Designing and building applications for extreme scale systems
CS598 SP2016
HW3: Matrix Transpose and spatial locality

Objectives

- Observe the impact of a lack of spatial locality
- Gain experience using blocking of loops to improve locality
- Explore the use of the cache oblivious approach to formulate algorithms

Tasks

Part 1:

Write a simple code to transpose a dense matrix of size n x n. Time the performance of your code for n = 20, 100, 500, 800, 1000, 1500, 2000, 4000.

Using the STREAM performance from HW2 for the same computer system, and assuming that the time cost of a transpose operation is just the cost of a STREAM COPY of the same amount of data, what is the time would that performance model predict that a transpose should take?

Note

You may find Fortran particularly convenient for this, but you may use C or C++. Do not use aggressive optimization from the compiler (e.g., try only low level optimization, such as –O0)

Part 2:

Take your simple code and block the both loops to work with a bsize x bsize matrix. In Fortran, that code looks like (the Fortran "do i=s, e, incr" is like the C "for(i=s; i<=e; i+=incr)")

```
do i=1,matSize, bsize
   do j=1,matSize, bsize
     do i1=i,min(i+bsize-1,matSize)
       do j1=j,min(j+bsize-1,matSize)
         matB(i1,j1) = matA(j1,i1)
       enddo
     enddo
   enddo
enddo
```

Using a bsize of 16, run the same tests as in Part 1 and record the timings in the table.

Part 3:
Run the code cotranspose.c, which is a simple cache oblivious code, for the same matrix sizes as in Part 1. Record the timings in the table.

The table should look like this:

Size	Basic	Model	Blocked	Cache Oblivious
20				
100				
500				
1000				
1500				
2000				
4000				

Which method (based on 3 sets of measurement) is best? Does it depend on the matrix size?

Submit
You should submit the PDF file with both the table and the plot, including all the data in one plot with necessary explanation. Also you should state clearly your analysis on the performance model in **Part 1**.

Think about the following (but you don't need to turn in):

1. Would it help to block separately for each level of cache? Why or why not?
2. In this exercise you used square blocks. Would rectangular blocks be better? If so, what shape and why? How would you test this?
3. Can you use the cache oblivious performance model to estimate the matrix size where the transpose performance changes abruptly?