MPJ Express: An Implementation of MPI in Java

MPJ-YARN Runtime User Guide

16th March 2015

Document Revision Track

Version	Updates	By
1.0	Initial version document	Hamza Zafar

1 Pre requisites

- Apache Hadoop v2.3.0 and above
- MPJ Express v0.44

2 Apache Hadoop Configuration

2.1 Single Node Configuration

To set up and configure a single-node Hadoop installation please follow the link given below:

http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/SingleCluster.html

2.2 Cluster Configuration:

To configure and manage non-trivial Hadoop clusters consisting of several nodes, please follow the link given below:

http://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-common/ClusterSetup.html

Expectations:

- You have successfully configured the necessary environment variables, such as $HADOOP_HOME
- You have configured the configuration files in $HADOOP_CONF_DIR
- The resourcemanager , nodemanager, datanode and namenode daemons are running

3 Getting Started with MPJ Express

This section shows how MPJ Express programs can be executed in the Hadoop YARN cluster.

3.1 Installing MPJ Express

This section outlines steps to download and install MPJ Express software.

1. Download MPJ Express v0.44 and unpack it
2. Set MPJ_HOME and PATH variables
    a. `export MPJ_HOME=/path/to/mpj/`
    b. `export PATH=$MPJ_HOME/bin:$PATH`

    These lines may be added to ".bashrc" file. However make sure that the shell in which you are setting variables is the 'default' shell. For example, if your default shell is 'bash', then you can set environment variables in .bashrc. If you are using 'tcsh' or any other shell, then set the variables in the respective files.
3. Create a new working directory for MPJ Express programs. This document assumes that the name of this directory is mpjusr.
4. To compile the MPJ Express Hadoop runtime :

```
cd $MPJ_HOME;
ant hadoop
```

3.2 Compiling User Applications

This section shows how to compile a simple Hello World parallel Java program.

1. Write Hello World parallel Java program and save it as HelloWorld.java

```
import mpi.*;

public class HelloWorld {

       public static void main(String args[]) throws Exception {
              MPI.Init(args);
              int me = MPI.COMM_WORLD.Rank();
              int size = MPI.COMM_WORLD.Size();
              System.out.println("Hi from <"+me+">");
              MPI.Finalize();
       }
}
```

2. Compile: `javac -cp .:$MPJ_HOME/lib/mpj.jar HelloWorld.java`

3. Create HelloWorld jar file: `jar cf HelloWorld.jar HelloWorld.class`

3.3 Running MPJ Express programs in Hadoop cluster

1. Assuming the user has successfully carried out Section 3.1 and Section 3.2.

2. Running HelloWorld

```
mpjrun.sh -yarn -np 2 -dev niodev -wdir /export/home/hamza.zafar/mpjusr/ -jar
/export/home/hamza.zafar/mpjusr/HelloWorld.jar HelloWorld
```

Note:

- The main class name should be mentioned after the jar file name
- Currently YARN runtime supports the niodev device only

Command line arguments:

No.	CMD	Optional	Description
1	-yarn	NO	Specifies the mpjrun module to invoke YARN runtime Usage: -yarn
2	-np	NO	Specifies the number of processes to launch Usage: -np <number-of-processes>
3	-dev	NO	Specifies the MPJ Express device name, currently YARN runtime supports the Ethernet based niodev Usage: -dev niodev
4	-wdir	NO	Specifies the current working directory Usage: -wdir <path-to-working-directory>
5	-jar	NO	Specifies the jar file containing the MPJ Express program. The -jar should be the last command line argument. The main class name should be mentioned after the jar file name, all other arguments after the main class

			are passed as arguments to the user's main class Usage: -jar <path-to-jar> <main-class-name> <space-separated-arguments-to-main-class>
6	-amMem	Yes	Specifies the Application Master container's memory, if the option is not mentioned the “2048mb” memory is used Usage: -amMem <amount-of-memory-e.g 512 or 1024>
7	-amCores	Yes	Specifies the Application Master container's virtual cores, if the option is not mentioned the “1” virtual core is used Usage: -amCores <number-of-cores-should-be-greater-then-zero>
8	-containerMem	Yes	Specifies the MPJ container's memory, if the option is not mentioned the “1024mb” memory is used Usage: -containerMem <amount-of-memory-e.g 512 or 1024>
9	-containerCores	Yes	Specifies the MPJ container's virtual cores, if the option is not mentioned the “1” virtual core is used Usage: -containerCores <number-of-cores-

			should-be-greater-then-zero>
10	-yarnQueue	Yes	Specifies the YARN's scheduling queue, if the option is not mentioned the “default” queue is used Usage: -yarnQueue <queue-name>
11	-appName	Yes	Specifies the application name Usage: -appName <application-name>
12	-amPriority	Yes	Specifies the AM container's priority, if the option is not mentioned the “0” is set as priority Usage: -amPriority <numerical value greater then or equals to 0>
13	-mpjContainerPriority	Yes	Specifies the MPJ container's priority, if the option is not mentioned the “0” is set as priority Usage: -mpjContainerPriority <numerical value greater then or equals to 0>
14	-hdfsFolder	Yes	Specifies the hdfs folder where the jar files will be uploaded. The specified folder should be first created in hdfs by the user. If the hdfsFolder option is not mentioned the “/” root folder is used for temporary uploading files. Usage: -hdfsFolder <hdfs-folder-path-e.g /mpj-

			yarn>
15	-debugYarn	Yes	Specifies the debug flag. Usage: -debugYarn

3. Running HelloWorld using the above mentioned arguments

```
mpjrun.sh -yarn -np 2 -dev niodev -wdir /export/home/hamza.zafar/mpjusr/ -amMem 512
-amCores 1 -containerMem 512 -containerCores 1 -yarnQueue default -appName
MPJYarn -amPriority 1 -mpjContainerPriority 1 -hdfsFolder /mpj-yarn/ -debugYarn -jar
/export/home/hamza.zafar/mpjusr/HelloWorld.jar HelloWorld
```

Contact:

Hamza Zafar (11bscshzafar@seecs.edu.pk)

Aamir Shafi (aamir.shafi@seecs.edu.pk)