The Multi-Stencil Language: orchestrating stencils with a mesh-agnostic DSL

Hélène Coullon, Julien Bigot, Christian Perez

INRIA team Avalon
Maison de la simulation (CEA)

5*th* JLESC Workshop - 28th June 2016, Lyon

Domain Specific Languages

+ Domain Specific Languages

Separation of concerns between the domain and its efficient implementation

- Easy language for the end-user
- Implicit optimizations
- Implicit parallelization

But none or only a few DSLs used in **production** !
ETP4HPC : programming models for non experts to reach exascale !

Domain Specific Languages

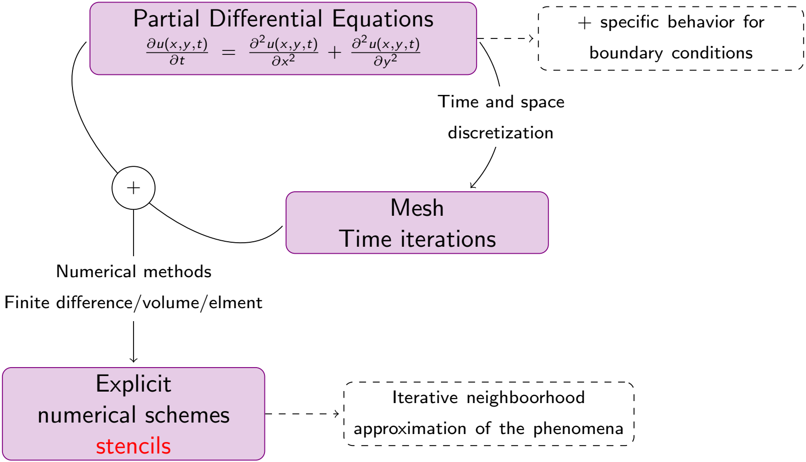
- Domain Specific Languages

- Choose a domain and a language more or less specific
- Difficult to combine DSLs (interoperability)
  - exascale applications = many specific domains and interactions
  - semantic / compilation / back-end codes
- Difficult to maintain

How to improve it ?

- Choose the good abstraction level to
  - stay efficient
  - be used in many domains (find common meta-domains)
- Choose good programming models for
  - the compiler programming
  - the back-end
- Try to reuse existing DSLs (compilation process or back-end codes)

Multi-Stencil application



Multi-Stencil Language : Yet another DSL for stencils !

Inputs

follow a descriptive grammar to simply declare

- the sequential order of computations
- what is read and written by each computation
- with or without a neighborhood access (stencil)

Outputs

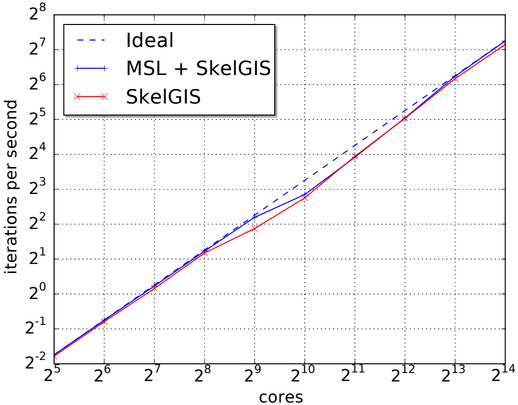
- parallel pattern of the multi-stencil application
- hybrid parallelism included : data and task parallelism
- choice of parallel languages (MPI, OpenMP 3.x/4.x, SkelGIS etc.)
- end-users have to fill computation functions into this pattern

Multi-Stencil Language : Yet another DSL for stencils !

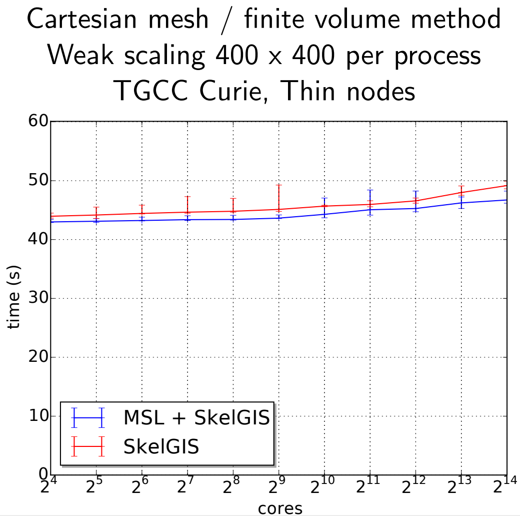
- Choosen abstraction level
  - mesh-agnostic description language
    - should be usable for unstructured, Cartesian, curvilinear meshes
  - description concerns splited from implementation concerns
    - usable with many different parallel libraries or languages back-ends
- Choosen programming models for back-end
  - Component programming model (maintainability, portability and composability)
  - Task scheduling model (portability and efficiency)
- Reuse of existing languages and DSLs
  - Use of a Distributed Data Structure DSeL (C++,MPI) : SkelGIS
  - MPI and OpenMP (3.x and 4.x)

Shallow-water equations

Cartesian mesh / finite volume method
Strong scaling 10k x 10k mesh with 1k iterations
TGCC Curie, Thin nodes



Shallow-water equations



Work in progress and collaborations

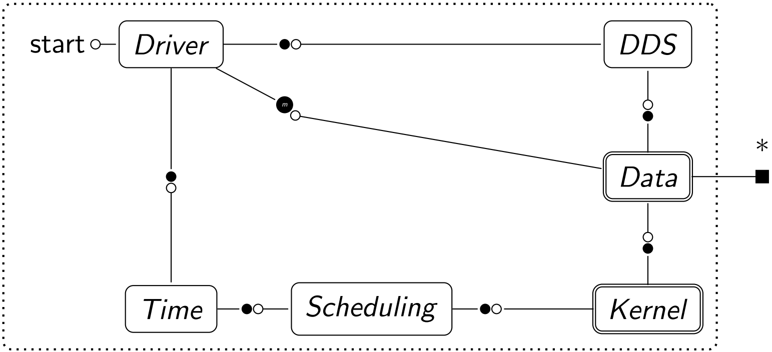
Work in progress

- Different types of scheduling and back-end in MSL
- Combine MSL implementation choices to an energy-aware framework
- New programming model back-end for MSL : combining component models and task models (Ph.D. Jérôme Richard)

Collaborations

- More application cases
- Back-end programming model : component models and task scheduling (Components in OmpSs ?)
- DSLs interoperability

MSL to Component-based runtime



Duplicated on each processor/core