A Novel Distributed Deep Learning Training Scheme Based on Distributed Skip Mesh List

Masaya Suzuki\textsuperscript{a)}, Kimihiro Mizutani

Department of Informatics, Kindai University 3-4-1 Kowakae, Hirashiki Osaka, Osaka, Japan
\textsuperscript{a}) suzuki.masaya@kindai.ac.jp

Abstract: Distributed large scale neural networks are widely used in complicated image recognition and natural language processing among several organizations, but it takes much maintenance cost for keeping the nodes (i.e., computation servers) performance and their network topology in a churn environment that nodes insertion/deletion occurs frequently. To reduce the cost in the environment, we proposed Distributed Skip Mesh List Architecture, which can provide high stability against node insertion/deletion and automatic node management to a distributed large scale neural network management. In the evaluation, we confirmed that it reduces the maintenance cost (e.g., transmission messages for managing nodes) with high stability.

Keywords: Distributed Algorithm, Overlay Network, Deep Learning

Classification: Network system

References

[1] K. Alexander, et al. "Big Transfer (BiT): General Visual Representation Learning," ArXiv Preprint ArXiv:1912.11370, 2019.
[2] D. Jacob, et al. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding," in Proc. the 2019 Conference of the North American section of the Association for Computational Linguistics, pp. 4171–4186, 2019.
[3] B. Tal, et al. "Demystifying Parallel and Distributed Deep Learning: An In-Depth Concurrency Analysis," ACM Computing Surveys, vol. 52, no. 4, pp. 1–43, 2019.
[4] D. Jeffrey, et al. "Large Scale Distributed Deep Networks." Advances in Neural Information Processing Systems 25, pp. 1223–1231, 2012.
[5] Y. Yue, et al. "Using Supercomputer to Speed up Neural Network Training," in Proc. 2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), pp. 942–947, 2016.
[6] G. Suyog, et al. "Model Accuracy and Runtime Tradeoff in Distributed Deep Learning: A Systematic Study," in Proc. the 26th International Joint Conference on Artificial Intelligence, pp. 4854–4858, 2017.
[7] P. William. "Skip Lists: A Probabilistic Alternative to Balanced Trees." Communications of The ACM, vol. 33, no. 6, pp. 668–676, 1990.
[8] A. Mohammad, et al. “A Scalable, Commodity Data Center Network Architecture.” ACM SIGCOMM Computer Communication Review, vol. 38, no. 4, pp. 63–74, 2008.
1 Introduction

A large scale neural networks are applied for advanced image recognition and natural language processing [1, 2]. To output a high accuracy learning result from the neural network, it is necessary to apply the distributed learning [3]. Several mechanisms to refine a large scale neural network with many nodes distributively have been proposed, in particular, Data Parallelism is a famous distributed training concept. In Data Parallelism, there are two roles for the nodes. A master node maintains a neural network model and distributes it to all computation nodes. A computation node keeps different training data from each other and calculates the learning error (i.e., gradient data) of the distributed neural network model with the own training data. In completing the training, a computation node returns the result into the master node. The master node completes the aggregate of learning error and tunes its own neural network with the aggregated learning error. By repeating this procedure, the refinement of the neural network held on the master node is done. This basic distributed training architecture is adopted in Distbelief[4]. However, this scheme causes large data traffic to the master node. So, this scheme doesn’t have scalability. The scheme applying a tree structure to the connection relationship of the nodes is proposed to enhance scalability[5, 6]. This scheme prevents the learning error traffic from concentrating on a single master node. But this scheme encounters two problems: scalability for node insertion/deletion and redundancy of aggregation learning error path. At first, this scheme assumes the tree structure is built by human operation, therefore, the operations for node insertion/deletion are not flexible for node increase/decrease. Secondly, the tree structure has a weak point, that is, each node has only one connection for a parent node; hence, the data aggregation process is not stable for a parent node failure. Furthermore, since data aggregation is performed many times, the decrease in the training efficiency due to node failure is immeasurable.

In this study, we propose distributed skip mesh list (DSML), and apply it to a distributed training management scheme to address the abovementioned problems. DSML is based on distributed skip list (DSL)[7] and Fat-tree[8, 9]. DSL can build the tree structure automatically and be flexible for node increase/decrease, however, it has the abovementioned fault tolerance problem. Hence, the data aggregation process is not stable for a parent node failure. So, we applied mesh tree topology referring to Fat-Tree to Distributed Skip
List (DSL), and we defined it as Distributed Skip Mesh List (DSML). Fat-tree is accepted in current large scale data center, and it is known that Fat-tree achieves high fault tolerance for aggregating lots traffic compared with tree data center topology. Therefore, DSML’s aggregation stability is higher than that of the conventional tree topology scheme and enhances the learning efficiency.

In our previous study, we evaluate DSML with the fixed parameter so that it is not enough to verify its feature [10]. In this paper, we confirm the node insertion/deletion cost (i.e., the number of transmissions for dealing with it) and data aggregation efficiency by changing the key parameter of DSML.

2 Distributed deep learning training on DSML

In this section, we describe how to construct DSML and its characteristics at first. Next, we explain the distributed training management on DSML.

2.1 Architecture of DSML

In DSML, each node has two IDs: 1) a HashID and 2) a level that represents a hierarchy on DSML. The HashID is determined by the node network location or the hashed IP address. The nodes in DSML are lined up in the order of the HashID. Level $l (0 \leq l \leq L - 1)$ is calculated as follows:

$$-\log_K RAND$$

(1)

where the natural number $K (1 < K)$ is the value for setting the $K$-ary tree of DSML and $RAND (\frac{1}{K^L} < RAND < 1)$ is the uniform random number. The node belonging to the highest level $L$ is only the master node. The node belonging to level $l$ also belongs to the levels below it (e.g., a node in level $l$ belongs to levels $l - 1, l - 2, ..0$), noting that all nodes belong to level $0$.

In DSML, a node has several connections for the other nodes, such as the neighbor, parent, and children nodes. We focus herein on the node with both HashID $i$ and level $l$ on DSML. The node has connections for the closest right-/left-side neighbor nodes in each level. It also has a connection for the parent nodes, which are right-/left-side closest $K$ node locating level $l + 1$. Furthermore, the node has connections for children nodes, which are right-/left-side closest $K^2$ nodes locating level $l - 1$. A node takes a keep-alive messaging (i.e., heartbeat) for the neighbor, parent, and children nodes. When a node recognizes no reply for a long time, it replaces the corresponding node’s connections with the other closest nodes\(^1\). We describe DSML($K=2$) abstraction in Figure 1 based on the abovementioned structure. For example, node 2 (HashID: 2) is locating level: 1 on DSML and has connections for the closest right-/left-side neighbor nodes: 4, 0, 1, 3 in each level: 1, 0. In addition, node 2 has connections for parent nodes: 4, 0, and children nodes: 1, 2, 3, 4. Notes that DSML admits to keep the connections for the

---

\(^1\)In evaluation, a node recognizes a node failure when three “keep-alive messaging” continuously fail.
same node because the total number of node connections bound for \(O(\log N)\) where \(N\) is the number of nodes, and its connection preserving cost are not needed at all. When a node searches for a certain HashID, it looks for the connection of the node with the closest HashID and sends a search request for the node. Through this process that is recursively executed on DSML, the request is forwarded only \(O(\log N)\) until it arrives at the destination. A node insertion/deletion process can be briefly realized through this node search process. For the node insertion, the inserted node assigns the process to a node on DSML. The assigned node searches for the closest node of the inserted node and entrusts the insertion process to the closest node. Finally, the closest node updates the children, neighbor, and parent node connections for the insertion. It also takes \(O(\log N)\) forwarding an insertion request per node insertion. In handling a node deletion, each node uses a heartbeat and appropriately replaces the node connection that disappeared.

![Abstraction of DSML](image)

**Fig. 1.** Abstraction of DSML

### 2.2 Applying DSML for distributed training

We describe herein step by step the distributed deep learning training on DSML\((K=2)\) with Figure 1. The highest level node is a master node whose rules involve the learning error aggregation from all nodes, updating the held neural network model with the aggregated learning errors, and distributing it for all nodes. The master node (i.e., node 0 in Figure 1) distributes the neural network model for all nodes. In detail, this master node sends the model for its own \(K\) children nodes. Subsequently, the children nodes also do this recursively. Receiving the model, each bottom node calculates the learning error of it with the own training data and returns the learning error for its own \(K\) parent nodes. Note that all nodes belong to level 0, all nodes calculate the learning error. In the Figure 1, node 5 calculates the learning error of it with the own training data and returns the learning error for its own \(K\) parent nodes: 4, 6. This duplicated learning error sending for multiple parent nodes is the strength of DSML. Receiving the learning error from the children nodes, the parent nodes return it for their own parents. For recursive returns, the learning errors are aggregated into the master node.
The master node then tunes its own model with the received learning error and redistributes it with this process. The master node's model is refined through iterative model distribution and learning error aggregation.

3 Evaluation

This section evaluates the communication cost for the node insertion/deletion, the success probability of aggregating each learning error, and the average elapsed time for aggregating each learning error to a master node. Notes that all evaluations are done on our original simulator written by Python language.

![Fig. 2. Evaluation results. (top-left) The average number of massage transmissions for node search, (top-right) The success probability of aggregating learning error, (bottom) The average elapsed time for aggregating learning error](image)

3.1 Communication cost for the node insertion/deletion,

We define the communication cost as the number of messages among nodes in a node search process because the processes of node insertion/deletion use node search function. In this evaluation, we constructed DSML by changing the number of nodes from 10, 100, 1,000, 10,000, to 100,000 and the parameter $K$ from 2, 5, to 10. We selected two nodes from DSML, and we then measured the number of message transmissions between them 1,000 times. The top-left in Figure 2 shows the node search result indicating that the av-
verage number of message transmissions does not significantly increase, even if the number of nodes increases, because DSML takes only $O(\log N)$ messages for the node search. From this experimental result, we conclude that the node insertion/deletion can be done at a low cost even when the number of nodes is huge.

### 3.2 The success probability of aggregating each learning error

We now evaluate the success rate of aggregation learning error in changing the node failure ratio. If this evaluation is better, our scheme has high stability for the distributed deep learning training. We compared the performances by building both DSML and the tree topology by using DSL [5, 6, 7]. In this experimental scenario, we built both DSML and DSL overlays with 10,000 nodes and caused the node failure depending on the node failure ratio in each overlay. We then randomly selected 1,000 nodes, and the selected each node sent its own learning error to the master node through the parent nodes. We executed these operations 100 times in each overlay and investigated the average aggregation success probability of DSL and DSML (the top-right in Figure 2). The result shows that DSML could improve the aggregation success rate against DSL because it takes multiple parent connections for each node.

### 3.3 The average elapsed time for aggregating each learning error

Finally, we measured the elapsed time for aggregating the learning error in DSL and DSML until a learning error arrives at a master node from the bottom node 100 times. In DSML, the same learning errors are sent to a master node through the $K$ parent nodes. We then accept the elapsed time of the fastest learning error that arrived. In both DSL and DSML, the extra aggregation delay occurs because of node failure. When a node transmits a learning error to a failed parent node, the node retransmits the learning error until the retransmit succeeds. In concrete terms, the node transmitting learning error to the failed node fixes it with a node deletion process and retransmits the learning error. A bandwidth limitation exists among nodes; hence, each node divides the learning error into several parts and transmits them multiple times. We measured the elapsed time in DSL and DSML by setting the node failure probability to 0.01% in each transmission. Then, DSL and DSML adopted $K = 2$ and $K = 2, 5$ respectively, and the level of master node is constant in DSL and DSML. The number of learning error division (i.e., duplicated learning error sending for parent nodes) changes from 10, 1,000, 5,000, to 10,000. In addition, we assumed each transmission is completed at one second. The bottom in Figure 2 shows the result indicating that as the parent nodes are more in DSML, its aggregation stability is higher, and it can improve the elapsed time for aggregating learning error about more than 30%.
4 Conclusion and Future Perspective

In this study, we applied DSML to manage large scale neural networks. As a result, DSML realizes stable neural network management with a low communication cost. Our evaluation confirmed that DSML did not only take a low communication cost to construct its distributed structure but also permanently aggregated learning errors. In the future, we aim to improve the transmission success rate of the learning results by adjusting the execution interval of the heartbeat.

Acknowledgments

This work was supported by JSPS KAKENHI Grant Number JP20K19791.