Time-related replication for p2p storage system

Kyungbaek Kim

University of California, Irvine
Computer Science-Systems
3204 Donald Bren Hall, Irvine, CA 92697-3435
E-mail: kyungbak@uci.edu or kyungbaekkim@gmail.com

Abstract

Many p2p based wide-area storage systems are appeared to provide scalable storage services with idle resource from many unreliable clients. To minimize the data loss, quick replication is important to replace lost redundancy on other nodes in reaction to failures. The popular approach is the availability based replication which uses the individual node availability. However, with the high churn some replicas leave or fail within similar time. As a result, it requires bursty data traffic and sometimes it loses data.

This paper explores the time-related replication which uses the information of the session time to prevent the bursty failures. It tries to get the primary replica which has enough time to replace the lost redundancy. Moreover, the sparse replicas spread on the timeline and the number of overlapped replicas lessens as best it can. Results from a simulation study suggest that the time-related replication keep the high data availability with less data copy traffic.

1 Introduction

Availability is a storage system property which measures the probability that data can be retrieved. Many wide-area storage systems which are composed of many participants replicate data to others for keeping high availability. Each participant monitors the availability of data and replaces lost redundancy on others in reaction to failures [2][5][6][7]. In these days, P2P becomes the very popular method in storage intensive applications. However, P2P systems suffer from high node churn and the replication in these systems should consider the efficiency which means minimizing the data traffic overhead. To do this, we are considered with the following two problems: When do nodes need replicas and which nodes are selected to store replicas?

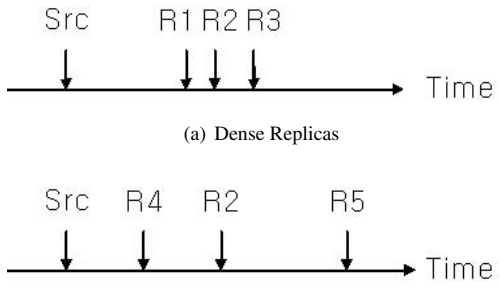


Figure 1. Timing effect of replicas. Sparse replicas are more efficient than dense replicas

When a node detects any change of its replications, it checks the availability of the data for which it is responsible. The popular approach in recent studies [1][8][9] for calculating the data availability is the following model: $A_d = 1 - (1 - A_1)(1 - A_2)..(1 - A_n)$ where n is the number of replicas and $A_1, A_2..A_n$ are each node availabilities of replicas. When the calculated data availability is below the target value, a node tries to make new replicas. At first time p2p emerged, the availability of each node is considered a constant value which represents the homogeneous characteristic [3][4] and a node select new replicas randomly. However, after p2p characteristics are revealed [13], the recent studies [1][9] pose variable node availability to data availability model and a node selects more available nodes as new replicas. According to this, p2p storage system can adhere to the high data availability reasonably with less data copy traffic than before.

This approach assumes that the each replica is independent to each other. That is, in RAID or other server based

978-0-7695-3106-9/08 $25.00 © 2008 IEEE
DOI 10.1109/ICN.2008.27

wide-area storage systems, the probability of failure is very low and when a node fails, there is enough time to replaces lost redundancy. However, in p2p system, most nodes has the high probability of failure and it is hard to assume that the probability of failure is independent. In figure 1, Src means the node which is responsible for the data and R1, R2, R3, R4 and R5 are the replicas for the data. Each arrow represents the leave time for each node. The dense replicas like figure 1(a) leave or fail within the similar time and the needed data copy traffic becomes bursty. That is, each replica is not independent and some replicas are meaningless. Sometimes there is not enough time to copy data for lost redundancy. If we spread the replicas more in order to reduce this bursty leave or fail, these sparse replicas like figure 1(b) can reduce the data copy traffic and increase the data availability.

In this paper, we suggest the time-related replication which uses the information of replicas' session time to achieve high data availability with low data copy traffic. The time-related replication is a kind of the availability based replication and uses the described data availability model for checking whether it needs more replicas or not. The major concern of the time-related replication is which nodes are selected for sparse replicas. At first, it tries to get the primary replica which has enough time to replace the lost redundancy. Moreover, it spreads the replicas on the timeline and tries to prevent overlapping replicas with each other. This behavior can diminish the number of meaningless replicas and reduces the bursty leave or fail. As a result, the time-related replication can reduce the needless data copy traffic and can also increase the data availability without any relation of both of the target availability and the node dynamicity. If we know the session time of all nodes exactly, we could get best performance of the time-related replication. Instead of the exact session time, we measure the MTTF (Mean Time To Failure) of each node and broadcast it to neighbor nodes.

We evaluate the effect of the time-related replication by using an event driven simulation. We compare the data traffic for various target availability and various node dynamics. The results represent that the time-related replication reduces the data copy traffic and achieves high data availability.

This paper is organized as follow. Section 2 introduces the detail of the time-related replication. The simulation environment and the performance evaluation are given in section 3. In section 4, we describe related works of the replication for the p2p storage system. Finally, we conclude in section 5.

2 Time-Related Replication

Time-related replication concentrates on how to select new replicas more than when nodes need new replicas. When a node is going to leave or finds failures of other nodes, it checks the data availability for which it is responsible by using the described model in section 1. If it needs new replica to keep the high availability, it should copy data from any available node which also has the same data and updates the replication information. We call the time for this replication procedure FRT (Fault Recovery Time). If the last node which is involved the replication procedure fails before it is done, we loose the data. To prevent this data loss, when a node needs new replicas, the priority is selecting the most available node among nodes whose EETs (Estimated End Time) are longer than α * FRT from BT (Base Time) and we call this selected replica the primary replica. The EET is the sum of the join time and the MTTF (Mean Time To Failure). BT is the base time of the node which needs a new replica. Basically BT is the node's EET, but after the current time passes this BT, BT sets to the current time. Because of this timing condition (EET $>$ BT + α * FRT), we can consider that the primary replica is more available than its natural availability. Whenever it satisfies the timing condition, it has additional availability value as its node availability and if it can not keep the condition anymore, it acts as a normal replica.

If a node still needs more replicas after selecting a primary replica, it tries to find the nodes whose EETs do not overlap with the other replicas' EETs. The time difference between EET of each replica and a new replica should be longer than β * FRT. If the time difference is smaller than β * FRT, it checks whether the node availability of a new node is bigger than the replica's node availability or not. If it is, a new node is selected as a new replica. Otherwise, this node is added to the pending list. This approach helps making sparse replicas and reducing data copy traffic. Moreover, if a node's EET is shorter than BT, it is also added to the pending list. This kind of a node will leave or fail very soon without any relation to the node availability. When a new node is selected as a new replica, its availability affects the data availability by using the previous model. After selecting the new replicas from the possible candidate nodes, if the data availability is still lower than the target availability, it selects new replicas from the pending list.

The figure 2 shows the example of the time-related replication. BT is the base time of a base node which needs new replicas. R1 and R2 are replicas for the data for which the base node is responsible and P3, P4 and P5 are monitored nodes which are candidate nodes for replicas. Each arrow except BT means each node's EET. In this case, R2 is a primary replica because R2's EET is longer α * FRT from BT. If the base node needs a new replica, it selects P4. P5's EET

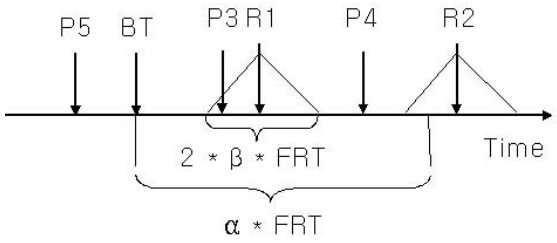


Figure 2. Example of time-related replication

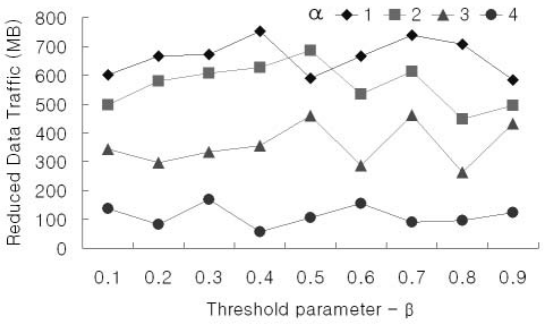
is shorter than BT and P5 is added to the pending list. P3 can be also added to the pending list because its EET overlaps with R1's EET. If the base node needs more replicas, it selects the most available node among the pending list.

This time-related replication uses the information of the session time. Each node could use more precise information if each node knows and broadcasts its end time honestly. A node can estimate its end time based on the current network status right after starting its p2p application, but this end time can be changed by the change of network status, p2p transaction policies or the dynamic p2p membership. That is, in p2p system which is composed of very diverse nodes, it is hard to estimate the correct end time. So we only get MTTF from each node and the time-related replication uses this information. To get MTTF, each node remembers both of the last join time and the last leave time and computes MTTF right after joining the p2p system. This calculated MTTF is distributed to the neighbor or monitored nodes for the replication by piggybacking it to periodic keep-alive messages. Each node stores this MTTF information at the MTTF cache to prevent some selfish nodes from modifying MTTF.

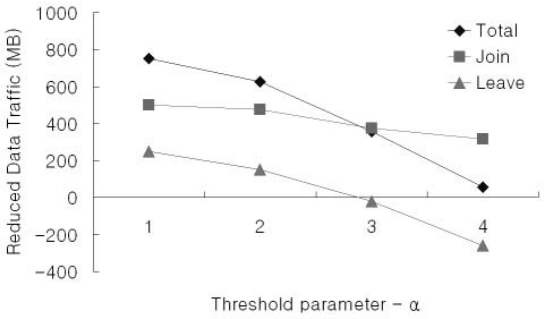
3 Evaluation

3.1 Simulation Setup

We make our p2p simulator which emulates behavior of nodes on the application layer. We implement a DHT based p2p algorithm, Pastry and apply simple replications and our time-related replication. The 160 bit ID space and 2000 nodes are used to organize a p2p system. Each node manages averagely 1024 objects whose average size is 20KB. The number of replicas is variable from 8 to 14 and the target availability is variable from 0.996 to 0.99994 which are decided by the number of replicas with 0.5 node availability. The Poisson distribution is used to distribute the dynamic characteristics of nodes and the exponential distribution is used to assign on/off duration of a node. According to this distribution, the lifetime of 80% of total nodes is below 60%



(a) α, β



(b) $\alpha, \beta = 0.4$

Figure 3. Preliminary Result, target availability = 0.999

of total simulation time, that is, only 20% of total nodes have the reliable server-like profile. Recent research [13] measures the life distribution of the p2p nodes and shows the similar distribution to this, and we can tell that this distribution is similar to the real world.

3.2 Preliminary Results

Before getting result, we should choose the values for α and β. α is used for selecting primary replica and β is used for making sparse replicas. The figure 3 shows the preliminary results with various α and β. The target availability is 0.999. In this figure, "Reduced Data Traffic" means how much data traffic is reduced by the time-related replication. In figure 3(a), α is various from 1 to 4 and β is various from 0.1 to 0.9. When α increases, the reduced data traffic decreases. When α is big, a node struggles to find a primary replica without any relation with node availability. If its availability is very low, a node needs more replicas and more data copies occur. On the other side, there is no clear pattern for β but we can find that the p2p system generally

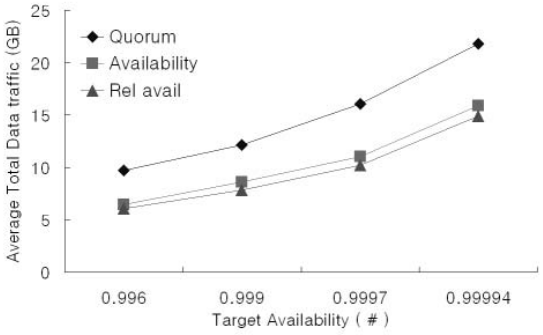
gets best performance when β is 0.4 or 0.5 and α is 1 or 2.

Figure 3(b) shows the reduced data traffic with various α and β = 0.4. In this figure, to find out the detailed effect of the time-related replication, we separate the join data traffic from the leave data traffic. When a node joins, the affected nodes update their replicas and we call the needed traffic the join data traffic. When a node leaves, the needed data traffic is called the leave data traffic. More detailed descriptions are on paper [1]. Regardless of α, the time-related replication reduces more join data traffic than leave data traffic. The careful selection of replicas for leave operations helps reducing the join data traffic. Moreover, when α increases, the leave data traffic decreases remarkably and this is the main reason of the degradation of the total reduced traffic. That is, to take more advantage of the time-related replication, the recovery time for a single failure should be short.

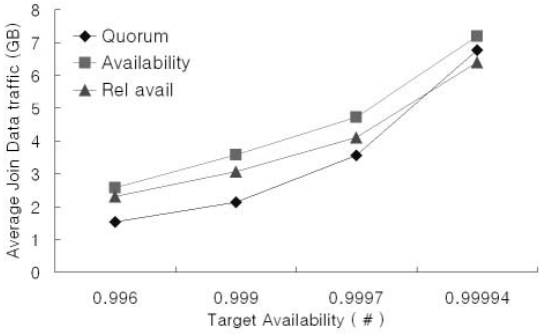
3.3 Data Copy Traffic

Figure 4 represents the comparison of average data traffic per a node with various target availability. α is 1 and β is 0.4. "Quorum" means the old and simple replication which uses the number of replication as a threshold value and "Availability" means the popular availability based replication. "Rel avail" means the time-related replication which we propose. As well known, the simple replication needs more data copy traffic than the availability based replication. According to the figure 4(b) and 4(c), though the simple replication can save more join data traffic with many replicas which are selected during leave operations, this strict failure recovery wastes too much data traffic.

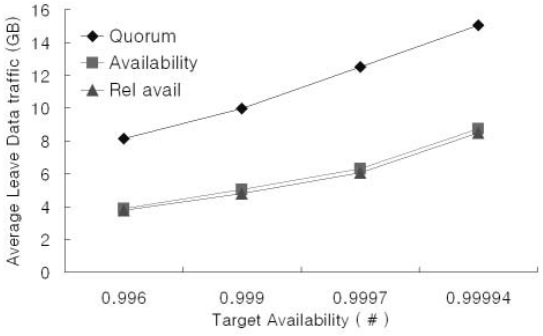
The time-related replication reduces more about 8% total data traffic than the availability based replication. The dominant partition of this reduction is the join data traffic. In figure 4(c), the time-related replication saves only about 3% leave data traffic, but figure 4(b) represents that the time-related replication reduces more about 13% join data traffic. The time-related replication is a kind of the availability based replication and uses the similar method for calculating data availability. Both of the availability based replication and the time-related replication need new replicas within similar timing and have similar leave data traffic. However, the time-related replication selects more independent nodes as new replicas and the number of useless replicas decreases. This more careful selection helps reducing join data traffic.



(a) Total



(b) Join



(c) Leave

Figure 4. Comparison of Data traffic with various target availability

3.4 Data Availability

To find out how the time-related replication affects the data availability, we check the data availability during the simulation. When a node will leave right now and there is no replica whose left session time is longer than FRT (Fault

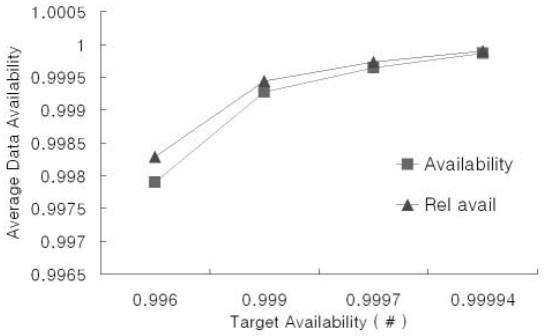
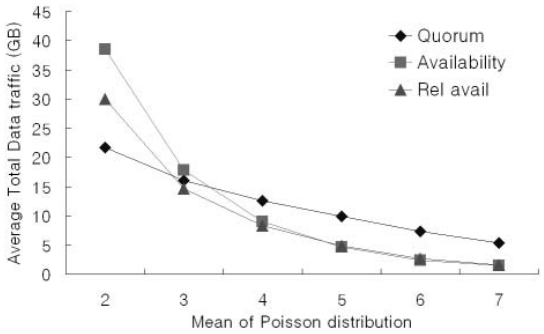
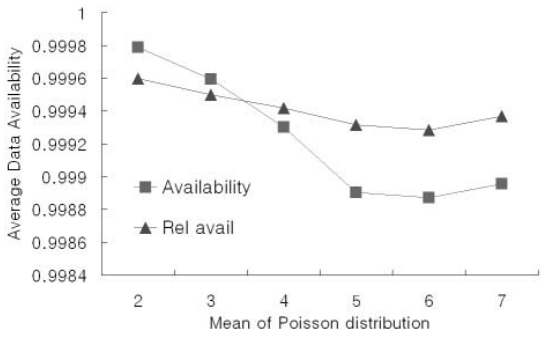


Figure 5. Comparison of Data availability with various target availability



(a) Data Traffic



(b) Data Availability

Figure 6. Comparison parameters with Various dynamic node behavior

Recovery Time), we count it as a data loss. We get the total data availability by dividing the total number of losses by the total attempt of checking the data availability. The figure 5 shows that the time-related replication improves the data availability. In the time-related replication, we try to get the primary replica which has enough time to replace the lost redundancy. Moreover, the sparse replicas spread on the timeline and the number of overlapped replicas lessens as best we can. These properties increase the probability of having any replicas which have enough time for the failure recovery. According to this, the time-related replication achieves higher data availability with less data copy traffic.

3.5 Effect of Dynamic Nodes

The figure 6 represents the comparison of data traffic and data availability with various mean of Poisson distribution. The target availability is 0.999. When the mean decreases, more nodes are short lived and have low node availability. Oppositely, when the mean increases, more nodes are long lived and have high node availability. As we expected, when the mean increases the data traffic decreases like figure 6(a). Generally the availability based replication requires less data copy traffic than the simple replication, but when the mean is small such as 2 and 3, it needs more data copy traffic. Especially, when the mean is 2 it need 2 times the data copy traffic. The time-related replication mitigates this phenomenon. In the figure 6(a), when the mean is 2, it reduces more about 25% of data copy traffic than the availability based replication. Moreover, when the mean is 3, its data traffic is less than the simple replication. It means that the time-related replication prevents from overlapping the replicas and makes the replicas more independent with each other.

We also compare the data availability with various node behaviors. In figure 6(b), the data availability of the availability based replication varies dynamically along the mean. When the mean is small, it picks more replicas which have low availability to keep the target data availability and overestimates the data availability. On the other side, when the mean is big, it relies on the highly available nodes too much and sometimes it loses data because of the overlapped replicas. As a result, it can not achieve the target data availability when the mean is bigger than 5. However, the time-related replication keeps the data availability above the target value and there is no remarkable fluctuation. The primary replica which has enough time for the failure recovery mitigates the overestimation of the data availability under the low mean. The sparse replicas which prevent the overlapped replicas help reducing the sudden loss of data under the high mean.

4 Related Works

The commercial p2p file sharing systems leave the data replication up to the popularity of the data. In this case, the popular data have very high data availability but the unpop-

ular data do not. To make the p2p storage system durable, the smart data replication methods is needed and the each inserted data is available for any time and has the similar data availability. [5], [6], and [7] use the quick replication which replaces the lost redundancy. However, this replication should be efficient. That is, the required data copy traffic should be minimized.

To address this problem, [1], [8] and [9] exploit the node availability to calculate the data availability by using the described availability model. Moreover, many approaches [10] [11] [12] try to improve the performance of the replication. [11] proposes that the node availability changes according to the number of total nodes. [12] suggests that the node availability means the steady-state availability and is the summation of the transient-state availabilities. As a result, the node availability is calculated by the function of time. [11] provides the proactive replication approach which is opposite concept to the quick replication. This can reduce data spikes which are caused by simultaneous node failures and normalize the required data traffic.

5 Conclusion

We propose and prove that the session time of replicas affects both of the data copy traffic and the data availability. Our time-related replication exploits the information of the replicas' session time, especially measured MTTF and prevents the overlapped replicas on the timeline. These sparse replicas mitigate the bursty failures and reduce the data copy traffic by 8%. Moreover, it uses a primary replica which has enough time to recover the failure. This helps keeping high data availability without any relation of the dynamic node behavior. Consequently, the proposed time-related replication achieves high data availability with low data copy traffic.

References

[1] K. Kim and D. Park. Reducing Data Replication Overhead in DHT Based Peer-to-Peer System. *In Proceedings of International Conference on High Performance Computing and Communications 2006*, September 2006

[2] K. Kim and D. Park. Efficient and Scalable Client Clustering For Web Proxy Cache. *IEICE Transaction on Information and Systems*, E86-D(9), September 2003.

[3] I.Stoica, R.Morris, D.Karger, M.F.Kaashoek, and H.Balakrishnan. Chord: a scalable peer-to-peer lookup service for internet applications. *In Proceedings of ACM SIGCOMM 2001*, August 2001.

[4] A.Rowstron and P.Druschel. Pastry: scalable, decentralized object location and routing for large-scale peer-to-peer systems. *In Proceedings of the International Conference on Distributed Systems Platforms(Middleware)*, November 2001.

[5] R. Bhagwan, K. Tati, Y. C. Cheng, S. Savage, and G. M. Voelker. Total Recall: System support for automated availability management. *In Proceedings of the 1st Symposium on Networked Systems Design and Implementation*, March 2004.

[6] B. G. Chun, F. Dabaek, A. Haeberlen, E. Sit, H. Weatherspoon, M. F. Kaashoek, J. Kubiatowicz, and R. Morris. Effcient replica maintenance for distributed storage systems. *In Proceedings of the 3rd Symposium on Networked Systems Design and Implementation*, May 2006.

[7] S. Rhea, B. Godfrey, B. Karp, J. Kubiatowicz, S. Ratnasamy, S. Shenker, I. Stoica, and H. Yu. OpenDHT: A public DHT service and its uses. *In Proceedings of the 2005 ACM SIGCOMM*, August 2005.

[8] C. Blake, and R. Rodrigues. High availability, scalable storage, dynamic peer networks: Pick two. *In Proceedings of the 9th HotOS*, May 2003.

[9] R. Bhagwan, S. Savage, and G. Voelker. Replication strategies for highly available peer-to-peer storage systems. *In Proceedings of FuDiCo: Future directions in Distributed Computing*, June 2002.

[10] E. Sit, A. Haeberlen, F. Dabek, B. G. Chun, H. Weatherspoon, R. Morris, M. F. Kaashoek, and J. Kubiatowicz. Proactive replication for data durability. *In Proceedings of IPTPS 2006*, February 2006.

[11] R. J. Dunn, J. Zahorjan, S. D. Gribble, and H. M. Levy. Presence-based availability and P2P systems. *In Proceedings of P2P 2005*, September 2005.

[12] J. Tian, Z. Yang and Y. Dai. A data Placement Scheme with Time-Related Model for P2P Storages. *In Proceedings of P2P 2007*, September 2007.

[13] S. Saroiu, P. K. Gummadi, and S. D. Gribble. A Measurement Study of Peer-to-Peer File Sharing Systems. *In Proceedings of MMCN 2002*, January 2002.