Satisfying Data-Intensive Queries Using GPU Clusters

Jeffrey Young, Haicheng Wu, Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology
Atlanta, Georgia, USA
{*jyoung9, hwu36*}*@gatech.edu, sudha@ece.gatech.edu*

***Abstract*—Data-intensive queries should be run on GPU clusters to increase throughput, and Global Address Spaces (GAS) should be used to support compiler optimizations that can increase total throughput by fully utilizing memory and GPUs across nodes in the cluster.**

***Keywords*-GPU clusters; data warehousing; compiler optimizations; global address spaces**

I. A New GPU Cluster Prototype

We propose that data-intensive queries should be run on GPU clusters for higher throughput. Large database applications could benefit from multi-GPU systems, but there are three major challenges to achieving significant advances in throughput. The first challenge is the efficient and effective implementation of database queries on GPUs. The second challenge is the limitations of traditional memory hierarchies, specifically the limited DRAM of the host environment to which the GPUs are connected. This latter challenge is amplified by the PCIe interconnection between the host and GPUs. Finally, clusters have limited inter-node bandwidth, which provides further constraints on data movement to GPUs. We propose to improve the throughput of GPU queries for large data sets by taking advantage of compiler optimizations that can improve data reuse on GPUs and by using GAS to increase the performance of data movement between different devices on multiple nodes. We are currently constructing a system that will provide the capability to run data-intensive GPU queries using a compiler framework called Red Fox and a GAS API called Oncilla [1], as shown in Figure 1.

The Red Fox compiler is designed to run complex queries with large amounts of data for a heterogeneous cluster. It is comprised of: i) A front-end to parse a descriptive language such as Datalog and to create an optimized query plan in the form of a graph of Relational Algebra (RA) primitives; ii) A compiler to map these RA primitives to their corresponding GPU kernels [2]; iii) A data movement module that leverages GAS hardware to automatically apply Kernel Fusion/Fission (KFF) optimizations that reduce the amount of data transferred and overlap data movement with computation [3][4]; iv) A runtime management system to orchestrate the above three components dynamically and to schedule the execution of primitives and data transfers on a hybrid cluster connected by GAS hardware.

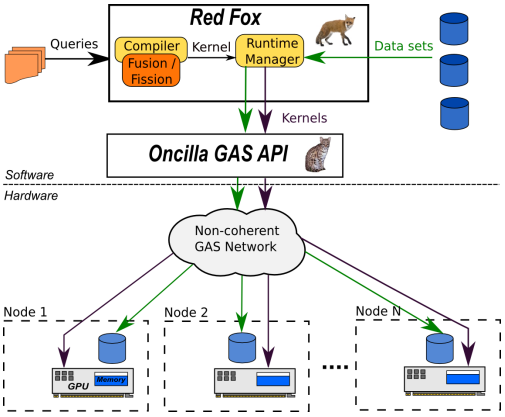

Figure 1. GPU cluster infrastructure

The Oncilla API enables the creation of a combined GAS for remote and local host memory and GPU memory on top of a simplified hardware interface for applications. The Red Fox system can then use this API to implement compiler optimizations and to schedule kernels across nodes and on multiple GPUs. Specifically, Oncilla can be used to provide high-bandwidth data transfer for KFF-based queries and for data sets between GPUs on different nodes.

Preliminary experiments with Query 1 of the TPC-H benchmark suite shows that the Red Fox framework generates a 4x speedup against traditional CPU-based database systems, even when KFF is not turned on. Moreover, the KFF techniques have shown more than 2x speedup when running on some micro-benchmarks picked from the TPC-H queries. These preliminary results lead us to believe that this infrastructure can handle data-intensive queries that are common in the HPC arena.

References

[1] S. Yalamanchili *et al.*, "Oncilla - Optimizing accelerator clouds for data warehousing applications (white paper)," 2012.

[2] G. Diamos *et al.*, "Relational algorithms for multi-bulk-synchronous processors," *PPoPP*, 2013.

[3] H. Wu *et al.*, "Kernel weaver: Automatically fusing database primitives for efficient GPU computation," *MICRO*, 2012.

[4] H. Wu *et al.*, "Optimizing data warehousing applications for GPUs using kernel fusion/fission," *IPDPS-PLC*, 2012.