Optimization of Asynchronous Volume Replication Protocol

Huanqing Dong and Zhanhuai Li

Dept. of Computer Science and Engineering,
Northwestern Polytechnic University,
No.127 West Youyi Road, Xi'an, Shaanxi, China 710072
hqdong@co-think.com
lizhh@nwpu.edu.cn

Abstract. To keep data consistency on backup volume, traditional asynchronous volume replication protocol has to transmit every block changed on the Primary to the Backup and keep block update order during replication. This paper presents an enhanced asynchronous volume replication protocol, which can decrease the number of blocks required to be transmitted and updated to the Backup during replication. The enhancement is especially useful when network bandwidth is a major consideration. Analysis shows that in most cases the new protocol can keep the same data consistency as traditional protocols.

1 Introduction

A common approach to building fault-tolerant distributed systems is to replicate servers that fail independently. The objective is to give the clients the illusion of service that is provided by a single server. The main approaches for structuring fault-tolerant servers are active (state-machine) replication and passive (primary-backup) replication [1]. In active replication, a group of identical servers maintain copies of the system state. Client update operations are applied atomically to all of the replicas so that after detecting a server failure, the remaining servers can continue the service. Passive replication, on the other hand, distinguishes one replica as the primary server, which handles all client requests. A write operation at the primary server invokes the transmission of an update message to the backup servers. If the primary fails, a failover occurs and one of the backups becomes the new primary.

This paper focuses on the asynchronous primary-backup volume replication approach. Unlike database replication [3] or file replication [4], volume replication operates at block level, so it is unable to guarantee the consistency of volume contents from a database management system or file system viewpoint [6]. Only when a database management system or file system has no transactions outstanding and no updated data in cache that has not yet been written to disk, its on-volume image is said to be internally consistent. Though volume replication protocol is unable to ensure internal consistency on a volume, it must ensure that the Backup volume always represents a state of the primary volume at some previous point by keeping

X. Zhou et al. (Eds.): APPT 2003, LNCS 2834, pp. 140–146, 2003.

update order [1][6]. Otherwise, in case of a primary disaster, an application failed over to the Backup may not be able to recover.

Traditional asynchronous volume replication protocol keeps data consistency on the Backup by strictly maintaining block update order, each block update are applied on the Backup in the same order as they are received by the primary.

In this paper, we presented an enhanced asynchronous volume replication protocol (EAVRP), which can decrease the number of blocks required to be transmitted and updated to backup volume during replication. The main points are:

1. On the primary, for those blocks that have been updated more than 1 times in a given window, only the latest block image will be transmitted to the Backup, and all the blocks to be transmitted will be marked as an Atomic Block Group (ABG);
2. On the Backup, the commission of an ABG is carried out as an atomic operation, all blocks in the ABG are committed as a single update.

We compared our protocol with the traditional protocol. The comparison result shows that our protocol is more network bandwidth saving than the traditional one, and in most cases our protocol can maintain consistency as well as the traditional one does.

This paper is organized as the following. The next section discusses related work on asynchronous replication models and consistency metrics. Section 2 describes the EAVRP protocol by highlighting how it reduces the number of blocks to be transmitted during replication on the Primary. Section 3 compares the new protocol with the traditional protocol from the view of network bandwidth consumption and consistency. Section 4 concludes this paper.

2 Related Works

2.1 Asynchronous Replication Models

Many of past work are related to asynchronous replication model. An asynchronous volume replication protocol adopted by VERITAS Volume Replicator was described in [6], and a database replication protocol was described in [7]. Both protocols are focused on primary-backup replication model.

The protocol presented in [6] maintains a log in persistent storage on the Primary, the log contains all the blocks that have been updated to the primary volume and have not been acknowledged by the Backup, ordered by timestamps, which indicates the time when the block request arrived; the protocol presented in [7] maintains a vector on the primary, which contains all the transaction tuples that have been executed on the primary server and have note been acknowledged by the Backup, ordered by the time they are executed.

The protocol presented in this paper is based on the Primary-Backup replication model, and it is similar to the protocol presented in [6] [7] in the way of maintaining a log or queue in the Primary, the contribution of this paper is the method of saving network bandwidth.

2.2 Consistency Metrics

Window-inconsistency [1] is a metric to indicate the staleness of the Backup. Timestamps $\tau^P(t)$ and $\tau^B(t)$ identify successive versions of object O at the primary and backup sites, respectively. At time t, the Primary P has a copy of O written by the application at time $\tau^P(t)$, while the Backup B stores a possibly older, version originally written on P at time $\tau^B(t)$, i.e., at time t, B represents the state of P at $\tau^B(t)$. At time t, a backup copy of object O has window-inconsistency $t - t'$, where t' is the maximum time such that $t' \le t$ and $\tau^P(t') = \tau^B(t)$. The consistency metric *Staleness* presented in [5] is similar to Window-inconsistency. In this paper, we adopted the *Window-Inconsistency* metric to evaluate our EAVRP protocol.

3 The EAVRP Protocol

Each site (P and B) has two threads related to volume replication, namely *Primary Volume Update Thread* (PVUT), *Primary Transmission Thread* (PTT), *Backup Volume Update Thread* (BVUT) and *Backup Transmission Thread* (BTT). The architecture is shown in Fig. 1.

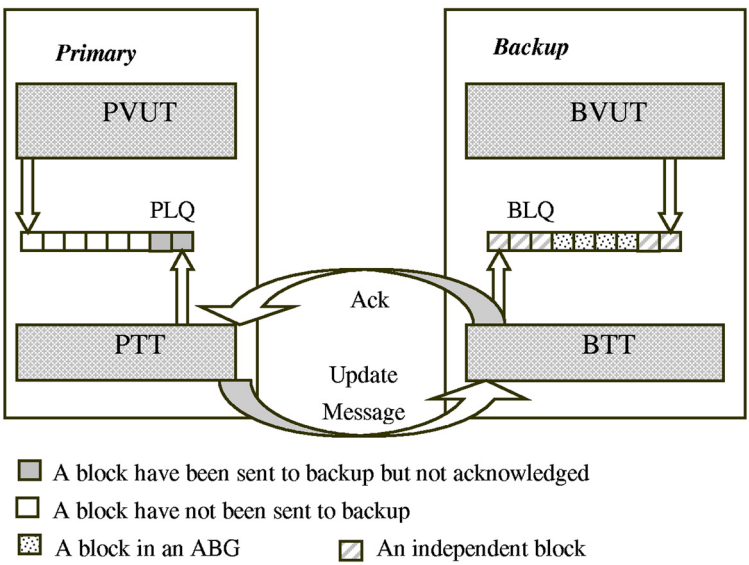

Fig. 1. Architecture of EAVRP protocol

3.1 Operations on the Primary

Definition 1. We define the *Primary Logical Queue* $PLQ = \{b_1, b_2, \cdots b_N\}$ to be the logical queue of blocks to be updated to B, N is the quantity of blocks currently in PLQ, blocks in PLQ are sorted by their timestamps in ascendant order. Physically, the PLQ can be arranged on a fixed length of continuous space.

Let S denote the quantity of blocks currently in PLQ that have been sent to B but not acknowledged, so the first block to be sent to B should be b_{S+1}.

Definition 2. We Define *Current Block Set* $CBS = \{b_i \mid S < i \le S + M\}$ to be the set of blocks from where we try to find duplicate block updates. The constant M represents the max quantity of block in an CBS, in the following description, we assume $M \le N - S$ is always TRUE.

Definition 3. Let n_i denote the block number (on volume) of b_i, We define *Current Duplicate Block Set* $CDBS = \{(b_i, b_j) \mid b_i, b_j \in CBS, and\ n_i = n_j\}$ to be a set of block pairs which represent duplicate block updates in the CBS.

Since both threads (PVUT and PTT) will operate on the PLQ independently, a lock mechanism is required to avoid conflict, each thread must obtain the lock before writing PLQ and must release it later. The lock mechanism is leaved out in this paper.

Upon accepting a block update request from the upper layer (File System or Database), PVUT will take the following steps:

1. Add the block in the update request to the end of PLQ;
2. Execute the update request (Update the block to logical volume), and return to the upper layer.

PTT operates as the following steps:

1. If exists $j, (b_{S+1}, b_j) \in CDBS$, then let $l = \{\max(w) \mid \exists i, (b_i, b_w) \in CDBS$, and $ABGOF = \{b_k \mid S+1 \le k \le l\}$ denote the original field of an ABG, figure out $ABG = \{b_k \mid b_k \in ABGOF, and\ \forall i, k < i \le l, (b_k, b_i) \notin CBDS\}$, which denotes the set of blocks to be sent to B and should be committed on B as an atomic operation. For duplicate block updates of a same block, only the latest block image will be included in the ABG.
2. Otherwise, send b_{S+1} as an independent block;
3. Check for arriving of acknowledge from B, if an acknowledge of an ABG arrives, remove all the blocks in corresponding ABGOF from PLQ; Otherwise, remove the acknowledged block from the head of PLQ;
4. Go to step 1.

3.2 Operations on the Backup

The BTT thread repeatedly receives block updates from P (PTT) and put them in the BLQ, blocks in BLQ are sorted by their timestamps in ascendant order.

To avoid out of order updates, the BVUT thread will commit all the blocks in an ABG as an atomic operation, and acknowledge the ABG as a whole; for independent blocks, BVUT will commit them according to their timestamps and acknowledge them one by one.

3.3 Failure Handling and Recovery

In case of a communication link error, some data (including block updates sent by the Primary and acknowledgments sent by the Backup) may be lost. But since each block will be kept in PLQ until the Backup has acknowledged it, The Primary will retransmit block updates from the first unacknowledged one when link recovers. If a communication link error occurs during replication of an ABG, then all the blocks in the ABG must be transmitted after link recovering.

Handling of other failures including overflow of PLQ, failure of the Primary Volume, etc., which are similar to that of [6], are leaved out in this paper.

4 Comparison of EAVRP with the Traditional Protocol

4.1 Network Bandwidth Consumption Comparison

As shown in Fig2, it is clear that EAVRP can reduce the quantity of blocks to be transmitted to the Backup when an ABG exists. In the example, 8 consecutive block updates (M was set to 8) were made to P, but only 5 blocks (In the form of an ABG) were transmitted to the Backup (B). In contrast, 8 blocks must be transmitted to the Backup (B') when traditional protocol is adopted. It can be deduced that the larger value of M is, the more possible that duplicate block updates would happen, so the fewer blocks are required to be transmitted to the Backup. But on the other hand, larger value of M will cause larger inconsistency window between the Primary and the Backup.

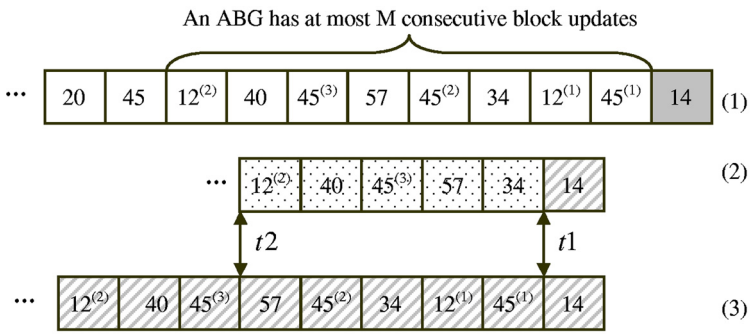

Fig. 2. (1) PLQ on the Primary; (2) BLQ on the Backup (B) when EAVRP protocol is adopted; (3) BLQ on the Backup (B') when traditional protocol is adopted

4.2 Window-Inconsistency Comparison

In this section, we will compare the staleness of the Backup (B) when using EAVRP and the staleness of the Backup (B') when using traditional protocol.

Definition 4. At time t, the backup has window-inconsistency $W(t) = t - t'$, where t' is the maximum time such that $t' \le t$ and $\tau^P(t') = \tau^B(t)$.

The value of $W(t)$ indicates the staleness of the Backup. Let $W_E(t)$ denote the window-inconsistency of the backup when EAVRP is used, and $W_T(t)$ denote the window-inconsistency of the backup when traditional protocol is used. Let's Assume $W_E(t) = W_T(t)$ before the ABG was transmitted to the Backup (at time $t1$, shown in Fig2).

1. Since B couldn't commit the ABG until all of its blocks have arrived, no update would happen on B during period $(t1, t2)$, while B' may update blocks incrementally. As shown in Fig2, during the period $(t1, t2)$, $W_E(t)$ will be greater than $W_T(t)$, i.e., B is staler than B' during period $(t1, t2)$.
2. Once the atomic update of the ABG is finished, B should have reflected all block updates in the ABG, $W_E(t)$ will be less than $W_T(t)$, i.e., B' is staler than B after $t2$.

The major disadvantage of EAVRP is that, if a primary fails when only part of the blocks in the ABG has been transmitted to the Backup, all the blocks in the ABG can't be committed on the Backup, so $W_E(t)$ will be larger than $W_T(t)$, it means more data loss if a recovery is to be done on the Backup at this time. Clearly, The distance between $W_E(t)$ and $W_T(t)$ is related to M, if M were set to 1, EAVRP would act the same as the traditional protocol do, and $W_E(t)$ would always be equal to $W_T(t)$.

5 Conclusion and Future Work

Traditionally, asynchronous volume replication protocol have to transmit every block changed on the Primary to the Backup, and each block must be updated to the Backup in the same order they are updated to the Primary. For replication in a WAN environment, in which case network bandwidth is a major consideration, it is very meaningful to decrease network traffic of replication procedure. Our protocol meet this goal by omitting some duplicate block updates when spreading updates to the Backup. The advantage of our protocol is that it can save network bandwidth, and the disadvantage is that the Backup may be staler than when traditional protocol is used.

Since the update order among blocks inside an ABG needn't be hold during the atomic commission, it is possible to optimize the atomic operation procedure in order to achieve better performance. We will investigate this issue in the future work.

The other issue we plan to study is how to decide the appropriate value of M . To do this, we must tradeoff between staleness of the Backup and network bandwidth consumption.

References

1. Ashish Mehra and Jennifer Rexford. Design and Evaluation of a Window-Consistent Replication Service[C]. IEEE Transactions on Computer, Vol. 46, NO.9, Sep 1997
2. Hengming Zou and Farnam Jahanian. Real-Time Primary-Backup Replication with Temporal Consistency Guarantees[C], In Proceedings of the IEEE International Conference on Distributed Computing Systems, June 1998
3. Rainer Gallersdorfer and Matthias Nicola. Improving Performance in Replicated Databases through Relaxed Coherency[C]. VLDB 95
4. Hurley, R.T and Soon Aun Yeap. File migration and file replication: a symbiotic relationship[C]. IEEE Transactions on Parallel and Distributed Systems, Volume 7, Issue 6, Jun 1996. Page(s): 578–586
5. Haifeng Yu and Amin Vahdat. The costs and limits of availability for replicated services[C]. In Proceedings of the 18th ACM symposium on operating systems principles, October 2001. Page(s): 29–42
6. Paul Massiglia. VERITAS Volume Replication and Oracle Databases [OL]. http://www.biffsocko.com/whitepapers/OracleAndVVR.pdf, VERITAS Corporation, 2000
7. Yang Zhaohong and Gong Yunzhan. A new primary lazy replica update protocol [J]. Computer Science, Vol29, No.8, Aug 2002.