

Research on Real-time Data Synchronization Technology for Database Migration Based on CDC

Dong Shao

Pepperdine Graziadio Business School, Pepperdine University, Malibu, 90263, California, USA

Keywords: Change data capture; Database migration; Real-time data synchronization; Online micro-batch processing; Transaction age; Queue optimization

Abstract: To address the challenge of balancing incremental backlog, transaction freshness, and commit overhead in online database migration, an age-constrained online micro-batch synchronization method, AOMB, is proposed. This method uses persistent complete CDC transactions as scheduling units, leverages recursive least squares to estimate the target server's service capacity, and combines a physical backlog queue, a virtual transaction age queue, and a Binlog switching barrier to select batches. Controlled-repetition experiments were conducted on a MySQL-Debezium-PostgreSQL link, and comparisons were made with fixed-batch, queued AIMD, MPA-TB, and offline Oracle methods. Results show that AOMB reduces switching time by 8.1%–10.9% and transaction age violation rate by 35.5%–39.6% under three types of unknown loads. However, its advantages diminish with increasing commit barrier wait times under high hotspot and capacity boundary conditions. The study demonstrates that this method can improve the switching timeliness of CDC migration, but its effectiveness is constrained by service margin and conflict levels.

1 Introduction

Database migration is shifting from offline transport to online migration. During the replication of existing data, the source database will continue to generate new, modified and deleted transactions. The target database needs to receive these changes synchronously and reduce incremental backlog and data obsolescence as soon as possible before switching. Modern stream processing systems have the ability to continuously capture, maintain state and recover from failures [1]. Change data capture based on transaction logs can also output row-level changes in a less intrusive way.

The target end commit strategy directly affects the migration catching-up efficiency. Per-transaction commit can reduce batch filling wait, but it will repeatedly generate network round trips, constraint checks and write-ahead log flushing; micro-batch can amortize fixed costs, but may increase waiting in low load phases because batches are difficult to fill in time. Transaction flow research usually takes throughput, tail latency and transaction semantics as common constraints[2], and the parallel scheduling effect is also affected by load characteristics, resource conditions and commit boundaries[3]. Online migration also needs to complete backlog clearing, source end rate

limiting, short pause write and log barrier catching-up. The system can only enter a verifiable switching state after the correctness is guaranteed.

Existing research has improved the performance of online migration and streaming from multiple directions. Related work focuses on downtime and transmission overhead [4], and also distinguishes between the resource targets of full initialization and incremental processing [5]; micro-batch research mainly deals with commit and checkpoint latency [6], while elastic streaming processing adjusts resources or event grouping according to the input rate [7]. Most of these methods use queue length, average latency or resource utilization as feedback, and have not yet incorporated the freshness of switching transactions and log barriers into a unified constraint. The previously proposed MPA-TB relies on empirical thresholds to identify the migration stage, and the stability boundary is not clear enough. Its effect may also be affected by development load and the selection of baseline parameters.

Based on the above issues, this study limits the boundary to unidirectional incremental CDC catching up after a full snapshot is completed, and the target database does not undertake read operations before the formal switchover. A safe switchover requires three conditions to be met simultaneously: the target application location crosses the Binlog barrier generated when writing stops at the source end; all transactions before the barrier are completed; and the transaction age service level (SLO) violation rate remains under control. Based on this, the study proposes an age-constrained online micro-batch synchronization method, focusing on answering two key questions: Can AOMB approach the optimal offline switchover time under the same constraints under unknown loads? And when capacity is feasible and the migration window is limited, can AOMB control the transaction-level age SLO violation rate within a specified range? The study validates this method through experiments using a complete fixed batch frontier, MPA-TB, online baseline, offline Oracle, and high-concurrency hotspots.

2 Research Methods

2.1 Research Design and System Boundary

The study employs a controlled, repetitive system experiment with a pre-defined protocol. The Debezium Embedded Engine reads the MySQL Binlog, the transaction assembler restores the complete source transaction, and the target application writes to PostgreSQL according to the source commit sequence. The experimental subjects are unidirectional CDC transaction flows generated after a full snapshot. The system does not handle DDL transformations, stored procedures, bidirectional synchronization, or pre-switch read traffic; triggers, external side effects, and transactions containing non-deterministic functions are also excluded from the sample to ensure verifiable final state equivalence.

Figure 1 shows the system structure. The complete transaction, source location and characteristics are written to the persistent queue by the same RocksDB WriteBatch; after the write to disk is completed, the system advances the capture location ℓ_t^{cap} . Business changes, source transaction ID and application location ℓ_t^{app} are committed in the same PostgreSQL transaction. After successful commit, the system deletes the queue record and keeps it $\ell_t^{app} \leq \ell_t^{cap}$. Each batch has a global sequence number, and after multi-threaded parallel execution, the commit order is restored by the commit barrier [8]. Transaction conflict keys are defined as follows $K(T_i) = \{(table, primary\ key)\}$; transactions with intersecting key sets enter the same ordered channel, and table-level conflict keys are used for tables without primary keys and range operations [9].

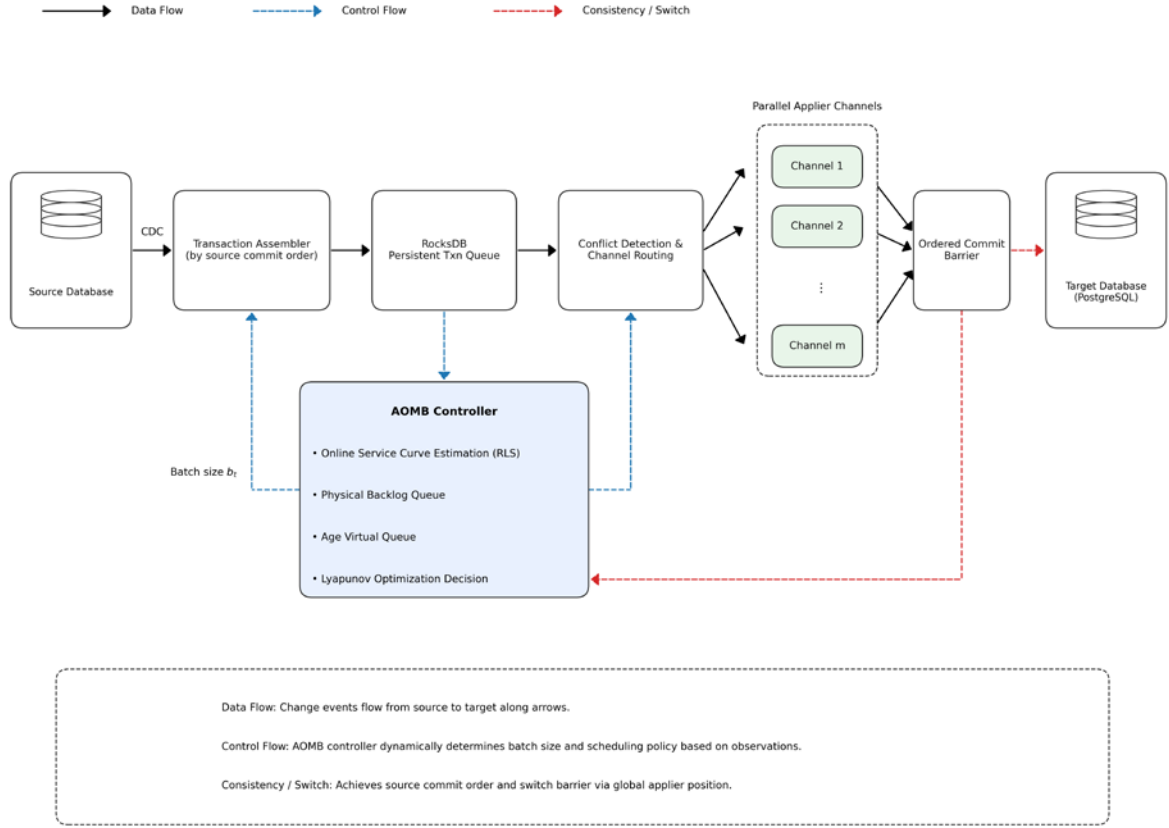


Figure 1. CDC Online Micro-batch Synchronization Architecture Based on Transaction Age Constraints

2.2 Queues, Transaction Age, and Switching Barriers

Let the control period be Δt , t the number of complete transactions entering the persistent queue within the period be N_t , the batch decision be b_t , the number of transactions actually completed on the target end be $S_t(b_t)$, and the number of transactions pending processing be Q_t , then

$$Q_{t+1} = \max\{Q_t - S_t(b_t), 0\} + N_t.$$

the transaction i in the source and target databases are t_i^c and , respectively t_i^a , and the end-to-end transaction age is :

$$G_i = t_i^a - t_i^c.$$

The system receives the switchover request at the t_c and throttles the source; the source records a Binlog barrier, ρ^{cut} , during a brief pause in writing. Each transaction is counted only once upon crossing the threshold: a transaction enters the "determined" set either when it completes while its age is no greater than τ , or when it crosses τ for the first time. Let M_t denote the number of newly determined transactions in period t , and X_t denote the number of transactions among them that cross the threshold for the first time. This statistical approach avoids the double-counting of the same over-age transaction across multiple cycles. The system simultaneously maintains a count, P_t^{cut} of transactions that remain incomplete prior to the barrier. The switch-ready time is defined as the earliest moment t at which the following conditions are met and continuously maintained for

K cycles:

$$T_{ready} = \inf\{t \geq t_c: \ell_t^{app} \geq \ell_t^{cut}, P_t^{cut} = 0, R_t^{(W)} \leq \rho\}$$

Here, $R_t^{(W)}$ represents the transaction-level age violation rate for the most recent period. The physical low watermark is only used to trigger a write halt at the source end and is not used as a sole criterion for migration completion.

2.3 Pre-decision estimation of observable service capacity

The transaction assembler records the number of row-level events e_i , the number of serialized payload bytes z_i , and the percentage of update and delete events before a transaction enters RocksDB u_i . At the start of each cycle, the controller only reads the transactions already persisted at the head of the queue. For candidate batches $b \leq Q_t$, the system aggregates the previous b transactions to obtain $e_t(b)$, $z_t(b)$ and $u_t(b)$; at that point, the actual candidate quantity is truncated to $Q_t < bQ_t$, and the unfilled portion is separately included in the fill wait estimate.

Let the batch characteristics be $x_t(b) = [1, b, b^2, e_t(b), z_t(b), u_t(b)]^T$ and the batch time estimate be :

$$\hat{d}_t(b) = x_t(b)^T \hat{\theta}_t.$$

The model parameters are updated using recursive least squares with a forgetting factor of 0.985. At system startup, stratified probing is performed on each candidate batch; during operation, every 20 periods, the adjacent batch with the widest confidence interval is detected when there are no age warnings. The controller $\bar{d}_t(b)$ calculates a conservative service volume using the 95% upper confidence bound of the service time and rounds it down:

$$\underline{S}_t(b) = \frac{\Delta t b}{\bar{d}_t(b)}.$$

The candidate set is fixed $\mathcal{B} = \{1, 2, 4, 8, 16, 32, 48, 64, 96\}$. A conservative estimate error is allowed for the actual service volume ξ_t :

$$S_t(b) \geq \underline{S}_t(b) - \xi_t, \quad \limsup_{T \rightarrow \infty} \frac{1}{T} \sum_{t=0}^{T-1} \xi_t \leq \bar{\xi}.$$

Model quality was tested using rolling R^2 , mean absolute percentage error, upper confidence boundary coverage, and perturbation window error.

2.4 Virtual queue of transaction ages and online decision-making

To constrain transaction-level violation rates, an age virtual queue is constructed:

$$Z_{t+1} = \max\{Z_t + X_t - \rho M_t, 0\}.$$

When Q_t is large, the controller needs to increase the conservative service volume; when Z_t continues to increase, the controller needs to reduce the fill-up waiting time. Defining $L_t = (Q_t^2 + Z_t^2)/2$, AOMB makes a selection in each cycle:

$$b_t^* = \operatorname{argmin}_{b \in \mathcal{B}_t} \{VC_t(b) - Q_t \underline{S}_t(b) + Z_t \hat{X}_t(b)\},$$

The number of transactions expected to exceed the age limit for the first time in the next cycle $\mathcal{B}_t$ is $C_t(b)$ composed of the commit time and the expected fill wait time, $\hat{X}_t(b)$ excluding candidate batches that exceed the memory, transaction size and commit barrier capacity.

Proposition 1 states that if arrival volume and service volume are bounded, and there exists a stationary strategy that makes the actual average service volume higher than the average arrival volume $\epsilon > \bar{\xi}$ and the transaction age violation rate lower than the average $\rho - (\epsilon - \bar{\xi})$, then Q_t it is Z_t strongly stable; the difference between the long-run average cost and the optimal stationary strategy is given by $O(1/V) + O(\bar{\xi})$, and the average queue is given by $O(V)$. By accumulating the virtual queue update relationship, a finite-time domain relationship can be obtained:

$$\frac{\sum_{t=0}^{T-1} X_t}{\sum_{t=0}^{T-1} M_t} \leq \rho + \frac{Z_T - Z_0}{\sum_{t=0}^{T-1} M_t}.$$

This theoretical result demonstrates that even with insufficient capacity margin or increased model error, short-term overruns can still occur within a finite transition window. The proposition provides stability and violation rate boundaries within the feasible capacity domain; the switching time benefit will be verified through subsequent experiments.

2.5 Transaction semantics, persistent queues, and fault recovery

AOMB treats complete source transactions as the smallest scheduling unit, and no transaction is split across batches. Multiple consecutive source transactions can be visible simultaneously in the same target batch. The system therefore guarantees atomicity within source transactions, source commit order, and final state equivalence, but does not guarantee one-to-one intermediate visibility for each source transaction.

The failure model covers four positions: before and after a RocksDB write, during the execution of the target transaction, after PostgreSQL commit but before queue record deletion, and after queue record deletion. When a RocksDB write fails, the capture position is not advanced; when the target transaction fails, the entire PostgreSQL batch is rolled back. If the target transaction has already committed but the queue record has not been deleted, the recovery program will read the relevant transaction again, but the unique constraint of the source transaction ID in the audit table will convert the replay into a side-effect-free confirmation. During system recovery, the capture position, application position, and barrier position are checked simultaneously. This design achieves an exact-once application effect, independent of distributed atomic commits between MySQL, Debezium, RocksDB, and PostgreSQL. Stateful stream recovery also requires that the processing state remain consistent with the input position.

2.6 Comparison Methods with Offline Oracle

The experiment included five comparative methods: all fixed batches, queued AIMD, the early-stage empirical method MPA-TB, AOMB, and offline Oracle. The validation set used fixed batches representing parameters that could be determined before deployment; the test set reported the frontier of all fixed batches, and used the post-test optimal fixed batch as the diagnostic upper bound. MPA-TB used parameters determined during the development phase and did not access test data.

Oracle and AOMB use the same batch sets, concurrency, maximum wait times, sequence barriers, and transaction semantics, with Oracle only needing additional information on future arrival sequences and actual service times. The FIFO age under a fixed trajectory is determined by the cumulative arrivals and completions, denoted as $s_t = (n_t, Z_t)$. The stage cost is written as:

$$g_t = \mathbb{I}[t < T_{ready}] + \kappa C_t(b) + \eta \max\{X_t - \rho M_t, 0\}.$$

The corresponding finite-time dynamic programming is written as:

$$J_t(s) = \min_{b \in \mathcal{B}_t} \{g_t(s, b) + J_{t+1}(F_t(s, b))\}.$$

The experimental time domain is 420 s, with 10H termination costs added if the action is not ready. The dynamic programming complexity is $O(HN|\mathcal{B}|)O(\log n)$. Oracle does not relax SLO, sequence, and failure constraints; the result only represents the offline optimal sequence for a given set of actions and the experimental time domain.

2.7 Data, Experimental Environment and Statistical Methods

The main experiment used BenchBase TPC-C, scaled to 4 warehouses, with an initial incremental backlog of 30,000 complete transactions. The source and target databases were deployed on fixed servers, each configured with an 8-core Intel Xeon E-2288G processor, 64 GB of RAM, and a 1 TB NVMe SSD. The synchronizer ran on a dedicated node with 4 cores and 16 GB of RAM. The main experiment used 2 target application threads and 2 connections, with a node benchmark round-trip latency of approximately 0.3 ms. Table 1 lists the main parameters.

Table 1 Experimental Environment and Research Parameters

Project	Set up	Project	Set up
Source library/target library	MySQL 8.0/PostgreSQL 16	CDC	Debezium 3.6.0.Final
Main load	TPC-C, 4 warehouses	initial backlog	30,000complete transactions
Batch Candidate	1, 2, 4, 8, 16, 32, 48, 64, 96	Maximum wait	300 ms
Age SLO	$\tau=1$ s	Permissible violation rate	$\rho=6\%$
Main application concurrency	2 threads, 2 connections	Number of repetitions	8 times/method/scenario
Service estimation	RLS, forgetting factor 0.985	Statistical methods	Pair Bootstrap 2000 times

Development, validation, and testing data used non-overlapping random seeds. Test loads included two-state Markov-modulated Poisson processes, truncated Pareto ON/OFF long burst processes, and TPC-C loads driven by real arrival intervals. Real-world tracks were derived from 2025 New York City yellow taxi pick-up timestamps. Data cleaning removed missing, out-of-year, and reverse-order records, retaining only duplicate events within the same second; eight non-overlapping 12-hour weekday windows corresponded to eight tests each. Scaling was determined solely based on the development set capacity. Let $\bar{\lambda}_{raw}$ be the original average arrival rate, and let $\mu_{dev} = 203$ transactions/s be the conservative lower bound of the service rate observed during the development phase. With the target average rate fixed at $\lambda_{tar} = 0.70\mu_{dev} = 142.1$ transactions/s, the scaling factor is computed as $s = \lambda_{tar}/\bar{\lambda}_{raw}$. The original inter-arrival intervals Δ_i are then scaled to $\Delta_i' = \Delta_i/s$ to generate the target workload. After scaling, the average arrival rate for each 420-s window was 142.1 transactions/s, P99 was 316 transactions/s, and the coefficient of variation was 0.76. All methods used the same window, scaling, and transaction seed. Public tracks only provided arrival intervals; transaction content was still generated by TPC-C.

The extended experiments examined 1, 2, 4, and 8 application threads, and network round-trip

latency of 0.3, 5, and 10 ms. High-concurrency experiments combined 8 or 16 warehouses with 8 threads, and set three key distributions: uniform, Zipf 0.8, and Zipf 1.2. The constant-rate group fixed the total arrival rate at 142.1 transactions/s, while the capacity group increased the input of 16 warehouses to 205 transactions/s. Inherent conflict rate was calculated using a fixed 200 ms window, and runtime conflict rate was statistically analyzed based on the overlap of in-transit transaction key sets. Barrier wait percentage was calculated as the ratio of ordered commit wait time to total batch service time, and channel load was measured using the coefficient of variation of transaction count. Continuous metrics included median, interquartile range, empirical cumulative distribution, and paired Bootstrap 95% confidence interval. Fault tolerance experiments injected data 8 times at each of the four types of fault locations.

3. Research Results

3.1 Switching between ready time and transaction age constraints

Table 2 reports the main test results. Under all three load types, AOMB reached the switchover condition earlier than the post-test optimal fixed batch, and also had a lower transaction age violation rate. The upper quartile bound in the Pareto scenario still exceeded 6%. Compared with Oracle, AOMB's switchover time difference remained at 8–9 seconds.

Table 2. Results of different methods under test load, median [interquartile range]

Scene	method	Switching time/s	Age violation rate / %	Backlog area / (10 ⁶ transactions/s)	Number of commits per 10,000 transactions
MMPP	Deployment fixed	118 [102, 137]	12.8[10.1,16.4]	5.91 [5.34, 6.62]	358 [337, 389]
	Posterior optimal fixed	105 [91, 123]	8.2 [6.4, 10.7]	6.02 [5.45, 6.78]	514 [470, 572]
	AIMD queue	110 [95, 130]	8.9 [7.0, 11.8]	6.25 [5.68, 7.11]	562 [514, 625]
	MPA-TB	104 [90, 122]	7.2 [5.6, 9.5]	6.08 [5.51, 6.86]	530 [486, 588]
	AOMB	95 [82, 112]	5.0 [3.9, 6.7]	5.99 [5.42, 6.73]	491 [451, 548]
	Offline Oracle	87 [75, 103]	4.0 [3.1, 5.4]	5.74 [5.19, 6.44]	463 [427, 509]
Pareto	Deployment fixed	133 [115, 156]	15.4[12.3,19.7]	6.59 [5.96, 7.42]	369 [346, 405]
	Posterior optimal fixed	119 [102, 141]	9.6 [7.4, 12.6]	6.74 [6.09, 7.63]	529 [484, 590]
	AIMD queue	125 [107, 149]	10.8[8.4, 14.2]	6.95 [6.28, 7.89]	584 [531, 651]
	MPA-TB	116 [99, 138]	8.4 [6.5, 11.2]	6.82 [6.16, 7.71]	547 [500, 607]
	AOMB	106 [91, 127]	5.8 [4.4, 7.8]	6.67 [6.01, 7.56]	505 [462, 563]
	Offline Oracle	98 [84, 117]	4.7 [3.6, 6.2]	6.39 [5.77, 7.21]	477 [439, 526]
Real arrival	Deployment fixed	109 [95, 127]	10.7 [8.5, 13.8]	5.18 [4.71, 5.79]	354 [333, 384]
	Posterior optimal fixed	99 [86, 116]	7.6 [5.9, 9.9]	5.32 [4.83, 5.95]	501 [459, 557]
	AIMD queue	102 [88, 120]	8.1 [6.3, 10.6]	5.47 [4.96, 6.13]	548 [501, 610]
	MPA-TB	97 [84, 114]	6.8 [5.2, 9.0]	5.39 [4.89, 6.03]	520 [477, 579]
	AOMB	91 [79, 107]	4.9 [3.8, 6.4]	5.28 [4.79, 5.91]	482 [443, 537]
	Offline Oracle	84 [73, 99]	4.1 [3.1, 5.5]	5.09 [4.62, 5.69]	456 [420, 501]

AOMB did not simultaneously reduce all cost metrics. Its improved switching time was accompanied by an increase in the number of target submissions, while the backlog area changed only slightly. The worst-case performance in all three scenarios did not show an order-of-magnitude degradation.

3.2 Fixed Batch Frontier, Distribution Results, and Online Behavior

Figure 2 (top left) shows the backlog and batch trajectories for 8 runs under Pareto load. The solid line represents the median, and the shaded area represents the quartile interval. The bottom left, top right, and bottom right sections show the cumulative distribution of switchover time experience, the cumulative distribution of complementary transaction ages, and the leading edge of all fixed batches in the test set, respectively. As the batch size increases from 1 to 16, the switchover time continuously decreases; further increasing the batch size leads to an increase in age violation rate and lock wait. When network round-trip latency increases, the leading edge inflection point shifts from 16 to 32 or 48. Under all three load types, AOMB is not simultaneously dominated by a single fixed batch; Oracle remains to its bottom left.

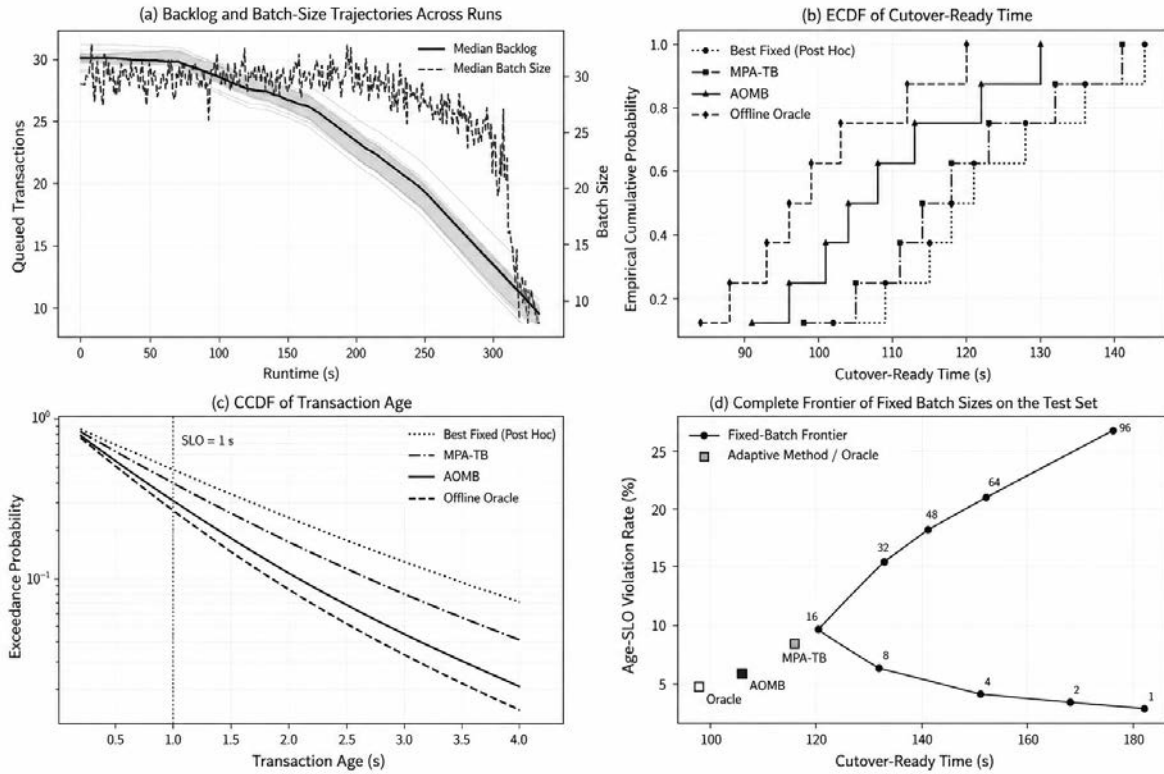


Figure 2. Distribution of multiple runs, tail of transaction age, and front of fixed batch.

The service model had a rolling R^2 of 0.87 [0.80, 0.92], a mean absolute percentage error of 8.9% [7.2%, 11.4%], and 95% upper confidence boundary coverage of 93.6% [91.2%, 96.1%]. During checkpoints and garbage collection, the model error rose to 15.8%, and the controller reconverged after 6–13 cycles. Each candidate batch received at least 7 observations in each run.

3.3 Parameters, Heating Points, Networks, and Fault Results

Table 3 reports the experimental results for parameters, hotspots, network, and faults. When V ranges from 12 to 25, the changeover time does not exceed 7.5%. In the constant-rate group,

changes in warehouse size do not lead to an increase in inherent conflicts; when hotspots are enhanced, runtime conflict rate and barrier wait increase simultaneously. In the capacity group, barrier wait accounts for 43.8% of batch service time, and the changeover time difference between AOMB and the posterior fixed batch is +2.2%.

Table 3. Parameters, Model Diagnostics, Extensibility, and Fault Tolerance Results

experiment	set up	Main results
V sensitivity	8/12/18/25/36	Pareto switching times: 110/107/106/109/117 seconds; violation rates: 4.3%/5.0%/5.8%/6.4%/7.5%
Service Model	Full Run/Disturbance Window	$R^2=0.87$; MAPE 8.9%/15.8%; Upper bound coverage 93.6%; Mean error 1.7 transactions/cycle
8-thread equal-speed conflict isolation	8W uniform / Zipf 0.8 / Zipf 1.2	AOMB 111 [100, 126]/135 [120, 156]/158 [141, 181]s; Advantages 5.9%/2.9%/0.4%; Inherent conflicts 12.4%/28.7%/40.9%; Runtime conflicts 17.6%/34.8%/46.5%; Barriers 14.5%/27.2%/34.8%
8-thread constant speed scale comparison	16W uniform / Zipf 0.8 / Zipf 1.2	AOMB 104 [94, 119]/128 [114, 148]/151 [136, 174]s; Advantages 6.7%/3.3%/0.6%; Inherent conflicts 8.1%/25.5%/38.8%; Runtime conflicts 12.7%/31.4%/44.7%; Barriers 10.9%/24.6%/33.1%
8-thread capacity limit	16W Zipf 1.2, 205 transactions/s	AOMB 234 [209, 269]s; Advantage -2.2%; Runtime conflict 56.3%; Barrier 43.8%; Channel load CV 0.47
Network latency	RTT 0.3/5/10 ms	Posterior optimal fixed batch size: 16/32/48; AOMB: 106/124/143 s, fixed: 119/138/158 s.
Fault recovery	32 times, 4 types of positions	Recovery time: 9.4 [7.6, 13.5] s; re-deployment of 24–103 transactions; duplicate business effect: 0; position invariants not violated.

All 32 fault injection attempts passed source-target row count, primary key set, and block hash verification. The fault occurring after the target transaction was committed but before queue record deletion resulted in the most re-injections. Three instances of insufficient disk space caused consecutive rollbacks; after releasing space, the system recovered from the original application location in each case. No instances of capture location rollback, application location crossing barriers, or partial commits were observed in the experiment.

4. Discussion

The main results show that AOMB still outperforms the post-test optimal fixed batch under unknown loads, reducing the likelihood of inappropriate baseline parameter selection as an explanation. The gap between AOMB and Oracle mainly occurs near burst termination, database checkpoints, and high-conflict barriers, locations prone to service estimation lag. MPA-TB has a relatively high age violation rate, indicating that directly constraining transaction age is more stable than relying on manual state thresholds.

The main difference between AOMB and existing methods lies in the control objective. Asynchronous micro-batch commit focuses on decoupling the commit and checkpoint, elastic stream processing adapts to load changes by adjusting parallel resources, while MPA-TB and queue AIMD mainly use stage or backlog feedback. AOMB does not expand resources, but uses batch size as a control variable and uses transaction age SLO, Binlog barrier and stopping time to jointly constrain the decision. The study combines the idea of information age [10] with CDC persistent queue, service capacity and migration barrier, so that the theoretical boundary reflects both service margin and estimation error[11].

Experimental observations show that the inflection point of batch benefits varies with system conditions. At low RTTs, the group commit benefits from larger batches are gradually offset by WAL refresh, lock holding, and sequence barrier waiting; as RTT increases, the inflection point shifts to 32 and 48. AOMB can adjust batches based on online service estimates, but the confidence boundary coverage on RLS is below the nominal 95%. This bias indicates that capacity margins should still be maintained during deployment, and model errors should be continuously monitored[12-13].

High-concurrency experiments further clarified the method's boundaries. In the constant-rate control, warehouse increases the ability to distribute key conflicts more evenly; however, the batch control advantage disappears when hotspots, capacity pressure, and commit barriers are simultaneously amplified. Dependency graph scheduling can serve as an in-batch execution extension of AOMB to reduce serial waits for conflict-free transactions, while AOMB continues to handle batch size and age constraints. Partition commit is only compatible with the sequential semantics presented in this paper if it maintains the global application watermark and coordinates cross-partition transactions. The relevant mechanisms still require independent experimental verification[14-15].

This performance gain also comes at a certain cost. A lower age violation rate requires more target commits, which may increase WAL records, synchronous disk flushing and transaction ID consumption. There are also several limitations to the scope of the study: the target database does not provide read services before the switch, batch transactions do not maintain inter-transaction visibility, and the actual arrival interval scenario only reuses the public timestamp. The transaction content still comes from TPC-C, so it cannot replace the production migration trajectory. Further research needs to further verify production-level load, other database combinations and conflict scheduling; schema and integrity constraints should still be handled separately from incremental scheduling boundaries [16].

5. Conclusion

Regarding the first research question, AOMB met the Binlog barrier, pre-barrier transaction clearing, and age SLO conditions earlier than the posterior optimal fixed batch under all three types of unknown test loads, and maintained a limited gap with offline Oracle under the same constraints. Estimation error and finite control cycle for online services remain key areas for future improvements.

Regarding the second research question, AOMB reduced transaction age exceeding limits for Fixed Batch, Queued AIMD, and MPA-TB when service capacity was feasible and conflict pressure was controlled. Experiments with high hotspots and capacity boundaries also showed that commit barriers may offset the benefits of online batch control; therefore, the effectiveness of the method still depends on service margin and conflict level.

This study transforms the micro-batch selection in CDC migration into an online queuing problem constrained by both transaction age and log barriers, forming a complete research chain covering pre-decision characteristics, persistent queues, service estimation, concurrency ordering, and precise one-time recovery of effects. AOMB improves switchover timeliness under unknown loads at the cost of moderately increasing the number of target commits.

References

- [1] Fragkoulis M, Carbone P, Kalavri V, Katsifodimos A. A survey on the evolution of stream processing systems[J]. *The VLDB Journal*, 2024, 33(2): 507-541. DOI: 10.1007/s00778-023-00819-8.

- [2] Zhang S, Soto J, Markl V. A survey on transactional stream processing[J]. *The VLDB Journal*, 2024, 33(2): 451-479. DOI: 10.1007/s00778-023-00814-z.
- [3] Zhao J, Mao Y, Yang Z, Liu H, Zhang S. Scalable transactional stream processing on multicore processors[J]. *IEEE Transactions on Knowledge and Data Engineering*, 2025, 37(7): 4254-4269. DOI: 10.1109/TKDE.2025.3556741.
- [4] CAO H, HUANG C, LIU S, et al. Aion: Live migration for in-memory databases with zero downtime and reduced redundant data transfer[J]. *Data Science and Engineering*, 2025, 10(2): 212-229. DOI: 10.1007/s41019-024-00276-5.
- [5] RUCCO C, LONGO A, SAAD M. Enhancing data ingestion efficiency in cloud-based systems: A design pattern approach[J]. *Data Science and Engineering*, 2026, 11(2): 287-302. DOI: 10.1007/s41019-025-00300-2.
- [6] Wang, B. (2025). *Methods of Load Optimization for Computer Systems Based on Physical Principles*.
- [7] Guo, X. (2025, March). *Research on Blockchain-Based Financial AI Algorithm Integration Methods and Systems*. In *2025 IEEE International Conference on Electronics, Energy Systems and Power Engineering (EESPE)* (pp. 816-821). IEEE.
- [8] Guo, X. (2025, April). *Research on Financial Trading Algorithms Based on Deep Reinforcement Learning*. In *2025 IEEE 3rd International Conference on Control, Electronics and Computer Technology (ICCECT)* (pp. 1711-1715). IEEE.
- [9] Yu, X. (2026). *Exploration of Multi-Channel Conversion Path Optimization Methods Based on A/B Testing*.
- [10] Jing, X. (2025, November). *In-Depth Analysis of the Optimization Decision Model of Financial Investment Portfolio Based on Machine Learning Algorithms in Python Scikit-Learn Library*. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 130-140). Cham: Springer Nature Switzerland.
- [11] Liu, X. (2026). *Research on the Application of Artificial Intelligence in Consumer Privacy Data Protection*. *Procedia Computer Science*, 279, 956-965.
- [12] Liu, X. (2026). *Construction of Consumer Data Privacy Protection System Based On Blockchain Technology*. *Procedia Computer Science*, 282, 2082-2091.
- [13] Han, W. (2025, November). *An Empirical Study on Building a High-precision Forecasting System for Financial Market Trends Based on the PyTorch Deep Learning Framework*. In *International Conference on Computational Technologies for Research in Data Analytics* (pp. 518-528). Cham: Springer Nature Switzerland.
- [14] Yuan, Y. (2026). *Research on Memory Management and Dynamic Reasoning Methods for Intelligent Agents Based on Large Language Models*.
- [15] Yang, Y. (2026). *Causal Inference-based Identification of Incremental Effects in Digital Advertising and Optimization Pathways for Resource Allocation*. *Advances in Computer and Communication*, 7(2).
- [16] Sun, L. (2026). *Consistency Optimization Strategies for Distributed Advertising Systems Under Cross-regional Deployment*. *Advances in Computer and Communication*, 7(2).