# A Brief Introduction to InfluxDB

Name:林佳琦

Student ID:2019141460510

InfluxDB is a [time series database](https://www.influxdata.com/time-series-database/) designed to handle high write and query loads. It is an integral component of the [TICK stack](https://influxdata.com/time-series-platform/). InfluxDB is meant to be used as a backing store for any use case involving large amounts of timestamped data, including DevOps monitoring, application metrics, IoT sensor data, and real-time analytics.

InfluxDB is developed by InfluxData and is written in Go language, focusing on high-performance read, high-performance write, efficient storage and real-time analysis of massive time-series data. It ranks first in the DB-Engines Ranking time-series database rankings.

Compared to traditional databases, InfluxDB has a certain difference in related concepts. The concept of database in InfluxDB is the same as that in traditional databases. Measurement means the part of the InfluxDB data structure that describes the data stored in the associated fields. Measurements are strings. They are similar to the tables in traditional database. Time represents the time at which the data was recorded. All data stored in InfluxDB has a column called time, which stores UTC timestamps. Time can be added itself when inserting data, and if not, InfluxDB creates it automatically. In InfluxDB, time can almost be considered a primary key. Tags are made up of tag keys and tag values. Both tag keys and tag values are stored as strings and record metadata. Tags in InfluxDB are used to create indexes, improve query performance, and generally store attribute information that identifies the source of the data. The tag set is the different combinations of all the tag key-value pairs. Field in InfluxDB is the key-value pair in an InfluxDB data structure that records metadata and the actual data value. Fields are required in InfluxDB data structures and they are not indexed - queries on field values scan all points that match the specified time range and, as a result, are not performant relative to tags. In InfluxDB, a point represents a single data record, similar to a row in a SQL database table. Each point has a measurement, a tag set, a field key, a field value, and a timestamp and is uniquely identified by its series and timestamp. Users cannot store more than one point with the same timestamp in a series. If users write a point to a series with a timestamp that matches an existing point, the field set becomes a union of the old and new field set, and any ties go to the new field set. In addition, InfluxDB has a unique concept: series, its meaning is a logical grouping of data defined by shared measurement, tag set, and field key.

Unlike relational databases, InfluxQL is the InfluxDB SQL-like query language for working with data in InfluxDB databases. Users can take advantage of InfluxQL to implement many operations, including data exploration, schema exploration, database management, functions, continuous queries, mathematical operation and authentication and authorization. Continuous queries (CQ) are InfluxQL queries that run automatically and periodically on realtime data and store query results in a specified measurement. By creating continuous queries, users can specify the time interval at which InfluxDB executes the continuous query, the time range for a single query, and the query rules. According to the rules specified by the user, InfluxDB will periodically save the original time series data from the past period of time into the new result measurement in the way the user expects, thereby reducing the time accuracy of storing the data and greatly reducing the amount of data in the new measurement. At the same time, the results of the query are saved in the specified measurement, which is also convenient for users to directly query the content they care about, thereby reducing the computational complexity of the query and improving the query efficiency. Data retention policy is an important part of InfluxDB and can be created through InfluxQL, which determines how long data is retained. InfluxDB calculates the difference between the local server time and the timestamp of the stored data, and if the difference is greater than the retention period set by the retention policy, the expired data is deleted.

Flux is a functional data scripting language designed for querying, analyzing, and acting on time series data. Its takes the power of [InfluxQL](https://docs.influxdata.com/influxdb/v1.8/query_language/spec/) and the functionality of [TICKscript](https://docs.influxdata.com/kapacitor/v1.6/tick/introduction/) and combines them into a single, unified syntax. Flux is designed to be usable, readable, flexible, composable, testable, contributable, and shareable. Its syntax is largely inspired by [2018’s most popular scripting language](https://insights.stackoverflow.com/survey/2018#technology), Javascript, and takes a functional approach to data exploration and processing.

The storage engine is the implementation method of how the database management system stores data, how to index the stored data, and how to update and query the data. The InfluxDB storage engine looks very similar to a LSM Tree. It has a write ahead log (WAL) and a collection of read-only data files which are similar in concept to SSTables in an LSM Tree. TSM files contain sorted, compressed series data.

InfluxDB will create a [shard](https://docs.influxdata.com/influxdb/v1.8/concepts/glossary/" \l "shard) for each block of time. For example, if you have a [retention policy](https://docs.influxdata.com/influxdb/v1.8/concepts/glossary/#retention-policy-rp) with an unlimited duration, shards will be created for each 7 day block of time. Each of these shards maps to an underlying storage engine database. Each of these databases has its own [WAL](https://docs.influxdata.com/influxdb/v1.8/concepts/glossary/#wal-write-ahead-log) and TSM files. The storage engine ties a number of components together and provides the external interface for storing and querying series data. It is composed of a number of components that each serve a particular role, including In-Memory Index, WAL, Cache, TSM Files, FileStore, Compactor, Compaction Planner, Compression. Writers and Readers. The four main components are Cache, WAL, TSM Files and Compactor. The WAL is a write-optimized storage format that allows for writes to be durable, but not easily queryable. Writes to the WAL are appended to segments of a fixed size. The Cache is an in-memory representation of the data stored in the WAL. It is queried at runtime and merged with the data stored in TSM files. TSM files store compressed series data in a columnar format. A TSM file is composed of four sections: header, blocks, index, and footer. The Header is a magic number to identify the file type and a version number. Blocks are sequences of pairs of CRC32 checksums and data. The block data is opaque to the file. The CRC32 is used for block level error detection. The length of the blocks is stored in the index. Following the blocks is the index for the blocks in the file. The index is composed of a sequence of index entries ordered lexicographically by key and then by time. The key includes the measurement name, tag set, and one field. The last section is the footer that stores the offset of the start of the index. The Compactor is responsible for converting less optimized Cache and TSM data into more read-optimized formats. It does this by compressing series, removing deleted data, optimizing indices and combining smaller files into larger ones.

Influxdb's data store has three directories, namely meta, wal, and data. The meta directory is used to store some metadata for the database, and there is a meta.db file under the meta directory. The wal directory holds the write-ahead log files, ending in .wal. The data directory holds the actual stored data files, ending in .tsm. The data directory holds the actual stored data files, ending in .tsm.

InfluxData, the company behind InfluxDB, provides a platform known as the TICK stack. This is a comprehensive platform to collect, store, analyse and visualise time series data. The TICK stack comprises Telegraf, InfluxDB, Chronograph and Kapacitor. Telegraf is a data collection agent that captures data from a growing list of sources and translates it into [InfluxDB line protocol format](https://docs.influxdata.com/influxdb/v1.8/write_protocols/line_protocol_reference/) for storage in InfluxDB. Telegraf’s extensible architecture makes it easy to create [plugins](https://docs.influxdata.com/telegraf/v1.23/plugins/) that both pull data (input plugins) and push data (output plugins) to and from different sources and endpoints. InfluxDB stores data for any use case involving large amounts of timestamped data. It provides functionality that allows you to conserve space on your machine by keeping data for a defined length of time, then automatically downsampling or expiring and deleting unneeded data from the system. Chronograf is the user interface for the TICK stack that provides customizable dashboards, data visualizations, and data exploration. It also allows users to view and manage [Kapacitor](https://docs.influxdata.com/platform/" \l "kapacitor) tasks. Kapacitor is a data processing framework that enables you to process and act on data as it is written to InfluxDB. This includes detecting anomalies, creating alerts based on user-defined logic, and running ETL jobs. The InfluxDB 2.0 platform consolidates InfluxDB, Chronograf, and Kapacitor from the InfluxData 1.x platform into a single packaged solution, with added features and flexibility. It includes [InfluxDB OSS 2.0](https://docs.influxdata.com/influxdb/v2.0/get-started/) which is an open source platform solution in a single binary, a hosted cloud solution called [InfluxDB Cloud](https://docs.influxdata.com/influxdb/cloud/get-started/) and [Telegraf](https://docs.influxdata.com/platform/#telegraf) to collect data.

Influx has many advantages. It focuses on DevOps monitoring, IoT monitoring and other scenarios, and is a set of software designed for timing storage, high-performance read and write, real-time operation, and high availability. Designed and developed from scratch, InfluxDB can efficiently collect, store, query, visualize, and perform predefined operations in real time by implementing a highly scalable data receiving and storage engine. It uses sampling and data retention policies to support keeping high-value, high-precision data in memory and low-value data to disk. As a well-designed and well-architected dedicated system, InfluxDB has obvious performance advantages and cost advantages compared with OpenTSDB, MongoDB, Graphite, Cassandra, etc.

As mentioned above, InfluxDB is useful in the age of big data. I believe that with the development of science and technology, time series databases will be more and more widely used in our lives and bring convenience to all of us.