PyPy status talk

Maciej Fijalkowski
merlinux GmbH

EuroPython 2008

July 7, 2008

What this talk is about

- more details about recent developments
- our plans for the near future
- how we're going to achieve them
- readers of our blog might know many of those things

Production ready

- we worked a lot on running existing applications on top of PyPy
- sometimes requiring to change applications slightly
- especially refcounting details tend to be a problem

```
open('xxx', 'w').write('stuff')
```

CTypes

- official way to have bindings to external (C) libraries for PyPy
- slow, but getting better
- can handle i.e. pysqlite-ctypes, pyglet, pymunk or Sole Scion
- ctypes is getting better as a side effect:
  - *errno handling*
  - *bugfixes*
  - *helper libraries*
- part of google sponsoring
- demo

CTypes configure

- our own small addition to general CTypes usefulness
- invokes C compiler for small details
- can handle #defines, types, structure layout etc.

Sqlite

- part of cpython stdlib since 2.5
- we use Gerhard Haering's CTypes version
- works reasonably well after some fixes

Django

- we run (almost) unmodified Django
- only sqlite DB backend for now
- cooperation with Django people to make sure that Django 1.0 works with PyPy
- http://www.djangoproject.com/

Pylons

- worked almost out of the box once eggs were working (1 day)
- no SQLAlchemy yet, obscure problems ahead
- unmodified passes all tests
- http://pylonshq.com/

Twisted & Nevow

- twisted have some glitches, but mostly works
- nevow works
- we don't support PyCrypto nor PyOpenSSL and we won't anytime soon (if nobody contributes CTypes or rpython versions)
- http://twistedmatrix.com/

Other software

- BitTorrent
- PyPy translation toolchain
- various smaller things, templating engines, most pure-python software

Obscure details that people rely on

- non-string keys in __dict__ of types
- exact naming of a list comprehension variable
- relying on untested and undocumented private stuff (zipimport._zip_directory_cache)
- exact message matching in exception catching code
- refcounting details

Conclusion on Compatibility

- lessons learned: There is no feature obscure enough for people not to rely on it. But PyPy can usually mimick CPython sufficiently.
- pypy-c probably most compatible to CPython Interpreter
- main blocker for running apps will be missing external modules

Speed - comparison with CPython

- we're something between 0.8-2x slower than CPython on various benchmarks.
- gcbench - 0.8 (because of our faster GC)
- steady but slow progress
- we hope for our JIT to be a huge leap ahead

Speed - decent GCs

- faster than refcounting
- better handling of unusual patterns
- troubles with for example communication with C
- details on different talk

Speed - JIT generator

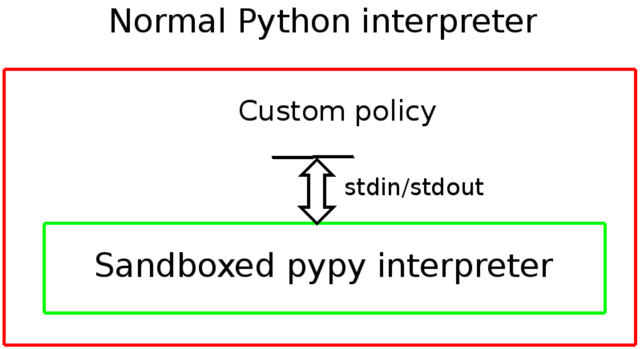
- not ready yet!
- will be super fast
- prolog prototype
- psyco is a nice proof that this approach would work

Other backends

- PyPy-jvm runs!
- more integration between pypy-cli and .NET
- general speed improvements
- both backends are progressing - very slowly though
- contributors wanted!

Sandboxing

- fully sandboxed python interpreter
- all external calls to C goes via another python process
- special library for making custom policies



A lot of cleanups

- got rid of semi-cool semi-working proofs of concept
- examples: CPython extension compiler, rctypes
- reduced code size
- allowed us to progress forward into advanced features
- included sprint dedicated to cleanup (on which noone was allowed to add features)

Plans

- more JIT - faster Python
- support full CPython's stdlib
- aim for more funding in pushing pypy forward (more at the next talk)

Links

- http://codespeak.net/pypy
- http://morepypy.blogspot.com
- this talk is already online