RDMA for Apache Hadoop 0.9.9 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: August 19, 2014

Contents

1 Overview of the RDMA for Apache Hadoop Project

RDMA for Apache Hadoop is a high-performance design of Hadoop over RDMA-enabled Interconnects. This version of RDMA for Apache Hadoop 0.9.9 is based on Apache Hadoop 1.2.1. This file is intended to guide users through the various steps involved in installing, configuring, and running RDMA for Apache Hadoop over InfiniBand. 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 0.9.9 are listed below.

- Based on Apache Hadoop 1.2.1
- High performance design with native InfiniBand and RoCE support at the verbs level for HDFS, MapReduce, and RPC components
- Compliant with Apache Hadoop 1.2.1 APIs and applications
- Easily configurable for native InfiniBand, RoCE, and the traditional sockets based support (Ethernet and InfiniBand with IPoIB)
- On-demand connection setup
- HDFS over native InfiniBand and RoCE
  - RDMA-based write
  - RDMA-based replication
  - Parallel replication support
- MapReduce over native InfiniBand and RoCE
  - RDMA-based shuffle
  - Prefetching and caching of map output
  - In-memory merge
  - Advanced optimization in overlapping
    - map, shuffle, and merge
    - shuffle, merge, and reduce
- RPC over native InfiniBand and RoCE
  - JVM-bypassed buffer management
  - RDMA or send/recv based adaptive communication
  - Intelligent buffer allocation and adjustment for serialization

- Tested with
  - Native Verbs-level support with Mellanox InfiniBand adapters (DDR, QDR and FDR)
  - RoCE support with Mellanox adapters
  - Various multi-core platforms
  - Different file systems with disks and SSDs

3 Installation Instructions

3.1 Prerequisites

Prior to the installation of RDMA for Apache Hadoop, 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 1.7 and later.

In order to use the RDMA-based features provided with RDMA for Apache Hadoop, 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 from `http://hibd.cse.ohio-state.edu/download/hibd/rdma-hadoop-0.9.9-bin.tar.gz`.

3.3 Basic Configuration

Most of the configuration files are in the directory “rdma-hadoop-0.9.9/conf”. Steps to configure RDMA for Apache Hadoop include:

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

```
export JAVA_HOME=/opt/java/1.7.0
```

2. Configure `core-site.xml` file. RDMA for Apache Hadoop 0.9.9 supports three different modes: IB, RoCE, and TCP/IP.

   Configuration of the IB mode:

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://node001:9000</value>
    <description>URL of NameNode</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.default.name</name>
    <value>hdfs://node001:9000</value>
    <description>URL of NameNode</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.default.name</name>
    <value>hdfs://node001:9000</value>
    <description>URL of NameNode</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.name.dir</name>
    <value>/home/hadoop/rdma-hadoop-0.9.9/HadoopName</value>
    <description>Path on the local filesystem where the NameNode
        stores the namespace and transactions logs</description>
  </property>

  <property>
    <name>dfs.data.dir</name>
    <value>/data01,/data02</value>
    <description>Lists of paths on the local filesystem of a
        DataNode where it should store its block</description>
  </property>
</configuration>
```

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

```
<configuration>
  <property>
    <name>mapred.job.tracker</name>
    <value>node001:9001</value>
    <description>Host or IP and port of JobTracker</description>
  </property>

  <property>
    <name>mapred.system.dir</name>
    <value>/hadoop/mapred/system/</value>
    <description>Path on the HDFS where the MapReduce framework
        stores system files</description>
  </property>
</configuration>
```

5. Configure `masters` file. Write the master hostname in this file.

```
node001
```

6. Configure `slaves` file. List all slave hostnames in this file.

```
node002
node003
```

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

3.4 Advanced Configuration

Some advanced features in RDMA for Apache Hadoop 0.9.9 can be manually enabled by users. Steps to configure these features in RDMA for Apache Hadoop 0.9.9 include:

1. Enable parallel replication in HDFS by configuring `hdfs-site.xml` file. By default, RDMA for Apache Hadoop 0.9.9 will choose the pipeline replication mechanism.

```
<configuration>
  <property>
    <name>dfs.replication.parallel</name>
    <value>true</value>
    <description>Enable the parallel replication feature in HDFS.
        Default value of dfs.replication.parallel is false.
        </description>
  </property>
</configuration>
```

2. Enable disk-based shuffle in MapReduce by configuring `mapred-site.xml` file. By default, RDMA for Apache Hadoop 0.9.9 assumes that disk-based shuffle is enabled. We encourage our users to disable this parameter if they feel that the local disk performance in their cluster is not good enough.

```
<configuration>
  <property>
    <name>mapred.disk.shuffle.enabled</name>
    <value>true</value>
    <description>Enable disk-based shuffle in MapReduce. Default
        value of mapred.disk.shuffle.enabled is true. </description>
  </property>
</configuration>
```

4 Basic Usage Instructions

RDMA for Apache Hadoop 0.9.9 has management operations similar to default Apache Hadoop 1.2.1. This section lists several of them for basic usage.

4.1 Startup

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

   ```
   $ bin/hadoop namenode -format
   ```

2. Start HDFS with the following command:

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

3. Start MapReduce with the following command:

   ```
   $ bin/start-mapred.sh
   ```

4. Start HDFS+MapReduce with the following command:

   ```
   $ bin/start-all.sh
   ```

4.2 Basic Commands

1. Use the following command to manage HDFS:

   ```
   $ bin/hadoop dfsadmin
   Usage: java DFSAdmin
           [-report]
           [-safemode enter | leave | get | wait]
           [-saveNamespace]
           [-refreshNodes]
           [-finalizeUpgrade]
           [-upgradeProgress status | details | force]
           [-metasave filename]
           [-refreshServiceAcl]
           [-refreshUserToGroupsMappings]
           [-refreshSuperUserGroupsConfiguration]
           [-setQuota <quota> <dirname>...<dirname>]
           [-clrQuota <dirname>...<dirname>]
           [-setSpaceQuota <quota> <dirname>...<dirname>]
           [-clrSpaceQuota <dirname>...<dirname>]
           [-setBalancerBandwidth <bandwidth in bytes per second>]
           [-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/hadoop dfsadmin -report
```

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

```
$ bin/hadoop fs
Usage: java FsShell
        [-ls <path>]
        [-lsr <path>]
        [-du <path>]
        [-dus <path>]
        [-count[-q] <path>]
        [-mv <src> <dst>]
        [-cp <src> <dst>]
        [-rm [-skipTrash] <path>]
        [-rmr [-skipTrash] <path>]
        [-expunge]
        [-put <localsrc> ... <dst>]
        [-copyFromLocal <localsrc> ... <dst>]
        [-moveFromLocal <localsrc> ... <dst>]
        [-get [-ignoreCrc] [-crc] <src> <localdst>]
        [-getmerge <src> <localdst> [addnl]]
        [-cat <src>]
        [-text <src>]
        [-copyToLocal [-ignoreCrc] [-crc] <src> <localdst>]
        [-moveToLocal [-crc] <src> <localdst>]
        [-mkdir <path>]
        [-setrep [-R] [-w] <rep> <path/file>]
        [-touchz <path>]
        [-test -[ezd] <path>]
        [-stat [format] <path>]
        [-tail [-f] <file>]
        [-chmod [-R] <MODE[,MODE]... | OCTALMODE> PATH...]
        [-chown [-R] [OWNER][:[GROUP]] PATH...]
        [-chgrp [-R] GROUP PATH...]
        [-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 can use the following command to list directory contents of HDFS:

```
$ bin/hadoop fs -ls /
```

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

```
$ bin/hadoop 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>]

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/hadoop job -list-active-trackers
```

4.3 Shutdown

1. Stop HDFS with the following command:

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

2. Stop MapReduce with the following command:

   ```
   $ bin/stop-mapred.sh
   ```

3. Stop HDFS+MapReduce with the following command:

   ```
   $ bin/stop-all.sh
   ```

5 Benchmarks

5.1 TestDFSIO

The TestDFSIO benchmark is used to measure I/O performance of HDFS. 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 hadoop-test-*.jar TestDFSIO -write -nrFiles 2
   -fileSize 1000
```

This command writes 2 files 1,000MB each.

5.2 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 hadoop-examples-*.jar randomwriter <out-dir>
   [<configuration file>]
```

More details about configurations can be found in:

http://wiki.apache.org/hadoop/RandomWriter.

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

```
$ bin/hadoop jar hadoop-examples-*.jar sort [-m <#maps>] [-r
   <#reduces>] <in-dir> <out-dir>
```

The "in-dir" of Sort can be the "out-dir" of RandomWriter. More details can be found in:

http://wiki.apache.org/hadoop/Sort.

5.3 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 hadoop-examples-*.jar teragen <number of 100-byte
   rows> <output dir>
```

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

```
$ bin/hadoop jar hadoop-examples-*.jar terasort <input dir> <output
   dir>
```

The "in-dir" of TeraSort can be the "out-dir" of TeraGen.