## Introduction

To estimate the effect of a shared memory multi-processor system by performing matrix multiplication on various cache configuration.

## Assumption

- 4 way set associative
- 4 words per block
- 32 kB and 8 kB cache size
- Number of processors: 1 to 50
- Matrix multiplication size: 50x50
- No layer 2 cache
- Memory arbitration takes 0 clock cycles

## Simulation method

Before the main part of simulator function, all the address(in order) need to be access by each processor is pre-calculated and placed into a two dimensional array. Then in the main simulation loop, each processor will take the next address from the array once the last calculation is done.

In the main simulation process, once processor accesses a memory location, the time delay for this access is calculated, then a resume time for this processor is calculated and stored in an array. Once the global counter reaches this resume time, such processor can access the next address stored in the address array, and another resume time is calculated.

Effect of Address distribution method

There are three address distribution methods considered: row major, column major and element major. If there are three processor, for element major, P0 will get element C(0,0), C(0,3), C(0,6), C(0,9)…, P1 will get C(0,1), C(0,4), C(0,7), C(0,10)…, and P2 will get C(0,2), C(0,5), C(0,8), C(0,11)… . For column major P0 will get column C(0,-), C(3,-), C(6,-)…, P1 get column C(1,-), C(4,-), C(7,-) and P2 get column C(2,-), C(5,-), C(8,-). Simular for row major.

Figure 1: Clock cycle vs Cache size vs Address distribution method

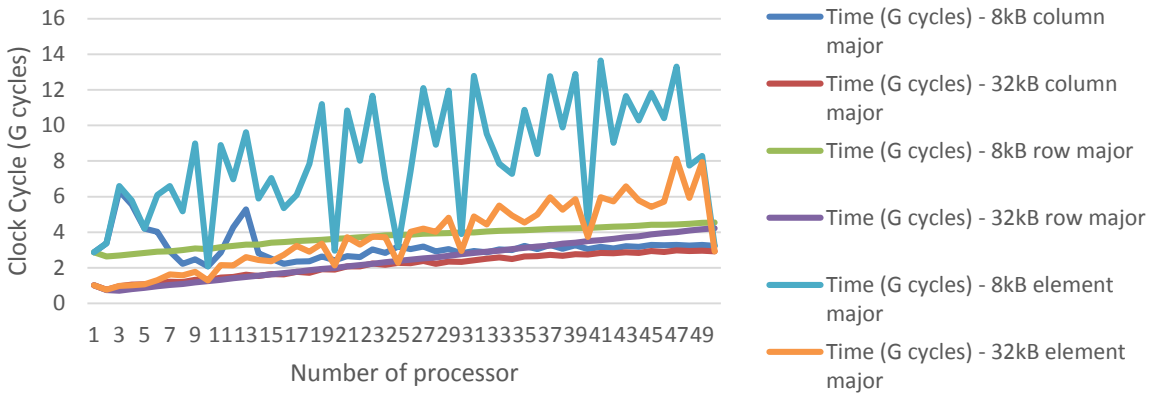



Figure 2: Clock cycle vs Address distribution method (8kB)

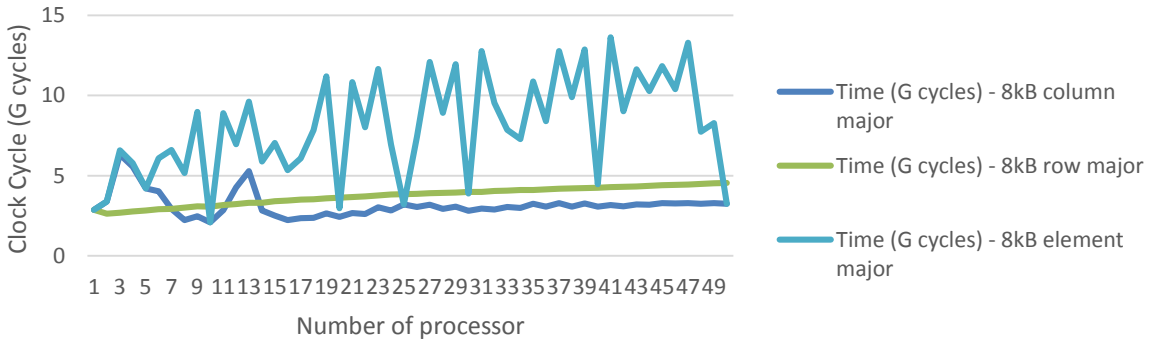
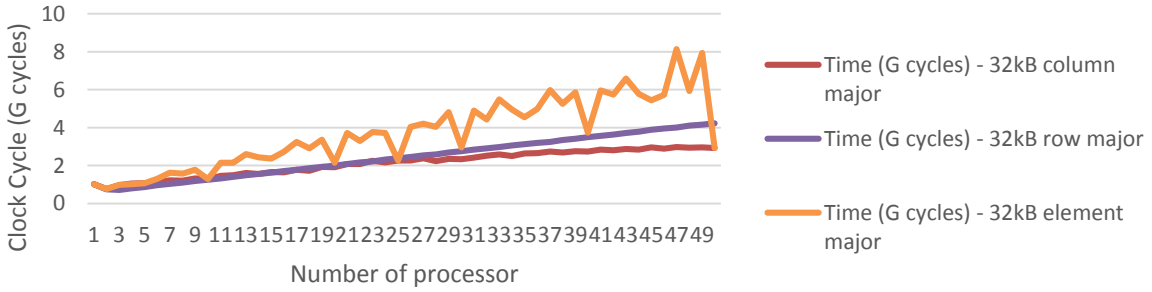



Figure 3: Clock cycle vs Address distribution method (32kB)

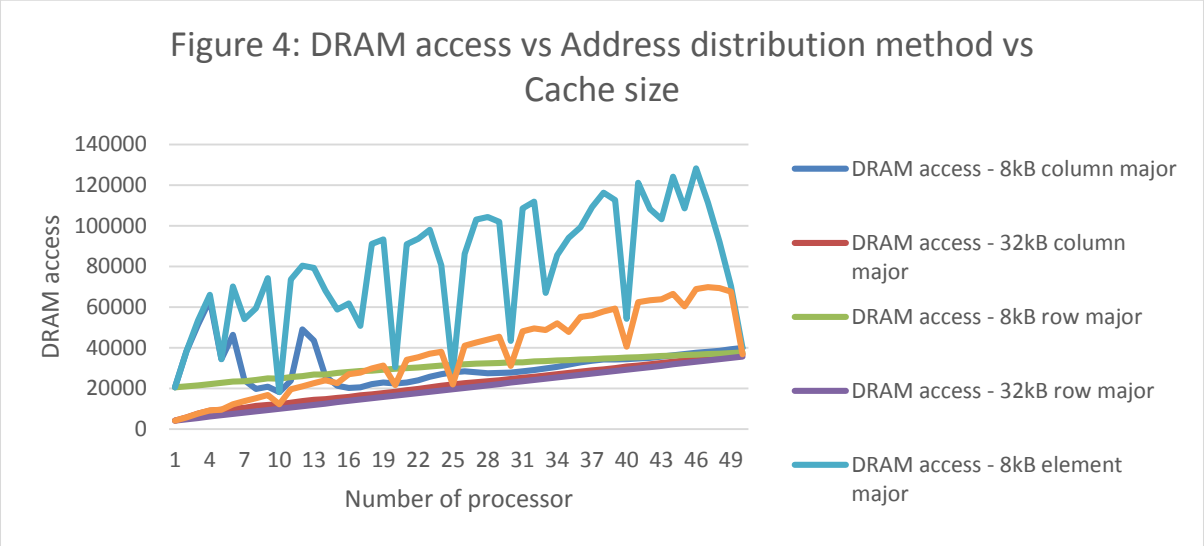


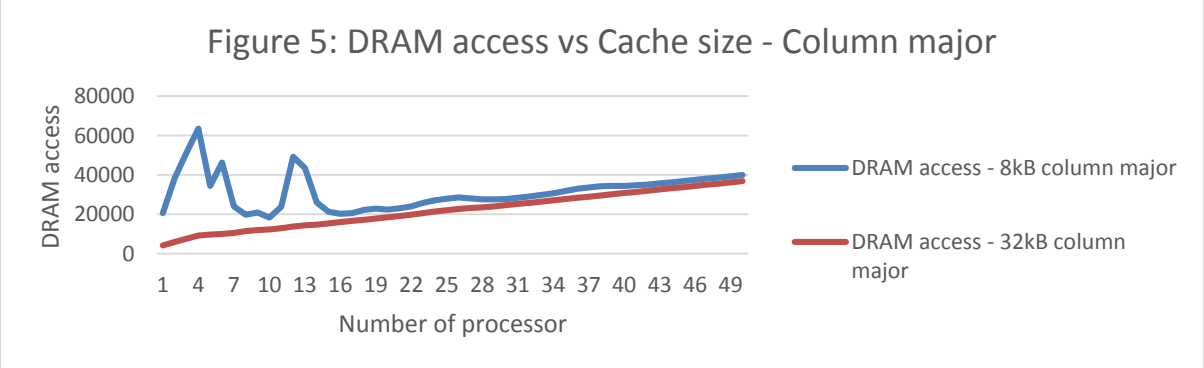

For element major method, when the number of process is a factor of matrix size such as 10, 20, 30 40 and 50, the processing speed is significantly faster. This is because one processor will be assigned to element in same column in matrix C more than once, hence increase the cache hit rate, reducing time wasted on waiting DRAM access.

Both the column major and row major method already have such advantage, hence their curve is much smoother than element major.

## Number of processor

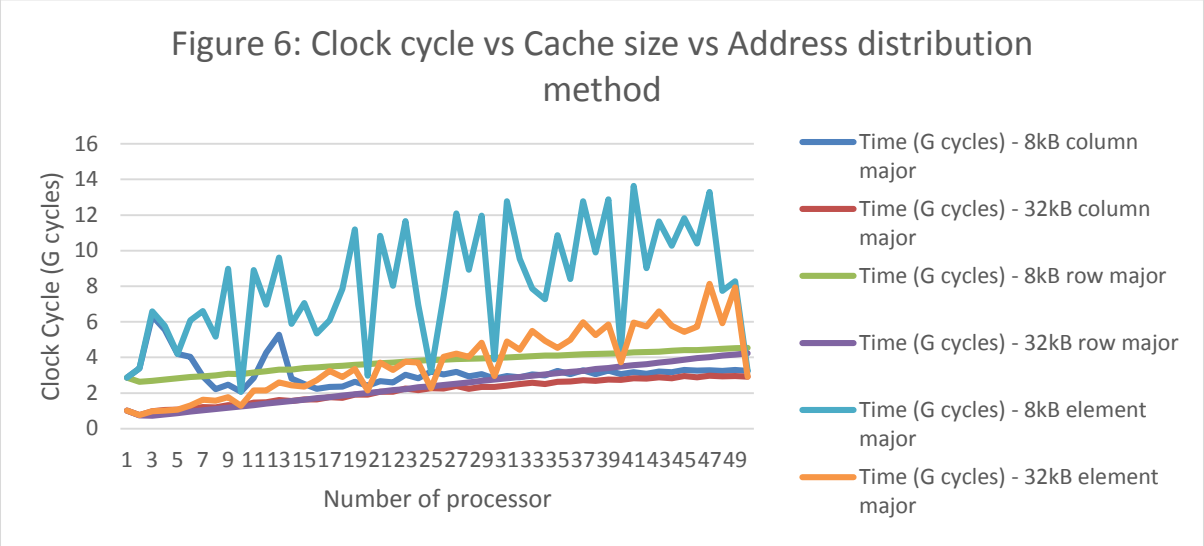
In ideal case, according to Amdahl's Law, for np number of processors, np times faster the overall speedup. However, in our case this only happed up to a certain number of processor with the best address distribution methods. Since the DRAM is shared between all processors, only one processor can access memory at same time. If more the one processor need to access memory at same time, they have to wait in a queue for the one to unlock the memory access. The more processor is used, the more DRAM access is required by processor, the more change multiple processor need access memory at same time. Once a processor is waiting in the queue, it can no longer contribute to processing result, hence reducing the advantage of multiple processor.




The plot above shows the DRAM access vs number of processors. It cleanly show when number of processor increases, the more DRAM accesses is required. This is also reflected on hit rate. The DRAM access has similar trend comparing to processing speed, which means the DRAM access delay is heavily limiting the speedup of multi-processor.

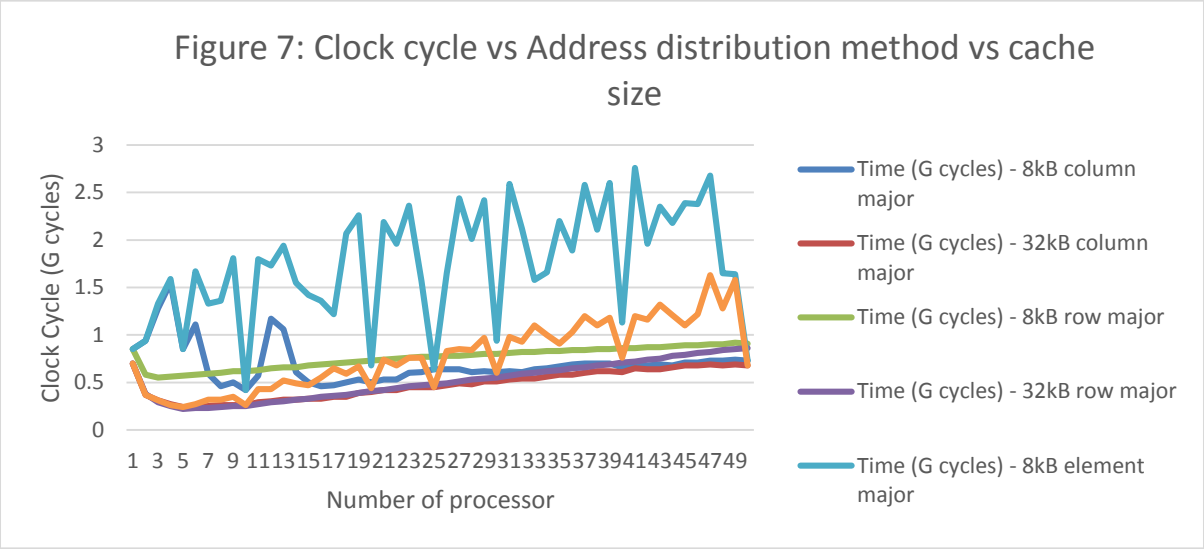
## Cache size




For a small 8kB cache size, not all of the 50 by 50 matrix can fit into cache. Hence some data in cache can be replaced when cache miss. A 32kB cache is large enough to fit all 50 by 50 data, increase the change of cache hit. Such increase in hit rate has major impact on clock cycles, because processor may need to wait a long time to accessing the shared memory once cache miss. In the result trend, no matter which address distribution method is chosen, 32kB always has a higher hit rate and lower clock cycles comparing to 8kB with some configuration.

## How to improve

In the implementation, the calculation result of each processor is independent of results from other processors. However, we find memory access delay is the main bottleneck and stop us to take full advantage of multi-processor. Once of the improvement can be done is shorten the DRAM delay.




Plot above same implementation but with five times smaller CAS can RAS. By reducing the DRAM access delay by a factor of five, it greatly reduces the limitation stated above. We can see the clock cycles reduces to a further numbers of processor value comparing to normal DRAM access delay. Hence by improving the performance of DRAM, we can take more advantage brought by multi-processor.

Another possible way to improve performant is a shared second layer cache. Usually second layer cache is slower in time but lot larger in capacity. A 50by50 matrix will easily fit into L2 cache. Since most of the matrix A and matrix B data can be used by all processors, a shared L2 cache is expecting to boost the performance a lot even with a small L1 cache.

## Conclusion

In conclusion, multi-processor can improve the system performance. However with the limitation of DRAM access delay, more and more processor power will be wasted on waiting data from main memory. Such limitation can be reduced by shorten DRAM delay or implement L2 cache.