CSCE 735

Parallel Processing

HW #1

By Sorush Omidvar

Under Dr. Sarin

Fall 2021

# Part A

1. Finding the execution time with different number of threads for n = 108.

Please see below for the performance of the application based on various number of threads. The best execution time belongs to P=128 which means the execution time drops with the increase of thread number in the beginning and then (after P=128) starts to increase again.

|  |  |  |  |
| --- | --- | --- | --- |
| P | Time | Speed Up (S) | Efficiency (E) |
| 1 | 1.2868 | 1 | 1 |
| 2 | 0.6478 | 1.98641556 | 0.99320778 |
| 4 | 0.3266 | 3.939987753 | 0.984996938 |
| 8 | 0.1667 | 7.719256149 | 0.964907019 |
| 16 | 0.0867 | 14.84198385 | 0.927623991 |
| 32 | 0.0465 | 27.67311828 | 0.864784946 |
| 64 | 0.042 | 30.63809524 | 0.478720238 |
| 128 | 0.0369 | 34.87262873 | 0.272442412 |
| 256 | 0.0378 | 34.04232804 | 0.132977844 |
| 512 | 0.0385 | 33.42337662 | 0.065280032 |
| 1024 | 0.0419 | 30.71121718 | 0.029991423 |
| 2048 | 0.0679 | 18.95139912 | 0.009253613 |
| 4096 | 0.1306 | 9.852986217 | 0.002405514 |
| 8192 | 0.2703 | 4.76063633 | 0.000581132 |

1. Finding the execution time with different number of threads for n = 1010.

Please see below for the performance of the application based on a various number of threads. The best execution time belongs to P=512, which means the execution time drops with the increase of thread number in the beginning and then (after P=512) starts to increase again. The main reason for such an increase should be found in the extra surcharge of task scheduling and communication between threads. The Poptimum is more than part 1 mainly because the task is harder to do and requires more calculation. Therefore, the optimum value of p, which is dictated by the task-scheduling/communication and calculation cost tradeoff, moves toward a bigger value of P.

|  |  |  |  |
| --- | --- | --- | --- |
| P | Time | Speed Up (S) | Efficiency (E) |
| 1 | 18.056 | 1 | 1 |
| 2 | 9.0507 | 1.994983813 | 0.997491907 |
| 4 | 4.5285 | 3.987192227 | 0.996798057 |
| 8 | 2.2681 | 7.960848287 | 0.995106036 |
| 16 | 1.1392 | 15.8497191 | 0.990607444 |
| 32 | 0.5778 | 31.24956732 | 0.976548979 |
| 64 | 0.4368 | 41.33699634 | 0.645890568 |
| 128 | 0.4054 | 44.53872718 | 0.347958806 |
| 256 | 0.3948 | 45.73454914 | 0.178650583 |
| 512 | 0.3897 | 46.33307673 | 0.09049429 |
| 1024 | 0.3928 | 45.96741344 | 0.044890052 |
| 2048 | 0.4076 | 44.2983317 | 0.021630045 |
| 4096 | 0.4345 | 41.55581128 | 0.010145462 |
| 8192 | 0.4948 | 36.49151172 | 0.00445453 |

1. As mentioned in the previous part, there is a tradeoff between task scheduling and computation which in this case depends on the number of n. Therefore, as n increases, we expect to see higher P for the optimum performance. Such a change was observed in my experiment where the optimum P changes from 128 to 512.
2. Doing this experiment proves that by increasing the number of n we will have a lower error which is expected considering the mathematics behind the question.

|  |  |
| --- | --- |
| n | Error |
| 1000 | 4.76E-02 |
| 10000 | 1.01E-02 |
| 100000 | 4.29E-03 |
| 1000000 | 1.21E-03 |
| 10000000 | 1.29E-03 |
| 100000000 | 2.08E-04 |
| 1000000000 | 8.04E-06 |

# Part B

1. Execution performance as a function of process number when n=108. The best performance can be seen when the process number of p equals 8. So, the run time decreases as P increases till P reaches 8 and then starts to increase again.

|  |  |  |  |
| --- | --- | --- | --- |
| P | Time | Speed Up (S) | Efficiency (E) |
| 1 | 0.0764 | 1 | 1 |
| 2 | 0.0401 | 1.905236908 | 0.952618454 |
| 4 | 0.0241 | 3.170124481 | 0.79253112 |
| 8 | 0.0204 | 3.745098039 | 0.468137255 |
| 16 | 0.0257 | 2.972762646 | 0.185797665 |
| 32 | 0.0325 | 2.350769231 | 0.073461538 |
| 64 | 0.0402 | 1.900497512 | 0.029695274 |

1. When n=1010 and p=64, we can run an experiment to see the effect of ntasks-per-node on the run time. Based on the results, the more core per task we have, the faster the program will run. This actually makes sense as the ntasks-per-node is the number of CPU in each motherboard available for our job. So, for ntasks-per-node=48, we have the best result and we cannot go any further as the GRACE nodes have only 48 cores.

|  |  |
| --- | --- |
| core per task | Time |
| 1 | 0.0993 |
| 2 | 0.1105 |
| 4 | 0.0244 |
| 8 | 0.0166 |
| 16 | 0.0048 |
| 32 | 0.0023 |
| 48 | 0.0023 |

1. For cases with p =64 and n=10k where k=2, 4, 6 and 8 we get the following execution time and error which suggests that by increasing the number of processes the execution time gets shorten if and only if there is enough task. In another word, similar to the previous section, there is a tradeoff between task scheduling and communication versus computation. Therefore, it makes sense to assign more processes to a more complicated task. Again, similar to part A, by increasing the number of n we can get a better estimation which results in the reduction of error.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| n | p | Error | Exe Time | Speed Up |
| 100 | 64 | 2.65E-06 | 0.029 | 0.003448 |
| 100 | 1 | 2.65E-06 | 0.0001 |  |
| 1000 | 64 | 2.65E-08 | 0.0043 | 0.023256 |
| 1000 | 1 | 2.65E-08 | 0.0001 |  |
| 10000 | 64 | 2.65E-10 | 0.003 | 0.033333 |
| 10000 | 1 | 2.65E-10 | 0.0001 |  |
| 100000 | 64 | 2.65E-12 | 0.0351 | 0.002849 |
| 100000 | 1 | 2.65E-12 | 0.0001 |  |
| 1000000 | 64 | 2.63E-14 | 0.0331 | 0.024169 |
| 1000000 | 1 | 3.51E-14 | 0.0008 |  |
| 10000000 | 64 | 2.83E-16 | 0.0326 | 0.233129 |
| 10000000 | 1 | 3.41E-14 | 0.0076 |  |
| 100000000 | 64 | 4.24E-16 | 0.0334 | 2.254491 |
| 100000000 | 1 | 1.35E-13 | 0.0753 |  |