Type-Aware Programming Models for Distributed Applications

Brandon Holt

A dissertation
submitted in partial fulfillment of the
requirements for the degree of

Doctor of Philosophy

University of Washington

2016

Reading Committee:

Luis Ceze, Chair

Mark Oskin, Chair

Dan Ports

Program Authorized to Offer Degree:
Computer Science and Engineering

University of Washington

Abstract

Type-Aware Programming Models for Distributed Applications

Brandon Holt

Co-Chairs of the Supervisory Committee:
Associate Professor Luis Ceze
UW CSE

Associate Professor Mark Oskin
UW CSE

The online world is a dangerous place for interactive applications. A single post by Justin Bieber can slow Instagram to a crawl; a black and blue dress going viral sends Buzzfeed scrambling to stay afloat; Star Wars movie ticket pre-sales cause service interruptions for Fandango and crash others. Developers of distributed applications must work hard to handle these high-contention situations because though they are not the average case, they affect a large fraction of users.

Techniques for mitigating contention abound, but many require programmers to make tradeoffs between performance and consistency. In order to meet performance requirements such as high availability or latency targets, applications typically use weaker consistency models, shifting the burden of reasoning about replication to programmers. Developers must balance mechanisms that reign in consistency against their effect on performance and make difficult decisions about where precision is most critical.

Abstract data types (ADTs) hide implementation details behind a clean abstract representation of state and behavior. This high-level representation is easy for programmers to build applications with and provides distributed systems with knowledge of application semantics, such as commutativity, which are necessary to apply contention-mitigating optimizations. With *inconsistent, probabilistic, and approximate (IPA) types*, we can express weaker semantics than with traditional ADTs, enabling techniques for weak consistency to be used and allowing precision to be explicitly traded for performance where applications can tolerate error. In previous work, we have demonstrated that ADT awareness can result in significant performance improvements: 3-50x speedup in transaction throughput for high-contention workloads such as an eBay-like auction service and a Twitter-like social network. IPA types are proposed for future work.

1. *Overview*

1.1. *Introduction*

To provide good user experiences, modern datacenter applications and web services must balance the competing requirements of application correctness and responsiveness. For example, a web store double-charging for purchases or keeping users waiting too long (each additional millisecond of latency [3, 5]) can translate to a loss in traffic and revenue. Worse, programmers must maintain this balance in an unpredictable environment where a black and blue dress [7] or Justin Bieber [6] can change application performance in the blink of an eye.

Recognizing the trade-off between consistency and performance, many existing storage systems support configurable consistency levels that allow programmers to set the consistency of individual operations [1, 2, 4, 8]. These allow programmers to weaken consistency guarantees only for data that is not critical to application correctness, retaining strong consistency for vital data. Some systems further allow adaptable consistency levels at runtime, where guarantees are only weakened when necessary to meet availability or performance requirements (e.g., during a spike in traffic or datacenter failure) [9, 10]. Unfortunately, using these systems correctly is challenging. Programmers can inadvertently update strongly consistent data in the storage system using values read from weakly consistent operations, propagating inconsistency and corrupting stored data. Over time, this *undisciplined* use of data from weakly consistent operations lowers the consistency of the storage system to its weakest level.

BIBLIOGRAPHY

[1] Apache Software Foundation. Cassandra. http://cassandra.apache.org/, 2015.

[2] Basho Technologies, Inc. Riak. http://docs.basho.com/riak/latest/, 2015.

[3] Brady Forrest. Bing and google agree: Slow pages lose users. Radar, June 2009. http://radar.oreilly.com/2009/06/bing-and-google-agree-slow-pag.html.

[4] Cheng Li, Daniel Porto, Allen Clement, Johannes Gehrke, Nuno Preguiça, and Rodrigo Rodrigues. Making geo-replicated systems fast as possible, consistent when necessary. In *Presented as part of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12)*, pages 265–278, Hollywood, CA, 2012. USENIX. ISBN 978-1-931971-96-6. URL https://www.usenix.org/conference/osdi12/technical-sessions/presentation/li.

[5] Greg Linden. Make data useful. Talk, November 2006. http://glinden.blogspot.com/2006/12/slides-from-my-talk-at-stanford.html.

[6] Cade Metz. How instagram solved its justin bieber problem, November 2015. URL http://www.wired.com/2015/11/how-instagram-solved-its-justin-bieber-problem/.

[7] Dao Nguyen. What it's like to work on buzzfeed's tech team during record traffic. http://www.buzzfeed.com/daozers/what-its-like-to-work-on-buzzfeeds-tech-team-during-record-t, February 2015.

[8] Yair Sovran, Russell Power, Marcos K. Aguilera, and Jinyang Li. Transactional storage for geo-replicated systems. In *ACM Symposium on Operating Systems Principles - SOSP'11*, SOSP. Association for Computing Machinery (ACM), 2011. doi:10.1145/2043556.2043592.

[9] Jeremy Stribling, Yair Sovran, Irene Zhang, Xavid Pretzer, Jinyang Li, M. Frans Kaashoek, and Robert Morris. Flexible, wide-area storage for distributed systems with WheelFS. In *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, NSDI'09, pages 43–58, Berkeley, CA, USA, 2009. USENIX Association. URL http://dl.acm.org/citation.cfm?id=1558977.1558981.

[10] Douglas B. Terry, Vijayan Prabhakaran, Ramakrishna Kotla, Mahesh Balakrishnan, Marcos K. Aguilera, and Hussam Abu-Libdeh. Consistency-based service level agreements for cloud storage. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles - SOSP 13*. ACM Press, 2013. doi:10.1145/2517349.2522731.