Introduction to TDengine

李鑫芮

2020141461064

After two weeks of time-series database design, I have learned a lot about time-series databases and TDengine, which I will introduce below.

1. A brief introduction to TDengine

TDengine, one of TaOS data's products, has launched its commercial version in August 2018. TDengine does not rely on any open source or third party software, with complete independent intellectual property rights, with high performance, high reliability, scalable, zero management, easy to learn and other technical characteristics. It provides functions such as caching, data subscription, and streaming computing to minimize the complexity of R&D and o&M. TDengine is an efficient platform for storing, querying and analyzing time-series big data, specially designed for the optimization of Internet of Things, Internet of vehicles, industrial Internet, operation and maintenance monitoring, etc. You can use it just like you use MySQL, a relational database, which is simple and convenient.

1. Why do you need TDengine

TDengine summed up 13 characteristics of iot data: data is temporal, must have a timestamp; Data is structured; Data is rarely updated or deleted; the data source is unique; Compared with Internet applications, write more and read less; Users focus on trends over time, not values at a particular point in time; Data has a retention period; Query analysis of data must be based on time periods and geographic regions; In addition to storing queries, various statistical and real-time computing operations are often required; Smooth and predictable flow; Some special calculations such as interpolation are often needed; The volume of data is huge, with more than 10 billion pieces collected in a single day.

After summarizing these characteristics, it can be found that the Internet of Things data is like log data, almost no update operation is possible, so the implementation of transaction processing in the database is completely redundant. Data is temporal, timestamp can be used as the primary key, there is no need for complex index structure;

Data in the Internet of Things is structured and stored by key-value like HBase and Cassandra, which greatly reduces the computing and storage efficiency. Therefore, structured storage should be used.

Internet of Things data is hot or cold based on time, with the data just collected being the hottest, rather than the user clicking on it. Therefore, efficient caching can be achieved with simple FIFO memory management, without the need for Redis at all;

Internet of Things data is a data flow from the perspective of a device. It is not the most natural thing to realize the flow calculation of sliding Windows. How can we use such a complex engine as Spark?

For data partition, simple partition by device partition by time period, it is easy to solve, there is no need for complex partition mechanism;

The flow of iot data is relatively smooth, and iot devices are bound to have caching capabilities, so you can ditch Kafka and implement a simple message queue and data subscription to meet your needs.

Then I found that there was a timing database. I immediately looked at their documentation and code and found that they made use of some characteristics of timing data, but they still did not make full use of it, and it was only positioned as a database.

Behind and learn about the industry has a real-time database and found the real-time database are old products, basically or Windows on research and development, the price is expensive, and there is no standard SQL, extending little, almost no big data analysis ability, have no ability to cope with the increasingly large amount of data and large data analysis requirements, will be eliminated sooner or later.

A message queue with distributed, highly reliable and persistent storage, in which messages are pushed by each mobile phone. Is there a difference between message queues and temporal data in the Internet of Things? Not per se. One is unstructured, one is structured; One is simple in and out, but the other requires analysis and calculation; There are no major differences in the architectural design of the system. Therefore, TDengine quickly positioned its product, that is, the big data platform of the Internet of Things. It should integrate the series of functions such as timing database, cache, message subscription and streaming computing to solve the big data problems of the Internet of Things in a one-stop way, so as to greatly reduce the complexity and cost of system development and maintenance.

After study the data characteristics of the Internet of things, TDengine made two technical innovation point, (1) "a device table" data model to greatly improve the data insert and query efficiency of single equipment (2) give each table a static labels, static labels will be completely separated from the data and dynamic data storage, to solve the problem of multiple tables aggregation query

1. The history of TDengine

In 2017, Taos data began developing TDengine

In August 2018, TDengine's first product was actually delivered to customers, and it took about a year and a half from the beginning to the launch of a working product

On July 12, 2019, Taos Data officially announced that TDengine's kernel (storage and computing engine) and community edition will be 100% open source

In January 2020, Taos data received A pre-A round investment of nearly ten million DOLLARS from GGV and other companies to develop TDengine, an open source big data platform for Internet of Things

In April 2020, Taos data announced the completion of over $10 million series A financing, two rounds of $10 million in three months

1. The advantage of TDengine

4.1. More than 10 times performance improvement: Defined innovative data storage structure, single core can process at least 20,000 requests per second, insert millions of data points, read more than 10 million data points, more than 10 times faster than the existing universal database.

4.2. Cost of hardware or cloud services reduced to 1/5: Computing resources are less than 1/5 of general big data solutions due to superior performance; With column storage and advanced compression algorithms, storage takes up less than 1/10 of a generic database.

4.3. Full stack temporal data processing engine: the database, message queues, caching, computation of flow function integration, application without having to integrate Kafka/Redis/HBase/Spark/HDFS software, greatly reduce the complexity of the application development and maintenance costs. Seamless connection with third party tools: Integration with Telegraf, Grafana, Matlab, AND R without a line of code. MQTT, OPC, Hadoop,Spark, etc. will be supported in the future, and BI tools will be seamlessly connected.

4.4. Powerful analysis function: whether the data is ten years ago or one second ago, the specified time range can be queried. Data can be aggregated on a timeline or across multiple devices. AD hoc queries can be made at any time by Shell, Python, R, and MATLAB.

4.5 .High availability and horizontal scaling: With distributed architecture and consistent algorithms, TDengine ensures high availability and horizontal scaling to support mission-critical applications through multiple replication and clustering features.

4.6. Zero o&M cost, zero learning cost: Easy and quick cluster installation, no need to separate databases and tables, real-time backup. Similar to standard SQL, support RESTful, support Python/Java/C/C++/C#/Go/Node.js, similar to MySQL, zero learning cost.

4.7. Core Open Source: TDengine is open source at its core, with the exception of some ancillary features. Enterprises are no longer tied to databases. This makes the ecosystem stronger, the product more stable, and the developer community more active.

1. Application scenarios of TDengine

TDengine is widely used in many manufacturing processes

Here are some examples from the whitepaper

Public safety: Internet access records, call records, individual tracking, interval screening

Power industry: centralized monitoring of smart electricity meters, power grids and power generation equipment

Communication industry: detailed bill, user behavior, base station/communication equipment monitoring

Financial industry: transaction record, access record, ATM, POS machine monitoring

Means of transportation: train/car/taxi/plane/bicycle real-time monitoring

Traffic industry: real-time road conditions, intersection flow monitoring, bayonet data;

Petroleum and petrochemical: real-time monitoring of oil Wells, transportation pipelines and transportation fleets

Internet: server/application monitoring, user access logs, AD click logs

Logistics industry: tracking monitoring of vehicles and containers

Environmental monitoring: weather, air, hydrology, geological environment monitoring;

Internet of things: elevators, boilers, machinery, water meters, gas meters and other networked equipment

Military industry: data acquisition and storage of various military equipment

Manufacturing: production process control, process data, supply chain data acquisition and analysis

1. The characteristics of TDengine

6.1. High-performance write

TDengine is faster than influxDB in all combinations and writes time increases significantly slower than influxBD in an increasing number of devices

6.2. High-performance query

TDengine is more efficient than influxDB in most cases, and the more timelines in the query criteria, the greater TDengine's query advantage

6.3. Open source distributed cluster solution

Compared with other similar types, TDengine cluster scheme open source points out the massive timeline and data volume, and the cluster architecture is divided into MNode cluster and VNode cluster. A TDengine physical cluster consists of one MNode and N VNodes

6.4 SQL support, easy to use

Has multiple programming language connectors

6.5. Quick and simple installation and deployment

6.6. Easy integration

6.7 Original super table concept

Only one super table needs to be created for a class of collection points, and all subtables of that class of collection points can be created automatically

Tips: All of the above TDengine introductions are derived from the relevant information in the TDengine white paper of Trox Data Company, which was summarized by me to write this article