High Performance Web Pages
Real World Examples: Netflix Case Study

Bill Scott
CS193H. Fall 2008.
Stanford University
The Situation

- All attention was focused on server side
  - Most pages 200 - 300 ms
  - Savings of 20 ms celebrated
- Server side is a tiny fraction of the performance pie
  - Typical Netflix page:
Typical Performance

![Graph showing performance over time](image)

- Server Response
- Post HTML to Onload
- Client Render
- Server + Client Render
- Unload to Onload
Same Profile on Member Home

- **Total response/render times:**
  - 75% of customers experience less than 10 seconds (which means 25% are experiencing greater than 10 seconds)
  - 42% less than 5 seconds
  - 29% less than 3 seconds

- **Server response times:**
  - Appear to be only at a maximum 4% of total response/render time
  - 75% of server response times are less than 1.15 seconds

- **Conclusion:** Lots of room for client side improvement
Goal

- Measure request-to-response cycle
  - Gets at what the user actually experiences

- Improve end user performance
  - Implement Steve Souder’s performance rules
The Plan

- Metrics Capture (round trip tracing)
- Apache - gzip, ffe, etag configuration, Proxy cache configuration
- Integrate new Starbar into website
- Better minification for JS & CSS (yui minifier)
- Sprite Bob Graphic Images
- Sprite Header Graphics
- Sprite Queue graphic images
- Configure CDN image assets to use FFE & create image version push system
- Move JS to the bottom, CSS to the top, better JS/CSS packaging
- Switch from graphics based buttons to CSS/Text based buttons
- Switch vignettes from graphics based to CSS/Text based graphic
- Lighter weight BOB, QACL, Menu Nav
- Reduce number of CDN cnames in use
Metrics Capture
Round Trip Tracers

- Client side javascript + server side Java captures time points
- Allows us to measure roundtrip time from request to render
Measurement Points

Prior Page

Request

Response

Page

Web App

js:unload()

A

B

C

D

E

F

G

js:load()

service()

<HEAD>

</BODY>
Measurement Points

- 8 time stamps
- 4 client, 3 server
Measurement Points

- 8 time stamps
- 4 client, 3 server
• C&D and E&F are don’t happen together

• servlet generation time vs browser render time
Prior Page till Client Rendered

Server to Client Render (G-D) + (C-B)

Server Response

Client Render

After HTML body

js:unload()

service() after customer acquired

emits <HEAD>

emits </BODY>

js:onload()

<HEAD>

getTime()

getTime()

</BODY>

Server Response

Prior Page

Request

Response

Page

Web App

Logging Database

browser render time

css loading, asset loading, javascript loading
• 5 metric values; 2 on client, 2 on server, 1 timed on both (G-D) + (C-B)... there is a gap however
elapsed_server_response = 691

(c-b)+gap = 658

elapsed_client_render = 2338

elapsed_server_plus_client (G-D) + (C-B) = 2996 + D-C gap?

browser render time

Prior Page

Request

Response

Page

Web App

js:unload()
• Navigated directly from another site (no G-A) available
elapsed_server_response = 684
(c-b)+gap = 400
elapsed_client_render = 3091
elapsed_server_plus_client (G-D) + (C-B) = 3491 + D-C gap?
(c-b)+gap = 400
elapsed_client_render_post_server = 533
elapsed_client_render = 3091
browser render time
css loading, asset loading, javascript loading
js:unload()
js:load()
prior
page
js:unload()

request

response

page

elapsed_server_response = 684

(c-b)+gap = 400

elapsed_client_render = 3091

elapsed_server_plus_client (G-D) + (C-B) = 3491 + D-C gap?

elapsed_client_render_post_server = 533

browser render time

css loading, asset loading, javascript loading
prior page

response

 elapsed_server_response = 684
 elapsed_server_plus_client (G-D) + (C-B) = 3491 + D-C gap?
 elapsed_client_render = 3091
 elapsed_client_render_post_server = 533
 (c-b)+gap = 400

Web App

browser render time

http request/response for HTML page

http request/responses for css, js, images, etc.
• Internal nav (G-A)
• Network packets happen in parallel
What to Capture

- Time-stamp
- Customer ID
- Page (logical name)
- URL
- Referrer
- Full round trip request time
- Server Response time
- Client render time
- Time from HTML processing to Onload
- Server Response + Client Render
What to Capture

- Details
- IP address
- User agent
- Connection type
- Server name
- Browser
- OS
- Bandwidth test
Logged Metrics

- Logged to database
- Self-Service Portal for plotting metrics
Firebug Extension

- Firebug extension that shows real-time performance measures for a given page
## Jiffy Firebug Extension

<table>
<thead>
<tr>
<th>Event</th>
<th>Duration</th>
</tr>
</thead>
<tbody>
<tr>
<td>PageStart</td>
<td>587 ms/587 ms</td>
</tr>
<tr>
<td>Load</td>
<td>587 ms</td>
</tr>
<tr>
<td>Fetch</td>
<td>1330 ms/3658 ms</td>
</tr>
<tr>
<td>AjaxRequestMade</td>
<td>4 ms</td>
</tr>
<tr>
<td>ResponseReceived</td>
<td>548 ms</td>
</tr>
<tr>
<td>ResponseHandled</td>
<td>14 ms</td>
</tr>
<tr>
<td>AjaxRequestMade</td>
<td>6 ms</td>
</tr>
<tr>
<td>ResponseReceived</td>
<td>745 ms</td>
</tr>
<tr>
<td>ResponseHandled</td>
<td>13 ms</td>
</tr>
<tr>
<td>Load Initial</td>
<td>5 ms/5 ms</td>
</tr>
<tr>
<td>EndLoad</td>
<td>5 ms</td>
</tr>
<tr>
<td>onLoad</td>
<td>11 ms/11 ms</td>
</tr>
<tr>
<td>carousel created</td>
<td>11 ms</td>
</tr>
<tr>
<td>ScrollNext</td>
<td>24 ms/734 ms</td>
</tr>
<tr>
<td>afterCalculate</td>
<td>0 ms</td>
</tr>
<tr>
<td>afterScrollTo</td>
<td>0 ms</td>
</tr>
<tr>
<td>scrollHandling</td>
<td>0 ms</td>
</tr>
<tr>
<td>loadHandling</td>
<td>10 ms</td>
</tr>
</tbody>
</table>
Performance Improvements
Performance Steps

- **GZIP HTML, Javascript and CSS**
  - (exception old netscape browsers and IE6 gets only compressed HTML)

- **Far future expires header for Javascript & CSS**

- **Turn off etags**
  - These often force unnecessary requests, we don't use them so turning them off can help performance.

- **Proxy Cache Configuration**
  - For browsers behind a proxy cache we tell the proxy server that we are handling this. This prevents some errors that might occur when a proxy cache server tries to serve up cached content for a browser that doesn't expect it (already served the URL to one that did).
GZIP
### Your Queue

#### DVD (266)

<table>
<thead>
<tr>
<th>List Order</th>
<th>Movie Title</th>
<th>Instant</th>
<th>Star Rating</th>
<th>Shipped</th>
<th>Est. Arrival</th>
<th>Report Problem</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>August Rush</td>
<td></td>
<td>★★★★☆</td>
<td>08/25/08</td>
<td>08/26/08</td>
<td>Report Problem</td>
</tr>
<tr>
<td>+</td>
<td>The Kite Runner</td>
<td></td>
<td>★★★★☆</td>
<td>05/27/08</td>
<td>05/28/08</td>
<td>Report Problem</td>
</tr>
</tbody>
</table>

#### Instant (287)

#### DVD Queue (257)

<table>
<thead>
<tr>
<th>List Order</th>
<th>Movie Title</th>
<th>Instant</th>
<th>Star Rating</th>
<th>Genre</th>
<th>Expected Availability</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Battlestar Galactica: The Miniseries</td>
<td></td>
<td>★★★★★</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Battlestar Galactica: Season 1: Disc 2</td>
<td>TOP</td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Battlestar Galactica: Season 1: Disc 3</td>
<td>TOP</td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Battlestar Galactica: Season 1: Disc 4</td>
<td>TOP</td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Battlestar Galactica: Season 1: Disc 5</td>
<td>TOP</td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Saints and Soldiers</td>
<td></td>
<td>★★★★☆</td>
<td>Drama</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MST3K: Mixed-Up Zombies</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Long Wait</td>
<td></td>
</tr>
<tr>
<td>Movie Title</td>
<td>Instant</td>
<td>Star Rating</td>
<td>Genre</td>
<td>Availability</td>
<td>Remove</td>
<td></td>
</tr>
<tr>
<td>-------------------------------------------------</td>
<td>---------</td>
<td>-------------</td>
<td>----------------</td>
<td>--------------</td>
<td>--------</td>
<td></td>
</tr>
<tr>
<td>Get Smart</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Comedy</td>
<td>Nov 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Chronicles of Narnia: Prince Caspian</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Children &amp; Family</td>
<td>Dec 2008</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Foreign</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aladdin: Platinum Edition</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Children &amp; Family</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dexter: Season 3</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Television</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Katyn</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Foreign</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Marx Brothers in a Nutshell</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Documentary</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>The Spanish Prisoner</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Drama</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VeggieTales: Lyle the Kindly Viking</td>
<td></td>
<td>✭✭✭✭✭✭</td>
<td>Children &amp; Family</td>
<td>Unknown</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Questions? Visit our FAQ section.
Rewriting Queue

- Queue was completely re-written from scratch
  - Changed from Java generated HTML to JSP generated HTML
  - Migrated to Struts 2 Framework
- Idea was to radically clean up old crusty code
Queue Performance Degraded

- Side effect: Larger payload due to whitespace in JSP
Gzip Components

- Can Gzip scripts, stylesheets, JSON, XML, etc.
- Reduces response size by about 70%
- 90% of all traffic is handled by browsers that support gzip
- For those browsers that don’t support gzip, apache supports Vary response headers automatically
- Requires apache configuration (mod_gzip or mod_deflate)
GZIP: Apache Configuration

<Proxy *>
 SetOutputFilter DEFLATE
</Proxy>

<Location />
 # GZIP COMPRESSION.
 # For all browsers turn on html, css and javascript gzip compression
 # For old browsers turn OFF all gzip compression
 # For IE6 gzip html only

 # Allow gzip compression for html, css, and javascript
 AddOutputFilterByType DEFLATE text/html text/javascript text/css application/x-javascript

 # Netscape 4.x has some problems...
 BrowserMatch `Mozilla/4 gzip-only-text/html

 # Netscape 4.06-4.08 have some more problems
 BrowserMatch `Mozilla/4\.[0678] no-gzip

 # MSIE masquerades as Mozilla, but it is fine
 BrowserMatch \bMSIE\s7 !no-gzip !gzip-only-text/html

 # Turn off gzip for images, pdf, zips and swfs
 SetEnvIfNoCase Request_URI \.(?:gif|jpe?g|png)$ no-gzip dont-vary
 SetEnvIfNoCase Request_URI \.pdf$ no-gzip dont-vary
 SetEnvIfNoCase Request_URI \.zip$ no-gzip dont-vary
 SetEnvIfNoCase Request_URI \.swf$ no-gzip dont-vary
</Location>
FFE, Cache Control, eTags

<LocationMatch "\.(css|js)$">
   # Far Future Expires for Javascript and CSS
   Header set Expires "Thu, 15 Apr 2020 20:00:00 GMT"
</LocationMatch>

<Location /
   Header unset ETag
   FileETag None

   #Header append Vary User-Agent env=!dont-vary
   Header set Cache-Control "private"
</Location>
GZIP, FFE, eTags, Cache: Results

- Queue Payload improvement

- Network outbound traffic cut in half
GZIP, FFE, eTags, Cache: Results

- Queue Payload improvement

<table>
<thead>
<tr>
<th>Empty Cache</th>
<th>Primed Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>804.8K 1HTML/Text</td>
<td>804.8K 1HTML/Text</td>
</tr>
<tr>
<td>284.5K 3JavaScript Files</td>
<td>284.5K 3JavaScript Files</td>
</tr>
<tr>
<td>73.4K 2Stylesheet Files</td>
<td>73.4K 2Stylesheet Files</td>
</tr>
<tr>
<td>8.0K 20CSS Images</td>
<td>0.0K 20CSS Images</td>
</tr>
<tr>
<td>55.0K 60Images</td>
<td>0.0K 60Images</td>
</tr>
<tr>
<td>1225.8K 86HTTP requests</td>
<td>1162.8K 86HTTP requests</td>
</tr>
<tr>
<td>Total size</td>
<td>Total size</td>
</tr>
</tbody>
</table>

- Network outbound traffic cut in half
GZIP, FFE, eTags, Cache: Results

- Queue Payload improvement

<table>
<thead>
<tr>
<th>Empty Cache</th>
<th>Primed Cache</th>
<th>Empty Cache</th>
<th>Primed Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>804.8K 1HTML/Text</td>
<td>804.8K 1HTML/Text</td>
<td>51.6K 1HTML/Text</td>
<td>51.6K 1HTML/Text</td>
</tr>
<tr>
<td>284.5K 3JavaScript Files</td>
<td>284.5K 3JavaScript Files</td>
<td>0.0K 1XMLHttpRequest</td>
<td>0.0K 1XMLHttpRequest</td>
</tr>
<tr>
<td>73.4K 2Stylesheet Files</td>
<td>73.4K 2Stylesheet Files</td>
<td>63.9K 3JavaScript Files</td>
<td>63.9K 3JavaScript Files</td>
</tr>
<tr>
<td>8.0K 20CSS Images</td>
<td>0.0K 20CSS Images</td>
<td>15.4K 2Stylesheet Files</td>
<td>15.4K 2Stylesheet Files</td>
</tr>
<tr>
<td>55.0K 60Images</td>
<td>0.0K 60Images</td>
<td>21.9K 23CSS Images</td>
<td>21.9K 23CSS Images</td>
</tr>
<tr>
<td>1225.8K Total size</td>
<td>1162.8K Total size</td>
<td>56.3K 61Images</td>
<td>56.3K 61Images</td>
</tr>
<tr>
<td>86 HTTP requests</td>
<td>86 HTTP requests</td>
<td>209.3K Total size</td>
<td>91 HTTP requests</td>
</tr>
<tr>
<td>91 HTTP requests</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Network outbound traffic cut in half
GZIP, FFE, eTags, Cache: Results

- Queue Payload improvement

<table>
<thead>
<tr>
<th>Empty Cache</th>
<th>Primed Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>804.8K 1HTML/Text</td>
<td>804.8K 1HTML/Text</td>
</tr>
<tr>
<td>284.5K 3JavaScript Files</td>
<td>284.5K 3JavaScript Files</td>
</tr>
<tr>
<td>73.4K 2Stylesheet Files</td>
<td>73.4K 2Stylesheet Files</td>
</tr>
<tr>
<td>8.0K20CSS Images</td>
<td>0.0K20CSS Images</td>
</tr>
<tr>
<td>55.0K60Images</td>
<td>0.0K60Images</td>
</tr>
<tr>
<td>1225.8KTotal size</td>
<td>1162.8KTotal size</td>
</tr>
<tr>
<td>86HTTP requests</td>
<td>86HTTP requests</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Empty Cache</th>
<th>Primed Cache</th>
</tr>
</thead>
<tbody>
<tr>
<td>51.6K 1HTML/Text</td>
<td>0.0K 1XMLHttpRequest</td>
</tr>
<tr>
<td>0.0K 1XMLHttpRequest</td>
<td>0.0K 1XMLHttpRequest</td>
</tr>
<tr>
<td>63.9K 3JavaScript Files</td>
<td>21.9K23CSS Images</td>
</tr>
<tr>
<td>21.9K23CSS Images</td>
<td>0.0K23CSS Images</td>
</tr>
<tr>
<td>56.3K61Images</td>
<td>0.0K61Images</td>
</tr>
<tr>
<td>209.3KTotal size</td>
<td>91HTTP requests</td>
</tr>
<tr>
<td>86HTTP requests</td>
<td>86HTTP requests</td>
</tr>
</tbody>
</table>

- Network outbound traffic cut in half
GZIP, FFE, eTags, Cache: Results

- Performance improvement: 13-25%
Far Future Expires

- Avoids unnecessary HTTP requests
- Requires version naming of files (which we do for some files)
- Only aids those with an primed cache
- What % come with empty cache?
- At Yahoo! it averaged 40-60%
How Browser Handles Expires

- **With Empty Cache (component not cached)**
  - GET Request for component

- **With Primed Cache (component is cached)**
  - Has Far Futures Expires Header
    - Browser finds in cache
    - Determines not stale
    - Reads from local cache
  - No FFE
    - Does NOT have Far Future Expires Header
      - Browser finds in cache
      - Determines it is “stale” (expired)
      - Makes a Conditional GET
      - If it has not changed
        - Reads from Local Cache
      - If it has changed
        - Performs a GET Request
There was an issue (Safari)

- For a given HTTP request, the server may respond with a HTTP status of 204. This means ‘no content’ changed.
  - Used throughout Netflix site
  - Started in 2000 with first star bars on the web!
- Once we turned on gzip, Apache for some reason attempted to compress NO CONTENT responses
- Some builds of Safari 3 choked on this
Star Bars
Star Bars everywhere

<table>
<thead>
<tr>
<th>List Order</th>
<th>Movie Title</th>
<th>Instant</th>
<th>Star Rating</th>
<th>Genre</th>
<th>Expected Availability</th>
<th>Remove</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Battlestar Galactica: The Miniseries</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Battlestar Galactica: Season 1: Disc 2</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Battlestar Galactica: Season 1: Disc 3</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Battlestar Galactica: Season 1: Disc 4</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Battlestar Galactica: Season 1: Disc 5</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Saints and Soldiers</td>
<td></td>
<td>★★★★☆</td>
<td>Drama</td>
<td>Now</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>MST3K: Mixed-Up Zombies</td>
<td></td>
<td>★★★★☆</td>
<td>Television</td>
<td>Long Wait</td>
<td></td>
</tr>
</tbody>
</table>
elapsed_server_response = 684

(c-b)+gap = 400

elapsed_client_render = 3091

elapsed_client_render_post_server = 533

elapsed_server_plus_client (G-D) + (C-B) = 3491 + D-C gap?

elapsed_server_plus_client (G-D) + (C-B) = 3491 + D-C gap?

browser render time

css loading, asset loading, javascript loading

js:unload()

Prior Page

Request

Response

Page

Web App

Request

Response

Page

Web App

/Queue
What HTTP requests?

<table>
<thead>
<tr>
<th>Where did the time go (180 item Q)</th>
<th>Where did the time go (180 item Q)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images</td>
<td>Images</td>
</tr>
<tr>
<td>Javascript</td>
<td>Star bars</td>
</tr>
<tr>
<td>CSS</td>
<td>Javascript</td>
</tr>
<tr>
<td>HTML</td>
<td>CSS</td>
</tr>
<tr>
<td>Redirect</td>
<td>HTML</td>
</tr>
<tr>
<td>Other</td>
<td>Redirect</td>
</tr>
<tr>
<td></td>
<td>Other</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>Everthing Else</td>
<td>Everthing Else</td>
</tr>
</tbody>
</table>

- Images: 57.1% (3873) vs 31.7% (3873)
- Javascript: 7.3% (493) vs 7.3% (493)
- CSS: 1.3% (91) vs 1.3% (91)
- HTML: 23.9% (1624) vs 23.9% (1624)
- Redirect: 5.2% (356) vs 5.2% (356)
- Other: 5.1% (346) vs 5.1% (346)

- Everthing Else: 74.6% (1723) vs 74.6% (1723)

Diagram: Shows the percentage of time spent on different types of requests.
Image Fetching is Costly

Lots of Time for Star Bars

- 75% Everthing Else
- 25% Star bars

Your Queue

- Images: 33%
- Star bars: 24%
- Javascript: 5%
- CSS: 5%
- HTML: 7%
- Redirect: 5%
- Other: 24%

Where did the time go (180 item Q)
Spriting Star Bars

- Originally 51 separate images; not sprited
- New version single sprite for all star bars
Surprising performance hit

<table>
<thead>
<tr>
<th>Queue</th>
<th>11-Jun Client</th>
<th>11-Jun Server+Client</th>
<th>4-Jun Client</th>
<th>4-Jun Server+Client</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=20</td>
<td>1422</td>
<td>1502</td>
<td>1374</td>
<td>1433</td>
</tr>
<tr>
<td>21 - 100</td>
<td>2511</td>
<td>2611</td>
<td>2156</td>
<td>2234</td>
</tr>
<tr>
<td>101 - 250</td>
<td>4922</td>
<td>5122</td>
<td>4006</td>
<td>4135</td>
</tr>
<tr>
<td>251 - 500</td>
<td>11812</td>
<td>12060</td>
<td>8670</td>
<td>8890</td>
</tr>
<tr>
<td></td>
<td>2688</td>
<td>2787</td>
<td>2359</td>
<td>2458</td>
</tr>
</tbody>
</table>

Bar chart showing:
- Old Stars
- New Stars

Comparison between <=20 and 21 - 100 ranges.
What went wrong?

- Old star bars did inline event attachment
  - onmouseover=""
  - generated with the page
- New starbars attach events on DomReady
  - with hundreds of events to attach this can cause a slowdown
- Solutions (3)
  - generate inline events (yuck)
  - change to container based event model (lots of global work)
  - use a flyweight pattern of a single interactive star bar shared across the page
- Solution: inline events
After inline events: improved

- Most members experienced another 10% improvement
Other Challenges
Large Table

- IE7 & IE6 suck when rendering large tables
- Some solutions
  - Break large table into smaller chunks
    - Rendering is faster since triggering re-render of smaller table is faster than re-rendering large table
  - Used fixed layout for table to prevent re-rendering
  - Use progressive loading
    - Either a manual approach to load additional
    - Or dynamically load in the background
    - Or a combination of both
JS at Bottom & CSS at Top

- Scripts scattered throughout the page
  - Causes browsers to block while script code is executed

- Queue Example
  - To speed up **perceived** page rendering time you can pre-load background images specified in CSS
  - However, this has to be near the top
  - On IE7 & IE6 this caused significant delays (5-10 seconds on large queue)
  - Removing the performance hack decreased page load time!

```html
<script>
if (document.images) {
    img1 = new Image();
    img2 = new Image();
    img1.src = "../path/to/image-01.gif";
    img2.src = "../path/to/image-02.gif";
}
</script>
```
Drag and Drop

- The sheer number of rows (up to 500) can cause a page to croak for adding drag & drop

- **Issues**
  - Exploding number of event handlers (use container events)
  - Dynamic cursors in IE6 (avoid)
  - Class switching (instead use style switching)
  - Extra calculations for drop targets at drag start (e.g., 500 rows)
    - Don’t measure everything. Measure prototypical row. Flag exceptions.
jQuery Optimization

- Need: find drop targets dynamically
  - Normal way was ok
    $$\$(\"td.dtc\ em\", \"#dvd\-queue\$$)
  - But more obtuse way is faster
    $$\$(\"em\", \"#dvd\-queue\$$)
    .filter(function() {
        return this.parentNode.className === 'dtc';
    })$$
Browser Variances
Browser Variance

- Easy to forget that the experience varies greatly per browser
- Browser share
  - IE7 50%
  - IE6 20%
  - FF3 22%
  - Safari 6%
Safari much faster

- Queue median times all sizes
Crazy Mistakes
Oops

- Script running got triggered by the loss of a boolean on the page
- Notice slightly elevated (before moving JS to bottom)
Oops

Simply removed a `<STYLE>` block in the middle of the page
Lessons Learned
Lessons

- Most of the Yahoo! recommendations are a sure bet (far futures, gzip, etag, etc.)
- The easiest win is gzip
- Not all pages are created equal
- A lot of time is spent fetching images
- Nothing beats simple design, separation of concerns, clean architecture
- It’s the little stuff multiplied that you have to watch out for
Lessons

- The surest way to improve performance (and keep improving performance) is to measure, measure, measure
  - Use some way to capture the full user experience time
  - Be able to log & graph trends
  - Median works best
  - Make sure your numbers are statistically significant

- Don’t be surprised by performance degradation when you expected performance improvement
Lessons

- Browsers are not created equal
  - Some browsers will eat your lunch (read IE)

- Tools are essential
  - Firebug, round trip tracing, self-service dashboard, HammerHead, yslow, Toad, Excel, Jiffy Extension, Round Trip Extension, and old-fashioned instrumentation.

- Science is messy
  - E.O. Wilson