Braden Ackles

CS 475

Project 2 Write UP

4/24/2016

N-Body Calculation

The Machine I ran this on was Flip, I ran very large data sets and collected A LOT of data. When I kicked off each of the 4 tests flip was sitting around 5-6% usage but each test took about 10 minutes to run so in that time the usage could have changed.

NOTE: I spent a solid 30 minutes trying to get the major information on to one graph in a way that made sense. I understand I was supposed to put it all on one graph but I failed to do so, so I apologize in advance. All my data and graphs are included in the zip folder as well.

Course Static: MegaCalcs Per Second

In this example we see what makes sense for speed up the more threads you use the more calculations per second you have. Pretty Straight forward. We see a sweet spot for reasonable number of threads at 4 threads. Then a decrease in the efficiency and then an steady slow increase all the way to 10 threads and the amount of megaCalcs per second.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 35.15 | 35.1733 | 35.1534 | 35.1854 | 35.0476 | 35.1275 | 35.1488 | 34.968 | 35.1445 | 35.0538 |
| 2 | 58.451 | 56.0091 | 57.7536 | 58.6724 | 58.4436 | 57.6198 | 57.8811 | 56.9902 | 58.1735 | 58.2524 |
| 3 | 97.5027 | 97.5639 | 97.5593 | 98.1819 | 97.7317 | 97.4647 | 98.6001 | 96.7709 | 97.5904 | 97.8925 |
| 4 | 119.63 | 122.49 | 120.069 | 122.063 | 122.462 | 120.73 | 118.357 | 122.165 | 122.328 | 122.136 |
| 5 | 86.2803 | 86.5104 | 86.3443 | 86.2977 | 86.3569 | 86.3531 | 86.3019 | 85.0367 | 86.2805 | 86.1479 |
| 6 | 103.312 | 103.996 | 103.495 | 103.796 | 103.848 | 103.772 | 103.895 | 103.697 | 103.378 | 104.085 |
| 7 | 110.395 | 110.767 | 111.023 | 111.302 | 111.132 | 111.204 | 111.201 | 111.124 | 108.889 | 105.483 |
| 8 | 125.441 | 125.979 | 126.358 | 126.263 | 126.126 | 126.051 | 126.414 | 126.338 | 126.078 | 126.33 |
| 9 | 136.597 | 129.964 | 136.859 | 137.94 | 137.454 | 137.768 | 137.495 | 136.77 | 137.896 | 137.803 |
| 10 | 149.567 | 150.34 | 150.36 | 150.226 | 150.893 | 150.334 | 149.982 | 150.471 | 150.519 | 150.809 |

Course Static: Time to Calculate

Still on the same example as last time the more threads we use the smaller amount of time it takes although there is diminishing returns as we get to higher and higher numbers. Also at 5 threads we see an increase in the amount of time taken to calculate the information. This could be due to an increase on demand on the server, poor optimizations or the way that static broke the chunks up between the Threads.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 28.4495 | 56.8613 | 85.3403 | 113.684 | 142.663 | 170.806 | 199.153 | 228.78 | 256.086 | 285.276 |
| 2 | 17.1084 | 35.7085 | 51.9448 | 68.1752 | 85.5526 | 104.131 | 120.938 | 140.375 | 154.71 | 171.667 |
| 3 | 10.2561 | 20.4994 | 30.7505 | 40.7407 | 51.1605 | 61.5607 | 70.9938 | 82.6695 | 92.2222 | 102.153 |
| 4 | 8.3591 | 16.3279 | 24.9856 | 32.7699 | 40.8289 | 49.6978 | 59.1432 | 65.4854 | 73.5724 | 81.8762 |
| 5 | 11.5901 | 23.1186 | 34.7446 | 46.3512 | 57.8992 | 69.4822 | 81.1106 | 94.077 | 104.311 | 116.079 |
| 6 | 9.67941 | 19.2315 | 28.987 | 38.5373 | 48.1471 | 57.819 | 67.3757 | 77.148 | 87.0594 | 96.0757 |
| 7 | 9.0584 | 18.056 | 27.0214 | 35.9384 | 44.9916 | 53.9551 | 62.9492 | 71.9915 | 82.653 | 94.802 |
| 8 | 7.97187 | 15.8756 | 23.7421 | 31.68 | 39.6429 | 47.5997 | 55.3736 | 63.3223 | 71.3842 | 79.1579 |
| 9 | 7.32079 | 15.3889 | 21.9204 | 28.9982 | 36.3757 | 43.5516 | 50.9109 | 58.4925 | 65.2667 | 72.5674 |
| 10 | 6.68598 | 13.3032 | 19.9522 | 26.6265 | 33.136 | 39.9111 | 46.6721 | 53.1663 | 59.7931 | 66.3091 |

Course Dynamic: MegaCalcs Per second

Much like the last example of course static, dynamic sees an increase in the amount of data calculated per second the more cores used. However instead of 5 threads being a drop in performance it was actually an increase in performance this could be attributed to dynamic re-assigning tasks to different threads if a task finished.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 34.8433 | 34.8482 | 34.8621 | 34.8617 | 34.8216 | 34.8588 | 34.8345 | 34.8608 | 34.8586 | 34.8437 |
| 2 | 56.7153 | 56.7261 | 56.1154 | 56.5784 | 57.0803 | 56.193 | 56.7924 | 56.2865 | 56.2725 | 56.4483 |
| 3 | 93.8319 | 93.0466 | 95.3113 | 95.7438 | 95.521 | 95.8809 | 96.0782 | 94.6172 | 95.963 | 95.1911 |
| 4 | 91.6725 | 97.3566 | 91.3984 | 92.9207 | 96.7 | 97.7602 | 95.6292 | 97.9118 | 96.3449 | 97.2162 |
| 5 | 123.075 | 121.076 | 125.337 | 124.174 | 125.553 | 124.61 | 124.565 | 123.877 | 123.654 | 125.244 |
| 6 | 105.83 | 105.28 | 107.316 | 107.559 | 108.842 | 109.779 | 109.624 | 108.269 | 109.099 | 109.04 |
| 7 | 121.36 | 121.246 | 119.992 | 123.617 | 124.131 | 123.022 | 122.257 | 124.009 | 124.047 | 122.638 |
| 8 | 129.723 | 128.533 | 127.026 | 129.076 | 129.276 | 129.306 | 128.602 | 129.52 | 127.87 | 128.045 |
| 9 | 133.603 | 134.353 | 135.571 | 134.252 | 134.479 | 134.78 | 135.324 | 135.122 | 134.231 | 133.623 |
| 10 | 136.723 | 136.421 | 135.198 | 134.724 | 136.91 | 136.797 | 136.844 | 136.481 | 136.896 | 137.127 |

Course Dynamic: Time to Calculate

This is just another way to look at the number of calculations per second but instead of examining how many per second it takes the problem as a whole and sees how long it took to complete the entire task. Just like the mega calcs per second the more threads used the faster the performance but there were diminishing returns after around 7 threads.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 28.7 | 57.3918 | 86.0533 | 114.739 | 143.589 | 172.123 | 200.95 | 229.484 | 258.186 | 286.996 |
| 2 | 17.6319 | 35.2571 | 53.4613 | 70.6983 | 87.5959 | 106.775 | 123.256 | 142.13 | 159.936 | 177.153 |
| 3 | 10.6574 | 21.4946 | 31.4758 | 41.7782 | 52.3445 | 62.5776 | 72.8573 | 84.5512 | 93.7862 | 105.052 |
| 4 | 10.9084 | 20.543 | 32.8233 | 43.0475 | 51.7063 | 61.3747 | 73.1994 | 81.7062 | 93.4144 | 102.863 |
| 5 | 8.1251 | 16.5186 | 23.9355 | 32.2128 | 39.8238 | 48.1502 | 56.1957 | 64.5804 | 72.7839 | 79.8439 |
| 6 | 9.44915 | 18.997 | 27.9549 | 37.189 | 45.9383 | 54.6554 | 63.8549 | 73.8903 | 82.4936 | 91.7093 |
| 7 | 8.23996 | 16.4954 | 25.0017 | 32.3579 | 40.28 | 48.7718 | 57.2565 | 64.5114 | 72.5533 | 81.5405 |
| 8 | 7.70874 | 15.5602 | 23.6172 | 30.9896 | 38.677 | 46.4017 | 54.4315 | 61.7666 | 70.3842 | 78.0978 |
| 9 | 7.48487 | 14.8862 | 22.1287 | 29.7948 | 37.1804 | 44.5171 | 51.7277 | 59.2058 | 67.0488 | 74.8372 |
| 10 | 7.31405 | 14.6605 | 22.1897 | 29.6904 | 36.5204 | 43.8606 | 51.1533 | 58.6161 | 65.7431 | 72.9251 |

Fine Static: MegaCalcs Per Second

Fine in general acted much different than Course. It would have an increase to around 3 or 4 threads and then a sharp decrease this could be attributed to the overhead of assigning so many small tasks to threads or to the fact that the problem was split up in to so many little problems. Because Unlike coarse which split up the problem in to NUMT big problems fine splits the problem in too many small problems. This difference is made by where the pragma is place either inside the outer for loop or inside the inner for loop.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 31.9796 | 32.0021 | 31.9593 | 32.0201 | 32.0205 | 31.8858 | 31.9764 | 31.8739 | 32.0182 | 31.95 |
| 2 | 44.59 | 44.9655 | 44.7991 | 44.7966 | 44.6904 | 44.7119 | 44.5718 | 44.8223 | 44.7804 | 44.7317 |
| 3 | 48.9393 | 49.0081 | 49.227 | 49.2046 | 49.0273 | 49.1081 | 49.2364 | 49.0504 | 49.2903 | 49.3146 |
| 4 | 46.1268 | 46.2683 | 46.1794 | 46.191 | 46.2717 | 46.2458 | 46.2767 | 46.1302 | 46.1047 | 46.1422 |
| 5 | 43.3065 | 43.631 | 43.59 | 43.4621 | 40.1713 | 26.6209 | 26.9375 | 26.6758 | 26.4178 | 27.0214 |
| 6 | 26.1679 | 26.672 | 26.5831 | 26.4538 | 26.596 | 26.5366 | 26.646 | 26.6235 | 26.7623 | 26.7366 |
| 7 | 22.2531 | 22.449 | 22.3345 | 22.6425 | 22.4958 | 22.5783 | 22.6669 | 22.656 | 22.6211 | 22.5709 |
| 8 | 18.7647 | 18.7856 | 18.7815 | 18.7937 | 18.7485 | 18.7818 | 18.7827 | 18.7447 | 18.5998 | 18.7387 |
| 9 | 11.0644 | 10.9779 | 11.0333 | 11.0786 | 11.0684 | 11.7529 | 11.7625 | 11.7291 | 11.8066 | 11.7609 |
| 10 | 12.9125 | 13.0049 | 12.9558 | 13.016 | 13.0214 | 13.0141 | 13.0117 | 13.0124 | 13.0198 | 12.9872 |

Fine Static: Time to Calculate

Just like the last graph this data is from the same sample which shows that as we use more threads the trend shows that it takes more time to calculate just like as it has fewer mega calcs per second. The weird split at around 5 threads is probably attributed to a temporary increase on CPU demand. The main thing to take away from this is Fine Static parallelism was effective till 3-4 threads after that it caused a steady decrease in the effects of parallelism.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 31.27 | 62.4958 | 93.8693 | 124.921 | 156.15 | 188.171 | 218.912 | 250.989 | 281.09 | 312.989 |
| 2 | 22.4266 | 44.4786 | 66.9656 | 89.2925 | 111.881 | 134.192 | 157.05 | 178.482 | 200.981 | 223.555 |
| 3 | 20.4335 | 40.8096 | 60.9422 | 81.2932 | 101.984 | 122.179 | 142.171 | 163.097 | 182.592 | 202.78 |
| 4 | 21.6794 | 43.2262 | 64.9641 | 86.5969 | 108.057 | 129.742 | 151.264 | 173.422 | 195.208 | 216.722 |
| 5 | 23.0912 | 45.8389 | 68.8232 | 92.0343 | 124.467 | 225.387 | 259.861 | 299.897 | 340.679 | 370.077 |
| 6 | 38.2148 | 74.9851 | 112.854 | 151.207 | 187.998 | 226.103 | 262.704 | 300.486 | 336.295 | 374.02 |
| 7 | 44.9376 | 89.091 | 134.321 | 176.659 | 222.264 | 265.742 | 308.82 | 353.108 | 397.858 | 443.049 |
| 8 | 53.2915 | 106.465 | 159.732 | 212.837 | 266.688 | 319.458 | 372.684 | 426.787 | 483.875 | 533.656 |
| 9 | 90.3797 | 182.184 | 271.904 | 361.055 | 451.735 | 510.511 | 595.11 | 682.062 | 762.286 | 850.278 |
| 10 | 77.4442 | 153.789 | 231.557 | 307.315 | 383.982 | 461.037 | 537.979 | 614.8 | 691.254 | 769.987 |

Fine Dynamic: MegaCalcs Per Second

In this case of parallelism, the fewer the threads used the more MegaCalcs per Second. This actually is very confusing to me and doesn’t make sense I figured fine dynamic would have been more efficient then Fine static because it can move calculations between threads. That being said it does make sense that a few number of threads would be more efficient. Which is true, the weird graph is most likely attributed to an increase in cpu demand for a short period of time.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 18.6585 | 18.6721 | 18.668 | 18.6637 | 18.6622 | 18.7165 | 18.6966 | 18.7036 | 18.6785 | 18.6688 |
| 2 | 10.1255 | 10.0596 | 10.1273 | 10.0664 | 9.98302 | 9.95607 | 10.0031 | 16.585 | 21.3795 | 21.3962 |
| 3 | 19.7881 | 19.7414 | 19.6532 | 19.7012 | 19.7613 | 12.0544 | 10.2543 | 10.1945 | 10.2468 | 11.0494 |
| 4 | 9.66462 | 9.69733 | 9.69598 | 9.69238 | 9.70453 | 9.7019 | 9.721 | 9.70015 | 9.68768 | 9.70635 |
| 5 | 9.86339 | 9.88255 | 9.94433 | 9.92747 | 9.85857 | 9.7967 | 9.40078 | 9.26149 | 9.22162 | 9.22095 |
| 6 | 7.65678 | 7.66148 | 7.66216 | 7.71878 | 7.72482 | 7.67255 | 7.6784 | 7.67364 | 7.67571 | 7.67131 |
| 7 | 10.7608 | 10.7584 | 10.7385 | 10.7583 | 10.7544 | 10.6822 | 10.4566 | 10.4654 | 10.4576 | 10.4684 |
| 8 | 9.74103 | 9.76013 | 9.72115 | 9.76275 | 9.76726 | 9.80705 | 9.83025 | 9.77808 | 9.76515 | 9.78163 |
| 9 | 10.4305 | 10.4395 | 10.4222 | 10.4399 | 10.4153 | 10.0375 | 9.99659 | 10.0094 | 9.99188 | 9.98599 |
| 10 | 7.96299 | 7.99326 | 7.98393 | 7.99139 | 7.98112 | 7.99109 | 7.98227 | 7.97878 | 7.98995 | 7.97587 |

Fine Dynamic: Time to Calculate

Just like the megaCalcs per second graph, it was faster with fewer threads and then slowed down as more cores were added. This behavior is confusing to me. Although it would make sense that as more threads are added that it would slow down because more overhead is added to assigning tasks to the threads.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Threads** | **10000** | **20000** | **30000** | **40000** | **50000** | **60000** | **70000** | **80000** | **90000** | **100000** |
| 1 | 53.5947 | 107.112 | 160.703 | 214.319 | 267.922 | 320.573 | 374.4 | 427.726 | 481.838 | 535.654 |
| 2 | 98.7608 | 198.816 | 296.229 | 397.362 | 500.85 | 602.647 | 699.781 | 482.364 | 420.964 | 467.372 |
| 3 | 50.5354 | 101.31 | 152.647 | 203.034 | 253.02 | 497.742 | 682.64 | 784.739 | 878.323 | 905.024 |
| 4 | 103.47 | 206.242 | 309.407 | 412.695 | 515.223 | 618.436 | 720.09 | 824.729 | 929.015 | 1030.25 |
| 5 | 101.385 | 202.377 | 301.68 | 402.923 | 507.173 | 612.451 | 744.619 | 863.792 | 975.967 | 1084.49 |
| 6 | 130.603 | 261.046 | 391.535 | 518.217 | 647.264 | 782.008 | 911.648 | 1042.53 | 1172.53 | 1303.56 |
| 7 | 92.9299 | 185.901 | 279.368 | 371.804 | 464.924 | 561.684 | 669.434 | 764.426 | 860.616 | 955.252 |
| 8 | 102.659 | 204.915 | 308.606 | 409.721 | 511.914 | 611.804 | 712.088 | 818.157 | 921.645 | 1022.32 |
| 9 | 95.8727 | 191.58 | 287.848 | 383.146 | 480.064 | 597.757 | 700.239 | 799.252 | 900.731 | 1001.4 |
| 10 | 125.581 | 250.211 | 375.755 | 500.539 | 626.478 | 750.836 | 876.944 | 1002.66 | 1126.42 | 1253.78 |