CSCE 659 Fall 2017

HW 3: Dense Matrix Computations with OpenMP

Due: 11:59pm Saturday, November 18, 2017

The matrix computation code included in this assignment computes the solution of a linear system Ax=b. The matrix A is initialized in `initialize_matrix` and the right hand side vector b is initialized in `initialize_vector`. LU factorization is computed in place by the routine `LU`. The routines `solve_L` and `solve_U` compute the solution of the systems Ly = b and Ux = y, respectively. To verify the accuracy of the solution, the program computes the size of the residual vector b–Ax. The routine `matvec` computes the product of A and x. The routine `saxpy` computes x + α y and `norm` computes the norm of a vector.

1. (35 points) Parallelization with OpenMP.
    a. (30 points) You are required to use OpenMP directives to parallelize the routines: `LU`, `solve_L`, `solve_U`, `matvec`, `saxpy`, and `norm`.
    b. (5 points) Determine the speedup and efficiency obtained by these routines on 1, 2, 4, 10, and 20 processors for a system of size 8192.
2. (15 points) The parallel performance of the routines depends on their use of shared caches among the cores of a processor and on work distribution across such processors on a node. Implement a strategy that influences how threads get assigned to cores in order to maximize the performance of these routines. For this problem, you may focus on the performance of a single routine, i.e., the `LU` routine.
    a. (10 points) Outline your strategy to exploit shared caches to improve the performance of the code. Provide details on how you implemented your approach.
    b. (5 points) Report the execution time, speedup, and efficiency obtained by the `LU` routine with your strategy and compare it with the performance of the routine without the strategy.

Submission: You need to upload the following to eCampus:

1. Submit the code you developed.
2. Submit a single PDF or MSWord document that includes the following.
    - Responses to Problems 1 & 2. Response to 1 should include a brief description of the changes made to the code, the parallel computer used, and how to compile and execute the code on the parallel computer

Helpful Information:

1. Source file(s) are available on ecampus.
2. Load the Intel software stack prior to compiling and executing the code. Use:

```
module load intel/2017A
```

3. Compile C programs using `icc`. For example, to compile `code.c` to create the executable `code.exe`, use

```
icc –o code.exe code.c
```

To compile and execute the parallelized code, you can use:

```
icc -qopenmp -o matrix.exe matrix.c
export OMP_NUM_THREADS=8
matrix_hw 1024
```

Actual experiments must be carried out in batch mode. A sample of the results obtained is also included.

4. The run time of a code should be measured when it is executed in dedicated mode. Create a batch file as described on hprc.tamu.edu and use the following command to submit it to the batch system:

   ```
   bsub < batch_file
   ```