**Introduction about InfluxDB**

**InfluxDB** is an open source time series data developed by InfluxData. It is written in Go and focuses on querying and storing time series data with high performance. InfluxDB is widely used in scenarios such as monitoring data of storage systems and real-time data in the IoT industry.

**1.Design concept**

* Writing the same data multiple times at the same time point is considered to be repeated writing
* It is rare to delete data. Deleting data is basically cleaning up expired data.
* Few updates to existing data and no controversial updates, time series data is always new
* The vast majority of writes are to data with the latest timestamp, and data is added in ascending time order
* The size of the data will be very large and must be able to handle a large number of read and write operations
* Being able to write and query data is more important than strong consistency
* Many time series exist very briefly, so the number of time series is relatively large
* No point is too important

**2.InfluxDB related terms**

measurement: table in the database;

points: a row of data in the table.

Some concepts unique to InfluxDB: Point consists of timestamp (time), data (field) and tags (tags)

**Point** is equivalent to a row of data in a traditional database, as shown in the following table:

|  |  |
| --- | --- |
| Point | Concepts in Traditional Databases |
| time | Each data record time, is the main index in the database (will be automatically generated) |
| fields | Various recorded values ​​(properties without index) are also recorded values: temperature, humidity |
| tags | Various indexed properties: area, elevation |

**Note**

In InfluxDB, the field must exist. Because the field is not indexed. If a field is used as a query condition, all field values ​​that meet the query condition will be scanned, and the performance is not as good as tag. By analogy, fields are equivalent to SQL columns without indexes.

tags are optional, but it is strongly recommended that you use them, because tags are indexed, and tags are equivalent to indexed columns in SQL. The tag value can only be of type string.

There is also an important term: series

All data in the database needs to be represented by charts, series (series) means that all the data in this table can be drawn as several lines on the icon (Note: the number of lines Calculated by the arrangement and combination of tags) .

The sample data is as follows: Where census is measurement, butterflies and honeybees are field keys, location and scientist are tag keys.

name: census

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| time | butterflies | honeybees | location | scientist |
| 2015-08-18T00:00:00Z | 12 | 23 | 1 | langstroth |
| 2015-08-18T00:00:00Z | 1 | 30 | 1 | perpetua |
| 2015-08-18T00:06:00Z | 11 | 28 | 1 | langstroth |
| 2015-08-18T00:06:00Z | 11 | 28 | 2 | langstroth |

There are three tag sets in the example

**Note that the point**

* Tag can only be a string type,
* Field type is unlimited,
* Does not support join,
* Supports continuous query operations (summarized statistics): CONTINUOUS QUERY
* Cooperates with Telegraf service (Telegraf can monitor system CPU, memory, network and other data)
* Cooperate with Grafana service (graphical interface for data presentation, visualize data in influxdb)

**3.Commonly used InfluxQL**

-- View all databases

**show databases;**

-- Use a specific database

**use database\_name;**

-- View all measurements

**show measurements;**

-- Query 10 data

**select \* from measurement\_name limit 10;**

-- The time field in the data displays a nanosecond timestamp by default, which is changed to a readable format

**precision rfc3339;** -- query later, the time is in rfc3339 standard format

-- or you can directly bring this parameter when connecting to the database

**influx -precision rfc3339**

-- View all tag keys in a measurement

**show tag keys**

-- View all field keys in a measurement

**show field keys**

-- view all retention policies in a measurement (there can be multiple, one identified as default)

**show retention policies;**

**4.InfluxDB Operating Modes**

InfluxDB provides two native operating modes: the influx command line tool and the InfluxDB API.

​The influx command line tool is a command line interface similar to the mysql command line tool, which can easily perform management, operation and maintenance, and debugging operations.

​InfluxDB API is a highly programmable, programming language-friendly RESTful API operation interface that supports HTTP and HTTPS protocols.

**4.1influx command line tool**

**Built-in commands**

Similar to the mysql command-line tool, influx also has built-in rich and powerful commands. The following will introduce the commonly used built-in commands in detail.

a. help command The help command outputs the supported built-in commands and their usage help information.

b. auth command The auth command prompts you to enter a user name and password, and then performs authentication when executing the influx command line operation.

When entering the username and password through the auth command, the password will not be displayed.

c. connect command The connect command connects to the InfluxDB server with the specified IP and port without exiting the command line shell. Default connection localhost:8086

d. consistency command

|  |  |
| --- | --- |
| write consistency level | describe |
| any | After any node writes successfully, or the receiving node has written data to the hinted handoff cache queue, it will return success to the client. |
| one | After any node is successfully written, it will immediately return success to the client, excluding the successful write to the hinted handoff cache queue. |
| quorum | When most nodes write successfully, they will return success to the client. This option only makes sense when the number of replicas is greater than 2, otherwise it is equivalent to all. |
| all | Returns success only after all nodes have written successfully. |

Set the write consistency level to all.

e. format command

format command, set the format of server response data: json, csv, column

f. insert command

insert command, write time series data in line protocol format.

g. select command

select command to query InfluxDB data

h. pretty command

pretty command, which supports pretty printing in json format.

i. The precision command

sets the format and precision of any timestamp returned by InfluDB

j. InfluxQL commands

InfluxDB OSS 2.0 supports InfluxQL read-only queries.

**4.2InfluxDB API Mode**

The InfluxDB API is a RESTful API-style interface that returns response data in json format, and supports authentication, JWT tokens, and rich HTTP responses.

When using the HTTP API mode, the InfluxDB responses mainly include the following:

2xx: 204 means no content, and the writing is successful.

200 means that InfluxDB can accept the request but did not complete the request. Usually there is an error message in the body.

4xx: InfluxDB could not parse the request.

5xx: An error has occurred in the system.

**5.Continuous query and retention strategy**

In actual production, continuous query can improve query efficiency and reduce query delay. Reduce storage costs by eliminating expired cold data through retention policies.

**5.1 Continuous query**

Continuous query is a query type in InfluxDB. It will automatically and periodically query real-time data and perform specified operations according to the query rules specified by the user, and then save the query results in a specified table.

By creating continuous queries, users can specify the time interval for InfluxDB to execute continuous queries, the time range of a single query, and query rules. According to the rules specified by the user, InfluxDB will periodically save the original time series data in the past period to the new result table in the manner expected by the user, thereby reducing the time precision of the stored data and greatly reducing the amount of data in the new table. . At the same time, the result of the query is stored in the specified data table, which also facilitates the user to directly query the content of interest, thereby reducing the computational complexity of the query and improving the query efficiency.

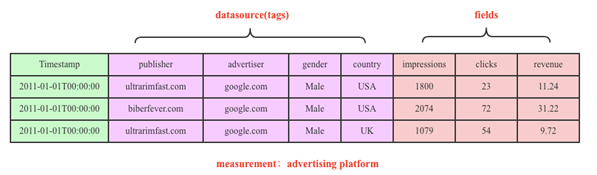
**5.2 Retention Policy**

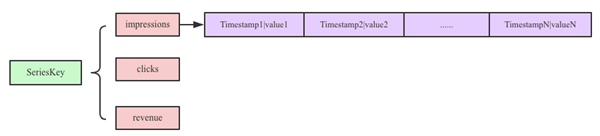
The data retention policy is an important part of InfluxDB. InfluxDB uses the retention policy to determine the length of data retention. InfluxDB will calculate the difference between the local server time and the timestamp of the stored data. If the difference is greater than the retention period set by the retention policy, the expired data will be deleted.

**6.InfluxDB time series data storage model design**

InfluxDB is a professional time series database that only stores time series data, so a lot of optimization work can be done for time series data in the storage of data models.

In order to ensure efficient writing, InfluxDB also adopts the LSM structure. Data is first written to the memory, and when the memory capacity reaches a certain threshold, it is flushed to the file. InfluxDB puts forward a very important concept in the design of time series data model: seriesKey, seriesKey is actually measurement+datasource(tags). Measurement in InfluxDB is like the concept of a table. InfluxDB uses fields to represent indicators, as shown in the following figure:



After the time series data is written into the memory, it is organized according to the seriesKey:

The memory is actually a Map: <SeriesKey+fieldKey, List<Timestamp|Value>>, a SeriesKey+fieldKey in the Map corresponds to a List, and the Timeline data is stored in the List. After the data comes in, it is combined into a SeriesKey according to measurement+datasource(tags), plus fieldKey, and then the Timestamp|Value combination value is written into the timeline data List. After the data in the memory is flushed to the file, the timeline data in the same SeriesKey will also be written into the same Block, that is, the data in a Block belong to the same field under the same data source.

We think this design is to pick out the time series data according to the time line.

**6.1InfluxDB has three major features:**

Time Series: You can use time-related correlation functions (such as max, min, sum, etc.) Metrics: You can perform calculations on large amounts of data in real time

Eevents: It supports arbitrary event data

**6.2Advantages**

Let's take a look at the advantages of this design:

**Benefit 1**:

Tags of the same data source are no longer stored redundantly. The data in a block all share a SeriesKey, just write the SeriesKey into the Trailer part of the block. The storage capacity of time series data is greatly reduced.

**Benefit 2**

The time series and value can be stored separately and independently in the same block, and the independent storage can compress the time column and the value column separately. The storage of time columns in InfluxDB draws on Beringei's compression method, and the use of delta-delta compression greatly improves the compression efficiency. The compression of Value can use the same compression efficiency for different data types.

**Benefit 3**

For data search for a given data source and time range, it can be searched very efficiently. This is the same as OpenTSDB.

InfluxDB internally implements the inverted index mechanism, that is, the mapping relationship between tags and SeriesKey is realized. If the user wants to search according to a certain tag, firstly find the corresponding SeriesKey in the inverted index according to the tag, and then according to the SeriesKey Locate specific timeline data. This storage engine of InfluxDB is called TSM, the full name is Timestamp-Structure Merge Tree, and the basic principle is similar to LSM.