Future Generation Computer Systems 106 (2020) 320–332

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

# PSNet: Reconfigurable network topology design for accelerating parameter server architecture based distributed machine learning

Ling Liu [a], Qixuan Jin [a], Dan Wang [a], Hongfang Yu [a,b,*], Gang Sun [a,*], Shouxi Luo [c]

[a] *Key Laboratory of Optical Fiber Sensing and Communications (Ministry of Education), University of Electronic Science and Technology of China, Chengdu, People's Republic of China*
[b] *Peng Cheng Laboratory, Shenzhen, People's Republic of China*
[c] *Southwest Jiaotong University, Chengdu, People's Republic of China*

A R T I C L E  I N F O

*Article history:*
Received 13 June 2019
Received in revised form 9 November 2019
Accepted 4 January 2020
Available online 15 January 2020

*Keywords:*
Distributed Machine Learning (DML)
Parameter Server (PS) architecture
Electrical Circuit Switch (ECS)
Network topology

A B S T R A C T

The bottleneck of Distributed Machine Learning (DML) has shifted from computation to communication. Lots of works have focused on speeding up communication phase from perspective of Parameter Server (PS) architecture, for example resource scheduling. Nonetheless, the performance improvement of these schemes is limited due to the agnostic of the physical topology to the communication pattern of the applications. Concurrently, some articles have also pointed out the impact of topology on DML performance. Besides, our analysis and experimental results also indicate that the general topologies cannot match well with the communication characteristics of DML based on PS architecture. However, to the best of our knowledge, no special topology is tailored for DML. Therefore, in this paper, we propose PSNet, a reconfigurable modular network topology for DML with consideration of the communication characteristics of PS architecture. The main idea of PSNet is that servers are firstly divided into two categories, namely workers configured with high-performance computing capability and parameter servers equipped with multiple Network Interface Cards (NICs). Then Electrical Circuit Switch (ECS) is exploited to connect workers and Top of Rack (ToR) switches for flexibility and reconfigurability in each module. Our theoretical analysis proves that PSNet not only provides high performance for DML tasks, but also achieves high fault tolerance and flexibility. In order to validate the performance of PSNet, we conduct large-scale simulations and small-scale testbed experiments, and the results of experiments demonstrate that PSNet performs 1.89× and 1.92× faster than FatTree for VGG-16 and ResNet50, respectively.

## 1. Introduction

In an unprecedented era in Artificial Intelligent (AI) research history, machine learning technologies are widely used in areas such as computer vision [1], speech [2,3], and language understanding [4]. With the increasing sizes of data and models, distributed system is becoming a prerequisite for solving large-scale machine learning problems [5–8]. Data parallelism is the most common parallel strategy of distributed machine learning (DML) due to its simplicity [9,10], where data are distributed over a network of machines and each machine trains models in a distributed manner. In order to train various ML models for AI-driven services, most leading IT companies such as Google and Microsoft, have built ML clusters with hundreds or thousands of servers.

Parameter Server (PS) [11] architecture is a commonly used solution for data parallelism of DML, and it has been widely deployed in machine learning frameworks such as TensorFlow [12], MXNet [13], Caffe [14] and Angel [15], due to its easy deployment, high scalability and error tolerance. There are two kinds of servers in PS architecture, namely workers that train model replicas on partitioned local data in parallel, and parameter servers that are responsible for synchronizing these model replicas [11]. Due to the guarantee of data-parallel convergence, synchronous mode is commonly used to enable reproducibility in ML cluster [16]. When in synchronous mode, each worker in PS architecture computes parameter updates (i.e., gradients) and then pushes them to parameter servers. The global model will not be updated until parameter servers have received gradients from all workers in each iteration. Then each worker continues the next-iteration training after pulling latest model parameters. In this case, per-iteration time directly affects the training time of a DML task. With the use of hardware accelerators such as GPU, FPGA and TPU [17], each machine can deal with more training samples per minute, leading

* Corresponding authors.
*E-mail addresses:* lingliu@mail.ustc.edu.cn (L. Liu), yuhf@uestc.edu.cn (H. Yu), gangsun@uestc.edu.cn (G. Sun).

https://doi.org/10.1016/j.future.2020.01.004
0167-739X/© 2020 Elsevier B.V. All rights reserved.

to more frequent network synchronization, which means more data should be transmitted through the network per minute. As a consequence, the performance bottleneck of DML has shifted from computation to communication [18–20].

Consequently, lots of works have been provided to accelerate the communication process of DML from perspective of PS architecture, such as resource allocation and scheduling [21–23] and improvement of PS-based frameworks [24], etc. Although these proposals accelerate the training speed to some extent, the performance improvement is constrained by the underlying physical network performance [25]. Meanwhile, physical network performance, such as network latency and throughput, has a severe effect on training performance [17]. Altering the network topologies to alleviate network bottleneck for better network performance is a common and well-studied approach in the literature. However, to the best of our knowledge, there is no special network topology tailored for DML. Most existing network topologies are designed agnostic to the specific requirements of the upper applications [26], but rather pursue some general purposes, such as scalability, fault tolerance and throughput. This mismatch could hurt the performance of DML.

Therefore, a topology meeting communication requirements of PS architecture plays a vital role in speeding up DML training efficiently. Unfortunately, existing network topologies are not fit for PS architecture which has a specific demand for network performance [27]. For example, parameter servers serving as data aggregation nodes, need higher bandwidth than workers [18,27]. Meanwhile, the distance and latency between each worker and each parameter server should be as close as possible for faster per-iteration time in synchronous mode. In general, current network topologies can be divided into three categories, namely, server-centric where servers are used to forward and route data, switch-centric where switches or routers are responsible for routing, and reconfigurable topologies where optical or wireless communication is introduced into network to provide configurable links. Most server-centric approaches, such as DCell [28], use recursively-defined structures to interconnect servers, achieving high scalability. Nevertheless, the distance between two servers increases as network level increases, and path lengths between servers vary greatly, which is not beneficial to parameter synchronization. On the other hand, switch-centric topologies are more widely used in industry, due to their easy construction and maintenance, for example FatTree [29]. However, these architectures are usually symmetrical, and all the servers are equivalent and have the same bandwidth, which will make the bandwidth of parameter server a potential performance bottleneck. In addition, bandwidth oversubscription in these architectures aggravate network congestion. Besides, the deployment of workers and parameter servers have a severe impact on training performance according to the results of our experiments in Section 5. Hence, existing switch-centric topologies are not well suited for PS architecture.

In recent years, the reconfigurable topology becomes more and more popular due to the ability to adjust its architecture according to the application demand. Many reconfigurable topologies using Optical Circuit Switch (OCS) [30–33] or wireless radios [34, 35] have been proposed for the purpose of alleviating local congestion or providing shorter paths. On the one hand, wireless communication has high demands for the environment and usually requires modification of the computer room [34]. Moreover, sophisticated aligning the beams of all nodes and anti-jamming measures are also required [35]. These make the maintenance of the network quite difficult. On the other hand, an OCS costs too much and the circuit reconfiguration latency is up to tens of milliseconds [36], which is too long for time-sensitive DML. In summary, these flexible reconfigurable topologies with OCS or wireless radios are not a good choice for DML. Accordingly, to effectively improve DML training performance, it is essential to design a reconfigurable topology tailored for PS architecture.

In this article, we present PSNet (**P**arameter **S**erver architecture **Net**work), a reconfigurable modular topology for PS architecture based DML. We not only fully consider the traffic characteristics of PS architecture, but also take the scalability and error tolerance into account, which ensures high error tolerance and high performance for PS architecture. PSNet is constructed with the idea of modularity. In each module, servers are classified into workers and parameter servers in terms of the characteristics of PS architecture. Besides, each parameter server is equipped with several network ports for large bandwidth and each worker is equipped with high-performance processors, such as GPU. In the intra-module construction, we exploit Electrical Circuit Switch (ECS) to connect workers and ToR switches for high flexibility, which has been introduced into network topologies in recent years because of its shorter circuit reconfiguration delay and lower price [37–39] compared with OCS. In the inter-module construction, several modules are connected into a regular graph. In a nutshell, PSNet provides multiple short and reconfigurable paths between parameter servers and workers. This not only provides large bandwidth for parameter servers, but also greatly improves fault tolerance and load balance for DML application. The main contributions of this work are as follows:

- We present PSNet, a reconfigurable and flexible modular network topology for PS architecture based on its communication requirements, which not only provides high performance for PS architecture, but also achieves high error tolerance and flexibility.
- We carry out theoretical analysis of the batch completion time (BCT) of DML tasks both in PSNet and the commonly used network topology FatTree. Besides, we carry out numerical simulations and analyze the performance of large-scale DML tasks deployed in both topologies.
- We conduct extensive experiments to evaluate the performance of PSNet and compare it with FatTree in our testbed. The results indicate that PSNet yields significant performance improvement, which is consistent with the theoretical analysis.

The remainder of this paper is organized as follows. Section 2 motivates the need of our work. Section 3 describes the details of PSNet, including physical architecture and control plane. Section 4 carries out theoretical performance analysis of PSNet. Section 5 presents system implementation and compares the performance of PSNet with FatTree through experiments. Section 6 makes more discussions on PSNet. Section 7 describes the related work. Finally, Section 8 concludes this work.

## 2. Motivation

Recently, some works [17,18] focusing on studying the influence of network on PS architecture, illustrate that network has become the performance bottleneck. Therefore, designing a tailored network topology that meets the communication requirements of PS architecture is critical importance for performance improvement.

By combing the analysis of flow pattern of PS architecture and various experiments results, we summarize the following requirements of PS architecture for network topology.

**The topology should distinguish between workers and parameter servers**. In PS architecture, workers and parameter servers have different functions, as well as different hardware resource demands. Workers are responsible for processing samples and require strong computational ability. While parameter

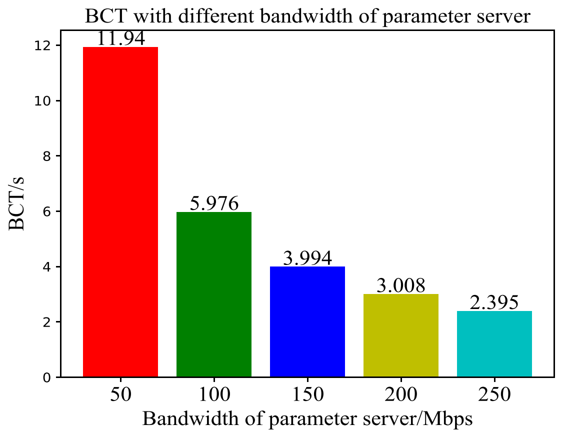


**Fig. 1.** The impact of bandwidth of parameter server on BCT when model is ResNet110, bandwidth of worker is 1000Mbps, batch size is 5 and dataset is CIFAR 10.

servers serving as data aggregation nodes, need sufficient bandwidth to receive data from workers at the same time. Therefore, it is necessary to classify parameter servers and workers in the network topology. However, the existing network topologies widely used in industry always treat servers as the same rather than distinguishing between different server types.

**The topology should provide parameter servers with large bandwidth**. With the popularity of hardware accelerators such as GPU, the throughput of processing samples of workers increase, as well as the number of iterations per minute [19]. That means the amount of data transmitted per minute between parameter server and worker is also increasing. In addition, there is a serious traffic imbalance between parameter servers and workers. Each parameter server has to receive data from all the workers at the same time, so larger bandwidth is required. Otherwise, the bandwidth of parameter server will become the performance bottleneck. The results of our experiment where 1 parameter server and 5 workers are connected through a switch illustrate this conclusion. As shown in Fig. 1, per-iteration time becomes shorter with the increase of bandwidth of parameter servers in the case of insufficient bandwidth. Besides, significant efforts have also verified this conclusion [17,18]. Nevertheless, since most topologies do not differentiate between server types, and each server has the same bandwidth. Therefore, these topologies could not provide large bandwidth to servers working as parameter servers.

Moreover, although OCS or wireless radios can be used to provide extra bandwidth for parameter servers, OCS is so expensive and has long circuit reconfiguration latency, which is not fit for latency-intensive DML. Meanwhile, the overall wireless bandwidth depends on the number of wireless radios. Besides, workers with the same frequency cannot send data to parameter server at the same time, which is not conductive to parameter synchronization.

**The attributes of paths between any worker and parameter server should be similar**. That is to say, the difference of the paths between any worker and parameter server should be small, including latency, bandwidth and path length. In synchronous mode, parameter server updates global model parameters only after receiving data from all workers. If the paths between workers and parameter servers vary greatly, the parameter server has to wait for the slowest transmission, which will increase the per-iteration time, leading to longer convergence time. For instance, the network diameter in server-centric topologies with recursively-defined structure always increases as the level increases, so the distances between nodes vary a lot. Therefore, the recursive server-centric topologies are not suitable for PS architecture.

**The topology should have high fault tolerance**. The topology for PS architecture should have high link error tolerance and node error tolerance. On the one hand, as a data aggregation node, the backup and fault tolerance of parameter server is vitally important, because a standby parameter server needs to be restarted quickly to ensure that the training task is not greatly affected when some parameter server fails. Hence, it is necessary for the topology to provide an alternative parameter server that has almost the same performance as the failed one. On the other hand, parameter server has to wait the data transmission from all workers in each iteration. If one link of the path between a worker and parameter server breaks, some other available link or path should be used instead, otherwise the parameter server will wait until the task is canceled. Therefore, the topology should also provide multiple links or paths between workers and parameter servers for high link error tolerance.

## 3. PSNet architecture

In this section, we introduce the PSNet construction according to the communication requirements of PS architecture in detail. We first present the PSNet structure, and then provide the control plane.

### 3.1. PSNet construction

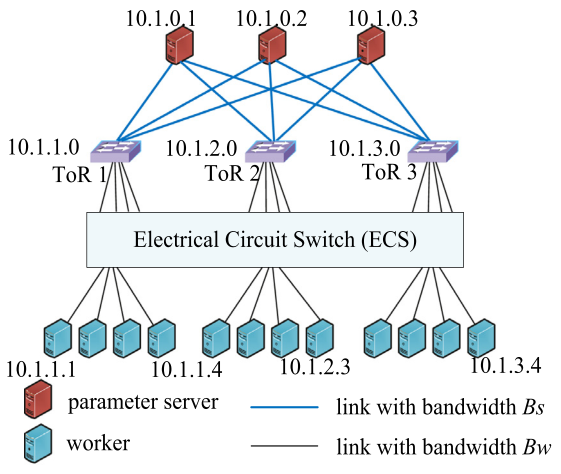
In the construction of PSNet, we exploit ECS for several reasons. First, the dynamic reconfigurability of ECS brings high fault tolerance and high flexibility. For instance, when some link fails, we only need to re-establish a link between communication nodes for normal communicating by circuit reconfiguration. Second, compared with OCS, ECS has simpler hardware architecture, so that it has much lower cost and shorter reconfiguration latency. In addition, it does not decode/encode or buffer packets and it only takes several nanoseconds for ECS to transfer the signal from one port to another [37]. This is fast enough to be transparent to the upper DML tasks.

PSNet is a modular network topology, which is constructed by multiple modules. In the following, the construction of the intra-module and inter-module will be described in detail.

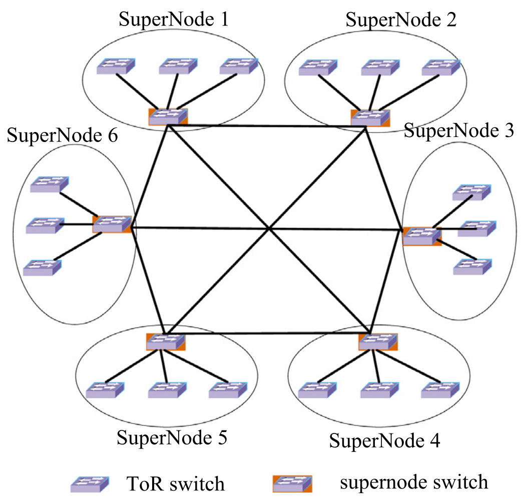
**Intra-Module Construction:** First, servers are divided into two types, namely parameter servers that are equipped with $R$ Network Interface Cards (NICs) and workers that are equipped with high-performance computing units, for example GPU. Second, an ECS with $W' \times W'$ ports is connected to $R$ ToR (top-of-rack) switches, and each ToR switch has $M_u \times M_d$ ports, where $R = \lfloor \frac{W'}{M_d} \rfloor$, $M_u$ is the number of uplink ports and $M_d$ is the number of downlink ports. For example, when an ECS has network ports as many as $192 \times 192$, and if a ToR switch can connect 64 workers, then the ECS can connect 3 ToR switches. Third, servers working as parameter servers are equipped with $R$ ports, and these ports are connected to the $R$ ToR switches respectively. Fourth, servers serving as workers are all connected to the ECS.

A basic module is called a SuperNode, and Fig. 2(a) depicts a module when $R = 3$ and $W' = 12$. In general, the features of SuperNode are as follows:

- High fault tolerance is achieved. On the one hand, even if some links between parameter server and ToR switches or some links connecting ToR switches and circuit switch fail, a worker still has an alternative path to communicate with parameter server, due to the use of circuit switch and the full connection between parameter server and ToR switches. On the other hand, every worker can communicate with all parameter servers in a SuperNode, even if one parameter server goes down, another parameter server can be substituted without degrading the training performance.



(a) Intra-module construction when $R = 3$, $W' = 12$.



(b) Inter-module construction when $R = 3$, $H = 12$.

**Fig. 2.** PSNet architecture.

- The bandwidth of parameter servers is large. Each parameter server has $R$ ports, and each worker can transmit data through $R$ ToR switches. Compared with multi-layer Clos topologies where all the servers are the same and each has only one port, the bandwidth of a parameter server in a SuperNode becomes $R$ times. If $B_s = K \times B_w$ ($B_s$ and $B_w$ are the uplink bandwidth and downlink bandwidth of a ToR switch, respectively), the total bandwidth of a parameter server is $R \times K \times B_w$.
- The hop distance between any worker and parameter server in a SuperNode is the same, and it is always 2. The time of the data passing through circuit switch is fast enough that the circuit switch can be perceived as a link. Therefore, the path length from any worker to parameter server always equals to 2. The same path condition facilitates the synchronization of model parameters.
- The load balance can be obtained. With the help of ECS, workers are no longer bound to a rack. Instead, their connectivities to parameter servers can be reconfigured dynamically so that a worker can transmit data to any parameter server via a same network hop distance regardless of their physical locations. Through reasonable circuit configuration, data from workers to parameter servers can be distributed equally to each ToR switch, so that load balance is achieved. Therefore, each link between parameter server and ToR switch has the same load and the same data transfer time, which helps speed up parameter synchronization.

**Inter-Module Construction**: First, for each SuperNode, a common commercial switch (called SuperNode switch) is used to connect $R$ ToR switches in each SuperNode. Second, $H$ SuperNodes are connected to form an $R$-regular graph via SuperNode switches.

The inter-module structure is shown in Fig. 2(b) when $R = 3$ and $H = 6$. In summary, the characteristics of inter-module structure are as follows:

- The link load between modules is balanced. The nodes and links in the regular graph are completely symmetric, and data transmitted from nodes in SuperNode $A$ to nodes in another SuperNode $B$ can be distributed evenly on the $R$ parallel paths that have no shared links. Therefore, overloads in some links like DCell [28] will not occur, and load balance is achieved.
- The fine-grained expansion of the topology is achieved. The expansion granularity of server-centric topologies in a recursive form is coarse. For example, DCell scales doubly exponentially as the node degree increases. Moreover, FatTree [29] scales 3 power of the number of switch ports. Compared to FatTree and DCell, PSNet expands by adding new modules, so network can grow with fewer severs. Therefore, fine-grained expansion can be obtained, which is important for efficient use of resources and cost saving.
- The diameter between modules will increase with the increase of modules. However, in our design, the workers and parameter servers of a task can be deployed in a SuperNode as far as possible. With the increase number of circuit switch ports as well as a server equipped with several processors [40], a single SuperNode can provide more and more workers and parameter servers, which will meet the demands of most DML tasks. For instance, when ECS has 384 ports, each ToR switch has 8 uplinks and 48 downlinks, and each server has 8 processors, a single module can provide 1536 workers and 224 parameter servers. That is large enough for most large-scale DML tasks. If some tasks have to be deployed across modules, we can place them on adjacent modules, so that the path length between parameter servers and workers can also be kept short. Therefore, most DML tasks will not be affected when new modules are added.

## 3.2. Addressing

All the IP addresses in the network are allocated within the private*10.0.0.0/8* block. Each switch is identified by its module ID and its switch ID: *10.mid.swid.0*, where *mid* denotes the module number (in [*1*,*H*]), and *swid* is the switch number (in [*1*,*R*], starting from left to right) and the supernode switch number is *R+1*. The address of a parameter server has an address of the form: *10.mid.0.psid*, where *psid* is the parameter server position in that subnet (in [*1*,*P*], starting from left to right). Besides, the workers are given addresses of the form *10.mid.swid.wid*, where *wid* ranges in $[1, \frac{W'}{B}]$. Note that although each worker can connect to any ToR switch through circuit reconfiguration, we still distribute workers equally under each ToR switch by default. Besides, the address of a worker will not change even if it is connected to another ToR switch through circuit reconfiguration. Fig. 2 shows examples of this addressing scheme for PSNet.

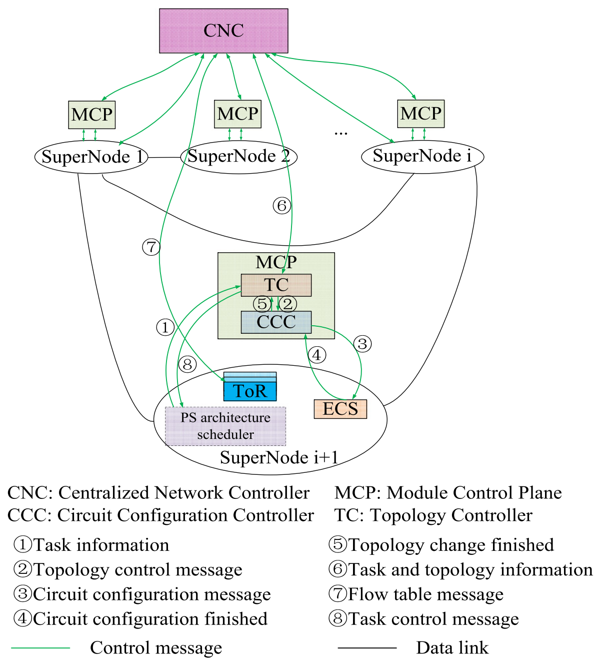

CNC
MCP
MCP
MCP
SuperNode 1
SuperNode 2
SuperNode i
...
MCP
TC
CCC
ToR
ECS
PS architecture scheduler
SuperNode i+1
①
②
③
④
⑤
⑥
⑦
⑧
CNC: Centralized Network Controller
MCP: Module Control Plane
CCC: Circuit Configuration Controller
TC: Topology Controller
①Task information
②Topology control message
③Circuit configuration message
④Circuit configuration finished
⑤Topology change finished
⑥Task and topology information
⑦Flow table message
⑧Task control message
Control message
Data link

**Fig. 3.** Control plane of PSNet.

## 3.3. Control plane of PSNet

The PSNet architecture requires a control plane (1) to program circuit switch connectivity and (2) to enforce routing for DML tasks. The control plane consists of two parts, namely Centralized Network Controller (CNC) to achieve global resource management and Module Control Plane (MCP) for each module.

Each module requires a MCP, whose main purpose is to control the flexible transformation of topology by effective use of the circuit switched paths according to the requirements of DML tasks, so as to minimize the completion time of tasks. MCP is composed of Circuit Configuration Controller (CCC) and Topology Controller (TC). CCC is the unmodified control software provided by circuit switch supplier, which is used to control the circuit configuration of circuit switch. TC is a user process running on a dedicated server in a separately connected control network, which has the following two functions:

- Managing circuit configuration. TC can obtain the DML tasks information such as the number of workers and the size of the model parameters from PS architecture, and calculates timely circuit configuration scheme on the basis of the current network information, then sends it to CCC. Our goal for computing a new circuit reconfiguration is to minimize the completion time of DML tasks, and it can be achieved by distributed workers equally to the ToR switches in the module, which is verified by the performance analysis and experiments in the following sections.
- Controlling the running states of DML tasks. TC is also responsible for controlling the operation states of DML tasks. For instance, when the circuit configuration has been completed, TC needs to send instructions to PS architecture to start a task.

The CNC is an emerging trend to achieve global network management [41]. There are existing works in the literature providing basic routing mechanisms we can learn, and software-defined networking (SDN) scheme is employed in this paper. After the MCP has determined the circuit connectivity in its module, our CNC can pre-compute the paths and program the routing decisions on switches using SDN protocols. Note that when it comes to communication between modules, all the *R* disjoint paths can be used at the same time to transmit data.

The control plane of PSNet is shown in Fig. 3. First, TC gets current task information from PS architecture scheduler. It is worth noting that PS scheduler can be deployed on parameter server or worker, or other servers. Second, TC computes a new topology for its module in order to accelerate the communication phrase of DML according to its module and task information. Then TC sends the new topology to its CCC. Third, CCC sends circuit configuration message to its ECS. Fourth, after the links have been re-established, ECS notifies the CCC, and fifth, CCC notifies TC. Sixth, TC sends its task and circuit configuration information to CNC. Seventh, CNC computes the paths between modules and sends flow tables to the ToR switches in all the modules related to the tasks. In the end, when all the flow tables related to the tasks have been established, TC in each module notifies PS scheduler to start task. Note that, the main contribution of this work is the PSNet network topology, and other alternative control plane designs can be also used.

## 3.4. Summary of features of PSNet

In summary, the features of PSNet are as follows:

- **Server types**: Servers in PSNet are divided into parameter servers and workers in terms of their functionalities and hardware requirements.
- **Bandwidth for parameter servers**: The parameter servers are equipped with several ports, workers can share these ports to communicate with each parameter server. Therefore, large bandwidth can be achieved, which is beneficial to parameter synchronization.
- **Path length**: The path length between any worker and parameter server in a module is always 2, regardless of the ECS that can be seen as transparent to upper applications. Besides, the bandwidth of these paths is set the same, so their latency can be regarded the same as well. Although the distance between any two modules will increase with the number of modules, this is acceptable. Because most DML tasks are only placed in one module or two adjacent modules, network expansion will not affect these tasks.
- **Fault tolerance**: PSNet provides good path redundancy. On the one hand, in a module each parameter server has several links connecting with all the ToR switches, and each worker can also communicate with parameter server through multiple paths by reasonable circuit reconfiguration. Therefore, PSNet is highly resilient to parameter server nodes failures and link failures. On the other hand, an *R*-regular random graph is almost surely *R*-connected [42], so the interconnection between modules can also maintain its structure in the face of link or SuperNode switch failures. Therefore, PSNet has high fault tolerance.
- **Scalability and flexibility**: The PSNet topology generation method is extremely scalable and flexible. On scalability, we mean that more servers and switches can be added into the topology through adding new modules. Compared to some Clos architectures (e.g., Fat Tree) and iterative defined server-centric topologies, which can only be built at certain sizes, PSNet is designed to support fine-grained incremental expansion. Moreover, it will not have any effect on the tasks deployed on the existing servers and switches in each module when network expansion, only affecting tasks across modules. On flexibility, we mean that the circuit reconfiguration of ECS gives flexible and dynamic changes to the

**Table 1**
Notation.

| Symbol | Definition |
|---|---|
| $P$ | # of parameter server of job |
| $W$ | # of worker of job |
| $B_s$ | bandwidth of parameter server |
| $B_w$ | bandwidth of worker |
| $P'$ | # of parameter servers in a SuperNode |
| $W'$ | # of workers in a SuperNode |
| $W1$ | # of workers in the first SuperNode |
| $P1$ | # of parameter servers in the second SuperNode |
| $P_w(samples/s)$ | processing throughput of worker |
| $P_s(Bytes/s)$ | processing throughput of parameter server |
| $BS(samples)$ | size of a batch of jo b |
| $S(bytes)$ | size of model parameters of job |
| $latency$ | latency of a link |
| $R$ | # of NICs of parameter server |
| $K$ | ratio of $B_s$ to $B_w$ |
| $T_{compute}$ | computation time of worker |
| $T_{ps\_update}$ | model parameters updating time of parameter servers |
| $T_{sync}$ | synchronous training time of ML framework |
| $T_{push}$ | push time of worker |
| $T_{pull}$ | pull time of worker |

network. Besides, PSNet is extremely flexible in terms of the type of network equipment (e.g. servers and switches). The network equipment in different modules can be heterogeneous.

In conclusion, the characteristics of PSNet enable it not only improve the performance of PS architecture, but also achieve high fault tolerance and scalability. In the next section, we will analyze the performance of DML in PSNet theoretically.

## 4. Performance analysis

In this section, we analyze the performance of a DML task with PS architecture in PSNet. Firstly, we analyze and model the BCT of a DML task, and then compare the performance of a large-scale task deployed in PSNet and FatTree respectively. Important notation is summarized in Table 1.

### 4.1. BCT modeling

In synchronous mode, the completion time of a DML task is equal to the number of training iterations required to achieve ideal accuracy times the per-iteration time which is almost the same, so we can analyze the per-iteration time to evaluate the performance of a task in PSNet. In most DML tasks, every worker only uses a small part of data instead of all the local training data. The training samples in each iteration is called batch size, so the per-iteration time is also named batch completion time (BCT).

The distributed synchronous training process when there are 2 workers and 1 parameter server is shown in Fig. 4. First, all workers pull model parameters from parameter servers. Second, all workers compute gradients, then push the gradients to parameter servers. Third, when parameter servers have received all gradient from workers, the mean of the gradients is calculated, and the model is updated based on that mean. Then the next iteration begins.

In reality, the computation and communication process may overlap, and the push process and pull process may also overlap. In the worst case when computation and communication do not overlap at all, the BCT consists of four parts:

$$BCT_{max} = T_{pull} + T_{compute} + T_{push} + T_{ps\_update} + T_{sync} \tag{1}$$

In the best case when computation and communication process overlap completely, and the push process and pull process

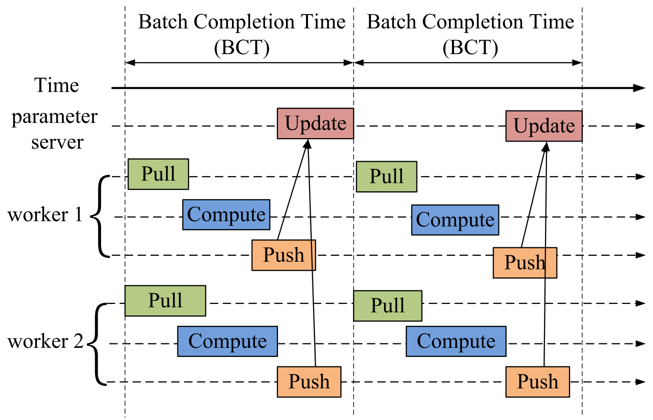


**Fig. 4.** Distributed synchronous training process.

are also completely overlapping, then

$$BCT_{min} = max((T_{compute} + T_{ps\_update} + T_{sync}), T_{pull}, T_{push}) \tag{2}$$

In general, the actual value is usually bounded between $BCT_{min}$ and $BCT_{max}$. In the above equations, $T_{compute}$ standards for the time workers calculate gradients based on local data. When the batch size is $BS$ and worker numbers is $W$, the number of training samples each worker owns is $\frac{BS}{W}$, when each worker has the same computing ability which is $P_w$ $(samples/s)$. Hence,

$$T_{compute} = \frac{BS}{W \times P_w} \tag{3}$$

$T_{ps\_update}$ is the time when parameter servers update the model parameters. When the data processing capacity of parameter server is $P_s$ (Bytes per second), and the data each parameter server processes are $W \times (\frac{S}{P})$, then

$$T_{ps\_update} = \frac{W \times S}{P \times P_s} \tag{4}$$

$T_{sync}$ is the time required to achieve synchronous training, which is related to the machine learning platform (e.g., TensorFlow and MXNet).

$T_{push}$ means the time workers push gradients to parameter servers, and $T_{pull}$ is the time workers pull parameters from parameter servers. The data transmitted and bandwidth are both the same in the two processes, so $T_{push}$ and $T_{pull}$ are regarded as the same. It must be mentioned that when given a task and its related information, such as batch size, number of workers and number of parameter servers, $T_{compute}$, $T_{ps\_update}$, $T_{sync}$ are constants. In contrast, $T_{push}$ and $T_{pull}$ are related to the deployments of workers and parameter servers, as well as the network topology.

Therefore, we only analyze the communication time ($T_{push}$ + $T_{pull}$ ) in the following two different scenarios in PSNet. Note that, in the following equations, $B_s = K \times B_w$.

*Scenario 1: All the workers and parameter servers are in the same module.*

As for each parameter server, the number of model parameters each parameter server takes in charge is $\frac{S}{P}$, and the overall bandwidth of each parameter server is shared equally by all workers. As for each worker, its bandwidth is divided by all parameter servers. Therefore,

$$T_{pull} = T_{push} = \frac{S/P}{min(\frac{R \times B_s}{W}, \frac{B_w}{P})} + 2 \times latency \tag{5}$$

*Scenario 2: All the workers and parameter servers are distributed in two modules. P1 of P parameter servers and W1 of W workers*

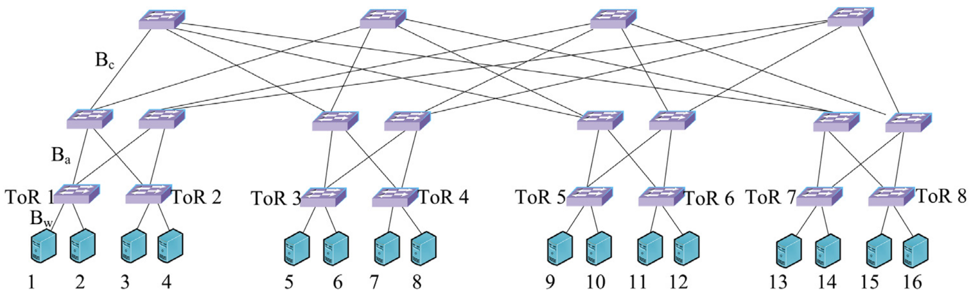


**Fig. 5.** FatTree topology when number of switch ports is 4.

*are placed in the first module, the others are deployed in the second module.*

Assume that the bandwidth between modules is equal to $B_s$, so the inter-module link bandwidth from each worker in the first module to each parameter server in the second module, and the inter-module link bandwidth from each worker in the second module to each parameter server in the first module are equivalent to $\frac{R\times B_s}{W1\times(P-P1)}$, $\frac{R\times B_s}{(W-W1)\times P1}$, respectively. Hence,

$$T_{pull} = T_{push} = \frac{S/P}{min(\frac{R\times B_s}{W}, \frac{B_w}{P}, \frac{R\times B_s}{W1\times(P-P1)}, \frac{R\times B_s}{(W-W1)\times P1})} + \alpha \times latency \quad (6)$$

where $\alpha$ is the max length path of the communication between any worker and any parameter server, which has to do with the scale of network.

As a contrast, we also analyze the BCT of a DML task in a Clos network which is the *de-facto* standard network architecture in industry due to easy implementation [40,43]. We adopt Fat-Tree [29] network as a representative, which is shown in Fig. 5. Bandwidth oversubscription can occur at any switch layer in a Clos network for low cost, hence we compute the BCT of a DML task in the oversubscribed FatTree network.

*Scenario 3: All parameter servers and workers are deployed in a FatTree network with bandwidth oversubscription r at ToR switch layer.*

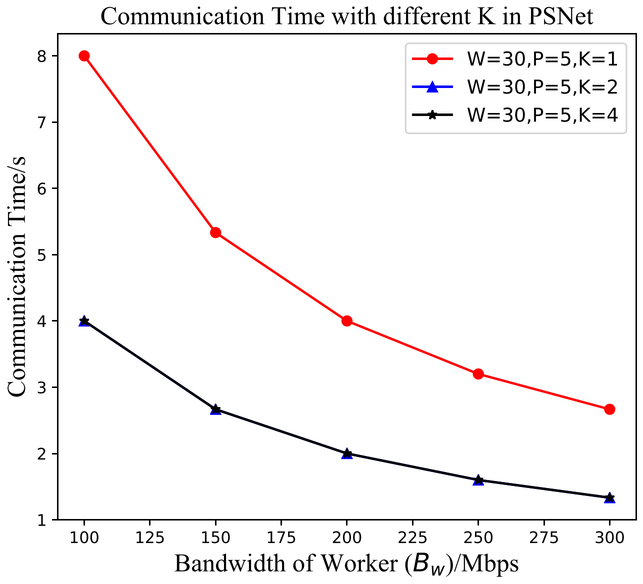
$$T_{pull} = T_{push} = \frac{S/P}{min(\frac{B_s}{W}, \frac{B_w}{P}, \frac{B_a}{C})} + \beta \times latency \quad (7)$$

where $B_a$ is aggregation layer bandwidth that is equal to core layer bandwidth $B_c$ in this work. When the bandwidth over-subscription is $r$, $B_a = B_c = \frac{B_w}{r}$. $C$ is the maximum number of concurrent communications passing an aggregation or a core link in the network, which is related to the routing strategy and DML task deployments. It is worth noting that different worker location may affect the $C$ and BCT, which is verified by our experimental results in the next section. $\beta$ is the max hop count between a worker and a parameter server, which is related to the location of workers and parameter servers.

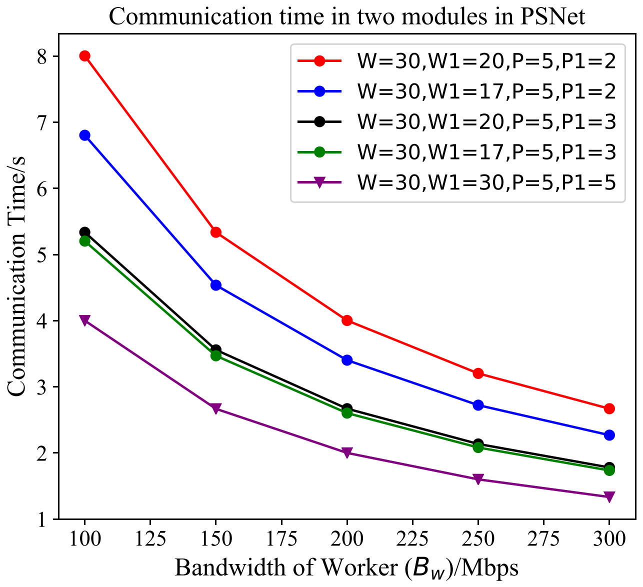
## 4.2. Performance evaluation at scale

In this section, we simulate the performance of large-scale DML tasks due to our limited experimental condition which only allows small-scale experiments. Given a DML task, except $T_{push}$ and $T_{pull}$, the others can be regarded as constants. Therefore, we only evaluate the communication time ($T_{push} + T_{pull}$) as network changes when *latency = 0.00002* s, *R = 3 and S = 200* Mbytes. In this section, the workers deployed in each module in PSNet are evenly connected to $R$ ToR switches through ECS. Besides, the workers and parameter servers in FatTree are deployed in two Pods.

Fig. 6(a) shows the communication time with different $K$ when a DML task is deployed in one module in PSNet. It can be seen



(a) Communication time of different $K$ in one module in PSNet.



(b) Communication time when in one module and in two modules in PSNet.

**Fig. 6.** Communication time in PSNet. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

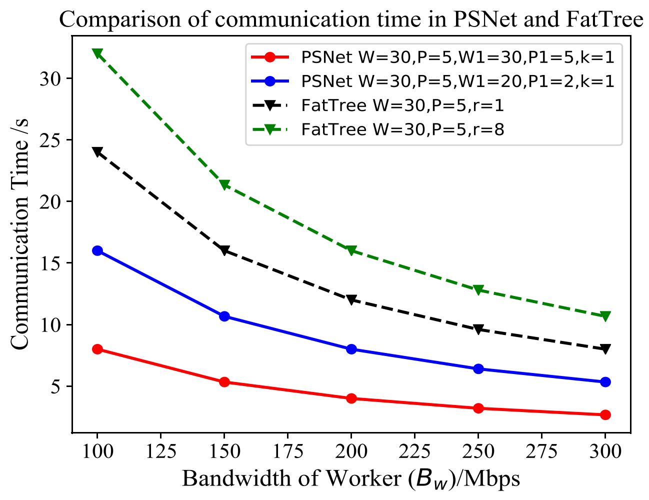


**Fig. 7.** Comparison of communication time at scale.

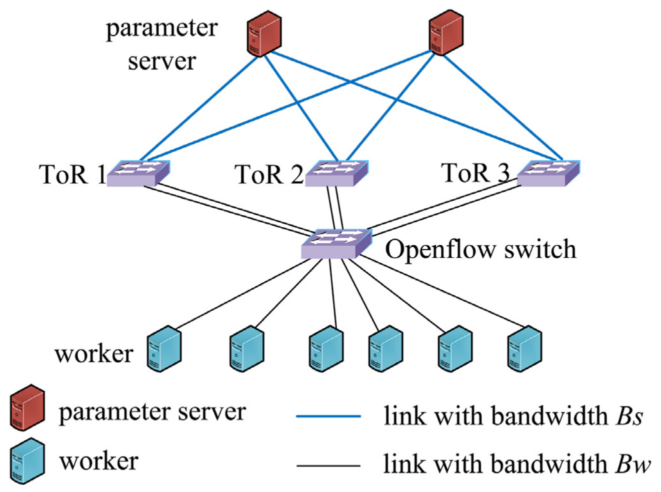


**Fig. 8.** The PSNet prototype in our experiments.

**Table 2**
Model information.

| Model | Size (MB) | Application domain | DataSet |
|---|---|---|---|
| VGG-16 | 528 | Image Classification | CIFAR 10 |
| ResNet50 | 97 | Image Classification | CIFAR 10 |

that the communication time goes down as the bandwidth of parameter servers increase (bigger $K$). Note that when $K = 2$, the bandwidth of worker has become the bandwidth bottleneck. In that case, increasing $K$ will not help to accelerate DML. Therefore, the blue line when $K = 2$ overlaps with the black one when $K = 4$. Fig. 6(b) depicts the communication time when a DML task deployed in one module and in two adjacent modules. The results validate that different node deployment schemes may lead to different communication time. That is because communication time is most affected by the most heavily loaded links.

Fig. 7 compares the communication time when a large-scale DML task is running in the two network topologies. Obviously, the PSNet can achieve better performance compared to FatTree due to larger bandwidth of parameter servers and short paths between workers and parameter servers. It can also be seen that larger oversubscription can cause more serious network congestion, leading to longer communication time. And the results are consistent with our analysis.

In conclusion, the simulation results evidence that the PSNet can achieve better performance than FatTree which is the presentative topology in industry. In the next section, we will do experiments on our testbed to verify the performance of PSNet.

## 5. Experiments and analysis

We implemented a PSNet prototype to study how DML tasks perform. Moreover, we setup a FatTree prototype for comparison in different scenarios. In our experiments, BCT is our main performance metric and we use two models, namely VGG-16 and ResNet50. The model information is shown in Table 2. In order to reduce measurement error, we compute this value by measuring several iterations continuously and then taking the average value.

### 5.1. Testbed details

In our experiments, we only test a DML task on a small scale, and there is no sophisticated circuit reconfiguration. Therefore, we use an OpenFlow switch instead of ECS and its control plane. The flow table of the OpenFlow switch is set to emulate the circuit configuration of ECS. The only difference is that the hop distance between worker and parameter server in this testbed is one longer than that in PSNet, due to the transparency of ECS to upper-level applications. However, the processing delay of the OpenFlow switch is much smaller than the BCT of a task in our experiments, hence it can be ignored. We leave the evaluation of the dynamic reconfiguration performance of PSNet for future work.

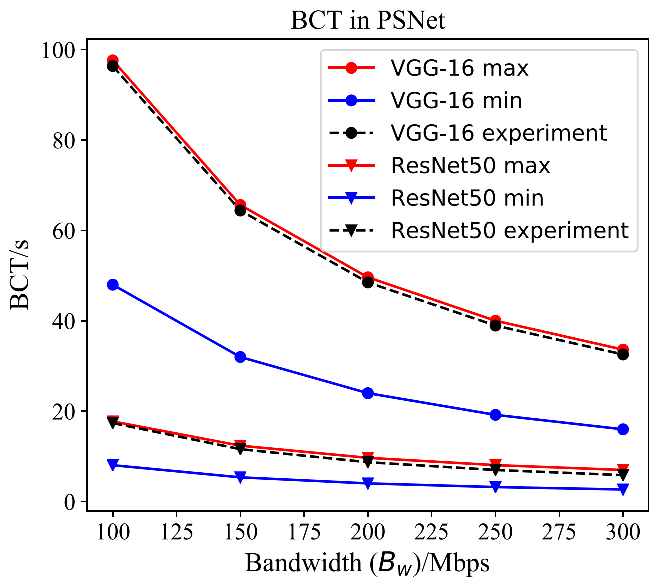
As shown in Fig. 8, in our PSNet prototype, there are 3 ToR switches, 2 parameter servers connected to each ToR switch, and 6 workers connected to the OpenFlow switch. Note that we must artificially change the flow table of OpenFlow switch to emulate the flexible circuit configuration when evaluating the impact of worker deployment schemes on the performance of a DML task.

To construct our prototype, we use 8 commodity servers, and each server is INSPUR NF6170M4 with a 6-core Inter Xeon 1.70 GHz processor, 32 GB of memory, and 4 Intel I350 1 Gbps NICs. Each server runs Ubuntu 16.04.3 with TensorFlow. Theoretically, we should use another kind of servers with GPUs in PSNet prototype. But it does not matter because we only evaluate the influence of different topologies on DML tasks in the same conditions. Hence, it is feasible as long as the servers in the two topologies are the same. Besides, our switches are Edge-core AS4610-54T with 48-GE RJ45 port and 4 × 10G SFP. In order to emulate the impact of bandwidth on performance of DML tasks, we use shaping of open vswitch (OVS) to adjust the bandwidth of switches.

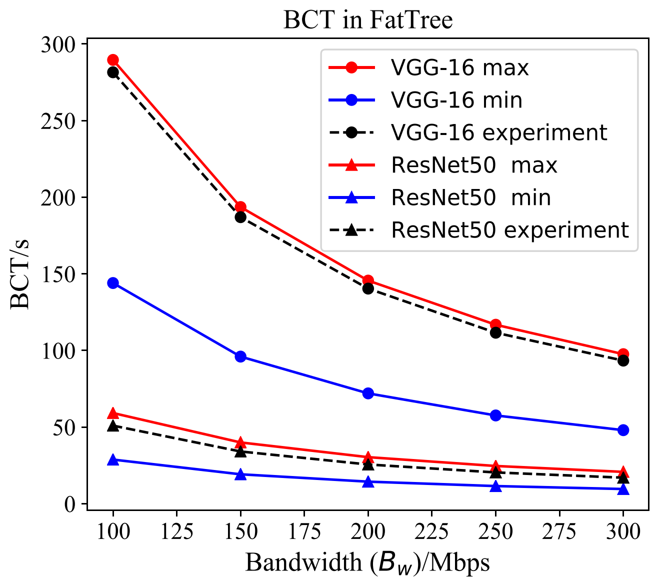
### 5.2. Performance evaluation

In the following, we use notation *worker(i,j,k)* to represent worker deployment in PSNet, which means there are *i,j,k* workers connected to the 1st, 2nd, 3rd ToR switch, respectively. Likewise, worker*(i,j,k,h)* in FatTree means there are *i, j, k, h* workers connected to the 1st, 2nd, 3rd, 4th ToR switch, respectively. Besides, other servers connected to the 4 ToR switches act as parameter servers.

**Verify the BCT modeling**: Firstly, we verify the BCT modeling through small-scale experiments when worker deployment is *worker(2,2,2)* in PSNet and *worker(0,2,2,2)* in FatTree, respectively. Considering that the worker with CPU computes slowly, we set batch size ($BS$) equal to 1. Fig. 9(a) and (b) shows the results as the bandwidth of worker changes when a task is deployed in PSNet and FatTree respectively, where theoretical maximum and



(a) BCT in PSNet.



(b) BCT in FatTree.

**Fig. 9.** BCT of VGG-16 and ResNet50 in PSNet and FatTree.

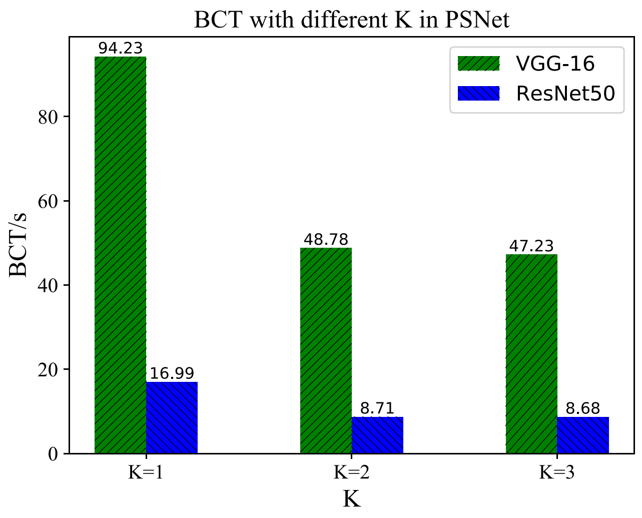


**Fig. 10.** BCT of different $K$ in PSNet.

minimum values are represented by solid lines and experimental values are represented by dotted lines. The results illustrate that the changing trend of theoretical value and the experimental value are basically the same. That verifies the validity of our BCT modeling.

**Comparison of BCT with different bandwidth of parameter server in PSNet**: PSNet provides large bandwidth for parameter servers. It is notable that different $K$ and $R$ can lead to different $B_s$. But the effects of changing $K$ and $R$ are interchangeable. For example, when $K$ is set to 2, that is the same thing as doubling $R$ when $K$ is 1. Therefore, in our test we evaluate the impact of $B_s$ on BCT of a DML task by changing $K$ for convenience. In this experiment, $W = 6$ and $P = 2$, and worker deployment is *worker(1,1,4)*. Fig. 10 plots the BCT of VGG-16 and ResNet50 respectively. The larger $K$ is, the more bandwidth of parameter server has to receive data from workers at the same time. This helps speed up data transmission and thus reduce BCT. It is worth noting that the BCT of $K = 2$ and $K = 3$ is almost the same. That is because when the bandwidth of workers becomes the communication bottleneck, it will be useless to increase bandwidth of parameter server.

**Comparison of BCT with different worker deployments in PSNet**: Fig. 11 depicts the impact of different worker deployments on BCT in PSNet. For comparison, we also evaluate that in FatTree, and the results are shown in Fig. 12. The figures illustrate that different worker deployments can lead to different BCT in both topologies. In synchronous mode, when parameter servers have received gradients from all workers, they can update the corresponding model parameters. Then workers can pull the new global model parameter from all parameter servers. In PSNet, if more workers are connected to a ToR switch than the others, it will lead to strong competition for the links between ToR switch and parameter servers. That will increase the communication time, which in turn increases the BCT. However, the use of ECS helps us to eliminate the impact of worker deployment on BCT by circuit reconfiguration. In other words, it is unnecessary for engineers to consider the worker deployment in PSNet, because it will not affect BCT when workers are distributed evenly among ToR switches in PSNet. Similarly, different worker deployments in FatTree may lead to different levels of link congestion, which directly affects the BCT. Therefore, in order to alleviate the link congestion, engineers must carefully consider the deployment of workers and parameter servers, as well as routing strategy in FatTree.

**Comparison of BCT in PSNet and FatTree**: In the above experiments, we have tested the effect of different factors of topology on BCT. In this test, we compare BCT when $K = 1$, worker deployment is $worker(2,2,2)$ in PSNet and $r = 1/2/5$, worker deployment is $worker(0,2,2,2)$ in FatTree in Fig. 13. It is observed that the bigger bandwidth oversubscription in FatTree results in longer BCT due to more serious bandwidth competition, which is consistent with our theoretical analysis. Specifically, even when FatTree is not oversubscribed, PSNet still achieves up to 1.89× and 1.92× faster than FatTree for VGG-16 and ResNet50 respectively, on account of large bandwidth of parameter servers, short and similar paths between parameter servers and workers in PSNet. Overall, these experiment results demonstrate that PSNet can fundamentally improve the training performance of DML task in PS architecture.

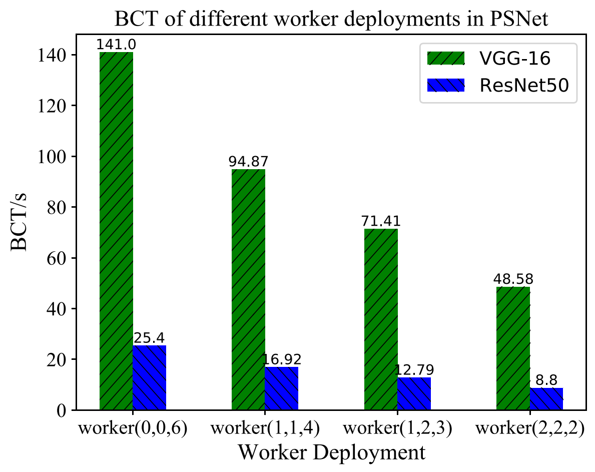


**Fig. 11.** BCT of different worker deployments when $B_w = 200$ Mbps in PSNet.

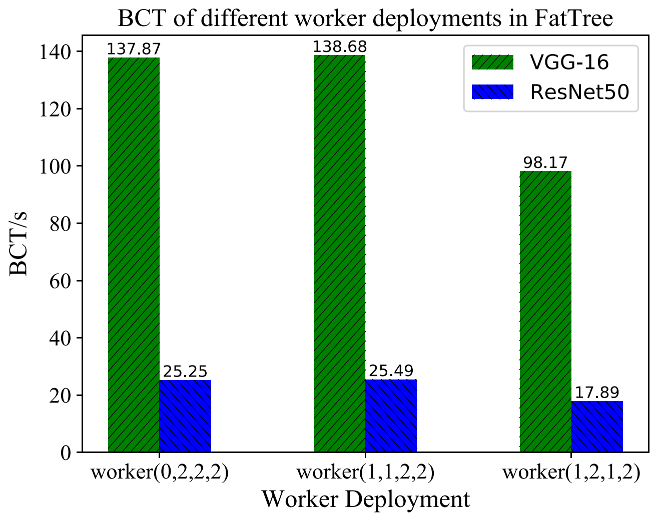
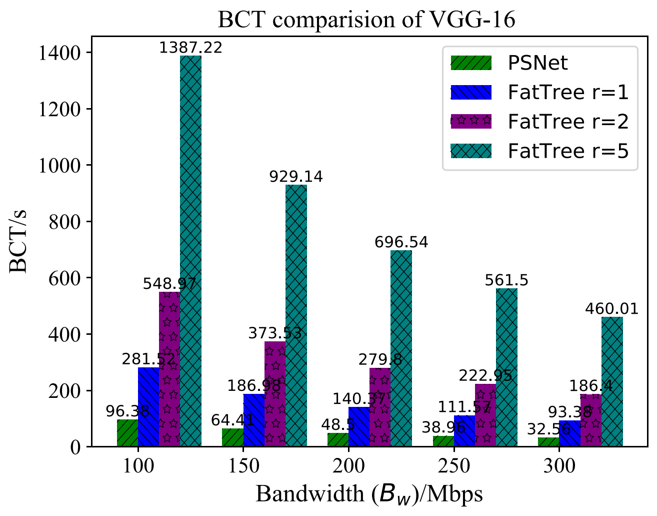
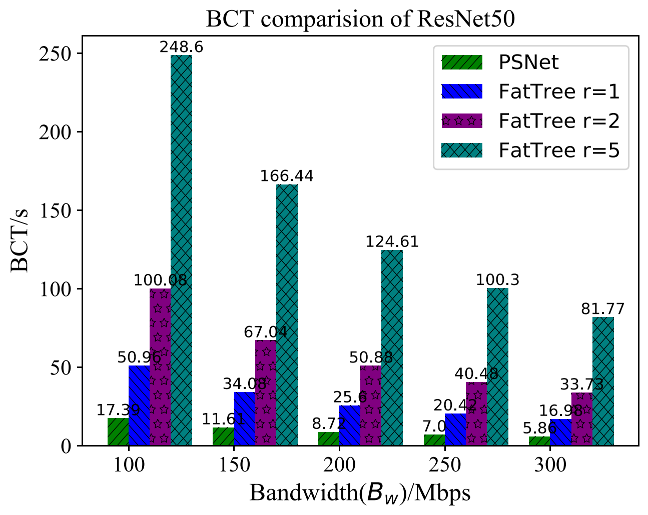


**Fig. 12.** BCT of different worker deployments when $r$=4 and $B_w = 800$ Mbps in FatTree.



(a) BCT comparision of VGG-16.



(b) BCT comparision of ResNet50.

**Fig. 13.** BCT of VGG-16 and ResNet50 in PSNet and FatTree.

## 6. Discussion

In this section, we make more discussions on PSNet.

**Feasibility and practicability.** In terms of feasibility, first we want to emphasize the transparency of ECS. The fast reconfiguration of ECS is transparent to the end-systems, and there is no need to modify the end-system operating systems or applications. Although the introduction of ECS in PSNet leads to extra cost, per-port cost of ECS is only \$3, which is much cheaper than the aggregation and core switches used in traditional networks. Besides, the modular design of PSNet yields itself naturally to fine-grained incremental expansion compared with Clos architectures and iterative defined server-centric structures, and it is also convenient to add even heterogeneous servers or switches in different modules. In terms of practicability, the construction of PSNet makes equal and short paths between workers and parameter servers, resulting shorter latency and lower iteration time, which facilitates DML training. This is impossible to achieve in the traditional Clos networks because the distance between workers and parameter servers depends on their deployments. Furthermore, the reconfigurable feature of ECS brings high error tolerance in case of node failure or link failure. Accordingly, all the features of PSNet motivate its application in practice.

**Limitations.** Compared with traditional Clos networks, the equal and shorter paths between workers and parameter servers, as well as the larger bandwidth of parameter servers, are beneficial to accelerate DML tasks. However, PSNet still has some limitations. First, since all workers are connected to the ECS, the number of workers in each module is limited by the port number of the ECS. Although multiple ECSes can be used to form a large ECS, it will reduce the fault tolerance of the network because a worker server can only connect to the corresponding ToR switch through ECS directly connected to it, not all ToR switches. Second, the network diameter will increase with the increase of modules. Therefore, we need to avoid deploying a DML task in two modules that are far apart because longer distance cause more delays, which is not conducive for speeding up DML. What is more, PSNet only applies to PS architecture based DML, not DML that based on other communication architectures, such as MPI AllReduce.

**Other open questions.** When a DML task is only placed within a module in PSNet, better performance can be achieved by evenly distributing workers to all ToR switches. However, when a DML task is deployed in two or more modules, routing algorithm is needed. The routing algorithm decides not only the configuration of ECS in each module, i.e. the connections between workers and ToR switches, but also the paths between workers and parameter servers in different modules. In addition, when multiple DML tasks need to be executed in the network at the same time, the deployments of the tasks and routing algorithms are both required. These are also our future work.

## 7. Related work

### *7.1. Distributed machine learning and PS architecture*

From computer vision to natural language processing, machine learning has proved to be crucial in solving complex problems that require to mine valuable information from the massive unstructured data [44]. In the era of big data, the powerful and complex models with $10^9$ to $10^{12}$ parameters are created [45]. Faced with such massive data and complex models, the use of DML greatly speedups the training tasks. DML typically uses data parallelism where each worker performs training based on local data, and then all workers exchange gradients to synchronize model updates. MPI AllReduce and Parameter Server (PS) [46] are two widely-used architectures for model synchronization.

In MPI AllReduce architecture, such as Binomial Tree [47], Ring [48]and Butterfly [49], all the nodes are compute nodes (i.e., workers) and all workers store a replica of model and communicate computed gradients with each other via collective communication primitives. In PS architecture, parameter servers are in charge of storing and synchronizing model, while workers perform local computation on partitioned data. MPI AllReduce architecture can only work in synchronous mode, while PS architecture can work both in synchronous mode and asynchronous mode. When in asynchronous mode, each worker computes parameter updates (i.e., gradients) and then pushes them to parameter servers independently. Once gradients from a worker have been received, a parameter servers updates its model parameters. While in synchronous mode, after parameter servers have received gradients from all workers in each iteration, the global model can be updated. The PS architecture is the most popular architecture [50] and has been widely used in machine learning frameworks thanks to its high scalability, fault tolerance and easy deployment.

In recent years, significant works have focused on PS architecture from various aspects. Some papers analyze the influence of network on training performance in PS architecture. For example, [17] verifies that network latency and throughput can affect the iteration time. Besides, some scholars study new PS-based designs. For instance, FlexPS [11] provides a flexible parallelism control in PS architecture, which can change the number of workers according to the workload. Besides, PHub [18] first analyzes the communication bottleneck and then proposes a rack-scale PS architecture by co-designed software and hardware. Moreover, some works [22,23,50,51] have studied the resource allocation and scheduling in PS architecture. Take Tiresias [50] as an example, it allocates GPUs for each job and schedules multiple jobs to minimize job completion times (JCTs) without information about jobs, such as job duration and resource requirements. In addition, [52] deals with the problem of straggler problem. However, none of these works aim to improve the efficiency of PS architecture from a physical network topological perspective.

### *7.2. Network topology*

Although some server-centric topologies have been proposed in the literature such as DCell [28] and FiConn [53], many existing network topologies [54–56] in industry still adopt a multi-layer Clos architecture which belongs to switch-centric category thanks to its easy deployment and high efficiency as well as high reliability, such as the topologies of Facebook and Google. Traditionally, these topologies adopt the fixed and symmetric network design, and the inter-layer connectivity is often oversubscribed, which is susceptible to congestion in network, leading to degradation of application performance. Besides, the hop distance between servers depends on their locations, which leads to different distances between servers. In addition, a failed upper switch can break down a large number of servers. Therefore, the extensively used multi-layer Clos topology cannot provide high performance for the PS architecture.

The recent interest on reconfigurable topologies [57] with the introduction of new communication methods such as optical transmission and wireless transmission, give rise to some dynamically changeable topologies. For most reconfigurable architectures with wireless communication, the wireless radios are often located on top of racks [34], so the number of concurrent wireless links is restricted because of strong inter-ratio interference. On the contrary, Diamond in [35] proposes to deploy wireless radios on all servers, which greatly increases the number of concurrent wireless transmission by introducing Ring Reflection Space (RRS) and a precise reflection scheme inside an RRS. However, the rooms for these wireless reconfigurable topologies need to be reshaped. Besides, the position of racks and reflectors, as well as the operation of wireless devices, should be extremely precise to reduce interference. In addition, wireless transmissions are easily disrupted by the external environment. Furthermore, the limited spectrum resources also limit the number of concurrent wireless links, which also cannot conquer the communication needs of sending data from a large number of workers simultaneously to a parameter server in the PS architecture.

Other researches, such as Helios [58], c-Through [36] and OmniSwitch [59], construct configurable network architectures by using Optical Circuit Switches (OCSes). Most of the proposals aim to provide high bandwidth to reduce congestion at locality. However, the per-port price of OCS costs up to a few hundred dollars [36], and the OCS has a long configuration delay which can be a few milliseconds. While recent efforts on fast optical switching have a much lower reconfiguration delay [30,60], they are not commercially available.

In recent years, several works [37–39] propose to use electrical circuit switches(ECSes) in topologies. Larry [37] uses a custom ECS at the ToR switch layer, so it can dynamically adapt the aggregate uplink bandwidth on each rack according to the traffic demand. The proposal in [38] connects the edge servers directly with ECS, realizing a rackless network architecture. Compared with OCS, ECS is much cheaper and has a faster configuration latency. The reconfiguration latency is only O(10) nanoseconds, and the per-port cost is only \$3 [38,39].

To sum up, most existing proposals do not take the communication characteristics of upper applications into consideration. However, the poor design of underlying physical topology could result in the performance loss of the upper applications. Therefore, we propose PSNet, a specific topology considering the communication requirements of PS architecture. In our design, we also use ECS for flexible, reconfigurable, and changeable topology due to its fast reconfiguration and low price.

## 8. Conclusion

In recent years, some papers have analyzed the impact of physical topology on DML performance. However, most existing schemes to accelerate DML ignore the impact of the underlying topology on the upper applications, limiting the performance improvement. In addition, there is no special network topology designed for DML. Therefore, in this paper, we propose to accelerate DML based on PS architecture from a topological point of view, which provides a completely new alternative approach for other scholars to speed up DML or other applications that have similar traffic characteristics to DML.

Specifically, we present PSNet, a flexible and reconfigurable modular network topology considering the communication requirements of PS architecture. In the inter-module construction, several modules are connected into a regular graph whose links have balanced load. In the intra-module construction, servers are divided into workers with high computation capability and parameter servers with multiple NICs. Besides, we use ECS to connect workers and ToR switches, achieving a reconfigurable and rackless architecture, which is beneficial for load balance and fault tolerance. In addition, the multiple NICs provide large bandwidth for parameter servers. On the whole, PSNet not only provides high performance for PS architecture, but also high fault tolerance and flexibility thanks to multiple paths between workers and parameter servers and dynamically adjustments of network by circuit reconfiguration of ECS. To validate our design, we conduct large-scale theoretical simulations and small-scale experiments on testbed, and the results show that PSNet can significantly reduce the BCT of DML tasks in comparison to FatTree. In our future work, as mentioned in Section 6, we will focus on the DML task deployments and routing algorithms in PSNet. Besides, designing new topology for DML based on MPI AllReduce architecture is also our future work.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgments

This research was partially supported by the National Key Research and Development Program of China (2019YFB1802805), PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (PCL2018KP001).

## References

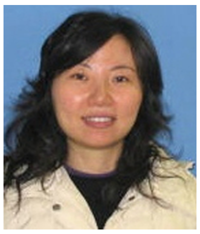
[1] L.C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, Yuille. A.L., Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs, IEEE Trans. Pattern Anal. Mach. Intell. 40 (4) (2018) 834–848.
[2] Z. Zhang, J. Geiger, J. Pohjalainen, A.E.D. Mousa, W. Jin, B. Schuller, Deep learning for environmentally robust speech recognition: An overview of recent developments, ACM Trans. Intell. Syst. Technol. (TIST) 9 (5) (2018) 1–28.
[3] G.E. Dahl, D. Yu, L. Deng, A. Acero, Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition, IEEE Trans. Audio Speech Language Process. 20 (1) (2012) 30–42.
[4] T. Young, D. Hazarika, S. Poria, E. Cambria, Recent trends in deep learning based natural language processing, IEEE Comput. Intell. Mag. 13 (3) (2018) 55–75.
[5] C. Zhang, H. Tian, W. Wang, F. Yan, Stay fresh: Speculative synchronization for fast distributed machine learning, in: 2018 IEEE 38th International Conference on Distributed Computing Systems, ICDCS, Vienna, Austria, 2018.
[6] E.P. Xing, Q. Ho, P. Xie, D. Wei, Strategies and principles of distributed machine learning on big data, Engineering 2 (2) (2016) 179–195.
[7] M. Li, D.G. Andersen, A.J. Smola, K. Yu, Communication efficient distributed machine learning with the parameter server, in: Advances in Neural Information Processing Systems, Montreal, Canada, 2014.
[8] K. Lee, M. Lam, R. Pedarsani, D. Papailiopoulos, K. Ramchandran, Speeding up distributed machine learning using codes, IEEE Trans. Inform. Theory 64 (3) (2017) 1514–1529.
[9] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, et al., Accurate, large minibatch sgd: Training imagenet in 1 hour, 2017, ArXiv preprint, arXiv:1706.02677.
[10] A. Kadav, E. Kruus, Fine-grain synchronization in data-parallel jobs for distributed machine learning, US Patent App. 15/980, 196, 2018.
[11] Y. Huang, T. Jin, Y. Wu, Z. Cai, X. Yan, F. Yang, et al., Flexps: Flexible parallelism control in parameter server architecture, Proc. VLDB Endow. 11 (5) (2018) 566–579.
[12] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, et al., Tensorflow: A system for large-scale machine learning, in: 12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI}). Savannah, GA, USA, 2016.
[13] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, et al., Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems, 2015, ArXiv preprint, arXiv:1512.01274.
[14] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, et al., Caffe: Convolutional architecture for fast feature embedding, in: Proceedings of the 22nd ACM International Conference on Multimedia, New York, NY, USA, 2014.
[15] J. Jiang, L. Yu, J. Jiang, Y. Liu, B. Cui, Angel: a new large-scale machine learning system, Natl. Sci. Rev. 5 (2) (2017) 216–236.
[16] A. Sapio, M. Canini, C.Y. Ho, J. Nelson, P. Kalnis, C. Kim, et al., Scaling distributed machine learning with in-network aggregation, 2019, ArXiv preprint, arXiv:1903.06701.
[17] M. Alan, A. Panda, D. Bottini, L. Jian, P. Kumar, S. Shenker, Network evolution for dnns, SysML doc/182 (2018).
[18] L. Luo, J. Nelson, L. Ceze, A. Phanishayee, A. Krishnamurthy, Parameter hub: a rack-scale parameter server for distributed deep neural network training, in: the ACM Symposium on Cloud Computing, Carlsbad, California, 2018.
[19] H. Zhang, Z. Zheng, S. Xu, W. Dai, Q. Ho, X. Liang, et al., Poseidon: An efficient communication architecture for distributed deep learning on {GPU} clusters, in: 2017 {USENIX} Annual Technical Conference ({USENIX} {ATC} 17), Santa Clara, CA, 2017.
[20] J.H. Park, S. Kim, J. Lee, M. Jeon, S.H. Noh, Accelerated training for cnn distributed deep learning through automatic resource-aware layer placement, 2019, ArXiv preprint, arXiv:190105803.
[21] S. Shin, Y. Jo, J. Choi, S. Venkataramani, V. Srinivasan, W. Sung, Workload-aware automatic parallelization for multi-gpu dnn training, in: ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP, Brighton, United Kingdom, 2019.
[22] G. Sun, Y. Zhang, D. Liao, H. Yu, X. Du, M. Guizani, Bus trajectory-based street-centric routing for message delivery in urban vehicular, Ad Hoc Netw. 67 (8) (2018a) 7550–7563.
[23] J.H. Park, S. Kim, J. Lee, M. Jeon, S.H. Noh, Accelerated training for cnn distributed deep learning through automatic resource-aware layer placement, 2019, ArXiv preprint, arXiv:1901.05803.
[24] S. Kim, G.I. Yu, H. Park, S. Cho, E. Jeong, H. Ha, et al., Parallax: Sparsity-aware data parallel training of deep neural networks, in: the Fourteenth EuroSys Conference 2019, Dresden, Germany, 2019.
[25] A. Singla, A. Singh, K. Ramachandran, L. Xu, Y. Zhang, Proteus: a topology malleable data center network, in: the 9th ACM SIGCOMM Workshop on Hot Topics in Networks, CA, USA, 2010.
[26] C. Avin, S. Schmid, Toward demand-aware networking: A theory for self-adjusting networks, ACM SIGCOMM Comput. Commun. Rev. 48 (5) (2019) 31–40.
[27] I. Forrest, Distributed Deep Neural Network Training: A Measurement Study, 2016.
[28] C. Guo, H. Wu, K. Tan, L. Shi, Y. Zhang, S. Lu, Dcell: a scalable and fault-tolerant network structure for data centers, ACM SIGCOMM Comput. Commun. Rev. 38 (4) (2008) 75–86.
[29] G. Sun, Z. Chen, H. Yu, X. Du, M. Guizani, Online parallelized service function chain orchestration in data center networks, IEEE Access 7 (1) (2019a) 100147–100161.
[30] W.M. Mellette, R. McGuinness, A. Roy, A. Forencich, G. Papen, A.C. Snoeren, et al., Rotornet: A scalable, low-complexity, optical datacenter network, in: the Conference of the ACM Special Interest Group on Data Communication, Los Angeles, CA, USA, 2017.
[31] D. Wu, X. Sun, Y. Xia, X. Huang, T.E. Ng, Hyperoptics: A high throughput and low latency multicast architecture for datacenters, in: 8th {USENIX} Workshop on Hot Topics in Cloud Computing, HotCloud 16, Denver, CO, 2016.
[32] G. Sun, Y. Li, H. Yu, A.V. Vasilakos, X. Du, M. Guizani, Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks, Future Gener. Comput. Syst. 91 (2019) 347–360.

[33] Y. Xiaoshan, X. Hong, G. Huaxi, L. Hao, Thor: A scalable hybrid switching architecture for data centers, IEEE Trans. Commun. 66 (10) (2018) 4653–4665.
[34] X. Zhou, Z. Zhang, Y. Zhu, Y. Li, S. Kumar, A. Vahdat, et al., Mirror mirror on the ceiling: Flexible wireless links for data centers, ACM SIGCOMM Comput. Commun. Rev. 42 (4) (2012) 443–454.
[35] Y. Cui, S. Xiao, X. Wang, Z. Yang, S. Yan, C. Zhu, et al., Diamond: Nesting the data center network with wireless rings in 3-d space, IEEE/ACM Trans. Netw. 26 (1) (2017) 145–160.
[36] G. Wang, D.G. Andersen, M. Kaminsky, K. Papagiannaki, T. Ng, M. Kozuch, et al., c-through: PArt-time optics in data centers, ACM SIGCOMM Comput. Commun. Rev. 41 (4) (2011) 327–338.
[37] A. Chatzieleftheriou, S. Legtchenko, H. Williams, A. Rowstron, Larry: Practical network reconfigurability in the data center, in: 15th {USENIX} Symposium on Networked Systems Design and Implementation, {NSDI} 18, Renton, WA, USA, 2018.
[38] D. Wu, A. Chen, T. Ng, The case for a rackless data center network architecture, in: the ACM SIGCOMM 2018 Conference on Posters and Demos, Budapest, Hungary, 2018.
[39] S. Legtchenko, N. Chen, D. Cletheroe, A. Rowstron, H. Williams, X. Zhao, Xfabric: A reconfigurable in-rack network for rack-scale computers, in: 13th {USENIX} Symposium on Networked Systems Design and Implementation, {NSDI} 16, Santa Clara, CA, 2016.
[40] A. Singh, J. Ong, A. Agarwal, G. Anderson, A. Armistead, R. Bannon, et al., Jupiter rising: A decade of clos topologies and centralized control in google's datacenter network, ACM SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 183–197.
[41] Y. Xia, X.S. Sun, S. Dzinamarira, D. Wu, X.S. Huang, T. Ng, A tale of two topologies: Exploring convertible data center network architectures with flat-tree, in: Proceedings of the Conference of the ACM Special Interest Group on Data Communication, Angeles, CA, USA, 2017.
[42] B. Bollobás, B. Béla, Random Graphs, vol. 73, Cambridge university press, 2001.
[43] A. Roy, H. Zeng, J. Bagga, G. Porter, A.C. Snoeren, Inside the social network's (datacenter) network, in: ACM SIGCOMM Computer Communication Review, London, UK, 2015.
[44] G. Sun, G. Zhu, D. Liao, H. Yu, X. Du, M. Guizani, Cost-efficient service function chain orchestration for low-latency applications in nfv networks, IEEE Syst. J. (2018).
[45] M. Li, D.G. Andersen, J.W. Park, A.J. Smola, A. Ahmed, V. Josifovski, et al., Scaling distributed machine learning with the parameter server, in: 11th {USENIX} Symposium on Operating Systems Design and Implementation, {OSDI} 14, Broomfifeld, CO, 2014b.
[46] E.P. Xing, Q. Ho, W. Dai, J.K. Kim, J. Wei, S. Lee, et al., Petuum: A new platform for distributed machine learning on big data, IEEE Trans. Big Data 1 (2) (2015) 49–67.
[47] O. Hartmann, M. Kühnemann, T. Rauber, G. Rünger, Adaptive selection of communication methods to optimize collective mpi operations, in: Workshop on Compilers for Parallel Computers, CPC, A Coruna, Spain, 2006.
[48] P. Patarasuk, X. Yuan, Bandwidth optimal all-reduce algorithms for clusters of workstations, J. Parallel Distrib. Comput. 69 (2) (2009) 117–124.
[49] R. Thakur, R. Rabenseifner, W. Gropp, Optimization of collective communication operations in mpich, Int. J. High Perform. Comput. Appl. 19 (1) (2005) 49–66.
[50] J. Gu, M. Chowdhury, K.G. Shin, Y. Zhu, M. Jeon, J. Qian, et al., Tiresias: A {GPU} cluster manager for distributed deep learning, in: 16th {USENIX} Symposium on Networked Systems Design and Implementation, {NSDI} 19, Boston, MA, USA0, 2019.
[51] Y. Bao, Y. Peng, C. Wu, Z. Li, Online job scheduling in distributed machine learning clusters, in: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, Honolulu, HI, USA, 2018.
[52] A. Harlap, H. Cui, W. Dai, J. Wei, G.R. Ganger, P.B. Gibbons, et al., Addressing the straggler problem for iterative convergent parallel ml, in: the Seventh ACM Symposium on Cloud Computing, Santa Clara, CA, USA, 2016.
[53] D. Li, C. Guo, H. Wu, K. Tan, Y. Zhang, S. Lu, Ficonn: Using backup port for server interconnection in data centers, in: IEEE INFOCOM 2009, Rio de Janeiro, Brazil, 2009.
[54] G. Sun, Y. Li, D. Liao, V. Chang, Service function chain orchestration across multiple domains: A full mesh aggregation approach, IEEE Trans. Netw. Serv. Manag. 15 (3) (2018) 1175–1191.
[55] G. Sun, Z. Xu, H. Yu, V. Chang, X. Du, M. Guizani, Toward slas guaranteed scalable vdc provisioning in cloud data centers, IEEE Access 7 (2019) 80219–80232.
[56] L. Luo, Y. Kong, M. Noormohammadpour, Z. Ye, G. Sun, H. Yu, et al., Deadline-aware fast one-to-many bulk transfers over inter-datacenter networks, IEEE Trans. Cloud Comput. (2019) 1–17.
[57] L. Luo, K.T. Foerster, S. Schmid, H. Yu, Dartree: deadline-aware multicast transfers in reconfigurable wide-area networks, in: Proceedings of the International Symposium on Quality of Service, 2019, pp. 1–10.
[58] N. Farrington, G. Porter, S. Radhakrishnan, H.H. Bazzaz, V. Subramanya, Y. Fainman, et al., Helios: a hybrid electrical/optical switch architecture for modular data centers, ACM SIGCOMM Comput. Commun. Rev. 41 (4) (2011) 339–350.
[59] Y. Xia, M. Schlansker, T.E. Ng, J. Tourrilhes, Enabling topological flexibility for data centers using omniswitch, in: 7th {USENIX} Workshop on Hot Topics in Cloud Computing, HotCloud 15, Santa Clara, CA, 2015.
[60] D. Alistarh, H. Ballani, P. Costa, A. Funnell, J. Benjamin, P. Watts, et al., A high-radix, low-latency optical switch for data centers, ACM SIGCOMM Comput. Commun. Rev. 45 (4) (2015) 367–368.

**Ling Liu** is pursuing her Doctor degree in Communication and Information System at University of Electronic Science and Technology of China. Her research interests include distributed machine learning and network topology.

**Qixuan Jin** is pursuing his Master degree in Communication and Information System at University of Electronic Science and Technology of China. His research interests include distributed machine learning and algorithms.

**Dan Wang** is pursuing her Master degree in Communication and Information System at University of Electronic Science and Technology of China. Her research interests include distributed machine learning and algorithms.


**Hongfang Yu** received her B.S. degree in Electrical Engineering in 1996 from Xidian University, her M.S. degree and Ph.D. degree in Communication and Information Engineering in 1999 and 2006 from University of Electronic Science and Technology of China, respectively. From 2009 to 2010, she was a Visiting Scholar at the Department of Computer Science and Engineering, University at Buffalo (SUNY). Her research interests include network survivability, network security and next generation Internet.

**Gang Sun** is an associate professor of Computer Science at University of Electronic Science and Technology of China (UESTC). His research interests include network virtualization, cloud computing, high performance computing, parallel and distributed systems, ubiquitous/pervasive computing and intelligence and cyber security.

**Shouxi Luo** received his B.S. degree in Communication Engineering and Ph.D. degree in Communication and Information System from University of Electronic Science and Technology of China in 2011 and 2016, respectively. From Oct. 2015 to Sep. 2016, he was an Academic Guest at the Department of Information Technology and Electrical Engineering, ETH Zurich. His research interests include data center networks and software-defined networks.