Anakin Kinsey
CSCI 473 – HW08
12/6/2019

Matrix Vector multiplication across different matrix sizes and Processor counts

Matrix muliplication is a computationaly intense process that can be useful in showing the benefit of parallelizing over multiple processors. In general as the process count increases, the time taken to complete the matrix multiplication goes down as well. Matrix vector multiplication (MM) consists of Ax = y, a matrix * a vector equals a vector, and can be broken down into two parts. File IO and computation time. File IO is not parallizable and in our case consits of reading a matrix and a vector, M*N matrix and N*1 vector, and writing a vector, M*1 vector. The second portion of MM is the computation time. This consists of parallizable matrix multiplication. The reduced time from parallel computation time portion of the MM is what represents the said benefits of parallel computation.

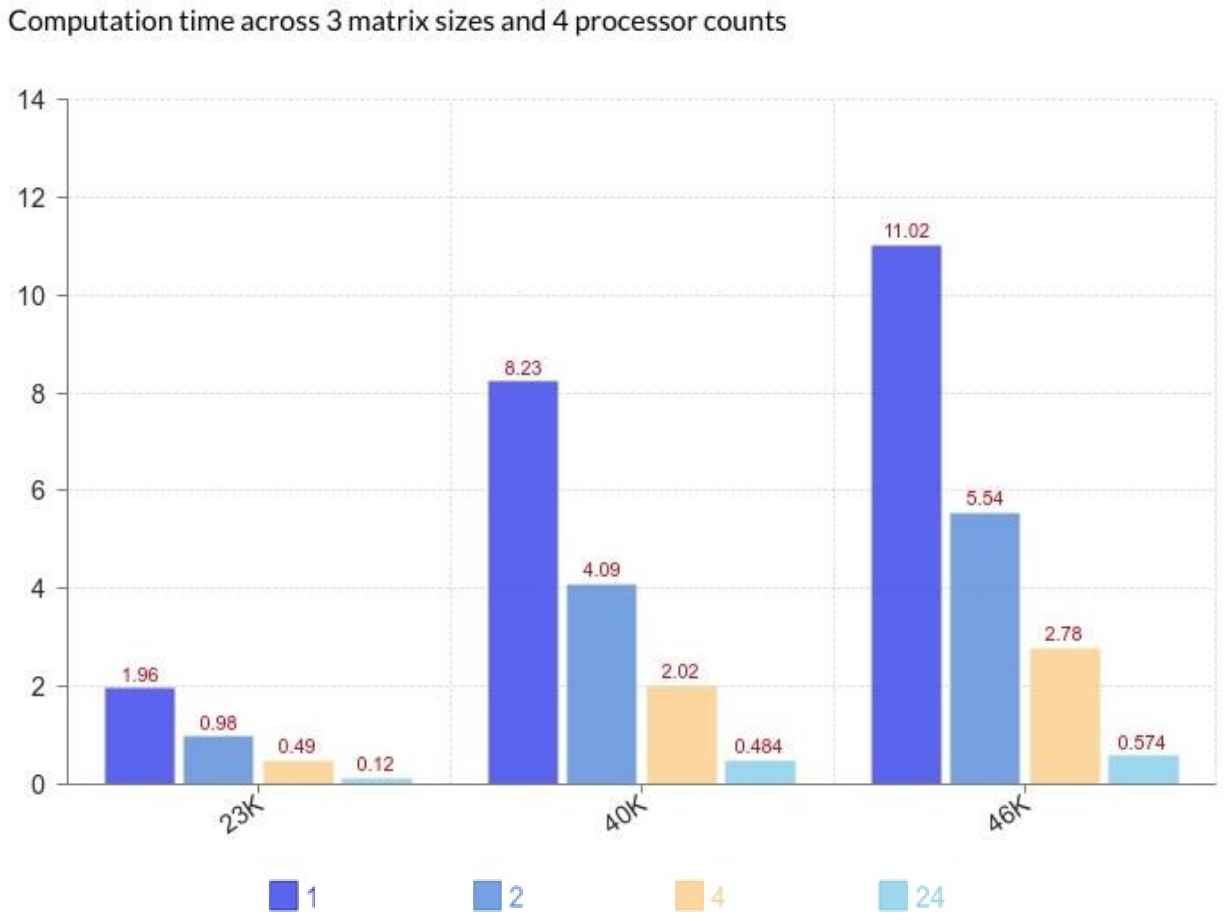

Figure 1

The best results for analyzing the performances of MM on multithreaded systems come from using large matrices. Here we use an A with square dimensions of 23K, 40K, and 46K. Larger matrices beyond 46K would be useful but because these matrices are stored in a flat file the total values in the array must be less than (2^31 – 1), the int maximum range, I attempted to use a longer data type rather that int in order to raise this limit but the Comet system still wouldn't run anything past a 46K by a 46K. From figure 1, we can see that as the process thread increases the computation time goes down sharply. This is relevant in Figure 2 where we can see that the speedup as we ascend in thread count gets sharply higher. As MM has little not overhead within the computation time portion the speedup is roughly the same as the number of processors it was ran on. This is not true for the matrices ran on 24 cores. When the problem size, ie work to be done, is small and distributed among a large number of processors the overhead starts to become more relevant to the time proportional to that actual work being done. As the problem size increases this problem is resolved as each thread has more work to do. If I had collected more data for an even larger matrix size we would expect the the cores equals 24 value to be approaching 24 eventually.

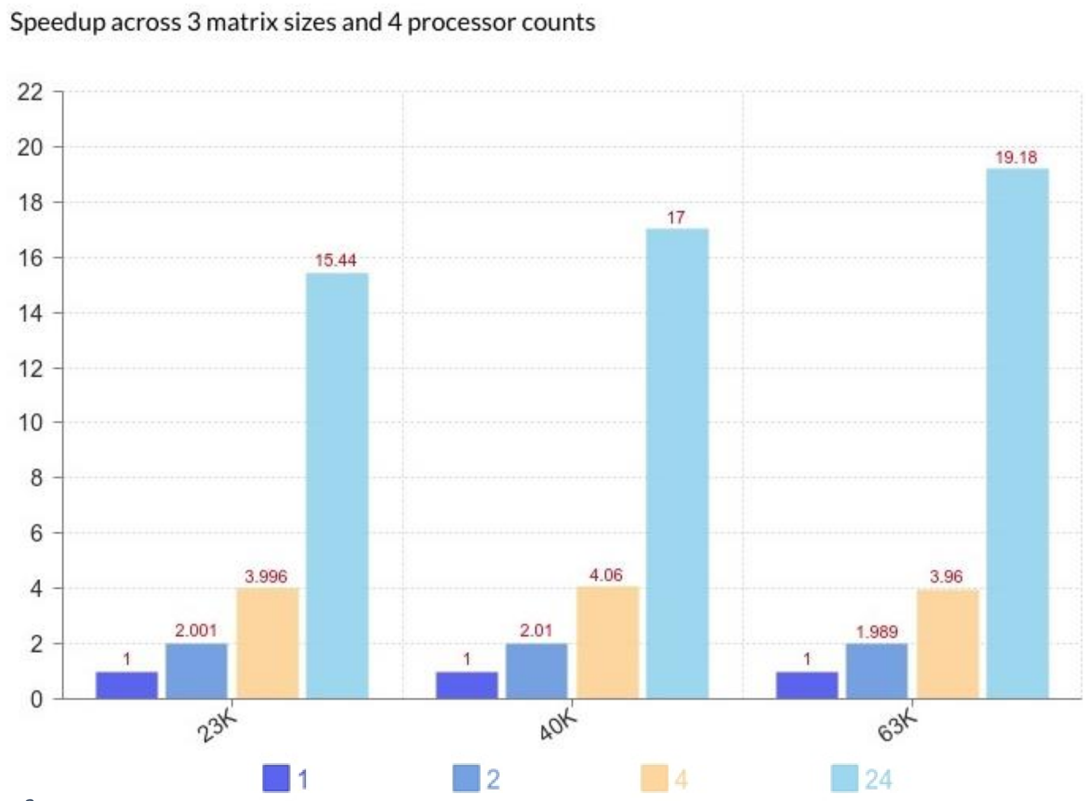

Figure 2

This idea of approaching the theoretical speedup is shown in Figure 3 and can be seen where the efficiencies are near 100%. Also you can see as the problem size increases the efficiency across the 24 cores data points beginapproach one. Interestingly enough we can see minor descrepancies among the data trends. One example of this is the 4 process 40K matrix had a bigger speedup than the theoretical maximum speedup should be, this is also present in the efficiencies that are more than 1. These can be atrributed to outside influences, such as other tasks being run on Comet, and would resolve itself if ran repeatedly and averaged.

Anakin Kinsey
CSCI 473 – HW08
12/6/2019

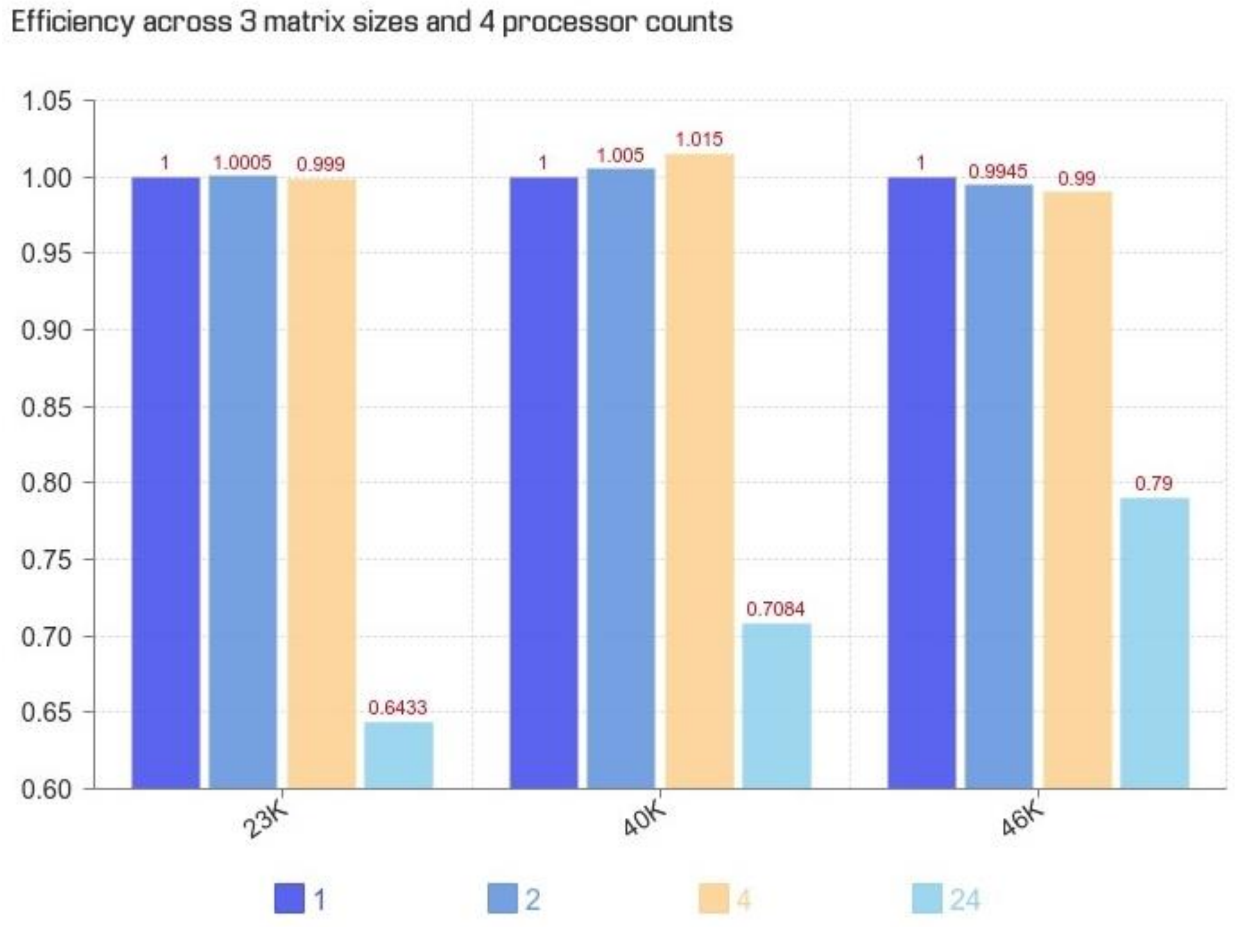

Figure 3