Andrew Alferman

CS 575 Spring 2017

Project #2

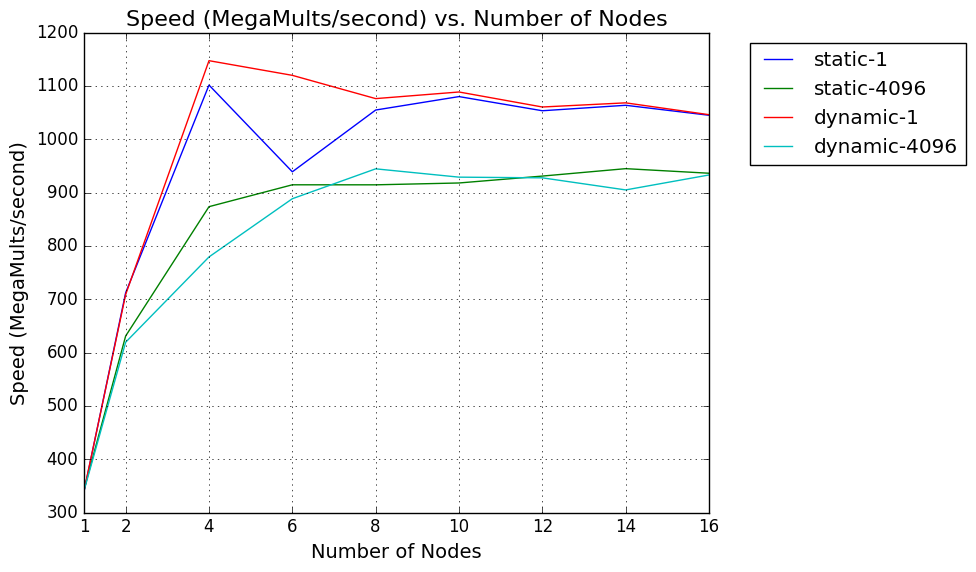
1. Tell what machine you ran this on.
   * The machine was a MacBook Pro (Retina, 13-inch, Early 2015) with a 2.9 GHz Intel Core i5 processor running macOS Sierra Version 10.12.3 with 8 GB 1867 MHz DDR3.
2. Create a table with your results.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Threads | static-1 | static-4096 | dynamic-1 | dynamic-4096 |
| 1 | 339.13 | 342.7 | 340.6 | 340.84 |
| 2 | 712.09 | 630.99 | 708.78 | 619.28 |
| 4 | 1101.46 | 873.43 | 1147.17 | 779.18 |
| 6 | 938.82 | 914.44 | 1119.67 | 888.36 |
| 8 | 1054.63 | 914.4 | 1076.04 | 944.37 |
| 10 | 1079.83 | 917.85 | 1088.46 | 928.78 |
| 12 | 1053.29 | 930.8 | 1060.31 | 927.5 |
| 14 | 1063.53 | 944.71 | 1068.1 | 904.83 |
| 16 | 1044.83 | 936.13 | 1045.91 | 932.87 |

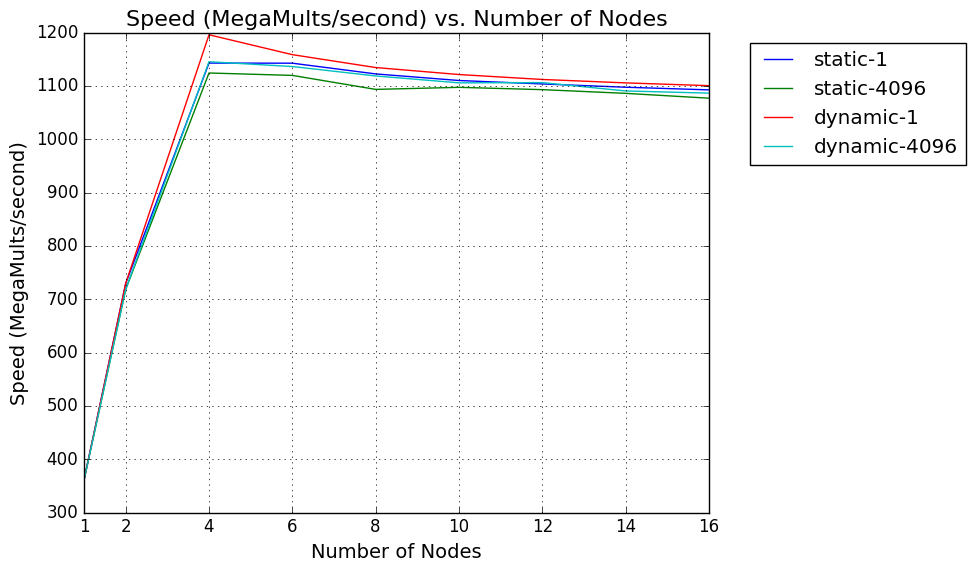
Note that the ARRAYSIZE was 32768 for the above results.

1. Draw a graph. The X axis will be the number of threads. The Y axis will be the performance in whatever units you sensibly choose. On the same graph, plot 4 curves:
   * static,1
   * static,4096
   * dynamic,1
   * dynamic,4096

Two such graphs were produced. Measurement of speed was in the recommended MegaMults/second. The first graph was produced for an ARRAYSIZE of 32768. The second graph was for a larger ARRAYSIZE of 262144. The larger size was produced to smooth out variability in performance from one run to the next.



ARRAYSIZE: 32768



ARRAYSIZE: 262144

1. What patterns are you seeing in the speeds?
   * Dynamic is faster than static, and a CHUNKSIZE of 1 is faster than a CHUNKSIZE of 4096. In the present example, the selection of CHUNKSIZE has a greater impact on performance than the selection of static vs. dynamic schedule. In many cases, with a CHUNKSIZE of 4096, it is unclear whether static or dynamic is faster. The speed is typically greatest with four threads, due to the use of a processor with four cores. The speed levels off or slightly declines with greater than four threads. A larger ARRAYSIZE results in a greater speed due to the increased parallelism.
2. Why does chunk size 1 vs. 4096 matter like this?
   * The number of computations needed to be completed is not consistent across the for loop. Because of this, when the for loop is being divided across however many threads, the threads are not given an even workload. In some cases, one thread may be given much more work than others, especially when as many as 4096 threads are assigned all at once. The uneven workload means that some threads may end up waiting for others to finish at times, which reduces parallelism and slows down the computation. When using a CHUNKSIZE of 1, this effect is not nearly as pronounced because the difference between one iteration of the for loop and the next one isn’t as large, thus the workload is more evenly distributed.
3. Why does static vs. dynamic matter like this?
   * Dynamic is slightly faster because it allows iterations of the for loop to be allocated while the computation is running. Thus, when an uneven workload exists, a thread doesn’t end up waiting for other threads to finish to the same degree as would happen if static were selected. With static allocation, the work is all divided up at the beginning of the computation, and the computer has no way of knowing just how much work will be subsequently assigned to each thread, which makes it more prone to distribute a more uneven workload.