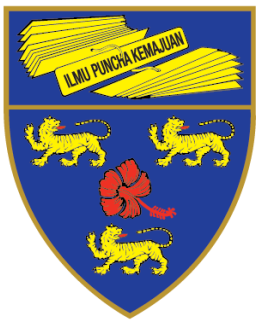


UNIVERSITI MALAYA

WIX 1002

FUNDAMENTAL OF PROGRAMMING

SEMESTER 1, 2022/2023

ASSIGNMENT REPORT

LECTURER	DR. LEE CHIEW SUN
OCCURRENCE	3

GROUP MEMBERS:

Name	Matrix Number
AINA NUR MALISA	U2101529/2
MUHAMMAD AFIQ	U2101729/2
IVAN KHIRUSMAN	U2102368/2
HAFIZ AIMAN	U2100875/2

REQUIREMENTS ANALYSIS

1) Number of jobs created and ended within a given time range
2) Number of jobs by partitions
3) Number of jobs causing error and the corresponding user
4) Arrange execution time of the jobs submitted to UMHPC

ARCHITECTURAL DESIGN

1) File Input

```
package fop;
import java.io.*;
import java.util.*;
public class Fop {
   public static void main(String[] args) {
      File file = new File("C:/Users/HP/Desktop/SE/WIX1002/extracted_log.txt");
      fileinput first = new fileinput(file);
      first.displaydata();
   }
}
```

2) Number of jobs Created and Ended within a given Time Range

```
package fop;
import java.io.*;
import java.util.*;
import java.util.regex.Matcher;
import java.util.regex.Pattern;

public class fileinput {
   private File file;
   public fileinput(File f){
      file = f;
   }

   public void jobs_created_done(){
```

```
Scanner input = new Scanner(System.in);
int counter = 0;
String month=null;

//find month in num and word
System.out.printf("Enter month from June to December [06-12]: ");
String monthnum = input.nextLine();
month = choosemonth(monthnum);

try{
   Scanner scf = new Scanner(new FileInputStream(file));

   //searching for number of jobs started
   while(scf.hasNextLine()){
      String line = scf.nextLine();
      if(line.matches("(.*)Allocate(.*)") && line.matches("(.*)-"+monthnum+"-(.*)") ){
         counter++;
         }
      }
}catch(FileNotFoundException e){
   System.out.println(e);
}
//display jobs started
System.out.println("Number of jobs started in "+month+" : " +counter);

counter =0;
try{
   Scanner scf = new Scanner(new FileInputStream(file));

   //searching for number of jobs completed
   while(scf.hasNextLine()){
      String line = scf.nextLine();
      if(line.matches("(.*)done(.*)") && line.matches("(.*)-"+monthnum+"-(.*)") ){
```

```
                counter++;
            }
        }
    }catch(FileNotFoundException e){
        System.out.println(e);
    }//display jobs done
    System.out.println("Number of jobs done: " +counter);
}

public String choosemonth(String num){
    switch(num){
        case "06","6" -> {
            return "June";
        }
        case "07","7" -> {
            return "July";
        }
        case "08","8" -> {
            return "June";
        }
        case "09","9" -> {
            return "June";
        }
        case "10" -> {
            return "June";
        }
        case "11" -> {
            return "June";
        }
        case "12" -> {
            return "June";
        }
     }
    return null;
```

```
}
```

3) Number of jobs Partitions

```
public void jobs_partition(){
    try{
        Scanner scf = new Scanner(new FileInputStream(file));

        // declaring the target words and counter for each partitions
        String word1 ="cpu-epyc";
        String word2 = "cpu-opteron";
        String word3 = "gpu-titan";
        String word4 = "gpu-v100s";
        String word5 = "gpu-k10";
        String word6 = "gpu-k40c";

        int epycCount = 0;
        int opteronCount = 0;
        int titanCount = 0;
        int v100sCount = 0;
        int k10Count = 0;
        int k40cCount = 0;

        //searching for no. of jobs by partitions
        while(scf.hasNext()){
            String line = scf.nextLine();

            if(line.contains("Partition=")){
                String[] words = line.split("Partition=");
                if(words[1].contains(word1))epycCount++;
                else if(words[1].contains(word2))opteronCount++;
                else if(words[1].contains(word3))titanCount++;
                else if(words[1].contains(word4))v100sCount++;
                else if(words[1].contains(word5))k10Count++;
                else if(words[1].contains(word6))k40cCount++;
```

```
            }
        }
        scf.close();

        //displaying the output
        System.out.println("\nNo of jobs in EPYC is "+ epycCount);
        System.out.println("No of jobs in Gpu-Titan is "+ titanCount);
        System.out.println("No of jobs in Opteron is "+ opteronCount);
        System.out.println("No of jobs in Gpu-v100s is "+ v100sCount);
        System.out.println("No of jobs in Gpu-k10 is "+ k10Count);
        System.out.println("No of jobs in Gpu-k40c is "+ k40cCount);

    }catch(FileNotFoundException e){
        System.out.println(e);
    }
}
```

4) Number of jobs Causing Error and the Corresponding User

```
public void jobs_error(){
    try{
        Scanner s = new Scanner(new FileInputStream(file));

        //count total lines
        int totallines=0;
        while(s.hasNextLine()){
            s.nextLine();
            totallines++;
        }
        s.close();

        //put lines into String arrays
        String[] line = new String[totallines];
        Scanner u = new Scanner(new FileInputStream(file));
        for(int y=0;y<totallines;y++){
```

```
            line[y] = u.nextLine();
        }
        u.close();

        //split lines into parts divided by space " "
        String[][] lineinfo = new String [totallines][];
        for(int y=0;y<totallines;y++){
            lineinfo[y] = line[y].split("\\W+");
        }

        //checking total number of errors and the corresponding user
        int numerrors = 0;
        String[]users = new String[totallines];
        for(int i=0;i<lineinfo.length-1;i++){
            if(lineinfo[i][7].equalsIgnoreCase("error")){
                if(lineinfo[i].length-1>=13&&lineinfo[i][13].equalsIgnoreCase("user")){
                    users[i] = lineinfo[i][14];
                    numerrors++;
                }
            }
        }
        System.out.println("\nNumber of errors: "+numerrors);

        //remove null from array of users
        String[] newusers =
Arrays.stream(users).filter(Objects::nonNull).toArray(String[]::new);

        //count number of same users
        Map<String, Integer> result = new HashMap<>();
        for(String g : newusers){

            if(result.containsKey(g)){
                //if the map contain this key then just increment your count
                result.put(g, result.get(g)+1);
```

```
            }else{
                //else just create a new node with 1
                result.put(g, 1);
            }
        }

        //converting map into String array
        String[] key = result.keySet().toString().split("\\W+");
        String[] values = result.values().toString().split("\\W+");

        //converting String array to int array
        int[] value = new int[values.length];
        for(int o=0;o<values.length;o++){
            if(!"".equals(values[o]))
                value[o] = Integer.parseInt(values[o]);
        }

        //display corresponding users and their count
        System.out.println("Users:");
        for (int i = 1; i < result.size(); i++) {
            System.out.println( key[i] + " = " + value[i] );
        }

    }catch(FileNotFoundException e){
            System.out.println("File is not found.");
    }
  }
```

5) Arrange Execution Time of the Jobs Submitted to UMHPC

```
public void jobs_execution_time(){
    String [][] data = new String [4][9000];
    //data[0] = id
    //data[1] = start line
```

```
        //data[2] = done line
        //data[4] = job duration

        //start storing data into the array
        //get id from done line
        try{
            Scanner scf = new Scanner(new FileInputStream(file));
            int counter=0;
            String id ;

            //skip the first 10 finished job line as some of its schedulde didnt include in the
data set
            //note that 42814 is where we start but 42815 is removed
            for(int i =0; i<10;){
                String line = scf.nextLine();
                if(line.matches("(.*)done(.*)")){
                    i++;
                    }
            }
            //there is 8469 done line left in the data
            while(scf.hasNextLine()){
                String line = scf.nextLine();
                if(line.matches("(.*)done(.*)") && line.matches("(.*)JobId(.*)")){
                    id = extractid(line);
                    data[0][counter]=id;
                    //id_array[counter]=id;
                    data[2][counter]=line;
                    //System.out.println(counter+"\t"+data[0][counter] + "\t" +data[2][counter]);
                    counter++;
                }
            }
            scf.close();
        }catch(FileNotFoundException e){
            System.out.println(e);
```

```
        }

        //********************get scheduled line *************
        try{
            Scanner scf = new Scanner(new FileInputStream(file));
            while(scf.hasNextLine()){
                String row = scf.nextLine();
                //find row with allocate keyword (start job)
                if(row.matches("(.*)Allocate(.*)")){
                    //get the id of the row
                    String rowid = extractid(row);
                    //find id in array that equals to rowid to store line with its partner
                    for(int i=0;i<data[0].length;i++){
                        String tempid  = data[0][i];
                        if(tempid == null)break;
                        if(tempid.equals(rowid)){
                            data[1][i] = row;
                        }
                    }
                }
            }
            scf.close();
        }catch(FileNotFoundException e){
            System.out.println(e);
        }
        //****************end of data collecting****************

        //**************calculate sum of time and input to array*******************
        //note that we can also compute average here*******************************
        for(int i=0;i<data[0].length;i++){
        //due to some consequence in handling job, some data in data array is null bcause
the job didnt finish or start properly
            if(data[0][i]==null)break;
            if(data[1][i]==null)continue;
```

```
            if(data[2][i]==null)continue;
            String start = data[1][i];
            String end = data[2][i];
            int sumtime=timediff(gettime(start), gettime(end));
            data[3][i]= String.valueOf(sumtime);//input duration for each job
        }
        //loop to calculate sum,counter and average (also can be include in previous loop)
        int counter=0;
        double sum =0;
        for(int i =0;i<data[0].length;i++){
            if(data[0][i]==null)break;//stop when all id has been read
            if(data[1][i]==null)continue;//skip if job start not in time range
            if(data[2][i]==null)continue;//skip if job end exceed time range
            sum+= Double.valueOf(data[3][i]);
            counter++;
        }
        System.out.printf("\nThe average time for each job is : %.3f seconds\n" ,
sum/counter);
    }
    //***************extract id from line ***************
    public static String extractid(String line){
        String[] words = line.split(" ");
        String id =null;
        for(String word : words){
            if(word.matches("(.*)JobId(.*)")){
                id = word;
                break;
            }
        }
        return id;
    }

    public static String gettime (String line){//to return date and time
        String date ="",time = "";
```

```
    //get the date
    Pattern datepattern = Pattern.compile("\\d{4}-\\d{2}-\\d{2}");
    Matcher datematcher = datepattern.matcher(line);
    if(datematcher.find()){
        date = datematcher.group();
    }

    Pattern timepattern = Pattern.compile("\\d{2}:\\d{2}:\\d{2}");
    Matcher timematcher = timepattern.matcher(line);
    if(timematcher.find()){
        time = timematcher.group();
    }
    return date +" "+time;
}

public static int timediff(String start,String end){

    int m=0,d=0,h=0,min=0,s=0;
    //the start date var
    String datestart = start.split(" ")[0];
    //split date and convert for calculate
    int monthstart = Integer.parseInt(datestart.split("-")[1]);
    int daystart = Integer.parseInt(datestart.split("-")[2]) ;
    //the end date var
    String dateend = end.split(" ")[0];
    //split to calculate
    int monthend = Integer.parseInt(dateend.split("-")[1]) ;
    int dayend = Integer.parseInt(dateend.split("-")[2]) ;

    //*****************now process the same for time
    String timestart = start.split(" ")[1];
    int hoursstart = Integer.parseInt(timestart.split(":")[0]) ;
    int minutestart = Integer.parseInt(timestart.split(":")[1]) ;
    int secondstart = Integer.parseInt(timestart.split(":")[2]) ;
```

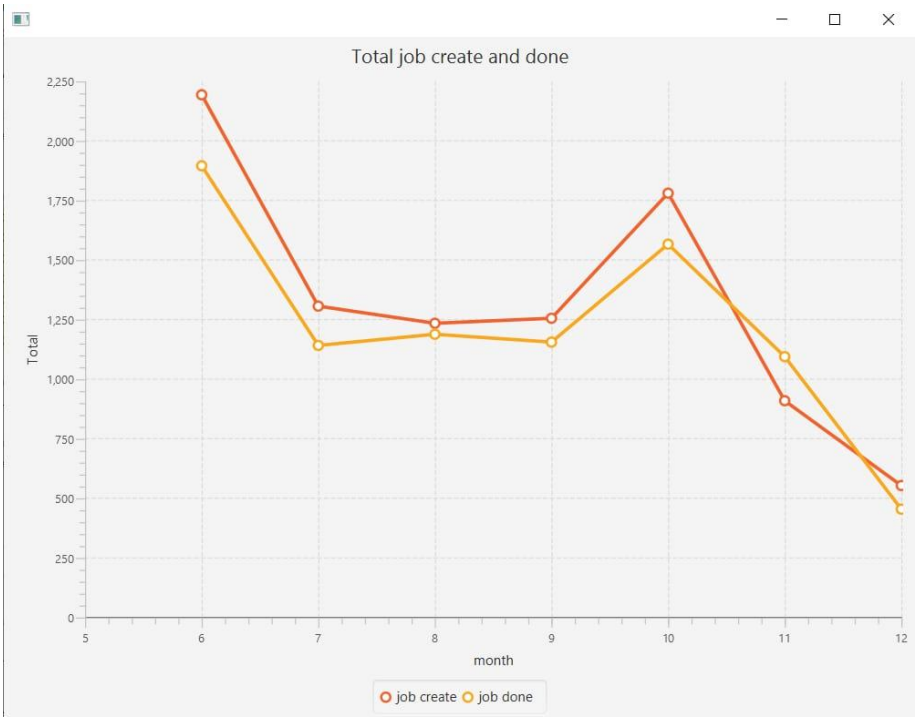
```
        String timeend = end.split(" ")[1];
        int hoursend = Integer.parseInt(timeend.split(":")[0]) ;
        int minuteend = Integer.parseInt(timeend.split(":")[1]) ;
        int secondend = Integer.parseInt(timeend.split(":")[2]) ;
        //assume each month has
30day********************************************************************************
        m = monthend-monthstart;
        if(m>0){
           dayend+=m*30;
        }
        d= dayend - daystart;
        if(d > 0){
           hoursend+=d*24;
        }
        h = hoursend-hoursstart;
        if(h>0){
           minuteend += h*60;
        }
        min = minuteend-minutestart;
        if(min> 0){
           secondend+=min*60;
        }
        s = secondend-secondstart;
        return s;
    }

    public void displaydata(){
        jobs_created_done();
        jobs_partition();
        jobs_error();
        jobs_execution_time();
    }
}
```

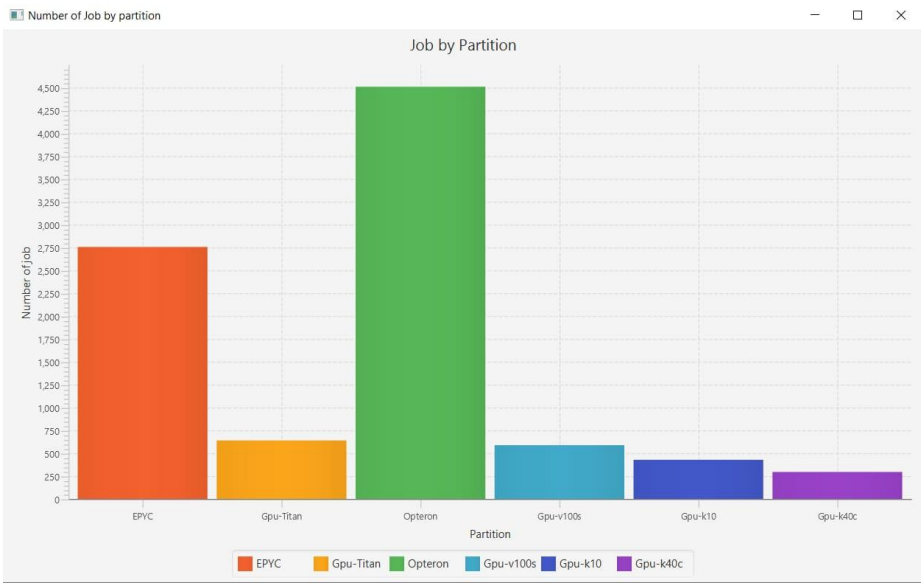
RESULTS AND FINDINGS

1) Output of the Program

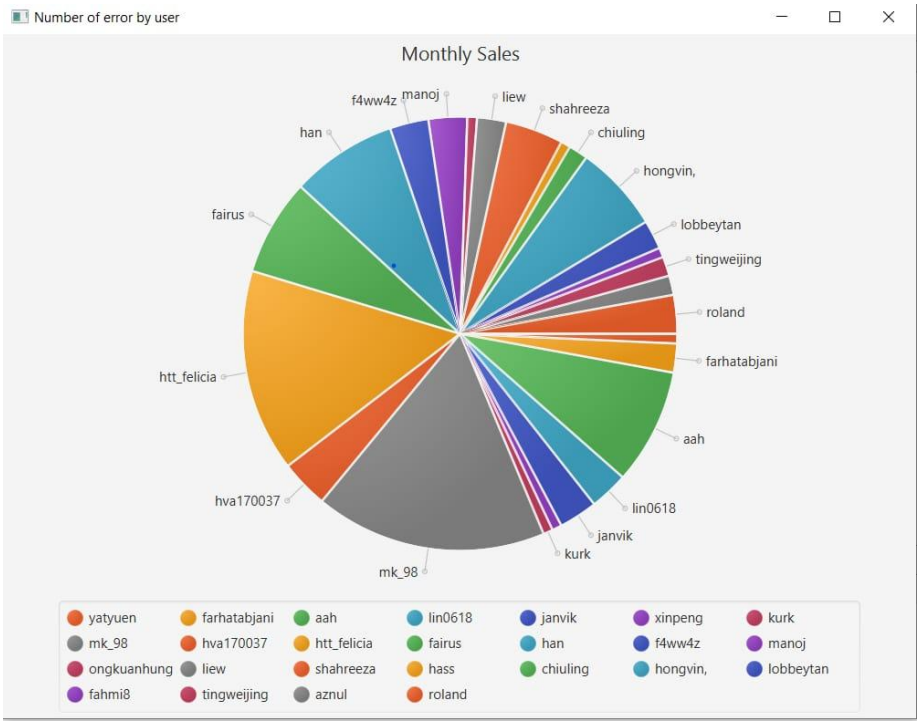
- Number of jobs created and ended per month



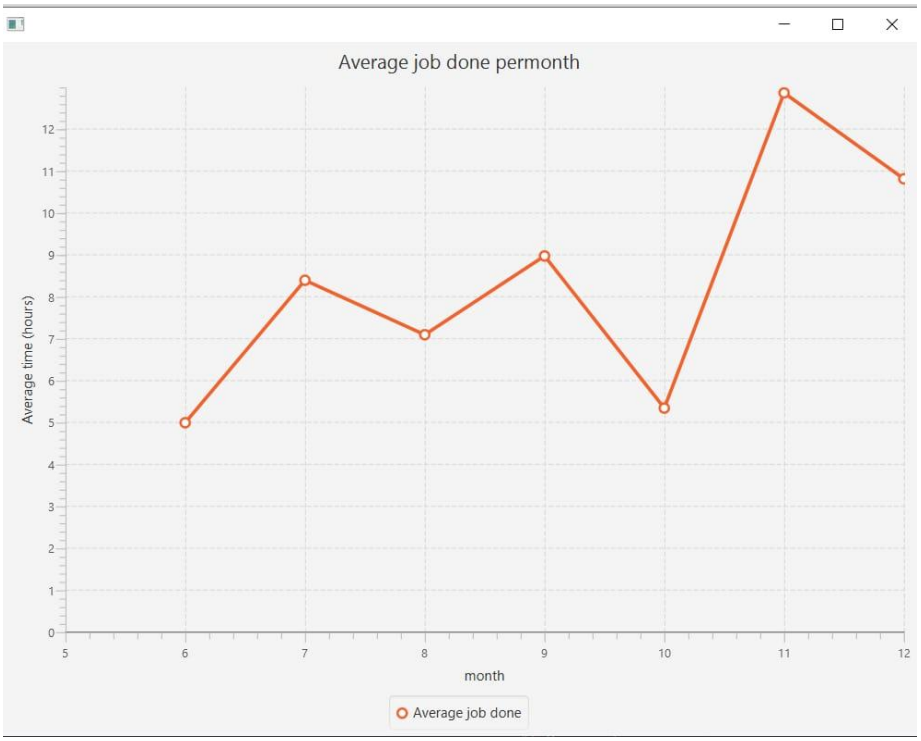
- Number of job partitions



- Number of jobs causing error and the corresponding user



- Average execution time for the jobs to be submitted to UMHPC



2) Challenges and Solutions

Challenges	Solutions
The log file had an unstructured format in a large scale making it difficult to extract the data	We organized some of the data by a given time range, for example by month, in order to allow more organization
The data in the log file had lines which were nearly similar and this made it difficult to understand the the events it described	We carefully read the data and made codes to find the keywords to the metrics we wanted to search for
The log file contained errors in some of the lines which made it difficult to extract specific contents	We made a code to search for the errors which helped us collect more accurately
Some of the job created are outside the range of the data	We skip the first 10 jobs when calculating the sum of time taken to finish a job.
The jobs by a partition are a bit confusing because the GPU and CPU are divided by many other partitions for example the GPU partition have other partitions like Titan, k40c, and v100s	We made a unique target keyword for every partition so that the program did not repeat the counting for the keywords.

3) Conclusion

We have successfully extracted data from the log file based on the four mentioned metrics using Java language despite the challenges faced during the process. The solutions to the challenges faced were also given. We provided various charts to represent the overall results from the file. The codes for the charts are in the link below.

Source code of system:
https://drive.google.com/drive/folders/1SfYu6wBebZ1xJAktQWxcJ9RkBCKSMEqd?usp=share_link