INTERNATIONAL JOURNAL OF INFORMATION AND COMMUNICATION TECHNOLOGY TRENDS

Vol. 6 No. 02 (2026)

e.ISSN: 2409-5222

https://ijictt.ictt.info

Heliotrope: Quorum- and SLO-Aware Adaptive Leader Placement for Raft-Based Geo-Distributed Databases

Victor Phillips [1*], ***Serena Tran*** [2]

[1]*School of Computing, Clemson University, USA*

[2]*IBM Corporation, USA*

Abstract

In Raft-based geo-distributed databases, every write pays the round trip from the client to the leader of its Raft group plus the round trip from the leader to the slowest member of the fastest majority, so the region that hosts each leader largely determines user-visible latency. Deployed schedulers move leaders toward the region that issues most writes; this rule ignores quorum geometry, reacts to noisy short windows, and treats leader transfers, which stall a group and are rate-limited in practice, as free. We present Heliotrope, a fleet-level leader placement controller for Multi-Raft deployments. Heliotrope computes the exact commit-latency distribution of every leader candidate from measured round-trip-time distributions with a Poisson-binomial order-statistic recursion, scores candidates by mean latency plus an SLO-excess term whose fleet-wide aggregate is linear in demand, estimates near-term regional demand by shrinking the latest observation toward a seasonal prior, and assigns the leaders of all groups jointly under leader-load capacity while pricing every transfer with a dual variable that tracks a transfer budget. We evaluate Heliotrope with measured AWS and GCP inter-region RTTs and a geo-temporal workload derived from 838,178 human-authored commits to 440 public repositories over 13 weeks. At a budget of two leader transfers per group per day, Heliotrope lowers mean write latency by 2.2% and the share of writes slower than 200 ms by 18.4% relative to the best budget-matched baseline, wins on every one of the 63 test days, and has the fewest overloaded epochs and the lowest write latency of all evaluated policies when leader capacity is tight. A message-level replay of Raft confirms the analytic latency model to within 0.7 ms.

Keywords *: Raft, consensus, geo-distributed databases, leader placement, tail latency, online optimization.*

1. Introduction

Globally distributed databases such as Spanner [1], CockroachDB [2], and TiDB [3] split their data into ranges and replicate each range with a consensus protocol, Paxos [4] or Raft [5]. A deployment therefore runs thousands of independent consensus groups, each with its own leader. In leader-based consensus a write first travels to the leader, which appends it to its log and waits until a majority of replicas have stored it before acknowledging the client. Across continents both legs cost tens to hundreds of milliseconds, so the region that hosts each leader is one of the few latency knobs that does not require changing the protocol.

The best region is a moving target. Users of a global service are active during their local working hours, so the regional mix of requests that reaches a group drifts over the day and the week. Production systems combine static preferences, such as a primary region, with feedback-driven schedulers that transfer a group's leader toward the region that generates most of its writes. That rule is simple, but it has three weaknesses. It ignores quorum geometry: a leader's commit latency also depends on how far it is from the other members of its quorum, and a busy region whose second-nearest replica is far away can be a poor leader. It reacts to short, noisy windows and therefore either lags or oscillates. And it decides group by group while treating transfers as free, although every transfer stalls the group for a few round trips, operators rate-limit transfers, and leaders concentrate CPU and network load on the nodes that host them. Recent leader managers model network delay and demand explicitly. GeoLM [6] scores leader candidates with a utility model of network delay, node popularity, and read-write ratio and damps switching to avoid oscillation; CD-Raft [7] periodically moves a global leader to the domain that minimizes total cross-domain latency computed from request counts and mean inter-domain delays; and Sharov et al. [8] optimize the leader placement and replica configuration of a production storage system from workload statistics. These approaches reason about mean delays of individual consensus groups. They do not target latency SLOs, do not price leader transfers against an operational budget, and do not couple the thousands of groups of a Multi-Raft deployment through shared leader capacity.

This paper presents Heliotrope, a leader placement controller for Raft-based geo-distributed databases that closes these gaps without modifying Raft. Our contributions are fourfold. First, we derive the exact commit-latency distribution of every leader candidate from measured round-trip-time (RTT) distributions with a Poisson-binomial order-statistic recursion, and we define an SLO-aware per-request cost whose fleet-wide aggregate is exactly linear in demand (Section III). Second, we design a controller that combines an empirical-Bayes demand estimate, transfer pricing with online budget pacing, and a capacity-constrained fleet assignment solved by MILP or by a Lagrangian method, and we show that it cannot oscillate when estimation errors are bounded (Section IV). Third, we build a trace-driven methodology from real geo-temporal activity (838,178 commits to 440 repositories over 13 weeks) and measured RTTs, validated by a message-level replay of Raft (Section V). Fourth, we show that at two transfers per group per day Heliotrope reduces mean write latency by 2.2% and writes slower than 200 ms by 18.4% relative to the best budget-matched baseline, stays within 3.9 ms of a clairvoyant oracle, and has the fewest overloaded epochs and the lowest write latency of all evaluated policies when leader capacity is tight (Section VI).

II. Background and Related Work

A. Raft in Geo-Distributed Databases

Raft [5] elects one leader per term; the leader appends client commands to its log, replicates them with AppendEntries messages, and commits an entry once a majority of replicas store it. Ongaro's thesis [9] adds leadership transfer: the leader stops accepting proposals, brings the target up to date, and sends it a TimeoutNow message so that the target starts an election at once. Leader leases let the leader serve linearizable reads locally. Multi-Raft systems such as TiKV in TiDB [3] and CockroachDB [2] replicate each range with its own Raft group, and a placement component decides where replicas and leaders live.

B. Wide-Area Consensus

A line of work changes the replication protocol itself. Mencius [10] rotates leadership among replicas, EPaxos [11] removes the leader and commits non-conflicting commands in one round trip, Flexible Paxos [12] decouples the quorums of the two Paxos phases, and WPaxos [13] combines flexible grid quorums with zone-local object ownership that migrates toward accessors. Atlas [14] shrinks quorums for planet-scale deployments, and Domino [15] uses network measurements to choose between fast and leader-based paths; Ailijiang et al. [16] dissect the performance of strongly consistent replication protocols. Heliotrope is complementary: it keeps Raft's majority quorums and safety argument and only decides where leaders live.

C. Leader and Data Placement

Besides the leader managers discussed above [6]–[8], Tuba [17] reconfigures a geo-replicated store to maximize SLA utility, and Akkio [18] migrates data shards to follow access locality. PBS [19] models quorum latency probabilistically by Monte-Carlo simulation, and the importance of latency tails in large services is well documented [20]. Heliotrope computes quorum-latency distributions exactly, optimizes an SLO-aware objective, and couples placement decisions across groups. Risk-aware digital-twin release validation illustrates how tenant-level operational constraints and tail-risk measures can support auditable infrastructure decisions [28].

D. Online Decisions with Switching Costs

Leader placement with transfer costs is an instance of a metrical task system [21]: every epoch charges a state-dependent cost, and changing state costs extra. Work on dynamic right-sizing of data centers [22] studies lazy online algorithms and receding-horizon control for related switching problems. Heliotrope uses seasonal exponential smoothing [23] for its demand prior, Lagrangian relaxation [24] for its fast fleet solver, and HiGHS [25] for exact assignments.

III. System Model and Problem Formulation

A. Deployment and Workload

We consider a set of regions R and a fleet G of Raft groups. Group g has a replica set R_g with $n_g=|R_g|$ replicas in distinct regions, and in control epoch t (five minutes in our experiments) its leader is $\ell_g(t)\in R_g$. Clients in region c issue $\lambda_{g,c}(t)$ operations to group g during epoch t, a fraction f_w of which are writes. Writes follow Raft's commit path, and linearizable reads are served by the leaseholder, which coincides with the leader.

B. Commit-Path Latency

We adopt the commit-path decomposition that Ding and Qin [26] use to analyze Raft [5], [9] under WAN conditions: the commit latency of a leader equals the RTT to the slowest member of the fastest majority quorum, i.e., the third-closest replica (counting the leader) of a five-replica group. Adding the client leg, a write from region c to a group whose leader is in region ℓ completes after

$$W_g(c,\ell) = X_{c\ell} + Q_g(\ell), \quad k = \lfloor n_g/2 \rfloor, \qquad (1)$$

where X_{ab} is the RTT between regions a and b and $Q_g(\ell)$ is the k-th smallest RTT from ℓ to the other replicas of R_g. A lease read costs $R(c,\ell)=X_{c\ell}$. The decomposition is usually stated for typical (median) RTTs; we extend it to full distributions, because SLO compliance is decided by the tail of W rather than by its median. With independent links, $Q_g(\ell)\le x$ holds exactly when at least k followers have answered by time x:

$$\mathrm{P}[Q_g(\ell)\le x] = \Sigma_{m=k}^{n_g-1}\, p_{n_g-1}(m;x), \qquad (2)$$

where $p_j(m;x)$ is the probability that exactly m of the first j followers satisfy $X_{\ell j}\le x$. It obeys the Poisson-binomial recursion

$$p_j(m;x) = p_{j-1}(m;x)(1-F_{\ell j}(x)) + p_{j-1}(m-1;x)F_{\ell j}(x) \qquad (3)$$

with $p_0(0;x)=1$, where $F_{\ell j}$ is the RTT distribution of link (ℓ,j). On a grid of T points the recursion costs $O(n_g^2 T)$, and the distribution of $W_g(c,\ell)$ follows by convolving the client-leader RTT distribution with that of $Q_g(\ell)$.

C. SLO-Aware Cost

For an SLO threshold τ and a weight η, Heliotrope charges every operation

$$\kappa_g(c,\ell) = f_w\psi(W_g(c,\ell)) + (1-f_w)\psi(R(c,\ell)), \qquad (4)$$

where $\psi(Y)=\mathrm{E}[Y]+\eta\mathrm{E}[(Y-\tau)^+]$ adds to the mean of a latency Y its expected excess over the threshold.

Proposition 1. For any placement, the fleet-wide mean latency and the fleet-wide mean excess over τ are both given, up to normalization by the total demand, by $\Sigma_g\Sigma_c\lambda_{g,c}\kappa_g(c,\ell_g)$; the objective is therefore exact rather than a surrogate.

Proof. The latency of a uniformly chosen operation is a mixture over (group, client region) pairs with weights proportional to $\lambda_{g,c}$, and the expectation of any function of a mixture is the weighted sum of the component expectations. ■

Quantiles are not linear in the mixture weights. The expected excess over τ is the tail-sensitive quantity that keeps the problem linear; we additionally report the fraction of writes above τ, which is also linear, as the SLO-violation rate.

D. Leader Transfers

Moving leadership of group g from region a to region b stalls the group for

$$D_g(a,b) = 1.5\mathrm{E}[X_{ab}] + \mathrm{E}[Q_g(b)], \qquad (5)$$

which covers the round trip in which the target catches up with the last entries, the one-way TimeoutNow message, and the vote round of the new leader. Lease reads wait one more quorum round for the new leader's first commit, and requests that reach the old leader are redirected. Section V-C validates (5) against a message-level replay. Beyond the stall, transfers consume operational capacity: schedulers rate-limit them, and they disturb caches and leases. We therefore bound the time-average number of transfers by a budget of B per group per day.

E. Leader Capacity and Problem Statement

Leaders perform most of a group's work. If region r can process μ_r leader operations per second, the leader load $\Lambda_r(t)=\Sigma_{g:\ell g(t)=r}\Sigma_c\lambda_{g,c}(t)/\Delta$ for epoch length Δ must satisfy $\Lambda_r(t)\le\rho_{\max}\mu_r$ for a planning utilization $\rho_{\max}$. Heliotrope chooses $\ell_g(t)$ online, using only observations from epochs before t, to minimize the time average of $\Sigma_g\Sigma_c\lambda_{g,c}(t)\kappa_g(c,\ell_g(t))$ plus transfer stalls, subject to the capacity constraint in every epoch and to the transfer budget on average.

IV. Heliotrope Design

A. Overview

Fig. 1 shows the control loop. At the start of every epoch, Heliotrope (1) estimates each group's regional demand, (2) looks up the cost of every leader candidate in precomputed oracle tables, (3) solves a fleet-wide assignment in which every transfer carries a price, and (4) issues standard Raft leadership transfers; a pacing rule then updates the transfer price so that the long-run transfer rate meets the budget.

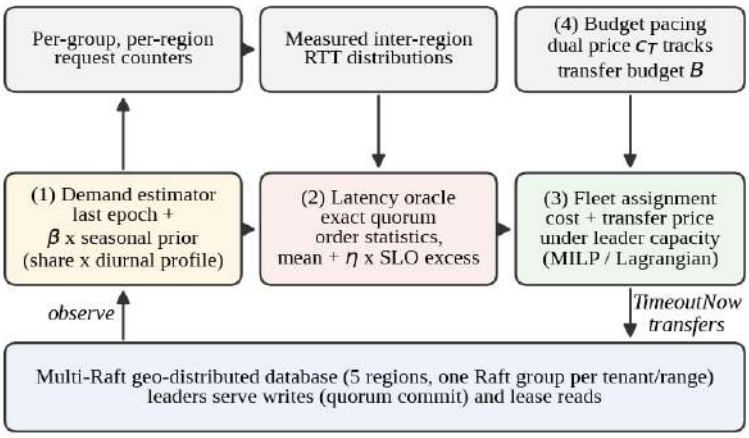


Figure 1Heliotrope control loop. Steps (1)–(3) run once per five-minute epoch; step (4) adjusts the transfer price cT toward the transfer budget B.

B. Latency Oracle

The oracle discretizes the RTT distribution of every directed link on a 0.5-ms grid up to 2 s, evaluates (2)–(3) for every replica set and leader, convolves the result with the client leg, and tabulates $\mathrm{E}[W]$, $\mathrm{E}[(W-\tau)^+]$, $\mathrm{E}[R]$ and $\mathrm{E}[(R-\tau)^+]$ for every (replica set, client region, leader) triple. Five regions with five replicas give 25 triples, and three replicas out of five regions give at most 150. Building all tables takes about 0.1 s, and a Monte-Carlo check with 2×10^5 samples per link reproduces the quorum quantiles within the grid resolution.

C. Demand Estimation

Per-group demand is sparse and bursty, so the last observation alone is a noisy guide, whereas long-run averages miss diurnal shifts. Heliotrope therefore shrinks the last observation toward a seasonal prior, an empirical-Bayes estimate with β pseudo-epochs of prior evidence:

$$m_{g,c}(t) = \lambda_{g,c}(t-1) + \beta\, s_{g,c}\, v_c\, P_c(d(t),h(t)), \qquad (6)$$

where $s_{g,c}$ is the long-run share of region c's demand that targets group g (exponentially weighted with a half-life of three days), $P_c(d,h)$ is an exponentially smoothed profile [23] of region c's total demand in five-minute slot h of day type d (weekday or weekend), and v_c corrects the level of the profile. On the tuning week, choosing every group's leader from the last observation alone incurs a one-step decision regret of 8.4% of the clairvoyant cost and the seasonal prior alone 28.6%, whereas shrinkage with β=0.1 reduces the regret to 5.0% (β=0.3: 5.2%; β=1: 6.3%). For a lookahead of H>1 epochs Heliotrope sums the predictions of the next H epochs, letting them decay toward the prior with factor φ=0.8 per epoch. We selected H on the tuning week with a pre-declared criterion, the combined read/write mean latency at the transfer budget, and H=1 won (78.76 ms versus 81.17 ms for H=3). Wang et al. develop an uncertainty-aware cloud scheduling framework that links workload forecasting with downstream resource decisions, providing a useful methodological reference for demand-sensitive infrastructure control [29].

D. Fleet Assignment

Let $C_{g,\ell}=\Sigma_c m_{g,c}\kappa_g(c,\ell)$ be the predicted cost of leader ℓ, let $\ell_g^{\,0}$ be the current leader, and let $\delta_g(a,b)$ be the expected stall cost of a transfer, i.e., the predicted request rate times $f_w D_g(a,b)^2/2$ plus the corresponding read term. With binary variables $x_{g,\ell}$, Heliotrope solves

$$\min \Sigma_g \Sigma_{\ell \in Rg}\, x_{g,\ell}[C_{g,\ell} + 1(\ell \neq \ell_g^{\,0})(c_T + \delta_g(\ell_g^{\,0},\ell))] \qquad (7)$$

$$\text{s.t. } \Sigma_\ell x_{g,\ell} = 1 \ \forall g, \quad \Sigma_g L_g x_{g,r} \le \rho_{\max} \mu_r \ \forall r, \qquad (8)$$

where L_g is the predicted leader load of group g and c_T is the transfer price. Without binding capacity the problem decomposes into independent per-group minimizations. Otherwise Heliotrope solves it exactly with HiGHS or uses a Lagrangian method [24]: region prices y_r are raised by subgradient steps while each group picks $\operatorname{argmin}_\ell(C_{g,\ell}+y_\ell L_g)$ together with its transfer terms, and a greedy repair moves the group with the smallest cost increase per unit of load out of any region that is still over capacity. Section VI-E reports the optimality gap of the fast method.

E. Transfer Budget Pacing

The transfer price is the shadow price of the budget. After an epoch with n_t transfers Heliotrope updates

$$c_T \leftarrow c_T \cdot \exp(\gamma(288 n_t/|\mathrm{G}| - B)/B), \qquad (9)$$

a multiplicative dual-ascent step on the time-average constraint (288 epochs per day, γ=0.002). The price rises when transfers run ahead of the budget and decays when they lag, so transfers are spent where predicted savings are largest.

F. Stability

Proposition 2. For a group g with current leader a, let $S_{ab}=C_{g,a}-C_{g,b}$ be the predicted saving of moving to b and S^*_{ab} the realized one. If $|S_{ab}-S^*_{ab}|\le\varepsilon<c_T$ for all epochs and candidates, then (i) every executed transfer realizes a saving larger than $c_T-\varepsilon>0$; (ii) under stationary demand no group ever moves $a\to b$ and later $b\to a$; and (iii) the number of transfers of a group during a stationary period is at most $(C_{\max}-C_{\min})/(c_T-\varepsilon)$.

Proof sketch. A transfer $a\to b$ requires $S_{ab}>c_T+\delta_g(a,b)\ge c_T$, hence $S^*_{ab}>c_T-\varepsilon$. Under stationarity $S^*_{ba}=-S^*_{ab}<0$, whereas a move back would require $S^*_{ba}>c_T-\varepsilon>0$. Each transfer lowers the true cost by more than $c_T-\varepsilon$, which bounds their number. ■ Budget pacing keeps c_T near the level at which the budget binds, which is the regime in which the price exceeds typical estimation noise.

G. Safety and Overhead

Heliotrope only issues Raft's standard leadership transfer and never changes membership or quorum rules, so Raft's safety properties are untouched; a transfer that fails, e.g., because the target lags, leaves the old leader in place. Our implementation (Python with NumPy and SciPy) places all 440 groups in about 1.2 ms per epoch on one CPU core when capacity does not bind, including the evaluation engine's bookkeeping.

V. Experimental Methodology

A. Measured Inter-Region RTTs

We use the inter-region ping measurements published with the fantoch framework that accompanies Atlas [14] (github.com/vitorenesduarte/fantoch). For every ordered pair of 19 AWS regions (June 2020) and 20 GCP regions, the data give the minimum, mean, maximum and standard deviation of the RTT. The main deployment uses five AWS regions, us-east-1 (NA-E), us-west-2 (NA-W), eu-west-1 (EU), ap-northeast-1 (AP-NE) and ap-southeast-1 (AP-SE), whose mean RTTs range from 70 to 234 ms (Fig. 2a). Each pair's RTT is modeled as the measured minimum plus a gamma-distributed jitter that matches the measured mean and standard deviation, truncated at the measured maximum. With five replicas, a leader's commit quorum costs on average 72.0 ms in NA-E, 100.5 ms in NA-W, 126.9 ms in EU, 100.6 ms in AP-NE and 162.5 ms in AP-SE, so the busiest region, EU, is far from the cheapest committer.

B. Geo-Temporal Workload

Public traces of geo-distributed databases with per-request client locations are not available, so we derive the workload from real, geolocated human activity. From a public ranking snapshot of popular GitHub repositories (github.com/EvanLi/Github-Ranking, October 22, 2026) we selected the 1,812 repositories active since September 1, 2026 and fetched their commit metadata with treeless shallow clones (1,769 succeeded). Of the 1,037,244 commits authored between June 28 and September 28, 2026, we removed automation accounts (10.9%) and mapped every remaining commit to the region implied by its author's UTC offset, a standard geolocation signal in software-repository mining [27]: offsets west of UTC−6:30 map to NA-W, UTC−6 to UTC−2 to NA-E, UTC−1 to UTC+4 to EU, UTC+4:30 to UTC+8 to AP-SE, and east of UTC+8 to AP-NE. The 440 repositories with at least 300 commits in the 13-week window starting on June 29 form the fleet, one Raft group per repository, with 838,178 commits. Every commit becomes a 30-minute session of database operations ending at the commit, commits per group, region and epoch are capped at ten to suppress scripted bulk commits (3.8% of the commits), and the trace is scaled to a mean of 20,000 operations per second, half of them writes. The session length, the scale and the write fraction are modeling choices; locations and timing are real.

Fig. 2b shows the resulting diurnal structure. EU issues 47.9% of all requests, AP-SE 20.7%, NA-W 16.2%, NA-E 11.3% and AP-NE 3.9%, and every region peaks in its local afternoon. Groups are mixed: the median group receives 77.4% of its requests from its dominant region, a quarter of the groups receive at most 57.7%, and the ten busiest groups carry 30.3% of the load. Fleet load averages 20,000 ops/s with a 95th percentile of 32,500 ops/s.

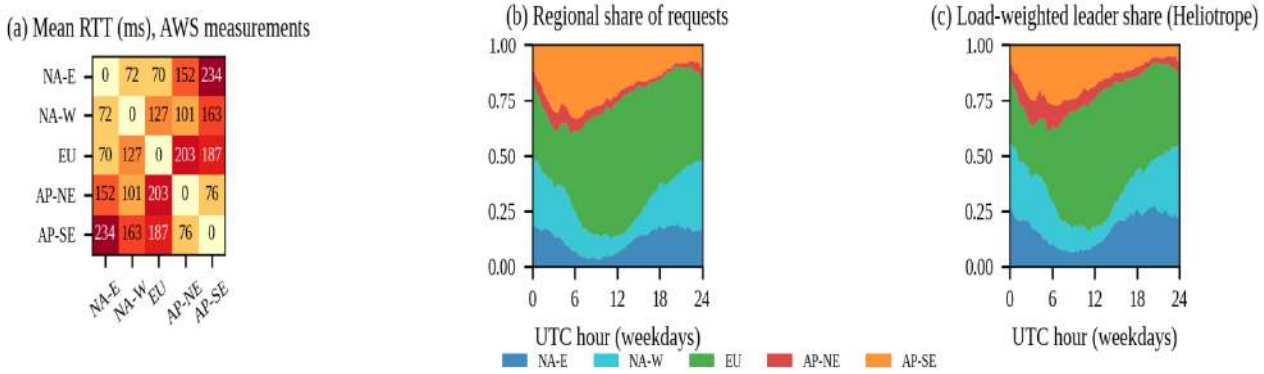


Figure 2Deployment and workload. (a) Mean RTT between the five AWS regions. (b) Weekday share of requests per region by UTC hour, derived from 838,178 commits. (c) Load-weighted share of leaders per region under Heliotrope during the test period: leaders follow

C. Evaluation Engine and Model Validation

The engine replays the trace in five-minute epochs. In every epoch it asks a policy for all leaders, accumulates the exact latency distribution of the epoch's operations as a demand-weighted mixture of the

oracle's distributions, adds transfer stalls according to (5), and optionally adds M/M/1 queueing at leaders. All statistics are computed from these mixtures on the 0.5-ms grid and therefore carry no sampling noise.

To check the oracle and the stall model, we built a message-level replay of Raft's replication path: every write is appended by the leader and shipped in its own AppendEntries message over FIFO links whose round trips are drawn from the measured distributions, followers acknowledge immediately, entries commit in log order once a majority holds them, and leadership transfer follows the TimeoutNow protocol with redirection of requests that reach the old leader. Table I compares the replay with the oracle for every leader region at 20 and 200 writes/s with the measured jitter and with five times that jitter. Means, medians and 99th percentiles agree within 0.7 ms, and the Wasserstein-1 distance never exceeds 0.91 ms, so in-order commit and FIFO coupling do not invalidate the independence assumption of the oracle. The replayed transfer windows match (5) within 0.4% on all five tested region pairs. The request time lost per transfer is 0.9–1.4 times the analytic estimate, because some clients keep contacting the old leader until they learn about the new one; at our transfer rates this changes fleet mean latency by less than 0.01 ms.

Table 1ORACLE VERSUS MESSAGE-LEVEL RAFT REPLAY (MAXIMUM OVER THE FIVE LEADER REGIONS, MS)

Setting	**|Δ mean|**	**|Δ p50|**	**|Δ p99|**	**W1**
Jitter ×1, 20 writes/s	0.73	0.37	0.28	0.91
Jitter ×1, 200 writes/s	0.42	0.63	0.52	0.59
Jitter ×5, 20 writes/s	0.40	0.30	0.38	0.53
Jitter ×5, 200 writes/s	0.74	0.32	0.41	0.54

D. Baselines

Static-Primary places all leaders in NA-E, which corresponds to the single-leader configuration. Static-Home places each leader in the group's dominant region during the four training weeks, and Static-Optimal places it at the quorum-aware optimum for the training-period mix. Hot-write-source (HWS) implements the feedback-driven placement that Ding and Qin [26] describe for the placement driver of TiKV [3]: in every epoch, a group's leader is transferred to the replica region that issued the most writes during the last w epochs whenever that region exceeds the leader's region by a ratio r. We report a default setting (w=6, i.e., 30 minutes, and r=1.2) and the best setting under the transfer budget. Myopic chooses in every epoch the leader that minimizes the mean-RTT commit model for the recently observed demand and switches only if the improvement exceeds a relative margin; it represents periodic latency-driven repositioning as in CD-Raft and damped switching as in GeoLM. Oracle is a clairvoyant per-group dynamic program over the true future demand whose transfer price is set by bisection to meet the budget; it ignores capacity coupling and serves as a lower bound.

E. Protocol and Metrics

Weeks 1–4 (June 29 to July 26) warm up the estimators and serve for tuning, and all selections use week 4 only. Weeks 5–13 (July 27 to September 27, 63 days) form the test period. The budget is B=2 transfers per group per day. The budget-matched baselines are the lowest-latency settings with at most B transfers on the tuning week (HWS: w=12, r=1.5; Myopic: half-life of two epochs and margin 0.3); unbudgeted settings are reported for reference. Heliotrope uses β=0.1, H=1, η=1, τ=200 ms and an initial price $c_T=3\times10^6$ request-ms. We report the mean, 95th and 99th percentile write latency, the fraction of writes above τ, the mean read latency, and transfers per group per day. Daily values are compared with the Wilcoxon signed-rank test over the 63 paired days, and 95% confidence intervals come from 5,000 bootstrap resamples (seed 20261022).

VI. Results

A. Five-Region Protocol Benchmark

We first re-instantiate the five-region comparison of Ding and Qin [26], which contrasts single-leader Raft, adaptive Multi-Raft, WPaxos with flexible quorums, and EPaxos under low and high conflict by their p50, p95 and p99 write-commit latency. We keep the five configurations, the three percentiles and the NA–EU–APAC topology, replace the illustrative RTTs by the measured distributions and the request stream by our trace, and add Multi-Raft driven by Heliotrope. WPaxos uses three nodes per zone, phase-2 quorums spanning the owner zone and one other zone, and object ownership that follows the hottest write source; EPaxos co-locates the command leader with the client and takes its slow path with probability 0.02 (low conflict) or 0.5 (high conflict). Fig. 3 shows the results. Adaptive Multi-Raft lowers the median from 142.9 to 127.2 ms relative to single-leader Raft, as in the original comparison, but its tail is worse (p99 of 349.4 versus 311.6 ms), because leaders that follow European writers expose clients in other regions to distant leaders. WPaxos attains the lowest median (70.5 ms) because its phase-2 quorum needs only the nearest other zone, at the price of tolerating fewer zone failures. Under measured geometry EPaxos behaves differently from an idealized setting: with low conflict it has the best p95 and p99 (162.9 and 253.8 ms), yet its median (127.0 ms) equals that of Raft, because a fast quorum formed from Europe must still reach the second-nearest replica 127 ms away; with high conflict its median rises to 162.7 ms. Heliotrope keeps Raft's majority quorums and fault tolerance and cuts the p95 of adaptive Multi-Raft from 313.3 to 254.4 ms and its p99 from 349.4 to 328.2 ms.

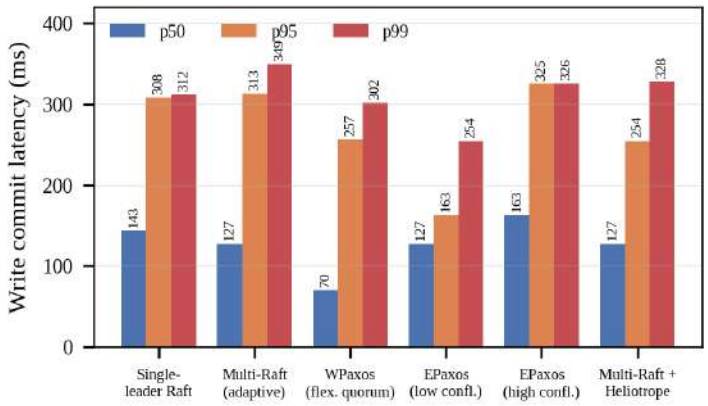


Figure 3Five-region write-commit latency of single-leader Raft, adaptive Multi-Raft, WPaxos and EPaxos with measured RTTs and the trace-driven request mix, extended with Multi-Raft driven by Heliotrope (test period).

B. Main Comparison

Table II summarizes the test period. Static placements are inadequate: the single-leader configuration violates the 200-ms threshold for 24.1% of writes and even the quorum-aware static optimum for 16.3%. Among the adaptive policies that respect the budget of two transfers per group per day, Heliotrope has the lowest mean (136.9 ms), p95 (254.4 ms), p99 (328.2 ms) and violation rate (7.4%). Static-Primary has a lower p99 (311.6 ms) only because all of its leaders sit in NA-E, whose quorum is the cheapest, at the price of a 34.0-ms higher mean and a 98.9-ms mean read latency. Compared with budget-matched Myopic, it reduces mean write latency by 3.0 ms (2.2%; 95% CI 2.9–3.2 ms) and violations by 18.4%; compared with HWS in its default setting, it lowers the mean by 3.2%, the p95 by 16.7% and violations by 20.2% while making fewer transfers. Heliotrope is better than every budget-matched baseline on all 63 test days (Wilcoxon $p \leq 5.2 \times 10^{-12}$). Its mean read latency, 19.1 ms, lies between those of budget-matched Myopic (19.8 ms) and default HWS (18.9 ms). The clairvoyant oracle remains 3.9 ms (2.9%) faster on writes, which bounds what better prediction could still gain.

Table 2TEST-PERIOD RESULTS (63 DAYS, AWS RTTS, FIVE REPLICAS)

Policy	W mean	W p95	W p99	Viol. %	R mean	Tr./g/d
Static-Primary	170.9	308.3	311.6	24.1	98.9	0.00
Static-Home	160.4	331.8	349.5	18.5	36.9	0.00
Static-Optimal	155.9	325.1	349.4	16.3	39.9	0.00
HWS (default)†	141.4	305.2	349.3	9.3	18.9	2.28
HWS (budget)	144.6	313.3	349.4	10.8	21.9	1.84
Myopic (budget)	139.9	263.7	349.2	9.1	19.8	1.80
Heliotrope	**136.9**	**254.4**	**328.2**	**7.4**	**19.1**	**2.01**
Oracle	*132.9*	*224.7*	*314.0*	*5.7*	*13.8*	*1.97*
HWS (no budget)†	137.0	263.2	349.1	7.6	15.5	2.77
Myopic (no budget)†	135.7	254.9	331.2	7.0	16.0	2.97

W = write latency, R = read latency (ms); Viol. = writes above 200 ms; Tr./g/d = leader transfers per group per day; † exceeds the budget of two transfers per group per day.

Fig. 4a–b plots the latency–transfer frontier of every family on the test period. Heliotrope's curve lies below both online baselines for budgets up to about 2.5 transfers per group per day, and HWS needs more transfers than Heliotrope at every latency level because it ignores quorum geometry. At larger budgets the curves converge: unbudgeted Myopic reaches 135.7 ms with 2.97 transfers, against 136.0 ms for Heliotrope with 2.95. Heliotrope is therefore most useful when transfers are scarce, which is the regime that operators enforce. Fig. 4c shows that the gains come from the tail: the CDFs nearly coincide below 150 ms, while Heliotrope places less probability mass between 250 and 350 ms than HWS and Myopic. Fig. 2c shows its leaders following the sun; NA-E and AP-NE host more leaders than their own demand share because their commit quorums (72 and 101 ms) are much cheaper than those of EU (127 ms) and AP-SE (163 ms).

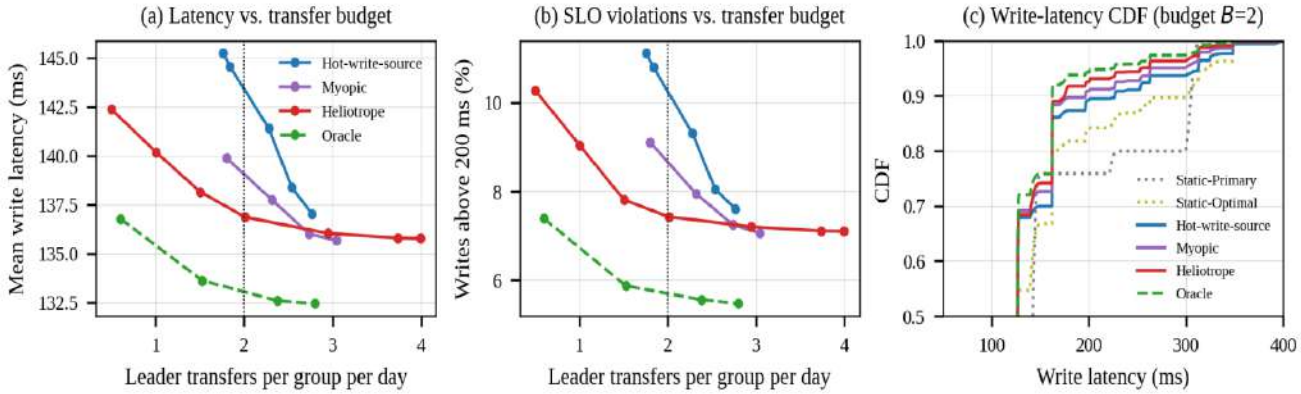


Figure 4Test-period results. (a) Mean write latency and (b) share of writes above 200 ms versus leader transfers per group per day for each policy family (dotted line: budget B=2). (c) Write-latency CDF at the budget.

C. Ablation

Table III removes one component at a time at the same budget. Replacing the quorum-aware oracle by client–leader proximity is the largest single loss: mean write latency grows by 2.4 ms and p99 to 349.2 ms, although reads become faster (17.0 ms) because they do not involve quorums. Dropping the SLO term mainly hurts the tail: p99 rises from 328.2 to 349.1 ms and violations from 7.4% to 7.6%, while reads become 1.1 ms faster. Without the seasonal prior, i.e., with the last observation alone, mean latency and violations are marginally lower (136.8 ms, 7.3%) but p99 grows to 334.6 ms; the prior mainly protects groups whose recent window is unrepresentative. Removing the transfer price raises transfers to 4.00 per group per day for a gain of 1.1 ms, so pricing retains most of the benefit with half the churn. Longer lookahead is harmful under the budget: H=3 and H=6 move leaders preemptively toward the seasonal pattern, which lowers p99 (314.2 and 313.9 ms) but raises mean write and read latency.

Table 3ABLATION AT THE SAME TRANSFER BUDGET (TEST PERIOD)

Variant	W mean	W p99	Viol. %	R mean	Tr./g/d
Heliotrope (full)	**136.9**	**328.2**	**7.4**	**19.1**	**2.01**
w/o seasonal prior	136.8	334.6	7.3	18.7	2.01
w/o SLO term	137.2	349.1	7.6	18.0	2.01
w/o quorum model	139.2	349.2	8.1	17.0	2.01
w/o transfer price†	135.8	325.2	7.1	17.2	4.00
lookahead H = 3	137.6	314.2	7.8	22.5	1.99
lookahead H = 6	140.9	313.9	9.5	30.5	1.17

D. Robustness

Table IV repeats the comparison with three replicas per group, placed in the two busiest regions of the training period plus the region that minimizes their quorum RTT, and with the five analogous GCP regions (us-east4, us-west1, europe-west1, asia-northeast1 and asia-southeast1) from a separate measurement campaign. With three replicas, Heliotrope again has the lowest mean write latency among the budgeted online policies (106.1 ms versus 108.8 ms for budget-matched Myopic) with 10.3% of writes above 200 ms (Myopic: 10.9%) and a p99 of 280.1 ms (Myopic: 309.4 ms), at 1.98 transfers per group per day. On GCP, Heliotrope again has the lowest mean write latency among the budgeted online policies (140.7 ms versus 144.6 ms for budget-matched Myopic) with 7.8% of writes above 200 ms (Myopic: 9.8%) and a p99 of 321.0 ms (Myopic: 434.4 ms), at 2.02 transfers per group per day. The oracle uses a fixed transfer price in these runs and therefore does not meet the budget exactly.

Table 4ROBUSTNESS TO REPLICATION FACTOR AND RTT CAMPAIGN (TEST PERIOD)

Policy	W mean	W p99	Viol. %	R mean	Tr./g/d
AWS, three replicas					
Static-Optimal	120.0	373.2	14.2	37.5	0.00
HWS (budget)	112.9	373.2	12.1	28.2	1.34
Myopic (budget)	108.8	309.4	10.9	25.6	1.43
Heliotrope	**106.1**	**280.1**	**10.3**	**22.9**	**1.98**
Oracle	*103.4*	*263.7*	*9.4*	*20.1*	*1.68*
GCP, five replicas					
Static-Optimal	158.8	432.1	17.0	45.5	0.00
HWS (budget)	151.5	441.5	12.4	26.0	1.84
Myopic (budget)	144.6	434.4	9.8	23.2	1.86
Heliotrope	**140.7**	**321.0**	**7.8**	**24.5**	**2.02**
Oracle†	*135.9*	*297.5*	*5.9*	*16.6*	*2.33*

E. Capacity Stress Test

Finally, we cap the leader capacity of every region at μ_r = 16,000 and 12,000 operations per second with a planning utilization of 0.8, and we add M/M/1 queueing at leaders, over the first two test weeks; at peak, placements that follow demand put more than half of the fleet load on EU. Baselines first decide as before and are then repaired either by a latency-oblivious balancer in the style of placement-driver leader balancing, which moves random leaders out of overloaded regions to the least-loaded replica region, or by a latency-aware greedy repair. Heliotrope plans with its predicted loads inside the assignment. Table V shows that Heliotrope overloads a region in at most 0.02% of the epochs, fewer than any other policy, and has the lowest mean write latency at both capacities (139.0 and 142.4 ms, against 142.3 and 155.2 ms for the best repaired baseline), while the balancers trade latency for many extra transfers. Without its capacity constraint Heliotrope overloads EU in 9.0% and 23.4% of the epochs and its p95 exceeds 1.9 s, which shows that capacity must be part of the decision rather than a post-processing step. On 16 capacity-binding epochs sampled from the 12,000-ops/s run, the Lagrangian solution was on average within 2.55% (maximum 11.65%) of the exact MILP optimum while taking 12.2 ms instead of 531 ms; at this fleet size the exact solver is affordable and preferable, and the Lagrangian method serves as a fallback for much larger fleets. On real instances tiled to 44,000 groups the Lagrangian method needs 14.79 s per epoch, whereas the MILP takes 3.6 s at 4,400 groups.

Table 5LEADER CAPACITY STRESS TEST WITH LEADER QUEUEING (FIRST TWO TEST WEEKS)

Policy	W mean	W p99	Viol. %	Overl. %	Tr./g/d
μ = 16,000 ops/s					
HWS + PD balance	152.7	349.5	15.1	0.02	8.09
Myopic + PD balance	148.6	349.3	12.4	0.30	9.67
Myopic + greedy repair	142.3	349.3	9.3	0.12	3.49
Heliotrope w/o capacity	274.9	≥2000	15.4	8.95	1.98
Heliotrope	139.0	325.4	7.8	0.00	1.99
μ = 12,000 ops/s					

HWS + PD balance	159.2	349.7	19.0	0.10	13.60
Myopic + PD balance	159.3	349.9	16.2	1.02	19.84
Myopic + greedy repair	155.2	395.0	11.4	1.31	5.74
Heliotrope w/o capacity	465.7	≥2000	25.1	23.36	1.98
Heliotrope	142.4	325.2	10.4	0.02	2.05

Overl. = epochs in which some region exceeds 98% leader utilization; latencies include queueing; ≥2000 marks values beyond the 2-s evaluation grid.

VII. Discussion and Limitations

Our evaluation replays measured RTTs and real activity through an exact analytic engine that we validated against a message-level replay, rather than running a database across cloud regions; production effects such as CPU contention, batching and garbage-collection pauses lie outside the model. The ping-based RTT distributions are light-tailed, so the tails in our results arise mainly from geography, and heavier-tailed links would make distribution-aware scoring more valuable. Commits are a proxy for database activity: their locations and timing are real, whereas the session length, the scale and the write fraction are assumptions. Transfers have secondary costs, such as cold caches and lease handover, that we capture only through the budget. Finally, the SLO term trades about 1.1 ms of mean read latency for a lower write tail; operators who value reads more can lower η.

VIII. Conclusion

Heliotrope treats leader placement in Raft-based geo-distributed databases as a fleet-level, budgeted and SLO-aware decision problem. Exact quorum order statistics expose why busy regions are not always good leaders, an empirical-Bayes demand estimate tames sparse per-group signals, and a priced, capacity-aware assignment spends scarce transfers where they matter. On a 13-week trace of real geo-temporal activity with measured RTTs, Heliotrope outperforms traffic-following and myopic latency-driven placement whenever transfers are budgeted, nearly eliminates leader overload when leader capacity is tight, and stays within 3.9 ms of a clairvoyant oracle. Future work includes running the controller inside a production placement driver, jointly optimizing replica placement, and learning RTT dynamics online.

References

J. C. Corbett *et al.*, “Spanner: Google’s globally-distributed database,” in *Proc. 10th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2012, pp. 251–264.

R. Taft *et al.*, “CockroachDB: The resilient geo-distributed SQL database,” in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2020, pp. 1493–1509, doi: 10.1145/3318464.3386134.

D. Huang *et al.*, “TiDB: A Raft-based HTAP database,” *Proc. VLDB Endowment*, vol. 13, no. 12, pp. 3072–3084, 2020.

L. Lamport, “The part-time parliament,” *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998, doi: 10.1145/279227.279229.

D. Ongaro and J. Ousterhout, “In search of an understandable consensus algorithm,” in *Proc. USENIX Annu. Tech. Conf. (ATC)*, 2014, pp. 305–319.

D. Xu *et al.*, “GeoLM: Performance-oriented leader management for geo-distributed consensus protocol,” in *Proc. IEEE Conf. Computer Communications (INFOCOM)*, 2025, pp. 1–10.

Y. Wang, Z. Cheng, Y. Dong, and Z. Xu, “CD-Raft: Reducing the latency of distributed consensus in cross-domain sites,” arXiv:2603.10555, 2026, doi: 10.48550/arXiv.2603.10555.

A. Sharov, A. Shraer, A. Merchant, and M. Stokely, “Take me to your leader! Online optimization of distributed storage configurations,” *Proc. VLDB Endowment*, vol. 8, no. 12, pp. 1490–1501, 2015.

D. Ongaro, “Consensus: Bridging theory and practice,” Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, USA, 2014.

X. Yue and R. Zhu, “An explainable hybrid machine learning framework for cash-flow-conditioned multi-horizon liquidity risk early warning: An SME-oriented benchmark study,” World Journal of Management Science, vol. 2, no. 1, pp. 63–72, 2024.

I. Moraru, D. G. Andersen, and M. Kaminsky, “There is more consensus in egalitarian parliaments,” in *Proc. 24th ACM Symp. Operating Systems Principles (SOSP)*, 2013, pp. 358–372, doi: 10.1145/2517349.2517350.

H. Howard, D. Malkhi, and A. Spiegelman, “Flexible Paxos: Quorum intersection revisited,” in *Proc. 20th Int. Conf. Principles of Distributed Systems (OPODIS)*, LIPIcs, vol. 70, 2016, pp. 25:1–25:14.

A. Ailijiang, A. Charapko, M. Demirbas, and T. Kosar, “WPaxos: Wide area network flexible consensus,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 1, pp. 211–223, 2020.

V. Enes, C. Baquero, T. F. Rezende, A. Gotsman, M. Perrin, and P. Sutra, “State-machine replication for planet-scale systems,” in *Proc. 15th European Conf. Computer Systems (EuroSys)*, 2020, pp. 24:1–24:15, doi: 10.1145/3342195.3387543.

X. Yan, L. Yang, and B. Wong, “Domino: Using network measurements to reduce state machine replication latency in WANs,” in *Proc. 16th Int. Conf. Emerging Networking Experiments and Technologies (CoNEXT)*, 2020, pp. 351–363, doi: 10.1145/3386367.3431291.

A. Ailijiang, A. Charapko, and M. Demirbas, “Dissecting the performance of strongly-consistent replication protocols,” in *Proc. ACM SIGMOD Int. Conf. Management of Data*, 2019, pp. 1696–1710.

M. S. Ardekani and D. B. Terry, “A self-configurable geo-replicated cloud storage system,” in *Proc. 11th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2014.

M. Annamalai *et al.*, “Sharding the shards: Managing datastore locality at scale with Akkio,” in *Proc. 13th USENIX Symp. Operating Systems Design and Implementation (OSDI)*, 2018.

P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica, “Probabilistically bounded staleness for practical partial quorums,” *Proc. VLDB Endowment*, vol. 5, no. 8, pp. 776–787, 2012.

J. Dean and L. A. Barroso, “The tail at scale,” *Commun. ACM*, vol. 56, no. 2, pp. 74–80, 2013, doi: 10.1145/2408776.2408794.

A. Borodin, N. Linial, and M. E. Saks, “An optimal on-line algorithm for metrical task system,” *J. ACM*, vol. 39, no. 4, pp. 745–763, 1992, doi: 10.1145/146585.146588.

M. Lin, A. Wierman, L. L. H. Andrew, and E. Thereska, “Dynamic right-sizing for power-proportional data centers,” *IEEE/ACM Trans. Netw.*, vol. 21, no. 5, pp. 1378–1391, 2013, doi: 10.1109/TNET.2012.2226216.

P. R. Winters, “Forecasting sales by exponentially weighted moving averages,” *Manage. Sci.*, vol. 6, no. 3, pp. 324–342, 1960, doi: 10.1287/mnsc.6.3.324.

M. L. Fisher, “The Lagrangian relaxation method for solving integer programming problems,” *Manage. Sci.*, vol. 27, no. 1, pp. 1–18, 1981, doi: 10.1287/mnsc.27.1.1.

Q. Huangfu and J. A. J. Hall, “Parallelizing the dual revised simplex method,” *Math. Program. Comput.*, vol. 10, no. 1, pp. 119–142, 2018.

J. Ding and Y. Qin, “Raft and beyond: Practical consensus mechanisms for geo-distributed data systems,” *Computer Life*, vol. 14, no. 1, pp. 54–63, 2026.

F. Zhou, “Moment-constrained side-force allocation for wind-assisted merchant ships: Hydrodynamic performance of retractable anti-leeway fins and flap rudders,” Innovation and Technology Studies, vol. 3, no. 1, pp. 27–41, 2026.

J. Ding and D. Teng, “A Risk-Aware Digital Twin Framework for Release-Readiness Validation of Multi-Tenant Zero Trust SASE Systems,” WEP Transactions on Computer Science and Intelligent Systems Research, vol. 13, pp. 247–259, 2026.

Y. Wang, X. Zou, Z. Lu, K. Wei, and M. Rhee, “PRESAGE: Uncertainty-Aware Predictive Elastic Scheduling for Cloud Analytics and GenAI Inference,” IEEE Access, early access, 2026, doi: 10.1109/ACCESS.2026.3734498.