Confidence Bounded Replica Currency Estimation

Yu Sun
Tsinghua University
sy17@mails.tsinghua.edu.cn

Zheng Zheng
McMaster University
zhengz13@mcmaster.ca

Shaoxu Song
Tsinghua University
sxsong@tsinghua.edu.cn

Fei Chiang
McMaster University
fchiang@mcmaster.ca

ABSTRACT

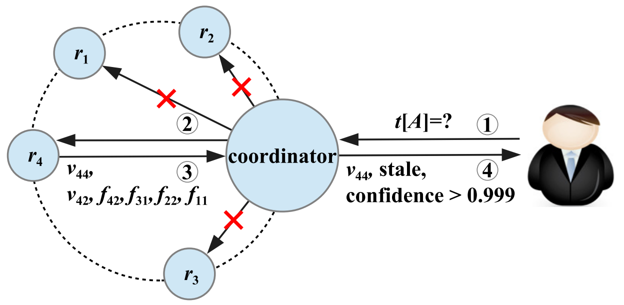
Replicas of the same data item often exhibit varying consistency levels when executing read and write requests due to system availability and network limitations. When one or more replicas respond to a query, estimating the currency (or staleness) of the returned data item (without accessing the other 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 consistency level in the open-source, NoSQL database, Cassandra. Experiments show that the proposed replica currency estimation is effective 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. Moreover, the overheads introduced due to estimation scoring and training are low, incurring only 0.006% to 0.03% of the original query processing and replica synchronization time costs, respectively.

PVLDB Reference Format:
Yu Sun, Zheng Zheng, Shaoxu Song, and Fei Chiang. Confidence Bounded Replica Currency Estimation. PVLDB, 14(1): XXX-XXX, 2020.
doi:XX.XX/XXX.XX

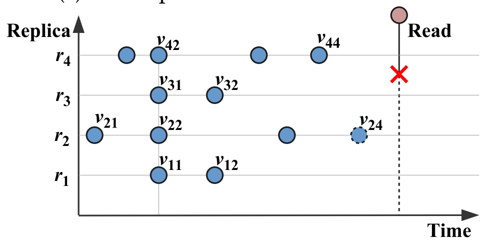
PVLDB Artifact Availability:
The source code and data have been made available at https://github.com /DynamicReplicaCurrency/DYNAMIC.

1 INTRODUCTION

Data in real systems becomes stale rapidly. Studies have shown that customer records decay at a rate of 70% per year [22]. Different values of the same entity arise if update queries are not applied simultaneously over all replicas. While each of these values was correct at some time, answering queries across these multiple value


r_1
r_2
r_3
r_4
coordinator
$t[A]$=?
①
②
③
④
v_{44}, v_{42}, f_{42}, f_{31}, f_{22}, f_{11}
v_{44}, stale, confidence > 0.999

(a) Four replicas in a cluster of nodes.



(b) Versions v_{ij} of a cell written to four replicas at different times.

Figure 1: Motivating example showing a cluster with replication factor 4, running across multiple data centers. Consistency indicates that a read returns the most recently written version, i.e., v_{24}, the 4th version stored in replica r_2. Unfortunately, this is not always possible due to system availability and network partitioning. It is highly demanded to estimate whether the response replica, e.g., v_{44}, is current or stale.

versions is challenging. Data currency is often synonymously referred to as data freshness in distributed database settings, and a critical data quality dimension [17]. One of the primary problems in data currency is to determine whether a value is stale, and if so, to compute its current value. Different models of currency have been proposed; including declarative currency constraints to determine partial currency order among records [18], and fixed vs. adaptive replication schemes to propagate updates leading to varying freshness guarantees [9, 32, 37].

In this study, we focus on distributed data stores with replication, such as BigTable [11], HBase [24], Dynamo [16], and Cassandra [29], carrying different versions of a data item (i.e., a key-value pair, or a cell in a table), where timestamps are used to determine their currency. Figure 1(a) shows a system with replication factor 4. A cell $t[A]$ in row t and column A of a table is replicated across four nodes in the cluster. According to Brewer's CAP theorem [10],

Proceedings of the VLDB Endowment, Vol. 14, No. 1 ISSN 2150-8097.
doi:XX.XX/XXX.XX

guarantees in system availability and partition-tolerance lead to sacrifices in consistency with different consistency levels across the replicas. We study the problem of stale data detection across a set of distributed nodes with varying consistency levels. We adopt a notion of currency, whereby if a pending update exists that has not yet materialized at a node, the replica is considered stale; otherwise, it is fresh/current [28, 30].

Example 1. *Let v_{ij} denote the j-th version of a cell in the i-th replica. Figure 1(b) shows the latest version v_{24} of the cell is stored only in replica r_2, 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 [29], 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 further accessing the other replicas, $r_1 - r_3$, the system cannot determine whether the retrieved version v_{44} is current or stale.*

By manually specifying a consistency level, existing systems such as Cassandra wait for a fixed number of replicas (ONE, QUORUM, ALL) to respond; such a strategy is ineffective and inefficient. At consistency level ONE, as illustrated in Figure 1, the returned replica may be stale. Alternatively, if we adopt the consistency level ALL, waiting for all replicas to respond, it blocks the read request (given the network partitioning in the cluster). The QUORUM level achieves a compromise by retrieving more than half of the replicas. However, the aforesaid issues regarding waiting for a fixed number of replicas, unfortunately, still exist.

In this paper, we offer a more intelligent option, and propose a new consistency level DYNAMIC, such that the system dynamically determines the number of replicas to retrieve. Specifically, we estimate whether the returned replicas are current or stale, and we provide a guaranteed confidence (with at least probability p) of the estimation. If the retrieved versions are known to be current with a high probability, we can directly return these values as query results. In contrast, if the values are likely to be stale, then users may decide that it is worthwhile to wait for the remaining replicas. It is worth noting that reading the latest version (using strict consistency) is not always possible due to the underlying network partitioning scheme. With the foundations of machine learning, we study and provide a lower bound p of the confidence estimation to determine whether a returned query answer is current or stale. This enables systems to adaptively decide whether to incur additional latency costs according to application requirements.

1.1 Challenges

We study the problem of computing a lower confidence bound for replica currency estimation, with the following challenges. (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.

Table 1: Notations

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

1.2 Contributions

Our major contributions in this study are as follows.

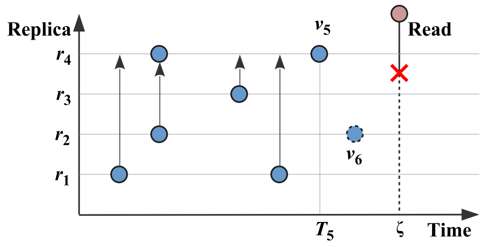
(1) We derive a theoretical bound on the confidence that a replica is current or stale, with respect to the ground truth (Section 2). The derivation is first based on idealistic scenarios, where the model is learned over the full update history across all replicas.
(2) We extend the theoretical bound for the replica currency estimation confidence to realistic scenarios in Section 3. We develop models that are learned locally within each replica node, and together with the data, sent to the coordinator node for query processing and confidence-bounded currency estimation.
(3) We implement[1] a novel 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 three fixed options (ONE, QUORUM, ALL), the proposed DYNAMIC consistency level offers more finely tunable confidence levels (e.g., $\eta = 0.99, 0.999, 0.9999, \dots$) to evaluate the trade-off between replica currency and query efficiency.
(4) We conduct extensive experiments with various query loads and cluster configurations. In most tested scenarios, our replica currency estimation demonstrates high confidence levels (at least 0.99), and is efficient (incurring only 0.006%-0.03% 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 times low (close to ONE).

Table 1 lists frequently used notations.

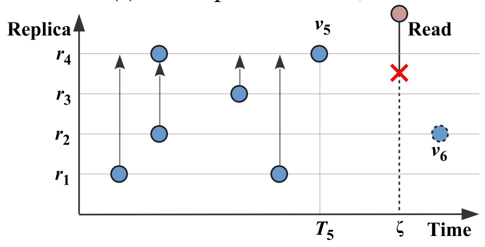
2 IDEALISTIC SCENARIOS

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

[1]Our present implementation uses Cassandra [2], and we believe that the replica currency estimation extends similarly to other (NoSQL) databases, such as HBase [4], which we intend to implement in the future.



(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.

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

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

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, 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) with respect to (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 (Proposition 1), we derive a lower bound on the confidence for replica currency estimation of T'_{n+1} (Propositions 2 and 3).

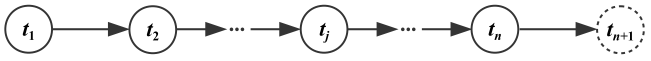


Figure 3: Prediction model for consecutive replica versions.

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 [14, 15], 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}. Figure 3 illustrates the Bayesian model for prediction [23]. 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 [35] 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 [35], as shown in Proposition 1. 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

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

We do not employ more complicated (deep) learning models, such as TLSTM [8], as our setting requires that model training and scoring need to be highly efficient, as described in Section 1.1. The performance of more sophisticated models rely on a large number of training samples [25], which may not be available in the replica write history. For instance, Cassandra regularly empties old commit logs and compacts SSTables. In our comparative evaluation (Section 5), we show that our simpler models outperform TLSTM in prediction accuracy.

2.1.2 *Generalization Bound.*
The generalization error of the linear regression model is bounded [35]. 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 2 and 3.

Proposition 1 ([35]). *Let $\hat{R}_Z(f)$ denote the empirical loss over the sample training data Z, where $|Z| = z$, and $\flat$ be the bound of the absolute loss function, i.e., $|f(t_{k-1}) - t_k| \leq \flat$. For any $\delta > 0$, with probability at least $1 - \delta$ over the sample Z, we have*

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

That is, the following probabilistic inequality holds

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

where

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

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.*
As discussed in Section 2.1, 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

$$\begin{aligned} P(v_n \text{ is current}) &= P(\zeta < T_{n+1}) \\ &= P(\zeta < T_n + t_{n+1}). \end{aligned} \quad (6)$$

Based on the generalization bound between t_{n+1} and the predicted t'_{n+1} in Proposition 1, we derive the lower bound on the confidence $P(\zeta < T_{n+1})$, given the estimation $\zeta < T'_{n+1} = T_n + t'_{n+1}$. For brevity, we provide proof sketches. All complete proofs can be found in the full version technical report [1].

Proposition 2. *To estimate a current replica,*

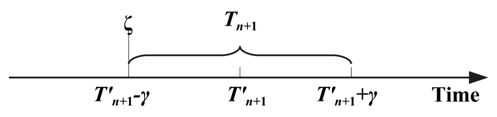
$$\zeta < T'_{n+1},$$

we have the estimation confidence as

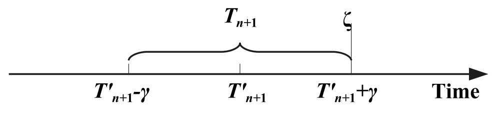
$$P(v_n \text{ is current}) = P(\zeta < T_{n+1}) > 1 - \delta,$$

where

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



(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.

Proof Sketch [1]. As shown in Figure 4, 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 now estimate the probability that v_n is stale, i.e., if $T_{n+1} < \zeta$, the next version v_{n+1} will occur before the query time ζ, but replica version has not yet been retrieved. Referring to Formula 1, the corresponding confidence of the replica version v_n being stale is

$$\begin{aligned} P(v_n \text{ is stale}) =& P(T_{n+1} \leq \zeta) \\ =& P(T_n + t_{n+1} \leq \zeta). \end{aligned} \quad (7)$$

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 studying the generalization error on t'_{n+1} and t_{n+1} in Proposition 1.

Proposition 3. *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

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

Proof Sketch [1]. Similar to the proof of Proposition 2, 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 as defined in Formula 1. □

3 REALISTIC SCENARIOS

Assuming the complete update history exists in a single node for model training and scoring is expensive and not always possible given resource constraints. 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 system (Section 4).

In Section 3.1, 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 3.3.

3.1 Replica Currency

In a distributed database system, the communication costs among different nodes are extremely sensitive. Data exchanges between replicas should be minimized as much as possible. The replicas are thus 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 should be conducted 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, is unlikely. Therefore, for a specific replica, we can only learn a local model based on its own replica versions. Our goal is to introduce as little overhead as possible, 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 in this realistic scenario is dependent on the invisible versions v_{qm} with timestamp T_{qm}, in a replica r_q that has not yet been retrieved. If there exists a version v_{qm} having

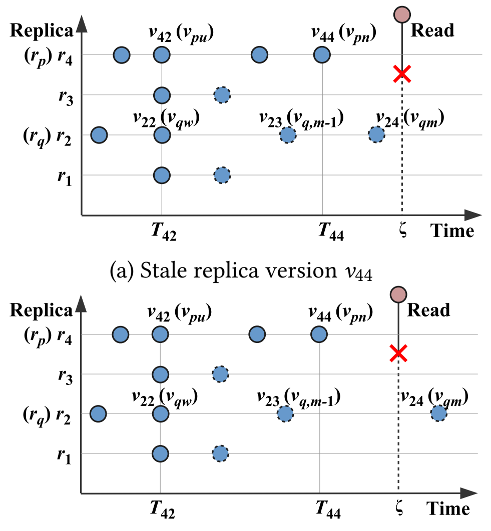
$$T_{pn} < T_{qm} \leq \zeta, \tag{8}$$

then version v_{pn} is stale, i.e., a more recent version v_{qm} is written to the other replicas but not retrieved yet. Otherwise, with

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

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}, with a response by 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., the next version v_{24} has not yet occurred, then the retrieved version v_{44} is current.*



(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}.

3.2 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 Proposition 1 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

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

3.2.1 *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. The predicted time distance t'_{qm} of replica version v_{qm} to $v_{q,m-1}$ has

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

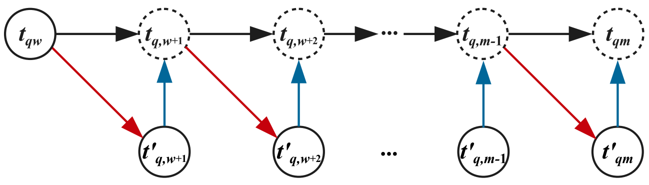


Figure 6: Predicting multiple, unseen replica versions. The predictions (shown as red arrows) are introduced in Formula 10, and the computed bounds (shown as blue arrows) are guaranteed by Formula 11.

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

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

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

3.2.2 *Generalization Bound of Timestamp Prediction.*

Our next goal is to compute the bound on the confidence, e.g., $P(T_{pn} < T_{qm} \leq \zeta)$ of the stale estimation $T_{pn} < T'_{qm} \leq \zeta$ (Section 3.2.4). To do so, we first derive the bounds of T_{qm} w.r.t. to 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 5 and 6.

Lemma 4. *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., $|f_q(t_{q,k-1}) - t_{q,k}| \leq \flat$. For any $\delta > 0$, the following probabilistic inequality holds*

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

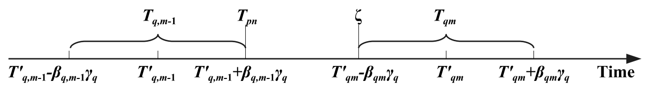
where

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

and

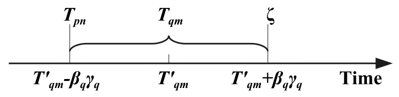
$$
\gamma_q = \hat{R}_{Z_q}(f_q) + \flat\sqrt{\frac{4\log\frac{ez_q}{2}}{z_q}} + \flat\sqrt{\frac{\log\frac{1}{\delta}}{2z_q}}. \qquad (16)
$$

Proof Sketch [1]. As shown in Figure 6, by combining Formulas 10 and 11, we have the generalization error bound of t'_{qm}. The bound of T'_{qm} is thus guaranteed by considering the error bound of all the previous time difference predictions t'_{qk}, $w < k \leq m$. □



$T_{q,m-1}$
T_{pn}
ζ
T_{qm}
$T'_{q,m-1}-\beta_{q,m-1}\gamma_q$
$T'_{q,m-1}$
$T'_{q,m-1}+\beta_{q,m-1}\gamma_q$
$T'_{qm}-\beta_{qm}\gamma_q$
T'_{qm}
$T'_{qm}+\beta_{qm}\gamma_q$
Time

(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).

3.2.3 *Confidence of Currency Estimation.*

As shown in Section 3.1, 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,

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

Referring to the generalization bound between T_{qm} and the predicted value T'_{qm} in Lemma 4, 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 5. *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

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

and

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

Proof Sketch [1]. According to Lemma 4, the timestamps 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 $T_{pn} \geq T_{q,m-1}$. □

3.2.4 *Confidence of Stale Estimation.*

Recall that if there exists at least one replica version v_{qm} occurring after the latest retrieved v_{pn}, and before the query time ζ, i.e., $T_{pn} < T_{qm} \leq \zeta$, then the retrieved replica version v_{pn} is stale. Referring to Formula 8, the corresponding confidence of the replica version r_{pn} being stale is

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

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 studying the generalization error on T'_{qm} and T_{qm} in Lemma 4.

Proposition 6. *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

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

and

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

Proof Sketch [1]. Similar to the proof of Proposition 5, 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},

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

3.3 Assembling Multiple Invisible Replicas

Section 3.2 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, e.g., r_1 and r_3 in addition to r_2 in Figure 5. The estimation and confidence from each invisible replica need not be equal, posing questions of how to aggregate and reconcile these values. In this section, we aggregate these (potentially conflicting) currency estimations on the last version v_{pn} retrieved thus far, 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, the corresponding stale estimation confidence aggregates 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 5,

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

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

This section presents our implementation in an open-source, distributed database, Apache Cassandra [2]. We believe that our replica currency estimation approaches work similarly in other (NoSQL) databases, such as HBase [4], which are avenues for future work. We implement two major modifications to the Cassandra system: (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, shown in Figure 5, a prediction model f_p is learned locally in each replica node r_p following the incremental learning method in Section 2.1.1. However, it is not necessary to immediately update the model after each write, which degrade write performance. Instead, model training can either be scheduled regularly (daily/weekly), possibly together with the regular anti-entropy operations in Cassandra, or irregularly when read repair on the data item occurs. Given this, 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., regularly by anti-entropy or occasionally by read repair in Cassandra. 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 where there exist 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 with replica response times with mean 1.35 ms and variance 0.05. 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. While we currently assume that T_{pu} is fixed according to anti-entropy or via read repair, we intend to study in future work how this synchronization time can be adaptive to the replica response times.

4.2 Query Processing

We now 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 as usual, as well as the additional version v_{pu} (synchronized with v_{qw}), and the models f_{qw} of all the replicas r_q (including v_{pu} of itself) during the last synchronization. Recall that this information is necessary to predict T'_{qm} for currency estimation in Formula 13.

Based on the replicas that have replied to the query 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, with bounded confidence as presented in 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 threshold, 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 with the following objectives:

1) We compare our estimation models under the idealistic and realistic scenarios, and against a state-of-the-art deep learning model.
2) We evaluate the accuracy and confidence of our estimation models, varying synchronization time intervals from the latest node sync time to the query read time and the next node sync time.
3) We evaluate our models by varying query workload characteristics such as network partition time, node down time, # unavailable nodes, # skewed updates/accesses, and # write replicas.
4) We compare our DYNAMIC read consistency level against the existing consistency levels and show its improved accuracy, lower number of retrieved replicas, 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, 64 GB memory, and 8 GB local storage, 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. We use five real datasets to model query updates. *Btcusd* [34] dataset is from the Bitfinex exchange, containing historical trading data (open price) of the trading pair 'Bitcoin vs. US dollar' at 1 min intervals. There are 5,167 total writes. *Sensor* [12] dataset contains temperature sensor readings on server racks in a data center over a week. Approximately 80% of the data is updated every 20 sec, and the total number of writes (as the query load) is 2,864. *Bike* [27] dataset contains bike sharing details in London, UK, with an update frequency of 1 min, with 17,414 total writes. *Mental* [31] dataset measures attitudes towards mental health, and the frequency of mental health disorders in the workplace. Updates occur irregularly, with 1,259 total writes. *Bank* [6] dataset describes bank transactions for customer bank accounts. Transactions occur irregularly, with 1,294 total writes.

5.1.2 Evaluation. While data items are written in the same sequence as in the datasets, we randomly pose read requests in the query loads. As discussed in Section 4, samples of the update history stored in replicas are used to incrementally train the prediction model f. For each read request, currency estimation of the returned replica is a binary classification, i.e., current or stale. We use F1-score [35] as our metrics to evaluate the currency estimation.

5.2 Comparing Prediction Models

We evaluate the accuracy of our prediction models for replica currency estimation under the idealistic and realistic scenarios, described in Sections 2 and 3, respectively. A state-of-the-art method TLSTM [8] is employed as a competitor to predict the timestamp of the next version. TLSTM extends the deep learning model LSTM [26] by supporting irregular time intervals. We use its open-source implementation [3]. In addition to the time between adjacent updates, also considered in our proposal, TLSTM uses more information by taking the update values as the additional input.

Figure 8 reports the results for varying training rates. With more training data, the accuracy of currency estimation improves. The corresponding confidence of the currency estimations also increases. Figure 9 presents the currency estimation performance of our models under various thresholds η. As illustrated in Figure 9(b), the percentage of all queries reduces with higher requirements of confidence η (or equivalently x). We note that the prediction accuracy of TLSTM, even using more information of update values, is lower than the linear regression used in our proposal. The result is not surprising given that the performance of deep learning models relies on a large number of training samples [25]. It is too expensive to use, as mentioned in Section 2.1.1, since our setting requires the model training and scoring to be highly efficient.

5.3 Node Synchronization Evaluation

This experiment studies the node synchronization strategies described in Section 4.1, where model training can be scheduled. Figures 10 and 11 vary the time intervals from the last node synchronization to the next node synchronization, and to the user query time, respectively. According to Propositions 5 and 6, larger intervals lead to lower confidence for the currency determination. It

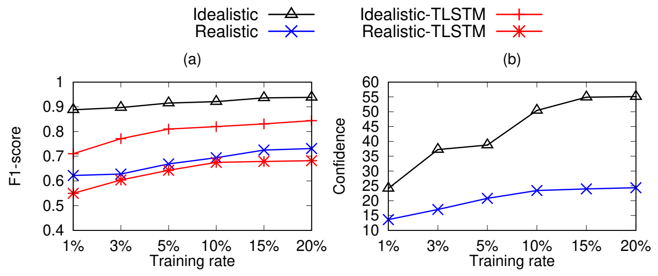


Figure 8: Replica currency estimation by varying training rates over Btcusd data.

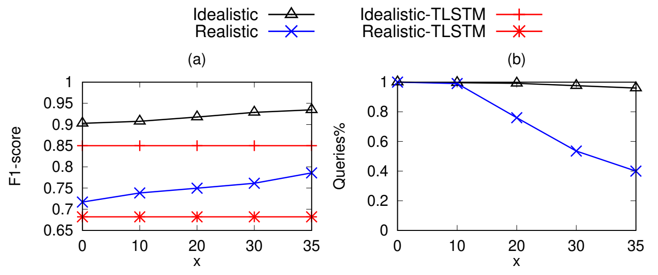


Figure 9: Replica currency estimation by varying confidence thresholds $\eta = 1 - e^{-x}$ over Btcusd data.

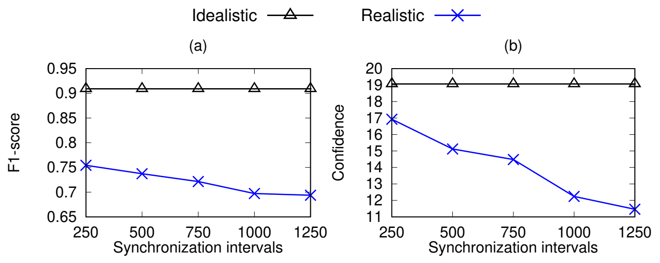


Figure 10: Varying time intervals between synchronizations intervals with two consecutive node synchronizations (e.g., T_{42} in Figure 5 and the one before it) using Btcusd data

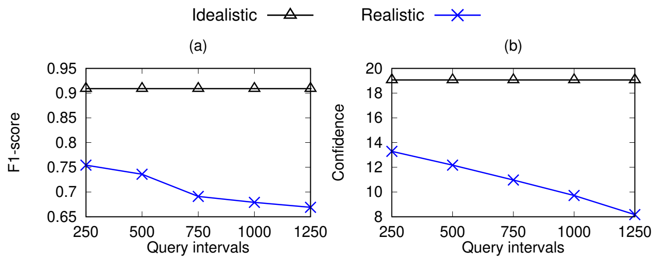


Figure 11: Varying time intervals between the latest node synchronization and query time (e.g., T_{42} and ζ in Figure 5)

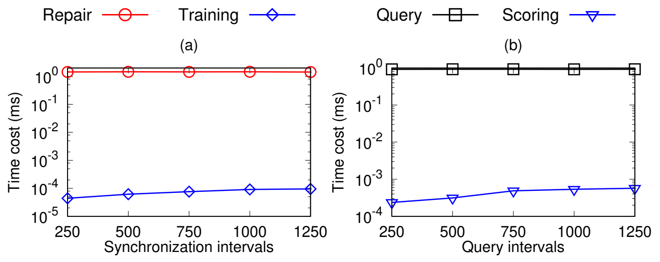


Figure 12: Time cost for varying synchronization and query time intervals using Btcusd data

is also not surprising that the corresponding currency estimation accuracy becomes lower. Figure 12(a) shows the time costs for read repair and training under various synchronization time intervals. The corresponding query and estimation scoring time costs are reported in Figure 12(b). Note that the estimation scoring and training time costs are only 0.006%-0.03% of the original costs in query processing and replica synchronization (i.e., during read repair).

5.4 Query Processing Evaluation

We evaluate currency estimation in query processing as introduced in Section 4.2 under various query loads.

5.4.1 Network Partition and Node Failure. We consider two query load scenarios: (i) network partitioning that specifies clusters to service reads or writes in separate nodes at specific times; and (ii) varying node failure times. Figure 13(a) shows the network partitioning scheme, where specific replicas service reads or writes exclusively for a specific time interval. Figure 13(b) shows the node failure scenario where both reads and writes are not serviced at offline/failure nodes, denoted as empty time intervals. Under the network partitioning case, Figure 14 shows that as we vary the time intervals of a network partition, the F1-score and the confidence decrease for longer network partitioning time intervals; making the prediction of the next version more challenging. Figure 15 shows a linear decline in F1 accuracy and confidence for increasing node down time. To study the impact of the number of failing nodes on currency estimation, Figure 16 shows, as expected, currency estimation becomes more difficult as more nodes become unavailable.

5.4.2 Skewed Updates and Accesses. We study the impact of varying read/write distributions on our estimation accuracy. Figure 17(a) shows homogeneous reads/writes across the replicas. Figure 17(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. Figure 18 shows the accuracy and confidence as we vary the skewness parameter S, where a value of S = 2.7 indicates the homogeneous case of the original dataset (Figure 17(a)). As expected, our estimation accuracy and confidence decline for increasing data skew as it becomes more difficult to learn an accurate model for currency estimation. Our model is adaptable to learn over data with periodic updates (regardless of the time periodicity) (Figure 21), with irregular updates (Figure 26 in [1]), and those with increasing data skew (Figure 18).

5.4.3 Write Consistency Level. We evaluate the impact of varying the number of write replicas, also known as the write consistency level. We simulate the number of write replicas using a Poisson distribution with parameter λ. Figure 19(a) shows the distribution

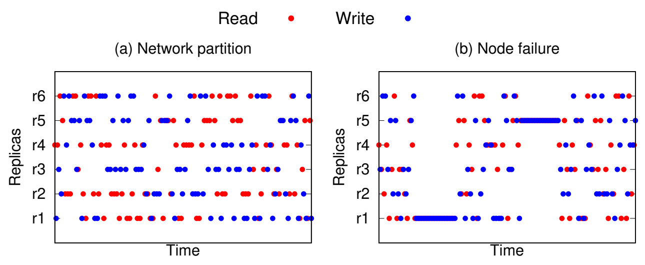


Figure 13: Query loads over (a) network partition with separate reads/writes and (b) node failure with suspended service

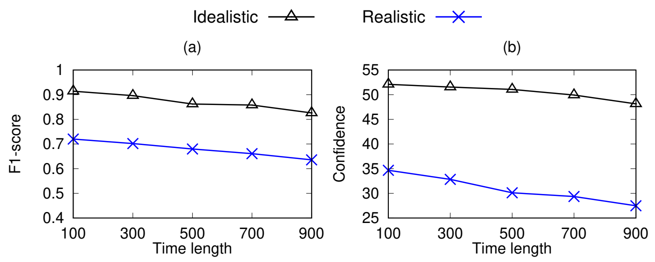


Figure 14: Replica currency estimation for varying time lengths of network partition (Figure 13(a)) using Btcusd data

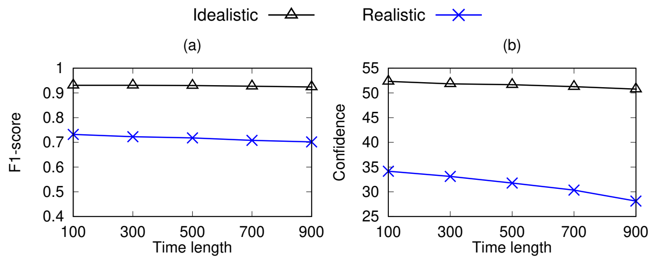


Figure 15: Replica currency estimation for varying node failure times (Figure 13(b)) using Btcusd data

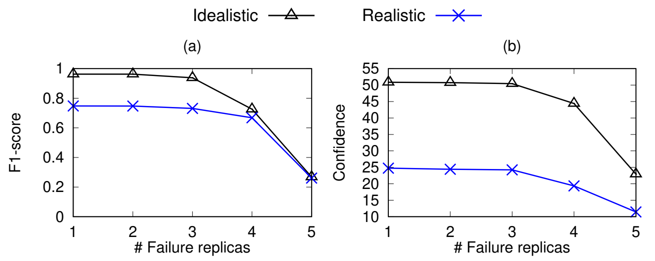


Figure 16: Varying number of node failures

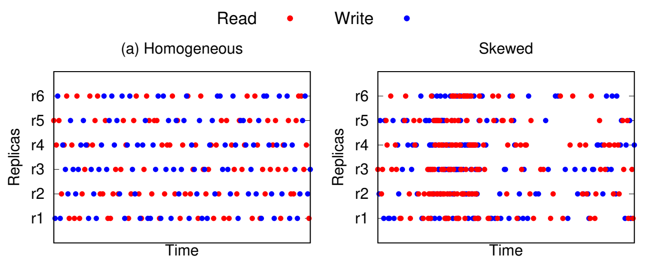


Figure 17: Query loads with (a) homogeneous and (b) skewed updates and reads

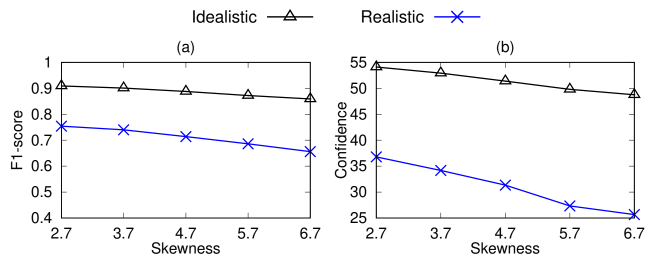


Figure 18: Replica currency estimation by varying the skewness of updates and accesses (Figure 17(b)) using Btcusd data

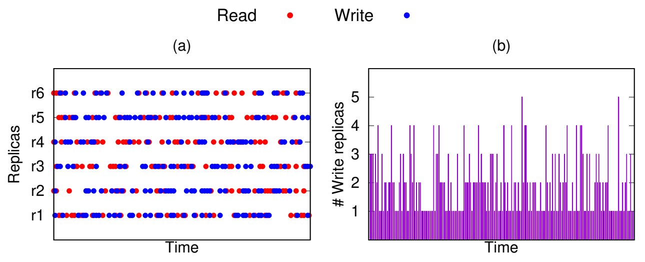


Figure 19: Query load by varying write consistency levels (varying number of write replicas)

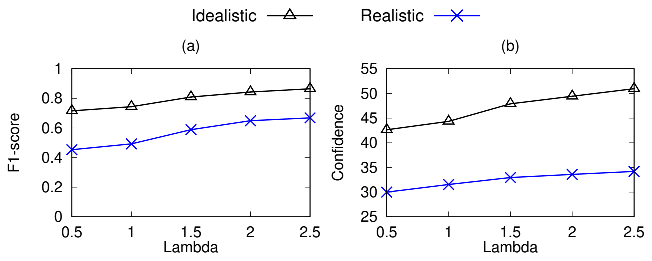


Figure 20: Replica currency estimation by varying write consistency levels, according to λ using Btcusd data

of reads/writes for each replica, and Figure 19(b) illustrates how writes are Poisson distributed across the number of write replicas. Figure 20 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 are able to learn and adapt to increases in λ, with corresponding increases in F1 score and confidence due to the larger number of write replicas.

5.5 DYNAMIC Read Consistency Level

In this experiment, we compare our DYNAMIC read consistency level proposed in Section 4 against the existing options (ONE, QUORUM, ALL) with ONE write, and with QUORUM writes. In such a scenario, QUORUM reads with QUORUM writes could guarantee the currency of results, but this is not guaranteed with ONE

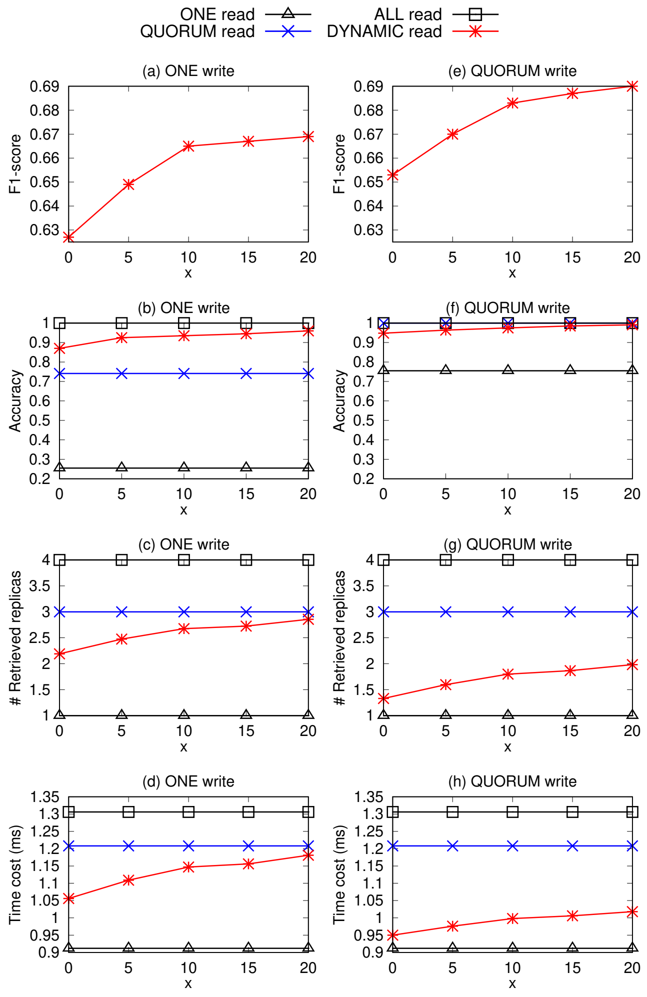


Figure 21: DYNAMIC consistency level under various confidence threshold $\eta = 1 - e^{-x}$ using Btcusd data

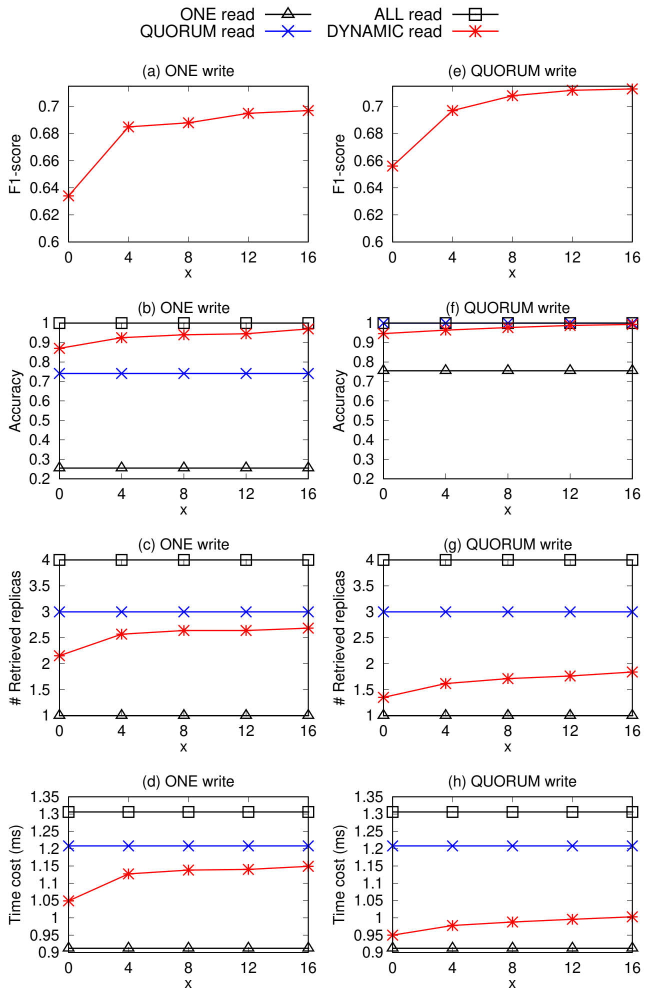


Figure 22: DYNAMIC consistency level under various confidence threshold $\eta = 1 - e^{-x}$ using Bike data

write.Instead of requiring a fixed number of nodes to respond to a query, the DYNAMIC consistency level immediately returns the response replicas if their confidence of being current satisfies threshold η. Since the existing consistency levels do not consider currency estimation, and all the returned results are assumed to be current, we compare their relative accuracy of being current.

Figures 21 and 22 show results of our DYNAMIC consistency level over the Btcusd and Bike datasets. Similar results are also observed in the other Sensor, Bank and Mental datasets described in Section 5.1, and presented in Figures 24, 25 and 26 in [1] due to limited space. Figures 21(a)-(d) present comparative results using ONE write, followed by QUORUM write in Figures 21(e)-(h). As expected, Figures 21(a) and 21(e) show that as we increase the confidence threshold, the F1 score of currency estimation for each replica increases, with higher accuracy using QUORUM write. In addition, Figure 21(b) and 21(f) report the accuracy of returning current replicas in query answering, which is guaranteed by ALL-read and QUORUM-read+QUORUM-write. As shown in Figure 21(b), the DYNAMIC consistency level achieves a higher accuracy of returning current answers than QUORUM (comparable to ALL) reads. The corresponding number of dynamically retrieved replicas as well as the time costs, however, are less than the fixed QUORUM in Figures 21(c) and 21(d), respectively. The reason is that our DYNAMIC consistency level immediately returns the answer after retrieving a high confidence current replica, whereas QUORUM reads must still wait for a fixed number of replicas to respond. Such an improvement is more significant with QUORUM write in Figures 21(g) and 21(h), since there exist more current replicas. The results across all datasets demonstrate that our DYNAMIC consistency level offers great opportunity for systems to decrease latency costs, and to lower query processing times.

Finally, we consider a real-application deployment with varying network performance and replica locations. We use the AWS cloud to geographically distribute cluster nodes across different regions to consider the communication delays corresponding to latencies within a region and across regions. Figure 23 shows the results as

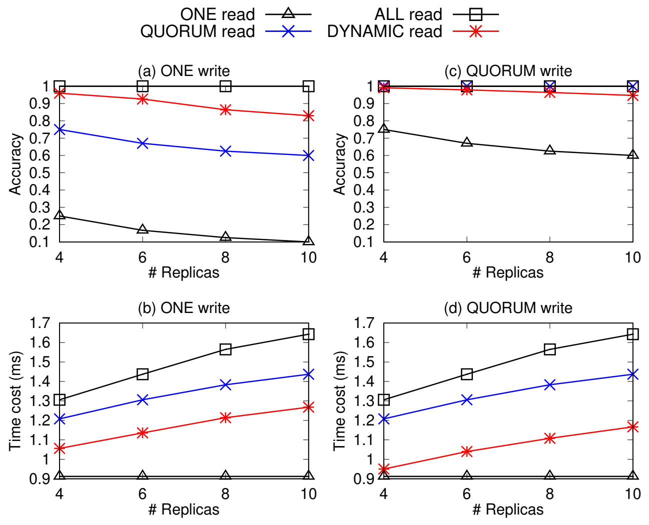


Figure 23: DYNAMIC consistency level under various number of servers using Btcusd data

we vary the numbers of servers located in different regions. It is not surprising that the time cost increases as the number of replicas increases, as shown in Figure 23(b) and Figure 23(d), which lead to the variance in production latency. Our DYNAMIC consistency level still achieves lower time cost (compared to QUORUM read). At the same time, more replicas also make the current replica rate smaller under ONE write and QUORUM write, leading to lower accuracy in Figure 23(a) and Figure 23(c).

6 RELATED WORK

6.1 Data Currency

Data currency is vital for correct and timely interpretation of data. Fan et. al, propose *currency constraints (CCs)* that declaratively define currency relationships according to the domain semantics [18, 19]. CCs are assumed to be given apriori, and elicit a partial currency order among tuples in a relation. For example, tuples t_1 and t_2 describing a person's status as 'working' and 'retired', respectively, indicate that t_1 precedes t_2. CCs have been used to determine currency orders for data values copied between data sources. Based on these copy relationships, CCs facilitate query answering involving only current data [20]. Extensions of this work include reasoning about CCs and their consistency, computation of certain answers regardless of the partial currency order, and studying whether copy functions can import sufficient data to answer a query [21]. CCs, however, rely on domain expertise, and are defined over static data instances. In more recent work, CurrentClean is a probabilistic framework for identifying and repairing stale cells by modeling spatio-temporal update patterns in the data [30]. Similar to our work, CurrentClean learns from a history of past updates to compute the probability that a cell value is current. However, our work considers replicated, multi-node settings where versioning of a cell value, and node unavailability can occur. We provide confidence bounded currency estimation in these settings.

6.2 Data Freshness

Existing work have studied protocols to satisfy user-given freshness guarantees (in the form of absolute time) using freshness locks [5], economic models that assign higher cost to recent updates [13], and controlling the read replica set to provide query answers [36]. A primary requirement and benefit of our work is to incur minimal overhead, making locking based solutions and re-calculation of read replica sets infeasible.

Probabilistically Bounded Staleness (PBS) [7] detects staleness for query results across a set of replicas. Despite this similar goal, there are several notable differences. First, PBS considers read requests where the last write was committed at least t time units ago, while our study does not have such constraints. Intuitively, PBS may report false positive cases where data is reported as current, but newer, uncommitted data exists since the last commit. We consider such cases in our work. Second, the definition and semantics of staleness are different. When PBS processes the read request (after t time units of the last write), a replica is said to be stale if a read arrives before a write. In contrast, we say a replica is stale if there exists another replica with a more recent (the latest) write.

6.3 Data Replication

Past work have proposed techniques to guarantee and improve data freshness under lazy-master replication settings [28, 32]. In eager replication, all replica copies are simultaneously updated using strategies such as quorum consensus [9] or two-phase-commit leading to mutual consistency and freshness [32]. In contrast, in lazy replication, deferred updates lead to varying data freshness levels across the replicas for unseen update transactions. Adaptive update policies take an on-demand approach based on access frequency, current transaction miss ratio, and system utilization [37]. Our work shares the same goal towards achieving data freshness (currency), but we do not focus on the update propagation strategy. In such systems, data freshness often depends on system conditions such as replica workload, frequency of updates, and network latency. However, our method provides confidence bounded estimations that are adaptive to these system conditions, and can be used together with existing solutions to reduce latency. By exchanging model parameters between nodes, we have shown lower overhead costs than message and update passing between nodes (about only 0.006%-0.03% of the original query processing and replica synchronization time costs).

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 5 and 6). 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 [2], 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 0.006%-0.03% 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. We believe that the replica currency estimation. and the DYNAMIC consistency level are extensible to other (NoSQL) databases, such as HBase [4], and intend to implement likewise such features as future work.

REFERENCES

[1] Full version. *https://sxsong.github.io/dynamic.pdf*.

[2] https://cassandra.apache.org/.

[3] https://github.com/illidanlab/t-lstm.

[4] https://hbase.apache.org/.

[5] 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.

[6] Apoorv Patne. Extracted bank account statements of various bank accounts. https://www.kaggle.com/apoorvwatsky/bank-transaction-data, 2019.

[7] 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.

[8] 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. ACM, 2017.

[9] P. A. Bernstein, V. Hadzilacos, and N. Goodman. *Concurrency Control and Recovery in Database Systems*. Addison-Wesley Longman Publishing Co., Inc., 1986.

[10] 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.

[11] 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.

[12] Cinnos Mission Critical. Data center readings. https://www.cas.mcmaster.ca/~zhengz13/Dataset/Sensor.rar, 2018.

[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] 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.

[15] 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.

[16] 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.

[17] W. Fan and F. Geerts. *Foundations of Data Quality Management*. Morgan & Claypool Publishers, 2012.

[18] W. Fan, F. Geerts, N. Tang, and W. Yu. Inferring data currency and consistency for conflict resolution. In *ICDE*, pages 470–481, 2013.

[19] W. Fan, F. Geerts, N. Tang, and W. Yu. Conflict resolution with data currency and consistency. *J. Data and Information Quality*, 5(1–2), 2014.

[20] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. In *PODS*, pages 71–82, 2011.

[21] W. Fan, F. Geerts, and J. Wijsen. Determining the currency of data. *TODS*, 37(4):25:1–25:46, 2012.

[22] F. Fatemi. Best practices for data hygiene. *Forbes*, 2019.

[23] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Dunson, A. Vehtari, and D. B. Rubin. *Bayesian data analysis*. CRC press, 2013.

[24] L. George. *HBase - The Definitive Guide: Random Access to Your Planet-Size Data*. O'Reilly, 2011.

[25] I. J. Goodfellow, Y. Bengio, and A. C. Courville. *Deep Learning*. Adaptive computation and machine learning. MIT Press, 2016.

[26] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

[27] Hristo Mavrodiev. London bike sharing dataset. https://www.kaggle.com/hmavrodiev/london-bike-sharing-dataset, 2019.

[28] 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.

[29] A. Lakshman and P. Malik. Cassandra: a decentralized structured storage system. *Operating Systems Review*, 44(2):35–40, 2010.

[30] M. Milani, Z. Zheng, and F. Chiang. Currentclean: Spatio-temporal cleaning of stale data. *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 172–183, 2019.

[31] Open Sourcing Mental Illness, LTD. Mental health in tech survey. https://www.kaggle.com/osmi/mental-health-in-tech-survey, 2016.

[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] Carsten. 400+ crypto currency pairs at 1-minute resolution. https://www.kaggle.com/tencars/392-crypto-currency-pairs-at-minute-resolution, 2020.

[35] M. J. Wooldridge. *Foundations of Machine Learning*. MIT Press,, 2012.

[36] 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.

[37] Yuan Wei, S. H. Son, and J. A. Stankovic. Maintaining data freshness in distributed real-time databases. In *Euromicro Conference on Real-Time Systems*, pages 251–260, 2004.

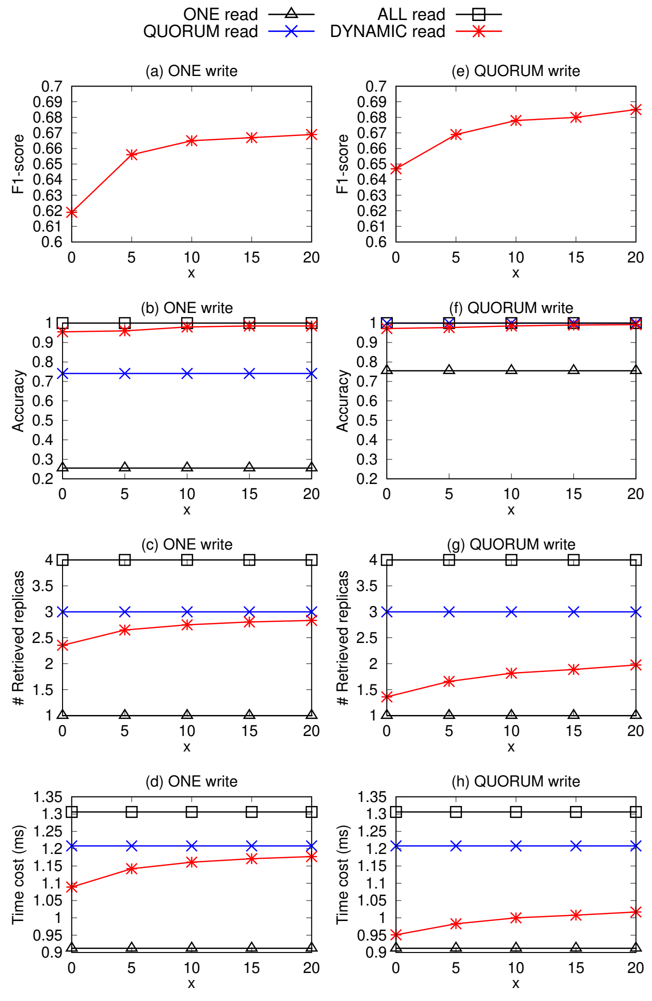


Figure 24: DYNAMIC consistency level under various confidence threshold $\eta = 1 - e^{-x}$ using Sensor data

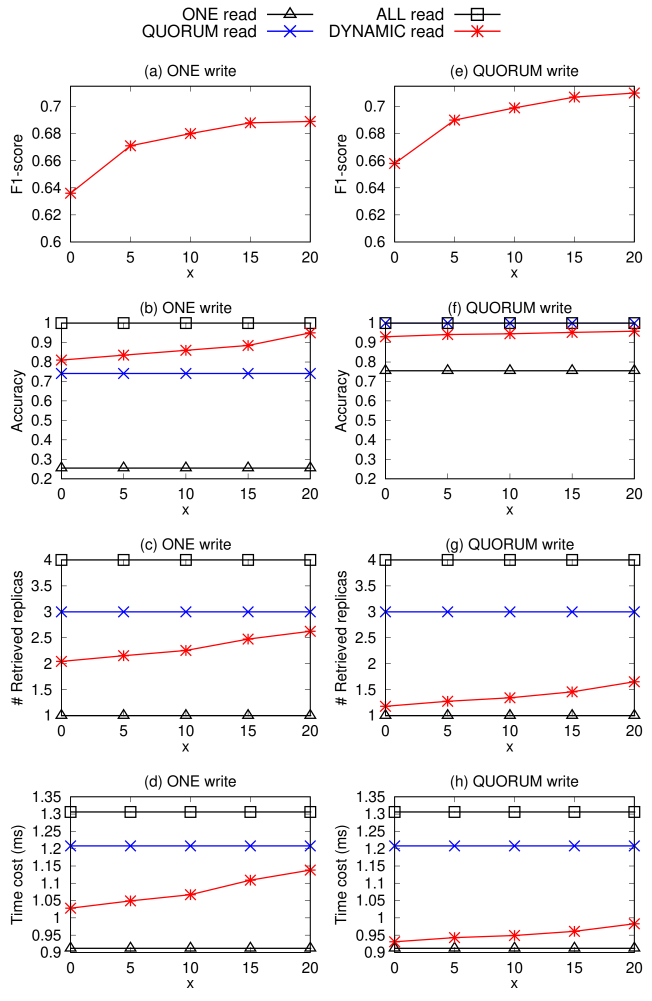


Figure 25: DYNAMIC consistency level under various confidence threshold $\eta = 1 - e^{-x}$ using Bank data

A ADDITIONAL EXPERIMENTS

B PROOFS

B.1 Proof of Proposition 2

According to Proposition 1, we have

$$P(|t_{n+1} - t'_{n+1}| < \gamma) > 1 - \delta \tag{22}$$
$$\Rightarrow P(-\gamma < t_{n+1} - t'_{n+1} < \gamma) > 1 - \delta$$
$$\Rightarrow P(t'_{n+1} - \gamma < t_{n+1} < t'_{n+1} + \gamma) > 1 - \delta.$$

Given $T'_{n+1} = T_n + t'_{n+1}$ and $T_{n+1} = T_n + t_{n+1}$, it follows

$$P(T'_{n+1} - \gamma < T_{n+1} < T'_{n+1} + \gamma) > 1 - \delta. \tag{23}$$

As shown in Figure 4, 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

$$\zeta = T'_{n+1} - \gamma, \tag{24}$$

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.

Referring to Formula 23, the corresponding probability has

$$P(\zeta = T'_{n+1} - \gamma < T_{n+1}) \geq P(T'_{n+1} - \gamma < T_{n+1} < T'_{n+1} + \gamma)$$
$$> 1 - \delta.$$

We combine Formula 24 and the definition of γ (Formula 5) to get,

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

The conclusion is proved.

B.2 Proof of Proposition 3

Similar to the proof of Proposition 2, as illustrated in Figure 4(b), if the query time has

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

we have $T_{n+1} \leq \zeta = T'_{n+1} + \gamma$, i.e., the version v_n is stale as defined in Formula 1. Similarly, referring to Formula 23 in the proof of

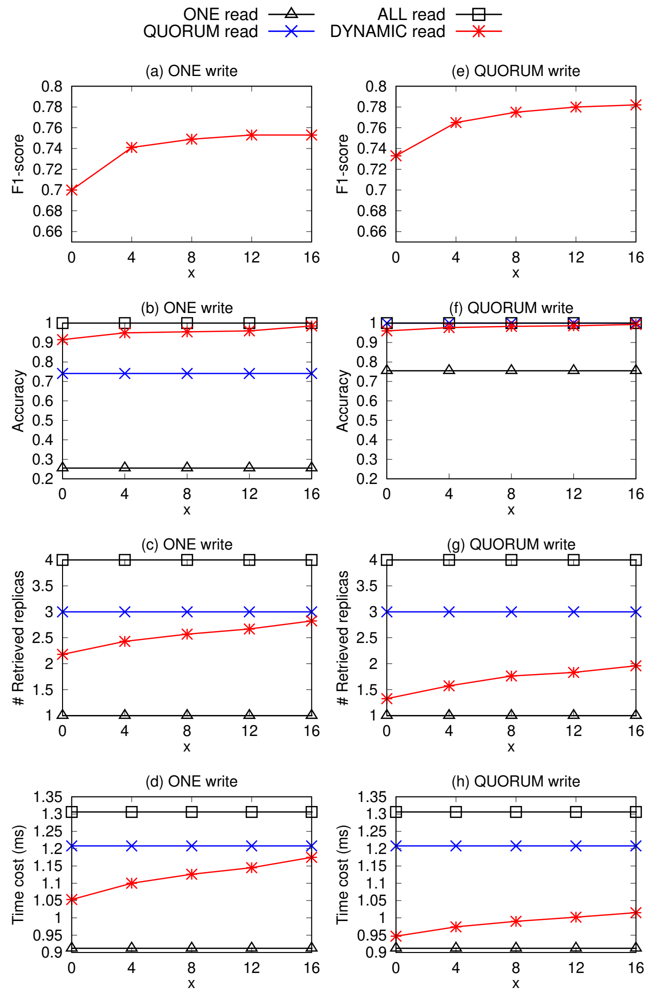


Figure 26: DYNAMIC consistency level under various confidence threshold $\eta = 1 - e^{-x}$ using Mental data

Proposition 2, the corresponding probability is

$$P(T_{n+1} \leq \zeta = T'_{n+1} + \gamma) > P(T'_{n+1} - \gamma < T_{n+1} < T'_{n+1} + \gamma) \\ > 1 - \delta.$$

Finally, we combine Formula 24 and Formula 5 (definition of γ) to get,

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

The conclusion is proved.

B.3 Proof of Lemma 4

As shown in Figure 6, by combining Formulas 10 and 11, we have

$$P\Big((\phi_{q1})^{m-w} t_{qw} + (\phi_{q0} - \gamma) \sum_{l=0}^{m-w-1} (\phi_{q1})^l < t_{qm} < \\ (\phi_{q1})^{m-w} t_{qw} + (\phi_{q0} + \gamma) \sum_{l=0}^{m-w-1} (\phi_{q1})^l\Big) > (1-\delta)^{m-w}. \quad (25)$$

Since $T'_{qm} = T_{qw} + \sum_{k=w+1}^{m} t'_{qk}$, the generalization error bound of T'_{qm} is guaranteed by considering the theoretical bound of all the previous time difference predictions t'_{qk}, $w < k \leq m$.

$$P\Big(T_{qw} + \sum_{k=2}^{m-w+1} \big((\phi_{q1})^{k-1} t_{qw} + (\phi_{q0} - \gamma) \sum_{l=0}^{k-2} (\phi_{q1})^l\big) < T_{qm} < \\ T_{qw} + \sum_{k=2}^{m-w+1} \big((\phi_{q1})^{k-1} t_{qw} + (\phi_{q0} + \gamma) \sum_{l=0}^{k-2} (\phi_{q1})^l\big)\Big) \\ > (1-\delta)^{\frac{(m-w)(m-w+1)}{2}}. \quad (26)$$

The conclusion is thus proved by introducing the definition of T'_{qm} in Formula 13, β_{qm} in Formula 15 and γ_q in Formula 16.

B.4 Proof of Proposition 5

According to Lemma 4, we have

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

As shown in Figure 7, the time stamp T_{qm} is in the range of $(T'_{qm} - \beta_{qm}\gamma_q, T'_{qm} + \beta_{qm}\gamma_q)$ with probability at least $(1-\delta)^{\frac{(m-w)(m-w+1)}{2}}$. If the query time ζ has

$$\zeta = T'_{qm} - \beta_{qm}\gamma_q,$$

i.e.,

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

as illustrated in Figure 7(a), we have $\zeta = T'_{qm} - \beta_{qm}\gamma_q < T'_{qm}$. Referring to Formula 26, the corresponding confidence has

$$P(\zeta = T'_{qm} - \beta_{qm}\gamma_q < T_{qm}) \\ \geq P(T'_{qm} - \beta_{qm}\gamma_q < T_{qm} < T'_{qm} + \beta_{qm}\gamma_q) \\ > (1-\delta_\zeta)^{\frac{(m-w)(m-w+1)}{2}} \quad (27)$$

Similarly, according to Lemma 4, we also have

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

As shown in Figure 7, the time stamp $T_{q,m-1}$ is in the range of $(T'_{q,m-1} - \beta_{q,m-1}\gamma_q, T'_{q,m-1} + \beta_{q,m-1}\gamma_q)$ with probability at least $(1-\delta_{m-1})^{\frac{(m-w-1)(m-w)}{2}}$. If the retrieved replica version v_{pn} has

$$T_{pn} = T'_{q,m-1} + \beta_{q,m-1}\gamma_q,$$

i.e.,

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

as illustrated in Figure 7(a), we have $T_{pn} = T'_{q,m-1} + \beta_{q,m-1}\gamma_q \geq T_{q,m-1}$. Referring to Formula 26, the corresponding confidence has

$$P(T_{pn} = T'_{q,m-1} + \beta_{q,m-1}\gamma_q \geq T_{q,m-1}) \\ > P(T'_{q,m-1} - \beta_{q,m-1}\gamma_q < T_{q,m-1} < T'_{q,m-1} + \beta_{q,m-1}\gamma_q) \\ > (1-\delta_{m-1})^{\frac{(m-w-1)(m-w)}{2}} \quad (28)$$

Combining Formulas 27 and 28 leads to the conclusion.

B.5 Proof of Proposition 6

Similar to the proof of Proposition 5, if the query time ζ has

$$\zeta = T'_{qm} + \beta_{qm}\gamma_q,$$

i.e.,

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

as illustrated in Figure 7(b), we have $\zeta = T'_{qm} + \beta_{qm}\gamma_q \geq T_{qm}$. Referring to Formula 26, the corresponding confidence has

$$\begin{aligned} &P(\zeta = T'_{qm} + \beta_{qm}\gamma_q \geq T_{qm}) \\ >&P(T'_{qm} - \beta_{qm}\gamma_q < T_{qm} < T'_{qm} + \beta_{qm}\gamma_q) \\ >&(1 - \delta_\zeta)^{\frac{(m-w)(m-w+1)}{2}}. \end{aligned} \tag{29}$$

And for the retrieved version v_{pn}, if its timestamp T_{pn} has

$$T_{pn} = T'_{qm} - \beta_{qm}\gamma_q,$$

i.e.,

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

as illustrated in Figure 7(b), we have $T_{pn} = T'_{qm} - \beta_{qm}\gamma_q < T_{qm}$. Referring to Formula 26, the corresponding confidence has

$$\begin{aligned} &P(T_{pn} = T'_{qm} - \beta_{qm}\gamma_q < T_{qm}) \\ \geq&P(T'_{qm} - \beta_{qm}\gamma_q < T_{qm} < T'_{qm} + \beta_{qm}\gamma_q) \\ >&(1 - \delta_m)^{\frac{(m-w)(m-w+1)}{2}} \end{aligned} \tag{30}$$

Combining Formulas 29 and 30 leads to the conclusion.