Impact of System Resources on Performance of Deep Neural Network

Parijat Dube and Zehra Sura
IBM Research, Yorktown Heights NY
pdube,zsura @us.ibm.com

***Abstract*—The training of deep neural networks (DNNs) require intensive resources both for computation and for memory/storage performance. It is important to enable rapid development, experimentation, and testing of DNNs by improving the performance of these codes. This requires understanding what system resources are exercised by deep learning codes, to what degree the utilization of different resources is impacted by changes in the compute intensity or size of data being processed by the neural network, and the nature of the dependencies between different resource bottlenecks. For this purpose, we are performing an extensive empirical evaluation by varying several execution parameters and running hundreds of experiments with different configurations of DNN training jobs. The goal is to gain a robust understanding of how to tailor system resources and training hyperparameters to the needs of a given deep learning job by accounting for both the DNN model and the dataset.**

Keywords—performance analysis, deep learning, docker

I. INTRODUCTION

Today machine learning is being pervasively applied to new application scenarios, and this is placing increasing demands on compute resources. Machine learning service is offered by major cloud services providers, including IBM Watson Machine Learning [1], Amazon Sagemaker [2], Google Cloud Machine Learning Engine [3], and Microsoft Azure [4]. These services support building, training, deployment, and inferencing for machine learning models. In particular, the training of deep neural networks (DNNs) used in deep learning [5] requires intensive resources both for computation and for memory/storage performance. There is a lot of potential for successful learning when using DNNs. However, the design of DNNs is not well understood, and developing a DNN that works well in a new application scenario includes a trial-and-error process. Therefore, it is important to enable rapid development, experimentation and testing of DNNs by improving the performance of these codes. In the context of cloud service providers, this means that their platforms need to offer efficient resource scaling, full life cycle management, fault tolerance, monitoring, and object storage.

A two-fold approach to improve performance includes: (1) improving the utilization of existing systems across multiple jobs from one or more users (e.g. in a cloud deployment), and (2) building systems that are more powerful and balanced to serve the needs of deep learning applications of interest. Both of these approaches require understanding what system resources are exercised by deep learning codes, to what degree the utilization of different resources is impacted by changes in the compute intensity or size of data being processed by the neural network, and the nature of the dependencies between different resource bottlenecks.

In this study, we aim to understand the balance of system resources required for efficient execution of deep neural networks, and how this balance is affected by changes in computation/communication ratios that occur due to the structure or design of the neural network or the input dataset. Note that the runtime of a job is a complex function of many factors, that is even more complex when using shared infrastructure. We gathered empirical performance data using training some well-known deep neural networks with ImageNet dataset [6] on a real system, while varying several system parameters. We ran hundreds of experiments to help us understand performance relationships and to make projections on likely bottlenecks based on current performance trends. In the following sections, we describe our experimental methodology, present preliminary results, and discuss potential impact of our work.

II. METHODOLOGY

A. Experimental System

Our experimental platform is a Kubernetes managed cloud cluster supporting nodes with heterogeneous GPUs. The deep learning (DL) jobs were run in a Kubernetes pod [7] deployed on cluster nodes. The node used in our experiments has 32 CPUs and 64 GBs of memory. Each DL job involves training a standard Convolutional Deep Neural Network for image classification using the Imagenet dataset.

B. Parameters Studied

The performance of machine learning jobs depends on several factors including:

- **DNN model**: includes model size, number of layers, number of neurons per layer, interconnection topology, compute intensity of activation functions.
- **DNN model framework**: includes different code implementations for the same functionality, e.g. Caffe [8], Caffe2, Torch, PyTorch, Tensorflow [9], Theano.
- **Dataset**: includes modality, size, and encoding of data, as well as choice of training and testing datasets.

0-7695-6375-9/18/$31.00 ©2018 IEEE
DOI 10.1109/ISPASS.2018.00025

- **Training framework and hyperparamters**: includes batch size, learning rate, and the use of distributed learning.
- **Resource configuration**: includes the number and type of cpus and gpus, the interconnection topology, size and type of memory, storage, and network links.

Table 1 lists the paramters that were used to derive the different runtime configurations in our initial set of experiments. Each row of the table gives the set of choices used for the corresponding parameter. We collect data for all possible combinations of these choices.

Table 1: List of Configuration Parameters

Parameter	Choices
DNN model	AlexNet, Inception3, ResNet50, VGG16
Framework	Tensorflow,Caffe
Batch size	64, 218
Number of cpus	2, 4, 8, 16, 24, 30
Number of gpus	1, 2
Type of gpu	NVIDIA P100 and V100 GPUs

C. *Metrics*

For each experimental configuration, we use two measures of performance in our evaluation: (1) throughput in terms of the number of images processed, and (2) GPU utilization numbers obtained from *nvidia-smi*.

III. Preliminary Results

For the first round of experiments, we focused on two configuration parameters, number of CPU threads and the batch size. The batch size is a hyperparmater in a DNN which determines the number of data samples used in one iteration of the training job. The experiments were conducted in a pod configured with one or two NVIDIA P100 GPUs [10].

A. *Impact of Number of CPU Threads*

We first study the impact of CPU threads allocated to a pod on the throughput of a DNN training job. Figure 1 and 2 show the sensitivity of throughput (images processed/sec) of a DNN training job to the number of CPU threads allocated to the pod.

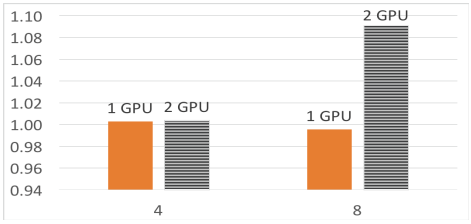

Figure 1: Throughput scaling of Caffe DNN with increasing CPU threads from 2 to 8 for 1 and 2 GPUs

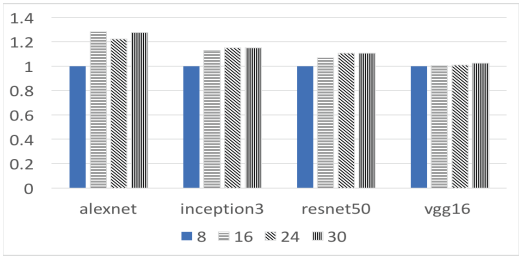

Figure 2: Throughput scaling of Tensorflow DNNs with increasing CPU threads from 8 to 30 at batch size 64.

While throughput of Caffe vgg16 model saturates at 4 and 8 CPU threads for 1 and 2 GPU case, the throughput of Tensorflow models benefit with increased CPU threads till 30. Further with Tensorflow, the sensitivity to CPU threads is different for different models, with alexnet benefitting the most (1.28x) and vgg16 the least (1.03x) when increasing CPU threads from 8 to 30. This gain is tied to DNN model implementation governing the local processing at GPU and the inter-GPU communication during each iteration.

B. *Impact of Batch Size*

The total number of iterations required to finish one training epoch is dependent on the batch size. Higher the batch size more is the GPU RAM required. Since GPUs also store the DNN model along with the batch samples, the maximum batch size that can be used with given GPU is also dependent on the size of the DNN model. The optimal batch size giving the right tradeoff between convergence time and model accuracy depends on the DNN model and the GPU type. Figure 3 shows the effect of increasing batch size from 64 to 128 for two differet Tensorflow DNNs. With CPU threads fixed (30), alexnet can gain upto 1.2 while inception3 gain is only 1.05x.

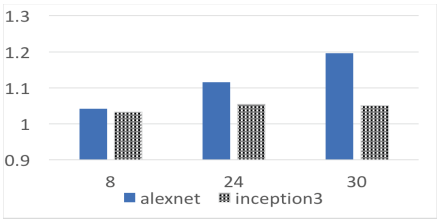

Figure 3: Throughput scaling of Tensorflow DNNs with increasing batch size per iteration from 64 to 128 at different number of cpu threads.

IV. Discussion

The experiments and results described here are a preliminary sample of the overall study that is underway. We will be expanding the configuration parameters and corresponding choices to perform a thorough empirical evaluation. The goal is to gain a robust understanding of how to tailor system resources and training hyperparameters to the needs of a given deep learning job, taking into account the DNN model and dataset. This will help us design better systems, improve cluster utilization and efficiency of deep learning jobs, and as a result help realize the potential of this exciting new technology.

REFERENCES

[1] IBM Corporation. 2018. IBM Watson Machine Learning. (2018). https://developer.ibm.com/clouddataservices/docs/ibm-watson-machine-learning/

[2] Amazon Web Services. 2017. Amazon Sagemaker. (2017). https://aws.amazon.com/sagemaker/

[3] Google Inc. 2018. Google Cloud Machine Learning Engine. (2018). https://cloud.google.com/ml-engine/

[4] Microsoft Azure. 2018. Machine Learning services. (2018). https://azure.microsoft.com/en-us/services/machine-learning-services

[5] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep Learning. *Nature* 521, 7553 (2015), 436–444.

[6] Imagenet, http://www.image-net.org/

[7] https://kubernetes.io/docs/concepts/workloads/pods/pod/

[8] BVLC Caffe, https://github.com/BVLC/caffe.

[9] Tensorflow, https://www.tensorflow.org/.

[10] Nvidia gpus, http://developer.nvidia.com/deep-learning/.

[11] P. Xu, S. Shi, and X. Chu, "Performance Evaluation of Deep Learning Tools in Docker Containers," *Published 2017 in ArXiv*, available at https://arxiv.org/pdf/1711.03386.pdf

[12] J. Gu, H. Liu, Y. Zhou, and X. Wang, "DeepProf: Performance Analysis for Deep Learning Applications via Mining GPU Execution Patterns," *Published 2017 in ArXiv*, available at https://arxiv.org/pdf/1707.03750.pdf

[13] H. Kim, H. Nam, W. Jung and J. Lee, "Performance analysis of CNN frameworks for GPUs," *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, Santa Rosa, CA, 2017, pp. 55-64.