Research on Scalable Software towards Exa-scale System

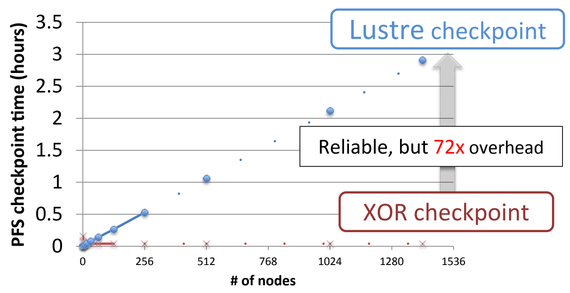
TOKYO TECH Pursuing Excellence

Scalable Asynchronous Checkpointing

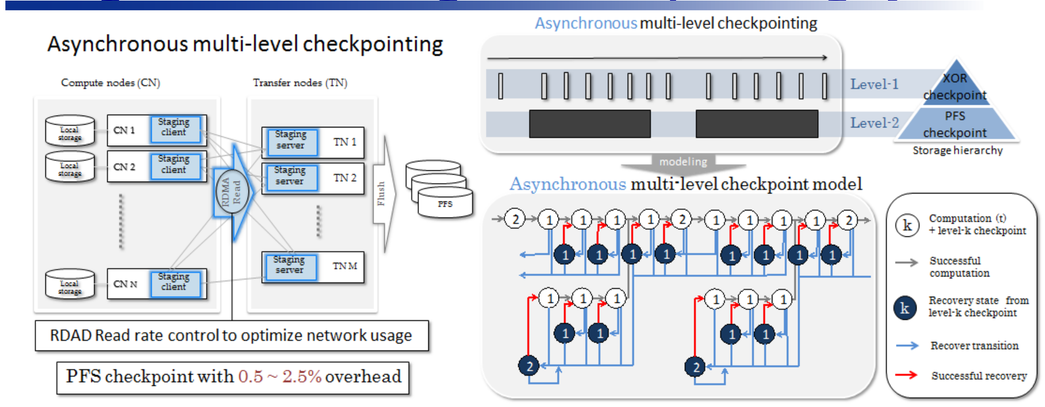
SC12 Tech Paper Tue 2:30pm Room 255-EF

- **Background**: Increasing system failures
  - e.g.) TSUBAME2.0@Tokyo Tech
    - 962 node failures (Nov 1st 2010 --- April 6th 2012)
    - A node failure occurred every 13.0 hours on average
  - A parallel file system (PFS) checkpointing overhead
- **Objective**: Reduce PFS checkpoint overhead
- **Proposed method**: Implementation and modeling an asynchronous checkpointing
  - Output to PFS via Staging nodes using RDMA
  - Determine the optimal checkpoint interval

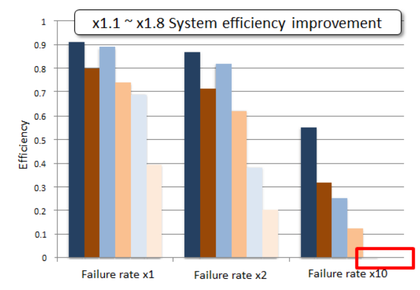
TSUBAME2.0 Lustre checkpoint time



Design and Modeling of Checkpointing System



Estimating Efficiency on TSUBAME2.0 System



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

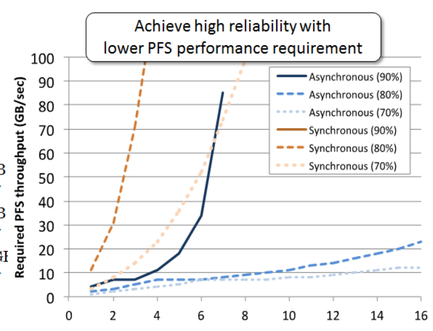
ideal_runtime: No failure and checkpoint

expected_runtime: Calculated by the models

- PFS cost x1 / Asynchronous
- PFS cost x1 / Synchronous
- PFS cost x2 / Asynchronous
- PFS cost x2 / Synchronous
- PFS cost x10 / Asynchronous
- PFS cost x10 / Synchronous

29/58GB memory; 58/58GB memory; 290/58GB memory

One TSUBAME thin node MTBF: 2.57 years
# of Nodes: 1408 nodes
x10 scale-out
No effective computation



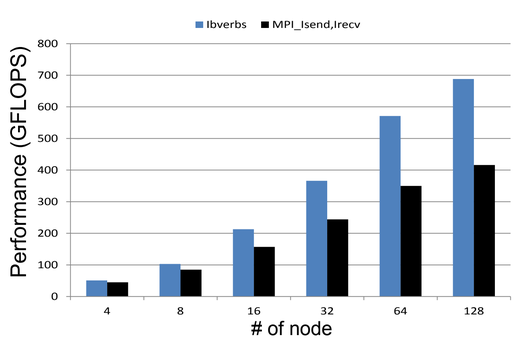
This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. LLNL-POST-582242

Scalable 3-D FFT on TSUBAME2.0

SC12 Tech Paper Wed 11:30am Room 255-BC

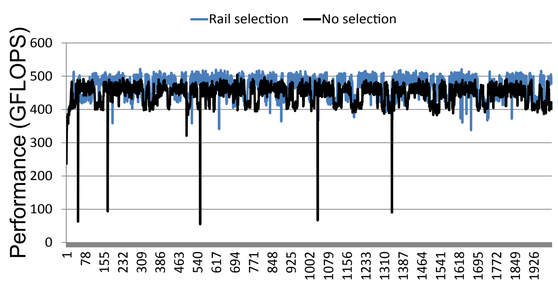
3-D FFT is used in many applications and contributes to reduce the amount of computation. Since many large-scale applications are accelerated by GPU clusters today, we need high-performance, scalable 3-D FFT for them. The performance is largely depends on the network data transfer (MPI and CUDA), because on-board computation is very fast. Multi-GPU 3-D FFT requires all-to-all communication between GPUs, which is a big challenge in large-scale systems.

Minimizing Overhead



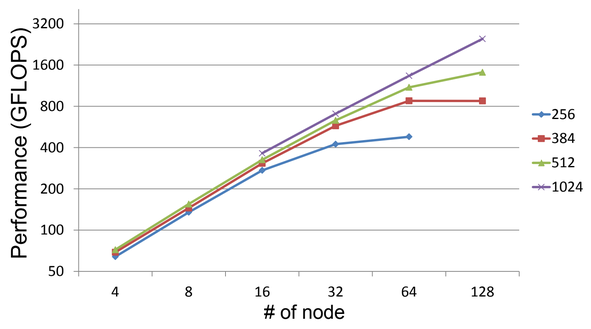
Usually, applications with 3-D FFT require scalable performance in strong-scaling. In all-to-all communication, this means the message size becomes smaller by increasing the number of nodes.
To minimize the overheads, we use ibverbs, which is a low-level API for IB. This allows full control of memory buffers, IB devices, and other resources.

Dynamic Rail Selection



Large-scale InfiniBand network consists of too many components. Occasionally some of them have problems which result in slow-down of data transfers. Even if we could remove all the errors, there still be a problem of network congestion that comes from load-imbalance of IB routing. Fortunately, IB network of some GPU-accelerated supercomputers have multiple IB rails. In this case, we can reduce the network congestion by dynamic rail selection.

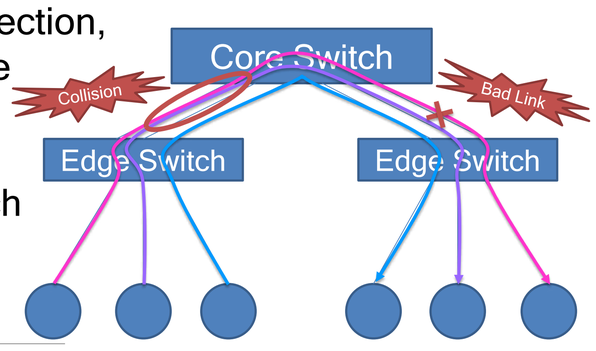
Scalability

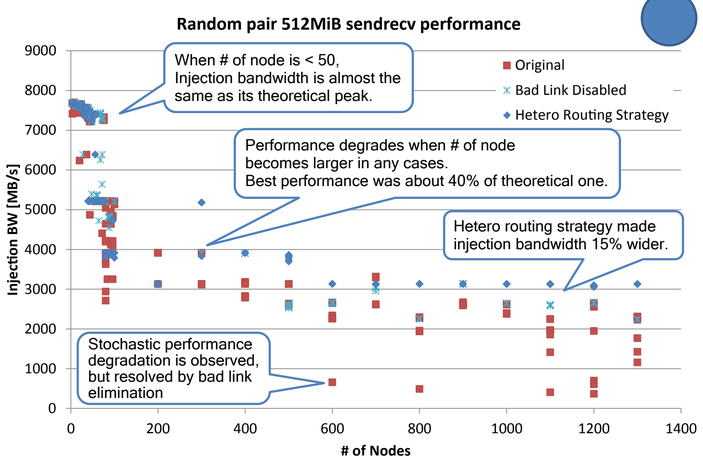


Using those optimizations, we achieved high scalability on TSUBAME 2.0 operating as usual. For larger transform sizes, three GPU per node case is faster than single GPU per node case. Remember that there is load-imbalance between three GPU on each node.

Improving Network Performance of TSUBAME2.0

Computing nodes in TSUBAME2.0 are connected by two rails of 4xQDR InfiniBand network and each rail forms fat-tree topology. Theoretically, the network is full-bisection, however, the observed performance becomes worse when many node communicate at once. This affects the performance of application which uses collective communications.



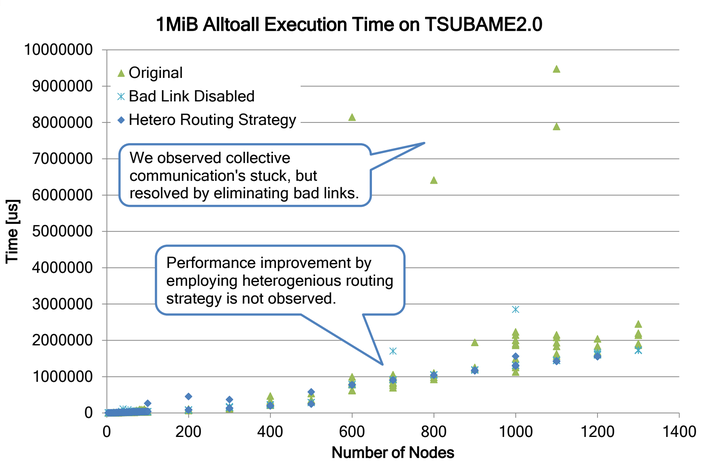


We found some degraded links:
- Slower link speed (e.g. 1xQDR)
- Malfunctioning link (e.g. excessive retransmit)

By disabling those links, we could resolve communication stuck observed in our network.

We also tried changing routing strategies to avoid packet collision in upper tier. Heterogeneous routing strategy is introduced not to share the congested route between rails. Performance improvement is observed in injection bandwidth test, but no improvement observed in collective communication performance.

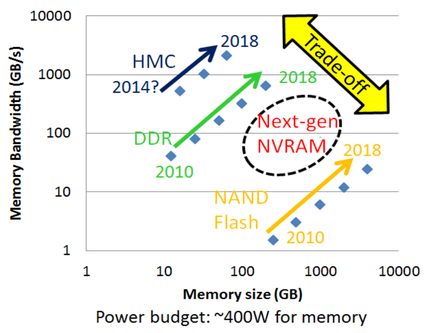
We will continue the research to provide better communication performance.



Memory Hierarchy Aware Software Stack

In Exa-scale supercomputing systems, the "**memory wall**" problem will become even higher, which prevents the realization of exa-scale real world simulations.
The approaches of our new project, "Software Technology that Deals with Deeper Memory Hierarchy in Post-petascale Era" are:

(1) To suppose supercomputer architecture with **deeper memory hierarchy** including hybrid memory devices, including non-volatile RAM (NVRAM).



Power budget: ~400W for memory

Hybrid Memory Cube (HMC):
DRAM chips are stacked with TSV technology. It will have advantage in bandwidth over DDR, but capacity will be smaller.

NAND Flash:
SSDs are already commodity. Newer products, such as IO-drive have O(GB/s) bandwidth.

Next-gen non-volatile RAM (NVRAM):
Several kinds of NVRAM such as STT-MRAM, ReRAM, FeRAM, etc. will be available in a few years.

(2) To develop new **software technology that efficiently utilizes the hybrid memory hierarchy**. The area of our research includes new compiler technology, scalable runtime system and application algorithms.

App Level Locality-aware Techniques
Temporal blocking, removal of matrix with re-computation...

Locality-aware Compiler
Loop-tiling, loop-fusion, data layout transformation, by using dynamic compilation techniques

Hierarchy-aware Scalable Runtime
Data movement over memory hierarchy including remote nodes, latency hiding

Upcoming Memory Architecture

DDR only architecture (in 2018--2020)
- 1EFlop/s
- 32PB/s → B/F = 0.032
- 10PB → B/(F/s) = 0.01

~10x worse than in current architecture!!

Our target (in 2018--2020)
- 1EFlop/s
- 100PB/s → B/F = 0.1 or higher
  BW of HMC. Our compiler makes user-visible BW even higher
- 100PB → B/(F/s) = 0.1
  Capacity of Flash. With our runtime, Flash can be used seamlessly with other memory hierarchy.

Supported by JST-CREST

http://www.gsic.titech.ac.jp/sc12