PyCon TW 2016

Interaction between Python & Java

Feng Li (李枫)
hkli2013@126.com
Jun 3，2016

Content

I. Introduction

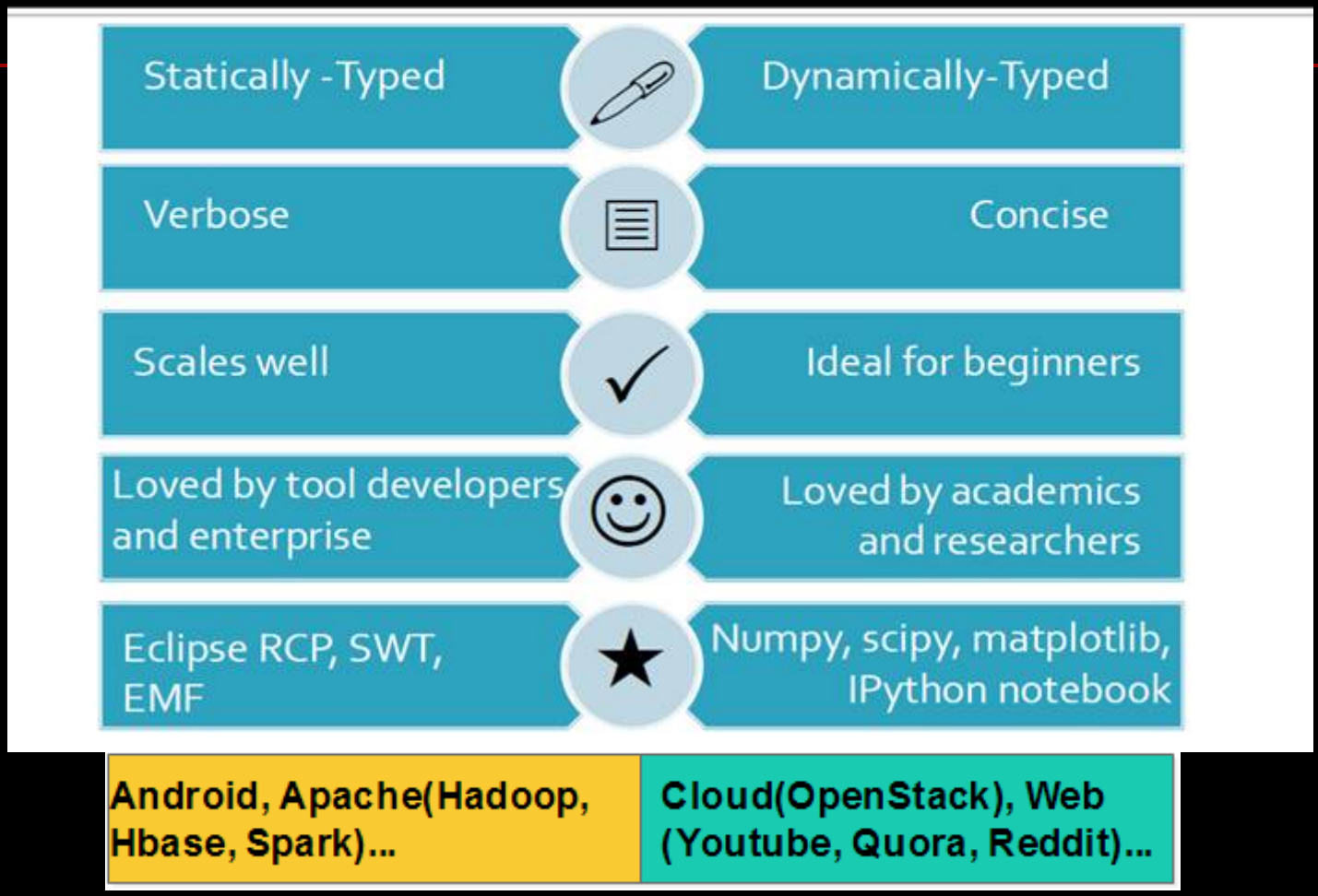
- http://www.tiobe.com/tiobe_index

May 2016	May 2015	Change	Programming Language	Ratings	Change
1	1		Java	20.956%	+4.09%
2	2		C	13.223%	-3.62%
3	3		C++	6.698%	-1.18%
4	5	^	C#	4.481%	-0.78%
5	6	^	Python	3.789%	+0.06%
6	9	^	PHP	2.992%	+0.27%
7	7		JavaScript	2.340%	-0.79%
8	15	^^	Ruby	2.338%	+1.07%
9	11	^	Perl	2.326%	+0.51%
10	8	v	Visual Basic .NET	2.325%	-0.64%

- **http://pypl.github.io/PYPL.html**

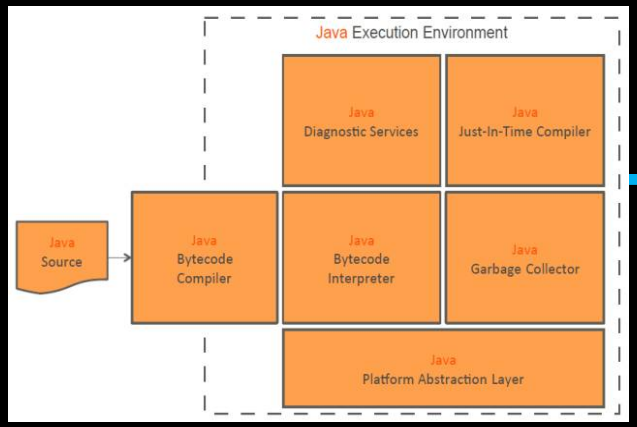
Worldwide, May 2016 compared to a year ago:

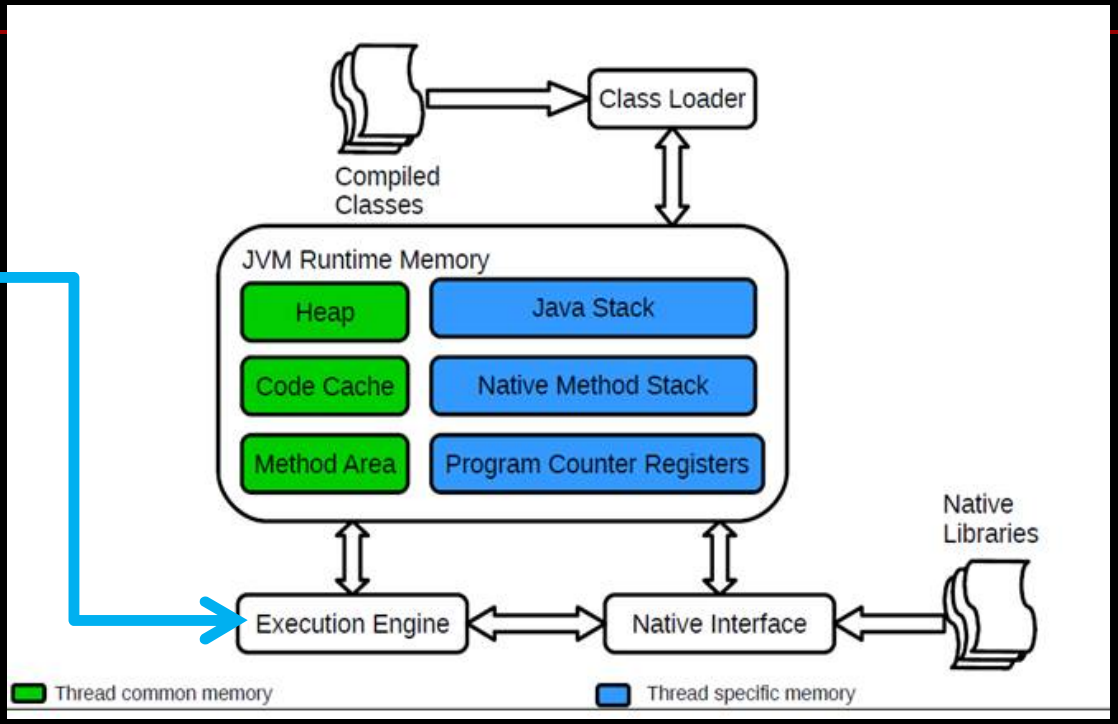
Rank	Change	Language	Share	Trend
1		Java	24.0 %	-0.2 %
2	↑	Python	12.4 %	+1.8 %
3	↓	PHP	10.6 %	-0.8 %
4		C#	8.9 %	-0.4 %
5	↑↑	Javascript	7.5 %	+0.5 %
6	↓	C++	7.4 %	-0.3 %
7	↓	C	7.2 %	+0.1 %
8		Objective-C	4.7 %	-0.7 %
9	↑	R	3.1 %	+0.5 %
10	↑	Swift	3.0 %	+0.4 %


JAVA
PYTHON
Statically -Typed
Dynamically-Typed
Verbose
Concise
Scales well
Ideal for beginners
Loved by tool developers and enterprise
Loved by academics and researchers
Eclipse RCP, SWT, EMF
Numpy, scipy, matplotlib, IPython notebook
Android, Apache(Hadoop, Hbase, Spark)...
Cloud(OpenStack), Web (Youtube, Quora, Reddit)...

JVM Architecture

- https://en.wikipedia.org/wiki/Java_virtual_machine





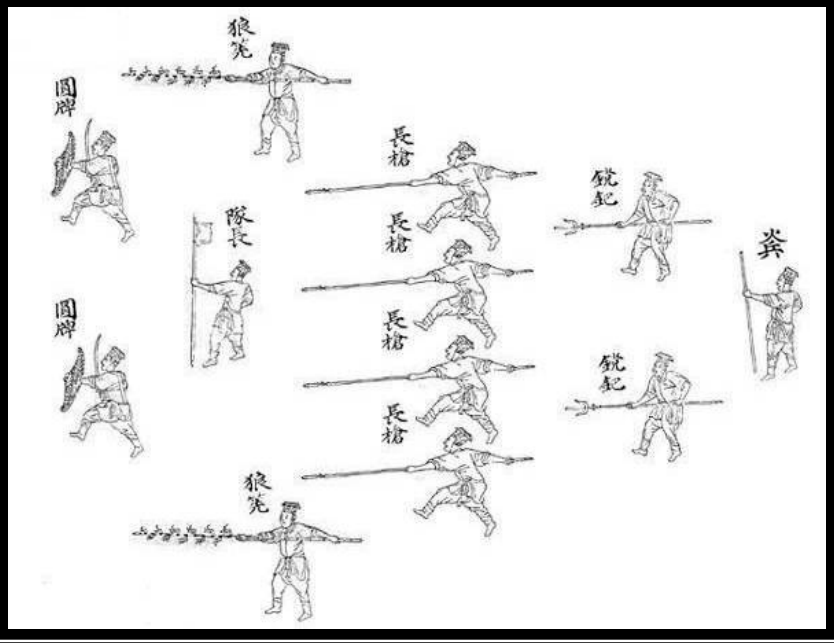
II. Python/Java Interop

1) Mixed-Language Programming

- https://en.wikipedia.org/wiki/Polyglot_(computing)
- https://msdn.microsoft.com/en-us/library/aa270930%28v=vs.60%29.aspx

Why

- Polyglot — use the best tool for the right jobs: high performance, scripting, web, functional programming, etc



Jython

- http://www.jython.org
- https://github.com/jythontools/jython.git
- a Java-based Python implementation and the most seamless way to integrate Python and Java

---

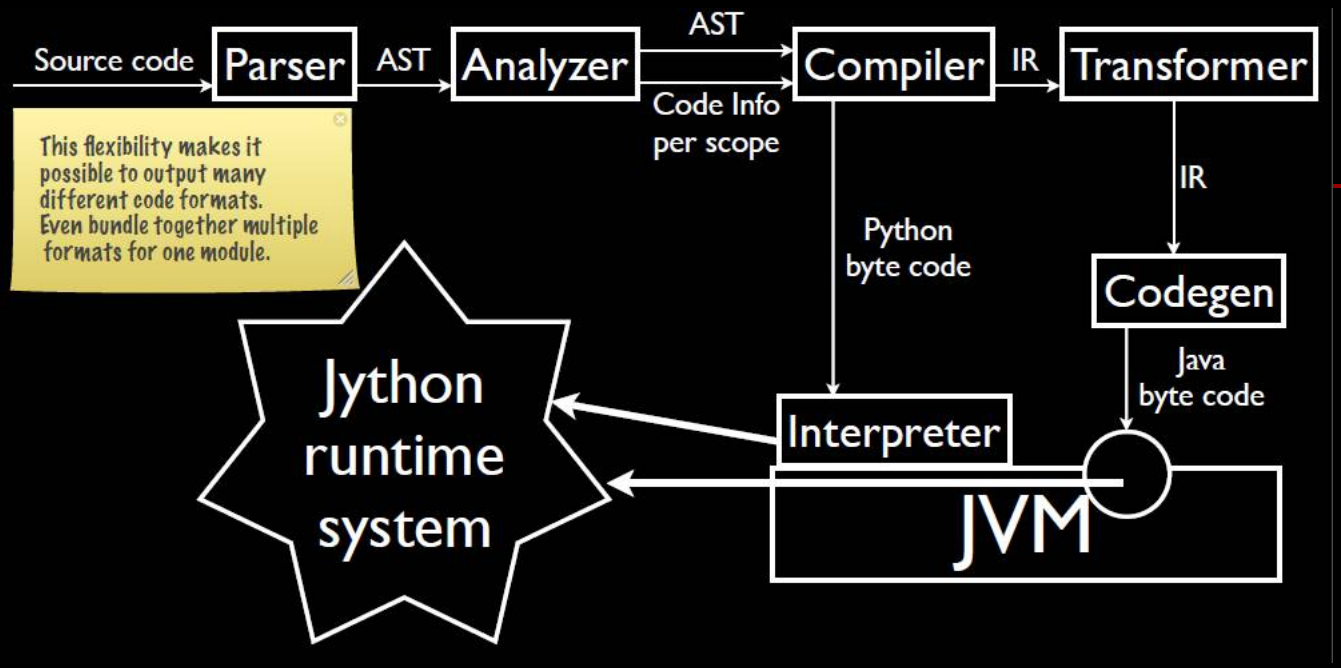
Why

- combines the flexibility/productivity of Python and the mature commercial support for Java
- re-use of Java infrastructure (GC…)
- features full multithreading support (i.e., it has no GIL issue)
- …

Jython3

- https://wiki.python.org/jython/Jython3000
- https://github.com/jython/jython3.git

Workflow



limitation & solution

- no support for native CPython-style C-extensions
- solution: JyNI (http://jyni.org)
- https://github.com/Stewori/JyNI.git
- seamless native extensions integration (no need to recompile the extension code or build a customized Jython fork)

Py4j

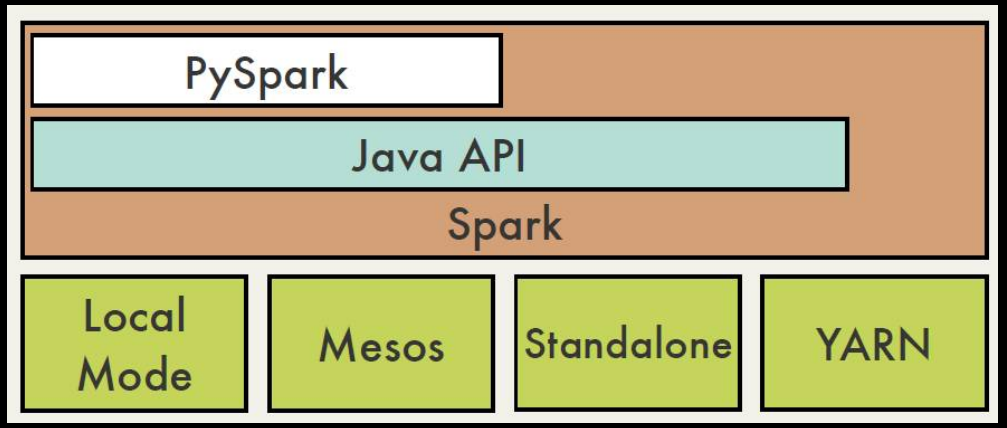
- https://www.py4j.org/
- https://github.com/bartdag/py4j

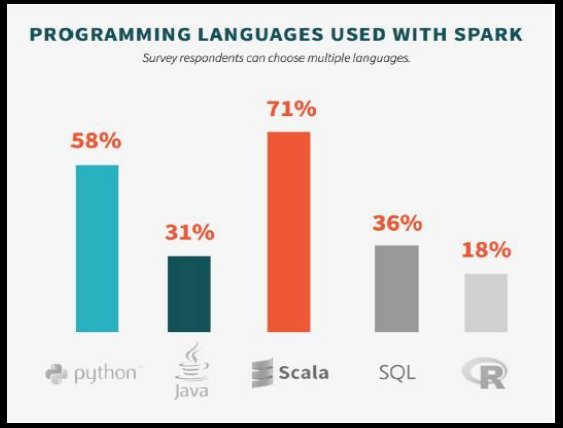
```
>>> from py4j.java_gateway import JavaGateway
>>> gateway = JavaGateway()                  # connect to the JVM
>>> random = gateway.jvm.java.util.Random()  # create a java.util.Random instance
>>> number1 = random.nextInt(10)             # call the Random.nextInt method
>>> number2 = random.nextInt(10)
>>> print(number1,number2)
(2, 7)
>>> addition_app = gateway.entry_point       # get the AdditionApplication instance
>>> addition_app.addition(number1,number2)   # call the addition method
9
```

socket

```
import py4j.GatewayServer;

public class AdditionApplication {

    public int addition(int first, int second) {
        return first + second;
    }

    public static void main(String[] args) {
        AdditionApplication app = new AdditionApplication();
        //app is now the gateway.entry_point
        GatewayServer gatewayServer = new GatewayServer(app);
        gatewayServer.start();
        System.out.println("Gateway Server Started");
    }

}
```

- https://cwiki.apache.org/confluence/display/SPARK/PySpark+Internals





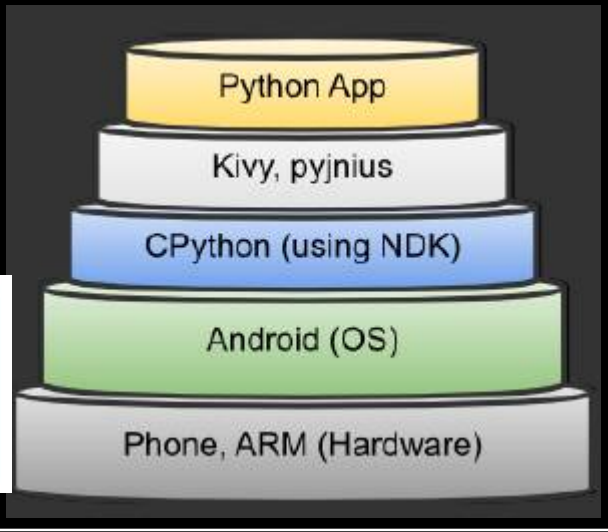
Pyjnius

- http://pyjnius.readthedocs.io/en/latest/
- https://github.com/kivy/pyjnius/
- Accessing Java Classes from Python，implemented with JNI and Java reflection

```
>>> from jnius import autoclass
>>> autoclass('java.lang.System').out.println('Hello world')
Hello world

>>> Stack = autoclass('java.util.Stack')
>>> stack = Stack()
>>> stack.push('hello')
>>> stack.push('world')
>>> print stack.pop()
world
>>> print stack.pop()
hello
```

- Available for mobile platforms

```
from jnius import autoclass

Hardware = autoclass('org.renpy.android.Hardware')
print 'DPI is', Hardware.getDPI()
```



Summary

- from my point of view:

	Jython	Py4j	PyJnius
Pros	mature, seamless, and concurrent	distributed/cloud; support Python 3; update actively	lightweight; support various platforms
Cons	lack of Python 3 support; update inactively	socket is needed	limited audience
Direction	bi-directional	bi-directional	unidirectional (Python-->Java)
Native	JFFI, JyNI	No(RPC)	JNI
App	WebSphere administration; Pylons	PySpark	Kivy
License	PSF v2	BSD	MIT

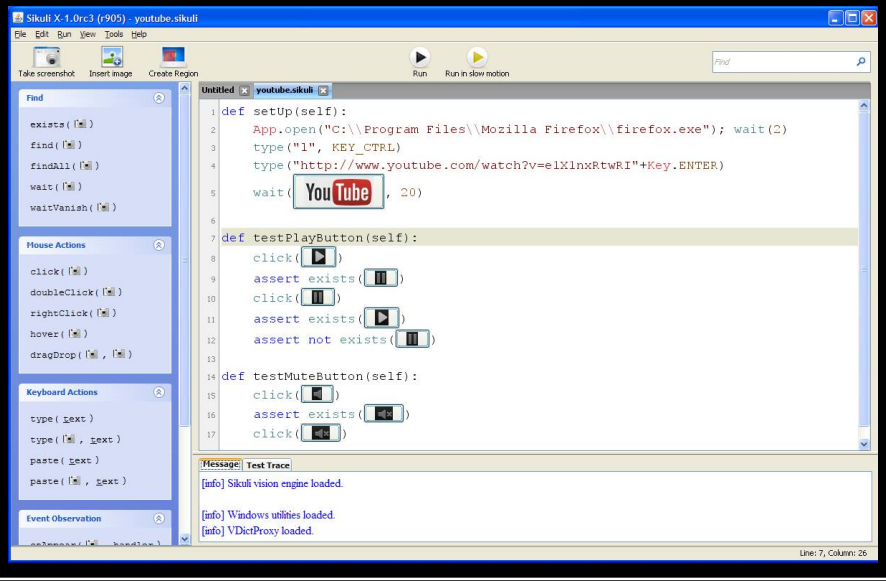
- many other Python-Java integration solutions like jpy, jpype, javabridge…

2) Samples

Sikuli

- http://sikuli.org
- Originally developed by by User Interface Design Group, MIT Computer Science and Artificial Intelligence Laboratory

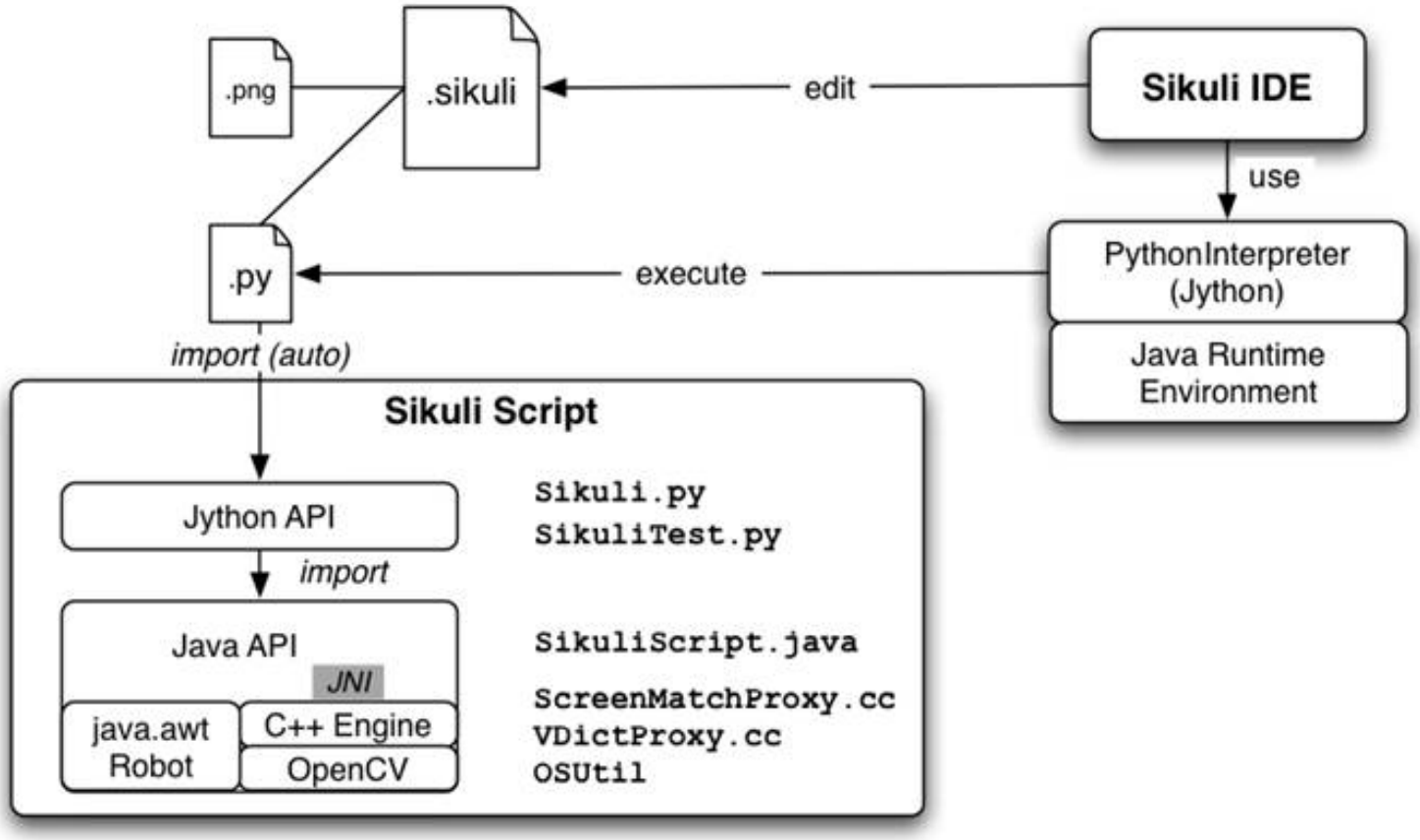
- Making Program



幻灯片 13

KL1 Koo Li, 2016/5/28

How Sikuli Works ?



Python for JDK development

Mercurial

- https://blogs.oracle.com/kto/entry/mercurial_openjdk_questions
- On December 2007, Sun moved the revision control of OpenJDK from TeamWare to Mercurial, as part of the process of releasing it to open source communities
- http://hg.openjdk.java.net/
- https://pypi.python.org/pypi/Mercurial

File	Type	Py Version	Uploaded on	Size
mercurial-3.8.1-cp27-cp27m-win32.whl (md5)	Python Wheel	2.7	2016-05-01	1MB
mercurial-3.8.1-cp27-cp27m-win_amd64.whl (md5)	Python Wheel	2.7	2016-05-01	1MB
mercurial-3.8.1.tar.gz (md5, pgp)	Source		2016-05-01	4MB

MX

- https://github.com/graalvm/mx
- https://wiki.openjdk.java.net/display/Graal/Instructions
- mx is a command line based tool for managing the development of (primarily) Java code. It includes a mechanism for specifying the dependencies as well as making it simple to build, test, run, update, etc the code and built artifacts
- mx --help

mx commands

```
mx [global options] [command] [command-specific options]
```

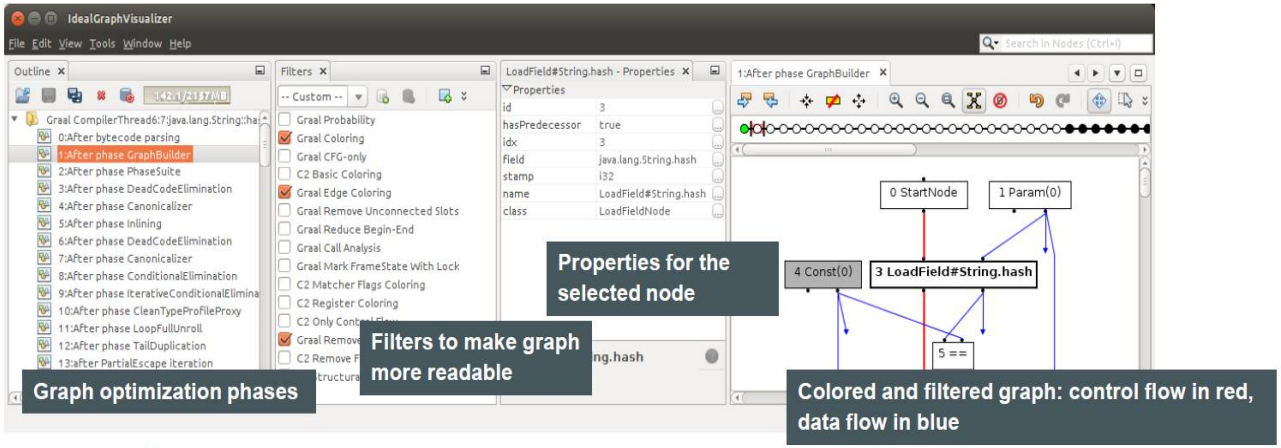
mx spull
mx build
mx vm
mx ideinit
mx igv
mx unittest
...

IR Example: Ideal Graph Visualizer

Start the Graal VM with graph dumping enabled

```
$ ./mx.sh igv &
$ ./mx.sh unittest -G:Dump= -G:MethodFilter=String.hashCode GraalTutorial#testStringHashCode
```

Test that just compiles `String.hashCode()`



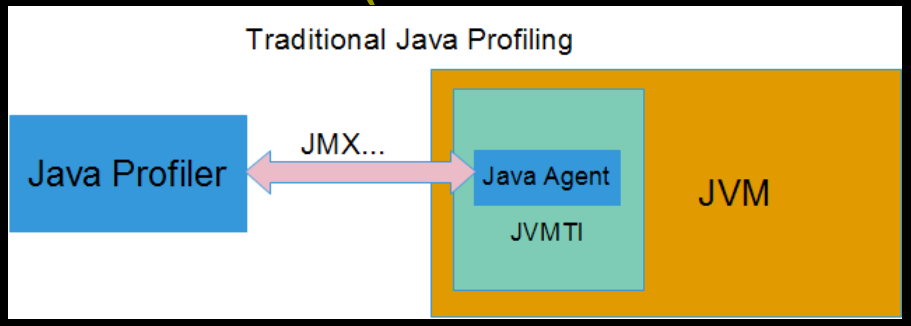
III. Python-assisted Java Development

1) Java Flight Recorder Scripting

- https://docs.oracle.com/javacomponents/
- http://hirt.se/blog

JFR (Java Flight Recorder)

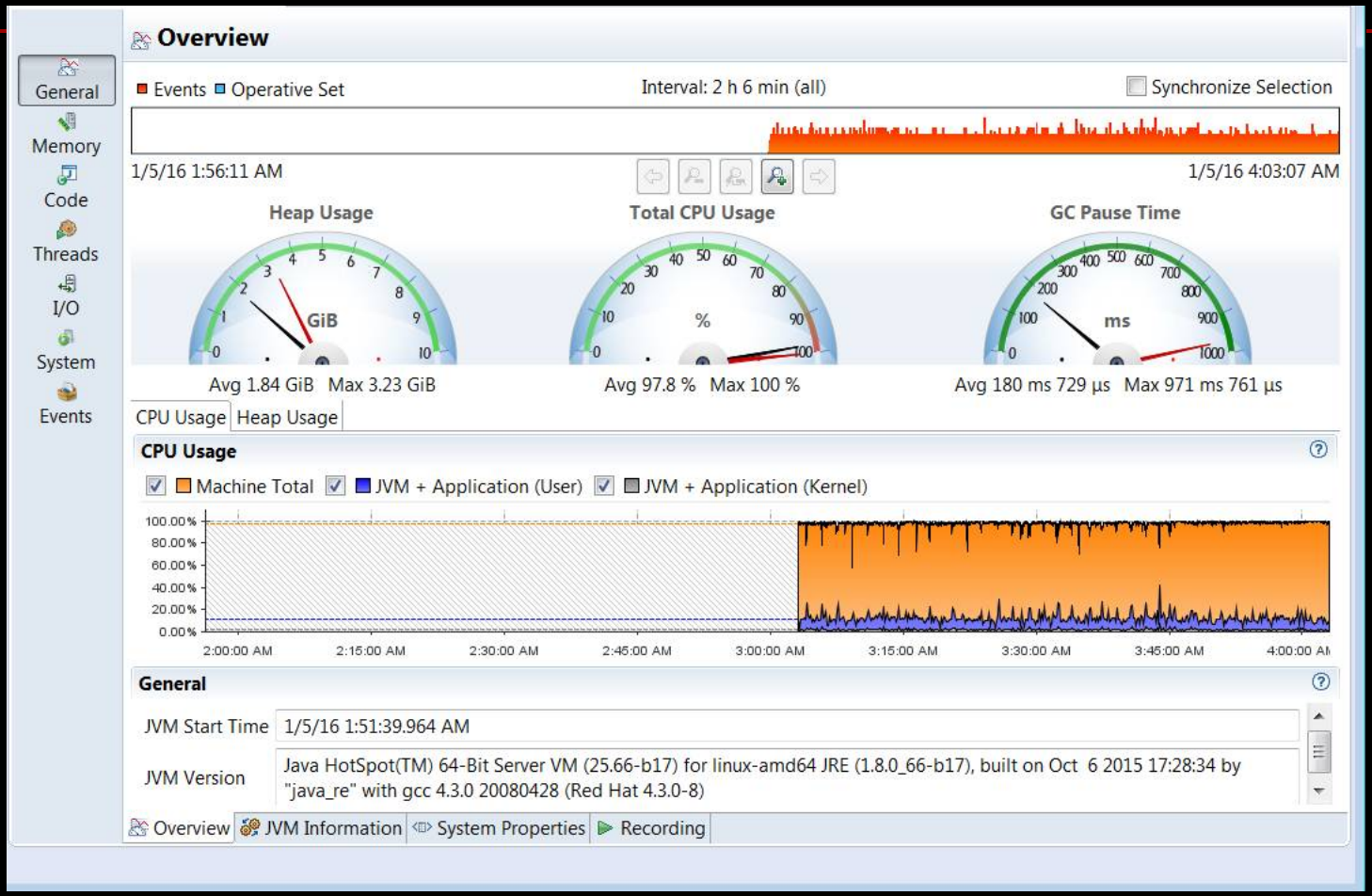
- Tracing and Profiling (first released in 7u40)
- Non-intrusive (built into the JVM itself)



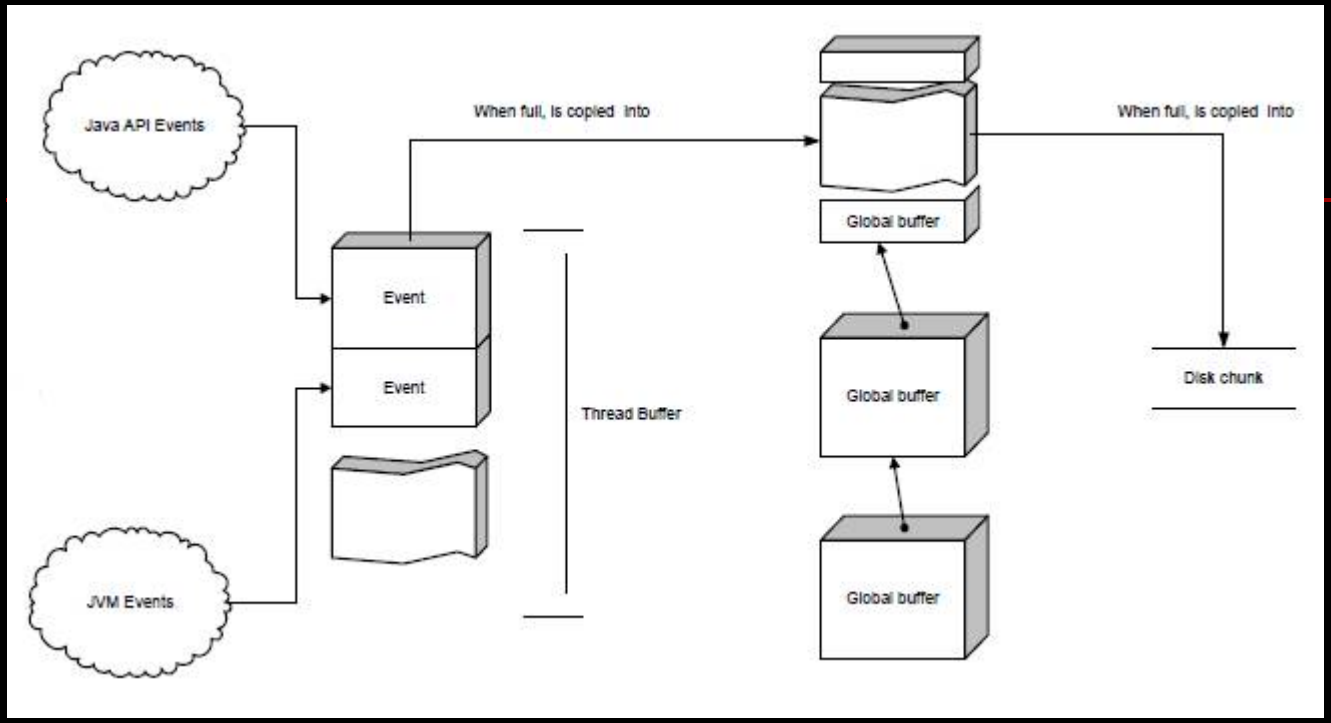
- On-demand profiling
- After-the-fact capture and analysis
- Captures both JVM and application data
  - GC, Synchronization, Compiler, CPU Usage, Exceptions, I/O…

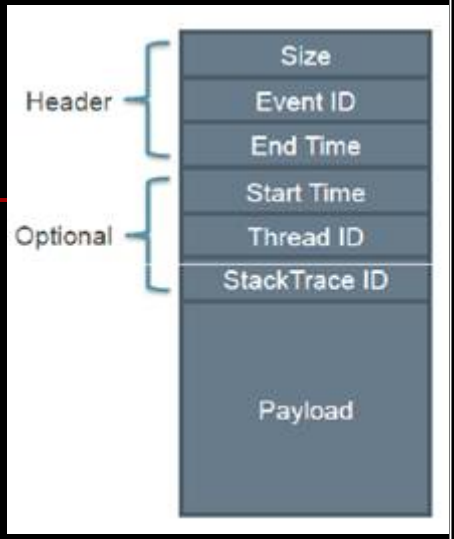
JMC (Java Mission Control)

- $JDK_HOME/bin/jmc
- Eclipse RCP based



- Event



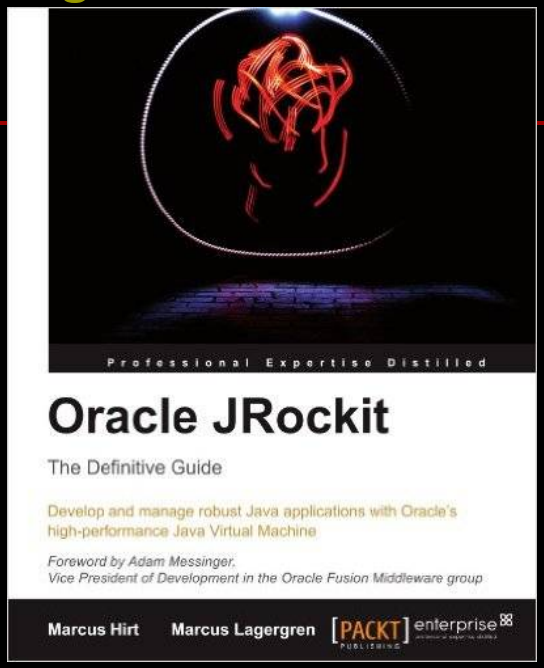


- Paths

```
$JDK_HOME/jre/lib/jfr.jar                 //Java interface
$JDK_HOME/jre/lib/amd64/libjfr.so         //JNI
$JDK_HOME/jre/lib/jfr/*.jfc               //config files
$JDK_HOME/lib/missioncontrol              //JMC
```

Summary

- Migrated from JRockit to HotSpot



- -XX:+StartAttachListener -XX:+UnlockCommercialFeatures -XX:+FlightRecorder
  e.g.:
  jcmd #PID# JFR.start name=MyRecording settings=profile dumponexit=true
  jcmd #PID# JFR.stop name=MyRecording filename=/tmp/MyRecording.jfr

- JFR, JMC…, all of them are closed source!

Demo

- http://hirt.se/blog/?p=446
- iterates through a recording and prints out the total event count:
- Core i5-2410M @ 2.3Ghz x 4, 6G RAM, Ubuntu-15.10-desktop64

```
export JAVA_HOME=/opt/DevSW/Java/JDK/Oracle/Std/jdk1.8.0_92
export ZIPPY_HOME=/opt/MyWorkSpace/MyProjs/Open-Source/Python/Impl/Java/ZipPy
export JYTHON_HOME=/opt/DevSW/Python/Impl/Jython/v2.7.0
export CLASSPATH=.:/opt/MyWorkSpace/MyProjs/Open-Source/Python/Impl/Java/ZipPy/zippy-mirror/zippy.jar:
                 $JAVA_HOME/lib/missioncontrol/plugins/com.jrockit.mc.flightrecorder_5.5.0.165303.jar:
                 $JAVA_HOME/lib/missioncontrol/plugins/com.jrockit.mc.common_5.5.0.165303.jar
export PATH=$ZIPPY_HOME/mx:$JAVA_HOME/bin:$JYTHON_HOME/bin:$PATH
```

- Java program

```
import java.io.File;
import java.util.Date;
import com.jrockit.mc.flightrecorder.FlightRecording;
import com.jrockit.mc.flightrecorder.FlightRecordingLoader;
import com.jrockit.mc.flightrecorder.spi.IEvent;
import com.jrockit.mc.flightrecorder.spi.IView;

public class FlightRecorderParserTest
{
    public static void main(String[] args) throws ClassNotFoundException
    {
        System.out.println("FlightRecorderParserTest: start at " + new Date(System.currentTimeMillis()));
        System.out.println("FlightRecorderParserTest: file name = " + args[0]);
        FlightRecording recording = FlightRecordingLoader.loadFile(new File(args[0]));
        IView view = recording.createView();

        int count = 0;
        for (IEvent event : view) {
            count++;
        }
        System.out.println("Found " + count + " events!");
        System.out.println("FlightRecorderParserTest: end at " + new Date(System.currentTimeMillis()));
    }
}
```

```
mydev@mydev-Dell-System-XPS-15Z:/opt/MyWorkSpace/Test/Java/Tuning/JFR/Parser$ time java FlightRecorderParserTest /opt/MyWorkSpace/MyProjs/Open-Source/Tuning/Java/JFR/Result/.jfr/RecordingTest.jfr
FlightRecorderParserTest: start at Sat May 28 22:37:26 CST 2016
FlightRecorderParserTest: file name = /opt/MyWorkSpace/MyProjs/Open-Source/Tuning/Java/JFR/Result/.jfr/RecordingTest.jfr
Found 1341752 events!
FlightRecorderParserTest: end at Sat May 28 22:37:33 CST 2016

real    0m7.517s
user    0m23.852s
sys     0m1.272s
```

Jython version

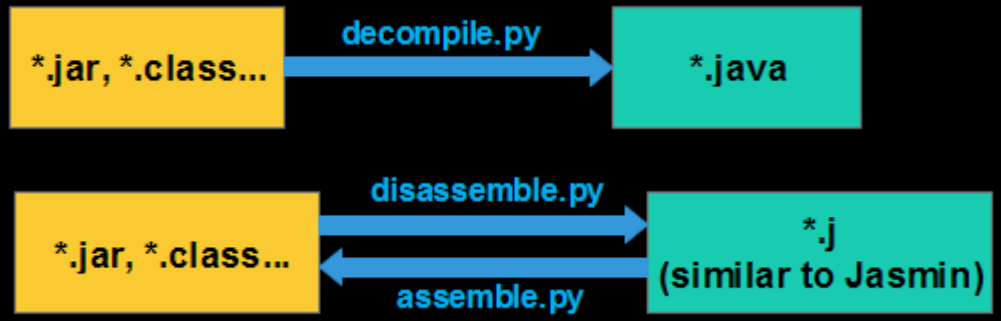
```
import os.path
import sys
from java.io import File
from com.jrockit.mc.flightrecorder import FlightRecording, FlightRecordingLoader
from com.jrockit.mc.flightrecorder.spi import IEvent, IView


def usage():
    prog = os.path.basename(__file__)
    print("usage:    %s jfrfile" % prog)
    print("args:")
    print("jfrfile   path of the jfr file(.jfr) to be parsed by this script")
    print("e.g.:     %s /opt/MyWorkSpace/Test/JFR/jfrfilename\n" % prog)

def FlightRecorderParserTest():
    count = 0
    recording = FlightRecordingLoader.loadFile(File(sys.argv[1]))
    view = recording.createView();
    for event in view:
        count = count + 1
    print "Found %d events!" % count

if __name__ == '__main__':
    if (len(sys.argv) != 2):
        usage()
        sys.exit()
    FlightRecorderParserTest()
```

```
mydev@mydev-Dell-System-XPS-15Z:/opt/MyWorkSpace/Test/Java/Tuning/JFR/Parser$ time jython JDKParserTest.py /opt/MyWorkSpace/MyProjs/Open-Source/Tuning/Java/JFR/Result/.jfr/RecordingTest.jfr
Found 1341752 events!

real    0m8.649s
user    0m27.516s
sys     0m1.456s
```

2) Java Bytecode Manipulation via Python

Krakatau

- https://github.com/Storyyeller/Krakatau
- a set of Python scripts for decompiling, disassembling, and assembling Java classfiles



- uses a much more heuristic approach to the decompilation, which makes it robust to minor obfuscation, though it has the drawback of not reconstructing the "original" source, leading to less readable output than a pattern matching decompiler would produce for unobfuscated Java classes

KL2 Koo Li, 2016/5/28

Demo

- tableswitch & lookupswitch

```
mydev@mydev-Dell-System-XPS-15Z:/opt/MyWorkSpace/MyProjs/Open-Source/Java/Bytecode/Python/Krakatau/Krakatau$ ./disassemble.py -out /opt/MyWorkSpace/Out/Hack/Java/Decompiliation/Krakatau /opt/MyWorkSpace/Test/Java
/Hack/SwitchStringTest.class
Krakatau  Copyright (C) 2012-16  Robert Grosse
This program is provided as open source under the GNU General Public License.
See LICENSE.TXT for more details.

processing target /opt/MyWorkSpace/Test/Java/Hack/SwitchStringTest.class, 1/1 remaining
Class written to /opt/MyWorkSpace/Out/Hack/Java/Decompiliation/Krakatau/SwitchStringTest.j
0.0139620304108  seconds elapsed
```

SwitchStringTest.java

```
1  public class SwitchStringTest
2  {
3      public static void main(String[] args)
4      {
5          switch (args[0])
6          {
7              case "first":
8                  System.out.println("one");
9                  break;
10             case "second":
11                 System.out.println("four");
12                 break;
13             case "third":
14                 System.out.println("eight");
15                 break;
16             default:
17                 System.out.println("default");
18         }
19     }
20 }
```

```
public static void main(String args[])
{
    String s = args[0];
    byte byte0 = -1;
    switch(s.hashCode())
    {
    case 97440432:
        if(s.equals("first"))
            byte0 = 0;
        break;
    case -906279820:
        if(s.equals("second"))
            byte0 = 1;
        break;
    case 110331239:
        if(s.equals("third"))
            byte0 = 2;
        break;
    }
    switch(byte0)
    {
    case 0: // '\0'
        System.out.println("one");
        break;
    case 1: // '\001'
        System.out.println("four");
        break;
    case 2: // '\002'
        System.out.println("eight");
        break;
    default:
        System.out.println("default");
        break;
    }
}
```

SwitchStringTest.j

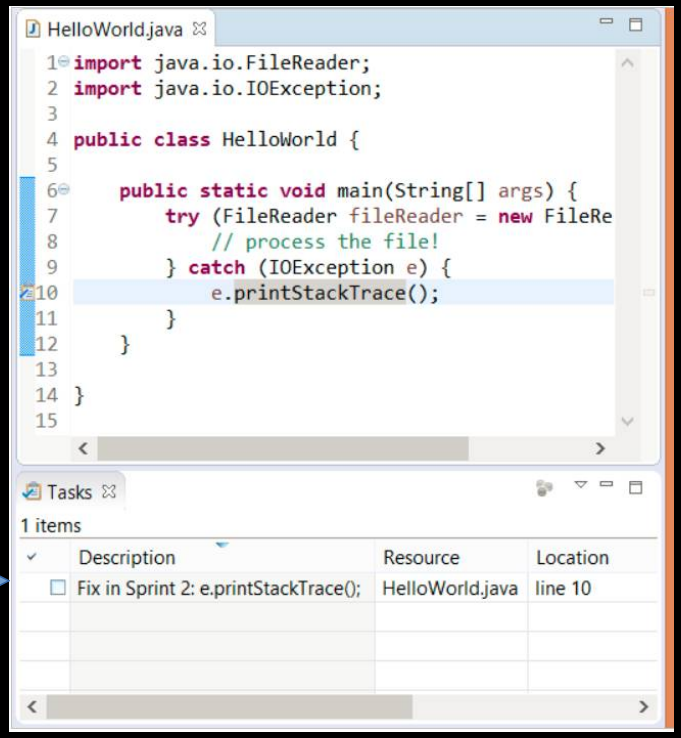
```
1  .version 52 0
2  .class public super SwitchStringTest
3  .super java/lang/Object
4
5  .method public <init> : ()V
6      .code stack 1 locals 1
7  L0:     aload_0
8  L1:     invokespecial Method java/lang/Object <init> ()V
9  L4:     return
10 L5:
11         .linenumbertable
12             L0 1
13         .end linenumbertable
14     .end code
15 .end method
16
17 .method public static main : ([Ljava/lang/String;)V
18     .code stack 2 locals 3
19 L0:     aload_0
20 L1:     iconst_0
21 L2:     aaload
22 L3:     astore_1
23 L4:     iconst_m1
24 L5:     istore_2
25 L6:     aload_1
26 L7:     invokevirtual Method java/lang/String hashCode ()I
27 L10:    lookupswitch
28             -906279820 : L58
29             97440432 : L44
30             110331239 : L72
31             default : L83
32
33
34         .stack append Object java/lang/String Integer
35 L44:    aload_1
36 L45:    ldc 'first'
37 L47:    invokevirtual Method java/lang/String equals (Ljava/lang/Object;)Z
38 L50:    ifeq L83
39 L53:    iconst_0
40 L54:    istore_2
41 L55:    goto L83
60         .stack same
61 L83:    iload_2
62 L84:    tableswitch 0
63             L112
64             L123
65             L134
66             default : L145
67
68
69         .stack same
70 L112:   getstatic Field java/lang/System out Ljava/io/PrintStream;
71 L115:   ldc 'one'
72 L117:   invokevirtual Method java/io/PrintStream println (Ljava/lang/String;)V
73 L120:   goto L153
74
75         .stack same
76 L123:   getstatic Field java/lang/System out Ljava/io/PrintStream;
77 L126:   ldc 'four'
78 L128:   invokevirtual Method java/io/PrintStream println (Ljava/lang/String;)V
79 L131:   goto L153
80
81         .stack same
82 L134:   getstatic Field java/lang/System out Ljava/io/PrintStream;
83 L137:   ldc 'eight'
84 L139:   invokevirtual Method java/io/PrintStream println (Ljava/lang/String;)V
85 L142:   goto L153
86
87         .stack same
88 L145:   getstatic Field java/lang/System out Ljava/io/PrintStream;
89 L148:   ldc 'default'
90 L150:   invokevirtual Method java/io/PrintStream println (Ljava/lang/String;)V
91
92         .stack chop 2
93 L153:   return
```

3) Eclipse Advanced Scripting Environment

- https://wiki.eclipse.org/EASE
- a scripting framework for easy extension and control of Eclipse IDE/RCP applications using your favourite scripting language

- Hack your IDE with Python & EASE

```
loadModule('/System/Resources')

from org.eclipse.core.resources import IMarker

for ifile in findFiles("*.java"):
    file_name = str(ifile.getLocation())
    print "Processing " + file_name
    with open(file_name) as f:
        for line_no, line in enumerate(f, start=1):
            if "printStackTrace" in line:
                marker = ifile.createMarker(IMarker.TASK)
                marker.setAttribute(IMarker.TRANSIENT,
True)
                marker.setAttribute(IMarker.LINE_NUMBER,
line_no)
                marker.setAttribute(IMarker.MESSAGE, "Fix
in Sprint 2: " + line.strip())
```



IV. Python Runtime Implementation

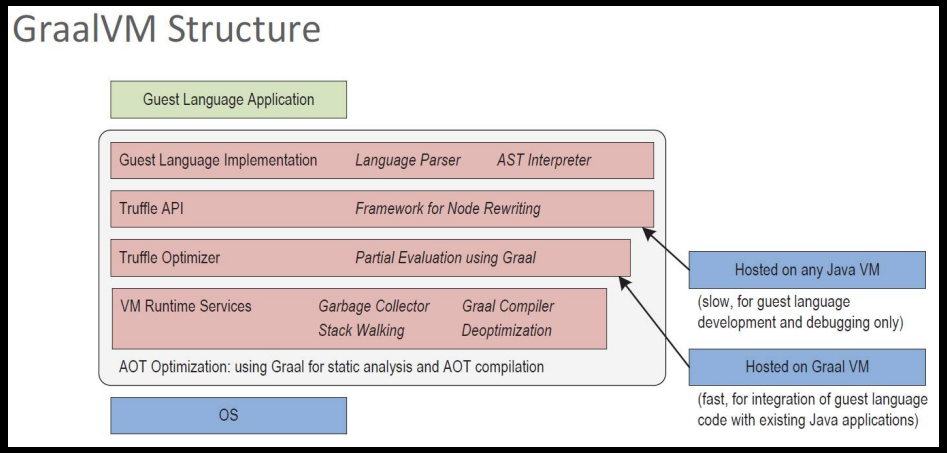
1) Truffle & Graal

- http://lafo.ssw.uni-linz.ac.at/

JKU JOHANNES KEPLER UNIVERSITY LINZ — SSW — Oracle Labs

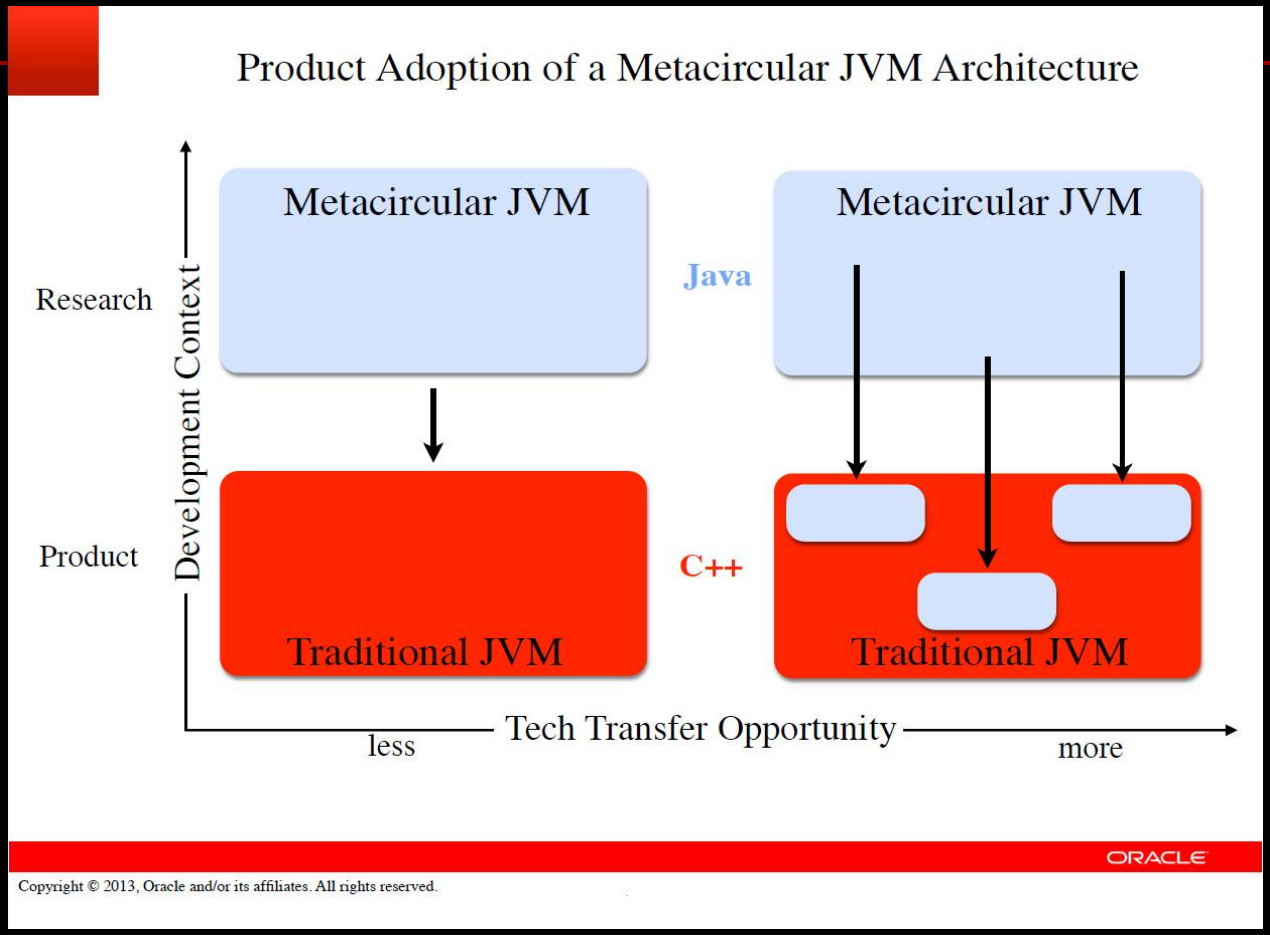
Institute for System Software

- https://github.com/graalvm
- http://www.oracle.com/technetwork/oracle-labs/program-languages



Maxine

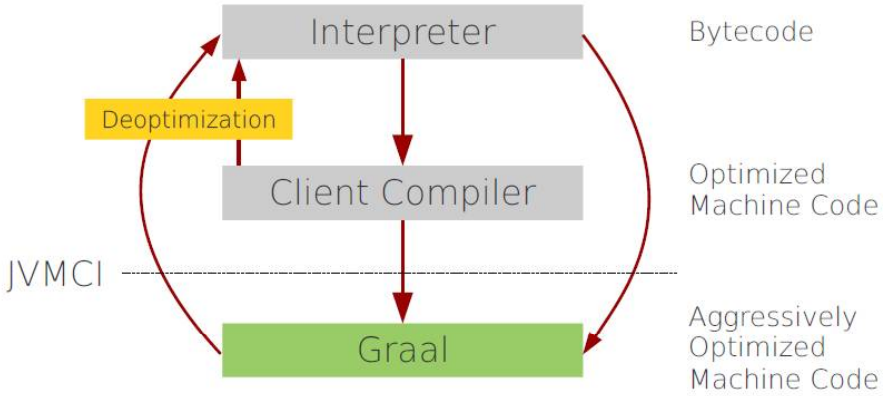
- https://en.wikipedia.org/wiki/Maxine_Virtual_Machine
- https://kenai.com/projects/maxine



Graal

- https://en.wikipedia.org/wiki/Graal_(compiler)
- A high-performance optimizing JIT compiler for the Java HotSpot VM
  - Written in Java and benefitting from Java's annotation and metaprogramming

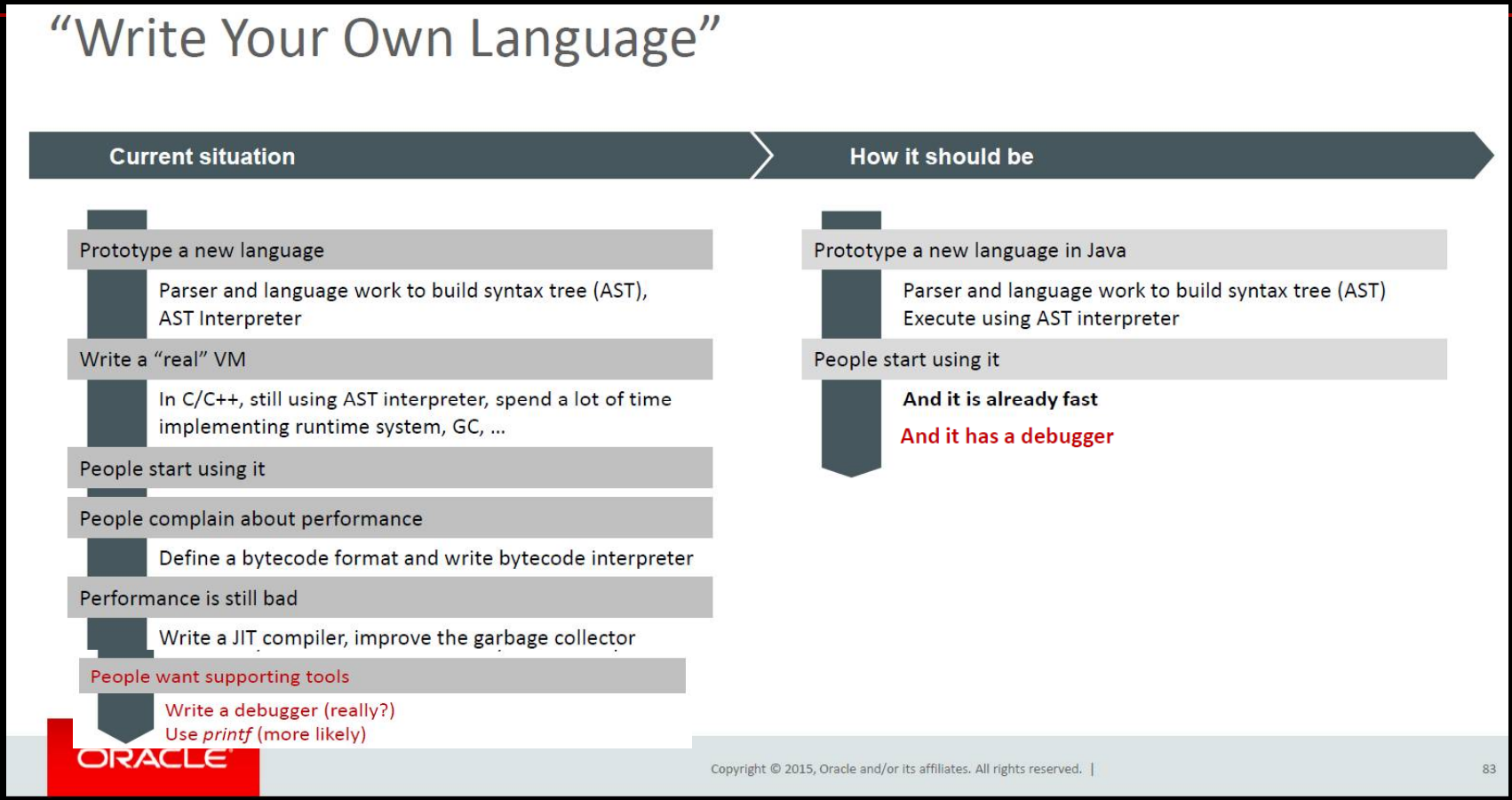
- Optimizing JIT compiler for the HotSpot VM
  - OpenJDK Project
    https://github.com/graalvm/graal-core
- Written in Java
  - Meta-circular
  - Benefiting from Java's annotation and meta-programming, IDE support



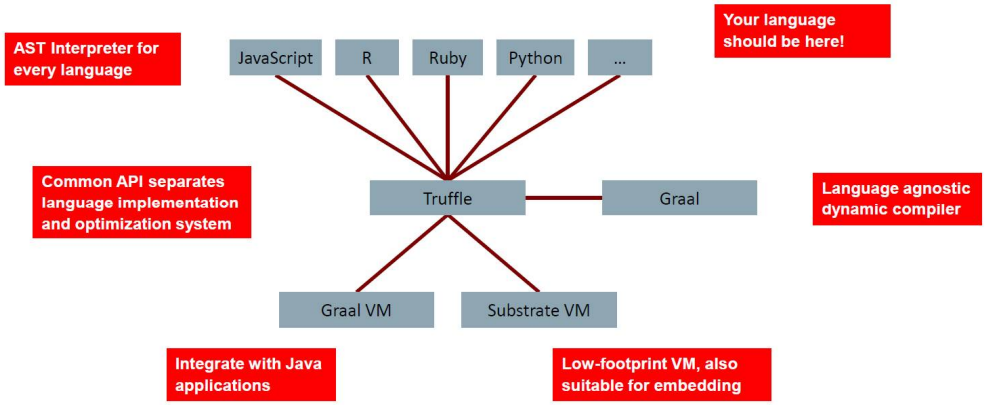

```
$ git clone https://github.com/graalvm/graal-core.git
$ cd graal-core
$ mx build
$ mx vm -version
```

Truffle

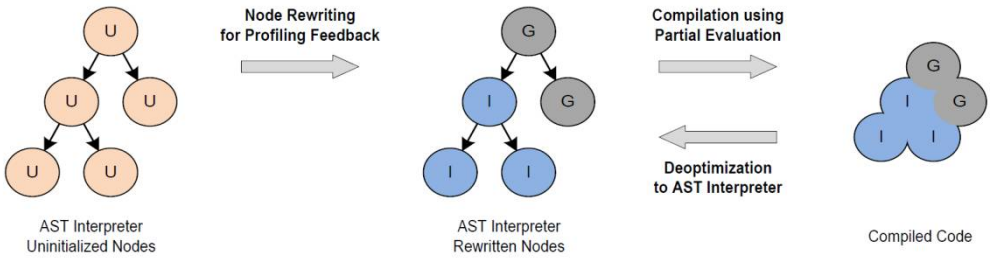
- **https://en.wikipedia.org/wiki/Graal_(compiler)**
- **A Language Implementation Framework that uses Graal for Custom Compilation**



Truffle System Structure



Truffle Approach



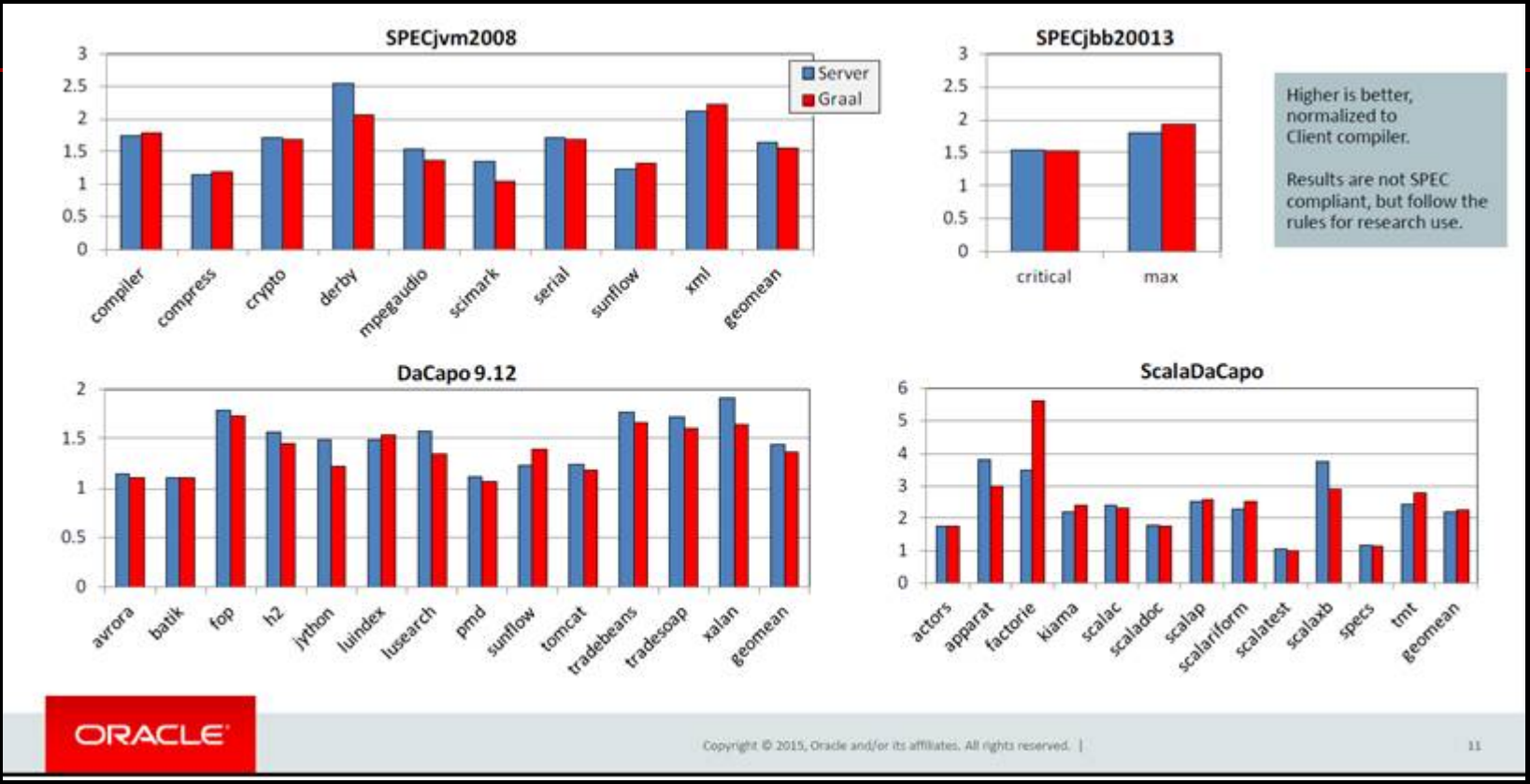
Benifits

- **Multilanguage**
  - **Is your language on Truffle/Graal?**
- **Customization**
- **Tooling**

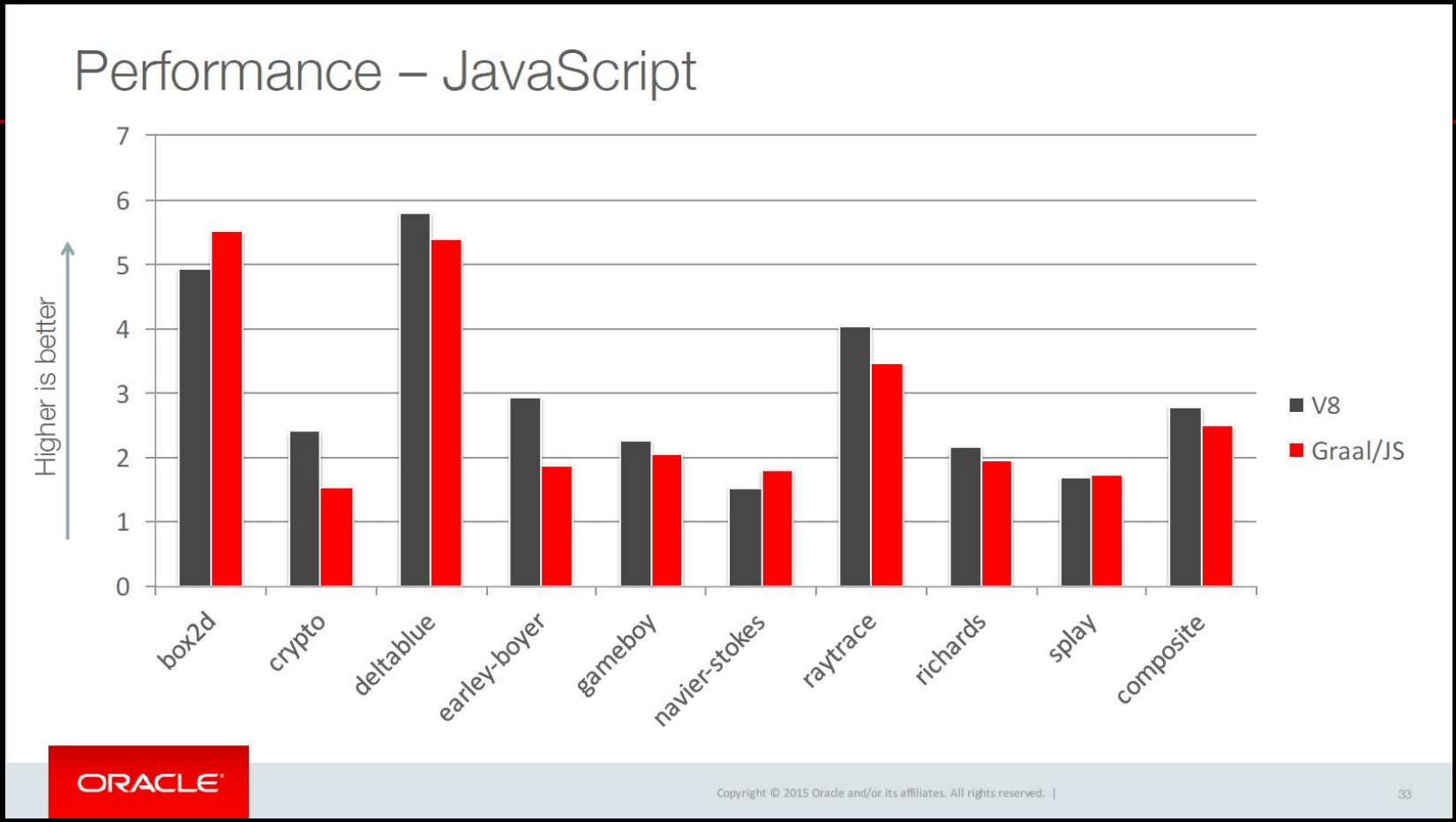
  - Truffle's instrumentation framework as target of:
    - Profilers
    - Debuggers
    - Coverage tools
    - ...

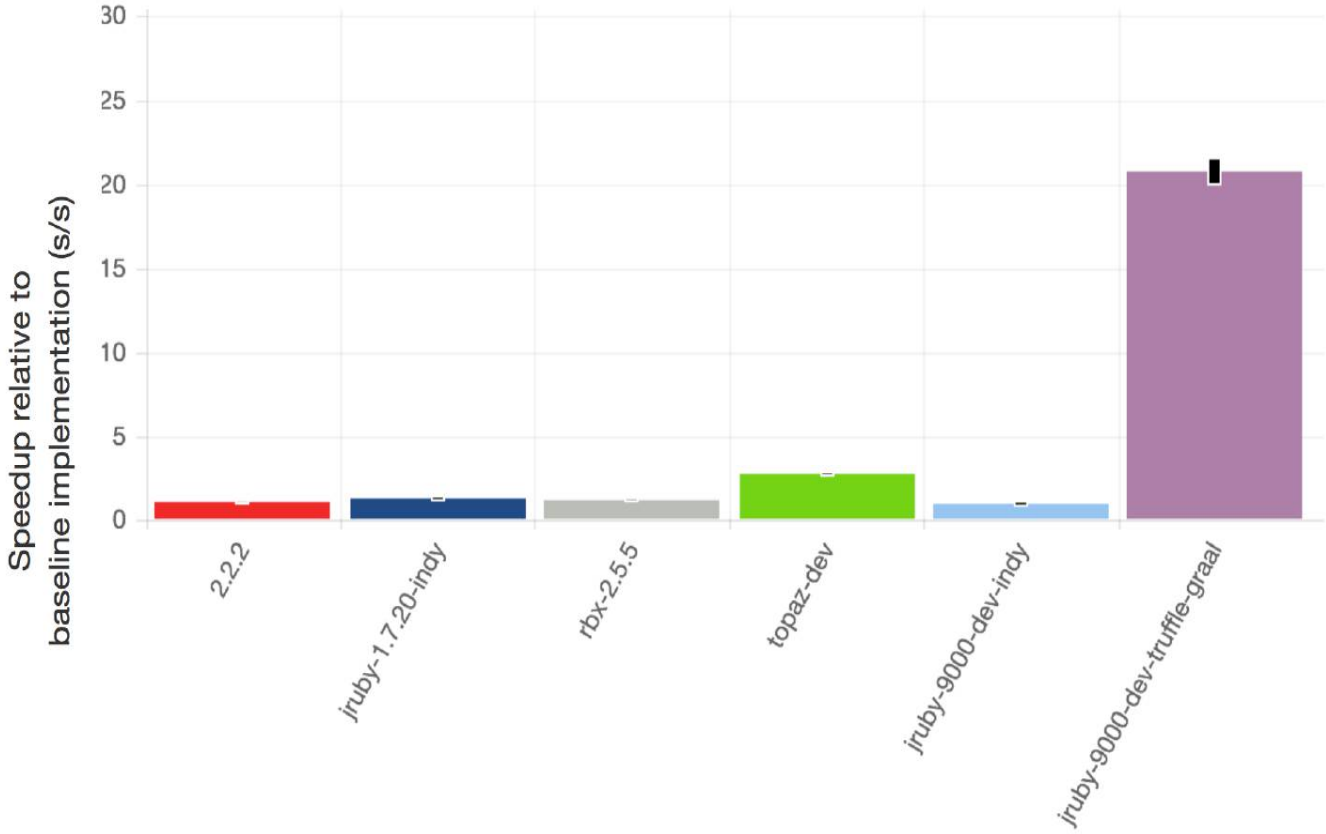
Benchmarking

- **Graal Benchmark Results**

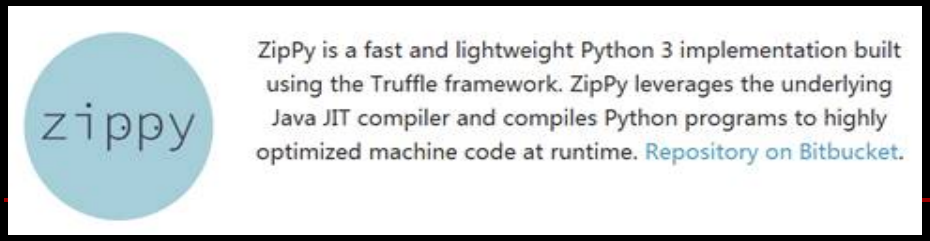


- **Implementations**

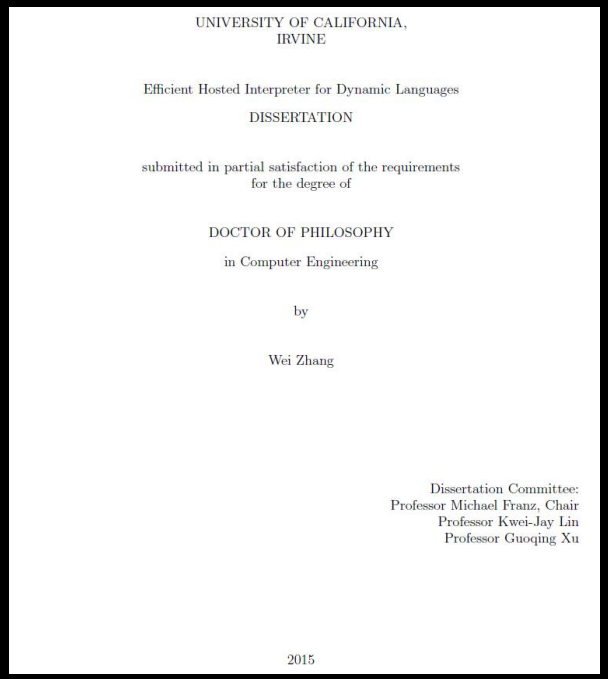




2) Performance Gained in ZipPy



- https://github.com/qunaibit/zippy-mirror
- http://thezhangwei.com/
  http://thezhangwei.com/documents/uci-thesis.pdf
- Optimizations
  - Numeric Types
  - Type Specializations
  - Efficient Data Representation
  - Control Flow Specializations
  - Generator Peeling
  - Optimizing Object Model and Calls



UNIVERSITY OF CALIFORNIA,
IRVINE

Efficient Hosted Interpreter for Dynamic Languages

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Computer Engineering

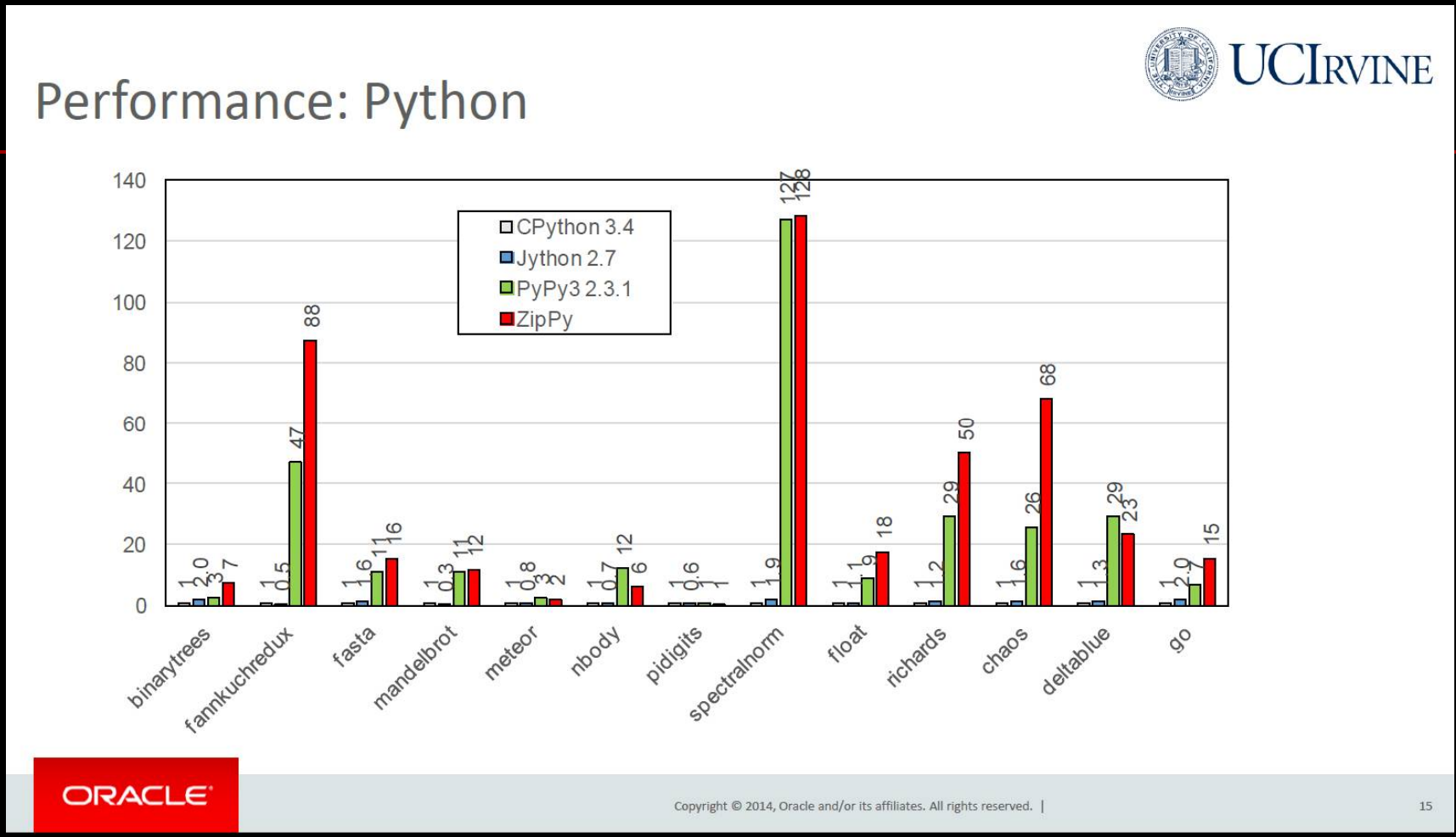
by

Wei Zhang

Dissertation Committee:
Professor Michael Franz, Chair
Professor Kwei-Jay Lin
Professor Guoqing Xu

2015

Performance



Benchmark	CPython3	CPython	Jython	PyPy	PyPy3	ZipPy
binarytrees	1.00	0.94	1.99	2.60	2.70	7.31
fannkuchredux	1.00	0.97	0.51	44.53	47.29	87.50
fasta	1.00	1.04	1.55	11.73	11.24	15.57
mandelbrot	1.00	1.08	0.34	10.91	10.82	11.69
meteor	1.00	1.02	0.77	2.64	2.62	2.13
nbody	1.00	0.97	0.73	12.13	12.06	6.17
pidigits	1.00	1.00	0.62	0.98	0.95	0.60
spectralnorm	1.00	1.33	1.89	127.33	127.25	128.10
float	1.00	0.95	1.05	8.64	8.67	17.71
richards	1.00	0.94	1.21	29.53	29.25	50.13
chaos	1.00	1.17	1.55	40.88	25.69	68.28
deltablue	1.00	0.85	1.33	30.08	29.14	23.46
go	1.00	1.08	1.99	6.79	6.66	15.41
mean	**1.00**	**1.02**	**1.05**	**12.15**	**11.68**	**15.34**

The speedups of Python VMs normalized to CPython3 running regular benchmarks

Benchmark	Score −GP	Score +GP	Speedup	No. gen	No. genexp	No. of lines
nqueens	69.09	313.14	**4.53**	2/2	5/5	41
euler11	71.42	941.73	**13.19**	2/2	5/5	61
euler31	47.70	134.35	**2.82**	1/1†	2/2	46
eratos	277.13	316.64	**1.14**	2/2	0/0	86
lyndon	37.89	859.91	**22.69**	3/3	0/0	127
partitions	50.36	217.56	**4.32**	1/1	0/0	228
pymaging	102.80	283.99	**2.76**	2/2	0/0	1528
python-graph	51.89	93.08	**1.79**	2/2	2/2	3136
simplejson	66.12	242.52	**3.67**	1/1	0/0	3128
sympy	198.55	259.68	**1.31**	4/5†	1/2	262k
whoosh	242.74	676.10	**2.79**	4/4	0/0	40k
mean			**3.58**			

† Contains recursive generator calls.

The performance numbers of generator peeling

limits

- Not fully compatible with Python 3
  – The Grand Unified Python Benchmark Suite, etc
- No “Standalone” mode like Jython

```
$ java -jar jython.jar script.py
```

- Base on mx for running

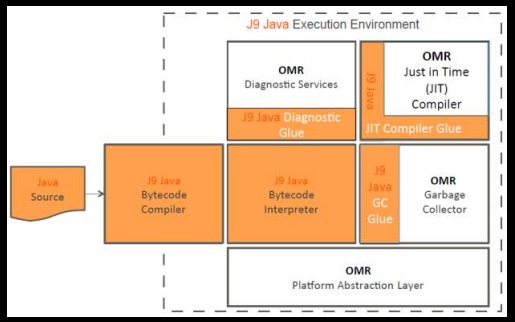
Run:

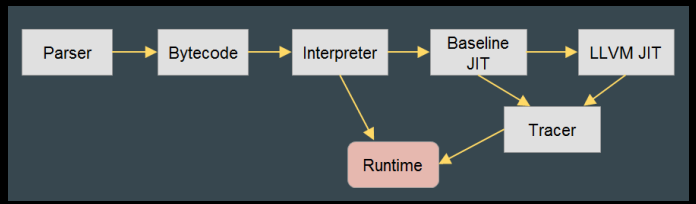
```
$ cd $ZIPPY_HOME/zippy
$ mx python <file.py>
```

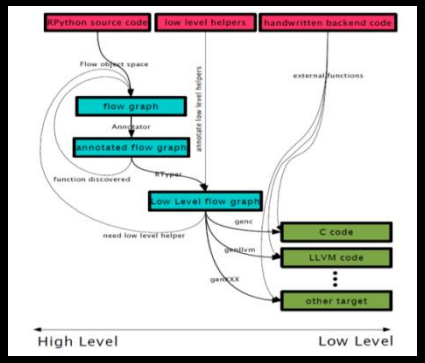
3) Summary

- from my point of view, various Runtime Frameworks for Python implementation:

	OMR	LLVM	PyPy	GraalVM
Pros	easily leverage new hardware features	high efficiency; mature	productivity(RPython); PyPy3, PyPy-STM; multiplatform support	leverage mature JVM technologies; productivity(Java); high performance
Cons	productivity (C++/C)?	dead of VMKit...	mainly for dynamic languages	?
Performance	not sure	not enough	not enough	not enough?
Native			CFFI, CPPYY	GNFI (Graal Native Function Interface)
Related Projects	JBM J9	Unladen Swallow, PyStON	Psyco	ZipPy
License	Eclipse v1.0/Apache v2.0	LLVM	MIT	GPL v2

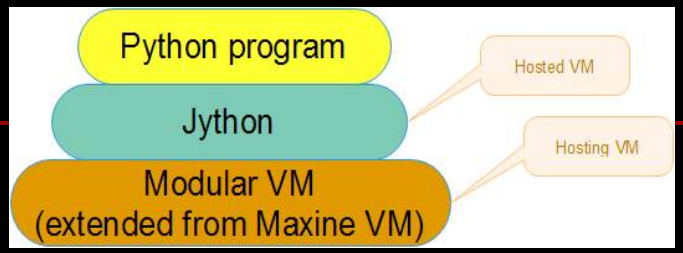






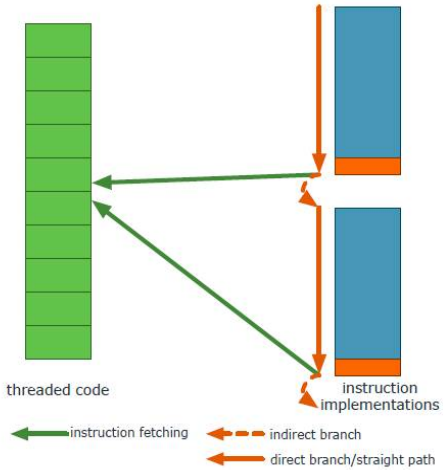
VM Design

- Threaded Code Generation

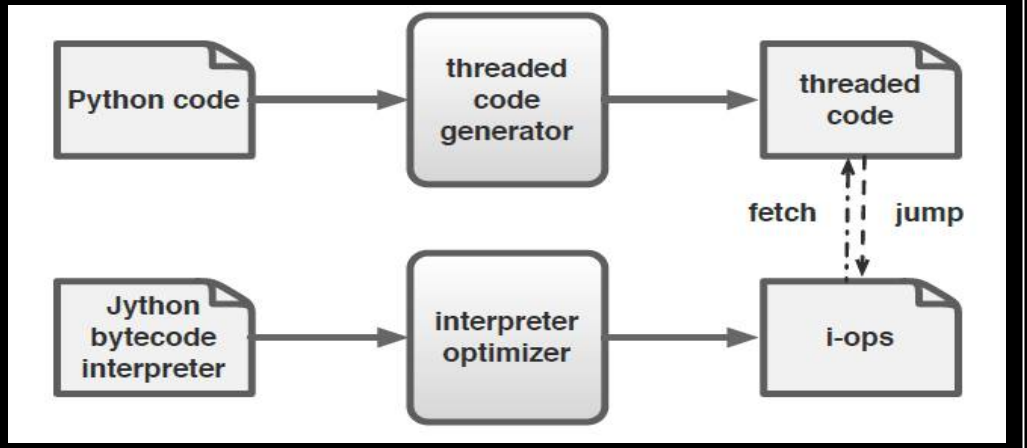


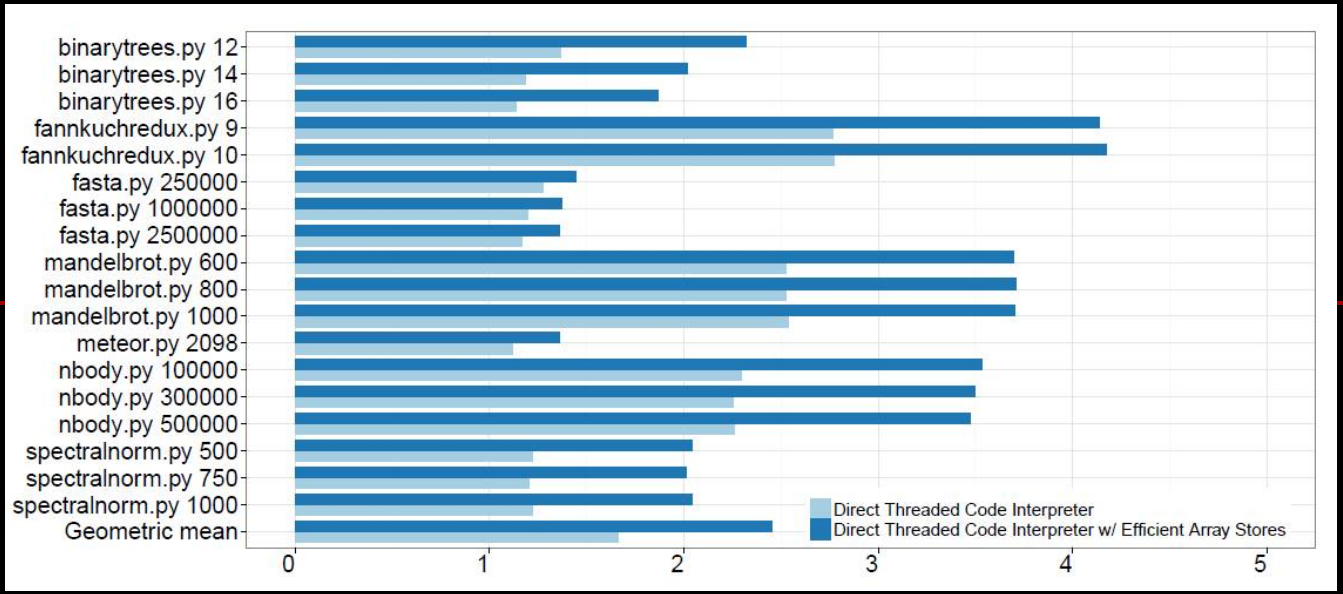

```
Inst thread[] =
          {&add, &pop...};
// starting point
goto *thread++;

// instruction implementations
add:
  sp[1] = sp[0] + sp[1];
  sp++;
  goto *thread++;
```

(a) direct threading interpreter

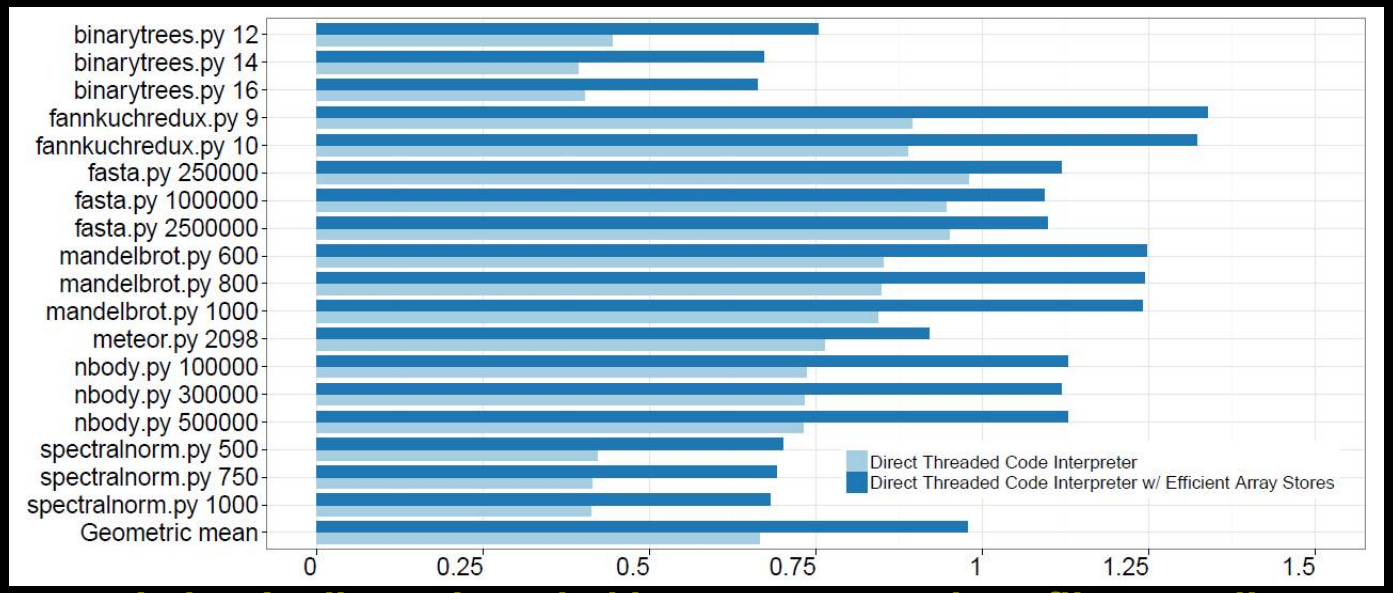


(b) branches in direct threading dispatch





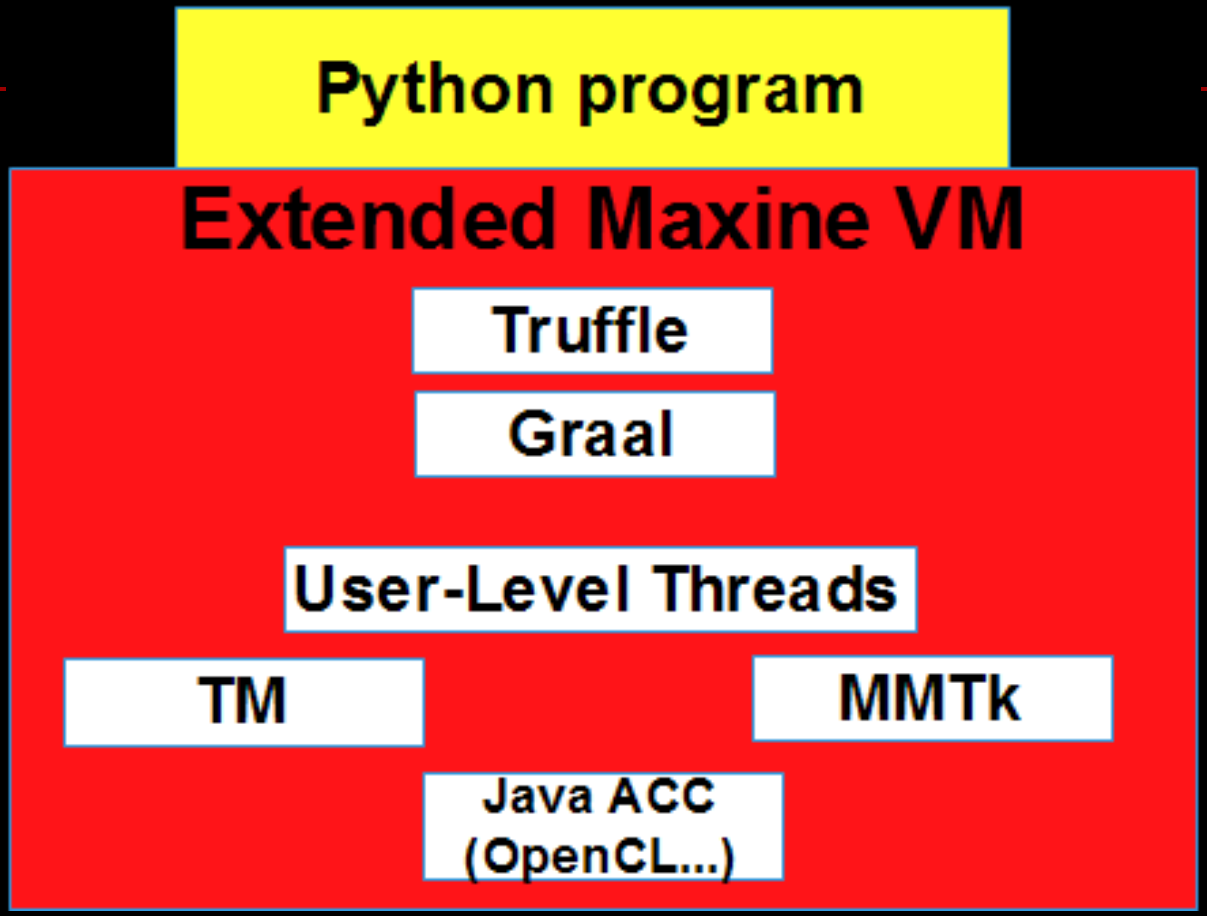
Jython's direct threaded interpreter vs. switch-based



Jython's direct threaded interpreter vs. class file compiler

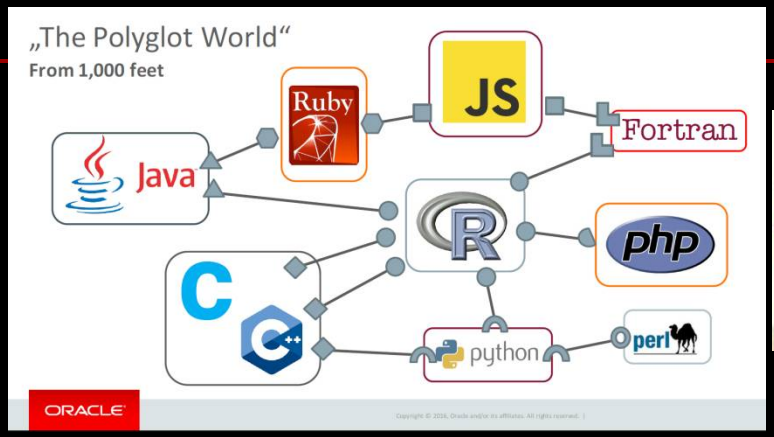
Future

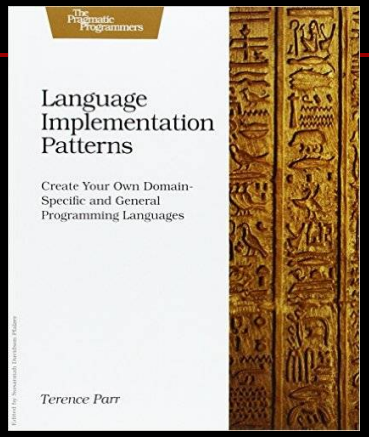
- How about a redesign…



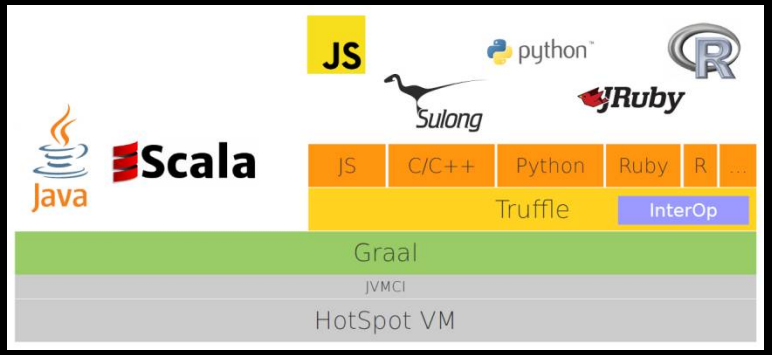
V. Summary

- Mixed-Language Programming **vs** Domain Specific Language





- So many great runtime frameworks today!



http://www.eclipse.org/omr
https://github.com/eclipse/omr
omr-dev@eclipse.org

VI. Reference

Slides/materials from many and varied sources:

- http://en.wikipedia.org/wiki/Wiki
- http://www.slideshare.net/
- https://www.python.org
- http://openjdk.java.net/
- https://jdk9.java.net/
- http://openjdk.java.net/jeps/0
- http://en.wikipedia.org/wiki/List_of_Java_virtual_machines
- https://en.wikipedia.org/wiki/Comparison_of_Java_virtual_machines
- http://www.neshkov.com/
- http://llvm.org
- http://en.wikipedia.org/wiki/Unladen_Swallow
- https://github.com/dropbox/pyston
- http://pypy.org
- https://cffi.readthedocs.org
- http://pypy.readthedocs.org/en/latest/cppyy.html
- https://opensource.com/life/16/2/how-use-python-hack-your-ide
- https://en.wikipedia.org/wiki/Runtime_system
- https://en.wikipedia.org/wiki/Intermediate_representation
- http://www.jikesrvm.org
- …

Q & A

Thanks!