Statistical Analysis of a Hybrid Replication Model

Emerson Rogério de Oliveira Junior, Ingrid Jansch Porto
e-mail: emerson@inf.ufrgs.br, ingrid@inf.ufrgs.br
Phone: (+55)5133166161 Fax: (+55) 5133167308
Federal University of Rio Grande do Sul – Instituto de Informática
Caixa Postal 15064 Zip Code 91501-970 Porto Alegre BRAZIL

Abstract: Replication is a technique commonly used to provide high-availability and fault tolerance in distributed systems. With multiple copies of the entities, a service can keep operation even when some copies are inaccessible because of a crash of the computer where a copy was stored, for instance. There are two main classes of replication techniques: passive and active replication. Passive replication suffers from a high reconfiguration cost in case of failure on the primary, and active replication has permanent processing redundancy.

The hybrid replication technique presented in this paper has the same advantages of the passive replication in good runs, and has much less processing overhead than the active replication. In this paper, we demonstrate the efficiency of our replication model by the comparison among the response time (for the client) of the passive, active and hybrid replication scenarios using statistical analysis.

Key words: Distributed systems, replication, statistical analysis, high-availability

1. INTRODUCTION

One of the potential benefits of distributed systems is their use in providing high-available services, that is, services that are likely to be up and accessible when needed. Availability is a desirable metric to fault-tolerant systems. However, fault-tolerant systems always introduce redundancy in the system. With redundant copies, a replicated entity can continue

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35599-3_29

B. Kleinjohann et al. (eds.), *Design and Analysis of Distributed Embedded Systems*

providing services in spite of the failure of some copies, without affecting their clients. One example of redundancy is replication.

The replication technique can be used in different computing science areas. Some of them are databases, where the proposals of Kemme [6,7], Pedone [8], Helal [9], Patinõ-Martínez [10] and Wiessman [11] are good examples. Additionally, Baker's work in cluster computing [12] and Zhou's research [13] in real-time systems are also good references.

In embedded systems, it is very common the use of diskless systems. This implies the use of other stations to store data, mainly when we need stable memory (fail safe environment). In distributed systems, instead of using specially implemented stable memory systems, other nodes may be used as secondary memory hosts. However, conventional disks (used in most of the distributed nodes) may not be considered as stable memory - a set of nodes may be used with this goal, using replication management techniques.

In distributed systems, the literature distinguishes two main classes of replication techniques: passive replication [1] and active replication [2]. In passive replication, the client's request is sent to the primary replica that handles this request, sends messages to the other replicas in order to update their state, and sends back the response to the client. In active replication, the client's request is sent to all the replicas, every replica handles the request and sends back the reply to the client (just one reply is enough depending on the supposed failure modes). One drawback of the passive replication is to significantly increase the response time in case of failure in the primary copy. By other way, the active replication presents redundancy of processing (because all replicas handle the client's request) as a significant overhead.

Motivated by these drawbacks, this paper proposes the use of a combination on these techniques - passive replication and active replication - constituting a hybrid replication model. The hybrid replication has no increased response time presented by passive replication, in case of the primary failure and has much less processing usage than active replication model. Some experiments show some preliminary results that illustrate our ideas.

The rest of the paper is organized as follows. Section 2 introduces background concepts and related work about the replication models found in literature. We briefly present the statistical concepts used in the analysis in Section 3. Section 4 describes the specification of the hybrid replication model proposed in this paper, with the algorithm used and a discussion about the failure scenarios. Section 5 presents the results obtained about validation of the hybrid replication model, using statistical analysis. In section 6, we present our conclusions and some points to improve our research.

2. REPLICATION MODELS

Fault-tolerance in distributed systems is typically achieved through replication. The main replication techniques are passive replication and active replication. Other hybrid replication techniques, derived from the main techniques, are found in literature. These hybrid replication techniques are known by semi-passive replication [4] and semi-active replication [5].

In passive replication - also called primary-backup replication [1] - one replica, called primary, plays a special role: it receives the request from the client process, and sends the response back. The invocation is handled as follows (Figure 1):

- The client sends the request to the primary. Then, the primary receives the request and handles it. At the end of the operation, the state of the primary is updated and the primary sends the update message to the backups. Upon reception of this message, the backups update their states and send an acknowledgement (*ack*) back to the primary. Once the primary has received the *ack* from all (non-crashed backups, the response is sent to client.

The passive replication technique makes no assumption on the determinism of the requests – it allows for non-deterministic operations as it ensures the linearizability through the primary replica. The main disadvantage is that the implementation of passive replication requires a mechanism to agree on the primary (e.g., a group membership service). This leads to an increased response time in the case of primary failure [5].

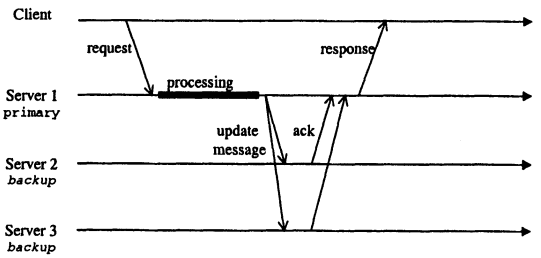

Figure 1. Passive replication.

In the active replication, also called state-machine approach [2], all replicas handle the client request. There is no centralized control, as in the passive replication. The request is handled as follows (Figure 2):

- The client's request is sent to all replicas and each replica processes the request, updates its state, and sends the response back to client. The client waits until it receives the first response or waits until it receives a majority of identical responses.
- To a client, all correct replicas should appear as having the same state. In order to guarantee this, all invocations sent by the clients should be

treated in the same order by all correct replicas. This is ensured by a total order multicast primitive – also called atomic multicast – that provides total ordering of messages multicast to a set of destination [15].

Active replication requires the operations on the replicas to be deterministic and waste extra resources through redundant processing, which is not the case of passive replication [3].

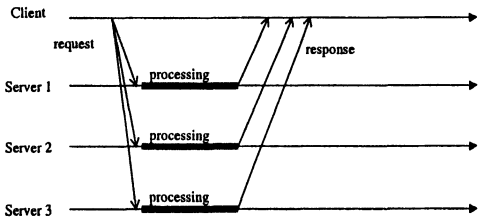

Figure 2: Active replication.

3. DEFINITION OF THE HYBRID REPLICATION MODEL

We consider an asynchronous system augmented with unreliable failure detectors and assume that the processors fail only under crash model. There are neither Byzantine failures nor link failures. The channels are reliable and the processes do not recover after a crash, as assumed by Chandra and Toueg [16]. In fact, as this system is to be implemented on a local network, the communication delays should be estimated with good results.

Similarly to passive replication, in the hybrid replication, the primary handles the requests and, after processing of each request, it sends an update message to the backups. In our solution, there are two primaries. We argue that the probability that one primary fail is smaller than the probability that two primaries fail simultaneously (before the system reconfigures and recovers from the last failure).

The hybrid replication model, illustrated in figure 3, has two primaries – named $\texttt{primary}_1$ and $\texttt{primary}_2$ and two (or more) backups. There is no restriction about the number of backups.

The actions performed by the hybrid replication model are:

- The client sends a request to both primaries ($\texttt{primary}_1$ and $\texttt{primary}_2$);
- Both, $\texttt{primary}_1$ and $\texttt{primary}_2$ handle the request and send an update message to the backups;
- When the backup states have been updated, an ack message is sent to primaries;

- The primaries send a response to the client. The client discards the second answer.

In this model, it is verified that there is more processing than the passive replication, but there is less than in active replication. The main advantage of the hybrid replication model is that, in case of crash failure of one primary, the other primary will send the response message to the client. This is not done by the passive replication because there is only one primary to process the client requisition.

We can assume that in the absence of server crash or failure suspicion, the client's request is handled at the same time with the hybrid replication as it was handled with the passive replication.

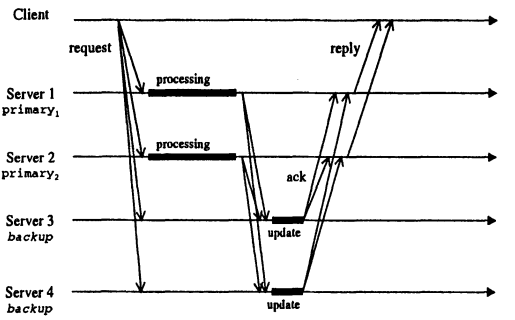

Figure 3: Hybrid replication model.

Six failure scenarios can occur in the hybrid replication:

1. One of the primaries fails before the client request is processed: in this case, the hybrid replication is equal to the passive replication in the sense that one of the backups will be elected to be the next primary and the client request processing will be finished.
2. One of the primaries fails during the processing of the client request: while in passive replication the client must resend the request to a new primary, in our hybrid replication is not necessary because the other primary processes the request.
3. One of the primaries fails after processing the request, but before sending the response to the client: the remaining primary receives the *ack* signal that was sent by the backups, and transmits the response to the client.
4. One of the primaries fails after sending the response to the backups: the other primary will continue the actions of the replication.
5. Both primaries fail: the new primaries will be elected among the remaining backups and the client must re-send the request to them.
6. One or more backups fail: the replication will proceed without interruptions because the primaries process the client request. In this case, new backups could be created. Notice that, in fact, as there are two

primaries, under one single faulty node assumption, even the acknowledgement of the backups is necessary to continue operation on the primaries (or to send the answer to the client).

4. STATISTICAL CONCEPTS

We use some statistical parameters to analyze our proposal. A statistic is a measure on the items in a random sample. Since the only reason to ever draw a random sample is to infer something about the population from which it came, it should be clear that when we calculate a given statistic we only do so in order to estimate a corresponding parameter of the population from which the sample was drawn.

When the evaluated values present a normal distribution, we can use the mean and the standard deviation as statistical parameters. Besides, if the distribution is not normal, we can use the Central Limit Theorem to prove our goals with statistics [14]. This allowed us to approximate the distribution of sample means with a normal distribution. This means that once we have the mean and standard deviation of the sample mean, we can construct a Z variable and compute probabilities using the standard normal distribution.

Another statistical evaluation is the hypothesis testing. In hypothesis testing, we start with a null hypothesis, H_0, which represents the status quo, or our default belief: it is what we continue to believe unless we judge that there is sufficient evidence to discard it [14].

5. VALIDATION OF THE HYBRID REPLICATION

In this section, we present the statistical analysis about the execution of the experiments with replication. To demonstrate the efficiency of our hybrid replication model, we have executed seven different experiments, which are:

- passive replication without failure;
- passive replication with failure in the primary;
- active replication without failure;
- active replication with failure in one of the servers;
- hybrid replication without failure;
- hybrid replication with one failure affecting the primary; and
- hybrid replication with two failures affecting both primaries.

The executions were performed using Linux, C language (gcc compiler) and sockets for the communication between the processes evolved. The experiments, with and without failures, were executed 1000 times each one

and we get the response time to the client request of each execution. So, we had 7000 response times for the client request.

The distribution obtained from the experiments was not a normal curve. All the histograms presented more than one mode. We believe that this occurs because the scheduling time used by the processor to execute the experiments is not a controllable variable, presenting random values. It causes a great response time variation. Because the distribution is not normal, we have used the Central Limit Theorem to evaluate the values presented in our experiments. Table 1 presents the statistical values observed about the response times obtained in our experiments without the occurrence of failures. Table 2 indicates the values obtained in executions with failures.

Table 1: Execution of the experiments without failures (in seconds).

Replication	Passive	Active	Hybrid
Amount of values (N)	1000	1000	1000
Mean (μ)	1,9054	1,1100	2,1155
Standard deviation (σ)	0,2541	0,4754	0,0452

Table 2: Execution of the experiments with failures (in seconds).

Replication	Passive	Active	Hybrid with failure in one primary	Hybrid with failure in both primaries
Amount of values (N)	1000	1000	1000	1000
Mean (μ)	5,9713	1,1113	2,2295	6,0371
Standard deviation (σ)	0,5720	0,0142	0,2938	0,8162

passive replication : mean = 1,9054s, standard deviation = 0,0008s;
active replication: mean = 1,1100s, standard deviation = 0,0150s;
hybrid replication: mean = 2,1155s, standard deviation = 0,0014s.

With the mean and standard deviation values, and using the probability of 95%, we have calculated the confidence interval (CI):

passive replication : CI = [1,9053s ; 1,9055s];
active replication: CI = [1,1091s ; 1,1109s];
hybrid replication: CI = [2,1154s ; 2,1156s].

From the presented values, we can be sure that, in 95% of the cases, the calculated means are in the presented confidence intervals. As it occurs in the execution with no failures, we observe that the response times obtained with the experiments with failures demonstrate that the distribution is not normal, either. The means are those presented in Table 2, but the standard deviation had to be modified according the Central Limit Theorem. So, we have that:

passive replication: mean = 5,9713s, standard deviation = 0,0181s;
active replication: mean = 1,1113s, standard deviation = 0,0004s;
hybrid replication with one failure primary: mean = 2,2295s, standard deviation = 0,0093s;
hybrid replication with two failure primaries: mean = 6,0371, standard deviation = 0,0258s.

With the mean and standard deviation values, and using the probability of 95%, we have calculated the confidence interval (CI):

passive replication: CI = [5,9702s ; 5,9724s];

active replication: CI = [1,1112s ; 1,1114s];

hybrid replication with one failure primary: CI = [2,2289s ; 2,2301s];

hybrid replication with two failure primaries: CI = [6,0355s ; 6,0387s].

As indicated by the presented values, in 95% of the cases, the calculated means of the measured times are in the presented confidence intervals.

5.1 Comparing execution with failures

As presented, it can be seen that the mean time needed to execute active replication with no failure is the smallest of all, and the biggest time is the observed in hybrid replication with no failure. It has happened before a very optimistic situation has been considered: just one client. In a concurrent scenario, order properties have to be considered and the numbers will be really different. In the case of executions with failure, the time presented by active replication is the smallest of all, again, and the hybrid replication with failure, in both primaries, is the biggest of all types of replication.

It can be observed that the mean time of hybrid replication with failure in both primaries is almost the same that the mean time needed with passive replication. Other interesting observation that can be considered is that the mean time needed to execute in hybrid replication with one failure is smaller than the mean time in passive replication. This is a goal that we have achieved with our replication model.

We argue that our hybrid replication model has much less processing usage than active replication model. To prove this, we have extracted the CPU time utilization needed to process the client's request in the experiments. The mean time to process the client's request was 0,2172s and the mean time to update the state of the servers was 0,0005s. Table 3 indicates the CPU mean time utilization of all seven experiments, considering how much times was processed the request and how much times it was necessary to update server states.

Table 3: CPU mean time utilization (in seconds).

Replication	CPU mean time processing	CPU mean time updating	Total CPU mean time
passive	1 x 0,2172s	2 x 0,0005s	0,2182s
active	3 x 0,2172s	0 x 0,0005s	0,6516s
hybrid	2 x 0,2172s	1 x 0,0005s	0,4349s
passive with failure	1 x 0,2172s	1 x 0,0005s	0,2177s
active with failure	2 x 0,2172s	0 x 0,0005s	0,4344s
hybrid with one failure primary	1 x 0,2172s	1 x 0,0005s	0,2177s
hybrid with two failure primaries	1 x 0,2172s	0 x 0,0005s	0,2172s

6. CONCLUDING REMARKS

Some observations can be extracted from the statistical analysis presented in this paper.

We can prove that the hybrid replication model with one failure uses the same amount of processing time that the passive replication model with no failure. To do this, we use the hypothesis test with values of Z distribution in passive replication with no failures *vs.* hybrid replication with one failure. The null hypothesis H0 is that the mean time of the experiments is the same. The Z observed value (Zo) is equal to –1,2142 and the Z critical value (Zc) is equal to 1,96. Comparing the absolute values achieved, we get that the relation $|Zo| < |Zc|$ is true, indicating that the null hypothesis must be accepted. This means that the mean time of passive replication with no failure and the hybrid replication with one failure is statistically the same.

The mean time needed to execute the hybrid replication with one failure was less than the mean time needed to execute the passive replication with one failure. It happens because in hybrid replication there is no intervention of the failure detector: the failure detector was only used in passive replication and in hybrid replication with failure in both primaries.

Another observation is that the mean time to execute the hybrid replication with two failure primaries is almost equal to the time to execute the passive replication. Comparing with the active replication, the hybrid replication needed less CPU time to process the request than the necessary in active replication.

In comparison with the active replication, the statistical analysis demonstrates that the response times, in mean, is greater than the response time observed in the hybrid model. However, our model executed the replication in less time than the necessary to execute the same replication, in active replication model.

The hybrid replication model presented can be used in an embedded system if the multicast primitives needed to reach replica consistency are inserted in this system. These primitives are necessary to guarantee the message changes between the replicas. In the same way, this replication model can be used in hard real time systems, too. In this case, if one primary fails to process the client request, the second one will continue with the replication action, preserving the time limits imposed by the hard real time system.

We observed that the use of the statistical analysis is a good manner to prove hypothesis. However, it was observed that a few papers found in literature, present their results using some type of statistical analysis.

At present, we are implementing the necessary framework to guarantee the consistency in our replication model. We expect that, with the replica

consistency, we will be able to run the experiments again with the same good results obtained until now.

7. REFERENCES

[1] BUDHIRAJA, N.; MARZULLO, K.; SCHNEIDER, F.; TOUEG, S. Optimal Primary-Backup Protocols. In: Int. Workshop on Distributed Algorithms (WDAG'92), Haifa – Israel, Proceedings, p. 362-378, 1992.

[2] SCHNEIDER, F.B. Replication Management using the State-Machine Approach, In Distributed Systems. p.169-198. Addison-Wesley. 1993.

[3] GUERRAOUI, R.; SCHIPER, A. Failure Tolerance by Replication in Distributed Systems. In Reliable Software Technologies. ADA-Europe'96, Proceedings, LNCS 1088, p. 38-57, Springer-Verlag, June 1996.

[4] DÉFAGO, X. Agreement-Related Problems: From Semi-Passive Replication to Totally Ordered Broadcast. n. 2229, 2000. 158p. (Ph.D. Thesis).

[5] POWELL, D. Delta-4: A Generic Architecture for Dependable Distributed Computing. ESPRIT Research Reports. Project 818/2252. Springer Verlag. 1991.

[6] KEMME, B.; ALONSO, G. A New Approach to Developing and Implementing Eager Database Replication Protocols. In ACM Trans on Database Systems, Sept. 2000.

[7] KEMME, B.; BARTOLI, A.; BABAOGLU, Ö. Online Reconfiguration in Replicated Databases Based on Group Communication. In: IEEE International Conference on Dependable Systems and Networks (DSN 2001), Proceedings, p.117..126. Goteborg, Sweden. July 2001.

[8] PEDONE, A.; KEMME, B. Exploiting Atomic Broadcast in Replicated Databases. In EUROPAR, Proceedings, Sep 1998.

[9] HELAL A. A.; HEDDAYA, A. A.; BHARGAVA, B. B. Replication Techniques in Distributed Systems. Kluwer Publishers, Boston-London-Dordrecht, 1996, 156p.

[10] PATIÑO-MARTÍNEZ, M.; JIMENEZ-PERIS, R.; KEMME, B.; ALONSO, B. Scalable Replication in Database Cluster. In: 14th Int. Symposium on Distributed Computing Systems (DISC2000), Proceedings, Toledo, Spain, Oct. 2000.

[11] WIESMANN, M.; PEDONE, F.; SCHIPER, A. Understanding Replication in Databases and Distributed Systems. In: 20th Int. Conf. on Distributed Computing Systems (ICDS2000), Proceedings, p.264-274, Taipei – R.O.C., Apr. 2000.

[12] BAKER, M. Cluster Computing White Paper. Univ. of Portsmouth, UK, July 2000.

[13] ZOU, H.; JAHANIAN, F. Optimization of a Real-Time Primary-Backup Replication Service. In: 17th IEEE Symp. on Reliable Distributed Systems (SRDS'98), Proc., West Lafayette, USA, p.177-185, 1998.

[14] SPIEGEL,M. Estatística. Coleção Schaum, McGraw-Hill do Brasil, RJ, 1980, 580p.

[15] FELBER, P. The CORBA Object Group Service: A Service Approach to Object Groups in CORBA. (Ph.D. Thesis). EPFL. 1998.

[16] CHANDRA, T. D.; TOUEG, S. Unreliable Failure Detectors for Reliable Distributed Systems. Journal of the ACM, 43(2). pp.225-267, 1996.