マルチメディア通信と分散処理　94－7
分散システム/インターネット運用技術　14－7
（1999．7．15）

複製データのビュー最新度制御方法および スケーラビリティに関する評価

山下　高生、
yamasita@slab.ntt.co.jp,

小野　諭
ono@slab.ntt.co.jp

日本電信電話株式会社
〒180-8585　東京都武蔵野市緑町3-9-11

概要：本論分では、複製データが非同期に更新される場合において、クライアントから見たデータの最新度に注目し、クライアントの要求する最新度をもつデータを提供する方法について提案を行う。本方法では、まず、木構造状に論理的に結合された複製が木構造に沿って更新が伝搬していくとき、その遅延の統計的な推定値もとに計算した複数の複製ノードからデータおよびトランザションログを読み出す。次に、読み出しを行った複製ノードに反映されている更新をすべて反映したデータをクライアントに提供するという方法である。本方法の評価として、複製ノード数に関するスケーラビリティーの評価を行った。その結果、異なる総ノード数の複製ノードについて、すべてのノードに更新が反映される時間に対する同一の比率の最新度を得るために読み出しを行わなければならい複製ノードの数の増加が、総ノード数の増加と比較して少ないことを示した。

A view divergence control method of replicated data and the evaluation of its scale dependency

Takao Yamashita　and　Satoshi Ono
yamasita@slab.ntt.co.jp,　ono@slab.ntt.co.jp

Nippon Telegraph and Telephone Corporation
3-9-11 Midori-cho, Musashino-shi, Tokyo180-8585 Japan

Abstract: This paper proposes a method to control the view divergence of replicated data when copies of sites in a replicated database are asynchronously updated. The view divergence of replicated data is the difference in the lateness of the updates reflected in the data acquired by clients. Our method accesses multiple sites and provides data to a client that reflects all the updates received by the sites when replica sites are logically connected with a tree structure. The selection way of replica sites in replicated database uses the probability distribution of the update delays. The evaluation of our method shows that the number of accessed replicas does not increase much to reduce the ratio of veiw divergence as the total number of replicas increases.

1　Introduction

As computer networks grow, many distributed applications on widely spread computers in a network will share various data in data stores and in telecommunication, decision support, and information-retrieval systems. The shared data are operated using read and update transactions. Applications require high availability, scalability, and a short response time in processing these transactions.

Data replication is an effective technique for attaining these properties, especially for data that are mostly read. It locates copies of the same data objects in sites to process transactions from clients. This technique is classified into eager and lazy replication methods [1]. Eager replication achieves one-copy-serializability [2] using write-all or quorum consensus methods [2].

Lazy replication methods [3][4] can be still more divided into lazy master and lazy group replication. The former has a master replica of each object that can originate an update to the object and gives ACID serializability [1]. In the latter, all the replicas can originate updates. The latter can be used when transaction processing to achieve convergence property is possible [1]. An update by a commutative transaction is an example. The lazy replication technique gradually propagates updates

into sites as a refresh transaction after the updates have been processed at a site. In this situation, applications require freshness, because applications need as late data as possible [5]. Freshness is defined as the rate of updated replicas from among all the replicas [5]. From a client viewpoint, the insufficient freshness causes the view divergence of replicated data that means the divergence of the lateness of an update transaction reflected in the data acquired by clients. Especially in lazy group replication, since a client can not decide which replica has the latest copy due to update origination from multiple sites, the view divergence should be improved.

For example, we consider the location discovery of a mobile terminal in a network when the location information of a mobile terminal is managed in sites at multiple locations. When a computer communicates with a mobile terminal, the computer has to know where the mobile terminal is connected in the network. If we cannot obtain the correct location of a mobile terminal at the present time and the next location to which a mobile terminal moves are managed at each site for only a time period T to reduce the amount of data managed at each site, the replicated database has to provide the data within T to find a mobile terminal by tracing the migration path from the obtained location to the present location.

The smaller delay of update propagation leads to smaller view divergence of replicated data. However, the increase in the number of sites needed to improve availability and reduce the response time worsens the divergence because the refresh transactions are transmitted little by little into sites to reduce the load for propagating updates from a site to a number of replicas. In addition, delay of an update propagation in a replicated database varies depending on the load of each site. Therefore, it is difficult for a replicated database to reliably continue to provide data all the time so that the data meets all the requirements of different applications.

In this paper, we propose a method to control the view divergence of replicated data that are asynchronously updated using lazy group replication. Our method accesses multiple sites and provides data to a client that reflects all the updates received by the sites. This method can reliably and adaptively control the view divergence based on update delay estimation in an environment where the delay of update propagation varies. To reduce the overhead of read transactions, our method determines the minimum number of sites in the condition where an update is propagated into sites connected using a tree structure. Finally, we evaluate the range of the view divergence we can feasibly control by means of simulations.

The remainder of this paper is organized as follows. Section 2 describes a method for controlling the view divergence of replicated data. In Section 3, we discuss the algorithm used to select the minimum number of sites accessed by a client to obtain the data with the required RDF. In Section 4, we evaluate the range of the view divergence feasibly controlled using our method by means of simulations. We conclude the paper in Section 5 by discussing remaining work.

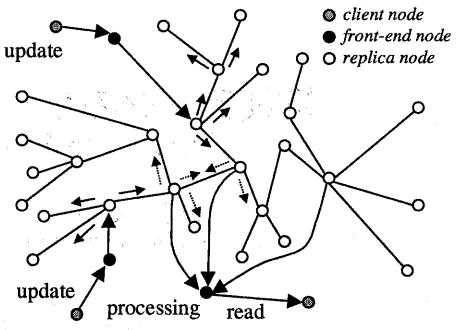

Figure 1: System architecture used in our method of controlling the view divergence of replicated data

2 View divergence control of replicated data

In our method, there are three types of node: client, front-end, and replica nodes. The transactions requested by clients are read and update. The update transaction requested by a client is sent to a front-end node. The front-end node that receives the update transaction sends it to a replica node. Then, the update is propagated into all the replica nodes as a refresh transaction. A read transaction requested by a client is also sent to a front-end node. A front-end node sends a read transaction to multiple replica nodes. After a front-end node receives the data and, if necessary, recent logs of updates received by each replica as a reply message to the transaction, the front-end node calculates the data that reflects all the updates received by the accessed replicas using this information and the timestamp associated with the data. In this calculation, a front-end node processes updates as a replica node does. Our method controls the view divergence by using this calculation process, which depends on the applications or the definition of consistency. For example, in the case of location discovery of a mobile terminal, this calculation is very simple. In this application, a replica node process updates by replacing old data with a newer update based on the timestamp associated with the data and update. Therefore, the calculation of data that reflects all the updates received by the accessed replicas is done by selecting the latest data among obtained data from replicas based on their timestamp.

There are a number of combinations of replicas that meet the view divergence required by a client. Since the response time increases as the number of replicas a front-end node has to access increases, the number of replicas should be the minimum. In addition, since there are $2^n - 1$ combinations of replicas, where n is the number of replicas, an effective algorithm to select the minimum number of replicas is necessary. An effective algorithm is described in the next section. In this algorithm, we use the probabilistic lateness of updates reflected in acquired data, called read data freshness (RDF). The degree of RDF represents a client's requirement about the view divergence. The formal definition of RDF T_p 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. In other words, any updates invoked between t_l and t will not come to any replicas in the future with probability p. We represent data with the degree of RDF T_p as data within T_p.

3 Method of selecting read replicas

3.1 System architecture

To select replicas so that the acquired data by a client meets the degree of RDF, we have to know how update propagation delay occurs. It depends on the topology of update propagation paths. Figure 1 shows the system architecture used in our method. Replicas are connected by logical links. The topology of the graph comprised of the logical links and the replica nodes is a tree. An update sent by a front-end node propagates in the tree through the links as a refresh transaction. When a replica node receives an update, the node forwards it to all the adjacent nodes except the replica node from which the updates came. The update delay is caused by communication delay, protocol overhead to record logs to fulfill recovery [5], time for aggregating some updates in one refresh transaction [6], and so on.

3.2 Assumptions

Our algorithm to select read replicas assumes the following.

1. Clocks of nodes are synchronized.
2. Delay time of update propagation can be statistically estimated.
3. If T_r is the response time for a client to obtain data, then $T_r > T_p$, where T_p is the degree of RDF required by the client.

Condition 1 is where the clocks on a client, a front-end, and replica nodes are synchronized with each other. This is necessary for sampling the delay time of message delivery of a read, an update, or a refresh transaction. Clock synchronization can be achieved using various methods [7][8].

Condition 2 means that we can estimate the upper confidence limit of delay between replicas with some probability by using samples of the measured delay time. This means that the probability distribution of the delay does not change much in the time it takes to obtain enough samples to estimate the delay. The upper confidence limit can be calculated by various statistical estimation methods [9].

For obtaining data, we need the time for communication between the client and a front-end node and communication between the front-end node and replica nodes. Therefore, when a client requires data with the degree of RDF T_p, a client can request only data that satisfies condition 3. When a replica transmits updates to another replica, some processes are necessary. For example, when an update is propagated, each replica needs to keep update logs so that if it fails, any updates are recovered and pending updates are reliably sent to other replicas. Since the delays caused by these processes occur at each replica, we consider this condition to be acceptable.

3.3 Algorithm

3.3.1 Terminology

To explain our algorithm, we define four terms, which are read replica (set), range originator, classified replica, and mandatory replica. A *read replica* is a replica node to which a front-end node sends a read transaction to obtain the value of a data object. We denote a set of read replicas to obtain the data with the degree of RDF required by a client 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 required 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 a set of range originators of replica i. Then, we say that replica j is *covered* and *uncovered* by replica i if $j \in O_i$ and $j \notin O_i$, respectively. A *classified replica* is a replica whose set of range originators is not a subset of a set of range originators of any other replicas. A *mandatory replica* is a classified replica that has one or more range originators which are not included in any other classified replicas' range originators.

3.3.2 Calculation of a minimum read replica set

In this section, we describe an algorithm to select a minimum read replica set when an update is propagated into replicas connected by a tree. When the delay time between replicas is given by the probability distribution and the upper confidence limit, the problem of finding a read replica set has some different properties from the problems of weighted graphs in graph theory. For example, when there is a replica j on the path from replica i to k, the distance from replica i to k is the sum of the distances from i to j and from j to k in the weighted graph. However, let d_{xy} be the upper confidence limit of the delay time from replica x to y with probability p. Then d_{ik} is not the sum of d_{ij} and d_{jk}. For another example, when there are replicas k on the paths from replica i to l and from j to l and when $d_{ik} \leq d_{jk}$, it is not always true that $d_{il} \leq d_{jl}$. To verify Lemmas 1 and 3, we use two properties about 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 all replica j that are on the path from replica i to k. The second property is that if $d_{ij} > T_p$, then $d_{ik} > T_p$ for all replicas k that satisfy the condition that replica j is on the path from replica i to k.

Our algorithm incrementally constructs a minimum read replica set R comprised of only classified replicas.

Theorem 1 *There is at least one read replica set consisting of only classified replicas and this set is of a minimum number of read replicas among all the read replica sets.*

Proof: Let R_o be one of the read replica sets that are of a minimum number of read replicas. Let O_i be a set of range originators of replica i. Then, for all read replicas $i \in R_o$ that are not classified replicas, there is at least one classified replica j that satisfies $O_i \subset O_j$. Let R_m be a read replica set constructed by replacing replica $i \in R_o$ with j. Thus, R_m is of a minimum number of read replicas and R_m consists of only classified replicas. □

$A := \{i \mid i \text{ is a replica node}\}$;
$R := \{\emptyset\}, V := \{\emptyset\}$;
$S := A$;
while $(\forall i \in A \setminus V : O_i \neq A \setminus V)$; ··· (1)
  for all $i \in S$, $O_i := O_i \setminus V$; ··· (2)
  $C := \{i \mid i \in S \wedge \forall j \in S : O_i \not\subset O_j\}$; ··· (3)
  for all $i, j \in C$ $(i \neq j)$:
    if $O_i = O_j$ then $C := C \setminus \{j\}$; ··· (4)
  $M := \{i \mid \exists j \in A \setminus V : \forall k \in C\ (k \neq i) :$
    $j \in O_i \wedge j \notin O_k\}$; ··· (5)
  $R := R \cup M$; ··· (6)
  $V := V \cup \{i \mid i \in O_j \wedge j \in M\}$; ··· (7)
  $T_s := constructMinimumTree(A \setminus V)$; ··· (8)
  $S :=$ a set of replicas in T_s; ··· (9)
$R := R \cup \{i\}$, where $O_i = A \setminus V$; ··· (10)

Figure 2: Algorithm to calculate a minimum read replica set.

Figure 2 shows our algorithm to calculate a minimum read replica set. T_o and T_s are the original tree for update propagation and a subtree of T_o constructed in 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.

In step (2), we remove replicas covered by replicas in R from O_i. In step (3), we calculate the classified replicas for T_s, which is initially T_o. 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, since a mandatory replica has at least one replica that is not included in the range originators of any classified replicas. In step (8), we construct the minimum subtree T_s including all the uncovered replicas, assuming the weight of all the edges is 1. The procedure from step (2) to (9) is iterated unless there is at least one replica that covers all the replicas uncovered by any replica $i \in R$ in T_s. Step (10) adds one of the replicas that covers all the replicas uncovered by any replica $i \in R$ in T_s to R and then the algorithm terminates.

Lemma 1 *Let i be a replica that is in T_o but is not in T_s. Then, there is at least one replica j in T_s that satisfies $O_i^{(m)} \subseteq O_j^{(m)}$, where $O_i^{(m)}$ is a set of range originators of replica i in the m^{th} iteration.*

Proof: Let i' be the nearest replica node among replicas in T_s from replica i, assuming the weight of all the edges is 1. Then, $O_i^{(m)} \subseteq O_{i'}^{(m)}$ because of the first property described at the beginning of this subsection. □

Lemma 2 *Let $O_i^{(m)}$ be a set of range originators of replica i in the m^{th} iteration. If there is replica i that is not a classified replica in T_o but a classified replica in T_s of the m^{th} iteration, then there is a classified replica j in T_o that satisfies $O_j^{(m)} = O_i^{(m)}$, where $O_j^{(m)} = O_j^{(1)} \setminus V$.*

Proof: From the assumption, there is replica k that is not in T_s, is in T_o, and satisfies $O_i^{(1)} \subset O_k^{(1)}$. Since $O_i^{(m)} = O_i^{(1)} \setminus V$, $O_i^{(m)} \subseteq O_k^{(m)}$. From Lemma 1, there is replica k' in T_s that satisfies $O_k^{(m)} \subseteq O_{k'}^{(m)}$. Because $O_i^{(m)} \subseteq O_j^{(m)}$ for all replica j on the path from k to i, $O_i^{(m)} \subseteq O_{k'}^{(m)}$. Since replica i is a classified replica in T_s, $O_i^{(m)} \not\subset O_{k'}^{(m)}$. So, $O_{k'}^{(m)} = O_i^{(m)}$ because $O_i^{(m)} \subseteq O_{k'}^{(m)}$ and $O_i^{(m)} \not\subset O_{k'}^{(m)}$. Hence, since $O_{k'}^{(m)} = O_i^{(m)}$, $O_k^{(m)} \subseteq O_{k'}^{(m)}$, and $O_i^{(m)} \subseteq O_k^{(m)}$, then $O_k^{(m)} = O_i^{(m)}$. □

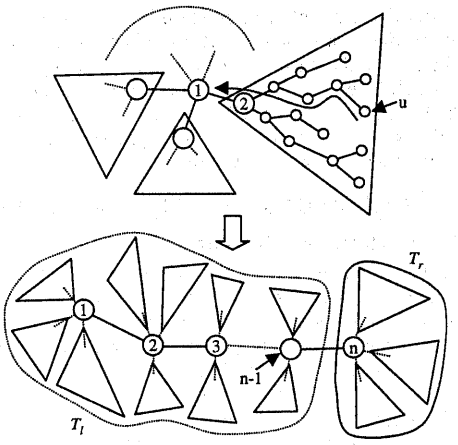

Figure 3: There is at least a mandatory replica in each iteration, when our algorithm does not terminate.

Lemma 3 *This algorithm terminates after a finite number of iterations.*

Proof: We consider the subtree T_s in the m^{th} iteration. Let $O_i^{(m)}$ be a set of range originators of replica i in the m^{th} iteration. For any replica i, when we suppose that i is the root of T_s, there is at least one leaf node from which an update cannot reach i within T_p with probability p when our algorithm does not terminate. All the leaf nodes are not covered by any $i \in R$ in each iteration. First, as shown in Figure 3, we label replica i "1". Next, we label a child of replica i "2" if there is at least one leaf node among the child and its descendants from which an update cannot reach i within T_p with probability p. Third, we assume the replica with "2" is the root of T_s and label an unlabeled child of the replica with "2" "3" if there is at least one leaf node among the child and its descendants from which the update transaction cannot reach the replica with "2" within T_p with probability p. We repeat this process until we find the replica with "n" that satisfies the condition that an update originated by unlabeled children of the replica with "n" and their descendants can reach the replica with "n" within T_p with probability p. Let T_r be a tree consisting of the replica with "n", its unlabeled children and their descendants, assuming that the replica with "n" is the root of T_s, as shown in Figure 3. Let T_l be a tree consisting of all the replica nodes that are included in T_s and not included in T_r. Then, all the replicas in T_r are range originators of the replica with "n" and there is no replica in T_l that covers all the replicas in T_r. Hence, there is no replicas i in T_l that satisfies $O_n^{(m)} \subset O_i^{(m)}$, where $O_n^{(m)}$ is the set of range originators of the replica with "n" in the m^{th} iteration. In addition, there is no replica in T_l that is uncovered by the replica

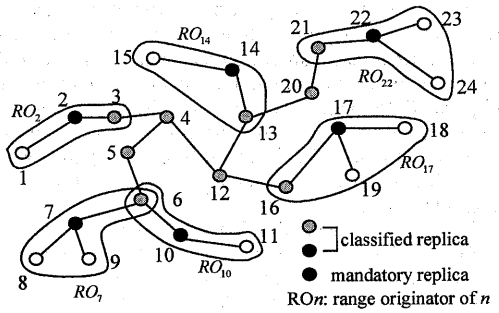

Figure 4: The tree for update propagation, and classified and mandatory replicas in the first iteration of our algorithm.

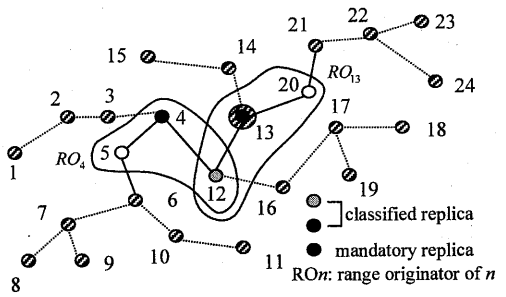

Figure 5: The classified, mandatory replicas, and range originators in the second iteration of our algorithm.

with "n" and that is covered by a replica covering all the replicas in T_r because of the second property described at the beginning of this subsection. Hence, there is no replica i in T_r that satisfies $O_n^{(m)} \subset O_i^{(m)}$. Therefore, one of the replicas with "n" or whose range originators is the same as the replica with "n" is a mandatory replica for replica u, where u is a replica in T_r and u is uncovered by the replica with "n-1". Since every iteration can find at least one mandatory replica, this algorithm terminates after a finite number of iterations. □

Theorem 2 *Our algorithm calculates a minimum read replica set.*

Proof: From Lemma 2, the addition of mandatory replicas in T_s to R in each iteration is equal to the addition of classified replicas in T_o. So, this algorithm adds only replicas that must be necessary in each iteration to construct a read replica set consisting of only classified replicas in T_o to R and adds only one replica after the iterations. Hence, our algorithm calculates a minimum read replica set from the principle of optimality and terminates after a finite number of iterations from Lemma 3. □

In steps (3), (4), and (5), $n(n-1)/2$ comparisons of O_i are necessary at most, where n is the number of replicas. In steps (2) and (7) at most n processes are necessary. Step (8) needs m processes, where m is the number of edges in T_s at most. The total number of iterations is at most n. Hence, the complexity of the algorithm is $O(n^3)$ at most.

3.4 Example

In this subsection, we demonstrate our algorithm using an example. Figure 4 shows the tree T_o for update propagation in the example. To simplify the example, we assume that delay time along one edge with probability p is equal to or less than the degree of RDF T_p required by a client, but that delay time along the path consisting of more than one edge with probability p is more than T_p.

In the first iteration, our algorithm calculates the classified replicas in the tree. The black and grey circles in the figure stand for classified replicas. Replica 2 is the mandatory replica of 1, replica 7 that of 8 and 9, 10 that of 11, 14 that of 15, 17 that of 18 and 19, and 22 that of 23 and 24. In this iteration, $R = \{2, 7, 10, 14, 17, 22\}$ and the uncovered replicas are 4, 5, 12, and 20. If we remove replica nodes covered by replicas in R from the tree, the two connected components remain: the trees are comprised of sets of nodes $\{5, 4, 12\}$ and $\{17\}$.

In the second iteration, we construct the minimum subtree of T_o T_s as shown in Figure 5. The hatched circles are covered by replicas in R. The tree including 4, 5, 12, and 20 is the tree comprised of 4, 5, 12, 13, and 20. In this iteration, we calculated the classified replicas for T_s again. The black and grey circles in the figure are the classified replicas. Replicas 4 and 13 are mandatory replicas for 5 and 20, respectively. Therefore, $R = \{2, 4, 7, 13, 14, 22\}$. Since all the replicas are covered by replicas in R after this iteration, our algorithm terminates.

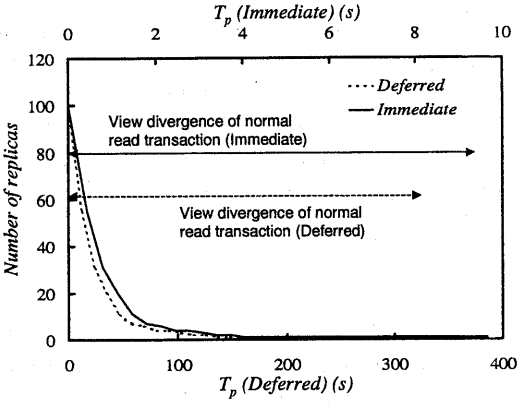

Figure 6: The relationship between the degree of the RDF and the number of read replicas to provide data with the RDF in the immediate and deferred propagation situations.

4 Evaluation

In this section, to evaluate the feasible range of the view divergence achieved by our method, we evaluate the relationship between the degree of RDF and the number of read replicas calculated by our algorithm. This relationship depends on the topology of update propagation and the delay time between replicas. We use a randomly generated tree as the topology. The maximum degree of a node in the tree is 5 and the total number of nodes is 100. The relationship also depends on the probability density

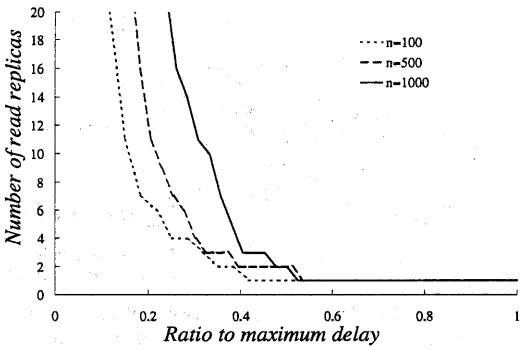

Figure 7: The relationship between the ratio of the RDF to the maximum delay and the number of read replicas.

function of the delay time for update propagation. The delay function of update propagation in a real network is very complicated. However, since the objective of this evaluation is a rough estimation of the relationship between the degree of RDF and the number of read replicas calculated by our algorithm, we use a Gamma function as the probability density function for the delay time on each direction of each link. This function is generally represented by

$$(x-c)^{r-1}\exp(-\alpha(x-c)). \quad (1)$$

We evaluate two delay time situations. One is the situation where every refresh transaction is immediately transmitted to other replicas that have not received it. The other situation is where some update transactions are aggregated in one request and transmitted [6]. We call the first situation immediate propagation and the second situation deferred propagation.

We used two probability density functions, each with different Gamma function parameters in each situation, and randomly assigned one of the two functions to each direction of each link. The parameters in the immediate propagation situation are $c = 200$ (ms), $\alpha = 20$, $r = 1$, and $c = 400$ (ms), $\alpha = 20$, $r = 2$. The parameters in the deferred propagation situation are $c = 10$ (s), $\alpha = 2$, $r = 1$, and $c = 20$ (s), $\alpha = 2$, $r = 2$.

The maximum delay in the tree is the delay from replica 94 to 98. The delay time with probability p is equal to or less than 9.35 s and 325.5 s in the immediate and deferred propagation situations, respectively. Therefore, the view divergence of the normal read transaction, which obtains data from a local replica of a client, is 9.35 s and 325.5 s in the immediate and deferred propagation situations, respectively.

Figure 6 shows the relationship between the minimum number of elements of a read replica set calculated using our algorithm and the degree of RDF. Our method can control the range of RDF from 4.029 to 9.35 s and from 136.3 to 325.5 s by acquiring data from 1 replicas in the immediate and deferred propagation situations, respectively. In addition the range of RDF by acquiring data from within 4 replicas in the immediate and deferred propagation situations are the range from 2.433 to 9.35 s and from 80.33 to 325.5 s, respectively.

From this results, when a normal read transaction is used in an ordinary environment and the view divergence of the normal read transaction meets the client's requirement, we found that our method can provide data with the required RDF even if the delay time becomes 4 times as long as in the ordinary environment. In other words, our method can reduce the degree of RDF of acquired data to about 1/4 that of the normal read transaction by accessing a small number of replicas.

Figure 7 represents the dependence on the scale of a replicated database system in our method. This figure shows the relationship between the ratio of the RDF to the maximum delay and the number of read replicas when the total number of replicas in randomly generated trees are 100, 500, and 1000. From this figure, as the number of replicas increases, the ratio of the RDF to the maximum delay increases. However, in order to achieve the same ratio, the increase in the number of read replicas is lower than that of the number of replicas.

5 Conclusion

We have proposed a method to control the view divergence of lateness of transaction reflected in data obtained by clients from a replicated database. We have evaluated the range of the view divergence achieved by our method. The evaluation is done by means of simulations in which 100 replicas exist. When the view divergence by the normal read transaction is 9.35 or 325.5 s with probability 0.95, our method can control the divergence in the range from 2.433 to 9.35 or from 80.33 to 325.5 s by reading from within four replicas.

References

[1] J. Gray, P. Helland, P. O'Neil, and D. Shasha, "The dangers of replication and a solution," in *Proc. ACM SIMOD International Conference on Management of Data*, pp. 173–182, 1996.

[2] P. A. Bernstein, V. Hadzilacos, and N. Goodman, *Concurrency Control and Recovery in Database Systems*. Addison-Wesley, 1987.

[3] J. J. Fischer and A. Michael, "Sacrificing serializability to attain high availability of data in an unreliable network," in *Proc. 1st ACM Symposium on Principles of Datbase Systems*, pp. 70–75, May 1982.

[4] R. Ladin, B. Liskov, and S. Ghemawat, "Providing high availability using lazy replication," *ACM Transactions on Computer Systems*, vol. 10, pp. 360–391, November 1992.

[5] E. Pacitti, E. Simon, and R. Melo, "Improving data freshness in lazy master schemes," in *Proc. IEEE 18th International Conference on Distributed Computing Systems*, pp. 164–171, May 1998.

[6] K. P. Birman, *BUILDING SECURE AND RELIABLE NETWORK APPLICATIONS*. Manning Publications, 1996.

[7] D. L. Mills, "Precision synchronization of computer network clocks," *Computer Communication Review*, vol. 24, pp. 28–43, Apr. 1994.

[8] T. Yamashita and S. Ono, "A statistical method for time synchronization of computer clocks with precisely frequency-synchronized oscillators," in *Proc. IEEE 18th International Conference on Distributed Computing Systems*, pp. 32–39, 1998.

[9] S. Shiba and H. Watanabe, *Statistical Method II Estimation*. Shinyosha, 1976.