PyPy: Jak uczynić pythona szybszym

Maciej Fijałkowski

SKA South Africa

PyCon PL, 8.10.2010

O czym będe mówił

- PyPy - co jak i dlaczego?
- Kompilacja Pythona
- Problemy z optymalizacja Pythona

Słowo wstępu

- PyPy - interpreter Pythona
- Motywacja - problemy z psyco
- Open Source
- Sponsorowany przez UE poprzez FP7

Jak działa interpreter pythona?

- Kompilacja ze źródeł do bytecode'u
- Wykonywanie bytecode'u jeden po drugim
- Maszyna stosowa
- Ramki są trzymane na stercie

CPython

- Zaleta - względnie prosty interpreter
- Wada - dość wolny

- ... ale nie zawsze
- przykłady: operacje na stringach, numpy, słowniki, etc.

CPython

- Zaleta - względnie prosty interpreter
- Wada - dość wolny

- ... ale nie zawsze
- przykłady: operacje na stringach, numpy, słowniki, etc.

A może skompilować do C?

- Niewiele da sie zoptymalizować
- Głównym problemem jest dynamika zmiennych globalnych
- Efekt nie będzie wiele lepszy (do 2x szybciej, ale na ogół nawet nie)
- Odmiana - cython

Na ratunek - dynamiczna kompilacja

- JIT - just-in-time compiler
- Popularny w takich środowiskach jak JVM
- Ale też ostatnio tracemonkey (firefox), v8 (chrome)
- Dla języków dynamicznych daje lepsze wyniki niż statyczna kompilacja (np. self)

Jak to działa?

- Kompiluje kod w pamięci bezpośrednio do assemblera
- Przed wykonaniem, albo podczas
- Na ogół tylko miejsca które są często wykonywane (ale niekoniecznie)

Zalety

- Podczas wykonania programu wiemy więcej niż przed wykonaniem
- ... i to wystarczy
- Można skompilować tylko "prawdopodobny scenariusz" i dokompilować reszte jak zajdzie potrzeba

Zalety

- Podczas wykonania programu wiemy więcej niż przed wykonaniem
- ... i to wystarczy
- Można skompilować tylko "prawdopodobny scenariusz" i dokompilować reszte jak zajdzie potrzeba

Wady

- Czas kompilacji wlicza się do czasu wykonywania programu
- Spekulatywne optymalizacje mogą doprowadzić do eksplozji wielkości kodu

Mając JITa możemy

- Traktować kod funkcji jako stałą
- Nawet jeżeli został stworzony przez `exec`
- Optymalizować najbardziej prawdopodobny przypadek

AOT vs Tracing JIT

- AOT (Ahead of time) - kompiluje metodę zaraz przed uruchomieniem (v8, hotspot, self, psyco)
- Tracing - uruchamia interpreter patrząc co sie dzieje, potem kompiluje informacje (tracemonkey, pypy)

Tracing JIT

- Kompilujemy liniowy kod
- Każde rozgałęzienie zastępujemy wartownikiem
- Jeżeli wartownik zbyt często wychodzi z assemblera, kompilujemy nową ścieżkę

Problemy z CPythonem (interpreterem)

- Licznik referencji jako Garbage Collector
- Napisany w C
- Obrzydliwe API w C

Problemy z Pythonem (językiem)

- Boxing
- Ramki stosu
- Odwołania przez słowniki (atrybuty, metody, zmienne globalne)

Boxing

- Wszystkie zmienne są pakowane w “boxy”
- Jeżeli istnieją krótko, można tego uniknąć
- Ramki stosu poważnie przeszkadzają

Ramki stosu

- Dostępne z poziomu aplikacji przez `sys._getframe()`
- `locals()`, `globals()`, `sys.exc_info()`
- Zaawansowana technologia do pozbycia się tego, pytać poza prezentacją :-)

Atrybuty obiektów

- Quiz: ile razy kod musi zajrzeć do słownika:

  `x.a`

- `__getattribute__` na klasie
- deskryptor na klasie
- atrybut na obiekcie

- optymalnie by było zero

Atrybuty obiektów

- Quiz: ile razy kod musi zajrzeć do słownika:

  `x.a`

- `__getattribute__` na klasie
- deskryptor na klasie
- atrybut na obiekcie

- optymalnie by było zero

Atrybuty obiektów

- Quiz: ile razy kod musi zajrzeć do słownika:

  `x.a`

- `__getattribute__` na klasie
- deskryptor na klasie
- atrybut na obiekcie

- optymalnie by było zero

Atrybuty obiektów - ukryte klasy

- W pythonie nie znamy kształtu obiektów
- Często jest on jednak dobrze określony
- Dynamicznie tworzymy zestaw kształtów obiektów, w których wiemy gdzie leżą atrybuty

Zmienne globalne

- Większość jest statyczna
- Ale ciężko wyczuć które
- Kompilujemy "zakładając że globalne są stałe", dorzucając wartownika

Pytania?

- http://pypy.org
- http://morepypy.blogspot.com
- Prezentacja będzie w necie jak sie pojawi wifi
- #pypy na irc.freenode.net

Teleskopy