Design of Geo-Distributed Shared Logs to Support Partially-Replicated Transactional Systems

(Extended Abstract of the MSc Dissertation)

INÊS CARDEIRA, Instituto Superior Técnico, Universidade de Lisboa, Portugal
SUPERVISOR: PROFESSOR LUÍS RODRIGUES, Instituto Superior Técnico, Universidade de Lisboa, Portugal

Partial replication enables scalability provided it can be supported with genuine ordering protocols, i.e., protocols in which only the nodes involved in a transaction need to participate. The coordinator-based Skeen ordering algorithm is genuine and uses a linear number of messages, but its performance depends on the coordinator's location. In this work we evaluate the benefits of informed coordinator selection in geo-distributed transactional systems. We present an experimental evaluation of these techniques to assess their advantages over uninformed alternatives.

Additional Key Words and Phrases: Distributed Transactions; Partial Replication; Distributed Logs; Genuine total order.

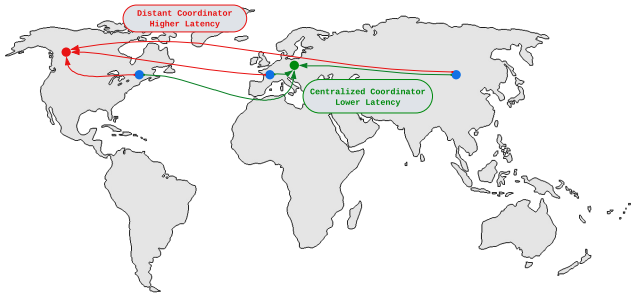

Fig. 1. Impact of coordinator placement on end-to-end latency: a distant coordinator (red) amplifies inter-continental RTTs; an RTT-central coordinator (green) shortens the coordination path.

1 Introduction

Many modern storage systems are simultaneously geo-distributed and partially replicated. These systems comprise multiple nodes—typically hosted in cloud data centers—located across different geographic regions and separated by high network latencies. Data are usually divided into partitions, and each node replicates one or more of these partitions.

To ensure high availability and durability, such systems replicate information across multiple regions. However, geographic replication introduces significant challenges for coordination among replicas, especially when the goal is to provide strong consistency guarantees. Many services require transactional guarantees such as serializability (or instantaneous isolation/strict serializability), which implies the need to establish a total order over transactions.

In recent years, many transactional systems have been built on shared distributed logs, which provide a total order of records as a fundamental property, enabling deterministic execution of operations at all nodes. Nonetheless, enforcing a total order in a geo-distributed, partially replicated system is non-trivial. Transaction ordering may require communication across multiple regions, which degrades performance when inter-regional latencies are high. In this article, we examine techniques for imposing a total order of transactions in geo-distributed, partially replicated systems.

More concretely, we focus on the design dilemma between genuine ordering algorithms and centralized (sequencer-based) ordering algorithms. Centralized ordering algorithms serialize operations in two communication steps with a linear number of messages. However, the cost of this round is essentially proportional to the distance between the transaction's originating region and the sequencer's region. As a result, workloads with strong locality—where transactions mostly interact among geographically close regions—still pay a high cost to contact a potentially remote sequencer.

Genuine ordering algorithms, such as Skeen's [5] algorithm, mitigate this problem by requiring coordination only among the regions that participate in a given transaction. To maintain a linear number of messages, Skeen selects a coordinator to aggregate and disseminate messages, using three communication steps. Consequently, the coordination latency becomes dependent on the coordinator's location.

Because Skeen adds one extra communication step compared to a single centralized sequencer, an appropriate choice of coordinator can help reduce this impact. Accordingly, we present an experimental evaluation to assess the benefits of informed coordinator selection in geo-distributed transactional systems.

2 Background

In the following paragraphs, we begin by contrasting sequencer-based ordering with genuine ordering, highlighting their latency and scalability trade-offs in geo-distributed settings. We then introduce Skeen's algorithm as a canonical genuine protocol, outlining its two-phase timestamping approach to achieve a consistent total order. Finally, we discuss practical execution variants and coordinator placement, explaining why the chosen aggregator's location critically shapes end-to-end latency.

2.1 Sequencer-Based vs. Genuine Ordering

Modern geo-distributed, partially replicated systems need a total order for multi-region transactions, but the way that order is produced has a first-order impact on latency. In sequencer-based designs, a designated site assigns positions in the log for all multi-region operations. This keeps message complexity low but stretches the critical path whenever participants are far from the sequencer—inter-continental round trips dominate even when the actual participants are close to each other. In practice, the sequencer can become both a latency and scalability pinch point in wide-area settings.

An alternative is genuine ordering [10], where only the regions that participate in a transaction exchange messages to place it in the total order. By confining coordination to the involved sites, genuine protocols preserve locality and avoid contacting uninvolved regions, so latency tracks nearby RTTs rather than a distant hub.

2.2 Skeen's Algorithm

A canonical genuine protocol is Skeen's algorithm. Each participant maintains a logical (Lamport) clock and the protocol runs in two phases:

(1) **Propose/Prepare:** each participating region assigns a local timestamp strictly greater than any previously issued and sends it;
(2) **Decide:** the global timestamp is the maximum of all proposals and defines the transaction's position in the total order at every participant, after which regions may advance their logical clocks and enqueue/commit consistently with local work.

There are several deployment variants. An all-to-all execution preserves genuineness but incurs quadratic messages as the participant set grows. Having the client aggregate proposals keeps messages linear but can inflate latency (and complicate fault-tolerance) if the client is remote. A common practical choice is a coordinator in one of the participating regions that aggregates proposals and disseminates the decided maximum—still linear messages and still genuine. However, end-to-end latency becomes highly sensitive to where that coordinator sits: a RTT-central coordinator shortens both phases; a distant one negates the locality benefits.

Because multi-home transactions in partially replicated systems frequently touch nearby regions, informed coordinator placement (choosing the RTT-central participant for each region-combination) can materially reduce Skeen's latency relative to both a fixed global sequencer and random coordinator choices, while retaining linear message complexity.

3 Related Work

This chapter surveys systems that tackle ordering and coordination in geo-distributed settings. We group prior work into shared (totally ordered) logs that rely on a sequencered path, tree-based architectures that scale metadata via hierarchical dissemination, and locality-aware/optimistic log designs that relax or restructure ordering to reduce wide-area coordination. For each group, we highlight the trade-offs among consistency guarantees, transaction support, and coordination mechanism. We then present a comparison table summarizing these dimensions to make cross-system differences explicit and to motivate our design choices.

3.1 Shared (Totally Ordered) Logs

Sequencered shared logs impose a single global order point that all multi-region operations must traverse. CORFU [3] establishes a centralized sequencer over a cluster of log servers, simplifying ordering but creating a WAN-distance sensitive critical path and a scalability choke under skew. vCorfu [18] keeps a sequencered design while adding transaction support, so multi-partition operations still pay the sequencer RTT. Scalog [8] reduces pressure on the sequencer with aggregation/indirection, improving elasticity but adding latency overhead in the ordering path. These patterns are strong on guarantees (strict serializability) but are fragile to inter-continental RTTs and hot-spotting in geo deployments.

3.2 Tree-Based Architectures

To move away from a single bottleneck, several systems propagate metadata along a hierarchy. Eunomia [11] organizes per-site update metadata and removes synchronous global stabilization from the critical path, lowering visibility latency for local operations. Saturn [6] uses compact causal labels and a serializer tree to enforce causal consistency with partial replication; its heuristic optimizes where to place serializers to minimize mismatch across sites. ENGAGE [4] targets edge/cloudlet settings, supporting session guarantees with bandwidth-aware dissemination. These systems scale metadata and relax coordination, but most do not support multi-partition transactions with strong guarantees; an exception is hierarchical deferred validation [2, 9] which validate single-partition transactions locally and multi-partition transactions hierarchically to enforce strict serializability.

3.3 Locality in Distributed Logs

Other lines of work vary ordering strength or reshape the ordering substrate to better reflect locality. FuzzyLog [15] lets many operations proceed on local edges and composes cross-site order as needed; it can be implemented with client-side aggregation patterns that avoid a fixed sequencer but are sensitive to where aggregation happens. SLOG [17] preserves strict guarantees using a sequencer; it performs well for local (single-home) paths but inherits centralized-path sensitivity for multi-home transactions. Detock [16] pushes optimistic local ordering with a DAG, cutting coordination on the happy path but facing conflict and deadlock issues under skew. Together, these highlight the trade-off surface between centralized total-order and more locality-preserving coordination.

3.4 Discussion

Table 1 outlines the key characteristics of the state-of-the-art systems previously discussed. We focus on three different aspects:

- **Consistency Model:** The consistency model is a critical aspect that determines the guarantees a system provides regarding the order and visibility of updates. Systems with strong consistency (e.g., CORFU, Scalog and SLOG) ensure strict serializability but often face significant coordination overhead in geo-distributed settings. In contrast, systems with weaker consistency models (e.g., Eunomia, Saturn and ENGAGE) prioritize scalability and efficiency but may limit applicability for workloads that require strict guarantees.
- **Transaction Support:** Transaction support is essential for systems that need to execute operations atomically across multiple partitions. Systems such as vCorfu, SLOG and Detock include robust transactional mechanisms, enabling them to handle operations that span multiple nodes. However, systems such as CORFU, Scalog and Saturn do not support transactions, which limits their applicability in use cases requiring atomicity.

Systems	Consistency	Trans.	Coordination
CORFU[3]	Strong	✗	Sequencer
Scalog[8]	Strong	✗	Sequencer w/ Aggregators
vCorfu[18]	Strong	✓	Sequencer
Eunomia[11]	Weak	✗	Tree
Saturn[6]	Weak	✗	Tree
ENGAGE[4]	Weak	✗	Tree
[2]	Strong	✓	Tree
FuzzyLog[15]	Strong	✓	Skeen (client-aggregated)
SLOG[17]	Strong	✓	Sequencer
Detock[16]	Strong	✓	Optimistic

Table 1. Comparison between systems presented in the Related Work

- **Coordination Mechanism:** The coordination mechanism determines how a system maintains order and consistency. Sequencer-based systems, such as CORFU and vCorfu, rely on centralized components to enforce total order. While this simplifies the coordination process, it introduces significant bottlenecks in scalability due to the reliance on a single point of failure. Scalog mitigates this scalability bottleneck by aggregating information regarding updates, but incurs a large latency overhead in the process. Fuzzylog adopts a client-based mechanism using Skeen's algorithm, which eliminates centralized coordination but can introduce latency if the client is geographically distant from participating nodes. Optimistic approaches, like the one used in Detock, allow the system to locally order multi-home transactions in each region assuming convergence without immediate synchronization. While this reduces the coordination overhead, it introduces potential challenges in resolving conflicts, especially under skewed workloads. Finally, tree-based mechanisms leverage hierarchical designs to propagate metadata efficiently, reducing coordination overhead and enabling scalability.

4 Proposed Solution

With our proposed architecture, we aim to address limitations observed in prior designs, particularly those of sequencer-based and traditional tree-based solutions when supporting transactional execution in geo-distributed settings [16, 17]. Sequencer-based approaches impose a single choke point: all transactions must traverse one location, which is suboptimal when participants are geographically distant from the sequencer, inflating both communication costs and end-to-end latency. These systems can also struggle under skewed workloads, where certain regions experience disproportionate transaction volumes.

Tree-based architectures mitigate some of these issues by structuring coordination hierarchically, avoiding a single central point. However, they still face bottlenecks for multi-home transactions. A key observation motivating our design is that nodes positioned at the "edges" of the logical tree (far left/right) may be close in physical geography but far in the logical hierarchy, forcing transactions to traverse the entire tree and incurring unnecessary latency.

4.1 System Model and Assumptions

Our architecture targets a transactional key-value store partitioned across geo-replicated regions. We assume a geo-distributed system with multiple data centers in distinct geographic locations. Each region hosts several servers and can replicate both data and control information locally. Without loss of generality, we assume that replication and concurrency control within a region are managed by a Paxos-style algorithm [12] that maintains a local log that records transactions that affect its corresponding object. This approach guarantees that updates are consistently applied across all replicas within the location. The system supports partial replication: each data fragment may be replicated in a distinct subset of regions, and some data may reside in a single region only.

Each region is the "home" of a subset of objects, as in [16, 17]. Transactions that access data stored solely within one region are called *single-home* transactions; transactions that access data homed across multiple regions are called *multi-home* transactions.

Single-home transactions do not require cross-region coordination. They are ordered by the region's local shared log and executed according to the total order defined by Paxos.

Multi-home transactions are ordered in two phases. First, an inter-regional coordination protocol establishes a total order across all multi-home operations. After a multi-home transaction is ordered with respect to other multi-home transactions, it is inserted into the local logs of the participating regions, interleaving with local transactions at each region. Within each region, Paxos ensures that replicas interleave transactions in a mutually consistent way. As in systems such as Spanner [7], we assume the Paxos leader in each region also acts as that region's representative for inter-regional coordination. Each step of the global coordination algorithm is recorded in the local log so coordination can continue if the leader replica must be replaced.

4.2 Coordinator-Based Architecture for Total Order

To order multi-home transactions, we employ Skeen's algorithm [5], which assigns timestamps to distributed events and imposes a total order by taking the maximum across proposed timestamps. Concretely, each participating region computes a local timestamp and proposes it to a coordinator; the coordinator determines the transaction's global timestamp by computing the maximum between all proposals, determining the transaction's position in the total order. Skeen's algorithm is *genuine* as defined by [10]: only the regions involved in a given transaction exchange messages to order it, which helps reduce latency.

A naïve all-to-all execution of Skeen incurs quadratic message cost in the number of participants. Delegating timestamp aggregation to the client (as in [15]) removes that quadratic factor but may increase latency if the client is geographically distant from the participating regions, and it complicates fault tolerance. To address these challenges, we introduce a set of coordinators: dedicated entities that aggregate proposed timestamps for each relevant combination of regions. Regions communicate only with the appropriate coordinator for their transactions, keeping message complexity linear with the number of participants and avoiding reliance on distant clients.

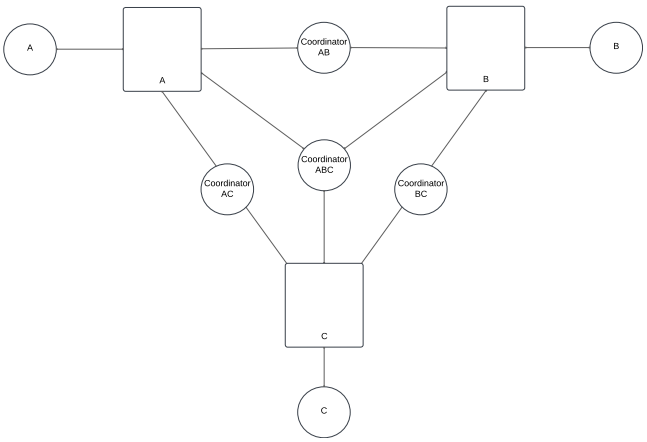

Fig. 2. Logical Representation

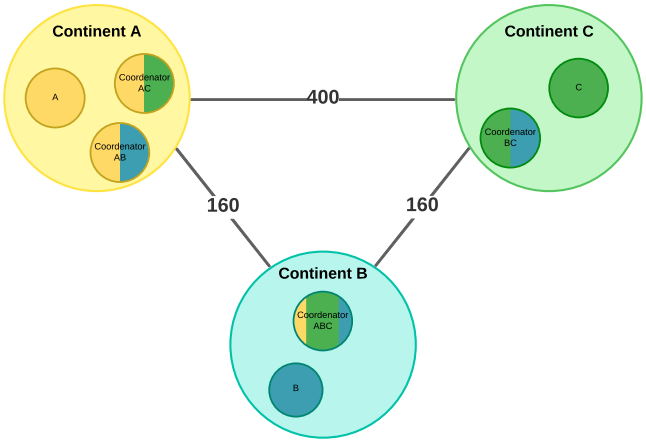

Fig. 3. Selection of Coordinators (Physical Representation)

In a system with n geo-locations, a coordinator is assigned to coordinate transactions involving all n locations. Additional coordinators manage subsets of locations, such as combinations of $n-1$ locations (with a total of $\binom{n}{n-1}$ such coordinators), combinations of $n-2$ locations (with $\binom{n}{n-2}$ coordinators), and so forth, down to combinations involving two geo-locations. This organization eliminates direct all-to-all communication among regions while ensuring that every multi-home transaction has a well-defined coordination endpoint.

Figure 2 provides a logical representation of the proposed architecture in a system composed of three distinct geo-locations. Each geo-location is responsible for managing a specific object ("A", "B" or "C"). The figure also illustrates the coordinators and their connections, highlighting how they interact. For example, a multi-home transaction involving geo-locations "A", "B" and "C" is assigned to the coordinator "ABC", which is responsible for determining a total order for transactions involving that combination of geo-locations.

In our design, Skeen's algorithm is executed collaboratively between regions and coordinators: regions propose timestamps; the designated coordinator for the involved combination of regions collects those proposals, computes the global timestamp as their maximum, and disseminates the result to the participants. Each region may then advance its logical clock if needed and enqueue the transaction for insert in the local shared log, ensuring that global transactions are consistently ordered across local logs.

At this stage, we assume fault tolerance will follow an approach similar to CORFU and NoPaxos [13]: upon a failure, the system briefly halts to reconfigure, ensuring that all participants agree on and are informed of the updated configuration (including coordinator assignments) before resuming.

4.3 Coordinator Placement

In some settings, the client submitting the transaction could act as Skeen's coordinator. We avoid this for two reasons: (i) if the client is remote, latency increases; (ii) client failures are harder to tolerate than server failures, whereas servers replicate their state via Paxos.

Coordinator selection can be random among the regions involved, but we evaluate an informed alternative that leverages topology (and, when available, load) to minimize latency. When a transaction is submitted, the system considers the involved regions and estimates the latency of Skeen's execution for each possible coordinator location, choosing the region that minimizes the estimated latency. This choice can be precomputed during system configuration. Thus, for every possible combination of regions, we assume a pre-assigned coordinator known to all participants.

Home regions are hosted in distinct data centers. Latencies are measured as round-trip times between data centers, so coordinators should be placed as centrally as possible with respect to the home regions they serve, minimizing the RTT of the message exchange for multi-home transactions.

Figure 3 illustrates this principle with three geo-locations, which in this example constitute the only data centers available to the system. Given the RTTs shown in the figure, the latency-minimizing choice for the three-way coordinator "ABC" is region "B", which is RTT-central with respect to "A" and "C". Placing the coordinator at "A" or "C" would systematically force one participant to traverse a longer RTT path, increasing end-to-end coordination time.

Because deployment is restricted to home-region data centers, the coordinator for a pair (e.g., "AB", "BC", "AC") should be placed in one of the two regions involved ("AB" in "A" or "B"; "BC" in "B" or "C"; "AC" in "A" or "C"). If additional data centers were available, the preferred location would be the site that minimizes RTT to both regions—the latency "midpoint". Note that this midpoint could either be a third location or still coincide with one of the two regions when that region already minimizes the combined RTT. To avoid recreating a sequencer-like bottleneck, we also avoid concentrating all pairwise coordinators in a single geo-location.

In the current prototype, coordinator choices are configured statically. Making this process dynamic is not conceptually complex and is left for future work. In this version we use network latency as the sole criterion, but we plan to incorporate regional load to further balance the system.

4.4 Transaction Processing

Transactions are classified as single-home or multi-home, with distinct handling paths.

4.4.1 Single-Home Transactions. Single-home transactions involve only one region and can be processed and committed locally without communication with other regions or coordinators.

(1) **Client Begins Transaction:** The client sends the transaction to the responsible region (e.g., a transaction over object "A" is sent to region "A").
(2) **Local Check and Commit:** The region determines the transaction is single-home. Since no external coordination is needed, the transaction is inserted directly into the region's local log, which maintains a totally ordered sequence of transactions using a Paxos-based mechanism. This minimizes latency and enables fast, efficient processing of local operations.

4.4.2 Multi-Home Transactions. Multi-home transactions span multiple regions and require coordination to ensure a globally consistent total order and coherent interleaving with local transactions.

(1) **Client Begins Transaction:** The client submits the transaction to all involved regions.
(2) **Local Check and Forwarding to Coordinator:** Each region checks whether the transaction affects other regions. If so, it classifies the transaction as multi-home. Each region maintains a monotonically increasing logical clock that advances whenever it processes a new multi-home transaction. The region assigns the transaction a local timestamp strictly greater than any previously issued at that location. Regions also record, locally, the ordered set of transactions they have timestamped to ensure later that confirmations respect local order.
(3) **Coordination via Skeen's Algorithm:** The involved regions and the pre-assigned coordinator for that combination (e.g., "AB" for "A" and "B") collaboratively run Skeen's algorithm. Each region sends its proposed timestamp to the coordinator, which aggregates proposals and computes the global timestamp as their maximum. This ensures a total order across participants while keeping communication overhead linear.
(4) **Logical Clock Update:** Upon receiving the global timestamp, all participating regions advance their logical clocks as needed to avoid assigning future timestamps smaller than the agreed value.
(5) **Order Check Before Commit:** Before committing a transaction, each region ensures there are no pending transactions to which it assigned smaller local timestamps. A transaction can only advance once all older-timestamped transactions have been confirmed, preserving local sequential consistency.
(6) **Commit:** After the global timestamp is established, the transaction is added to a queue at each participating region. When it reaches the top, Paxos is used to insert the transaction into the local log in the globally agreed order.

5 Evaluation

In this chapter, we evaluate the impact of carefully choosing the coordinator for each transaction under different levels of locality for global transactions. We compare our architecture against two baselines: SLOG [17], which uses a single centralized coordinator for all global transactions, and Skeen's algorithm with random coordinator selection.

To ensure a fair comparison, we reused and extended SLOG's public C++ codebase to build a shared foundation that supports both our approach and SLOG's centralized coordinator. Both variants run on the same networking substrate and common implementation, ensuring uniform evaluation conditions.

In SLOG, the ordering of global transactions is always performed by a single, preselected region, regardless of which regions the transaction touches. In our setup, that central coordinator is placed in Continent A, at *a0*. This fixed choice penalizes latency when the accessed data is far from *a0*.

Our workload consists exclusively of global transactions. We vary (i) the number of participating regions and (ii) the degree of locality among participants to study how locality shapes performance.

5.1 Experimental Setup

We use nine physical machines, each representing one region of the geo-distributed system. The same machines also host concurrent clients, with 9 client processes per machine. Each transaction reads and writes 9 objects, uniformly distributed across the regions it touches.

Contention is controlled via a dispersion parameter d, which governs the fraction of "hot" objects (i.e., those more frequently accessed). Larger d spreads accesses across more objects, reducing contention. Following the state-of-the-art [16], we set a high dispersion value of d=10000, corresponding to very low contention.

To emulate a realistic geo-distributed setting, we introduce artificial inter-region latencies. These values are hand-configured to reflect typical geographic relationships—lower RTTs within a continent and higher RTTs across continents. Figure 4 shows the system topology and configured RTTs between regions and continents. Region *a0* is chosen as SLOG's central coordinator (in red in Figure 4). In our system, the inter-region coordinator is placed at region *b0* (shown in yellow in Figure 4), which minimizes the RTT for these transactions under this topology. The central regions of each continent are *a0*, *b0*, and *c0*, respectively.

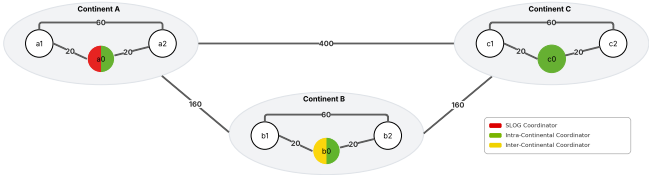

Fig. 4. System topology. The red region denotes SLOG's central coordinator. Green regions denote coordinators for intra-continental Skeen transactions. The yellow region denotes the coordinator for inter-continental Skeen transactions.

5.2 Intra-Continental Locality

We begin with a high-locality scenario. In this experiment, global transactions interact only with regions within their origin continent, and they touch all regions in that continent. We refer to these as intra-continental transactions.

Figure 5 shows the CDF of end-to-end latency for global transactions originating from each continent: Continent A, Continent B, and Continent C. Under high locality, Skeen outperforms SLOG in regions far from the centralized sequencer, achieving up to a 4× latency improvement in Continent C. Although Skeen requires twice as many coordination steps as SLOG, the aggregate cost of intra-continental communication is significantly lower than inter-continental communication. This additional step cost is visible in Continent A, where SLOG performs better.

We also observe that the choice of coordinator within a continent materially affects Skeen's performance. Consider Continent B, where *b0* is centrally located relative to *b1* and *b2*. Placing the coordinator at *b0* minimizes Skeen's execution time in two ways: (i) it reduces the protocol's worst-case round-trip (from start to the arrival of the final timestamp at all participants), directly lowering transaction latency from the client's perspective; and (ii) it shortens the time transactions remain tentative in regions, which helps unblock commits that are waiting behind concurrently ordered transactions. Together these effects reduce mean inter-region transaction latency by roughly 35%.

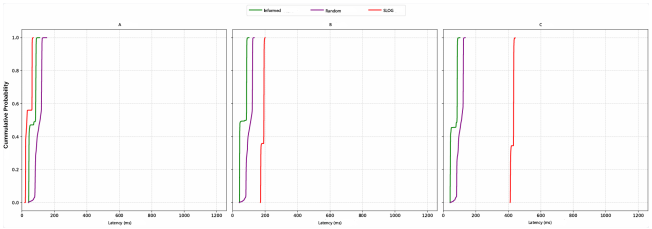

Fig. 5. Per-region latency CDFs, comparing random coordinator selection with informed selection of the most central region.

Inter-Continental Locality

We now consider a mixed scenario in which, besides intra-continental transactions, a small fraction (10%) of transactions are inter-continental, touching one region from each continent. The choice of 10% reflects typical edge workloads in which the vast majority of transactions are intra-region or locality-preserving (e.g. 90%) [1, 14]; we introduce a modest inter-continental tail to stress the system under realistic asymmetry.

Figure 6 reports the latency CDF per region. Introducing inter-continental transactions significantly affects Skeen-based protocols, while SLOG remains comparatively stable. The degradation arises from the large disparity between intra- and inter-continental communication times. Intra-continental transactions obtain their final timestamps much faster than inter-continental ones, creating queues that wait for the final global order of slow inter-continental operations—i.e., the convoy effect.

In the presence of this effect, the latency gains of Skeen diminish substantially: Skeen exhibits worse mean latency than SLOG in Continent A and Continent B. However, both systems still show better mean latency in Continent C, where the communication cost to the Continent A–based centralized coordinator dominates SLOG's end-to-end time.

While informed coordinator selection still helps, its impact is reduced by the convoy effect, yielding only about a 10% mean-latency improvement for transactions originating in Continent A and Continent C. The gains remain significant in Continent B—about 40% versus random—because the informed choice more strongly shortens Skeen's execution time there.

For a Continent B–origin inter-continental transaction, the time the transaction remains tentative in Continent B is approximately 360, 160, and 360 milliseconds when the Skeen coordinator is in Continent A, Continent B, or Continent C, respectively—an average of 293 ms under random choice, about 80% higher than the constant 160 ms under informed placement. In contrast, for Continent C– and Continent A–origin transactions, the average benefit of informed selection over random is only about 7%, as coordination time is dominated by the Continent C and Continent A communication cost regardless of where the coordinator is placed.

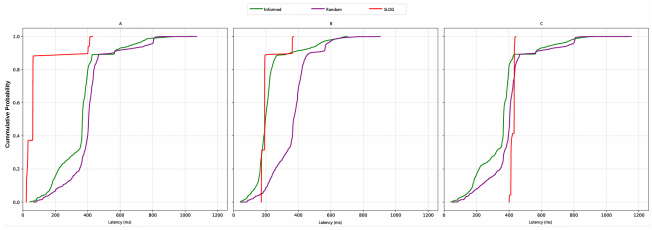

Fig. 6. Latency CDFs by region, comparing the impact of coordinator selection in the presence of inter-region transactions.

5.3 Visualizing the Convoy Effect

To make the convoy effect concrete, Figure 7 shows a timeline at region *c1*. A slower inter-continental transaction $T0$ completes after long RTTs, while a faster intra-continental transaction $T1$ arrives later but cannot be confirmed because $T0$ holds a smaller local timestamp. The shaded interval highlights the waiting time before $T1$ can be confirmed, which accumulates across many such cases and shifts the CDFs reported above.

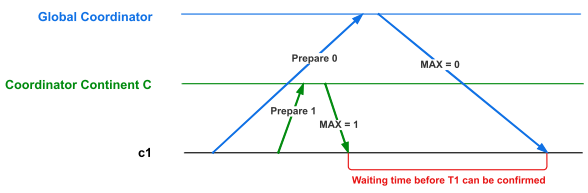

Fig. 7. Illustrative timeline at region *c1*. The shaded interval marks the waiting time before $T1$ can be confirmed. A slower inter-continental $T0$ finalizes after long RTTs; a faster intra-continental $T1$ must wait because $T0$ has the smaller local timestamp.

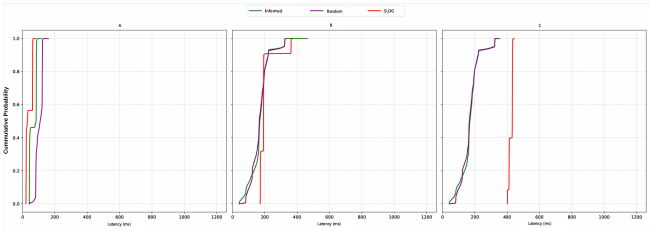

Fig. 8. Per-region latency CDFs with inter-continental transactions limited to Continent B–Continent C. Removing the Continent C–A leg reduces Skeen's execution time and its convoy.

5.4 Isolating the Convoy Effect

To isolate the effect, we modify the mixed workload so inter-continental transactions span only Continent B and Continent C (one region from each), excluding Continent A. Because these transactions touch two continents, the coordinator for such transactions is chosen at random between the two involved regions.

Figure 8 shows per-region CDFs. Skeen-based systems perform markedly better in Continent B and Continent C than in the previous mixed case. The improvement stems from removing the slow Continent C–A leg from inter-continental coordination, which shortens Skeen's execution time and mitigates its associated convoy. In continents still affected by the convoy, informed selection offers little advantage over random—the slow path dictates the critical path, overshadowing intra-continental gains.

6 Conclusions

In this work, we investigated genuine total ordering for geo-distributed transactional systems, focusing on Skeen's algorithm and the role of informed coordinator selection. A coordinator-based design was presented to retain linear message complexity, together with a physical placement principle grounded in RTT-centrality among participating home regions. Evaluation showed substantial latency gains under high intra-continental locality while also exposing a limiting factor: in mixed workloads, RTT asymmetry combined with local admission rules induces a convoy effect that diminishes Skeen's advantages and can favor a centralized sequencer in some regions. Overall, informed coordinator placement is effective but insufficient on its own; robustness in heterogeneous wide-area settings requires additional mechanisms that explicitly address convoy formation.

Acknowledgments

I would like to thank Rafael Soares for his help and for the thoughtful discussions that shaped and strengthened this thesis.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia, I.P. (FCT) under projects UID/50021/2025 and UID/PRR/50021/2025 and GLOG (financed by the OE with ref. LISBOA2030-FEDER-00771200).

References

[1] Ailidani Ailijiang, Aleksey Charapko, Murat Demirbas, and Tevfik Kosar. 2020. WPaxos: Wide Area Network Flexible Consensus. *IEEE Trans. Parallel Distributed Syst.* 31, 1 (2020), 211–223.

[2] J. Amaro. 2018. A Distributed and Hierarchical Architecture for Deferred Validation of Transactions in Key-Value Stores. (2018).

[3] Mahesh Balakrishnan, Dahlia Malkhi, John D. Davis, Vijayan Prabhakaran, Michael Wei, and Ted Wobber. 2013. CORFU: A distributed shared log. *ACM Trans. Comput. Syst.* 31, 4 (2013), 10.

[4] Miguel Belém, Pedro Fouto, Taras Lykhenko, João Leitão, Nuno M. Preguiça, and Luís Rodrigues. 2022. ENGAGE: Session Guarantees for the Edge. In *31st International Conference on Computer Communications and Networks, (ICCCN)* (Honolulu (HI), USA).

[5] Kenneth P. Birman and Thomas A. Joseph. 1987. Reliable communication in the presence of failures. *ACM Trans. Comput. Syst.* 5, 1 (1987), 47–76.

[6] Manuel Bravo, Luís E. T. Rodrigues, and Peter Van Roy. 2017. Saturn: A Distributed Metadata Service for Causal Consistency. In *Proceedings of the Twelfth European Conference on Computer Systems (EuroSys)* (Belgrade, Serbia).

[7] James C. Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, J. J. Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, Wilson C. Hsieh, Sebastian Kanthak, Eugene Kogan, Hongyi Li, Alexander Lloyd, Sergey Melnik, David Mwaura, David Nagle, Sean Quinlan, Rajesh Rao, Lindsay Rolig, Yasushi Saito, Michal Szymaniak, Christopher Taylor, Ruth Wang, and Dale Woodford. 2013. Spanner: Google's Globally Distributed Database. *ACM Trans. Comput. Syst.* 31, 3 (2013), 8.

[8] Cong Ding, David Chu, Evan Zhao, Xiang Li, Lorenzo Alvisi, and Robbert van Renesse. 2020. Scalog: Seamless Reconfiguration and Total Order in a Scalable Shared Log. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)* (Santa Clara (CA), USA).

[9] Jon Grov and Peter Csaba Ölveczky. 2013. Scalable and Fully Consistent Transactions in the Cloud through Hierarchical Validation. In *International Conference on Data Management in Cloud, Grid and P2P Systems* (Prague, Czech Republic).

[10] Rachid Guerraoui and André Schiper. 2001. Genuine atomic multicast in asynchronous distributed systems. *Theoretical Computer Science* 254, 1 (2001), 297–316.

[11] Chathuri Gunawardhana, Manuel Bravo, and Luís E. T. Rodrigues. 2017. Unobtrusive Deferred Update Stabilization for Efficient Geo-Replication. In *2017 USENIX Annual Technical Conference (ATC)* (Santa Clara (CA), USA).

[12] Leslie Lamport. 2002. Paxos Made Simple, Fast, and Byzantine. In *Procedings of the 6th International Conference on Principles of Distributed Systems.* (Reims, France). 7–9.

[13] Jialin Li, Ellis Michael, Naveen Kr. Sharma, Adriana Szekeres, and Dan R. K. Ports. 2016. Just Say NO to Paxos Overhead: Replacing Consensus with Network Ordering. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)* (Savannah (GA), USA).

[14] Yi Lin, Bettina Kemme, Marta Patiño-Martínez, and Ricardo Jiménez-Peris. 2007. Enhancing Edge Computing with Database Replication. In *26th IEEE Symposium on Reliable Distributed Systems (SRDS 2007), Beijing, China, October 10-12, 2007* (Beijing, China). 45–54.

[15] Joshua Lockerman, Jose Faleiro, Juno Kim, Soham Sankaran, Daniel Abadi, James Aspnes, Siddhartha Sen, and Mahesh Balakrishnan. 2018. The FuzzyLog: A Partially Ordered Shared Log. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)* (Ann Arbor (MI), USA). 357–372.

[16] Cuong D. T. Nguyen, Johann K. Miller, and Daniel J. Abadi. 2023. Detock: High Performance Multi-region Transactions at Scale. *Proceedings of the ACM on Management of Data* 1, 2 (2023), 1–27.

[17] Kun Ren, Dennis Li, and Daniel J. Abadi. 2019. SLOG: Serializable, Low-latency, Geo-replicated Transactions. *Proceedings of the VLDB Endowment* 12, 11 (2019), 1747–1761.

[18] Michael Wei, Amy Tai, Christopher J. Rossbach, Ittai Abraham, Maithem Munshed, Medhavi Dhawan, Jim Stabile, Udi Wieder, Scott Fritchie, Steven Swanson, Michael J. Freedman, and Dahlia Malkhi. 2017. vCorfu: A Cloud-Scale Object Store on a Shared Log. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (Boston (MA), USA).