중앙대학교
CHUNG ANG UNIVERSITY

# 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

# 1. 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

# 2. Table / Graph

## 1) Execution Time

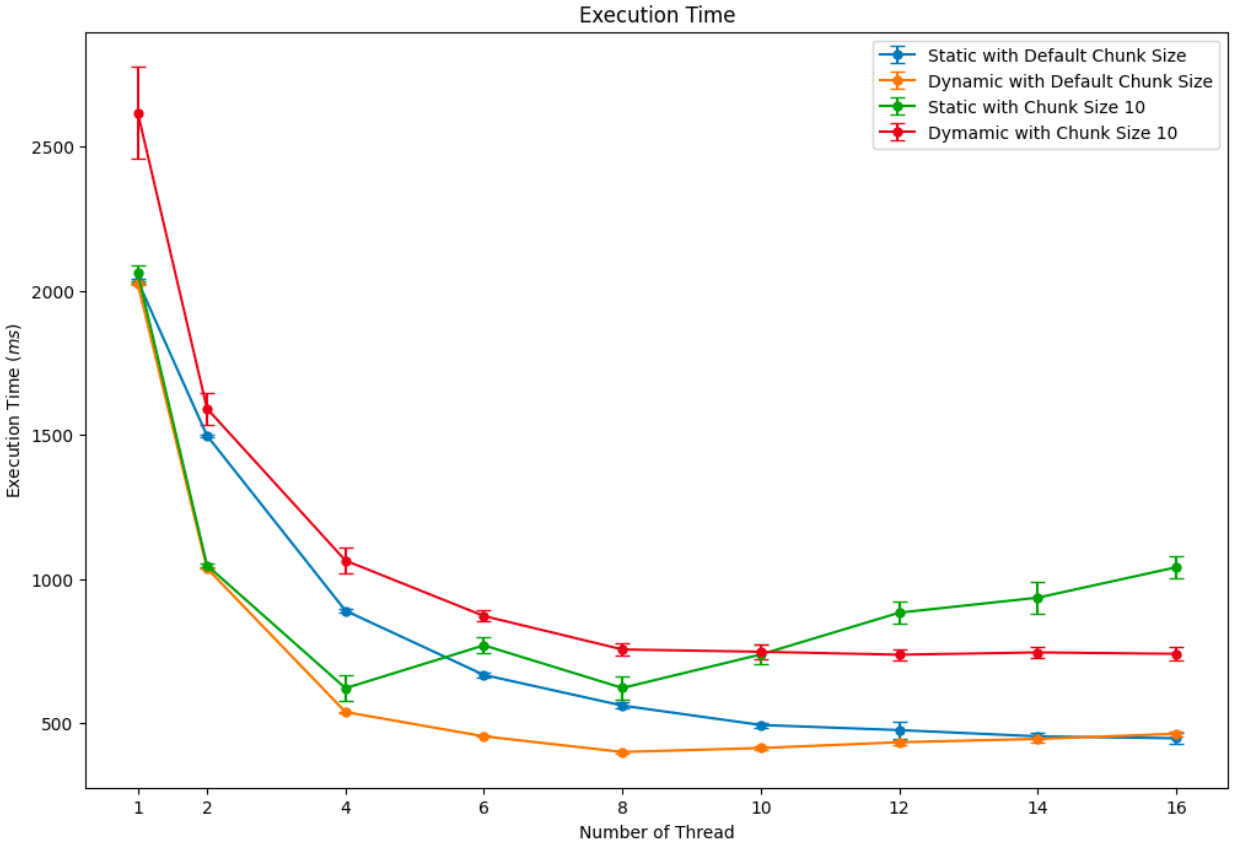


*Figure 1. Error Bar Graph of Execution Time using Static and Dynamic Scheduling with Chunk Size 10 and Default (10-Fold).*

*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 |

## 2) Performance

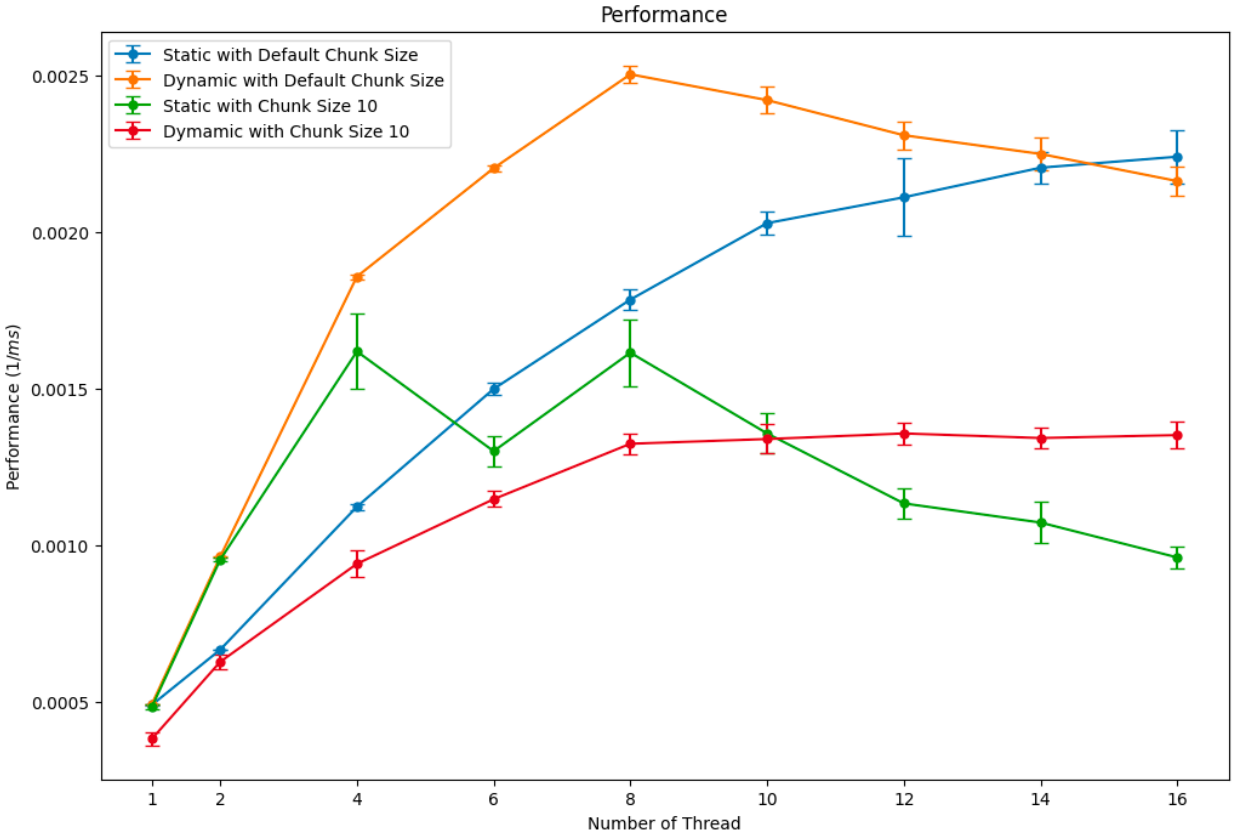


*Figure 2. Error Bar Graph of Performance using Static and Dynamic Scheduling with Chunk Size 10 and Default (10-Fold).*

*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) | 0.00049 173 | 0.00066 794 | 0.00112 415 | 0.00149 999 | 0.00178 506 | 0.00202 896 | 0.00211 144 | 0.00220 592 | 0.00224 099 |
| Dynamic (Default Chunk Size) | 0.00049 405 | 0.00096 46 | 0.00185 855 | 0.00220 439 | 0.00250 389 | 0.00242 161 | 0.00230 949 | 0.00224 989 | 0.00216 327 |
| Static (Chunk Size 10) | 0.00048 53 | 0.00095 519 | 0.00161 969 | 0.00130 149 | 0.00161 568 | 0.00135 812 | 0.00113 478 | 0.00107 369 | 0.00096 238 |
| Dynamic (Chunk Size 10) | 0.00038 325 | 0.00062 93 | 0.00094 197 | 0.00114 828 | 0.00132 511 | 0.00134 016 | 0.00135 793 | 0.00134 343 | 0.00135 271 |

# 3. 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.