CS3226

WEB PROGRAMMING AND APPLICATIONS

LECTURE 10 - PERFORMANCE AND SCALABILITY

DR STEVEN HALIM

STEVENHALIM@GMAIL.COM

Use this to check the performance of your web app

OUTLINE

- Motivation
- Performance
- Scalability

WEBSITE PERFORMANCE

Good website (application) loads faster, ranks higher in (many) search engines, attracts more (new) visitors, and retains more (returning) visitors

Some (extra) reading:

1. Website speed influences search ranking
2. Faster page load time can increase conversions
3. Google top 10 search results, Gmail, Google Maps
4. Search the Internet for recent related articles

WEBSITE SCALABILITY (1)

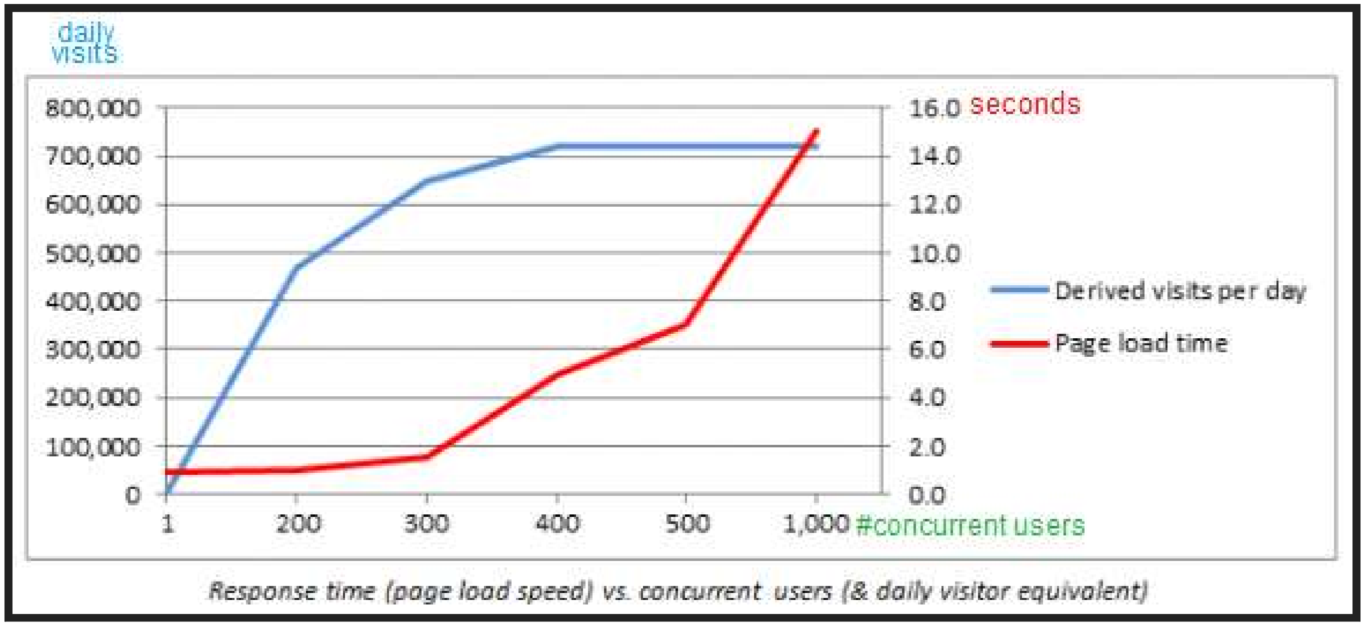
Website (application) *scalability* is related to its *performance*

After we deal with the performance question, hopefully we are then forced to answer the scalability question:

Does it still perform well when there are more (and more) concurrent users?

WEBSITE SCALABILITY (2)

Typically, many web application performance drops with more users as illustrated with plot like this



IMPROVING WEBSITE (APP) PERFORMANCE

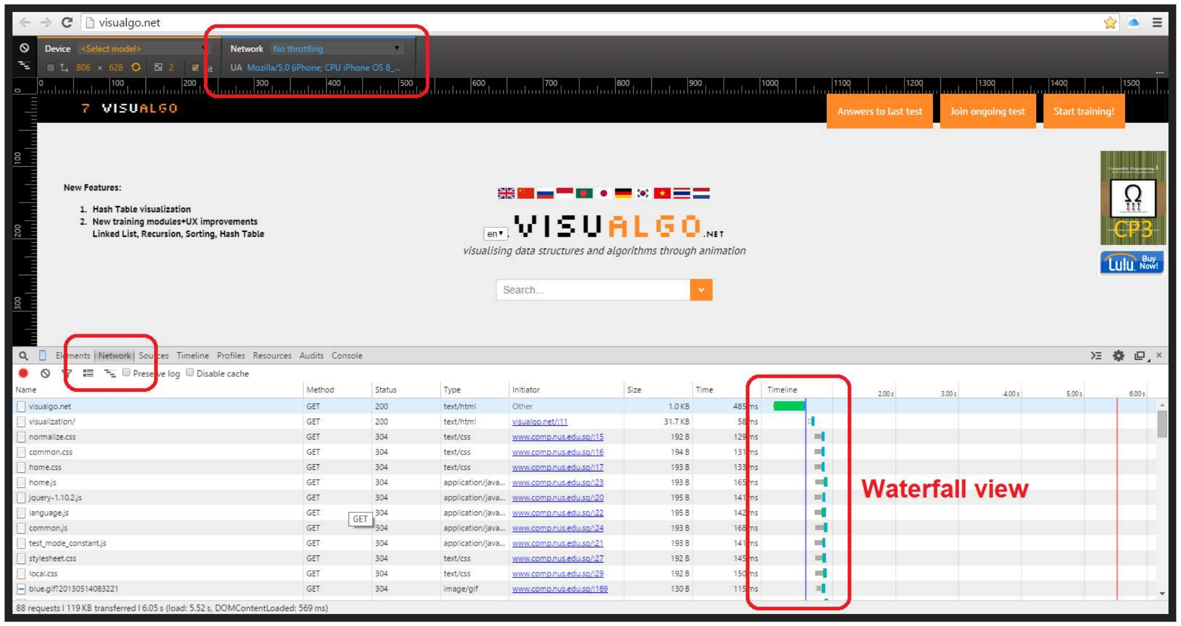
BUILT-IN STOPWATCH (1)

The Network tab of Google Chrome Developer tool is a "built-in stopwatch" to measure website performance

You can view the timeline of events during page load in waterfall view) that can reveal potential areas of improvements

BUILT-IN STOPWATCH (2)

You can also use the "simulated network" drop down list to "simulate" the effect of "slower network" towards your web application performance



Network No throttling

New Features:

1. Hash Table visualization
2. New training modules+UX improvements
   Linked List, Recursion, Sorting, Hash Table

Name	Method	Status	Type	Initiator	Size	Time
visualgo.net	GET	200	text/html	Other	1.0 KB	485 ms
visualization/	GET	200	text/html	visualgo.net/:11	31.7 KB	58 ms
normalize.css	GET	304	text/css	www.comp.nus.edu.sg/:15	192 B	129 ms
common.css	GET	304	text/css	www.comp.nus.edu.sg/:16	194 B	131 ms
home.css	GET	304	text/css	www.comp.nus.edu.sg/:17	193 B	133 ms
home.js	GET	304	application/java...	www.comp.nus.edu.sg/:23	193 B	165 ms
jquery-1.10.2.js	GET	304	application/java...	www.comp.nus.edu.sg/:20	195 B	141 ms
language.js	GET	304	application/java...	www.comp.nus.edu.sg/:22	195 B	142 ms
common.js	GET	304	application/java...	www.comp.nus.edu.sg/:24	193 B	168 ms
test_mode_constant.js	GET	304	application/java...	www.comp.nus.edu.sg/:21	193 B	141 ms
stylesheet.css	GET	304	text/css	www.comp.nus.edu.sg/:27	192 B	145 ms
local.css	GET	304	text/css	www.comp.nus.edu.sg/:29	192 B	150 ms
blue.gif?20130514083221	GET	304	image/gif	www.comp.nus.edu.sg/:189	130 B	115 ms

88 requests | 119 KB transferred | 6.05 s (load: 5.52 s, DOMContentLoaded: 569 ms)

TIPS FOR FASTER WEB APP

In the next few slides, we have a compilation of several known techniques on how to improve the speed of your website (app)

This list is certainly not exhaustive and will continually be improved over time

1. Use less or smaller images

   If you really need to use images, utilize appropriate compression techniques, e.g. JPG (lossy), PNG (lossless), SVG (smaller and scalable, good for simple logo), etc

   Do not test your web application's visitors with gigantic images like this exaggerated example

2. If we need to use lots of small (and usually related) cliparts, consider using image sprites to minimize # of HTTP requests/responses, read this external article for more details

   Note that Bootstrap does something similar with its Glyphicons (fonts, SVG, CSS rules)

   Similarly with jQuery UI, see this

3. Use JS (or sometimes CSS) minification, e.g. jquery.min.js, but pre-optimize using human brain first before doing so

   Exercise: Optimize this before minifying it

   Tool: Google Closure Compiler

   Note: Keep the development copy (without minification) as you may want to update it later in the future

4. Minimize client-server communications and the size of messages transmitted back and forth

   Network speed is much slower than internal CPU+register/cache memory speed

   Use some form of data compression whenever possible when sending data between client and server

   Package the data in complex JSON and submit the data in one go instead of sending multiple HTTP requests

5. Minimize requests to database and/or filesystem

   Accessing a file in a web server further slows down website (app) performance

   Example: A web app needs to dynamically display a table with several columns and the user can choose which column will be the primary sort key

   Instead of using SQL ORDER BY command and thus issuing another HTTP request to the back-end database, we can just load the data once from the web server and then do the dynamic sorting of the unchanged data in *client's* machine using JavaScript

6. Use AJAX technology whenever possible

   So that the user interface does not get frozen when the server is serving client's request, give visual waiting cues (typically an hourglass) when this happens

7. Set up your web server so that static content has long cache expiry date that can be leveraged by the client's web browser (unless they are always in incognito mode)

8. Other techniques to be added in the future :)...

IMPROVING WEBSITE (APP) SCALABILITY

SOL 1: SCALE THE HARDWARE (1)

First steps: Understand what is the issue, find the bottleneck

- If you have access to the web server, benchmark/load test your system first, e.g. with Apache Bench (ab), Apache JMeter, or these four other Linux server monitoring tools
- Monitor the performance of critical components (CPU, DISK, NETWORK, MEMORY) on various usage scenarios

SOL 1: SCALE THE HARDWARE (2)

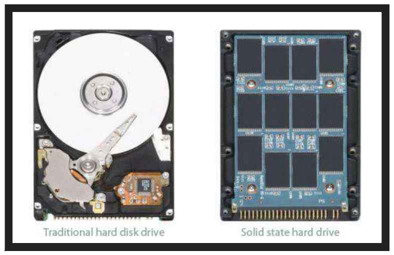
If you have the money, spend it to improve the bottleneck component of your web server

- Get *more* RAM (access time in order of nano seconds)
- Use faster disks (5400RPM vs 10000RPM, lower seek time, more cache)
- Use RAID (Redundant Array of Independent/Inexpensive Disks) for redundancy, durability, and performance

SOL 1: SCALE THE HARDWARE (3)

- Switch traditional HDD to SSD

	HDD	SSD
seek	4-12ms	<0.1ms
latency	2-5ms	0
transfer	125MB/s	> 540MB/s
$	.08/GByte	.60/GByte



Latency is time between request and response
For disk, rotational latency is time between head on the right track and data rotating under read/wite head

SOL 1: SCALE THE HARDWARE (4)

- Utilize *caching* by copying higher use and most recently used data in higher speed (SSD) devices

  Search for data from higher speed to lower speed devices
- Get more network bandwidth
- Get either faster or more (core) CPUs for your web server

SOL 1: SCALE THE HARDWARE (5)

Major issue: This "solution" is mostly NOT in your control, unless you have access to the web server hardware and it can be (very) expensive...

SOL 2: SCALE THE WEB APP (1)

This one may seemingly cost less than the obvious solution 1 earlier, but a good software company will pay good $$$ to have good programmers who can do these:

- Use better algorithms in the scripts, e.g. O(N log N) vs O(N^2) sort, various graph matching algorithms for this semester's Lab4

SOL 2: SCALE THE WEB APP (2)

- Use better data structures in the scripts

  Example: In Lab4, we do not send a full **N*M** Adjacency Matrix of the Bipartite Graph but only the list of weighted edges actually present in that bipartite graph (usually much less but up to **N*M**)

SOL 2: SCALE THE WEB APP (2)

- Consider time/space tradeoffs, so pre-compute and store useful information to speed up response time

  Example: Google does not do heavy page ranking computation when a user enter a new search keyword but just show the results that has been pre-computed earlier

  For our Lab5-7, as we have resorted to fixed `graph_id`, you can choose not to compute solutions of our puzzle in the real time every time such `graph_id` is requested (slow and the answer will always be the same) but compute the answer just once and store the results in database/hardcoded in the PHP script directly

SOL 3: WEB SERVER ALTERNATIVES

We can consider these alternatives

- nginx (Engine X), nginx architecture
- node.js
- Cloud Computing Based solutions
  1. Amazon Web Services
  2. Google App Engine
  3. Digital Ocean
  4. IBM Bluemix

Again, as with solution 1 (scale the hardware), this solution is mostly NOT in your control unless you understand how to setup web server

THE END

Note: Use this to check the performance of your web app