Probabilistically Consistent

Indranil Gupta
Department of Computer Science
University of Illinois at Urbana-Champaign
indy@illinois.edu
FuDiCo 2015

Abstract

In an extensible distributed system, timeliness and consistency (or correctness) requirements from clients and applications are naturally at odds with the unpredictability endemic in distributed systems and networks, such as due to network delays and failures. Over the past decade and half, the community (both research and open-source) has treated this as largely a hard choice between extremes. Examples include the CAP theorem (seen as a choice between consistency and availability in storage systems that are partitionable) and checkpointing (for 100% accuracy in computations under failures).

We argue that this choice between extremes is in many cases (and should be treated as) a probabilistic tradeoff. We first present a probabilistic version of the CAP theorem (with the original CAP theorem becoming a special case), and then use this to incorporate probabilistic SLAs/SLOs into popular key-value storage systems. We then show why checkpointing should be jettisoned altogether for some types of computations, like distributed graph processing, in favor of an opportunistic, zero-cost failure recovery approach that provides high probabilistic accuracy when under failures.

Extended Abstract

The emerging era of extensible distributed systems will rely heavily on both storage and computation services running in remote and large-scale datacenters, which cater to many clients (e.g., at mobiles, tablets, computers, etc.). In order to be useful, usable, and conform to human expectations, such services will need to be timely, consistent, and correct.

Timeliness and consistency (or correctness) requirements are naturally at odds with the unpredictability endemic in distributed systems and networks, such as due to network delays and failures. Over the past decade and half, the community (both research and open-source) has treated this as largely a hard choice between extremes. A primary instance of this is in distributed storage where the CAP theorem has been cast as a binary choice between consistency and availability--this has lead to the emergence of highly-available (and weakly consistent) NoSQL databases. A second instance is in computation services, where checkpointing is implemented (and recommended) in many systems in order to produce “always correct” results in spite of failures--however, its high overhead means many users disable checkpointing!

We argue that this choice between extremes is in fact a probabilistic tradeoff. Human expectations from such services are naturally probabilistic in nature. This philosophy also enables one to build into these systems adaptivity, and satisfaction of service level agreements/objectives (SLAs/SLOs). To illustrate this, we first generalize the CAP theorem into a probabilistic CAP theorem which delineates the unachievable envelope (instead of hard choice) among C, A, and P using probabilistic parameters [1]. We use this result to design and implement adaptive variants of NoSQL databases that can adaptively satisfy consistency SLAs and latency SLAs, even as network conditions change dynamically. We have implemented these techniques in Apache Cassandra and Riak, the two most popular NoSQL databases. Second, we show that in distributed computation services like graph processing, checkpointing should be jettisoned altogether in favor of a zero-cost (in the common case of no failure) approach which reacts to a failure by “scrounging” available data from surviving servers. We show that for a wide variety of graphs and benchmarks this zero-cost approach results in very high result accuracy even after multiple failures [2].

[1] M. Rahman, L. Tseng, S. Nguyen, I. Gupta, N. Vaidya, “Characterizing and Adapting the Consistency-Latency Tradeoff in Distributed Key-value Stores,” arXiv:1509.02464, 2015. URL: http://arxiv.org/abs/1509.02464

[2] M. Pundir, L. Leslie, I. Gupta, R. Campbell, “Zorro: Zero-Cost Reactive Failure Recovery in Distributed Graph Processing,” Proc. ACM Symposium on Cloud Computing (ACM SoCC), 2015.