Masterpraktikum Scientific Computing

High Performance Computing Solution Sheet

Session 2: OpenMP

Authors
Mantosh Kumar (Matriculation number : 03662915)
João Trindade (Matriculation number : 03673251)

Course of Study: Master of Science Informatics

Exercise 1: "Shared memory - PI calculation"

How to compile, generate binaries, execute using automated test suite:

1. $ sudo apt-get install python-matplotlib (Used for publication quality plots and if you wish to watch the plot being generated in front of you then only, otherwise skip it)
   Note: If you don't have this package then you can't generate the plots as none of the python file could be executed successfully. Instructions dependent on this package is denoted by [@]
2. $ cd tutorial_2/exercise 1
3. $ cp scripts/* $PWD

Observation 1: OpenMPs vs Non-OpenMP and speedup calculation

Analysis of PI calculation in terms of problem sizes, computation time, speedup and efficiency for both Non-OpenMP and OpenMP environment.
To run the automated test cases and generate the plot please execute these following command.

```
$ make clean
$  make
$ sh only_load_increase.sh      (this will generate the data)
[@]  $ python   plot_time_div_all_method_compare.py
[@]  $ python   plot_speedup_compare.py
```

Generated data locations and format:
The generated data are stored in ".txt" format.

"**./generated_data/log_non_openmp_time_vs_prblmSize.txt**"
"**./generated_data/log_r_time_prlm_sz.txt**"
"**./generated_data/log_c_time_prlm_sz.txt**"
"./generated_data/log_r_speedup_vs_prblm_sz.txt"
"./generated_data/log_c_speedup_vs_prblm_sz.txt"

Generated Plot: "**./generated_plots/comparision.png"**
"./generated_plots/speedup_comparision.png"

Observation 2 : Weak Scaling for Reduction

To analyse PI calculation in terms of weak scaling please follow following steps.
To run the automated test cases and generate the plot please execute these following commands.

```
$ make clean
$ make
$ sh   weak_scaling.sh      (this will generate the data)
[@]  $ python   plot_1.py
[@]  $ python   plot_3.py
```

Generated data locations and format:
The generated data are stored in ".txt" format.
'./generated_data/log_non_openmp_time_vs_prblmSize.txt'
./generated_data/log_r_time_prlm_sz.txt'
./generated_data/log_r_eff_vs_prblm_sz.txt'
./generated_data/log_r_speedup_vs_prblm_sz.txt'

./generated_data/log_r_eff_vs_threads.txt'
'./generated_data/log_r_time_vs_threads.txt'
./generated_data/log_r_speedup_vs_threads.txt'
./generated_data/log_r_ideal_eff.txt'

Generated Plots: "./generated_plots/weak_scaling_r_time_eff_speedup_VS_size.png"
"./generated_plots/scaling_time_eff_speedup_VS_threads.png"

Observation 3 : Weak Scaling for Critical

To analyse PI calculation in terms of weak scaling please follow following steps.
To run the automated test cases and generate the plot please execute these following commands.

$ make clean
$ make
$ sh weak_scaling.sh (this will generate the data)
[@] $ python plot_2.py
[@] $ python plot_4.py

Generated data locations and format:

The generated data are stored in ".txt" format.
'./generated_data/log_non_openmp_time_vs_prblmSize.txt'
./generated_data/log_c_time_prlm_sz.txt'
./generated_data/log_c_eff_vs_prblm_sz.txt'
./generated_data/log_c_speedup_vs_prblm_sz.txt'
./generated_data/log_c_eff_vs_threads.txt'
'./generated_data/log_c_time_vs_threads.txt'
./generated_data/log_c_speedup_vs_threads.txt'
./generated_data/log_c_ideal_eff.txt'

Generated Plots: "./generated_plots/weak_scaling_c_time_eff_speedup_VS_size.png"
"./generated_plots/scaling_time_eff_speedup_VS_threads.png"

Observation 4 : Strong Scaling

To analyse PI calculation in terms of strong scaling please follow following steps.
To run the automated test cases and generate the plot please execute these following command.

$ make clean
$ sh strong_scaling.sh (this will generate the data)
[@] $ python plot_4.py (this will plot graph)
[@] $ python plot_5.py (this will plot graph)

Generated data locations and format: look into "./generated_data" directory.
The generated data are stored in ".txt" format.

Generated Plots: **"./generated_plots/scaling_time_eff_speedup_VS_threads.png"**
"./generated_plots/strong_red.png"

Exercise 2: "STREAM Benchmark"

Instructions.
Please start by compiling every file with: make all
For exercise 2.2 run the script run_cache_16_core.sh
This will set the OMP_NUM_THREADS env. var to 16 and run the stream benchmark once.

For exercise 2.3 run the script run_cache_8_core.sh
This will set the OMP_NUM_THREADS env. var to 8 and run the stream benchmark twice with diferent values for the KMP_AFFINITY env. variable.

For exercise 2.4 run the script run_remote_test.sh
This will set the OMP_NUM_THREADS env. var to 8 and the KMP_AFFINITY variable and run the stream benchmark so that the data is stored in the memory of the processor that is not executing the majority of the tasks.

For exercise 2.5 run the script run_cache_test.sh
This will set the OMP_NUM_THREADS env. var to 1 and run the stream benchmark 3 times with different N values

Exercise 3: "Quicksort"

Instructions.
Please start by compiling every file with: make all
Run ./quicksort <problem_size> <number_of_threads>
OR
Run the script test_prob_size.sh, this script will execute the quicksort program for a problem size of 2'000'000 (2 Bilion) starting with 2 threads, up to 28 threads.

Exercise 4: "Matrix-Matrix-Multiplication II"

How to compile, generate binaries, execute using automated test suite:

4. $ sudo apt-get install python-matplotlib (Used for publication quality plots and if you wish to watch the plot being generated in front of you then only, otherwise skip it)
   Note: If you don't have this package then you can't generate the plots as none of the python file could be executed successfully. Instructions dependent on this package is denoted by [@]
5. $ cd tutorial_2/exer4
6. $ cp scripts/* $PWD

Observation 1: OpenMP vs Non-OpenMP

Analysis of matrix multiplication in terms of problem sizes, computation time, speedup and efficiency for both Non-OpenMP and OpenMP environment.
To run the automated test cases and generate the plot please execute these following command.

```
$ make clean
$ sh  seq_vs_parallel_diff_prob_sizes_vs_matrix_size.sh     (this will generate the data)
[@]  $ python   plot_1.py
```

Generated data locations and format:
The generated data are stored in ".txt" format.

- Non-OpenMP problem size - computation time data:
  "**./generated_data/log_non_opemp_time_vs_prblmSize.txt**"
  Format: Problem Size - Computation time
- OpenMP problem size - computation time data: "**./generated_data/log_openmp_time_prlm_sz.txt**"
  Format: Problem Size - Computation time
- OpenMP problem size - Speedup data: "**./generated_data/log_openmp_speedup_vs_prblm_sz.txt**"
  Format: Problem Size - Speedup
- OpenMP problem size - Efficiency data:
  "**./generated_data/log_openmp_eff_vs_prblm_sz.txt"**
  Format: Problem Size - Efficiency

Generated Plot: "**./generated_plots/sacaling_time_eff_speedup_VS_size.png"**

Observation 2 : Weak Scaling

To analyse matrix multiplication in terms of weak scaling please follow following steps.
To run the automated test cases and generate the plot please execute these following commands.

```
$ make clean
$ sh  weak_scaling.sh     (this will generate the data)
[@]  $ python   plot_1.py
[@]  $ python   plot_3.py
```

Generated data locations and format:
The generated data are stored in ".txt" format.

- Non-OpenMP problem size - computation time data:
  "**./generated_data/log_non_opemp_time_vs_prblmSize.txt**"
  Format: Problem Size - Computation time
- OpenMP problem size - computation time data: "**./generated_data/log_openmp_time_prlm_sz.txt**"
  Format: Problem Size - Computation time
- OpenMP problem size - Speedup data: "**./generated_data/log_openmp_speedup_vs_prblm_sz.txt**"
  Format: Problem Size - Speedup
- OpenMP problem size - Efficiency data: "**./generated_data/log_openmp_eff_vs_prblm_sz.txt"**
  Format: Problem Size - Efficiency
- OpenMP Number of threads - Efficiency data: **"./generated_data/log_openmp_eff_vs_threads.txt"**
  Format: Number of threads - Efficiency
- OpenMP Number of threads - Speedup data: **"./generated_data/log_openmp_speedup_vs_threads.txt"**
  Format: Number of threads - Speedup
- OpenMP Number of threads - Computation time data:
  "./generated_data/log_openmp_time_vs_threads.txt"

Format: Number of threads - Computation time

Generated Plots: "**./generated_plots/scaling_time_eff_speedup_VS_size.png**"
"**./generated_plots/scaling_time_eff_speedup_VS_threads.png**"

Observation 3 : Strong Scaling

To analyse matrix multiplication in terms of strong scaling please follow following steps.
To run the automated test cases and generate the plot please execute these following command.

$ make clean
$ sh strong_scaling.sh (this will generate the data)
[@] $ python plot_3.py (this will plot graph)

Generated data locations and format:
The generated data are stored in ".txt" format.

- Non-OpenMP problem size - computation time data:
  "**./generated_data/log_non_opemp_time_vs_prblmSize.txt**"
  Format: Problem Size - Computation time
- OpenMP problem size - computation time data: "**./generated_data/log_openmp_time_prlm_sz.txt**"
  Format: Problem Size - Computation time
- OpenMP problem size - Speedup data: "**./generated_data/log_openmp_speedup_vs_prblm_sz.txt**"
  Format: Problem Size - Speedup
- OpenMP problem size - Efficiency data: "**./generated_data/log_openmp_eff_vs_prblm_sz.txt"**
  Format: Problem Size - Efficiency
- OpenMP Number of threads - Efficiency data: **"./generated_data/log_openmp_eff_vs_threads.txt"**
  Format: Number of threads - Efficiency
- OpenMP Number of threads - Speedup data: **"./generated_data/log_openmp_speedup_vs_threads.txt"**
  Format: Number of threads - Speedup
- OpenMP Number of threads - Computation time data:
  "./generated_data/log_openmp_time_vs_threads.txt"
  Format: Number of threads - Computation time

Generated Plots: **"./generated_plots/scaling_time_eff_speedup_VS_threads.png"**