Lab 3: OpenMP and MPI+OpenMP

EEL 6763 – Spring 2023

Part 1: OpenMP matrix multiply

Using the following matrix multiply code snippet, write an OpenMP program to parallelize the code using OpenMP (parallel and work-sharing) directives. Only the master thread (Thread 0) populate A and B; but all threads (including Thread 0) should participate in the calculation of the matrix multiply.

```
int NRA atof(argv[1]);
int NCA_RB atof(argv[2]);
int NCB atof(argv[3]);
int A[NRA][NCA_RB], B[NCA_RB][NCB], C[NRA][NCB];
int i, j, k;
for (i=0; i<NRA; i++)
   for (j=0; j<NCB; j++)
   {
     C[i][j] = 0;
     for (k=0; k<NCA_RB; k++)  /* NCA = NRB = NCA_RB*/
       C[i][j] = C[i][j] + A[i][k] * B[k][j];
   }
```

These values are inputted through the command line:

- NRA: number of rows for Matrix A
- NCA_RB: number of rows for A and columns for B
- NCB: number of columns for B

a) Run the OpenMP code with original matrix sizes (NRA=60, NCA_RB=12, NCB=10), with 4 threads, and compare your result with the MPI-only code. Include this result in your report.

b) Create a **"strong-scaling"** bar graph (illustrated by an example in Figure 1), with the following specifications:

Keep the size of matrices constant 200X200 (NRA=NCA_RB=NCB=200), vary **(x-axis)** #threads 1, 2, 4, 8, 16. Output **(y-axis)** is total time. For each number of thread greater than 1 (i.e., 2, 4, 8, 16), indicate the speedup, if any, in the parenthesis).

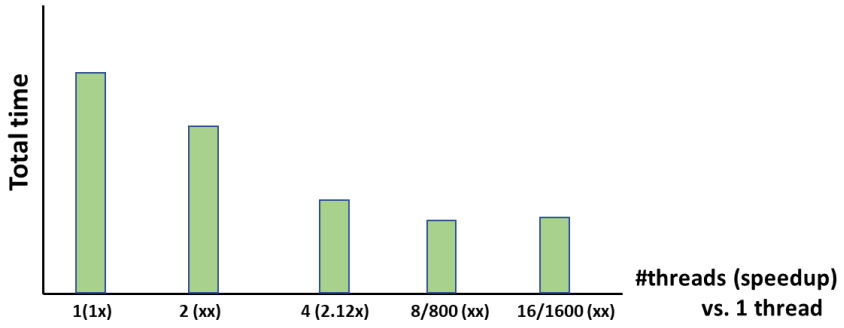

Figure 1: Strong-scaling graph

c) As a baseline for comparison, **keep number of threads = 1**; vary matrix sizes 100X100, 200X200, 400X400, 800X800, 1600X1600 (output is total time).

Then, **vary both #threads/matrix size**s: 2/200X200; 4/400X400; 8/800X800; 16/1600X1600

Using the above data, create a **"weak-scaling"** bar graph (as shown below in Figure 2)

- **Y-axis** again is the total time. For the **x-axis,** there will **5 entries**, as shown in Figure 2.
- The first entry is the baseline: one (green) bar of the result of 1 thread, N=100
- The last 4 entries have 2 bars each. The x-axis label **is #threads/N (speedup)**. For the example illustrated below, the label **4/400(2.48)** indicates:
  - Both bars are for N=**400** (400X400 matrix). The green bar is the total time of 1 thread; the red bar is for the total time of **4 threads** (2.48 is the **speedup of 4 vs 1 thread**)

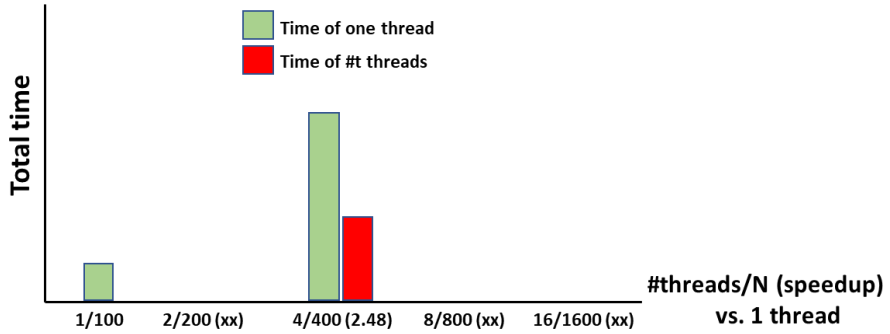

Figure 2: weak-scaling graph

Notes for SLURM script for OpenMP code: In the given **job-hw_omp.sh**, keep **nodes=1** and **ntasks=1**. However, for best performance, **cpus-per-task** should equal to the **number of threads**. For example, **cpu-per-task**=1 for 1 thread; cpu-per-task=4 for 4 threads; cpu-per-task=16 for 16 threads.

Part 2: Hybrid programming model: MPI+OpenMP

a) Using the sample MPI matrix multiply code (from Lab 1), modify it as follows and extend it to use both MPI and OpenMP.

b) These values are inputted through the command line:
- NRA: number of rows for Matrix A
- NCA_RB: number of rows for A and columns for B
- NCB: number of columns for B

c) In the **master** rank, use OpenMP to have **threads populate the matrices in parallel**. In the **worker** ranks, use OpenMP to have **threads to perform the calculation in parallel**.

d) Run the modified code with original matrix sizes (NRA=60, NCA_RB=12, NCB=10) and compare your result with the original (MPI only) code. Include this results in your report so I can see that your code functionally works.

e) **Create Table 1** with 4 rows:

- Use matrix size of **200X200** (NRA=NCA_RB=NCB=200):
  Row 1: 1 node, 8 ranks/node, 8 threads/rank, total time
  Row 2: 2 nodes, 4 ranks/node, 8 threads/rank, total time
  Row 3: 4 nodes, 2 ranks/node, 8 threads/rank, total time
  Row 4: 8 nodes, 1 rank/node, 8 threads/rank, total time

f) **Create Table 2** with 4 rows:

- Use matrix size of **1600X1600** (NRA=NCA_RB=NCB=1600):
  Row 1: 1 node, 8 ranks/node, 8 threads/rank, total time
  Row 2: 2 nodes, 4 ranks/node, 8 threads/rank, total time
  Row 3: 4 nodes, 2 ranks/node, 8 threads/rank, total time
  Row 4: 8 nodes, 1 rank/node, 8 threads/rank, total time

Note that, in parts (e) and (f), for each row (i.e., each run), 64 threads are used. However, these 64 threads are distributed differently and they must run within 32 cores (class limit).

Note for compilation:

- For OpenMP code: gcc -fopenmp yourcode.c -o …
- For MPI+OpenMP: mpicc -qopenmp yourcode.c -o …

~~**Notes for SLURM script:** In the given **job-hw_omp.sh**, keep **nodes=1** and **ntasks=1.** However, for best performance, **cpu s-per-task** should equal to the **number of threads**. For example, **cpu-per-task=**1 for 1 thread; cpu-per-task=4 for 4 threads; cpu-per-task=16 for 16 threads.~~

SUBMISSION INSTRUCTIONS

Make sure your name is at the top of every file, including the report.pdf. You are to submit 2 files on Canvas:

(1) Create a directory named **Lab3**. Use the following structure and zip the entire directory and submit the **zip file** on Canvas using the following name: lastNamefirstNameLab3.zip

Lab3/Part1

.c file
batch script(s)
All output files used to produce Figures 1 and 2:

- Name each output file appropriately: e.g., Part1_Fig1_8threads, Part1-Fig2_16_1600.
- Also, to maximize your credit, use printf statements to provide partial results so I can give your partial credit.

Lab3/Part2

.c file
batch script(s)
All output files used to produce Tables 1 and 2:

- Name each output file appropriately.
- Again, to maximize your credit, use printf statements to provide partial results so I can give your partial credit.

(2) **A pdf file** (Lab 3 report) using the following name: lastNamefirstNameLab3.pdf

- Put your name at the top of the report and any information or explanation that you want to give me.
- **Part 1:**
  - After you have run the OpenMP code with original matrix sizes (NRA=60, NCA_RB=12, NCB=10), include the printout of this result here so I can see it functionally works.
  - **Figure 1** and a paragraph discussing the figure (strong scaling).
  - **Figure 2** and a paragraph discussing the figure (weak scaling).
- **Part 2:**
  - After you have run the MPI-OpenMP code with original matrix sizes (NRA=60, NCA_RB=12, NCB=10), include the printout of this result here so I can see it functionally works.
  - **Table 1** and a paragraph discussing the table.
  - **Table 2** and a paragraph discussing the table.