



**User Guide ONNXRT v1.5.1 - ZenDNN v3.0**  
**April 2021**

© 2020 - 2021 Advanced Micro Devices, Inc. All rights reserved.

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD's products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. Any unauthorized copying, alteration, distribution, transmission, performance, display, or other use of this material is prohibited.

## **Trademarks**

AMD, the AMD Arrow logo, AMD-V, AMD Virtualization, and combinations thereof, are trademarks of Advanced Micro Devices, Inc.

Windows is a registered trademark of Microsoft Corporation.

MMX is a trademark of Intel Corporation.

Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

## **Document Information**

Software Version: 3.0  
Document Version: 3.0  
Last Updated: 21<sup>st</sup> April 2021

# Contents

<b>1</b>	<b>Installing ZenDNN.....</b>	<b>1</b>
1.1	Binary Release .....	1
1.2	Docker Release .....	1
1.2.1	Instructions for New Session .....	2
<b>2</b>	<b>Directory Structure .....</b>	<b>2</b>
<b>3</b>	<b>High Level Overview - Block Diagram.....</b>	<b>3</b>
<b>4</b>	<b>ONNXRT CNN and BERT Benchmarks .....</b>	<b>3</b>
<b>5</b>	<b>Environment Variables .....</b>	<b>4</b>
<b>6</b>	<b>Tuning Guidelines .....</b>	<b>5</b>
<b>7</b>	<b>Support for Blocked Format .....</b>	<b>7</b>
7.1	Optimal Settings.....	7
<b>8</b>	<b>License .....</b>	<b>8</b>
<b>9</b>	<b>Technical Support .....</b>	<b>8</b>

# 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 `ONNXRT_v1.5.1_ZenDNN_v3.0.zip`.
2. Execute the following commands:
  - a. `unzip ONNXRT_v1.5.1_ZenDNN_v3.0.zip`
  - b. `cd ONNXRT_v1.5.1_ZenDNN_release*/ZenDNN`
  - c. Ensure that the environment variable **ZENDNN\_BLIS\_PATH** and **ZENDNN\_AOCC\_COMP\_PATH** is 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 **onnxrt-1.5.1-zendnn-rel-env** already present, delete the conda environment **onnxrt-1.5.1-zendnn-rel-env** (using command `conda remove --name onnxrt-1.5.1-zendnn-rel-env --all`) before running `scripts/zendnn_aocc_env_setup.sh`.
- e. `source scripts/ONNXRT_v1.5.1_ZenDNN_setup_release.sh`  
This installs the ONNXRT wheel package provided in the zip file and downloads the ONNXRT BERT and ResNet50 benchmarks.
  - f. To run the benchmarks with ResNet50 and BERT models at ONNXRT level, refer to the ONNXRT CNN and BERT Benchmarks section.

Ensure that the environment variable **ZENDNN\_BLIS\_PATH** and **ZENDNN\_AOCC\_COMP\_PATH** are set.

## 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 `ONNXRT_v1.5.1_ZenDNN_v3.0_docker.tar.gz`.

### Prerequisites

Install docker on 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 ONNXRT_v1.5.1_ZenDNN_v3.0_docker.tar.gz`

5. Run Docker in privileged mode:

```
docker run --privileged -itd --name zendnn-onnxrt-rel onnxrt_v1.5.1_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
```
7. To run the benchmarks script at ONNXRT level, refer to the ONNXRT CNN and BERT 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 `onnxrt_v1.5.1_zendnn:3.0`:  

```
docker ps -a
```

```
#CONTAINER ID    IMAGE                COMMAND              CREATED          STATUS
#2c8af0175b8a   onnxrt_v1.5.1_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

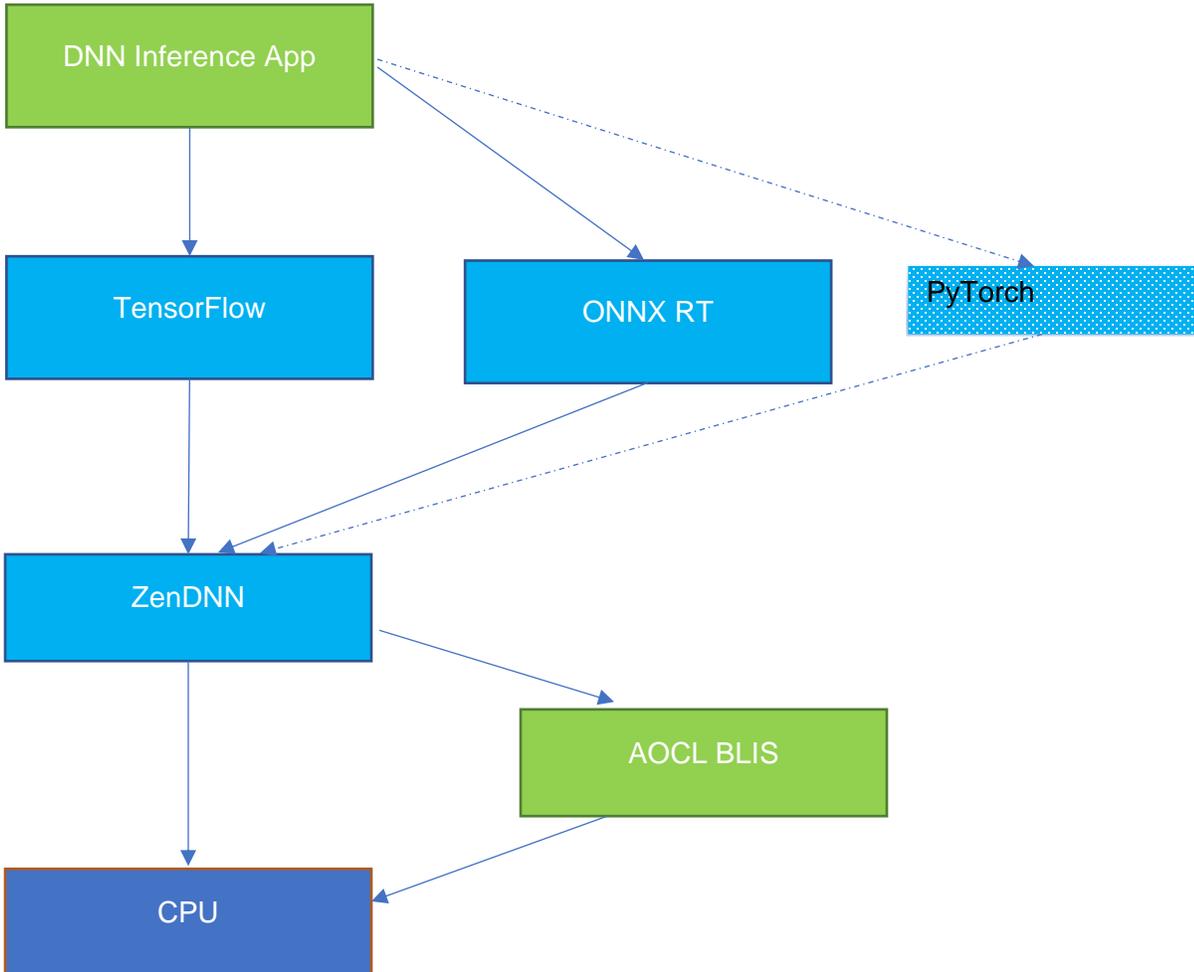
The release folder consists of ONNXRT *whl* file, `onnxruntime`, and ZenDNN binary package.

ZenDNN consists of the following directories:

- **benchmark:** contains the benchmarks utility.
- **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 ONNXRT CNN and BERT Benchmarks

The following benchmark scripts provide performance benchmarking at the ONNXRT level, printing latency and throughput results for the ResNet50 and BERT models.

To use the benchmarking scripts, execute the following commands:

1. `cd ONNXRT_v1.5.1_ZenDNN_release*/ZenDNN`
2. `source scripts/zendnn_aocc_env_setup.sh`
3. `source scripts/onnxrt_resnet50v1_benchmark.sh`
4. `source scripts/onnxrt_bert_benchmark.sh`

**Note:**

- For optimal settings, refer to the Tuning Guidelines section. Current setting refers to 64C, 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: _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 <sup>1</sup>
ZENDNN_BLIS_PATH	AOCL BLIS path <sup>1</sup>
ONNXRUNTIME_GIT_ROOT	ONNX Runtime folder path
ZENDNN_MEMPOOL_ENABLE	1
ZENDNN_PRIMITIVE_CACHE_CAPACITY	The default value is set to 500, user can change it as required <sup>2</sup>

<sup>1</sup> User must set these environment variables explicitly.

<sup>2</sup> These environment variables work only for Blocked Format.

<b>ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE</b>	0
<b>OMP_DYNAMIC</b>	FALSE

Following is a list of environment variables to tune performance:

Environment Variable	Default Value/User Defined Value
<b>OMP_NUM_THREADS</b>	The default value is set to 24. User can set it as per the number of cores in the user system <sup>1</sup> .
<b>OMP_WAIT_POLICY</b>	ACTIVE
<b>OMP_PROC_BIND</b>	FALSE
<b>GOMP_CPU_AFFINITY</b>	Set it as per the number of cores in the system being used <sup>1</sup>
<b>ZENDNN_TENSOR_POOL_LIMIT</b>	16
<b>ZENDNN_BLOCKED_FORMAT</b>	The default value is set to 0. User can change to 1 to enable the Blocked Format support <sup>2</sup> .

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** and **ONNXRUNTIME\_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:

<b>Model name</b>	2 <sup>nd</sup> Gen AMD EPYC 7742 64-Core Processor
<b>CPU MHz</b>	1497.181
<b>No of Cores</b>	64
<b>1P/2P</b>	2
<b>SMT: Thread(s) per Core</b>	2
<b>Mem-Dims</b>	16x32 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=64
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
GOMP_CPU_AFFINITY=0-63
```

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 7742, 64-Cores, 2P, and SMT=ON), with the above environment variable values, **OMP\_NUM\_THREADS=64**, **GOMP\_CPU\_AFFINITY="0-63"**, and **Batch size=640** yield the best throughput numbers.

Batch Size is a sensitive factor for 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.

## 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; up to 50% improvement on 64 core 2nd Gen AMD EPYC machines for certain models was observed using blocked format. 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 Settings

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 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](#)

## 9 Technical Support

---

Please email [zendnnsupport@amd.com](mailto:zendnnsupport@amd.com) for questions, issues, and feedback regarding ZenDNN.