SEJITS
Chick Markley: Staff Programmer

- Selected
- Embedded
- Just-In-Time
- Specialization
SEJITS:

It’s a methodology

– Write in a high level language
– Identify performance bottle neck
– Use JIT code generation to optimize
– Used in the projects of numerous graduate students
# Implemented ASP Specializers

<table>
<thead>
<tr>
<th>DSEL/Library</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stencil/Structured Grid (Shoaib Kamil)</td>
<td>x86+OpenMP</td>
</tr>
<tr>
<td>Semantic Graphs Filtering &amp; Semiring Operations in KDT (Aydin Buluc)</td>
<td>x86+MPI</td>
</tr>
<tr>
<td>Parallel Map (Michael Driscoll)</td>
<td>x86+processes, cloud</td>
</tr>
<tr>
<td>PyCASP/Gaussian Mixed Models/GMM (Katya Gonina)</td>
<td>CUDA, Cilk Plus</td>
</tr>
<tr>
<td>CA Matrix Powers for CA Krylov Subspace Methods (Jeffrey Morlan)</td>
<td>x86+pthreads</td>
</tr>
<tr>
<td>Bag of Little Bootstraps (Peter Birsinger, Richard Xia, Aakash Prasad)</td>
<td>x86+Cilk Plus, Cloud via Spark</td>
</tr>
<tr>
<td>Map Over Combinations (Shoaib Kamil)</td>
<td>Python</td>
</tr>
</tbody>
</table>
SEJITS:

• It’s a software framework
  – Let’s build on what’s gone before.
  – Ctree implement SEJITS as a software library
  – Integrate with autotuning.
  – Most of the projects implemented by undergraduates
Anatomy of specialization

while x:
    x += y
z = x / y

int func0(
    int x = 0;
    ...
}

DSL program embedded in Python

Python AST

Semantic Model AST

C AST

C code for JIT
Selective activity app

.f()

h()

OS/HW

PLL Interp

Productivity app

Specializer

DSEL Compiler

Specialization

SEJITS Review

Selective Embedded JIT

.f()

h()

.py

.C

cc/ld

$

.so

OS/HW
# Implemented ASP Specializers

<table>
<thead>
<tr>
<th>DSEL/Library</th>
<th>Platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support Vector Machine (Michael Lin)</td>
<td>x86</td>
</tr>
<tr>
<td>A* Search (Hugo Barreto)</td>
<td>x86</td>
</tr>
<tr>
<td>Latte (Full Python Caffe (Leonard Truong)</td>
<td>x86 + GPU</td>
</tr>
<tr>
<td>Skye (real-time movement tracking (Mihir Patil)</td>
<td>X86 + OpenCV</td>
</tr>
<tr>
<td>Generating Chisel from stencils (Sean Roberts)</td>
<td>Python</td>
</tr>
<tr>
<td>Magnetic Particle Imaging (Mihir Patil)</td>
<td>X86, GPU</td>
</tr>
<tr>
<td>PyGMG: Snowflake (Nathan Zhang, Shiv Sundram)</td>
<td>X86, MPI, GPU</td>
</tr>
</tbody>
</table>
Snowflake & PyGMG
Lessons Learned

• As time went on we had fewer and fewer graduate students
• The SEJITS team moved to staff and more undergraduates
• Pattern coverage increased
• New deep results decreased
Solid underpinnings

- SEJITS relied on many external libraries.
- Some were small and poorly supported
- Bugs in libraries forced upgrades
- Upgrades led to incompatibilities.
- Changing horses required re-engineering
Finding Good Applications

• Sometimes close is good enough
• When 2 or 3 order of magnitudes improvements are achieved easily
• Somehow the effort to get another few percent is harder to find
Writing specializers

• The skill is orthogonal to the skills of performance and productivity programming
• Even in selective scopes it’s still writing compiler.
• Hard to teach, hard to learn.
SEJITS, not just performance library, it’s also a really good sieve
Thank you
Thank you.
SEJITS: “Selective, Embedded, Just-In Time Specialization”

• SEJITS bridges productivity and efficiency layers through specializers embedded in a modern high-level productivity language, Python
  – Embedded “specializers” use language facilities to map high-level patterns to efficient low-level code (at run time, install time, or development time)
  – Specializers can incorporate or package auto-tuners
Productivity-Performance Gap

- Productivity programmers want to concisely express a problem in an application domain.
- Efficiency programmers focus on packaging efficient frameworks and libraries for use in the productivity layer.
- What kinds of frameworks and libraries should the efficiency layer provide?

Efficiency Languages (C++, C/Assembler, CUDA/OpenCL, Java)

Productivity Languages (Python, Scala)