Project 1 Report CS 475 Jacob Leno

1. All tests were ran on OSU’s flip server (flip2), Load average at time of tests: 2.10, 2.08, 1.84
2. As the number of nodes in the test increase the actual volume seems to be approaching **25.3125**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Total Time per X Nodes (in Seconds)** | | | | | | | | | |
|  |  | **Nodes** | | | | | | | |
|  |  | 100 | 500 | 1000 | 2000 | 5000 | 7500 | 10000 | 15000 |
| **Threads** | 1 | 0.0007678 | 0.0189589 | 0.0753287 | 0.3030982 | 1.8905377 | 4.2109385 | 7.5706165 | 16.9835941 |
| 2 | 0.0004674 | 0.0096632 | 0.0381476 | 0.1528458 | 0.9563699 | 2.1496600 | 3.8100713 | 8.5229272 |
| 4 | 0.0003251 | 0.0051329 | 0.0197251 | 0.0783633 | 0.4887746 | 1.0973176 | 1.9535194 | 4.4010615 |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **MegaHeights per Second** | | | | | | | | | |
|  |  | **Nodes** | | | | | | | |
|  |  | 100 | 500 | 1000 | 2000 | 5000 | 7500 | 10000 | 15000 |
| **Threads** | 1 | 13023.77 | 13186.41 | 13275.16 | 13197.04 | 13223.75 | 13358.07 | 13208.96 | 13248.08 |
| 2 | 21397.38 | 25871.31 | 26213.96 | 26170.16 | 26140.51 | 26166.93 | 26246.23 | 26399.38 |
| 4 | 30760.95 | 48705.02 | 50696.73 | 51044.27 | 51148.32 | 51261.37 | 51189.66 | 51124.03 |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Nodes** | | | | | | | |
|  | 100 | 500 | 1000 | 2000 | 5000 | 7500 | 10000 | 15000 |
| **Volume** | 25.313303 | 25.312531 | 25.312509 | 25.312503 | 25.312500 | 25.312500 | 25.312501 | 25.312499 |

1. The MegaHeights per second computed starts out with smaller values for smaller numbers of nodes and then levels off as the number of nodes increase. However the total time per x nodes decreases steadily as the number of threads increases. Total time per x nodes also increases as would be expected when the number of nodes increases.
2. I think the MegaHeights per second works this way because the overhead associated with using OpenMP causes it not to reach optimal efficiency until there are more values to iterate over. The total time per x nodes decreases as the number of threads increase because there are more threads to iterate over the for loop
3. Inverse Amdahl’s Law:
4. max *Speedup* =