

15-319 Introduction to Cloud Computing

Project 3 “To Hadoop Or Not To Hadoop?”

Assigned Date: February 22nd, 2010
Deadline: March 23rd, 2010 at 11:59 p.m.

Goals:

1. Design and Implement a basic document indexing algorithm in Hadoop.
2. Develop a basic search application in Hadoop.
3. Tweak a Hadoop application for performance

Background:

Hadoop excels in text processing. In this project you will be asked to build an index of words from the complete works of Shakespeare. You will design the application in the map-reduce paradigm, write code in Hadoop and run it on your cloud. Our input data set is the plain-text collection of the complete works of Shakespeare courtesy *James Matthew Farrow* (<http://www.cs.usyd.edu.au/~matty/Shakespeare/>)

Useful resources:

- Chapters 2 to 5 of the *Hadoop : The Definitive Guide* by Tom White
- Video lectures from Cloudera
(<http://www.cloudera.com/resources/?type=Training>)

Part I: Preparing a word list

1. The complete works of Shakespeare is available at [/afs/qatar.cmu.edu/course/15/319/data/shakespeare.tar.gz](http://afs/qatar.cmu.edu/course/15/319/data/shakespeare.tar.gz)
2. Unpack the archive. Locate the file **glossary**
3. Write a Hadoop program to read this file as input and create a new file **list** containing only the words and not their definition. One way to do this would be to extract the uppercase words in the file. Your program should take the filenames **glossary** and **list** as command line arguments.

Example:

INPUT	OUTPUT
ABATE to shorten	ABATE
To cast down	ABATEMENT
To blunt	ABHOR
ABATEMENT diminution	
ABHOR protest;	
disgust	

4. Your program will also have to ignore duplicate words and words that are less than 2 characters.
5. **Deliverable:** Your Hadoop Project in a folder called “**WordList**”, and the word list you’ve generated as **list.txt**. Include the runtime of your program as a comment in your main Hadoop map reduce program..

Part II: Building a Word Index of Shakespeare’s Work

1. Write a Hadoop program to index all the words in **list.txt** (generated in the previous part) with all of Shakespeare’s works (each file inside the directories in the archive **shakespeare.tar.gz**). For each word, print the filenames in which the word appears in
2. Sample Output:


```
abate tamingoftheshrew, cymbeline, midsummernightsdream ...  
abatement cymbeline, twelfthnight ...  
abhor loveslabourslost, comedyoferrors, muchadoaboutnothing ...  
abide measureforemeasure, merchantofvenice, midsummernightsdream
```
3. **Deliverables:** Your Hadoop project in a folder called “**Indexer**”, and the generated index file as “**index.txt**”. Include the runtime of your Hadoop program as a comment in your main Hadoop map reduce program.
4. **Bonus:** The following extras are highly recommended and will contribute heavily to your overall project grade:
 - a. Count the number of times each word appears in each file
 - b. Optimization Techniques – Use various optimizations on your program code and/or Hadoop job configuration (eg: varying no. of mappers / reducers). Describe the various optimizations that you have attempted in detail and the effect on the program runtime in your project write-up. Include bar charts that show the difference in the program runtime across various versions.

Part III: Creating your own Shakespeare Google!

1. Write a Hadoop program to search for a particular word in the index and return the filenames that contain that word.

Example:

INPUT	OUTPUT
Abate	tamingoftheshrew, cymbeline, midsummernightsdream

2. **Deliverables:** Your Hadoop project in a folder called “**Search**”, and the output of a few sample queries as “**searchoutput.txt**”. Include the runtime of your Hadoop program to process a query as a comment in your main Hadoop map reduce program
3. **Mega Bonus:** Create a web application on your master node that accepts user queries from a browser and returns the filenames that contain the word. Your web application should search through the word index (generated in part II) from the HDFS location of your cloud.

For example if your master node is 10.2.160.23, we should be able to access a web interface on your master node using firefox via the socks proxy (<http://10.2.160.23>) and type a query to get results from within the browser itself.

Submission:

Deliverable: Hand in a project write-up discussing the problem and sketch of the techniques you used to solve it in Hadoop. Make sure to include a results section including a bar graph of all the runtimes of wordlist generator and index generator and search Hadoop programs. Use the 2-column ACM format for this paper (abstract, problem definition, methods used, results and comparison, conclusions and references). Include a full set of references, including books, tutorials and blogs that you’ve used to complete this project.

Add the write-up and all the deliverable folders from each part into a single zip file (project3.zip) and place it in:

`/afs/qatar.cmu.edu/course/15/319/handins/username/`

This file is to be submitted once and the final timestamp on the server will determine your submission time.

Grading:

As mentioned in the syllabus, this project is worth 15% of your final grade. You have over a month to finish the project. The following scheme will apply:

Projects	
Deliverables	75
+	
Student Written Code	25
<i>consists of:</i>	
Property	Percentage
Technique/Algorithm	70
Performance	15
Documentation/Cleanliness	15