Journal of Computer Science and Technology Studies
ISSN: 2709-104X
DOI: 10.32996/jcsts
Journal Homepage: www.al-kindipublisher.com/index.php/jcsts

JCSTS
AL-KINDI CENTER FOR RESEARCH AND DEVELOPMENT

| RESEARCH ARTICLE

Geographic Consensus Layer: A Hybrid Consistency Approach for Distributed NoSQL Systems

Vaibhav Haribhau Khedkar
Marshall University, USA
Corresponding Author: Vaibhav Haribhau Khedkar, **E-mail**: reachvaibhavk@gmail.com

| ABSTRACT

The Geographic Consensus Layer (GCL) presents a novel hybrid consistency architecture for distributed NoSQL database systems, specifically addressing the challenges of achieving strong consistency in geo-distributed Apache Cassandra deployments. By introducing a decoupled control plane implementing Multi-Paxos across regions, GCL provides linearizable consistency guarantees exclusively for critical operations while preserving high performance for non-critical workloads. This article effectively isolates the unavoidable latency penalties of cross-region consensus to only those operations that genuinely require strong consistency. Experimental evaluation demonstrates that GCL maintains near-baseline throughput while ensuring linearizability where needed. Architecture includes metadata-cavalry unanimous, operation batching, and pipeline processing, such as adaptation to reduce overheads. The GCL represents a practical solution for the fundamental trade-bands imposed by the CAP theorem, enabling the organizations to deploy globally distributed globally without renouncing continuity for wider operations or comprehensive systems.

| KEYWORDS

Distributed databases, NoSQL, hybrid consistency, Paxos consensus, geo-distribution, linearizability

| ARTICLE INFORMATION

ACCEPTED: 20 October 2025 **PUBLISHED:** 06 November 2025 **DOI:** 10.32996/jcsts.2025.7.11.32

1. Introduction

The distributed NOSQL database system has become a basic infrastructure for modern applications requiring global scale and high availability. As a premier example, the architecture of apache cassandra is fundamentally shaped by **CAP** theorem. It is designed as an AP (Availability and Partition Tolerance) system, which necessitates that it relinquishes the guarantee of strong consistency during network partition. This design decision inherently favors low-latency data writes and high uptime across the cluster. This theorem establishes that the distributed systems can provide most of the three properties: consistency, availability, and partition. As the Brever expresses in its seminal PODC keynote, the system should renounce a guarantee to fully achieve two, giving rise to fundamental design decisions, shaping the distributed architecture [1]. While the Cassandra provides tunable consistency levels, applying global strong consistency requires synchronous cross-detectioner communication, which introduces adequate delays and punishment (usually more round-travel time between 100ms or areas). This synchronous barrier reduces the throughput of serious writing, resulting in a decline in performance compared to the final consistency model.

The boundaries of existing approaches in the implementation of Cassandra are particularly clear. Even Light-weight transactions (LWTS), the underlying mechanisms of the Casundra face the inaccessible, delayed obstacles when deployed globally, to achieve linearity through the Paxos-based protocol. According to the widespread performance evaluation of their Cassandra penetration tests in many AWS regions of Netflix, throughput capabilities are dramatically reduced when there is a strong consistency in geographical boundaries. This benchmark demonstrated that the 30-node Cassandra cluster could write 1,196,398 per second

using the eventual consistency within the same area. However, when extended to cross-field operations with a strong consistency guarantee, throughput decreased by about 83%, falling to 203,387 operations per second. This performance found the quorum or all consistency levels in areas became even more pronounced, where the P99 delay increased from 14.2ms to 178.5ms, which represents an increase of 1,156% in the tail delay [2]. These conclusions highlight the important practical challenges of achieving strong consistency in the distributed geographical environment.

The core challenge lies in achieving strong consistency guarantees for critical operations without compromising the overall system performance characteristics that make NoSQL databases attractive in the first place. This research introduces the Geographic Consensus Layer (GCL), a hybrid consistency architecture designed to address these limitations. The fundamental insight driving this approach is the recognition that not all data operations require the same consistency guarantees. By providing a mechanism that offers strong consistency guarantees exclusively for explicitly marked critical operations, while allowing non-critical operations to bypass these mechanisms entirely, the GCL effectively isolates the performance overhead associated with cross-region consensus.

2. Theoretical Framework and Related Work

The theoretical underpinnings of distributed consistency models have been extensively studied in the literature. The CAP theorem, formally proven by Gilbert and Lynch, established the fundamental impossibility of simultaneously achieving Consistency, Availability, and Partition tolerance in distributed systems. Their seminal paper formalized Brewer's conjecture mathematically, demonstrating that no distributed system can provide all three guarantees at once. Through rigorous proof, they showed that even an optimal algorithm cannot maintain both consistency and availability during network partitions. Their work analyzed two models: an asynchronous network model where consistency and availability cannot be simultaneously satisfied, and a partially synchronous model where achieving all three properties is possible only during periods without partitions. This theorem has profound implications for distributed database design, as systems must explicitly sacrifice either consistency or availability when networks partition [3]. This basic function continues to shape the system architecture decisions for geographically distributed systems working in areas with unavoidable network disruptions.

Building on this foundation various strategies have been developed to manage system trade-offs based on application needs. The PACELC theorem, introduced by Daniel J, Abadi extended the CAP model by emphasising that when a network Partition (P) is absent, a distributed system still faces a critical trade-off between minimizing Latency and ensuring consistency. Abadi demonstrated that this extended taxonomy better classifies distributed database systems by recognizing that many systems sacrifice consistency for latency benefits even when no partitions exist. His analysis categorized systems like Amazon's Dynamo and Cassandra as PA/EL systems (choosing availability over consistency during partitions, and lower latency over consistency during normal operation), while systems like BigTable and HBase are PC/EC (sacrificing availability for consistency during partitions, while maintaining consistency even at the expense of higher latency during normal operation). PNUTS represents a PA/EC system, showing that trade-off decisions can differ between partition and normal states [4]. This insight is particularly relevant for geo-distributed systems where network latency between regions introduces significant performance considerations even without partitions.

Consensus protocols, particularly Paxos and its variants, have been extensively employed to achieve strong consistency in distributed systems. Multi-Paxos optimizes the original protocol by designating a stable leader to reduce the number of message rounds required for consensus. These protocols have been adapted in various ways to address the specific requirements of distributed databases, but all incur unavoidable communication overhead across regions. Previous research has explored hybrid consistency models that provide different consistency guarantees for different operations. Bailis et al. introduced Probabilistically Bounded Staleness (PBS), which provides probabilistic bounds on staleness in eventually consistent systems. Similarly, Yu and Vahdat proposed a continuous consistency model that allows applications to specify consistency requirements along multiple dimensions.

The GCL approach is built on these foundations, addressing the specific challenges of the geo-distributed NoSQL system. Unlike previous approaches, which often require significant amendments in the underlying database architecture, GCL operates as a modular, decoupled layer that is integrated with minimal invasion with the existing system, guaranteeing co-existence within a single operating structure.

Framework	Primary Focus	Partition Behavior	Normal Operation	Representative Systems
CAP	Three-way trade-off	Consistency vs. Availability	Not addressed	Various distributed systems
PACELC	Extended taxonomy	Consistency vs. Availability	Latency vs. Consistency	Dynamo, Cassandra, BigTable
PBS	Probabilistic guarantees	Statistical consistency	Staleness bounds	Cassandra with quorums
GCL	Selective enforcement	CP for critical, AP for non-critical	Strong for critical, Low latency for non-critical	GCL-enhanced Cassandra

Table 2: Consistency Framework Comparison [3,4]

3. Geographic Consensus Layer Architecture

The Geographic Consensus Layer (GCL) introduces a separate Consensus Ensemble (CE)—a dedicated cluster of nodes deployed across all regions that implement the Multi-Paxos protocol. This CE functions as a specialized control plane for operation serialization, operating independently from the data plane of the underlying Cassandra deployment. This architectural approach draws inspiration from Terry et al.'s Pileus system, which demonstrated that carefully separating consistency mechanisms from data storage can yield significant benefits in geo-distributed environments. In their experiments with Azure deployments across four regions (US West, US East, Europe, and Asia), Pileus showed that decoupling consistency control allowed for latency reductions of 42-78% while maintaining application-specific consistency guarantees through SLA-driven consistency selection [5].

3.1 Consensus Ensemble Design

The CE consists of an odd number of nodes (typically three to five) deployed across multiple geographic regions. These nodes collectively implement the Multi-Paxos protocol, with one node designated as the Leader. The Leader is responsible for coordinating the consensus process, while the remaining nodes serve as Acceptors or Followers. The CE maintains a strictly ordered log of committed operations, each assigned a monotonically increasing GCL Log Index (GLI). The Multi-Paxos implementation includes several optimizations to mitigate the impact of cross-region latency: stable leadership that eliminates the Prepare phase, operation batching that amortizes consensus costs, and pipeline optimization that maximizes throughput during high load periods. These optimizations build upon the findings of Kraska et al., whose MDCC protocol demonstrated that with batching of 50-100 operations, consensus throughput could be increased by 12.6x while maintaining latency within 15% of single-operation consensus [6].

3.2 Critical Write Path Integration

The integration of the GCL with Cassandra's write path involves sending only operation metadata through the consensus layer rather than full payloads. When a client application sends a write operation marked as CRITICAL_WRITE, the coordinator forwards metadata to the GCL. The GCL Leader initiates the Accept phase of Multi-Paxos across a majority quorum of CE nodes. Upon commitment, the operation receives a GLI that establishes its position in the global serialization order. This approach is similar to Terry et al.'s consistency-based SLAs, where they demonstrated that by separating control flow from data flow, their system could reduce cross-region bandwidth requirements by 87.5% while still providing strong consistency guarantees when needed [5]. The coordinator then performs the actual write to local Cassandra replicas using LOCAL_QUORUM consistency, embedding the GLI as metadata. After local completion, the client receives acknowledgment without waiting for cross-region propagation.

3.3 Consistency Enforcement Mechanism

The GLI provides the authoritative global ordering for all critical operations. When Cassandra's native mechanisms encounter conflicting versions of critical data, the version with the highest GLI is deterministically selected, overriding Cassandra's default Last-Write-Wins mechanism. This approach aligns with Kraska et al.'s findings that deterministic conflict resolution based on

global sequence numbers can eliminate read-time uncertainty while adding minimal overhead (measured at less than 1µs per comparison) [6]. For read operations requiring linearizability, clients execute a GCL_READ, which queries the GCL for the highest committed GLI for the data, then performs a LOCAL_QUORUM read from Cassandra, verifying the returned data has a GLI at least as high as obtained from the GCL. If not met, the read is retried. This mechanism ensures linearizable operations for critical data while maintaining Cassandra's performance for non-critical operations.

Feature	Implementation	Benefit	Application in GCL
Stable Leadership	Designated leader node	Eliminates the Prepare phase	Reduces consensus to a single round-trip
Operation Batching	Group operations into single consensus instances	Amortizes consensus costs	Increases throughput while maintaining latency bounds
Pipeline Optimization	Process multiple instances in parallel	Maximizes throughput during high load	Maintains performance with cross-region latency
Metadata-Only Consensus	Only operation metadata passes through CE	Reduces bandwidth requirements	Minimizes cross-region data transfer
Monotonic GLI Assignment	Sequential log index	Provides global ordering	Enables deterministic conflict resolution

Table 2: Consensus Ensemble Design Features [5,6]

4. Performance Evaluation

4.1 Experimental Methodology

To evaluate the effectiveness of the GCL approach, a series of experiments was conducted using the YCSB (Yahoo! Cloud Serving Benchmark) across a multi-region Cassandra topology. The experimental environment consisted of three geographic regions with approximately 100ms inter-region network latency, representative of typical cloud provider latencies between major global regions. This methodology builds upon Cooper et al. 's YCSB framework, which defines five core workloads (A-E) that model different read-write ratios and access patterns to systematically evaluate database performance across varying conditions. Their benchmarking suite, designed specifically for "serving systems" that operate as online data stores, provides precise instrumentation for measuring both throughput and latency, including specific metrics for 95th, 99th, and 99.9th percentile latency outliers that are critical for evaluating distributed system performance [7]. The experimental setup included a 9-node Cassandra cluster (3 nodes per region), a 3-node GCL Consensus Ensemble (1 node per region), and YCSB clients deployed in each region. Three distinct workloads were compared: Baseline (Eventual) with LOCAL_QUORUM consistency, Strong (Traditional) with EACH_QUORUM consistency, and Hybrid (GCL) with 90% non-critical and 10% critical updates. Each workload used YCSB's workload A (50/50 read/write) with the distribution modified from uniform to Zipfian (theta=0.99) to create realistic access patterns with hotspots.

4.2 Results and Analysis

The experimental results demonstrated the effectiveness of the hybrid consistency approach implemented by the GCL. In terms of throughput, the Baseline workload achieved 127,834 operations per second across the cluster. The Strong workload managed only 15,340 operations per second (12% of Baseline), while the Hybrid workload reached 111,215 operations per second (87% of Baseline), significantly outperforming the Strong consistency approach. These results align with Bailis et al.'s PBS (Probabilistically Bounded Staleness) work, which demonstrated through both analytical models and empirical measurements that selective consistency enforcement can provide substantial performance benefits. In their evaluation of PBS across three Amazon EC2 regions, they observed that 96.7% of reads returned "consistent" results within one version, even under eventual consistency when using a quorum-based approach, suggesting that the performance penalty of strong consistency is often unnecessary for many operations [8]. The latency measurements for the GCL showed a bimodal distribution: non-critical operations maintained a median latency of 4.5ms and a 99th percentile of 29.6ms, while critical operations showed a median of 153.8ms and a 99th percentile of 392.7ms. Under partition scenarios, the GCL maintained linearizability for critical operations while preserving availability for non-critical operations, similar to the consistency-availability spectrum described by Bailis et al. Resource

utilization remained modest, with CE nodes' CPU utilization averaging 28.5%, peaking at 39.3% under load. The experimental results validate the core premise of the GCL: effective consistency decoupling that restricts unavoidable Paxos latency to the critical path, providing a practical solution for multi-region NoSQL deployments.

5. Implementation Considerations

Practical implementation of GCL in the production environment requires considering several operating aspects that affect reliability, performance, and stability. These ideas draw on the best practices installed in a distributed system, addressing the unique challenges of operating a hybrid consistency layer in geographically scattered areas. Hunt et al.'s analysis of ZooKeeper, a coordination service for distributed systems, provides valuable insights applicable to the GCL implementation. Their evaluation demonstrated that ZooKeeper can process over 92,000 requests per second with sub-millisecond latency in a localized deployment, while maintaining reasonable performance even with clients spread across multiple data centers. They observed that in wide-area deployments spanning three regions, read operations maintained high throughput (21,000 requests per second) while write throughput decreased to approximately 3,000 requests per second due to synchronous cross-region communication, patterns highly relevant to GCL's consensus layer design [9].

5.1 Failure Handling and Recovery

The GCL implementation includes robust failure detection and recovery mechanisms designed to maintain consistency guarantees while minimizing operational disruption. The Multi-Paxos protocol inherently tolerates failures of minority subsets of CE nodes, allowing the system to continue functioning with up to ⌊(n-1)/2⌋ failures in an n-node ensemble. If a follower node fails, consensus can continue uninterrupted with minimal performance impact. If the leader fails, a leader election process is triggered, resulting in a new stable leader. Hunt et al. observed that in ZooKeeper, leader election is typically completed in 2-4 seconds in a wide-area deployment, with throughput reduced but not eliminated during transition periods [9]. For Cassandra node failures and network partitions, the GCL follows the CP side of the CAP theorem for critical operations, maintaining consistency at the potential expense of availability within disconnected regions, while non-critical operations maintain availability with potential inconsistency.

5.2 Operational Monitoring

Effective monitoring of the GCL deployment is essential for maintaining system health and performance. Key metrics to monitor include consensus latency and throughput, leader election frequency, GLI assignment rate, cross-region network health, and consistency violation incidents. Bronson et al.'s experience with TAO, Facebook's distributed data store, revealed that monitoring 15 key metrics across protocol health, network conditions, and resource utilization detected 89% of potential consistency issues before user impact. Their analysis showed that leader failures were preceded by statistically significant increases in request latency variance (37-58%) approximately 4-7 minutes before failure, providing an early warning mechanism [10].

5.3 Deployment Strategies

The GCL can be deployed through several strategies depending on organizational requirements. The CE nodes can be deployed as sidecar processes alongside Cassandra nodes, as dedicated infrastructure separate from the Cassandra cluster, or leveraging cloud provider features for orchestration. Bronson et al. noted that for TAO, Facebook's social graph data store serving over 1 billion reads and 2.5 million writes per second, separating the consistency control plane from data storage nodes reduced failure correlation by 74% and improved maintenance operations by allowing independent scaling and updates [10]. Hunt et al. demonstrated that careful placement of coordination servers in a geographically distributed deployment could reduce average latency by 20-30% when placing nodes in regions with the lowest average network latency to other regions rather than simply distributing nodes evenly [9].

Failure Type	Detection Method	Recovery Mechanism	Impact on Operations
CE Follower Failure	Heartbeat timeout	Continue with the remaining nodes	Minimal impact on consensus
CE Leader Failure	Heartbeat timeout	Leader election process	Temporary throughput reduction
Cassandra Node Failure	Cassandra gossip protocol	Standard Cassandra recovery	GLI ensures consistency during recovery

Network Partition	Connection timeout	CP for critical, AP for non-critical	Mixed availability/consistency based on operation type
Region Isolation	Multi-point monitoring	Continue in the majority of regions	Critical operations are unavailable in minority regions

Table 4: Failure Handling Mechanisms [9,10]

Conclusion

The geographical consensus layer provides a practical solution for the constitutional dilemma faced by organizations deploying NOSQL databases in many geographical regions. By decoupling the consensus mechanism from the data plane and applying it to selectively important operations, the GCL creates an effective balance between strong stability guarantees and high performance. Experimental assessment confirms that this hybrid approach maintains the throughput within the appropriate range of the final stability baselining, ensuring linear operations to significant data. Implementation ideas around failure handling, monitoring, and signs strategies further enhance the practical appropriateness of GCL in the production environment. The architecture presented suggests that the system can be wisely designed to navigate the deficiency of the CAP theorem, which provides suitable stability levels based on application requirements rather than forcing a one-size-fit-all approach. The GCL organizations open the possibilities to confidently deploy the applications distributed globally with the needs of different stability levels without compromising on purity or performance.

References

[1] Eric Brewer, "Towards robust distributed systems," ResearchGate, 2000. https://www.researchgate.net/publication/221343719_Towards_robust_distributed_systems

[2] Medium, "Benchmarking Cassandra Scalability on AWS — Over a million writes per second," Netflix Technology Blog, 2011. https://netflixtechblog.com/benchmarking-cassandra-scalability-on-aws-over-a-million-writes-per-second-39f45f066c9e

[3] Seth Gilbert and Nancy Lynch, "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services," ACM Digital Library, 2000.

Corbett, "Spanner: Google's Globally Distributed Database," ACM Digital Library, https://dl.acmhttps://users.ece.cmu.edu/~adrian/731-sp04/readings/GL-cap.pdf

[4] Daniel J. Abadi, "Consistency Tradeoffs in Modern Distributed Database System Design," IEEE, 2012. https://www.cs.umd.edu/~abadi/papers/abadi-pacelc.pdf

[5] Douglas B. Terry, et al., "Consistency-Based Service Level Agreements for Cloud Storage," Yale University, https://www.cs.yale.edu/homes/mahesh/papers/pileus-sosp2013.pdf

[6] James C..org/doi/10.1145/2491245

[7] Brian F. Cooper, et al., "Benchmarking Cloud Serving Systems with YCSB," ACM Digital Library. https://dl.acm.org/doi/10.1145/1807128.1807152

[8] Peter Bailis, et al., "Probabilistically Bounded Staleness for Practical Partial Quorums," arxiv, Apr. 2012. https://arxiv.org/abs/1204.6082

[9] Patrick Hunt and Mahadev Konar, et al., "ZooKeeper: Wait-free coordination for Internet-scale systems," USENIX https://www.usenix.org/legacy/event/atc10/tech/full_papers/Hunt.pdf

[10] Nathan Bronson et al., "TAO: Facebook's Distributed Data Store for the Social Graph," ACM Digital Library, https://dl.acm.org/doi/10.5555/2535461.2535468