IEICE TRANS. INF. & SYST., VOL.E104–D, NO.8 AUGUST 2021

PAPER

Hybrid Electrical/Optical Switch Architectures for Training Distributed Deep Learning in Large-Scale

Thao-Nguyen TRUONG[†a)], ***Nonmember and*** **Ryousei TAKANO**[†b)], ***Member***

SUMMARY Data parallelism is the dominant method used to train deep learning (DL) models on High-Performance Computing systems such as large-scale GPU clusters. When training a DL model on a large number of nodes, inter-node communication becomes bottle-neck due to its relatively higher latency and lower link bandwidth (than intra-node communication). Although some communication techniques have been proposed to cope with this problem, all of these approaches target to deal with the large message size issue while diminishing the effect of the limitation of the inter-node network. In this study, we investigate the benefit of increasing inter-node link bandwidth by using hybrid switching systems, i.e., Electrical Packet Switching and Optical Circuit Switching. We found that the typical data-transfer of synchronous data-parallelism training is long-lived and rarely changed that can be speed-up with optical switching. Simulation results on the Simgrid simulator show that our approach speed-up the training time of deep learning applications, especially in a large-scale manner.

key words: *distributed deep learning, high performance computing (HPC), optical circuit switching, hybrid switching*

1. Introduction

In recent years, Deep Learning (DL) plays an important role in a variety of engineering applications and research fields such as language processing and computer vision. Trends in DL show an increase in training dataset sizes as well as the introduction of deeper models to improve accuracy [1], [2]. Therefore, large-scale training on High-Performance Computing (HPC) systems or clusters of GPUs is becoming common to achieve faster training time for such large models and datasets [1], [3]. Typically, in the data parallelism approach, a DL model is duplicated onto multiple computing elements, e.g, GPUs or cores. Each computing element process one portion of data in batches to learn the weights of the DL model and iteratively minimizing the loss function, i.e., using Stochastic Gradient Descent (SGD). At the end of each iteration, all the computing elements have to share their local gradients to obtain the averaged global gradients. This weight update phase requires performing a collective allreduce communication.

When the DL model becomes bigger and deeper or the number of compute nodes increases, the communication becomes a bottleneck due to the allreduce collective operation. It is because of (i) a large communication message size of the weight update phase (the number of model's parameters), e.g., up to 138 and 600 million gradients that lead to hundreds or thousands of Megabytes of message size as in VGG [4], AmoebaNetB(6,512) [5] respectively. Especially, in recent years, DL models in the area of language modeling and natural language processing are large as with billions of parameters, e.g., 8.3B in Megatron-LM [6] and 17B in Turing-NLG [7]. It is also because (ii) the modern inter-node network architectures relatively have a higher network latency and smaller link bandwidth than the intra-node architectures. For example, most of the DL-accelerated HPC systems such as ABCI and NV-cluster use Infiniband EDR (100Gbps) links for the inter-node network and NVLINK (400Gbps) for the intra-node network, respectively.

To deal with the communication challenges of large-scale distributed training DL application, several research directions have been taken into account. One direction is to optimize the collective algorithm (allreduce) for supporting large-scale training and large message size, especially considering the specific network architecture of HPC systems [8]–[10]. Another approach targets to reduce the communicated message size by using (i) fewer bits to represent the DL model, i.e., weights or gradients [11], [12], and by (ii) skipping the transfer of the trivial gradient and only share the significant gradients [13], [14]. However, there is no scientific evidence or theoretical proof that ensures those techniques can work well with any datasets and DL models. This approach faces a limitation of finding a good trade-off between the transmitted data reduction rate (compression ratio) and the DL model accuracy.

On the other hand, researchers focus on techniques to overlap communication and computation as much as possible [15], [16]. For example, Yamazaki et al. divide a DL model into some groups of layers. The communication in the weight updated phase of a group is performed at the same time with the computation in the backward phase of the next group [15]. Thus, for a given DL model, this approach meets the limitation of finding the balance point for the best (ideal) overlapping. Besides, when dividing the DL model, the communicating message size becomes smaller but it performs more collective alreduce operation. This technique leads to a consideration of latency-bound instead of only bandwidth-bound as in the conventional allreduce-algorithm for DL.

All of the mentioned approaches are targeted to deal with the issue of large message size. In this work, we aim at speeding up training DL from a network architec-

Manuscript received October 9, 2020.
Manuscript revised March 12, 2021.
Manuscript publicized April 23, 2021.
[†]The authors are with the National Institute of Advanced Industrial Science and Technology (AIST), Tokyo, 152–8550 Japan.
a) E-mail: nguyen.truong@aist.go.jp (Corresponding author)
b) E-mail: takano-ryousei@aist.go.jp
DOI: 10.1587/transinf.2020EDP7201

Copyright © 2021 The Institute of Electronics, Information and Communication Engineers

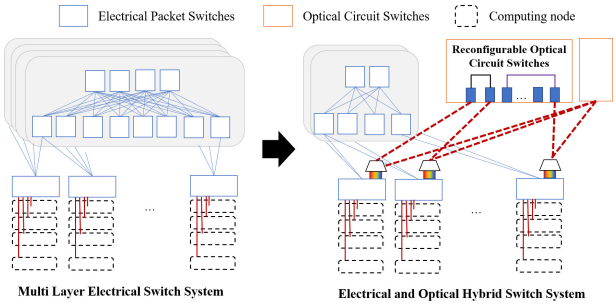


Fig. 1 Overview of our approach.

ture point of view (the second and core issue). We try to answer the question: "*is there any way to increase the link bandwidth while maintain/reduce the network latency of the inter-node network architecture?*". That is, how to provide communication with a bandwidth of 400 Gbps per link and small switch latency. In this study, we consider using hybrid electrical/optical switching systems (as shown in Fig. 1). This study is an extended version of our previous study [17]†. In which, electrical packet switches (EPS) are used for short non-persistent data flows while optical circuit switches (OCS) are used for high-bandwidth and usually long-lived communication, e.g., communication of DL application. To the best of our knowledge, this is the first work that considers the use of a hybrid switch system for parallel training DL application. Our main contributions are:

- To speed up the training process of Deep Learning applications, we proposed to use a hybrid EPS-OCS switching network instead of using the conventional EPS network for inter-node communication. It is well-known that the limitation of using the OCS in a large-scale system lays on the large reconfiguration time. We identify the typical communication pattern of DL does not require reconfiguration during training. Thus it fully takes advantages of OCS systems such as (i) larger link bandwidth, e.g., 400 Gbps links, (ii) small switching latency without reconfiguration.
- We provide a numerical analysis of the communication cost of training DL applications based on such kind of system. We also demonstrate the benefits of this approach by simulating a micro-benchmark application on a discrete-event simulator, Simgrid, on a range of typical message size and allreduce algorithm.

The paper is organized as follows: Sect. 2 provides a summary of large-scale distributed deep learning and HPC system support for DL. We then present our proposal in Sect. 3 and the simulation results in Sect. 4. Finally, we conclude in Sect. 5.

†The current version extends the other with the analysis of multi-level electrical packet switching system as well as new experiments based on the Simgrid simulator.

2. Background

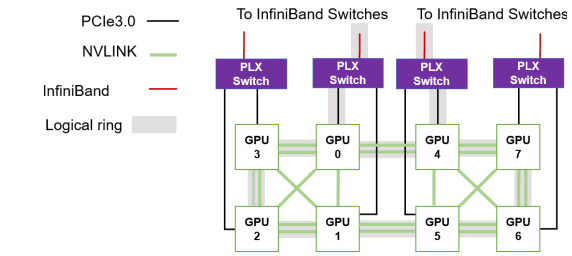
2.1 Distributed Deep Learning

There are two major strategies for parallelizing the training process of Deep Learning (DL), i.e., data and model parallelism. Because the data parallelism approach is simple and effective, it is commonly used for DL in practice. Hence, we focus on this approach in this work. The data parallelism partitions the dataset and duplicates the DL model onto p computing elements, i.e., typically GPUs. Each GPU i then performs the training (forward and backward phases) on its local data to get the local gradient of W, i.e., ∇W_i. To train the deep learning model, all the node need to update the weights by $W^{iter+1} \leftarrow W^{iter} - \rho \frac{1}{pB} \sum_{i=1}^{p} \nabla W_i$ where B is the number of data samples processed on a GPU (named Batch size), and ρ is called the learning rate. Thus, an allreduce operation need performing along all the GPUs to share the local gradients ∇W_i among GPUs.

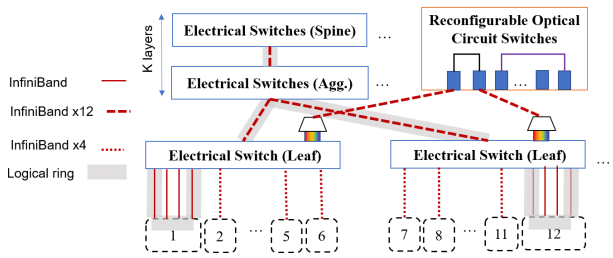
The message size of DL application is much larger than the conventional High Performance Computing (HPC) application, i.e., MBs [1] and up to GBs [5] for DL and KBs in the conventional applications. Thus, communication becomes a bottleneck for speeding up distributed training, especially in a large-scale manner where the cost for inter-node communication is much more significant than the intra-node communication [10]. In practice, ring-based allreduce algorithms are implemented for training DL, e.g., in NVIDIA Collective Communication Library (NCCL) [18] or the Baidu framework [19]. This algorithm is first proposed by Patarasuk to supports an arbitrary number of ranks for tree-based network topology such as clusters of workstations [8], [20]. Basically, this algorithm performs a reduce-scatter followed by an allgather operation. Let $P_i, i \in [1, p]$ denote the p GPUs and N is the size of sharing gradients, i.e., the size of communication message. Both two sub-operations are performed along a logical ring such as $P_0 \rightarrow P_1 \rightarrow \cdots \rightarrow P_{p-1} \rightarrow P_p \rightarrow P_0$. For each sub-operation, the N-size data in each GPUs firstly are partitioned into p segments with size $\frac{N}{p}$. Each GPU then sends 1 segment of data to the successive GPU on the logical ring, i.e., in $p-1$ steps for all the data. In each step, a process sends and receives a segment of data with size $\frac{N}{p}$. In summary, the logical ring algorithm is performed in $2(p-1)$ steps. Thus, the time of this algorithm can be modeled as the combination of (1) $2(p-1)\alpha$ where α is the time flies from a source GPU to a destination GPU, and (2) the time to inject all the message into the network $2\frac{(p-1)}{p}N\beta$ where β is the time to process one byte.

2.2 Accelerated HPC System for Deep Learning

Recently, High Performance Computing systems are increasingly adapted for training DL. For example, Piz-Diant and Cori supercomputer systems [21] have been upgraded by adding new DL-accelerated compute nodes. Each node



(a) Intra-node network



(b) Inter-node network

Fig. 2 Overview of the simulated systems.

Table 1 Latency and bandwidth of the network.

Network	Latency (second/ switch)	Link Bandwidth
Mellanox EDR IB (Spine) [24]	400×10^{-9}	12.5 GBps
Mellanox EDR IB (Leaf) [25]	90×10^{-9}	12.5 GBps
PLX swich + PCIe link [26]	110×10^{-9}	16 GBps
NVLINK [27]	$\approx 9 \times 10^{-6}$	50 GBps

includes one CPU and one GPU connected via PCIe-based interconnect. To scale the training process of DL on multiple GPUs, two directions have been considered: scaling-up in a single node and scaling-out across multiple compute nodes. In the former direction, NVIDIA proposed an on-node design with 8 (as in DGX-1 system), and 16 (as in GDX-2) P100/V00 GPUs that communicate directly through a high-speed inter-connect named NVLINK [22], and NVSwitch [23], respectively. For the inter-node scale-out approach, many AI-accelerated supercomputers have been implemented. ABCI (AI Bridging Cloud Infrastructure), and Sierra supercomputer systems combine the architecture of 4-GPUs per node and an inter-node interconnection network based on fat-tree network topology. NVIDIA proposed the NVcluster [22] that uses DGX-1 architecture for intra-node and EDR Infiniband Interconnection for the inter-node network. Figure 2 (a) illustrates an example of intra-node network architecture that uses DGX-1 as a compute node.

As mentioned, there exists a difference in latency factor and link bandwidth between intra-node and inter-node network architecture. We summarize the typical values of these factors from the actual systems in Table 1. For a large-message collective communication, inter-node communication becomes a bottleneck due to a relatively slower speed of InfiniBand link (in the comparison with NVLink and PCIe links), e.g., 12.5 GBps versus 50 GBps and 16 GBps, respectively. Many works try to diminish the performance impact of that bottleneck such as exploiting the hierarchical implementations of allreduce operations [10], [28], [29]. It is reported that such kind of hierarchical implementations helps to significantly reduce the inter-node communication time, e.g., $2\frac{(p-k)}{pk}N\beta_{inter}$ where k is the number of GPUs per compute node as in [10]. However, when the number of used compute nodes becomes bigger, the bottle-neck issue still appears. In this work, our target is to deal with this problem from the network architecture point of view. We discuss the techniques that help to achieve a larger link bandwidth for inter-node interconnect while keeping the small latency factors.

3. Hybrid EPS-OCS System for Large-Scale Distributed Deep Learning

3.1 Our Basic Approach

The interconnect network plays an important role in large-scale High Performance Computing Systems (HPC), especially in the system cost and performance. Issues of deep learning such as the increase in message size or emerging delay-sensitive applications lead to a demand for interconnection networks. That is to archive a larger throughput (link bandwidth), lower latency, and lower power consumption.

As described in Sect. 2, the typical HPC system support for training DL use electrical-packet-switch (EPS) networks to interconnect computing elements, e.g., CPUs/GPUs. As the number of computing elements becomes larger, i.e.,1088x4 GPUs and 4320x4 GPUs as in ABCI and Sierra, respectively [21], providing larger link bandwidth, e.g., 50 GBps with small end-to-end latency, e.g., less than 1 microsecond, is one of the critical challenges. To achieve 50-GBps links, it is reported that, an additional latency overhead around 100-200ns/switch is required by the use of Forwarding Error Correction (FEC) components to maintain the same bit error rate (BER) as that in the traditional interconnection network. This latency overhead significantly degrades the performance of parallel applications in HPC systems [30]. Besides, the cost of the future high-bandwidth EPS network is expected to increase due to expensive optical transceivers needed between switches optical-electrical-optical conversions at each switch). Thus, the trend of using optical switching networks for data center and HPC systems become common [31], [32].

There have been several challenges to employing optical switching technologies such as optical circuit switching (OCS) and optical packet switching (OPS) in existing electrical packet switching (EPS). Several optical packet switching architectures have been described and demonstrated such as based on using wavelength-selective switch (WSS) and wavelength-tunable converter (TWC) and an array waveguide grating router (AWGR) [32]. OPS is capable of switching traffic with switching times less than 10 ns [31], [33] but this technology still has several fundamen-

tal problems, especially the lack of flexible optical buffering. Therefore, the survey in [33] emphasizes that several proposed architectures involving OPS still need buffering in the electronic domain. It hence requires extra optical-electrical (OE) and electrical-optical (EO) conversions. That significantly increases power consumption and introduces limitations for the capacity upgrade.

In contrast, optical circuit switching (OCS) can be relatively inexpensive and power-efficient compared to OPS because OCS manipulates the light beams without any O-E-O conversion are needed. The main drawback of this OCS network is the relatively large latency for reconfiguring the connection between switches e,g., approximate 10 microseconds reported in [32], [34].

In this work, we propose to use the OCS network for long-lived communication as those of training DL. Note that, communication in large-scale deep learning is mainly used for performing the collective allreduce operation. Implementation of this communication in practical DL libraries and frameworks such as NCCL [18] or Baidu [19] is based on the ring-based algorithm. In which, one process/GPU will communicate with only one other GPU, i.e.., the successive one on the logical ring, all the time. Thus, for inter-node communication via the OCS network, the system performs the OCS configuration only one time, i.e., no reconfiguration is needed. Thus, this high reconfigure latency overhead will not be a performance impediment. For other small (message-size) flows/collective communication operation (which may not be based on a ring-based algorithm), we propose to use EPS network.

3.2 Hybrid Switching Network Architecture

Figure 2 (b) shows our inter-node electrical and optical hybrid switch system. In which each computing node connects to one electrical switch (named leaf switch). Those leaf switches can be Top of Rack (ToR) switches when implemented in the real server room like in the NVLINK systems [22] or several leaf switches are packaged into one rack, e.g., 2 leaf switch as in ABCI system. We then construct a hybrid switch system to connect those leaf switches. The EPS network is constructed similar to the conventional multi-layer EPS network, e.g., Clos or fat-tree topology. The network is organized into k stages or layers of switches including aggregation switches and the Spine switches. By contrast, the OCS network consists of $k = 2$ layer, i.e., spine switches because reducing switching stages by using large port count OCS switches helps to reduce the number of transceivers and interconnection fibers as reported in [35]. The number of optical switches and port per switch can be varied based on the number of compute nodes.

The mentioned hybrid EPS-OCS architecture has been proposed to use in the datacenter as mentioned in [32], [33] and use for HPC computer systems [31], [35]. These prior works have already considered the system cost, power consumption as well as network performance. In this work, our target is to identify the performance of such a system in terms of speeding up the training process of the DL application. In the following, we discuss the cost of communication for such kind of new application.

3.3 Communication Cost

In this section, we use the α-β model to analyze the communication cost of ring-based allreduce algorithm on EPS and OCS networks. We assume that at a given time, a process/GPU can send and receive only one message. The cost of sending one message with size N from source to destination with the absence of contention is modeled by $\alpha + N\beta$ where α is the total time taken for a message transferred from source to the destination and β is the time to inject one bye of data into the network. Typically, β is the inverse number of smallest link bandwidth on the routing path, e.g., the bandwidth of inter-node InfiniBand EDR cables. As mentioned in Sect. 2, at each iteration of DL, the weight update phase perform a flat ring-based allreduce operation which requires $2(p-1)$ steps that cost $2(p-1)\alpha + 2(p-1)\frac{N}{p}\beta$ where p is the number of processes (or GPUs). Let D is the size of the training dataset and B is the number of training samples processed in one GPU at each iteration, the total communication time of training to cover all the data samples (one epoch) with $\frac{D}{B.p}$ iterations become:

$$T_{comm} = \frac{2(p-1)D}{B.p}\left(\alpha + \frac{N}{p}\beta\right) \quad (1)$$

In the conventional EPS network (k-layers tree-based network topology), the latency factor α is the combination of total switching latency and the end-to-end propagation delay (cable latency). The switching latency depends on the number of intermediate switches on the routing path between source and destination, e.g., $2(k-1)$ switches and one spine switch with L_l and L_s second for switching, respectively. On the other hand, the propagation delay is in proportional to the total cable length on the routing path L_c. Thus, the communication cost of the EPS network becomes:

$$T_{EPS} = \frac{2(p-1)D}{B.p}\left((2(k-1)L_l + L_s + L_c) + \frac{N}{p}\beta_{EPS}\right) \quad (2)$$

The switching latency of an OCS switch includes the time of configuration L_{conf} and circuit switching time of the spine switch which is trivial so that we assume it to be ≈ 0 seconds. Since communication of DL does not require the reconfiguration and the OCS network has only one layer, the communication cost of the OCS network becomes:

$$T_{OCS} = L_{conf} + \frac{2(p-1)D}{B.p}\left((2L_l + L_c) + \frac{N}{p}\beta_{OCS}\right) \quad (3)$$

The OCS system outperforms the EPS system when:

$$\Delta = T_{OCS} - T_{EPS} < 0$$
$$L_{conf} - \frac{2(p-1)D}{B.p}\left(2(k-2)L_l + L_s + \frac{N}{p}(\beta_{EPS} - \beta_{OCS})\right) \quad (4)$$
$$< 0$$

When comparing EPS and OCS systems with the same link-bandwidths, i.e, $\beta_{EPS} = \beta_{OCS}$:

$$\Delta = L_{conf} - \frac{2(p-1)D}{B.p}(2(k-2)L_l + L_s) < 0 \quad (5)$$

Equation (5) shows that the absolute benefit gained from using OCS systems depend on the dataset, e.g., number of samples D, memory capacity of GPU, e.g, number of sample per GPU B, as well as the network architecture, e.g., number of switch's level k, and switch latency L_l, L_s. When the bigger size of training dataset or small batch size due to the memory limitation, the configuration time in OCS network contributes less in the total training time that leads to the convergence to the ideal use case of OCS network. Equation (5) also implies that the benefit of using OCS systems does not depend on the number of GPUs p involved into the training process (when p is big enough, i.e., $\frac{(p-1)}{p} \approx 1$). It is important to remind that in the comparison with a OCS system, a EPS system with the same link-bandwidth requires much higher cost (for transceivers) and power consumption [31], [32], [35]. Thus, performance improvement by using an OCS system also implicitly comes from the increase of link bandwidth, e.g., contribute by $\frac{N}{p}(\beta_{EPS} - \beta_{OCS})$ in Eq. (4). This contribution depends on the size of DL models (number of parameters) and the number of involved GPUs.

4. Evaluation

4.1 Methodology

In this section, we use the discrete-event simulator Simgrid framework version 3.21 [36] to evaluate the performance of an allreduce micro-benchmark that is also used in [10]. We vary the data size N as well as the allreduce algorithms. We aim to analyze the communication time of each allreduce algorithm with a very large message size from several to hundreds of MBs as in DL workloads and with a different parallel scale. Because typical DL libraries and frameworks use ring-based algorithm [18], [19], we select a flat ring-based algorithm [8] (*lr*) and a hierarchical ring-based algorithm [10] (*lr_lr*). We then evaluate the execution time of training a real deep learning model.

In this work, we compare our hybrid switch system with the conventional electrical switch systems that are used in the modern DL-accelerated HPC systems such as ABCI, Sierra, and NV-Cluster. Our target system is shown in Fig. 2. We assume that each compute node includes 8 GPUs arranged in a hypercube mesh as in DGX-1 system [22]. In which each GPU has 6 NVLinks, that enable to construct a ring with two NVLinks per connection along the ring, i.e., following the 0-3-2-1-5-6-7-4 order (the gray shapes in Fig. 2). This design helps to improve the efficiency of collective algorithms by double the bandwidth of the intra-communication. For the inter-node communication, our hybrid switch systems route the message through an optical switch system with a high link bandwidth, e.g., 50 GBps (named as HYBRID). We consider the conventional electrical switch system which uses a lower link bandwidth, e.g., 12.5 GBps, as the baseline (named as BASE). We also estimate the case of an electrical switch system with a high link bandwidth as a reference (ELEC). Because NVIDIA recently introduce the NVSwitch in their DGX-2 systems [23], and NVLINKs are designed to connect CPU and GPU in the Summit supercomputer system, we predict that the use of PCIe interconnect between a GPU and outside Infiniband, e.g., NIC, will be soon replaced by NVLINK technique. Thus, we consider that case in our evaluation (ELE_NV and HYB_NV). Table 2 summary our simulation configurations. It is important to mention that the results in this section do not focus on comparing the performance of the hybrid switch systems (HYBRID, HYB_NV) with the same link-bandwidth electrical switch systems (ELEC, ELEC_NV). The main benefit of using a hybrid switch system in this case, is in the construction cost and power consumption [35]. Instead, we study how system performance varies when increasing the link bandwidth (which could be archived by implementing a hybrid switch system).

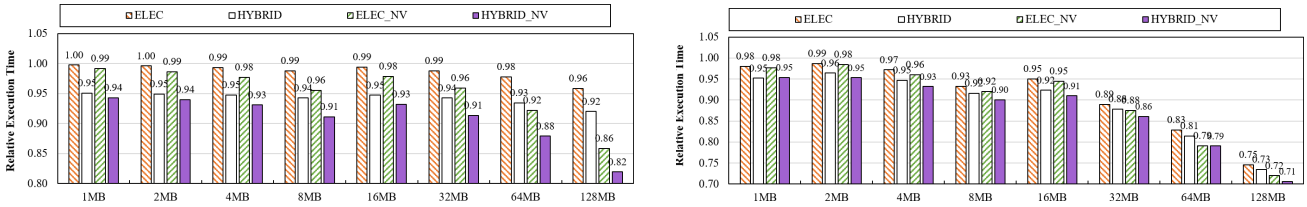
Table 2 Network configuration for SimGrid simulation.

	Intra-node switching and link bandwidth	**Inter-node switching and link bandwidth**
BASE	PLX + PCIe (16 GBps)	EPS (12.5 GBps)
ELEC	PLX + PCIe (16 GBps)	EPS (50 GBps)
HYBRID	NVSwitches (50 GBps)	OCS (50 GBps)
ELE_NV	PLX + PCIe (16 GBps)	EPS (50 GBps)
HYB_NV	NVSwitches (50 GBps)	OCS (50 GBps)

For all the architectures, we set the leaf, spine, and PLX switching latency to 90, 400, and 110 nanoseconds per switch, respectively (summary in Table 1). We pick up those values from the latency of actual Infiniband switches. We use NVIDIA Tesla V100 that can reach 15.7 single-precision FLOPS. We configure Simgrid with the minimal route between GPUs and compute nodes. We also set the configuration time of the OCS switch to 10us as reported in [32], [34]. We assume the total cable length between all pairs of compute node is 60m [31], and 5ns/m for optical cable. We also assume that hybrid switch systems are constructed with only two stage fat-tree, e.g., $k = 2$, while that is varied for electrical switch systems. We set $k = 2$ for those in the result of Sect. 4.2 and Sect. 4.3.

4.2 Allreduce Micro-Benchmark Result

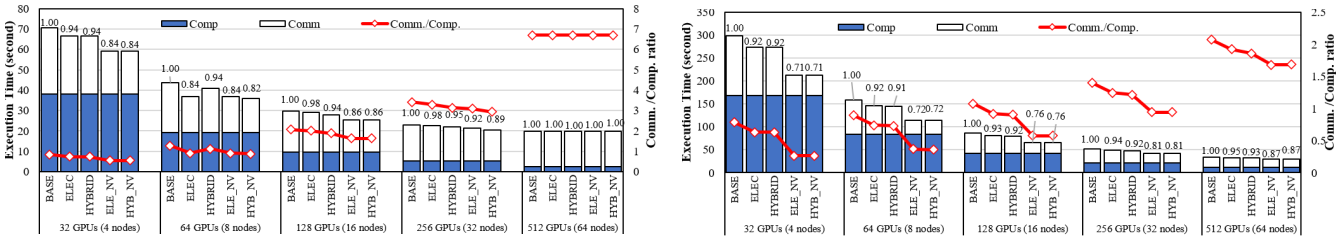
Figure 3 (a) and 3 (b) shows the relative execution time of the allreduce benchmark application versus message size that simulated on a 256-GPUs system with flat and hierarchical ring-based allreduce algorithms, respectively. The values are normalized to those of the baseline (BASE). Through our analysis in Eqs. (2) and (3), we expect that the HYBRID (or HYB_NV) systems are always better than the ELEC (ELE_NV) systems that use the same inter-node link bandwidth and allreduce algorithm. For example, when the message size is changed, HYBRID outperforms the ELEC



(a) Flat ring-based algorithm *lr*

(b) Hierarchical ring-based algorithm *lr_lr*

Fig. 3 Relative Execution time of allreduce micro-benchmark versus message size simulated on a 256-GPU system. Values are normalized to those of the baseline (BASE).



(a) ResNet50.

(b) VGG16.

Fig. 4 Simulated training time in 1-epoch of ResNet-50 and VGG16. Red line show communication/computation ratio. The label at each column show its relative execution time to those of the baseline (BASE).

from 4-5% in the case of the flat ring-based algorithm because it removes the spine switch latency L_s from communication cost and uses fewer intermediate switches on the routing path, e.g., 2-level tree instead of a k-level tree. The benefit gains from that latency differences will become larger when the routing path in the OCS network becomes longer, i.e., the use of higher tree-based network topology for larger network architecture such as 3-level fat-tree topology in ABCI system.

Secondly, all the hybrid systems significantly speeding up the allreduce benchmark, e.g., reduce up to 8% and 27% in the comparison with the baseline (BASE) when flat and hierarchical ring-based algorithms are used, respectively. This is an expected result due to the improvement of the inter-node link bandwidth, i.e., from 12.5 GBps to 50 GBps. Besides, it is necessary to note that only increasing the inter-node link bandwidth (HYBRID) will not release the inter-node communication bottleneck because of the use of PLX switches and PCIe links. We found that the future architecture should use the NVSwitch and NVLink to fully utilize the benefit of the OCS system (the HYB_NV).

4.3 Real Workload Result

In the following, we investigate the overall performance of our proposed system with real Deep Learning (DL) workload such as VGG [4] and ResNet50 [37] by numerically estimate the number of parameters and operations required for each layer and then simulated it on Simgird. The simulation does not run the actual code for training the DL models. It just estimates the time for performing the same number of floating-point operations as those required by each layer,

Table 3 Overview of DL models used in experiments.

Model	Dataset	Parameters	Sample/GPU
ResNet-50 [37]	ImageNet	> 25M	32
VGG16 [4]	1,280,000 samples	> 169M	64

i.e., via SMPI_SAMPLE_FLOPS function. We evaluate the training process of our targeted DL models (as shown in Table 3) when varying the number of GPUs and fixing the number of samples per GPU, i.e., weak scaling. For example, the training process of ResNet-50 needs $\frac{1280000}{32\times256} = 157$ iterations per epoch in the case of 256 GPUs.

Figure 4 illustrates the simulated training time in 1-epoch of ResNet-50 and VGG16 by using the flat ring-based algorithm *lr*. We also show in detail the breakdown of training time (columns) as well as the communication/computation ratio (red line) for understanding the effect of selected network architecture on the overall performance.

As expected, when the number of involving GPUs becomes bigger, the total training time decrease. It is caused by the reduction of the number of iteration/ epoch (linearly) although the communication time/iteration increase while the computation time/iteration does not change (weak-scaling). The communication speedup leads to the reduction of communication over computation time ratio and hence it will be helpful to apply the communication/computation overlapping techniques [15], [16].

Considering performance when changing the network architectures, the relative relation between our targeted configurations is not changed. The HYB_NV always achieves the best performance because the computation time is not changed but communication time is significantly reduced. Similar to the result in Sect. 4.2, the speedup of the HY-

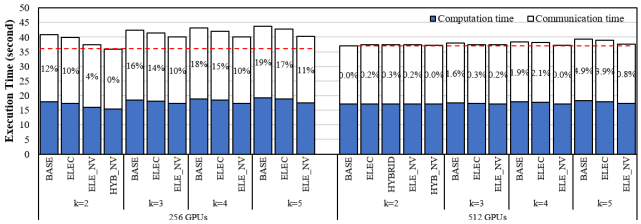


Fig. 5 Simulated training time in 1-epoch of ResNet-50 with the k-stages electrical switch systems. The label inside each column shows the difference between execution times of those and the hybrid switch system (HYB_NV, $k = 2$) on percentage. We use red dash line to mark the time of the hybrid switch system.

BRID systems over the baseline (BASE) becomes bigger when the size of DL models (number of parameters) is bigger. For instance, in the case of 32 GPUs, HYB_NV needs only 83% and 71% of training time in the comparison with BASE in the case of ResNet-50 and VGG16, respectively. These results imply that most of the communication time in those cases are dominated by the bandwidth-factor in Eq. (4), i.e., $\frac{N}{p}(\beta_{EPS} - \beta_{OCS})$. As a result, when the number of GPUs p increases, the benefit gained from the Hybrid system becomes smaller. This result reflects our analysis and communication cost model in the previous section. It is interesting to show in the case of 512 GPUs, the total training time of ResNet50 of all the configurations are quite similar. This is the point where the contribution of bandwidth-factor in Eq. (4) is small enough so that most of the time reduction comes from the latency factor, i.e., $2(k-2)L_l + Ls$. Because we set $k = 2$ in this evaluation, such kind of latency factor is also small. When the model size is bigger, such kind of breaking point also increases. For example, unlike the case of ResNet50, hybrid switch systems still outperform the baseline in the case of VGG16, 512 GPUs. Saying that this breaking point is still big enough to be satisfied in most practical large-scale situations, especially when the DL model sizes are bigger and bigger as mentioned in Sect. 1.

Figure 5 shows the difference between execution times of ResNet50 on k-stages electrical switch systems and the hybrid switch system (HYB_NV, $k = 2$) on percentage. The result shows that, even in the systems which are constructed based on the higher stages-tree, the impact of the latency factor on the overall training time is still trivial. This impact will become more significant when training in a low-bandwidth systems such as in commodity data centers or clusters with 10Gbps links.

5. Conclusion

Communication is the major bottleneck of training deep learning (DL) on large-scale High Performance Computing (HPC) or GPU cluster system. At each training iteration, all the computing elements, i.e., process, CPU or GPU, have to share their local gradients of weight to calculate the global gradients that used in the weight update phase. This leads to collective communication, i.e., an allreduce operation. As the deep learning models become bigger and deeper due to accuracy demands, communication becomes a critical bottleneck especially as the message sizes become bigger. Besides, the relative small link bandwidth of the inter-node network (smaller than the intra-node network) limits the scaling of training deep learning. Many active approaches have been proposed to cope with this problem including (i) optimizing the allreduce algorithm, (ii) reducing the communication message size, and (iii) overlapping the communication and computation. All of these approaches target to deal with the large message size issue while diminishing the effect of the limitation of the inter-node network. In this work, we target to improve the inter-node interconnection network with a higher link bandwidth, e.g., 50GBps rather than 12.5GBps and lower switch latency by using hybrid electrical packet switching (EPS) and optical circuit switching (OCS) system. We show that by using OCS network in a proper way for DL traffic pattern, we can fully take the advantages of OCS system including larger link bandwidth, small switching latency without reconfiguration and potentially lower cost and power consumption. Simulated result in a discrete-event simulator, Simgrid, on a range of typical message size and allreduce algorithm show that using hybrid switch system helps to speed up the training process of deep learning application, especially in a large-scale manner.

Acknowledgments

This paper is based on results obtained from a project, JPNP16007, commissioned by the New Energy and Industrial Technology Development Organization (NEDO).

References

[1] T. Ben-Nun and T. Hoefler, "Demystifying parallel and distributed deep learning: An in-depth concurrency analysis," CoRR, vol.abs/1802.09941, 2018.
[2] Y. You, Z. Zhang, C.-J. Hsieh, J. Demmel, and K. Keutzer, "ImageNet Training in Minutes," Proceedings of the 47th International Conference on Parallel Processing (ICPP 2018), New York, NY, USA, pp.1:1–1:10, ACM, 2018.
[3] Y. You, A. Buluç, and J. Demmel, "Scaling deep learning on GPU and knights landing clusters," Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp.1–9, ACM, 2017.
[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.
[5] Y. Huang, Y. Cheng, D. Chen, H. Lee, J. Ngiam, Q.V. Le, and Z. Chen, "GPipe: Efficient Training of Giant Neural Networks using Pipeline Parallelism," CoRR, vol.abs/1811.06965, 2018.
[6] M. Shoeybi et al., "Megatron-lm: Training multi-billion parameter language models using model parallelism," ArXiv, vol.abs/1909.08053, 2019.
[7] "Turing-NLG: A 17-billion-parameter language model by Microsoft." https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/ [01 April 2020].
[8] P. Patarasuk and X. Yuan, "Bandwidth efficient all-reduce operation on tree topologies," IEEE International Parallel and Distributed Processing Symposium, IPDPS 2007, pp.1–8, IEEE, 2007.
[9] M. Bayatpour, J.M. Hashmi, S. Chakraborty, H. Subramoni, P.

Kousha, and D.K. Panda, "Salar: Scalable and adaptive designs for large message reduction collectives," IEEE International Conference on Cluster Computing, CLUSTER 2018, Belfast, UK, pp.12–23, 2018.
[10] T.T. Nguyen, M. Wahib, and R. Takano, "Hierarchical distributed-memory multi-leader mpi-allreduce for deep learning workloads," Sixth International Symposium on Computing and Networking, CANDAR Workshops 2018, Takayama, Japan, pp.216–222, IEEE, 2018.
[11] F. Seide, H. Fu, J. Droppo, G. Li, and D. Yu, "1-bit stochastic gradient descent and application to data-parallel distributed training of speech DNNs," Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH 2014), pp.1058–1062, Sept. 2014.
[12] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "TernGrad: Ternary Gradients to Reduce Communication in Distributed Deep Learning," Advances in Neural Information Processing Systems 30, ed. I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, pp.1509–1519, Curran Associates, Inc., 2017.
[13] N. Strom, "Scalable distributed DNN training using commodity GPU cloud computing," Sixteenth Annual Conference of the International Speech Communication Association (Interspeech 2015), 2015.
[14] F. Sattler, S. Wiedemann, K. Müller, and W. Samek, "Sparse binary compression: Towards distributed deep learning with minimal communication," CoRR, vol.abs/1805.08768, 2018.
[15] M. Yamazaki, A. Kasagi, A. Tabuchi, T. Honda, M. Miwa, N. Fukumoto, T. Tabaru, A. Ike, and K. Nakashima, "Yet Another Accelerated SGD: ResNet-50 Training on ImageNet in 74.7 seconds," CoRR, vol.abs/1903.12650, 2019.
[16] N. Dryden, N. Maruyama, T. Moon, T. Benson, A. Yoo, M. Snir, and B. Van Essen, "Aluminum: An asynchronous, gpu-aware communication library optimized for large-scale training of deep neural networks on hpc systems," 2018 IEEE/ACM Machine Learning in HPC Environments (MLHPC), pp.1–13, Nov. 2018.
[17] T.T. Nguyen and R. Takano, "On the feasibility of hybrid electrical/optical switch architecture for large-scale training of distributed deep learning," 2019 IEEE/ACM Workshop on Photonics-Optics Technology Oriented Networking, Information and Computing Systems (PHOTONICS), pp.7–14, 2019.
[18] "NVIDIA Collective Communications Library (NCCL), Multi-GPU and multi-node collective communication primitives." https://developer.nvidia.com/nccl, accessed April. 1. 2019.
[19] "Baidu Allreduce." https://github.com/baidu-research/baidu-allreduce, accessed April. 1. 2019.
[20] P. Patarasuk and X. Yuan, "Bandwidth optimal all-reduce algorithms for clusters of workstations," Journal of Parallel and Distributed Computing, vol.69, no.2, pp.117–124, 2009.
[21] Top 500 Supercomputer Sites. http://www.top500.org/
[22] NVIDIA, "White Paper NVIDIA DGX-1 System Architecture, The Fastest Platform for Deep Learning." https://www.azken.com/images/dgx1_images/dgx1-system-architecture-whitepaper1.pdf, accessed April. 1. 2019.
[23] A. Ishii, D.F. withEric Anderson, B. Dally, G.D. Dennison, M. Hummel, and J. Schafer, "Nvswitch and dgx-2 nvlink-switching chip and scale-up compute server," 2018 IEEE Hot Chips 30 Symposium (HCS), pp.1–30, Aug. 2018.
[24] Mellanox, "CS7500 InfiniBand EDR 100Gb/s Switch System." http://www.mellanox.com/related-docs/prod_ib_switch_systems/pb_cs7500.pdf, accessed April. 1. 2019.
[25] Mellanox, "SB7800 InfiniBand EDR 100Gb/s Switch System." http://www.mellanox.com/related-docs/prod_ib_switch_systems/pb_sb7800.pdf, accessed April. 1. 2019.
[26] PLX Technology, "Product Brief: PEX 8716, PCI Express Gen 3 Switch, 16 Lanes, 4 Ports.." https://docs.broadcom.com/docs/12351846, accessed April. 1. 2019.
[27] A. Li, S.L. Song, J. Chen, J. Li, X. Liu, N.R. Tallent, and K.J. Barker, "Evaluating Modern GPU Interconnect: PCIe, NVLink, NV-SLI, NVSwitch and GPUDirect," CoRR, vol.abs/1903.04611, 2019.
[28] A.A. Awan, K. Hamidouche, J.M. Hashmi, and D.K. Panda, "S-caffe: co-designing MPI runtimes and caffe for scalable deep learning on modern GPU clusters," ACM Sigplan Notices, vol.52, no.8, pp.193–205, 2017.
[29] A.R. Mamidala, G. Kollias, C. Ward, and F. Artico, "MXNET-MPI: Embedding MPI parallelism in Parameter Server Task Model for scaling Deep Learning," arXiv preprint arXiv:1801.03855, 2018.
[30] T.T. Nguyen, H. Matsutani, and M. Koibuchi, "Low-reliable low-latency networks optimized for hpc parallel applications," 2018 IEEE 17th International Symposium on Network Computing and Applications (NCA), pp.1–10, Nov. 2018.
[31] K.J. Barker, A. Benner, R. Hoare, A. Hoisie, A.K. Jones, D.K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker, "On the feasibility of optical circuit switching for high performance computing systems," SC '05: Proceedings of the 2005 ACM/IEEE Conference on Supercomputing, p.16, Nov. 2005.
[32] C. Kachris, K. Kanonakis, and I. Tomkos, "Optical interconnection networks in data centers: recent trends and future challenges," IEEE Communications Magazine, vol.51, no.9, pp.39–45, Sept. 2013.
[33] J. Chen, Y. Gong, M. Fiorani, and S. Aleksic, "Optical interconnects at the top of the rack for energy-efficient data centers," IEEE Communications Magazine, vol.53, no.8, pp.140–148, Aug. 2015.
[34] K.I. Kitayama, Y.C. Huang, Y. Yoshida, R. Takahashi, T. Segawa, S. Ibrahim, T. Nakahara, Y. Suzaki, M. Hayashitani, Y. Hasegawa, Y. Mizukoshi, and A. Hiramatsu, "Torus-topology data center network based on optical packet/agile circuit switching with intelligent flow management," J. Lightwave Technol., vol.33, no.5, pp.1063–1071, March 2015.
[35] K. Sato, "Realization and application of large-scale fast optical circuit switch for data center networking," 2017 European Conference on Optical Communication (ECOC), pp.1–3, 2017.
[36] SimGrid: Versatile Simulation of Distributed Systems. http://simgrid.gforge.inria.fr/, accessed April. 1. 2019.
[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," Proceedings of the IEEE conference on computer vision and pattern recognition, pp.770–778, IEEE, 2016.



Thao-Nguyen Truong received the BE and ME degrees from Hanoi University of Science and Technology, Hanoi, Vietnam, in 2011 and 2014, respectively. He received the Ph.D. in Informatics from the Graduate University for Advanced Studies, Japan in 2018. He is currently working at AIST-Tokyo Tech Real World Big-Data Computation Open Innovation Laboratory (RWBC-OIL), National Institute of Advanced Industrial Science and Technology (AIST), where he focuses on the topics of communication for HPC system and beyond.



Ryousei Takano is a senior research scientist of the Institute of Advanced Industrial Science and Technology (AIST), Japan. He received his Ph.D. from the Tokyo University of Agriculture and Technology in 2008. He joined AXE, Inc. in 2003 and then, in 2008, moved to AIST. His research interests include operating systems and distributed parallel computing. He is currently exploring an operating system for heterogeneous accelerator clouds.