Java for z/TPF

Applications Development Subcommittee

Jim Johnston
Software Engineer

IBM **z/TPF**

April 12, 2016

Disclaimer

Any reference to future plans are for planning purposes only. IBM reserves the right to change those plans at its discretion. Any reliance on such a disclosure is solely at your own risk. IBM makes no commitment to provide additional information in the future.

3 Minutes	**Why Java on z/TPF?**
13 Minutes	**How do we get there?**
1 Minute	**Get started with Beta Today!**
3 Minutes	**Q&A**

The Problem Facing the z/TPF Community

z/TPF assembler skills are difficult to find. The skills the community has are reaching retirement.

How do we maintain vitality going forward and rebalance skills requirements?

Bring Java™ to z/TPF

Large Skill Base

Platform Independent (easy to port packages to different platforms)

Widely used in enterprise applications

Easy to learn

A Java developer can create new or extend existing z/TPF applications using Java without requiring any z/TPF knowledge.

Standard Development

Seamless Integration

z/TPF Management

A Java developer can develop and unit test for z/TPF using a standard development environment

- Many Java™ Platform Standard Edition applications exist which work as-is
- Off-the-shelf Java applications tend to be feature rich

Plenty of Java IDEs

Eclipse
TPF Toolkit
NetBeans
JDeveloper
IntelliJ IDEA
Maven (IDE extension)
Other IDE plug-ins (from Eclipse Marketplace)

Standard Tools

Vast Java Libraries & Utilities

Java SE Packages – java/lang, java/util, java/io

Reusable Utilities - Apache Commons, Google Guava

Test Driven Development – JUNIT, mockito

Logging – slf4j, log4j

JSON & XML – Jackson, JDOM, Xerces

Protocols – MQ, HTTP, REST, SOAP

The lab is working towards certifying Java Standard Edition to ensure Java compliance and fulfill Java's promise of "build once, run anywhere"

Standard Development

Seamless Integration

z/TPF Management

A Java developer can call existing z/TPF applications without transformation code

Standard Development

Seamless Integration

z/TPF Management

“As-Is” Java Native Interface (JNI)

Some drawbacks with JNI…

- JNI Transformation code can get complex very quickly
- Need to be careful to avoid performance pitfalls
- Java programmer needs to know z/TPF

```
#include <service_newService.h>

JNIEXPORT jobject JNICALL Java_service_newService_callService

 (JNIEnv *env, jobject instance, jstring name, jlong id, jobject addrObj)  {

    //Transformation calls to convert Java parms to native types
    char *natName = (char *) env->GetStringUTFChars(name, NULL);
    jclass addrClass = env->GetObjectClass(addrObj);
    jstreetfieldID = env->GetFieldID(addrClass, "street", "Ljava/lang/String;");
    jstring street = env->GetObjectField(addrObj, jstreetfieldID);
    char *natStreet = (char *) env->GetStringUTFChars(street, NULL);
    tpf_fsysc();              //zTPF native unique calls start here.
    dblookup(natName, natStreet);              //possible TPF database calls
    tpf_cresc();    //further processing to arrive at final response

   //Transformation calls to create Java result from native result
    return jobjResult;
}
```

This example only shows fetching one field.

Java programmer now needs to know z/TPF or z/TPF programmer would need to know JNI.

"AS-IS" with JNI

Standard Development

Seamless Integration

z/TPF Management

“To-Be” TPF Native Services (TNS)

TNS: Java calling z/TPF native code

- Java programmer doesn’t need to know z/TPF
- z/TPF programmer doesn’t need to know Java
- No changes to existing native z/TPF applications

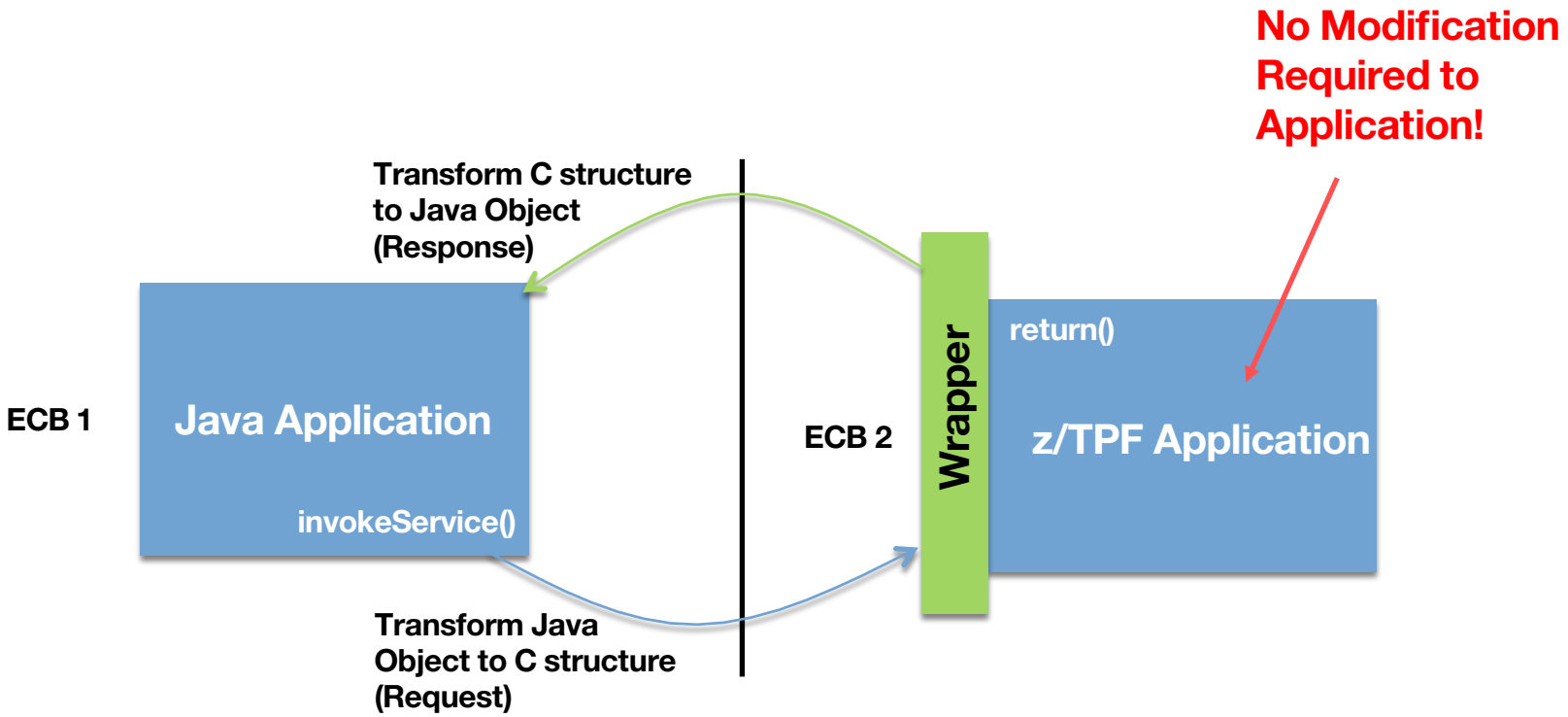
z/TPF native code calling Java will be post-GA

```
ObjectFactory serviceFactory = new ObjectFactory();

newService TPFservice = serviceFactory.newService();
Request request = serviceFactory.createRequest();
Address addr = serviceFactory.createAddress();
request.setName("Jim");
request.setID(1123456);
addr.setstreet("18 Laurel Rd");
addr.setcity("Orlando");
addr.setzip("32827");
request.setAddress(addr);
TPFservice.setRequest(request);

try {
    TPFservice.invokeService();     //Blocks until request fulfilled
    System.out.println(TPFservice.getResponse().getscore());
} catch  (TNSException anExObj)  {
    // error occurred … example, ECB exited from native code, or possibly conversion of errno values
}
```

Familiar JAXB setters/getters!

Invoke takes care of serialization/deserialization of Java objects on native side.

After TNS for Java



TNS – Java to Native

Standard Development

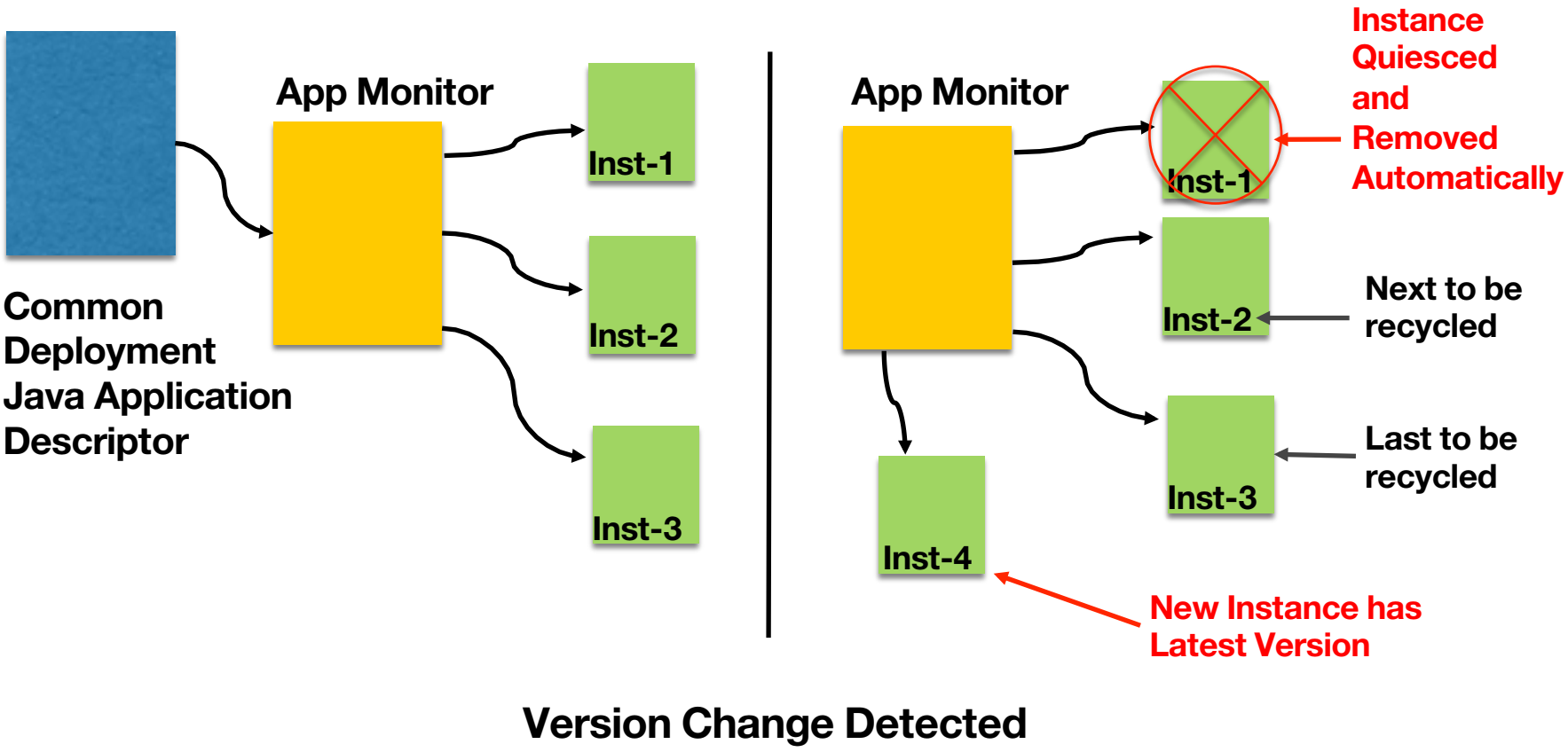
Seamless Integration

z/TPF Management

A z/TPF operator can manage Java workloads using familiar z/TPF commands

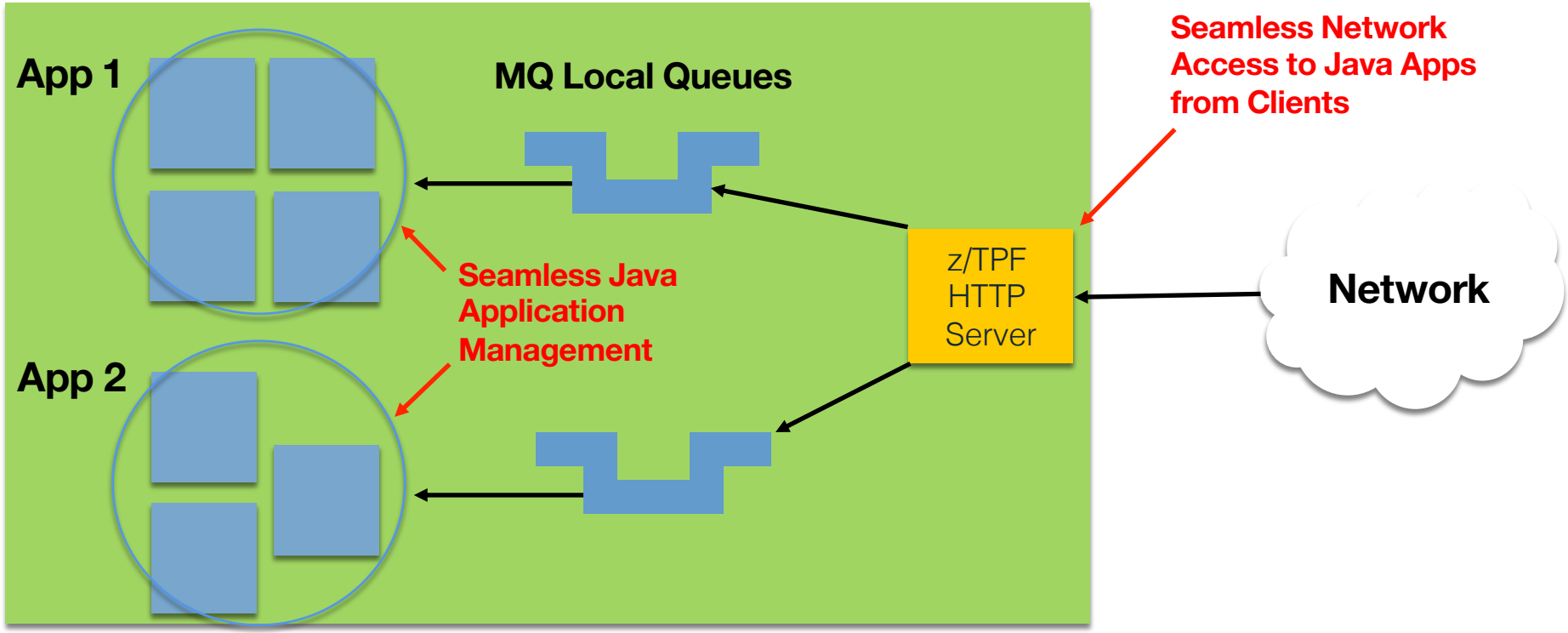
- Secure Java program loads via z/TPF Loader
- Dynamic and non-disruptive program loads via E-type loader
- Long-Running Java Applications

Long-Running Java Applications



Version Change Detected

Long-Running Java Applications



Standard Development

Seamless Integration

z/TPF Management

We want to enable Java Developers on z/TPF but…

Key Takeaway

Need Java and z/TPF to Play Nice

Key Takeaway

Getting started with Java on z/TPF

z/TPF Java Sponsor Users Community

- Multiple customers are involved and collaborating
- Beta Code available for download
- Presentations available in community
- Talk to your CSR to become a sponsor user, it's not too late!

Summary

- The community wants to rebalance skill distribution to adjust for retiring HLASM programmers
- Integration of Java on z/TPF is an evolution
- Support of Java standards enables the Java ecosystem to apply to z/TPF
- Use TNS to call native code from Java without requiring z/TPF knowledge
- Support familiar z/TPF program management facilities for Java
- Java beta is now available

Thank you!

Questions or comments?

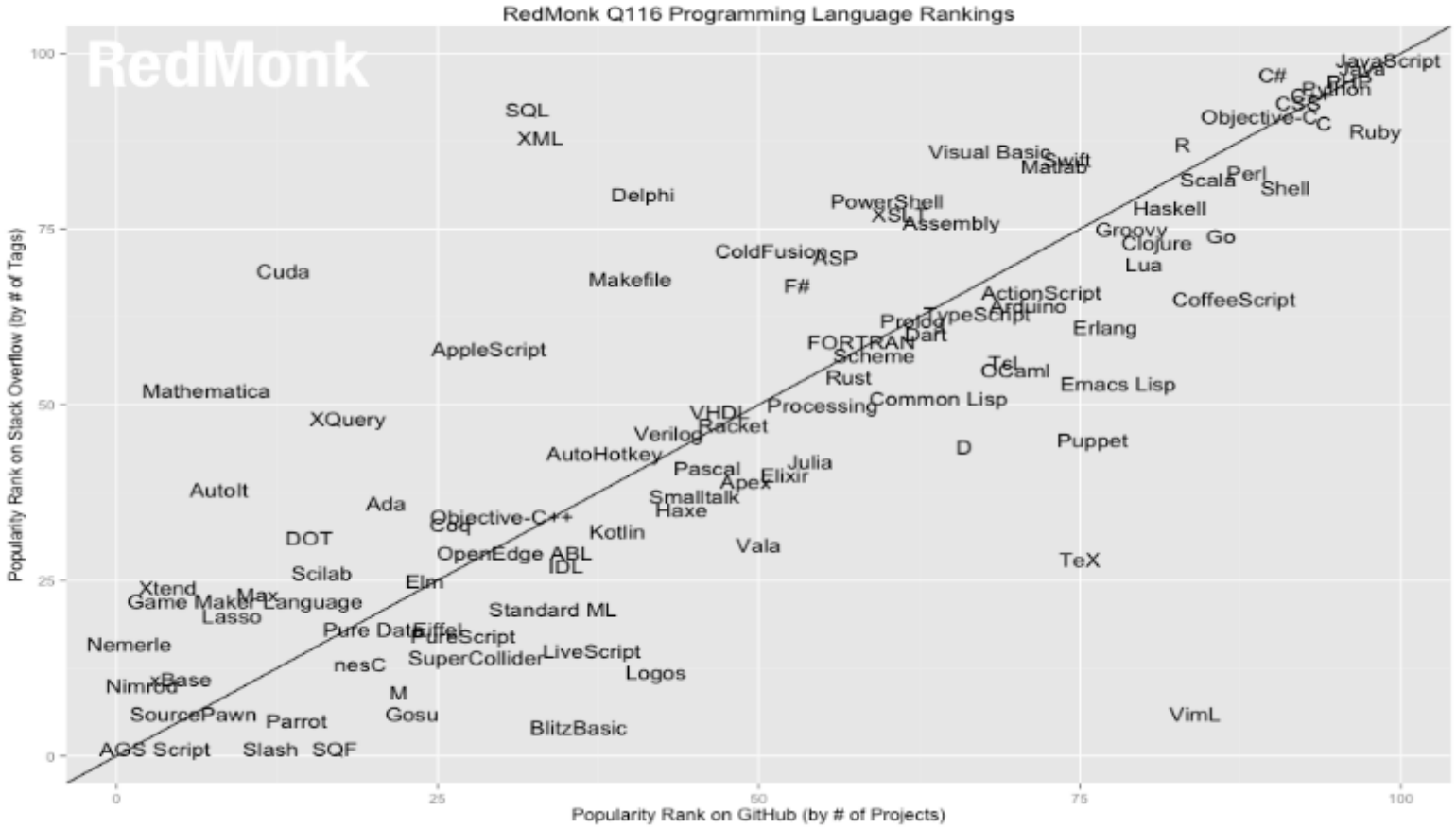
Trademarks

- IBM, the IBM logo, ibm.com and Rational are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.
- Java and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates

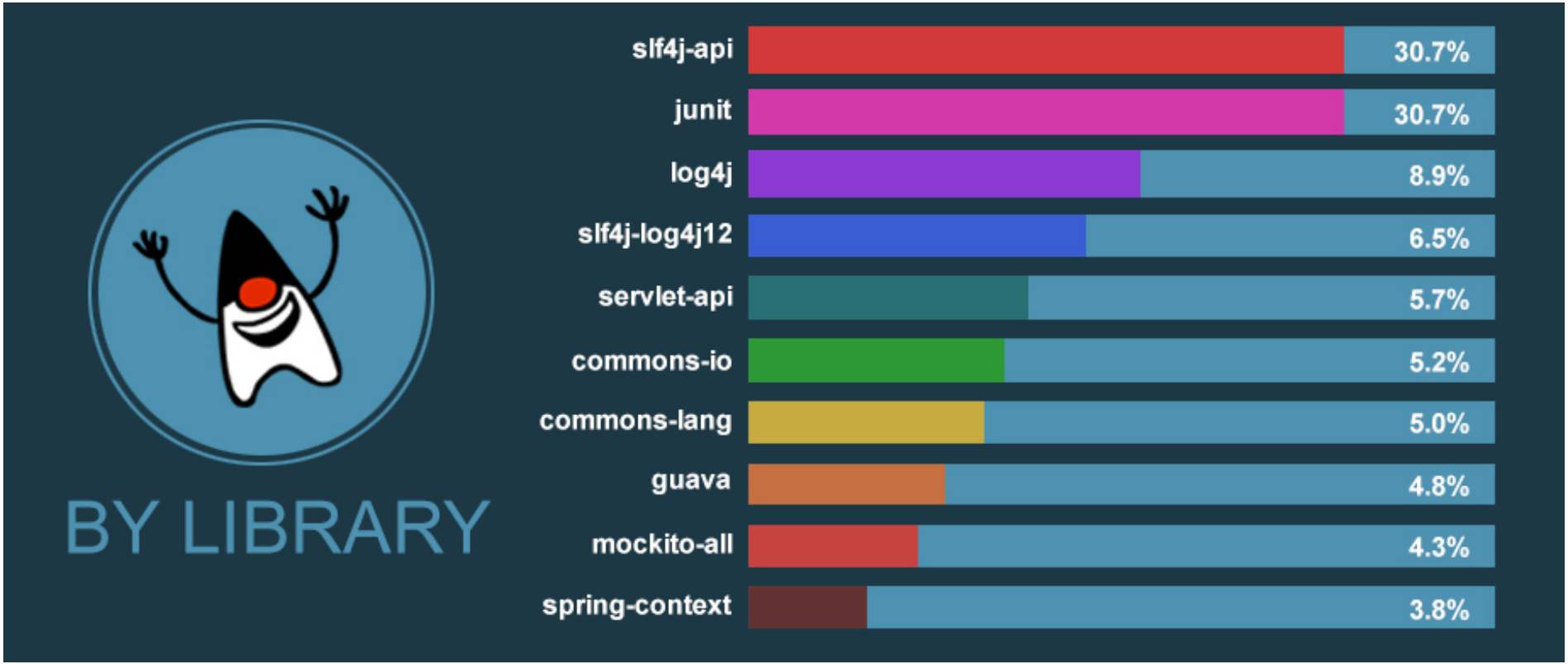
Notes

- Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.
- All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.
- This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.
- All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.
- Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.
- Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.
- This presentation and the claims outlined in it were reviewed for compliance with US law. Adaptations of these claims for use in other geographies must be reviewed by the local country counsel for compliance with local laws.

Backups


RedMonk Q116 Programming Language Rankings
RedMonk
Popularity Rank on Stack Overflow (by # of Tags)
Popularity Rank on GitHub (by # of Projects)
0
25
50
75
100
SQL
XML
Delphi
Cuda
Makefile
ColdFusion
ASP
F#
PowerShell
XSLT
Assembly
Visual Basic
Matlab
Swift
C#
JavaScript
Java
PHP
Python
CSS
Objective-C
C
Ruby
R
Scala
Perl
Shell
Haskell
Groovy
Clojure
Go
Lua
ActionScript
Arduino
CoffeeScript
TypeScript
Erlang
Prolog
Dart
FORTRAN
Scheme
Tcl
OCaml
Rust
Emacs Lisp
Common Lisp
Processing
VHDL
Racket
Verilog
AutoHotkey
Pascal
Julia
Apex
Elixir
Smalltalk
Haxe
D
Puppet
AppleScript
Mathematica
XQuery
AutoIt
Ada
Objective-C++
Coq
Kotlin
DOT
OpenEdge ABL
IDL
Vala
TeX
Scilab
Elm
Xtend
Game Maker Language
Max
Lasso
Standard ML
Pure Data
Eiffel
PureScript
LiveScript
SuperCollider
Logos
Nemerle
nesC
Nimrod
xBase
M
Gosu
SourcePawn
Parrot
BlitzBasic
VimL
AGS Script
Slash
SQF

source: www.tiobe.com



source: www.tiobe.com

Programming Language	2016	2011	2006	2001	1996	1991	1986
Java	1	1	1	3	30	-	-
C	2	2	2	1	1	1	1
C++	3	3	3	2	2	2	8
C#	4	5	6	10	-	-	-
Python	5	6	7	25	16	-	-
PHP	6	4	4	22	-	-	-
Visual Basic .NET	7	189	-	-	-	-	-
JavaScript	8	9	9	7	32	-	-
Perl	9	7	5	4	3	-	-
Objective-C	10	8	42	-	-	-	-