**CSE-443/543: High Performance Computing**

**Exercise #6**

Max Points: 20

|  |
| --- |
| **You should save/rename this document using the naming convention MUid\_ex6.docx (example: raodm\_ex6.docx).**  **Objective**: The objective of this exercise is to:   * Understand the statistics reported by /usr/bin/time   Fill in answers to all of the questions. For almost all the questions you can simply copy-paste appropriate text from the shell/output window into this document. You may discuss the questions with your instructor. |

|  |  |
| --- | --- |
| **Name:** | **Maciej Wozniak** |

# Part #1: Setting up the exercise on OSC

*Estimated time: < 10 minutes*

## Exercise

1. Log into OSC’s OnDemand portal via <https://ondemand.osc.edu/>. Login with your OSC id and password.
2. Startup a VS-Code server and connect to VS-Code. Ensure you switch to your workspace. Your VS-Code window should appear as shown in the adjacent screenshot.
3. Next, create a new VS-Code project in the following manner:
   1. Start a new terminal in VS-Code
   2. In the VS-Code terminal use the following commands:

|  |
| --- |
| $ # First change to your workspace directory  **$ cd ~/cse443**  $ # Use ls to check if workspace.code-workspace file is in pwd  $ # Next copy the basic template for a C++ project  **$ cp -r /fs/ess/PMIU0184/cse443/templates/basic exercise6**  $ # Copy the starter code for this exercise  **$ cp /fs/ess/PMIU0184/cse443/exercises/exercise6/\* exercise6** |

1. Now add the newly created exercise4 directory to VS-Code Briefly study the starter code in main.cpp. It is a very simple program that generates a lot random numbers, sorts them and prints the middle value.
2. Now from the terminal (not via NetBeans) compile the program as shown below. **Note**: the whole command is on 1-line but appears wrapped due to the line being long. The flags “-fopenmp -D\_GLIBCXX\_PARALLEL” flags enable automatic multithreading of the std::sort() algorithm.

|  |
| --- |
| $ g++ -g -Wall -std=c++17 -O3 -fopenmp -D\_GLIBCXX\_PARALLEL main.cpp -o main |

1. Next create an interactive job on the cluster to reserve 8-cores on 1-compute node:

|  |
| --- |
| $ sinteractive -A PMIU0184 -N 1 -n 8 |

1. Once the job has started run the program using /usr/bin/time command using different numbers of threads (by changing OMP\_NUM\_THREADS=*n*, change the value of *n* to 1, 2, 4, 6, and 8) using the command below and record the runtime statistics in the table below. Run each thread setting 3 times and record the values from the 3rd run.

|  |
| --- |
| $ OMP\_NUM\_THREADS=n /usr/bin/time -v ./main |

**Observations:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **n** | **Time in seconds** | | | **%CPU** |
| **User** | **System** | **Elapsed** |
| 1 | 10.76 | 0.08 | 0:10.92 | 99 |
| 2 | 11.23 | 0.18 | 0:06.19 | 184 |
| 4 | 11.47 | 0.19 | 0:03.68 | 316 |
| 6 | 12.63 | 0.2 | 0:02.92 | 439 |
| 8 | 12.95 | 0.19 | 0:02.45 | 535 |

|  |  |
| --- | --- |
| Macintosh HD:private:var:folders:2f:zrk8zvtn1xq6xmt2ws8l3lzh9vz293:T:TemporaryItems:stopsignchinese.png | Once you have recorded your timings, double check your timings with your neighbor or your instructor. |

1. Finally stop your interactive job using the exit command.

# Part #2: Inferences

Using the above observations draw inferences about the following runtime characteristics of the program:

1. From the lecture, briefly describe what does user time indicate?

|  |
| --- |
| total cpu time for running instructions of user s code |

1. From the lecture, briefly describe what does elapsed time indicate?

|  |
| --- |
| runtime of program (as measured by stopwatch) **the most important for many ppl** |

1. From the lecture, briefly describe what does %cpu indicate?

|  |
| --- |
| e.g. if %cpu is 190%, on average 1.9 CPUs were used for this job |

1. Why does the elapsed time of the program decrease as the number of threads are increased?

|  |
| --- |
| Instructions are divided on multiple cores, decreases elapse time, each core has “less” things to do |

1. If elapsed time is decreasing, then why does user time of the program slightly increase as the number of threads are increased?

|  |
| --- |
| We have same number of instructions but we are spreading them among multiple nodes, which still has to do the same job |

# Part #3: Submit files to Canvas

Upload the following files to Canvas:

1. Upload this MS-Word document (duly filled with the necessary information) saved as PDF using the naming convention **MUid\_ex6.pdf**.