FROM RESEARCH TO INDUSTRY

Deep-sea project
NabLab WP4 contribution

Benoît LELANDAIS - François LETIERCE - **Rolih MEYNARD** - Marie-Pierre OUDOT
CEA, DAM, DIF, F-91297 Arpajon, France

Deep-sea - September 17, 2021

www.cea.fr

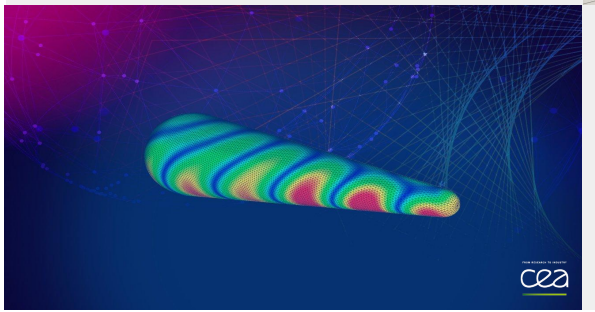
Table of Contents

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

NabLab : language & environment

NabLab is both a DSL for numerical analysis and an Eclipse environment

NabLab-runtime - NabLabExamples/src/explicitheatequation/ExplicitHeatEquation.n - Eclipse Platform

File Edit Source Code Navigate Search Project Run Window Help

Project Explorer

- MoniLoggers
- NabLabExamples
  - JRE System Library [JavaSE-11]
  - Plug-in Dependencies
  - src
    - explicitheatequation
      - ExplicitHeatEquation.json
      - ExplicitHeatEquation.n
      - ExplicitHeatEquation.ngen
    - glace2d
    - heatequation
    - implicitheatequation
      - ImplicitHeatEquation.json
      - ImplicitHeatEquation.n
      - ImplicitHeatEquation.ngen
    - iterativeheatequation
  - src-gen-java
  - META-INF
  - src-gen
  - src-gen-cpp
  - build.properties
  - README.txt
- swan
- swandoc

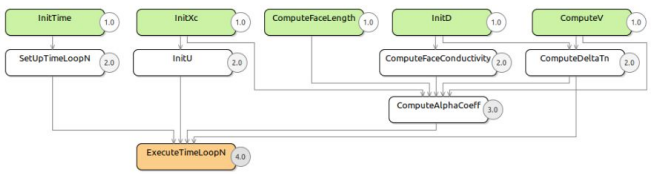
Outline

- Module ExplicitHeatEquation
- Job InitTime
- Job InitXc
- Job InitU
- Job InitD
- Job ComputeDeltaTn
- Job ComputeV
- Job ComputeFaceLength
- Job ComputeFaceConductivity
- Job ComputeAlphaCoeff
- Job UpdateU
- Job ComputeTn

ExplicitHeatEquation.n

```
module ExplicitHeatEquation;

with Math.*;
with CartesianMesh2D.*;

option R u0 = 1.0;
let R[2] vectOne = R[2](1.0);

// Simulation options
option R stopTime = 1.0;
option N maxIterations = 500000000;

// User options
let R δt = 0.001;
R t;
R[2] X{nodes}, Xc{cells}; // Position of nodes and cells center of gravity
R u{cells}; // Temperature
R V{cells}; // Volume of cells
R D{cells}; // Cell centered conductivity
R faceLength{faces}, faceConductivity{faces};
R α{cells, cells};

iterate n while (t^{n+1} < stopTime && n+1 < maxIterations);

InitTime: t^{n=0} = 0.0;

InitXc: ∀c∈cells(), Xc{c} = 0.25 * ∑{p∈nodesOfCell(c)}(X{p});  // Only valid on parallelograms

InitU: ∀c∈cells(),
    if (norm(Xc{c} - vectOne) < 0.5)
        u^{n}{c} = u0;
    else
        u^{n}{c} = 0.0; // Initial circle in the center with value u0

InitD: ∀c∈cells(), D{c} = 1.0;

ComputeDeltaTn: δt = Min{c∈cells()}(V{c}/D{c}) * 0.24;
ComputeV: ∀c∈cells(), V{c} = 0.5 * ∑{p∈nodesOfCell(c)}(det(X{p}, X{p+1}));
ComputeFaceLength: ∀f∈faces(), faceLength{f} = 0.5 * ∑{p∈nodesOfFace(f)}(norm(X{p} - X{p+1}));
ComputeFaceConductivity: ∀f∈faces(), faceConductivity{f} = 2.0 * ∏{c1∈cellsOfFace(f)}(D{c1}) /
    ∑{c2∈cellsOfFace(f)}(D{c2});
```

ExplicitHeatEquation



LaTeX | Job Graph | Console | Properties | Problems | Error Log | Search

$$\texttt{ComputeFaceLength} : \forall f \in faces(),\ faceLength_f = 0.5 \cdot \sum_{p \in nodesOfFace(f)} \|X_p - X_{p+1}\|$$

Writable | Insert | 49 : 54 : 1518

Open-source project

https://github.com/cea-hpc/NabLab

Language

```
// Assembling of the diffusion matrix
ComputeAlphaCoeff: ∀c∈cells(), {
    let ℝ αDiag = 0.0;
    ∀d∈neighbourCells(c), ∀f∈commonFace(c,d), {
        let ℝ αExtraDiag = - δt / V{c} * (faceLength{f}
            * faceConductivity{f}) / norm(Xc{c} - Xc{d});
        α{c, d} = αExtraDiag;
        αDiag = αDiag + αExtraDiag;
    }
    α{c, c} = 1 - αDiag;
}
```

- Supports basic types (ℝ, ℕ, Γ, ℝ[2]...) and mesh types (ℝ[2] X{nodes}...)
- Allows loops on mesh elements and space iterators on variables

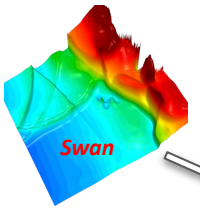
```
ℝ Uini{faces};
InitUini: ∀f∈innerFaces(), Uini{f} = 0.0;
```

- Allows time loops and time iterators on variables

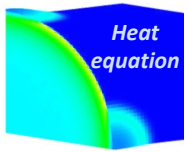
```
iterate n while (t^{n+1} < stopTime && n < maxIter);
ComputeTn: t^{n+1} = t^{n} + δt;
```

- Natively integrates
  - mesh library (cells, nodes, faces …) ,
  - mathematical library (∑, ∏, abs, cos, matVectProduct...) and
  - linear algebra library
- Allows declaration of functions (NabLab functions and external functions)

NabLab Compilation Chain



…



NabLab

Transform passes

Intermediate representation

Java

- ❑ MPI
- ✓ Multi-threads
- ❑ Vectoriel
- ❑ GPU

KOKKOS

- ❑ MPI
- ✓ Multi-threads
- ❑ Vectoriel
- ❑ GPU

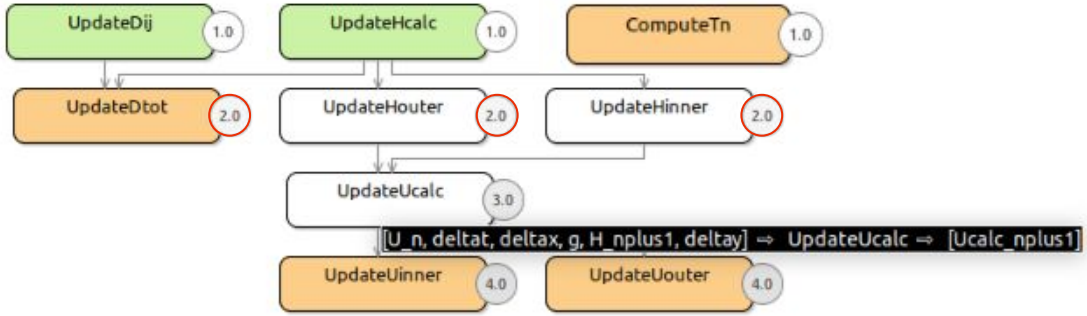
OpenMP

- ❑ MPI
- ✓ Multi-threads
- ❑ Vectoriel
- ❑ GPU

- ❑ MPI
- ❑ Multi-threads
- ❑ Vectoriel
- ❑ GPU

■ Work in progress

Scheduling



- Construction of the dataflow graph by analyzing the input variables and output variables of the jobs.

- Job scheduling thanks to Hierarchical Logital Time. Jobs with a lower HLT will be executed before. Jobs with the same HLT can be executed in parallel.

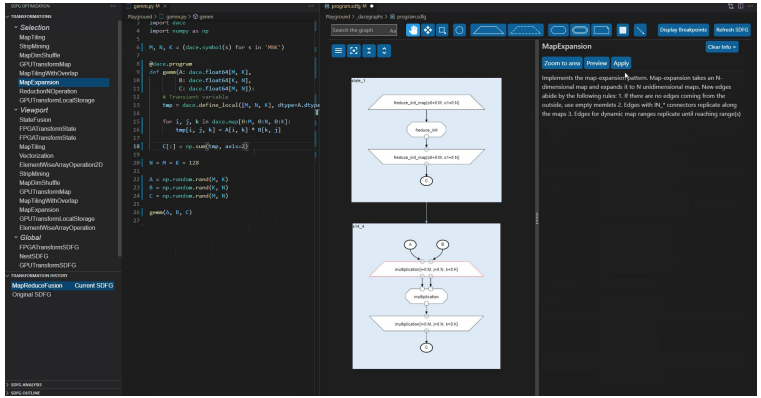
- Circular dependencies on jobs are not tolerated inside a time loop iteration.

Table of Contents

Goal

CEA/DAM contribution for Deep-sea project : generate DaCe Sdfg from a NabLab model

Then this SDFG could be imported in DaCe environment (Visual Studio Code plugin for example) to benefit from transformation and optimisation capabilities



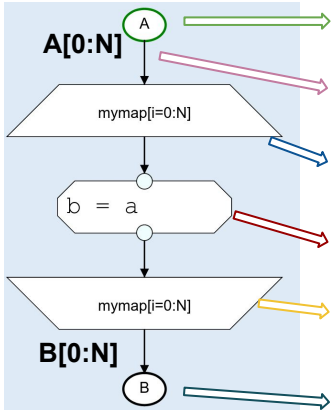
CEA/DRT contribution for Deep-sea project : generate sdf3 data centric representation from a NabLab model

Very similar goal, but different technology

DaCe - Data-Centric Parallel Programming

DaCe is a parallel programming framework: takes Python code and maps it to high-performance programs (**CPU, GPU, and FPGA**) which can be optimized.

DaCe uses a Stateful DataFlow multiGraph (SDFG) *data-centric intermediate representation* and provides a Python API to create SDFG

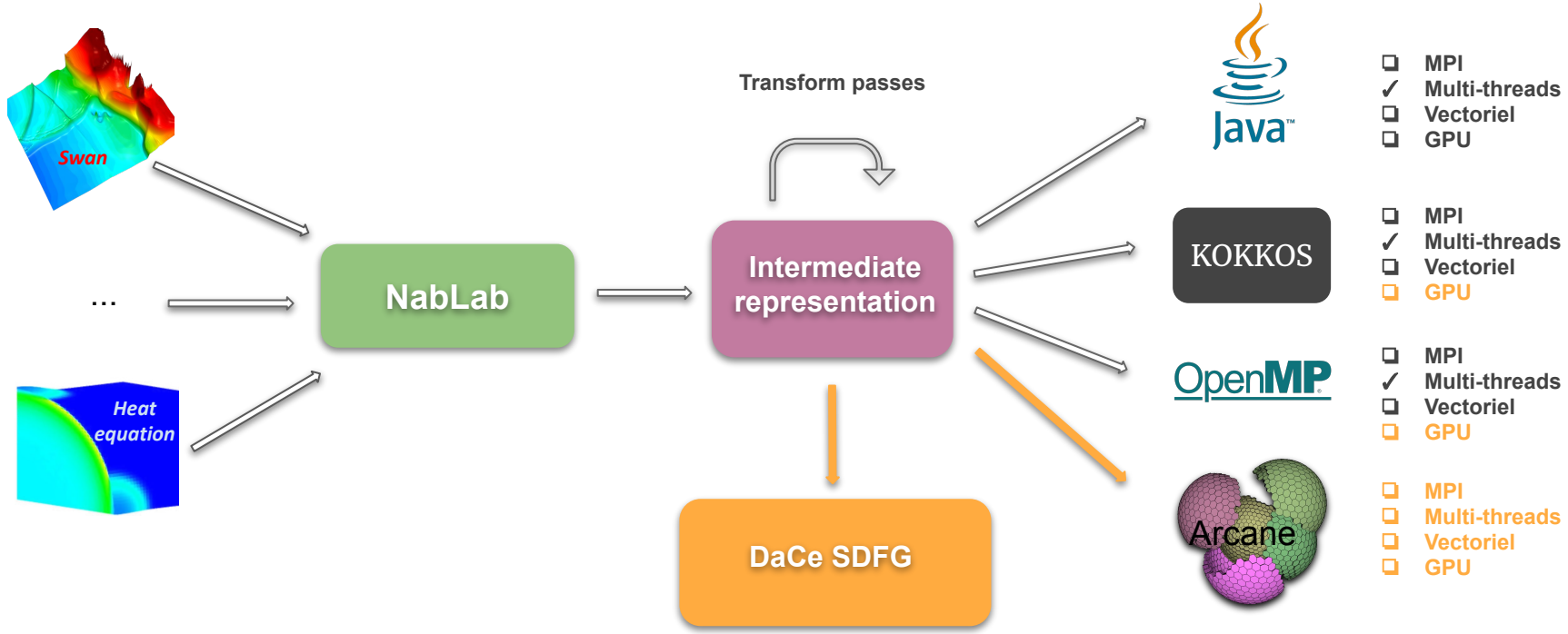


Example of a SDFG graph

Input data	A: 1-dimensional array container (With N elements)
Memlet	Data movement descriptor
Map entry	Represent parallel computation (loop to iterate on all the elements of the vector A, each elements of the vector A are read)
Tasklet	Tasklets nodes contain arbitrary computational functions of any granularity
Map exit	Represent parallel computation (loop to iterate in order to affect all the elements of the vector A into the vector B)
Output data	B: 1-dimensional array container (contains the N elements of the Vector A)

Method

The idea is to generate from a NabLab IR (intermediate representation), a Python script that creates the corresponding SDFG.



■ Work in progress

Work progress

Workload set for 24 months to generate DaCe SDFG from NabLab model (started since 2 months)

(1) NabLab program

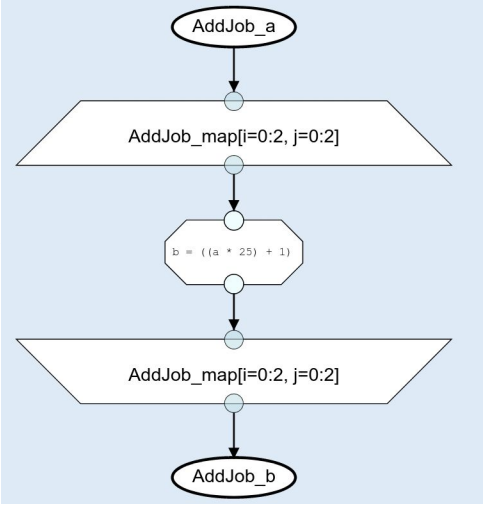
SdfgAdditionMatrix2D.n

```
module SdfgAdditionMatrix2D;

let ℕ[2,2] a =  ℕ[2,2](5);
let ℕ[2,2] b =  ℕ[2,2](0);

AddJob: b = a * 25 + 1;
```

(2) generated Python script

```
b = dp.ndarray([2, 2], dp.int64)
b[:] = dp.int64(0), dp.int64(0)
a = dp.ndarray([2, 2], dp.int64)
a[:] = dp.int64(5), dp.int64(5)

mysdfg = SDFG('SdfgAdditionMatrix2D')

AddJob = mysdfg.add_state()

AddJob_tasklet = AddJob.add_tasklet('AddJob', {'a'}, {'b'}, 'b=a*25+1')

AddJob_a = mysdfg.add_array('AddJob_a', [2, 2], dp.int64)
AddJob_b = mysdfg.add_array('AddJob_b', [2, 2], dp.int64)

map_entry, map_exit = AddJob.add_map('AddJob_map', dict(i='0:2',j='0:2'))

AddJob.add_memlet_path(AddJob.add_read('AddJob_a'),map_entry, AddJob_tasklet, dst_conn='a',memlet=dace.Memlet('AddJob_a[i,j]'))
AddJob.add_memlet_path(AddJob_tasklet, map_exit, AddJob.add_write('AddJob_b'), src_conn='b',memlet=dace.Memlet('AddJob_b[i,j]'))

mysdfg(AddJob_a=a,AddJob_b=b)

mysdfg.view('programSDFGFromNabLab.html')
```

(3) generated DaCe SDFG



Continuation of the work: risk removal on various technical points before proposing an implementation schedule

Conclusion

- NabLab is a Domain Specific Language (DSL) for numerical analysis to generate optimized code for different targets and architectures

∇ NabLab

- NabLab comes with a dedicated Eclipse environment

eclipse

- For DEEP-SEA project, we will generate DaCe SDFG from NabLab

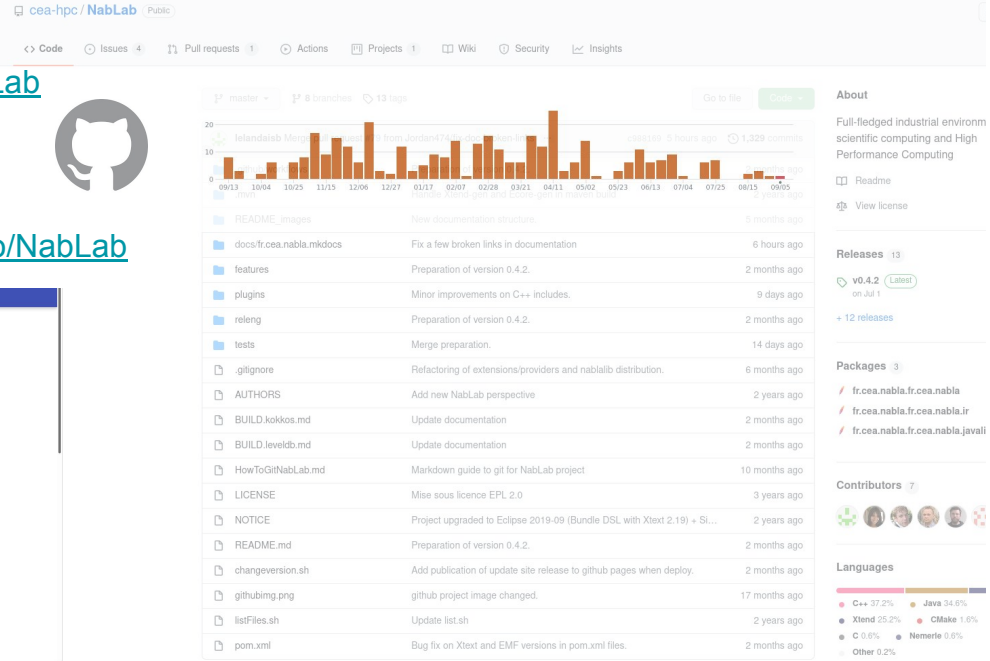
DaCe - Data-Centric Parallel Programming

- Thanks to this SDFG, we will have the opportunity to optimize NabLab programs by using Visual Studio Code plugin

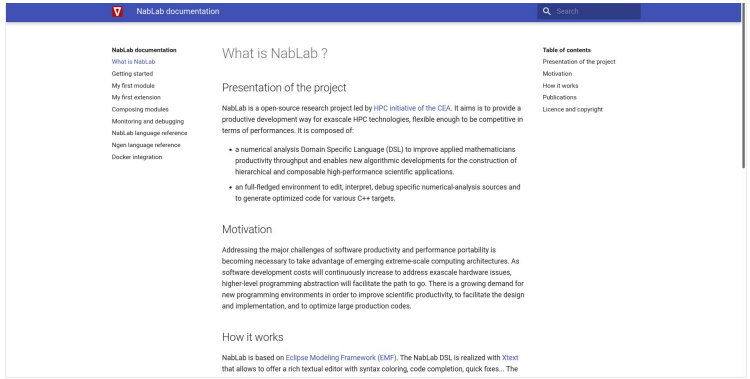
Visual Studio

More information

- See NabLab github: https://github.com/cea-hpc/NabLab



- See NabLab documentation: https://cea-hpc.github.io/NabLab



- Contact: Rolih.MEYNARD@cea.fr

Questions are welcome

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