The Flexible Replication Method in An Object-Oriented Data Storage System[1]

Youhui Zhang, Jinfeng Hu, Weimin Zheng

Institute of High Performance Computing Technology
Dept. of Computer Science, Tsinghua Univ.
100084, Beijing, P.R.C
zyh02@tsinghua.edu.cn

Abstract. This paper introduces a replication method for object-oriented data storage that is highly flexible to fit different applications to improve availability. In view of semantics of different applications, this method defines three data-consistency criteria and then developers are able to select the most appropriate criteria for their programs through storage APIs. One criterion realizes a quasi-linearizability consistency, which will cause non-linearizability in a low probability but may not impair the semantic of applications. Another is a weaker one that can be used by many Internet services to provide more read throughput, and the third implements a stronger consistency to fulfill strict linearizability. In addition, they all accord with one single algorithm frame and are different from each other in a few details. Compared with conventional application-specific replication methods, this method has higher flexibility.

1 Introduction

We implement an object-oriented data management layer as a cluster infrastructure software, specifically for the construction of Internet services. The impedance mismatch problem [1] is avoided because its interface is compatible with Java Data Objects API [2], an emerging standard for transparent data access in Java.

However, existing replication methods [3][4][5] cannot be adopted by OStorage. Therefore, we design one replication method that can be embedded in storage APIs. Users can employ it to set parameters on replication consistency and ease the development.

In the replication method, the principles, including high flexibility, low latency and adjustability are emphasized. The flexible replication method defines and implements three consistency criteria, soft consistency, general consistency and rough consistency. General consistency realizes a quasi-linearizability consistency, which will cause non-linearizability in a low possibility but do not impair the semantic of

[1] Supported by High Technology and Development Program of China (No. 2002AA1Z2103).

applications. Moreover, it does not introduce any total order multicast to solve consensus problem so that the latency is very low. Soft consistency is a weaker one that does not support linearizability at all to support more read throughput. And the third implements a stronger consistency to fulfill strict linearizability. They accord with one single algorithm frame and are different from each other in some details.

The rest of this paper is organized as follows. Section 2 presents three consistency criteria and emphasizes solutions of key issues. Section 3 summarizes this paper.

2 Replication Algorithms

To simplify the presentation of the method, the system model is described abstractly as follows.

- Reliable point-to-point communication is supported by the low-level network protocol with FIFO property. Reliability means the network layer guarantees a receiver will get the message in latency μ after sending, or the receiver can be assumed wrong.
- A system-global loosely synchronized clock is implemented using Network Time Protocol, whose max time error, τ, is less than 1ms.
- Nodes and network links may crash, but we assume Byzantine failures and network partition will not occur.
- The atomic data cell is called object, which is a data block with variable size.
- Three kinds of permitted commands upon an object are read, overwrite and recover-read, while creating and deleting are two forms of overwrite operation. When a crashed storage node, named Brick, recovers, the objects it holds is outdated and it will send a recover-read command to other Bricks to update data.

In addition, some terminologies are introduced.

- A command C is described as <type, O, D, timestamp> where type is one of overwrite, read and recover-read, O is the object to operate, D is null if type is not overwrite and timestamp denotes the time when this command is invoked.
- An object O is described as <OID, D, ctype, ctimestamp> where OID is its identification, ctype is the type of the latest command on it and ctimestamp is timestamp of that command.

Several copies of an object O are distributed to a set of Bricks that is noted as view(O). Each copy is called a replica. Before an objected O is created, some Bricks should be selected to form view(O).

2.1 General Consistency Algorithm

General consistency is almost equal to linearizability and can be used in most applications. A global loosely synchronized clock algorithm is applied to keep the order of commands in general consistency criteria. And there may be a tiny clock error

between every two nodes, which makes causality unsatisfied in a low probability. Its principles of different operations are described as follows.

Read: One read command is sent to any Brick in view(O). While no successful response returns, it will be redirected to another Brick until the data is obtained.

Write: A overwrite command is sent to every Brick in view(O) and waits for a response. Once the first successful response returns, the overwrite command is assumed accomplished. In addition, Thomas write rule [6] is employed here. That is, it attaches timestamps to commands and objects, and only the latest update is accepted. Then, the timestamp of an object is the one from its latest command.

2.2 Soft Consistency Algorithm

Linearizability is too strict for some applications that need faster response (especially for read). Soft consistency algorithm is therefore designed. Compared with linearizability, soft consistency is defined as follows.

CS is the set of all actual commands. The execution is said to be soft consistent if there exists a sequence S on CS satisfies these two conditions.

- Ca is before Cb in S if Ca<<Cb; But if Cb is a read command, it is not necessary to satisfy this condition.
- This condition is as same as that of linearizability.

It allows read commands to obtain stale data in fact. Compared with general consistency, it is only different in the implementation of read. That is, read command is executed as soon as received without concerning any other conditions. For many Internet services it is valuable. For example, a cluster-based email server maintains two mailboxes for every user, and it will save any email to both boxes. On reception of an email for user U, the server updates box u1 firstly. At that time, the user visits his/her box u2 that has not been refreshed, so he/she will get the stale data. But it does not impair the usage of mailboxes because users can imagine the email is still in transmission, which still observes email protocols.

2.3 Rough Consistency Algorithm

Rough consistency offers strict linearizability, but it introduces more latency. This algorithm employs a global token system [7] to replace the loosely synchronized clock.

Before a command is sent, a global token is applied to attach it whose number is increased seriatim. Bricks operate commands in accordance with the sequence of tokens, and it is prohibitive to operate a command which token number is not larger one than the prior. In this case, the Brick has to wait for commands between the twos.

In addition, only if all successful responses of every Brick in view(O) have returned, the overwrite command is assumed accomplished. So total order is maintained strictly and the adverse circumstance in general consistency will never happen.

2.4 How to Recover

When a crashed Brick recovers, the data it holds is outdated and should be refreshed. So it sends a recover-read command to other Bricks and buffers commands received until all data are updated. Because general consistency algorithm does not solve consensus problem, after carrying out a recover-read command on object o, one Brick may receive a overwrite command on O accomplished in another Brick before the recover-read command. Therefore linearizability cannot be maintained. The solution is a the command will be put off for a period of time, T_{lim} (equal to T) to execute.

3 Conclusion

General consistency realizes a quasi-linearizability consistency, which will cause non-linearizability in a low possibility but developers can adopt rough consistency or improve the applications to avoid this case. Soft consistency is a weaker one that permits applications to obtain stale data and is fit for some Internet services. Moreover, rough consistency realizes strict linearizability but introduces longer latency.

References

1. Steven D. Gribble and Eric A. Brewer and Joseph M. Hellerstein and David Culler. Scalable, Distributed Data Structures for Internet Service Construction, 4th Symposium on Operating System Design & Implementation, San Diego (2000).
2. C. Russell, Java Data Objects 1.0 Proposed Final Draft, JSR12, Sun Microsystems Inc., available from http://access1.sun.com/jdo (2001).
3. Idit Keidar. Totally Ordered Broadcast In The Face Of Network Partitions. Chapter 3 Exploiting Group Communication for Replication in Partitionable Networks, Laboratory for Computer Science Massachusetts Institute of Technology (1999).
4. R. Guerraoui and A. Schiper. Fault-Tolerance by Replication in Distributed Systems. In Proc Conference on Reliable Software Technologies (invited paper). Springer Verlag, LNCS 1088 (1996) 38-57.
5. David L. Mills. Improved algorithms for synchronizing computer network clocks. In the Proceedings of ACM SIGCOMM, London, UK (1994). 317–327.
6. Robert Thomas. A majority consensus approach to con-currency control for multiple copy databases. ACM Trans. on Database Systems (1979). 180-209.
7. Y. Amir, L. Moser, P. Melliar Smith, D. Agarwal, and P. Ciarfella. Fast message ordering and membership using a logical token-passing ring. In Proceedings of the 13th International Conference on Distributed Computing Systems, Pittsburgh, Pennsylvania, USA (1993). 551-560.