Scalable Asynchronous Checkpointing

**Objective**
- Reduce PFS checkpoint overhead
- Output to PFS via Staging nodes using RDMA

**Proposed method** Implementation and modeling of an asynchronous checkpointing.
- Output to PFS via Staging nodes using RDMA
- Determine the optimal checkpoint interval

**Background**
- Increasing system failures
- 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

Our target (in 2018–2020)

**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).

(2) To develop new software technology that efficiently utilizes the hybrid memory hierarchy.

**Estimated Efficiency on TSUBAME2.0 System**

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

**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 nodes communicate at once. This affects the performance of application which uses collective communications.

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.

**Scalable 3-D FFT on TSUBAME2.0**

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.

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 bitvectors, which is a low-level API for IB. This allows full control of memory buffers, IB devices, and other resources.

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.

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.

**Room 255-EF**

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

Supported by JST-CREST