Client-centric benchmarking of eventual consistency for cloud storage systems

Wojciech Golab[1], Muntasir Raihan Rahman[2], Alvin AuYoung[3],
Kimberly Keeton[3], Jay J. Wylie[4], and Indranil Gupta[2]

[1]University of Waterloo, wgolab@uwaterloo.ca
[2]University of Illinois at Urbana-Champaign, {mrahman2,indy}@illinois.edu
[3]HP Labs, Palo Alto, firstname.lastname@hp.com
[4]LinkedIn Inc., jwylie@linkedin.com

Eventually consistent storage systems give up the ACID semantics of conventional databases in order to gain better scalability, higher availability, and lower latency. A side-effect of this design decision is that application developers must deal with stale or out of order data. As a result, substantial intellectual effort has been devoted to studying the behavior of eventually consistent systems, in particular finding quantitative answers to the questions "how eventual" and "how consistent"?[1]

Existing tools for evaluating eventual consistency have two primary shortcomings. First, they measure deviation from strong consistency based on the behavior of the storage system using either a white-box system model [2] or active measurement [4], as opposed to the behavior of the client. Second, these tools do not capture the lack of precision inherent in measuring consistency from various vantage points, such as a collection of storage servers and clients. To overcome the first shortcoming, in our earlier work [5], we proposed the Δ (Delta) metric [3], which captures the "deviation" of a given execution trace from Lamport's atomicity property (see [5] for more details). The Δ-atomicity property allows reads to return values up to Δ time units stale.

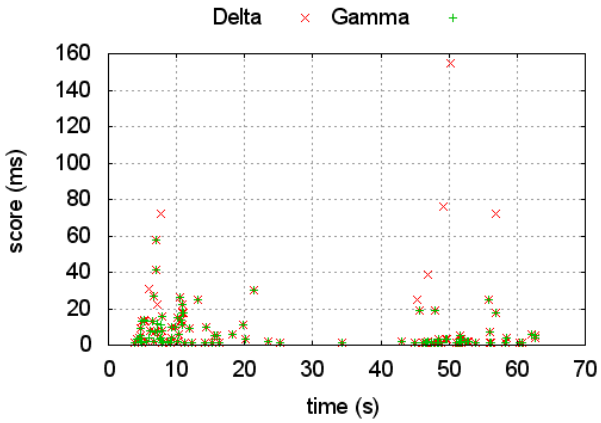

Figure 1: Δ and Γ scores vs. experiment time.

To overcome the second shortcoming, in this work, we propose the Γ (Gamma) metric which can quantify both stale reads and out of order write operations simultaneously. In contrast to the Δ computation, where we just shift the start point of each read to the left, in the Γ metric computation, we shift both the start point to the left and the finish point to the right for every operation. Due to this, the Γ metric can quantify how badly out of order write operations can be compared to the "happens before" order in a trace, thereby overcoming both shortcomings.

Figure 1 demonstrates a separation between Γ and Δ scores[2] for the same trace for an experiment using Cassandra [1]. We have also shown that the Γ metric is sensitive to various configuration and workload parameters.

[1]This work is supported in part by the NSERC Discovery Grants Program, in part by AFOSR/AFRL grant FA8750-11- 2-0084, and in part by NSF CCF grant 0964471. We are grateful to Ashraf Aboulnaga, Robbert van Renesse and Hakim Weatherspoon for computing support and insightful comments.

SoCC'13, 1–3 Oct. 2013, Santa Clara, California, USA.
ACM 978-1-4503-2428-1.
http://dx.doi.org/10.1145/2523616.2525935

[2]Please see [5, 3] for the technical definition of score functions.

References

[1] Cassandra. `http://cassandra.apache.org/`.

[2] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica. Probabilistically bounded staleness for practical partial quorums. *Proc. VLDB Endow.*, 5(8):776–787, Apr. 2012.

[3] W. Golab, X. Li, and M. A. Shah. Analyzing consistency properties for fun and profit. In *PODC'11: Proc. of the 30th Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing*, pages 197–206, 2011.

[4] S. Patil, M. Polte, K. Ren, W. Tantisiriroj, L. Xiao, J. López, G. Gibson, A. Fuchs, and B. Rinaldi. YCSB++: benchmarking and performance debugging advanced features in scalable table stores. In *SOCC'11: Proc. of the 2nd ACM Symposium on Cloud Computing*, pages 9:1–9:14, 2011.

[5] M. R. Rahman, W. Golab, A. AuYoung, K. Keeton, and J. J. Wylie. Toward a principled framework for benchmarking consistency. In *Eighth USENIX Workshop on Hot Topics in System Dependability*, HotDep'12, 2012.