FLUSEPA - a Navier-Stokes Solver for Unsteady Problems with Bodies in Relative Motion : Toward a Task-Based Parallel Version over a Runtime System

Jm Couteyen Carpaye, Jean Roman, P Brenner

FLUSEPA - a Navier-Stokes Solver for Unsteady Problems with Bodies in Relative Motion : Toward a Task-Based Parallel Version over a Runtime System

JM. COUTEYEN CARPAYE[1,2], J. ROMAN[1] et P. BRENNER[2]
[1] Inria Bordeaux - Sud-Ouest
[2] AIRBUS DS, TSOEM536, Les Mureaux

About FLUSEPA

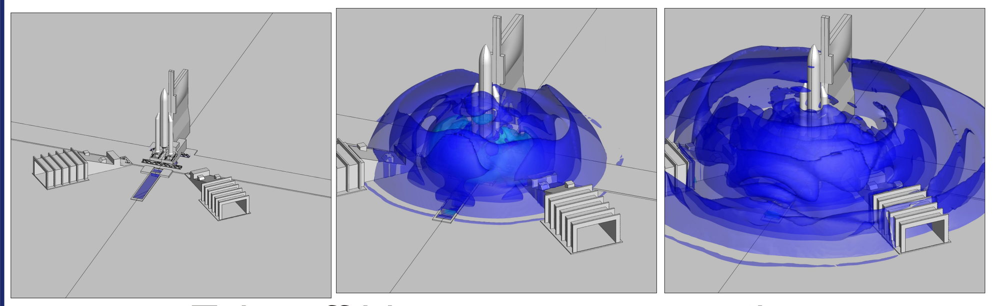
Cell-centered finite volume ❶❷
Unsteady and reactive flows ❶
Explicit temporal adaptive time integration ❶
Bodies in relative motion ❷
MPMD with specialized processes ❶❷❸
MPI/OpenMP parallelization ❸

About StarPU

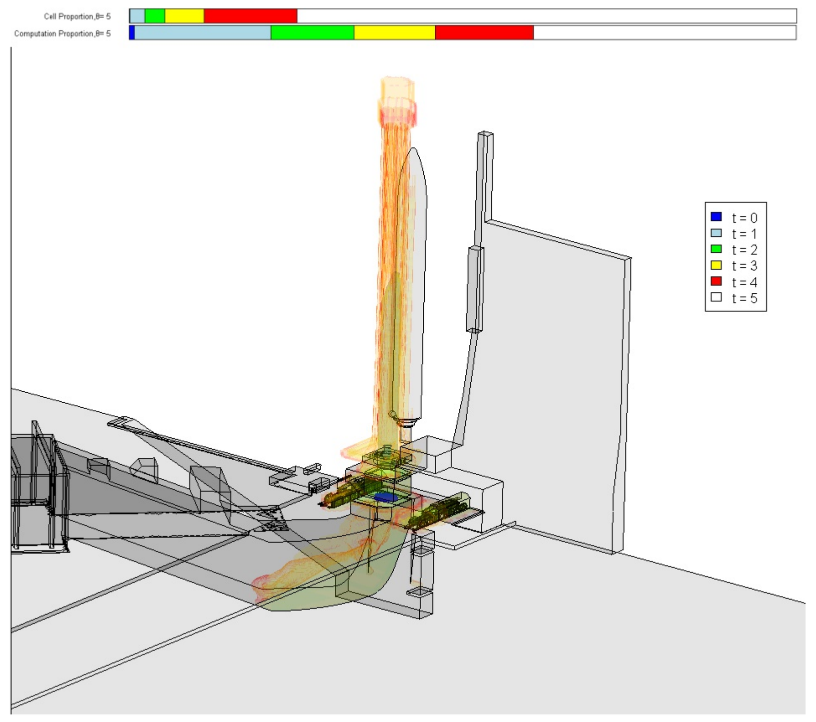
Runtime System ❹
Task scheduling ❹❺
Heterogeneous multicore architectures ❹
Unified view of ressources ❹
C Library, not a new langage ❺
Used for the new version of FLUSEPA ❻

1 Aerodynamic Solver

The aerodynamic solver of FLUSEPA is particularly suited for unsteady computations, even if they do not imply bodies in relative motion.



Take off blastwave computation.



With temporal adaptive, cells have a different computational cost. Small cells take more iterations than bigger ones to reach the same time.

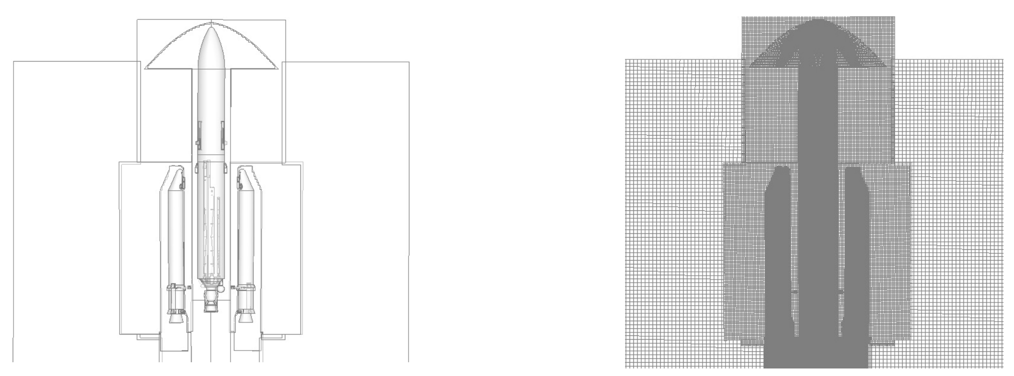
Computation needs to be done in a certain order to ensure consistency.

This leads to difficulties to parallelize efficiently the aerodynamic solver.

In the picture, the different colors represent different class of cells for a take-off blast-wave computation.

2 Motions and Intersections

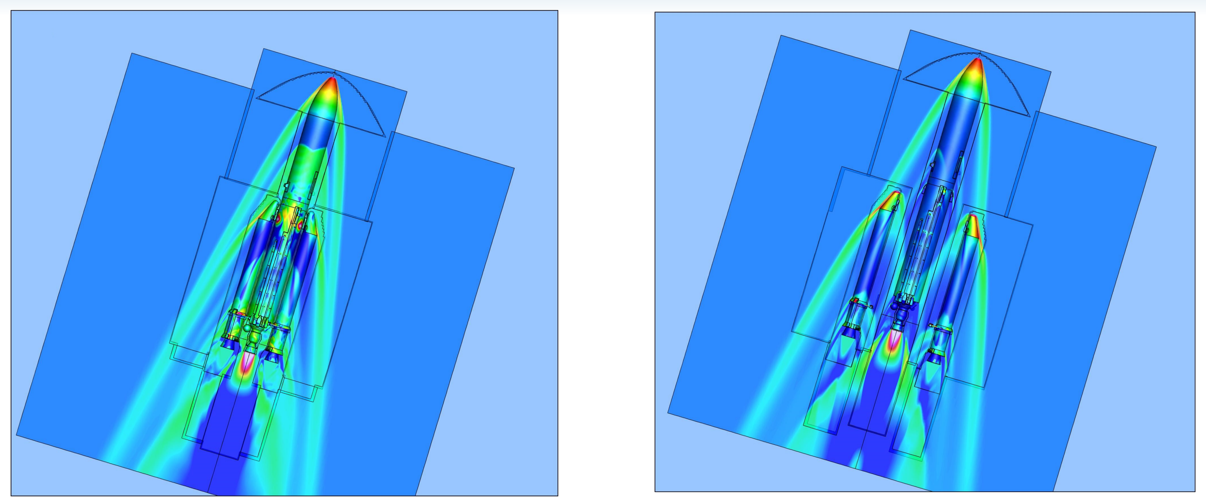
Multiple meshes around several bodies.



Boosters and the main stage are meshed independtly.

Load are gathered during aerodynamics computation then a 6DoF formulation is used to compute the relative motion.

When necessary, a new intersection is computed.

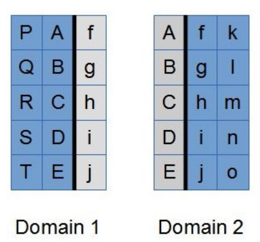


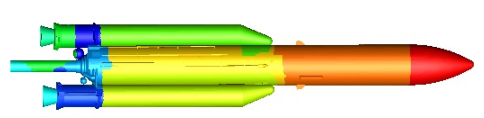
Computation of booster stage separation from distancing rocket ignition to their extinction.

3 Parallelization and Limitations

The parallelization of the aerodynamic solver relies on domain decompotisions and ghost cells.

Ghost cells (in gray) allow to communicate between different domains. Values of the neighbor domains are filled using communications.



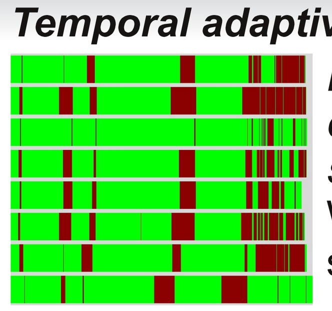


A domain decomposition is used. The first decomposition can be altered when temporal classes evolves.

Intersections can be computed asynchronously while aerodynamics is still computed.

An extrapolation of the kinematic is computed in order to compute intersections early.

Temporal adaptive synchronization issue



Each bar represents a domain, green is computation while red is time lost in synchronization.
With the current parallelization, lot of synchronization are implied.

Intersection / aerodynamic load balancing issue

- The number of processes dedicated to intersections or aerodynamics is defined at the beginning of the computation.
- The respective loads vary during the computation.

Interest of using a runtime

- Take advantage of a task description of the problem to exploit the actual dependencies of the aerodynamic solver.
- Co-schedule "Intersections" and "Aerodynamics" should lead to better use of computational ressources and less data transfers.

StarPU : a runtime system ❹

Rationale
- Implement the sequential task flow programming model
- Map computations on heterogeneous computing units

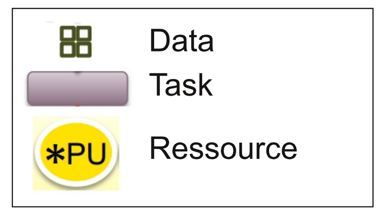
Programming Model
- Task
- Data
- Relationships
  - Task ↔ Task
  - Task ↔ Data

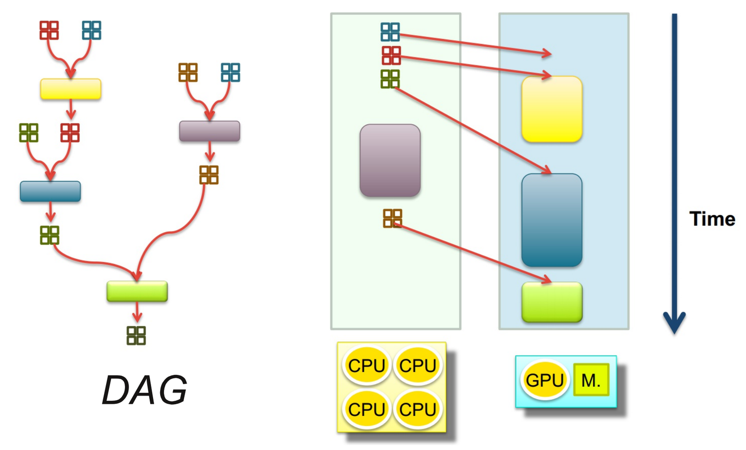
Runtime System
- Heterogeneous Task scheduling
- Application Programming Interface (Library)

C. Augonnet, S. Thibault, R. Namyst, and P.-A. Wacrenier. **StarPU: A Unified Platform for Task Scheduling on Heterogeneous Multicore Architectures.** *Concurrency and Computation: Practice and Experience, Special Issue: Euro-Par 2009, 23:187-198, February 2011.*

A Directed Acyclic Graph (DAG) is generated on the fly : submitting a task is a non-blocking operation.
Task are then scheduled around the computational units.

Mapping the DAG on the hardware :
- Allocating computing resources
- Enforcing dependency constraints
- Handling data

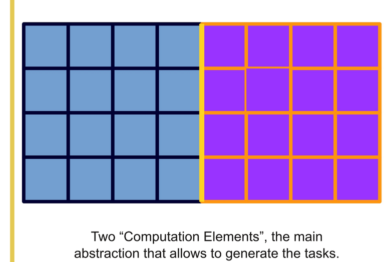




DAG

Task generation (aerodynamic solver) ❺

Use of a domain decomposition inside each node in order to generate tasks.



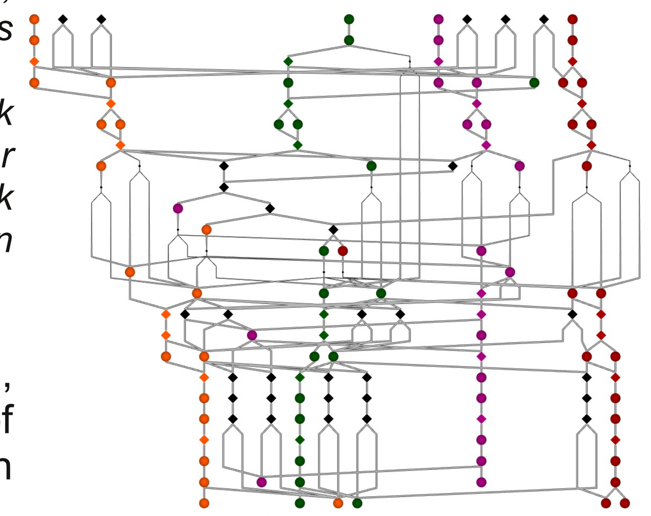
For each sub-domain, cells and faces are considered differently, and are represented by "handles" in StarPU.
Using those handles and generation functions, tasks and dependencies are generated.

Sub-domain 1 : Cells - Faces
Sub-domain 2 : Cells - Faces
Faces between sub-domain 1 and sub-domain 2

For the distributed version, communications are inserted just like tasks, and the access are consistent with the application.

With the generations functions, a DAG is generated. In the DAG to the right, the colors represents differents subdomains.
Circles represent tasks that work mainly on cells, while diamond are for thoses which work on faces. Black diamonds are for faces between subdomains.

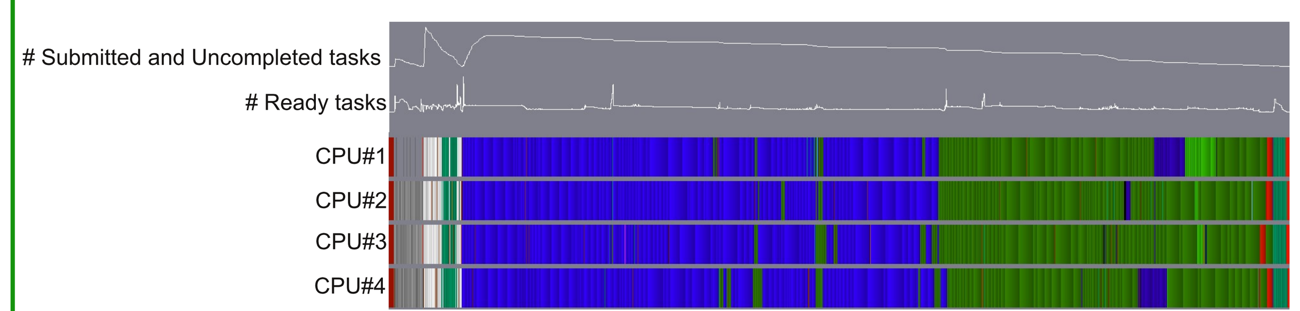
When waiting for a communication, work may be available because of the finer grain obtained by domain decomposition.
It is then possible to tune the scheduling in order to minimize computation time.



Extract of a DAG generated by FLUSEPA

Current results and perspectives ❻

Results in shared memory



Red is synchronization while other colors represents operation on differents temporal levels.

With the new task system, unstead of waiting on OpenMP-DO barriers, other ready tasks can be started.
Some low level operations can be started before the end of the computation, while this was not possible with the OpenMP version.

Perspectives for the Aerodynamic solver

-Distributed version
  - Validate the results
  - Incorporate a load balancing system

- Possible Improvment by the task description of the problem
  - Pipeline iterations of the solver. The method only implies a global communication for setting the time step, but this is manageable in a different way.

Perspectives for the whole application

- Rewrite Intersection with tasks.
- Co-scheduled Intersection and Aerodynamics