**2023-1 Multicore Computing, Project #1**

Problem 1

2023-04-18

CAU SW 20184286

LEE DONGHWA

**Environment**

CPU type\_ Intel Core i5-8265U 1.60GHz, Hyper Threading ON

# of core\_ 4

Memory size\_ 8GB

OS type\_ Window 11 Pro

**Result**

Tables\_ (unit : ms, task size : 10)

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Exec time** | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 32 |
| Static(block) | 6250 | 3962 | 2670 | 2260 | 2116 | 2096 | 1959 | 1898 | 1952 | 2064 |
| Static(cyclic) | 4793 | 3128 | 2193 | 2071 | 1713 | 1753 | 1896 | 1788 | 1760 | 1707 |
| Dynamic | 5348 | 2963 | 2199 | 1907 | 1737 | 1708 | 1650 | 1705 | 1679 | 1749 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Performance** | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 | 32 |
| Static(block) | 0.00016 | 0.000252 | 0.000375 | 0.000442 | 0.000473 | 0.000477 | 0.00051 | 0.000527 | 0.000512 | 0.000484 |
| Static(cyclic) | 0.000209 | 0.00032 | 0.000456 | 0.000483 | 0.000584 | 0.00057 | 0.000527 | 0.000559 | 0.000568 | 0.000586 |
| Dynamic | 0.000187 | 0.000337 | 0.000455 | 0.000524 | 0.000576 | 0.000585 | 0.000606 | 0.000587 | 0.000596 | 0.000572 |

Graphs\_

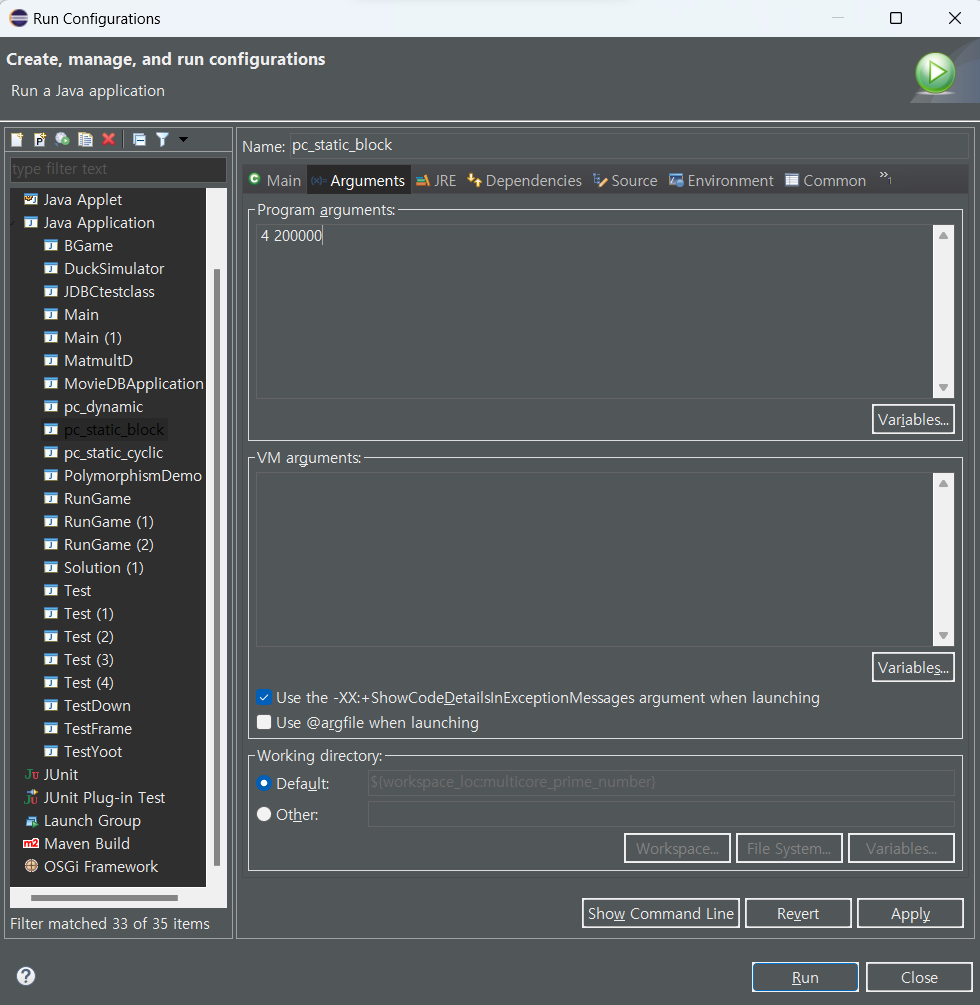
Explanation / Analysis\_

First, when comparing block and cyclic, in the case of block, the threads with lower numbers finished executing quickly, while the threads with higher numbers finished execution more slowly. Through this, it was clear that as the number gets larger, more calculations are required to determine whether it is a prime number or not.

On the other hand, cyclic can assign threads evenly even for large numbers, resulting in almost equal execution time among all threads. In contrast to cyclic, where work can be evenly distributed among threads, block was uneven, causing relatively lower performance due to one thread being overloaded and having to wait for it to finish.

Secondly, as the number of threads increased, performance increased in all cases. Particularly noteworthy is that when one thread became two, there was a very high increase in performance, with almost a 2x decrease in execution time for 1-4 threads. However, as the number of threads increased to 12-32, the efficiency of the increase was not as significant. This may be due to Amdahl's law, which states that a computer program is composed of parallelizable and non-parallelizable parts, so there is a limit to the improvement in performance unless further parallelization is possible.

**Program Execution**



In eclipse, set the argument value and file input by using the menu [Run]->[Run Configurations]. And then, [Argument] menu, you can change the number of threads. Applies to pc\_static\_block, pc\_static\_cyclic and pc\_dynamic. And press ‘Run’ button then you can execute my code.

Execution Output\_

텍스트이(가) 표시된 사진

자동 생성된 설명

1. pc\_static\_block
2. pc\_static\_cyclic
3. pc\_dynamic.

**Source Code**

1. pc\_static\_block.java

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

1. pc\_static\_cyclic.java

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명

1. pc\_dynamic.java

텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명 텍스트이(가) 표시된 사진

자동 생성된 설명