Experiments with multi-level parallelism runtimes on a CFD code with unstructured meshes

Yvan Fournier, Emmanuel Jeannot, Benjamin Lorendeau

Experiments with multi-level parallelism runtimes on a CFD code with unstructured meshes

Yvan Fournier, *EDF R&D (MFEE)*
Emmanuel Jeannot, *INRIA (TADAAM)*
Benjamin Lorendeau, *EDF R&D & INRIA*

Abstract

Large applications for parallel computers and more specifically unstructured Computational Fluid Dynamics codes are often based on the bulk synchronous parallelism approach (BSP model) and therefore mostly exploit parallelism using runtime systems like the Message Passing Interface (MPI) which has enabled strong, relatively portable and quite durable performances for these codes for many years. However, as MPI was developed for distributed computing, we do not expect its standalone use to be the best fit for recent many-core architectures.

Indeed, the ever growing performance of high performance machines we are witnessing comes mostly through the aggregation of different computing devices in order to build heterogeneous machines. Those are the combination of traditional computing units (CPUs) with accelerators, namely many-core architectures such as GPGPUs or Intel Xeon-Phi accelerators. This hybridization of HPC clusters requires current scientific code developers to master more and more techniques and programming models in order to harness the quintessence of their machines.

As the post-petascale era has long been foreseen, runtime systems developers have been investigating other parallelism paradigms. Notably, the task based approach has gained a lot of popularity recently as it is expected to deliver portability and performance with a relatively simple programming model. Tasks can be both local or distant, so a single model can handle both inter-node and intra-node aspects. In addition, computation-communication overlap is straightforward. Opposed to the Coarse Grain Parallelism, performance depends strongly on choosing a good data grain size for each task, which should require performance measuring and tuning but no additional programming effort.

As *Code_Saturne* [1], our CFD code at EDF R&D is based on unstructured meshes, with a significant part of its code being memory-bound, refined parallelism through the use of MPI + X solutions such as MPI + OpenMP often fails to deliver significant (or any) performance improvements, though it does reduce the memory consumption per thread. Using a simple "loop-local" OpenMP model, as we increase the number of threads per MPI rank, performance drops rapidly, since many secondary loops are not threaded; and avoiding data races often requires specific renumbering strategies, which may not be easily adapted everywhere with a reasonable programming effort. These diminishing returns tend to limit the efforts which are worthwhile to spend in addition to the base MPI model.

We may see HPC current technologies evolution as unfavorable to an unstructured CFD code like *Code_Saturne* in its current form. This is why we decided to investigate recent HPC techniques and runtime systems for a sustainable, easy to propagate, portable and efficient solution to bring better performance and adaptability to *Code_Saturne*.

As many teams are already dedicating their work to propose new solvers and solve dense linear algebra, we decided to focus on another part of the puzzle, namely our gradient reconstruction computation. As a significant portion of our main current numerical schemes, it has a high impact over the performance of our code and an intermediate computational intensity. As such, we propose in this article a review of different implementations of our gradient computation towards the implementation of a task based approach through the use of task-based HPC runtime systems, and more specifically the PaRSEC [2] framework. The Parallel Runtime Scheduling and Execution Control (PaRSEC) framework implements a task-based dataflow-driven programming model aimed at offering high performance while relieving developers of supercomputers' hardware complexity.

We show that our first implementation offers comparable performance while increasing the arithmetic intensity of its computation (see figure 1). Moreover, by removing some data dependencies, our cell based approach paves the way for a more refined grain parallelism approach. We then push this approach to our prime objective – task based approaches – and implement our gradient computation using the PaRSEC runtime. Finally, we propose some insights on the use of

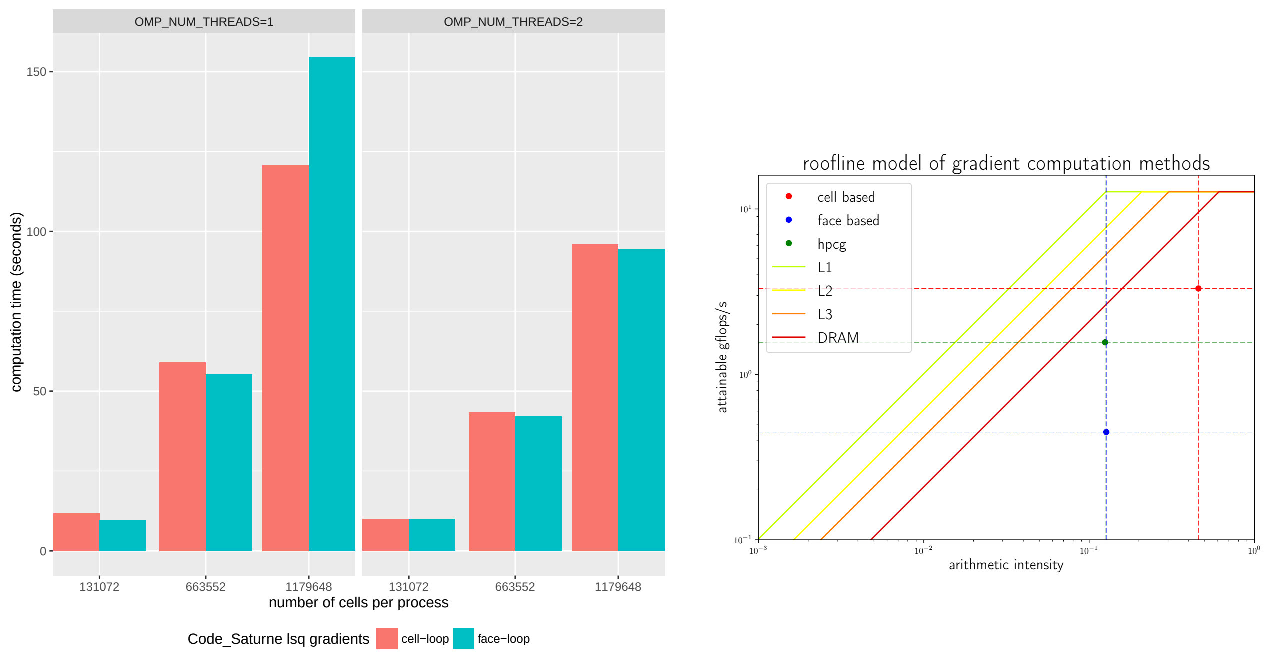

Figure 1 – Computation time comparison between face based and cell based lsq gradient (lower is better). We observe roughly same performances while arithmetic intensity of cell based approach is more than the double of the face based approach.

such runtime systems over more common HPC techniques based on their ease of their performance scalability, ease of incremental deployment in a legacy CFD code, and maintainability.

References

[1] F. Archambeau, N. Méchitoua, and M. Sakiz. Code saturne: A finite volume code for the computation of turbulent incompressible flows-industrial applications. *International Journal on Finite Volumes*, 2004.

[2] G. Bosilca, A. Bouteiller, A. Danalis, T. Herault, P. Lemarinier, and J. Dongarra. Dague: A generic distributed {DAG} engine for high performance computing. *Parallel Computing*, 38(1–2):37 – 51, 2012. Extensions for Next-Generation Parallel Programming Models.