Intel® instances for AI workloads on Microsoft* Azure

A step by step instructions user guide

Ouchiyama Hiroshi

11/29/19

V1.0

CONTENTS

DISCLAIMERS

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation.

Performance varies depending on system configuration.

No product or component can be absolutely secure. Check with your system manufacturer or retailer or learn more at intel.com

INTRODUCTION

Machine Learning (ML)/ Deep learning (DL) are next generation AI technologies bringing new revolutions in the automation. With the availability of the data and compute power, DL applications accuracy is raising above the human level accuracy in certain complex technical challenges which are difficult to handle with previous technologies.

This user guide is to recommend the right Microsoft* Azure cloud instances along with relevant services and tools to launch and utilize the cloud instances effectively for machine learning and deep learning workloads on Intel® architectures. Here are some of the advantages of using the Intel® Xeon® scalable family of servers for ML/DL workloads -

- Great for handling high memory intensive workloads and 3D-CNN topologies used in Medical imaging, GANs, seismic analysis, genomic sequencing etc.
- Great for real-time inferencing even at lower batch sizes due to flexible core control with simple numactl commands.
- Great ecosystem supported hardware for distributed training over large clusters (i.e. compute at source of data), thereby avoiding the need of adding bulk storage and expensive cache mechanisms for training over scale-up architectures (one server with many accelerator cards)
- Great TCO due to support of heterogeneous workloads (HPC/BigData/AI) on same cluster.
- SIMD capabilities to address the computational requirements of many practical deep learning applications.
- Single production level scalable infrastructure for both training (also transfer learning) and inference.

Typical Deep learning application development and deployment involves the following stages –

These stages require multiple resources along with their orchestration mechanisms given below -

- Computational power
- Memory
- Storage for Datasets
- Communication link between compute nodes
- Optimized software

Selection of the right resources needs comprehensive study and benchmarking for great TCO (Total Cost of Ownership). Here are the technical capabilities and the ecosystem that comes along with Xeon® Scalable family of servers due to Intel® legacy in computation -

COMPUTE POWER AND MEMORY

Usually the requirement for compute power and its resources varies depending on the type of algorithm (ML) or neural network topology (DL) used for your application. Although, compute power is measured with raw flops, arithmetic intensity of an algorithm decides the utilization of raw flops. Xeon® Scalable family of servers are designed to handle both serial and parallel workloads with flexible core control to get full utilization of memory and cores for most of the workloads. Here are some capabilities which gives a glimpse of their strength to use for deep learning -

- 512-bit SIMD capability (AVX-512 engines) with which each core can compute up to 64 flops per cycle which is enough for most of the practical deep learning applications [1].
- DL Boost with VNNI instruction fused into AVX-512 pipeline which can compute up to 256 tops per core per cycle accelerating the INT8 inference throughput and reducing the latency (available in second generation of Xeon® scalable family i.e. cascade lake and future generation of servers).
- Increase in memory capacity per socket (up to 4.5TB) with DCPMM in the Cascade lake servers to support 3D CNN and high dimensional tensor computation.

STORAGE

Often datasets size required for deep learning training will be hundreds of GB which requires fast storage access mechanisms. Due to the non-availability of efficient scalability methods for distributed training, many deployments in the early stages of DL revolution encouraged towards the scaleup architecture adding to the below overheads –

- limiting the scalability to single node with multiple accelerator cards.
- Latency involved in distributed datasets to move to single node (server) before training
- Adding bulk storage at single node or expensive cache mechanisms for fast training

Recent development of MPI based distributed training with Horovod made the scaling easy across nodes. Horovod with Xeon® clusters can be utilized to scale over thousands of nodes, thereby reducing the time to train and overheads mentioned above. Xeon® ecosystem has the support for multiple storage mechanisms (block/file/object storage) to go for production level scalability.

COMMUNICATION LINK

For DL inference and Single node training, there won't be any need of high network Band width communication between compute nodes. So 10Gig link will be good enough for most of the development and deployment. But for distributed computing, communication latency decides the scalability performance. Practical experimentation showed that 25 Gig ethernet will be enough to get above 90% scaling up to 8 nodes (varies based on computation latency of a topology). But beyond 8

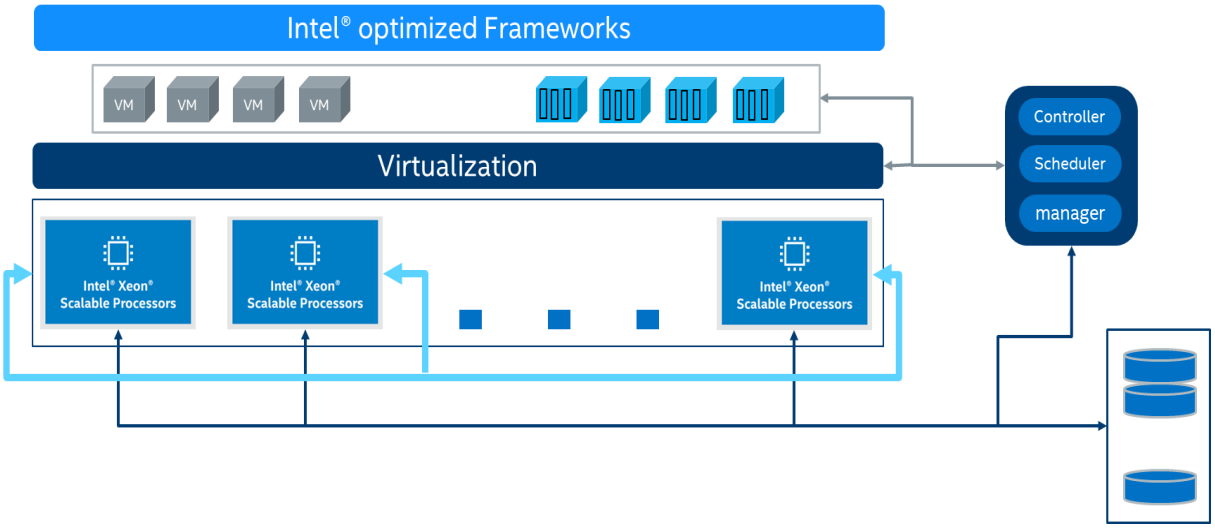
nodes, it is recommended to use Ethernet with RDMA, IB or OPA as the communication link. Intel® Xeon® servers supports all these mechanisms seamlessly.

OPTIMIZED SOFTWARE

Intel® optimized many publicly available popular frameworks by integrating open source MKL-DNN library which extracts the full potential of Xeon® servers for deep learning algorithms [2]. Along with optimized frameworks, Intel® has heterogeneous inferencing toolkit like openVINO™ to provide the seamless deployment across multiple hardware [3]. Bigdata analytics users can move from ML analytics to sophisticated intelligent analytics with BigDL library fused seamlessly into spark framework along with Analytics Zoo [4].

Below is the block diagram which stitches all these resources together for scalable deep learning training and inference. Here are the some of the prominent mechanisms used for scaling DL applications –

- **Compute nodes** – Gold (6 series) and platinum servers of Intel® Xeon® Scalable family
- **Management and storage nodes** – Bronze or Silver servers of Xeon® scalable family or Xeon® Broadwell
- **Storage mechanisms** – CEPH, S3, NAS, Luster, GlusterFS etc.
- **Virtualization Mechanisms** - KVM, VMware, XenServer, Hyper-V etc.
- **Container orchestration** - Docker Swarm, Kubernetes*, Mesos, OpenShift etc.
- **Schedulers and Management** – PBS, SLURM, VMSphere, XenCentre, OpenStack* etc.



All the above technology ecosystem is adopted with Intel® Xeon® servers to implement scalable deep learning pipeline (collect, store, preprocess, analyze and scale) efficiently. Here are the list of Azure tools and services to accomplish the same -

- Compute nodes – Azure VM instances for computing
- Storage – Azure Storage (DISK Storage, BLOB Storage, Azure Data Lake Storage, File Storage)
- Management – AKS, ACI and Azure Machine Learning service
- Virtualization – customized version of Hyper-V
- Container orchestration – Kubernetes*
- Schedulers and Management – Azure Scheduler, Azure Monitor

AZURE CLOUD INSTANCES FOR AI APPLICATIONS

There are the many cloud instances available in Azure to develop and productize multiple applications. But below sections give information on the instances suitable for ML/DL applications which are equipped with Intel® Xeon® family of servers having the DL computation capabilities explained in previous sections.

COMPUTE OPTIMIZED INSTANCES

- Fsv2-series (Intel® Xeon® Platinum 8168 CPU @ 2.70GHz –**Skylake, hyperthreading off**). We can choose from minimum 2 vCPUs to maximum 72 vCPUs.

HIGH PERFORMANCE COMPUTE

- HC-series (Intel® Xeon® Platinum 8168, 44 vCPUs – **Skylake, hyperthreading on**). It is provided as 2 sockets instance (22 cores on each).

Note: As of Nov 2019, there is no Cascade lake instance on Azure.

RECOMMENDATION FOR VARIOUS ML/DL WORKLOADS

- ✓ **Fsv2 instances (<= 8 vCPUs)** are suitable ML applications and low compute DL inference applications.
- ✓ **Fsv2 instances (> 8 vCPUs)** are suitable for all kinds of Deep learning inference workloads
- ✓ **HC instance** are suitable for Distributed Deep learning training due to high NW performance required for inter-node communication.
    - For fast training scalability over large cluster **HC** are preferred which comes with 192GB memory suffice for most of the topologies.
- ✓ **HC instance** are suitable for memory intensive workloads which use 3D-CNN topologies with memory requirement more than 192GB.

Note:

- On Azure, Hyper threading is disabled in general so 1 vCPU can be said 1 physical core. However, a few VM series recently enable the Hyperthreading and this Fsv2 series is one of them. This means, in Fsv2 series, 2 vCPU equals 1 physical core. On the other hand, HC is no hyperthreading so 1 vCPU equals 1 physical core. In Fsv2 series, more than 64 vCPU's require one of these supported guest OSes: Windows Server 2016, Ubuntu 16.04 LTS, SLES 12 SP2, and Red Hat Enterprise Linux, CentOS 7.3, or Oracle Linux 7.3 with LIS 4.2.1
- In Fsv2 series, 72 vCPU Instance is isolated to hardware dedicated to a single customer.
- HC VMs feature 100 Gb/sec Mellanox EDR InfiniBand in a non-blocking fat tree configuration for consistent RDMA performance. HC VMs support standard Mellanox/OFED drivers such that all MPI types and versions, as well as RDMA verbs, are supported as well.

For ease of understanding, below table gives recommendation based on the ML/DL topologies. This table suggests the minimum number of vCPUs with each instance types to get reasonable performance and latency requirements.

Topology	Azure Instance
Training and inference of ML and small NN topologies Ex : Regression, K-means, KNN, SVD, PCA, LDA, RCF, XGBoost, Lenet, Alexnet, Squeezenet etc..	All Fsv2 instances with **<=8** vCPUs
Inference of medium NN topologies for classification, object detection, segmentation Ex: Resnet-50/101/152, Inception-v3/v4, VGG-16/19, SSD-mobilenet, Yolo-tiny-v2/v3 etc..	Fsv2 **>=8** vCPUs for FP32
Inference of other Deep NN topologies for object detection, segmentation, NLP, Reinforcement learning, GANs Ex: Faster-RCNN, SSD-VGG, Yolov2,v3, GNMT, Transformer, U-net, V-net, NCF, W&D, A3C, BERT, GAN etc..	Fsv2 **<=32** vCPUs for FP32
Inference on memory and compute intensive NN topologies Ex: 3D-U-Net, 3D-GAN etc...	HC for inference
Single node Training of medium and Deep NN topologies	Fsv2 **<=2**vCPUs
Distributed training on multiple nodes with all medium and deep NN topologies	**Fsv2 >=48** vCPUs, HC
Distributed training with further memory intensive like 3D CNN topologies	HC

Users are expected to benchmark their topology and select the instance suitable for production.

Rule of thumb

- use **Fsv2** series instances are suitable for Deep learning inference and single node training
- use **HC** instances are suitable for distributed training over a larger cluster due to high throughput communication network.

AZURE TOOLS & SERVICES FOR DEEP LEARNING

Azure has multiple tools and services to ease the deep learning application development and deployment.

VM Images

Azure is providing various VM base images with components required for Deep learning in Azure Marketplace

Intel® Optimized Data Science VM for Linux - These VMs come with Python environments optimized for deep learning on Intel® Xeon® Processors. These environments include open source deep learning frameworks with Intel® MKL-DNN as a backend for optimal performance on Intel® Xeon® Processors. These environments require no changes to existing code and accelerate deep learning training and inference. Additionally, this offering includes all software packages available on the base DSVM with several popular tools for data science and ML which are already pre-installed, configured and tested. This base image is available only on Fsv2 series and HC.

The Deep Learning Reference Stack – This is an integrated, highly-performant open source stack optimized for Intel® Xeon® Scalable platforms and includes highly tuned software components across the operating system (Clear Linux OS), deep learning framework (TensorFlow* and others), deep learning libraries (Intel® Math Kernel Library for Deep Neural Networks (MKL-DNN)) and other software components.

BigDL: Distributed Deep Learning for Apache Spark - A distributed deep learning library for Apache Spark with high performance and efficiently scale out.

Analytics Zoo: A unified Analytics + AI platform - A unified analytics + AI platform for distributed TensorFlow*, Keras and BigDL on Apache Spark

Recommendation: use the **Intel® Optimized Data Science VM for Linux (Ubuntu)** for Deep Learning Frameworks and **Azure Databricks** described below for Spark users.

Service Engines

Azure provides many services to ease the job of deep learning. Here are some of the tools and services that can help novice to expert data scientists to play with Deep learning.

Azure Cognitive Services (Pre-trained models) – Several pre-trained deep learning models are provided as a web service. Users can invoke them by using REST call from their applications.

- Computer Vision
- Face
- Video Indexer
- Ink Recognizer
- Form Recognizer

- Content Moderator
- Anomaly Detector
- Personalizer
- Speech Services
- Speaker Recognition
- Bing Visual Search
- Text Analytics
- Translator Text
- Immersive Reader
- etc…

Azure Cognitive Services (Tools to build custom models) – Azure is providing some GUI tools to build user's custom models using the deep learning technology. By using these services, every user can build a model without coding and some service can export the model in various format such as TensorFlow*, ONNX, etc…

- Custom Vision
    - User can build an image classification model and an object detection model only by operating on GUI. After building a model, it can be exported as various format.
- Language Understanding (LUIS)
- Custom Translator
- Custom Speech
- etc…

Azure Machine Learning service – This service provides cloud-based environment for users to preprocess data, train, test, deploy, manage, and track machine learning models. This service fully supports open-source frameworks such as PyTorch*, TensorFlow*, and libs like scikit-learn. These services and can be used for any kind of machine learning, from classical ml to deep learning, supervised and unsupervised learning either by writing own code or using the visual interface. Moreover, it has Automated ML feature which is including the feature execution, the algorithm selection and the hyper-parameter tuning. Trained model is can be deployed as a web API into Azure Kubernetes* Services (AKS) by simple operations.

Azure Batch – This is a tool to run large-scale parallel and high-performance computing (HPC) batch jobs efficiently in Azure. Azure Batch creates and manages a pool of compute nodes (virtual machines), installs the applications users want to run, and schedules jobs to run on the nodes. There is no cluster or job scheduler software to install, manage, or scale. Instead, users use Batch APIs and tools, command-line scripts, or the Azure portal to configure, manage, and monitor their jobs. This can be used for the batch inference job.

ACI - Azure Container Instance – This service offers a simple way to run a container in Azure, without having to manage any virtual machines and without having to adopt a higher-level service. For scenarios where you need full container orchestration, including service discovery

across multiple containers, automatic scaling, and coordinated application upgrades, next Azure Kubernetes* Service (AKS) is recommended.

AKS – Azure Kubernetes* Service – This service helps in deploying and management of a cluster for distributed training using Kubernetes*. AKS is used by Azure Machine Learning service as a deployment platform for trained models. Users with little knowledge on container management can use this service to deploy and scale.

Azure Notebook – This is like Jupyter Notebook as a service. User can use it free of charge.

Azure Databricks – This service is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform. Designed with the founders of Apache Spark, Databricks is integrated with Azure to provide one-click setup, streamlined workflows, and an interactive workspace that enables collaboration between data scientists, data engineers, and business analysts. In addition, Intel's Analytics-zoo can be integrated into Azure Databricks by following this guide.

Azure market place – Here Intel® optimized VM images are available to use on the CPU instances.

Recommendation: get MKL-DNN optimized Xeon® based containers and VM images for both training and inference

Storage Service

Azure has multiple storage mechanisms. For single node applications Disk Storage will be good enough. But Blob Storage or Azure Data Lake Storage is right mechanism for scalable training and deployment over production clusters.

INSTRUCTIONS TO DEVELOP AND DEPLOY ML APPLICATIONS ON AZURE

Development and deployment of machine learning applications is straight forward by choosing relevant Azure VM instances mentioned above. But to get the optimum performance for NumPy, SciPy and Scikit-learn based applications, install Intel® distribution of Python (IDP). IDP contains Intel® Data analytics acceleration library (DAAL) to get optimized machine learning performance on Azure Intel® instances.

Azure VM:

- Launch VM instances mentioned in the previous section depending on the use case and computational requirement. https://docs.microsoft.com/en-us/azure/virtual-machines/linux/quick-create-portal
- Download and install IDP which consists of DAAL too https://software.intel.com/en-us/distribution-for-python
- To install only DAAL, please follow the link https://software.intel.com/en-us/daal.
- Here is the developer guide for DAAL https://software.intel.com/en-us/daal-programming-guide
- Check references [5], [6] for IDP and DAAL benchmarks.

Azure market place doesn't have Intel® DAAL images so just follow above instructions.

INSTRUCTIONS TO DEVELOP AND DEPLOY DL APPLICATIONS ON AZURE

After choosing the right compute instance, communication network, storage and optimized software, Azure services are required to automate and scale on a production cluster.

In this section we discuss on the procedure to develop and productize the deep learning training and inference applications using Azure IA instances with multiple Azure services.

Rule of thumb

Install frameworks with below commands and set KMP_AFFINITY, KMP_BLOCKTIME and OMP_NUM_THREADS.

- **TensorFlow***: pip(3) install intel-tensorflow
- **MxNet*** : pip(3) install mxnet-mkl
- **PyTorch*** : pip(3) install torch
- **bigDL** : pip(3) install BigDl

Use Horovod with Intel® MPI for distributed training over a single node or a cluster of nodes.

Whatever may be the service from Azure, use the Intel® Azure instances mentioned in above sections and follow these guidelines -

- After launching an instance, run **lscpu | grep avx** and make sure AVX-512 is available in flags list
- After launching a script, set **export MKL_VERBOSE=1** and make sure MKL logs are seen (unset after testing as logging will add latency)
- Follow the MPI settings guideline mentioned here – https://software.intel.com/en-us/articles/using-intel-mpi-library-on-microsoft-azure

STEP BY STEP GUIDE TO SETUP A CLUSTER FOR DL TRAINING OVER AZURE

This section consists of step-by-step instructions to launch the CPU instances and to run Deep learning training on a cluster using Azure services. Every service has its own procedure to setup the cluster for DL training or inference.

Azure VM Service setup:

Launching Deep learning training on Azure VM service is straight forward –

- Launch the desired Intel® hosted instance with Deep learning Base Image described above.

AKS Service setup:

Here are the steps to setup the AKS service and Kubeflow on it for deep learning training and inference TF models.

Azure Machine Learning service setup:

Azure Machine Learning service is complete automated tool to train and deploy deep learning models. Moreover, this tool is recommended as a default ML/DL development tool for Azure users. Initial setup of the training cluster is automated through set of APIs filled with framework, model code, and dataset relevant information.

- Follow the link to setup a training environment first. We recommend using CPU based VM as a computing target.
- While creating the model estimator use the CPU instance type as shown below. Detail is here.

```
compute_config = AmlCompute.provisioning_configuration(vm_size=' Standard_F32s_v2 ',
max_nodes=<desired number of nodes>)
compute_target = ComputeTarget.create(ws, cluster_name, compute_config)
compute_target.wait_for_completion(show_output=True, min_node_count=None,
timeout_in_minutes=20)

est = TensorFlow(source_directory=script_folder,
            entry_script='tf_mnist.py',
            script_params=script_params,
            compute_target=compute_target,
            use_gpu=False)
```

- Although Azure Machine Learning service has the capability of TensorFlow* training with both parameter server and Horovod and PyTorch* training with Horovod as well, we recommend using Horovod for both cases as given in the link –

  https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-train-tensorflow#distributed-training

  https://docs.microsoft.com/en-us/azure/machine-learning/service/how-to-train-pytorch#distributed-training

STEP BY STEP GUIDE TO RUN DL TRAINING OVER AZURE CPU INSTANCES

Azure VM Service:

- Here is a step-by-step guide to launch the training on Azure VM instance - https://www.intel.ai/intel-optimized-data-science-virtual-machine-azure/#gs.6e8hk0
    - Above link shows the instructions using "Intel® Optimized Data Science VM for Linux" image.
    - For instances with multiple sockets instances, we recommend you follow the multi-worker deep learning using Horovod on single node. Here is the link with step-by-step instructions to create multi-worker deep learning on single node - https://builders.intel.com/docs/aibuilders/best-practices-for-scaling-deep-learning-ztraining-and-inference-with-tensorflow-on-intel-xeon-processor-based-hpc-infrastructures.pdf

AKS Service:

- Here are the instructions to run the training on AKS (on Kubernetes*) - https://github.com/Azure/kubeflow-labs/tree/master/2-kubernetes
- Here are the instructions to run the training on kubeflow on AKS service - https://github.com/Azure/kubeflow-labs/tree/master/6-tfjob
- Here are the instructions to run distributed training over a cluster using Kubeflow and Distributed TensorFlow* - https://github.com/Azure/kubeflow-labs/tree/master/7-distributed-tensorflow

Azure Machine Learning service:

- Call the submit method to run the training after setting up the cluster - https://docs.microsoft.com/ja-jp/azure/machine-learning/service/how-to-train-ml-models?#train-with-an-estimator

STEP BY STEP GUIDE TO RUN DEEP LEARNING INFERENCE

Instructions to run the inference is almost like the training. Use the Azure CPU instances along with containers for inference. Please note that inference containers are different from training containers.

Rule of thumb

Use this guide to split the cores for optimized inference with multiple streams even at lower batch sizes - https://www.intel.ai/accelerating-deep-learning-training-inference-system-level-optimizations/#gs.3d03w7

INFERENCE WITH DL FRAMEWORKS

Azure VM:

You can find the instructions to run inference in below link- https://docs.microsoft.com/en-us/azure/machine-learning/data-science-virtual-machine/linux-dsvm-walkthrough#deep-learning-tutorials-and-walkthroughs

ACI:

Deploying to ACI is commonly operated by using Azure Machine Learning service. Here are the instructions - https://docs.microsoft.com/ja-jp/azure/machine-learning/service/how-to-deploy-and-where

AKS Service:

Here are the instructions to run inference on Kubeflow on AKS.

https://www.kubeflow.org/docs/azure/azureendtoend/

Another way is using Azure Machine Learning service to deploy a model into AKS. The instructions are described next section.

Azure Machine Learning service:

Here is the complete reference to deploy models to AKS - https://docs.microsoft.com/ja-jp/azure/machine-learning/service/how-to-deploy-azure-kubernetes-service

As the other options, Azure Machine Learning service can deploy a model to below environments besides ACI, AKS.

- On-Premise web server
- Notebook VM web service
- Azure Machine Learning Compute
- Azure IoT Edge
- Azure Data Box Edge

Azure Batch:

Here are not dedicated instructions for deep learning inference but basically, we can follow and arrange it to fit to deep learning inference. - https://docs.microsoft.com/en-us/azure/batch/tutorial-parallel-python

INFERENCE WITH DL BOOST

As described above, there is no CLX instance on Azure yet so you cannot use DL boost currently.

INFERENCE WITH OPENVINO

OpenVINO is an optimization tool which converts the trained model from multiple frameworks and runs efficiently on Intel® Xeon® CPUs.

Here are the instructions to install the openVINO on Ubuntu/CentOS/Windows based VM image and use it for optimized inference - https://docs.openvinotoolkit.org/latest/index.html

Here are the instructions to build docker images to run on containers using ACI/AKS/Azure Machine Learning service –
https://docs.openvinotoolkit.org/latest/_docs_install_guides_installing_openvino_docker_linux.html

Here is the step-by-step instructions for openVINO model server to go for production with openVINO - https://www.intel.ai/openvino-model-server-boosts-ai-inference-operations/#gs.3cwfl4

INFERENCE WITH BIGDL

If you want to use Azure VM for BigDL you could follow this blog - https://azure.microsoft.com/en-au/blog/bigdl-spark-deep-learning-library-vm-now-available-on-microsoft-azure-marketplace/

If you want to use Azure HDInsight for BigDL, you could follow this blog - https://blogs.msdn.microsoft.com/azuredatalake/2017/03/17/how-to-use-bigdl-on-apache-spark-for-azure-hdinsight/

And If you want to use Azure Databricks for BigDL, you could follow these blogs - https://databricks.com/blog/2017/02/09/intels-bigdl-databricks.html

https://docs.azuredatabricks.net/applications/deep-learning/index.html

Actually, Microsoft* priors Azure Databricks as a Spark service so 3rd option is mostly recommended.

REFERENCES

[1] Avx-512 info: https://colfaxresearch.com/skl-avx512/

[2] Intel® optimized frameworks: https://software.intel.com/en-us/frameworks

[3] Intel® distribution of OpenVINO™ toolkit: https://docs.openvinotoolkit.org/

[4] Intel® Analytics zoo: https://github.com/intel-analytics/analytics-zoo

[5] Hands-on IDP and DAAL : https://software.intel.com/en-us/videos/get-your-hands-dirty-with-intel-distribution-for-python

[6] IDP benchmarks: https://software.intel.com/en-us/distribution-for-python/benchmarks

[9] Intel® DL Boost : https://www.intel.ai/increasing-ai-performance-intel-dlboost/#gs.3cxhiw