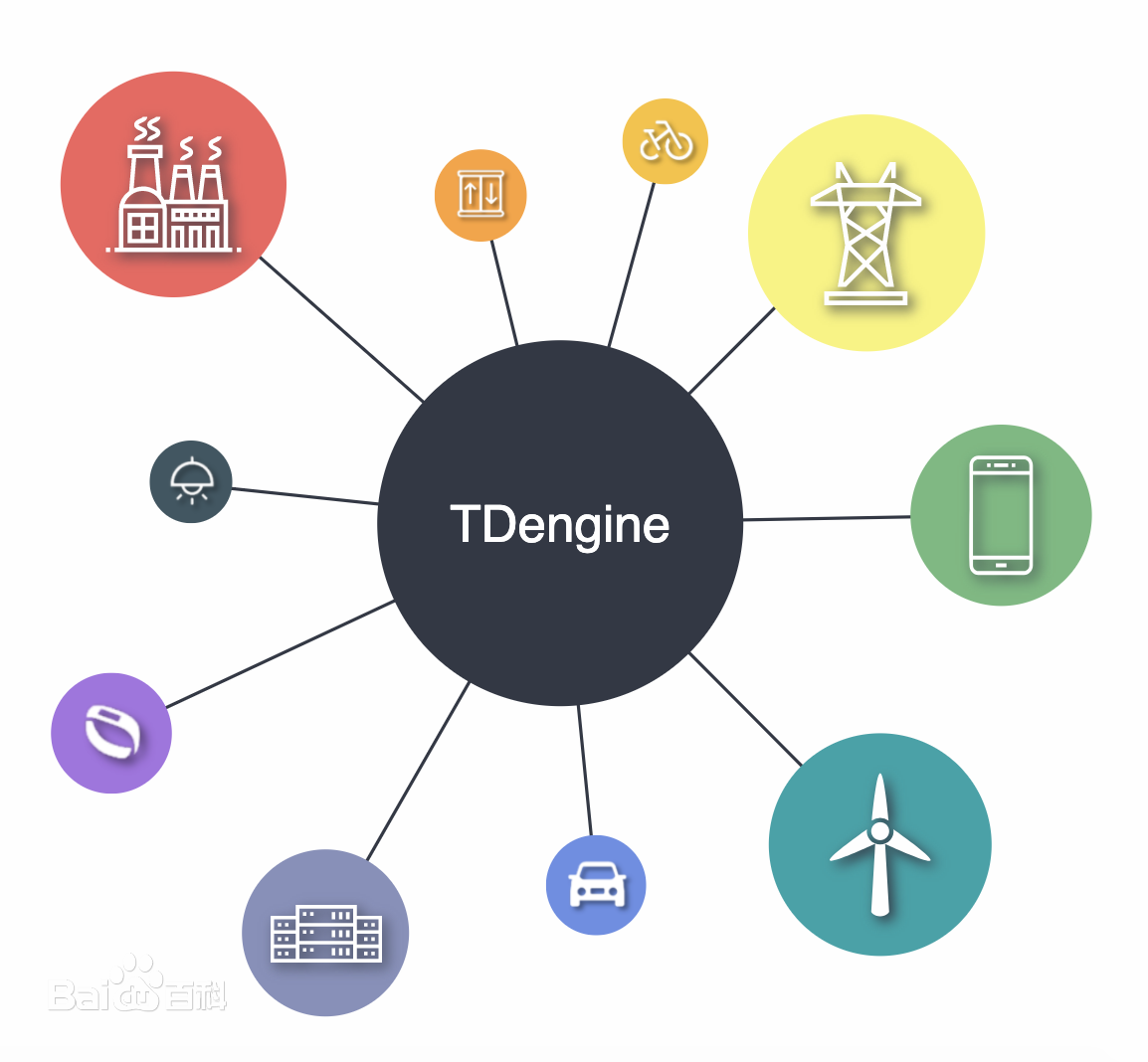
**A brief introduction to Tdengine**

**Writer：Liu Shuai StuNo：2020141461154**

**Abstract：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.**

1. The development course

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, and has completely independent intellectual property rights. TDengine has the technical characteristics of high performance, high reliability, scalability, zero management, easy to learn and so on. Provides functions such as caching, data subscription, and streaming computing to minimize the complexity of R&D and operation and maintenance. 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 received nearly $10 million pre-A round of investment from GGV and others to develop TDengine, an open source Internet of Things big data platform. In April 2020, Taos announced the completion of A round of financing of over $10 million, and completed two rounds of financing of $10 million within three months.



1. Features and Advantages

One of the modules of TDengine is the temporal database. But in addition, in order to reduce the complexity of research and development and the difficulty of system maintenance, TDengine also provides caching, message queuing, subscription, streaming computing and other functions, and provides full-stack technical solutions for the processing of big data of the Internet of things and industrial Internet. It is an efficient and easy to use Internet of things big data platform. Compared with typical big data platforms such as Hadoop, it has the following distinct characteristics:

1. More than 10 times performance improvement: defines the innovative data storage structure, single-core per second can process at least 20,000 requests, insert millions of data points, read out more than 10 million data points, more than 10 times faster than the existing universal database.
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, the storage space is less than 1/10 of a common database.
3. Full stack temporal data processing engine: the database, message queues, caching, flow calculation, and other functions, application without having to integrate Kafka/Redis/HBase/Spark/HDFS software, greatly reduce the complexity of the application development and maintenance costs.
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. Seamless connection with third party tools: Integration with Telegraf, Grafana, EMQ, HiveMQ, 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.
5. 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
6. RESTful,support Python/Java/C/C++/C#/Go/Node.js, similar to MySQL, zero learning cost.

TDengine can significantly reduce the total cost of ownership of typical Internet of Things, Internet of vehicles and industrial Internet big data platforms. However, it should be pointed out that it cannot be used to deal with general data such as web crawler, Microblog, wechat, e-commerce, ERP, CRM, etc., because it makes full use of the characteristics of time series data of the Internet of Things.

1. Data model and overall architecture

Because the collected data is generally structured, TDengine uses a traditional relational database model to manage data in order to reduce the learning barrier. So users need to create libraries first, then tables, before they can insert or query data. TDengine uses structured storage rather than NoSQL's key-value storage.

In order to make full use of the timing and other data characteristics of its data, TDengine requires a separate table to be built for each data collection point (for example, if there are 10 million smart electricity meters, 10 million tables need to be created, and d1001, D1002, D1003 and D1004 in the above table need to be built separately). Used to store time series data collected at this collection point. This design has several advantages:

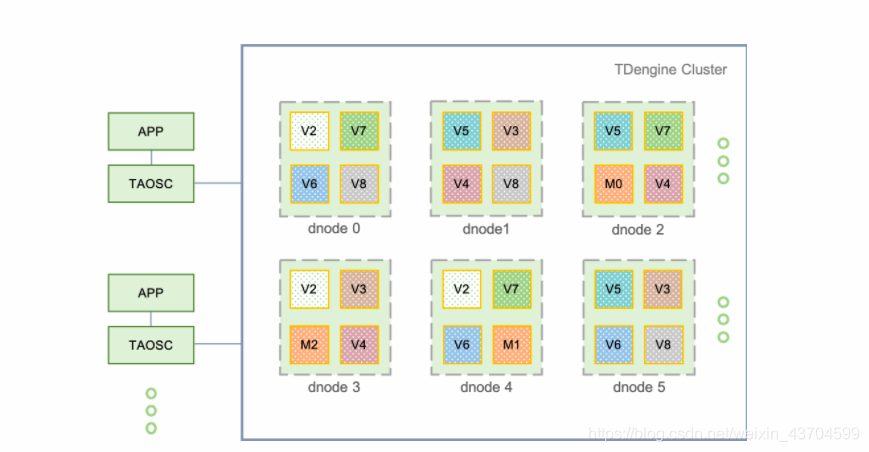
The data of a collection point can be guaranteed to be continuously stored in blocks on the storage medium. If the data is read over a period of time, it can significantly reduce random read operations and increase read and query speed by an order of magnitude. Because the process of data generation by different acquisition devices is completely independent, the data source of each device is unique, and there is only one writer for a table, which can be written in a lockless way and the writing speed can be greatly improved. For a data collection point, the data generated is sequential, so the operation of write can be implemented in the way of appending, which further greatly improves the data write speed.If the traditional way is used to write the data of multiple devices into a table, due to the uncontrollable network delay, the time sequence of the data of different devices arriving at the server is not guaranteed, the write operation is protected by locks, and the data of a device is difficult to ensure continuous storage together. Using a table for each data collection point can maximize the performance of single data collection point insertion and query is optimal.

TDengine recommends using the name of the data collection point (D1001 in the table above) for the table name. Each data collection point may simultaneously collect multiple physical quantities (such as Curent, VOLTAGE, phase in the above table), each physical quantity corresponds to a column in a table, and the data type can be integer, floating point, string, etc. In addition, the first column of the table must be a timestamp, that is, a data type of TIMESTAMP. TDengine will automatically index the collected data according to the timestamp, but will not build any index for the collected physical quantity. Data is stored in column storage.

1. Clusters and basic logical units

TDengine's design is based on the assumption that a single hardware and software system is unreliable, and that no single computer can provide sufficient computing power and storage capacity to handle massive amounts of data. TDengine is therefore designed from day one with a distributed, highly reliable architecture that supports horizontal scaling, so that any hardware failure or software failure on one or more servers does not affect system availability and reliability. At the same time, through node virtualization and automated load balancing technology, TDengine can maximize the utilization of computing and storage resources in heterogeneous clusters to reduce hardware investment.

The logical structure diagram of TDengine distributed architecture is showed below:



A complete TDengine system runs on one to more physical nodes, which logically include data nodes (DNodes), TDengine application drivers (TAOSC), and applications (APPS). One to multiple data nodes exist in the system, and these data nodes form a cluster. Applications interact with TDengine clusters through TAOSC apis. Each logical unit is briefly described below.

Physical Node (PNode): A PNode is a computer that runs independently and has its own computing, storage, and network capabilities. It can be a physical machine, virtual machine, or Docker container installed with an OS. A physical node is identified by its Fully Qualified Domain Name (FQDN).

Data Node (DNode): A DNode is a running instance of the TDengine server code TaOSD on a physical node. 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). The unique identity of a DNode in the system is determined by the End Point (EP) of the instance. EP is the combination of the Fully Qualified Domain Name (FQDN) of the physical node where the DNode resides and the configured network Port number. By configuring different ports, a physical node (a physical machine, virtual machine, or container) can run multiple instances or have multiple data nodes.

Virtual node (VNode): Data nodes are virtualized into multiple virtual nodes (vnode, V2, V3, V4 in the figure) to better support data fragmentation and load balancing and prevent data overheating or skew. Each VNode is a relatively independent unit of work, the basic unit of sequential data storage, with independent running threads, memory space and persistent storage path. A VNode contains a certain number of tables (data collection points). When a new table is created, the system checks whether a new VNode needs to be created. The number of VNodes that can be created on a data node depends on the hardware resources of the physical node where the data node resides. A Vnode belongs to only one DB, but a DB can have multiple VNodes. In addition to storing temporal data, a VNode also stores the schema and label values of the contained tables. A virtual node is uniquely identified in the system by the EP and VGroup ID of the data node to which it belongs. It is created and managed by the management node.