Distributed Systems
Principles and Paradigms

Maarten van Steen

VU Amsterdam, Dept. Computer Science
Room R4.20, steen@cs.vu.nl

Chapter 12: Distributed Web-Based Systems

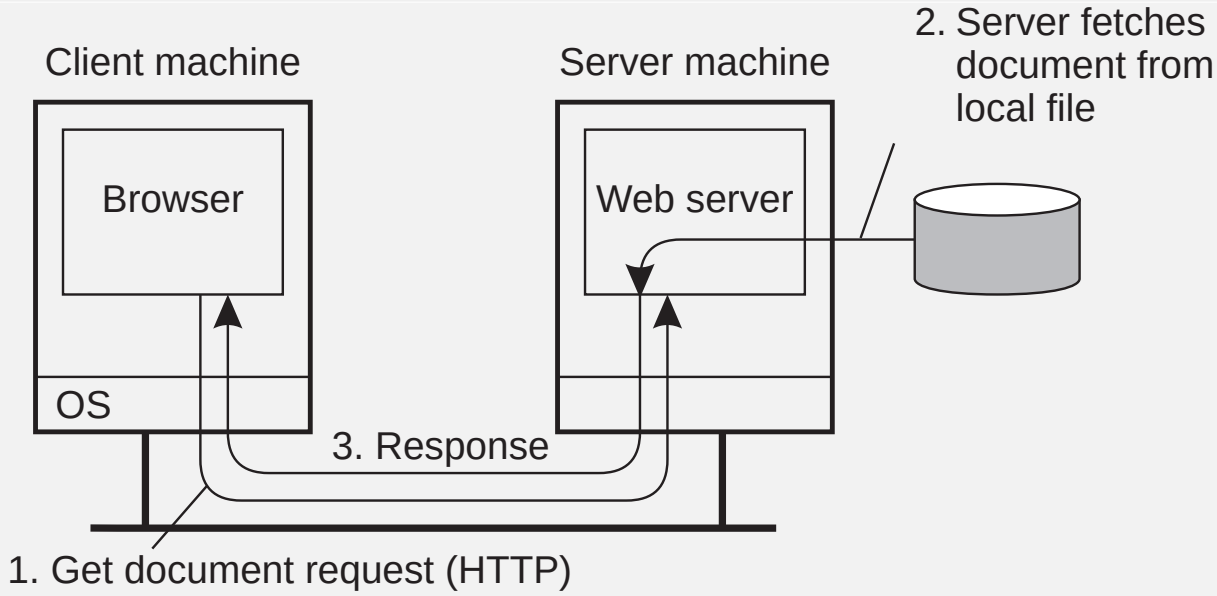
Version: December 10, 2012

Distributed Web-based systems

Essence

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

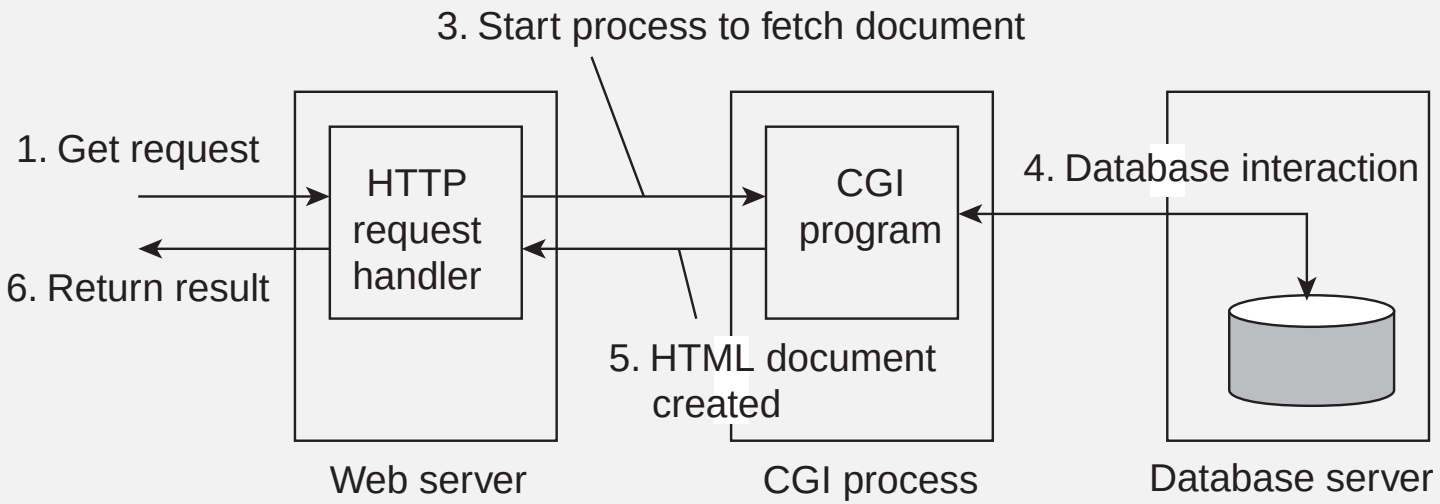
- Documents are often represented in text (plain text, HTML, XML)
- Alternative types: images, audio, video, applications (PDF, PS)
- Documents may contain scripts, executed by client-side software



Multi-tiered architectures

Observation

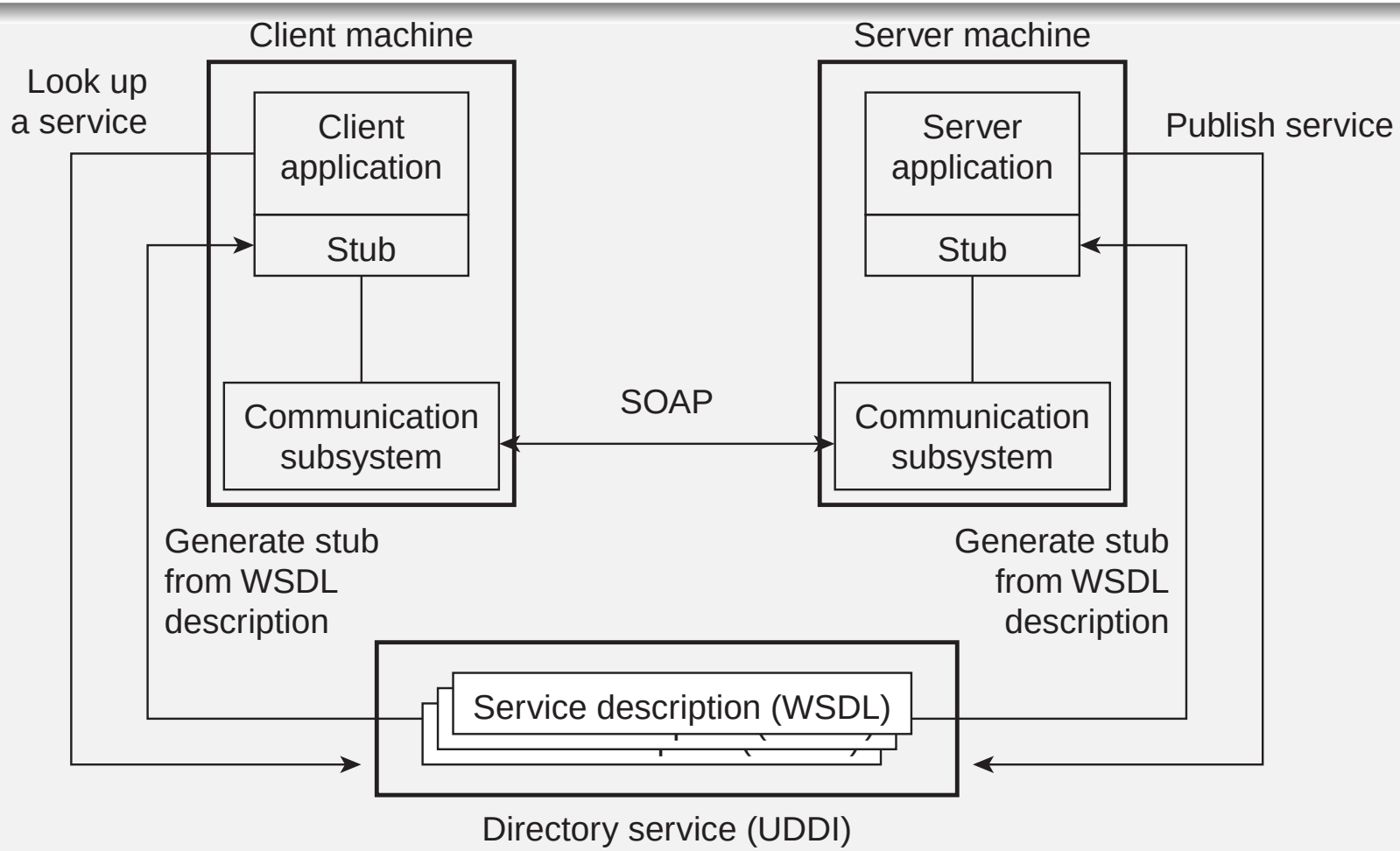
Already very soon, Web sites were organized into three tiers.



Web services

Observation

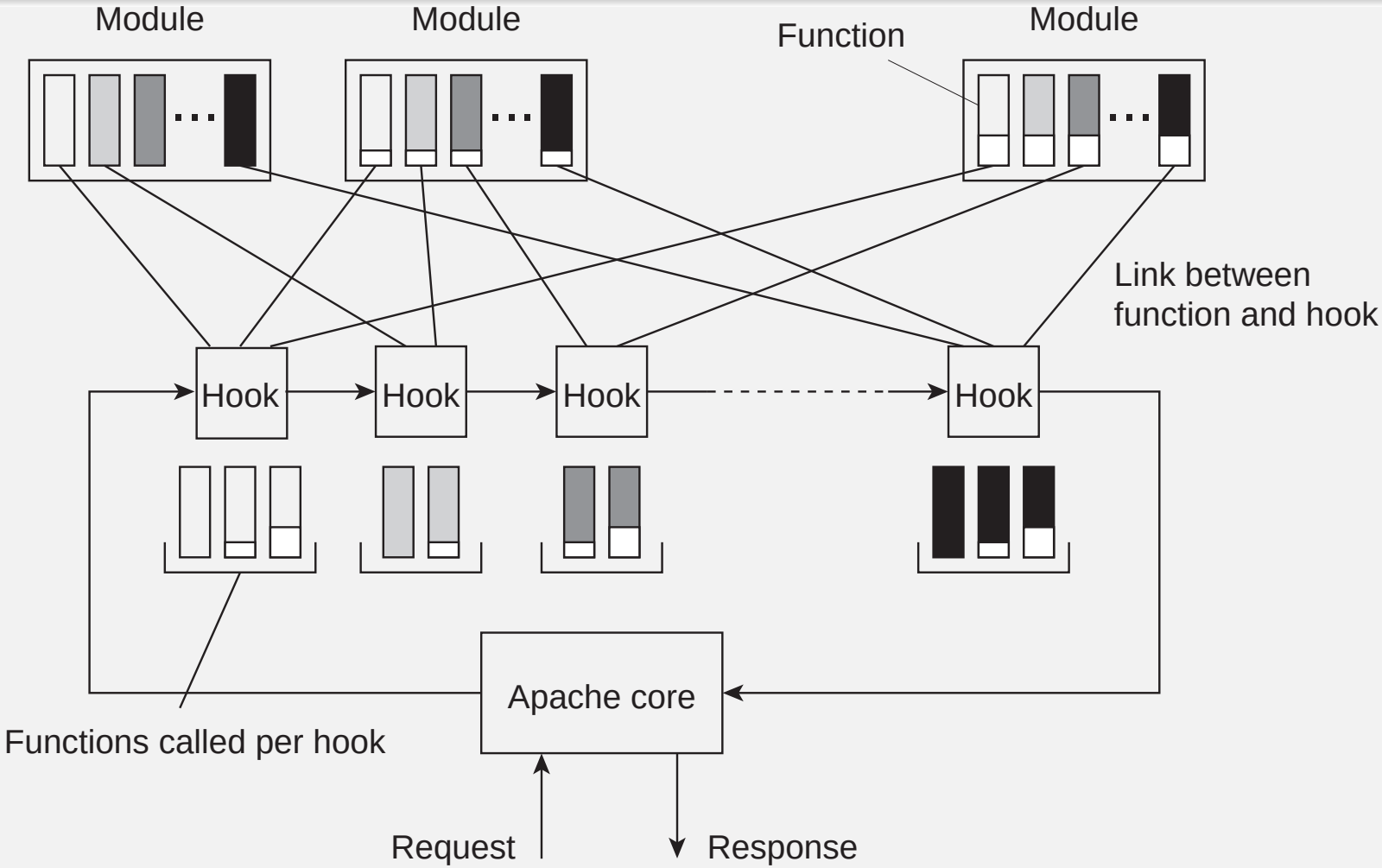
At a certain point, people started recognizing that it is was more than just user $\leftrightarrow$ site interaction: sites could offer services to other sites $\Rightarrow$ standardization is then badly needed.



Apache Web server

Observation: More than 52% of all 185 million Web sites are Apache.

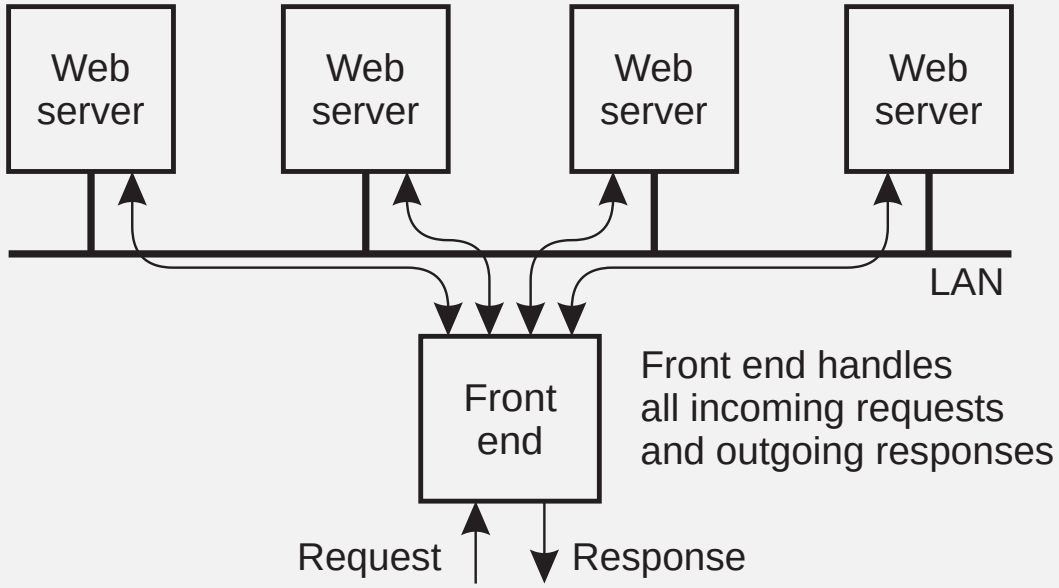
The server is internally organized more or less according to the steps needed to process an HTTP request.



Server clusters

Essence

To improve performance and availability, WWW servers are often clustered in a way that is transparent to clients.



Server clusters

Problem

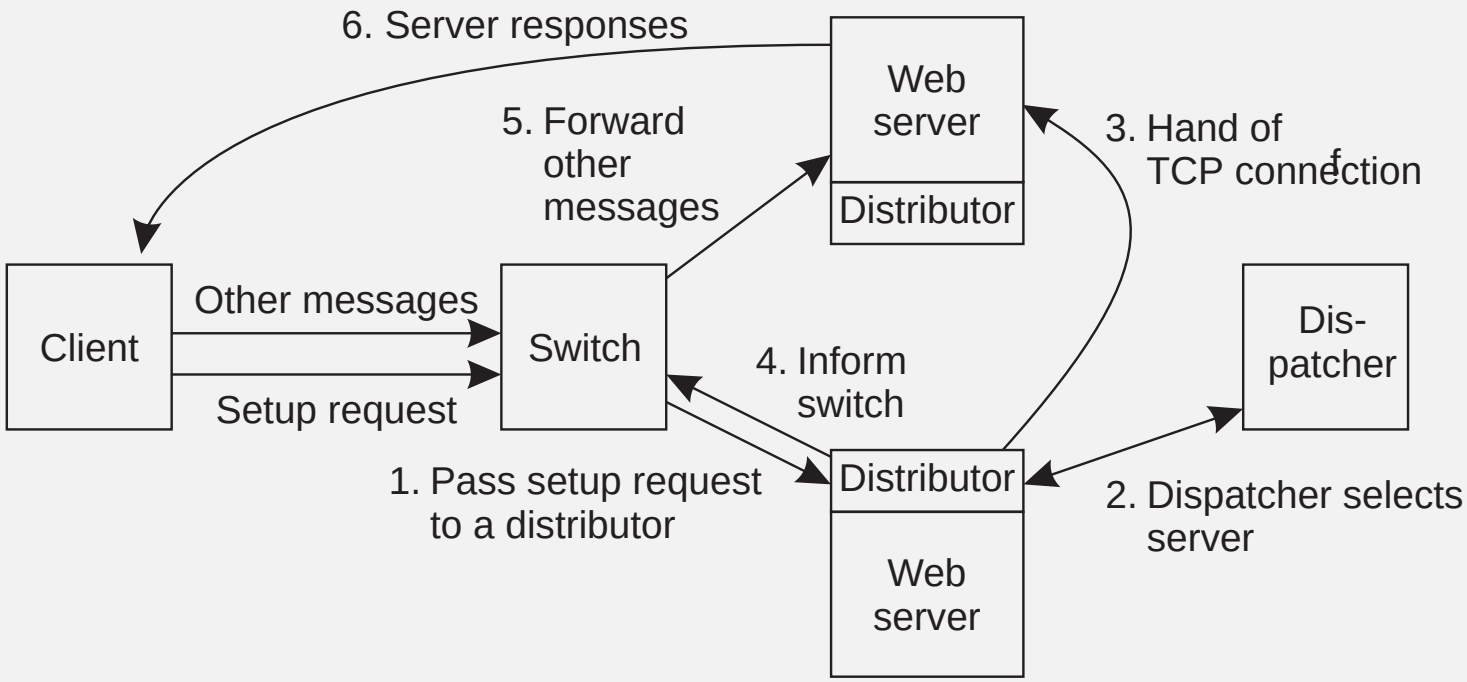
The front end may easily get overloaded, so that special measures need to be taken.

- Transport-layer switching: Front end simply passes the TCP request to one of the servers, taking some performance metric into account.
- Content-aware distribution: Front end reads the content of the HTTP request and then selects the best server.

Server Clusters

Question

Why can content-aware distribution be so much better?



Web proxy caching

Basic idea

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

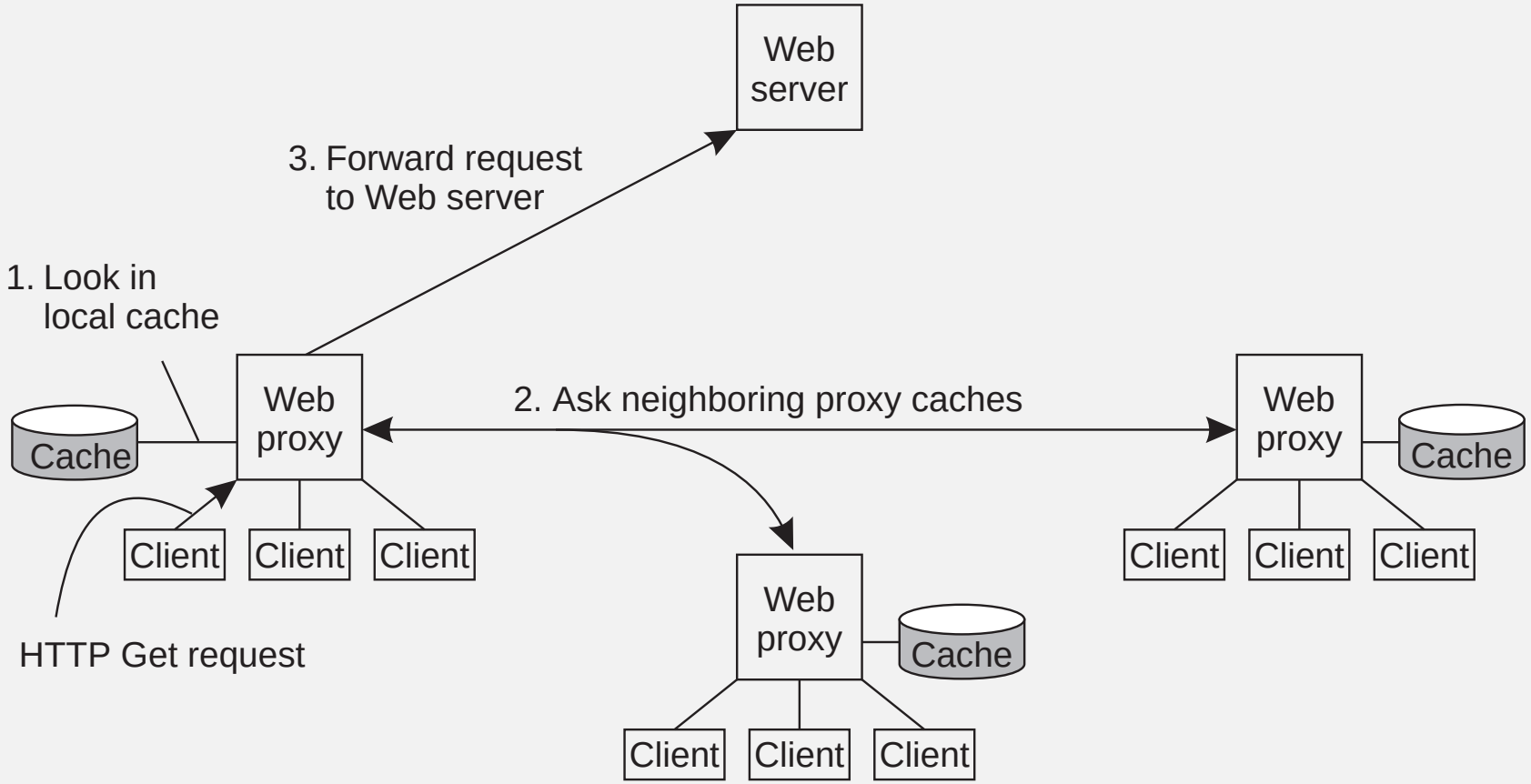
- Always verify validity by contacting server
- Age-based consistency:

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

Web proxy caching

Basic idea (cnt'd)

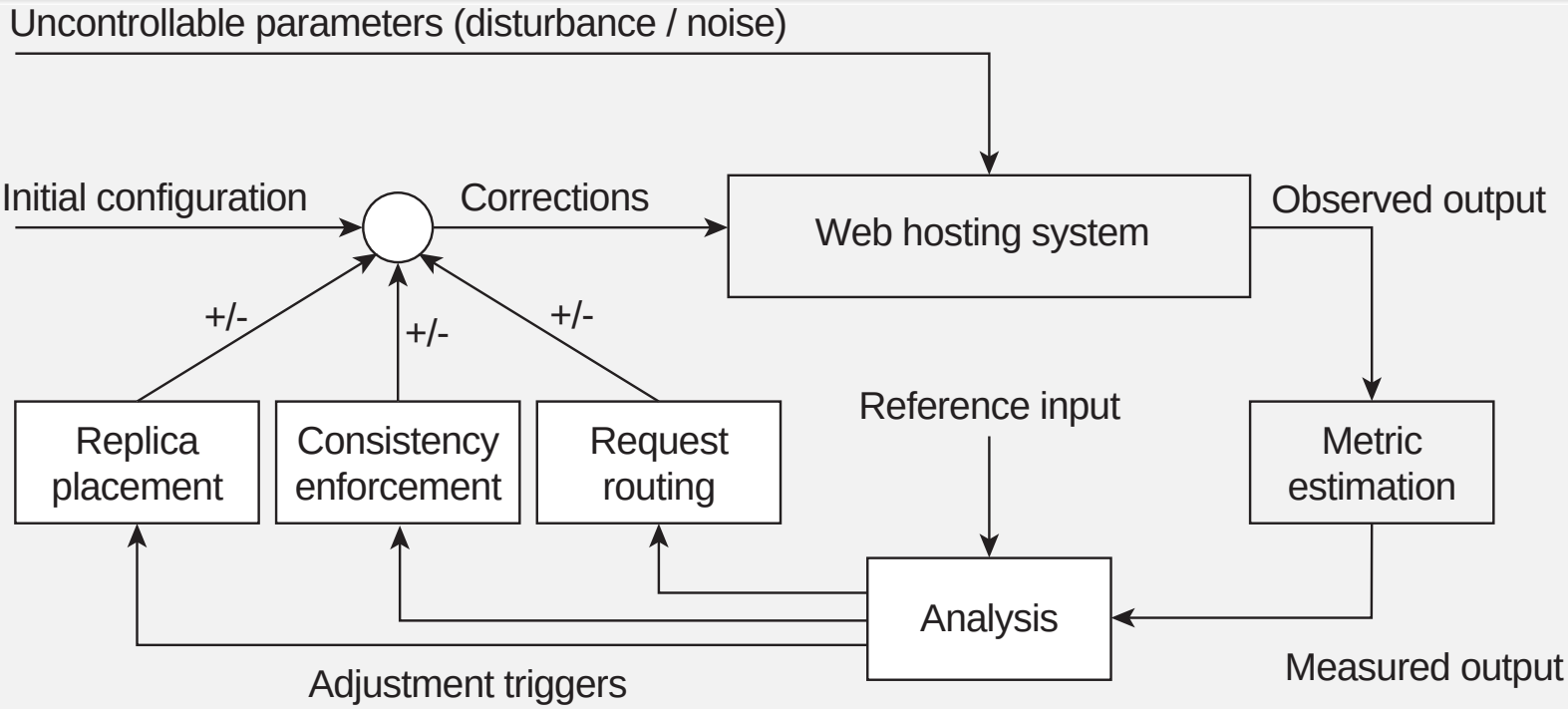
- Cooperative caching, by which you first check your neighbors on a cache miss



Replication in Web hosting systems

Observation

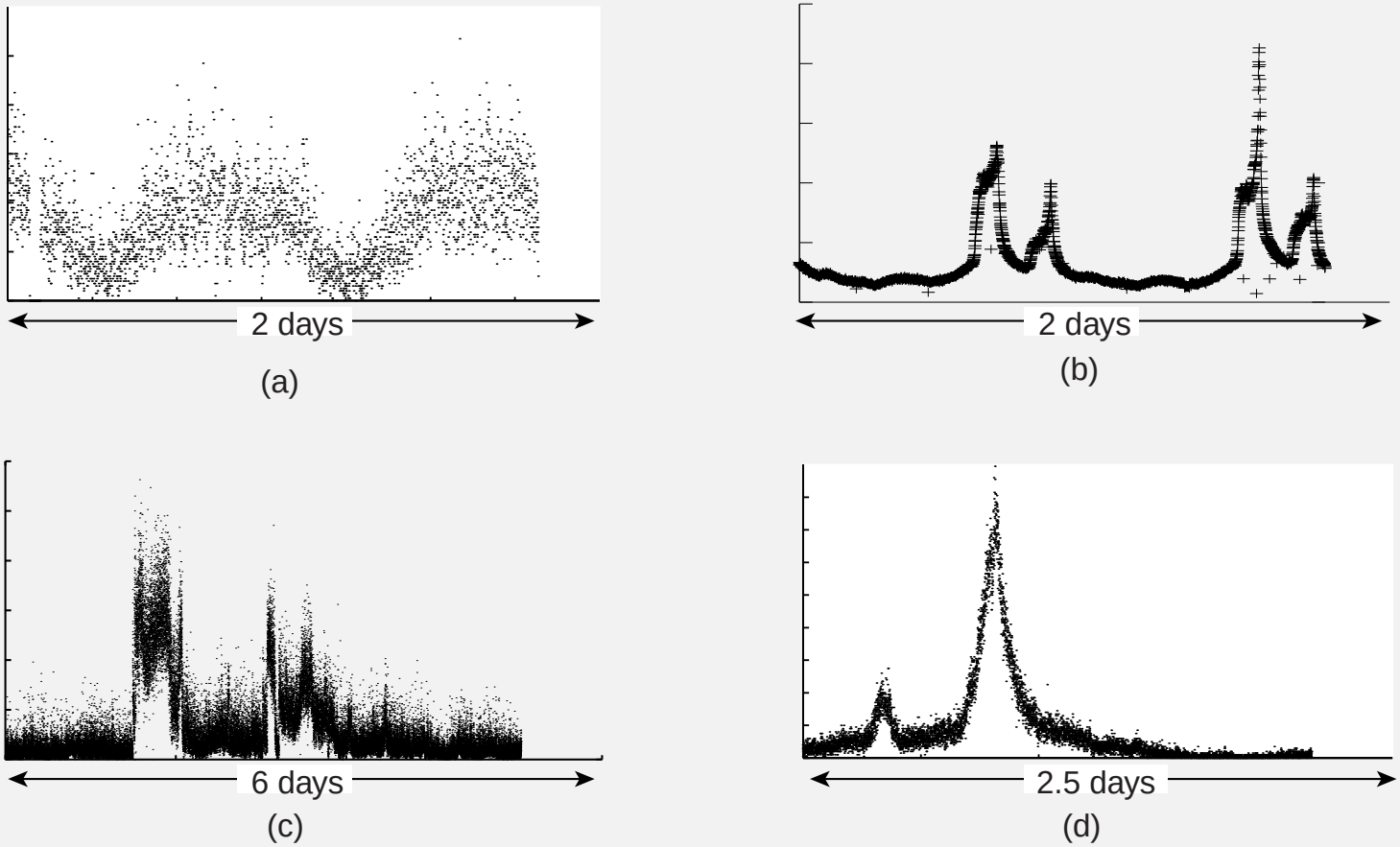
By-and-large, Web hosting systems are adopting replication to increase performance. Much research is done to improve their organization. Follows the lines of self-managing systems.



Handling flash crowds

Observation

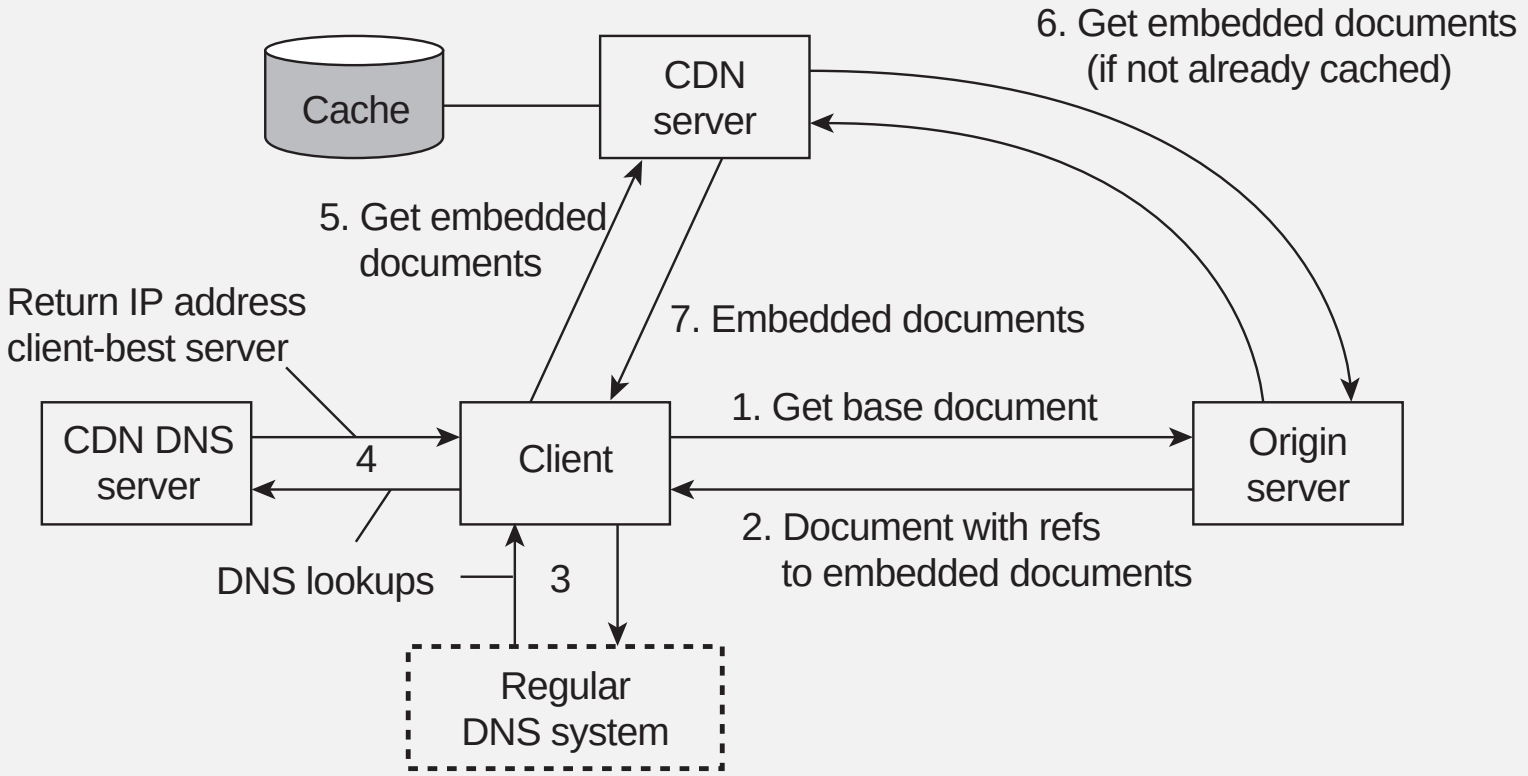
We need dynamic adjustment to balance resource usage. Flash crowds introduce a serious problem.



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 applications

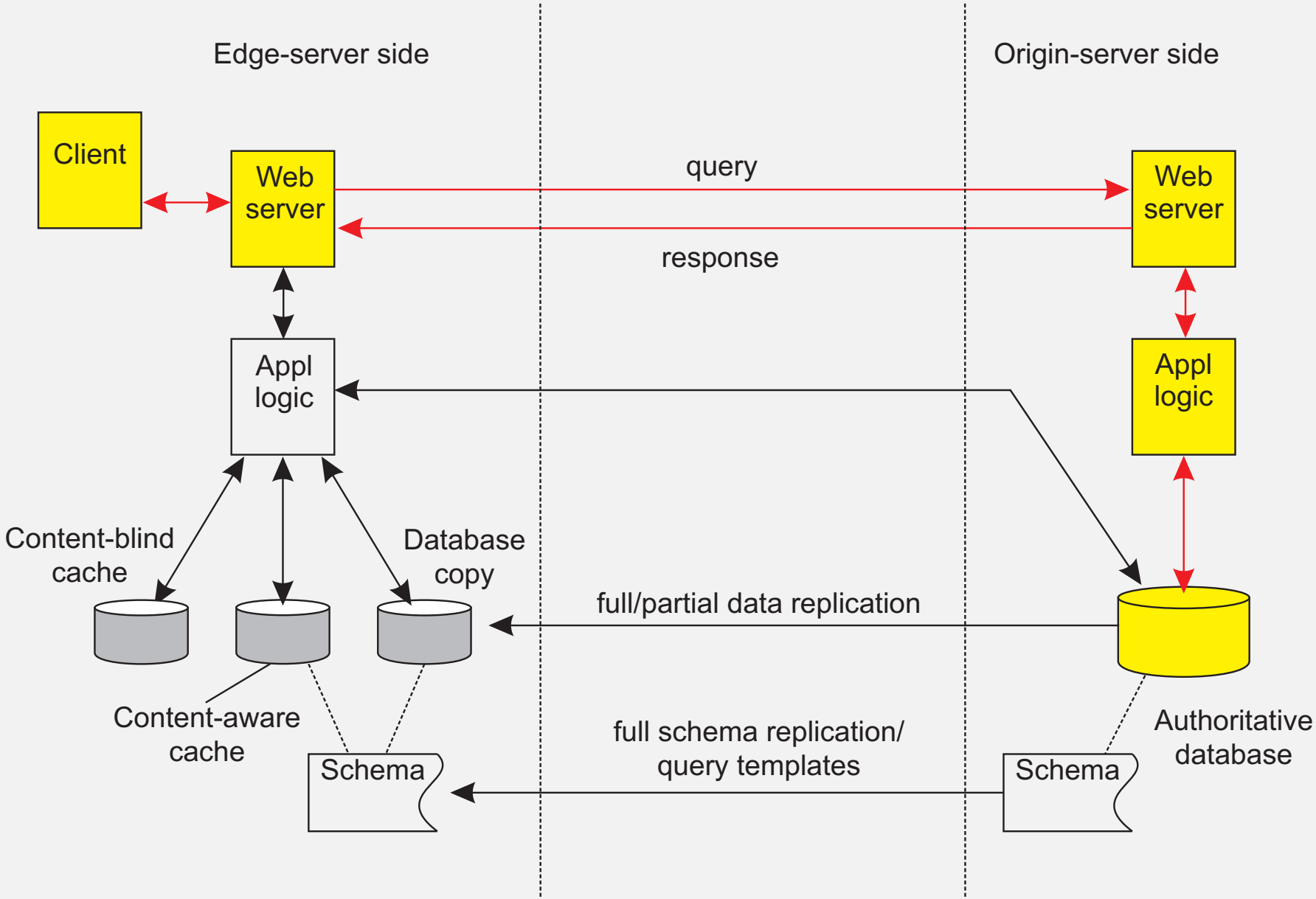
Observation

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 applications: normal



Replication of Web applications

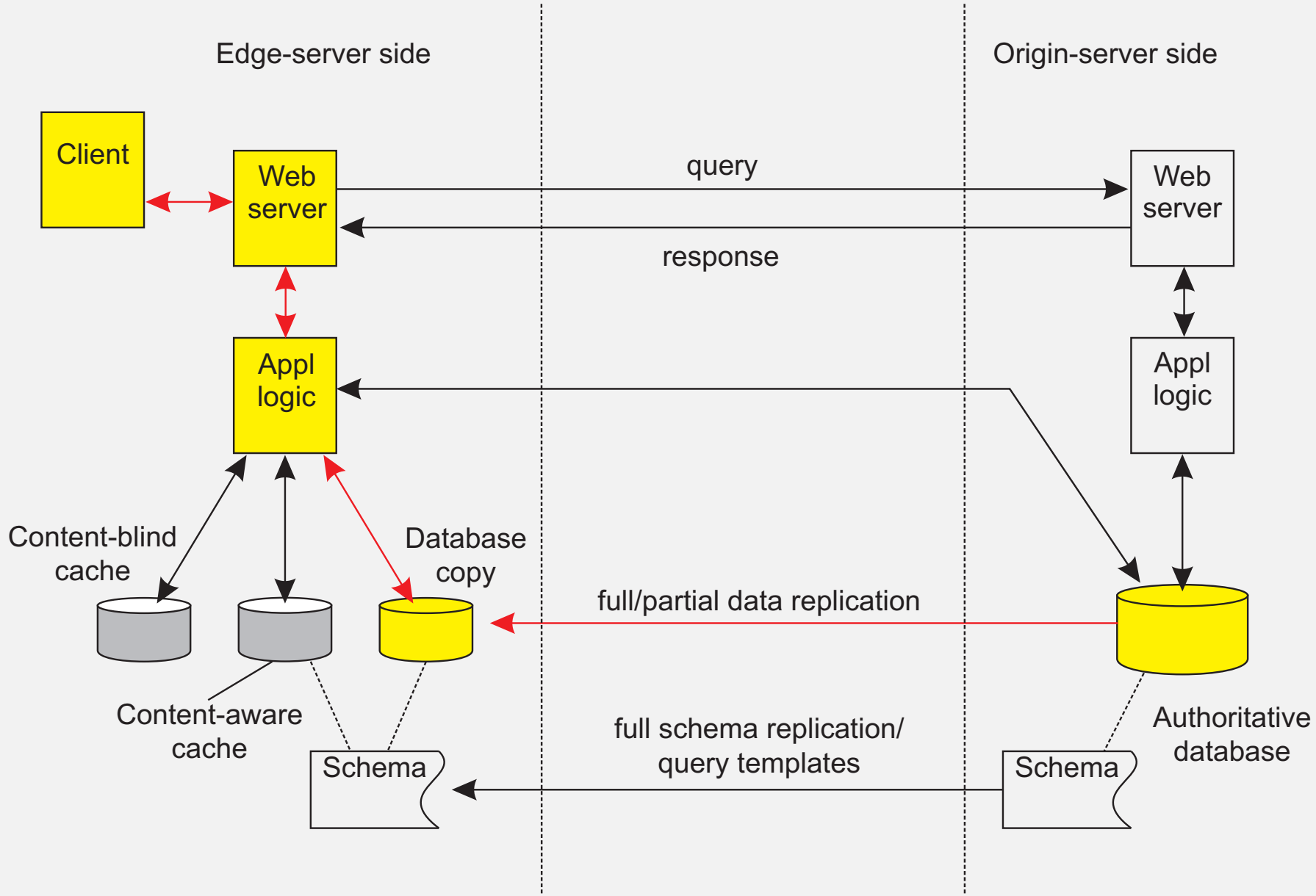
Alternative solutions

- Full replication: high read/write ratio, often in combination with complex queries.
- Partial replication: high read/write ratio, but in combination with simple queries
- 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).

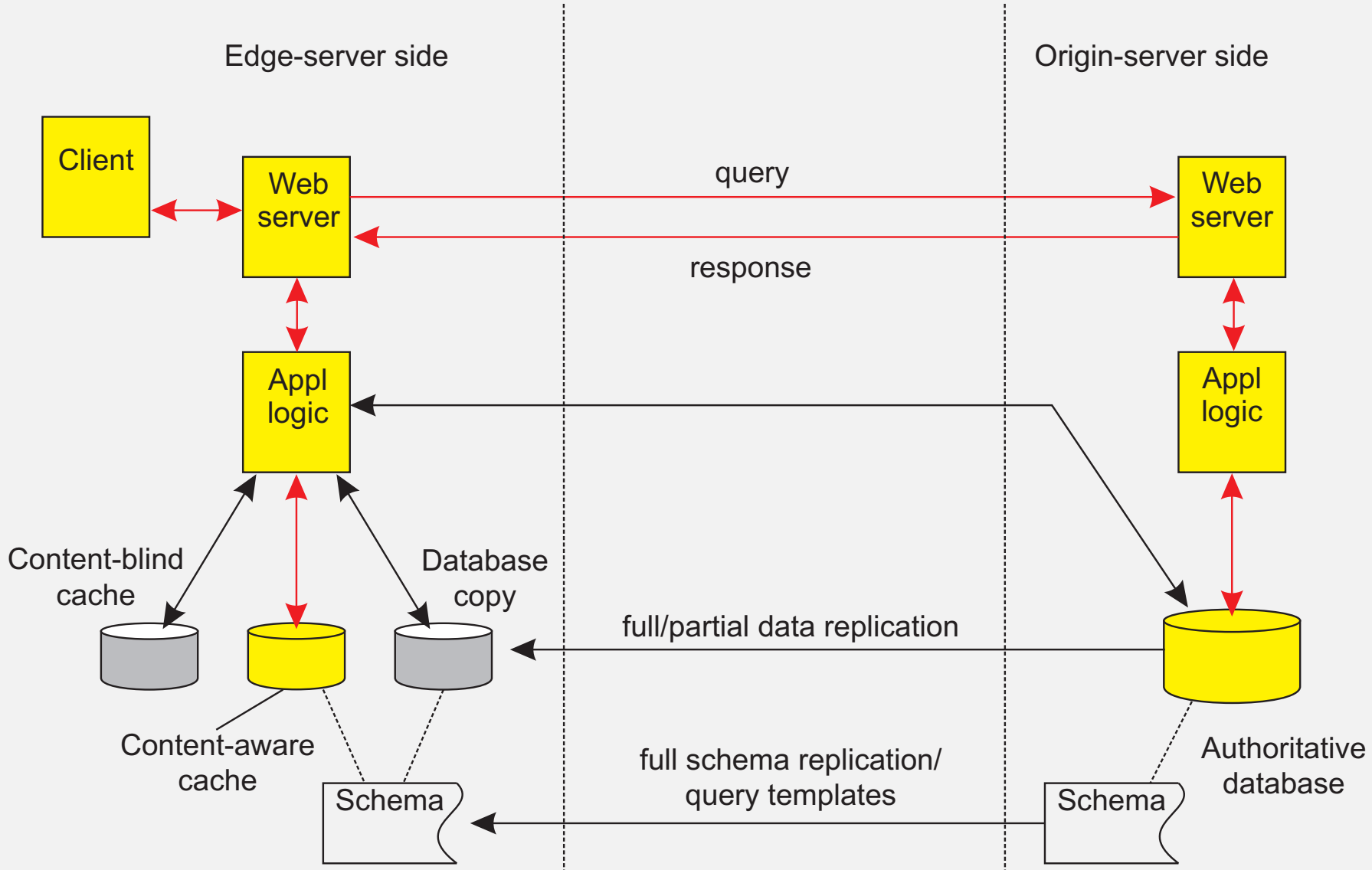
Question

What can be said about replication vs. performance?

Replication Web apps.: full/partial replication



Replication Web apps.: content-aware caching



Replication Web apps.: content-blind caching