VM, Micropython

Paolo Giarrusso Sigurd Meldgaard

December 22, 2008

Micropython

A (fast) interpreter of Python bytecodes.
Written in C with small assembly stubs.
64 and 32 bits, both support of Linux and Mac OS X.

- GC
- Indirect threading
- Tagged Integers

Some statistics

524 Revisions

```
$ wc -l *.[ch]
   4351 total
```

GC

We implemented a real garbage collector.

- Simple semi-space stop and copy.
- No handles.
- Smart trick for handling overflow.

Indirect threading

As we learned the first week on the course, threading could improve dispatch performance considerably.

- And ...
- It did.

Tagged integers

Object pointers are tagged with LSB 1, and integers with 0.

- A real speedup for integer intensive benchmarks.
- No real slowdown on dereferencing a pointer -1.
- A mess to handle ...
- If not for `tagged_el_t` (incompatible with both pointers and long)!

General optimizations

- Keep important variables local (register allocated).
- Inlining of as much as possible (evt. with macros).

Benchmarks

Benchmark	MP(32)	CP(32)	Faster	MP(64)	CP(64)	Faster
`list`	11.70	27.91	58.1%	6.72	21.97	69.4%
`while`	6.86	15.18	54.8%	4.66	11.01	57.7%
`fact`	7.13	20.93	66.0%	6.18	16.33	62.2%
`gc`	5.32	17.47	69.6%	7.03	13.40	47.5%

Figure: Timings of our interpreter (MP) vs. CPython (CP) on our two platforms (32/64 bits) and how many percent we are faster. All measurements are in seconds as given by the Unix `time` command.

OProfile

Measuring indirect branches.

```
# opcontrol -e CPU_CLK_UNHALTED:1000000:0:1:1 \
  -e BR_IND_EXEC:1000000:0:1:1
$ sudo opcontrol --start; \
make mega_bench_micro; \
make mega_bench_python; \
sudo opcontrol --shutdown
```

OProfile

```
$ opreport interp_release /usr/bin/python
CPU: Core 2, speed 2000 MHz (estimated)
Counted BR_IND_EXEC events (Indirect Branch instructions ex
  with a unit mask of 0x00 (No unit mask) count 1000000
Counted CPU_CLK_UNHALTED events (Clock cycles when not halt
  with a unit mask of 0x00 (Unhalted core cycles) count 100
BR_IND_EXEC:10...|CPU_CLK_UNHALT...|
  samples|      %|  samples|      %|
------------------------------------
     4921 55.1867    110453 72.5124 python2.5
     3996 44.8133     41870 27.4876 interp_release
```