**Project 3**

**Report**

**Team Members:**

**Arpitha N Kashyap (arpkashy) and Saloni Sharma(sharma36)**

1. What is Hadoop Distributed Cache and how is it used in this program?

Hadoop distributed Cache is a facility provided by the Map-Reduce framework to cache files such as text files, archives, jars etc. Several applications need these files. Before any tasks for the job are executed on that node, the framework copies the necessary files onto the slave node. Its efficiency is based on the fact that the files are copied only once per job and the ability to cache archives which are un-archived on the slaves. We generate the two absolute location file paths by adding the database and program archive to the distributed cache. Then we copy the database and the binary program executable from the HDFS to the local file system. This is done along with the required dependencies to execute each and every map task.

1. Write the two lines that put and get values from Distributed cache. Also include the method and class information.

To put values in the distributed cache, we have

In DataAnalysis.java file,

DistributedCache.addCacheArchive(new URI(programDir), jc); /\*Adding the archive that needs to be localized to the conf\*/

To get values from the Distributed Cache, we have

In RunnerMap.java file,

Path[] local = DistributedCache.getLocalCacheArchives(conf);

//Getting the path array of the localized caches that contain the Blast binary and the db

1. In previous projects we used Hadoop’s TextInputFormat to feed in the file splits line by line to map tasks. In this program, however, we want to feed in a whole file to a single map task. What is the technique used to achieve this? Also, briefly explain, what are the key and value pairs you receive as input to a map task and what methods are responsible for producing these pairs?

We use a customized Hadoop MapReduce InputFormat called DataFileInputFormat.java. This generates key-value pairs: <filename, filepath on HDFS>. It uses the custom FileRecordReader which has the getCurrentKey() and getCurrentValue() method. These methods are used to produce these key-value pairs.

1. Do you think this particular implementation will work if the input files are larger than the default HDFS block size? Briefly explain why. [Hint: you can test what will happen by concatenating the same input file multiple times to create a larger input file in the resources/blast\_input folder]

Here we increased the input file size to greater than 64mb. This is the default HDFS block size got by concatenating the same input file multiple times. Then we execute it using the input file which was modified. We have to modify the timeout property such that it is greater than 600 seconds (the default) which is specified in the mapred-site.xml.

<property>

<name>mapred.task.timeout</name>

<value>1800000</value> <!-- 30 minutes -->

</property>

We will get the output result for the modified input file if we increase the timeout until we get the buffer where the file is entirely executed.

1. If you wanted to extend this program such that all output files will be concatenated into a single file, what key and value pairs would you need to emit from the map task? Also, how would you use these in the reduce that you would need to add?

We can write the output as follows:

-> Using a fixed text value as the key and

-> outFile as the value.

We send the output to the reducer. The reducer will have all the outFiles as the value. This will then be concatenated and written to the output file.