Praveen Doluweera

Part A: Foundation

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

These are the basic steps for building a parallel program:

1. Decomposition - This is identifying the parts that can executed simultaneously and concurrently
2. Assignment - Specifying mechanism that is used to divide the work among different processes.
3. Orchestration – This is naming the data, structure the communication and organizing everything to make a model.
4. Mapping – Determine which processes will run on which processor.

An example of parallel programming is scientific simulations like weather forecasting and radar.

**What is MapReduce?**

MapReduce is a technique and program model that is used for processing and generating big data sets using a distributed algorithm on a cluster.

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

Map takes a function and a series of values as inputs and then applies that function to each value in the series. A reduce is a binary operation to combine all the elements in a series.

**Why MapReduce?**

MapReduce is used because is a very cost-effective process. This is very useful when dealing with large amounts of raw data that need to be distributed to different machines quickly.

**Show an example for MapReduce**

An example for MapReduce would be searching for a counting the appearances of a certain character or word in a very large word document. Map Reduce has a function that counts the appearances of each word and then the binary operation of reduce sums up that number to give you a final count.

**Explain in your own words how MapReduce model is executed?**

In MapReduce, the map takes an input pair and produces a set of intermediate key/value pairs. The MapReduce library then groups together all the intermediate values associated with the same intermediate key and then passes it on to the Reduce function. In the Reduce function accepts an intermediate key and a set of values for the key. It then merges the values to form a smaller set of values.

**List and describe 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 copies the supplied intermediate data to the output.

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

Inverted Index – In this the map function parses each document and emits a sequence of <word, document> id pairs. The reduce function accepts all pairs for a given word, sorts the corresponding document ID’s and emits a <word, list(document ID)> pair. The set of all output pairs form a simple inverted index. This is used to keep track of word positions.

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

OpenMP is used to introduce shared memory parallelism to the code. It can split the loop into multiple threads, each of which can process a large chunk of the loop’s iterations.

MPI or Message Passing Interface is a distributed memory parallel model implementation which is typically used to create and develop parallel scientific applications. This is because many scientific applications are tightly synchronous code and well load balanced.

MapReduce is typically used when you have terabytes of data what you want to do operations like extract, transform and load to.

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

Drug designing is an incredibly difficult and lengthy process. We can use a DNA problem to create ligands in a trial and error format to see if they will work well with a protein. It will then sort the different ligands and give a set of high scoring options that can be eventually tested.

Part B: Programming Drug Design and DNA in Parallel

I first put all the required folders and installed the library libtbb-dev which is the thread building blocks in order to make executable files.

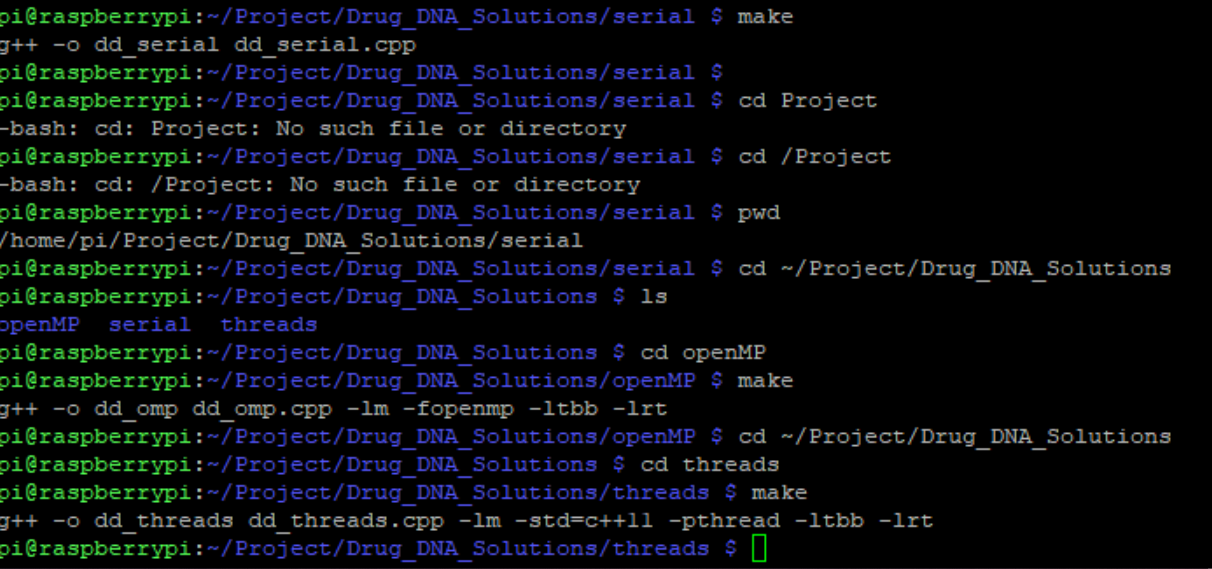


Figure 1

I then ran it to measure the run time with the default values as shown below in Figure 2.

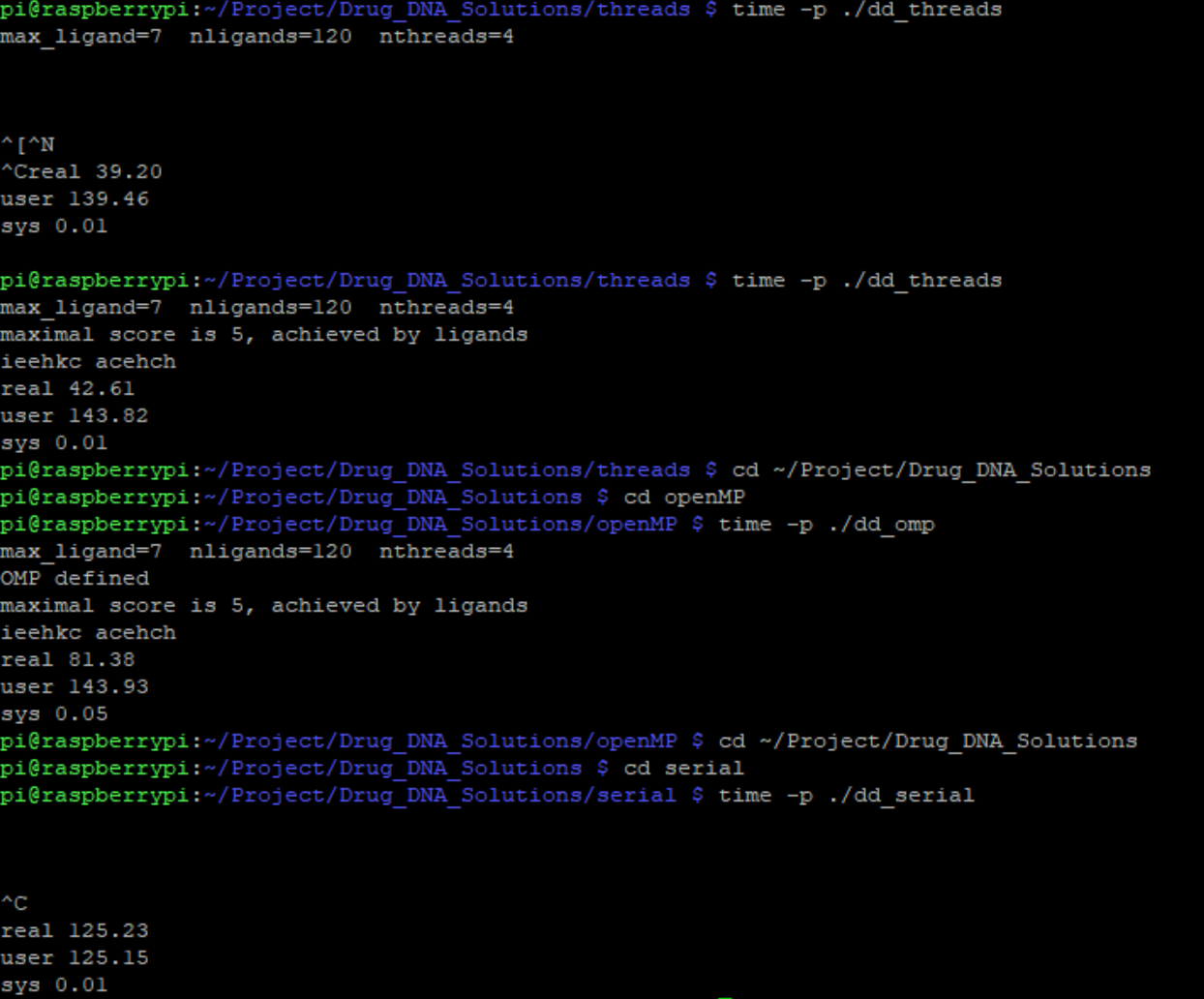


Figure 2

You can see that the Serial took the longest, followed by openMP then threads was the fastest. I then ran the values in the instruction as shown in Figure 3. I then ran the instructions to fill the table as shown below

|  |  |
| --- | --- |
| Implementation | Time(s) |
| dd\_serial (default) | 125.23 |
| dd\_omp (default) | 81.38 |
| dd\_threads(default) | 42.61 |
| dd\_omp 1 (max\_ligand =1 as instructed) | 0.02 |
| dd\_threads 1(max\_ligand = 1 as instructed) | 0.02 |

I then ran the openMP version and the threads with the specs of max\_ligand = 7, nligands = 120, and nthreads = {2,3,4} for each.

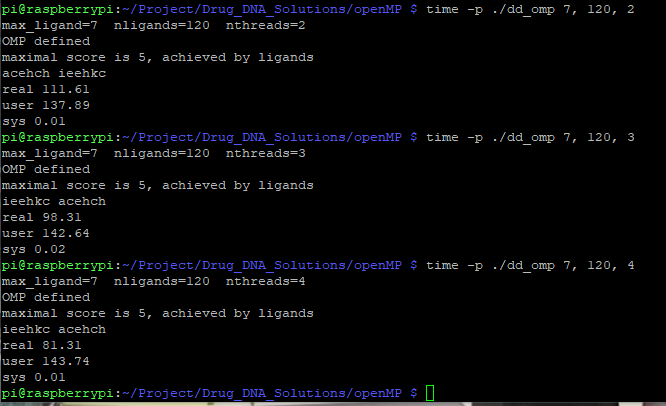


Figure 3

I then ran it for threads with the same conditions as before.



Figure 4

As you can see the more threads given the faster the program executes. Threads is also faster than the openMP version for each case. The times are shown below.

|  |  |  |  |
| --- | --- | --- | --- |
| Implementation | Time 2 Threads | Time 3 Threads | Time 4 Threads |
| dd\_omp | 111.61 | 98.31 | 81.31 |
| dd\_threads | 78.62 | 55.05 | 41.81 |

Discussion Questions

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

The fastest version is the threads version. Also regardless of the approach, increasing the number of threads makes the program run faster.

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

I found that the serial version is the shortest with 170 lines. It is followed by OpenMP which has 193 and Threads is the longest with 207 lines.

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

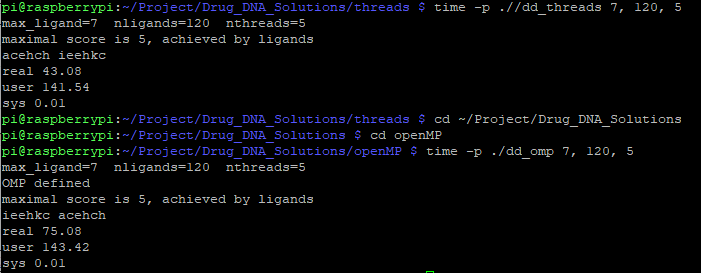


Figure 5

The run times are shown above in Figure 5 and recorder in the table below

|  |  |
| --- | --- |
| Implementation | Time 5 Threads |
| dd\_omp | 75.08 |
| dd\_threads | 43.08 |

As you can see these results are in line with the observations noted above. Threads was faster and in both cases utilizing 5 threads resulted in faster times.

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

This has already been done and shown in Figure 2 as 7 is the default for the value of max\_ligand. The values are recorded below in the table.

|  |  |
| --- | --- |
| Implementation | Time(s) |
| dd\_serial (default) | 125.23 |
| dd\_omp (default) | 81.38 |
| dd\_threads(default) | 42.61 |