International Journal of Scientific & Engineering Research Volume 8, Issue 5, May-2017
ISSN 2229-5518

A Comparative Study of ACID and BASE in Database Transaction Processing

Mr. Keith Machado , Mr. Rohan Kank ,Ms. Jeenal Sonawane ,Mr. Shirshendu Maitra

Abstract - Two disparate database reliability models are the primary issue discussed within this paper: ACID (Atomicity, Consistency, Isolation, and Durability) and BASE (Basically Available, Soft state, Eventual consistency). In the old, yet still extant, world of vertical scaling, the concept of ACID, which is proven standard for SQL-centric and other relational databases, has been around for 30+ years and works remarkably well. In the new world of Data Management, with the rise of social networking, NoSQL, Big Data and other leviathans, popularity of BASE has increased only recently, over the past 10 years or so. The need for real-time availability constraints of web-based transaction processing, ever expanding horizontally scaled distributed networks, alongside non-relational data stores gave rise to the requirement of BASE. With Brewer's CAP Theorem acting as the referee in the middle forcing tough decisions on each team, they essentially represent two competing groups, although there are now more crossovers and negotiations between the two models.

Index Terms – BASE,ACID,transaction

1 INTRODUCTION

ACID and NoSQL are not the antagonists they were once thought to be; NoSQL works well under a BASE model, but also some of the innovative NoSQL systems fully conform to ACID requirements. Database engineers have puzzled out how to get non-relational systems to work within an environment that demands high availability, scalability, with differing levels of recovery and partition tolerance. BASE is still a leading innovation that is wedded to the NoSQL model, and the evolution of both together is harmonious. But that doesn't mean they always have to be in partnership; there are several options. So while the opening anecdote is true in many cases, organizations that need more diverse possibilities can move into the commercial arena and get the specific option that works best for them.

Relational database systems are almost always ACID complaint because relational indexing is centralized and so there is no advantage in BASE. An ACID database functions

like a unit that is fully consistent with transactional updates; while a BASE system functions like independent units that are eventually consistent, and without transactional updates. There were no other choices than relational systems in the past; but the advent of NoSQL now gives organizations that need a different model more choices. They can do ACID or BASE, so the variety is much greater. It is now possible to provide ACID constraints and other enterprise features within specific NoSQL systems if that is what the organization requires to meet its needs.

2 ACID

The primary work with database reliability constraints began in the 1970s with Jim Grey. He formulated the first three elements of the acronym – Atomicity, Consistency, and Durability – in his seminal work "The Transaction Concept: Virtues and Limitations" that was published in 1981 [1]. The paper looked at transactions in terms of contract law, whereby each transaction had to conform to

specific "transformations of a system state." All transactions had to obey the laws defined within the contract parameters. According to Grey, each transaction within a database had to obey protocols, either happened or didn't happen, and could not be changed once committed. In 1979, Bruce Lindsay et al. expanded on Grey's preliminary findings with their paper "Notes on Distributed Databases."[2] The paper focused on the essentials for achieving consistency within distributed database management systems, data replication, authorization and access controls, recovery management, two-stage commits, and others. The final foundational element of ACID came in 1983 with the publication of Andreas Reuter and Theo Härder's paper "Principles of Transaction-Oriented Database Recovery."[3] They added the principle of Isolation to the discussion and officially coined the acronym ACID; it has persisted for the past 30 years as the indispensable constraint for achieving reliability within database transactions, and in simple terms means:

Atomicity: Either the task (or all tasks) within the

transaction are performed or none of them are. This is the all or none principle. If one element of a transaction fails the entire transaction fails.

Consistency: The transaction must meet all

protocols or rules defined by the system at all times. The transaction does not violate those protocols and the database must remain in a consistent state at the beginning and end of a transaction; there are never any half completed transactions.

Isolation: No transaction has access to any other transaction that is in an intermediate or unfinished state.

Thus, each transaction is independent unto itself. This is required for both performance and consistency of transactions within a database.

Durability: Once the transaction is complete, it will persist as complete and cannot be undone[4]; it will survive system failure, power loss and other types of

system breakdowns.

There are of course many facets to those definitions and within the actual ACID requirement of each particular database, but overall in the RDBMS world, ACID is overlord and without ACID reliability is uncertain. The pH of ACID is low, roughly similar to battery acid (0) or maybe vinegar (2), the data and its constraints are exceedingly active. Therefore, at any given microsecond in a database that uses ACID as its system constraint all the data (hydrogen atoms) are undergoing constant checks to make sure they fulfill those constraints. Such requirements had worked quite well for many years in the smaller, horizontally schema-driven, scalable, normalized, relational world of the Pre-Social Networking bygone age. Such past truisms are longer the case; Unstructured Data, non-relational data structures, Big Data, distributed computing systems and eventual consistency are now becoming more common place; new requirements mean new acronyms and a new pH.

ACID constraints have provided transaction processing with a reliable foundation from which to build for decades, and would have continued were it not for the advent of the Internet, the growth of distributed data stores, the unprecedented increase in data volumes and variability, the need to document and store unstructured data, and the subsequent need for more flexibility in terms of scaling, design, processing, cost, and disaster recovery. This is not a claim that ACID requirements are no longer essential to transaction processing, because they are. Web-scale applications, non-relational data stores, and global distribution of data centers required the creation of new alternatives. Nevertheless, with new evolutions in the database field, it is now possible to have a fully functional NoSQL database that conforms to strict ACID compliance – such systems are still within the industry at this time, but they do now exist for organizations that need such an evolution.

3 BASE

BASE is a clever acronym, especially when paralleled with ACID – data professionals are the chemists of the IT universe. While it is not known for sure who originated the term, most people give credit to Dr. Eric Brewer for at minimum popularizing the term. In 2000, his keynote address at the ACM Symposium titled "Towards Robust Distributed Systems"7 proved to be the shining moment when many in the industry nodded their heads and knew, at least in their guts, that momentous changes were on the horizon. BASE is essentially the diametric opposite to ACID, with the limitations outlined by Brewer falling across the spectrum. The BASE acronym entails:

- Basically Available – the system guarantees some level of availability to the data even in regards to node failures. The data may be stale, but will still give and accept responses.
- Soft State – the data is in a constant state of flux; so, while a response maybe given, the freshness or consistency of the data is not guaranteed to be the most current.
- Eventual Consistency – the data will eventually be consistent through all nodes and in all databases, but not every transaction at every moment. It will reach some guaranteed state eventually.

In Brewer's address he presented a simple table that outlined the essential traits of each of the two models:[5]

ACID	BASE
• Strong consistency • Isolation	• Difficult evolution (e.g. schema)
• Focus on "commit"	• Availability first
• Nested transactions	• Best effort
• Availability?	• Approximate answers OK
• Conservative (pessimistic)	• Aggressive (optimistic)
• Difficult evolution (e.g. schema)	• Simpler
	• Faster
	• Easier evolution

He stressed that the entire balance between the two is a spectrum; database engineers had to choose the specifics of what they needed and wanted versus what they could attain when developing their particular application. The real focus of the opposition between the two competing models is demonstrated with Brewer's CAP Theorem, which outlines the three major characteristics of database transaction processing, and contends that only two of the characteristics can be met at any given time. The three central elements of CAP (Consistency, Availability, and Partition Tolerance) have since been expanded to include much more detail, along with extensive experimentation by engineers around the world to verify and quantify the real-world results of such a theorem.

4 CONSISTENCY - HOW IS THE DATA PERCEIVED?

ACID constraints provide strong consistency, all the time, no matter what. Such requirements often have repercussions, though; especially regarding availability. If the system must always remain in a consistent state, so all parties see the same view of the data at the beginning and end of a transaction, then across thousands of nodes that data may not always be available. The same repercussion affects disaster recovery and the loss of nodes – if one part of a distributed database collapses, then strong consistency would not allow any further updates until the entire system is realigned. Thus, we come to Eventual Consistency and a range of other consistency guarantees:

A. Strong (Strict) Consistency – All read operations return the value from the last finalized write operation. It doesn't matter which replica the operation completed the write to; all replicas must be in the same state for the next operation to occur on those values.

B. Eventual Consistency – This has the greatest variability of potential values returned. At any given point readers will see some written value, but there is no guarantee that any two readers will see the exact same write. All replicas will eventually have the latest update; it's just a matter of time when that will happen.

C. Monotonic Read Consistency – This is also known as a session guarantee. Reads are similar to eventual consistency in that the data could still be stale; but monotonic read consistency guarantees that over time the client will get a more up-to-date read (or the same read) if they request a read from the same object.

D. Read Your Own Writes (or Read My Writes) – guarantees that the client always reads their most recent writes, but other may not see the same updates. It doesn't matter what replica the writes are going to, the client always sees their most updated one.

E. Causal Consistency – if a client reads one value (a) and then writes the next value (b), and another client then reads the value of (b) they will also see the value of (a) since they are connected to each other. Therefore, any writes that are causally related must be seen by all processes in the specific order they were written. [6]

F. There are many other consistency guarantees with different names, including (but not limited to) casual+, sequential, consistent prefix, entry consistency, release consistency, FIFO consistency, and bounded staleness. But the main issue is the fact that application programmers must weigh their options when deciding on the consistency requirements of any given transaction. The other two characteristics of CAP Theorem impose restrictions on what level of consistency can be guaranteed.

5 CONCLUSION

The crucial question of ACID versus BASE, the implementation of relational versus non-relational data stores, consistency versus availability, all hinges on one primary element: success in the marketplace. It doesn't matter if an organization's developers are fawning over the possible integration of a new NoSQL-based system if it is an unnecessary or untenable addition to a business that still only requires a relational platform. Certainly, development planning for the future must include an IT strategy that focuses on growth, flexibility, security, and cost. The exponential growth of data, and its subsequent collection and analysis, are mitigating factors that now require many organizations to adopt a Big Data solution, with some kind of NoSQL platform as its foundation. Yet, the exact design of that solution, and its countless considerations are entirely dependent on the needs of the enterprise: server virtualization, the impact of Cloud-based services, data center utilization and consolidation, more efficient application performance, streamlined data governance processes, hardware costs, and a host of other issues ultimately govern the decision to implement a NoSQL solution and what that solution will mean to the organization. Much of the financial industry still requires strict ACID compliance for absolute transaction integrity in all banking operations, as do many military and other governmental organizations; the same could be said of the health care industry, especially regarding patient records. An online travel service may be able to relax consistency during the search process through various soft state protocols and updating mechanisms so that higher availability can be ensured. A social networking site that starts small, but plans on exponential growth, would want to implement a system with availability and scalability in mind from the beginning, and may decide to use some sort of eventual or tunable consistency with various user "click traffic" that doesn't require strict consistency. The multitude of options available in the marketplace makes deciding on a specific NoSQL solution challenging. Do you need a graph database or document store? Key-value store or columnar database? What sort of executive support is there for the project? Is it an enterprise-wide implementation, or only for a small program to begin with? Will outside consultants be necessary? What about application programmers? Should the solution be developed in-house, or should an off-the-shelf solution be purchased? These questions and innumerable others will decide the fate of a particular solution. But in the end, the need for such a solution distills down to one essential factor: the need to remain innovative and flexible in an ever-changing business environment. ACID and BASE are just clever acronyms for complex problems. Luckily for modern organizations, there are myriad solutions for those problems. device like Raspberry Pi or BeagleBoard, but it would be overkill for the task.

6 REFERENCES

[1]Grey, J. (June 1981). The Transaction Concept:
Virtues and Limitations. Tandem Computers Incorporated. Retrieved from http://research.microsoft.com/en-us/um/people/gray/papers/theTransactionConcept.pdf

[2] Lindsay, B.G., Selinger P.G., Galtieri, C., Gray, J. N., Lorie, R. A., Price, T. G., Putzolu, F., Traiger, I. L.,

Wade, B. W. (July 1979) Notes on Distributed Databases. Cambridge University Press. Retrieved from http://domino.research.ibm.com/library/cyberdig.nsf/papers/A776EC17FC2FCE73852579F100578964/$File/RJ2571.pdf
Reuter, A., Härder, T. (December 1983) Principles of Transaction-Oriented Database Recovery.

Computing Surveys. Retrieved from http://cc.usst.edu.cn/Download/5b953407-339b-46c3-9909-66dfa9c3d52a.pdf.
Pritchett, D. (2008). BASE: An Acid Alternative. Retrieved from

http://queue.acm.org/detail.cfm?id=1394128.
[5] Terry, D. (October 2011) Replicated Data Consistency Explained Through Baseball. Microsoft Research. Retrieved from http://research.microsoft.com/apps/pubs/default.aspx?id=1574 11. And, ©2013 DATAVERSITY Education, LLC. All rights reserved. 16
Strauch, C. (2012). NoSQL Databases. Stuttgart Media University. Retrieved from http://www.christof-strauch.de/nosqldbs.pdf. And, Burckhardt, S., Leijen, D., Fähndrich, M., Sagiv, M.(2012). Eventually Consistent Transactions. Springer-Verlag. Retrieved from http://research.microsoft.com/pubs/158085/ecr-esop2012.pdf. And, Wada, H., Fekete, A., Zhao, L., Lee, K., Liu, A., (January 2011). Data Consistency Properties and the Trade-offs in Commercial Cloud Storages: the Consumers' Perspective.Retrieved from http://www.nicta.com.au/pub?doc=4341