Introduction about TDengine

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 would use MySQL, a relational database. It is simple and convenient, so it has a high level of ease of use.

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.  [1] It provides caching, data subscription, streaming computing and other functions to minimize the complexity of R&D and operation.

TDengine is established based on its summary of 13 characteristics of Internet of Things data:

1. The data is sequential and must have time stamps;

2. Data is structured;

3. Data is rarely updated or deleted;

4. The data source is unique;

5. Write more and read less than Internet applications;

6. Users focus on trends over time rather than values at a particular point in time;

7. Data has a retention period;

8. Data query and analysis must be based on time periods and geographic regions;

9. In addition to storing queries, various statistical and real-time computing operations are often required;

10. Smooth and predictable flow;

11. Some special calculations such as interpolation are often needed;

12. The volume of data is enormous, with more than 10 billion pieces collected in a single day.

Based on 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 studying the characteristics of Internet of Things data, TDengine made two technological innovations, the data model of "one table for one device", to greatly improve the data insertion and query efficiency of a single device,  each table is labeled with static label, and the static label data is stored separately from the collected dynamic data to solve the problem of multi-table aggregation query

TDengine, which is developed based on the characteristics of big data of the Internet of Things, has huge advantages that traditional databases do not have completely.

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 general database.

2. The cost of hardware or cloud service is reduced to 1/5: Due to super performance, the computing resources are less than 1/5 of the general big data scheme, with column storage and advanced compression algorithms, storage takes up less than 1/10 of a generic database.

3. The whole stack temporal data processing engine: the database, message queues, caching, streaming computing functions such as 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. Powerful analysis function: whether it 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.

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

6. Zero operation and maintenance costs, 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.

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.

With such huge advantages, TDengine has a wide range of applicable scenarios and application space in the current Internet of Things environment.  Also as a foundational software, in principle, it can be used wherever machines, devices and sensors are used to collect data.

TDengine's performance advantage is based on an excellent write storage strategy.

1. Single point write

Although the amount of time sequence data is huge, the data source of each device is unique because the data generation process of different acquisition devices is completely independent.  When a table has only one writer, there is no need to waste resources on locking.  Remember, write operations in traditional relational databases must be protected by locks.  TDengine uses lockless writing to save resources and speed up writing.

2. Continuous storage

Second, for a data collection point, it is a natural data structure for sorting because the data it produces is sequential.  Therefore, subsequent data writes are implemented by orderly append (append), which gives full play to disk performance and further improves data write speed.

3. Super table

TDengine has a table for each device.  But information, such as device address numbers, is not necessarily written to disk.  So they put a bunch of tables together into a super table, and then they just filter the query with address numbers and so on

4. Column compression

TDengine uses column storage.  Because the format of the contents of each column under the column store is similar, it is beneficial for compression and space saving.  Moreover, different compression algorithms are adopted for different types of data.  After the targeted compression, the general compression is done again.  So the data that is finally written to the disk takes up very little space.

In summary, it is not difficult to conclude the characteristics of TDengine:

It is specially designed for Internet of Things data, and realizes the function that each collection point corresponds to one table by taking advantage of the timing characteristics of Internet of Things data.  But it is not suitable for handling general Internet data.

Column storage + compression is used to save hardware costs.  (High compression efficiency: it is highly efficient to use the characteristics of little fluctuation of Internet of Things data, compress after DIF interpolation, and then compress second order.)

Supports high availability by dividing each physical node into multiple virtual data nodes and virtual management nodes.  Virtual data nodes store data, and virtual management nodes manage MetaData.  Virtual data nodes and virtual management nodes are distributed on different physical nodes to achieve high availability of data set applications.

In storage structure, each collection point is used to create an independent table for storage.  In this way, the data of each collection point can be stored continuously and the reading efficiency can be improved.  Since each table has only one data source, lockless write is implemented to improve the write rate.

The concept of super table is introduced for variable aggregation.  The same type of collection device can create a super table.  When creating a super table, you can specify labels for these tables to filter the tables in the database during query. In this way, you can achieve fast multi-table aggregation even if there are many tables in the database.

The installation package is very small and easy to install and use.  Support SQL, syntax is similar to MySQL.