1 Title

WAN data replication

2 Byline

Maarten van Steen
Vrije Universiteit Amsterdam
http://www.cs.vu.nl/∼steen/

3 Synonyms

Wide-area data replication

4 Definition

The field of WAN data replication covers the problems and solutions for distributing and replicating data across wide-area networks. Concentrating on databases alone, a wide-area database is defined as a collection of multiple, logically interrelated databases distributed and possibly replicated across sites that are connected through a wide-area network.

The characteristic feature here is that data are spread across sites that are separated through wide-area links. Unlike links in local-area networks, the quality of communication through wide-area links is relatively poor. Links are subject to latencies of tens to thousands of milliseconds, there are often severe bandwidth restrictions, and connections between sites are much less reliable.

In principle, WAN data replication also covers the distribution and replication of plain files. These issues are traditionally handled by wide-area distributed file systems such as AFS [11] and NFS [3, 12], which are both widely used. These distributed file systems aim at shielding data distribution from applications, i.e., they aim at providing a high degree of distribution transparency. As such, they tackle the same problems that wide-area databases need to solve. However, matters are complicated for databases, because relations *between* and *within* files (i.e., tables) need to be taken into account as well.

5 Historical background

WAN data replication is driven by the need for improving application performance across wide-area networks. Performance is generally expressed in terms of client-

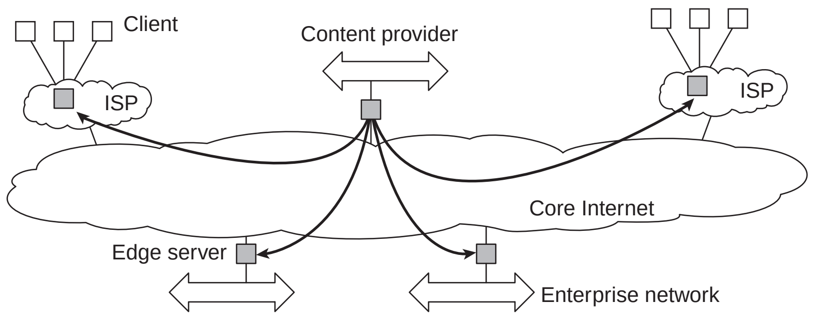

Figure 1: Web-based data replication deploying servers that are placed at the edge of the Internet. *Adapted from Tanenbaum, van Steen: "Distributed Systems" (2nd ed.), Prentice-Hall, 2007.*

perceived quality of service: response times (latency), data transfer rates (bandwidth), and availability. Replication may also be driven by the requirement for reducing monetary costs, as the infrastructure over which data distribution and replication takes place is generally owned by a separate provider who may be charging per transferred byte.

Replication of data has been explored in early wide-area systems such as Grapevine [2], Clearinghouse [4], and Lotus Notes [6]. All these systems concentrated on achieving a balance between data consistency, performance, and availability, recognizing that acceptable quality of service can be achieved only when weak consistency can be tolerated. In general, this required exploring application semantics making solutions more or less specific for an application.

In the mid-90s, researchers from Xerox explored client-centric consistency models by which data were distributed and replicated in a wide-area database such that a notion of strong data consistency could be presented to users individually [8]. For example, data that had been modified by a user when accessing the database at location *A* would be propagated to location *B* before the user would access the system again at *B*, but not to other locations.

The need for WAN data replication became widely recognized with the explosion of the Web, which instantly revealed the shortcomings of its traditional client-server architecture. Up to date, virtually all Web sites are centrally organized, with end users sending requests to a single server. However, this organization does not suffice for large commercial sites, for which high performance and availability is crucial. To address these needs, so-called *Content Delivery Networks* (CDNs) came into play. A CDN is essentially a Web hosting service with many servers placed across the Internet (see Figure 1). Its main goal is to ensure that a single Web site is automatically distributed and replicated across these servers in such a way that negotiated performance and availability requirements are met (see also [9]).

CDNs surfaced when most Web sites were still organized as a (possibly very large) collection of files that could be accessed through a single server. Modern sites, however, are no longer statically organized, but deploy full-fledged databases from which Web content is dynamically generated by application servers. As a consequence, the so-called edge servers to which client requests are initially directed are gradually turning into servers hosting partially or fully replicated databases, or database caches. These issues are discussed below.

6 Scientific fundamentals

A popular model used to understand the various issues involved in wide-area data(base) is the one in which every data item has a single associated server through which all its updates are propagated. Such a primary or origin server as it is called, simplifies the handling of conflicts and maintenance of consistency. In practice, an origin server maintains a complete database that is partially or completely replicated to other servers. In contrast, in an *update anywhere* approach updates may be initiated and handled at any replica server. The main problem with this approach is that it requires global consensus among the replica servers on the ordering of updates if strong consistency is to be preserved. Achieving such consensus introduces serious scalability problems, for which reason various optimistic approaches have been proposed (optimistic in the sense that corrective actions may later be necessary). These are discussed in detail in the entry "Optimistic replication and resolution."

Queries are initially forwarded to edge servers, which then handle further processing. This could mean that subqueries are issued to different origin servers, but it is also possible that the edge server can compute the answer locally and send the response directly to the requesting client without further contacting the origin. In this context, key issues that need to be addressed for wide-area data replication are replica placement and consistency.

6.1 Replica placement

Somewhat surprisingly, many researchers do not make a clear distinction between placement of server machines and placement of data on servers. Nevertheless, this distinction is important: where data placement can be often be decided at runtime, this is obviously not the case for placement of server machines. Moreover, the criteria for placement are different: server placement should be done for many data objects, but deciding on the placement of data can be optimized for individual data objects.

Both problems can be roughly tackled as optimization problems in which the best K out of N possible locations need to be selected. There are a number of variations of this problem, but important is the fact that heuristics need to be employed due to the exponential complexity of known solutions. For this reason, runtime data placement decisions deploy simpler solutions. An overview is provided in [14].

Relevant in this context is comparing different placements to decide which one is best. To this end, a general cost function can be used in which various metrics are combined:

$$cost = w_1 \cdot res_1 + w_2 \cdot res_2 + \ldots w_n \cdot res_n, res_k \geq 0, w_k > 0$$

where res_k is a monotonically increasing varaiable denoting the cost of resource k and w_k its associated weight. Typical resources include distance (expressed in delay or number of hops) and bandwidth. Note that the actual dimension of *cost* is irrelevant. What matters is that different costs can be compared in order to select the best one.

Using a cost-driven placement strategy also implies that resource usage must be measured or estimated. In many cases, estimating costs may be more difficult than one would initially expect. For example, in order for an origin server to estimate the delay between a client and a given edge server may require mapping Internet locations to coordinates in a high-dimensional geometric space [7].

6.2 Data consistency

Consistency of replicated data has received considerable attention, notably in the context of distributed shared-memory parallel computers. This class of computers attempts to mimic the behavior of traditional shared-memory multiprocessors on clusters or grids of computers. However, traditional distributed systems have often generally assumed that only strong consistency is acceptable, i.e., all processes concurrently accessing shared data see the same ordering of updates everywhere. The problem with strong consistency is that it requires timely global synchronization, which may be prohibitively expensive in wide-area networks. Therefore, weaker consistency models often need to be adopted.

To capture different models, Yu and Vahdat defined *continuous consistency* [15], a framework that captures consistency along three different dimensions: time, content, and ordering of operations. When updates are handled in a centralized manner, then notably the first two are relevant for WAN data replication. Continuous time-based consistency expresses to what extent copies of the same data are allowed to be stale with respect to each other. Continuous content-based consistency is used to express to what extent the respective *values* of replicated data may differ, a metric that is useful when dealing with, e.g., financial data. The differences in consistency are subsequently expressed as numbers. By exchanging this information between sites, consistency enforcement protocols can be automatically started without further interference from applications.

There are essentially three ways to bring copies in the same state. First, with *state shipping* the complete up-to-date state is transferred to a replica server. This update form can be optimized through *delta shipping* by which only the difference between a replica's current state and that of a fresher replica is computed and transferred. These two forms of update are also referred to as passive replication, or asymmetric update processing In contrast, with active replication, *function shipping* takes place, meaning that the operations that led to a new state are forwarded to a replica and subsequently executed. This is also known as *symmetric update processing*.

Differences may also exist with respect to the server taking the initiative for being updated. In *pull* approaches a replica server sends a request to a fresher replica to send it updates. In the *push* approach, updates are sent to a replica server at the inititative of a server that has just been updated. Note that for pushing, the server taking the initiative will need to know every other replica server as well as its state. In contrast, pulling in updates requires only that a replica server knows where to fetch fresh state. For these reasons, pull-based consistency enforcement is often used in combination with caches: when a request arrives at a caching server, the latter checks whether its cache is still fresh by contacting an origin server.

An important observation for WAN data replication is that it is impossible to simultaneously provide strong consistency, availability, and partition tolerance [5]. In the edge-server model, this means that despite the fact that an origin server is hosting a database offering the traditional ACID properties, the system as a whole cannot provide a solution that guarantees that clients will be offered continuous and correct service as long as at least one server is up and running. This is an important limitation that is often overlooked.

7 Key Applications

Wide-area data(base) replication is applied in various settings, but is arguably most prevalent in Web applications and services (see also [1]). To illustrate, consider a Web service deployed within an edge-server infrastructure. The question is what kind of replication schemes can be applied for the edge servers. The various solutions are sketched in Figure 2.

First, full replication of the origin server's database to the edge servers can take place. In that case, queries can be handled completely at the edge server and problems evolve around keeping the database replica's consistent. Full replication is generally a feasible solution when read/write ratios are high and queries are complex (i.e., spanning multiple database tables). A main problem is that if the number of replicas grow, one would need to resort to lazy replication techniques by which an edge server can continue to handle a query in parallel to bringing all replicas up-to-date, but at the risk of having to reconcile conflicting updates later on [10]. If queries and data can be organized such that access to only a single table is required (effectively leading to only simple queries), partial replication by which only the relevant table is copied to the edge server may form a viable optimization.

An alternative solution is to apply what is known as content-aware caching. In this case, an edge server has part of the database stored locally based on caching techniques. A query is assumed to fit a template allowing the edge server to efficiently cache and lookup query results. For example, a query such as *"select * from items where price $<$ 50"* contains the answer to the more specific query *"select * from items where price $<$ 20."* In this case, the edge server is actually building up its own version of a database, but now with a data schema that is strongly related to the structure of queries.

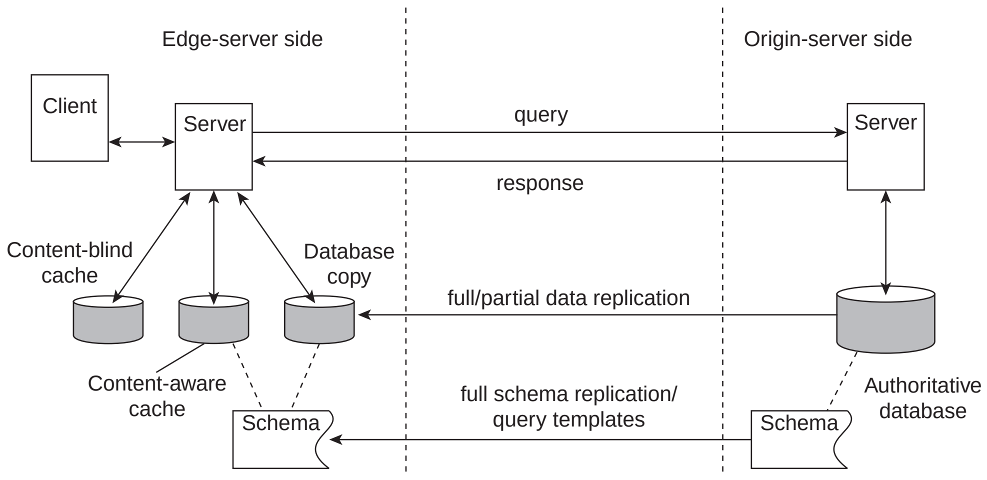

Figure 2: Caching and replication schemes for Web applications. *Adapted from Tanenbaum, van Steen: "Distributed Systems" (2nd ed.), Prentice-Hall, 2007.*

Finally, an edge server can also deploy content-blind caching by which query results are simply cached and uniquely associated with the query that generated them. In other words, unlike content-aware caching and database replication, no relationship with the structure of the query or data is kept track of.

As discussed in [13], each of these replication schemes has its merits and disadvantages, with no clear winner. In other words, it is not possible to simply provide a single solution that will fit all applications. As a consequence, distributed systems that aim to support WAN data replication will need to incorporate a variety of replication schemes if they are to be of general use.

8 Cross references

1-COPY SERIALIZABILITY
AUTONOMOUS REPLICATION
CONSISTENCY MODELS FOR REPLICATED DATA
DATA REPLICATION
DISTRIBUTED DATABASES
EVENTUAL CONSISTENCY
OPTIMISTIC REPLICATION AND RESOLUTION
PARTIAL REPLICATION
REPLICA CONTROL
REPLICATION BASED ON GROUP COMMUNICATION
REPLICATION FOR HIGH AVAILABILITY

Recommended reading

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju. *Web Services: Concepts, Architectures and Applications*. Springer-Verlag, Berlin, 2004.

[2] A. Birrell, R. Levin, R. Needham, and M. Schroeder. "Grapevine: An Excercise in Distributed Computing." *Communications of the ACM*, 25(4):260–274, Apr. 1982.

[3] B. Callaghan. *NFS Illustrated*. Addison-Wesley, Reading, MA., 2000.

[4] A. Demers, D. Greene, C. Hauser, W. Irish, J. Larson, S. Shenker, H. Sturgis, D. Swinehart, and D. Terry. "Epidemic Algorithms for Replicated Database Maintenance." In *Proc. 6th Symposium on Principles of Distributed Computing*, pp. 1–12, Aug. 1987. ACM.

[5] S. Gilbert and N. Lynch. "Brewer's Conjecture and the Feasibility of Consistent, Available, Partition-tolerant Web Services." *ACM SIGACT News*, 33(2):51–59, June 2002.

[6] L. Kawell, S. Beckhardt, T. Halvorsen, R. Ozzie, and I. Greif. "Replicated Document Management in a Group Communication System." In *Proc. 2nd Conference Computer-Supported Cooperative Work*, pp. 226–235, Sept. 1988.

[7] E. Ng and H. Zhang. "Predicting Internet Network Distance with Coordinates-Based Approaches." In *Proc. 21st INFOCOM Conference*, June 2002. IEEE Computer Society Press, Los Alamitos, CA.

[8] K. Petersen, M. Spreitzer, D. Terry, and M. Theimer. "Bayou: Replicated Database Services for World-wide Applications." In *Proc. 7th SIGOPS European Workshop*, pp. 275–280, Sept. 1996. ACM.

[9] M. Rabinovich and O. Spastscheck. *Web Caching and Replication*. Addison-Wesley, Reading, MA., 2002.

[10] Y. Saito and M. Shapiro. "Optimistic Replication." *ACM Computing Surveys*, 37(1):42–81, Mar. 2005.

[11] M. Satyanarayanan. "Distributed File Systems." In S. Mullender, (ed.), *Distributed Systems*, pp. 353–383. Addison-Wesley, Wokingham, 2nd edition, 1993.

[12] S. Shepler, B. Callaghan, D. Robinson, R. Thurlow, C. Beame, M. Eisler, and D. Noveck. "Network File System (NFS) Version 4 Protocol." RFC 3530, Apr. 2003.

[13] S. Sivasubramanian, G. Pierre, M. van Steen, and G. Alonso. "Analysis of Caching and Replication Strategies for Web Applications." *IEEE Internet Computing*, 11(1):60–66, Jan. 2007.

[14] S. Sivasubramanian, M. Szymaniak, G. Pierre, and M. van Steen. "Replication for Web Hosting Systems." *ACM Computing Surveys*, 36(3):1–44, Sept. 2004.

[15] H. Yu and A. Vahdat. "Design and Evaluation of a Conit-Based Continuous Consistency Model for Replicated Services." *ACM Transactions on Computer Systems*, 20(3):239–282, 2002.