YAH: YET ANOTHER HADOOP

Team Members:
Ajay P: PES2UG19CS020
Aparna Kalla: PES2UG19CS059
R Shailesh: PES2UG19CS307
Ramit Bathula: PES2UG19CS319

Design Details: We have implemented Hadoop and the mapper and reducer functions.

Surface level implementation details about each unit:

Datanodes:

Each datanode is implemented using directories and the file inputted by the user is split into blocks and stored into each of these blocks based on the hashing function.

Namenodes:

Namenodes is implemented using a namenode.py file which is always running in the background. If any of the blocks in the datanodes are deleted due to some error then the namenode makes sure to replicate the deleted namenodes.

Datanode log:

The datanode log will just have information of when the datanode is updated. Each datanode has its own datanode log file

Namenode log:

The namenode log file stores information about each datanode in a dictionary format. The dictionary contains the datanode number as the key and the values are the number of blocks in each datanode along with block names.

Filesystem:

The filesystem is implemented using a directory as well and every new folder and file created is stored in the filesystem directory.

Secondary Namenode:

This is implemented using a python file. It is a backup to our primary Namenode and if the primary Namenode ever fails, the Secondary Namenode takes its place.

Reason behind design decisions:

We chose this particular design because a distributed file system is a very efficient way to manage and store large files and is an easy way to run different operations on these files using the concept of Hadoop's mapper and reducer.

Take Away from the project: We learnt a lot of interesting new topics based on the implementation of hadoop. Uptill now we weren't bothered about how hadoop distributed our files, but with this project we learnt how to distribute files to each datanode, how to run the namenode and secondary namenode in the background and checking for any failures in our datanodes.