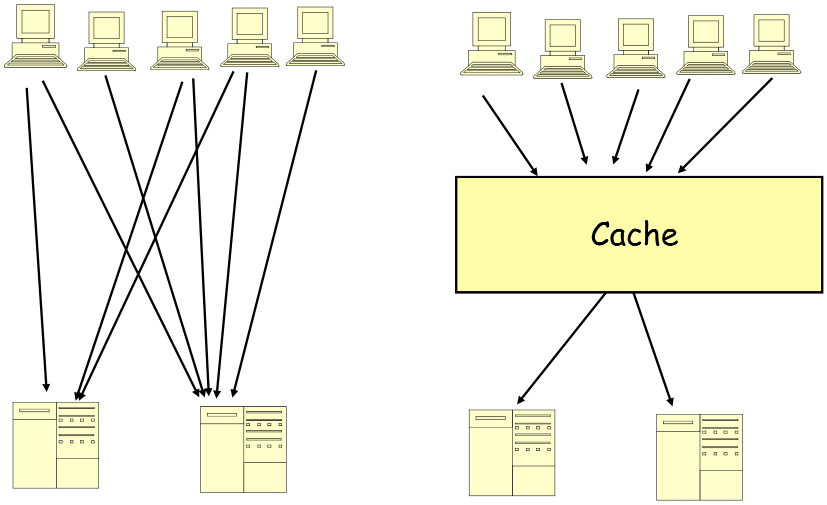
Caching

The Basic Idea

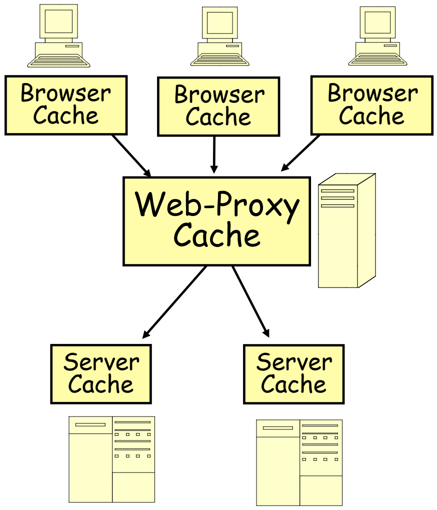


Cache

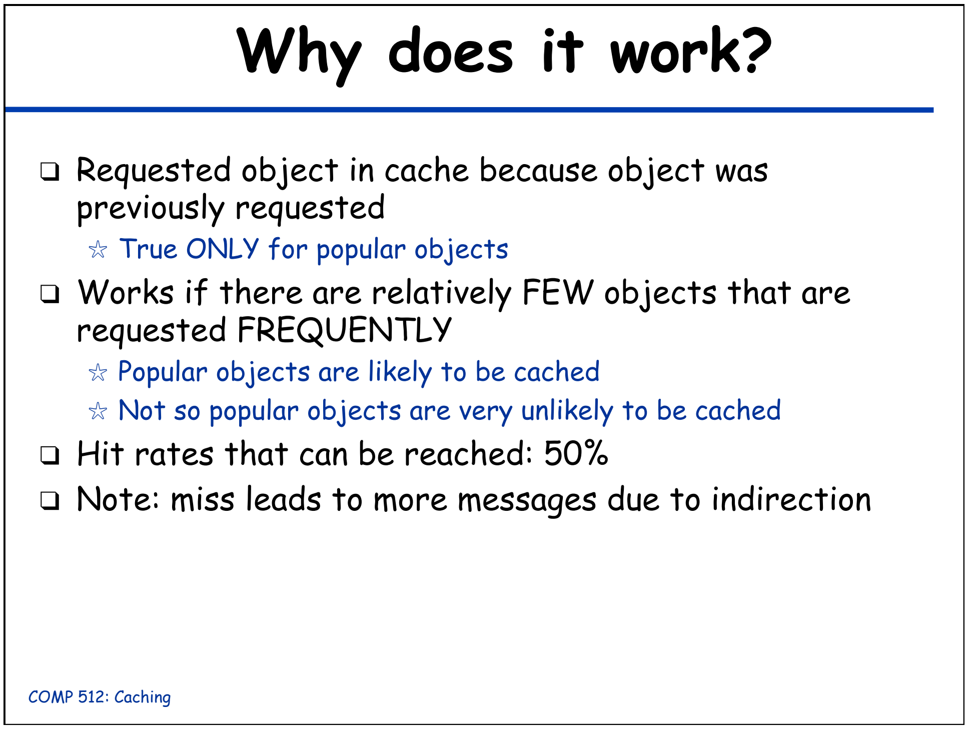
Why Web Caching?

- ❑ Client
  - ☆ Reduce Response Time
    - fast local access similar to replication
- ❑ Server:
  - ☆ Reduce Load
    - load distribution between cache and server
- ❑ Network:
  - ☆ Save bandwidth

Where to Cache?

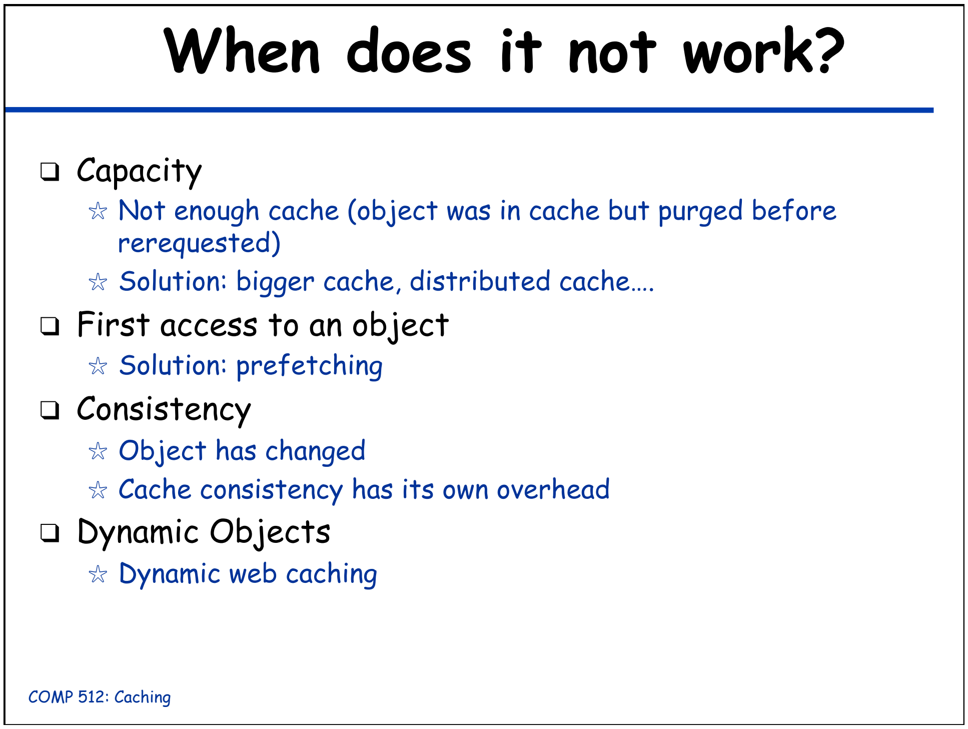


- ❑ Browser
  - ☆ Small: 10MBs memory, 100MB disk
    - Recursive caching (memory vs. disk)
    - 20% hit rate
- ❑ Organization (client-side proxy)
  - ☆ Large: GByte with disk
  - ☆ 50% hit rate
- ❑ Cache-Server Organization
  - ☆ Between Clients and Servers
- ❑ In front of server
  - ☆ Large: GByte
- ❑ Server itself (in memory)



Why does it work?

- ❑ Requested object in cache because object was previously requested
  - ☆ True ONLY for popular objects
- ❑ Works if there are relatively FEW objects that are requested FREQUENTLY
  - ☆ Popular objects are likely to be cached
  - ☆ Not so popular objects are very unlikely to be cached
- ❑ Hit rates that can be reached: 50%
- ❑ Note: miss leads to more messages due to indirection



When does it not work?

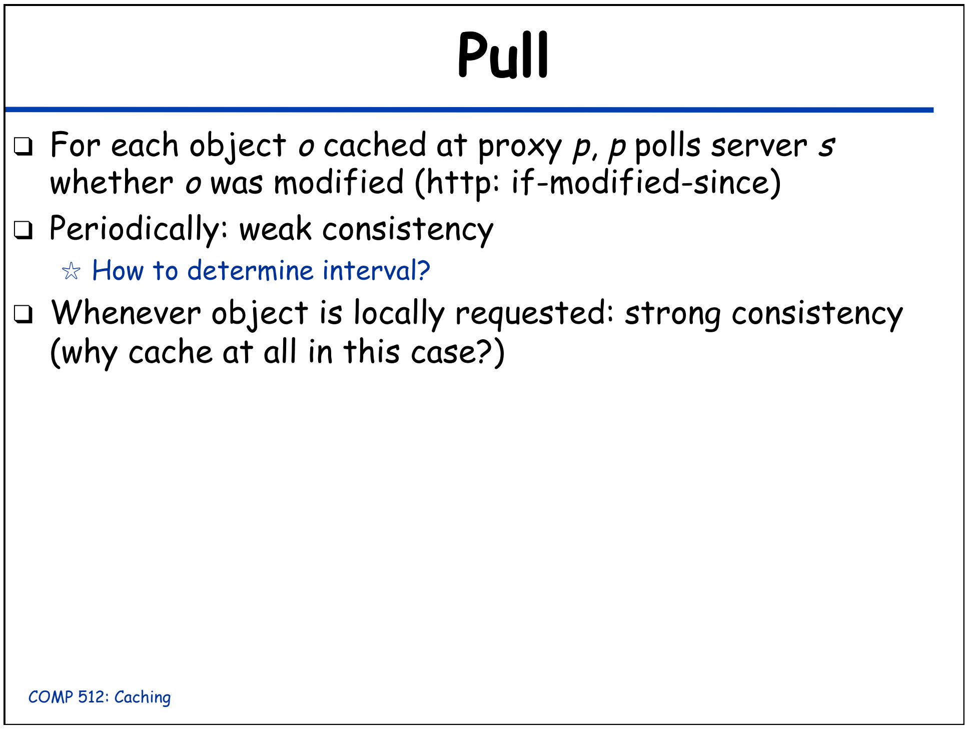
- ❑ Capacity
  - ☆ Not enough cache (object was in cache but purged before rerequested)
  - ☆ Solution: bigger cache, distributed cache….
- ❑ First access to an object
  - ☆ Solution: prefetching
- ❑ Consistency
  - ☆ Object has changed
  - ☆ Cache consistency has its own overhead
- ❑ Dynamic Objects
  - ☆ Dynamic web caching

Cache Consistency

- ❑ If object at server changes, cached copies become stale
- ❑ Cache consistency mechanisms
  - ☆ Server Push
  - ☆ Client Pull

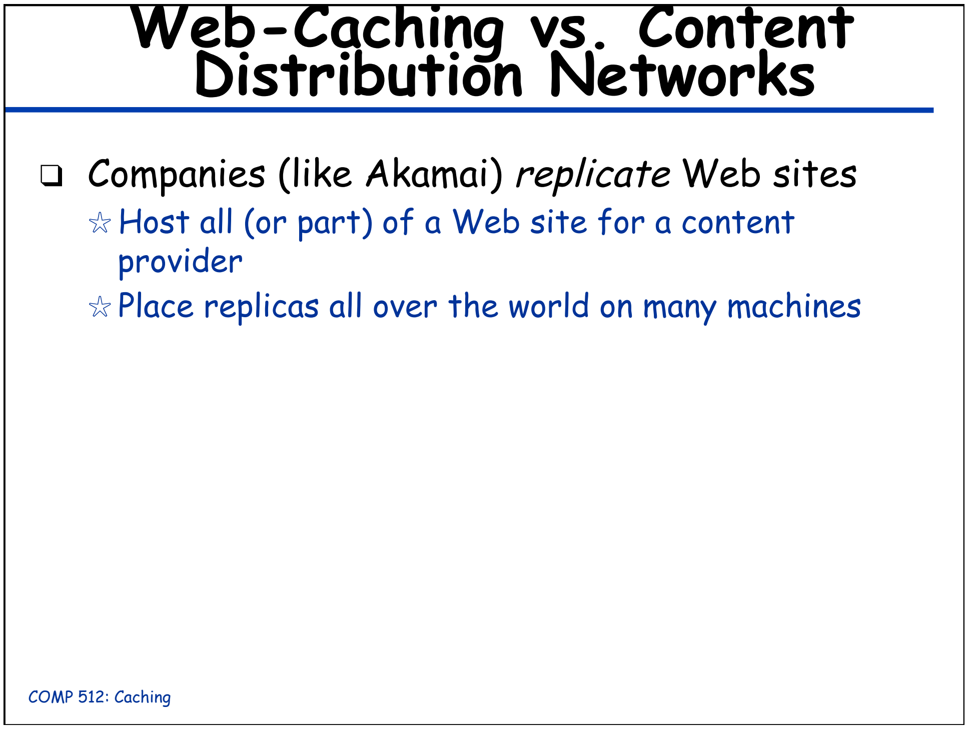
Push

- ❑ For each object *o* of server *s*, *s* keeps track of set P of proxies that have requested *o*
- ❑ When *o* is modified, notify all proxies in P
  - ☆ Invalidate
  - ☆ Update
  - ☆ Adv. / disadv.?
- ❑ Strong consistency
  - ☆ compare to strong consistency in context of replication



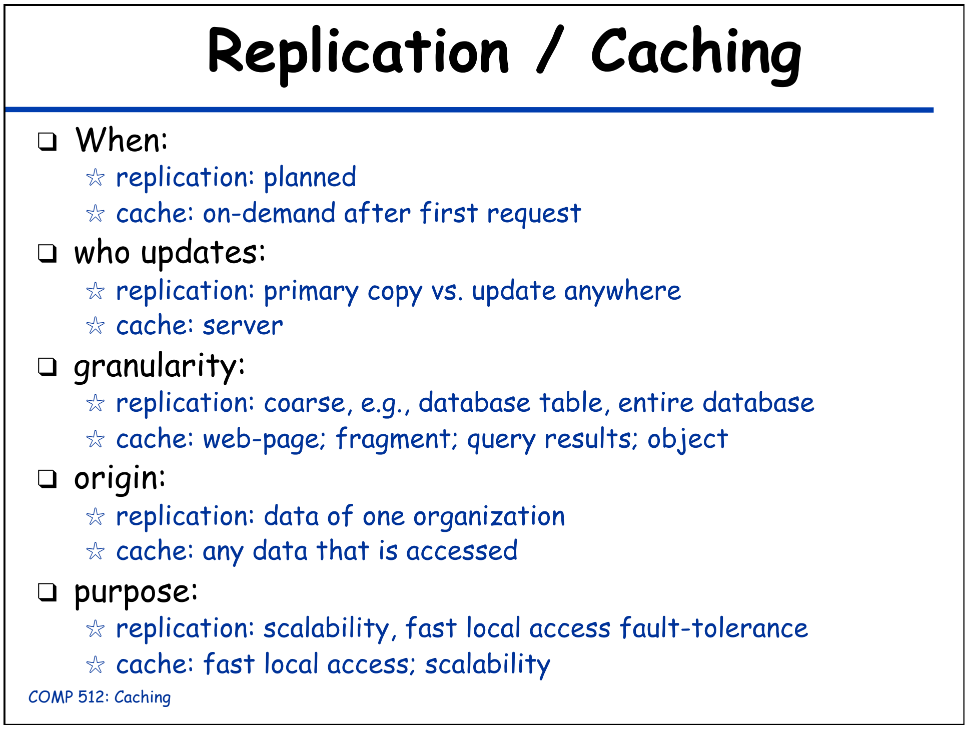
Pull

- ❑ For each object *o* cached at proxy *p*, *p* polls server *s* whether *o* was modified (http: if-modified-since)
- ❑ Periodically: weak consistency
  - ☆ How to determine interval?
- ❑ Whenever object is locally requested: strong consistency (why cache at all in this case?)



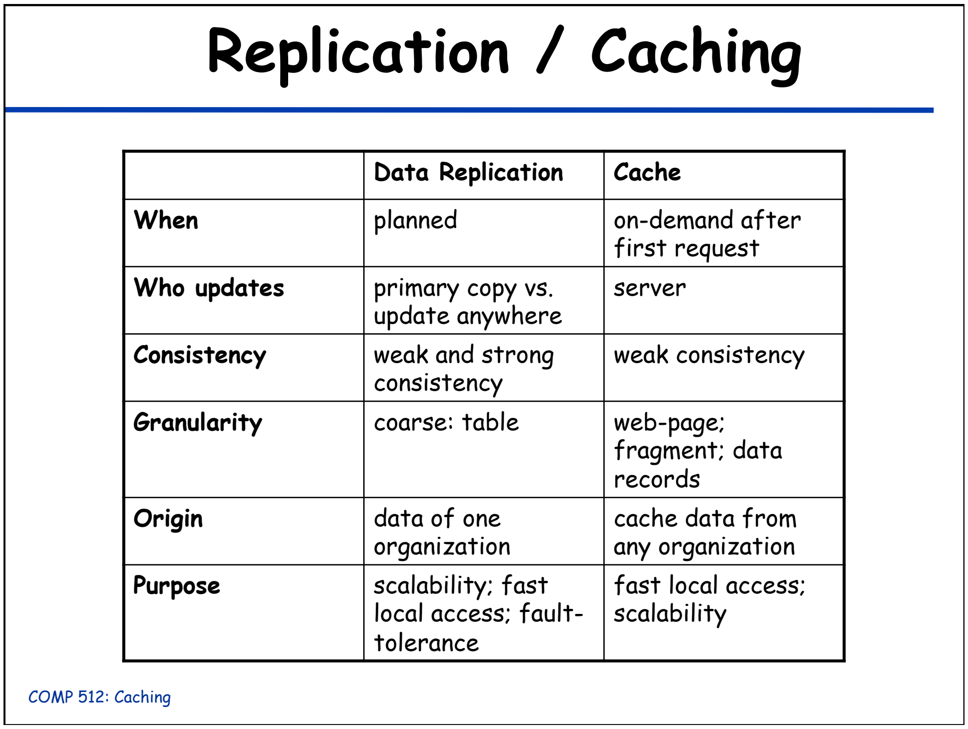
Web-Caching vs. Content Distribution Networks

- ❑ Companies (like Akamai) *replicate* Web sites
  - ☆ Host all (or part) of a Web site for a content provider
  - ☆ Place replicas all over the world on many machines



Replication / Caching

- ❑ When:
  - ☆ replication: planned
  - ☆ cache: on-demand after first request
- ❑ who updates:
  - ☆ replication: primary copy vs. update anywhere
  - ☆ cache: server
- ❑ granularity:
  - ☆ replication: coarse, e.g., database table, entire database
  - ☆ cache: web-page; fragment; query results; object
- ❑ origin:
  - ☆ replication: data of one organization
  - ☆ cache: any data that is accessed
- ❑ purpose:
  - ☆ replication: scalability, fast local access fault-tolerance
  - ☆ cache: fast local access; scalability



Replication / Caching

	Data Replication	**Cache**
When	planned	on-demand after first request
Who updates	primary copy vs. update anywhere	server
Consistency	weak and strong consistency	weak consistency
Granularity	coarse: table	web-page; fragment; data records
Origin	data of one organization	cache data from any organization
Purpose	scalability; fast local access; fault-tolerance	fast local access; scalability