CS™ Series Urika®-CS AI and Analytics Installation Guide

(1.0.UP00)

S-3023 Rev A

Contents

1 About the CS™ Series Urika®-CS AI and Analytics Installation Guide Rev A

The *CS™ Series Urika®-CS AI and Analytics Installation Guide Rev A* includes procedures for installing the Cray Urika-CS software on Cray CS systems. It provides an overview of the Urika-CS software components, installation related troubleshooting and quick reference information, as well as post installation verification tests.

Table 1. Record of Revision

Publication Title	Date	Release
CS™ Series Urika®-CS AI and Analytics Installation Guide	July 2018	Urika-CS 1.0UP00
CS™ Series Urika®-CS AI and Analytics Installation Guide Rev A	August 2018	Urika-CS 1.0UP00

This revision include updates to the topics *Install Urika-CS on a BCM Based System* on page 16 and *Install Urika-CS on an ACE Based System* on page 11 and include changes to installation paths.

Scope and Audience

This publication is written for system administrators of the Cray CS systems.

Typographic Conventions

`Monospace`	Indicates program code, reserved words, library functions, command-line prompts, screen output, file/path names, and other software constructs.
`Monospaced Bold`	Indicates commands that must be entered on a command line or in response to an interactive prompt.
`Oblique` or *`Italics`*	Indicates user-supplied values in commands or syntax definitions.
Proportional Bold	Indicates a **GUI Window**, **GUI element**, cascading menu (**Ctrl**→**Alt**→**Delete**), or key strokes (press **Enter**).
\ (backslash)	At the end of a command line, indicates the Linux® shell line continuation character (lines joined by a backslash are parsed as a single line).

Trademarks

The following are trademarks of Cray Inc. and are registered in the United States and other countries: CRAY and design, SONEXION, Urika-GX, and YARCDATA. The following are trademarks of Cray Inc.: APPRENTICE2, CHAPEL, CLUSTER CONNECT, ClusterStor, CRAYDOC, CRAYPAT, CRAYPORT, DATAWARP, ECOPHLEX, LIBSCI, NODEKARE. The following system family marks, and associated model number marks, are trademarks of Cray Inc.: CS, CX, XC, XE, XK, XMT, and XT. The registered trademark LINUX is used pursuant to a

sublicense from LMI, the exclusive licensee of Linus Torvalds, owner of the mark on a worldwide basis. Other trademarks used in this document are the property of their respective owners.

2 About Urika®-CS

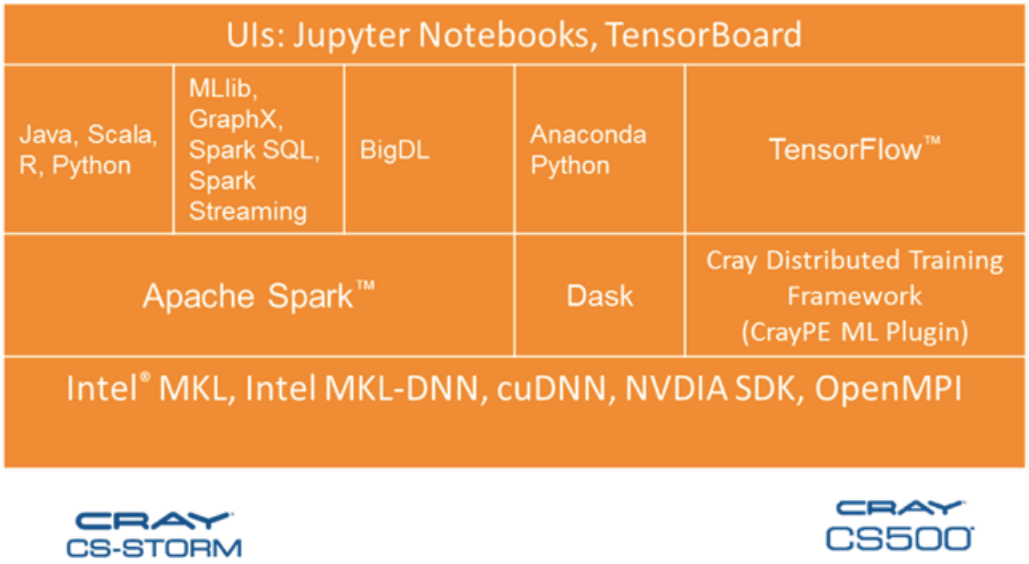
Cray Urika®-CS is an optimized big data software stack, which is tuned for multiple work-flows and runs on Cray CS systems. It features a comprehensive analytics software stack for performing machine and deep learning tasks.

Features and Analytic Components

- **Cray PE ML Plugin** - This portable plugin leverages features of MPI as well as a novel "delayed synchronization" variant of the Stochastic Gradient Descent (SGD) algorithm to allow scaling of deep learning training without the normal convergence penalties. These capabilities improve scale, performance, and ease of use of work loads.
- **Support for Jupyter Notebook** - Jupyter Notebook is a web application that enables creating and sharing documents that contain live code, equations, visualizations, and explanatory text. For more information, visit *http://jupyter.org*
- **Support for GPUs** - Urika®-CS enables running TensorFlow on Nvidia GPU nodes.
- **Open Source Analytics (OSA) Images** - Urika®-CS provides Open Source Analytics (OSA) OSA images that run inside Singularity containers. Software provided in these images includes:
    - **Apache™ Spark™** - Spark is a general data processing framework that simplifies developing big data applications. It provides the means for executing batch, streaming, and interactive analytics jobs. In addition to the core Spark components, Urika®-CS software ships with a number of Spark ecosystem components. For more information, visit *https://spark.apache.org*
    - **Anaconda® Python and R** - Anaconda is a distribution of the Python and R programming languages for large-scale data processing, predictive analytics, and scientific computing. It aims at simplifying package management and deployment. For more information, visit *https://anaconda.org*
    - **Dask and Dask Distributed** - Dask is a parallel programming library that combines with the Numeric Python ecosystem to provide parallel arrays, data-frames, machine learning, and custom algorithms. For more information, visit *http://dask.pydata.org*
    - **Intel® BigDL** - BigDL is a distributed deep learning library for Spark that can run directly on top of existing Spark or Apache Hadoop clusters. Deep learning applications can be written as Scala or Python programs. For more information, visit *https://www.intel.com*
    - **TensorFlow™ and TensorBoard** - TensorFlow is a software library for dataflow programming across a range of tasks. It is a Math library, which is also used for machine learning applications, such as neural networks. TensorFlow provides a utility called TensorBoard that can display an interactive graphic of the computational graph. For more information, visit *https://www.tensorflow.org*

  Urika®-CS's software stack is depicted in the following figure:

Figure 1. Urika®-CS Analytic Software Stack



3 About the Urika-CS Installation Process

The Urika-CS software installation process involves the following tasks that need to be performed in the order listed:

1. Installation of supporting software components. Refer to *Install Supporting Software* on page 9 for instructions.
2. Installation of the Urika-CS and Cray ML PE plugin RPMs. Refer to *Install Urika-CS on an ACE Based System* on page 11 or *Install Urika-CS on a BCM Based System* on page 16 for instructions.
3. Verification of the installation. Refer to *Urika-CS Troubleshooting Information* on page 24 for more information.

Workload Managers and Management Systems

The procedures included in this publication assume that they are being performed on systems that use:

- ACE or BCM for system management
- Slurm or PBS Pro for workload management

Contact Cray Support if other system or workload management tools are being used.

Terminology about Nodes

The node where the cluster is managed from is referred to as '*head node*' in this publication.

General Limitations and Recommendations

- It is recommended to save the Urika-CS RPMs in a temporary location with enough space to accommodate at least 60GBs and cleanup afterwords.
- It is recommended not to change the Urika-CS image directory. If it is required to change the path to the Urika-CS image directory, the `__ANALYTICS_SINGULARITY_IMG_DIR` variable in `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc` must be changed to match.
- Adding new containers or modifying the containers shipped with the Urika-CS software is currently not supported. Similarly, Cray does not currently support customer modification to the Urika-CS release. For more information, please contact a Cray service representative.
- The Cray PE ML plugin currently does not support Intel® Omni-Path.

4 Required Software Components

Urika-CS software needs to be installed on login nodes and compute nodes images.

Required RPMs

The Urika-CS installation process requires the following RPMs, which can be downloaded from CrayPort. Refer to *https://portal.cray.com/* to register with CrayPort or contact a Cray Support person to set up a CrayPort account.

RPM containing Singularity images

```
cray-analytics-singularity-images-1.01.0000.201807301732_0117-1.0101.0117.f9f2519.x86_64.rpm
```

RPMs containing the Cray PE ML plugin

- `craype-ml-plugin-py2-1.1.1-201806111428.b359b88bdae0e.el7-0.x86_64.rpm`
- `craype-ml-plugin-py3-1.1.1-201806111428.b359b88bdae0e.el7-0.x86_64.rpm`

Required Supporting Software Components

Urika-CS requires the following supporting software, which can be downloaded from online resources.

Table 2.

Component	Required Version	Reference
Singularity	2.5 or later	*http://singularity.lbl.gov/docs-installation*
OpenMPI	3.0.0 or later	*https://www.open-mpi.org/*
CUDA toolkit	9.0	*https://docs.nvidia.com/*
CuDNN	7.1.2	*https://docs.nvidia.com/*

5 Install Supporting Software

Prerequisites

- This procedure requires root privileges.
- Software required for this procedure includes:
  - Singularity, version 2.5 or later
  - OpenMPI, version 3.0.0 or later
  - CUDA toolkit, version 9.0
  - CuDNN, version 7.1.2

About this task

This procedure provides information about the supporting software components that need to be installed on the Cray CS system before installing Urika-CS software.

> **NOTE:** Each of the supporting software components listed in this procedure needs to be available/accessible from every node in the system.

Procedure

1. Install Singularity.

   Refer to *http://singularity.lbl.gov/docs-installation* for installation instructions and refer to *Troubleshoot Singularity Related Issues* on page 24 for troubleshooting information.

2. Install OpenMPI.

   Refer to *https://www.open-mpi.org/* for installation instructions.

   The OpenMPI libraries should be built as a shared library. Runtime failures have been noticed if the `--enable-static` is used. These flags should not be used during the build. The following build flags have been verified for Urika-CS's installation:

   - `--with-slurm`
   - `--with-pmi=`*`pmi_libs_paths`*

   The fabric driver libraries, as well as any other dependencies of `libmpi.so` are located under `/lib64`.

   While installing OpenMPI, use the `--with-tm` option on the command line to ensure that the OpenMPI components required for Torque/PBS support are built. For more information, execute `./configure --help`, or refer to *https://www.open-mpi.org*

   Refer to *Resolve I/O Related Issues while Using the Cray PE ML Plugin* for troubleshooting information.

3. Skip this step if the system does not use GPUs. If using a CS-Storm system with Nvidia GPUs, install CuDNN, Cuda toolkit and Nvidia drivers if the system does not have these software components installed already.
   Install CUDA toolkit (9.0) and CuDNN (7.1.2) on the GPU compute nodes. Refer to *https://docs.nvidia.com/* for installation instructions.

   Refer to *Troubleshoot NVIDIA, CUDA and CuDNN Related Issues* on page 27 to verify that Nvidia, CUDA, and CuDNN are properly installed and configured properly.

   In addition to the preceding list of software, Urika-CS uses `patchelf` to search for MPI dependent libraries. If it is not pre-installed, Urika-CS will download `patchelf` during runtime. If its not possible to invoke `wget` at runtime due to connectivity issues, install `patchelf` manually and include it in the `PATH`.

6 Install Urika-CS on an ACE Based System

Prerequisites

This procedure:

- requires root privileges
- assumes that the installation is being done on an ACE based Cray CS system that supports Singularity and OpenMPI
- assumes Slurm or PBS Pro is being used as the workload manager. Please contact Cray Support if using other workload managers.
- assumes that the supporting software components listed in *Install Supporting Software* on page 9 have been installed on the system.

About this task

Use the following instructions to install and configure Urika-CS software. Major software components that are installed during this process include:

- Urika-CS RPMs
- Cray PE ML plugin

Procedure

1. Log on to the head node as root.
2. Install Urika-CS software.

   Urika-CS is packaged as RPMs. The Urika-CS package includes:

   - Urika-CS RPM, which includes the Singularity image, user scripts, man pages, and the Urika-CS module file.
   - Cray PE ML plugin RPMs:
     - Cray PE ML plugin Py2 RPM
     - Cray PE ML plugin Py3 RPM

   Urika-CS image, scripts, and the Cray ML PE plugin need to be installed and available on all the nodes.

```
# rpm -ivh \
/opt/cray/analytics-singularity-images/1.01.201807301732_0117/analytics-urikacs-1.0.0000-latest.simg
/opt/cray/analytics/1.01.201807301732_0117/bin
/opt/cray/analytics/1.01.201807301732_0117/bin/analytics.shrc
/opt/cray/analytics/1.01.201807301732_0117/bin/configure.shrc
/opt/cray/analytics/1.01.201807301732_0117/bin/run_training
/opt/cray/analytics/1.01.201807301732_0117/bin/start_analytics
/opt/cray/analytics/1.01.201807301732_0117/man
/opt/cray/analytics/1.01.201807301732_0117/man/man1
/opt/cray/analytics/1.01.201807301732_0117/man/man1/run_training.1
```

```
/opt/cray/analytics/1.01.201807301732_0117/man/man1/start_analytics.1
/opt/cray/modulefiles/analytics/1.01.20180730
```

3. Install Cray PE ML plugin (version 1.1.1) and set the path in the Singularity configuration.

   The plugin is installed by default under `/opt/cray/pe`.

```
# rpm -ivh --nodeps craype-ml-plugin-py3-1.1.1-201803051608.6614ddecd7024.el7-1.x86_64.rpm
# rpm -ivh --nodeps craype-ml-plugin-py2-1.1.1-201803051608.6614ddecd7024.el7-1.x86_64.rpm
```

4. Skip this step if the system uses modules as a way to set the right environment. If the CS machine does not use modulefiles or modules, edit the `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc` file to make the following changes:

   a. Set `MODULES_EXIST=no`

   b. Set the value of the `__ANALYTICS_SINGULARITY_IMG_DIR` variable to the path to the Urika-CS image location:

```
# export __ANALYTICS_SINGULARITY_IMG_DIR=\
/opt/cray/analytics-singularity-images/1.01.0000.201806211250_0111/
```

   c. Uncomment and set the following variables without changing the values of the variables.

```
#export __ANALYTICS_CONTAINER_NAME=urikacs
#export ANALYTICS_IMG=analytics-urikacs-1.0.0000-latest.simg
```

5. Configure Urika-CS by setting the environmental variables needed for Urika-CS runtime in the `configure.shrc` script, which is located under `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc`.

   Variables to configure include:

   - The `OMPI_DIR` and `OPAL_PREFIX` variables are required to run the Cray PE ML plugin.
   - The network interface available on the compute nodes can be set using `URIKACS_DEV_INF` variable. Urika-CS by default assumes InfiniBand is running. If it is not running, Urika-CS uses the available Ethernet interface. If this assumption is not true then the admin can set the `URIKACS_DEV_INF` variable.
   - A default `SPARK_EVENT_DIR` is used to log Spark History. This can be changed by setting `SPARK_EVENT_DIR` variable.

   A sample `configure.shrc` configuration file for an ACE based system is shown below:

```
###### This file has to be configured ######
###### The network interface on compute nodes ######
#export URIKACS_DEV_INF="ib0"
###### The OpenMPI 3.0.0 installation directory #####
export OPAL_PREFIX=/global/opt/openmpi/3.0.0/gcc/
###### The OPAL_PREFIX needed for mpirun #####
export OMPI_DIR=/global/opt/openmpi/3.0.0/gcc/
###### configure SPARK_EVENT_DIR #####
#export SPARK_EVENT_DIR=$HOME/.urikacs/sparkHistory
```

6. Check if symlinks to `/opt/cray` from `/global/opt/cray` exist on the head node, and create them if they do not exist.

   ACE based systems contain the `/opt/cray` directory. In addition, on ACE based system, NFS is mounted on the `/global/opt/cray` directory, which is visible on all the nodes.

a. Execute the following on the head node:

```
$ ls –l /opt/cray
/opt/cray -> /global/cray
```

b. Execute the following on the compute nodes:

```
$ ls –l /opt/cray
/opt/cray -> /global/opt/cray
```

c. Execute the following on the login nodes:

```
$ ls –l /opt/cray
/opt/cray -> /global/opt/cray
```

d. If the preceding steps indicate that symlinks exist, skip to the next step, otherwise create symlinks by using the following command:

```
$ ln –s /global/opt/cray /opt/cray
```

7. Configure Singularity.

Urika-CS runtime commands assume that some directories from the host are mounted onto the container. The Singularity configuration needs to be modified to mount these directories automatically. The following lines are to be added to `singularity.conf` file, which is typically located under *`singularity_installation_Dir`*`/etc/singularity`.

The directory on the host needs to be bound to the directory inside the container.

Example for ACE based systems:

```
# Mount host libs into the container. Typically this is the location where
#all the fabric drivers are located.
bind path = /lib64:/opt/hostlib
# IBVerbs for Infiniband
bind path = /etc/libibverbs.d:/etc/libibverbs.d
# Slurm WLM libs
bind path = /opt/local/slurm/default/lib64/:/opt/wlm
# Cray ML Plugin
# bind path = Install_Location_of_Cray_PE_ML_Plugin:/opt/ml_comm # Example for
Cray ML Plugin
#For example
bind path = /opt/cray/pe/:/opt/ml_comm

# OpenMPI 3.0.0 installation path
# Mount the location for openmpi txt files required by mpirun at runtime
# bind path = Install_Location_of_OpenMPI
# Example for OpenMPI
bind path = /opt/cray/openmpi/gcc/3.0.0

If it is CS-Storm m/c the following paths are needed.
# CUDA Toolkit directory
bind path = /opt/nvidia:/opt/nvidia
# Nvidia GPU driver directory
bind path = /usr/lib64:/opt/driverlib
```

In some CS systems, the Singularity configuration may vary across nodes, depending on the host directory of the libraries.

8. Verify that the installation was successful.

a. Load the `analytics` module.

```
# module load analytics
```

- If the preceding command is successful, load the OpenMPI module, allocate resources and start the analytics cluster:

```
# module load openmpi/gcc/3.0.0
# salloc -N 2 #For PBS Pro based systems, execute qsub -I -l nodes=1
# start_analytics
The default Spark event log directory /lus/scratch/sparkHistory does not exist.
Using /home/users/userName/.urikacs/sparkHistory instead.  To choose an alternate location, set
SPARK_EVENT_DIR.
Using default per-node loopback filesystem for Spark per-node local temporary storage (controlled by
SPARK_LOCAL_DIRS).
WARNING: Too few nodes allocated.  Running a local spark instance instead.
Running the interactive image (/opt/cray/analytics-singularity-images/1.01.0000.201805171508_0105/analytics-
urikacs-1.0.0000-latest.simg).
preparing Spark interactive worker
‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/userName/spark/conf/docker.properties.template’
‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/userName/spark/conf/fairscheduler.xml.template’
‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/userName/spark/conf/log4j.properties.template’
‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/userName/spark/conf/metrics.properties.template’
‘/usr/spark/conf/slaves.template’ -> ‘/tmp/userName/spark/conf/slaves.template’
‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/userName/spark/conf/spark-defaults.conf.template’
‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/userName/spark/conf/spark-env.sh.template’
adding test data to home directory
Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
```

- If loading the `analytics` module indicates that the system does not recognize module files:

  1. Set the PATH to point to Urika-CS installation directory, as shown in the following example:

     ```
     # export PATH=/opt/cray/analytics/1.01.0000.201807301732_0117/bin/:$PATH
     ```

  2. Ensure that the `configure.shrc` file, located under `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc`, is updated to reflect this change.

  3. Execute the following commands to load the OpenMPI module, allocate resources and start the analytics cluster:

     - Example for Slurm:

       ```
       # module load analytics
       # module load openmpi/gcc/3.0.0
       # salloc -N 2 #For PBS Pro based systems, use qsub -I -l nodes=1
       # start_analytics
       The default Spark event log directory /lus/scratch/sparkHistory does not exist.
       Using /home/users/userName/.urikacs/sparkHistory instead.  To choose an alternate
       location, set SPARK_EVENT_DIR.
       Using default per-node loopback filesystem for Spark per-node local temporary storage
       (controlled by SPARK_LOCAL_DIRS).
       WARNING: Too few nodes allocated.  Running a local spark instance instead.
       Running the interactive image (/opt/cray/analytics-singularity-images/
       1.01.0000.201805171508_0105/analytics-urikacs-1.0.0000-latest.simg).
       preparing Spark interactive worker
       ‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/userName/spark/conf/
       docker.properties.template’
       ‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/userName/spark/conf/
       fairscheduler.xml.template’
       ‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/userName/spark/conf/
       log4j.properties.template’
       ‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/userName/spark/conf/
       metrics.properties.template’
       ‘/usr/spark/conf/slaves.template’ -> ‘/tmp/userName/spark/conf/slaves.template’
       ‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/userName/spark/conf/spark-
       defaults.conf.template’
       ‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/userName/spark/conf/spark-
       env.sh.template’
       adding test data to home directory
       Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
       ```

     - Example for PBS Pro:

```
$ qsub –I –lnodes=2
$ module load analytics
$ module load openmpi/gcc/64/3.0.0
$ start_analytics
The default Spark event log directory /lus/scratch/sparkHistory does not exist.
Using /home/users/userName/.urikacs/sparkHistory instead.  To choose an alternate
location, set SPARK_EVENT_DIR.
Using default per-node loopback filesystem for Spark per-node local temporary storage
(controlled by SPARK_LOCAL_DIRS).
WARNING: Too few nodes allocated.  Running a local spark instance instead.
Running the interactive image (/opt/cray/analytics-singularity-images/
1.01.0000.201805171508_0105/analytics-urikacs-1.0.0000-latest.simg).
preparing Spark interactive worker
‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/userName/spark/conf/
docker.properties.template’
‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/userName/spark/conf/
fairscheduler.xml.template’
‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/userName/spark/conf/
log4j.properties.template’
‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/userName/spark/conf/
metrics.properties.template’
‘/usr/spark/conf/slaves.template’ -> ‘/tmp/userName/spark/conf/slaves.template’
‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/userName/spark/conf/spark-
defaults.conf.template’
‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/userName/spark/conf/spark-
env.sh.template’
adding test data to home directory
Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
```

Additional troubleshooting and post installation verification tests are documented at *Urika-CS Troubleshooting Information*

7 Install Urika-CS on a BCM Based System

Prerequisites

This procedure:

- requires root privileges
- assumes that the installation is being done on a BCM based Cray CS system that supports Singularity and OpenMPI
- assumes Slurm or PBS Pro is being used as the workload manager. Please contact Cray Support if using other workload managers.
- assumes that the supporting software components listed in *Install Supporting Software* on page 9 have been installed on the system.

About this task

Use the following instructions to install and configure Urika-CS software. Major software components that are installed during this process include:

- Urika-CS RPMs
- Cray PE ML plugin

Procedure

1. Log on to the head node as root.
2. Install Urika-CS software.

   Urika-CS is packaged as RPMs. The Urika-CS package includes:

   - Urika-CS RPM, which includes the Singularity image, user scripts, man pages, and the Urika-CS module file.
   - Cray PE ML plugin RPMs:
     - Cray PE ML plugin Py2 RPM
     - Cray PE ML plugin Py3 RPM

   Urika-CS image, scripts, and the Cray ML PE plugin need to be installed and available on all the nodes.

```
# rpm -ivh \
/opt/cray/analytics-singularity-images/1.01.201807301732_0117/analytics-urikacs-1.0.0000-latest.simg
/opt/cray/analytics/1.01.201807301732_0117/bin
/opt/cray/analytics/1.01.201807301732_0117/bin/analytics.shrc
/opt/cray/analytics/1.01.201807301732_0117/bin/configure.shrc
/opt/cray/analytics/1.01.201807301732_0117/bin/run_training
/opt/cray/analytics/1.01.201807301732_0117/bin/start_analytics
/opt/cray/analytics/1.01.201807301732_0117/man
/opt/cray/analytics/1.01.201807301732_0117/man/man1
/opt/cray/analytics/1.01.201807301732_0117/man/man1/run_training.1
```

```
/opt/cray/analytics/1.01.201807301732_0117/man/man1/start_analytics.1
/opt/cray/modulefiles/analytics/1.01.20180730
```

3. Install Cray PE ML plugin (version 1.1.1) and set the path in the Singularity configuration.

   The plugin is installed by default under `/opt/cray/pe`.

```
# rpm -ivh --nodeps craype-ml-plugin-py3-1.1.1-201803051608.6614ddecd7024.el7-1.x86_64.rpm
# rpm -ivh --nodeps craype-ml-plugin-py2-1.1.1-201803051608.6614ddecd7024.el7-1.x86_64.rpm
```

4. Skip this step if the system uses modules as a way to set the right environment. If the CS machine does not use modulefiles or modules, edit the `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc` file to make the following changes:

   a. Set `MODULES_EXIST=no`

   b. Set the value of the `__ANALYTICS_SINGULARITY_IMG_DIR` variable to the path to the Urika-CS image location:

```
# export __ANALYTICS_SINGULARITY_IMG_DIR=\
/opt/cray/analytics-singularity-images/1.01.0000.201806211250_0111/
```

   c. Uncomment and set the following variables without changing the values of the variables.

```
#export __ANALYTICS_CONTAINER_NAME=urikacs
#export ANALYTICS_IMG=analytics-urikacs-1.0.0000-latest.simg
```

5. Configure Urika-CS by setting the environmental variables needed for Urika-CS runtime in the `configure.shrc` script, which is located under `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc`.

   Variables to configure include:

   - The `OMPI_DIR` and `OPAL_PREFIX` variables are required to run the Cray PE ML plugin.
   - The network interface available on the compute nodes can be set using `URIKACS_DEV_INF` variable. Urika-CS by default assumes InfiniBand is running. If it is not running, Urika-CS uses the available Ethernet interface. If this assumption is not true then the admin can set the `URIKACS_DEV_INF` variable.
   - A default `SPARK_EVENT_DIR` is used to log Spark History. This can be changed by setting `SPARK_EVENT_DIR` variable.

   A sample `configure.shrc` configuration file for a BCM based system is shown below:

```
###### This file has to be configured ######
###### The network interface on compute nodes ######
#export URIKACS_DEV_INF="ib0"
###### The OpenMPI 3.0.0 installation directory #####
export OMPI_DIR=/cm/shared/apps/openmpi/gcc/64/3.0.0/
###### The OPAL_PREFIX needed for mpirun #####
export OPAL_PREFIX=/cm/shared/apps/openmpi/gcc/64/3.0.0
###### configure SPARK_EVENT_DIR #####
#export SPARK_EVENT_DIR=$HOME/.urikacs/sparkHistory
```

6. Create symlinks.

   BCM based systems do not contain a `/opt/cray` directory by default. Create symbolic links to `/opt/cray` from `/cm/shared/apps` on the head node. This allows `/cm/shared/apps/opt/cray` to be visible on login and compute nodes.

```
$ cd /cm/shared
$ mkdir –p opt/cray
$ ln –s /cm/shared/opt/cray /opt/cray
$ ls –l /opt/cray
/opt/cray -> /cm/shared/opt/cray
```

7. Configure Singularity.

   Urika-CS runtime commands assume that some directories from the host are mounted onto the container. The Singularity configuration needs to be modified to mount these directories automatically. The following lines are to be added to `singularity.conf` file, which is typically located under *`singularity_installation_Dir`*`/etc/singularity`.

   The directory on the host needs to be bound to the directory inside the container.

   Example for BCM based systems

```
# Mount host libs into the container. Typically this is the location where
# all the fabric drivers are located.
bind path = /lib64:/opt/hostlib
# IBVerbs for Infiniband
bind path = /etc/libibverbs.d:/etc/libibverbs.d

# Slurm WLM libs
# mount Slurm WLM libs
# for example on an ACE machine
bind path = /cm/shared/apps/slurm/current/lib64/slurm:/opt/wlm

# Cray ML Plugin
# bind path = Install_Location_of_Cray_PE_ML_Plugin:/opt/ml_comm
# Example for Cray ML Plugin
# bind path = /cm/shared/opt/cray/pe/:/opt/ml_comm

# OpenMPI 3.0.0 installation path
# Mount the location for openmpi txt files required by mpirun at runtime
bind path = /cm/shared/apps/openmpi/gcc/64/3.0.0/

#If it is CS-Storm  the following paths are needed.
# CUDA Toolkit directory
# For example
bind path = /cm/shared/apps/cuda90/toolkit/:/opt/nvidia/cudatoolkit
# # Nvidia GPU driver directory
# For example
bind path = /cm/local/apps/cuda-driver/libs/390.46/lib64:/opt/driverlib
```

   > **NOTE:** By default, only the `home` directory is mounted on Singularity, whereas the Lustre file system is not. If there is storage available on a Lustre file system, the Lustre file system directory will need to be explicitly mounted.

   There are two ways to mount the Lustre file system:

   - Add the bind path to the Singularity configuration, as shown in the following example:

     ```
     bind path = /lus:/lus
     ```

   - Use the `--mount` flag on the Urika-CS CLI commands.

   In some CS systems, the Singularity configuration may vary across nodes, depending on the host directory of the libraries.

8. Verify that the installation was successful.

a. Load the `analytics` module.

```
# module load analytics
```

- If the preceding command is successful, load the OpenMPI module, allocate resources and start the analytics cluster:

```
# module load openmpi/gcc/64/3.0.0
# salloc -N 2 #For PBS Pro based systems, execute qsub -I -l nodes=1
# start_analytics
The default Spark event log directory /lus/scratch/sparkHistory does not exist.
Using /home/users/userName/.urikacs/sparkHistory instead.  To choose an alternate location, set
SPARK_EVENT_DIR.
Using default per-node loopback filesystem for Spark per-node local temporary storage (controlled by
SPARK_LOCAL_DIRS).
WARNING: Too few nodes allocated.  Running a local spark instance instead.
Running the interactive image (/opt/cray/analytics-singularity-images/1.01.0000.201805171508_0105/analytics-
urikacs-1.0.0000-latest.simg).
preparing Spark interactive worker
‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/userName/spark/conf/docker.properties.template’
‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/userName/spark/conf/fairscheduler.xml.template’
‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/userName/spark/conf/log4j.properties.template’
‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/userName/spark/conf/metrics.properties.template’
‘/usr/spark/conf/slaves.template’ -> ‘/tmp/userName/spark/conf/slaves.template’
‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/userName/spark/conf/spark-defaults.conf.template’
‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/userName/spark/conf/spark-env.sh.template’
adding test data to home directory
Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
```

- If loading the `analytics` module indicates that the system does not recognize module files:

  1. Set the PATH to point to Urika-CS installation directory, as shown in the following example:

     ```
     # export PATH=/opt/cray/analytics/1.01.0000.201807301732_0117/bin/:$PATH
     ```

  2. Ensure that the `configure.shrc` file, located under `/opt/cray/analytics/1.01.0000.201807301732_0117/bin/configure.shrc`, is updated to reflect this change.

  3. Execute the following commands to load the OpenMPI module, allocate resources and start the analytics cluster:

     - Example for Slurm:

       ```
       # module load analytics
       # module load openmpi/gcc/64/3.0.0
       # salloc -N 2
       # start_analytics
       The default Spark event log directory /lus/scratch/sparkHistory does not exist.
       Using /home/users/userName/.urikacs/sparkHistory instead.  To choose an alternate
       location, set SPARK_EVENT_DIR.
       Using default per-node loopback filesystem for Spark per-node local temporary storage
       (controlled by SPARK_LOCAL_DIRS).
       WARNING: Too few nodes allocated.  Running a local spark instance instead.
       Running the interactive image (/opt/cray/analytics-singularity-images/
       1.01.0000.201805171508_0105/analytics-urikacs-1.0.0000-latest.simg).
       preparing Spark interactive worker
       ‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/userName/spark/conf/
       docker.properties.template’
       ‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/userName/spark/conf/
       fairscheduler.xml.template’
       ‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/userName/spark/conf/
       log4j.properties.template’
       ‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/userName/spark/conf/
       metrics.properties.template’
       ‘/usr/spark/conf/slaves.template’ -> ‘/tmp/userName/spark/conf/slaves.template’
       ‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/userName/spark/conf/spark-
       defaults.conf.template’
       ‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/userName/spark/conf/spark-
       env.sh.template’
       adding test data to home directory
       Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
       ```

     - Example for PBS Pro:

```
$ qsub –I –lnodes=2
$ module load analytics
$ module load openmpi/gcc/64/3.0.0
$ start_analytics
The default Spark event log directory /lus/scratch/sparkHistory does not exist.
Using /home/users/userName/.urikacs/sparkHistory instead.  To choose an alternate
location, set SPARK_EVENT_DIR.
Using default per-node loopback filesystem for Spark per-node local temporary storage
(controlled by SPARK_LOCAL_DIRS).
WARNING: Too few nodes allocated.  Running a local spark instance instead.
Running the interactive image (/opt/cray/analytics-singularity-images/
1.01.0000.201805171508_0105/analytics-urikacs-1.0.0000-latest.simg).
preparing Spark interactive worker
‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/userName/spark/conf/
docker.properties.template’
‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/userName/spark/conf/
fairscheduler.xml.template’
‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/userName/spark/conf/
log4j.properties.template’
‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/userName/spark/conf/
metrics.properties.template’
‘/usr/spark/conf/slaves.template’ -> ‘/tmp/userName/spark/conf/slaves.template’
‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/userName/spark/conf/spark-
defaults.conf.template’
‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/userName/spark/conf/spark-
env.sh.template’
adding test data to home directory
Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
```

Additional troubleshooting and post installation verification tests are documented at *Urika-CS Troubleshooting Information*

8 Additional Supporting Information

8.1 Resource Allocation

Before an analytics cluster can be started, the desired number of nodes needs to be allocated using the system's workload manager. If *N* number of nodes are allocated, one of them will be allocated as a master and one of them will be allocated as an interactive node.

In addition, if the system uses:

- Slurm, *N*-2 worker containers will be launched.
- PBS Pro, *N*-2 worker containers will be launched.

8.2 Urika-CS Log File Locations

- Log files for a given Urika®-CS service are located on the node(s) that the respective service is running on.
- Spark - Default Spark log levels are controlled by the `/tmp/spark/conf/log4j.properties` file. Default Spark settings are used when the system is installed, but can be customized by creating a new `log4j.properties` file. A template for this customization can be found in the `log4j.properties.template` file. The Spark service does not need to be restarted if the log level is changed.
    - **Spark event Logs** - Urika-CS stores Spark event logs in per-user directories. By default, the location is `/lus/scratch/sparkHistory/` if it is available, or `$HOME/.urikacs/sparkHistory` if it is not. Users may override this and select their own event log directory by setting the environment variable `SPARK_EVENT_DIR` prior to running `start_analytics`. Users may copy these event logs to their local machines, and locally execute the Spark History Server or any other tools which parse event logs.
    - **Spark worker logs** - Spark worker logs - These logs reside in the `$HOME/.urikacs/sparkHistory` directory on the local nodes they run on.

8.3 Local Storage Options for Apache Spark

On Urika-CS, Spark in Singularity containers will default to looking for the `/tmp` directory on the local disk for scratch space/local storage data.

It is important to note the following items when using Spark in Singularity containers:

- If local storage exists and is mapped to `/tmp`, then no change required to the Singularity configuration.
- If local storage exists but `/tmp` is not mapped to the local storage, administrators can create a local directory and mount it onto the container at `/tmp`, using the bind path option in the container.
- If local storage does not exist, use NFS or Lustre and mount it at `/tmp` in the container.

8.4 Quick Reference Information

Software Components

Table 3. Urika®-CS Software Component Versions

Software Component	Version
Apache Spark	2.2.0
Anaconda Distribution of Python	5.0.0
Dask	0.14.3 and later
Dask Distributed	1.16.3 and later
Intel BigDL	0.5.0
Analytics Programming Environment	
Python	3.6 as part of Anaconda 5.0.0. Anaconda also supports creating `python` environments with 2.7, 3.4, and 3.5
Java	1.8
Scala	2.11.8
R	3.5.0
Maven	3.3.9
SBT	0.13.9
ANT	1.9.2
TensorFlow	1.6.0
TensorBoard	1.6.0
Jupyter NoteBook	4.3.0

Urika-CS CLI Commands

Table 4. CLI Command Reference

Command	Description
`start_analytics`	Starts an analytics cluster, which can be used to run Spark and/or the analytic programming environment commands.
`run_training`	Runs a command in a Singularity container.

File Locations

Table 5. File Location Reference

File	Location
Urika-CS module file	`/opt/cray/modulefiles/analytics`
`configure.shrc` file	`/opt/cray/analytics-singularity-images/1.01.0000.201806210126_0012/bin` on all the nodes.
Urika-CS image directory	`/opt/cray/analytics-singularity-images/default`

9 Urika-CS Troubleshooting Information

Module Considerations

- If a module is not available and module files are not recognized by the system, skip the `module load analytics` command in the following sections.
- If a module is not available and Urika-CS is already installed on the system, set the `PATH` to point to Urika-CS installation directory as described in *Install Urika-CS Software*.

Consideration for Running on PBS Pro Based Systems

On PBS Pro run the following commands after allocating and loading the required modules.

For example:

```
$ qsub –I –lnodes=4
$ module load analytics
$ module load openmpi/gcc/64/3.0.0
```

The path shown in the preceding example for loading the openMPI module depends on the system.

After executing the preceding commands, execute `start_analytics` or `run_training`, specifying the required options.

9.1 Troubleshoot Singularity Related Issues

- **Installation Checks**

  Use the following commands to check if Singularity has been installed correctly. These commands need to be run on the nodes that Urika-CS needs to be used on.

  ```
  $ salloc -N 2
  $ module load singularity
  ```

  Executing the preceding command may cause the system to return the following message if the module file has not have been created in the correct directory.

  ```
  ModuleCmd_Load.c(244):ERROR:105: Unable to locate a modulefile for 'singularity'
  ```

  Ensure that the module file is installed at `/opt/cray/modulefiles/analytics` and then execute the following to ensure Singularity has been installed correctly:

  ```
  $ module load singularity
  $ singularity --version
  2.5
  $ singularity selftest
  + sh -c test -f /global/opt/singularity/2.5.0/etc/singularity/singularity.conf       (retval=0) OK
  ```

```
+ test -u /global/opt/singularity/2.5.0/libexec/singularity/bin/action-suid           (retval=0) OK
+ test -u /global/opt/singularity/2.5.0/libexec/singularity/bin/mount-suid            (retval=0) OK
+ test -u /global/opt/singularity/2.5.0/libexec/singularity/bin/start-suid            (retval=0) OK
```

- **Accessibility Checks**

  Execute the following command to ensure that no error is returned, which indicates that all the compute nodes can access Singularity:

  ```
  $ salloc -N 4
  $ module load singularity
  $ srun which singularity
  ```

- **Miscellaneous Checks**

  Check if Singularity can open a shell inside an image:

  ```
  $ module load singularity
  $ singularity shell /opt/cray/analytics-singularity-images/default/analytics-urikacs-1.0.0000-latest.simg
  ```

  Executing the preceding command may cause the system to return one of the following errors:

  - `ERROR : Failed to resolve path to /usr/local/var/singularity/mnt/container`
  - `ABORT : Retval = 255`
  - `ERROR : Failed invoking the NEWUSER namespace runtime: Invalid argument`
  - `ERROR : No valid /bin/sh in container`
  - `ABORT : Retval = 255`

  If any of the preceding errors is returned, check if Singularity has been installed on the NFS mounted drive. Ensure that Singularity is installed on each node and is accessible from all the nodes that Urika-CS needs to run on.

For more information, refer to installation of Singularity in *Install Supporting Software* on page 9

9.2 Troubleshoot Issues Related to Accessibility and User Commands

- **Accessibility Checks**

  Ensure that Urika-CS is accessible from all the nodes that it needs to be used on.

  ```
  $ module load analytics
  ```

  If executing the preceding command indicates that the `analytics` module is not available, check if the module has been installed and accessible from that node.

- **User Command Checks**

  Execute the following command to verify that user commands are working as expected:

  ```
  $ module load analytics
  $ salloc -N 2
  $ run_training -n 2 --no-node-list 'hostname'
  prod-0078
  prod-0077
  $ start_analytics
  The default Spark event log directory /lus/scratch/sparkHistory does not exist.
  Using /home/users/alice/.urikacs/sparkHistory instead.  To choose an alternate
  location, set SPARK_EVENT_DIR.
  ```

```
Using default per-node loopback filesystem for Spark per-node local temporary
storage (controlled by SPARK_LOCAL_DIRS).
WARNING: Too few nodes allocated.  Running a local spark instance instead.
Running the interactive image (/opt/cray/analytics-singularity-images/
1.01.0000.201805291754_0106/analytics-urikacs-1.0.0000-latest.simg).
preparing Spark interactive worker
‘/usr/spark/conf/docker.properties.template’ -> ‘/tmp/alice/spark/conf/
docker.properties.template’
‘/usr/spark/conf/fairscheduler.xml.template’ -> ‘/tmp/alice/spark/conf/
fairscheduler.xml.template’
‘/usr/spark/conf/log4j.properties.template’ -> ‘/tmp/alice/spark/conf/
log4j.properties.template’
‘/usr/spark/conf/metrics.properties.template’ -> ‘/tmp/alice/spark/conf/
metrics.properties.template’
‘/usr/spark/conf/slaves.template’ -> ‘/tmp/alice/spark/conf/slaves.template’
‘/usr/spark/conf/spark-defaults.conf.template’ -> ‘/tmp/alice/spark/conf/spark-
defaults.conf.template’
‘/usr/spark/conf/spark-env.sh.template’ -> ‘/tmp/alice/spark/conf/spark-
env.sh.template’
adding test data to home directory
Analytics cluster ready.  Type 'spark-shell' for an interactive Spark shell.
```

If the preceding command returns an error message, check if Urika-CS is installed correctly with PATH being set to include Urika-CS installation directory. The Urika-CS installation directory should be available on all the nodes.

9.3 Troubleshoot gPRC and TensorFlow Related Issues

Execute the following command to verify that gPRC and TensorFlow are working as expected:

```
$ module load analytics
$ salloc -N 2
$ run_training -n 2 --no-node-list 'python $CRAYPE_ML_PLUGIN_BASEDIR/examples/tf_cnn_benchmarks/tf_cnn_benchmarks.py \
--num_gpus=1 --batch_size=32 --model=trivial --variable_update=parameter_server'

Step    Img/sec loss
1       images/sec: 2.7 +/- 0.0 (jitter = 0.0)  30.208
1       images/sec: 2.7 +/- 0.0 (jitter = 0.0)  30.208
10      images/sec: 2.7 +/- 0.0 (jitter = 0.0)  96.949
10      images/sec: 2.7 +/- 0.0 (jitter = 0.0)  96.949
20      images/sec: 2.7 +/- 0.0 (jitter = 0.0)  147.888
20      images/sec: 2.7 +/- 0.0 (jitter = 0.0)  147.888
30      images/sec: 2.7 +/- 0.0 (jitter = 0.0)  238.592
30      images/sec: 2.7 +/- 0.0 (jitter = 0.0)  238.592
```

The preceding result indicates that GPRC and TensorFlow are working as expected. However, if the preceding command returns an error, verify that the Singularity configuration and Cray PE ML plugin directory are mounted properly, as specified in the configuration.

9.4 Troubleshoot Apache Spark Related Issues

Execute the following command to verify that Spark is working as expected:

```
$ module load analytics
$ salloc –N 2
```

```
$ start_analytics
>> run-example SparkPi
```

- Example result of a successful execution:

```
Pi is roughly 3.1433757168785843
```

- Example result of failure:

```
/opt/rt_scripts/bin/setup_rt_env.sh: line 347: /root/.urikacs/startuplogs/job.
26/
master.log: No such file or directory
Running the 2 worker images (/opt/cray/analytics-singularity-images/
1.01.0000.201806061400_0108/analytics-urikacs-1.0.0000-latest.simg).
Logging to /root/.urikacs/startuplogs/job.26/workers.log. \
/opt/rt_scripts/bin/setup_rt_env.sh: line 347: \
/root/.urikacs/startuplogs/job.26/ workers.log: \
No such file or directory
/opt/rt_scripts/bin/setup_rt_env.sh: \
line 347: /root/.urikacs/startuplogs/job.26/ workers.log:\
No such file or directory
..... .
18/06/08 20:21:08 ERROR \
SparkContext: Error initializing SparkContext.
org.apache.spark.SparkException: \
Invalid master URL: spark://:7077
The preceding result indicates that the directory /root/.urikacs is not
writable from within the container. \
Rerun the command as a non-root user to resolve the issue
```

The preceding result indicates that the directory `/root/.urikacs` is not writable from within the container. Rerun the command as a non-root user to resolve the issue.

9.5 Troubleshoot NVIDIA, CUDA and CuDNN Related Issues

Execute the following commands on GPU nodes to verify the Singularity configuration settings with respect to GPU libraries:

```
$ module load analytics
$ salloc -N 2
$ run_training -n 2 --cudnn-libs path_to_cudnn "python -c 'import tensorflow as tf; print(tf.__version__)'"
1.6.0
/opt/anaconda/lib/python3.6/site-packages/h5py/__init__.py:34: FutureWarning: \
Conversion of the second argument of issubdtype from `float` to `np.floating` \
is deprecated. In future, it will be treated as `np.float64 == np.dtype(float).type`.
  from ._conv import register_converters as _register_converters
```

Use the following table to resolve issues encountered while executing the preceding command:

Table 6. NVIDIA, CUDA and CuDNN Related Issues and Resolutions

Error Message	Resolution
ImportError: libcublas.so.9.0: cannot open shared object file: No such file or directory	Check if CUDA toolkit is mounted per the installation instructions.
libcuda.so.1: cannot open shared object file: No such file or directory	Ensure that the Nvidia drivers for `GPU - libcuda.so.1` are located under `/usr/lib64` on the compute nodes. If this is not the location,

Error Message	Resolution
	pass the `--gpu-libs` as an additional argument to the `run_training` command
ibcudnn.so.7: cannot open shared object file: No such file or directory	Ensure that CuDNN version used is v7.1.2 for CUDA toolkit 9.0.

For more information, refer to installation of Nvidia, CUDA, and CUDA toolkit in *Install Supporting Software* on page 9

9.6 Troubleshoot Cray PE ML Plugin and TensorFlow Related Issues on ACE Based Systems

Execute the following commands to verify that functionality of the Cray PE ML plugin with TensorFlow.

```
$ module load analytics
$ module load openmpi/gcc/3.0.0
$ salloc -N 2
$ run_training -n 2 --craype-plugin --no-node-list \
'python $CRAYPE_ML_PLUGIN_BASEDIR/examples/tf_cnn_benchmarks/tf_cnn_benchmarks.py \
--num_gpus=1 --data_format=NHWC --batch_size=32 --model=trivial --train_dir=$HOME/tf_cnn_train \
--data_name=imagenet --variable_update=ps_ml_comm  --local_parameter_device=cpu'

Step    Img/sec loss
1   images/sec: 3.2 +/- 0.0 (jitter = 0.0)  10.681
1   images/sec: 3.3 +/- 0.0 (jitter = 0.0)  10.681
```

If there are issues running TensorFlow using the Cray PE ML plugin, check if `openmpi` has been installed. If the module file already includes settings for the `OPAL_PREFIX` and `OMPI_DIR` variables, setting them again is not required.

Use the following table to troubleshoot any errors returned by executing the preceding commands:

Table 7. Cray PE ML Plugin and TensorFlow Related Issues and Resolutions

Error	Resolution
• Sorry! You were supposed to get help about: opal_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-opal-runtime.txt: No such file or directory. Sorry! • Sorry! You were supposed to get help about: orte_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-orte-runtime: No such file or directory. Sorry! • Sorry! You were supposed to get help about: mpi_init:startup:internal-failure But I couldn't open the help file: /global/opt/	Mount the OpenMPI installation if it is not installed in a standard location. For example in the first example on the left, use the `--mount` option with the `run_training` command, as shown in the following example: `--mount /gloabal/opt/ompi:/global/opt/ompi` Alternatively, add it to the Singularity configuration as the bind path.

Error	Resolution
ompi/share/openmpi/help-mpi-runtime.txt: No such file or directory. Sorry! • Sorry! You were supposed to get help about: mpi_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-mpi-runtime.txt: No such file or directory. Sorry! • *** An error occurred in MPI_Init_thread *** on a NULL communicator *** MPI_ERRORS_ARE_FATAL (processes in this communicator will now abort, *** and potentially your MPI job) [prod-0077:80273] Local abort before MPI_INIT completed completed successfully, but am not able to aggregate error messages, and not able to guarantee that all other processes were killed!	
We encountered issues with Slurm and OMPI inter-dependency.	Ensure that OpenMPI dependent libraries are available at `/usr/lib64`. In addition, ensure that this directory is mounted in the Singularity configuration at `/opt/hostlib`

ibibverbs: Warning: couldn't load driver 'librxe-rdmav2.so': \ librxe-rdmav2.so: cannot open shared object file: No such file or directory libibverbs: Warning: couldn't load driver 'libqedr-rdmav2.so': \ libqedr-rdmav2.so: cannot open shared object file: No such file or directory mca_base_component_repository_open: unable to open mca_btl_openib: \librdmacm.so.1: cannot open shared object file: No such file or directory	Ensure that the `libibverbs` dependent libraries are available at `/usr/lib64` on all the nodes.
	• On Slurm based systems, execute:
	`$ ompi_info`
	`Configure command line: \`
	`'--prefix=/global/opt/openmpi/3.0.0/gcc'`
	`'--with-pmi=/opt/local/slurm/default'`
	• On PBS Pro based systems, execute:

For more information, refer to installation of OpenMPI in *Install Supporting Software* on page 9

9.7 Troubleshoot Cray PE ML Plugin and TensorFlow Related Issues on BCM Based Systems

Execute the following commands to verify that functionality of the Cray PE ML plugin with TensorFlow.

```
$ module load analytics
$ module load openmpi/gcc/64/3.0.0
$ salloc -N 2
$ run_training -n 2 --craype-plugin --no-node-list \
'python $CRAYPE_ML_PLUGIN_BASEDIR/examples/tf_cnn_benchmarks/tf_cnn_benchmarks.py \
--num_gpus=1 --data_format=NHWC --batch_size=32 --model=trivial --train_dir=$HOME/tf_cnn_train \
```

```
--data_name=imagenet --variable_update=ps_ml_comm  --local_parameter_device=cpu'

Step    Img/sec loss
1   images/sec: 3.2 +/- 0.0 (jitter = 0.0)  10.681
1   images/sec: 3.3 +/- 0.0 (jitter = 0.0)  10.681
```

If there are issues running TensorFlow using the Cray PE ML plugin, check if `openmpi` has been installed. If the module file already includes settings for the `OPAL_PREFIX` and `OMPI_DIR` variables, setting them again is not required.

Use the following table to troubleshoot any errors returned by executing the preceding commands:

Table 8. Cray PE ML Plugin and TensorFlow Related Issues and Resolutions

Error	Resolution
• Sorry! You were supposed to get help about: opal_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-opal-runtime.txt: No such file or directory. Sorry! • Sorry! You were supposed to get help about: orte_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-orte-runtime: No such file or directory. Sorry! • Sorry! You were supposed to get help about: mpi_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-mpi-runtime.txt: No such file or directory. Sorry! • Sorry! You were supposed to get help about: mpi_init:startup:internal-failure But I couldn't open the help file: /global/opt/ompi/share/openmpi/help-mpi-runtime.txt: No such file or directory. Sorry! • *** An error occurred in MPI_Init_thread *** on a NULL communicator *** MPI_ERRORS_ARE_FATAL (processes in this communicator will now abort, *** and potentially your MPI job) [prod-0077:80273] Local abort before MPI_INIT completed completed successfully, but am not able to	Mount the OpenMPI installation if it is not installed in a standard location. For example in the first example on the left, use the `--mount` option with the `run_training` command, as shown in the following example: `--mount /cm/shared/apps/openmpi/gcc/64/3.0.0:/cm/shared/apps/openmpi/gcc/64/3.0.0` Alternatively, add it to the Singularity configuration as the bind path.

Error	Resolution
aggregate error messages, and not able to guarantee that all other processes were killed!	
We encountered issues with Slurm and OMPI inter-dependency.	Ensure that OpenMPI dependent libraries are available at `/usr/lib64`. In addition, ensure that this directory is mounted in the Singularity configuration at `/opt/hostlib`

ibibverbs: Warning: couldn't load driver 'librxe-rdmav2.so': \ librxe-rdmav2.so: cannot open shared object file: No such file or directory libibverbs: Warning: couldn't load driver 'libqedr-rdmav2.so': \ libqedr-rdmav2.so: cannot open shared object file: No such file or directory mca_base_component_repository_open: unable to open mca_btl_openib: \librdmacm.so.1: cannot open shared object file: No such file or directory	Ensure that the `libibverbs` dependent libraries are available at `/usr/lib64` on all the nodes.
	• On Slurm based systems, execute:
	`$ ompi_info`
	`Configure command line: \`
	`'--prefix=/cm/shared/apps/openmpi/gcc/64/3.0.0'`
	`'--with-pmi=/opt/local/slurm/default'`
	• On PBS Pro based systems, execute:

For more information, refer to installation of OpenMPI in *Install Supporting Software* on page 9