CMPE 300 - Analysis of Algorithms
Fall 2014
Programming Project

Due date: January 10, 2015 17:00

In this project you are going to implement a parallel algorithm with C/C++ language using MPI library.

MapReduce

In order to handle and analyze large volume data distributed processing is the generally used, but designing an efficient distributed system is a challenging task. There are some general distributed programming frameworks in order to simplify the implementation. One of them is the MapReduce model. This model consists of map and reduce steps. In the map step, master node takes the input, divides it and distributes to worker nodes and each worker node works on its own data independently. In the reduce step, the master node collects the answers from the workers, and combines them to generate the final result.

In this project, you are going to demonstrate a distributed data processing solution using the MapReduce programming model.

Problem Definition

You are going to analyze the sleep data of a person taken by Inertial Measurement Unit (IMU) of a smartphone. The data contains time in total seconds and acceleration values in X, Y and Z coordinates. The sampling rate for the data is 1 Hz, i.e. only 1 data is taken at a second.

The command line to execute the program should be in the format:

mpiexec -n numProcessors programName fileName startOfSleepTime

Example for 4 processors: mpiexec -n 4 sleepAnalysis sleepData.txt 2:17

When your program starts, the master node should load the data, divide and distribute it among the worker processors. The columns should not be separated, i.e. a data should be sent in [time,accX,accY,accZ] format to a worker processor.

The worker nodes should analyze the change in acceleration data and separate it to three classes:

- Deep sleep: Acceleration change <= 0.008 corresponds to deep sleep.
- Light sleep: Acceleration change > 0.008 and < 0.03 corresponds to light sleep.
- Awake: Acceleration change >= 0.03 corresponds to awaking from sleep or being awake.

If any of the accX, accY and accZ values have a change more than 0.008 than the person should be in light sleep (>= 0.03 for being awake.)

Acceleration change is calculated by: $|acc[i+1]-acc[i]|$. This should be calculated by **worker** nodes.

A movement is counted if the person enters to **light sleep from deep sleep or awake from light or deep sleep**.

Example:

From 3:20-3:40 the person is in deep sleep, then at 3:41 the person moves and is in light sleep. Number of movements in the data is increased by one. Between 3:41-3:45 the person keeps moving (in light sleep range), this data is assigned as light sleep however number of movements is **NOT** increased. Afterwards, the person enters deep sleep again, and then awakes at 4:20, then number of movements is increased by one, and awake time (in seconds) is noted. The person is awake from 4:20-4:24 but goes back to sleep at 4:25. The number of moves between 4:20-4:24 is **NOT** counted.

For simplicity, you can assume that the person would go into deep sleep from light sleep or into awake from light or deep sleep in less than 8 minutes, i.e. any two data in light sleep or awake should be larger than 8 minute range.

The number of movements increases if:

- If you were in deep sleep and now in light sleep, and there is 8 minute difference between 2 light sleep times,i.e. currentTime > light[x]+8min

- Else if you were in deep or light sleep and now you are awaken, and there is 8 minute difference between 2 awaken times AND if there is 8 minute difference between current time and the last light sleep time, i.e. currentTime > awake[y]+8min AND currentTime > light[x]+8min

The awaken times are noted if you are awaken and currentTime > awake[x]+8min. (Don't look at light sleep times for this)

For deep sleep time you can either disregard the 8 minute rule or use it as follows:

Deep sleep time counter is increased if:

- If you were in light sleep and you are in deep sleep now(look at the changes between accelerations rule) and current time > light[x]+8min

- Else if you were awaken and you are in deep sleep now(look at the changes between accelerations rule) and current time > awake[x]+8min

- Else if you are not in light sleep or awaken stage

Check the 8 minute rule in each processor and also in the master node in the end. Note that x is not time! (You need to find what it is)

Therefore, the move in the beginning of the sleep data (when the person hasn't started sleep yet) is **NOT** counted.

Each worker should count total number of movements, total deep sleep time and find the times for awaking in their data and send this information to master node. Master node should collect this information and find the total number of movements in a night, total deep sleep time, and find the actual times for awaking. (The master node should calculate the hour:time format)

Actual time for awaking is found by:

Total Minute: Sleep start minute + $\lfloor$ (awake second % 3600) /60 $\rfloor$
Awake at Minute: TotalMinute %60
Awake at Hour: (Sleep start hour+ $\lfloor$ (awake second/3600) $\rfloor$ +$\lfloor$ TotalMinute/60 $\rfloor$) %12

Example:

Total Minute: $(17 + \lfloor(23290\%3600)/60\rfloor) = 45$
Awake at Minute: $45\%60 = 45$
Awake at Hour: $(2 + \lfloor 23290/3600\rfloor + \lfloor 45/60\rfloor)\%12 = 8$

Awake at: 8:45

Your output should be in the following format, and you should output your file to *output217.txt* file, that is your output file name should contain sleep start time:

Not considering 8 minute rule:

```
Number of moves:
16
Total deep sleep:
7:8
Awake at:
2:17
7:15
8:32
9:1
9:27
```

Considering 8 minute rule:

```
Number of moves:
16
Total deep sleep:
5:5
Awake at:
2:17
7:15
8:32
9:1
9:27
```

NOTE: It is very important that you follow this structure in your output as your output will be judged by a program. You should output the beginning of the awake at time (e.g. if a person is awake from 2:17-2:19, you should output 2:17). Take floor for outputting times. (e.g. 5:5:20 is 5:5)

Important Points

- The acceleration data set for the sleep analysis is on the website. The output for the dataset are on the website.
- The master node is supposed to distribute all of the data and control the process. In your design, it will only collect and combine the workers answers. It is allowed to process ONLY 8 minute differences between number of movements and deep sleep count in the end.
- Try to send as less data as possible between master and workers. For example, sending all measurements to master processor and letting it do all the calculations is not appropriate.
- Be sure to test your design with different number of workers processors.
- If you choose C++ for programming, you still need to use C MPI bindings since C++ MPI bindings depreciated.
- The project will be done individually, not as a group.
- See Programming Project Documentation on the website for general information (submission process, documentation, etc.) about programming projects.

Submission

The following guidelines are taken from Prof. Tunga Güngör's website:
$http://www.cmpe.boun.edu.tr/\sim gungort/$

- The program source code should be written in a good style (indentation, etc.) and be heavily commented.
- A design and implementation document which clearly explains the project must be prepared. Follow the structure given in the Programming Project Documentation. The document will be an important part of the project. Your project will not be graded if you do not prepare a

document. The suggested size of the document is about 10 pages, though it depends on the particular project. **NOTE:** You can write 3-5 pages, but make sure it explains your algorithm well. (Don't just copy paste your code, explain what your functions do and how they work)

- The project materials must be sent via e-mail as an attachment. The subject field of the e-mail must be in the format cmpexxxpy, where xxx is the course number, and y is the project number. (The subject field is used for automatic placement of e-mails into the appropriate folder. Therefore, not following this format for the subject may cause your e-mail to be ignored.)

- All project materials (program source code, program document, input and output files, test files, graphics files, etc. - all those stated in the project announcement document) must be compressed into a single zip file named as namesurname.zip, where the file name identifies you. If the project is worked on as a group, then the file name can be the name of any one of the group members (do not write the names of all members). The files within this zip file must be given legitimate names (e.g. the source code can have a name related to the project topic and an extension of the language used, the document can be named as document.xxx with a suitable extension, the input/output/test files can be named as input1.xxx, input2.xxx, output.xxx, test1.xxx, test2.xxx, etc., and so on). However, if any specific names are stated in the project announcement document, then these names must be used.

- The content of the e-mail, other than the subject field and the attachment, must be empty.

- All project materials mentioned in the previous item must also be submitted in hard-copy. Place the materials in a single package in the following order: A cover sheet (including course code and name, your name or names of group members, a project title), program document, program source code, any additional materials stated in the project announcement document.