Lancaster University

SCC.306 Internet Applications Engineering
Introduction

Matthew Broadbent and Matthew Bradbury
m.broadbent@lancaster.ac.uk, m.s.bradbury@lancaster.ac.uk

Who are we?

Introductions

Course Aims

- To provide you with an appreciation of the issues facing developers of large scale, high-performance and multi-device web sites
  - From a technical and architectural level
  - But also in terms of accessibility, responsiveness, security, etc.
- To raise awareness of technologies and the approach to engineering software at scale
- To highlight relevant technologies and skills that are useful when preparing to work in Industry

How is the course taught?

- One 2hr Lecture session:
  - Online and live (via Teams)
  - Video material + *interactive elements*
- One 1hr practical session (Weeks 2 to 9):
  - In-person (or online via Teams, if arranged previously)
  - In Science & Technology building (aka Old Engineering)
  - Experimental focus (e.g. measuring, gaining results)
  - Online quiz (weeks 5 & 9)
  - Support & feedback for practical coursework elements

How is the course taught?

- Moodle will be the authoritative source of information for the course
  - *All* material and content will be posted there
- This includes:
    - Links to join the live sessions: lectures/workshops and labs (if applicable)
    - Recordings of all Lecture material
    - Lab workbooks and submission points
    - Quizzes
    - Additional reading
    - Announcements

Provisional Lecture Schedule

Lancaster University

** Dates & topics subject to change*

Week	Topic	Who
1	Web Architecture & Performance	Me!
2	Architecting Online Services	Johnathan Ishmael – BBC
3	Values in Computing	Lucy Hunt – Lancaster University
4	Responsive Web Design	Josh Tumath – BBC
5	Accessibility for Web and Apps	Emma Pratt Richens
6	Web Security	Alex Collins – BT
7	Distributed Identity and Smart Contracts	Matthew Howard – Chia
8	Production-quality Software Testing	James Van Hinsbergh and Fraser Hart - Tesco
9	Web Analytics	Booking.com
10	Embedded Computing & the Internet	Matthew Bradbury

How this Course is Assessed

- 1) Exam **60%**
- 2) Coursework **40%**
  - Details on the next slide…

Coursework Components

Component	Title	Deadline	Weight (%)	Submission Method	Feedback Method	Mark Returned
1	Practical Element 1	Friday, Week 4	40	Moodle	Verbal	By 3/12/21
2	Quiz A	Week 5	10	Moodle	Moodle	
3	Practical Element 2	Friday, Week 8	40	Moodle	Verbal	By 3/1/22
4	Quiz B	Week 9	10	Moodle	Moodle	

Lab Schedule

Lancaster University

Week	Topic	Notes
1	N/A	
2	Practical 1 (Support)	
3	Practical 1 (Support)	
4	Practical 1 (Support)	Submit Friday
5	**Quiz A**	
6	Practical 2 (Support)	
7	Practical 2 (Support)	
8	Practical 2 (Support)	Submit Friday
9	**Quiz B**	
10	1:1 Coursework Feedback	

What is Plagiarism?

- Passing off someone else's work as your own, including:
  - Submitting (e.g.) answers or a report that someone else provided
  - Paying for someone else to do it for you
  - Working on a piece of non-group work together as a group, and submitting it as individual work
  - Sharing of answers/data that you then possibly adapt
- If you **give someone else your work**, you can also be called in for plagiarism
- Coursework is submitted online and checked for plagiarism automatically

What We Expect from You

- Integrity (no plagiarism, no faking results) and effort (active learning):
  - Attend lectures
  - Go to our labs (they're to support you!)
  - Use our/the world's resources effectively
  - Take notes
  - Read around the subject/try things for yourself
  - Ask us questions in lectures and labs
  - Take notes (again, because the slides are not enough when you try to revise, really…!)

What You Can Expect from Us

- We'll do our best
  - To make all our lecture notes available on moodle
  - To personally check the labs are running smoothly and the TAs are offering support
  - To arrange extra support if you've already tried the normal routes (web, forum, TAs)
  - To offer prompt feedback on coursework

Online Expectations

- Online tools will be used to facilitate some aspects of learning e.g. Moodle, Teams, etc.
- However, this is a reminder that the use of these is governed by *existing* policies that you are all currently bound by and have agreed to
- Academic malpractice and plagiarism still applies online
- Direct sharing of code, sharing solutions and/or partial solutions with other students, either privately or in an open chat, is **not acceptable**

Online Expectations

- Don’t forget, these are your fellow students and staff, not some anonymous person on the Internet
- If you’re not sure if you should post or share something, please ask first
- If you see content or a post that you don’t like, in the first instance, message or email the course tutor to alert them to it
- We want these tools to be used; they will give you the best online experience!
  - However, we are asking that you use them sensibly and with respect

How do I get help?

- Please use the labs (online or in-person) to ask for help
- Please ask the TA's in the labs: they are experts when it comes to the coursework!
- Please use the course forum on Moodle (outside of the Labs)

- I will also be posting common Q&As in the General channel of Teams so that everyone can benefit from them

Lancaster University

Part I: Web Architecture

FT.com Engine Room Blog

http://engineroom.ft.com/2016/04/04/a-faster-ft-com/

Lancaster University

- "It's clear from our test that the speed of our website affects both of these revenue streams, over the short term, to the tune of hundreds of thousands of pounds, and in the long-term millions."

Mean % drop in article views between variants and control

Page load time	7 days impact	28 days
1 second slower	-4.9%	-4.6%
2 second slower	-	-5.0%
3 second slower	-7.2%	-7.9%

- Omitted as data did not reach 95% statistical significance

Web Servers
But now they're everywhere

Lancaster University

- The primary function of a web server is to deliver web pages to networked clients
- A web server is just a piece of software that:
  - Takes a request in HTTP format for some named resource over the network
  - Serves that resource back to the client as payload to an HTTP response (HTML/ MIME formats)

- What's changed since this was originally envisioned?

Web and HTTP

Lancaster University

- First, a quick review…
  - web page consists of objects
  - object can be HTML file, JPEG image, video file, audio file,…
  - web page consists of base HTML-file which includes several referenced objects
- each object is addressable by a URL, e.g.,

`www.clevername.com/someDept/pic.gif`

host name | server specific path

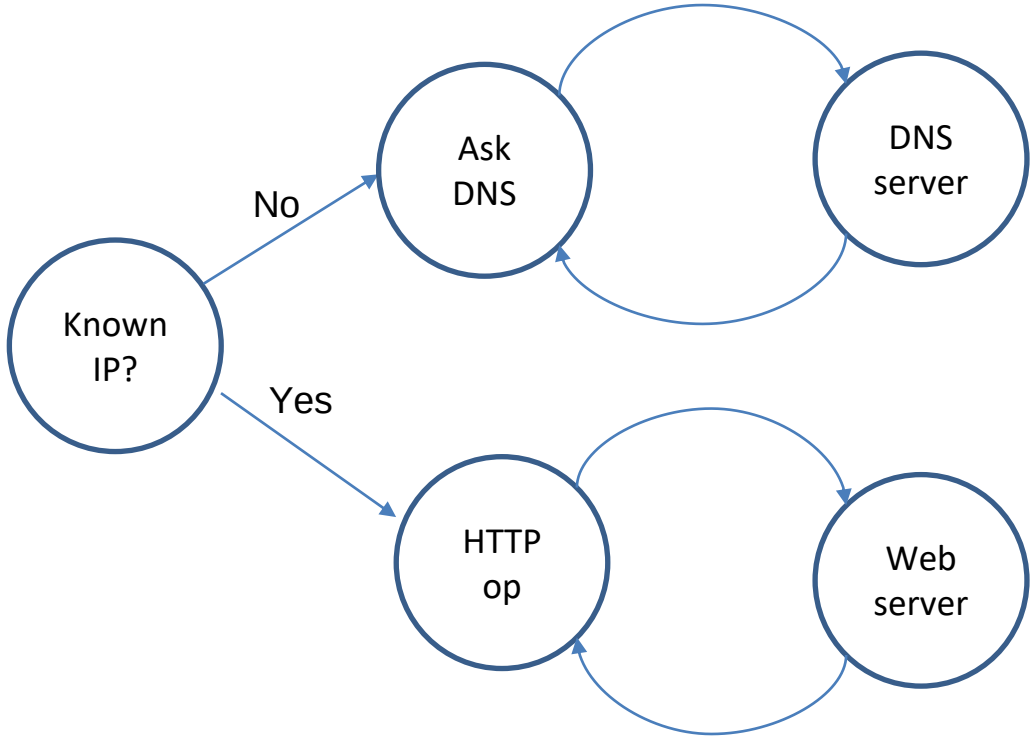
But what about DNS?

Lancaster University

- Before we can request the page we want, we need the IP (Internet Protocol) address of the server
  - bbc.co.uk has address 212.58.246.78
  - lancaster.ac.uk has address 148.88.2.80
  - Your operating system and network routers use this address to decide how to route the data packets making up your request, hop by hop throughout the network
  - If we don't know the IP, we need a way to find it out. The **domain name system (DNS)** provides a distributed database we can query

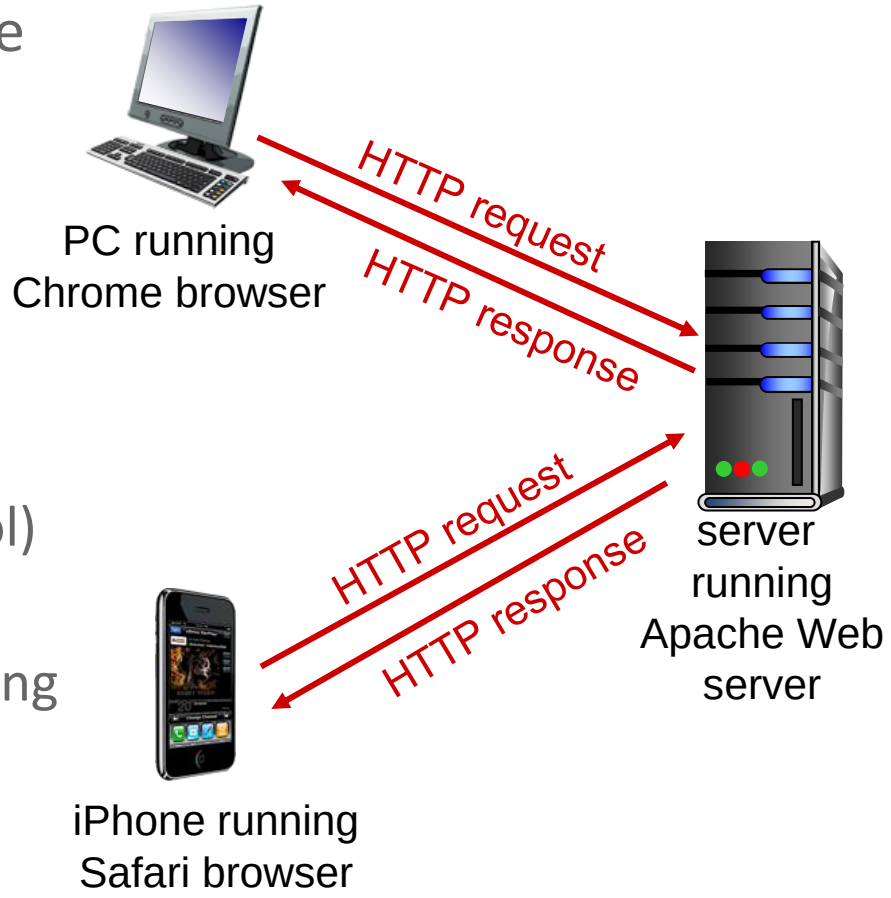
DNS is a network service

- DNS's role, vs. the web server



HTTP request/ response

- For each resource host, resolve IP using DNS
- Make request using HTTP: hypertext transfer protocol
- client/server model
  - client: browser that requests, receives, (using HTTP protocol) and “displays” Web objects
  - server: Web server sends (using HTTP protocol) objects in response to requests



HTTP & TCP

Lancaster University

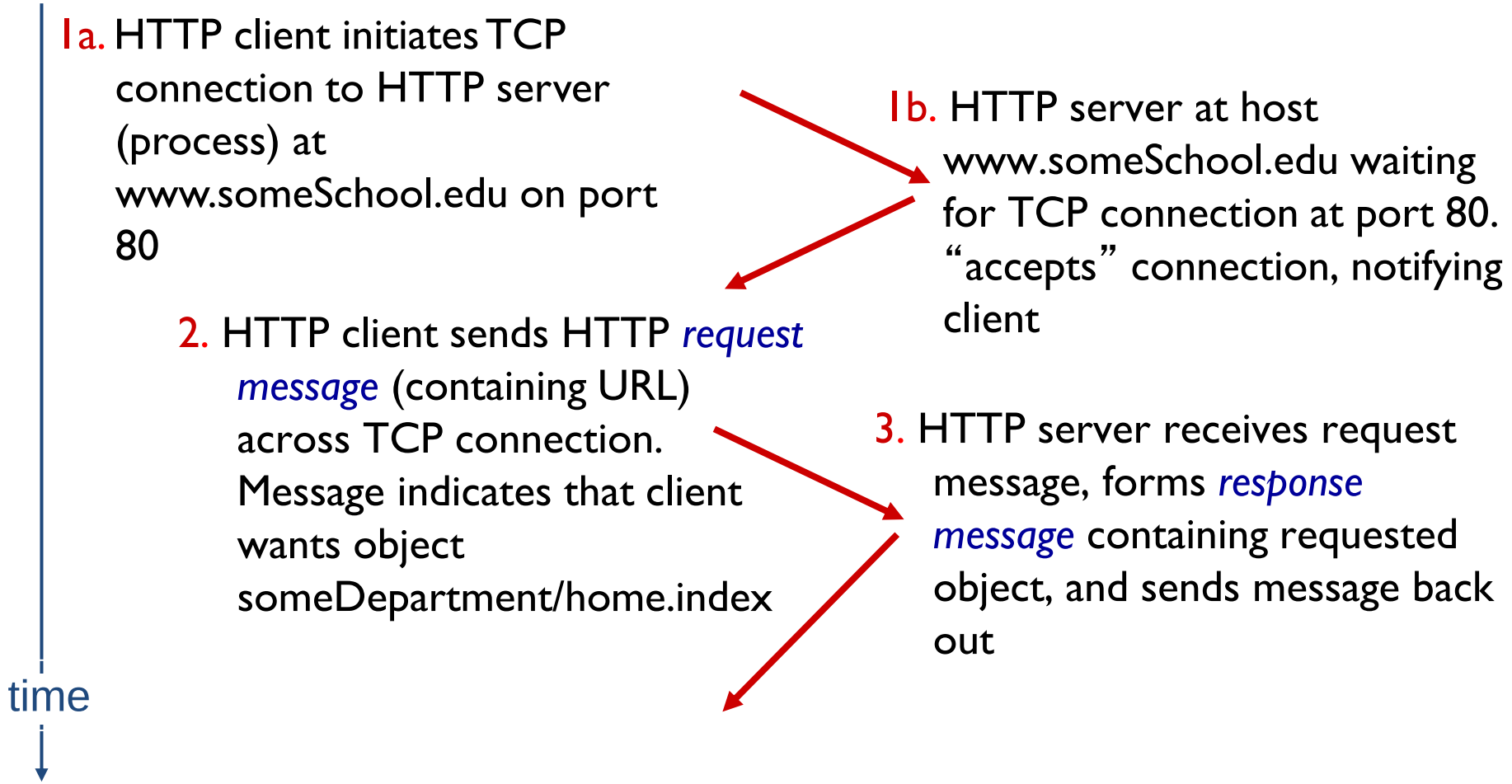
- Uses **Transmission Control Protocol** (TCP):
  - Client initiates TCP connection (creates socket) to server, port 80
  - Server accepts TCP connection from client
  - HTTP messages (application-layer protocol messages) exchanged between browser (HTTP client) and Web server (HTTP server)
  - TCP connection closed

HTTP in Action

suppose user enters URL:

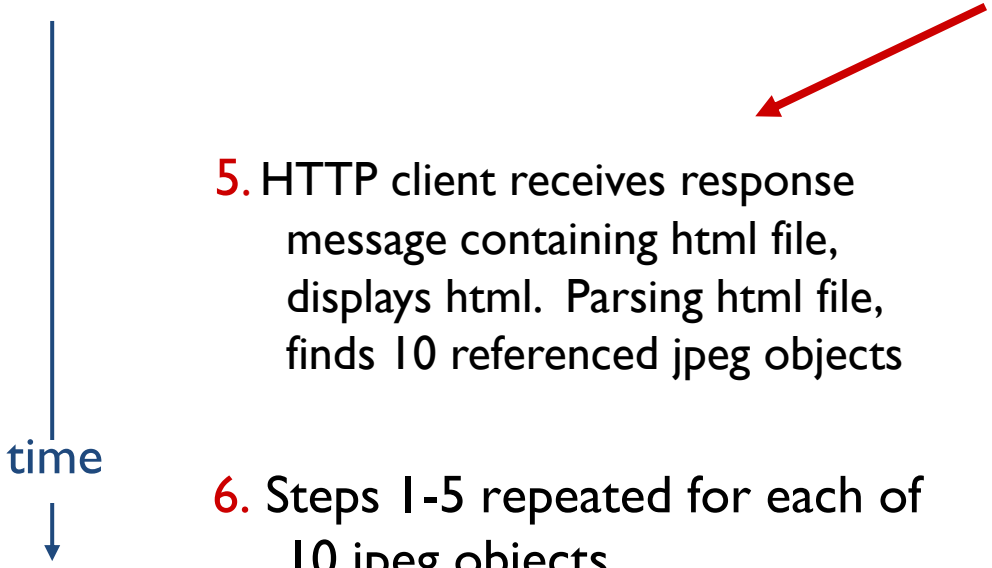
`www.someSchool.edu/someDepartment/home.index`

(contains text, references to 10 jpeg images)



HTTP in Action (cont.)

Lancaster University



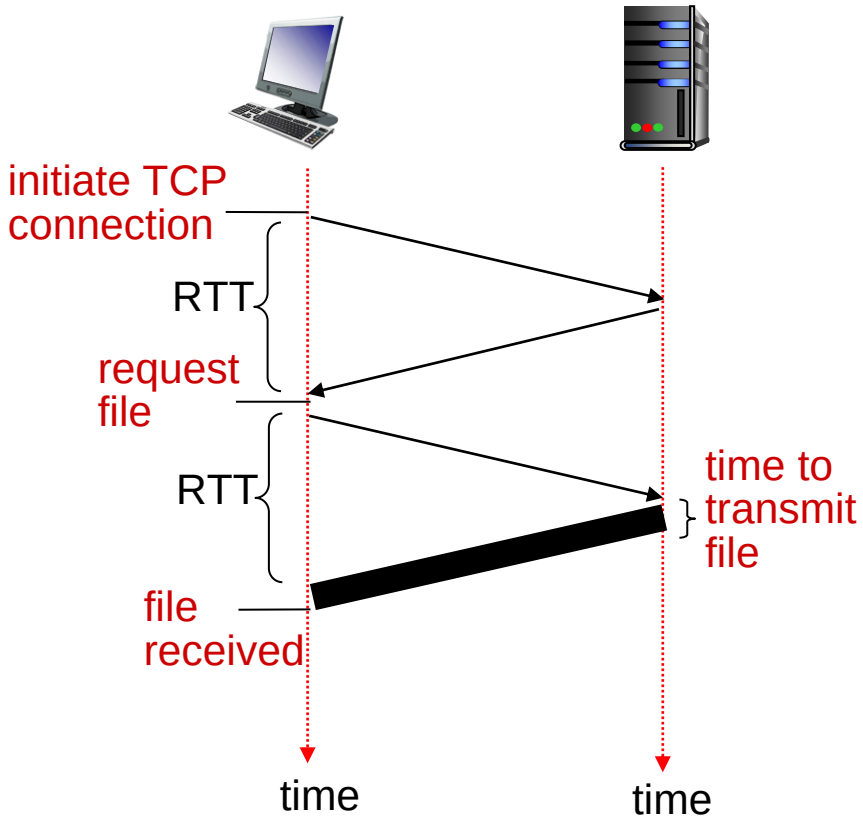
4. HTTP server closes TCP connection.

5. HTTP client receives response message containing html file, displays html. Parsing html file, finds 10 referenced jpeg objects

6. Steps 1-5 repeated for each of 10 jpeg objects

HTTP response time

- RTT (definition): time for a packet to travel from client to server and back
- HTTP response time:
  - one RTT to initiate TCP connection
  - one RTT for HTTP request and first few bytes of HTTP response to return
  - file transmission time
  - non-persistent HTTP response time =
    - 2RTT+ file transmission time



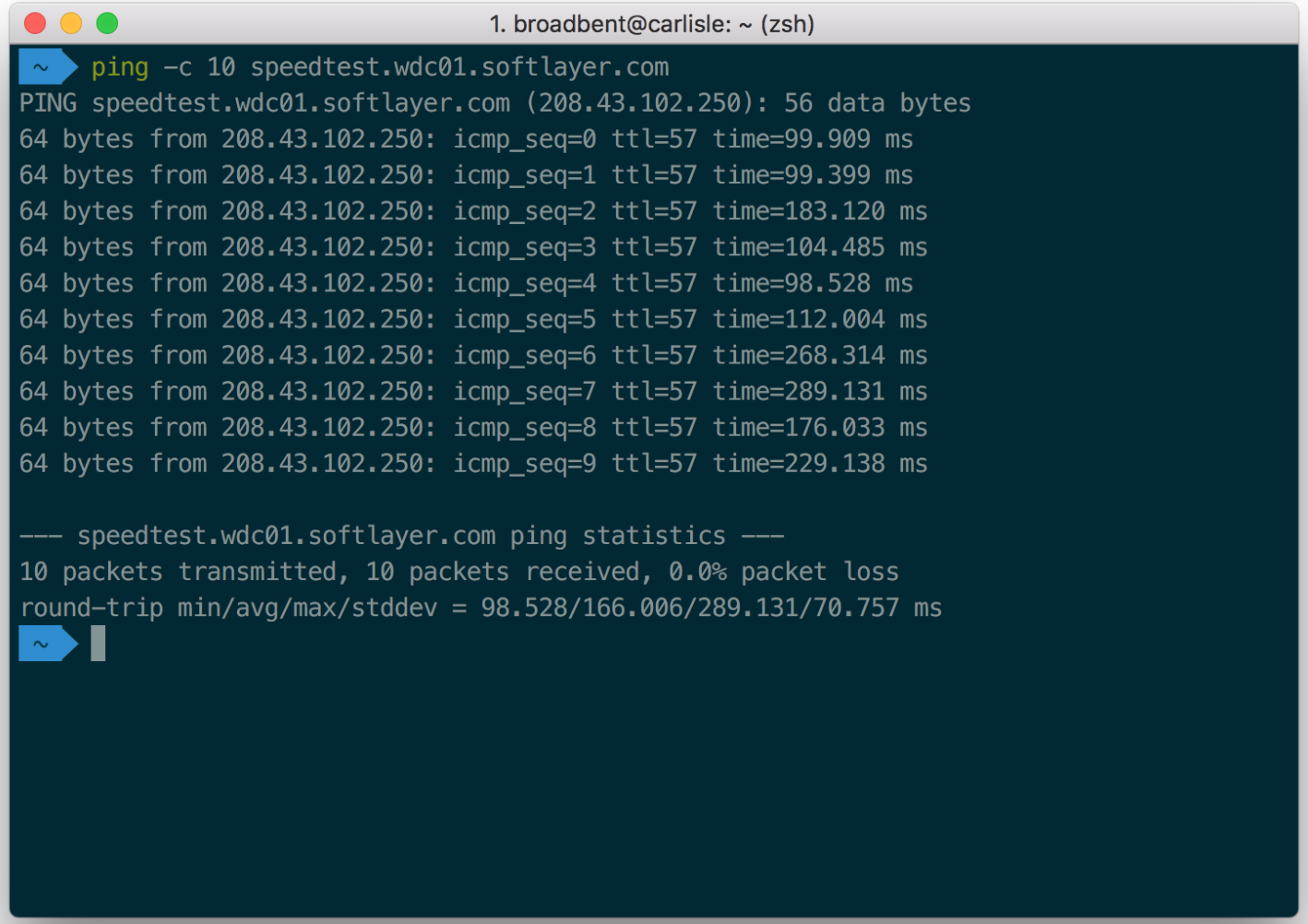
Defining Latency

Lancaster University

- A measure of the time **delay** experienced by a system
- In a computer network: *the time it takes for a packet of data to get from one designated point to another*

Defining Latency

Lancaster University



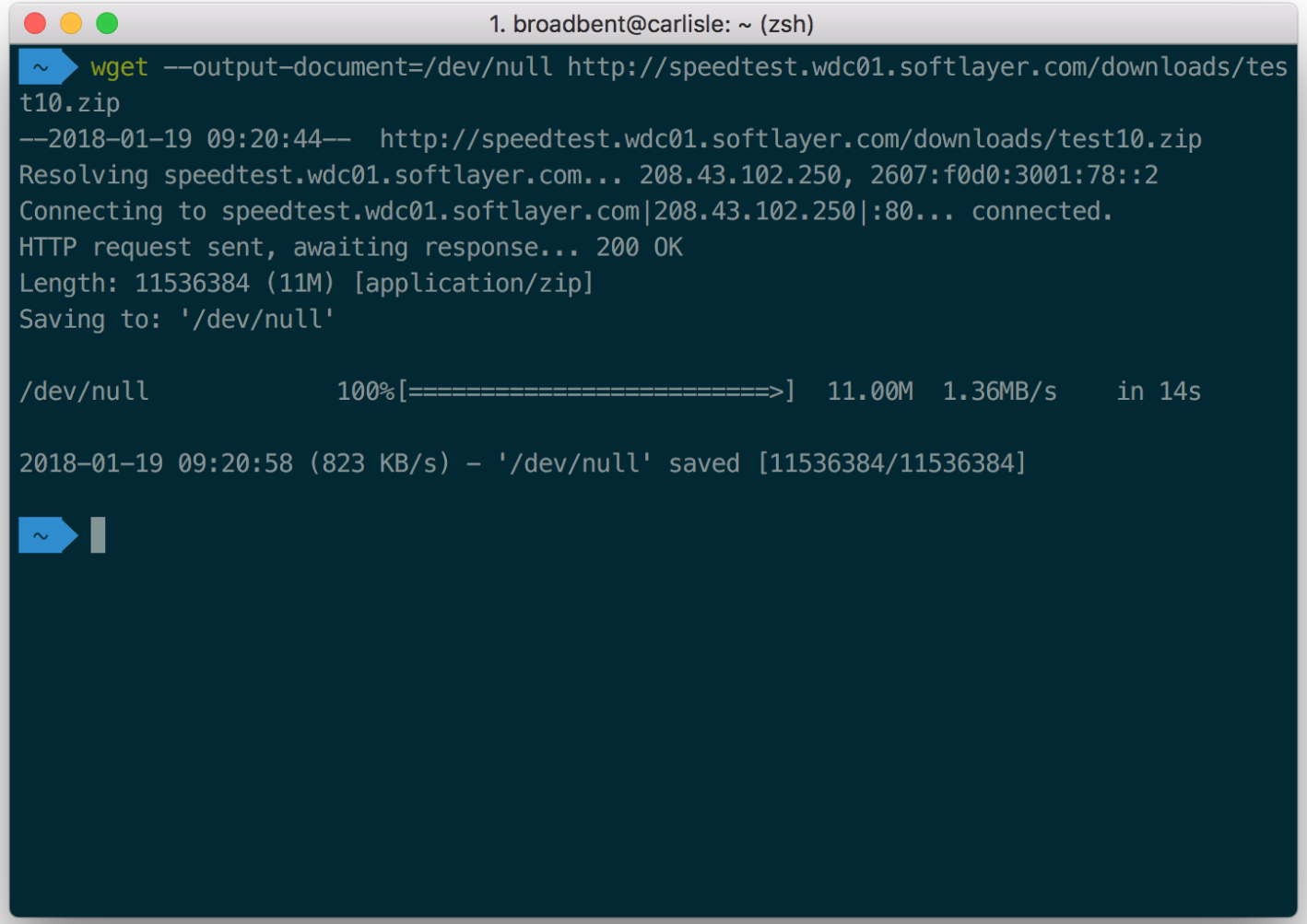
Defining Throughput

Lancaster University

- **Bandwidth**: the amount of information that can be transmitted over a **network** in a given time
- Synonymous with: **network throughput**, the amount of data moved successfully from one place to another in a given time period

Defining Throughput

Lancaster University



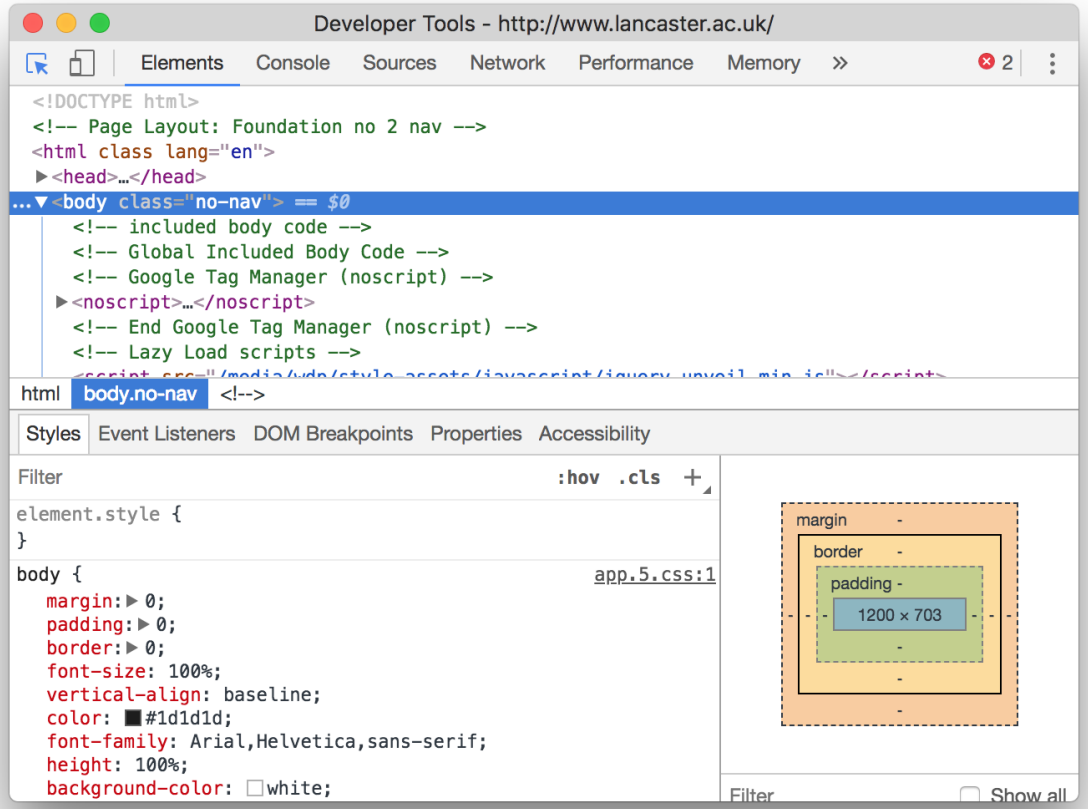
Lancaster University

Part II: Front-end Web Performance and Measurement

The Document Object Model (DOM)

Lancaster University

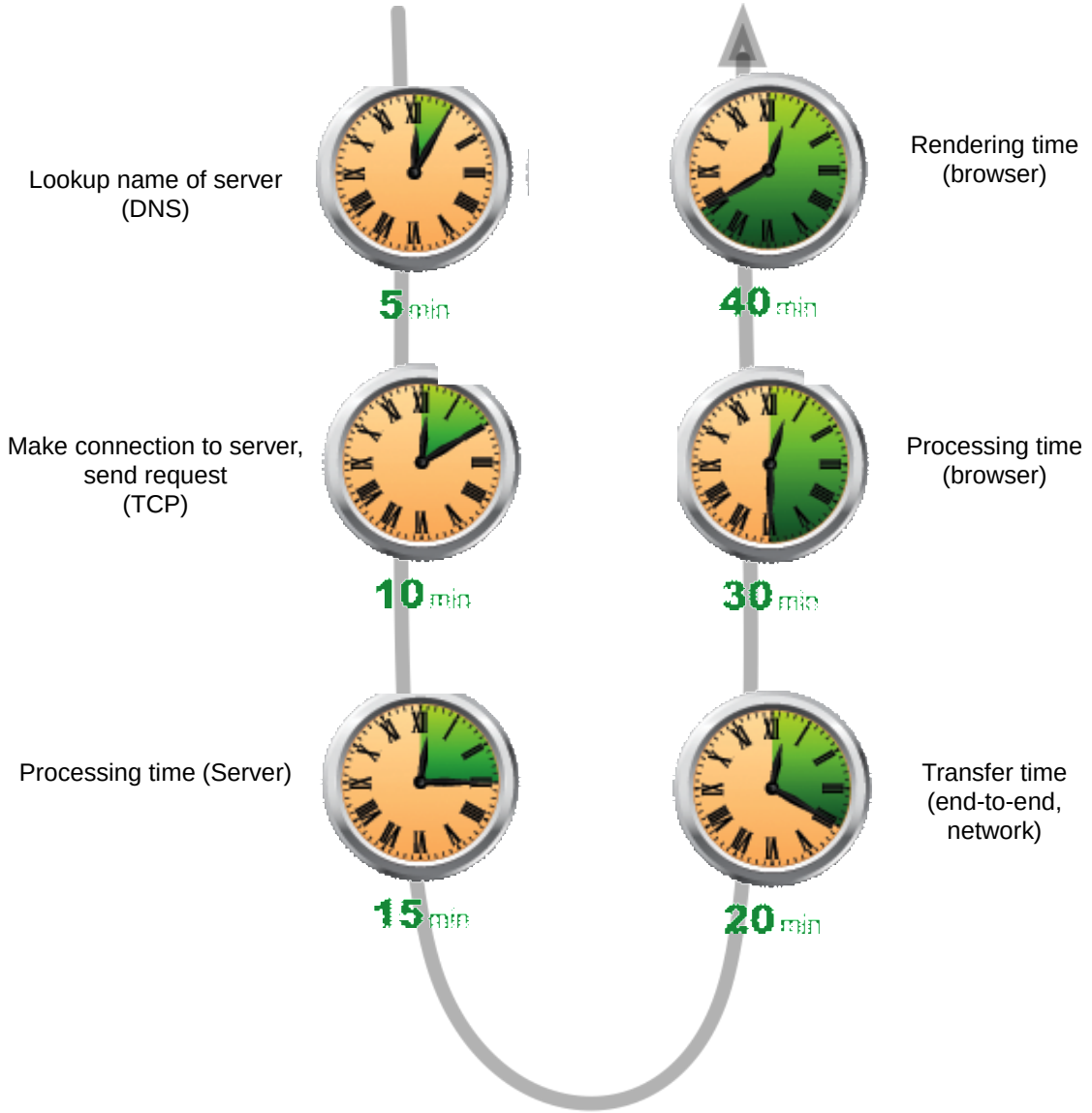
- Way to manipulate the structure & style of HTML
- Represents internals of the page as the browser sees it



Where does the time go?

- Latency (in milliseconds, not minutes ☺), i.e. the time taken to complete each stage before the content can be displayed.

- (Much) more detail: http://bit.ly/1uHwuPD



This means…

Lancaster University

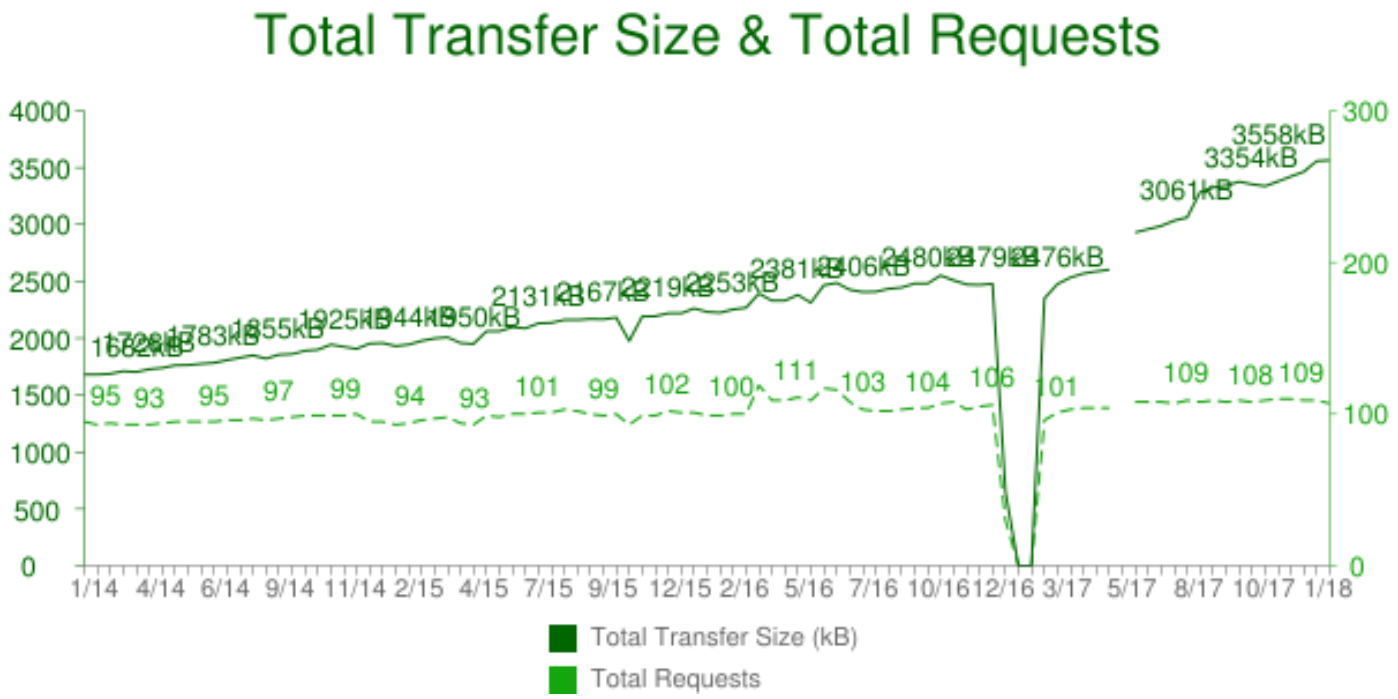
- Each time we fetch a piece of content we incur time penalties (**latency**)
  1. Looking up the domain name (DNS)
  2. Sending and receiving HTTP messages (due to the latency / delay and bandwidth & congestion of the network and the size of the content)
  3. The performance and load on the server
  4. The complexity of the pages we render and performance of the client

  This is all dynamic, and changes over time.

HTTP Archive
Top 100 Stats

Lancaster University

- http://httparchive.org/interesting.php
- http://httparchive.org/trends.php



Jan 2014 – Jan 2018

Lancaster University

Understanding website performance

Waterfalls 101 – note: green and blue lines

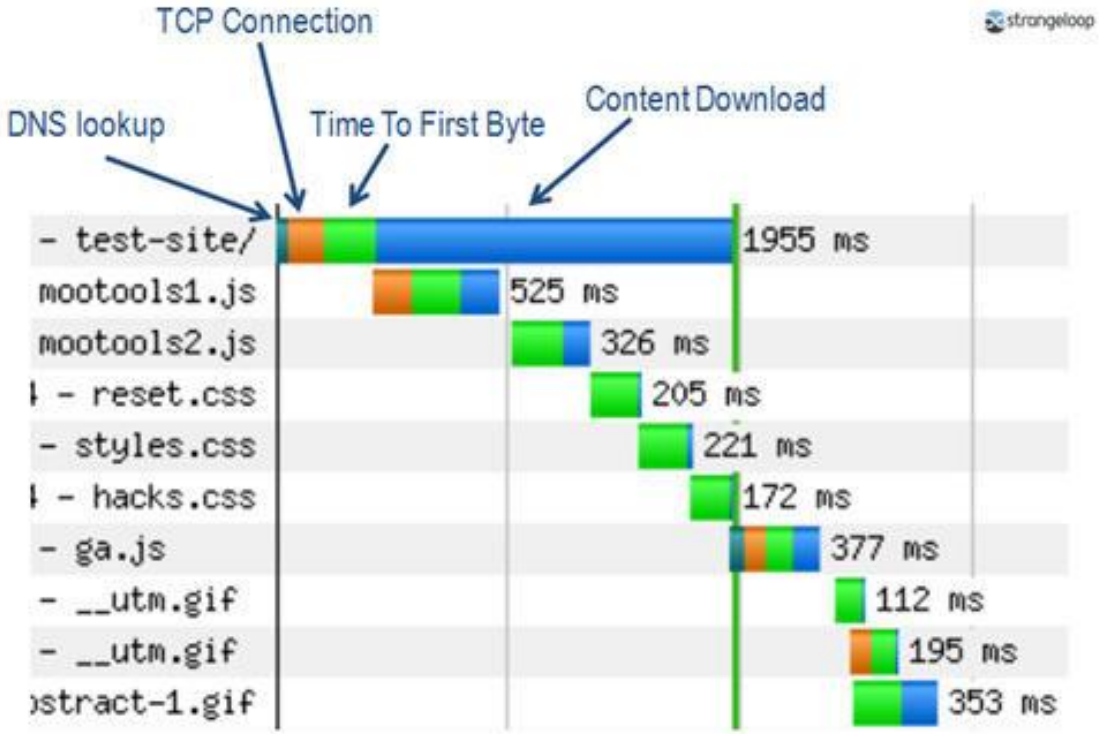
Lancaster University

http://bit.ly/9WRauQ

Each web page contains 50-100 objects†

Lancaster University

- Each making a round-trip between the browser and the server (request out, processing time, response back)
- †http://bit.ly/9WRauQ



Definitions

- **Time to first byte** is the time from the request to the server until the first byte of the response is received by the browser
  - > 100ms or so, and you have a slow server
- **Start render** is when content begins to display in the user's browser (doesn't mean useful content!)
  - Should be consistent across pages and less than 2 seconds
  - *Note: browsers continue to evolve when they draw (e.g. Chrome)*
- **DOMContentLoaded** is when the initial HTML document has been completely loaded and parsed, sufficient to build the DOM (without necessarily waiting for some objects, such as images of known size, to finish loading).
    - DOMContentLoaded = Start Render?

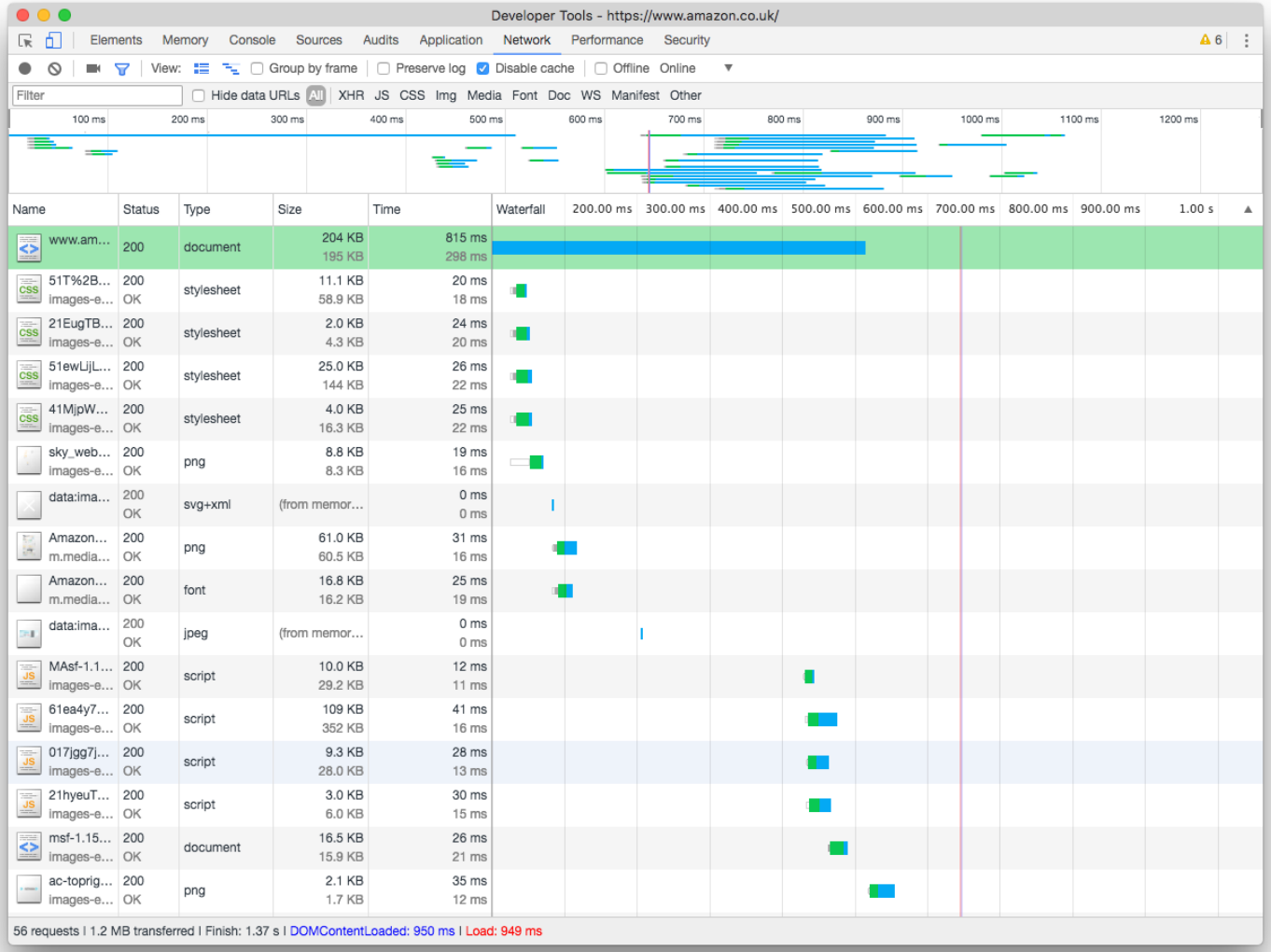
Definitions

- **Load time** *or* **document complete** time *or* **onLoad** time
  - All of the document's resources (i.e. images and CSS files, etc) have been fully loaded
  - Usually a site is interactive before this
    - May be when some key javascripts activate
    - Helps compare coarsely with other sites
    - Content might be *after the fold** (i.e. outside the viewport) but still loading

* Jakob Nielsen estimates that, *"…web users spend 80% of their time looking at information above the page fold. Although users do scroll, they allocate only 20% of their attention below the fold."*

Waterfalls a Common Visualisation Technique

Used in this course: Chrome DevTools



Lancaster University

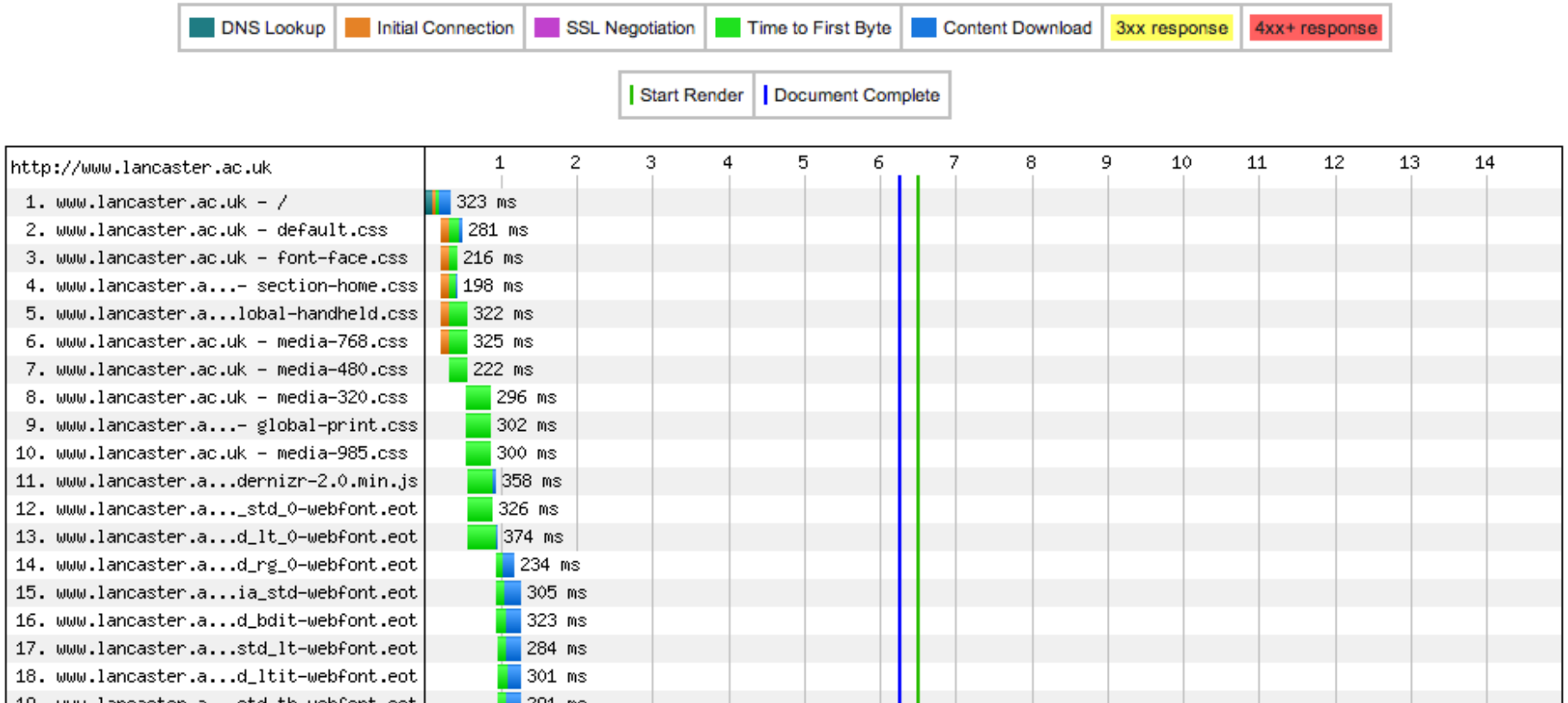
Drilling down on performance problems

So, where *exactly* is the problem?

Example page
(with webpagetest.org)

							Document Complete			Fully Loaded		
Load Time	First Byte	Start Render	Visually Complete	Speed Index	DOM Elements	Result (error code)	Time	Requests	Bytes In	Time	Requests	Bytes In
6.242s	0.163s	6.486s	15.540s	12790	548	0	6.242s	88	1,421 KB	15.460s	111	1,694 KB

Waterfall View



Lancaster University

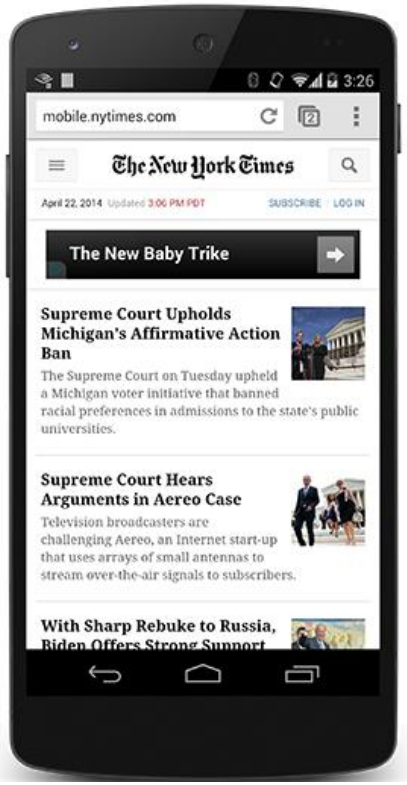
So, a good place to start is to ***focus on render time***

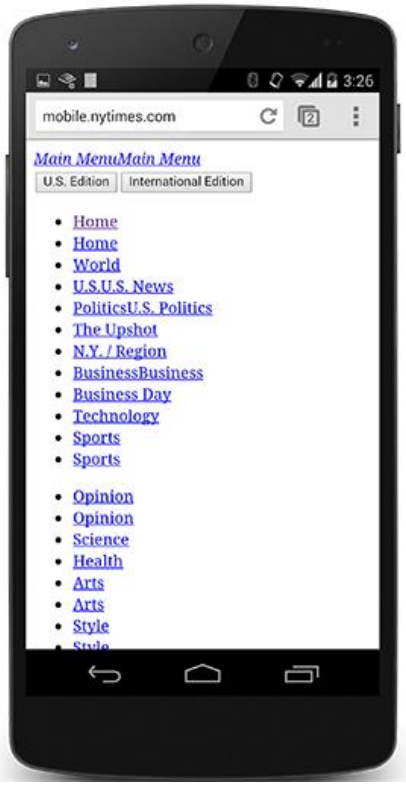
Before we render

- The browser must *fetch any linked content* that is *required* to render the page
  - i.e. might effect the Document Object Model (**DOM**)
  - This includes style sheets (**CSS**)
  - And external javascript source files (**JS**)
  - External files depend on the full network round trip and can block rendering
  - By default, CSS is **render blocking** unless marked otherwise
    - Although this is changing: browsers evolving to animate changes

HTML and CSS are “render blocking” resources

- The HTML is obvious, since without the DOM we would not have anything to render
  - But the CSS?
- http://bit.ly/1ydoYhV





Lancaster University

How can we improve performance?

Tip 1 – Order is important

- Some browsers block rendering until the stylesheet is loaded to avoid redraws
  - CSS goes in the head so the DOM can start render more quickly
  - Javascript can often go in the body as you don't need the scripts until the page is available

Source Code: Submit Code »

```
<!DOCTYPE html>
<html>
<body>

<h1>My Web Page</h1>

<p id="demo">A Paragraph.</p>

<button type="button" onclick="myFunction()">Try it</button>

<p><strong>Note:</strong> myFunction is stored in an external file called
"myScript.js".</p>

<script src="myScript.js"></script>

</body>
</html>
```

Tip 2 – do less

- A simpler page (simpler DOM) will be faster
  - Reduces number of page elements to load and process
- Also, hard to make TCP go faster, but can combine files (CSS, JS)
- Remove duplicates (e.g. redundant JS)!

e.g. CSS

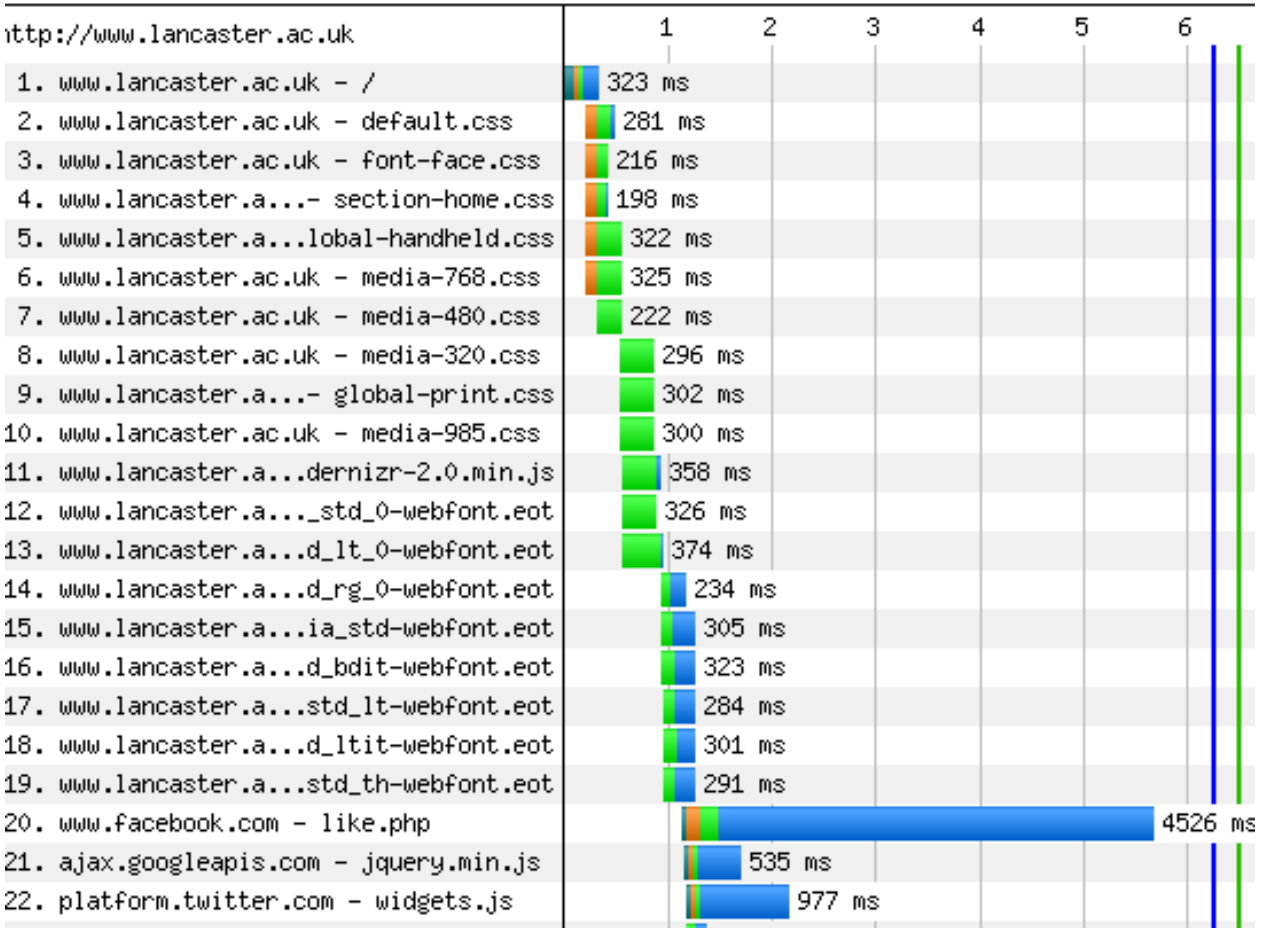
= 1 CSS link

Improving TCP

- “It’s not easy to speed up TCP connection, but you can control how many times the TCP connection takes place”, http://bit.ly/9WRauQ

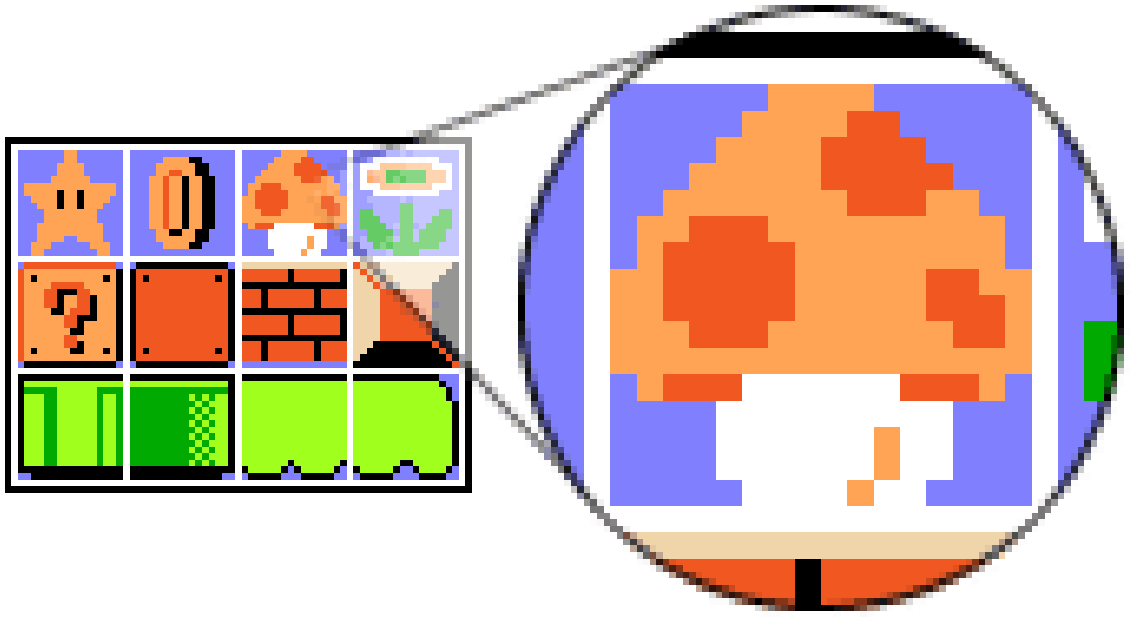
Note large collection of CSS, fonts and scripts blocking rendering

- Until the final stylesheet is loaded… You want less rows!



CSS Sprites

- We can combine multiple image files into a larger **image map** (sprite) that is then ‘dissected’ in CSS – you load one larger image, rather than many small ones, amortising overhead (see http://bit.ly/XqV6dx)

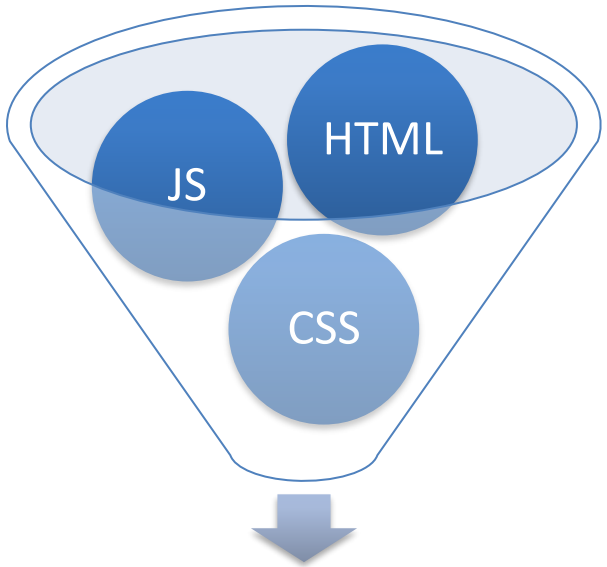


EXAMPLE SPRITES

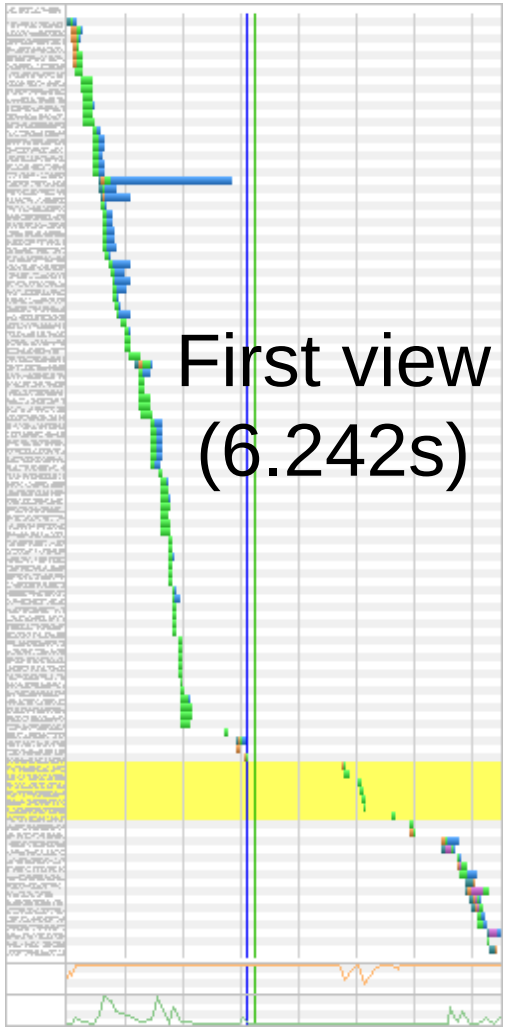
Amazon example

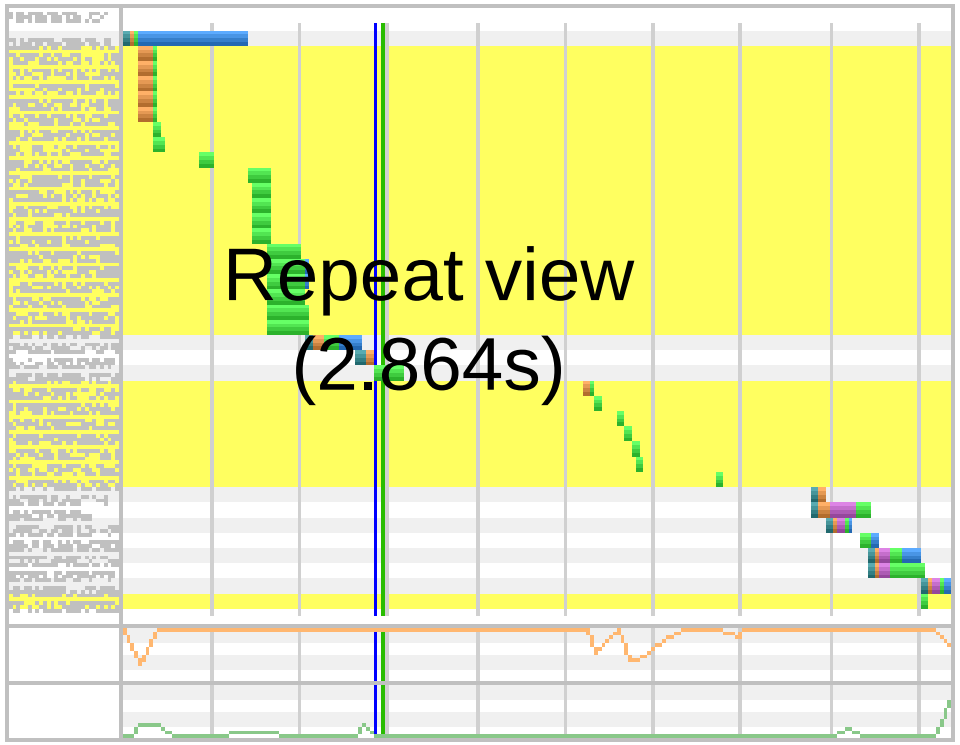
Tip 3 – make things smaller

- Optimise
  - Image sizes/files, make progressive
  - Enable compression (reduces response by 70%, in 90% of browsers!)
  - ‘Minify’ javascript and CSS (e.g. using Google closure compiler/ YUI compressor (+use ‘minified’ versions of JS libs)



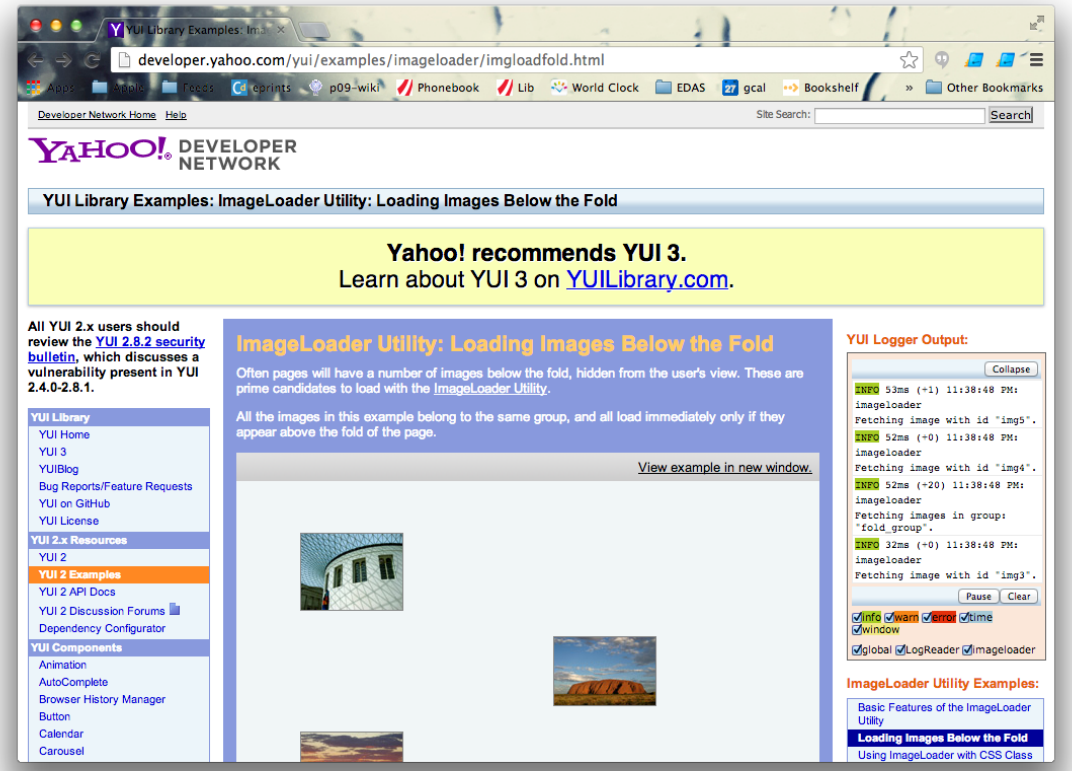
Tip 4 – support caching





Tip 5 – loading after ‘the fold’

- Delay loading of components not visible to the user until they’re needed, e.g. images using YUI ImageLoader.

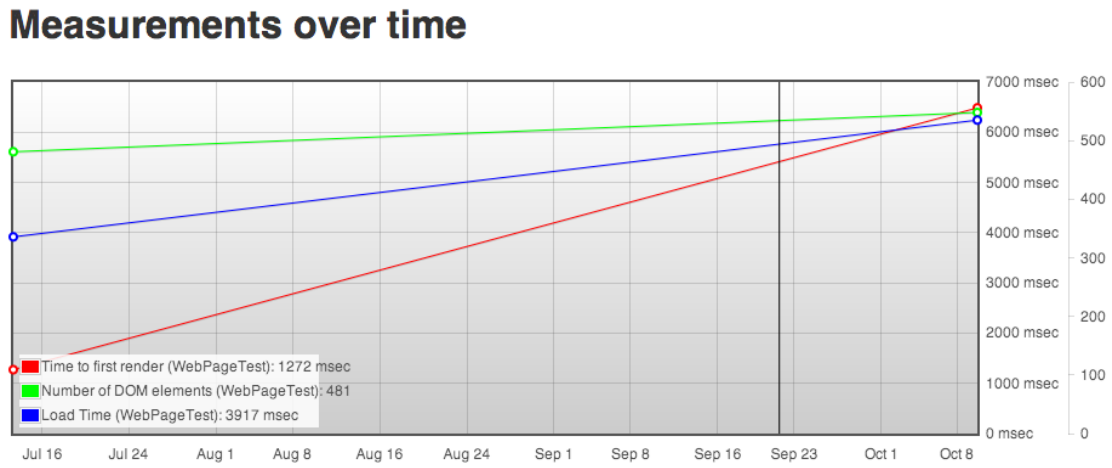


Lancaster University

Pointers for Getting Started

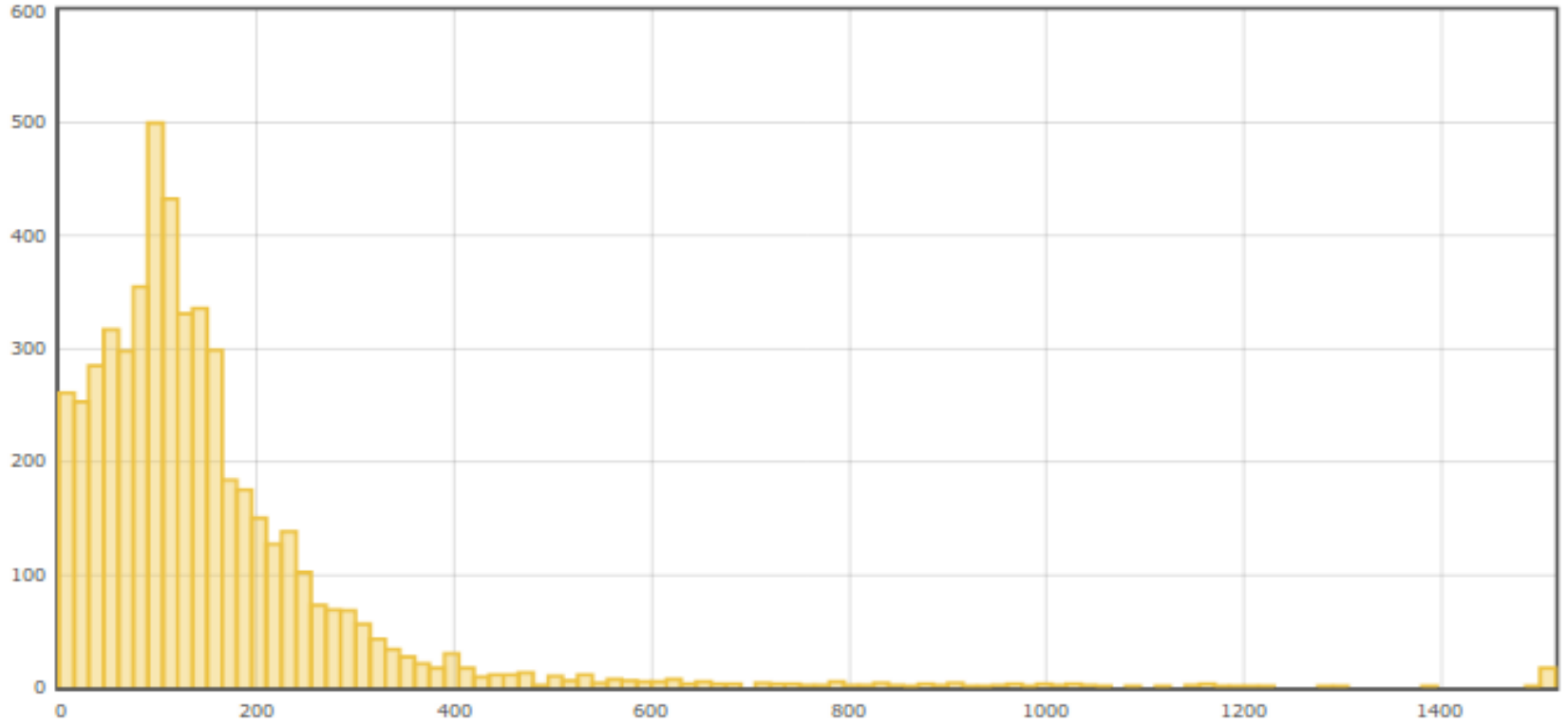
Intuition and Statistics

- Be warned:
  - Intuition about statistics can be very bad…
  - Don’t base your measurements on **just one run** (sampling error) or from one location (typically local!).
- Draw a histogram of responses



Intuition and Statistics

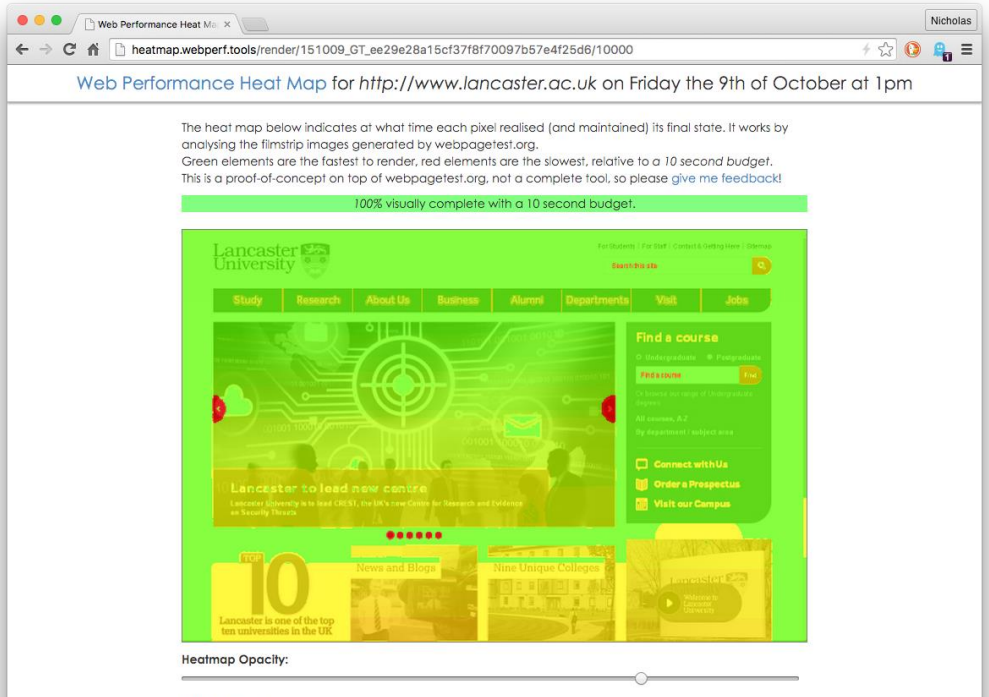
Graph:



(The X axis is the server response in MS, and the Y axis is number of hits that fell into that response bracket)

Other Tools

- http://heatmap.webperf.tools - red means slow
- http://yellowlab.tools – analyse performance bottlenecks
- http://lab.speedcurve.com – visualise performance trends



Lancaster University

Part III: Back-end Web Performance and Scaling

What is scale?

Lancaster University

- The ability handle an increasing number of (possibly concurrent) requests
  - While maintaining performance targets
    - Low latency (short **time to first byte**)
    - High throughput (short **content download**)

Why is scale important?

Lancaster University

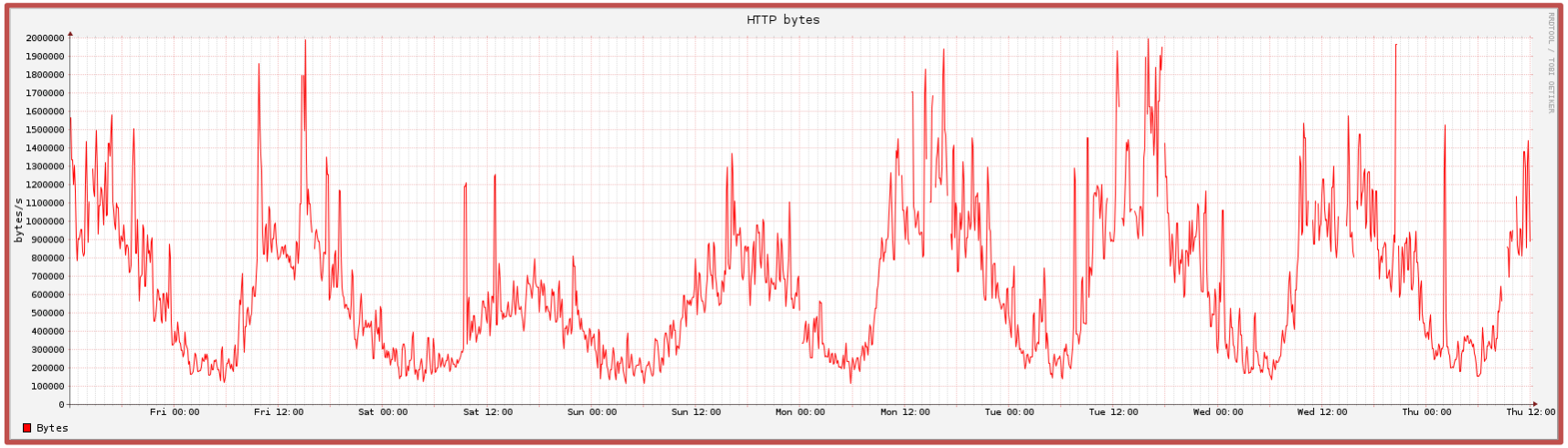
- We've already seen the impact that page load times can have on performance
- But what influence does the infrastructure have on this?
- Consider the amount of people that now have access to the Internet
  - It is inherently global!
- How do we build around this popularity?
  - Handling load *well* is key

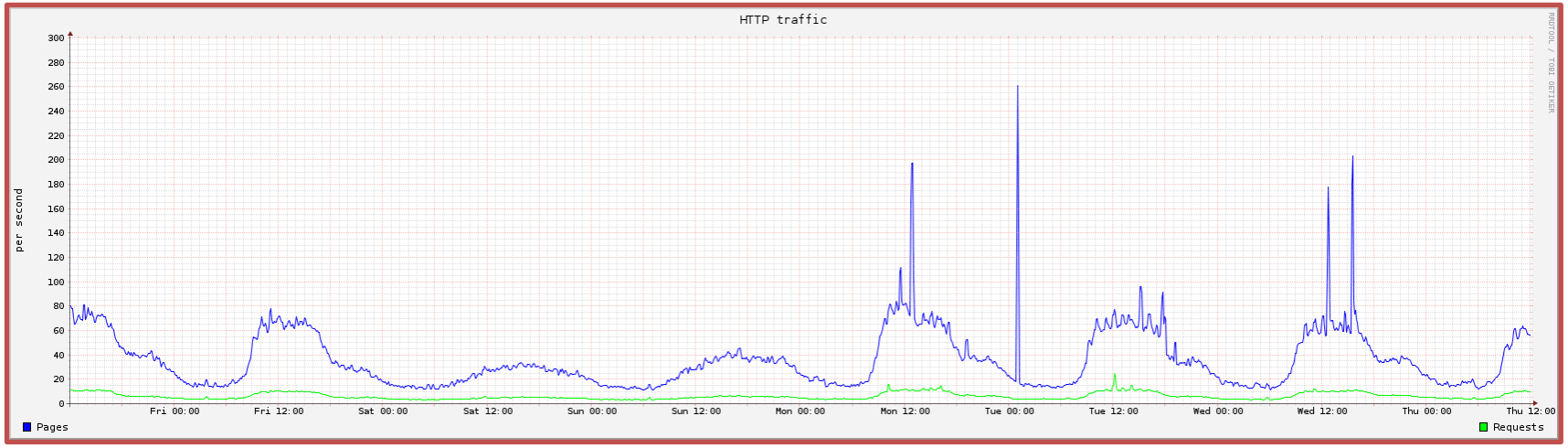
Load changes over time!

Part of the reason that we take multiple measures (recall: *sampling bias*)

www.lancaster.ac.uk

Lancaster University





Metrics that matter

Lancaster University

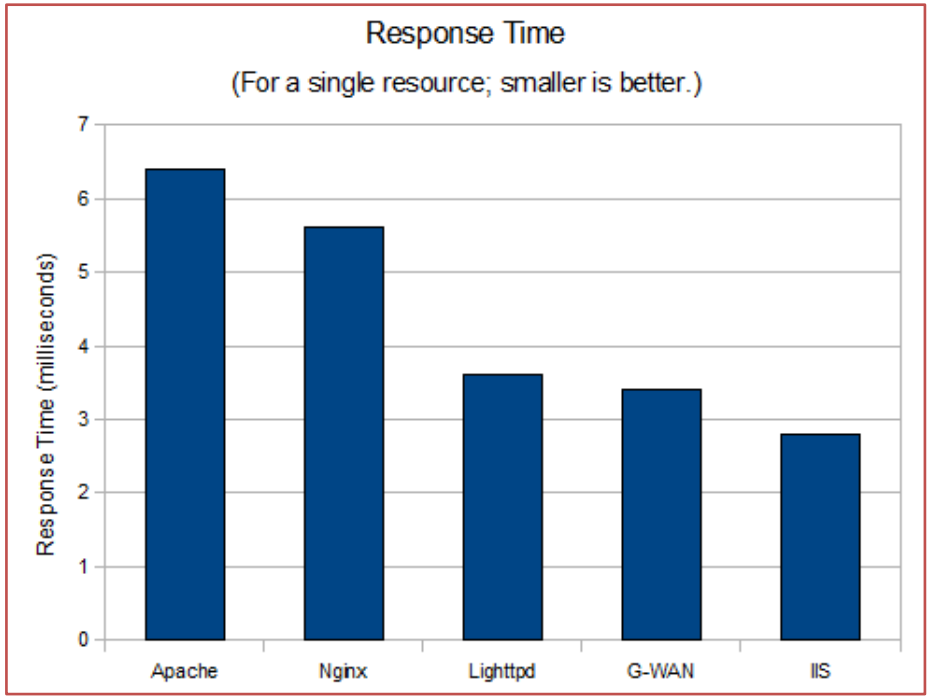
- Number of requests served per second (depends on type of request and work/ request!)
- Latency (response time) for each new connection or request (in ms)
- Throughput in bytes per second (depends on file size, cached or not cached content, available network bandwidth…)

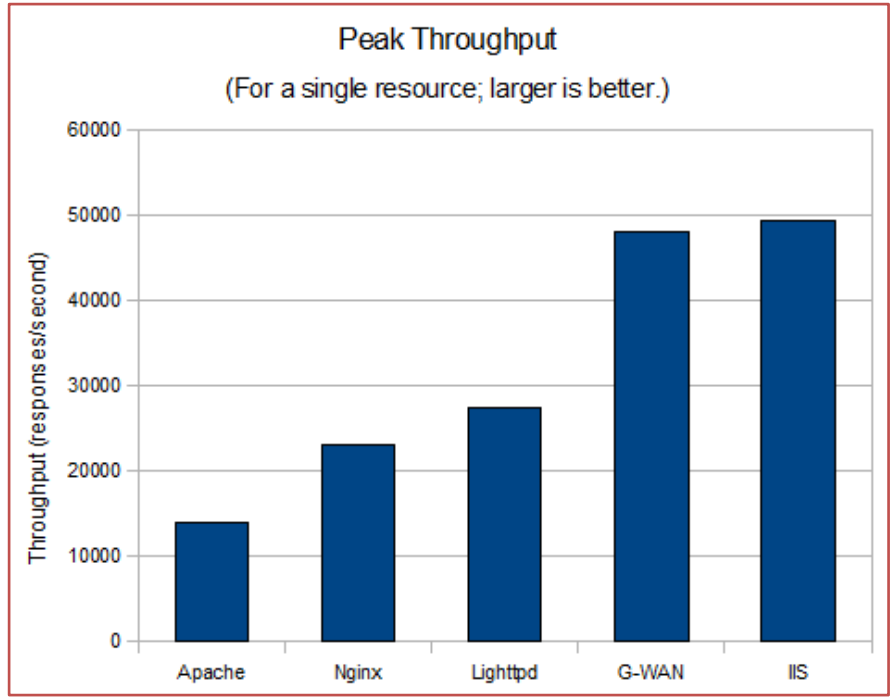
Lancaster University

$$Response\ time \approx \frac{concurrent\ sessions}{requests \times second^{-1}}$$

Response time is inversely proportional to requests per second

We sometimes have to provision our system to trade these off. If we spend too much time beyond steady state, then we need to *do something about it!*





And not all web servers are the same

Web Performance, 'The fastest web server', 16.11.2011. https://www.webperformance.com/load-testing-tools/blog/2011/11/what-is-the-fastest-webserver/

Lancaster University

$$Requests/sec \approx \frac{connections}{time}$$

Requests per second is page requests / time

This can be gleaned from the logs.

Lancaster University

$$Bytes/sec \approx \frac{\sum size\ of\ requests}{time}$$

Bytes per second is the sum of the size of each request / time

Again, this can be gleaned from the logs.

Lancaster University

Optimise *before* scaling

Reducing server load (by far the easiest/cheapest option!)

Buying a bigger box is quick.
Redesigning software is not.

Spend months federating your database, or buy loads of RAM? ☺

Vertical Scaling == bigger/ upgrades

More processors, more memory, gets exotic & expensive

Has the nice property that nothing drastically changes

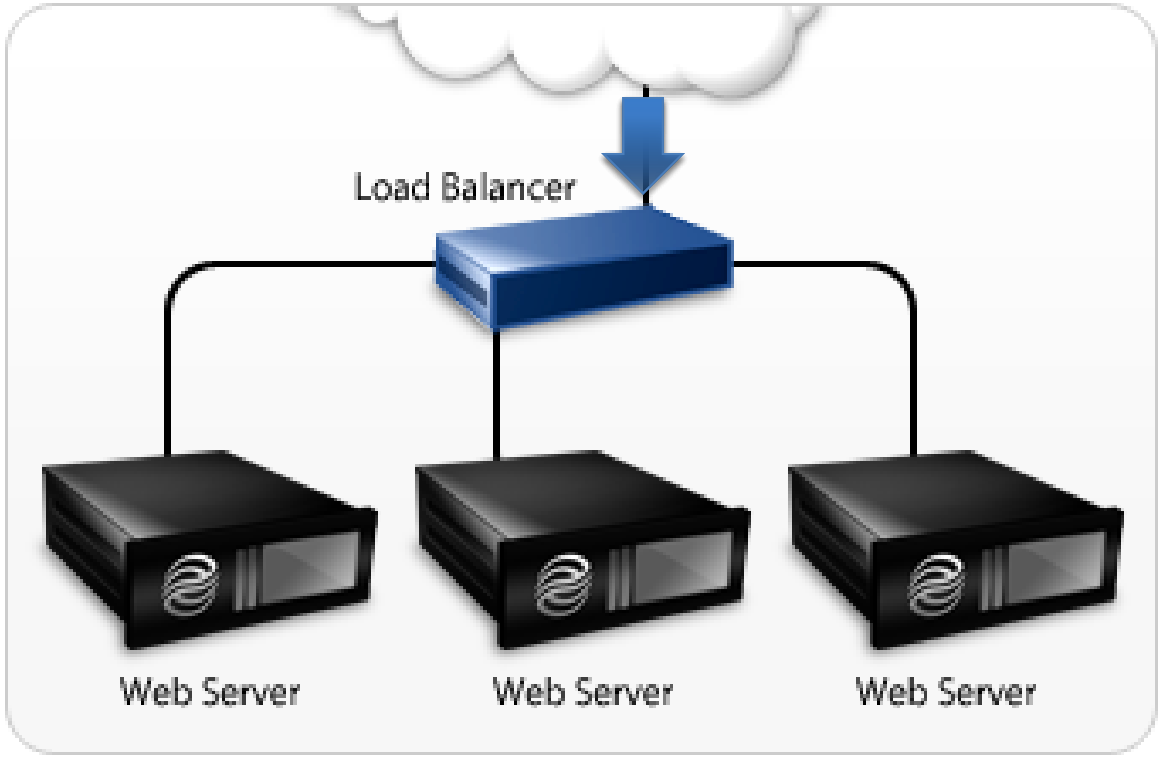
Horizontal Scaling == more hardware

But typically simpler and lower cost (the hardware). This will complicate all but the simplest of scenarios!

Load balancers

Lancaster University

- Distribute the load evenly across two or more servers
  - High performance/cost hardware: Load Balancer
    - Rewrites headers at Layer 4/adjusts DNS
    - TCP connection between client and server (end-to-end)
  - Software (perlbal, nginx) – ‘reverse proxy’
    - Echos HTTP requests to particular servers (layer 7)
    - TCP connection between client and balancer



EXAMPLE OF A LOAD BALANCED HOSTING ENVIRONMENT

http://www.liquidweb.com/

Replication

All servers store the same set of content. Read-only things, ‘static resources’, scale well.

What replicates well?

Lancaster University

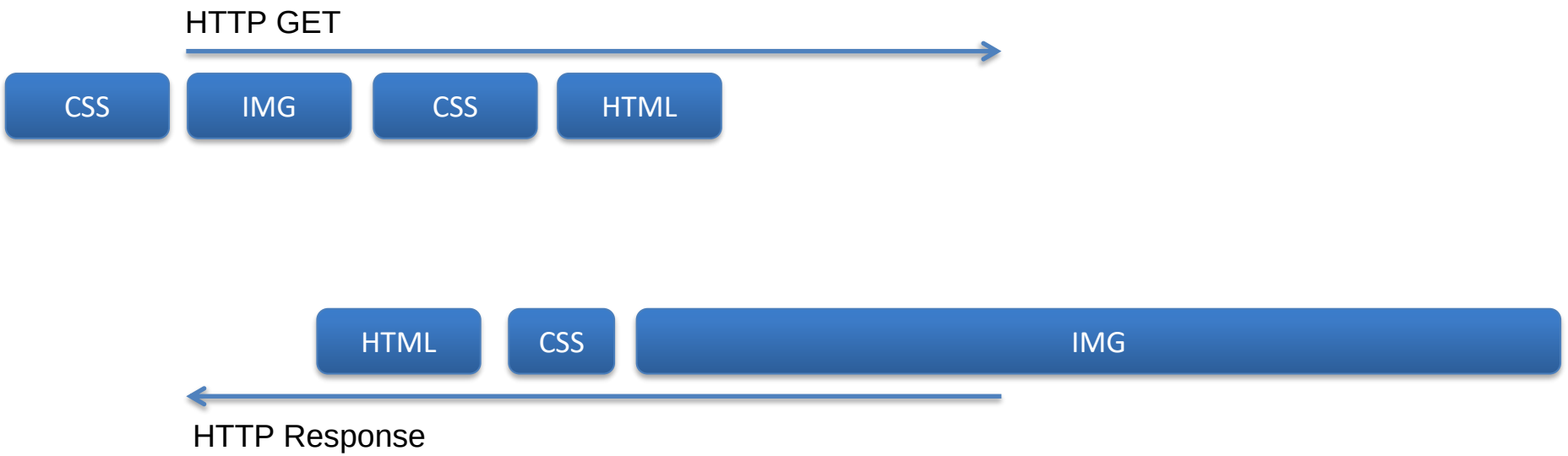
- Things that don’t change very often
  - Media assets (images, movies, sounds)
- But also self-contained tasks:
  - Image conversion
  - Audio transcoding
  - Video transcoding
  - Web crawling
  - Compute

Session tracking and state

Lancaster University

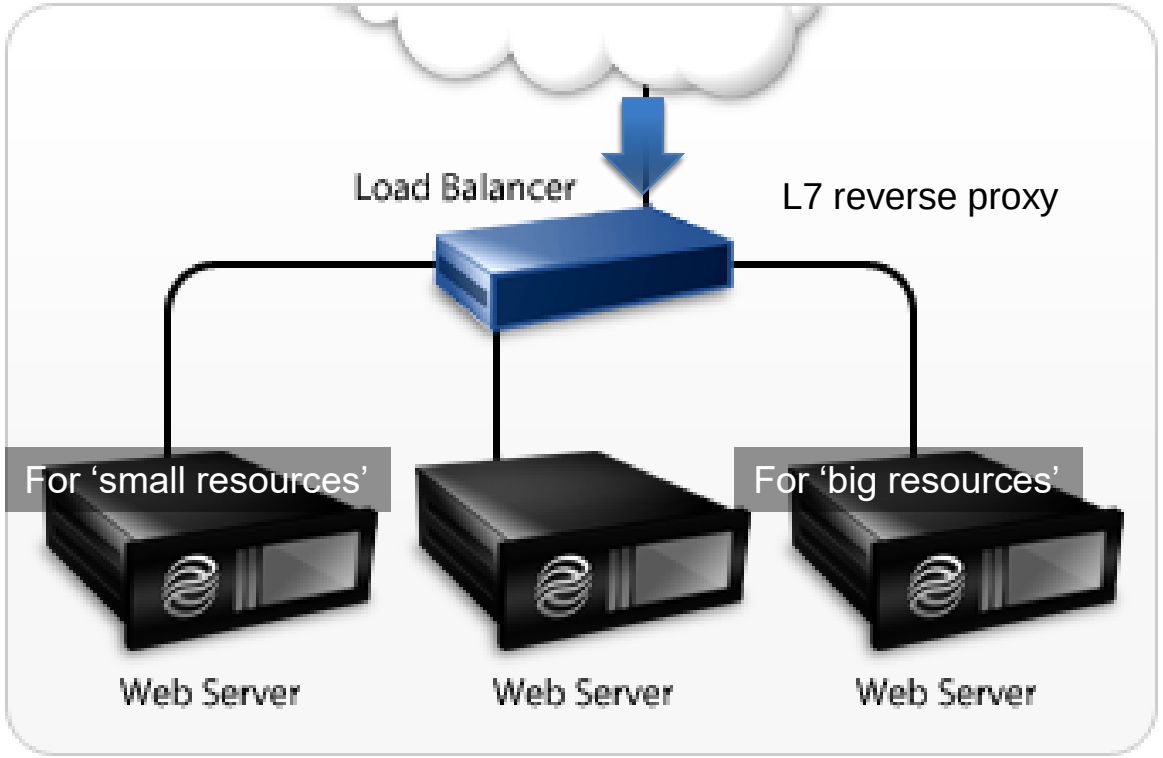
- Fly in the ointment of scaling: personalisation
  - User sessions, e.g. state of interaction, shopping basket etc.
  - Sessions are state that is built up between the client and the server
  - Load balancers have ‘affinity’ to certain servers, but *this defeats the object!*
- Also: User generated content
  - OK, some of it may go viral
  - But most of it is highly specialised and interesting only to a small group of people
  - Is it worth caching?

HTTP 1.1 Pipelines requests to the same server in order



The response to the 2nd CSS request is 'blocked'

Requests are serviced in order… so the small requests have to wait

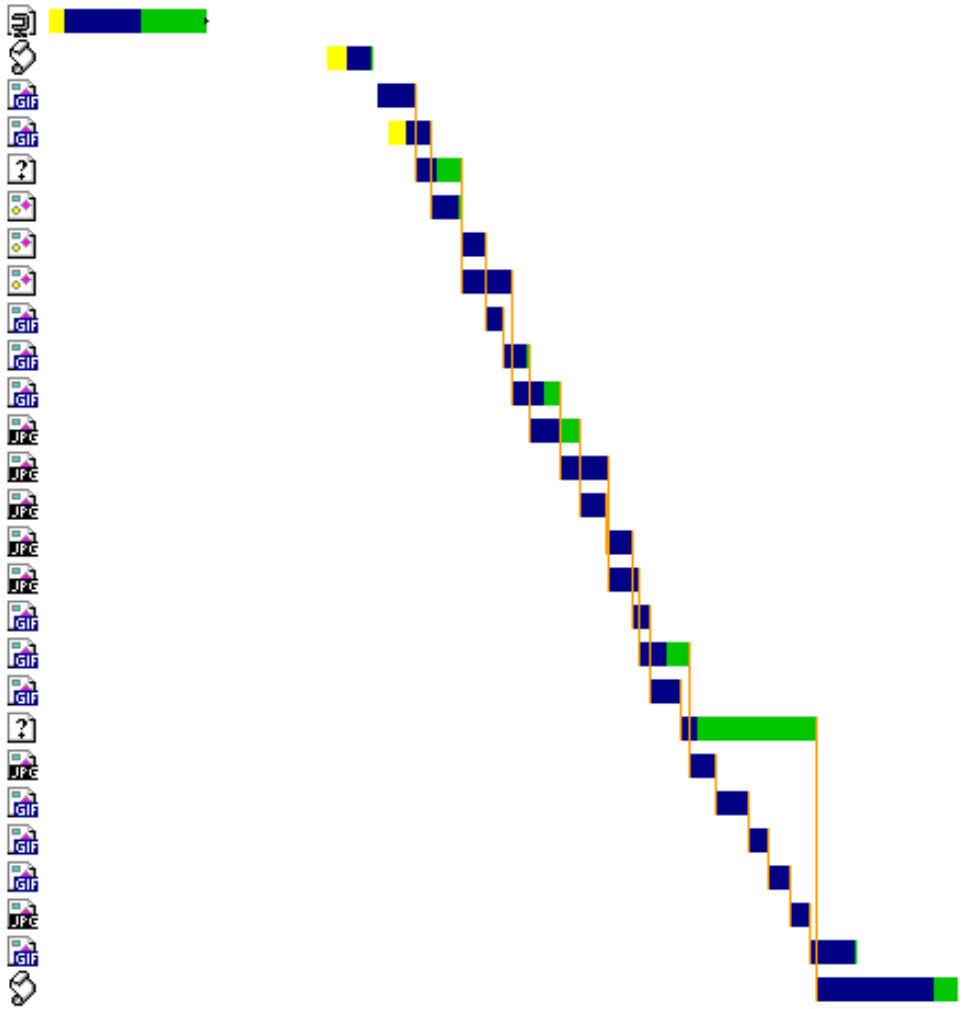


EXAMPLE OF A LOAD BALANCED HOSTING ENVIRONMENT

http://www.liquidweb.com/

Distribution

Split requests across groups of servers. Serve the essentials for rendering the page quickly. Let the large media items fetch more slowly from other servers.



Domain Sharding

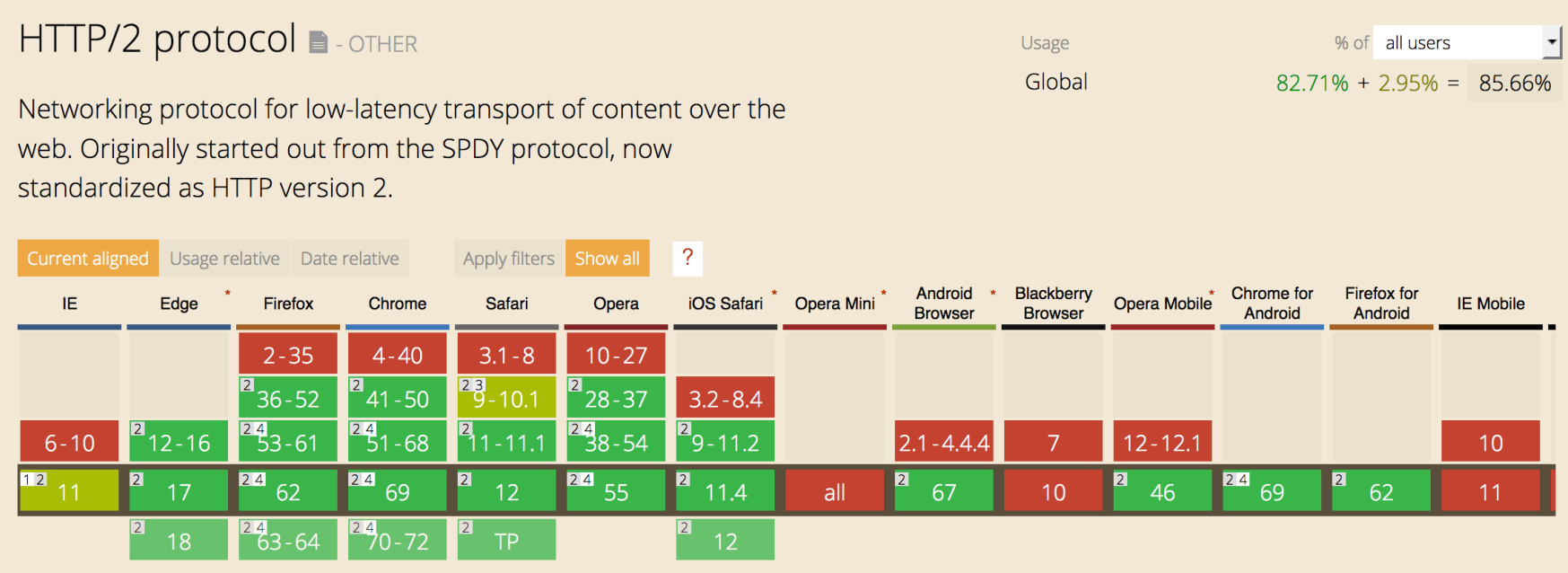
http://www.stevesouders.com/

Different server name = different connection! YouTube uses i1.ytimg.com, i2.ytimg.com, i3.ytimg.com, and i4.ytimg.com. Live Search uses ts1.images.live.com, ts2.images.live.com, ts3.images.live.com, and ts4.images.live.com…

HTTP/2

Lancaster University

- Multiplexed streams
- HTTP header compression
- Servers can *push* content



Lancaster University

Where is the bottleneck?

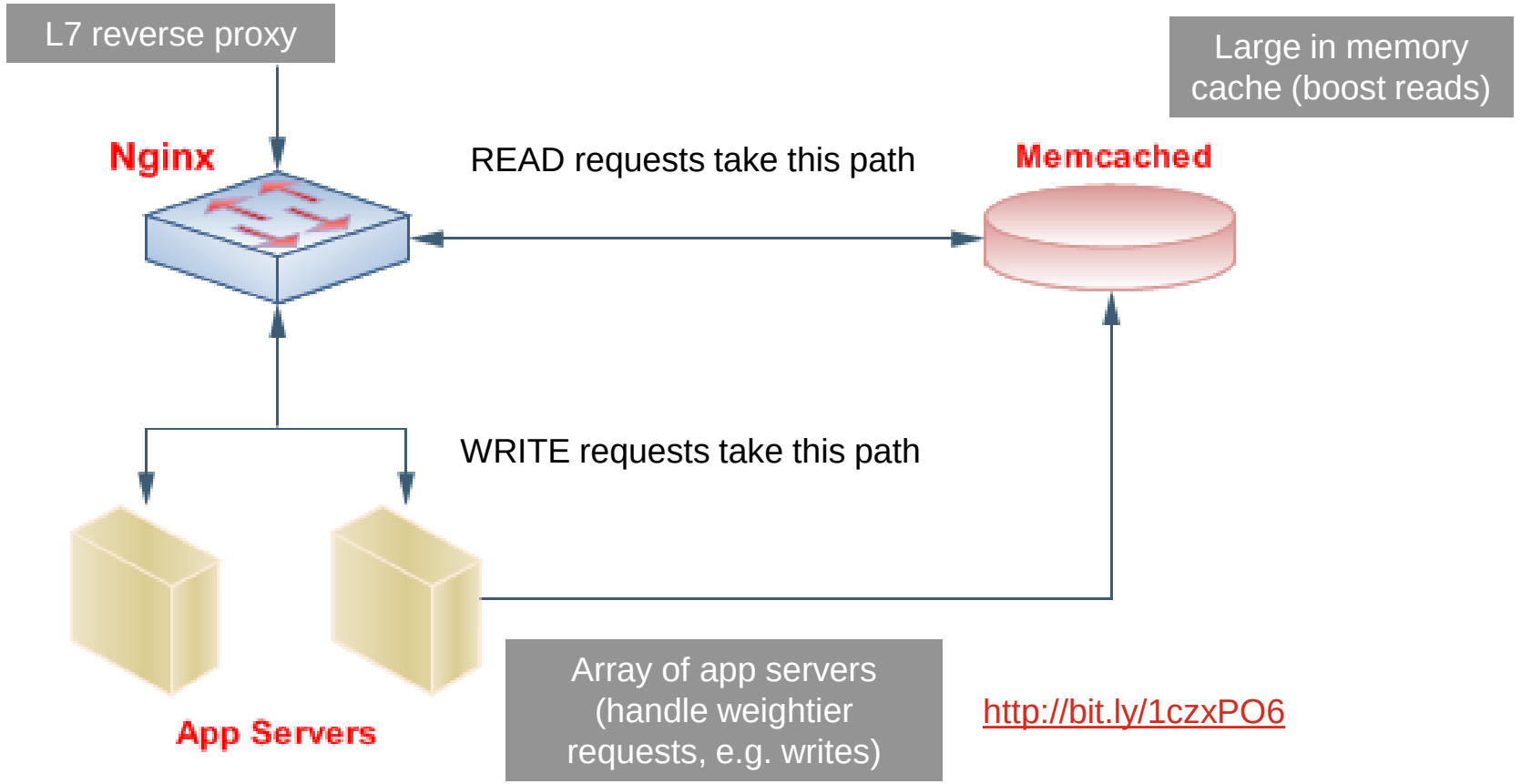
How can we avoid it?

Caching Exploits Locality

Lancaster University

- Move content (web pages, videos, images, etc.) closer to the user
- Less hops == less latency (less transport and in network cost)
- Hopefully less narrow pipes between you/them
- Or less congestion (e.g. the transatlantic link)

- Caching can be done:
  - In-browser
  - In-network
  - At the server

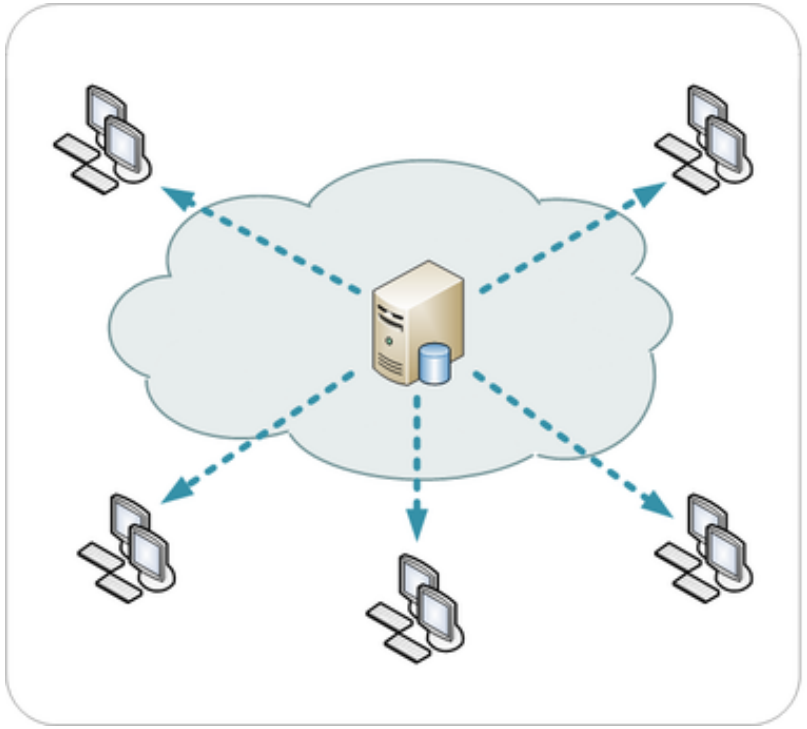


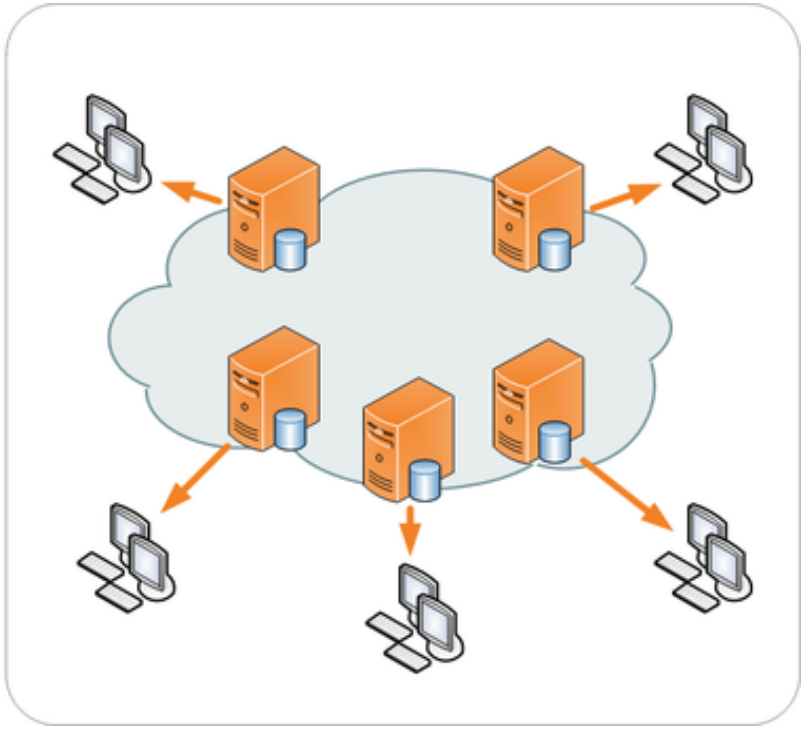
(At the server) Caching

Optimising read performance of infrequently updated or static content helps a lot!

Content Delivery Networks (CDNs)

Lancaster University





Infrastructure as a Service (IaaS)

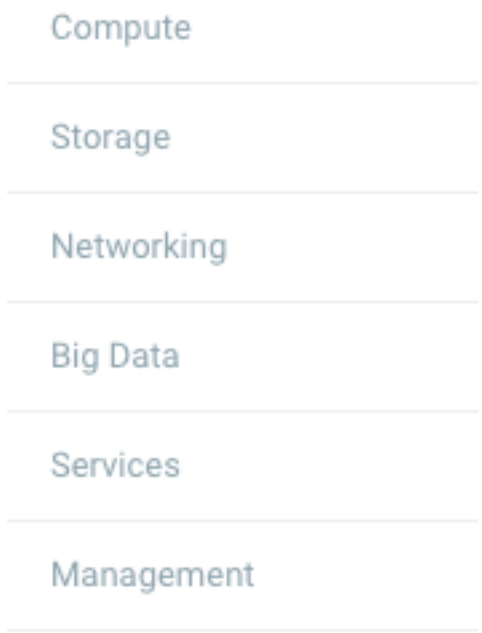
Lancaster University

- It's hardware, only in software
  - Machines are 'virtual machine instances'
  - Can be provisioned for various roles (frontend, backend, memcached, compute intensive); anything really!
  - Can be re-provisioned (vertical scaling)
  - Can be replicated (horizontal scaling)
  - Can be created and destroyed (almost) instantly
  - Can be connected however you want
- Inherently flexible and scalable!
- These are the services that allow start-ups to scale massively, without requiring their own datacentre

Platform as a Service (PaaS)

Lancaster University

- Deploy services instead of servers (e.g. Google App/Engine)
  - Standalone and replicated data stores (SQL and NoSQL)
  - Arbitrary web applications (e.g. in Python, Java or Go)
  - Store data (with backup)
  - Compute
  - Content Delivery Network!



Advantages of PaaS

Lancaster University

- Resilience to hardware and network failures (99.999% reliability)
- A run-time proportion of CPU, memory, storage, network IO (specifiable configurability)
- On-demand scalability (‘elasticity’)
- Lower cost – and pay per use…
- Global availability

Disadvantages of PaaS

Lancaster University

- Latency is no longer guaranteed; my machines could be anywhere around the globe!
- Interoperability issues (moving cloud providers?)
- Hidden (on) costs (hidden supplier charges or even suppliers!) – transnational borders (e.g. US)
- Data recovery (can they?)

Lecture Summary

- Performance bottlenecks (at the client)
  - Latency and where it comes from
  - How lots of files, resources and when they're loaded delays rendering
  - Tips for avoiding blocking and load delays, inc. reducing connections, exploiting caching and compression
- Performance bottlenecks (at the server)
  - Where do scaling related bottlenecks come from
  - Metrics for detecting scale problems
  - What types of scaling up can we do
  - Strategies for optimising web architectures

Further reading

- http://developer.yahoo.com/performance/rules.html
  - **35** best practices for optimising the performance of your website
- http://yslow.org/user-guide/
  - **23** performance rules (also open-source site analyser, c.f. Google's PageSpeed)
- http://bit.ly/1smkVN8
  - Google's 'critical rendering path' tutorial