HADOOP BLAST

Project Report – Hadoop BLAST

Indiana University

Revision	Date	Description	Author
1.0	24-Feb-2017	Questions 1,2,3	Melita Dsouza
1.1	25-Feb-2017	Questions 4,5	Karthik Vegi

HADOOP BLAST

PROJECT DESCRIPTION

In this project, we implement BLAST (Basic Local Alignment Search Tool) that regions of local similarity between sequences. The program compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches. It can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families.

Questions

1. What is Hadoop Distributed Cache?

Hadoop's distributed cache allows distribution of datasets instead of serializing the data in the job configuration. It helps to cache files such as text, archives, jars etc. to the task nodes in time for the tasks to use them when they run thereby saving valuable network bandwidth.

How is it used in this program?

The input format of the Hadoop-Blast program is a set of FASTA files, a text-based format for representing either nucleotide sequences or peptide sequences, in which nucleotides or amino acids are represented using single-letter codes. The two java classes, *DataFileInputFormat* and *FileRecordReader* are used to collect the input files from HDFS and create key-value pairs for the Mapper. In this program, we have 4 input files which will be distributed to the nodes using the Hadoop's Distributed Cache feature.

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

PUT to Distributed Cache

Class: *DataAnalysis*

Method: *void launch (int numReduceTasks, String programDir, String execName, String workingDir, String databaseArchive, String databaseName, String dataDir, String outputDir, String cmdArgs)*

Line: *DistributedCache.addCacheArchive(new URI(programDir), jc);*

GET from Distributed Cache

Class: *RunnerMap*

Method: *void setup (Context context);*

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

3. 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?

HADOOP BLAST

To feed in a whole file to a single map task, we write our own custom input format and set ***isSplittable*** to false. This can be achieved by using ***DataFileInputFormat*** which invokes the ***FileRecordReader*** class, which extends the ***RecordReader*** class.

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?

The key is the *filename*, and the value includes the full HDFS path for each uploaded input files. The methods responsible for this are *getCurrentKey()* and *getCurrentValue()* methods in the ***FileRecordReader*** class.

4. Do you think this implementation will work if the input files are larger than the default HDFS block size? Briefly explain why?

The default block size of HDFS is 64MB. If the input files are larger than the default block size, HDFS splits them into 64MB blocks by default.

We tested this case by making two inputs files of 75MB and the other two input files to 64kb. This resulted in 6 input paths to be processed which confirms to the above explanation. However, we saw the log and confirmed that this approach doesn't work as there were multiple time out errors. The job was restarted multiple times but always failed to complete.

5. If you wanted to extend this program such that all output files will be concatenated into a single, 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?

The main function of the Map class is to create a java process to call the Blast program. Each map task downloads the assigned input file from HDFS, and passes this input to run the Blast program.

The below line creates an external process for the Blast executable:

```
Process p = Runtime.getRuntime().exec(execCommand)
```

The following line copies the (Blast) output back to HDFS:

```
fs.copyFromLocalFile(new Path(outFile),outputFileName)
```

Instead of sending the output files to the **OutputHandler**, we can send all the output files and the output path as **<key,value>** pairs to the reducer class. To generate a single output on HDFS, we need to pass it through a single reducer. This can be done using the below code:

```
Job.setNumberOfReducer(1)
```

An alternative way to concatenate files is to use Hadoop -getmerge command which can be used in the below way:

```
hadoop fs -getmerge /output/dir/ /desired/output/outfile.txt
```

REFERENCES

- **Hadoop Docs**: https://hadoop.apache.org/docs/stable/hadoop-mapreduce-client/hadoop-mapreduce-client-core/MapReduceTutorial.html
- **Hadoop API**: https://hadoop.apache.org/docs/r1.2.1/api/org/apache/hadoop/mapred/Mapper.html
- **Blast:** https://blast.ncbi.nlm.nih.gov/Blast.cgi
- Hadoop Definitive Guide, Tom White