

2023.1 Multicore Computing, Project #3

Problem 1

|  |  |
| --- | --- |
| Course / Class: | Multicore Computing / Class 01 |
| Instructor: | Bong-Soo Sohn |
| Date: | 2023. 05. 24 |
| Student ID: | 20184757 |
| Student Name: | Youngseok Joo |

# **Table of Contents**

[Table of Contents 2](#_Toc135839237)

[1. Environment 3](#_Toc135839238)

[2. Table / Graph 4](#_Toc135839239)

[1) Execution Time 4](#_Toc135839240)

[2) Performance 5](#_Toc135839241)

[3. Explanation 6](#_Toc135839242)

# **Environment**

* Hardware
  + MacBook Air (M2, 2022)
  + Processor: Apple M2 (8 Core — 4 Efficiency + 4 Performance, Maximum clock speed 3.49 GHz)
  + Memory: 16 GB (SoC — 6,400 MT/s LPDDR5 SDRAM in a unified memory configuration)
* Operating System
  + macOS Ventura 13.3.1
* Testing Environment
  + macOS Terminal (version 2.13) — zsh
  + clang — gcc -Xclang -fopenmp -lomp   
    -L/opt/homebrew/opt/libomp/lib -I/opt/homebrew/opt/libomp/include
    - macOS gcc compiler is linked automatically to clang compiler

# **Table / Graph**

## Execution Time

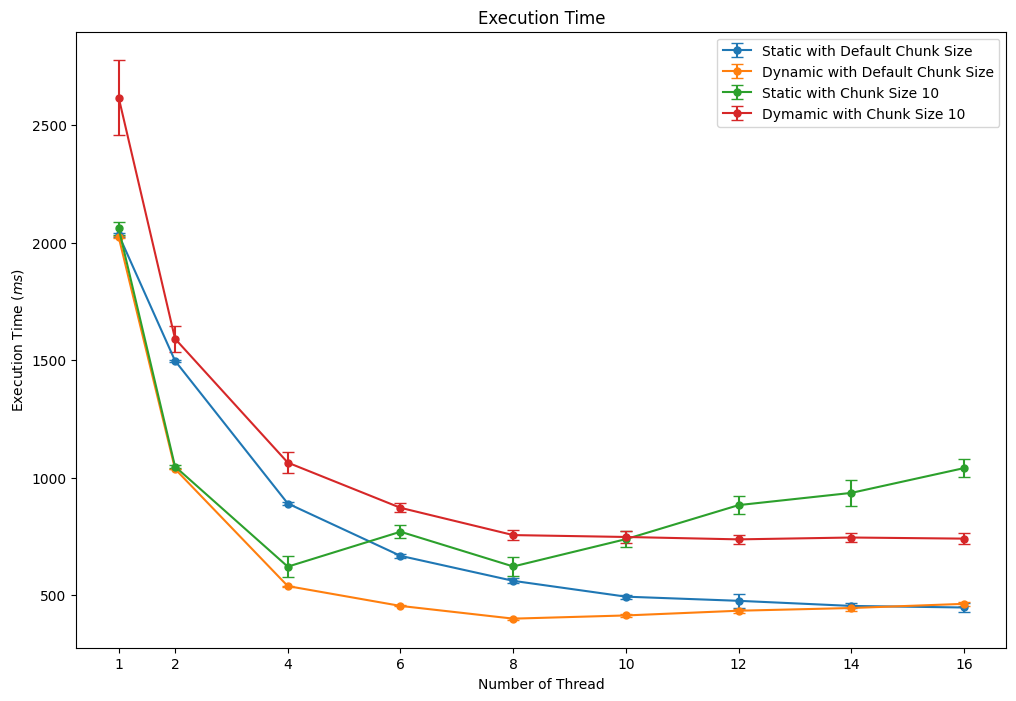


Figure 1. Error Bar Graph of Execution Time using Static and Dynamic Scheduling with Chunk Size 10 and Default (10-Fold).

Table 1. Table showing Average Execution Time using Static and Dynamic Scheduling with Chunk Size 10 and Default (10-Fold).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Execution Time *(ms)* | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| Static (Default Chunk Size) | 2033.6903 | 1497.1466 | 889.6232 | 666.7727 | 560.4144 | 493.0341 | 475.3363 | 453.5708 | 446.9308 |
| Dynamic (Default Chunk Size) | 2024.1164 | 1036.7059 | 538.0645 | 453.6482 | 399.4294 | 413.0776 | 433.1608 | 444.7208 | 462.4706 |
| Static (Chunk Size 10) | 2060.9837 | 1046.9497 | 620.7251 | 769.4425 | 621.5246 | 737.8764 | 882.8032 | 934.8341 | 1040.4399 |
| Dynamic (Chunk Size 10) | 2617.8197 | 1591.0779 | 1063.6797 | 871.2862 | 755.1588 | 747.1143 | 736.9287 | 744.819 | 740.0463 |

## Performance

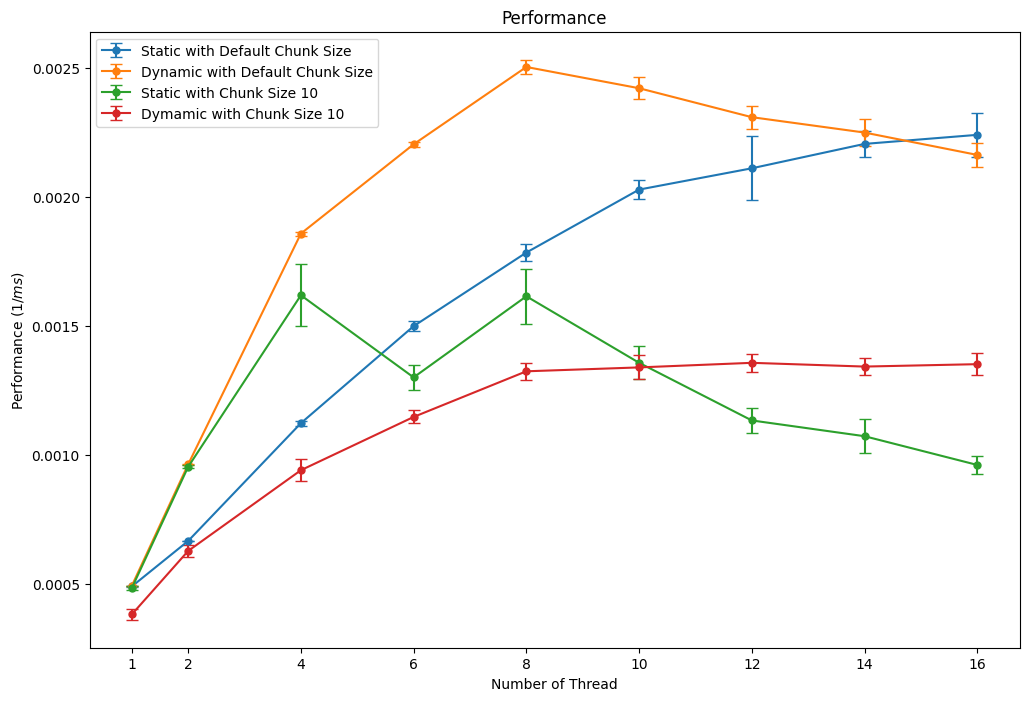


Figure 2. Error Bar Graph of Performance using Static and Dynamic Scheduling with Chunk Size 10 and Default (10-Fold).

Table 2. Table showing Average Performance using Static and Dynamic Scheduling with Chunk Size 10 and Default (10-Fold).

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Performance *(1/ms)* | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| Static (Default Chunk Size) | 0.00049173 | 0.00066794 | 0.00112415 | 0.00149999 | 0.00178506 | 0.00202896 | 0.00211144 | 0.00220592 | 0.00224099 |
| Dynamic (Default Chunk Size) | 0.00049405 | 0.0009646 | 0.00185855 | 0.00220439 | 0.00250389 | 0.00242161 | 0.00230949 | 0.00224989 | 0.00216327 |
| Static (Chunk Size 10) | 0.0004853 | 0.00095519 | 0.00161969 | 0.00130149 | 0.00161568 | 0.00135812 | 0.00113478 | 0.00107369 | 0.00096238 |
| Dynamic (Chunk Size 10) | 0.00038325 | 0.0006293 | 0.00094197 | 0.00114828 | 0.00132511 | 0.00134016 | 0.00135793 | 0.00134343 | 0.00135271 |

# **Explanation**

First, before analyzing the results, expectations on execution time in each case can be made. The default chunk size is *loop\_count/number\_of\_threads* in static scheduling, which is the same as block decomposition, and *1* in dynamic scheduling. Then, we can guess that cases with chunk size ten will have shorter execution times, and cases with default chunk size will have longer execution times.

As shown in Figure 1, among four scheduling methods, dynamic with default chunk size had the shortest execution time in all number of threads. Also, static with default chunk size had the second shortest execution time when using more than four threads.

In three cases, static with default chunk size, chunk size 10, and dynamic with chunk size 10 had similar execution times when using a single thread. However, dynamic with chunk size 10 had longer execution times than others, not only when using a single thread but in almost every thread number. There is no software-related reason to guess this, but the test execution order may have caused this result. The test was executed as ‘Static with default chunk size,’ ‘Dynamic with default chunk size,’ ‘Static with chunk size 10,’ and ‘Dynamic with chunk size 10’. Since testing hardware has no cooling fans and testing was done without any resting time between each execution, too much heat could have been generated from the CPU, and throttling may have caused poor performance and longer execution time.

Static with chunk size 10 had two lowest peaks in execution time, four and eight threads. The number of physical cores may have caused this. As specified in Section 1, the testing environment CPU has four performance cores and four efficiency cores. It is unclear which cores were assigned the work, but it is possible to guess that there is a relationship with the number of physical cores. Also, execution order may have caused this increase and instability in execution time.