Software-Addressable Optical Accelerators for Data-Intensive Applications in Cluster-Computing Platforms

P. Samadi[(1)], V. Gupta[(2)], B. Birand[(2)], H. Wang[(1)], R. Jensen[(3)], G. Zussman[(2)], K. Bergman[(1)]

[(1)] Lightwave Research Laboratories, Department of EE, Columbia University, New York, NY, USA, ps2805@columbia.edu

[(2)] Wireless and Mobile Networking Lab, Department of EE, Columbia University, New York, NY, USA

[(3)] Polatis, Burlington Road, Bedford, MA, USA

Abstract *We present a control plane architecture to enable software-addressable optical acceleration from the application layer. The architecture is experimentally examined on a cluster-computing test-bed by enabling physical layer optical multicasting on-demand for the application layer to achieve non-blocking performance.*

Introduction

With the growing interest in Big Data analysis and cloud computing, new challenges and opportunities arise in the design and control of the interconnection networks that supports these cluster-computing platforms. Software-defined networking (SDN) paradigms that provide central network management and dynamic configuration of the architecture are considered as potential future directions to address these challenges. Furthermore, optical interconnect technologies that can provide high bandwidths and lower energy consumption are becoming a part of the future interconnection networks efforts.

Hybrid network architectures that combine Ethernet and optical circuit switching has been proposed for point-to-point optical connectivity in data center networks [1, 2]. In these architectures, optical space switches (OSS) are used to connect the Top-of-Rack (ToR) switches and a SDN-based control plane to manage the traffic between the two networks. The optical links connected by the OSS are primarily utilized for large packet transmission (Elephant flows) since the 25ms switching speed does not allow for switching individual or small groups of packets (Mouse flows). Data centers support a wide variety of applications that generate flows of all sizes, thus further investigations are required to improve such hybrid architectures.

Big data analysis generally involves parallel data processing techniques such as Hadoop, which uses a distributed file system and a MapReduce [3] type algorithm for data analysis. These operations require large data transfers with richer traffic patterns (Multi/In-cast) between the clusters (i.e. ToR switches) that introduce heavy traffic to the interconnection network. The shuffle stage of MapReduce and the data storage in the distributed file system require Incast (many-to-one) traffic delivery. Multicast (one-to-many) traffic delivery is also required in various applications including data replication (for improving the reliability of distributed file systems), parallel database join operation, data dissemination in virtual machine (VM) provisioning and in-cluster software updating, as well as data analysis in broadcast phase of Orchestra [4] (a control architecture to manage intra- and inter-transfer activities).

In current platforms, these patterns are managed either by sequence of unicast transmissions or more advanced software solutions such as peer-to-peer method. These methods are naturally inefficient since multiple copies of the same data are transferred on the network. For instance, Orchestra system transmits 12 copies of the same data. Physical layer transparent data replication using passive optical splitters by setting up a multicast tree between the source and the destinations [5] can provide a significantly more efficient multicasting operation since only one copy of data is sent.

Optical modules can clearly provide functionalities beyond point-to-point data transmission. However, due to the complexities in configuring optical devices and the circuit-based nature of optical switching, integration of optics with current network architectures is challenging. Cross-layer architectures can potentially overcome these complexities and provide optical functionalities more seamlessly to the application layer.

In this work, we present a control plane architecture for a hybrid interconnection networks that can accelerate data delivery for data-intensive applications. The control plane architecture is experimentally examined on our 10G Hybrid Cluster-Computing Test-bed with a demonstration of physical layer optical multicasting. The implemented control plane employs a messaging system, a resource allocation algorithm, and APIs to control the optical resources and manage the network.

Architecture

Fig. 1 demonstrates the network layers diagram consisting of the Application, Control Plane and Data Plane layers. The Control Plane has a central network controller that manages the control plane and includes a resource allocation algorithm. The network controller communicates with the data plane layer via southbound APIs that are Floodlight for the OpenFlow switches, OSS API, that is a python-based API developed in-house for the OSS and other APIs for configuration of active optical

resources. For the northbound API of the control plane, we have developed a pub/sub messaging system using Redis [6]. Byte size messages are transmitted on the Ethernet network amongst hosts and controllers. Compared to the REST API, typically used for the northbound API, this system has much lower latency and only adds Byte size traffic. We measured the latency by sending 200 messages of 20 bytes among hosts and measured average latency of 300 µs.

Requests for the optical resources are generated directly by the application layer. One approach in providing the optical resource to the application is an explicit request to the controller, from the host running the application. However, since our architecture is mainly designed for data-intensive applications that run parallel processes on several clusters, we assume that a central application controller provides the traffic matrix to the network controller. We believe future applications will have a central application controller (scheduler) that manages different processes running on the clusters. This approach is also coherent with the SDN concept, which performs a central management of the network architecture. The performance of data-intensive applications running on cluster-computing platforms can significantly improve through an intelligent management of the network architecture by combining the global knowledge of the interconnection network and the traffic matrix of the applications.

The resource allocation algorithm runs on the network controller. Its objective is to schedule the optical splitters across the multicast requests, by maximizing the obtained throughput. It also supports the cascading of two splitters if a single splitter cannot serve the request. We model this problem as an Integer Program (IP). We denote R the number of multicast requests and H the number of available hosts. Each request i is associated with a transmission size s_i and a number p_i of required optical splitters. The binary variable a_i is 1, if request i is active as a result of the computation. The constant m_{ij} is 1 if the i^{th} request requires host j as a sender or a receiver. The number of available splitters is denoted by S and we assume that all splitters have an identical number of ports (our model can trivially be extended to support different number of ports). Finally, the variable x_{ij} is 1 if request i is allocated to host j. The problem is formulated as following:

$$\max \quad \sum_i a_i s_i \tag{1}$$

$$\sum_{i=1}^{R} x_{ij} m_{ij} \;\le\; 1 \qquad \forall j = 1 \rightleftharpoons H \tag{2}$$

$$a_i \le \sum_{j=1}^{H} x_{ij} m_{ij} \;\le\; 1 \quad \forall i = 1 \rightleftharpoons R \tag{3}$$

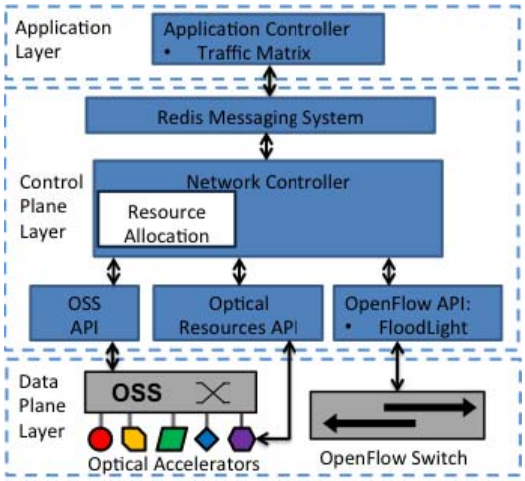

Fig. 1: Layer Diagram of the network architecture

$$\sum_i a_i p_i \;\le\; S \tag{4}$$

The objective (1) is to maximize the throughput defined as the size of the traffic offloaded via the optical network. A host can be the sender or the receiver of at most one request, which is modeled by (2). Constraint (3) guarantees that all source and destination hosts of active requests must be allocated to the request. Finally, the limit on the number of optical splitters is modeled by (4). We solve this IP using the GLPK solver that implements a branch-and-bound method to yield the optimal solution.

The general algorithm of the network controller is as following: The application controller submits the traffic matrix of the jobs that require optical resources to the network controller via the Redis messaging system. For multicasting, the traffic matrix includes the source, destinations and the size of the multicast job. The resource allocation algorithm schedules the jobs according to the IP described above. Then, the network controller generates the network configuration for the next job using the result of IP algorithm. The ToR switches are configured using Floodlight, and the OSS connections are applied using our API. Since we examine optical multicasting using passive optical splitters, the network controller first notifies the receivers using the Redis messaging system. Then, the Senders are notified to begin the transmission. Each receiver sends a message to the controller, notifying the completion of the job. Once the controller receives these messages from all the receivers for that job, it updates the traffic matrix and reruns the algorithm.

10G Hybrid Cluster Computing Test-bed:

Fig. 2 shows our test-bed consisting of 8 nodes, each with 10G network interface card (NIC), connected to a Pronto OpenFlow-enabled 10G Ethernet switch. The switch is partitioned into eight logical ToRs, with 10G-capable optical and Ethernet uplink/downlink ports. The 8 ToRs are aggregated in a hybrid architecture by a Juniper EX4500 10GbE Ethernet

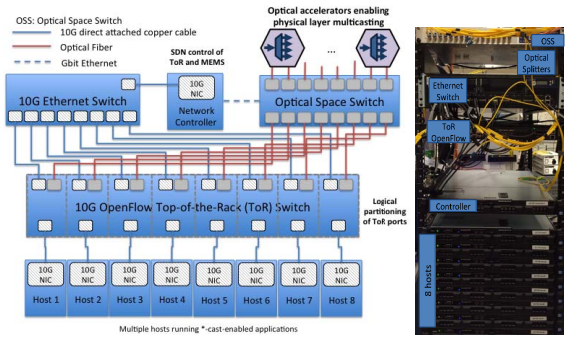

Fig. 2: The 10G hybrid cluster-computing test-bed using OSS as a substrate to connect optical accelerators

Switch and also a piezo-beam-steering based Polatis optical space switch that acts as a connectivity substrate of optical accelerators. The hosts are DELL servers with two Intel Xeon E5-2430 2.20GHz processors operating Scientific Linux 6.5.

We implement the sender and receivers at the end hosts through UDP multicast. The sender processes runs on several cores with each core transmitting a chunk of the file. The number of cores, sender-receiver buffers and UDP packet size are optimized for the best throughput and error performance.

For our experiment we generated a random traffic matrix with uniform distribution of 50 multicast jobs. The group sizes (number of receivers in multicast jobs) are from 2 to 6 and the file sizes are 1-5 GB. Two 1:4 optical splitters are connected to the OSS. The resource allocation algorithm is capable of cascading multiple splitters to generate groups larger than 4 receivers. The average latency of the control plane on the 50 jobs excluding the resource allocation algorithm is 10 ms.

We evaluated the performance of our architecture for multicasting in comparison with IP multicast over Ethernet since it is more efficient than peer-to-peer and sequence of unicasts. Two sets of 50 multicast jobs, with the maximum group size of 4 and 7 were generated. The former involves maximum half of the nodes and one splitter. The latter can involve the whole network and requires cascading of splitters. We measured the total completion time by optical multicast versus IP multicast on the Ethernet switch (Tab. 1). Our software-addressed optical multicast performs similar to the ideal fully non-blocking network. This means all the nodes in the multicast tree receive the data at the line rate that is not achievable in a real interconnection network due to over-subscription. In order to evaluate the performance improvement of our architecture in a more realistic scenario, we compared our architecture against IP multicast on a 1:10 over-subscribed Ethernet network. Due to equipment limitations, we were not able to over-subscribe the Ethernet network on our test-bed. However, we evaluated the performance of IP multicast on 1:10 over-subscribed network of 4 hosts working at 1Gbps and measured factor of 8 performance degradation [7]. We calculated the completion time for our 50-job traffic matrix on the 1:10 over-subscribed network and as Tab.1 shows, the software-addressed optical multicast significantly improves the completion time close to an order of magnitude. To summarize, our architecture provides line rate non-blocking multicast between the nodes that is not possible in practical interconnection networks. Additionally, it substantially decongests the over-subscribed Ethernet network by offloading the traffic to the optical network.

Conclusions and Future Works

We have presented a SDN-compatible control plane architecture for hybrid interconnection networks that enables software-addressable optical acceleration of data delivery, directly requested by the application layer. This architecture achieves non-blocking performance on the multicast tree. Our future work includes using nanosecond optical switches to accelerate Incast traffic delivery. These switches can be connected to the OSS similar to the optical splitters and configured by the control plane. Our proposed architecture can decongest the over-subscribed interconnection network of cluster-computing platforms running data-intensive applications such as Hadoop, that involve frequent large Incast/Multicast traffic delivery.

Tab. 1: Completion time of 50 multicast jobs (seconds) Proposed software-addressed optical multicast, IP multicast over a non-blocking and 1:10 over-subscribed

	Group Size	
	7	4
Proposed	226.179	187.348
Non-blocking	218.517	191.68
Over-subscribed	1748.136	1533.44

Acknowledgements

We would like to thank Juniper networks for their generous donations to our test-bed and also Polatis for providing the optical space switch. This work was supported in part by CIAN NSF ERC under grant EEC-0812072.

References

[1] N. Farrington et al. “Helios: A hybrid electrical/optical switch architecture for modular data centers,” in Proc. ACM SIGCOMM (2010).

[2] G. Wang et al., “c-Through: Part-time optics in data centers,” in Proc. ACM SIGCOMM (2010).

[3] J. Dean et al., “MapReduce: Simplified Data Processing on Large Clusters”, Google Inc., Communications Of The ACM, Vol. 51, No. 1, January 2008.

[4] M. Chowdhury et al., “Managing data transfers in computer clusters with orchestra,” in Proc. ACM SIGCOMM (2011).

[5] H. Wang et al., “Rethinking the physical layer of data center networks of the next decade: Using optics to enable efficient *-cast connectivity”, ACM Com. Comm. R., V. 43, No. 3, (2013).

[6] http://www.redis.io

[7] P. Samadi et al., "Accelerating Cast Traffic Delivery in Data Centers Leveraging Physical Layer Optics and SDN," 18th Int. Con. on Opt. Net. Des. and Mod. (2014).