Web Servers

Markus Richter

Institute for Software Engineering and Programming Languages

December 19, 2013

Table of Contents

It’s not a bug!

The terms Web and Internet are proper nouns and thus are capitalized!

Introduction

- The Internet is growing
- Not for information gain only
- Services for business, communication, entertainment
- Key to the success of the Internet

$\Longrightarrow$ Made possible by Web servers

Introduction

As Web servers play an important role

1. How to support development of Web sites towards more attractive and modern Web sites?
2. How to assure high performance in the Web?

Definitions - Web server

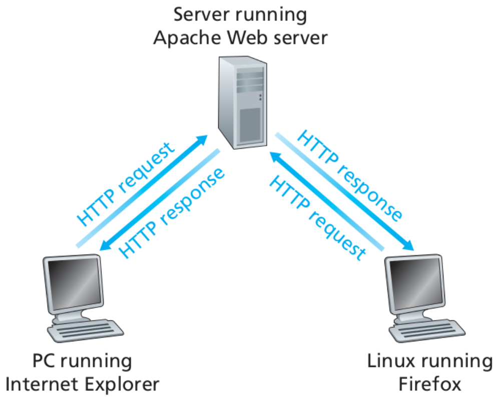


Figure 1 : Client Server Architecture. Taken from [Kurose and Ross, 2012]

Definitions - HTTP

- HyperText Transfer Protocol
- Application-layer protocol
- Server Client Architecture
- Communication by using HTTP messages

Concept

- Mainly based on HTTP
- Communication by using HTTP messages
- Static behaviour

Files addressed by URL:

- HTML files
- jpg, png, pdf etc.
- . . .

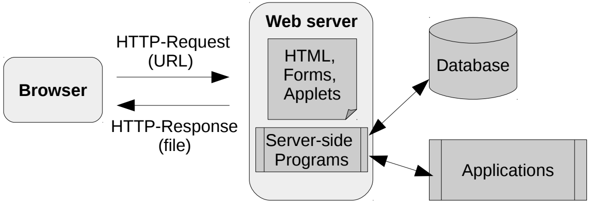


Figure 2 : Concept of an HTTP Web server

Example HTTP - Request

Example

```
GET /somedir/page.html HTTP/1.1
Host: www.someschool.edu
Connection: close
User-agent: Mozilla/5.0
Accept-language: fr
```

Listing 1 : Simple HTTP request message. Taken from [Kurose and Ross, 2012]

Example HTTP - Response

Example

```
HTTP/1.1 200 OK
Connection: close
Date: Tue, 09 Aug 2011 15:44:04 GMT
Server: Apache/2.2.3 (CentOS)
Last-Modified: Tue, 09 Aug 2011 15:11:03 GMT
Content-Length: 6821
Content-Type: text/html

 (data data data data data ...)
```

Listing 2 : Simple HTTP response message. Taken from [Kurose and Ross, 2012]

History

- Closely tied to the history of the Internet
- Web servers still rely on HTTP and HTML

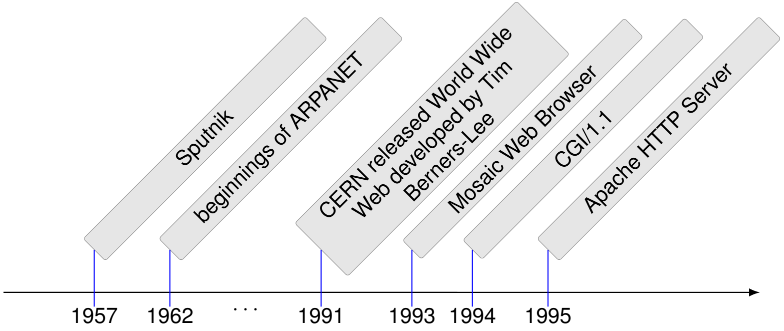
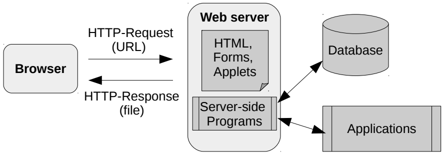


Figure 3 : Chronology of the Web

Server-side Technologies

- Need for interactivity grew
- Number of competing technologies evolved
- Used as external programs or modules
- Examples: CGI, ASP.NET, PHP, JSP



CGI - Common Gateway Interface

Definition [Coar, 1998]

A simple interface for running external programs, software or gateways under an information server in a platform-independent manner

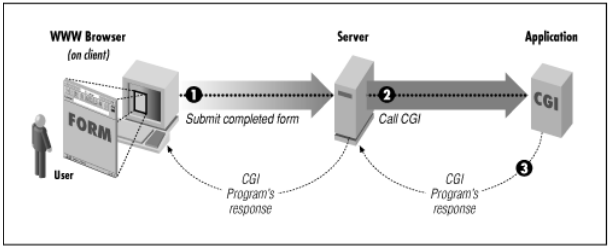


Figure 4 : How CGI works. Taken from [Gundavaram, 1996]

CGI - Common Gateway Interface

Example

```
GET /cgi-bin/welcome.pl HTTP/1.0
Accept: www/source
Accept: text/html
Accept: image/gif
User-Agent: Lynx/2.4 libwww/2.14
From: shishir@bu.edu
```

Listing 3 : Client request for CGI program. Taken from [Gundavaram, 1996]

CGI - Common Gateway Interface

Result of execution can be

- a new document
- an URL to an existing one

Advantages:

- CGI is platform-independent

Significant drawbacks:

- Low scalability
- Bad performance

$\Longrightarrow$ Most modern Web servers provide their own solutions for popular technologies, e.g. *mod_php*, *mod_perl* for Apache.

PHP:Hypertext Preprocessor

Definition [PHP.net, 2013]

A widely-used Open Source general-purpose scripting language that is especially suited for Web development and can be embedded into HTML. Its syntax draws upon C, Java, and Perl, and is easy to learn.

- According to [W3Techs, 2013] by far the most popular used technology in Web development
- Needs to be interpreted
- Interpreter can be a module or a CGI binary
- Embedded inside HTML and executed every time the HTML file is accessed
- Usually used together with Linux, Apache Web server, MySQL (LAMP architecture)

PHP:Hypertext Preprocessor

Example

```
<?php
echo "<p>Order processed at ";
echo date('H:i, jS F Y');
echo "</p>";
?>
```

Listing 4 : PHP embedded in HTML. Taken from [Welling and Thomson, 2008]

PHP:Hypertext Preprocessor

Advantages of PHP:

- High performance
- High scalability
- Object oriented support
- Database integration
- Low costs

Disadvantages:

- Code maintenance
- Not fully object oriented
- Problems with stability and interdependencies

JSP - JavaServer Pages

- Similar to PHP, but easier to achieve more structure
- Uses Java
- Meant to be used in a MVC design fashion
- All components are wrapped inside a container
- Container manages communication between JSP technology and Web server
- A popular container today is e. g. Tomcat

JSP - JavaServer Pages

- JavaBeans encapsulates the data and methods to work on it
- Servlet gets the requests and sends back responses to the Web server
- JSPs are responsible for the view

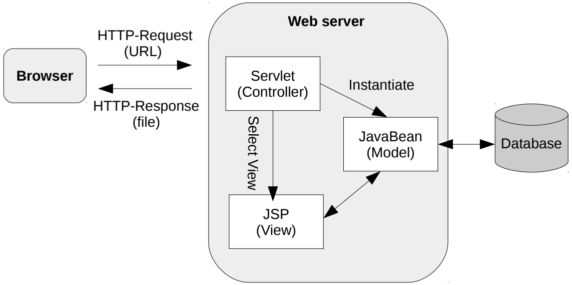


Figure 5 : Model of JSP

JSP - JavaServer Pages

Similarly to PHP, JSPs are HTML files with embedded Java code

But

Usually the code is meant as directives to display data. It should not contain logic.

JSP - JavaServer Pages

Example

```
Hobby:
<input type="text" name="hobby"
value="${refData.hobby}">
<br>
Aversion:
<input type="text" name="aversion"
value="${refData.aversion}">
```

Listing 5 : Java embedded in HTML. Taken from [Downey, 2008]

Advantages of JSP:

- Encourages more structure
- Supports Java Code

Disadvantages:

- Difficult to trace errors
- Not as good performance as PHP initially: JSPs need to be compiled

Increasing Performance

- Number of Web users grows rapidly
- Expanding Web infrastructure is expensive
- Today's Web servers use fair scheduling

$\Longrightarrow$ Possible solution: Size-based unfair scheduling [Harchol-Balter et al., 2003] and [Biersack et al., 2007]

Size-based scheduling

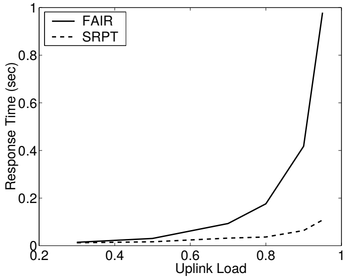
- Fair scheduling: Web server partitions its resources fairly among requests ready to receive service
- Size-based unfair scheduling: Prioritise short requests or those with short remaining file size
- Claim: Expected response time of every HTTP request can be reduced and minimise number of connections[Harchol-Balter et al., 2003] and [Biersack et al., 2007]
- Apache Web server an Linux were used for measurements
- Implementations had to be done at kernel level

SRPT - Shortest Remaining Processing Time first

- Size-based scheduling goes along with SRPT

SRPT

Preemptive Shortest Remaining Process Time first algorithm



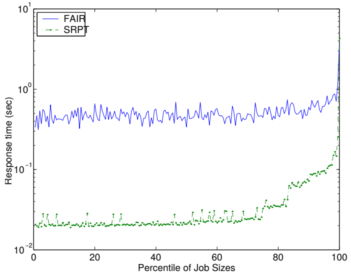


Figure 6 : (Left) Mean response time for static requests. (Right) Response time as a function of the size of the requested file, system load is fixed at $\rho = 0.8$. Taken from [Biersack et al., 2007]

SRPT - Shortest Remaining Processing Time first

Reservations against SRPT: Fear that big jobs will starve. But [Biersack et al., 2007] shows that

- Web file sizes exhibit highly variable statistical distributions with heavy tails
- Little if any unfairness to large requests

SRPT - Shortest Remaining Processing Time first

- Promising approach for HTTP requests
- SRPT needs to know length of transaction before execution
- Only suitable for static files

How to increase performance for dynamic content?

Increasing Performance

- LAS - Least-Attained-Service guesses the remaining service time
- LAS converges towards SRPT behaviour
- Bottleneck in processing dynamic web requests: Database backend
- Existing database management systems do not support effective transaction prioritisation for web based transaction workloads
- Lock-bound and thus need lock scheduling

PAbort and NPrionher

[McWherter et al., 2004] analyses following algorithms:

PAbort - Preemptive Abort

- Blocking low-priority transactions gets immediately preempted
- Causes overhead due to rolling back and restarting

NPrionher

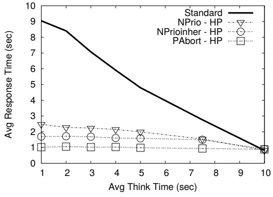
- Grands blocking low-priority transactions temporarily high priority
- Causes worse high priority performance

POW

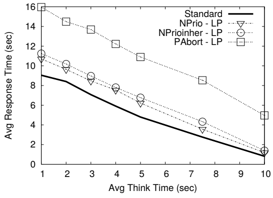
POW - Preempt-On-Wait

- Preempts low-priority transactions in favour of high-priority ones
- But if and only if the low-priority transaction currently, or in the future, has to wait for lock
- Guarantees that already work done will not be lost
- Compromise between PAbort and NPrionher

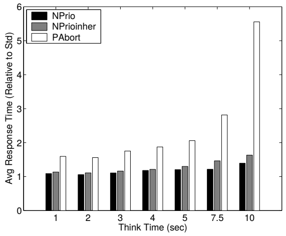
POW, PAbort and NPrionher



(a) High-Priority



(b) Low-Priority



(c) Overhead

Figure 7 : Average response time for high- and low-priority transactions for *POW*, *PAbort*, and *NPrioinher* as a function of load (a) and (b). Aggregate high- and low-priority average response time relative to *Standard* (c). Taken from [McWherter et al., 2004]

Increasing Performance

- SRPT for static content
- POW for dynamic processing
- Combination of both forms appealing solution for increasing performance

Conclusion

Initial Questions:

1. How to support development of Web sites towards more attractive and modern Web sites?
2. How to assure high performance in the Web?

Any questions?

What server-side technologies did you encounter?

Biersack, E. W., Schroeder, B., and Urvoy-Keller, G. (2007).
Scheduling in practice.
SIGMETRICS Perform. Eval. Rev., 34(4):21–28.

Coar, K. A. L. (1998).
The WWW Common Gateway Interface Version 1.2.
http://ken.coar.org/cgi/cgi-120-00a.html.
[Online; Date of last access: 21.11.2013].

Downey, T. (2008).
Web Development with Java: Using Hibernate, JSPs and Servlets.
Springer.

Gundavaram, S. (1996).
CGI Programming on the World Wide Web.
Nutshell Handbook. O'Reilly Vlg. GmbH & Company.

Harchol-Balter, M., Schroeder, B., Bansal, N., and Agrawal, M. (2003).
Size-based scheduling to improve web performance.
ACM Trans. Comput. Syst., 21(2):207–233.

Kurose, J. and Ross, K. (2012).
Computer Networking: A Top-Down Approach.

Always learning. Pearson Education, Limited.

McWherter, D. T., Schroeder, B., Ailamaki, A., and Harchol-Balter, M. (2004).
Priority mechanisms for oltp and transactional web applications.
In *Proceedings of the 20th International Conference on Data Engineering*, ICDE '04, pages 535–, Washington, DC, USA. IEEE Computer Society.

PHP.net (2013).
`http://www.php.net/manual/en/preface.php`.
[Online; Date of last access: 21.11.2013].

W3Techs (2013).
W3Techs - World Wide Web Technology Surveys.
`http://w3techs.com`.
[Online; Date of last access: 21.11.2013].

Welling, L. and Thomson, L. (2008).
PHP and MySQL Web Development.
Developer's Library. Pearson Education.