High-Performance Code Generation Using CorePy

Andrew Friedley

Open Systems Laboratory, Indiana University

SciPy Conference – August 20, 2009

pervasivetechnologylabs
AT INDIANA UNIVERSITY

CorePy

CorePy is a Python-based object-oriented assembler.

But, CorePy is ***not*** your usual assembler…

- Instructions and processor resources are first-class objects in Python
- Programs can be generated and transformed using Python
- All machine-level code is synthesized directly in Python – no compilers or assemblers required!

pervasivetechnologylabs
AT INDIANA UNIVERSITY

A Simple (x86_64) Example

```
# Create a simple synthetic program
>>> prgm = x86_env.Program()
>>> code = prgm.get_stream()

# Synthesize assembly code
>>> code += x86.mov(prgm.gp_return, 31)
>>> code += x86.add(prgm.gp_return, 11)
>>> prgm += code

# Execute the synthetic program
>>> proc = Processor()
>>> result = proc.execute(prgm)
>>> print result
42
```

$r = 31 + 11$

Supported Platforms

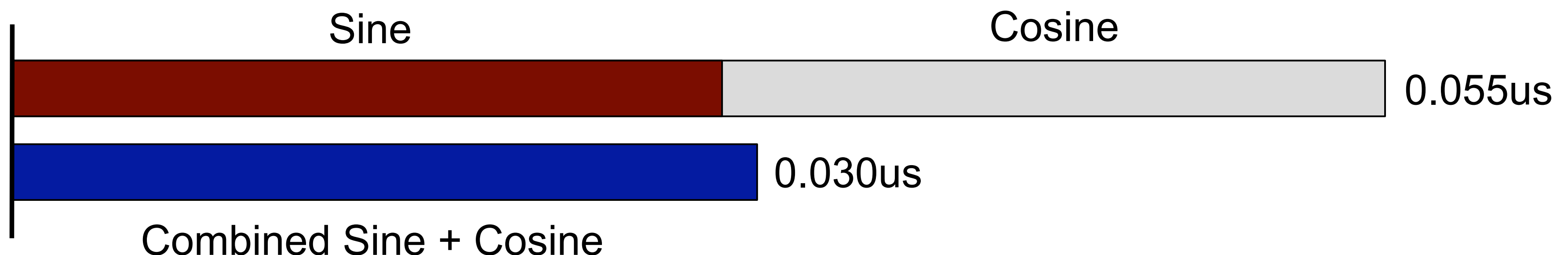
- Operating Systems:
  - Linux
  - OSX
- Architectures:
  - PowerPC (incl. Altivec)
  - Cell SPU
  - **x86 32- and 64-bit** (incl. all SSE)
  - **ATI GPU**
  - **NVIDIA GPU** (in progress)

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Instruction Scheduling

- Automate the task of rearranging instructions for performance
  - Saves time/effort for the programmer

- Combine and interleave code segments to take advantage of pipelining
  - Sine and cosine operations, for example:



(can apply the same technique to unrolled loops!)

CoreFunc Framework

- Google Summer of Code Project:
  Implementing Ufuncs Using CorePy
- Write only the loop bodies
  - Framework initializes state, generates loop code and atomic reduction operations
- Multi-core parallelization is automatic
  - C function wraps user code and splits work

Example: Addition

```
def gen_ufunc_add_float32():
  def main_fn(prgm, reg):
    code = prgm.get_stream()

    code += x86.movaps(xmm0, MemRef(reg['args'][0], data_size = 128))
    code += x86.addps(xmm0, MemRef(reg['args'][1], data_size = 128))
    code += x86.movntps(MemRef(reg['args'][2], data_size = 128), xmm0)
    return code

  def special_fn(prgm, reg):
    code = prgm.get_stream()

    code += x86.movss(xmm0, MemRef(reg['args'][0], data_size = 32))
    code += x86.addss(xmm0, MemRef(reg['args'][1], data_size = 32))
    code += x86.movss(MemRef(reg['args'][2], data_size = 32), xmm0)
    return code

  return gen_ufunc(2, 1, main_fn, special_fn, 4,
                   x86.addss, ident_xmm_0, XMMRegister, 32)
```

Example: Addition

```
>>> import numpy
>>> import corefunc

>>> a = numpy.arange(10, dtype=numpy.int32)
>>> b = numpy.arange(10, dtype=numpy.int32)

# NumPy ufunc
>>> numpy.add(a, b)
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18], dtype=int32)

# CorePy ufunc
>>> corefunc.add(a, b)
array([ 0,  2,  4,  6,  8, 10, 12, 14, 16, 18], dtype=int32)

# Reduction works the same too:
>>> corefunc.add.reduce(a)
45
```

Addition Performance

Throughput (Mops/sec)

10000
1000
100
10
1
0.1

10 100 1000 10000 100000 1e+06 1e+07 1e+08

Array length (elements)

NumPy
CorePy 1 core
CorePy 2 cores
CorePy 4 cores
CorePy 8 cores

Two Intel Xeon L5320: Quad-core 1.86GHz 8Mb Cache

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Addition Reduce Performance

Throughput (Mops/sec)

100000
10000
1000
100
10
1

10 100 1000 10000 100000 1e+06 1e+07 1e+08

Array length (elements)

NumPy
CorePy 1 core
CorePy 2 cores
CorePy 4 cores
CorePy 8 cores

Two Intel Xeon L5320: Quad-core 1.86GHz 8Mb Cache

Custom Ufuncs

- Vector Normalization

```
def npy_normal(a, b):
  l = numpy.sqrt(a**2 + b**2)
  return (a / l, b / l)
```

  - Reduce memory accesses
    - No temporary storage other than registers
  - Take advantage of special instructions
    - Reciprocal-squareroot

pervasivetechnologylabs AT INDIANA UNIVERSITY

Vector Normalization Performance

Throughput (Mops/sec)

10000
1000
100
10
1
0.1

10 100 1000 10000 100000 1e+06 1e+07

Array length (elements)

NumPy
CorePy 1 core
CorePy 2 cores
CorePy 4 cores
CorePy 8 cores

Two Intel Xeon L5320: Quad-core 1.86GHz 8Mb Cache

Ideas & Feedback

- Suggestions for lowering CorePy entry barrier?
- CorePy feature requests?
- What could CorePy provide to better integrate with NumPy?
- How can I make my ufunc work more useful?

pervasivetechnologylabs
AT INDIANA UNIVERSITY

Thank You!

<u>CorePy Development Team</u>

Andrew Friedley

Ben Martin

Chris Mueller

<u>Funding/Enablement</u>

Lilly Foundation

Andrew Lumsdaine/Open Systems Lab

CorePy is now BSD licensed!

www.corepy.org