Improving Performance Portability and Software Productivity with the ∇ Numerical Programming Language

CAMIER Jean-Sylvain
CEA, DAM, DIF, F-91297 Arpajon, France

The ∇ Specific Language

Nabla (∇) is an **open-source** Domain Specific Language (**DSL**) introduced in [1] whose purpose is to translate numerical analysis algorithmic sources in order to generate optimized code for different runtimes and architectures.

Objectives & Roadmap

The objectives and the associated roadmap have been motivated since the beginning of the project with the goal to provide a programming model that would allow:

- **Performances**. The computer scientist should be able to instantiate efficiently the right programming model for different software and hardware stacks.
- **Portability**. The language should provide portable scientific applications across existing and fore-coming architectures.
- **Programmability**. The description of a numerical scheme should be simplified and attractive enough for tomorrow's software engineers.
- **Interoperability**. The source-to-source process should allow interaction and modularity with legacy codes.

The ∇ Toolchain

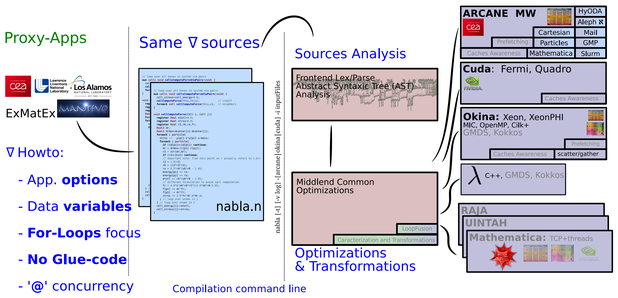


Figure 1: Three parts of the ∇ Toolchain: Sources Analysis (**Frontend**), Optimizations & Transformations (**Middlend**) and Generation Stages (**Backends**).

The backends hold the effective generation stages for different targets or architectures: **ARCANE** [2], **Multi-Processor Computing framework** (MPC) [3], **CUDA**, **OKINA** (fully-vectorized), **LAMBDA**, (WIP: **RAJA** & **UINTAH**).

Main Proxy Applications Ported to ∇

Numerical Methods	Application	# of ∇ lines
Explicit in time, Unstructured	LULESH (**1.0**)(LLNL)	1030
Explicit in time, Structured	HYDRO(CEA)	757
Implicit in time, Unstructured	M-NL-DDFV (CEA/DAM) Schrödinger (CEA/DAM)	2304 375
Monte-Carlo	MCTB(CEA/DAM)	828
Structured Mesh Molecular Dynamics	CoMD(LANL) MiniMD(SNL)	293 474
SPH	SPH(CEA/DAM)	2500

Overview of the ∇ DSL

The ∇ language allows the conception of multi-physics applications, according to a logical time-triggered approach. Nabla embeds the C language and follows a source-to-source approach. The method is based on different concepts: no central *main* function, a multi-tasks based parallelism model and a hierarchical logical time-triggered scheduling.

```
∀ cells void geomComputeQuadSurface(void)
  out (cell cell_A) @ -10.0 if (option_quads){
  ℝ³ fst_edge = coord[2]-coord[0];
  ℝ³ 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;
  ∀ cell node_A+=cell_A/nbNode;
}
```

```
∀ cells void computeStdFluxΣ(void)
  in (cell p,u,ū,AQs,CQs)
  out (cell mfΣ, EfΣ) @ -∞,-1.0,π²{
  ∀ node{
    const ℝ³ Δu = u-ū;
    const ℝ³ FQs = AQs⊗Δu + p*CQs;
    mfΣ -= FQs;
    EfΣ -= FQs⊙ū;
  }
}
```

Hierarchical Logical Time

Table 1: ∇ Logical Time Diagrams: b is the totally-ordered time-diagram from a typical mini-application ported to ∇ with consecutive *for-loops*; c is the diagram of a better partially-ordered numerical scheme.

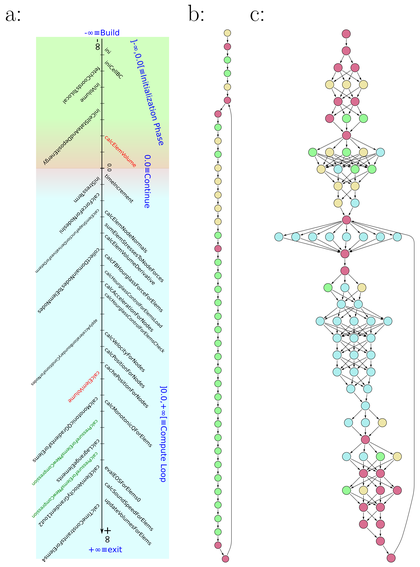
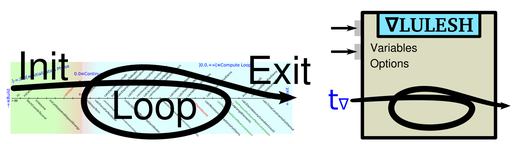


Table 2: ∇ LULESH Logical Timeline of ∇ LULESH (≠ Point Of Views)



∇-Lulesh Results on Xeon-SNB & Haswell

Each figure presents cells-updates-per-μs for different runs:

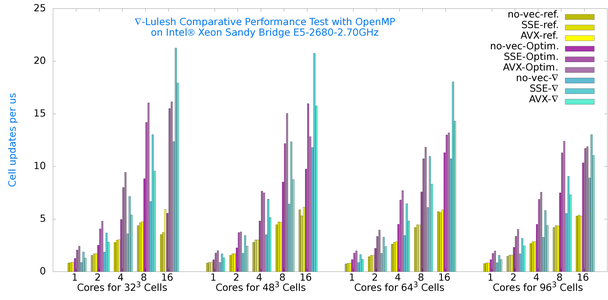


Figure 2: Reference (ref.), Optimized (Optim.) and ∇ Lulesh Performances Tests on Intel Xeon-SNB with the C/C++ Standalone OKINA+OpenMP Backend and no-vec., SSE or AVX Intrinsics. Higher is better.

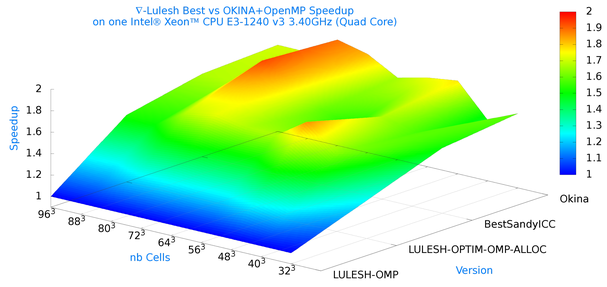


Figure 3: ∇-Lulesh Speedups on a quad core Intel Xeon Haswell

∇-Lulesh Speedup on XeonPHI (KNC)

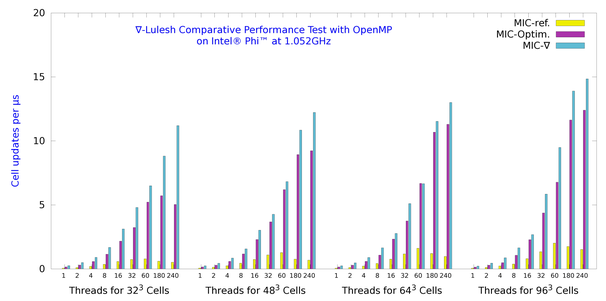


Figure 4: Reference (ref.), Optimized (Optim.) and ∇ Lulesh Performances Tests on Intel Xeon PHI with the C/C++ Standalone OKINA+OpenMP Backend and AVXMIC Intrinsics

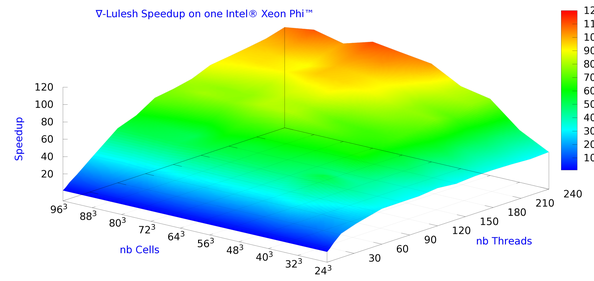


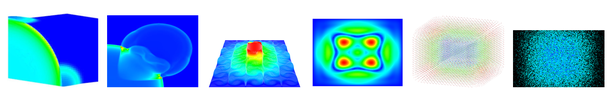
Figure 5: ∇-Lulesh Speedups on Intel Xeon PHI:

Discussion and Future Work

⇒ These results emphasize the **opportunity for DSL**!
⇒ Doing so opens up a potential path forward for for **enhanced expressivity and performance**.
⇒ **∇ raises the loop-level's abstractions**, allowing to be prepared to address future systems.
⇒ There is **no need to choose today** the best programming model for **tomorrow's architectures**.

www.nabla-lang.org

∇ is **open-source**, ruled by the French CeCILL license, which is a free software license, explicitly **compatible with the GNU GPL.**



Bibliography

[1] JS. Camier, ∇-*Nabla: A Numerical-Analysis Specific Language for Exascale Scientific Applications*, SIAM PP14, www.nabla-lang.org, 2014, www.nabla-lang.org.

[2] Gilles Grospellier and Benoit Lelandais, *The arcane development framework*, Proceedings of the 8th Workshop on Parallel/High-Performance Object-Oriented Scientific Computing (New York, NY, USA), POOSC '09, ACM, 2009, pp. 4:1–4:11.

[3] Marc Pérache, Hervé Jourdren, and Raymond Namyst, *MPC: A Unified Parallel Runtime for Clusters of NUMA Machines*, Proceedings of the 14th International Euro-Par Conference on Parallel Processing, Euro-Par'08, 2008.