RDMA for Apache Spark 0.9.4 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: March 26, 2017

Contents

1 Overview of the RDMA for Apache Spark Project

RDMA for Apache Spark is a high-performance design of Spark over RDMA-enabled Interconnects. This version of RDMA for Apache Spark 0.9.4 is based on Spark 2.1.0. This file is intended to guide users through the various steps involved in installing, configuring, and running RDMA for Apache Spark over InfiniBand. If there are any questions, comments or feedback regarding this software package, please post them to rdma-spark-discuss mailing list (rdma-spark-discuss@cse.ohio-state.edu).

2 Features

High-level features of RDMA for Apache Spark 0.9.4 are listed below. New features and enhancements compared to 0.9.3 release are marked as (NEW).

- (NEW) Based on Apache Spark 2.1.0
- Built with Apache Hadoop 2.7.3
- High performance design with native InfiniBand and RoCE support at the verbs level for Spark
  - RDMA-based data shuffle
  - SEDA-based shuffle architecture
  - Support pre-connection, on-demand connection, and connection sharing
  - Non-blocking and chunk-based data transfer
  - Off-JVM-heap buffer management
- (NEW) Compliant with Apache Spark 2.1.0 APIs and applications
- RDMA support for Spark SQL
- Integration with HHH in RDMA for Apache Hadoop 2.x
- Easily configurable for native InfiniBand, RoCE, and the traditional sockets based support (Ethernet and InfiniBand with IPoIB)
- Tested with
  - Mellanox InfiniBand adapters (DDR, QDR, FDR, and EDR)
  - RoCE support with Mellanox adapters
  - Various multi-core platforms
  - RAM Disks, SSDs, and HDDs

3 Setup Instructions

3.1 Prerequisites

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

3.2 Download

The latest version of RDMA for Apache Spark package can be downloaded from http://hibd.cse.ohio-state.edu/download/hibd/rdma-spark-0.9.4-bin.tar.gz. By default, the package is built with Apache Hadoop 2.7.3.

If you want to run RDMA for Apache Spark package with HHH in RDMA for Apache Hadoop 2.x, you need to download RDMA for Apache Hadoop 2.x from http://hibd.cse.ohio-state.edu/download/hibd/rdma-hadoop-2.x-1.1.0-bin.tar.gz.

We strongly recommend users to run RDMA for Apache Spark package with HHH in RDMA for Apache Hadoop 2.x, since this combination gives you the best performance.

3.3 Installing RDMA for Apache Spark Package

The RDMA for Apache Spark can be installed using the intergrated distribution. Following steps can be used to install the integrated RDMA for Apache Spark package.

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

   ```
   tar zxf rdma-spark-0.9.4-bin.tar.gz
   ```

2. Change directory to rdma-spark-0.9.4-bin

   ```
   cd rdma-spark-0.9.4-bin
   ```

If you want to run RDMA for Apache Spark package with HHH in RDMA for Apache Hadoop 2.x, you need to install RDMA for Apache Hadoop 2.x also. The detail steps of installing RDMA for Apache Hadoop 2.x can be found at:
http://hibd.cse.ohio-state.edu/static/media/rdma-hadoop/rdma-hadoop-2.x-1.1.0-userguide.pdf.

3.4 Basic Configuration

The configuration files can be found in the directory `rdma-spark-0.9.4-bin/conf` of the intergrated RDMA for Apache Spark package. RDMA for Apache Spark 0.9.4 supports three different modes: IB, RoCE, and TCP/IP. Steps for enabling these modes in the Spark Standalone mode are shown below.

1. Configure `spark-env.sh` file to conform to the configuration parameters of default spark. More information can be found at `http://spark.apache.org/docs/latest/configuration.html`. Make sure `spark-env.sh` has at least SPARK_MASTER_IP environment variable configured. A simple working spark-env.sh may look like:

```
export SPARK_LOCAL_IP=`hostname -s`
export SPARK_MASTER_IP=node01
export SPARK_WORKER_MEMORY=64g
export SPARK_WORKER_CORES=8
export SPARK_WORKER_INSTANCES=1
export SPARK_DAEMON_MEMORY=2g
```

2. Configure `slaves` file. List all slave hostnames in this file, one per line.

```
node002
node003
```

3. Configure `spark-defaults.conf`. The three different modes supported by RDMA for Apache Spark 0.9.4 can be configured here. All other parameters can be configured similar to default spark. More information can be found at `http://spark.apache.org/docs/latest/configuration.html`

3.4.1 Configuration of the IB mode

This mode basically runs RDMA-enabled mode for InfiniBand. Steps to configure RDMA for Apache Spark to run in IB mode:

```
spark.ib.enabled  true
hadoop.ib.enabled  false
spark.executor.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
spark.driver.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
```

IB mode is used as the default mode if none of the others are explicitly enabled.

If you want to run RDMA for Apache Spark package with HHH in RDMA for Apache Hadoop 2.x with IB mode, you have to change "hadoop.ib.enabled" to true. The full configuration is as follows:

```
spark.ib.enabled  true
hadoop.ib.enabled  true
spark.executor.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
spark.driver.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
```

3.4.2 Configuration of the RoCE mode

Steps to configure RDMA for Apache Spark to run in RoCE mode:

```
spark.roce.enabled  true
hadoop.roce.enabled false
spark.executor.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
spark.driver.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
```

If you want to run RDMA for Apache Spark package with HHH in RDMA for Apache Hadoop 2.x with RoCE mode, you have to change "hadoop.roce.enabled" to true. The full configuration is as follows:

```
spark.roce.enabled  true
hadoop.roce.enabled true
spark.executor.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
spark.driver.extraLibraryPath
   $SPARK_HOME/lib/native/Linux-amd64-64:$HADOOP_HOME/lib/native
```

3.4.3 Configuration of the TCP/IP mode

This mode basically runs default spark. Steps to configure RDMA for Apache Spark to run in TCP/IP mode:

```
spark.ib.enabled   false
hadoop.ib.enabled   false
```

3.5 Advanced Configuration

In this section, we describe some advanced features in RDMA for Apache Spark 0.9.4 that can be manually enabled by users, and steps to configure them.

3.5.1 Support Running with SparkSQL

RDMA for Apache Spark 0.9.4 enables running SQL queries over Hive tables. We also facilitate employing RDMA-enhanced Heterogeneous HDFS design or HHH to run in the SparkSQL mode. In order to run SparkSQL over HHH, the following parameters needs to be configured:

1. Add the following configurations to `hdfs-site.xml` to enable RDMA Spark to run SQL queries over data stored as Hive tables:

```
<property>
       <name>dfs.rdma.spark.sql</name>
       <value>true</value>
       <description>Enable SparkSQL mode over HHH.</description>
 </property>
```

More details on RDMA for Apache Hadoop 2.x can be found here.

3.5.2 Block Transfer Chunk Size

RDMA for Apache Spark 0.9.4 enables chunk-based data transfer for optimal performance. The size of the chunk can be tuned by users. Larger chunk sizes are recommended if ample memory is available. This parameter can be tuned to suit the underlying network and cluster configuration. The chunk size for data transfer can be configured as follows:

```
spark.shuffle.rdma.chunk.size  <chunk-size-in-bytes>
```

The chunk size is set to 512 KB (524288) by default.

3.5.3 Memory Slab Size

RDMA for Apache Spark 0.9.4 supports slab-based memory management off-JVM-heap. Larger slab sizes are recommended if ample memory is available. This parameter can be tuned to suit the underlying cluster configuration. The memory slab size can be configured as follows:

```
spark.shuffle.rdma.memory.slab.size  <chunk-size-in-bytes>
```

The slab size is set to 32 MB (33554432) by default.

3.5.4 Pre-Connection

RDMA for Apache Spark 0.9.4 supports pre-connection to hide connection overhead. Enable this feature is recommended if your application is communication-intensive. This parameter can be tuned to suit the workload characteristics. The pre-connection can be enabled as follows:

```
spark.shuffle.rdma.pconn.enabled  true
```

This feature is enabled by default.

4 Basic Usage Instructions

RDMA for Apache Spark 0.9.4 can be deployed in either stand-alone mode, with YARN or Apache Mesos, similar to default Apache Spark 2.1.0. This section lists steps for stand-alone mode.

4.1 Startup

To start RDMA for Apache Spark 0.9.4 in stand-alone mode:

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

4.2 Basic Commands

Applications can be submitted using with `spark-submit` script as follows:

```
./bin/spark-submit \
  --class <main-class>
  --master spark://<master-node-hostname>:7077 \
  --deploy-mode <deploy-mode> \
  --conf <key>=<value> \
  ... # other options
  <application-jar> \
  [application-arguments]
```

Here is an example:

```
./bin/spark-submit \
  --class org.apache.spark.examples.GroupByTest \
  --master spark://node001:7077 \
  --executor-memory 16G \
  --num-executors 100\
  /path/to/examples.jar \
  32 131072 4092 32
```

4.3 Shutdown

To stop RDMA for Apache Spark 0.9.4 running in stand-alone mode:

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

5 Running Spark Benchmarks

5.1 OHB Micro-benchmarks

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

5.1.1 Building Spark Micro-benchmarks in OHB

The OHB 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 (version 3.3.0 or higher) as follows:

```
$ mvn clean package
```

The OHB Spark micro-benchmarks jar is built and installed based on Apache Spark version 2.1. It can be found in $OHB_HOME/spark/target/ohb-spark-0.9.2.jar. More details on building and running the OHB Micro-benchmarks are provided in the README.spark.txt.

5.1.2 Running Spark Micro-benchmarks in OHB

To test the performance with transformations with wide dependencies we provide GroupByTest and SortByTest.

(1) GroupBy: To run Spark stand-alone micro-benchmark that uses the groupByKey Spark transformation, we can use the following command:

```
$OHB_HOME=<OHB-INSTALL-PATH> MASTER=spark://$MASTER_IP:7077
   $OHB_HOME/ohb-run-example edu.osu.hibd.spark.GroupByTest
   [numMappers] [numKVPairs] [ValueSize] [numReducers]
```

For example to run 24GB GroupByTest on a 12 core machine with 8 spark workers (96 * 65536 * 4096 (4 bytes for key by default) gives 24 GB), one can issue the command:

```
$OHB_HOME=<OHB-INSTALL-PATH> MASTER=spark://$MASTER_IP:7077
   $OHB_HOME/ohb-run-example edu.osu.hibd.spark.GroupByTest 96 65536
   4092 96
```

(2) SortBy: To run Spark stand-alone micro-benchmark that uses the sortByKey Spark transformation, we can use the following command:

```
$OHB_HOME=<OHB-INSTALL-PATH> MASTER=spark://$MASTER_IP:7077
   $OHB_HOME/ohb-run-example edu.osu.hibd.spark.SortByTest
   [numMappers] [numKVPairs] [ValueSize] [numReducers]
```

For example to run 24GB GroupByTest on a 12 core machine with 8 spark workers (96 * 65536 * 4096 (4 bytes for key by default) gives 24 GB), one can issue the command:

```
$OHB_HOME=<OHB-INSTALL-PATH> MASTER=spark://$MASTER_IP:7077
   $OHB_HOME/ohb-run-example edu.osu.hibd.spark.SortByTest 96 65536
   4092 96
```

5.2 HiBench

This is the bigdata micro benchmark suite which provides typical micro workloads. HiBench version

used is HiBench-6.0. The HiBench package can be downloaded from here https://github.com/intel-hadoop/HiBench. Next step is to build it using HiBench/bin/build-all.sh

Please refer to the Readme for setup configurations and other details in the directory file HiBench/README.md

To run RDMA for Apache Spark, one can include the following configuration in the directory file 'HiBench/conf/spark.conf'.

- spark.executor.extraLibraryPath $SPARK_HOME/lib/native/Linux-amd64-64
- spark.driver.extraLibraryPath $SPARK_HOME/lib/native/Linux-amd64-64

Other specific Spark configuration (Rdma/Non-Rdma) can go here.

To run the benchmark workloads, need to start Hadoop and Spark first. Then one can go to the specific workload dir to generate input data and run the workload. Please refer to the HiBench/README.md for details on where to view the benchmark reports. Following are some of the benchmark workloads that can be run.

5.2.1 Sort

This workload sorts its text input data, which is generated using RandomTextWriter. prepare.sh generates the input data and run.sh runs Sort on the input data

```
HiBench/bin/workloads/micro/sort/prepare/prepare.sh
HiBench/bin/workloads/micro/sort/spark/run.sh
```

5.2.2 TeraSort

Its input data is generated by Hadoop TeraGen example program. prepare.sh generates the input data and run.sh runs TeraSort on the input data

```
HiBench/bin/workloads/micro/terasort/prepare/prepare.sh
HiBench/bin/workloads/micro/terasort/spark/run.sh
```

5.2.3 PageRank

This workload benchmarks PageRank algorithm implemented in Spark-MLLib/Hadoop (a search engine ranking benchmark included in pegasus 2.0) examples. The data source is generated from Web data whose hyperlinks follow the Zipfian distribution. prepare.sh generates the input data and run.sh runs PageRank on the input data

```
HiBench/bin/workloads/websearch/pagerank/prepare/prepare.sh
HiBench/bin/workloads/websearch/pagerank/spark/run.sh
```

5.3 BigDataBench SparkSQL

The BigDataBench suite (v3.0) can be used to run Hive SQL queries over Spark. These benchmarks typically involve benchmarks with queries such as Select, Aggregation, Join, etc. The BigDataBench package can be downloaded from here http://prof.ict.ac.cn/BigDataBench/. Using the BigDataBench suite involves two steps: (1) table data generation using the BDGS tool, and, (2) running the query with SparkSQL. Detailed steps can be found here.

Below is a high-level overview of the steps involved in starting up the SparkSQL cluster and running BigDataBench over it:

1. Setup and start HDFS cluster. Steps to install and setup Apache Hadoop 2.7.3/RDMA-Hadoop-2.x can be found here.
2. Install and configure the Hive Metastore (version 1.2.1 or above). The hive binary tarball can be downloaded at https://hive.apache.org/downloads.html. The steps to set up are as follows:
   (a) Configuring a PostgreSQL/MySQL Database for the Hive Metastore (to store the metadata for Hive tables).
   (b) Setup hive-site.xml to contain the ConnectionURL to the PostgreSQL/MySQL database and the Hive Metastore URIs:

```
<property>
  <name>javax.jdo.option.ConnectionURL</name>
  <value>jdbc:mysql://myhost/metastore</value>
  <description>the URL of the PostgreSQL/MySQL
     database</description>
</property>

<property>
  <name>hive.metastore.uris</name>
  <value>thrift://<n.n.n.n>:9083</value>
  <description>IP address (or fully-qualified domain name) and
     port of the metastore host</description>
</property>
```

   (c) Install and configure Zookeeper (version 3.4.0) in stand-alone mode on the master mode. The Zookeeper package can be downloaded from https://zookeeper.apache.org/.
   (d) Start the Hive Metastore service on the master node (in the background).

```
$HIVE_HOME/bin/hive --service metastore &
```

   More details can be found at https://cwiki.apache.org/confluence/display/Hive/GettingStarted.
3. Download the Postgres/MySQL JDBC jar corresponding to your database and Java versions. Postgres JDBC jars can be found at https://jdbc.postgresql.org/download.html and install it in $SPARK_HOME/lib/.

4. Setup and start Spark in Stand-alone mode as instructed in Section 4.1.
5. For running BigDataBench, generate data using the BDGS tool.
6. Start the `spark-sql` shell to load data generated and run queries. The queries can be found in `$BIGDATABENCH_HOME/Interactive_Query`.

6 Troubleshooting with RDMA for Apache Spark

If you are experiencing any problems with RDMA for Apache Spark package, please feel free to contact us by sending an email to rdma-spark-discuss@cse.ohio-state.edu.