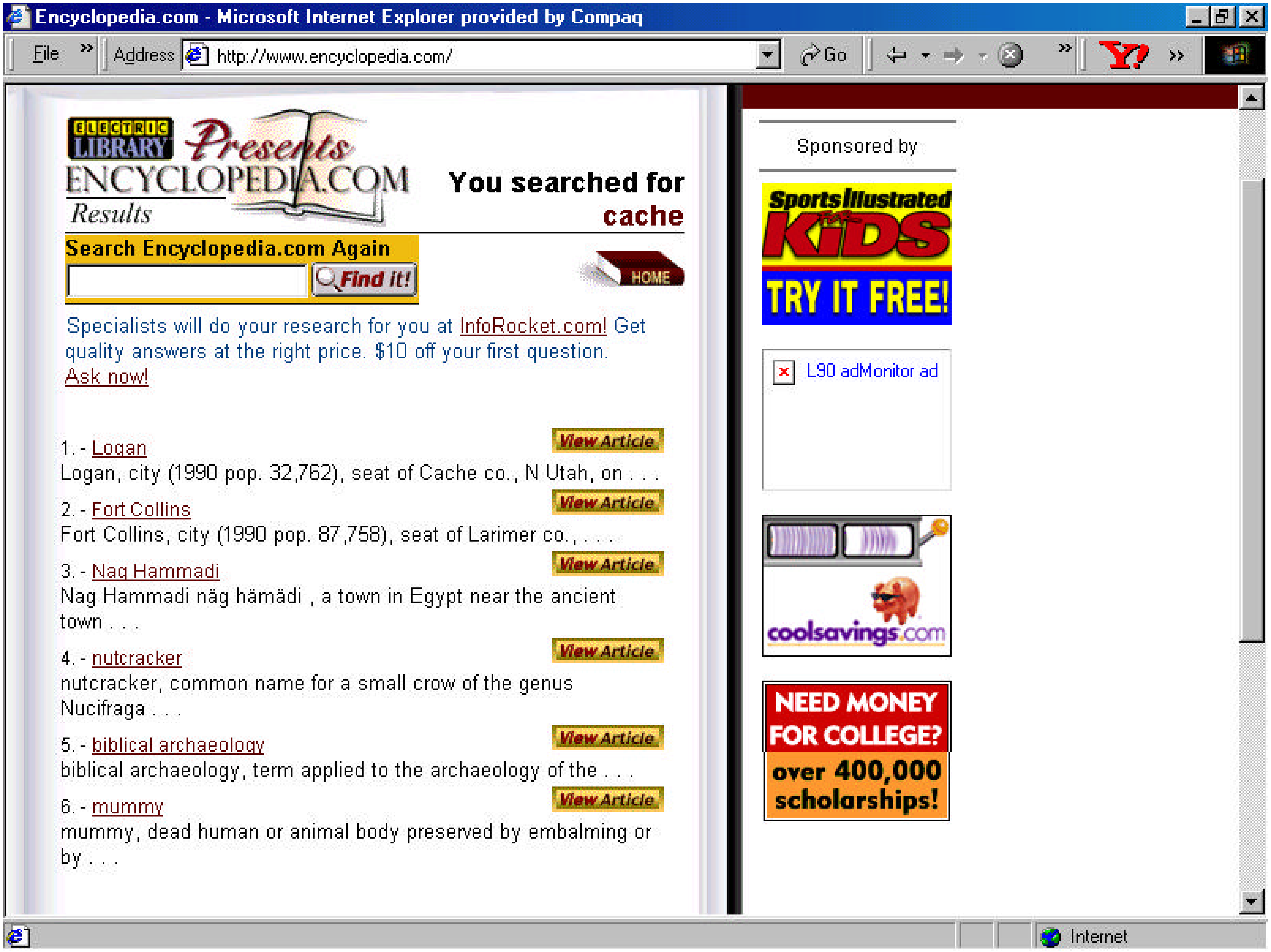
Consistency Control Algorithms for Web Caching

Leon Cao
University of Waterloo
February 28, 2001

What is a CACHE ?



Encyclopedia.com - Microsoft Internet Explorer provided by Compaq

File » Address http://www.encyclopedia.com/ Go

ELECTRIC LIBRARY Presents ENCYCLOPEDIA.COM Results

You searched for **cache**

Search Encyclopedia.com Again

Find it!

HOME

Specialists will do your research for you at InfoRocket.com! Get quality answers at the right price. $10 off your first question. Ask now!

1. - Logan View Article
Logan, city (1990 pop. 32,762), seat of Cache co., N Utah, on . . .

2. - Fort Collins View Article
Fort Collins, city (1990 pop. 87,758), seat of Larimer co., . . .

3. - Nag Hammadi View Article
Nag Hammadi näg hämädi , a town in Egypt near the ancient town . . .

4. - nutcracker View Article
nutcracker, common name for a small crow of the genus Nucifraga . . .

5. - biblical archaeology View Article
biblical archaeology, term applied to the archaeology of the . . .

6. - mummy View Article
mummy, dead human or animal body preserved by embalming or by . . .

Internet

cache

SYLLABICATION: cache

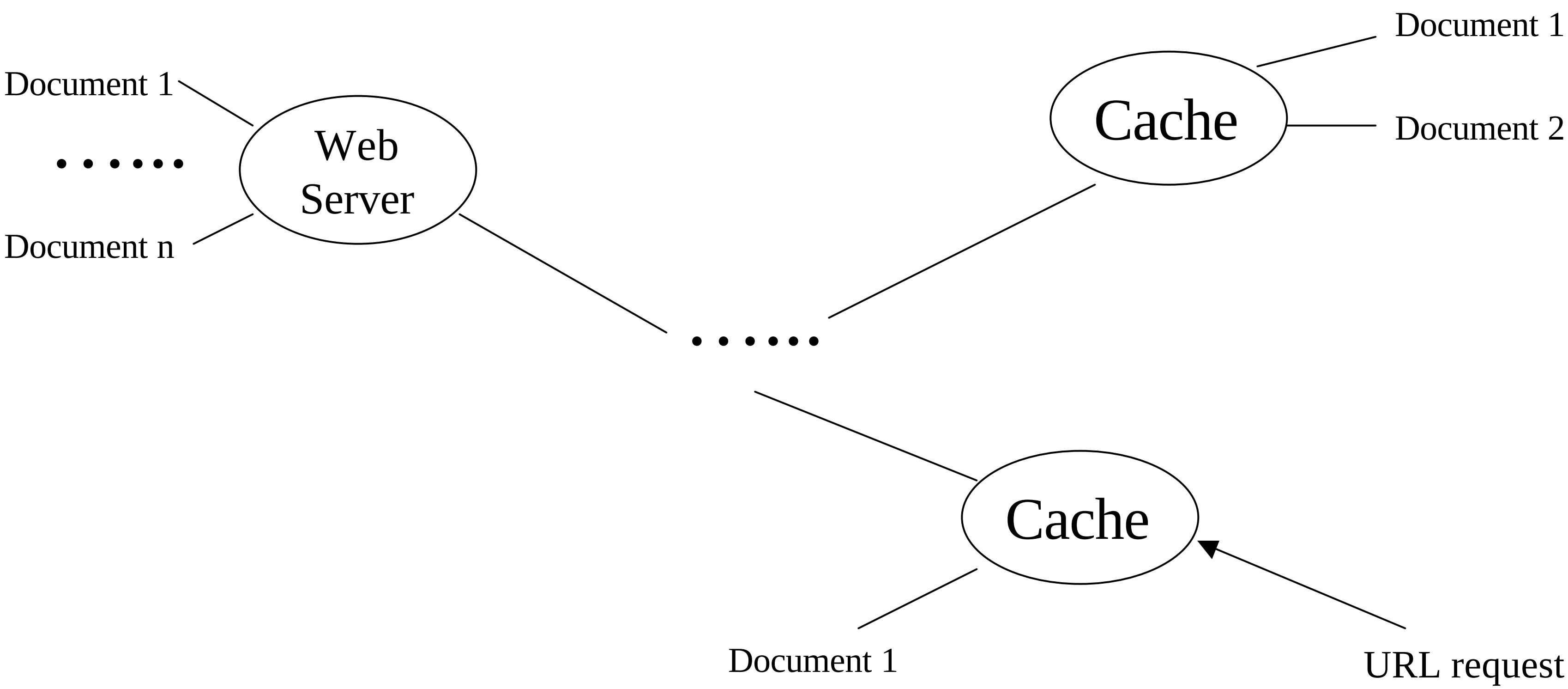
PRONUNCIATION: kăsh

NOUN: **1. a.** A hiding place used especially for storing provisions. **b.** A place for concealment and safekeeping, as of valuables. **c.** A store of goods or valuables concealed in a hiding place: "maintained a cache of food in case of emergencies." **2.** *Computer Science* A fast storage buffer in the central processing unit of a computer. Also called **cache memory** .

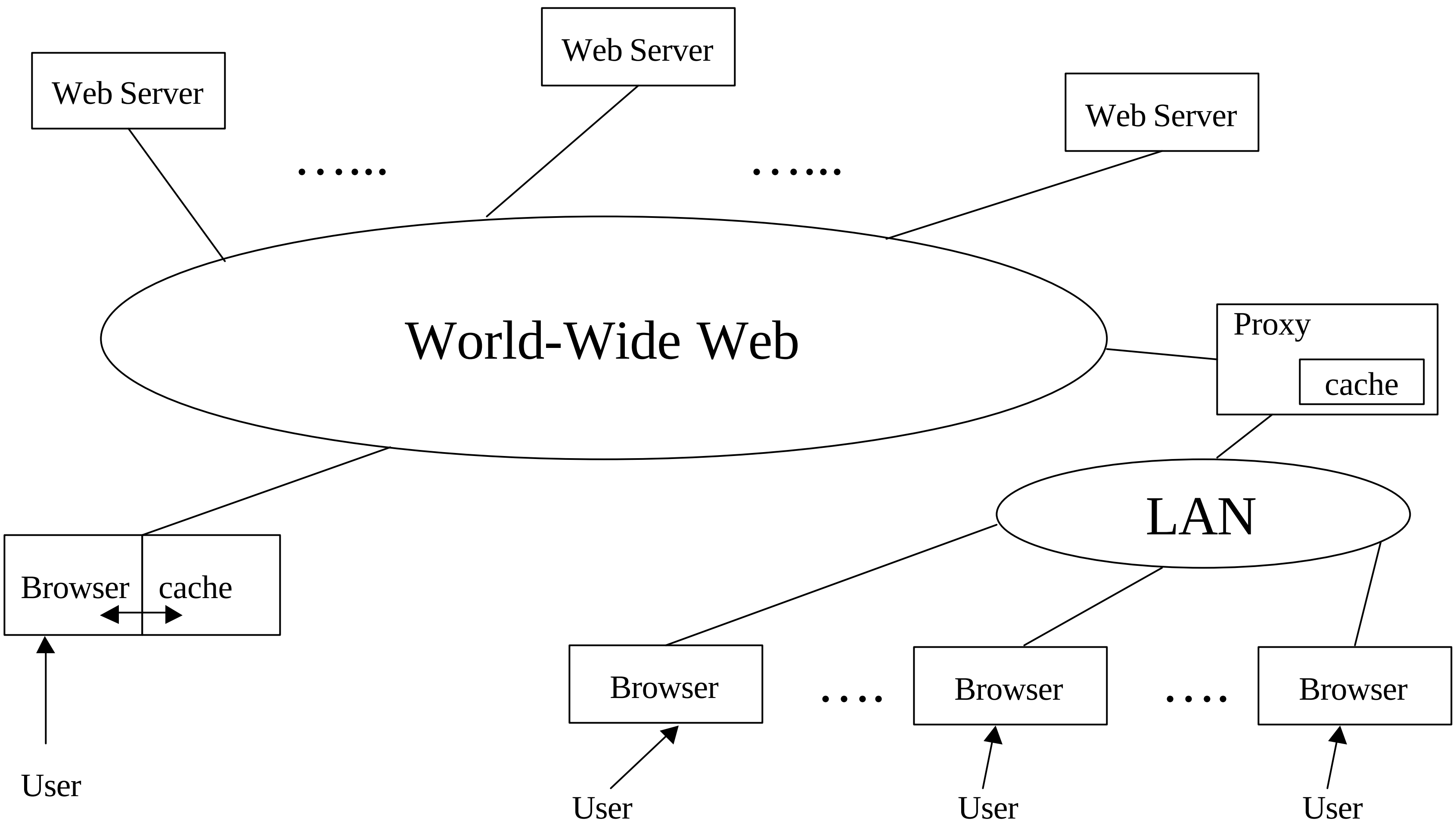
TRANSITIVE VERB: Inflected forms: **cached**, **cach·ing**, **cach·es**
To hide or store in a cache. See **hide[1]**.

ETYMOLOGY: French, from *cacher*, to hide, from Old French, to press, hide, from Vulgar Latin * *coāctĭcāre*, to store, pack together, frequentative of Latin *coāctāre*, to constrain, from *coāctus* past participle of cōgere, to force. See **COGENT**.

Generally, A Web cache checks if the requested information is available in its local storage, if so, a reply is sent back to the user with the requested data; otherwise the cache forwards the request on behalf of the user to either another cache or to the original server.



There are two basic types of Web cache: *browser cache* and *proxy cache*.



Advantages of Web Caching

- Reduced network bandwidth consumption
- Reduced server load
- Reduced client latency
- Sometimes more reliability

Disadvantages of Web Caching

- Potential of stale data access
- Increases latency on requests for non-cached pages
- Increases local administrative complexity and cost for disk space
- Online advertising is unable to know how many times a certain page has been viewed

Why cache consistency algorithms?

- By introducing caching mechanism, multiple copies of a same object are created and stored in various caches all over the Internet. How to keep them consistent? How to ensure the data user accesses is always valid?
- The value of cache is greatly reduced if cached copies are not updated when the original data change.
- Cache consistency algorithms ensure that cached copies of data are eventually updated to keep consistency with the original data.
- An ideal cache consistency solution will enforce the consistency to the maximum extent, while reducing the network bandwidth consumption and server load.
- There are basically two categories of cache consistency approaches: *weak cache consistency* and *strong cache consistency*.

Weak Cache Consistency

- Under weak cache consistency algorithm, it is possible for the user to get a stale document from the cache, because the cache only validates the document's freshness with the server periodically so as to reduce network bandwidth and server workload.
- TTL (Time-To-Live) and Client Polling are two algorithms that fall in to this category

TTL (Time-To-Live)

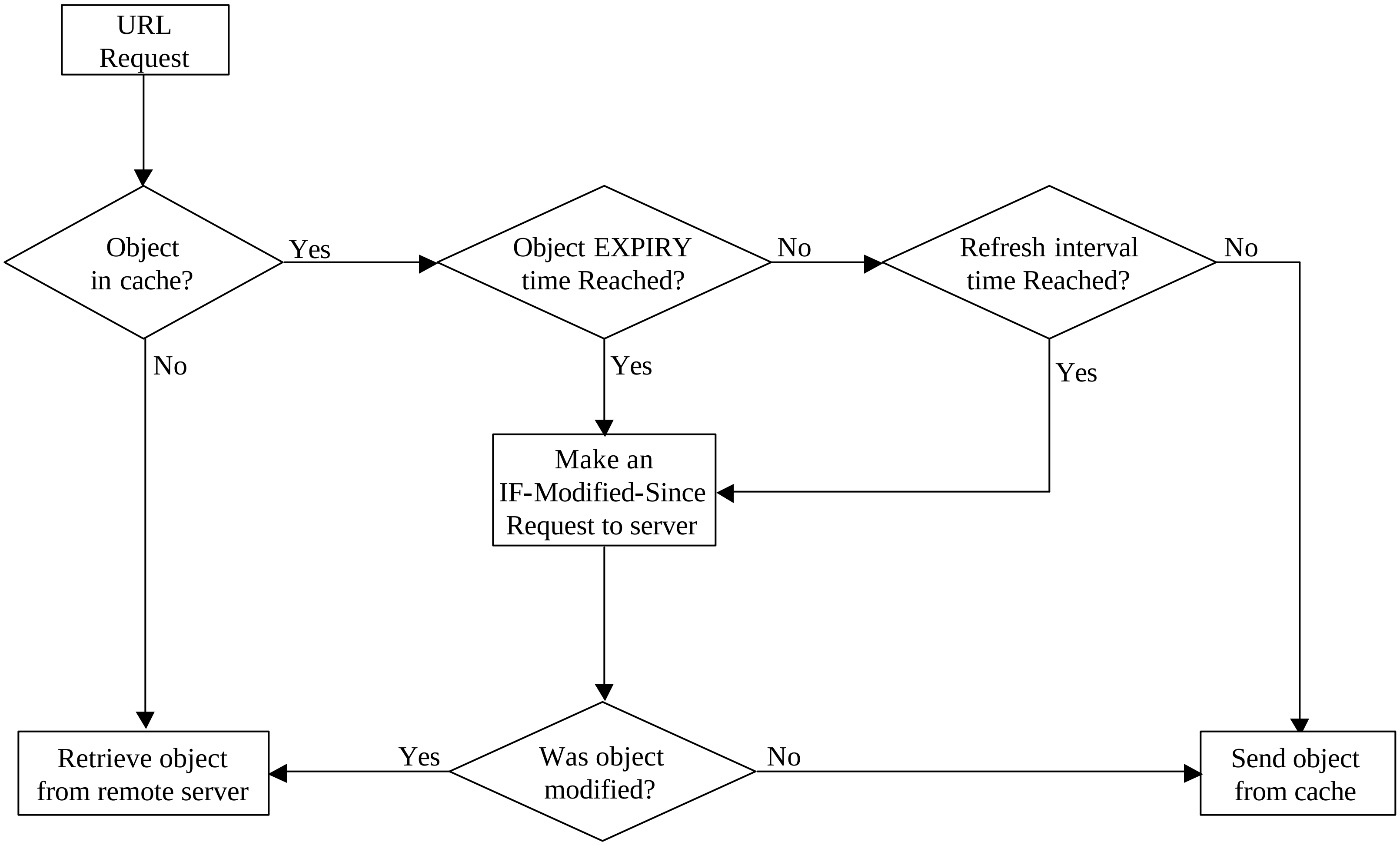
- Under TTL approach, each object is assigned a time-to-live value, which is an estimate of the object's lifetime, after which its supposed to change.
- When the TTL expires, the data is considered invalid, and the next request for the object will cause the object to be requested from the original server.
- TTL -based strategies are easy to implement, by using the "expires" header field in HTTP format. Following is an example of an HTTP header that applies the "expires" field:

```
HTTP/1.1 200 OK
Date: Fri, 09 Feb 2001 10:19:29 GMT
Server: Apache/1.3.3 (Unix)
Cache-Control: max-age=3600, must-revalidate
Expires: Fri, 09 Feb 2001 11:19:29 GMT
Etag: "3e86-410-3596fbbc"
Content-Length: 1040
Content-Type: text/html
…
```

- The challenge in supporting this approach lies in selecting an appropriate TTL value.

Client Polling

- Under this approach, the client (cache) periodically checks back with the server to determine if cached objects are still valid.
- A typical algorithm is called Update Threshold. The update threshold is expressed as a percentage of the object's age.
- For example, consider a cached file whose age is 30 days and the update threshold is set to 10%.



CERN Proxy Cache logic

Summary

- We could see from the introduction of weak cache consistency that weak consistency control algorithms save network traffic and user latency at the expense of returning stale documents to the server.
- Weak cache consistency is an economic approach user situations where document modification doesn’t happen very frequently, or user doesn’t have strict requirement on the freshness of the document.
- However, if the validity of the data is important (e.g. weather forecast), weak cache consistency is not applicable. A strong consistency algorithm has to be applied.

Invalidation

- The Web server is responsible for keeping track of the copy of data.
- Once the data is modified on the server, the server sends out invalidation message to all those caches that keep the copy.
- Invalidation guarantees document freshness.

Polling-Every-Time

- Once the cache receives request from end-user, it polls the server to confirm if the data it caches is still fresh, therefore also guarantees freshness.
- Potentially there will be a lot of message transfers.
- Given a short document lifetime and frequent requests from the user, this is feasible.

Trace	SASK, 51471 requests		
Modification	1148 files modified		
Approach	TTL	Polling	Invalidation
Hits	16456	16565	16268
Get Requests	35015	34906	35203
If-Modified-Since	922	16565	0
Reply 200	35388	35689	35203
Reply 304	549	15782	0
Invalidations	0	0	6028
Total Messages	71874	102942	76434
File Xfer bytes	185MB	187MB	183MB
Ctrl Msg bytes	3.91MB	7.09MB	4.29MB
Messages bytes	189MB	194MB	187MB
Stale Hits	< 410	0	0
Avg. Latency	0.124	0.138	0.134
Min Latency	0.010	0.039	0.010
Max Latency	32.1	12.2	107
Server CPU	26.0%	30.2%	27.6%
DISK RW/s	.37;2.2	.41;2.3	.41;2.5

Trace	SDSC, 25430 requests		
Modification	57 files modified		
Approach	TTL	Polling	Invalidation
Hits	4907	4907	4905
Get Requests	20523	20523	20525
If-Modified-Since	239	4907	0
Reply 200	20535	20549	20525
Reply 304	227	4881	0
Invalidations	0	0	248
Total Messages	41524	50860	41298
File Xfer bytes	263MB	263MB	263MB
Ctrl Msg bytes	2.39MB	3.38MB	2.36MB
Messages bytes	265MB	266MB	265MB
Stale Hits	< 14	0	0
Avg. Latency	0.16	0.173	0.165
Min Latency	0.010	0.038	0.010
Max Latency	12.2	12.2	12.2
Server CPU	34.1%	35.6%	32.7%
DISK RW/s	.94;2.3	1.4;2.0	1.0;2.2

Results from “Maintaining Strong Cache Consistency in World-Wide Web” by P. Cao & C. Liu

Limitations

- Due to space limitation, some of the experiments in the research papers are performed in a local area network instead of the Internet.
- The problem with update threshold is how to decide the individual update threshold value for each document.
- Invalidation approaches are often expensive.
- Another problem with invalidation is how to deal with failures.

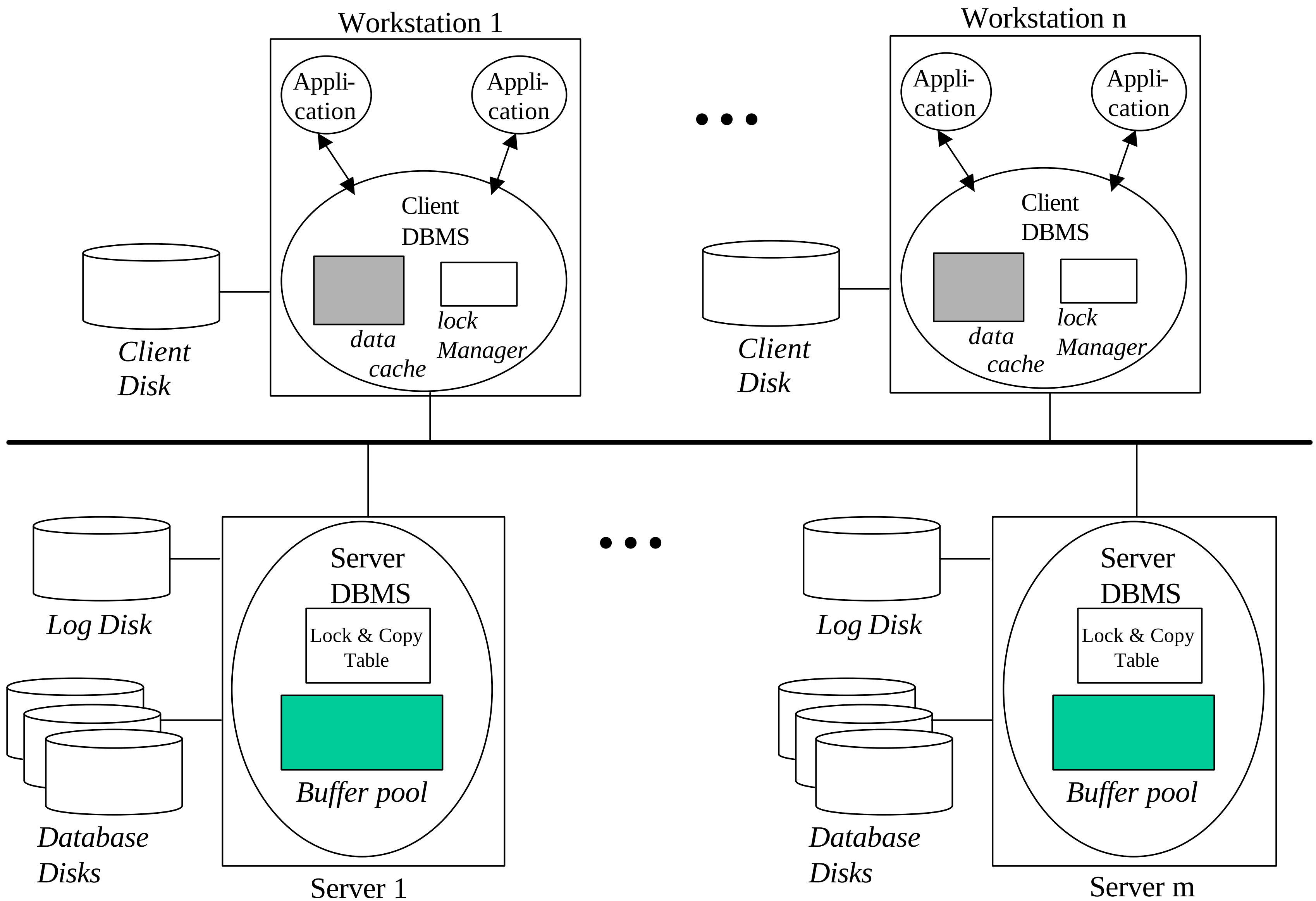
Improvements

- Add some invalidation function to the server while implementing adaptive TTL.
- Two-tier-lease-augmented invalidation algorithm ([3]):
  - A “lease” field is added to all the documents sent from the server to a client cache
  - Server promises to notify the client if the document changes before the lease expires
  - Client promises to send an “if-modified-since” message to the server once the lease expires and the client still wants to keep the document
  - For regular “get-object” request, the server assigns a very short lease value (e.g. 0) and a regular lease to “if-modified-since” requests
- Pre-fetching could also be used to reduce the stale rate.

Now let’s take a look at...

cache consistency in transactional Client/Server environment

Reference architecture for a data-shipping client/server DBMS



- Most cache consistency algorithms in client/server architecture could be categorized into detection-based or avoidance-based, depending on the choice of Invalid Access Prevention.
- Algorithms that use avoidance for invalid access prevention ensure that at any time, all cached data is up-to-date; those that use detection allow stale data to remain in client caches and ensure that transactions are allowed to commit only if it can be verified that they have not accessed such stale data.
- Transactional cache consistency algorithms must ensure that no transactions that access stale data are allowed to commit.
- [1] presented a taxonomy that partitions consistency control algorithms into two classes according to whether their approach to preventing stale data access is detection-based or avoidance-based.

- Detection-based algorithms allow stale data copies to reside in a client’s cache for for some time.
- There are three levels of differentiation in the detection-based side of the taxonomy:
    - Validity Check Initiation
        - Synchronous
        - Asynchronous
        - Deferred
    - Change Notification Hints
        - Optimistic
        - Pessimistic
    - Remote Update Action
        - Propagation
        - Invalidation
        - Dynamically choosing

- Avoidance-based algorithms enforce cache consistency by making it impossible for transactions to access stale data in their local cache.
- These algorithms use a read-one/write-all (ROWA) approach to replica management, which ensures that all existing copies of an updated item have the same value when an updating transaction commits.
- There are 4 levels in the avoidance-based half of the taxonomy:
    - Write Intention Declaration
        - Synchronous
        - Asynchronous
        - Deferred
    - Write Permission Duration
        - One particular transaction
        - Span multiple transactions
    - Remote Conflict Priority
        - Wait
        - Preempt
    - Remote Update Action

Conclusions

- Cache is also used to keep mummies...
- A good cache consistency algorithm is essential to reduce client latency as well as bandwidth requirement for delivering web contents.
- Taking all factors into consideration, a really good consistency control algorithm is hard to find.
- One possible solution to this is to combine the advantage of each algorithm. For example, add invalidation mechanism to server while implementing adaptive TTL at client cache, or use asynchronous message transfer to reduce message block overheads, while enforce strong cache consistency.
- A lot of work to do...

Main References

[1] Michael J. Franklin, Michael J. Carey and Miron Livny. Transactional client-server cache consistency: Alternatives and performance. *ACM Transaction on Database Systems*, 1997.

[2] James Gwertzman and Margo Seltzer. World-Wide Web Cache Consistency. *International Conference USENIX*, San Diego, CA, 1996

[3] Pei Cao and Chengjie Liu. Maintaining Strong Cache Consistency in the World-Wide Web. *Proceedings of the 17th International Conference on Distributed Computing Systems (ISDCS '97)*, 1997

[4] Duane Wessels. Intelligent Caching for World-Wide Web Objects. *International Conference of the Internet Society (INET)*, Honolulu, HI, 1995