DE LA RECHERCHE À L'INDUSTRIE

www.cea.fr

∇-Nabla: A Numerical-Analysis Specific Language for Exascale Scientific Applications

PPSC 2014 | **JS. Camier**

CEA, DAM, DIF F-91297, Arpajon France

Portland OR, February 2014

Introduction

Background → Transposed to HPC

- 7 years of HW design and engineering
  - Think: "simulate/validate, synthesis, place & route"
  - A single source for multiple computer architectures
- 3 years of Real-Time Critical Systems development and methodology

Objectives/Roadmap since 2009

- Improve mathematicians' and numerical scientists' productivity throughput
- Provide portable/adaptable/maintainable scientific applications
- Target a variety of today and future architectures

DE LA RECHERCHE À L'INDUSTRIE

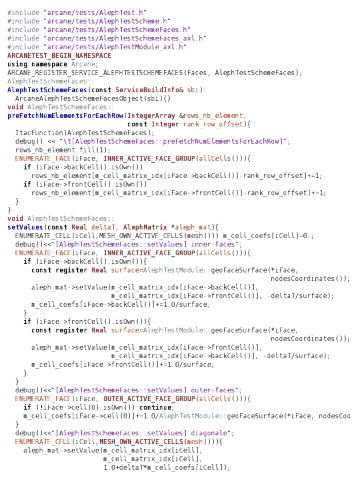
cea

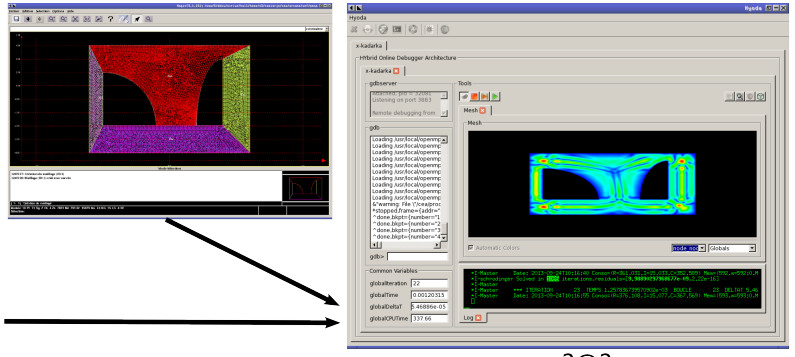
Outline

∇ Code Development Strategy

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

Schrödinger - 2D Finite Differences Scheme





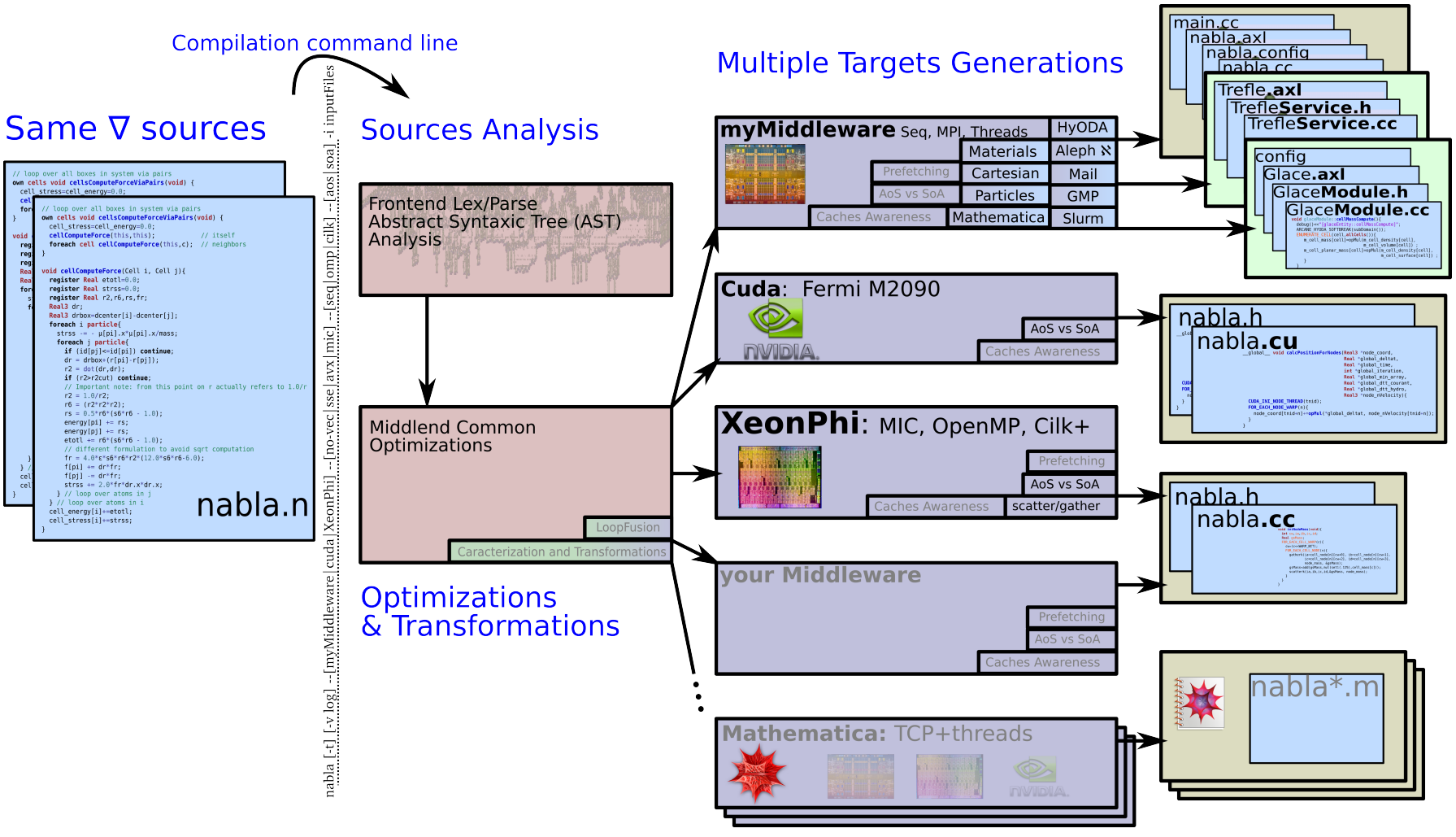
?@?

Concurency, Vectorization, Memory access, Storage, Resiliency

⇒ Proposition of the ∇ Domain Specific Language Toolchain

- Raise the level of abstraction: (superset of a subset of C)-to-source generation
- Address effective use of existing and fore-coming SW/HW stacks
- Provide a Bottom-UP component approach that targets:
  - Existing middlewares (mine, yours, native open ones, . . .)
  - Hetrogeneous Executions models
- Allow to expose additional concurrency
- Exploit algorithmic or low-level resiliency techniques

∇ Code Development Methodology



Outline

∇ Parallel Programming Approach (1/2)

- Libraries, Options and mesh variables **items** declaration:

```
with 3D,ℵ,cartesian;
options{
  Real option_δt_ixed             = -1.0e-7;
  Integer option_max_iterations   = 1024;
  Bool option_quads               = true;
};
```

```
cells {Real cell_A; Integer cell_nb_nodes;};
nodes {Real node_A;};
faces {Real α,β,δ,γ,σ; Real Cosθ,Sinθ;};
particles{Real3 r; Real3 μ;};
```

- Data-parallelism is **implicitely** expressed via underlying **items** of **jobs**

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

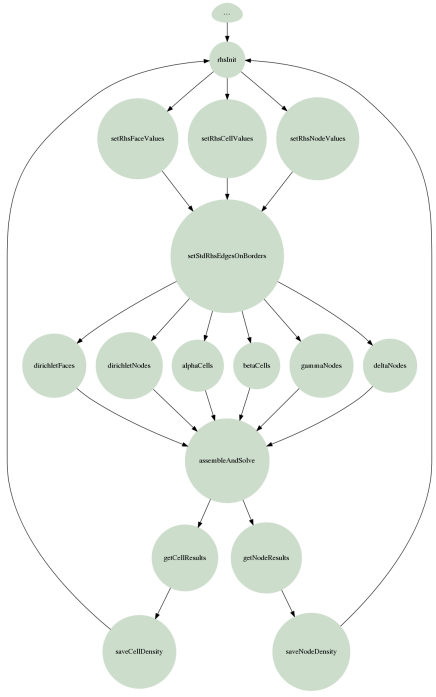
```
nodes void geomComputeNodeArea(void)
  in (cell cell_A) out (node node_A) @ -8.5{
  node_A=0.0;
  foreach cell node_A+=cell_A/nbNode;
}
```

- Tasks-parallelism is **explicitely** declared via **logical time**
  - the **@** statements ensure a partial order
  - **MPMD** concurency is reached **by construction**
- Consistency and liveliness can be analysed and prooved offline
- Optimization and characterization stages are inserted before generation
  - loop fusion, data structures, prefetching, caches awareness, vectorization

∇ Parallel Programming Approach (2/2)

```
                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 {…}
```

⇒




```
cells void calcEnergyForElems1(void)
  in (cell e_old, cell delvc, cell p_old, cell q_old, cell work)
  inout (cell e_new)@ 7.1{
  e_new = e_old - ½*delvc*(p_old + q_old) + ½*work;
  e_new = (e_new < option_emin)?option_emin;
}
```

```
own nodes void deltaNodes(void) @ 3.4{
  register Real δn,Σδ=0.0;
  if (node_is_an_edge) continue;
  foreach face {
    Node other_node = (node(0)==*this)?node(1):node(0);
    δn=δ/node_area;
    Σδ+=1.0/(Cosθ*sdivs);
    ℵ matrix addValue(node_θ,this, node_θ,other_node, -δn);
  }
  Σδ*=δt/node_area;
  ℵ matrix addValue(node_θ,this, node_θ,this, 1.0+Σδ);
}
```

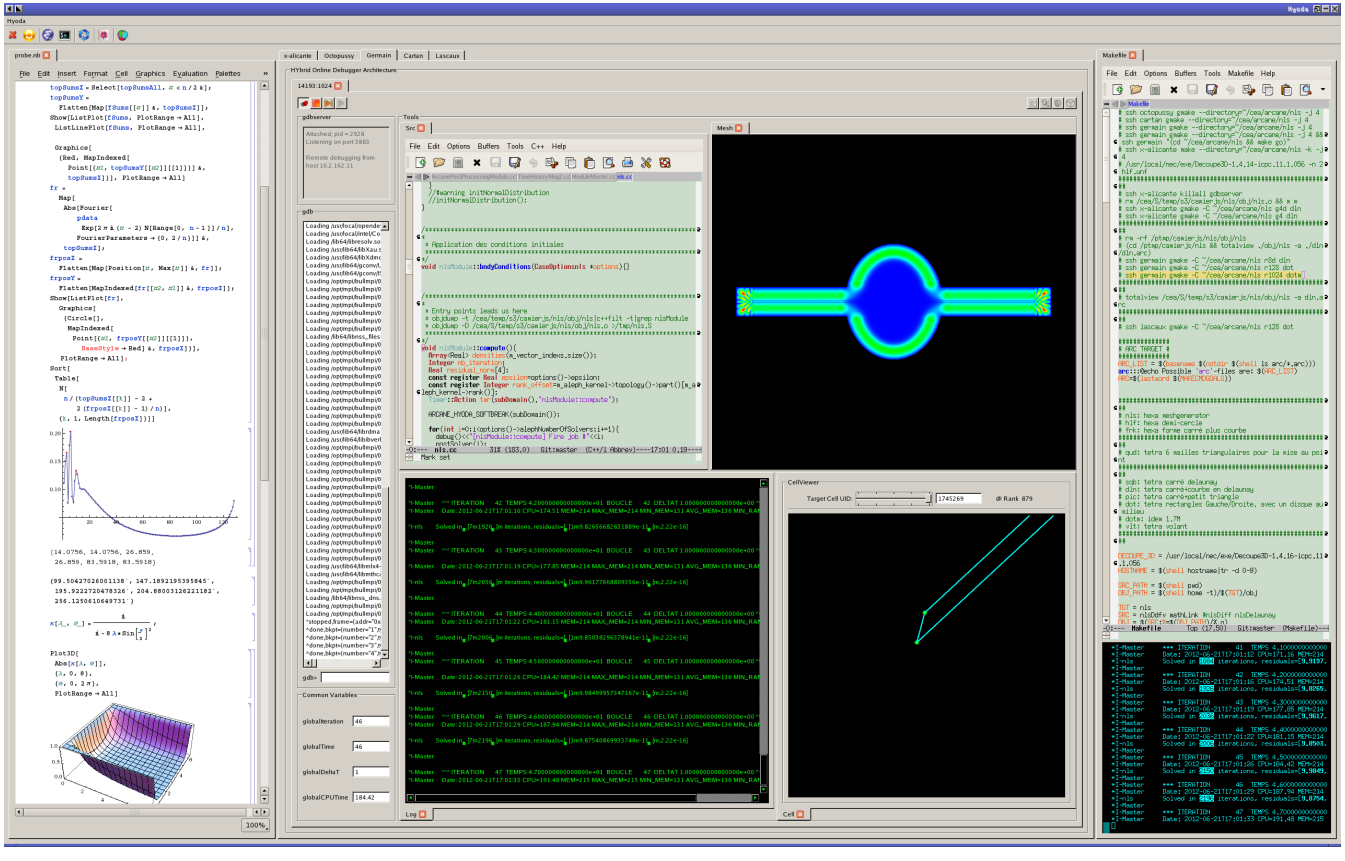
DE LA RECHERCHE À L'INDUSTRIE

cea

Outline

Hybrid Online Debugger Architecture

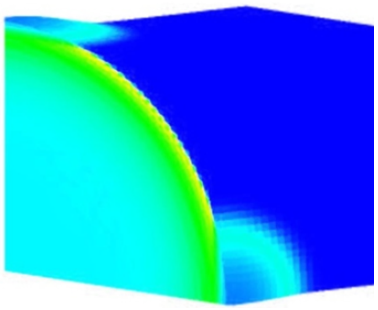
- HyODA does **not** replace a debugger at instructions level
- Debugging concepts are raised at ∇'s level of abstraction
  - Variables, mesh, cells, listing, profiling sumary in situ analysis
  - Matrix gathered view via third party tools (Mathematica, Matlab, SciLab)
  - Hybrid: 1 TCP socket hooked on a single core + MW's communication layer



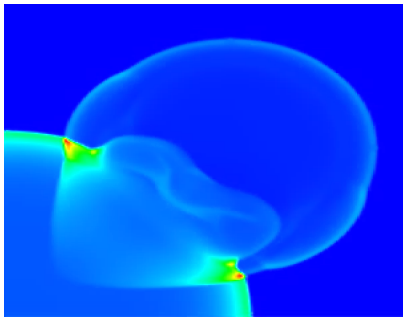
Outline

Main Proxy Applications ported to ∇

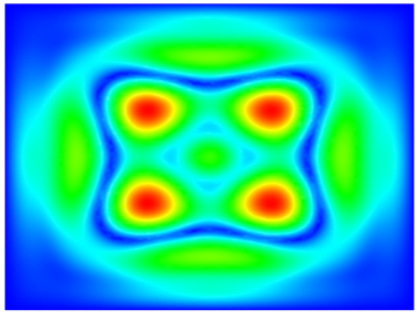
Numerical Schemes Types	Application	Lines of code
Explicites Unstructured	LULESH(LLNL)	1030
Explicites Cartesians	HYDRO(CEA/DAM)	757
Implicites	Schrödinger (CEA/DAM)	375
Monte-Carlo	MCTB(CEA/DAM)	828
Molecular dynamics	CoMD(LANL) MiniMD(SANDIA)	293 474



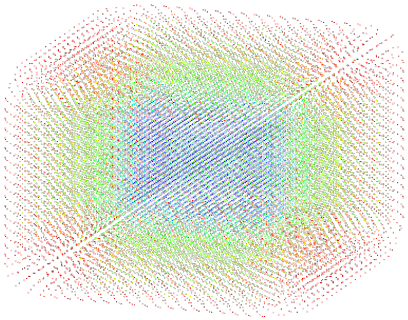
LULESH



HYDRO



Schrödinger



CoMD

LULESH - Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics

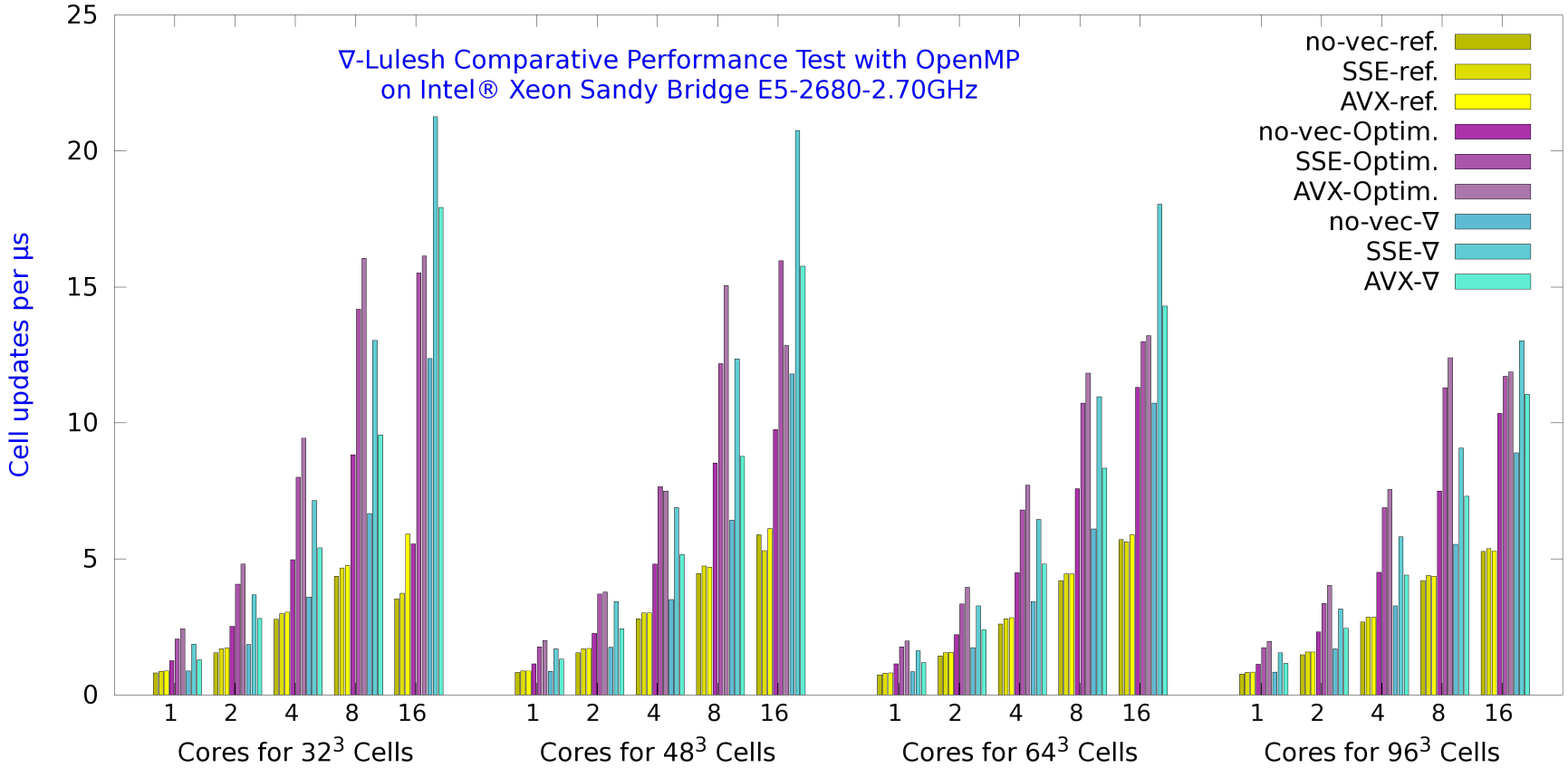
LULESH

- Proxy application in DOE co-design efforts for exascale
- Has been ported to a number of programming models
- Has been optimized for a number of advanced platforms

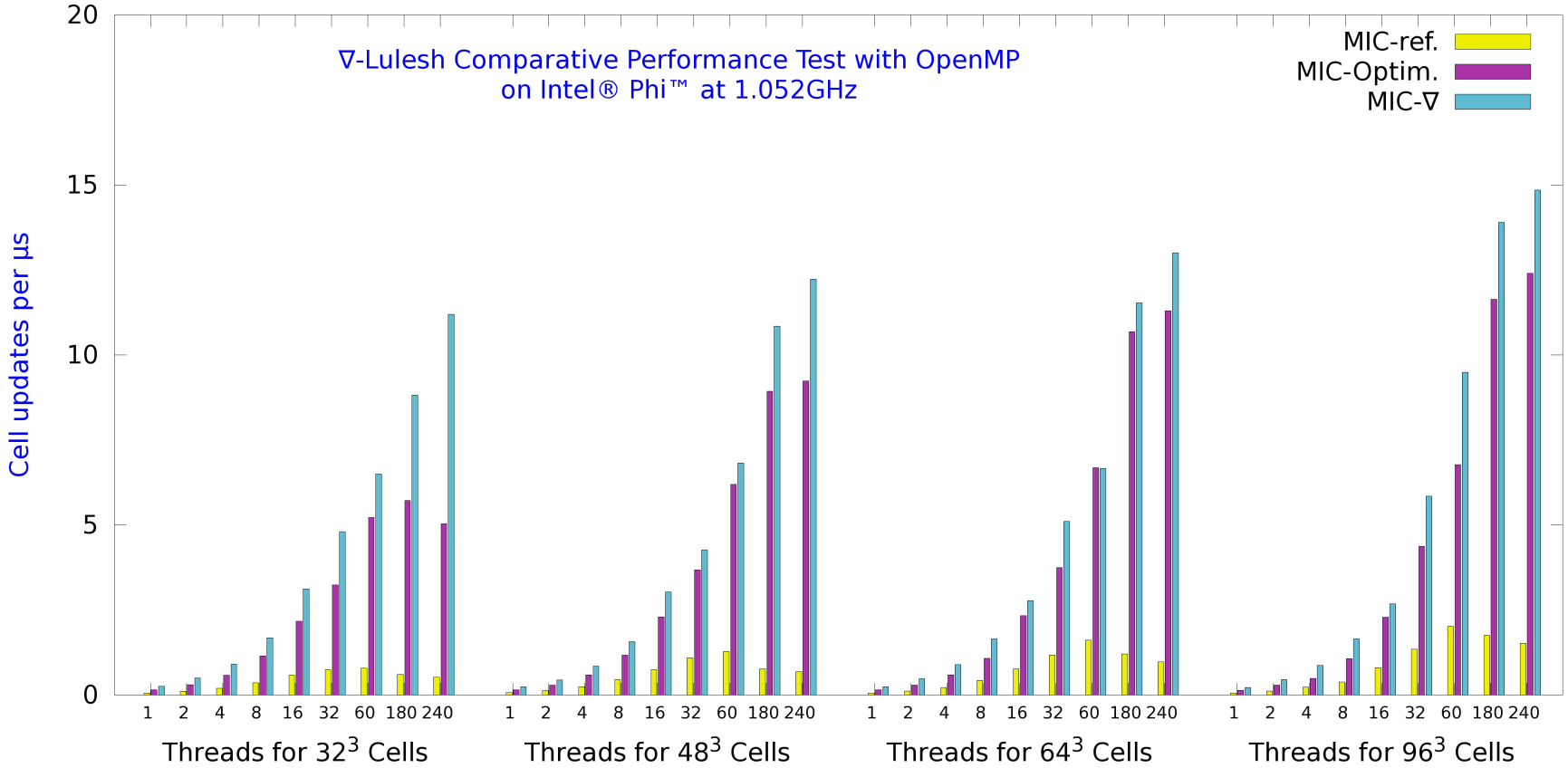
∇-LULESH

- LULESH serial reference has been correctly ported
- ∼3k sloc serial programing model vs ∼1k sloc w/o comments
- Differents backends are then available:
  - myMiddleware with Threads, MPI, MPI+Threads, MPC
  - Cuda: single-node code generation
  - Native + (OpenMP or Cilk+) with no-vec, SSE, AVX or MIC
  - Yours . . .
- Full **intrinsics** code generation
- Introduced **binary selection operator**
  - lhs = (test)?value; (no ternary ':' !)

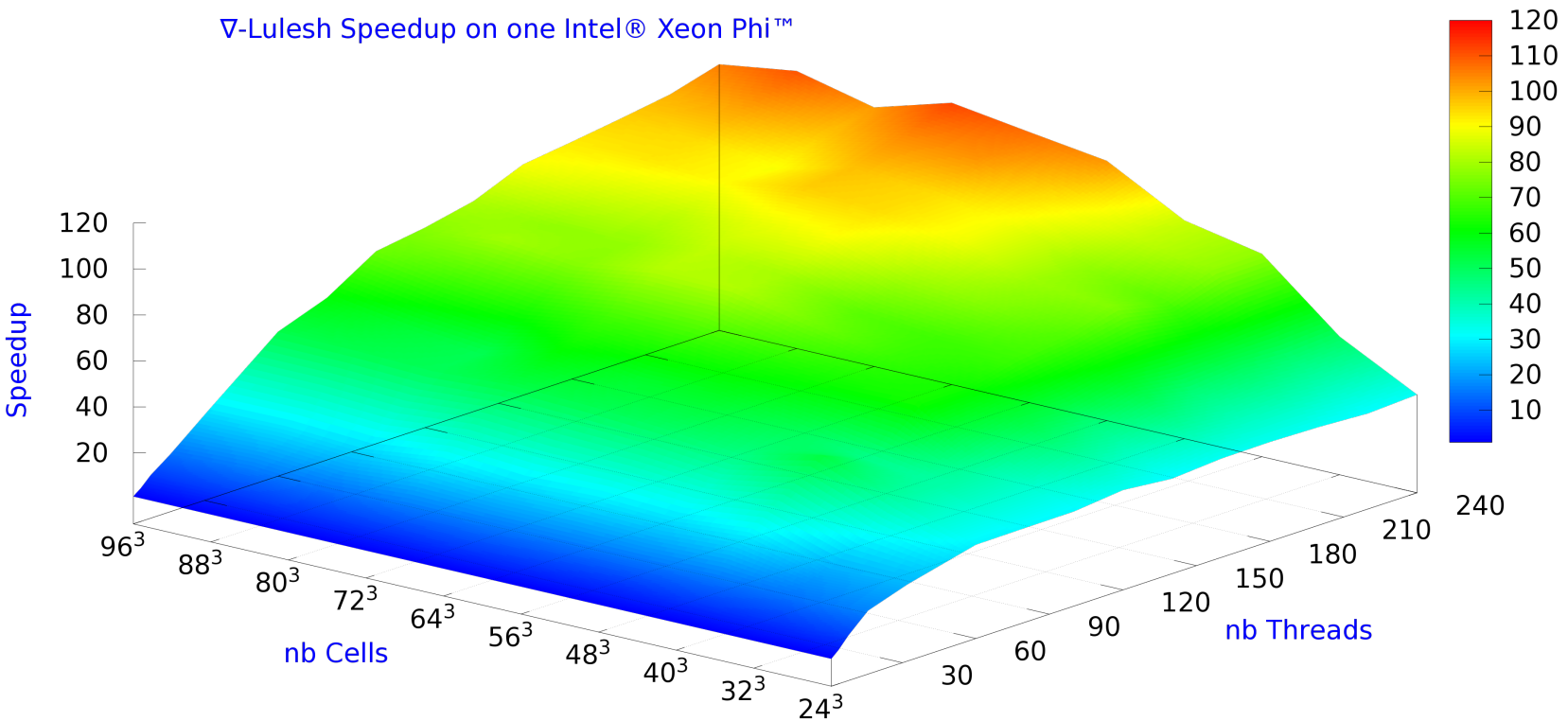
∇-LULESH Comparative Performance Test on Intel® Xeon Sandy Bridge™



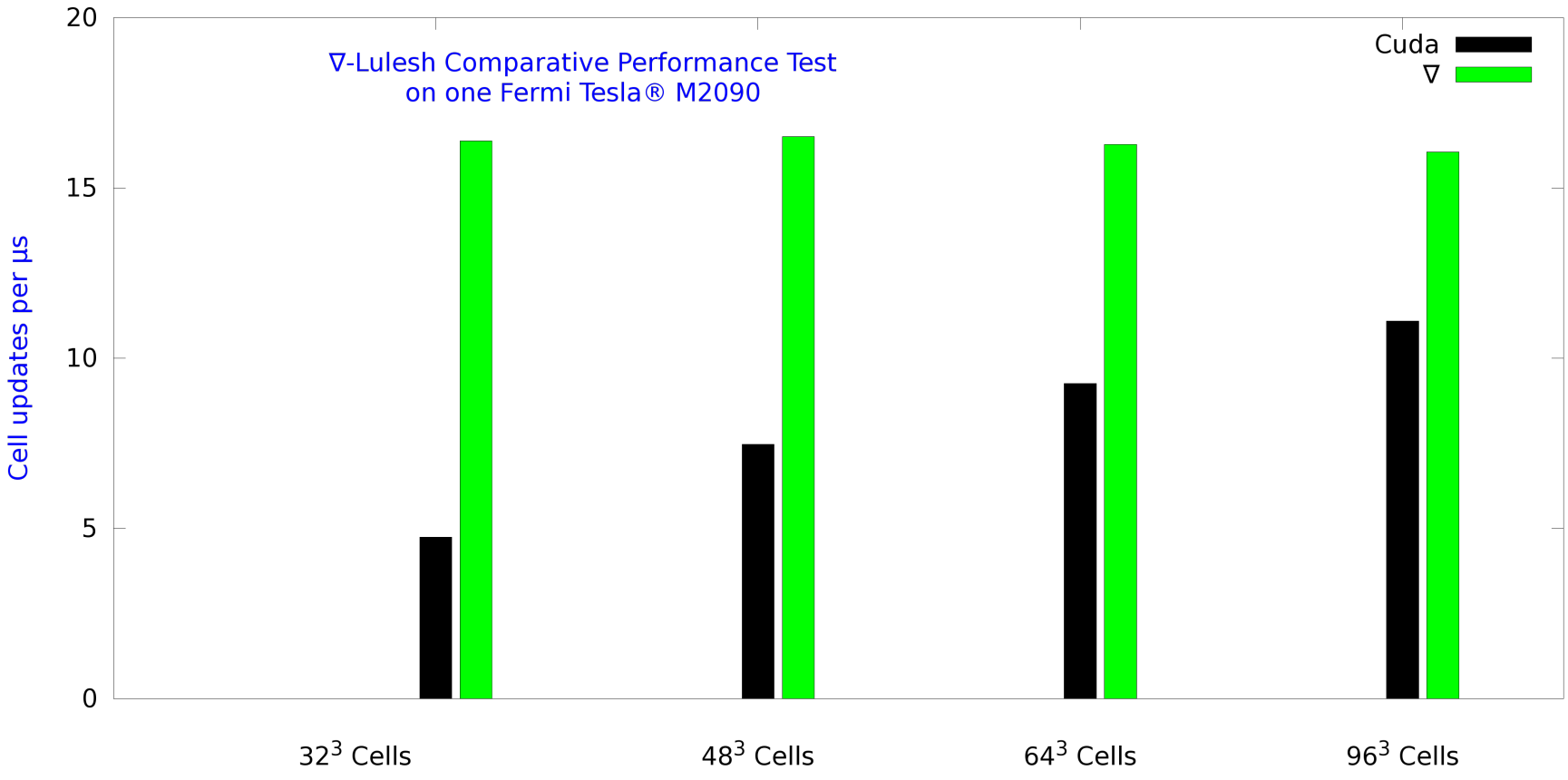
∇-LULESH Comparative Performance Test on Intel® XeonPhi™



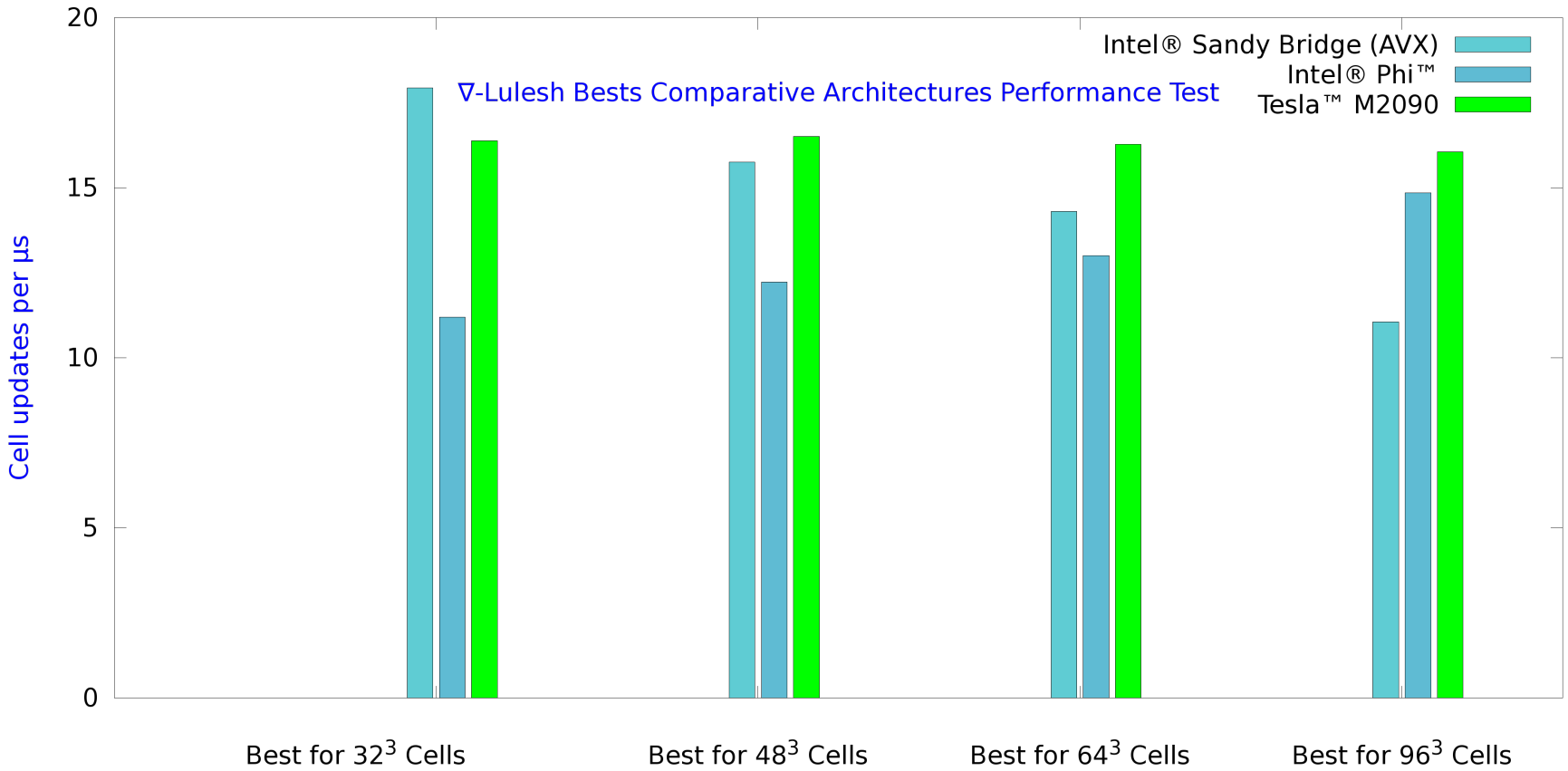
∇-LULESH Speedup on single Intel® XeonPhi™



∇-LULESH Comparative Performance Test on NVIDIA® Tesla™ Fermi M2090



∇-LULESH Best Comparative Architectures Performance Test



Outline

Conclusions and perspectives

Conclusions

- DSL proposition: simple, synthetic, performant (shown here on LULESH)
- Raises the level of abstraction
  - Improves productivity throughput
  - Provides portable scientific applications on different computer architectures
  - Introduces logical time and new statements to address more concurrency
- Ready to tackle the new computer science challenges ahead

Perspectives

- ∇ DSLanguage Specifications
- New code developments:
  - Backends: KokkosArray, OpenCL, PGAS, etc.
  - Techniques: for resilient OS/runtime systems
  - Proxy applications: ports and performance evaluations
- Open Source: www.sourceforge.org/?, www.nabla?.?

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 08 66 30 | F. +33 (0)1 69 08 66 30
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
Service numérique environnement et constantes