March 30, 2023

Jaleta Fanta Tesgera

CSC 474: Operating Systems Concepts

Professor Jun Kong

*Project 1 Summary*

Before filling out the summary and proceeding with a discussion, I wanted to run the code a few times and use the average of three attempts for a more accurate prediction. I have used seconds accurate to decimal points as unit to fill into my table.

**File 1**

With 1 child process

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.0001146 | 0.001015 | 0.001000 | 0.0007099 |

With 2 child processes

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.001958 | 0.001920 | 0.001789 | 0.001889 |

With 4 child processes

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.003174 | 0.003761 | 0.003505 | 0.0034793 |

**File 2**

With 1 child process

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.001344 | 0.001437 | 0.001206 | 0.001329 |

With 2 child processes

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.001601 | 0.001919 | 0.001721 | 0.001747 |

With 4 child processes

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.003285 | 0.003237 | 0.003155 | 0.0032256 |

**File 3**

With 1 child process

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.002389 | 0.002248 | 0.003884 | 0.0028403 |

With 2 child processes

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.001589 | 0.004723 | 0.002595 | 0.002969 |

With 4 child processes

|  |  |  |  |
| --- | --- | --- | --- |
| *Trail 1* | *Trail 2* | *Trail 3* | *Average* |
| 0.003496 | 0.004839 | 0.007160 | 0.005165 |

Finally this is what the table values looked like

|  |  |  |  |
| --- | --- | --- | --- |
| Table 1 | File 1 | File 2 | File 3 |
| 1 Process | 0.0007099 | 0.001329 | 0.0028403 |
| 2 Processes | 0.001889 | 0.001747 | 0.002969 |
| 4 Processes | 0.0034793 | 0.0032256 | 0.005165 |

*Discussion of Data and Results*

Looking at the numbers it is clearly visible that most of the trails have had results around the same range; however, there are a few anomalies like the first data with one child process on the first try and the rest re different by a whole decimal point. Furthermore, we can also infer that the data is generally increasing as the number of processes increases. Taking a deeper look, we can see that the first process is much closer to the second process, and the four process data is much larger and further away than its counterparts. The time also increases across the table, such that as the number of lines in the data increases it takes more time to process the data. The other anomaly we can notice is that it takes less time to process a 10,000 line file with 2 child processes than 1,000 lines of data with two processes. The fastest time is processing a 1,0000 line file with one process and the slowest is processing 100,000 lines of data with 4 child processes.

Enough is said about the number results, and now let us move on to a more theoretical discussion. As we have learned in class we expect that having four process will run faster than a single process; however, we are noticing the opposite in which one child process is running faster than all the other trails. There are several reasons that this might be happening, which as mentioned might be the run time resources. These resources are the availability of hardware and software to assist a program during a run time, these include CPU or memory. Since I am running this code on a Linux virtual machine installed on windows, the CPU might be busy executing other tasks which might generally slow down multiple process execution. Moreover, when running multiple processes I have declared more variables that will take the sum and pipe it to the parent and other variables that will take the incoming variables; as a result more memory occupation. However, when running a single process its just one smooth straight line, and it prevents multiple memory usage. Moreover, on a multiple process has so multiple pipes that the parent has to wait for all the process to report them which this communication and wait time slows down execution. However, in a single execution the parent only waits for one pipe, and it reports the sum to the user. Picking off the discussion about multiple variable declaration, in the multiple process there is time it takes to execute the arithmetic of adding multiple reports from child process before reporting the final sum. However, in a single execution, there is no need to perform execution, it takes the value and reports it to the user. Finally, even creating(forking) those multiple process and dividing the data/work among those processes takes more time and a result this will be much slower than running a process with a single child.

To reproduce my results I have submitted the zip file with the code and the test data that I have used. In the future, to hopefully align with out theories, and see that multiple process might be faster we might need a very large data, in billion lines, and running through them with a single and two child process and maybe this will be where we can see the difference.