Assignment 2: Cluster Computing

Background:

In this assignment you are going to work with a parallel code written in C using the OpenMP library[1]. The necessary files that need to be used are indicated in the or can be found in your home directory on Snellius[2] (`$HOME/linux-cluster-computing/cluster/batch/`)

Assignment:

Analyse of the computation of PI based on the Leibniz Series[3, 4]

Given the previous introduction, perform the following tasks:

- **Task1**: Thread Scalability Study for Parallel Pi Computation
  - Compile and run a sequence of pi computations with different number of threads: from 8 threads, you should increase the number of threads in 4 for every successive simulation, until you reach 48 threads.
  - Plot a graph to show the execution times of the computation (Y-axis) with respect to the amount of threads (X-axis).
  - **Question**: How would you explain the observed performance, considering that you have requested 32 cores from a node on Snellius?
- **Task2**: Execution Time Analysis of Pi Computation
  - Compile the code with 31250000, 62500000, 125000000, 250000000, 500000000, 1000000000 and 2000000000 numbers in the Leibniz series (changing the value of "**`niter`**") and run the pi computation only with **`32 threads`** in all cases.
  - Plot a graph to show the execution times of the computation (Y-axis) with respect to the amount of iterations (X-axis).
  - **Question**: How does the execution time scale with the number of iterations in the Leibniz series?

Submission:

Submit a short report in Canvas before the deadline, including the graphs and answers to the questions.
Note: Submitting after the deadline will result in losing points (1 point for every 30 minutes after the deadline)

[1] The parallel code is provided, you don't need to understand it, there will be a full workshop on parallel programming (OpenMP and MPI)

[2] https://www.surf.nl/diensten/snellius-de-nationale-supercomputer

[3] https://en.wikipedia.org/wiki/Leibniz_formula_for_π

[4] https://www.youtube.com/watch?v=NaL_Cb42WyY

Code and Scripts:

- The code needs to be compiled with the following commands on Snellius:

```
#!/bin/bash
module load 2022
module load GCCcore-11.3.0
gcc -fopenmp -o pi pi.c -lm
```

- The basis script to submit the job would be as follows:

```
#!/bin/bash
#SBATCH --job-name="pi"
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --cpus-per-task=32
#SBATCH --time=00:10:00
#SBATCH --partition=normal
#SBATCH --output=pi_%j.out
#SBATCH --error=pi_%j.err
module purge
module load 2022
module load GCCcore-11.3.0
echo "OpenMP parallelism"
echo
for ncores in `seq 8 4 48`
do
export OMP_NUM_THREADS=$ncores
echo "CPUS: " $OMP_NUM_THREADS
echo "CPUS: " $OMP_NUM_THREADS >&2
./pi
echo "DONE "
done
```

- C PI program:

```
#include <omp.h>
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

int main (int argc, char *argv[])
{

  //initialize variables
  int i;
  double pi = 0;
  int niter = 1000000000;

  // Get timing
  double start,end;
  start=omp_get_wtime();
  // Calculate PI using Leibnitz sum

  /* Fork a team of threads */
  #pragma omp parallel for reduction(+ : pi)
  for(i = 0; i < niter; i++)
  {
     pi = pi + pow(-1, i) * (4 / (2*((double) i)+1));
  }
  /* Reduction operation is done.
     All threads join master thread
     and disband */

  // Stop timing
  end=omp_get_wtime();

  // Print result
  printf("Pi estimate: %.20f, obtained in %f seconds\n", pi,
end-start);

}
```