**DUE DATE:** **By the deadline specified on Canvas**

# CSE-443/543: High Performance Computing

# Exercise #7

Max Points: 20

**You should save/rename this document using the naming convention MUid_ex7.docx (example: `raodm_ex7.docx`).**

**Objective**: The objective of this exercise is to:

- Understand creating and running microbenchmarks
- Review the process of running jobs on the OSC cluster.
- Determine if a `switch` statement can improve performance when compared to a semantically equivalent `if-else` construct using a micro-benchmark.

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**

Warnings from C++ compilers should be taken seriously because in many cases warnings indicate a potential runtime problem with your program. Consequently, all grading, particularly for homework, will require that programs compile without warnings or style issues. Points will be deducted for each warning generated by the compiler and style issues programs.

## Background

C++ provides two different approaches to implement control flow either with a series of `if-else` statements or using a `switch` statement. Recollect that in C++, the `switch` statement can be used only with primitive data types (such as `int`), which gives the compiler an opportunity to optimize the `switch` statement (if possible). Often there are questions and discussions (search `stackoverflow.com` if you are interested) about the performance difference between `if` vs. `switch`.

### *Scientific question:*

Is there a performance difference between using a series of `if-else` statements versus using a `switch` statement?

### *Hypothesis:*

When a `switch` statement is used for a contiguous range of `int` values (say 0…9) the compiler can better optimize the `switch` statement when compared to a semantically equivalent series of `if-else` statements.

## Design of Microbenchmark

In order to verify the above hypothesis (i.e., accept or not accept the hypothesis) the following operations needs to be performed:

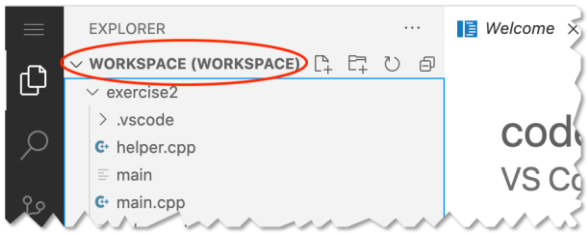
1. A suitable Microbenchmark needs to be developed with semantically equivalent `if-else` statements and `switch` statements that operate on a contiguous range of integer values.
2. Runtime of the benchmark using the 2 constructs to measure runtime
3. Compare runtime with different optimization levels to see if a difference in runtime is observed.
4. Using the runtime data conclusions need to be drawn if one construct is better than the other in the given scenario.

## Part #1: Completing the microbenchmark

*Estimated time: < 15 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.




1. Next, create a new VS-Code project in the following manner:
   a. Start a new terminal in VS-Code
   b. 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 exercise7
$ # Copy the starter code for this exercise
$ cp /fs/ess/PMIU0184/cse443/exercises/exercise7/* exercise7
```

2. Now add the newly created `exercise7` directory to VS-Code Briefly study the starter code in `main.cpp`.
3. Now using the implementation in `useSwitch` method as reference, implement the same functionality in `useIf` method using a series of `if-else` statements as suggested in the adjacent code fragment.

```
int useIf(const int i) {
    if (i == 0) {
        return 1;
    } else if (i == 1) {
        return 3;
    } else if (i == 2) {
        //…
```

4. Validate your implementation using the following command-line for a short run (it should run in < 1 second):

```
$ ./Exercise6 switch 5
Sum = 8845740
$ ./Exercise6 if 5
Sum = 8845740
```

   Needless to add, both versions (`switch` and `if-else`) should produce the same results.

## Part #2: Recording observations

***Expected time for completion 10 minutes***

1. **Apparatus (experimental platform)**
   The apparatus used for the experiments documented in this report were conducted on the following platform (replace commands shown in grey with correct information obtained using the command in grey):

| Component | Details |
|---|---|
| Hostname | pitzer-login02.hpc.osc.edu |
| CPU Model | Intel(R) Xeon(R) Gold 6148 CPU @ 2.40GHz |
| CPU/Core Speed | 1195.019MH |
| Main Memory (RAM) size | 19687328 kb |
| Operating system used | Linux |

2. Review the supplied SLURM script for this project. Note how the script compiles and runs the program.
3. Submit your SLURM job as

```
$ sbatch Exercise7_slurm.sh
```

   The above job will be queued and then take several seconds to start and run. Once the job ends, you will see an output text file in the format `slurm-<`*JobID*`>.out` (e.g., `slurm-12345.out`) with the results.

4. Next modify the compile command in `Exercise7_slurm.sh` script to use `-O3` (capital oh 3) optimization-level compiler flag as shown below:

```
OPT_LEVEL="-O3"
```

5. Submit another job to obtain data at `-O3` optimization level.

6. Once your jobs have finished record all the 12 timings in the observations table below.

## Observations:

Record the timing information collated from your experiments conducted using your micro-benchmark in the tables below:

| Using if-else statements | Elapsed Time (sec) measured using `/usr/bin/time` command | |
|---|---|---|
| | **Opt: -O2** | **Opt: -O3** |
| Observation #1 | 11.54 | 7.05 |
| Observation #2 | 11.5 | 7.03 |
| Observation #3 | 11.4 | 7.05 |
| **Averages (of 3 runs)** | **11.48** | **7.043** |

| Using switch statements | Elapsed Time (sec) measured using `/usr/bin/time` command | |
|---|---|---|
| | **Opt: -O2** | **Opt: -O3** |
| Observation #1 | 6.3 | 7.23 |
| Observation #2 | 6.29 | 7.26 |
| Observation #3 | 6.35 | 7.22 |
| **Averages (of 3 runs)** | **6.31** | **7.243** |


Once you have recorded your timings, double check your timings with your neighbor or your instructor.

### *Results and Conclusions*

Using the above data develop a report (about 4-to-5 sentences) discussing the performance aspects and conclude if you accept or reject the hypothesis. Indicate what suggestion you would provide to a programmer based on your experiment.

We can clearly see that -O3 optimized compiler outperforms -O3 with more advanced functions such as *switch*. Even though compiler speeds up *switch* it performs slightly worse in simple *else-if* based function. This can be cause by a variety of things, but the operation is simpler, we do not need to use more advanced compiling methods.

## 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_ex7.pdf`**.