SJSU CMPE 282 HW MapReduce SPRING 2016

Theme: Java-based MapReduce

Description

You are given three weblog files from Canvas, weblog-1995-7-1.txt, weblog-1995-7-2.txt, and weblog-1995-7-3.txt, extracted from http://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html. Please write Java-based code on the MapReduce framework to aggregate the hit count of each URI in those three weblog files and sort hit count in ascending order.

Example:

If the aggregated hit count is

uri1	3
uri2	1
uri3	2
uri4	1
uri5	3

then the final sorted output is

uri2	1
uri4	1
uri3	2
uri1	3
uri5	3

To keep it simple, you are *not* allowed to specify any partitioning function in MapReduce.

Each line in the weblog is of a fixed format. One easy way is to utilize regular expression to parse and match each component in each line, and grab the URI field.

Environment

You are highly encouraged to utilize CMPE vCenter Server to deploy VM from either one of two templates (based on Cloudera QuickStart VM and HortonWorks). Each template (and therefore the deployed VM) contains Hadoop 2.x in pseudo-distributed mode. Alternatively, if your laptop has plenty memory (at least 8+ GB), you could run the VM on top of either VMware Player or Oracle VirtualBox (both are free) on your laptop. The VM can be downloaded from

http://www.cloudera.com/content/www/en-us/downloads/quickstart_vms/5-5.html
http://hortonworks.com/hdp/downloads/

Please follow "CMPE vCenter Server Lab Rules" (see slide).

Hints

- Logically the work can be achieved by two MapReduce jobs. Job1 aggregates the hit count for each URI, and job2 sorts hit count in to ascending order.
- MapReduce sorts the map output's intermediate key-value pairs by their keys before invoking the reduce function. This is also why you are not allowed to use any partitioning function (or else you could use the built-in TotalOrderPartitioner function).

Questions

Q1. For *each* job:

- At high level, describe the logic of your MapReduce job
- Input directory on the VM =
- Output directory on the VM =
- # of map tasks =
- # of reduce tasks =

Your screenshots (specified later) must match these info.

Q2. Use the aforementioned example hit count and sorted output to illustrate step by step ***how*** your jobs perform the work correctly.

Q3. Enclose screenshots for *each* of the following steps for *each* job:

- Before job execution, show input directory on VM
  - hadoop fs -ls <inputDir>
- During execution, capture output from "hadoop jar … ". In particular, highlight the # of map tasks and # of reduce tasks.
- After job execution, show output directory on VM
  - hadoop fs -ls <outputDir>
- After execution, for *each* part-r-* output file, show first few lines with head and last few lines with tail:
  - hadoop fs -cat <outputFile> | head
  - hadoop fs -cat <outputFile> | tail

Q4. Comment on the performance and scalability of the 2nd MapReduce job. Discuss if there is any way to improve its performance and scalability.

Additional requirements

- Map/reduce/driver Java class must end with <YourName><L3SID>, e.g., public class WordCountDemo123 {…}
- Any .jar file must end with <YourName><L3SID>, e.g., wcDemo123.jar

Submission

Submit the followings as *separate* files to Canvas

- CMPE282_HW3_<YourName>_<L3SID>.zip: All .java source files. Do *not* include .jar, .class files.
- CMPE282_HW3_<YourName>_<L3SID> (.pdf, .doc, or .docx): the report consists of answers and screenshots to questions specified in **Question**.
  - You receive no credit if your report is not .pdf, .doc, or .docx.
  - If a screenshot is unreadable, it will be treated as if you did not turn in that screenshot.
  - If you do not follow requirements (including naming conventions), you will receive no credit.
  - Any additional unique design or features you are proud of.

The ISA and/or instructor leave feedback to your homework as comments and/or in Crocodoc of your submission. To access Crocodoc, click "view feedback" button. For details, see the following URL:
http://guides.instructure.com/m/4212/l/106690-how-do-i-use-the-submission-details-page-for-an-assignment

(Optional) Extra credit

In addition to the original homework, use Spark to solve the same hit count aggregation and sorting problem. Include the source code (any language is OK), detailed steps, and supporting screenshots.