A JIT Compiler (almost) for Free

Carl
Friedrich
Bolz

Laurence
Tratt

KING'S College LONDON

Software Development Team

ECOOP 2016 Summer School

Overview

An Alternate Approach to VM Construction

- Most productions VMs are written by hand in C/C++
- VMs are very tedious and costly to write
- Particularly for complex dynamically typed languages

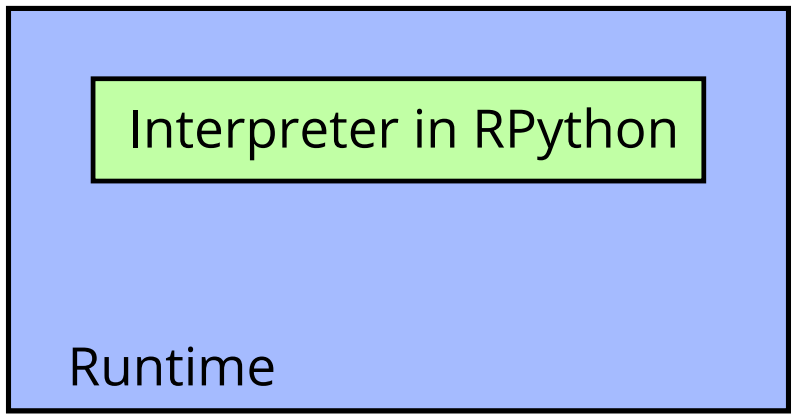
Approach: Separation of Concerns

Separate the following VM implementation concerns:

- Language semantics
- Generic JIT compilation issues
- Generic optimizations
- Language-specific optimizations

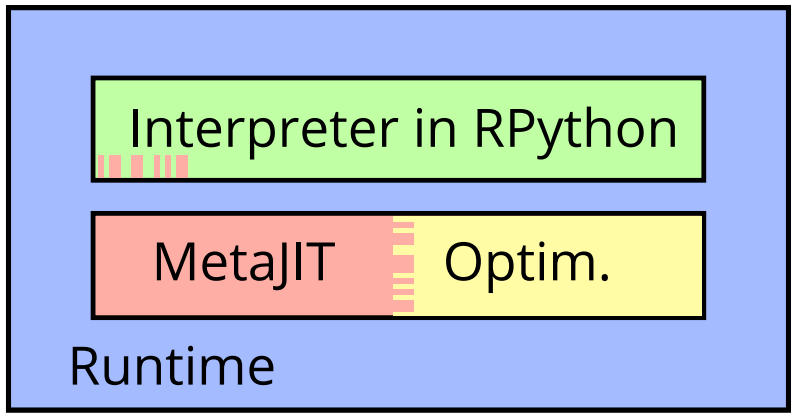
The RPython Project and PyPy

- RPython is a language to implement interpreters
- Interpreters are translated into C-based VMs
- Various extra features are added, e.g. GC
- Most mature interpreter is PyPy: Python in RPython
- Long-running project, many contributors



A Meta-Tracing JIT for RPython

- Apply the meta-tracing approach to RPython
- Insert meta JIT into the generated VM
- Contains generic JIT infrastructure: backends, integration, GC
- Needs some hints from the interpreter author
- Hints specify the main dispatch loop and the program counter
- Typical optimizations, plus some specific ones

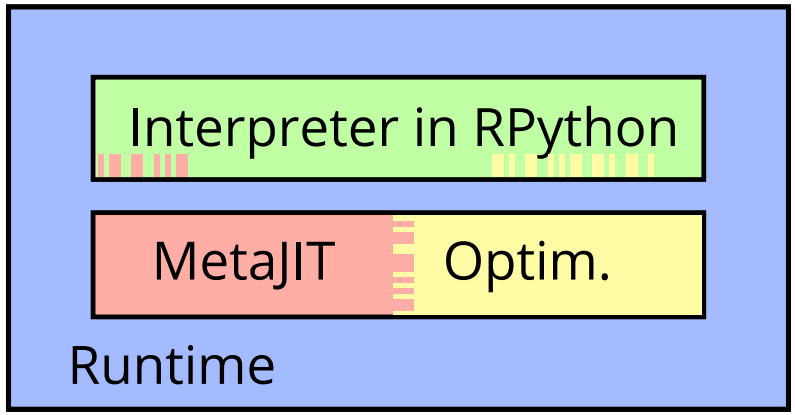


PyPy, an RPython VM for Python 

- How the RPython project got started.
- Very compatible and fast implementation of Python 2.7.
- Big developer community with various interests.
- Around 60K LoC (interpreter) and 190K LoC (modules) of RPython code.

Summary

- Using the RPython JIT, most of the JIT infrastructure is shared between languages
- Only an interpreter and some hints are needed
- Can support languages as different as Prolog and Python
- Other languages: PHP, Ruby, Smalltalk, Racket, SQLite, CPU emulators, ...
- “Gives you 80% of a great JIT 20% of the effort”



Hands-On

Goal: Write a Small Language With JIT

- Write an interpreter for a small stack-based language in RPython
- We give you some scaffolding and tests
- Add JIT hints
- ...
- performance

RPython

- “Restricted Python“
- Does type inference on Python programs to translate to C
- Feels like Python a lot of the time

RPython

- “Restricted Python“
- Does type inference on Python programs to translate to C
- Feels like Python a lot of the time
- Except when it doesn’t

RPython

- “Restricted Python“
- Does type inference on Python programs to translate to C
- Feels like Python a lot of the time
- Except when it doesn’t
- All variables need a fixed type (`a = 1; a = 'x'` forbidden)
- Can use built-in data types (ints, floats, strings, lists, dicts)
- Single inheritance classes, methods, polymorphism, type checks
- Think Java type system without generics

Practicalities

- Download the files: http://cfbolz.de/stuff/tla.zip
- Install PyPy dependencies
- Run tests using `pytest`, included in the checkout:
- `pypy/pytest.py test_tla.py`

Translating to C

- When you pass some tests, you can try to translate to C
- `python pypy/rpython/bin/rpython -O2 targettla.py`
- there's a bug in the `*.tla.py` files: replace `RETURN` with `EXIT`
- assemble the file: `PYTHONPATH=...../pypy/ python tla_assembler.py loopabit.tla.py`
- afterwards, run the C version of the interpreter:
- `./targettla-c loopabit.tla 1000`

Adding a JIT

- Need to add a call to `jitdriver.jit_merge_point` to the bytecode dispatch loop
- Then run the last test, which is emulating the JIT
- Translate with JIT:
- `pypy/rpython/bin/rpython -Ojit targettla.py`

Viewing Traces after Translation

- binary will produce log files if you set an environment variable
- `PYPYLOG=jit-log-opt:outfile targettla-c ...`
- inspect `outfile` afterwards
- if you want to compare against not having a jit, run `targettla-c -jit off`

More ideas

The are various extensions possible to the language:

- adding more datatypes, like strings and floats
- adding function calls
- adding objects