

# Kan Python være hurtigere end C?

Anders Lehmann  
[anders@hih.au.dk](mailto:anders@hih.au.dk)  
AU Herning

# Anders Lehmann

- 1989 DTU
- 1990 Bang & Olufsen
- 2005 DFKI Saarbrücken
- 2007 AutIQ Fredericia
- 2008 Vestas
- 2008 AU Herning

# Disposition

- Hvorfor er Python langsom?
- PyPy historie
- PyPy arkitektur
- PyPy Just In Time compiler
- Resultater
- Spørgsmål

# Python hurtigere end C

- Hvad betyder det at være hurtigere?
  - Er Python altid hurtigere?
  - Hvordan måler vi det?
  - Hvad betyder vores forudindfattede meninger?

# Demo

- Videoprocessing

# Python er langsom

- Eksempel  $c = a + b$ 
  - slå `__add__` op i MRO for a og kald `a.__add__(b)`
  - hvis `__add__` ikke findes slå `__radd__` op i MRO for b, og kald `b.__radd__(a)`
  - check om der er sket en exception

# Python er langsom

- Metode kald:
  - Byg en tuple med argumenterne
  - slå metode op i MRO
  - ...

# Python er langsom

- Bytecodes
- Exceptions

# PyPy historie

- 2003 Behov for en mere fleksibel fortolker
- 2004 EU projekt
- 2005 Fortolker kan køre 94% of Python's test
- 2006 Selfhosting
- 2007 First JIT

# PyPy Historie

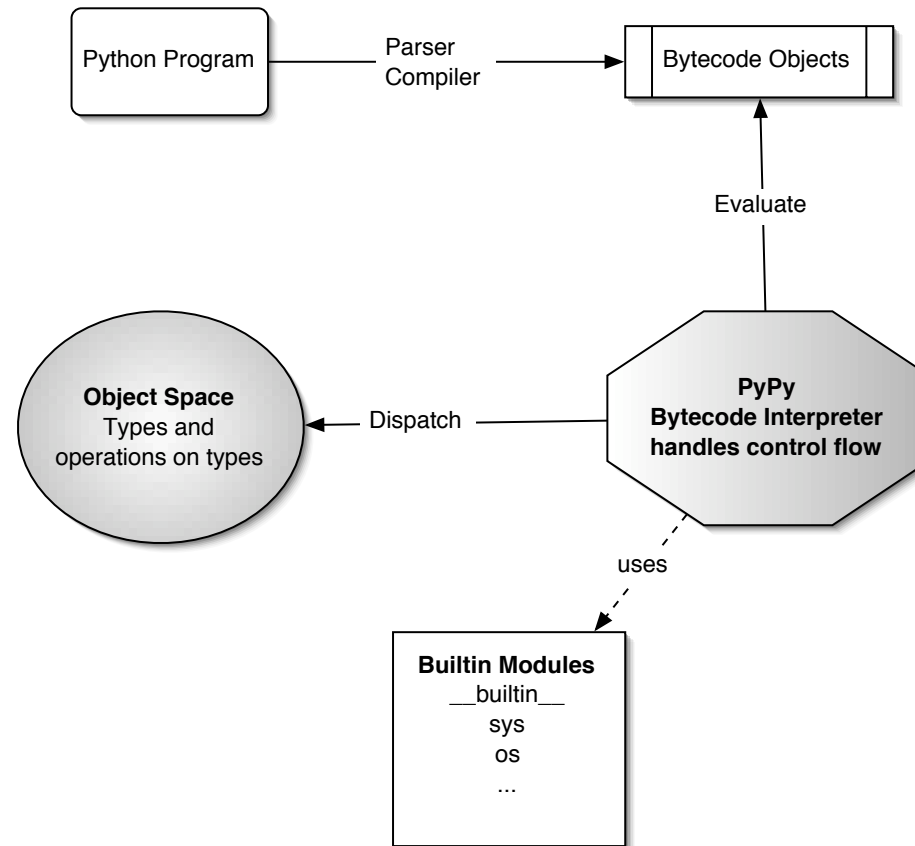
- 2008 Post EU pause
- 2009 2. JIT, nye GC's
- 2010 3. JIT, CPYext, [speed.pypy.org](http://speed.pypy.org)
- 2011 optimeringer

# PyPy filosofi

- Hvis Python kun er det næstbedste sprog, hvordan kan vi gøre det bedre?
- Tests, Tests, Tests
- Afkobling af afhængigheder
- Brug options

# Arkitektur

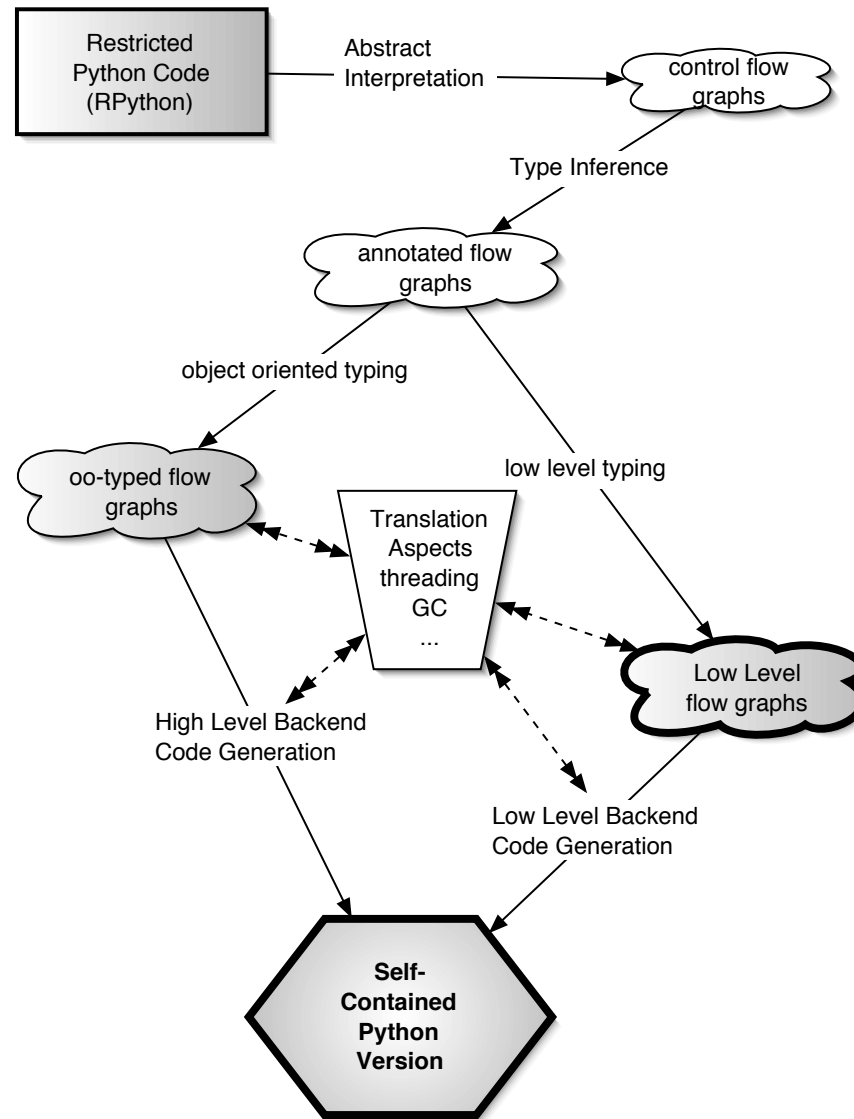
## PyPy Python Implementation Architecture



# Translation tool chain

- Control flow via FlowObjSpace
- Annotation (type inference)
- rtyping
- Garbage Collection, stackcheck, allocation removal, JIT generation
- Platform dependent code generation

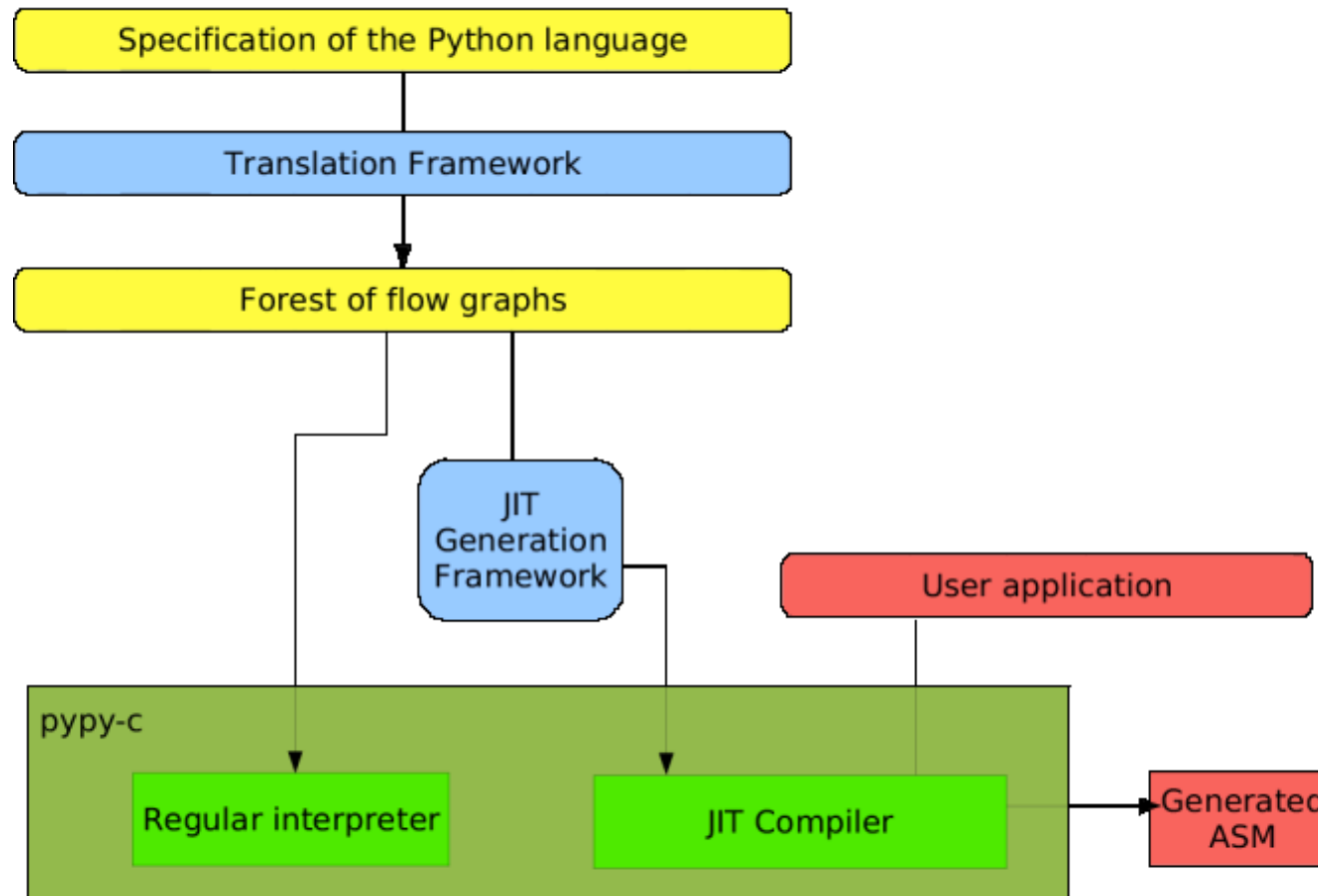
# PyPy Translation Architecture



# RPython

- “Restricted Python”
- Defineret ved hvad, der kan oversættes
- Til ting der ikke skal oversættes, kan hele Python bruges (initialisering)
- Kun til implementering af fortolkere

# PyPy JIT



# PyPy JIT

- Tracing JIT
- Fortolker bytecode
- Finder løkker
- konverterer “Hot loops” til maskinkode

# PyPy JIT

- Find løkker
  - Ved hjælp af “hints”
  - Løkkerne er “hot” hvis JIT compileren ser løkken mange gange

# Problemer med JIT's

- Warmup time
- tracelength

# PyPy MultiJIT

- Specielle JIT's
  - En generel for fortolkeren
  - map
  - Regular expressions
  - Numpy

# Optimeringer

- Loop unroll
- LICM
- Virtuals

# PyPy JIT

- Virker på hele fortolkeren (og dermed hele Python)
- Er ikke bundet til Python
-

# Resultater

- Speed.pypy.org
- MyHDL
- Quora
- Video processing

# JIT Viewer

- Værktøj til at forstå hvad JIT compileren gør

# Potentielle overraskelser

- Husk at luk filer
- Skriv programmerne “idiomatisk” (Zen of Python)
- Premature optimisation is the root of evil

# Fremtid

- Numpy
- More jit backends (Arm, ppc, Mips)
- STM (kill the GIL)
- Python 3
- ...

# Konklusion

- PyPy tilbyder en fleksibel platform til udvikling af fortolkere
- Forskellige GC, JIT, stackcheck, flere forskellige backends - Gratis
- En hurtig Python og kompatibel fortolker

# Konklusion

## (fortsat)

- [speed.pypy.org](http://speed.pypy.org)
- blogpost

# Konklusion

## (fortsat)

- Python (i form af PyPy) kan nogen gange være hurtigere end C
  - og (næsten) altid meget hurtigere end Cpython
- Med PyPy kan få både kort udviklingstid og hurtig eksekvering

# Links

- [pypy.org](http://pypy.org)
- [speed.pypy.org](http://speed.pypy.org)
- [bitbucket.org/pypy](http://bitbucket.org/pypy)
- [#pypy](https://freenode.org/#pypy) on freenode.org
- [anders@hih.au.dk](mailto:anders@hih.au.dk)

# Spørgsmål

# Interessante hjørner i PyPy

- Sandbox
- stackless