**Developing Soft and Parallel Programming Skills Using Project-Based Learning**

Spring-2019 **JupitorInk**

Nhi Ong, Alejandro Herbener, Alec Buchinski, Johnathan Moore

**Table of Contents**

1. Project Description……………………………………………………………………2
   1. Purpose of the Project…………………………………………………………….2
2. Introduction to the Team………………………………………………………………2

2.1 Members Roles and Assignments…………………...……………………………2

1. Planning and Scheduling……………………………………………………………2-4

3.1 Role as Coordinator……………………………………………………………….3

3.2 Additional Tasks……...………………………………………………………...3-4

1. Parallel Programming Skills……………………………………………………….4-12

5.1 Foundation………………………………………………………………………4-5

5.2 Parallel Programming Basics Report...….......…………………………………5-12

1. Appendix……………………………………………………………………………..15
2. **Project Description**
   1. **Purpose of the Project**

The purpose of this project is to 1) become familiar with unified, team-based project management tools and communication applications in order to develop soft skills, 2) identify and analyze the basics of parallel computing architectures and programming using OpenMP and C language 3) MapReduce

1. **Introduction to the Team**
   1. **Members’ Roles and Assignments**

Coordinator: Jonathon Jack Moore

He is the coordinator for this assignment. In addition to his responsibilities as a coordinator, he was also responsible for tasks one, two, and five. He created a new assignment task chart, sent an invitation to our Slack workspace to the TA, created a new GitHub repository and projects page, and edited our video and uploaded it to the YouTube channel.

Team Member: Alec Buchinski

This team member is responsible for task three. Alec answered the questions that for the Parallel Programming Foundation and completed the programming portion of this task. He also took screenshots and created a written report of his findings.

Team member: Nhi Ong

This team member is responsible for task five. I created this report after compiling each member’s work from their corresponding tasks.

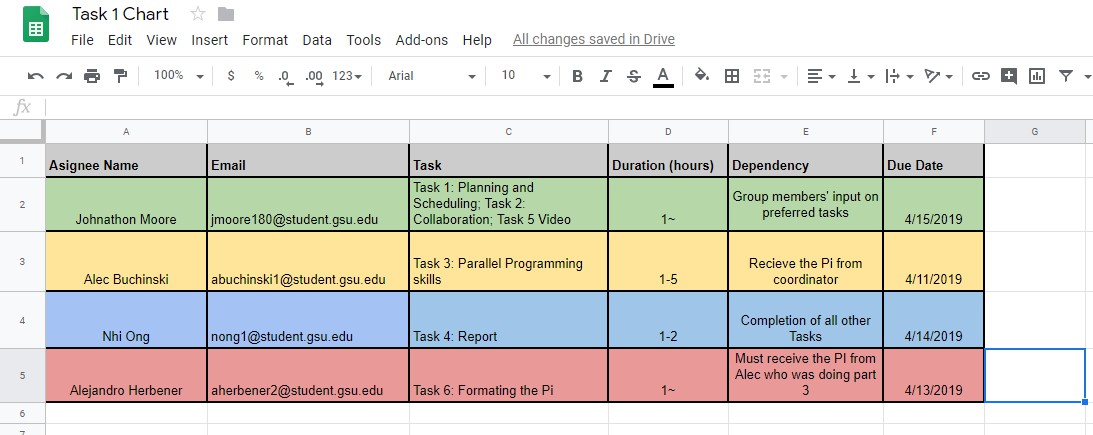
Team Member: Alejandro Herbener

This team member is responsible for task six. He formatted the PI in preparation for its return. He will hand it to the coordinator who will complete the final hand off.

1. **Planning and Scheduling**

**3.1 Role as Coordinator**

As the coordinator, Johnathon was responsible for assigning us with our tasks, updating our assignment table, creating a new repository for the assignment, and adding the TA to our Slack workspace. He first contacted us to discuss our preferences for our roles in this assignment, which we each quickly selected and promptly began working on our tasks. He updated the assignment task table soon thereafter and notified us of our responsibilities, dependencies, and due date. (see Figure 1.1)



*Figure 1.1 – Task One, Assignment Chart*

**3.2 Additional Tasks**

Johnathon was responsible for creating a new repository for this assignment with a readme page. He added a projects page which details our tasks and allows for any member to update their status on their role. (see Figure 1.2) In addition to this, he also edited our video and uploaded it to YouTube. (see Appendix A, for link to video)

A screenshot of a computer

Description automatically generated

*Figure 1.2 – Task Two, GitHub Projects Page Screenshot*

1. **Parallel Programming Skills**

**5.1 Foundation**

*What are the basic steps in building a parallel program?*

First, one must identify the tasks in a program that can run simultaneously. A task can run simultaneously if it has no dependencies or communications between tasks. Next, implementation techniques such as master/worker are used.

*What is MapReduce?*

MapReduce is a programming model that programmers often use when creating large sets of data on a cluster in a parallel and distributed algorithm.

*Why MapReduce?*

MapReduce can be useful because it allows for processing of large amounts of raw data while keeping the details of parallelization, data distribution, load balancing and fault tolerance hidden.

*Show an example for MapReduce?*

MapReduce can be used for a program that requires a term vector which summarizes the most important words that occur in a document stored in <word,frequency> pairs.

*Explain in your own words how MapReduce is executed.*

The library shards the files into a certain amount of pieces and begins creating many copies of the program on a number of machines. There is one copy designated master which assigns work to the workers. Workers are assigned a map task or a reduce task. The master is also responsible for forwarding locations of buffered pairs to the reduce workers. When these workers receive this information from the master, it reads the buffered data using a remote procedure call and sorts through the intermediate data and passes the key to the corresponding set of values in the Reduce function. When all tasks are finished, the master wakes up the program.

*List and describe the three examples that are expressed as MapReduce computations.*

Distributed Grep: The map function emits a line if it matches a given pattern. The reduce function is an identity function that just copies the supplied intermediate data to the output.

Count of URL Access Frequency: The map function processes logs of web page requests and outputs <URL, 1>. The reduce function adds together all values for the same URL and emits a <URL, total count> pair.

Reverse Web-Link Graph: The map function outputs <target, source> pairs for each link to a target URL found in a page named "source". The reduce function concatenates the list of all source URLs associated with a given target URL and emits the pair: <target, list(source)>.

*When do we use OpenMP, MPI, and MapReduce and why?*

Because OpenMP is a directive based library, it is most desirable to use it when you want to include shared memory parallelism in your program. It makes use of threads. MPI on the other hand, is usually used in parallel scientific applications. It can run parallel code over many machines at once. Some programmers practice what is called “hybrid programming” and mix both MPI and OpenMP together, MapReduce is a technique used on large data where the data is first distributed and then reduced. It is also used for scientific applications but it does not have the best performance.

*In your own words, explain what a Drug Design and DNA problem is in no more than 150 words.*

A Drug Design and DNA problem usually has to do with the creation and design of medicine. Our bodies rely on proteins made by DNA in our body. The shape of the protein will decide what kind of function it performs in the body. When scientists design new drugs, they find what are called ligands to change a proteins shape. If there is a disease that requires a medicine, first the protein associated with the disease must be found and modeled through molecular moddeling computations. They will then a set of computations is completed which tests collections of liqands for orientations that bind to the protein in satisfactory manners. Each of these computations are scored and the highest scoring sets will be tested further for drug development.

**5.2 Parallel Programming Basics**

* 1. Discussion Questions

Measure Run-Time

\*\*\* Real is the wall clock time, from start to finish of the call

\*\*\* User is the amount of CPU time spent outside the kernel within the process

\*\*\* Sys is the amount of CPU time spent in the kernel within the progress

\*\*\* User+Sys is the total CPU time the process used. If multi-thread, might exceed t he wall clock time.

Times for each implementation:

time -p ./dd\_serial : 122.49

A screenshot of a computer

Description automatically generated

time -p ./dd\_omp 1: Real: 0.02, User: 0.01

A screenshot of a computer

Description automatically generated

time -p ./dd\_threads 1: Real: 0.02, User: 0.01

A screenshot of a computer

Description automatically generated

time -p ./dd\_omp 2: Real: 0.02, User: 0.02, Sys: 0.02A screenshot of a computer screen

Description automatically generated

time -p ./dd\_threads 2: Real: 0.02, User: 0.01, Sys: 0.01

A screenshot of a computer

Description automatically generated

time -p ./dd\_omp 3: Real: 0.04, User: 0.13, Sys: 0.00

A screenshot of a computer screen

Description automatically generated

time -p ./dd\_threads 3: Real: 0.04, User: 0.08, Sys: 0.03

A screenshot of a computer

Description automatically generated

time -p ./dd\_omp 4: Real: 0.21, User: 0.56, Sys: 0.01

A screenshot of a computer screen

Description automatically generated

time -p ./dd\_threads 4: Real: 0.16, User: 0.56, Sys: 0.01

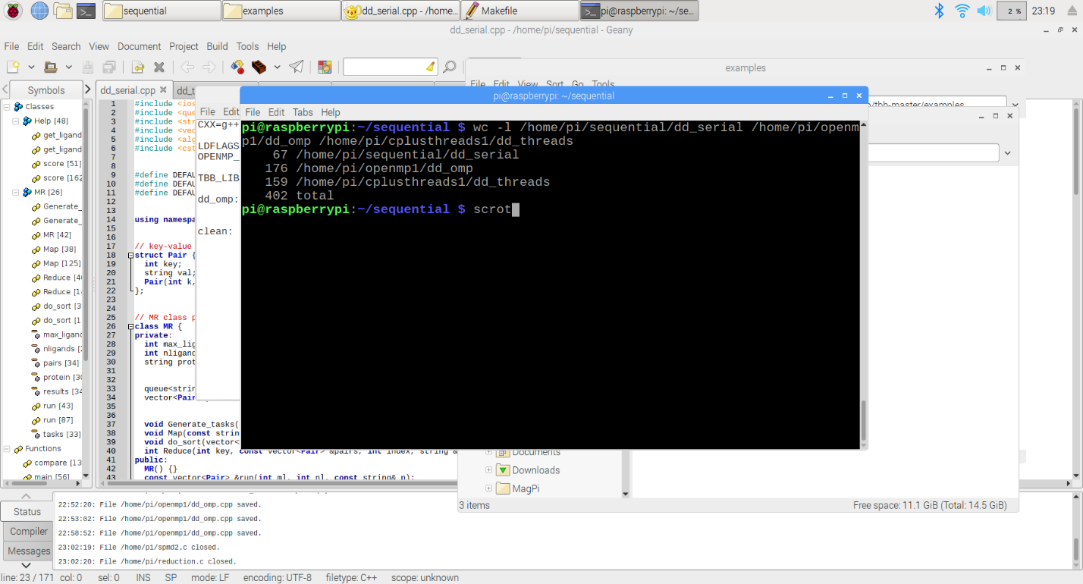
A screenshot of a computer

Description automatically generated

1) *Which approach is the fastest?*

Using the total of User and Sys time, dd\_threads is the fastest. The user time increases less than the dd\_omp user time, even with a longer sys time.

2) *Determine the number of lines in each file using wc-1. How does the C++11 implementation compare to the OpenMP implementations?*



A screenshot of a computer

Description automatically generated

3) *Increase the number of threads to 5. What is the run time for each?*

dd\_omp 5: Real: 0.95, User: 2.6, Sys: 0.02

A screenshot of a computer screen

Description automatically generated

dd\_threads 5: Real: 0.68, User: 2.58, Sys: 0.01

A screenshot of a computer

Description automatically generated

The run times using User+Sys for dd\_omp is 2.62, and for dd\_threads is 2.59.

Using real, the times are 0.95 for dd\_omp, and 0.68 for dd\_threads.

4) *Increase the maximum ligand length to 7, and rerun each program. What is the run time for each?*

The default maximum ligand length is already 7 in each of the programs, so there are no reported changes.

1. **Appendix**
2. **Link to Slack, GitHub, YouTube Channel, YouTube Video**

Slack:

<https://csc-3210.slack.com>

GitHub:

<https://github.com/jupitorink>

GitHub Assignment Five Repository:

<https://github.com/jupitorink/Assignment5>

YouTube Channel:

<https://www.youtube.com/channel/UChTgBqHzG9G-Mvv-q7uK96g>

YouTube Video:

***INSERT YOUTUBE VIDEO HERE***

1. **Task One Screenshot**

Task Assignment Table:

<https://github.com/jupitorink/Assignment5/blob/master/Task%201%20Schedule.jpg>

1. **Task Two Screenshot**

GitHub Projects Page:

<https://github.com/jupitorink/Assignment5/blob/master/projectspage.png>

1. **Task Three Parallel Programming Report, Screenshots**

Parallel Programming Report, Screenshots

<https://github.com/jupitorink/Assignment5/blob/master/FiveThreads_dd_omp.png>

<https://github.com/jupitorink/Assignment5/blob/master/FiveThreads_dd_threads.png>

<https://github.com/jupitorink/Assignment5/blob/master/FourThreads_dd_omp.png>

<https://github.com/jupitorink/Assignment5/blob/master/FourThreads_dd_threads.png>

<https://github.com/jupitorink/Assignment5/blob/master/OneThread_dd_omp.png>

<https://github.com/jupitorink/Assignment5/blob/master/OneThread_dd_threads.png>

<https://github.com/jupitorink/Assignment5/blob/master/Question2_WordCount.png>

<https://github.com/jupitorink/Assignment5/blob/master/RunTime_dd_serial.png>

<https://github.com/jupitorink/Assignment5/blob/master/ThreeThreads_dd_omp.png>

<https://github.com/jupitorink/Assignment5/blob/master/ThreeThreads_dd_threads.png>

<https://github.com/jupitorink/Assignment5/blob/master/TwoThreads_dd_omp.png>

<https://github.com/jupitorink/Assignment5/blob/master/TwoThreads_dd_threads.png>