Intel® instances for AI workloads on AWS

A step by step instructions user guide

Devarampati Vinod, Ouchiyama Hiroshi, Yuping Zhao

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 AWS 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

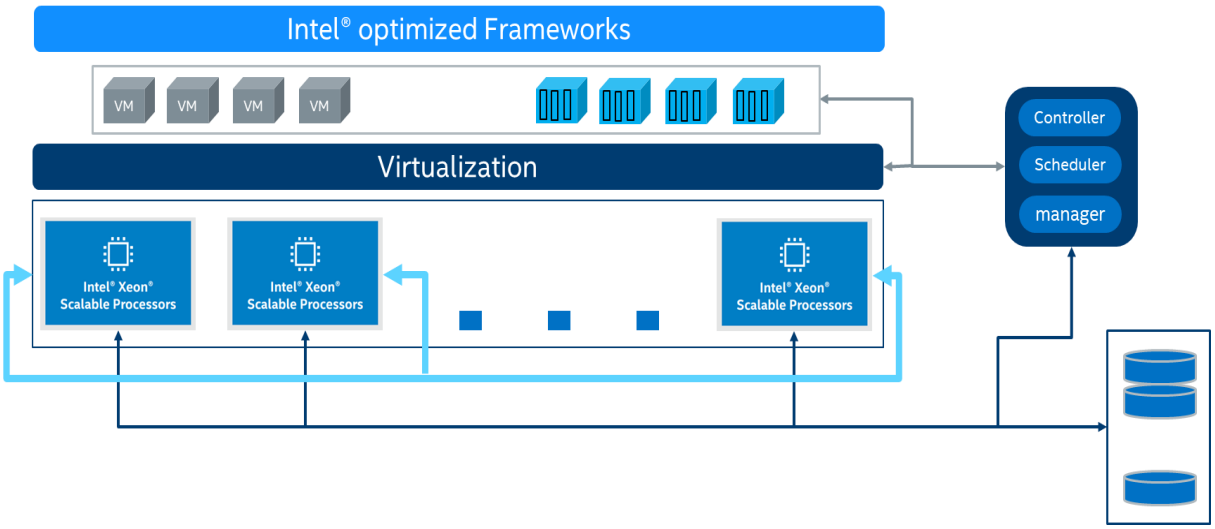
For DL inference and Single node training, there won't be any need of high network Band width communication between compute nodes. So 10Gig link will be good enough for most of the development and deployment. But for distributed computing, communication latency decides the scalability performance. Practical 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 most of the 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 AWS tools and services to accomplish the same -

- Compute nodes – EC2 instances for computing
- Storage – EBS or S3
- Management – ECS, EKS and Sagemaker
- Virtualization – AWS Nitro System
- Container orchestration – Kubernetes*

AWS CLOUD INSTANCES FOR AI APPLICATIONS

There are the many cloud instances available in AWS 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. Here are the various AWS cloud instances suitable for ML/DL applications -

COMPUTE INSTANCES

- C5 and C5n (Xeon® Platinum 8124M, 3GHz, 18 cores – Code name **Skylake**) except **c5.12xlarge, c5.24xlarge, c5.metal** (Xeon® platinum 8**2**75CL, 3GHz, 24 cores, Code name **Cascade lake)**

MEMORY OPTIMIZED

- R5 (Xeon® 8175M, 2.5 GHz, 24 cores - **Skylake**)

GENERAL PURPOSE INSTANCES

- T3 and M5 (Xeon® 8175, 3.1 GHz, 24 cores - **Skylake**)

RECOMMENDATION FOR VARIOUS ML/DL WORKLOADS

- ✓ **T3 instances** are suitable ML applications and low compute DL inference applications.
- ✓ **M5, C5 instances** are suitable for all kinds of Deep learning inference workloads
  - **c5.12xlarge, c5.24xlarge, c5.metal** are upgraded with 2nd Generation of Xeon® SP which will benefit with **DL Boost (VNNI)** for Inference. These can be used for Single node training too.
- ✓ **C5n instances** are suitable for Distributed Deep learning training due to high NW performance required for inter-node communication.
  - For training scalability over large cluster **c5n.18xlarge, c5n.metal** are preferred which comes with 192GB memory suffice for most of the topologies.
- ✓ **R5 instances** are suitable for memory intensive workloads which use 3D-CNN topologies with memory requirement more than 192GB.

Note:

- **z1d and i3en instances** also use the Xeon® scalable family which can be used for Deep learning workloads for memory or I/O intensive workloads.
- Out of all above instances, ***.metal instances** are preferred for large topologies as HVM based instances adds ~10% performance overhead.

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	AWS 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 T3 instances and C5/M5 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..	C5/M5 **>=8** vCPUs for FP32 For High throughput and low latency inference with **DL Boost (INT8),** C5.12xlarge, C5.24xlarge, C5.metal instances are recommended.
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..	C5/M5 **>=32** vCPUs for FP32 For High throughput and low latency inference with **DL Boost (INT8),** C5.12xlarge, C5.24xlarge, C5.metal are recommended.
Inference on memory and compute intensive NN topologies Ex: 3D-U-Net, 3D-GAN etc...	C5, R5 **>=32** vCPUs for inference
Single node Training of medium and Deep NN topologies	C5 **>=36** vCPUs
Distributed training on multiple nodes with all medium and deep NN topologies	C5n **>=36** vCPUs
Distributed training with further memory intensive like 3D CNN topologies	R5 **>= 48** vCPUs

Rule of thumb

- use **C5** instances are suitable for Deep learning inference and single node training
- use **C5n** instances are suitable for distributed training over a larger cluster due to high throughput communication network.

AWS TOOLS & SERVICES FOR DEEP LEARNING

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

AMIs

AWS is providing various base images with components required for Deep learning

AWS Deep Learning Base AMI - For developers who wants to install MKL-DNN optimized frameworks with version of their choice.

AWS Deep Learning AMI – an image with pre-installed frameworks. Supports multiple framework versions for TensorFlow*, MxNet*, PyTorch*, Chainer*, Theano*, CNTK*, Keras* etc.

Recommendation: use the image with tag **Intel® MKL-DNN**

Containers

To ease the job of installation, AWS is providing multiple container images with MKL-DNN optimized frameworks. Although TensorFlow* and PyTorch* are available, containers for other frameworks can be built and used through ECS.

Recommendation: use the below container images while using the EC2 instances which are integrated with MKL-DNN or generate MKL-DNN optimized framework containers

- tensorflow-training:<version>-**cpu**-py<version>-<os_version>
- tensorflow-inference:<version>-**cpu**-py<version>-<os_version>
- mxnet-training:<version>-**cpu**-py<version>-<os_version>
- mxnet-inference:<version>-**cpu**-py<version>-<os_version>

Service Engines

AWS 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.

Amazon SageMaker Ground Truth – An automated and manual mechanism to label and prepare the datasets for training.

EC2 – Elastic cloud compute – This service provides flexible compute capability in the form instances. These instances can be VM or bare metal over which deep learning applications can be built and deployed using any of the below services. Users with expertise in handling infrastructure and DL processing pipeline can use this service.

ECS - Elastic Container Service – This service is highly scalable and high-performance container orchestration helps in deploying containerized applications for production grade. ECS has the parameter-server based training through gRPC for TensorFlow*. It also supports

Horovod based training through MPI for all frameworks. Users who has the expertise with container orchestration can avail this service.

EKS - Elastic Kubernetes* Service – This service helps in deploying and management of a cluster for distributed training using Kubernetes*. EKS uses kubeflow for distributed TensorFlow* training. Users with little knowledge on container management can use this service to deploy and scale.

Sagemaker – This is a complete automated tool to build, train and deploy the deep learning applications over a scalable production cluster. Novice users can utilize this service.

AWS market place – Here intel® optimized containers and images are available to use on the CPU instances.

Recommendation: get MKL-DNN optimized containers and images and use for both training and inference on Intel® CPU instances

AWS AI services – Amazon launched below services to add deep learning intelligence into the customer applications

- Amazon Comprehend
- Amazon Comprehend Medical
- Amazon Forecast
- Amazon Lex
- Amazon Personalize
- Amazon Polly
- Amazon Rekognition
- Amazon Textract
- Amazon Transcribe
- Amazon Translate

Recommendation: use the Intel® Xeon® based instances to avail these services as these are already optimized with intel® MKL-DNN library.

Storage Service

AWS has multiple storage mechanisms. For single node applications EBS will be good enough. But S3 storage is right mechanism for scalable training and deployment over production clusters.

INSTRUCTIONS TO DEVELOP AND DEPLOY ML APPLICATIONS ON AWS

Development and deployment of machine learning applications is straight forward by choosing relevant EC2 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 AWS Intel® instances.

EC2:

- Launch EC2 instances mentioned in the previous section depending on the use case and computational requirement. https://aws.amazon.com/ec2/getting-started/#console
- 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.

AWS market place has multiple intel® DAAL images for optimized inference - https://aws.amazon.com/marketplace/search/results?x=0&y=0&searchTerms=intel+DAAL

INSTRUCTIONS TO DEVELOP AND DEPLOY DL APPLICATIONS ON AWS

After choosing the right compute instance, communication network, storage and optimized software, AWS services are required to automate and scale on a production cluster. Please make sure you have privileged account to AWS as some services needs admin account.

In this section we discuss on the procedure to develop and productize the deep learning training and inference applications using AWS IA instances with multiple AWS 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 AWS, use the Intel® AWS 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://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-mkl.html

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

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

EC2 Service setup:

Launching Deep learning training on EC2 service is straight forward after setup

- Follow the prerequisite instructions https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-ec2-setup.html
- Launch the desired Intel® hosted instance with Deep learning Base Image or Deep learning AMI containing intel® optimized frame works
- Here is the link to launch DL AMI https://aws.amazon.com/getting-started/tutorials/get-started-dlami/

ECS Service setup:

Follow the prerequisites before launch https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-ecs-setup-prerequisites.html

Here are the instructions to create ECS cluster with DL AMI https://aws.amazon.com/getting-started/tutorials/get-started-dlami/

Launching Deep learning training on ECS cluster through command line requires below steps:

- Create an Amazon ECS cluster in the Region that contains the key pair and security group that you created previously.
  aws ecs create-cluster --cluster-name ecs-ec2-training-inference --region <region_name>
- Launch one or more deep learning EC2 instances with Amazon Linux or Amazon Linux 2 ECS-optimized AMIs
- Create a file named my_script.txt with the following contents. Reference the same cluster name that you created in the previous step

```
#!/bin/bash
echo ECS_CLUSTER=ecs-ec2-training-inference >> /etc/ecs/ecs.config
```

- Create a file named my_mapping.txt with the following content, which changes the size of the root volume after the instance is created (optional)

```
[
  {
    "DeviceName": "/dev/xvda",
    "Ebs": {
      "VolumeSize": 100
    }
```

- Launch an Amazon EC2 instance with the Amazon ECS-optimized AMI and attach it to the cluster. Use your security group ID and key pair name that you created and replace them in the following command.

```
aws ec2 run-instances --image-id ami-0dfdeb4b6d47a87a2 \
    --count 1 \
    --instance-type <instance_name> \
    --key-name key-pair-1234 \
    --security-group-ids sg-abcd1234 \
    --iam-instance-profile Name="ecsInstanceRole" \
    --user-data file://my_script.txt \
    --block-device-mapping file://my_mapping.txt \
```

```
--region <region_name>
```

EKS Service setup:

Here are the prerequisites for EKS https://docs.aws.amazon.com/ja_jp/dlami/latest/devguide/deep-learning-containers-eks-setup.html

Here are the steps to setup the EKS service

- Build and keep the container images ready with desired framework
- Finish the required security configuration
- Gateway node, a controller for the cluster is needed which can be an EC2 instance with DL Base AMI is loaded and launched.
- Install or upgrade AWS CLI
  ```
  sudo pip install --upgrade awscli
  ```
- Install eksctl by running the following commands
- ```
  curl --silent \
  --location
  "https://github.com/weaveworks/eksctl/releases/download/latest_release/eksctl_$(uname -s)_amd64.tar.gz" \
  | tar xz -C /tmp
  $ sudo mv /tmp/eksctl /usr/local/bin
  ```
- Install kubectl by running the following commands.
  ```
  curl -o kubectl https://amazon-eks.s3-us-west-2.amazonaws.com/1.11.5/2018-12-06/bin/linux/amd64/kubectl
  chmod +x ./kubectl
  mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl && export PATH=$HOME/bin:$PATH
  ```
- Install aws-iam-authenticator by running the following commands
  ```
  curl -o aws-iam-authenticator https://amazon-eks.s3-us-west-2.amazonaws.com/1.11.5/2018-12-06/bin/linux/amd64/aws-iam-authenticator
  chmod +x aws-iam-authenticator
  cp ./aws-iam-authenticator $HOME/bin/aws-iam-authenticator && export PATH=$HOME/bin:$PATH
  ```
- Run aws configure for the IAM user from the Security Configuration section.
- Install ksonnet
  ```
  export KS_VER=0.13.1
  export KS_PKG=ks_${KS_VER}_linux_amd64
  wget -O /tmp/${KS_PKG}.tar.gz
  https://github.com/ksonnet/ksonnet/releases/download/v${KS_VER}/${KS_PKG}.tar.gz
  mkdir -p ${HOME}/bin
  tar -xvf /tmp/$KS_PKG.tar.gz -C ${HOME}/bin
  sudo mv ${HOME}/bin/$KS_PKG/ks /usr/local/bin
  ```
- Create EKS cluster
  ```
  eksctl create cluster <cluster-name> \
                --version 1.11 \
  ```
  ```

```
--nodes 3 \
--node-type=<cpu instance type> \
--timeout=100m \
--ssh-access \
--ssh-public-key <key_pair_name> \
--region <region_name> \
--auto-kubeconfig
```

- run a kubectl command on the cluster to check its status
  `kubectl get nodes -o wide`
- Verify that the cluster is active
  `aws eks --region <region_name> describe-cluster --name <cluster-name> --query cluster.status`
- Now cluster setup is done. Delete the cluster once the training is over.

Sagemaker setup:

Amazon Sagemaker is complete automated tool to train and deploy deep learning models. Initial setup of the training cluster is automated through set of APIs filled with framework, model code, and dataset relevant information.

- Sagemaker automates through containers for many frameworks and we recommend using CPU containers.
- Follow the setup instructions https://docs.aws.amazon.com/sagemaker/latest/dg/gs-set-up.html
- Follow the below setup steps to initiate the service
  - https://docs.aws.amazon.com/sagemaker/latest/dg/gs-config-permissions.html
  - https://docs.aws.amazon.com/sagemaker/latest/dg/gs-setup-working-env.html
  - https://docs.aws.amazon.com/sagemaker/latest/dg/ex1-prepare.html
- While creating the model estimator use the CPU instance type as shown below

```
tf_estimator = TensorFlow(entry_point='tf-train.py', role='SageMakerRole',
                          train_instance_count=1, train_instance_type=<cpu_instance_type>,
                          framework_version='1.12', py_version='py3')
```

- Although Sagemaker has the capability of TensorFlow* training with both parameter server and Horovod, we recommend using Horovod as given in the link - https://sagemaker.readthedocs.io/en/stable/using_tf.html#training-with-horovod

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

EC2 Service:

- Here is a step-by-step guide to launch the training on AWS EC2 instance - https://aws.amazon.com/getting-started/tutorials/train-deep-learning-model-aws-ec2-containers/
    - In step 3d, choose the Skylake or cascade lake instance mentioned in previous sections depending on the type of topology used
    - For instances with multiple sockets or *.metal 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

ECS Service:

- Here is the step-by-step guide to run the training on ECS cluster https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-ecs-tutorials-training.html

EKS Service:

- Here are the instructions to run the training over EKS service - https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-eks-tutorials-cpu-training.html
- Here are the instructions to run distributed training over a cluster using Kubeflow and Horovod - https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-eks-tutorials-distributed-gpu-training.html
    - In the previous section, setup is shown with CPU instances. While creating the training script, same CPU docker image details and AWS CPU instances information needs to be added.

Sagemaker service:

- Call the fit method to run the training after setting up the cluster https://sagemaker.readthedocs.io/en/stable/using_tf.html#call-the-fit-method
- Here are is a python notebook tutorial https://github.com/awslabs/amazon-sagemaker-examples/blob/master/sagemaker-python-sdk/tensorflow_iris_dnn_classifier_using_estimators/tensorflow_iris_dnn_classifier_using_estimators.ipynb

STEP BY STEP GUIDE TO RUN DEEP LEARNING INFERENCE

Instructions to run the inference is almost like the training. Use the AWS 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

EC2 Service:

Here are the instructions to run inference - https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-ec2-tutorials-inference.html

ECS Service:

Here are the instructions to run inference - https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-ecs-tutorials-inference.html

EKS Service:

Here are the instructions to run inference https://docs.aws.amazon.com/dlami/latest/devguide/deep-learning-containers-eks-tutorials-cpu-inference.html

Sagemaker:

Here is the complete reference to deploy models on multiple frameworks - https://sagemaker.readthedocs.io/en/stable/index.html

INFERENCE WITH DL BOOST

Intel® DL boost added in 2nd generation of Xeon® scalable platform has high throughput and low latency INT8 inference capability with VNNI instructions [9]. Here is the complete guide to convert the trained model from FP32 to INT8 and use for high performance inference on the AWS cluster with Second generation of Xeon® servers - https://www.intel.ai/tensorflow-containers-optimized-intel/#gs.2ass5x

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 AMI 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 ECS/EKS/Sagemaker services – https://docs.openvinotoolkit.org/2018_R5/_docs_install_guides_installing_openvino_docker.html

Here is calibration tool to run the inference with DL Boost - https://docs.openvinotoolkit.org/latest/_inference_engine_tools_calibration_tool_README.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

Load the AMI for BigDL from AWS marketplace on to C5 instances https://aws.amazon.com/marketplace/pp/B075KZ8NDY?qid=1568619952467&sr=0-1&ref_=srh_res_product_title

Here are the step-by-step instructions to run BigDL on AWS - https://github.com/intel-analytics/BigDL/blob/master/docs/docs/ProgrammingGuide/run-on-ec2.md

Here is the AWS blog to launch BigDL inferencing on the EC2 instances - https://aws.amazon.com/blogs/machine-learning/running-bigdl-deep-learning-for-apache-spark-on-aws/

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

[7] AWS DL guide: https://d1.awsstatic.com/whitepapers/Deep_Learning_on_AWS.pdf?did=wp_card&trk=wp_card

[8] AWS ECS guide: https://docs.aws.amazon.com/AmazonECS/latest/developerguide/ecs-dg.pdf

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