CS 610 Semester 2020–2021-I: Assignment 3

3rd October 2020

Due Your assignment is due by Oct 11 2020, 11:59 PM IST.

General Policies

- You should do this assignment ALONE.
- Do not plagiarize or turn in solutions from other sources. You will be PENALIZED if caught.
- We MAY check your submission(s) with plagiarism checkers.

Submission

- Submission will be through mooKIT.
- Write your programs in C++ .
- Feel free to use Intel DevCloud. Include the system description (for e.g., levels of private caches, L1/L2 cache size, processor frequency) and compiler versions in your report.
- Submit a compressed file with name "<roll-no>.zip". The compressed file can have the following structure.

```
 -- roll-no
-- -- report.pdf
-- -- <problem1-dir>
-- -- -- <source-files>
-- -- <problem2-dir>
-- -- -- <source-files>
```

  Submit a PDF file with name "report.pdf". We encourage you to use the LaTeX typesetting system for generating the PDF file. The file should include your name and roll number, and results and explanations for the following problems. Include the exact compilation instructions for each programming problem, for example, `g++ -O2 problem2.cpp`, and any other information and assumptions that you think will help the evaluation.

- You will get up to TWO LATE days to submit your assignment, with a 25% penalty for each day.

Evaluation

- Please write your code such that the EXACT output format (if any) is respected.
- We will evaluate your implementations on a Unix-like system, for example, a recent Debian-based distribution.
- We will evaluate the implementations with our OWN inputs and test cases, so remember to test thoroughly.

Problem 1 **[100 marks]**

Consider the following code snippet.

```
#define N 4096
#define Niter 10;

double x[N], y[N], z[N], A[N][N];

for (int t = 0; t < Niter; t++) {
  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      y[j] = y[j] + A[i][j]*x[i];
      z[j] = z[j] + A[j][i]*x[i];
    }
  }
}
```

Perform loop transformations to improve the performance of the following code snippet for sequential execution on one core (i.e., no multithreading).

1. Modify the `optimized()` function in the template code and perform evaluations.
2. Implement a version of the best-performing variant from the previous step using intriniscs. You can make use of any valid data type supported with intrinsics (for e.g., `__m128d`).

We will provide a template code to help with the computation.

You may use any transformation we have studied, for e.g., loop permutation and loop tiling, but NO ARRAY PADDING OR LAYOUT TRANSFORMATION OF ARRAYS are allowed.

Present speedup results with both the GNU GCC (`gcc -O2`) and Intel ICC (`icc -O2`) compiler. Briefly explain the reason for the poor performance of the provided reference version, your proposed optimizations (with justifications) to improve performance, and the improvements achieved.

Problem 2 **[100 marks]**

The description is similar to Problem 1.

```
  #define N 1024

  double A[N][N], B[N][N], C[N][N];

  for (int i = 0; i < N; i++) {
    for (int j = 0; j < N; j++) {
      for (int k = 0; k < i+1; k++) {
        C[i][j] += A[k][i] * B[j][k];
      }
    }
  }
```

Problem 3 **[50 marks]**

Consider the following code snippet.

```
#define N (1<<12)
#define ITER 100
double X[N][N];
int i, j, k;
for (k=0; k<ITER; k++) {
  for (i=1; i<N; i++) {
    for (j=0; j<N-1; j++) {
      X[i][j+1] = X[i-1][j+1] + X[i][j+1];
    }
  }
}
```

Create a parallel version using OpenMP. Feel free to apply *valid* loop transformations. Report the speedup you get with the OpenMP parallel version.