PyPy status talk

Holger Krekel Maciej Fijalkowski
Merlinux GmbH

PyCon UK 2008 - Birmingham

September 13 2008

What this talk is about

- a bit of motivation
- tell what Python Interpreter can run today
- what we are heading for with the 1.1 release
- Questions and Answers

PyPy - motivation

- CPython is nice, but not flexible enough
- IronPython, Jython - bound to the specific VM
- Separate language specification from low-level details, such as GC or platform to run
- Psyco and Stackless Python hard to maintain

PyPy - user motivation

- One should never be forced to write anything in C for performance reasons (with some exceptions: embedded devices etc.)
- Just-in-time compiler should make number-crunching and static-enough code fast enough
- One should never care about low-level details

Brief history of PyPy

- first sprint 2003, about 30 more by now
- CPython/Psyco/Jython/Stackless developers participating
- MIT-License, more sprints
- EU Research project 2004-2007
- 2007-2008 - open source project
- some google sponsoring

Getting 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
- can handle i.e. pysqlite-ctypes, pyglet, pymunk or Sole Scion, almost whatever....
- contribution to original ctypes (better errno handling, bugfixes, tests...)
- part of google sponsoring
- note: 32bit and a bit slow

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 unmodified Django 1.0
- only sqlite DB backend for now

http://www.djangoproject.com
http://code.djangoproject.com/wiki/DjangoAndPyPy

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 works (60/4500 tests failing)
- 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

- pure python should just work
- BitTorrent
- PyPy translation toolchain
- py lib
- sympy
- various smaller things, templating engines

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

Transition to 2.5

- SOC project Bruno Gola
- almost complete
- missing more testing, stdlib porting

Conclusion on Compatibility

- lessons learned: There is no feature obscure enough for people not to rely on it.
- pypy-c interpreter probably the most compatible to CPython
- main blocker for running apps will be missing external modules

Speed - comparison with CPython

- we're something between 0.8-4x slower than CPython on various benchmarks.
- steady but slow progress
- we hope for our JIT to be a huge leap ahead
- pypy-c has fastest Interpreter startup

Speed - JIT generator

- not ready yet!
- will be super fast
- some prototypes, research ongoing
- psyco is a nice proof that this approach would work

Memory - comparison with CPython

- PyPy has pluggable Garbage Collection
- gcbench - 0.8 (because of our faster GCs)
- better handling of unusual patterns
- care needed with communication with C
- GCs are semi-decent

Threading / Stackless

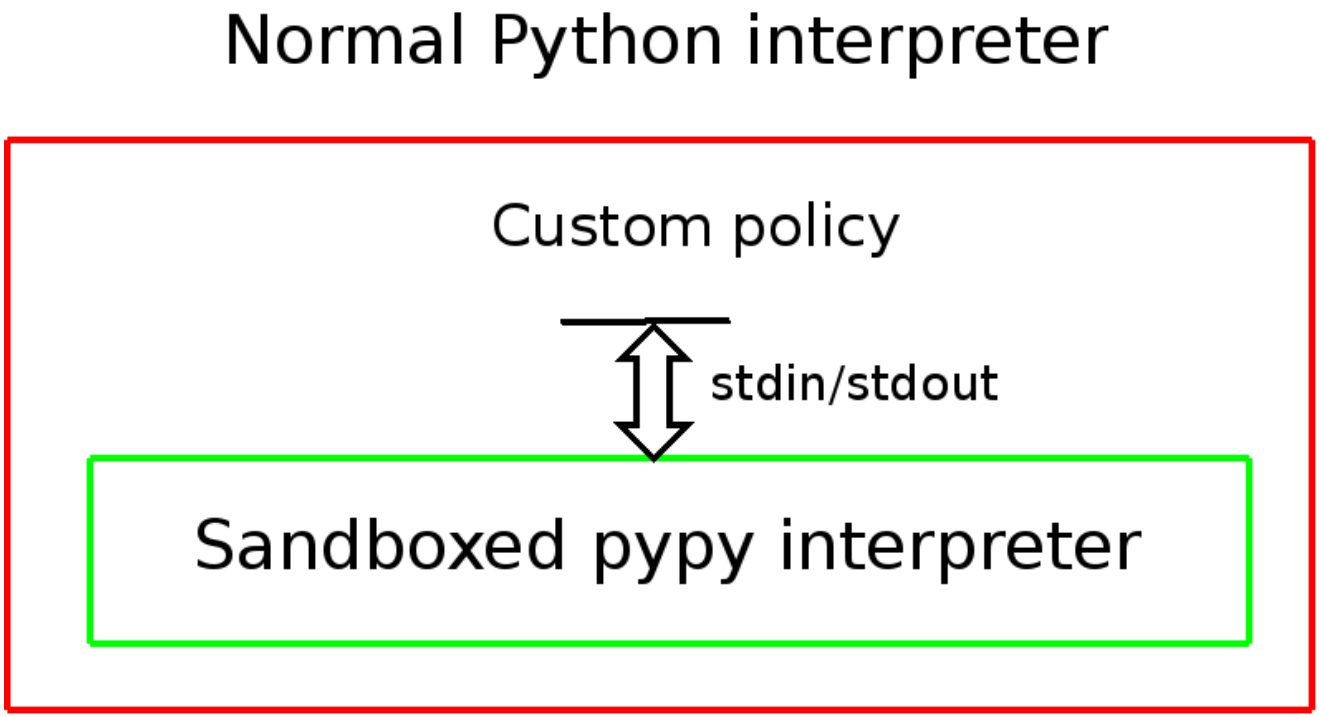
- currently using GIL, quite robust
- free threading? “it’s work”
- pypy-c has software threading / stackless
- added during translation

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



pypy-c on small devices

- cross-compilation
- startup time
- security
- RAM usage
- share interpreter state across processes
- pypy approach a very good fit!

1.1 release goals

- compatible to Python 2.5.2
- well tested on win/linux 32 bit
- running major packages unmodified
- easy_install/distutils working
- help e.g. by writing ctypes modules

Contact / Q&A

holger krekel, Maciej Fijalkowski at http://merlinux.eu
PyPy: http://codespeak.net/pypy
Blog: http://morepypy.blogspot.com