FROM RESEARCH TO INDUSTRY

cea

Fostering Metamodels and Grammars Within a Dedicated Environment for HPC

The NabLab Environment (tool)

Benoît LELANDAIS - Marie-Pierre OUDOT

CEA, DAM, DIF, F-91297 Arpajon, France

Benoit COMBEMALE

Univ. Toulouse & Inria, France

SLE 2018 - November 6th, 2018

Preprint: https://hal.inria.fr/hal-01910139

UNIVERSITÉ TOULOUSE Jean Jaurès

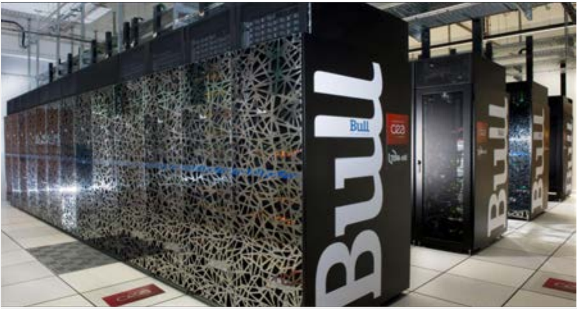
Inria inventors for the digital world

www.cea.fr

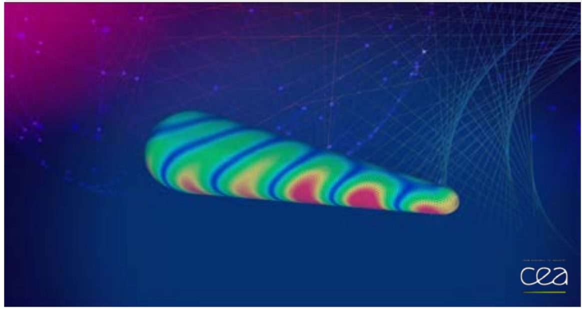
Context

CEA is the French Alternative Energies and Atomic Energy Commission

CEA is a major player in High Performance Computing (HPC) through the ***Simulation Programme***
CEA simulates hyperbolic systems and gas dynamics for transport and diffusion equations



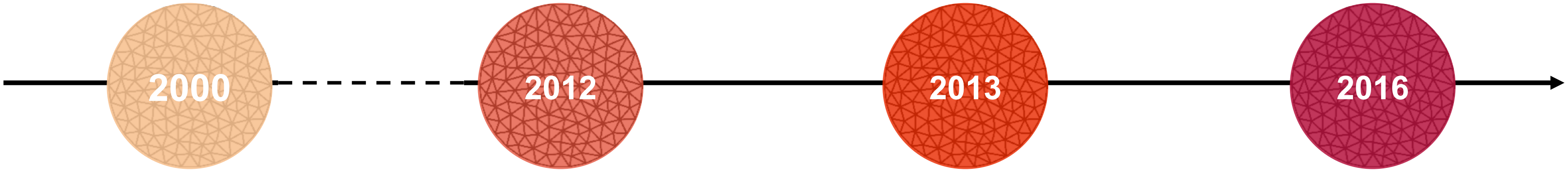
Simulation covers **wide physics phenomena**. It takes more than 10 years for a simulator to go into production



CEA co-designs with Atos future generations of Bull calculators and deploys **new architectures** and models for programming every 5 years

We need to abstract business knowledge from technical knowledge.
It is vital for our applications given the rapid evolution of hardware

Towards Higher Abstraction



2000

2012

2013

2016

Arcane framework

C++ framework[1] for the simulation of 2D/3D parallel codes

Mesh, variables, memory, and parallelism management

Modane environment

EMF-Based modeling environment that deals with the architectural aspect of simulation codes

Provides a user-friendly environment for designing numerical simulators

nabla language

Flex/Bison DSL dedicated to numerical-analysis

http://nabla-lang.org

Translates numerical-analysis sources and generates optimized code for different targets and architecture

NabLab environment

EMF-Based language and development environment inspired from Nabla

Provides a user-friendly environment for mathematicians

[1] Co-developed with IFPEN

Why DSLs are good candidates?

DSL is a good candidate for capturing and perpetuating **domain knowledge**

DSL is a good candidate for improving **software quality and performance**

DSL is a good candidate for supporting **brainstorming**

DSL is a good candidate for **communication and training**

…and all along the lifecycle of simulators thanks to code generation

We have defined a **DSL dedicated to HPC** and built a custom design environment based on this DSL for our end-users

Outline

01

Context

Context - Towards greater abstraction - Why is DSL a good candidate ?

02

Description of developments

What is ∇ ? - What is NabLab ? - NabLab Compilation Chain - Demo

03

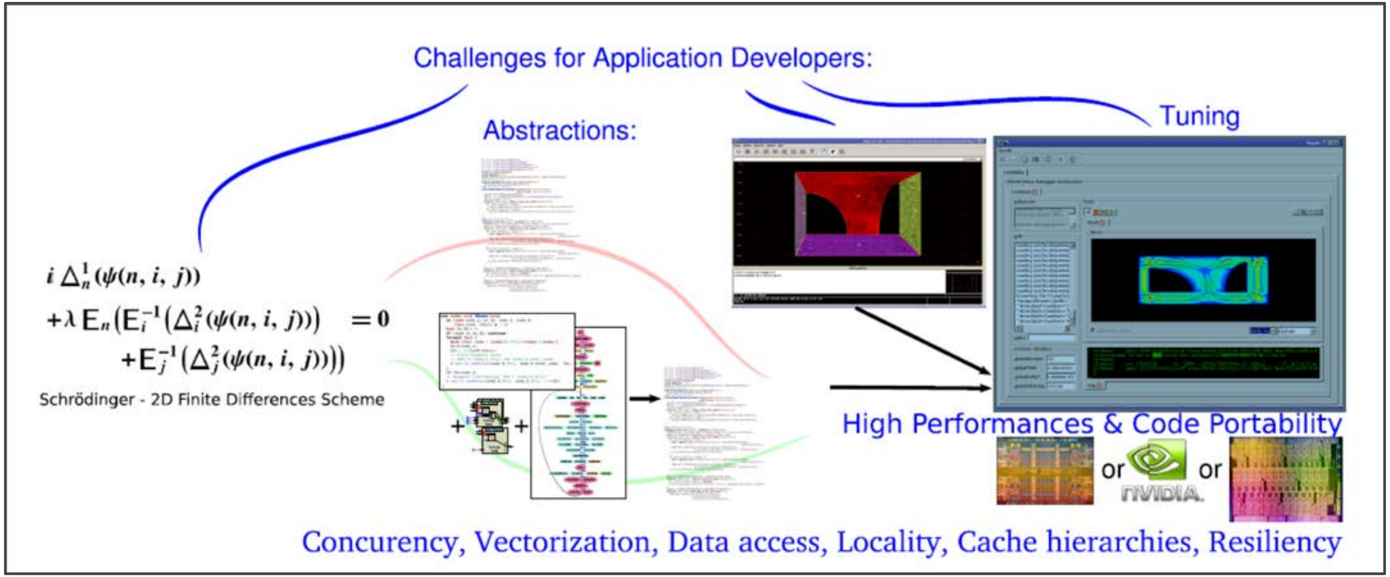
Conclusion & Perspectives

Grammar based DSL vs Metamodel based DSL - Towards a new Generation Strategy - Perspectives

What is ∇?

Numerical Analysis Specific Language - http://nabla-lang.org

nabla is an open source Domain Specific Language (DSL) for numerical analysis algorithms.
The nabla compiler can generate optimized source code for various hardware and software architectures



Challenges for Application Developers:

Abstractions:

Tuning

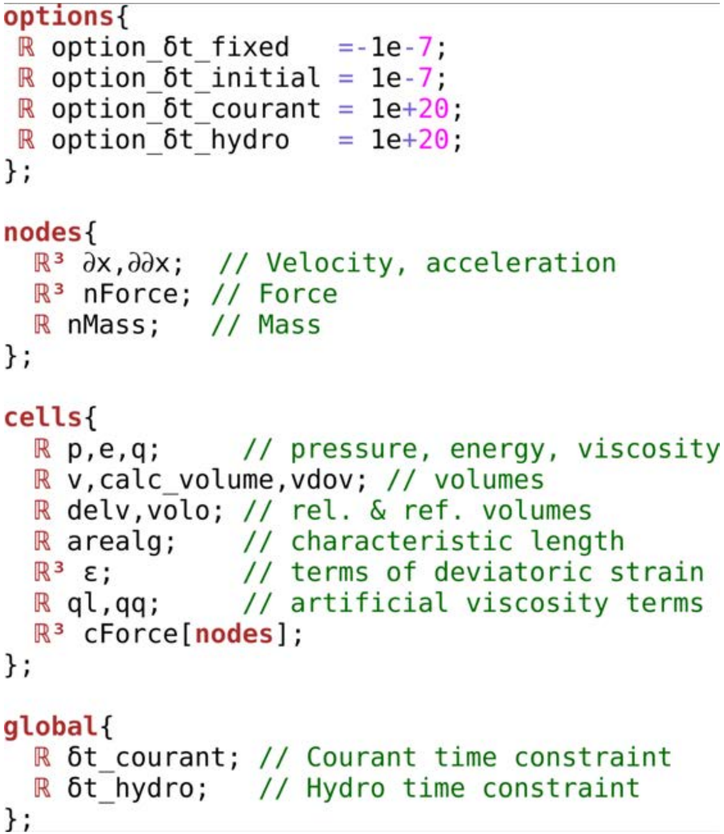
$$i\,\Delta_n^1(\psi(n,i,j)) + \lambda\,E_n\left(E_i^{-1}\left(\Delta_i^2(\psi(n,i,j))\right) + E_j^{-1}\left(\Delta_j^2(\psi(n,i,j))\right)\right) = 0$$

Schrödinger - 2D Finite Differences Scheme

High Performances & Code Portability

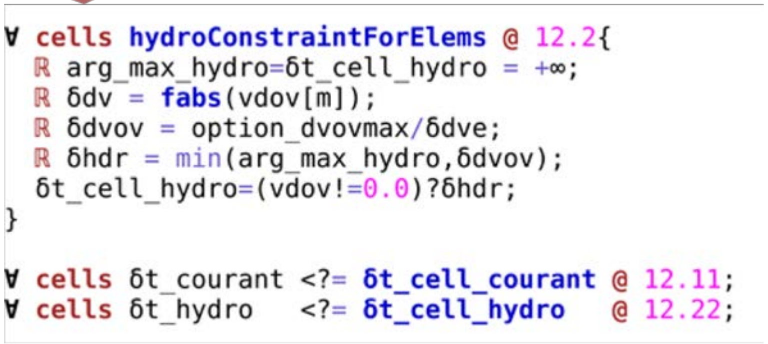
Concurency, Vectorization, Data access, Locality, Cache hierarchies, Resiliency

∇'s syntax



```
options{
 ℝ option_δt_fixed    =-1e-7;
 ℝ option_δt_initial  = 1e-7;
 ℝ option_δt_courant  = 1e+20;
 ℝ option_δt_hydro    = 1e+20;
};

nodes{
  ℝ³ ∂x,∂∂x;  // Velocity, acceleration
  ℝ³ nForce; // Force
  ℝ nMass;   // Mass
};

cells{
  ℝ p,e,q;      // pressure, energy, viscosity
  ℝ v,calc_volume,vdov; // volumes
  ℝ delv,volo; // rel. & ref. volumes
  ℝ arealg;     // characteristic length
  ℝ³ ε;         // terms of deviatoric strain
  ℝ ql,qq;      // artificial viscosity terms
  ℝ³ cForce[nodes];
};

global{
  ℝ δt_courant; // Courant time constraint
  ℝ δt_hydro;   // Hydro time constraint
};
```

Data-parallelism is implicitly expressed via jobs items



```
∀ cells hydroConstraintForElems @ 12.2{
  ℝ arg_max_hydro=δt_cell_hydro = +∞;
  ℝ δdv = fabs(vdov[m]);
  ℝ δdvov = option_dvovmax/δdve;
  ℝ δhdr = min(arg_max_hydro,δdvov);
  δt_cell_hydro=(vdov!=0.0)?δhdr;
}

∀ cells δt_courant <?= δt_cell_courant @ 12.11;
∀ cells δt_hydro   <?= δt_cell_hydro   @ 12.22;
```

Jobs parallelism is explicitly declared via Hierarchical Logical Time (HLT)

What is NabLab ?

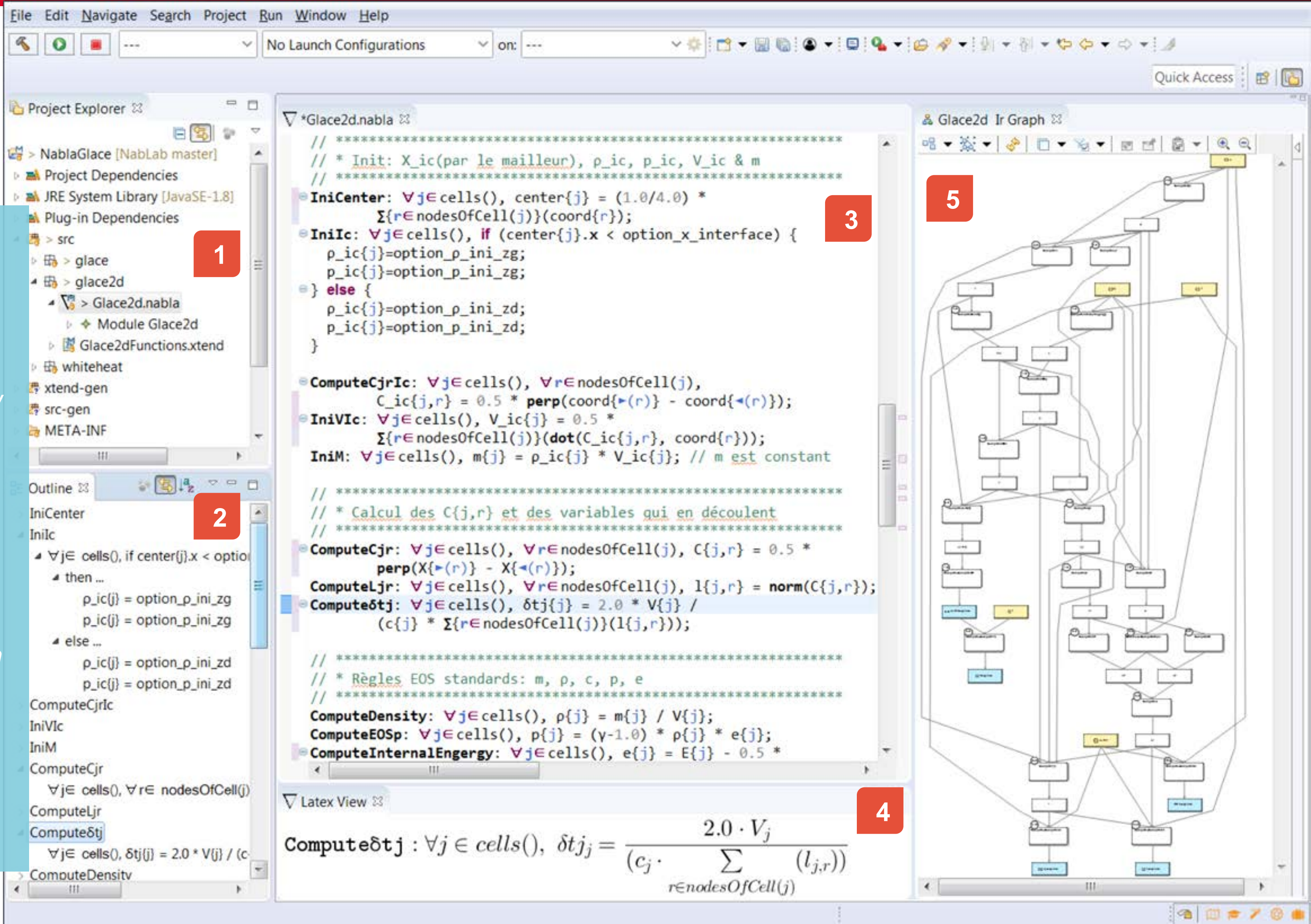
1 **Model Explorer**
to browse the model and semantic concepts

2 **Outline**
to give a clear overview of the model

3 **Textual Editor**
to enjoy all the benefits of an advanced editor

4 **Latex View**
to get scientific notation for your equations

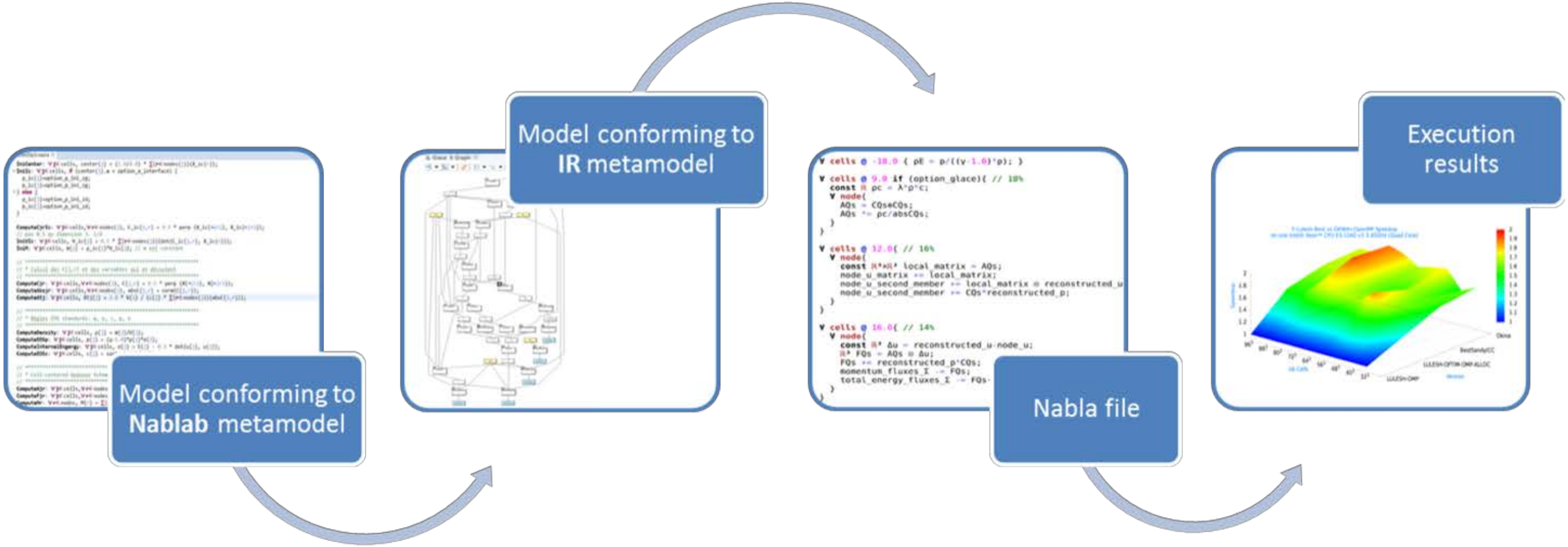
5 **Data-flow Graph**
to identify the order of execution of jobs



```
// ******************************************************
// * Init: X_ic(par le mailleur), ρ_ic, p_ic, V_ic & m
// ******************************************************
IniCenter: ∀j∈cells(), center{j} = (1.0/4.0) *
        Σ{r∈nodesOfCell(j)}(coord{r});
IniIc: ∀j∈cells(), if (center{j}.x < option_x_interface) {
  ρ_ic{j}=option_ρ_ini_zg;
  p_ic{j}=option_p_ini_zg;
} else {
  ρ_ic{j}=option_ρ_ini_zd;
  p_ic{j}=option_p_ini_zd;
}

ComputeCjrIc: ∀j∈cells(), ∀r∈nodesOfCell(j),
        C_ic{j,r} = 0.5 * perp(coord{►(r)} - coord{◄(r)});
IniVIc: ∀j∈cells(), V_ic{j} = 0.5 *
        Σ{r∈nodesOfCell(j)}(dot(C_ic{j,r}, coord{r}));
IniM: ∀j∈cells(), m{j} = ρ_ic{j} * V_ic{j}; // m est constant

// ******************************************************
// * Calcul des C{j,r} et des variables qui en découlent
// ******************************************************
ComputeCjr: ∀j∈cells(), ∀r∈nodesOfCell(j), C{j,r} = 0.5 *
        perp(X{►(r)} - X{◄(r)});
ComputeLjr: ∀j∈cells(), ∀r∈nodesOfCell(j), l{j,r} = norm(C{j,r});
Computeδtj: ∀j∈cells(), δtj{j} = 2.0 * V{j} /
        (c{j} * Σ{r∈nodesOfCell(j)}(l{j,r}));

// ******************************************************
// * Règles EOS standards: m, ρ, c, p, e
// ******************************************************
ComputeDensity: ∀j∈cells(), ρ{j} = m{j} / V{j};
ComputeEOSp: ∀j∈cells(), p{j} = (γ-1.0) * ρ{j} * e{j};
ComputeInternalEngergy: ∀j∈cells(), e{j} = E{j} - 0.5 *
```

$$\texttt{Compute}\delta\texttt{tj} : \forall j \in cells(),\ \delta tj_j = \frac{2.0 \cdot V_j}{(c_j \cdot \sum_{r \in nodesOfCell(j)} (l_{j,r}))}$$

NabLab Compilation Chain



Model conforming to Nablab metamodel

Model conforming to IR metamodel

Nabla file

```
∀ cells @ -18.0 { ρE = p/((γ-1.0)*ρ); }

∀ cells @ 9.0 if (option_glace){ // 10%
  const ℝ ρc = λ*ρ*c;
  ∀ node{
    AQs = CQs⊗CQs;
    AQs *= ρc/absCQs;
  }
}

∀ cells @ 12.0{ // 10%
  ∀ node{
    const ℝ²×ℝ² local_matrix = AQs;
    node_u_matrix += local_matrix;
    node_u_second_member += local_matrix ⊗ reconstructed_u
    node_u_second_member += CQs*reconstructed_p;
  }
}

∀ cells @ 16.0{ // 14%
  ∀ node{
    const ℝ² Δu = reconstructed_u-node_u;
    ℝ² FQs = AQs ⊗ Δu;
    FQs += reconstructed_p*CQs;
    momentum_fluxes_Σ -= FQs;
    total_energy_fluxes_Σ -= FQs·
  }
}
```

Execution results

Table of Contents

01

Context

Context - Towards greater abstraction - Why is DSL a good candidate ?

02

Description of developments

What is ∇ ? - What is NabLab ? - NabLab Compilation Chain - Demo

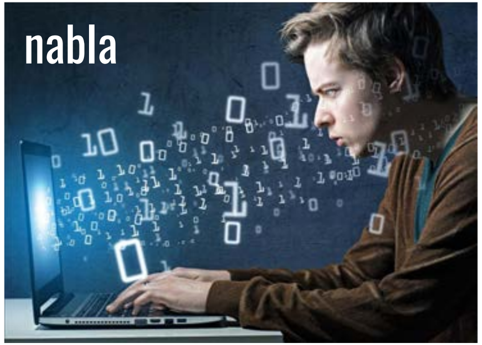
03

Conclusion & Perspectives

Grammar based DSL vs Metamodel based DSL - Towards a new Generation Strategy - Perspectives

Grammar based DSL *vs.* Metamodel based DSL

2013



➕ C++ language & time-honored technologies like Flex/Bison

➖ No facilities for editing code (UTF8, @) or debugging

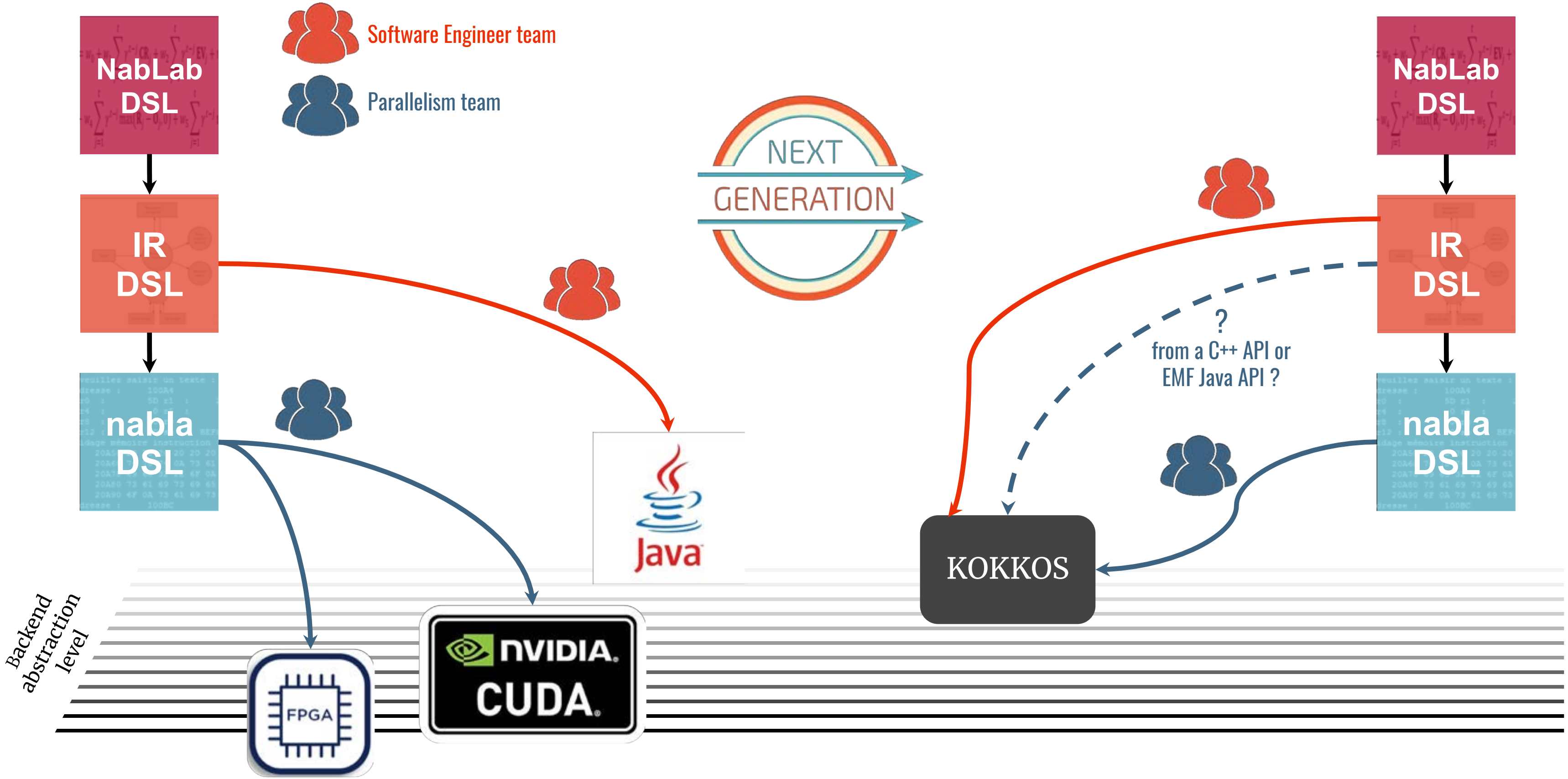
From mathematics
to optimization

2016

➖ Far from optimization concerns

➕ EMF-based language that offers user-friendly environment

Towards a New Strategy



Software Engineer team

Parallelism team

NabLab DSL

IR DSL

nabla DSL

NEXT GENERATION

NabLab DSL

IR DSL

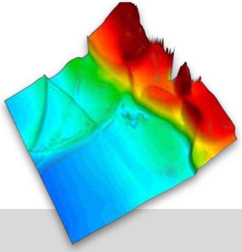
nabla DSL

?
from a C++ API or
EMF Java API ?

KOKKOS

Backend abstraction level

Perspectives



Future work includes **debugging** and **visualization** of results (step-by-step execution, variable inspection, curves, 2D/3D mesh visualization)

We plan to study the use of the **LLVM** Compiler Infrastructure **as an Intermediate Representation**

The ultimate challenge would be to **coordinate NabLab with Modane**, another CEA DSL dedicated to the design, to cover both the architectural and behavioral aspects of the simulation codes

Open to collaboration with the scientific computing community

Take Away Messages

- Abstraction is key
  - For advanced developments
  - For optimized computations
- Complementarity of domain-specific frameworks
  - Domain experts (business domain)
  - Dedicated execution environments (technical domain)
- Fruitful separation of concerns (software eng., optimization eng., and mathematicians and physicists)
- Important tradeoffs regarding the language workbenches, acceptable learning curve in general

Commissariat à l'énergie atomique et aux énergies alternatives
DAM Ile de France | Bruyères le Châtel
91297 ARPAJON CEDEX

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019