DE LA RECHERCHE À L'INDUSTRIE

# The ∇-Nabla Time-Composite Approach for Multi-Physics Applications Productivity

CAMIER Jean-Sylvain

CEA, DAM, DIF F-91297, Arpajon France

March 10, 2015

www.cea.fr

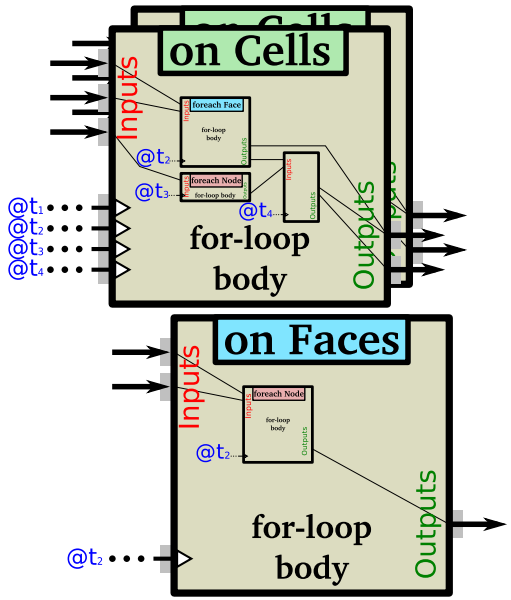
DE LA RECHERCHE À L'INDUSTRIE

# Outline

# Introduction: Where does it come from?

## Background → Transposed to HPC

- 7 years of HW design and engineering
  - Influenced by VHDL
  - A re-usable source for multiple target architectures
  - Think: "Simulate/Validate, Synthesis, Place & Route"
- 3 years of RT/Crit. Systems dev. and methodology




## Objectives & Roadmap since 2009

- **Performances**: Instantiate the right programming model for ≠ SW/HW stacks
- **Portability**: Provide portable scientific applications across architectures
- **Programmability**: Attractive approach for tomorrow's SW engineers
- **Interoperability**: Allow modularity with legacy codes

DE LA RECHERCHE À L'INDUSTRIE

# Outline

# ∇ Code Development Strategy

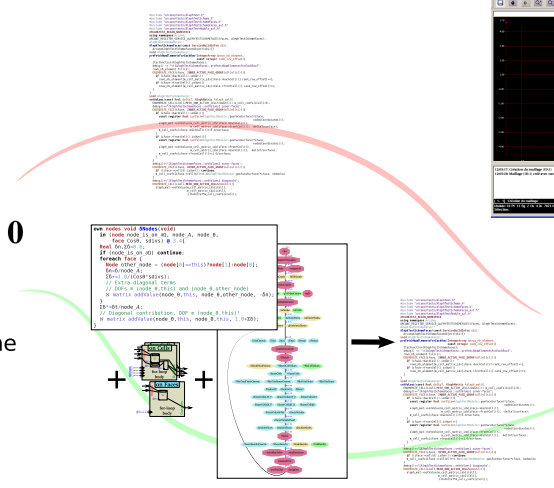
Challenges for Application Developers:

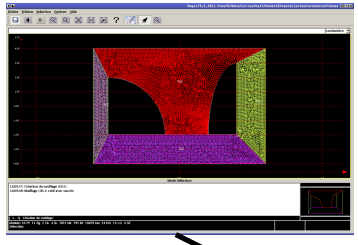
Abstractions:

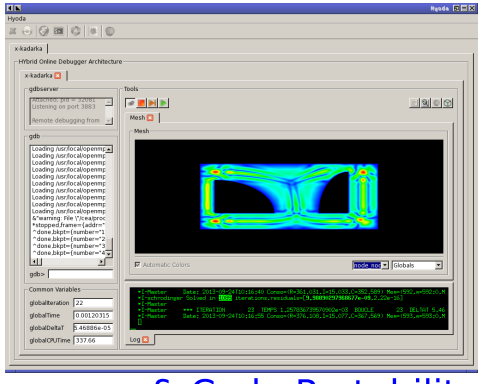
Tuning

$$i\,\Delta_n^1(\psi(n,i,j)) + \lambda\,\mathsf{E}_n\left(\mathsf{E}_i^{-1}\left(\Delta_i^2(\psi(n,i,j))\right) + \mathsf{E}_j^{-1}\left(\Delta_j^2(\psi(n,i,j))\right)\right) = 0$$

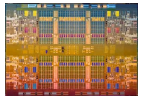
Schrödinger - 2D Finite Differences Scheme



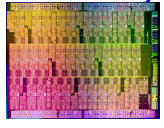




High Performances & Code Portability



or NVIDIA. or



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

- ⇒ Proposition of the ∇ **mesh-based numerical operations** Specific Language
  - Raises the level of **abstraction**: superset of a subset of C
  - Provides a Bottom-UP **component** approach that provides:
    - A Methodology for us to **Co-Design** between: **Applications** ⟺ **SW** layers
    - Existing Middlewares, Heterogeneous Executions models (Compute+GPU)
  - Introduces **logical time** partial-ordering to gain an additional dimension in concurrency

# ∇ Code Development Methodology

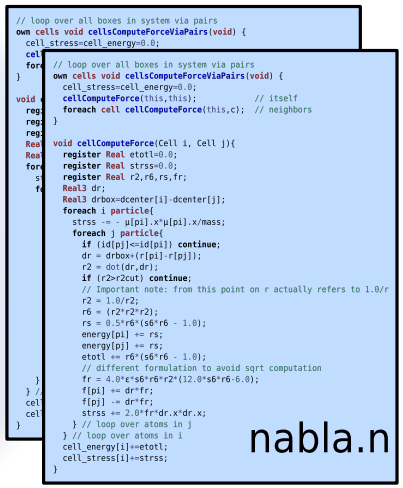
## Proxy-Apps

ExMatEx

## ∇ Howto:

- App. **options**
- Data **variables**
- **For-Loops** focus
- **Constraints**:
  no glue-code
  '@' concurrency

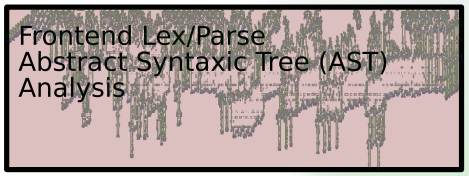
## Same ∇ sources

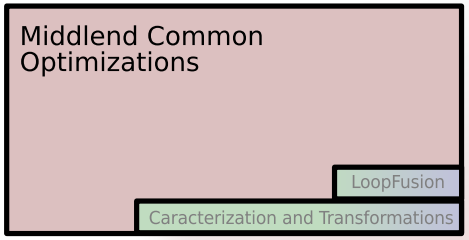



nabla [-t] [-v log] -[arcane|okina|cuda] -i inputFiles

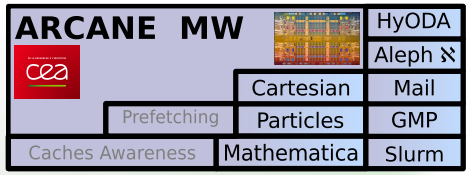
Compilation command line

## Sources Analysis

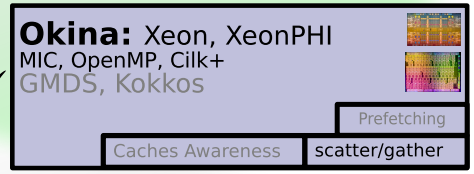


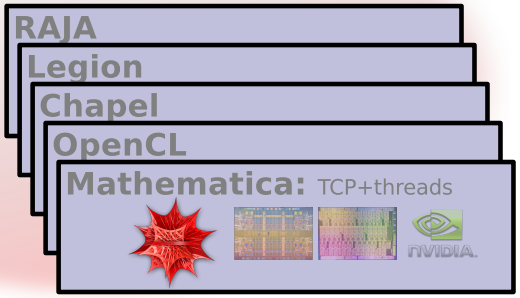

## Optimizations & Transformations








## Multiple Targets Generation



DE LA RECHERCHE À L'INDUSTRIE

cea

# Outline

# ∇'s Parallel Programming Approach (1/3)

## Explicit declaration: **Libraries**, **Options** and mesh **Variables**:

```
with ℵ,cartesian;
// ℵ ≡ aleph ≡ Linear Algebra Extended
//             to Hybrid Parallelism
options{
  Real γ                          = 1.4;
  Real option_δt_fixed            = 1.e-7;
  Integer option_max_iterations   = 1024;
  Bool option_quads               = true;
};
```

```
cells {
  Real cell_A;
  Real3 CQs[nodes];
  Real3x3 AQs[nodes];
  Integer cell_nb_nodes;
};

nodes {
  Real3 u,ū;
  Real node_A;
};
```

```
faces {
  Real α,β,δ,γ,σ;
  Real Cosθ,Sinθ;
};

particles{
  Real3 ρ;
  Real3 μ;
};
```

## Data-parallelism is **implicitely** expressed via jobs **items**:

```
cells void geomComputeQuadSurface(void)
  out (cell cell_A) @ -10.0 if (option_quads){
  Real3 fst_edge = coord[2]-coord[0];
  Real3 snd_edge = coord[3]-coord[1];
  cell_A=½*(fst_edge⊗snd_edge);
 }

nodes void geomComputeNodeArea(void)
  in (cell cell_A) out (node node_A) @ -8.5{
  node_A=0.0;
  foreach cell node_A+=cell_A/nbNode;
}
```

```
cells void geomComputeQuadSurface(void)
  out (cell cell_A) @ -10.0 if (option_quads){
  Real3 fst_edge = coord[2]-coord[0];
  Real3 snd_edge = coord[3]-coord[1];
  cell_A=½*(fst_edge⊗snd_edge);
 }

nodes void geomComputeNodeArea(void)
  in (cell cell_A) out (node node_A) @ -8.5{
  node_A=0.0;
  foreach cell node_A+=cell_A/nbNode;
}
```

# ∇'s Parallel Programming Approach (2/3)

Jobs parallelism is **explicitely** declared via Hierarchical Logical Time (HLT)

- the '@' statements ensure a partial order between all jobs

```
cells void computeAQs(void)
  in (cell λ,ρ,c,CQs,absCQs)
  out (cell AQs) @ -1.0,16.0{
  Real ρc = λ*ρ*c;
  foreach node{
    AQs = CQs⊗CQs;
    AQs *= ρc/absCQs;
  }
}
```

```
cells void computeStdFluxΣ(void)
  in (cell p,u,ū,AQs,CQs)
  out (cell mfΣ, EfΣ) @ -∞,-1.0,π²{
  foreach node{
    Real3 Δu = u-ū;
    Real3 FQs = AQs⊗Δu;
    FQs += p*CQs;
    mfΣ -= FQs;
    EfΣ -= FQs⊙ū;
  }
}
```

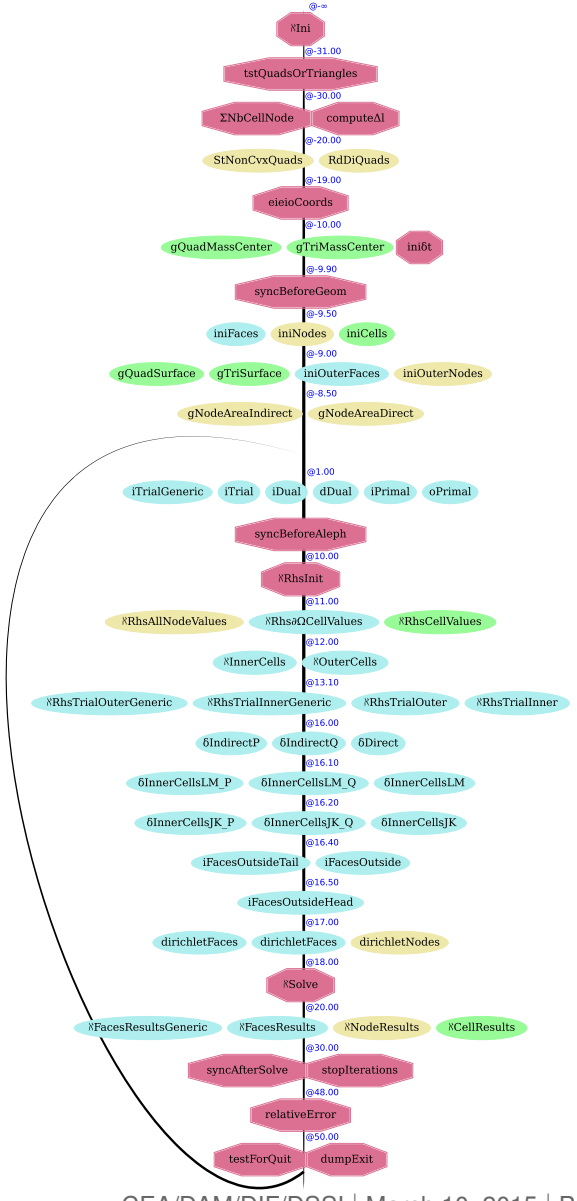
- Consistency and liveliness can be analysed and prooved offine
- Instrumentation can be integrated in the framework
- Optimization and characterization stages are inserted before generation
  - loop fusion, data layout, prefetching, caches awareness, vectorization

# ∇'s Parallel Programming Approach (3/3)

```
own nodes void δNodes(void)
  in (node node_is_on_∂Ω, node_A, node_θ,
      face Cosθ, sdivs) @ 3.4{
  Real δn,Σδ=0.0;
  if (node_is_on_∂Ω) continue;
  foreach face {
    Node other_node = (node[0]==this)?node[1]:node[0];
    δn=δ/node_A;
    Σδ+=1.0/(Cosθ*sdivs);
    // Extra-diagonal terms
    // DOFs ≡ (node_θ,this) and (node_θ,other_node)
    ℵ matrix addValue(node_θ,this, node_θ,other_node, -δn);
  }
  Σδ*=δt/node_A;
  // Diagonal contribution, DOF ≡ (node_θ,this)²
  ℵ matrix addValue(node_θ,this, node_θ,this, 1.0+Σδ);
}
```

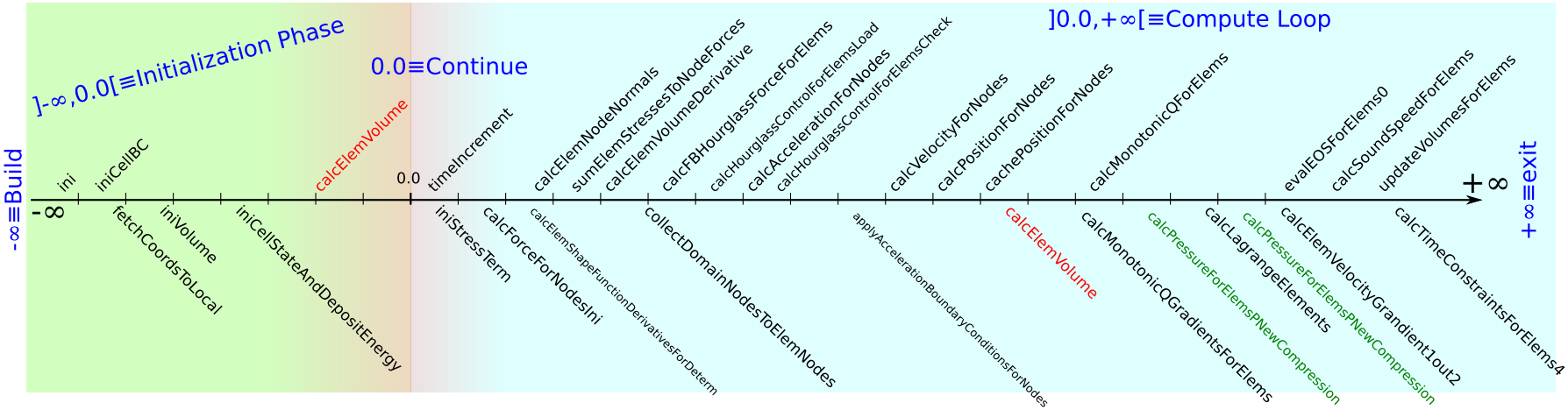
```
                void rhsInit(void) @ 1.0 {…}
      own cells void setRhsCellValues(void) … @ 1.1 {…}
      own nodes void setRhsNodeValues(void) … @ 1.1 {…}
own outer faces void setRhsFaceValues(void) … @ 1.1 {…}
own outer faces void setStdRhsEdgesOnBorders(void) … @ 1.2 {…}
      own cells void alphaCells(void) … @ 3.0 {…}
      own cells void betaCells(void) … @ 3.0 {…}
      own nodes void gammaNodes(void) … @ 3.0 {…}
      own nodes void deltaNodes(void) … @ 3.0 {…}
own outer faces void dirichletFaces(void) … @ 3.0 {…}
      own nodes void dirichletNodes(void) … @ 3.0 {…}
                void assembleAndSolve(void) @ 4.0 {…}
      own cells void getCellResults(void) … @ 4.3 {…}
      own nodes void getNodeResults(void) … @ 4.3 {…}
      own cells void saveCellDensity(void) … @ 4.4 {…}
      own nodes void saveNodeDensity(void) … @ 4.4 {…}
```

⇒

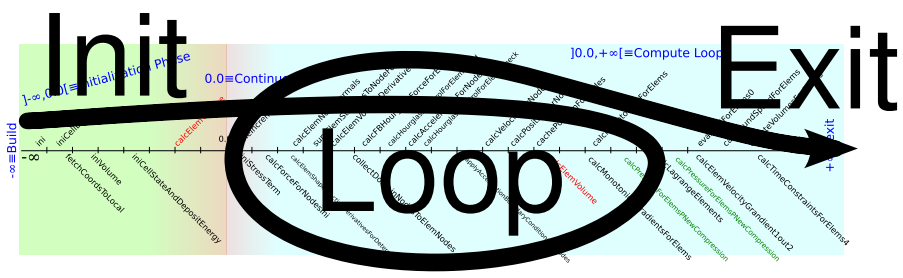


# Hierarchical Logical Time: Presentation on ∇ LULESH

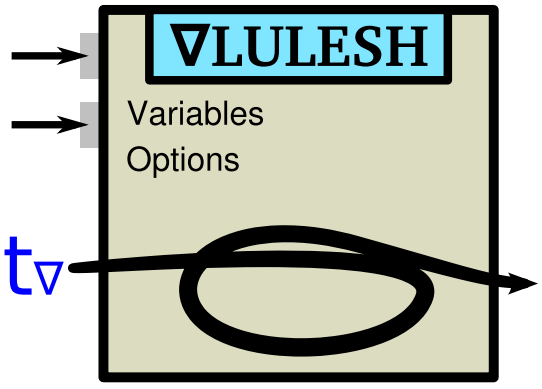
Table: ∇ LULESH Logical Time (≠ Point Of Views)




1 - Logical Timeline of ∇ LULESH jobs



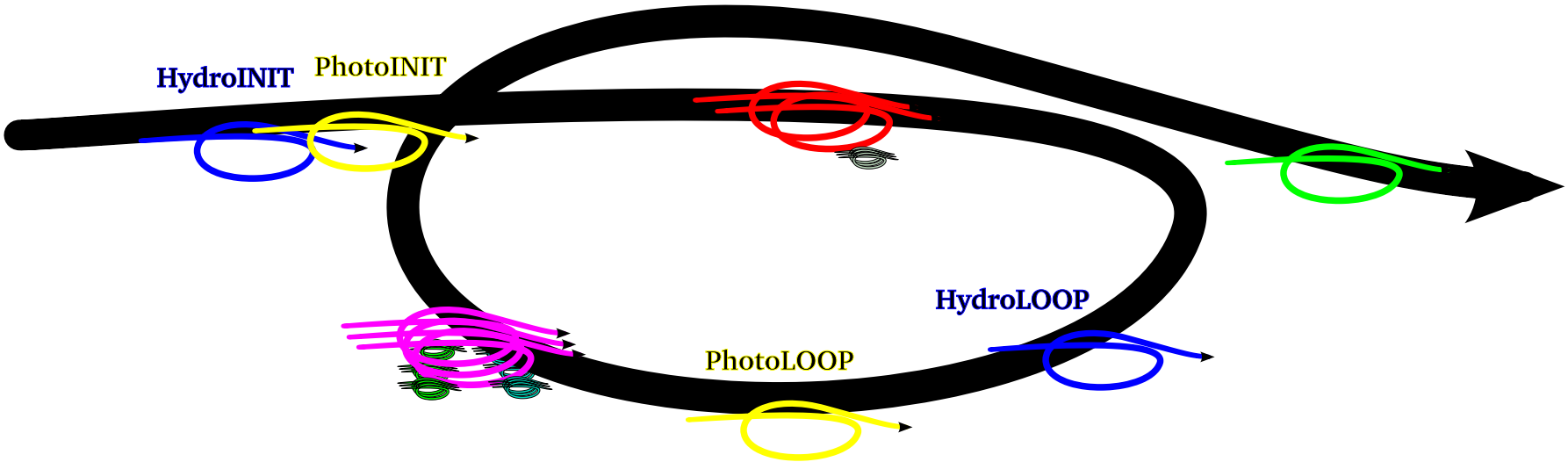

2 - Timeline ⇒ Swirl-loop point of view




3 - Logically-timed component pov

# Hierarchical Logical Time: Nested Composition

- **Application** $\equiv$ **Nested Composition** of such logically-timed **components**




- Each *component* or *entity* is instanciated hierarchically
  - Now via command-line
- $\Rightarrow$ The need of front-end tools will rapidly be crucial as applications will grow bigger!

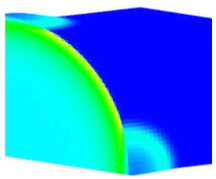
DE LA RECHERCHE À L'INDUSTRIE

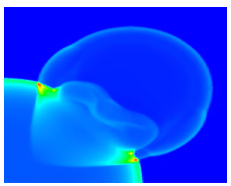
cea

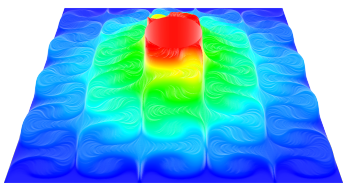
# Outline

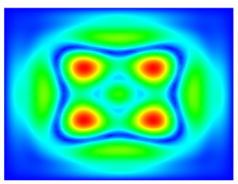
# Main Proxy Applications ported to ∇

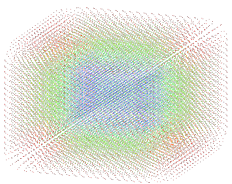
| Numerical Methods | Application | # of ∇ lines |
|---|---|---|
| Explicite Unstructured (+scatter/gather) | LULESH (**1.0**)(LLNL) | 1030 |
| Explicite Structured | HYDRO(CEA) | 757 |
| Implicite | M-NL-DDFV (CEA/DAM)<br>Schrödinger (CEA/DAM) | 2304<br>375 |
| Monte-Carlo | MCTB(CEA/DAM) | 828 |
| Dynamique Molecular | CoMD(LANL)<br>MiniMD(SANDIA) | 293<br>474 |
| SPH | SPH(CEA/DAM) | 2500 |

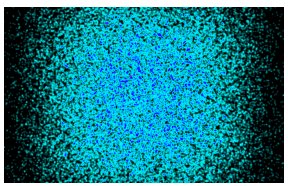

LULESH


HYDRO


MNLDDFV


Schrödinger


CoMD


SPH

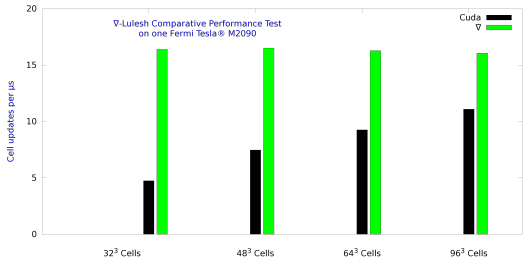
DE LA RECHERCHE À L'INDUSTRIE

cea

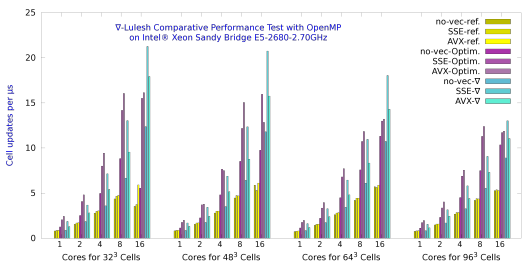
# Outline

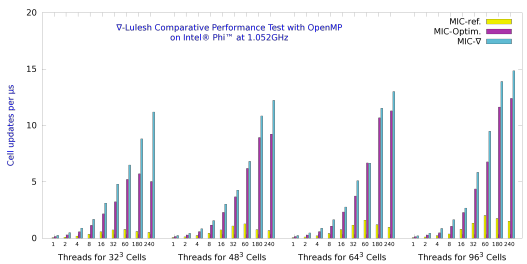
# ∇-LULESH: 1.0

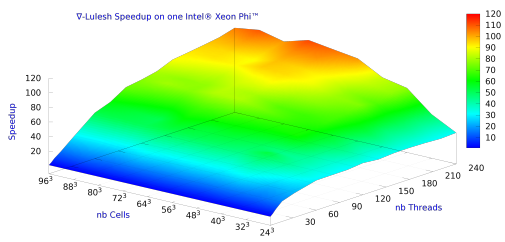
- Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics
- 3k sloc serial programing model vs 1k sloc w/o comments
  - Same average ratio for other programs
- Differents backends are directly available:
  - **ARCANE**: with Threads, MPI, MPI+Threads, MPC
  - **CUDA**: single-node code generation




- **OKINA**: Stand-alone C/C++ native programming model
  - (OpenMP or Cilk+) with **no-vec**, **SSE**, **AVX**, **AVX512** or **MIC**
  - First try with **full intrinsics** and **gather/scatter generated instructions**

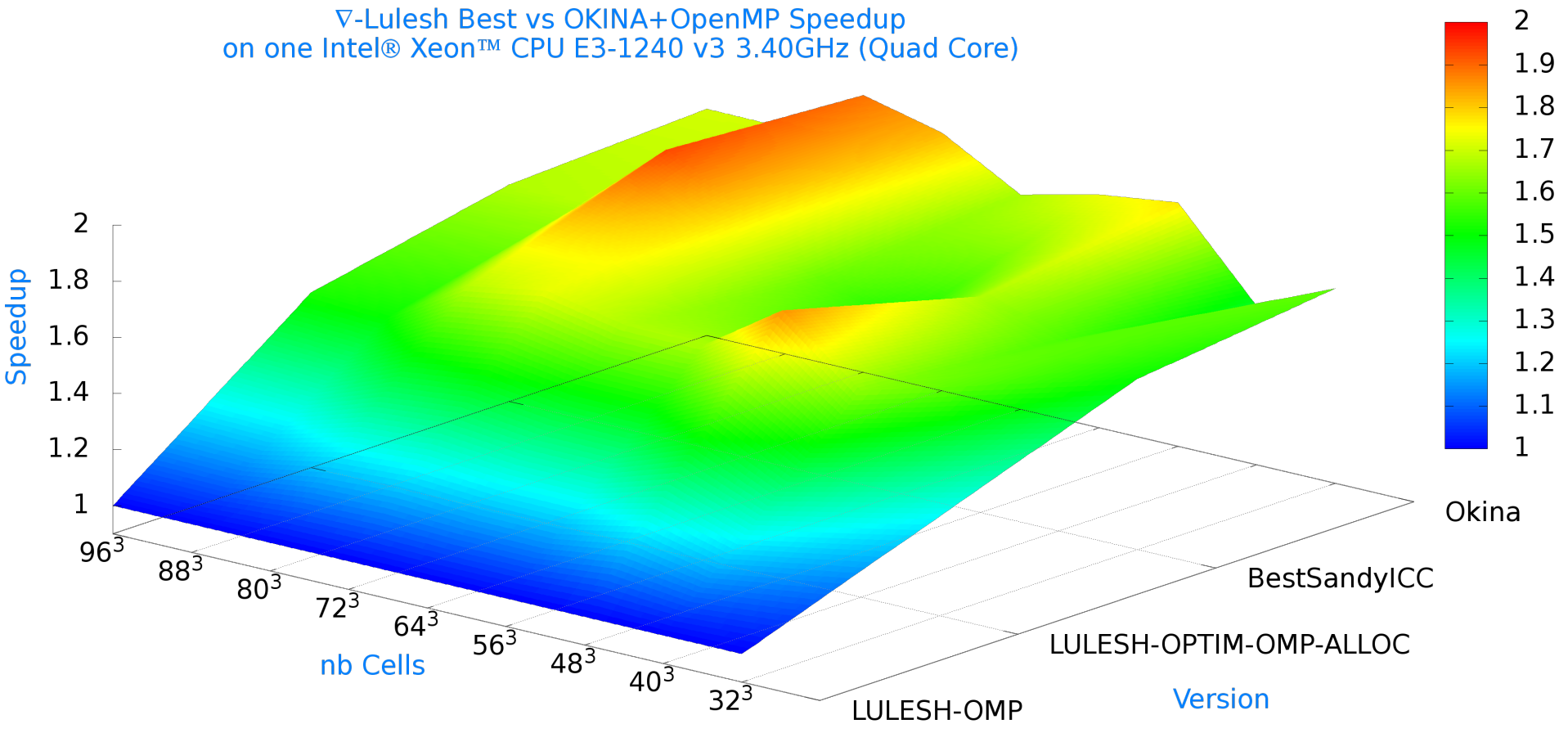



- **Yours**. . .

# ∇-LULESH Comparative Performance Test on Intel® Xeon Haswell™



# Outline

# Conclusion & Roadmap

## Conclusion

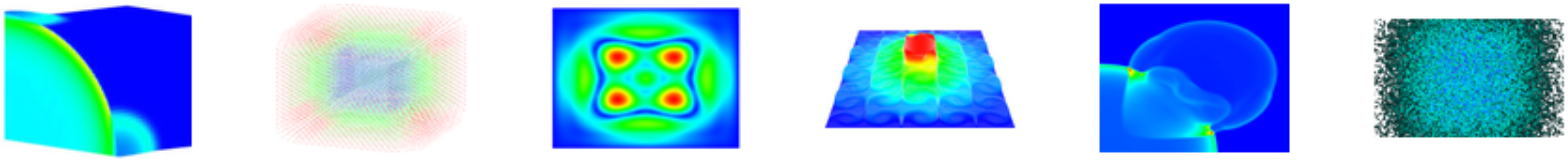
- ∇ DSL work-in-progress ⇒ Raise Loop-Level's **abstraction**!
  - No need to *choose* today the best programming model for tomorrow's architectures
  - Does not require to code multiple versions of kernels for ≠ models
- Helps transition from Bulk-Synchronous-Programming (BSP)
  - **Performances**: Allowing specific optimization phases before code generation
  - **Portability**: Adaptable towards diversified & complex emerging architectures
  - **Productivity**: Simple, Attractive & Elegant
- **CeCILL** v2.1 license, under French law ⇒ www.nabla-lang.org
  - **The CeCILL is a free software license, explicitly compatible with the GNU GPL**

## Backends & Roadmap

- Lawrence Livermore National Laboratory : **RAJA** Portability Layer, Los Alamos NATIONAL LABORATORY EST. 1943: **Kokkos**, **Legion**
- cea: **Okina** (MPC, OpenMP, Cilk+)
- NVIDIA: **CUDA**, CRAY CHAPEL: **Chapel**
- Proxy applications: ports and performance evaluations

∇-Nabla: Numerical Analysis BAsed LAnguage

Home Overview Licence Getting Started Publications Downloads



## ∇ Presentation

The ∇ domain-specific language (DSL) provides a productive development way for exascale HPC technologies, flexible enough to be competitive in terms of performances.

This software is a computer program whose purpose is to translate specific numerical-analysis sources and generate optimized code for different targets and architectures.

## Latest Tarball

**Latest Tarball is 150218**

**<2015-02-18 Wed> Prototype #150218 released**

This is the first release, with ∇ Lulesh as the main example. The ∇ compiler can generate sources from ∇ LULESH to three different backends:

- **Arcane**: our CEA's middleware,
- **Cuda**,
- **Okina**: a C/C++ stand-alone fully vecorized backend for standard compilers with OpenMP, Cilk+ or MPC.

Several other tests are given as examples:

- CoMD: a ∇ port of the 'Molecular Dynamics Proxy Applications Suite', one of many proxy applications that support of

Commissariat à l'énergie atomique et aux énergies alternatives
Centre DAM-Ile de France - Bruyères-le-Châtel 91297 Arpajon Cedex
T. +33 (0)1 69 26 63 09 | F. +33 (0)1 69 26 70 12
Établissement public à caractère industriel et commercial
RCS Paris B 775 685 019

Direction des Applications Militaires
Département Sciences de la Simulation et de l'Information