High Performance Computing
Exercise Sheet 3

FS 21
Dr. Douglas Potter

`http://www.ics.uzh.ch/`

Teaching Assistants:
Jozef Bucko, jozef.bucko@uzh.ch
Hugues de Laroussilhe, hugues@ics.uzh.ch

Issued: 12.03.2021
Due: 17.03.2021

Exercise 1 [Linking]

You can find a serial version of the program of last week `cpi.c` (and the other files needed for this session) in the folder `exercise_session_03` of `https://github.com/hlasco/hpc_esc_401`. You can easily get them by "pulling" the repository you cloned in your `$HOME` directory last week. Remember to commit your solutions in your personal git repository.

- Extract the `getTime()` function and make a separate module out of it, which can be called from your main program.
- Modify the serial version so that it includes your new module. Add instructions to the makefile to compile the new module and the new serial version.
- Modify the MPI version `cpi_mpi.c` to call `getTime()` in the beginning of the program (before `MPI_Init()`) and at the end (after `MPI_Finalize()`). Print the subtracted times to get your execution time. Adapt the makefile, compile and run the program. Two execution times will be outputted.
- ***Commit :*** What are the values of the execution times? Why are they different?

Exercise 2 [Compiler optimization]

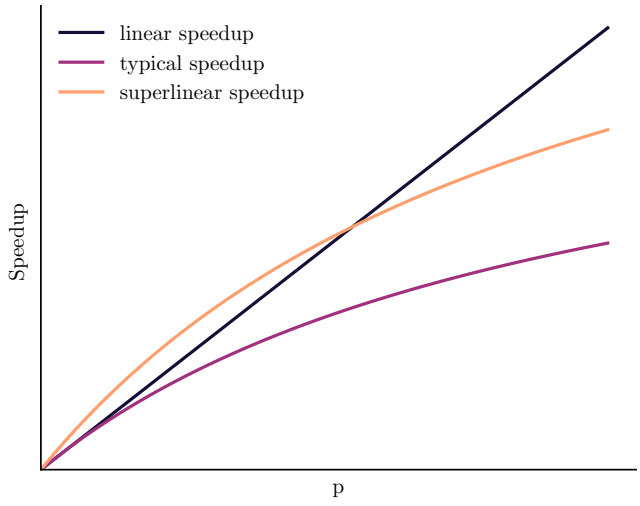
- Copy the file `sum.c` to your repository and compile. Time your code by using `getTime()` from Exercise 1.
- Compile with different optimisation flags (`-O0`,`-O1`,`-O2`,`-O3`).

- ***Commit :*** What results does your code produce? Do you know other ways to get timings? How big is the difference in execution time with the different flags? How did the compiler optimize the code? How can you make it even faster?
- Parallelize the code by adding the following line

  ```
  #pragma omp parallel for reduction(+ : sum)
  ```

  and including the appropriate header. Hint: think what part needs to be parallelized. Compile and run the code in parallel.
- ***Commit :*** Where did you insert the line? How did you compile? Provide the job script.
- Run the code with various number of cpus (but don't overdo it!).
- ***Commit :*** Plot the speedup and discuss.

Exercise 3 [BONUS: Scaling behaviour]



In some situations you can encounter what is called superlinear speedup. This is when a parallel program has a speedup greater than the number of processes. How can you explain such behaviour?