Performant Synchronization in Geo-Distributed Databases

DULING XU, Renmin University of China, China
TONG LI*, Renmin University of China, China
ZEGANG SUN, Renmin University of China, China
ZHENG CHEN, Tsinghua University, China
WEIXING ZHOU, Northeastern University, China
YANFENG ZHANG, Northeastern University, China
WEI LU, Renmin University of China, China
XIAOYONG DU, Renmin University of China, China

The deployment of databases across geographically distributed regions has become increasingly critical for ensuring data reliability and scalability. Recent studies indicate that distributed databases exhibit significantly higher latency than single-node databases, primarily due to consensus protocols maintaining data consistency across multiple nodes. We argue that synchronization cost constitutes the primary bottleneck for distributed databases, which is particularly pronounced in wide-area network (WAN). Fortunately, we identify opportunities to optimize synchronization costs in real production environments: 1) Network clustering phenomena, 2) Triangle inequality violations in transmission, and 3) Redundant data transfers. Based on these observations, we propose GeoCoCo, a synchronization acceleration framework for cross-region distributed databases. First, GeoCoCo presents a group rescheduling strategy that adapts to real-time network conditions to maximize WAN transmission efficiency. Second, GeoCoCo introduces a task-preserving data filtering method that reduces data volume transmitted over the WAN. Finally, GeoCoCo develops a consistency-guaranteed transmission framework integrating grouping and pruning. Extensive evaluations in both trace-driven simulations and real-world deployments demonstrate that GeoCoCo reduces synchronization cost—primarily by lowering WAN bandwidth usage—by up to 40.3%, and this improves system throughput by up to 14.1% in GeoGauss.

CCS Concepts: • **Information systems → Middleware for databases**.

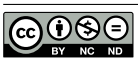
Additional Key Words and Phrases: Geo-distributed databases, synchronization, consistency, WAN, distributed systems

ACM Reference Format:
Duling Xu, Tong Li*, Zegang Sun, Zheng Chen, Weixing Zhou, Yanfeng Zhang, Wei Lu, and Xiaoyong Du. 2026. Performant Synchronization in Geo-Distributed Databases. *Proc. ACM Manag. Data* 4, 1, Article 63 (February 2026), 28 pages. https://doi.org/10.1145/3786677

1 INTRODUCTION

With the growth of global-scale services, distributed databases have become the standard for managing data across geographically dispersed regions, offering high availability and strong consistency [17, 19, 30, 36, 72]. Yet, maintaining consistency requires frequent synchronization,

* Tong Li is the corresponding author (tong.li@ruc.edu.cn).
Authors' Contact Information: Duling Xu, Renmin University of China, Beijing, China, xuduling@ruc.edu.cn; Tong Li*, Renmin University of China, Beijing, China, tong.li@ruc.edu.cn; Zegang Sun, Renmin University of China, Beijing, China; Zheng Chen, Tsinghua University, Beijing, China; Weixing Zhou, Northeastern University, Shenyang, China; Yanfeng Zhang, Northeastern University, Shenyang, China; Wei Lu, Renmin University of China, Beijing, China; Xiaoyong Du, Renmin University of China, Beijing, China.



ACM 2836-6573/2026/2-ART63
https://doi.org/10.1145/3786677

which introduces significant overhead—especially in geo-distributed settings where transmission can dominate transaction latency (over 60% in systems like Spanner and GeoGauss) [17, 92]. These challenges highlight the need for a synchronization-oriented communication framework that reduces coordination complexity and minimizes wide-area communication cost.

Existing efforts to improve synchronization efficiency in geo-distributed databases mainly fall into three categories. First, transaction-level optimizations mitigate synchronization latency through scheduling and locality-aware execution [14, 46, 60, 72]. For instance, SLOG [60] pre-orders cross-region transactions near their data to reduce coordination overhead. Second, network-level approaches accelerate data transfer via advanced transport protocols [9, 38, 90]. BBR [9], for example, estimates bottleneck bandwidth and RTT to maintain high throughput and low latency. Third, data-reduction techniques compress replicated payloads to lower WAN traffic [55, 67, 81]. zlib [55] exemplifies this by providing efficient, lossless compression for synchronization data.

Despite notable performance gains, prior approaches treat synchronization as a coarse-grained monolithic service, missing opportunities for fine-grained hardware utilization. In a fully replicated distributed database, one synchronization round requires $n(n-1)$ point-to-point communications over $n(n-1)$ distinct paths. While intra-datacenter settings assume near-uniform latency (e.g., ~5 ms) and symmetric bandwidth, this assumption breaks down in geo-distributed environments. For instance, latency within California is under 4 ms, but rises to 81.12 ms between Northern California and Central Canada, and 288.45 ms to Cape Town—spanning two orders of magnitude, with inter-region paths varying by 2–5×. Executing uniform communication over such heterogeneous WAN paths introduces synchronization stalls, motivating a fine-grained re-examination of synchronization bottlenecks.

We conduct empirical measurements and trace-driven analyses to understand synchronization behavior in production-like environments. Our study reveals three key phenomena in geo-distributed deployments: (1) strong geographical aggregation, where certain data centers naturally act as traffic hubs; (2) high redundancy in update propagation, especially among neighboring replicas; and (3) speculative triangular paths that can outperform direct links. These findings indicate that substantial communication overhead can be removed by redesigning synchronization to exploit these real-world characteristics. Motivated by this, we construct latency-aware synchronization groups aligned with structural properties of geo-distributed systems: (1) identifying natural aggregation hubs for routing and consolidation, (2) eliminating redundant transmissions by leveraging update overlap, and (3) exploiting low-latency triangular paths for faster synchronization. Achieving this requires addressing three challenges. First, group formation and aggregation scheduling must jointly optimize global latency under dynamic and asymmetric WAN conditions. Second, the space of possible grouping strategies grows rapidly with node count, raising scalability concerns. Finally, practical deployment demands a lightweight, modular solution that integrates transparently by modifying only communication invocation logic.

To address these challenges, we propose GeoCoCo, a general-purpose synchronization framework for geo-distributed databases. GeoCoCo decouples synchronization from point-to-point communication and instead performs group-based synchronization in two stages: first aggregating updates within groups, then coordinating inter-group transmission via designated aggregators. To optimize end-to-end performance, GeoCoCo incorporates a latency-aware group planner and a dynamic aggregation scheduler that jointly minimize makespan. In addition, it exposes a high-level interface that enables seamless integration by only modifying communication invocation logic, without altering transaction or storage modules. We empirically compare GeoCoCo against state-of-the-art baselines in real WAN settings using TPC-C and YCSB. GeoCoCo improves end-to-end throughput by up to 14.1% on GeoGauss and 11.5% on CockroachDB, while reducing synchronization (WAN) cost by up to 40.3%. These results validate GeoCoCo's effectiveness across systems and workloads. In

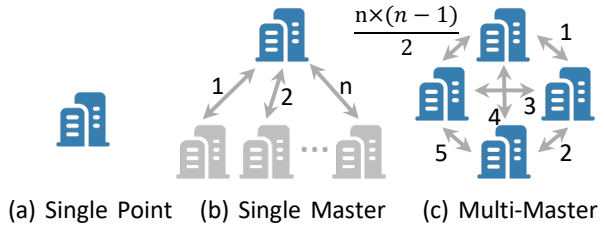


Fig. 1. Cross-Domain Communication Complexity in three Database Architectures: (a) Single Point vs. (b) Single Master vs. (c) Multi-Master.

summary, we address the performance bottlenecks of synchronization in geo-distributed databases and make the following three **contributions**:

- ***New Opportunity:*** **Fine-Grained Analysis of Synchronization Inefficiencies.** We identify key inefficiencies in geo-distributed synchronization and uncover three actionable opportunities: geographical aggregation hubs, redundant data transmissions, and speculative low-latency triangular paths. These insights lay the foundation for communication optimization beyond traditional designs.
- ***New Transmission Schedule:*** **Hierarchical Latency-Aware Synchronization.** We propose a hierarchical synchronization design that forms latency-aware groups and routes updates through aggregation hubs, leveraging both structural and speculative transmission advantages. This reduces cross-region overhead and accelerates global coordination.
- ***New Data Reduction Method:*** **Redundancy-Aware Update Dissemination.** We reduce communication volume by identifying overlapping update contents among replicas and proactively removing redundancy through the two-level synchronization hierarchy. This redundancy-aware design reduces transmission cost without compromising consistency.

2 BACKGROUND

2.1 Geo-distributed Database System

To support multinational operations, enterprises deploy data centers across regions, driving the growth of geo-replicated distributed databases. These systems must ensure high availability, strong consistency, and low-latency cross-region access, and are widely adopted in finance, e-commerce, social platforms, and online gaming. The key challenge lies in efficient synchronization and transaction processing over WANs, where high latency, limited bandwidth, and asynchrony are inherent. With the expansion of cloud and edge computing, demand for scalable and flexible data services continues to rise, accelerating the evolution of geo-replicated databases [8, 17, 71, 75, 92]. Representative systems such as Google Spanner [17], CockroachDB [71], Amazon Aurora [75], and GeoGauss [92] have advanced consensus, replication, and network-aware optimizations, establishing geo-replicated databases as critical infrastructure for enterprise competitiveness. Mainstream systems adopt either **Single-Master** or **Multi-Master** architectures (Figure 1). In Single-Master systems (e.g., Spanner [17], TiDB [35], CockroachDB [71]), leaders handle writes and replicate to followers, incurring high cross-region latency due to centralized routing and 2PC overhead [25]. In contrast, Multi-Master systems eliminate the single-leader bottleneck but introduce $O(n^2)$ synchronization cost, as each update requires $n(n-1)$ cross-region messages [92]. Consequently, communication scalability remains the primary barrier to global-scale deployment.

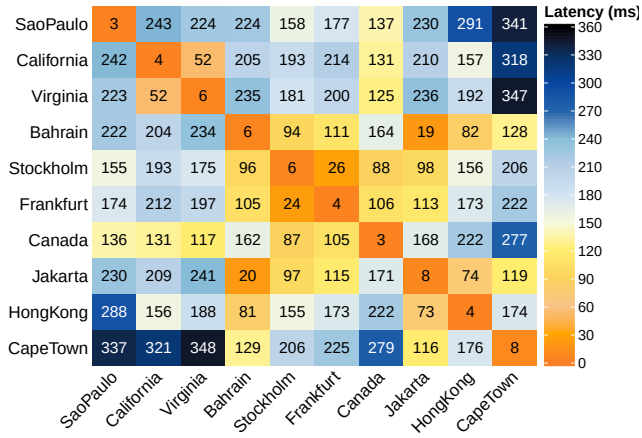
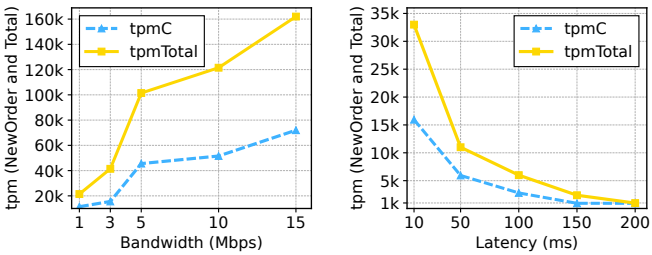


Fig. 2. Measured network latency between Amazon EC2 sites in 10 different regions.



(a) Varying Bandwidth. (b) Varying Latency.

Fig. 3. Performance Impact of WAN Bandwidth and Latency on Multi-Master Systems Transactions.

2.2 WAN Bandwidth and Latency

The network characteristics of Wide Area Network (WAN) differ fundamentally from those of intra-datacenter Local Area Networks (LANs), posing significant challenges to the design of geo-distributed systems. On modern cloud platforms, WAN environments exhibit three critical limitations: limited bandwidth, high and variable latency, and unpredictable latency fluctuations [49]. **First**, WAN bandwidth is significantly lower than LAN bandwidth—on average 15× and up to 60–80× lower in extreme cases [33, 84]. Purchasing WAN bandwidth comparable to LAN speeds is also substantially more expensive, often 10× higher per unit [42, 62]. These cost-performance trade-offs make large-scale, bandwidth-heavy tasks (e.g., all-reduce, global synchronization) highly constrained under WAN settings [59]. **Second,** WAN suffer from high absolute latency and inter-region variability(see Figure 2). Measurements from 10 global regions on Amazon's cloud [63] show latencies ranging from 26 ms (Stockholm–Frankfurt) to over 337 ms (São Paulo–Cape Town), introducing nearly 10× variability. Latency jitter further exacerbates coordination unpredictability. In Multi-Master databases, even a single region's delay can postpone global snapshots and validation, stalling progress across all sites. In short, WAN introduces severe performance bottlenecks that undermine the efficiency of geo-distributed coordination. Addressing these bandwidth and latency constraints forms the basis for our system's design and analysis in subsequent sections.

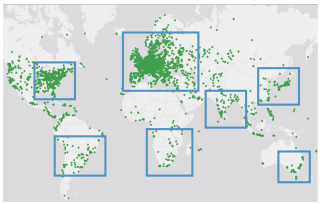


Fig. 4. Observed Geographical Clustering of Data Centers Worldwide.

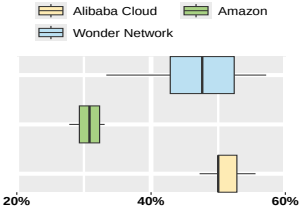


Fig. 5. Proportion of nodes that violate the triangle inequality (%).

2.3 Distributed Database System on WAN

Geographically distributed Multi-Master databases face inherent WAN limitations. While they enable low-latency local access, maintaining global consistency incurs high WAN overhead, as every update must reach remote replicas. We evaluate this effect on GeoGauss using a five-node geo-distributed setup (Figure 3) running the TPC-C workload [74]. Reducing bandwidth from 15 Mbps to 1 Mbps rapidly saturates the network and degrades throughput, as each transaction's writes are broadcast to all replicas. Likewise, increasing RTT from 10 ms to 200 ms significantly raises latency and lowers throughput, since strong consistency requires acknowledgments from all sites [17, 45]. These results confirm that WAN latency and bandwidth dominate system performance, shifting the bottleneck from computation to communication and motivating our WAN-aware synchronization design.

In this work, we focus on abstracting these collective communication patterns over WAN, as geo-distributed multi-master databases require a collective communication framework that operates beyond the boundaries of a single data center.

3 MOTIVATION

After conducting an in-depth exploration of distributed databases deployed in real-world wide area networks (WANs), we present three key observations.

Observation #1: Geographically-close nodes naturally form low-latency clusters. Due to geographic proximity, data centers naturally form small clusters(see Figure 4). Nodes within the same cluster experience significantly lower communication latency, whereas inter-cluster communication exhibits considerably higher delays. This characteristic provides an optimization opportunity for layered communication strategies: prioritizing intra-cluster communication and treating inter-cluster communication as an independent layer for optimization. For example, by minimizing the use of high-latency inter-cluster paths, the latency bottleneck can be effectively alleviated, and more refined group-based communication management becomes feasible.

Observation #2: At high conflict ratios, some WAN traffic becomes wasteful and ineffective. We define *white data* as updates that are eventually discarded during synchronization without affecting the final system state. In geo-distributed multi-master databases, such data often stems from conflicting writes, duplicate updates, or stale transactions [36, 72, 78]. Although white data consumes considerable WAN bandwidth—especially under all-to-all patterns—it holds no semantic value and is ultimately filtered by the application layer [21]. Eliminating such data reduces WAN overhead without compromising consistency and forms a core optimization target in our design. To quantify its impact, we measured white data ratios in a geo-distributed database. Depending on workload and conflict intensity, it accounts for 20.12–45.23% of transmitted updates in our TPC-C and YCSB experiment, introducing unnecessary bandwidth usage and queuing delay [69].

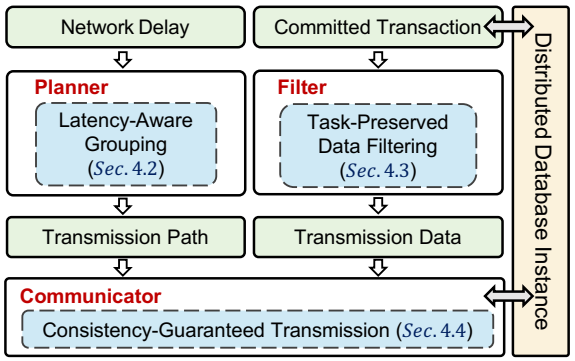


Fig. 6. Overview of GeoCoCo.

In our experience with Huawei's production environments, up to 20–45% of synchronized updates are "white data" with no observable remote effect, demonstrating filtering's practical value under standard benchmarks. Early filtering reduces total transmission cost and improves end-to-end latency. When combined with semantic aggregation at intermediate nodes, communication becomes significantly more efficient.

Observation #3: The link latencies in wide-area networks often violate the triangle inequality. In data center networks, the triangle inequality typically holds, meaning that the sum of two latency segments is always greater than the direct latency of a single hop. However, in WAN, this rule is frequently violated, we call it Triangle Inequality Violation (TIV). For instance, the direct communication latency between nodes A and C may be 100ms, whereas routing through an intermediate node B (A → B → C) results in a total latency of only 80ms. This forms a triangle in which the latency inequality is violated. As shown in Figure 5, statistical data from 3 real-world WAN datasets indicate that 28 %-57 % of node pairs exhibit such violations [2, 63, 77]. This counter-intuitive phenomenon creates an opportunity for latency optimization: by dynamically selecting indirect paths when they offer lower delay, communication latency—and thus makespan—can be reduced. This is especially beneficial for latency-sensitive operations such as all-reduce and all-to-all.

4 GeoCoCo DESIGN

4.1 GeoCoCo Overview

GeoCoCo is a distributed transmission framework designed to accelerate synchronization in geo-distributed database systems by optimizing collective communication, thereby enhancing overall database performance. As illustrated in Figure 6, GeoCoCo is deployed between the communication subsystem and distributed database instances, acting as an intelligent adaptation framework that dynamically optimizes cross-domain data flows. The system consists of three core modules: the Planner, the Filter, and the Communicator, each responsible for distinct dimensions of optimization.

Planner. The Planner module is responsible for optimizing transmission coordination through real-time WAN monitoring and adaptive participant grouping. Observation #1 inspires the design of GeoCoCo's grouped transmission mechanism. The Planner continuously monitors wide-area network conditions—such as link latency—via a built-in real-time observation component. Based on the observed network dynamics, it invokes a linear programming–based grouping strategy orchestrator that adaptively clusters transmission participants to balance latency and resource utilization. By intelligently forming sender groups, this mechanism effectively reduces the number

of WAN round-trips required per transmission round and collaborates with the Communicator module to suppress redundant traffic through selective filtering.

Filter. This component filters out "white data"—updates irrelevant to the receiver's current task context—thereby reducing unnecessary communication overhead in geo-distributed environments. By analyzing task dependencies and application semantics, the filter selectively transmits only data that contributes to the current transaction, minimizing bandwidth usage and alleviating the burden on downstream consistency mechanisms. Observation #2 motivates the design of GeoCoCo's filtering module, with the grouping plan providing essential conditions for identifying irrelevant data.

Communicator. The Communicator module is responsible for coordinating the final stage of transmission. It receives the optimized transmission path plan from the *Planner* and the filtered data from the *Filter* module, and accordingly controls the actual data delivery process. The Communicator integrates a consistency-preserving transmission mechanism that ensures the pruned data still meets the consistency requirements of the distributed database system. By decoupling transmission coordination and data filtering from the final delivery mechanism, this module enables efficient and reliable synchronization across geo-distributed nodes without requiring additional coordination. The hierarchical design in Communicator is jointly inspired by Observation #1 and Observation #3.

Novelties. To our knowledge, GeoCoCo is the first system that systematically optimizes all-to-all synchronization in geo-distributed transactional databases. It introduces three key techniques: (1) a **latency-aware clustering framework** for topology-adaptive grouping; (2) a **semantic filtering mechanism** that removes "white data" to reduce redundant communication; and (3) a **group-based transmission mechanism** enabling efficient aggregated synchronization across groups. Together, these techniques jointly enhance the structural, semantic, and execution efficiency of synchronization.

4.2 Latency-aware Grouping Strategy Orchestrator

In this section, we introduce the process of generating the group plan, including real-time latency monitoring and a latency-aware grouping strategy.

Real-Time Latency Monitoring. We conduct real-time monitoring of inter-node latency in the distributed system. This component dynamically produces an $N \times N$ latency matrix L, where each entry $L[i, m]$ represents the current measured latency between node i and node m. This latency matrix is updated in real-time to reflect changing WAN conditions, and it serves as the foundational input to our grouping strategy. By feeding fresh latency data into the planning algorithm, GeoCoCo ensures that grouping decisions remain adaptive to the network's state.

Design Goal. Given the latency matrix L and a desired number of groups k, our goal is to partition the N nodes into k communication groups and select one aggregator node for each group, so as to minimize the overall communication delay T across the system.

Problem Formulation. We formalize the latency-aware grouping optimization problem with the following variables and objective.

1. Decision Variables. We define binary variables $x_{i,j}$ and $y_{i,j}$ to represent grouping and aggregator selection: $x_{i,j} = 1$ if node i belongs to group j, and $y_{i,j} = 1$ if node i is the aggregator of group j. Each node is assigned to exactly one group, and each group has exactly one aggregator. We further introduce continuous auxiliary variables to capture latency costs, where l_j denotes the maximum intra-group latency of group j, and L denotes the total inter-group latency.

2. Objective Function. Based on the aforementioned variables, we define the total cost of a single synchronization round as follows:

$$T = max(l_j) + max(L) \tag{1}$$

Algorithm 1 Latency-Aware Grouping via LP Optimization

Require: Latency matrix $\mathcal{L}$ ($N \times N$), number of groups k
Ensure: Optimal group assignment and aggregator selection

1: **Initialize LP Objective**
2: Minimize total communication delay T
3: **Define Variables**
4: $x_{i,j}$: Node i belongs to group j
5: $y_{i,j}$: Node i is the aggregator of group j
6: l_j: max intra-group delay for group j
7: L: total inter-group delay; T: overall delay
8: **Add Constraints**
9: **for** each node i **do**
10: $\sum_{j=1}^{k} x_{i,j} = 1$ ▷ node must join one group
11: **end for**
12: **for** each group j **do**
13: $\sum_{i=1}^{N} y_{i,j} = 1$ ▷ one aggregator per group
14: **end for**
15: $y_{i,j} \leq x_{i,j}$ ▷ Only group members can be aggregators
16: Define $z_{i,m,j}$: node i, m in group j
17: Define w_{i,m,j_1,j_2}: cross-group pairs
18: $l_j \geq \mathcal{L}[i,m] \cdot z_{i,m,j}$
19: $L \geq \sum \mathcal{L}[i,m] \cdot w_{i,m,j_1,j_2}$
20: Ensure $T \geq \max_j l_j$ and $T \geq L$
21: **Solve and Return**
22: Use LP solver to minimize T
23: Extract $x_{i,j}$ and $y_{i,j}$ to get *group_plan*
24: **return** *group_plan*

where

$$L_j = max(L[k,j]), \forall x_{i_j} = 1 \tag{2}$$

and

$$L = max(L[u,v]), \forall y[u,u] = 1 \wedge y[v,v] = 1 \wedge u \neq v \tag{3}$$

3. Constraints. To correctly model the grouping problem, we impose a set of linear constraints that ensures a valid grouping and binds the T value to the latency measures. With the above formulation in place, we leverage a linear programming solver (more precisely, a mixed-integer linear program, given the binary variables) to determine the optimal grouping. Algorithm 1 outlines this latency-aware grouping procedure. In summary, we initialize the solver to minimize T (Lines 1–2) and define variables (Lines 3-7). Then we add all the constraints (Lines 9–20). We impose three constraints. First, each node must join exactly one group (Lines 9–11), ensuring no node is unassigned or duplicated. Second, each group must have exactly one aggregator (Lines 12–14). Third, a node may serve as a group's aggregator only if it belongs to that group (Line 14). Solving the model yields the optimal $x_{i,j}$ and $y_{i,j}$ values (Lines 22–24), specifying group membership and aggregator selection, and producing a plan that minimizes latency cost T.

This latency-aware orchestrator is invoked periodically or when network conditions change, ensuring that groups adapt to maintain low-latency communication. In practice, wide-area network

dynamics are typically episodic rather than continuous, exhibiting diurnal demand patterns, bursty anomalies, and event-driven routing changes [32, 84]. As a result, stable communication phases often last long enough to amortize the cost of grouping decisions. GeoCoCo does not rely on per-packet or per-millisecond reconfiguration; instead, it performs periodic or event-driven plan updates, avoiding excessive control overhead while still tracking meaningful network changes. To further prevent oscillation caused by transient RTT noise, GeoCoCo adopts a Re-group damping strategy: re-grouping is triggered only under sustained latency deviation (e.g., >20% over a sliding window), while unnecessary reconfigurations are suppressed. The grouping strategy is crucial for enabling hierarchical transmission (where aggregators forward data among themselves in a second tier of communication, see subsection 4.4) and for providing a structured basis on which task-level data filtering can operate efficiently (subsection 4.3). In essence, by clustering nodes intelligently, GeoCoCo reduces costly WAN round-trips and confines most traffic to low-latency local groups, substantially lowering both intra- and inter-data-center communication costs.

GeoCoCo in Larger Cluster. In practical geo-distributed database deployments, each logical node typically corresponds to a data center, and the number of such nodes is usually limited. Real-world cross-region systems commonly involve up to around 25 data center nodes [10]. This aligns well with the practical range of group numbers k considered in our model. However, to ensure scalability and generality, we also evaluate scenarios with larger node counts to explore future expansion and stress-test the planner's performance in §6.4. Moreover, considering the computational cost of LP-based planning, we conduct a theoretical analysis of the optimal group number k^* to guide the search space and improve planner efficiency under different node scales.

The Setting of Group Number. As the group size increases (i.e., as the number of nodes N grows), the computation time of LP-based grouping also increases accordingly. To optimize the overall completion time in practical systems, our goal is to determine an appropriate number of communication groups k. Although the actual completion time may also be affected by factors such as bandwidth limitations, network congestion, and dynamic path selection, communication latency typically remains the dominant performance bottleneck. Therefore, selecting an appropriate k to balance computational overhead and communication efficiency is crucial for performance optimization. To guide the design space, we adopt a refined cost model of the form of communication cost:

$$C_{\text{total}} = 2N\left(\frac{N}{k} - 1\right) + 2k(k-1) \tag{4}$$

$$k^* = \left(\frac{N^2}{2}\right)^{1/3} \tag{5}$$

which more accurately reflects the total communication load under a hierarchical all-to-all scheme. The first term represents the cumulative intra-group cost across k groups, assuming each group of size $n = \frac{N}{k}$ performs full pairwise exchange. The second term models inter-group communication cost among k aggregators. By analyzing the derivative of the model to identify its minimum, we find that the total communication cost is minimized when the number of groups satisfies Equation(5). The exact value of the optimal group count k^* varies with the cluster size N. For smaller clusters ($N \leq 25$), k^* typically falls within the range of $[N/3, N/2]$. As N increases, the ratio k^*/N decreases rapidly. Therefore, we narrow the search range around k^* with a small tolerance. This strategy enables efficient and accurate group selection across different cluster scales, and we validate its effectiveness in §6.6.

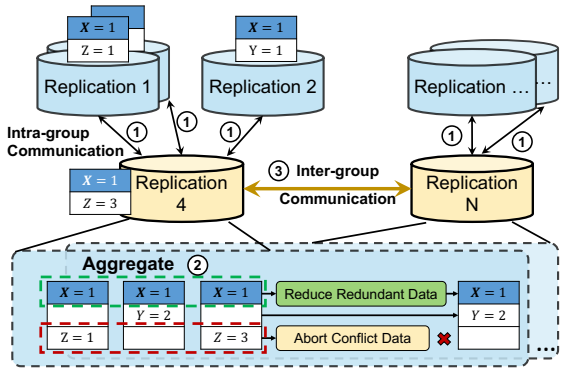


Fig. 7. The filtering process during synchronization in geo-distributed multi-master databases (e.g., GeoGauss synchronization).

4.3 Task-Preserved Data Filtering

GeoCoCo's filtering mechanism is integrated into its hierarchical communication process, using contextual metadata about each update to decide its relevance. In a multi-master database scenario (such as the GeoGauss [36]), each replica executes transactions locally using OCC [36, 41, 76] and periodically exchanges batched updates with other replicas. GeoCoCo leverages metadata inherent in this process—e.g., transaction identifiers, read/write sets, timestamps, and update payloads—to perform rule-based filtering that prunes out irrelevant data before it leaves the local region.

Definition and Impact of White Data. In geo-distributed systems, *white data* refers to updates that are transmitted but ultimately discarded during synchronization, contributing nothing to the final state of the receiving replica. Filtering white data—defined earlier—requires balancing trade-offs between transmission reduction, correctness, and computational overhead. **Redundant content**: semantically identical updates repeatedly sent; **Conflicting or stale updates**: caused by asynchronous execution or validation failures; **Null or sparse data**: updates with no meaningful payload.

While white data consumes WAN bandwidth and introduces queuing overhead, it has no effect on system correctness or transactional integrity. Eliminating such data reduces communication cost and improves latency. This concept, generalized from our observations in Section 3, forms the foundation of GeoCoCo's filtering design. By identifying and removing white data early, GeoCoCo achieves lossless filtering: the system's externally visible behavior (e.g., commit results or model convergence) remains unchanged, but with significantly reduced WAN traffic.

Design of the Task-Preserved Filtering Mechanism. We illustrate the white data filtering process using a multi-master database as an example. In such systems, write operations are typically handled via OCC, where transactions execute locally and periodically exchange batched updates across replicas (e.g., GeoGauss). The filtering workflow is depicted in Figure 7, assuming group members and leaders have already been determined: (1) Intra-group transmission. Each member node sends local information (e.g., read/write sets) to its designated leader. If a group consists only of a leader, it proceeds directly to the next step. (2) Information aggregation. The leader aggregates metadata from its members and applies rule-based screening. In database settings, this first includes conventional conflict detection based on read/write-set validation, where conflicting transactions are aborted before cross-group propagation. In addition, GeoCoCo identifies white data—updates that do not change the visible database state—and filters them out to avoid unnecessary WAN transmission. The filtering logic performs constant-time version-vector and hash checks over local

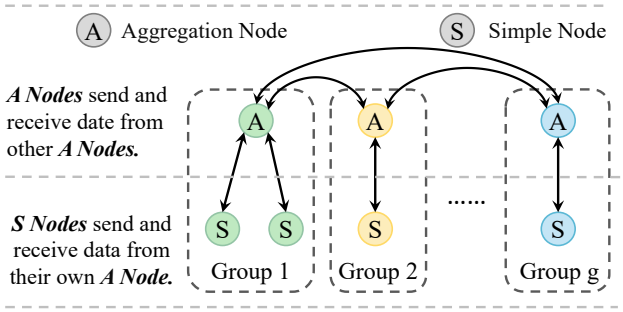


Fig. 8. Hierarchical transmission.

metadata at the aggregation node, avoiding global coordination and ensuring $O(1)$ per-update overhead. Because the checks are entirely local and metadata footprint is bounded, the filtering cost avoids blow-up as the system scales, ensuring it does not become a bottleneck even in large clusters. (3) Inter-group commit and broadcast. The leader commits the filtered and aggregated results across groups and disseminates the final data to all group members.

In summary, GeoCoCo's filtering framework integrates two complementary strategies. First, it performs hierarchical path optimization by leveraging the Triangle Inequality through multi-hop grouping, enabling speculative yet low-latency synchronization across regions. Second, it applies content-aware filtering that proactively detects and eliminates conflicting or irrelevant updates—such as aborted transactions and white data—early in the transmission process. This ensures that only valid, task-preserving operations are propagated through the hierarchy, significantly reducing bandwidth consumption and improving end-to-end latency.

4.4 Consistency-Guaranteed Transmission

Hierarchical Transmission Flow. GeoCoCo introduces a hierarchical, consistency-preserving transmission mechanism that optimizes WAN data flow without altering the underlying consistency model. Nodes are dynamically partitioned into latency-aware groups via an LP-based planner, and one *Aggregation node (A)* is selected per group (Figure 8). Each *Simple node (S)* sends updates only to its local A, which aggregates intra-group data (with filtering; see §4.3) and forwards consolidated results to other aggregation nodes. After inter-group exchange, each A disseminates received updates back to its group. This design clusters low-latency peers and minimizes long-haul WAN edges, significantly reducing synchronization cost while preserving correctness. Aggregators adapt across rounds based on network conditions, ensuring efficiency under dynamic WAN behavior. Importantly, S nodes never communicate cross-group in a round—A nodes orchestrate all inter-group transfer—allowing GeoCoCo to reshape communication paths while fully retaining the system's original consistency semantics.

Correctness under Message Reordering, Delayed Updates, and Network Partition. GeoCoCo inherits the correctness guarantees of GeoGauss's epoch-aware delta-CRDT replication model, which provides strong convergence even under unreliable communication [36, 64]. **First**, the CRDT merge function $\oplus$ satisfies three key algebraic properties: *Commutativity*, *Associativity*, and *Idempotence* (ACI). Let S be the current system state and $\mathcal{U} = \{u_1, u_2, \ldots, u_m\}$ the set of updates produced in an epoch. If all updates are applied exactly once in any order, the merged state is

$$S' = S \oplus u_1 \oplus u_2 \oplus \cdots \oplus u_m.$$

More generally, consider any permutation π of the update set and any multiplicity vector $\mathbf{k} = (k_1, k_2, \ldots, k_m)$, where k_i indicates how many times u_i is delivered (including duplicates). The

resulting state after arbitrary reordering and duplication is

$$S' = S \oplus \underbrace{u_{\pi(1)} \oplus \cdots \oplus u_{\pi(1)}}_{k_{\pi(1)} \text{ times}} \oplus \cdots \oplus \underbrace{u_{\pi(m)} \oplus \cdots \oplus u_{\pi(m)}}_{k_{\pi(m)} \text{ times}}.$$

Because $\oplus$ is **commutative and associative**, the order of updates does not affect the result; and because it is **idempotent** ($x \oplus x = x$), repeated applications of the same update have no effect. Thus, for any π and $\mathbf{k}$:

$$S \oplus \bigoplus_{i=1}^{m} \bigoplus_{j=1}^{k_i} u_i = S \oplus u_1 \oplus u_2 \oplus \cdots \oplus u_m.$$

Intuitively, $\oplus$ behaves like a set union: duplicates collapse to one, and the final result depends only on the set of updates, not their order or number of deliveries. This algebraic property is the foundation for GeoCoCo's correctness under message reordering, duplication, and delayed delivery. **Second**, GeoCoCo strictly enforces *epoch boundaries*, preventing cross-round reordering. Delayed updates that miss epoch e are incorporated into epoch $e + 1$, delaying visibility but not affecting correctness or convergence. **Third**, under message delay, the additional visibility latency is bounded. If τ denotes the retransmission timeout and Δ_{WAN} denotes the maximum WAN propagation delay, the **worst-case additional visibility delay** is:

$$T_{\text{delay}}^{\max} \leq \tau + \Delta_{\text{WAN}},$$

after which the delayed update is guaranteed to be merged in the subsequent epoch. **Finally**, in the presence of *network partitions*, GeoCoCo preserves *safety* but may temporarily reduce liveness. Because GeoGauss generates an epoch snapshot only after receiving and merging all updates from the epoch, cross-partition updates are buffered and not committed until the partition heals. This prevents divergent states: after reconciliation, all replicas deterministically converge to the same state as if no partition had occurred. Thus, network partitions may increase visibility latency but cannot violate correctness. **In summary**, GeoCoCo preserves end-to-end application-level consistency under arbitrary message reordering, delivery delays, and network partitions. Reordering and delay are fully absorbed by CRDT algebraic properties and epoch isolation, while partitions affect only progress, not correctness.

Transmission Round Guarantee. We guarantee that the number of communication rounds required for synchronization under hierarchical transmission is strictly less than that of the baseline approach. Formally, for a grouping plan with N nodes and k groups, each node sends messages to and receives messages from all other $N - 1$ nodes, resulting in transmissions per node:

$$C_{baseline} = 2 \times (N - 1) \tag{6}$$

In our grouping-based GeoCoCo scheme, each **Simple Node** communicates only with its aggregator, sending and receiving one message, totaling 2 transmissions. Each **Aggregation Node** receives and sends messages to group members, and also communicates with the other $k - 1$ aggregation nodes. In an N-node environment, the number of group members cannot exceed $N - k$. Therefore, the total communication rounds is bounded by:

$$C_{GeoCoCo} \leq 2 \times [(N - k) + (k - 1)] = 2 \times (N - 1) \tag{7}$$

Combining Equation (6) and Equation (7), we obtain $C_{GeoCoCo} \leq C_{baseline}$, which guarantees that even in the worst case, all nodes in GeoCoCo require no more communication rounds than the baseline.

Aggregator Failover and Fault Tolerance. Aggregator failures do not compromise correctness. If a relay node fails or becomes unreachable, group members immediately fall back to direct transmission, and the *Grouping Strategy Orchestrator* reconfigures groups in the next round to restore optimized paths. If a regular node fails, we cannot send or receive data anyway, so we simply skip to the next round and reconfigure the grouping. Failures are visible to the upper layer, but whether they affect application semantics depends on the application itself; GeoCoCo guarantees that collective transmission among surviving nodes remains correct and consistent. CRDT idempotence ensures that retransmissions or duplicates during failover do not cause inconsistency, only potential extra communication or latency. These mechanisms are transparent to the database: commit and merge proceed normally, with fewer redundant messages. GeoCoCo enforces epoch boundaries to prevent cross-round reordering, keeping each epoch self-contained for validation (e.g., commit or model convergence). Group or aggregator changes across rounds do not affect prior results or future merges.

Reducing Cross-Region Bandwidth Overhead. By performing semantic-aware pre-filtering and in-group aggregation, GeoCoCo transmits only essential and non-redundant data across regions. This pre-transmission pruning strategy significantly reduces WAN-level traffic and alleviates communication bottlenecks under limited bandwidth conditions. Minimizing Latency along Critical Paths GeoCoCo adopts a hierarchical communication architecture that confines most synchronization operations within low-latency local domains, while limiting inter-region exchange to a small subset of data. It further leverages Triangle Inequality Violation-aware path optimization to circumvent suboptimal links, effectively shortening the end-to-end delay along synchronization-critical paths. Enhancing global transaction throughput by combining intelligent grouping with redundancy elimination mechanisms, GeoCoCo alleviates the classic "slowest-link bottleneck" problem inherent in fully-connected communication schemes. This design improves the overall transaction processing efficiency in geo-distributed deployments without incurring additional synchronization overhead or compromising consistency guarantees.

5 IMPLEMENTATION

We implement the protocol components of GeoCoCo with C++ and Python, and integrate them into GeoGauss and CRDB to build our prototype system. This section presents the details.

Delay Monitoring. In GeoCoCo, each replica launches a lightweight background thread during system initialization, which periodically probes link RTTs and reports them to a designated coordinator (the *Monitor*). The Monitor role can be assigned to an existing worker or to a dedicated node, depending on deployment constraints. The probing runs asynchronously and does not block normal transaction execution. Since WAN conditions tend to shift in *episodic patterns* rather than continuous fluctuations, GeoCoCo only updates transmission plans when significant and persistent changes are detected, rather than reacting to transient jitter. This design avoids frequent plan churn while maintaining responsiveness to meaningful network events. Additionally, to remain scalable when the cluster grows to thousands of nodes—where maintaining a full $N \times N$ latency matrix becomes increasingly costly, we employ network coordinate techniques [11, 44] (e.g., Vivaldi [18]) to approximate inter-node latencies. To ensure estimation accuracy, we integrate a verification mechanism that periodically samples and corrects deviations between predicted and measured RTTs. This hybrid design effectively reduces monitoring traffic and computation overhead while maintaining latency fidelity at scale.

LP Solver. We utilize Gurobi [28] to solve our linear programming (LP) problems. Gurobi is a commercial optimization solver developed by Gurobi Optimization, LLC, that has been extensively utilized in both academia and industry [53, 57, 83]. It provides state-of-the-art algorithms for solving

linear programming, mixed-integer programming (MIP), quadratic programming (QP), and other mathematical optimization problems. Gurobi employs a *concurrent optimization strategy*, running both *dual simplex* and *barrier methods* in parallel for linear programming. In theory, for convex LPs solved with barrier methods, the complexity of finding an ε-approximate solution is bounded by $O(\sqrt{M} \cdot \ln(V/\varepsilon))$ Newton iterations (where M is the condition number, V the initial duality gap, and ε the accuracy tolerance), where each requires $O(n^3)$ operations per step (n = problem dimension) [6, 29]. In our experimental environment (Aliyun g7ne instance), Gurobi can solve a 15-node LP-based grouping problem in less than 10 ms. This ultra-low computational delay enables us to generate latency-aware transmission topologies in real time, making our dynamic grouping strategy practically deployable.

K-Center–Based Scalable Planner. To further improve scalability when the number of nodes grows large, we replace the LP-based grouping in GeoCoCo with a lightweight K-center–based heuristic. This algorithm aims to minimize the maximum intra-group latency by iteratively selecting the most distant node (in terms of network delay) as a new group center, and assigning all remaining nodes to their nearest center. By doing so, it approximates the optimal latency-aware grouping with only $O(N \times k)$ complexity, while guaranteeing that the resulting maximum intra-group delay is within twice of the theoretical optimum. Compared with the LP solver, the K-center method avoids quadratic complexity and enables near–real-time group planning even when the cluster size scales to hundreds or thousands of nodes. This heuristic thus preserves GeoCoCo's latency-awareness and correctness, while substantially reducing the computational cost of planning.

Overlay-based Implementation. GeoCoCo exploits Triangle Inequality Violations without modifying Border Gateway Protocol (BGP) or any low-level routing. Instead, it realizes indirect paths purely at the application layer through lightweight user-space relays (e.g., TCP port-forwarding or gRPC proxies) deployed on ordinary cloud VMs or existing replicas. Relay nodes are selected dynamically based on online RTT measurements and health checks, and GeoCoCo automatically falls back to the direct path if a relay fails or does not provide sufficient latency gain. This overlay-based deployment requires no privileged access and can be easily implemented in public clouds by simply enabling inter-region ports. All experiments in this paper use this deployment strategy in practice.

Transactional Isolation. To ensure system correctness in the presence of concurrent updates, we enforce a transactional isolation mechanism when modifying transmission plans. Specifically, at the beginning of each synchronization cycle, the system creates a snapshot of the current transmission plan. All transmission operations within that cycle are then executed strictly based on this immutable snapshot, regardless of whether new plans are generated during execution. This design effectively prevents interference between the planning and execution phases, ensuring that ongoing transmissions are not affected by configuration updates. As a result, the system maintains consistent behavior and predictable semantics even under dynamic reconfiguration. In other words, even if the system undergoes configuration changes at runtime—such as updating transmission plans, switching aggregation nodes, or adjusting scheduling strategies—the actual execution remains correct, ordered, and consistent with expected semantics.

Collective Communication. We integrate GeoCoCo as an intermediate communication layer between the database commit protocol and the transport stack. In GeoGauss, this is achieved by replacing direct point-to-point `ZeroMQ` calls with GeoCoCo's collective communication APIs (`AllToAll`, `Broadcast`, `AllReduce`, `Gather`, `AllGather`). By default, GeoGauss uses low-level `ZeroMQ` primitives (e.g., `zmq_send`, `zmq_recv`) [15, 31, 36], tightly coupling database logic with transport behavior and limiting WAN adaptability. GeoCoCo instead provides a high-level, intent-driven interface: the database specifies the communication pattern, and GeoCoCo automatically selects optimal WAN routing and execution via topology-aware grouping, relay scheduling, and

filtering. This design preserves the architecture of GeoGauss and only replaces the network invocation path. Collective operations (e.g., epoch-based update dissemination) become a single `geococo.all_to_all()` call, enabling automatic scheduling and WAN optimization. As a result, GeoCoCo improves WAN performance and can be easily ported to other geo-distributed databases via its general-purpose API.

Extensions – Integration with CockroachDB. To demonstrate the generality of GeoCoCo, we discuss its applicability to Raft-based geo-replicated databases such as CockroachDB. Since GeoCoCo operates at the communication layer, it can be seamlessly integrated with Raft's transport subsystem without modifying the consensus protocol itself. Specifically, GeoCoCo can hook into *RaftTransport* to intercept replication messages (e.g., *AppendEntries, Heartbeats*, and *Snapshots*). The Planner and Communicator modules enable latency-aware grouping and hierarchical forwarding of these messages, while the Filter module can be applied to replication streams to remove redundant updates before WAN transmission. This design reduces cross-region latency and bandwidth consumption while preserving Raft's quorum semantics.

6 EVALUATION

6.1 Setup

Baselines. To contextualize our gains, we evaluate GeoCoCo against both native database systems and representative synchronization optimization techniques. Specifically, we reference the following three mainstream directions:

- **GeoGauss** [36]: We use the default GeoGauss system as the baseline, which performs full all-to-all synchronization across replicas without grouping or filtering.
- **CockroachDB** [43, 71]: We also compare against unmodified CockroachDB, which relies on its native Raft replication pipeline and direct WAN message delivery.
- **Other Representative Techniques:** (i) congestion control protocols such as **BBR** [9], (ii) compression applied to replication streams such as **zlib** [24, 55], and (iii) scheduling and locality-aware execution strategies exemplified by *Calvin* and **SLOG** [60].
- **Grouping Strategies:** To evaluate our latency-aware grouping mechanism, we compare the cost and per-round performance of our LP-based strategy against several alternatives, including hierarchical agglomerative clustering [85], KMeans with 2 and 3 clusters [54, 58], random grouping, and a no-grouping baseline.

Workload. We use a widely adopted online transaction processing (OLTP) benchmark in database systems, Transaction Processing Performance Council benchmark-C (TPC-C) [74] and Yahoo! Cloud Serving Benchmark(YCSB) [16].

• **TPC-C:** To evaluate the system under varying transactional patterns and communication demands, we design four representative TPCC-style workloads—TPCC-A, TPCC-B, TPCC-C, and TPCC-D—corresponding to write-intensive, read-intensive, balanced, and real-time scenarios, respectively. While the official TPC-C benchmark defines a fixed ratio of five transaction types (NewOrder, Payment, OrderStatus, Delivery, and StockLevel), it is a common and accepted practice in prior work (e.g., Calvin [72], GeoGauss [36]) to customize this distribution to reflect real-world workloads or highlight system behaviors. (1) TPCC-A (Write-Intensive): Dominated by NewOrder and Payment (>90%), simulating peak write-heavy scenarios. (2) TPCC-B (Read-Intensive): Skewed toward OrderStatus and StockLevel, modeling frequent backend queries. (3) TPCC-C (Balanced): Even distribution of all five transactions, serving as a standard OLTP baseline. (4) TPCC-D (Real-Time): Emphasizes OrderStatus for responsive user interactions, with moderate write components. In addition, we follow standard TPC-C performance metrics to measure throughput: **tpmC** (transactions per minute C) represents the number of committed `NewOrder` transactions per minute, which

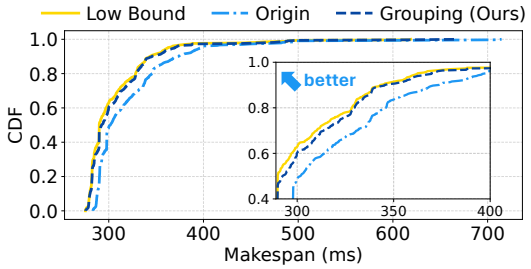


Fig. 9. CDF Distribution of Single-Round Makespan.

is the primary official performance indicator of TPC-C and reflects core write throughput. **tpmTotal** represents the total number of transactions of all five types processed per minute, providing a broader view of the system's overall transactional capacity under mixed workloads.

• **YCSB:** YCSB is one of the most popular key-value transaction benchmarks, encompassing insert, delete, and update operations on key-value pairs. We synthesized datasets with varying conflict rates using YCSB to validate GeoCoCo's performance under different contention levels. We adjust the conflict rate by tuning the θ parameter in the Zipfian distribution, which controls the key access skew. Such configurations are widely used in OLTP system evaluations and reflect typical access patterns in geo-distributed deployments.

Cluster Setup. We evaluate GeoCoCo using both emulated and real-world geo-distributed environments.

• **Trace-driven Simulation:** We emulate WAN conditions using local servers and the TC *netem* module [22], guided by real-time link delay traces from Amazon AWS [3]. To capture dynamic behaviors such as delay spikes and jitter, we apply Piecewise Cubic Hermite Interpolating Polynomial fitting [23] to AWS latency data, preserving both mean and distributional properties. This fitting process produces over 10,000 synthetic 10×10 delay matrices, which are replayed via TC [22] to simulate time-varying inter-node delays. Real-time monitoring ensures emulation accuracy. This setup enables faithful simulation of WAN environments for evaluating the performance and robustness of geo-distributed databases.

• **Real-world Testbed Deployment:** We further deploy a 5-node geo-distributed database across multiple geographic regions to evaluate performance under real-world production conditions. The deployment includes Kalgan, Hohhot, and Hong Kong. Each server is provisioned as an Alibaba Cloud c5 instance, configured with 24 vCPU 48.0 GiB of RAM, and running CentOS 7.6 Server (64-bit) [1].

• **Bandwidth Utilization Measurement:** To estimate the actual WAN bandwidth consumption during system execution, we collect network interface statistics via the Linux /proc [52] filesystem. This kernel-level mechanism enables direct access to per-interface traffic counters, including outbound data sent through the NIC. All measurements are recorded at the NIC level, ensuring an accurate reflection of the actual WAN egress traffic per node.

6.2 Micro Benchmark

Theoretical Performance Metric. We define the completion time of a single-round all-to-all transmission *Makespan* as the system's theoretical performance metric. All-to-all is one of the most latency-sensitive and bandwidth-intensive collective communication patterns in geo-distributed environments. Due to network heterogeneity, even if most node pairs complete their transmissions quickly, a few high-latency links can dominate the overall round time, leading to a bottleneck effect.

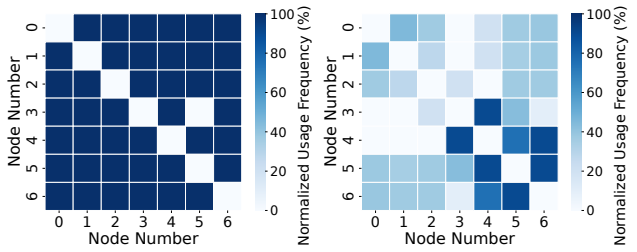


Fig. 10. (a) Heatmap of Communication Frequency Without Grouping (Baseline). (b) Heatmap of Communication Frequency with Transmission Grouping Using GeoCoCo.

In such systems, the global progress is often gated by the slowest transmission pair, making the Makespan a natural and critical indicator of system efficiency. Furthermore, in many distributed systems (e.g., databases, machine learning frameworks), communication rounds serve as consistency or synchronization boundaries. Optimizing across rounds risks breaking correctness guarantees. Therefore, we focus on *per-round performance* and avoid cross-round reordering or pipelining. Formally, we aim to minimize the worst-case transmission latency in a round:

$$\min T_{\text{Makespan}} = \min \left(\max_{i,j \in \{1,2,\ldots,N\}} T_{i,j} \right) \tag{8}$$

This objective reflects our design principle: achieving global efficiency by eliminating the longest delays that constrain system progress.

Single-Round Makespan Reduction. Figure 9 shows the CDF of single-round all-to-all makespan under three schemes: *Origin* (default), *Grouping (Ours)*, and *Low Bound* (theoretical optimum). The x-axis denotes makespan (ms), and the y-axis shows the CDF across runs. Our grouping consistently outperforms the baseline: the CDF curve shifts left, indicating more rounds finishing at lower latency. At the 90th percentile, our approach reduces makespan by over 100 ms. The inset (300–400 ms range) shows a tighter spread and lower tail latency, moving closer to the theoretical bound. These gains stem from latency-aware LP grouping, which forms compact groups to bypass slow links and reduce long-path dependencies. While *Origin* suffers from scattered worst-case latency due to flat all-to-all communication, GeoCoCo mitigates bottlenecks from outlier nodes, reducing variance and approaching the optimal frontier. Overall, GeoCoCo significantly accelerates single-round synchronization, reduces variance, and approaches the theoretical minimum, validating our objective in Equation (8).

Cutting Communication Rounds via Hierarchical Design. We conducted a communication frequency profiling experiment across 400 rounds of all-to-all transmission over 7 nodes, under a trace-driven simulated WAN latency environment. Figure 10 shows the normalized heatmap of point-to-point communication frequency. Compared to the baseline scheme (left, (a)), GeoCoCo (right, (b)) significantly reduces the number of communication rounds for most nodes. Post-hoc inspection reveals that communication is concentrated on a few aggregation nodes (e.g., nodes 3, 4, 5, and 6). Although these aggregation nodes handle more relay traffic, their total communication count remains below that of any single node in the baseline.

6.3 Macro Benchmark

End-to-end Benefits on GeoGauss. To evaluate the end-to-end benefits of GeoCoCo in a real database system, we integrate it into GeoGauss, a full-replica multi-master DBMS, and run TPC-C workloads [74]. Following the deployment in §6.1, we deploy GeoGauss across 5 geo-distributed data centers (two in Kalgan, two in Hohhot, and one in Hong Kong). We compare (1) GeoGauss

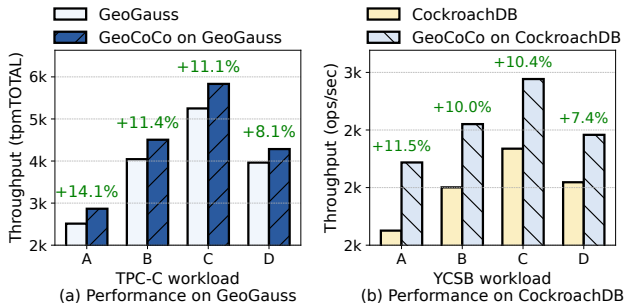


Fig. 11. Throughput improvement of GeoCoCo (a) over GeoGauss under TPC-C workload A-D and (b) over CockroachDB under YCSB workloads A–D.

with GeoCoCo optimizations and (2) the default system with native messaging. To stress inter-node coordination and highlight communication effects, we run 100 TPC-C warehouses, ensuring frequent cross-site updates. Both systems run to a steady state, after which we collect metrics over a sustained sampling window for statistical reliability. Figure 11(a) presents the average transaction throughput (tpmTOTAL) under four representative TPC-C workloads. We make the following observations: (1) Consistent Performance Gains: Across all four workloads—TPCC-A,B,C and D)—the system integrated with GeoCoCo consistently outperforms the baseline GeoGauss. (2) Significant Improvement in Write-Intensive Scenarios: In the TPCC-A workload, which involves frequent cross-node write operations, GeoCoCo achieves the highest improvement of 14.1%, indicating its ability to mitigate communication bottlenecks in high-contention settings. (3) Broad Applicability Across Workload Types: Even in read-intensive (TPCC-B), balanced (TPCC-C), and real-time update (TPCC-D) workloads, GeoCoCo delivers non-trivial throughput improvements ranging from 8.1% to 11.4%, highlighting its general effectiveness across diverse OLTP scenarios.

End-to-end Benefit on CockroachDB(CRDB). GeoCoCo is integrated into CockroachDB's Raft-based replication stack by hooking into the RaftTransport layer to intercept replication messages, without modifying the Raft consensus protocol; this non-intrusive design preserves standard quorum semantics. We evaluated the integrated system under YCSB workloads A–D in a real-world wide-area testbed arranged in a full-mesh topology. Figure 11(b) shows the throughput for these workloads, comparing baseline CockroachDB with CockroachDB+GeoCoCo. GeoCoCo improves throughput by up to 11.5% to the baseline (reduces p99 latency by 14.49% at the same time). These results demonstrate that GeoCoCo's latency-aware coordination remains effective in practical Raft-based systems such as CockroachDB.

6.4 GeoCoCo Deep Dive

Comparison of Different Grouping Strategies. We evaluate the cost and per-round performance of our Linear Programming (LP)-based grouping scheme against several alternatives, including hierarchical agglomerative clustering [85], KMeans with 2 and 3 clusters [54, 58], random grouping, and a no-grouping baseline. To assess the contribution of Triangle Inequality Violations, we additionally evaluate a grouping variant with TIV support disabled, enabling an isolated measurement of the benefit brought by TIV-aware grouping(in Figure 12's GeoCoCo-TIV). The x-axis represents the average computation time per grouping, while the y-axis shows the makespan of a single transmission round as shown in Figure 12. The contour lines indicate the trade-off boundary where the grouping plan is updated every 10 rounds. We highlight 4 key observations: **First**, GeoCoCo's LP-based method consistently outperforms all baselines under both 12-node and 15-node settings, demonstrating strong generality and stability across scales. **Second**, although GeoCoCo incurs

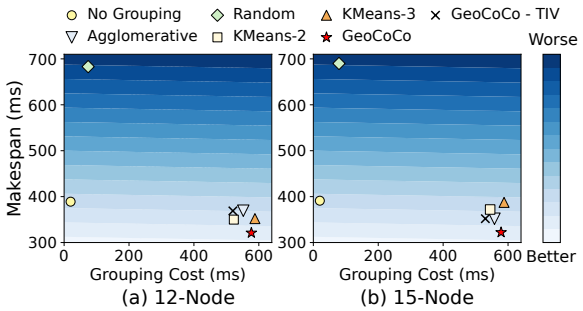


Fig. 12. Grouping cost vs. communication efficiency under 12-node (a) and 15-node (b) settings. Lighter areas with an arrow indicate better trade-offs.

higher computational overhead as the cluster size grows, its performance gains scale accordingly. For example, with 12 nodes, GeoCoCo achieves a 16.46% improvement in makespan, which is sufficient to offset the cost within a single synchronization round. When the number of nodes increases to 15, the solving time rises (comparable to KMeans and hierarchical agglomerative clustering), but GeoCoCo yields a 17.63% improvement—significantly higher than the best baseline (11.0% for hierarchical clustering). This makes the additional computation cost easily amortized within just a few rounds, and because grouping runs asynchronously, the overhead does not block ongoing execution and is fully offset after only a small number of synchronization rounds. **Third**, Enabling TIV grouping (Full GeoCoCo) reduces makespan by 7.62%–12.44% beyond grouping alone, demonstrating that TIV exploitation offers an independent and additive benefit. **Last**, GeoCoCo consistently provides performance benefits. This is because it avoids frequent regrouping and only updates plans when stable, ensuring sustained efficiency over time.

Cost and Benefit in Larger Cluster. We systematically evaluate the scalability of GeoCoCo through trace-driven simulation experiments with node counts ranging from 5 to 50. The 50-node deployment represents a geo-distributed setup spanning multiple continents and regions, exceeding the largest data-center-scale configurations observed in existing geo-distributed systems—25 nodes in PigPaxos [10] and 48 nodes across 3 AZs in CRDB [71]. Figure 13 compares GeoCoCo with the original non-hierarchical transmission scheme. The Cost curve (yellow circles) shows the one-time overhead of computing the hierarchical plan, while the Benefit curve (blue stars) shows the cumulative reduction in synchronization time over 1000 rounds. Each round is triggered every 10,ms, consistent with the default GeoGauss setting. Although planning overhead increases with cluster size, the cumulative benefit rapidly exceeds the cost. From 5 to 50 nodes, the planner's cost accounts for only 6.65% to 7.07%. This is enabled by our guided group search strategy, which reduces planning complexity by about an order of magnitude relative to exhaustive LP-based search across all group counts in $[2, N-1]$. In addition, planning occurs before data transmission and can be offloaded to an idle node, avoiding interference with runtime execution. Overall, GeoCoCo scales efficiently to large clusters, providing substantial synchronization gains at minimal amortized planning cost.

Bandwidth Efficiency and Filtering Overhead. To assess the impact of aggregation on bandwidth utilization, we conduct controlled experiments using the YCSB benchmark with a total of 1,000,000 operations. As shown in Figure 14, the baseline (gray bar) represents no filtering, while the remaining bars correspond to conflict ratios of 5%–40%. The y-axis measures the total network traffic across all nodes in the system (in MB), which reflects the overall bandwidth consumption. The total network traffic across all nodes decreases consistently as the conflict ratio increases—by 8.7%, 27.2%, 32.2%, 35.7%, and 40.3%, respectively. This monotonic reduction confirms that higher contention

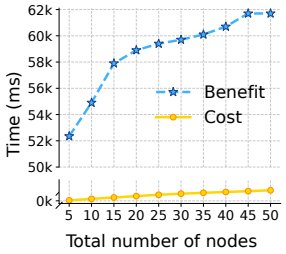


Fig. 13. Comparison of grouping cost and cumulative benefit of GeoCoCo under increasing cluster sizes.

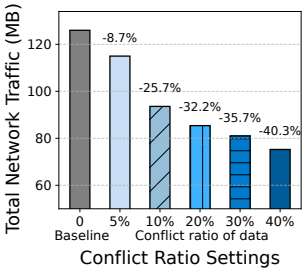


Fig. 14. Bandwidth Reduction under Conflict-Aware Aggregation.

Table 1. Filtering overhead and WAN savings across conflict ratios (YCSB).

Conflict Ratio	Config	CPU↑	p99 (ms)	WAN↓
0%	Without Filtering	–	–	–
	With Filtering	+0.97%	+0.8	~0%
30%	Without Filtering	–	–	–
	With Filtering	+2.41%	+8.4	35.7%
40%	Without Filtering	–	–	–
	With Filtering	+2.76%	+13.22	40.3%

introduces more 'white' data (this trend aligns with Observation #2), which filtering effectively eliminates to reduce WAN transmission costs. We further tested an extreme conflict-free case (randomized primary key reads). In this setting, the filtering phase is effectively bypassed, and the system's performance remains within 1% of the grouping-only baseline. This confirms that the filtering mechanism is lightweight and imposes negligible cost when no redundant writes exist. To quantify the runtime cost of white-data identification under normal workloads, we profile CPU and tail latency with filtering enabled and disabled. The detection path performs O(1) in-memory version checks and conflict hints with no additional coordination. Table 1 illustrates that across YCSB workloads, filtering incurs <2.76% CPU and < 9.22 ms change in p99 latency (excluding the benefits of grouping), while reducing cross-region traffic by up to 40.3%. These results confirm that filtering is lightweight and non-intrusive, and does not form a bottleneck even under high load.

Cost of Delay Monitoring. We measure the runtime overhead of delay monitoring under different cluster scales. In our standard configuration, even with per-epoch high-frequency probing on a 50-node full mesh, the total monitoring traffic is only about 5.2 MB/s (0.1 MB/s per node) with 0.7–1.1% CPU utilization, indicating negligible runtime impact at moderate scales. To further ensure scalability at very large scales, we employ a *Vivaldi-based network coordinate system* (NCS) to estimate inter-node latencies instead of performing full pairwise measurements. Under a 1,024-node setting using real-world latency traces, the NCS-based approach reduces detection overhead by 96.4% while maintaining estimation accuracy within 18% of direct probing, demonstrating its practicality for large-scale deployments.

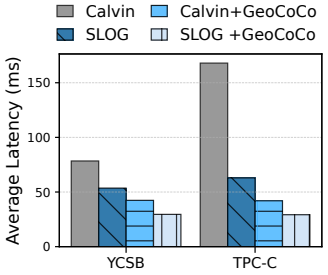


Fig. 15. Combining GeoCoCo with Calvin and SLOG under YCSB and TPC-C.

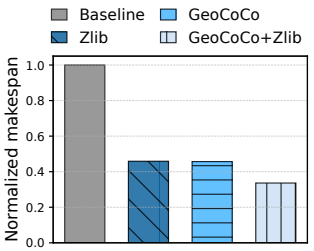


Fig. 16. Makespan under compression baseline (Zlib) and GeoCoCo (normalized).

6.5 Combining with other work

Many studies [9, 24, 55, 60, 72] accelerate geo-distributed databases by optimizing transaction scheduling, improving network efficiency, or reducing data volume via compression. GeoCoCo targets synchronization efficiency, orthogonal to these directions. We further evaluate its performance when combined with such optimizations.

Combining with Other Transaction and Synchronization Scheduling Protocols. We evaluate GeoCoCo together with two representative deterministic and locality-aware **execution protocols**, Calvin [72] and SLOG [60]. Both systems deterministically coordinate cross-partition transactions and employ execution scheduling to mitigate wide-area delays. We integrate GeoCoCo into their communication layer without modifying their transaction ordering logic or guarantees. As shown in Figure 15, GeoCoCo consistently lowers end-to-end latency on both YCSB and TPC-C workloads. When integrated with Calvin, GeoCoCo reduces average latency by 45.95%, and by 53.53% when integrated with SLOG. This result indicates that GeoCoCo can achieve performance benefits under different execution protocols.

Combining with Compression Techniques. Compression is a widely used technique for reducing data volume. We select zlib [24, 55], one of the most popular compression libraries, to evaluate the performance of integrating compression with GeoCoCo. We conducted experiments on the YCSB workload, where the data transmission block size was set to 4 MB. Figure 16 reports the normalized makespan of one synchronization round across four configurations: Baseline, zlib, GeoCoCo, and GeoCoCo+zlib. Compression alone reduces the makespan by 54.13% compared to the baseline by shrinking data volume, while GeoCoCo achieves a larger reduction through WAN-aware path planning and semantic filtering. Combining zlib with GeoCoCo yields the lowest makespan, which is 33.62% of the baseline. The results show that byte-level compression and GeoCoCo target complementary dimensions and can be effectively stacked.

Combining with Advanced Network Protocols. A variety of network transmission protocols have been developed to address the challenges of network loss and jitter in wide-area settings. BBR [9] is a representative work, which estimates the bottleneck bandwidth and round-trip propagation delay to pace traffic at an optimal rate, achieving high throughput while maintaining low latency. We evaluate the performance of combining GeoCoCo with BBR under different network conditions. In practical, we extend the trace-driven setup from §6.1 with controlled network impairments using Linux `tc netem`. We inject (i) packet loss at 1% and 5%, and (ii) RTT inflation of +30 ms and +50 ms to emulate WAN jitter while preserving the original latency distribution. We measure throughput, p99 latency, and single-round makespan. As shown in Figure 17, we summarize the results as follows. (1) Under 1% and 5% packet loss, GeoCoCo sustains higher

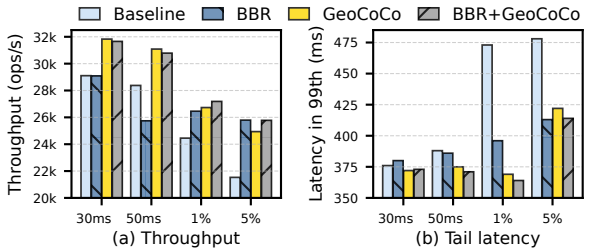


Fig. 17. **WAN loss and jitter robustness with BBR comparison.** (a) shows throughput under packet loss (1%, 5%) and RTT jitter (+30 ms, +50 ms) across 4 configurations. (b) shows the corresponding p99 latency.

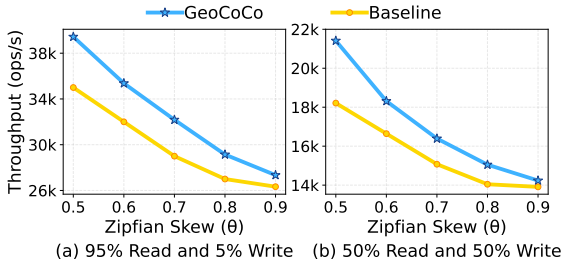


Fig. 18. Impact of Access Skew on Throughput Under Different Read/Write Ratios.

throughput (by 9.3%–15.8%) and reduces p99 latency by 56–104,ms (up to 22%) relative to the Baseline, demonstrating effective handling of loss-induced retransmissions and WAN uncertainties. (2) With +30,ms and +50,ms RTT inflation, GeoCoCo consistently lowers tail latency (by 4–13,ms) and improves throughput by 9.3%–9.6%, indicating strong robustness to time-varying WAN delays. This indicates that GeoCoCo provides robustness complementary to transport-layer congestion control, continuing to deliver performance benefits even in lossy and jitter-heavy WAN environments.

6.6 Discussion

Impact of Access Skew (Zipfian). We vary the Zipfian skew parameter θ from 0.5 to 0.9 to emulate increasingly hotspot-dominated workloads. Figure 18 reports results under both read-dominant (95% read) and balanced (50% read) settings. Under the 95/5 workload, GeoCoCo improves throughput over the baseline by 12.7%, 10.5%, 10.9%, and 7.9% at θ=0.5, 0.6, 0.7, and 0.8, respectively, and maintains similar performance at extreme skew (θ=0.9; 27.3k vs. 26.3k ops/s). Under the 50/50 workload, the benefit persists: GeoCoCo delivers 17.6%, 10.0%, 8.7%, and 7.2% higher throughput at θ=0.5–0.8, and remains slightly ahead even at θ=0.9 (14.2k vs. 13.9k ops/s). These results confirm that GeoCoCo sustains clear gains across skew levels and read/write ratios, particularly in realistic moderate-skew regimes (θ=0.5–0.8), where WAN communication remains the dominant bottleneck.

The Setting of Group Number. The number of groups is a key hyperparameter in GeoCoCo, influencing both planning cost and communication efficiency. Figure 19 evaluates different group sizes under 10- and 15-node clusters across two WAN settings. The x-axis shows the group count, and the y-axis shows the makespan reduction over no grouping. The theoretical optimum k^* (§4.2) is derived from expected communication frequency, and the empirical optima (4 and 5 groups) align with our analysis. This illustrates the core trade-off: more groups reduce inter-group cost but increase intra-group coordination. Overall, the results confirm that our cost model effectively guides group selection in practice.

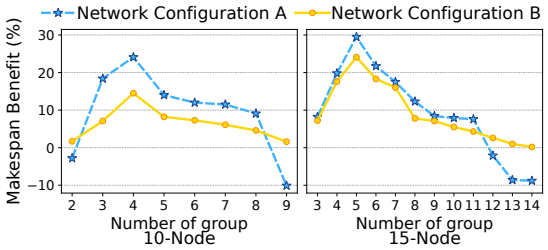


Fig. 19. Optimal Group Number Analysis for Efficient Geo-Distributed Communication.

Inter-group conflicts. Our filtering mechanism primarily focuses on intra-group redundancy, as redundant updates (e.g., repeated writes to the same key) are typically concentrated within latency-proximate groups that align with geographic or workload locality. Although inter-group redundancy (e.g., hot keys or shared counters) can occur, supporting cross-group filtering would require additional global coordination and digest exchanges across aggregators, which could introduce nontrivial overhead and undermine the lightweight nature of GeoCoCo. Given that intra-group redundancy dominates in realistic geo-distributed workloads, this design strikes a practical balance between effectiveness and complexity, while leaving room for future exploration of global coordination when justified by workload characteristics.

7 RELATED WORK

Numerous research efforts have been dedicated to enhancing transmission performance in distributed database systems [4, 9, 14, 24, 26, 27, 34, 34, 37–40, 46, 47, 49, 51, 55, 61, 64, 66–68, 70, 73, 80, 81, 87, 89, 90, 93], primarily focusing on 1) cross transaction-transmission scheduling, 2) transmission optimization, 3) data transmission reduction.

Cross Transaction-Transmission Scheduling. The goal of data transmission in a distributed database is to ensure the correctness of transactions. Considering the hardware feature of network devices, many researchers have dedicated for optimizing the scheduler between transaction and transmission [14, 34, 39, 40, 46, 47, 61, 66, 68, 70, 79, 89, 93]. For example, AlNiCo [46] leverages FPGA-based smart NICs to schedule transaction requests, thereby reducing contention among multiple CPU cores. Hydra [14] uses programmable switches (P4) to perform transaction ordering at the network layer, offloading the ordering logic from servers to reduce their processing burden. SwitchTx [47] offloads distributed transaction coordination to a tree of programmable switches. It reorders in-flight transaction messages based on semantic priorities and tightly couples admission control with congestion control to optimize traffic. Xenic [61] serves as a smart-NIC-optimized transaction engine. It adopts an asynchronous aggregation execution model with custom NIC logic to maximize both network and core efficiency, thereby reducing commit latency. Compared to approaches that integrate transaction processing with network-level scheduling using custom hardware such as programmable switches or smart NICs, GeoCoCo does not rely on any dedicated network devices or modify the underlying transaction scheduling mechanism. This design allows GeoCoCo to be seamlessly integrated with existing concurrency control schemes at minimal integration cost, providing complementary benefits.

Cross-region transmission optimization. Cross-region transmission optimization improves communication efficiency across geographically distributed data centers. Compared to intra-datacenter networks, WANs exhibit higher latency, fluctuating bandwidth, and dynamic link conditions, posing unique challenges for efficient data transfer. To address these issues, modern systems adopt optimizations across multiple layers [4, 9, 20, 26, 27, 37, 38, 49–51, 73, 90, 91]. At the application layer, RPC frameworks like gRPC [26, 37] leverage HTTP/2 to enable long-lived, pipelined streaming.

At the transport layer, QUIC—built on UDP—integrates TLS [38, 73, 90], 0-RTT handshake [27], multiplexing, and connection migration, making it well-suited for WAN conditions. At the **network level**, protocols like TACK [51] reduce ACK overhead on high-latency paths, while Copa [4] balances latency and fairness in congestion control. GeoCoCo differs from these designs in two key ways: it integrates real-time latency into dynamic planning and optimizes transmission at a topology level, rather than treating flows in isolation. This enables GeoCoCo to adapt to live network conditions and optimize WAN-wide performance.

Data transmission reduction. Reducing network traffic to improve transmission efficiency constitutes a classic research direction [5, 7, 11–13, 24, 55, 56, 64, 65, 67, 80–82, 84–88]. To minimize data transfer, geo-replicated systems typically leverage a combination of compression [87], deduplication [80, 81], delta encoding [67], and aggregation [5, 11, 48, 56, 65, 84–86]. In particular, delta encoding transmits only the differences or compressed change logs rather than full records. For instance, dbDedup [80] is an online deduplication scheme for databases that leverages similarity-aware, record-level delta encoding to reduce both storage usage and replication traffic. PHLIP [67], etc. introduces a wide-area data replication architecture that integrates stream-informed delta compression into existing deduplication systems, effectively addressing bandwidth limitations in remote backup scenarios. Cougar [85], etc. pushes operators to intermediate nodes for in-network aggregation, avoiding the transmission of raw readings from all sources and significantly reducing network traffic. GeoCoCo represents the first work applying these data reduction principles at the network layer of distributed databases. Its novel approach significantly reduces network load while strictly maintaining transaction correctness, improving transmission performance by 40.3% by reducing bandwidth usage.

8 CONCLUSION

This paper presents GeoCoCo, a latency-aware group-based synchronization framework designed to address the inefficiencies of geo-distributed database coordination. By systematically identifying structural opportunities—such as regional aggregation hubs, speculative low-latency triangular paths, and data update redundancy—GeoCoCo rethinks the way synchronization is performed across WANs. Our proposed hierarchical group-based transmission design, combined with redundancy-aware filtering and aggregation mechanisms, jointly improves system throughput, reduces overall makespan and transmission cost, while preserving the transactional correctness of data delivery. Experiments on GeoGauss and CockroachDB, as well as large-scale trace-driven evaluations, demonstrate up to 40.3% reduction in synchronization cost and 14.1% throughput improvement, with scalable planning overhead amortized in only a few synchronization rounds. These results highlight the importance of topology-aware design in scaling distributed databases across geographically dispersed infrastructures.

ACKNOWLEDGMENTS

This work is partly inspired by early-stage ideas that arose from discussions with Huawei experts Yang Ren and Jiaqi Liang, and is incubated with support from our collaboration project with Huawei. We are also grateful for insightful conversations with Yunpeng Chai, Zili Meng, and Feng Zhang. This work is supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62441230, the Scientific Research Innovation Capability Support Project for Young Faculty (SRICSPYF-ZY2025001), and the NSFC under Grant Nos. 62572473, 62202473, 62272466, 62322213, and 62461146205. This work is also supported by the China Postdoctoral Science Foundation, including the Postdoctoral Fellowship Program (Grant No. GZC20251044) and Grant No. 2025M781495; and by the Shuimu Scholar Fellowship of Tsinghua University (Grant No. 2025SM061).

References

[1] Alibaba Cloud. Alibaba cloud elastic compute service (ecs) access methods. https://www.alibabacloud.com/product/ecs, 2023. Accessed: 2023-11-15.

[2] Alibaba Cloud. Network performance observation: Inter-region latency metrics. Alibaba Cloud Network Intelligence Service documentation, 2025.

[3] Inc. Amazon Web Services. *AWS Latency Monitoring*, 2024. Accessed: 2024-07-23.

[4] Venkat Arun and Hari Balakrishnan. Copa: Practical delay-based congestion control for the internet. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 329–342, 2018.

[5] Giovanni Bartolomeo, Mehdi Yosofie, Simon Bäurle, Oliver Haluszczynski, Nitinder Mohan, and Jörg Ott. Oakestra: A lightweight hierarchical orchestration framework for edge computing. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, pages 215–231, 2023.

[6] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[7] Jingkun Cao, Tong Li, Bowen Hu, Duling Xu, Kezhi Wang, and Bo Wu. Reducing bandwidth demand for video streaming to kbps. In *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, 2025.

[8] Wei Cao, Feifei Li, Gui Huang, Jianghang Lou, Jianwei Zhao, Dengcheng He, Mengshi Sun, Yingqiang Zhang, Sheng Wang, Xueqiang Wu, et al. Polardb-x: An elastic distributed relational database for cloud-native applications. In *2022 IEEE 38th International Conference on Data Engineering (ICDE)*, pages 2859–2872. IEEE, 2022.

[9] Neal Cardwell, Yuchung Cheng, C Stephen Gunn, Soheil Hassas Yeganeh, and Van Jacobson. Bbr: Congestion-based congestion control: Measuring bottleneck bandwidth and round-trip propagation time. *Queue*, 14(5):20–53, 2016.

[10] Aleksey Charapko, Ailidani Ailijiang, and Murat Demirbas. Pigpaxos: Devouring the communication bottlenecks in distributed consensus. In *Proceedings of the 2021 International Conference on Management of Data*, pages 235–247, 2021.

[11] Xenofon Chatziliadis, Eleni Tzirita Zacharatou, Alphan Eracar, Steffen Zeuch, and Volker Markl. Efficient placement of decomposable aggregation functions for stream processing over large geo-distributed topologies. *Proceedings of the VLDB Endowment*, 17(6):1501–1514, 2024.

[12] Zheng Chen, Feng Zhang, JiaWei Guan, Jidong Zhai, Xipeng Shen, Huanchen Zhang, Wentong Shu, and Xiaoyong Du. Compressgraph: Efficient parallel graph analytics with rule-based compression. *Proceedings of the ACM on Management of Data*, 1(1):1–31, 2023.

[13] Zheng Chen, Feng Zhang, Yifei Xia, Wentao Zhang, Xiaowei Zhu, Wenguang Chen, and Xiaoyong Du. Compressgnn: Accelerating graph neural network training via hierarchical compression. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, pages 286–297, 2025.

[14] Inho Choi, Ellis Michael, Yunfan Li, Dan RK Ports, and Jialin Li. Hydra:{Serialization-Free} network ordering for strongly consistent distributed applications. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 293–320, 2023.

[15] ZeroMQ Community. Zeromq: High-performance asynchronous messaging library. https://github.com/zeromq, 2025. Accessed: January 1, 2025.

[16] Brian F Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM symposium on Cloud computing*, pages 143–154, 2010.

[17] James C Corbett, Jeffrey Dean, Michael Epstein, Andrew Fikes, Christopher Frost, Jeffrey John Furman, Sanjay Ghemawat, Andrey Gubarev, Christopher Heiser, Peter Hochschild, et al. Spanner: Google's globally distributed database. *ACM TOCS*, 31(3):1–22, 2013.

[18] Frank Dabek, Russ Cox, Frans Kaashoek, and Robert Morris. Vivaldi: A decentralized network coordinate system. *ACM SIGCOMM Computer Communication Review*, 34(4):15–26, 2004.

[19] Hung Dang, Tien Tuan Anh Dinh, Dumitrel Loghin, Ee-Chien Chang, Qian Lin, and Beng Chin Ooi. Towards scaling blockchain systems via sharding. In *Proceedings of the 2019 international conference on management of data*, pages 123–140, 2019.

[20] Xinle Du, Tong Li, Guangmeng Zhou, Zhuotao Liu, Hanlin Huang, Xiangyu Gao, Mowei Wang, Kun Tan, and Ke Xu. Pred: Performance-oriented random early detection for consistently stable performance in datacenters. In *22nd USENIX Symposium on Networked Systems Design and Implementation (NSDI 25)*, pages 1–20, 2025.

[21] Hua Fan and Wojciech Golab. Ocean vista: gossip-based visibility control for speedy geo-distributed transactions. *Proc. VLDB Endow.*, 12(11):1471–1484, July 2019.

[22] Linux Foundation. *Traffic Control (tc) and Network Emulation (netem)*, 2024. Accessed: 2024-07-23.

[23] Frederick N Fritsch and Ralph E Carlson. Monotone piecewise cubic interpolation. *SIAM Journal on Numerical Analysis*, pages 238–246, 1980.

[24] Jean-loup Gailly and Mark Adler. zlib: A massively spiffy yet delicately unobtrusive compression library. https://github.com/madler/zlib, 2025. GitHub repository; accessed: 2025-07-01.

[25] James N Gray. Notes on data base operating systems. *Operating systems: An advanced course*, pages 393–481, 2005.

[26] gRPC Authors and Contributors. grpc: A high-performance open source rpc framework. https://github.com/grpc, 2025. Accessed: January 1, 2025.
[27] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-rtt key exchange with full forward secrecy. In *Advances in Cryptology–EUROCRYPT 2017: 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30–May 4, 2017, Proceedings, Part III 36*, pages 519–548. Springer, 2017.
[28] LLC Gurobi Optimization. Gurobi optimizer, 2023. Commercial mathematical optimization solver.
[29] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*, 2024.
[30] Jelle Hellings and Mohammad Sadoghi. Byshard: Sharding in a byzantine environment. *Proceedings of the VLDB Endowment*, 14(11):2230–2243, 2021.
[31] P Hintjens. *ZeroMQ: Messaging for Many Applications*. O'Reilly Media, 2013.
[32] Chi-Yao Hong, Srikanth Kandula, Ratul Mahajan, Ming Zhang, Vijay Gill, Mohan Nanduri, and Roger Wattenhofer. Achieving high utilization with software-driven wan. In *Proceedings of the ACM SIGCOMM 2013 Conference on SIGCOMM*, pages 15–26, 2013.
[33] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia:{Geo-Distributed} machine learning approaching {LAN} speeds. In *14th USENIX symposium on networked systems design and implementation (NSDI 17)*, pages 629–647, 2017.
[34] Chunyue Huang, Shuang Liu, Xinyi Zhang, Wenhao Li, Wei Lu, and Xiaoyong Du. Chimera: Mitigating ownership transfers in multi-primary shared-storage cloud-native databases. *Proceedings of the VLDB Endowment*, 18(10):3368–3381, 2025.
[35] Dongxu Huang, Qi Liu, Qiu Cui, Zhuhe Fang, Xiaoyu Ma, Fei Xu, Li Shen, Liu Tang, Yuxing Zhou, Menglong Huang, et al. Tidb: a raft-based htap database. *Proceedings of the VLDB Endowment*, 13(12):3072–3084, 2020.
[36] iDC NEU. Geogauss: A high-performance distributed database for geo-distributed environments. https://github.com/iDC-NEU/GeoGauss, 2025. Accessed: 2025-01-10.
[37] Kasun Indrasiri and Danesh Kuruppu. *gRPC: up and running: building cloud native applications with Go and Java for Docker and Kubernetes*. O'Reilly Media, 2020.
[38] J. Iyengar and M. Thomson. QUIC: A UDP-Based Multiplexed and Secure Transport. Technical report, IETF, May 2021.
[39] Theo Jepsen, Alberto Lerner, Fernando Pedone, Robert Soulé, and Philippe Cudré-Mauroux. In-network support for transaction triaging. *Proceedings of the VLDB Endowment*, 14(9):1626–1639, 2021.
[40] Xin Jin, Xiaozhou Li, Haoyu Zhang, Nate Foster, Jeongkeun Lee, Robert Soulé, Changhoon Kim, and Ion Stoica. {NetChain}:{Scale-Free}{Sub-RTT} coordination. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*, pages 35–49, 2018.
[41] Tim Kraska, Martin Hentschel, Gustavo Alonso, and Donald Kossmann. Consistency rationing in the cloud: Pay only when it matters. *Proceedings of the VLDB Endowment*, 2(1):253–264, 2009.
[42] Dhruv Kumar, Sohaib Ahmad, Abhishek Chandra, and Ramesh K Sitaraman. Aggnet: Cost-aware aggregation networks for geo-distributed streaming analytics. In *2021 IEEE/ACM Symposium on Edge Computing (SEC)*, pages 297–311. IEEE, 2021.
[43] Cockroach Labs. Cockroachdb latency benchmarks. https://www.cockroachlabs.com/docs/stable/performance-latency.html, 2020.
[44] Jonathan Ledlie, Paul Gardner, and Margo I Seltzer. Network coordinates in the wild. In *NSDI*, volume 7, pages 299–311, 2007.
[45] Cheng Li, Daniel Porto, Allen Clement, Johannes Gehrke, Nuno Preguiça, and Rodrigo Rodrigues. Making geo-replicated systems fast as possible, consistent when necessary. In *10th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 265–278, Hollywood, CA, USA, 2012. USENIX Association.
[46] Junru Li, Youyou Lu, Qing Wang, Jiazhen Lin, Zhe Yang, and Jiwu Shu. {AlNiCo}:{SmartNIC-accelerated} contention-aware request scheduling for transaction processing. In *2022 USENIX annual technical conference (USENIX ATC 22)*, pages 951–966, 2022.
[47] Junru Li, Youyou Lu, Yiming Zhang, Qing Wang, Zhuo Cheng, Keji Huang, and Jiwu Shu. Switchtx: scalable in-network coordination for distributed transaction processing. *Proceedings of the VLDB Endowment*, 15(11):2881–2894, 2022.
[48] Tong Li, Wei Liu, Xinyu Ma, Shuaipeng Zhu, Jingkun Cao, Duling Xu, Zhaoqi Yang, Senzhen Liu, Taotao Zhang, Yinfeng Zhu, Bo Wu, Kezhi Wang, and Ke Xu. Accelerating loss recovery for content delivery network. *IEEE Transactions on Computers*, 74(7):2223–2237, 2025.
[49] Tong Li, Duling Xu, Bo Wu, Xiongwen Guo, Daijun Jiang, Cheng Luo, Wei Lu, and Xiaoyong Du. A survey on transmission technology in wide-area deterministic networking. *Journal of Software*, 36(1):371–398, 2025. In Chinese.
[50] Tong Li, Xu Yan, Bo Wu, Cheng Luo, Fuyu Wang, Jiuxiang Zhu, Haoyi Fang, Xinle Du, and Ke Xu. Autorec: Accelerating loss recovery for live streaming in a multi-supplier market. *arXiv preprint arXiv:2511.22046*, 2025.
[51] Tong Li, Kai Zheng, Ke Xu, Rahul Arvind Jadhav, Tao Xiong, Keith Winstein, and Kun Tan. Tack: Improving wireless transport performance by taming acknowledgments. In *Proceedings of SIGCOMM 2020*, 2020.

[52] Linux man-pages Project. proc(5) — linux manual page. https://man7.org/linux/man-pages/man5/proc.5.html. Accessed: 2025-07-18.

[53] Xuting Liu, Behnaz Arzani, Siva Kesava Reddy Kakarla, Liangyu Zhao, Vincent Liu, Miguel Castro, Srikanth Kandula, and Luke Marshall. Rethinking machine learning collective communication as a multi-commodity flow problem. In *Proceedings of the ACM SIGCOMM 2024 Conference*, pages 16–37, 2024.

[54] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[55] Jean loup Gailly and Mark Adler. zlib home site: A massively spiffy yet delicately unobtrusive compression library. https://www.zlib.net/?hl=zh-cn. Accessed: 2025-01-02.

[56] Samuel R Madden, Michael J Franklin, Joseph M Hellerstein, and Wei Hong. Tinydb: an acquisitional query processing system for sensor networks. *ACM Transactions on database systems (TODS)*, 30(1):122–173, 2005.

[57] Kabir Nagrecha and Arun Kumar. Saturn: An optimized data system for multi-large-model deep learning workloads. *Proc. VLDB Endow.*, 17(4):712–725, December 2023.

[58] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12:2825–2830, 2011.

[59] Qifan Pu, Ganesh Ananthanarayanan, Peter Bodik, Srikanth Kandula, Aditya Akella, Paramvir Bahl, and Ion Stoica. Low latency geo-distributed data analytics. *ACM SIGCOMM Computer Communication Review*, 45(4):421–434, 2015.

[60] Kun Ren, Dennis Li, and Daniel J Abadi. Slog: Serializable, low-latency, geo-replicated transactions. *Proceedings of the VLDB Endowment*, 12(11), 2019.

[61] Henry N Schuh, Weihao Liang, Ming Liu, Jacob Nelson, and Arvind Krishnamurthy. Xenic: Smartnic-accelerated distributed transactions. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*, pages 740–755, 2021.

[62] Amazon Web Services. Amazon ec2 price. https://aws.amazon.com/cn/ec2/pricing/, 2025. 20250628.

[63] Amazon Web Services. Infrastructure Performance (latency metrics) in aws network manager. AWS Documentation, 2025.

[64] Marc Shapiro, Nuno Preguiça, Carlos Baquero, and Marek Zawirski. Conflict-free replicated data types. In *Proc. of the Symposium on Self-Stabilizing Systems (SSS'11)*, volume 6976 of *LNCS*, pages 386–400. Springer, 2011.

[65] Chien-Chung Shen, Chavalit Srisathapornphat, and Chaiporn Jaikaeo. Sensor information networking architecture and applications. *IEEE Personal communications*, 8(4):52–59, 2001.

[66] Weihai Shen, Yang Cui, Siddhartha Sen, Sebastian Angel, and Shuai Mu. Mako: Speculative distributed transactions with geo-replication. In *19th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, 2025.

[67] Phlip Shilane, Mark Huang, Grant Wallace, and Windsor Hsu. Wan-optimized replication of backup datasets using stream-informed delta compression. *ACM Transactions on Storage (ToS)*, 8(4):1–26, 2012.

[68] Cha Hwan Song, Xin Zhe Khooi, Raj Joshi, Inho Choi, Jialin Li, and Mun Choon Chan. Network load balancing with in-network reordering support for rdma. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 816–831, 2023.

[69] Won Wook Song, Myeongjae Jeon, and Byung-Gon Chun. Swan: Wan-aware stream processing on geographically-distributed clusters. In *Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems*, pages 78–84, 2022.

[70] Guangda Sun, Mingliang Jiang, Xin Zhe Khooi, Yunfan Li, and Jialin Li. Neobft: Accelerating byzantine fault tolerance using authenticated in-network ordering. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 239–254, 2023.

[71] Rebecca Taft, Irfan Sharif, Andrei Matei, Nathan VanBenschoten, Jordan Lewis, Tobias Grieger, Kai Niemi, Andy Woods, Anne Birzin, Raphael Poss, et al. Cockroachdb: The resilient geo-distributed sql database. In *ACM SIGMOD*, pages 1493–1509, 2020.

[72] Alexander Thomson, Thaddeus Diamond, Shu-Chun Weng, Kun Ren, Philip Shao, and Daniel J Abadi. Calvin: fast distributed transactions for partitioned database systems. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, pages 1–12, 2012.

[73] M. Thomson and S. Turner. Using TLS to Secure QUIC. Technical report, IETF, May 2021.

[74] TPC. *TPC-C Benchmark Specification*, 2025. 20250620.

[75] Alexandre Verbitski, Anurag Gupta, Debanjan Saha, Murali Brahmadesam, Kamal Gupta, Raman Mittal, Sailesh Krishnamurthy, Sandor Maurice, Tengiz Kharatishvili, and Xiaofeng Bao. Amazon aurora: Design considerations for high throughput cloud-native relational databases. In *ACM SIGMOD*, pages 1041–1052, 2017.

[76] Tianzheng Wang and Hideaki Kimura. Mostly-optimistic concurrency control for highly contended dynamic workloads on a thousand cores. *Proceedings of the VLDB Endowment*, 10(2):49–60, 2016.

[77] WonderNetwork. Ping: Test your latency to any of our servers. https://wondernetwork.com/pings. Accessed: January 2025.

[78] Yingjun Wu, Chee-Yong Chan, and Kian-Lee Tan. Transaction healing: Scaling optimistic concurrency control on multicores. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1689–1704, 2016.

[79] Duling Xu, Dafang Zhang, Tong Li, Yunpeng Chai, Zegang Sun, Weiming Li, Yangfan Liu, Qipeng Wang, Jiaqi Liang, Yang Ren, et al. Geolm: Performance-oriented leader management for geo-distributed consensus protocol. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2025.

[80] Lianghong Xu, Andrew Pavlo, Sudipta Sengupta, and Gregory R Ganger. Online deduplication for databases. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1355–1368, 2017.

[81] Lianghong Xu, Andrew Pavlo, Sudipta Sengupta, Jin Li, and Gregory R Ganger. Reducing replication bandwidth for distributed document databases. In *Proceedings of the Sixth ACM Symposium on Cloud Computing*, pages 222–235, 2015.

[82] Qian Xu, Juan Yang, Feng Zhang, Zheng Chen, Jiawei Guan, Kang Chen, Ju Fan, Youren Shen, Ke Yang, Yu Zhang, et al. Improving graph compression for efficient resource-constrained graph analytics. *Proceedings of the VLDB Endowment*, 17(9):2212–2226, 2024.

[83] Zhiying Xu, Francis Y Yan, Rachee Singh, Justin T Chiu, Alexander M Rush, and Minlan Yu. Teal: Learning-accelerated optimization of wan traffic engineering. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 378–393, 2023.

[84] Feng Yao, Qian Tao, Wenyuan Yu, Yanfeng Zhang, Shufeng Gong, Qiange Wang, Ge Yu, and Jingren Zhou. Ragraph: A region-aware framework for geo-distributed graph processing. *Proceedings of the VLDB Endowment*, 17(3):264–277, 2023.

[85] Yong Yao and Johannes Gehrke. The cougar approach to in-network query processing in sensor networks. *ACM Sigmod record*, 31(3):9–18, 2002.

[86] Ye Yuan, Delong Ma, Zhenyu Wen, Yuliang Ma, Guoren Wang, and Lei Chen. Efficient graph query processing over geo-distributed datacenters. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 619–628, 2020.

[87] Feng Zhang, Zheng Chen, Chenyang Zhang, Amelie Chi Zhou, Jidong Zhai, and Xiaoyong Du. An efficient parallel secure machine learning framework on gpus. *IEEE Transactions on Parallel and Distributed Systems*, 32(9):2262–2276, 2021.

[88] Feng Zhang, Jidong Zhai, Xipeng Shen, Dalin Wang, Zheng Chen, Onur Mutlu, Wenguang Chen, and Xiaoyong Du. Tadoc: Text analytics directly on compression. *The VLDB Journal*, 30(2):163–188, 2021.

[89] Qian Zhang, Jingyao Li, Hongyao Zhao, Quanqing Xu, Wei Lu, Jinliang Xiao, Fusheng Han, Chuanhui Yang, and Xiaoyong Du. Efficient distributed transaction processing in heterogeneous networks. *Proceedings of the VLDB Endowment*, 16(6):1372–1385, 2023.

[90] Xumiao Zhang, Shuowei Jin, Yi He, Ahmad Hassan, Z.Morley Mao, Feng Qian, and Zhi-Li Zhang. Quic is not quick enough over fast internet. *ACM SIGCOMM/IMC (extended version on arXiv)*, 2023.

[91] Yinchao Zhang, Su Yao, Yong Feng, Kang Chen, Tong Li, Zhuotao Liu, Yi Zhao, Lexuan Zhang, Xiangyu Gao, Feng Xiong, et al. Pegasus: A universal framework for scalable deep learning inference on the dataplane. In *Proceedings of the ACM SIGCOMM 2025 Conference*, pages 692–706, 2025.

[92] Weixing Zhou, Qi Peng, Zijie Zhang, Yanfeng Zhang, Yang Ren, Sihao Li, Guo Fu, Yulong Cui, Qiang Li, Caiyi Wu, et al. Geogauss: Strongly consistent and light-coordinated oltp for geo-replicated sql database. *Proceedings of the ACM on Management of Data*, pages 1–27, 2023.

[93] Qiyu Zhuang, Xinyue Shi, Shuang Liu, Wei Lu, Zhanhao Zhao, Yuxing Chen, Tong Li, Anqun Pan, and Xiaoyong Du. Geotp: Latency-aware geo-distributed transaction processing in database middlewares. In *2025 IEEE 41st International Conference on Data Engineering (ICDE)*, pages 433–445. IEEE, 2025.

Received July 2025; revised October 2025; accepted November 2025