Modèles et techniques en programmation parallèle hybride et multi-cœurs: Travail pratique 2

Francesco Clerici

The exercise consists in the parallelization through OpenMP of an algorithm to solve a PDE in a 3D domain, already parallelized with MPI.

Fine-grain approach

In the fine-grain approach only little more effort is required with respect to the plain OpenMP parallelization. In fact, we are going to parallelize the loops contained `values.cxx` and `scheme.cxx`, with a special attention to the quantity `du_sum`. Note that when we initialize MPI, we are going to call `MPI_Init_thread(&argc,&argv, MPI_THREAD_FUNNELED, &provided );`. In this case we allow the `MPI_THREAD_FUNNELED` MPI access level, as we expect to parallelize through OpenMP only small portions of code, and hence there is no need for other processes to communicate.

Coarse-grain approach

Within the coarse-grain approach we write directly a parallel block inside the main function. Then, we take advantage of the functions providing local partitions of each sub-domain to focus on a number of `nthreads` slices of the sub-domain. No matter how we choose the dimension on which we cut the sub-domain. We compute `imin_loc` and `imax_loc` of each thread, likewise the plain OpenMP parallelization, as

```
int imin_th, imax_th;

#if defined(_OPENMP)
int id_th = omp_get_thread_num();
#else
int id_th = 0;
#endif

imin_th = m_P.imin_thread(0, id_th); // divide the MPI-domain in
imax_th = m_P.imax_thread(0, id_th); // nthreads-slices of dimension
                                     // (dx/nthreads, dy, dz)
```

Within the coarse-grain approach we we allow the `MPI_THREAD_SERIALIZED` access level, so that inside `scheme.cxx`, all the threads of each MPI process add safely their contribution to `du_sum` by calling in a serial way the function `MPI_Allreduce(&du_sum_local, &du_sum, 1, MPI_DOUBLE, MPI_SUM, m_P.comm());`