PROJECT 2: DATA-STRUCTURES AND ALGORITHMS ENABLING DATA-INTENSIVE COMPUTING

Purpose:

1. To understand the data-structures and algorithms for **data-intensive computing**
2. To design and implement the solutions for several representative data-intensive problems using MapReduce and Hadoop DFS
3. Understand **virtualization** and deploy a single data-node **HDFS** using **VMware player** or any other virtual machine technology such as Oracle's VirtualBox or Microsoft's Virtual PC (**for prototyping and demos**).
4. Study and compare the scalability of MapReduced solution for various cluster sizes (2, 4, 8, 16, 32) and core sizes (4, 8, 12, 16).
5. Study the effect of Combiner and Partitioner. Repeat the above experiments with (i) combiners (ii) partitioners and (iii) combiners and partitioner for one of the problems.
6. **(optional)**To explore designing and implementing data-intensive computing solutions on a **cloud environment**: in this case on **Amazon Compute Cloud** (EC2)[1] and Google App Engine (GAE)[2]).

Problem Statement:

Project 1 focused on content accumulation methods and the three tier web application (or regular application) infrastructure to enable data-intensive applications. This project will focus on parallel processing including by *mapreducing* ultra-scale data. MapReduce is a framework for processing large scale data sets. It exploits the write-once-read-many (WORM) characteristic typical of unstructured data to represent and process (i) as key-value pairs <key,value> and (ii) using a suite of operations such as map, partition, reduce, shuffle etc. A special data repository (like a file system used in a traditional operating system) is needed to efficiently and reliably store and deliver the large data set. Google file system (GFS)[4] is such a system, and Hadoop Distributed File System (HDFS)[5] is another GFS-like system available under Apache open source license.

Preparation before lab:

1. Read the chapters 1-5 in the Lin and Dryer text [6].
2. Review the foundations of MapReduce and Hadoop Distributed File System [3], [4], [5].
3. You will be working on CCR (Center for Computation research) cluster. Make sure you are able to access the facility (username, password); run the sample wordcount and address any problems that may arise.

Assignment:

Problem 1 (40 points) (deadline: 3/23/2013)

1. You may use any existing solutions. For data use any of the books from Project Gutenberg free ebook source (http://www.gutenberg.org/).
2. Design the solution for the word co-occurrence matrix problem described in Chapter 3 of Lin and Dryer's text [6]. Specifically study the problem described in section 3.2. **Implement the "naïve", "pairs" approach as well as the "stripes" approach for any**

given corpora of data from Project Gutenberg. Choose the data appropriately: for example, Shakespeare's works may **not** be a good selection since it contains very old usage of words and may not yield good co-occurrence.

2.1. The context for co-occurrence can be a sentence, a paragraph, or a chapter. In the case of corpora with large number of document, even the document can be the context. We will choose "paragraph" as ***co-occurrence context***.
2.2. In order to configure paragraph as the context you will have customize or modify the "split" or the "record reader" of the mapreduce framework.
2.3. Instead of counts of co-occurrence, you will store the ***relative frequencies*** of the co-occurrences as discussed in equation 3.1
2.4. Compare the run-time for various sizes of data for the pairs and stripes implementations.
2.5. Choose a large corpus, and evaluate the scalability of your algorithm for (i) different cluster sizes (2, 4, 8, 16, 32) for the same number of cores (8 cores?), (ii) for different cores (4, 8, 12, 16). Don't go more than 32 nodes cluster.

Problem 2 (30 points) (deadline: 4/6/2013)

3. **Graph Algorithms**: Now you have a good understanding of the basics. In this step you will move onto a more useful and practical problem. In this case, the problem you will solve is the shortest path algorithm by Dijkstra. A MapReduce (MR) solution for this is available in a presentation by J. Lin [6]. Design the map and reduce functions for this problem and implement it, and test it using the sample graph given in [6]. Then use a larger graph that will be given to you by the TAs and test your solution.

Problem 3 (30 points) (4/27/2013)

4. **Clustering: K-means clustering using numerical data:** Most critical aspect of this project is converting K-means clustering into a MR parallel algorithm. In particular, what will be Map, what will be the Reduce function for K-means and what are input and output for the respective functions, and how to implement K-means using these functions? We will discuss this during lecture. See [7,8] that are intermittently available.

System Architecture

The system architecture for the assignment given is shown in Figure 1. In Figure 1, the development environment is shown on the left, that is a single-node (or multi-node) HDFS deployed in your local environment. For the second part of the assignment you will work with the same problem as above but in the ***production environment*** shown on the right. You will transfer the data for the two applications into 2 buckets of the amazon simple storage service (s3) and upload the map and reduce functions designed in the part1. You will then create a MR workflow to process the data. (Working with amazon and Google App Engine is optional)

Working with CCR means you are sharing the resources with the user community of CCR. The cluster and size you need may not be available at your disposal. For example on March 3rd CCR cluster is going through downtime for maintenance. Nothing will be available. Also as go higher in cluster size you will be waiting in the "batch" queue for a longer time. Be aware of this. Don't try to do all in the last week of due date. To address this "procrastination" problem I have set intermediate deadlines. These are indicated with the problems.

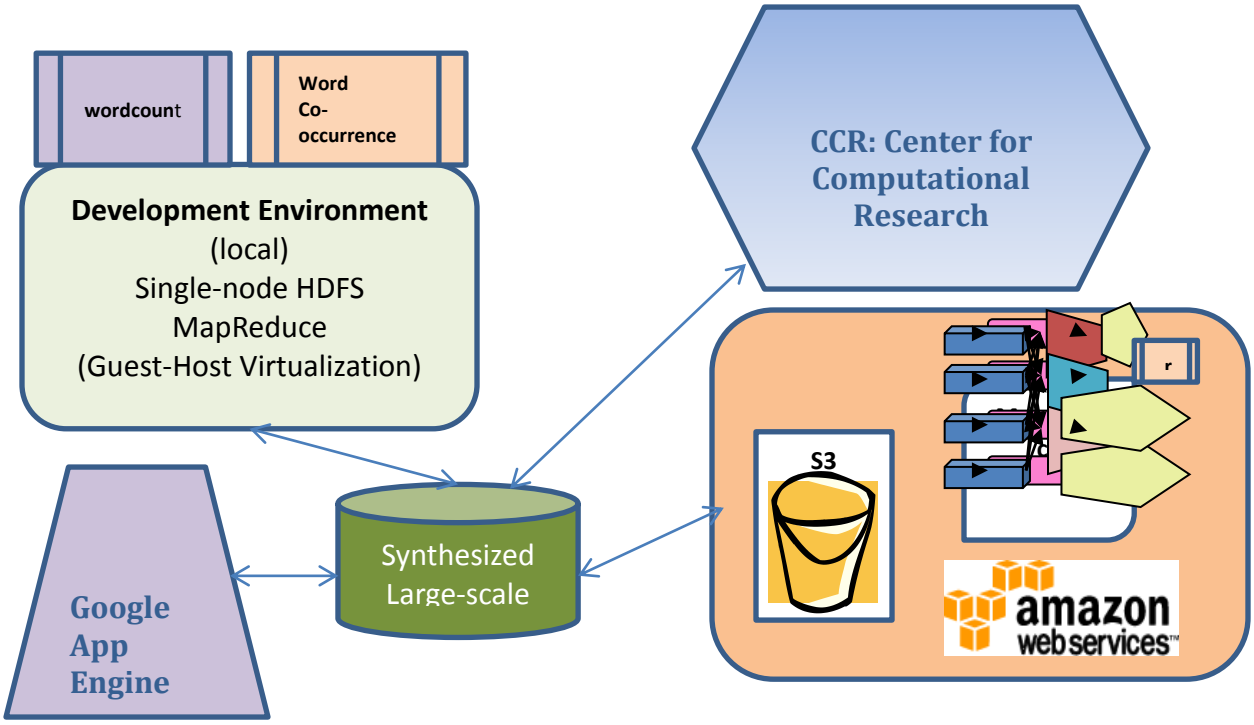

Figure 1: System Architecture for Data-intensive HDFS-MapReduce Application Development

Project Deliverables:

1. A tar file of the single node implementation of the MR-wordcount including the data; a tar file of the single node implementation of the co-occurrence for pairs approach and another for the stripes approach. (in all, three jar/tar files). Similarly jar files for graph and K-means algorithms.
2. An experience report providing all the details of the project design, implementation and the performance evaluation report. This should have the user's manual, programmer's manual and any design diagrams. **Due date 4/29/2013**
3. You may have to demo the project to the TA.

Submission Details: You will have to submit from CSE systems. So file transfer code to cse system and then submit it.
submit_cse487 files separated by space
submit_cse587 files separated by space

References:

1. Amazon Elastic Compute Cloud. http://aws.amazon.com/ec2/, last viewed October 31, 2011.
2. Google App Engine (GAE). http://code.google.com/appengine/, last viewed October 31, 2011.

3. MapReduce on Amazon. Advanced MapReduce Features. http://developer.yahoo.com/hadoop/tutorial/module5.html, last viewed October 31,2011.
4. Dean, J. and Ghemawat, S. 2008. **MapReduce: simplified data processing on large clusters.** *Communication of ACM* 51, 1 (Jan. 2008), 107-113.
5. Hadoop Distributed File System (HDFS). Apache Hadoop: http://hadoop.apache.org, last viewed September, 2010.
6. Lin, J and Dryer, C. Data-Intensive Text Processing with MapReduce 2010, Vol. 3, No. 1, Pages 1-177, (doi:10.2200/S00274ED1V01Y201006HLT007). **An online version of this text** is also available through UB Libraries since UB subscribes to Morgan and Claypool Publishers.
7. K-means Cluster with MapReduce, http://codingwiththomas.blogspot.com/2011/05/k-means-clustering-with-mapreduce.html
8. Benchmarks: http://wiki.apache.org/hama/Benchmarks#K-Means_Clustering