

Jython

Python for the Java Platform

Jim Baker, Committer
jbaker@zyasoft.com
twitter.com/jimbaker
zyasoft.com/pythoneering

Jython Milestones

- 9 years ago - Released JPython 1.1
- 8 years ago - Released 2.0
- 7 years ago - Released 2.1
- 2 years ago - Released 2.2, "newcompiler"
- 1 year ago - Released 2.5 alpha
- Now - 2.5.0 in June, 2.5.1 RC3 any day
- Future - frequent, incremental releases

Development

- 8 active committers (= core developers)
- Hudson CI, 6000+ test cases
- Open source Jython book from Apress, November (4 committers + expert user)

Working with C(ore)Python

- Pull in stdlib, then patch
- Now doing the same with docset
- Moving to Mercurial (hg) in advance of CPython
- Started merging stdlib work upstream
- Recently fixed AST bugs in CPython 2.7

Defining...

- Python performance model
- Python memory model
 - Safe publication
 - Visibility
 - Atomicity of ops on collections

Python Community Support

- Bug reporting and build bots -
 - Django, Jinja2, PyAMF, Pylons, SQLAlchemy, SymPy, Twisted, and others
- Filed bugs with us or vice versa
- Cross committers - Pylons, CPython, PyPy
- *Extremely warm and supportive of Jython*

Shared JVM Infrastructure

no longer worry about "Pure Java" - pragmatic first

Security managers set the policy anyway - but want a common code base for setup and running in a server

- Shared (ugly) startup scripts
- Native OS support for functionality like chdir Posix, Windows support through JNA
- JLine

Future Shared JVM Infrastructure

- [http://wiki.jvmlangsummit.com/
Shared_JVM_Infrastructure](http://wiki.jvmlangsummit.com/Shared_JVM_Infrastructure)
- JLine, OS-specific enhancements
- “Son of a Nailgun” - support command line
- Jar package scanner and cache
- Common IR and compiler backend?

panel up next!

Opportunities for Innovating Python

- Concurrency
- Service fabric support
- Gradual typing
- Integration with Java object models, such as
 - clamp project
 - callables \Leftrightarrow single method interfaces

Performance Optimizations

- Small optimizations
- invokedynamic
 - attribute resolution
 - MOP, reduce argparser overhead
- Support advanced functionality

Mini VMs

- cPickle - serialization
- sre - regex

Python Bytecode VM

- Alternative to Java bytecode, to enable functionality or work around limits
- Code objects can use JBC or PBC
- ~1600 LOC - but needs companion work to compile to PBC (use CPython for now)
- Most of the test suite passes

Android, applets - can't readily generate new bytecode on the fly

64K method limits

Introspection, directly run code that emits PBC, extend the execution model

Polymorphic Instructions

- Eliminate large switches
- Polymorphic instructions

indy

Jython Dispatch Today

- Complex dynamic lookup rules with no call site caching (so poor performance)
- Proxy wrappers for all calls into Java methods
- Large blobs of generated bytecode with mangled names

Promise of JSR-292

- Call site caching with deep VM support for our dynamic lookup rules
- Method Handles will eliminate the need for proxies to Java methods
- Anonymous classes will eliminate the need for name mangled generated bytecode

Invokedydynamic on Jython so far

- Doesn't immediately crash anymore :)
- Simple indy calls are able to call into Jython internal dispatch
- JRuby patterns look very applicable

Simple linkDynamic Against Jython

```
CallSite site = new CallSite(caller, name, type);
ThreadState ts = Py.getThreadState();
PyCode func_code = ((PyFunction)ts.frame.getname(name)).func_code;

MethodType oneArgCall = MethodType.make(
    PyObject.class, //return type
    ThreadState.class, // state
    PyObject.class, // arg1
    PyObject.class, // globals
    PyObject[].class, // defaults
    PyObject.class // closure
);
MethodHandle call = MethodHandles.lookup().findVirtual(
    PyCode.class, "call", oneArgCall);
call = MethodHandles.insertArguments(call, 0, func_code, ts);
call = MethodHandles.convertArguments(call, type);
site.setTarget(call);
```

Python Invocation Semantics

- Evaluate target
- Load method object as attribute
- Evaluate arguments, left to right
- Call method object

- Difficulty in reordering/folding chains

Argument Parsing

- Parsing positional and keyword args is expensive
- Each call site typically invokes the same call target, with the same mapping each time
- `indy` - Arg parse on the first invocation, then reuse

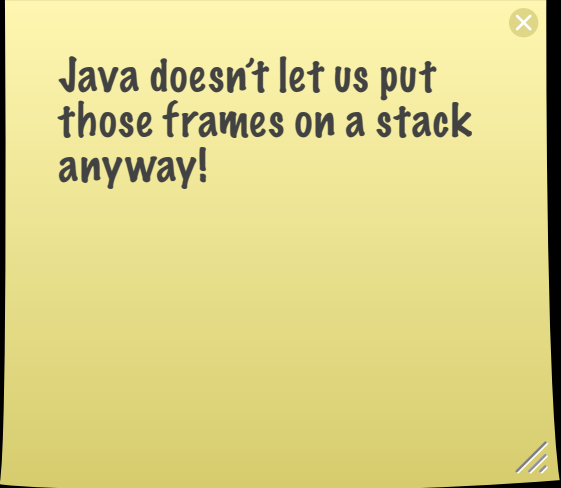
Numeric Python for Java

- Port NumPy to Jython
- Large data set support from Unidata
- FJ parallelism, parallel array support
- Java easier to use to write ufuncs
- Showcase gradual typing

Rationale: large class of problems that benefit from numerical computing not currently being served by high-level langs.

Such as "collective intelligence" apps

Stackless Support



Java doesn't let us put those frames on a stack anyway!

- Already allocate frames dynamically
- Leverage our Python Bytecode VM
 - Implement a trampoline
- Future: as we get TCO, we can move back to JBC - but let's advance for now

Joe

- Stripped down Jython, for experimentation and fun
- Think Surinx
- Possible alternative implementation language
- Tobias' baby

Questions?

jbaker@zyasoft.com

twitter.com/jimbaker

zyasoft.com/pythoneering