At Scale, is it Worth Compromising on Stability for the Sake of Throughput?

Nicolás D'Cotta – 24th February, 2022 *(last modified 25th February, 2022)*

***Abstract*—We consider different business domains where compromises must be made between stability and throughput and reflect on the motivations behind them; mostly drawing from published articles and personal experience.**

I. When Throughput is Critical

Stability in itself is a vague term hard to quantify when talking about a system. We will define it in terms of availability (the proportion of time a system is not down) and consistency (to what extent the data provided by the system reflects reality or is up to date).

A great example of a business domain where consistency was sacrificed for the sake of throughput is Amazon's Dynamo database [1]. Consider the system responsible for displaying an online marketplace where customers perform purchases: it will suffer heavy read-only load, with critical write updates involving purchases and shopping cart modifications. These updates drive Amazon's business, and so the team behind Dynamo prioritised being able to cope with write updates over providing a consistent view of the data. Additionally, as a distributed database, Dynamo was designed to run on clusters of unreliable commodity hardware. The inherent instability of the underlying hardware leads to a system that must tolerate failure – again, Dyanmo does so at the expense of consistency.

Dynamo is therefore a good example of a sacrifice in stability (in this case in the area of data consistency) for the sake of being able to achieve high write throughput. For more on the motivations behind this compromise and how it was achieved, refer to [1].

II. When Stability Cannot be Sacrificed

Spanner is a different distributed database system developed by Google [2]. The use-case for Spanner was Google's advertising backend, F1 – one of the main revenue drivers of the company. Google wanted their data to be able to survive natural disasters affecting entire datacenters as well as remaining consistent. For this, they implemented synchronous cross-datacenter data replication, which makes throughput matching Dynamo's impossible as computers communicate across continents. Spanner also supports transactionality (much like a normal SQL database) which gives it the ability to provide strong consistency guarantees.

Paxos, one of the systems powering Spanner, is a lower-level abstraction that makes a similar compromise [3]. It is a protocol for state-machine replication that tolerates failure of some underlying participants. The more participants, the more failures it can tolerate (and is therefore more stable!), but its throughput decreases because replication must be done across more nodes. Why Paxos is a good fit for Spanner is clear in Google's requirements for F1 of availability in the face of failure.

While running on hardware similar to Dynamo's, Spanner prioritises guarantees on stability (via consistency and availability) rather than throughput, for the sake of being able to provide a consistent view of data available under all circumstances.

III. When Stability Requires Throughput

In some systems and business models, a trade-off between a stable and available design and a high-throughput one is not always possible.

Take the example of *Blockchain.com*, a crypto finance house that processes payments. It has a ledger service that receives updates from other services and updates users' balances in its persistent state [4]. When there are more messages per second than what the ledger can process at a time (which can happen during big market swings), this creates inconsistent views of balances displayed to users and some other services. For example, a user might have made a payment but cannot see the new balance in their account. This lack of capacity throughput-wise then causes a stability problem as other services struggle to deal with such monetary inconsistencies, which in turn translates into losses for the company.

The fintech industry in general is a great example where the domain does not allow to compromise on stability nor throughput: systems must be able to scale requests with market swings (which always increase load) while remaining consistent and available so as to avoid missing out on profitable market activity. Systems must be designed with these priorities in mind – which can involve sacrifices in other areas, such as cost (by over-provisioning) or maintainability.

IV. Conclusion

As with most compromises to be made when designing a system, the right decision most often comes down to the needs of the business domain – if revenue is driven by throughput and not stability, then the right compromise between the two is likely to favour throughput (and vice-versa!).

While there are cases where both stability and throughput are critical, both can still be achieved by compromising on other areas, most commonly cost.

References

[1] G. DeCandia, D. Hastorun, M. Jampani, *et al.*, "Dynamo: Amazon's highly available key-value store," in *Proceedings of Twenty-First ACM SIGOPS Symposium on Operating Systems Principles*, ser. SOSP '07, Stevenson, Washington, USA: Association for Computing Machinery, 2007, 205–220, ISBN: 9781595935915. DOI: 10.1145/1294261.1294281. [Online]. Available: https://doi.org/10.1145/1294261.1294281 (visited on 02/25/2022).

[2] J. C. Corbett, J. Dean, M. Epstein, *et al.*, "Spanner: Google's globally distributed database," *ACM Transactions on Computer Systems (TOCS)*, vol. 31, no. 3, pp. 1–22, 2013. [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//archive/spanner-osdi2012.pdf (visited on 02/25/2022).

[3] R. Van Renesse and D. Altinbuken, "Paxos made moderately complex," *ACM Computing Surveys (CSUR)*, vol. 47, no. 3, pp. 1–36, 2015. DOI: 10.1145/2673577.

[4] Author Anonymous for Review, Personal Experience, Blockchain.com, 2022.