A 10G NetFPGA Prototype for In-Network Aggregation

Vincent T. Lee, Jacob Nelson, Mark Oskin, Luis Ceze
Department of Computer Science and Engineering
University of Washington
(vlee2, nelson, oskin, luisceze)@cs.washington.edu

Abstract

An emerging class of distributed applications such as graph processing and key-value store require fine grained non-uniform memory access and communication, which commodity networking infrastructure is ill-equipped to handle. By batching messages on the end host we can increase bandwidth utilization at the cost of latency and memory. However, by batching messages in the network instead of exclusively at the end hosts, it is possible to reduce these memory and latency penalties. We introduce a 10G NetFPGA prototype that performs in-network message aggregation which for commodity systems can offer increased throughput at minimal latency and memory increases.

1 Introduction

Data center networks face unprecedented challenges as network architects struggle to scale bisection bandwidth for emerging applications [1], and end host systems struggle to maximize the gains of faster network interconnects [5]. Link level bandwidth performance has seen great improvements in recent years with 10, 40, and even 100 Gigabit transceiver technology. These increases in network interconnect performance naturally result in better bisection bandwidth. At the same time, increased network bandwidth capacity has created new challenges for commodity interconnects and network stacks to maximize impact of these gains. With the end of silicon scaling, compute and memory budgets for standard networking stacks and systems have struggled to operate at line rate.

Commodity systems and network stacks have been traditionally optimized to move large packets and accelerate TCP and UDP offloads, not small packets and custom workloads. For applications which require fine grained non-uniform memory access (NUMA) patterns such as distributed graph processing, and key-value store, commodity systems fare poorly. One solution is to batch small messages together on the end hosts of the network into larger messages which achieves higher bandwidth utilization and overall throughput [4]. Unfortunately, end host aggregation generates its own set of challenges; batching messages necessarily requires both a latency increase and larger memory footprint. In the worst case, when each end host allocates an aggregation buffer to communicate with every other end host in the system, the memory footprint scales quadratically.

Instead, we propose an alternate solution, in-network aggregation which offloads the routing and reaggregation into the network fabric similar to the techniques proposed by L. Mai et al. [3]. Under this approach, messages are batched regardless of destination at the end host for each outgoing network interface in the system; each hop in the network then unbundles, routes, and reaggregates messages bound for the same destination interface. By offloading aggregation into the network, we can reduce the latency, and memory footprint on end hosts since the buffer count now scales with the number of network ports in the system. In the worst case, a naive all-to-all network topology with full bisection bandwidth incurs a quadratic memory footprint, but for other network topologies such as a fat tree, in-network aggregation can achieve sub-quadratic memory scaling.

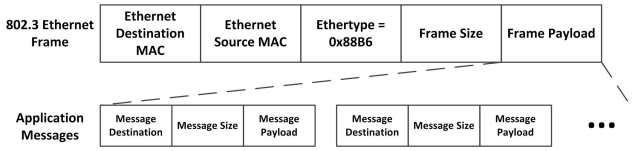

Figure 1: Ethernet Frame Format and Application Message Format

2 Hardware 10G NetFPGA Prototype

We have built a prototype router on a 10G NetFPGA to evaluate the benefits of in-network aggregation [2]. The 10G NetFPGA platform provides 4 x 10 Gigabit Ethernet SFP+ interfaces which are operated by AEL2005 PHYs, and two Samtec expansion connectors which are operated by 20 GTX dual transceivers. The PHYs are connected to a Xilinx Virtex-5 TX240T and communicate to XGMAC cores on the FPGA in full duplex mode. To minimize packet overhead, our design operates directly on raw ethernet frames carrying a custom header. Each ethernet frame payload contains a batch of application level messages which have a message destination field, message size field, and message payload (Figure 1).

We bootstrapped our prototype design from the example reference router project in the NetFPGA-10G-live Github repository. Our design replaced most of the data plane routing logic in the reference design with our aggregation core but reused the XGMAC core peripherals and board configuration files. Each data bus in our design is 64 bits wide and operates at 156.25 MHz offering 10 Gbps of throughput and full bisection bandwidth. To simplify the packet processing, we align payload data to 8 byte boundaries.

Our router prototype follows a virtual output queue switch architecture supplemented with our aggregator and deaggregator cores (Figure 2). Each receiving data bus from the XGMAC cores passes frames into a Deaggregator core; frames are then stripped of their headers and have their payloads split into applications messages. Application messages are then routed by a programmable routing table into the Virtual Output Queues based on their destination field. An Aggregator Queue for each outbound XGMAC bus then collects the application messages from the Virtual Output Queues and reaggregates them. A Transmission Module then generates a ethernet header and initiates a transmission when the size

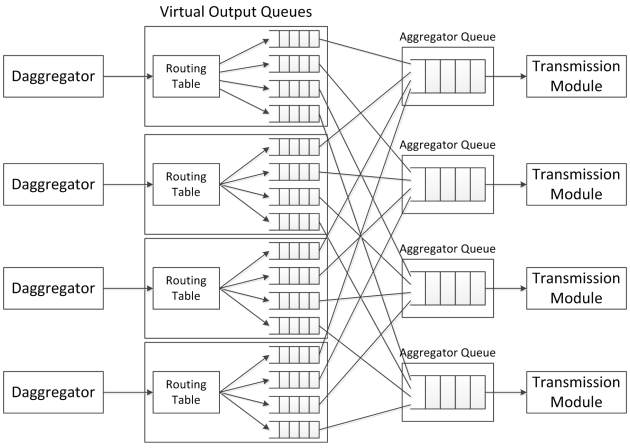

Figure 2: Router Prototype Architecture

of aggregated data in the Aggregator Queue reaches a programmable threshold or a timeout occurs.

A Microblaze soft processor handles any control signals required to initialize the system and any user reconfiguration. Our prototype supports reconfiguring the routing tables, aggregation thresholds, timeout period, and egress MAC address. All reconfiguration is communicated over a serial port to a Microblaze soft processor which issues reconfiguration commands to our design.

Our design took 6 months to implement, test, and validate by a single student. A notable portion of the development time was devoted to set up costs and negotiating project infrastructure. We have also built an extended design which also supports communication over the NetFPGA's two Samtec connectors using Aurora transceivers. We envision these connectors to be used as inter-board links to overcome the four port limitation of a single NetFPGA board, effectively extending our in-network aggregator prototype to 12 x 10G interfaces.

3 Challenges Operating at 10G

Operating at 10 Gigabit line rate is a challenge for standard commodity interfaces and Linux networking stacks. Using standard Linux sockets today for a 10 Gigabit Ethernet interface, we can achieve transmission rates shown in Figure 3 and find for small packets we are limited by packet injection rate. Even at these limited transmission rates, we find that we cannot receive packets from the network interface card (NIC) using standard Linux sockets without using higher level protocol NIC accelerators such as UDP offloads. To operate at higher line rates, we are looking at custom networking stacks and drivers such as PF_RING which can operate in kernel bypass mode to eliminate kernel overhead to evaluate our design.

4 Project Status and Future Work

We have completed a preliminary simulation using the Omnet++ network simulator to validate the potential benefits of in-network aggregation. In order to improve the fidelity of the simulation, we have designed and prototyped a router that can perform in-network aggregation. To evaluate our prototype, we plan to benchmark Gigaupdates Per Second (GUPS) throughput which randomly issues updates across nodes. We chose the GUPS benchmark because it approximates the same communication patterns as more complex applications

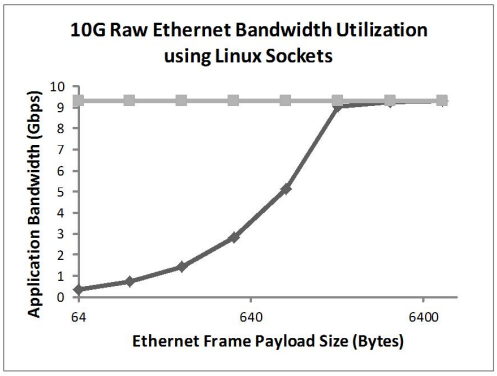

Figure 3: Maximum network transmission bandwidth for Linux raw 10G Ethernet sockets for various frame payload sizes with end host aggregation (black) and in-network aggregation (grey)

such as distributed key-value store, and graph processing. As a comparison point, a standard commodity switch will be used to measure end host aggregation performance baselines with and without aggregation enabled. We will then evaluate our design with and without aggregation enabled on our prototype in place of the commodity switch. To evaluate the design at scale, we will use our prototype results to build a simulation model for larger network topologies, and different levels of oversubscription.

References

[1] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. Vl2: A scalable and flexible data center network. *SIGCOMM Comput. Commun. Rev.*, 39(4):51–62, August 2009.

[2] John W. Lockwood, Nick McKeown, Greg Watson, Glen Gibb, Paul Hartke, Jad Naous, Ramanan Raghuraman, and Jianying Luo. Netfpga–an open platform for gigabit-rate network switching and routing. In *Proceedings of the 2007 IEEE International Conference on Microelectronic Systems Education*, MSE '07, pages 160–161, Washington, DC, USA, 2007. IEEE Computer Society.

[3] Luo Mai, Lukas Rupprecht, Abdul Alim, Paolo Costa, Matteo Migliavacca, Peter Pietzuch, and Alexander L. Wolf. Netagg: Using middleboxes for application-specific on-path aggregation in data centres. In *Proceedings of the 10th ACM International on Conference on Emerging Networking Experiments and Technologies*, CoNEXT '14, pages 249–262, New York, NY, USA, 2014. ACM.

[4] Jacob Nelson, Brandon Holt, Brandon Myers, Preston Briggs, Luis Ceze, Simon Kahan, and Mark Oskin. Grappa: A latency-tolerant runtime for large-scale irregular applications. Technical Report UW-CSE-14-02-01, University of Washington, 2 2014.

[5] Simon Peter, Jialin Li, Irene Zhang, Dan R. K. Ports, Doug Woos, Arvind Krishnamurthy, Thomas Anderson, and Timothy Roscoe. Arrakis: The operating system is the control plane. In *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, OSDI'14, pages 1–16, Berkeley, CA, USA, 2014. USENIX Association.