分布式系统
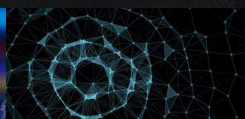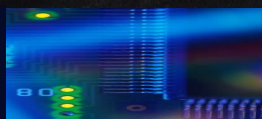## Distributed Systems

陈鹏飞
计算机学院
chenpf7@mail.sysu.edu.cn
办公室：学院楼310
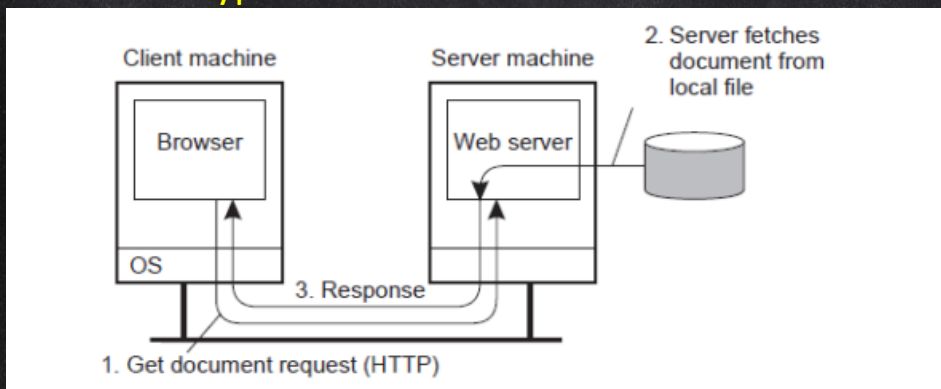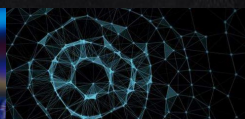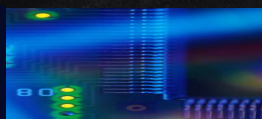主页：http://sdcs.sysu.edu.cn/node/3747

第十二讲 — 分布式**Web**系统

Distributed Web-Based Systems

➢ **Essence**

   The WWW is a huge client-server system with millions of servers; each server hosting thousands of hyperlinked documents:


```
Client machine              Server machine      2. Server fetches
                                                    document from
  ┌─────────┐                 ┌─────────────┐       local file
  │ Browser │                 │ Web server  │
  └─────────┘                 └─────────────┘
  OS                              3. Response
      1. Get document request (HTTP)
```
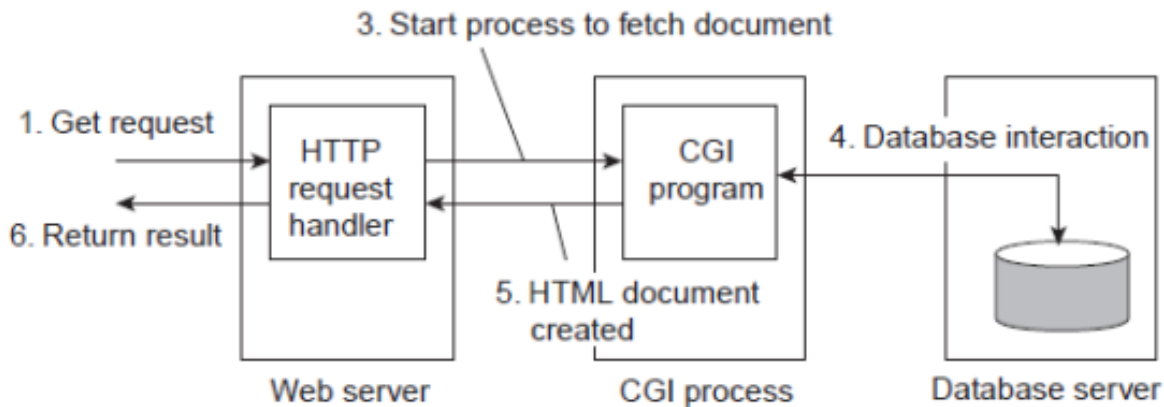
➢ Documents are generally represented in text (plain text, HTML, XML)

➢ Alternative types: images, audio, video, but also applications (PDF, PS)

➢ Documents contain scripts that are executed by the client-side software

Multi-tiered Architectures

➢ **Observation**

Conventionally, web sites were organized into three tiers.

3. Start process to fetch document

1. Get request

HTTP request handler

CGI program

4. Database interaction

6. Return result

5. HTML document created

Web server

CGI process

Database server

Web services

➢ **Observation**

　　At a certain point, people started recognizing that it is was more than just user <-> site interaction: sites could offer services to other sites => standardization is then badly needed.
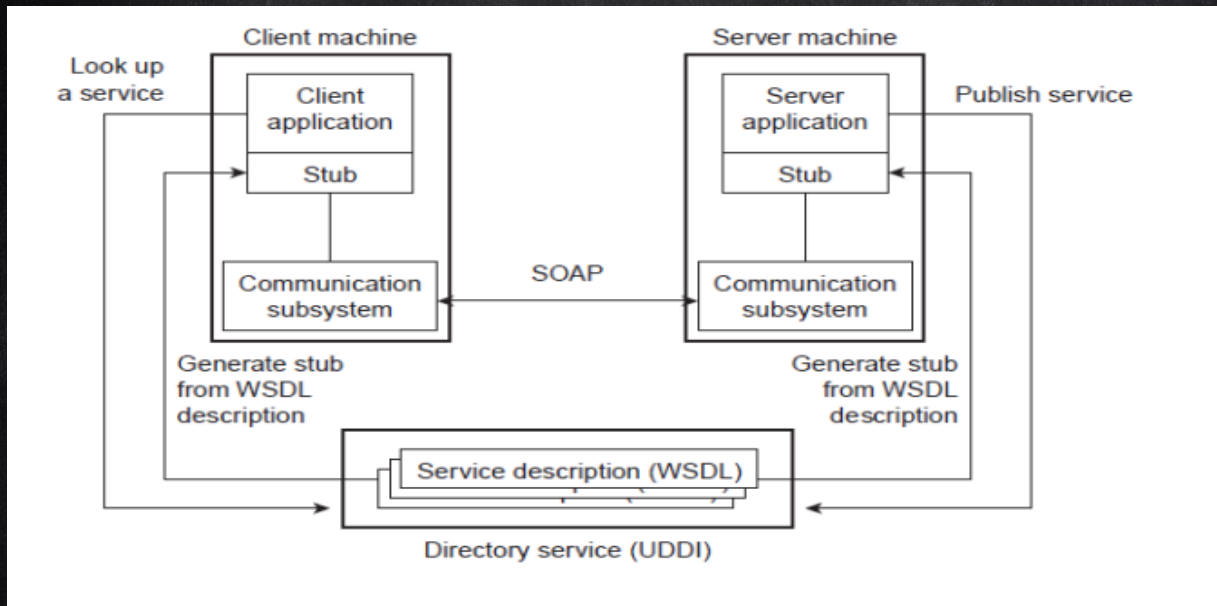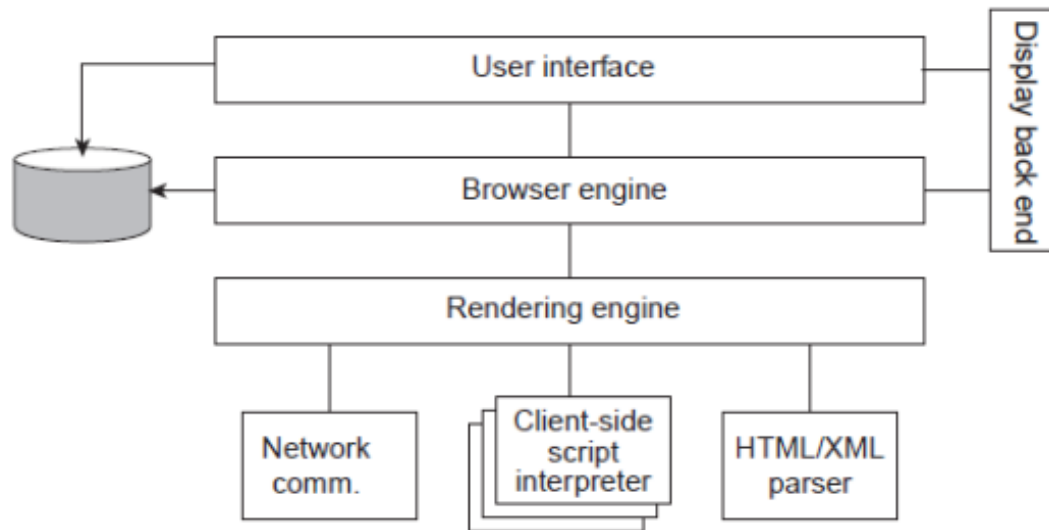
Clients: Web browsers

➢ **Observation**

browsers form the Web's most important client-side software. They are used to be simple, but that is long ago.
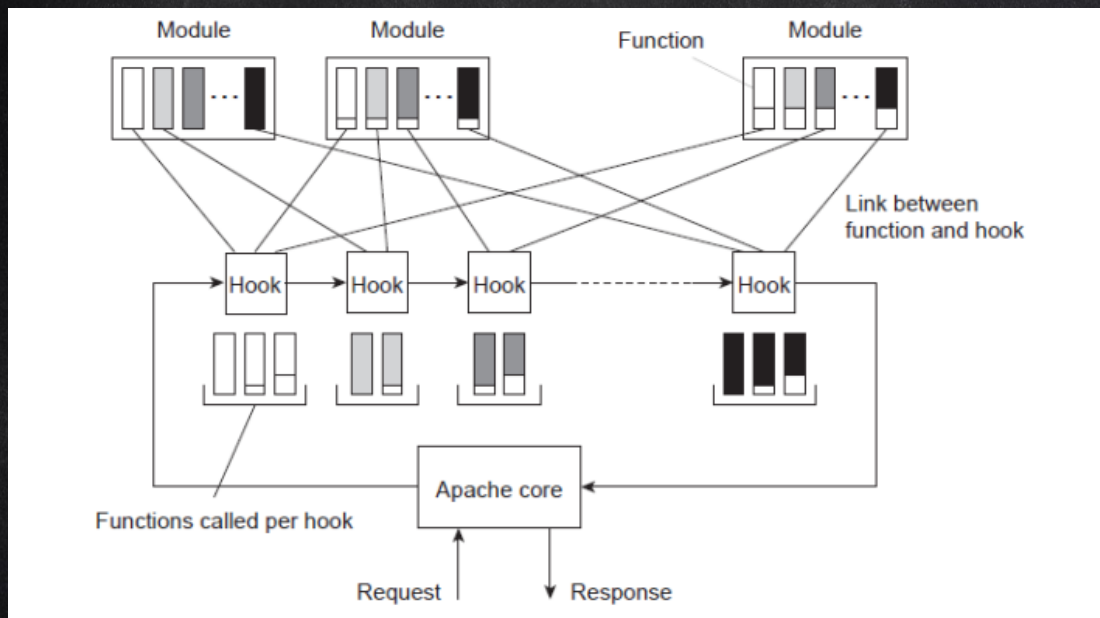
Apache Web Server

➢ **Observation**

　　More than 70% of all Web sites are based on Apache. The server is internally organized more or less according to the steps needed to process an HTTP request:
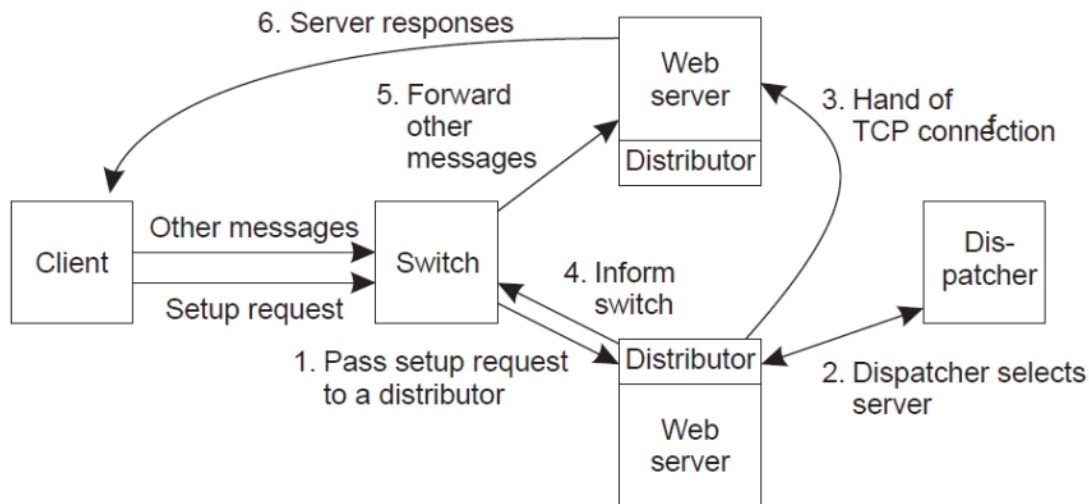
Server Clusters (2/2)

➢ **Question**

Why can content-aware distribution be so much better?

Communication (1/2)

➢ **Essence**

Communication in the Web is generally based on HTTP; a relatively simple client-server transfer protocol having the following request messages:

Operation	Description
Head	Request to return the header of a document
Get	Request to return a document to the client
Put	Request to store a document
Post	Provide data that are to be added to a document (collection)
Delete	Request to delete a document

Communication (2/2)

Header	C/S	Contents
Accept	C	The type of documents the client can handle
Accept-Charset	C	The character sets are acceptable for the client
Accept-Encoding	C	The document encodings the client can handle
Accept-Language	C	The natural language the client can handle
Authorization	C	A list of the client's credentials
WWW-Authenticate	S	Security challenge the client should respond to
Date	C+S	Date and time the message was sent
ETag	S	The tags associated with the returned document
Expires	S	The time for how long the response remains valid
From	C	The client's e-mail address
Host	C	The TCP address of the document's server
If-Match	C	The tags the document should have
If-None-Match	C	The tags the document should not have
If-Modified-Since	C	Tells the server to return a document only if it has been modified since the specified time
If-Unmodified-Since	C	Tells the server to return a document only if it has not been modified since the specified time
Last-Modified	S	The time the returned document was last modified
Location	S	A document reference to which the client should redirect its request
Referer	C	Refers to client's most recently requested document
Upgrade	C+S	The application protocol sender wants to switch to
Warning	C+S	Information about status of the data in the message

SOAP

Simple Object Access Protocol: Based on XML, this is the standard protocol for communication between Web services.

➢ SOAP is bound to an underlying protocol (i.e., it is not independent from its carrier, HTTP、SMTP)

➢ Conversational exchange style: Send a document one way, get a filled-in response back.

➢ RPC-style exchange: Used to invoke a Web service.

➢ **Observation**
XML has the advantage of allowing self describing documents.

```
env:Envelope xmlns:env="http://www.w3.org/2003/05/soap-envelope">
    <env:Header>
        <n:alertcontrol xmlns:n="http://example.org/alertcontrol">
            <n:priority>1</n:priority>
            <n:expires>2001-06-22T14:00:00-05:00</n:expires>
        </n:alertcontrol>
    </env:Header>
    <env:Body>
        <m:alert xmlns:m="http://example.org/alert">
            <m:msg>Pick up Mary at school at 2pm</m:msg>
        </m:alert>
    </env:Body>
</env:Envelope>
```

Naming: URL

➢ **URL**

Uniform Resource Locator tells how and where to access a resource.

Scheme	Host name	Pathname
http ://	www.cs.vu.nl	/home/steen/mbox

(a)

Scheme	Host name	Port	Pathname
http ://	www.cs.vu.nl :	80	/home/steen/mbox

(b)

Scheme	Host name	Port	Pathname
http ://	130.37.24.11 :	80	/home/steen/mbox

(c)

http	HTTP	http://www.cs.vu.nl:80/globe
mailto	Mail	mailto:steen@cs.vu.nl
ftp	FTP	ftp://ftp.cs.vu.nl/pub/minix/README
file	Local file	file:/edu/book/work/chp/11/11
data	Inline data	data:text/plain;charset=iso-8859-7, %e1%e2%e3
telnet	Remote login	telnet://flits.cs.vu.nl

Web Proxy Caching

➢ **Basic Idea**

Sites install a separate proxy server that handles all outgoing requests. Proxies subsequently cache incoming documents. Cache-consistency protocols:

- Always verify validity by contacting server
- Age-based consistency:

$$T_{expire} = \alpha \cdot (T_{cached} - T_{last\_modified}) + T_{cached}$$

- Cooperative caching, by which you first check your neighbors on a cache miss:
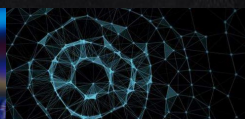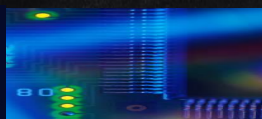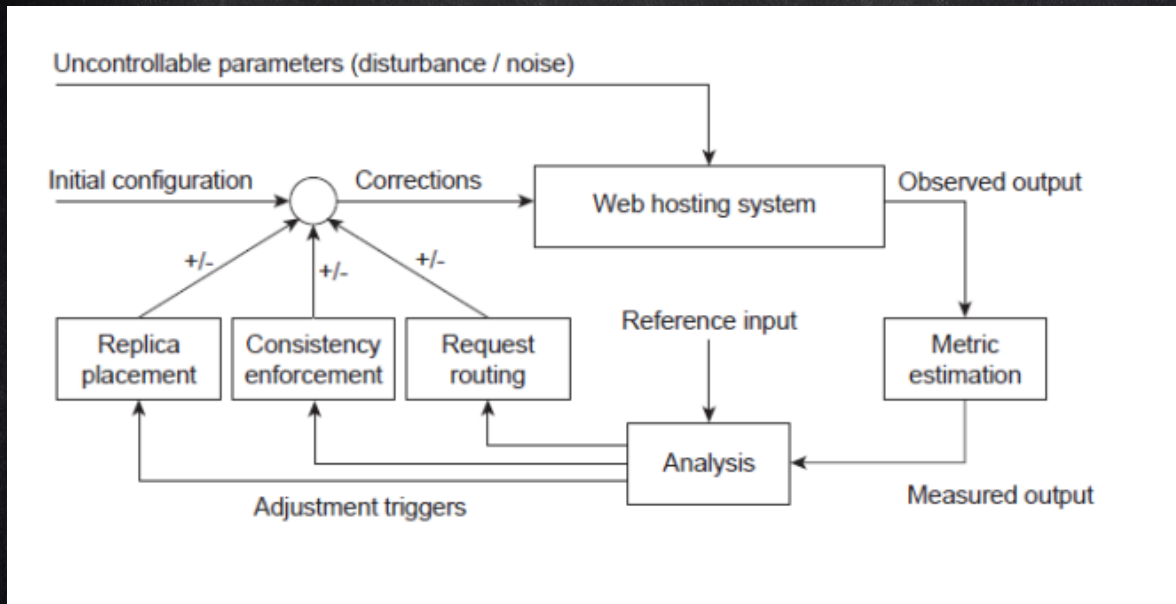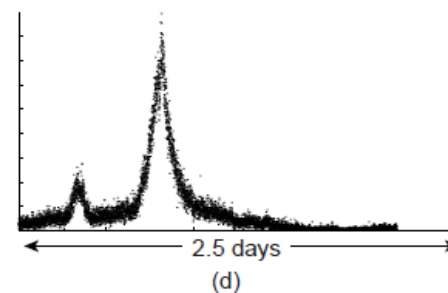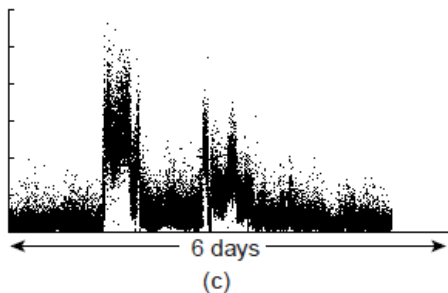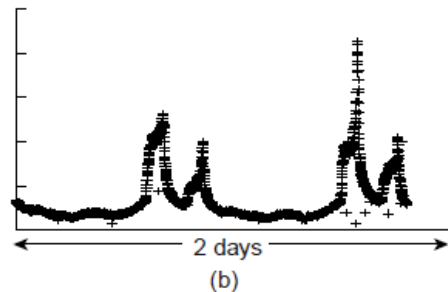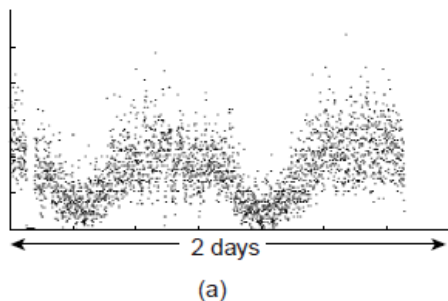
Web Proxy Caching
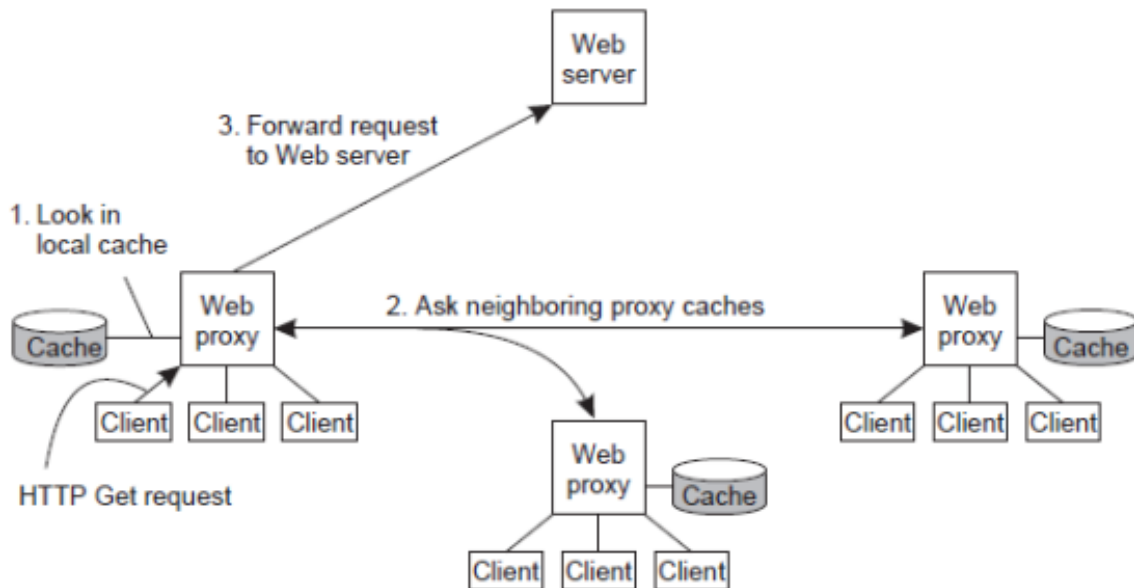
Replication in Web Hosting Systems

➢ **Observation**

Sites install a separate proxy server that handles all outgoing requests. Proxies subsequently cache incoming documents. Cache-consistency protocols:

➤ **Observation**

We need dynamic adjustment to balance resource usage. Flash crowds introduce a serious problem:

Server Replication

➢ **Content Delivery Network**

   CDNs act as Web hosting services to replicate documents across the Internet providing their customers guarantees on high availability
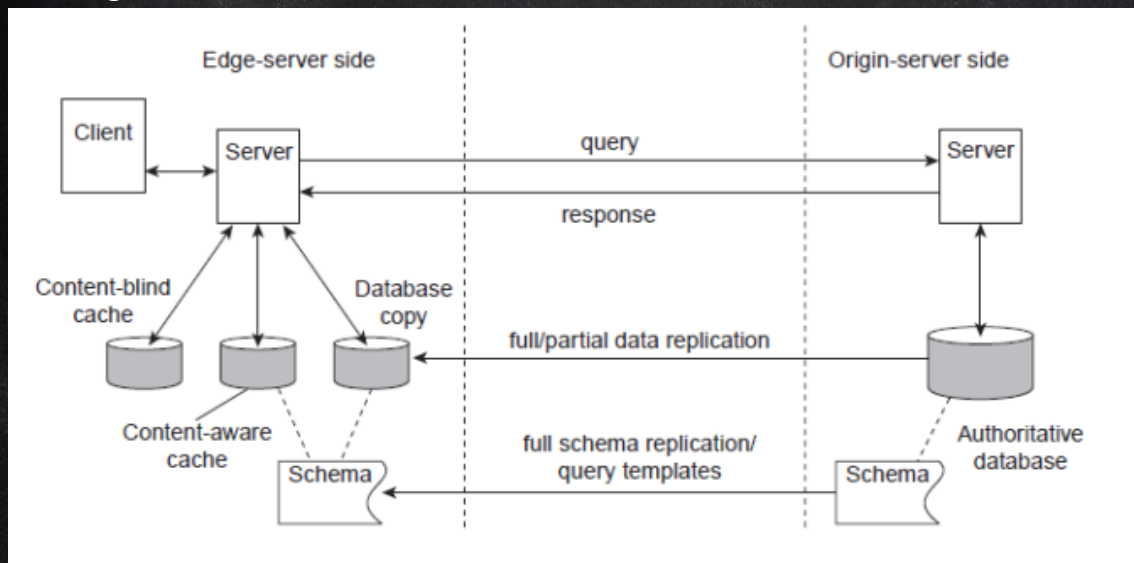and performance (example: Akamai).
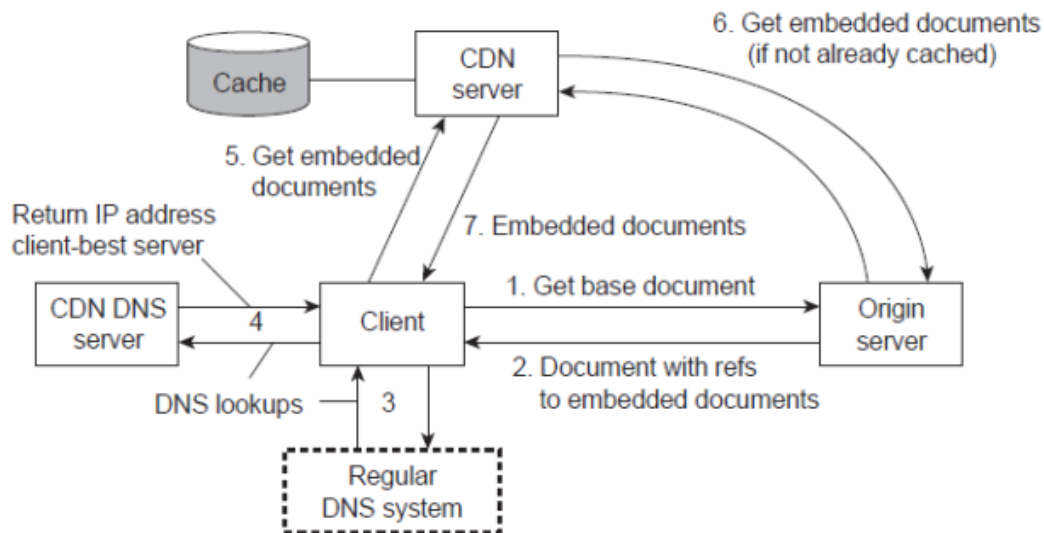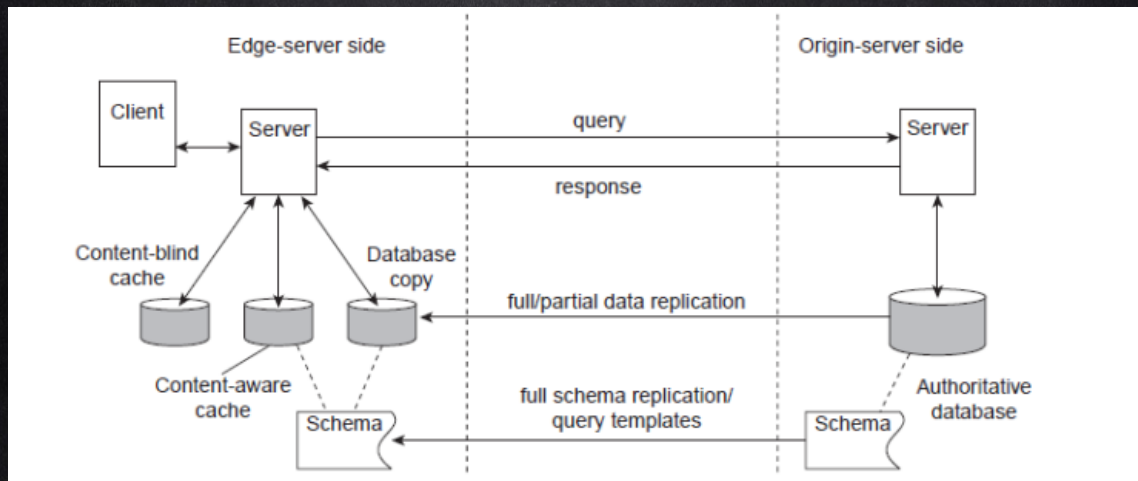
Replication of Web Apps. (1/3)

➢ **Observation**

Replication becomes more difficult when dealing with databses and such. No single best solution.
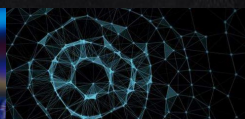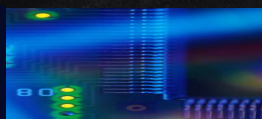

Assumption: Updates are carried out at origin server, and propagated to edge servers.
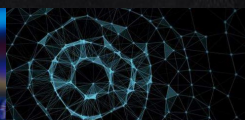
Replication of Web Apps. (2/3)

> **Full replication:** high read/write ratio, often in combination with complex queries. **Note:** replication may possibly speed-down performance when R/W ratio goes down.
> **Partial replication:** high read/write ratio, but in combination with simple queries

Replication of Web Apps. (3/3)
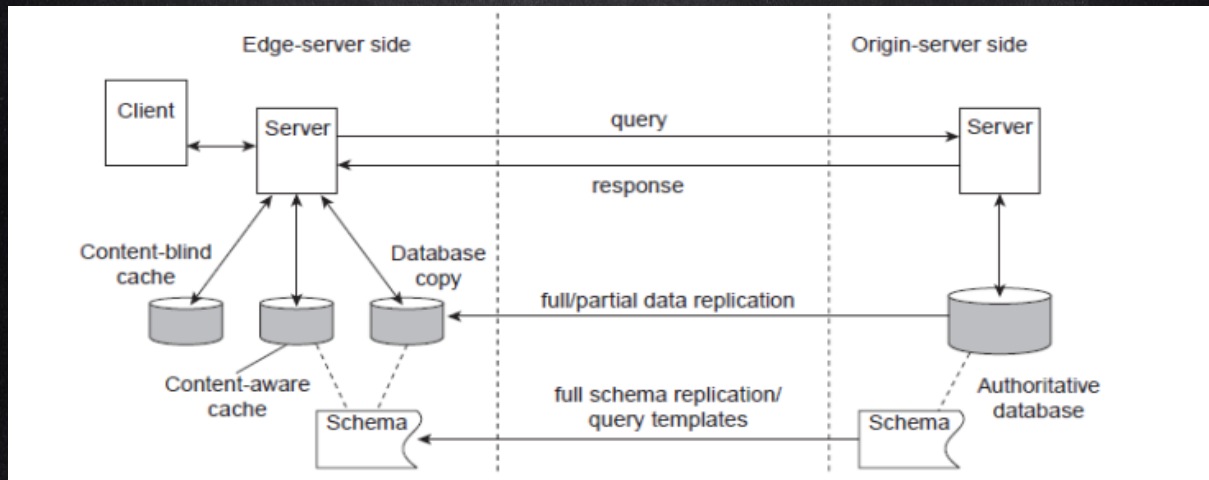
➢ **Content-aware caching:** Check for queries at local database, and subscribe for invalidations at the server. Works good with range queries and complex queries.

➢ **Content-blind caching:** Simply cache the result of previous queries. Works great with simple queries that address unique results (e.g., no range queries).

谢谢！