In3030

Marius Madsen mariumh

Oblig2

This Program is constructed to take any n\*n matrices and multiply them. It can either be calculated sequential or parallel, also whether you want to solve it the classical a\*b way, a transposed \* b or a \* b transposed. However these three solutions have different calculation time compared to one another.

First of all is the sequential way to solve them all:

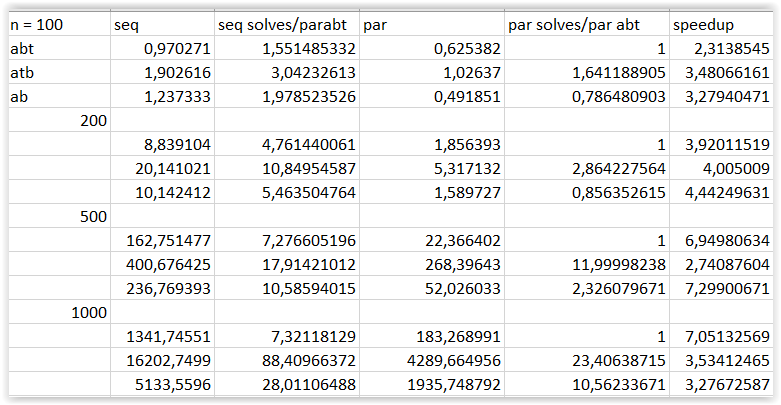
a\*b is the common way to solve it on a piece of paper. The formula is given by  
this solution is not recommended because the way b is accessed is bad considered it is bad cache accessing. This solution will have a computing time of n3.  
  
a transposed \*b, is also a way to solve it. Transposing means to change every rows of the matrix with the corresponding column.

This one is even worse considering it just is not b that has bad cache accessing, but now we use a transposed which is as bad. On top of that it has n3+n2.

a\*b transposed, is the final way it is solved, by transposing the b matrix instead, which will avoid the bad cache accessing. The formula is given by

Will prove to be solved faster than (a\*b) even though it has also n3+n2.operatoins, given that n is sufficiently large

All of these solutions have also been solved parallel. The way they have been done that is that each thread get assigned a portion of the 2 matrices, and put into a shared matrix called c. there is no need for synchronising since they don’t access the same indexes in the c matrix. The transponation method is also implemented to be able to be parallel.



From the data we get we can see a transposed \* b is a horrible solution. We can also see a speedup if we use the parallel version compared to sequential even at n = 100. But still n = 100 would mean at least 100\*100\*100 = 1000000 operations. We can also see that until n = 500 there is very little use a\*b transposed instead of a\*b

These numbers were generated by running the program on a computer with a Intel(R) Core(TM) i7-6700HQ CPU @ 2.60GHz, 2601 Mhz, 4 cores, 8 logical processors

To run the program type “javac \*java && java Oblig2 n seed” where n is the size the matrices will be having, and the seed is the seed of the matrices generated.

The conclusion is to use parallel a\*b calculation until n = 500 and after use a\*b transpose.