Why do we need a time-series database

First of all, let me introduce time-series database(which will be called TSDB in the following passage) to you. TSDB is a special type of database, which a mostly used for storing time-series data. And the most important reason for TSDB is the coming big data era. If we subdivide every data type, we will find that every data type have some store space to optimize, and what decide that we should use the optimizing is whether the optimizing has increase a new subdivide to become more valuable.

The biggest application scenario of TSDB is monitoring services (Sentry). Taking Sentry as an example, sentry will deploy various script clients on the service server to collect server indicator data (IO indicator, CPU indicator, bandwidth and memory indicator, etc.). Business related data (abnormal method call number, delayed response, the JVM GC data, etc.), database related data (read, write, delay delay, etc.), obviously, these data are related to the time sequence, after the client acquisition will be sent to the sentinel server, the sentinel server will these data for storage, and provide the page to the user query. As shown in the figure below, users can log in to the sentinel system to check the load of a server. The load curve is drawn according to time, with obvious timing characteristics:

In fact, TSDB's potential hasn't exploded, at least not yet. In the foreseeable future 3-5 years, with the arrival of the Internet of Things and industry 4.0, all devices will carry sensors and network, and the timing data collected by sensors will rely heavily on TSDB's real-time analysis ability, storage ability and query statistics ability.

The figure above is a schematic diagram of a smart factory. All the equipment in the factory will carry sensor devices. These sensor devices will collect the basic information of the equipment such as temperature and pressure in real time and send it to the server for real-time analysis, storage and later query and statistics. In addition, all kinds of popular wearable devices can be connected to the Internet in the future, and the heartbeat information, blood flow information and somatosensory information collected on the wearable devices will be transmitted to the server in real time for real-time analysis, storage, query and statistics.

One might ask, why not use a common database?

The truth is you can, and some people already have. However, TSDB is the fastest growing database category today for at least two reasons: size and availability.

Scale: Temporal data accumulates very quickly, and regular databases are designed to handle this scale (at least not in an automated way). Relational databases perform poorly on very large data sets, while NoSQL databases perform better on scale (although relational databases fine-tuned for sequential data can actually perform better, as we showed in benchmark tests compared to InfluxDB, Cassandra, and MongoDB). In contrast, sequential databases (whether relational or noSQL-based) introduce benefits that are only possible if you consider time as a primary consideration. These benefits enable them to provide large-scale performance improvements, including higher throughput and faster large-scale queries, as well as better data compression.

Availability: TSDB also typically includes built-in functions and operations commonly used for temporal data analysis, such as data retention policies, continuous queries, flexible time aggregation, and so on. Even if you're just starting to collect this type of data and don't need to worry about scale right now, these features can still provide a better user experience and make the task of analyzing data easier. Using built-in functions and features to analyze ready-to-use trends in the data layer often reveals unexpected value, no matter how large or small your data set is.

This is why developers are increasingly adopting sequential databases and applying them to a variety of scenarios:

· Monitoring software systems: virtual machines, containers, services, applications, etc

· Monitoring physical system: equipment, machinery, connecting equipment, environment, family, human body, etc

· Asset tracking applications: vehicles, trucks, physical containers, pallets, etc

· Financial trading systems: typical securities, nascent cryptocurrencies, etc

· Event application: tracking user/customer interaction data, etc

· Business intelligence tools: Track key metrics and overall business health, etc

, etc.

Once the sequential data is used to store more information, we still have to choose the sequential database that best fits the business data model, read and write mode, and development technology line. Although NoSQL time series databases have gained popularity as the preferred storage medium over the past decade, more and more developers see the disadvantages of storing sequential data separately from business data (most sequential databases do not provide good support for relational data). In fact, this poor developer experience is one of the main reasons we developed FastData For TSDB. Keeping all data in one system can greatly reduce application development time, as well as facilitate quick critical decisions.

With the rise of self-service business intelligence tools like Tableau, Power BI, and even Excel, it's hard to make timely, business-critical analytics and observations when valuable sequential data is separated from business data. Instead, users find themselves relying on these third-party tools to analyze meaningful information from the data. There are many valid reasons to use these powerful tools, but the ability to quickly query timing data and meaningful metadata information should not be one of them. SQL has been tested for decades and provides a fairly mature and effective way to generate these valuable aggregations and analyses.

The point is, knowing where temporal data is, and where to store it, will have a huge impact on future developments.

Advantages of sequential databases

(1) Scalability: Time series databases specialize in handling more data writes and ultimately remain consistent, even better than distributed storage, which means less anxiety for those who care about data.

(2) Availability: TSDB usually also includes built-in functions and operations commonly used in time series data analysis, such as data retention policies, continuous queries, flexible time aggregation, etc., which can still provide a better user experience and make data analysis tasks easier.

(3) Performance improvement: Time series data accumulation is very fast. The benefits of a time series database over other databases (both relational and non-relational) can only be realized by putting time first. These advantages enable them to provide large-scale performance improvements, including higher throughput and faster large-scale queries, as well as better data compression.

Characteristics of time series Database

Timing database generally has the following characteristics:

High throughput write capability

This is tailored to the feature that sequential services continuously generate massive data. Currently, to achieve high throughput write of the system, two basic technical requirements must be met: horizontal scalability of the system and single-node LSM architecture. It is easy to understand the horizontal scalability of the system, which cannot be supported by a single machine. The system must be clustered, and it is easy to add nodes to expand. In the final analysis, it is not aware of the business when expanding. The LSM architecture is used to ensure high throughput write on a single machine. In the LSM architecture, data writing only needs to be written to the memory and written to the log, so there is no need to write data randomly to the disk. Currently, HBase, Kudu, Druid and other systems that require high write performance adopt this architecture.

Storage tiering /TTL

This is a technical feature customized for the hot and cold nature of temporal data. Data tiering requires that the latest hour data can be stored in memory, the latest day data can be stored on SSDS, and the older data can be stored on cheaper HDDS or TTL expiration.

High compression rate

There are two reasons to provide high compression rate. One is cost saving, which is easy to understand. If you compress 1 TB data to 100 GB, you can reduce the disk cost of 900 GB, which is a great temptation for businesses. Another aspect is the compressed data can be easier to ensure that stored in the memory, such as the recent data is 1 t 3 hours, I now only 100 gb of memory, if without compression, there will be 900 gigabytes of data are forced into the hard disk, it will query overhead is very large, and will use compression this 1 t data into memory, the query performance will be very good.

Efficient time window query ability.

The query requirements of time sequence services are divided into two categories. One is real-time data query, which reflects the status of the current monitored object. Second, it is mainly to query historical data of a certain period. The amount of historical data is very large, so it is necessary to optimize the query for a large number of data in the time window.

Multi-dimensional query capability

Temporal data usually have labels of multiple dimensions to describe a single data, which is the dimension column mentioned above. How to efficiently query according to random dimensions is a problem that must be solved, which usually needs to consider bitmap index or inverted index technology.

High efficiency polymerization capacity

A common requirement for sequential services is aggregated statistical report queries, such as the sentinels need to see the total number of exceptions on an interface in the last day, or the maximum elapsed time for an interface to execute. Such aggregation is actually simple count and Max. The problem is how to efficiently query and aggregate the original data that meets the conditions on the basis of such a large amount of data. You should know that the original value of statistics may not be in memory because of a long time ago, so this may be a very time-consuming operation. Currently, the most mature solution in the industry is to use pre-aggregation, which is to complete basic aggregation operations as data is written in.

Batch delete capability

Sequential services delete expired data in batches.

Transaction capability is not usually required

The sequential database is different from the traditional relational database. The traditional relational database focuses on adding, deleting, modifying and checking and transaction functions, while the sequential database writes massive data and reads and queries mostly within a period of time.