RC24412 (C0711-010) November 13, 2007
Computer Science

IBM Research Report

A Database Scale-Out Solution for Emerging Write-Intensive Commercial Workloads

Yi Ge, Chen Wang, Xiaowei Shen, Honesty Young
IBM Research Division
China Research Laboratory
Building 19, Zhouguancun Software Park
8 Dongbeiwang West Road, Haidian District
Beijing, 100094
P.R.China

Research Division
Almaden - Austin - Beijing - Cambridge - Haifa - India - T. J. Watson - Tokyo - Zurich

A Database Scale-Out Solution for Emerging Write-Intensive Commercial Workloads

Yi Ge, Chen Wang, Xiaowei Shen, Honesty Young

IBM China Research Lab

Contact email: {geyi,wangcwc,xwshen,honesty}@cn.ibm.com

ABSTRACT

Among the increasing number of online businesses, a series of write-intensive commercial workloads are emerging recently on the Internet. These workloads generate many more write transactions on the backend database than read transactions. Most of such workloads require the database to handle high-volume write transactions in real-time. Based on the observation on the workloads, this paper proposes a multi-tier database scale-out architecture with the write caching technology. Our preliminary result shows that the database scale-out architecture can handle extremely high-volume write transactions with excellent scalability.

1. Introduction

A large number of commercial workloads are shifted from offline to online, and converged to a small number of web-based enterprises, such as eBay, Yahoo and Google. To cope with extremely high-volume transactions in these workloads, scale-out technologies have been introduced in database configurations and implementations. For example, Data Partitioning is one of the most popular features in commercial database products to scale out the database system for performance. Moreover, for read-intensive commercial workloads, the database bottleneck could be eliminated or reduced by leveraging the scale-out capability of middleware in a 3-tier database access architecture. Database caching technology [1][2] provides a mechanism to maintain data records or query results at the middleware layer, therefore applications do not need to access the backend database at run time. The total amount of cache size grows with the number of servers. The emergence of rack-mounted servers and blade servers makes it cost-effective to build a scale-out solution for commercial workloads.

With increasing number of businesses on the network, a series of the write-intensive commercial workloads are emerging. In commercial workloads such as billing, monitoring on-the-fly, online gaming and virtual community, more write transactions are generated than read transactions. In a typical online lottery application, for example, seventeen of total twenty-seven SQL transactions in a core function are write operations. The database is responsible for logging all the steps of lottery games in real-time. Near a billion transactions can be issued to the backend database in a day. In most cases, database write operations are the bottleneck for such write-intensive workloads. Not only the majority of database operations are update transactions (Insert/Update/Delete SQLs), but also the write transaction processing is more complex and easier to cause I/O and lock contentions. A new database solution is required to handle such high-volume write-intensive workloads. Database scale-out is probably the only feasible solution to meet the throughput requirement. However, resource contentions and consistence synchronization can greatly impair the effectiveness and scalability of the solution. It becomes even worse on write-intensive workloads that usually involve frequent synchronizations for data consistency. Based on the analysis of database systems and the characteristics of emerging write-intensive workloads, this paper proposes a novel architecture for building a scale-out solution for such commercial applications.

2. Database Scale-out Architecture

For emerging write-intensive commercial workloads, we find that the heavy load of write transactions comes from the confluence of multiple user threads. Each thread focuses on the transactional logic to deal with a single user status or account. Most of the time transactions associated with an account can be processed on the same node of a cluster system. The most severe contention happens for the branch balance. However, the business logic does not check the branch balance frequently. A middleware level solution could be devised to provide the current branch balance when requested.

Furthermore, simple SQL commands are commonly used in write transactions. That is, each operation modifies only on a single table or just a single row. And, a couple of most frequently updated tables in the database receive most of the write requests. For those infrequently modified tables, previous database scale-out technologies such as database caching can be adopted. In the following discussion, we only focus on the scale-out solution for frequently updated tables.

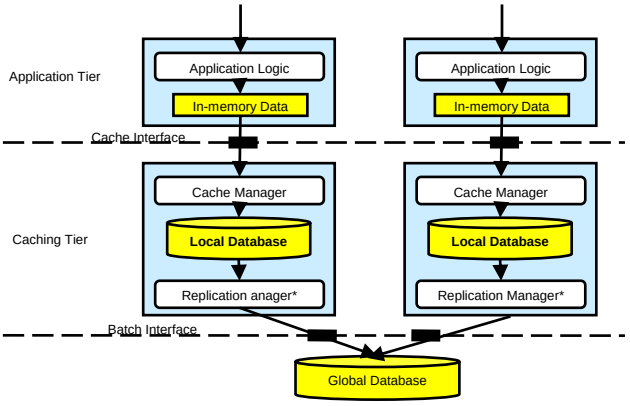

Fig.1 database scale-out architecture for write transactions

As shown in Fig.1, the proposed system architecture introduces a new database caching layer (front-end layer) between normal application logic and the global database layer. In other words, database write transactions are processed and temporarily cached at the local databases of the caching layer. The local databases only keep the most frequently updated tables and provide the

same database interface to application logics. From the aspect of application logics, transactions behave as if they were committed or aborted at the unified database layer. In fact, transaction requests arriving at database caching interface are properly routed. Database hit requests are handled at the caching layer: the records of the corresponding requests are cached in local databases and the results are returned to applications. Meanwhile, other transaction requests bypass the caching layer, and are directly processed at the global database(backend database). The caching layer validates transaction requests, and logs the corresponding data records separately. Alternatively, the system may choose to log the SQL commands and submit them to the global database for later execution. The replication manager first replicates the completed records to the global database in a batch mode periodically and asynchronously, and then drops those temporary copies at the caching layer.

Each node or partition handles write transaction requests separately, and replicates the latest records of completed transactions to the global database. By maintaining data dependency only within the single node, the caching layer can scale out without synchronization overhead, achieving close to linear scalability. Front-end nodes are treated as the transaction processing engines rather than record owners. This greatly simplifies the configuration of traffic router and load balancer, as well as node failure recovery.

In our proposed architecture, arriving requests are first maintained in buffers at the caching layer for a short duration, and later submitted in a batch to the local database. The application returns after obtaining the results from the corresponding local database. General performance enhancement is achieved from the batch operation to reduce time spent on the execution plan, log entry I/O, and spare space seeking. Requests are dispatched to caching buffers according to the policies as following:

1) Single-statement transactions can go to any pending buffer. When the buffer is full, all the transactions in the buffer are submitted to the local database in a batch.

2) All requests from a multi-statement transaction are scheduled into one buffer. The transactions are committed as a whole when a user performs the commit operation.

This technique helps guarantee the ACID properties of transactions while still maintaining low response time for database write requests. Moreover, an adaptive sizing technique has been developed for the balance between the response latency and the transaction throughput.

Leveraging the spare data storage and processing capabilities at the front-end layer, a replication manager has little performance impact on the transaction, in terms of transaction response time and throughput. Asynchronous and batch replication can significantly reduce the contention at the global database. The integrity and constraint checking could be skipped in the backend database, since the corresponding validations have already been performed at the front-end layer. In our scale-out prototype, the gross overhead for one node replication is less than 2% at the backend database server. For workloads with burst transactions, the adaptive batching technique in replication manager can achieve load-balance between application servers and the backend database.

3. Experiment Results

In the scale-out experiments, 16 IBM blade servers are used as the front-end database caching nodes. The backend node is an X-server with two AMD dual-core processors (Opteron™ 64, 8GB memory). Multiple application servers simulate an online lottery gaming logic and generate the corresponding transaction requests, including read and write operations. The requests are distributed to the front-end nodes, handled in this layer or routed to the backend database. Both the throughput and backend CPU utilization are monitored to evaluate the overall scalability. Two kinds of the scalabilities are evaluated in the tests: 1) the scalability of front-end nodes(backend nodes perform no write caching and are not considered in the scalability calculation); 2) and the overall scalability among all the database nodes, including both front-end and backend(the backend node also has caching logic to get full utilization on the system). For n-node scalability, we divide the n-node throughput by means of n single node throughput. The results are shown in Fig.2.

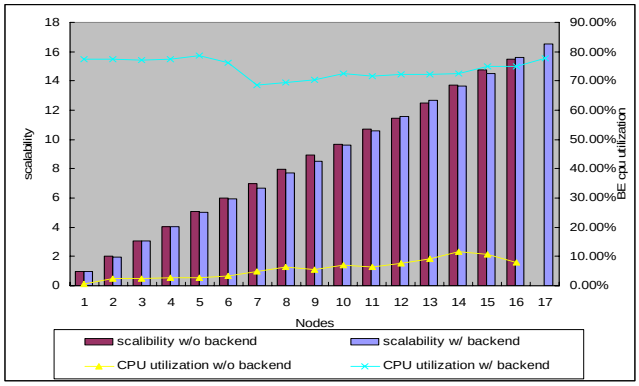

Fig.2 Scalability and CPU utilization of database write caching architecture

The database write caching architecture has close to linear scalability. As can be seen in Fig.2, our preliminary experiment result shows that the database write caching architecture achieves the scalability over 0.97 on a sixteen-node platform. The overall CPU utilization for sixteen-node replication in the backend node is approximately 10%, which is equivalent to less than 1% per front-end node. This has further demonstrated that the backend server can efficiently support a large number of front-end nodes simultaneously.

4. REFERENCES

[1] C. Bornhövd, M. Altinel, Sailesh K., *etc*., DBCache: middle-tier database caching for highly scalable e-business architectures. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data，San Diego, California, 2003, 662.

[2] C. Bornhövd, M. Altinel, C. Mohan, *etc*., Adaptive Database Caching with DBCache. IEEE Data Eng. Bull. 27(2), 2004, 11-18.

[3] Philip A. Bernstein, Alan Fekete, Hongfei Guo, *etc*., Relaxed-currency serializability for middle-tier caching and replication. In Proceedings of the ACM SIGMOD'06, 599-610.