EECS
Electrical Engineering and Computer Sciences

Productive Auto-tuning of Stencil Grid Kernels

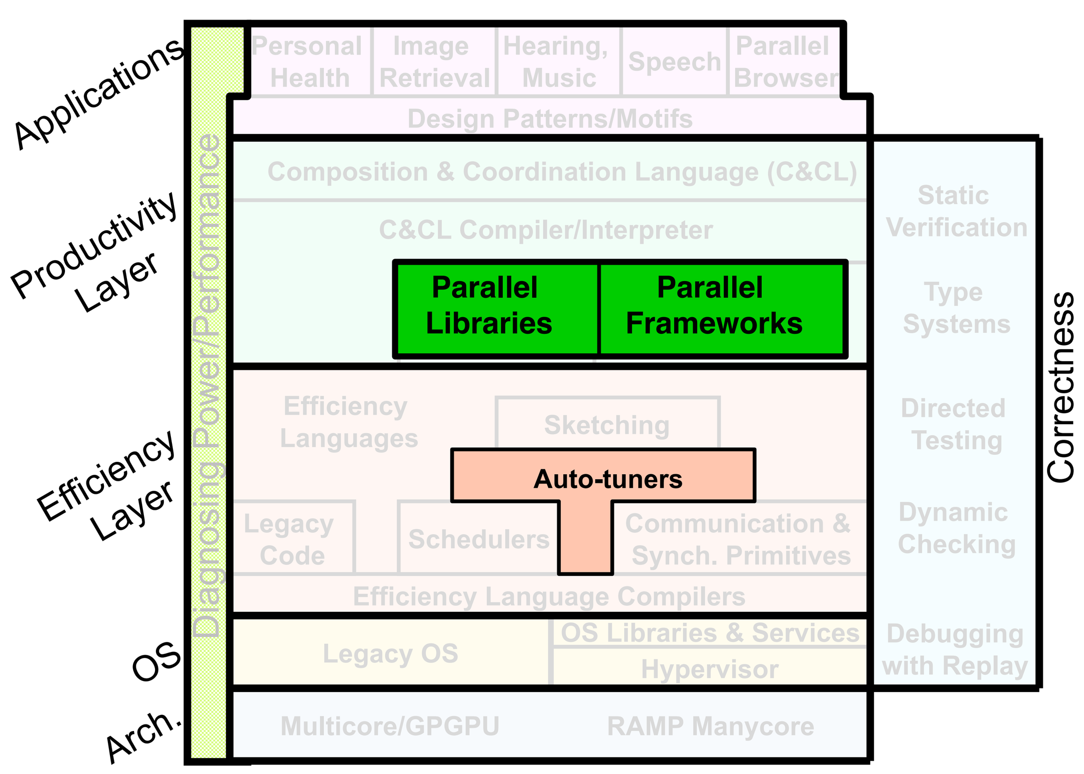
S. Kamil, C. Chan, K. Datta, S. Williams, J. Shalf, L. Oliker, K. Yelick

Berkeley Par Lab

PARALLEL COMPUTING LABORATORY

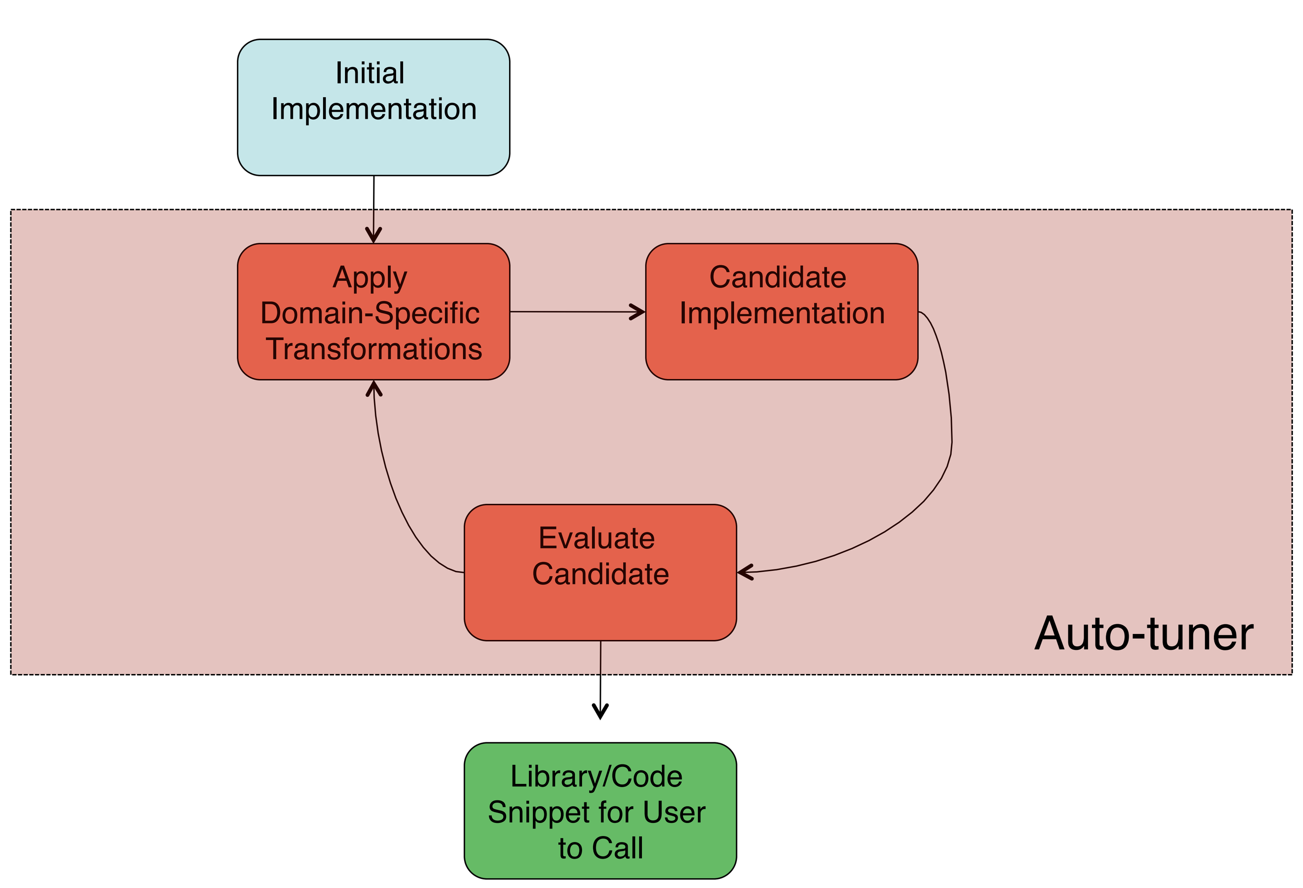
Where this fits in the ParLab

- ParLab is the interdisciplinary group at UC Berkeley investigating the impact of many-core parallelism
- Software stack is divided into Efficiency and Productivity Layers
- This work crosses between both layers, providing framework/library support for arbitrary structured grid kernels



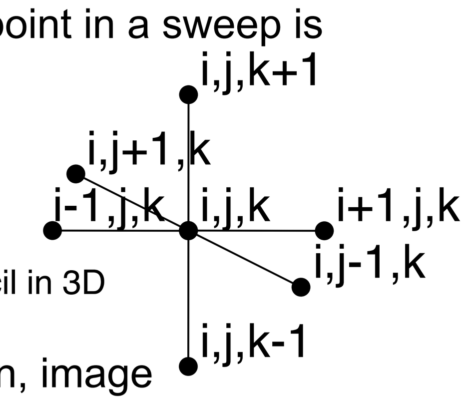
Productive Auto-Tuning

- Auto-tuning is an automated, general system for performance portability across architectures, using domain-specific knowledge
- Traditional optimization speeds up **one** kernel on **one** platform
- Traditional Auto-tuning speeds up **one** kernel on **many** platforms
  - Strategy in many numerical libraries such as Atlas and OSKI
- Productive Auto-tuning: goal is to speed up **many** kernels on **many** platforms
  - Build tuner for a *class* of kernels
  - Use high-level knowledge of the motif to optimize specific instantiations



Structured Grid: Stencil Kernels

- Many computations on grids with regular structures can be represented as "sweeps" over the grid, where each point in a sweep is a arithmetic combination of the point's neighbors

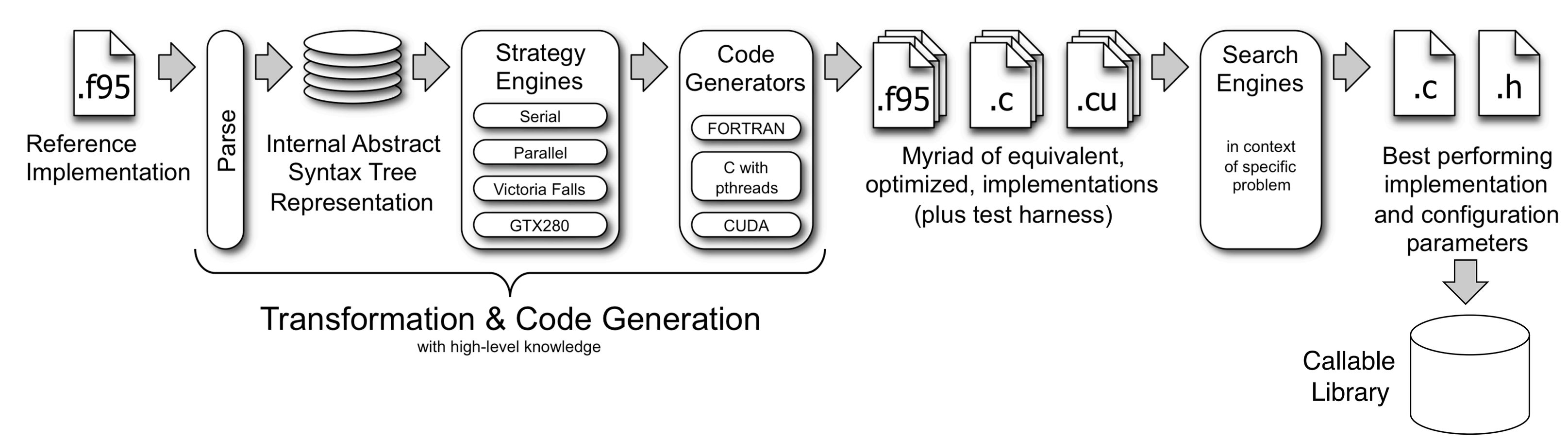


Example: 7 point stencil in 3D

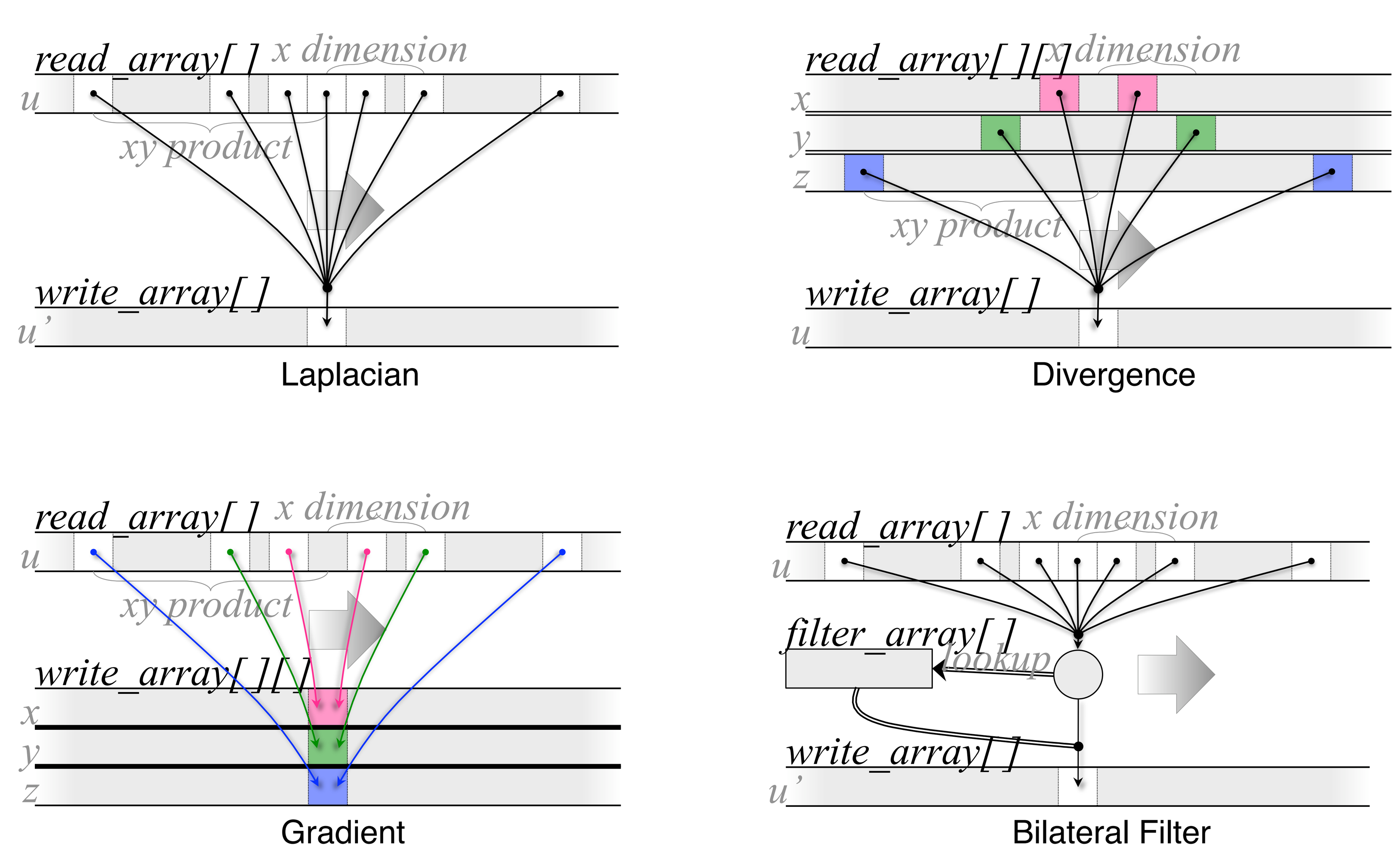
- Applications include PDE solvers, climate simulation, image processing/filtering

Auto-tuner Design

- Stencils described in a Domain-Specific Language (DSL)
  - Currently a simple subset of Fortran
- Auto-tuner takes annotated description of kernel and performs the following steps:
  1. Auto-tuner parses stencil code into an intermediate representation
  2. Representation is transformed into auto-parallelized version
  3. Backend strategy engines enumerate applicable parameter space and
     1. Generate timing stub
     2. Generate each version of code
     3. Iterate over versions & measure performance
     4. Choose optimal version
  4. Best-performing version is packaged into callable library

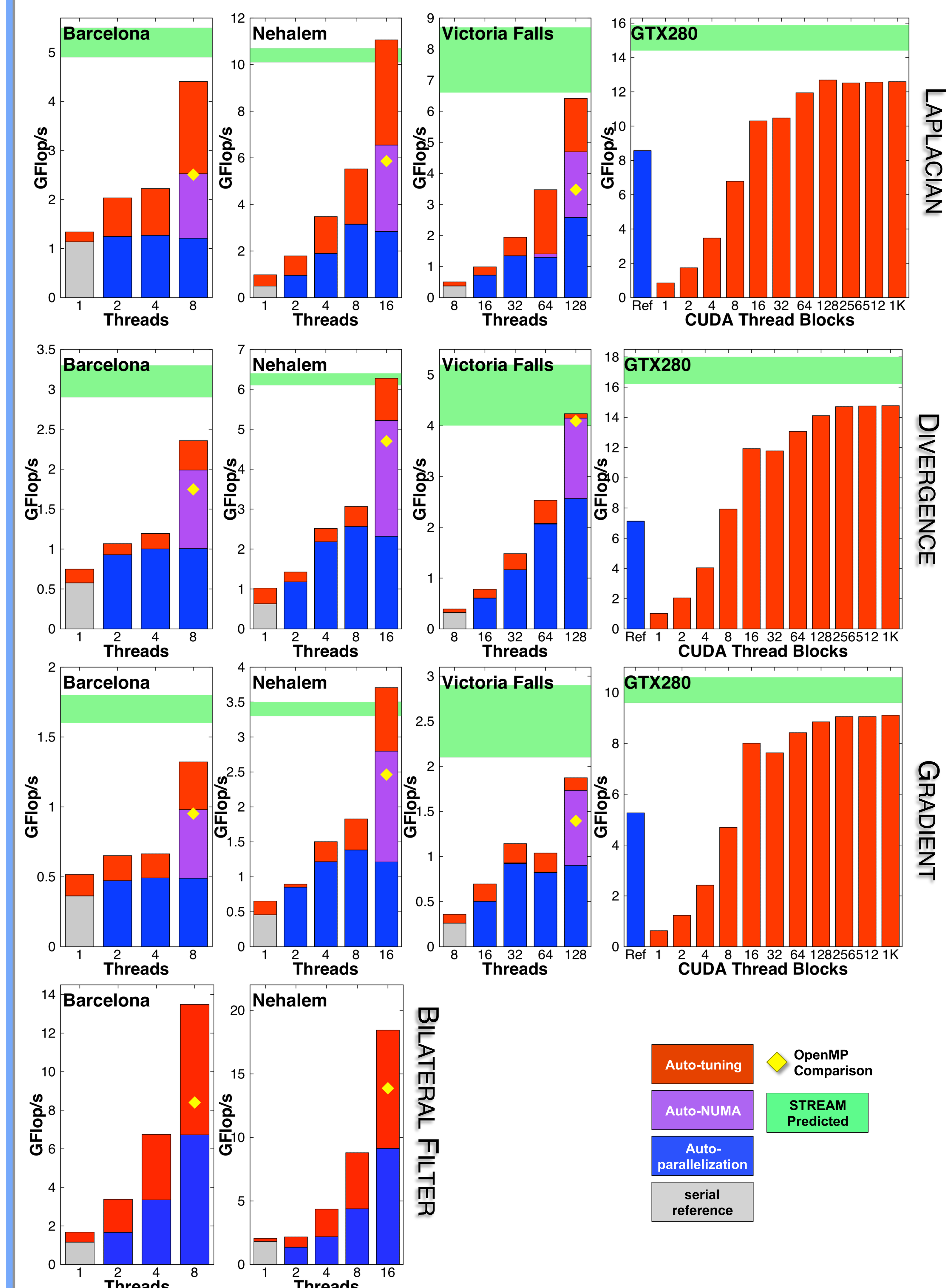


Example Stencil Kernels



- Original Fortran code uses multidimensional arrays
- Auto-tuned C code uses flat single-dimensional arrays
- Bilateral filter has *radius* parameter that changes stencil footprint

Performance Results



- Excellent performance gains
  - Up to 22x for memory-bound kernels
  - Near-perfect scaling for compute-bound bilateral filter kernel
  - Comparable to kernel-specific auto-tuning performance on multicore from previous work
- Productivity gains: Performance and Kernel portability
  - Laplacian, Divergence, and Gradient tuned fully-automatically
  - Required no changes to auto-tuning framework
  - Compared to auto-tuning each kernel individually: large portion of performance at a fraction of the effort
    - Hand-tuner for a single kernel requires weeks to months of effort!
- Lots of future work
  - Extend generality of supported kernels (and define the domain)
  - Improve CUDA/OpenCL tuner to incorporate better optimizations
  - Extend supported backends (Local store archs, etc.)
  - Friendly frontend: also support higher-level definitions of kernels