SEJITS
HPC optimization and auto-tuning in Python

Chick Markley,
Michael Driscoll, Leonard Truong
AspireLab
HPC optimization and auto-tuning for Python code w/SEJITS

- SEJITS (Selected Embedded Just-In-Time Specialization)

**Presenters**
- Chick Markley, Staff Programmer for the Aspire Lab in EECS
- Lenny Truong, Undergraduate Research Assistant on the SEJITS project

**Framing Questions**
- How do I know my use case is a good fit for SEJITS?
- Will my existing Python code benefit without rewriting part of it?
- Is it worth investing time to write a specializer if I move code to other systems which may not have SEJITS installed?
- Can I install it myself on my own laptop or cluster, or does it require special expertise and privileges to take advantage of the hardware?
Algorithms and Specializers for Provably Optimal Implementations with Resiliency and Efficiency

**Applications**
- Interactive Cloud
- Cancer Genomics
- Machine Learning
- Graph Processing
- Multimedia Analysis
- Computer Vision
- Software Radio

**Computational and Structural Patterns**

**Communication-Avoiding Algorithms**

**Productivity Languages** (Python, Scala) with **Pattern Frameworks**

**Efficiency Languages**
- C++
- CUDA/OpenCL
- JVM

**Vendor-Pattern-Specific VMs**

**Chisel Patternflow**

**Hardware Patterns**

**Chisel HDL**

**Efficiency**

**Productivity Layer**

**Efficiency Layer**

**Architecture**
- COTS CPU/GPU
- ESP: Ensembles of Specialized Processors
- Hurricane Spatial Computing Fabric
- COTS Tools
- FPGA
- ASIC

**Runtimes, OS, Hypervisor, Cluster Manager**

**COTS Tools**

**Memories, Interconnects, I/O**

**Energy-Efficient Resilient Circuit Design**

**Racks (10s kW)**

**Embedded (10 kW-1W)**

**Mobiles (1W)**

**Simulations and Modeling**
• Selective
• Embedded
• Just-In-Time
• Specialization
Anatomy of SEJITS

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

DSL program embedded in Python

- Python AST
- Semantic Model
- C AST

void *fn
Our Niche is where...

- Run time information affects performance
- Run-time tuning can find optimal configuration
- Meta-specialization combines patterns
Our audience

- Domain experts
- Efficiency experts
- Specialized hardware producers
- Ultimately: Scientists needing a performance edge
Success Stories

• Stencil specializers
• Linear algebra for optical processing (Hindemith)
• Audio event detection using gaussian mixture models (GMM)
Thanks

Sejits.org

Chick Markley  chick@eecs.berkeley.edu
### Performance Gap

<table>
<thead>
<tr>
<th>Graph Processing</th>
<th>Multimedia Analysis</th>
<th>Computer Vision</th>
<th>Software Radio</th>
</tr>
</thead>
</table>

#### Efficiency vs. Productivity

- **Productivity** programmers want to concisely express a problem in the 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?

**Productivity Languages (Python, Scala)**

<table>
<thead>
<tr>
<th>Software Radio</th>
<th>Computer Vision</th>
<th>Machine Learning</th>
<th>Cancer Genomics</th>
<th>Graph Processing</th>
<th>Multimedia Analysis</th>
<th>Interactive Cloud</th>
<th>Multimedia Analysis</th>
</tr>
</thead>
</table>

- **10 LOC**
- **100-1000 LOC**
SEJITS

Selective Embedded Just-In-Time Specialization
Achieving near optimal performance in high level productivity languages

About SEJITS

The SEJITS project provides tools and frameworks to bridge the gap between high level productivity languages and high performance hardware. Specialized kernels written with SEJITS can provide near optimal performance using information available only at run-time.

- Multiple frameworks for specializer development
- Multiple backend support: OpenCL, OpenMP, CUDA
- Automatic project generation
- Travis Continuous Integration
- IPython Integration
- Custom AST transformation GUI
- Integrated energy monitor API
- Specializer composition tools
- Open source on github
- Complete set of docs

Tweets

- UCB SEJITS
  - Prepping for the ASPIRE retreat.
  - And we are off.

Load More
Example: ArrayDoubler

```python
class Doubler(ArrayOp):
    def apply(n):
        return n * 2

c_doubler = Doubler()

A = numpy.array(512)
c_doubler(A)
```

```c
static inline float apply(float n) {
    return n * 2;
}

void apply_all(float* A) {
    for (int i = 0; i < 512; i++) {
        A[i] = apply(A[i]);
    }
}
```
Other accomplishments

- Backed by a continuous integration process
- Programmer visualization tools
- WattsUpMeter built in API
- Project generator
- Re-worked stencil specializer
- Re-coding Hindemith [Composition]
Lessons from StencilGrid

- Fewer lines of code (~15%)
- Equivalent or better performance
- OpenCL so far performs as well as OpenMP
- OpenTuner vastly simplifies specializer auto-tuning
WattsUp Power Meter API

- Sits between outlet and computer
- Uses USB cable
- Simple API allows application to query power over controlled intervals
OpenTuner support

• Declare tuning space up front:
  – tuner.add_parameter( IntegerParam("cx", 32, 1024) )
  – tuner.add_parameter( BooleanParam("unroll") )

• Declare search objective(s):
  – Minimize time, energy, size
  – Maximize accuracy, confidence

• Tuner yields an infinite stream of “program configurations”:
  – Uniquely identify program to be built.
  – Used in caching generated code.
  – Specializer must report its own performance.
DGEMM with tuning for energy vs. for time

• Simple code generator for DGEMM.
  – Square matrices only.
  – 1 level of cache blocking (cx, cy).
  – Register blocking via AVX (rx, ry).
  – OpenTuner search driver.
  – WattsUp power meter.

• 100 trials (~2 sec/each):
  – Energy-optimal solution: 1.41 sec, 38.3 joules
  – Time-optimal solution: 1.16 seconds, 73.8 joules
Initial Results

![Graph showing energy-optimized and time-optimized results.](image)
Testing and integration

```script
# run test suite from home directory to verify installation
- cd ${HOME}
- nosetests --where=${TRAVIS_BUILD_DIR}/test

# run test suite again from build dir to get coverage info
- cd ${TRAVIS_BUILD_DIR}
- nosetests --verbose --with-coverage --cover-package=ctree --

after_success:

# return early if not building ucb-sejits/ctree
- "if [[ \"x${TRAVIS_REPO_SLUG}\" != \"xucb-sejits/ctree\" ]]; then echo 'No

# only build docs using Python 2.7
- "if [[ \"x$PYTHON_VERSION\" != \"x(2, 7)\" ]]; then echo 'No
```
C AST

Returned from transform()
IPython

In [5]: from ctree.transformations import PyBasicConversions
t = PyBasicConversions()
tree2 = t.visit(tree1)
ctree.ipython_show_ast(tree2)
Custom GUI

```python
from numpy import *
from stencil_code.stencil_kernel import *
from stencil_code.stencil_grid import *

import sys
alpha = 0.5
beta = 1.0

class LaplacianKernel(StencilKernel):
    def __init__(self, alpha, beta):
        super(LaplacianKernel, self).
        self.constants = {'alpha': alpha, 'beta': beta}

    def kernel(self, in_grid, out_grid):
        for x in in_grid.
            for y in in_grid.
                out_grid[x + y] = alpha * in_grid[x + y] + beta *

        nx = int(sys.argv[1])
        ny = int(sys.argv[2])
        nz = int(sys.argv[3])
        input_grid = StencilGrid(nx, ny, nz)
        output_grid = StencilGrid(nx, ny, nz)

        for x in input_grid.
            for y in input_grid.
                laplacian = LaplacianKernel(alpha, beta)
                input_grid[x, y] = random.randint(nx)

        laplacian.kernel(input_grid, output_grid)
```

10/13/14 Chick Markley
Project Generator


ctree

options:
-h, --help            show this help message and exit
-sp STARTPROJECT, --startproject STARTPROJECT
-wu, --writeupdates   write updates
-p PORT, --port PORT
-v, --verbose         be verbose

PyCharm

main.py - chick_specializer - [-/work/chick_specializer]

chick_specializer (~/work/chick_specializer)

bin
  cache

chick_specializer
  __init__.py

main.py

doc
tests
c.
cree.cfg
  __init__.py
  README.rst
  setup.py

External Libraries
Hindemith

- A SEJITS specializer
- Written by Michael Anderson
- Python and C++ not using our SEJITS framework
- All about composition
- Planned for conversion to CTree this summer
- Done so quickly by Lenny Truong that we had to make him the next speaker.
Collaborations

• HWACHA integration
  – Write test and validation code in python

• AmpLab
  – Increase leaf-node performance of large collaborative filtering algorithms
  – Leverage this work towards Scala as an alternative development environment

• Optics
  – Use SEJITS for computation intensive measurement and control of wave effects

• With you
  – ????