Lecturer:
Prof. Dr. Florina M. Ciorba

Computer Lab. U1.001
Spiegelgasse 1,
CH-4051 Basel

Assistants:
Aurélien Cavelan, Ph.D.
Danilo Guerrera, M.Sc.
Ahmed Eleliemy, M.Sc.
Ali Mohammed, M.Sc.

http://informatik.unibas.ch/fs2018/lecture-high-performance-computing/

High Performance Computing (17164-02) **Spring Semester 2018**

Assignment 5: Correctness, Fault tolerance, Performance analysis, and Reproducibility

(20 + 5 (discussion) Points)

Starting Date: May 03, 2018
Deadline: May 23, 2018 - 23:59:59

Objectives:
1. Use MUST to check the code correctness of a given MPI code.
2. Use algorithm-based fault tolerance (ABFT) technique to overcome
silent data corruption (SDC) in matrix multiplication code.
3. Use Score-P and Vampir to analyze the performance of a given application.
4. Enhance performance of a given application based on the performance analysis.
5. Experience the importance and usefulness of the reproducibility of experiments.

Please make sure that your optimizations do not affect the correctness of the results.
The delivered solution should be in one tar file.

1 Code correctness *(5 Points)*

Given the file *T1.c* that contains certain MPI code, it is required to check the correctness of this code using MUST, exactly as explained in the lecture.
To use MUST on the miniHPC type the following commands:

```
ml MUST
```

To compile using OpenMPI on miniHPC, type the following command:

```
mpicc T1.c —o T1
```

Hints:
1. Type mustrun -h to know how to use this command.
2. use firefox to read the output of MUST by typing firefox output_file_path

a. Deliver the output of the MUST and discuss the root-cause of the reported error. *(2 Points)*

b. Propose, implement, and discuss the required correction(s). *(2 Points)*

c. Deliver the output of the MUST for your corrected version of T1.c *(1 Point)*

2 Fault tolerance *(5 Points)*

Given the file *mat_mul_ABFT.pdf* that contains slides explaining an algorithm-based fault tolerant technique for matrix multiplication, you are required to modify the code in the file *T2.c* of the naive matrix multiplication code to implement the described ABFT technique in the slides.

a. Implement the `TODOs` in the *T2.c* to allocate the needed extra rows and columns and compute the checksums and do the multiplication. *(2 Points)*

b. Implement the `injectSDC()` function to simulate an SDC by randomly changing the value of an element in the output matrix. *(1 Point)*

c. Implement the `detect_correct()` function to detect the SDC in the output matrix and correct it. *(2 Points)*

3 Performance analysis *(5 Points)*

In this exercise you are going to use the performance tools available on miniHPC to examine the execution of a multi-threaded parallel sparse matrix multiplication.
To use Score-P

```
ml Score—P
```

To compile with Score-p instrumentation

```
scorep gcc —fopenmp —O3 T3.c —o T3
```

To run the instrumented code and generate a trace of the execution

```
export SCOREP_ENABLE_TRACING=true
export SCOREP_TOTAL_MEMORY=20MB
srun ./T3 1000
```

The output trace will be in a folder named *scorep_datetime*. The generated trace file is named *traces.otf2*. To visualize the obtained execution trace *traces.otf2* via Vampir.

```
vampir
```

Given *T3.c* which implements the naive sparse matrix multiplication using OpenMP. It is required to use the Score-P to instrument this code and produce a trace of its execution.

a. Compile the given code without instrumentation, run and report the performance for 16 threads and matrix size $1,000$. *(1 Point)*

b. Instrument the code using Score-P, generate a trace for its execution on 16 threads and matrix size $1,000$. Deliver a screenshot of the trace visualization using Vampir. *(1 Point)*

c. Discuss the potential reasons that degrade the performance of the application and how to overcome them. *(1 Point)*

d. Implement your suggestions to enhance the performance of the application and report the performance of the enhanced version. *(1 Point)*

e. Instrument the enhanced version using Score-P, generate a trace for its execution on 16 threads and matrix size $1,000$. Deliver a screenshot of the trace visualization using Vampir. *(1 Point)*

4 Reproducibility *(5 Points)*

Provided the code of a former student of the HPC course for the matrix multiplication using MPI + OpenMP in the file *T4.c* and the results in the file *mat_mul_MPI_OpenMP.pdf*, you are required reproduce the performance reported in the results file. The code was run for numbers of MPI ranks = $\{1, 2, 4\}$ and numbers of threads = $\{2, 4, 8, 16\}$ for matrix size of $5,000$ using intel compiler on miniHPC system.

Write a short report (one page maximum) with your reproduced results, discuss how successful was the reproducibility of the results of the former student, and what extra information was needed to be reported other than the provided in this task to replicate the exact results. *(5 Points)*