Programming of Supercomputers

Assignment 2
Milestone 1 - Data Distribution
03.11.2014

Deadline: `17.11.2014 @ 08:00 CET`

The goal of the entire second assignment is to generate a parallel optimal version of the Fire benchmark. In the first milestone you are asked to implement the distribution of the computation domain and of the initial data.

1 Domain distribution

For the distribution among the processes of the computation domain, you should implement the following strategies:

1. one *classic* distribution, where the volume cells are distributed in the order given by the input file: first k cells to the first process, next k cells to the second one, and so on;
2. two *metis* distributions, where the volume cells are distributed based on the two mesh partitioning algorithms: *dual* and *nodal*.

Please note that in the Fire benchmark the computation domain is given by the internal cells, but values from the external cells are also being accessed at computation time.

Also note, with respect to the implementation, that:

- users must be able to choose at command line between the three strategies.
- code replication is strongly discouraged.

2 Input data

For reading the initialization data, you are required to implement the following strategies:

1. *oneread* approach, where the master process reads all data and then sends the appropriate chunks of data to all other processes;
2. *allread* approach, where all processes read in the input files, but then only store the data which is needed locally for each of the processes.

3 Test and evaluation

In the last milestone of this assignment, Milestone 4, you are going to carry out complete performance analysis of your code. For all the milestones before that, you are only asked to implement some test and partial evaluation functions.

3.1 Execution time

Instrument your code with PAPI for measuring the initialization time. Using the given `write_pstats_exectime()` function, output the results for the `drall.geo.bin` and `cojack.geo.bin` input files for executions on 4 and 12 processes and for all combinations of the implemented strategies.

3.2 Partitioning testing

Using the `write_pstats_partition()` function, output the number of internal and external local cells for each of the distribution strategies. Use the `pent.geo.bin` and `cojack.geo.bin` input files and execute with 9 processes.

3.3 Partitioning visualization

Using ParaView, generate the following images from an execution with 9 processes and corresponding strategies and input files:

- pent.CGUP.classic.rank5.jpg
- pent.CGUP.classic.rank8.jpg
- pent.CGUP.dual.rank5.jpg
- pent.CGUP.dual.rank8.jpg
- drall.SU.dual.rank3.jpg
- drall.SU.nodal.rank3.jpg

4 Submission

1. the code should run as:
   `mpiexec -n <NP> ./gccg <input_file> <partition_alg> <read_alg>`
   - `partition_alg` is one of: `classic`, `dual` or `nodal`.
   - `read_alg` is one of: `oneread` or `allread`.
2. Commit to your git repository the following files and folder structure:

- folder `A2.1/code/`: your *.c, *.h and Makefile
- folder `A2.1/scripts/`: your job files and other scripts
- folder `A2.1/data/`: the measurement results (given format in implementation)
- folder `A2.1/plots/`: image files from ParaView

Note: Files `gccg.c` and `test_functions.c` should not be modified. Any custom code added here will be overwritten in the automatic tests.
Also make sure to update the `Makefile` with any custom files you might have created. They should be added to the static library `libpos.a` too.