White Paper

AI Applications and Analytics
Intel® Xeon® Processors

Accelerating AI Applications on Alibaba Cloud with Analytics Zoo and Bfloat16

This paper describes how to use Analytics Zoo and Brain Floating Point 16-bit (bfloat16) to improve the performance of artificial intelligence (AI) applications running on seventh-generation Alibaba Cloud Elastic Compute Service (ECS) instances. Seventh-generation Alibaba Cloud ECS instances are powered by 3rd Generation Intel Xeon Scalable processors, and they provide bfloat16 support.

Background information

- Seventh-generation Alibaba Cloud high-frequency ECS instances use the third generation of X-Dragon Architecture and 3rd Generation Intel Xeon Scalable processors. Compared with the previous generation, seventh-generation Alibaba Cloud high-frequency ECS instances with high clock speeds can improve compute performance by up to 260 percent.[1] You can also use the advanced pipeline feature of Analytics Zoo on Alibaba Cloud ECS instances. For example, you can use Intel-optimized deep learning (DL) models, such as TensorFlow and PyTorch, to develop DL applications.

  3rd Generation Intel Xeon Scalable processors deliver industry-leading and workload-optimized platforms by using enhanced Intel® Deep Learning Boost (Intel DL Boost), which is a built-in artificial intelligence (AI) acceleration feature. Enhanced Intel DL Boost provides the first x86 support for bfloat16 in the industry, which enhances AI inference and training performance.

  3rd Generation Intel Xeon Scalable processors can process complex AI workloads. By using enhanced Intel DL Boost, 3rd Generation Intel Xeon Scalable processors can deliver up to 1.93 times the AI training performance,[2] up to 1.87 times the AI inference performance for image classification,[3] up to 1.7 times the AI training performance for natural language processing (NLP),[4] and up to 1.9 times the AI inference performance for NLP, compared with previous-generation processors.[5] Many AI-training workloads from industry sectors such as healthcare, financial, and retail can benefit from the bfloat16 support provided by these processors.
- Analytics Zoo is an open source unified analytics and AI platform. It can seamlessly scale AI models such as TensorFlow, Keras, and PyTorch to distributed big data platforms such as Apache Spark, Apache Flink, and Ray. As seen in Figure 1, Analytics Zoo provides the following features:
  - End-to-end pipelines for applying AI models such as TensorFlow, PyTorch, and the OpenVINO™ toolkit to big data platforms. For example, developers can embed TensorFlow or PyTorch code in Spark code for distributed training and inference. Developers can also use native DL models such as TensorFlow, Keras, PyTorch, and BigDL in Spark machine learning (ML) pipelines.
  - High-level ML workflows, such as cluster serving and scalable AutoML, for automated ML tasks. Cluster serving is an automatic and distributed inference solution for models such as TensorFlow, PyTorch, and the OpenVINO toolkit. Scalable AutoML is used for time-series predictions.
  - Built-in models for recommendation, time series, computer vision, and NLP applications.
- Bfloat16 is a numeric format that is widely used in neural networks.
- ResNet-50 is a residual network that is 50-layers deep. This neural network is widely used for image-classification tasks.

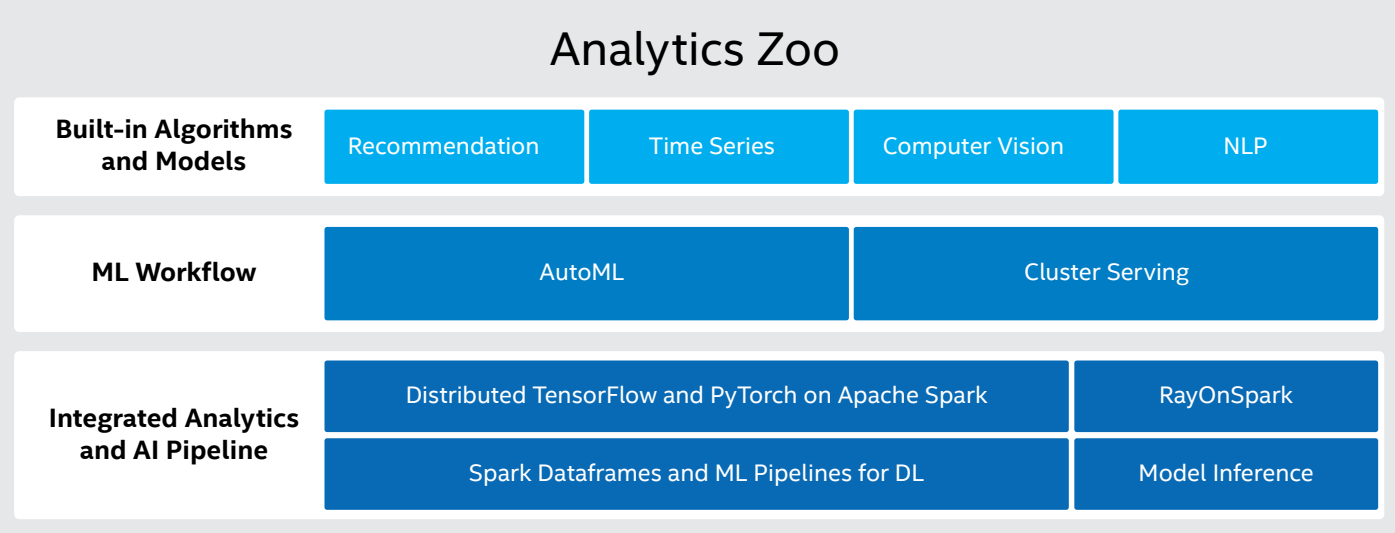

Figure 1. The Analytics Zoo open source analytics and AI platform incorporates numerous features to accelerate AI applications on Alibaba Cloud

Procedure

To use Analytics Zoo and bfloat16 to accelerate AI applications (such as in the ResNet-50 training task example provided later in this document) on an Alibaba Cloud ECS instance, perform the following steps:

Step 1: Create an ECS instance with a high clock speed

Step 2: Prepare an Analytics Zoo environment that supports bfloat16

Step 3: Train the ResNet-50 model on the ECS instance by using bfloat16 to improve performance

Step 1: Create an ECS instance with a high clock speed

To create an ECS instance, perform the following operations:

1. Navigate to the ECS buy page.
2. Create an instance that belongs to the *hfc7* instance family. For more information, see "Create an instance by using the wizard."

   When you configure the instance-type parameter, you can select an instance type only from the *hfc7* or *hfg7* instance family in this scenario. For more information about instance types, see "Instance families with high clock speeds."
3. On the **Instances** page, find the new instance, and then click the **instance ID**. On the page that appears, view and confirm the instance type.

Step 2: Prepare an Analytics Zoo environment that supports bfloat16

Analytics Zoo provides a pre-built Docker image that supports bfloat16. You can obtain the Docker image on the ECS instance by using Method 1, below. You can also use the Analytics Zoo nightly build to support bfloat16. For more information, see Method 2, below. For more information about code description, see "Code sample: How Analytics Zoo uses bfloat16 to accelerate the training of deep learning models."

- Method 1: Obtain the pre-built Docker image of Analytics Zoo on the ECS instance:
  1. Connect to the ECS instance. For more information, see "Connect to an ECS instance."
  2. Run the following commands to install and start Docker:

     ```
     yum install docker-io -y
     systemctl start docker
     ```
  3. Run the following command to obtain the Analytics Zoo Docker image that supports bfloat16:

     ```
     docker pull intelanalytics/analytics-zoo:0.8.1-bigdl_0.10.0-spark_2.4.3-bf16
     ```
  4. Run the following command to run the Docker container:

     ```
     docker run -itd --name az1 --net=host  --privileged
     intelanalytics/analytics-zoo:0.8.1-bigdl_0.10.0-spark_2.4.3-bf16
     ```
  5. Run the following command to enter the container:

     ```
     docker exec -it az1 bash
     ```
- Method 2: Use the nightly Analytics Zoo build to support bfloat16:
  1. Connect to the ECS instance. For more information, see "Connect to an ECS instance."
  2. Run the following commands to download and decompress the latest version of the nightly build pre-build package of Analytics Zoo:

     ```
     wget
     https://oss.sonatype.org/content/repositories/snapshots/com/intel/analytics/zoo/
     analytics-zoo-bigdl_0.11.1-spark_2.4.3/0.9.0-SNAPSHOT/analytics-zoo-bigdl_0.11.1-spark_2.4.3-
     0.9.0-20201026.210040-51-dist-all.zip
     unzip analytics-zoo-bigdl_0.11.1-spark_2.4.3-0.9.0-{datetime}-dist-all.zip -d analytics-zoo
     ```
  3. Run the following command to install Git:

     ```
     yum -y install git
     ```
  4. Run the following commands to download the TensorFlow source code:

     ```
     git clone https://github.com/Intel-tensorflow/tensorflow.git
     git checkout v1.15.0up1
     ```
  5. Run the following commands to compile TensorFlow:

     ```
     bazel build --cxxopt=-D_GLIBCXX_USE_CXX11_ABI=0 --copt=-O3 --copt=-Wformat
     --copt=-Wformat-security --copt=-fstack-protector --copt=-fPIC
     --copt=-fpic --linkopt=-znoexecstack --linkopt=-zrelro
     --linkopt=-znow --linkopt=-fstack-protector --config=mkl --define
     build_with_mkl_dnn_v1_only=true --copt=-DENABLE_INTEL_MKL_BFLOAT16
     --copt=-march=native
     //tensorflow/tools/lib_package:libtensorflow_jni.tar.gz
     //tensorflow/java:libtensorflow.jar
     //tensorflow/java:libtensorflow-src.jar
     //tensorflow/tools/lib_package:libtensorflow_proto.zip
     ```
  6. Run the following commands to collect the library files that are required by Analytics Zoo:

     ```
     cd bazel-bin/tensorflow/tools/lib_package
     mkdir linux-x86_64
     tar -xzvf libtensorflow_jni.tar.gz -C linux_x86-64
     rm libtensorflow_framework.so
     rm libtensorflow_framework.so.1
     ```

```
mv libtensorflow_framework.so.1.15.0 libtensorflow_framework-zoo.so
cp ../../../../_solib_k8/_U@mkl_Ulinux_S_S_Cmkl_Ulibs_Ulinux___Uexternal_Smkl_Ulinux_Slib/*
./
```

7. Run the following commands to update the Analytics Zoo JAR files:

```
cd ~/analytics-zoo/lib/
cp ~/tensorflow/bazel-bin/tensorflow/tools/lib_package/linux-x86_64 ./
jar -ufanalytics-zoo-bigdl_0.11.1-spark_2.4.3-0.9.0-SNAPSHOT-jar-with-dependencies.jar linux-
x86_64/*
```

Step 3: Train the ResNet-50 model on the ECS instance and use bfloat16 to improve performance

1. Run the following command to enter the Analytics Zoo Docker container:

```
docker exec -it az1 bash
```

2. Run the following commands to modify the *spark-defaults.conf* file in the /opt/work/spark-2.4.3/conf/ directory to configure Spark:

```
spark.authenticate=false
spark.ui.killEnabled=true
spark.eventLog.enabled=true
spark.history.ui.port=18080
spark.eventLog.dir=file:///var/log/spark/spark-events
spark.history.fs.logDirectory=file:///var/log/spark/spark-events
spark.shuffle.service.port=7337
spark.master=spark://$(hostname):7077
```

3. Run the following commands to start the Spark master:

```
cd /opt/work/spark-2.4.3
./sbin/start-master.sh
```

4. Create the following script in the /opt/work/spark-2.4.3/bin directory to run the *numactl* command to start eight Spark workers and bind each worker to 12 vCPUs:

```
numactl -C 0-11 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 12-23 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 24-35 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 36-47 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 48-59 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 60-71 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 72-83 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
numactl -C 84-95 ./spark-class org.apache.spark.deploy.worker.Worker spark://$(hostname):7077 &
```

5. Run the following command to check whether the eight Spark workers are started. If `8` is returned, the eight Spark workers are started.

```
jps | grep Worker | wc -l
```

6. Run the following commands to download the sample ResNet-50 code from GitHub:

```
git clone https://github.com/yangw1234/models-1.git
git checkout branch-1.6.1-zoo
```

7. Run the *run.sh* script in the models-1/models/image_recognition/tensorflow/resnet50v1_5/training/mlperf_resnet directory. Add the *--use_bfloat16* option to enable bfloat16-based training. If you do not add this option, *FP32* is used for training by default.

```
# Register the model as a source root
export PYTHONPATH="$(pwd):${PYTHONPATH}"
export KMP_BLOCKTIME=0
# 8 instances
```

```
export OMP_NUM_THREADS=6
export KMP_AFFINITY=granularity=fine,compact,1,0
export KMP_SETTINGS=1
export ANALYTICS_ZOO_HOME=/opt/work/analytics-zoo/dist
export SPARK_HOME=/opt/work/spark-2.4.3
bash $ANALYTICS_ZOO_HOME/bin/spark-submit-python-with-zoo.sh --master
spark://$(hostname):7077 \
--executor-cores 1 --total-executor-cores 8 --driver-memory 20g --executor-memory 18g \
--conf spark.network.timeout=10000000 --conf spark.executor.heartbeatInterval=100000 \
imagenet_main.py 1 --model_dir ./logs --batch_size 128 --version 1 \
--resnet_size 50 --train_epochs 90 --data_dir /opt/ILSVRC2012/ --use_bfloat16
```

The following table shows the training results:[6]

ResNet-50 model training	FP32	Bfloat16	Performance improvement by using bfloat16
Throughput (images/second)	119.636	212.315	1.775

Summary

This paper uses ResNet-50 training as an example to demonstrate how users can use Analytics Zoo and bfloat16 to accelerate AI applications running on Alibaba Cloud with 3rd Generation Intel Xeon Scalable processors. 3rd Generation Intel Xeon Scalable processors provide bfloat16 support on x86 platforms for the first time in the industry. In addition, the seventh-generation Alibaba Cloud high-frequency ECS instances are some of the first cloud platforms to support bfloat16 AI acceleration. Analytics Zoo, the unified data-analytics and AI platform, is able to unleash bfloat16 acceleration on Alibaba Cloud. Given ResNet-50 as example, Analytics Zoo using bfloat16 can achieve up to 1.775 times the performance compared to using FP32 in the given Alibaba Cloud ECS environment. For more information, visit the Analytics Zoo GitHub site or the Analytics Zoo documentation website.

Sample code: How Analytics Zoo uses bfloat16 to accelerate the training of DL models

The following sample code snippets show how Analytics Zoo uses bfloat16 to accelerate the training of DL models such as ResNet-50. These code snippets are already contained in the Docker image of Analytics Zoo, and they are used for reference only. You do not need to compile the code.

1. The following code snippet is used to convert input images into the bfloat16 format:

```
if use_bfloat16 == True:
dtype = tf.bfloat16
features = tf.cast(features, dtype)
```

2. The following code snippet is used to compile *custom_dtype_getter*:

```
DEFAULT_DTYPE = tf.float32
CASTABLE_TYPES = (tf.float16,tf.bfloat16)
def _custom_dtype_getter(getter, name, shape=None, dtype=DEFAULT_DTYPE,
      *args, **kwargs):
    if dtype in CASTABLE_TYPES:
      var = getter(name, shape, tf.float32, *args, **kwargs)
      return tf.cast(var, dtype=dtype, name=name + '_cast')
    else:
      return getter(name, shape, dtype, *args, **kwargs)
```

3. The following code snippet is used to create *variable_scope* and to build a model in the scope:

```
def _model_variable_scope():
    return tf.compat.v1.variable_scope('resnet_model',
                       custom_getter=_custom_dtype_getter)


with _model_variable_scope():
   logits = _resnet_50_model(features)
```

4. The following code snippet is used to convert *logits* into the *float32* format for the loss calculation. This conversion can help ensure numerical stability.

```
logits = tf.cast(logits, tf.float32)
```

The TFPark module of Analytics Zoo is used to perform distributed training.
For more information, see Analytics Zoo for distributed TensorFlow.

[1] Based on Alibaba Cloud testing. Source: Alibaba Cloud. "Detailed explanation of the performance parameters of the seventh-generation high-frequency instance of Alibaba Cloud Server ECS." June 2020. https://developer.aliyun.com/article/765901.

[2] Up to 1.93x higher AI training performance with a 3rd Generation Intel Xeon Scalable processor supporting Intel DL Boost with BF16 vs. a prior-generation processor with ResNet-50 throughput for image classification. **New configuration:** 1 node, 4 x 3rd Generation Intel Xeon Platinum 8380H processor (pre-production 28 cores, 250 W) with 384 GB total memory (24 x 16 GB, 3,200 GHz), 800 GB Intel SSD drive, ResNet-50 v1.5, ucode 0x700001b, Intel Hyper-Threading Technology (Intel HT Technology) on, Intel Turbo Boost Technology on, and running Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic. Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642769358b388d8f615ded9c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, ImageNet dataset, oneDNN 1.4, BF16, BS=512, tested by Intel on 5/18/2020. **Baseline:** 1 node, 4 x Intel Xeon Platinum 8280 processor with 768 GB total memory (24 x 32 GB, 2,933 GHz), 800 GB Intel SSD, ucode 0x4002f00, Intel HT Technology on, Intel Turbo Boost Technology on, with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, ResNet-50 v1.5. Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388d8f615ded0c213f10c99a, Model Zoo: https://github.com/intelai/models -b v1.6.1, ImageNet dataset, oneDNN 1.4, FP32, BS=512, tested by Intel on 5/18/2020.

[3] Up to 1.87x higher AI inference performance with a 3rd Generation Intel Xeon Scalable processors supporting Intel DL Boost with BF16 compared to prior-generation processors using FP32 on ResNet-50 throughput for image classification. **New configuration**: 1 node, 4 x 3rd Generation Intel Xeon Platinum 8380H processor (pre-production, 28 cores, 250 W) with 384 GB total memory (24 x 16 GB, 3,200 GHz), 800 GB Intel SSD, ucode 0x700001b, Intel HT Technology on, Intel Turbo Boost Technology on with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, ResNet-50 v1.5. Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388e8r615ded0c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, ImageNet dataset, oneDNN 1.4, BF16, BS=56, 5 instances, 28 cores/instance, tested by Intel on 5/18/2020. **Baseline:** 1 node, 4 x Intel Xeon Platinum 8280 processors with 768 GB total memory (24 x 32 GB, 2,933 GHz), 800 GB Intel SSD, ucode 0x4002f00, Intel HT Technology on, Intel Turbo Boost Technology on, with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, ResNet-50 v1.5. Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388d8f615ded0c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, ImageNet dataset, oneDNN 1.5, FP32, BS=56, 4 instances, 28 cores/instance, tested by Intel on 5/18/2020.

[4] Up to 1.7x more AI training performance with a 3rd Generation Intel Xeon Scalable processor supporting Intel DL Boost with BF16 vs. a prior-generation processor on BERT throughput for natural language processing. **New configuration**: 1 node, 4 x 3rd Generation Intel Xeon Platinum 8380H processor (pre-production, 28 cores, 250 W) with 384 GB total memory (24 x 16 GB, 3,200 GHz), 800 GB Intel SSD, ucode 0x700001b, Intel HT Technology on, Intel Turbo Boost Technology on with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, BERT-Large (QA). Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388e8r615ded0c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, Squad 1.1 dataset, oneDNN 1.4, BF16, BS=12, tested by Intel on 5/18/2020. **Baseline:** 1 node, 4 x Intel Xeon Platinum 8280 processors with 768 GB total memory (24 x 32 GB, 2,933 GHz), 800 GB Intel SSD, ucode 0x4002f00, Intel HT Technology on, Intel Turbo Boost Technology on, with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, BERT-Large (QA). Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388d8f615ded0c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, Squad 1.1 dataset, oneDNN 1.5, FP32, BS=12, tested by Intel on 5/18/2020.

[5] Up to 1.9x higher AI inference performance with a 3rd Generation Intel Xeon Scalable processor supporting Intel DL Boost with BF16 vs. a prior-generation processor with FP32 for BERT throughput for natural language processing. **New configuration:** 1 node, 4 x 3rd Generation Intel Xeon Platinum 8380H processor (pre-production, 28 cores, 250 W) with 384 GB total memory (24 x 16 GB, 3,200 GHz), 800 GB Intel SSD, ucode 0x700001b, Intel HT Technology on, Intel Turbo Boost Technology on with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, BERT-Large (QA). Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388e8r615ded0c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, Squad 1.1 dataset, oneDNN 1.4, BF16, BS=32, 4 instances, 28 cores/instance, tested by Intel on 5/18/2020. **Baseline:** 1 node, 4 x Intel Xeon Platinum 8280 processors with 768 GB total memory (24 x 32 GB, 2,933 GHz), 800 GB Intel SSD, ucode 0x4002f00, Intel HT Technology on, Intel Turbo Boost Technology on, with Ubuntu 20.04 LTS, Linux 5.4.0-26,28,29-generic, BERT-Large (QA). Throughput: https://github.com/Intel-tensorflow/tensorflow -b bf16/base, commit#828738642760358b388d8f615ded0c213f10c99a, Model Zoo: https://github.com/IntelAI/models -b v1.6.1, Squad 1.1 dataset, oneDNN 1.5, FP32, BS=32, 4 instances, 28 cores/instance, tested by Intel on 5/18/2020.

[6] 1.775x performance improvement based on Alibaba Cloud testing using FP32 and bfloat16 on an Alibaba Cloud ECS ecx.hfc7.24xlarge instance. For details on how to get the specified cloud instance, see "Step 1: Create an ECS instance with a high clock speed."

Printed in USA 0321/RG/PRW/PDF Please Recycle 346506-001US