John Naylor

CMSC 257

Computer Systems

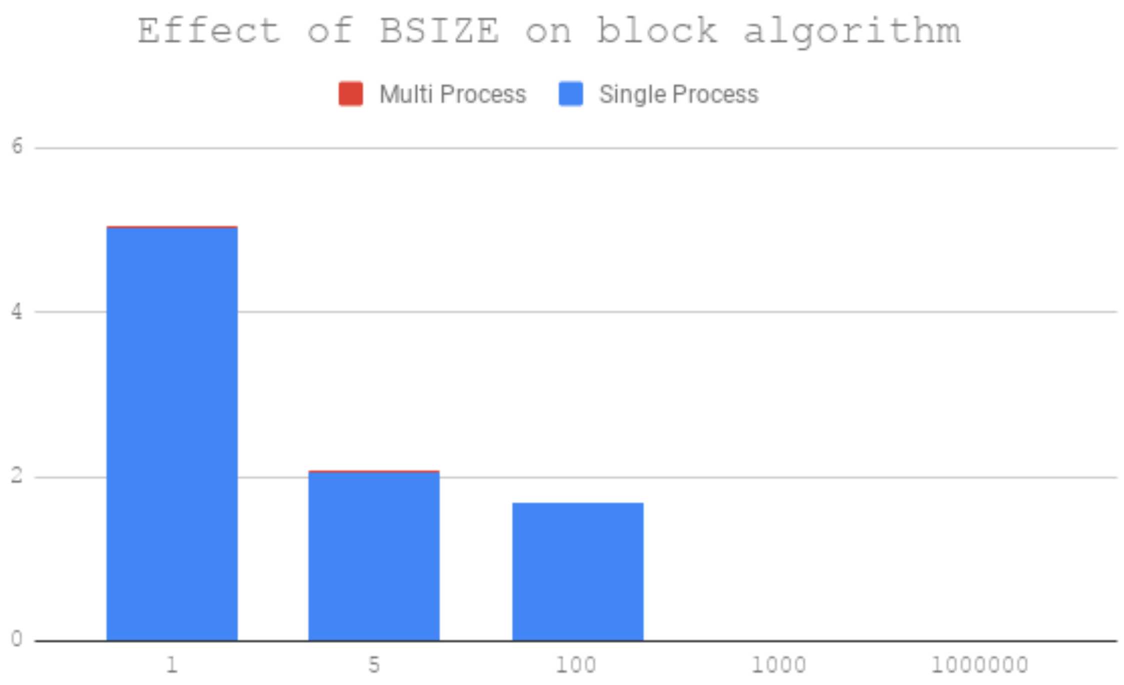
Assignment 4 Cache & Memory Utilization in Matrix Multiplication

1.

Forked Block algorithm does not output the correct time. To calculate the time, the program uses tms_utime which does not include system time which is crucial in a multiprocess application with many system calls and a semaphore.

2.

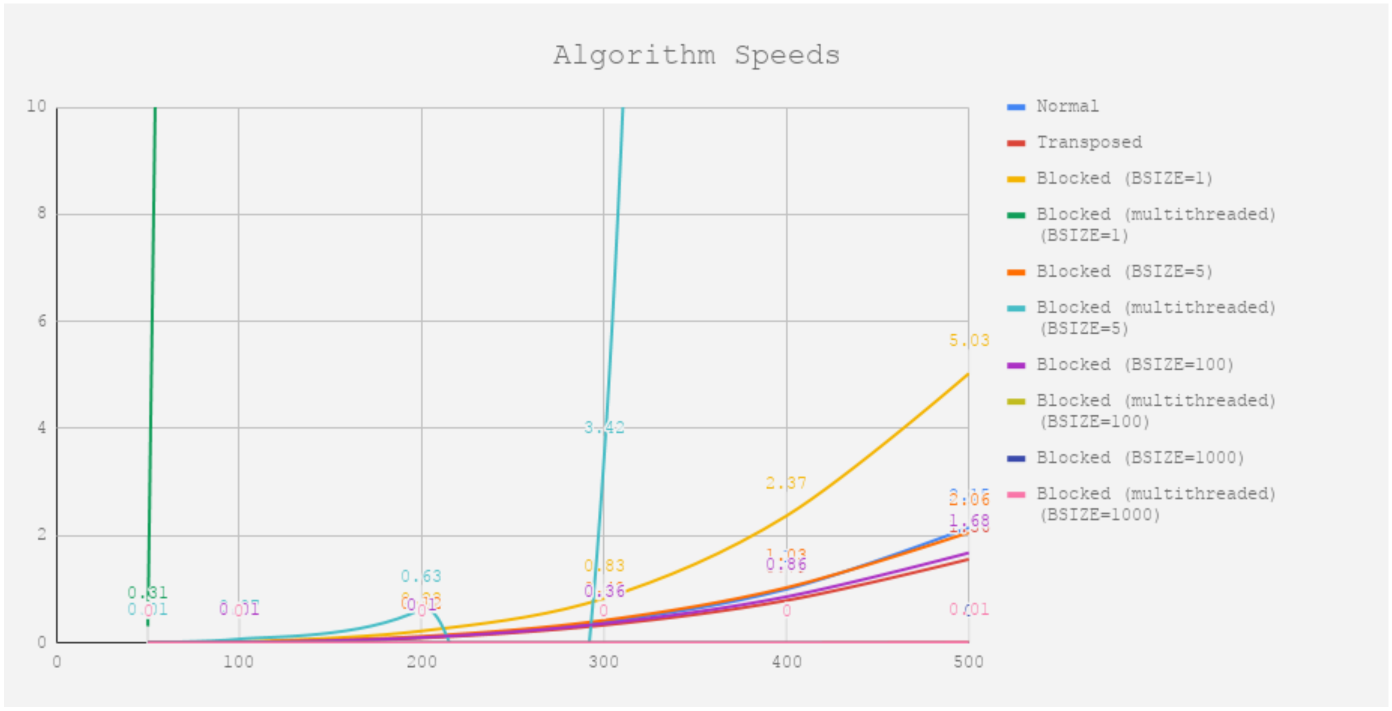
Ideal BSIZE depends on your hardware and the dimension of the matrices that need to be multiplied. Large BSIZEs can make multiplying large matrices very fast but most computers can't handle the huge influx of forks.



BSIZE VS TIME

3.

All depends on BSIZE which depends on CPU and the value 'n'. For large 'n', it seems like a blocked approach is the best way to go. For small values of 'n', the transposed algorithm worked the fastest.



N VS TIME

4.

Normal: The standard way to multiply matrices students learn in Linear Algebra. Rows multiplied by columns

Transpose: Same thing as normal but the array is transposed to make indexing more efficient in RAM.

Blocked: Breaks things into blocks and calculates partial sums in those blocks to discover the product of the entire final matrix

Blocked (Forked): Same thing as Blocked but each block is given its own process to execute all the sums

5.

Depends on your size of ’n’. Multithreaded should be the fastest but because it requires setting up the extra processes and address spaces, it can be slower for small values of ’n’.

6.

Ideal BSIZE depends on the hardware. When trying on my laptop at home it seems that at some point increasing BSIZE didn’t make the algorithm faster however, when doing the algorithm on the VCU compile server it seemed the even large values of BSIZE preformed well.

7.

Forked Matrix Multiplication used semaphores to handle collisions. This isn’t efficient because a process has to wait for someone else when it could be moving on to do more calculations.