15-640 Distributed Systems

Lab3 – Map Reduce Engine System Administrator Manual

-Amey Ghadigaonkar

7/14/2014

**15 - 640 Distributed Systems**

**Lab 3 – Map Reduce Engine – System Admin Manual**

**-Amey Ghadigaonkar.**

**Index:**

1. System Requirements
2. Configuration Steps
   1. Generic K-V pairs
   2. Low latency for launching mappers/reducers through pre-launched JVMs
   3. High failure tolerance for datanodes through task queues
3. Confirm Successful Configuration
4. How to:
   1. Start the system
   2. SAFELY Stop the system
5. **System requirements:**

**Java + Linux (for using build file)**

Should preferably have at least 2Gb of space allocated to the java process.

1. **Configuration steps:**

Open the conf.Constants file. It contains the available configuration options.

* RMI\_SERVICE\_NAME is the name that the server will bind and the datanodes will search for in the registry
* REPLICATION\_FACTOR is assumed to be 3 at ALL TIMES and is not configurable

It defines the number of times a file is replicated in the DFS

* MIN\_BLOCK\_SIZE is the minimum size of file blocks i.e. smaller parts of the file
* FRAME\_SIZE is frame size for communication
* MAX\_REDUCER\_SIZE maximum size that a reducer is allowed to handle
* NUMBER\_OF\_CORES is the default number of cores which is assumed 6. In case it is run on Andrew, this value is overridden by value inside /proc/cpuinfo

1. **Confirm successful configuration**

The system should start normally if all configuration options are correct.

1. **Start registry server**

registryserver &

1. **Start NameNode**

java namenode.NameNode

1. **Start DataNodes**

java datanode.DataNode <LocalRootPath> <LocalSourceFilesPath>

e.g. java datanode.DataNode C:/Temp/1/

LocalRootPath **is where we want all the MR operations to happen**

LocalSourceFilesPath is where all source files are stored

**NOTE: all datanodes if launched on same machine must have different rootpaths!**

1. **Operations:**
2. Send local file to DFS

localtohdfs <localFilePath> <hdfs\_file\_path>

e.g. C:/Temp/bible.txt /bible

1. Send DFS file to local

hdfstolocal <localFilePath> <hdfs\_file\_path>

e.g. C:/Temp/newLocalFile.txt /bible

Also, the DFS paths should exist in the FS e.g. /output/a/ , then /output/a should exist in the DFS.

1. startjob

startJob <jobName> <jarFileName.jar> <mapperClass> <reducerClass> <inputFile> <outputPath>

the filename is the DFS file name. Thus you have to convert local to DFS first and then use that file name.

For e.g., To start a sample job:

registryserver &

java namenode.NameNode

java datanode.DataNode C:/Temp/1/ C:/Users/Amey/workspace/lab3

java datanode.DataNode C:/Temp/2/ C:/Users/Amey/workspace/lab3

java datanode.DataNode C:/Temp/3/ C:/Temp/1/ C:/Users/Amey/workspace/lab3

Now, on one of the datanodes write the following (assuming the local files exist):

localtohdfs C:/Temp/bible.txt /bible

localtohdfs C:/Temp/exp.jar /expjar

mkdir /a/

startjob job1 /expjar temperaturetest.Mapper1 temperaturetest.Reducer1 /bible /a/

**At any point, type in “help” in the datanode to get help.**

**For namenode,**

You need to type in 5 to see the datanodes registered into the namenodes via RMI

You need to type in 6 to see the jobs running in the system and monitor them.

Jobs are started and managed at the datanodes. Namenode only coordinated jobs according to my design.

To safely stop jobs by closing all launched processes, type in “destroy” at each datanode before closing them with “ctrl+C”.

The output of the mapreduce job is in the **‘out’** file in the directory specified in the startjob command.