Numerical Solution of PDEs Using the Finite Element Method

Exercise 6: Shared memory parallelisation

Jean-Paul Pelteret (jean-paul.pelteret@fau.de)
Luca Heltai (luca.heltai@sissa.it)

20 March 2018

Some useful resources

`https://www.dealii.org/8.5.1/doxygen/deal.II/step_6.html`
`https://dealii.org/8.5.1/doxygen/deal.II/group__threads.html`
`https://dealii.org/8.5.1/doxygen/deal.II/classThreads_1_1TaskGroup.html`
`https://www.dealii.org/8.5.1/doxygen/deal.II/namespaceWorkStream.html`

---

1. Using `step-6` or the outcome of the previous exercise as a base:
   (a) Parallelise the following parts of your code using `TaskGroup` class:
      i. The system setup function. Can you parallelise the two calls the fill the `constraints`?
      ii. The assembly loop.
      iii. The (manual) calculation of the solution $\mathcal{L}^2$ norm.
   (b) Parallelise the following parts of your code using `TBB` via the `WorkStream` class:
      i. The assembly loop.
      ii. The (manual) calculation of the solution $\mathcal{L}^2$ norm.
   (c) Investigate the possible speed-up by playing around with the number of threads set in the call to `Utilities::MPI::MPI_InitFinalize`.
   Tip: This class needs to be created in the `main` file before you create and execute your problem class.
2. Additional tasks
   (a) What influence do the `queue_length` and `chunk_size` have on the efficiency of the various parallel operations that you have implemented?
   (b) Perform some timings and compare the results:
      i. The serial version of this code.
      ii. The TBB threaded version of the code, but enforcing the use of one thread via the call to `Utilities::MPI::MPI_InitFinalize`.
      iii. The TBB threaded version of the code using the maximum number of threads.