Privacy-Preserving Mapping

1. Define Problem

Mapping human genomic data to reference genome in a privacy-preserving manner, due the data contains sensitive information, with the help of Cloud.

1.1 Background

This project is to design and implement a novel method to map large human genomic data, about **1T bytes**, to reference genome in a privacy-preserving manner, due the data contains sensitive information, such as phenotype and disease potential, of data donor. The usual method of aligning two strings in bioinformatics can be Smith-Waterman method, which runs in O(mn) and requires O(mn) space, where m and n are lengths of these two separate strings. However, it won't work now in our case, because we have up to 3 billion chars in a string, not to mention that each DNA is composed of two strings, with their sequence order reversed. So in this case, we want to enhance the performance and efficiency of aligning, first, we define a k-length subarray to be k-mer. Then we enumerate every possible k-mers and their positions in the genomic string, of course, we need to use Spark to store these data in different machines. Then we use the k-mer of a query, to find the locations, if any, in the genomic string. Basically this is equivalent to a huge hash table, whose keys are the k-mers, values the positions in the genomic string. Once we find the position of the k-mer in the genomic string, then we can expand to both ways: left and right to align the data.

There is no cloud-based read mapping tool on human genomic data. With the reduction of sequencing cost and precision medicine initiative, this kind of tool is urgent to utilize both huge amount of human genomic data and powerful clouds. Diyue and I are in the same research group, advised by Prof. Haixu Tang, and we will do the work together with our lab member, Yongan Zhao, who has substantial amount of research experience in reads-mapping and big data analysis. This is the first step to utilize the power of cloud computing in human genomic data processing. And then we want to explore more opportunities in privacy-preserving cloud computing on human genomic data.

1.2 Goals and Objectives

Mapping human genomic data to reference genome in a privacy-preserving manner, due the data contains sensitive information, with the help of Cloud.

1.3 Tasks to Complete

1. Set up all needed frameworks and software in India, aka set up VMs in India, then set up Hadoop clusters on our own. In our case, we set up a single node Hadoop cluster with the name **hadoop-master**.
2. Applying Hadoop to map query against reference.

1.4 Disclaimer

We were revoked access to Juliet last year, and we communicated with AIs. We successfully booted several VMs on India.futuresystems.org and set up Hadoop cluster in the end. You can find details in our Manual below.

2. Define Software

2.1 Techniques

Since our project is different from what were provided by BDAA group, we have to do the configuration of related big data frameworks ourselves. There are hundreds of platforms there, and we did our research about which ones to go with our project, and we made our choices. We set up VMs in india.futuresystem.org and set up a single node Hadoop cluster on **hadoop-master** VM.

2.1.1 Hadoop

Hadoop is a framework that allows for the distributed processing of large data sets across clusters of computers, just like Juliet, using simple programming models. It can tolerate errors and can work even if a node went down. We need to install the Hadoop Distributed File System, also known as HDFS, to provide high-throughput access to application data. In our case, we need HDFS to access our 1T bytes data efficiently. We also need Hadoop Yarn, a framework for job scheduling and cluster resource management. MapReduce, a YARN-based system for parallel processing of large data sets is also required.

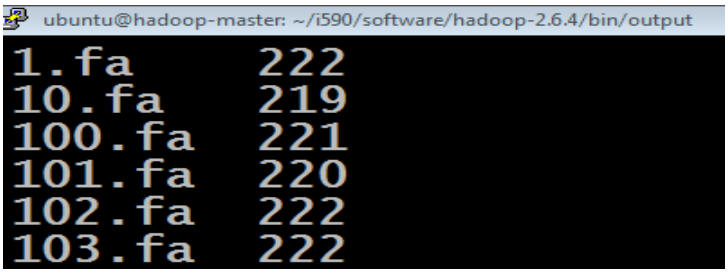
2.1.2 Spark – To do

As is advertised, Spark is a fast and general engine for large-scale data processing. Spark can access HDFS, HBase etc. and run programs up to 100x faster than Hadoop MapReduce in memory, or 10x faster on disk. We can use Spark as a big table to do the O(1) time look-up operation.

3. Produce Results

3.1 Results

Our program will output the locations where the query occurs in which genome part ID. Below is a screenshot of our result. A little bit of elaboration, the first row below interprets as we found the query in the file 1.fa, and the index is 222, of course staring from index 0.



In the above picture, each row consists of two columns, of which the first is the name of file (file containing part of the gene sequence), and the second is the index of substring of query in this file, starting from 0 index. If there are multiple substrings which is very rare, we just the first occurrence. Thus we have completed our project.

3.2 Prove Installation/Configuration/Execution

We have took many screen shots of the whole process as proof, for the sake of simplicity, please regards to the Manual part. We guarantee that this project can definitely be reproduced, given that you follow our Manual part.

3.3 Future Improvement

In the future, we hope to deep the understanding of Hadoop, Spark, HBase, and related Big Data Techniques. We also hope that we can get privilege of using Juliet again to analyze our data with HBase, then compare the results and performance of using Spark and HBase separately.

4 Manual

4.1 Documentation

Because we use the standard frameworks and software, both have the official installation.

4.2 Results Reproducibility

Our results can be reproduced using the same environment, same programs, and same command.

4.3 Details of Development Environments

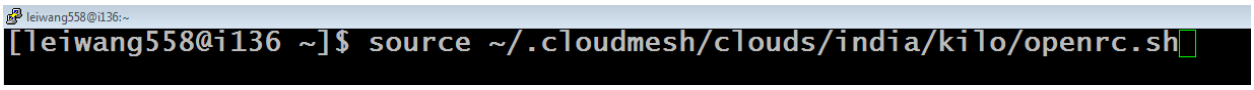
We installed JAVA 1.7+ and hadoop 2.6.4 under ~/i590/software/ on ubuntu@10.0.7.165 on india.futuresystems.org. Note that we set up a single node Hadoop cluster for simplicity.

4.4 Instructions to use our software

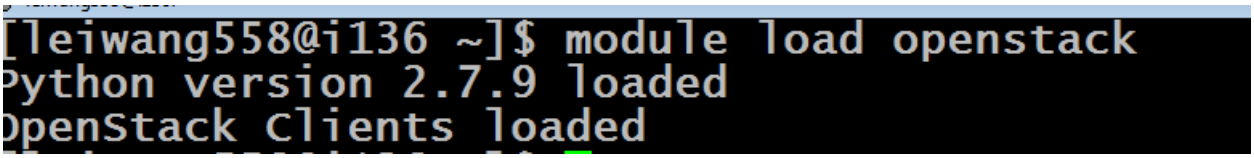
We have set up VMs in india.futuresystems.org server and set up a single node Hadoop cluster on VM **hadoop-master**, and we have deployed our programs into it.

Log into india.futuresystems.org

Load the .bashrc file



Load openstack



Check our VMs using nova list

```
[leiwang558@i136 ~]$ nova list
+--------------------------------------+---------------------+--------+---------
---+-------------+----------------------+
| ID                                   | Name                | Status | Task Sta
te | Power State | Networks             |
+--------------------------------------+---------------------+--------+---------
---+-------------+----------------------+
| 3434e6b8-c1ef-4ca8-9323-37d6ca609618 | hadoop-master       | ACTIVE | -
   | Running     | fg495-net=10.0.7.165 |
| 0d029cd9-1781-4226-a917-17f75d6d8ed1 | hadoop-master-trial | ACTIVE | -
   | Running     | fg495-net=10.0.7.168 |
| 7b438d75-d1ad-4896-a6ce-e01de8613a23 | hadoop-slave-1      | ACTIVE | -
   | Running     | fg495-net=10.0.7.166 |
| b74589f9-beaa-4dd4-82b0-c90af312bffd | hadoop-slave-2      | ACTIVE | -
   | Running     | fg495-net=10.0.7.167 |
| ae24ecf7-42ad-4c45-be7b-626e3e158cfc | hadoop-slave-3      | ACTIVE | -
   | Running     | fg495-net=10.0.7.169 |
```

Since we set up a single node Hadoop cluster on VM **hadoop-master**, then **ssh** to it

```
leiwang558@i136:~
[leiwang558@i136 ~]$ ssh ubuntu@10.0.7.165
```

Check project tree structure by

cd ~/i590/PrivacyPreservingMapping

tree .

```
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping$ tree .
.
├── report
│   └── I590-FinalProject.pdf
└── src
    ├── data
    │   ├── 500SeqDB_1.fa
    │   └── files
    ├── README
    ├── readsmapping
    │   ├── buildAndRun.sh
    │   ├── classes
    │   │   ├── ReadsMapping.class
    │   │   ├── ReadsMapping$Map.class
    │   │   └── ReadsMapping$Reduce.class
    │   ├── clean.sh
    │   ├── files
    │   ├── query.fa
    │   ├── ReadsMapping.class
    │   ├── readsmapping.jar
    │   ├── ReadsMapping.java
    │   ├── ReadsMapping$Map.class
    │   ├── ReadsMapping$Reduce.class
    │   └── rm.jar
    └── splitfiles
        ├── build.sh
        ├── classes
        │   └── SplitFiles.class
        ├── clean.sh
        ├── Screenshot-splitfiles.png
        ├── splitfiles.jar
        └── SplitFiles.java
```

Then type

cd src/data

ls

to check the **500SeqDB_1.fa, which is the preprocessed genome** where query is mapped against.

```
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/data$ ls
500SeqDB_1.fa
```

Then cd ../splitfiles folder, and simply run the following commands which are to split the genome file into single files containing one gene sequence each.

ls

./clean.sh

./build.sh

```
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/splitfiles$ ls
build.sh  classes  clean.sh  splitfiles.jar  SplitFiles.java
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/splitfiles$ ./clean.sh
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/splitfiles$ ./build.sh
added manifest
adding: SplitFiles.class(in = 1837) (out= 1054)(deflated 42%)
**************************
**The part 1 of PPM proj***
**Split fasta into subs****
**************************
../data/500SeqDB_1.fa
SplitFiles are done
Source code compiled!
cp files/ to ../data/
```

So now, we have split the genome into single gene seqs located at ../data/ folder. Note that, this step does not utilize Hadoop yet.

Then type

cd ../readsmapping

then simply run

./buildAndRun.sh

```
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/splitfiles$ cd ../readsmapping/
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/readsmapping$ ./buildAndRun.sh
################################################
#### Make sure you finish splitfiles/ first ####
####        Else Please Ctrl + C            ####
################################################
```

Type **Y** when you see this

```
16/10/22 20:51:18 INFO namenode.NNConf: Maximum size of an xattr: 16384
Re-format filesystem in Storage Directory /tmp/hadoop-ubuntu/dfs/name ? (Y or N)
```

This is a screenshot showing when it's done.

```
16/10/22 16:18:31 INFO mapreduce.Job:  map 99% reduce 33%
16/10/22 16:18:52 INFO mapreduce.Job:  map 100% reduce 33%
16/10/22 16:19:01 INFO mapreduce.Job:  map 100% reduce 100%
16/10/22 16:19:01 INFO mapreduce.Job: Job job_1477151555399_0001 completed successfully
16/10/22 16:19:01 INFO mapreduce.Job: Counters: 50
        File System Counters
                FILE: Number of bytes read=4594
                FILE: Number of bytes written=53443164
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1773864
                HDFS: Number of bytes written=3870
                HDFS: Number of read operations=1503
                HDFS: Number of large read operations=0
                HDFS: Number of write operations=2
        Job Counters
                Killed reduce tasks=1
                Launched map tasks=500
                Launched reduce tasks=2
                Data-local map tasks=500
                Total time spent by all maps in occupied slots (ms)=1262327
```

Then just wait for it to finish, and the programs are done. This simply integrated bash script handle all the rest work.

Since the results are stored in HDFS, and my bash script file **./buildAndRun.sh**, automatically retrieves the result from HDFS and stores it under folder $PPM/output

Then you can

cd $PPM/output

less part-r-00000

```
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/readsmapping$ cd $PPM/output
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/output$ less part-r-00000
```

4.5 Dataset

4.5.1 Testing Genome Dataset

Preprocessing is needed for our dataset, since human genome is up to 1T bytes. The huge genome should be split into parts where each part should be given an ID like “>1”, then followed directly with the piece of gene sequence. You can find such an example in our directory PrivacyPreserveMapping/src/data/500SeqDB_1.fa.

If you want to test our programs on other datasets, please make sure it satisfies our conditions above. Also, please put it under $PPM/src/data, and then cd $PPM/src/splitfiles and change the input file name from ../data/500SeqDB_1.fa to the name of your file, in the script ./build.sh

```
# Execute the program and copy generated split files to ../data/files
java -cp splitfiles.jar SplitFiles ../data/500SeqDB_1.fa
```

4.5.2 Query

The query is in the file $PPM/src/readsmapping/query.fa, simply put any query you want into the query.fa, then you can map it to the whole genome using our programs.

```
ubuntu@hadoop-master:~/i590/PrivacyPreservingMapping/src/readsmapping$ ls
buildAndRun.sh  classes  clean.sh  files  query.fa  readsmapping.jar  ReadsMapping.java
```