Date: 13th March 2019

CMPT 756: Systems for Big Data (Spring 2019) – Assignment 1

Consistency, Latency or Quorum Size?

A Quantitative Analysis of Tradeoffs in a Distributed Database System Design

Anurag Bejju – abejju@sfu.ca

---

1. Introduction

In today's technology-driven world, we can witness a strong desire among companies to find efficient solutions for scaling systems without the need for additional resources (computing power, storage systems etc.). In an ideal scenario, most big data system designers want to develop a scaling strategy with the least amount of design trade-offs possible. Since distributed databases provide an optimum solution that addresses scalability concerns, a majority of current tech companies use them as part of their system design. But with it, there comes unique challenges and tradeoffs that need to be resolved based on the application needs and requirements. Some of those tradeoffs include availability, response time, consensus and replication strategies, event ordering & time, failure tolerance, dependencies etc.

2. Objective

In this particular assignment, we would be accessing *consistency*, *latency*, and *quorum size* design trade-offs for four implementations (i.e *A, B, C,* and *D*) of a replicated database with each implementation being replicated *three* times.

Latency: This is one design aspect that greatly affects how usable and enjoyable the user experience is with a particular product or application. [1] In fact, research shows that latency is a critical factor for online interactions and an increase as small as 100ms can drastically reduce the probability of customer retention [2]. Therefore, by observing the latency distribution of responses from the various machines in the system, we can sometimes better access the load distribution and latency measures of the entire design. Here we observe two types of latency distributions:

- ***Uniform:*** Latency is distributed uniformly over a relatively narrow range. There is a maximum latency and it is relatively near the minimum.
- ***Ragged:*** Latency is distributed exponentially. There is no maximum possible value and extremely large values are possible.

Consistency: In a distributed system, each node holds a local copy of the application [3]. It is nearly an impossible task to ensure complete consistency is maintained among all of them. Therefore, most system designers go for a level of reduced consistency in order to improve the overall latency of a system. In this assignment, we measure two kinds of consistencies (Internal consistency and External consistency) for our analysis [4] .

- ***Internal consistency*** measures the consistency between responses to a single request.
- ***External consistency*** measures the consistency between successive calls to the database where each database value is timestamped.

Quorum Size: In a quorum-based implementation, a distributed transaction has to obtain majority votes in order to perform an operation. This can be used as a replica control method as well as a commit method to ensure transaction atomicity in the presence of network partitioning [5] . In this analysis, we collect statistics for all possible quorum sizes for a three-way replicated database and send a request to all the three instances of the database. The only difference is in quorum algorithms where a combination of responses is used. The three quorum algorithms are:

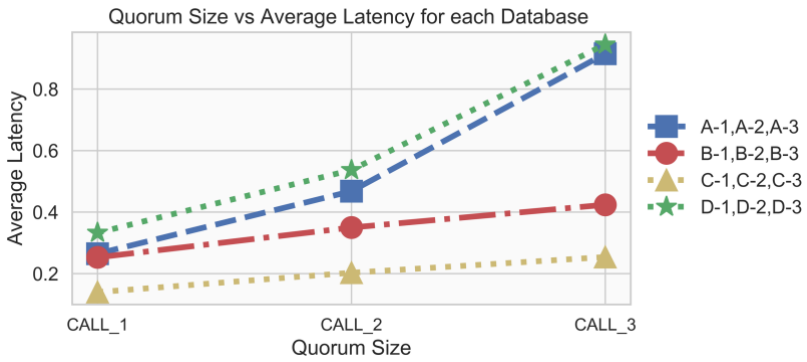
- ***CALL_1:*** Only the first response is considered and the remaining two are discarded
- ***CALL_2:*** First two responses are considered and the last one is discarded.
- ***CALL_3:*** All three responses are considered.

3. Quantitative Analysis of Tradeoffs in 4 Distributed Database Implementations

The following consistency-latency statistics have been collected for all possible quorum sizes on four implementations (i.e A, B, C, and D) of a replicated database.

Consistency-Latency- Quorum Size Statistics					
Database	**Quorum Size**	**Average Latency**	**Average External Consistency**	**Average Internal Consistency**	**Period**
A-1,A-2,A-3	CALL_1	0.333	68	100	10
	CALL_2	0.481	75	84	10
	CALL_3	0.863	92	84	10
B-1, B-2,B-3	CALL_1	0.239	100	100	10
	CALL_2	0.34	100	100	10
	CALL_3	0.415	100	100	10
C-1,C-2,C-3	CALL_1	0.134	85	100	10
	CALL_2	0.182	82	84	10
	CALL_3	0.235	92	84	10
D-1,D-2,D-3	CALL_1	0.354	100	100	10
	CALL_2	0.526	100	100	10
	CALL_3	0.904	100	100	10

3.1. Latency VS Quorum Size:



As you can see in the line graph plotted between *Quorum Size* and *Average Latency of each database implementation*, Databases B performs the same way as C and Database A performs the same way as D. Let's access each database's Latency vs Quorum Size relationship.

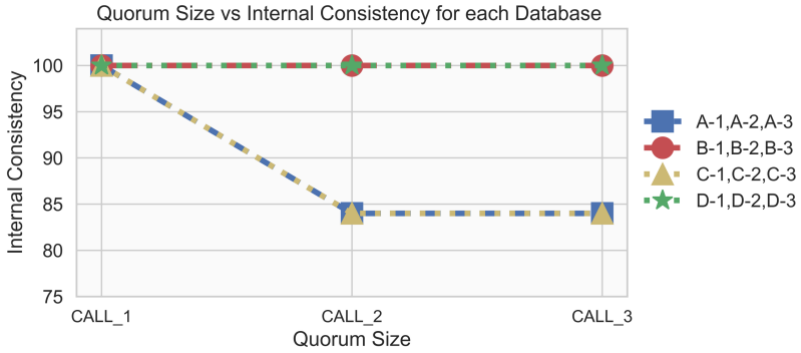
Database Implementation A and D: The average latency is lowest (*0.333ms for A and 0.354ms for D*) when we only consider the very first response it receives (CALL_1). Subsequently for CALL_2 Quorum Size this increase to (*0.481ms and 0.526ms*) and for CALL_3 Quorum Size it increases to a max of (*0.841ms and 0.904ms*). Since we can observe that the values are exponentially increasing as Quorum Size increases, we can conclude that these two databases have *Ragged Latency distribution*.

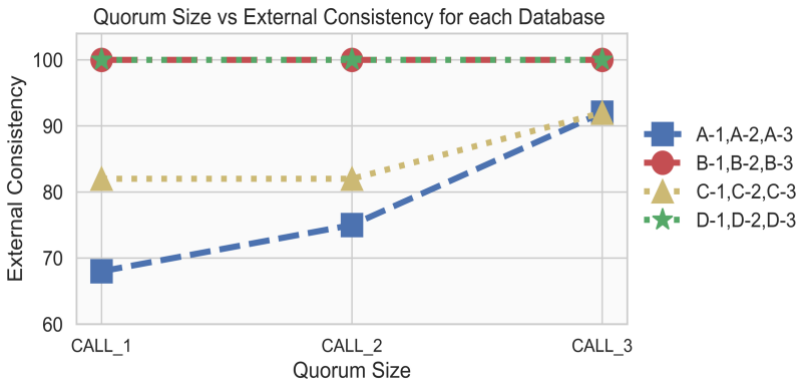
Database Implementation B and C: In this case, the average latency is the lowest (*0.239ms for B and 0.134ms for D*) for the CALL_1 Quorum Size and uniformly increases as the Quorum Size increases (i.e for CALL_2 Quorum Size it increases to (*0.34 ms and 0.182 ms*) and CALL_3 Quorum Size it increases to a max of (*0.415 ms and 0.235 ms*). Therefore, we can conclude that the database has a *Uniform Latency distribution*.

Verdict: On an overall scale, *C has the best average latency time for all possible quorum* sizes followed by B and D has the worst average latency time followed by A database implementation. Therefore, if latency is a crucial part of your design, based on these two factors (latency, Quorum Size) you have to choose C database implementation. It can also be observed that in all cases, the *Latency increases as the Quorum Size increases*.

3.2. Consistency VS Quorum Size:

As you can see in the line graph plotted between Quorum Size and Average Consistency (both Internal and External) for each database implementation, Databases B and D have maximum internal and external consistency as compared to A and C. Now let's access each database's Internal and External consistency vs Quorum Size relationship.





Database Implementation A and C:
The overall Internal and External consistency for A and C database implementation is not too high. It obviously has 100% internal consistency for CALL_1 Quorum Size but falls to 84 % as we increase the size to CALL_2 and CALL_3. Whereas in the case of External consistency, it performs poorly for CALL_1 Quorum Size with just 68% and 85% each. But interestingly, for CALL_2, the percentage falls to 82% for C and increases to 75% for A. Subsequently, it increases for both to 92% for CALL_3 Quorum Size.

Based on these observations, we can say that these database implementations are fairly *inconsistent*

Database Implementation B and D:
Both Internal and External consistency for B and D database implementation is 100% for all possible Quorum Sizes. Therefore we can conclude that they are fairly *consistent.*

Verdict:
On an overall scale*, B and D are highly consistent* (100%) for all possible quorum sizes whereas *A followed by C are very inconsistent* in nature. Therefore, if consistency is an integral requirement for an application, based on just these two factors (Consistency, Quorum Size) we can go for B and D database implementation. Unlike latency, for the above A and C databases, internal consistency largely remained the same for CALL_2 and CALL_3 quorum sizes. Whereas on average, it either remained the same or increased with increase in quorum size.

3.3. Consistency vs Latency

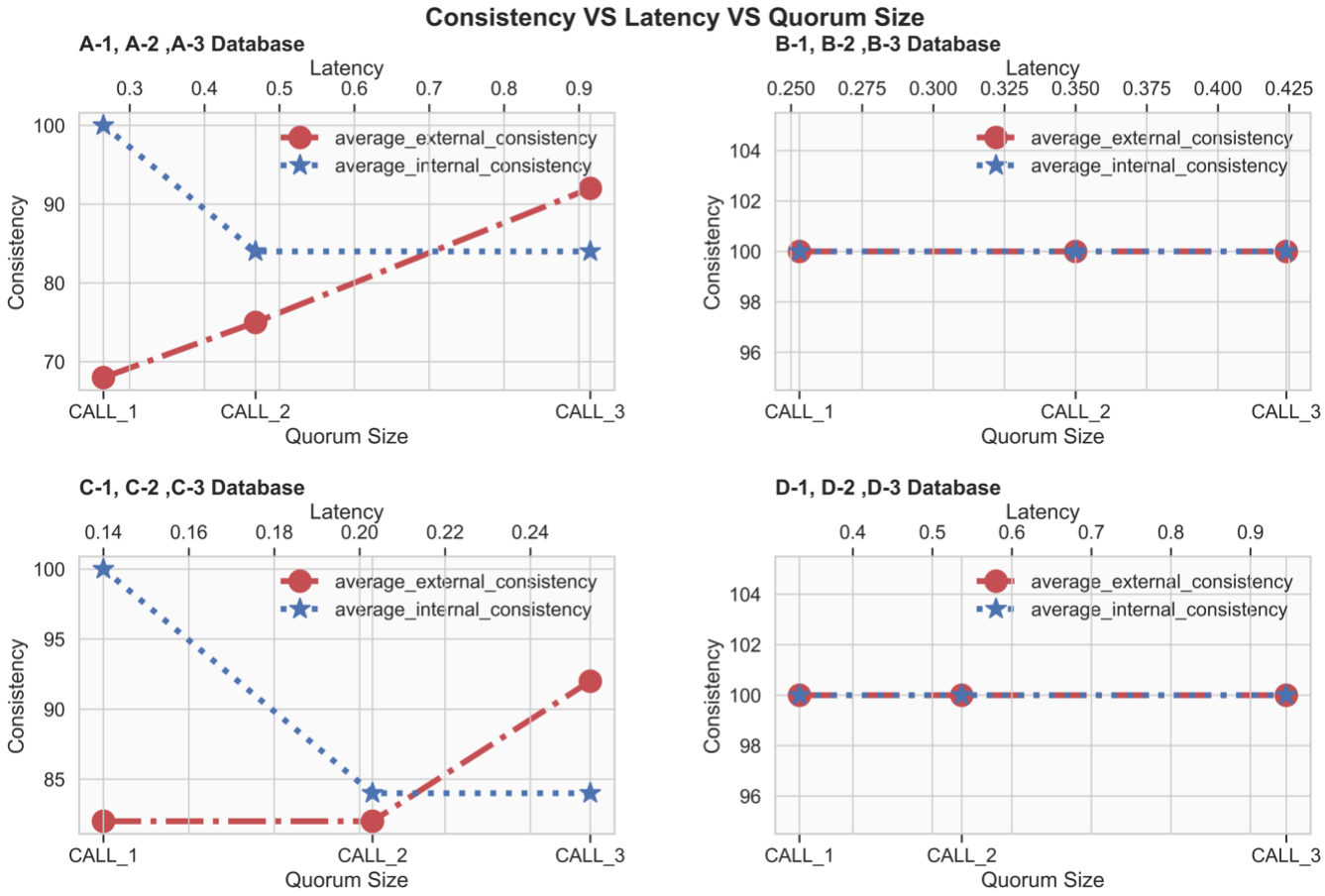
Consistency vs Latency Analysis			
Consistency	**Average Latency**	**Average External Consistency**	**Average Internal Consistency**
Consistent	0.46	100	100
Inconsistent	0.37	82.33	89.33

The above table provides us information about the relationship between Consistency and Latency for all database implementations. Even though this particular analysis doesn't consider all the influencing factors and just focuses on consistency and latency values, on a very broader scale we can observe that for *higher consistency (in this case 100%), the average latency (0.46ms) is more as compared to a fairly inconsistent database implementation (0.37ms)*. This observation is in line with the *PALEC Conjecture* where they state that in case of a partition (P) the system either chooses availability (A) vs consistency (C) or latency (L) vs consistency (C).

3.4. All factors in perspective: Consistency vs Latency VS Quorum Size:

The below 4 plots compare all the three tradeoff's (*Consistency, Latency* and *Quorum Size*) for all the 4 database implementations. This analysis will help you choose the best design with the least amount of tradeoffs by keeping all factors in perspective. So let's analyze each database and finally summarize our findings.

Database Implementation A:
This is a case of an *Inconsistent Database having Ragged Latency*. We can observe that the external consistency uniformly increases as well as Internal consistency drops to 84% and remains the same as the latency and quorum size increases. In case you want low latency (0.33ms) with CALL_1 Quorum Size then you would be trading off external consistency for it (68%). But if you want a fairly externally consistent system with CALL_3 Quorum Size, then you would be trading it for high latency (0.863ms). The average case would have a CALL_2 Quorum Size with 75% external consistency and 0.481ms latency.



Database Implementation C:
This is a case of an *Inconsistent Database having Uniform Latency*. We can observe that the external consistency reaches only a high of 92% for a CALL_3 Quorum Size but has a very low latency metrics ranging from 0.134ms to 0.235ms for all CALL_3 Quorum Sizes. This database implementation would be good if you don't mind the system being a little inconsistent to have a tradeoff for low latency.

Database Implementation B and D:
B is an example for *Consistent Database having Uniform Latency* whereas D is an example for *Consistent Database having Ragged Latency*. Both the database implementations achieve 100% consistency (internal and external) but differ only when it comes to latency. Here, B has the best implementation with just 0.239ms of average latency and 100% consistency for CALL_1 quorum size. In the case of D, it has 0.354ms of average latency for CALL_1 quorum and increases exponentially with Quorum Size.

4. Conclusion:

As we have seen in our analysis, each database has a unique take on *latency vs consistency tradeoff*. But by properly evaluating all the factors (like we did) a system designer can make an informed decision by finding the right database implementation with the least amount of tradeoffs. Below is a table that on a broad scale summarizes our results. This table doesn't necessarily shed light on quorum size (which is important as seen in the D database implementation), it fairly captures other factors for mean quorum size results.

Overall Analysis					
Database	**Consistency**	**Latency**	**Average Latency**	**Average External Consistency**	**Average Internal Consistency**
A-1,A-2,A-3	Inconsistent	Ragged	0.559	78.33	89.33
B-1,B-2,B-3	Consistent	Uniform	0.33	100	100
C-1,C-2,C-3	Inconsistent	Uniform	0.18	86.33	89.33
D-1,D-2,D-3	Consistent	Ragged	0.60	100	100

Based on this table and the overall analysis conducted, database implementation *B with CALL_1 Quorum Size does a great job by having the least tradeoffs between consistency and latency*. All the other implementations have tradeoffs that need to be evaluated based on application needs and requirements.

5. Appendix:

Link to the Jupyter Notebook where this quantitative analysis was performed.

https://csil-git1.cs.surrey.sfu.ca/abejju/cmpt-756-systems-for-big-data-/tree/master/A2

6. References

[1] Consistency or Latency? A Quantitative Analysis of Replication Systems Based on Replicated State Machines - Xu Wang, Hailong Sun, Ting Deng, Jinpeng Huai – IEEE 2013

[2] Fine-tuning the Consistency-Latency Trade-off in Quorum-Replicated Distributed Storage Systems - Marlon McKenzie, Hua Fan, Wojciech Golab - 2015 IEEE International Conference on Big Data (Big Data)

[3] Consistency Tradeoffs in Modern Distributed Database System Design - Daniel J. Abadi, Yale University - IEEE Computer Society 2015

[4] On the tradeoff of availability and consistency for quorum systems in data center networks Xu Wang, Hailong Sun, Ting Deng, Jinpeng Huai – Computer Networks – Computer Networks - Elsevier

[5] Ozsu, Tamer M; Valduriez, Patrick (1991). "12". Principles of distributed database systems (2nd ed.). Upper Saddle River, NJ: Prentice-Hall, Inc. ISBN 978-0-13-691643-7