PyPy status talk

Maciej Fijalkowski
Merlinux GmbH

Politechnika Wroclawska

January 22 2009

What this talk is about?

- general overview of dynamic languages vm
- example: python
- challenges of classic approach
- possible solution - pypy

Dynamic languages VM's

- written in lower level language (C, Java)
- usually hard coded design decisions (ie GC, object layout, threading model)
- hard to maintain
- a challenge between performance and maintainability

Example - python

- primary implementation - CPython
- written in C
- hard-coded - Global Interpreter Lock
- hard-coded - refcounting for garbage collection
- psyco - very hard to maintain

Example - python (2)

- Jython, IronPython - bound to a specific VM
- about the same performance as CPython
- Java is still not the best language ever
- both are compilers, harder to maintain

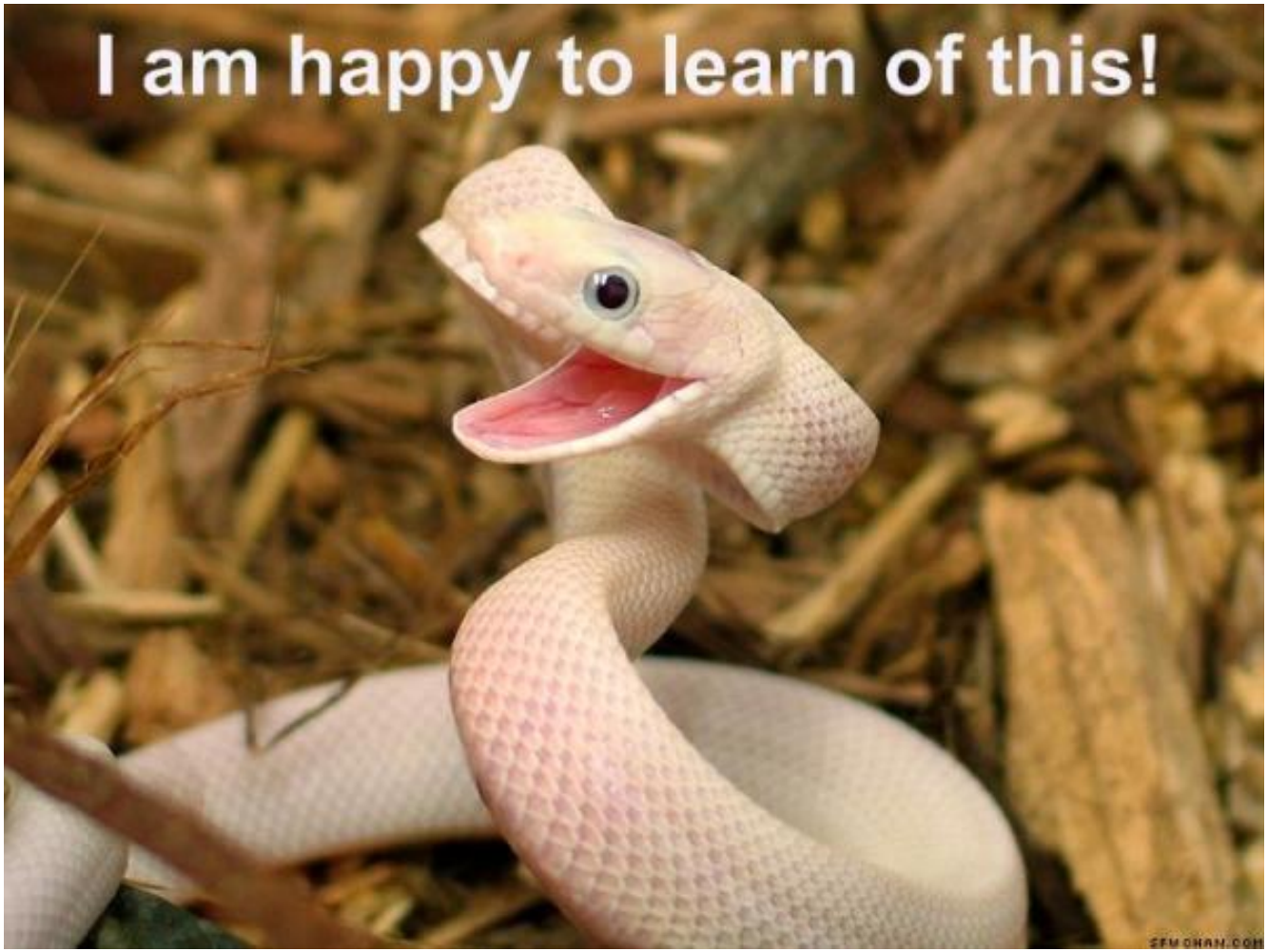
Ideally, we would ...

- use a high level language, to describe an interpreter
- get performance by dynamic compilation
- separate language semantics from design decisions

n*m*l problem

- n - dynamic languages
- m - design decisions (GC, JIT, etc.)
- l - platforms (JVM, .NET, C/Posix)
- we want an `n+m+l` effort, instead of `n*m*l` !

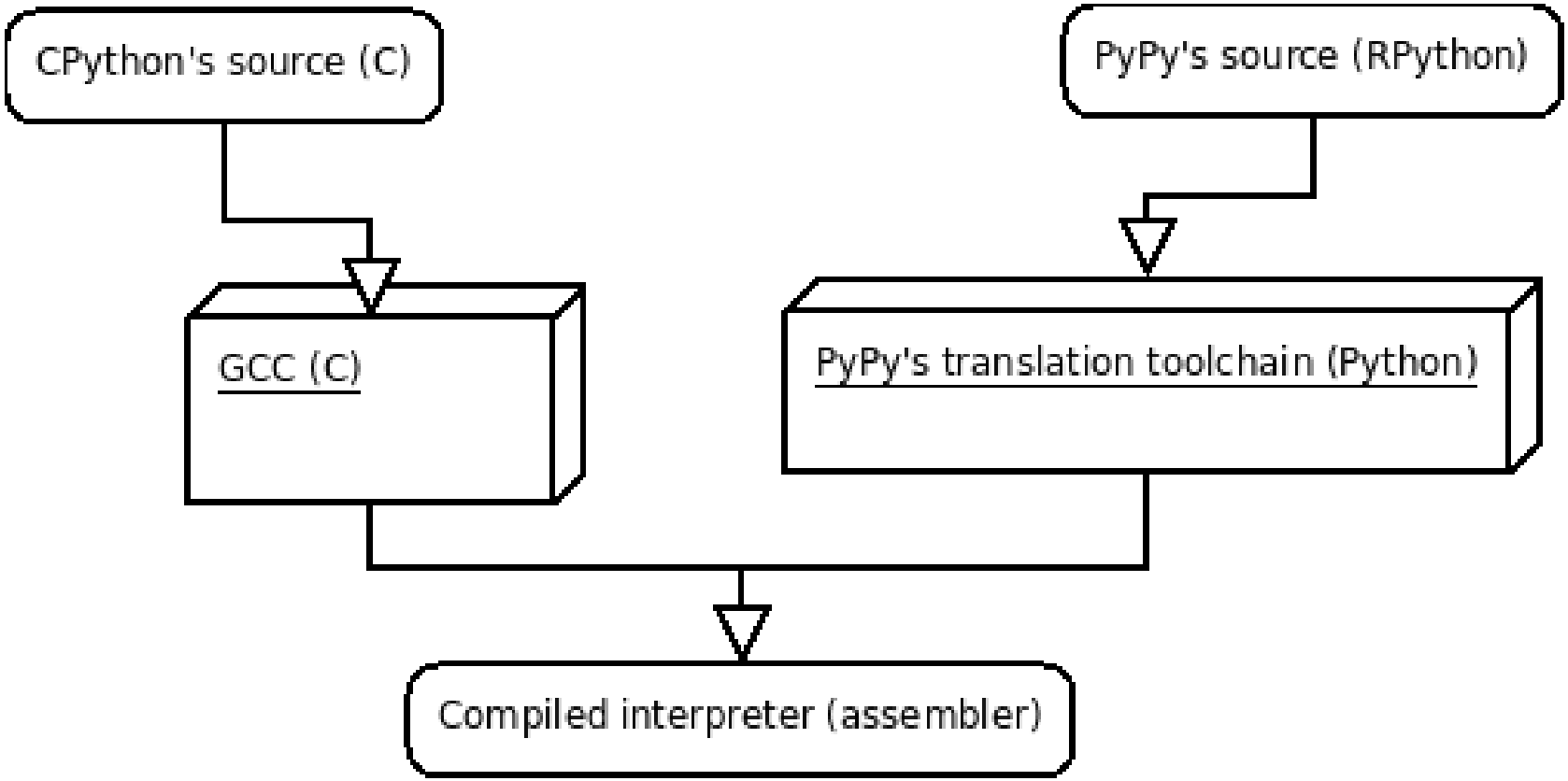
Happy snakes



PyPy - high level goals

- solve n*m*l problem
- create a nice, fast and maintainable python implementation
- that runs on C/Posix, .NET, JVM, whatever
- with JIT

PyPy - high level architecture



PyPy - implementation language

- RPython - restricted susbset of Python
- but still a valid python
- static enough to compile to efficient code
- not necesarilly nice language
- ... but better than C