Server-Side Technologies

MPRI 2.26.2: Web Data Management

Antoine Amarilli

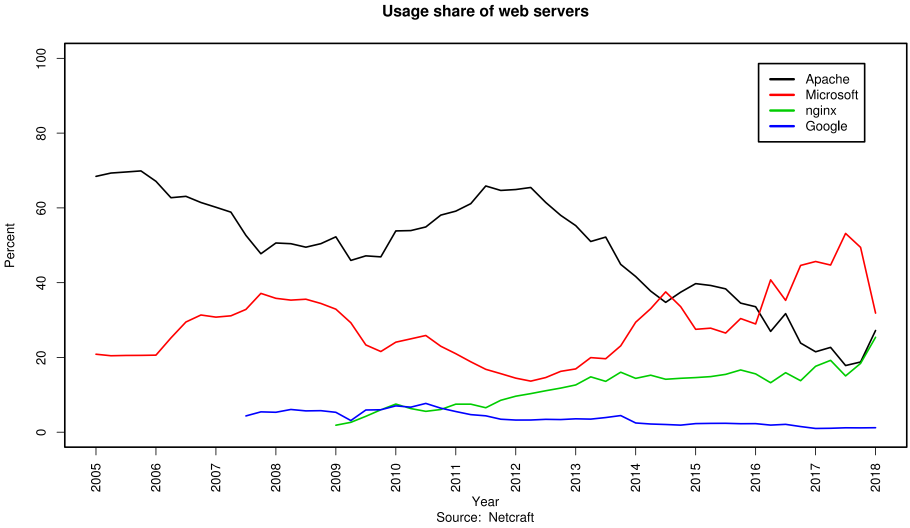
Friday, December 7th

Servers

Handle the query (simple case) or route it to another program (complex case)

Apache	Free and open-source, released in 1995
IIS	Provided with Windows, proprietary
nginx	High performance, free and open-source (but commercial Plus version), released in 2002
GWS	Google Web Server, internal
lighttpd	Lightweight alternative to Apache
Caddy	Supports HTTP/2 and HTTPS, released in 2015
Others	Rare, experimental, embedded systems…

Market share



Simple static websites

- Each resource is stored in a file on disk:
  - → `/var/www/page.html`, `/var/www/style.css`...
- Pages are organized as a hierarchy of folders
- The requested paths correspond to the folders
  - → `GET /a/b.html` corresponds to `/var/www/a/b.html`
- If a directory is queried:
  - Serve `index.html` if it exists
  - Otherwise, serve a list of the files in the folder

Apache extensions

- Parameter Apache (and other servers) with `.htaccess` files:
  - `deny from all` to block clients from accessing a directory
  - HTTP Basic authentication
- URL rewriting:
  - `RewriteRule (.*\.png) /images/$1`
- Server Side Includes:
  - `<!-#include virtual="/footer.html" ->`

Logging and statistics

Traditional Web servers log queries (NCSA common log format):

- IP address of the client
- Date and time
- First line of the HTTP query (includes the path)
- HTTP status code
- Size of the response
- Often: User-Agent and Referer

```
208.115.113.88 - - [22/Jan/2012:06:27:00 +0100]
"GET /robots.txt HTTP/1.1" 404 266 "-"
"Mozilla/5.0 (compatible; Ezooms/1.0; ezooms.bot@gmail.com)"
```

What do logs divulge

- Most visited web pages
- Number of unique visitors
- Time spent on website and paths (but complicated by tabs)
- Browser market share, most common bots
- Geographical origin of visitors (IP)
- Links to the website, search engine terms

Modern logging

- The historical way is to process these logs directly
- Also: PHP scripts that will do their own logging (e.g., Matomo)
- Also: Javascript analytics, e.g., Google Analytics
- Also: third-party info, e.g., Google Search Console

Table of Contents

CGI

- Historical way: the Web server calls an external program
- The program is given the parameters of the query
- The query result is what the program returns
- Drawback: it's heavy to create one process per query:
  - → FastCGI and other such mechanisms, or
  - → Integrate the programming language to the server, e.g., PHP

PHP

- Released in 1995 and used by **hundreds of millions** of websites[1]
- From **dirty hacks** (Personal Home Page) to a **full** language
- **Added** to HTML pages and run by the server

```
<ul>
<?php
  $from = intval($_POST['from']);
  $to = intval($_POST['to']);
  for ($i = $from; $i < $to; $i++) {
    echo "<li>$i</li>";
  }
?>
</ul>
```

[1] https://secure.php.net/usage.php

Drawbacks of PHP

- Not initially designed as a complete programming language
- Historically encouraged bad security practices
  - → e.g., making `$_POST['from']` available as `$from`
- Interpreted language so bad performance
  - → Now, virtual machines with JIT compilation (HHVM by Facebook)

Other server-side languages

ASP.NET	Microsoft
ColdFusion	Adobe, commercial and proprietary
JSP	Integrating Java and a Web server (e.g., Apache Tomcat)
node.js	Chrome’s JavaScript engine (V8) plus a Web server
Python	Web frameworks: Django, CherryPy, Flask
Ruby	Web frameworks: Ruby on Rails, Sinatra

Databases

- SQL databases: MySQL, PostgreSQL, or proprietary solutions
- NoSQL databases (e.g., MongoDB) for documents, graphs, key-value pairs, triples, etc.
- Distributed databases

$\rightarrow$ See Pierre's class

Table of Contents

Frameworks

- Set of functions and tools, organized around a language, for Web applications
- AJAX integration, Javascript code generation…
- MVC:

Model	The structure of application data and how to manipulate it
View	The presentation of data seen on the client
Controller	The control of the interaction between the model and the view

Templates again

- **Templates** for HTML pages with instantiable **fields**
- **Example** (with Jinja2), note that it is **procedural** (`for`):

```
<h1>Results for "{{ query }}"</h1>
<ul>
  {% for object in results %}
    <li>
      <a href="details/{{ object.id }}">
        {{ object.name }}
      </a>
    </li>
  {% endfor %}
</ul>
```

URL routing

- Routing depending on the path and method
- Example (with Flask):

```
@app.route('/')
def index():
  pass # Prepare the index page

@app.route('/message/<int:message_id>')
def message(message_id):
  pass # Prepare the display of message <message_id>

@app.route('/upload', methods=['POST'])
def upload():
  pass # Manage an upload
```

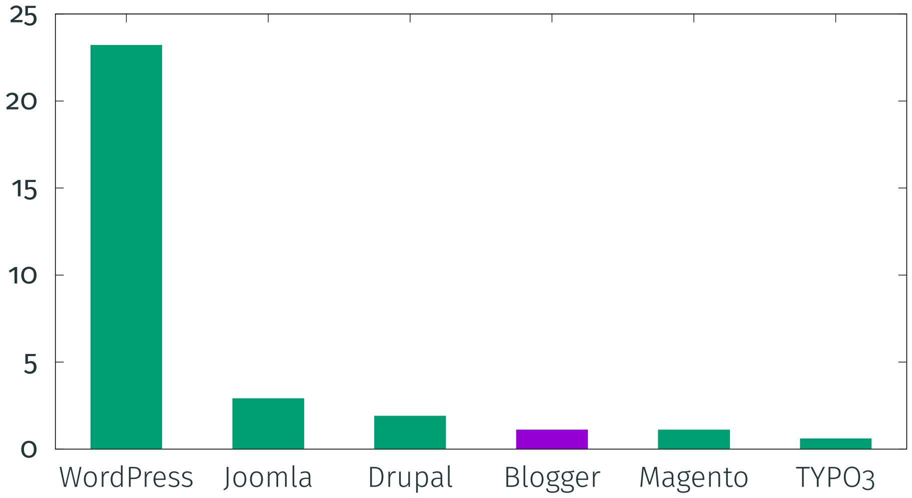
CMS

- Content Management System
- Allows users to design websites without programming:
    - Edit pages with a rich text editor or shorthand (Markdown, etc.)
    - File hosting
    - User management
    - Predefined themes
- Different kinds:

Wikis	MediaWiki, MoinMoin, PmWiki…
Forums	phpBB, PunBB, Phorum, vBulletin…
Blogs	WordPress, Movable Type, Drupal, Blogger…
QA	(like StackOverflow): Shapado, OSQA, AskBot
Shops	Magento, PrestaShop…

- Other hosted services: Weebly, Wix, etc.

Market share



Websites with each CMS (November 2018); all in PHP or hosted.

Source: https://w3techs.com/technologies/overview/content_management/all

node.js

- The MEAN platform:
  - MongoDB (see Pierre's class)
  - Express.js (minimal framework for Node.js)
  - Angular
  - Node.js
- Advantage: Isomorphic JS: same code on the client and server:
  - Idea: first run the code on the server to compute the view
  - Then: send the complete Javascript code in background and run it on the client
  - Other advantages for code reuse (e.g., validating input)
- Package manager for libraries: `npm` (or `yarn`); also `bower`

Meteor

- Integrated solution on top of node.js
- Database everywhere: Have a partial cache of the database on the client which is transparently synchronized with the server
  - $\rightarrow$ More efficient and simpler to code
  - $\rightarrow$ Access rules to limit what the client can edit
- Latency compensation: optimistically perform changes on the client and then sync them with the server in the background
- Session and user management
- Various routing libraries: FlowRouter, IronRouter, …

Javascript tooling

- Javascript minification and obfuscation
  - $\rightarrow$ Make it reversible for debugging with source maps
- Linting
- Documentation: JSDoc, like Doxygen
- Packing together code and dependencies: webpack
- Elimination of dead code or tree shaking
- Transpiling to other Javascript variants, e.g., Babel, Google Closure Tools
- Task running system: grunt, gulp

Identifying server technologies

Client technologies are easy to identify (up to JS minification and obfuscation), but server technologies are harder to spot

- `whois`: database which (often) has information about the owner of a domain name
- Geolocation of servers based on their IPs
- `traceroute` to know the **network path** to a host
- Scanning tools (`nmap`) to find machines and fingerprint their OS
- `Server` header (may be missing or wrong)
- Format for cookies, session identifiers...
- Paths and **extensions**: `.php`, `.asp`, ...
- **Comments** in HTML code

Table of Contents

How to host your website

- You can use a **hosted solution**, e.g., Wordpress, Weebly (simplest)
- ISPs often propose hosting with **limited space**, sometimes PHP/MySQL
- You can rent a **VPS** or **dedicated server** to run your own (a few EUR/month)
- You can host a website **in your home** behind an optic fiber connection with a fixed IP address
- You can host a web application **on the cloud** (serverless computing)

Hosting infrastructure

- **Server**: machine which is always on and answers Web queries
- **Datacenter**: building containing servers with a good Internet connection, reliable electricity, air conditioning, physical security
- **VPS:** Virtual Private Server: a virtual machine that pretends to be a true machine
- **Cloud**: easy way to rent machines at a large scale
  - $\rightarrow$ Can adapt the number of rented machines depending on load
- **CDN:** Content delivery network, acts as a proxy

How to self-host your website

- Rent a domain name (around 15 EUR/year)
- Have a server (your own machine, or a VPS or dedicated server)
- Configure SSH to connect to the machine and set it up
- Install a Web server, your framework, your CMS, etc.

Concrete example: Wikimedia

- `wikipedia.org`, 5th most visited Website in 2018.[2]
- Not-for-profit charity, around 300 employees in 2018
- 81.9 million USD in revenue in 2016 (mostly donations)
- Technical costs in 2016: a few million USD
- About one thousand servers in total (2013)

[2] `http://www.alexa.com/topsites`

General statistics

- 137 115 active users on `en.wikipedia.org`[3]
- Around 1000 edits per minute in total[4]
- 16 billion page views per month on all projects[5]
- Around 7 000 per second on average with peaks at 50 000[6]
- 825 million unique devices per month on the English Wikipedia[7]

[3] https://en.wikipedia.org/wiki/Wikipedia:Wikipedians#Number_of_editors

[4] https://grafana.wikimedia.org/dashboard/

[5] https://stats.wikimedia.org/v2/

[6] https://arstechnica.com/information-technology/2008/10/wikipedia-adopts-ubuntu-for-its-server-infrastructure/

[7] https://stats.wikimedia.org/v2/

General infrastructure

- Datacenters:
  - → Main site: Ashburn, Virginia (Equinix)
  - → For Europe (network and cache), Amsterdam (EvoSwitch, Kennisnet)
  - → Cache: San Francisco (United Layer), Singapore (Equinix)
  - → Other sites: Dallas, Chicago
  - → Backup datacenter: Carrollton, Texas (CyrusOne)
- Dell servers running Ubuntu[8] and Debian
- `puppet` to manage the server configuration
- Monitoring software: Icinga, Grafana
  - → `grafana.wikimedia.org`
  - → `status.wikimedia.org`

[8] `https://insights.ubuntu.com/2010/10/04/wikimedia-chooses-ubuntu-for-all-of-its-servers/`

Main tasks (as of 2013)

- Wiki management software: MediaWiki, in PHP
- Apache server, using HHVM[9]
  - → 192 machines (in Ashburn)
- Database: MariaDB
  - → 54 database machines
  - → 10 storage machines with 12 hard drives of 2 TB in RAID10
- Distributed file system: Ceph (previously Swift)
  - → 12 servers
- Asynchronous task servers (NoSQL database Redis)
  - → 16 servers

[9] https://blog.wikimedia.org/2014/12/29/how-we-made-editing-wikipedia-twice-as-fast/

Caches (as of 2013)

- Squid: 40 machines
  - $\rightarrow$ 8 machines for multimedia
  - $\rightarrow$ 32 machines for text
- Varnish
  - $\rightarrow$ 8 machines
- Cache invalidation with MediaWiki
- Memcached between MediaWiki and the database
  - $\rightarrow$ 16 machines

$\rightarrow$ 90% of traffic only uses the cache and not Apache.[10]

[10] `https://blog.wikimedia.org/2013/01/19/wikimedia-sites-move-to-primary-data-center-in-ashburn-virginia/`

Other services (as of 2013)

- SSL termination proxies with nginx:
  - $\rightarrow$ 9 machines
- Load balancing with LVS (Linux Virtual Server):
  - $\rightarrow$ 6 machines
- Indexation for the search function:

Lucene	25 machines
Solr	3 machines

- Media file resizing:

Images	8 machines
Videos	2 machines

- Statistics: 27 machines
- Online payment processing: 4 machines
- DNS servers, snapshots, various services, etc.

Example of a rack (2015)

Image credits: `https://meta.wikimedia.org/wiki/File:Wikimedia_Foundation_Servers_2015-90.jpg`

Crédits

- Matériel de cours inspiré de notes par Pierre Senellart
- Merci à Pierre Senellart pour sa relecture
- Transparent 3: chiffres Netcraft `https://news.netcraft.com/archives/category/web-server-survey`, graphe arichnad, licence CC-BY-SA, cf `https://commons.wikimedia.org/wiki/File:Usage_share_of_web_servers_(Source_Netcraft).svg`