Automated, Open-Source Web Performance Analysis

Daniel Hanks

Help me – to help you..

- What questions do you have?
- What are you hoping to get out of this presentation?

Why should you worry about web performance?

Why should you worry about web performance?

- Bing: 2 seconds slower page load = 4.3% drop in revenue / user
- Google: 400ms delay = 0.59% drop in searches / user
- Yahoo: 400ms delay = 5-9% drop in full-page traffic
- Shopzilla: 5 second speed up = 7-12% increase in conversion.
- Edmunds: 9s -> 1.4s on the homepage = 3% increase in ad revenue.

- Sources:
  - http://www.stevesouders.com/blog/2010/05/07/wpo-web-performance-optimization/
  - http://technology.edmunds.com/blog/2010/11/how-edmunds-got-in-the-fast-lane.html

Web Performance - Who to Follow

- Steve Souders (Google)
  - http://www.stevesouders.com, @souders
- Stoyan Stefanov (Yahoo)
  - http://www.phpied.com/, @stoyanstefanov
  - http://www.slideshare.net/stoyan/psychology-of-performance
- Joshua Bixby (StrangeLoop Networks)
  - http://www.webperformancetoday.com/, @joshuabixby
- Mehdi Daoudi (CatchPoint) / @drit
  - http://www.catchpoint.com/ , @mdaoudi
- Patrick Meenan (WebPageTest)
  - http://blog.patrickmeenan.com/, @patmeenan
- Velocity Conference (O'Reilly)
  - http://velocityconf.com, @velocityconf
- Sergey Chernyshev (New York Web Performance Meetup)
  - http://www.sergeychernyshev.com/, @sergeyche
- https://twitter.com/#!/danhanks/webperf

Web Performance – Who to follow

- Keynote Systems
- Compuware Gomez
- CatchPoint
- Neustar WebMetrics
- StrangeLoop Networks
- DynaTrace
- SmarBear (Have a look at LoadUI – load test from the cloud)
- Pingdom
- New Relic
- Akamai / Limelight / CDNs, etc.
- And a buncha others…

Web Performance Toolbox

- Yslow
  - http://yslow.org/
- Google Page Speed
  - https://developers.google.com/speed/pagespeed/
- WebPageTest
  - http://www.webpagetest.org/
- HARviewer
  - http://www.softwareishard.com/blog/har-viewer/
- Firebug Net Panel (Firefox)
- Chrome Developer Tools (Chrome)
- Fiddler (IE, any browser)
- HTTPWatch (non-open/free, proprietary, but very good)

Analyzing web performance – Waterfall chart basics

- webpagetest.org
  - Demo
  - Intro to waterfall graphs
  - Response time components (DNS, Connect, SSL, TTFB, Content Download, etc.)
  - HAR download
  - Page render time
  - Take into account local browser cache (First-time view & repeat view)
- Firebug/Yslow (Firefox)
  - Demo
- Chrome Network panel
  - Demo
- IE – HTTPWatch / Fiddler

Automating Web Performance Analysis

- The Goal:
  - Automated runs of your transaction.
  - Automated capture of the resulting performance data (HAR)
  - Automated processing of the data
  - Analysis / Visualization

- You can do a lot of this with commercial tools for $$$

- But there's some flexibility you can gain through rolling-your-own with Open Source.
  - And it's pretty easy.

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - Record/Capture a web transaction (Selenium)
  - Play it back repeatedly (Selenium)
  - Export the results in HAR format (Firebug / NetExport)
  - Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)
  - Extract summary data and prep for analysis (perl/sh)
  - Analyze / Visualize the data (R)

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - **Record/Capture a web transaction (Selenium)**
  - Play it back repeatedly (Selenium)
  - Export the results in HAR format (Firebug / NetExport)
  - Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)
  - Extract summary data and prep for analysis (perl/sh)
  - Analyze / Visualize the data (R)

Capturing the web transaction - Selenium

- Selenium IDE
  - http://seleniumhq.org/docs/02_selenium_ide.html
  - Demo – recording a transaction
- Modify the resulting script as needed.
  - Sample script:

```
use strict;
use warnings;
use Time::HiRes qw(sleep);
use Test::WWW::Selenium;
use Test::More "no_plan";
use Test::Exception;
my $sel = Test::WWW::Selenium->new( host => "localhost",
                                    port => 4444,
                                    browser => "*chrome", ### May need to adjust
                                    browser_url => "http://conference.utos.org/" );
$sel->open_ok("/");
$sel->click_ok("link=About");
$sel->wait_for_page_to_load_ok("30000");
$sel->click_ok("link=Volunteer");
$sel->wait_for_page_to_load_ok("30000");
$sel->click_ok("link=Contact");
$sel->wait_for_page_to_load_ok("30000");
```

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - Record/Capture a web transaction (Selenium)
  - **Play it back repeatedly (Selenium)**
  - Export the results in HAR format (Firebug / NetExport)
  - Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)
  - Extract summary data and prep for analysis (perl/sh)
  - Analyze / Visualize the data (R)

Replaying the transaction – Selenium Server

- Install Test::WWW::Selenium (and other dependencies)
- Install Firefox
- Create a clean Firefox profile.
  - (Start Firefox with –p to get into the profile manager)
- Install a JVM (gcj-java didn't work for me)
- Download and run Selenium server
  - http://seleniumhq.org/download/
  - Download the jar file, run, referencing the new Firefox profile:

```
/usr/lib/jvm/java-1.6.0-openjdk-1.6.0.0/jre/bin/java \
  -jar selenium-server-standalone-<version>.jar \
  -firefoxProfileTemplate \
   /home/<user>/.mozilla/firefox/fx4sp2le.SeleniumTester"
```

- Run your selenium script referencing the running instance of this server.
- Selenium will open Firefox, with the clean profile, and run the transaction.
- Automate with cron.

Capturing / Replaying the transaction – Other options

- Selenium WebDriver (IE / Firefox / Chrome / Opera / iPhone / Android)
  - "a collection of language specific bindings to drive a browser"
    - May not need Selenium server
  - Browser-specific plugins and libraries
  - Bindings in lots of languages
- HTTPWatch Automation (IE / Firefox)
  - http://apihelp.httpwatch.com/#Automation Overview.html
- Watir (Web Application Testing in Ruby)
  - IE only on windows
  - Can hook into other browsers via WebDriver
- Capybara
  - See slides from yesterday's session
  - Can drive Selenium
  - More for incorporating web testing into a full testing suite.

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - Record/Capture a web transaction (Selenium)
  - Play it back repeatedly (Selenium)
  - **Export the results in HAR format (Firebug / NetExport)**
  - Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)
  - Extract summary data and prep for analysis (perl/sh)
  - Analyze / Visualize the data (R)

Capturing the results – HTTP Archive (HAR)

- What is HAR?
  - A portable, textual representation of web performance data.
    - http://www.softwareishard.com/blog/har-12-spec/
    - JSON
    - Sample
- Have a look at Steve Souders' HTTP Archive
  - http://httparchive.org/
  - Historical web performance data for lots of websites (maybe yours?)
  - Schema and source code available to roll your own instance.
- Lots of tools export / work with HAR
  - (http://www.softwareishard.com/blog/har-adopters/)
  - harviewer demo
    - http://www.softwareishard.com/blog/har-viewer/

Capturing the results – HAR export in Firefox

- In Firefox:
  - Install Firebug (add-ons)
    - Make sure Firebug opens in your profile when you go to the site you're testing.
  - Install NetExport (add-ons)
    - NetExport: Default Log Directory (prefs.js: extensions.firebug.netexport.defaultLogDir)
    - NetExport Options: Auto-export (prefs.js: extensions.firebug.netexport.alwaysEnableAutoExport, true)
    - Will drop a HAR file after each page load
      - As you automate, probably best to create a directory per execution.
  - Gotchas
    - When does NetExport detect page load?
      - May need to add 'dummy' pages at the beginning and end of your transaction to get first/last pages.
    - You can automate the export dir by munging the Firefox prefs.js.
- Demo

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - Record/Capture a web transaction (Selenium)
  - Play it back repeatedly (Selenium)
  - Export the results in HAR format (Firebug / NetExport)
  - **Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)**
  - Extract summary data and prep for analysis (perl/sh)
  - Analyze / Visualize the data (R)

Induce artificial latency with netem

- http://www.linuxfoundation.org/collaborate/workgroups/networking/netem
- Why?
  - Simulate a data-center move. What impact will it have on our site/app if we add *N*ms of latency?
  - Simulate the impact of poor network conditions (What's your site like for folks on dial-up?)
- How to use:

```
# Add 100ms delay to all outbound packets:
# /sbin/tc qdisc add dev eth0 root netem delay 100ms
# Turn it off:
# /sbin/tc qdisc del dev eth0 root
```

- Many more options available (Packet loss, duplication, corruption, re-ordering,
- Linux only (Windows/MAC alternatives?)
- Requires root access (judcious use of sudo to automate).
- Quick demo

Tie it all together

- A simple script to regularly:
  - Set the export dir (perhaps add a date stamp or other identifying factors),
  - invoke netem,
  - Run the selenium test,
  - remove netem latency.

```
#!/bin/bash
### Add netem delay
sudo /sbin/tc qdisc add dev eth0 root netem delay 100ms
### Set the export dir in Firefox prefs
perl –p –i –e “s{regex to match the export dir in prefs.js}{generated
dir name} \
    /home/user/.mozilla/firefox/<profile_id>.name/prefs.js
### Run the selenium test which will drop a HAR file in the process
/path/to/selenium_test.pl
### Remove netem delay
sudo /sbin/tc qdisc del dev eth0 root
```

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - Record/Capture a web transaction (Selenium)
  - Play it back repeatedly (Selenium)
  - Export the results in HAR format (Firebug / NetExport)
  - Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)
  - **Extract summary data and prep for analysis (perl/sh)**
  - Analyze / Visualize the data (R)

Summarize the results

- Perl Archive::HAR module
- gem install har ### For ruby
- http://code.google.com/p/pyhar/
- Or whatever you have that reads JSON.
- A basic script that pulls out timestamps, page titles, and page load times
- Spit out data which is consumable in R
  - Or write an R module that can grok HAR files (use rjson) – would make a good open-source project.
- Sample Perl script (though I had to mod Archive::HAR slightly):

Sample summary script

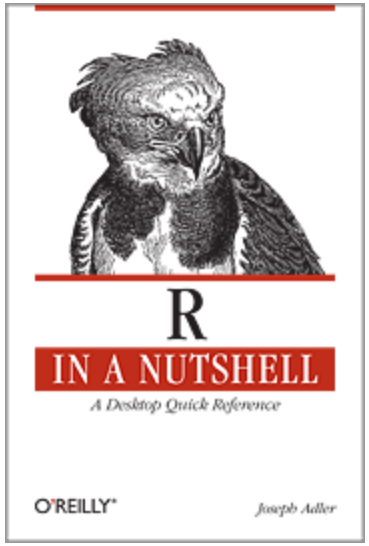
```
#!/usr/bin/perl –w
use strict;
use Archive::Har;
use Perl6::Slurp;

### Header row
print “Page,Timestamp,ResponseTime\n”;

### Print a data row for each harfile we pass in.
for my $file (@ARGV) {
  my $har = Archive::Har->new();
  $har->string(slurp($file));
  for my $page ($har->pages()) {
    print join(‘,’,
      $page->title(),
      $page->startedDateTime(),
      $page->pageTimings->onLoad()
    ) . “\n”;
  }
}
```

The Automated, Open-Source, Web Performance Analysis Stack

- Tasks
  - Record/Capture a web transaction (Selenium)
  - Play it back repeatedly (Selenium)
  - Export the results in HAR format (Firebug / NetExport)
  - Optionally introduce artificial network latency/effects to simulate different network scenarios (netem)
  - Extract summary data and prep for analysis (perl/sh)
  - **Analyze / Visualize the data (R)**

Visualizing the results

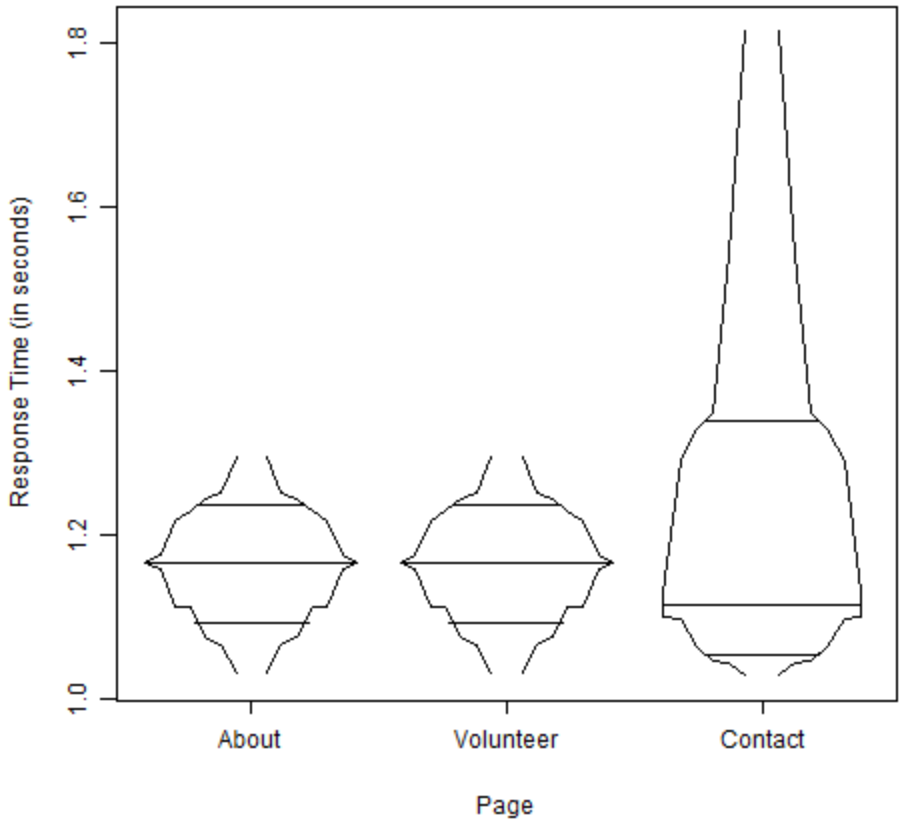
- R
  - http://www.r-project.org/
  - There are a number of free books / tutorials online
- Load the data into a data frame
- Make pretty graphs
- Statistical analysis of performance over time, etc.
- Or use whatever other visualization tool you like
  - (D3, Processing, POV-Ray, Flash, etc.)
- Sample R script:



Sample R analysis / visualization script

```
### Read the data into a data frame (Page, timestamp, response_time)
> page_data <- read.csv(file="har_summary", sep=',',)

### Statisical analysis
> summary(page_data$ResponseTime)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
     -1    1030    1106    1038    1240    1813


### Make a box-percentile plot (requires the Hmisc package)
> png("my_plot.png")
> bpplot (
    (page_data[page_data$Page=='About',]$ResponseTime/1000),
    (page_data[page_data$Page=='Volunteer',]$ResponseTime/1000),
    (page_data[page_data$Page=='Contact',]$ResponseTime/1000),
    name=c('About','Volunteer','Contact'),
    xlab="Page",
    ylab="Response Time (in seconds)"
)
```

Sample R plot



Next steps

- Add some geo diversity (how fast is your site around the world?)
  - Push this stack to a cloud instance (US East, West, EU, APAC, SA) for geo diversity.
  - Store exported data in a central location.
  - Put harviewer or an HTTPArchive instance in front of it for visualization.

Thank you!

- Slides will be available here: http://brainshed.com
- @danhanks
- danhanks@gmail.com


Adobe