

2023.1 Multicore Computing, Project #3

Problem 2

|  |  |
| --- | --- |
| 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](#_Toc135861697)

[1. Environment 3](#_Toc135861698)

[2. Table / Graph 4](#_Toc135861699)

[1) Execution Time 4](#_Toc135861700)

[2) Performance 5](#_Toc135861701)

[3. Explanation 7](#_Toc135861702)

# **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

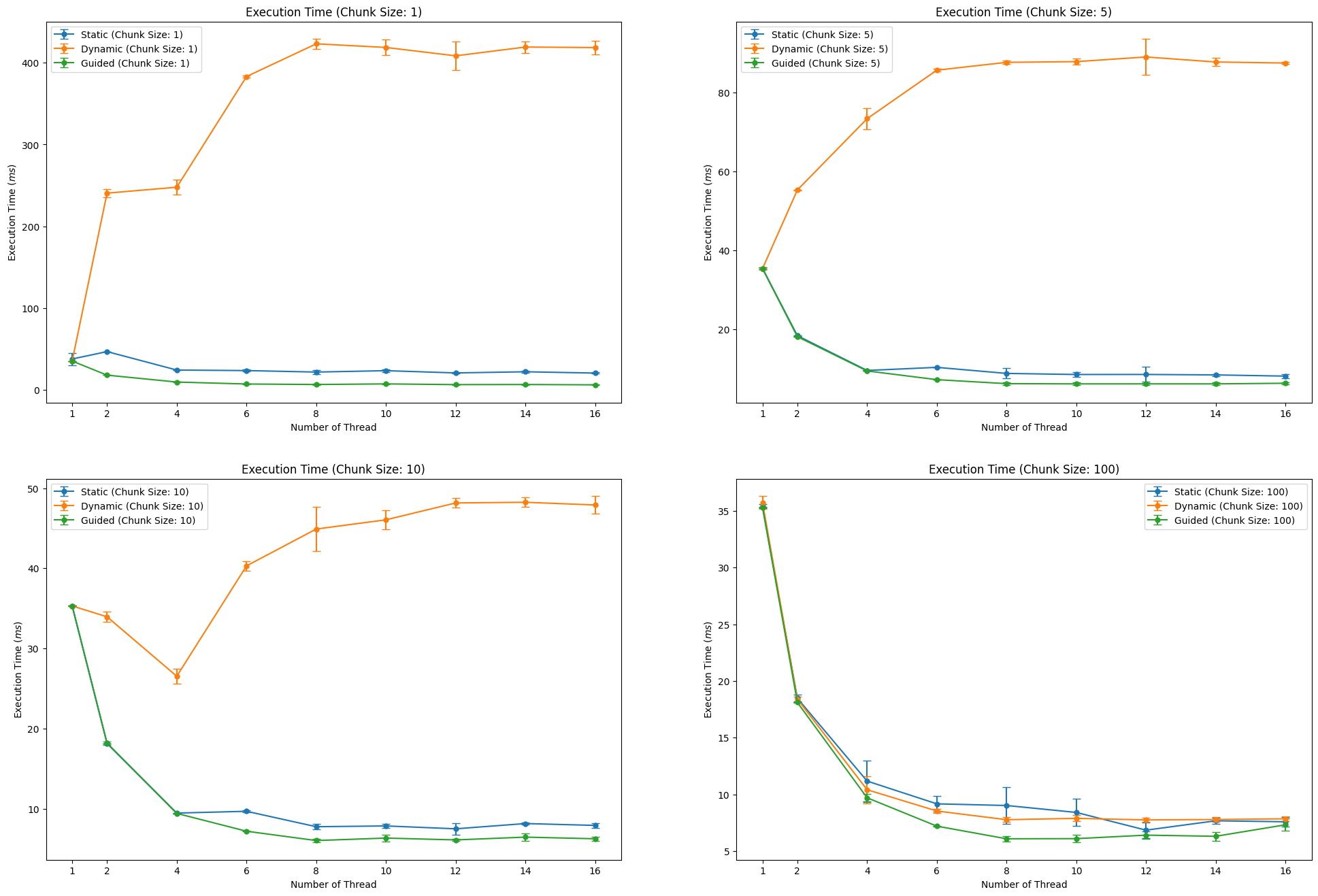


Figure 1. Error Bar Graphs of Execution Time using Static, Dynamic, and Guided Scheduling in Different Chunk Sizes (10-Fold).

A picture containing line, plot, diagram, text

Description automatically generated

Figure 2. Error Bar Graphs of Execution Time using Chunk Sizes 1, 5, 10, and 100 in Different Scheduling Schemes (10-Fold).

Table 1. Table showing Average Execution Time using Different Scheduling Schemes and Chunk Sizes (10-Fold).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Execution Time *(ms)* | Chunk Size | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| Static | 1 | 37.8461 | 46.8988 | 24.3494 | 23.6849 | 21.881 | 23.5707 | 20.8306 | 22.2372 | 20.6411 |
| Dynamic | 35.3322 | 240.559 | 247.7957 | 383.0341 | 423.0457 | 418.6088 | 408.3619 | 419.1155 | 418.4885 |
| Guided | 35.3105 | 18.1108 | 9.6657 | 7.2931 | 6.6585 | 7.416 | 6.5209 | 6.6462 | 6.2761 |
| Static | 5 | 35.4016 | 18.3613 | 9.5432 | 10.3336 | 8.7958 | 8.5212 | 8.532 | 8.4181 | 8.1129 |
| Dynamic | 35.3603 | 55.2999 | 73.3646 | 85.6627 | 87.6504 | 87.825 | 89.0107 | 87.7361 | 87.4747 |
| Guided | 35.3939 | 18.1142 | 9.4267 | 7.2172 | 6.2039 | 6.1433 | 6.1495 | 6.1461 | 6.3064 |
| Static | 10 | 35.3046 | 18.14 | 9.4579 | 9.6841 | 7.7582 | 7.8543 | 7.4974 | 8.1527 | 7.9212 |
| Dynamic | 35.3056 | 33.9541 | 26.5344 | 40.3051 | 44.9028 | 46.0538 | 48.1588 | 48.2437 | 47.9024 |
| Guided | 35.3099 | 18.196 | 9.4205 | 7.1963 | 6.0319 | 6.3407 | 6.1114 | 6.4599 | 6.2535 |
| Static | 100 | 35.375 | 18.472 | 11.1781 | 9.1725 | 9.0264 | 8.4157 | 6.8534 | 7.6796 | 7.5932 |
| Dynamic | 35.7212 | 18.3933 | 10.4156 | 8.5418 | 7.7752 | 7.8939 | 7.7548 | 7.7914 | 7.8533 |
| Guided | 35.3035 | 18.1077 | 9.7051 | 7.2072 | 6.0936 | 6.1058 | 6.4143 | 6.3137 | 7.3257 |

## Performance

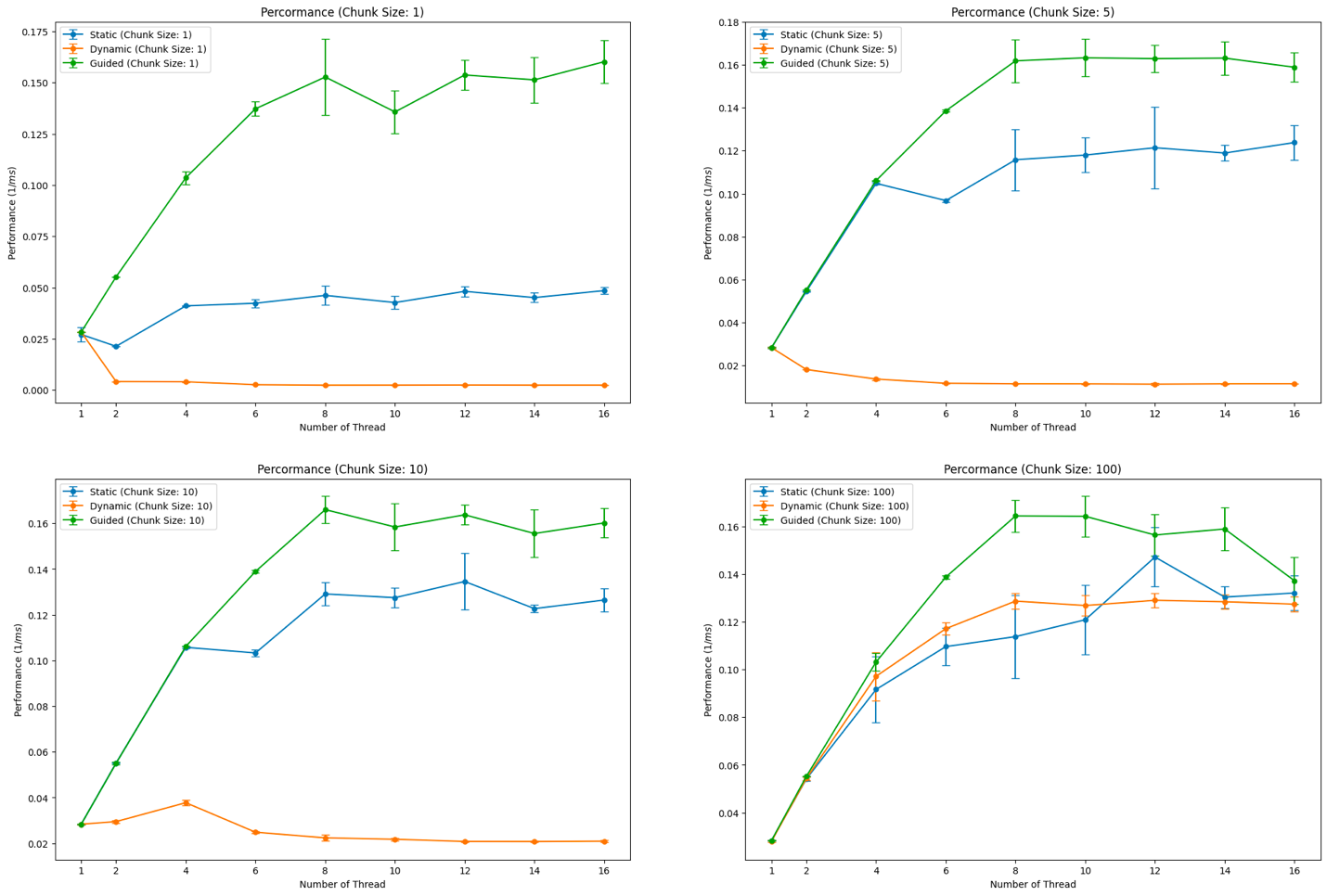


Figure 3. Error Bar Graphs of Performance using Static, Dynamic, and Guided Scheduling in Different Chunk Sizes (10-Fold).

A picture containing line, diagram, plot, screenshot

Description automatically generated

Figure 4. Error Bar Graphs of Performance using Chunk Sizes 1, 5, 10, and 100 in Different Scheduling Schemes (10-Fold).

Table 2. Table showing Average Performance using Different Scheduling Schemes and Chunk Sizes (10-Fold).

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Performance *(1/ms)* | Chunk Size | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| Static | 1 | 0.02712064 | 0.02132277 | 0.04106971 | 0.04233131 | 0.04618477 | 0.04267625 | 0.04813753 | 0.04509577 | 0.04850948 |
| Dynamic | 0.02830284 | 0.00415867 | 0.00404089 | 0.00261075 | 0.0023643 | 0.00239018 | 0.00245356 | 0.00238667 | 0.0023905 |
| Guided | 0.0283202 | 0.0552157 | 0.10355654 | 0.13721478 | 0.15268157 | 0.13566228 | 0.15368792 | 0.15130013 | 0.16014816 |
| Static | 5 | 0.02824895 | 0.05446257 | 0.10478793 | 0.09677542 | 0.1157781 | 0.11796117 | 0.12138253 | 0.11890557 | 0.12379145 |
| Dynamic | 0.02828131 | 0.01808326 | 0.01364867 | 0.01167391 | 0.01140917 | 0.01138711 | 0.01126096 | 0.0113994 | 0.01143201 |
| Guided | 0.02825478 | 0.05520535 | 0.10608214 | 0.13855987 | 0.16184 | 0.16326861 | 0.16286677 | 0.16311258 | 0.15887703 |
| Static | 10 | 0.02832496 | 0.0551269 | 0.1057336 | 0.10328684 | 0.1291147 | 0.12747518 | 0.13456278 | 0.12267753 | 0.12644179 |
| Dynamic | 0.02832414 | 0.02946289 | 0.03773387 | 0.0248159 | 0.02235605 | 0.02172862 | 0.02076766 | 0.02073111 | 0.02088737 |
| Guided | 0.02832073 | 0.05496409 | 0.10615151 | 0.13896206 | 0.16601674 | 0.15840394 | 0.16374366 | 0.15556381 | 0.16017388 |
| Static | 100 | 0.02826928 | 0.05415035 | 0.09165278 | 0.10961399 | 0.11379596 | 0.12086746 | 0.14717086 | 0.13036949 | 0.13209999 |
| Dynamic | 0.02800162 | 0.05437681 | 0.0971723 | 0.11712926 | 0.12869917 | 0.1268299 | 0.1290275 | 0.12842167 | 0.12741767 |
| Guided | 0.02832583 | 0.05522514 | 0.1031765 | 0.13875365 | 0.16438897 | 0.16424284 | 0.15638838 | 0.15892644 | 0.13723166 |

# **Explanation**

As shown in Figure 1, dynamic scheduling has terrible performance when using chunk sizes 1, 5, and 10, with its execution time growing as the number of threads increases and converges after using eight threads. Especially when using chunk size 1, the converged execution time of the dynamic scheduling is longer than any other chunk sizes or scheduling methods, about five times longer than dynamic scheduling with chunk size 5 and 10 times longer than dynamic scheduling with chunk size 10. It is easy to find that the multiplying number of converged execution times is the same as the dividing number of chunk sizes. When the chunk size becomes five times larger, the converged execution time gets five times faster, and if the chunk size is ten times larger, the converged execution time is ten times faster. This is also applied to dynamic scheduling with a chunk size 100, which can also be found in Figure 2. The reason for the longer execution time of dynamic scheduling is that dynamic scheduling gets more communication overhead when the chunk size gets smaller.

Also, static scheduling and guided scheduling have almost similar performance. Static scheduling, with a specified chunk size, is cyclic decomposition, so it has good load balancing and low communication overhead. On the other hand, guided scheduling, which is dynamic scheduling starting with a large chunk size and decreasing to handle load imbalance, had the shortest execution time in every chunk size among other scheduling methods. The specified chunk size is the minimum number of the chunk size, but there was almost no impact on execution time, which can be found in Figure 2. Since guided has both the pros of static and dynamic scheduling, it has the shortest execution time.