HippyVM - yet another attempt at PHP performance

Maciej Fijalkowski

September 19, 2014

Who am I?

- Maciej Fijalkowski
- PyPy core developer for about 8 years
- main author of HippyVM
- founder of baroquesoftware.com

This talk

- HippyVM project
- performance and measurments
- a bit about PyPy, HippyVM and performance oriented virtual machines

HippyVM

- a PHP interpreter (essentially), runs php code
- has a **just in time compiler**, which makes it fast
- dual licensing, commercial and open source

HippyVM

- a PHP interpreter (essentially), runs php code
- has a **just in time compiler**, which makes it fast
- dual licensing, commercial and open source

HippyVM

- a PHP interpreter (essentially), runs php code
- has a **just in time compiler**, which makes it fast
- dual licensing, commercial and open source

HippyVM status

- runs a lot of PHP test suite (over half)
- misses a lot of builtin functions
- can kind of "run" wordpress, mediawiki, squirrelmail, getting there
- cgi, fastcgi (not open source)
- performance oriented

HippyVM demo

- famous fibonacci
- fannkuch
- richards
- bigger stuff*

HippyVM status - short

- fast, compliant
- not quite ready

HippyVM - Python bridge

- demo

Let’s talk about performance

- "I want my website to run faster"
- "I’m going to compare various languages, implementations"
- "I’ll use a recursive fibonacci function"

Let’s talk about performance

- "I want my website to run faster"
- "I’m going to compare various languages, implementations"
- "I’ll use a recursive fibonacci function"

Performance - breakdown

- user code
- runtime
- IO, DB, ... - libraries and ecosystem
- non-trivial interactions between the pieces

- I can't guess

Performance - breakdown

- user code
- runtime
- IO, DB, ... - libraries and ecosystem
- non-trivial interactions between the pieces

- I can't guess

Performance

- a very complex picture
- bottlenecks depend on the use case
- libraries, programmers, styles matter

Performance - let's try science

- get a set of programs (small, medium, large)
- compare them
- don't use a single number

Bad benchmarking - example 1

- benchmarks game
- limited set of benchmarks
- limited set of bottlenecks
- relentlessly optimized

Bad benchmarking - example 2

- pystone
- the idea - measure time consumed by each operation
- should give you the idea how the program will operate

- it's wrong

Bad benchmarking - example 2

- pystone
- the idea - measure time consumed by each operation
- should give you the idea how the program will operate

- it's wrong

Why it's wrong?

- the interaction is non-trivial
- e.g. garbage collection
- Python 2.6 vs 2.7, "minor" improvement in the GC
- PyPy translation toolchain takes 1h instead of 3h

Good benchmarking - example

- speed.pypy.org
- a decent set of small medium and large benchmarks
- I strongly encourage people to come up with the same for PHP

PHP performance

- number of requests per second
- loading the code
- accessing the DB
- gluing it all together

PHP performance landscape

- Zend PHP
- HHVM
- php-ng
- HippyVM
- other projects

Current HippyVM performance

- we can't really compare large things (mediawiki ~on par)
- small and medium between 2x faster - 2x slower than HHVM
- very hand-wavy, but you really need to do it yourself
- consider bottlenecks differ depending on implementation

Current HippyVM performance

- we can't really compare large things (mediawiki ~on par)
- small and medium between 2x faster - 2x slower than HHVM
- very hand-wavy, but you really need to do it yourself
- consider bottlenecks differ depending on implementation

Current HippyVM performance

- we can't really compare large things (mediawiki ~on par)
- small and medium between 2x faster - 2x slower than HHVM
- very hand-wavy, but you really need to do it yourself
- consider bottlenecks differ depending on implementation

PHP performance problems

- low performance of the reference implementation
- relative immaturity of more advanced implementations
- reload all the code creates problems
- the language quirks can be worked around
- in my opinion, dynamism, etc. does not matter

PHP performance problems

- low performance of the reference implementation
- relative immaturity of more advanced implementations
- reload all the code creates problems

- the language quirks can be worked around

- in my opinion, dynamism, etc. does not matter

PHP performance problems

- low performance of the reference implementation
- relative immaturity of more advanced implementations
- reload all the code creates problems

- the language quirks can be worked around

- in my opinion, dynamism, etc. does not matter

Performance - personal opinions

- the language should be easy **for a programmer**
- we, as a PyPy team, never discuss language design
- the language implementation can be complex
- libraries, patterns and the ecosystem matter for anything non-trivial

HippyVM history

- started as a facebook research project
- got some funding to pursue as a commercial project
- an offspin of PyPy technology

PyPy

- a fast, compliant Python interpreter
- 0.5%-1% of Python market share
- a framework for building efficient interpreters
- 10 years of research
- fully Open Source, EU funded project

Let's go back 10 years

- Python is (like PHP) a very complex language
- writing an interpreter is hard
- writing a just-in-time compiler is even harder
- we decided to write a framework instead

Just in time compiler?

- "lower level" languages, like C, have compilers that turn C into assembler
- "higher level" have interpreters that run a virtual machine
- just in time is a hybrid - run in an interpreter, compile to assembler on the fly

Why not a compiler?

- "Just compile X to Lisp/C/JavaScript, it'll be fast!"
- mismatch in semantics
- hard-to-prove edge cases
- ends up not being that fast

Why not a compiler?

- "Just compile X to Lisp/C/JavaScript, it'll be fast!"
- mismatch in semantics
- hard-to-prove edge cases
- ends up not being that fast

Just in time - benefits

- you can choose what to compile
- you can ignore the edge cases (but have a way to deoptimize)
- guesses are a lot easier
- a lot of things fall in naturally, e.g. late imports

A simple example

- an interpreter written in C/C++
- e.g. CPython, Zend PHP, MRI Ruby, ...

A typical more advanced example

- interpreter, written in C/C++
- just in time compiler that repeats the semantics, written in C/C++, emits assembler
- another layer, e.g. a method JIT

- becomes harder and harder to keep up with semantics
- examples: V8, HHVM, *monkey

A typical more advanced example

- interpreter, written in C/C++
- just in time compiler that repeats the semantics, written in C/C++, emits assembler
- another layer, e.g. a method JIT

- becomes harder and harder to keep up with semantics
- examples: V8, HHVM, *monkey

PyPy approach

- write an interpreter in a machine-readable language
- just in time compiler gets generated from the description
- a lot of work once, but prevents a lot of problems

How well it works?

- HippyVM - 4 people, 1.5 years
- HHVM - 5 years, team of up to 20

- a lot of effort is reusable
- Truffle is another example of a similar approach

How well it works?

- HippyVM - 4 people, 1.5 years
- HHVM - 5 years, team of up to 20

- a lot of effort is reusable
- Truffle is another example of a similar approach

HippyVM - future

- find more funding
- run open source projects efficiently
- develop a benchmark suite

Questions?

- hippyvm.com
- baroquesoftware.com
- talk to me!