Accelerating Collective Communications with Mutual Benefits of Optical Rackless DC and In-Network Computing

Weichi Wu[1], Xiaoliang Chen[1], Zhihuang Ma[1], Xuexia Xie[1], Ke Meng[2], Weiguang Wang[2], and Zuqing Zhu[1]

1. University of Science and Technology of China, Hefei, Anhui 230027, China, Email: zqzhu@ieee.org
2. Huawei Institute of Nanjing, Nanjing, Jiangsu 210012, China, Email: mengke6@huawei.com

Abstract: We propose a novel architecture to explore the mutual benefits of optical rackless data center (ORDC) and in-network computing for accelerating collective communications. It reduces job completion time of MapReduce clusters by 27.4% to 43.3% over traditional ORDC in experiments.
OCIS codes: (060.1155) Software-defined optical networks; (060.4251) Networks, assignment and routing algorithms.

1. Introduction

Recently, jobs with collective communications start to dominate data centers (DCs) due to the rising of data-intensive applications, such as Big Data analytics (MapReduce, Spark) and large language model training. As shown in Fig. 1(a), collective communications involve a group of processes participating in synchronized computing and communication phases iteratively [1]. In a computing phase, each process finishes its share of data processing, and thus there is no or very little inter-process data exchange, while huge traffic with a predetermined pattern (broadcast, incast, all-to-all, *etc.*) can be generated in a communication phase. Hence, collective communications further shift the major bottleneck in DCs from computation to network, and push harder for network reconfigurability, especially at the rack-level. This motivated the introduction of optical rackless DC (ORDC) [2], which removes the topological rack boundaries in DCs by inserting optical circuit switches (OCS') in between servers and top-of-rack (ToR) switches, to avoid job fragmentation across racks, and adapt to dynamic traffic patterns. Fig. 1(b) shows an ORDC pod, where 8 servers connect to three ToR switches through an OCS, facilitating various rack configurations to adapt to application needs.

Although the benefits of ORDC have been explored in [2], we argue that collective communications can be supported much more efficiently after introducing in-network computing (INC) into ORDC, even only sparsely. Recent advances in P4-based programmable data plane (PDP) [3] enable packet aggregation and arithmetic operations on fields directly in PDP switches, at line-rate (100 Gbps or beyond). Therefore, instead of solely relying on servers, INC can offload certain computing phases in a collective communication to PDP-based ToR switches (namely, P-ToR switches), to accelerate them effectively [4]. Taking the collective communication of a MapReduce cluster as an example, Fig. 1(c) explains the idea of INC-based acceleration. The collective communication includes four phases: 1) *map* (the master broadcasts data to workers), 2) *local shuffle* by the workers, 3) *upload* (the workers incast their results to the master), and 4) *reduce* (the master aggregates workers' results). As the aggregation operation in the fourth phase can be realized by PDP, we can leverage INC to offload it to a P-ToR switch, which makes the fourth phase happen concurrently with the third one and accelerates the collective communication effectively. Meanwhile, since the P-ToR switch only sends the INC-aggregated results to the master, it absorbs half of the worker-to-master traffic in the third phase.

Although INC is promising for accelerating collective communications, P-ToR switches bring in additional capital expenditures (CAPEX) and operating expense (OPEX) and the duty cycle of offloadable computing phases in a collective communication restricts the utilization of INC on its P-ToR switch. This motivates us to combine ORDC with INC. Specifically, the mutual benefits of ORDC and INC for collective communications are as follows. The network reconfigurability enabled by ORDC facilitates time sharing of the INC resources on P-ToR switches, improving their utilization over time to accelerate offloadable computing phases better. The traffic absorption achieved by INC relieves network bottlenecks in each ORDC pod, shortening communication phases of related collective communications. In this work, we propose and experimentally demonstrate *P4-ORDC*, which is a novel pod-level DC architecture combining ORDC and P4-based INC. Specifically, we partially replace ToR switches in an ORDC pod with P-ToR switches

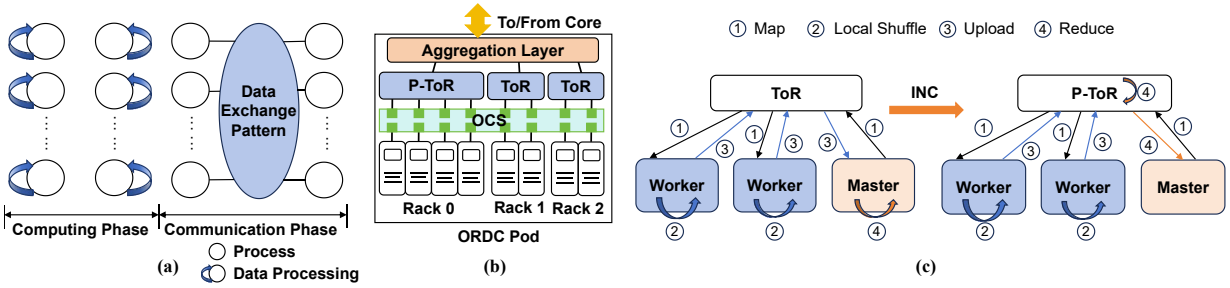

Fig. 1. Examples on (a) collective communications, (b) optical rackless DC, and (c) in-network computing.

(as exampled in Fig. 1(b)), design and implement P4-based packet processing pipelines in them to enable INC, and architect a control plane to explore the mutual benefits of ORDC and INC for accelerating collective communications. To demonstrate its performance, we prototype P4-ORDC with commercial products, build a small-scale ORDC pod, and run Hadoop MapReduce clusters in it. Experimental results show that our proposed P4-ORDC effectively accelerates the collective communications of MapReduce clusters, and significantly outperforms existing benchmarks (ORDC without INC [2] and optically-interconnected P-ToR switches [5]) on average job completion time (JCT).

2. System Design and Operation Principle

Fig. 2(a) shows the data plane of P4-ORDC. The data plane is generally based on the architecture of ORDC pod [2], except that we replace certain ToR switches in it with P-ToR switches and implement a packet encoder in each server to assist INC operations in the P-ToR switches. We leverage the data plane development kit (DPDK) [6] to develop the packet encoder for ensuring high-speed packet processing and transmitting. Specifically, this work leverages the WordCount application in Hadoop MapReduce [7], which counts the number of occurrences of each word in an input set, to demonstrate INC in P4-ORDC. As the collective communication of WordCount operates with the four phases in the left subplot of Fig. 1(c), we program the DPDK-based packet encoder to 1) intercept all the workers-to-master packets in the third phase (*upload*) and 2) encode them in the packet format shown in Fig. 2(a). Here, *JobID* is assigned by the control plane to distinguish different MapReduce clusters, *isAgg* is the flag to tell whether the packet is received from a worker (*isAgg* = 0) or it is about to be sent to the master (*isAgg* = 1), *key* denotes a word, and *value* stores the occurrences of the word. We also program the P4 pipeline in the P-ToR switch to realize INC, which makes the third and fourth phases of the collective communication happen concurrently (as shown in the right subplot of Fig. 1(c)). Specifically, the flow chart of the INC is shown in Fig. 2(a). After receiving a packet, the parser first checks its header to see whether it should by processed by INC. If yes, the subsequent P4 pipeline hashes the *JobID* and *key* in the packet to point to a register in the P-ToR switch. Then, it aggregates the *value* in the packet with the register's value, and increments the register's counter by 1. Next, it checks whether packets from all the workers have been aggregated by examining the register's counter. If yes, it updates the *isAgg* and *value* in the packet as 1 and the register's value, respectively, and sends the packet to the master. Otherwise, it waits for other packets from the workers.

The control plane of P4-ORDC is illustrated in Fig. 2(b), which plays a crucial role to leverage the mutual benefits of ORDC and INC to accelerate collective communications. As for the ToR switches, servers, and OCS, we design the ToR controller, job handler, and OCS controller to respectively manage them, through corresponding south-bound interfaces. The ToR controller collects the statistics of traffic through ToR switches (both traditional ToR and P-ToR switches) to update the traffic engineering database (TED). The servers in the P4-ORDC pod register their jobs with collective communications to the job handler, which checks the JobID database to assign a unique *JobID* to each job and forward the jobs' information (durations of their computing phases, data sizes of their communication phases, *etc.*) to the TED. The job handler also interacts with the INC scheduler to assign INC resources in P-ToR switches to offloadable computing phases. The TED analyzes the information about traffic and jobs in the data plane to abstract a 3-dimensional (3D) traffic matrix that represents the data transfers between server pairs over a series of future time periods. Then, as shown in the right subplot of Fig. 2(b), the topology engineering (TPE) module obtains a sequence of OCS configurations, each of which has been optimized to group servers into racks such that the collective communications during the corresponding future period can be accelerated the most. Finally, the OCS configurations will be applied to the OCS as planned by the OCS controller, and the TED will calculate the routing schemes for traffic based on each OCS configuration and implement them in the ToR switches through the ToR controller.

3. Experimental Demonstrations

We build a small-scale P4-ORDC pod with 8 servers, three ToR switches, and a 32×32 optical cross-connect (OXC), and the line-rate of the links to connect them is 10 Gbps. One ToR switch is a P-ToR switch based on Tofino ASICs.

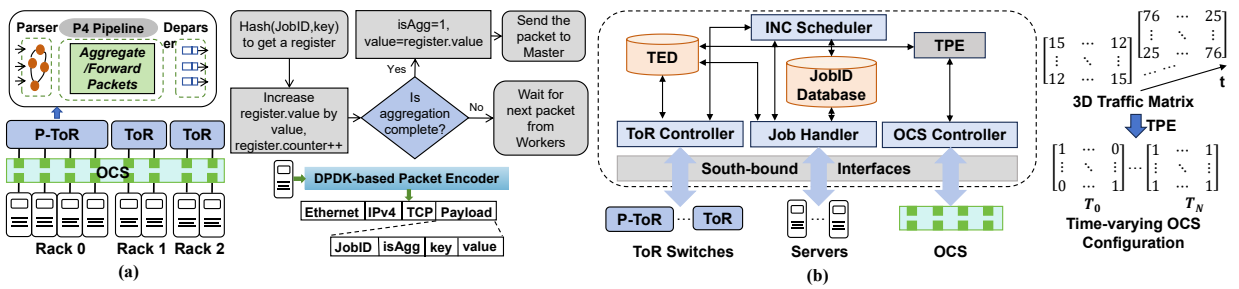

Fig. 2. System design of P4-ORDC, (a) data plane, and (b) control plane, TED: traffic engineering database, TPE: topology engineering.

The physical setup of the P4-ORDC pod is the same as that in Fig. 1(b), where the P-ToR switch connects to the OXC with four ports and each traditional ToR allocates two ports to the OXC. For convenience, we index the servers in Fig. 1(b) from left to right as *Servers* 0-7, respectively, and run WordCount jobs on them. The size of data that need to be analyzed in each job is evenly distributed within $[50, 100]$ MB. We first conduct a simple experiment to demonstrate the mutual benefits of ORDC and INC on accelerating collective communications illustratively. Specifically, we place two sets of jobs on *Servers* 0-3 and 4-7, respectively, and let P4-ORDC orchestrate the collective communications of the jobs to share INC resources on the P-ToR switch over time. Fig. 3(a) shows the jobs' traffic that goes through the P-ToR switch, indicating that the control plane schedules the *upload* phases of the two server groups alteratively over time. Specifically, the *upload* phase of *Group* 1 (*Servers* 0-3) overlaps with the *map* and *local shuffle* phases of *Group* 2 (*Servers* 4-7), and *vice versa*, and during the *upload* phase of a group, the OXC arranges all the servers in the group as a rack under the P-ToR switch to leverage the INC there for offloading the subsequent *reduce* phase simultaneously. In this experiment, the mutual benefits of ORDC and INC achieve an average JCT reduction of 66%.

We then conduct experiments to compare P4-ORDC to three existing architectures: 1) FatTree, which removes the OXC, replaces the P-ToR switch with a normal ToR switch, and interconnects the ToR switches with Ethernet switches in a fat-tree, 2) ORDC [2], the traditional ORDC without INC, and 3) P4INC-AOI [5], which interconnects P-ToR and ToR switches from the top with an OXC. For fair comparisons, the numbers of servers, ToR switches and ports on each ToR switch are kept the same in the four architectures. In each experiment, we run a number of WordCount jobs in the architectures, respectively, while the job placement on servers (randomly placed) is the same for each architecture. The results on average JCT are plotted in Fig. 3(b). Due to the lack of network reconfigurability and INC, FatTree provides the longest JCT in all the experimental scenarios, and it is followed by ORDC, justifying the necessity of network reconfigurability in serving collective communications. INC can accelerate collective communications effectively, since P4INC-AOI and P4-ORDC achieve significantly shorter JCTs than the other two, but compared with P4INC-AOI, P4-ORDC further reduces the average JCT by 23.4% to 26.4%, confirming the mutual benefits of ORDC and INC. Finally, we conduct experiments to analyze the sensitivity of P4-ORDC to job placement scenarios. This time, instead of placing workers and masters of jobs randomly, we consider two job placement scenarios: 1) worker-localized scenario, which groups as many workers as possible on each server, and 2) cluster-localized scenario, which tries to use the smallest number of servers to carry the workers and master of each cluster. Fig. 3(c) shows the numbers of jobs that are completed in P4-ORDC when jobs are placed with the two scenarios. P4-ORDC can complete more jobs when they are placed with the worker-localized scenario, because this scenario helps to maximize the utilization of INC on the P-ToR switch and thus amplifies the effectiveness of TPE. On the other hand, although the cluster-localized scenario helps to reduce inter-rack traffic in P4-ORDC, it can fragment the placement of workers over servers, making it harder to find proper OCS configurations to enable effective time sharing of INC.

4. Conclusion

We proposed P4-ORDC, which is a novel architecture to explore the mutual benefits of ORDC and INC for accelerating collective communications. Our experiments verified that P4-ORDC effectively reduces JCT of MapReduce clusters.

References

[1] C. Cameron *et al.*, "What is collective communication?" [Online]. Available: https://cvw.cac.cornell.edu/mpicc/intro/index.
[2] W. Wang, *et al.*, "RDC: Energy-efficient data center network congestion relief with topological reconfigurability at the edge," *NSDI 2022*.
[3] P. Bosshart, *et al.*, "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, pp. 87-95, Jul. 2014.
[4] A. Sapio *et al.*, "Scaling distributed machine learning with in-network aggregation," *NSDI 2021*.
[5] X. Xie *et al.*, "P4INC-AOI: When in-network computing meets all-optical interconnect for adaptive and low-latency optical DCN," *OFC 2023*.
[6] Data Plane Development Kit (DPDK). [Online]. Available: https://www.dpdk.org.
[7] MapReduce. [Online]. Available: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html.

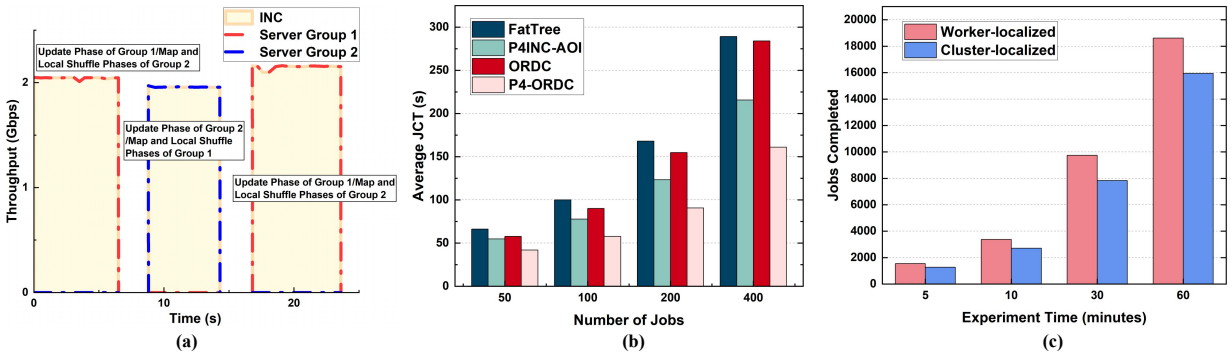

Fig. 3. Experimental results.