Improving Efficiency and Enhancing Concurrency of Untrusted Storage

Christian Cachin* Idit Keidar† Alexander Shraer†

Many users no longer keep all their data on local storage. Instead, it often resides on remote, online service providers. Examples include network filesystems, online collaboration servers such as Wikis, repositories using versioning tools like SVN, and web-based email providers. Because the data resides in another trust domain, users need methods for protecting the remotely stored data.

We are interested in the integrity of stored data and consider an untrusted server providing a storage service to multiple clients that connect to the server using an asynchronous network. The clients locally maintain a small amount of trusted memory but do not communicate with each other.

In this model, some malicious actions by the server can *never* be prevented. For example, the server may use an outdated value in the reply to a reader and omit a more recent update. The *SUNDR* system [2, 4] showed how such attacks can be exposed and made easily detectable. It ensures that whenever the server causes the views of two clients to differ in a single operation, the two clients may never again see each other's updates after that. Such a divergence can easily be detected through out-of-band communication.

SUNDR guarantees that the read and write operations observed by every client are atomic, preserving *linearizability*, and, furthermore, that for any operation visible to multiple clients, the sequence of operations occurring before that operation is the same. This notion has been called *fork-linearizability*. For a system with n clients, the SUNDR protocol incurs a communication overhead proportional to n^2 bits per operation.

Recently Cachin *et al.* [1] presented a more efficient fork-linearizable storage protocol with only $O(n)$ communication overhead per operation and showed that no fork-linearizable storage protocol is *wait-free*, even in executions where the server is correct. This means that even in the failure-free case, concurrent operations by different clients may slow each other down. Indeed, no existing fork-linearizable storage protocols guarantees progress for correct clients when other clients fail by crashing, even if the server is correct.

*IBM Research, Zurich Research Laboratory, CH-8803 Rüschlikon, Switzerland. cca@zurich.ibm.com

†Department of Electrical Engineering, Technion, Haifa 32000, Israel. {idish@ee, shralex@tx}.technion.ac.il

In ongoing work, we have examined two weaker consistency conditions that have been introduced recently: *fork-sequential-consistency* [5] and *fork-* consistency* [3]. Can they guarantee *wait-freedom* for operations by concurrent clients? We have discovered that neither of them allows linearizability and wait-freedom when the server is correct. Moreover, we show that fork-* consistency allows violations of causality in the sense that a client reads an outdated value from storage, although there was a more recent update that influenced the client.

We introduce a *new storage access protocol* that eliminates all these problems and achieves a notion that we call *weak fork-linearizability*. In particular, in every execution where the server is correct, the protocol is linearizable and wait-free, i.e., clients never wait for each other, even if some clients crash. If the server is faulty, the protocol still guarantees causality, data integrity, and linearizability for all but the last operation observed by every client. At the cost of exchanging only a small amount of information, clients can detect whether their states are consistent. In case a violation is detected, they can roll back to the latest consistent state. The protocol is efficient, requires only a single round of message exchange between a client and the server for every operation and has a communication overhead of only $O(n)$ bits per request.

References

[1] C. Cachin, abhi shelat, and A. Shraer, "Efficient fork-linearizable access to untrusted shared memory," in *Proc. 26th ACM Symposium on Principles of Distributed Computing (PODC)*, 2007.

[2] J. Li, M. Krohn, D. Mazires, and D. Shasha, "Secure untrusted data repository (SUNDR)," in *Proc. 6th Symp. Operating Systems Design and Implementation (OSDI)*, 2004.

[3] J. Li and D. Mazières, "Beyond one-third faulty replicas in Byzantine fault-tolerant systems," in *Proc. 4th Symp. Networked Systems Design and Implementation (NSDI)*, 2007.

[4] D. Mazières and D. Shasha, "Building secure file systems out of Byzantine storage," in *Proc. 21st ACM Symposium on Principles of Distributed Computing (PODC)*, 2002.

[5] A. Oprea and M. K. Reiter, "On consistency of encrypted files," in *Proc. 20th International Conference on Distributed Computing (DISC)* (S. Dolev, ed.), LNCS 4167, 2006.