International Journal of Parallel Programming
https://doi.org/10.1007/s10766-018-00623-w

Improving the Performance of Distributed MXNet with RDMA

Mingfan Li[1] · Ke Wen[1] · Han Lin[1] · Xu Jin[1] · Zheng Wu[1] · Hong An[1] · Mengxian Chi[1]

Received: 18 September 2018 / Accepted: 18 December 2018

Abstract
As one of the most influential deep learning frameworks, MXNet has achieved excellent performance and many breakthroughs in academic and industrial fields for various machine learning situations. The initial implementation of MXNet uses proxy-socket interface, which delivers suboptimal performance in distributed environment. In a massive parallel training task, parameters are updated frequently during each training loop, in which case network performance becomes the main factor of overall performance. Over the past decade, high performance interconnects have employed remote direct memory access (RDMA) technology to provide excellent performance for numerous scientific domains. In this paper, we describe an efficient design that extends the open-source MXNet to make it RDMA capable via RDMA-based parameter server interfaces. With modest optimizations towards memory usage and transmission overhead, RDMA-based MXNet achieves great performance improvement over the original software. Our experiments reveal that, for the communication subsystem of MXNet, the new design achieves 16x speedup (up to 21x at peak) over 1 Gigabit Ethernet (1GigE). For the two training cases on MXNet, the optimized implementation gains 5x and 9x speedup, respectively. Compared to experiments on the IP-over-InfiniBand (IPoIB) protocol, it achieves nearly 30% performance improvement, as well as better scalability and alleviation of bottlenecks.

Keywords Distributed MXNet · Parameter server · RDMA · InfiniBand · Network optimization

1 Introduction

In recent years, artificial intelligence has outperformed many state-of-the-art approaches in conventional fields. Particularly, deep learning achieves great success

✉ Mingfan Li
mingfan@mail.ustc.edu.cn

[1] University of Science and Technology of China, Hefei 230026, Anhui, China

Published online: 01 January 2019

for *object detection*, *natural language processing* and *medical image analysis* [1–3]. Moreover, benefiting from sorts of specialized accelerators, such as Google TPUs and NVIDA Volta GPUs, deep learning has drawn wider attention from experts across areas.

Deep learning usually demands a large amount of training data and powerful computing resources for data analysis. For one thing, the scale and complexity of machine learning algorithms are still increasing for exploring better models. A recent study shows that a deep neural network can even contain over 1000 layers [4]. In addition, the training-data becomes more complex for wider application scenarios. One application scenario is the detection of lung nodules using 3D convolutional network models. Each input sample is a computed tomography (CT) image of human lung, whose size is approximately $500 * 500 * 500$. It is difficult for a contemporary accelerator to store a whole image and its intermediate computation results, regardless of training or inference [3].

Training deep network with large input on a single node has its limitation, thus distributed training is needed. Recent experiments where 90-epoch ImageNet training with 2048 knights landing (KNL) just need minutes show vast potential for distributed training [5]. With the increase of cluster scale and accelerators, communication has become the bottlenecks for distributed application. While the sockets interface provides a great degree of portability, the byte-stream model in MXNet entails additional processing overheads. High performance networks and software APIs, such as OpenFabrics, provide RDMA capability that fits well with the system [6].

In this paper, we make a thorough analysis of MXNet's communication system, especially for its multilevel framework in Fig. 1. On the high level, MXNet interacts with the parameter server (namely ps-lite) through a global key-value store. At the bottom level, ps-lite finishes data transfer by ZMQ library [7]. With the fundamental RDMA semantic model, we propose a naive design to transform TCP/IP based communication into RDMA implementation. After that, we make some skilled optimizations: for the memory view, we design a circular memory buffer to reuse the RDMA pinned memory; for the efficiency view, we minimize the transmission overhead in batch-based messaging style.

Early this year (Jan 29, 2018), *DMLC* releases the official RDMA enabling ps-lite. The open literature shows the ps-lite operations obtains 34% improvements. Their implementation is similar to our design, but realized with RDMA verbs. RDMA verbs are wrappers for the lower level VPI verbs, the latter of which are applied in our design [8]. Apart from the optimizations towards memory usage and transimission, little disclosure about experiments on MXNet are discussed from their documents.

Our experiments illustrate that, the RDMA-capable parameter server ultimately gains 16x (up to 21x at peak) speedup over the original software. The RDMA-based MXNet obtains 5x and 9x speedup over the tcp-based version on two datasets. The performance is even improved by 36% on communication subsystem over IP-over-InfiniBand (IPoIB) [9]. For the evaluation of MXNet, two cases finally achieve 25% and 31% speedup. Our optimizations achieve great improvement on performance, as well as better scalability and alleviation of bottlenecks.

The rest of this paper is organized as follows. Section 2 introduces background material of our work. Section 3 demonstrates our contributions to MXNet, including

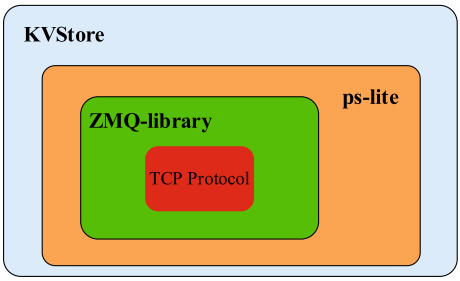

Fig. 1 Software stack of MXNet communication framework

our analysis of communication framework and naive RDMA-based implementation. Further optimizations are also discussed here. Section 4 describes the performance benefits from RDMA, where we conduct experiments for ps-lite and MXNet. Section 5 finally introduces our discussions and conclusions.

2 Background Review

2.1 MXNet Architecture

MXNet is a dataflow-based deep learning system, utilizing a dependency engine to perform resource management and task scheduling. Each resource unit is registered to the engine with a unique tag and all operations (e.g., matrix operation, data communication) are pushed into the engine by specifying the required resource tags. The system works in an asynchronous multithread pattern to achieve data- and model-parallelism for better resource utilization.

Analogous to traditional machine learning frameworks, MXNet divides model training into two roles: the worker performs local computation in active while the server passively maintains global parameters. For each mini-batch iteration, the workers first read the input data, then fetch the weight from the server. Then the gradient vector is computed and push to the servers. At the same time, the servers will aggregate the workers' submissions and update global weight.

2.2 Parameter Server Framework

For the purpose of solving distributed machine learning problems, parameter server framework supports flexible consistency models, elastic scalability and fault tolerance [10]. In general, the primary communication characteristics of MXNet come from the ps-lite. In ps-lite, nodes are divided into two kinds by their roles: clients are responsible for data and workload while all servers maintain global shared parameters. In MXNet, the corresponding representation of the parameters is neural network model. In addition, ps-lite prepares a channel for servers to execute user-defined functions, which is the raw interface for updating parameters of network models. Such characteristics ensure easier encapsulation of ps-lite into MXNet's dataflow-driven pattern.

Particularly for machine learning, ps-lite designs two types of communication mechanisms: *request/response* and *push/pull*. The former is used to exchange control-messages for few communication routines of scheduler. The latter pair is used to transfer data messages between nodes. It's notable that network traffic originates from frequent data communication, which is addressed by data transfer operations, *push* and *pull*. The *push* sends local modified data entries to remote (client submits gradients to server). In turn, the *pull* retrieves remote data entries as requested (client fetches weight from server).

2.3 RDMA and InfiniBand

InfiniBand is an open industry standard switched fabric designed to interconnect nodes in high performance computation (HPC) clusters [11]. It is a high-speed, general purpose I/O interconnect for worldwide computing centers. One of the main features of InfiniBand is remote direct memory access (RDMA). This feature allows allows full remote CPU bypass by letting one machine directly read or write the memory of another machine without involving the remote CPU, which facilitates implementation of advanced communication protocols.

InfiniBand Architecture provides two means of message transport: channel and memory semantics. In memory semantics, *RDMA write* and *RDMA read* operations are introduced. In channel semantics, *send/receive* operations are used for communication like the *TCP/IP* protocol. Particularly, RDMA operations are one-sided and incur no software overhead at the remote side, that enables true application bypass message passing. Using RDMA requires the destination address must be known beforehand. Furthermore, the detection of incoming message must be handled explicitly at the receiver side.

IB verbs is a low level communication interface for RDMA ability over InfiniBand. It contains necessary APIs for communication transactions. Programmers rely on these interface for creating, modifying and destroying resources, such as queue-pairs (QP), completion queues (CQ), memory regions (MR) and protection domains (PD). The sending and receiving of work requests to QPs, getting completions from CQs are also guaranteed.

3 Contributions

As discussed in MXNet's communication architecture, data synchronization is supported with two primitives: *push* and *pull*. These primitives are actually implemented by the parameter server framework. Thus, we advocate optimizing MXNet by extending ps-lite to RDMA-capable interface.

3.1 Communication Framework

The communication architecture of MXNet provides seamless work between multi-device on a single machine (i.e., GPUs and CPUs), as well as multi-machine for

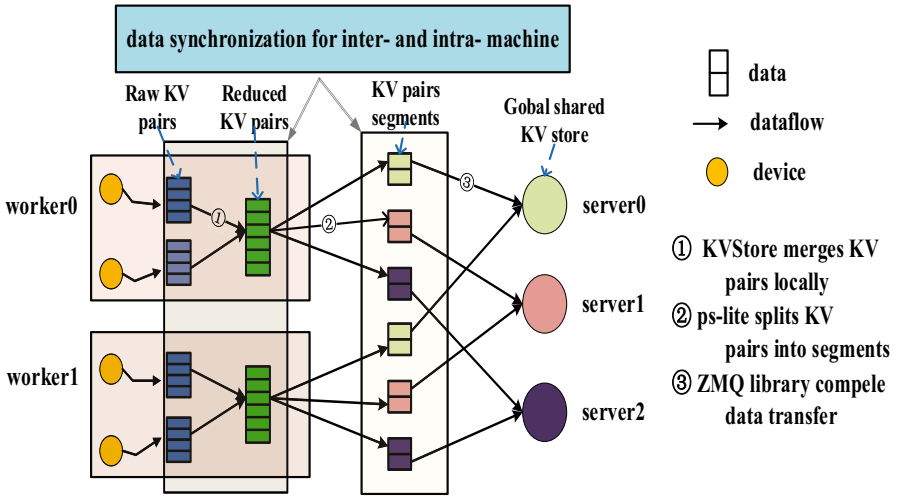

Fig. 2 Details for communication architecture and dataflow in MXNet

an entire cluster. Smoothly combined with KVStore, ps-lite and ZMQ, the blend architecture achieves balance from aspects of execution model, program logic and high performance. It implements efficient and reliable data transfer, especially for heterogeneous environments.

At first, the workload is scattered into several compute devices within a node. In turn, we will receive a number of the intermediate values from different devices. For example, each device performs iterations and produces independent key-value (KV) pairs parallelly. Instead of pushing these raw KV pairs, the KVStore merges intermediate values into local aggregated results by identical keys. After that, the ps-lite splits the reduced KV pairs into multiple fragments, which will be submitted into target servers finally. The partition policy and destination for segments are predefined by the servers' number and global key range. Figure 2 presents basic dataflow among distributed MXNet system, containing 2 workers equipped with total 4 compute devices and 3 servers.

Data synchronization is divided into two-levels, inter- and intra- machine. After the reduction and partition, the outbound segments are ultimately transmitted by ZMQ interfaces. The aggregation strategy avoids extra data movement and minimizes network traffic for sending data, which dramatically decreases the bandwidth requirement. It's notable that the communication subsystem is transparent to computation. In other words, our optimizations for communication interfaces are independent on whether CPU or GPU is used on computing.

One acceptable description of the communication system is to take it as joint framework from three close layers: KVStore takes a role as distributed database for parameters stored on MXNet; ps-lite is then responsible for conducting communication logic among distributed deep learning; the bottom ZMQ will server as high performance media for data transfer tool.

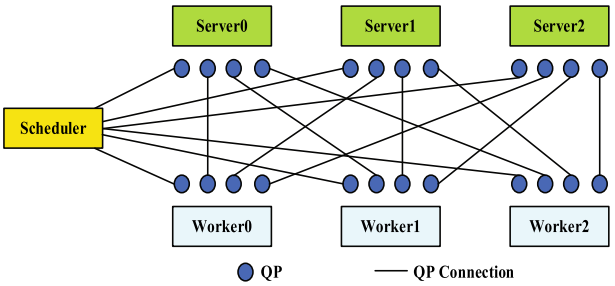

Fig. 3 A basic model for parameter server porting RDMA

3.2 Basic Model

The software architecture and communication pattern are crucial for RDMA optimizing. As for MXNet, the three-level constructure of subsystem plays different role for distributed training. Consider that the ps-lite is specially designed for MXNet and distributed machine learning. We consequently propose a typical design to implement the RDMA-capable MXNet by replacing the ZMQ interfaces with RDMA operations. In this case, MXNet's communication architecture is smoothly equipped with RDMA technology.

There are three roles in ps-lite: scheduler, workers, and servers. Figure 3 shows a prototype model that consists of 1 scheduler, 3 workers and 3 servers. The necessary connections between these QPs are also marked out.

(a) the scheduler conduct management with control message among all roles
(b) the worker will only communicates with other roles (control message with scheduler and data message with multi servers)
(c) similar to the woker, the server communicates with non-server roles

With these established QP connections, it's possible to transfer message with RDMA among cluster. This model is similar to connection orinted TCP, which means reliable and stable communication service. Self communication is identical to communication between distinct nodes. The design originates from the *subscribe/publish* model for ZMQ solution, which treats the communication process for both inter- and intra- nodes based on gernal format in *host:port*.

We implement basic design with replacing the ZMQ APIs (i.e., *zmq_msg_send/zmq_msg_recv*) by RDMA interfaces: *ibv_post_send/ibv_post_recv*. We override the communication interfaces, and redesign the communication for RDMA, which controls the data transfer of ps-lite. The fundamental workflow is depicted in Fig. 4. The initialization is still based on the raw ZMQ transport, as well as RDMA connection preparation. After that, message is migrated into InfiniBand by substituting ZMQ functions with the RDMA verbs. Our implementation selects the memory semantic RDMA operation according to an empirical conclusion that the memory semantic is generally faster than message semantic.

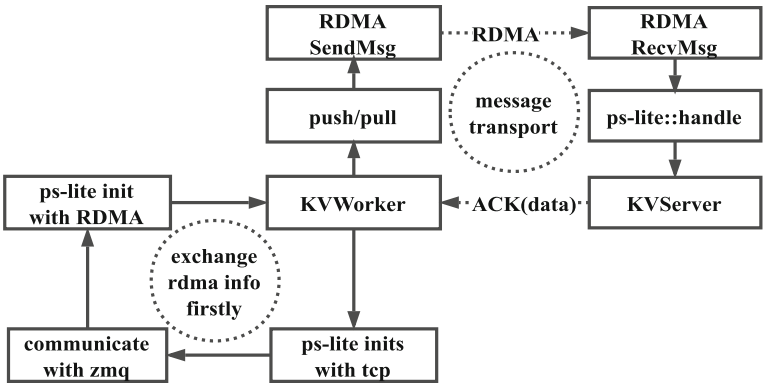

Fig. 4 The implementation of RDMA-based ps-lite

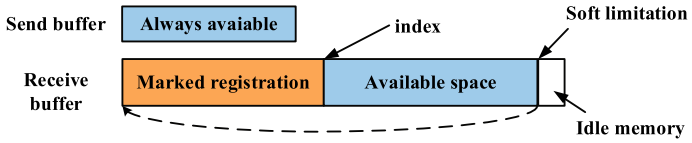

Fig. 5 Memory usage: circular receiving buffer and direct sending buffer

3.3 Memory Reuse

RDMA operations need pinned memory as data buffer for each QP connection. One buffer is prepared for sending data, the other is for receiving data. These buffers are accessible memory for IB hardware to direct data writing or reading without the involvement of host CPU. This hardware DMA ability of IB equipment that is connected with remote devices is the true reason for RDMA to be efficient. The pinned memory for RDMA buffers is persistent for the physical device it mapped to, i.e. it cannot be swapped out by OS. This is another reason for ultra-low latency and high speed of RDMA.

For a ps-lite system including n servers and m workers, if buffer size is k, we will need $m \times n \times 2 \times k$ buffer space at total (each QP connection requires 4 buffers). The space complexity for RDMA communication will be $O(n^2)$ for the system scale. However, abusing the pinned memory may incur the cripple of OS performance for the decrement of manageable memory.

In order to improve the utilization of pinned memory, we implement self-management for the memory buffer. Consider that the traditional strategy will *allocate/free* physical memory and record every transaction in a list as reference for later operation. These transactions slice contiguous space into many discrete smaller pieces, which violates the RDMA requirement for contiguous memory. Therefore, we choose another intuitional mechanism. As shown in Fig. 5, we build the large memory as a circular buffer and record a buffer flag, *index*.

To prevent overflow, we set a soft limitation as maximum flag. As soon as the index is over the flag, we will clear the index simultaneously and arrange the next request from scratch. Avoiding slicing the memory by allocation and free operation, we just mark

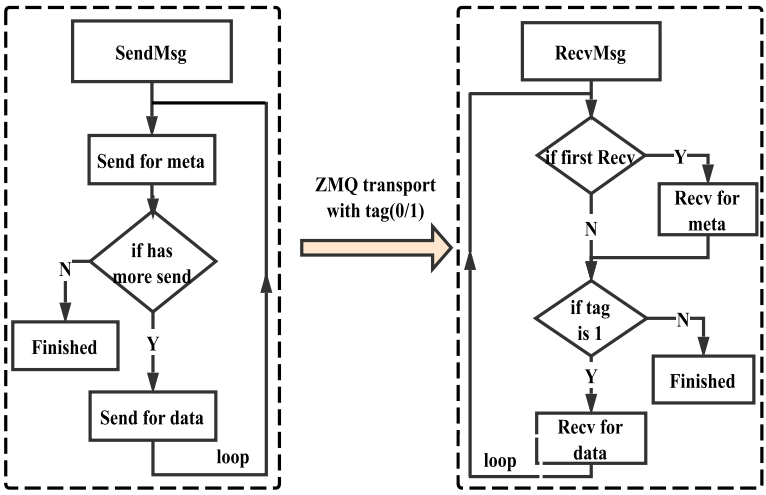

Fig. 6 ZMQ-based transfer procedures for ps-lite

the limitation flag and select next available memory address. Furthermore, we design a continuous proxy daemon for data preparation by the thread-safe queue for managing the buffer operation, which helps reduce the overhead from memory management.

3.4 Batch-based Transfer

The original ps-lite uses a number of shared points to avoid data movement. There is no explicit data movement for communication process, beginning at the KVStore invocations and ending at the ZMQ APIs. Such architecture further improves the performance and decreases the latency. Meanwhile, the discrete data locations for a whole message require that ps-lite delivers one message with several transfer operations.

The complete procedures are presented in Fig. 6, which shows the detail for messaging between objects. The meta and several data are one-by-one delivered by sequential ZMQ transport operation, when a tag is marked for receiver to judge the end of current message and to prepare the next incoming message. After the receiver gets all subparts of message, it assembles the subparts into complete message as claimed by the first meta description.

Naive solution for porting the ps-lite into RDMA is just to replace the ZMQ transport with RDMA APIs for the control flow, where the tag can just be represented by the immediate data in RDMA Write. Consider that our previous introduction for RDMA memory design, the outbound message lies in a large contiguous pinned memory for RDMA operation.

We can integrate multiple data block into single message, which removes the process for slicing and reassembling the messages for transfer transactions. The sender wraps the meta and all data blocks into a special block, and the encapsulated message can then be extracted by the receiver. Besides the row meta field and multiple data fields for each message, we add several marks for distinguishing the data field, as in Table 1.

Table 1 RDMA-based message format

Size/bit	Description
`8`	Data block number (N)
`N * 8`	Size of each block, from block 1 to N
`Raw meta size`	Original metadata
`Total data size`	Original data block

The combination field of one *data_block_num* flag and multiple *data_block_size* flags is newly meta-data for assembling the received message.

The RDMA communication is an immediately, blocking mechanism, which means that two objects with connected QP pair can't execute the next data transfer request until the completion of current work request. For each transfer, we have to performe a few routines, such as pre-processing for transfer, request submitting, waiting and polling result from completion queue, and post-processing. Particularly, the polling process can't be returned until the completion of each transfer. After organizing the data into RDMA-based message format, the batch-based transimission reduces the overhead for original data transfer.

4 Performance Evaluation

We conduct our evaluation from two aspects, the pure communication efficiency in ps-lite and the whole distributed training process in MXNet. According to the analysis in Sect. 3, the performance evaluation towards the communication has nothing to do with the compute device. Our platforms for evaluation are:

(a) Cluster A: 3 Sugon nodes with dual Intel Xeon E5-2660 processors. Each node has 128GB of main memory. The interconnect hardware consisits of 1000M Ethernet and Mellanox 40G QDR InfiniBand.
(b) Cluster B: 9 Sugon nodes and each node are the same to Cluster A configurations.

We measure the pure communication efficiency in Cluster A, where 3 nodes respectively play a role in system, like scheduler, server, worker. The whole distributed training test are among Cluster B, where 8 nodes are servers and workers, within additional scheduler node. We perform the specified test cases on different software and hardware environments (Un-optimized and RDMA-optimized software, Ethernet and InfiniBand). With different elapsed time for the same test case, we can judge the performance improvement from the optimizations.

4.1 Evaluation for Ps-lite

The ps-lite provides two fundamental services for distributed system, *request/response* model for control message and *push/pull* model for data transfer. The KVStore implementation from the key-value pair just suits the test case that sends 100,000-length key-value pairs for 10,000 times. For distributed system, we increase number of the worker and server, from 1 up to 32 to evaluate the scalability. We measure the total

Fig. 7 Performance for purely communication in ps-lite

execution time of the same tasks on 3 different versions (TCP, IPoIB, RDMA), and we define the speedup ratio of Y over X as following equation:

$$Speedup(Y, X) = \frac{T(X)}{T(Y)} \tag{1}$$

where $x \in \{TCP, IPoIB, RDMA\}$ and $T(x)$ means the elapsed time.

Figure 7 shows the result of experiments on ps-lite. Compared with the traditional Ethernet, the performance gets over 16x speedup, up to 21x at peak. Removing the hardware advantage, our software obtains 36% improvements, compared to the IPoIB protocol.

Obviously, the InfiniBand hardware contributes a lot to the improvements of both IPoIB protocol and RDMA. But the pure software optimizations also gains splendid outcome especially for heavy traffic.

(1) With little workers/servers (less than 8), the system performance is not much influenced by the slight traffic. In this case, the performance improvement depends larger degree on the hardware, the InfiniBand over Ethernet. Compared with original TCP-based software, we achieve the 21x speedup for RDMA, with 19x speedup for IPoIB. The comparison between RDMA and IPoIB shows that only 11% improvements. We blame it on the weak CPU overhead for communication.
(2) While the system scale increases, the overhead from communication incurs CPU resource damage for heavy traffic. The general speedup by tcp version decreases to 16x for our RDMA implementation, while the corresponding IPoIB implementation is severely influenced, from 19x to 12x speedup. The relative comparison between RDMA and IPoIB shows 36% improvements. Our RDMA-based ps-lite shows better alleviation of bottlenecks for heavier traffic.

Generally speaking, RDMA releases the CPU resources that is greatly consumed by tedius message processing from TCP/IP protocol stack. With heavier traffic, RDMA will be more effective, which meets our evaluation for the better scalability of RDMA-based ps-lite over the IPoIB version.

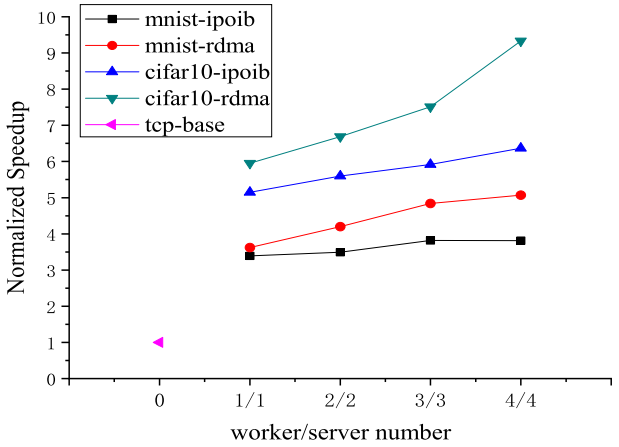

Fig. 8 Performance difference for testcases in MXNet

4.2 Evaluation for MXNet

We perform model training with MXNet for several datasets on image processing field, the MNIST for handwritten recognition and the cifar-10 for image classification. To represent the heavy network communication than computation, we select the fundamental MLP network (it just contains two fully connected network layers, and requires a lot communication for weights updating). To emulate high strength communication among the large-scale clusters, we make further increase for both two hidden neural number up to 2048.

In Fig. 8, we show the performance improvement from RDMA optimizations, evaluated on the MXNet. For the MNIST test case with the tcp protocol, the speedup is pretty clear, the 3.13x speedup on just 2 nodes up to 4.62x on the whole 8 nodes. It also shows the better result that IPoIB protocol on both performance and scalability, from 6% improvement on 2 nodes to 25% on 8 nodes.

Results from cifar-10 experiments obtain similar and better outcome than the MNIST. The performance speedup over tcp protocol starts begin by 5x at 2 nodes and comes up to 9x for the entire cluster, and the corresponding improvement over IPoIB version varies from 14 to 31%. We obtain clearly better efficiency and scalability, and the optimized software achieves the linear promotion with the cluster node.

The evaluation of both test cases, MNIST and cifar-10, proves that our work gains great success on the distributed training process on MXNet. However, there are obvious differences between those two test cases. The cifar-10 obtains better performance improvement, the 5x speedup versus 9x over tcp, corresponding 25% improvement versus 31% over IPoIB. We blame the difference on the different account of communication workload for the whole training process. This attributes to the larger network parameters along with cifar-10 test case than mnist. The three channel RGB input in cifar-10 is 3 times over the single channel grayscale in mnist. Therefore, it receives better outcome from RDMA than the non-optimized software.

5 Discussion and Conclusion

5.1 Related Works

Architecture design Complementing software with RDMA contains two different designs. In a direct approach, the original communication framework is removed, and new-designed module takes its role for porting network traffic into InfiniBand. An alternatively method simply replaces the partial communication interfaces, and it conserves the original software feature as much as possible. In order to fulfil the efficient optimizations, some researches, like RDMA-based HDFS or Memcached [12–14], utilize the first solution to redesign their own unified communication runtime (UCR) [15]. However, similar work like recent RDMA-capable Tensorflow or Hadoop RPC over InfiniBand prefer the second approach, they contribute adaptation for original software with extra RDMA bypass for tensor messages [16,17]. RDMA enabling Tensorflow realizes the high performance data transfer on RDMA but also relies on the original grpc for "administrative" tasks. From the perspective of MXNet, the three-level subsystem collectively makes up the communication system. We adopt the second solution which replaces the ZMQ library with RDMA operation.

RDMA buffer technique As for the registration of pinned buffer, one sophisticated approach is to register a large contiguous memory in advance and manage it for later memory request. To prepare the registered memory, typical design is to apply the *REQ/ACK* message to build automatic state machine for monitoring memory usage. However, we use a simple index and thread-safe queue for circular buffer maintenance. Instead of recording a transaction list, we modify the index flag and maintain the execution queue. Past studies about RDMA have shown good performance in achieving computation/communication overlap with self-management, such as RDMA-capable Tensorflow or RDMA Key-Value Store [16,18]. We learn from their work and present our proposal.

Distributed machine learning A distributed system should meet numerous system level requirements, including consistency, fault tolerance, communication and resource management. Relative researches about MXNet communication mainly concentrate on software infrastructures. Zhang's Poseidon presents a hybrid communication model for different neural network layer, based on the distribution of parameters among different layers, especially for GPU environments [19]. Latest work about embedding MPI parallelism for parameter server model, MXNet-MPI, achieves excellent performance [20]. Different from their work, our work pursuits performance improvement by optimizing MXNet with RDMA ability. Some RDMA-based researches aim at crafting a middleware for distributed system, such as the MPICH2 over InfiniBand or RDMA engine for TCP/IP processor [21,22]. And Kalia's research for building efficiently key-value sevices also inspires us on KVStore module [23].

5.2 Conclusion

This paper presents a RDMA-capable design of MXNet by implementing a low-level communication subsystem, RDMA-based ps-lite, by replacing ZMQ with RDMA operations. Our experiments demonstrate great performance improvement on both communication and overall training for MXNet.

We analyze the workflow of distributed MXNet and focus on multi-layer distributed KVStore. After a naive RDMA design of replacing the ZMQ transmission with the RDMA verbs, we then make several optimizations on memory management and transmission overhead. In our final evaluation, the RDMA optimized ps-lite achieves over 16x speedup, and more than 21x speedup at peak for purely communication workload. The MXNet equipped with RDMA-based ps-lite shows 5x and 9x improvement on different test cases. It also shows pretty scalability with increasing of nodes and training workload from our evaluation. Apart from the hardware advantage, the optimizations for software obtains 30% performance improvement Compared with IPoIB.

In conclusion, our RDMA-based MXNet provides a possible approach to equip complex frameworks the RDMA ability by redesigning intermediate interfaces. Two techniques introduced in the paper: self-management circular buffer and batch-based operation, are common but typical for architecture design and software optimizations.

Acknowledgements The work is supported by the National Key Research and Development Program of China(Grants No. 2016YFB1000403).

References

1. de Bruijne, M.: Machine learning approaches in medical image analysis: from detection to diagnosis. Med. Image. Anal. **33**, 94–97 (2016). https://doi.org/10.1016/j.media.2016.06.032
2. Young, T., Hazarika, D., Poria, S., Cambria, E.: Recent trends in deep learning based natural language processing. IEEE Comput. Intell. Mag. **13**(3), 55–75 (2018). https://doi.org/10.1109/MCI.2018.2840738
3. Pérez, G., Arbeláez, P.: Automated detection of lung nodules with three-dimensional convolutional neural networks. Proc. SPIE 10572, 10572-1-10572-10 (2017). https://doi.org/10.1117/12.2285954
4. Huang G., Sun, Y., Liu, Z., Sedra, D.,Weinberger, K.Q.: Deep networks with stochastic depth. In: European Conference on Computer Vision, pp. 646–661. Springer (2016)
5. You, Y., Zhang, Z., Hsieh, C., Demmel, J., Keutzer, K.: ImageNet training in minutes. CoRR. arXiv:1709.05011 (2017)
6. Grun, P., Hefty, S., Sur, S., Goodell, D., Russell, R.D., Pritchard, H., Squyres, J.M.: A brief introduction to the OpenFabrics interfaces: a new network API for maximizing high performance application efficiency. In: 2015 IEEE 23rd Annual Symposium on High-Performance Interconnects, pp. 34–39 (2015). https://doi.org/10.1109/HOTI.2015.19
7. Hintjens, P.: ZeroMQ: the guide. http://zguide.zeromq.org/page:all (2010)
8. MacArthur, P., Liu, Q., Russell, R.D., Mizero, F., Veeraraghavan, M., Dennis, J.M.: An integrated tutorial on InfiniBand, verbs, and MPI. IEEE Commun. Surv. Tutorials **19**(4), 2894–2926 (2017). https://doi.org/10.1109/COMST.2017.2746083
9. RDMA Consortium and others: Architectural specifications for RDMA over TCP/IP (2009)
10. Li, M., Zhou, L.,Yang, Z., Li, A., Xia, F., Andersen, D.G., Smola, A.: Parameter server for distributed machine learning. In: Big Learning NIPS Workshop, vol. 6, p. 2 (2013)
11. Buyya, R., Cortes, T., Jin, H.: An introduction to the InfiniBand architecture. In: High Performance Mass Storage and Parallel I/O: Technologies and Applications (2002). https://doi.org/10.1109/9780470544839.ch42

12. Liu, J., Wu, J., Panda, D.K.: High performance RDMA-based MPI implementation over InfiniBand. Int. J. Parallel Program. **32**(3), 167–198 (2004). https://doi.org/10.1023/B:IJPP.0000029272.69895.c1
13. Islam, N.S., Rahman, M.W., Jose, J., Rajachandrasekar, R., Wang, H., Subramoni, H., Murthy, C., Panda, D.K.: High performance RDMA-based design of HDFS over InfiniBand. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, p. 35. IEEE Computer Society Press (2012)
14. Jose, J., Subramoni, H., Luo, M., Zhang, M., Huang, J., Wasi-ur Rahman, M., Islam, N.S., Ouyang, X., Wang, H., Sur, S., et al.: Memcached design on high performance rdma capable interconnects. In: 2011 International Conference on Parallel Processing (ICPP), pp. 743–752. IEEE (2011)
15. Jose, J., Luo, M., Sur, S., Panda, D.K.: Unifying UPC and MPI runtimes: experience with MVAPICH. In: Proceedings of the Fourth Conference on Partitioned Global Address Space Programming Model, p. 5. ACM (2010)
16. Jia, C., Liu, J., Jin, X., Lin, H., An, 412 H., Han, W., Wu, Z., Chi, M.: Improving the performance of distributed TensorFlow with RDMA. Int. J. Parallel Program. **46**(4), 674–685 (2018). https://doi.org/10.1007/s10766-017-0520-3
17. Lu, X., Islam, NS.,Wasi-Ur-Rahman, M., Jose, J., Subramoni, H.,Wang, H., Panda, D.K.: High-performance design of Hadoop RPC with RDMA over InfiniBand. In: 2013 42nd International Conference on Parallel Processing (ICPP), pp 641–650. IEEE (2013)
18. Mitchell, C., Geng, Y., Li, J.: Using one-sided RDMA reads to build a fast, CPU-efficient key-value store. In: USENIX Annual Technical Conference, pp. 103–114 (2013)
19. Zhang, H., Zheng, Z., Xu, S., Dai, W., Ho, Q., Liang, X., Hu, Z., Wei, J., Xie, P., Xing, E.P.: Poseidon: an efficient communication architecture for distributed deep learning on GPU clusters. arXiv preprint arXiv:1706.03292 (2017)
20. Mamidala, A.R., Kollias, G., Ward, C., Artico, F.: MXNET-MPI: embedding MPI parallelism in parameter server task model for scaling deep learning. ArXiv e-prints arXiv:1801.03855. http://adsabs.harvard.edu/abs/2018arXiv180103855M (2018)
21. Liu, J., Jiang,W.,Wyckoff, P., Panda, D.K., Ashton, D., Buntinas, D., Gropp,W., Toonen, B.: In: 18th International Parallel and Distributed Processing Symposium, 2004 (IEEE, 2004), p. 16
22. Pandya, A.A.: TCP/IP processor and engine using RDMA (2008). US Patent 7,376,755
23. Kalia, A., Kaminsky, M., Andersen, D.G.: Using RDMA efficiently for key-value services. ACM SIGCOMM Comput. Commun. Rev. **44**(4), 295–306 (2015)