Task-based parallelization and dynamic load balance of finite element assembly

G. Houzeaux[*], M. Garcia-Gasulla[*], R. Ferrer[*], J. Labarta[*], and M. Vázquez[*,†]

[*] Barcelona Supercomputing Center,
Nexus II Building c/Jordi Girona 29, 08034 Barcelona, Spain
e-mail: guillaume.houzeaux@bsc.es, web page: http://www.bsc.es

[†] IIIA-CSIC, 08193 Bellaterra, Spain

ABSTRACT

Depending on the physics under consideration, the matrix assembly of a finite element code can result a costly task. This work presents a hybrid MPI+X parallelization of a finite element code, where X consists of a task-based programming model together with dynamic load balance.

Shared memory parallelization has been traditionally based on loop parallelization, mainly using OpenMP pragmas. In the finite element context, there exists a race condition as elements of different threads can access the same memory positions during the assembly. To handle mutual exclusion in order to avoid the race condition, OpenMP offers the `ATOMIC` or `CRITICAL` directives. These mechanisms are very practical from the programer point of view, but they can have a large overhead. Several method have been proposed (e.g. coloring, partitioning) to avoid this overhead, but they all suffer some drawbacks.

We propose an alternative to overcome all these drawbacks, using the OmpSs programming model. OmpSs is based on the same syntax as OpenMP and in fact includes most of the existing OpenMP 4.1 features. OmpSs is also one of the forerunners for OpenMP standard. The two original features considered in this work are *multidependencies* between tasks and *commutative* tasks [3]. Firstly, subdomains are defined as disjoint sets of elements. Secondly, an adjacency graph is created to express the multidependencies between the subdomains based on common nodes. The element loop is then splitted into two loops. A loop over the subdomains and then a loop over the elements of these subdomains. Thus, each task consists of a loop over the elements of a subdomain. In addition to this task parallelism, a dynamic load balance is used at runtime using the DLB library to share the resources between the different MPI tasks of the supercomputer nodes.

As illustrative example, we consider the assembly of the Navier-Stokes equations [1, 2]. In particular, we will show: firstly, that the spatial locality is much better than with a coloring technique; secondly, the gain in IPC by avoiding the use of `ATOMIC` directive; thridly, that the dynamic load balance is a very efficient technique to accelerate even further the element assembly. These three characteristics contribute conjointly to the speed up of the assembly.

REFERENCES

[1] Houzeaux, G. and Príncipe, J. A Variational Subgrid Scale Model for Transient Incompressible Flows, *Int. J. CFD*, Vol. **22**, pp. 135-152, (2008).

[2] Houzeaux, G., Aubry R. and Vázquez, M. Extension of fractional step techniques for incompressible flows: The preconditioned Orthomin(1) for the pressure Schur complement, *Computers & Fluids*, Vol. **44**, pp. 297-313, (2011).

[3] Vidal, R., Casas, M., Moretó M, Chasapis, D., Ferrer R., Martorell, X., Ayguadé, E., Labarta, J., Valero, M. OpenMP: Heterogenous Execution and Data Movements, Chapter *Evaluating the Impact of OpenMP 4.0 Extensions on Relevant Parallel Workloads*, Vol. **9342** of the series Lecture Notes in Computer Science, pp. 60-72, (2015).