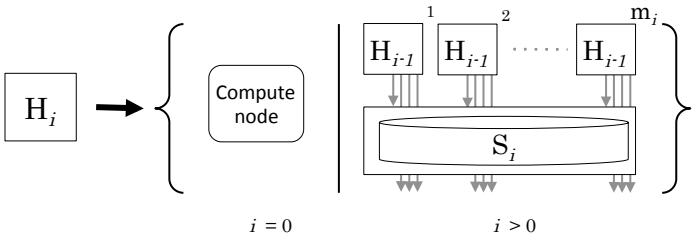
C/R strategy model

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

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

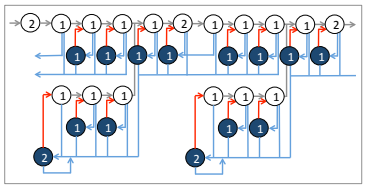
+

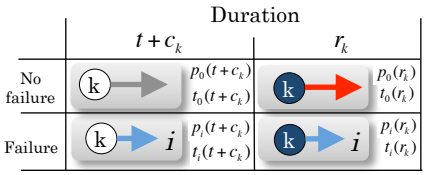
Recursive structured storage model



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

MLC model [2]





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 $p_i(T)$

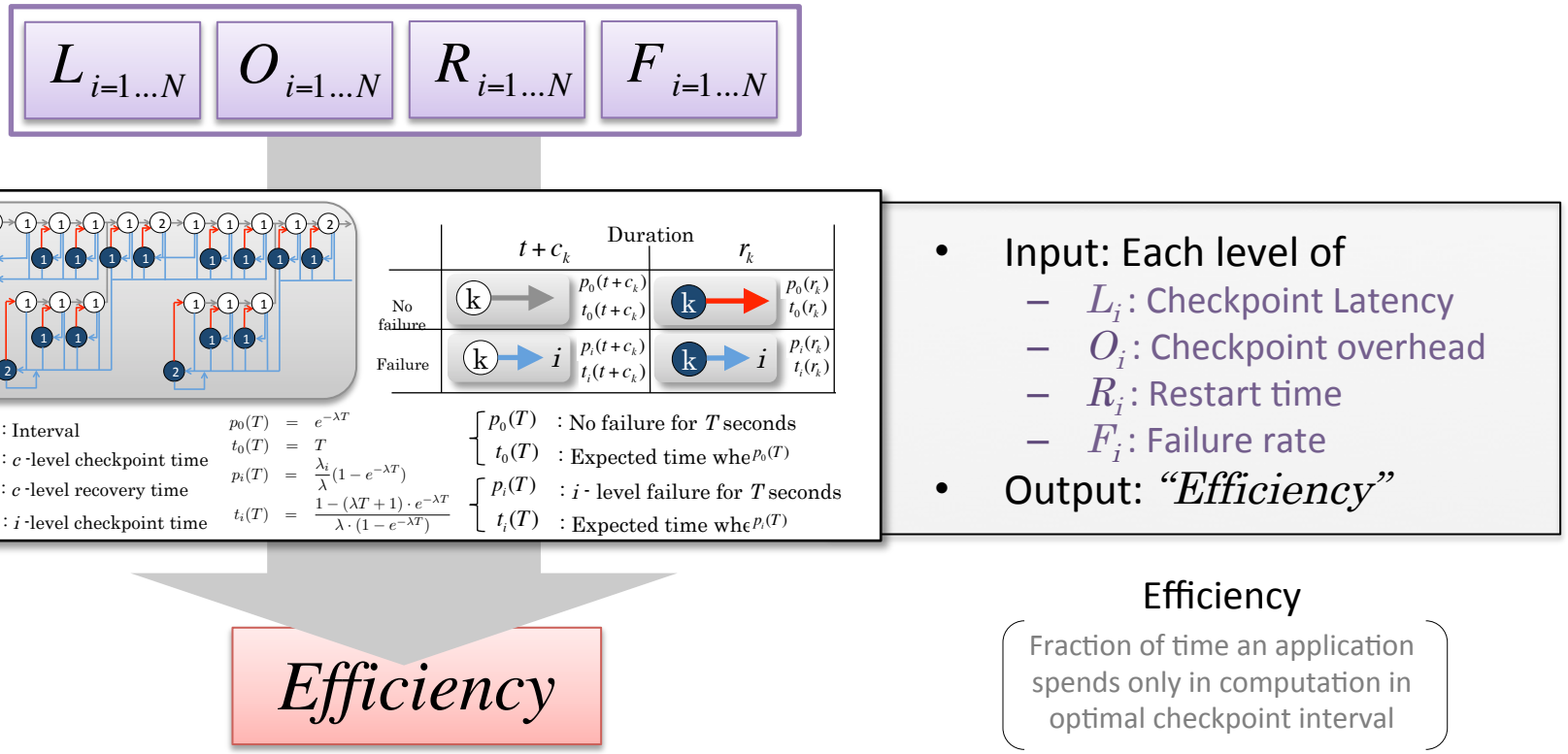
Efficiency

Fraction of time an application spends only in useful computation

[2] 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

Multi-level Asynchronous C/R Model [2]

- Optimize checkpoint intervals and compute checkpoint/restart "*Efficiency*" using Markov model
  - Vertex: Compute state OR Checkpointing state OR Recovery state
  - Edge: Completion of each state



- Input: Each level of
  - L_i : Checkpoint Latency
  - O_i : Checkpoint overhead
  - R_i : Restart time
  - F_i : Failure rate
- Output: "*Efficiency*"

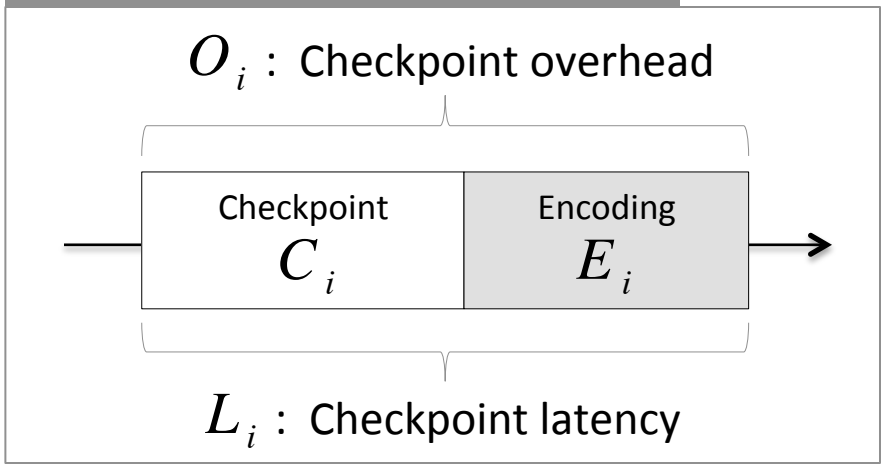
Efficiency

Fraction of time an application spends only in computation in optimal checkpoint interval

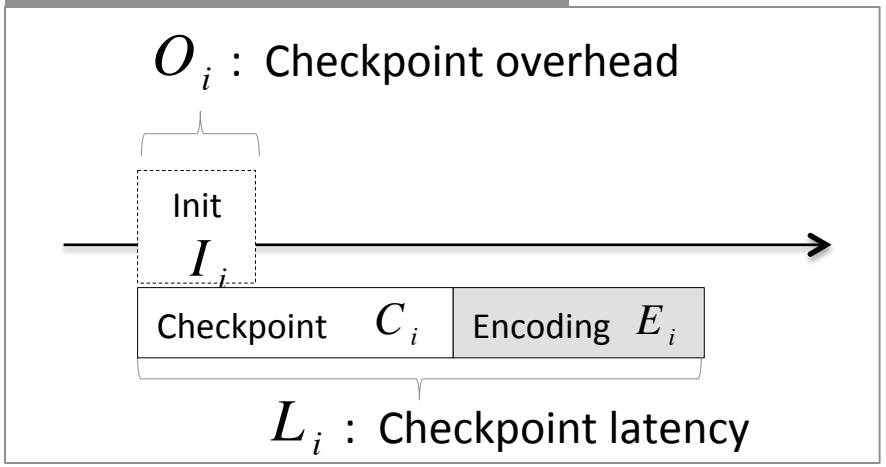
[2] Kento Sato, Adam Moody, Kathryn Mohror, Todd Gamblin, Bronis R. de Supinski, Naoya Maruyama and Satoshi Matsuoka, "Design and Modeling of a Non-blocking Checkpointing System", SC12

Modeling of C/R Strategies

Synchronous checkpointing (Diskless C/R)



Asynchronous checkpointing (PFS)



- L_i : Checkpoint Latency
  - Time to complete a checkpoint (C_i) and encoding (E_i)

$$L_i = C_i + E_i$$

- C_i & R_i : Checkpoint/Restart time

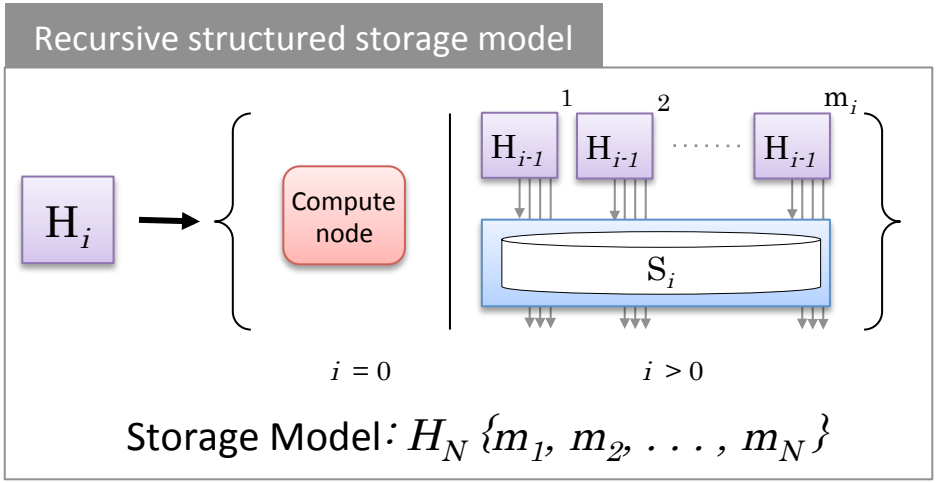
- O_i : Checkpoint overhead
  - The increased execution time of an application

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

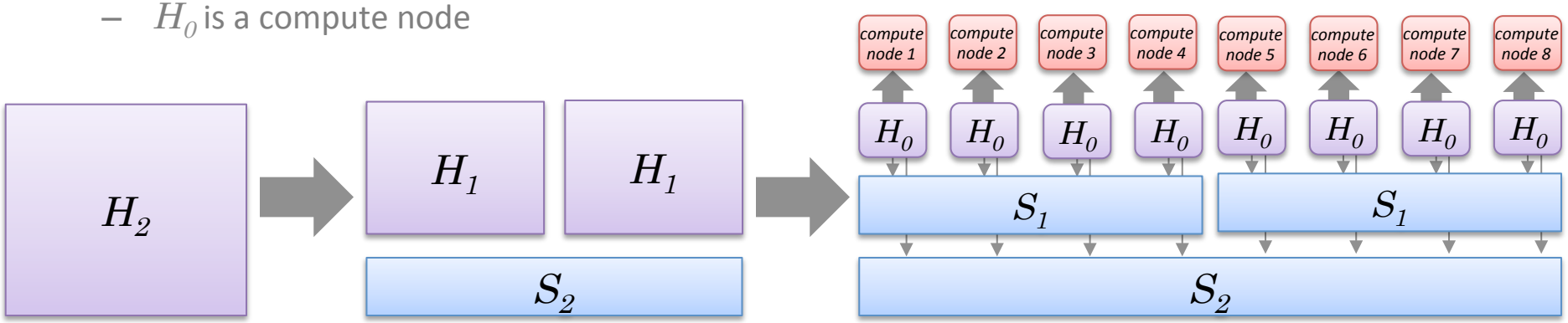
$$C_i \text{ or } R_i = \frac{\text{< C/R data size / node >} \times \text{<\# of C/R nodes per } S_i^* \text{>}}{\text{< write perf. (} w_i \text{) > or <read perf. (} r_i \text{) >}}$$

Recursive Structured Storage Model

- Generalization of storage architectures with "*context-free grammar*"
  - A tier i hierarchical entity (H_i), has a storage (S_i)shared by (m_i) upper hierarchical entities (H_{i-1})
  - $H_{i=0}$ is a compute node
  - $H_N\{m_1, m_2, \ldots, m_N\}$



- e.g.) $H_2\{4, 2\}$
  - H_2 has an S_2 shared by 2 H_1
  - H_1 has an S_1 shared by 4 H_0
  - H_0 is a compute node



Recursive Structured Storage Model (cont'd)

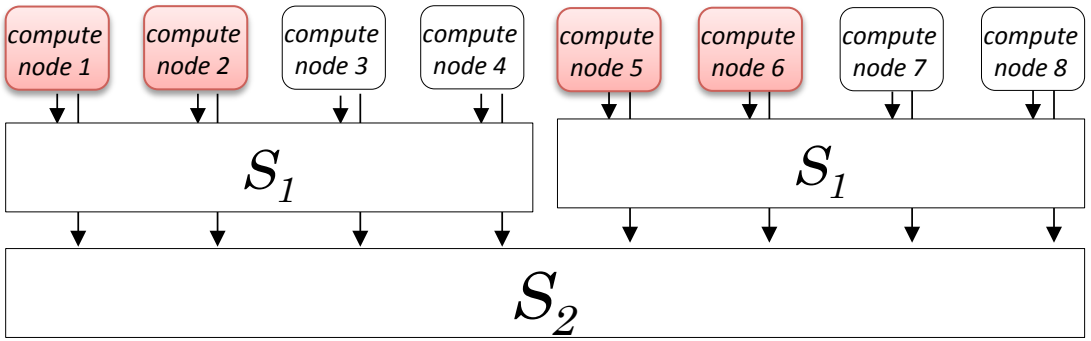
- The number of nodes accessing to S_i

$$\text{<\# of C/R nodes per } S_i\text{>} = \frac{K}{\text{<\# of } S_i\text{>}}$$

$$\text{<\# of } S_i\text{>} = \begin{cases} \Pi^{N}_{k=i+1} m_k & (i < N) \\ 1 & (i = N) \end{cases}$$

K : C/R cluster size

- e.g.) $K = 4$
  - # of C/R nodes per S_1
    - 4/2 = 2 nodes
  - # of C/R nodes per S_2
    - *4/1 = 4* nodes

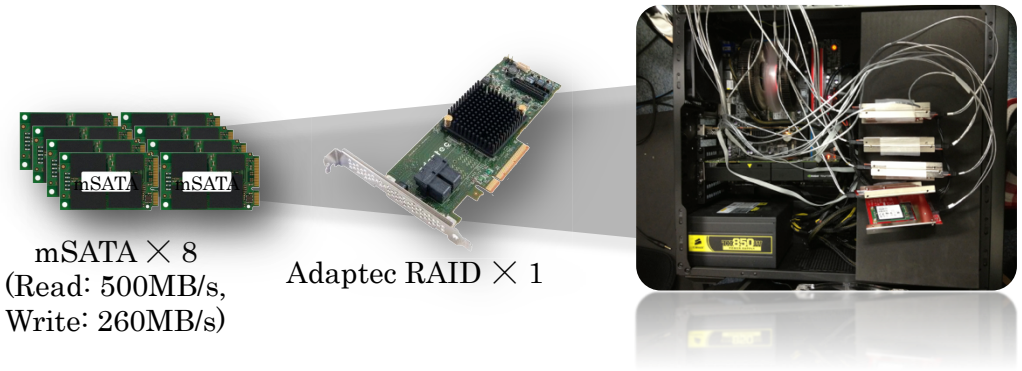


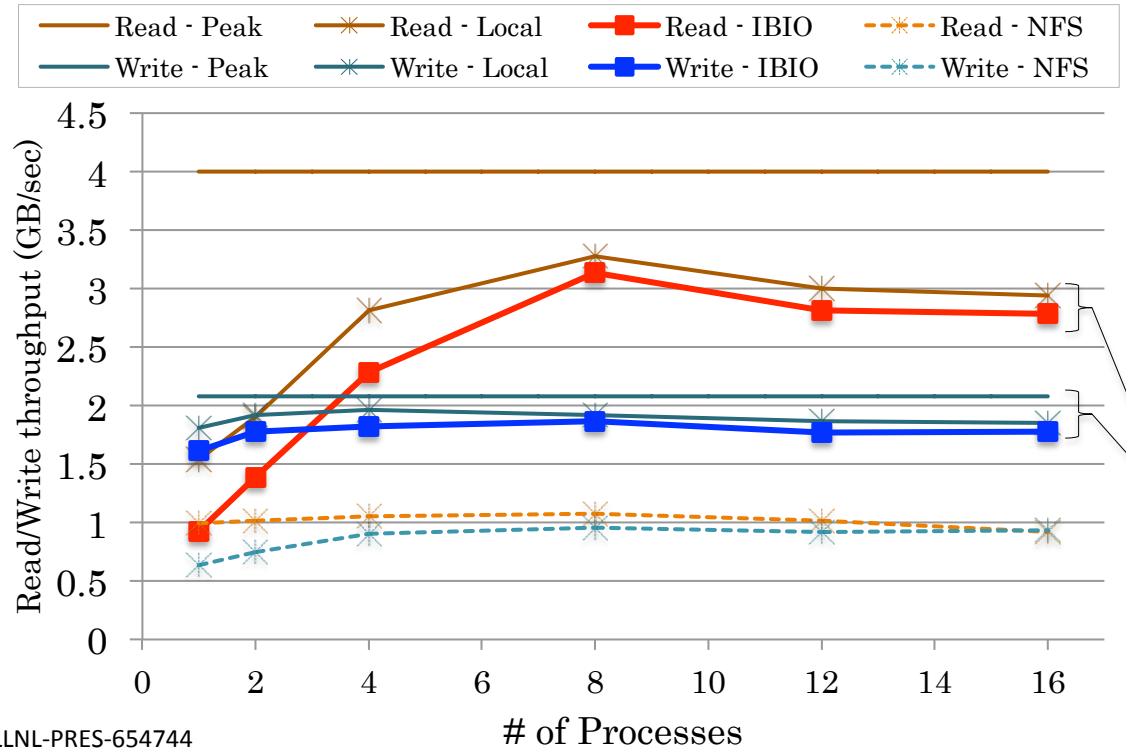
Evaluations

- IBIO performance
  - Sequential read/write for C/R
- Several system efficiency evaluations

Sequential IBIO read/write performance

- Set chunk size to 64MB for both IBIO and NFS to maximize the throughputs





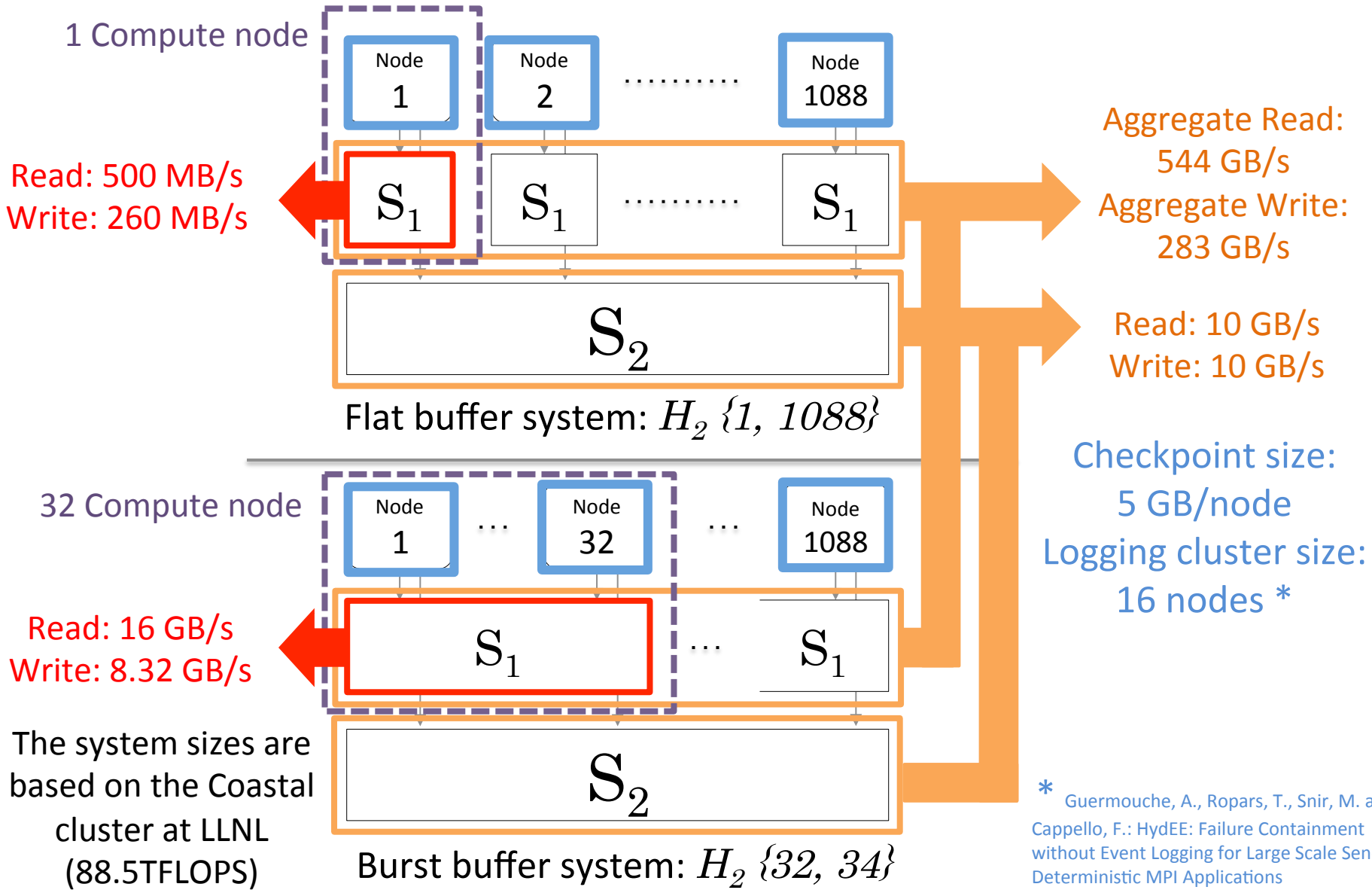
Node specification

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)

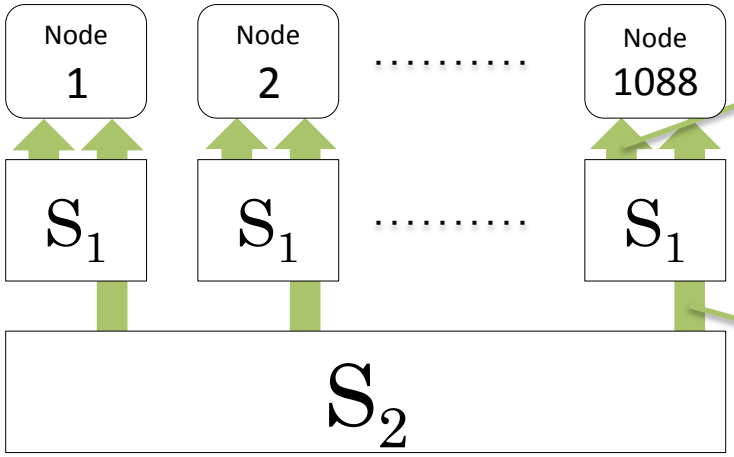
IBIO achieve the same remote read/write performance as the local read/write performance by using RDMA

Experimental setup



* Guermouche, A., Ropars, T., Snir, M. and Cappello, F.: HydEE: Failure Containment without Event Logging for Large Scale Send-Deterministic MPI Applications

Experimental setup



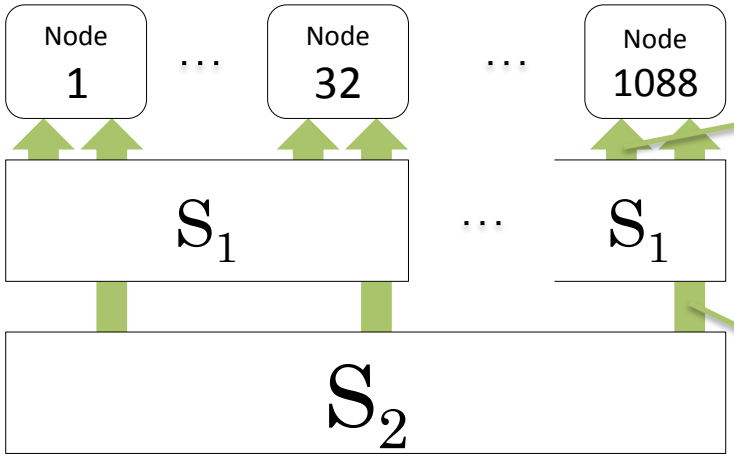
Flat buffer system: H_2 {1, 1088}

Level 1
(XOR checkpoint required)
2.14×10^{-6}

Level 2
(PFS checkpoint required)
4.28×10^{-7}

Estimated failure rates are based on failure analysis on the Coastal cluster at LLNL (88.5TFLOPS) [1]

[1] A. Moody, G. Bronevetsky, K. Mohror, and B. R. de Supinski, "Design, Modeling, and Evaluation of a Scalable Multi-level Checkpointing System (SC 10)



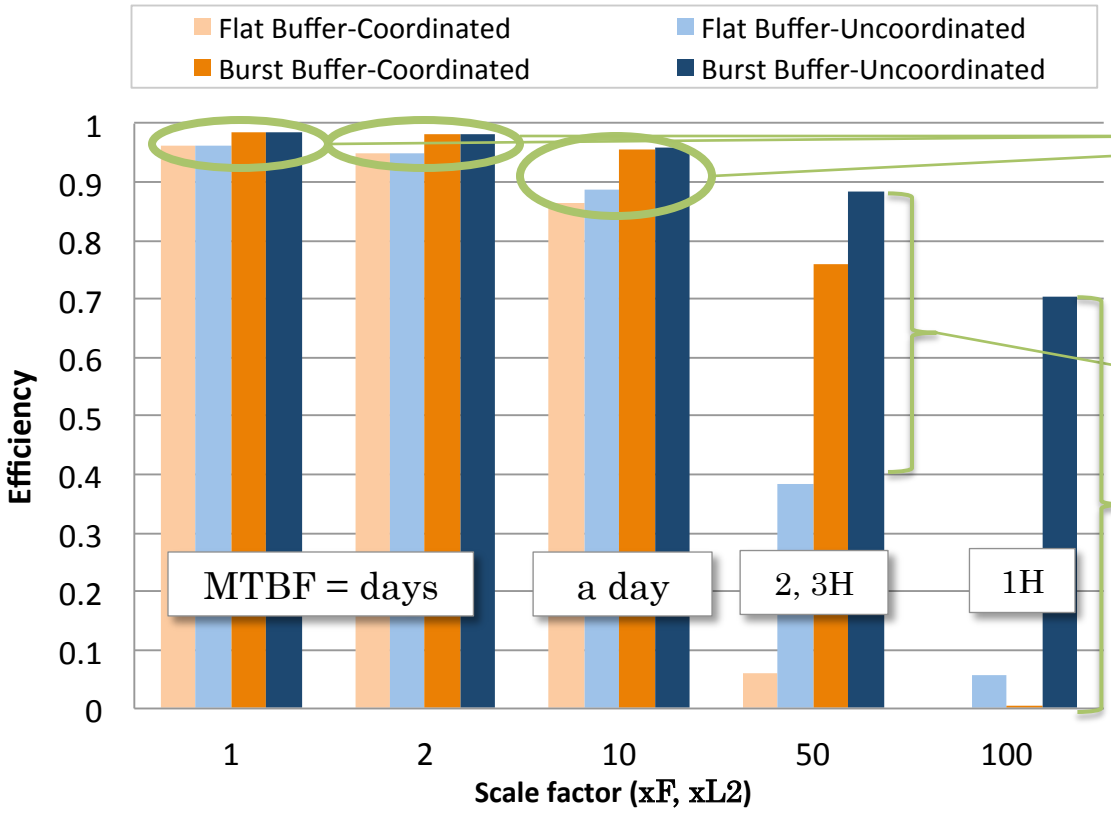
Burst buffer system: H_2 {32, 34}

Level 1
(XOR checkpoint required)
2.63×10^{-6}

Level 2
(PFS checkpoint required)
1.33×10^{-8}

Efficiency with Increasing Failure Rates and Checkpoint Costs

- Assuming there is no message logging overhead



In days or a day of MTBF, there is no big efficiency differences

In a few hours of MTBF, with burst buffers, systems can still achieve high efficiency

Even in a hour of MTBF, with uncoordinated, systems can still achieve 70% efficiency

⇒ Partial restart accelerate recovery time from burst buffers and PFS checkpoint

Allowable Message Logging overhead

Message logging overhead allowed in uncoordinated checkpointing to achieve a higher efficiency than coordinated checkpointing

Flat buffer		Burst buffer	
scale factor	Allowable message logging overhead	scale factor	Allowable message logging overhead
1	0.0232%	1	0.00435%
2	0.0929%	2	0.0175%
10	2.45%	10	0.468%
50	84.5%	50	42.0%
100	≈ 100%	100	99.9%

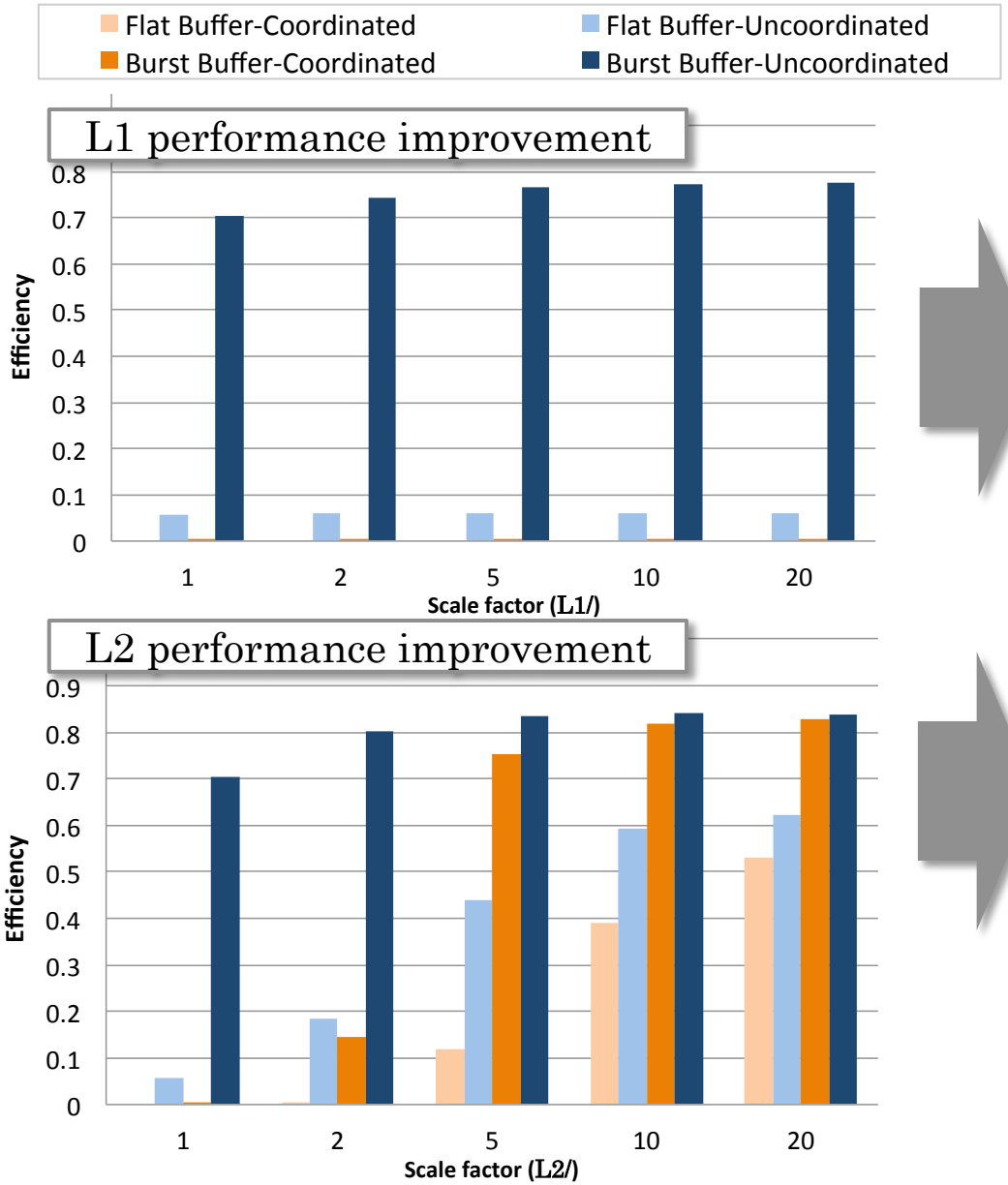
Coordinated

Uncoordinated

- Logging overhead must be relatively small, less than a few percent in days or a day of MTBF
  - In a few hours or a hour, very high message logging overheads are tolerated

⇒ Uncoordinated checkpointing can be more effective on future systems

Effect of Improving Storage Performance



To see which storage impact to efficiency, we increase performance of level-1 and level-2 storage while keeping MTBF a hour

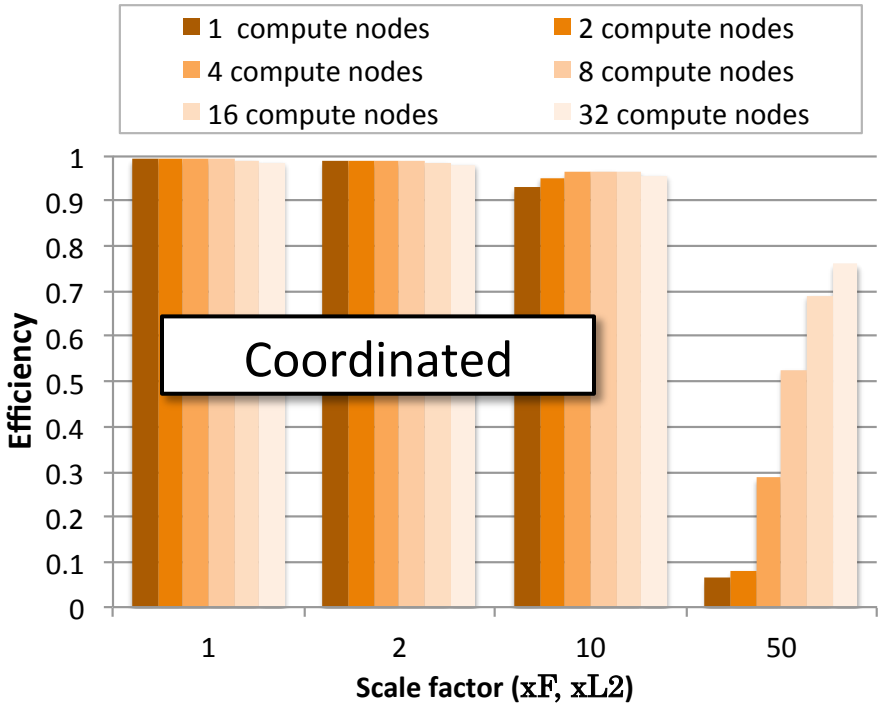
Improvement of level-1 storage performance does not impact efficiency for both flat buffer and burst buffer systems

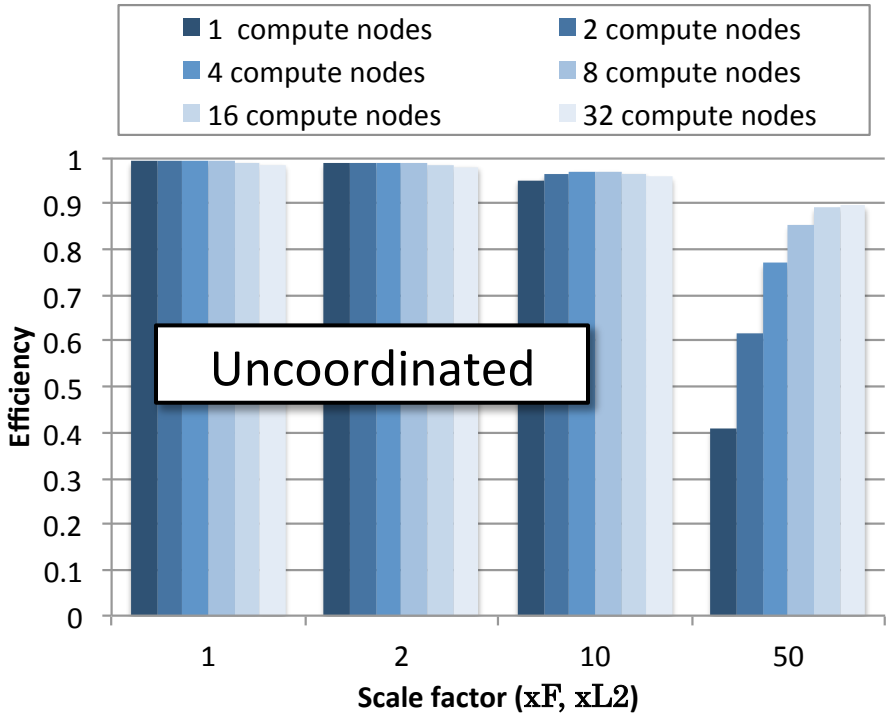
Increasing the performance of the PFS does impact system efficiency

L2 C/R overhead is a major cause of degrading efficiency, so reducing level-2 failure rate and improving level-2 C/R is critical on future systems

Ratio of Compute nodes to Burst Buffer nodes

Another thing to consider when building a burst buffer system is the ratio of compute nodes to burst buffer nodes





- The ratio is not important matter when MTBF is from a day to days
- When MTBF is a few hours, a larger number of burst buffer nodes decreases efficiency

  ⇒ Adding additional burst buffer nodes increases the failure rate which degrades system efficiency more than the efficiency gained by the increased bandwidth

Towards resilient extreme scale computing

1. Burst buffers
   - Burst buffers are beneficial for C/R at extreme scale

2. Uncoordinated C/R
   - When MTBF is days or a day, uncoordinated C/R may not be effective
   - If MTBF is a few hours or less, will be effective

3. Level-2 failure, and Level-2 performance
   - Reducing Level-2 failure and increasing Level-2 performance are critical to improve overall system efficiency

4. Fewer number of burst buffers
   - Adding additional burst buffer nodes increases the failure rate
   - May degrades system efficiency more than the efficiency gained by the increased bandwidth
   - We need to be careful a trade-off between I/O performance and reliability of burst buffers

Conclusion

- Fault tolerance is critical at extreme scale
  - Both C/R strategy and storage design are important
- We developed IBIO to maximize remote access to burst buffers, and modeled C/R strategy and storage design
- We listed up key factors to build resilient systems based on our evaluations
- We expect our findings can benefit system designers to create efficient and cost-effective systems

Q & A

Speaker:

Kento Sato (佐藤 賢斗)
kent@matsulab.is.titech.ac.jp
Tokyo Institute of Technology (Tokyo Tech)
Research Fellow of the Japan Society for the Promotion of Science
http://matsu-www.is.titech.ac.jp/~kent/index_en.html

Collaborators

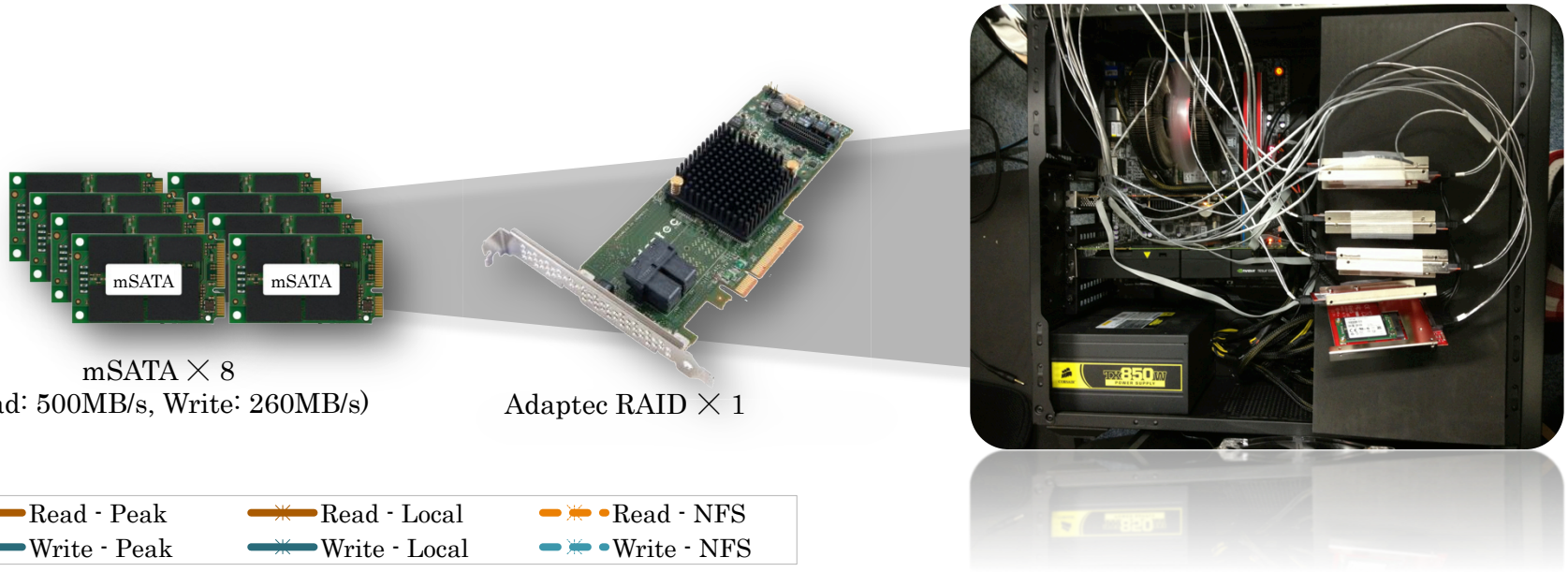
Kathryn Mohror, Adam Moody, Todd Gamblin, Bronis R de. Supinski,
Naoya Maruyama, Satoshi Matsuoka

Acknowledgement

This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. (LLNL-CONF-645209). This work was also supported by Grant-in-Aid for Research Fellow of the Japan Society for the Promotion of Science (JSPS Fellows) 24008253, and Grant-in-Aid for Scientific Research S 23220003.

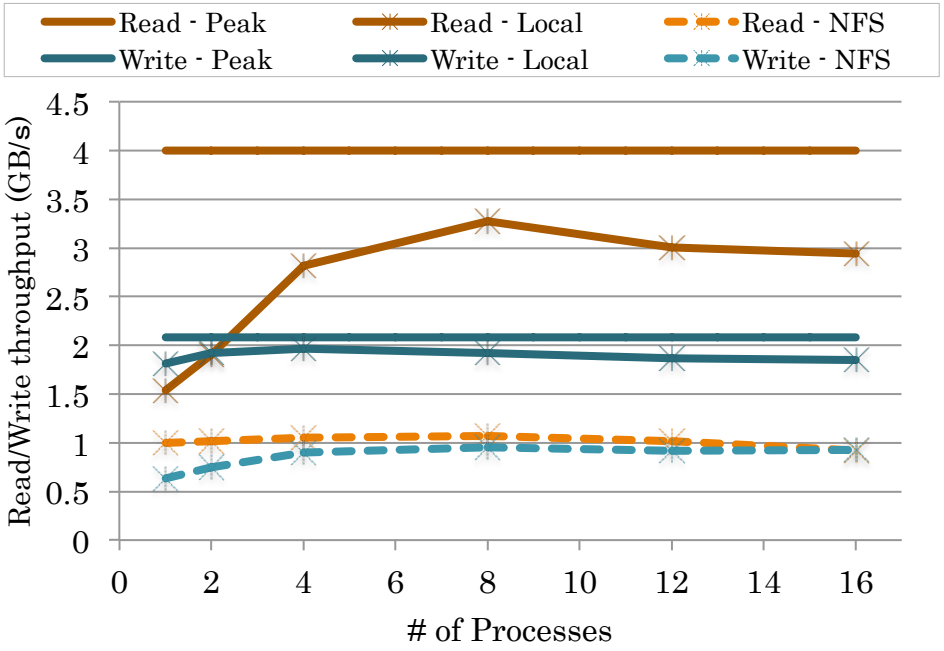
BACKUP

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



mSATA × 8
(Read: 500MB/s, Write: 260MB/s)

Adaptec RAID × 1



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)