RPython

Leonard de HARO

June 11, 2012

The Pypy Project

- New Python VM, written in RPython

The Pypy Project

- New Python VM, written in RPython
- New language to design VMs: RPython

The Pypy Project

- New Python VM, written in RPython
  The Interpreter
- New language to design VMs: RPython
  The Translation Toolchain

The Pypy Project

- New Python VM, written in RPython
  The Interpreter
- New language to design VMs: RPython
  The Translation Toolchain

Fact

The Translation Toolchain provides a JIT compiler on demand!

Sure... But what's a JIT compiler?

Two kinds:

- Method JITs
  e.g. HotSpot in JVM)
- Tracing JITs
  e.g. Pypy

Method JITs

- Works on the bytecode (linear)

Method JITs

- Works on the bytecode (linear)
- Notices ”Hot Spots”

Method JITs

- Works on the bytecode (linear)
- Notices ”Hot Spots”
- Compiles them (native code)

Method JITs

- Works on the bytecode (linear)
- Notices ”Hot Spots”
- Compiles them (native code)
- Uses the native code version

Tracing JIT

- Works during execution

Tracing JIT

- Works during execution
- Finds a ”hot loop”

Tracing JIT

- Works during execution
- Finds a ”hot loop”
- Traces the execution

Tracing JIT

- Works during execution
- Finds a ”hot loop”
- Traces the execution
- Optimizes it (including guards)

Tracing JIT

- Works during execution
- Finds a ”hot loop”
- Traces the execution
- Optimizes it (including guards)
- Uses the optimized traced version

Tracing JIT

- Works during execution
- Finds a ”hot loop”
- Traces the execution
- Optimizes it (including guards)
- Uses the optimized traced version

Fact

*Pypy is a **meta-tracing JIT** compiler: feed it a properly annotated interpreter, it gives you back a tracing JIT interpreter.*

Question

Examples of Pypy's JIT all work with bytecode. Can we make it work on ASTs?

General properties of RPython

- Strict and valid subset of Python

General properties of RPython

- Strict and valid subset of Python
- Statically typed (with exception)

General properties of RPython

- Strict and valid subset of Python
- Statically typed (with exception)
- Output C (when used in TT)

General properties of RPython

- Strict and valid subset of Python
- Statically typed (with exception)
- Output C (when used in TT)
- **Creates JITing VMs** (for under 10 lines of code)

General properties of RPython

- Strict and valid subset of Python
- Statically typed (with exception)
- Output C (when used in TT)
- **Creates JITing VMs** (for under 10 lines of code)
- Still in development although useable

Writing a JIT VM

- Write your interpreter

Writing a JIT VM

- Write your interpreter
- Add RPython instructions for translation

Writing a JIT VM

- Write your interpreter
- Add RPython instructions for translation
- Add RPython instructions for JITing

Writing a JIT VM

- Write your interpreter
- Add RPython instructions for translation
- Add RPython instructions for JITing
  - `can_enter_jit`

Writing a JIT VM

- Write your interpreter
- Add RPython instructions for translation
- Add RPython instructions for JITing
  - `can_enter_jit`
  - `jit_merge_point`

Writing a JIT VM

- Write your interpreter
- Add RPython instructions for translation
- Add RPython instructions for JITing
  - `can_enter_jit`
  - `jit_merge_point`
  - Declare *Red* and *Green* variables

Writing a JIT VM

- Write your interpreter
- Add RPython instructions for translation
- Add RPython instructions for JITing
  - `can_enter_jit`
  - `jit_merge_point`
  - Declare *Red* and *Green* variables
- Optimize (*e.g.* insert *assert* to help the interpreter or use fixed-size lists)

Demo Time !
(see Andrew Brown's tutorial on Pypy's Blog)