Java is a trademark of Sun Microsystems, Inc.

JavaOne℠

Tips and Tricks for Ajax Push and Comet TS-4629

Jeanfrancois Arcand
Sun Microsystems

Ted Goddard, Ph.D.
ICEfaces.org

Ajax Push? Comet?

A revolution in web interactivity

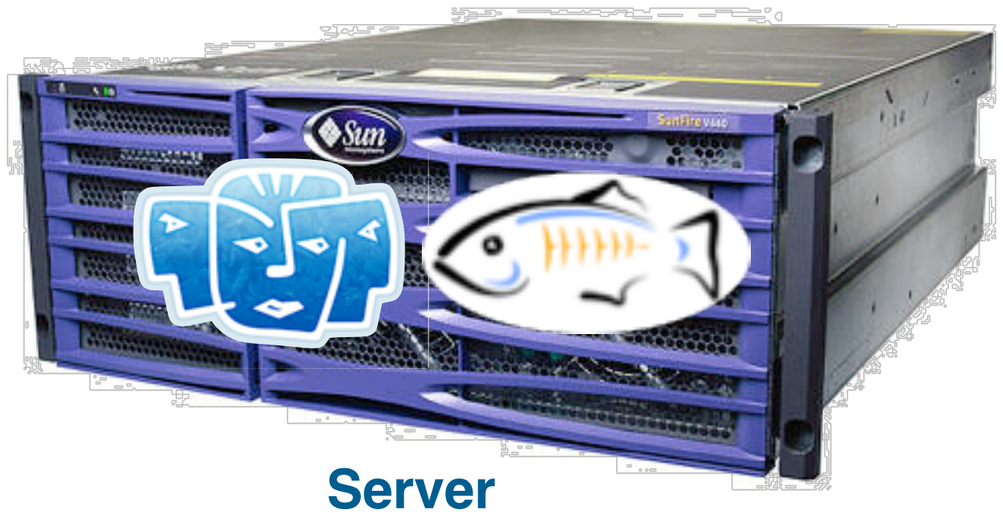
- Ajax hides network operations
  - better user experience via full browser capability
- It's not about writing your application in JavaScript™
- Typical Ajax is not really "asynchronous"
  - client-driven page updates
- Ajax Push/Comet is fully asynchronous
  - server-driven page update

Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Server-mediated Collaboration

The full realization of Ajax.



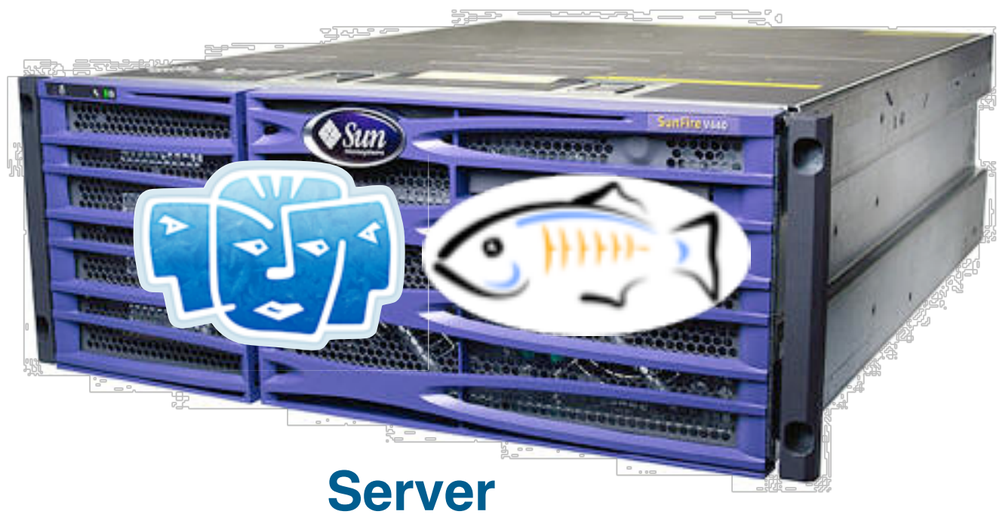
Server

Jeanfrancois

Ted

Server-mediated Collaboration

The full realization of Ajax.



Server

Jeanfrancois

Ted

Server-mediated Collaboration

The full realization of Ajax.

Server

Jeanfrancois

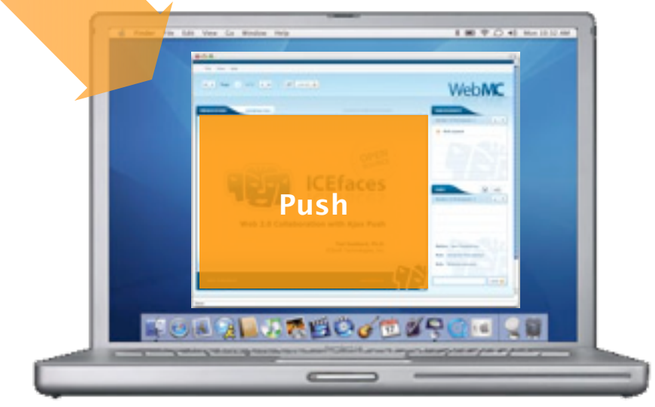
Ted

Server-mediated Collaboration

The full realization of Ajax.

Server

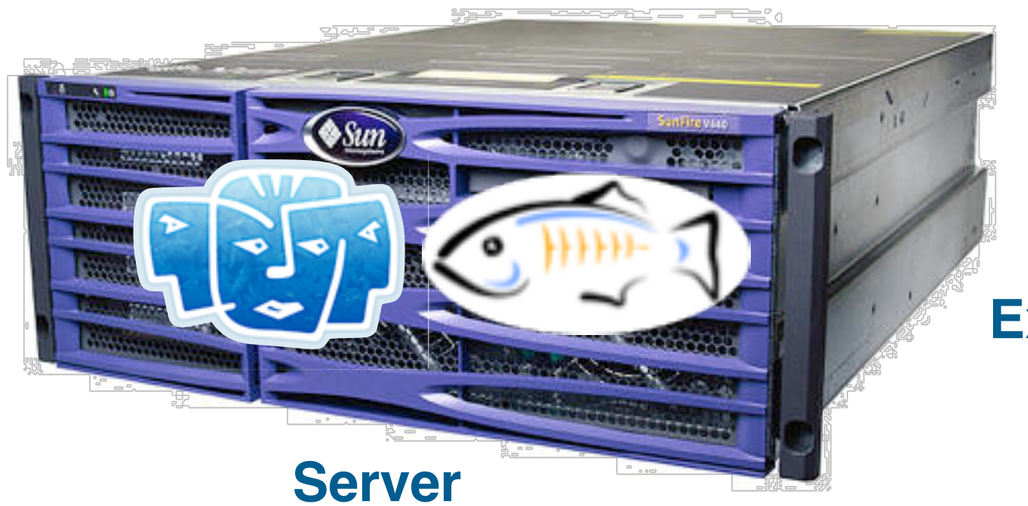
Jeanfrancois



Ted

Server-initiated Updates

Stocks, sensors, and status.



Server

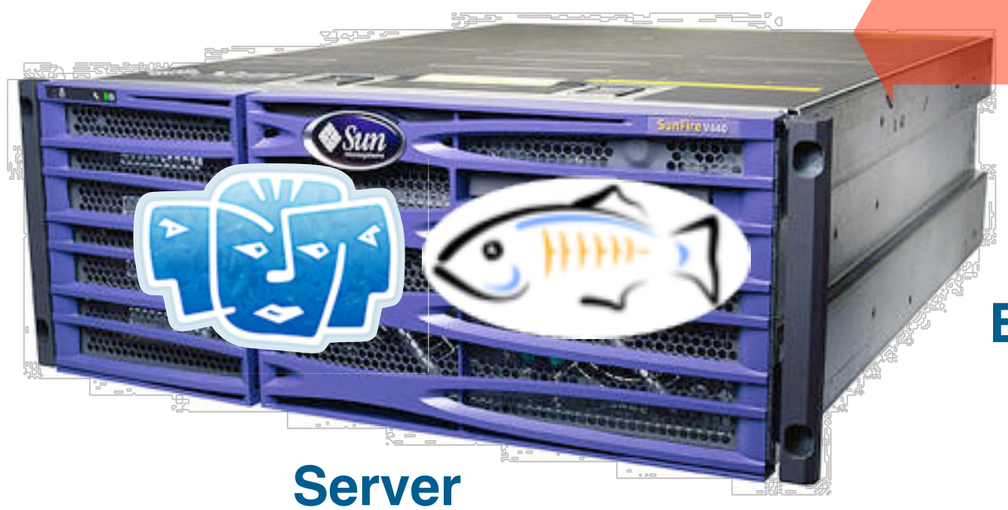
External Application

Jeanfrancois

Ted

Server-initiated Updates

Stocks, sensors, and status.



Server

External Application

Jeanfrancois

Ted

Server-initiated Updates

Stocks, sensors, and status.

Server

External Application

Jeanfrancois

Ted

Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Application Development

Don’t be distracted by push.

- Develop the basic user-driven features first
- Update the model/database with current state
- Determine application events that warrant push
  - Most page updates are still user-driven
- Push current updates to the browser

DWR

Reverse Ajax JavaScript RPC.

```
import org.directwebremoting.proxy.dwr.Util;

scriptSessions =
  webContext.getScriptSessionsByPage(currentPage);
  util = new Util(scriptSessions);
```

To “Reverse Ajax” and invoke arbitrary JavaScript:

```
util.addScript(ScriptBuffer script);
```

Asynchronously and elsewhere in the application ...

```
util.setValue(“form:chat:_id3”, “Howdy”);
```

JavaServer™ Faces for Ajax

A language for Developers and Designers

PageBean.java

```
public class PageBean {
    String message;

    public String getMessage() {
        return message;
    }

    public void
    setMessage(String message) {
        this.message = message;
    }

}
```

Page.xhtml

```
<f:view
    xmlns:f="http://java.sun.com/jsf/core"
    xmlns:h="http://java.sun.com/jsf/html" >

  <html>
    <body>
      <h:form>
        <h:inputText
            value="#{pageBean.message}" />
      </h:form>
    </body>
  </html>

</f:view>
```

Render-based Push

Ajax Push in two lines of code with ICEfaces.

To keep track of groups of users:

```
SessionRenderer.addCurrentSession("javaone");
```

Asynchronously and elsewhere in the application ...

```
user.setSlide(7);
SessionRenderer.render("javaone");
```

The JSF lifecycle runs, updating each user's page from the component tree and current model state.

Servlet 3.0
Standard ARP.

```
AsyncContext asyncContext;

public void service(HttpServletRequest request, ...)
    asyncContext = request.startAsync();
    ...
}
```

Asynchronously and elsewhere in the application ...

```
response = asyncContext.getResponse();
out = response.getWriter();
out.println("Howdy");
context.complete();
```

Atmosphere

Portable Comet annotations on any container.

```
@Suspend
@GET @Produces("text/html")
public String cometGet() {
        return "<!-- Suspended Response -->";
}
```

Asynchronously from a browser POST ...

```
@Broadcast
@Resume
@POST @Consumes("application/x-www-form-urlencoded")
@Produces("text/html")
public String cometPost( @FormParam("name") String name,
        MultivaluedMap form) {
    return “<p>Hello from “ + name + “</p>”;
}
```

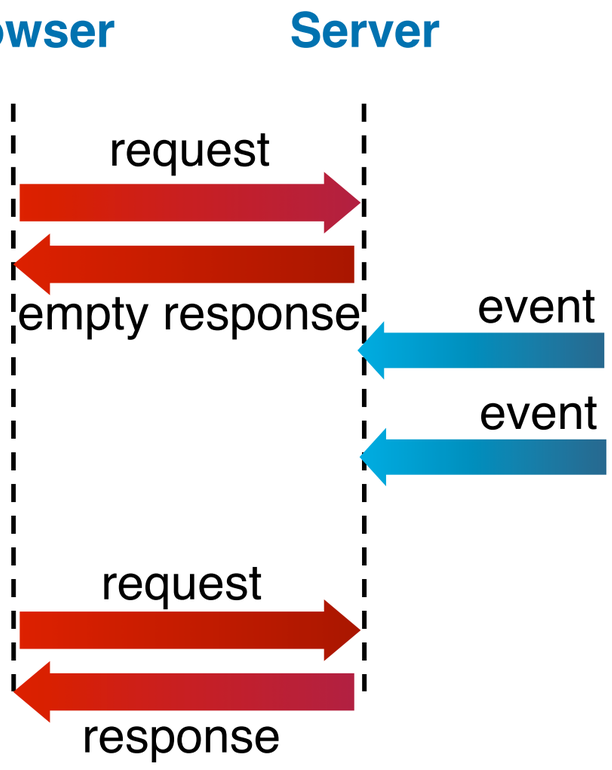
Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

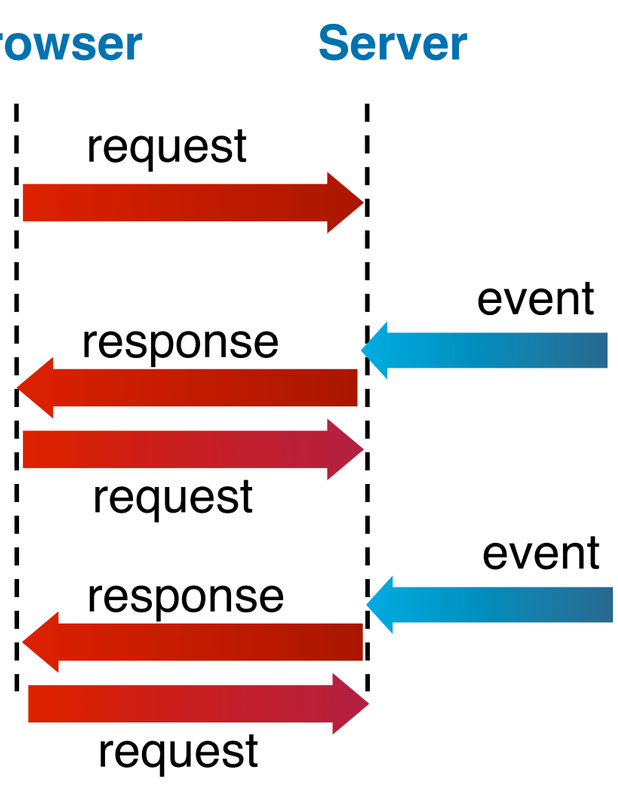
Polling, Streaming, and Ajax Push

Bending the rules of HTTP.

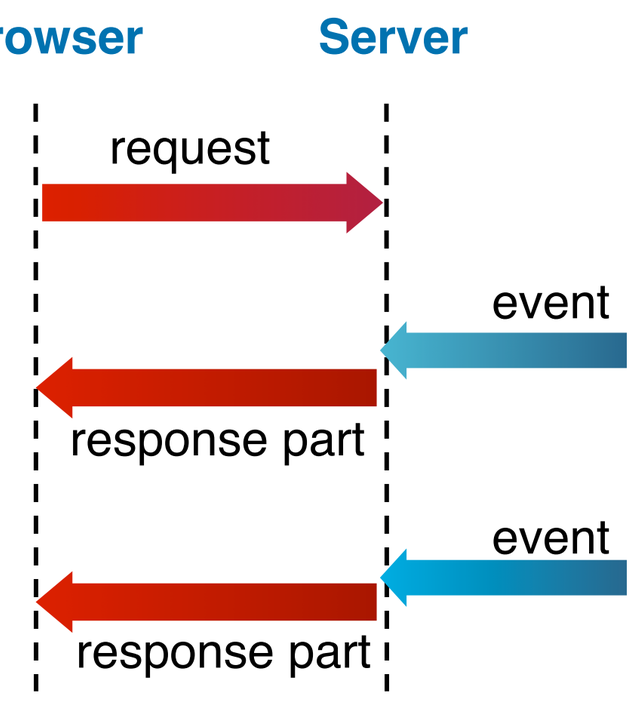
Polling



Ajax Push
(Long Polling)



HTTP Streaming



*Use GET when suspending a connection

HTTP Polling

Repeatedly checking for updates.

- Uses the HTTP protocol in a standard way, but requests are regularly invoked
- Easy to implement, but not efficient or event-based

```
setTimeOut("poll", 10000);
```

HTTP Polling

Repeatedly checking for updates.

```
GET /chatLog HTTP/1.1
Accept: */*
Connection: keep-alive
```

- Uses the HTTP protocol in a standard way, but requests are regularly invoked
- Easy to implement, but not efficient or event-based

```
setTimeOut("poll", 10000);
```

HTTP Polling

Repeatedly checking for updates.

```
GET /chatLog HTTP/1.1
Accept: */*
Connection: keep-alive

<message>One</message>
```

- Uses the HTTP protocol in a standard way, but requests are regularly invoked
- Easy to implement, but not efficient or event-based

```
setTimeOut("poll", 10000);
```

Asynchronous HTTP Streaming

The long response.

```
GET /chatLog HTTP/1.1
Accept: */*
Connection: keep-alive
```

- Parse most recent message in JavaScript
- The original 1999 “Push” technique (Netscape 1.1)

Asynchronous HTTP Streaming

The long response.

```
GET /chatLog HTTP/1.1
Accept: */*
Connection: keep-alive

<message>One</message>
<message>Two</message>
<message>Three</message>
<message>Four</message>
```

- Parse most recent message in JavaScript
- The original 1999 “Push” technique (Netscape 1.1)

Ajax Push (Long Polling)

HTTP message flow inversion.

Ajax Push (Long Polling)

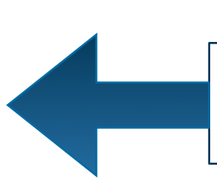
HTTP message flow inversion.

```
GET /auctionMonitor/block/receive-updates?
icefacesID=1209765435 HTTP/1.1
Accept: */*
Cookie: JSESSIONID=75CF2BF3E03F0F9C6D2E8EFE1A6884F4
Connection: keep-alive
Host: vorlon.ice:18080
```

Ajax Push (Long Polling)

HTTP message flow inversion.

```
GET /auctionMonitor/block/receive-updates?
icefacesID=1209765435 HTTP/1.1
Accept: */*
Cookie: JSESSIONID=75CF2BF3E03F0F9C6D2E8EFE1A6884F4
Connection: keep-alive
Host: vorlon.ice:18080
```

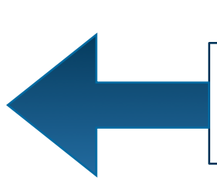


Chat message “Howdy”

Ajax Push (Long Polling)

HTTP message flow inversion.

```
GET /auctionMonitor/block/receive-updates?
icefacesID=1209765435 HTTP/1.1
Accept: */*
Cookie: JSESSIONID=75CF2BF3E03F0F9C6D2E8EFE1A6884F4
Connection: keep-alive
Host: vorlon.ice:18080
```




```
HTTP/1.1 200 OK
Content-Type: text/xml;charset=UTF-8
Content-Length: 180
Date: Thu, 27 Apr 2009 16:45:25 GMT
Server: GlassFish/v3

<updates>
  <update address="_id0:_id5:0:chatText">
    <span id="_id0:_id5:0:chatText">Howdy</span>
  </update>
</updates>
```

Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Why now?

- Before NIO (Non Blocking I/O) in Java™, supporting Ajax Push/Comet required a Thread per suspended response/ connection:
  - Any I/O operations needed to execute and complete using the same Thread
  - For Servlet, it means the complete response must be written before the Servlet.service() terminate. Then the Web Server can commit the response.
  - To support Ajax Push/Comet, need to “block” the Thread inside the Servlet.service() method to avoid committing the response and allow upcoming write.
  - Works fine with low number of concurrent suspended connections (~2000)
  - Quite difficult to make it scale with large number of connections like 20 000 (too many Threads and Thread’s memory required)
- Supported by Jetty Continuations, Atmosphere Framework

Why now?

- NIO allows Web Servers to “detach” the suspended connection from a Thread, hence delaying the “commit” of the response:
  - Threads are required only when servicing requests or when executing a broadcast to a set of suspended connections.
  - Significantly improve scalability of any Ajax Push/ Comet application.
  - Significantly reduce the number of Threads required to execute an Ajax Push/Comet application.
- Grizzly (GlassFish™), Jetty, Tomcat all support Ajax Push/ Comet using NIO/AIO.

Who and when?

- Grizzly (GlassFish), Jetty, Tomcat all support Ajax Push/ Comet using NIO/AIO.
- Servlet 3.0 will introduce Asynchronous operations thanks to NIO.
- JDK 7 NIO.2 (jsr 203) introduces Asynchronous I/O
  - now easier for Web Servers to support asynchronous operations.
- Today: TS-4222, Asynchronous I/O Tricks and Tips, Gateway 102-103

Demo: Atmosphere Framework on Tomcat 5

Blocking Thread per suspended response/connection

- More on Ajax Push/Comet scalability

  BOF-5049: **Scaling the Asynchronous Web**,

  Tonight 8h30pm - 9h20pm

  Esplanade 307-310

Note to reviewer: source code will be demoed using IDE

Demo: Atmosphere Framework on Tomcat 6

NIO to the rescue!

- More on Atmosphere:
  - BOF-5009, **Atmosphere: Comet for everyone, everywhere**
  - Tonight 9h30pm - 10h20pm
  - Esplanade 307-310

Note to reviewer: source code will be demoed using IDE

Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

The Two-Connection Limit

RFC 2616: HTTP 1.1 taken too literally

> Clients that use persistent connections **SHOULD limit** the number of simultaneous connections that they maintain to a given server. A single-user client SHOULD NOT maintain more than **2 connections with any server** or proxy. ... These **guidelines** are intended to improve HTTP response times and avoid congestion.

- Two-connection limit is a guideline
- Is a "client" a browser or a window?
  - windows have isolated JavaScript memory spaces
- "Share" a single connection across windows
  - notify windows of updates via cookie polling

The Two-Connection Limit

When two means six.

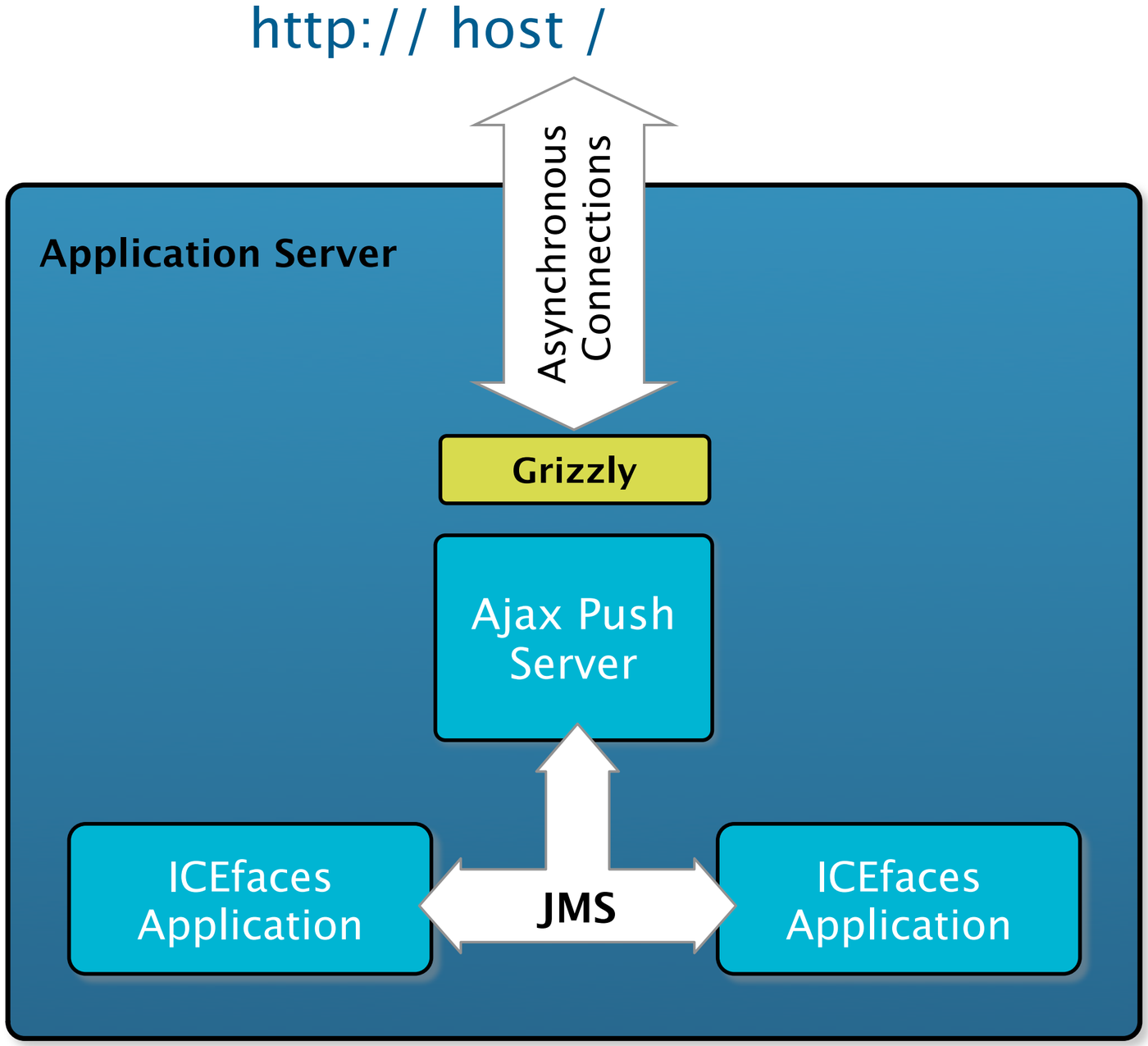
- Like increasing the heap size to “fix” a memory leak

- Safari: two connections per window
- FireFox 3: six connections per browser
- IE 6/7: two connections per browser
- IE 8: six connections per browser

- Should be: two connections per JavaScript sandbox
- HTTP pipelining? HTML5 postMessage()?

Ajax Push Server

One connection is all you need.



Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Long Polling

When standards mean nothing!

- Javascript is standard among browsers...NOT!
- When using long polling, it may works as-it-is...NOT!
  - Firefox always works, e.g on the server side you just suspend the connection.
  - IE never works (as expected ;-)):
    - XMLHttpRequest component doesn't tell you anything about the response until the connection has closed - even if you try polling it instead of relying on onReadyStateChange
    - Suspending the connection *without* sending any data first "breaks" IE.

Long Polling

IE, Chrome and WebKit nightmare

- The trick is to send some comments/"junk" before suspending the connection.
  - Makes IE happy :-)
- Google Chrome "suffers" the same issue.
- WebKit (Safari) as well.
- Use only Firefox...or always send some comments
  - <!-- Ted and Jeanfrancois are great speakers-->
- That one always works ;-)

Demo: Chat on GlassFish

Browser sends GET, GlassFish suspends without writing

Note to reviewer: source code will be demoed using IDE

Demo: Chat on GlassFish

Browser sends GET, GlassFish writes and then suspends

Note to reviewer: source code will be demoed using IDE

Long Polling

My application is losing messages!

- A suspended connection resumes when a server event happens (a broadcast).
  - The server writes the broadcasted message
  - Then resumes the request
  - The browser reconnects, the server suspends the connection.
- What happens if a server event happens before the reconnect? Your application misses the broadcast.
- Solution: use streaming...NOT :-)

Long Polling
Don't panic!

- Possible solution:
  - From the browser, add a special header that can be used for tracking what was the last message received.
  - The server can derive which message to send based on that information.
    - May need to cache/store messages for delivery guarantees.
- Also important when an application is used inside a cluster (sticky session...more to come).

Demo: ICEfaces AuctionMonitor on GlassFish

The header saved my life!

Note to reviewer: source code will be demoed using IDE

Demo: ICEfaces AuctionMonitor on GlassFish

Grrr I've lost the bid!

Note to reviewer: source code will be demoed using IDE

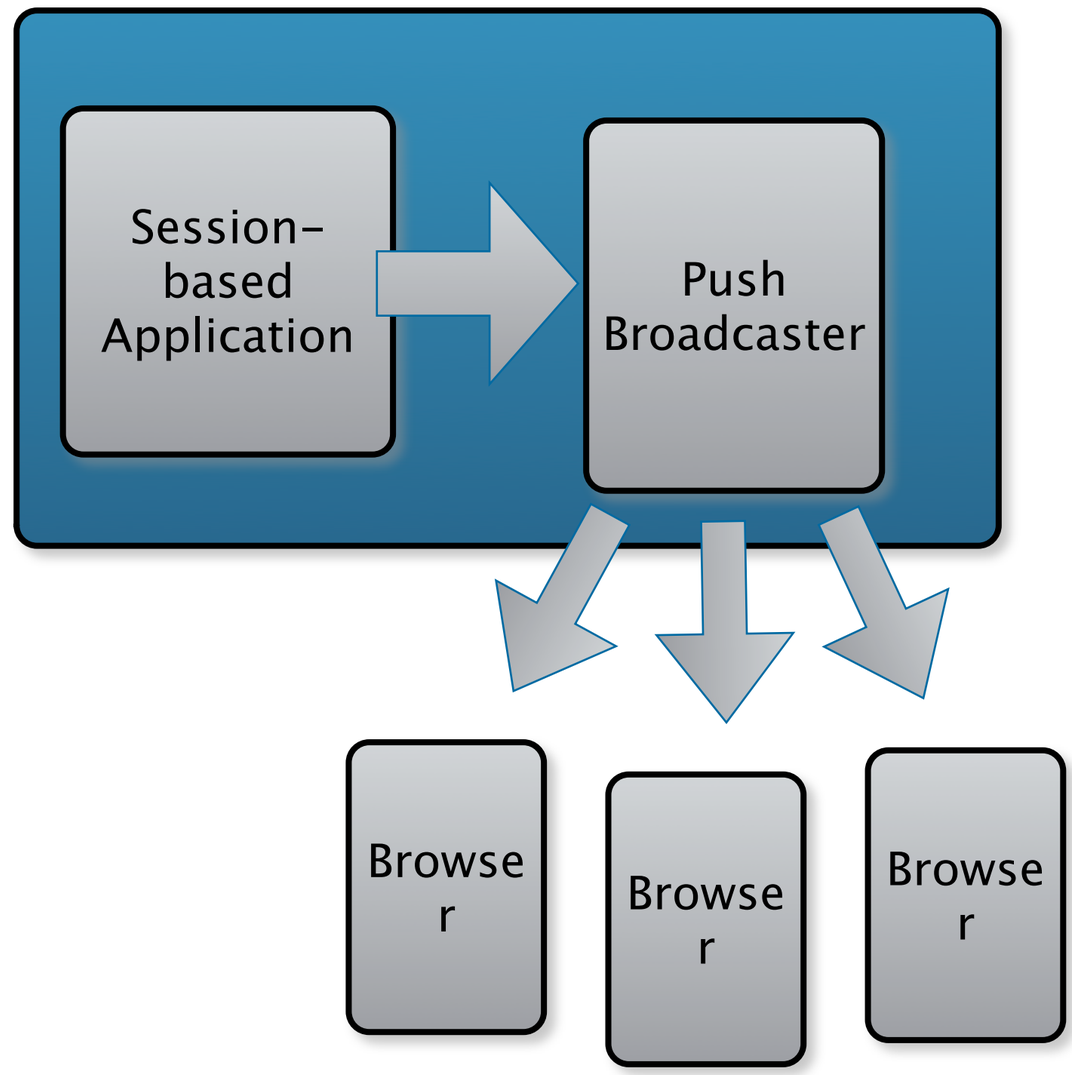
Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Broadcast-based Push

Session state is expensive.

- Replicate push updates to many clients
- Suitable for broadcast applications: news, conferences
- Significantly reduced memory and CPU overhead per user (100x)



Broadcast strategy
New Technology, New Problem

- When the server decides to broadcast an event, which strategy should it use when:
  - Writing a response blocks, e.g the browser is not reading the data fast enough
  - 20 000 responses need to be written
  - 20 000 responses need to be resumed
- The server needs to broadcast a very large number of events every X seconds

Demo: Magnets with Grizzly Comet

Works better when polling!

Note to reviewer: source code will be demoed using IDE

Broadcast strategy

The bottleneck strikes back!

- Be careful when broadcasting events to avoid major pitfalls like:
  - First 100 browsers get a response in less than 0.2s
  - Next 100 browsers get a response 2 seconds later
  - One browser decides to not read the response, effectively blocking all other browsers from receiving server events.

Broadcast strategy
Symptoms!

- The common mistakes:
  - The broadcast is done using the a single thread, e.g. the application or the server uses a single thread to sequentially invoke suspended connections.
  - The broadcast is done using a thread pool, but all threads from the pool are locked because the network is slow or the browsers aren't reading the event. Events get queued waiting for a Thread to become available.

Broadcast strategy
Life after Death

- Use a Thread Pool:
  - Never initiate a broadcast operation using the application thread. Always use a Thread Pool to free the request Thread
  - Make sure the server you are running on is also using a Thread Pool when broadcasting events.
- Timeout idle I/O connections
  - If the browser refuses to read or if the network is slow, avoid “blocking” on the I/O operations
    - Resume the connection automatically
    - Decides what to do, e.g reconnect or not.

Demo: Magnets with Grizzly Comet

It’s getting better.

Note to reviewer: source code will be demoed using IDE

Broadcast strategy

I'm using a Thread Pool, still not working!

- The network may be exhausted
  - The speed gets slower and slower
- Overloading the network will not help.
- The same situation can happen when too many events happen at the same time
  - Like in October 2008 on Wall Street: stock ticker applications suddenly started having "some" update trouble.

Broadcast strategy
Aggregate and Filter

- Aggregate events:
  - Instead of writing events every second, aggregate them and write them in a chunk.
- Filter events:
  - some events can be discarded

Demo: Magnets with Grizzly Comet

Finally it works!

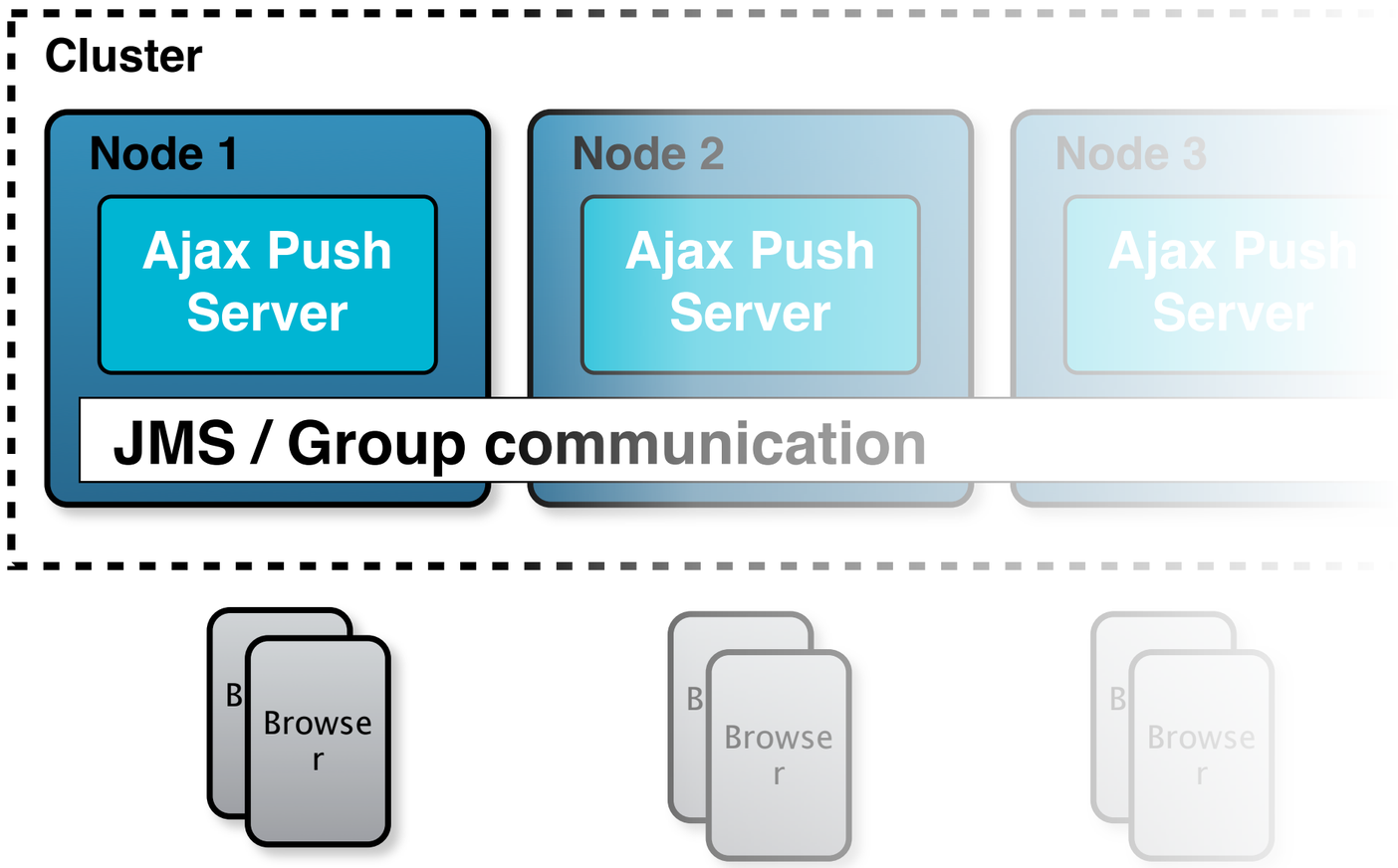
Note to reviewer: source code will be demoed using IDE

Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Horizontal Scalability

Node. Users. Repeat.



- Clients easily served independently in parallel
- “Push” messages broadcast across the cluster

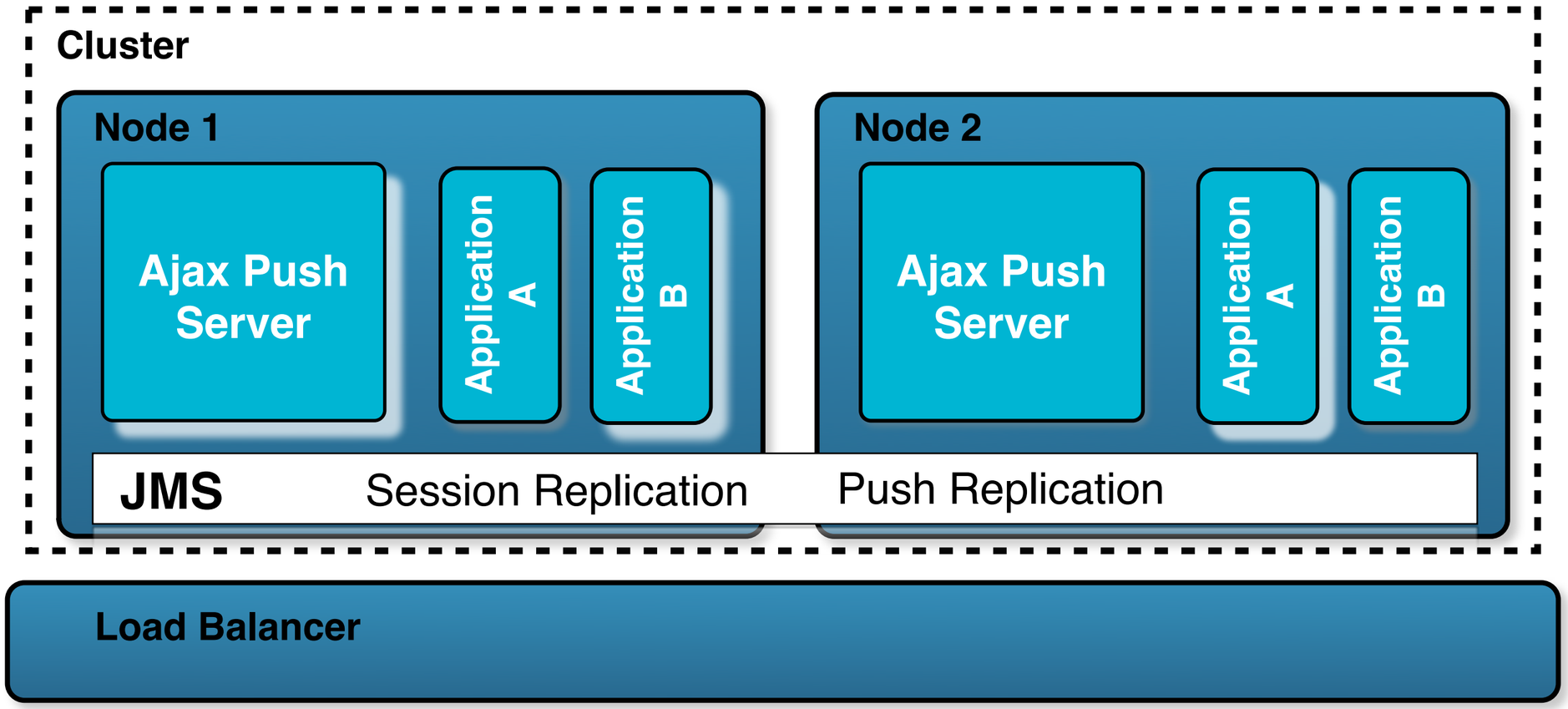
Clustered Ajax Push

My application doesn't get all events!

- Inside a cluster, suspending a connection on a different server will work, but
  - when server 1 decides to broadcast an event, server 2 will never have a chance to receive that event
  - Hence an application will lose events.
- When deploying inside a cluster, communication amongst servers is required
  - Several solutions already exist like JGroups, JMS, etc.

Clustered Ajax Push

Sticky and replicated.



- Sticky session affinity for efficient Ajax
- Session replication for high availability failover

Agenda

- Ajax Push? Comet?
- Application-level APIs
- On the Wire
- 20,000 Connections?
- The Two Connection Limit
- Long Polling Tricks
- Become a Broadcaster
- Join the Cluster
- Conclusion

Conclusion

Join the asynchronous web revolution.

- Ajax Push and Comet is the key to multi-user collaboration on the web
- Polling, Long Polling, and Streaming choices
- Thread scalability requires ARP/NIO
  - (check out Grizzly and Servlet 3.0)
- Connection limits add complexity to implementation
  - (just use ICEfaces, it's open source)

  - Ajax Push is easy with the right framework

More Information

Ping us directly!

- Atmosphere: http://atmosphere.dev.java.net
- Grizzly: http://grizzly.dev.java.net
- ICEfaces: http://www.icefaces.org/
- Ted: http://www.jroller.com/tedgoddard/
- Jeanfrancois: http://weblogs.java.net/blog/jfarcand/

JavaOne℠

Thank You

Jeanfrancois Arcand
jeanfrancois.arcand@sun.com

Ted Goddard, Ph.D.
ted.goddard@icesoft.com