Mapreduce Simple Tutorial & Homework

Wen-Chieh Wu

May 18, 2014

HDFS

Hadoop Distributed File System : Designed to reliably store very large files across machines in a large cluster

- hadoop fs -ls
  List files in the /user/[your account name]
  ex: /user/r01922003
- hadoop fs -mkdir [dir]
  Create folder in the /user/[your account name]/[dir]
- hadoop fs -rm -r [dir]
  Delete folder /user/[your account name]/[dir]

HDFS(cont.)

- hadoop fs -put [localfile] [dir]/[file]
  Push files [localfile] to /user/[your account name]/[dir]/[file]
- hadoop fs -get [dir]/[file] [localfile]
  Get files from /user/[your account name]/[dir]/[file] to [localfile]
- hadoop fs -getmerge [dir] [localfile]
  From [dir], get the merged result

MapReduce

Programming model for processing large data sets with a parallel, distributed algorithm on a cluster.
Written in Java, usually we divided to three .java files, take wordcount as an example

- WordCount.java
- WordCountMapper.java
- WordCountReducer.java

MapReduce(cont.)

How to compile? shortcut: $ make

```
$ javac -classpath ‘yarn classpath‘ WordCount.java
WordCountMapper.java WordCountReducer.java
$ jar -cvf WordCount.jar WordCount.class
WordCountMapper.class WordCountReducer.class
```

How to execute? shortcut: $ make run

```
$ hadoop jar WordCount.jar WordCount inputdir outputdir
```

How to know the result? shortcut: $ make get

```
$ hadoop fs -getmerge outputdir result
$ cat result
```

MapReduce(cont.)

Notes

- The cluster executes one application a time, you need to wait for it.
- Jobtracker: http://140.112.2.92:8088/

Examples

- helloworld
  Output Hello World message
- stringsort
  Sort files of strings
- wordcount
  Classic example that calculate the word count of files

Examples(cont.)

Let's take a look at the examples
$ cp -r /tmp/mapreduce.example .

Homework

Use mapreduce to sort file size for files, from min. to max.

Homework(cont.)

The problem can be regarded as a file size sorting problem, the first column is the file size and the latter column is the file name.

Input

100	apple
33	orange
100	banana
10	pineapple

Output

10	pineapple
33	orange
100	apple,banana

Homework(cont.)

There are four dataset

- /data/easy, 16 sec
- /data/tiny, 21 sec
- /data/small, 1 min 19 sec
- /data/big, 12 min 9 sec

Homework(cont.)

Requirement

- Processing 4 dataset without error, output format is right and each part is sorted by file size.(70%)
- The file name is sorted by alphabet.(15%)
- The "-getmerge" result is sorted by file size.(15%)

Homework(cont.)

Hints

- The range of the file size is between 0 and 2147483647.
- There are five reducers only.
- Please research "MapReduce Partitioner", it will be useful for the requirement 3.
- We will provide "IteratorSort.java", it can help you sort the file names.

Homework(cont.)

IteratorSort.java Usage, remember to compile it with other .java files

```
1 Text word = new Text();
2 Iterator<Text> it = values.iterator();
3 while(it.hasNext()){
4   word.set(it.next());
5 }
```

```
1 Text word = new Text();
2 Iterator<String> it_str =
IteratorSort.Sort(values.iterator());
3 while(it_str.hasNext()){
4   word.set(it_str.next());
5 }
```

Homework(cont.)

In the zip file

- pdp.hw3.mapreduce.pdf
- IteratorSort.java
- easy.ans
- tiny.ans

you can use “diff” to check the answer

Homework(cont.)

Please upload your program to CEIBA, in the zip file contain a folder of your id and the program *.java, a readme.txt telling me how to compile & execute the program.
EX:

- r01922003
  - *.java
  - readme.txt

Homework(cont.)

Grading

- Deadline: 6/4 14:00
- 10% off for late submission each week, latest submission date: 6/25
- If your program executes for too long, you may lose some grades.