**Operating Systems assignment 3**

Mahmoud Abdallah – 140296 – Section 2

Important Note:

Due to some limitations using [**python.threading**](https://docs.python.org/3/library/threading.html) (threading can’t work in parallel, only concurrently) so I’ve used [**python.Multiprocessing**](https://docs.python.org/3/library/multiprocessing.html) which is a multiprocessing library that can run in parallel and it work exactly like parallel threads.

**Code**

threadsmanager.py:

A file contains custom built classes ParentThread and ChildThread,

every ParentThread contains a target function and other related children that going to wait until its children finish execution, then start its own target function execution.

tasks.py:

The that contains all the utilities needed in Part1. A.

Tasks class can generate random data set and pass it to any other function inside or outside the class using generate\_dataSet() and get\_dataSet()

Beside calc\_max(), calc\_min() and calc\_avg() that used in Part1. A.

**Part 1**

Note :

I’m going to merge part C with A and B in each part, that will make the report more precise and organized.

**A**

We will call the function parallel () which is going to create threads for each process, and we will measure its execution time using [python.timeit](https://docs.python.org/3/library/timeit.html)

Then we will call single () which is going to execute the code as normal serial process using one thread.

Results

103

Text

Description automatically generated

The result shows that the single computing process actually work faster than parallel threading due to two reasons:

1. Data set is too small for the computer and won’t really challenge the CPU
2. Initializing threads and managing them takes small amount of the overall process time and we can notice that when the data set is small therefore using threads when dealing with small data set and non-complex computation isn’t a great idea!

106

Text

Description automatically generated

106 still isn’t challenging enough for parallel threading as we notice parallel and single computation has almost the same execution time!

108

Text

Description automatically generated

Finally, when the data set is big enough to challenge parallel and single computation, single computation loses with almost x1.5 the time that parallel needs to finish the execution.

**B**

Results

103

Text

Description automatically generated

We got the same result we had in A when we used the same data set size and its for the same reasons

106

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

Text

Description automatically generated

The result show that even when the data set is relatively large parallel threading won’t help in such a task because there is no multiple process needed to get executed in parallel!

So, single and parallel take almost the same execution time.

Conclusion:

Parallel threading is useful if the code satisfies two conditions:

1. Data set is large relative to code complexity.
2. Code has multiple functions to get executed at the same time.

Otherwise, there is no need to use parallel threading.

**Part 2**

Note :

Like what I did in part1 I will merge a. with b. for a better explanation.

Results

103

Text

Description automatically generated

We go the same result as part1. A. and its for the same reasons.

106

Text

Description automatically generated

Data set is challenging enough for single thread processing while parallel multithreading is working slightly better now.

108

Text

Description automatically generated

Result shows a huge performance gap between parallel and single computing with more than twice the performance!

Since we are working for sets of (25\*106 ) in parallel, while (25\*106 ) for four times in single thread computing.