Towards Extreme-scale Graph Processing with Deepening Memory Hierarchy

Hitoshi Sato

hitoshi.sato@gsic.titech.ac.jp

Tokyo Institute of Technology

1. Introduction

Emerging new commodity devices such as GPU accelerators and NVM (Non-Volatile Memory) devices are key technologies for future extreme-scale big data processing. For example, GPUs can provide high peak performance and rich memory bandwidth for applications with specific workload patterns, while CPUs offer flexibility and generality over wide-ranging classes of applications. Also, NVMs such as Flash have positive aspects of inexpensive cost, high energy-efficiency, and huge capacity compared with conventional DRAM devices, as well as negative aspects of low throughput and latency. These new devises may bring various benefits to the design of BigData-oriented extreme-scale machines; however, side effects of deepening memory hierarchy may require elaborate data management for application programmers, and efficient implementation techniques and abstractions for algorithms and data structures for big data to overcome deepening memory hierarchy are becoming essential.

To address the above issues, we have developed an extended memory software stack for supporting extreme-scale graph computing, as an instance of big data applications for extreme-scale computing. Utilizing GPU and NVM devices as hierarchical semi-external memory layers, we design highly efficient hierarchical data management techniques, fast data offloading techniques, and optimized I/O interfaces for graph applications. This document introduces overviews of our work towards extreme-scale graph processing for future computing systems with deepening memory hierarchy.

2. GPU-MapReduce-based Graph Processing for Large-scale Systems

MapReduce is a successful programing model for efficient scalable massive data processing in clouds with large-scale commodity compute clusters, since MapReduce can conceal elaborate efforts in distributed systems, such as localized data access for petabyte-scale large data volumes, communication between thousands of nodes, and fault tolerance, etc. On the other hand, MapReduce may be a good programming model for GPU accelerators for hiding massive parallelism and deep hierarchical memory. However, how much MapReduce-based applications can be accelerated on large-scale GPU-based heterogeneous clusters is an open problem.

In order to investigate the above issue, we have developed a MapReduce-based Graph application for multi-GPU environments, as an instance of MapReduce-based GPU applications[1]. We extend the existing single-GPU-based MapReduce library (Mars) for supporting multi-GPU environments using the MPI library and implements several graph kernels, such as PageRank, Random Walk, and Connected Component, etc., based on the MapReduce-based GIM-V (Generalized Iterative Matrix-Vector multiplication) algorithm. Our implementation also includes several optimization and load balance techniques.

Experimental results on the TSUBAME2.0 supercomputer using 256 nodes (6144 hyper-threaded CPU cores, 768 GPUs) showed that our GPU-based implementation performed 87.04 ME/s on 2^{30} (1.07 billion) vertices and 2^{34} (17.2 billion) edges, and 1.52 times faster than the CPU-based naive implementation with 2^{29} and 2^{33} edges.

We are currently implementing the original GPU-Mapreduce-based framework based on the above experiences and improving data management features to support large-scale data sets that exceed the memory capacity of GPU devices.

3. Hybrid BFS Approach Using Semi-External Memory

As an instance of implementation to overcome deepening memory hierarchy in extreme-scale computing systems, we have developed a fast graph processing implementation[2] with a novel graph offloading technique using NVMs for Hybrid BFS (Breadth-First Search) algorithm[3] that is widely used in the Graph500 benchmark. Hybrid BFS uses a mixture of two approaches, a conventional top-down approach and a bottom-up approach by changing search directions using parameters. Based on our collaborator's NUMA-based optimized Hybrid BFS implementation called NETAL (NETwork Analysis Library), which achieves 10.5 GTEPS on the Graph500 list (November 2012), we carefully offload infrequent accessed graph data to secondary semi-external memory devices such as NVMs and directly read the data on the devices on demand.

Based on the results using Graph500 problems, we confirmed that our approach with highly localized data access can achieve competitive performance with the conventional DRAM only approach, although we aggressively extend memory footprints onto NVMs. Using the implementation with some improvements, we have also achieved 4.35MTEPS/Watt on a Scale 30 problem and ranked the 4th position in the big data category in the Green Graph500 (November 2013), which is the 1st position using a single commodity server in the big data category.

4. Local Storage Configuration for Supporting Extreme Big Data I/O

Recent modern supercomputers designed for performing I/O-intensive operations tend to be equipped with NVM devices as local storage volumes on their compute nodes. For example, TokyoTech's TSUBAME2 has about 200TB of SSDs in total as local storage volumes. Basically, NVMs are costly devices compared with conventional HDDs, so how to utilize these NVMs as local storage volumes at a low cost with high performance, large capacity, and low energy consumption for supporting HPC and BigData applications, especially combining with many-core accelerator, is becoming an open issue.

In order to support various massive I/O operations from GPUs and NVMs using cheap commodity devices, we have designed a novel I/O prototype machine that consist of multiple GPU accelerators and multiple mini SATA (mSATA) SSD devices. In particular, aggregation of multiple mSATA SSDs and strategy for utilization the GPU and NVM devices are essential to perform high bandwidth, high IOPS as well as large capacity and low energy consumption.

Our preliminary results based on basic I/O bandwidth to the prototype with 16 of mSATA SSD devices by fio I/O benchmarks exhibit that the sequential read bandwidth achieves 7.69GB/s (92.4% of theoretical peak), and the write bandwidth achieves 3.8GB/s (90.2% of theoretical peak). The results also exhibit that using multiple mSATA SSDs performs 3.20 to 7.60 times faster than a common PCI-e attached flash memory device, and 11.1 to 17.8 times faster than conventional SSDs attached on a local compute node of TSUBAME 2.5. We also measure the performance of a matrix-vector multiplication benchmark that overlaps file I/O from/to NVMs, memory copy between host and GPUs, and computation on GPUs. The results using 280MB to 140GB of input matrices, which exceed the GPU memory capacity, exhibit that our prototype can achieve 3.06GB/s from 8 mSATA SSDs to a GPU device by using the RAID0 configuration with appropriate stripe size. We also found that using pinned memory and setting small chunk size for overlapping significantly affect data transfer performance.

References

[1] Koichi Shirahata, Hitoshi Sato, Toyotaro Suzumura, Satoshi Matsuoka. A Scalable Implementation of a MapReduce-based Graph Processing Algorithm for Large-scale Heterogeneous Supercomputers. In proceedings of the 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid2013), May 2013.

[2] Keita Iwabuchi, Hitoshi Sato, Yuichiro Yasui, Katsuki Fujisawa. Performance Analysis of Hybrid BFS Approach Using Semi-External Memory. In proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'13) (Poster), November 2013.

[3] Scott Beamer, Krste Asanović, David Patterson. Direction-optimizing breadth-first search. In proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC'12), November 2012.