Problem 2

소프트웨어학부 20204898 박소은

# Environment

* **Processor**: Intel(R) Core(TM) i7-1065G7 CPU @ 1.30GHz 1.50 GHz
* **Number of cores**: 4
* **RAM**: 16.0GB
* **OS**: Window 11 (64 bit)

# Tables and graphs

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| exec time | chunk size | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| static | 1 | 0.444291 | 0.246296 | 0.188144 | 0.177182 | 0.17452 | 0.18497 | 0.187023 | 0.182787 | 0.18554 |
| dynamic | 0.523303 | 0.472047 | 0.46261 | 0.486447 | 0.509843 | 0.504809 | 0.517833 | 0.528168 | 0.532738 |
| guided | 0.439086 | 0.22355 | 0.11241 | 0.078545 | 0.079472 | 0.090731 | 0.097595 | 0.117503 | 0.088084 |
| static | 5 | 0.435648 | 0.242498 | 0.155559 | 0.128669 | 0.149407 | 0.137178 | 0.141968 | 0.157498 | 0.14254 |
| dynamic | 0.450172 | 0.404378 | 0.335581 | 0.330263 | 0.340171 | 0.338306 | 0.335508 | 0.335836 | 0.352766 |
| guided | 0.446144 | 0.2154 | 0.109825 | 0.080536 | 0.085715 | 0.079979 | 0.087289 | 0.075839 | 0.09547 |
| static | 10 | 0.443584 | 0.241867 | 0.151993 | 0.14062 | 0.133576 | 0.132264 | 0.134204 | 0.126253 | 0.141576 |
| dynamic | 0.463189 | 0.363377 | 0.310272 | 0.285163 | 0.294232 | 0.295213 | 0.301459 | 0.332782 | 0.302244 |
| guided | 0.467248 | 0.224505 | 0.116302 | 0.089851 | 0.081728 | 0.094305 | 0.090735 | 0.100114 | 0.091296 |
| static | 100 | 0.449662 | 0.23416 | 0.174131 | 0.123938 | 0.126779 | 0.133901 | 0.137898 | 0.09752 | 0.082855 |
| dynamic | 0.445432 | 0.192383 | 0.106857 | 0.080091 | 0.090317 | 0.097189 | 0.105372 | 0.097958 | 0.084178 |
| guided | 0.450153 | 0.224617 | 0.110209 | 0.080666 | 0.08254 | 0.078047 | 0.075459 | 0.107674 | 0.099522 |

## Execution time

## Performance

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| performance | chunk size | 1 | 2 | 4 | 6 | 8 | 10 | 12 | 14 | 16 |
| static | 1 | 2.250777 | 4.060155 | 5.315078 | 5.643914 | 5.730002 | 5.406282 | 5.346936 | 5.470849 | 5.389673 |
| dynamic | 1.910939 | 2.118433 | 2.161648 | 2.055722 | 1.961388 | 1.980947 | 1.931125 | 1.893337 | 1.877095 |
| guided | 2.277458 | 4.473272 | 8.896006 | 12.73156 | 12.58305 | 11.02159 | 10.24643 | 8.510421 | 11.3528 |
| static | 5 | 2.295431 | 4.123745 | 6.428429 | 7.77188 | 6.693127 | 7.289799 | 7.043841 | 6.349287 | 7.015575 |
| dynamic | 2.221373 | 2.472934 | 2.979906 | 3.02789 | 2.939698 | 2.955904 | 2.980555 | 2.977644 | 2.83474 |
| guided | 2.241429 | 4.642526 | 9.105395 | 12.41681 | 11.66657 | 12.50328 | 11.4562 | 13.18583 | 10.47449 |
| static | 10 | 2.254364 | 4.134504 | 6.57925 | 7.111364 | 7.486375 | 7.560636 | 7.451343 | 7.920604 | 7.063344 |
| dynamic | 2.158946 | 2.751963 | 3.222979 | 3.506766 | 3.398679 | 3.387385 | 3.317201 | 3.00497 | 3.308585 |
| guided | 2.140191 | 4.454244 | 8.598304 | 11.12954 | 12.23571 | 10.60389 | 11.02111 | 9.988613 | 10.95338 |
| static | 100 | 2.223893 | 4.270584 | 5.742803 | 8.06855 | 7.887742 | 7.468204 | 7.251737 | 10.25431 | 12.06928 |
| dynamic | 2.245012 | 5.197964 | 9.358301 | 12.4858 | 11.07211 | 10.28923 | 9.490187 | 10.20846 | 11.87959 |
| guided | 2.221467 | 4.452023 | 9.073669 | 12.3968 | 12.11534 | 12.81279 | 13.25223 | 9.287293 | 10.04803 |

## Chunk size: 1

## Chunk size: 5

## Chunk size: 10

## Chunk size: 100

# Explanation / Analysis

Regardless of the chunk size or scheduling type, it can be seen that when the number of threads is one, similar performance comes out.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

In the case of static scheduling, the larger the chunk size, the higher the performance from 5.38 (chunk size: 1) to 12.06 (chunk size: 100). When the chunk size is small, one thread is responsible for only one for statement, so the switching time of the threads was longer than the execution time of the for statement, resulting in lower performance. In addition, compared to other scheduling types, static scheduling has the second highest performance after guided.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 타이포그래피이(가) 표시된 사진

자동 생성된 설명

In the case of dynamic scheduling, the performance improved as the chunk size increased. When the chunk size was 1, the performance was 1.87, and when the chunk size was increased to 100, it was confirmed that the performance increased to 11.87. This is because if the chunk size is small, there will be more thread switching. In addition, in the case of dynamic, it can be confirmed that the performance is worse than other scheduling types because even runtime overhead occurs.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

Guided scheduling performed the best among scheduling types. Guided allocates tasks if there is a valid thread like dynamic, but as the program progresses, each thread is assigned a task of a constant reduced chunk size. Since the chunk size decreases as the task is repeatedly assigned, it shows more effective performance.

텍스트, 폰트, 스크린샷, 화이트이(가) 표시된 사진

자동 생성된 설명텍스트, 폰트, 스크린샷, 블랙이(가) 표시된 사진

자동 생성된 설명