TASK-BASED PARALLELIZATION OF A CFD CODE OVER A RUNTIME SYSTEM

JEAN-MARIE COUTEYEN[†*], JEAN ROMAN[*] AND PIERRE BRENNER[†]

[†] Airbus Defence and Space
51-61 Route de Verneuil, 78130 Les Mureaux, France

[*]Inria Bordeaux - Sud-Ouest
200 avenue de la Vieille Tour, 33405 Talence, France

Key words: CFD, Parallelization over Runtime, Task-based parallelism

Abstract. Task-based parallelization of an industrial aerodynamic solver is presented.

1 INTRODUCTION

FLUSEPA is an advanced simulation tool which performs a large panel of aerodynamic studies. It is the unstructured finite-volume solver developed by Airbus Defence & Space company to calculate compressible, multidimensional, unsteady, viscous and reactive flows around bodies in relative motion [2](Figure 1). The numerical strategy in FLUSEPA is designed for highly compressible flows and to remain accurate when using non-Cartesian grids. The meshing strategy is based on multi-overlapping grid intersection which is conservative and allows to quickly mesh 3D complex geometries. The time integration in FLUSEPA is done using an explicit temporal adaptive method [3]. Instead of using a time step imposed by the slowest cell, cells are grouped inside temporal classes according to their maximum allowed time step. This method allows to perform a fewer number of operations at the expense of some complexity. An iteration of the aerodynamic solver is separated into several sub-iterations. Every cell is not integrated during each sub-iteration but cells which share a face with a slower one are interpolated. At the end of the iteration, each cell has reached the same time. Between iterations, temporal classes can evolve.

The current version of FLUSEPA is parallelized using a classical MPI-OpenMP approach and domain decomposition: border faces are duplicated and ghost cells are used. However, the way the time is integrated leads to important time wasted in synchronization despite computation-communication overlapping. The time integration implies an order for the cells to be processed depending on their temporal class: neighbor cells must be at the same time during processing. This locality information is partially lost with the current parallelization. In this paper, we present a way to overcome this issue by working on sub-domains inside each process. We capture the dependencies more precisely through the use of a task-based parallel expression with the help of a well suited runtime system.

2 TASK-BASED PARALLELIZATION OVER RUNTIME

A runtime system is a software component that aims to hide the complexity of the architectures (including heterogeneous architectures with accelerators) to the programmer. A Directed Acyclic Graph (DAG) for which nodes are tasks and edges are dependencies is usually used. The tasks are then scheduled on the computational units.

In StarPU[1], one way to build a DAG (Figure 2) is to insert tasks sequentially: each task is inserted with information about the data it accesses and writes and those informations are used to build the dependencies. It is also possible to insert communications. Task generation functions are designed according to the data access patterns of the code: computations on faces, computations on cells, computations on faces using cells... In order to respect this new formalism, a refactoring step has been performed. The expected benefits on large multicore clusters rely on the capability to exploit asynchronism and indeterminism of execution. A first shared memory version of the task-based version has been developed and leads to promising performances with regards to the previous version. In this first step and in order to limit the overhead induced by the use of the runtime system (mainly for task insertion), "parallel tasks" have been introduced leading to an hybrid version based on a StarPU-OpenMP programming paradigm. This also allows to use more efficiently the available computational power by using all the cores while few tasks are left. The second step concerns a distributed version achieved by using the domain decomposition and by adding MPI exchanges between the subdomains. This step is currently in progress. The talk will present this programming approach and the benefits on the FLUSEPA parallel implementation.

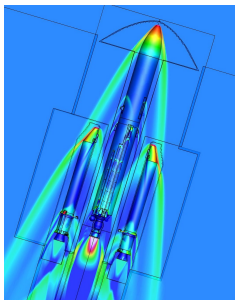

Figure 1: Stage separation.

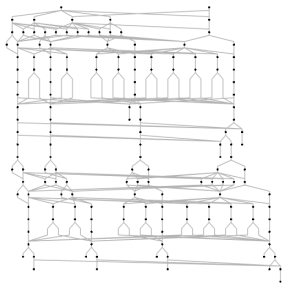

Figure 2: DAG : Nodes are tasks and edges are dependencies between them.

References

[1] Cédric Augonnet et al. "StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures". In: *Concurrency and Computation: Practice and Experience, Special Issue: Euro-Par 2009* (2011), pp. 187–198.

[2] P. Brenner et al. "Simulation d'interactions aérodynamiques instationnaires autour de plusieurs corps en mouvement relatif". In: *AAAF : Les interactions en aérodynamique.* Marseille, 1998.

[3] W. L. Kleb et al. "Temporal adaptive Euler/Navier-Stokes algorithm involving unstructured dynamic meshes". In: *AIAA Journal* Vol. 30 (1992), pp. 1980–1985.