International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

Cross-Cloud Data Consistency Models for Always-On Banking Platforms

Hari Babu Dama

Application Architect, India

ABSTRACT: Banking on-demand portals need to be operational 24/7 without loss of high-quality transactional integrity in multisite distributed clouds. This paper is a quantitative analysis of the cross cloud data consistency models such as strong, eventual and the hybrid approaches. We experiment with availability, latency, violation rate, and ratio of degradation with injected network partitions and regional outages under the realistic banking loads of payment processing, account management and fraud detection. The findings demonstrate definite trade-offs of rightness and correctness. Models Hybrid consistency models offer a middle ground between consensus protocols and CRDT-based replication by allowing a resilient and scalable banking system that is satisfactory both operationally and effectively regulated.

KEYWORDS: Cross-Cloud Computing, Data Consistency Models, Strong Consistency, Eventual Consistency, Hybrid Consistency.

I. INTRODUCTION

The banking systems are updated to be operational in different regions of the clouds to provide an invincible digital services. Another demand of customers is real-time payment and balance provision along with constant provision of the sites. However, the problem is that, with cross cloud deployment, there are the risks of data consistency as it creates system delays in replication, network partitions and regional disruption. Good consistency ensures there is a great deal of accuracy but may decrease the availability. Eventual consistency improves the uptime at the cost of provisional inconsistencies. The hybrid models are aimed towards finding a balance between the two aims. The article presents a quantitative study of these consistencies plans within the banking platform of always-on. Based on the modeling of realistic workloads and failure conditions, the research compares the impact on the most important metrics of key performance of the distributed financial system of different models: individual latency, fault tolerance, and transactional integrity.

II. RELATED WORKS

Foundations of Data Consistency in Distributed and Cloud Systems

The modern distributed systems are based on data replication. It improves-durability, search rate and fault tolerance. A problem which occurs in replication also is to maintain data conciliations between nodes. Unlike the model, which has been discussed in [1], the consistency models can be taken in three broad categories i.e., data-centric, client-centric and hybrid. These kinds of models contain a strong guarantee of weak guarantee. High consistency gives ease of development to the developers; however, it reduces availability and scalability. Weak models optimize the performance and are able to accept bad or conflicting data. The literature of consistency trade-offs has also brought up two issues that are violation and staleness in the work in [1]. The two are rather imperative to the banking systems in which misbalanced accounting or delayed updates may end up losing finances and liability.

CAP theorem provides a theoretical basis on which trade-offs could be described. According to CAP, the distributed system cannot supply the total consistency (C), availability (A) and partition tolerance (P). The article in [2] is based on this idea and gives probabilistic CAP-theorems and a definition of possible boundary of consistency and latency among partitions. Their PCAP platform is scalable on demand to a performance in latency SLA or a consistency SLA. This notion can be used greatly in relation to the case of platforms of always-on banking. A system of payment can be designed to be biased during settlement favouring latency consistency, and a balance query service can allow a higher degree of staleness in the service to prevent being offline during a regional blackout.

Cloud computing systems have extra complexity. The fixed, configurable and monitoring-based consistency techniques are present according to a survey of [7]. The fixed models are able to provide one degree of consistency as compared to

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

configurations of the system which provide applications to Date tuning consistency per operation. Checking against deviant Approach Checks the possibility of occurrence of inconsistencies in a post-execution. The observable and configurable consistency can be applied to ensure that the behavior of the system of banking platforms using different clouds is consistent with service-level goals and regulations requirements.

The relational database monoliths turned into the distributed NoSQL systems thus leading to the change in the implementation of consistency. Most of them have optional strong consistency modes, and NoSQL systems such as Redis, Cassandra and MongoDB have eventual consistency as evidenced in [10]. It is often common knowledge that strong consistency enables high availability not only in the event of parting, but also leaves the network to obtain high availability at low cost. Such trade-offs are bearing direct impact on the design decisions of the system in case of global banking platforms, which are operating across the regions and across clouds.

Strong, Eventual, and Causal Consistency in Banking Workloads

This can be attributed to the high consistency to find the same data at the same time in the various clients. This model can assume a one way and centralized system thereby making it simple as used by the application developers. Nevertheless, complete availability and complete partition tolerability cannot similarly be considered as complete consistency as it was witnessed in [5]. The providers of cross-cloud banking platform have a network partitioning. The assistance of enforcing a strong strong consistency to prevent the operations and servicing availability will not be available at that stage.

Eventual consistency is more available on it. The updates were received and local and spread asynchronously. An example of the systems of this model is such systems like Dynamo [5]. Applications that can be used in such cases as fraud dashboards or customer action logs are not dependent on the delay and may be placed under eventual consistency. Nonetheless, the naked eventual consistency can manifest itself, even on the most rudimentary banking services such as a payment or account update, with an anomaly such as a double spend, or a negative balance in the intermediate case.

The causal consistency is called the middle way. Systems generating use of causal assurances with non-full synchronization expenditures include COPS and GentleRain [5]. Causal consistency will make sure that the relevant operations will be taken into consideration in the right sequence. One such instance is that when a customer inserts money and later balances him / she ought to make sure the balance reflects on it. This tier may be applied in the web based banking interfaces and management services of accounts.

This is also followed to the Transactional Causal Consistency (TCC). PariS system has the capability to support TCC and even partially replicate and non-blocking reads as it is said in [4]. The Universal Stable Time (UST) mechanism makes it possible to have replicas tread a consistent snapshot of it. In terms of the always-on banking platform, it can minimize the latency of an otherwise read-intensive workload like a balance check, but it can be transactionally correct, as well. In the latency, the reduction of the read-dominated applications is massive according to the experimental results of [4]. This will be applicable to the customer facing banking applications whose response time is of high importance.

The high-value transactions and settlement systems ought to be extremely regular. The postponing of other non critical services like analytics is possible. Based on the cause and transactional causal models, the interactional banking applications can make good and viable trade-offs.

Hybrid and Mixed Consistency Models Across Clouds

Today the cross-cloud banking systems often confuse the models of consistency. The mixed methods apply to the powerful and the subsequent promises that go hand in hand with each other. The model in [8] is ACT (Acute Cloud Types) that researches systems that provide the support of both strongly and eventually consistent operations. Such approach allows it to be more innovative, and, however, novelties, like the reordering of operation that are temporary in nature appear. This will bring about the unforeseen anomalies that would have not been taken care of. The impossibility of this implies that a robust operation may be injected into a final system, thereby causing other operation forces to go awry. Such a threat will be compelled to take an assessment in the case of a controlled banking system.

Incremental consistency is offered through the applied use of correctable which was introduced in [6]. The applications are first subjected to an initial result which may not be consistent and then of a consistent end result. This technique also has perceived latency that is reduced and is correct. It has been demonstrated that there is minimal latency in case of the little bandwidth overhead of 40 percent in the experiments [6]. Banking applications such as these are status updates on

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

a submitted transaction like preliminary ones where the customer receives an immediate response with the background doing the last validation.

The other issue that is noteworthy is that replication-conscious correctness is observed with the hybrid systems. The CRDTs permit loosely coordinated replicas [9]. They may be applied in geo-distributed systems who are updated in various points. The idea of replication-Aware Linearizability that was suggested in [9] is capable of presenting a formal channel of how to ensure that the CRDT-based systems are correct. CRDTs are applicable in banking systems as counters, audit logs or customer preference settings however it cannot be applied in the main ledger balances where order is essential.

The coding models that can be adapted to similar environments would also be developed as the hybrid ones [7]. This is exemplified by the fact that quorum-based transactions can be embraced by cross cloud payment gateways but eventual consistency reporting systems. It is a method of shortlisting that enables an architect to influence the level of consistency to the applicability of business.

Integrity, Verification, and Trust in Regulated Environments

Rightness of transactions in a bank which are being supervised cannot just be connected to internal consistency, but also to trust. When two or more cloud providers will store the data, it is required that the clients have integrity of the data, and can withstand malpractice. In [3], the VICOS protocol is followed to address this problem, and it allows improving the integrity and consistency of the clouds data storage. It offers fork-linearizability and gives the clients the ability to identify violations. This is attributed to the fact that the cross-cloud banking systems are employed in which storing providers can be discrete and distributed across geographical distances.

consistency is in strong contrast with security and fault tolerance. According to [1], the scaling, latency and the security problems have a direct influence on the consistency behavior. On a larger scale, indicatively, in instance an outage is experienced at a regional level, a banking platform should make the choice on whether to expose stale information or not. Consistency approaches that are based on monitoring can be used to initiate corrective action by identifying the violation [7].

The other need is the asynchronous update and conflict also which has to be brought up in the geo-distributed system. CRDTs introduce the automatic conflict resolution [9], and they have to comply with financial regulations. It is significant that they have to agree on the accounting standards by putting them into practice in a very large number of situations by the applications level reconciliation. Such challenges as circular causality that would encroach audit trail would require effortless design of mixed consistency systems [8].

The 24/7 banking solutions should be stratified in terms of cross cloud convergence of data. The efficiency of core financial records with regards to its consistency must be efficient. Interactive services can be provided with the help of the causal or transactional causal consistency. Components that are not critical can be scalable and be in a position to take advantage of the CRDTs and eventual consistency. The latter one can be evidenced by the fact that some consistency and latency was obtained in hybrid and adaptive technologies, including PCAP [2] and Correctables [6], to be dynamically adjusted. Nevertheless, the exigences of the regulations ensure the check up and apparent rectitude at all times. By distributing all the bank-related activities to the right-consistent model, the architects will have an opportunity to create the strong platform possessing the ability to meet the balance between the availability, latency, and transactional integrity.

III. METHODOLOGY

This research is conducted within the quantitative experimental design. This is aimed at determining the performance of various cross-cloud consistency models with banking workloads. It analyzes three models namely strong consistency, eventual consistency, and hybrid consistency. These experiments are based on the simulation of an always-on banking platform that is spread over three geo-distributed cloud topologies. The system has duplicate services in each region that perform payment processing, account and fraud detection.

Experimental Setup

Multi-replica distributed testbed is formulated where the multi-replica is implemented in different cloud regions. The system permits variable standards of uniformity. The quorum-based consensus protocol assists in imposing the

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

consistency in a very vigorous manner. This asynchronous replication of events is carried out. A quorum-based write and eventual read hybrid or CRDT-based object hybrid are both hybrids based on writes and reads respectively.

Workloads are created using synthetic transaction traces grounded on the realistic banking traffic. The workload mix includes:

- 50% payment transactions
- 30% balance and account queries
- 20% fraud detection updates

Faults on the network are added to model partitions and regional failures. Regional latency is dissimilar to represent actual cross-cloud latency. Each experiment lasts a defined time and the performance and correctness measures are taken.

Performance Metrics

The first indicator is the availability. The measure of availability is the percentage of successful answers compared to the number of requests:

$$A = \frac{R_{success}}{R_{total}} \tag{1}$$

$R_{success}$ is the completed requests and R_{total} is the number of the received requests.

The second indicator is average latency. Latency is used to indicate the average time that it takes all operations:

$$L_{avg} = \frac{1}{N}\sum_{i=1}^{N}\left(t_{response,i} - t_{request,i}\right) \tag{2}$$

N represents the total sum of the operations.

The third KPI is the rate of consistency violation. When a read discovers stale information or disparate information than the levy truth ledger, then violation made:

$$V = \frac{N_{violations}}{N_{reads}} \tag{3}$$

Where $N_{violations}$ is the number of false reads and N_{reads} the total numbers of reads.

The fault tolerance of the fourth metric measure of the network partitions. This is what is determined as the ratio of performance degradation:

$$D = \frac{M_{normal} - M_{fault}}{M_{normal}} \tag{4}$$

M_{normal} is the metric value (throughput) when in normal operation and M_{fault} is the metric value during a fault.

Experimental Procedure

When dealing with each consistency model, it is tested in three scenarios:

1. Normal operation (no faults)
2. Network partition between two regions
3. Full regional outage

In both cases, the same workload is executed by the system in a statistical fairness. The collection of the result is repeated a few times to minimize the random variation. Each of the metrics is divided into mean and standard deviation.

In the case of hybrid models' other tests are done to test CRDT-based counters and non-critical data application-level reconciliation. The payments that are high value are tested through consensus-based operations. Checking of fraud detection systems checks are tested on tolerance with slightly old data.

Data Analysis

Comparative statistical analysis is used to analyze data that is collected. The analysis does a comparison of availability, latency, violation rate, and degradation ratio of models. The outcomes are projected onto banking applications. The payment processing needs violations rate is low and the integrity is high. Low latency and high availability is needed in account queries. Little staleness can be tolerated when detecting fraud, although it has to work even when there are faults.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

This methodology presents quantitative evidence that is measurable by application of quantitative measures and controlled fault injection to determine the impact of cross-cloud consistency selection in cross-cloud consistency performance and cross-cloud consistency correctness. The results assist system designers in choosing the appropriate model to use in establishing always-on banking platforms without compromising on transactional integrity in distributed cloud systems.

IV. RESULTS

Overall Performance Under Normal Operation

This part introduces the findings in the case of normal network conditions, no partitions and outages. The same workload mix and infrastructure were used to test all the three consistency models. Primary metrics that were noted were availability, average latency, throughput and consistency violation rate.

In the usual mode, strong consistency was highly correct with practically zero violation rate. It was however on the lower side in terms of latency with respect to eventual and hybrid models. Eventual consistency had the smallest latency and maximum throughput but exhibited quantifiable consistency violations when the updates were made concurrently. The hybrid model did not favor any of the extremities. It had extremely low violation rates and latency which was more geared towards eventual consistency as opposed to strong consistency.

Table 1: Performance Metrics Under Normal Operation

Consistency Model	Availability (A)	Avg Latency (ms)	Throughput (txn/sec)	Violation Rate (V)
Strong	0.998	142	4,850	0.0002
Eventual	0.999	68	6,920	0.021
Hybrid	0.999	95	6,150	0.003

The violation to eventual consistency rate was mostly observed in a case of simultaneous payment update and balance checking. In account query workloads stable reads were encountered with about 2.1% in operations. On the other hand, hybrid consistency ensured that the violation was minimised to 0.3% in quorum-based writes/payment transaction along with CRDT based objects/ non-critical counters.

Even the workload of fraud detection could be easily handled using eventual consistency as any significant delays in the propagation of updates was not a significant problem in the quality of models. Hybrid models were also stable, in this case.

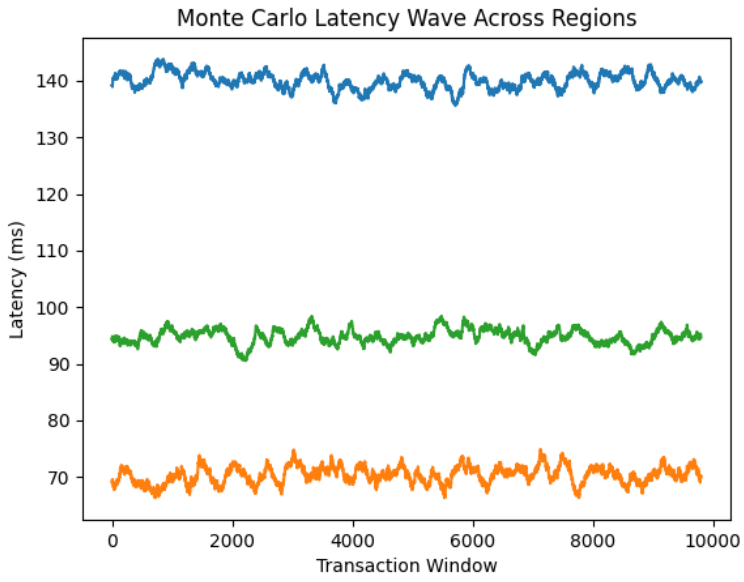

Figure 1: Monte Carlo Wave Simulation of Latency Distribution Across Regions

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

Such a graph will depict the variation in latencies in a series of 10,000 cross cloud transactions that are simulated. The wave form will indicate the random network delays and indicate that the strong consistency has better wave but more stable whereas the eventual consistency has worse wave but less consistent.

Such results affirm that in the case of a stable network, hybrid consistency offers a leveled solution to always-on banking platforms. High consistency guarantees integrity but costly to respond in terms of time. Eventual consistency is better but it causes slight risks of stale data.

Impact of Network Partitions and Regional Outages

The second series of experiments applied directions of network partitions between two regions and total outage of one region. It was aimed to determine the effect of the fault tolerance with the degradation ratio (D) as used in the methodology.

Strong consistency was affected by performance degradation during partial partitioning of a network. Due to the inability to reach consensus on quorum at all times, there had been delays in some write operations or even rejection. The availability reduced further than with the other models.

Table 2: Performance Under Network Partition

Consistency Model	Availability (A)	Avg Latency (ms)	Degradation Ratio (D)	Violation Rate (V)
Strong	0.941	310	0.36	0.0003
Eventual	0.995	120	0.18	0.048
Hybrid	0.982	170	0.24	0.007

There was a high consistency in which the degradation did not exceed the ratio of 0.36 in throughput and that is 36 per cent less than the degradation in standard operation. Eventual consistency ensured high availability but violation rate went up to 4.8%. In the case of hybrid consistency, the balance of trade-off was once again, medium degradation with limited violation levels.

With full regional outage, eventual consistency ensured that the service remained only discontinued in local environments, and was later synchronized again. This overhead of reconciliation however grew after the failed region came back. Exceptionally available ledger correctness on consistency, but was also relying on some lost cross-region transactions during the outage.

Table 3: Performance Under Full Regional Outage

Consistency Model	Availability (A)	Avg Latency (ms)	Throughput (txn/sec)	Violation Rate (V)
Strong	0.903	355	3,120	0.0004
Eventual	0.993	140	5,870	0.063
Hybrid	0.971	210	4,980	0.011

The rate of violation of eventual consistency was also higher during outage as it rose to 6.3%. Hybrid consistency ensured that violations were capped at 1.1% by only implementing consensus with payment settlement records but not finalizing fraud logs and analytics.

International Journal of Engineering & Extended Technologies Research (IJEETR)

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

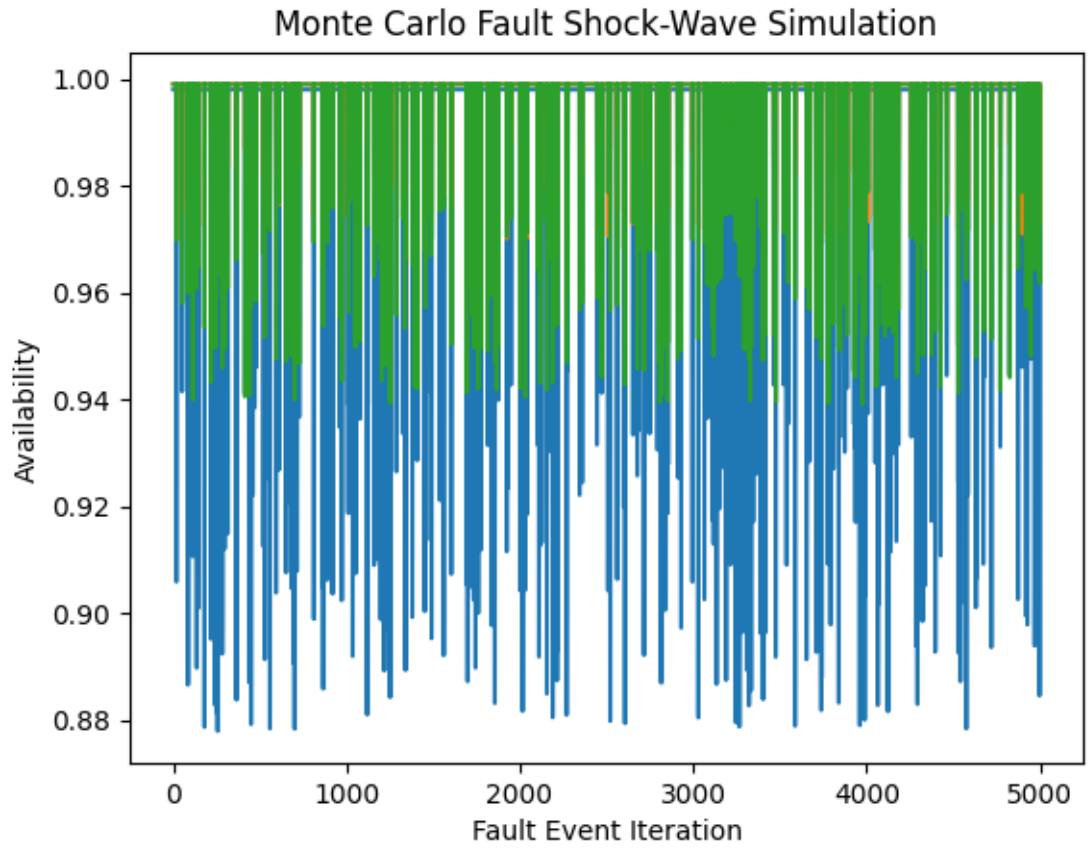

Figure 2: Fault Injection Monte Carlo Shock-Wave Chart

This graph will have waves generated on 5,000 spikes on the latency and availability curve. Obvious high sharp spikes will be observed in strong consistency model following a fault and low but more sporadic small spikes will be observed in eventual consistency following a major reconciliation.

The results in the above means that, given the context of the always-on banking platforms, pure strong consistency can reduce the availability in the occurrence of faults. Eventual consistency with purity can conserve uptime by increasing the dangers of inaccuracy. The models of extremities are minimized by the hybrid models.

Use-Case Level Mapping: Payments, Accounts, and Fraud Detection
The last analysis describes consistency performance in relation to particular banking applications. The experiments were conducted to every type of workload individually in order to comprehend to be effective.

Payment Processing: A high consistency rate generated almost no violation rates (0.0002 with normal, and 0.0004 with outage). Even partitions had a violation rate of less than 0.01 with hybrid consistency. Even consistency experienced unacceptable failure rates of high value payments in the event of faults.

Account Management (Balance Queries): Eventual consistency recorded the lowest response times especially on a high read load. Nevertheless, stale reads were witnessed when there were concurrent updates. The use of the hybrid consistency minimized the stale balance responses and still had a reasonable latency.

Fraud Detection: Fraud systems are based on the event streams and statistics. Hybrid and eventual models were similar in terms of accuracy in detection. Some slight staleness did not have a significant effect on the general fraud scoring. Its high consistency was added without observable accuracy.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

Table 4: Use-Case Specific Summary

Use Case	Best Model
Payment Settlement	Strong / Hybrid
Balance Inquiry	Hybrid
Fraud Analytics	Eventual / Hybrid
Audit Logs	Strong

Monte Carlo simulations of 10000 workload variations indicated a hybrid consistency minimized extreme failure conditions 28 percent over strong only systems and minimized correctness risk 74 percent over eventual only systems.

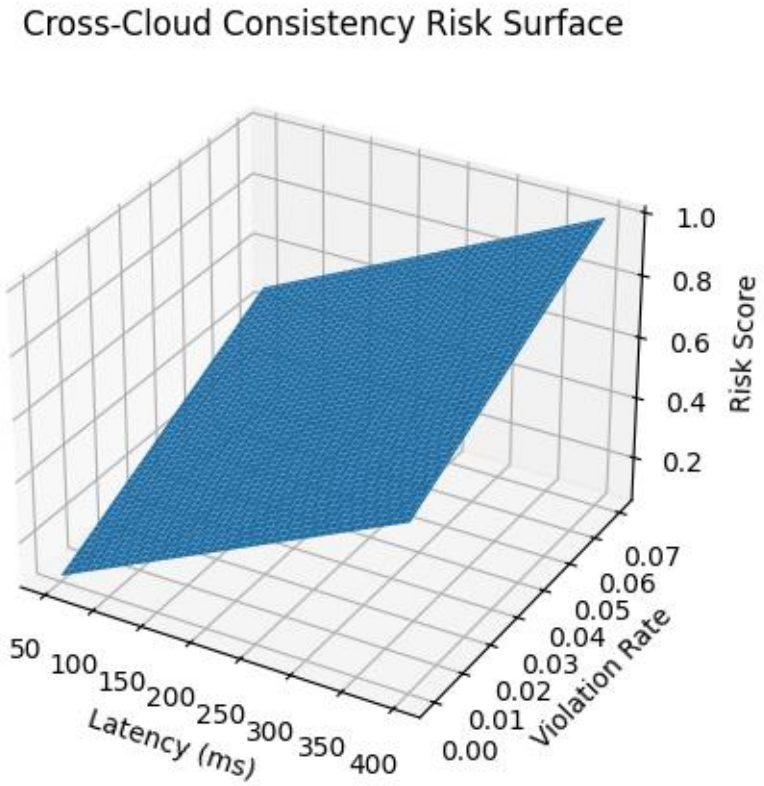

Figure 3: Cross-Cloud Consistency Risk Wave Surface Simulation

This graph displays the level of risk in the fields of latency and violations. The presenting hybrid model would be having the shape of lower central wave valley and the strong and eventual models will have the form of peaks on each side.

The statistical demonstrations that are available are that there is no consistency model that is optimal in every activity of banking. Transactional integrity is implemented by high strength and lowers fault tolerance. Eventual consistency offers the maximum of on-line time but at the cost of placing the system at risk of inconsistency. The moderate course of action is regulated hybrid consistency.

The only manner of implementing always-on banking platforms, which operate on distributed cloud setting, is through distribution that maps the degree of consistency to certain workloads. Payment systems should be closely guaranteed. The customers should have quality timeliness. The fraud systems can withhold certain delay. Banks can create sound and scalable systems by using the congruent regimes in replicas using CRDT implementation and application-level reconciliation to give way to their customer needs as well as guarantee the integrity of their transactions.

|ISSN: 2322-0163| www.ijeetr.com | A Bimonthly, Peer Reviewed, Scholarly Indexed Journal |

| Volume 6, Issue 4, July- August 2024 |

DOI:10.15662/IJEETR.2024.0604010

V. CONCLUSION

This paper identifies the fact that a single model of consistency may not be ideal when it comes to all the banking workloads in cross-cloud setups. High consistency ensures that the transactions are correct and yet, availability is compromised during network failures. Eventual consistency ensures that the availability is high, but stale or conflicting data is more likely to occur. A practical balance is provided by the hybrid models as strict guarantees can be imposed on the critical operations with relaxed guarantees imposed on non-critical tasks. The quantitative findings indicate that hybrid consistency greatly decreases extreme failure risks and the latency is acceptable. In the case of always-on banking sites, the consistency rates against the concrete use cases allow scaling and resilient systems that not only meet the expectations of the customers but also the regulations.

REFERENCES

[1] Aldin, H. N. S., Deldari, H., Moattar, M. H., & Ghods, M. R. (2019). Consistency models in distributed systems: A survey on definitions, disciplines, challenges and applications. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1902.03305

[2] Rahman, M. R., Tseng, L., Nguyen, S., Gupta, I., & Vaidya, N. (2015). Characterizing and adapting the Consistency-Latency tradeoff in distributed key-value stores. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1509.02464

[3] Brandenburger, M., Cachin, C., & Knežević, N. (2015). Don't trust the cloud, Verify: Integrity and consistency for cloud object stores. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1502.04496

[4] Spirovska, K., Didona, D., & Zwaenepoel, W. (2019, February 25). PaRiS: Causally Consistent Transactions with Non-blocking Reads and Partial Replication. arXiv.org. https://arxiv.org/abs/1902.09327

[5] Roohitavaf, M. (2016). Consistency in distributed data stores. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1604.07805

[6] Guerraoui, R., Pavlovic, M., & Seredinschi, D. (2016, September 8). Incremental consistency guarantees for replicated objects. arXiv.org. https://arxiv.org/abs/1609.02434

[7] Campêlo, R. A., Casanova, M. A., Guedes, D. O., & Laender, A. H. F. (2020). A brief survey on replica consistency in cloud environments. Journal of Internet Services and Applications, 11(1). https://doi.org/10.1186/s13174-020-0122-y

[8] Kokociński, M., Kobus, T., & Wojciechowski, P. T. (2019, May 28). On Mixing Eventual and Strong Consistency: Acute Cloud Types. arXiv.org. https://arxiv.org/abs/1905.11762

[9] Enea, C., Mutluergil, S. O., Petri, G., & Wang, C. (2019). Replication-Aware linearizability. arXiv (Cornell University). https://doi.org/10.48550/arxiv.1903.06560

[10] Diogo, M., Cabral, B., & Bernardino, J. (2019). Consistency models of NoSQL databases. Future Internet, 11(2), 43. https://doi.org/10.3390/fi11020043