IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 12, 2018, accepted April 20, 2018, date of publication May 11, 2018, date of current version June 5, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2830799

OEHadoop: Accelerate Hadoop Applications by Co-Designing Hadoop With Data Center Network

YINAN TANG, HONGXIANG GUO, TONGTONG YUAN, QI WU, XIANG LI, CEN WANG, XIONG GAO, AND JIAN WU, (Member, IEEE)
Beijing University of Posts and Telecommunications, Beijing 100876, China

Corresponding author: Hongxiang Guo (hxguo@bupt.edu.cn)

This work was supported by the NSFC Program under Grant 61331008 and Grant 61471054.

ABSTRACT Big data applications in Hadoop usually cause heavy bandwidth demand and network bottleneck in the current data center network (DCN). On one hand, the design of DCN does not take the traffic demand and the traffic patterns of Hadoop applications into account. On the other hand, Hadoop suffers from inherent performance limitations due to its solution for transmitting massive data sets based on application-layer overlays, which ignores the architecture of DCN. To improve the performance of Hadoop applications, in this paper we propose the OEHadoop, a modified Hadoop which is built by co-designing Hadoop with hybrid optical and electrical data center network. For Hadoop, we redesign the pipeline-based replication process of MapReduce jobs to optical multicast. For the DCN architecture, we build a reconfigurable optical multicast system to adapt the DCN architecture to multicast traffic. A software-defined networking controller is implemented in data center to adjust the DCN architecture and exchange information with application layer. In order to accelerate the MapReduce jobs, a new algorithm to properly schedule the multicast requests is presented and deployed in the controller. We build a small-scale prototype of the OEHadoop to evaluate the control overhead, and to demonstrate the feasibility of our OEHadoop. In a simulation at a scale of real DCN, we present that our multicast requests scheduling algorithm outperforms related state-of-the-art solutions, and the MapReduce jobs in our OEHadoop speed up to about 2 times faster on average than native Hadoop.

INDEX TERMS Optical interconnections, computer networks, multicast communication, Hadoop, data center network.

I. INTRODUCTION

In the age of big data, large-scale data-parallel computations are widely implemented in commercial data centers [1], [2]. Due to the massive network traffic volume caused by these distributed computations, the traffic leads to increasing burden on data center networks (DCN). Such bottleneck of network limits the performance of applications, and results in slowing down the computations.

Hadoop [3] is a popular distributed computation framework to handle big data mining problems. A Hadoop implementation contains a master node and several worker nodes. The master node, called NameNode, manages the request jobs from users, divides each job into several tasks, and then assigns the tasks to workers. The workers called DataNodes are in charge of data storage and executing the tasks scheduled by NameNode. The computing jobs in Hadoop are running based on Hadoop MapReduce [4], which is one of the most commonly used distribution programming model. In the MapReduce procedure, there are three steps involving network traffic transmission during their execution. The three steps are: (a) Mappers fetch the input data splits from remote DataNodes of Hadoop distribution file system (HDFS) [3] if the data is not locally existed. (b) Mappers shuffle the intermediate results to reducers. (c) Reducers write the output result replicas to a central number (typical three) of DataNodes in HDFS by pipeline communication model. Although the designers of native Hadoop try to reduce the network communication during data-parallel computations (e.g. the pipeline model in replication stage, the locality of the data import process in mapper, the job placement [5]), the Hadoop applications still suffer from the large burden caused by network communication [6], [7]. Figure 1 shows the network communication during a MapReduce job. The traffic caused by step (a) is small because mappers are usually allocated and launched at DataNodes which are close to the input splits [3]. In the other two steps, not

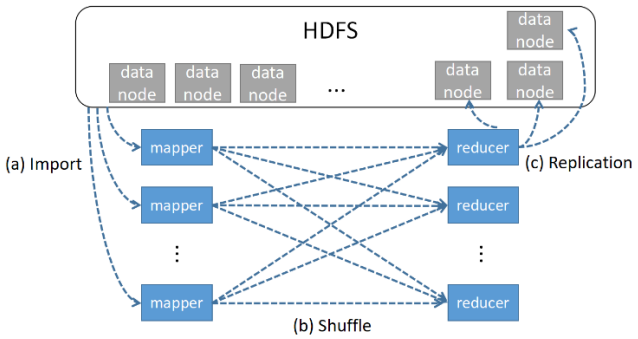


FIGURE 1. Traffic pattern of each stage in a MapReduce job.

only the all-to-all network traffic generated by shuffle stage (b) largely affects the performance of MapReduce jobs [34], but also the fan-out traffic in replication stage (c) makes up a large proportion of whole network traffic in a job. According to the previous studies [8], [28], [38], the write operation in step (c) can produce nearly 50% the total network traffic in Hadoop clusters and significantly affects the performance of application jobs. As a result, the solutions to improve the network performance of Hadoop applications are urgently needed.

Due to the potential of low latency, high throughput and low power of optical communication, hybrid optical and electrical data center architecture has been regarded as a promising solution for next generation data center architecture. Till now, numerous hybrid electrical and optical DCN solutions have been proposed [9]–[12] to accelerate the Hadoop applications in DCN. The main idea of these researches is to the offload the heavy long-live network traffic of Hadoop jobs by optical communication and transmit the fine-grained traffic of the jobs by electrical switching. However, these methods fail to consider the native communication patterns of Hadoop applications. Besides the above methods which change the DCN architecture, to improve the performance of Hadoop applications in DCN, several proposals attempt to modify application itself [13], but result in sub-optimal performance because they fail to take the DCN architecture into account.

The most preferred way to improve the performance of Hadoop jobs is to co-design the network architecture and Hadoop applications. Although several works have made some trials, they do not achieve promising improvement due to some limitations. For example, Xia *et al.* [14] and Samadi *et al.* [15] utilize optical circuit switches (OCS) to establish optical-layer one-to-many connections and fully offload the multicast traffic between top of racks (ToRs) caused by the replication stage to the optical network. These methods suffer from non-negligible reconfigurable delay and exclusive optical links. Bao *et al.* [16] use hybrid electrical and optical DC architecture to accelerate DC applications, but its greedy optical resources assignment algorithm cannot achieve good performance in terms of Hadoop application completion time.

In order to accelerate the Hadoop jobs more effectively, in this paper we present optical-electrical Hadoop (OEHadoop), a modified Hadoop to accelerate Hadoop applications by co-designing Hadoop with hybrid optical and electrical data center. In our OEHadoop, to reduce the bottleneck caused by the replication stage, we change the pipeline communication model to optical multicast. To relieve the pressure in electrical packet switching (EPS) based core switches, we offload the heavy one-to-many traffic into a specially designed optical switching system and multicast it by passive optical splitters. An effective multicast requests scheduling algorithm is implemented in OEHadoop to properly assign multicast requests to DCN and meanwhile reduce the frequency of reconfiguring OCS as much as possible. Moreover, an SDN-based control plane to exchange information between the application layer and the network control layer is also included in the OEHadoop.

In our work, a small-scale prototype is built to verify the feasibility and capability of the OEHadoop. Through experiment we demonstrate the efficiency of our control plane and the requests scheduling algorithm. With the cooperation of Hadoop and DCN architecture, our OEHadoop achieves better applications completion time compared with native Hadoop. We also build a large scale simulation of the OEHadoop to investigate its performance. Results show that the requests scheduling algorithm in our OEHadoop achieves much better performance than state-of-the-art methods. In the simulation, we also discuss the MapReduce job completion time of a real dataset under different over-subscription ratios. Simulation results indicate that OEHadoop achieves up to nearly 2× better performance than native Hadoop under the over-subscription ratio 1:10.

The remainder of this paper is organized as follows. In Section II we introduce the related works in recent years. Section III describes the details of OEHadoop, including the DCN architecture of OEHadoop, the design and modification of Hadoop, and the control plane implementation. Section IV presents the prototype of OEHadoop and experiment results. Section V reports and analyzes the simulation results of OEHadoop in comparison to the performance of existing methods. Finally, we conclude our work in Section VI.

II. RELATED WORKS

Recent years, researchers have proposed various approaches to improve the performance of the cluster computing applications running in DCN. According to the research Benson *et al.* [6], almost 80% of the flows in MapReduce jobs are smaller than 10KB and the most traffic volume are carried in the top 10% of elephant flows. As a result, researches [9], [10] present hybrid optical and electrical DCN architectures to offload the long-live heavy traffic of Hadoop applications by optical switching technology. Besides, the proposed method in [11] utilizes reconfigurable all-optical interconnection architecture to adapt the DCN topology to the traffic demand of applications. Although these architectures can offer significant

reductions in terms of power consumption, amount of switching elements and cabling, they ignore the native traffic patterns of applications and result in sub-optimal applications completion time. The programmable OCS based scalable optical DCN architecture [12] can support diverse communication patterns (*-cast [17]) based on topology reconfiguration by a central controller. However, the interconnection between ToR pairs in its DCN architecture only contains OCS layer connections, so the fine-grained traffic caused by applications (e.g. the traffic generated by the shuffle stage in Hadoop applications) is hard to be fully satisfied. Moreover, when the scale of DCN becomes larger, the difficulty to control the whole DCN will rapidly increase. The above researches attempt to change the architecture of DCN in order to improve the performance of DCN, but none of them successfully solves the problem of the application traffic in DCN. Different from the solutions above, Wu *et al.* [13] modify the replication stage of MapReduce jobs. Instead of using native TCP-based pipeline replication method, the researchers develop a congestion-controlled reliable multicast socket for HDFS. Although this method uses multicast to solve the bottleneck problem of one-to-many traffic in some degree, the heavy multicast traffic with large volume is still coexisting with the all-to-all shuffle network traffic, and causes congestion in EPS based DCN.

Most relative works to our research are [14]–[16]. [15] proposes an optical multicast system which uniquely integrates passive optical splitters in a hybrid network architecture for simpler and faster delivery of multicast traffic flows. [14] uses the similar optical multicast system in [15] to replace the application overlay of delivering the massive data sets in distribution computing. These methods deploy reconfigurable OCS multicast system to offload all multicast traffic, but suffer from exclusive optical links and non-negligible reconfigurable delay when small multicast requests occupy a large proportion. Besides, the flow scheduling algorithms in [14] and [15] aim to achieve maximum throughput and minimize the negative effect of reconfiguring OCS topology. Although these methods exhibit good performance in terms of network, the performance of Hadoop applications is sub-optimal because they ignore the application layer information (the details will be shown in session III). Another research named HERO [16] integrates hybrid electrical and optical multicast to accelerate high performance DC applications. However, HERO uses greedy optical links assignment algorithm which schedules network by a FIFO way, leading to a poor performance. It is worth to mention that although the goal of these methods is to accelerate the cluster computing applications, instead of using a real application traffic model, they simply demonstrate the performance of their system by transmitting a batch of multicast flows. Because the one-to-may traffic is only a part of whole network traffic inside an application job, the actual performance improvement of applications in these works is still needed to be further discussed.

To overcome the disadvantages of previous researches, our OEHadoop is built by co-designing Hadoop with hybrid optical and electrical data center. The details of OEHadoop will be discussed in the next section.

III. DETAILS OF OEHadoop

In this section we will introduce OEHadoop, including the network architecture, the modifications of Hadoop, the network control plane, and the requests scheduling algorithm.

A. NETWORK ARCHITECTURE OF OEHadoop

In order to support the traffic model of MapReduce jobs, we build a hybrid optical and electrical data center architecture for our OEHadoop, which is shown in figure 2. The bottom of figure 2 shows a typical tree based EPS architecture. On the basis of the EPS structure, we add a MEMS-based optical switching system as shown on the top of the figure 2. In the optical switching system, several passive optical splitters are connected to the optical ports of an optical space switch (OSS). Because the replication number of output files is 3, the size of the splitters is determined as 1:3. The top-of-rack switches (ToRs) in the EPS structure are connected to the OSS based on point-to-point bidirectional optical links. Based on properly setting the optical path in the OSS, the optical switching system can support one-to-many traffic demand by optical multicast.

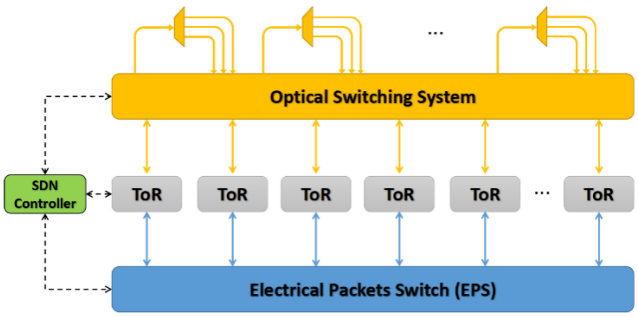


FIGURE 2. Hybrid electrical and optical data center architecture of OEHadoop.

To design and build a cloud-based DCN, application-driven networking [18] and software-defined networking (SDN) control framework [19] are candidate solutions. As shown in figure 2, in OEHadoop, we use SDN technology to realize the control plane of whole network. The ToRs, electrical core switches and OSS in DCN are OpenFlow protocol [35] supported and each of them is connected with an SDN controller through an independent control channel. The controller can schedule the whole network resources according to the application layer network traffic requests and the requests scheduling algorithm (detailed in Section III. D).

B. MODIFICATIONS OF HADOOP

1) REDESIGN OF COMMUNICATION IN THE REPLICATION STAGE

In the native MapReduce applications, the communication dataflows can be broadly characterized into three steps: import, shuffle and replication. Because most map tasks read their input splits locally, the communication in the import step

is rare compared with other steps, so we ignore the traffic generated by import step in OEHadoop. When the shuffle stage starts, the reducer starts to fetch intermediate result from mappers, and there is one segment in each mapper typically. These fetch operations usually cause heavy all-to-all traffic among DCN. According to the recent researches [6], [7], [20], the network traffic in the shuffle stage consists of a large proportion of short-live small flows, and leads to complexity in the network resources scheduling. So a good choice is to guarantee that the free network bandwidth resources are as large as possible when shuffle appears in DCN. However, in the third step of MapReduce, the replication stage will also bring heavy network traffic. The result data is partitioned into several fix-sized data blocks (typical 128 MB or 256 MB, the last data block of a result may be smaller) and each data block is replicated to three DataNodes in HDFS by a pipeline communication model. If the total output traffic volume is large, the traffic caused by the replication step can be regarded as heavy long-live background flows. In a large scale commercial DCN, the interaction between multiple Hadoop jobs is inevitable. So in Hadoop clusters, the replication flows and the shuffle flows will coexist, leading to resource competition and congestion in the network and affecting the performance of Hadoop applications.

The most preferred method to offload the background flows is to transmit them by OCS. However, the traffic pattern of the replication stage is one-to-many type, which is ineffective if we deliver it based on unicast one by one [14], [15]. Fortunately, data multicast by optical splitters is demonstrated to be lossless and effective [17], [36], [37]. Based on the specific optical switching system shown in figure 2, the one-to-many traffic in Hadoop applications can be effectively supported by optical multicast.

Native Hadoop places date replicas at 3 DataNodes, including a source node, a different node in the same rack, and a different node in a different rack. In our OEHadoop, we modify the replication stage by multicasting the replicas to three DataNodes in different racks by optical multicast. Such modification can not only improve the reliability (the number of racks with replicas is increased from 2 to 3) of storage system [21], but also it will accelerate the replication stage in MapReduce applications [15]. The Fault-tolerant problem of optical layer multicast can be solved by reliable multicast NORM [22] or CCRMSocket [13]. Based on such modification, the replication traffic is transmitted by the OCS layer optical multicast, while the shuffle traffic is delivered by the EPS network. The completion time of the shuffle stage and the replications stage will be reduced by this traffic separation. As a result, such modification will accelerate the completion time of MapReduce jobs.

2) MESSAGE EXCHANGE SYSTEM IN OEHadoop

To realize cooperative work between the application layer and the network layer, an additional message exchange system is crucial to be realized. Due to the limitation of the optical resources (e.g. number of optical transceiver and splitters) in the optical switching system, the replication network demands need to be properly allocated. When a replication request appears, instead of directly starting data transmission, the application should report the request information to the SDN controller and obey the assignment from the controller. So we design a message exchange system in Hadoop to exchange control information with the SDN controller.

In OEHadoop, when a MapReduce job starts, the NameNode will firstly notify the total number of the reducers in this task to the SDN controller. Then when a reducer finishes its reduce task, it will send the message <job ID, reducer IP, replication destination IPs, replication output size> about its replication requests to the controller. All of the information is readily available at the NameNode, or the application manager, of a MapReduce job [20]. Because multi-jobs are possible to be submitted at a same time in a Hadoop cluster, we use the job ID to identify a unique job. The reducer IP and the replication destination IPs are indirect factors which indicate the location of source and destination nodes. According to the IP address, the SDN controller is capable of building multicast tree by allocating control messages to ToRs and the OSS. Moreover, the replication output size and the number of the reducers are crucial parameters in our flow scheduling algorithm. After notifying the controller, the reducers need to wait for the allocation result. Once a reducer receives the order, the reducer starts to transmit the output data blocks by optical switching system. Moreover, the reducers also need to suspend its transmission at any moment if the new arrival scheduling result doesn't allow their transmission.

C. CONTROL PLANE IMPLEMENTATION

The greatest challenge to build a control plane is that the optical multicast does not exist in current network protocol stack. Because the passive optical splitters are unidirectional devices, switches cannot automatically discover such physics link and will ignore these additional optical paths. Since existing multicast protocols depend on bidirectional communication for topology discovery, if we simply use traditional multicast to transmit the replicas, the current IGMP protocol will automatically build multicast trees and schedule the multicast requests to the EPS network. Although modifying the network stack is one of the solutions, the difficulty will be large because we have to redesign the kernel program in ToR switches.

Fortunately, SDN technology provides possibility to simply solve the problems above. The SDN technology allows application-network interactions and complex decisions to be programmable. A centralized control model used in the SDN technology can collect the global information, conveniently manage the network and determine the routing, so the SDN controller is capable of adjusting the network topology and building multicast trees. As a result, in OEHadoop, we realize a central controller based on the SDN technology.

In the SDN controller, the whole network topology information is recorded. To build the multicast tree in DCN,

according to the the source and destinations of a multicast request, the SDN controller will build network paths along the multicast tree. First, the SDN controller will reconfigure the OSS ports. In the OSS, the input port connected with source ToR will be linked to the input port of a 1:3 splitter, while the output ports of the splitter will be linked to three destination ToRs. Then, after the reconfiguration, the controller sends control messages to the source and destination ToRs to match the correspond field in packets and forward these packets along the multicast tree. As a result, the optical multicast can be successfully supported.

D. FLOW SCHEDULING AND OCS RECONFIGURATION ALGORITHM

In OEHadoop, the OCS resources and the number of passive optical splitters are limited. So how to reasonably utilize the restricted bandwidth and the optical resources is the most important problem of the control algorithm.

1) PROBLEM ANALYSIS

Given the multicast traffic requests, the goal of the algorithm is to minimize the average complete time of MapReduce jobs. So the SDN controller needs to compute the priority of each network request based on the scheduling algorithm, and firstly schedules the multicast flows with higher priority into the optical switching system. The control algorithms in previous works [14]–[16] are not comprehensive because they only take independent multicast requests into consideration. In order to accelerate MapReduce applications as much as possible, our scheduling algorithm introduces the network abstract concept named Coflow [23], which can express most communication patterns of nowadays data-parallel applications. Because there is not only one reduce task in a MapReduce job, the job completion time is determined by the finishing time of the slowest replication traffic in this job. In other words, the job completion time is depended on a group of multicast requests caused by multiple reducers in this job. Inspired by the theory of coflow, we regard these relevant multicast requests from a same job as a co-multicast, and all these relevant requests need to be taken into account at the same time. The multicast flows can be grouped into co-multicasts by their job IDs.

According to the theory about smallest-first scheduling policy to minimize the average flow completion time [24], [25], the smallest co-multicast first scheduling mechanism is a natural choice. We can borrow the method of these researches to some extent to determine the priority of co-multicast requests. However, the requests waiting for allocating are multicast flows instead of unicast flows, while the data plane is based on OCS but not EPS architecture. So in OEHadoop, the co-multicast scheduling problem needs to be reconsidered and reformulated. Figure 3 shows three co-multicast requests submitted to controller. Different with the EPS network, the schedule of optical multicasts is limited by optical resources. Because the OCS links are passive optical link without intermediate switching, the source and

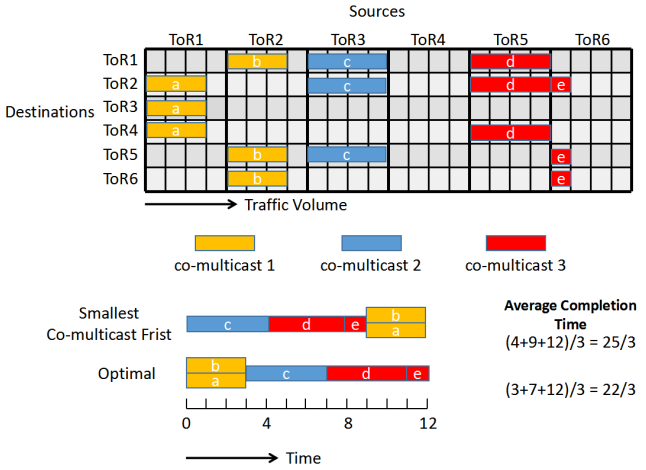


FIGURE 3. This picture shows the comparison between optimal scheduling and smallest-first scheduling. Assume that each ToR in this case only equips one optical transceiver. The top of the picture shows an example of current co-multicast requests where multicast requests belong to a same co-multicast have the same color. Different multicast requests are identified by different letters. The multicast requests in the same row or column have the same destinations or source. The length of requests stands for the traffic volume units. We assume that the optical multicast system can transmit one unit of traffic volume in a time unit. The bottom shows the average completion time of co-multicast requests based on different scheduling methods.

destinations of a multicast request must synchronously send and receive. Besides, due to the limitations of the optical transceivers in ToRs, the multicast requests with the same source or destinations cannot be synchronously transmitted if the source or destination ToRs have no more available optical transceivers. For example, the requests c and d in figure 3 cannot be scheduled at a same time due to the conflict in their destinations. According to the smallest-first scheduling, because the traffic volume in the co-multicast 2 is 4 units, the first request to be satisfied should be the co-multicast 2. But in this scenario, although the traffic volume of the co-multicast 1 is 6, the multicast flows in the co-multicast 1 can be simultaneously transmitted. We can find that if we firstly schedule the co-multicast 1 into the system, the average complete time will be reduced.

To improve the performance of Hadoop applications, given the size, the source and destinations of each multicast flow in co-multicast requests, the control algorithm must decide the priority of each multicast request under the restrictions of OCS link resources to minimize the average complete time of submitted co-multicasts. In this process, new arrival requests will also affect the scheduling results. Although there is no existing method for such scheduling problem, we envision that the optimal co-multicast scheduling problem is an NP-hard problem which is useless in a changeable online network system. So we develop a heuristic method to solve the scheduling problem.

2) HEURISTIC ALGORITHM

In the scheduling algorithm, the first problem is how to decide the priority of each request. To effectively allocate

the co-multicasts with low complexity, for each co-multicast, we define a width W which takes both the traffic volume and the optical resources limitation into account. Given the traffic volume V, the source and destinations, the relationship of each multicast request i in a co-multicast m, and the number of optical transceivers t in each ToR, we define the traffic burden $TB_{m,N}$ as the minimum time needed to completely transmit or receive all the network demand of m in the source or destination node N. The $TB_{m,N}$ can be calculated by algorithm 1.

Algorithm 1 Traffic Burden Calculation

Input:

A source (or destination) node N.

The traffic volume V_i of each multicast request i in the co-multicast m if i has the same source (or destination) with node N.

The number of optical transceivers t in each ToR.

Output:

The traffic burden $TB_{m,N}$ of m carried by node N.

Steps:

1. Sort the traffic demand volume of the multicast requests by V_i.
2. Assign V_i from the biggest to the smallest to each idle optical port p in N until there are no more idle ports in N. Record the traffic demand volume DV_p carried by each port p.
3. Assign the next largest request to the port with the smallest DV_p and update DV_p. Loop this step until all the requests of m in node N are assigned.
4. Calculate the traffic burden by $TB_{m,N} = \frac{Max(DV_p)}{Link\ speed}$.

Based on the algorithm 1, we can calculate the traffic burden $TB_{m,N}$ of a co-multicast m whether N is a source ToR or a destination ToR. After calculating the $TB_{m,N}$ of all the ToRs, we formulate the width W of this co-multicast m as:

$$W_m = Max(MaxTB_{m,S}, MaxTB_{m,D}), \tag{1}$$

where S and D stand for all the source and destination ToRs in DCN. For example, in figure 3, the W_1, W_2, W_3 are 3, 4, 5 respectively. To minimize the average completion time of co-multicast requests, the co-multicast with smaller W is assigned a higher priority.

After obtaining the width W of each co-multicast request, the co-multicast scheduling problem can be regarded as a variant of graph coloring problem [26]. Each multicast request can be represented as a vertex v in $G(v,e)$. In OEHadoop, when an OCS optical multicast tree is built based on passive splitters, the multicast requests with different source or destinations cannot be transmitted in the same OCS multicast tree. To express this restriction, we add edges e in the $G(v,e)$ between two v if the sources of the two multicast requests are the same ToR or their destination sets have intersection. As we know, in the vertex coloring problem, the two vertexes with an adjacent edge cannot have the same color. With this condition, after coloring the graph, the v with the same color will have no conflict in optical layer transmission.

The details of the co-multicast scheduling algorithm are shown in algorithm 2. Although the algorithm cannot achieve optimal solution, the algorithm significantly reduces the complexity of the co-multicast scheduling problem and it outperforms other state-of-the-art methods (details in section V).

Algorithm 2 Co-Multicast Requests Scheduling

Input:

The <job ID, reducer IP, destination IP, output size> of current co-multicast requests.

The number of optical transceivers t in each ToR. The number of splitters z.

Output:

Scheduling results of multicast requests

Steps:

1. Initialize the $G(v,e)$, and the number of optical transceivers t. Calculate W of all co-multicast requests.
2. Sort the W and find the co-multicast m with smallest W if there are any unallocated multicast requests in m.
3. Choose a color to draw the $G(v,e)$ if v is a member of m until all the v of m are colored or no more v in m is available to be colored.
4. Record the colored v into a result set R one by one. If the number of v in R is equal to z, submit R as the scheduling result and break the algorithm.
5. Remove the colored v and all their adjacent v in $G(v,e)$. If there are still any v existing in G, change m to the co-multicast with next smallest W and go to step 3, otherwise go to step 6.
6. $t = t - 1$, refresh the $G(v,e)$. Remove all the recorded v of R from $G(v,e)$.
7. Go to step 2 until t is 0.
8. Submit R as the scheduling result.

3) TRIGGER CONDITION OF THE SCHEDULING ALGORITHM

Before we deploy the algorithm into an online network system, the trigger condition of the scheduling algorithm is still needed to be further considered. Because of the control overhead and the delay of reconfiguring the OSS, frequently changing the current transmitting requests will lead to bad performance. Besides, once we act our scheduling algorithm as soon as a new multicast request arrives, there will be an inevitable problem. If a new arrival multicast request belongs to a new job, because the new co-multicast consists of only one multicast request, the new arrival co-multicast request will have great possibility to have small W and to be firstly scheduled into optical switching system. As a result, we design following principles for our scheduling algorithm:

1) To save optical resources, if there are available optical resources for a new arrival multicast request,

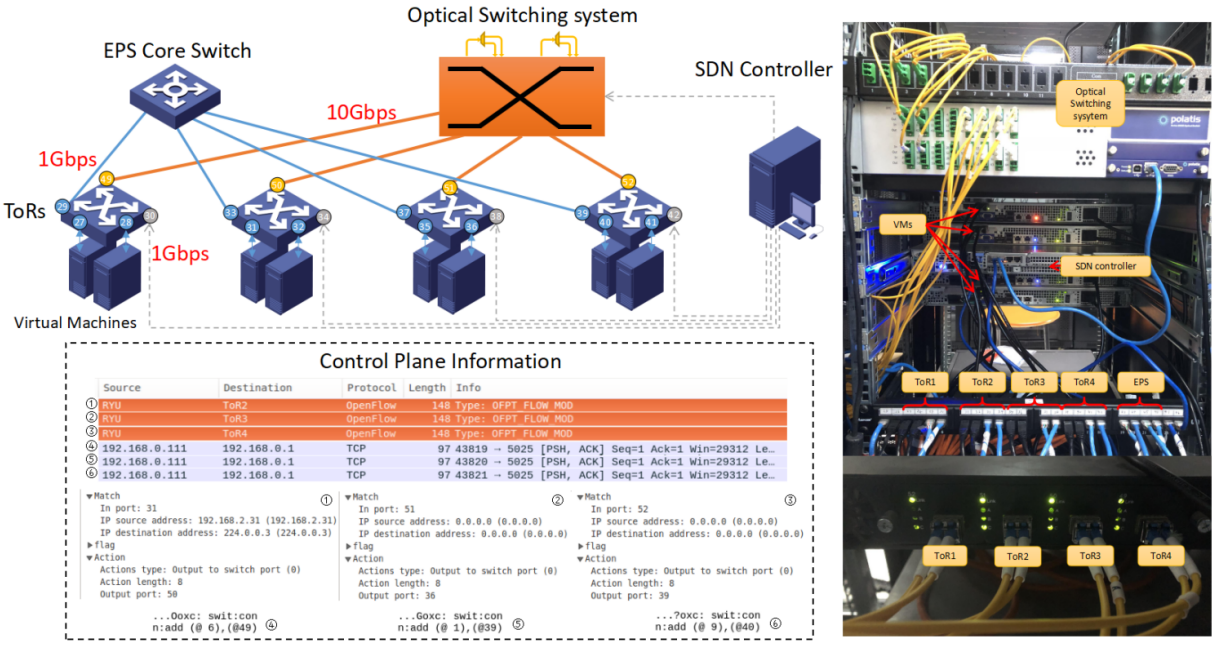


FIGURE 4. OEHadoop prototype architecture and control information.

the request will be immediately scheduled into the optical switching system.

2) Knowing the number of the reducers in a MapReduce job, the co-multicast request will not be taken into account unless all the multicast information of this co-multicast request has been received.
3) When the information of a new co-multicast is entirely collected, the scheduling algorithm is triggered. Before scheduling, the controller needs to firstly update the remaining traffic volume of the last allocated multicast requests. The already transmitted traffic volume can be easily obtained in the application layer.
4) If there are no new arrival co-multicast requests, the algorithm will not be triggered. Once there are idle optical resources in the optical switching system, the SDN controller will firstly schedule the available multicast requests which belong to co-multicasts with smaller W.

IV. EXPERIMENT DEMONSTRATION

To verify the feasibility of our OEHadoop, we built a small prototype of OEHadoop. As shown in figure 4, our OEHadoop platform consisted of 4 ToRs, where the ToRs were simulated by generating virtual network bridge in a Pica8 SDN switch and each ToR was connected to 2 nodes. Each node was a virtual machine (VM) generated by Dell PowerEdge R220 server. All the nodes were equipped with a virtual dual-core Xeon E3-1220 v3 CPU, 2GB memory, and a 1GbE physical link connected to the SDN switch. We used RYU [33] as our SDN controller which was installed on a server with Intel Core i7-6700 CPU and 16G RAM. The optical multicast system was built by a Polatis OSS and 2 1:2 passive optical splitters. The operating system in each VM was CentOS 6 kernel 2.6.32. Figure 4 demonstrates the details of our small scale OEHadoop.

The workload was generated based on the data set FB-2009 in the SWIM project [27], [28], which was a job trace gathered from the historical Hadoop trace on a 600-machine cluster at Facebook. The data set recorded the total traffic volume of the shuffle stage and the replication stage in each MapReduce job. We extracted 10 jobs in FB-2009 as the workload of our experiment (job 1671 $\sim$ job 1680). To simulate the network traffic of real MapReduce jobs, in our experiment, each job consisted of several randomly placed reducer tasks, which firstly fetched the shuffle traffic from some random nodes and then replicated the data blocks to 2 random remote nodes. The total traffic volume of each stage was determined by FB-2009. All the flows in our experiment were generated by Iperf 2.0.5 [29].

If there was a reducer ready to replicate its data to HDFS, the reducer submitted the multicast request through the north-bound interface of the SDN controller. According to the co-multicast scheduling algorithm, the controller would allocate a unique multicast group IP address for each multicast request if the request would be scheduled to the optical switching system. Then, the controller would reconfigure the OSS and note the reducer to start its replication stage. We realized unidirectional multicast transmission by the up-mode of the Pica8 SDN switch. The completion time of a job was defined as the duration between the submission time of the job and the latest complete time among all the replication stages in this job.

We firstly evaluated the processing latency of calculating a scheduling result. In our controller, we realized our

scheduling algorithm. We assumed that the DCN consisted of 150 ToRs and each ToR was equipped with three optical transceivers. We randomly generated 100 co-multicast requests as the existing requests in the system and then submitted a new integrated co-multicast request and evaluated the processing latency. Table 1 shows the delay of each process from submitting the request to starting the transmission. As we can see that our OEHadoop prototype could handle a co-multicast request with latency about 37ms.

TABLE 1. Time consumption of control plane.

Procedure	Time consumption
Network communication	3.25ms
Control algorithm	14.32ms
OSS configuration	19.51ms
Total	37.08ms

Then we compared our OEHadoop with native Hadoop which used the pipeline model to replicate date without optical communication. Because the optical network resources in our small scale prototype were rare, we allowed that the waiting multicast requests which belonged to the co-multicasts with higher priority could be transmitted in the EPS network if the total number of current waiting multicast requests exceeded 5. In this experiment, the result indicated that the average completion time of the 10 jobs in native Hadoop was 472.46s, while the average completion time of them in OEHadoop was 320.23s. By co-designing Hadoop with optical network, our OEHadoop performed much better than native Hadoop in terms of average jobs completion time. In the bottom of Figure 4, we list the control information collected from the controller during building an optical multicast tree. After the accurate configuration, the multicast traffic was transmitted from its source server (192.168.2.31) to the port 50 and was copied to two replicas by the optical switching system. The replicas finally arrived at their destinations ToRs through the ports 52 and 51 respectively.

V. SIMULATION

In this section we evaluate the performance of OEHadoop in a large scale cluster by simulation. Our simulation includes two parts. In order to prove the superiority of our co-multicast scheduling algorithm, we firstly compare our scheduling algorithm with related state-of-the-art optical multicast approaches. The traffic demand involves a set of single multicast flows and two sets of co-multicast flows. Secondly we demonstrate the performance of OEHadoop by testing the MapReduce jobs completion time under a job trace which is gathered from Facebook's production cluster and published in the SWIM project [27], [28].

A. PERFORMANCE OF CO-MULTICAST SCHEDULING ALGORITHM

1) SCHEDULING ALGORITHM IN RELATED WORKS

The control algorithm deployed in Blast [14] is based on greedy heuristic. In Blast, the multicast groups are sorted according to the scoring function s = volume/#rack, where the volume stands for the traffic volume of a multicast request and the #rack stands for the number of ToRs involved. This score balances the group size and the traffic volume to ensure that the optical ports are occupied by the most profitable multicast groups. The algorithm in controller iteratively selects multicast request with the highest scores until no more optical ports can be used. The research named optical multicast system (OMS) [15] uses greedy algorithm which iteratively selects the multicast requests with the maximum values of traffic and checks whether the request can be scheduled under the limitation of available optical splitters and optical ports in ToRs. This algorithm tries to maximize the multicast traffic offloaded by the OMS. In HERO [16], the controller implements greedy optical link assignment algorithm to firstly find free optical resources when a new multicast request arrives.

All of the above methods evaluate the performance of the algorithm by transmitting a group of multicast requests based on their network system.

2) SIMULATION SETTING

In this scenario, we build a simulation including 100 ToRs, each with 40 servers. Each ToR equips a central number of optical transceivers connected with the OSS, and there are 25 1:4 passive optical splitters connected with the OSS. In the simulation, we take the 19.5 ms OSS reconfiguration delay and the 14.3 ms control algorithm delay into account. The link speed in our simulation is 10 Gbps. To fairly compare the performance of different algorithms, in this simulation, we do not connect the ToRs with EPS network, as the model described in [14] and [15].

We evaluate the performance of our scheduling algorithm under three different kinds of traffic demands. The first group of traffic demand is consisted of a group of independence multicast flows. The traffic volume of each multicast flow follows a uniform distribution from 500 Mbits to 2.5Gbits. The source and destinations are randomly chosen, and the number of the destinations of each multicast flow is determined by a uniform distribution from 2 to 10 (a tenth of the total number of ToRs, which is similar to the traffic generating method in [15]). This set of traffic demand aims to compare the performance of different scheduling algorithms under normal multicast requests. The second group of traffic demand is generated by co-multicast requests, which is more close to the real traffic demand in big data applications. The count of multicast flows in a co-multicast is randomly generated by a uniform distribution from 2 to 10. In the third group, on the base of the second group, the number of the destinations of each multicast flow is adjusted to 3, which is similar to the traffic pattern in the replication stage of MapReduce jobs.

3) SIMULATION RESULTS

We use average multicast flow completion time and average co-multicast completion time as the evaluation indexes in our simulation. By adjusting the total number of multicast requests and the optical transceiver ports in each ToR, we obtain the simulation results which are shown in figure 5. All the results in our simulation are obtained

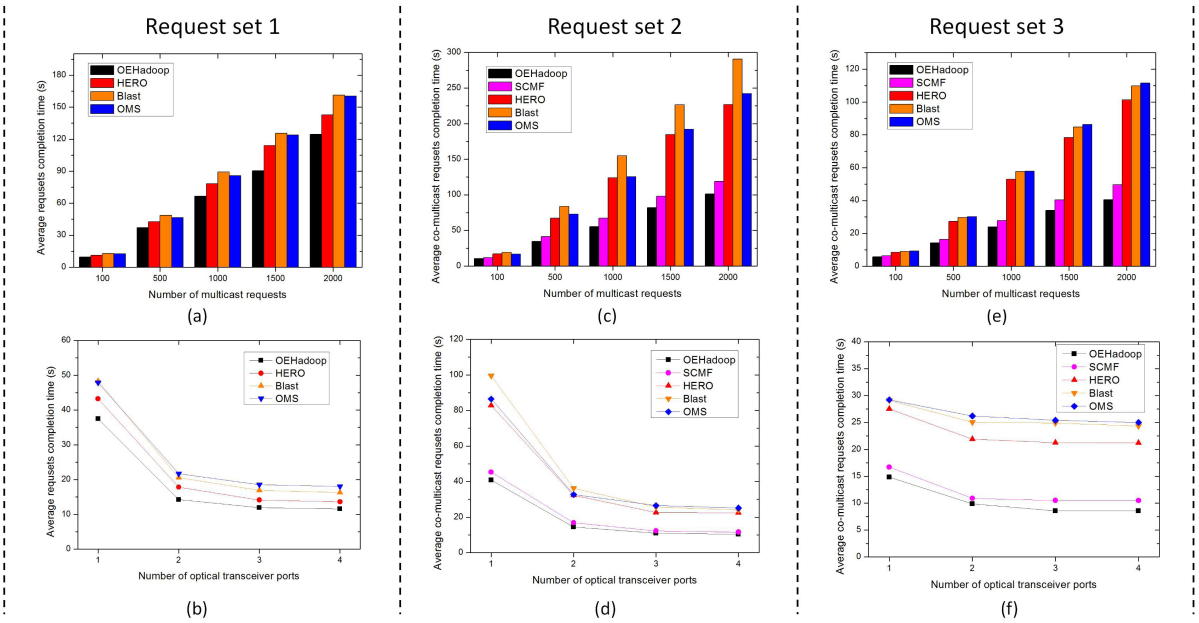


FIGURE 5. Simulation results: (a) and (b) are the results when the traffic demand consists of independent multicast requests. (a) shows the average requests completion time of different scheduling methods under different number of multicast requests. (b) expresses the average completion time when the number of optical transceiver ports is different. (c)(d) show the result when the traffic demand consists of a group of co-multicast requests and (e)(f) show the results when the number of the destinations of each multicast request in co-multicast requests is 3.

by averaging the results of 50 time runs. As we can see in figure 5, OEHadoop outperforms all state-of-the-art scheduling methods under any data sets and any situations. Figure 5 (a) proves that our scheduling method can reduce the average completion time when the requests are consisted of independence multicast flows. As shown in figure 5 (c) (e), compared with the request set 1, the improvement of our method is much more remarkable under request sets 2 and 3. That is because the previous scheduling algorithms ignore the relationship between the multicast requests in the real applications. Besides, we can find that with the increase of the amount of the multicast requests in the system, the performance of OEHadoop turns better compared with other methods. Although there is no existing method to schedule co-multicast requests based on the small co-multicast first (SCMF), we realize it as a baseline in our simulation. Under the request set 2 and 3, the performance of SCMF is much better than previous methods, but it is worse than our OEHadoop because it fails to take into account the characteristics of co-multicast requests under the constraints of the optical transceivers. The influence caused by the number of optical transceiver ports is shown in figure 5 (b) (d) (f), which is obtained under 500 multicast requests. With the increase of the number of optical ports, the bottleneck of the optical switching resources is gradually changed from the number of optical transceiver ports to the number of passive optical splitters. However, the best result is still determined by the scheduling algorithm. Obviously we can find that our OEHadoop outperforms other methods under different ports number.

B. PERFORMANCE OF MapReduce JOBS IN OEHadoop

The completion time of applications is critical to evaluate the performance of DCN. However, previous methods [14]–[16] only discuss the completion time of multicast requests. So in this section we firstly take the complete time of MapReduce jobs into consideration.

1) SIMULATION SETTING

In our simulation, we did not use the existing Hadoop simulators such as [30] or [31] because these simulators mainly aim to evaluate resource management or job scheduling, and they are unable to simulate the network traffic of applications. The ns-2 based MRPerf [32] is a packet-level simulator, which leads to long simulation time in a large-scale DCN.

In order to capture the link status during running MapReduce jobs, we build a flow-level simulation, which is similar to the HFlowSim discussed in [13]. The simulation is based on discrete time ticks, where each tick stands for 1 ms of running time. In the EPS architecture, we updated the throughput of flows following the additive-increase/multiplicative-decrease (AIMD) rule, which is a widely deployed protocol of TCP traffic. Based on this rule, the speed of a flow will be increased by 1MB/s if all the links it transmitting through has 1 MB more residual bandwidth, while the speed of the flow will decrease by half if any link of its path has no more available bandwidth. The links in optical switching system will never congest because all the multicasts will be scheduled by the line rate of optical paths.

To make our simulation more closed to the situation in real commercial DCN, we use the data set FB-2009 in SWIM

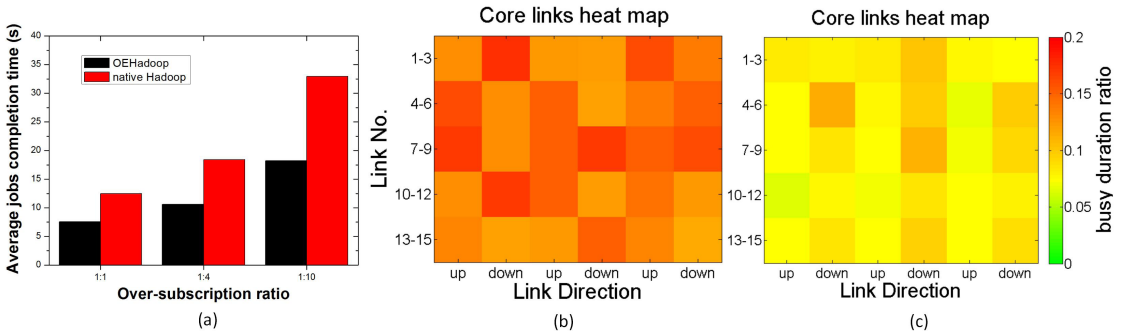


FIGURE 6. (a) shows the average completion time of jobs in the OEHadoop and native Hadoop respectively under different over-subscription ratios. (b)(c) express the busy duration ratio of each ToR to core links under 1:4 over-subscription ratio, where (b) shows the result of native Hadoop and (c) is the result of the OEHadoop. We define a unidirectional link as busy if the occupied bandwidth of the link exceeds 80% of its capacity. The busy duration ratio of the unidirectional link stands for the proportion of busy duration in total running time.

project. In our simulation, we ignore the import stage of a MapReduce job because the locality of scheduling mapper tasks usually leads to little network traffic. We model the traffic demand of each job by the same way in our experiment. Each job consists of a set of reduce tasks, and each task has two stages: shuffle and replication. In the shuffle stage, the reducers fetch data from several randomly chosen servers in DCN. When a reducer finishes its shuffle stage, it will immediately start its replication stage. In the replication stage, the reducers randomly choose 3 servers and submit a multicast request to the controller. When a reducer receives the scheduling result, the reducer will start its transmission. According to the real tasks placement method in Hadoop system [5], in our simulation, one server can only host one reduce job at a time. All the traffic volume of the flows in a job is generated based on randomly separating the recorded total network traffic volume of the job in FB-2009. We use the first 1000 jobs in FB-2009 including 1 hour and 15 minutes running duration. The submission time of each job is determined by the records in FB-2009, and the completion time of each job is defined as the time duration between the submission time of a job and the finishing time of the last replication stage in this job.

To ensure that the DCN is capable of carrying the computing and network demand of FB-2009, we build a same scale simulation including 600 servers. The network architecture is similar to figure 2. The link speed from servers to ToRs is 1Gbps, while the link speed from ToRs to core switch is 10Gbps. We obtain the simulation results under different over-subscription ratios. To adjust the ratio, maintaining the number of servers, we change the number of ToRs and the number of servers connected to ToRs. For example, when the over-subscription ratio is 1:4, there will be 40 servers connected to each ToR, while the number of ToRs will be 15. We assume that there are 3 optical transceivers in each ToR and there are 50 1:4 passive optical splitters connected to the OSS.

2) SIMULATION RESULTS

We compare our OEHadoop with native Hadoop, which uses the pipeline traffic model to replicate data to destination nodes. Figure 6 shows the results of simulation. As we can see in figure 6 (a), our OEHadoop achieves better performance than native Hadoop along with the increase of over-subscription ratio. Under the over-subscription ratio 1:10 which is a universal ratio deployed in real DCN, our OEHadoop can achieve up to x1.97 faster than native Hadoop. Figure 6 also shows the busy duration ratio of the links between ToRs and core switch in the EPS network under the over-subscription ratio 1:4, where (b) expresses the native Hadoop and (c) expresses the OEHadoop. In the figure 6 (b)(c) we can find that by offloading the replication traffic into optical switching system, OEHadoop reduces the network congestion in the EPS network and accelerates the transmission of both the shuffle traffic and the replication traffic. Our co-multicast scheduling algorithm also plays an importance role in our OEHadoop.

VI. CONCLUSIONS

In this paper, we propose OEHadoop, a modified Hadoop which is built by co-designing the application layer and the network layer to accelerate the MapReduce jobs. On the basis of offloading the long-live large-size replication traffic in to the optical switching system by optical multicast, we design a specific hybrid optical and electrical data center architecture for OEHadoop. We implement an SDN control plane, a message exchange system, and a co-multicast scheduling algorithm in OEHadoop. A small scale prototype is realized in our experiment to prove the feasibility of OEHadoop. Through simulation, we demonstrate that not only the scheduling algorithm of OEHadoop can achieve better performance than state-of-the-art methods, but also our OEHadoop can accelerate MapReduce jobs compared with native Hadoop.

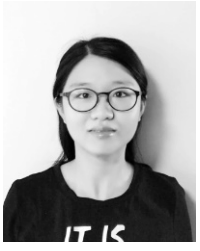
REFERENCES

[1] N. Farrington and A. Andreyev, "Facebook's data center network architecture," in *Proc. IEEE Opt. Interconnects Conf.*, May 2013, pp. 49–50.
[2] A. Singh *et al.*, "Jupiter rising: A decade of clos topologies and centralized control in Google's datacenter network," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 183–197, 2015.
[3] K. Shvachko *et al.*, "The hadoop distributed file system," in *Proc. IEEE 26th Symp. Mass Storage Syst. Technol. (MSST)*, 2010.

[4] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
[5] V. K. Vavilapalli, "Apache Hadoop YARN: Yet another resource negotiator," in *Proc. 4th Annu. Symp. Cloud Comput.*, 2013, Art. no. 5.
[6] T. Benson, A. Akella, and D. A. Maltz, "Network traffic characteristics of data centers in the wild," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas.*, Nov. 2010, pp. 267–280.
[7] D. Kilper, K. Bergman, V. W. Chan, I. Monga, G. Porter, and K. Rauschenbach, "Optical networks come of age," *Opt. Photon. News*, vol. 25, no. 9, pp. 50–57, 2014.
[8] K.-H. Lee, Y.-J. Lee, H. Choi, Y. D. Chung, and B. Moon, "Parallel data processing with MapReduce: A survey," *ACM SIGMOD Rec.*, vol. 40, no. 4, pp. 11–20, Dec. 2011.
[9] N. Farrington *et al.*, "Helios: A hybrid electrical/optical switch architecture for modular data centers," in *Proc. SIGCOMM*, Sep. 2010, pp. 339–350.
[10] G. Wang *et al.*, "c-Through: Part-time optics in data centers," in *Proc. SIGCOMM*, 2010, pp. 327–338.
[11] A. Singla, A. Singh, K. Ramachandran, L. Xu, and Y. Zhang, "Proteus: A topology malleable data center network," in *Proc. 9th ACM SIGCOMM Workshop Hot Topics Netw. (HotNets)*, 2010, Art. no. 8.
[12] Z. Zhu, S. Zhong, L. Chen, and K. Chen, "Fully programmable and scalable optical switching fabric for petabyte data center," *Opt. Exp.*, vol. 23, no. 3, pp. 3563–3580, 2015.
[13] J. Wu and B. Hong, "Multicast-based replication for Hadoop HDFS," in *Proc. 16th IEEE/ACIS Int. Conf. IEEE Softw. Eng., Artif. Intell., Netw. Parallel/Distrib. Comput. (SNPD)*, 2015, pp. 1–6.
[14] Y. Xia, T. E. Ng, and X. S. Sun, "Blast: Accelerating high-performance data analytics applications by optical multicast," in *Proc. IEEE Conf. Comput. Commun. (INFOCOM)*, May 2015, pp. 1930–1938.
[15] P. Samadi, V. Gupta, J. Xu, H. Wang, G. Zussman, and K. Bergman, "Optical multicast system for data center networks," *Opt. Exp.*, vol. 23, no. 17, pp. 22162–22180, 2015.
[16] J. Bao, B. Zhao, D. Dong, and Z. Gong, "HERO: A hybrid electrical and optical multicast for accelerating high-performance data center applications," in *Proc. SIGCOMM Posters Demos*, 2017, pp. 17–18.
[17] H. Wang, Y. Xia, K. Bergman, T. S. E. Ng, and S. Sahu, "Rethinking the physical layer of data center networks of the next decade: Using optics to enable efficient *-cast connectivity," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 3, pp. 52–58, 2013.
[18] J. Lee *et al.*, "Application-driven bandwidth guarantees in datacenters," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 467–478, 2014.
[19] D. Kreutz, F. M. V. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig, "Software-defined networking: A comprehensive survey," *Proc. IEEE*, vol. 103, no. 1, pp. 14–76, Jan. 2015.
[20] Y. Peng, K. Chen, G. Wang, W. Bai, Z. Ma, and L. Gu, "Hadoopwatch: A first step towards comprehensive traffic forecasting in cloud computing," in *Proc. IEEE INFOCOM*, May 2014, pp. 19–27.
[21] S. A. Weil, S. A. Brandt, E. L. Miller, and C. Maltzahn "CRUSH: Controlled, scalable, decentralized placement of replicated data," in *Proc. ACM/IEEE Conf. Supercomput.*, Nov. 2006, Art. no. 122.
[22] B. Adamson, C. Bormann, M. Handley, and J. Macker, *NACK-Oriented Reliable Multicast (NORM) Transport Protocol*, document RFC 5740, 2009.
[23] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. 11th ACM Workshop Hot Topics Netw.*, 2012, pp. 31–36.
[24] M. Alizadeh *et al.*, "pFabric: Minimal near-optimal datacenter transport," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 435–446, 2013.
[25] C.-Y. Hong, M. Caesar, and P. Godfrey, "Finishing flows quickly with preemptive scheduling," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 42, no. 4, pp. 127–138, 2012.
[26] T. R. Jensen and B. Toft, *Graph Coloring Problems*. Hoboken, NJ, USA: Wiley, 1995.
[27] Y. Chen, A. Ganapathi, R. Griffith, and R. Katz, "The case for evaluating mapreduce performance using workload suites," in *Proc. IEEE 19th Annu. Int. Symp. Modelling, Anal., Simulation Comput. Telecommun. Syst.*, Jul. 2011, pp. 390–399.
[28] Y. Chen, S. Alspaugh, and R. Katz, "Interactive analytical processing in big data systems: A cross-industry study of mapreduce workloads," *Proc. VLDB Endowment*, vol. 5, no. 12, pp. 1802–1813, 2012.
[29] A. Tirumala, T. Dunigan, and L. Cottrell, "Measuring end-to-end bandwidth with Iperf using Web100," Tech. Rep. SLAC-PUB-9733, 2003.
[30] S. Hammoud, M. Li, Y. Liu, N. K. Alham, and Z. Liu, "MRSim: A discrete event based MapReduce simulator," in *Proc. 7th Int. Conf. Fuzzy Syst. Knowl. Discovery*, vol. 6. Aug. 2010, pp. 2993–2997.
[31] Y. Liu, M. Li, N. K. Alham, and S. Hammoud, "Hsim: A mapreduce simulator in enabling cloud computing," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 300–308, 2013.
[32] G. Wang, A. R. Butt, P. Pandey, and K. Gupta, "A simulation approach to evaluating design decisions in mapreduce setups," in *Proc. IEEE Int. Symp. Modeling, Anal, Simulation Comput. Telecommun. Syst.*, Sep. 2009, pp. 1–11.
[33] R. Kubo *et al.*, "Ryu SDN framework: Opensource SDN platform software," NTT Tech. Rev. 12.8, 2014.
[34] M. Zaharia, D. Borthakur, J. S. Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, "Delay Scheduling: A simple technique for achieving locality and fairness in cluster scheduling," in *Proc. EuroSys*, 2010, pp. 265–278.
[35] N. McKeown *et al.*, "Openflow: Enabling innovation in campus networks," *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 2, pp. 69–74, 2008.
[36] P. Samadi, D. Calhoun, H. Wang, and K. Bergman, "Accelerating cast traffic delivery in data centers leveraging physical layer optics and SDN," in *Proc. Int. Conf. Opt. Netw. Design Modeling*, May 2014, pp. 73–77.
[37] H. Wang, C. Chen, K. Sripanidkulchai, S. Sahu, and K. Bergman, "Dynamically reconfigurable photonic resources for optically connected data center networks," in *Proc. OFC/NFOEC*, 2012, paper OTu1B-2.
[38] J. Deng *et al.*, "Keddah: Capturing Hadoop network behaviour," in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2017, pp. 2143–2150.

YINAN TANG received the bachelor's degree in telecommunication engineering from the Beijing University of Posts and Telecommunications (BUPT) in 2015. He is currently with the Institute of Information Photonics and Optical Communications, BUPT. His current research interests include optical interconnection in data center network, software-defined networking control plane, data center network, distribution computing system, and optical interconnection topology reconstruction in data center network.

HONGXIANG GUO received the B.E. and Ph.D. degrees in electrical engineering from the Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2000 and 2005, respectively. From 2005 to 2008, he was a Post-Doctoral Researcher with KDDI R&D Laboratories, Inc., Saitama, Japan, where he was involved in research on intelligent all-optical networks and their control and management technologies. He is currently with BUPT. His current research interests include the designing and modeling of photonic networks.

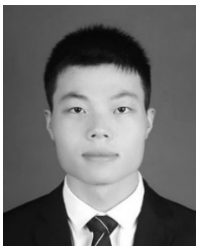


TONGTONG YUAN received the bachelor's degree in telecommunication engineering from the Beijing University of Posts and Telecommunications, Beijing, China, in 2015, where she is currently pursuing the Ph.D. degree. Her research interests include image retrieval, image hashing, and computer vision.

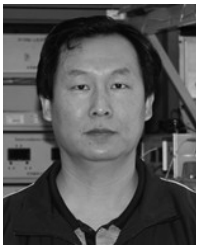
QI WU received the B.S. degree in communication engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2015. She is currently pursuing the master's degree in electronic and communication engineering with the Beijing University of Posts and Telecommunications, Beijing, China. His research interests include software-defined networking and data center optical network.

XIANG LI received the bachelor's degree in communications engineering from the Chongqing University of Posts and telecommunications in 2016. She is currently pursuing the master's degree in electronics and communications engineering with the Beijing University of Posts and Telecommunications. Her research interests include software-defined optical networks.

CEN WANG received the bachelor's degree in telecommunication engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2014. He is currently pursuing the Ph.D. degree with the Beijing University of Posts and Telecommunication. He has the internship experience with KDDI Research Laboratories, Inc., Japan. His research interests include optical interconnection, AI-assisted network resource scheduling, and software-defined networking control.



XIONG GAO received the B.S. degree in communication engineering from the Chongqing University of Posts and Telecommunications, Chongqing, China, in 2016. He is currently pursuing the Ph.D. degree in electronic science and technology with the Beijing University of Posts and Telecommunications (BUPT), Beijing, China. His research interests include routing algorithm in complex network and data center optical network.



JIAN WU (M'99) received the B.S. degree in optoelectronics from the Beijing Institute of Technology, Beijing, China, in 1995, and the Ph.D. degree in physical electronics from Tsinghua University, Beijing, in 1999. From 1999 to 2001, he was a Post-Doctoral Fellow with the Optical Communication Center, Beijing University of Posts and Telecommunications (BUPT), Beijing, where he was involved in the area of high-speed optical networks and all-optical signal processing. He is currently a Professor with the State Key Laboratory of Information Photonics and Optical Communications, BUPT. His research interests include optical packet/burst switching networks, network architecture and simulation, all-optical signal processing, optical time-domain multiplexing, high-speed optical transmission systems, nonlinear fiber optics, and high-speed optoelectronic devices.