Evaluation of a parallel task-based approach to accelerate high-order CFD calculations on heterogeneous architectures.

Emeric Martin [(1)], Raphaël Blanchard [(2)], Florent Renac [(1)]

[(1)] ONERA / CFD department
[(2)] ONERA PhD Student - Inria teams STORM,TADaaM

Journée Runtime, Groupe Calcul, 20/01/17, Inria/Paris

ONERA
THE FRENCH AEROSPACE LAB

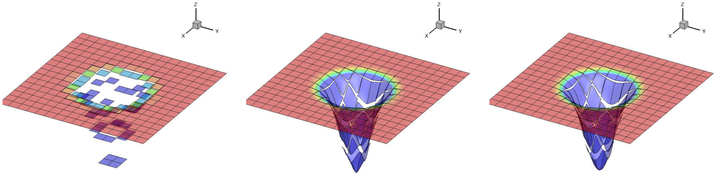
retour sur innovation

Outline

Outline

Context

- Scientific challenge
  - High accuracy and efficiency for compressible flows in aerodynamics
    - ⋆ Flow features: vorticity, turbulence, ...
    - ⋆ Quantities of interest: lift, drag, ...
- Investigations of the potential of high-order schemes
  - A promising candidate: the Discontinuous Galerkin (DG) method
    - ⋆ Look for a piecewise polynomial of degree p
    - ✓ Spatially high-order accurate numerical schemes
    - ✓ Compact stencil: well-suited to unstructured meshes, parallelism, etc.
    - ✗ Large memory requirements, high computational cost



(a) $p = 0$ (b) $p = 1$ (c) $p = 2$

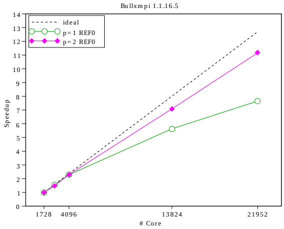
Aghora.

Context

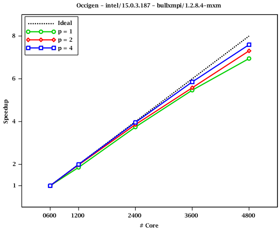
- Aghora, a research project at ONERA
  - Solution of complex turbulent flows with multilevel models
    - ⋆ RANS, URANS (high Reynolds, transonic)
    - ⋆ LES, VMS (moderate Reynolds, subsonic)
    - ⋆ DNS (low Reynolds, subsonic)
  - Contributions to european projects : IDIHOM,ANADE,TILDA
- Software prototype
  - Around 120,000 lines in Fortran 2003
  - Generic and modular programming
  - Data structures for unstructured meshes
    - ⋆ per edge and per element to ensure data locality
    - ⋆ mainly : structures of arrays
  - Intel ecosystem (compilers, profiling tools, MKL)

Context

- Programming models for parallel computing
  - Non-blocking and synchronous communications with large overlaps
  - Already tested : classic MPI, hybrid MPI/OpenMP



Taylor-Green Vortex (Curie)



Bump Delery (Occigen)

Sensitivity of the polynomial degree p on the strong scalability

Context

- But modern computing platforms are increasingly heterogeneous
  - Multi-core sockets
  - Many-core accelerators (GPUs, MICs)
  - Specialized cores
- Limitations of our previous parallel approaches
  - Which distribution (processes,threads) on which resource ?
  - Heterogeneity could come from many aspects
    - Geometry of each element (different metrics)
    - Local value of the polynomial degree (arithmetic intensity)
    - Co-treatement during calculation for certain elements
- How to fully exploit such complex architectures in this context ?
  - Evaluation of a task-based programming model ...

Outline

Overview of StarPU runtime

- Runtime on different supports: CPUs, GPUs, MICs
- A graph of tasks with data dependencies (Directed Acyclic Graph)
- Different scheduling policies to resolve the DAG
- Decision to assign tasks to supports is coming at the execution
- Portability of performance
- Task paradigm of StarPU can be combined with MPI

An iterative time integration scheme

```
WHILE(phys_time<Tmax) !Time Loop
   CALL sutherland_law
   DO kr = 1,krk  !RK Loop
      CALL bc_matching
      CALL compute_integral
      CALL rk_sub_step
   ENDDO
   CALL update_phys_time
ENDDO
```

- sutherland_law
  - Physical variables
- bc_matching
  - Prepare match conditions between domains
- compute_integral
  - Compute residuals
- rk_sub_step
  - Update DOFs at each Runge-Kutta steps

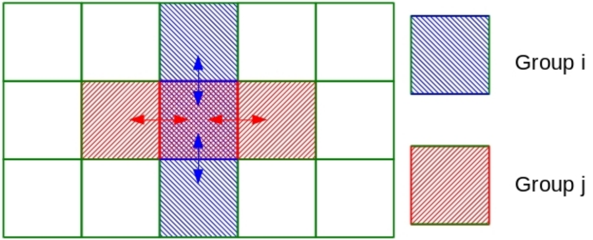
Adaptations of the algorithm to StarPU

```
WHILE(phys_time<Tmax) !Time Loop

   CALL sutherland_law

   DO kr = 1,krk !RK Loop

      CALL bc_matching

      IF(kr==1)THEN
         CALL compute_integral
      ENDIF

      CALL nowhere

      IF(kr/=krk)THEN
         CALL rk_integral
      ELSE
         CALL rk_sub_step
      ENDIF

   ENDDO

   CALL update_phys_time

ENDDO
```

- All functions are tasks
- nowhere:
  - New function
  - Empty task
  - Add dependencies between tasks
- rk_integral:
  - Execute rk_sub_step
  - kr = kr + 1
  - Execute compute_integral

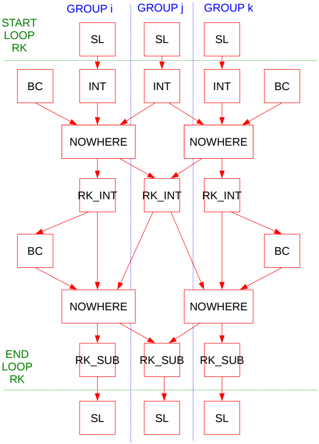
Implementation into Aghora solver

- Task scheduling implies an overhead: how to deal with that?
  - Arithmetic intensity (FLOP/Byte) must be enforced
  - Introduction of groups of elements: 3 elements with 2 faces
  - One group for one task



Basic example with a 2D Cartesian grid

DAG: Directed Acyclic Graph



- SL: sutherland_law
- BC: bc_matching
- INT: compute_integral
- RK_INT: rk_integral
- RK_SUB: rk_sub_step

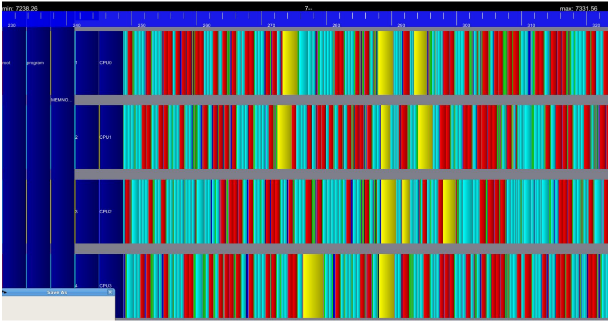
Outline

Numerical experiments

- Taylor Green Vortex test-case
  - 3D compressible Navier-Stokes equations ($M = 0.1$, $Re = 500$)
  - Explicit time discretization (SSP RK4)
  - Coarse mesh with 16^3 elements
  - Polynomial degrees: $2 \leq p \leq 6$
- Architectures:
  - One multi-core socket of a Haswell node
    - ⋆ Bi-socket E5-2690 v3 @2.60GHz, 64 GB by socket
  - Xeon Phi in native mode
    - ⋆ KNC: 61 cores @1.2Ghz, 4 threads per core, 16 GB
  - Intel compilers 16, IntelMPI 5.1.1.109

One socket: example of a tasks distribution

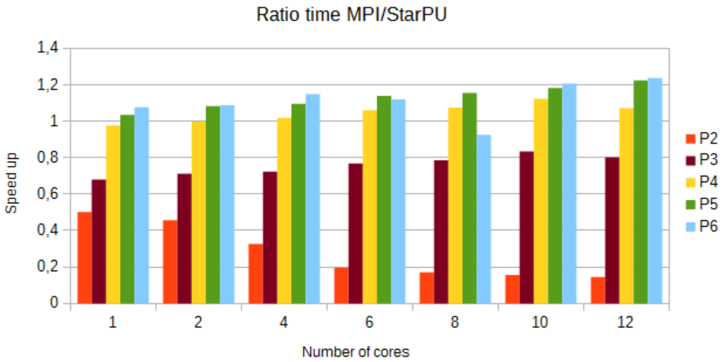
- The time iterative process is desynchronized
  - Not always the same kind of tasks at the same moment
  - Probably less stress on memory bus.



	State name	Color
1	BC_LIFT_BR2	Red
2	BC_MATCHING	Green
3	COMPUTE_MATCH_BASE	Blue
4	INTELT_MODAL_BASE	Yellow
5	RK_FINAL_3D	Magenta
6	RK_INTELT	Cyan
7	SUTHERLAND_LAW	Dark magenta

Tasks distribution over 4 cores during execution

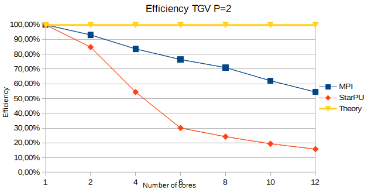
One socket: execution time

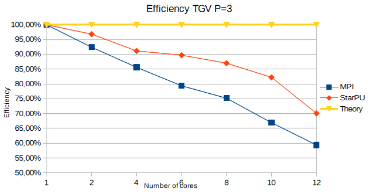


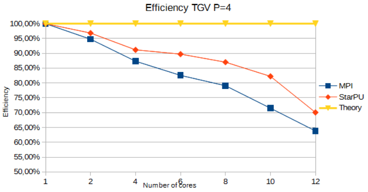
Polynomial degree p	$p=1$	$p=2$	$p=3$	$p=4$	$p=5$	$p=6$
Arithmetic intensity	4	17	44	87	150	236

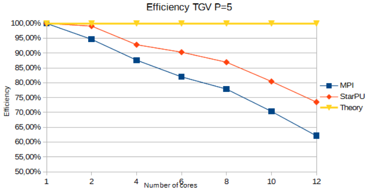
Arithmetic intensity of one task with group as a function of p

One socket: parallel efficiency









One socket: some remarks

- For Starpu approach
  - Schedulers prio and lws offer same performance
  - MPI master process is binded with Intel MPI environment variables
  - Intel MKL must not be able to generate dynamically its own threads
  - StarPU parameters to limit number of submitted tasks
    - Impact on memory consumption

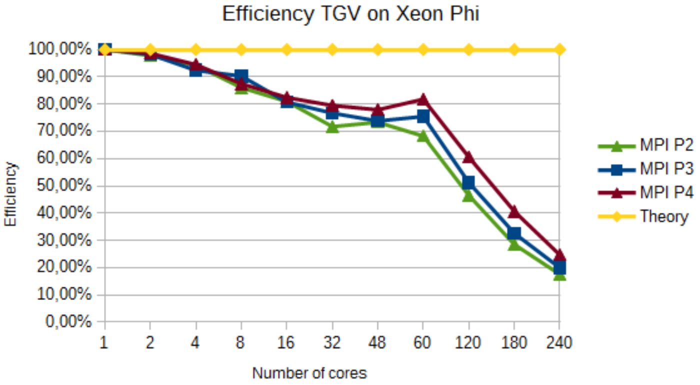
One socket: some remarks

- Four parallel approaches of Aghora solver have been developped
  - MPI reference version
  - MPI version with groups of elements
  - MPI/OpenMP version with OpenMP threads on groups
  - StarPU version
- Numerical comparison : around $1e^{-6}$
- A part of StarPU gain comes from introduction of groups

# cores	01	02	04	06	08	10	12
(MPI ref)/(StarPU)	1.11	1.12	1.22	1.17	1.18	1.21	1.17
(MPI grp)/(StarPU)	1.01	1.01	1.03	1.06	1.04	1.09	1.12

Sensitivity of the number of cores to StarPU gain for $p = 6$

One Xeon Phi: parallel efficiency



One Xeon Phi: some remarks

- For StarPU approach
  - Larger execution times for $p < 4$
  - Quite similar performance for $p = 4$ from 1 to 32 cores
  - Scheduler prio ; balanced affinity type for StarPU threads
  - Limitation of the memory available on Xeon Phi
    - ⋆ Use of meshes with a smaller number of elements to increase p

Outline

Concluding remarks

- In an homogeneous environment, our task-based approach offers:
  - Gains in CPU time for high-order degree ($p \geq 4$ on TGV test-case)
  - A competitive parallel behavior ($p \geq 3$)
  - A promising trade-off for accelerating calculations
    - ⋆ Break sequential iterative process
    - ⋆ Not always the same kind of tasks at the same moment
    - ⋆ Increase temporal locality
    - ⋆ Better flexibility with scheduling strategies during execution
  - On Intel Xeon Phi architecture
    - ⋆ Possibilities to obtain similar performance as MPI version

Concluding remarks

- Perspectives:
  - In an heterogeneous environment : 1 socket + 1 KNC
    - ⋆ Work in progress...
  - Evaluation of our parallel versions on KNL
  - Research axes to improve performance
    - ⋆ Increase arithmetic intensity and execution time of tasks
    - ⋆ Exploitation of data locality
    - ⋆ Different schedulers (possibility to define a new scheduler)
    - ⋆ Combination of StarPU threads and MPI processes
    - ⋆ Vectorization to be developped

Concluding remarks

- Feedback:
  - Interoperability of Fortran with C
    - For StarPU to manage Fortran pointers
  - Adaptations of the data structure to increase weight of tasks
    - With a constraint to preserve globally the original one
  - Get numerical validity with StarPU version
    - Not so easy to analyse and debug

We acknowledge the Bull Center for Excellence in Parallel Programming (CEPP) and the Méso-centre PlaFRIM for awarding us access to Phis resources

Thank you for your attention