User Guide TensorFlow v1.15 - ZenDNN v3.0

April 2021

Document Information

Software Version: 3.0
Document Version: 3.0
Last Updated: 21st April 2021

Contents

1 Installing ZenDNN

1.1 Binary Release

Complete the following steps to install the ZenDNN binary release:

1. Copy the zipped release package to the local system being used. The name of the release package will be similar to *TF_v1.15_ZenDNN_v3.0.zip*.
2. Execute the following commands:
   a. unzip *TF_v1.15_ZenDNN_v3.0.zip*
   b. cd *TF_v1.15_ZenDNN_release*/ZenDNN*
   c. Ensure that the environment variables **ZENDNN_BLIS_PATH** and **ZENDNN_AOCC_COMP_PATH** are set.
   d. source scripts/zendnn_aocc_env_setup.sh
      Note:
      - Ensure that it is sourced only from the ZenDNN folder.
      - If there is any conda environment named **tf-1.15-zendnn-rel-env** already present, delete the conda environment **tf-1.15-zendnn-rel-env** (using command conda remove --name tf-1.15-zendnn-rel-env --all) before running scripts/zendnn_aocc_env_setup.sh.
   e. source scripts/TF_v1.15_ZenDNN_setup_release.sh
      This installs the TensorFlow wheel package provided in the zip file and downloads the TensorFlow CNN benchmarks.
   f. To run the benchmarks with different CNN models at the TensorFlow level, refer to the TensorFlow CNN Benchmarks section.

1.2 Docker Release

ZenDNN release is also available as a docker image, which is supported on many Linux® flavors. The name of the docker image will be similar to *TF_v1.15_ZenDNN_v3.0_docker.tar.gz*.

Prerequisites

Install docker in the Linux® server being used.

Note: If docker has already been installed, please skip this step.

For more information, refer to Docker documentation.

Complete the following steps to install the ZenDNN Docker image:

1. Invoke sudo: sudo su
2. Check if docker is installed on the machine:
   docker --version
   >>Docker version 19.03.12, build 48a66213fe
3. Copy ZenDNN docker image to the Linux® system being used.
4. Load the Docker image:
   docker load -i TF_v1.15_ZenDNN_v3.0_docker.tar.gz
5. Run Docker in privileged mode:
   docker run --privileged -itd --name zendnn-tf-rel tf_v1.15_zendnn:3.0

   >>2c8af0175b8acc35ab89d8b2111b72310f48d5c5b01d0811fb59ba6dc51ff961

Note: This UUID will change for each user/run. Use the above generated UUID in the step 6.

6. Use the UUID (long identifier) generated from step 5:
   docker exec -it 2c8af0175b8acc35ab89d8b2111b72310f48d5c5b01d0811fb59ba6dc51ff961 bash
   The command prompt changes to *(tf-1.15-zendnn-rel-env) root@2c8af0175b8a:/*.
7. To run the benchmarks with different CNN models at the TensorFlow level, refer to the In the current release, ZenDNN is integrated with TensorFlow and ONNXRT. The dotted component in the diagram (PyTorch) is not supported in this release but is planned for a future release.
8. TensorFlow CNN Benchmarks section.

1.2.1 Instructions for New Session

If the terminal where the docker image was installed is closed, please complete the following steps upon opening a new terminal to continue:

1. Invoke sudo: sudo su
2. Check the previous Docker container for this image tf_v1.15_zendnn:3.0:
   docker ps -a

   #CONTAINER ID IMAGE COMMAND CREATED STATUS

   #2c8af0175b8a tf_v1.15_zendnn:3.0 "/bin/bash" 23 minutes ago Exited (255) 3 minutes ago

3. Check the **STATUS** and do one of the following:
   - If it is **EXITED**, start the docker container: docker start <container-id>
   - If status is not **EXITED** or **CREATED**, go to step 6.
   - If no containers for ZenDNN are found, start from step 5.

2 Directory Structure

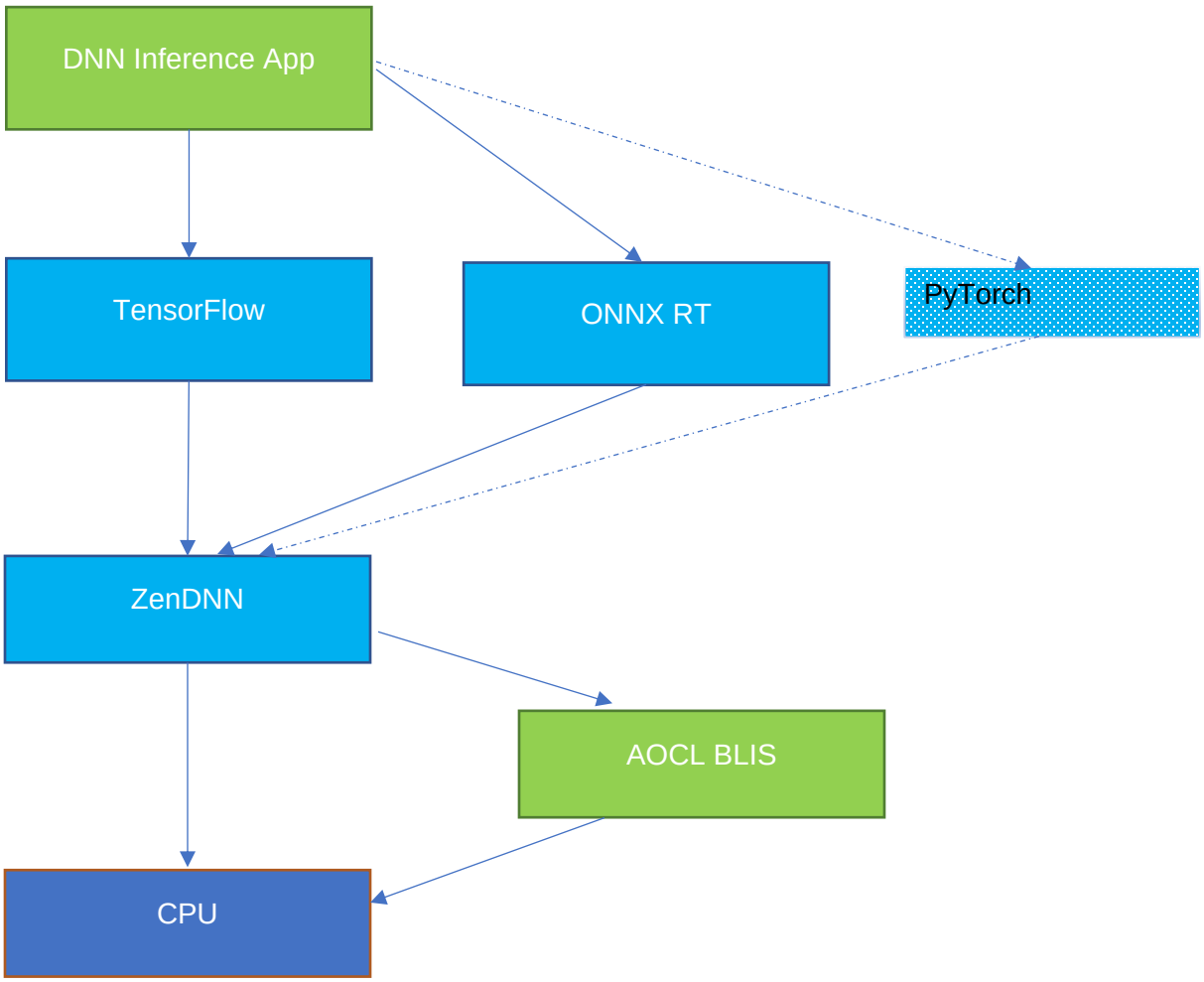
The release folder consists of a TensorFlow *whl* file and ZenDNN binary package.

ZenDNN consists of the following directories:

- _**out/lib:** contains the *ZenDNN.so* file.
- **scripts:** contains scripts to set up the environment and run test cases.

3 High Level Overview - Block Diagram

The following is a high-level block diagram for the ZenDNN library, which uses the AOCL BLIS library internally.



In the current release, ZenDNN is integrated with TensorFlow and ONNXRT. The dotted component in the diagram (PyTorch) is not supported in this release but is planned for a future release.

4 TensorFlow CNN Benchmarks

The benchmark scripts provide performance benchmarking at the TensorFlow level, printing latency and throughput results for AlexNet, GoogLeNet, Inception-v3, Inception-v4, ResNet50, ResNet152, VGG16, and VGG19 models.

For latency using *tf_cnn_benchmarks.py*, execute the following commands:

1. cd TF_v1.15_ZenDNN_release*/ZenDNN
2. source scripts/zendnn_aocc_env_setup.sh
3. source scripts/tf_cnn_benchmarks_latency.sh

For throughput using *tf_cnn_benchmarks.py*, execute the following commands:

1. cd TF_v1.15_ZenDNN_release*/ZenDNN
2. source scripts/zendnn_aocc_env_setup.sh
3. source scripts/tf_cnn_benchmarks_throughput.sh

Note:

- For optimal settings, refer to the Tuning Guidelines section. Current setting refers to 24C, 2P, SMT=ON configuration.
- If the below warning is seen on the terminal, it can be ignored. During environment setup, there is an optional script to gather information about hardware, OS, Kernel, and BIOS and it requires a few utilities (lscpu, lstopo-no-graphics, dmidecode etc.) to be present. If these utilities are not present, users may see the below warning.

```
scripts/gather_hw_os_kernel_bios_info.sh
bash: lstopo-no-graphics: command not found
bash: lstopo-no-graphics: command not found
bash: lstopo-no-graphics: command not found
bash: lstopo-no-graphics: command not found
bash: lstopo-no-graphics: command not found
bash: lstopo-no-graphics: command not found
bash: lstopo-no-graphics: command not found
bash: _HW_LSTOPO_NUM_L2CACHE/_HW_LSTOPO_PACKAGES: division by 0 (error token is
"_HW_LSTOPO_PACKAGES")
bash: _HW_LSTOPO_NUM_L3CACHE/_HW_LSTOPO_PACKAGES: division by 0 (error token is
"_HW_LSTOPO_PACKAGES")
sudo: dmidecode: command not found
sudo: dmidecode: command not found
sudo: dmidecode: command not found
```

5 Environment Variables

ZenDNN uses the environment variables listed below to setup paths, control logs, and tune performance:

Environment Variable	Default Value/User Defined Value
ZENDNN_LOG_OPTS	ALL:0
ZENDNN_GIT_ROOT	Dir, where ZenDNN folder is located
ZENDNN_PARENT_FOLDER	Parent folder for ZenDNN
ZENDNN_AOCC_COMP_PATH	AOCC compiler path[1]
ZENDNN_BLIS_PATH	AOCL BLIS path[1]
TF_GIT_ROOT	TensorFlow folder path
BENCHMARKS_GIT_ROOT	TensorFlow CNN benchmark directory

[1] User must set these environment variables explicitly.

TF_ZEN_PRIMITIVE_REUSE_DISABLE	False
ZENDNN_MEMPOOL_ENABLE	1
ZENDNN_PRIMITIVE_CACHE_CAPACITY	The default value is set to 500, user can change it as required[2]
ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE	0
ZENDNN_TF_INTEROP_THREADS	Default value is set to 1. For ZENDNN_TF_INTEROP_THREADS > 1, disable ZENDNN_MEMPOOL_ENABLE by setting it to 0
OMP_DYNAMIC	FALSE
ZENDNN_INFERENCE_ONLY	Default value is set to 1. User can change to 0 to enable training which will fall back to TensorFlow Vanilla as training is not supported by ZenDNN.

Following is a list of environment variables to tune performance:

Environment Variable	Default Value/User Defined Value
OMP_NUM_THREADS	The default value is set to 24. User can set it as per the number of cores in the user system[1].
OMP_WAIT_POLICY	ACTIVE
OMP_PROC_BIND	FALSE
GOMP_CPU_AFFINITY	Set it as per the number of cores in the system being used[1]
ZENDNN_TENSOR_POOL_LIMIT	16
ZENDNN_BLOCKED_FORMAT	The default value is set to 0. User can change to 1 to enable the Blocked Format support[2].
ZENDNN_INT8_SUPPORT	The default value is set to 0. User can change to 1 to enable the INT8 data type support. This only works with NHWC format (ZENDNN_BLOCKED_FORMAT=0).
ZENDNN_RELU_UPPERBOUND	The default value is set to 0. User can change to 1 to enable the Relu6 fusion. This only works with INT8 (ZENDNN_INT8_SUPPORT =1).

[2] These environment variables work only for Blocked Format.

ZENDNN_TF_CONV_ADD_FUSION_SAFE	The default value is set to 0. User can change to 1 to enable Conv, Add fusion. Currently it is safe to enable this switch for resnet50v1_5, resnet101, and inception_resnet_v2 models only

When *source scripts/zendnn_aocc_env_setup.sh* is invoked, the script initializes all the environment variables except the one which must be set manually. The environment variables **ZENDNN_PARENT_FOLDER**, **TF_GIT_ROOT**, and **BENCHMARKS_GIT_ROOT** are initialized relative to the path defined by **ZENDNN_GIT_ROOT**. To ensure that the paths are initialized correctly, it is important that the script is invoked from the ZenDNN folder.

6 Tuning Guidelines

The hardware configuration, OS, Kernel, and BIOS settings play an important role in performance. The details for the environment variables used on a 2nd Gen AMD EPYC server to get the best performance numbers are as follows:

2nd Gen AMD EPYC Machine Used:

Model name	2nd Gen AMD EPYC 7352 24-Core Processor
CPU MHz	1494.914
No of Cores	24
1P/2P	2
SMT: Thread(s) per Core	2
Mem-Dims	16x16 GB

OS Used: Ubuntu 18.04.05 LTS (Kernel 4.15.0-135-generic #139-Ubuntu SMP)

Environment Variables Used:

ZENDNN_LOG_OPTS=ALL:0

OMP_NUM_THREADS=24

OMP_WAIT_POLICY=ACTIVE

OMP_PROC_BIND=FALSE

ZENDNN_TF_INTEROP_THREADS=1

OMP_DYNAMIC=FALSE

ZENDNN_MEMPOOL_ENABLE=1

ZENDNN_TENSOR_POOL_LIMIT=16

ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE=0

ZENDNN_BLOCKED_FORMAT=0

ZENDNN_GIT_ROOT=/home/<user_id>/my_work/ZenDNN

ZENDNN_PARENT_FOLDER=/home/<user_id>/my_work

ZENDNN_AOCC_COMP_PATH=/home/<user_id>/my_work/aocc-compiler-3.0.0

ZENDNN_BLIS_PATH=/home/<user_id>/my_work/aocl-linux-aocc-3.0-6/amd-blis

TF_GIT_ROOT=/home/<user_id>/my_work/tensorflow

BENCHMARKS_GIT_ROOT=/home/<user_id>/my_work/benchmarks

PYTORCH_GIT_ROOT=/home/<user_id>/my_work/pytorch

PYTORCH_BENCHMARK_GIT_ROOT=/home/<user_id>/my_work/pytorch-benchmarks

ONNXRUNTIME_GIT_ROOT=/home/<user_id>/my_work/onnxruntime

ZENDNN_PRIMITIVE_CACHE_CAPACITY=500

ZENDNN_INT8_SUPPORT=0

ZENDNN_RELU_UPPERBOUND=0

ZENDNN_INFERENCE_ONLY=1

ZENDNN_TF_CONV_ADD_FUSION_SAFE=0

GOMP_CPU_AFFINITY=0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23

As mentioned in the Environment Variables section, the script **scripts/zendnn_aocc_env_setup.sh**, initializes all the environment variables except the one which users must set manually. The environment variables **OMP_NUM_THREADS, OMP_WAIT_POLICY**, **OMP_PROC_BIND**, and **GOMP_CPU_AFFINITY** can be used to tune performance. For optimal performance, the **Batch Size** must be a multiple of the total number of cores (used by the threads). On a 2nd Gen AMD EPYC server (configuration: AMD EPYC 7352, 24-Cores, 2P, and SMT=ON), with the above environment variable values, **OMP_NUM_THREADS=24**, **GOMP_CPU_AFFINITY="0-23"**, and **Batch size =480** yield the best throughput numbers.

Batch Size is sensitive factor for the throughput performance of any model. The following formula calculates optimal Batch Size:

Batch Size = number_of_physical_cores * batch_factor

batch_factor may vary from 16-32; 32 usually gives the best performance.

A few of the models (e.g., publicly available ResNet50 model) gain performance with Transparent Huge Pages settings (THP). THP can be enabled as a sudo user using the command:

echo always > /sys/kernel/mm/transparent_hugepage/enabled

7 Support for Blocked Format

ZenDNN supports the Beta version of Blocked Format. It is also known as *nChw8c* format, which may provide optimized performance for some ML workloads. This can be enabled with the environment variable **ZENDNN_BLOCKED_FORMAT**.

export ZENDNN_BLOCKED_FORMAT=1

The environment variable must be unset to fall back to the default path (NHWC) again.

7.1 Optimal Setting

Optimal performance of several ZenDNN workloads is observed when interleaving is enabled in conjunction with NPS4 mode.

A sample command line to run a Python code with 64C in NPS4 mode is as follows:

export GOMP_CPU_AFFINITY=0-63 && export ZENDNN_BLOCKED_FORMAT=1 && export OMP_NUM_THREADS=64 && numactl --cpunodebind=0-3 --interleave=0-3 python workload.py

8 Support for INT8

Quantization is an active area of research and a popular compression technique to accelerate neural network performance. Several competitive submissions from MLPerf leverage lower precisions to showcase hardware capability.

A few of these quantized neural networks models and pb files are publicly available. On AMD 3rd Gen EPYC platforms, ZenDNN offers options to enable experimental mode for INT8 quantization with widely used Resnet50 and MobileNetV1 models. The experimental models can be leveraged using publicly available Intel® AI models/benchmarks.

This can be enabled with the following environment variables:

export ZENDNN_BLOCKED_FORMAT=0

export ZENDNN_INT8_SUPPORT=1

export ZENDNN_RELU_UPPERBOUND=1

9 License

Please refer to the AMD ZenDNN EULA for more details on the third-party programs used in this release and their corresponding licenses. Upon selecting a ZenDNN package to download, Developer Central will redirect users to said AMD ZenDNN EULA document. The terms and conditions set forth in the EULA must be accepted before the download will commence.

This distribution includes the following third-party softwares, which are governed by separate license terms described in the AMD ZenDNN EULA:

- oneDNN
- Xbyak
- Boost C++ Libraries
- TensorFlow
- Caffe
- gtest

10 Technical Support

Please email zendnnsupport@amd.com for questions, issues, and feedback regarding ZenDNN.