Why do we need a time series database

Software engineering 2020141461170 易小涵

Our world is changing at an incredible rate, reflecting our ability to collect and analyze more and more data.

We now have a lot of apps that are changing our lives. Automated trading algorithms, smart homes, same-day delivery, tracking daily COVID-19 statistics. They all need a special type of data. This is the time series database.

For the first time, the world's interest in time series data has peaked in the most unexpected of ways. The COVID-19 pandemic exercises billions of people around the world as consumers of time series data and demand accurate and timely information on COVID-19 trends from day to day. This has led to the popularity of time series databases. The usage patterns of software developers already reflect the same trend. Over the past two years, the Time Series Database (TSDB) has been the fastest growing database.

To understand why we need a time series database , we first need to know what is time series data. In my opinion, time series data is the data recorded and indexed in chronological order. Devices of all types, such as the Internet of Things and the Industrial Internet, generate huge amounts of temporal data. These will account for more than 90% of the world's data. Time series data sets have three characteristics. One is that the data being written is almost new. Second, data is usually written in chronological order. Third, take time as the main axis. That is to say, sequential data writing is usually "appended" under normal circumstances.

The data state of a time series database changes over time. Suppose you have a sensor that, in a traditional database, collects new data that overwrites previous readings. In the time series database, the sensor creates a new data. So we can compare the changes in the same state that the sensors pick up.

So Time Series Database is a specialized Database for storage and management of Time Series data. It has the characteristics of more write and less read, clear hot and cold, high concurrent write, no transaction requirements, and continuous write of massive data.

Write more read less

In a time series database, there are many more writes than reads. The value of each state is recorded at almost every preset interval. So write is happening all the time. But reading is definitely much less than writing, and most data is read only when necessary. The advantage of this is that data loss rarely occurs. You can find old data in the database.

Hot and cold and clear

Data that is recent has a high probability of being read and the query granularity is fine. On the other hand, the probability of data being read is low and the query granularity is coarse. Internet of things system data storage capacity is large. Data hot and cold distinct, different time of data query needs are different. Data storage also implements data storage of various precision, usually according to the time dimension statistics.

High concurrent write

The way of storing data in a time series database determines that the write operation must be highly concurrent. Because it is based on time, every state needs to be recorded.

Distributed (cluster)

Time series databases should naturally consider features such as distribution and partitioning, distributing storage and queries to different servers to support large-scale data collection and query requests. It should also be scalable and fail-able. With the increase of data volume, the number of servers required will also increase, and the ability to dynamically add or subtract servers is a basic requirement. At the same time, with the increase of servers, a variety of server software or network failure probability will increase, at this time the failure of switching is also very important, not because of the failure of a machine and lead to the whole cluster can not work.

Time series database application scenarios

Common business requirements

1. Obtain the latest status and query the latest data (such as the latest sensor status).

2. Display interval statistics, specify the time range, query statistics, such as average, maximum, minimum, count, etc.

3. Obtain abnormal data and filter abnormal data based on specified conditions

Common Service Scenarios

1.Monitoring software systems: VMS, containers, services, and applications

2.Physical monitoring system: hydrologic monitoring, equipment monitoring in manufacturing plants, national security-related data monitoring, communication monitoring, sensor data, blood glucose meter, blood pressure change, heart rate, etc

3.Asset tracking applications: cars, trucks, physical containers, shipping pallets

4.Financial trading systems: traditional securities, emerging cryptocurrencies

5.Event applications: Track user and customer interaction data

6.Business intelligence tools: Track key metrics and the overall health of the business

7.In the Internet industry, there is also a lot of time series data, such as the behavior trajectory of users visiting websites, log data generated by applications and so on.

The importance of time series database is undeniable. However, the time series database appeared relatively late, and the mature time series database only has a history of two or three years. Time series database still faces several problems that need to be solved.

1. Sequential data writing: how to support hundreds of millions of data writes per second.
2. Sequential data reading: how to support grouping and aggregation of hundreds of millions of data at the second level.
3. Cost-sensitive: The cost of massive data storage. How to store data at a lower cost will become the most important task for sequential database.

In terms of database architecture, I think computational storage separation is a good direction. The underlying storage is like HDFS, where data is written (decompressed, compressed) once, leaving two direct copies to transfer (or do EC), wonderful. The top layer is the sequential database engine and the bottom layer is distributed file/block storage. The obvious benefit of this compaction is that it only takes one compaction (read, write, copy, etc.) to create the same data, eliminating the need for a bad drive or a physical machine to go down. Data expansion/hot and cold separation is also convenient. At the same time, it is relatively friendly to write and read more. The disadvantage is that it is troublesome for multiple compute nodes to write the same data, and distributed locks are required for synchronization. However, in iot, devices are naturally separable, and device data in device area 1 does not need to be joined with monitoring data in device area 2. Therefore, why not write unrelated device data in different instances? This seems to ease the pressure of writing. Sequential databases are really specialized in iot/ monitoring and have huge advantages in sequential data writing/querying/data compression, addressing many user pain points. However, the existing sequential database is still inadequate in storage, either stand-alone or difficult to maintain (opentsdb). There's a lot of room for improvement.

An example of a time series database, TDengine.

TDengine is an open source time series database in China. It is an open source time series database developed by Jeff Tao and his team for scenarios such as Internet of Things, Internet of vehicles and industrial Internet. TDengine is favored by many large manufacturers. For example, TDengine's storage space is reduced to one-tenth of MySQL's in a standalone database monitoring scenario. TDengine helps jd cloud IoT data statistical transformation. TDengine's landing practice in the ideal automotive Internet of Things business scenarios and dozens of cases.

TDengine is designed to allow users to be completely transparent in their processing of historical and real-time data without distinguishing between historical and real-time data. The user only needs to specify the time period in the SQL statement, TDengine automatically determines whether to obtain data from memory, local hard disk, or network storage, so the implementation of the application becomes simple. Data on each device is stored in blocks, and each data block has been pre-aggregated (for example, and, maximum, and minimum values). In this way, various statistical operations on a device in a period of time can be calculated without scanning original data, greatly improving performance. Even if some calculations need to scan the original data, because the data is continuously stored piece by piece, the reading speed is far faster than the general database, and the calculation and analysis speed is also greatly improved. And because of structured storage, after decompression, do not have to do any parsing, read into the memory can be directly calculated, compared with NoSQL database, calculation and analysis speed is also greatly improved.

Summary

I believe that time series database will become the mainstream of database in the future. It has advantages that traditional databases do not have. Many time series databases have emerged in the past decade, but they all have their pros and cons. Thank you, Mr. Jeff. The course you described has expanded my international vision and made me understand the importance of time series database. I also remember TDengine's three main advantages: high-performance, distributed, SQL-enabled sequential databases. I know TDengine has more than just these three advantages. But let's keep these three advantages in mind. This is also very enlightening for future entrepreneurship. Finally, with the development of the Internet, we will need time series databases more and more.