Asp Specializers

Shoaib Kamil
Armando Fox, Katherine Yelick, Derrick Coetzee, Jeffrey Morlan, Young Kim, David Johnson & many more
Par Lab/UPCRC Retreat, Summer 2011
Outline

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

Selective Embedded JIT

- .py
- f()
- h()

PLL Interp

Specialization

Specializer

cc/ld

$ .so

Productivity app

OS/HW
Asp is a SEJITS infrastructure for Python

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
Status and Highlights

- 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?

App author (PLL)  Specializer author (ELL)  SEJITS team  3rd party library

Application  Specializer  Asp core  e.g. MKL

Kernel  Domain-Specific Transforms  Target AST  Compiled libraries

Kernel call & Input data  Utilities  Asp Module

Results
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
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]

...
Specializer Structure: First Run

Machine Parameters

User Code

def kernel(...)

Run Pure Python

If false

void kernel_v1()
void kernel_v2()
...

Check

Generate Variants

For Each Variant

Compile/Link

Dynamic Load

Python AST

Transformation Phases

Low-Level Code

Asp Specializer Writer
Domain Programmer

Python AST Transformation Phases

Low-Level Code
Specializer Structure: Run

Pick Variant

Problem Instance

in_grid, out_grid

Check f()

Specializer-supplied

Previous Runs DB

Run Pure Python Or Re-Specialize

If none

Run

Record Runtime

Return Result

Machine Parameters

Asp
Specializer Writer
Domain Programmer

Return Result
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 walkthrough example
Transformation with Domain Knowledge

Generic $\Rightarrow$ Domain-Specific

For $x$

Call interior_points

For $y$

Call neighbors($x$, 1)

out_grid[$x$] = out_grid[$x$] + in_grid[$y$]

StencilInterior out_grid, $x$

for $x$ in out_grid.interior_points

StencilNeighbor in_grid, $y$, 1

for $y$ in in_grid.neighbors

out_grid[$x$] += out_grid[$x$] + in_grid[$y$]

out_grid[$x$] + in_grid[$y$]
Recent Results: Matrix Powers

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

Time for CG Solve, per iteration

<table>
<thead>
<tr>
<th></th>
<th>512by512</th>
<th>cfd2</th>
<th>cant</th>
</tr>
</thead>
<tbody>
<tr>
<td>scipy</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>specialized</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>time in Akx</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recent Results: Stencil

- 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