Oncilla - Optimizing Accelerator Clouds for Data Warehousing Applications

Sudhakar Yalamanchili, Jeffrey Young, Ashwin Lele

School of Electrical and Computer Engineering
Georgia Institute of Technology, Atlanta, GA

1 Problem Statement and Overview

Data warehousing applications represent an emergent application arena that requires the processing of queries and computations over massive amounts of data. Large-scale, multi-GPU systems potentially present a vehicle for major improvements in throughput and consequently overall performance. We are addressing two major challenges to achieving significant advances in throughput using scalable clusters with GPUs. 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.

To address the first challenge, we utilize optimized primitives for the implementation of relational database operators that have been created by our group as part of the Red Fox compiler project. Built on Ocelot, Red Fox is used to compile queries implemented in Datalog to CUDA kernels that run on NVIDIA GPUs. To address the second challenge, we propose to construct an experimental cluster with special-purpose network interface cards (NICs) that provide hardware support to aggregate DRAM across nodes and transparently make this aggregated memory available to the local GPUs. Abstractly, the goal is to change the ratio of host memory to GPUs by aggregating memory across cluster nodes. Operationally, a host memory access now appears as a NUMA memory access for GPU transfers from a global, non-coherent physical address space, as shown in Figure 1. Advances in networking coupled with NVDIA's GPUDirect technology make such high performance aggregation of memory and GPUs possible.

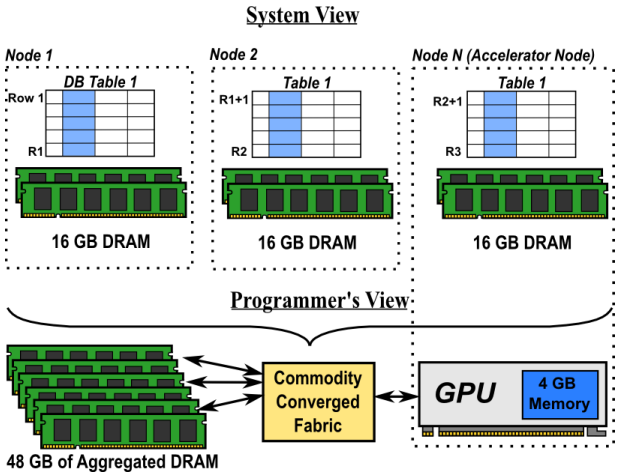

Figure 1: Conceptual View of Non-coherent Global Address Space for Memory Aggregation

This project is a collaboration with several European colleagues (Holger Fröning at University of Heidelberg and Federico Silla and José Duato at Polytechnic University of Valencia) along with three active industry partners with an interest in advancing GPU technology for data warehousing applications - LogicBlox Inc., Advanced Industrial Computer Inc. (AIC), and NEC. These companies represent three different segments of the data center marketplace. LogicBlox is a supplier of enterprise software solutions to the retail industry, and AIC is a blade vendor who is providing the node platforms and much of the hardware for our experimental cluster. NEC is collaborating with us on compiler & scheduling optimizations and application requirements.

2 The Hardware Testbed

Our proposed approach is made possible by the use of special-purpose EXTOLL NICs that are developed at the University of Heidelberg and are used in the MEMSCALE project [4]. The EXTOLL programmable NIC [3] provides a commodity-based solution (via an on-board HyperTransport interface, aka HTX) to support non-coherent remote memory accesses to globally addressable memory that is both high-performance and has lower programmer overhead than traditional techniques such as message passing. The EXTOLL prototype has shown bandwidths similar to Infiniband (~6-25 Gbps) and latencies that are typically lower than Infiniband, especially for small messages (~2-3 µs) [3].At the same time, EXTOLL provides system support for establishing non-coherent shared memory regions across nodes [4]. We propose building on this low-level layer to provide seamless aggregation of memory *and* GPUs across nodes (e.g., extend NVIDIA's UVA across nodes) by exposing this hardware to our Red Fox compiler. An outline of this infrastructure for the two node case is shown in Figure 2.

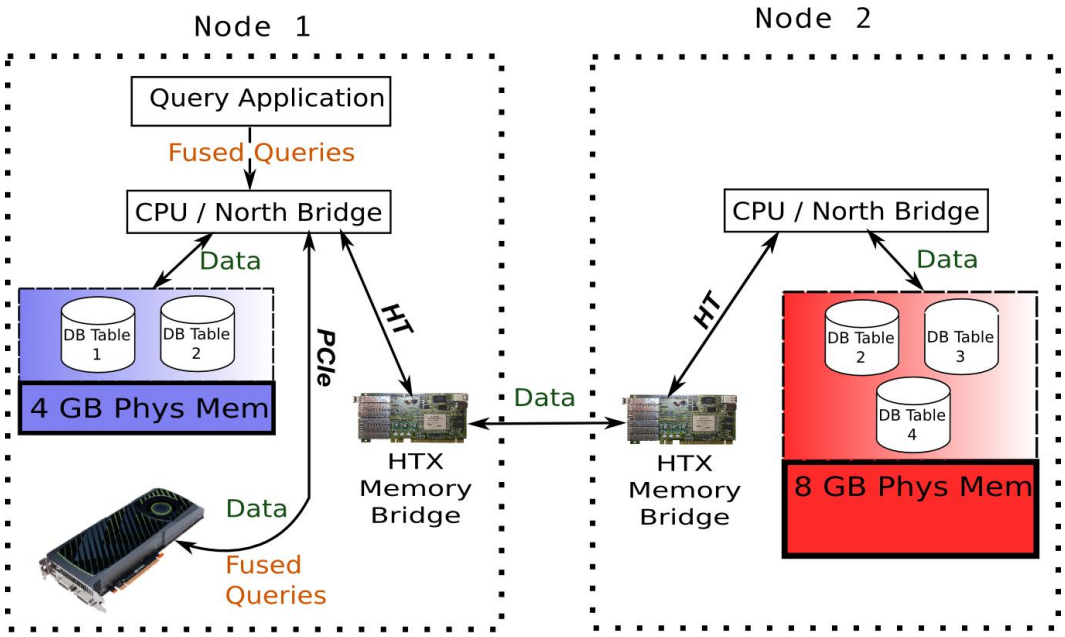

Figure 2: Demonstration of GAS support for Multi-Node Accelerators, providing a flat 12 GB physical address space for both nodes

3 Technical Approach

Our groups overall approach towards addressing the use of GPUs for data warehousing applications is comprised of three thrusts.

1. Implementation of Primitives Optimized for GPUs: The first set of primitives that have been implemented are relational algebra primitives.
2. Optimizations for Data Movement: We are currently developing (with NEC) compiler-based data movement optimizations across PCIe in a single node. An extension of this thrust focuses on inter-node data movement between GPUs and aggregated memory.
3. Memory and GPU Aggregation: Exploration of hardware and software techniques for seamless aggregation of memory and GPUs across nodes, which will enable much larger data sets to be processed before hitting the disk wall.

The front-end of Red Fox is based on the Datalog declarative language while the back-end is based on Ocelot and our Harmony runtime. LogicBlox has implemented all TPC-H queries in Datalog, and we are currently implementing these queries in CUDA as a point of comparison. Towards this end, our group has implemented efficient versions of relational algebra primitives on NVIDIA GPUs (the author of these primitives is now with NVIDIA Research – Greg Diamos). We are now engaged in robustness and scalability testing of these primitives on the Amazon EC2 cloud with GPU instances.

In many cases, the performance of these cluster and cloud-based architectures is generally limited by the amount of host memory per node and the inter-node latency and bandwidth for inter-GPU data movement and sharing. For this reason, we seek to explore the use of memory and GPU aggregation across clusters using a low-latency interconnection fabric, such as EXTOLL. The availability of high-performance, transparent memory aggregation effectively enables dynamic changes in the ratio of host DRAM accessible to GPUs, enabling much larger data sets to be processed by the GPU before going to disk or SSD. Our project aims to demonstrate that this larger host memory ratio provides demonstrable benefits to more efficient GPU processing of queries across large data sets.

To reach this goal, we will first port the Ocelot dynamic compilation environment to our experimental cluster. The primary change will be to the existing socket-based remote device interface that launches kernels on remote nodes – this must change this interface to use lower-latency shared memory via the EXTOLL NIC. We can then use the back-end of the Red Fox enterprise compiler (built on Ocelot) to implement all of our optimized relational algebra operators on local and remote GPUs. Of particular interest will be the ability to process relational queries on a GPU where the data set spans multiple nodes. Low-overhead orchestration of data movement will be key here, which is why we believe that the usage of the low-latency EXTOLL network and GPUDirect's shared memory pages is important to create a high-performance implementation.

Once this basic infrastructure is in place, performance measurements and analysis will become the main focus. We will then seek to apply GPUDirect 1.0 (use of CUDA-pinned pages) to improve query performance when the data sets span multiple host nodes and then study the opportunity presented by GPUDirect 2.0 (we expect to move from HTX-based interfaces to PCIe-based interfaces in the future) to send data between the

programmable EXTOLL NIC and a GPU. These two goals should enable a natural extension of NVIDIA's Unified Virtual Address space across nodes while enhancing the performance of copying input data between nodes. We plan to test the full suite of TPC-H benchmarks which are available in Datalog from LogicBlox and several of which have been implemented by our group in CUDA. While other projects [4] have looked at in-core databases that support global address spaces or accelerating database queries on GPUs [1], to the best of our knowledge no one has yet investigated the creation of a non-coherent, commodity implementation of a global address space model that also includes GPUs.

4 Collaborations and Points of Leverage

This work has multiple points of leverage. We have an active collaboration with NEC that is implementing kernel-level optimizations (kernel fusion, fission, and scheduling) and data movement optimizations using optimized primitives we have developed for relational algebra operators (as well as others). These are currently targeted to NEC's cluster and Amazon EC2. Our collaboration with LogicBlox and the Datalog front-end is also currently focused on the Amazon EC2 implementation. Both NEC and LogicBlox have an interest in understanding memory and GPU aggregation technology.

We are also collaborating with European colleagues who are interested in using GAS systems to support in-core databases (and who provided the EXTOLL NICs). Our colleagues have a 1024 core shared memory cluster that uses the EXTOLL cards (the MEMSCALE project) and a modified MySQL instance to implement a basic in-core database using non-coherent shared global memory. We plan to investigate further the integration of our Red Fox back-end with their MySQL front-end, possibly resulting in a multi-GPU, large memory (aggregated across commodity nodes) implementation of MySQL that uses NVIDIA GPUs.

5 Related Work and Motivation

The increased usage of heterogeneous architectures to handle highly-parallel data warehousing tasks has increased in the previous years while at the same time in-core databases have proliferated as an alternative to traditional disk-based storage [4][7][8]. The benefits of in-core databases are evident due to DRAM access latencies in the range of tens of nanoseconds compared to 30 or more microseconds for the current fastest PCIe-based SSDs [6]. At the same time, the benefits of parallel GPU implementations for processing of large data sets in data warehousing applications is just beginning to be explored due to the difficulty of parallel multi-GPU implementations and due to the limitations of transferring data between the memory, CPU cache, and I/O domains.

One of the difficulties in using accelerators for in-core databases is the need to efficiently move large amounts of data between nodes. Traditional techniques such as message passing and RDMA suffer from high setup overheads [2] as well as the complexity of moving data between coherence domains on separate nodes. At the same time, techniques such as software NUMA typically depend on custom, expensive interconnects [5]. As we have illustrated above, one potential solution to these issues is to use a global address space model, non-coherent memory sharing, and commodity hardware for a low-cost implementation to provide a large, aggregated amount of DRAM.

References

1. Bakkum, P., et al. *Accelerating SQL database operations on a GPU with CUDA*. GPGPU '10.
2. Fröning, H. et al., *Efficient Hardware Support for the Partitioned Global Address Space*. 10th Workshop on Communication Architecture for Clusters (CAC 2010). Paper Link.
3. Fröning, H. *EXTOLL Introduction*. HPC User Forum, HLRS and SARA Meeting, Amsterdam, Netherlands, October 2010. Presentation link.
4. Montaner, H. et al. *MEMSCALE™: A Scalable Environment for Databases.* 13th IEEE International Conference on High Performance Computing and Communications (HPCC-2011), Sept. 2-4, 2011, Banff, Canada. Paper link.
5. *Numascale Numaconnect Whitepaper*. Accessed December, 2011.
6. *OCZ Z-Drive R4 SSD Review*. Published September 2011.
7. Ousterhout, J. et al. *The Case for RAMClouds: Scalable High-Performance Storage Entirely in DRAM*, SIGOPS Operating Systems Review, Vol. 43, No. 4, December 2009, pp. 92-105 Paper link.
8. *SAP Hana In-memory Database Overview*. Accessed December, 2011.