HIGH AVAILABILITY • LOW LATENCY

Scalable Database Design for 5G and Beyond

APRIL 2026

ENEA

Executive Summary

The telecommunications industry is navigating an inflection point. As networks evolve from 4G to 5G Standalone and as early 6G research begins to shape the decade ahead, the infrastructure layer that manages subscriber state, session data, and signaling has emerged as a critical determinant of service quality and business continuity. Operators who once operated within looser latency constraints now demand millisecond-level responsiveness, and those who once planned for occasional maintenance windows must now architect for continuous availability across multiple geographically distributed sites.

The consequences of getting this wrong are quantifiable. Unplanned network outages expose operators to direct revenue loss, regulatory fines, and service-level agreement penalties; costs that escalate rapidly for large-scale operators serving tens of millions of subscribers. Beyond just revenue impact of an outage, control-plane latency spikes cascade into failed registrations, degraded session quality, and reduced overall network capacity. In 5G Standalone deployments, where a single subscriber registration can require dozens of real-time interactions with the subscriber database, even modest increases in per-request latency translate into measurable degradation at scale.

Enea's Stratum is a purpose-built, cloud-native distributed database engineered specifically for these conditions. Deployed by Tier-1 operators across North America and Europe, storing hundreds of millions of subscriber records and servicing millions of transactions per second, Stratum has been designed from the ground up to satisfy three simultaneous requirements: high availability, low latency, and operational sustainability across geographically distributed, multi-site deployments.

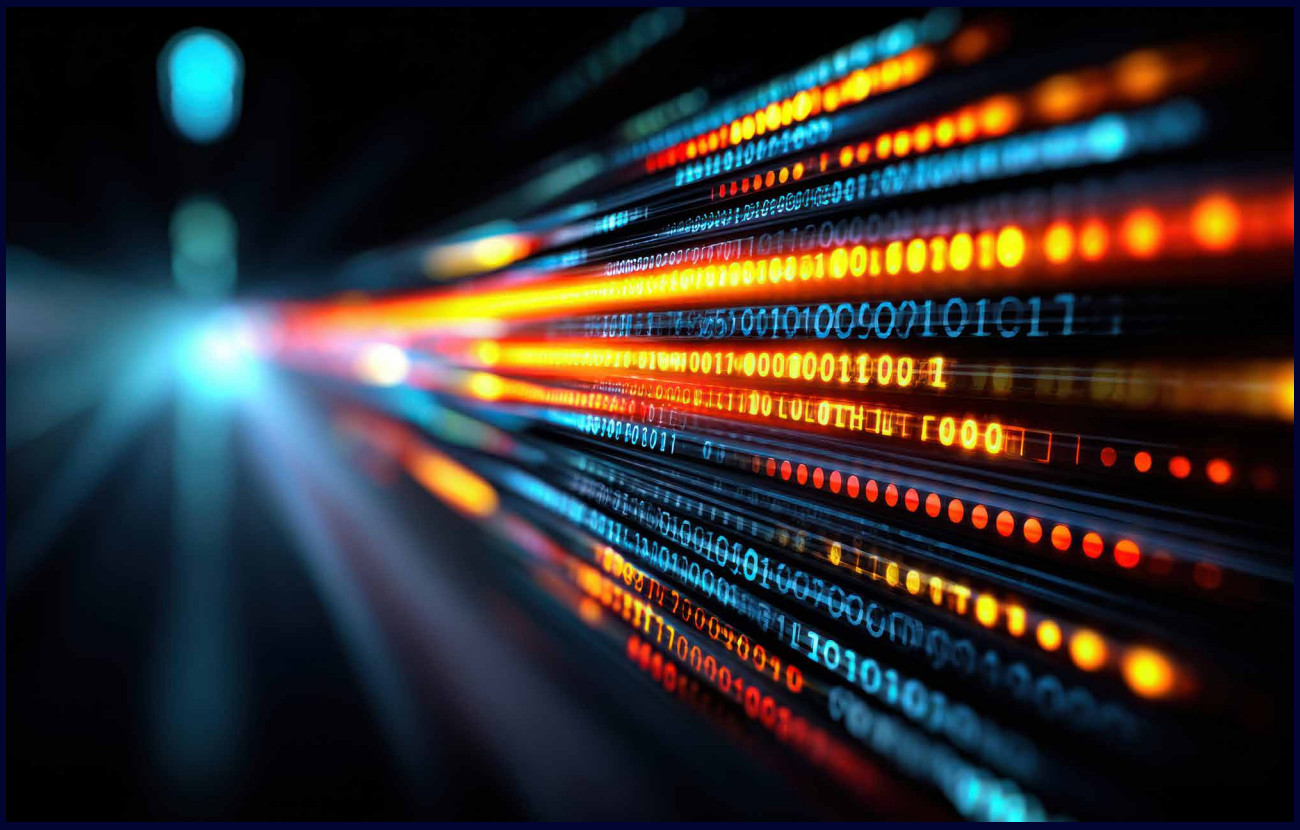
This paper explains the architectural trade-offs that define the distributed database problem space, articulates why traditional database designs fall short of modern operator requirements, and describes the mechanisms through which Stratum resolves these constraints, all without sacrificing the predictability and operational transparency that telecom deployments demand.

Authors

Andrej Gregić
Senior Software Architect
Enea

Erik Sigurdsson Rosenborg
Director Product Management
Enea

Contents



CHAPTER 1

Achieving Scale and High Availability in Telecom Databases

The Evolving Scale of the Operator Challenge

The past two decades have transformed what it means to operate a mobile network. The shift from 2G voice systems to data intensive 4G LTE, the verbose signaling architecture of 5G (New Radio and 5G Standalone), and the approaching horizon of 6G have each imposed new and compounding demands on the infrastructure that manages subscriber state. As of late 2025, there were approximately 2.9 billion[1] 5G connections worldwide, and cellular IoT subscriptions continue to expand at scale. Global mobile data traffic is projected to reach over 140[1] exabytes per month at the end of 2025, with a growing proportion carried by 5G networks operating at a level of signaling complexity that has no precedent in prior generations.

This shift is not merely one of volume. 5G Standalone (5G SA) architectures fundamentally decompose the monolithic control plane functions of earlier generations into disaggregated, cloud-native network functions that communicate over shared data infrastructure. In this model, the subscriber database is no longer a peripheral component behind a single service, it has become a distributed state substrate that all control plane functions depend on simultaneously. The result is a qualitative change in what the database must deliver; not just capacity but spatial distribution, transactional throughput, and predictable behavior under both steady-state load and failure conditions.

Looking ahead, early 6G research envisions connectivity speeds and service densities an order of magnitude beyond current 5G capabilities. With 6G deployment timelines pointing toward the 2030s, operators planning infrastructure investments today must anticipate not only the demands of current 5G deployments, but the architectural headroom required to evolve toward these next-generation service models. In this context the database architecture is not a feature decision, it is a foundational infrastructure commitment.

Why Availability is Non-Negotiable

The consequences of database unavailability in a telecom context are direct, severe, and immediate. A subscriber database outage does not merely degrade service, it halts network registrations, disrupts active call sessions, and prevents billing records from being written. For a large mobile operator serving tens of millions of subscribers, even minutes of downtime translates into significant revenue loss and legal implications, and the broader impact extends well beyond the immediate outage window.

Industry data illustrates the financial stakes. A 2024 Oxford Economics study[2] found that unplanned downtime costs Global 2000 enterprises an average of $200 million annually. Telecom operators, as critical infrastructure providers, face an additional layer of exposure: regulatory fines for service degradation, service-level agreement penalties with enterprise customers, and the reputational damage that follows any publicly visible network disruption. In regulated markets, the compliance consequences of a subscriber database outage can persist long after service has been restored.

The industry's historical architectural response to this challenge of deploying single-writer primary databases with hot standby replicas has reached its architectural limits. Traditional designs serialize all write operations through a single node, introduce geographic single points of failure, and depend on leader election and failover procedures that take seconds to minutes to complete. This is not a sustainable operational model for a 5G Core where control-plane functions must maintain continuous operation across multiple regions and availability zones.

Why Latency is a Capacity and Quality Driver

Database latency is not an abstract performance metric in the telecom control and service paths; it is a direct constraint on subscriber experience and network capacity. The 5G Core is architected around decomposed network functions that interact with shared data stores for every signaling procedure. A typical subscriber network registration can involve dozens of read and write operations to the subscriber database (the 5GC UDR in 3GPP terms). For operators managing geographically distributed infrastructure across wide geographical areas, the physics of wide-area networking impose hard limits. Even a small increase in per-request latency accumulates into visible degradations of subscriber experience, turning what would normally be a sub-second registration into a multi-second one.[3]

Three Simultaneous Requirements, One Architecture

Taken together, the operational realities of modern telecom deployments define a rigorous and demanding problem space. The database architecture that supports a large-scale mobile operator must simultaneously satisfy three dimensions of scale:

- Spatial scale: active operation across multiple sites and geo-regions, separated by wide-area network links with variable latency, jitter, and the possibility of complete network partitions.
- Traffic scale: sustained high-throughput transaction processing with predictably low response times, including sudden and unpredictable traffic spikes driven by subscriber activity or network events.
- Operational scale: continuous availability through software and hardware upgrades, planned maintenance, site additions, and partial failures. These need to be treated as routine operational conditions rather than exceptional events.

No general-purpose database engine was designed to optimize this specific combination of constraints. The telecom subscriber data layer has requirements that differ meaningfully in character from the enterprise transactional workloads, web-scale content platforms, and financial services databases. The telecom data layer demands deterministic behavior under failure, fine-grained control over consistency trade-offs, and the operational transparency to allow engineering teams to reason about the system's state without resorting to manual reconciliation.

This is the problem space that Enea's Stratum was engineered to address. Developed in close collaboration with Tier-1 operators in North America and Europe and continuously evolving through production deployments at scale, the Stratum embodies a set of deliberate architectural choices rooted in the formal trade-off frameworks that govern all distributed systems. The following sections introduce these frameworks and explain how they directly inform Stratum's design.

CHAPTER 2

The Database Trilemma

Distributed systems of any kind are inherently forced to make design decisions that involve trade-offs in maintaining a global unified state across all instances. Process failures, hardware issues, and the realities of operating over imperfect networks, where delays, outages, and complete partitions (communication between instances is severed) make such trade-offs unavoidable.

The CAP theorem, as one of the most well-known distributed system theorems, formalizes this fundamental limitation into a simple trilemma: in the presence of a network Partition (P), a replicated data store cannot simultaneously guarantee both strong data Consistency (C) and Availability (A).[4]

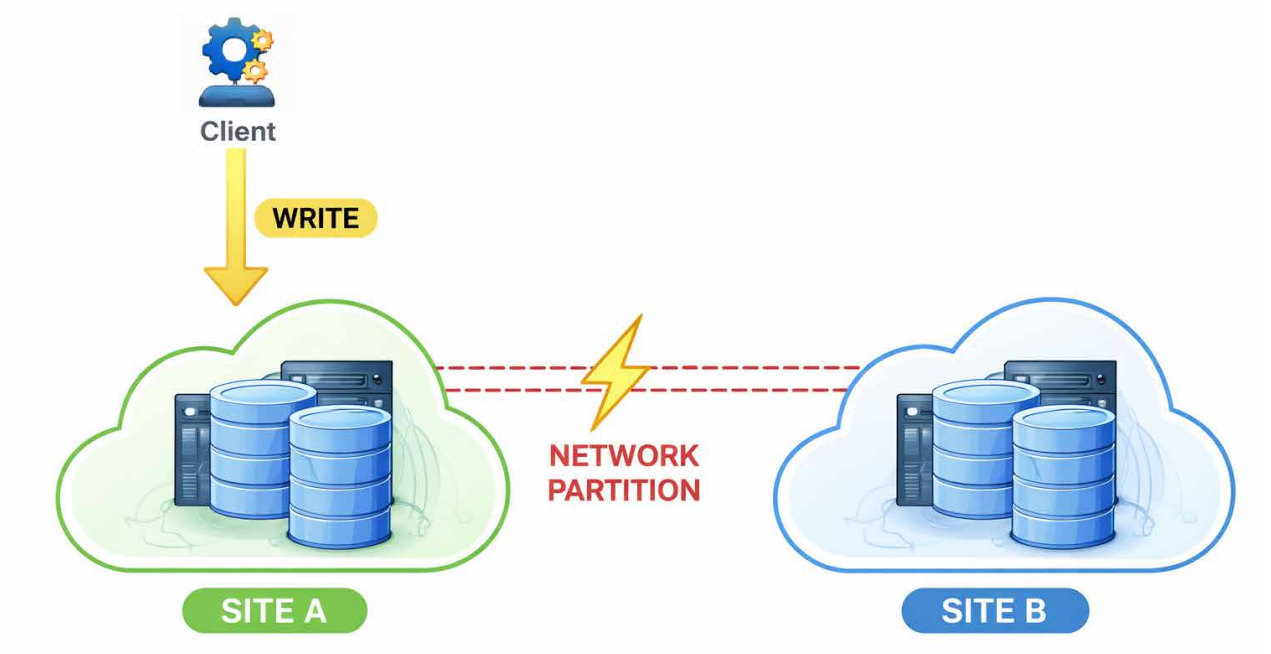


Figure 1: Visualization of the CAP trilemma in a network partition.

The replicated data store must make a deliberate design choice: either preserve data consistency by rejecting the client write request or preserve availability by continuing to accept the write request even while replicas temporarily diverge. Because tolerance to network partitions is non-negotiable for geographically distributed deployments (sooner or later they will occur), real-world systems are ultimately defined around how they balance consistency and availability when communication between sites is disrupted.

Network partitions, however, are only one class of failure relevant to telecom database design. Even during normal operation, wide-area networks introduce latency, jitter, and packet loss. Any requirement for synchronous cross-site coordination increases client response times and exposes users to network variability. As a result, system designers must consider system design trade-offs not only for network partition failures but during steady-state operation as well.

The PACELC design principle extends the CAP theorem by describing these broader design trade-offs:

- If a Partition (P) occurs, a system must choose between Availability (A) or Consistency (C).
- Else (E) - under normal operation - a system must choose between Latency (L) and Consistency (C).

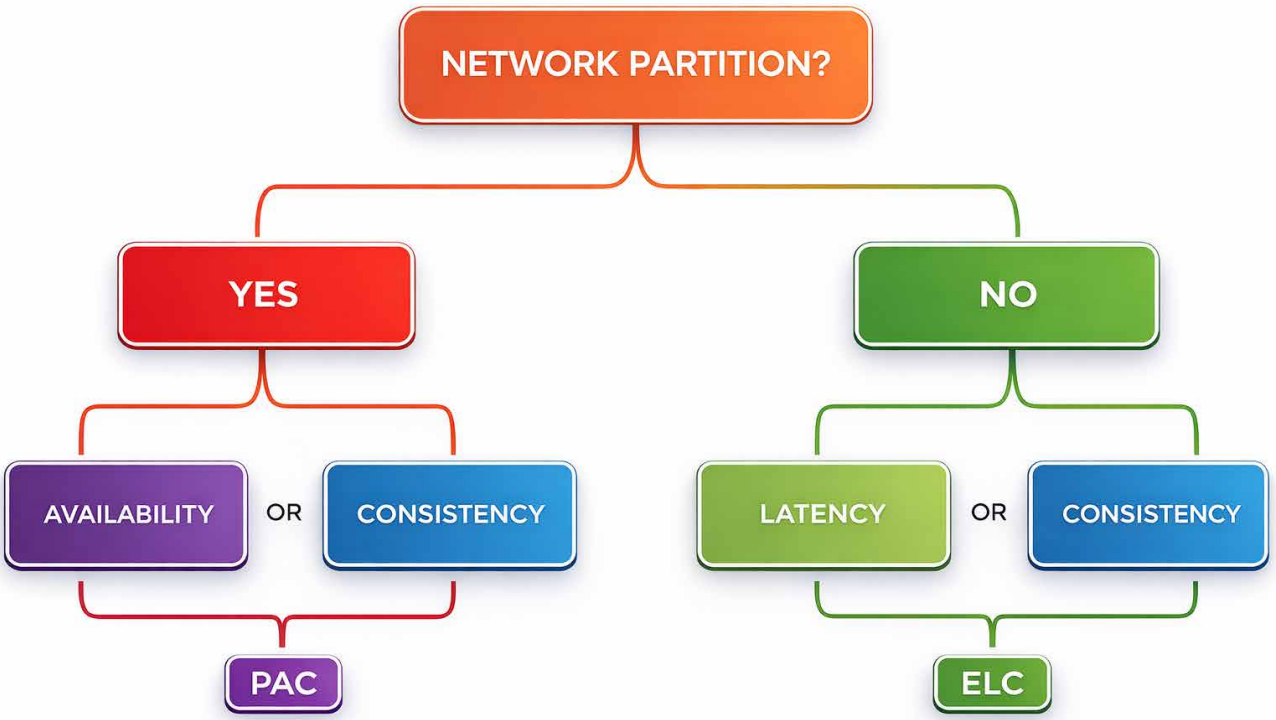
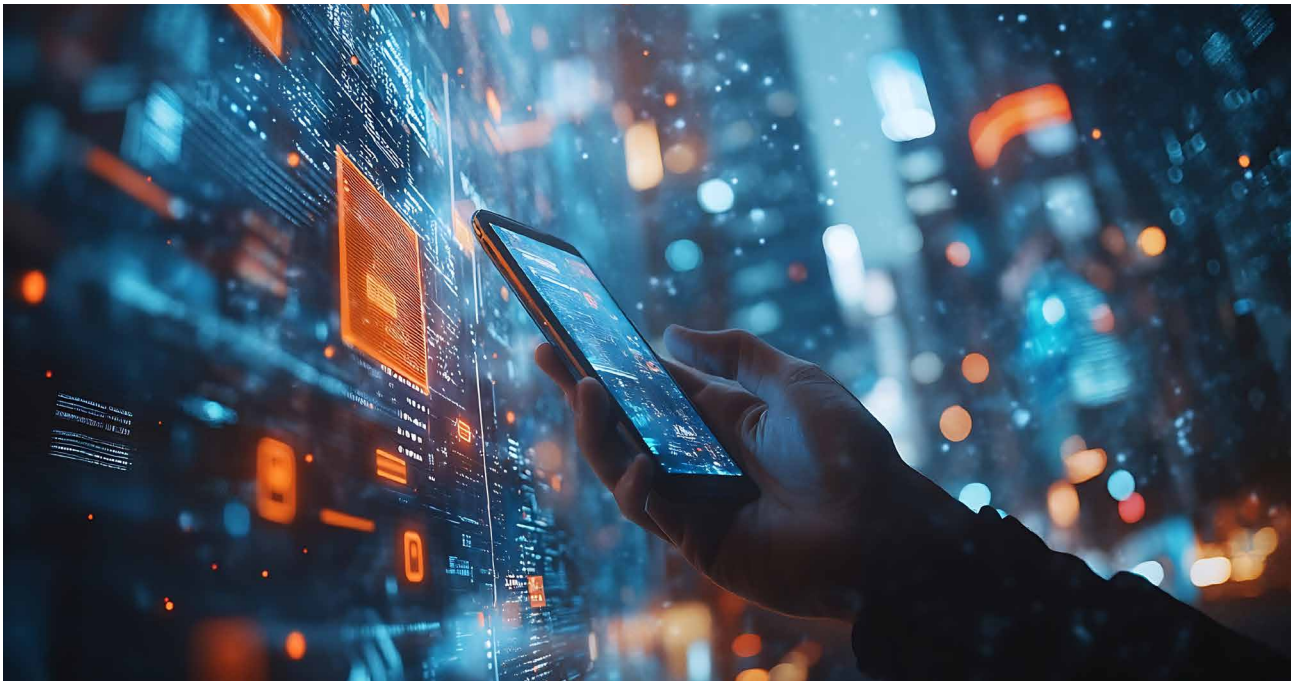


Figure 2: PACELC theorem decision diagram.

In other words: even in the absence of a network partition, maintaining strict data consistency across geographically distributed sites requires coordination over wide-area networks, and that coordination introduces additional client-visible latency.[5]

From a PACELC perspective, systems that require consistently low response times in a dynamic, partitionable network are forced to relax their expectations of strict, global data consistency not only during failures, but during normal operation as well.



CHAPTER 3

A Database Optimized for Telcos

Stratum's requirements as a distributed data store for telecoms are clear: high availability must be guaranteed and is a non-negotiable requirement, while low client response latencies should be prioritized. Within the PACELC framework, this design represents a deliberate choice for Stratum to prioritize Partition tolerance and Availability (PA) during failures and Latency over strict Consistency (EL) during normal operation.

Attempting to enforce strict real-time consistency across all replicas introduces additional coordination overhead on every transaction, increasing response latency and cross-replica communication. Sustaining such a model demands greater compute, memory, and network capacity to absorb these costs. However, even with substantial infrastructure investment, the latency cost cannot be eliminated; it is intrinsic to the consistency model itself. In practice, for most types of data, the small possibility of a stale database read is less costly than the performance penalties and architectural rigidity imposed by strict consistency.

Grounded in this reasoning, Stratum adopts a model of practically bounded eventual consistency: write requests are accepted locally to preserve responsiveness and availability, while background replication mechanisms ensure that data automatically and reliably converges across datacenters. The system is engineered so that convergence is guaranteed and operationally transparent even following failures, eliminating the need for manual intervention.

Over the past decade, many globally distributed platforms have intentionally adopted similar designs to achieve scale, high availability, and low latency. Platforms such as Netflix and Twitter (X) operate large-scale distributed systems built on eventually consistent storage models to support massive write throughput and continuous operation across geographic regions.[6,7]

Practical studies and production usages have shown that when designed correctly, eventually consistent systems behave in a way that is as predictable and reliable from the client's perspective as a strictly consistent system. At the same time, they deliver the key benefits required by large-scale services: uninterrupted availability, low-latency local writes, and efficient horizontal scaling across geographically distributed datacenters.[8] That is the lens through which Stratum's architecture was founded and has continuously evolved in close cooperation with world leading telecom operators, scaling to store hundreds of millions of records while servicing millions of transactions per second.

Instead of adapting general-purpose databases, Enea's Stratum implements a purpose-built document database engine designed from the ground up as a telecom network data layer, with an architecture design that delivers three outcomes:

- **High Availability through Write-anywhere:** clients can write to any site, avoiding wide-area-network latency in the write path and eliminating complex leader/ownership failover behavior, with deterministic conflict resolution ensuring safe convergence.
- **Horizontal scalability through push-based data replication:** replication has a fixed cost and replication lag is effectively equal to the inter-site network latency.
- **Operational sustainability through continuous data anti-entropy:** a background process reconciles missed updates, network partitions, process restarts, and new-site bootstrap - ensuring data convergence without operator intervention.

This article explains why the Stratum design goals align with the real trade-offs for operators planning large-scale telecom deployments.

CHAPTER 4

Write-Anywhere Availability and Conflict Resolution

Traditional database architectures have historically relied on a single-writer or single-leader model for handling write requests. In the single-writer approach, one primary site is responsible for processing all write operations, often supported by a hot standby replica that can take over during failures. Read scalability is then typically achieved by deploying read-only replicas or caches that serve queries without modifying data.[9] It should be noted that such a system is already subjected to eventual consistency, as data caches cannot guarantee to have the latest data state at all times.

This has a fundamental limitation: write throughput is constrained by the capacity of a single node or location. As workloads that necessitate high write throughputs grow in demand (especially applicable to the needs of the 5G SA), scaling becomes increasingly difficult or impossible. In addition, write traffic is limited to the location of the writable instance, making it impossible to fully utilize distributed infrastructure or maintain low-latency writes across geographical regions.

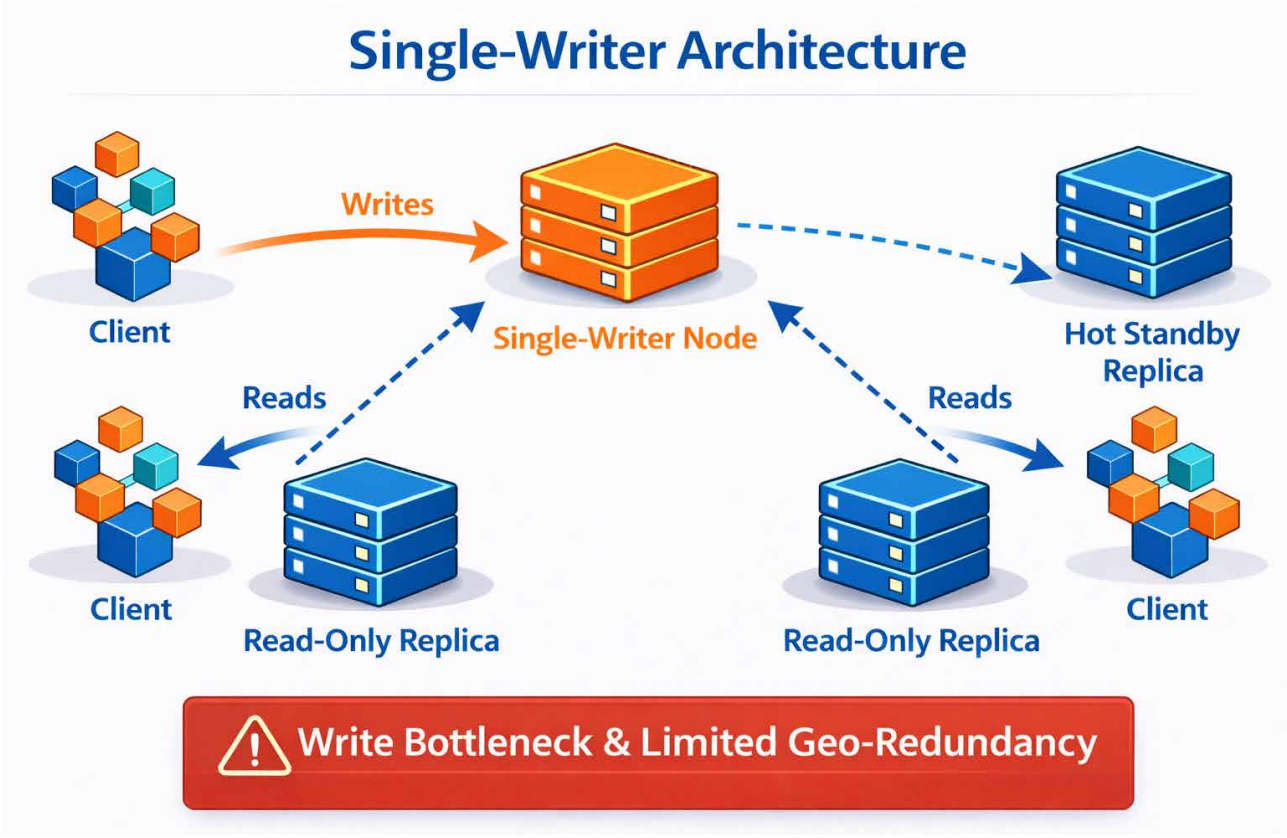


Figure 3: Visualization of a single-write architecture.

A variation of this traditional design is the so-called single-leader approach, in which multiple sites may accept write requests, but each database record is assigned to a single specific leader site. If a client attempts to update a record on a site that does not hold the leadership for that record, the request is automatically (re)routed to the assigned leader.

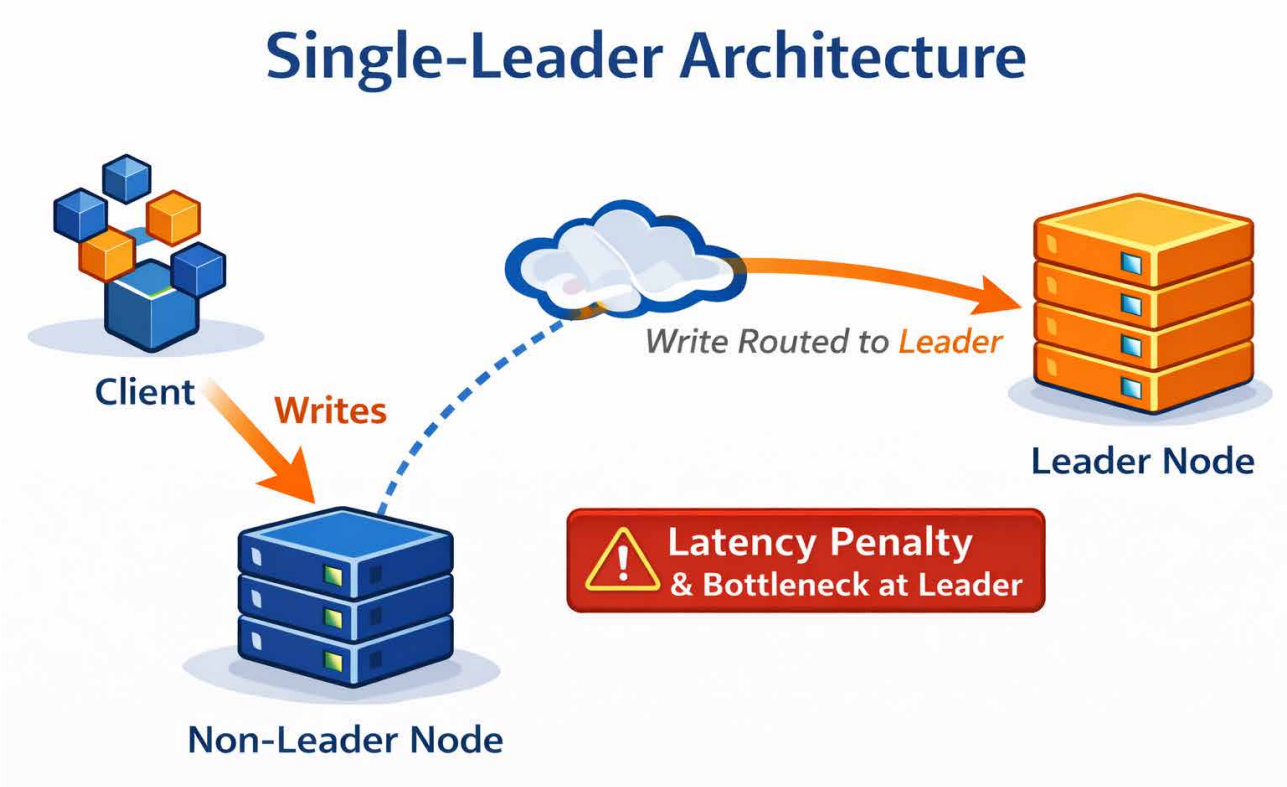


Figure 4: Visualization of a write request re-route in a single-leader architecture.

This approach may appear attractive at first because it avoids concurrent write conflicts and prevents stale reads, but the inherent fragility and latency costs of traversing wide-area networks remain unavoidable, leading to:

- **High client response latencies:** re-routed client requests incur wide-area network latency and variance.
- **Network fragility becoming user-visible:** packet loss and jitter translate into timeouts and retries.
- **Complex failover decisions:** leader election/ ownership transfer must balance rapid recovery against split-brain risks (multiple replicas acting as leaders that serve the same data).
- **Scalability ceilings:** leaders become coordination chokepoints; re-routed traffic becomes a disproportionate contributor to tail latency.

The PACELC theorem predicts this systematically: maintaining strong consistency without partitions still necessitates latency costs due to coordination.

Enea's Stratum design allows clients to perform writes on any site ("multi-writer", "multi-active"); a deliberate first principle aimed at achieving high availability and low client response latency.

This is consistent with the design intent of so-called "optimistic" replication families: replicas remain available and responsive, and updates propagate and converge asynchronously as connectivity permits.[10]

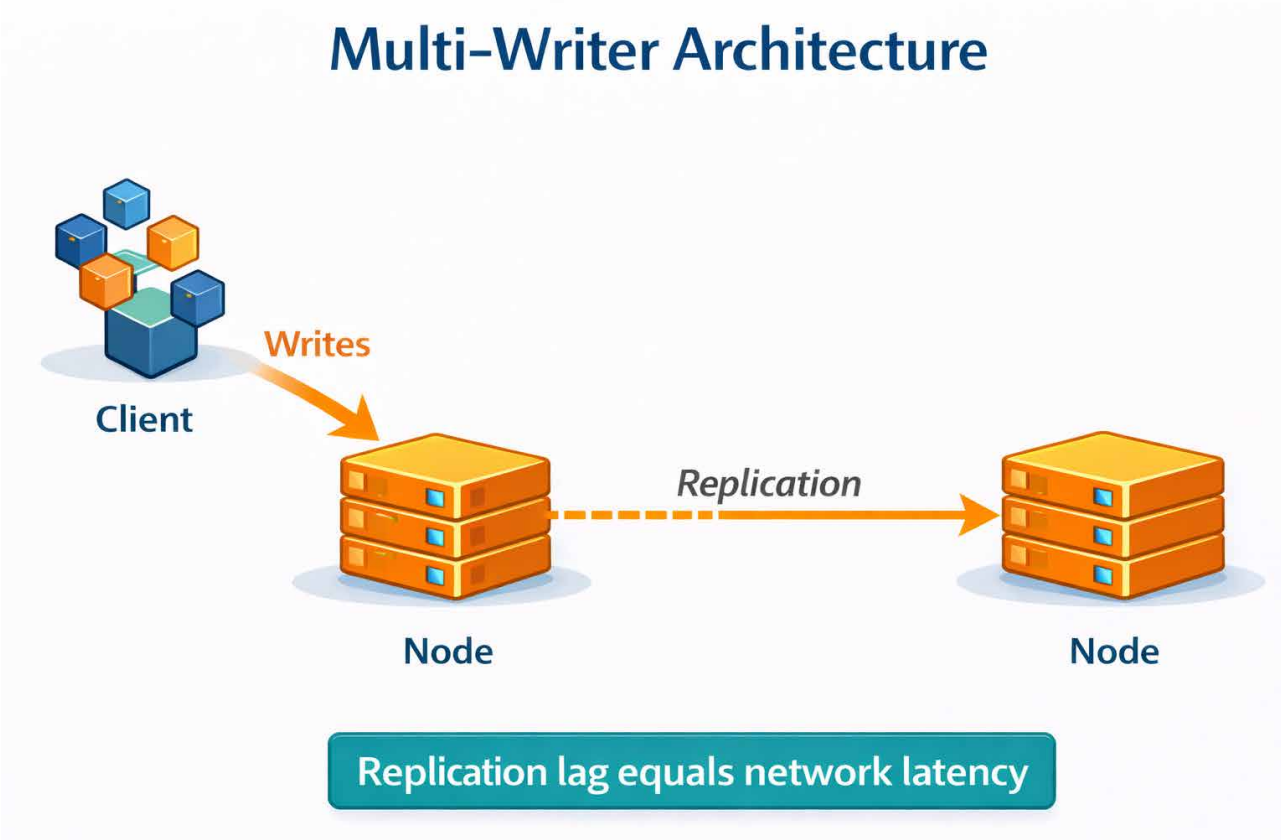


Figure 5: Visualization of a write to local site in a multi-writer architecture.

It is intuitively clear that if clients are allowed to write to any site at any time, concurrent updates to the same record across sites can lead to conflicts, especially in geographically distributed systems, where network delays or partitions are common. Naively allowing multiple clients to write simultaneously risks producing incorrect data states unless conflicts are carefully managed.

One fundamental approach to avoiding such conflicts is thoughtful data modeling: independent clients operate on separate, non-overlapping update domains, such as different records or distinct fields within a record. Another strategy is session affinity, where clients "stick" to a particular site for the duration of a request flow, ensuring that writes remain localized. Beyond these measures, deterministic merge semantics are applied by the database so that, when concurrent updates do occur, replicas reconcile them in a predictable and safe manner, guaranteeing that all sites converge to a consistent state without manual intervention.

Conflict-free Replicated Data Types (CRDTs) formalize this merge principle. CRDTs are data structures that allow replicas to accept updates independently while guaranteeing convergence of the data once all updates have been received. By design, they provably ensure that the data will converge safely and correctly, even in the presence of concurrent writes or prolonged network partitions.[11] Many modern database have opted to implement some form of CRDTs to achieve strong eventual consistency without requiring centralized coordination across datacenters.

Stratum's document storage engine implements CRDT-like deterministic merging, allowing local, low-latency writes on any site while guaranteeing that concurrent modifications converge safely and predictably. For example, if two clients update the same document concurrently on different datacenters, merge semantics ensure that both updates are preserved and automatically reconciled, resulting in a single, well-defined document state.

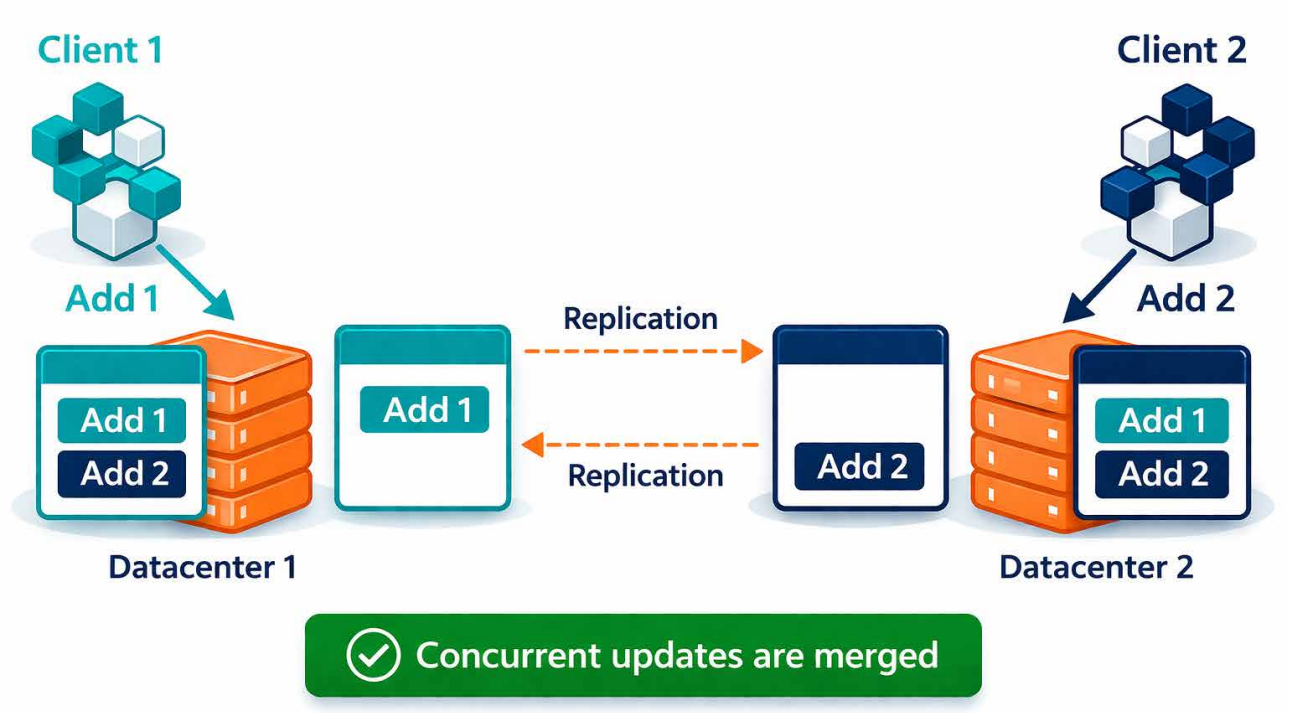


Figure 6: Merge of two concurrent Add operations for the same record on different datacenters.

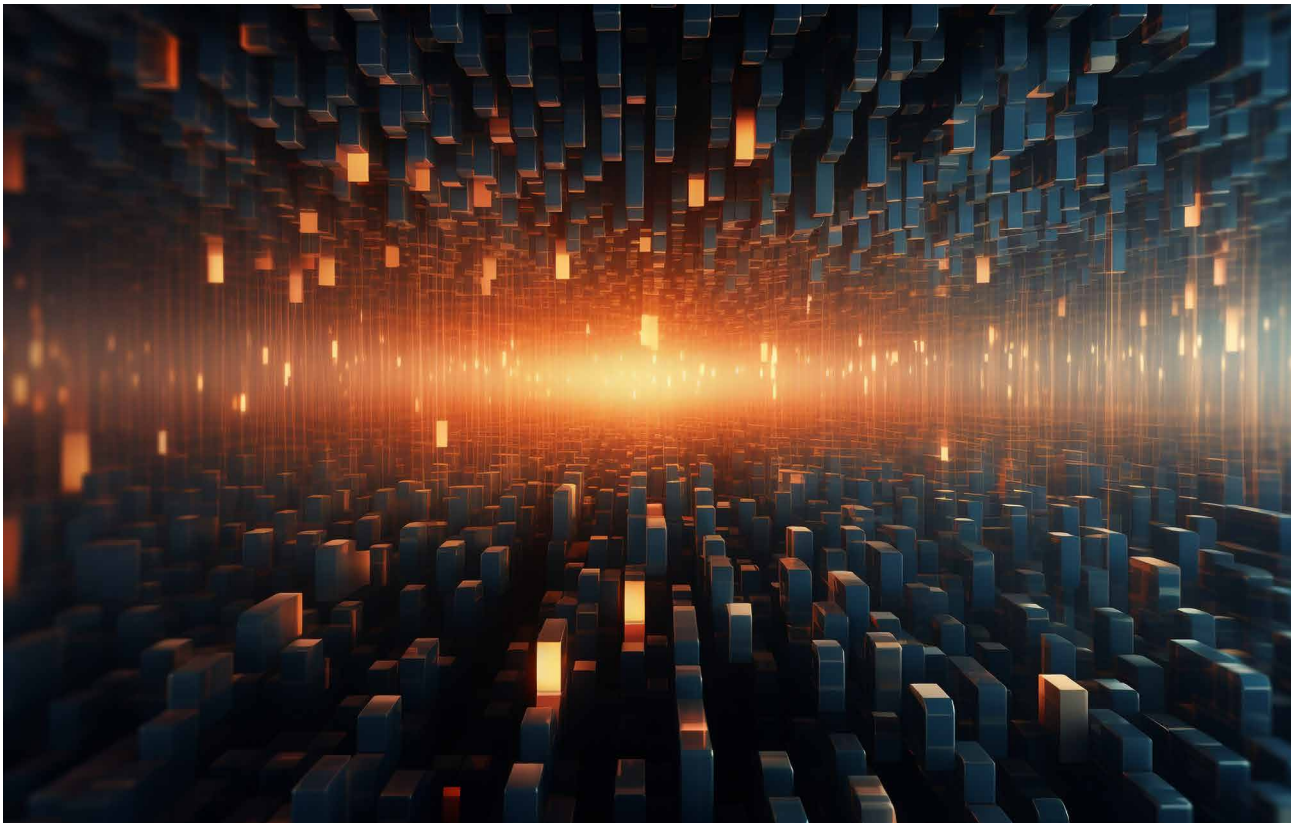
CHAPTER 5

Push Replication Enables Horizontal Scale

IIn many distributed deployments, data replication introduces an operational tax, including backlogs of write changes, periodic catch-up events, and persistent uncertainty around cross-site synchronization. Expanding to additional datacenters can increase this overhead, and in poorly designed systems, the cost of replication can grow unpredictably, creating bottlenecks and limiting scalability.

Stratum addresses these challenges with scalable push-based replication, in which updates are immediately sent (“pushed”) from each replica to its peers as soon as they are committed locally. Replication is “fire-and-forget”: messages are sent without maintaining a replay history, and updates are applied independently on each replica. This design keeps the common successful replication path simple and efficient. As the replication processing time on the receiving side is negligible in comparison to the network latency, the overall replication lag is effectively equal to that one-way network delay. When replication messages are lost due to a network or process failure, inconsistencies are resolved by the continuous anti-entropy process (covered in the next chapter) rather than by explicit replication mechanisms.

The predictable behavior of this replication approach ensures that horizontal scaling comes with a measurable and bounded replication cost. Operators can expand deployments with new sites and accurately estimate the capacity gains, while understanding how returns diminish as the system scales, all without encountering unexpected replication lag, processing stalls, or nonlinear scaling penalties.



CHAPTER 6

Continuous Anti-Entropy Data Repair for Sustainable Operations

Wide-area networks inevitably experience transient failures, packet loss, and latency spikes, all of which can disrupt inter-site communication and lead to missed replication updates. Over time, data divergence is not a possibility; it is a certainty unless there is an explicit reconciliation mechanism.

Anti-entropy (as an allusion to thermodynamics) refers to the process of detecting and repairing divergence between replicas, restoring consistency across distributed datasets. Many criticisms of eventual consistency stem not from the model itself but from operational challenges such as manual repair procedures, prolonged inconsistencies, and limited visibility into reconciliation processes.[12]

Stratum's continuous anti-entropy process is implemented as a built-in background repair loop that:

- Detects and repairs missed replication updates
- Reconciles divergence after prolonged network partitions
- Synchronizes data after site restarts or failures
- Bootstraps new sites by automatically converging them to current state

Anti-entropy is a well-established concept in distributed storage systems, with similar mechanisms appearing in widely deployed databases through repair processes, read-repair techniques, and background synchronization. However, in many deployments, achieving full data convergence still requires scheduled or operator-driven repair actions, increasing operational overheads and recovery times.[13]

Stratum instead embeds anti-entropy as a continuous, automated process while still exposing tuning controls, metrics, and alarms that give operators visibility and control without requiring manual intervention. Production experience has shown that the CPU cost and impact on clients associated with anti-entropy is minimal, even under failure or bootstrap conditions that require a high degree of data repair. This efficiency is achieved by leveraging the system's distributed architecture and by using optimized data structures, such as the elastic binary Merkle tree.[14]

Combined with scalable push-based replication, this design enables rapid replica convergence under normal conditions. The vast majority of replication updates propagate successfully on the first attempt and converge within the inter-site network latency window. For the very small fraction of updates affected by transient network failures, the background anti-entropy process detects and reconciles divergence at configurable intervals, allowing operators to balance system performance considerations against acceptable time windows of data staleness. It should be noted that because each replication update conveys the current state of the record, any updates lost due to transient failures are corrected by the next successful replication update, eliminating the need for explicit repair in chains of successive writes.

As a result, the system delivers behavior that appears strongly consistent from a client perspective during normal operation, while preserving availability during severe failure scenarios such as network partitions.

Probabilistically Bounded Staleness (PBS) is often utilized as a consistency model that quantifies the probability of data staleness between database replicas in an eventually consistent system. PBS provides a systematic explanation for why eventually consistent databases often appear strongly consistent in practice, even without deterministic bounds on staleness.[15]

To illustrate, we used a Monte Carlo simulation that repeatedly generates random executions of the system under a given set of parameters, observing the frequency of stale reads. The simulation accounts for factors such as inter-replica network latency, replication failure probability, and the configured anti-entropy check interval.

For example, consider a system with 3 replicas and a 30 millisecond inter-replica network latency, a variable probability of network failure, and a 300 millisecond anti-entropy check interval.

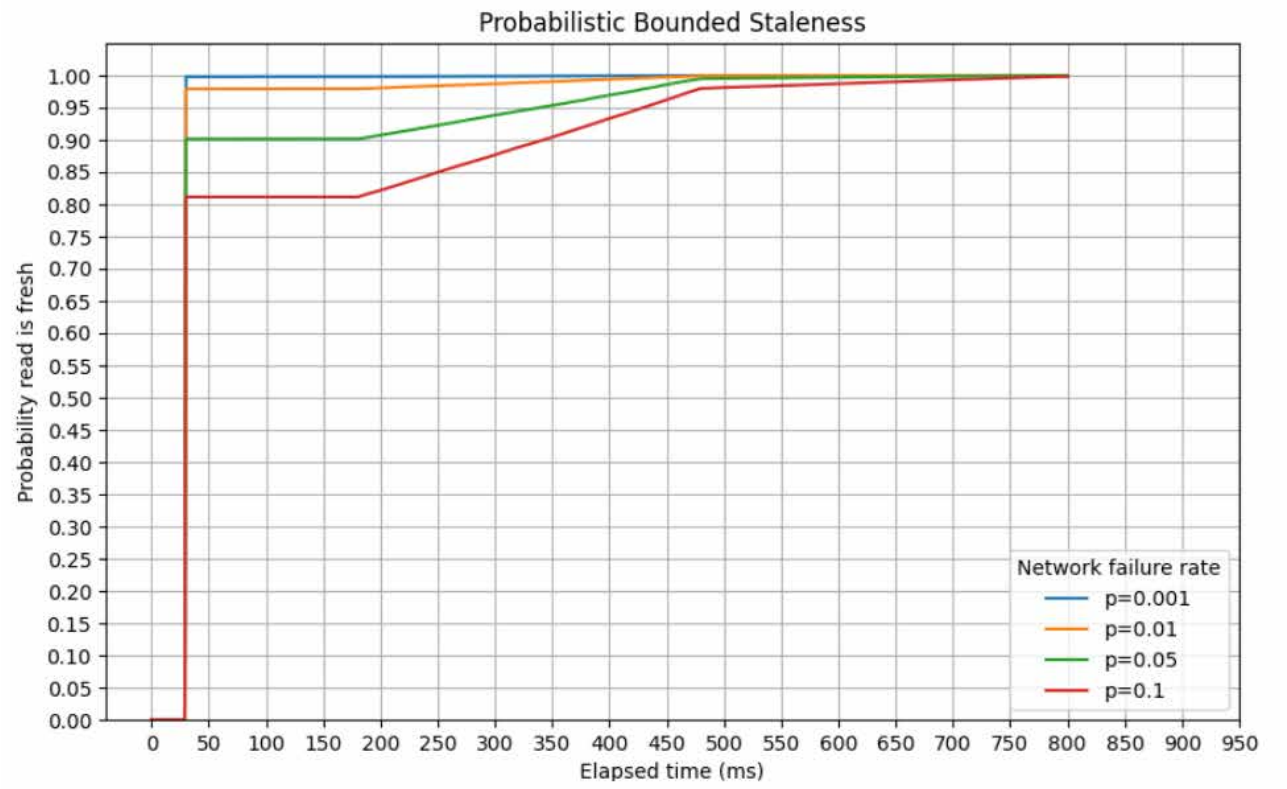


Figure 7: Merge of two concurrent Add operations for the same record on different datacenters.

The resulting PBS curves exhibit a sharp increase at the network latency boundary (30ms), indicating that reads become fresh in the majority of cases as soon as the replication update reaches the remote replica. A small probabilistic tail represents rare situations in which the replication message is lost due to the configured network failure rate and reconciliation must occur through the anti-entropy mechanism.

As the probability of network failures increases, the likelihood that anti-entropy requires multiple reconciliation rounds also increases. Nevertheless, the probability distribution shows that replicas converge rapidly, and data becomes consistent with very high probability within only a few anti-entropy cycles.

This demonstrates that the likelihood of observing stale data beyond the network latency window remains extremely small under normal network conditions, causing the system to appear effectively strongly consistent from the perspective of an external client.

It is important to note that this scenario does not include full network partitions or severe network degradation failures; such failures would fall under the network partition category, where PACELC necessitates a trade-off favoring availability over strict consistency.

CHAPTER 7

Conclusion

Telecom infrastructure cannot afford a database design that becomes unavailable or slow whenever the wide-area network is imperfect. CAP and PACELC, as fundamental distributed system theorems, formalize the unavoidable trade-offs between consistency, availability, and latency in distributed systems under real-world failure scenarios, such as network partitions.

For telecom deployments, the question operators face is clear: does the deployment allow temporary data divergence in a way that is safe, practically bounded, observable, and self-healing, while keeping the most common operations fast and highly available? If the answer is yes, then a highly available system which allows clients to perform writes on any site becomes a competitive operational advantage.

Aligned with research on optimistic replication and the operational principles of highly available distributed stores, the Stratum document database is purpose-built for telecom workloads. It combines high availability, low latency, and operational reliability through deterministic merge rules and continuous anti-entropy, ensuring automatic, fast, and predictable convergence across geographically distributed sites. These architectural mechanisms have proven to be necessities for modern telecom deployments, meeting operators' needs for predictable, efficient behavior across their infrastructure.

See more about **Stratum**.

References

[1] Ericsson. Ericsson Mobility Report November 2025 (2025)

[2] Oxford Economics. “The Hidden Costs of Downtime” (2024)

[3] Gilbert, Seth; Lynch, Nancy. "Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services” (2002)

[4] Golab, Wojciech. "Proving PACELC" (2018)

[5] Anand, S. “Netflix’s Transition to High-Availability Storage Systems” (2010)

[6] X Engineering Blog. “Manhattan, our real-time, multi-tenant distributed database for Twitter scale” (2014)

[7] Bailis, P., Ghodsi, A., & Ali, M. “Eventual Consistency Today: Limitations, Extensions, and Beyond” (2013)

[8] Hanwen Li, Mingtao Sun, Fan Xia, Xiaolong Xu, Muhammad Bila. “A Survey of Edge Caching: Key Issues and Challenges” (2022)

[9] Ladin, R., Liskov, B., & Shrira, L. “Lazy Replication: Exploiting the Semantics of Distributed Services” Technical Report, MIT (1990)

[10] Shapiro, M., Preguiça, N., Baquero, C., & Zawirski, M. “Conflict-Free Replicated Data Types. In Stabilization, Safety, and Security of Distributed Systems” (2011)

[11] Rahman, Golab, AuYoung, Keeton, J. Wylie. “Toward a Principled Framework for Benchmarking Consistency” (2012)

[12] Apache Cassandra and ScyllaDB use repair processes and Merkle-tree–based synchronization to reconcile divergent replicas. Amazon’s Dynamo and DynamoDB employ background replication and read-repair mechanisms to maintain eventual convergence.

[13] Willy Tarreau. “Elastic Binary Trees - ebtree” (2011)

[14] Bailis, Venkataraman, Franklin, Hellerstein, Stoica. “Probabilistically Bounded Staleness for Practical Partial Quorums“ (2012)

About Enea

Enea is a global specialist in advanced telecom and cybersecurity software, with a vision to make the world's communication safer and more efficient. Through continuous innovation, the solutions help secure, optimize, and monetize communication services. The company serves over 300 customers in more than 100 countries, including telecom operators, communication platform providers, cybersecurity vendors, and government agencies. Every day, more than 3 billion people rely on Enea software for seamless connectivity between people, businesses, and connected devices. The Enea headquarters is located in Stockholm, Sweden, and the company is listed on Nasdaq Stockholm. To learn more: www.enea.com

Authors

Andrej Gregić, Senior Software Architect
Erik Sigurdsson Rosenborg, Director Product Management