## Introduction to TDengine

### Overview

TDengine is an open source big data framework. It is a big data system designed for the Internet of things (IoT). It is a highly scalable, reliable, and high-performance free big data platform for the processing of large-scale data. TDengine comes with features such as caching, stream computing, message queuing, and many more for reducing development and operation costs. It serves as a relational database and a fast platform for storing, querying, and analyzing time-series data. Because of its storage design, this open source big data platform is at least 10 times faster than other databases. On a single-core machine, TDengine can handle up to 20K queries per second and retrieve up to 10 million data points per second.

TDengine provides a robust data analysis tool that allows companies to do Ad Hoc queries or analyzes using the TDengine shell, Python, R, or Matlab. Developers can swiftly install and run it. Further, there are no other dependencies. Moreover, Standard SQL can be used with connectors including C/C++, Java, Python, Go, RESTful, and Node.JS. This big data framework can be easily integrated with other tools without a single line of code including Telegraf, Grafana, Matlab, R MQTT, OPC, Hadoop, Spark, and many more. TDengine is developed in C, Python, Java and comes with all the documentation regarding development and deployment. The license for this open source big data platform is AGPL-3.0.

### Features

* 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. It supports SQL, syntax is similar to MySQL.

### Why exists

TDengine summarizes 13 characteristics of iot data:

1. The data is sequential and must have a timestamp;
2. Data is structured;
3. Data is rarely updated or deleted;
4. The data source is unique;
5. Compared with Internet applications, write more and read less;
6. Users focus on trends over time, not values at a particular point in time;
7. Data has a retention period;
8. Query analysis of data 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 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; here 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.

### Development

* 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

### Applicable scene

TDengine as a basic software, the application range and its wide, in principle, all the use of machines, equipment, sensors to collect data can be used.

Some typical scenarios are listed below:

* **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

### Advantage

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.
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.
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. **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.
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.
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.
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.