All You Need to Know about Scheduling Deep Learning Jobs

Wencong Xiao
Beihang University and Microsoft Research

1. Introduction

With the recent breakthrough in deep neural network, there is an emerging class of data center with accelerated hardware to support efficient training on neural network model [1, 2]. The accelerated hardware (e.g. GPU, FPGA, TPU [9], Cambricon [11]), interconnected with high speed network (e.g. infiniband) and coupled with large training data [7], provides orders of magnitude training speedup.

In this paper, we study the new challenge of resource management derived from the characteristic of deep learning workload in a cluster with accelerated hardware.

The first challenge is to find an extensible resource abstraction to represent the diversified and fast evolving accelerated devices. Deep learning job should be able to learn the resource type and its usage, and be able to request for a certain type of devices with specific topology requirement. In Section 2, we introduce detailed hardware configuration in a typical data center for deep learning and propose a resource abstraction to address this challenge.

The second challenge results from a tension in deep learning job scheduling. For multiple deep learning jobs, we find the system should "spread" them away to avoid mutual interference. While for a large deep learning job that requires multiple accelerated devices, the system should "pack" it to the devices that are close to each other to avoid significant loss of training speed. The spreading will lead to the fragmented usage of the accelerated devices, while the packing would require the consecutive slots in the devices. In Section 3, we quantify the effects of job interference and demonstrate the significant performance difference for a large job with different locality setting. We then discuss some possible way to resolve the tension introduced by job spreading and packing.

2. Heterogeneity in a deep learning cluster

Typically, a deep learning cluster contains multiple infiniband domains, each consists of multiple racks. Different rack could install with different accelerated devices, such as different generation of GPU, FPGA, TPU and ASIC [9, 11]. The accelerated devices may have a PCIe interface, they may further interconnect to each other with vendor specific link technique, such as NVLINK [5]. The device connects to the CPU directly or through PCIe switch. Figure 2a shows a hardware configuration of a server in a GPU cluster. The server contains two CPUs, each connects to two PCIe switches hosting two GPUs.

The resource management system should capture the resource usage of diversified hardware and allow deep learning jobs to request for a certain type of hardware with a certain topology requirement. To this end, we design a compact resource abstraction. A bitmap is introduced to represent the availability of accelerated devices on each server. A scheduler can use the bitmap to learn the runtime resource usage and express resource request. We further use a configuration set to denote a set of homogeneous servers (e.g., servers within a rack). Each configuration set includes detailed meta information to describe the device type (e.g., GPU), device topology within a server (e.g., a 8-bit bitmap denotes two CPUs, each with 2 PCI-e switches hosting 2 GPUs where they interconnect with NVLINK), and network topology (within an IB domain). The meta data for a configuration set is highly extensible and seldom changes. And there are not many different configuration sets in a cluster. The size of total meta data is not large and can be kept in a read-only memory block when making scheduling decisions.

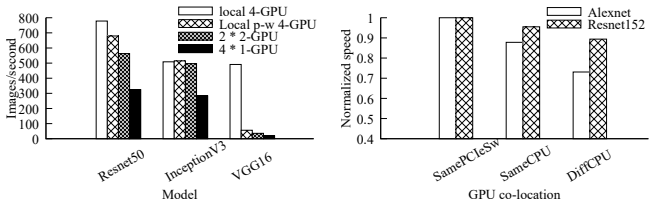
3. The characteristics of deep learning job

A deep learning job often lasts for hours and some even lasts for weeks. The performance is sensitive to locality: a small percentage of performance changes could result in hours of training time variance. Moreover, deep learning workload usually requires gang-scheduling, the training process cannot start until all required accelerated devices are granted. In this section, we use GPU as an example of accelerated device to show the characteristics of deep learning job.

3.1 Inner job performance

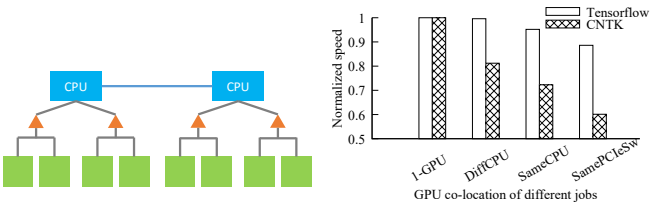
We have found that for a job runs on multiple GPUs, packing GPUs as close as possible could achieve significantly better performance.

Figure 1a quantifies the performance on different topology, using three CNN models [8, 12, 13] in the Tensorflow [6] benchmark on NVIDIA P100 GPU machines. "Local 4-GPU" is the result when Tensorflow runs on a single server using 4 GPUs. "Local p-w 4-GPU" shows the result



(a) Inter-server locality. (b) Intra-server locality.

Figure 1: Mutli-GPU performance with locality setting.



(a) Server architecture. (b) Effect of job interference.

Figure 2: Server architecture and job interference.

in the hardware setting while Tensorflow is configured to run distributedly in parameter server mode [10], with one server and one worker on the same server. "2x2-GPU" denotes the result on 2 servers each using 2 GPUs. And "4x1-GPU" shows the result on 4 1-GPU servers. As shown in the figure, spreading resources among different servers slows down the job. For the 4-GPU ResNet50 case, the performance reduces by 28% when breaking into 2 servers, and significantly drops by 59% when spreading to 4 servers. For VGG16, even configuring Tensorflow in local parameter-server mode would lead to 90% slowdown. Moreover, the GPU locality within the same server also affects the performance. For example, CNTK [3] benefits from GPU interconnecting technique like NCCL [4], which leverages GPUDirect to directly access other GPU memory, avoiding extra data copy and hence accelerating model training. Figure 1b shows the 2-GPU CNTK job performance on different GPU locality level for Alexnet and ResNet152. GPUs located under different CPUs will lead to 27% slowdown for Alexnet and 11% for ResNet152, comparing to the GPUs under the same PCIe switch. On the other hand, some models like InceptionV3 can tolerate a certain degree of resource spreading: the "2x2-GPU" case barely has any performance loss.

From Figure 1a and 1b we learn that, spreading resouce generally has negative impact to deep learning jobs. But even for deep learning jobs with the same hardware requirement (2 or 4 GPUs), different type of deep learning workload can tolerate different level of resource spreading. A scheduling framework should have a flexible contract with the deep learning workload. It not only should support scheduling request for a specific hardware and locality configuration, but also should allow the job to express the degree of tolerance in resource spreading.

3.2 Inter job interference

A deep learning job relies on GPU to accelerate computation, but it still requires frequent communication with CPU through the shared PCIe bus. Thus jobs run in the same server may interfere with each other. We observe noticeable job inference in the RNN benchmark on both Tensorflow and CNTK. In Figure 2b, we use 1-GPU job running solely in the server as the baseline and compare two 1-GPU jobs running on the same server with different GPU locality settings (on different CPUs, on the same CPU, and under the same PCI-e switch). The performance drops as the GPUs of the two jobs are placed closer. It shows up to 12% slowdown in Tensorflow and up to 40% slowdown in CNTK when under the same PCI-e switch.

From the experiments we can see the requirements for inter-job and intra-job scheduling inherently conflict with opposite preference. Inter-job scheduling tends to spread among different machines to avoid interference, therefore result in resource fragmentation. Contrarily, large jobs prefer devices to co-locate closely with stringent topology-aware locality constraint for better performance. Large jobs suffer more from the fragmentation. Based on our observation in a real deep learning cluster, such large jobs suffer more from such unfairness as about 80% jobs are 1-GPU jobs. However, large jobs are often more important, handling larger dataset and with larger model to achieve better accuracy.

4. Conclusion and future work

We design a new scheduling system for heterogeneous data center with accelerated hardware to speedup deep learning workloads. It adopts a flexible and compact resource abstraction to represent the evolving heterogeneous hardware with locality and topology awareness.

The scheduling system embraces a decentralized design to decouple cluster-wide policy from individual job scheduling. Each job scheduler leverages the resource abstraction view to learn cluster-wide resource usage and make scheduling decision with locality and topology preference based on its own hardware requirement and the specific characteristic of deep learn model. A centralized scheduling component maintains the lightweight cluster view, and quickly approves or rejects the gang-scheduling request from individual job scheduler based on resource status.

The scheduling system further adopts an adaptive mechanism. When the cluster is under light load, it tends to spread out jobs to avoid interference. While when the cluster is under heavy load, it will pack the jobs together to make place for multi-GPU jobs. A migration decision module will pack small jobs closer during runtime to make place for large jobs with strong locality constraint.

References

[1] AWS GPU. https://aws.amazon.com/ec2/

instance-types/p2/.

[2] Azure GPU. http://gpu.azure.com/.

[3] CNTK. http://www.cntk.ai/.

[4] NCCL. https://developer.nvidia.com/nccl/.

[5] NVIDIA NVLINK. http://www.nvidia.com/object/nvlink.html/.

[6] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[8] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[9] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, R. Boyle, P. luc Cantin, C. Chao, C. Clark, J. Coriell, M. Daley, M. Dau, J. Dean, B. Gelb, T. V. Ghaemmaghami, R. Gottipati, W. Gulland, R. Hagmann, C. R. Ho, D. Hogberg, J. Hu, R. Hundt, D. Hurt, J. Ibarz, A. Jaffey, A. Jaworski, A. Kaplan, H. Khaitan, A. Koch, N. Kumar, S. Lacy, J. Laudon, J. Law, D. Le, C. Leary, Z. Liu, K. Lucke, A. Lundin, G. MacKean, A. Maggiore, M. Mahony, K. Miller, R. Nagarajan, R. Narayanaswami, R. Ni, K. Nix, T. Norrie, M. Omernick, N. Penukonda, A. Phelps, and J. Ross. In-datacenter performance analysis of a tensor processing unit. 2017. URL https://arxiv.org/pdf/1704.04760.pdf.

[10] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su. Scaling distributed machine learning with the Parameter Server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*. USENIX Association.

[11] S. Liu, Z. Du, J. Tao, D. Han, T. Luo, Y. Xie, Y. Chen, and T. Chen. Cambricon: An instruction set architecture for neural networks. In *Proceedings of the 43rd International Symposium on Computer Architecture*, pages 393–405. IEEE Press, 2016.

[12] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.