Scalable Load Distribution for View Divergence Control of Data Freshness in Replicated Database Systems

TAKAO YAMASHITA†

A scalable load distribution method for view divergence control of statistically defined data freshness in replicated database systems is proposed. This method enables a number of mobile and fixed client nodes to continue their processes even when they lose connectivity to a network or do not have sufficient bandwidth to meet application requirements, which is very likely to occur to mobile client nodes. This can be achieved by renewing copies of data in client nodes while they maintain connectivity to a network so that their copies of data are sufficiently fresh to meet application requirements while they lose network connectivity. The load distribution for view divergence control is achieved by determining multiple sets of replicas from which client nodes retrieve the values of data through read transactions. Client nodes calculate the value of data that reflect updates which have already reached one or more elements in the determined set of replicas. We show that our method reduces the load of processing read transactions to less than about 1/40 of that in the original method in order to improve data freshness to about 2/5 of the maximum update delay in a large-scale network.

1. Introduction

With the progress of data networks, distributed computers in a network share a variety of data. In ubiquitous computing, distributed computer nodes have to update and retrieve data according to application requirements by autonomously cooperating with each other. To process transactions to update and retrieve data with high scalability and availability, data replication techniques are effective.

Data replication techniques are classified into two types: eager and lazy. Eager replication synchronously updates data of multiple replicas by using read-one-write-all (ROWA), ROWA available, or quorum consensus. In lazy replication, updates are gradually reflected among all replicas after they are initially processed by one replica. Lazy replication is suitable to achieve high scalability and availability because the overhead of processing updates is lighter in lazy replication than in eager replication. Lazy replication is further divided into two types: lazy-master and lazy-group. In lazy-master replication, one replica can initially process updates, while any replica can initially process updates in lazy-group replication. Lazy-master replication can accomplish strict data consistency, while its performance of processing updates is lower than that of lazy-group replication. On the contrary, lazy-group replication can achieve high performance of processing updates, while it cannot offer strict data consistency. Lazy-group replication is a very important technique because there are a number of applications that allow weak consistency, but require high scalability.

The freshness of data in lazy-group replication depends on replicas because updates are gradually propagated among replicas. On the other hand, there are applications that require a particular degree of data freshness such as decision-support systems. Though it is important to control data freshness for client nodes in any environment, it is more important in mobile computing environments. When a mobile node moves from one geographic location to another, its network location may change. While it is moving, a mobile node may lose connectivity to a network or may not have sufficient bandwidth during a particular time period. Therefore, it has to make a copy of data in itself so that it can continue its processes even when it does not have sufficient connectivity to a network.

As a result, mobile nodes running such applications need to autonomously retrieve data with various degrees of freshness according to their connectivity to a network. In Fig. 1, there are a number of replicas, and mobile node c₁, which is a client, moves from (a) to (b), (c), and (d) as time elapses. When mobile node c₁ is in (a), it retrieves data from the closest replica, 11, through a front-end node between times t₁ and t₂. Mobile node c₁ has to make a copy of data whose freshness satisfies application re-
requirements between times $t_2$ and $t_3$ because it cannot access replicas between times $t_2$ and $t_3$.

We have proposed an optimal view divergence control (OVDC) method for lazy-group replication, which enables computers to retrieve data with a required degree of freshness that is statistically defined\(^{14}\). This method can be used in any computing environments. View divergence of data freshness means a difference in the recentness of data obtained by clients. The OVDC method determines a set of replicas called a read replica set. A computer obtains the values of a data object in all the elements of the read replica set through read transactions. It then calculates the value of a data object that reflects updates which have already reached replicas accessed by the computer. The OVDC method determines only one read replica set whose size is minimum, the minimum read replica set. However, especially in ubiquitous computing, a number of mobile nodes need to autonomously retrieve data with various degrees of freshness. Therefore, it is necessary to distribute the load of processing read transactions by determining multiple read replica sets.

In this paper, we propose a scalable load distribution method to control the view divergence of data freshness for lazy-group replication. Our method determines multiple read replica sets for the load distribution of processing read transactions. This is achieved by using the characteristics of the change in the size of a read replica set determined by the OVDC method depending on data freshness.

The remainder of this paper is organized as follows. Section 2 briefly describes the OVDC method. In Section 3, we describe our method to distribute the load of processing read transactions scalably. In Section 4, we evaluate by simulation the efficiency of the load distribution achieved by the proposed method. In Section 5, we compare our method with related work. Section 6 concludes the paper.

2. Optimal View Divergence Control

2.1 System Architecture

In the OVDC method, there are four types of nodes: client, front-end, control, and replica, of which only client nodes can be mobile. Replicas are divided into replica groups to improve availability and reliability. Replicas in the same group are synchronously updated by using an eager replication method, such as ROWA, ROWA available, or quorum consensus\(^{4,10}\)\(^{-13}\). We use a tree-topology network for update propagation among replica groups to estimate update delay in a simple manner. A tree-topology network is designed by administrators according to the distances between replica groups in a physical network. The OVDC method uses lazy-group replication to update replica groups, where updates are initially processed by any replica group and then asynchronously propagated among replica groups. Updates are propagated along edges in a tree-topology network.

In this system, there are two types of transactions: update and read. An update transaction requested by a client is sent to a front-end node. The front-end node sends it to a replica group. The update is then propagated to all replica groups as a refresh transaction. A read transaction requested by a client is also sent to a front-end node. The front-end node sends a read transaction to multiple replica groups. After the front-end node receives in reply the data and, if necessary, recent logs of updates received by each replica groups, it calculates data to reflect all the updates that the accessed replica groups have received by using this information and the timestamps associated with the data. The data calculated by the front-end node is sent to the client that issues the read transaction. A front-end node selects multiple replicas to which it sends read transactions so that the acquired data value satisfies the recentness specified by a client. The set of replica groups that are selected by front-end nodes is calculated by control nodes using the statistically estimated delays between pairs of replica groups, which are gathered by the control nodes from replica groups. Messages including the statistically estimated delays and the set of replica groups selected by
front-end nodes are directly transmitted from replica groups to control nodes and from control nodes to front-end nodes, respectively. In the remainder of this paper, we refer to a replica group as a replica.

We assume a scale for our system that requires 100–1,000 replicas depending on the types of applications. The number of mobile and fixed client nodes at this scale also depends on the types of applications. For example, we consider the number of client nodes when our system is used in decision-support systems. The performance of a database system is frequently measured using Transaction Processing Performance Council (TPC)\textsuperscript{15} benchmark tests. The TPC Benchmark H (TPC-H\textsuperscript{2}) is used to measure the performance of decision-support systems. The benchmark results of TPC-H for a 100-GB database indicate about $2.5 \times 10^3$–$1.2 \times 10^4$ queries per hour depending on the hardware and software. If every user issues one query per hour, then the assumed number of client nodes in our system is in the range between about $2.5 \times 10^5$ and $1.2 \times 10^7$.

Our method uses the probabilistic recentness of updates reflected in the acquired data, called read data freshness (RDF) because the number of updated replicas increases probabilistically. The degree of RDF, $T_p$, represents a client’s required view divergence. Its formal definition is $T_p = t_c - t$ if all the updates invoked before time $t$ somewhere in a network are statistically estimated to be reflected in data acquired by a client with probability $p$, where $t_c$ is the present time. This means that if the last update reflected in the acquired data was invoked at $t_l < t$, no updates are invoked between $t_l$ and $t$ with probability $p$. In other words, updates invoked between $t_l$ and $t$ will not arrive at any replicas with probability $p$. Data with degree of RDF $T_p$ are termed data within $T_p$ in this paper.

In the OVDC method, we assume that we can statistically estimate the propagation delay of updates from the time replicas initially process them to the time they reach other replicas. This can be accomplished by using clock synchronization techniques\textsuperscript{16} among replicas and nonparametric estimation methods\textsuperscript{14,17}.

An example of the system architecture for the OVDC method is shown in Fig. 1. When client $c_2$ sends an update to replica 8 through a front-end node, it is processed by replica 8 and then asynchronously propagated to replicas 3, 9, 10, and then 1 and 11. An update is eventually propagated among all replicas. When client $c_3$ reads the value of a data object through a front-end node, it sends read transactions to replicas 2, 14, and 19. The front-end node then calculates the data value reflecting all updates received by those replicas and sends it to client $c_3$.

### 2.2 Behavior in the Event of Network Failure

When network failures occur and redundant paths exist in a network, a routing function can find a redundant path\textsuperscript{18,19}. This means that an underlying network function can restore packet transmission among replicas when network failures occur. When no redundant path exists, update propagation among separated replicas is suspended until recovery from network failures is achieved. The change in physical path for update propagation by a routing function and suspended update propagation increase update delay. When the change in update delay and the duration of network failures are reflected in the statistical information of update delay, our method selects read replicas to provide data values satisfying the degree of freshness requested by clients. In other words, the more correct the statistical information about network failures, the more precisely our method can provide data values with the degree of freshness requested by clients because the statistical estimation of update delay can be performed precisely using this information.

While network partitioning occurs due to there being no redundant path, a front-end node may not be able to access one or more replicas in the calculated read replica set. This means that it is essentially difficult to completely achieve view divergence control during network partitioning, even if we use a network with more redundant paths between replicas for update propagation instead of a tree-topology network. From this point of view, when applications require data freshness, we should design a sufficiently reliable network for packet transmission. However, our method can improve the probability that a front-end node can access read replicas during network partitioning by determining multiple read replica sets, as described in this paper. In addition, our method enables applications to determine the
uncertainty of their calculated results by notifying them of which replicas are not covered by read replicas that are accessible from a front-end node.

2.3 Terminology
To explain our algorithm, we define the following four terms.
• read replica (set)
• range originator
• classified replica
• mandatory replica

A read replica is a replica to which a client node sends a read transaction to obtain the value of a data object. We denote the set of read replicas used to obtain data with the required RDF as a read replica set. A range originator of replica \( r \) is a replica that originates a refresh transaction and from which the transaction can reach \( r \) within time \( T_p \) with probability \( p \) when the degree of RDF requested by a client is \( T_p \). This means that the upper confidence limit of the delay from a range originator to replica \( r \) with probability \( p \) is \( T_p \). Let \( O_i \) be the set of range originators of replica \( i \). We say that replica \( j \) is covered or uncovered by replica \( i \) if \( j \in O_i \) or \( j \not\in O_i \), respectively. A classified replica is a replica whose set of range originators is not a subset of those of any other replica. A mandatory replica is a classified replica that has one or more range originators that are not range originators of any other classified replicas.

2.4 Calculation of a Minimum Read Replica Set
In this section, we describe an algorithm for calculating a minimum read replica set used in the OVDC method when an update is propagated to replicas connected by a tree. When the delay between replicas is given by a probability distribution and an upper confidence limit, the problem of finding a read replica set has different properties from those of the problems of weighted graphs in graph theory. For example, when there is a replica, \( j \), on the path from replica \( i \) to replica \( k \), the distance from replica \( i \) to replica \( k \) is the sum of the distances from replica \( i \) to replica \( j \) and from replica \( j \) to replica \( k \) in the weighted graph. However, let \( d_{xy} \) be the upper confidence limit of the delay time from replica \( x \) to replica \( y \) with probability \( p \). Then, \( d_{ik} \) is not the sum of \( d_{ij} \) and \( d_{jk} \). For another example, when there is any replica \( k \) on the paths from replica \( i \) to replica \( l \) and from replica \( j \) to replica \( l \) and when \( d_{ik} \leq d_{jk} \), it is not always true that \( d_{il} \leq d_{jl} \).

The algorithm used in the OVDC method calculates a minimum read replica set, which is proved by using two properties of the upper confidence limit of delay in a tree. The first property is that if \( d_{ik} \leq T_p \), then \( d_{ij} \leq T_p \) and \( d_{jk} \leq T_p \) for any replica \( j \) on the path from replica \( i \) to replica \( k \). The second property is that if \( d_{ij} > T_p \), then \( d_{ik} > T_p \) for any replica \( k \) that satisfies the condition that replica \( j \) is on the path from replica \( i \) to replica \( k \).

Our algorithm incrementally constructs a minimum read replica set \( R \) composed of only classified replicas. This algorithm is shown in Fig. 2. \( T_o \) is the original tree for update propagation and \( T_s \) is a subtree of \( T_o \) constructed by our algorithm. \( V, S, C, \) and \( M \) are the sets of replicas covered by \( i \in R \), replicas in \( T_s \), classified replicas for \( T_s \), and mandatory replicas in each iteration, respectively. Step (2) removes replicas covered by replicas in \( R \) from \( O_i \). Step (3) identifies the classified replicas for \( T_s \). Step (4) removes the classified replicas that have the same range originators. Step (5) calculates the mandatory replicas. In step (6), all the mandatory replicas are added to \( R \), because a mandatory replica has at least one replica that is not included in the range originators of any classified replicas. Step (8) constructs a minimum subtree \( T_s \) including all the uncovered replicas, assuming that the weight of all edges is 1. Steps (2) through (9) are iterated until there is at least one replica that covers all the replicas uncovered by any replica \( i \in R \) in \( T_s \). One of these replicas that covers all the replicas uncovered by any replica is added to \( R \) by step (10), and then the algorithm terminates.

\[
A := \{ i \mid i \text{ is a replica } \}; \\
R := \{ \emptyset \}, V := \{ \emptyset \}; \\
S := A; \\
\text{while } (\forall i \in A \setminus V : O_i \neq A \setminus V); \\
\text{for all } i \in S, O_i := O_i \setminus V; \\
C := \{ i \mid i \in S \land \forall j \in S : O_i \not\subset O_j \}; \\
\text{for all } i, j \in C (i \neq j) : \\
\text{if } O_i = O_j \text{ then } C := C \setminus \{ j \}; \\
M := \{ i \mid \exists j \in A \setminus V : \forall k \in C (k \neq i) : \\
\quad j \in O_i \land j \not\in O_k \}; \\
R := R \cup M; \\
V := V \cup \{ i \mid i \in O_j \land j \in M \}; \\
T_s := \text{constructMinimumTree}(A \setminus V); \\
S := \text{a set of replicas in } T_s; \\
R := R \cup \{ i \}, \text{ where } O_i = A \setminus V; \\
\]
3. Load Distribution for View Divergence Control

The OVDC method calculates one minimum read replica set. Read transactions are processed by all the elements of the calculated minimum read replica set. To distribute the load of processing read transactions, we have to determine multiple read replica sets. When clients request the RDF of $T_p$, requested by a client, but that the delay along a path consisting of more than one edge with probability $p$ is more than $T_p$. In this example, the algorithm terminates in two iterations. In the first iteration illustrated in Fig. 3 (a), replicas 2, 7, 10, 14, 17, and 22 are selected as mandatory replicas, which are read replicas. In the second iteration illustrated in Fig. 3 (b), replicas 4 and 13 becomes read replicas. This example is described in more detail in our previous work\textsuperscript{14}.

3.1 Change in the Size of the Minimum Read Replica Set

The OVDC method calculates a minimum read replica set. Therefore, the size of the minimum read replica set is monotonically decreasing as the degree of RDF increases.

**Proof:** Let $T_{p_1}$ and $T_{p_2}$ be the degrees of RDF, and $N_1$ and $N_2$ be the size of the minimum read replica sets when the degrees of RDF are $T_{p_1}$ and $T_{p_2}$, respectively, where $T_{p_1}$ is less than $T_{p_2}$. Assume that $N_1$ is less than $N_2$. A read replica set that satisfies $T_{p_1}$ also satisfies $T_{p_2}$ with probability $p$ or more. Hence, the size of the minimum read replica set when the degree of RDF is $T_{p_2}$ should be $N_1$. This contradicts the above assumption. In addition, because the diameter of a network is finite, the size of the minimum read replica set becomes 1 when the degree of RDF is more than a particular value. Hence, the size of the minimum read replica set is monotonically decreasing as the degree of RDF increases.

When the degree of RDF is very small, the size of the minimum read replica set rapidly decreases as the degree of RDF increases\textsuperscript{14}. When the degree of RDF is greater than a particular value, the size of the minimum read replica set gradually decreases as the degree of RDF increases\textsuperscript{14}. Finally, the size of the minimum read replica set converges to 1. In Section 4, we demonstrate how the size of the minimum read replica set changes with the evaluation of the proposed load distribution method.

To calculate multiple read replica sets that satisfy the degree of RDF with probability $p$, we consider two cases: the single-element and multiple-element cases. Let $T_p^{(m)}$ be the minimum degree of RDF where the minimum read replica set calculated by the OVDC method contains one element. In addition, let $r_p^{(m)}$ be the read replica when the degree of RDF is $T_p^{(m)}$. The single-element case is where the degree of RDF is equal to or more than $T_p^{(m)}$. The multiple-element case is where the degree of RDF is less than $T_p^{(m)}$. We describe how to determine alternate read replica sets in the single-element and multiple-element cases in Sections 3.2 and 3.3, respectively.

3.2 Alternatives in the Single-element Case

In the single-element case, a tree-topology network for update propagation is divided into subtrees composed of an adjacent replica of $r_p^{(m)}$ and its descendants, where we assume that $r_p^{(m)}$ is the root of a network, as shown in Fig. 4. In the figure, $S_i$ denotes the subtree composed of an adjacent replica $i$ of $r_p^{(m)}$ and its descen-
dants. Assume that clients request the RDF of $T_p$ with probability $p$. When $T_p - T_p^{(m)}$ is greater than a particular value, updates originated by replicas in $S_j$ can reach replica $i$ within $T_p$ with probability $p$ or more for any adjacent replica $j$ of $r^{(m)}$ except for replica $i$. On the other hand, when $T_p$ is equal to or more than $T_p^{(m)}$, updates originated by replicas in $S_i$ can reach replica $i$ within $T_p$ with probability $p$ or more according to the first property of the upper confidence limit of delay in a tree, which is described in Section 2. As a result, replica set \{i\} becomes an alternate read replica set for $T_p$. Therefore, the greater the value of $T_p$, the greater the number of alternate read replica sets. In addition, the computational complexity to determine alternate read replica sets in the single-element case is contained in that of the algorithm used in the OVDC method because alternate read replica sets are determined as replicas whose range originator sets contain all replicas in step (4) of the algorithm shown in Fig. 2.

An example of the single-element case is shown in Fig. 5, where we assume for simplicity that the update delay along every edge has the same probability density function. In this example, $T_p^{(m)}$ and $r^{(m)}$ are the delay along four edges with probability $p$ and replica 4, respectively. When the degree of RDF requested by clients is $T_p$ with probability $p$, and $T_p$ is equal to the delay along five edges with probability $p$, the OVDC method designates one read replica, which is replica 1. When $T_p$ is equal to the delay along five edges, updates originated by replicas in areas $A$ and $B$ can reach replicas 14 and 1, respectively, at least after they have passed through replica 4 within $T_p$ with probability $p$. In addition, updates originated by replicas in areas $A$ and $B$ can reach replicas 1 and 14, respectively, within $T_p$ with probability $p$ or more. Therefore, replica sets \{1\}, \{4\}, and \{14\} are alternate read replica sets.

We roughly analyze the relationship between the number of alternate read replica sets and the degree of RDF to estimate the region of RDF where sufficient load distribution is possible. Assume that the degree of all nodes except for leaf nodes in a network is $N_d$. When the degree of RDF is $T_p^{(m)}$, the set of replicas $\{r^{(m)}\}$ is the read replica set, from its definition.

Node $r^{(m)}$ has $N_d$ adjacent replicas because its degree is $N_d$, as shown in Fig. 6. In general, when updates from a set of replicas, $g$, reach replica $r$ within $D_p$ with probability $p$ and can further reach all replicas whose hop count distance from $r$ is $c$ within $D_p^{(c)}$ with probability $p$ or more, we call $c$ the equivalent hop count from $D_p$ to $D_p^{(c)}$ for $g$. When the equivalent hop count from $T_p^{(m)}$ to $T_p$ for the set of all replicas is 1, the set of a replica composed of either replica $r^{(m)}$ or one of its adjacent replicas is an alternate read replica set for $T_p$. This means that the number of alternate read replica sets is at least $1 + N_d$. Adjacent replicas of $r^{(m)}$ have $N_d - 1$ adjacent replicas except for $r^{(m)}$. When the equivalent hop count from $T_p^{(m)}$ to $T_p$ is 2, the number of alternate read replica sets is at least $1 + N_d + N_d(N_d - 1)$. Similarly, when the equivalent hop count from $T_p^{(m)}$ to $T_p$ is $c$, the number of alternate read replica sets is at least $1 + \sum_{i=1}^{c} N_d(N_d - 1)^{i-1}$. From this rough analysis, the increase in the equivalent hop count rapidly increases the number of alternate read replica sets. Therefore, the number of alternate read replica sets rapidly increases as $T_p$ increases to a value that is greater than $T_p^{(m)}$. 
3.3 Alternatives in the Multiple-element Case

When the degree of RDF is $T_p$ with probability $p$, it is ideal to enumerate alternate read replica sets whose size is equal to that of the minimum read replica set achieving $T_p$ with probability $p$. Let $N_r$ and $N_m(T_p,p)$ be the number of replicas in a tree-topology network for update propagation and the size of the minimum read replica set that achieves the RDF of $T_p$ with probability $p$, respectively. To enumerate read replica sets whose size is $N_m(T_p,p)$, we have to calculate the sets of replicas covered by $N_r!/N_m(T_p,p)!(N_r-N_m(T_p,p))!$ combinations of replicas. However, when $T_p$ is not very small, the number of these combinations requires a high computational complexity.

The OVDC method determines a minimum read replica set for any degree of RDF. In addition, the size of the minimum read replica set gradually decreases as the degree of RDF increases when the degree of RDF is greater than a particular value, as described in Section 3.1. Hence, we can calculate alternate read replica sets with low computational complexity by using the above property, as follows:

1. When the degree of RDF is $T_p$, we use another degree of RDF, $T^*_p$, to calculate the minimum read replica set using the OVDC method, where $T^*_p$ is equal to or less than $T_p$. The minimum read replica set for $T^*_p$ can apparently be a read replica set for $T_p$ with probability $p$ or more.

2. We assign a set of replicas called an assigned range originator set to every read replica. For low computational complexity to determine alternate read replica sets, we determine assigned range originator sets so that replica $r$ must be included in the assigned range originator set of one of the read replicas that cover $r$.

3. For all read replicas $i$, we find alternative replicas whose range originator sets are equal to or are supersets of the assigned range originator set of $i$.

When $T_p-T^*_p$ is small, the size of alternate read replica sets determined by the above procedures is close to that of the minimum read replica set for $T_p$.

First, to evaluate how much $T_p-T^*_p$ is needed to obtain the large number of alternate read replica sets, we consider the condition where $T^*_p$ is equal to $T_p$. In an iteration of the algorithm used in the OVDC method, mandatory replicas, which are read replicas, tend to be located as far as possible from leaf replicas of a network because the range originator set of a leaf replica tends to be a subset of the range originator sets of replicas located closer to the center of a network. Then, replicas covered by mandatory replicas are removed from a network before the next iteration. In the next iteration, mandatory replicas also tend to be located as distant as possible from leaf replicas. Therefore, mandatory replicas determined in different iterations tend to be located as distant as possible from each other.

When a replica receives updates from its range originators, let $L_p^{(i)}$ be the maximum delay with probability $p$ through its adjacent replica $i$. If $L_p^{(k)}$ of replica $i$ is small for any adjacent replicas $k$ except for $j$ so that replica $j$ can cover replicas in the range originator set of $i$, adjacent replica $j$ of read replica $i$ becomes its alternative. However, because of the tendency of mandatory replicas to be located far from leaf replicas, as described just above, when $T^*_p$ is equal to $T_p$, it hardly occurs that $L_p^{(k)}$ is generally small for any read replica so that adjacent replicas of replica $i$ can be its alternative.

An example operation of the OVDC method is shown in Figs. 7 (a) and (b). We assume for simplicity that the update delay along every edge has the same probability density function. In the figures, the degree of RDF is the delay along one edge with probability $p$. The statuses of replicas in the first and second it-
erations are represented in Figs. 7(a) and (b), respectively. In the first iteration, replicas 6, 7, 10, 16, and 19 are selected as mandatory replicas. They are located as far as possible from leaf replicas. Replicas that are not covered by mandatory replicas are 1, 2, 4, and 14. In the subtree including these replicas, replicas 2 and 4 are selected as mandatory replicas in the second iteration. They are also located as far as possible from leaf replicas of the subtree. This means that replicas 2 and 4 are located as far as possible from the mandatory replicas selected in the first iteration. There are no alternatives for read replicas 6, 7, 10, 16, and 19, though there are alternatives 1 and 4 for read replicas 2 and 4, respectively, by making their assigned range originator sets \{1, 2\} and \{4, 14\}, respectively. However, the number of alternate read replica sets is 1 because not all read replicas have alternatives.

Next, we consider the condition where \(T_p^*\) is less than \(T_p\). When replica \(r\) is a read replica and the equivalent hop count of replica \(r\) from \(T_p^*\) to \(T_p\) for the range originator set of \(r\) is \(c\), the number of alternate read replicas of replica \(r\) is \(1 + \sum_{i=1}^{c} N_d(N_d - 1)^{i-1}\) for the same reason as that given in the rough analysis in Section 3.2, where \(N_d\) is the degree of nodes. When we determine the assigned range originator sets of read replicas, there are a number of combinations of replicas. If we determine the assigned range originator set of read replica \(i\) so that its size is as small as possible, replica \(i\) tends to have many alternatives. However, other read replicas might have a small number of alternatives because their assigned range originator sets have many elements. The number of alternate read replica sets depends on the minimum number of alternatives among read replicas. Therefore, we determine assigned range originator sets so that the difference in the number of read replica alternatives is small as follows. When replica \(i\) is covered by multiple read replicas, we make it an element of the assigned range originator set of the read replica that can receive updates from replica \(i\) in the shortest time. As in the single-element case, the number of alternate read replica sets rapidly increases as \(T_p - T_p^*\) increases in the multiple-element case. In addition, the computational complexity for the multiple-element case is at most \(O(N_r^2)\) because in procedures (2) and (3) to determine alternate read replica sets, the computational complexity of \(O(N_m(T_p, p)N_r)\) is needed to assign every replica to a read replica. Therefore, we can effectively determine a large number of alternate read replica sets by means of a small increase in the number of read replicas.

3.4 Example

An example of alternate read replica sets in the multiple-element case is shown in Fig. 8, where we assume for simplicity that the update delay along every edge has the same probability density function. In the figure, \(T_p^*\) and \(T_p\) are the delays along two and three edges with probability \(p\), respectively. Replica 3 has three alternatives, which are replicas 1, 7, and 10 because replicas covered by multiple mandatory replicas are assigned to the closest mandatory replica. The alternatives of replicas 5 and 15 are \{2, 6, 22, 23\} and \{14, 16, 19\}, respectively. As a result, there are four alternate read replica sets. Therefore, our load distribution method reduces the load of processing read transactions to 1/4 of that in the OVDC method by increasing the number of read replicas by 1 because the number of read replicas is two when the degree of RDF is \(T_p\).

4. Evaluation

We used randomly generated tree-topology networks to evaluate the proposed method. We used a Gamma function as the probability density function for the delay of update propagation on each direction of each link. This function is generally represented by

\[
(x - c)^{r-1} \exp(-\alpha(x - c)).
\]

There are two types of strategies to propagate updates among replicas: immediate and deferred. In the immediate strategy, every refresh transaction is immediately transmitted to replicas that have not received it. In the deferred strategy, transactions are aggregated into one request for a particular time period and then transmitted. We assumed that the deferred strategy should be used because it is suitable for high scalability. To evaluate the proposed method, we used two parameter sets of
a Gamma function for the deferred strategy: i) $c = 10 \, (s), \alpha = 2, \text{ and } r = 1$ and ii) $c = 20 \, (s), \alpha = 2, \text{ and } r = 2$. The parameter sets were randomly assigned to each direction of each link. The evaluation result described below is the average value for 50 different networks.

The numbers of alternate read replica sets when the average equivalent hop counts are 0.00, 1.33, and 2.67 to determine $T_p^*$ are shown in Fig. 9. For the probability of $p$ in the evaluation, we used a value of 0.95 as a sufficiently high probability that is frequently used in statistical tests. In this evaluation, the number of replicas in a tree-topology network and the maximum degree of nodes are 1,000 and 8, respectively. The horizontal axis is the RDF normalized by the maximum delay with a probability of 0.95. In the figure, the number of read replicas is also shown. The lower bounds of normalized RDF ranges where the number of read replicas is less than 10 are $3.70 \times 10^{-1}, 4.06 \times 10^{-1}, \text{ and } 4.04 \times 10^{-1}$ when the maximum degrees of nodes are 5, 8, and 12, respectively. When the proposed method is used, a greater maximum degree of nodes results in a greater number of alternate read replica sets in the multiple-element case, as shown by the rough analysis in Section 3.3.

The change in the number of alternate read replica sets depending on the number of replicas in a tree-topology network is shown in Fig. 11, where the maximum degree of nodes is 8. In the figure, the numbers of alternate read replica sets in tree-topology networks that have 100, 500, and 1,000 replicas are shown, where the average equivalent hop counts to determine $T_p^*$ are 0.00 and 2.67. In the multiple-element case, a greater number of replicas results in a greater number of alternate read replica sets. According to the rough analysis described in Section 3.3, the number of alternate read replica sets is independent of the number of replicas. However, the number of alternate read replica sets in a small-scale network is smaller than that in a large-scale network, as shown in this
Fig. 12 Change in number of alternate read replica sets depending on equivalent hop count in single-element case.

Fig. 13 Change in number of alternate read replica sets depending on equivalent hop count in multiple-element case.

simulation result. This is because a smaller number of replicas results in a shorter hop count distance from a read replica to leaf replicas. In other words, though the rough analysis described in Section 3.3 assumes that a read replica is not located close to leaf replicas, there are edges of a network close to read replicas in a small-scale network.

The ratio of the number of alternate read replica sets for $T_p$ to that for $T_p^{(m)}$ is shown in Fig. 12. The horizontal axis of the figure indicates the average equivalent hop count from $T_p^{(m)}$ to $T_p$. This ratio rapidly increases as the equivalent hop count increases, as shown by the rough analysis in Section 3.2.

The ratio of the number of alternate read replica sets to that where the method described in Section 3.3 is not used is shown in Fig. 13. As shown by the rough analysis in Section 3.3, the number of alternate read replica sets rapidly increases as the equivalent hop count increases in the figure though the ratio in a large-scale network is greater than that in a small-scale network. The reason for this is the same as that for the change in the number of alternate read replica sets depending on the number of replicas, which is described in the fourth paragraph of this section. The ratio in the figure is small in the single-element case because the number of alternate read replica sets is large unless the method described in Section 3.3 is used. In addition, the ratio in the figure is small when the normalized RDF is small because many read replicas are located close to leaf replicas in a network and have a small number of alternatives.

The above discussion shows that the proposed method provides a large number of alternate read replica sets in the single-element case. In addition, it increases the number of alternate read replica sets by adding a small number of read replicas in the multiple-element case. It thus achieves high scalability in terms of the load distribution of processing read transactions while guaranteeing the freshness of data provided to clients.

5. Related Work

Data replication methods are classified into eager and lazy replication types. Eager replication can be achieved by using ROWA, ROWA available, and quorum consensus, as described in Section 1. Eager replication provides up-to-date data to clients. In other words, clients can retrieve data with the necessary freshness. However, this type of replication requires a high overhead to process transactions. Lazy-group replication is suitable to improve scalability, but the freshness of data depends on replicas accessed by clients. Our method achieves both high scalability of transaction processing by using lazy-group replication and the provision of data with the freshness requested by clients.

In probabilistic quorum systems, a quorum intersects any of the other quorums with some probability to offer effective load reduction on servers and high availability. In probabilistic quorum systems, replicas in a quorum are simultaneously updated as in replica control methods using quorum consensus. In lazy-group replication, an update is first processed by one replica and then gradually propagated among replicas. When using quorum consensus, replicas included in a write quorum are simultaneously updated. From the viewpoint of sets of updated replicas, we can regard the set of replicas to which updates are propagated for a particular time period after they were issued.
as being an equivalent write quorum, though quorum consensus is a technique for one-copy serializability\textsuperscript{3}). In other words, a write quorum first consists of only one replica in lazy-group replication, and the number of its members is then increased as time elapses. When the RDF requested by a client is $T_p$, the data value read by a client has to reflect any updates that were issued before $t_c - T_p$, where $t_c$ is the present time. Therefore, for any replica, there is an equivalent write quorum whose members reflect updates initially processed by it before $t_c - T_p$. The changes in equivalent write quorums in lazy-group replication at times $t_1$, $t_2$, and $t_3$ are shown in Figs. 14 (a), (b), and (c), respectively, where $t_1$, $t_2$, and $t_3$ are the times at which updates are propagated along zero, two, and three edges, respectively. At time $t_1$, there are seven equivalent write quorums that consist of only one replica. At time $t_2$, there are three equivalent write quorums. At time $t_3$, there is only one equivalent write quorum because updates initiated by any of replicas can be propagated to all replicas. Our algorithm shown in Fig. 2 determines a read quorum that probabilistically intersects any equivalent write quorums, as in probabilistic quorum systems where statically defined quorums intersect each other with some probability. In our method, clients can obtain data with a freshness appropriate to their requirements by retrieving data from replicas in a read quorum determined according to their requests.

Recently, update propagation methods in replicated peer-to-peer systems consisting of unreliable nodes have been studied\textsuperscript{23),24)}. Such systems are usually composed of individually administrated hosts that dynamically join and leave systems. These methods can probabilistically provide effective and reliable update propagation when operational replicas satisfy particular conditions. In these methods, when certain conditions are satisfied, a node can access data reflecting updates after a particular time has elapsed since they were issued. In such systems, the freshness of data that a node can access depends on the conditions of a system. However, our method enables a client node to retrieve data with a variety of degrees of freshness according to its requirements.

6. Conclusion

This paper has proposed a scalable load distribution method for view divergence control of data freshness in replicated database systems. The proposed method reduces the load of processing read transactions by using the characteristics of the change in the size of the read replica set depending on data freshness. We showed that our method reduces the load of processing read transactions to about 1/40 of that in the OVDC method in the range where normalized RDF is more than about 0.4 in treeto topology networks where the number of replicas and the maximum degree of replica nodes are 1,000 and 8, respectively. As a result, the proposed method achieves extremely high scalability for a wide range of RDF values in a large-scale network.

References

1) Fischer, J.J. and Michael, A.: Sacrificing Serializability to Attain High Availability of Data in an Unreliable Network, Proc. 1st ACM Symposium on Principles of Database Systems, pp.70–75 (1982).
2) Bernstein, P.A. and Goodman, N.: An Algorithm for Concurrency Control and Recovery in Replicated Distributed Databases, ACM Transactions on Database Systems, Vol.9, No.4, pp.596–615 (1984).
3) Bernstein, P.A., Hadzilacos, V. and Goodman, N.: Concurrency Control and Recovery in Database Systems, Addison-Wesley (1987).
4) Gray, J., Helland, P., O’Neil, P. and Shasha, D.: The Dangers of Replication and a Solution, Proc. ACM SIGMOD International Conference on Management of Data, pp.173–182 (1996).
5) Helal, A., Heddaya, A. and Bhargava, B.: Replication Techniques in Distributed Systems, Kluwer Academic Publishers (1996).
6) Cox, P. and Noble, B.D.: Fast Reconciliations in Fluid Replication, Proc. International Conference on Distributed Computing Systems, pp.449–458 (2001).
7) Yamashita, T.: Dynamic Replica Control Based on Fairly Assigned Variation of Data with Weak Consistency for Loosely Coupled Distributed Systems, Proc. 22nd IEEE Inter-
national Conference on Distributed Computing Systems, pp.280–289 (2002).
8) Cuenca-Acuna, F.M., Martin, R.P. and Nguyen, T.D.: Autonomous Replication for High Availability in Unstructured P2P Systems, Proc. 22nd IEEE International Symposium on Reliable Distributed Systems, pp.99–108 (2003).
9) Gopalakrishnan, V., Silaghi, B., Bhattacharjee, B. and Keleher, P.: Adaptive Replication in Peer-to-Peer Systems, Proc. IEEE International Conference on Distributed Computing Systems, pp.360–369 (2004).
10) Barbara, D. and Garcia-Molina, H.: The Reliability of Voting Mechanisms, IEEE Trans. Comput., Vol.36, No.10, pp.1197–1208 (1987).
11) Garcia-Molina, H. and Barbara, D.: How to Assign Votes in a Distributed System, J. ACM, Vol.32, No.4, pp.841–860 (1985).
12) Gifford, D.K.: Weighted Voting for Replicated Data, Proc. 7th Symposium on Operating System Principles, pp.150–162 (1979).
13) Jajodia, S. and Mutchler, D.: Dynamic Voting, Proc. ACM SIGMOD Annual Conference, pp.227–238 (1987).
14) Yamashita, T. and Ono, S.: View Divergence Control of Replicated Data Using Update Delay Estimation, Proc. 18th IEEE Symposium on Reliable Distributed Systems, pp.102–111 (1999).
15) Transaction Processing Performance Council. http://www.tpc.org
16) Mills, D.L.: Precision Synchronization of Computer Network Clocks, Computer Communication Review, Vol.24, No.2, pp.28–43 (1994).
17) Shiba, S. and Watanabe, H.: Statistical Methods II: Estimation, Shinyosha (1976).
18) Rekhter, Y. and Li, T.: A Border Gateway Protocol 4 (BGP-4), RFC1771 (1995).
19) Moy, J.: OSPF Version 2, RFC2328 (1998).
20) Birman, K.P.: Building Secure and Reliable Network Applications, Manning Publications (1996).
21) Malkhi, D., Reiter, M. and Wright, R.: Probabilistic Quorum Systems, Proc. ACM Symposium on Principles of Distributed Computing, pp.267–273 (1997).
22) Malkhi, D., Reiter, M. and Wool, A.: Probabilistic Quorum Systems, Information and Computation, Vol.170, No.2, pp.184–206 (2001).
23) Datta, A., Hauswirth, M. and Aberer, K.: Updates in Highly Unreliable, Replicated Peer-to-Peer Systems, Proc. IEEE International Conference on Distributed Computing Systems, pp.76–88 (2003).
24) Wang, Z., Das, S.K., Kumar, M. and Shen, H.: Update Propagation through Replica Chain in Decentralized and Unstructured P2P systems, Proc. P2P ’04, pp.64–71 (2004).

(Received May 9, 2005)  
(Accepted November 1, 2005)  
(Released February 8, 2006)

Takao Yamashita received the B.S. and M.S. degrees in electronics engineering from Kyoto University in 1990 and 1992, respectively. In 1992, he joined Nippon Telegraph and Telephone Corporation. His current research interests encompass distributed algorithms and loosely-coupled distributed systems. He is a member of the IEEE, IEICE, and IPSJ.