TP1 : Introduction to Map-Reduce

2013/2014

1 Running a local job

Download the eclipse project which contains a wordcount example. Execute it locally to ensure that everything is properly configured.

2 Connecting to the Hadoop cluster

A Hadoop cluster was deployed for this exercise. It contains 1 master and 4 slaves. To test your programs in a grid environment, you should use this cluster. The IP address of the master is 152.77.78.100, and the Hadoop executables are in /usr/local/hadoop/bin. The data for this work has been placed on HDFS.

- Check that you can access this computer with ssh and see if you have a home dir.
- Check that you have access to the http interfaces of the hadoop cluster. Open the following URLs in the browser:

  `http://152.77.78.100:50070/`

  `http://152.77.78.100:8088/`
- Do a test:

  ```
  bin/hadoop jar share/hadoop/mapreduce/hadoop-mapreduce-examples-2.2.0.jar pi 10 100
  ```
- If everything works, you can start working. Otherwise, ask for help.

3 WordCount

3.1 Getting started

- Using the project provided, prepare the WordCount example for the grid.
- Compilez ce fichier et faites-en un `jar` comme vu en cours.
- Create a simple document and copy it to HDFS.
- Run your program on MapReduce and check the output is correct.
- Start again, but this time give as input the directory containing *Misérables* by Victor Hugo.

3.2 Interpreting counters

- **Q 1:** Take a look at the counters for your job in the JobTracker interface:
  1. What does *Reduce input groups* mean? How could you change its value?
  2. What does *Map input records* mean? And *Map output records*?
  3. In your opinion, what is the link between *Map output records* and *Reduce input records*?
- **Q 2:** Add a *combiner* as seen in class. Execute the updated code on the 5 tomes of *les Misérables*.
  1. Which counters allow you to see that the combiner worked properly?
  2. Which counter is, in your opinion, the most important to evaluate the gain provided by the combiner?

4 Query suggestion

Many search engines on Internet suggest associated words when users do a web search. Our goal for this exercise is to compute the frequency of words associations to build such a service. To this end, we will analyze an AOL search engine query log.

Q 3: Create a class implementing the WritableComparable interface to represent a pair of words.

Q 4: Test it by modifying WordCount in order to compute the occurrences of each pair of words used in a query. Be careful about the AOL data format, the split function of the String class may help you. Test your code with 2 reducers (cf Hadoop job javadoc).

Q 5: We are only interested in pairs of words that appear more than 3 times. How many are there? Which counters gives this information?

Q 6: We would like to know how many queries of the search engine contain more than 3 words. To do it in a non intrusive way, it is possible to add custom counters, which will be handled by Hadoop as its own counters and appear in the results of the job. Create a LONG_QUERY counter in the mapper and increment it when the size of a query is at least 4. How many long queries are there? Do not hesitate to look online for documentation about counters. A counter can easily be added by declaring an `enum` in java, and is incremented with the context object of the mapper.

Q 7: We now want to know, for each word, the 5 words which are the most frequently associated with it in queries. Modify your code to perform this computation. In the reducer, find a way to avoid loading all words in memory before sorting them. When several Map-Reduce jobs are chained, it is often more convenient to keep a binary format instead of parsing text. You will probably need to use SequenceFileInputFormat and SequenceFileOutputFormat.