Homework 1, Parallel Programming – Monsoon 2012

Suresh Purini, IIIT-H

1. **In-place Matrix Transposition Problem**

   (a) Write a program to transpose an $N \times N$ matrix. Check which compiler optimization flags can possibly enhance your program performance.

   (b) Write a tiled version of the above program.

      i. Analytically determine the optimal tile size.

      ii. Find the optimal tile size using an empirical search technique.

      iii. What is the complexity of your empirical search techniques in terms of the number of searches you make?

      iv. Does your empirical search technique necessarily gives you the optimal tile size?

   (c) Write a 2-level tiled version of the program. Find the optimial tile sizes using an empirical search technique.

   (d) Write a cache oblivious algorithm for the same.

   (e) Write a program for matrix transposition using the Intel Math Kernel Library.

   Compare the performance of compiler optimized naive matrix transposition, single tiled, 2-tiled, cache oblivious and Intel MKL programs with respect to time, cache hit ratios and other performance parameters which you envisage. Plot the time, cache hit ratios by varying matrix sizes. Your largest matrix size should be atleast 1 GB large.

   Challenge Problem: Can you propose a method to determine the cache architecture, like the number of levels and size of cache at each level by analyzing the variance in the program performance for various matrix dimensions.

2. **Out-of-place Matrix Transposition Problem** Solve all the parts from the previous problem assuming out-of-place matrix transposition for rectangular matrices of dimension $M \times N$.

3. **Prefix Sum Problem**

   (a) Write a program for the prefix-sum problem which contains the upward and downward sweeps. You should do everything **in-place**. Check which compiler optimization flags can possibly enhance your program performance.

   (b) Find the cache miss complexity. Is this algorithm cache oblivious?

   (c) Come up with a tiled version of program for the prefix-sum problem and compute the optimal tile sizes.

   (d) Write a 2-level tiled version and compute the optimal tile sizes through empirical search techniques.

Compare the performance of the compiler optimized program, single tiled and two tiled versions with respect to time and cache usage. You can also report other parameters of interest if any.

Challenge Problem Can you propose a cache oblivious algorithm for prefix sum?