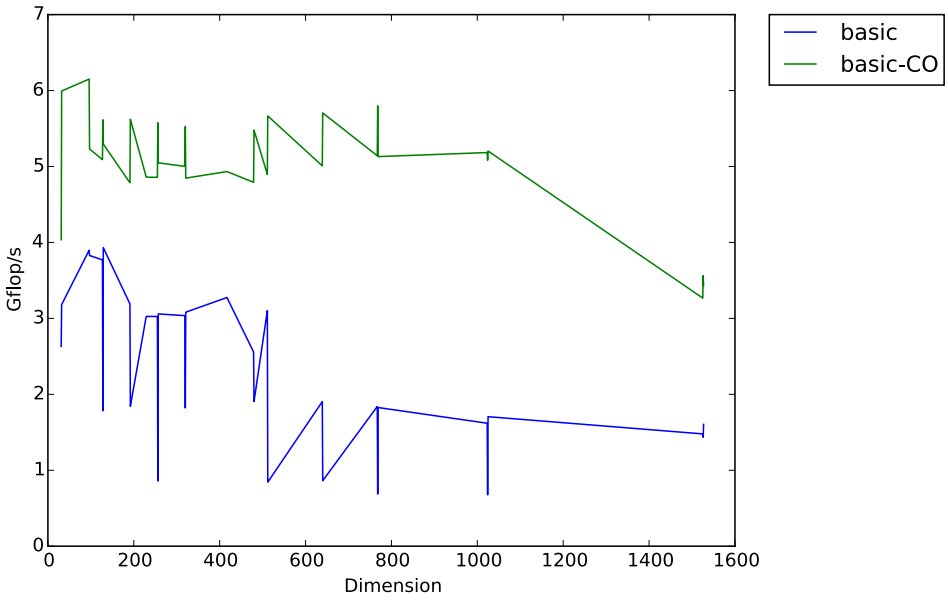
Optimization of Matrix Multiplication

Markus Salasoo, Vikram Thapar, Yalcin Ozhabes

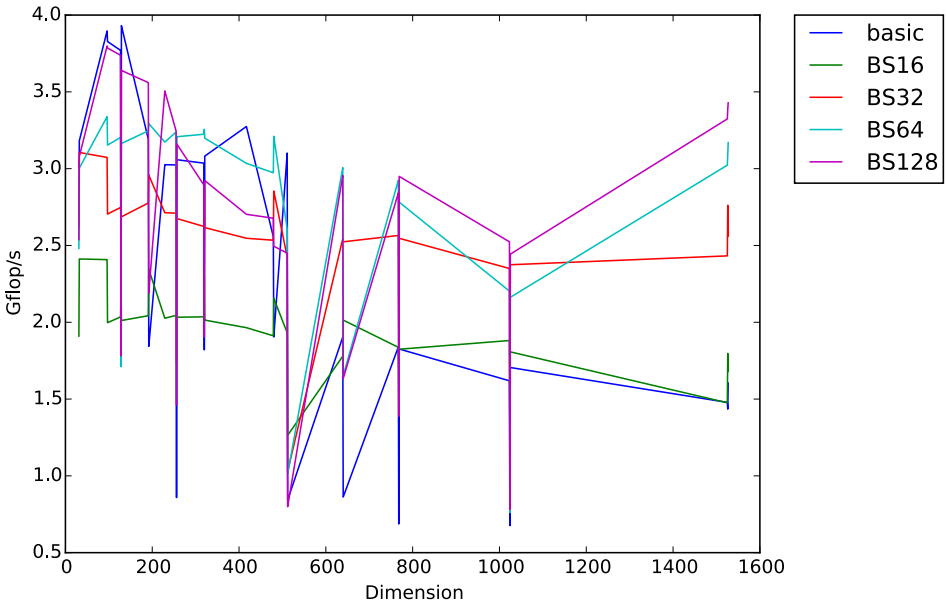
In our initial report, we list here the following attempted optimizations along with obtained results

Copy Optimization of Basic Code



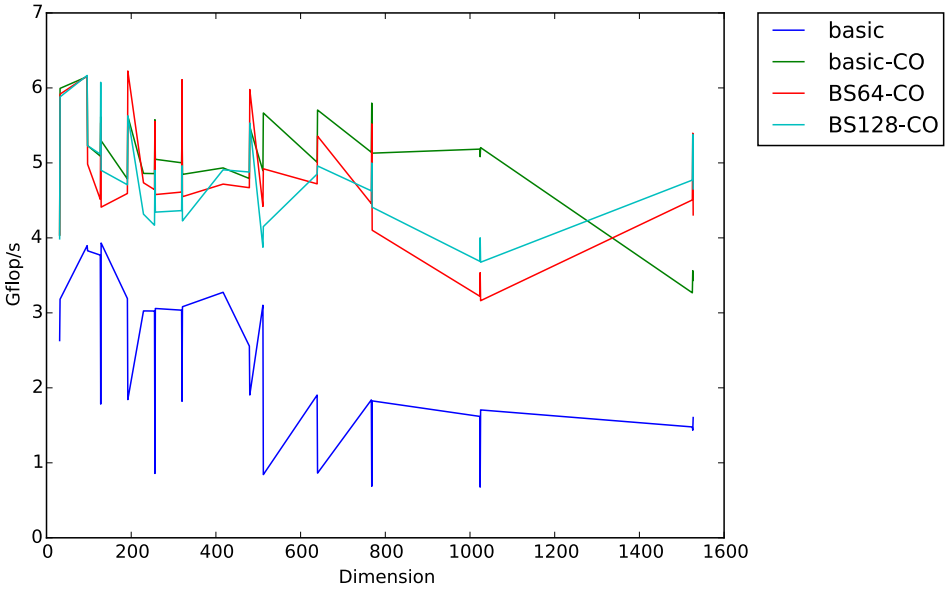
In the basic code, first we perform the copy optimization. Copy optimization involves storing each row of A in spatially contiguous memory. As shown in Figure 1, this improves the performance significantly for small sized matrices but on further increasing the matrix size, there is a dip in the performance.

Blocking



Given the cache sizes, 32KB, 256KB and 15MB for L1, L2 and L3 caches we estimated the size of the largest matrices fit in. We need to access two matrices at the same time. In order to fit in L1, the block size should be less than 45; for L2, less than 128; for L3, less than 990. We tried different block sizes and the results are shown in the above figure. At multiples of 2, we observe a dip in the performance because of extraordinary amount of cache misses.

Blocking + Copy Optimization



Copy optimization helps reduce the dips in performance at powers of 2 and further enhances the performance because of spatial locality.

Compilation Flags

We tried a handful of compiler flags, but did not see a significant increase in performance. There was some variation, but we did not consider any single flag to contribute a significant increase to our flop rate.

Conclusion

Following are the results shown for 1526by1526 Matrix.

Type of Optimization	Performance Enhancement Factor
Basic	1.0
Basic + Copy Optimization	2.5
Basic + Copy Optimization + Blocking (Block Size = 128)	3.8