Asp Specializers

Shoaib Kamil
Armando Fox, Katherine Yelick,
Derrick Coetzee, Jeffrey Morlan,
Young Kim, David Johnson & many more

Par Lab/UPCRC Retreat, Summer 2011

EECS
Electrical Engineering and Computer Sciences

Outline

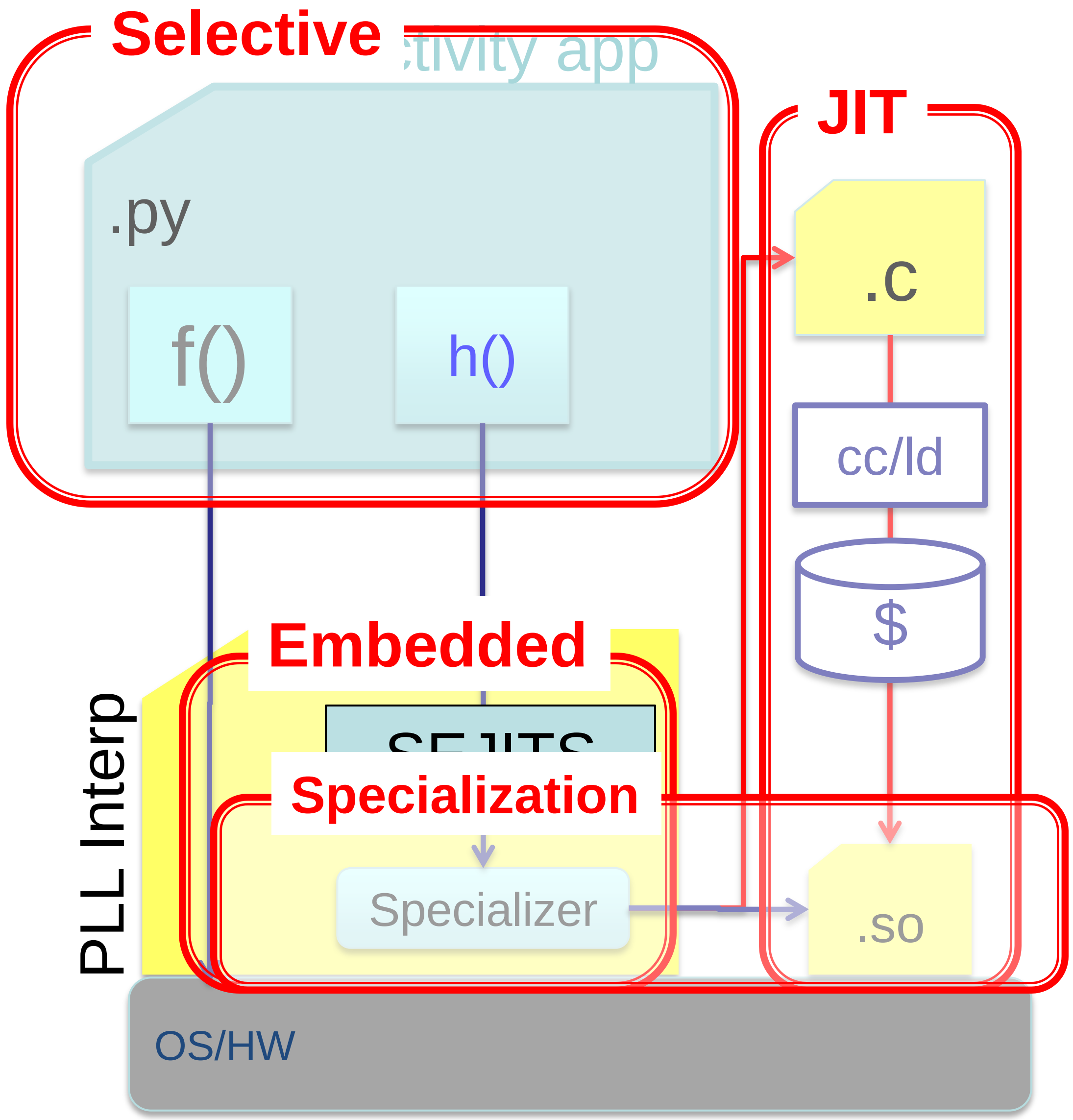
BERKELEY PAR LAB

- Review of SEJITS & Asp
- Asp Status & Highlights
- SEJITS & Separation of Concerns
- Specializer Structure
- Recent Results
- Getting Involved

EECS
Electrical Engineering and Computer Sciences

SEJITS Review

BERKELEY PAR LAB



Asp Review

- **A**sp is a **S**EJITS infrastructure for **P**ython
- Enables building specializers for Python
  - Specializers = domain-specific code translators + autotuners
  - Specializers expose an understandable, Pythonic interface for domain scientists
  - Behind the scenes, specializers use Abstract Syntax Tree manipulation and code templates to do translation

EECS
Electrical Engineering and Computer Sciences

Status and Highlights

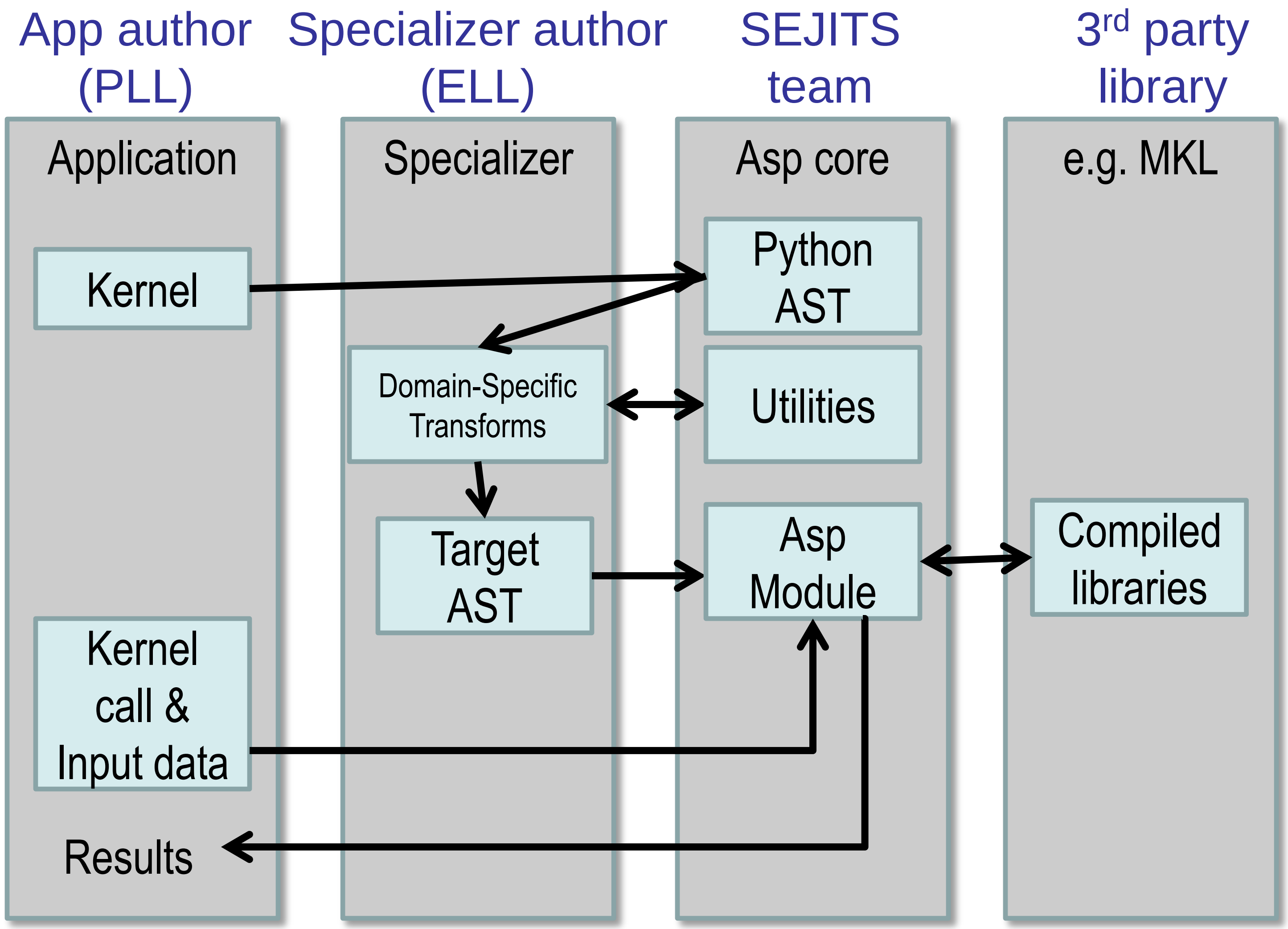
BERKELEY PAR LAB

- Infrastructure now enables building non-trivial specializers
- 3 specializers mature enough to have performance results, 2 integrated in driving apps
  - See my poster for Stencil
  - Jeffrey Morlan’s poster for Communication-Avoiding Matrix Powers
  - Katya & Henry’s poster (and talk) for Gaussian Mixture Modeling

Status and Highlights

- Begun applying ML techniques to recorded performance of auto-tuned/specialized code
  - Orianna Demasi's poster on Decision Trees for Stencil Tuning
- Developer Preview planned to coincide with SciPy 2011 Conference in July
  - We will be giving a talk about Asp at the conference

Asp: Who Does What?



Specializer Structure

- Templates vs. Abstract Syntax Tree manipulation
  - Templates useful for many parts of computation
  - Some specializers only use templates: can build without knowing AST manipulation
- AST Manipulation for Code Transformation & Translation
  - Use full capabilities of Asp
  - Let specializer users write code

Stencil Example

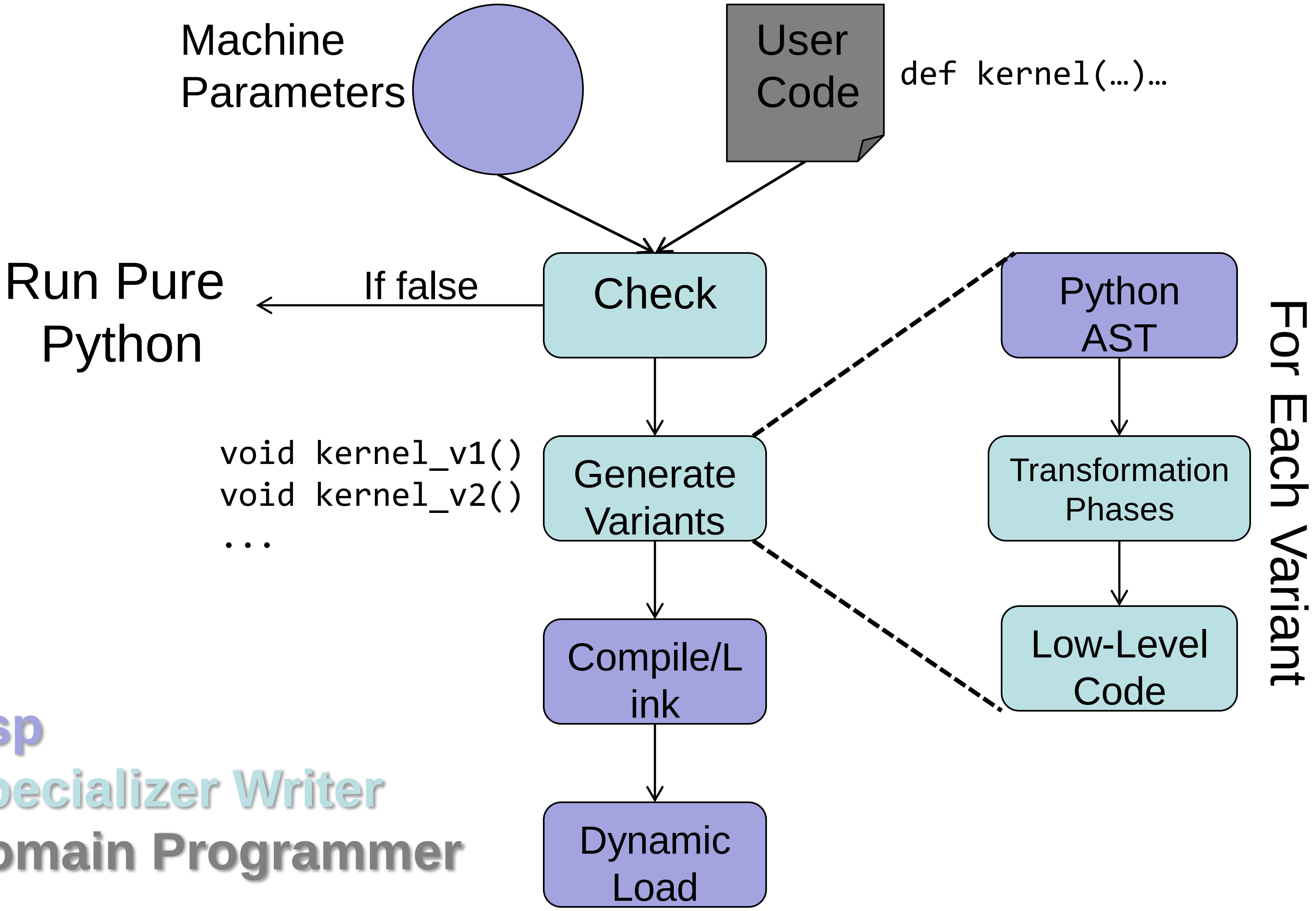
```
import stencil_kernel as sk

class LaplacianKernel(sk.StencilKernel):
  def kernel(self, in_grid, out_grid):
    for x in out_grid.interior_points():
      for y in in_grid.neighbors(x, 1):
        out_grid[x] = out_grid[x] + (1/6) * in_grid[y]

...

LaplacianKernel().kernel(in_grid, out_grid)
```

Specializer Structure: First Run



Asp

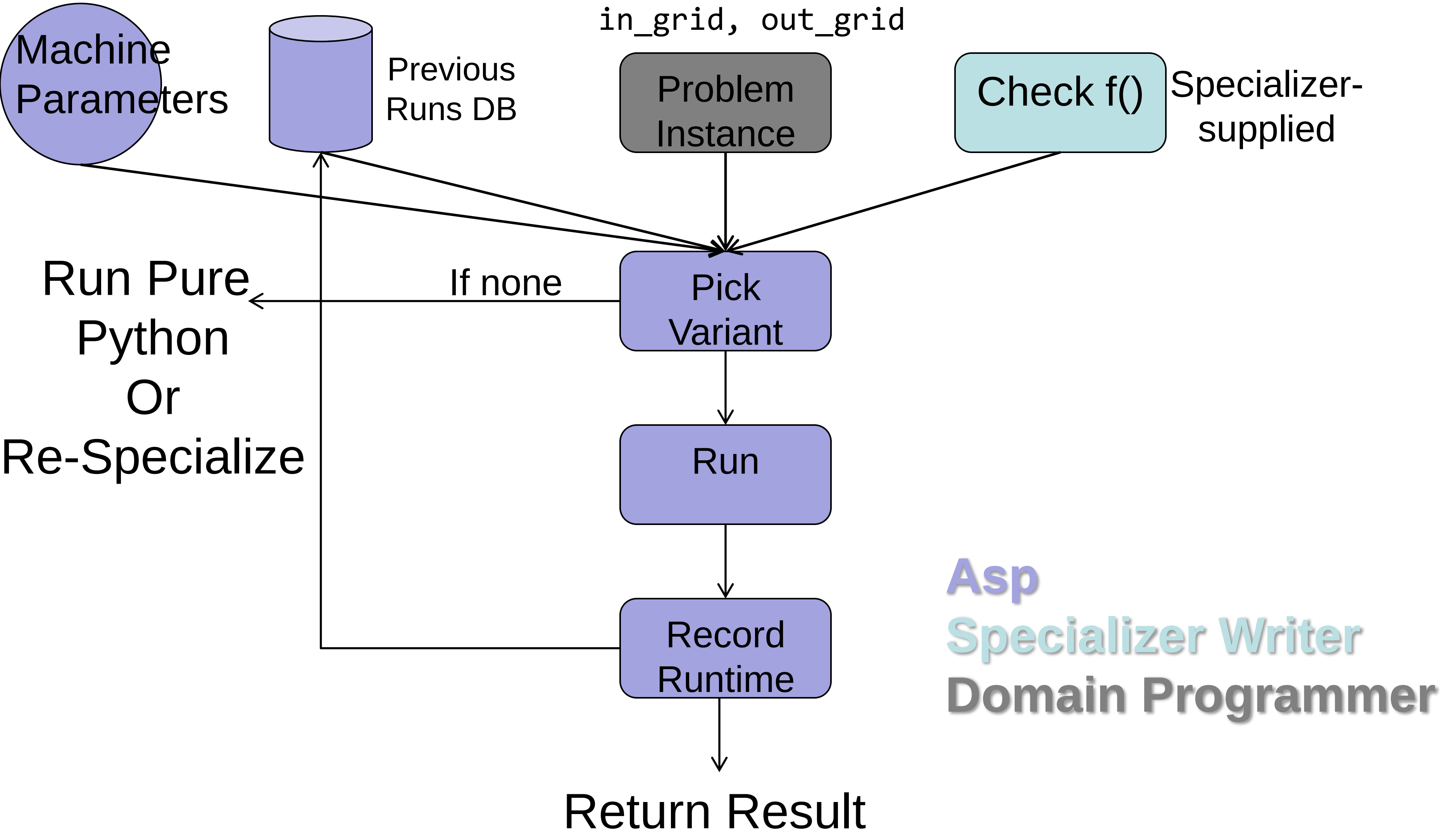
Specializer Writer

Domain Programmer

EECS
Electrical Engineering and Computer Sciences

Specializer Structure: Run

BERKELEY PAR LAB



AST-based Specializers: 4 Phase Transformation

1. Python AST => domain-specific AST
2. Optimize domain-specific AST
3. Domain-specific AST => platform AST
4. Platform AST => code generation

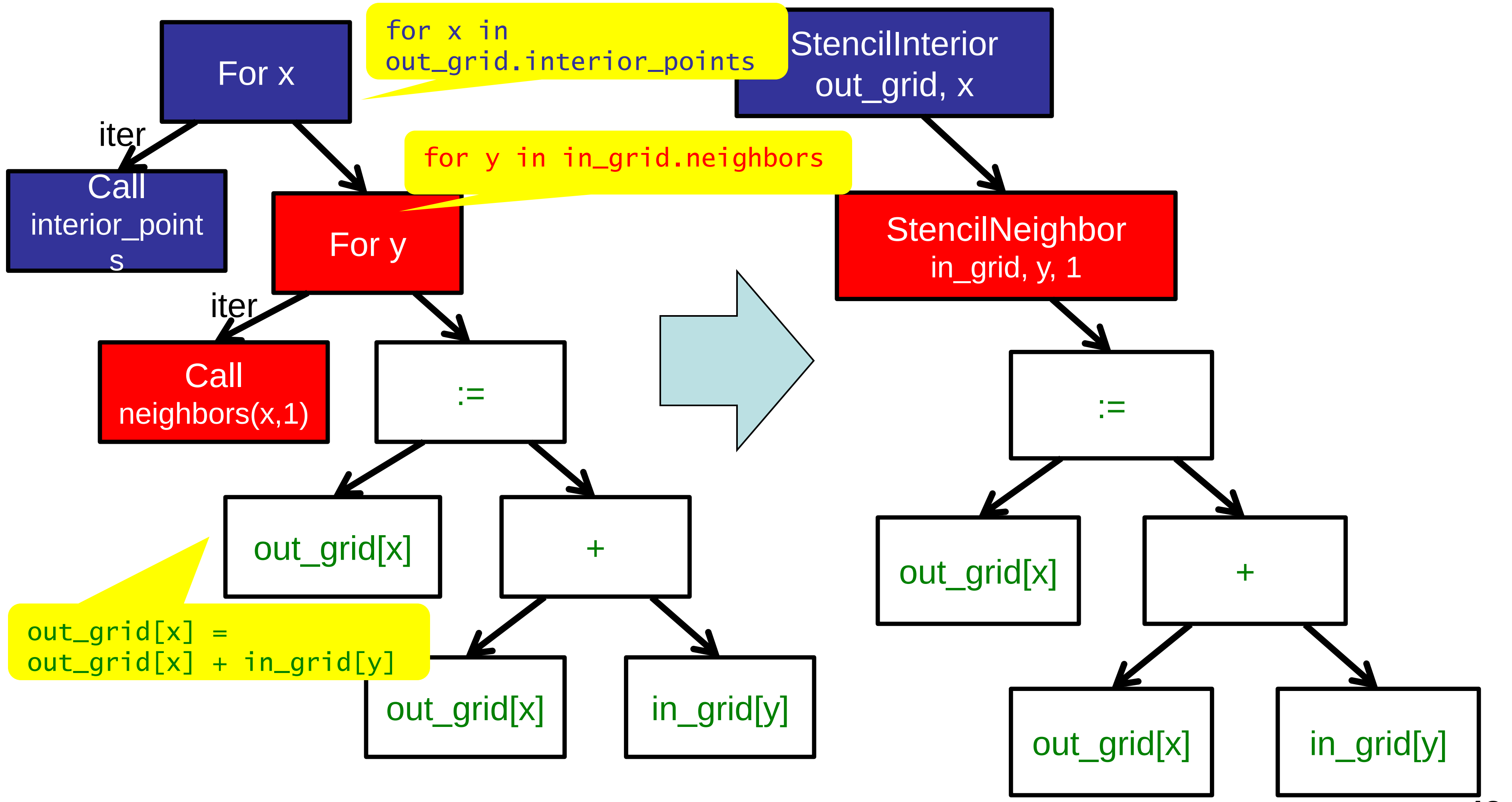
- All steps use tree visitor pattern
- Write “handlers” that are called when a node type is encountered
- See Derrick Coetzee’s poster for a walkthru example

EECS
Electrical Engineering and Computer Sciences

Transformation with Domain Knowledge

BERKELEY PAR LAB

Generic => Domain-Specific

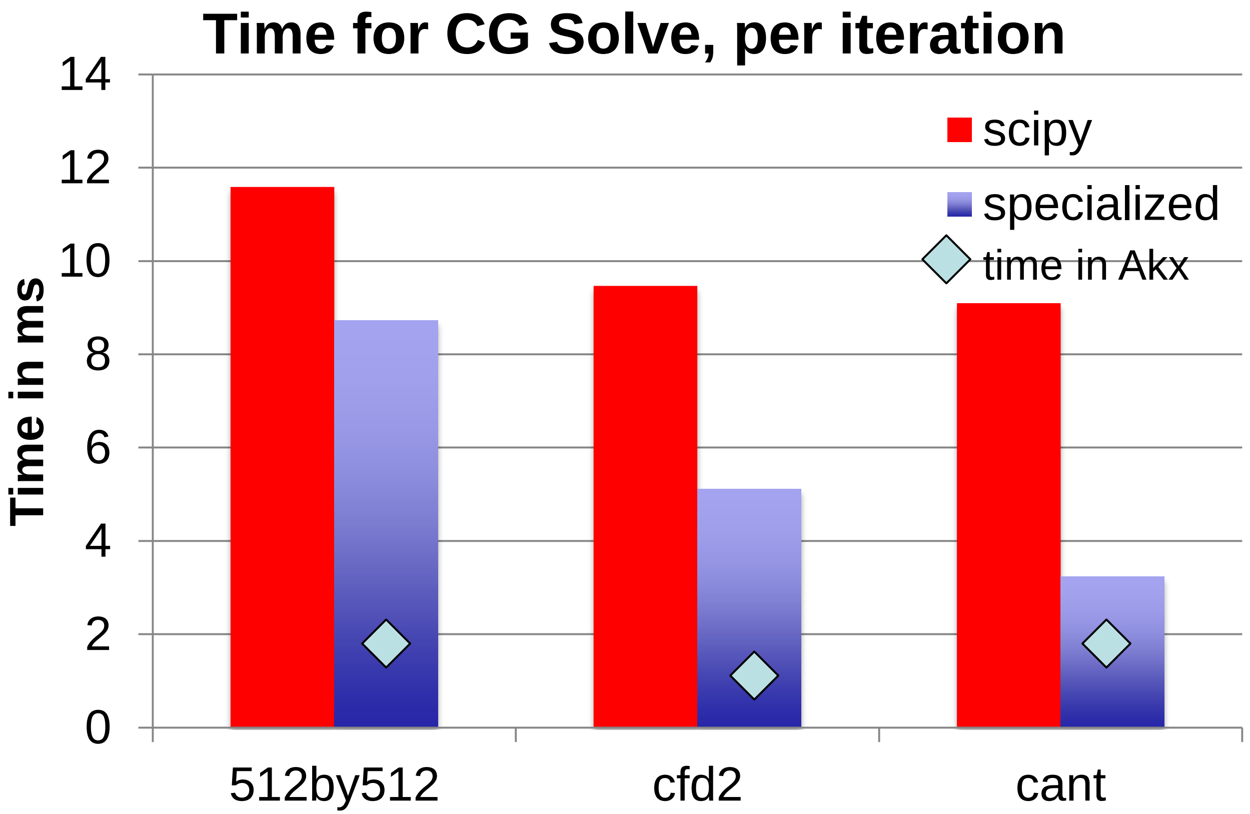

For x
iter
Call interior_points
for x in out_grid.interior_points
For y
for y in in_grid.neighbors
iter
Call neighbors(x,1)
:=
out_grid[x]
+
out_grid[x]
in_grid[y]
out_grid[x] = out_grid[x] + in_grid[y]
StencilInterior out_grid, x
StencilNeighbor in_grid, y, 1
:=
out_grid[x]
+
out_grid[x]
in_grid[y]

EECS
Electrical Engineering and
Computer Sciences

Recent Results: Matrix Powers

BERKELEY PAR LAB

- Now have a Communication-Avoiding CG using our CA Matrix Powers kernel
  - Matrix Powers is auto-tuned

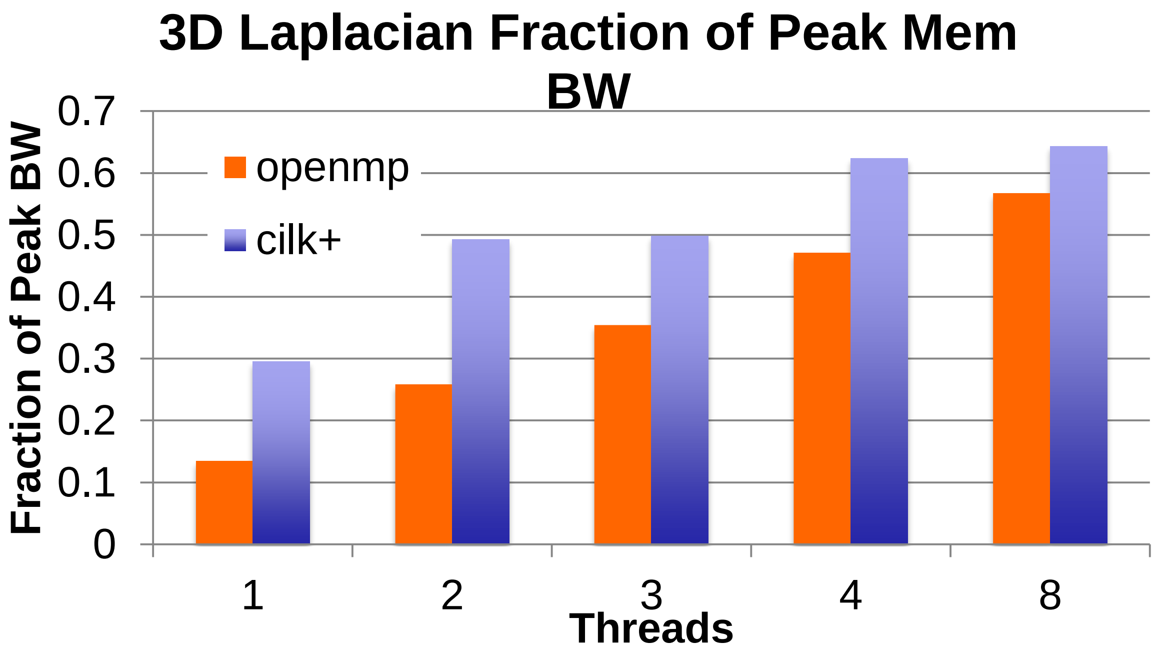


EECS
Electrical Engineering and Computer Sciences

Recent Results: Stencil

BERKELEY PAR LAB

- Testbed for AST transformations
- Supports many stencils already
- Optimizations, auto-tuning being added
  - Only register blocking enabled, already >65% of peak
- Believe can obtain >90% of peak



Get Involved

- We want your feedback
  - Many open questions
- Goal: Make it easy to start development
  - Quick development VM available

- Source
  - http://github.com/shoaibkamil/asp.git
  - Wiki: http://github.com/shoaibkamil/asp/wiki

Thank You