**Developing Soft and Parallel Programming Skills**

**Using Project – Based Learning**

CSC 3210 Spring – 2020

8 – Bit

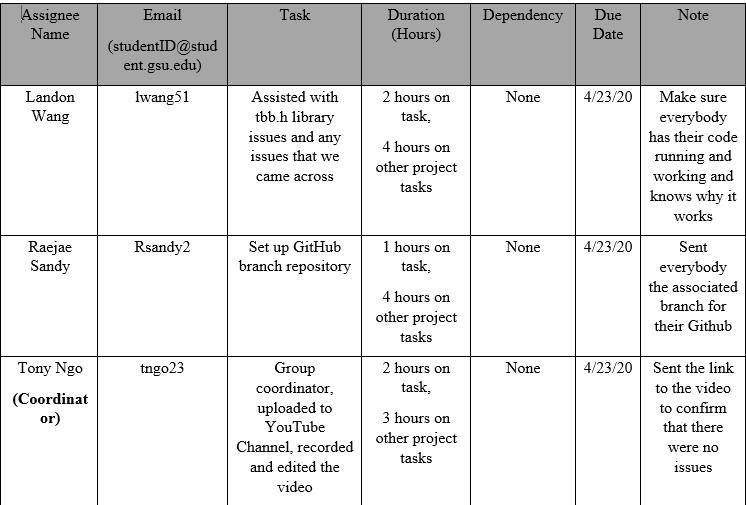
Landon Wang

Raejae Sandy

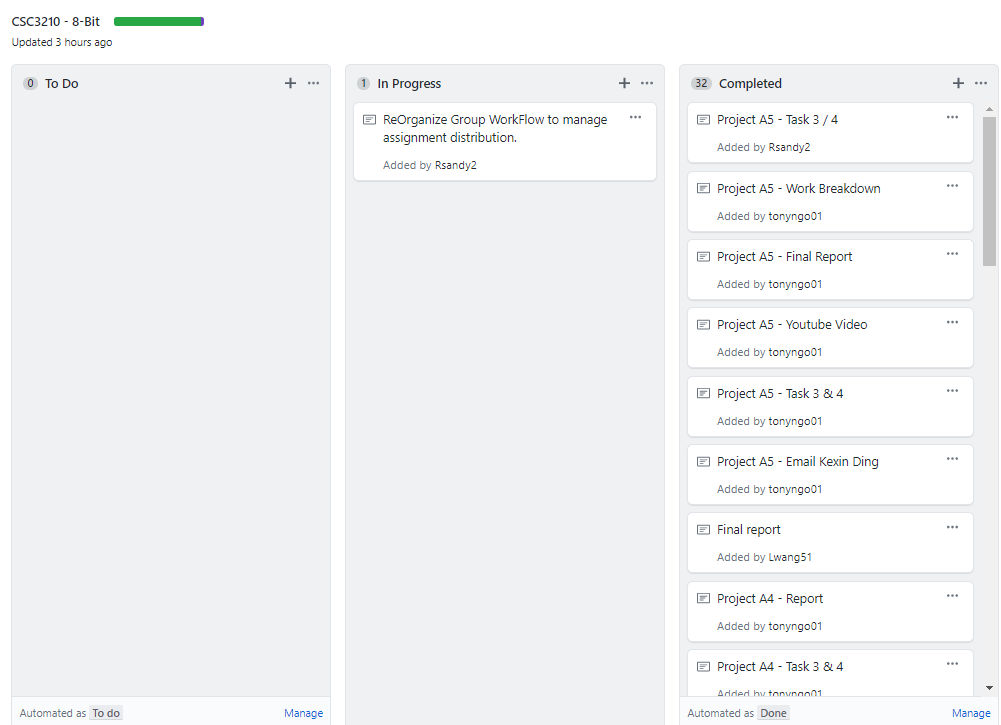
Tony Ngo

**Planning and Scheduling**

**Work Breakdown Structure**

****

**GitHub Project Page**

****

**Parallel Programming Skills and Basics**

**Parallel Programming Skills**

By: Landon Wang

**Part 1 (76p):**

* (56p) Read this paper “Introduction to Parallel Programming and MapReduce” and

answer the following questions:

* + (15p) What are the basic steps (show all steps) in building a parallel program? Show at least one example.

1. **Identify the set of tasks that can be executed concurrently and the partitions of data that can be processed concurrently**
   * **Fibonacci Function:**

**cannot be executed using parallelism because of dependency (each value is dependent on previous calculated value)**

1. **If the program and data can be executed using parallelism, develop a parallel solution**
   * **Master-Worker Solution**

**Calculating PI, where the master sends worker tasks, and workers return the results (how many points lies both in the circle and the square) to the master so the master can calculate PI**

* + **Fork-Join solution**
  + **Task Parallelism**
  + **Data Parallelism**
  + (5p) What is MapReduce?

**MapReduce is developed within Google as a tool for processing large amount of data (crawled documents or web request logs).**

**It is an abstraction that allows Google engineers to hid the details of parallelism, data distribution, load balancing, and fault tolerance while preforming calculations.**

* + (10p) What is map and what is reduce?

**The Map (developed by the user of the MapReduce library) takes an input pair and returns a set of key/value pairs. The library will then group all values associated with the same key and call the reduce function. The reduce (also developed by the user of the MapReduce library) function then merge the values (associated with the same key) together to form a smaller set of values.**

* + (5p) Why MapReduce?

**MapReduce helps reduce the amount of time needed to process large amount of data (by shrinking the data) in a reasonable time.**

* + (5p) Show an example for MapReduce.

**One of the most famous examples of MapReduce is counting how many times a set of words occurred in a number of documents. For example, say we are searching for how many times the words “map and “reduce” occurred in one hundred documents and we have 5 machines to do the job for us. We can have each machine take in 20 documents and look through the documents for how many times those words occur. The machines will perform calculations without communicating to each other in the map function. Furthermore, the words will be the key, and the number of times they appeared will be the value. Say machine A counted “map” 15 times and “reduce” 13 times in the first 20 documents; machine B counted “map” 11 times and “reduce” 9 times in the next 20 documents; machine C counted “map” 5 times and “reduce” 12 times in the next 20 documents; machine D counted “map” 1 times and “reduce” 3 times in the next 20 documents; machine E counted “map” 10 times and “reduce” 13 times in the last 20 documents. Next, these 5 machines will give their results to a machine (it can be one of the 5 machines or another machines). The machines receiving the results will add up all the values associated with the same key, and return the final result (42 count for “map and 50 count for “reduce”).**

* + (10p) Explain in your own words how MapReduce model is executed?

**There are four phases of MapReduce. First, we have Input Splits, where the input is being split up. Next, we have Mapping, where the data in each split is processed through a map function that produced key/value output. Then, we have Shuffling, which combines the results from the previous phase. Last, we have the Reducing phase, which merges the values from the previous phase to produce a single output.**

* + (6p) List and describe three examples that are expressed as MapReduce

computations.

**Count of URL Access Frequency:**

**Mapping function processes the web page request logs and returns <URL, 1>, and the reduce function merges all values of the same URL and returns <URL, total count> pair.**

**Reverse Web-Link Graph:**

**Mapping function return <target, source> pairs for each link to target URL found of “source” page, and the reduce function merges the list of all source URLs associated with given target URL and returns <target, list(source)>.**

**Term-Vector per Host:**

**Mapping function returns a <hostname, term vector> for each input document (hostname is the document’s URL), and the reduce function merges together the term values associated with the same hostname. It then discards infrequent terms and returns a final <hostname, term vector> pair.**

* (6p) When do we use OpenMP, MPI and, MapReduce (Hadoop), and why?
  + **OpenMP**

**We can use this model when we want to use shared memory parallelism in our code. It is an efficient directive-based library. Although it is neat and powerful, we must be careful when using it to avoid performance and correctness errors.**

* + **MPI**

**This is a distributed memory parallel model that is usually used in scientific programs, because it is tightly synchronous code and well balanced. This can also be used to develop pretty much any parallel code and can also be mixed with OpenMP (hybrid programming).**

* + **MapReduce (Hadoop)**

**We can use this model when we are dealing with large amount of data, and we want to apply some operation for each data element. The data is typically distributed using HDFS (Hadoop Distribution File System), a distributed file system designed to run on commodity hardware. The results can than eb reduced using the reduce function. When we have terabytes of data to ETL (Extract, transform, and load), it more efficient if we use this model than MPI, because of its fault tolerance feature (one of its key features).**

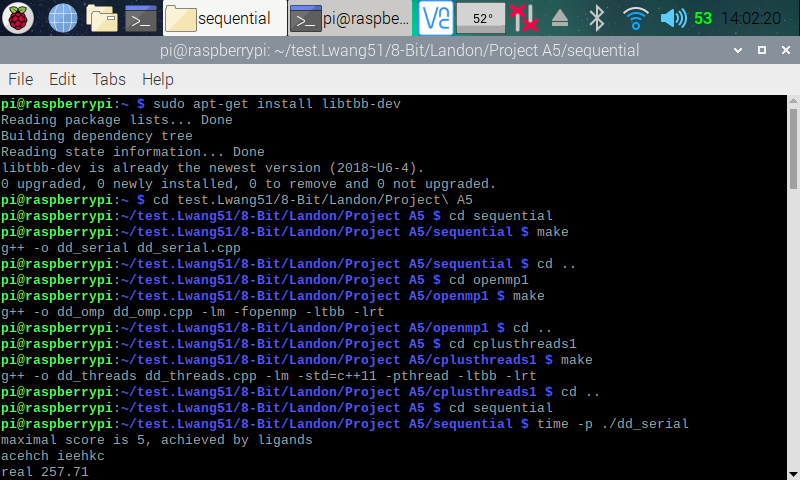
* (14p) In your own words, explain what a Drug Design and DNA problem is in no more than 150 words.

**After identifying a disease of interest’s protein, its three-dimensional structure can be found through experiments or molecular modeling computation. Then, a collection of ligands is tested against the protein, and computation is done to see how well the ligand binds with the protein in useful ways (tying up biologically active regions). The ligands will have a score based on their binding properties, and the ligands with high scores will be identified as good drug candidates.**

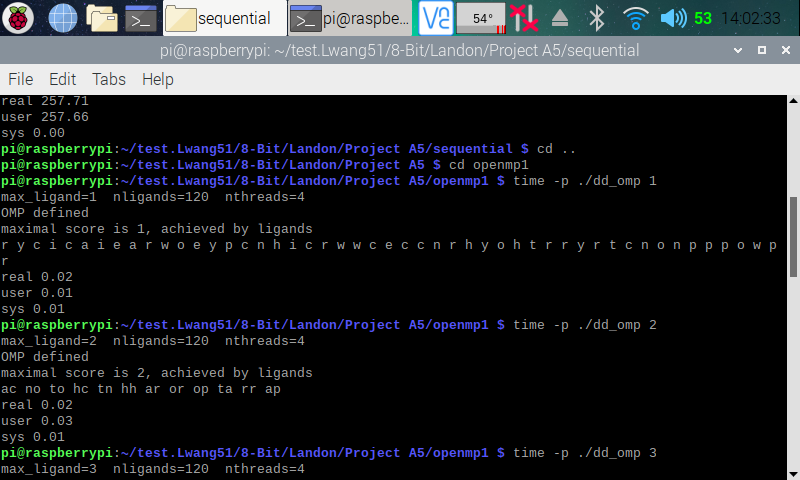
**Parallel Programming Basics**

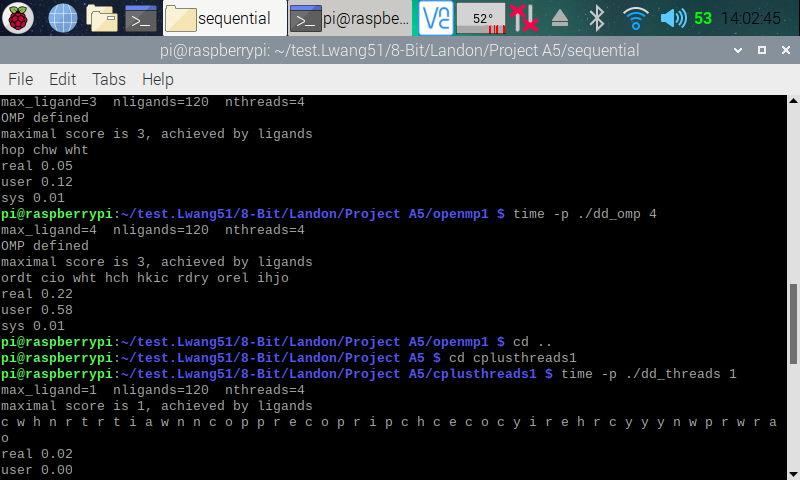
By: Landon Wang

**Part 1: Sequential, OpenMP, and C++ 11 Threads Solutions and Measuring Run Time**

****

Here, (in the screenshot above), I ran into an error while compiling the OpenMP and C++ 11 Threads program. The error stated “*tbb/concurrent\_vector.h: no such file or directory”*. After doing some research and looking through my teammate’s explanation, I found out that I had to install the TBB library onto my Raspberry Pi. Using the commands “*sudo apt-get install libtbb-dev”*, I installed the library then compiled the OpenMP and C++ 11 Threads solution programs.







Here (in the three screenshots above), I used the command *time -p ./ProgramName* to find the runtime of each program. I also tested the runtime for the two parallel solution programs with 2, 3, and 4 threads. We can see that the runtime increase as we increase the amount of threads.

**Part 2: Discussion Questions**

|  |  |
| --- | --- |
| Implementation | Times (s) |
| dd\_serial | 257.71 |
| dd\_omp | 0.02 |
| dd\_threads | 0.02 |

|  |  |  |  |
| --- | --- | --- | --- |
| Implementation | Time (s) 2 Threads | Time (s) 3 Threads | Time (s) 4 Threads |
| dd\_omp | 0.02 | 0.05 | 0.22 |
| dd\_threads | 0.02 | 0.04 | 0.19 |

1. **Which approach is the fastest?**

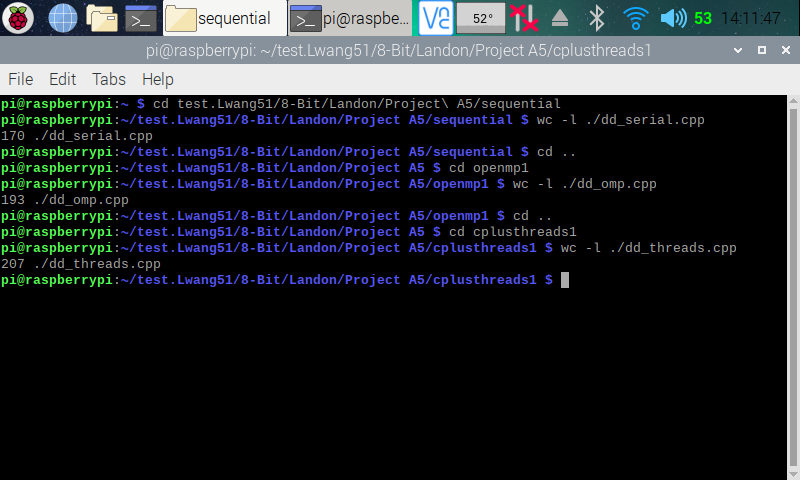
Dd\_threads implementation was faster than the other two solutions

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

Sequential Program: 170 lines

OpenMP Program: 193 lines

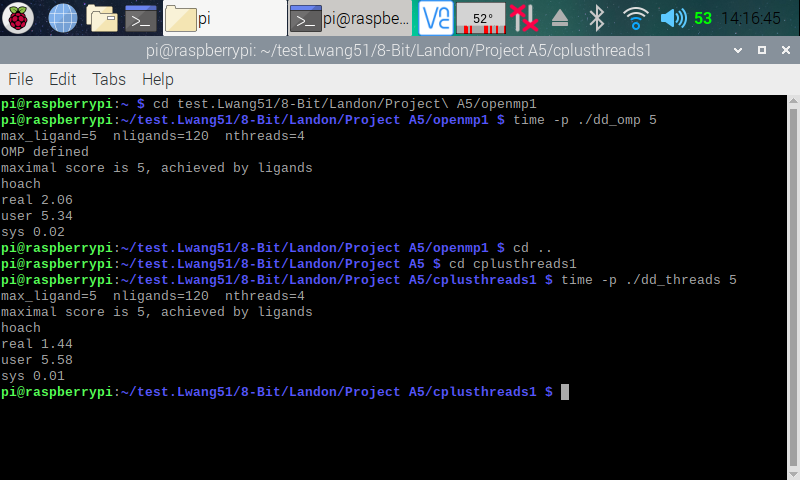
C++ 11 Threads Program: 207 lines



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

OpenMP Program: 2.06 seconds

C++ 11 Threads Program: 1.44 seconds

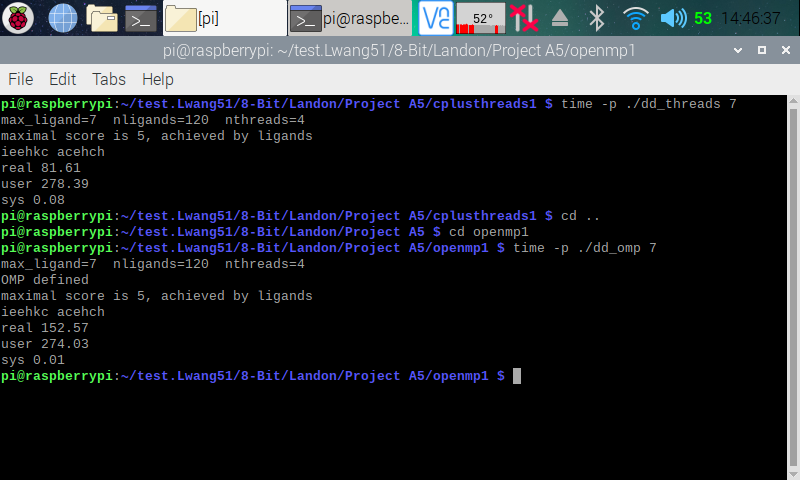


1. **Increase the maximum ligand length to 7, and rerun each program. What is the run**

**time for each?**

OpenMP Program: 152.57 seconds

C++ 11 Threads Program: 081.61 seconds



**Parallel Programming Skills**

By: Raejae Sandy

**(15p) What are the basic steps (show all steps) in building a parallel program? Show**

**at least one example.**

When it comes to building a parallel program, one should first understand what can be broken down to parallel. Some functions require previous input to finish while others can work in conjunction with one another. The example give was the Master worker workload where the dataset is split into even sequence amongst the workers, and then processed. Finding PI is an example of how this can be enforced.

**o (5p) What is MapReduce?**

MapReduce allows us to process large datasets with parallel algorithms by distributing workloads.

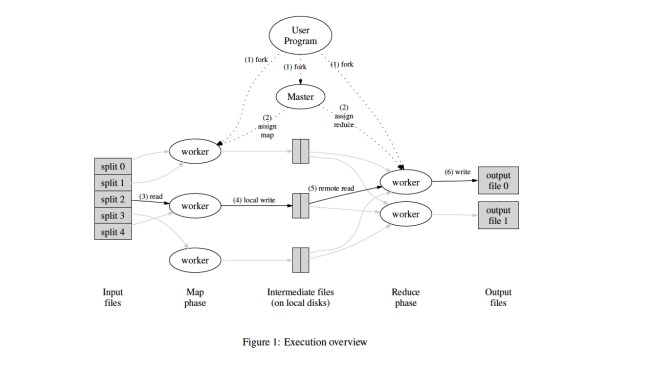
**o (10p) What is map and what is reduce?**

Similar to HashMaps Map is a function the generates key value pairs as well as intermediate value pairs which afterwards Reduce merges these pairs.

**o (5p) Why MapReduce?**

Operation Management. MapReduce gives the benefit of system management when checking load and has a developed tolerance to manage stress.

**o (5p) Show an example for MapReduce.**

****

**o (10p) Explain in your own words how MapReduce model is executed?**

**o (6p) List and describe three examples that are expressed as MapReduce**

**computations.**

**Distributed Grep :** Map function will return a line if a given pattern is matched. Reduce function copies intermediate data to output.

**Count of URL Access Frequency :** Map function processes logs of web pages request and outputs. The reduce function then adds all together for the same URL and returns a pair. If I’m correct Google actually uses this for data representation.

**Term-Vector per Host :** Term vector summarize the most important words in a document in pairs of frequency tied to word, and then emits a pair for each input. Reduce function passes all vectors for a host and then adds together throwing away infrequent terms.

**- (6p) When do we use OpenMP, MPI and, MapReduce (Hadoop), and why?**

OpenMP: We use OpenMP when we want to introduce shared memory parallelism in our code. It is very useful in taking singular loads and distributing them.

MPI: Message Passing Interface is useful for parallel code that runs over multiple machines. It is useful for parallel implementation for scientific applications

MapReduce: We use MapReduce to reduce the amount of data used in operations over large sets of data.

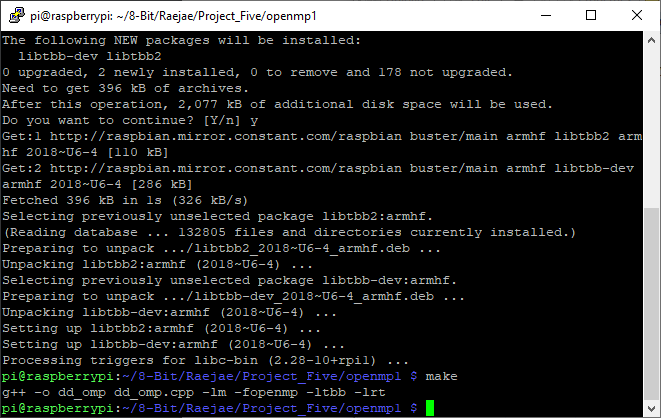
**- (14p) In your own words, explain what a Drug Design and DNA problem is in no more than**

**150 words.**

The process for this involves an object called ligands. These are needed to try for proteins. We will give a score to each of these and then determine the highest bond ratio. We will use parallel programming to hash the ligands and distribute the association process

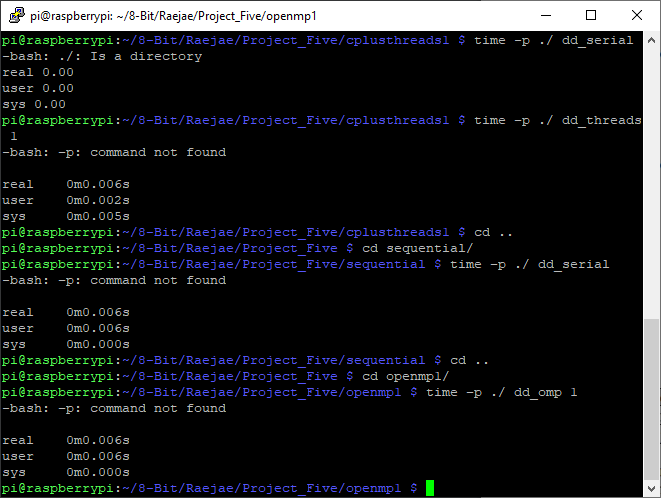
**Parallel Programming Basics**

By: Raejae Sandy

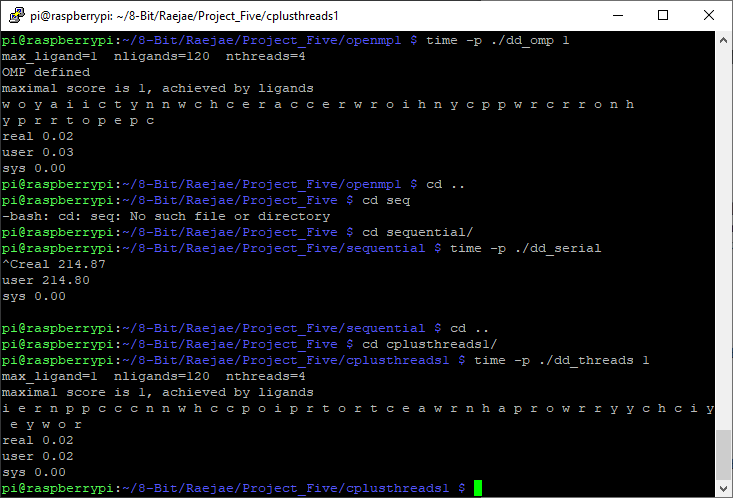


If my pictures look different. This stage of the project I decided to get a little investigative (I know late) I was curious on what SSH really was, and while playing around with it I found I could connect to PI from PC, and edit files the same, and then I could use WinSCP to transfer files. It was really helpful as transferring the files in such a manner initially seemed rather daunting. However, this moment here was a moment that finally made everything sail smooth again. After compiling sequential properly, I was expecting the same for OpenMp and Threads; however, I received bugs on both. I did a little searching and found out that I basically didn’t have to library installed so after installing it to the PI I could compile properly! The above code it is installing LITBTBB2

Everything went smoothly shortly after. While I followed instructions



Well. . . I lied when I said smooth sailing. The first mistake I made was obviously running the command for a directory outside of the one I was in. While that was questionable, I did get a little better.



|  |  |
| --- | --- |
| **Implementation** | **Time(s)** |
| dd\_serial | 214.87 |
| dd\_omp | 0.02 |
| dd\_threads | 0.02 |

For serial it never fully went through however I tried it a couple of times, and it always seemed to land on 215. I’m guessing it’s slower since it doesn’t engage parallel processing.

|  |  |  |  |
| --- | --- | --- | --- |
| **Implementation** | **Time(s) 2 Threads** | **Time(s) 3 Threads** | **Time(s) 4 Threads** |
| dd\_omp | 0.02 | 0.04 | 0.35 |
| dd\_threads | 0.02 | 0.05 | 0.16 |

2.3 Discussion Questions

1. Which approach is the fastest?

Using the thread solutions was the fastest.

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

Sequential Lines: 170

OpenMP Lines: 192

C++11 Lines: 207

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

OpenMP Real Time : 2.02

C++ Real Time: 1.47

4. Increase the maximum ligand length to 7, and rerun each program. What is the run

time for each?

OpenMP Time capped out at 127.16

C++11 Time capped out at 81.79

**Parallel Programming Skills**

By: Tony Ngo

**Task 3A**

1. **What are the basic steps (show all steps) in building a parallel program? Show at least one example.**

The first step is figuring out which tasks can be running parallel with one another. Then it splits the workload into a master/worker. The master creates the array and splits it amongst the available workers, then sends each worker into its own subarray, and then receives the computation from each worker. The worker performs the processing on the subarray. An example of this would be how to find the value for pi.

1. **What is MapReduce?**

MapReduce is a way to processing big data sets with a parallel and distributed algorithm on a cluster.

1. **What is map and what is reduce?**

Map is a function that creates a key/value pair to generate a set of intermediate key/value pairs, then the reduce function merges all of the intermediate values that are associated with the same key.

1. **Why MapReduce?**

Compared to other similar structures (i.e. MPI), MapReduce has built in fault tolerance, which means that a system is still able to operate

1. **Show an example for MapReduce.**

A picture containing text, map

Description automatically generated

1. **Explain in your own words how MapReduce model is executed?**The allocated memory given implements a master, then the master separates the memory into different sections called the workers who are given functions Map and Reduce. Once given their instructions, the worker creates a key/value pair and an intermediate key/value pair is created by the Map Workers. Then the pairs created are written to the local disk and given locations; the locations are given to the master who has to give these locations to the workers. Once the Reduce Worker is given the location, it reads the data from the intermediate keys and sorts them in a way that all same keys are together. Once all of these tasks are completed, the master calls the user program and returns the code.
2. **List and describe three examples that are expressed as MapReduce computations.**

Count of URL Access Frequency: The map function processes logs of who wants to access the webpage and outputs the value <URL,1>, then the reduce function adds all values of the same URL together and gives the result <target, list(source)>.

Distributed Grep: The map function will output a line if it matches a given pattern, then the reduce function will copy the intermediate data into the output.

Reverse Web-Link Graph: The map function shows <target, source> pairs for each link to a target URL from the source. The reduce function takes the list of source URLs with a target URL and outputs the pair: <target, list(source)>.

1. **When do we use OpenMP, MPI and, MapReduce (Hadoop), and why?**

OpenMP: We use OpenMP when we want to introduce shared memory parallelism in our code. It serves as a library that enables multithreading/parallel programming.

MPI: Message Passing Interface is used to develop parallel applications, specifically for science, because they can implement load balance well.

MapReduce: We use MapReduce to reduce the amount of data used in operations and reduce complex problems.

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

When drugs are designed, they find ligands, new pieces, to change the shape of a protein. Once a new ligand is found, they compute a “score” for it, which is calculated based on: how well it fits that protein and produces a desired shape. Then, all new ligands are compared, and they identify the ones with the highest score. The issue is computing the score takes a long time; a way to reduce this is using parallelism in the computational thread for different ligands.

**Parallel Programming Basics**

By: Tony Ngo

**Task 3B**

**A screenshot of a social media post

Description automatically generated**

This is FileZilla, the application that I used to transfer the files from my MacBook to my Raspberry PI.

2.2 Compilation:

A screen shot of a computer

Description automatically generated

I did not run into any issues while creating the 2.2 compilation. However, trying to get the files from my MacBook onto my Raspberry PI was an issue. Through various types of troubleshooting, how I did it was I used FileZilla to transfer the downloaded file from iCollege onto my PI. I created the folder using the command “mkdir sequential” and then copied the folder into there.

3.2 Compilation:

A screen shot of a computer

Description automatically generated

I created the folder using the command “mkdir openmp1”. I transferred the files “openmp1.cpp” and “MakeFile” from my MacBook to the PI using FileZilla. For the 3.2 compilation, I was running to a fatal error where it stated, “**fatal error:** ttb.h: No such file or directory”. I tried using the command “g++ -o dd\_omp dd\_omp.cpp” to alleviate it but I still came across the issue. After some troubleshooting, to fix this error, I had to install the ttb library using the command, “sudo apt-get install libtbb-dev”. Once I did that I was able to run the “make” command to create dd\_omp.

4.1 Compilation:

A picture containing drawing

Description automatically generated

I created the folder using the command “mkdir cplusthreads1”. I transferred the files “dd\_threads.cpp” and “Makefile” from my Macbook to the PI using FileZilla. For the compilation, it went seamlessly because I was able to fix the fatal error by downloading the tbb.h library in the 3.1 compilation.

**5**

|  |  |
| --- | --- |
| **Implementation** | **Time(s)** |
| dd\_serial | 125.53 |
| dd\_omp | 0.02 |
| dd\_threads | 0.02 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Implementation** | **Time(s) 2 Threads** | **Time(s) 3 Threads** | **Time(s) 4 Threads** |
| dd\_omp | 0.02 | 0.04 | 0.22 |
| dd\_threads | 0.02 | 0.04 | 0.15 |

**2.3 – Discussion Questions**

1. **Which approach is the fastest?**

The fastest approach for each is using the C++11 thread solutions

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

Sequential Lines: 170

OpenMP Lines: 193

C++11 Lines: 207

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

OpenMP Real Time (s): 0.95

C++11 Real Time (s): 0.68

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

OpenMP Real Time (s): 73.91

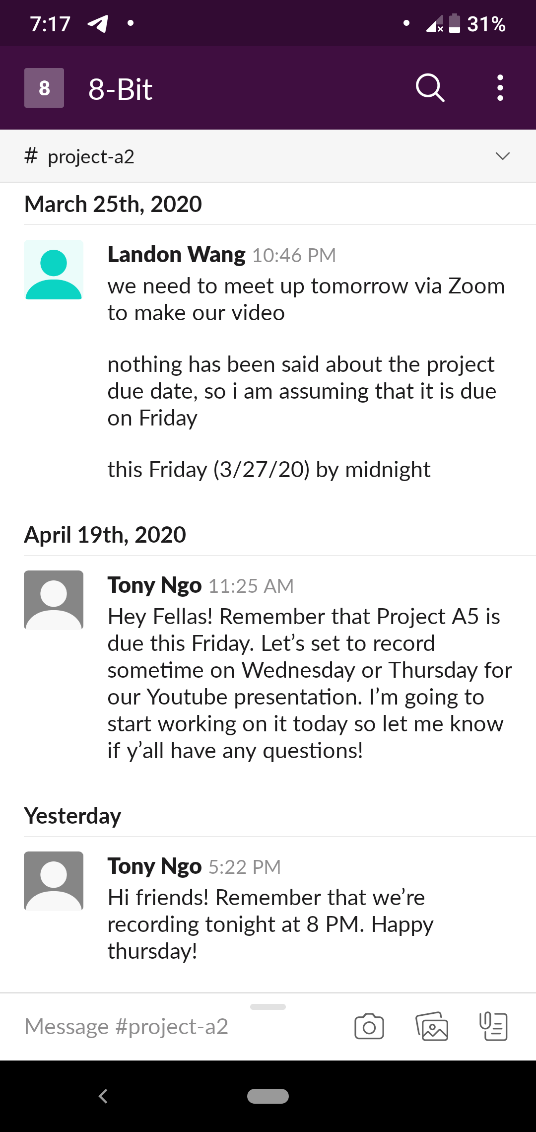
C++11 Real Time (s): 41.66

**Appendix**

Slack: <https://app.slack.com/client/TTK19C222/CTGQG7H7A/user_profile/UTGB5U71S>

GitHub: <https://github.com/Rsandy2/8-Bit>

Presentation on YouTube: https://youtu.be/j84GG4R2bNs

Screenshot of Slack Chat: