Design and Validation of the P-Store Replicated Data Store in Maude*

Peter Csaba Ölveczky

[1] University of Oslo
[2] University of Illinois at Urbana-Champaign

Introduction. Many large applications—such as Google search, Gmail, Facebook, Dropbox, eBay, online banking and card payment processing—are expected to be *available* continuously, even under peak load, congestion in parts of the network, server failures, and during scheduled hardware or software upgrades. Such applications also typically manage large amounts of data. To achieve the desired availability, the data must be *replicated* across geographically distributed sites, and to achieve the desired scalability and elasticity, the data store may have to be *partitioned* across multiple partitions.

It is well known [2] that it is hard or impossible to both guarantee strong correctness properties (such as serializability), high availability, and strong fault tolerance. The lack of strong correctness guarantees is acceptable for applications such as Google search, Facebook, and online newspapers, but is unacceptable in, e.g., online banking, online commerce (eBay and airplane reservation systems), and medical information systems. P-Store [6] is a well-known replicated and partitioned data store that provides both serializability and some fault tolerance (e.g., transactions can be validated even when some nodes participating in the validation are down).

Wide-area replicated data stores designs are typically evaluated on real implementations or by using simulation tools, both of which are laborious tasks which cannot check "corner cases" to guarantee the absence of errors. In this talk I will talk about the use of the rewriting-logic-based Maude language and tool [3] to formally specify and analyze P-Store.

Why is this interesting? Although cloud computing data stores are complex artifacts, members of the University of Illinois Assured Cloud Computing center have exploited the simplicity and expressiveness of Maude to formally define and analyze complex cloud computing data stores such as Google's Megastore and Apache Cassandra [4,5]. So, why is formalizing P-Store interesting? First, it is a well-known data store in its own right with many good properties that combines both wide-area replication, data partition, some fault tolerance, serializability, and limited use of atomic multicast. Second, it uses atomic multicast to order concurrent transactions. Third, it uses "group communication" for atomic commitment. The point is that both atomic multicast and group communication commitment seem to be key generic building blocks in the design of cloud computing data stores (see, e.g., [1]) and that previous efforts have not formalized

* This work was partially supported by AFOSR/AFRL Grant FA8750-11-2-0084.

atomic multicast or group communication commitment. There are different protocols aimed at achieving atomic multicast. Instead of formalizing one of those, one of the main contributions of the P-Store formalization is an abstract Maude model of atomic multicast that allows any possible ordering of message reception consistent with atomic multicast.

Results. I have modeled both versions of P-Store, and performed model checking on small system configurations. Maude analysis showed some significant errors, like read-only transactions never getting validated in certain cases. An author of the original P-Store paper [6] confirmed that I had indeed found a nontrivial mistake in their algorithm and suggested a way of correcting the mistake. Maude analysis of the corrected algorithm did not show this error. I also found another error; the P-Store author said that this was not an error, but acknowledged that P-Store relied on a crucial assumption that was entirely missing from the paper. Finally, a key definition was very easy to misunderstand because of how it was phrased in English; fortunately, the P-Store author helped us clarify its meaning. However, it emphasizes the need for a formal specification in addition to the standard prose-and-pseudo-code description of the algorithm.

At the moment, I model check the system by defining interesting concurrent transactions and analyze the possible outcomes. In the near future, I should develop a systematic way of analyzing serializability, possibly by adding some kind of observer to the system state. A general technique for analyzing serializability of concurrent transactions would also be an interesting contribution. The executable Maude specifications of the different versions of P-Store, together with analysis commands, are available at `http://folk.uio.no/peterol/WADT16`.

References

1. Ardekani, M.S., Sutra, P., Shapiro, M.: G-DUR: a middleware for assembling, analyzing, and improving transactional protocols. In: Proc. Middleware'14. ACM (2014)
2. Brewer, E.A.: Towards robust distributed systems (abstract). In: Proc. ACM PODC'00. ACM (2000)
3. Clavel, M., et al.: All About Maude, LNCS, vol. 4350. Springer (2007)
4. Grov, J., Ölveczky, P.C.: Formal modeling and analysis of Google's Megastore in Real-Time Maude. In: Specification, Algebra, and Software, LNCS, vol. 8373. Springer (2014)
5. Liu, S., Rahman, M.R., Skeirik, S., Gupta, I., Meseguer, J.: Formal modeling and analysis of Cassandra in Maude. In: Proc. ICFEM'14. LNCS, vol. 8829. Springer (2014)
6. Schiper, N., Sutra, P., Pedone, F.: P-Store: Genuine partial replication in wide area networks. In: Proc. SRDS'10. IEEE Computer Society (2010)