# Programming of Supercomputers

## Assignment 2
## Milestone 4 - Performance Tuning and Final Report

Deadline: 26.01.2015 @ 8:00 CET

The goal of this last milestone is to achieve an optimal version of your parallel implementation of the Fire benchmark.

## 1 Tools

You are asked to make use of the following tools available on the SuperMUC machine:

- Score-P
  `module load scorep`
  Please make sure to use the default version of Score-P on SuperMUC, which is the one using the `mpi.ibm` module. This is different from the `mpi.intel` module used in the previous milestones.
- Cube
  `module load cube`
- Vampir
  `module load vampir/8.0`
  Note that the default version, Vampir 7.5, cannot read the `.otf2` files generated by Score-P.

You might also find useful the parallel debugger TotalView, also available on SuperMUC. Use `module load totalview`.

## 2 Minimal performance requirements

The minimal performance requirement for the milestone to be accepted is: linear speedup of the computation phase up to 8 processes for `pent` input geometry using the `dual` distribution.

Please note that this is only the minimum to be achieved. You are encouraged to optimize your code any better.

## 3 Performance analysis

Besides achieving a minimal performance of your code, you are also asked to carry out a basic set of measurements and corresponding analysis:

1. compare the scalability of the `oneread` and `allread` input algorithms for `drall` and `cojack` input files using different number of processes. Consider both execution time and amount of exchanged data.

2. find out the number of processes up to which there is an increase in speedup for the *computation* phase for the `pent` and `cojack` files, for all distribution algorithms. For the speedup graphs use as reference the parallel execution with one process. Also include in your report a comparison to the serial execution time.

3. compare the MPI overhead of the three execution phases, *initialization*, *computation* and *finalization*, for the `pent` file on 8 processes. Input algorithm and distribution at choice.

4. analyse one iteration of the *computation* phase for a configuration at your choice. Check the MPI overhead, map execution steps (from the timeline view) to the corresponding source code operations. Identify regions with low number of floating point operations.

# 4 Performance optimization

Any of the above analysis steps could result in optimization ideas for your code. Present any identified bottlenecks and the corresponding optimizations in your report. Do not forget to also run and present the validation of the implemented optimizations. Submit to the repository both the original version of the code, as well as the modified one.

Some transformations might not result in the expected improvements. Include them nevertheless in your report.

For the current milestone we are evaluating both the final result of your work, i.e. the optimized implementation, as well your skills in using the performance tools and removing bottlenecks.

Please also note that your code should have a good performance, but also a good coding style. Carefully comment your sources, eliminate code replication and avoid longish functions.

# 5 Final report

The final report should prove your thorough understanding of all the implementation steps of this lab course.

Give a short sum up of the observations you have made while solving the first assignment.

For the second assignment, present the key issues you had to consider while solving milestones 1, 2, and 3. Include any particular algorithms or strategies you might have used.

Allocate more space for presenting the fourth milestone. Include here the analysis steps you have carried out, along with graphs and **screenshots**. You should clearly specify the configuration used for each of the presented graphs and images. Highlight the optimization potential you discovered and the results achieved with the tuned version of your implementation.

The report should be between 8 and 15 pages (font size at your choice :-) ).

## Submission

Add to your git repository the appropriate folders for this new milestone:

- folder `A2.4/code/final`: *.c, *.h, and Makefile of your final (best) solution
- folder `A2.4/code/optimizationX/original`: *.c, *.h, and Makefile
  folder `A2.4/code/optimizationX/optimized`: *.c, *.h, and Makefile
  where X is an index for your optimization step.
- folder `A2.4/plots/`:
  - pent_computation_speedup_1-8proc.png
  - drall_inputalgorithm_scalability.png
  - cojack_inputalgorithm_scalability.png
  - pent_computation_speedup.png
  - cojack_computation_speedup.png
  - pent_MPIoverhead_8proc.png
- folder `A2.4/data`:
  - *.xlsx or *.ods files with measured data and generated graphs.
  - folder `A2.4/data/optimizationX`: profile and/or tracing results of the validation step for the current optimization (results folder generated by scorep).
- folder `A2.4/scripts`: job scripts and any other scripts you might have used.
- folder `A2.4/report`: *teamXX_Surname1_Surname2*.pdf - final report in pdf format