BDA 5-7

1

Hello everyone, I am Haiying Che, from Institute of Data Science and knowledge Engineering

School of Computer Science, in Beijing Institute of Technology , in this session we discuss In memory Database.

2

From the perspective of computing architecture, the data processing system is divided into data algorithm layer, computing model layer, computing platform and engine layer, etc.

Computing models are the way that different kinds of big data is processed in different scenarios, We have learned the batch processing, stream computing, **Large-scale concurrent processing model** for structured data, and in this session we discuss In-memory database.

3

There have been multiple advances in the industry in both hardware and software architecture, which makes memory-based computing more feasible today than in the past, as outlined in the diagram below.

In a nutshell, the availability of new classes of hardware with the support of 64bit CPU can now support 2TB on a single device. In addition, the advances in software architecture and solutions toward distributed architecture and cloud make it easier to utilize these new hardware capabilities.

**In-Memory Computing**

In many ways, In-Memory Computing is a close relative of In-Memory Databases. As with many databases, it was designed to enable all the data management aspects that are often expected from traditional databases, such as queries and transactions, with the difference that the data is managed on RAM devices and not disks and thus comes with potentially x1000 better performance and latency according to various benchmarks.

The main **differences** between the traditional in-memory databases and in-memory-based-computing are that In-Memory-Computing is:

1）Designed for distributed and elastic environments

2）Designed for In-Memory data processing

HANA combine the hardware advantages and software advantages to make the in-memory Database possible.

The development of the hardware technology, multi-core architecture, parallel scaling across blades, 64-bit address space, 2 TB in current server board, 25 GB/s data throughput.

All these made the cost-performance ratio rapidly declining, which means the advanced speed processing hardware become cheaper and affordable.

Hana database also includes some advanced software technology and special methods, which include combining row based and column-based store, compression, partitioning, No aggregation tables, insert only and on-the-fly extensibility.

1） row based store is designed to support transactions processing, OLTP and the column based store to support analysis processing, such as OLAP.

2 ）because the memory size cannot be as big as the hard disk, we must try any method to reduce the size of the DB, HANA used a lot of compression methods.

3） in order to increase the DB performance, the database can be partitioned to achieve the parallelization and improve the performance.

4） For eliminating the aggregation tables, normally for data analysis, the data warehouse is built to collect the relevant tables data from multiple data sources and get some aggregations to facilitate the further analysis.

But in HANA， it doesn’t need the aggregation table , the aggregation in memory can be done quickly, no need to do it in advance, on the fly aggregation performance will not be affected, and the aggregation table made the redundant data, which cost the memory space.

5） **And what are the benefits of the insert only approach?**

Well, since the main store is compressed (which is good for reading performance) changes would be expensive. Just imagine you want to make a change in a compressed text file.

You need to uncompress, make the change and compress again. For this reason data is only appended to the uncompressed delta store where change operations are much cheaper.

The merge moves the data from delta to main and thus compresses only from time to time and not with every change.

Setting the deletion flag is based on rowids and works for compressed and uncompressed data.

4

In the column store HANA will only *insert* data. That is whenever you update or delete data HANA will

just *insert*. For performance reasons, the column store is divided in a main and a delta part. New data is going

to the delta and will be merged frequently to the main. Changing statements work as follows:

INSERT:

An INSERT statement will just *insert* a new record in the delta. The merge process will move the record from

delta to main.

DELETE:

A DELETE statement will select the record and mark it as invalid by setting a flag (for main or delta). The mergeprocess will delete the record from memory once there is no open transaction active for it anymore.

UPDATE:

An UPDATE statement will *insert* a new version of the record. The merge process will move the latest version from

delta to main. Old versions will be deleted once there is no open transaction active for them anymore. The update

therefore, consists internally of an *insert*+ delete.

The following picture illustrates this principle schematically. Please note that this picture does not reflect the real

storage in column store. You can see a deleted record in main (id 7) and 2 old versions (marked in grey) of

records (id 8 and 14) where the name has been updated, which lead to an insert of a new version of the records.

5

HANA is an in-memory database. But what exactly do we mean by that? Don't all databases use memory? Yes, but with a *big* difference.

Other databases store data in physical pages on hard drives, and move "copies" of some of the data into memory;

this is called *caching*. When data is missing from the memory cache, it must be fetched from the disk, which takes a very long time, with a minimal latency of 10,000 micro-seconds, holding up the queries and calculations.

Not only that, since the data in memory is a copy of the disk, it is stored in a way to reflect the disk-structure, in the form of linked pages, sectors, etc., which adds additional overhead when fetching the data.

HANA, on the other hand stores the data *directly* in memory, optimized not for how the data is stored on disk, but optimized for efficient access, and as we shall see, highly compressed, using the most efficient memory representations that computer scientists can come up with.

Even in the event of a cache-miss, it only takes 0.1 micro-seconds to fetch the missing data from DRAM. That is 100,000 times faster, and this makes all the difference.

Note that when a transaction changes a piece of the data, HANA logs a record of that change, so that it can be recovered in the event of a crash.

6

There have been two industry trends that enable HANA performance.

The first trend is the astonishing way that memory prices have gone down in the past several years.

What was unimaginable only a few years ago, has become totally affordable today, as you can see from this graph, which shows the price of a Gigabyte of DRAM over time.

Notice how little time it took for a 10X reduction from $100/GB to $10/GB.

Today, a Terabyte of memory costs much less than ten thousand dollars, and computer makers are now putting together servers with four, and soon up to 12 terabytes of memory in a single system.

**7**

The second trend is the way compute power has gone up. A little over a decade ago, we would buy a computer with a single processor.

Then, several years later, Intel introduced dual-core processors, followed, not long ago, with 64-bit processors with eight, dual-threaded, cores.

The first servers for HANA would have as many as four such processors. Now, we are seeing Intel Sandy Bridge processors with 12 cores, and servers with up to 16 processors.

That is not just a 200X performance jump in a decade, this really heralds a challenge for modern software – in-memory now also means that the software must be written to leverage the multi-core architecture of modern processors and multi-processor servers, and even scale out beyond.

8

Disk-centric computing is also a major factor that forces separation of transactional and analytical workloads.

Specialized data warehouses for reporting and analytics required the moving, transformation and pre-processing of transactional data.

This introduces a huge complexity: sometimes an enterprise may hold three different copies of the same data…

Leveraging the power of in-memory computing allows HANA to bring OLTP, transaction processing, and OLAP, data analytics, back together in one database.

HANA reverses the industry trend for separated, specialized analytics databases.

The result is a significant landscape simplification.

9

This image is a simplified architectural picture of SAP HANA.

HANA is certified and delivered on powerful enterprise servers from partners like IBM, HP, Fujitsu, Cisco, Dell and others.

These highly optimized servers scale up to 192 cores, and managed by a Linux operating system from partners like SUSE and Red Hat.

The HANA software is carefully optimized for the hardware it runs on, and takes advantage of the massive parallelism and memory architecture of the modern processing cores on which it runs.

A variety of storage systems can be used for high availability and backup support, including solid state or flash disks.

HANA is fully compatible with other database systems, and supports SQL, JDBC, ODBC and analytical data source interfaces like MDX-Multi Dimensional Expressions.

This means that existing applications can use HANA without any modifications. In addition, HANA supports web-based interfaces which we will discuss later in this course.

10

HANA was designed from the ground up to be massively parallel at every level of operation, one of the main reasons why it performs so well.

The first level is simple: multiple concurrent users and queries are allocated their separate compute resources.

The HANA optimizer then creates an execution plan, that parallelizes the operations within a query.

For instance, I can scan two columns at the same time on different processing cores.

And by partitioning the data, I can parallelize such operations even more, even over multiple servers.

For instance, I can sum table A over as four separate concurrent aggregations, and finally sum the four results together.

And by using processor techniques like hyper-threading, we get even more concurrency.

And all of that is internal to HANA; the applications are not aware of this, except that they enjoy better performance!

11

Here is another very important attribute of HANA: the way that data is stored in memory.

Traditionally, databases store data in rows. One reason is that this is much more convenient when the database is stored on a disk: you simply write a new row to the end of the file.

HANA, on the other hand, stores its tables in columnar order.

This means that it is much more efficient to perform an aggregation (sum of sales, for instance), because all the relevant data is in consecutive memory locations, and most values will be pre-fetched by the processor.

Not only this, but by storing data in columns like this, we can sort and compress it much more efficiently than with conventional databases.

12

In traditional databases, it is often impossible to perform data analytics on the raw, original transaction data.

Therefore, modeling experts are forced to introduce extra tables with pre-calculated summations, average, etc., to increase analytic performance, giving up detail.

For instance, we could sum sales by country in the table above, resulting in a another, smaller table, which is faster to access for generating reports.

These extra tables are called “materialized aggregations”.

The trouble with these aggregations is that they make applications very complicated.

Every transaction also requires the subsequent recalculation and updating of all affected aggregation tables, which may slow operations down significantly.

Another approach, updating the aggregates periodically, means that reports are often not up to date, or don’t match the transaction data.

But HANA can aggregate massive data sets with high performance, over 15 million aggregations per second per core, so there is often really no need for aggregates anymore.

This immediately yields: simpler data models which fewer tables; no loss of detail; Much simpler application logic, with much fewer write-locks and logs; better up-to-datedness

All of this improves the performance dramatically.

This is one of the reasons why customers report hundreds of times of performance improvement.

13

Business intelligence and data analytics often use normalized data models, referred to as star-schema.

This involves holding all the fact-data in one large table, with references to other information in separate tables, called dimensions.

For instance, we could have all our sales data in the central fact table, and details like country, and product in dimension tables.

HANA has a specially optimized execution engine for these types of models, which often speeds up operations on such tables ten times or more, and contributes greatly to the overall performance impact of HANA.

14

Scanning column data is so fast that a lot of indexes are usually not required;

HANA by default only indexes the primary key.

this saves memory as well as index update time’

Dynamic views can be computed on the fly; at over a billion records / sec, aggregations are very fast as well.

And you can see how it is easy to change the table, for instance to add a new column, without complete table re-organization;

each column is managed separately.

And of course, looking only at some of the columns (an operation called projection), is trivial in a column-oriented database.

Compression reduces main-memory access.

This doesn’t only represent a performance bonus due to improved cache behavior, but illustrates another advantage of an in-memory database: the same amount of memory holds a lot more data;

it is not uncommon to see a data compression ratio of 5x.

15

In this session, we learned in memory database HANA, which applied the advanced hardware technologies and developed software technologies.

thank you for your attention, if you have any question, feel free to contact me.