The introduction of TDengine

1. Challenges in the era of big data

With the popularity of mobile Internet, the sharp decline in the cost of data communication, and the emergence of various low-cost sensing technologies and smart devices, in addition to the traditional mobile phones, computers in real-time data collection, From wristbands, shared bikes, taxis, smart electricity meters, environmental monitoring equipment to elevators, large equipment, industrial production lines and so on, massive real-time data are constantly generated and sent to the cloud.

Close examination revealed that the data generated by all the machines, devices, sensors and trading systems was temporal, and many carried location information. These data have obvious characteristics. 1. 2. Data is structured. 3: Data is rarely updated or deleted; 4: transaction processing without traditional database; 5: Write more than read less than Internet applications; 6: Users pay attention to the trend of a period of time, rather than the value of a particular point in time; 7. Data has a retention period; 8. Data query and analysis must be based on time period and geographical area; 9: In addition to storage queries, but also often need a variety of statistical and real-time computing operations; 10: The amount of data is huge, more than 10 billion pieces of data can be collected in a day.

Seemingly simple things, but because of the huge number of data records, real-time data writing becomes a bottleneck, query analysis is very slow, become a new technical challenge. Traditional relational database or NoSQL database and streaming computing engine do not make full use of the characteristics of these data, and their performance is very limited. They have to rely on cluster technology and invest more computing and storage resources to process, and enterprise operation and maintenance costs rise sharply.

2. characteristics

TDengine is the innovative big data processing product launched by TaOS Data in the face of the rapid growth of the Internet of Things big data market and technical challenges. It does not rely on any third party software, nor does it optimize or package an open source database or streaming computing product. It is independently developed after absorbing the advantages of many traditional relational databases, NoSQL databases, streaming computing engines, message queues and other software, and has its own unique advantages in the processing of big data in sequential space.

• More than 10-fold performance improvement: Defining an innovative data storage structure, a single core can process 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 generic databases.

• 1/5 cost of hardware or cloud services: Computing resources are less than 1/5 of common big data solutions due to superior performance; With column storage and advanced compression algorithms, the storage space is less than 1/10 of a common database

, full stack temporal data processing engine: the database, message queues, caching, streaming computing with functions such as fusion, application without having to integrate Kafka/Redis/HBase/Spark/HDFS software, greatly reduce the complexity of the application development and maintenance costs.

• 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. Temporary queries can be made at any time by Shell,Python, R, and Matlab.

• Seamless connectivity with third party tools: Integration with Telegraf, Grafana, EMQ,Prometheus, Matlab, R, and more without a line of code. OPC, Hadoop, Spark, etc will be supported in the future, and BI tools will be seamlessly connected.

• Zero operation and maintenance cost, zero learning cost: installation and cluster can be completed in a second, no need to separate database and table, real-time backup. Standard SQL, support JDBC, RESTful, support Python/Java/C/C++/Go, similar to MySQL, low learning cost.

TDengine can reduce the overall cost of typical Internet of Things, Internet of vehicles and industrial Internet big data platforms to 1/5 of the existing cost. TDengine can increase system processing power and capacity more than fivefold for the same hardware resources. However, it should be pointed out that the characteristics of time series data of the Internet of Things are fully utilized, which cannot be used to process general data such as web crawler, Microblog, wechat, e-commerce, ERP, CRM and so on.

3. The data model

TDengine still follows the traditional relational database model. Depending on the application scenario, the user needs to create one to multiple libraries and then create multiple tables in each library, with two significant differences.

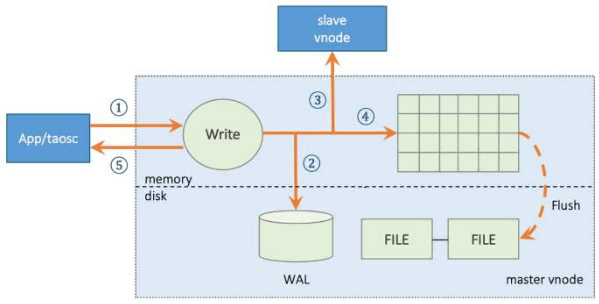
One table for one data collection point: To make full use of the characteristics of timing data, TDengine requires a separate table for each data collection point to store the timing data collected by this collection point. This design has several advantages. 1. It can ensure that the data of a collection point is continuous piece by piece on the storage medium. If the data of a time comes to shape, it greatly reduces the reading and query speed of random operation and comes to shape. 2. Because the data generation process of different equipment is completely independent, and the data source of each equipment is unique, there is only one writer in a table. Thus, the writing speed of former table can fully shape and shape. 3. As a data collection point comes to shape, the writing operation can be implemented in a fully updated way. The writing speed of data comes to shape. Using one table for each data collection point can maximize the performance of single data collection point insertion and query.

Super table: Because one data collection point has one table, the number of tables is huge and difficult to manage. Moreover, applications often need to perform aggregation operations among the collection points, and the aggregation operation becomes complicated. To solve this problem, TDengine introduces the concept of a Super Table (STable for short). A supertable is a collection of data collection points of a particular type. For the same type of data collection point, the structure of the table is exactly the same, but the static attributes (tags) of each table (collection point) are different. In TDengine's design, tables are used to represent a specific data collection point, and supertables are used to represent a set of data collection points of the same type. When creating a table for a specific data collection point, the user uses the super table definition as a template and specifies the label value for the specific data collection point (table). In contrast to traditional relational databases, a table (a data collection point) is labeled statically.

4. TDengine storage structure

In memory, TDengine uses line storage, SkipList indexing, and First In First Out memory management to reduce memory overhead and effectively deal with out-of-order time issues. However, in order to fully take advantage of former shape data, TDengine adopts column storage, and the data of each table is continuously stored in shape, and comes to fully shape compression ratio and data reading speed.

The specific data writing process is shown in the figure:



When data is written, data points are written to WAL logs and then forwarded to other replicas. When the remaining memory space reaches a critical value, the memory data is written to disks. Memory is managed in a first-in, first-out queue to ensure that the latest data is stored in memory.

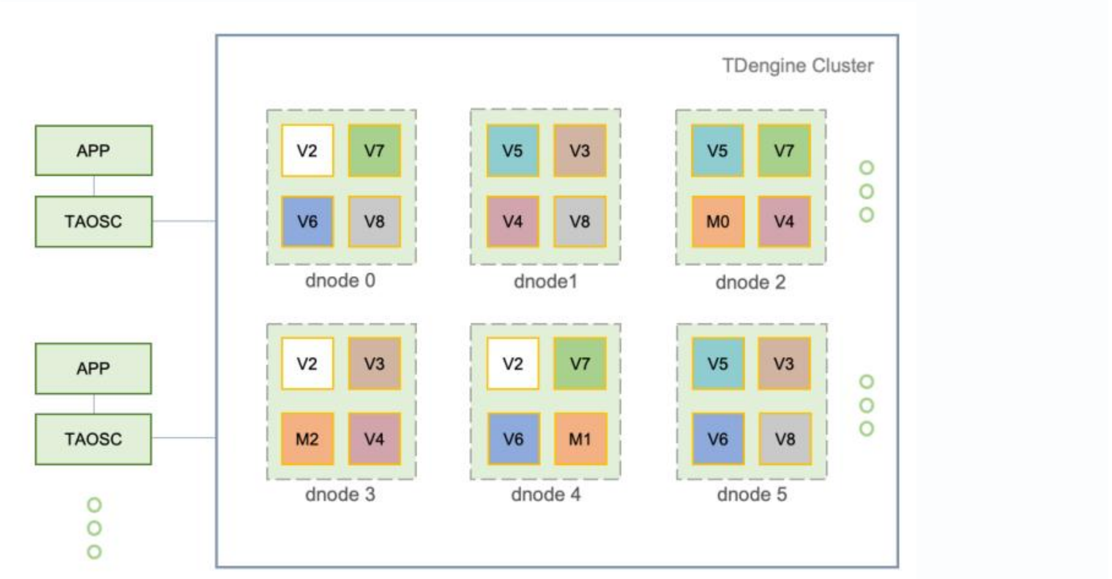
Data is written to the hard disk in the form of adding logs to greatly improve the speed of falling disks. To avoid merge operations, data for each collection point (table) is also stored in blocks. Within a block, data points are stored consecutively in columns, but not consecutively from block to block. TDengine maintains an index for each table, storing information about the offset of each data block in the file, start time, data points, compression algorithm, and so on.

Data partitioning: Each data file only holds data for a fixed period of time (such as a week, configurable), so the data of a table is distributed among multiple data files. When querying, TDengine will figure out which data file to look for will be in, based on a given time period, and then read it. This drastically reduces the number of hard disk operations. The design of multiple data files also facilitates synchronization, recovery, and automatic deletion, and enables data to be stored on different physical media according to the degree of old and new data. For example, the latest data is stored on SSDS, and the oldest data is stored on large-capacity but slow hard disks. In this way, TDengine reduces random reads from hard disks to almost zero, which greatly improves write and query efficiency, making TDengine superior performance even on very inexpensive storage devices.

In TDengine, the data of a table is partitioned by time period, but not stored across nodes, so that individual tables can be inserted, queried, and computed quickly and efficiently. Even if a table (a collection point) generates 100 bytes of data per second, it is only 3 gigabytes of data per year, and when compressed, it is often less than 300 MB, so this processing does not pose a problem.

5. TDengine distributed architecture

TDengine is based on the assumption that the hardware and software systems are unreliable and will fail, and that any single computer is not capable of handling massive data. Therefore, TDengine is designed according to the distributed highly reliable architecture, which is completely decentralized from the first day of development. The overall system structure of TDengine is shown in figure 2. Some basic concepts are introduced below.



Data node: DNode is a running instance of taOSD on a physical machine, virtual machine, or container. A working system must have at least one data node. Dnodes contain zero to multiple logical virtual nodes (VNodes) and zero or at most one logical management node (MNodes).
Virtual data node: Stores specific temporal data. All insertion and query operations for temporal data are carried out on the virtual data node (marked with V in legend). Virtual data nodes on different physical machines can form a virtual data node group (for example, V2 in Dnode0, V2 in DNode3, and V2 in DNode4 form a group). Data of virtual nodes in the virtual node group can be synchronized in real time by asynchronous replication. Finally, data consistency is realized to ensure that a copy of data can be copied on multiple physical machines. Even if a physical

machine breaks down, there are always virtual nodes on other physical machines that can process data requests, thus ensuring high reliability of system operation.

Virtual management node: collects running status of all nodes, balances load on nodes, and manages MetaData, including users, databases, and tables (marked with M in legend). When an application needs to insert or query a table, if it does not know which data node the table is located on, the application connects to the management node to obtain this information. The management of Meta Data also needs to be highly reliable. The system adopts the master-slave mechanism, allowing many to five virtual management nodes to form a virtual management node cluster (for example, M0, M1,M2 in the legend). The creation of the virtual management node cluster is completely automatic without any human intervention, and the application does not need to know which physical machine the virtual management node is running on.

TAOSC: TAOSC is a driver provided by TDengine for applications. TAOSC is responsible for interface interaction between applications and clusters, and is embedded in JDBC, C, Python, and Go connection libraries. Applications interact with the entire cluster through TAOSC rather than directly connecting to data nodes in the cluster. This module is responsible for fetching and caching metadata; Forward insert, query and other requests to the correct data node; The final level of aggregation, sorting, filtering, and so on is required when the results are returned to the application.

6. Other features

- Data fragmentation, horizontal expansion
- High reliability system
- Efficient multi-table aggregation
- Real-time streaming computing
- Convenient installation, deployment, and maintenance