Dynamic Sets for Search

David Steere, M. Satyanarayanan, and Jeannette Wing
School of Computer Science, Carnegie Mellon University, Pittsburgh, PA

The success of the proposed *information highway* hinges partly on solving the problem of locating useful objects out of the terabytes of data available. Therefore, a wide area data storage system (WADSS) must provide support for search. We view search as a directed iterative process; at each stage a query is run on a set of objects to reduce the focus to a more interesting subset. Search through a WADSS differs from search through a (distributed) database in that users are willing to trade consistency for performance. For instance, a user would be satisfied if a query were to miss some relevant objects, include irrelevant ones, or obtain out-of-date objects as long as the objects are returned promptly. Moreover, queries in a WADSS may be long-running, execute over geographically distant repositories, and may be prematurely aborted by the user. A key question is "What is the meaning of a set in this context?"

This question breaks into two related questions: "What is the proper definition of set membership?" and "How current do the members of the set need to be?"[1]. The first question arises because the state of the system in which a query is started (σ_0) may not be the same state in which it ends (σ_e). For example, if an object f returned by the query existed at time σ_0, but was deleted in σ_e, should it be considered part of the set S? The second question arises because the objects returned by the query may be modified after membership in the set has been determined. If an object f that is part of S has been modified since it was read by the query, should the user see the value f_{σ_0} which satisfied the query, or the current value f_{σ_e}, which may not?

The most obvious model in which to answer these questions runs the query atomically (force $\sigma_0 \equiv \sigma_e$). This model would be ideal since both set membership and the currency of the objects in the set reflect the latest state of the system. Unfortunately, this model is clearly impractical to implement. The potentially large number of users locking a large number of objects would effectively prevent any mutation from occurring in the system. A related model that runs the query atomically in some intermediate state σ_q, $\sigma_0 \leq \sigma_q \leq \sigma_e$ [1] suffers the same disadvantage in the presence of failures (see discussion of Weak Consistency in [1]). Unfortunately, failures will be a common occurrence in any system that spans a significant portion of the Internet[2].

A second model relaxes the semantics of set membership, but retains the guarantee that the members are current with respect to the state of the repository. An object f is defined to be a member of S if it satisfied the query at some intermediate state σ_i, $\sigma_0 \leq \sigma_i \leq \sigma_e$. The system will preserve a copy of f in the case it is deleted before σ_e. However, the system will present the latest available copy of f when the user peruses the set. One disadvantage is that the system must maintain state to preserve the currency of the objects in the set. Another disadvantage is that repeatedly examining a member may yield a different value each time.

Our new abstraction, *dynamic sets*, further relaxes the second model by only guaranteeing that an object is at least as current as it was in the state σ_i. These relaxed semantics are acceptable in the context of a WADSS because we expect users to prefer expediency over correctness for search. Also, we expect most information of interest to be either write-once or slowly changing, e.g. major releases of source code, compilations of technical articles, or messages on electronic bulletin boards. Once a sufficiently small subset of particularly interesting objects has been found, a user may wish to revalidate the objects in order to get a tighter bound on currency.

There are several advantages of dynamic sets over other existing models. First, they provide weaker semantics than database models. These weaker semantics more closely match the meaning of the kinds of queries users would run in the extremely large, unstructured, and decentralized environments that characterize the information highway. Second, dynamic sets do not require the system to maintain the currency of cached copies, allowing the system to scale without the additional state maintenance necessitated by stricter models[3]. Third, dynamic sets place no implicit ordering on the running of the query or the capture of the set members. The inherent "no ordering" property of sets has two important practical advantages. First, the system can fetch the members of a set in a cheapest-to-obtain manner, overlapping the latency of obtaining more expensive members with user-level processing of the earlier ones. And second, the system is free to prefetch the objects in the set aggressively (since membership is a strong hint of future access) or to evaluate the query lazily and to fetch the members on demand. Thus it can dynamically adapt its behavior to suit its current networking environment, local resource availability, and server load (hence our use of the term *dynamic sets*).

We are currently exploring these advantages by adding the dynamic set abstraction to the Coda File System running on Mach 2.6. Although dynamic sets could be used in a variety of systems, Coda provides a particularly rich environment for exploring issues of adaptability, as it supports a diversity of clients (from mobile hosts to powerful desktop computers) and modes of connectivity (from disconnected to high bandwidth communication).

This research has been supported by the Advanced Research Projects Agency (Hanscom Air Force Base under Contract F19628-93-C-0193, ARPA Order No A700; and the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, under grant number F33615-93-1-1330), IBM Corp., Digital Equipment Corp, Intel Corp., and Bellcore. The views and conclusion expressed in this paper are those of the authors, and should not be interpreted as those of the funding organizations or Carnegie Mellon University.

[1] $\sigma_a \leq \sigma_b$ means the state at time a occurred before or is the same state as the one at time b

PODC 94 - 8/94 Los Angeles CA USA
© 1994 ACM 0-89791-654-9/94/0008.$3.50

References

[1] H. Garcia-Molina and G. Wiederhold. Read-only transactions in a distributed database. *ACM Transactions on Database Systems*, 7(2), June 1982.

[2] D.D.E Long, J.L Carroll, and C.J. Park. A study of the reliability of internet sites. In *Proceedings of the 10th IEEE Symposium on Reliable Distributed Systems*, Pisa, Italy, Sept 1991.

[3] J. K. Ousterhout. The role of distributed state. In R. F. Rashid, editor, *CMU Computer Science, a 25th Anniversary Commemorative*, chapter 8. ACM Press, 1991.