Introduction about InfluxDB

2020141461005 张浩博

Before introducing the influxDB, let us first introduce what is a timing database (TSDB).

A Sequential database is a database that stores time series data (a series of data indexed by time dimension and continuously generated over time, which can be divided into two parts: TAG and Point). Its main characteristics are as follows: 1. Continuous high concurrent write, with little or no data update; 2. 2. Data compression and storage; 3. Query latency is very low. Its main data attributes are: 1. Each data point contains a timestamp for indexing, aggregation, and sampling. The data can also be multidimensional and correlated; 2. Write more and read less, need to support second, millisecond or even nanosecond high frequency write; 3. A summary view of data (for example, downsampled or aggregated views, trendlines) may provide more insight than a single data point. For example, given network unreliability or abnormal sensor readings, we might set alerts when a certain average exceeds a threshold over a period of time, rather than at a single data point; Data analysis usually requires access to it over a period of time. Common TSDB options include InfluxDB, Opentsdb, TDengine, and Prometheus. Also, it focuses on high-performance read, high-performance write, efficient storage and real-time analysis of massive time series data. It is widely used in DevOps monitoring, loT monitoring and real-time analysis. Some basic introductions to influxDB are shown below.

Some basic concepts of InfluxDB.

InfluxDB is an open source distributed temporal, time, and metric database written in the GO language. InfluxDB takes full advantage of the Go language's technical implementation and can be deployed independently without any external dependencies. It is simple to deploy, has a friendly interface and is easy to use. InfluxDB supports multiple protocols. In addition to common protocols such as HTTP and UDP, it is also compatible with communication protocols of components such as OpenTSDB. InfluxDB has rich aggregation computing and sampling capabilities. It also provides flexible data storage policies to set the retention period and number of copies of data. In addition to ensuring data reliability, it deletes expired data in time, releases storage space, and provides flexible continuous query for massive data sampling. At the same time, InfluxDB is not a single time series database. It integrates collection, storage, analysis and visualization capabilities with TICK, which is composed of Telegraf, Chronograf and Kapacitor. All modules cooperate and complement each other, and jointly combine to focus on DevOps monitoring, IoT monitoring, real-time analysis and other scenarios. This also puts Influxdb at the top of the chronological database rankings.

Basic features of Influx DB.

In addition to the characteristics of time series database, such as high read, write and storage rate, InfluxDB also has the following characteristics: 1. Developed by GO language, it has no dependent environment and is easy, quick and convenient to deploy. 2, use unstructured data model, flexible and powerful performance. 3, with native HTTP interface, avoid plug-in configuration and three-party dependence.

Some differences between Influx DB and traditional databases.

Different from the concepts of database, table, and attribute in traditional databases, InfluxDB is structured as Database, measurement, and Point. Database and Measurement correspond to database and table, and the point data structure is composed of time stamp, label and FEILD. A timestamp records the time when data is recorded. Tags record various indexed attributes; the remaining attributes that are not indexed are logged in the field. Furthermore, InfluxDB also has a unique concept, series, which is a collection of the same elements as retention Policy, measurement, and tag sets. Also, the number of series cannot be too large. By default, a single database contains less than one million series. Also, it is important to note that InfluxDB does not need to create tables like a traditional database, and tables for InfluxDB are created automatically by inserting data for the first time.

InfluxDB Retention Policy.

Since InfluxDB is often used to process large amounts of data and does not itself provide deletion operations, how to treat the storage of historical data is a problem. With this issue in mind, Influx DB introduced a retention Policy that allows us to customize the retention time of data. A default storage policy is automatically created when each database is created, and the retention time of data is set to be permanent. The user can then set the retention policy for the data, and each database can also set multiple retention policies. Another important concept associated with retention policy is Shard. There are many shards under each Retention Policy, and each shard stores data for a specified period of time and does not duplicate. Each shard corresponds to an underlying TSM storage engine with an independent cache, WAL, and TSM file.

InfluxDB Storage engine TSM.

TSM is optimized and improved on the basis of LSM. It consists of cache, WAL, TSM File, and Compactor. Cache and WAL: Data is written into the Cache and WAL to prevent data loss. The Cache can be used as the Cache of WAL files in memory.

When the data in the cache reaches the threshold, it is written to a TSM file. Since the data is inserted sequentially into WAL files, the write efficiency is very high. However, if the data is written in a haphazard manner instead of in chronological order, the data will be routed to different shards according to the time. Each SHard has its own WAL file, so that the data is not written in a complete sequence, which may affect performance. There is talk in the official community that it will be optimized to use only one WAL file instead of creating wal files for each shard. TSM file: 2GB for storing data. TSM File has many optimizations for query performance and compression due to its own format design. Compactor: The Compactor component runs continuously in the background, checking every second to see if it has data that needs to be compressed and merged. The main operations are as follows: 1. After the size of the cache data reaches the threshold, take a snapshot of the cache and save it to a TSM file. 2. Merge small TSM files to make each TSM file as large as possible to reduce the number of files. One of TSM's optimized improvements over LSM is the removal of large amounts of data. In LSM, data is deleted by inserting deletion markers, and data is not actually deleted until the file is compressed or merged, so deleting large amounts of data is inefficient for LSM. In TSM, you can set the retention Policy to set the data retention time. When data in a SHard is detected to expire, the shard resources are released and related files are deleted. This makes massive data deletions efficient.

InfluxDB Directory and file structure of the data store.

InfluxDB's data store consists of three directories, meta, wal, and data by default. Meta is used to store metadata of the database. There is a meta. Db file in the meta directory. The wal directory stores pre-write log files, ending with. Wal. The data directory stores the actual stored data files, ending with.tsm.

Some problems with InfluxDB.

1. By default, there is no timeout period for the influxDB query data. Since influxDB is often used to process large amounts of data, CPU utilization is often high. If large queries affect small queries, you can manually set the maximum query time to resolve the problem. 2. When the data in the cache reaches the threshold, the influxDB snapshots are taken and written to the TSM file. As a result, when the current cache is flooded with data, snapshots are written to the TSM file. When the previous snapshot is not fully transferred to the TSM file, the data size in the cache reaches the threshold again. In this case, subsequent write operations become invalid during the processing of influxDB. Users need to wait for the recovery before writing data again.

InfluxDB access and operation optimization.

InfluxDB access is HTTP access in nature. At startup, start influxd.exe on the server first, and then open syringe.exe on the client. Because it is an SQL-like language, except in the data update can only ensure the same tag and timestamp by insert operation, the operation is similar to SQL, but in the process of operation, there are several details need to pay attention to: 1, control the number of series 2, use batch write; 3. Use appropriate time granularity; 4. Try to sort the tags when storing them; 5. Adjust the duration of the shard according to the data; 6. Write irrelevant data to different databases; 7. Control the size of the Tag Key and Tag Value; 8. Storage separation: Map the WAL directory and data directory to different disks to reduce the interaction between read and write operations. To manage the database on the Web, download chronogrf. exe. GO language development needs to rely on the package, pay attention to the use of time zone.