Blast: Accelerating High-Performance Data Analytics Applications by Optical Multicast

Yiting Xia
Rice University

T. S. Eugene Ng
Rice University

Xiaoye Steven Sun
Rice University

***Abstract*—Multicast data dissemination is the performance bottleneck for high-performance data analytics applications in cluster computing, because terabytes of data need to be distributed routinely from a single data source to hundreds of computing servers. The state-of-the-art solutions for delivering these massive data sets all rely on application-layer overlays, which suffer from inherent performance limitations. This paper presents Blast, a system for accelerating data analytics applications by optical multicast. Blast leverages passive optical power splitting to duplicate data at line rate on a physical-layer broadcast medium separate from the packet-switched network core. We implement Blast on a small-scale hardware testbed. Multicast transmission can start 33ms after an application issues the request, resulting in a very small control overhead. We evaluate Blast's performance at the scale of thousands of servers through simulation. Using only a 10Gbps optical uplink per rack, Blast achieves upto 102× better performance than the state-of-the-art solutions even when they are used over a non-blocking core network with a 400Gbps uplink per rack.**

I. Introduction

In this era of big data, large-scale data-intensive applications are implemented on commodity cluster machines for data-parallel computations. One use case that draws particular attention recently is the high-performance data analytics applications, which involve a rich set of iterative machine learning algorithms for stock market predictions, disaster early warnings, disease diagnoses, product recommendations, user preference analyses, etc.

These iterative machine learning algorithms for massive data analytics usually run on cluster computing frameworks, MapReduce [13] being the most commonly used one. In either the Hadoop [2] or the Spark [27] MapReduce implementation, these algorithms observe the underlying communication patterns in Figure 1. A machine learning job may take hundreds or even thousands of iterations to converge. Each iteration starts from the multicast of a big data set of model parameters from the master node to all the worker nodes (from tens to hundreds depending on the cluster scale). The workers perform computations based on the given parameters. Sometimes, the computations require data exchange among the workers, thus creating a shuffle phase. Each worker updates the model parameters partially. These partial results are aggregated at the master node to form the global model parameters for the computations in the following iteration.

The multicast communication is a performance bottleneck considering the big fan-out, large data size, and great number of iterations. For instance, using the Latent Dirichlet Allocation algorithm for text mining [8] requires the word distribution of all the sampled topics to be multicasted iteratively, which can easily exceed 1GB of data per iteration and the learning usually takes over 1000 iterations, leading to terabytes of data to be multicasted. Other examples include the Logistic Regression algorithm for Twitter spam filtering and the Alternating Least Squares algorithm for Netflix movie rating prediction [12]. Both jobs take hundreds of iterations. In the multicast phase of each iteration, a data source distributes 300MB and 385MB data respectively, creating tens of gigabytes of multicast data. Multicast communications account for 30% and 45% of the job completion time respectively.

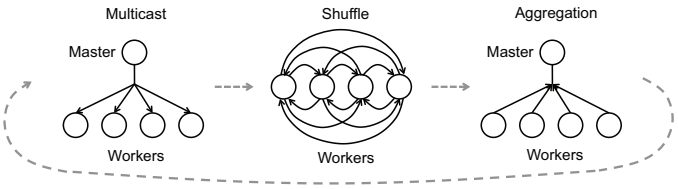

Fig. 1: Communication patterns in different phases of typical iterative machine learning jobs for massive data analytics

Therefore, high performance data analytics applications routinely need to distribute terabytes of data from a central data source to hundreds of servers for processing. The state-of-the-art solutions for transporting these massive data sets all rely on application-layer overlays. In Hadoop, multicast data distribution is through the Hadoop Distributed File System (HDFS) [2]. The multicast sender stores the data in HDFS and a multitude of receivers retrieve the data from a few data replicas, creating very high fan-out on the replicas. Spark attempts to improve the HDFS overlay by a BitTorrent style P2P overlay among the recipient nodes [27], but BitTorrent suffers from suboptimal multicast trees that render high link stress. A study shows that HDFS becomes the bottleneck when the receiver nodes exceed a certain number, roughly 40 to 50, and that BitTorrent actually has worse performance than HDFS in practice [5].

These solutions all suffer from the inherent performance limitations of application-layer overlays. However, not a single solution today uses in-network multicast. This begs the question: Why are decades of in-network multicast research and development not leveraged to address this problem? The reason lies within the network core. At 10Gbps or higher speeds, traditional CSMA/CD broadcast is not supported. Only full duplex point-to-point links are defined by the IEEE 802.3 standard at speeds higher than 10Gbps. Thus, to multicast data to N top-of-rack (ToR) switches at line rate, N or more

sending switch ports in the network core must be consumed, an expensive proposition when the network core is often oversubscribed by unicast traffic alone. It is also notoriously difficult to perform TCP-friendly congestion control on in-network multicast traffic. Therefore, employing in-network multicast may introduce more performance problems than it solves. Existing protocols for multicast routing and group management are also not able to construct multicast trees in milliseconds as required by data analytics applications.

This paper presents Blast, a new approach for accelerating data analytics applications by in-network multicast. Blast uses optical transmission to realize a physical-layer broadcast medium, via passive optical power splitting, to connect a data source to its receivers. Consequently, the broadcast data no longer have to traverse the packet-switched network core and do not interfere with unicast traffic in the core. Blast is designed to leverage several technology trends. As transmission rates evolve to 100Gbps and beyond, optical transmission becomes the standard. Long range optical transceivers are capable of transmitting at high power, enough to reach 100km or more. When used across a short distance in a computer cluster, these transceivers can support hundreds of optical splits and still achieve high reliability. Active optical power amplification can further increase the number of achievable optical splits. In addition, optical power splitting is a very reliable, mature, and inexpensive technology.

A new control plane is necessary to realize this approach. Blast includes a tailor-made control plane, capable of collaborating with data analytics applications interactively, making resource allocation decisions nearly optimally, and directing the data flows in optical and electrical components in the network. We implement Blast on a small-scale hardware testbed. Blast's control plane is very responsive. Multicast transmission can start 33ms after an application issues the request, resulting in a very small control overhead. We evaluate the Blast approach at the scale of thousands of servers through simulation. Using only a 10Gbps optical uplink per rack, Blast achieves upto $102\times$ better performance than the state-of-the-art solutions even when they are used over a non-blocking core network with a 400Gbps uplink per rack.

II. Motivation of Optical Multicast

A. Optical Power Splitting

Optical power splitter is the most basic device for wavelength and bit-rate transparent physical layer data duplication [21]. Depending on its specific design parameters, a power splitter duplicates an incoming data stream by dividing the incident optical power at predetermined ratios to its output ports. Typically implemented using fused fibers, optical power splitter is a basic building-block of the telecom networks for a long time, with applications ranging from in-core routers to FTTx (Fiber to the *x*) installations. Our work is also inspired by the experimental work conducted by Wang *et. al.* [25][26][22], which demonstrates that multicasting through an optical power splitter can enable lossless end-to-end multicast transmission between 1Gbps Ethernet sender and receiver NICs, and it can be faster than multicasting through a non-blocking packet-switched network that is shared by unicast traffic.

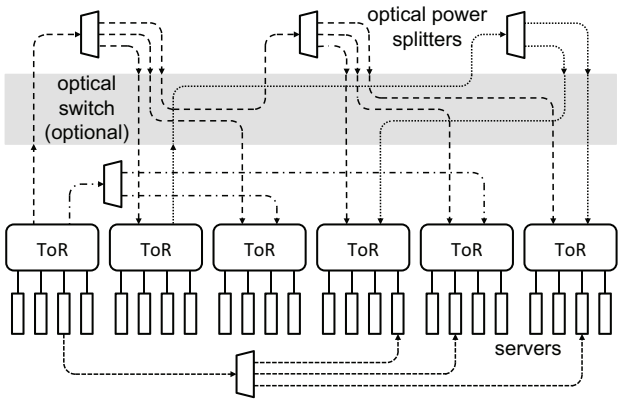

Fig. 2: Using optical power splitters for optical multicast: static connections to servers or to ToR switches; dynamic connections to ToR switches via an optical switch

The excess optical power losses of today's passive power splitters are minimal, but they introduce an insertion loss, e.g. -3 dB for a balanced power split. Since the power of the optical signal must be greater than the sensitivity of the receiver to ensure error-free data recovery, optical interconnects are engineered with sufficient optical link budgets to accommodate these losses. To mitigate the power budget limitation associated with larger-degree power splits, optical amplification is typically leveraged to increase the effective link budget of a given system. While amplifiers are active devices that require additional power, they maintain data format transparency to effectively decouple energy consumption from bit rate.

As a result of mature commoditization, optical power splitters are of low cost and high reliability. On today's market, a 1-to-4 optical power splitter costs $40, a 1-to-16 one $80, and a 1-to-64 one $250. Our measurement confirms that the power of a 1Gbps Ethernet signal can be reduced by a factor of 900 and yet the packet loss rate remains less than 10^{-4}. Optical amplification can further raise this limit, and the cost of an amplifier is moderate, at roughly $1000.

Figure 2 depicts different ways to facilitate optical multicast. Optical power splitters can be connected to servers directly to accommodate static multicast groups. The servers should have an optical transceiver plugged in the optical-compatible network interface card for optical signal transmission. Optical power splitters can also be connected to ToR switches, so that a server beneath the sender ToR can multicast to many servers across the destination ToRs. Most ToR switches nowadays have standard ports for optical transceivers, making the deployment simple. A high-radix optical space switch, e.g. a 3D MEMS switch, can be used as a connectivity substrate for dynamic allocation of optical power splitters. Optical power splitters of various fan-out are connected to a subset of the ports on the optical switch. The ToR switches each have a number of fiber connections to the remaining optical switch ports. Through run-time configuration of the optical switch connections, the optical power splitters can be connected to ToRs as needed. In any of the above ways, optical power splitters of different sizes can be cascaded together to reach a larger number of receiver nodes.

B. Advantages of Optical Multicast

In data analytics applications, the multicast group size varies from 10^2 to 10^3 physical servers. Considering contiguous server placement, a multicast group reaches out to several hundred racks at the maximum. By cascading optical power splitters of various sizes and amplifying the optical signals, trees of up to 1000-way multicasts are realizable [21].

Because optical data duplication is data-rate transparent, data can be transmitted as fast as the transceiver's sending capacity. Using the off-the-shelf 10Gbps or 40Gbps transceivers, high-volume multicast up to several gigabytes can be completed instantaneously. Optical multicast also gives optimal link stress, since the network does not carry duplicated packets. This is more efficient than the state-of-the-art HDFS and BitTorrent overlays that suffer from suboptimal multicast trees. Moreover, using optical multicast, the multicast traffic circumvents the packet-switched network core, causing no interference with unicast traffic in the core. In network overlays, the multicast traffic is transmitted as a large number of flows that take away bandwidth from other unicast flows. Worse still, in-network multicast schemes are not congestion controlled by TCP thus can drive out unicast traffic completely.

The low cost of optical power splitters and optical amplifiers are already discussed. At the cost of a few thousand dollars, we can build static multicast trees to alleviate hot spots of multicast communications. In the case of using an optical switch to dynamically allocate optical power splitters, a 320-port MEMS switch costs $85,000, comparable to a high-end Ethernet switch. With around $100,000, it is feasible to build optical multicast functions in the entire region of a moderate-scale computer cluster.

The power consumption of the optical devices are much less than their electrical counterparts, making the solution even more desirable. Today's 10–40 Gbps optical transceivers can draw anywhere from 1–3.5W per port depending on technology (i.e., single- or multi-mode). Optical power splitters are passive; a 320-port commercial optical switch draws 50W; and optical amplifiers consume little power, on the order of tens of Watts. Moreover, data rate transparency means future proof. The optical devices, once deployed, do not need to be upgraded as the transceiver technology advances.

Comparing to the come-and-go recipients in the Internet multicast scenario, the group members in data analytics applications are fixed, i.e. all the computing nodes for a job. Obtaining this information allows the distributed join/leave protocols to be bypassed and the multicast trees to be built directly. In case of minor membership change, e.g. VM migration or node failure, it is inexpensive to tune the multicast tree accordingly. Furthermore, each multicast tree over the optical network is optimal and has homogeneous depth from the sender to each receiver, ensuring low latency and easy tree construction.

The notorious multicast problems such as congestion control and packet loss recovery are greatly simplified. Photonic interconnects operate at extremely low bit-error rates, i.e. $< 10^{-12}$. Data can flow through the optical channels without experiencing any congestion. The only places of potential congestion are the source ToR switch and the destination ToR switch. Commodity ToR switches are already non-blocking. Thus, to guarantee congestion-free data transmission end-to-end, at the sender ToR switch, Blast simply needs to ensure an optical tree is exclusively used by the assigned multicast sender. At the receiver ToR switch, multicast traffic towards an output port connected to a receiver should be given the highest queueing and transmission priority. Accordingly, data analytics applications can also be customized to pause other unicast traffic during multicast transmission and reception. Thus, congestive packet losses are eliminated, and error induced packet losses are rare, so simple recovery mechanisms such as unicast retransmission can suffice. This is in contrast to a regular packet-switched multicast session where congestion may occur anywhere along the multicast tree and even sophisticated reliable multicast protocols suffer from feedback (NACK/ACK) implosion and packet retransmission congestion collapse [19]. An orthogonal issue is flow control, i.e. to choose a maximum sending rate at which the receivers can keep up with the incoming data. This is an application layer decision because the rate depends on the complexity of the computation being performed by the receivers on the data.

III. Control Plane Design

A. SDN Control

The greatest challenge to the control plane design is that optical data duplication does not fit in with the existing network stack. Imagine the result of connecting routers to the input and outputs of an optical power splitter. The splitter can only send data unidirectionally, from input to output but not vice versa. The unicast routing protocols assume bidirectional links and thus are ignorant of these additional optical paths. Since multicast routing protocols depend on unicast routing protocols for topology discovery, multicast trees will not be built over the optical power splitter. Utilizing the optical multicast function requires multicast data delivery to be performed over the optical power splitter while all other traffic for maintaining the multicast tree to be transmitted through paths elsewhere.

This asymmetric routing behavior can be achieved by a specialized distributed multicast routing protocol. However, the protocol needs to discover the optical paths, to learn the multicast group members, to construct multicast trees, and to route traffic. Considering the high capacity of optical channels, the latency caused by these control procedures is significant compared with the time for multicast data dissemination. High-performance data analytics applications call for agile control mechanisms that can quickly enable multicast transmission.

The software-defined networking (SDN) paradigm provides a means to achieve this goal. First, the centralized control model used in SDN allows the controller to be configured with the topological location of the optical power splitters, and it is possible to allocate them intelligently according to specific application needs. Second, SDN allows arbitrarily complex control decisions and application-network interactions to be programmed, so that it is possible to learn the multicast groups and configure the network to service them instantaneously.

We design an SDN control plane for Blast, which runs on the network controller. As shown in Figure 2, if the optical power splitters are statically placed, the control plane is simple because multicast traffic can be directed to predefined optical

paths straightforwardly; while in case of dynamic provisioning of optical power splitters, the control plane needs to schedule the optical resources and configure the network at run time. Our design focuses on the latter case.

B. Application Awareness

To service multicast traffic, the control plane first needs to collect the traffic demands. Existing unicast-only optical networked systems infer traffic demands from the network without application involvement. For instance, c-Through [24] learns the traffic volume of a unicast flow by reading the socket buffer occupancy; Helios [14], OSA [11], and Mordia [20] compute the bandwidth share of a unicast flow by getting the flow counts on the traversing switches.

However, it is hard, if not impossible, to extend these approaches to multicast communications. The source and destination are clear for unicast transmissions, but multicast needs to obtain the group membership before data delivery can get started. If transparent to applications, the system must fall back to complicated group joining protocols, whose overheads are huge considering the fast transmission speed on the optical channels. Also, the network-level statistics can lead to inaccurate estimation of real traffic demands [7]. This may be fine for the unicast-only electrical-optical hybrid networks, because they can switch unicast traffic back and forth between the electrical network and the optical network and adjust the traffic demands from time to time. However, multicast communications require very accurate traffic demands, because once the transmission starts in other means, such as HDFS and BitTorrent, on the electrical network, it becomes very difficult to synchronize the receivers and migrate the transmission to the optical network.

Fortunately, traffic demand collection can be simplified greatly if we take advantage of the application knowledge. The multicast group membership and traffic volume are readily available at the application manager, or master node, of a data analytics job [2][27]. If we let application managers inform the traffic demands explicitly, the network controller is able to build optimal multicast trees and allocate optical resources instantaneously. We design an application-controller protocol, the details of which are shown in Section IV-A.

C. Control Algorithm

1) Problem Formulation: Given the traffic demands, the network controller runs the control algorithm to compute the optical network topology and to allocate optical power splitters. On a MEMS-based optical circuit switch, an input (output) port can connect to only one output (input) port, which forms a typical matching problem [18].

When multicast traffic comes into being, the problem is turned into hyper-graph matching. A hyper-graph is a generalization of a graph in which an edge can connect any number of vertices. Let $H = (V, E)$ be a hyper-graph, where V is a set of vertices and E is a set of non-empty subsets of V called hyper-edges [18]. A hyper-graph matching is a set of disjoint hyper-edges. Blast can be abstracted as a hyper-graph, where V represents the racks, and E represents the multicast groups that embody different subsets of racks. In practice, each ToR switch may connect to the optical switch through multiple ports, so we generalize the problem to hyper-graph b matching, which is to seek a subset E' of E such that each vertex $v \in V$ is met by at most b hyper-edges in the subset. In our case, b is the number of optical ports per rack. We also associate each hyper-edge with a weight, i.e. the multicast traffic volume.[1]

Our goal is to maximize the amount of multicast traffic undertaken by the optical network, which can be formulated as a maximum weighted hyper-graph b-matching problem. We seek a maximum weight sub-collection of multicast groups such that each rack is occupied by at most the number of optical ports attached to it. In addition, Blast leverages cascaded optical power splitters. The insertion loss of the optical power splitters allows a limited number of splits and the total number of optical power splitters is fixed, so we need to consider these additional constraints:

1) Any multicast group to be serviced cannot occupy more optical ports than the maximum allowable number of cascaded optical power splitters can provide. Suppose N is the output port count of an optical power splitter and k is the maximum number of optical power splitters that can be cascaded. For a particular multicast group i that involves r_i racks, we have $r_i \leq k(N+1) - 2(k-1)$.
2) The total number of consumed optical power splitters cannot exceed the given number. A multicast group of r_i racks consumes $m_i = \lceil \frac{r_i - 2}{N-1} \rceil$ optical power splitters.[2] Let K be the number of optical power splitters in the system, we have $\sum_i m_i = \sum_i \lceil \frac{r_i - 2}{N-1} \rceil \leq K$.

2) Greedy Heuristic: Hyper-graph matching is NP-hard [18]. Although there is no existing model for maximum weighted hyper-graph b-matching with additional constraints, we envision it has similar complexity. We develop a greedy algorithm to approximate the solution.

The multicast groups are sorted by the scoring function $s = volume/\#rack$, which divides the traffic volume (weight) of a multicast group by the number of racks involved. This scoring function gives the traffic volume per optical port. Since optical resources are limited, it balances the traffic volume and the group size such that the optical ports are occupied by the most profitable multicast groups.

The multicast groups that violate Constraint 1 are rejected immediately. Then the algorithm iteratively selects the multicast groups with the highest score as long as every involved rack has appeared in less than b previously selected multicast groups and the cumulative number of consumed optical power splitters does not violate Constraint 2. The algorithm continues until no more multicast groups can be serviced with all the constraints satisfied.

We avoid moving the multicast traffic between the electrical network and the optical network. Once a multicast group is

[1] The multicast traffic volume is defined as the data volume multiplied by the number of receivers, which is the cumulative traffic volume to transmit

[2] To support a multicast group from a sender rack to $r_{\max}$ receiver racks using optical power splitters with N output ports, a complete N-ary tree of cascaded optical power splitters with a depth of $\lceil \log_N(r_{\max} - 1) \rceil - 1$ must be constructed. Since the number of nodes in an N-ary tree increases as a geometric series, the total number of N-port optical power splitters needed to support $r_{\max}$ is $m = \left\lceil \frac{r_{\max} - 2}{N-1} \right\rceil$.

TABLE I: Control algorithm computation time of various optical architectures

Architecture	Problem Formulation	System Configuration	Traffic Pattern	#Rack	Time (ms)
Blast	hyper-graph b-matching	Intel Xeon 2.83GHz CPU, 4GB RAM	real unicast and multicast traffic	1000	166
c-Through	bipartite matching	Intel Xeon 3.2GHz CPU, 2GB RAM	random unicast traffic matrices	1000	640
Helios	bipartite matching	not reported	synthetic unicast traffic	128	118
OSA	b-matching	not reported	average of synthetic and real unicast traffic	80	48

selected, it is dedicated to the optical network until it finishes. So, the control algorithm takes the multicast groups with ongoing transmission as already selected and only allocates the residual resources in each decision process. After a selected multicast group is done with the transmission, the optical power splitters and the optical switch ports occupied by the involved racks are released.

3) Approximation Accuracy and Computation Time: We evaluate the performance of our control algorithm in a previous technical report. Due to space limitation, we summarize the experimental results about the approximation accuracy and the computation time at the high level.

We use "optimality" as the metric for measuring the approximation accuracy. It is defined as the total traffic volume outputted by our control algorithm divided by the optimal solution. We generate unicast traffic according to real data center traffic statistics [6]. For multicast traffic, we vary group size from 100 to 1000 and use data size of 300MB to mimic data analytics jobs. We feed the mixture of unicast and multicast traffic to our control algorithm and an ILP solver respectively. We observe that on a 200-rack setting, optimality is over 99% for all the cases, showing the greedy approximation algorithm is near optimal.

The computation time is measured on a processor core of an IBM iDataPlex system, where each core runs at 2.83GHz and has 4GB RAM. We observe that the computation time is 166ms on a 1000-rack setting. Table I compares the computation time of our control algorithm against that of the existing MEMS-based optical network architectures. In these systems, the optical network can only handle unicast traffic. So, they formulate the resource scheduling as matching problems in general graphs and use existing optimal polynomial-time algorithms for solutions. Despite different system configurations and traffic patterns, we can still perceive the general trend. We seek to solve a more challenging problem, NP-hard in theory, but our control algorithm outperforms that of the other architectures, proving it is responsive enough for practical system implementation.

IV. Control Plane Implementation

A. Application-Controller Protocol

Application managers talk to the network controller via the application-controller protocol. Specific properties of Blast pose the following fundamental requirements for the protocol.

Because responsiveness is critical, the communication overhead should be minimal. Sending messages frequently back and forth imposes significant challenge on the processing power of the network controller. So, application managers should only communicate with the network controller when necessary, e.g. sending multicast requests. The network controller should also be designed to keep states for the requests and only respond to them when the requests are to be serviced.

Because of uncertainties in the network, applications should be allowed to adjust the requests. Although the multicast group membership is pre-known, there are cases in which the members change, such as VM migration and node failure. Also, even if the multicast traffic volume is fixed, the duration of data transmission is unpredictable. In these cases, application managers should update the network controller with the latest status, though still with the minimal number of control messages.

Because optical resources are scarce, the requests should not be serviced on a first-come-first-serve basis. The network may run out of optical resources, so applications cannot expect all requests to be serviced immediately. To avoid an unserviced request blocking all the others, the network controller should communicate with applications asynchronously. Because different requests require different set of resources, all pending requests are considered by the controller at each decision making point. Applications should also be allowed to withdraw a request that has been pending for a long time.

The protocol behaves as follows to meet these requirements. Application managers make requests to convey the multicast traffic volume and the group membership. The requests are rejected immediately if they violate the system constraints. Otherwise, they wait to be serviced until the required optical resources become available. After a request is serviced, the network controller returns the estimated service time and the optical resources are dedicated during this period. Application managers can extend the serviced requests with updated requirements, such as tailoring the group membership and extending the service time. Application managers can also withdraw issued requests. If the withdrawn request has been serviced already, the allocated optical resources are recycled.

Application managers may make requests in different programming languages, and the communication latency is key to the control plane responsiveness. The application-controller protocol can be realized using cross-language RPC frameworks, because they provide well-defined APIs for various programming languages and a complete network stack with optimized end-to-end performance. We choose the Thrift RPC library [1], because it is widely used and supported.

The network controller runs as a Thrift server and the application managers are Thrift clients. The server should be non-blocking to incoming connections and be able to handle concurrent requests, so we choose an asynchronous server mode that uses a single thread for non-blocking network I/O and a separate pool of worker threads to process client requests.

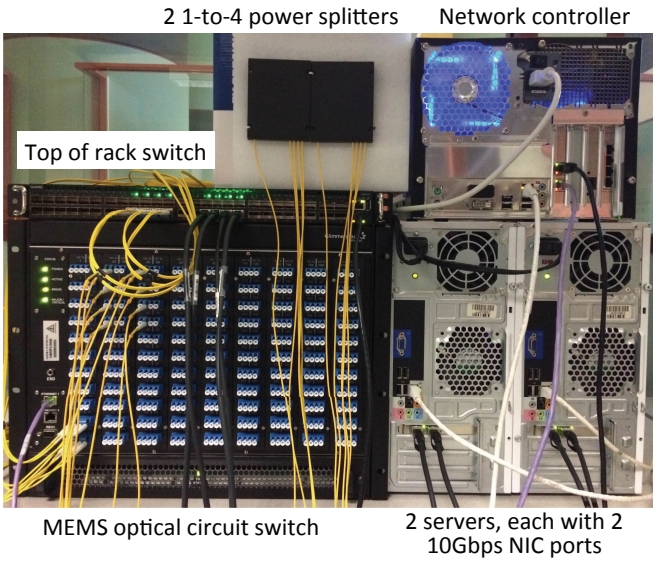

Fig. 3: Blast testbed setup

B. Controller-Network Protocol

The network controller uses the controller-network protocol to reconfigure the network. First, the internal connections on the optical switch need to be created to build multicast trees on the optical network. Second, new OpenFlow rules need to be set on the SDN-capable ToR switches to forward the multicast traffic to the constructed multicast trees.

The network controller can instruct the reconfiguration of physical connections on the optical switch through a software interface, called TL1. To set OpenFlow rules easily, we implement the network controller as a module on the Floodlight SDN controller [4]. The Floodlight default routing behaviors are disabled. Our network controller inserts and deletes OpenFlow rules through the Floodlight REST API. It can set OpenFlow rules and configure the optical switch simultaneously to reduce the overall reconfiguration delay.

V. Testbed

A. System Setting

We implement Blast on a small-scale hardware testbed to evaluate the efficacy and performance of its control plane. The testbed is shown in Figure 3.

Our end-nodes consist of two servers running Ubuntu Linux, each with an Intel Core(TM)2 Quad Q9300 CPU @ 2.50GHz and 4GB RAM @ 800MHz, and two 10Gbps NIC ports. They are connected to a 10Gbps Quanta SDN-capable ToR switch by all the NIC ports (black cables in figure) to emulate four individual end servers each connected to a separate ToR switch. Each logical ToR switch has an optical uplink, consisting of an input fiber and an output fiber (yellow fibers in figure), connected to a Glimmerglass MEMS optical circuit switch. A subset of the optical switch ports are also attached to two 1-to-4 optical power splitters. The network controller runs on a machine with an AMD Athlon(tm) 64 X2 6000+ dual-core CPU @ 3GHz and 4GB RAM @ 667MHz. It is connected to the ToR switch (black cable in figure) and the optical switch (purple cable in figure) each through a management port.

We deploy the control plane on the network controller. The control algorithm is triggered if there are pending requests, and it handles them all at once. We develop a simple multicast application, which complies with the application-controller protocol and runs iperf UDP to generate multicast traffic.

TABLE II: Break down of the control plane latency

Thrift communication	1.747 ms
Control algorithm computation	8.553 ms
OpenFlow rules insertion	8.711 ms
Optical switch configuration	14.172 ms
Total	33.183 ms

B. Experimental Results

We test the functionality of the system by sending requests, extending requests, withdrawing requests, and sending concurrent requests with and without resource conflicts. The system behaves correctly. The sender can send at line rate ($\sim$10Gbps), and the receiver is able to receive at the same rate without any packet loss over the network.

We measure the processing latency of a single request, i.e. the time from sending the multicast request to starting the multicast transmission. Table II shows the break down of the latency. The total latency is 33.183ms, a small overhead to the subsequent data transmission. For instance, sending 1GB data over a 10Gbps link takes 800ms. It should be clarified that a multicast request is issued per iteration, not per job, so that when a job is in its shuffle or aggregation phases, the optical multicast resources can be released and be used by other jobs. The control plane can be optimized to set OpenFlow rules and configure the optical switch in parallel. By doing so, we can reduce the control plane latency to less than 25ms. This means Blast can handle 40 requests per second in the worst case where requests are simply handled one at a time. Because a job sends a single request every iteration, an iteration lasting tens to hundreds of seconds [12], at 40 requests per second, Blast can easily handle hundreds of concurrent jobs running on thousands of servers. To give a concrete example, suppose 50 optical splitters are available, each has sufficient fan-out to handle a multicast iteration. At 40 requests per second, each sender can transmit for 1.25s per iteration or 1.56GB of data at 10Gbps, enough to meet the needs of the data analytics applications that motivated Blast.

VI. Simulation

This section evaluates the performance of Blast in a big cluster setting using flow-level simulations. We demonstrate its performance benefits by comparing to the state-of-the-art multicast approaches for data analytics applications.

A. Simulation Setting

1) Simulation Methodology: There are 120 racks in the simulated network, each with 40 servers. We use the multicast completion time as our performance metric, computed based on the flows' max-min fair share bandwidth. Multicast transmission is completed when the data reach all the receivers.

Detailed transport layer protocol behaviors are not considered in this flow-level simulation, so our simulation results provide an ideal-case lower-bound on completion time for each of the approaches compared. We believe this bound is relatively tight for Blast, because packet loss is expected to be rare in optical channels. In contrast, the bound is fairly loose for the other approaches, because packet loss is more common in a congested network environment. We consider the control plane overhead for Blast as well, which makes the bound even tighter. Specifically, we implement the control algorithm, as shown in Section III-C, and add the request processing delay, as measured in Section V-B, to the multicast completion time.

2) Communication Patterns: We use the multicast phase of one data analytics job iteration as the multicast traffic pattern for our simulation. We also add synthetic unicast traffic of different intensity as background traffic to stress the network to different degrees.

Multicast traffic: In accordance with the group size and the data size of the example data analytics job shown in the introduction, we let 120 servers form a multicast group and 300MB data to be sent. Assuming contiguous server placement, a multicast group spans 3 racks. We randomly choose 3 racks and let the first server in one rack send to all the other servers in the same group. We suppose 90% of the servers are involved in data analytics jobs, so there are 36 simultaneous multicast groups in total and different multicast groups do not share common group members.

Unicast traffic: We create **no**, **light**, and **heavy** rack-level traffic patterns to stress the network core with increasing load. The racks are indexed from 0 to 119 and the servers in each rack are indexed from 0 to 39. The traffic shifts round by round, with old flows removed and new flows created at the beginning of each round. Each round lasts for 1 second.

1) **No unicast traffic**: There is no unicast background traffic at any time.
2) **Light unicast traffic**: In round t, any server j in rack i talks to server j in racks $(i+t\pm1)$ mod 120, $(i+t\pm2)$ mod 120, and $(i+t\pm3)$ mod 120.
3) **Heavy unicast traffic**: In round t, any server j in rack i talks to server j in racks $(i+t\pm1)$ mod 120, $(i+t\pm2)$ mod 120, $(i+t\pm3)$ mod 120, ..., $(i+t\pm40)$ mod 120.

3) Networks Compared: We simulate a 10:1 oversubscribed network and a non-blocking network respectively to show the performance variance under different oversubscription ratios. We assume a single core switch connecting up all the ToR switches. The links between servers and ToR switches are 10Gbps. The links between ToR switches and the core switch are 40Gbps for the 10:1 oversubscribed network and 400Gbps for the non-blocking network. The state-of-the-art approaches for multicast data delivery in data analytics applications are HDFS and P2P overlay. We add naive unicast as the baseline for comparison. We simulate these three approaches on top of the oversubscribed and the non-blocking network. Detailed descriptions of the systems are as follows.

1) **Naive unicast 10:1 oversubscribe/non-blocking**: The data source distributes the data to the destinations by individual flows all at the same time.
2) **HDFS 10:1 oversubscribe/non-blocking**: Hadoop uses HDFS as a medium for multicast. The multicast sender writes the data to HDFS and the receivers read from it. HDFS chunks up the data into blocks and creates replicas for each block. We use the default block size of 64MB and replication factor of 3. That is, each data block is replicated at 3 places, one at the data source, one on a different server in the same rack, and one in a different rack. When the multicast receivers read from HDFS, the data blocks are retrieved from the closest replica.
3) **Overlay 10:1 oversubscribe/non-blocking**: Spark uses BitTorrent to multicast data [27]. BitTorrent suffers from suboptimal multicast trees, and its performance is very poor in practice [5]. The Spark developers propose to use the SplitStream [10] overlay system for performance improvement [5]. We simulate SplitStream as an upper bound for BitTorrent. SplitStream builds optimal interior-node-disjoint multicast trees and thus distributes the forwarding load among all participating peers. We simulate the principle of topology-awareness in the Spark BitTorrent implementation as well [12]. To minimize cross-rack communications, we choose a leading server for the receivers within the same rack. We form a swarm among the leading servers across racks, and they subsequently distribute the content to servers in the same rack as another swarm.
4) **Blast**: We simulate Blast on the 10:1 oversubscribed network. The optical network has a 320-port optical switch, which connects to 40 1-to-4 optical power splitters and to each ToR switch by 1 port. All optical links are 10Gbps. Although the electrical network (40Gbps) and the optical network (10Gbps) are separate, the network core has 50Gbps total bandwidth, making the cumulative oversubscription ratio 8:1. We set the reconfiguration delay as 25ms according to the discussions in Section V-B. The control algorithm computation delay is measured at run time.

B. Simulation Results

Figure 4 plots the simulation results in log scale. We have the following observations:

Using only a 10Gbps optical uplink per rack, Blast achieves upto 102× better performance than HDFS and upto 37× better performance than overlay, even when they are used over a non-blocking core network with a 400Gbps uplink per rack. Blast is significantly better than the other approaches because of two essential reasons. First, it has optimal link stress, so no data get transmitted unnecessarily. Alternative systems, however, even if given tremendous core bandwidth, have a large number of duplicated flows that reduce the transmission speed. Second, it has a separate optical network dedicated to the multicast communications. So, the multicast traffic can transmit on a "clean" network core, without sharing bandwidth with the background flows.

The benefit of Blast becomes more notable when the network is heavily stressed. As the unicast background traffic increases, the multicast completion time of the Blast only

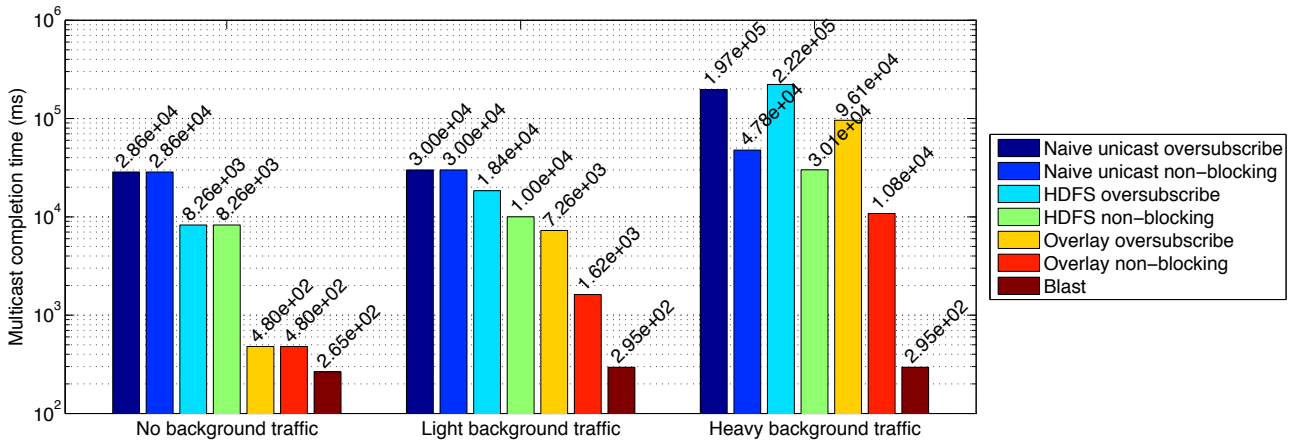

Fig. 4: Average multicast completion time (ms) of different approaches under various background traffic

grows from 0.265s to 0.295s, in contrast to dramatic degradations of the other approaches. With no background traffic, the servers get the full link capacity and thus the transmission time is optimal. In case of background traffic, whether light or heavy, the multicast transmission remains at a constantly fast speed. This is because the cross-rack background flows get rate-limited by the congested core of the oversubscribed electrical network, thus the fixed amount of residual bandwidth at the edge can be utilized by the multicast transmission.

Blast is expected to get better performance even compared with IP multicast on a separate 10Gbps network, because it does not require complex multicast routing protocols. IP multicast involves complex protocols for maintaining the group membership, building the multicast tree, achieving congestion control and flow control, etc, which are very heavy overheads for the data transmission. Yet the Blast control plane design simplifies these functions greatly. Even with the same link stress and on networks of the same capacity, Blast will outperform IP multicast.

We also analyze the weaknesses of the other approaches as follows:

Naive unicast has a link stress as high as the number of receivers. On the oversubscribed network, due to the sender's high fan-out, the network edge is more congested than the network core given light unicast background traffic. Adding bandwidth to the network core brings no improvement in this case. So, the non-blocking network is only helpful when the background traffic is heavy enough to stress the network core.

HDFS is more efficient than naive unicast, but its benefit diminishes as the background traffic increases. In HDFS, receivers read from data replicas in different places, so the load for data distribution gets balanced. Since large data are chunked up into blocks that can be retrieved simultaneously, the sending capacity is multiplied as well. However, since data blocks are always replicated across racks, the replication process slows down when a large number of background flows share the core bandwidth. This also explains why the non-blocking network helps for HDFS. On the oversubscribed network with heavy background traffic. The data replication is so slow that the overall multicast completion time is even worse than that of the naive unicast.

Overlay leads to faster transmission because the sender creates many unicast sessions to send a piece of data to each of the receivers, thus taking a greater cumulative fair share bandwidth from the competing flows. Interestingly, the Spark principle of minimizing cross-rack communications is not as effective as expected. Although the peers can avoid seeking data from other racks at reduced speed, having the minimal number of cross-rack flows loses the power of grabbing bandwidth as a swarm and the downstream flows are thus bottlenecked. This effect is observed as the background traffic increases, and the non-blocking network helps mitigate the effect by accelerating the cross-rack overlay flows.

VII. Related Work

We have discussed the state-of-the-art approaches for multicast data dissemination in computer clusters. Hadoop builds an overlay over HDFS to spread data via multiple data replicas [2]. Cornet develops a BitTorrent-like protocol optimized for the cluster computing applications [12], and the method is adopted by Spark [27]. Twitter uses BitTorrent to distribute software updates to its servers [3]. We have shown in Section VI that Blast can outperform these solutions by a large factor. Using only a 10Gbps optical uplink per rack, Blast achieves upto 102$\times$ better performance than these solutions even when they are used over a non-blocking core network with a 400Gbps uplink per rack.

Datacast presents another overlay approach based on in-switch packet caching and edge-disjoint Steiner trees [9], which is more efficient than end-host overlays. Blast has higher transmission speed than Datacast, because it uses high-rate optical channels dedicated for multicast traffic. Multicast traffic can be sent at line rate as it no longer competes with unicast traffic in the packet-switched network. Datacast is not readily implementable because packet caching adds significant complexity to switches, and only specialized network structures such as BCube and CamCube can benefit from multiple Steiner tree forwarding. In contrast, Blast is practical because it uses off-the-shelf SDN switches and optical power splitters.

A number of studies aim to reduce the switch states required for implementing IP multicast trees in the data center environment. Dr. Multicast selectively maps multicast to unicast transmissions to reduce the number of multicast trees required [23]. Li *et al.* design a novel multi-class Bloom Filter to efficiently compress the multicast forwarding table [15]. ESM reduces the switch states required by multicast by combining in-packet Bloom Filter and in-switch routing entries [16]. Blast makes these solutions unnecessary, because it eliminates the need to build multicast trees in the packet-switched network. RDCM assists reliable group data delivery by repairing lost packets in a peer-to-peer way [17]. Blast can eliminate congestive packet losses, but may leverage RDCM to help repair occasional error-induced packet losses.

VIII. Conclusion

This paper describes the design and implementation of Blast, a system for accelerating data analytics applications by physical-layer multicast using the optical power splitting technology. Blast uses an SDN control plane to incorporate optical data duplication into the existing network stack. By getting multicast requests directly from applications, the control plane can allocate optical resources intelligently based on the applications' traffic demands and group memberships. A greedy-based control algorithm can achieve near-optimal optical resource allocation with relatively low time cost. The reconfiguration mechanisms can setup optical paths and change the network routing behaviors at run time to realize multicast routing accordingly.

We implement Blast on a small-scale hardware testbed. Multicast transmission can start 33ms after an application issues a request, resulting in a very small control overhead. The control plane can easily handle hundreds of concurrent data analytics jobs running over thousands of servers. With a few tens of optical splitters, a sender can transmit on the order of a gigabyte of data per iteration, meeting the needs of data analytics applications that motivated this research. We evaluate Blast's performance at the scale of thousands of servers through simulation. Using only a 10Gbps optical uplink per rack, Blast achieves upto 102× better performance than the state-of-the-art solutions even when they are used over a non-blocking core network with a 400Gbps uplink per rack.

Acknowledgment

This research was sponsored by the NSF under CNS-1422925, CNS-1305379 and CNS-1162270, by an Alfred P. Sloan Research Fellowship, an IBM Faculty Award, and by Microsoft Corp.

References

[1] Apache Thrift, http://thrift.apache.org.

[2] Hadoop, https://hadoop.apache.org.

[3] Murder, http://engineering.twitter.com/2010/07/murder-fast-datacenter-code-deploys.html.

[4] Project Floodlight, http://www.projectfloodlight.org/floodlight/.

[5] Spark Technical Report, http://www.cs.berkeley.edu/ agearh/cs267.sp10/files/mosharaf-spark-bc-report-spring10.pdf.

[6] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan. DCTCP: Efficient Packet Transport for the Commoditized Data Center. In *ACM SIGCOMM'10*, New Delhi, India, Aug. 2010.

[7] H. H. Bazzaz, M. Tewari, G. Wang, G. Porter, T. S. E. Ng, D. G. Andersen, M. Kaminsky, M. A. Kozuch, and A. Vahdat. Switching the Optical Divide: Fundamental Challenges for Hybrid Electrical Optical Datacenter Networks. In *SOCC '11*, pages 1–8, Cascais, Portugal, Oct. 2011.

[8] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, 3:993–1022, Mar. 2003.

[9] J. Cao, C. Guo, G. Lu, Y. Xiong, Y. Zheng, Y. Zhang, Y. Zhu, and C. Chen. Datacast: a Scalable and Efficient Reliable Group Data Delivery Service for Data Centers. In *CoNEXT '12*, pages 37–48, Nice, France, Dec. 2012.

[10] M. Castro, P. Druschel, A.-M. Kermarrec, A. Nandi, A. Rowstron, and A. Singh. SplitStream: High-bandwidth Multicast in Cooperative Environments. In *SOSP '03*, pages 298–313, Bolton Landing, NY, USA, Oct. 2003.

[11] K. Chen, A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, X. Wen, and Y. Chen. OSA: An Optical Switching Architecture for Data Center Networks with Unprecedented Flexibility. In *NSDI '12*, San Joes, CA, USA, April 2012.

[12] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica. Managing Data Transfers in Computer Clusters with Orchestra. In *SIGCOMM '11*, pages 98–109, Toronto, Canada, Aug. 2011.

[13] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[14] N. Farrington, G. Porter, S. Radhakrishnan, H. H. Bazzaz, V. Subramanya, Y. Fainman, G. Papen, and A. Vahdat. Helios: A Hybrid Electrical/Optical Switch Architecture for Modular Data Centers. In *SIGCOMM '10*, page 339, New Delhi, India, Aug. 2010.

[15] D. Li, H. Cui, Y. Hu, Y. Xia, and X. Wang. Scalable Data Center Multicast Using Multi-class Bloom Filter. In *ICNP '11*, pages 266–275, Vancouver, Canada, Oct. 2011.

[16] D. Li, Y. Li, J. Wu, S. Su, and J. Yu. ESM: Efficient and Scalable Data Center Multicast Routing. *IEEE/ACM Transactions on Networking*, 20(3):944–955, June 2012.

[17] D. Li, M. Xu, M.-C. Zhao, C. Guo, Y. Zhang, and M.-Y. Wu. RDCM: Reliable Data Center Multicast. In *INFOCOM '11*, pages 56–60, Shanghai, China, Apr. 2011.

[18] L. Lovasz and M. D. Plummer. *Matching Theory*. AMS Chelsea Publishing. American Mathematical Society, 2009.

[19] K. Obraczka. Multicast Transport Protocols: a Survey and Taxonomy. *Communications Magazine, IEEE*, 36(1):94–102, 1998.

[20] G. Porter, R. Strong, N. Farrington, A. Forencich, P. Chen-Sun, T. Rosing, Y. Fainman, G. Papen, and A. Vahdat. Integrating microsecond circuit switching into the data center. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, SIGCOMM '13, pages 447–458, New York, NY, USA, 2013. ACM.

[21] R. Ramaswami, K. Sivarajan, and G. H. Sasaki. *Optical Networks: A Practical Perspective*. Morgan Kaufmann, 3rd edition, 2009.

[22] P. Samadi, D. Calhoun, H. Wang, and K. Bergman. Accelerating Cast Traffic Delivery in Data Centers Leveraging Physical Layer Optics and SDN. In *Optical Network Design and Modeling, 2014 International Conference on*, pages 73–77. IEEE, 2014.

[23] Y. Vigfusson, H. Abu-Libdeh, M. Balakrishnan, K. Birman, R. Burgess, G. Chockler, H. Li, and Y. Tock. Dr. multicast: Rx for Data Center Communication Scalability. In *EuroSys '10*, pages 349–362, Paris, France, Apr. 2010.

[24] G. Wang, D. G. Andersen, M. Kaminsky, K. Papagiannaki, T. S. E. Ng, M. Kozuch, and M. Ryan. c-Through: Part-time Optics in Data Centers. In *SIGCOMM '10*, page 327, New Delhi, India, Aug. 2010.

[25] H. Wang, C. Chen, K. Sripanidkulchai, S. Sahu, and K. Bergman. Dynamically Reconfigurable Photonic Resources for Optically Connected Data Center Networks. In *OFC/NFOEC '12*, 2012.

[26] H. Wang, Y. Xia, K. Bergman, T. S. E. Ng, S. Sahu, and K. Sripanidkulchai. Rethinking the Physical Layer of Data Center Networks of the Next Decade: Using Optics to Enable Efficient *-Cast Connectivity. *SIGCOMM Computer Communication Review*, 43(3):52–58, July 2013.

[27] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica. Spark: Cluster computing with working sets. In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Cloud Computing*, HotCloud'10, pages 10–10, Berkeley, CA, USA, 2010. USENIX Association.