

Big Data Analytics Lab

List of Experiments

Experiment 1:

Implement the following Data structures in Java

- a) Linked Lists
- b) Stacks
- c) Queues
- d) Set
- e) Map

Experiment 2:

- (i) Perform setting up and Installing Hadoop in its three operating modes: Standalone, Pseudo distributed, Fully distributed
- (ii) Use web based tools to monitor your Hadoop setup.

Experiment 3:

Implement the following file management tasks in Hadoop:

Adding files and directories

Retrieving files

Deleting files

Experiment 4:

Run a basic Word Count MapReduce program to understand MapReduce Paradigm.

Experiment 5:

Write a map reduce program that mines weather data.

Weather sensors collecting data every hour at many locations across the globe gather a large volume of log data, which is a good candidate for analysis with Map Reduce, since it is semi structured and record-oriented.

Experiment 6:

Use MapReduce to find the shortest path between two people in a social graph.

Hint: Use an adjacency list to model a graph, and for each node store the distance from the original node, as well as a back pointer to the original node. Use the mappers to propagate the distance to the original node, and the reducer to restore the state of the graph. Iterate until the target node has been reached.

Experiment 7:

Implement Friends-of-friends algorithm in MapReduce.

Hint: Two MapReduce jobs are required to calculate the FoFs for each user in a social network. The first job calculates the common friends for each user, and the second job sorts the common friends by the number of connections to your friends.

Experiment 8:

Implement an iterative PageRank graph algorithm in MapReduce.

Hint: PageRank can be implemented by iterating a MapReduce job until the graph has converged. The mappers are responsible for propagating node PageRank values to their adjacent nodes, and the reducers are responsible for calculating new PageRank values for each node, and for re-creating the original graph with the updated PageRank values.

Experiment 9:

Perform an efficient semi-join in MapReduce.

Hint: Perform a semi-join by having the mappers load a Bloom filter from the Distributed Cache, and then filter results from the actual MapReduce data source by performing membership queries against the Bloom filter to determine which data source records should be emitted to the reducers.

Experiment 10:

10. Install and Run Pig then write Pig Latin scripts to sort, group, join, project, and filter your data.

Experiment 11:

11. Install and Run Hive then use Hive to create, alter, and drop databases, tables, views, functions, and indexes