**PROJECT 3: Hadoop Blast March 1, 2015**

**INTRODUCTION:**

We have implementing a parallel version of BLAST (Basic Local Alignment Search Tool) using the programming interfaces of the Hadoop and MapReduce framework.

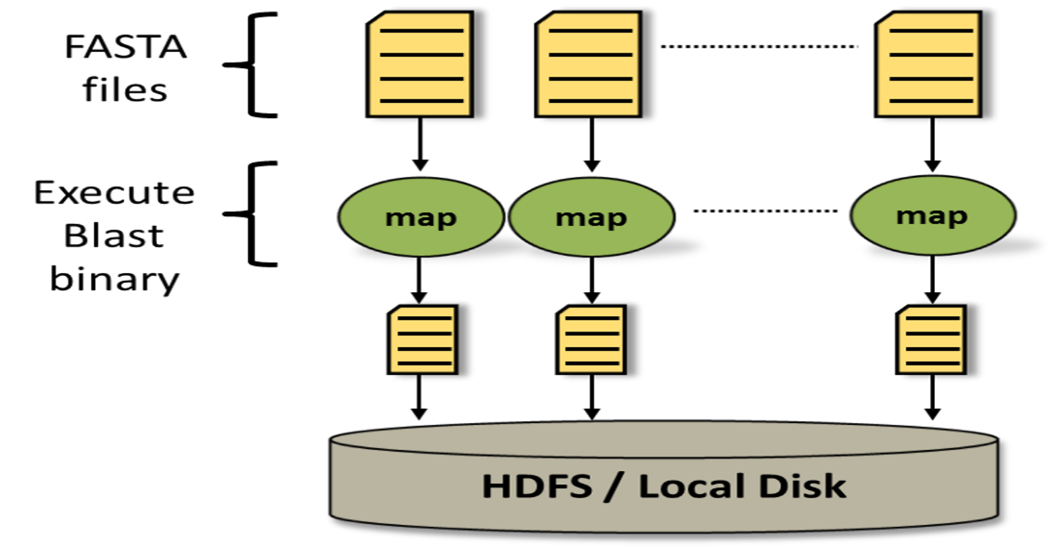
**IMPLEMENTATION:**

Let us understand the different parts of the project:

1. **BLAST:**

Basic Logical Alignment Search Tool (BLAST) finds regions of local similarities between biological sequences [1].

1. **Project Hadoop Blast**: An advanced Hadoop program which allows BLAST, a bioinformatics application, to utilize the parallel programming capability of Hadoop and MapReduce framework.
2. **Process:**
   1. *Input*: FASTA files stored in the local file system, BLAST Binary executable file, optimized database as distributed cache. All the files are initially stored on the local file system.
   2. *RunnerMap.java*: The Map class which takes the BLAST (v2.2.23) binary program and optimized database as distributed cache, then executes BLAST binary as java external process with the assigned FASTA file to local system. These are passed as key-value pairs of <filename, filepath on HDFS> handled by a provided customized Hadoop MapReduce InputFormat DataFileInputFormat.java
   3. *DataAnalysis.java*: The main program which sets the Hadoop MapReduce job configuration.
   4. As shown in figure 1, the input data will be uploaded to the HDFS and distributed across the various nodes. Hadoop framework reads the application records from HDFS with the InputFormat interface and generates <key, value> pair input streams as the intermediate output.
   5. A customized Hadoop MapReduce InputFormat DataFileInputFormat.java is used to generate the key-value pairs of <filename, filepath on HDFS>.



*Figure 1: Hadoop Blast Dataflow*

* 1. The map function of the Hadoop BLAST program initially sets up the distributed cache and generates the two absolute location filepaths for BLAST binary and BLAST Database.
  2. Next, the program copies the assigned FASTA file to a local disk by looking up the file from HDFS and generating an absolute filepath.
  3. Once the above process is successfully executed and the file dependencies are stored in the local disk, an external java process executes the BLAST binary with the correct parameters.
  4. *Output:* At the end, the output FASTA file of the BLAST binary will be uploaded back to HDFS.
  5. We do not require a reducer since the mapper simply compares the FASTA files with the BLAST database and returns the protein sequences that are similar.

**FILES:***RunnerMap.java:* Mapper file

*pracshah\_abhivija\_ppkapoor\_HadoopBlast\_output.fa.zip: The .zip file contains four .fa output files*

*pracshah\_abhivija\_ppkapoor\_project3\_report.docx: Serves as the* README and Report file for the project.

**TOOLS:**Ubuntu OS installed on a Virtual Box with pre-installed Hadoop setup, vim editor.

**COMPILE AND RUN:………………………………………………………………………………………………**$ cd /root/MoocHomeworks/HadoopBlast/

# usage: ./compileAndExecHadoopBlast.sh

$ ./ compileAndExecHadoopBlast.sh

The one-click script file *compileAndExecHadoopPageRank.sh* compiles the two .java files and outputs compilation errors, if any.

**OUTPUT:**The result is generated as /root/MoocHomeworks/HadoopBlast/output/HadoopBlastOutput.fa.

$ cd /root/MoocHomeworks/HadoopPageRank/

$ cat output/\*

**REFERENCES:**[1] BLAST Assembled Genomes. http://blast.ncbi.nlm.nih.gov/Blast.cgi   
[2] ARCHIVED: Hadoop Blast. https://kb.iu.edu/d/bcsp