DALI

DU-09049-001 _v0.6.1 Beta Release | March 2019

Installation Guide

TABLE OF CONTENTS

Chapter 1. Overview..... 1
Chapter 2. DALI And NGC..... 2
Chapter 3. Installing DALI..... 3
3.1. Installing Prebuilt DALI Packages..... 3
3.1.1. Prerequisites..... 3
3.1.2. Binary Installation..... 3
3.2. Compiling DALI From Source (Bare metal)..... 4
3.2.1. Prerequisites..... 4
3.2.2. GitHub Installation..... 5
3.2.2.1. CMake Build Parameters..... 5
3.2.3. Installing Python Bindings..... 6
3.3. Compiling DALI from Source (Docker)..... 6
Chapter 4. Executing ResNet-50 Input Pipeline..... 7
Chapter 5. FFmpeg..... 8
Chapter 6. Uninstalling DALI..... 9

Chapter 1. OVERVIEW

Deep learning applications require complex, multi-stage pre-processing data pipelines. Such data pipelines involve compute-intensive operations that are carried out on the CPU. For example, tasks such as: load data from disk, decode, crop, random resize, color and spatial augmentations and format conversions, are mainly carried out on the CPUs, limiting the performance and scalability of training and inference.

In addition, the deep learning frameworks have multiple data pre-processing implementations, resulting in challenges such as portability of training and inference workflows, and code maintainability.

NVIDIA Data Loading Library (DALI) is a collection of highly optimized building blocks, and an execution engine, to accelerate the pre-processing of the input data for deep learning applications. DALI provides both the performance and the flexibility of accelerating different data pipelines as a single library. This single library can then be easily integrated into different deep learning training and inference applications.

Highlights of DALI are:

- Full data pipeline—accelerated from reading disk to getting ready for training/ inference.
- Flexibility through configurable graphs and custom operators.
- Support for image classification and segmentation workloads.
- Ease of integration through direct framework plugins and open source bindings.
- Portable training workflows with multiple input formats - JPEG, PNG (fallback to CPU), TIFF (fallback to CPU), BMP (fallback to CPU), raw formats, LMDB, RecordIO, TFRecord.
- Extensible for user specific needs through open source license.

Chapter 2. DALI AND NGC

DALI is pre-installed in the NVIDIA GPU Cloud TensorFlow, PyTorch, and MXNet containers in version 18.07 and later.

Chapter 3. INSTALLING DALI

DALI can be installed either directly using a pre-built binary or by compiling the sources from GitHub.

3.1. Installing Prebuilt DALI Packages

3.1.1. Prerequisites

Ensure you meet the following minimum requirements:

- Linux x64
- NVIDIA Driver (384.xx or later driver releases) supporting CUDA 9.0 or later
- One or more of the following deep learning frameworks:
  - MXNet 1.3 or later
    - Version 1.3 from the Python package with the following command:

      ```
      pip install mxnet-cu90==1.3.0
      ```
  - PyTorch 0.4
  - TensorFlow 1.7 or later.

3.1.2. Binary Installation

1. Install DALI using **pip**:

   ```
   pip install --extra-index-url
   https://developer.download.nvidia.com/compute/redist nvidia-dali
   ```
2. Install DALI Tensorflow plugin

   The **nvidia-dali** package contains prebuilt versions of the dali tensorflow plugin for several versions of tensorflow. Since release 0.6.1 there is also a possibility to install dali tensorflow plugin for the currently installed version of tensorflow, thus allowing forward compatibility:

```
pip install --extra-index-url https://developer.download.nvidia.com/compute/
redist nvidia-dali-tf-plugin
```

3. Install **nvidia-dali-tf-plugin** package

   Installing this package will install **nvidia-dali** and its dependencies if not already installed. The package **tensorflow-gpu** must be installed before attempting to install **nvidia-dali-tf-plugin**.

   The package **nvidia-dali-tf-plugin** has a strict requirement with **nvidia-dali** as its exact same version. Thus, installing **nvidia-dali-tf-plugin** at its latest version will replace any older **nvidia-dali** versions already installed with the latest. To work with older versions of DALI, please provide the version explicitly to the pip install command.

```
OLDER_VERSION=0.6.1
pip install --extra-index-url https://developer.download.nvidia.com/compute/
redist nvidia-dali-tf-plugin==$OLDER_VERSION
```

3.2. Compiling DALI From Source (Bare metal)

3.2.1. Prerequisites

Ensure that you meet the below minimum requirements:

- Linux x64.
- GCC 4.9.2 or later.
- NVIDIA CUDA 9.0 (CUDA 8.0 compatibility is provided *unofficially*[1]) .
- nvJPEG library (This can be *unofficially* disabled[1]).
- protobuf version 2 or later (version 3 or later is required for TensorFlow TFRecord file format support).
- CMake 3.5 or later.
- FFmpeg 3.4.2 recommend using version 3.4.2 compiled with the instructions provided in this document.
- libjpeg-turbo 1.5.x or later (This can be *unofficially* disabled[1]).
- OpenCV 3 or later (OpenCV 2.x compatibility is provided *unofficially*[1]).
- liblmdb 0.9.x or later.
- One or more of the following deep learning frameworks:
  - MXNet 1.3 or later
    - Version 1.3 from the Python package with the following command:

      ```
      pip install mxnet-cu90==1.3.0
      ```
  - PyTorch 0.4
  - TensorFlow 1.7 or later.

[1] Items marked *unofficial* are community contributions that are believed to work but not officially tested or maintained by NVIDIA.

3.2.2. GitHub Installation

1. Download the DALI source package from GitHub.

```
git clone --recursive https://github.com/NVIDIA/dali
cd dali
```

2. Create the build directory.

```
mkdir build
cd build
```

3. Compile DALI.
   a) To build DALI without LMDB support, issue the following command:

```
cmake ..
make -j"$(nproc)"
```

   b) To build DALI with LMDB support, issue the following command:

```
cmake -DBUILD_LMDB=ON ..
make -j"$(nproc)"
```

   c) To build DALI using Clang, issue the following command:

Caution This build is experimental, meaning it is not maintained and tested like the default configuration, therefore, it's not guaranteed to work. We recommend using GCC for production builds.

```
cmake -DCMAKE_CXX_COMPILER=clang++ -DCMAKE_C_COMPILER=clang  ..
make -j"$(nproc)"
```

3.2.2.1. CMake Build Parameters

You can use the following optional CMake build parameters when configuring DALI:

BUILD_PYTHON
Use this parameter to build Python bindings. The default is **ON**.

BUILD_TEST
Use this parameter to include building the test suite. The default is **ON**.

BUILD_BENCHMARK
Use this parameter to include building benchmarks. The default is **ON**.

BUILD_LMDB
Use this parameter to build with support for LMDB. The default is **OFF**.

BUILD_NVTX
Use this parameter to build with NVTX profiling enabled. The default is **OFF**.

BUILD_TENSORFLOW
Use this parameter to build the TensorFlow plugin. The default is **OFF**.

WERROR
Treats all build warnings as errors. The default is **OFF**.

BUILD_JPEG_TURBO*(unofficial)*
Use this parameter to build with libjpeg-turbo. The default is **ON**.[2]

[2] Items marked *unofficial* are community contributions that are believed to work but not officially tested or maintained by NVIDIA.

BUILD_NVJPEG *(unofficial)*
Use this parameter to build with nvJPEG. The default is **ON**.[3]

3.2.3. Installing Python Bindings

Issue the **pip install dali/python** command to install Python bindings.

3.3. Compiling DALI from Source (Docker)

1. Ensure you installed the below prerequisites:
   - Linux x64
   - Docker. Follow the Docker installation guide here.
2. Building Docker Image.

   Change directory (**cd**) into Docker directory and run **./build.sh**. If needed, set the following environment variables:

   - PYVER - Python version. Default is **2.7**.
   - CUDA_VERSION - CUDA toolkit version. Default is **10**.
   - NVIDIA_BUILD_ID - Custom ID of the build. Default is **1234**.
   - CREATE_RUNNER - Create Docker image with cuDNN, CUDA and DALI installed inside. It will create the **Docker_run_cuda** image, which needs to be run using **nvidia-docker** and DALI wheel in the **wheelhouse** directory under **DALI/`**. Default is **NO**.
   - CREATE_WHL - Create a wheel also. Default is **YES**.

[3] Items marked *unofficial* are community contributions that are believed to work but not officially tested or maintained by NVIDIA.

Chapter 4.
EXECUTING RESNET-50 INPUT PIPELINE

After you've installed DALI, you can run a pre-configured, ResNet-50 model accelerated by DALI, on MXNet, PyTorch, and TensorFlow frameworks for image classification training. Each of the following samples offload image loading and augmentation operations onto GPUs.

You can use Python toolchain from the command shell or Jupyter notebook to start the ResNet-50 training session.

The DALI integrated ResNet-50 Python samples are located:

- MXNet
- PyTorch
- TensorFlow

Chapter 5. FFMPEG

This software uses code of FFmpeg licensed under the LGPLv2.1 and its source can be downloaded here.

FFmpeg was compiled using the following command line:

```
./configure \
 --prefix=/usr/local \
 --disable-static \
 --disable-all \
 --disable-autodetect \
 --disable-iconv \
 --enable-shared \
 --enable-avformat \
 --enable-avcodec \
 --enable-avfilter \
 --enable-protocol=file \
 --enable-demuxer=mov,matroska \
 --enable-bsf=h264_mp4toannexb,hevc_mp4toannexb
./make
```

Chapter 6. UNINSTALLING DALI

Uninstall DALI.

```
pip uninstall -y nvidia-dali
```

Notice

THE INFORMATION IN THIS GUIDE AND ALL OTHER INFORMATION CONTAINED IN NVIDIA DOCUMENTATION REFERENCED IN THIS GUIDE IS PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE INFORMATION FOR THE PRODUCT, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE. Notwithstanding any damages that customer might incur for any reason whatsoever, NVIDIA's aggregate and cumulative liability towards customer for the product described in this guide shall be limited in accordance with the NVIDIA terms and conditions of sale for the product.

THE NVIDIA PRODUCT DESCRIBED IN THIS GUIDE IS NOT FAULT TOLERANT AND IS NOT DESIGNED, MANUFACTURED OR INTENDED FOR USE IN CONNECTION WITH THE DESIGN, CONSTRUCTION, MAINTENANCE, AND/OR OPERATION OF ANY SYSTEM WHERE THE USE OR A FAILURE OF SUCH SYSTEM COULD RESULT IN A SITUATION THAT THREATENS THE SAFETY OF HUMAN LIFE OR SEVERE PHYSICAL HARM OR PROPERTY DAMAGE (INCLUDING, FOR EXAMPLE, USE IN CONNECTION WITH ANY NUCLEAR, AVIONICS, LIFE SUPPORT OR OTHER LIFE CRITICAL APPLICATION). NVIDIA EXPRESSLY DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY OF FITNESS FOR SUCH HIGH RISK USES. NVIDIA SHALL NOT BE LIABLE TO CUSTOMER OR ANY THIRD PARTY, IN WHOLE OR IN PART, FOR ANY CLAIMS OR DAMAGES ARISING FROM SUCH HIGH RISK USES.

NVIDIA makes no representation or warranty that the product described in this guide will be suitable for any specified use without further testing or modification. Testing of all parameters of each product is not necessarily performed by NVIDIA. It is customer's sole responsibility to ensure the product is suitable and fit for the application planned by customer and to do the necessary testing for the application in order to avoid a default of the application or the product. Weaknesses in customer's product designs may affect the quality and reliability of the NVIDIA product and may result in additional or different conditions and/or requirements beyond those contained in this guide. NVIDIA does not accept any liability related to any default, damage, costs or problem which may be based on or attributable to: (i) the use of the NVIDIA product in any manner that is contrary to this guide, or (ii) customer product designs.

Other than the right for customer to use the information in this guide with the product, no other license, either expressed or implied, is hereby granted by NVIDIA under this guide. Reproduction of information in this guide is permissible only if reproduction is approved by NVIDIA in writing, is reproduced without alteration, and is accompanied by all associated conditions, limitations, and notices.

Trademarks

NVIDIA, the NVIDIA logo, and cuBLAS, CUDA, cuDNN, cuFFT, cuSPARSE, DALI, DIGITS, DGX, DGX-1, Jetson, Kepler, NVIDIA Maxwell, NCCL, NVLink, Pascal, Tegra, TensorRT, and Tesla are trademarks and/or registered trademarks of NVIDIA Corporation in the Unites States and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2019 NVIDIA Corporation. All rights reserved.