RDMA for Apache Hadoop 3.x 0.9.1 User Guide

HIGH-PERFORMANCE BIG DATA TEAM
http://hibd.cse.ohio-state.edu

NETWORK-BASED COMPUTING LABORATORY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
THE OHIO STATE UNIVERSITY

Last revised: February 22, 2021

Contents

1 Overview of the RDMA for Apache Hadoop 3.x Project

RDMA for Apache Hadoop 3.x is a high-performance design of Hadoop over RDMA-enabled Interconnects. This version of RDMA for Apache Hadoop 3.x 0.9.1 is based on Apache Hadoop 3.0.0 and is compliant with Apache Hadoop 3.0.0.This file is intended to guide users through the various steps involved in installing, configuring, and running RDMA for Apache Hadoop 3.x over InfiniBand.

In this package, many different modes have been included that can be enabled/disabled to obtain performance benefits for different kinds of applications in different Hadoop environments.

Following are the different modes that are included in our package.

HHH: Heterogeneous storage devices with hybrid replication schemes are supported in this mode of operation to have better fault-tolerance as well as performance. This mode is enabled by **default** in the package.

HHH-M: A high-performance in-memory based setup has been introduced in this package that can be utilized to perform all I/O operations in-memory and obtain as much performance benefit as possible.

Running with Slurm and PBS: Supports deploying RDMA for Apache Hadoop 3.x with Slurm and PBS in different running modes (HHH and HHH-M).

If there are any questions, comments or feedback regarding this software package, please post them to rdma-hadoop-discuss mailing list (rdma-hadoop-discuss@cse.ohio-state.edu).

2 Features

High-level features of RDMA for Apache Hadoop 3.x 0.9.1 are listed below.

- Based on Apache Hadoop 3.0.0
- Compliant with Apache Hadoop 3.0.0 APIs and applications
- Support for RDMA Device Selection
- High performance design with native InfiniBand and RoCE support at the verbs level for HDFS component
- Supports deploying Hadoop with Slurm and PBS in different running modes (HHH and HHH-M)
- Easily configurable for different running modes (HHH and HHH-M) and different protocols (native InfiniBand, RoCE, and IPoIB)
- On-demand connection setup
- HDFS over native InfiniBand and RoCE
  - RDMA-based write
  - RDMA-based replication
  - Overlapping in different stages of write and replication

  - Enhanced hybrid HDFS design with in-memory and heterogeneous storage (HHH)
    * Supports two modes of operations
      · HHH (default) with I/O operations over RAM disk, SSD, and HDD
      · HHH-M (in-memory) with I/O operations in-memory
    * Policies to efficiently utilize heterogeneous storage devices (RAM Disk, SSD, and HDD)
      · Greedy and Balanced policies support
      · Automatic policy selection based on available storage types
    * Hybrid replication (in-memory and persistent storage) for HHH default mode
    * Memory replication (in-memory only with lazy persistence) for HHH-M mode
      · No HDFS replication
      · Reduced local storage space usage

- Tested with
  - Mellanox InfiniBand adapters (DDR, QDR, FDR, and EDR)
  - RoCE support with Mellanox adapters
  - RAM Disks, SSDs, and HDDs
  - OpenJDK and IBM JDK

3 Setup Instructions

3.1 Prerequisites

Prior to the installation of RDMA for Apache Hadoop 3.x, please ensure that you have the latest version of JDK installed on your system, and set the `JAVA_HOME` and `PATH` environment variables to point to the appropriate JDK installation. We recommend the use of JDK version 1.8 and later.

In order to use the RDMA-based features provided with RDMA for Apache Hadoop 3.x, install the latest version of the OFED distribution that can be obtained from http://www.openfabrics.org.

3.2 Download

Download the most recent distribution tarball of RDMA for Apache Hadoop package for x86 clusters from http://hibd.cse.ohio-state.edu/download/hibd/rdma-hadoop-3.x-0.9.1-x86-bin.tar.gz.

3.3 Installation steps

Following steps can be used to install the RDMA for Apache Hadoop package.

1. Unzip the RDMA for Apache Hadoop distribution tarball using the following command:

```
tar zxf rdma-hadoop-3.x-0.9.1-x86-bin.tar.gz
```

2. Change directory to rdma-hadoop-3.x-0.9.1-x86

```
cd rdma-hadoop-3.x-0.9.1-x86
```

3.4 Basic Configuration

The configuration files can be found in the directory `rdma-hadoop-3.x-0.9.1-x86/etc/hadoop`, in the RDMA for Apache Hadoop package.

3.4.1 HDFS with Heterogeneous Storage (HHH)

Steps to configure RDMA for Apache Hadoop 3.x include:

1. Configure `hadoop-env.sh` file.

```
export JAVA_HOME=/opt/java/1.8.0
```

2. Configure `core-site.xml` file. RDMA for Apache Hadoop 3.x 0.9.1 supports three different modes: IB, RoCE, and TCP/IP.

   Configuration of the IB mode:

```
<configuration>
 <property>
  <name>fs.defaultFS</name>
  <value>hdfs://node001:9000</value>
  <description>NameNode URI.</description>
 </property>

 <property>
  <name>hadoop.ib.enabled</name>
  <value>true</value>
  <description>Enable the RDMA feature over IB. Default value of
     hadoop.ib.enabled is true.</description>
 </property>

 <property>
  <name>hadoop.roce.enabled</name>
  <value>false</value>
  <description>Disable the RDMA feature over RoCE. Default value
     of hadoop.roce.enabled is false.</description>
 </property>
</configuration>
```

Configuration of the RoCE mode:

```
<configuration>
 <property>
  <name>fs.defaultFS</name>
  <value>hdfs://node001:9000</value>
  <description>NameNode URI.</description>
 </property>

 <property>
  <name>hadoop.ib.enabled</name>
  <value>false</value>
  <description>Disable the RDMA feature over IB. Default value of
     hadoop.ib.enabled is true.</description>
 </property>

 <property>
  <name>hadoop.roce.enabled</name>
  <value>true</value>
  <description>Enable the RDMA feature over RoCE. Default value of
      hadoop.roce.enabled is false.</description>
 </property>
</configuration>
```

Configuration of the TCP/IP mode:

```
<configuration>
 <property>
  <name>fs.defaultFS</name>
  <value>hdfs://node001:9000</value>
  <description>NameNode URI.</description>
 </property>

 <property>
  <name>hadoop.ib.enabled</name>
  <value>false</value>
  <description>Disable the RDMA feature over IB. Default value of
     hadoop.ib.enabled is true.</description>
 </property>

 <property>
  <name>hadoop.roce.enabled</name>
  <value>false</value>
  <description>Disable the RDMA feature over RoCE. Default value
     of hadoop.roce.enabled is false.</description>
 </property>
</configuration>
```

Note that we should not enable "hadoop.ib.enabled" and "hadoop.roce.enabled" at the same time.

3. Configure `hdfs-site.xml` file.

```
<configuration>
 <property>
  <name>dfs.namenode.name.dir</name>
  <value>file:///home/hadoop/rdma-hadoop-3.x-0.9.1-x86/Name</value>
  <description>Path on the local filesystem where the NameNode
     stores the namespace and transactions logs
     persistently.</description>
 </property>

 <property>
  <name>dfs.datanode.data.dir</name>
  <value>[RAM_DISK]file:///data01, [SSD]file:///data02,
     [DISK]file:///data03</value>
  <description>Comma separated list of paths of the local storage
     devices with corresponding types on a DataNode where it
     should store its blocks.</description>
 </property>
</configuration>
```

4. Configure `yarn-site.xml` file.

```
<configuration>
 <property>
  <name>yarn.resourcemanager.address</name>
  <value>node001:8032</value>
  <description>ResourceManager host:port for clients to submit
     jobs.</description>
 </property>

 <property>
  <name>yarn.resourcemanager.scheduler.address</name>
  <value>node001:8030</value>
  <description>ResourceManager host:port for ApplicationMasters to
      talk to Scheduler to obtain resources.</description>
 </property>

 <property>
  <name>yarn.resourcemanager.resource-tracker.address</name>
  <value>node001:8031</value>
  <description>ResourceManager host:port for
     NodeManagers.</description>
 </property>
```

```
  <property>
    <name>yarn.resourcemanager.admin.address</name>
    <value>node001:8033</value>
    <description>ResourceManager host:port for administrative
       commands.</description>
  </property>

  <property>
    <name>yarn.nodemanager.aux-services</name>
    <value>mapreduce_shuffle</value>
    <description>Shuffle service that needs to be set for MapReduce
       applications.</description>
  </property>
</configuration>
```

5. Configure `mapred-site.xml` file.

```
<configuration>
  <property>
    <name>mapreduce.framework.name</name>
    <value>yarn</value>
    <description>Execution framework set to Hadoop
       YARN.</description>
  </property>

  <property>
    <name>yarn.app.mapreduce.am.command-opts</name>
    <value>-Xmx1024m -Dhadoop.conf.dir=${HADOOP_CONF_DIR}</value>
    <description>Java opts for the MR App Master processes. The
       following symbol, if present, will be interpolated: @taskid@
       is replaced by current TaskID. Any other occurrences of '@'
       will go unchanged. For example, to enable verbose gc logging
       to a file named for the taskid in /tmp and to set the heap
       maximum to be a gigabyte, pass a 'value' of: -Xmx1024m
       -verbose:gc -Xloggc:/tmp/@taskid@.gc
Usage of -Djava.library.path can cause programs to no longer
   function if hadoop native libraries are used. These values
   should instead be set as part of LD_LIBRARY_PATH in the map /
   reduce JVM env using the mapreduce.map.env and
   mapreduce.reduce.env config settings.
    </description>
  </property>

  <property>
    <name>yarn.app.mapreduce.am.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME},LD_LIBRARY_PATH=
```

```
    ${HADOOP_HOME}/lib/native:${LD_LIBRARY_PATH}</value>
    <description>User added environment variables for the MR App
       Master
    processes. Example :
    1) A=foo This will set the env variable A to foo
    2) B=$B:c This is inherit tasktracker's B env variable.
    </description>
  </property>
  <property>
    <name>mapreduce.map.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME},LD_LIBRARY_PATH=
    ${HADOOP_HOME}/lib/native:${LD_LIBRARY_PATH}</value>
  </property>
  <property>
    <name>mapreduce.reduce.env</name>
    <value>HADOOP_MAPRED_HOME=${HADOOP_HOME},LD_LIBRARY_PATH=
    ${HADOOP_HOME}/lib/native:${LD_LIBRARY_PATH}</value>
  </property>

</configuration>
```

6. Configure `workers` file. List all worker hostnames in this file, one per line.

```
node002
node003
```

   We can also configure more specific items according to actual needs. For example, we can configure the item `dfs.blocksize` in `hdfs-site.xml` to change the HDFS block size. To get more detailed information, please visit http://hadoop.apache.org.

3.4.2 HDFS in Memory (HHH-M)

We can enable MapReduce over in-memory HDFS using the following configuration steps:

1. Configure `hdfs-site.xml` file.

```
<configuration>
  <property>
    <name>dfs.namenode.name.dir</name>
    <value>file:///home/hadoop/rdma-hadoop-3.x-0.9.1-x86/Name</value>
    <description>Path on the local filesystem where the NameNode
       stores the namespace and transactions logs
       persistently.</description>
  </property>

  <property>
```

```
    <name>dfs.datanode.data.dir</name>
    <value>[RAM_DISK]file:///data01, [SSD]file:///data02,
       [DISK]file:///data03</value>
    <description>Comma separated list of paths of the local storage
       devices with corresponding types on a DataNode where it
       should store its blocks.</description>
  </property>

  <property>
    <name>dfs.rdma.hhh.mode</name>
    <value>InMemory</value>
    <description>Select In-Memory mode (HHH-M). </description>
  </property>
</configuration>
```

2. Configure the `yarn-site.xml` file as specified in Section 3.4.1.
3. Configure the `mapred-site.xml` file as specified in Section 3.4.1.
4. Configure the `core-site.xml` file as specified in Section 3.4.1.
5. Configure the `slaves` file as specified in Section 3.4.1.

3.5 Advanced Configuration

Some advanced features in RDMA for Apache Hadoop 3.x 0.9.1 can be manually enabled by users. Steps to configure these features in RDMA for Apache Hadoop 3.x 0.9.1 are discussed in this section.

3.5.1 RDMA Device Selection

Users can specify which RDMA device Hadoop should utilize in the case where more than one of such devices is installed. Users can select the RDMA device to be used either by ID or by name in the `core-site.xml` file. By default, RDMA for Apache Hadoop 3.x 0.9.1 will choose the device with ID 0 if none of the following parameters are defined. Theses parameters are applicable for all modes.

```
<property>
  <name>hadoop.rdma.dev.name</name>
  <value>mlx4_0</value>
  <description>Specify the name of the RDMA device to be used. This
     takes precedence over hadoop.rdma.dev.num if both are set.
     </description>
</property>

<property>
  <name>hadoop.rdma.dev.num</name>
  <value>0</value>
```

```
  <description>Specify the ID of the RDMA device to be used.
     </description>
</property>
```

3.5.2 RDMA Port

Users can specify which port Hadoop should listen for establishing RDMA connections by configuring `hdfs-site.xml` file. By default, RDMA for Apache Hadoop 3.x 0.9.1 will listen on 9999. This parameter is applicable for all modes.

```
<property>
  <name>dfs.datanode.rdma.port</name>
  <value>12345</value>
  <description>Specify the port to be used for establishing RDMA
     connections (default is 9999).</description>
</property>
```

3.5.3 Max EndPoint Number

Users can specify the max number of end points attached to each RDMA context by configuring `hdfs-site.xml` file. This parameter is applicable for all modes.

```
<property>
  <name>dfs.rdma.max-ep-num</name>
  <value>16</value>
  <description>Specify the max number of end points attached to each
     RDMA context (default is 8).</description>
</property>
```

3.5.4 Packet Size for RDMA Write

Users can specify the packet size for RDMA write by configuring `hdfs-site.xml` file. This parameter is applicable for all modes.

```
<property>
  <name>dfs.rdma.client-write-packet-size</name>
  <value>262400</value>
  <description>Specify the packet size for RDMA write (default is
     524800, i.e., 512KB).</description>
</property>
```

3.5.5 Placement Policy Selection

Select specific placement policy (Greedy or Balanced) in HDFS by configuring `hdfs-site.xml` file. By default, RDMA for Apache Hadoop 3.x 0.9.1 selects policy automatically based on the storage types of the HDFS data directories. This parameter is applicable for HHH mode as discussed in Section 3.4.1.

```
<property>
  <name>dfs.rdma.placement.policy</name>
  <value>Greedy/Balanced</value>
  <description>Enable specific data placement policy. </description>
</property>
```

3.5.6 Automatic Placement Policy Selection

By default, RDMA for Apache Hadoop 3.x 0.9.1 selects policy automatically based on the storage types of the HDFS data directories. This parameter can be used if the user wants to disable automatic policy detection. This parameter is applicable for HHH mode as discussed in Section 3.4.1.

```
<property>
     <name>dfs.rdma.policy.autodetect</name>
     <value>false</value>
     <description>Disable automatic policy detection (default is
        true). </description>
</property>
```

In order to use the storage policies of default HDFS, users should not use the dfs.rdma.placement.policy parameter as discussed in Section 3.5.5 and disable policy auto detection.

3.5.7 Threshold of RAM Disk Usage

Select a threshold of RAM Disk usage in HDFS by configuring `hdfs-site.xml` file. By default, RDMA for Apache Hadoop 3.x 0.9.1 uses 70% of RAM Disk when RAM Disk is configured as a HDFS data directory. This parameter is applicable for HHH and HHH-M modes as discussed in Section 3.4.1, Section 3.4.2, respectively.

```
<property>
  <name>dfs.rdma.memory.percentage</name>
  <value>0.5</value>
  <description>Select a threshold (default = 0.7) for RAM Disk usage.
     </description>
</property>
```

3.5.8 No. of In-Memory Replicas

Select the number of in-memory replicas in HHH mode by configuring `hdfs-site.xml` file. By default, RDMA for Apache Hadoop 3.x 0.9.1 writes two replicas to RAM Disk and one to persistent storage (replication factor = 3). The no. of in-memory replicas can be changed from one to no. of replication factor (all in-memory). This parameter is applicable for HHH mode as discussed in Section 3.4.1.

```
<property>
  <name>dfs.rdma.memory.replica</name>
  <value>3</value>
  <description>Select no. of in-memory replicas (default = 2).
     </description>
</property>
```

4 Basic Usage Instructions

RDMA for Apache Hadoop 3.x 0.9.1 has management operations similar to default Apache Hadoop 3.0.0. This section lists several of them for basic usage.

4.1 Startup

4.1.1 MapReduce over HDFS

To run MapReduce over HDFS with any of the modes (HHH/HHH-M), please follow these steps.

1. Use the following command to format the directory which stores the namespace and transactions logs for NameNode.

   ```
   $ bin/hdfs namenode -format
   ```

2. Start HDFS with the following command:

   ```
   $ sbin/start-dfs.sh
   ```

3. Start YARN with the following command:

   ```
   $ sbin/start-yarn.sh
   ```

4.2 Basic Commands

1. Use the following command to manage HDFS:

```
$ bin/hdfs dfsadmin
Usage: java DFSAdmin
Note: Administrative commands can only be run as the HDFS
   superuser.
        [-report]
        [-safemode enter | leave | get | wait]
        [-allowSnapshot <snapshotDir>]
        [-disallowSnapshot <snapshotDir>]
        [-saveNamespace]
        [-rollEdits]
        [-restoreFailedStorage true|false|check]
        [-refreshNodes]
        [-finalizeUpgrade]
        [-rollingUpgrade [<query|prepare|finalize>]]
        [-metasave filename]
        [-refreshServiceAcl]
        [-refreshUserToGroupsMappings]
        [-refreshSuperUserGroupsConfiguration]
        [-refreshCallQueue]
        [-printTopology]
        [-refreshNamenodes datanodehost:port]
        [-deleteBlockPool datanode-host:port blockpoolId [force]]
        [-setQuota <quota> <dirname>...<dirname>]
        [-clrQuota <dirname>...<dirname>]
        [-setSpaceQuota <quota> <dirname>...<dirname>]
        [-clrSpaceQuota <dirname>...<dirname>]
        [-setBalancerBandwidth <bandwidth in bytes per second>]
        [-fetchImage <local directory>]
        [-shutdownDatanode <datanode_host:ipc_port> [upgrade]]
        [-getDatanodeInfo <datanode_host:ipc_port>]
        [-help [cmd]]

Generic options supported are
-conf <configuration file> specify an application configuration
   file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated
   files to be copied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar
    files to include in the classpath.
-archives <comma separated list of archives> specify comma
   separated archives to be unarchived on the compute machines.
```

For example, we often use the following command to show the status of HDFS:

```
$ bin/hdfs dfsadmin -report
```

2. Use the following command to manage files in HDFS:

```
$ bin/hdfs dfs
Usage: hadoop fs [generic options]
        [-appendToFile <localsrc> ... <dst>]
        [-cat [-ignoreCrc] <src> ...]
        [-checksum <src> ...]
        [-chgrp [-R] GROUP PATH...]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-copyFromLocal [-f] [-p] <localsrc> ... <dst>]
        [-copyToLocal [-p] [-ignoreCrc] [-crc] <src> ...
           <localdst>]
        [-count [-q] <path> ...]
        [-cp [-f] [-p] <src> ... <dst>]
        [-createSnapshot <snapshotDir> [<snapshotName>]]
        [-deleteSnapshot <snapshotDir> <snapshotName>]
        [-df [-h] [<path> ...]]
        [-du [-s] [-h] <path> ...]
        [-expunge]
        [-get [-p] [-ignoreCrc] [-crc] <src> ... <localdst>]
        [-getfacl [-R] <path>]
        [-getmerge [-nl] <src> <localdst>]
        [-help [cmd ...]]
        [-ls [-d] [-h] [-R] [<path> ...]]
        [-mkdir [-p] <path> ...]
        [-moveFromLocal <localsrc> ... <dst>]
        [-moveToLocal <src> <localdst>]
        [-mv <src> ... <dst>]
        [-put [-f] [-p] <localsrc> ... <dst>]
        [-renameSnapshot <snapshotDir> <oldName> <newName>]
        [-rm [-f] [-r|-R] [-skipTrash] <src> ...]
        [-rmdir [--ignore-fail-on-non-empty] <dir> ...]
        [-setfacl [-R] [{-b|-k} {-m|-x <acl_spec>} <path>]|[--set
           <acl_spec> <path>]]
        [-setrep [-R] [-w] <rep> <path> ...]
        [-stat [format] <path> ...]
        [-tail [-f] <file>]
        [-test -[defsz] <path>]
        [-text [-ignoreCrc] <src> ...]
        [-touchz <path> ...]
        [-usage [cmd ...]]

Generic options supported are
```

```
-conf <configuration file> specify an application configuration
   file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated
   files to be copied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar
    files to include in the classpath.
-archives <comma separated list of archives> specify comma
   separated archives to be unarchived on the compute machines.
```

For example, we can use the following command to list directory contents of HDFS:

```
$ bin/hdfs dfs -ls /
```

3. Use the following command to interoperate with the MapReduce framework:

```
$ bin/mapred job
Usage: JobClient <command> <args>
      [-submit <job-file>]
      [-status <job-id>]
      [-counter <job-id> <group-name> <counter-name>]
      [-kill <job-id>]
      [-set-priority <job-id> <priority>]. Valid values for
         priorities are: VERY_HIGH HIGH NORMAL LOW VERY_LOW
      [-events <job-id> <from-event-#> <#-of-events>]
      [-history <jobOutputDir>]
      [-list [all]]
      [-list-active-trackers]
      [-list-blacklisted-trackers]
      [-list-attempt-ids <job-id> <task-type> <task-state>]

      [-kill-task <task-id>]
      [-fail-task <task-id>]

Usage: CLI <command> <args>
      [-submit <job-file>]
      [-status <job-id>]
      [-counter <job-id> <group-name> <counter-name>]
      [-kill <job-id>]
      [-set-priority <job-id> <priority>]. Valid values for
         priorities are: VERY_HIGH HIGH NORMAL LOW VERY_LOW
      [-events <job-id> <from-event-#> <#-of-events>]
      [-history <jobHistoryFile>]
      [-list [all]]
      [-list-active-trackers]
```

```
      [-list-blacklisted-trackers]
      [-list-attempt-ids <job-id> <task-type> <task-state>]. Valid
         values for <task-type> are REDUCE MAP. Valid values for
         <task-state> are running, completed
      [-kill-task <task-attempt-id>]
      [-fail-task <task-attempt-id>]
      [-logs <job-id> <task-attempt-id>]

Generic options supported are
-conf <configuration file> specify an application configuration
   file
-D <property=value> use value for given property
-fs <local|namenode:port> specify a namenode
-jt <local|jobtracker:port> specify a job tracker
-files <comma separated list of files> specify comma separated
   files to be copied to the map reduce cluster
-libjars <comma separated list of jars> specify comma separated jar
    files to include in the classpath.
-archives <comma separated list of archives> specify comma
   separated archives to be unarchived on the compute machines.
```

For example, we can use the following command to list all active trackers of MapReduce:

```
$ bin/mapred job -list-active-trackers
```

4.3 Shutdown

4.3.1 MapReduce over HDFS

1. Stop HDFS with the following command:

   ```
   $ sbin/stop-dfs.sh
   ```

2. Stop YARN with the following command:

   ```
   $ sbin/stop-yarn.sh
   ```

5 Running RDMA for Apache Hadoop with SLURM/PBS

To run RDMA for Apache Hadoop with SLURM/PBS, scripts in `HADOOP_HOME/bin/slurm_pbs/` directory can be used. These scripts can be used in interactive mode or batch mode. In the interactive mode, the user must allocate interactive nodes, and explicitly use the startup, run benchmark, and shutdown commands described Sections 5.2 or **??**, in their interactive session. In the batch mode, the users must create and lauch a SLURM/PBS batch script with the startup, run benchmark, and shutdown commands described Sections 5.2 or **??**.

Detailed steps for using these scripts are described in Sections 5.1, 5.2 and **??**.

5.1 Usage

This section gives an overview of the SLURM/PBS scripts and their usage for startup and shutdown for running a Hadoop cluster.

5.1.1 Configure and Start a Hadoop Job

For installing, configuring, and starting RDMA for Apache Hadoop with any particular mode of operation, `hibd_install_configure_start.sh` can be used. This script can configure and start Hadoop depending on the parameters provided by the user. Detailed parameter options available with this script are mentioned below:

```
$ ./hibd_install_configure_start.sh ?
Usage: hibd_install_configure_start.sh [options]
   -h <dir>
      specify location of hadoop installation a.k.a. hadoop home

   -m <hhh | hhh-m>
      specify the mode of operation (default: hhh). For more
      information, visit http://hibd.cse.ohio-state.edu/overview/

   -c <dir>
      specify the hadoop conf dir (default: ""). If user provides
      this directory, then the conf files are chosen from this
      directory. Otherwise, the conf files are generated automatically
      with/without user provided configuration with flag '-u'

   -j <dir>
      specify jdk installation or JAVA_HOME (default: ""). If user
      does not provide this, then java installation is searched in
      the environment.

   -u <file>
      specify a file containing all the configurations for hadoop
      installation(default: n/a). Each line of this file must be
      formatted as below:
      "<C|H|M|Y>\t<parameter_name>\t<parameter_value>"
      C = core-site.xml, H = hdfs-site.xml, M = mapred-site.xml,
      Y = yarn-site.xml

   -r <dir>
      specify the ram disk path to use for hhh and hhh-m modes
      (default: /dev/shm)
```

```
   -s
      specify to start hadoop after installation and configuration

   -?
      show this help message
```

5.1.2 Shutdown Hadoop Cluster

After running a benchmark with the script indicated in Section 5.1.1, stopping the Hadoop cluster with cleanup of all the directories can be achieved by using the script `hibd_stop_cleanup.sh`. Similar to the startup script, this cleanup script can makes different parameters available to the user. Detailed parameter options available with this script are mentioned below:

```
$ ./hibd_stop_cleanup.sh ?
Usage: hibd_stop_cleanup.sh [options]
   -h <dir>
      specify location of hadoop installation a.k.a. hadoop home

   -m <hhh | hhh-m>
      specify the mode of operation (default: hhh).
      For more information, visit
      http://hibd.cse.ohio-state.edu/overview/

   -c <dir>
      specify the hadoop conf dir (default: "").

   -r <dir>
      specify the ram disk path to use for hhh and hhh-m modes
      (default: /dev/shm)

   -d
      specify to delete logs and data after hadoop stops

   -?
      show this help message
```

Details of the usage of the above-mentioned scripts can also be found in slurm-script.sh

5.2 Running MapReduce over HDFS with SLURM/PBS

The user can run MapReduce over HDFS in one of the three modes: HHH or HHH-M. Based on the parameters supplied to the `hibd_install_configure_start.sh` script, Hadoop cluster will start with the requested mode of operation configuration setting.

5.2.1 Startup

To start Hadoop in HHH mode, the following command can be used:

```
$ hibd_install_configure_start.sh -s -m hhh-default -r /dev/shm -h
   $HADOOP_HOME -j $JAVA_HOME
```

To start Hadoop in HHH-M mode, the following command can be used:

```
$ hibd_install_configure_start.sh -s -m hhh-m -r /dev/shm -h
   $HADOOP_HOME -j $JAVA_HOME
```

5.2.2 Running Benchmarks

User can launch a benchmark after successful start of the Hadoop cluster. While running a benchmark, user should provide the Hadoop config directory using `--config` flag with hadoop script. If user does not have any pre-configured files, the default config directory will be created in the present working directory named as conf concatenated with job id.

```
$ $HADOOP_HOME/bin/hadoop --config ./conf_<job_id> jar
   $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
   randomwriter -Dmapreduce.randomwriter.mapsperhost=4
   -Dmapreduce.randomwriter.bytespermap=67108864 rand_in

$ $HADOOP_HOME/bin/hadoop --config ./conf_<job_id> jar
   $HADOOP_HOME/share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
   sort rand_in rand_out
```

5.2.3 Shutdown

In order to stop Hadoop and cleanup the corresponding logs, the following command can be used:

```
$ hibd_stop_cleanup.sh -d -r /dev/shm -h $HADOOP_HOME
```

5.2.4 Running Benchmarks

Running benchmark for MapReduce over Lustre with local disks follows the same guidelines as shown above. For benchmarks running on MapReduce over Lustre without local disks, the following command should be used.

```
$ /tmp/hadoop_install_<job_id>/bin/hadoop jar
   /tmp/hadoop_install_<job_id>/share/hadoop/mapreduce/
   hadoop-mapreduce-examples-*.jar randomwriter
   -Dmapreduce.randomwriter.mapsperhost=4
   -Dmapreduce.randomwriter.bytespermap=67108864
   file://<lustre_path>/hibd_data_<job_id>/rand_in
```

```
$ /tmp/hadoop_install_<job_id>/bin/hadoop jar
   /tmp/hadoop_install_<job_id>/share/hadoop/mapreduce/
   hadoop-mapreduce-examples-*.jar sort
   file://<lustre_path>/hibd_data_<job_id>/rand_in
   file://<lustre_path>/hibd_data_<job_id>/rand_out
```

5.2.5 Shutdown

For stopping Hadoop and clean the used directories, the same command as shown above can be used.

6 Benchmarks

6.1 TestDFSIO

The TestDFSIO benchmark is used to measure I/O performance of the underlying file system. It does this by using a MapReduce job to read or write files in parallel. Each file is read or written in a separate map task and the benchmark reports the average read/write throughput per map.

On a client node, the TestDFSIO write experiment can be run using the following command:

```
$ bin/hadoop jar
   share/hadoop/mapreduce/hadoop-mapreduce-client-jobclient-*-tests.jar
    TestDFSIO -write -nrFiles <nfiles> -fileSize <fsize>
```

This command writes 'nfiles' files and 'fsize' MB each.

To run the same job with MapReduce over Lustre, one additional config parameter must be added in `mapred-site.xml`.

```
 <property>
  <name>test.build.data</name>
  <value><lustre-path-data-dir>/benchmarks/TestDFSIO</value>
 </property>
```

After adding this config parameter, the same command as mentioned above can be used to launch TestDFSIO experiment on top of Lustre.

6.2 Sort

The Sort benchmark uses the MapReduce framework to sort the input directory into the output directory. The inputs and outputs are sequence files where the keys and values are `BytesWritable`. Before running the Sort benchmark, we can use RandomWriter to generate the input data. RandomWriter writes random data to HDFS using the MapReduce framework. Each map takes a single file name as input and writes random `BytesWritable` keys and values to the HDFS sequence file.

On a client node, the RandomWriter experiment can be run using the following command:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    randomwriter -Dmapreduce.randomwriter.bytespermap=<nbytes>
   -Dmapreduce.randomwriter.mapsperhost=<nmaps> <out-dir>
```

This command launches 'nmaps' maps per node, and each map writes 'nbytes' data to 'out-dir'.

On a client node, the Sort experiment can be run using the following command:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    sort -r <nreds> <in-dir> <out-dir>
```

This command launches 'nreds' reduces to sort data from 'in-dir' to 'out-dir'.

The input directory of Sort can be the output directory of RandomWriter.

To run the same job with MapReduce over Lustre, the following commands can be used.

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    randomwriter -Dmapreduce.randomwriter.bytespermap=<nbytes>
   -Dmapreduce.randomwriter.mapsperhost=<nmaps>
   file:///<lustre-path-data-dir>/<out-dir>
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    sort -r <nreds> file:///<lustre-path-data-dir>/<in-dir>
   file:///<lustre-path-data-dir>/<out-dir>
```

6.3 TeraSort

TeraSort is probably the most well-known Hadoop benchmark. It is a benchmark that combines testing the HDFS and MapReduce layers of a Hadoop cluster. The input data for TeraSort can be generated by the TeraGen tool, which writes the desired number of rows of data in the input directory. By default, the key and value size is fixed for this benchmark at 100 bytes. TeraSort takes the data from the input directory and sorts it to another directory. The output of TeraSort can be validated by the TeraValidate tool.

Before running the TeraSort benchmark, we can use TeraGen to generate the input data as follows:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    teragen <nrows> <out-dir>
```

This command writes 'nrows' of 100-byte rows to 'out-dir'.

On a client node, the TeraSort experiment can be run using the following command:

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    terasort <in-dir> <out-dir>
```

This command sorts data from 'in-dir' to 'out-dir'.

The input directory of TeraSort can be the output directory of TeraGen.

To run the same job with MapReduce over Lustre, the following commands can be used.

```
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    teragen <nrows> file:///<lustre-path-data-dir>/<out-dir>
$ bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-*.jar
    terasort file:///<lustre-path-data-dir>/<in-dir>
   file:///<lustre-path-data-dir>/<out-dir>
```

6.4 OHB Micro-benchmarks

The OHB Micro-benchmarks support standalone evaluations of Hadoop Distributed File System (HDFS), Hadoop Database, HBase, Spark, and Memcached (See here). These benchmarks help fine-tune each component avoiding the impact of others.

OSU HiBD-Benchmarks (OHB) have HDFS benchmarks for Sequential Write Latency (SWL), Sequential Read Latency (SRL), Random Read Latency (RRL), Sequential Write Throughput (SWT), Sequential Read Throughput (SRT).

The source code can be downloaded from http://hibd.cse.ohio-state.edu/download/hibd/osu-hibd-benchmarks-0.9.2.tar.gz. The source can be compiled with the help of the Maven. More details on building and running the OHB Micro-benchmark are provided in the README.

A brief description of the benchmark is provided below:

Sequential Write Latency (SWL): This benchmark measures the latency of sequential write to HDFS. The benchmark takes five parameters: file name (`-fileName`), file size (`-fileSize`), block size (`-bSize`), replication factor (`-rep`), and buffer size (`-bufSize`). The mandatory parameters are file name and size (in MB). The output of the benchmark is the time taken to write the file to HDFS. The buffer size indicates the size of the write buffer. HDFS block size and replication factor can also be tuned through this benchmarks. The benchmark also prints the important configuration parameters of HDFS.

Sequential Read Latency (SRL): This benchmark measures the latency of sequential read from HDFS. The benchmark takes two parameters: file name and buffer size. The mandatory parameter is file name. The output of the benchmark is the time taken to read the file from HDFS. The buffer size indicates the size of the read buffer. The benchmark also prints the important configuration parameters of HDFS.

Random Read Latency (RRL): This benchmark measures the latency of random read from HDFS. The benchmark takes four parameters: file name (`-fileName`), file size (`-fileSize`), skip size (`-skipSize`) and buffer size (`-bufSize`). The mandatory parameters are file name and file size. The benchmark first creates a file 2x the file (read) size and then randomly reads from it with a default skip size of 10. The output of the benchmark is the time taken to read the file from HDFS. The buffer size indicates the size of the read buffer. The benchmark also prints the important configuration parameters of HDFS.

Sequential Write Throughput (SWT): This benchmark measures the throughput of sequential write to HDFS. The benchmark takes five parameters: file size (`-fileSize`), block size (`-bSize`), replication factor (`-rep`), buffer size (`-bufSize`), and an output directory (`-outDir`) for the output files. The mandatory parameters are file size (in MB) and the output directory. Linux xargs command is used to launch multiple concurrent writers. File size indicates the write size per writer. A hostfile contains the hostnames where the write processes are launched. The benchmark outputs the total write throughput in MBps. The buffer size indicates the size of the write buffer. HDFS block size and replication factor can also be tuned

through this benchmarks.

Sequential Read Throughput (SRT): This benchmark measures the throughput of sequential read from HDFS. The benchmark takes three parameters: file size (`-fileSize`), buffer size (`-bufSize`), and an output directory (`-outDir`) for the output files. The mandatory parameters are file size (in MB) and the output directory. Linux xargs command is used to launch multiple concurrent readers. File size indicates the write size per reader. A hostfile contains the hostnames where the write processes are lauched. The benchmark outputs the total read throughput in MBps. The buffer size indicates the size of the read buffer.