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
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 speed up 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 learning 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