Future Generation Computer Systems 74 (2017) 232–240

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Research and implementation of a distributed transaction processing middleware

Jianjiang Li [a], Qian Ge [a], Jie Wu [b], Yue Li [a,*], Xiaolei Yang [a], Zhanning Ma [a]

[a] Department of Computer Science and Technology, University of Science and Technology Beijing, China

[b] Department of Computer and Information Sciences, Temple University, USA

HIGHLIGHTS

- A middleware-level distributed system is complemented for improving the performance of transaction processing.
- By making partition extension to Berkeley DB, this paper overcomes the disadvantage of non-support parallel writing across multiple nodes.
- Monitoring nodes of the distributed database system by the middleware ensures correct execution and migration of transaction.

ARTICLE INFO

Article history:
Received 1 September 2015
Received in revised form
8 January 2016
Accepted 30 January 2016
Available online 8 February 2016

Keywords:
Distributed system
Transaction processing
Middleware
Partition replication body

ABSTRACT

Currently, increasingly transactional requests require high-performance transaction processing systems as support. The performance of a distributed transaction processing system is superior to that of traditional single-node transaction processing system, and the characteristic of multi-node determines that distributed transaction processing systems should pay more attention to availability. For example, in traditional single-node systems, the performance of Berkeley DB is high, but its shortcoming of not supporting parallel writing across multiple nodes is weakening its availability and scalability in the distributed environment. This paper has designed and implemented a middleware-level distributed transaction processing system called POST, including a distributed database system called POSTBOX which is based on Berkeley DB and data partition, and a distributed transaction processing middleware called POSTMAN. POSTBOX inherits the availability of highly available Berkeley DB, and expands it with data partition. By Partition Replication Body (PRB), POSTBOX overcomes the native weakness of highly available Berkeley DB, which indicates that highly available Berkeley DB does not support parallel writing across multiple nodes; POSTMAN is a middleware adapting PRB. POSTMAN monitors POSTBOX in real-time via Partition Replication Body State Array (PRBSA), and ensures the correctness of transaction processing and transactions migration in the case of node failure. The actual test results show that POST possesses high availability, and has an obvious improvement of write performance compared with highly available Berkeley DB.

1. Introduction

Historically, OnLine Transaction Processing (OLTP) [1] refers to submitting traditional transactions such as ordering goods or transferring payments to the OLTP system, based on Relational DataBase Management System (RDBMS). With the rapid development of Internet and Internet application, transaction occurs some changes, one of the most significant features of which is the explosive growth of transaction throughput [2]. For instance, excellent multi-user game based on the web can produce a large amount of interactions within one second, and the growth of smart phone use and other mobile terminals has given rise to the development of mobile transaction. These Internet applications produce more transaction requests than the capability of the traditional OLTP system, and it is difficult for RDBMS to deal with high concurrent transaction requests. In addition, RDBMS cannot support expansion without offline and distribution very well. For example, SQL query [3] of a table with massive records in a relational database management system will cost an amount of time. Although it can be solved by data segmentation and table segmentation, it also increases the difficulties of programming, data backup, database

* Corresponding author.
E-mail addresses: lijianjiang@ustb.edu.cn (J. Li), greenday0925@gmail.com (Q. Ge), jiewu@temple.edu (J. Wu), liyuepkoneal@outlook.com (Y. Li), chinayangxiaolei@163.com (X. Yang), ningzhanma@163.com (Z. Ma).

http://dx.doi.org/10.1016/j.future.2016.01.021
0167-739X/

expansion and some other issues. In order to enhance the performance of system, the most direct solution is to purchase a machine which has stronger performance, but its higher cost is often prohibitive for most enterprises.

Distributing data and loading it to multiple nodes by using distributed database systems [4,5] is an effective method to improve the performance of the transaction processing system. Currently, application and deployment of a distributed system are becoming known more and more widely, especially in the background of expansion of cloud computing and big data [6,7]. Cloud computing [8,9] requires using low-cost servers instead of expensive machines as a hardware infrastructure platform, and obtains high availability and scalability through redundancy between nodes. Berkeley DB [10] is a powerful key/value database engine: its high availability version (referred to highly available Berkeley DB) provides a distributed database solution based on master–slave replication, with high availability and better reading scalability. Berkeley DB provides full ACID [11] transactional guarantees. This ensures that highly available Berkeley DB can be applied not only to lower requirements for data consistency (for example, state updates of social network users do not need to immediately synchronize to the entire application), but also to higher requirements for data consistency, such as financial systems or order processing systems, because these systems are intolerable to abandoning transaction and data consistency [12].

Highly available Berkeley DB supports high availability and read scalability, it does not have write scalability, that is to say, it does not support parallel write cross multiple nodes. In addition, compared to building a centralized or client/server system, it is quite difficult to build a truly distributed database system, because distributed database systems may have multi-node failures and problems with security of data storing [13,14], inter-node communication is relatively complex. Middleware is an effective way to solve the fault tolerance of distributed database systems, communication difficulties, and other problems. An effective distributed transaction processing middleware [15–18] can effectively manage a distributed database system, and reduce the programming difficulties.

In response to these problems, this paper designs and implements a distributed transaction processing middleware system called POST, which consists of a distributed database system called POSTBOX based on Berkeley DB and data partition, and a distributed transaction processing middleware called POSTMAN. POSTBOX makes partition extension to highly available Berkeley DB, and overcomes the problem that Berkeley DB does not support parallel write cross multiple nodes via Partition Replication Body (PRB). POSTMAN, which is deployed on top of POSTBOX, and fully adapted to the PRB of POSTBOX, can provide an access interface for the interaction application and POSTBOX. POSTMAN monitors the status of each node of POSTBOX by Partition Replication Body State Array (PRBSA), and ensures correct execution and migration of transaction when a node fails through an efficient scheduling mechanism.

The rest of this paper is organized as follows. Section 2 describes the highly available of Berkeley DB. Section 3 describes the system architecture of POST, and introduces the distributed database system called POSTBOX based on Berkeley DB and data partition, and distributed transaction processing middleware called POSTMAN. Section 4 provides analysis of the availability and performance of the POST system. Section 5 provides experimental results and analyses. Section 6 introduces related work. Finally, this paper makes a summary in Section 7.

2. High availability of Berkeley DB

Berkeley DB achieves high availability by the replication group. Replication group is a collection of Berkeley DB environments

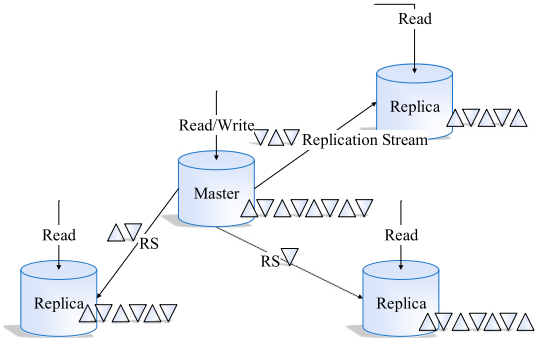

Fig. 1. Replication stream of highly available Berkeley DB.

distributed on different physical nodes. Nodes in replication group have the following three states:

(1) Master: "Master node" is chosen by a simple majority of electable nodes. It can process both read and write transactions.
(2) Replica: "Replica node" is in communication with a Master node via a replication stream which is used to keep track of changes made at the Master node. It only supports read transactions.
(3) Detached: "Detached node" has been shut down. It is still a member of the replication group, but is not an active node. A node that is not in the detached state is also referred to as being active.

There is only one Master node in the replication group. The Master node can read and write data, and the Replica node can only read data. The Master node and Replica node both belong to active nodes in the replication group. One node in a detached state means that this node is not an active node, and it is still a member of a replication group. The following introduces a replication group in two aspects of the replication stream and generation mechanism of the Master node.

2.1. Replication stream

Write transactions executed on a Master node are replicated to Replica nodes by a logical replication stream established on a TCP/IP connection. This connection is a dedicated connection between a Master node and each Replica node. Replication stream contains some descriptions of the logical changes. These logical operations are computed from log entries of the current Master node and are replayed at each Replica node by an efficient internal replay mechanism.

As shown in Fig. 1, there are four nodes in a replication group, including a Master node and three Replica nodes. The Master node has completed eight write operations (the log entry is represented by a white and gray triangle), and because the progress of the three Replica nodes is behind the Master node, the Master node sends log entries to all Replica nodes by a replication stream (dashed line with an arrow), so that all Replica nodes can replay according to the log entry to keep up with the progress of the Master node. In a replication group, Replica nodes are distributed to multiple physical machines, so this can ensure that a single failure cannot affect other nodes.

2.2. The generation process of master node

In a replication group, only the Master node has write access. When it is shut down, it is essential for the replication group to regenerate a new Master node. The generation of a Master node is influenced by two factors: the log progress and the priority of the