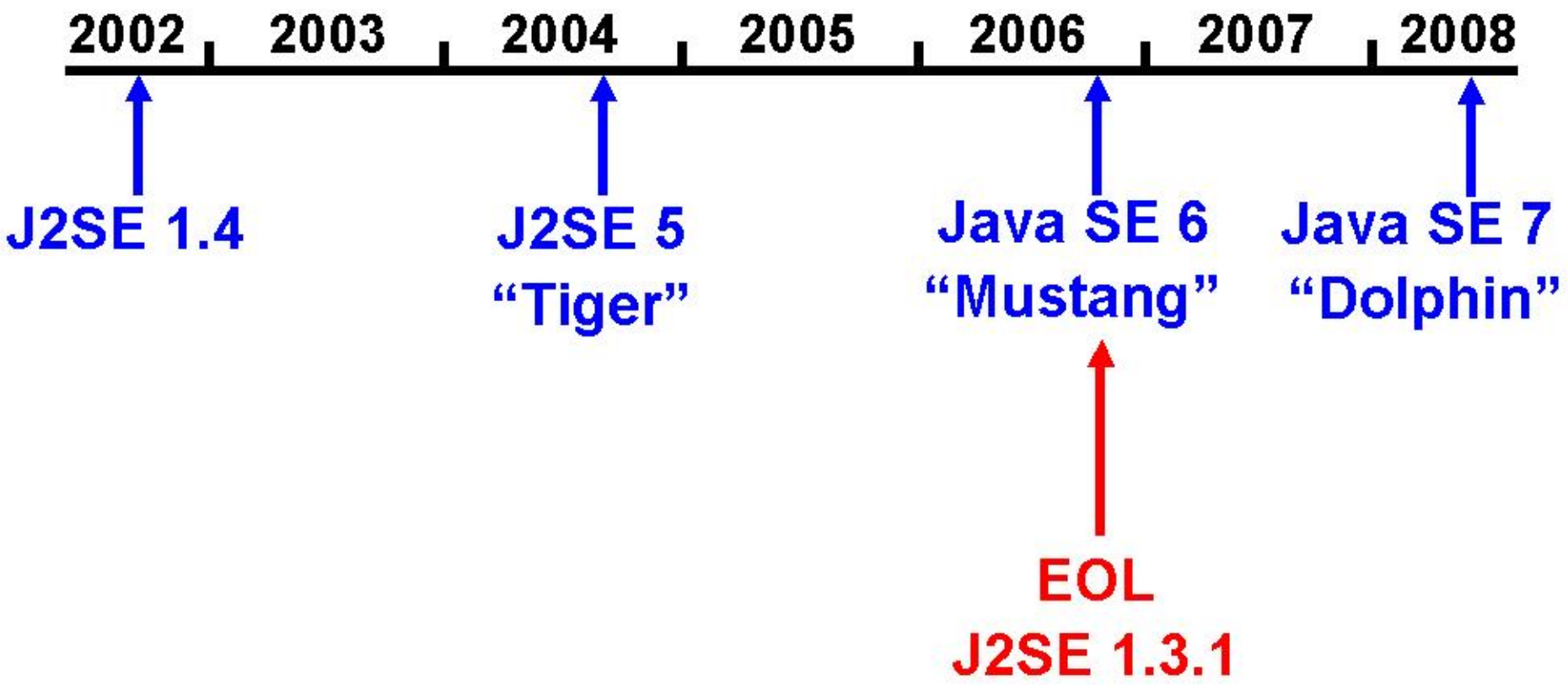

Java SE Language Features :
Today and Tomorrow
양수열
Java Champion
Inpion consulting

- Why switch to Java SE 5 / Java SE 6 ?
- New Features in Java SE 6
- What about Java SE 7?
- Resources and Summary

Java SE road map



1.4.0	Merlin	2002/2/13
1.4.1	**Hopper**	**2002/10/16**
1.4.2	**Mantis**	**2003/5/29**
5.0	Java SE 5	2004/9/30
6	Java SE 6	2006/10
7	Java SE 7	2008/H2

No more maintenance releases (5.1, 6.1, ...)

Update releases every 8–16 weeks (latest: 5.0u9)

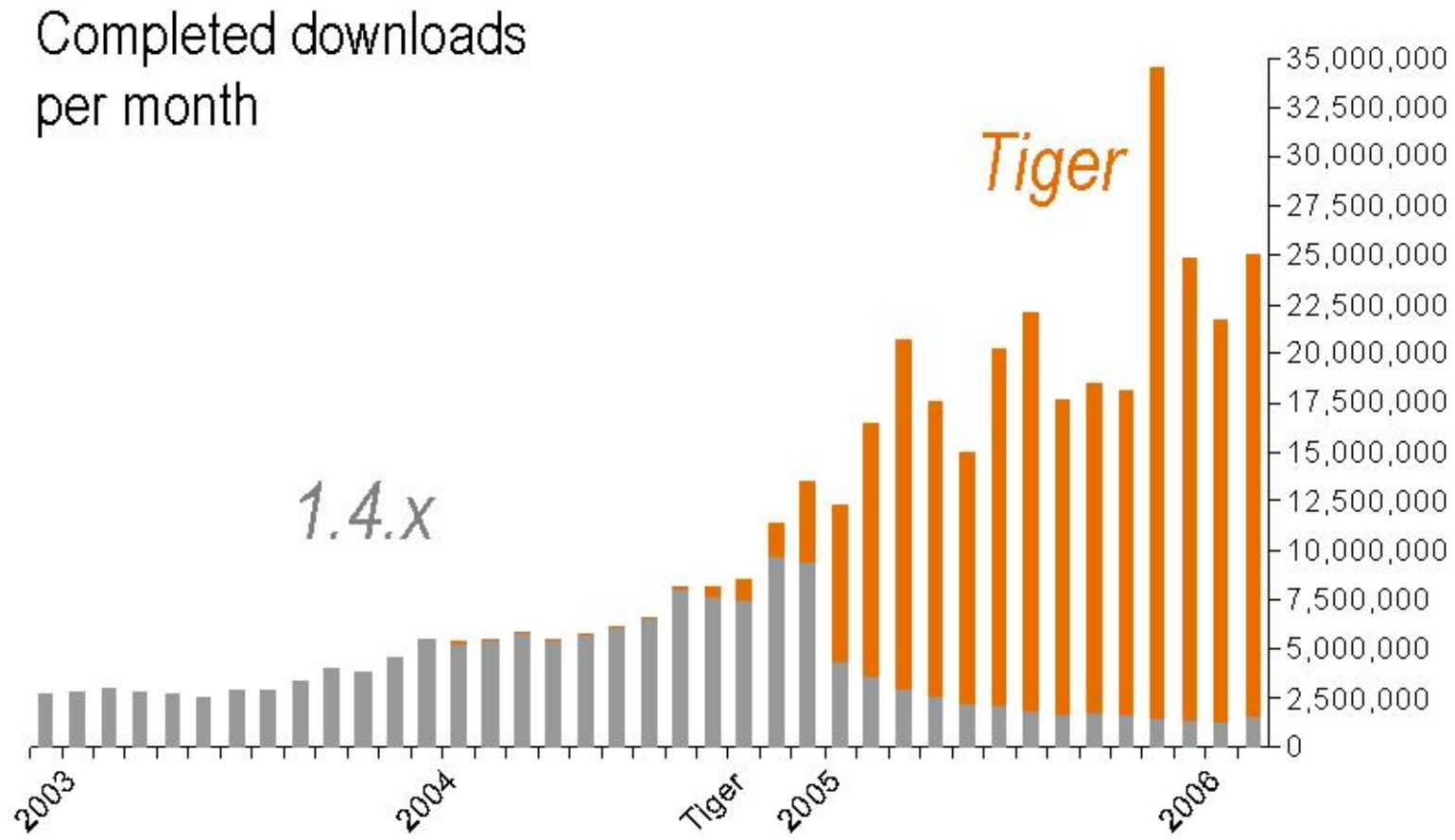

JMX™
BigDecimal
z-ordering
jstat
RMI dynamic proxies
updates
Generics
Gnome Skins
jps
JDBC™ Rowsets
Autoboxing
extended for loop
faster startup
jconsole
printf
JVM™ sharing
synth L&F
SAX 2.0
Concurrency utilities
Unicode Surrogates
importing constants
OpenGL
IP reachability
Ocean L&F
SASL
performance
XAWT
unsynchronized StringBuffer
improved cookie support
XML Schema
apt
DOM 3
JVM Monitoring
fatal error handlers
Stack trace API
Remote JMX
improved footprint
varargs
swing printing
AMD64
Enumerated types
XDnD
Packed JARs
metadata
OCSP
New Memory Model
scanning
JVMPI Profiling
performance ergonomics
javaSE5

It's Time to Upgrade!

Tiger is *Everywhere*
Tiger is *Stable*
Tiger is *Fast*

"1.4 apps run great on 5.0!"

Tiger Is Everywhere



Tiger Is Everywhere

Over

262,295,496

downloads served!

Tiger Is Everywhere

Pre-installed on

> 60 %

of new PCs

Tiger is Stable

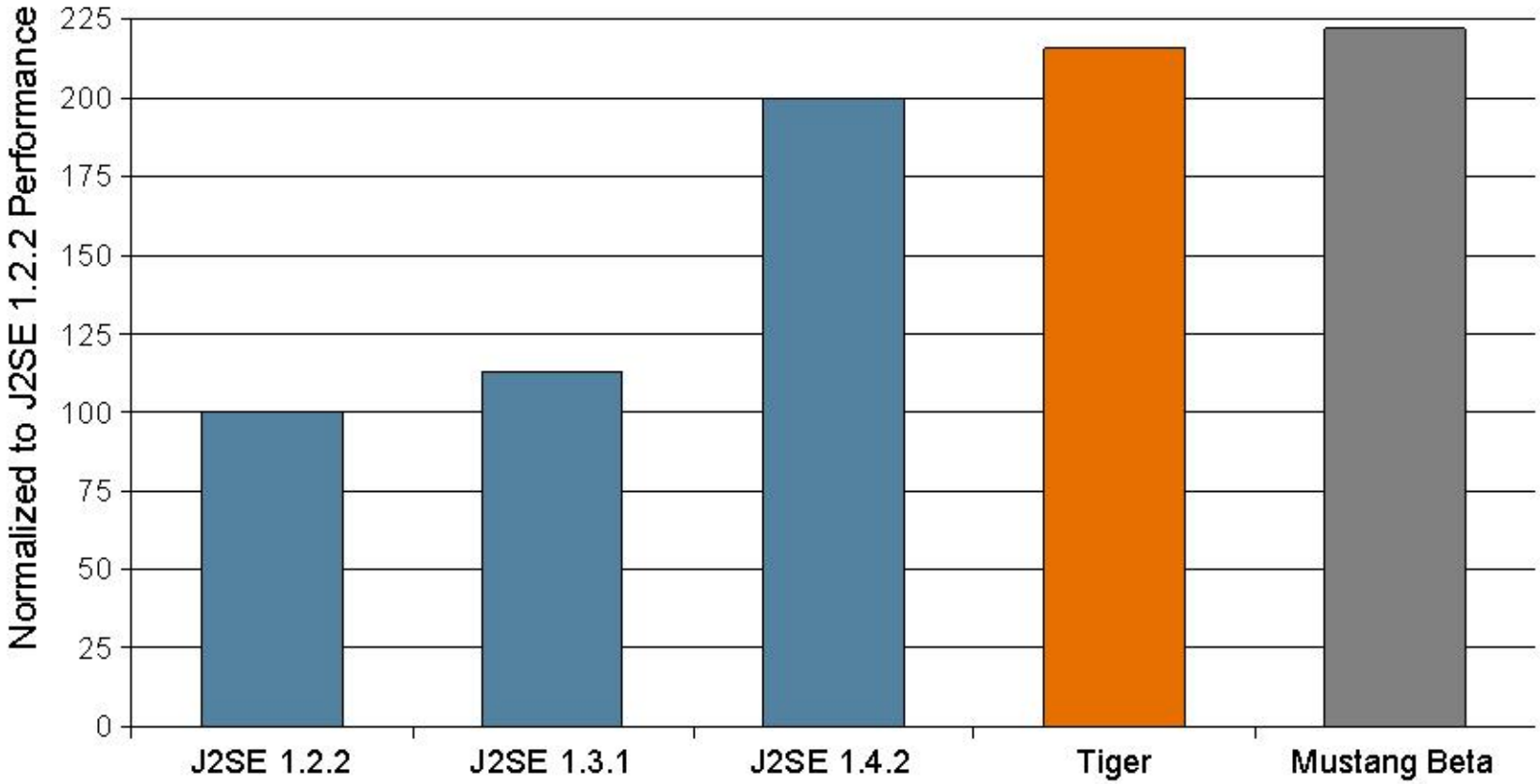
$\geq$ 99.999 %

availability

Previous best: 99.98% for 1.4.2

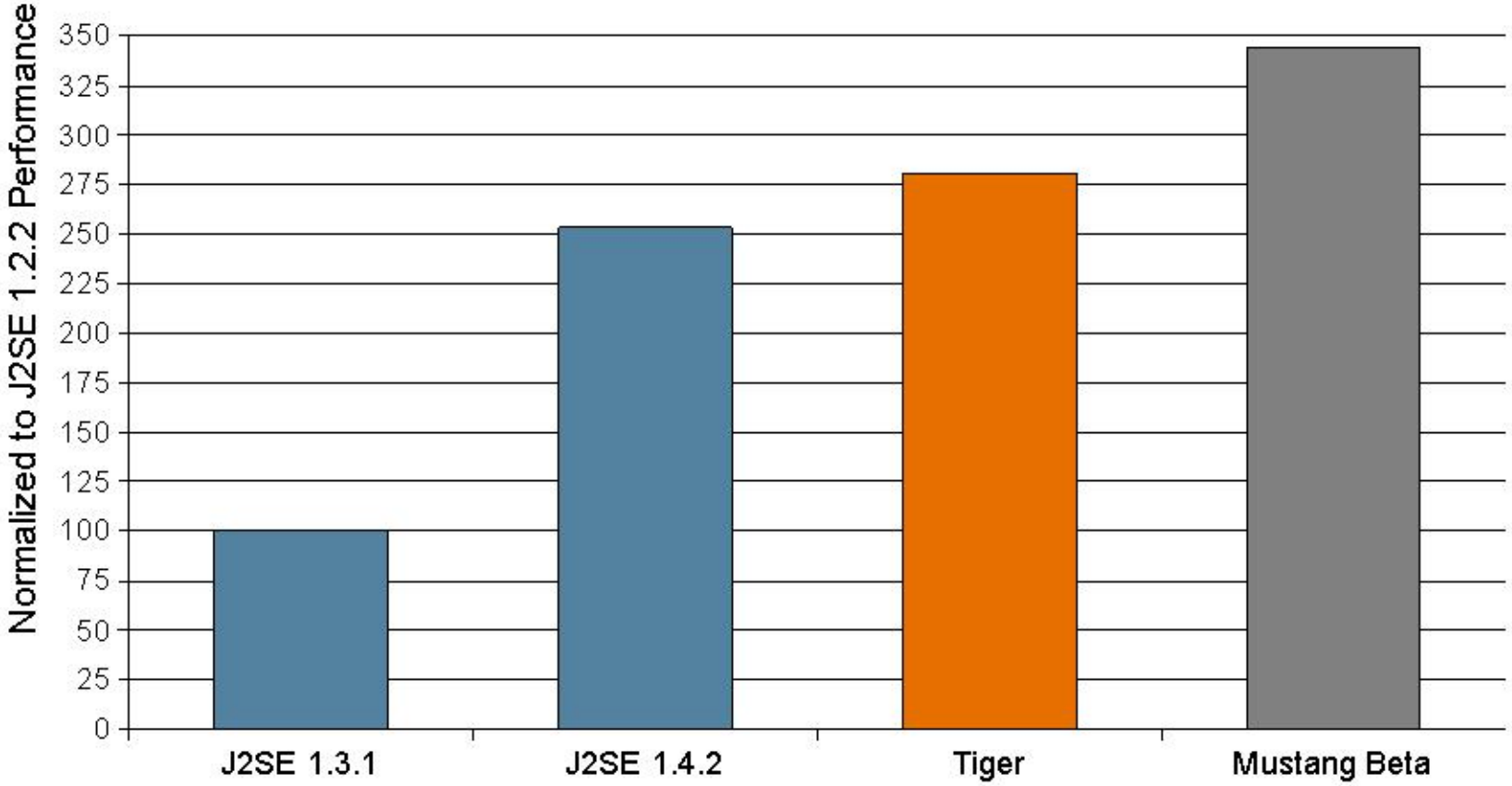
Tiger is Fast

Client Benchmark: SwingMark

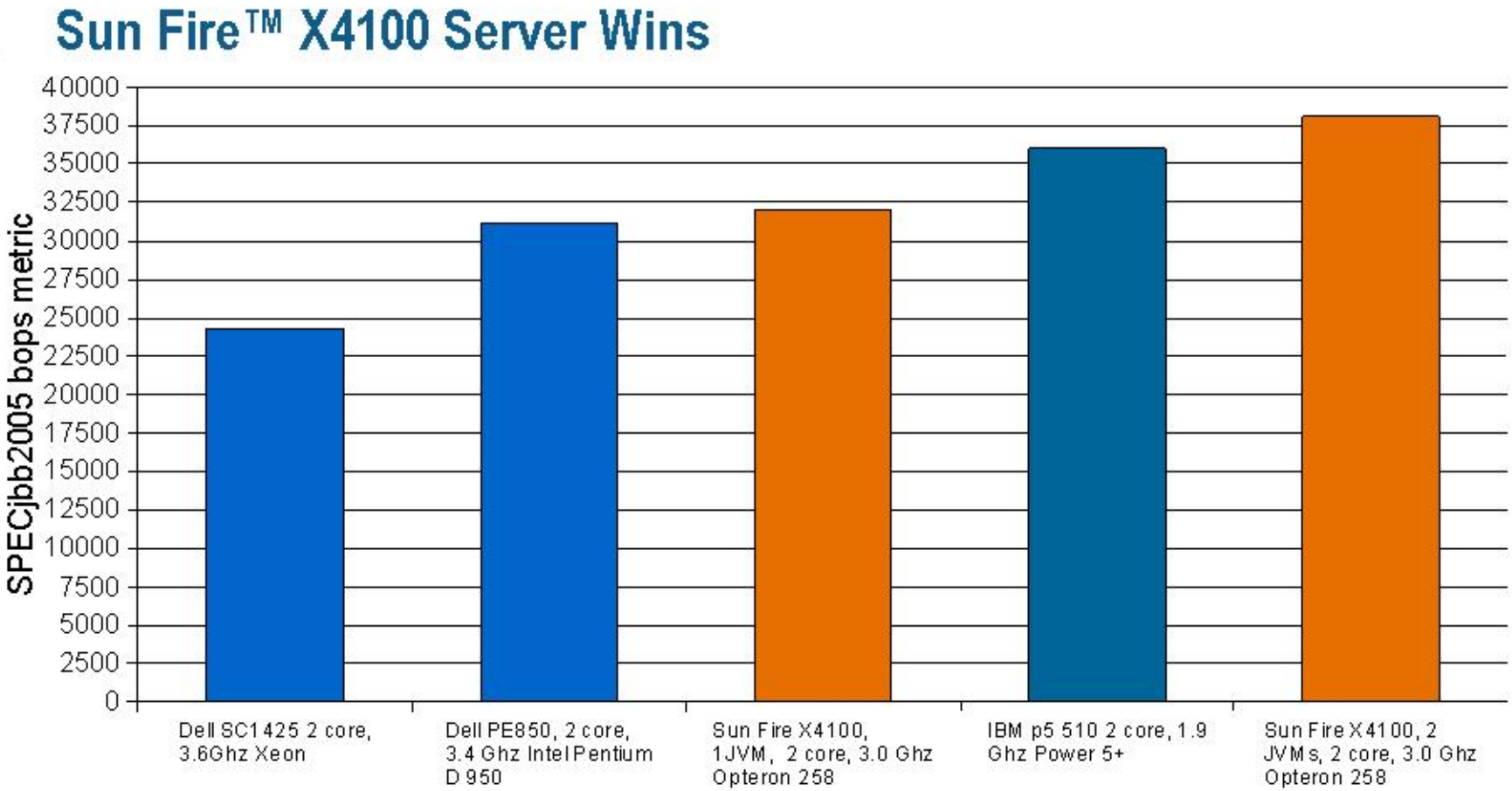


Tiger Is Fast

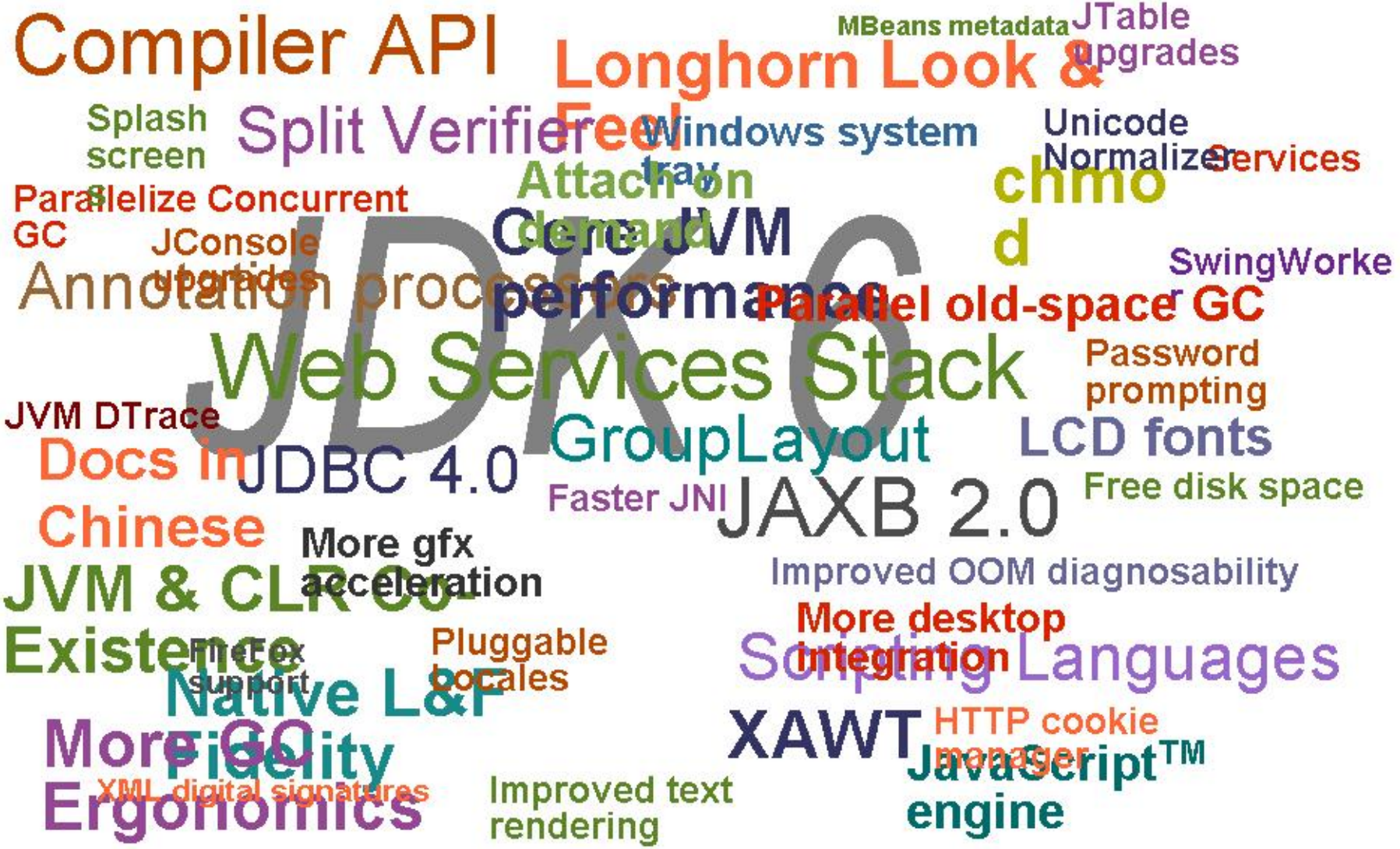
Server Benchmark: SPECjbb2000



x64 SPECjbb2005 Performance



SPECjbb2005 Sun Fire X4100 (2 chip, 2 core, 2 threads) 32,018 SPECjbb2005 bops, 32,018 SPECjbb2005 bops/JVM, Sun Fire X4100 (2 chip, 2 core, 2 threads) 38,090 SPECjbb2005 bops, 19,045 SPECjbb2005 bops/JVM, submitted for review, IBM eServer p5 510 (2 chips, 2 cores, 4 thread) 36,039 bops, 36,039 bops/JVM, Dell SC1425 (2 chips, 2 cores, 4 thread) 24,208 SPECjbb2005 bops, 24,208 SPECjbb2005 bops/JVM, Dell PE 850 (1 chip, 2 cores, 2 thread) 31,138 SPECjbb2005 bops, 31,138 SPECjbb2005 bops/JVM. SPEC® and the benchmark name SPECjbb™ are trademarks of the Standard Performance Evaluation Corporation. Competitive benchmark results stated above reflect results published on www.spec.org as of May 11, 2005. For the latest SPECjbb2005 benchmark results, visit http://www.spec.org/osg/jbb2005.


JDK 6
Compiler API
Longhorn Look & Feel
MBeans metadata
JTable upgrades
Splash screen
Split Verifier
Windows system tray
Unicode Normalizer
Services
Parallelize Concurrent GC
Attach on demand
chmod
JConsole upgrades
Core JVM performance
SwingWorker
Annotation processing
Parallel old-space GC
Web Services Stack
Password prompting
JVM DTrace
GroupLayout
LCD fonts
Docs in Chinese
JDBC 4.0
Faster JNI
JAXB 2.0
Free disk space
More gfx acceleration
Improved OOM diagnosability
JVM & CLR Co-Existence
More desktop integration
Pluggable locales
Scripting Languages
Native L&F
XAWT
HTTP cookie manager
More Fidelity
JavaScript™ engine
XML digital signatures
Improved text rendering
Ergonomics

Java SE 6, JSR 270 Status

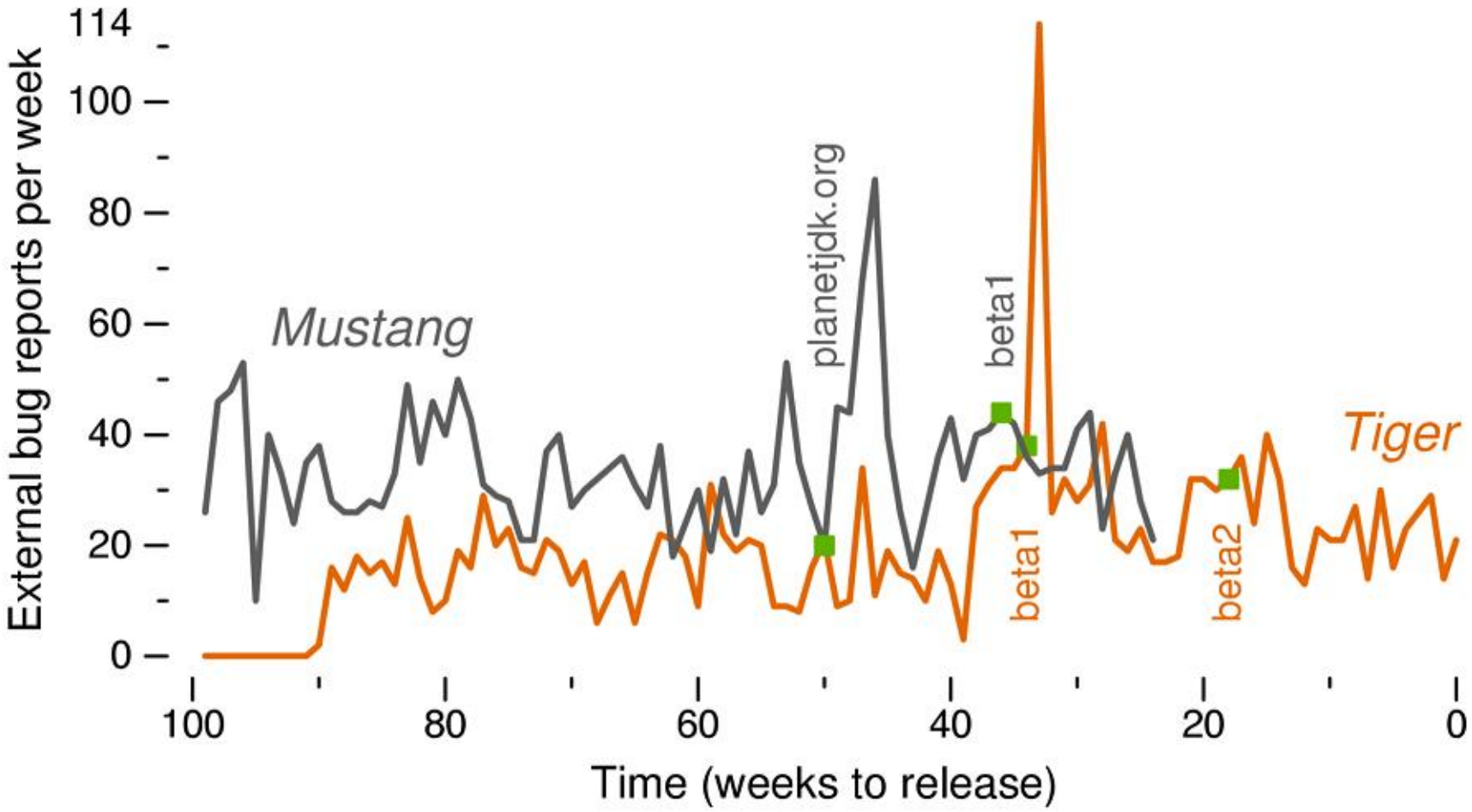
- Public Draft posted 2006/8/22
  - `http://jcp.org/en/jsr/detail?id=270`
- Changes from original JSR submission:
  - Dropped JSR 260: Javadoc™ tool Tag Update
  - Dropped JSR 268: Smart-Card I/O API
- Additional changes since Early Draft Release:
  - Dropped "Reflective access to parameter names" feature (#402)
  - Added `javax.swing.GroupLayout`

JDK 6 Milestone Releases

Beta1	2006/02/16
Beta2	2006/06
Release Candidate	2006/09
Final Release (FCS)	2006/10

JDK 6 Snapshots

Weekly Binary Snapshots: Experimental Results



JDK 6 Snapshot: Conclusion

- *Effects of weekly snapshot builds*
  - *More feedback throughout the release (≥ doubled)*
  - *Formal beta releases don't yield more feedback*
- *Formal beta releases are very expensive*
  - *Forked line of development*
- *Upshot: JDK 6 beta2 very lightweight*

JDK Software Community Status

community.java.net/jdk

- 100,000+ visitors/month
- 1000+ members
- 300+ contributors
- 270+ contributions so far
- Projects: Tiger, Mustang, MVM, Dolphin
- Starter bugs
- Active technical forums and blogs: planetjdk.org
- Real culture change within Sun


I NEED
YOU
for JDK
DEVELOPMENT
community.java.net/jdk

Java SE 5 Features Overview

Seven Major New Language Features

- Generics
- Autoboxing/Unboxing
- Enhanced for loop ("foreach")
- Type-safe enumerations
- Varargs
- Static import
- Metadata

Annotation Processing Tool

- Annotated base file generates derived files
  - **`apt`** is tool to generate derived files
  - Derived files can be new source files, deployment descriptor, etc
- Uses annotation processors
  - Processors can be developed using the **`com.sun.mirror`** packages
  - **`process()`** method processes specific annotations
  - Access to **`Filer`** to create derived files
- **`apt`** can compile the derived files if required

JDK 6 Features Overview

JDK 6 Component JSRs

202: Class File Update (split verifier)

199: Compiler API

269: Annotation Processors — *Ease of Development*

221: JDBC™ 4.0

223: Scripting

105: XML Digital Signature

173: Streaming API for XML — *XML*

222: JAXB 2.0

250: Common Annotations

181: WS Metadata — *Web Services*

224: JAX-WS 2.0

See JSR 270 at http://jcp.org

Top Ten List

Interesting Stuff That You Might've Missed...

10. Attach-on-demand monitoring
9. JConsole plugin API
8. jhat OQL (jmap for heap dump)
7. Solaris™ Dynamic Tracing (DTrace) support
6. Annotation processing done by javac
5. Class-path wildcards
4. Free disk-space API
3. Password prompting
2. `javax.swing.GroupLayout`
1. JAX-WS can do RESTful web services

202: Class File Update

JDK 6 Component JSRs

Primary change: Split verification

- Adopted from Java 2ME™ platform
- Verifier checks compiler-generated assertions instead of generating and checking assertions itself

Why?

- Simplicity
- Performance
- Sharing code between Java ME & Java SE

199: Compiler API

JDK 6 Component JSRs: Ease of Development

- Invoke a Java language compiler programmatically

```
import javax.compiler.*;

Compiler c = Compiler.newInstance();
c.setOutputDirectory(new File("build/classes"));
c.setSourcePath(new File("src"), new File("gensrc"));
c.setOption("Xlint", "all");
c.run(new File("src/Main.java"));
```

Free Disk Space

- Discover the amount of free space in a filesystem and for methods to manipulate file attributes.

```
void safeCopy(File src, File dstDir)
    throws IOException
{
    //bytes available to this virtual machine
    //on the partition
    if (src.length() > dstDir.getUsableSpace())
        throw new IOException("Insufficient space");
    doCopy(src,
           new File(dstDir.getParent(),
                    src.getName()));
}
```

Free Disk Space

```
void df(File dir) {
    out.format("Total MB    Used    Free  Use%%%n");
    //size of the partition
    long t = dir.getTotalSpace();

    //unallocated bytes in the partition
    long f = dir.getFreeSpace();
    out.format("  %6d  %6d  %6d   %2d%%%n",
               t >> 20,(t - f) >> 20,
               f >> 20,
               ((t - f) * 100) / t);
}

% java DF /a
Total MB    Used    Free  Use%
   32766   28632    4134   87%
%
```

Password Prompting

- Define a new API for interactive console I/O that can disable character echoing while reading passwords

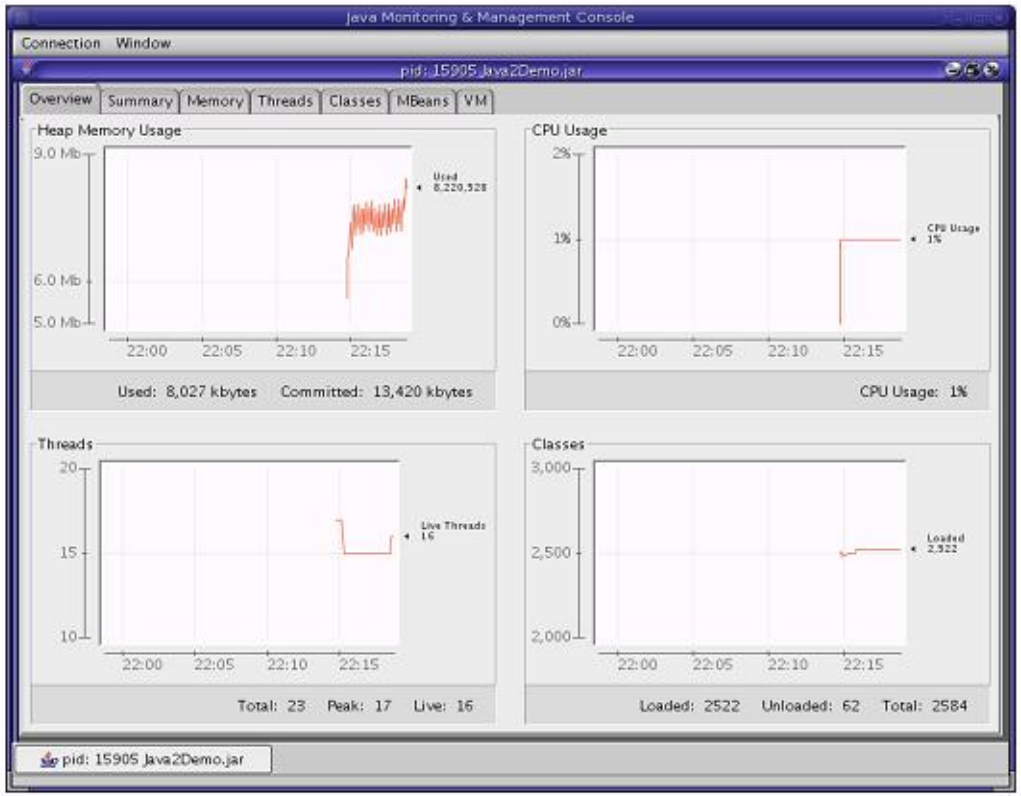
```
import javax.security.auth.callback.*;

class ConsoleCallbackHandler
    implements CallbackHandler
{
    public void handle(Callback[] callbacks) {
        Console cn = System.console();
        for (Callback cb : callbacks) {
            if (cb instanceof NameCallback) {
                String nm = cn.readLine("Username: ");
                ((NameCallback)cb).setName(nm);
            } else if (cb instanceof PasswordCallback) {
                char[] pw = cn.readPassword("Password: ");
                ((PasswordCallback)cb).setPassword(pw);
} }    }   }
```

Class-Path Wildcards

```
% javac -cp .:./jaxb/lib/jaxb-api.jar\
  :./jaxb/lib/jaxb-impl.jar\
  :./jaxb/lib/jsr173_1.0_api.jar\
  :./jaxb/lib/activation.jar \
  Sum.java
```

```
% javac -cp '.:./jaxb/lib/*' Sum.java
```

JConsole improvements



- Reworked UI
- Attach-on-demand
- Deadlock detector

Web Services Client Stack

JSRs 105,173,181,222,224,250

- We listened to you!
- Subset of Java EE 5 stack (JAX-WS 2.0 & JAXB 2.0).
- Client focused, but also lightweight server.
- Includes updated core XML stack.
- Smaller footprint for your applications.
  - JAX-WS 2.0 runtime is included in the JRE.
  - With annotations, no need to generate stubs or ties.
- Use cases: server notifications, callbacks, web service clients, management and testing.
- No more embedded servlet containers and other complexities.

Example: Simple Publishing

One method call and you're done!

```
1. @WebService
2. public class MyEndpoint {
3.     public String sayHello(String s) { ... }
4. }

5. MyEndpoint impl = new MyEndpoint()
6. EndpointFactory f = EndpointFactory.newInstance();
7. f.publish("http://localhost:8080/hello", impl);
```

- **No deployment step**
- **WSDL/schema generated at runtime**
- **Default binding is SOAP 1.1/HTTP**

Asynchronous Client API

When you don't want your application to freeze…

- Async methods generated on-demand
- Two models:
  - Polling: You ask for the response whenever you're ready
  - Callback: Notification as soon as the response arrives
- No need to spawn threads in the application
- No changes needed on the server
- For wscompile in the client use:
  - <enableAsyncMapping>true</enableAsyncMapping>
  - Generate async polling and callback operations & the synchronous method when it compiles a WSDL file.

Key Types When Using Async

Using Java language generics

- Future<T>
  - A promise to deliver a value of type T when it becomes available
  - Use get() :
    - If result ready, result returned
    - If not ready, calling thread blocked
- Response<T>
  - Extends Future<T>
  - Contains a value of type T plus the JAX-WS 2.0 context
- AsyncHandler<T>
  - User-written handler for callback notifications
- Future<?>
  - Useful only for cancelling the invocation

Example: Polling

```
1. @WebService
2. public interface CreditRatingService {
3.
4.    // sync operation
5.    Score getCreditScore(Customer customer);

6.    // async operation w/polling
7.    Response<Score>
8.      getCreditScoreAsync(Customer customer);

9.    // async operation w/callback
10.    Future<?>
11.      getQuoteAsync(Customer customer,
12.              AsyncHandler<Score> handler);
13.}
```

Example: Polling Client

```
1. CreditRatingService svc = ...;

2. Response<Score> response =
3.    svc.getCreditScoreAsync(customerFred);

4. // client app does other things...

5. // ready to deal with the response
6. // no cast needed, thanks to generics
7. Score score = response.get();

8. // or use
9. // Score score = response.get(10L,TimeUnit.SECONDS);
10.// to wait 10 seconds
```

Example: Callback

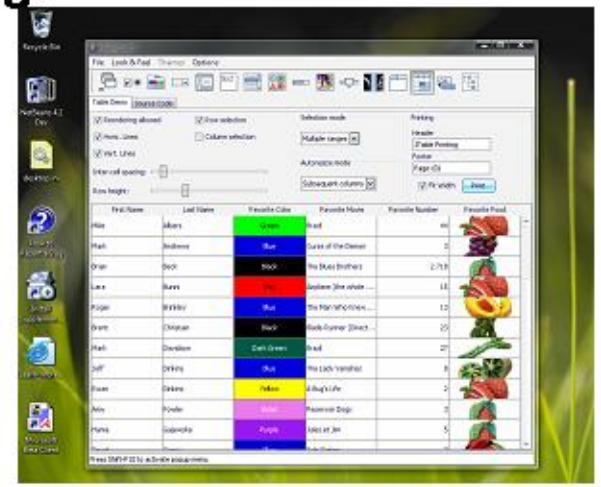
```
1. @WebService
2. public interface CreditRatingService {
3.
4.     // sync operation
5.     Score getCreditScore(Customer customer);

6.     // async operation w/ polling
7.     Response<Score>
8.       getCreditScoreAsync(Customer customer);

9.     // async operation w/callback
10.    Future<?>
11.      getQuoteAsync(Customer customer,
12.                AsyncHandler<Score> handler);
13.}
```

Example: Callback Client

```
1. CreditRatingService svc = ...;

2. Future<?> invocation =
3.    svc.getCreditScoreAsync(customerFred,
4.       new AsyncHandler<Score>() {
5.         public void handleResponse
6.              (Response<Score> response) {
7.            Score score = response.get();
8.            // do work here...
9.         }
10.      });

11.// to cancel the request, use
12.// invocation.cancel(true);
```

GUI and Deployment Improvements

- ***No more grey rects! Per window double-buffering.***
- ***Improved native look and feel, including Vista support.***
- ***Layout enhancements, JTable sorting/filtering.***
- ***SwingWorker updates, integration.***
- ***AWT desktop integration: splashscreens, toolbars, tray icons, app launching.***
- ***LCD text antialiasing, curved primitives, single-threaded rendering.***

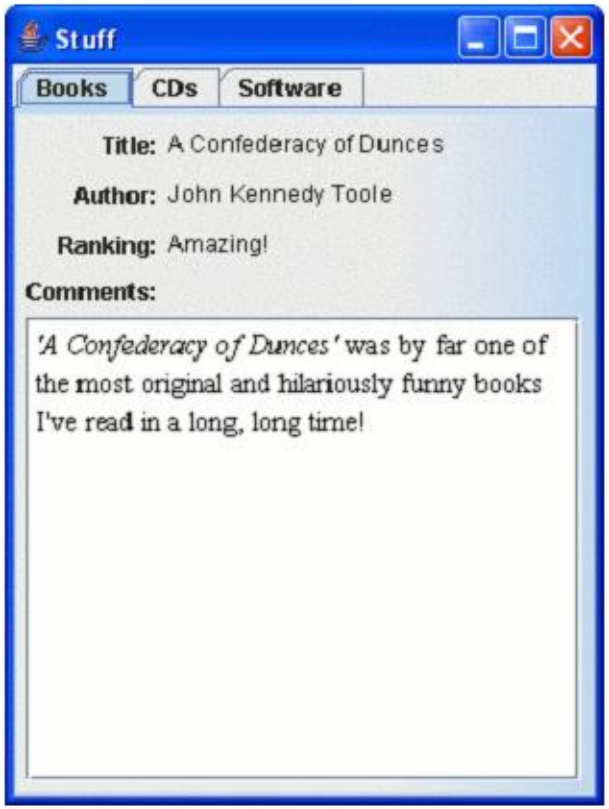


- ***Easier JRE deployment, JNLP launching from browsers.***
- ***Improved user experience and unified download engine for plugin and webstart.***

http://weblogs.java.net/blog/chet/archive/2006/02/these_are_some.h

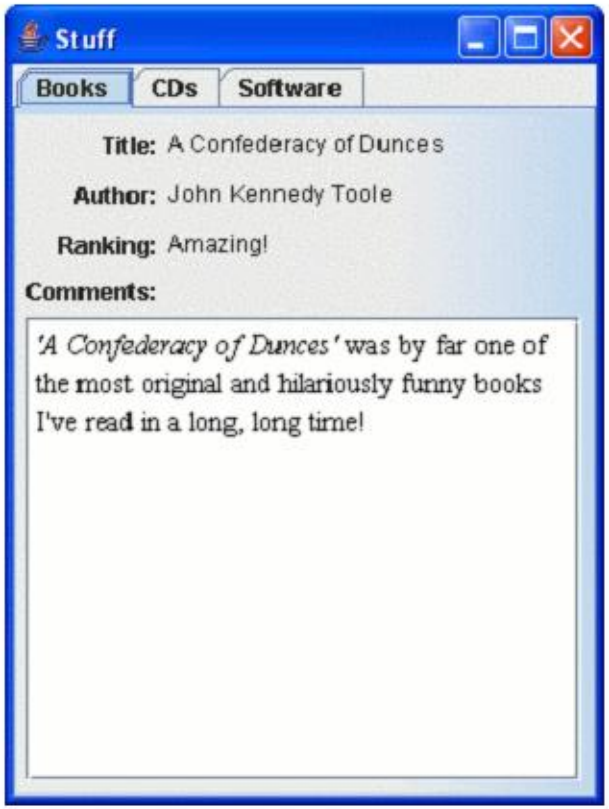
True Double Buffering

Old Behavior

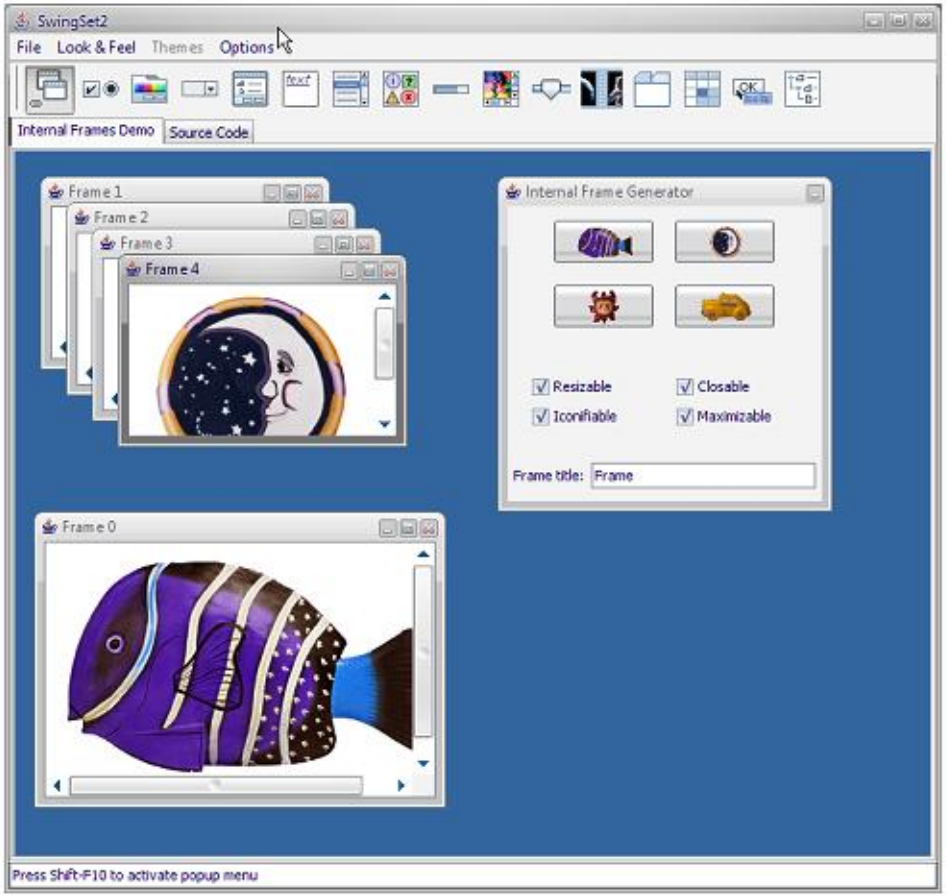


True Double Buffering

New Behavior



SwingSet on Vista with Mustang

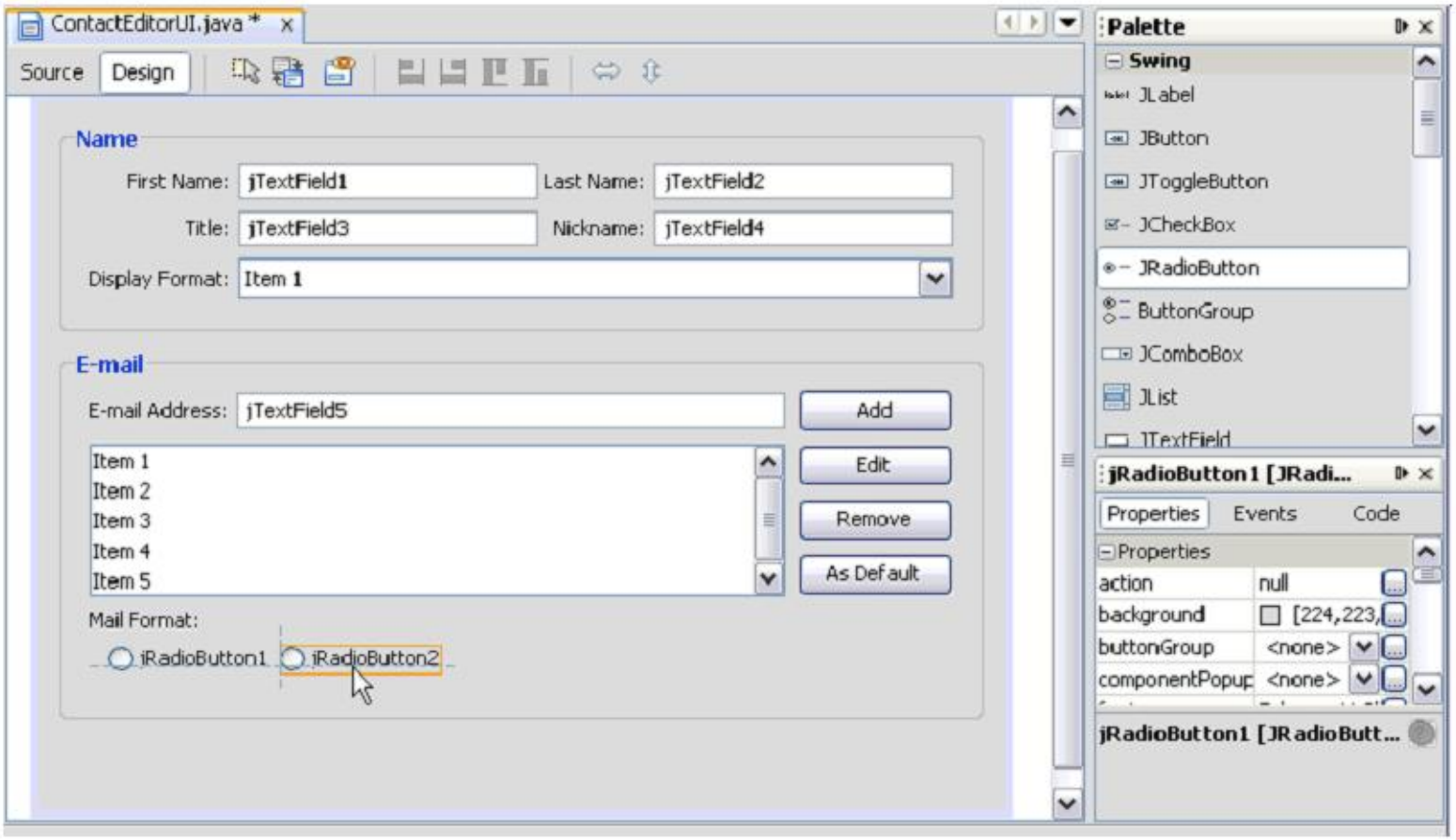


http://weblogs.java.net/blog/chet/archive/2006/10/java_on_vista_y.html

Matisse

- Makes layout easy as pie!
  - No need to learn about LayoutManagers!
- Collaborative effort between Swing and NetBeans
- Debuts in NetBeans 5.0
- Works with 1.4.2 or later
- Requires open source run time
  - Portions of this in 1.6, trying to get remainder in now!

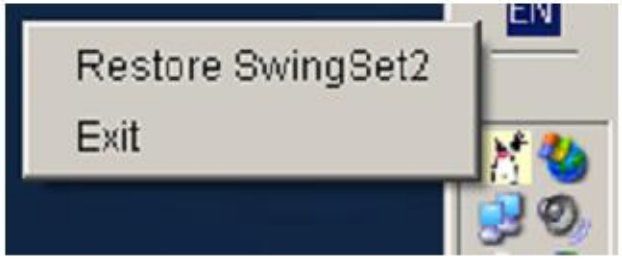
Matisse in Action



Tray Icon

ToolTip and Image

Popup Menu



Message

Listener for various events

Sub-Pixel Text

- ***Plain***

- ***Antialiased***

- ***Sub-pixel***

Scripting for the Java Platform

- "Java" technology is both the language and the platform
  - The platform includes Java VM and JDK™ APIs
  - Language choice—dynamically typed, scripting languages as well
- JSR 223—Scripting for the Java Platform
- Pluggable framework for third-party script engines
- javax.script package
- Optional javax.script.http package ("web scripting")

Scripting in JDK 6

- javax.script included
- javax.script.http not included
- JavaScript technology reference engine
- Based on Mozilla Rhino Engine
- Few optional components of Rhino not included
- Command-line language-independent script shell **jrunscript** (JDK only)

Scripting in JDK 6 - Rhino

- Using Rhino version 1.6R2
- javax.script.Compilable implemented by storing JavaScript interpreter bytecodes
- **JavaAdapter that supports implementation of a single interface**
- Renamed Rhino classes—sun.org.mozilla*
- Can drop later version of Rhino—but can't use that through javax.script API

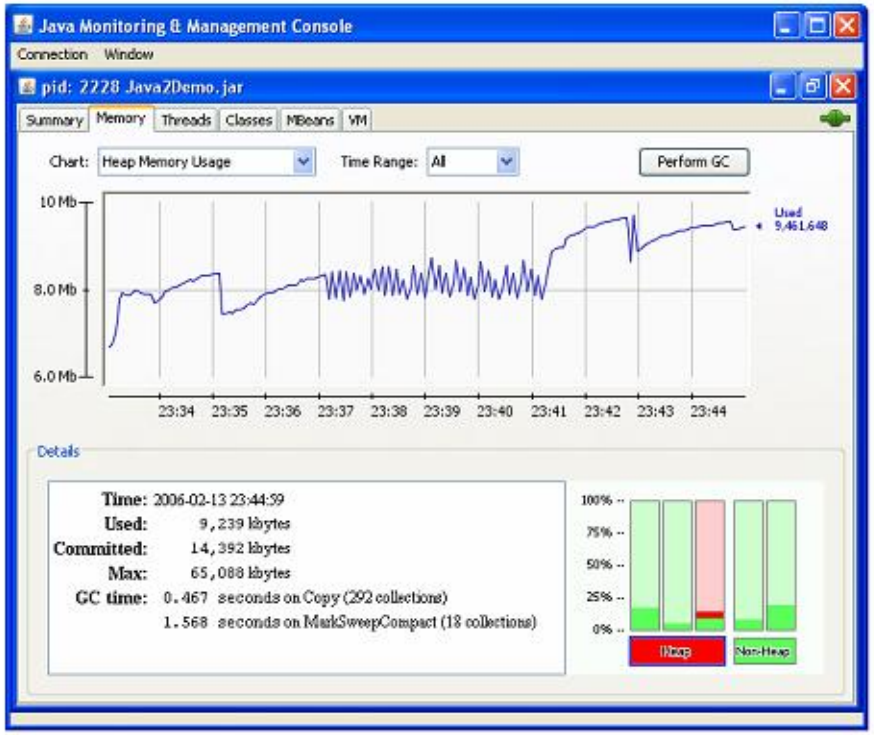
Java Desktop integration

- **java.awt.Desktop API**
  - **Launching the default browser**
  - **Launching the default email client**
  - **Launching the default application to handle file open, file edit and file print**

```
if (!Desktop.isDesktopSupported()) { //exit}

Desktop desktop = Desktop.getDesktop();

if (!desktop.isSupported(Desktop.PRINT)
    { //print }
Desktop.getDesktop().browse(new
URI(urlField.getText()));
```

Monitoring, Management, Diagnostics

- GUI tools: JConsole, jhat, VisualGC (NetBeans), dynamic attach
- Command line tools: jps, jstat, jstack, jmap, jinfo
- Diagnostics: CTRL-Break handler, heap dump, better OutOfMemoryError and fatal error handling, JNI crashes
- Tracing/logging: VM tracing and HotSpot probes, DTrace integration



http://blogs.sun.com/roller/page/dannycoward/2006031

Apache Derby in JDK 6

http://db.apache.org/derby

- Derby: A small-footprint (2MB) standards-based embedded relational database engine
- Will be co-bundled in Sun's Mustang JDK™ software
  - Will not be in the JRE (it's too big)
  - Is ***not*** a required part of the Java SE 6 specification
- Great out-of-the-box starter database for developers
  - Will implement all the new JDBC 4.0 software features
- Developers can redistribute Derby with their applications

Java DB

- Based on Apache Derby, full-featured 100% Java database cross-platform testing and deployment.
  - No need to separately acquire, download, and install a database.
  - Standards-based: JDBC, J2EE, and SQL.
  - Stored procedures, triggers, user-defined data types.
  - Secure: encryptable and password protected.
- Ideal for embedded or client-server use.
  - Small -- 2MB.
  - Zero administration.
  - Embed in browser for better Web 2.0.
    - Easy application + DB distribution.
    - True persistence – no lost or inaccessible data.

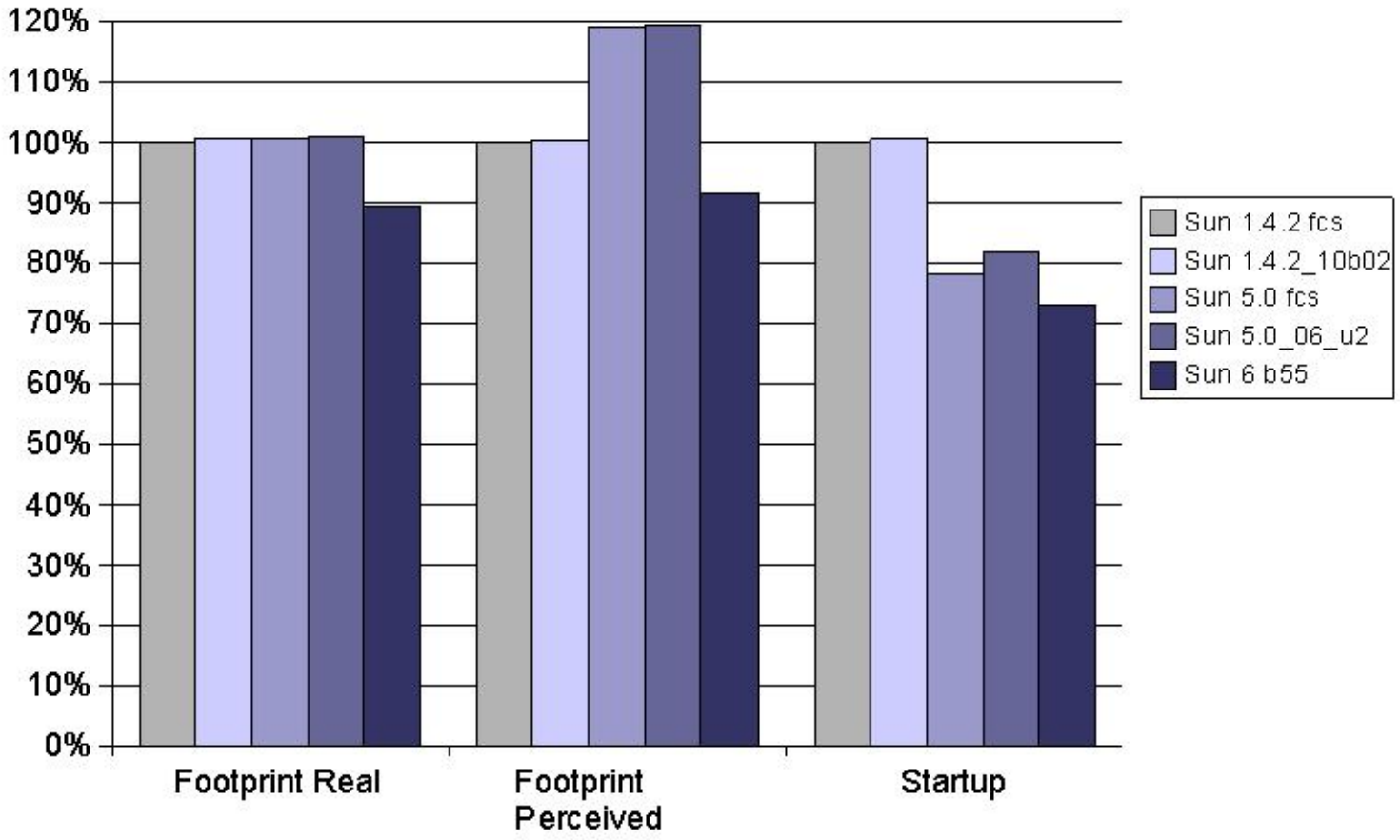
Performance and Compatibility

JSR 202

- Performance portal: http://java.sun.com/performance
- New Type-checking Verifier
  - Uses compiler generated assertions rather than generating and checking assertions at run time.
  - Adopted from J2ME™ technology.
  - Simpler, faster.
- Focus on version to version compatibility – as always.
  - Regressions Challenge engaged community in testing.
  - Extensive, multi-phase Beta in process.
  - Compatibility documentation: http://java.sun.com/javase/6/webnotes/compatibility.html

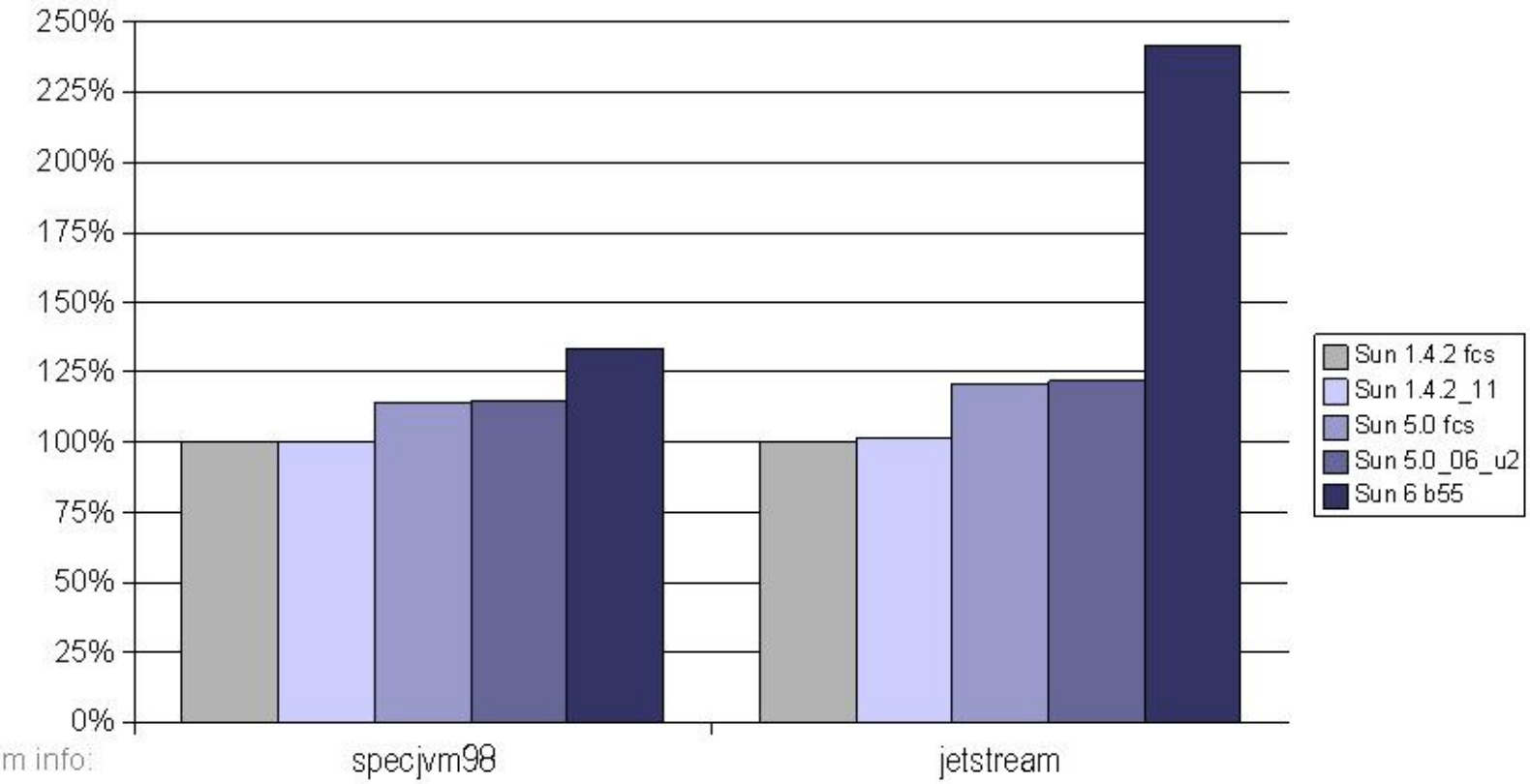
Desktop Performance: Startup and Footprint

Smaller is better



Desktop Performance: Runtime

Bigger is better



System info:

All JVMs are 32-bit

Source: Java Performance Engineering

Resources and Summary

Call for Action!

Donwload and install JDK 5 or JDK 6
Start using the new JDK features ASAP

And Open source Java!!!!

Stay in Touch With the Java SE Platform

- JDK Software Community
  - planetjdk.org
  - community.java.net/jdk
- JDK 6
  - https://jdk6.dev.java.net/
  - http://jcp.org/en/jsr/detail?id=270
- JDK 7
  - https://jdk7.dev.java.net/
  - blogs.sun.com/dannycoward