LARGE SCALE INTERNET SERVICES

2110414 Large Scale Computing Systems
Natawut Nupairoj, Ph.D.

Outline

- Overview
- Background Knowledge
- Architectural Case Studies
- Real-World Case Study

Overview

Overview

- Internet services become very essential and popular
  - Google serves hundreds of millions of search requests per day
- Main requirements
  - Availability
  - Scalability

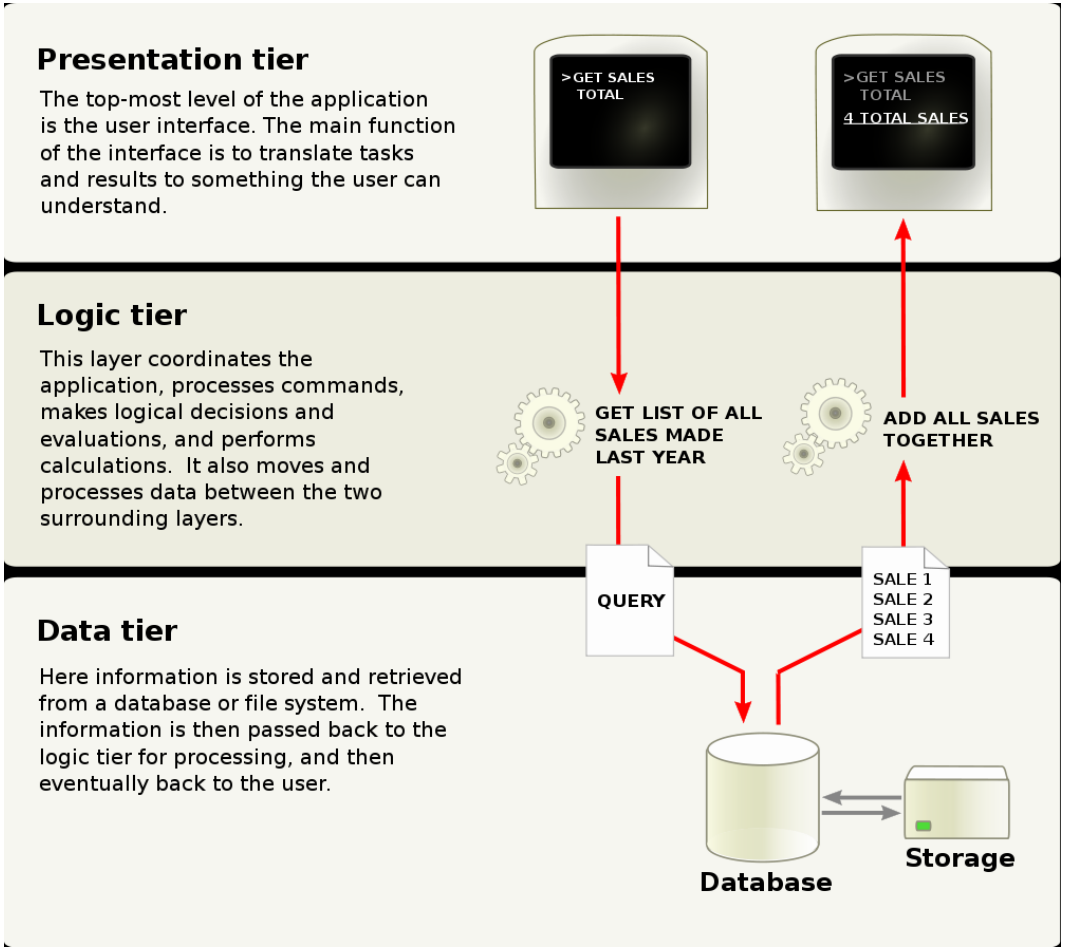
Internet Service Application Characteristics

Table 1. Comparison of Internet service application characteristics with those of traditional applications.

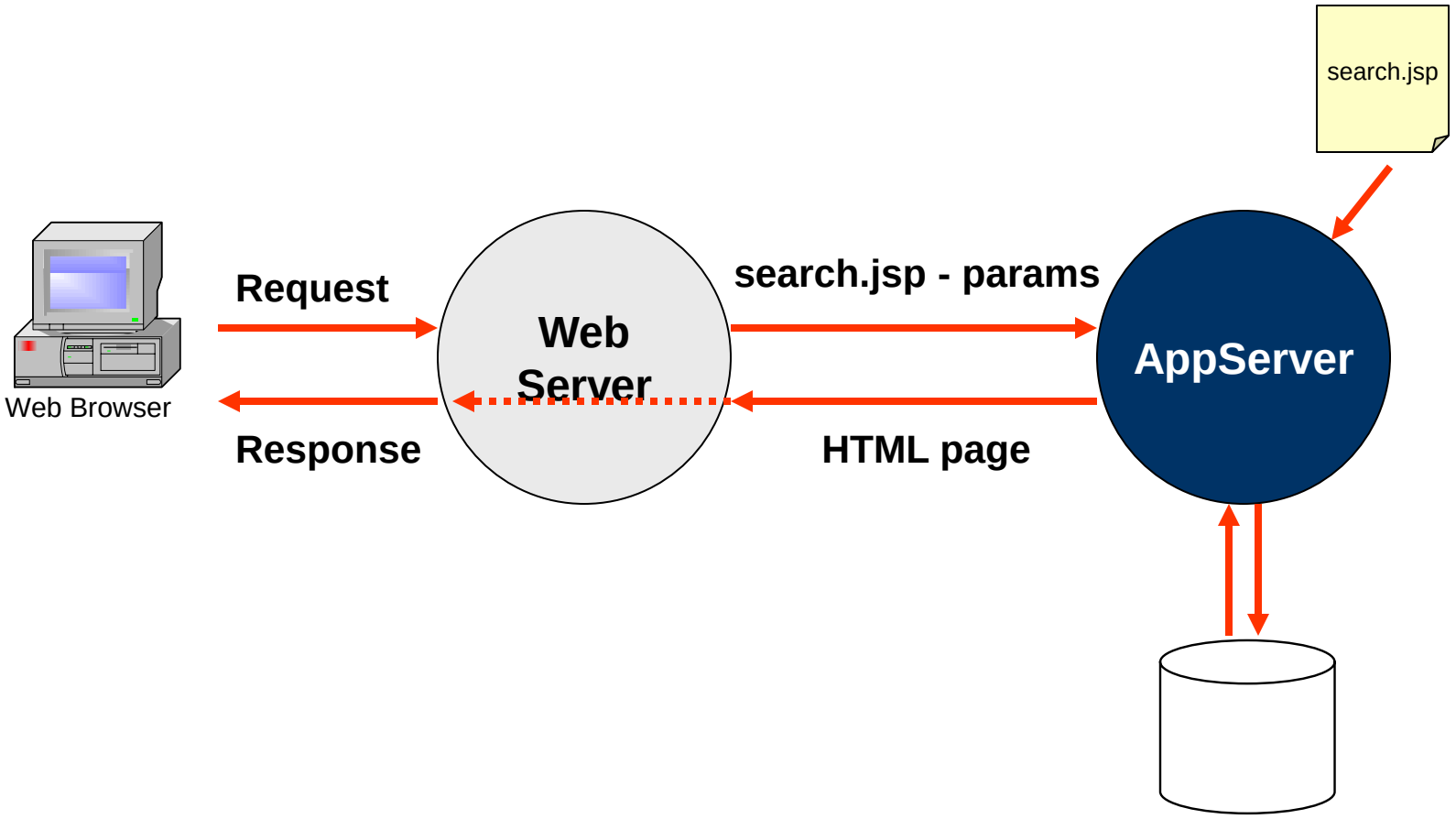
Characteristic	Traditional desktop applications	Traditional high-availability applications	Large-scale Internet service applications
Dominant application	Productivity applications, games	Database, enterprise messaging	E-mail, search, news, e-commerce, data storage
Typical hardware platform	Desktop PC	Fault-tolerant server or failover cluster	Clusters of hundreds to thousands of cheap PCs, often geographically distributed
Software model	Off-the-shelf, standalone	Off-the-shelf, multitier	Customized, multitier
Software release frequency	Months	Months	Days or weeks
Networking environment	None (standalone)	Enterprise-scale	Within cluster, on the Internet between data centers, and to and from user clients
Operator	end user	Corporate IT staff	Service operations staff, data center/collocation site staff
Typical physical environment	Home or office	Corporate machine room	Collocation facility
Key metrics	Functionality, interactive latency	Availability, throughput	Availability, functionality, scalability, manageability, throughput

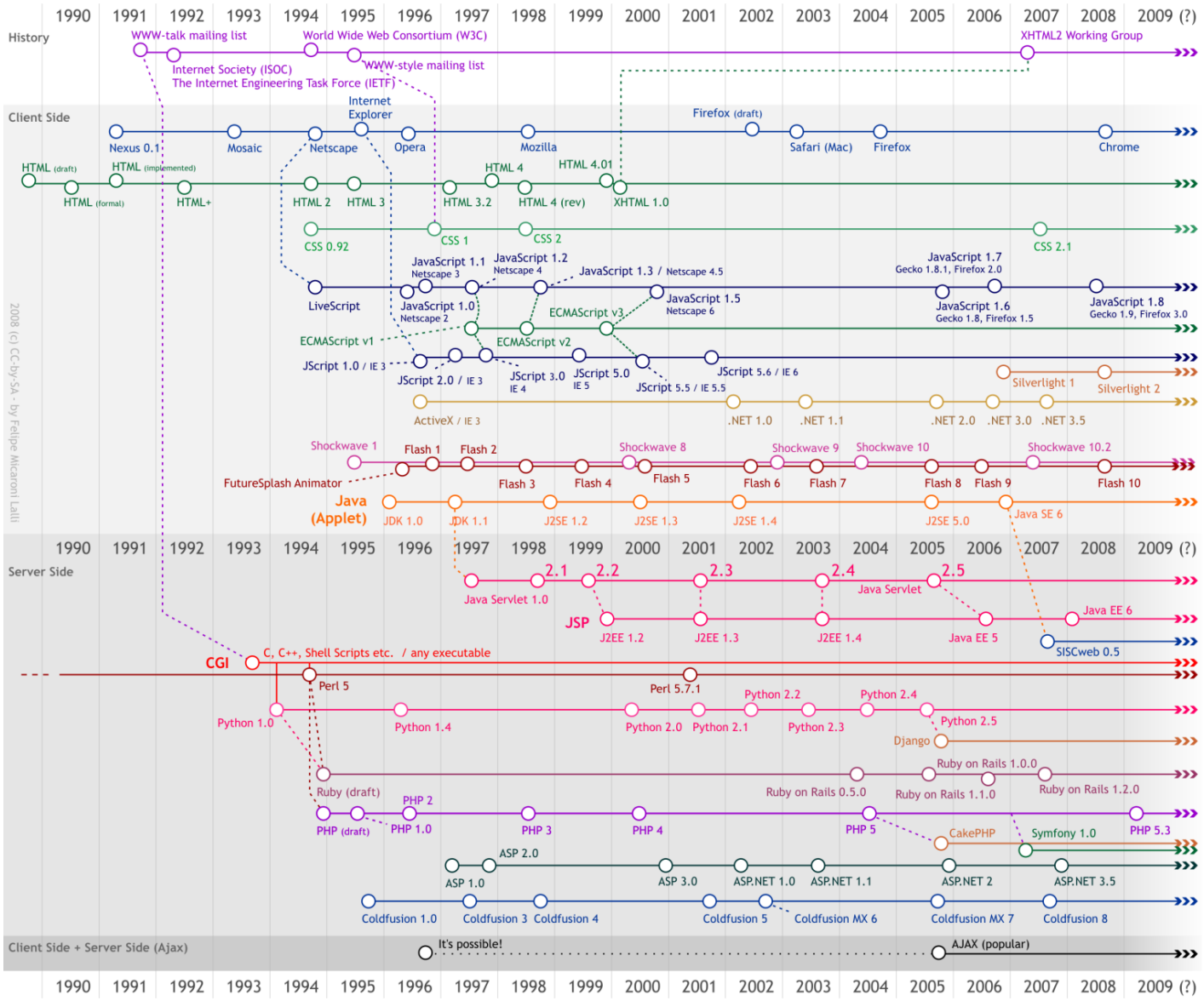
Background Knowledge

Multi-Tier Architecture



Web Based Architecture Revisited

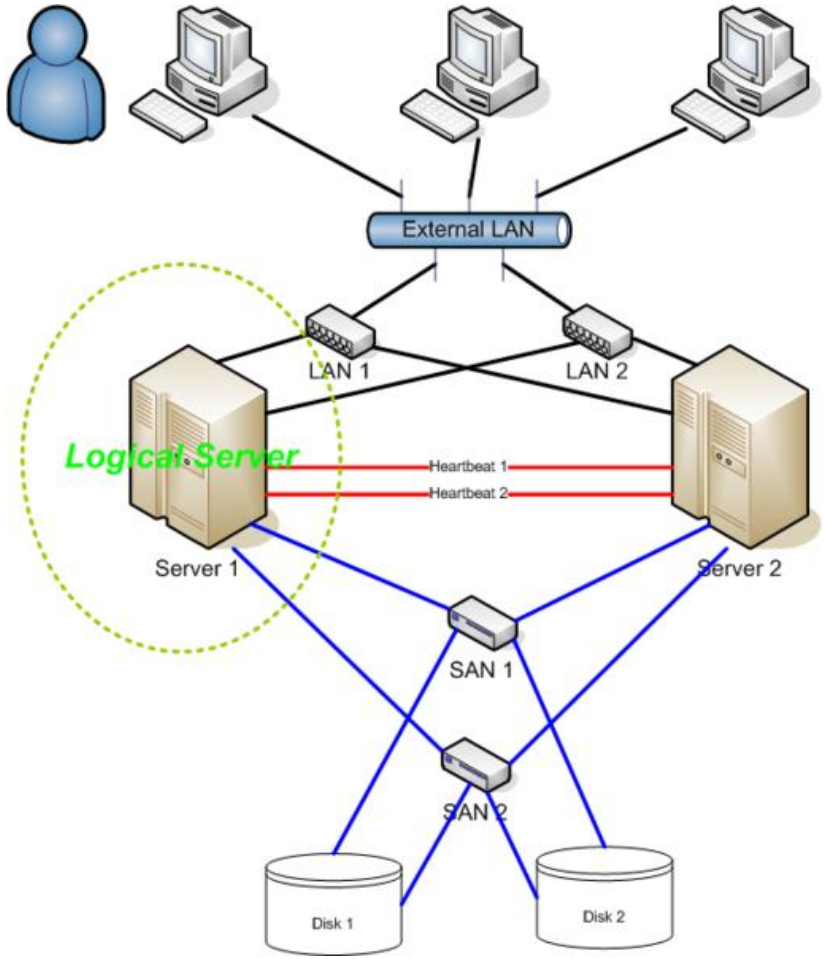



1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 (?)
History
WWW-talk mailing list
World Wide Web Consortium (W3C)
XHTML2 Working Group
Internet Society (ISOC)
The Internet Engineering Task Force (IETF)
WWW-style mailing list
Client Side
Internet Explorer
Firefox (draft)
Nexus 0.1
Mosaic
Netscape
Opera
Mozilla
Safari (Mac)
Firefox
Chrome
HTML (draft)
HTML (implemented)
HTML (formal)
HTML+
HTML 2
HTML 3
HTML 3.2
HTML 4
HTML 4 (rev)
HTML 4.01
XHTML 1.0
CSS 0.92
CSS 1
CSS 2
CSS 2.1
JavaScript 1.1 Netscape 3
JavaScript 1.2 Netscape 4
JavaScript 1.3 / Netscape 4.5
JavaScript 1.7 Gecko 1.8.1, Firefox 2.0
LiveScript
JavaScript 1.0 Netscape 2
JavaScript 1.5 Netscape 6
JavaScript 1.6 Gecko 1.8, Firefox 1.5
JavaScript 1.8 Gecko 1.9, Firefox 3.0
ECMAScript v3
ECMAScript v1
ECMAScript v2
JScript 1.0 / IE 3
JScript 2.0 / IE 3
JScript 3.0 IE 4
JScript 5.0 IE 5
JScript 5.5 / IE 5.5
JScript 5.6 / IE 6
Silverlight 1
Silverlight 2
ActiveX / IE 3
.NET 1.0
.NET 1.1
.NET 2.0
.NET 3.0
.NET 3.5
Shockwave 1
Flash 1
Flash 2
Shockwave 8
Shockwave 9
Shockwave 10
Shockwave 10.2
FutureSplash Animator
Flash 3
Flash 4
Flash 5
Flash 6
Flash 7
Flash 8
Flash 9
Flash 10
Java (Applet)
JDK 1.0
JDK 1.1
J2SE 1.2
J2SE 1.3
J2SE 1.4
J2SE 5.0
Java SE 6
2008 (c) CC-by-SA - by Felipe Micaroni Lalli
1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 (?)
Server Side
2.1
2.2
2.3
2.4
2.5
Java Servlet 1.0
Java Servlet
JSP
J2EE 1.2
J2EE 1.3
J2EE 1.4
Java EE 5
Java EE 6
SISCweb 0.5
CGI
C, C++, Shell Scripts etc. / any executable
Perl 5
Perl 5.7.1
Python 1.0
Python 1.4
Python 2.0
Python 2.1
Python 2.2
Python 2.3
Python 2.4
Python 2.5
Django
Ruby (draft)
Ruby on Rails 0.5.0
Ruby on Rails 1.0.0
Ruby on Rails 1.1.0
Ruby on Rails 1.2.0
PHP (draft)
PHP 2
PHP 1.0
PHP 3
PHP 4
PHP 5
CakePHP
Symfony 1.0
PHP 5.3
ASP 2.0
ASP 1.0
ASP 3.0
ASP.NET 1.0
ASP.NET 1.1
ASP.NET 2
ASP.NET 3.5
Coldfusion 1.0
Coldfusion 3
Coldfusion 4
Coldfusion 5
Coldfusion MX 6
Coldfusion MX 7
Coldfusion 8
Client Side + Server Side (Ajax)
It's possible!
AJAX (popular)
1990 1991 1992 1993 1994 1995 1996 1997 1998 1999 2000 2001 2002 2003 2004 2005 2006 2007 2008 2009 (?)

System Availability

- How to ensures a certain absolute degree of operational continuity during a given measurement period
- Availability includes ability of the user community to access the system, whether to submit new work, update or alter existing work, or collect the results of previous work
- Model of Availability
  - Active-Standby: HA Cluster or Failover Cluster
  - Active-Active: Server Load Balancing

HA Cluster

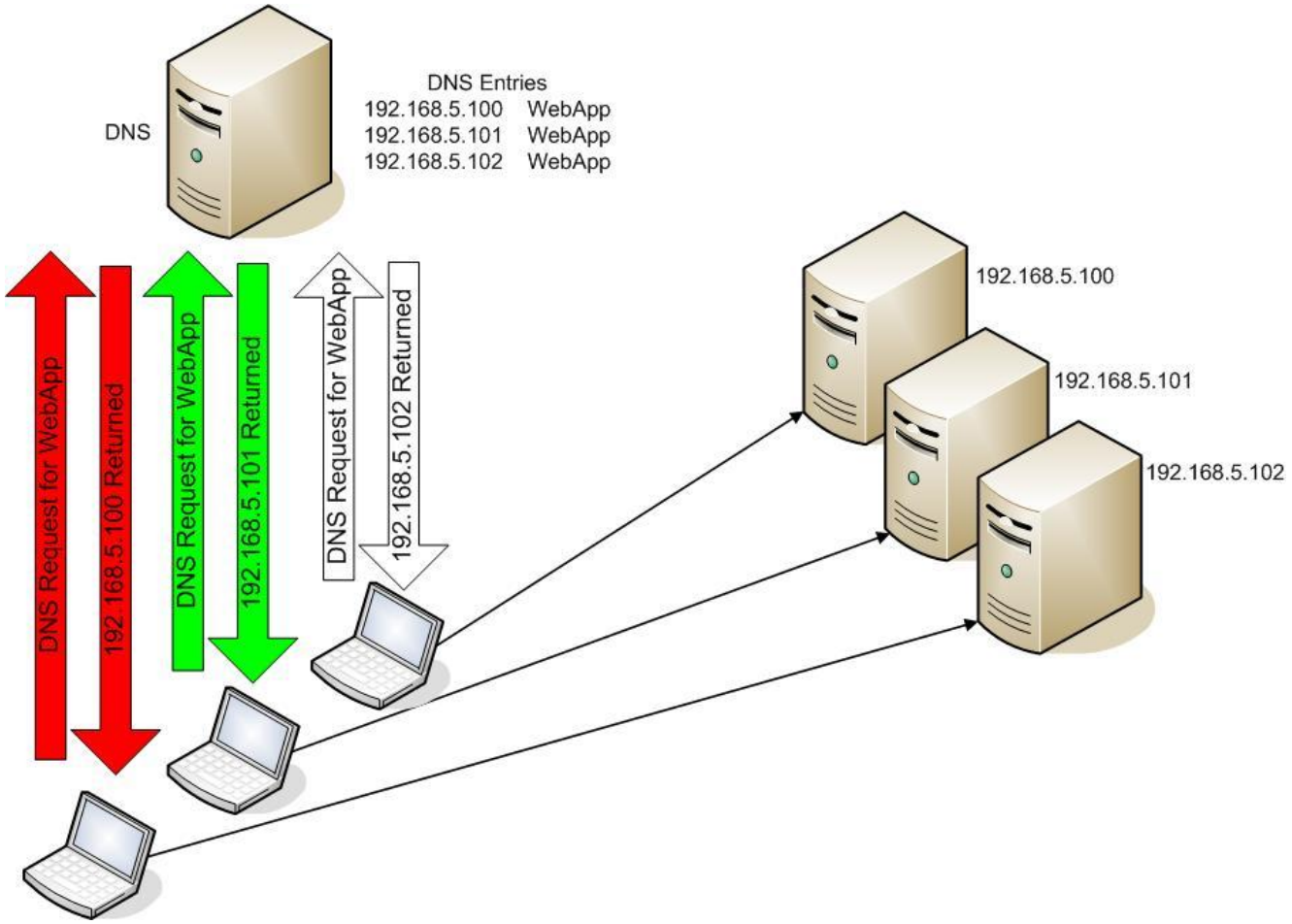


- Redundant servers and other components
  - Only one server is active (master)
  - One server is standing-by
  - Shared storages
- Pro:
  - Simple
  - Half software license costs
- Con:
  - Double hardware cost with single performance

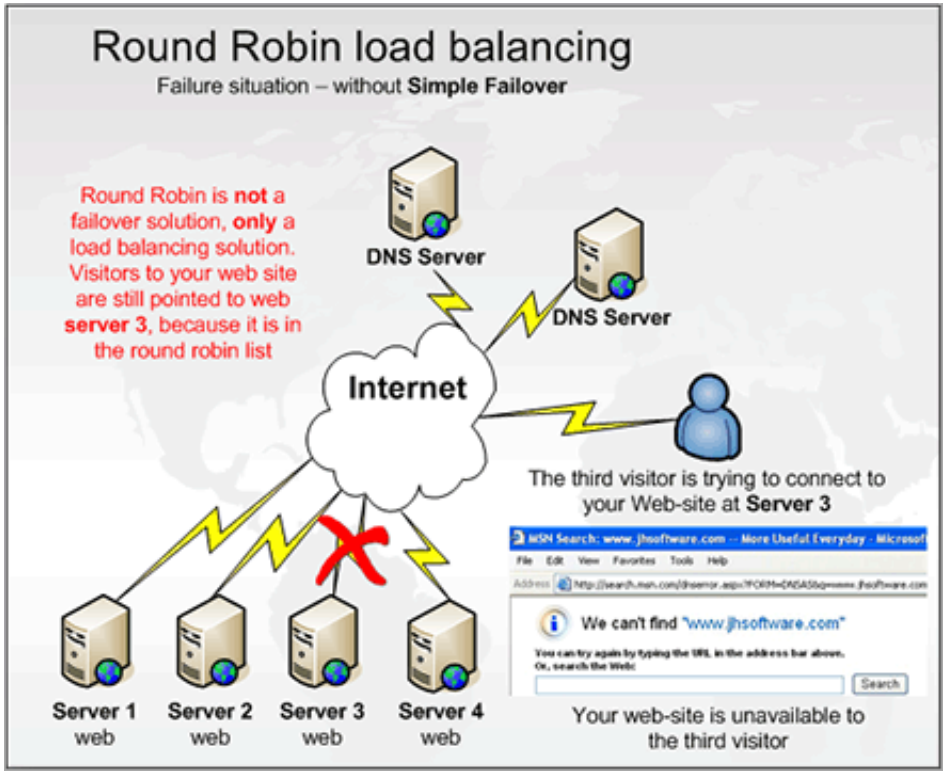
Server Load Balancing

- Spread work between two or more computers, network links, CPUs, hard drives, or other resources, in order to get optimal resource utilization, throughput, or response time
- Approaches
  - DNS Round-Robin
  - Reverse Proxy
  - Load Balancer

DNS Round-Robin

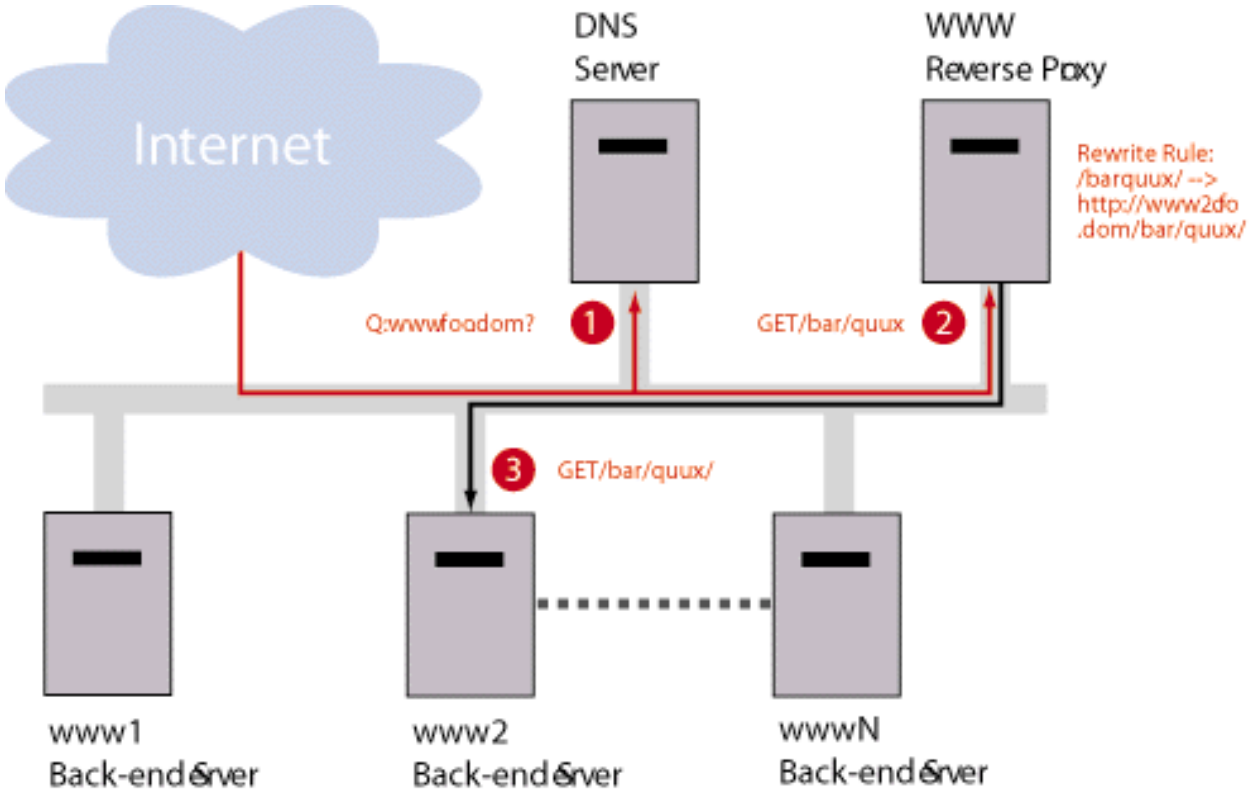


DNS Round-Robin

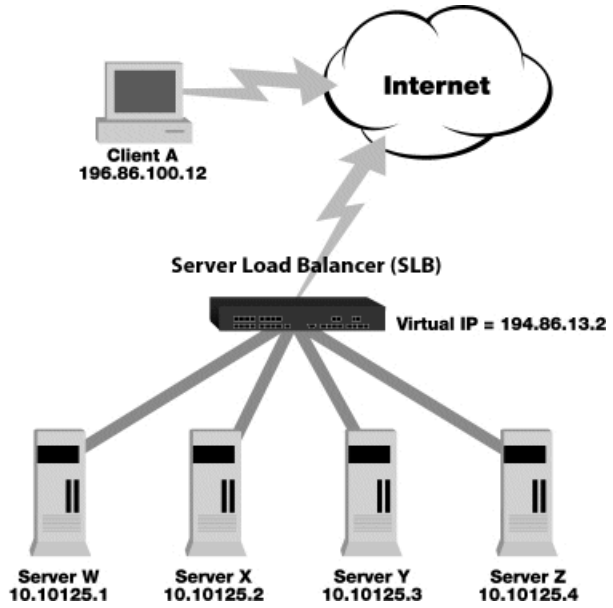


- Pro:
  - Inexpensive
- Con:
  - Load distribution, but not high availability
  - Problem with DNS caching

Reverse Proxy



Server Load Balancing

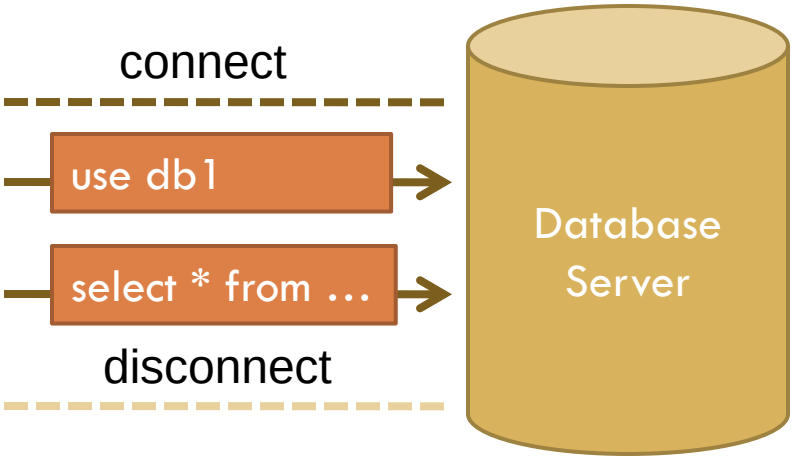


- Special equipment "load balancer" to distribute request to servers
- Clients will see only single "virtual" host based on "virtual" IP

Stateful vs. Stateless Servers

- Stateful server
  - server maintains some persistent data
  - Allow current request to relate to one of the earlier requests, “session”
- Stateless server
  - server does not keep data
  - A request is independent from earlier requests
  - Example: Web server, NFS

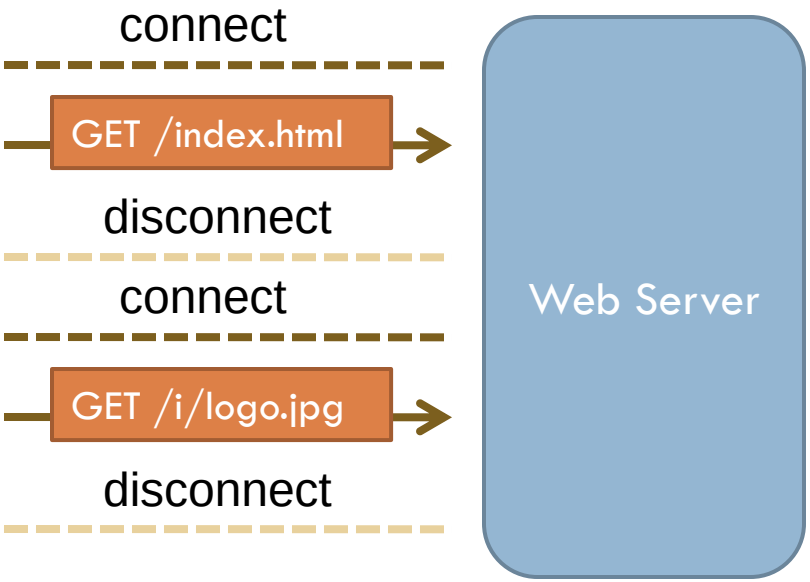
Stateful Servers



- Example: Database, FTP

- Server has to maintain some "session" information of each connection
  - Current request may depend on previous requests
- Consume server's resources (memory, TCP port, etc.)
- Lead to limit number of clients it can service
- If connection is broken, the service is interrupted

Stateless Servers



- Server does not maintain information of each connection
- Connect-request-reply-disconnect cycle
- Consume less server's resources
- Lead to large number of clients it can service

- Example: Web server, NFS

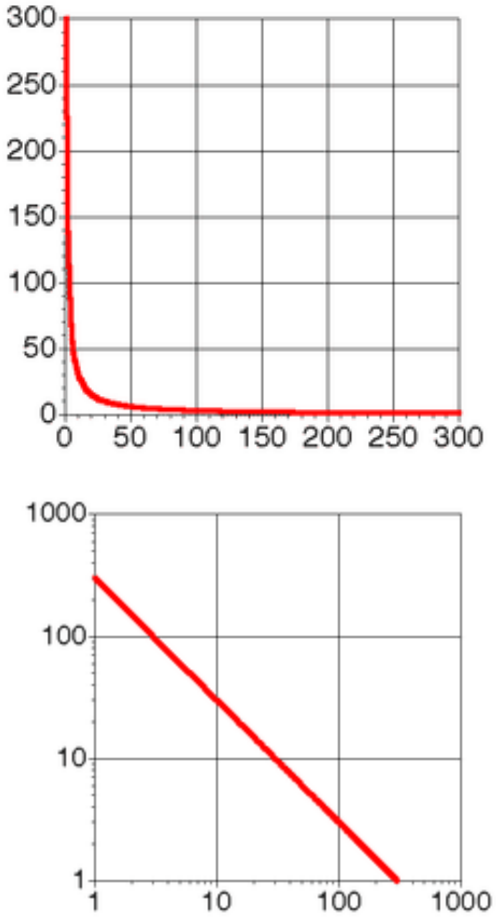
Web Caching

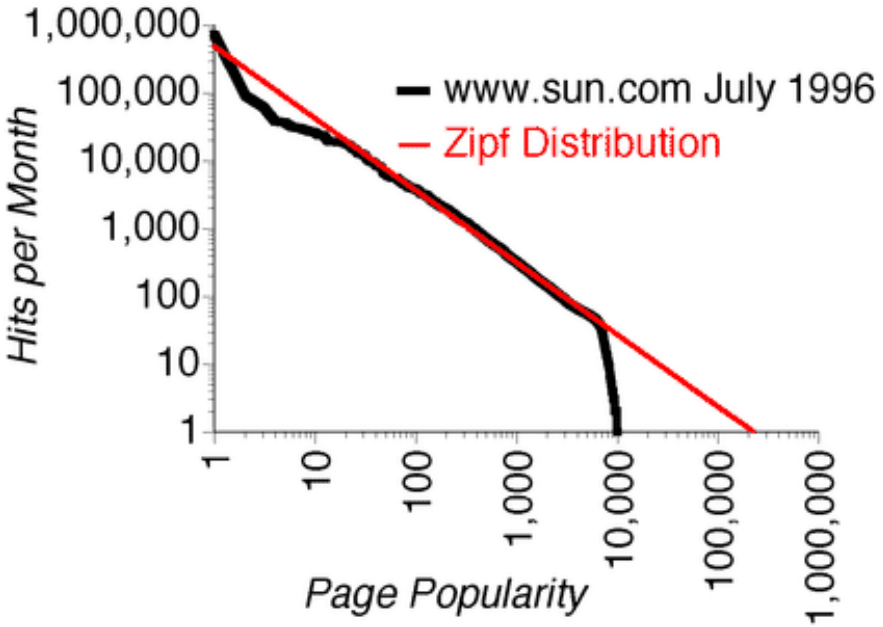
- Utilize the fact that LAN has more bandwidth and less accessing latency than WAN

```
t = accessing latency + data size / bandwidth
```

- Web pages usually have some “popularity”
  - User usually goes back-and-forth between pages
  - Users tend to share the same interest (fashion)

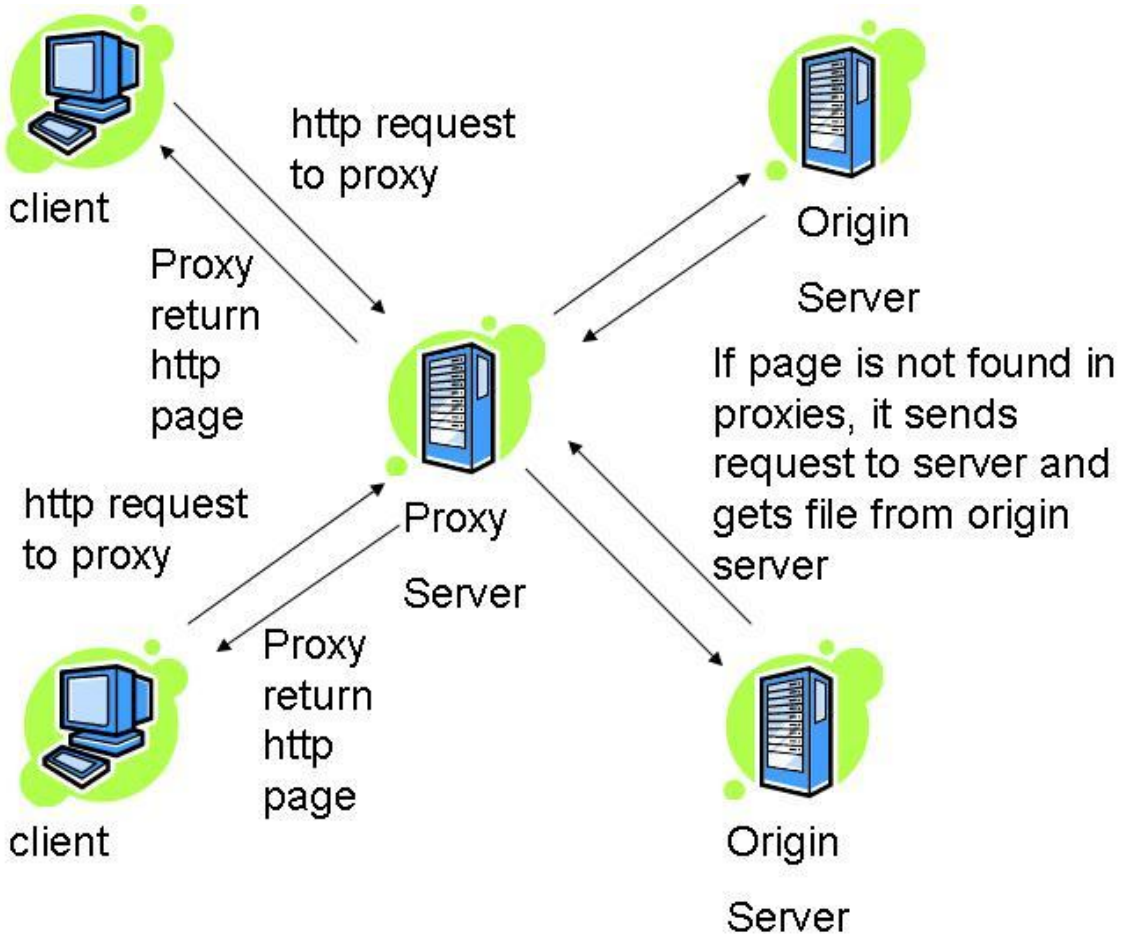
Web Page Popularity





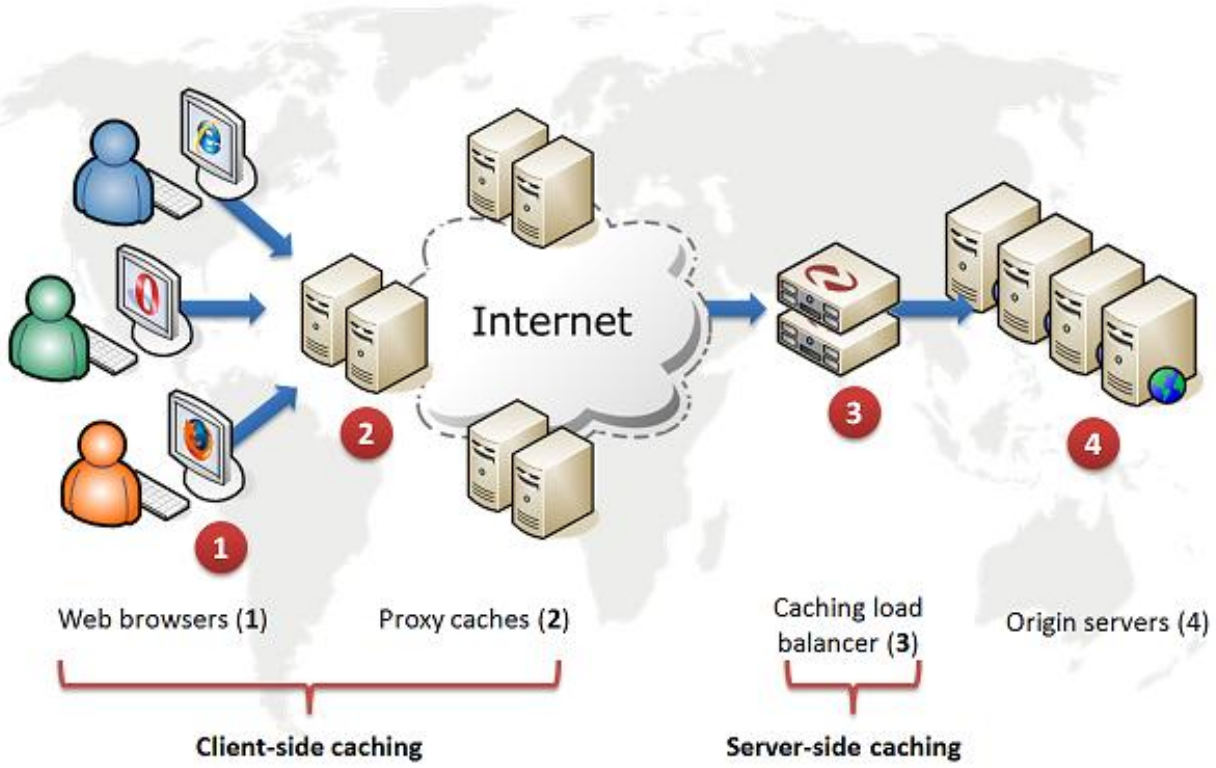
Source: http://www.useit.com/alertbox/zipf.html

Web Caching Mechanism



Source: http://en.wikibooks.org/wiki/Computer_Networks/HTTP

Web Caching Location



Source: http://knowledgehub.zeus.com/articles/2009/08/05/cache_your_website_for_just_a_second

Architectural Case Studies

D. Oppenheimer and D. Patterson, “Architecture and Dependability of Large-Scale Internet Services”, IEEE Internet Computing, Sept-Oct 2002

Case Studies

Table 2. Characteristics of the large-scale Internet services examined.

Characteristic	Online	Content	ReadMostly
Hits per day	~100 million	~7 million	~100 million
Number of machines	~500, at two data centers	~500, at four data centers plus client sites	>2,000, at four data centers
Hardware	Sparc and x86	x86	x86
Operating system	Solaris	Open-source x86	Open-source x86
Relative read/write ratio	High	Medium	Very high (and users update very little data)

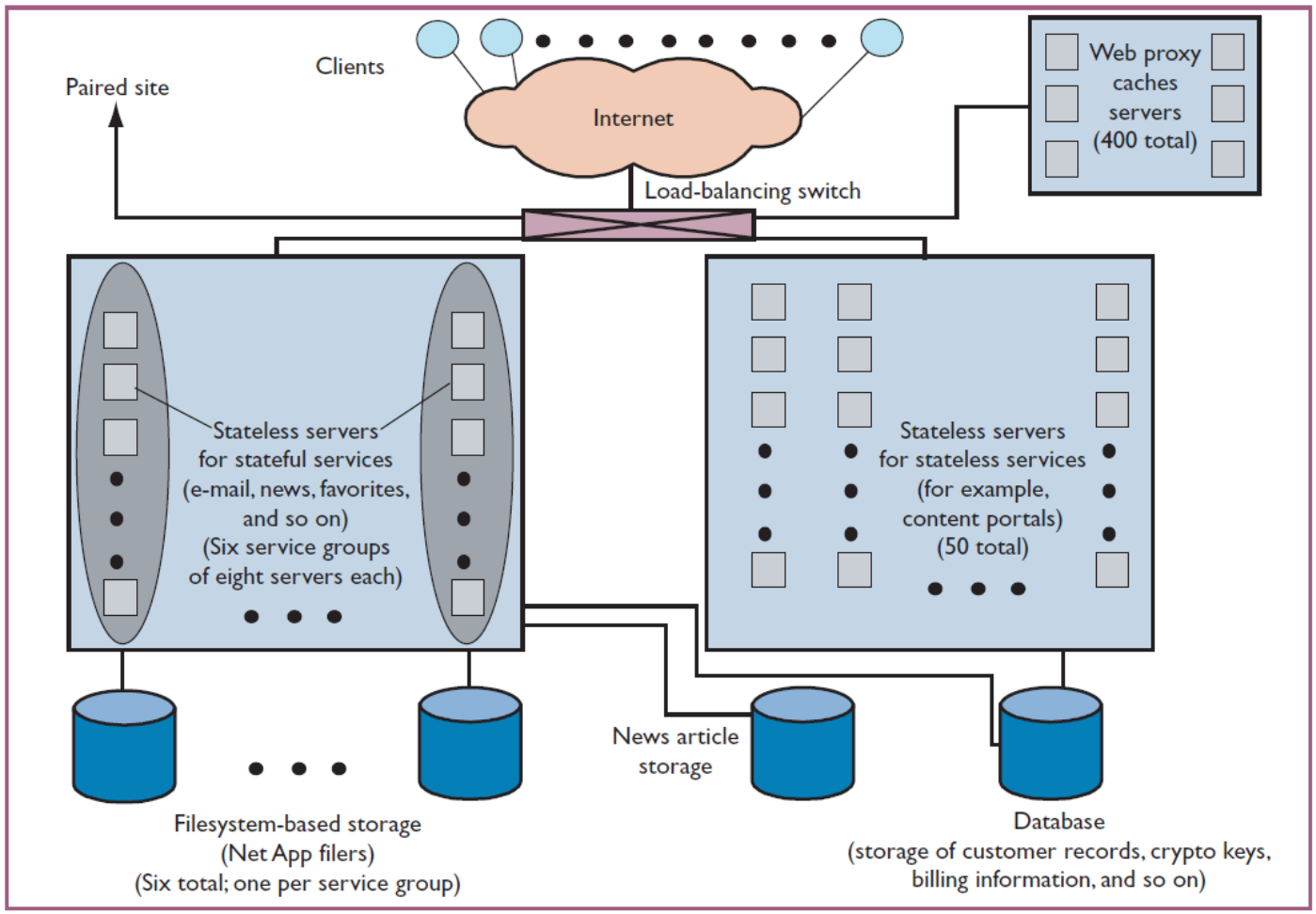
- Online - an online service/Internet portal (Hotmail)
- Content - a global content-hosting service (File sharing)
- ReadMostly - a high-traffic Internet service with a very high read-to-write ratio (Wikipedia)

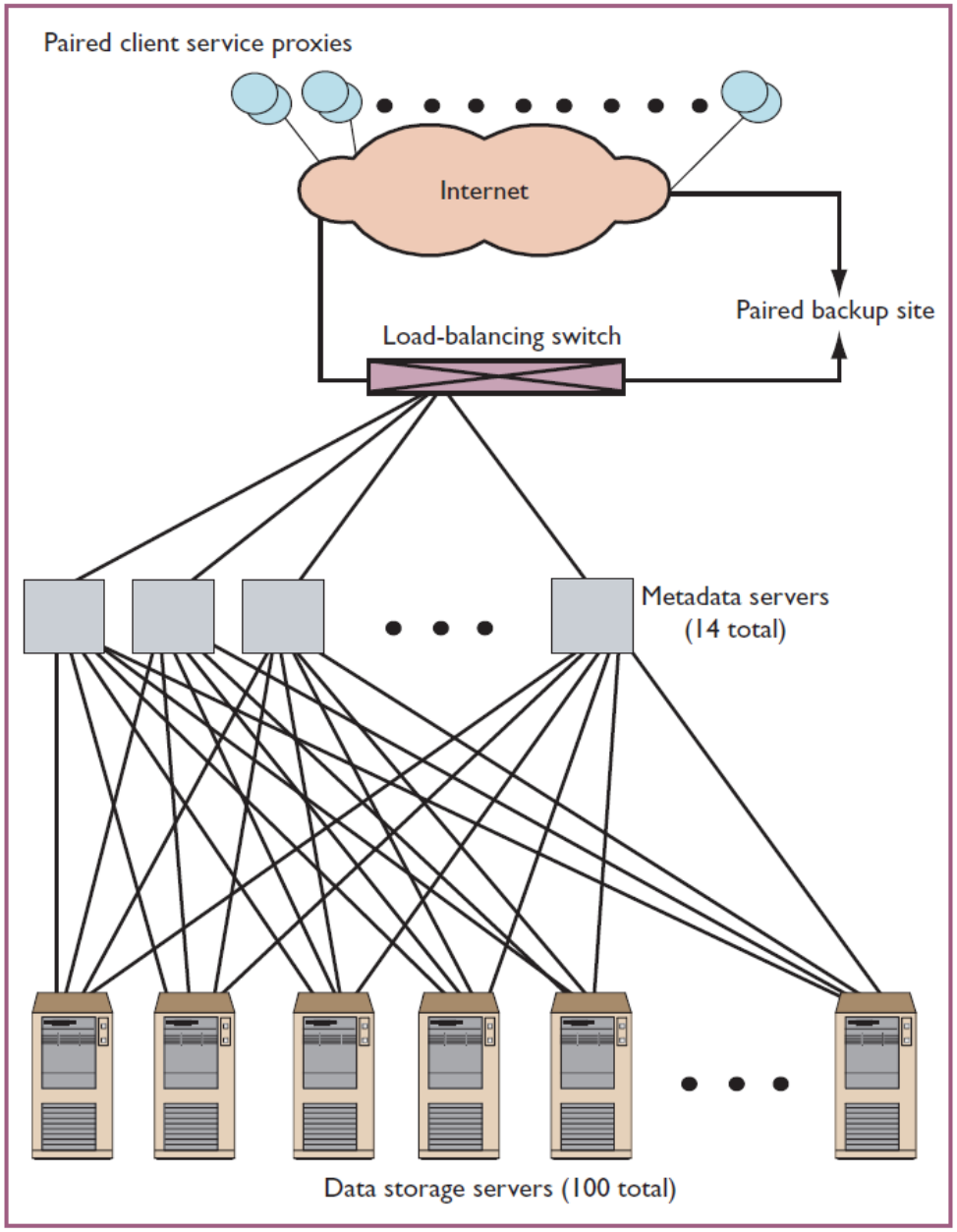
Site Architecture

- Load balancing servers
- Front-end servers
  - Run stateless codes to service requests and gather data from back-end servers
  - Web server / AppServer
- Back-end servers
  - Provide persistent data (databases, files, emails, user profiles)
  - Should utilize RAID-based storages

Online Site

Front-end: functional partitioned
Back-end: single file, single database

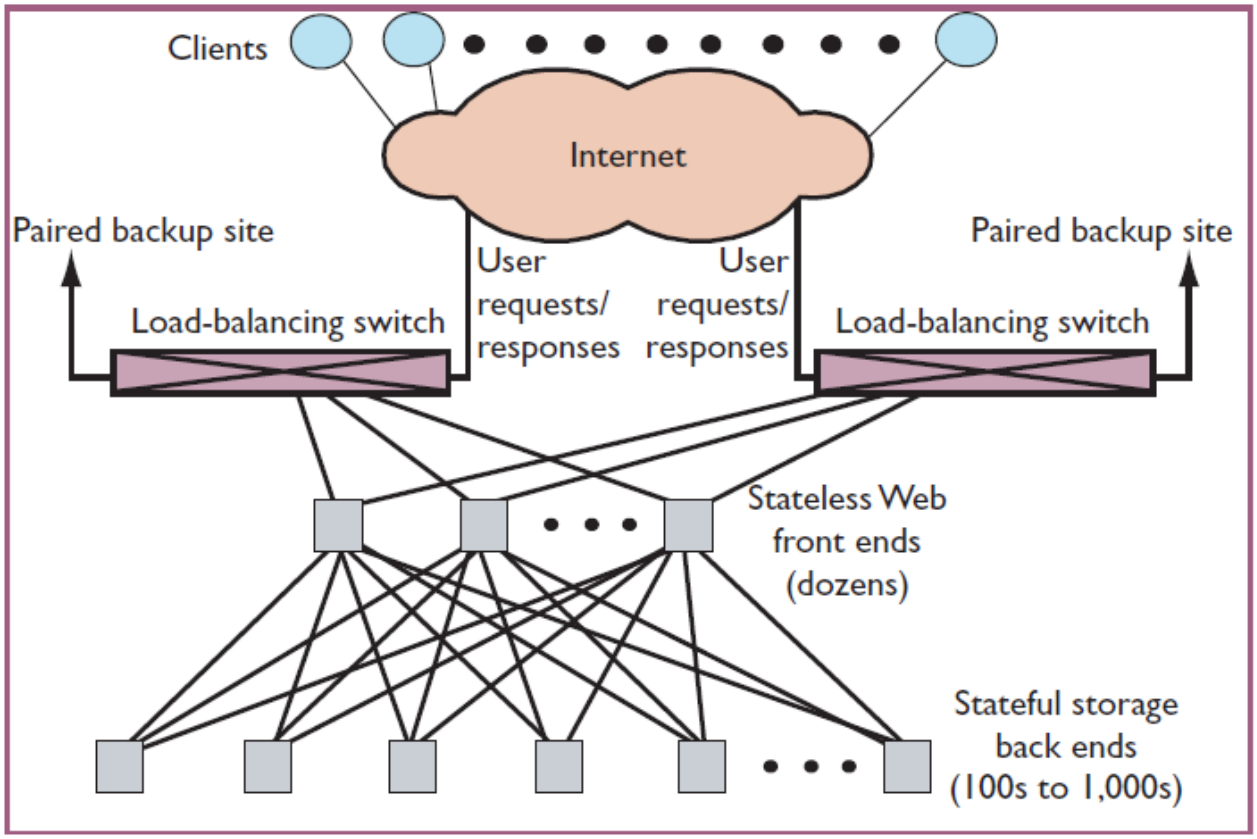




Front-end: all the same
Back-end: data partitioned

ReadMostly

Front-end: all the same
Back-end: full replication



Real-World Case Study: eBay

R. Shoup and D. Pritchett,

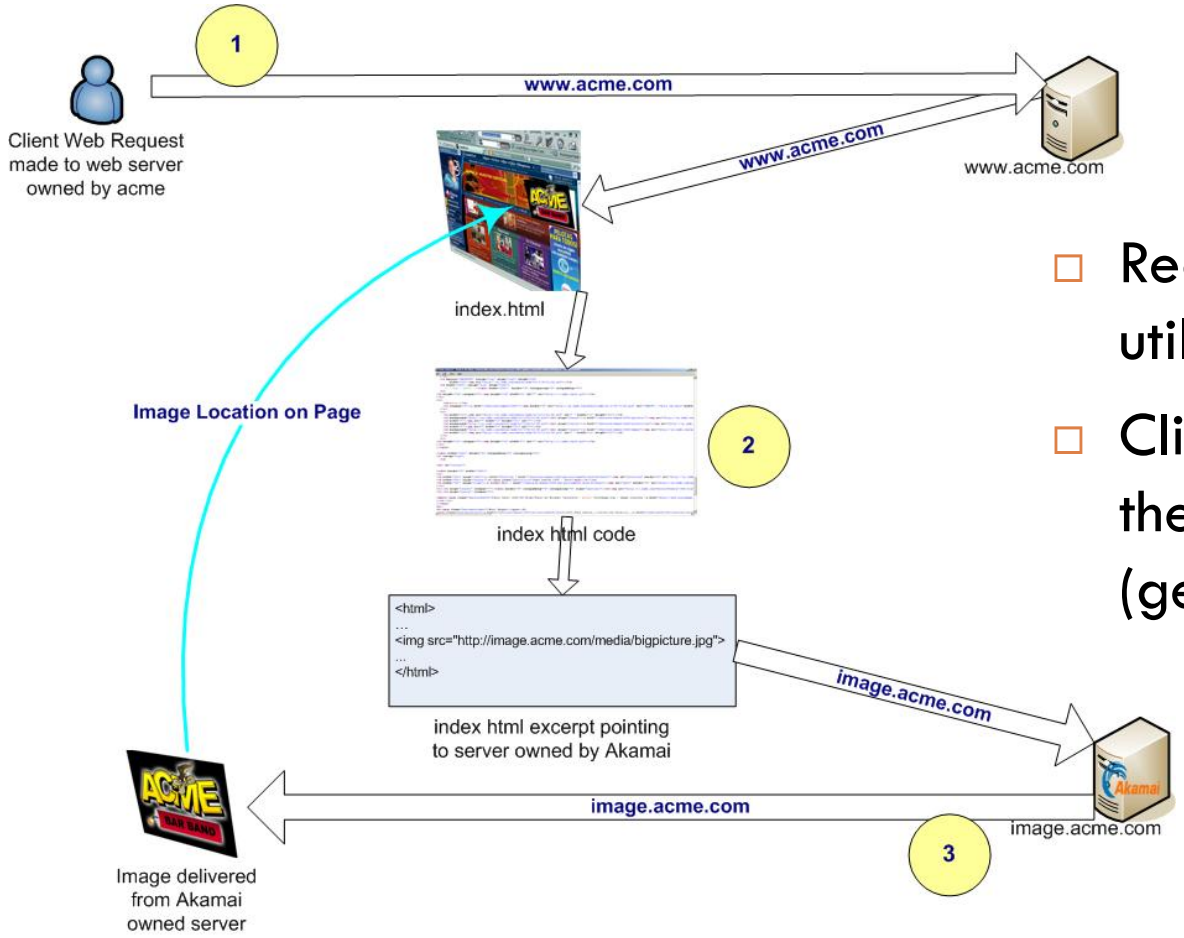
"The eBay Architecture", SD Forum 2006

eBay

- Lots of workloads
  - 212 millions registered users
  - 1 billion page views a day
  - 2 petabytes of data
- Large number of servers
  - 15,000 AppServers (IBM WebSphere)
  - 100 database servers (Oracle)
  - Utilize Akamai (CDN) for static contents

CDN: Akamai

Source: http://en.wikipedia.org/wiki/Akamai_Technologies



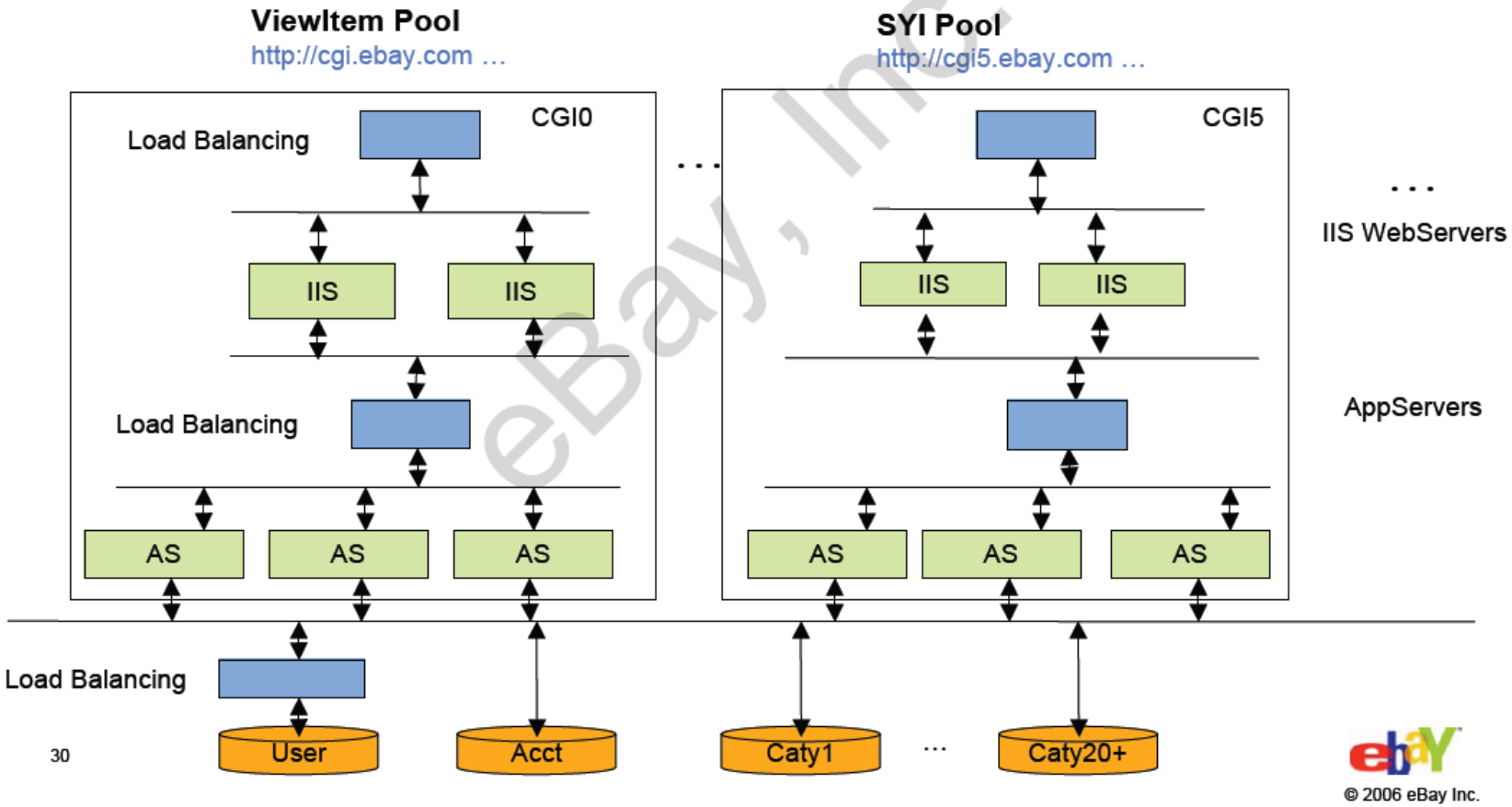
- Reduce bottlenecks by utilizing geographic
- Client gets contents from the nearest servers (geographically)

eBay Architecture Design Principles

- Application Tier
  - Segmented by function
  - Horizontal load-balancing
  - Minimize dependencies
- Data Tier
  - Data partitioned by functional areas
  - Minimize database work
    - No stored procedure / business logic in database
    - Move CPU-intensive work to applications (no join, sort, etc.)
    - AppServers are cheap, databases are bottlenecks

eBay Architecture

Source: R. Shoup and D. Pritchett, “The eBay Architecture”, SD Forum 2006



eBay, Inc.

References

- D. Oppenheimer and D. Patterson, “Architecture and Dependability of Large-Scale Internet Services”, IEEE Internet Computing, Sept-Oct 2002
- S. Hanselman, “A reminder on "Three/Multi Tier/Layer Architecture/Design" brought to you by my late night frustrations”, http://www.hanselman.com/blog/AReminderOnThreeMultiTierLayerArchitectureDesignBroughtToYouByMyLateNightFrustrations.aspx, June 2004
- R. Shoup and D. Pritchett, “The eBay Architecture”, SD Forum 2006