**Introduction of TDengine**

1. **Why TDengine is needed**
2. The data is temporal and must carry a time stamp.
3. the data is structured.
4. the data is rarely subject to update or delete operations.
5. the data source is unique.
6. more writes and fewer readers relative to Internet applications.
7. the user is concerned with trends over time, rather than values at a characteristic point in time.
8. the data is retained for a limited period of time
9. query analysis of the data must be based on time periods and geographic regions.
10. various statistical and real-time computational operations are often required in addition to stored queries.
11. smooth and predictable flows.
12. some special calculations, such as interpolation, are often required.
13. The volume of data is huge, and the data collected in a day can exceed 10 billion.

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

IoT data is structured, like HBase, Cassandra with Key-Value to store, the computational efficiency and storage efficiency is greatly reduced, structured storage should be used to do so.

The hot and cold degree of IoT data is determined by time, and the data just collected is the hottest, not by user clicks. Therefore efficient caching can be well achieved with simple FIFO memory management, and Redis is not needed at all.

IoT data from a device is a data stream, and achieving a sliding window for stream computation that is not a most natural thing to do, which uses such a complex engine as Spark.

For data partitioning, a simple partition by device partition by time period is an easy solution, no need for complex partitioning mechanisms at all.IoT data flow is relatively smooth, and the IoT devices themselves must have caching capabilities, and it is entirely possible to discard these kits of Kafka and implement a simple message queue and data subscription to meet the demand.

Then found that there is a temporal database, immediately look at their documentation and code, found that they make use of some characteristics of temporal data, but still not fully utilized, and only positioned as a database.

Later on, we learned that there are real-time databases in industry, and found that these real-time databases are old products, basically they are still developed on Windows, expensive, and there is no standard SQL, almost no horizontal expansion, almost no big data analysis capability, completely incapable of coping with the growing volume of big data and big data analysis needs, and will be eliminated sooner or later.

A distributed highly reliable, persistent storage message queue, each phone needs to push the message is placed in a queue. Is there a difference between message queues and IoT time-series data? Essentially, no.

One is unstructured and one is structured.

One is simple in and out, but the other requires analysis and computation.

There is no major difference in the architectural design of the two systems.

Therefore, TDengine quickly positioned the product it wanted to make, which was the IoT big data platform, to integrate a series of functions such as temporal database, cache, message subscription, streaming computing, etc. together to solve the IoT big data problem in a one-stop way, so that the complexity and cost of system development and maintenance could be reduced significantly.

After studying the characteristics of IoT data, TDengine has made two technical innovations.

1.The data model of "one device, one table" greatly improves the data insertion and query efficiency of a single device

2.Static tagging of each table and complete separation of static tagging data from the dynamic data collected to solve the multi-table aggregation query problem

1. **TDengine's write storage policy**
2. **Single-point writing**

Although the amount of temporal data is extremely large, the data source of each device is unique because the process of generating data from different collection devices is completely independent. And when a table has only one writer, naturally there is no need to waste resources on the lock mechanism. It is important to know that the write operation of traditional relational database must be protected by locks.

Tdengine uses lock-free way to write, which will save a lot of resources and speed up the writing speed.

1. **Continuous storage**

Secondly, for a data collection point, it is a naturally well-sorted data structure because the data it produces is temporal. Therefore, subsequent writes are achieved by sequential append (append), which can give full play to hard disk performance and further increase data writing speed.

1. **Super table**

TDengine each device corresponds to a table. But information such as the device address number is not necessary to write to disk. So they put a lot of tables together into a super table, and then the query directly with the address number and other information to do the filter on the line

1. **Columnar compression**

TDengine uses columnar storage. Because the content format of each column under the column storage is close, so it is conducive to compression to save space. Moreover, for different types of data will take different compression algorithms. Then after targeted compression, a regular compression is done again. So the final data written to the hard disk will occupy very little space.

1. **TDengine's features**

* Designed for IoT data, it takes advantage of the time-series nature of IoT data to achieve this function of one table for each collection point. However, it is not suitable for handling general-purpose Internet data.
* Columnar storage + compression is used to save hardware cost. (High compression efficiency: high efficiency by taking advantage of the characteristic of IoT data that does not fluctuate much in change, dif interpolation followed by compression, and then second-order compression.)
* Support 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 dataset applications.
* In the storage structure, an independent table is created for each collection point to store. This achieves continuous storage of data from each collection point and improves the reading efficiency. Since there is only one data source for each table, it can realize lock-free writing and improve the writing rate.
* For multivariate aggregation, the concept of super table is introduced. One super table can be created for the same type of collection device. When creating a super table, you can assign tags to such tables and filter the tables in the database by the tags during query, so that even if there are very many tables in the database, fast multi-table aggregation can be achieved.
* The installation package is very small and easy to install and use. Support SQL, syntax is similar to MySQL.

1. **Advantages of TDengine**
2. **More than 10 times performance improvement:** Defined an innovative data storage structure, single core can handle at least 20,000 requests per second, insert millions of data points and read more than 10 million data points, more than 10 times faster than existing general-purpose databases.
3. **Hardware or cloud service cost down to 1/5:** Due to super performance, computing resources are less than 1/5 of general-purpose big data solutions; through columnar storage and advanced compression algorithms, storage occupies less than 1/10 of general-purpose databases.
4. **Full-stack temporal data processing engine**: integrating database, message queue, cache, streaming computing and other functions, applications no longer need to integrate Kafka/Redis/HBase/Spark/HDFS and other software, significantly reducing the complexity of application development and maintenance costs. Seamless connection with third-party tools: Integrate with Telegraf, Grafana, Matlab, R without a single line of code. Later, MQTT, OPC, Hadoop, Spark, etc. will be supported, and BI tools will be seamlessly connected.
5. **Powerful analytics:** Whether the data is ten years old or one second old, specify a time range to query. Data can be aggregated on a timeline or across multiple devices. On-the-fly queries can be performed at any time via Shell, Python, R, MATLAB.
6. **High availability and horizontal scalability:** With distributed architecture and consistency algorithms, TDengine ensures high availability and horizontal scalability to support mission-critical applications through multiple replication and clustering features.
7. **Zero O&M cost and zero learning cost:** Easy and fast cluster installation, no need for separate libraries and tables, and real-time backup. Standard SQL-like, RESTful support, Python/Java/C/C++/C#/Go/Node.js support, similar to MySQL, zero learning cost.
8. **Core open source:** In addition to some auxiliary functions, TDengine's core is open source. No longer will companies be tied to a database. This makes the ecology more robust, the product more stable, and the developer community more active.
9. **TDengine's application scenarios**

TDengine, as a basic software, has a wide range of applications. In principle, it can be used in all places where machines, devices and sensors are used to collect data.

Some typical scenarios are listed as follows:

* **Public security:** Internet records, call records, individual tracking, interval screening
* **Power industry:** smart meters, power grids, centralized monitoring of power generation equipment
* **Communication industry:** phone bill details, user behavior, base station/communication equipment monitoring
* **Financial industry:** transaction records, access records, ATM, POS machine monitoring
* **Travel tools:** real-time monitoring of train/car/taxi/airplane/bicycle
* **Transportation industry:** real-time road conditions, intersection traffic monitoring, chokepoint data.
* **Petroleum and petrochemical:** real-time monitoring of oil wells, transport pipelines, transport fleets
* **Internet:** server/application monitoring, user access logs, ad click logs
* **Logistics industry:** tracking and monitoring of vehicles and containers
* **Environmental monitoring:** monitoring of weather, air, hydrology, geological environment, etc.
* **Internet of things:** elevators, boilers, machinery, water meters, gas meters and other various networking devices
* **Military industry:** data collection and storage of various military equipment
* **Manufacturing industry:** production process control, process data, supply chain data collection and analysis

1. **Summary of TDengine**

TDengine is an efficient platform for storing, querying and analyzing time-series big data, designed for optimization of IoT, Telematics, Industrial Internet, O&M monitoring, etc.. You can use it like a relational database MySQL, easy and convenient.