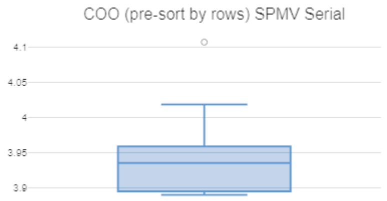
Homework 3 Report

1 Test Environment

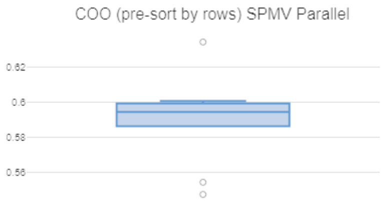
I tested my code on department's ix server. It has two sockets with AMD Opteron 6376 on each socket. Each CPU has 8 cores, 16 threads. So, there are 32 hardware threads in total.

2 Test Results

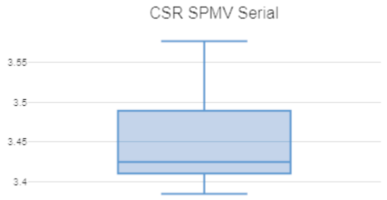
2.1 Results over 10 runs: Execution Time (seconds)



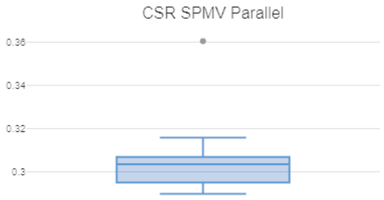
(a) Average: 3.9490364



(b) Average: 0.5898402

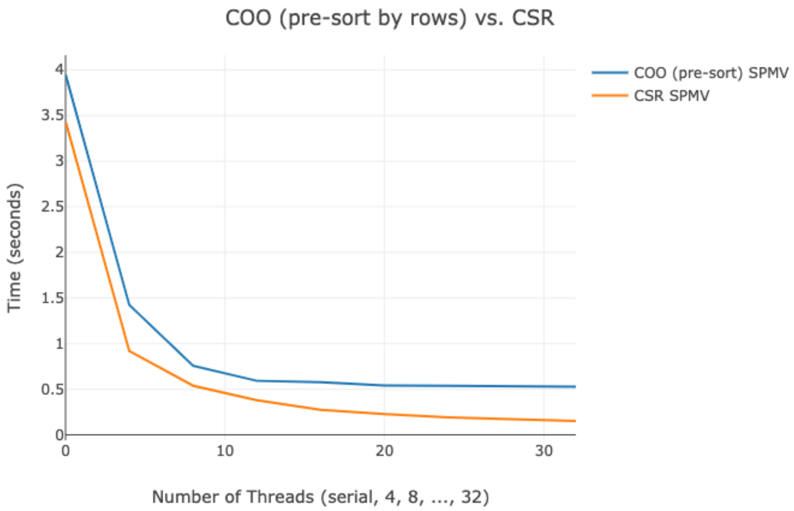


(a) Average: 3.4472758



(b) Average: 0.3070956

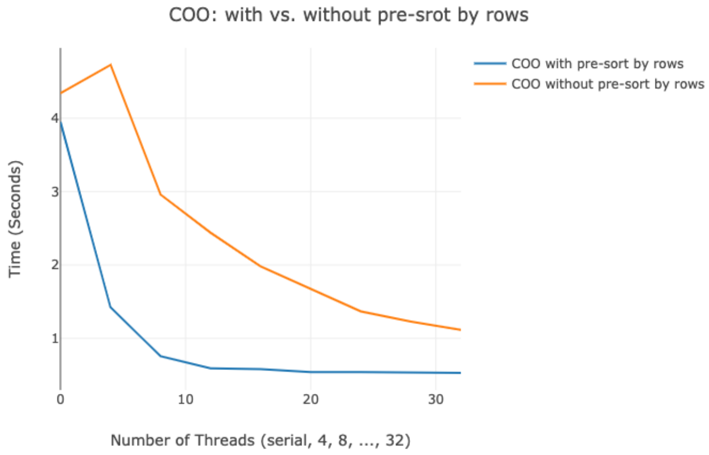
2.2 Performance: COO (pre-sort by rows) vs. CSR



2.3 Accuracy (diff results): COO vs. CSR

	0	4	8	12	16	20	24	28	32
COO	-	-	-	-	-	-	<3266.0819162971 >3266.0819162970	<1271.8511032950 >1271.8511032951	<-4759.7834986281 >-4759.7834986282 <2964.0054889568 >2964.0054889569
CSR	-	-	-	-	-	-	-	-	-

2.4 COO: With Pre-Sort vs. Without Pre-Sort



3 Findings

- CSR needs to be pre-sorted by rows to be used for matrix multiplication. This is because, we need to sort rows to know row offsets, which is used to calculate row pointers.
- For sparse matrix multiplication (serial or parallel), CSR performs better than COO (with or without pre-sort by row).
- For parallelization, the number of threads affects CSR more than it affects COO.
- CSR's accuracy is not affected by the number of threads; COO's accuracy is affected when using more than 24 threads. However, the difference (10^{-10}) is acceptable and it is only for one or two elements.
- For COO format without pre-sort by rows, without lock, parallelization does not yield the correct result.
- For COO format without pre-sort by rows, but with lock, parallelization will yield result within 10^{-10} difference.
- For COO format with pre-sort by rows, adding locks will hurt the performance. This is because once COO is pre-sorted by rows, adding locks is not needed.