Commodity Converged Fabrics for Global Address Spaces in Accelerator Clouds

Jeffrey Young and Sudhakar Yalamanchili
Department of Electrical and Computer Engineering
Georgia Institute of Technology
jyoung9@gatech.edu, *sudha@ece.gatech.edu*

***Abstract*—Hardware support for Global Address Spaces (GAS) has previously focused on providing efficient access across remote memories, typically using custom interconnects or high-level software layers. New technologies, such as Extoll, HyperShare, and NumaConnect now allow for cheaper ways to build GAS support into the data center, thus making high-performance coherent and non-coherent remote memory access available for standard data center applications.**

At the same time, data center designers are currently experimenting with a greater use of accelerators like GPUs to enhance traditionally CPU-oriented processes, such as data warehousing queries for in-core databases. However, there are very few workable approaches for these accelerator clusters that both use commodity interconnects and also support simple multi-node programming models, such as GAS.

We propose a new commodity-based approach for supporting non-coherent GAS in accelerator clouds using the HyperTransport Consortium's HyperTransport over Ethernet (HToE) specification. This work details a system model for using HToE for accelerated data warehousing applications and investigates potential bottlenecks and design optimizations for an HToE network adapter, or HyperTransport Ethernet Adapter (HTEA).

Using a detailed network simulator model and timing measured for queries run on high-end GPUs [34], we find that the addition of wider deencapsulation pipelines and the use of bulk acknowledgments in the HTEA can improve overall throughput and reduce latency for multiple senders using a common accelerator. Furthermore, we show that the bandwidth of one receiving HTEA can vary from 2.8 Gbps to 24.45 Gbps, depending on the optimizations used, and the inter-HTEA latency for one packet is 1,480 ns. A brief analysis of the path from remote memory to accelerators also demonstrates that the bandwidth of today's GPUs can easily handle a stream-based computation model using HToE.

I. Introduction

Previous work in Partitioned Global Address Space (PGAS) systems [4] has permitted for the definition of a system-wide, shared address space that can be used to efficiently share memory between nodes for high-performance clusters. Many of these solutions have focused on creating software layers that can be used to easily program large systems, including the UPC language [30] and APIs that support PGAS on multiple hardware infrastructures, such as GASNet [3]. Explicit hardware support for GAS has long been the provenance of high-performance machines in the supercomputing arena [27], but emerging widespread, commercial solutions for GAS include high-performance point-to-point interconnects such as AMD's HyperTransport (HT) and Intel's QPI that allow for coherent [21] and non-coherent remote memory accesses [13]. Concurrently, the HyperTransport Consortium's HyperShare platform [7] has focused on specifications that provide low-level GAS hardware support for traditional data center fabrics like HT, Ethernet, and InfiniBand. Taken as a whole we refer to this new support for hardware GAS as "commodity converged fabrics", which means that these widely available fabrics support a combination of typical on-board and on-chip interconnects, such as PCIe, HyperTransport, and QPI, and off-chip interconnects, such as Ethernet, InfiniBand, PCIe, and HT. This definition of commodity converged fabrics also implies widely available interconnects that are being marketed towards data center usage. Cray's HyperTransport-based SeaStar and Gemini interconnects are related technology but would not be "commodity" by this definition.

At the same time that commodity GAS is becoming viable for supporting remote memory in the data center, new compute accelerators have emerged as cost-effective and more efficient multiprocessors for certain data center workloads. These accelerators, specifically general purpose graphics processing units (GPGPUs) such as NVIDIA's Fermi and AMD's Fusion, have introduced new types of compute platforms that contain on-board memory but that are still limited by the bandwidth of transfers to and from host-based memory (DRAM). GPUs hold tremendous potential for accelerating certain data center applications, including complex queries for in-core data warehousing applications [34], but efficient movement of large data sets between disks, DRAM, and GPUs still remains a challenging performance optimization problem.

This problem is further complicated with in-core databases that use large amounts of aggregated DRAM across multiple nodes to minimize accesses to slower disk drives. In addition, the cost and power requirements of current GPUs means that equipment cost and Total Cost of Ownership (TCO) may lead to a data center where not every blade has a local GPU but rather must share one high-end GPU between multiple nodes. In this case, the performance of these shared accelerators is limited by the amount of on-board memory on the GPU, which limits performance for large data sets that may need to be copied to and from host memory or disk drives. NVIDIA has

This research was supported in part by NSF grant CCF-0905459

greatly improved the speed and ease of data movement within a node with the introduction of the GAS-like Unified Virtual Address (UVA) space [8], but efficient and programmer-friendly inter-node communication still remains an unsolved problem.

This paper advocates and evaluates a system approach to the use of GPU accelerators for large data problems in the data center. Specifically, we are concerned with the following problem: How can large data sets for in-core data warehousing applications easily be moved between host memory and GPUs in a multi-node environment, and how can we enable this data movement with minimal changes to on-board hardware and existing software stacks? Our approach has the following elements: First, we propose the use of a system-wide, non-coherent, global physical address space (GAS) where the amount of physical host memory directly accessible to a GPU accelerator can be dynamically changed by transparently accessing remote DRAM. Second, we propose to implement hardware support for this GAS using a new converged fabric - HyperTransport over Ethernet (HToE). Third, we explore the design of a HToE network interface card that enables the preceding capabilities. Fourth, we evaluate the performance of this network interface via detailed simulations driven by workload specifications derived from our CUDA-based implementations of some of the queries of the TPC-H benchmark suite.

The rest of this paper discusses our proposed system model in more detail as well as a detailed simulation model of a cluster based on the HToE specification. Section II discusses the motivation for using HToE and converged fabrics in general while Section IV introduces the basics of the HyperTransport protocol and HToE. In Section V we describe our system model and potential bottlenecks in accelerator and memory clouds as well as the design of the network adapter. Section VI discusses the experimental setup and specific optimizations that were tested, and Section VII discusses results from simulations that demonstrate the network adapter's important features and the effects of HTEA optimizations.

II. GAS and Converged Fabrics in Accelerator Clouds

The recent proliferation of low-latency commodity interconnects means that both traditional HPC clusters and newer data centers can easily incorporate dedicated hardware support for GAS-based memory sharing via one-sided put/get operations. In addition to software-oriented stacks such as GASNet, several recent specifications support dedicated hardware for commodity GAS. The HyperShare platform includes three such specifications: native HyperTransport [11], encapsulation of HT over Ethernet [36], and HT over InfiniBand (HToIB) [2]. Other groups have designed custom off-chip fabrics, such as the EXTOLL group and associated corporation [13] and NumaScale, which provides coherent HyperTransport support between nodes with its NumaConnect adapter [21]. In addition to these converged fabrics, a recent specification has proposed the use of InfiniBand's Verbs layer over Ethernet, also known as RDMA over Converged Enhanced Ethernet (RoCEE) [5],

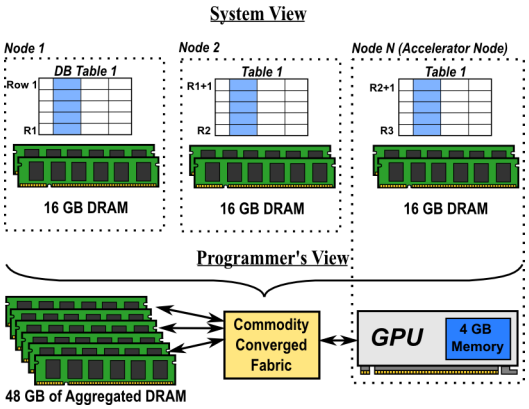

Figure 1. Logical Model of Accelerator Cloud Using Converged Fabrics

and some companies have shown interest in using PCI Express over Ethernet [28]. In fact, the new PCI Express 3.0 standard alludes to this future of converged fabrics with the addition of a "PMux" feature that supports sending packets from other protocols over PCIe links.

Each of these protocols support the use of converged commodity fabrics, but we have focused on HToE due to Ethernet's importance in the data center and the open nature of the non-coherent HyperTransport specification. As shown in Figure 1, our goal is to change the ratio of host memory available to GPUs by aggregating memory across cluster nodes while also enabling a simplified memory model for applications that use remote memory or GPUs. Operationally, the non-coherent, system-wide global address space means that transfers of data from remote host memory to GPU memory now appear as a NUMA memory access. This allows data center designers to be able to size accelerator clouds for an appropriate, domain-specific acceleration capacity and to scale these clouds without worrying about coherency traffic constraints.

Further reasons for focusing on an implementation of the HToE specification are outlined in detail in [35], but the most relevant comparison points are: 1) Ethernet has a large install base and many data center operators are well-versed in its usage 2) For many data center operators, the latency and bandwidth of 10, 40, and 100 Gigabit Ethernet may be "good enough" for many of their applications 3) the HToE specification is supported by "enhanced" Ethernet standards [18] that provide features that make Ethernet more comparable to InfiniBand (e.g., support for lossless operation). These new standards have led vendors to create other new "converged" specifications, including RoCEE and FibreChannel over Ethernet (FCoE). 4) A low-level encapsulation protocol such as HToE has several advantages related to overhead with respect to other RDMA-based protocols, including RoCEE. These advantages include a reduced need for OS-based data transfer setup costs in the tens to hundreds of microseconds [9] and easier programming due to a simpler API.

In short, while many of the previously discussed protocols could support our proposed logical model for accelerator clouds, we have chosen HToE due to its open, commodity nature, its support for low-latency data transfer, and especially

for its simplified programming model.

III. Related Work

Global address spaces and converged fabrics have traditionally been investigated in custom hardware built for supercomputers, including Cray's coherent Opteron-based interconnects, SeaStar, and its successor, Gemini [32]. Recently, other research groups have been investigating fine-grained, high-performance hardware support for GAS systems, specifically the Extoll project [13] and the many variants of GASNet that typically support UPC applications on a variety of "conduits". HyperTransport over Ethernet is comparable to the Extoll interconnect in that both support fine-grained transfers and non-coherent access for high-performance communication, although EXTOLL favors a direct-connected network as opposed to using standard Ethernet switches. The creators of EXTOLL also have their own in-core, non-coherent memory database project called MEMSCALE [20] and an RDMA software layer that supports remote CUDA execution for accessing remote GPUs called rCUDA [10].

With regards to GASNet implementations and related infrastructure, we posit that HToE is more focused on providing high-performance hardware to support varying GAS architectures than on providing a full PGAS HW and SW stack as is available using GASNet and UPC. An implementation of HToE could be considered similar to a GASNet "conduit" as discussed in [37].

Support for accessing remote accelerators has been greatly improved through the efforts of the high-performance community, specifically projects by NVIDIA and the MPI community. NVIDIA has greatly simplified device memory addressing through the use of their Unified Virtual Addressing in CUDA 4.0 and the inclusion of GPUDirect support that can facilitate peer-to-peer transfers between GPUs (GPUDirect 2.0) [25] and efficient transfers of data from the GPU to the NIC using pinned pages in the host operating system (GPUDirect 1.0) [26].

The MPI community has built on top of these improvements in CUDA to provide a seamless layer that integrates CUDA and MPI two-sided and one-sided transfer functionality. MVAPICH [33] and OpenMPI [31] have both provided implementations that allow for using MPI to efficiently transfer data between remote and local GPUs and remote memory. However, due to the current limitations of GPUDirect, both APIs must copy data through host memory when doing a transfer between a local and a remote GPU. The MVAPICH group also has started work on supporting standard data center applications by creating an interposer library that hides the complexity of the InfiniBand Verbs stack from the application developer [24]

Other groups have also investigated optimizations for in-core databases, most notably the RAMcloud project [23]. Oracle, SAP, and others also have commercial solutions, typically based on TCP/IP implementations. However, this work is one of the first to focus on the specific problem of using a limited number of accelerators along with GAS to improve the performance of in-core databases.

IV. HyperTransport and HyperTransport over Ethernet

The HyperTransport 3.1 specification [6] defines requirements for a low-latency, high-bandwidth interconnect for coherent memory access between sockets with a maximum one-way bandwidth of 25.6 GB/s. A non-coherent variant of HyperTransport can also interact with I/O devices connected to a motherboard by an HyperTransport eXpansion (HTX) connector, such as with the network adapter discussed in this work. HT packets contain either a 4 or 8 byte command word and up to 64 bytes of data, and each HT packet is classified according to one of three virtual channels: posted (no response needed), non-posted (response needed), and response packets (return data or notifications). Command and data packets are transmitted between HT devices using a credit-based flow control where one credit matches up with one physical buffer for either a command or data packet. Credits and changes in the link status are transmitted using HT NOP packets.

Due to HT's usage as an I/O protocol that requires deadlock-free messaging, the specification defines an ordering protocol that allows for a semi-relaxed ordering between packets in different virtual channels. For instance, non-posted packets can be defined to be part of a ordered sequence, and packet transmission must maintain virtual channels from different sources and destinations do not have any dependencies on ordering and may be reordered at will. This HT ordering requirement is important for HToE in that packets from the same source node must following ordering constraints. However, packets from different sources can be reordered with respect to each other, and the proposed receiver-based optimizations in Section VI-B take advantage of this feature.

A. HyperTransport over Ethernet

The HyperTransport over Ethernet specification [36] presents a basic framework for an L2 implementation of GAS hardware support for data centers using a non-coherent HT interconnect with 10, 40, or 100 Gigabit Ethernet. However, this specification leaves implementation details up to individuals or institutions that wish to build HToE hardware, which means that many performance-enhancing details can be specified based on the desired application. Our proposed system model is the first design to implement performance-related features for the HToE specification.

The unique properties of normal (lossy) Ethernet as well as the credit-based semantics of the HyperTransport protocol present unique challenges in designing high-performance hardware. While a related paper presents [35] the requirements in more detail, the requirements for flow control and ordering in the HToE adapter are listed here.

HToE uses credit-based flow control where one physical buffer corresponds to the space required to hold one HT command word (8 to 12 B) and one full HT data packet (up to 64 B). HToE flow control requires one standard HT credit for each HT packet that is encapsulated into an Ethernet frame. Flow control is managed using source-destination pairs (each with a unique MAC address), called Virtual Links (VL), and each Ethernet packet can contain multiple HyperTransport

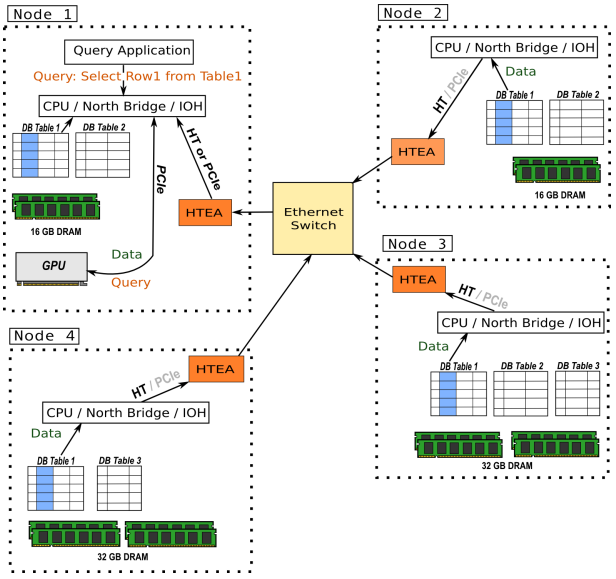

Figure 2. Four Node System Overview

packets and HT credits (NOPs) up to the maximum MTU. HT packets are queued at the sending HTEA in physical buffers while waiting for credits, and a lack of credits results in queuing from the sending HTEA back up to the sending processor or device on the local node. However, a lack of credits for one VL does not preclude packets from another VL being sent, as long as the sending VL has enough credits for the packets it is encapsulating.

Although HToE can take advantage of some of the newer IEEE Data Center Bridging or Converged Enhanced Ethernet standards such as as per-flow flow control [15], the HToE specification relies on a simpler end-to-end retry algorithm (Go-Back-N) to ensure that Ethernet packets encapsulating HT packets can operate over standard lossy Ethernet links purely at the link layer (L2 layer).

Since the HToE specification is designed mainly to support GAS-type memory sharing in data centers, it does not specify requirements on TCP/IP support for the HToE adapter. The HToE specification does provide for an EtherType flag to support handling both memory traffic and higher level protocols using the same adapter, and other hardware vendors, such as Mellanox, have already implemented similar solutions for their multi-protocol hardware [19]. For this reason, we do not investigate this requirement further.

V. System Model

Our proposed system includes N nodes which each have some amount of host memory on each node. Of these N nodes, a smaller subset will have a high-end GPU. While all nodes could be provisioned with GPUs, equipment cost and power as part of TCO means that many data center operators are unlikely to be able to provision GPUs with large amounts of on-board memory, such as NVIDIA Tesla GPUs, for each node. Every node has a fast on-chip network, such as HyperTransport, that allows for non-coherent data transfers as well as a standard off-chip network, such as 10 or 40 Gbps Ethernet. We assume that a standard switched topology is used with 24 - 32 nodes connected by a single Ethernet switch. In addition, data transfers to GPU memory are performed using HT posted writes exclusively and any data notifications, such as communicating whether the transfer finished, are handled by software protocols, as needed.

The data warehousing application in this model assumes that data tables are stored in host memory and can be either segregated to one table per node or striped across multiple nodes. A query application runs on one node, presumably close to the node containing a high-end GPU, and it initiates the transfer of input data for the query using GAS put/get commands to access remote memory.

In addition to these network characteristics, we also propose two changes to the handling of data at the receiving node. To preserve the host memory (DRAM) to be used for additional in-core database space with our application, we propose that each "accelerator" node will participate in peer-to-peer communication with an enhanced NIC that implements our converged fabric specification (i.e., HToE). This allows the NIC to copy data directly from its buffers to the GPU's memory without needing to pin pages in the host and without needing to interact with the operating system (except to set up the transfer). This model is very similar to that proposed by NVIDIA's GPUDirect 2.0 [25] except that the peers are a GPU and a NIC, instead of two GPUs. To further optimize the movement of data for our data warehousing application, we propose that an asynchronous streaming model [1] be used so that the GPU can execute the kernel using input data as it becomes available in the GPU's global memory. Since the output of the queries for our particular application tend to be small, this allows computation and data movement to overlap, especially for queries which are reasonably trivial to execute (e.g., select on a large data set).

A four node example of the complete system model is shown in Figure 2. Note that this figure presents two alternate interconnects connected to the HTEA, one for HyperTransport which connects directly to a northbridge based on the AMD Opteron, and one that routes data through an I/O Hub (IOH) that supports the PCI Express protocol. Future adapters could feasibly support either the HTX motherboard slot or a standard PCI Express slot by performing the needed HT to PCIe transformation in the HTEA. However, this work only addresses the former configuration (HTX-based).

As part of this system model, we are also interested in specific bottlenecks that result from sharing high-end GPUs between multiple nodes. A simple diagram of bottlenecks for our system model is shown in Figure 3. Several possible bottlenecks include 1) the link bandwidth of the Ethernet link 2) the incoming processing speed of the adapter and the size of the buffers in the HTEA, 3) the return rate of credits to "sending" nodes and 4) the transfer bandwidth of PCI Express to the GPU on the "accelerator" node. The total rate of data flowing through each of these links must not exceed the size of the GPU's global memory and 5) the rate of data consumption by the GPU once a query kernel is launched. The experiments in the next section use measurements from current implementations on GPU hardware for 4) and 5) and

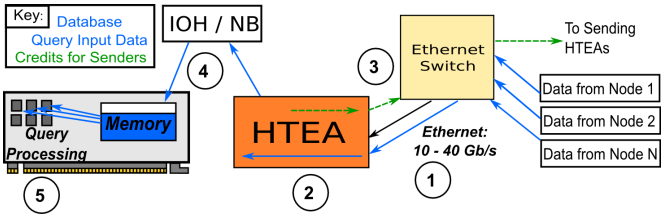

Figure 3. Potential Bottlenecks in Shared Accelerator Cloud System Model

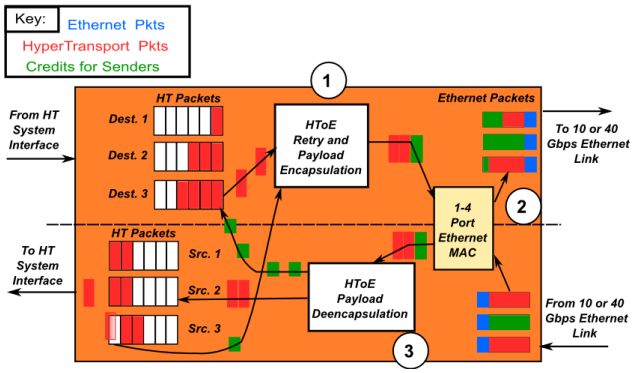

Figure 4. HTEA bottlenecks

a combination of published numbers and detailed simulation numbers to investigate bandwidth- and latency-limiting factors for accelerator clouds.

A. Application Workload

The data warehousing application that is addressed in our system model and the associated system model is based on our implementation of the TCP-H benchmark suite [29] using a database language, Datalog [14] and NVIDIA's CUDA programming language to run queries on a GPU as part of our single-node Red Fox infrastructure [34]. Two TCP-H queries, Query 1 and Query 21, are used to simulate workloads from a large in-core data warehousing application where the database table is striped across multiple nodes. Traditional query processing on the GPU requires that the entire input dataset be copied into the on-board memory of the GPU, but we assume that asynchronous processing [1] can be used to handle processing of a stream of input data with the same query. The network adapter does overlapped "read" or "get" operations to transfer the database from remote DRAM to the local accelerators.

Query 1 (Q1) performs a query based on one large table named "lineitem" that can be very large, but the query operation is relatively simple. Query 21 (Q21) performs a more complex query across multiple input tables but where "lineitem" is the largest table and could potentially be split across multiple nodes.

TCP-H represents a type of application that has a large input data set but typically a relatively small resultant data set, so we are focused on the input data set rather than the small resultant data set. Additional timing information for both queries is presented in Section VI-A.

B. HTEA Implementation

The HToE specification does not specify the implementation details for an HTEA. Therefore we propose the following design for supporting data warehousing application traffic: 1) Both the outgoing and incoming paths for the HTEA have a queuing module that buffers packets from the on-package HT system interface (outgoing) or the Ethernet link (incoming). Packets are queued in FIFO order, and packets are dequeued on a per-VL basis using a round-robin algorithm as credits are available for each specific destination. 2) HT Packets and credits are packed into Ethernet frames using a block-based algorithm that collects P packets before a payload is sent to be encapsulated. Multiple payloads can be constructed and buffered at the same time, but only M payloads can be encapsulated for transmission, where M is the number of Ethernet MACs and TX queues that are available to the HTEA. 3) Buffer sizing in the HTEA is based on recent network adapters with on-chip buffering as opposed to using DRAM-based buffering of data. We use 64 MB split equally between the outgoing and incoming path to buffer all HT packets.

Figure 4 shows a simplified diagram of the HTEA along with the following potential bottlenecks that can affect adapter throughput and overall performance: 1) Outgoing HT packets must receive credits (via NOPs) to be dequeued, and each group of HT packets must pass through an HToE encapsulation (header and CRC generation) module in FCFS order. 2) Outgoing packets must be encapsulated by an Ethernet MAC in FCFS order, and frames that contain NOP credits may be queued with other outgoing traffic. 3) Incoming packets must be deencapsulated both by the Ethernet MAC and the HToE deencapsulation module before HT packets can be sent to the local HT system interface and before credits can be processed for packets on the outgoing path. This last scenario is most relevant to accelerator clouds since often there are a large number of senders transmitting data to a single receiving node as shown in Figure 2.

VI. Experimental Setup

The simulations investigating the design of an HTEA are built on top of the NS-3 event-driven network simulator. NS-3 provides support for building Ethernet-like topologies and also contains a basic switch model. This study does not incorporate a DRAM simulator since it assumes a "sending" node can fully saturate the outgoing HT system interface with data from DRAM.

Our simulations use 2, 4, and 8 nodes, with one "accelerator" node in each simulation and a 1 GB data set (about 16 million HT packets, each with a payload of 64 B) striped over the remaining nodes. For example, this means that the 2 node case has the entire 1 GB data set on one node while the 8 node case spreads the data set across 7 nodes equally. This setup allows us to test the full capabilities of the network adapter and also to compare optimizations discussed in Section VI-B.

Module	Latency (ns)
HToE mapping, queuing	192
HToE credits, retry, encap	244
Ethernet MAC (out)	122
Eth switching and link	244
Ethernet MAC (in)	298
HToE deencapsulation	222
HToE queuing (in)	156
Total	**1480**

TABLE I. Estimated latency for HToE, one HT pkt

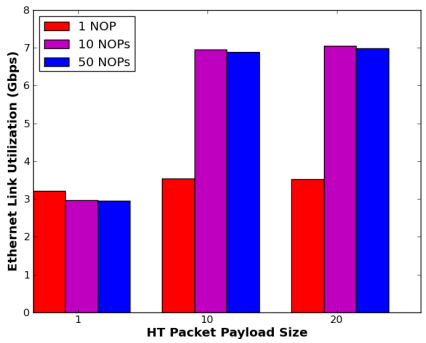

Figure 5. Link Utilization vs. HT Payload Size

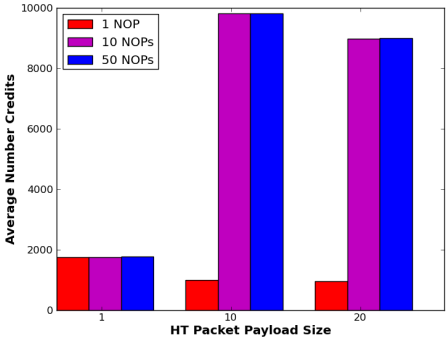

Figure 6. Average Number HT Credits vs. HT Payload Size

A. Simulation Timing

Timing models for each stage and critical path element in the HTEA were derived from previous FPGA prototypes [37] and the hardware implementation developed by the EXTOLL project, which is estimated to run at 300 MHz with a critical path latency in the low hundreds of nanoseconds [22]. Packet generation rates for reading each piece of a database table from DRAM and sending it to the HTEA are based on a 3 GHz processor and the maximum bandwidth of a HT 3.1 link with 16-bit lanes, 12.8 GB/s [6]. As mentioned previously, this simulation assumes a direct path from the HTEA to the destination accelerator via a network like PCIe (bypassing OS memory with the use of a peer-to-peer transfer through the IOH).

Timing for the Ethernet MAC is based on results from EB Engineering [12] and Ethernet switching times are based on the best-case latency for 10 Gbps switches, 200 ns [17].

The timing numbers for the PCIe transfer time and query computation time on the GPU are calculated from running TCP-H queries number 1 and 21 on a single node using an NVIDIA C2070 GPU with varying amounts of input data. Queries are written in Datalog and then the Red Fox framework [34] uses a dynamic compiler to convert each query to NVIDIA's PTX representation, which can then be run on the GPU. Transfer and computation time is extrapolated from these results based on a linear model that was built from the measured results. The PCIe transfer time for Query 1 with 1 GB of data was 0.175347 seconds while the transfer time for Query 21 was 0.331495 seconds. Query 1 took 0.002844 seconds to run on the GPU while Query 21 took 0.069619 seconds.

B. Tested Optimizations

As Figure 4 showed, the HTEA adapter design has many constraints that can affect the performance of certain applications, especially as in our data warehousing application where many large chunks of a database are copied to one receiver node. In an effort to improve the performance of the network adapter we test two specific optimizations and characterize their effects on overall throughput.

The first optimization deals with how long credits (HT NOPs) are buffered. While a single credit can be encapsulated in a single Ethernet frame (lowest latency), we varied the number of NOPs that were buffered before encapsulating them in an Ethernet frame to determine what effect this had on available credits for senders. The second optimization focuses on addressing the bottlenecks in the encapsulation and deencapsulation process with wider pipelines for the outgoing and incoming HToE modules as well as the use of dual-port Ethernet MACs, such as Intel's 82599 10 Gbps Ethernet MAC, which supports two full-duplex 10 Gbps ports [16]. This particular MAC is currently in use in 4 port Ethernet adapters, and this optimization assumes that each TX/RX port is either connected directly to the switch (requiring 4 cables) or a 40 Gbps link is used to multiplex packets from 4 virtual TX/RX queues in the adapter. Also note that each VL must use the same ingress or egress port (VL number modulo P ports) to satisfy HT packet ordering requirements.

VII. Results

Table I shows the combined timing for one HT packet passing through a sending and receiving HTEA based on our simulations and the Ethernet MAC timing from [12]. The end-to-end latency is reasonably low for an encapsulation-based adapter with an end-to-end latency of about 1.5 μs. It should also be noted that the most timing-intensive functions are on the receiving path. This delay is due to the need for comparing headers and validating checksums on each received Ethernet frame and HToE payload.

In Figure 5, the link utilization is shown for one sending HTEA and one receiving HTEA with varying HT packet payload sizes: 1 to 20 data packets in an Ethernet frame and 1 to 50 NOP packets in a frame. The low utilization for the

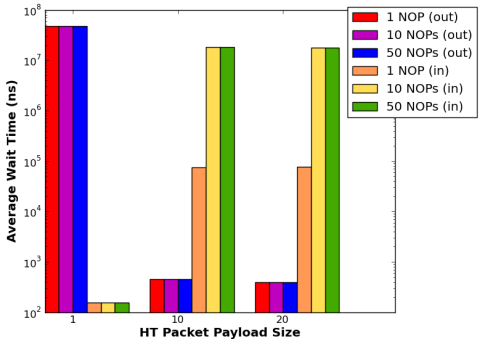

Figure 7. Average HTEA Delays vs. HT Payload Size

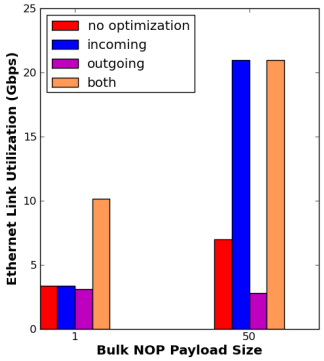

Figure 8. Link Utilization vs. NOP Payload Size for 4 Nodes

base case (1 HT packet) and the other 1 NOP cases result from the same phenomenon. In both cases, HT packets are encapsulated and sent as fast as possible. When the size of the sender's HT packet payload exceeds the size of bulk credit acknowledgments from the receiver, the sender quickly runs out of credits.

As the size of the bulk credit acknowledgments and the size of the HT data payload increases, the average number of available credits for the sending HTEA increases, as shown in Figure 6. Both 10 and 50 NOPs in an acknowledgment payload result in the same average number of credits available at the sender. This indicates that the encapsulation process for HT data packets takes slightly longer than processing received credits and that the bulk NOP size doesn't need to be dramatically larger than HT data payload size. However, it should be noted that use of the bulk NOPs decreases link bandwidth slightly (Figure 5) because it reduces the total number of Ethernet frames (and associated overhead) that are sent in the same amount of time.

Figure 7 demonstrates that using smaller or larger payloads can also affect delay within the network adapter. The outgoing path represents the delay that results when an HT packet has a credit but must wait to be encapsulated due to the FIFO nature of an unoptimized HTEA. Similarly, the incoming path delay results from the latency required to deencapsulate Ethernet frames and HToE payloads. Delay for outgoing HT data payloads ranges from a low of 390.84 ns (20 HT packet payload) to 48.26 ms (1 HT packet payload), and delay for incoming payloads ranges from 154.74 ns (1 HT packet

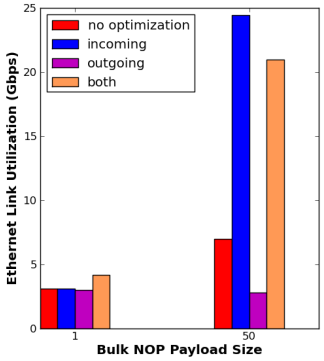

Figure 9. Link Utilization vs. NOP Payload Size for 8 Nodes

payload) to 17.90 ms (20 HT packet payload). Small packet payloads have higher delay on the outgoing path due to head-of-line blocking, but they also are processed faster at the receiver.

Figures 8 and 9 show link utilization with the use of our dual-MAC (4 parallel processing stages) pipeline optimizations for either the incoming path (deencapsulation), outgoing path (encapsulation), or for both paths. These graphs for 20 HT packet payloads and varying bulk NOP payload sizes illustrate two important concepts: 1) Pipeline width optimizations can't make up for a mismatch in the speed of NOPs returning from the receiver to senders, even though the wider outgoing pipeline reduces the average wait time for encapsulation of NOPs from 42.29 ms to 1774 ns. 2) Deencapsulation pipelining affects performance much more than encapsulation pipelining, potentially boosting bandwidth up from 3 Gbps to 24.45 Gbps. This improvement results from optimizing deencapsulation, which is a much more time-intensive operation (Table I). In some cases, aggressive encapsulation pipelines can actually reduce bandwidth as shown by the unoptimized, 50 NOP case (6.99 Gbps) and the outgoing optimization, 50 NOP case (2.80 Gbps). This reduction in bandwidth seems to be due to an unusual case where the optimized sender fills the receiving HTEA's buffers and must wait until the receiver processes some of the incoming Ethernet frames before it can accept more frames from the sender.

In summary, the use of wider pipelines can definitely provide much better performance for the HTEA but only in conjunction with the use of bulk NOPs. Bulk NOP acknowledgments allow the rate of HT packets from senders to increase (assisted in part by wider pipelines) while making sure that the overhead of processing credits does not delay the encapsulation process.

A. Evaluating System-wide Bottlenecks

Finally, we check our HTEA design and optimizations against the potential bottlenecks described in Figure 3. As seen in Figure 10, our maximum link bandwidth was about 24 Gbps (labeled as 8Nd) over a 40 Ethernet Gbps link, which means that the "optimized" implementation would not be limited by the available Ethernet bandwidth. At the receiving HTEA, the incoming bandwidth onto the local HT system interface maxed

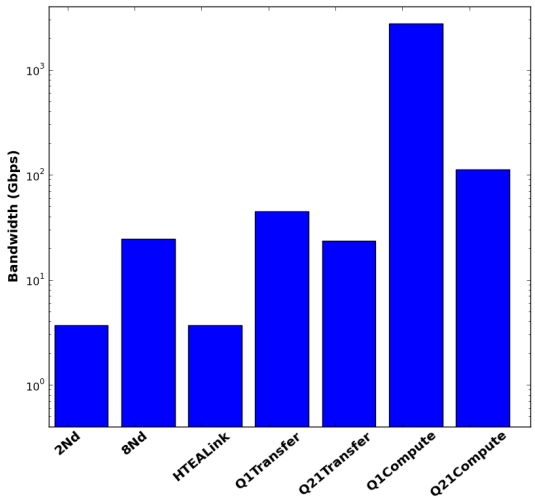

Figure 10. Relative Bandwidths of HTEA, Networks, and GPU

out at 3.69 Gbps, meaning that the incoming packet stream is more limited by the HTEA than by the HT system interface. Finally, the query processing bandwidth of the TCP-H Q1 and Q21 queries was much higher (2,747 Gbps and 112 Gbps) than the PCIe transfer bandwidth (44.55 and 23.57 Gbps), so we can safely assume that in our system model the only potential limitation is at the network adapter. However, as the discussed optimizations prove, there is still a great deal of potential performance in the development of future converged commodity fabrics.

VIII. Conclusions

This work proposes the use of a commodity, converged fabric for incorporation into future data centers that act as memory and accelerator clouds for data warehousing applications. Using experiments run on GPUs and a detailed simulator, we have demonstrated the design of a HToE-based network adapter to support these types of applications. The results showed the importance of using bulk credit acknowledgments to ensure a steady stream of credits as well as the usefulness of having a wider receiving pipeline in the HTEA to help mitigate long deencapsulation times. Future work will focus on extending these optimizations to further reduce delays in the HTEA.

IX. Acknowledgments

We greatly appreciate the comments of the anonymous reviewers as well as the helpful suggestions and insights from the following members of the HyperTransport Consortium: Emilio Billi, Mario Cavalli, Brian Holden, and Paul Miranda.

References

[1] M. Bauer, H. Cook, and B. Khailany. CudaDMA: Optimizing GPU memory bandwidth via warp specialization. SuperComputing 2011.
[2] E. Billi, J. Young, and B. Holden. HyperTransport over InfiniBand specification, 1.0. 2011. http://www.hypertransport.org.
[3] D. Bonachea. GASNet specification, v1.1. Technical report, 2002.
[4] P. Charles et al. X10: an object-oriented approach to non-uniform cluster computing. OOPSLA 2005.
[5] D. Cohen et al. Remote Direct Memory Access over the Converged Enhanced Ethernet fabric: Evaluating the options. In *Hot Interconnects*, 2009.
[6] H. Consortium. Hypertransport specification, 3.10. 2008. http://www.hypertransport.org.
[7] H. Consortium. HyperShare technology overview. 2011. http://www.hypertransport.org.
[8] Cuda 4.0 toolkit overview. 2011. http://developer.download.nvidia.com/compute/cuda/4_0/CUDA_Toolkit_4.0_Overview.pdf.
[9] D. Dalessandro, P. Wyckoff, and G. Montry. Initial performance evaluation of the NetEffect 10 Gigabit iWARP adapter. In *Cluster Computing*, 2006.
[10] J. Duato et al. rCUDA: Reducing the number of GPU-based accelerators in high performance clusters. In *HPCS*, July 2010.
[11] J. Duato et al. Scalable computing: Why and how (white paper). 2010. http://www.hypertransport.org.
[12] Xilinx FPGA simulation with 10 Gbps Ethernet MAC - obtained from tests run by Emilio Billi of EBE. 2012. http://www.emiliobilli.com/.
[13] H. Fröning et al. A case for FPGA based accelerated communication. In *ICN*, 2010.
[14] S. S. Huang, T. Green, and B. T. Loo. Datalog and emerging applications: an interactive tutorial. 2011. http://www.logicblox.com/presentations/norcaldb-2011-aref.pdf.
[15] IEEE 802.1 Working Group. IEEE 802.1Qbb standards page. http://www.ieee802.org/1/pages/802.1bb.html.
[16] Intel 8255 10 Gigabit Ethernet controller datasheet. http://www.intel.com/content/www/us/en/ethernet-controllers/82599-10-gbe-controller-datasheet.html.
[17] Hardware specification for Intel FM2224 10 GE 24 port switch. 2012. http://ark.intel.com/products/64419/Intel-Ethernet-Switch-FM2224.
[18] M. Ko et al. A case for Convergence Enhanced Ethernet: Requirements and applications. In *ICC*, May 2008.
[19] ConnectX-3 VPI product brief, 2012. http://www.mellanox.com/related-docs/prod_silicon/ConnectX3_VPI_Silicon.pdf.
[20] H. Montaner et al. MEMSCALE: A scalable environment for databases. In *Proceedings of the 2011 IEEE International Conference on High Performance Computing and Communications*, 2011.
[21] Numaconnect (white paper). http:www.//numascale.com/numa_WP_NumaConnect.html.
[22] M. Nüssle. High performance interconnect leveraging HyperTransport - EXTOLL. 2010. http://www.extoll.de/images/extoll_slides.pdf.
[23] J. Ousterhout et al. The case for RAMClouds: Scalable high-performance storage entirely in DRAM. 2011.
[24] D. K. Panda. Acceleration for Big Data, Hadoop and Memcached (presentation). *HPC Advisory Council Workshop*, Mar. 2012.
[25] T. C. Schroeder. Peer-to-Peer and Unified Virtual Addressing. *Cuda Webinar*, 2011.
[26] G. Shainer et al. The development of Mellanox/NVIDIA GPUDirect over InfiniBand-a new model for GPU to GPU communications. *Journal of Computer Science - Research and Development*, June 2011.
[27] H. Shan et al. A preliminary evaluation of the hardware acceleration of the Cray Gemini interconnect for PGAS languages and comparison with MPI. In *PMBS*, 2011.
[28] J. Suzuki et al. Adaptive memory system over Ethernet. In *Proceedings of the 2nd USENIX conference on Hot topics in storage and file systems*, HotStorage'10, Berkeley, CA, USA, 2010.
[29] TCP-H benchmark specification. http://www.tpc.org/tpch/.
[30] UPC language specifications, v1.2. Technical report, 2005.
[31] R. vandeVaart. NVIDIA and OpenMPI (presentation). *Open MPI State of the Union Community Meeting SC11*, 2011. www.open-mpi.org/papers/sc-2011/Open-MPI-SC11-BOF-1up.pdf.
[32] C. Vaughan et al. Investigating the impact of the Cielo Cray XE6 architecture on scientific application codes. In *IPDPSW*, May 2011.
[33] H. Wang et al. MVAPICH2-GPU: optimized GPU to GPU communication for InfiniBand clusters. *Journal of Computer Science - Research and Development*, 26(3-4):257266, June 2011.
[34] H. Wu et al. Optimizing data warehousing applications for GPUs using kernel fusion/fission. In *Multicore and GPU Programming Models, Languages and Compilers Workshop, IPDPS-PLC*, May 2012.
[35] J. Young et al. HyperTransport over Ethernet - a scalable, commodity standard for resource sharing in the data center. *WHTRA*, 2011.
[36] J. Young and B. Holden. HyperTransport over Ethernet specification, 1.0. 2010. http://www.hypertransport.org.
[37] J. Young and S. Yalamanchili. Dynamic Partitioned Global Address Spaces for power efficient DRAM virtualization. *International Conference on Green Computing*, 2010.