Implementing and Evaluating an Eventually-Serializable Data Service

Oleg M. Cheiner* Alex A. Shvartsman†

Replication is used in distributed systems to improve availability and to increase throughput. The disadvantage of replication is the additional effort required to maintain consistency among replicas when serializing client operations. Fekete et al. [1] proposed the Eventually-Serializable Data Service (ESDS) that maintains replicated objects and allows the clients of the service to relax consistency requirements in return for improved responsiveness. ESDS guarantees eventual consistency of the replicated data, while allowing the clients to choose between *strict* operations with guaranteed consistent responses and *non-strict* operations that return responses consistent with some ordering of operations. ESDS builds on the work of Ladin et al. [3], who defined a data service with relaxed consistency and presented an algorithm based on *lazy replication*. The ESDS paper [1] includes a service specification and an abstract distributed algorithm; both are given in terms of I/O automata of Lynch and Tuttle [2]. The abstract algorithm is shown to implement the service specification, in the sense of trace inclusion.

The work summarized here aims to extend the formal foundations of ESDS [1] into the realm of practical system implementation. We describe implementation and evaluation of a distributed system building block based on the abstract ESDS algorithm. Our contributions are as follows. (1) We formulate a framework that assists a programmer in mapping algorithms formally specified using I/O automata to distributed implementations that use message passing. (2) Using the framework, we develop an optimized implementation of ESDS. This implementation is combined with different data types and clients, thus demonstrating the utility of the service as a building block suitable for serving as a distributed operating system component. (3) The implementation has been experimentally evaluated on a network of workstations. In this setting the implementation scales well with the number of processors and reflects a designed trade-off between consistency and performance.

We created a distributed implementation of the abstract ESDS algorithm from [1]. We also designed and implemented an optimized version of the algorithm. Both implementations use a framework for converting I/O automata to distributed systems that we devised for this purpose.

One of the main design goals was to show that the ESDS algorithm is suitable for implementation as a building block from which concrete applications can be build with minimal effort. The algorithm is specified to be independent of the serial data type of the replicated data object to facilitate its use as a building block. We built three distinct applications on top of ESDS to demonstrate that our implementation preserved this independence.

To make the implementation practical, we refined the ESDS algorithm to include several optimizations. The main optimizations are as follows. (1) *Incremental gossip:* Replicas send only new or changed information in gossip messages used for coordination, significantly reducing communication cost. (2) *Removal of self-gossip:* This is a subtle modification, since the correctness of the abstract algorithm depends on self-gossip. (3) *Memoizing stable state:* The ESDS paper [1] suggested an optimization that involves *memoizing* stable state at each replica. We implemented a variation of this optimization that stabilizes operations more aggressively while preserving correctness.

We evaluated the optimized implementation to see how its response time, throughput, and deviation of responses from strict consistency depend on the number of replicas and on the system load. Our data showed a near linear increase in throughput when the number of replicas is increased from 1 to 10. Due to increased replica coordination overhead, the throughput using 10 replicas was observed to be about three times the throughput using 1 replica. On a system with a constant overall load, the response time tends to increase with the first few additional replicas (due to coordination overhead), and then level off as the load on individual replicas lightens and they are able to keep up with both coordination activities and user requests.

We also performed tests to observe the intended trade-off between response time and consistency when the number of operations which required strictly consistent responses went up or down. Predictably, the percentage of inconsistent responses goes down linearly as the percentage of such strict operations climbs. On the other hand, the latency of responses to strict operations is significantly higher than to non-strict operations. This difference between strict and non-strict operation latencies explains the observed linear increases of average latency with the increasing percentage of strict operations. These results confirmed that ESDS represents a tradeoff between consistency and performance, and that it is possible to shift the tradeoff balance in either direction according to the user's needs.

*Carnegie Mellon University, Email: oleg@cmu.edu. The work of the first author was substantially done at the Massachusetts Institute of Technology.

†Laboratory for Computer Science, Massachusetts Institute of Technology, Email: alex@theory.lcs.mit.edu and Department of Computer Science and Engineering, University of Connecticut, Email: aas@cse.uconn.edu.

References

[1] A. Fekete, D. Gupta, V. Luchangco, N. Lynch, and A. Shvartsman. Eventually-Serializable Data Services. *PODC* 1996, pp. 300-310.

[2] N. Lynch and M. Tuttle. An introduction to Input/Output automata. *CWI-Quarterly*, 2(3):219–246, September 1989. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

[3] R. Ladin, B. Liskov, L. Shrira, and S. Ghemawat. Lazy replication: Exploiting the semantics of distributed services. *ACM Transactions on Computer Systems*, 10(4):360-391, Nov. 1992.