CAP Twelve Years Later: How the "Rules" Have Changed

Eric Brewer

Overview

The goal of this article from the magazine Computer is to redefine the CAP theorem and show that with careful deployment of invariants and management of partitions, it is possible to have consistency, availability and tolerance to partitions without having to trade one for the other two. The paper first explains why the '2 of 3' choice, especially developers having to prioritize partitions and therefor being forced to choose either consistency or availability is a misconception. The author proposes a strategy that allows developers to have all three components of CAP: detect partitions → enter a partition mode that may limit operations → eventually move to a recovery mode that restores the partition and compensates for mistakes.

The author then defines partitions and how they are formed in the event of a timeout. After this, managing partitions in a 'partition mode' are discussed, regarding what operations to carry out and how to track and then later synchronize operations that take place in partition mode. This ties into the next section, which deals with a few approaches on how to restore consistency once communication is restored between nodes across the partition. The next section discusses compensation for mistakes that may have occurred during the partition mode. The paper concludes by stating that the new techniques discussed in the paper along with nuanced deployments mean that developers would not have to choose between either consistency or availability in the face of partitions.

Contributions and Positive Aspects

The paper's aim is to prove that the conventional wisdom of the '2 of 3' is misconceived. It does this by stating that in most cases, systems, partitions are rare, and that choosing between consistency and availability is heavily dependent on the system in question. The author also takes into account the level of nuance in the concepts of consistency and availability. Using these concepts, the author is then able to propose the strategy of using partitions to keep nodes available without forfeiting consistency.

The paper also introduces some new techniques to help with distributed systems. While the author himself has not come up with these, the article contains a good roundup of techniques to optimize partition techniques such as version vectors, Concurrent Versioning Systems (CVS) and Commutative Replicated Data Types (CRDTs). Of these, I appreciate version vectors the most, since it seems to be an application of Lamport's vector clocks to order events, which I am more familiar with than the concepts in CVS and CRDTs.

The paper explains the concept of partitions very well and the sections on managing them, the trade-offs and consequences associated with a particular action and recovering partitions are all clearly and concisely explained. The use of real-world distributed systems as well as general examples to illustrate points were all very helpful.

Drawbacks

In my opinion, the biggest limitation of this paper is that it stretches the definition of consistency, by allowing some invariants to be violated so that they can later be rectified. Since the definition of consistency as per this paper is that a single up-to-date copy of the data should exist, while the system may eventually become consistent, there is a period where consistency is violated. The section on compensating for mistakes addresses this issue, but its very existence means that there has been a violation of consistency.

More largely, this ties into the problem that the paper contradicts itself. The paper's thesis is that by using partitions and invariants strategically, it is possible to have consistency, availability and tolerance to partitions without sacrifice any of the three. However, the explanation for the implementation of the strategy, it becomes repeatedly clear that consistency is being forfeited, in favour of availability and partition tolerance. For example, when describing the actions, a developer might take in the event of a timeout, for partition management to take place, the developer would have to pick the option of proceeding with the operation, the option that sacrifices consistency. The version vector technique, again, by the paper's admission is a developer's best bet for some kind of consistency when a developer has chosen to focus on availability.

The key components of maintaining consistency after focusing heavily on availability in the partition mode are partition recovery and compensating mistakes. However, the fact that there could be mistakes that need to be compensated means that consistency has failed. Moreover, the paper's examples of external mistake recovery seem to indicate that either the customer or the developer will be short-changed in these scenarios; either the developer/provider must provide the customer some monetary compensation, in which case their mistake has cost them, or if the customer does not take any action after a mistake, they have received sub-optimal service from a provider they are possibly paying. In either case, it seems to make more sense for the developer to prioritize consistency a bit more, but the article makes no such recommendation.

Another drawback, in my opinion is that the paper relies heavily on a great number of assumptions in order for the proposed strategy to work. The first assumption is that communication eventually resumes between nodes on either side of the partition. There is no discussion of a scenario in which the partition has permanently failed. Secondly, the partition recovery section seems to assume that some invariants may be violated in the partition mode. Critical systems, where invariants cannot fail are not discussed in great depth and as such there is no explanation for how recovery across partitions would work if invariants have not been violated. Possibly, there would be no need for recovery in this case because the partition would not have diverged, but it is not explicitly stated. Lastly, while it is not exactly an assumption, the paper states that consistency and availability can be mapped to continuous values not a binary. While it is easy to see how this may be the case with availability (e.g. certain features being available but not others), the paper does not explain how consistency might have levels, nor do later sections satisfactorily explain this nuanced definition of consistency satisfactorily, in my opinion.

Comments and Suggestions

One question I have is whether the strategy in this paper of partition management has actually been adopted by distributed systems providers, either because of this paper or whether they arrived at it independently. Looking through the number of citations on Scholar suggests that this hasn't been the most influential work, but that probably isn't the best metric to gauge whether any aspects of this strategy have been adopted.

In my opinion, this paper could have benefitted from stating its scope, and clearly defining the types of distributed systems that would benefit from adopting the proposed strategy. As it is, given how varied and context-dependent distributed systems are, I don't believe that there's any one-size-fits-all approach to solving problems, and this strategy is included. Listing systems where this kind of partitioning and sacrificing consistency works and doesn't work would help illustrate the applicability of the strategy more clearly.

I also personally feel that the author should have been more explicit about some things, such as how even in this case consistency is being sidelined for availability, which does uphold the conventional wisdom of '2 of 3'. I think that although it is strongly implied, the paper could have been franker about how mistake compensation in the incarnation in the paper does not seem like an ideal or fair solution at all.

This might be a minor nitpick, but I fail to understand why the author mentions that partitions are very rare in distributed systems, and so generally since there are no partitions, it is possible to have both consistency and availability. Aside from their being no source for this statement (I'm assuming because maybe timeouts being rare is understood to be a given?), it seems odd to me to have that statement in a paper that goes on to deal with managing partitions in as much depth as it does.

Lastly, I appreciated the sidebars with extra but not crucial information about the concepts presented in the paper. I found them interesting, and that they helped enrich my understanding of the paper.

Conclusion

Although this paper introduces intriguing concepts such as nuance in availability and consistency as well as techniques like version vectors and CRDTs, in my opinion, by using a strategy that heavily sidelines consistency for availability, and with an unconvincing plan for rectifying mistakes, it fails to prove its central thesis that with careful management of partitions, it is possible to have availability without consistency.