Assignment No. 1

CS476-01\_SPG14

Type of CPU: 4-core

* 10240 sized Square Matrix addition using Static Work Division

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cilk Workers | Splits | Running Time (s) | Work (sec) | Span (sec) |
| 2 | 2 | 2.343311 | 3.627817 | 1.883090 |
| 2 | 4 | 2.838697 | 4.124814 | 2.367377 |
| 2 | 8 | 4.065786 | 5.382235 | 3.584843 |
| 4 | 2 | 2.756937 | 5.211218 | 2.756601 |
| 4 | 4 | 3.376593 | 5.834673 | 3.376389 |
| 4 | 8 | 5.040445 | 7.511002 | 5.040048 |
| 8 | 2 | 3.697928 | 7.582701 | 3.696910 |
| 8 | 4 | 4.466459 | 7.533300 | 4.460053 |
| 8 | 8 | 6.011868 | 9.074766 | 6.007690 |

* 10240 sized Square Matrix addition using Recursive Divide and Conquer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cilk Workers | Splits | Running Time (s) | Work (sec) | Span (sec) |
| 2 | 2 | 2.271900 | 3.975793 | 1.410358 |
| 2 | 4 | 2.200768 | 3.907629 | 1.385264 |
| 2 | 8 | 2.227816 | 3.969477 | 1.324053 |
| 4 | 2 | 2.147496 | 5.480157 | 1.874084 |
| 4 | 4 | 2.122275 | 5.432798 | 1.843671 |
| 4 | 8 | 2.154942 | 5.478157 | 1.884223 |
| 8 | 2 | 3.405458 | 8.427035 | 3.172163 |
| 8 | 4 | 3.051964 | 7.746084 | 2.847504 |
| 8 | 8 | 3.194836 | 7.959859 | 3.034933 |

* 1000 sized Square Matrix multiplication using Static Work Division

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cilk Workers | Splits | Running Time (s) | Work (sec) | Span (sec) |
| 2 | 2 | 50.625950 | 50.666959 | 50.623628 |
| 2 | 4 | 111.462931 | 111.473167 | 111.431978 |
| 2 | 8 | 189.881500 | 189.888026 | 189.846345 |
| 4 | 2 | 62.680738 | 62.744615 | 62.613742 |
| 4 | 4 | 147.070748 | 147.127749 | 147.065432 |
| 4 | 8 | 211.990465 | 212.016243 | 211.968223 |
| 8 | 2 | 64.084532 | 64.130868 | 64.076621 |
| 8 | 4 | 124.526744 | 124.591427 | 124.496117 |
| 8 | 8 | 354.517914 | 354.548252 | 354.477441 |

* 1500 sized Square Matrix multiplication using recursive Divide and Conquer

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Cilk Workers | Splits | Running Time (s) | Work (sec) | Span (sec) |
| 2 | 2 | 92.183296 | 184.119632 | 0.720758 |
| 2 | 4 | 108.795938 | 217.040290 | 1.470940 |
| 2 | 8 | 109.288790 | 217.248298 | 2.977345 |
| 4 | 2 | 52.803052 | 210.370557 | 1.002721 |
| 4 | 4 | 90.569243 | 360.010729 | 2.606683 |
| 4 | 8 | 101.771609 | 401.925402 | 5.156104 |
| 8 | 2 | 59.999270 | 475.715357 | 1.809310 |
| 8 | 4 | 52.880774 | 415.860868 | 2.528491 |
| 8 | 8 | 87.162958 | 677.334593 | 7.382017 |

**Analysis:**

Matrix addition using static work division:

* For running time, with constant cilk workers and increasing split size, the running time gets increased.
* Increasing more cilk workers does not help in reducing the running time.
* As cilk workers and splits increase, specifically at 4 cilk workers and 2 splits, the difference between running time and span gets negligible.
* The running time was always better than the work time or the time it would take if the work was done sequentially.

Matrix addition using recursive Divide & Conquer:

* Increasing the workers is not helping the running time considerably.
* For all number of workers tested, the best running time was when it had 4 splits.
* The running time was always better than the work time or the time it would take if the work was done sequentially.
* When we increase the workers, the difference between the running time and the span gets reduced.

Matrix multiplication with static work division:

* For running time, with constant cilk workers and increasing split size, the running time gets increased.
* For all number of workers tested, the best running time was when it had 2 splits.
* The difference between the running time, work, and span was minimal for all of the test cases.
* From the running times we observed for 1000 size matrix, the time to observe a 10240 size matrix would have been huge.

Matrix multiplication using recursive Divide & Conquer:

* For running time, with constant cilk workers and increasing split size, the running time gets increased.
* For cilk workers 2 & 4, the best running time is obtained at 2 splits, whereas with 8 cilk workers, the best time achieved was at 4 splits.
* The running time was always considerably better than the work time or the time it would take if the work was done sequentially.
* The difference between the running time and span was considerably large which shows that there is still some room for improvement.