Bright Cluster Manager 9.2

Machine Learning Manual

Revision: 24fd036c0

Date: Mon May 16 2022

Trademarks

Linux is a registered trademark of Linus Torvalds. PathScale is a registered trademark of Cray, Inc. Red Hat and all Red Hat-based trademarks are trademarks or registered trademarks of Red Hat, Inc. SUSE is a registered trademark of Novell, Inc. NVIDIA, CUDA, GPUDirect, HPC SDK, NVIDIA DGX, NVIDIA Nsight, NVLink and Tesla are registered trademarks of NVIDIA Corporation. FLEXlm is a registered trademark of Flexera Software, Inc. PBS Professional, PBS Pro, and Green Provisioning are trademarks of Altair Engineering, Inc. All other trademarks are the property of their respective owners.

Rights and Restrictions

All statements, specifications, recommendations, and technical information contained herein are current or planned as of the date of publication of this document. They are reliable as of the time of this writing and are presented without warranty of any kind, expressed or implied. Bright Computing, Inc. shall not be liable for technical or editorial errors or omissions which may occur in this document. Bright Computing, Inc. shall not be liable for any damages resulting from the use of this document.

Limitation of Liability and Damages Pertaining to Bright Computing, Inc.

The Bright Cluster Manager product principally consists of free software that is licensed by the Linux authors free of charge. Bright Computing, Inc. shall have no liability nor will Bright Computing, Inc. provide any warranty for the Bright Cluster Manager to the extent that is permitted by law. Unless confirmed in writing, the Linux authors and/or third parties provide the program as is without any warranty, either expressed or implied, including, but not limited to, marketability or suitability for a specific purpose. The user of the Bright Cluster Manager product shall accept the full risk for the quality or performance of the product. Should the product malfunction, the costs for repair, service, or correction will be borne by the user of the Bright Cluster Manager product. No copyright owner or third party who has modified or distributed the program as permitted in this license shall be held liable for damages, including general or specific damages, damages caused by side effects or consequential damages, resulting from the use of the program or the un-usability of the program (including, but not limited to, loss of data, incorrect processing of data, losses that must be borne by you or others, or the inability of the program to work together with any other program), even if a copyright owner or third party had been advised about the possibility of such damages unless such copyright owner or third party has signed a writing to the contrary.

Table of Contents

Preface

Welcome to the *Machine Learning Manual* for Bright Cluster Manager 9.2.

0.1 About This Manual

This manual is aimed at helping cluster administrators install, understand, configure, and manage basic machine learning capabilities easily using Bright Cluster Manager. The administrator is expected to be reasonably familiar with the *Administrator Manual*.

0.2 About The Manuals In General

Regularly updated versions of the Bright Cluster Manager 9.2 manuals are available on updated clusters by default at `/cm/shared/docs/cm`. The latest updates are always online at `http://support.brightcomputing.com/manuals`.

- The *Installation Manual* describes installation procedures for a basic cluster.
- The *Administrator Manual* describes the general management of the cluster.
- The *User Manual* describes the user environment and how to submit jobs for the end user.
- The *Cloudbursting Manual* describes how to deploy the cloud capabilities of the cluster.
- The *Developer Manual* has useful information for developers who would like to program with Bright Cluster Manager.
- The *Edge Manual* describes how to deploy Bright Edge with Bright Cluster Manager.
- The *Machine Learning Manual*—this manual—describes how to install and configure machine learning capabilities with Bright Cluster Manager.

If the manuals are downloaded and kept in one local directory, then in most pdf viewers, clicking on a cross-reference in one manual that refers to a section in another manual opens and displays that section in the second manual. Navigating back and forth between documents is usually possible with keystrokes or mouse clicks.

For example: `<Alt>-<Backarrow>` in Acrobat Reader, or clicking on the bottom leftmost navigation button of xpdf, both navigate back to the previous document.

The manuals constantly evolve to keep up with the development of the Bright Cluster Manager environment and the addition of new hardware and/or applications. The manuals also regularly incorporate customer feedback. Administrator and user input is greatly valued at Bright Computing. So any comments, suggestions or corrections will be very gratefully accepted at `manuals@brightcomputing.com`.

There is also a feedback form available via Bright View, via the Account icon, , following the clickpath:

`Account→Help→Feedback`

0.3 Getting Administrator-Level Support

If the reseller from whom Bright Cluster Manager was bought offers direct support, then the reseller should be contacted.

Otherwise the primary means of support is via the website `https://support.brightcomputing.com`. This allows the administrator to submit a support request via a web form, and opens up a trouble ticket. It is a good idea to try to use a clear subject header, since that is used as part of a reference tag as the ticket progresses. Also helpful is a good description of the issue. The followup communication for this ticket goes via standard e-mail. Section 17.2 of the *Administrator Manual* has more details on working with support.

0.4 Getting Professional Services

Bright Computing normally differentiates between professional services (customer asks Bright Computing to do something or asks Bright Computing to provide some service) and support (customer has a question or problem that requires an answer or resolution). Professional services can be provided after consulting with the reseller, or the Bright account manager.

1

Introduction And Machine Learning Packages Installation

1.1 Introduction

From Bright Cluster Manager version 7.3 onward, a number of machine learning and deep learning library and framework packages can be used. The packages provided make it faster and easier for organizations to install the latest state-of-the-art libraries, and gain insights from rich, complex data.

1.1.1 Bright Cluster Manager Versions—Associated Repositories And Support For Machine Learning Packages

- In Bright Cluster Manager versions 7.3 and 8.0 the machine learning and deep learning packages are experimentally accessible to a cluster from the standard Bright `cm` repository. However, from Bright Cluster Manager version 8.1 onward, these packages are distributed via a separate Bright `cm-ml` repository.
- In Bright Cluster Manager versions 8.1 and 8.2, cluster administrators have to activate the Data Science Add-on using the online wizard at:

  `http://licensing.brightcomputing.com/licensing/activate-data-science/`

  The add-on enables the dedicated Bright `cm-ml` repository.
- From Bright Cluster Manager version 9.0 onward the activation step via a wizard is not needed—the repository is automatically available and enabled.

Only if the cluster is licensed for the Data Science Add-on package does Bright provide support for the integration of packages distributed in the dedicated repository with Bright Cluster Manager.

For convenience, a summary of the Bright for Data Science (B4DS) repository configuration requirements is shown in table 1.1:

Table 1.1: Bright Cluster Manager offering overview for Machine Learning packages

Bright CM	Packages repository	Repository configuration	Support
7.3	Traditional (`cm`)	Automatically available and enabled	Not available
8.0	Traditional (`cm`)	Automatically available and enabled	Not available
8.1	Dedicated (`cm-ml`)	Requires B4DS Add-on activation*	Requires B4DS Add-on
8.2	Dedicated (`cm-ml`)	Requires B4DS Add-on activation*	Requires B4DS Add-on
9.0	Dedicated (`cm-ml`)	Automatically available and enabled	Requires B4DS Add-on
9.1	Dedicated (`cm-ml`)	Automatically available and enabled	Requires B4DS Add-on
9.2	Dedicated (`cm-ml`)	Automatically available and enabled	Requires B4DS Add-on

* Activation is via `http://licensing.brightcomputing.com/licensing/activate-data-science/`

An administrator who is upgrading a cluster that has machine learning and deep learning packages installed on it should always make sure that the dedicated Bright `cm-ml` repository is accessible, if required by the new Bright Cluster Manager version.

1.1.2 Bright Cluster Manager Versions—Supported Distributions And Architecture For Machine Learning Packages

At the time of writing, December 2021, a number of different Linux distribution versions and architectures are supported, depending on the Bright Cluster Manager version. For convenience, a support matrix for this is shown in table 1.2:

Table 1.2: Supported Linux distributions and architectures for Bright Machine Learning packages

Bright CM	Architectures	Linux distributions
7.3	x86_64	CentOS 7, RHEL 7
8.0	x86_64	CentOS 7, RHEL 7
8.1	x86_64	CentOS 7, RHEL 7
8.2	x86_64	CentOS 7, RHEL 7, Ubuntu 18.04
9.0	x86_64	CentOS 7, Rocky 8, RHEL 7, RHEL 8, SLES 15, Ubuntu 18.04
9.1	x86_64	CentOS 7, RHEL 7, RHEL 8, Rocky 8, SLES 15, Ubuntu 18.04, Ubuntu 20.04
9.2	x86_64	CentOS 7, RHEL 7, RHEL 8, Rocky 8, SLES 15, Ubuntu 18.04, Ubuntu 20.04

An updated list of the supported Linux distributions and architectures available for the various Bright Cluster Manager versions can be found at `https://support.brightcomputing.com/feature-matrix/`, in the section of the `Feature` column dedicated to machine learning.

1.2 Available Packages

An updated list of the machine learning packages available for the various Bright Cluster Manager versions can be found at `https://support.brightcomputing.com/packages-dashboard/`. Most of the machine learning packages are to be found within the `ml` group. However, some of them are found within the `cm` group for legacy reasons.

At the time of writing, December 2021, the following packages were available:

Table 1.3: Machine Learning packages provided by the Bright Cluster Manager repositories

Package name	Description
`cm-chainer-py39-cuda11.2-gcc9`	A flexible framework for neural networks, designed to write complex architectures simply and intuitively.
`cm-cub-cuda11.2`	A flexible library of cooperative threadblock primitives and other utilities for CUDA kernel programming.
`cm-fastai2-py39-cuda11.2-gcc9`	A library that simplifies training fast and accurate neural nets using modern best practices.
`cm-gpytorch-py39-cuda11.2-gcc9`	A Gaussian process library implemented using PyTorch.
`cm-horovod-pytorch-py39-cuda11.2-gcc9`	A distributed training framework for TensorFlow, Keras, PyTorch, and MXNet.
`cm-horovod-tensorflow2-py39-cuda11.2-gcc9`	A distributed training framework for TensorFlow, Keras, PyTorch, and MXNet.
`cm-ml-distdeps-cuda11.2`	Meta-package containing distribution-specific dependencies for machine learning frameworks.
`cm-ml-pythondeps-py39-cuda11.2-gcc9`	Meta-package containing Python dependencies for machine learning frameworks. It includes, for example, NumPy, PyCUDA, pandas, NetworkX, pydot and Matplotlib.
`cm-nccl2-cuda11.2-gcc9`	Version 2 of NCCL, an implementation of multi-GPU and multi-node collective communication primitives that are performance optimized for NVIDIA GPUs.
`cm-onnx-pytorch-py39-cuda11.2-gcc9`	An open format built to represent machine learning models.
`cm-opencv4-py39-cuda11.2-gcc9`	An open-source BSD-licensed library that includes several hundreds of computer vision algorithms.
`cm-protobuf3-gcc9`	Version 3 of Protocol Buffers, a language-neutral, platform-neutral extensible mechanism for serializing structured data.
`cm-pytorch-extra-py39-cuda11.2-gcc9`	A collection of models, libraries, dataset and useful extra functionality for PyTorch.

...continues

Table 1.3: Machine Learning Packages Included...continued

Package name	Description
`cm-pytorch-py39-cuda11.2-gcc9`	An optimized tensor library for deep learning using GPUs and CPUs. It provides tensor computation (like NumPy) with strong GPU acceleration, and deep neural networks built on a tape-based autograd system. It now includes Caffe2.
`cm-tensorflow2-extra-py39-cuda11.2-gcc9`	A collection of models, libraries, dataset and useful extra functionality for TensorFlow 2.
`cm-tensorflow2-py39-cuda11.2-gcc9`	An open source software library for numerical computation using data flow graphs originally developed by researchers and engineers working on the Google Brain team.
`cm-xgboost-py39-cuda11.2-gcc9`	An optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the gradient boosting framework.

Legend:

Package are available for every distribution unless otherwise tagged.

1.2.1 Considerations

There are some considerations that the cluster administrator should be aware of with the packages.

- Some packages may be labelled in the table 1.3 as deprecated. "Deprecated" in the software industry is not a well-defined term. Here it is used by Bright Computing to mean a package that may no longer be offered in a future release, or for which a newer existing version is preferred.
- Several different packages may be provided for the same machine learning library or framework. For example, TensorFlow may be provided by:
  - `cm-tensorflow-py36-cuda10.1-gcc` or
  - `cm-tensorflow2-py39-cuda11.2-gcc9`

  As is the norm with other package management systems in the software industry, the name given to a Bright Computing package includes the most relevant dependencies required to build and use it. The dependencies commonly highlighted in this manner are:
  - Python interpreter version used (e.g. `*-py36*`, `*-py37*` and `*-py39*`)
  - accelerator library used (e.g. `*-cuda10.1*`, `*-cuda10.2*`, `*-cuda11.2*` and `*-mkl*`)
  - compiler used (e.g. `*-gcc*`, `*-gcc8*` and `*-gcc9*`)

  The availability of different variants of the same package makes it easier for administrators to set up a working environment that is suited to their needs.
- Machine learning packages are designed to coexist, and can therefore all be installed at the same time. This also applies to different variants of the same library or framework.

  This means that administrators can install different versions of the same machine learning library or framework, simply by using different variants. For example: an older `*-py36*` version of TensorFlow, as well as a more recent `*-py37*` version of TensorFlow, can both be installed at the same time.
- As is done with other packages provided by Bright Computing, the updates released for machine learning libraries and frameworks generally leave their major versions unchanged.

  Whenever a major version for a third party machine learning library or a framework is publicly released, a new package or a set of packages is typically placed in the repository.

  Such package(s) imply or contain a reference to the major version in the name. For example:

   - `cm-tensorflow-*` is the name used for TensorFlow major version 1
   - `cm-tensorflow2-*` is the name for TensorFlow major version 2

  As a result, administrators can safely upgrade cluster packages without breaking backward compatibility with users' applications.

- MPI modules and libraries should not be blindly added by the cluster administrator. During module loading, warnings are typically given to suggest an MPI library (Open MPI, or an MPICH or MVAPICH implementation of Open MPI) is required. However, the exact implementation of the MPI library that can be used depends upon the hardware (GPU, interface, architecture) used and requires judgment of suitability by the cluster administrator. Bright Cluster Manager uses the `cm-openmpi4-cuda10.2-ofed50-gcc8` package in this manual as the reference MPI library implementation. This driver package corresponds with using Open MPI with Gigabit Ethernet networking, InfiniBand networking, and NVIDIA GPUs.

1.3 Requirements

The following requirements must be met before installing the preceding machine learning packages.

- RHEL users must have access to the YUM repositories and EPEL repository.
- There must be enough free space for the packages that are installed on the head node and compute nodes. The actual amount depends on the packages installed.
- 8 GB of RAM on the nodes is the minimum recommended amount.
- In order to use packages built with the CUDA toolkit accelerator library version 10.2 and below (e.g. `*-cuda10.2*`), the NVIDIA GPUs must be Maxwell or more recent, with compute capability 3.5 or later. CUDA compute capability 6.0 or later is recommended.
- In order to use packages built with the CUDA toolkit accelerator library version 11.0 and above (e.g. `*-cuda11.2*`), the NVIDIA GPUs must be Maxwell or more recent, with compute capability 5.2 or later. CUDA compute capability 6.0 or later is recommended.
- In order to use packages built using CUDA as the accelerator (i.e. `*-cuda*`), the CPU must support the AVX/AVX2, FMA, and SSE4.2 instructions. This can be checked by inspecting the CPU flags:

  Example

  ```
  [root@node ~]# egrep -m1 -o '(avx|avx2|fma|sse4_2)' /proc/cpuinfo
  fma
  sse4_2
  avx
  avx2
  ```

- In order to use packages built using MKL as the accelerator (i.e. `*-mkl`), the CPU must support the AVX-512 Vector Neural Network Instructions (VNNI). Examples of such CPUs are Intel Xeon Scalable processors with Deep Learning Boost.

1.4 Machine Learning Packages Installation

Head Node Installation

Bright Cluster Manager machine learning packages are installed in the `/cm/shared` directory, which is by default exported over NFS. Packages installed on the head node are therefore also available to all the compute nodes by default.

The .rpm and .deb files have proper dependencies defined. This means that the cluster administrator does not need to spend time figuring out what needs to be installed to set up a working environment. Whenever a package is installed or updated, the required dependencies will be also automatically fetched, if necessary. As a result, packages can be installed with the usual package manager that is provided by the Linux distribution in the usual way (page 535 of the *Administrator Manual*).

For example, the administrator can install cm-pytorch-py37-cuda10.2-gcc8 as follows:

Example

```
[root@bright92 ~]# yum install cm-pytorch-py37-cuda10.2-gcc8       #on RHEL 7 and 8

[root@bright92 ~]# zypper install cm-pytorch-py37-cuda10.2-gcc8    #on SLES 15

[root@bright92 ~]# apt-get install cm-pytorch-py37-cuda10.2-gcc8   #on Ubuntu 18.04
```

The package managers also automatically install the corresponding dependencies, such as

- cm-ml-distdeps-cuda10.2
- cm-ml-pythondeps-py37-cuda10.2-gcc8
- cm-protobuf3-gcc8
- cuda10.2-toolkit
- cm-cudnn7.6-cuda10.2
- cm-nccl2-cuda10.2-gcc8

Machine learning packages share several dependencies, usually providing useful Python or system libraries. For convenience, these dependencies are grouped in the cm-ml-pythondeps-* and cm-ml-distdeps-* meta-packages.

- cm-ml-pythondeps-*: This meta-package provides the application libraries such as numba, numpy, scikit-learn, and scipy.
  - If RHEL8 is being run, and cm-ml-pythondeps-py37-cuda11.2-gcc8 is being installed, then the codeready-builder-for-rhel-8-x86_64-rpms repository needs to be enabled. For example using:
    subscription-manager repos --enable=codeready-builder-for-rhel-8-x86_64-rpms
- cm-ml-distdeps-*: This meta-package, on the other hand, provides development libraries such as blas-devel, libjpeg-devel and libpng-devel, and the utility library gnuplot.

The appropriate meta-packages are automatically installed whenever a package installation requires it.

Administrators only need to make sure that their clusters meet the preceding hardware requirements listed at the start of section 1.3. If that is not done, then unexpected failures may occur during run time, such as segmentation faults.

Examples of common mistakes are

- using packages requiring CUDA (e.g. cm-pytorch-py37-cuda10.2-gcc8) on clusters without GPUs
- using packages requiring VNNI (e.g. cm-tensorflow-py37-mkl-gcc8) on CPUs not supporting the instruction set

Compute Nodes Installation

The `cm-ml-distdeps-*` meta-packages must be also installed onto all compute nodes that are to run machine learning applications.

For example, if the name of the software image is `gpu-image`, then the administrator can install `cm-ml-distdeps-cuda10.2` on RHEL 7 as follows:

Example

```
[root@bright92 ~]# yum install --installroot=/cm/images/gpu-image cm-ml-distdeps-cuda10.2
```

The preceding command must be applied to all software images that are used to run machine learning applications.

There are equivalents to the `--installroot` option of `yum` for the other distribution package managers.

For SLES the installation command equivalent is:

```
[root@bright92 ~]# zypper --root /cm/images/gpu-image install cm-ml-distdeps-cuda10.2
```

For Ubuntu the installation command equivalent is:

```
[root@bright92 ~]# cm-chroot-sw-img /cm/images/gpu-image
[root@bright92 ~]# apt install cm-ml-distdeps-cuda10.2
[root@bright92 ~]# exit     #get out of chroot
```

Details on using `zypper` and `apt` commands for installation to software images are given on page 535 of the *Administrator Manual*.

The preceding command must be applied to all software images that are used to run machine learning applications.

No automatic install of `cuda-driver` and `cuda-dcgm` since `cm-ml-distdeps-cuda-*` v3.0.0: The `cuda-driver` and `cuda-dcgm` packages used to be automatically installed during installation of earlier versions of the `cm-ml-distdeps-cuda-*` meta-package. This behavior has changed.

Version 3.0.0 onward of the package requires a manual install of the `cuda-driver` and `cuda-dcgm` packages. Installation of the `cuda-driver` and `cuda-dcgm` packages is covered in section 7.4.1 of the *Installation Manual*.

The version number of the available or installed `cm-ml-distdeps-cuda*` package can be found with `yum info`:

Example

```
[root@bright92 ~]# yum info cm-ml-distdeps-cuda11.2
...
Available Packages
Name         : cm-ml-distdeps-cuda11.2
Version      : 2.7.2
...
```

The manual installation of the `cuda-driver` and `cuda-dcgm` packages allows a wider range of NVIDIA drivers and cluster configurations to be installed. Version 3.0.0 onward now allows the administrator to install custom NVIDIA drivers, for example for special hardware such as DGX machines. It also allows the administrator to install different versions of NVIDIA drivers for different groups of compute nodes.

Just as for the `cm-ml-distdeps-*` meta-packages, the custom NVIDIA drivers must be installed onto all the compute nodes that are to run machine learning applications.

1.4.1 Module Loading

Bright Cluster Manager provides environment module definitions for all the machine learning packages. The environment module files are also compatible with the Lmod software introduced in Bright Cluster Manager 7.3. They can be listed once the `shared` module is loaded, if it has not already been loaded:

```
[root@bright92 ~]# module purge; module available
---------------------------------- /cm/local/modulefiles ----------------------------------
boost/1.74.0             cm-setup/master     cuda-dcgm/2.0.15.1  lua/5.4.0     null
cluster-tools/master     cmake-gcc8/3.18.4   dot                 luajit        openldap
cm-bios-tools            cmd                 freeipmi/1.6.6      mariadb-libs  python3
cm-image/master          cmjob               gcc/10.2.0          module-git    python37
cm-scale/cm-scale.module cmsh                ipmitool/1.8.18     module-info   shared
[root@bright92 ~]# module load shared; module available
---------------------------------- /cm/local/modulefiles ----------------------------------
boost/1.74.0             cm-setup/master     cuda-dcgm/2.0.15.1  lua/5.4.0     null
cluster-tools/master     cmake-gcc8/3.18.4   dot                 luajit        openldap
cm-bios-tools            cmd                 freeipmi/1.6.6      mariadb-libs  python3
cm-image/master          cmjob               gcc/10.2.0          module-git    python37
cm-scale/cm-scale.module cmsh                ipmitool/1.8.18     module-info   shared

--------------------------------- /cm/shared/modulefiles ----------------------------------
blacs/openmpi/gcc/64/1.1patch03    hwloc/1.11.11
blas/gcc/64/3.8.0                  intel-tbb-oss/ia32/2020.3
bonnie++/1.98                      intel-tbb-oss/intel64/2020.3
cm-pmix3/3.1.4                     iozone/3_490
cuda10.2/blas/10.2.89              lapack/gcc/64/3.9.0
cuda10.2/fft/10.2.89               ml-pythondeps-py37-cuda10.2-gcc8/4.4.2
cuda10.2/toolkit/10.2.89           mpich/ge/gcc/64/3.3.2
cudnn7.6-cuda10.2/7.6.5.32         mvapich2/gcc/64/2.3.4
default-environment                nccl2-cuda10.2-gcc8/2.8.4
fftw2/openmpi/gcc/64/double/2.1.5  netcdf/gcc/64/gcc/64/4.7.3
fftw2/openmpi/gcc/64/float/2.1.5   netperf/2.7.0
fftw3/openmpi/gcc/64/3.3.8         openblas/dynamic/(default)
gcc8/8.4.0                         openblas/dynamic/0.3.7
gdb/9.2                            openmpi/gcc/64/1.10.7
globalarrays/openmpi/gcc/64/5.7    protobuf3-gcc8/3.10.1
hdf5/1.10.1                        scalapack/openmpi/gcc/2.1.0
hdf5_18/1.8.21                     tensorflow2-py37-cuda10.2-gcc8/2.4.1
hpl/2.3                            ucx/1.8.1
```

For example, after having installed the `cm-tensorflow2-py37-cuda10.2-gcc8` package, the associated TensorFlow module can be loaded with:

```
[root@bright92 ~]# module load tensorflow2-py37-cuda10.2-gcc8
Loading tensorflow2-py37-cuda10.2-gcc8/2.4.1
  Loading requirement: openblas/dynamic/0.3.7 hdf5_18/1.8.21 gcc8/8.4.0 python37
    cuda10.2/toolkit/10.2.89 cudnn7.6-cuda10.2/7.6.5.32
    ml-pythondeps-py37-cuda10.2-gcc8/4.4.2 protobuf3-gcc8/3.10.1 nccl2-cuda10.2-gcc8/2.8.4
```

The machine learning environment modules automatically load additional environment modules as dependencies, with the notable exception of Open MPI implementations for the reasons given in section 1.2.1.

The module dependencies are achieved via the module definition files:

Example

```
[root@bright92 ~]# module show tensorflow2-py37-cuda10.2-gcc8
-------------------------------------------------------------------
/cm/shared/modulefiles/tensorflow2-py37-cuda10.2-gcc8/2.4.1:

module-whatis   adds TensorFlow2 to your environment variables
module          load ml-pythondeps-py37-cuda10.2-gcc8
module          load protobuf3-gcc8
module          load cudnn7.6-cuda10.2
module          load nccl2-cuda10.2-gcc8
prepend-path    PYTHONPATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/lib/python3.7/\
site-packages/
prepend-path    PYTHONPATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/lib64/python3.7/\
site-packages/
prepend-path    LD_LIBRARY_PATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/lib/python3.7/\
site-packages/tensorflow
prepend-path    LD_LIBRARY_PATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/lib/python3.7/\
site-packages/tensorflow_core
prepend-path    LD_LIBRARY_PATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/lib64/python3.7/\
site-packages/tensorflow
prepend-path    LD_LIBRARY_PATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/lib64/python3.7/\
site-packages/tensorflow_core
prepend-path    PATH /cm/shared/apps/tensorflow2-py37-cuda10.2-gcc8/2.4.1/bin
-------------------------------------------------------------------
```

1.5 Machine Learning Packages Removal

Machine learning packages can be removed in the usual way with the package manager commands used by the Linux distribution. For example, the administrator can remove the `cm-pytorch-py37-cuda10.2-gcc8` package with:

Example

```
[root@bright92 ~]# yum remove cm-pytorch-py37-cuda10.2-gcc8       #on RHEL 7 and 8
```

or

```
[root@bright92 ~]# zypper remove cm-pytorch-py37-cuda10.2-gcc8    #on SLES 15
```

or

```
[root@bright92 ~]# apt-get remove cm-pytorch-py37-cuda10.2-gcc8   #on Ubuntu 18.04
```

Bright Cluster Manager machine learning packages are installed in the `/cm/shared` directory, which is by default exported over NFS. Packages removed from the head node are therefore also removed from all the compute nodes by default.

The `cm-ml-distdeps-*` meta-packages must be also removed from all compute nodes that are to run machine learning applications. For example, if the name of the software image is `gpu-image`, then the images directory is `/cm/images/gpu-image`, and then the administrator can remove `cm-ml-distdeps-cuda10.2` from the image as follows in RHEL 7:

Example

```
[root@bright92 ~]# yum remove --installroot=/cm/images/gpu-image cm-ml-distdeps-cuda10.2
```

The preceding command must be applied to all software images that are used to run the machine learning applications.

The equivalents to the `--installroot` option of `yum` for the other distribution package managers are described in section 1.4.

2

Running TensorFlow

This chapter goes through some example runs with TensorFlow. Some output messages have been removed or simplified in the runs for readability.

The sample runs assume that TensorFlow and its extra libraries have been installed from the Bright Cluster Manager repository with a package manager. For example with

```
yum install cm-tensorflow2-py37-cuda10.2-gcc8 cm-tensorflow2-extra-py37-cuda10.2-gcc8
```

In addition to requiring the extra libraries, TensorFlow requires an OpenMPI implementation to work. Chapter 3 of the *User Manual* describes the different OpenMPI packages that the Bright repositories provide. The different OpenMPI packages allow the user to choose which one to use. For example, depending on which interconnect is being used, or depending on if CUDA support is required.

In this chapter, the `cm-openmpi4-cuda10.2-ofed50-gcc8` package is used.

More information on the examples can be found at `https://github.com/tensorflow/examples`.

2.1 Hello World

A “`Hello World`” interactive example that just shows that the software is in place for TensorFlow can be run as follows:

Example

```
[root@bright92 ~]# module load shared
[root@bright92 ~]# module load tensorflow2-py37-cuda10.2-gcc8
[root@bright92 ~]# module load openmpi4-cuda10.2-ofed50-gcc8
[root@bright92 ~]# python3.7
Python 3.7.9 (default, Mar  1 2021, 14:10:18)
[GCC 10.2.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import tensorflow as tf
<> I <> Successfully opened dynamic library libcudart.so.10.2

>>> hello = tf.constant('TensorFlow 2 Hello World')
<> I <> Successfully opened dynamic library libcuda.so.1
<> I <> Found device 0 with properties:
 pciBusID: 0000:00:08.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
 coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB deviceMemoryBandwidth: 681.88GiB/s
<> I <> Successfully opened dynamic library libcuda.so.1
<> I <> Successfully opened dynamic library libcudart.so.10.2
<> I <> Successfully opened dynamic library libcublas.so.10
<> I <> Successfully opened dynamic library libcublasLt.so.10
```

```
<> I <> Successfully opened dynamic library libcufft.so.10
<> I <> Successfully opened dynamic library libcurand.so.10
<> I <> Successfully opened dynamic library libcusolver.so.10
<> I <> Successfully opened dynamic library libcusparse.so.10
<> I <> Successfully opened dynamic library libcudnn.so.7
<> I <> Adding visible gpu devices: 0
<> I <> Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 14975 MB memory) ->
 physical GPU (device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)

>>> tf.print(hello)
TensorFlow 2 Hello World

>>> a = tf.constant(10)
>>> b = tf.constant(32)
>>> tf.print(a+b)
42
>>>
```

2.2 Deep Convolutional Generative Adversarial Network

The following trains a deep convolutional generative adversarial network (DCGAN).

The example code is included in the TensorFlow extra package. Once its module has been loaded, the example directory is defined with the `CM_TENSORFLOW2_EXTRA` environment variable.

The example picks up training images and labels from the MNIST site, and places them in a directory `tensorflow_datasets/` if it needs to. The images are then used to train the model. End users would be expected to train the neural network within their home directories.

Example

```
[root@bright92 ~]# module load tensorflow2-extra-py37-cuda10.2-gcc8
[root@bright92 ~]# module load openmpi4-cuda10.2-ofed50-gcc8
[root@bright92 ~]# cd ${CM_TENSORFLOW2_EXTRA}/tensorflow_examples/models/dcgan/
[root@bright92 dcgan]# python3.7 dcgan.py --epochs 5
<> I <> Successfully opened dynamic library libcudart.so.10.2
<> I <> Generating dataset mnist (/root/tensorflow_datasets/mnist/3.0.1) Downloading and preparing
dataset Unknown size (download: Unknown size, generated: Unknown size, total: Unknown size) to
/root/tensorflow_datasets/mnist/3.0.1...
<> I <> Downloading https://storage.googleapis.com/cvdf-datasets/mnist/t10k-images-idx3-ubyte.gz into
/root/tensorflow_datasets/downloads/cvdf-datasets_mnist_t10k-images-idx3-<>.gz.tmp.<>...
<> I <> Downloading https://storage.googleapis.com/cvdf-datasets/mnist/t10k-labels-idx1-ubyte.gz into
/root/tensorflow_datasets/downloads/cvdf-datasets_mnist_t10k-labels-idx1-<>.gz.tmp.<>...
<> I <> Downloading https://storage.googleapis.com/cvdf-datasets/mnist/train-images-idx3-ubyte.gz into
/root/tensorflow_datasets/downloads/cvdf-datasets_mnist_train-images-idx3-<>.gz.tmp.<>...
<> I <> Downloading https://storage.googleapis.com/cvdf-datasets/mnist/train-labels-idx1-ubyte.gz into
/root/tensorflow_datasets/downloads/cvdf-datasets_mnist_train-labels-idx1-<>.gz.tmp.<>...
Dl Completed...: 100% [==============================] 4/4 [00:00<00:00, 21.68 url/s]
Extraction completed...: 100% [==============================] 4/4 [00:00<00:00,  5.95 file/s]
Extraction completed...: 100% [==============================] 4/4 [00:00<00:00, 11.83 file/s]
Dl Size...: 100% [==============================] 10/10 [00:00<00:00, 14.87 MiB/s]
Dl Completed...: 100% [==============================] 4/4 [00:00<00:00,  5.94 url/s]
Generating splits...:   0% [                              ]
<> I <> Found device 0 with properties:
 pciBusID: 0000:00:08.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
 coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB deviceMemoryBandwidth: 681.88GiB/s
<> I <> Successfully opened dynamic library libcuda.so.1
<> I <> Successfully opened dynamic library libcudart.so.10.2
```

```
<> I <> Successfully opened dynamic library libcublas.so.10
<> I <> Successfully opened dynamic library libcublasLt.so.10
<> I <> Successfully opened dynamic library libcufft.so.10
<> I <> Successfully opened dynamic library libcurand.so.10
<> I <> Successfully opened dynamic library libcusolver.so.10
<> I <> Successfully opened dynamic library libcusparse.so.10
<> I <> Successfully opened dynamic library libcudnn.so.7
<> I <> Adding visible gpu devices: 0
<> I <> Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 14975 MB memory) ->
 physical GPU (device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
<> I <> Done writing mnist-train.tfrecord. Number of examples: 60000 (shards: [60000])
Generating splits...:  50% [===============               ] 1/2
Done writing mnist-test.tfrecord. Number of examples: 10000 (shards: [10000])
Dataset mnist downloaded and prepared to /root/tensorflow_datasets/mnist/3.0.1.
Subsequent calls will reuse this data.
<> I <> Constructing tf.data.Dataset for split train, from /root/tensorflow_datasets/mnist/3.0.1
Training ...
Epoch 0, Generator loss 1.0254955291748047, Discriminator Loss 0.8931688666343689
Epoch 1, Generator loss 1.170407772064209, Discriminator Loss 0.8915461301803589
Epoch 2, Generator loss 0.9688068628311157, Discriminator Loss 1.1547539234161377
Epoch 3, Generator loss 1.4020311832427979, Discriminator Loss 0.805593729019165
Epoch 4, Generator loss 1.1278810501098633, Discriminator Loss 1.2198209762573242
[root@bright92 dcgan]# rm -rf /root/tensorflow_datasets/
[root@bright92 dcgan]#
```

2.3 Image-to-Image Translation with Conditional Adversarial Nets

The following trains a conditional adversarial networks as a general-purpose solution to image-to-image translation problems. The trained model is capable of completing different tasks, such as colorizing black and white photos.

The example code is included in the TensorFlow extra package. Once its module has been loaded, the example directory is defined with the `CM_TENSORFLOW2_EXTRA` environment variable.

The example uses a preprocessed copy of the CMP Facade Database, helpfully provided by the Center for Machine Perception at the Czech Technical University in Prague. The original dataset includes 606 rectified images of facades from various sources, which have been manually annotated. The facades are from different cities around the world and diverse architectural styles.

The example conveniently downloads the dataset in a temporary directory and then trains the model. End users would be expected to train the neural network within their home directories.

Example

```
[root@bright92 ~]# module load tensorflow2-extra-py37-cuda10.2-gcc8
[root@bright92 ~]# module load openmpi4-cuda10.2-ofed50-gcc8
[root@bright92 ~]# cd ${CM_TENSORFLOW2_EXTRA}/tensorflow_examples/models/pix2pix/
[root@bright92 pix2pix]# python3.7 data_download.py --download_path /tmp
<> I <> Successfully opened dynamic library libcudart.so.10.2
Downloading data from https://people.eecs.berkeley.edu/~tinghuiz/projects/pix2pix/datasets/facades.tar.gz
30171136/30168306 [==============================] - 15s 1us/step
[root@bright92 pix2pix]# python3.7 pix2pix.py --path /tmp/facades --epochs 5
<> I <> Successfully opened dynamic library libcudart.so.10.2
<> I <> Found device 0 with properties:
 pciBusID: 0000:00:08.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
 coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB deviceMemoryBandwidth: 681.88GiB/s
<> I <> Successfully opened dynamic library libcuda.so.1
<> I <> Successfully opened dynamic library libcudart.so.10.2
```

```
<> I <> Successfully opened dynamic library libcublas.so.10
<> I <> Successfully opened dynamic library libcublasLt.so.10
<> I <> Successfully opened dynamic library libcufft.so.10
<> I <> Successfully opened dynamic library libcurand.so.10
<> I <> Successfully opened dynamic library libcusolver.so.10
<> I <> Successfully opened dynamic library libcusparse.so.10
<> I <> Successfully opened dynamic library libcudnn.so.7
<> I <> Adding visible gpu devices: 0
<> I <> Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 14975 MB memory) ->
 physical GPU (device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
Epoch 0, Generator loss 33.75794219970703, Discriminator Loss 1.3662723302841187
Epoch 1, Generator loss 32.59027862548828, Discriminator Loss 0.5289707183837891
Epoch 2, Generator loss 36.31702423095703, Discriminator Loss 1.6457607746124268
Epoch 3, Generator loss 38.491973876953125, Discriminator Loss 0.43443232774734497
Epoch 4, Generator loss 45.60129928588867, Discriminator Loss 0.5746432542800903
[root@bright92 pix2pix]#
```

2.4 Neural Machine Translation With Attention

The following trains a sequence to sequence model for neural machine translation with attention using gated recurrent units (GRUs).

The example code is included in the TensorFlow extra package. Once its module has been loaded, the example directory is defined with the `CM_TENSORFLOW2_EXTRA` environment variable.

The example conveniently downloads a dataset for Spanish to English translation in a temporary directory and then trains the model. End users would be expected to train the neural network within their home directories.

Example

```
[root@bright92 ~]# module load tensorflow2-extra-py37-cuda10.2-gcc8
[root@bright92 ~]# module load openmpi4-cuda10.2-ofed50-gcc8
[root@bright92 ~]# cd ${CM_TENSORFLOW2_EXTRA}/tensorflow_examples/models/nmt_with_attention/
[root@bright92 nmt_with_attention]# python3.7 train.py --epochs 5 --download_path /tmp
<> I <> Successfully opened dynamic library libcudart.so.10.2
Downloading data from http://storage.googleapis.com/download.tensorflow.org/data/spa-eng.zip
2646016/2638744 [==============================] - 0s 0us/step
<> I <> Found device 0 with properties:
 pciBusID: 0000:00:08.0 name: Tesla P100-PCIE-16GB computeCapability: 6.0
 coreClock: 1.3285GHz coreCount: 56 deviceMemorySize: 15.90GiB deviceMemoryBandwidth: 681.88GiB/s
<> I <> Successfully opened dynamic library libcuda.so.1
<> I <> Successfully opened dynamic library libcudart.so.10.2
<> I <> Successfully opened dynamic library libcublas.so.10
<> I <> Successfully opened dynamic library libcublasLt.so.10
<> I <> Successfully opened dynamic library libcufft.so.10
<> I <> Successfully opened dynamic library libcurand.so.10
<> I <> Successfully opened dynamic library libcusolver.so.10
<> I <> Successfully opened dynamic library libcusparse.so.10
<> I <> Successfully opened dynamic library libcudnn.so.7
<> I <> Adding visible gpu devices: 0
<> I <> Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 14975 MB memory) ->
 physical GPU (device: 0, name: Tesla P100-PCIE-16GB, pci bus id: 0000:00:08.0, compute capability: 6.0)
Epoch: 0, Train Loss: 1.7577663660049438, Test Loss: 2.344970941543579
Epoch: 1, Train Loss: 0.9539323449134827, Test Loss: 1.8177989721298218
Epoch: 2, Train Loss: 0.5993071794509888, Test Loss: 1.6585650444030762
Epoch: 3, Train Loss: 0.4077189564704895, Test Loss: 1.6183968782424927
Epoch: 4, Train Loss: 0.28957614302635193, Test Loss: 1.6796191930770874
```

```
[root@bright92 nmt_with_attention]#
```

3

Running PyTorch

This chapter goes through some example runs with PyTorch. Some output messages have been removed or simplified in the runs for readability.

The sample runs assume that PyTorch and its extra libraries have been installed from the Bright Cluster Manager repository with a package manager. For example with:

```
yum install cm-pytorch-py37-cuda10.2-gcc8 cm-pytorch-extra-py37-cuda10.2-gcc8
```

In addition to requiring the extra libraries, PyTorch requires an OpenMPI implementation to work. Chapter 3 of the *User Manual* describes the different OpenMPI packages that the Bright repositories provide. The different OpenMPI packages allow the user to choose which one to use. For example, depending on which interconnect is being used, or depending on if CUDA support is required.

In this chapter, the `cm-openmpi4-cuda10.2-ofed50-gcc8` package is used.

More information on the examples can be found at `https://github.com/pytorch/examples`.

3.1 Variational Autoencoders

The following example shows how to train *Variational Autoencoders*[1], powerful generative models that can be used for a wide variety of applications.

The example code is included in the PyTorch extra (`cm-pytorch-extra-*`) package. Once its module has been loaded, the example directory is defined with the `CM_PYTORCH_EXTRA` environment variable.

Example

```
[root@bright92 ~]# module load shared
[root@bright92 ~]# module load pytorch-extra-py37-cuda10.2-gcc8
[root@bright92 ~]# module load openmpi4-cuda10.2-ofed50-gcc8
[root@bright92 ~]# cd ${CM_PYTORCH_EXTRA}/vae
```

The Variational Autoencoders network is trained by default for 10 epochs. The required dataset can automatically be downloaded and extracted with:

Example

```
[root@bright92 vae]# python3.7 main.py
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz to
../data/MNIST/raw/train-images-idx3-ubyte.gz
9920512it [00:04, 2041116.37it/s]
Extracting ../data/MNIST/raw/train-images-idx3-ubyte.gz to ../data/MNIST/raw
```

[1] Original paper: "`Auto-Encoding Variational Bayes`" by Diederik P Kingma and Max Welling: `https://arxiv.org/abs/1312.6114`.

```
...
Processing...
Done!
Train Epoch: 1 [0/60000 (0%)] Loss: 550.187805
Train Epoch: 1 [1280/60000 (2%)] Loss: 323.104736
Train Epoch: 1 [2560/60000 (4%)] Loss: 237.460938
...
Train Epoch: 1 [58880/60000 (98%)] Loss: 130.540909
====> Epoch: 1 Average loss: 164.1742
====> Test set loss: 127.8219
Train Epoch: 2 [0/60000 (0%)] Loss: 127.949753
...
Train Epoch: 10 [58880/60000 (98%)] Loss: 107.980888
====> Epoch: 10 Average loss: 106.1472
====> Test set loss: 105.8715
```

The output sampled digits can be found in the `results` directory:

```
[root@bright92 vae]# ls results/
reconstruction_10.png  reconstruction_4.png  reconstruction_8.png  sample_2.png  sample_6.png
reconstruction_1.png   reconstruction_5.png  reconstruction_9.png  sample_3.png  sample_7.png
reconstruction_2.png   reconstruction_6.png  sample_10.png         sample_4.png  sample_8.png
reconstruction_3.png   reconstruction_7.png  sample_1.png          sample_5.png  sample_9.png
[root@bright92 vae]#
```