Getting to Jython 2.7 and Beyond

Jim Baker

jim.baker@{python.org, rackspace.com}

Jython background

- Implementation of Python for the Java platform

Jython background

- Implementation of Python for the Java platform
- Compiles to Java bytecode

Jython background

- Implementation of Python for the Java platform
- Compiles to Java bytecode
- Under development since 1997

Jython background

- Implementation of Python for the Java platform
- Compiles to Java bytecode
- Under development since 1997
- Jython 2.7 release candidate 2 now available

Questions you might have or heard from others

- Isn't Jython a dead project?

Questions you might have or heard from others

- Isn't Jython a dead project?
- Doesn't Jython have a GIL, just like CPython?

Questions you might have or heard from others

- Isn't Jython a dead project?
- Doesn't Jython have a GIL, just like CPython?
- Isn't Jython much slower than other Python implementations?

Questions you might have or heard from others

- Isn't Jython a dead project?
- Doesn't Jython have a GIL, just like CPython?
- Isn't Jython much slower than other Python implementations?
- Doesn't Jython only implement a subset of Python?

Questions you might have or heard from others

- Isn't Jython a dead project?
- Doesn't Jython have a GIL, just like CPython?
- Isn't Jython much slower than other Python implementations?
- Doesn't Jython only implement a subset of Python?

Answer: **No**

About me

I have a vested interest:

- Core developer of Jython

About me

I have a vested interest:

- Core developer of Jython
- Co-author of *Definitive Guide to Jython* from Apress

About me

I have a vested interest:

- Core developer of Jython
- Co-author of *Definitive Guide to Jython* from Apress
- Committing on Jython since 2007. . .

About me

I have a vested interest:

- Core developer of Jython
- Co-author of *Definitive Guide to Jython* from Apress
- Committing on Jython since 2007...
- User of Python (including Jython) since 2003

About me

I have a vested interest:

- Core developer of Jython
- Co-author of *Definitive Guide to Jython* from Apress
- Committing on Jython since 2007...
- User of Python (including Jython) since 2003
- Software developer at Rackspace

Various tweets

December 2013:

> *Can Jython be saved or is it pretty much dead at this point?*

Various tweets

December 2013:

> *Can Jython be saved or is it pretty much dead at this point?*

July 2014:

> *So is jython basically dead or what?*

Various tweets

December 2013:

> *Can Jython be saved or is it pretty much dead at this point?*

July 2014:

> *So is jython basically dead or what?*

January 2015:

> *Is it still around? I thought the project was dead :V*

Sometimes we are *too* optimistic

From my note to reviewers of this talk proposal:

> *I assume Jython 2.7.0 will see a final release by the end of this year, and certainly well before PyCon.*

Sadly, one last bug precludes this being true. But we were close!

Demo RC2

But here is release candidate 2!

Try importing the GIL

```
$ bin/jython
Jython 2.7rc2+ (default:0213400c518f, Apr 9 2015, 23:
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporatio
Type "help", "copyright", "credits" or "license" for 
>>> from __future__ import GIL
```

No global interpreter lock:

```
$ bin/jython
Jython 2.7rc2+ (default:0213400c518f, Apr 9 2015, 23:
[Java HotSpot(TM) 64-Bit Server VM (Oracle Corporatio
Type "help", "copyright", "credits" or "license" for 
>>> from __future__ import GIL
  File "<stdin>", line 1
SyntaxError: Never going to happen!
```

One performance benchmark. . .

```
$ python2.7 -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 8.87509
This machine benchmarks at 112675 pystones/second
```

One performance benchmark. . .

```
$ python2.7 -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 8.87509
This machine benchmarks at 112675 pystones/second
```

vs

```
$ jython27 -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 6.24945
This machine benchmarks at 160014 pystones/second
```

One **terrible** performance benchmark. . .

```
$ jython -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 6.24945
This machine benchmarks at 160014 pystones/second
```

vs

```
$ python2.7 -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 8.87509
This machine benchmarks at 112675 pystones/second
```

- Really, pystone??!!!

One **terrible** performance benchmark...

```
$ jython -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 6.24945
This machine benchmarks at 160014 pystones/second
```

vs

```
$ python2.7 -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 8.87509
This machine benchmarks at 112675 pystones/second
```

- Really, pystone??!!!
- Does not consider JVM startup time nor JIT warmup

One **terrible** performance benchmark. . .

```
$ jython -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 6.24945
This machine benchmarks at 160014 pystones/second
```

vs

```
$ python2.7 -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 8.87509
This machine benchmarks at 112675 pystones/second
```

- Really, pystone??!!!
- Does not consider JVM startup time nor JIT warmup
- Ignores GC issues

PyPy 2.5.1

Only 21x faster!

```
$ pypy -m test.pystone 1000000
Pystone(1.1) time for 1000000 passes = 0.29528
This machine benchmarks at 3.38662e+06 pystones/secon
```

On the other hand. . .

- But still, considered that we only focused on compatibility, not bad performance

On the other hand. . .

- But still, considered that we only focused on compatibility, not bad performance
- Most of the performance improvement from the efforts to improve Java 7 or Java 8

On the other hand. . .

- But still, considered that we only focused on compatibility, not bad performance
- Most of the performance improvement from the efforts to improve Java 7 or Java 8
- Actually do care about pystone because it measures some of the costs of dynamic overhead

Understanding projects

- Look at the email lists, wikis, code, what's new, bug reports, linked PRs and patches

Understanding projects

- Look at the email lists, wikis, code, what's new, bug reports, linked PRs and patches
- And especially the commit log, as guided by the above

Understanding projects

- Look at the email lists, wikis, code, what's new, bug reports, linked PRs and patches
- And especially the commit log, as guided by the above
- Commits to tell you what has changed

What about subset of Python language?

- Not just language features

What about subset of Python language?

- Not just language features
- Standard library

What about subset of Python language?

- Not just language features
- Standard library
- Python ecosystem

What about subset of Python language?

- Not just language features
- Standard library
- Python ecosystem

How have we responded?

Example: Deleted text in ACKNOWLEDGMENTS

```
- Python's inventor Guido van Rossum and
- the rest of PythonLabs continues to help
- and support Jython by their understanding
- of how Jython must live with the limits of
- Java.
```

Example: Deleted text in ACKNOWLEDGMENTS

```
- Python's inventor Guido van Rossum and
- the rest of PythonLabs continues to help
- and support Jython by their understanding
- of how Jython must live with the limits of
- Java.
```

Let's not do that!

Example: Inserted text in ACKNOWLEDGMENTS

This change makes it much better:

```
+ Jython: Python for the Java Platform

+ Jython follows closely the Python language
+ and its reference implementation CPython,
+ as created by Guido van Rossum.
+ Jython 2.7 corresponds to CPython 2.7.
```

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty
- requests and its beautiful API

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty
- requests and its beautiful API
- pip and setuptools, via ensurepip

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty
- requests and its beautiful API
- pip and setuptools, via ensurepip
- while also support this functionality for Windows

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty
- requests and its beautiful API
- pip and setuptools, via ensurepip
- while also support this functionality for Windows
- including executable zip archives

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty
- requests and its beautiful API
- pip and setuptools, via ensurepip
- while also support this functionality for Windows
- including executable zip archives
- and on localized platforms, including fixed issues to support Finnish, Japanese, Turkish, and more

Examples of compatibility changes

Support we have added:

- six and all of its crazy import hook magic - single source for Python 2 and 3
- characteristic and its class decorator magic
- socket/select/ssl using Netty
- requests and its beautiful API
- pip and setuptools, via ensurepip
- while also support this functionality for Windows
- including executable zip archives
- and on localized platforms, including fixed issues to support Finnish, Japanese, Turkish, and more
- which also means CJK encodings

Demo

Demo testing requests on Windows

Clamp

- Precise integration with Java

Clamp

- Precise integration with Java
- Java can directly import Python modules (at last!)

Clamp

- Precise integration with Java
- Java can directly import Python modules (at last!)
- Integrates with setuptools to produce jars

Clamp

- Precise integration with Java
- Java can directly import Python modules (at last!)
- Integrates with setuptools to produce jars
- Includes future integration as well with Maven via Aether

Clamp

- Precise integration with Java
- Java can directly import Python modules (at last!)
- Integrates with setuptools to produce jars
- Includes future integration as well with Maven via Aether
- Sprint topic for this week!

Python class, extending Java interfaces

```
from java.io import Serializable
from java.util.concurrent import Callable

class BarClamp(Callable, Serializable):

  def call(self):
    return 42
```

Python class, clamped

To import a Python class that you want to import into Java, add a couple of lines:

```
from java.io import Serializable
from java.util.concurrent import Callable
from clamp import clamp_base

BarBase = clamp_base("bar")  # Java package prefix

class BarClamp(BarBase, Callable, Serializable):

  def call(self):
    return 42
```

Clamping your class

Key insight: ahead-of-time builds through setuptools to produce a jar for Java linkage:

```
import ez_setup
ez_setup.use_setuptools()

from setuptools import setup, find_packages

setup(
  name = "clamped",
  version = "0.1",
  packages = find_packages(),
  install_requires = ["clamp"],
  clamp = ["clamped"],
)
```

Using from Java

Simply import clamped Python classes into Java code!

```java
import bar.clamped.BarClamp;

public class UseClamped {
  public static void main(String[] args) {
    BarClamp barclamp = new BarClamp();
    try {
      System.out.println("BarClamp: " +
        barclamp.call());
    } catch (Exception ex) {
      System.err.println("Exception: " + ex);
    }
  }
}
```

Still. . .

Not exactly interesting code!

Fireside

- Blazing fast WSGI bridge for servlet containers

Fireside

- Blazing fast WSGI bridge for servlet containers
- Passes standard WSGI tests in `wsgiref.validate`

Fireside

- Blazing fast WSGI bridge for servlet containers
- Passes standard WSGI tests in `wsgiref.validate`
- Sprinting on adding `ServletFilter` support this Friday (April 17)

Fireside

- Blazing fast WSGI bridge for servlet containers
- Passes standard WSGI tests in `wsgiref.validate`
- Sprinting on adding `ServletFilter` support this Friday (April 17)
- And uses Clamp!

Plug in with standard WAR support

Add to your `web.xml` these config directives:

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee"
 ...
 <servlet>
  <servlet-name>fireside</servlet-name>
  <servlet-class>
  org.python.tools.fireside.servlet.WSGIServlet
  </servlet-class>
  <init-param>
   <param-name>wsgi.handler</param-name>
   <param-value>hellowsgi.simple_app</param-value>
  </init-param>
 </servlet>
```

Fireside code

Start with

```
from javax.servlet.http import HttpServlet
from clamp import clamp_base

ToolBase = clamp_base("org.python.tools")
```

218 lines of code currently

Bridge to HttpServlet

Implement init method for `javax.servlet.http.HttpServlet`:

```
class WSGIServlet(ToolBase, HttpServlet):
  def init(self, config):
    application_name = config.getInitParameter(
      "wsgi.handler")
    parts = application_name.split(".")
    if len(parts) < 2 or not all(parts):
      raise Exception(...)
    module_name = ".".join(parts[:-1])
    module = __import__(module_name)  # DYNAMIC CODE!
    self.application = getattr(module, parts[-1])
    self.servlet_environ = dict(BASE_ENVIRONMENT)
    self.servlet_environ.update({
      "wsgi.errors": AdaptedErrLog(self)
    })
```

HelloWSGI

Demo project

```
$ pip install bottle mako
$ pip install \
git+https://github.com/jythontools/clamp.git
$ pip install \
git+https://github.com/jythontools/fireside.git
```

https://github.com/jimbaker/hellowsgi

Hello, World code

```
from bottle import Bottle, MakoTemplate, route

simple_app = app = Bottle()
hello_template = MakoTemplate(
  '<b>Hello ${name}</b>!')

@app.route('/hello/<name>')
def index(name):
  return hello_template.render(name=name)
```

Create a single jar

```
$ jython setup.py install singlejar
```

Package a war file

```
$ jar cf hellowsgi.war -C warpack .
```

Run with Jetty

```
$ java -jar jetty-runner.jar hellowsgi.war
```

Run Apache Benchmark

```
$ ab -k -c 20 -n 50000 localhost:8080/hello/world
```

Before JIT and loading modules

```
Percentage of the requests served within a certain ti
  50%      2
  66%      3
  75%      3
  80%      4
  90%     13
  95%     27
  98%     46
  99%     63
 100%   6774 (longest request)
```

Should fix this startup time!

With JIT warmup

```
Percentage of the requests served within a certain ti
  50%      1
  66%      2
  75%      2
  80%      2
  90%      2
  95%      3
  98%      3
  99%      4
 100%     11 (longest request)
```

So steady state performance is definitely decent

Better Java integration

Demo

Existing C integration: JFFI

- Standard project from Java Native Runtime

Existing C integration: JFFI

- Standard project from Java Native Runtime
- Heavily used by JRuby

Existing C integration: JFFI

- Standard project from Java Native Runtime
- Heavily used by JRuby
- Used internally by Jython - example: Posix support

Existing C integration: JFFI

- Standard project from Java Native Runtime
- Heavily used by JRuby
- Used internally by Jython - example: Posix support
- But not part of standard Python ecosystem

Better C integration: JyNI

- Idea: simply add JyNI jar to the Java CLASSPATH to enable C extension API support

Better C integration: JyNI

- Idea: simply add JyNI jar to the Java CLASSPATH to enable C extension API support
- Written by Stefan Richthofer

Better C integration: JyNI

- Idea: simply add JyNI jar to the Java CLASSPATH to enable C extension API support
- Written by Stefan Richthofer
- Now works for a number of packages - tkinter

Better C integration: JyNI

- Idea: simply add JyNI jar to the Java CLASSPATH to enable C extension API support
- Written by Stefan Richthofer
- Now works for a number of packages - tkinter
- Challeng is adding full GC support (!)

Better C integration: JyNI

- Idea: simply add JyNI jar to the Java CLASSPATH to enable C extension API support
- Written by Stefan Richthofer
- Now works for a number of packages - tkinter
- Challeng is adding full GC support (!)
- Next steps: ctypes, cffi, . . .

Better C integration: JyNI

- Idea: simply add JyNI jar to the Java CLASSPATH to enable C extension API support
- Written by Stefan Richthofer
- Now works for a number of packages - tkinter
- Challeng is adding full GC support (!)
- Next steps: ctypes, cffi, ...
- Stefan has applied for GSOC

Python bytecode support

Consider this simple function:

```
def f(a):
    return a + 1
```

Python bytecode disassembly

Results in this code body for function `f`:

```
>>> import dis
>>> import simple
>>> dis.dis(simple.f)
  2           0 LOAD_FAST                0 (a)
              3 LOAD_CONST               1 (1)
              6 BINARY_ADD
              7 RETURN_VALUE
```

Running Python bytecode with ceval.c

```
case BINARY_ADD:
    w = POP();
    v = TOP();
    if (PyInt_CheckExact(v) && PyInt_CheckExact(w)) {
        /* INLINE: int + int */
        register long a, b, i;
        a = PyInt_AS_LONG(v);
        b = PyInt_AS_LONG(w);
        /* cast to avoid undefined behaviour
           on overflow */
        i = (long)((unsigned long)a + b);
        if ((i^a) < 0 && (i^b) < 0)
            goto slow_add;
        x = PyInt_FromLong(i);
    }
```

Just a bit more

```
else if (PyString_CheckExact(v) &&
          PyString_CheckExact(w)) {
    x = string_concatenate(v, w, f, next_instr);
    /* string_concatenate consumed the ref to v *
    goto skip_decref_vx;
}
else {
  slow_add:
    x = PyNumber_Add(v, w);
}
Py_DECREF(v);
```

A little bit more

```
skip_decref_vx:
  Py_DECREF(w);
  SET_TOP(x);
  if (x != NULL) continue;
  break;
```

PyBytecode.java

```
case Opcode.BINARY_ADD: {
    PyObject b = stack.pop();
    PyObject a = stack.pop();
    stack.push(a._add(b));
    break;
}
```

Code

Dive into `ceval.c` and `PyBytecode.java`

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration
- Plan to work on 2.7.x as long as Python 2.7 in wide use

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration
- Plan to work on 2.7.x as long as Python 2.7 in wide use
- Time-based releases, not feature-based

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration
- Plan to work on 2.7.x as long as Python 2.7 in wide use
- Time-based releases, not feature-based
- Every 6 months seems reasonable

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration
- Plan to work on 2.7.x as long as Python 2.7 in wide use
- Time-based releases, not feature-based
- Every 6 months seems reasonable
- Ideally use new workflow CPython is working on

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration
- Plan to work on 2.7.x as long as Python 2.7 in wide use
- Time-based releases, not feature-based
- Every 6 months seems reasonable
- Ideally use new workflow CPython is working on
- Java 9 may also add more features to optimize dynamic languages

Jython 2.7.x, where $x > 0$

- Mostly around performance, Java integration, and of course the usual bug fixes
- Python bytecode compiler for Android, large complex methods
- More hooks for Java integration
- Plan to work on 2.7.x as long as Python 2.7 in wide use
- Time-based releases, not feature-based
- Every 6 months seems reasonable
- Ideally use new workflow CPython is working on
- Java 9 may also add more features to optimize dynamic languages
- Integrating Zippy to provide PyPy-like performance (requires Graal JVM)

Jython 3.x?!

Plan to sprint on language support this week:

- Comes up periodically!

Jython 3.x?!

Plan to sprint on language support this week:

- Comes up periodically!
- Would be nice for unicode strings and bytestrings to have direct correspondence to Java

Jython 3.x?!

Plan to sprint on language support this week:

- Comes up periodically!
- Would be nice for unicode strings and bytestrings to have direct correspondence to Java
- Remove code!

Jython 3.x?!

Plan to sprint on language support this week:

- Comes up periodically!
- Would be nice for unicode strings and bytestrings to have direct correspondence to Java
- Remove code!
- Release schedule: we will get there at some point!

Jython 3.x?!

Plan to sprint on language support this week:

- Comes up periodically!
- Would be nice for unicode strings and bytestrings to have direct correspondence to Java
- Remove code!
- Release schedule: we will get there at some point!
- But target 3.5 or maybe 3.6 of the Python language