On Scheduling DML Jobs in All-Optical DCNs with In-Network Computing

Xiaoyan Dong, Hao Yang, Yuxiao Zhang, Xuexia Xie, and Zuqing Zhu[†]
School of Information Science and Technology, University of Science and Technology of China, Hefei, China
[†]Email:{zqzhu}@ieee.org

***Abstract*—Enabled by programmable data plane (PDP), in-network computing (INC) can offload the computation phase of distributed machine learning (DML) training to accelerate the execution of parameter servers (PS'). Meanwhile, all-optical interconnect (AOI) can effectively improve the network throughput for DML training. This work studies how to schedule a batch of parallel DML jobs in an all-optical data center network (DCN) to fully explore the mutual benefits of INC and AOI. Specifically, for each DML job, we determine where to deploy its workers, whether and where to offload/place its PS, and how to plan the routing of data transfers between the workers and the PS, such that its job completion time (JCT) is minimized. We formulate a mixed integer linear programming (MILP) model to optimize the job scheduling for the cases with and without INC, and propose a heuristic to tackle the case with INC quickly. Extensive simulations confirm the effectiveness of our proposals.**

***Index Terms*—All-optical data-center networks (DCNs), In-network computing (INC), Distributed machine learning (DML).**

I. Introduction

In the era of artificial intelligence (AI), the rapid surge in data volume and exponential growth in computing demands have bought unprecedented challenges to the traditional data center networks (DCNs) based on electrical packet switching (EPS), on energy consumption, scalability, and adaptivity [1, 2]. Therefore, optical circuit switching (OCS) has been gradually deemed as a preferable choice over EPS in DCNs due to its higher energy efficiency, larger port capacity, and shorter data transfer latency [3–8]. This leads to the amalgamation of OCS and EPS to realize hybrid optical/electrical DCNs (HOE-DCNs) [3], but the coexistence of OCS and EPS would also result in more intricate system design and maintenance. On the other hand, the major part of the fast-growing applications in DCNs (*e.g.*, distributed machine learning (DML)) are much more likely to generate elephant flows than legacy DC services [9]. Hence, the all-optical DCN that purely relies on OCS for inter-rack/pod connections can be built with a flat architecture (*e.g.*, in Leaf-Spine [10]), to serve these bandwidth-intensive applications with effectively-reduced bandwidth competition and communication delay. For instance, Fig. 1 shows a simple example on such an all-optical DCN, where the top-of-rack (ToR) switches are interconnected by an optical cross-connect (OXC) to form a Leaf-Spine-type inter-rack topology.

DML is introduced to address the situations where the training data is inherently distributed or cannot be stored on a single server [9], and it is known that all-optical DCNs can effectively accelerate DML jobs [11]. There are four typical types of DML frameworks: Parameter Server (PS), Distributed Data Parallel (DDP), Ring-AllReduce (Ring), and Peer-to-Peer (P2P), among which PS is arguably the most widely adopted one due to its superior scalability [12]. The PS framework accomplishes the training of a DML model with workers and PS', where each worker trains a local replica of the DML model with a subset of the training data, and each PS talks with specific workers to pull and push model parameters. Therefore, the communication phase of DML training in PS uses a tree-type topology, which can be easily mapped to an all-optical DCN in Leaf-Spine. To this end, a number of existing studies have considered accelerating the communication phase of DML training in PS in all-optical DCNs [11, 13].

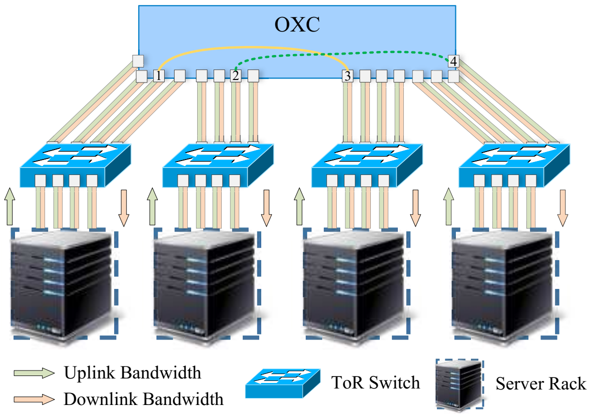


Fig. 1. Example on all-optical DCN in Leaf-Spine.

In addition to the communication phase, the computation phase of DML training in PS' can also be accelerated, especially with the in-network computing (INC) enabled by the programmable data plane (PDP) [14]. Specifically, we can offload the computing tasks of averaging model parameters on one PS to a PDP ToR switch, such that the tasks can run much faster with the hardware-based line-speed processing in the ToR switch and inter-rack traffic can be avoided if the PS and its workers are not located in the same rack. Previously, in [15], we have experimentally demonstrated that INC and all-optical interconnect (AOI) can mutually benefit each other when accelerating distributed computing jobs in tree-type clusters. More specifically, the large capacity of AOI better utilizes the INC at line-speed on high-throughput ToR switches, while the reshaping of inter-rack traffic achieved by INC helps to reduce the frequency of AOI reconfigurations.

However, the advantages of the symbiosis of INC and AOI cannot be fully explored without a carefully-designed approach to schedule DML jobs in it. Specifically, for each DML job,

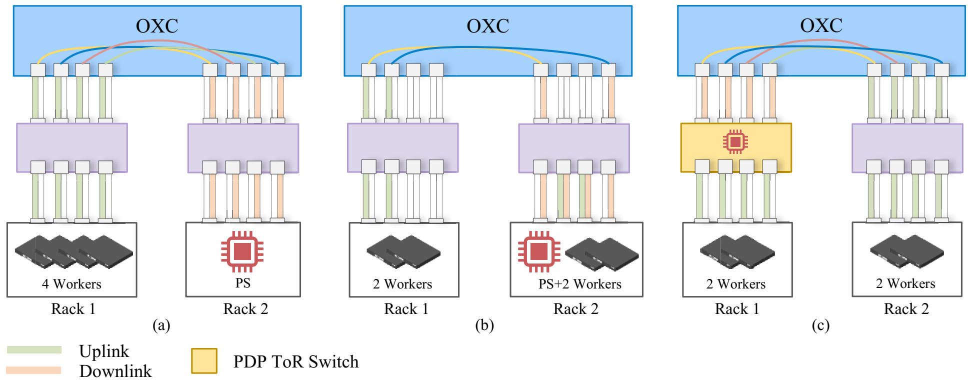


Fig. 2. Examples on scheduling one DML job with multiple racks in an all-optical DCN with INC.

the approach needs to determine where to deploy its workers, whether and where to offload/place its PS, and how to plan the routing of data transfers between the workers and the PS, such that the mutual benefits of INC and AOI can be exploited to minimize its job completion time (JCT). To the best of our knowledge, this problem has not been studied in the literature yet. In this work, we tackle it for a batch of parallel DML jobs. A mixed integer linear programming (MILP) model is first formulated to solve the problem exactly, followed by a time-efficient heuristic based on worker cluster grouping. Extensive simulations confirm the effectiveness of our proposals.

The rest of this paper is organized as follows. Section II describes the network model and formulates the MILP model. We propose the heuristic based on worker cluster grouping in Section III. The performance of our proposals is evaluated in Section IV. Finally, Section V summarizes the paper.

II. Problem Description

In this section, we first describe the network model of an all-optical DCN with INC, then formulate the MILP model to optimize the scheduling of DML jobs in it.

A. Network Model

Fig. 1 depicts the all-optical DCN considered in this work, where the servers in each rack have direct connections with their ToR switch and the ToR switches can be connected with each other through an OXC[1]. The bandwidth capacity between each ToR switch and its server pool is equal to that between the ToR switch and the OXC. The OXC consists of N_{OXC} ports, and each port can only be connected to one other port, then duplex communication can be realized through the pair of ports. Therefore, the all-optical DCN can be modeled as a non-blocking $N_{\text{OXC}} \times N_{\text{OXC}}$ OCS network [13]. There is a batch of parallel DML jobs to schedule in the all-optical DCN, and each DML job includes a PS and several workers.

To schedule the DML jobs, we first need to determine where to deploy their workers and PS'. Here, the workers can only be deployed in server pools, while a PS can be placed on either a PDP ToR switch (*i.e.*, offloaded with INC) or a server. Note that, although deploying the workers and PS of a DML job in a same rack helps to reduce both communication latency and inter-rack traffic, it might also lead to the resource fragmentation that restricts the utilization of bandwidth and IT resources in a DCN [17]. Hence, we allow the workers and the PS of a DML job to be deployed in different racks.

As synchronous training usually converges faster than asynchronous training [18], we assume that all the DML jobs use synchronous training, and do not consider special training strategies like dropout. Hence, each worker of a DML job updates all the model parameters synchronously in each iteration. In each iteration of the DML training in PS, there are four key steps: local calculation, pull, PS update, and push. As the local calculation step is mainly accomplished by workers, its calculation time can be treated as a constant for each DML job and ignored from the optimization. This is because workers can only be deployed in server pools. The duration of the PS update step depends on whether the PS is deployed on a server or a PDP ToR switch. According to our experimental results in [15], the duration in the former case is proportion to the data size of model parameters, while that in the latter case is so short that it can be approximated as 0. The duration of the pull/push step is the longest flow completion time (FCT) to transmit all the model parameters, which is determined by the bandwidth allocated to the corresponding flow. Since we consider synchronous training, the flow sizes for the pull and push steps of a DML job are equal. Therefore, we only need to optimize one of the two steps and apply the same setting to the other, *i.e.*, the two steps are treated equally in our optimization.

The examples in Fig. 2 explain the schemes that schedule a DML job in multiple racks. Here, the DML job includes 4 workers and a PS, each rack is equipped with a 4×4 ToR switch, where the uplink/downlink capacity of each port is 100 units/s, and the size of each flow in pull/push step is 500 units. The scheme in Fig. 2(a) deploys all the workers in *Rack* 1 and places the PS in *Rack* 2, and a bandwidth of 100 units/s is allocated between each worker and the PS. Then, the length of the pull/push step is 5 seconds. In Fig. 2(b), the scheme places two workers in *Rack* 1, deploys the rest of the DML job in *Rack* 2, and still allocates 100 units/s to each worker-PS pair due to the downlink capacity of each port on the ToR switch

[1]As a commercial OXC can easily scale up to a port count of $N_{\text{OXC}} = 384$ or even more [16], this work only considers one OXC in the all-optical DCN.

of *Rack* 2. The length of the pull/push step is still 5 seconds. In Fig. 2(c), we deploy the workers as in Fig. 2(b) but offload the PS to the ToR switch of *Rack* 1. Then, compared with the schemes in Figs. 2(a) and 2(b), inter-rack traffic is reduced by half and 200 units/s can be assigned to each worker-PS pair, shortening a pull/push step to 2.5 seconds. Moreover, the scheme in Fig. 2(c) also shortens the PS update step.

B. MILP Model

We formulate an MILP model to optimize the scheduling of DML jobs in an all-optical DCN exactly, and the parameters and variables are tabulated in Table I. We denote the set of all racks and those with PDP ToR switches as R_s and R_c, respectively ($R_c \subseteq R_s$). Therefore, the MILP also covers the all-optical DCNs without INC, when we have $R_c = \emptyset$.

TABLE I
PARAMETERS AND VARIABLES OF MILP

Parameters	
$\mathcal{J}$	the set of DML jobs ($i \in \mathcal{J}$).
W_i	the set of workers of *Job* i ($j \in W_i$).
R_s	the set of racks ($k \in R_s$).
R_c	the set of racks with PDP ToR switches.
B	the uplink/downlink capacity of each port on ToR switches.
N_{ToR}	the number of ToR switch ports connected to server pool/OXC.
N_k	the set of ToR switch ports connected to OXC from *Rack* k.
D	the set of data to transfer for DML jobs ($d_i \in D$).
N_{OXC}	the set of OXC ports ($n \in P$).
$S_k^{\text{CPU}}/S_k^{\text{GPU}}$	the CPU/GPU capacity of *Rack* k.
S_k^{MEM}	the memory capacity of *Rack* k.
$P_i^{\text{MEM}}/P_i^{\text{CPU}}$	the memory/CPU resources required by PS of *Job* i in server pool.
C_k	the number of PS' that the PDP ToR switch on *Rack* k can accommodate.
$W_{ij}^{\text{CPU}}/W_{ij}^{\text{GPU}}$	the CPU/GPU resources required by *Worker* j of *Job* i.
W_{ij}^{MEM}	the memory resources required by *Worker* j of *Job* i.
α	the ratio of PS update to communication time in server.
Variables	
p_{ik}^c	the binary variable that equals 1 if PS of *Job* i is on PDP ToR switch of *Rack* k, and 0 otherwise.
p_{ik}^s	the binary variable that equals 1 if PS of *Job* i is on server pool of *Rack* k, and 0 otherwise.
w_{ijk}	the binary variable that equals 1 if *Worker* j of *Job* i is on server pool of *Rack* k, and 0 otherwise.
L_{nm}	the binary variable that equals 1 if *Ports* n and m of OXC are connected, and 0 otherwise.
$b_{ijst}^{\text{w2p,s}}/b_{ijst}^{\text{w2p,c}}$	the uplink bandwidth between *Worker* j of *Job* i on *Rack* s and PS on server pool/ToR switch of *Rack* t.
$b_{ijst}^{\text{p2w,s}}/b_{ijst}^{\text{p2w,c}}$	the downlink bandwidth between *Worker* j of *Job* i on *Rack* s and PS on server pool/ToR switch of *Rack* t.
b_{ijs}^s	the bandwidth between *Worker* j of *Job* i and PS on server pool of *Rack* s.
b_{ijs}^c	the uplink bandwidth between *Worker* j of *Job* i and PS on ToR switch of *Rack* s.
f_{ij}	the inverse of JCT of *Worker* j of *Job* i.
M	a large positive constant.

Constraints:

$$\sum_{i\in\mathcal{J}} p_{ik}^c \le C_k, \quad \forall k \in R_c. \tag{1}$$

Eq. (1) ensures that the PS' offloaded to the PDP ToR switch of each rack do not exceed the switch's capacity [17].

$$\begin{cases} \sum\limits_{i\in\mathcal{J}} p_{ik}^s \cdot P_i^{\text{CPU}} + \sum\limits_{i\in\mathcal{J}}\sum\limits_{j\in W_i} w_{ijk}\cdot W_{ij}^{\text{CPU}} \le S_k^{\text{CPU}}, \\ \sum\limits_{i\in\mathcal{J}}\sum\limits_{j\in W_i} w_{ij}^k \cdot W_{ij}^{\text{GPU}} \le S_k^{\text{GPU}}, \\ \sum\limits_{i\in\mathcal{J}} p_{ik}^s \cdot P_i^{\text{MEM}} + \sum\limits_{i\in\mathcal{J}}\sum\limits_{j\in W_i} w_{ijk}\cdot W_{ij}^{\text{MEM}} \le S_k^{\text{MEM}}, \end{cases} \quad \forall k \in R_s. \tag{2}$$

Eq. (2) ensures that the deployments of workers and PS' satisfy the IT resource constraints of each rack.

$$\sum_{k\in R_s} w_{ijk} = 1, \quad \forall i \in \mathcal{J},\ j \in W_i. \tag{3}$$

Eq. (3) ensures each worker runs on one and only one rack.

$$\sum_{k\in R_s} (p_{ik}^s + p_{ik}^c) = 1, \quad \forall i \in \mathcal{J}. \tag{4}$$

Eq. (4) ensures that the PS of each job is deployed on either the server pool or PDP ToR switch of a rack.

$$\begin{cases} p_{ik}^c \le 1, & \forall i \in \mathcal{J},\ k \in R_c, \\ p_{ik}^c = 0, & \text{otherwise}. \end{cases} \tag{5}$$

Eq. (5) ensures that if a PS is offloaded to a ToR switch with INC, then the ToR switch has to be a PDP-based one.

$$\begin{cases} z_{ijst}^{\text{w2p,s}} \ge w_{ijs} + p_{it}^s - 1, \\ z_{ijst}^{\text{w2p,s}} \le \frac{1}{2}\cdot(w_{ijs} + p_{it}^s), \\ b_{ijst}^{\text{w2p,s}} \le M \cdot z_{ijst}^{\text{w2p,s}}, \end{cases} \forall i \in \mathcal{J}, j \in W_i, \{s,t \in R_s : s \ne t\}, \tag{6}$$

$$\begin{cases} z_{ijst}^{\text{w2p,c}} \ge w_{ijs} + p_{it}^c - 1, \\ z_{ijst}^{\text{w2p,c}} \le \frac{1}{2}\cdot(w_{ijs} + p_{it}^c), \\ b_{ijst}^{\text{w2p,c}} \le M \cdot z_{ijst}^{\text{w2p,c}}, \end{cases} \forall i \in \mathcal{J}, j \in W_i, \{s,t \in R_s : s \ne t\}. \tag{7}$$

Eqs. (6) and (7) ensure that the values of variables $\{b_{ijst}^{\text{w2p,s}}\}$ and $\{b_{ijst}^{\text{w2p,c}}\}$ are correctly set, where $z_{ijst}^{\text{w2p,s}}$ and $z_{ijst}^{\text{w2p,c}}$ are the auxiliary variables used for linearization.

$$\begin{cases} z_{ijk}^s \ge w_{ijk} + p_{ik}^s - 1, \\ z_{ijk}^s \le \frac{1}{2}\cdot(w_{ijk} + p_{ik}^s), \\ b_{ijk}^s \le M \cdot z_{ijk}^s, \end{cases} \forall i \in \mathcal{J}, j \in W_i, k \in R_s, \tag{8}$$

$$\begin{cases} z_{ijk}^c \ge w_{ijk} + p_{ik}^c - 1, \\ z_{ijk}^c \le \frac{1}{2}\cdot(w_{ijk} + p_{ik}^c), \\ b_{ijk}^c \le M \cdot z_{ijk}^c, \end{cases} \forall i \in N, j \in W_i, k \in R_s. \tag{9}$$

Eqs. (8) and (9) ensure that the values of variables $\{b_{ijk}^s\}$ and $\{b_{ijk}^c\}$ are correctly set, where z_{ijk}^c and z_{ijk}^c are the auxiliary variables used for linearization.

$$\sum_{t\in R_s}\sum_{i\in\mathcal{J}}\sum_{j\in W_i}\left(b_{ijst}^{\text{w2p,s}} + b_{ijst}^{\text{w2p,c}}\right) + \sum_{i\in\mathcal{J}}\sum_{j\in W_i}\left(b_{ijs}^c + b_{ijs}^s\right) \le N_{\text{ToR}}\cdot B, \quad \forall s \in R_s. \tag{10}$$

$$\sum_{t\in R_s}\sum_{i\in\mathcal{J}}\sum_{j\in W_i}\left(b_{ijst}^{\text{p2w,s}} + b_{ijst}^{\text{p2w,c}}\right) \le N_{\text{ToR}}\cdot B, \quad \forall s \in R_s. \tag{11}$$

$$\sum_{t\in R_s}\sum_{i\in\mathcal{J}}\sum_{j\in W_i} b_{ijst}^{\text{p2w,s}} + \sum_{i\in\mathcal{J}}\sum_{j\in W_i} b_{ijs}^s \le N_{\text{ToR}}\cdot B, \quad \forall s \in R_s. \tag{12}$$

Eqs. (10)-(12) ensure that the total uplink/downlink bandwidth allocated to the workers and PS' running on each rack satisfies the bandwidth constraints of the rack.

$$\begin{cases} L_{nm} = L_{mn}, \\ \sum\limits_{n \in N_{\text{OXC}}} L_{nm} \leq 1, \\ \sum\limits_{i \in \mathcal{J}} \sum\limits_{j \in W_i} \left(b^{\text{w2p,s}}_{ijst} + b^{\text{w2p,c}}_{ijst} \right) \leq \sum\limits_{n \in N_s} \sum\limits_{m \in N_t} L_{nm} \cdot B, \\ \sum\limits_{i \in \mathcal{J}} \sum\limits_{j \in W_i} \left(b^{\text{p2w,s}}_{ijst} + b^{\text{p2w,c}}_{ijst} \right) \leq \sum\limits_{n \in N_s} \sum\limits_{m \in N_t} L_{nm} \cdot B, \end{cases} \tag{13}$$
$$\{n, m \in N_{\text{OXC}} : n \neq m\}, \ \{s, t \in R_s : s \neq t\}.$$

Eq. (13) ensures that the OXC ports are connected one-to-one, and inter-rack traffic can only go through two connected OXC ports and needs to satisfy the bandwidth constraints.

$$\begin{cases} f_{ij} - \left(\frac{b^{\text{w2p,s}}_{ijst}}{2 \cdot d_i \cdot (1+\alpha)} \right) \leq M \cdot \left(1 - z^{\text{w2p,s}}_{ijst} \right), \\ \left(\frac{b^{\text{w2p,s}}_{ijst}}{2 \cdot d_i \cdot (1+\alpha)} \right) - f_{ij} \leq M \cdot \left(1 - z^{\text{w2p,s}}_{ijst} \right), \end{cases} \tag{14}$$
$$\forall i \in \mathcal{J}, \ j \in W_i, \ \{s, t \in R_s : s \neq t\},$$

$$\begin{cases} f_{ij} - \left(\frac{b^{\text{w2p,c}}_{ijst}}{2 \cdot d_i} \right) \leq M \cdot \left(1 - z^{\text{w2p,c}}_{ijst} \right), \\ \left(\frac{b^{\text{w2p,c}}_{ijst}}{2 \cdot d_i} \right) - f_{ij} \leq M \cdot \left(1 - z^{\text{w2p,c}}_{ijst} \right), \end{cases} \tag{15}$$
$$\forall i \in \mathcal{J}, \ j \in W_i, \ \{s, t \in R_s : s \neq t\}.$$

Eqs. (14) and (15) obtain f_{ij}, which is the inverse of the JCT of *Worker* j of *Job* i (after excluding the local calculation step), where α is the ratio of PS update step to communication time when the PS is placed in server pool, and it is proportional to the data size of model parameters. As we have explained before, the length of the PS update step can be treated as 0 if the PS is offloaded to a PDP ToR switch with INC [15].

$$0 \leq f \leq f_{ij} \quad \forall i \in \mathcal{J}, \ j \in W_i. \tag{16}$$

Eq. (16) gets the minimum value of $\{f_{ij}\}$ as f.

Objective:

As f is the minimum of all the inverses of JCTs, $\frac{1}{f}$ is the longest JCT. Hence, the optimization objective of minimizing the longest JCT of all the DML jobs is equivalent to

$$\text{Maximize} \quad f. \tag{17}$$

III. Algorithm Design

Although the MILP can optimize the scheduling of DML jobs in an all-optical DCN exactly, solving it can become intractable when the problem size is relatively large. Therefore, we, in this section, propose a time-efficient heuristic based on worker cluster grouping (WCG) to speed up the problem-solving, as shown in *Algorithm* 1. *Lines* 1-6 explain how to get the worker clusters of DML jobs in $\mathcal{J}$, group them, and deploy them to racks accordingly. Specifically, we put the workers of each job in a cluster to deploy in one rack, and try to shorten the longest pull/push steps of the DML jobs by distributing the clusters evenly in data transfer sizes on the racks. Then, we place the PS of each job (*Lines* 7-24), *i.e.*, we first try to offload the PS to a PDP ToR switch (*Lines* 9-15), and if this cannot be done, we deploy it on a server (*Lines* 16-23).

Algorithm 1: Heuristic Based on Worker Cluster Grouping

```
1  put workers of each DML job in $\mathcal{J}$ in a cluster;
2  sort worker clusters in descending order of data transfer sizes;
3  try to put $|R_s|$ worker clusters in one group in sorted order until
   each cluster is assigned to one group, and obtain a group set $\mathcal{G}$;
4  for each group $g \in \mathcal{G}$ do
5      utilize a resource balancing strategy to deploy each cluster
       in $g$ to a rack $k \in R_s$;
6  end
7  sort DML jobs in $\mathcal{J}$ in descending order of data transfer sizes;
8  for each job $i \in \mathcal{J}$ in sorted order do
9      sort the racks that carry workers of Job $i$ in descending
       order of worker numbers, and set $flag = 0$;
10     for each rack $k$ in sorted order do
11         try to deploy PS of Job $i$ on the ToR switch of Rack $k$;
12         if the PS is successfully deployed then
13             $flag = 1$, break;
14         end
15     end
16     if $flag = 0$ then
17         for each rack $k$ in sorted order do
18             try to deploy PS of Job $i$ on a server of Rack $k$;
19             if the PS is successfully deployed then
20                 break;
21             end
22         end
23     end
24 end
25 get the inter-rack data transfer matrix $\mathbf{D}$ according to
   deployments of workers and PS';
26 stuff $\mathbf{D}$ to obtain a perfect matrix $\mathbf{D}'$;
27 decompose $\mathbf{D}'$ into a set of permutation matrices with specific
   coefficients by Birkhoff-von Neumann decomposition;
28 determine the number of physical connections through OXC for
   each rack pair based on the permutation matrices' coefficients;
29 allocate uplink/downlink bandwidth through ToR switch/OXC
   to each worker-PS pair in proportion to its data transfer size;
30 calculate the longest JCT of all the DML jobs in $\mathcal{J}$;
```

Next, *Lines* 25-28 explain how to connect racks through the OXC. First, we denote the inter-rack data transfer sizes with matrix $\mathbf{D}$ (*Line* 25). Then, *Line* 26 stuffs $\mathbf{D}$ to get a perfect matrix $\mathbf{D}'$. According to Birkhoff-von Neumann (BvN) theorem [19], any perfect matrix[2] can be decomposed into a set of permutation matrices with specific coefficients. By leveraging the analysis in [20], we find that if physical connections through the OXC are allocated to rack pairs in proportion to the permutation matrices' coefficients, the longest duration of pull/push steps of the DML jobs can be minimized. This explains the rationale behind *Lines* 27-28. Finally, *Line* 29 allocates bandwidth for the communications between workers and their PS', and *Line* 30 obtains the longest JCT of all the DML jobs with the obtained scheduling scheme.

Fig. 3 provides an illustrative example on how to connect racks through the OXC with *Lines* 25-28. Initially, the inter-rack data transfer matrix $\mathbf{D}$ is irregular and cannot be decomposed into a set of permutation matrices. Here, a permutation matrix is a square binary matrix that has exactly one element of 1 in each row and each column (denoting a connection between a pair of OXC ports) with all the other elements as 0, where the i-th row/column of a permutation matrix represents

[2]A perfect matrix is a square matrix of non-negative real numbers, where the sum of elements is the same for all the rows/columns.

the ingress/egress ports of the i-th rack. We stuff $\mathbf{D}$ to a perfect matrix $\mathbf{D}'$ to ensure the feasibility of decomposition to permutation matrices, where all the diagonal elements of $\mathbf{D}'$ are kept as 0 since no intra-rack traffic will use the OXC.

$$\overset{\mathbf{D}}{\begin{bmatrix} 0 & 204 & 151 & 76 \\ 664 & 0 & 103 & 76 \\ 317 & 344 & 0 & 76 \\ 505 & 144 & 409 & 0 \end{bmatrix}} \xrightarrow{\textbf{stuff}} \overset{\mathbf{D}'}{\begin{bmatrix} 0 & 864 & 546 & 76 \\ 664 & 0 & 103 & 719 \\ 317 & 478 & 0 & 691 \\ 505 & 144 & 837 & 0 \end{bmatrix}} = 664 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} + 478 \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

$$+ 173 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} + 76 \times \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + 68 \times \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} + 27 \times \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

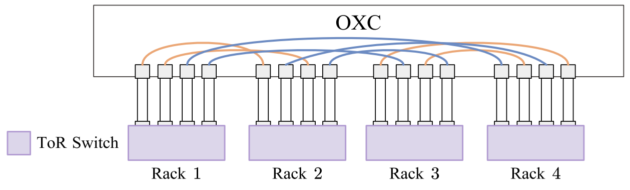


Fig. 3. Example on determining OXC configuration with BvN decomposition.

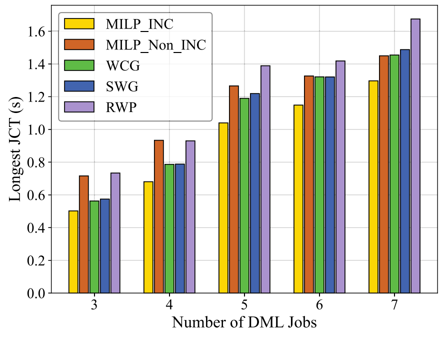
In Fig. 3, $\mathbf{D}'$ is decomposed to 6 permutation matrices, each of which is with a coefficient. A larger coefficient suggests that more data in $\mathbf{D}'$ can be transmitted directly through the OXC ports configured according to the corresponding permutation matrix. Hence, allocating physical connections through the OXC to rack pairs in proportion to the permutation matrices' coefficients helps to reduce the durations of pull/push steps of the DML jobs. For instance, the OXC in Fig. 3 connects 4 ports to each ToR switch and each permutation matrix consumes 2 ports on each ToR switch. Therefore, we can only select $\frac{4}{2} = 2$ matrices to configure the OXC, and the coefficients of the first two permutation matrices suggest that they should be selected (*i.e.*, $\frac{664}{1486} \times 2 \approx 1$ and $\frac{478}{1486} \times 2 \approx 1$).

The time complexity of *Algorithm* 1 is $O(\sum_{i \in \mathcal{J}} |W_i| + |\mathcal{J}| \cdot (1 + \log_2 |\mathcal{J}| + |R_s| + |R_s| \cdot \log_2 |R_s|) + |R_s|^2 + |N_{\text{OXC}}| + |R_s|^{2.5} \cdot N_{\text{ToR}})$, where $O(|R_s|^{2.5} \cdot N_{\text{ToR}})$ is the complexity of the BvN decomposition algorithm in *Line* 27.

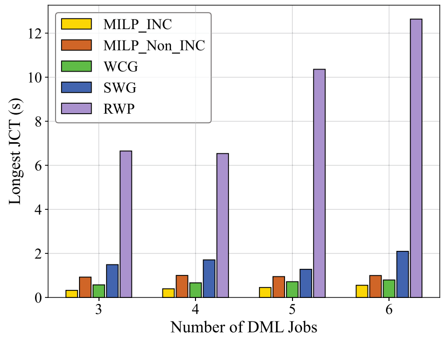
IV. Performance Evaluation

A. Simulation Setup

Our simulations consider both small-scale and large-scale scenarios. The small-scale simulations address an all-optical DCN that consists of $\{2, 4, 6\}$ racks, where half of the ToR switches are based on PDP, each of which can accommodate the INC tasks of two PS' at most. Each ToR switch connects to the OXC via four 10-Gbps ports. The server pool of each rack is conceptualized as a cluster with $[50, 80]$ GPUs, $[20, 30]$ CPUs, and $[1.5, 2]$ GB of memory, while the actual amounts of IT resources in each server pool are determined based on the setting of DML jobs in each simulation. Each small-scale simulation considers the cases with and without INC. Specifically, the case without INC is only solved with the MILP in Section II to minimize the longest JCT, while that with INC is solved with the MILP, our heuristic based on WCG, and two other benchmarks. Here, one benchmark is based on random worker placement (RWP), which places the workers of DML jobs in available racks randomly, and the other uses single worker grouping (SWG), which first sorts all the workers in descending order of their data transfer sizes and then put $|R_s|$ workers in one group in sorted order.



(a) 2 racks



(b) 6 racks

Fig. 4. Small-scale simulation results.

The large-scale simulations only consider WCG, RWP, and SWG, and address an all-optical DCN with 64 racks, where PDP switches are still half of the ToR switches. Each PDP ToR switch can accommodate the INC tasks of 5 PS' at most. The OXC is in 384×384, and thus each ToR switch connects to it via 12 10-Gbps ports. The IT resources in the server pool of each rack are 200 GPUs, 130 CPUs, and 60 GB memory.

In all simulations, the data transfer size of one pull/push step for a worker-PS pair is assumed to be within $[92, 552]$ MB, by fitting the parameter sizes of some well-known DML models (*e.g.*, ResNet-50, ResNet-152, AlexNet and VGG-Net) [21]. The numbers of GPUs and CPUs occupied by each worker are randomly set within $[1, 32]$ and $[1, 8]$, respectively, and the IT resources required by a worker is proportional to its data transfer size. The simulations run on a server with 2.10 GHz Intel(R) Xeon(R) Silver 4110 CPU and 30 GB memory, and the software environment is PyCharm-2023.3.5 with Gurobi 11.0.1 and Python with tensorflow 2.3.0. To ensure statistical accuracy, the simulation result of each data point is obtained by averaging the results of 10 independent runs.

B. Performance Analysis

Fig. 4 shows the results of small-scale simulations, where the number of workers for each DML job is within $[1, 4]$. By comparing the results of the MILPs for cases with and without INC, we find that the introduction of INC significantly reduces the longest JCT of DML jobs, and the reduction becomes more pronounced when the scale of all-optical DCN increases. As for the case with INC, WCG always approximates the MILP the best among the heuristics. When there are only two racks, SWG performs similarly as WCG because their principles are similar when the number of racks is very limited, but the difference between them becomes significant when the number of racks increases to 6. This is because WCG tries to place all the workers of a DML job in one rack, thereby minimizing inter-rack traffic and making it easier to find the best location for their PS to explore the benefit of INC. The algorithms' running time are listed in Table II, which indicates that the heuristics runs much faster than the MILPs and WCG uses the shortest running time, further confirming its time-efficiency.

Fig. 5 plots the results of large-scale simulations, which do not consider the MILPs due to their computational complexity. Here, we have 64 racks in the all-optical DCN, and increase the number of workers for each job to $[4, 16]$ and $[16, 64]$. WCG still reduces the longest JCT better than SWG and RWP. Moreover, the gaps between WCG and the two benchmarks become significantly larger than those in Fig. 4, suggesting that WCG can explore the advantages of the symbiosis of INC and AOI for DML better when the network scale increases. Finally, we notice that the gaps in Fig. 5(b) are smaller than those in Fig. 5(a). This is because when there are more workers in each DML job, it becomes more difficult for WCG to put all the workers of a DML job in one rack to reduce inter-rack traffic, which offsets its advantage to a certain extent.

TABLE II
RUNNING TIME OF SIMULATIONS (SECONDS)

Number of racks	MILP_INC	MILP_Non_INC	WCG	SWG	RWP
2	1.651	0.705	0.165	0.669	0.473
4	22.145	2.799	0.314	0.718	0.581
6	1555.911	30.615	0.349	0.840	1.305
64	–	–	0.674	9.488	8.486

V. Conclusion

In this paper, we studied how to schedule a batch of parallel DML jobs in an all-optical DCN equipped with PDP ToR switches that support INC. We first formulated an MILP model to optimize the scheduling of DML jobs for the cases with and without INC, to minimize the longest JCT, and then proposed a heuristic based on WCG to tackle the case with INC quickly. Extensive simulations confirmed that our proposals can effectively explore the mutual benefits of INC and AOI and reduce the longest JCT better than the benchmarks.

Acknowledgments

This work was supported by the NSFC project 62371432.

References

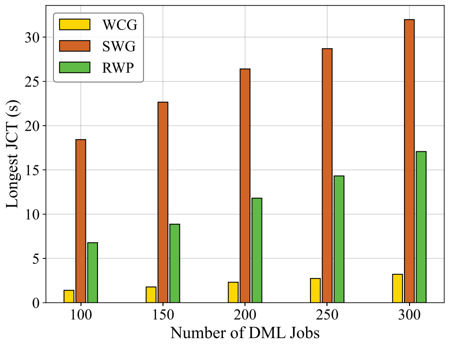
[1] P. Lu *et al.*, "Highly-efficient data migration and backup for Big Data applications in elastic optical inter-datacenter networks," *IEEE Netw.*, vol. 29, pp. 36–42, Sept./Oct. 2015.

[2] M. Ghobadi *et al.*, "Projector: Agile reconfigurable data center interconnect," in *Proc. of ACM SIGCOMM 2016*, pp. 216–229, Aug. 2016.

[3] W. Lu *et al.*, "AI-assisted knowledge-defined network orchestration for energy-efficient data center networks," *IEEE Commun. Mag.*, vol. 58, pp. 86–92, Jan. 2020.

[4] L. Gong and Z. Zhu, "Virtual optical network embedding (VONE) over elastic optical networks," *J. Lightw. Technol.*, vol. 32, pp. 450–460, Feb. 2014.

[5] Z. Zhu, W. Lu, L. Zhang, and N. Ansari, "Dynamic service provisioning in elastic optical networks with hybrid single-/multi-path routing," *J. Lightw. Technol.*, vol. 31, pp. 15–22, Jan. 2013.


(a) $[4, 16]$ workers in each DML job

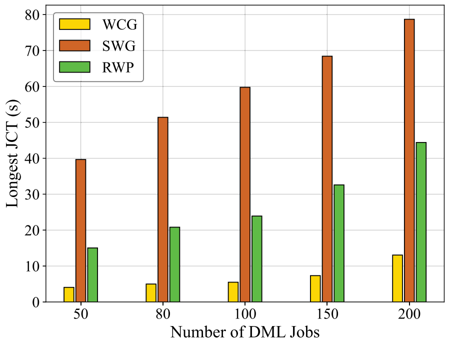

(b) $[16, 64]$ workers in each DML job

Fig. 5. Large-scale simulation results (64 racks).

[6] L. Gong *et al.*, "Efficient resource allocation for all-optical multicasting over spectrum-sliced elastic optical networks," *J. Opt. Commun. Netw.*, vol. 5, pp. 836–847, Aug. 2013.

[7] Z. Zhu *et al.*, "Impairment- and splitting-aware cloud-ready multicast provisioning in elastic optical networks," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 1220–1234, Apr. 2017.

[8] H. Yang and Z. Zhu, "Traffic-aware configuration of all-optical data center networks based on Hyper-FleX-LION," *IEEE/ACM Trans. Netw.*, vol. 25, pp. 1220–1234, Apr. 2024.

[9] J. Verbraeken *et al.*, "A survey on distributed machine learning," *ACM Comput. Surv.*, vol. 53, pp. 1–33, Mar. 2020.

[10] M. Alizadeh and T. Edsall, "On the data path performance of leaf-spine datacenter fabrics," in *Proc. of IEEE HOTI 2013*, pp. 71–74, Aug. 2013.

[11] H. Yang, Z. Zhu, R. Proietti, and B. Yoo, "Which can accelerate distributed machine learning faster: Hybrid optical/electrical or optical reconfigurable DCN?" in *Proc. of OFC 2022*, pp. 1–3, Mar. 2022.

[12] Y. Jiang *et al.*, "A unified architecture for accelerating distributed DNN training in heterogeneous GPU/CPU clusters," in *Proc. of OSDI 2020*, pp. 463–479, Nov. 2020.

[13] L. Liu *et al.*, "Online job scheduling for distributed machine learning in optical circuit switch networks," *Knowl. Based Syst.*, vol. 201, p. 106002, Jun. 2020.

[14] A. Feng *et al.*, "In-network aggregation for data center networks: A survey," *Comput. Commun.*, vol. 198, pp. 63–76, Jan. 2023.

[15] X. Xie, H. Yang, and Z. Zhu, "P4INC-AOI: When in-network computing meets all-optical interconnect for adaptive and low-latency optical DCN," in *Proc. of OFC 2023*, pp. 1–3, Mar. 2023.

[16] Polatis Series 7000 Software-Defined Optical Circuit Switch. [Online]. Available: https://www.polatis.com/series-7000-384x384-port-/software-controlled-optical-circuit-switch-sdn-enabled.asp.

[17] J. Gu *et al.*, "Tiresias: A GPU cluster manager for distributed deep learning," in *Prof. of NSDI 2019*, pp. 485–500, Feb. 2019.

[18] H. Cui *et al.*, "GeePS: scalable deep learning on distributed GPUs with a GPU-specialized parameter server," in *Proc. of EuroSys 2016*, pp. 1–16, Jan. 2016.

[19] H. Ryser, *Combinatorial Mathematics*. Rahway, NJ: The Mathematical Association of America, 1965.

[20] Z. Qiu, C. Stein, and Y. Zhong, "Minimizing the total weighted completion time of coflows in datacenter networks," in *Prof. of SPAA 2015*, pp. 294–303, Jun. 2015.

[21] Y. Li, C. Fan, X. Zhang, and Y. Chen, "Placement of parameter server in wide area network topology for geo-distributed machine learning," *J. Commun. Netw.*, vol. 25, pp. 370–380, Jun. 2023.