The Pochoir Stencil Compiler

Status Update

Story

- Stencils are prevailing
- Conventional numerical library focus on optimizing individual computation operator
- Highly cache-efficient stencil algorithm is known yet hard to write from case to case.
- How to automate the optimization of a family of computation (such as stencil) in one framework is yet open question.
  - Library?
  - Compiler's pragma?
  - Autotuner?
  - DSL?
  - EDSL? (Pochoir's approach)

Summary

- *What we have done:*
  - Simple, concise, declarative, and easily verifiable DSL embedded in C++, with Intel Cilk Plus extension.
  - Arbitrary shaped, arbitrary depth stencil on arbitrary d-dimensional space-time grid, with complex boundary condition.
- *What we plan to do:*
  - Further improve the performance
  - Multiple inhomogeneous kernels (possibly overlapping)
    - Macroscopic inhomogeneity
    - Microscopic inhomogeneity
  - Generalized dependency
    - From orthogonal grid to general graph
  - JIT compiler for stencil
  - Platform portability

How Intel Can Help (market plan)

- Embed Pochoir technology in the ICC compiler
  - Optimizing techniques for a family of computation
  - Bind EDSL with compiler and runtime system
- Extending Pochoir technology to new architectures such as GPU, MIC, etc.
- JIT interface to ICC (call icc as a standalone library)
- Collaborating with Intel Exascale Computing lab on complex stencil kernels
- Help with Intel Cilk Plus and related tools, such as cilkprof
- Incorporate ISAT with heuristic autotuning
- Research support for Yuan Tang
- Open source like FFTW?

Roadmap

- Release 0.5 (Feb. 2011)
- Release 1.0 (Aug. 2012)
- Release 2.0 (TBD)

Release 0.5

- Released in Feb. 2011
- Published in SPAA'11 & HotPar'11
- Simple, concise, declarative, and easily verifiable DSL embedded in C++, with Intel Cilk Plus extension.
- Arbitrary shaped, arbitrary depth stencil on arbitrary d-dimensional space-time grid, with complex boundary condition.

Current User List

- Oscar Barenys, Univ. Politechnica of Catalonia, Spain.
- Volker Strumpen, Johannes Kepler University, Austria.
- Nicolas Pinto, MIT/Harvard
- Nicolas Vasilache, Reservoir Lab.
- Patrick S. McCormick, Los Alamos National Lab.
- Mohammed Shaheen, Max Planck Institut Informatik, Germany
- Wim Vanroose, Universiteit Antwerpen, Belgium.
- Tom Henretty, Ohio State Univ.
- Protonu Basu, Univ. of Utah.
- Shoaib Kamil, Berkeley.
- Hal Finkel, Argonne National Lab.
- Matthias Christen, Klingelbergstrass, Basel, Switzerland.
- Vinayaka Bandishti, Indian Institute of Science, Bangalore, India.
- Hans Vandierendonck, Ghent University, Belguim.

Benchmark Suite

- Physics
  - Heat equation
  - Wave equation
  - Maxwell's equation
  - Lattice Boltzmann Method
- Computational Biology
  - RNA secondary structure prediction
  - Pairwise sequence alignment
- Computational Finance
  - American Put Stock Option Pricing
- Mechanical Engineering
  - Compressible Euler Flow
- Others
  - Conway's Game of Life
  - …

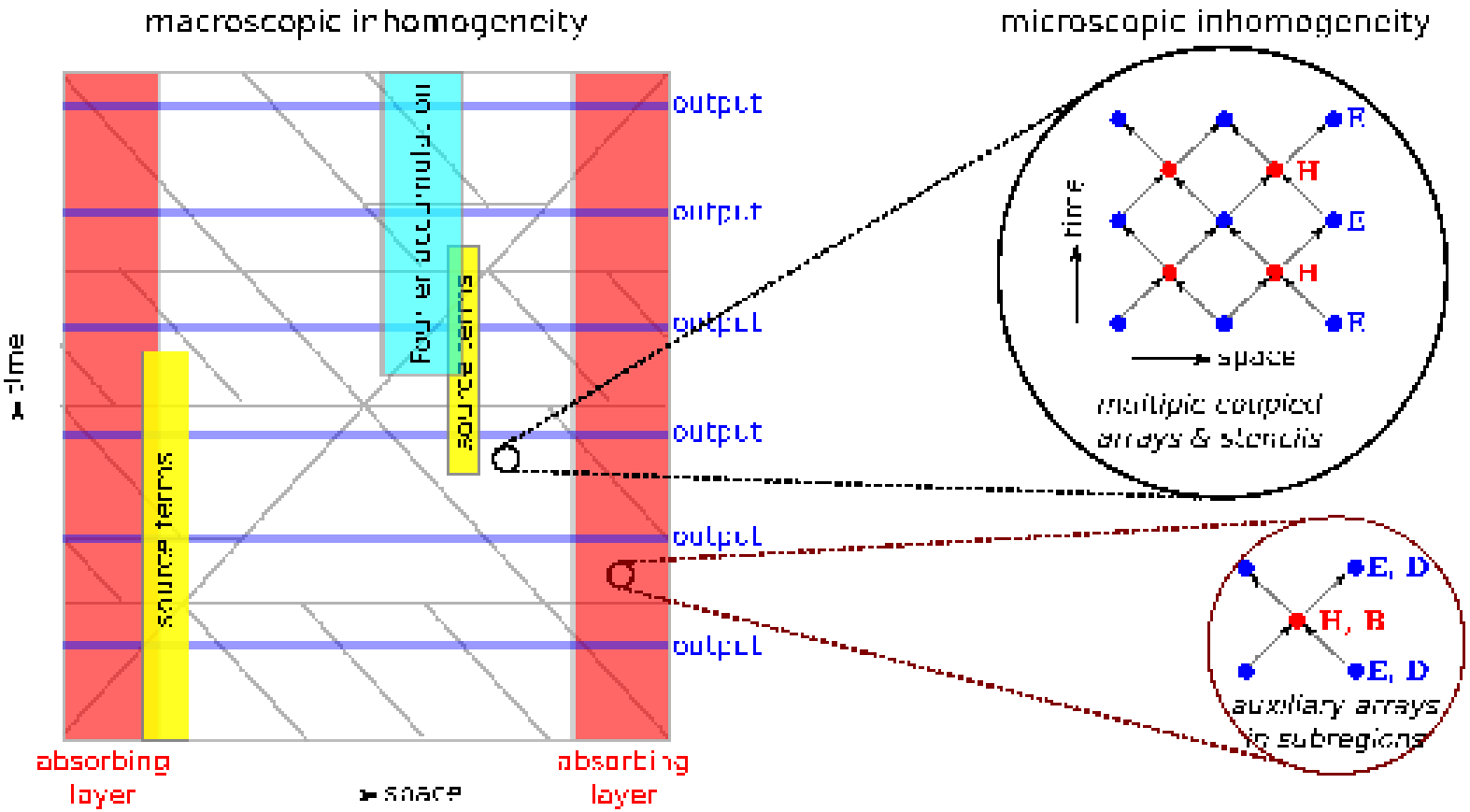
Release 1.0

- Aug. 2012
- Bug Fixes
- User's feedback
- Variadic Template Support
  - Even Simpler user interface

Release 2.0

- TBD
- Inhomogeneity
  - Macroscopic Inhomogeneity
  - Microscopic Inhomogeneity
- Generalized Dependency
  - Both PUSH and PULL
  - Slope 0 cut
    - from orthogonal grid to general graph
- Beyond Multicore
  - Distributed memory clusters
  - MIC
  - GPU
  - FPGA
  - Exascale …

Inhomogeneity



Inhomogeneity

- How to specify the stencil
  - Macroscopic specification:
    - Pochoir.Register_Tile_Kernel(guard, kernel);
    - Pochoir.Register_Tile_Kernel(guard, tile);
  - Tile for microscopic inhomogeneity
- How to execute the specification
  - Define an “Inhomogeneity” metric
    - partial order, norm, arithmetic op, etc.
  - Jitting the kernels
    - Preprocessing stage versus runtime ?
    - Range Bit-wise Operation Query
      - Meta-algorithm, triangular query, polygon query
    - Code clone selection puzzle
      - Prune to save code generation overhead
      - Minimize the # if conditionals in inner-most loops

JIT framework

- Preprocess
  - Get the Inhomogeneity metric
  - Prepare data structures for runtime query
- Generate Only necessary (possibly pruned) kernels
  - genstencils + icc
  - Disk as medium (what if everything in memory?)
  - Can we call icc as a standalone library?
- Generate the kernels at preprocessing stage versus at runtime.

Summary

- *What we have done:*
  - Simple, concise, declarative, and easily verifiable DSL embedded in C++, with Intel Cilk Plus extension.
  - Arbitrary shaped, arbitrary depth stencil on arbitrary d-dimensional space-time grid, with complex boundary condition.
- *What we plan to do:*
  - Further improve the performance
  - Multiple inhomogeneous kernels (possibly overlapping)
    - Macroscopic inhomogeneity
    - Microscopic inhomogeneity
  - Generalized dependency
    - From orthogonal grid to general graph
  - JIT compiler for stencil
  - Platform portability

How Intel Can Help (market plan)

- Embed Pochoir technology in the ICC compiler
  - Optimizing techniques for a family of computation
  - Bind EDSL with compiler and runtime system
- Extending Pochoir technology to new architectures such as GPU, MIC
- JIT interface to ICC (call icc as a standalone library)
- Collaborating with Intel Exascale Computing lab on complex stencil kernels
- Help with Intel Cilk Plus and related tools, such as cilkprof
- Incorporate ISAT with heuristic autotuning
- Research support for Yuan Tang
- Open source like FFTW?