Australian National University

Sharpen Your Axe: A Micro Virtual Machine Which You Can Depend On

Kunshan Wang, Yi Lin

Supervisor: Steve Blackburn

Joint work with:	Antony Hosking	Purdue University
	Michael Norrish	NICTA, ANU

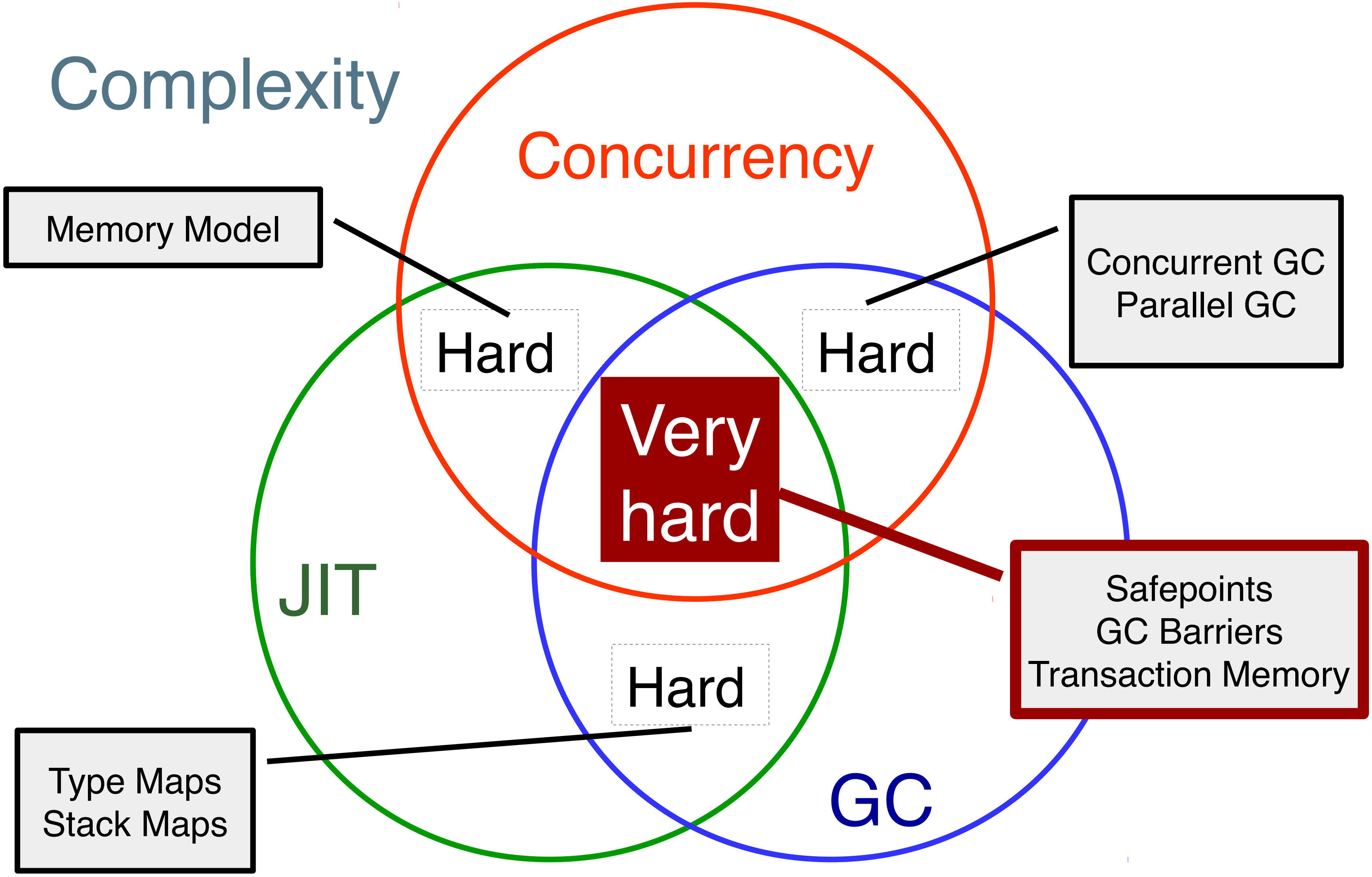
Modern Virtual Machines

Concurrency

Just-in-time Compiling

Garbage Collection

Complexity

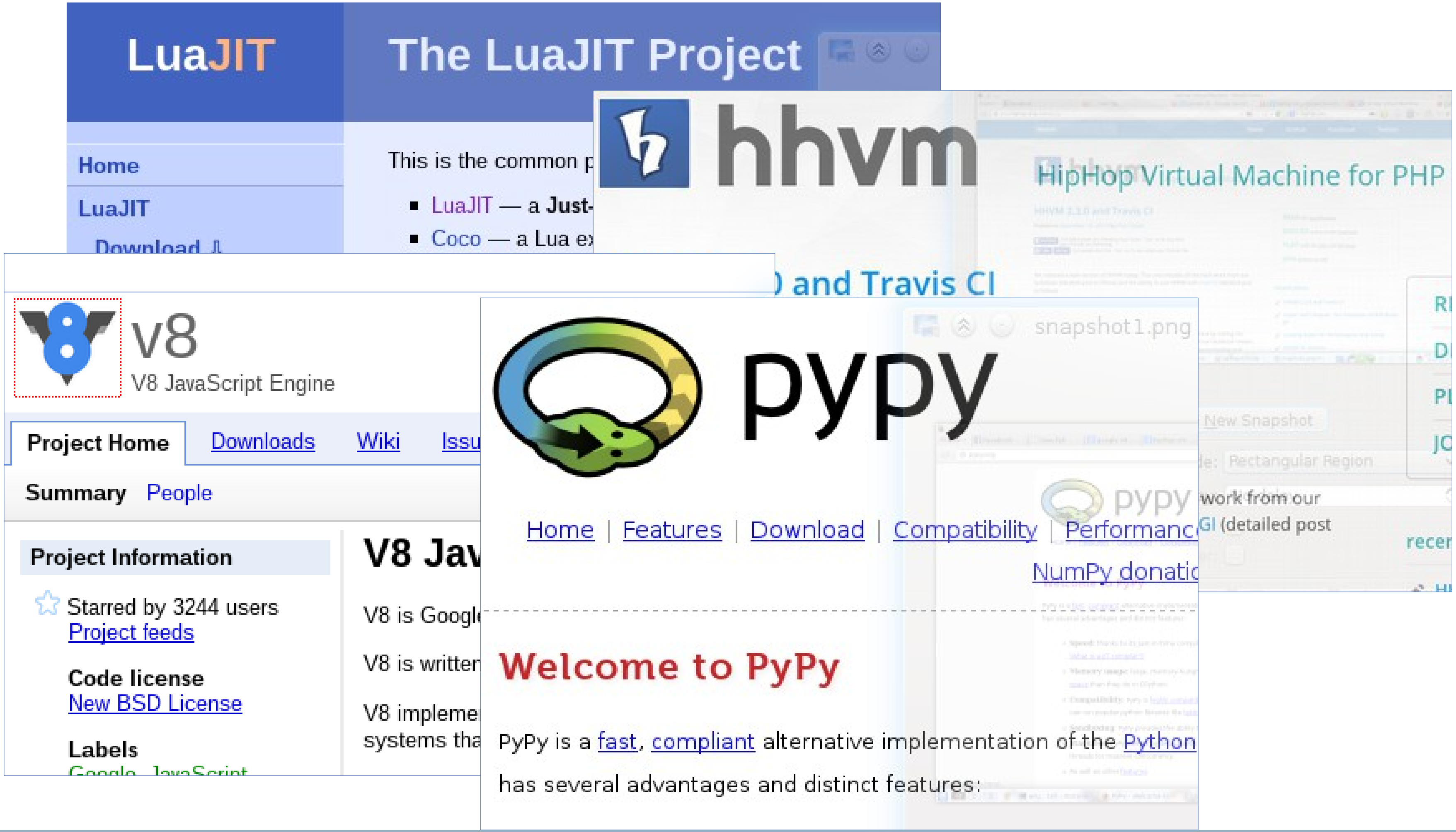

Concurrency
Memory Model
Hard
Hard
Concurrent GC
Parallel GC
Very hard
JIT
Safepoints
GC Barriers
Transaction Memory
Hard
Type Maps
Stack Maps
GC

List of Real-world VMs

Lang/Impl.	JIT Compiling	Concurrency	GC
CPython	Interpreted	GIL	Naive RC
PyPy	Tracing JIT	GIL	MMTK-like
Unladen Swallow	Template JIT	Same as CPython	Same as CPython
Jython	To JVM byte-code	JVM threading	JVM GC
PHP	Interpreted	?	Naive RC
PHP (HipHop)	Tracing JIT	?	Naive RC
Ruby (MRI)	Interpreted	GIL	Mark-sweep
Perl	Interpreted	?	Naive RC
Lua	Interpreted	No threading	Mark-sweep
LuaJIT	Tracing JIT	Same as Lua	Same as Lua

* Most implementations rewrite GC from scratch

People Re-inventing Wheels



LuaJIT

The LuaJIT Project

Home

LuaJIT

Download ⇩

This is the common p

- LuaJIT — a Just-
- Coco — a Lua ex

hhvm

HipHop Virtual Machine for PHP

) and Travis CI

v8

V8 JavaScript Engine

Project Home | Downloads | Wiki | Issu

Summary | People

Project Information

☆ Starred by 3244 users
Project feeds

Code license
New BSD License

Labels
Google, JavaScript

V8 Jav

V8 is Googl

V8 is writter

V8 impleme
systems tha

snapshot1.png

pypy

Home | Features | Download | Compatibility | Performanc
NumPy donatio

New Snapshot

de: Rectangular Region

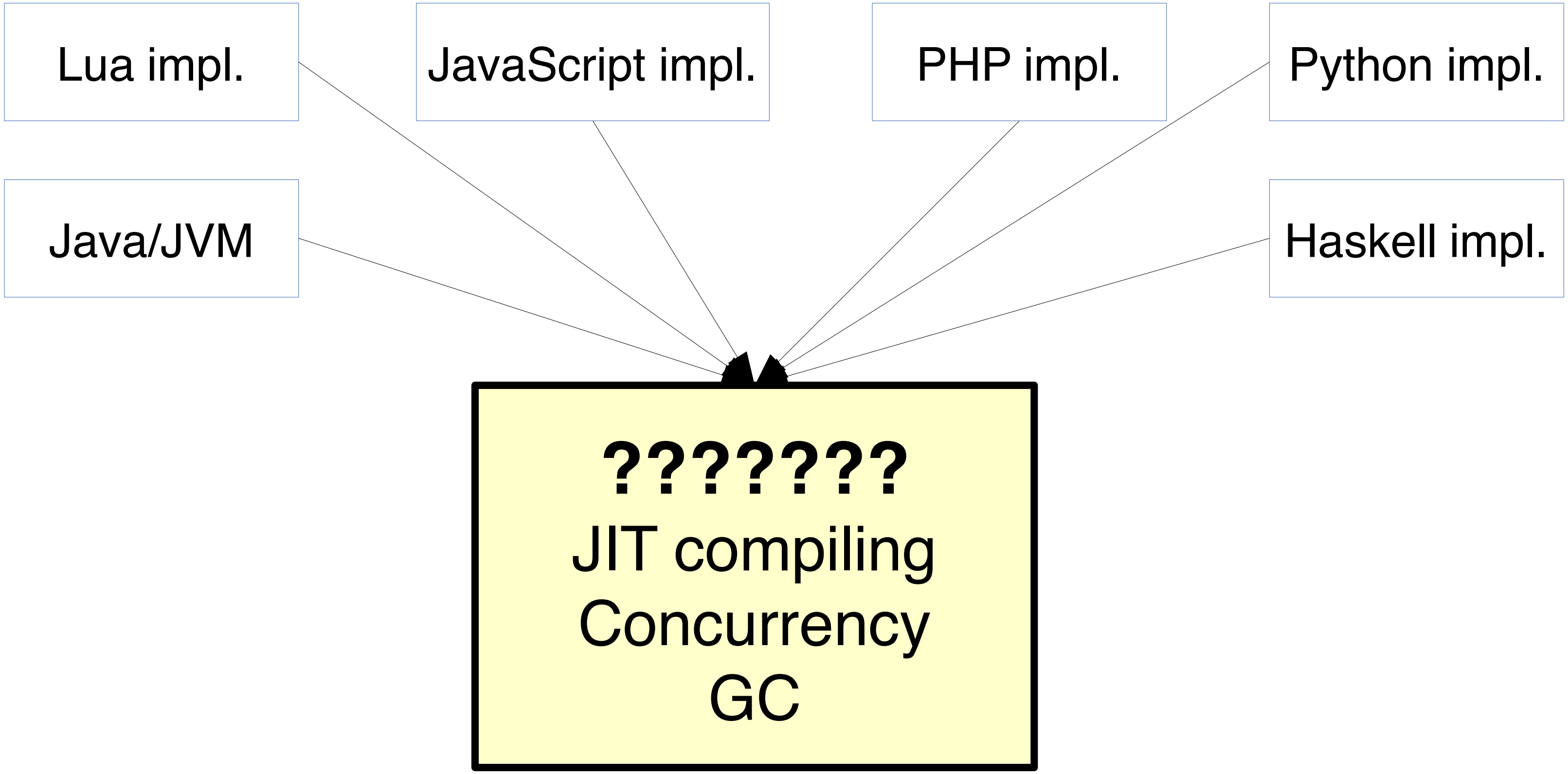
work from our
GI (detailed post

recen

Welcome to PyPy

PyPy is a fast, compliant alternative implementation of the Python
has several advantages and distinct features:

Do Not Re-invent the Wheel



Lua impl.

JavaScript impl.

PHP impl.

Python impl.

Java/JVM

Haskell impl.

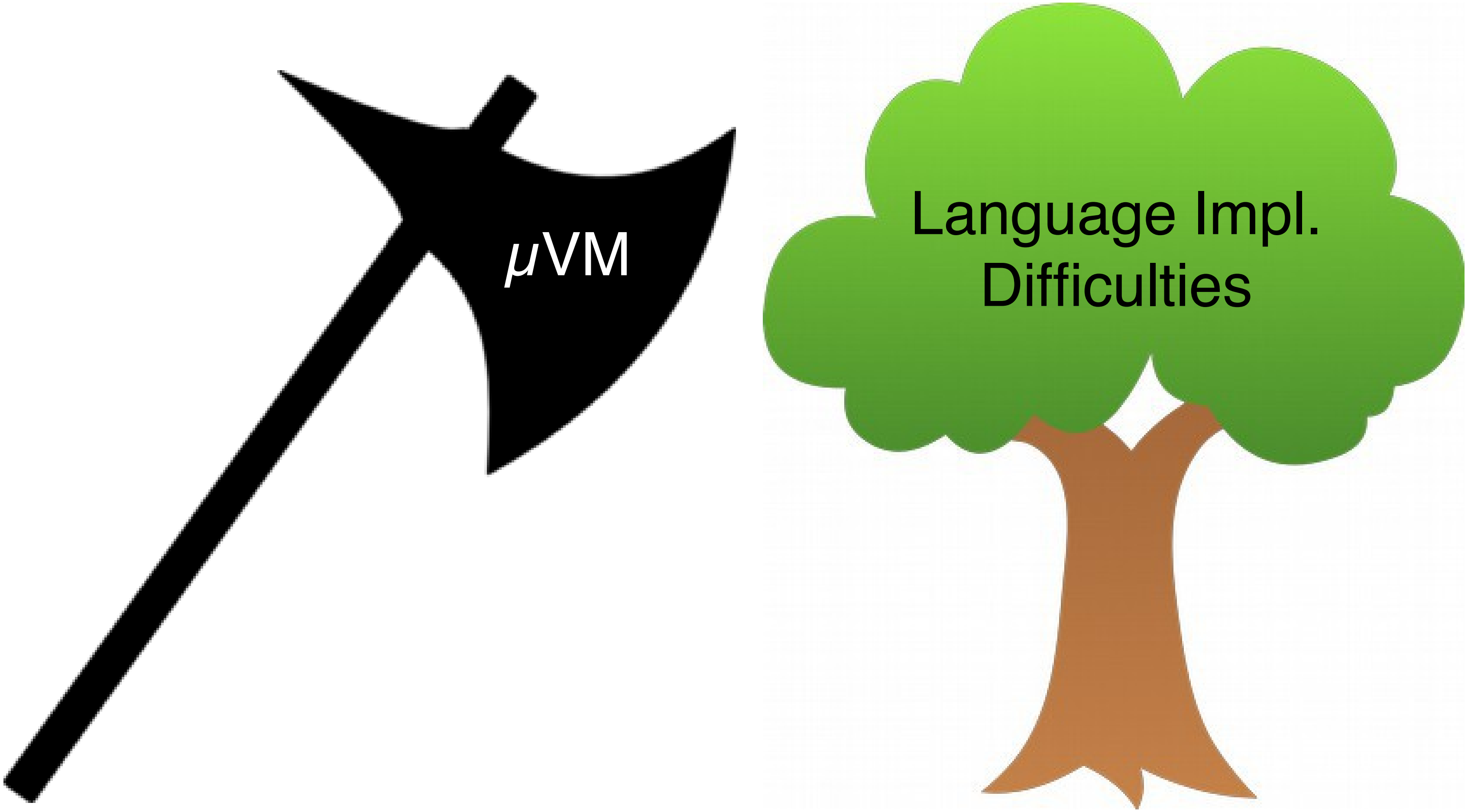
???????

JIT compiling

Concurrency

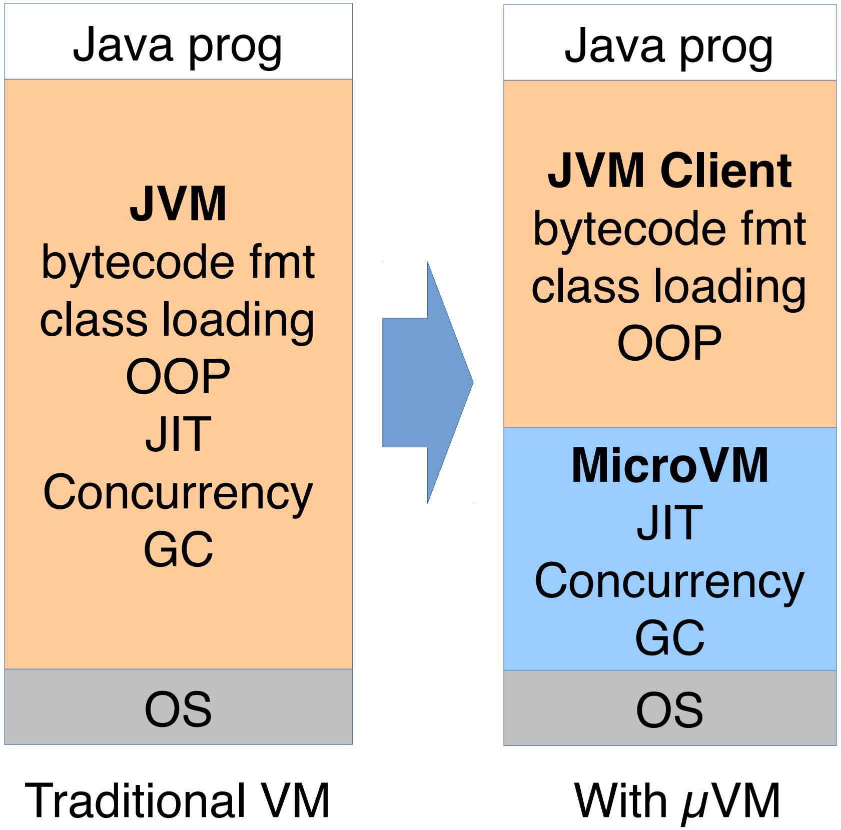
GC

Sharpening Your Axe
Does Not Delay Your Wood-cutting Job



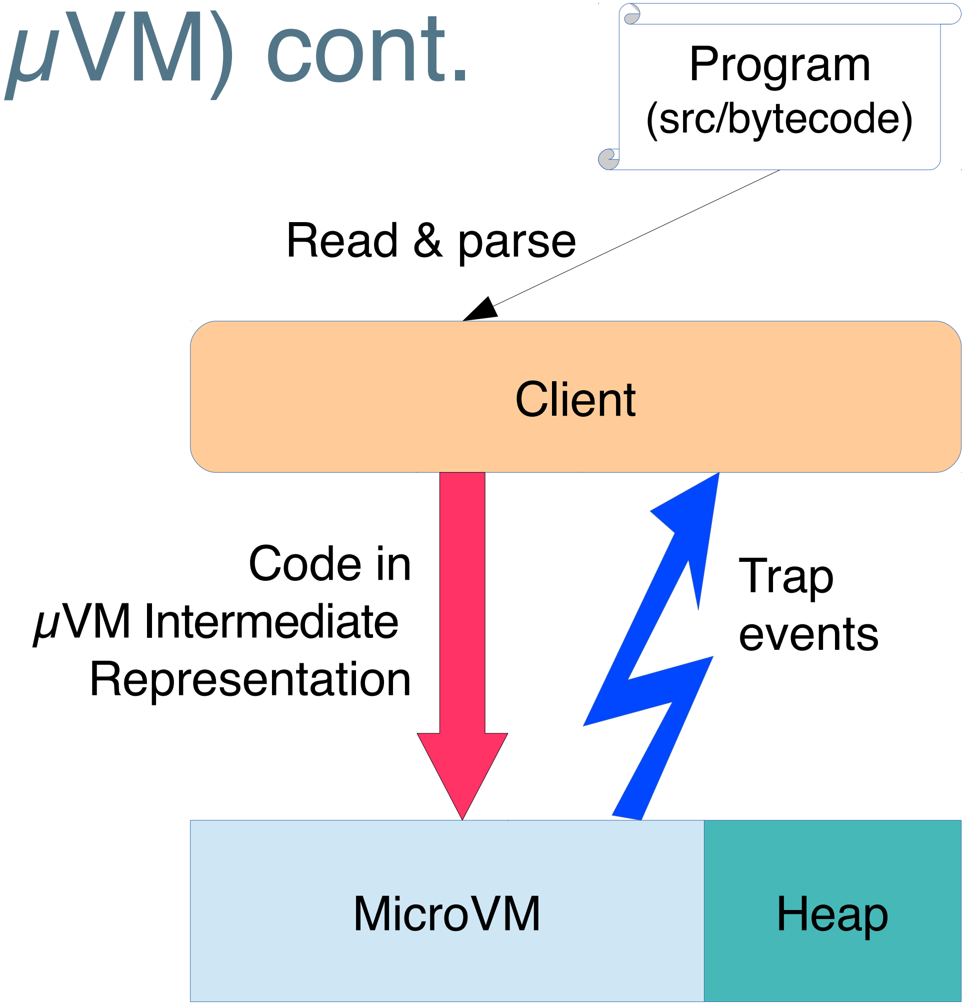
MicroVM (a.k.a. μVM)

- A very small VM
  - – JIT compiling
  - – Concurrency
  - – GC



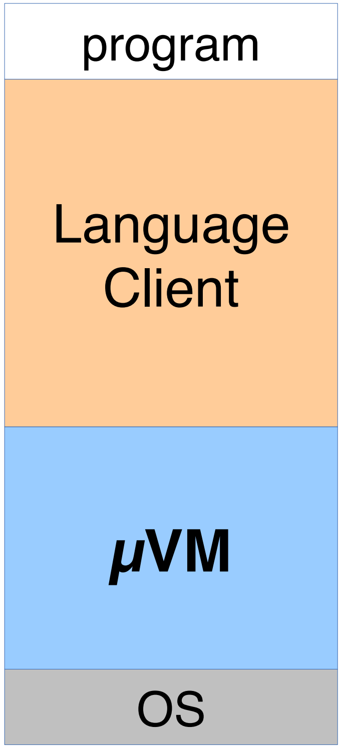
MicroVM (a.k.a. *μ*VM) cont.

- Client:
  - implements PL
  - delivers code
  - handles traps
- *μ*VM:
  - JIT compiles
  - manages heap
  - manages threads

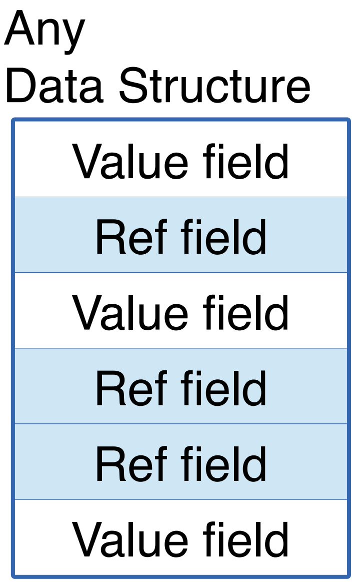


*μ*VM is a Low-level Abstraction

- *μ*VM is low-level.
- Machine-like data types:
  - Int(n), n=1,8,16,32,64, ...
  - Float(), Double(), Struct, Array
- Machine-like operations:
  - BinOp, Cmp, Select, conversions..



Garbage-collection



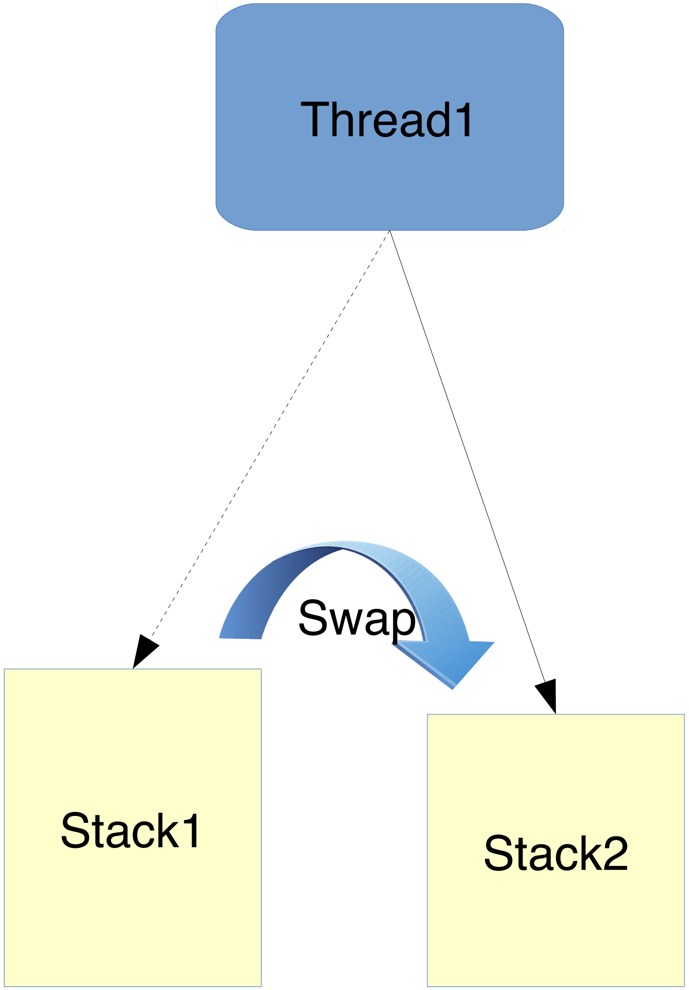
Any
Data Structure

Value field
Ref field
Value field
Ref field
Ref field
Value field

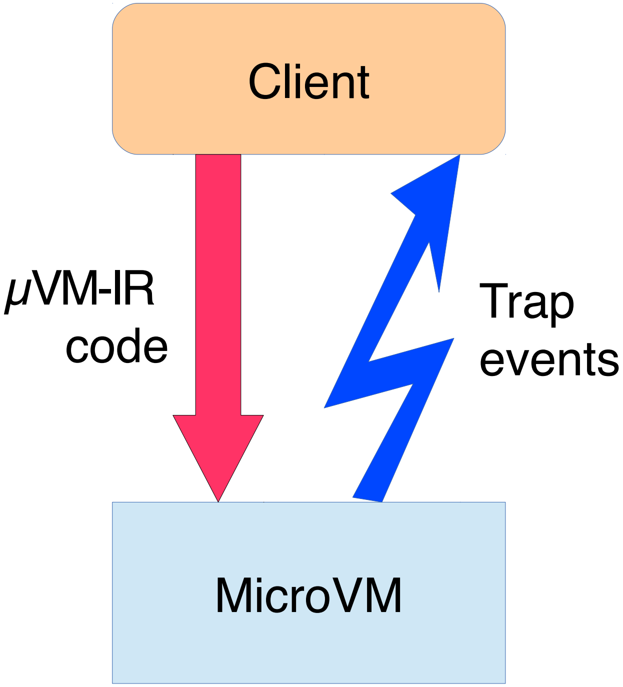
- References (no pointer)
  - And object field operations
- Precise GC
  - μVM can always find all references
- Easy for the Client
  - Depend on the μVM

Concurrency/Parallelism

- Thread and Stack primitives
- Memory Model and atomic operations
- SwapStack
  - Light-weighted threads
  - Coroutines



Designed for JIT-compiling



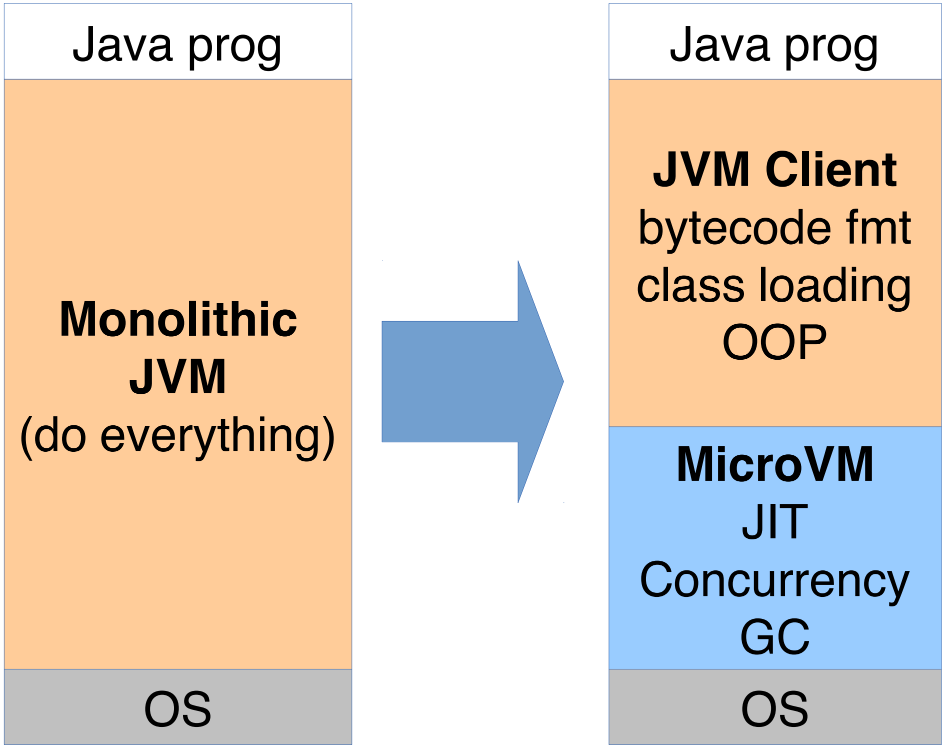
- Static single assignment (SSA)
- Run-time function redefinition
- Trap: way back to the Client
- On-stack Replacement

Status

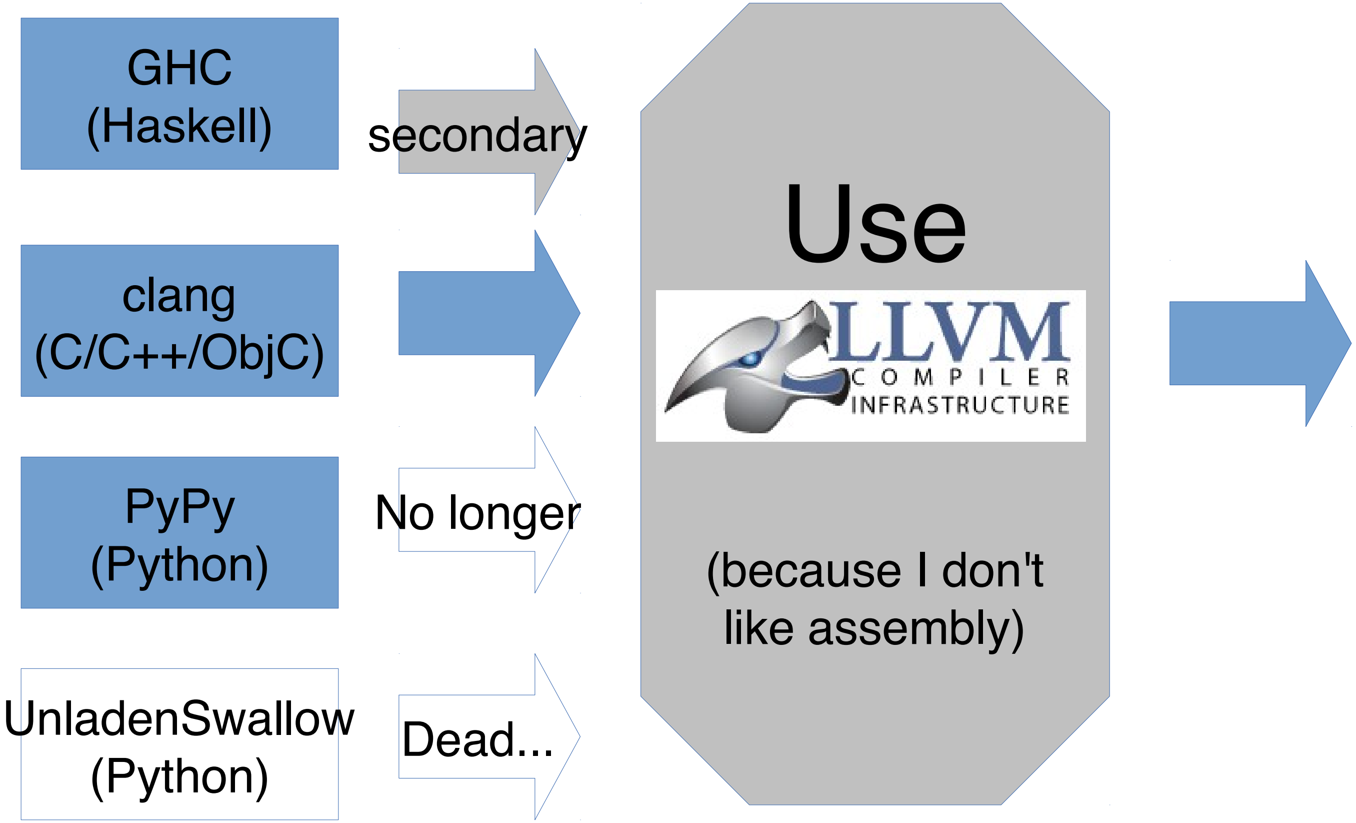
- Prototyping for functions and interfaces
  - Two prototypes in Python, two Lua clients
- Next milestone
  - Rewrite *μ*VM on RJava
  - And a RJava client on *μ*VM

Summary (THE END)

- MicroVM (a.k.a. *μ*VM) is a very small VM
    - JIT compiling
    - Concurrency
    - GC



Appendix1: LLVM is good, but must be used in the right way.



GHC (Haskell)

secondary

clang (C/C++/ObjC)

PyPy (Python)

No longer

UnladenSwallow (Python)

Dead...

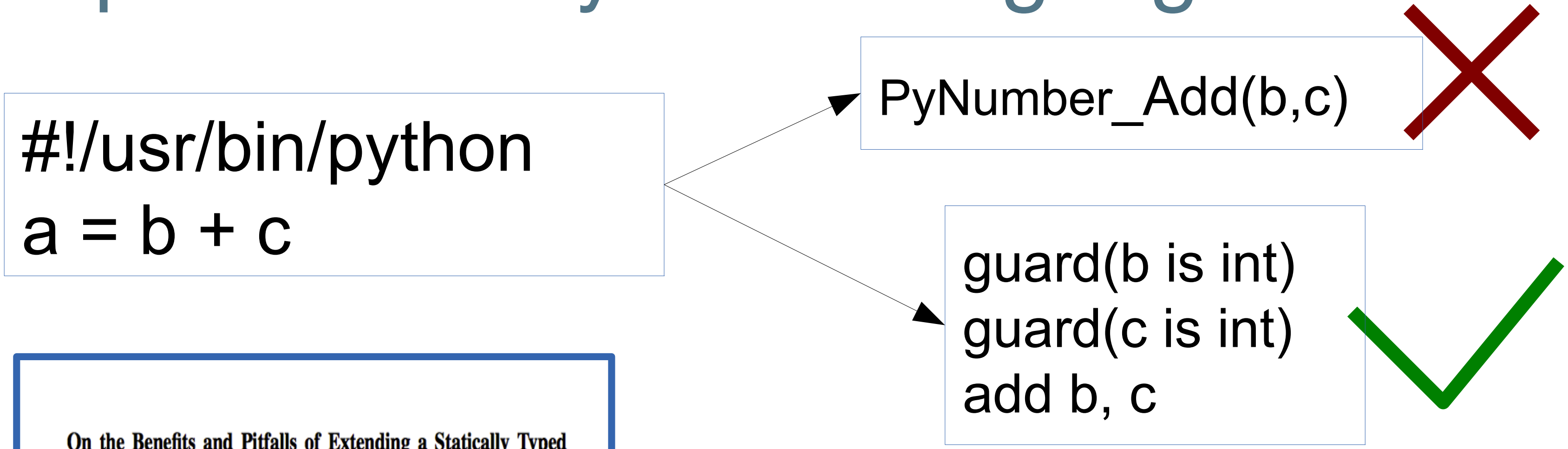
Use

LLVM COMPILER INFRASTRUCTURE

(because I don't like assembly)

It works, and we can still call it (reasonably) fast.

Appendex2: Type inference is important for dynamic languages.



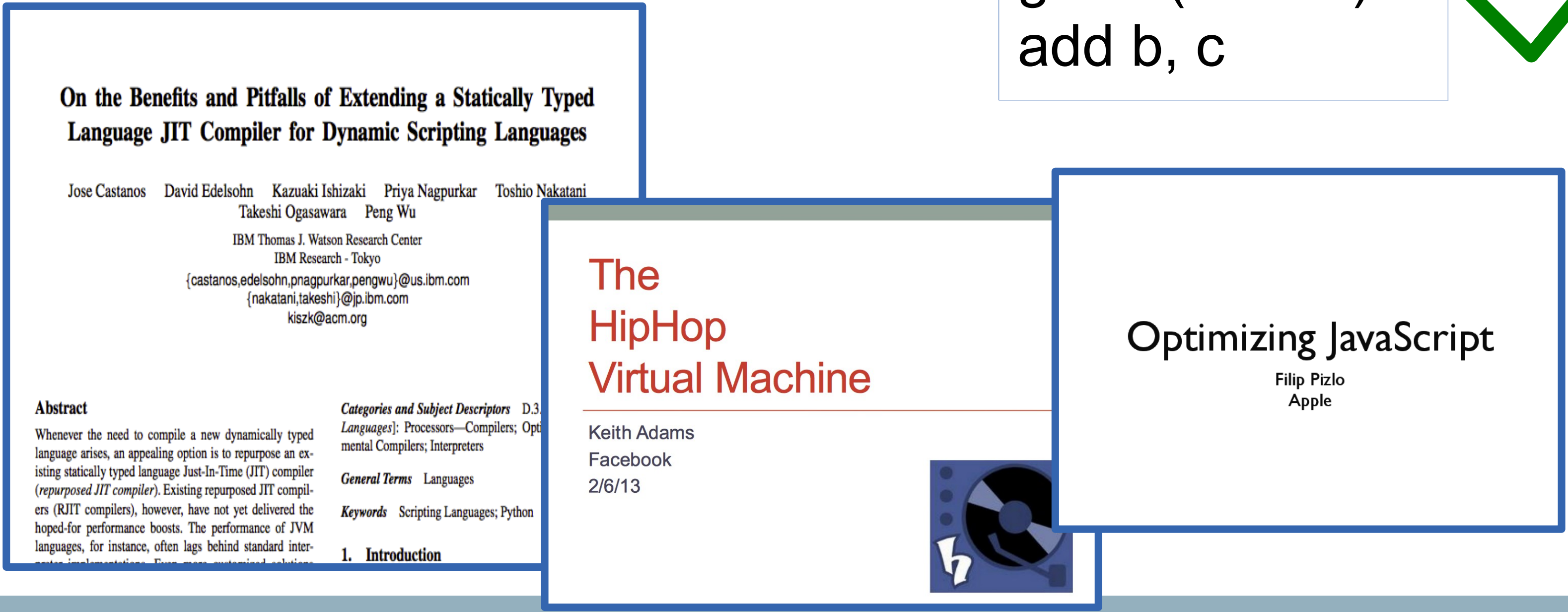
```
#!/usr/bin/python
a = b + c
```

```
PyNumber_Add(b,c)
```

✗

```
guard(b is int)
guard(c is int)
add b, c
```

✓



On the Benefits and Pitfalls of Extending a Statically Typed Language JIT Compiler for Dynamic Scripting Languages

Jose Castanos David Edelsohn Kazuaki Ishizaki Priya Nagpurkar Toshio Nakatani
Takeshi Ogasawara Peng Wu
IBM Thomas J. Watson Research Center
IBM Research - Tokyo
{castanos,edelsohn,pnagpurkar,pengwu}@us.ibm.com
{nakatani,takeshi}@jp.ibm.com
kiszk@acm.org

Abstract

Whenever the need to compile a new dynamically typed language arises, an appealing option is to repurpose an existing statically typed language Just-In-Time (JIT) compiler (*repurposed JIT compiler*). Existing repurposed JIT compilers (RJIT compilers), however, have not yet delivered the hoped-for performance boosts. The performance of JVM languages, for instance, often lags behind standard inter-

Categories and Subject Descriptors D.3 *Languages*]: Processors—Compilers; Opti mental Compilers; Interpreters

General Terms Languages

Keywords Scripting Languages; Python

1. Introduction

The HipHop Virtual Machine

Keith Adams
Facebook
2/6/13

Optimizing JavaScript

Filip Pizlo
Apple