Final Project Report for CSE590 Spring 2012

Large Scale RNA Expression Analysis

Piyush Kansal

&

Himanshu Jindal

under the guidance of

Prof. Rezaul Chowdhury,

Stony Brook University

&

Prof. Michael Schatz,

Cold Spring Harbor Laboratory

ABSTRACT

The human genome is a long linear molecule composed of ~3 billion nucleotides. It contains the genetic instructions for how your body functions, as do the genomes of all living organisms. Each cell in your body contains an identical copy of the genome, although different sets of genes may be active (transcribed into RNA) depending on the cell type or environment the cell is living in. For example, healthy heart, lungs, brains, eyes, and skin cells have very different qualities, as do cancerous cells from healthy cells of the same cell type. As such an extremely important problem in computational biology is to map out which genes are active (expressed) in a given experiment, and measure their dynamics over time.

The basic technique for measuring gene activity is called *RNA-sequencing* in which we collect a sample of RNA from a cell, and sequence short fragments of the molecules that are present. At this point the biological problem fundamentally becomes a computer science & engineering problem: how can we align many billions of short sequences to the reference genome, intersect the coordinates of the alignments to the coordinates of the genes, and then statistically compare the expression level over time to discover differentially expressed and other active genes.

HOW TO SOLVE THE PROBLEM?

The goal of the project is to:

- Build a pipeline to compute the differential gene expression using Hadoop to address the issue of slow speeds when computing large size of data
- We are supposed to leverage the existing genomics tools as much as possible for the analysis:
  - **BWA** [1] is a widely used tool for mapping individual reads to a reference genome
  - **SAMTools** [2] is a widely used program for scanning the read alignments
  - Once the alignments and coverage measurements are complete, use the statistical package **R** [3] for computing metrics on the dynamics of the genes over multiple conditions

WHY HADOOP?

As suggested by Prof. Michael Schatz from Cold Spring Harbor Laboratory, the genome sequencing data processing requirement for current year, 2012, is 15 petabytes and it is increasing at the rate of 5 times/year, so by next year, 2013, the data requirements will grow up to 60 petabytes; in 2014, it will be 300 petabytes. Looking at this scale, we obviously need a system, which can easily adapt itself to the ever-increasing data processing demand. And since Hadoop has been designed and developed to address such problems, it becomes our obvious choice.

PREVIOUS EFFORTS

A similar version of this pipeline was published in 2010 in a tool called Myrna [4]. However, Myrna uses an old program (Bowtie [5]) for the read alignment and has limited functionality for analyzing the results. . The results in Myrna are returned in the form of per gene P-values and Q-values for differential expression, a raw count table, an RPKM table (of reads per kilo base of exon model per million mapped reads) and coverage plots.

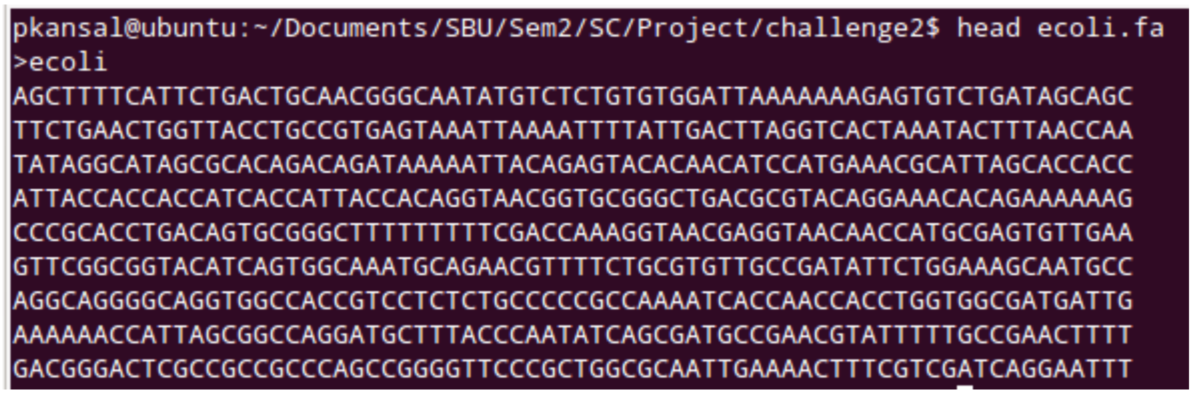
HOW IS OUR IMPLEMENTATION DIFFERENT

- We use a better program BWA for reads alignment which is fast and accurate for short reads
- We give heat map and parallel coordinates for easy differential expression analysis

WHAT ARE THE INPUTS AND OUTPUTS?

Input: The input consists of following files:

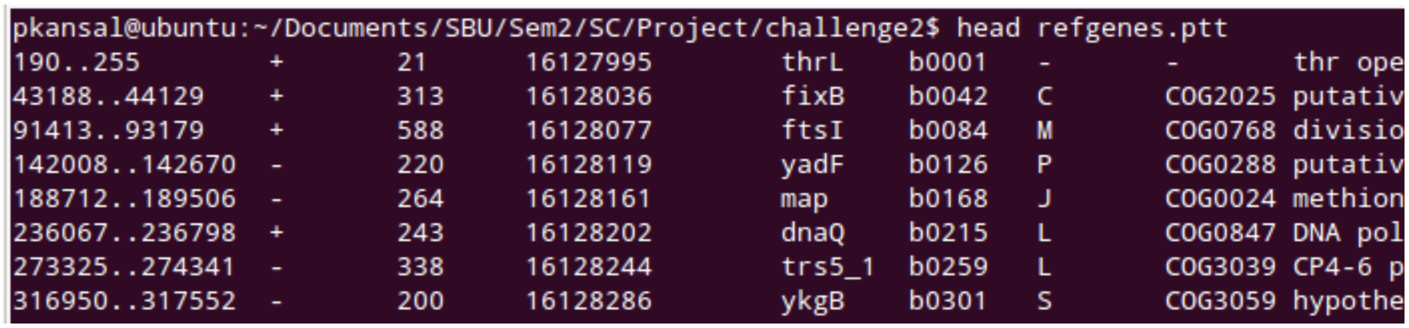
- **Genome File**: this file is in FASTA[6] format and contains the required genome to be analyzed. A typical file looks like following:



```
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2$ head ecoli.fa
>ecoli
AGCTTTTCATTCTGACTGCAACGGGCAATATGTCTCTGTGTGGATTAAAAAAAGAGTGTCTGATAGCAGC
TTCTGAACTGGTTACCTGCCGTGAGTAAATTAAAATTTTATTGACTTAGGTCACTAAATACTTTAACCAA
TATAGGCATAGCGCACAGACAGATAAAAATTACAGAGTACACAACATCCATGAAACGCATTAGCACCACC
ATTACCACCACCATCACCATTACCACAGGTAACGGTGCGGGCTGACGCGTACAGGAAACACAGAAAAAAG
CCCGCACCTGACAGTGCGGGCTTTTTTTTTCGACCAAAGGTAACGAGGTAACAACCATGCGAGTGTTGAA
GTTCGGCGGTACATCAGTGGCAAATGCAGAACGTTTTCTGCGTGTTGCCGATATTCTGGAAAGCAATGCC
AGGCAGGGGCAGGTGGCCACCGTCCTCTCTGCCCCCGCCAAAATCACCAACCACCTGGTGGCGATGATTG
AAAAAACCATTAGCGGCCAGGATGCTTTACCCAATATCAGCGATGCCGAACGTATTTTTGCCGAACTTTT
GACGGGACTCGCCGCCGCCCAGCCGGGGTTCCCGCTGGCGCAATTGAAAACTTTCGTCGATCAGGAATTT
```

Figure 1

- **Reference Genes**: this file is in PTT format and contains the genes, which needs to be analyzed. A typical file looks like following:

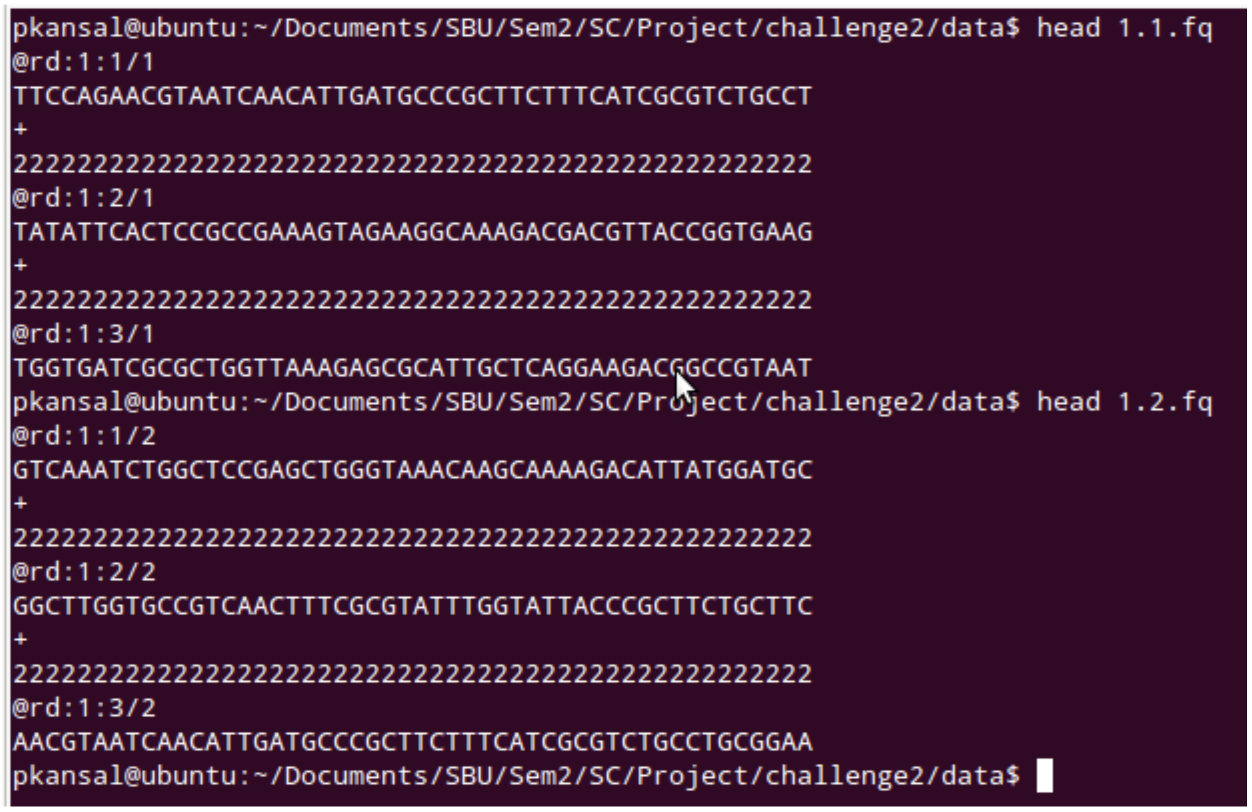


```
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2$ head refgenes.ptt
190..255        +       21      16127995        thrL    b0001   -       -       thr ope
43188..44129    +       313     16128036        fixB    b0042   C       COG2025 putativ
91413..93179    +       588     16128077        ftsI    b0084   M       COG0768 divisio
142008..142670  -       220     16128119        yadF    b0126   P       COG0288 putativ
188712..189506  -       264     16128161        map     b0168   J       COG0024 methion
236067..236798  +       243     16128202        dnaQ    b0215   L       COG0847 DNA pol
273325..274341  -       338     16128244        trs5_1  b0259   L       COG3039 CP4-6 p
316950..317552  -       200     16128286        ykgB    b0301   S       COG3059 hypothe
```

Figure 2

  - Each such file contains multiple genes at a time, lets denote it as "g"
  - Each row in this file contains 1 gene. The column for our importance are:
    - First column: contains the range of the gene, lets denote it as "r". Obviously as you can see "r" varies for different genes
    - Fourth column: contains gene-id

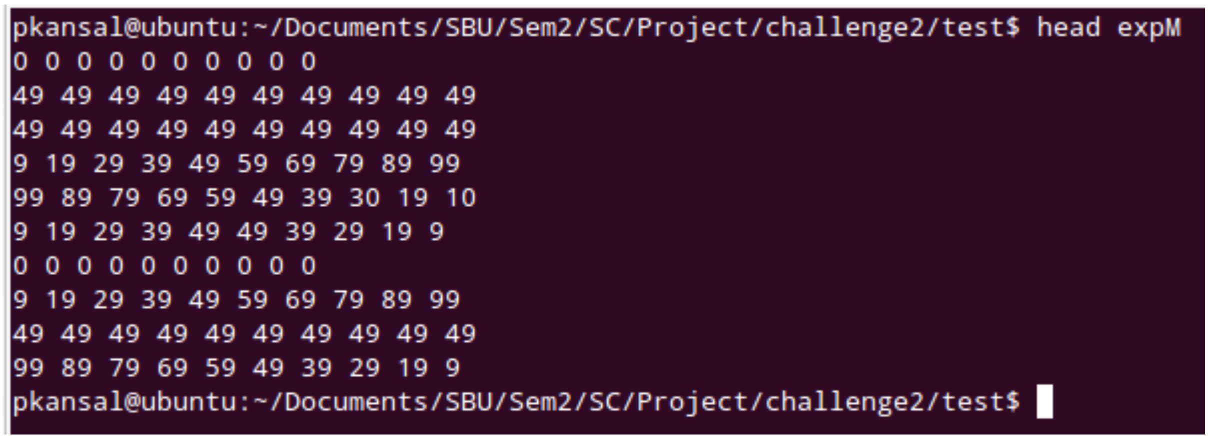
- **Paired-ended Short Reads**: these represent the reads of the genome taken at different times (lets say at time t1, t2 … t10). To measure the gene expression, multiple reads (lets say "t") are required over a period of time. A typical paired-ended file for time t1 looks like following:



```
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/data$ head 1.1.fq
@rd:1:1/1
TTCCAGAACGTAATCAACATTGATGCCCGCTTCTTTCATCGCGTCTGCCT
+
22222222222222222222222222222222222222222222222222
@rd:1:2/1
TATATTCACTCCGCCGAAAGTAGAAGGCAAAGACGACGTTACCGGTGAAG
+
22222222222222222222222222222222222222222222222222
@rd:1:3/1
TGGTGATCGCGCTGGTTAAAGAGCGCATTGCTCAGGAAGACGGCCGTAAT
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/data$ head 1.2.fq
@rd:1:1/2
GTCAAATCTGGCTCCGAGCTGGGTAAACAAGCAAAAGACATTATGGATGC
+
22222222222222222222222222222222222222222222222222
@rd:1:2/2
GGCTTGGTGCCGTCAACTTTCGCGTATTTGGTATTACCCGCTTCTGCTTC
+
22222222222222222222222222222222222222222222222222
@rd:1:3/2
AACGTAATCAACATTGATGCCCGCTTCTTTCATCGCGTCTGCCTGCGGAA
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/data$
```

Figure 3

Output: The output consists of just one file which is an expression matrix. So, this file contains "g" rows and "t" columns to represent the gene expression of "g" genes over a period of "t". A typical file looks like following:



```
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/test$ head expM
0 0 0 0 0 0 0 0 0 0
49 49 49 49 49 49 49 49 49 49
49 49 49 49 49 49 49 49 49 49
9 19 29 39 49 59 69 79 89 99
99 89 79 69 59 49 39 30 19 10
9 19 29 39 49 49 39 29 19 9
0 0 0 0 0 0 0 0 0 0
9 19 29 39 49 59 69 79 89 99
49 49 49 49 49 49 49 49 49 49
99 89 79 69 59 49 39 29 19 9
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/test$
```

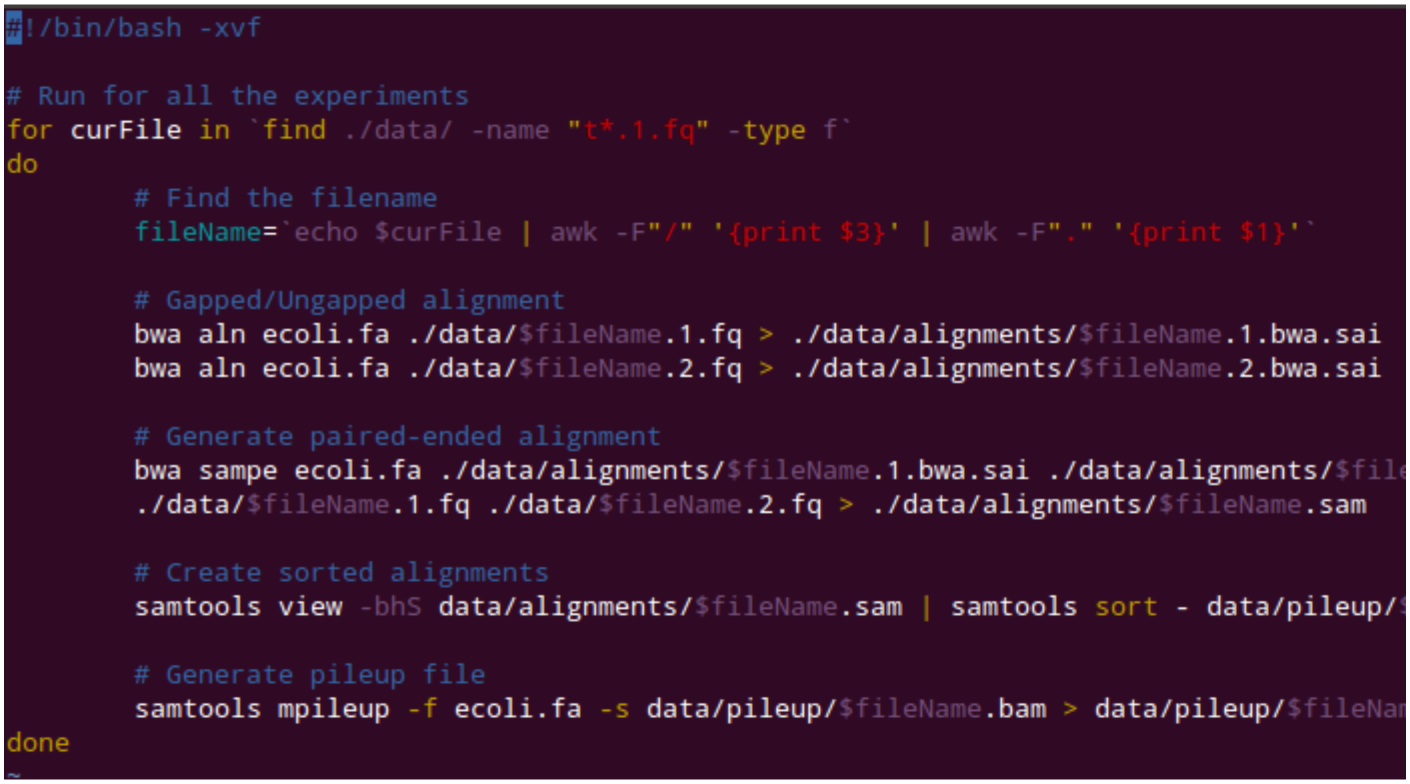
Figure 4

From these expression matrixes we are supposed to generate heat map, parallel coordinates and time series using R.

WHAT WE HAVE DONE?

- We did the project in two steps
  - Serial Implementation
    - First, we implemented a serial pipeline to understand the complexities involved
    - This involved understanding and setting up the various tools used in the pipeline
    - Understanding the features which jnomic tools offers and what it does not (the features which will have to be implemented by us)
  - Second, we implemented a parallel pipeline to solve the problem
    - Implementing the pipeline which connects the features already provided by Jnomic-tools
    - Extending the functionality of Jnomic-tools to get the pileup file required to generate the expression matrix
    - Writing the map reduce program which will generate the expression matrix

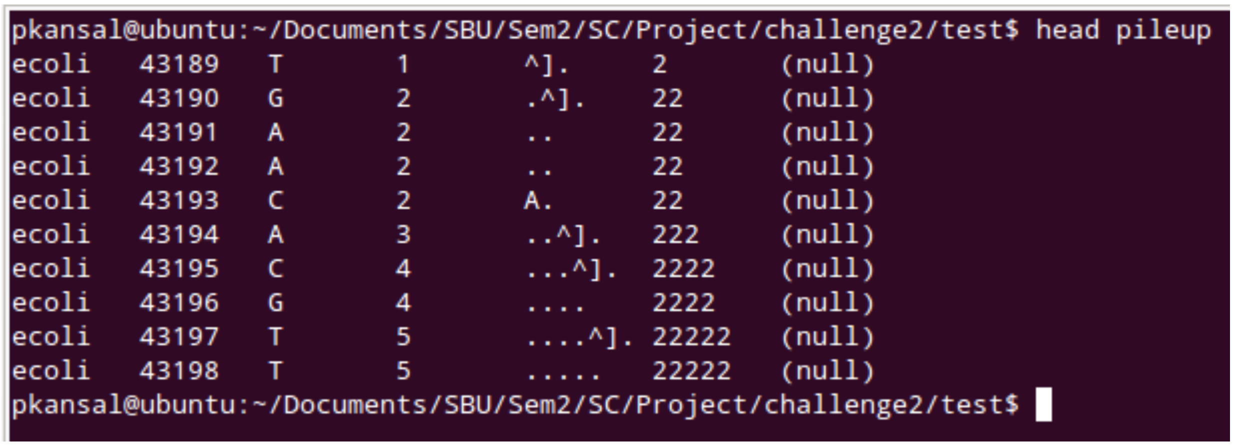
- **Serial Pipeline**
  - We first chained multiple commands together in *genPileUp.sh* to create serial pipeline as shown below



```
#!/bin/bash -xvf

# Run for all the experiments
for curFile in `find ./data/ -name "t*.1.fq" -type f`
do
        # Find the filename
        fileName=`echo $curFile | awk -F"/" '{print $3}' | awk -F"." '{print $1}'`

        # Gapped/Ungapped alignment
        bwa aln ecoli.fa ./data/$fileName.1.fq > ./data/alignments/$fileName.1.bwa.sai
        bwa aln ecoli.fa ./data/$fileName.2.fq > ./data/alignments/$fileName.2.bwa.sai

        # Generate paired-ended alignment
        bwa sampe ecoli.fa ./data/alignments/$fileName.1.bwa.sai ./data/alignments/$file
        ./data/$fileName.1.fq ./data/$fileName.2.fq > ./data/alignments/$fileName.sam

        # Create sorted alignments
        samtools view -bhS data/alignments/$fileName.sam | samtools sort - data/pileup/$

        # Generate pileup file
        samtools mpileup -f ecoli.fa -s data/pileup/$fileName.bam > data/pileup/$fileNam
done
```

Figure 5

  - This generated the required pileup for us. A typical pile file looks like following:



```
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/test$ head pileup
ecoli   43189   T       1       ^].     2       (null)
ecoli   43190   G       2       .^].    22      (null)
ecoli   43191   A       2       ..      22      (null)
ecoli   43192   A       2       ..      22      (null)
ecoli   43193   C       2       A.      22      (null)
ecoli   43194   A       3       ..^].   222     (null)
ecoli   43195   C       4       ...^].  2222    (null)
ecoli   43196   G       4       ....    2222    (null)
ecoli   43197   T       5       ....^]. 22222   (null)
ecoli   43198   T       5       .....   22222   (null)
pkansal@ubuntu:~/Documents/SBU/Sem2/SC/Project/challenge2/test$
```

Figure 6

    - There are two columns of our importance, second and fourth:
        - *First column*: mentions the index of the nucleotide "T" (see 3rd column) in the genome "ecoli" (see 1st column)
        - *Fourth column*: mentions the depth of the nucleotide

- We then created the expression matrix using a C++ program, *calAvgDepth.cpp*, to calculate the gene expression matrix by calculating the average depth of the genes present in an input file, *refgenes.ptt*. A high level pseudo-code is shown below:

```
Generate pileup files by leveraging existing tools (Figure 5)
For each gene, gi, present in PTT file
    For each pileup file, pi
        Load pileup file in a data structure, an array i.e
            array[pi.index] = pi.depth (pi.index is always in sorted order)
        totDepth = 0
        For index in gi.start to gi.end
            do a binary search on "array"
                if index found in array, calculate totDepth
                    totDepth += array[index]
        totDepth /= gi.end - gi.start + 1
        expMatrix[gi][pi] = totDepth
Dump expression matrix in a file
```

- So, the total running time of serial implementation is ***O(gt(n + rlogn))***, where:
    - *g*: number of genes in PTT file
    - *t*: number of experiments
    - *n*: number of entries in each pileup file
    - *r*: average range of genes in PTT file

- **Parallel Pipeline**
  - We created the parallel pipeline using another script, *genExpMat.sh*
  - This script leverages the parallel version of BWA and SAMTools present in Jnomics [7] to create the pileup files
  - Following pseudo-code explains the whole operation:

```
Generate pileup files by leveraging parallel version of existing tools in Jnomics

Process each pileup file using using 1 mapper in which we do following:

    In setup(), initialize a data structure, refGenes:

        For each record gi in PTT file:

            Store gi.geneid against (gi.start, gi.end) in refGenes

    For each record pi in a single pileup file:

        In map (), binary search the pi.index in refGenes to figure out the gene id:

            If pi.index found in refGenes

                geneID = refGenes[pi.index]

                experimentID = get the experiement id from HDFS folder name

                Emit <geneID-geneRange-experimentID, pi.depth>

Process the records emitted by mapper in reducer:

    In setup():

        Initialize hash map, hm to 0

    In reduce():

        For each value in Iterable values

            expID = key.experimentID

            hm[expID] += value.depth

        Final value = hm[expID]/key.range
```

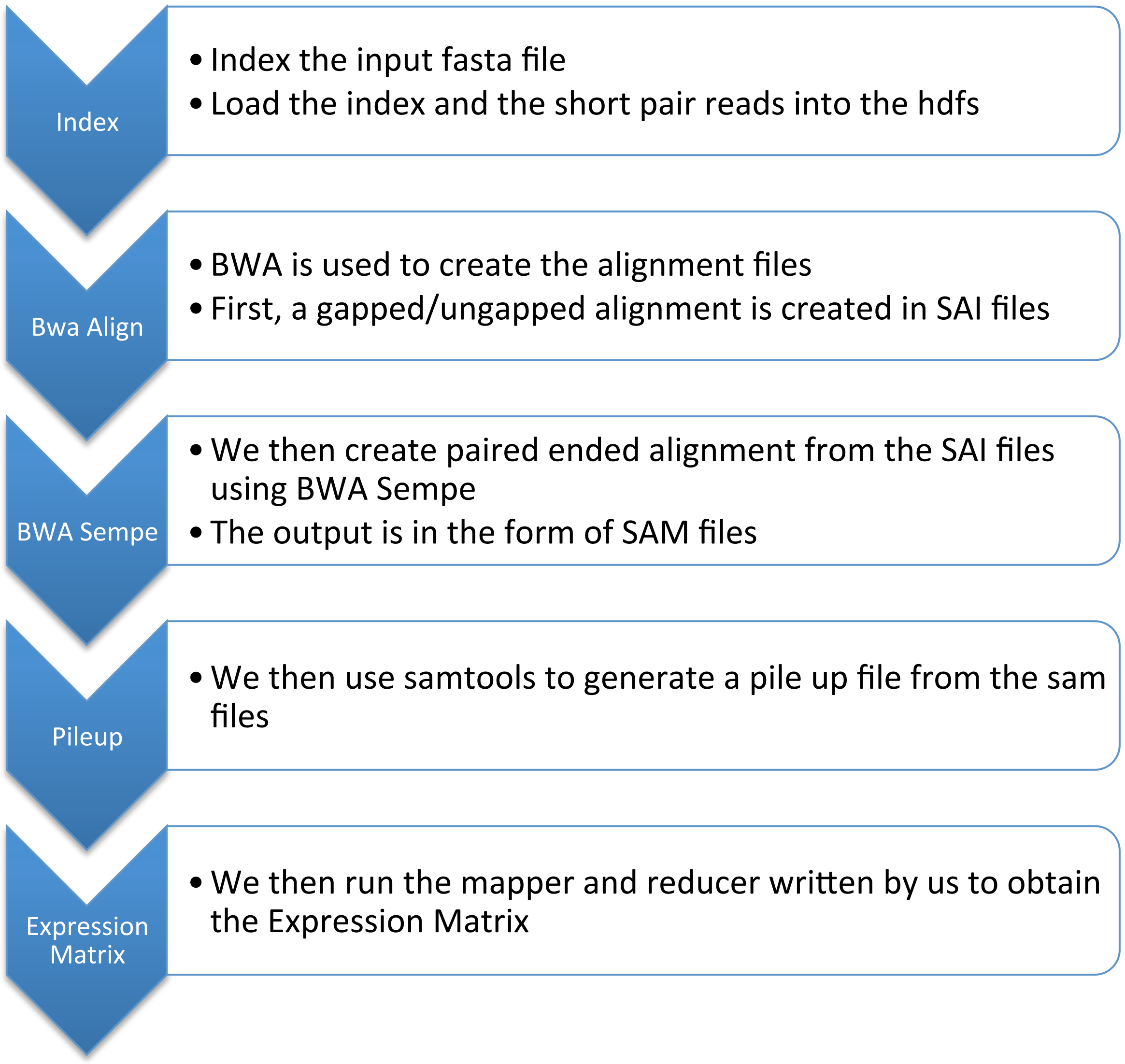
Write the value to the file "*geneID*"

In mergeExp.java

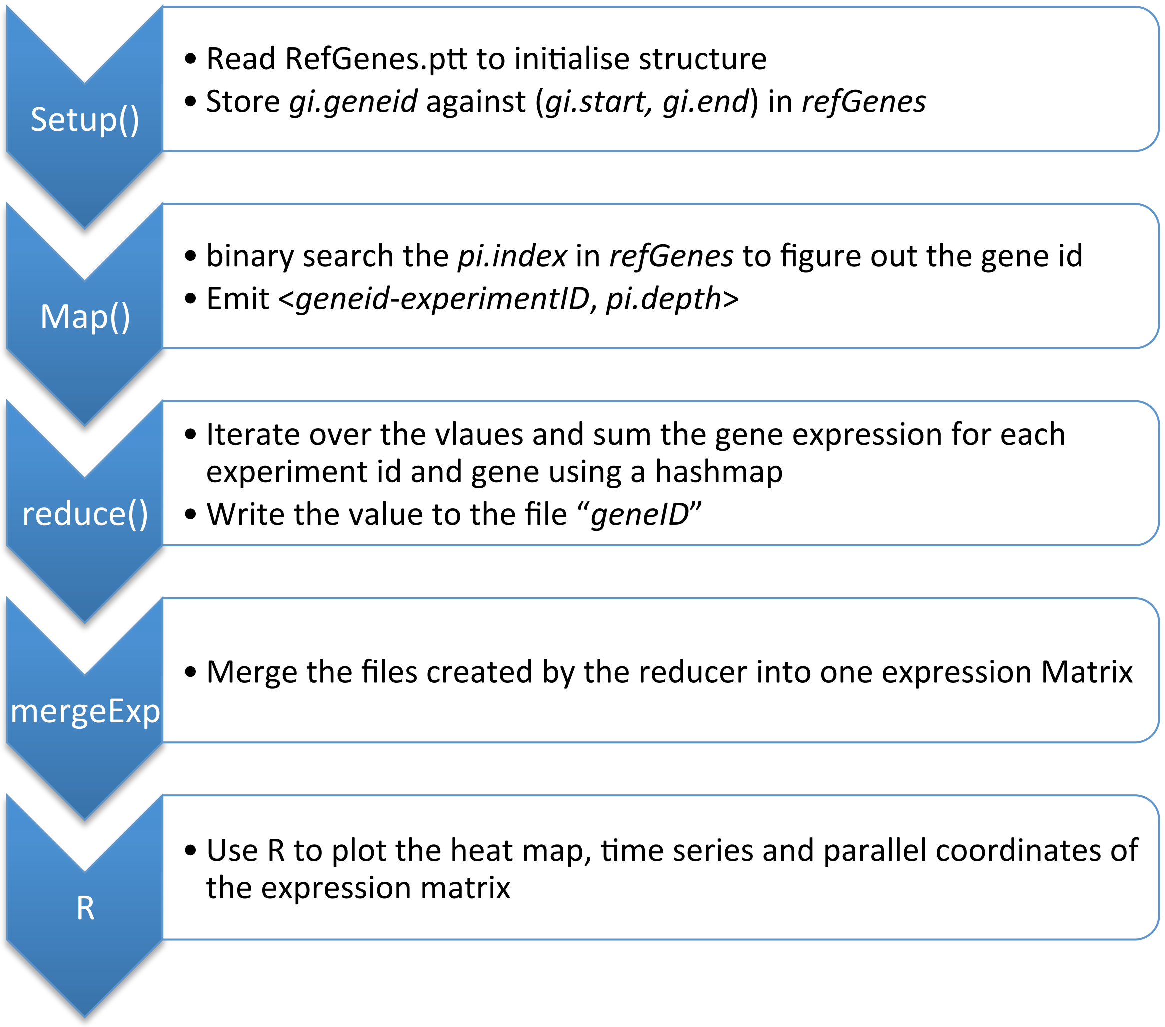
Merge all the files to a single file "*ExpMatrix*"

- The running time of this algorithm is **O (g+n.log(g)+n*t+g)**

Flowchart to summarize the pipelining of the various tools

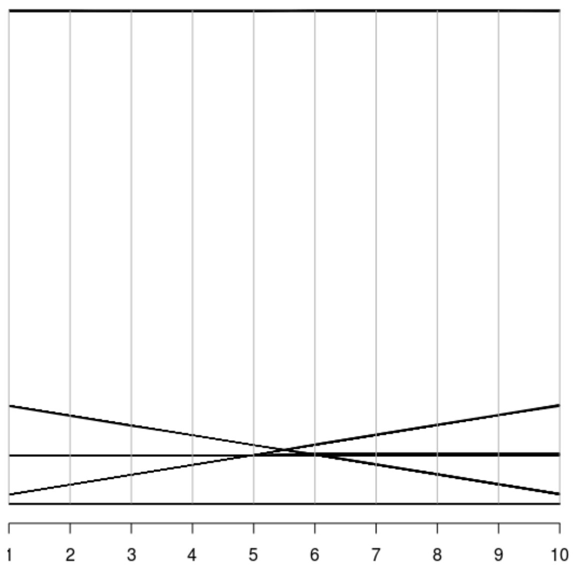

Index
• Index the input fasta file
• Load the index and the short pair reads into the hdfs
Bwa Align
• BWA is used to create the alignment files
• First, a gapped/ungapped alignment is created in SAI files
BWA Sempe
• We then create paired ended alignment from the SAI files using BWA Sempe
• The output is in the form of SAM files
Pileup
• We then use samtools to generate a pile up file from the sam files
Expression Matrix
• We then run the mapper and reducer written by us to obtain the Expression Matrix

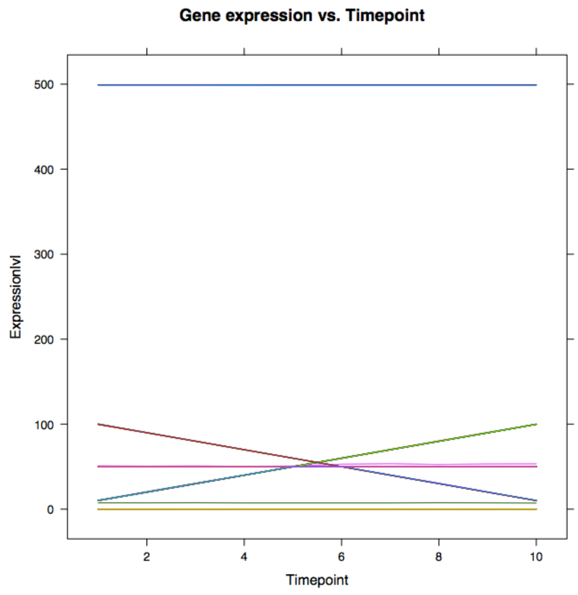
Flowchart to summarize the parallel MapReduce program written


Setup()
• Read RefGenes.ptt to initialise structure
• Store *gi.geneid* against (*gi.start, gi.end*) in *refGenes*
Map()
• binary search the *pi.index* in *refGenes* to figure out the gene id
• Emit <*geneid-experimentID*, *pi.depth*>
reduce()
• Iterate over the vlaues and sum the gene expression for each experiment id and gene using a hashmap
• Write the value to the file "*geneID*"
mergeExp
• Merge the files created by the reducer into one expression Matrix
R
• Use R to plot the heat map, time series and parallel coordinates of the expression matrix

RESULTS

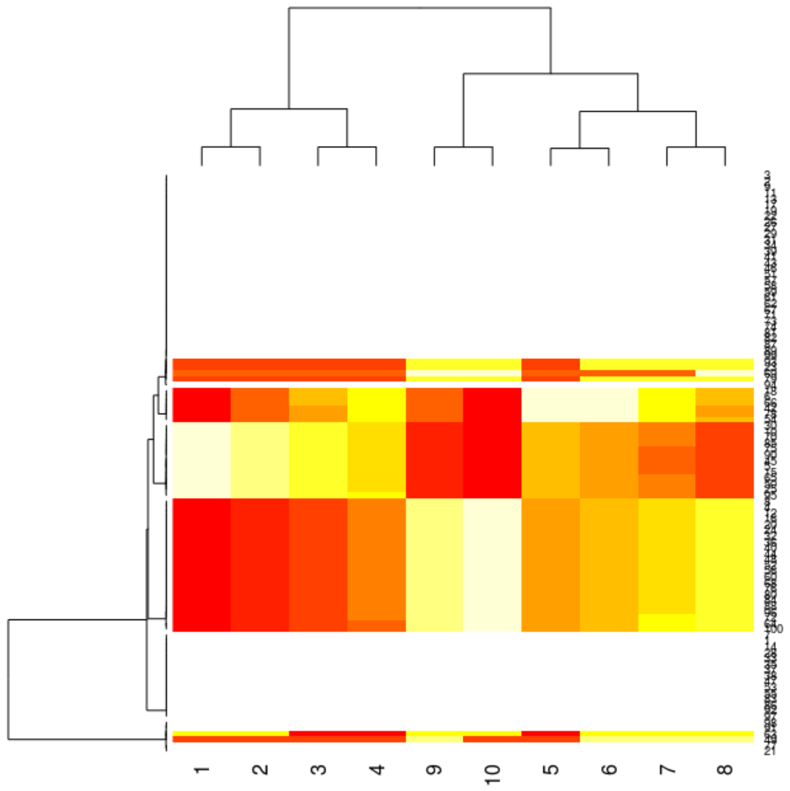
The graphs obtained for the data set 1 are shown below



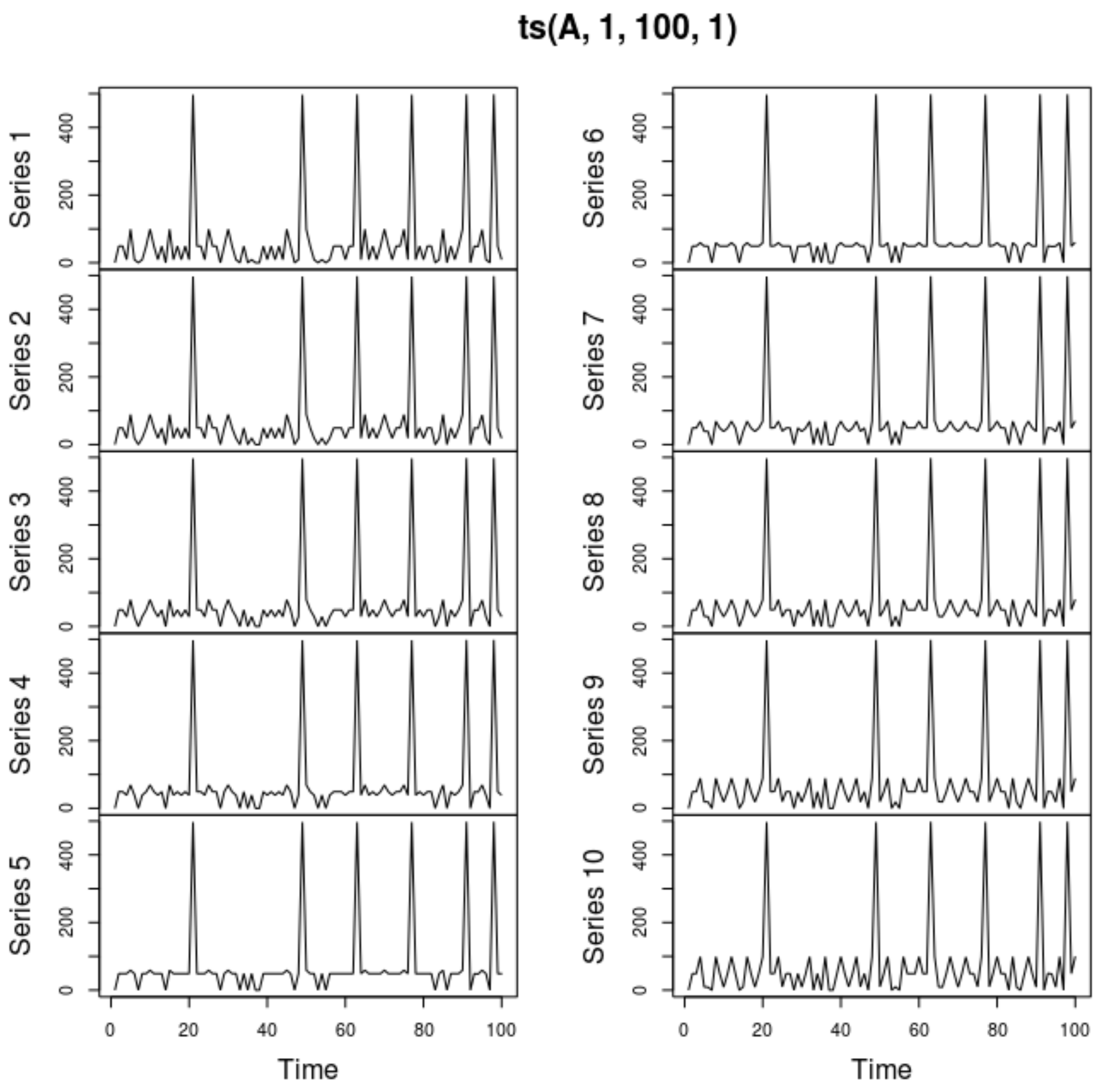


OUR RESULTS **EXPECTED RESULTS**

Parallel Coordinate results for the DataSet1 provided

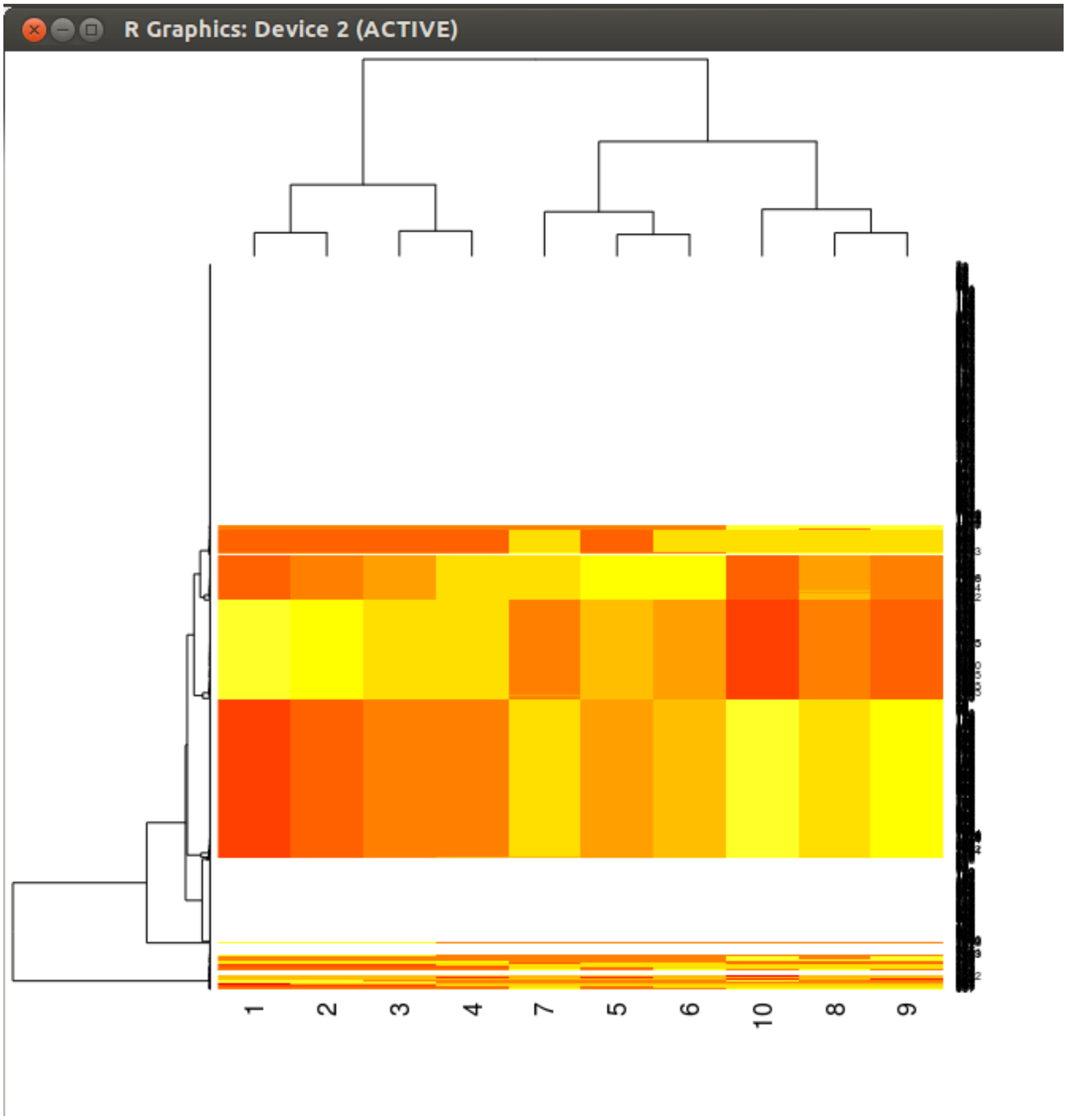


Heat Map depicting the distribution for the genes

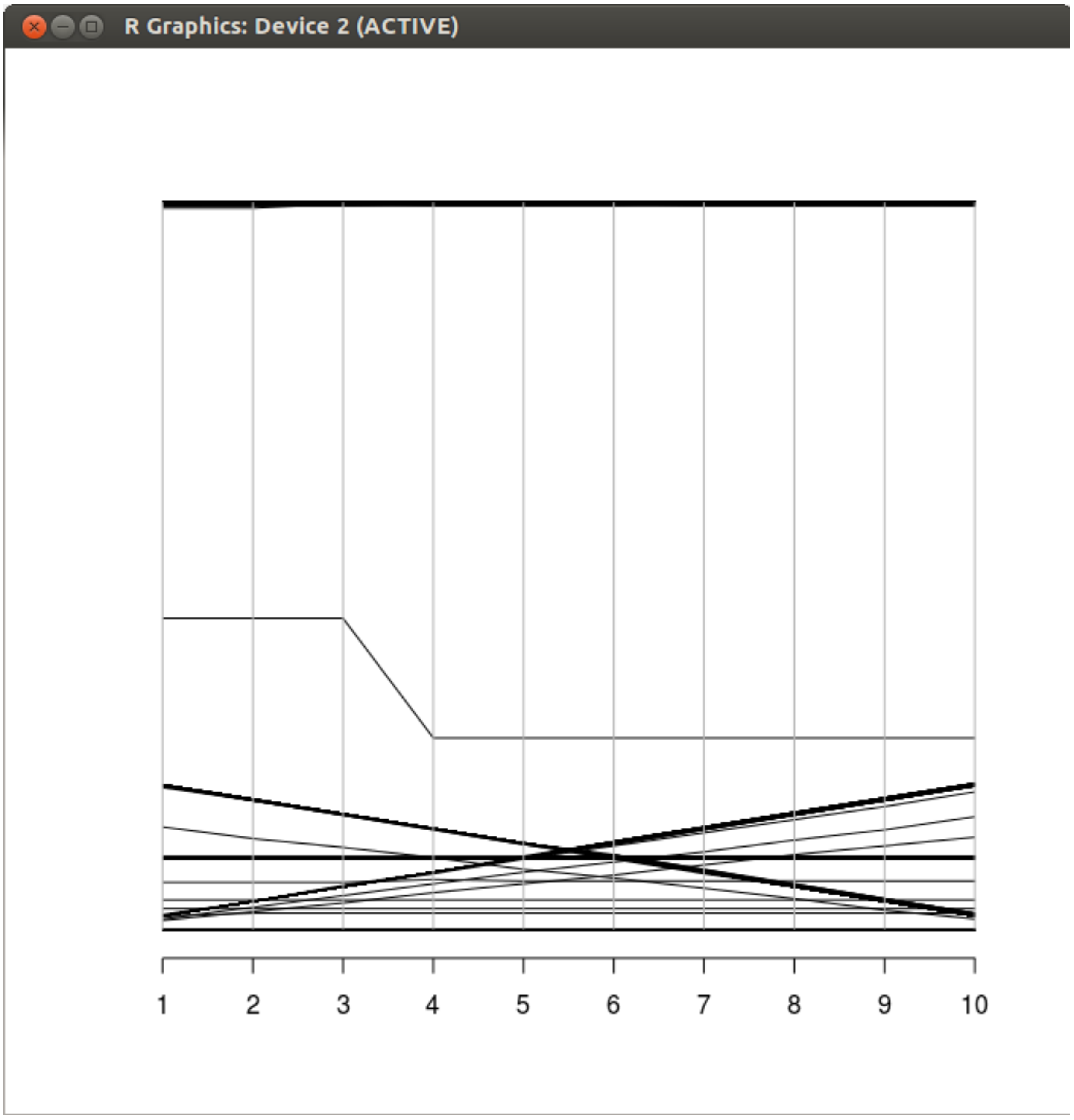


Time series for the Data Set1

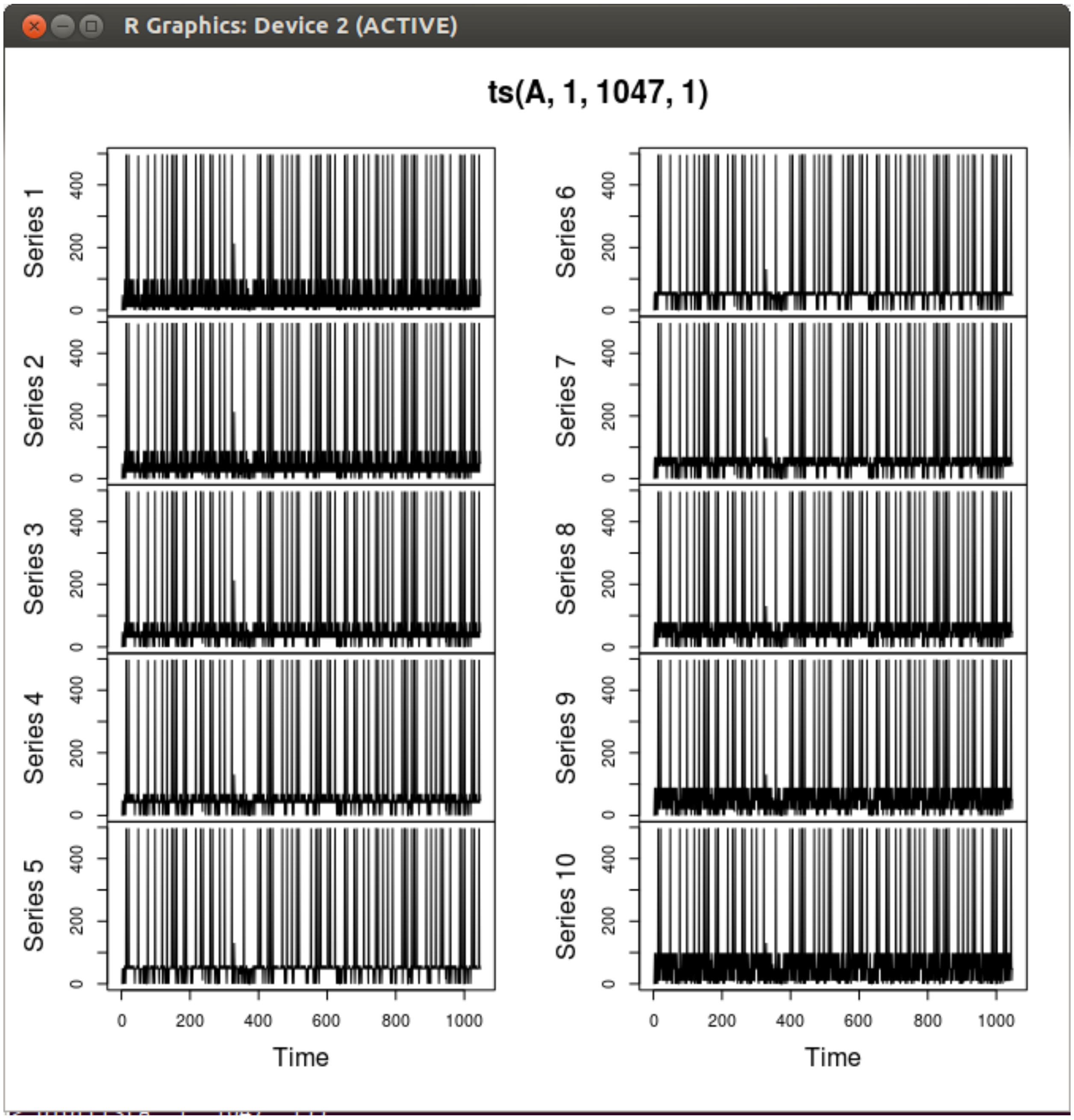
Results for the dataset 2 which was 10 times larger than dataset 1



Heat Map



Parallel Coordinate graph



Time Series for data set 2

SOURCE CODE

Base Code

JnomicsTools already have an existing platform to schedule hadoop jobs to calculate alignment of genes. However, we made changes and added features to the following files to compute the gene expression matrix

JnomicsMain.java

This is the main job handler class, which handles the commands fired from the shell script. It handles the arguments and calls the required functions from the respective classes to do the necessary computation.

SamtoolsSnpReduce.java

This class did not exist in the project. However, we needed a reducer, which would produce a pileup file. Hence, we wrote a reducer class, which takes input from SamTools map and reduces it to a pileup file.

geaMap.java

This is the mapper class, which reads the input file (refGenes.ptt) and the pileup file created from the above class. It emits data with Key – GeneID-Experiment_ID and value – depth od expression.

geaReduce.java

This takes in a key value pair and sums up the depth of expression of each gene for a given experiment id and writes it to a file

mergeExp.java

This merges all the files created by the reducer to create a single expression Matrix

EXECUTION

Folder Contents

- Parallel
    - Contains parallel implementation
    - Contains following files/folders:
    - genExpMat.sh - main script to trigger the creation of expression matrix
    - jnomics-code/ - contains Jnomics code
    - Serial
    - Contains serial implementation
    - Contains following files:
        - genPileUp.sh
        - calAvgDepth.cpp
    - Please note that there is a minor bug in serial implementation. The bug is that we get an answer, which is close to the actual answer but is not exactly the same. We did not find it a good idea to spend much time on it as the main priority was
- Parallel implementation.
    - Output
        - Generated on our m/c for 100 genes
            - heatmap.png
            - timeseries.png
            - parcoord.png
        - Generated on Prof. Schatz's cluster for 1047 genes
            - heatmap-1047.png
            - timeseries-1047.png
            - parcoord-1047.png
- How to compile
    - parallel
        - cd parallel
        - chmod 755 genExpMat.sh
        - cd jnomics-code
        - ant jar
        - this will create jnomics-tools.jar
- How to execute
    - make sure you have latest version of R, BWA, SAMTools and bcftools installed
    - first of all, unzip following file in a folder named "test":
    - http://schatzlab.cshl.edu/data/challenges/challenge2.tgz
    - mv challenge2.tgz test.tgz
    - copy the following binaries into "test" folder:

- cp `which bwa` test
- cp `which samtools` test
- cp `which bcftools` test
- from "test" directory, fire following command:
- start hadoop
- rm -f nohup.out; ../parallel/genExpMat.sh ecoli.fa refgenes.ptt . 10 ../parallel/jnomics-code/jnomics-tools.jar bwa bcftools samtools expM
- this will create an o/p file "expM" in "test" directory
- now run R by typing "R" on command line and then do following:
  - A <- matrix(scan("expM", n = 100 * 10), 100, 10, byrow = TRUE)
  - 100 is the number of genes in test/refGenes.ptt file
  - 10 is the number of experiements test/t1.1.fq ... test/t10.1.fq
  - you can change these values as per the data set
  - heatmap(A): plots the heatmap (pops up a new window)
    - you can verify it with the output/heatmap.png
  - plot(ts(A, 1, 100, 1)): plots the time series (pops up a new window)
    - you can verify it with the output/timeseries.png
  - parcoord(A): plots the parallel coordinates (pops up a new window)
    - you can verify it with the output/parcoord.png

TEST RESULTS

We have executed the serial and parallel version of the program for 3 datasets. These 3 datasets are pretty small, as compared to the actual genomic data set that the algorithm is meant for. However, it does demonstrate the scalability of our pipeline.

Data Set	Time Taken (seconds)		
	Serial	Parallel	Speedup
100 genes x 10 time series	2.607	23.425	0.111291355
1000 genes x 10 time series	32.619	92.19	0.353823625
1000 genes x 100 time series	420.195	193.1	2.176048679

- As we can see, the speedup = serial / parallel, increases as the dataset increases

REFERENCES

[1] http://bio-bwa.sourceforge.net/

[2] http://samtools.sourceforge.net/

[3] http://www.r-project.org/

[4] http://genomebiology.com/2010/11/8/R83

[5] http://bowtie-bio.sourceforge.net/index.shtml

[6] http://en.wikipedia.org/wiki/FASTA_format

[7] http://sourceforge.net/projects/jnomics/