Shaun Howard 2/2/17
EECS 438: HW1 smh150

Matrix Multiplication

The row-major and mixed row and column-major matrix multiplication algorithms were run and timed on stampede starting with two square double-precision floating point matrix (modeled as an array) of dimension size 2000, i.e. 2000x2000. The experiment was run for 10 epochs for both ordering combinations, and the dimension of both arrays was increased 100 per timing loop until size 2900.

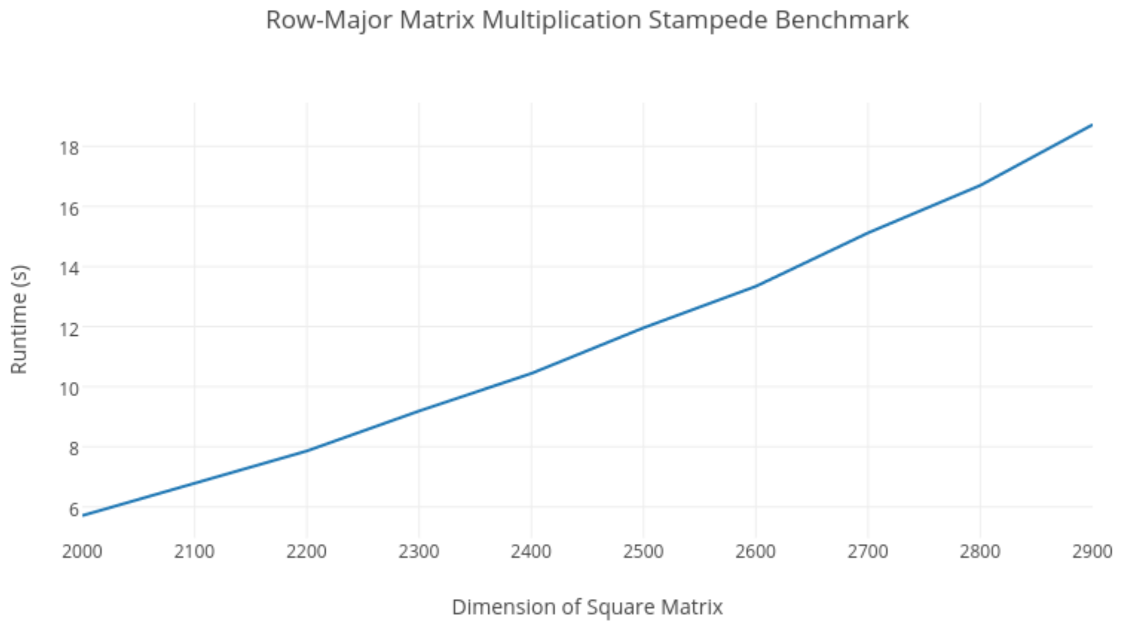
The starting size of 2000 was chosen because that was the size that made the server run for at least 1 second using either of the ordering techniques. The results were obtained from the implementation included in hw1_matrix_mult.c

Both sets of results were attained in under 10 minutes.

1. Below is a table of results from running row-major order matrix multiplication between two square matrices of growing size starting at 2000x2000 on Stampede cluster:

Square Matrix Size	**Stampede Compute Node Runtime (s)**
2000	5.71
2100	6.79
2200	7.86
2300	9.19
2400	10.44
2500	11.96
2600	13.34
2700	15.12
2800	16.7
2900	18.72

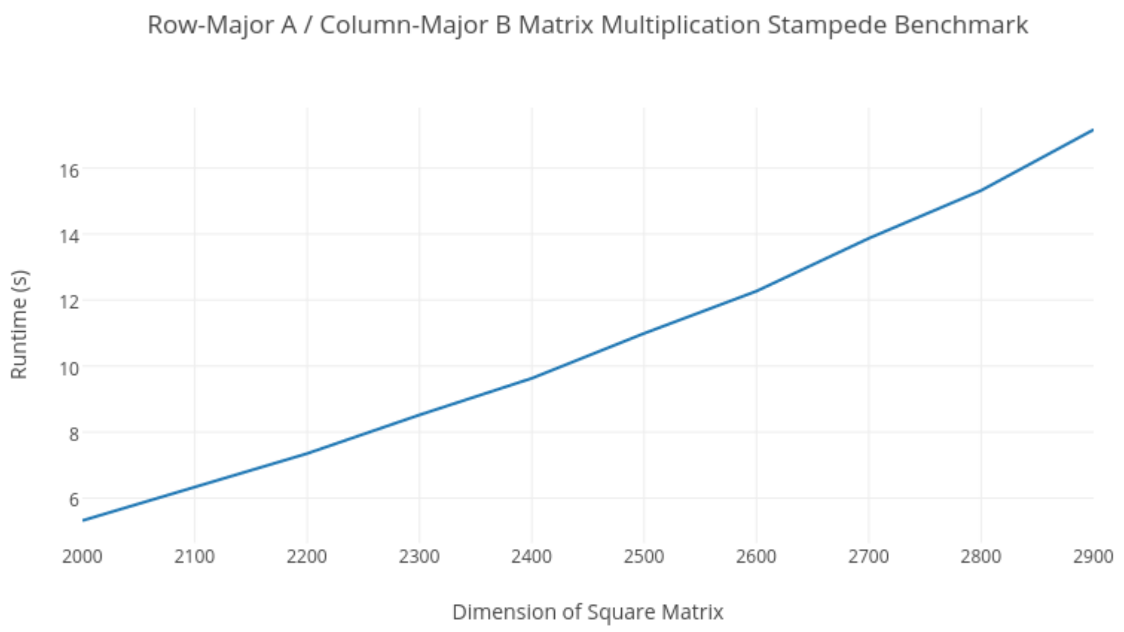
Below is a graph of these results, which show a linear relationship between the square dimension size of the matrices and the runtime taken on the compute nodes.



2. Below is a table of results from running mixed row-major order for matrix A and column-major order for matrix B matrix multiplication between two square matrices of growing size starting at 2000x2000 on Stampede cluster:

Square Matrix Size	**Stampede Compute Node Runtime (s)**
2000	5.32
2100	6.34
2200	7.35
2300	8.52
2400	9.63
2500	10.99
2600	12.27
2700	13.87
2800	15.32
2900	17.16

Below is a graph of these results, which also show a mostly linear relationship between the square dimension size of the matrices and the runtime taken on the compute nodes.



Conclusion:

By analyzing these results, we can see that the row-major indexing order ran slower than the mixed row-major and column-major ordering. The minimum and maximum runtime values are greater for the completely row-major order version of the code. Hence, the mixed row-major order for matrix A and column-major order for matrix B is faster than just row-major order matrix multiplication using the C language. This conclusion makes sense because the column-major method uses a pointer to the beginning of the column array, and accesses contiguous column elements in array B for the inner loops, rather than accessing the data separately and skipping rows as mentioned in the assignment. Therefore, we have determined that in order to remain most time efficient, one must use contiguous elements of arrays to speed up access times in large arrays, such as using pointers and column-major ordering for inner loops. These implementation changes have the overall affect of speeding up matrix math operations like matrix multiplication as shown above.