InfoFrame DataBooster for High-speed Processing of Big Data

KAWABATA Terumasa, HAMADA Mitsuyasu, TAMURA Minoru, HAKUBA Tomohiro

Abstract

The age of the information explosion has recently been raising various opportunities for big data analysis. On the other hand, the progress in hardware developments has seen a significant increase in the capacities of mountable memories. The InfoFrame DataBooster has been meeting the needs for high-speed processing of big data by using an in-memory data processing technology based on column store. This paper introduces differences between the regular RDB and the InfoFrame DataBooster. It goes on to discuss the background and method of development of the SQL interface of the latest version, the method of use of the InfoFrame Databooster and the application domains (case studies).

Keywords

in-memory, database, high-speed batch processing, column store, SQL

1. Introduction

According to the Green IT Promotion Council, the amount of information distributed in society in 2025 will be 200 times that of 2006. This will be caused by a rapid increase in the data generated by sensors and other devices, structured data such as corporate transaction data and log data, and unstructured data such as video, audio and Internet images, TV and radio. However, the distributed data is still not fully utilized, which makes the creation of new values from the large amount of data an important issue for the future.

In order to achieve this issue, the necessity of processing and analyzing various types of big data is increasing dramatically. At NEC, we categorize the big data processing products into three layers. These are the "data/media analysis layer," the "correction/integration layer" and the "platform layer" (**Fig. 1**).

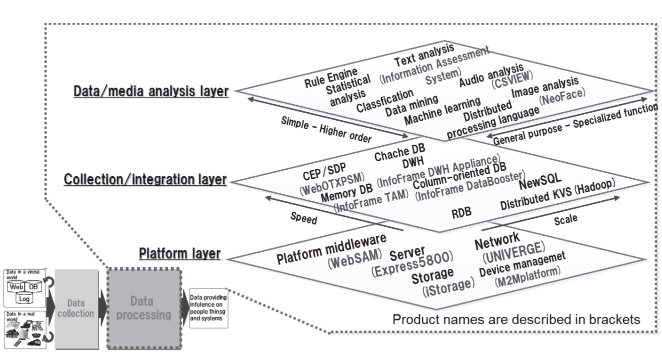

Fig. 1 Big data processing products.

The InfoFrame DataBooster (hereinafter referred to as the DataBooster) is a product belonging to the collection/integration layer, specifically an in-memory type column-oriented (column store) DB. In the next section 2, we discuss the reason why the in-memory type data processing products are becoming important.

2. Background to the debut of In–Memory Data Processing

The need to reduce processing time in order to deal with the increase in the data volume can be dealt with by enhancing the CPU performance. However, in data processing, it is more effective to reduce the time taken for I/O (input/output). This reasoning has led to the data processing time technique by holding data in memory rather than in storage. However, this technique is disadvantageous from the viewpoint of cost because memory is not only faster but is also more expensive than storage.

Nevertheless, since 2010, the significant increase in the memory quantity mounted per server has been reducing the cost per unit data (**Fig. 2**). As of June 2012, there is even a server that can incorporate up to 2 TB of memory (Express5800/A1080).

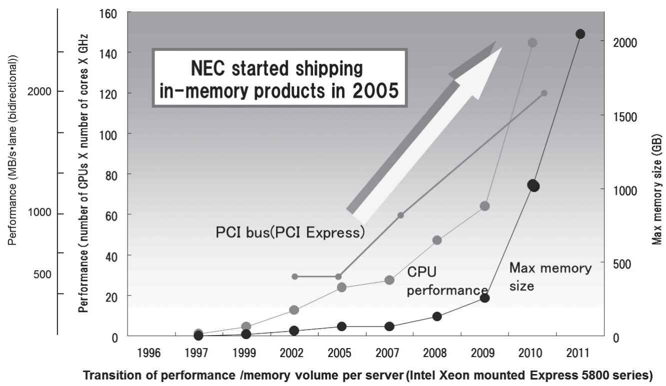

Fig. 2 Memory quantity increase and cost reduction.

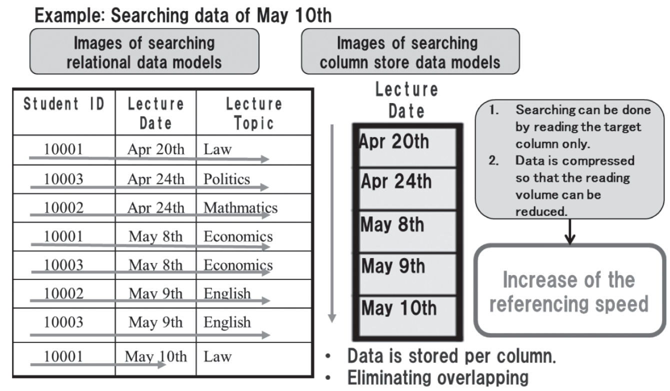

Fig. 3 Differences between "relational" and "column store" databases.

Such a product makes it possible to create an environment that does not use both storage and memory but holds data to be processed exclusively in the memory. This trend has reduced the time of I/O processing of the data processing considerably.

As servers with higher memory capacities are expected to be marketed in the future, the in-memory data processing is expected to attract even more attention.

3. Data Processing Technology with a Different Application Domain

The relational database (RDB) that became available in the 1980's is in charge of all data processing operations including updating such as the OLTP (On-Line Transaction Processing) as well as for aggregation such as via the OLAP (On-Line Analytical Processing). However, the recent leap in the data quantity has made it difficult to respond to current storage needs using the RDB alone. The trend has increased interest in other techniques including the KVS (Key Value Store) suitable for massive updating with a high scale-out potential and MapReduce, a strong big data analysis solution. Among these solutions there is one called the "column store data model" (hereinafter "column store").

The "column store" stores data in a table format like the ordinary relational data model (hereinafter the "relational"). The difference lies in the fact that the "column store" deals with the data of each column collectively while the "relational" stores data per record (row).

This difference gives the "column store" two advantages, one of which is an increase in the referencing speed. In other words the "relational" is required to read all of the records until the target record is reached, while on the other hand, "column store" stores data in columns so that only the column of the target record is read.

The second advantage is a reduction in the stored data quantity. Since the "column store" stores data per column, overlapping often occurs so that compression of the data is easy. In **Fig. 3** , for example the "relational" has to read eight records in order to check the presence of data on the "Lecture Date" of May 10th. Meanwhile, since the "column store" stores data per "Lecture Date" column, it is not necessary to read the "Student ID" or "Lecture Topic" data. Such compression also contributes to an increased referencing speed.

4. Features of the InfoFrame DataBooster

The DataBooster adopts a technique that is obtained by optimizing the "column store" for in-memory use.

As shown in **Fig. 4** , the DataBooster separates the table-format data into a unique structure that is composed of three vector components including the order set "Ordset," value number "VNo" and value list "VL" and deploys VL on the memory in the sorted form, so the consumed memory capacity can be reduced and the data search speed can be increased. In addition, the speed of the entire processing procedure can also be increased by executing batch processing per column instead of by the usual sequential processing per record.

Thanks to the technology specialized in memory access and the large memory capacity, the DataBooster can manifest a very high-speed performance in processing operations

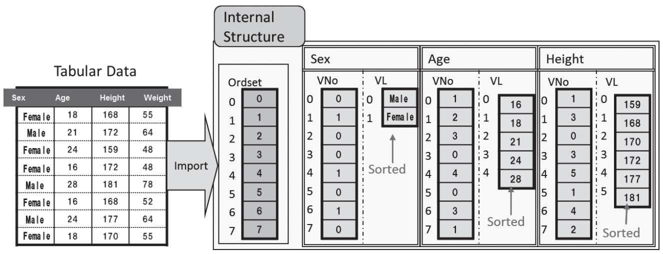

Fig. 4 Internal data structure of DataBooster.

executed in batch mode, such as aggregation and data cleansing.

For instance, if an increase in the data quantity makes it impossible to finish the nighttime batch processing for data mart creation by the next morning, the business of the next day will be affected. In such a case, the high speed of the DataBooster enables completion of the processing in time and also permits allocation of time to other operations than the batch processing such as backup.

5. Development and Implementation of the SQL Interface

5.1 Use of PostgreSQL

The standard API of the DataBooster uses C-language and Java. As each API executes a single function such as sorting, aggregation, search, join or computation, programming becomes easy and highly productive once accustomed.

However, because the positioning of the DataBooster as a product is supplemental to the database and the users of the product often have experience in database systems, we have added an interface for executing SQL statements using the JDBC in DataBooster V4.1 (released in November 2011).

In the development of the SQL interface, the SQL statement processing component employs the PostgreSQL. This is not only used in lexical and syntax analyses but also in the subsequent processing such as in the execution planning and execution components. Therefore, SQL statements that cannot be executed with the DataBooster function alone, such as the functions not found in the DataBooster, are processed using the PostgreSQL functions.

The DataBooster SQL can execute most of the SELECT, INSERT, UPDATE and DELETE statements supported by the PostgreSQL. In addition, it can also execute PREPARE, EXECUTE, EXPLAIN, etc.

Before starting processing (selection, updating, etc.), the processing target table needs to be defined and its data deployed in memory. This is done using the COPY statements of the DataBooster-original specifications that can be imported from an external CSV-format file. Unlike ordinary RDBs, the access using the JDBC (Java DataBase Connectivity) links the DataBooster library to the application so that everything proceeds via the same process.

5.2 Properties and Issues of DataBooster API

As the degree of freedom related to the formulae described in the search conditions and calculations of the DataBooster APIs is low compared to ordinary SQL, processing of a search condition with a high degree of freedom like the SQL is implemented by combining more than one processing operation. For instance, if there is a search condition “WHERE column A = column B + 1,” column X is newly added and column B+1 is stored in this column. Then, “1” is stored if column A and column X are same, and “0” is stored if they are different. Finally, search is executed by assigning the condition of column X = 1 to column X. In this way, we use the DataBooster for processing whenever possible by omitting the lexical and syntax analyses, planning, etc. of the SQL so that the DataBooster can manifest its high-speed function.

The DataBooster has a “column transfer” function to perform JOIN. This function makes it possible to add a column of one of the combined tables to the other table. This function can increase the performance speed of JOIN, but it is difficult to determine the column to be transferred or to judge if it is possible to execute the subsequent processing simply by transferring the column. In the future, we will implement a DataBooster capable of efficiently executing these kinds of processing in order to improve performance and usability.

Addition of the SQL interface has improved the productivity of application developers accustomed to the SQL language. Nevertheless, since the standard APIs of the DataBooster use the C-language and Java, it is recommended to use a C/Java API if it is required to secure a high enough speed or develop an efficient program. The C/Java APIs are capable of processing per function so program design, reuse of interim results, performance estimation, tuning and debugging are easily performed.

6. Usage Images of the InfoFrame DataBooster

In the present section, we describe usages of the technology by introducing two case studies as examples.

Identifying how users access a website that has been published externally by an enterprise is indispensable for improving its products and services. If the enterprise is a popular one, the access log contains a huge amount of data so that a long time is required to collect data, process it in an analyzable form and store it in the analysis database (RDB). As a result, the following two points become important issues.

1) Impossibility of timely referencing of analysis results.
2) Necessity of limitless additional purchases of servers and data processing software licenses for use in parallel processing to support data processing speed increases.

The DataBooster can solve these issues. Its high-speed data processing ability can shorten the time taken for storage in the analysis RDB to a fraction of its previous configuration (from a few hours to about 15 minutes in the case of **Fig. 5**). Therefore, the user can obtain fresher analysis data than before. Assuming that the same data processing is performed, the DataBooster can work with fewer CPU cores than other data processing-capable software such as the ETL (Extract/Transform/Load). Consequently, the number of servers can be smaller assuming that servers with the same number of CPU cores are available. Even when the amount of processing increases, the number of server additions can be smaller than for other data processing-capable software.

Next, let us introduce a case in which the DataBooster is used directly as a data analysis tool.

For example, when it is required to forward direct mails and product catalogues to end users, sending them arbitrarily leads to wastage of costs and time. In addition the corporate image might suffer if information on a product is sent that does not find the specific end user. Therefore, in general, the most

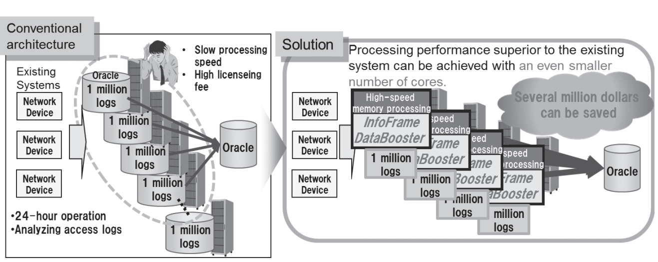

Fig. 5 Case 1: Access log aggregation system.

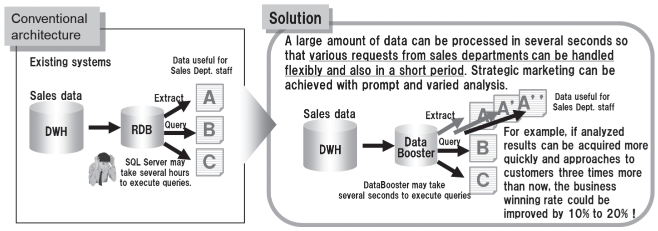

Fig. 6 Case 2: Data analysis system.

effective approach target is confirmed from the corporate sales data before sending direct mails and product catalogues. However, an issue arises here. If the stored sales data is huge, a long time is necessary merely to extract data for the analysis. Furthermore, when the extracted data is analyzed in terms of age, area, sex and articles purchased in the past, the time until the results are obtained will be so long that sometimes the analyses might be abandoned because of the time factor. When the DataBooster is introduced in such a case (**Fig. 6**), the following merits can be obtained.

- Data search and analyses in a few seconds.
- Analyses with more detailed conditions than before.

This allows the enterprise (sales department) to approach specific end users in a timely manner, thereby improving the business winning rate and reducing wastages in cost and time.

7. Conclusion

Now is the time that data processing should be performed by selecting suitably focused software. With its speedy processing capability the DataBooster is expected to be utilized universally as a solution for dealing with the anticipated increase in data processing times.

*Hadoop is a registered trademark or trademark of The Apache Software Foundation.

*Intel and Xeon are trademarks of Intel Corporation in the U.S. and other countries.

*Java is a registered trademark of Oracle Corporation and/or its affiliates in the U.S. and other countries.

*PostgreSQL is a registered trademark or trademark of PostgreSQL Global Development Group.

InfoFrame DataBooster for High-speed Processing of Big Data

Authors' Profiles

KAWABATA Terumasa
Assistant Manager
3rd IT Software Division
IT Software Operations Unit

HAMADA Mitsuyasu
Manager
3rd IT Software Division
IT Software Operations Unit

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

HAKUBA Tomohiro
Assistant 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