Effect of Performance on Containerized Deep Learning Applications

Vinutha G Siddaramanna
IBM India Pvt Ltd
Bangalore, India
vinuthags@in.ibm.com

Anto Ajay Raj John
IBM India Pvt Ltd
Bangalore, India
antojohn@in.ibm.com

***Abstract* - The advent of machine learning, deep learning has brought in a paradigm shift on workloads running on server systems, with the heavy cost associated with running them on superior hardware. In the application-development and cloud deployment domain, containers have become the cynosure of all the developers. Containers come with a low overhead, provide increased distributed computing capabilities and reduced complexity by abstracting away application dependencies. Thereby, containers provide for easier application sharing, migration and cloning, by clubbing the application and its development environment into a single component. Deep Learning(DL) frameworks have many dependencies and it has a rapid development cycles and changes. Containerization helps developers overcome these challenges and therefore are rising in importance with deep learning frameworks.**

In this paper, we compare the performance of native versus containerized Deep Learning applications using deep bench and fathom benchmarks. These two benchmarks almost cover all the aspects of deep learning computation from a micro and macro application level. Deep bench identifies fundamental operations in Deep Learning applications like gemm, reduce etc. and evaluate the same; while Fathom provides a collection of deep learning workloads built around Tensorflow covering the spectrum of current deep learning applications like Convolution neural network, Recurrent neural network, etc.

We have demonstrated that except for a few aspects of performance differences; largely containers prove to be a great platform to package and host deep learning applications.

Keywords – Artificial Intelligence, Deep learning, Docker ,GPU, Containers, Deep bench, Fathom ,Neural Networks, Performance

I. INTRODUCTION

With the availability of large amounts of digital data and the availability of the huge amount of computing power in the form of Graphical Processing Units [GPU] along with a high degree of specialisation in solving complex algebraic problems, a new realm has started in Artificial Intelligence [AI].

A. *Machine Learning and Deep Learning*

Machine learning [ML] is one of the biggest fields in artificial intelligence. Machine learning can be described as scientific representation of decision making, means the ability to represent a human decision process in mathematical form which is implemented as algorithm. Which in turn means, a machine needs to be taught and trained. This allows the system to learn based on experience and to interpret the results and continuously improve the accuracy of the results.

Deep Learning [DL], is a subset of Machine Learning. Deep learning attempts to model decision making using Neural Networks, mimicking the way the human brain itself processes senses (sight, sound, and others). Deep learning algorithms are based on Artificial Neural Networks [ANN], which is inspired by biological neural networks consisting of multiple layers of neurons.
Implementing neural network algorithms is not an easy task. It requires the understanding of neural network logic, the multiple complex layers and connections, heavy training process across different machines and to make them run on both CPU and GPU. This needs several lines of code to implement.

This is where Deep Learning Frameworks come into rescue, they take care of most of the complexities involved in running neural network algorithms, and all the code necessary to interact with the CPU/GPU library is readily available. They provide higher level APIs to make the deep learning code easier to implement. Some of the popular Deep Learning frameworks are Caffe, TensorFlow and IBM Caffe.
There has been widespread adoption of the deep learning frameworks as these frameworks abstract the underlying hardware and shift the onus on them to produce high performance vectorized CPU code or GPU code. The frameworks also use optimized libraries thereby accentuating performance with little involvement from the developer. Thus, the developer could focus his efforts building the model than spending time to optimize performance of the program for a hardware platform.

B. *Containers*

Linux Containers (LXC), have been there from very long time, with a goal of providing isolation for applications through operating system-level virtualization. Containers/Dockers, which are built up on Linux Containers brought in even more significant improvement to the LXC's capabilities by using the Linux kernel's Namespace and CGroup features. Docker an open source container virtualization technology, has simplified application development. While there are multiple frameworks, libraries, tools and packages that needs to be installed for setting up the platform for Deep Learning application development, all these can be containerised into one single image.

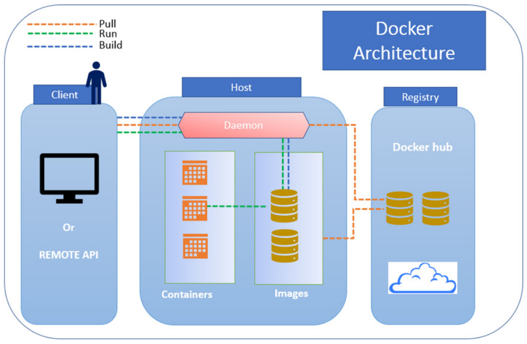


Figure 1 – Docker architecture

Docker removes the computational overhead required to virtualize the hardware for a guest OS to use it substantially. Docker's simple architecture removes the unnecessary overhead by leveraging the Namespace and CGgroups of Linux kernel. Docker provides the required isolation for applications by stacking the required packages into one single image, simplifies managing different tools for Deep Learning, and later can be transferred and quickly installed onto any other docker enabled host. Docker images can be shared and distributed easily.

GPUs usage in image processing and parallel computing applications has been widespread and many studies have shown that significant speedup could be achieved with GPUs for various applications . GPUs have huge computing power and allow processing on parallel data elements simpler and faster. Machine Learning typically involves two phases training and inference phase. In the training phase the model is built by learning from a huge dataset. And the training phase could be applied parallel on datasets. GPU becomes an excellent fit to take advantage of the parallel computation on datasets. Various studies have shown that training could be speedup for various deep learning and machine learning algorithms .

II. BACKGROUND

A. *Nvidia-docker*

Complex models like convolutional neural networks (CNNs) and recurrent neural networks rely heavily on GPUs. Neural network performs complex computations on tens and thousands of matrices. GPU's parallel computing for these basic operations plays a crucial role and they can accelerate the computation. Nvidia-Docker[8] is open source project that brought ease and agility of containers to CUDA programming model. It's a wrapper around docker CLI that transparently provisions a container with the necessary dependencies to execute code on the GPU.

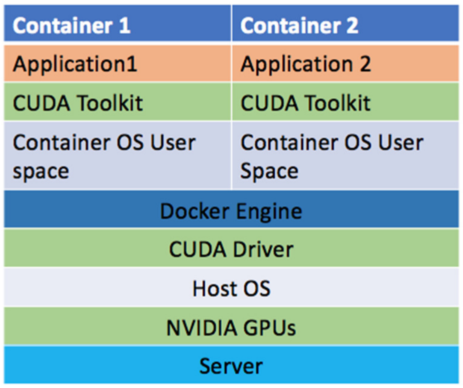


Figure 2: nvidia-docker application stack

B. *Deep bench*

Deep bench's goal is to benchmark operations important to deep learning applications on different hardware platforms and identify which hardware provides the best performance for basic operations used in deep neural networks.

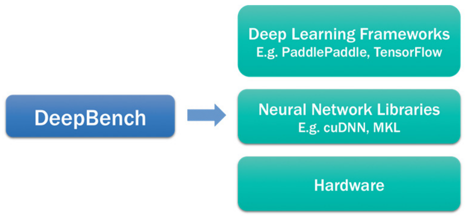


Figure 3: Deep bench

DeepBench does not use deep learning frameworks, rather it uses the neural network libraries to benchmark the performance of basic operations on different hardware. We there by cannot use DeepBench to find the time required to train the entire model. DeepBench benchmarks only underlying operations in a deep learning network and helps hardware vendors and software developers understand performance bottlenecks in deep learning training and inference

DeepBench benchmark suite consists of the following programs targeting the basic operations and we will describe them below individually in detail:

gemm_bench,- benchmarks dense matrix multiples which are commonly used in implementing fully connected layers and are building blocks for other recurrent layers. The GEMM operation is A*B=C. We can specify the matrix multiplications with different sizes.

conv_bench - targets convolutions that account for a vast majority of computation for networks operating on images and videos. Convolutions form important parts of speech and natural language modelling networks. Thus, performance benchmarking convolutions, is vital as they are the single most important layer from a performance viewpoint. The data to the network is presented in image, feature maps, rows and columns.

rnn_bench - benchmarks three types of recurrent cells i.e.. vanilla RNNs, LSTMs and GRUs.

all_reduce - since neural networks are often trained using multiple GPUs or systems each with multiple GPUs in a distributed fashion.Thereby, performance evaluation of Multi GPU communication is a primitive need. NCCL's nccl_single_node and nccl_multiple_node and osu_allreduce benchmarks are also included to understand the communication patterns of deep learning networks.

C. FATHOM

Fathom is a collection of workloads representing deep-learning models used in areas such as machine translation, natural language processing, speech recognition, image classification and reinforcement learning. Representativeness, diversity and impact were the selection criteria for the workloads chosen. All workloads in Fathom use TensorFlow and contain no post or pre-processing steps. The following contains a brief description of each of the fathom workloads:

Sequence to Sequence translation
seq2seq is a technique developed by Google to solve machine translation using a recurrent neural network. The technique involves a multi-layer pipeline of LSTMs to extract meaning of sentence and then produce a translation in another language. The model keeps track of original sentence context with the help of an attention model. The core neural network is composed of three 7-neuron layers.

End to end memory networks
They are extension to memory networks developed by Facebook's AI research team to have a model that can explicitly store and recall information. They remove the need for input type annotation and greatly systematize model training.
Deep Speech
It's a scalable deep learning speech recognition model by Baidu research which uses spectrogram inputs and learns to transcribe phenomes. By using a connectionist temporal classification loss function, it reduces cost of training data production. It was designed well to perform well on a GPU.

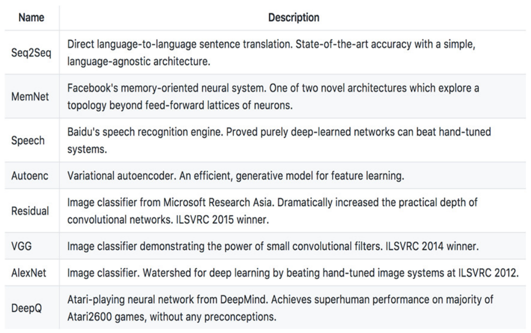
Deep reinforcement learning
This is a deep Q learning algorithm that chooses actions based on the in-game feedback it receives. The method uses a CNN composed of 2-3 convolutional layers and 2-3 dense layers that decides the actions. The program leverages the Atari emulation environment

VGG-19
vgg is an implementation of 19-layer CNN that was developed drawing motivation from AlexNet.

Residual networks
Residual networks are a novel solution from Microsoft Research Asia to address the problem that model depth diminishes both training and validation error. They added identity connections across every pair of convolutional layers and thus could train neural networks 150 layer deep.

AlexNet
With widespread adoption among the architecture community, AlexNet was chosen to ensure continuity and to serve as reference point for other models in Fathom.

Thus, Fathom provides a diverse set of workloads reflecting the state of the art research in deep learning. Readers could refer to https://github.com/rdadolf/fathom for a more detailed understanding of Fathom workloads. Table below provides an overview of the Fathom workloads.



Name	Description
Seq2Seq	Direct language-to-language sentence translation. State-of-the-art accuracy with a simple, language-agnostic architecture.
MemNet	Facebook's memory-oriented neural system. One of two novel architectures which explore a topology beyond feed-forward lattices of neurons.
Speech	Baidu's speech recognition engine. Proved purely deep-learned networks can beat hand-tuned systems.
Autoenc	Variational autoencoder. An efficient, generative model for feature learning.
Residual	Image classifier from Microsoft Research Asia. Dramatically increased the practical depth of convolutional networks. ILSVRC 2015 winner.
VGG	Image classifier demonstrating the power of small convolutional filters. ILSVRC 2014 winner.
AlexNet	Image classifier. Watershed for deep learning by beating hand-tuned image systems at ILSVRC 2012.
DeepQ	Atari-playing neural network from DeepMind. Achieves superhuman performance on majority of Atari2600 games, without any preconceptions.

Figure 4: Fathom benchmarks description

III. APPROACH

To understand if there are any performance difference between host and on Docker containers, we ran DeepBench benchmark and Fathom workload on both host and Docker containers. For our experiment, we used a Power8NVL hardware which has 4 P100-SXM2 GPU cards and host OS was RHEL7.4. Some of the important packages that are needed for DeepBench and Fathom to work are listed in below table, it also includes the version details. We used PowerAI to seamlessly install various frameworks on host and Nvidia-Docker container.

To build docker image, we used 9.0-cudnn7-runtime-centos7, as the base docker image upon which the other required packages for Deep Bench and Fathom were installed on both host and nvidia-docker. To ensure the Benchmark and Workload results are accurate, we made sure that the system was clean and there was no other workload running, and we also checked to make sure the CPU and GPU's were not used for any other workloads.

The packages details can be seen in below table.

PowerAI	Power-mldl-cuda9.1
CUDA	9.1
Tensorflow	1.5.0
NCCL	1
cuDNN	9.1

IV. EXPERIMENTAL RESULTS

We started with DeepBench benchmark suites on host first. The RNN_Bench, GEMM_Bench and CONV_BENCH results were acquired for host and then on container. The charts below show the results on both host and container and the comparison.
We tried different types of CONV_BENCH kernels and found that the time taken to execute a kernel is same when executed on the host and container. (Figure. 5).

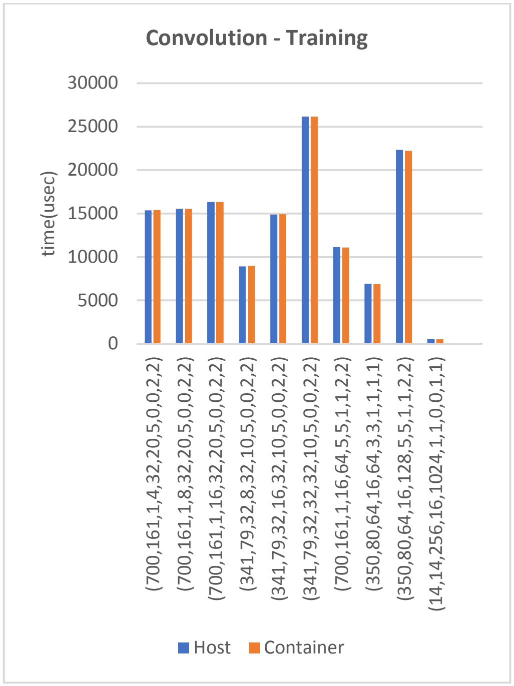


Figure. 5. DeepBench – CONV_BENCH

The same kind of behaviour was also found when we executed the kernels for GEMM operations. (Fig. 6). However, a slightly different behaviour was observed in the RNN_BENCH kernels.

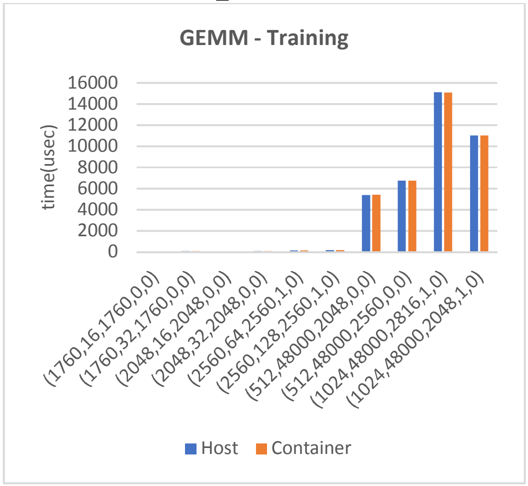


Figure. 6. DeepBench – GEMM_BENCH

In Figure. 7,8,9 we show the detailed time split for three different RNN operation like Vanilla, LSTM and GRU. In the case of Vanilla and LSTM, the kernels launched on host and container takes almost the same time. GRU based kernels showed a consistent poor performance for kernels launched on the containers. Examining the NVprofs, we found that GRUs are fast and frequent GPU compute kernels. This meant that there was very frequent kernel launches which showed poor performance on the containers. The launch time per kernel when it was frequently called showed a degradation.

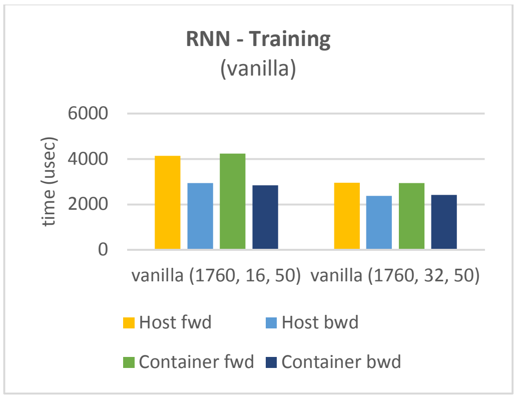


Figure. 7. DeepBench – RNN_BENCH (vanilla)

We found that running simple bandwidth tests to demonstrate the memory transfers between host and device showed considerable performance difference on the container (Figure 8). This performance degradation is intermittent as it depends on where and how the container is NUMA located in a server. In a two socket system, there is a possibility that the GPU's used for the work can come from one socket and the container resides on the other socket, thus resulting in performance penalty. When we tried to affinitize the container to the socket where the GPUs were residing the performance of bandwidth tests increased a lot. (Figure 10)

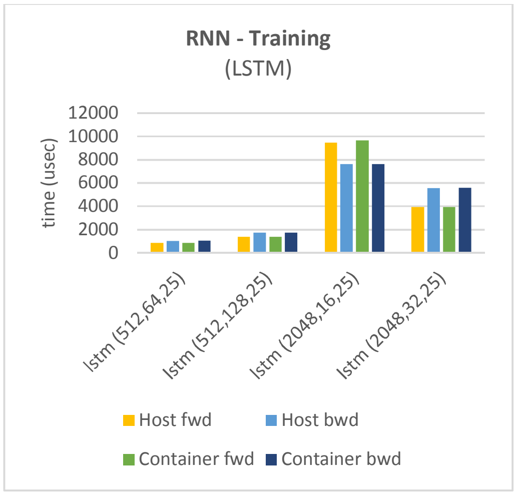


Figure. 8. DeepBench – RNN_BENCH (LSTM)

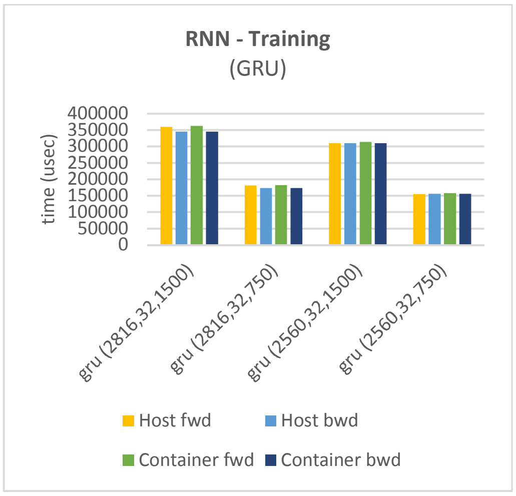


Figure. 9. DeepBench – RNN_BENCH (GRU)

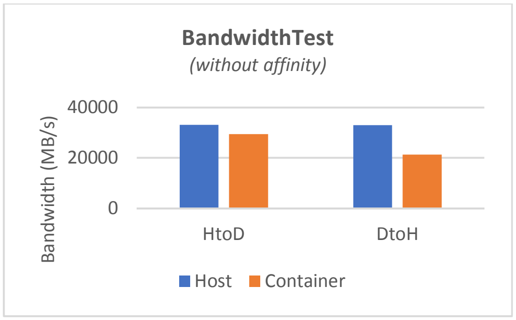


Figure. 10. BandwidthTest (without affinity)

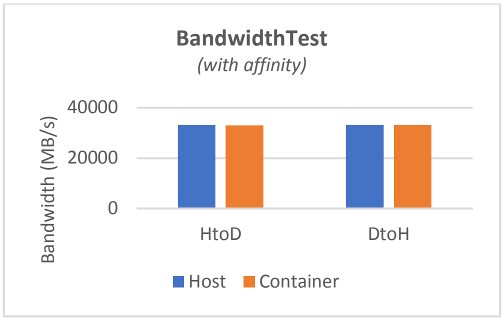


Figure. 11. BandwidthTest (with affinity)

Further experiments with Fathom on different convolution-based benchmarks showed the host and container performance to be on par.

V. CONCLUSION

Cloud computing becoming more ubiquitous, makes it a must for all deep learning applications to run on them without any performance hiccups. In this paper, we had discussed the different deep learning benchmarks and their effect on running them in a container. Even though in most cases, we see that the performance of these applications running on host is on par with a container; we still find some aspects like data loading, affinitizing, frequent kernel launches which can alter the performance of the containers. Hence, one has to take care of these vital properties of the containers and the applications to run the artificial intelligence workloads seamlessly on cloud using container as a vehicle.

With our success in making sure that we could get similar performance on container as that of hosts, we plan to create a container performance manager which can monitor and identify performance issues of deep learning applications on the container.

VI. REFERENCES

[1] Jürgen Schmidhuber, Deep learning in neural networks: An overview,Neural Networks, aaVolume 61,2015,Pages 85-117,ISSN 0893-6080,

[2] Rajat Raina, Anand Madhavan, and Andrew Y. Ng. 2009. Large-scale deep unsupervised learning using graphics processors. In Proceedings of the 26th Annual International Conference on Machine Learning (ICML '09). ACM, New York, NY, USA, 873-880.

[3] Peter Bakkum and Kevin Skadron. 2010. Accelerating SQL database operations on a GPU with CUDA. In Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU-3). ACM, New York, NY, USA, 94-103.

[4] P. Trancoso and M. Charalambous, "Exploring graphics processor performance for general purpose applications," 8th Euromicro Conference on Digital System Design (DSD'05), 2005, pp. 306-313.

[5] T. Haryanto, H. Suhartanto and X. Lie, "Past, present, and future trend of GPU computing in deep learning on medical images," 2017 International Conference on Advanced Computer Science and Information Systems (ICACSIS), Bali, 2017, pp. 21-28

[6] Y. J. Mo, J. Kim, J. K. Kim, A. Mohaisen and W. Lee, "Performance of deep learning computation with TensorFlow software library in GPU-capable multi-core computing platforms," 2017 Ninth International Conference on Ubiquitous and Future Networks (ICUFN), Milan, 2017, pp. 240-242.

[7] S. Shi, Q. Wang, P. Xu and X. Chu, "Benchmarking State-of-the-Art Deep Learning Software Tools," 2016 7th International Conference on Cloud Computing and Big Data (CCBD), Macau, 2016, pp. 99-104.

[8] Deep Bench – https://github.com/baidu-research/DeepBench

[9] Fathom - https://github.com/rdadolf/fathom