6CS005 Learning Journal - Semester 1 2020/21

Poojan Shrestha, 2040364

Table of Contents

1 Parallel and Distributed Systems

1.1 Answer of First Question

A thread is a small collection of instructions programmed independently of the parent to be scheduled and executed by the CPU. For instance, a program might have an open thread waiting for a certain occurrence to occur or running a different job, freeing other tasks to be executed by the main program.

In a modern programming language, threads are designed because whenever a process has several tasks to execute independently of others.

1.2 Answer of Second Question

The two process scheduling policies are as follows:

- **Pre-emptive Scheduling**
  The tasks are often allocated to their priorities in Preemptive Scheduling. In this scheduling tasks are often prioritized according to their importance such as the higher priority task are executed even though the lower priority task is still running. The lower task is paused for a while when the higher priority task completes its execution.
- **Co-operative Scheduling**
  The Processor has been assigned to a particular process in this form of a scheduling system. Either by swapping the background or terminating, the mechanism that holds the CPU busy will free the CPU. It is the only strategy for different hardware platforms that can be used. That is because, like preemptive scheduling, it does not require special hardware (for example, a timer).
  Pre-emptive is more preferred and the thread scheduling algorithm of the Java runtime framework is also pre-emptive.

1.3 Answer of Third Question

In Centralized System, all the computing or tasks are performed on a single computer. It is a system that uses terminals temporarily connected to the central computer to compute at a central location.
In Distributed System all the computing or tasks are distributed to multiple computers. It is a system that has a set of independent computers interconnected where tasks are distributed for concurrent processing without a server.

1.4 Answer of Fourth Question

Transparency is an essential aspect of distributed systems, as it makes running them more user-friendly, simpler, or visible in the eyes of the user. Users should be ignorant of the position of the services and should be clear about the switch from a local computer to a remote machine.

1.5 Answer of Fifth Question

B = A + C is a flow dependency, C = B + D is an anti-dependency, B = C + D is an output dependency.

1.6 Answer of Sixth Question

First Program: 349151
Second Program: 500000

Because both programs use a thread function that performs a thread count for the provided unassigned integer [N]. The first program runs an extra loop.

2 Applications of Matrix Multiplication and Password Cracking using HPC-based CPU system

2.1 Single Thread Matrix Multiplication

- The analysis of the algorithm's complexity. (1 mark)
  Ans: **$O(n^3)$** is the complexity of the given program.

- Suggest at least three different ways to speed up the matrix multiplication algorithm given here. (Pay special attention to the utilisation of cache memory to achieve the intended speed up). (1 marks)
  Ans: Different ways to speed up the matrix multiplication would be Divide and Conquer and Strassen's Matrix Multiplication.

- Write your improved algorithms as pseudo-codes using any editor. Also, provide reasoning as to why you think the suggested algorithm is an improvement over the given algorithm. (1 marks)

```
if n = threshold then compute
            Z = x + y is a conventional matrix.
      Else
            Partition x into four sub matrices x11, x12, x21, x22.
            Partition y into four sub matrices y11, y12, y21, y22.
            Strassen (n/2, x11 + x22, y11 + y22, w1)
            Strassen (n/2, x21 + x22, y11, w2)
            Strassen (n/2, x11, x12 - y22, w3)
            Strassen (n/2, x22, y21 - y11, w4)
            Strassen (n/2, x11 + x12, y22, w5)
            Strassen (n/2, x21 - x11, y11 + y22, w6)
            Strassen (n/2, x12 - x22, y21 + y22, w7)

            Z = w1 + w4 - w5 + w7          w3 + w5
                w2 + w4                w1 + w3 - w2 - w6

      end if

      return(W)

end
```

- Write a C program that implements matrix multiplication using both the loop as given above and the improved versions that you have written. (1marks)

  Include your code using a text file in the submitted zipped file under name Task2.1

- Measure the timing performance of these implemented algorithms. Record your observations. (Remember to use large values of N, M and P – the matrix dimensions when doing this task). (1 marks)
  Ans: The timing performance of the improved program is 20.4385 seconds.
  Insert a paragraph that hypothesises how long it would take to run the original and improved algorithms. Include your calculations.
  Explain your results of running time.
  Ans: Time complexity of Original Program: $O(n^3)$
  Time complexity of Improved Program: $O(n^{2.80})$.
  Hence, Improved Program is much faster than the Original Program

2.2 Multithreaded Matrix Multiplication

- Include your code using a text file in the submitted zipped file under name Task2.2
- Insert a table that has columns containing running times for the original program and your multithread version. Mean running times should be included at the bottom of the columns.
- Insert an explanation of the results presented in the above table.

Number of Time Programs ran	Time taken by Original Program	Time taken by Multithread Program
1	20.412	3
2	20.156	3
3	20.652	3
4	20.235	3
5	20.512	3
6	20.623	3
7	20.452	3
8	20.265	3
9	20.653	3
10	20.425	3
Average (seconds)	**20.4385**	**3**

Ans: The first program which is Original program was ran on CodeBlocks and the second program which is Multithread program was ran on Ubuntu Terminal.
Comparing both program, Original program used a single thread whearas Multithread program used two thread which broke down a single process into two process making it faster to execute

2.3 Password cracking using POSIX Threads

- Include your code using a text file in the submitted zipped file under name Task2.3.1, Task2.3.3, Task2.3.5
- Insert a table of 10 running times and the mean running time.
- Insert a paragraph that hypothesises how long it would take to run if the number of initials were to be increased to 3. Include your calculations.
- Explain your results of running your 3 initial password cracker with relation to your earlier hypothesis.
- Write a paragraph that compares the original results with those of your multithread password cracker.

Number of Programs	Time in nanosecond	Time in seconds
1	148831480552.00	148.8314806
2	145810733719.00	145.8107337
3	145903921347.00	145.9039213
4	145833303542.00	145.8333035
5	145674117202.00	145.6741172
6	145592184620.00	145.5921846
7	145611231362.00	145.6112314
8	145605059608.00	145.6050596
9	145936199061.00	145.9361991
10	145927396998.00	145.927397
Average	**146072562801.10**	**146.0725628**

For the three initials a loop is added on the two initials code, so the loop goes through another 26 characters. So,

Estimated Time = Original Time * 26 (Here, 26 is the number of alphabets)

= 146.0725628 * 26

= 3,797.8866328 seconds

Now,

Seconds into minutes = 3,797.8866328 / 60

= 63.29811054666667 minutes

Hence, the estimated time for three initials is 63 minutes.

Number of Programs	Time in nanosecond	Time in seconds
1	5299177490263.00	5299.17749
2	5280573811093.00	5280.573811
3	5289536270541.00	5289.536271
4	5281426879345.00	5281.426879
5	5291635214781.00	5291.635215
6	5298254194655.00	5298.254195
7	5299132456847.00	5299.132457
8	5298324569158.00	5298.324569
9	5298789654126.00	5298.789654
10	5297874998463.00	5297.874998
Average	**5293472553927.20**	**5293.472554**

So, the average running time for three initials is 5293.472554 seconds which is 88.224542566667 minutes i.e., Approximately 1.47 hours. The estimated time for three initials was 63.29811054666667 minutes but the original running time was a lot more than estimated which is a huge difference. For such a difference in the running time, the reason could be due to the background process during the execution of three initials.

Number of Time Programs ran	Time taken by Original Program	Time taken by Multithread Program
1	148.831480552	142.051412568
2	145.810733719	138.248301761
3	145.903921347	135.939179243
4	145.833303542	139.407503836
5	145.674117202	141.657376717
6	145.592184620	142.386132090
7	145.611231362	137.325570259
8	145.605059608	139.540573378
9	145.936199061	136.086533270
10	145.927396998	143.868812312
Average (seconds)	**146.072562801**	**139.651139543**

So, the results were not too different, but the multithread ran a bit faster that took 139.6511395 seconds than the original program that took about 146.0725628 seconds because in original program only single thread was used but in Multithread program two threads were used.

3 Applications of Password Cracking and Image Blurring using HPC-based CUDA System

3.1 Password Cracking using CUDA

- Include your code using a text file in the submitted zipped file under name Task3.1
- Insert a table that shows running times for the original and CUDA versions.
- Write a short analysis of the results

Number of Time Programs ran	Time taken by Original Program	Time taken by Multithread Program	Time taken by CUDA Program
1	148.831480552	142.051412568	0.292592451
2	145.810733719	138.248301761	0.243579181
3	145.903921347	135.939179243	0.245739428
4	145.833303542	139.407503836	0.255705678
5	145.674117202	141.657376717	0.244173225
6	145.592184620	142.386132090	0.250527895
7	145.611231362	137.325570259	0.256544903
8	145.605059608	139.540573378	0.241956623
9	145.936199061	136.086533270	0.245235216
10	145.927396998	143.868812312	0.254830959
Average (seconds)	**146.072562801**	**139.651139543**	**0.253088556**

So, the time taken by the CUDA version of password cracking is about 0.25 seconds which is a lot faster than the other two i.e., Original program and Multithread Program. The reason for such significant difference is CUDA run on GPU and POSIX run CPU which make GPU substantially with more CUDA and with a capability of parallel computing which can large amount of data parallelly.

3.2 Image blur using multi dimension Gaussian matrices

- Include your code using a text file in the submitted zipped file under name Task3.2
- Insert a table that shows running times for the original and CUDA versions.
- Write a short analysis of the results

Number of Time Programs ran	Time taken by Original Program	Time taken by CUDA Program
1	0.06800574	0.139346108
2	0.068203702	0.126243154
3	0.068025118	0.115408242
4	0.068162071	0.117525246
5	0.068605571	0.115750248
6	0.071627892	0.116583102
7	0.06925862	0.115026537
8	0.072021059	0.113833336
9	0.069532367	0.116385869
10	0.068682408	0.117078551
Average (seconds)	**0.069212455**	**0.119318039**

So, the results were quite outstanding the CPU was faster than the GPU Image Blur program. Reason could be because of the hardware performance.