Confidence Bounded Replica Currency Estimation

Yu Sun
BNRist, Tsinghua University
Beijing, China
sy17@mails.tsinghua.edu.cn

Zheng Zheng
McMaster University
Hamilton, Ontario, Canada
zhengz13@mcmaster.ca

Shaoxu Song
BNRist, Tsinghua University
Beijing, China
sxsong@tsinghua.edu.cn

Fei Chiang
McMaster University
Hamilton, Ontario, Canada
fchiang@mcmaster.ca

ABSTRACT

Replicas of the same data item often exhibit varying consistency levels when executing read and write requests due to system availability and network limitations. When one or more replicas respond to a query, estimating the currency (or staleness) of the returned data item (without accessing the other replicas) is essential for applications requiring timely data. Depending on how confident the estimation is, the query may dynamically decide to return the retrieved replicas, or wait for the remaining replicas to respond. The replica currency estimation is expected to be accurate and extremely time efficient without introducing large overhead during query processing. In this paper, we provide theoretical bounds on the confidence of replica currency estimation. Our system computes with a minimum probability p, whether the retrieved replicas are current or stale. Using this confidence-bounded replica currency estimation, we implement a novel DYNAMIC read consistency level in the open-source, NoSQL database, Cassandra. Experiments show that the proposed replica currency estimation is *intuitive* and *efficient*. In most tested scenarios, with various query loads and cluster configurations, we show our estimations with confidence levels of at least 0.99 while keeping query latency low (close to reading ONE replica).

CCS CONCEPTS

• **Computing methodologies** → **Distributed computing methodologies**.

KEYWORDS

replica currency, distributed database

ACM Reference Format:
Yu Sun, Zheng Zheng, Shaoxu Song, and Fei Chiang. 2022. Confidence Bounded Replica Currency Estimation. In *Proceedings of the 2022 International Conference on Management of Data (SIGMOD '22), June 12–17, 2022, Philadelphia, PA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3514221.3517852

SIGMOD '22, June 12–17, 2022, Philadelphia, PA, USA

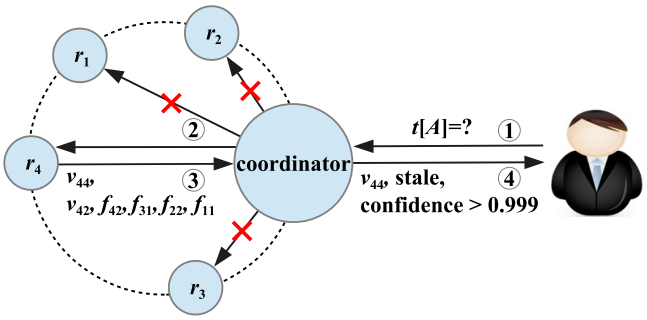
ACM ISBN 978-1-4503-9249-5/22/06…$15.00
https://doi.org/10.1145/3514221.3517852

1 INTRODUCTION

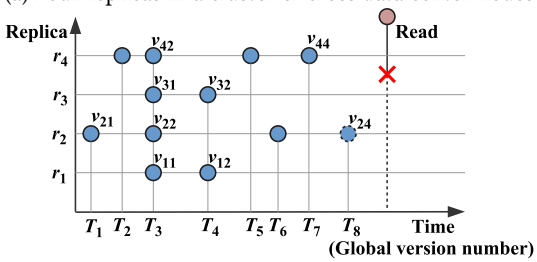
Data currency is often synonymously referred to as data freshness in distributed database settings, and a critical data quality dimension [18]. One of the primary problems in data currency is to determine whether a value at a replica is stale, as it makes answering queries across these multiple value versions challenging. Distributed data stores with replication, such as BigTable [12], HBase [20], Dynamo [17], and Cassandra [27], carry different versions of a data item (i.e., a key-value pair, a tabular cell), where timestamps (or global version numbers) are used to determine currency. Replicated storage for large-scale applications such as Web services [30], and in e-commerce platforms [25] must consider the trade-off between stronger forms of consistency and performance requirements. Although stronger consistency provides improved guarantees of data currency, it often requires heavier-weight implementations that increase latency and/or decrease throughput [30]. Therefore, systems providing strong consistency guarantees are limited in scalability and availability [21], and are unable to handle network partitions [23]. Referring to Brewer's CAP theorem [9] on consistency, availability, and partition tolerance, many distributed systems [10, 12, 17, 20, 27, 31] choose weaker forms of consistency to provide low latency and high throughput [30].

By manually specifying a consistency level, existing systems such as Cassandra wait for a fixed number of replicas (e.g., ONE, QUORUM, ALL) to respond with respect to (w.r.t.) the success of read/write operations. In cases of workload surge, for example, as reported by Alibaba, there are up to 491,000 sales transactions per second, during the 2018 Singles' Day Global Shopping Festival [25]. To handle this, the write consistency level ONE is preferred, i.e., returning to the client as long as one node is written, to ensure high throughput. The remaining replicas cannot guarantee immediate and perfect consistency, especially in Wide Area Networks (WANs) with frequent network issues, hardware unreliability, or overloaded nodes that experience pauses due to garbage collection. In addition, studies have shown that periods of inconsistency follow write requests as updates are not uniformly propagated to all replicas due to buffering and communication delays [7].

To achieve strong consistency, where all client reads view the most recent write, the read consistency level ALL is required, which waits for all replicas to respond, given the aforesaid write consistency level ONE. However, this increases latency, which is undesirable in many real applications [37]. For example, 500 ms of latency in Google's search leads to 20% decrease in traffic [29], and 100



(a) Four replicas in a cluster of cross data center nodes



(b) v_{ij} the j-th version written in replica r_i at different times

Figure 1: Motivating example showing a cluster with replication factor 4, and write consistency level ONE, running across multiple data centers. A read returns the most recently written version, i.e., v_{24}, the 4th version stored in replica r_2. Unfortunately, this does not always occur due to system availability and network partitioning. For downstream applications, it is critical to estimate whether the response replica, e.g., v_{44}, is current or stale.

ms of latency has resulted in a 1% drop in sales at Amazon [28]. Therefore, existing tunable consistency levels sacrifice some consistency to achieve total availability. For instance, in Cassandra, the read consistency level ONE or QUORUM chooses to retrieve, respectively, one or more than half of the replicas. Obviously, the returned replicas may be stale (as illustrated in Figure 1), given the aforesaid write consistency level ONE. This solution falls short in two aspects: (1) The client has no information about the currency of the returned values from part of the replicas. (2) The consistency level is tuned w.r.t. the number of responding replicas rather than the likelihood of returning replicas with current values, i.e., not intuitive. Without any estimation of data currency, it is difficult for users to configure an appropriate consistency level. In particular, data currency varies over time such that a fixed consistency level does not always apply.

Example 1. *Consider a vehicle sensor monitoring application at our industrial partner. The workload is write-intensive with a mean inter-arrival of 2 ms, requiring write consistency ONE. The application must retrieve the current engine oil pressure values to trigger safety alarms. In such critical settings, having high confidence, current and accurate readings is essential. Relying on consistency level ONE may trigger false positive alerts, e.g., in Cassandra, and using fixed ALL and QUORUM levels leads to unnecessary delays.*

Figure 1(a) shows a system with replication factor 4. A cell $t[A]$ in row t, and column A of a table denoting the oil pressure value is replicated in four nodes in the cluster. Let v_{ij} denote the j-th version of the oil pressure value in the i-th replica. Figure 1(b) shows the latest version v_{24} of the oil pressure value is stored only in replica r_2 with write consistency level ONE, due to write efficiency or network partitioning tolerance. Read requests are also processed at different consistency levels, again, for either read efficiency or network failure. With read consistency level ONE, Cassandra immediately returns the version held by the first node that responds to the query [27], i.e., v_{44} of replica r_4. We can see that version v_{44} is stale, given the more recent v_{24}. However, without accessing the other replicas, $r_1 - r_3$, the system cannot determine whether the retrieved version v_{44} is current or stale, hindering the real-time monitoring of oil pressure values.

In this paper, we propose to (1) estimate the currency of a returned replica, and (2) devise a new consistency level DYNAMIC with a tunable confidence level of replica currency, such that the system intelligently determines the number of replicas to retrieve. Specifically, we estimate whether the returned replicas are current or stale, together with a guaranteed confidence (with at least probability p) of the estimation. This enables systems to *adaptively* decide the number of replicas to retrieve instead of a fixed ONE, QUORUM or ALL, thereby improving system efficiency. If the retrieved versions are known to be current with a high probability, the query can directly return these values as results without incurring the additional latency.

Note that reading the latest version (using strong consistency) is not always possible due to the underlying network partitioning scheme. Therefore, the new consistency level DYNAMIC provides a tunable confidence level of replica currency that is *user-intuitive*, rather than relying on a fixed number of replicas to retrieve with no guidance on the currency of results. The proposed DYNAMIC read consistency level is especially preferred in write intensive workloads, since replicas are more likely to be inconsistent among such intensive writes, and need currency estimation during reads.

1.1 Challenges

The problem of computing a lower confidence bound for replica currency estimation is challenging. (1) While efficient learning models are directly applicable, we expect a guaranteed confidence for the currency prediction. It is highly non-trivial to derive a lower bound of the probability of whether the replica is current or stale, with respect to the ground truth. (2) The problem becomes even harder, given that the replicas are distributed across cluster nodes with limited synchronization. Our currency prediction models need to be learned locally in each replica node, such that they are small enough to propagate to the other nodes with low overhead.

1.2 Contributions

Our major contributions in this study are as follows.

(1) We derive a theoretical bound on the confidence that a replica is current or stale, with respect to the ground truth (Section 2). The derivation is first based on idealistic scenarios, where the model is learned over the full update history across all replicas.
(2) We extend the theoretical bound for the replica currency estimation confidence to realistic scenarios in Section 3. We develop

Table 1: Notation

Symbol	Description
r_i	the i-th replica of the data item (queried by the user)
v_{ij}	the j-th version of the i-th replica for the data item
T_{ij}	time stamp of replica version v_{ij}
t_{ij}	time distance of replica version v_{ij} to the previous version $v_{i,j-1}$, i.e., $t_{ij} = T_{ij} - T_{i,j-1}$
ζ	the current time of user query to the database
f	model for predicting t_{ij}
ϕ	parameter of model f
Z	a number of z samples for training model f

models that are learned locally within each replica node, and together with the queried data, sent to the coordinator node for query processing and confidence-bounded currency estimation.

(3) We implement a novel read consistency level DYNAMIC, in Section 4. The system immediately returns the response replicas if the currency estimation confidence is greater than a threshold η. Instead of the fixed options (ONE, QUORUM, ALL), the proposed DYNAMIC consistency level offers more intuitive, finely tunable confidence levels (e.g., $\eta = 0.99, 0.999, 0.9999, \ldots$) to evaluate the trade-off between replica currency and query efficiency.

(4) We conduct extensive experiments with various query loads and cluster configurations. In the tested scenarios, our replica currency estimation demonstrates high confidence levels (at least 0.99), and is efficient, incurring only 0.76%-1.17% of the original query processing and replica synchronization time costs. Remarkably, the proposed DYNAMIC consistency level achieves higher accuracy to return current answers than QUORUM (comparable to ALL), while keeping latency low (close to ONE).

Table 1 lists frequently used notations.

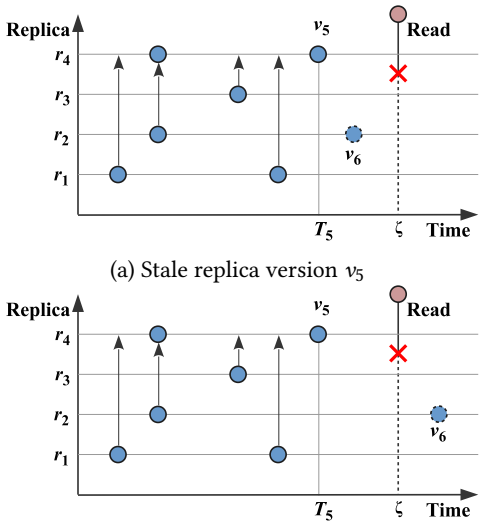
2 IDEALISTIC SCENARIOS

We first consider an idealistic scenario, where the full update history across all replicas is available, and is propagated to the responding node. Although this is costly in practice, we consider this scenario for the following reasons: (1) By predicting a bounded confidence over the full history with an idealistic node, we can extend this to use a partial history in a realistic node, as presented in Section 3. (2) Prediction using an idealistic node (by propagating the full write history at no cost) serves as a baseline of the best effort results. This enables us to demonstrate the accuracy of our estimation models (in practice) to the ideal case (Section 5).

2.1 Replica Currency in Idealistic Scenarios

Let v_n with timestamp T_n be the most recent version of the queried data item in the response nodes, and ζ be the query time. Replica currency is determined by the next version v_{n+1} with timestamp T_{n+1}, which may not have been retrieved yet from the remaining replicas, or is beyond the query time. If

$$T_{n+1} \leq \zeta, \tag{1}$$



(a) Stale replica version v_5

(b) Current replica version v_5

Figure 2: Replica currency estimation in an idealistic scenario. The update history for all replicas is propagated to a responding node for learning and currency estimation.

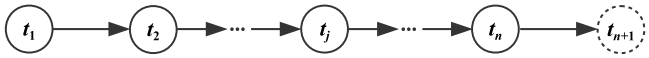


Figure 3: Prediction model for consecutive replica versions.

the version v_n is stale, i.e., a more recent version v_{n+1} is written to the other replicas but not yet retrieved. Otherwise, if

$$\zeta < T_{n+1}, \tag{2}$$

we can conclude that v_n is current, i.e., the next write operation has not yet occurred and will happen beyond the query time.

Example 2. *Figure 2(a) shows that version v_5 is the latest version with timestamp T_5 from responding replica r_4. However, if there is a version v_6 written to replica r_2 with timestamp $T_6 < \zeta$ that has not responded, then v_5 is a stale version. In contrast, Figure 2(b) shows that if there is no write between T_5 and ζ, i.e., the next version v_6 does not occur, then the retrieved version v_5 is current.*

Example 2 shows two cases should be considered to determine the currency of a data item v_n: (i) whether the next version v_{n+1} exists in a replica, but has not been retrieved; or (ii) v_{n+1} has not yet occurred (written) w.r.t. the current query time, i.e., the corresponding time stamp T_{n+1} is unknown. Thus, we can evaluate replica currency by estimating the predicted T'_{n+1} from T_{n+1}. By bounding the generalization error in the prediction (Section 2.1.2), we derive a lower bound on the confidence for replica currency estimation of T'_{n+1} (Propositions 1 and 2).

2.1.1 Prediction Model. Let $v = \{v_1, \ldots, v_k, \ldots, v_n\}$ be the replica versions that are ideally propagated to the response node of replica r_i. Rather than directly predicting the timestamp of the next version v_{n+1}, as illustrated in [15, 16], it is more reasonable to train a model

over the time distance to the last version. That is, we consider the time distance $t_k = T_k - T_{k-1}$ of replica version v_k to the previous version v_{k-1}. Since the data may be skewed [25], it is more intuitive to apply a machine learning model for prediction, rather than basic statistical analysis. Figure 3 illustrates the Bayesian model for prediction [19]. The prediction model f predicts t_k referring to the previous time distance t_{k-1},

$$f(t_{k-1}) \rightarrow t_k, \tag{3}$$

having $t_k = f(t_{k-1}) + \varepsilon$, where ε is the error term. We use a simple linear regression model [39] as f, for the following reasons,

$$t'_k = \phi_1 t_{k-1} + \phi_0, \tag{4}$$

where ϕ_0 and ϕ_1 are the parameters. (1) The generalization error of the linear regression model is bounded [39], as shown in Section 2.1.2. This serves as a foundation for the confidence-guaranteed replica currency estimation in Section 2.2. (2) The model is small enough to be propagated efficiently among the nodes in a cluster. This will be particularly important in the realistic scenarios considered in Section 3, and in our Cassandra implementation in Section 4. (3) Linear regression can be efficiently trained, incrementally, as new versions arrive [33].

Consider a sample of training data $Z = ((t_1, t_2), \ldots, (t_{n-1}, t_n))$ obtained from the write history v of replica versions, for learning the model parameters in f. Let $X = \begin{pmatrix} t_1 \cdots t_{n-1} \\ 1 \cdots 1 \end{pmatrix}$ and $Y = \begin{pmatrix} t_2 \\ \vdots \\ t_n \end{pmatrix}$. The linear regression parameters can be learned by setting $\phi = (X^\top X)^{-1} X^\top Y$. As new versions arrive, v_{n+1}, this leads to an increment $Z' = (t_n, t_{n+1})$ of training data, the new parameters w.r.t. $Z \cup Z'$ are incrementally updated

$$\phi' = (X^\top X + X'^\top X')^{-1}(X^\top Y + X'^\top Y').$$

2.1.2 Generalization Bound. The generalization error of the linear regression model is bounded [39]. This guarantees the prediction accuracy of the estimated t'_{n+1} (computed by model f) is bounded compared to the true t_{n+1}. We derive the lower bound for the confidence of replica currency estimation in Propositions 1 and 2.

Let $\hat{R}_Z(f)$ denote the empirical loss over the sample training data Z, where $|Z| = z$, and $\mathfrak{b}$ be the bound of the absolute loss function, i.e., $|f(t_{k-1}) - t_k| \leq \mathfrak{b}$. According to [39], for any $\delta > 0$, with probability at least $1 - \delta$ over the sample Z, we have

$$|t'_{n+1} - t_{n+1}| < \hat{R}_Z(f) + \mathfrak{b}\sqrt{\frac{4\log\frac{ez}{2}}{z}} + \mathfrak{b}\sqrt{\frac{\log\frac{1}{\delta}}{2z}}.$$

That is, the following probabilistic inequality holds

$$P\left(|t'_{n+1} - t_{n+1}| < \gamma\right) > 1 - \delta,$$

where

$$\gamma = \hat{R}_Z(f) + \mathfrak{b}\sqrt{\frac{4\log\frac{ez}{2}}{z}} + \mathfrak{b}\sqrt{\frac{\log\frac{1}{\delta}}{2z}}. \tag{5}$$

2.2 Confidence Bounds for Replica Currency

Based on the predicted t'_{n+1} in Formula 4 and timestamp T_n of the last retrieved version v_n, we estimate the timestamp of the next version v_{n+1}, i.e., $T'_{n+1} = T_n + t'_{n+1}$. Referring to Formulas 1 and 2, if $T'_{n+1} \leq \zeta$, we estimate that v_n is stale; otherwise, v_n is current.

In this section, we derive a theoretical bound on the confidence to estimate whether a replica is current or stale. Specifically, what is the probability of the true $T_{n+1} \leq \zeta$, given the stale estimation $T'_{n+1} \leq \zeta$? Similarly, for currency estimation, $T'_{n+1} > \zeta$, what is the confidence that the true $T_{n+1} > \zeta$?

2.2.1 Confidence of Currency Estimation. The replica currency of v_n is determined by the next version v_{n+1}. If $\zeta < T_{n+1}$, i.e., the next replica version v_{n+1} occurs after the current query time, we conclude the retrieved version v_n is current. From Formula 2, we denote the confidence of a replica version v_n being current by

$$P(v_n \text{ is current}) = P(\zeta < T_{n+1}) = P(\zeta < T_n + t_{n+1}). \tag{6}$$

Based on the generalization bound between t_{n+1} and the predicted t'_{n+1} in Section 2.1.2, we derive the lower bound on the confidence $P(\zeta < T_{n+1})$, given the estimation $\zeta < T'_{n+1} = T_n + t'_{n+1}$.

Proposition 1. *To estimate a current replica,*

$$\zeta < T'_{n+1},$$

we have the estimation confidence as

$$P(v_n \text{ is current}) = P(\zeta < T_{n+1}) > 1 - \delta,$$

where

$$\delta = e^{-2z\left(\frac{T'_{n+1} - \zeta - \hat{R}_Z(f)}{\mathfrak{b}} - \sqrt{\frac{4\log\frac{ez}{2}}{z}}\right)^2}.$$

As shown in Figure 4, since $T'_{n+1} = T_n + t'_{n+1}$ and the prediction accuracy of t'_{n+1} is bounded by Section 2.1.2, the timestamp T_{n+1} is in the range of $(T'_{n+1} - \gamma, T'_{n+1} + \gamma)$ with probability at least $1 - \delta$. If the query time has $\zeta = T'_{n+1} - \gamma$, as illustrated in Figure 4(a), we have $\zeta = T'_{n+1} - \gamma < T_{n+1}$, i.e., the version v_n is current as defined in Formula 2.

2.2.2 Confidence of Stale Estimation. We estimate the probability that v_n is stale, i.e., if $T_{n+1} < \zeta$, then v_{n+1} will occur before the query time ζ, but the replica version has not yet been retrieved. Referring to Formula 1, the confidence of version v_n being stale is

$$P(v_n \text{ is stale}) = P(T_{n+1} \leq \zeta) = P(T_n + t_{n+1} \leq \zeta). \tag{7}$$

Similarly, the lower bound of the confidence $P(T_{n+1} \leq \zeta)$, given the estimation for a stale replica $T'_{n+1} \leq \zeta$, is guaranteed by the generalization error on t'_{n+1} and t_{n+1} in Section 2.1.2.

Proposition 2. *To estimate a stale replica,*

$$T'_{n+1} \leq \zeta,$$

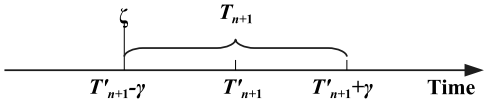
we have the estimation confidence as

$$P(v_n \text{ is stale}) = P(T_{n+1} \leq \zeta) > 1 - \delta,$$

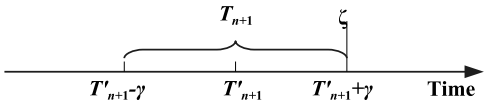
where

$$\delta = e^{-2z\left(\frac{-T'_{n+1} + \zeta - \hat{R}_Z(f)}{\mathfrak{b}} - \sqrt{\frac{4\log\frac{ez}{2}}{z}}\right)^2}.$$

Similar to the proof of Proposition 1, in Figure 4(b), if the query time is $\zeta = T'_{n+1} + \gamma$, we have $T_{n+1} \leq \zeta = T'_{n+1} + \gamma$, i.e., the version v_n is stale (Formula 1).



(a) Current estimation of version v_n



(b) Stale estimation of version v_n

Figure 4: Replica currency estimation with model generalization bound w.r.t. γ in Formula 5.

3 REALISTIC SCENARIOS

Assuming the complete update history exists in a single node for model training and scoring is expensive and unrealistic in our settings with heterogeneous clusters of varying response times, node availability, and writes not being propagated to all nodes. In this section, we consider more realistic scenarios where models are learned locally at each replica node, and together with the data, sent to the coordinator node for query processing and currency estimation. We extend the theoretical bound of the replica currency estimation confidence (derived under a centralized environment in Section 2.2) to distributed scenarios. This facilitates implementation of replica currency estimation in industrial-strength distributed database systems (Section 4).

We first illustrate the distributed settings in Section 3.1. In Section 3.2, we describe the realistic settings of replica currency estimation in a distributed environment. We then present our solution that trains individual prediction models for each replica, and compute the corresponding theoretical bounds for the confidence estimation. Lastly, we assemble the currency estimation results across all the unseen replicas in Section 3.4.

3.1 System Model

To define the most recent item, we use a global version number (global time) as shown in Figure 1(b). The communication and synchronization time is modeled in Equation 13, where the last synchronization occurs at time T_{qw} with version number v_{qw}, due to a read repair request or anti-entropy. We visually depict the synchronization time at T_3 in Figure 1(b) where versions are synchronized across all replicas, i.e., v_{11}, v_{22}, v_{31}, v_{42} are the same version stored in replicas r_1, r_2, r_3, r_4, respectively. In our evaluation, we study the impact of varying time intervals between synchronizations. Figure 9 shows that frequent communication and synchronization leads to higher estimation confidence, and accuracy.

Synchronization methods and triggers. In the realistic scenarios, synchronization among the replicas is triggered minimally, i.e., during regular anti-entropy, and occasionally by read repair, as in Cassandra. During synchronization, e.g., at time T_3 in Figure 1(b), not only the versions of data v_{11}, v_{22}, v_{31}, v_{42} are shared among replicas, but also their individually learned models known as $f_{11}, f_{22}, f_{31}, f_{42}$ for currency estimation. Our methods make no assumptions on the frequency of node synchronization. When there are more frequent read repairs or anti-entropy operations, we expect our models to compute more accurate currency estimations, as evaluated in Figures 9 and 10, albeit with longer wait times, as the latest models and values are exchanged between replicas.

Relation to federated learning problems. Our problem shares conceptual similarity to the federated learning problem where our local models contain partial data and exchange model parameters. We consider a federated learning system as a learning process in which the data owners collaboratively train a model, and do not expose their data to others [41]. However, in our distributed database setting, data is periodically synchronized between replicas during anti-entropy or read repair operations, which is not the case in federated systems.

3.2 Replica Currency

In a distributed database system, the communication cost among different nodes is extremely sensitive. This is especially evident in WANs, with geographically distributed nodes, where data exchange between replicas should be minimized as much as possible. At the same time, low response times are critical to satisfy strict performance targets, leading to weaker consistency levels. In addition to the above settings, we assume the replicas are asynchronous, i.e., a specific replica version may only exist in a subset of replicas in the distributed system. For example, as shown in Figure 5, version v_{44} is written only in replica r_4. Synchronization among the replicas occurs minimally, e.g., regularly by anti-entropy or occasionally by read repair, as in Cassandra. For instance, at time T_{42} in Figure 5, versions are synchronized in all replicas, i.e., v_{42} and v_{22} are the same version stored in replicas r_4 and r_2, respectively.

In such a real, distributed setting, propagating the full history of all writes among nodes as in Figure 2 in Section 2.1, is unlikely. For a specific replica, we learn a local model based on its own replica versions. Our goal is to introduce minimal overhead, where the learned models are incidentally shared with other replicas during regular and irregular synchronization. For instance, in Figure 5, the model f_2 learned locally in replica r_2 is propagated to r_4 when performing a read repair at time T_{42}.

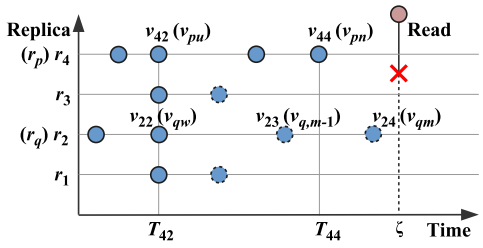
Once the latest synchronization is complete, subsequent writes are not visible between replicas. For example, in Figure 5, the version v_{23} written in r_2 after timestamp T_{42} cannot be seen by replica r_4. In addition, we do not assume in this scenario that all versions before the latest version are available for model training and scoring, i.e., v_5 in Figure 2. This relaxation introduces new challenges to estimate the currency of the retrieved version v_{44} in Figure 5.

Let v_{pn} with timestamp T_{pn} be the most recent version of the queried data item in the response nodes, and ζ be the query time. The replica's currency is dependent on the invisible versions v_{qm} with timestamp T_{qm}, in a replica r_q that is not yet retrieved. If there exists a version v_{qm} having

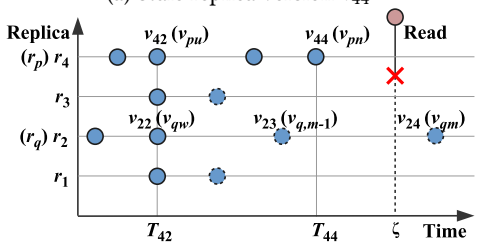
$$T_{pn} < T_{qm} \leq \zeta, \tag{8}$$

then version v_{pn} is stale. Otherwise, with

$$T_{q,m-1} \leq T_{pn}, \text{ and } \zeta < T_{qm}, \tag{9}$$



(a) Stale replica version v_{44}



(b) Current replica version v_{44}

Figure 5: Replica currency estimation in a realistic scenario, where versions since the last synchronization are not available nor shared, such as at T_{42}.

we can conclude that v_{pn} is current. In this case, no writes have occurred between the latest version v_{pn} and the query time ζ, and the next write operation will occur beyond the current query time.

Example 3. *Figure 5 shows v_{44} is the latest retrieved version with timestamp T_{44} at response replica r_4. If there is a version v_{24} written in replica r_2 with timestamp $T_{44} < T_{24} < \zeta$, and replica r_2 has not responded yet, as illustrated in Figure 5(a), then v_{44} is stale. In contrast, if there is no version (across all the invisible replicas) between T_{44} and ζ, i.e., v_{24} has not yet occurred, then v_{44} is current.*

3.3 Estimating Single Invisible Replica

To estimate the currency of the observed replica version v_{pn}, similar to the idealistic scenario, we need to predict the timestamp T_{qm} of unseen version v_{qm}, namely T'_{qm}. Specifically, we predict t'_{qm}, for each invisible replica r_q that has not yet responded to the query. Let f_q be the model in Formula 3 that is learned locally over the written versions in replica r_q. According to Formula 4, we have

$$t'_{q,k+1} = \phi_{q1} t_{qk} + \phi_{q0}, \tag{10}$$

where v_{qk} denotes the k-th version of the q-th replica r_q (that has not yet responded to the query), and ϕ_{q0}, ϕ_{q1} be the model parameters. Referring to Section 2.1.2 with

$$\gamma_q = \hat{R}_{Z_q}(f_q) + \flat\sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}} + \flat\sqrt{\frac{\log\frac{1}{\delta}}{2z_q}},$$

the generalization error between t'_{qk} and t_{qk} is bounded by

$$P(t'_{qk} - \gamma_q < t_{qk} < t'_{qk} + \gamma_q) > 1 - \delta. \tag{11}$$

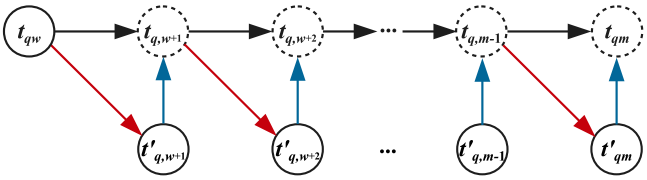


Figure 6: Predicting multiple, unseen replica versions. The predictions (shown as red arrows) are introduced in Formula 10, and the computed bounds (shown as blue arrows) are guaranteed by Formula 11.

3.3.1 *Predicting Timestamps for Currency Estimation.*

There may exist multiple versions of r_q that are invisible to the responding replica r_p, after the last synchronization at time T_{qw} with version number v_{qw}, due to a read repair request or anti-entropy.

To predict T_{qm}, or equivalently t_{qm}, we iteratively apply the prediction in Formula 10, starting from the known version v_{qw} of the last synchronization. By estimating the distance between adjacent time intervals, instead of simpler methods such as averaging over any two timestamps, our model more accurately captures periods of skewed updates. The predicted time distance t'_{qm} of replica version v_{qm} to $v_{q,m-1}$ has

$$t'_{qm} = (\phi_{q1})^{m-w} t_{qw} + \phi_{q0} \sum_{l=0}^{m-w-1} (\phi_{q1})^l. \tag{12}$$

Given $T'_{qm} = T'_{q,m-1} + t'_{qm}$ for the predicted timestamp of replica version v_{qm}, it follows that

$$\begin{aligned} T'_{qm} &= T_{qw} + \sum_{k=w+1}^{m} t'_{qk} \\ &= T_{qw} + \sum_{k=2}^{m-w+1} ((\phi_{q1})^{k-1} t_{qw} + \phi_{q0} \sum_{l=0}^{k-2} (\phi_{q1})^l). \end{aligned} \tag{13}$$

Referring to Formula 8, if the predicted time stamp T'_{qm}, $T_{pn} < T'_{qm} \le \zeta$, the observed version v_{pn} is estimated to be stale. Otherwise, as in Formula 9, with $T'_{q,m-1} \le T_{pn}$ and $\zeta < T'_{qm}$, we can estimate that v_{pn} is current. The confidence of this currency estimation is the probability of $T_{pn} < T_{qm} \le \zeta$, given the stale estimation $T_{pn} < T'_{qm} \le \zeta$, and similarly for the current estimation.

3.3.2 *Generalization Bound of Timestamp Prediction.*

To compute the confidence bound, $P(T_{pn} < T_{qm} \le \zeta)$ of the stale estimation $T_{pn} < T'_{qm} \le \zeta$, we first derive the bounds of T_{qm} w.r.t. the timestamp T_{qw} of the last known synchronization, by iteratively applying the bounds in Formula 11. We combine the relationships between T'_{qm} and T_{qw} in Formula 13, to get the bounds on T_{qm} and T'_{qm} below, which we use to derive the confidence bounds in Propositions 4 and 5.

Lemma 3. *Let $\hat{R}_{Z_q}(f_q)$ denote the empirical loss over sample training data Z_q in replica r_q, where $|Z_q| = z_q$, and $\flat$ be the bound of absolute loss function, i.e., $|f_q(t_{q,k-1}) - t_{q,k}| \le \flat$. For any $\delta > 0$, the*

following probabilistic inequality holds

$$P(T'_{qm} - \beta_{qm}\gamma_q < T_{qm} < T'_{qm} + \beta_{qm}\gamma_q) > (1-\delta)^{\frac{(m-w)(m-w+1)}{2}}, \tag{14}$$

where

$$\beta_{qm} = \sum_{k=2}^{m-w+1}\sum_{l=0}^{k-2}(\phi_{q1})^l \tag{15}$$

and

$$\gamma_q = \hat{R}_{Z_q}(f_q) + \mathfrak{b}\sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}} + \mathfrak{b}\sqrt{\frac{\log\frac{1}{\delta}}{2z_q}}. \tag{16}$$

We iteratively apply Formulas 10 and 11 to derive the generalization bound of t'_{qm}, as shown in Figure 6. We guarantee the bound of T'_{qm} via the error bounds of all previous time difference predictions t'_{qk}, $w < k \leq m$.

3.3.3 Confidence of Currency Estimation. As shown in Section 3.2, if $T_{q,m-1} \leq T_{pn}$ and $\zeta < T_{qm}$, i.e., no writes occur between the latest version v_{pn} and the query time ζ, we consider the retrieved version v_{pn} to be current. According to Formula 9, we have the confidence of a replica version v_{pn} being current as,

$$P(v_{pn} \text{ is current}) = P(T_{q,m-1} \leq T_{pn})P(\zeta < T_{qm}). \tag{17}$$

Referring to the generalization bound between T_{qm} and the predicted value T'_{qm} in Lemma 3, we derive the lower bound of the confidence $P(T_{q,m-1} \leq T_{pn})P(\zeta < T_{qm})$, given the current estimation $T'_{q,m-1} \leq T_{pn}$ and $\zeta < T'_{qm}$.

Proposition 4. *For a currency replica estimation on v_{pn} having*

$$T'_{q,m-1} \leq T_{pn} \text{ and } \zeta < T'_{qm},$$

the corresponding estimation confidence is bounded by

$$\begin{aligned} P(v_{pn} \text{ is current}) =& P(T_{q,m-1} \leq T_{pn})P(\zeta < T_{qm}) \\ >& (1-\delta_{m-1})^{\frac{(m-w-1)(m-w)}{2}}(1-\delta_\zeta)^{\frac{(m-w)(m-w+1)}{2}}, \end{aligned}$$

where

$$\delta_{m-1} = e^{-2z_q\left(\frac{T_{pn}-T'_{q,m-1}-\beta_{q,m-1}\hat{R}_{Z_q}(f_q)}{\mathfrak{b}\beta_{q,m-1}} - \sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}}\right)^2}$$

and

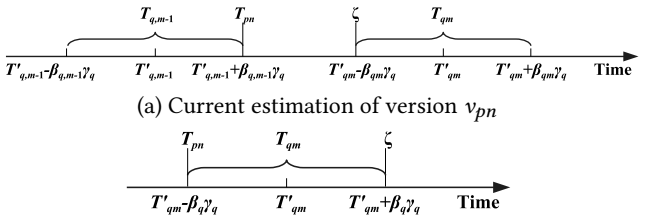
$$\delta_\zeta = e^{-2z_q\left(\frac{T'_{qm}-\zeta-\beta_{qm}\hat{R}_{Z_q}(f_q)}{\mathfrak{b}\beta_{qm}} - \sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}}\right)^2}.$$

According to Lemma 3, T_{qm} and $T_{q,m-1}$ can be theoretically bounded, as shown in Figure 7. If the query time ζ has $\zeta = T'_{qm} - \beta_{qm}\gamma_q$, as illustrated in Figure 7(a), we have $\zeta < T'_{qm}$. Moreover, if the retrieved replica version v_{pn} has $T_{pn} = T'_{q,m-1} + \beta_{q,m-1}\gamma_q$, we have $T_{pn} \geq T_{q,m-1}$.

3.3.4 Confidence of Stale Estimation. Recall if there exists at least one replica version v_{qm} such that $T_{pn} < T_{qm} \leq \zeta$, then v_{pn} is stale. Referring to Formula 8, the corresponding confidence of the replica version r_{pn} being stale is

$$P(v_{pn} \text{ is stale}) = P(T_{pn} < T_{qm} \leq \zeta) = P(T_{pn} < T_{qm})P(T_{qm} \leq \zeta).$$

Similarly, the lower bounds of the confidence $P(T_{pn} < T_{qm} \leq \zeta)$, given a stale replica version estimation $T_{pn} < T'_{qm} \leq \zeta$, are guaranteed by generalization error on T'_{qm} and T_{qm} in Lemma 3.



(a) Current estimation of version v_{pn}

(b) Stale estimation of version v_{pn}

Figure 7: Replica currency estimation with model generalization bound w.r.t. β_q (Formula 15), and γ_q (Formula 16).

Proposition 5. *For a stale replica estimation on v_{pn} having*

$$T_{pn} < T'_{qm} \leq \zeta,$$

the corresponding estimation confidence is bounded by

$$\begin{aligned} P(v_{pn} \text{ is stale}) =& P(T_{pn} < T_{qm} \leq \zeta) \\ >& (1-\delta_m)^{\frac{(m-w)(m-w+1)}{2}}(1-\delta_\zeta)^{\frac{(m-w)(m-w+1)}{2}}, \end{aligned} \tag{18}$$

where

$$\delta_m = e^{-2z_q\left(\frac{T'_{qm}-T_{pn}-\beta_{qm}\hat{R}_{Z_q}(f_q)}{\mathfrak{b}\beta_{qm}} - \sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}}\right)^2}$$

and

$$\delta_\zeta = e^{-2z_q\left(\frac{\zeta-T'_{qm}-\beta_{qm}\hat{R}_{Z_q}(f_q)}{\mathfrak{b}\beta_{qm}} - \sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}}\right)^2}.$$

Similar to the proof of Proposition 4, if the query time ζ has $\zeta = T'_{qm} + \beta_{qm}\gamma_q$, as illustrated in Figure 7(b), we have $\zeta \geq T_{qm}$. For the retrieved version v_{pn}, if its timestamp T_{pn} has $T_{pn} = T'_{qm} - \beta_{qm}\gamma_q$, we have $T_{pn} < T_{qm}$.

We note that there may exist multiple versions v_{qm}, having $T_{pn} < T'_{qm} \leq \zeta$, all of which estimate the version v_{pn} being stale. The overall (enhanced) confidence of the stale replica estimation on v_{pn} is thus obtained by aggregating the confidences of all v_{qm},

$$P(v_{pn} \text{ is stale}) = 1 - \prod_{m\in\{m|T_{pn}<T'_{qm}\leq\zeta\}}(1 - P(T_{pn} < T_{qm} \leq \zeta)). \tag{19}$$

3.4 Assembling Multiple Invisible Replicas

Section 3.3 studies currency estimation w.r.t. one invisible replica, such as r_2 in Figure 5. In real scenarios, there may exist multiple replicas that do not reply. The estimation and confidence from each invisible replica may not be equal, and we aggregate these (potentially conflicting) currency estimations on the last version v_{pn} retrieved, as well as their corresponding confidence values.

Intuitively, if there exist one invisible replica r_q that estimates the retrieved version v_{pn} is stale, e.g., r_2 in Figure 5(a), it is sufficient to conclude a stale estimation. Similar to Formula 19, we aggregate the corresponding confidence values over all the invisible replicas that have a stale estimation on v_{pn},

$$\begin{aligned} &P(v_{pn} \text{ is stale}) \\ =&1 - \prod_{q\in\{q|r_q \text{ invisible}\}}\prod_{m\in\{m|T_{pn}<T'_{qm}\leq\zeta\}}(1 - P(T_{pn} < T_{qm} \leq \zeta)). \end{aligned} \tag{20}$$

For currency, only when all the invisible replicas estimate the retrieved version v_{pn} is current, e.g., in Figure 5(b), we then conclude v_{pn} is current. The confidence is also calculated by combining the estimations of all the invisible replicas in Proposition 4,

$$P(v_{pn} \text{ is current}) = \prod_{q \in \{q | r_q \text{ invisible}\}} P(T_{q,m-1} \leq T_{pn}) P(\zeta < T_{qm}), \tag{21}$$

where m corresponds to version v_{qm} for the invisible replica r_q, leading to the current estimation $T'_{q,m-1} \leq T_{pn}$ and $\zeta < T'_{qm}$.

4 CASSANDRA IMPLEMENTATION

We implement our currency estimation models in an open-source, distributed database, Apache Cassandra [1]. Two major modifications are made in Cassandra: (1) we deploy training models in individual nodes, and propagate them during node synchronization; and (2) we perform replica currency estimation, and enable the novel DYNAMIC consistency level during query processing.

4.1 Replica Synchronization

In the realistic scenarios as in Figure 5, a prediction model f_p is learned locally in each replica r_p following the incremental learning method (Section 2.1.1). However, it is not necessary to immediately update the model after each write, which degrades write performance. Instead, model training can either be scheduled regularly (daily/weekly), with regular anti-entropy operations, or irregularly when read repair on a data item occurs. Thus, our implementation introduces no additional overhead costs for write operations.

The learned model f_p is propagated to all the other replicas of the data item for currency estimation and query processing. Let T_{pu} be the timestamp of a replica synchronization operation e.g., during anti-entropy or by read repair. The latest version v_{pu} of the data item will be synchronized across all replicas. Let f_{pu} be the latest version of model f_p at time T_{pu}, which is also propagated to all replicas. That is, when a node is re-synchronized (e.g., recovers from failure or offline), the process learns and propagates the latest models f_{pu} across all replicas. After the synchronization, each replica r_q carries both the latest version of the data item and the latest prediction models f_{pu} from the other replicas r_p. For instance, at time T_{42} in Figure 5, versions are synchronized across all replicas, i.e., v_{42} and v_{22} are the same version stored in replicas r_4 and r_2, respectively. Moreover, in addition to the data values, after incremental learning as new versions arrive, the latest prediction model f_{22} of replica r_2 is also sent to replica r_4, and vice versa.

Our work enables systems to reduce query latency when there is variance among the replica response times. This occurs in cases such as network performance instability, varying node reliability, and geographic distribution of replicas. We study the latter two cases in our evaluation. For instance, in Section 5.3, we deploy servers globally using AWS. We empirically find that increasing synchronization time intervals, and time from synchronization to user query time, results in an average 2.07% and 2.94% decline in query accuracy, as expected, as more stale data arises.

4.2 Query Processing

We introduce the necessary changes to query processing to support replica currency estimation, as illustrated in Figure 1(a) and Figure 5. The client connects to any node in the cluster as the coordinator for the read requests. The coordinator then contacts all the replicas of the requested data item. Responding replicas r_p send their latest version v_{pn} of the data item, and the additional version v_{pu} (synchronized with v_{qw}), and the models f_{qw} of all replicas r_q (including v_{pu} itself) during the last synchronization. This is necessary to predict T'_{qm} for currency estimation in Formula 13.

Based on the responding replicas thus far, the coordinator dynamically determines whether waiting for the remaining replicas is warranted, namely using the DYNAMIC consistency level. First, it estimates whether the retrieved latest version v_{pn} is stale or current, and with a bounded confidence, as per Formulas 20 and 21, respectively. Rather than retrieving a fixed number of replicas, in the case of existing protocols (ONE, QUORUM, ALL), the system immediately returns v_{pn} if its currency estimation confidence satisfies threshold η. Our proposed DYNAMIC consistency level offers more finely tunable confidence (e.g., η = 0.99, 0.999, 0.9999, ...) to enable users and applications to evaluate the trade-off between replica currency and query efficiency.

For instance, in Figure 5 at query time ζ, replica r_4 responds to the coordinator, by returning the latest version v_{44} (as in the Cassandra implementation), and also v_{42} that is synchronized with other replicas such as v_{22}, and their corresponding models f_{22}. Recall that this information is propagated and stored in all the replicas including r_4 during the last synchronization at time T_{42}. If version v_{44} is estimated to be current, with confidence at least η, say 0.999, it will be returned as the query answer together with the lower bound of the currency estimation confidence.

5 EXPERIMENTS

We implement our models using Apache Cassandra 4.0, and evaluate our models with the following objectives:

1) We compare our prediction models under the idealistic and realistic scenarios, and against a state-of-the-art deep learning model and a probabilistic approach.
2) We evaluate the accuracy, confidence, and time costs for varying node synchronization time intervals from the latest node sync time to the query read time, and to the next node sync time.
3) We measure performance for varying query workload characteristics such as # skewed updates/accesses, # write replicas, heterogeneous replicas, and # dynamic clients.
4) We compare our DYNAMIC read consistency level against existing consistency levels, and show its improved accuracy and reduced time costs.

Our experimental highlights are: (i) the replica currency estimation is accurate with high confidence; (ii) the DYNAMIC consistency level built on replica currency estimation is effective and efficient.

5.1 Experimental Setup

We setup a global data centre topology consisting of 10 Amazon EC2 m5.4xlarge instances, where each server contains 16 CPU cores and 64 GB memory, running Ubuntu 18.04 LTS. The servers

are distributed across 5 regions, namely, Canada, China, Germany, South Africa, and the US.

5.1.1 Datasets. Three real datasets are used to model query updates.

HHAR [38]: contains 43,930,257 sensor readings from smart phones and watches for human activity recognition. Readings occur irregularly to reflect heterogeneous devices, and varying user behaviour.

Btcusd [35]: contains trading data (open price) of 'Bitcoin vs. US dollar' from the Bitfinex exchange, occurring between 1 - 23 mins, with a mean 1.4 mins inter-arrival time, variance 1.6, and 5,176 writes.

Vehicle: collects from a vehicle monitoring application at our industrial partner. It is a write intensive workload, with a mean inter-arrival of 2 ms and 62,219 writes.

5.1.2 Evaluation. We generate the writes to the distributed system following the inter-arrival time between adjacent updates, given in the datasets. For adjacent updates, we randomly generate a read request between them. The arrival rate of writes/reads distributed to a replica is controlled by specifying an acceptance rate at each replica, which is defined per workload. For instance, for workloads with larger skewness S, this leads to a larger variance of acceptance rates across replicas (Section 5.4.1). Larger Poisson distribution parameter λ values lead to larger acceptance rate of the replica in Section 5.4.2.

We use the update history stored in replicas to incrementally train the prediction model f. For each read request, estimation of the returned replica is a binary classification, i.e., current or stale. F1-score [39] and the confidence (in the form of $-\log(1-\eta)$) are reported as the metrics to evaluate the currency estimation.

5.2 Comparing Prediction Models

We evaluate our **C**onfidence **B**ounded **R**eplica **C**urrency (CBRC) estimation models against the state-of-the-art methods TLSTM [6] and PBS [5]. TLSTM extends the deep learning model LSTM [24] by supporting irregular time intervals. We use its open-source implementation [2]. Similar to our work, the TLSTM prediction applies in idealistic and realistic settings. It considers the time between adjacent updates, as well as the update values as additional input. Given the differing staleness semantics with PBS, discussed in Section 6, we make the following adaptations to ensure a fair comparative evaluation. (1) In PBS, we set $k = 1$ to estimate staleness w.r.t. the latest version. (2) We set parameter t as the difference between the latest committed write among all the responding replicas and the current (read) time. (3) Similar to our proposal, we specify a staleness threshold for PBS. If the computed staleness probability is less than the threshold, the responding replica result will be immediately returned. Otherwise, we wait for more replica responses.

Our work achieves comparable accuracy to TLSTM (Figure 8(a)), but with significantly lower training time (Figure 8(c)), and scoring time (Figure 8(d)) costs. Figure 8(c) shows that TLSTM spends between 14.56 - 129.07 mins to train the prediction model during synchronization, when training rates range from 1% to 20%. TLSTM incurs a training time between 610,258 - 5,408, 600 times larger than the synchronization time cost (shown in Figure 11(a)) making it infeasible to train between synchronizations. TLSTM costs between

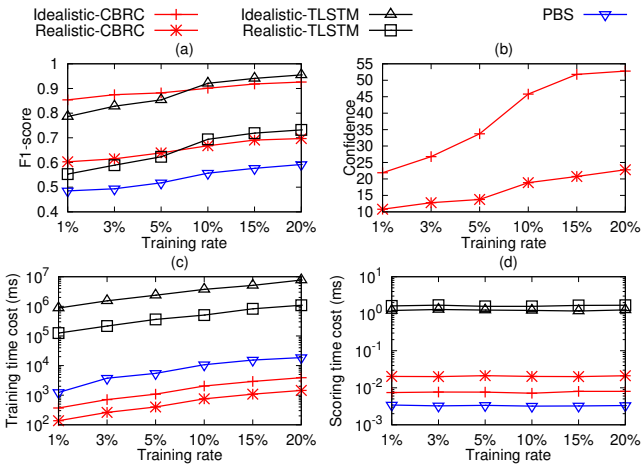


Figure 8: Replica currency estimation vs. training rates over HHAR data

1.2 - 1.3 ms in Figure 8(d) to score the response in query processing, taking longer than the query time, making it very costly to score. These results show that models such as TLSTM, which may achieve higher accuracy for large training rates, incur much higher overhead. These models are more complex, difficult to implement in distributed settings, are large to exchange between nodes, more expensive to train and evaluate during query processing, and have unknown error bounds.

In short, while more complex models may outperform for larger training sizes, our models achieve comparable accuracy but with much lower overhead. In our problem setting, where communication costs and latency must be minimized, and having current reads with high confidence is critical (e.g., our vehicle engine sensor example), using such complex models is not feasible.

In Figure 8(a), our models achieve higher accuracy than PBS as our work models write-arrival rates locally at each node. In contrast, PBS assumes all replicas share the same model, leading to less accurate probability estimations. Figure 8(b) shows, as expected, our model confidence increases as the training rate increases. Figure 8(c) shows that PBS incurs higher training overhead than our models. To estimate the probability of t-visibility, PBS must compute the CDF describing the number of write replicas containing a value v within t time units ago, which requires summing the conditional probabilities for varying numbers of write replicas (as writes are asynchronously propagated). This overhead is not incurred by our models. In Figure 8(d), our models and PBS incur comparably low scoring time costs, both less than 0.02 ms for each estimation. Both inference costs are tiny compared to the query processing time costs (0.92 ms) as illustrated in Figure 11(b), and much lower than those of LSTM. PBS assumes a single model configuration among all read operations, thereby reducing scoring times. In our work, we differentiate the number of replicas to return over the read operations to efficiently guarantee currency estimation bounds.

5.3 Node Synchronization Evaluation

We evaluate the node synchronization strategies described in Section 4.1, where model training can be scheduled. Figures 9 and 10

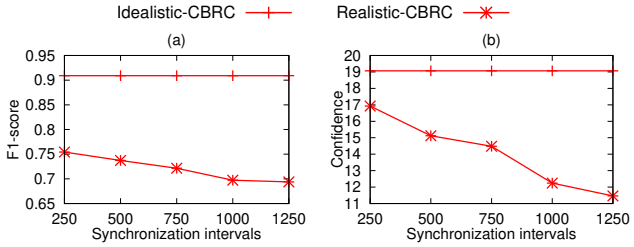


Figure 9: Varying time intervals between synchronizations intervals with two consecutive node synchronizations (e.g., T_{42} in Figure 5 and the one before it) using Btcusd data

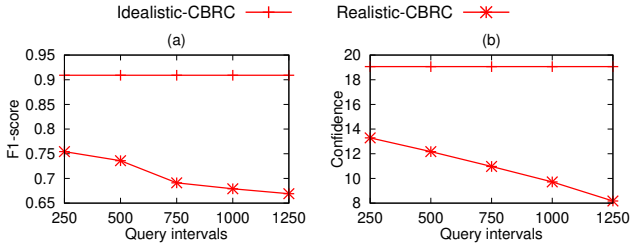


Figure 10: Varying time intervals between the latest node synchronization and query time (e.g., T_{42} and ζ in Figure 5) using Btcusd data

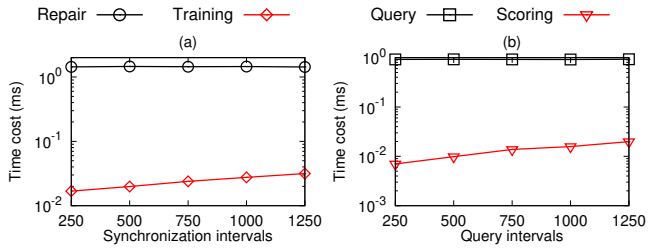


Figure 11: Time cost for varying synchronization and query time intervals using Btcusd data

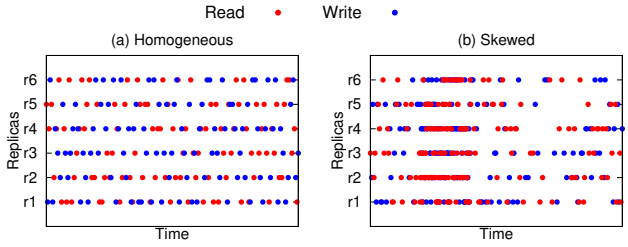


Figure 12: Query loads with (a) homogeneous and (b) skewed updates and reads

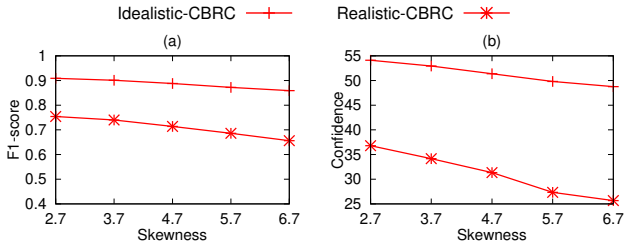


Figure 13: Replica currency estimation by varying the skewness of updates and accesses (Figure 12(b)) using Btcusd data

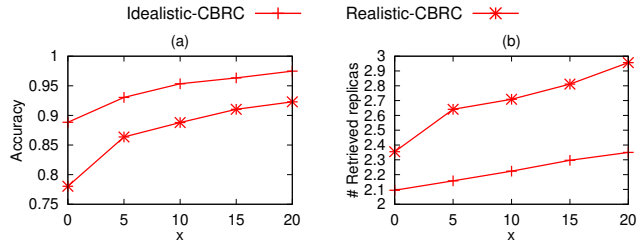


Figure 14: Query performance under various confidence threshold $\eta = 1 - e^{-x}$ using Btcusd data with the skewed updates and accesses (skewness = 6.7)

vary the time intervals from the last node synchronization to the next node synchronization, and to the user query time, respectively. According to Propositions 4 and 5, larger intervals lead to lower confidence and accuracy, as we empirically show in Figures 9 and 10. Figure 11(a) shows the time costs for read repair and training under varying synchronization time intervals. The corresponding query and estimation scoring time costs are reported in Figure 11(b). Note that the estimation scoring and training time costs are only 0.76%-1.17% of the original costs in query processing and replica synchronization (i.e., during read repair).

5.4 Varying Workloads

We evaluate currency estimation in query processing (as introduced in Section 4.2) under varying workloads.

5.4.1 Skewed Updates and Accesses. We study the impact of varying read/write distributions on our estimation accuracy. Figure 12(a) shows homogeneous reads/writes across the replicas. Figure 12(b) illustrates skewed reads/writes, which reflect data access patterns varying across time, e.g., more frequent reads/writes during the weekdays vs. the weekends. Skewness is used to measure the lack of homogeneity for read/write distributions [11].

Figure 13 shows the accuracy and confidence as we vary the skewness S, where a value of S = 2.7 indicates the homogeneous case of the original dataset (Figure 12(a)). Our estimation accuracy and confidence decline for increasing data skew, albeit rather slowly, reflecting the model's adaptivity to changing read/write patterns. That is, our model learns over data with increasing data skew.

It is possible that new data does not follow previously seen patterns. As shown in Figure 12(b), updates are clustered at the beginning (during training), followed by (previously unseen) sparse updates and reads (during testing). Figures 13(a) and 13(b) show that under skewed workloads with irregularities not necessarily seen during the training phase, our models achieve lower accuracy and confidence, respectively. We address this by increasing the confidence threshold and waiting for more replica responses. Figure 14(a) shows that with a higher confidence threshold, the query accuracy increases, and as expected, more replicas must be retrieved as reported in Figure 14(b). These cases highlight the trade-off between accuracy and the time cost.

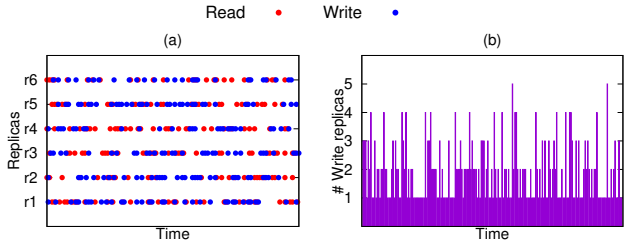


Figure 15: Query load by varying number of write replicas

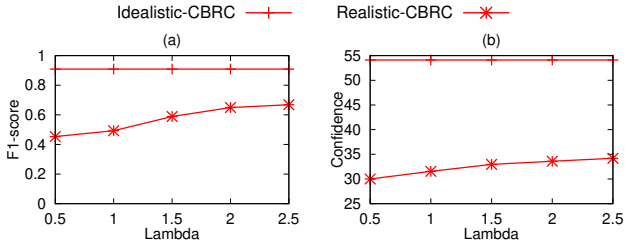


Figure 16: Replica currency estimation by varying write replicas according to λ using Btcusd data

5.4.2 Varying Write Replicas. We evaluate the impact of varying the number of write replicas. We use the Poisson distribution [14] to model the number of replicas that are written in a fixed interval of time, i.e., replica arrival rates. For instance, the blue dots in Figure 15(a) denote the write events in replicas. Figure 15(b) counts the corresponding number of write replicas in each time interval. It follows a Poisson distribution with parameter $\lambda = 2$, i.e., the expected number of write replicas arrived in each time interval is 2.

Figure 16 reports the results as we vary λ from 0.5 to 2.5, where larger λ leads to more write replicas. The results show that our models learn and adapt to increases in λ, with higher F1 scores and confidence. As the number of write replicas increases, we develop a more complete picture of the update history, which improves the reliability and accuracy of locally trained prediction models.

5.4.3 Heterogeneous Replicas. In practice, geographic proximity, priority reads, and SLAs influence how reads/writes are serviced across the replicas. In this work, our objective was to take the first steps towards a confidence-bounded currency estimation model, while keeping the model simple to minimize overhead. When we consider these factors, we have a set of heterogeneous replicas with varying rates of reads/writes. Our model is able to learn local read/write patterns per replica and thus treat them differently. We use the exponential distribution to model the heterogeneity among the replicas, by varying a parameter τ, where larger values represent greater heterogeneity. Figure 17(a) shows the heterogeneous distribution of reads/writes for six replicas. Figure 18 shows that the F1-accuracy and confidence increase for increasing heterogeneity among the replicas. When this occurs, a larger proportion of the reads/writes are propagated to a subset of the nodes. Replicas receiving more data will achieve more accurate models, leading to higher confidence estimations. In the case of $\tau = 2.5$, this is analogous to the idealistic scenario, where all the updates are available in almost one node, with a high estimation accuracy close to the idealistic scenario. Overall, our models are able to adapt to the evolving read/write distributions caused by geographic and SLA factors that lead to heterogeneous nodes.

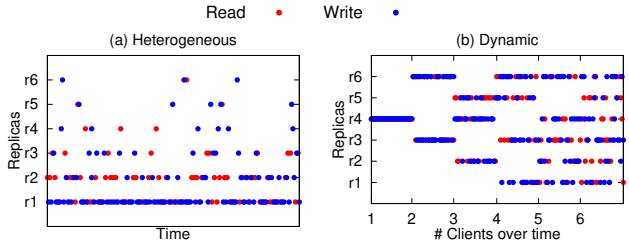


Figure 17: Query loads over (a) heterogeneous replicas and (b) dynamic clients

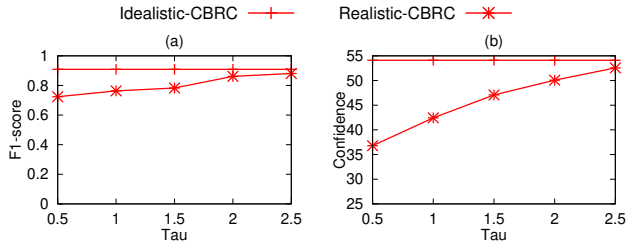


Figure 18: Replica currency estimation by varying the replica heterogeneity (Figure 17(a)) using τ in Btcusd data

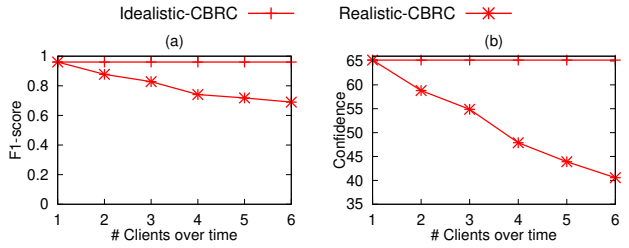


Figure 19: Replica currency estimation by varying clients over time (Figure 17(b)) using Vehicle data

5.4.4 Dynamic Clients. We measure our estimation performance in dynamic workloads that vary the number of clients over time. Figure 17(b) shows the dynamic workload with six different periods, where # clients ranges from one to six. Initially, there is only one client to service reads and writes, and more clients are added as time passes, thereby distributing the read and write responses. If there is only a single client, this is analogous to the idealistic scenario, where all updates are available in a single node for model training and currency estimation. Therefore, the idealistic model and the realistic model achieve the same performance in this case, with the same prediction model and replica responses. Under the dynamic workload, Figure 19 shows that as we increase the number of clients, the F1-score and the confidence decrease, since the updates are distributed among different replicas and fewer updates are available for local model learning and currency estimation. With less updates, this leads to a less complete picture of the update history, which reduces the reliability and accuracy of locally trained prediction models, in addition to lower estimation confidence.

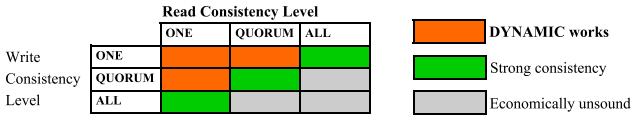


Figure 20: DYNAMIC read consistency level vs. existing read/write consistency levels

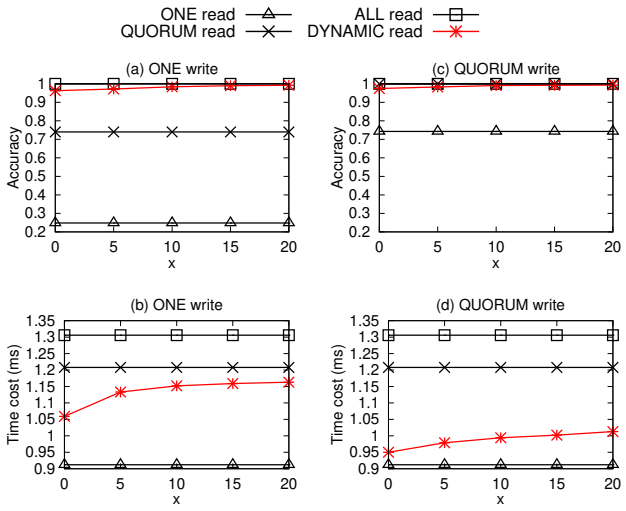


Figure 21: DYNAMIC consistency level under various confidence threshold $\eta = 1 - e^{-x}$ in write-intensive Vehicle data

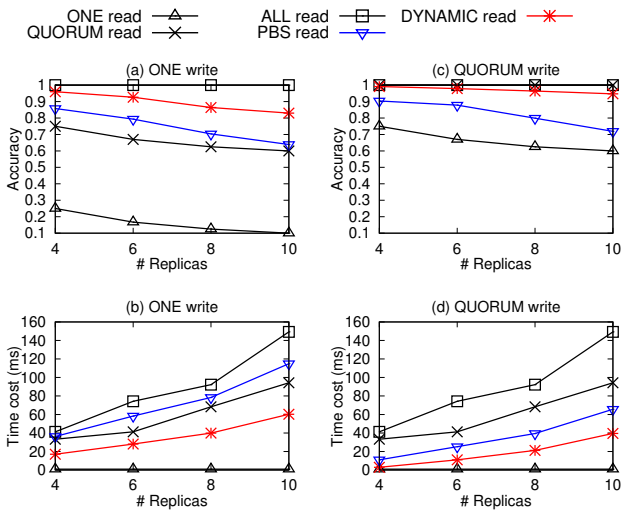


Figure 22: DYNAMIC consistency level under various number of replicas using Btcusd data

5.5 DYNAMIC Read Consistency

We evaluate our DYNAMIC read consistency level in decentralized system Cassandra considered in Section 3, and leader-based mechanism HBase analogous to Section 2.

5.5.1 Decentralized System. We evaluate the performance of DYNAMIC consistency level in write intensive dataset Vehicle, by comparing against existing read consistency levels (ONE, QUORUM, ALL) with ONE write, and with QUORUM writes. As discussed in the Introduction, our proposed DYNAMIC performs especially when strong consistency cannot be achieved. Therefore, as illustrated in Figure 20, we mainly study the write consistency levels ONE and QUORUM in this experiment. Read consistency levels ONE, QUORUM and ALL are compared with the proposed DYNAMIC. We will show that DYNAMIC is almost as accurate as the strong consistency settings, while much more efficient (with time cost close to reading ONE replica).

Figures 21(a) and 21(b) present comparative results using ONE write, followed by QUORUM write in Figures 21(c) and 21(d). Figures 21(a) and 21(c) report the accuracy of returning current replicas in query answering, guaranteed by QUORUM-read+QUORUM-write and ALL-read. As expected, as we increase the confidence threshold, the accuracy of returning responses increases, with higher accuracy using QUORUM write. In Figure 21(a), DYNAMIC achieves a higher accuracy of returning current answers than QUORUM reads (comparable to ALL). Figure 21(b) shows that DYNAMIC outperforms fixed QUORUM w.r.t. the time cost. DYNAMIC immediately returns the query answer after retrieving a high confidence current replica, thereby avoiding additional latency. This performance gain is more significant under QUORUM write in Figure 21(d), as there exist more current replicas. The results demonstrate that our DYNAMIC consistency level performs well in the write intensive workloads, by offering great opportunity for systems to decrease latency costs, and to reduce query processing times.

Moreover, we consider a real application, WAN deployment with varying network performance and replica locations. We use the AWS cloud to geographically distribute cluster nodes across continental regions to model real communication delays and network latencies within a region and across regions. Figure 22 shows accuracy and time costs as we vary the number of replicas. Figure 22(a) and Figure 22(c) show that as the number of replicas increases, the likelihood of a replica with the current version decreases, under ONE write and QUORUM write, leading to lower accuracy. It is not surprising that the time cost increases as the number of replicas increases (Figure 22(b) and Figure 22(d)), which leads to the variance in production latency. While we note that DYNAMIC may produce higher tail latency than QUORUM read to obtain the current replica and to satisfy η, our DYNAMIC consistency level achieves clearly lower time cost in average compared to QUORUM read.

In Figures 22(a) and 22(c), DYNAMIC achieves improved accuracy as PBS does not directly model inter-arrival writes among the replicas. In DYNAMIC, we immediately return the responding replicas if the currency thresholds are satisfied. For PBS, its inaccurate currency estimation leads to more reads of replicas and thus higher query overhead. Even worse, as mentioned in [5], the probability of achieving t-visibility decreases for an increasing number of replicas. This is shown in Figures 22(b) and 22(d) as longer wait times occur to satisfy currency thresholds for an increasing number of replicas.

5.5.2 Leader-based Mechanism. In a leader-based mechanism, a centralized leader sequences all writes, exactly the idealistic scenarios in Section 2. That is, the problem setting is simplified, and

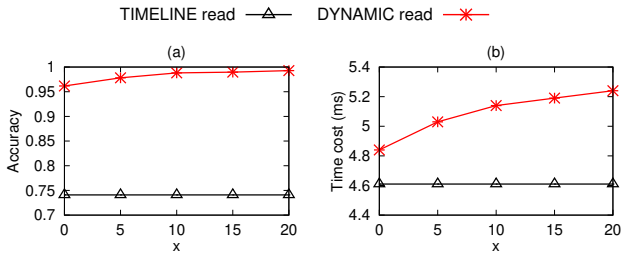


Figure 23: DYNAMIC consistency level in HBase under various confidence threshold $\eta = 1 - e^{-x}$ using Btcusd data

it is easier to reason about the freshness of the data given the full history in the centralized leader. We conduct an evaluation using HBase [3], a leader-based system. Since all the updates are written to the primary region first, and then propagated to the other secondary regions by StoreFile refresher or asynchronous replication mechanisms, we train a global model in the primary region and propagate it to replicas as well. With the TIMELINE consistency, the read request will be sent to the primary region server first. After a short interval, if the primary region replica does not respond, the read request for secondary region replicas will also be sent. HBase returns the result from the first responding replica, possibly stale.

In contrast, our DYNAMIC consistency determines the currency of the returned replica and dynamically decides whether to wait for more replicas. It guarantees that the returned data satisfies the expected currency confidence bounds. Figure 23 shows our work with HBase. By using the update history for model training, we achieve improved accuracy compared to the standard TIMELINE read, and adaptively retrieve the number of replicas to satisfy currency thresholds. Our proposal thus shows very good performance to estimate the data currency for the leader-based mechanism, with a time cost comparable to TIMELINE read.

6 RELATED WORK

Prior approaches have used two common metrics to measure staleness: absolute time and versions. In lazy-group replication, existing works have studied techniques to satisfy currency thresholds (in absolute time) using freshness locks [4]. In LazyBase, updates are batched and pipelined to answer read queries over consistent versions of the data, where more stringent currency thresholds impose higher costs [13]. In FRACS, replicas buffer updates up to a staleness window α, forcing a synchronization among all replicas, and a pause of new service requests until all buffered updates are cleared. Larger α values sacrifice consistency for improved availability [42]. In DVDC, a read replica set is used to control the divergence of data currency across the replicas by sending updates to the read set most likely to satisfy given staleness (time) bounds [40].

Adaptive approaches have traditionally studied the trade-off between consistency and availability. In network partitioned systems, methods such as inflation and deflation are proposed to dynamically balance replica availability, system efficiency, and reduce latency [23]. Inflation reduces write quorums to improve availability, whereas deflation shrinks read quorums to improve efficiency and decrease latency. The goal is to maximize transaction availability and completion in a single node.

Probabilistically Bounded Staleness (PBS) measures staleness across a set of replicas w.r.t. version history and time [5]. Despite this similar goal, there are several notable differences with our work. (1) PBS considers the last k versions to be current (k-staleness), while we impose a more strict notion with k = 1, i.e., only the latest version. (2) PBS requires the last write commit to occur t time units ago (t-visibility). Since in-flight writes may be posed by other clients, PBS does not consider this, thereby increasing #false positives. In contrast, we adaptively learn the time interval between writes to provide more accurate estimations. (3) PBS assumes all nodes share the same model (a cumulative density function), while our proposal learns a local model for each node, i.e., to accurately capture a cluster of heterogeneous nodes with various response times and availability.

Deterministic solutions have studied service protocols for reads/writes to satisfy currency and staleness bounds under different replication settings [4, 8, 13, 22, 26, 32, 34, 36, 40, 42], such as the currency-bounded replication protocol [36], relaxed currency constraint [22], freshness constraints [8]. Our work does not focus on the update propagation strategy which varies across workloads. We take a probabilistic approach to predict the currency of a read among a replica set. We address shortcomings of PBS to define localized models per replica that provide more accurate estimations. By specifying a target confidence, we dynamically adjust the number of responding replicas to guarantee a desired currency bound. We believe that our solution provides a more intuitive confidence bound for users to set rather than explicit time or version currency bounds which vary and are difficult to tune across workloads.

7 CONCLUSIONS

In this paper, we study how machine learning techniques advance replica currency estimation in distributed databases, and enable a novel DYNAMIC consistency level. Remarkably, the confidence of replica currency estimation is theoretically bounded (in Propositions 4 and 5). By referring to the guaranteed confidence of a replica's currency, the system dynamically decides whether to wait for other replicas to respond. We integrate our techniques in the open-source, distributed database Apache Cassandra [1], and conduct an extensive evaluation under various query loads and cluster configurations. The replica currency estimation is highly confident (at least 0.99 confidence levels) without introducing much overhead (incurring only about 0.76%-1.17% of the original query processing and replica synchronization time costs). The proposed DYNAMIC consistency level is effective and efficient compared to existing consistency levels (ONE, QUORUM, ALL) in Cassandra that retrieve a fixed number of replicas.

Acknowledgement

This work is supported in part by National Natural Science Foundation of China (62072265, 62021002), National Key Research & Development Plan (2021YFB3300500, 2019YFB1705301, 2019YFB1707001), BNR2022RC01011, MIIT High Quality Development Program 2020, and Natural Sciences and Engineering Research Council of Canada (2020-05711). Shaoxu Song is the corresponding author.

REFERENCES

[1] https://cassandra.apache.org/.
[2] https://github.com/illidanlab/t-lstm.
[3] https://hbase.apache.org/.
[4] F. Akal, C. Türker, H.-J. Schek, Y. Breitbart, T. Grabs, and L. Veen. Fine-grained replication and scheduling with freshness and correctness guarantees. In *International Conference on Very Large Data Bases*, pages 565–576, 2005.
[5] P. Bailis, S. Venkataraman, M. J. Franklin, J. M. Hellerstein, and I. Stoica. Probabilistically bounded staleness for practical partial quorums. *Proc. VLDB Endow.*, 5(8):776–787, 2012.
[6] I. M. Baytas, C. Xiao, X. Zhang, F. Wang, A. K. Jain, and J. Zhou. Patient subtyping via time-aware LSTM networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, August 13 - 17, 2017*, pages 65–74, 2017.
[7] D. Bermbach and S. Tai. Eventual consistency: How soon is eventual? an evaluation of amazon s3's consistency behavior. In *Proceedings of the 6th Workshop on Middleware for Service Oriented Computing, MW4SOC 2011, Lisbon, Portugal, December 12-16, 2011*, page 1, 2011.
[8] P. A. Bernstein, A. D. Fekete, H. Guo, R. Ramakrishnan, and P. Tamma. Relaxed-currency serializability for middle-tier caching and replication. In S. Chaudhuri, V. Hristidis, and N. Polyzotis, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, Illinois, USA, June 27-29, 2006*, pages 599–610. ACM, 2006.
[9] E. A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing, July 16-19, 2000, Portland, Oregon, USA*, page 7, 2000.
[10] N. Bronson, Z. Amsden, G. Cabrera, P. Chakka, P. Dimov, H. Ding, J. Ferris, A. Giardullo, S. Kulkarni, H. C. Li, M. Marchukov, D. Petrov, L. Puzar, Y. J. Song, and V. Venkataramani. TAO: facebook's distributed data store for the social graph. In *2013 USENIX Annual Technical Conference, San Jose, CA, USA, June 26-28, 2013*, pages 49–60, 2013.
[11] S. Brown. Measures of shape: Skewness and kurtosis, 2011.
[12] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. Gruber. Bigtable: A distributed storage system for structured data (awarded best paper!). In *7th Symposium on Operating Systems Design and Implementation (OSDI '06), November 6-8, Seattle, WA, USA*, pages 205–218, 2006.
[13] J. Cipar, G. Ganger, K. Keeton, C. B. Morrey, C. A. Soules, and A. Veitch. Lazybase: Trading freshness for performance in a scalable database. In *European Conference on Computer Systems*, pages 169–182, 2012.
[14] R. Cooper. *Introduction to Queueing Theory*. North Holland, 1981.
[15] C. Cuadras and C. Arenas. A distance based regression model for prediction with mixed data. *Communications in Statistics-Theory and Methods*, 19(6):2261–2279, 1990.
[16] A. H. de Souza Júnior, F. Corona, G. D. A. Barreto, Y. Miché, and A. Lendasse. Minimal learning machine: A novel supervised distance-based approach for regression and classification. *Neurocomputing*, 164:34–44, 2015.
[17] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: amazon's highly available key-value store. In *Proceedings of the 21st ACM Symposium on Operating Systems Principles 2007, SOSP 2007, Stevenson, Washington, USA, October 14-17, 2007*, pages 205–220, 2007.
[18] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012.
[19] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC press, 2013.
[20] L. George. *HBase - The Definitive Guide: Random Access to Your Planet-Size Data*. O'Reilly, 2011.
[21] J. Gray, P. Helland, P. E. O'Neil, and D. E. Shasha. The dangers of replication and a solution. In *Proceedings of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 173–182, 1996.
[22] H. Guo, P. Larson, R. Ramakrishnan, and J. Goldstein. Relaxed currency and consistency: How to say "good enough" in SQL. In G. Weikum, A. C. König, and S. Deßloch, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Paris, France, June 13-18, 2004*, pages 815–826. ACM, 2004.
[23] M. Herlihy. Dynamic quorum adjustment for partitioned data. *ACM Trans. Database Syst.*, 12(2):170–194, 1987.
[24] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.
[25] G. Huang, X. Cheng, J. Wang, Y. Wang, D. He, T. Zhang, F. Li, S. Wang, W. Cao, and Q. Li. X-engine: An optimized storage engine for large-scale e-commerce transaction processing. In *Proceedings of the 2019 International Conference on Management of Data, SIGMOD Conference 2019, Amsterdam, The Netherlands, June 30 - July 5, 2019*, pages 651–665, 2019.
[26] A. Labrinidis and N. Roussopoulos. Exploring the tradeoff between performance and data freshness in database-driven web servers. *The VLDB Journal*, 13(3):240–255, 2004.
[27] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *Operating Systems Review*, 44(2):35–40, 2010.
[28] G. Linden. Make data useful. *Online at: https://sites.google.com/site/glinden/Home/StanfordDataMining.2006-11-29.ppt*, 2006.
[29] G. Linden. Marissa mayer at web 2.0. *Online at: http://glinden.blogspot.com/2006/11/marissa-mayer-at-web-20.html*, 2006.
[30] H. Lu, K. Veeraraghavan, P. Ajoux, J. Hunt, Y. J. Song, W. Tobagus, S. Kumar, and W. Lloyd. Existential consistency: measuring and understanding consistency at facebook. In *Proceedings of the 25th Symposium on Operating Systems Principles, SOSP 2015, Monterey, CA, USA, October 4-7, 2015*, pages 295–310, 2015.
[31] R. Nishtala, H. Fugal, S. Grimm, M. Kwiatkowski, H. Lee, H. C. Li, R. McElroy, M. Paleczny, D. Peek, P. Saab, D. Stafford, T. Tung, and V. Venkataramani. Scaling memcache at facebook. In *Proceedings of the 10th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2013, Lombard, IL, USA, April 2-5, 2013*, pages 385–398, 2013.
[32] E. Pacitti, E. Simon, and R. Melo. Improving data freshness in lazy master schemes. In *International Conference on Distributed Computing Systems*, pages 164–171, 1998.
[33] D. Potts and C. Sammut. Incremental learning of linear model trees. *Mach. Learn.*, 61(1-3):5–48, 2005.
[34] X. Qin, X. Zhang, M. Q. Yasin, S. Wang, Z. Feng, and G. Xiao. SUMA: A partial materialization-based scalable query answering in OWL 2 DL. *Data Sci. Eng.*, 6(2):229–245, 2021.
[35] Carsten. 400+ crypto currency pairs at 1-minute resolution. https://www.kaggle.com/tencars/392-crypto-currency-pairs-at-minute-resolution, 2020.
[36] U. Röhm, K. Böhm, H. Schek, and H. Schuldt. FAS - A freshness-sensitive coordination middleware for a cluster of OLAP components. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB 2002, Hong Kong, August 20-23, 2002*, pages 754–765. Morgan Kaufmann, 2002.
[37] E. Schurman and J. Brutlag. Performance related changes and their user impact. In *velocity web performance and operations conference*, 2009.
[38] A. Stisen, H. Blunck, S. Bhattacharya, T. S. Prentow, M. B. Kjærgaard, A. Dey, T. Sonne, and M. M. Jensen. The heterogeneity activity recognition data set. https://archive.ics.uci.edu/ml/datasets/Heterogeneity+Activity+Recognition, 2015.
[39] M. J. Wooldridge. *Foundations of Machine Learning*. MIT Press,, 2012.
[40] T. Yamashita. Distributed view divergence control of data freshness in replicated database systems. *IEEE Transactions on Knowledge and Data Engineering*, 21(10):1403–1417, 2009.
[41] Q. Yang, Y. Liu, T. Chen, and Y. Tong. Federated machine learning: Concept and applications. *ACM Trans. Intell. Syst. Technol.*, 10(2):12:1–12:19, 2019.
[42] C. Zhang and Z. Zhang. Trading replication consistency for performance and availability: an adaptive approach. In *23rd International Conference on Distributed Computing Systems (ICDCS 2003), 19-22 May 2003, Providence, RI, USA*, pages 687–695, 2003.