# TensorFlow-ZenDNN Plug-in User Guide


| | |
|---|---|
| Publication # **CSG-1** | Revision # **0.1** |
| Issue Date **March 2023** | |

# Contents

# List of Figures

# List of Tables

# Revision History

| Date | Revision | Description |
|---|---|---|
| March 2023 | 0.1 | Initial version. |

# Chapter 1 Installing TensorFlow-ZenDNN Plug-in

***Note:*** *Refer to the ZenDNN v3.3 User Guide before starting the installation.*

## 1.1 TensorFlow-ZenDNN Plug-in Setup

This section describes the procedure to setup the TensorFlow-ZenDNN plug-in for TensorFlow v2.12.

### 1.1.1 TensorFlow v2.12

Complete the following steps to install the TensorFlow-ZenDNN plug-in:

1. Install TensorFlow v2.12:

```
pip install tensorflow-cpu==2.12.0
```

2. Download the TensorFlow-ZenDNN plug-in wheel file from the *Community supported TensorFlow builds*.
3. Install TensorFlow-ZenDNN plug-in:

```
pip install tensorflow_zendnn_plugin-0.1.0-cp38-cp38-manylinux_2_17_x86_64.manylinux-
2014_x86_64.whl
```

4. Set the following environment variables to enable ZenDNN for inference:
   - **TF_ENABLE_ZENDNN_OPTS=1**
   - **TF_ENABLE_ONEDNN_OPTS=0**

     By default, TensorFlow is shipped with oneDNN enabled.

The release binaries for TensorFlow-ZenDNN plug-in v0.1 are compiled with manylinux2014 and they provide compatibility with some older Linux distributions.

The release binaries are tested with the recent Linux distributions such as:

- Ubuntu 20.04 and later
- RHEL 9.0 and later

To run a sample with the installed TensorFlow-ZenDNN plug-in, follow the instructions in *Unified Inference Frontend (UIF) 1.1 User Guide - Run a CPU Example*.

# Chapter 2 High-level Overview

The following is a high-level block diagram for the TensorFlow-ZenDNN plug-in package which utilizes ZenDNN as the core inference library:

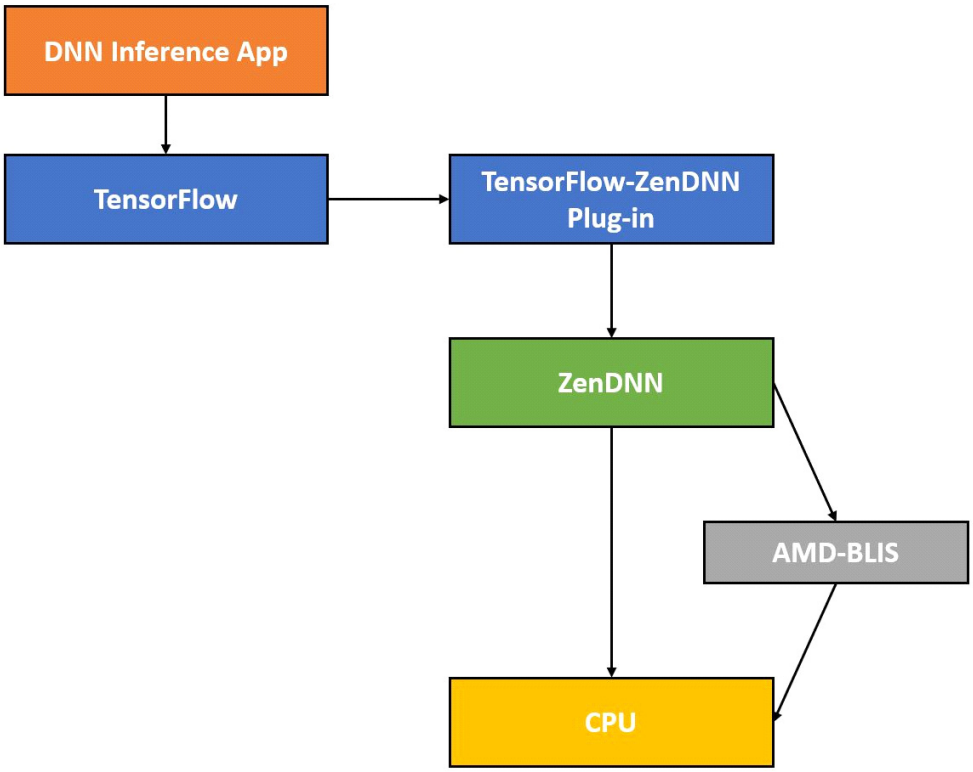


**Figure 1. TensorFlow-ZenDNN Plug-in**

# Chapter 3 TensorFlow-ZenDNN Plug-in v0.1

TensorFlow-ZenDNN plug-in v0.1 is the first release with Pluggable device approach of TensorFlow:

- This plug-in is supported for TensorFlow v2.12 and later.
- It is integrated with ZenDNNv3.3 as the core inference library and compiled with GCC v9.3.1.
- As compared to the current TensorFlow-ZenDNN direct integration releases, this release provides:
  - On par performance for models, such asRefineDet, Inception, and VGG variants.
  - Sub-optimal performance for models, such as ResNet, MobileNet and EfficientNet.

# Chapter 4 Environment Variables

TensorFlow-ZenDNN plug-in uses the following environment variables to tune performance and control logs:

**Table 1. ZenDNN Environment Variables-Generic**

| Environment Variable | Default Value/User Defined Value |
|---|---|
| ZENDNN_LOG_OPTS | ALL:0 |
| TF_ZEN_PRIMITIVE_REUSE_DISABLE | False |
| ZENDNN_ENABLE_MEMPOOL | The default value is set to 1, you can provide the value 0 to disable it. 1 is for Node-based MEMPOOL and 2 is for Graph-based MEMPOOL. |
| ZENDNN_PRIMITIVE_CACHE_CAPACITY | The default value is set to 1024, you can modify it as required |
| ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE | 0 |
| TF_ENABLE_ZENDNN_OPTS | Default value is set to 0. Set it to 1 along with TF_ENABLE_ONEDNN_OPTS=0 for enabling ZenDNN for inference. You can set it to 0 when you want to enable vanilla training and inference. |
| TF ENABLE_ONEDNN_OPTS | Default value is set to 1. By default, TensorFlow is shipped with oneDNN optimizations enabled. Hence, set it to 0 when you enable ZenDNN. |

**Table 2. ZenDNN Environment Variables-Optimization**

| Environment Variable | Default Value/User Defined Value |
|---|---|
| OMP_NUM_THREADS | Set it as per the number of cores in the user system[a]. |
| OMP_DYNAMIC | Set it to FALSE for optimal performance[a]. |
| OMP_PROC_BIND | Set it to FALSE for optimal performance[a]. |
| GOMP_CPU_AFFINITY | Set it as per the number of cores in the system being used[a]. |
| ZENDNN_TENSOR_POOL_LIMIT | The default value is set to 32. You can modify it to 256 for optimal performance. |
| ZENDNN_BLOCKED_FORMAT | The default value is set to 0. You can modify it to 1 to enable the Blocked Format support. |
| ZENDNN_NHWC_BLOCKED | The default value is set to 0. You can modify it to 1 to enable implicit Blocked Format support. |
| ZENDNN_GEMM_ALGO | The default value is 0. ZenDNN library offers several execution paths tailored for different workloads. The value 0 represents ZenDNN GEMM path. You can modify it to 1 to enable BLIS path or 2 for partial-BLIS execution. |

a. You must set these environment variables explicitly.

# Chapter 5 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 4th Gen AMD EPYC™ server to achieve the optimal performance numbers are as follows:

## 5.1 System

A system with the following specifications has been used:

**Table 3. System Specification**

| | |
|---|---|
| **Model name** | 4th Gen AMD EPYC™ 9654P 96-Core Processor |
| **DPU MHz** | Up to 3.7 GHz |
| **No of Cores** | 96 |
| **1P/2P** | 1 |
| **SMT: Thread(s) per Core** | 2 |
| **Mem-Dims** | 12x64 GB |

**OS Used:** Ubuntu 20.04.02 LTS

## 5.2 Environment Variables

The following environment variables have been used:

**ZENDNN_LOG_OPTS=ALL:0**

**OMP_NUM_THREADS=96**

**OMP_WAIT_POLICY=ACTIVE**

**OMP_PROC_BIND=FALSE**

**OMP_DYNAMIC=FALSE**

**ZENDNN_ENABLE_MEMPOOL=1**

**ZENDNN_GEMM_ALGO=0**

**ZENDNN_TENSOR_POOL_LIMIT=32**

**ZENDNN_TENSOR_BUF_MAXSIZE_ENABLE=0**

**ZENDNN_BLOCKED_FORMAT=0**

**ZENDNN_NHWC_BLOCKED=0**

**ZENDNN_PRIMITIVE_CACHE_CAPACITY=1024**

**TF_ENABLE_ZENDNN_OPTS=1**

**TF_ENABLE_ONEDNN_OPTS=0**

**GOMP_CPU_AFFINITY=0-95**

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 4$^{th}$ Gen AMD EPYC$^{TM}$ server (configuration: AMD EPYC$^{TM}$ 9654P 96-Core, 2P, and **SMT=ON**) with the above environment variable values, **OMP_NUM_THREADS=96** and **GOMP_CPU_AFFINITY=0-95** yield the best throughput numbers for a single socket.

**Batch Size** is a sensitive factor for the throughput performance of any model. The following formula could be used to calculate the optimal **Batch Size**:

**Batch Size = number_of_physical_cores * batch_factor**

**batch_factor** may vary from 8-32. Usually, the value 32 gives the optimal performance.

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

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

# Chapter 6 Blocked Format Support

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 variables **ZENDNN_BLOCKED_FORMAT** (explicit) or **ZENDNN_NHWC_BLOCKED** (implicit) as follows:

**export ZENDNN_BLOCKED_FORMAT=1**

With this format, an Op that operates on BLOCKED format requires explicit reordering of input buffer from NHWC to BLOCKED (nChw8c) when the input is not in BLOCKED format.

**export ZENDNN_NHWC_BLOCKED=1**

With this format, an Op that operates on BLOCKED format handles reordering of input buffer from NHWC to BLOCKED (nChw8c) when the input is not in BLOCKED format.

The environment variable must be set to 0 or unset altogether to fall back to the default path (NHWC) again.

## 6.1 Optimal Interleaving Setting

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

A sample command line to run a Python code with 96C in NPS4 mode is as follows:

```
export GOMP_CPU_AFFINITY=0-95 && export ZENDNN_BLOCKED_FORMAT=1 && export OMP_NUM_-
THREADS=96 && numactl --cpunodebind=0-3 --interleave=0-3 python workload.py
```

# Chapter 7 License

TensorFlow-ZenDNN plug-in is licensed under Apache License Version 2.0. Refer to the "LICENSE" file for the full license text and copyright notice.

This distribution includes third party software governed by separate license terms.

**3-clause BSD license:**

- Xbyak (*https://github.com/herumi/xbyak*)
- Googletest (*https://github.com/google/googletest*)
- Instrumentation and Tracing Technology API (*https://github.com/intel/ittapi*)

**Apache License Version 2.0:**

- oneDNN (*https://github.com/oneapi-src/oneDNN*)
- Xbyak_aarch64 (*https://github.com/fujitsu/xbyak_aarch64*)
- TensorFlow (*https://github.com/tensorflow/tensorflow*)

**Boost Software License, Version 1.0:**

Boost C++ Libraries (*https://www.boost.org/*)

**BSD 2-Clause license:**

Caffe (*https://github.com/BVLC/caffe*)

This third-party software, even if included with the distribution of the Advanced Micro Devices software, may be governed by separate license terms, including without limitation, third-party license terms, and open-source software license terms. These separate license terms govern use of the third-party programs as set forth in the THIRD-PARTY-PROGRAMS file.

# Chapter 8 Technical Support

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