"InfoFrame Relational Store," a New Scale-Out Database for Big Data

SUKENARI Teruki, TAMURA Minoru

Abstract

Existing relational databases experience problems when used with big data because they cannot deal flexibly with increases in the amount of data and number of accesses (scaling out). On the other hand, key-value store is an advanced technology but it is also troubled by the problem of lacking data access with SQL and the transaction processing required for mission-critical operations. The InfoFrame Relational Store (IERS) is a scale-out-capable database software optimal for big data utilization equipped with 1) SQL interfacing, 2) transaction processing and 3) high reliability that makes it applicable to mission-critical operations. This paper introduces the features of the IERS and its architecture.

Keywords

big data, key-value store, scale-out, SQL, transaction, small start
New SQL, mission-critical operations, memory database, high reliability

1. Introduction

The recent expansion of cloud computing services and the rapid dissemination of smartphones and sensor devices have brought an explosive increase in the amount of data distributed inside and outside enterprises, including user information, data collected from various sensors and text data sent by SNS. It is now indispensable to utilize this "big data" overflowing around enterprises in order to improve their competitiveness.

The utilization of big data requires a database for storing these large and increasing amounts of data. The traditional RDB (Relational DataBase) is capable of storing big data through partitioning into multiple databases. However, with an RDB, the application needs to process data using the data shared between databases. As a result, extending the database scale in the middle of operations by adding new servers according to the increase in data quantity and access count, or scaling out, has been difficult.

On the other hand, KVS (Key Value Store) is attracting attention as a database capable of scaling out. KVS achieves scale-out using a structure that manages data as pairs of "the key that is the index of the data" and "the value that is the actual data." Nevertheless, since KVS is not capable of transaction processing and is not equipped with a unified standard interface such as SQL (Structured Query Language), the application developer must master the design skill for each KVS product to be used.

At NEC, we recently developed the InfoFrame Relational Store (IERS), a scale-out database optimal for big data utilization, which makes it easy to design a system featuring flexible scalability according to increases in data quantity and access count as well as high reliability, and started providing it in April 2012.

2. InfoFrame Relational Store, a Database Optimal for Big Data Utilization

The IERS is a new breed of database software for the big data era, providing both the advantages of the existing RDB and those of the advanced KVS technology (**Table**). It can build a system using commodity servers and improve data processing capability linearly by increasing these servers. This section describes the individual features of the IERS and their effects.

2.1 Scale–out

When the amount of data to be stored with the IERS or the number of accesses to the IERS increases, the database scale

Table Features of the InfoFrame Relational Store.

	RDB	KVS	InfoFrame Relational Store
Scale-out (flexibly scalable depending on increases in data quantity and access count)	×	○	◎
SQL interface	◎	×	○
Transaction processing	○	×	○
High reliability	○	×	○

◎: Easily possible ○: Possible ×: Impossible

(the data quantity that can be stored and the access count that can be accommodated) can be extended (scaled out) linearly by adding new servers. The IERS can process queries from applications even in the middle of scale-out, so operations do not need to be stopped during system extension as is the case with RDB.

The initial installation cost of an RDB is very high because it is necessary to predict future demands (maximum data quantity and access count) and introduce a database with a scale supporting the predicted demands from the beginning. With the IERS, on the other hand, a small start with a lower initial installation cost than with an RDB is possible because it is only necessary to build a database matching the current demand.

2.2 SQL Interface

The IERS can develop business applications using the SQL interface.

Application assets developed for an RDB can be reused for the IERS and applications can be developed using SQL, which is already familiar to many developers. When we introduced the IERS for advance testing in the image management service of BIGLOBE, an internet service provider, we were able to demonstrate a high diverting capability for 99% of existing application programs. (The diversion rate is dependent on the types and number of SQL statements issued in the application program.)

2.3 Transaction Processing

Since KVS uses an architecture in which duplicated key-value data is distributed among multiple servers, it has difficulty with transaction processing that updates data while maintaining data consistency. The IERS adopts the "micro-sharding mechanism" [1] developed by NEC Laboratories America, Inc. to implement the transaction processing indispensable for mission-critical operations.

Support of transaction processing allows the IERS to be applied to online transaction services that have been unable to use KVS because of considerations of data consistency (e-commerce sites, telecommunications carriers, financial institutions, etc.).

2.4 High Reliability

The servers processing transactions (transaction servers) and those storing actual data (storage servers) save data in multiplexed storage on multiple servers.

A transaction server performs synchronous replication of the data managed by the master server on backup servers (two or more units recommended). When a backup server detects a fault with the master server, it can take over master server operations and complete recovery within one second.

3. Architecture of the InfoFrame Relational Store

The IERS is composed of 1) Partiqle servers that process data accesses from applications, 2) transaction servers that execute in-memory transaction processing by minimizing disk accesses and 3) storage servers that store the actual data (key-value data) on disks for permanent storage (**Fig. 1**). Any of these servers can be built using a commodity server on which Red Hat Enterprise Linux 5 can run and the storage servers store the key-value data on the built-in disks of the commodity servers.

This system enables meticulous scale-out according to the required purpose by adding Partiqle servers against increases in application accesses, transaction servers against increases in updated data quantity and storage servers against increases in stored data quantity. Any server can be scaled out while applications are running, so the system can be applied to mission-critical operations.

3.1 Partiqle Servers

An application can utilize the SQL interfaces provided by

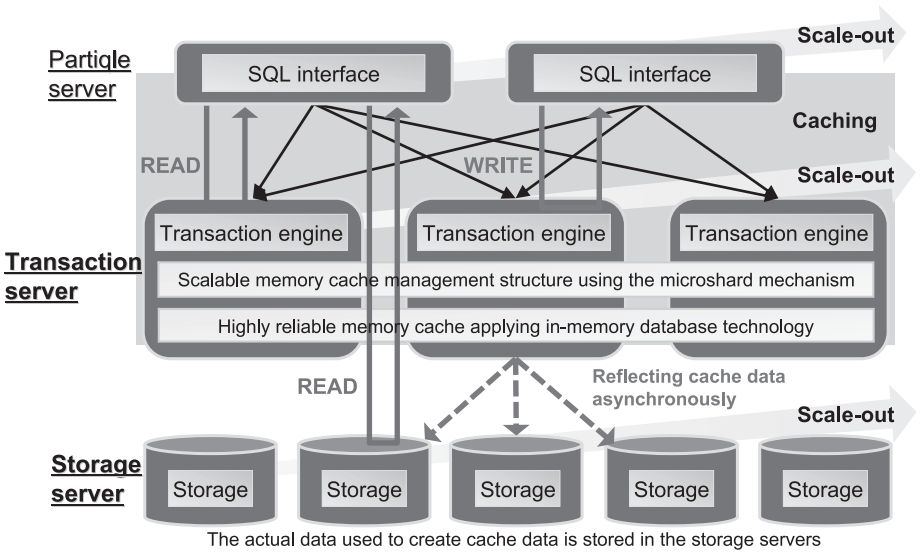

Fig. 1 Architecture of the InfoFrame Relational Store.

Partiqle servers by loading the JDBC driver for IERS. After connecting to the desired Partiqle server, the application issues SQL statements such as SELECT, UPDATE or INSERT to the Partiqle server. The Partiqle server interprets the SQL received from the application and automatically creates an execution plan out of the API for controlling the key-value data (hereinafter called the "KVS-API"). If it is a transaction execution plan, it acquires the data to be updated/referenced in the transaction from the storage server and loads the data into the memory of a transaction server for updating and referencing. If it is an execution plan of a reference request, not a transaction, it references the data to be stored in the storage server directly by bypassing the transaction servers.

The possibility of using SQL interface frees application developers from the need to have knowledge of the KVS-API. However, the execution plan created automatically by a Partiqle server controls the key-value data by specifying the key in the same way as with KVS. As a result, if the SQL is a reference request like SELECT, for example, it is suitable for big data search using a key (which corresponds to the index with an RDB) (a search in which the key is specified as the equal condition for the WHERE clause) but not suitable for a full-data search. It is therefore necessary to design the workload with an awareness of these properties.

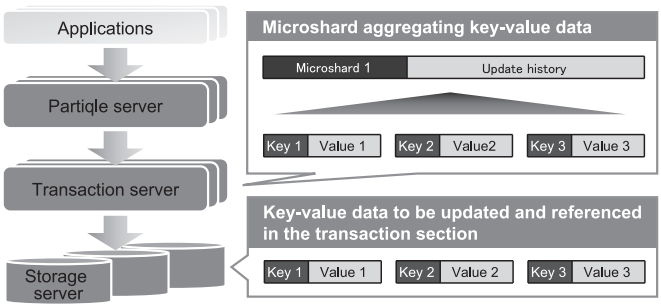

Fig. 2 Microsharding mechanism.

3.2 Transaction Servers

Transaction processing can be used by means of the microsharding mechanism (**Fig. 2**). Upon reception of a transaction, the transaction server collects all the key-value data to be updated or referenced in the transaction section from the storage servers. Applying the in-memory database technology, the transaction server aggregates the collected multiple items of key-value data into logical units called "microshards" for use as memory caches. The transaction server completes the transaction at the moment it has written the updating logs in the

microshards into memory and returns a commitment success notification to the application.

With transaction processing using the microsharding mechanism, the transaction is completed on a single server, rather than multiple servers, so highly efficient transactions are possible without using complicated commit processing such as the two-phase commit protocol. Since a transaction can be completed by appending an update log to the microshards in the memory of a transaction server, transactions can be processed at high speed by minimizing disk accesses to the storage servers.

The microshards in memory are saved on multiplexed storage by means of synchronous replication by multiple transaction servers. In addition, the update history of the multiple items of key-value data aggregated in the microshards is reflected asynchronously on the disks of storage servers at a certain interval for permanent storage. The multiplexing and permanent storage of data make it possible to protect data securely even in cases of transaction server failure.

Even if the transaction server working as the master server fails, the backup transaction server detects the failure from the "heartbeats" exchanged periodically between transaction servers and promotes itself automatically as the master server to take over its operations.

Other technical features of the transaction servers include the following:

- **Simultaneous transaction execution control using optimistic lock**
  RDBs generally employ a pessimistic lock, which locks the update target data of execution of a transaction while applying exclusive processing to other transactions. On the other hand, the IERS adopts an optimistic lock, which reads the version number embedded in the update target data (microshard) in advance and applies exclusive processing by comparing the read-out version number with that after the data write. If the two version numbers are different, it is judged that another transaction that has been processed in parallel has updated the data before the current transaction and that the commit has therefore failed. As the optimistic lock does not use shared resources such as lock control information, it makes scale-out easier.
- **Auto-leveling of transaction server processing loads**
  Transaction server memory usage is permanently monitored and the processing amounts of transactions are redistributed automatically so that the processing loads of the transaction servers are leveled. This makes it possible to utilize all of the transaction server resources effectively in a well-balanced manner even when the number of servers has increased after repeated scale-outs.

3.3 Storage Servers

The storage servers employ Voldemort [2], an open-source KVS system. The key-value data received from the transaction servers is saved using multiplexed storage on multiple storage servers for permanent storage.

When a new storage server is added, part of the data stored on other storage servers is migrated to the newly added server. This contributes to leveling the deviations in disk usage between storage servers. If the data migration source server receives a referencing request during migration and cannot find the requested data, the data migration destination server is identified and the reference request is routed to the destination server. This ensures scale-out of storage servers without interrupting operations.

4. Conclusion

The InfoFrame Relational Store is a scale-out-capable database software that lowers the hurdles of big data utilization by means of 1) data access using SQL, 2) transaction processing using the microsharding mechanism and in-memory database technology and 3) high reliability to make the software applicable to mission-critical operations.

In the future, we will study linkage with Hadoop to deal with processing for which performance is hard to manifest due to coverage of a wide range of data (batch processing, etc.) and also expand the SQL support range. At NEC, we are determined to enhance this product continually through joint verification with customers and to promote other efforts toward the more active utilization of big data.

References

1) Junichi Tatemura, Oliver Po, Hakan Hacigumus: “Microsharding: A Declarative Approach to Support Elastic OLTP Workloads,” ACM SIGOPS Operating Systems Review, Vol. 46, Issue 1, 2012.1
2) Project Voldemort:A distributed database,(refer to 2012.7) http://project-voldemort.com/

Authors' Profiles

SUKENARI Teruki
Assistant Manager
3rd IT Software Division
IT Software Operations Unit

TAMURA Minoru
Manager
3rd IT Software Division
IT Software Operations Unit

Information about the NEC Technical Journal

Thank you for reading the paper.
If you are interested in the NEC Technical Journal, you can also read other papers on our website.

Link to NEC Technical Journal website

Japanese

English

Vol.7 No.2 Big Data

Vol.7 No.2
September, 2012

Special Issue TOP