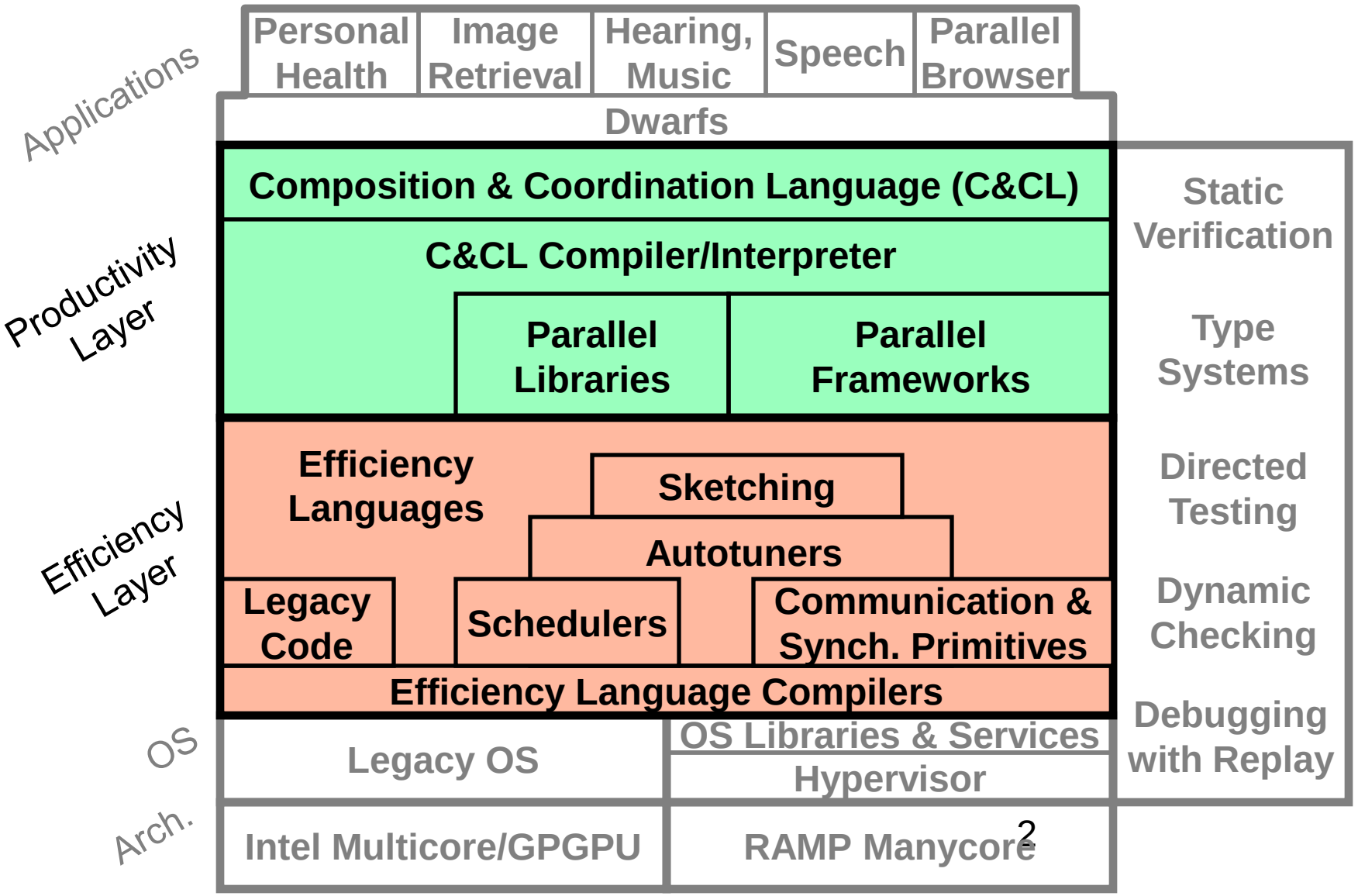
SEJITS: High Performance for Productivity Applications

Shoaib Kamil & many others

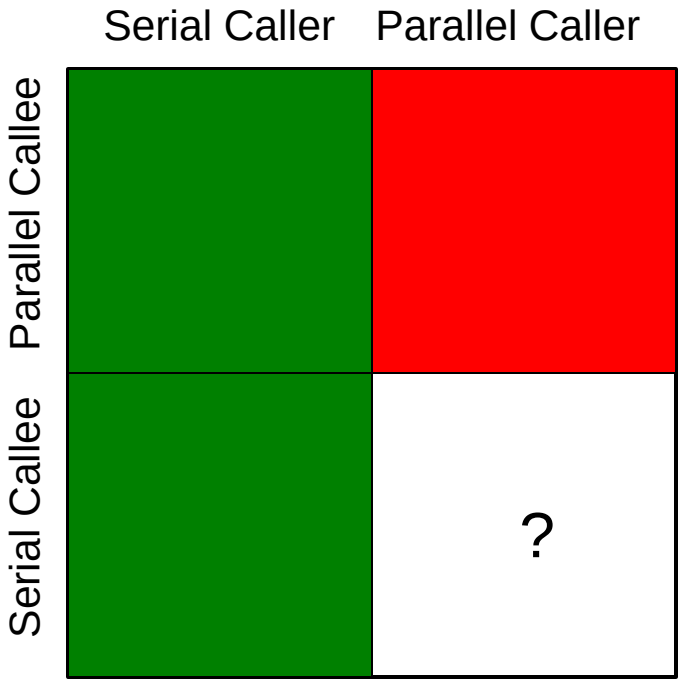
Parlab End of Project Celebration

May 30, 2013

Where Did We Start?



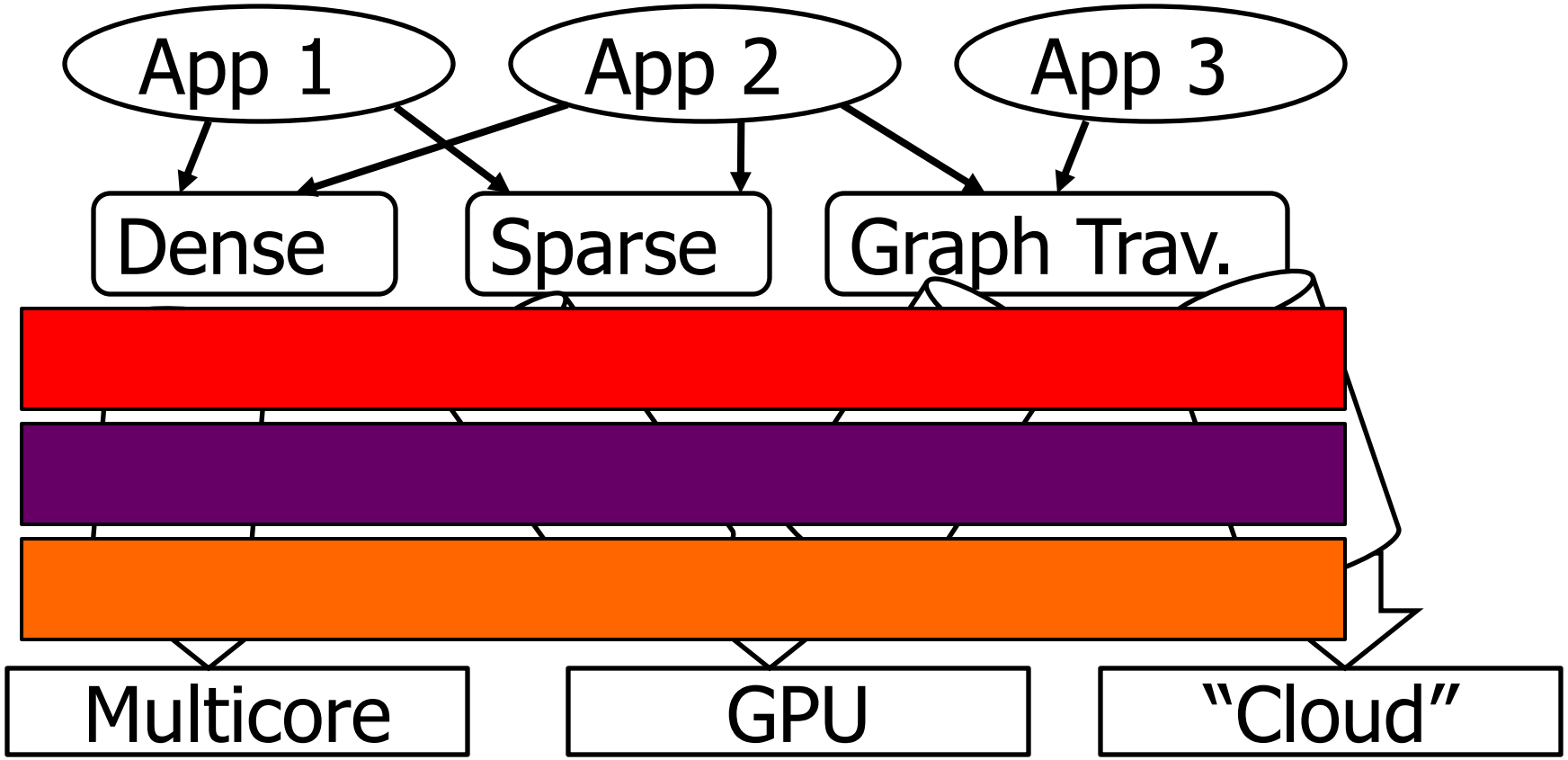
The Big Problems



- Knew how to coordinate serial caller with serial or parallel callee
  - How can we do the other two, in a way understandable to non-expert programmers?

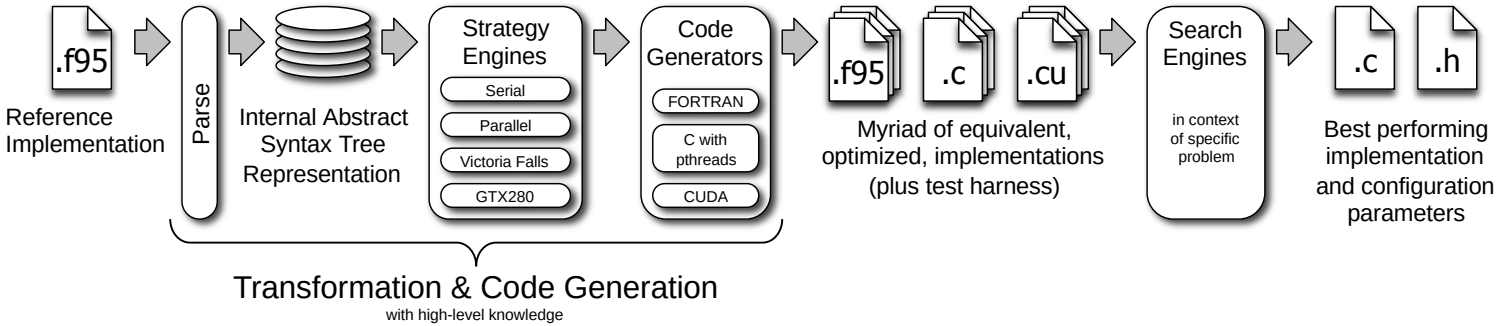
The Big Problems

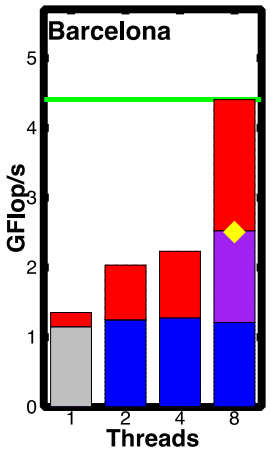
- Layering is good for engineering, bad for performance

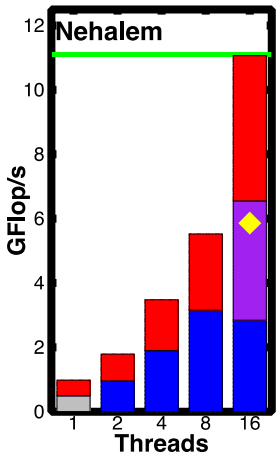


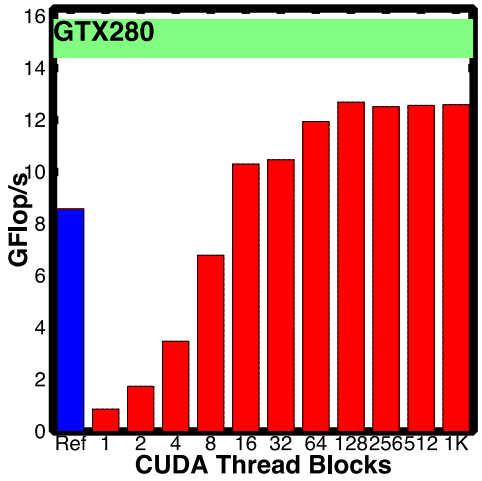
Inspiration: Auto-tuned DSL for Stencils

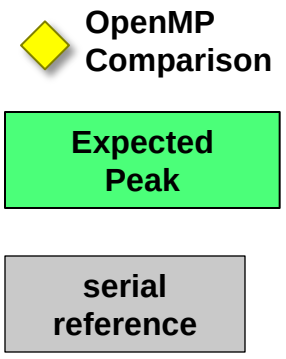
- Began with a large productivity-programmer-created codebase: new climate code
  - Use code transformation to go from productive to efficient code











[IPDPS10]

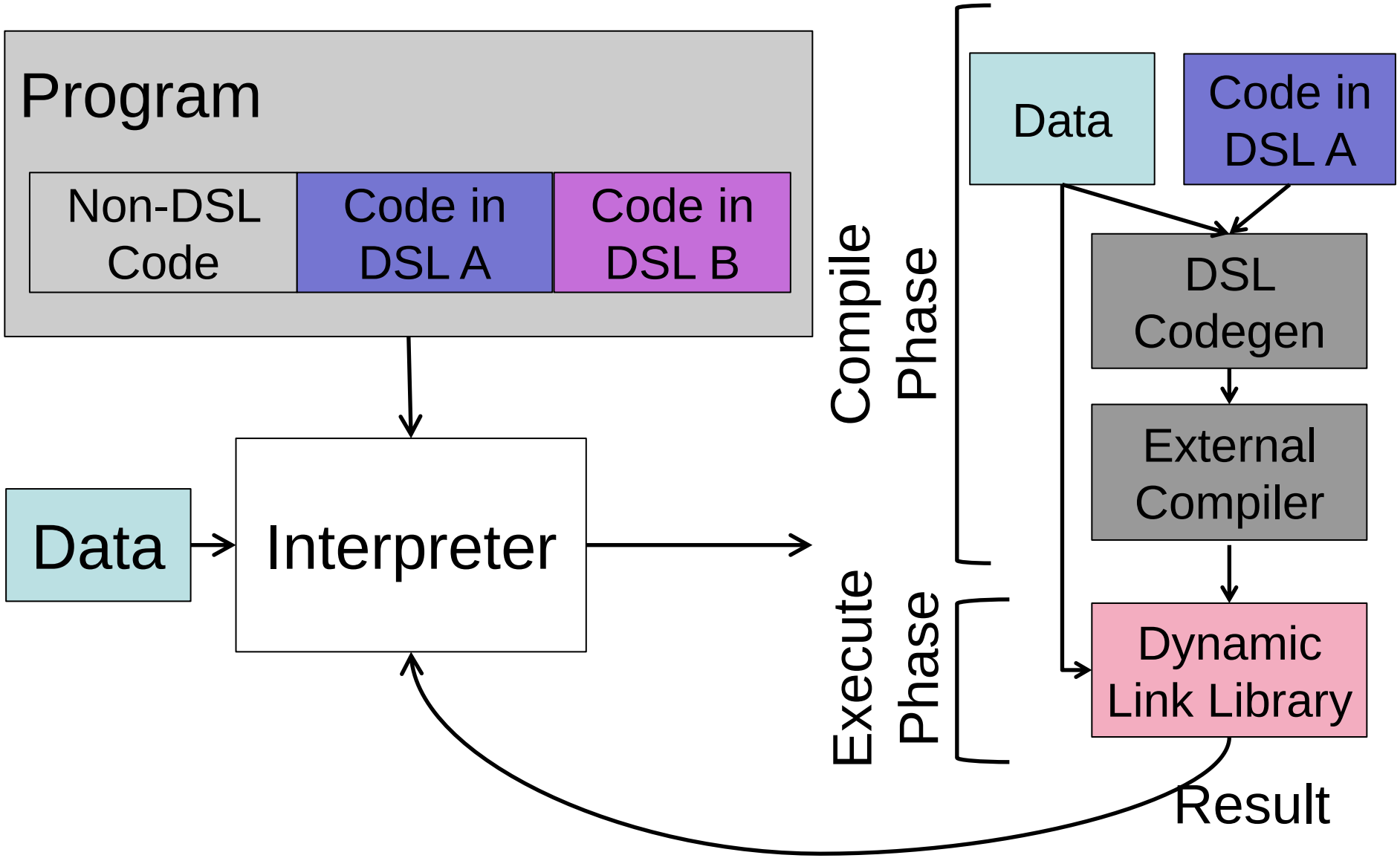
Inspiration: ActiveRecord


```
Product.find_by_price(95)

Product.all.where("quantity > ?", 0).where(:color => "red")
```

- Ruby on Rails ORM
- Complex queries generated from simple chains of method invocations
- Productivity programmer doesn't need to know SQL

SEJITS In A Nutshell



Case Study: Sepya

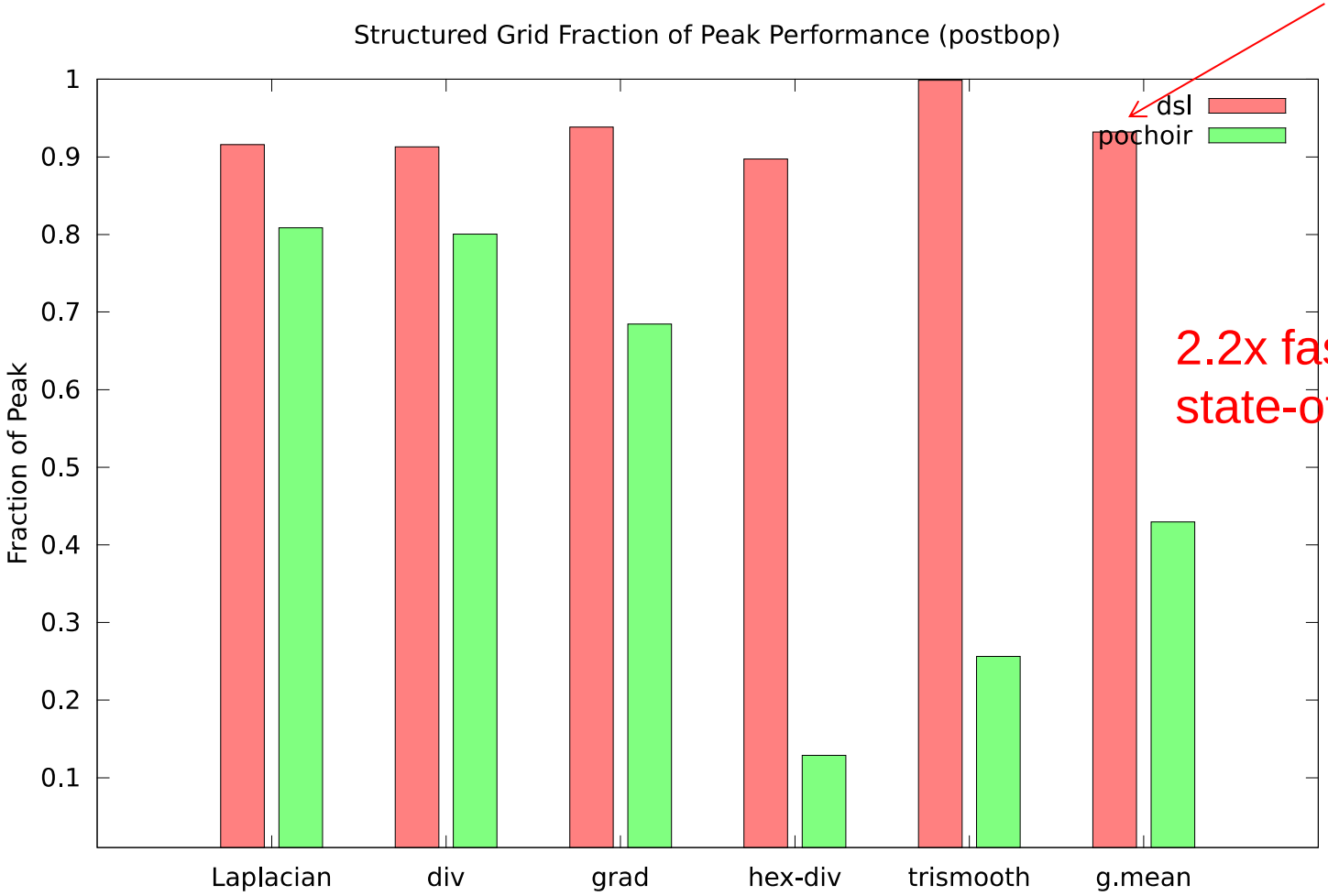
- Stencils Embedded in Python with Auto-tuning
  - Simple, declarative source for fast stencil computations on multicore/multisocket CPUs

```
class My1DKernel(StencilKernel):
  def __init__(self):
    super(MyKernel, self).__init__()
    # set the neighbors set 1 as the point
    # before and the point after
    self.set_neighbors(1, [[1],[-1]])

  # the actual kernel function
  def kernel(self, out_grid, in_grid):
    for x in out_grid.interior_points():
      for y in self.neighbors(in_grid, x, 1):
        out_grid[x] += 0.5 * in_grid[y]
```

Case Study: Sepya

93% of Roofline Peak



2.2x faster than state-of-the-art

Asp is SEJITS for Python

- Library for building auto-tuned DSELs
  - Introspection of Python
  - Code generation via templates or tree transformations
  - Auto-tuning support
  - Data structure interoperability
  - Automatic compiler detection & demand-driven compilation with caching

- Ongoing development & support
  - http://sejits.org
  - Screencasts available

[SciPy11, PPoPP12]

Copperhead

- Data parallel compiler for Python subset

```
from copperhead import *
import numpy as np

@cu
def axpy(a, x, y):
  return [a * xi + yi for xi, yi in zip(x,
y)]

x = np.arange(100, dtype=np.float64)
y = np.arange(100, dtype=np.float64)

with places.gpu0:
  gpu = axpy(2.0, x, y)

with places.openmp:
  cpu = axpy(2.0, x, y)
```

- Bryan Catanzaro et al (NVIDIA Research)
- http://copperhead.github.io

Ongoing Work

- Knowledge Discovery Toolbox (John Gilbert, Adam Lugowski, and Aydin Buluc, UCSB & LBNL) [IPDPS13]
- PyCASP: Python-based Content Analysis Using Specialization (Katya Gonina) [ASRU11]
- Bag of Little Bootstraps (Peter Birsinger, Richard Xia, et al) [SciPy12]
- Communication-Avoiding Recursive Algorithms (David Eliahu, Omer Spillinger, Oded Schwartz, Ben Lipshitz, Jim Demmel) [IPDPS13]
- Communication-Avoiding Solvers (Jeffrey Morlan) [iWAPT12]
- Three Fingered Jack (David Sheffield) [HotPar13]
- ASPIRE Project

SEJITS Participants

Bryan Catanzaro, Kurt Keutzer, Katya Gonina, Jeffrey Morlan, Armando Fox, Richard Xia, Henry Cook, Peter Birsinger, David Patterson, Katherine Yelick, James Demmel, Tim Mattson, Henry Gabb, David Sheffield, Michael Anderson, Juan Vargas, Jessica Miller, Scott Beamer, Omer Spillinger, David Eliahu, Aakash Prasad, Oded Schwartz, Ben Lipshitz, David Howard, Derrick Coetzee, Tayfun Elmas, Koushik Sen

Demo & Testimonial

- Demo: Derrick Coetzee showing a simple image processing kernel
- Testimonial: Tim Mattson, Intel