
CERN

Time Series Databases with InfluxDB

Andrei Dumitru
CERN IT Department // Database Services Group
20th of September 2017

Time Series Scenario

Time series data workload assumptions

Normally insert or append data workload

- High ingestion rates in some cases
- Recent timestamps and the data is added in time ascending order
- Rare updates and deletes
- Large deletes to free up disk space

Query operations different from RDBMs

- Individual points are not too important
- Aggregate data and large data sets
- Time-centric filtering and calculations

InfluxDB overview

- Purpose-built for Time-Series
- Open source (MIT License)
- Native HTTP(S) APIs
- SQL-like query language
- Schema-less
- Low hardware sizing to handle most of the use cases
  - Compression
  - Downsampling and data retention capabilities
- High availability
  - Clustering only available in Enterprise version (not free)

InfluxDB Key Concepts

Measurement

- Container for tags, fields and timestamp
- Conceptually similar to a table

```
> show measurements
name: measurements
name
----
cpu
cpu_load
cpu_temp
temperature
>
```

Tag

- The key-value pair that records metadata
- Tags are optional and they are indexed

```
> show tag keys from "cpu_load"
name: cpu_load
tagKey
------
host
region
>
```

Field

- The key-value pair that records metadata and data
- Fields are mandatory and they are not indexed

```
> show field keys from temperature
name: temperature
fieldKey fieldType
-------- ---------
inside   float
outside  float
>
```

Series

- Collection of data that share a retention policy, measurement and tag set

```
> show series from "cpu_load"
key
---
cpu_load,host=serverA,region=Meyrin
cpu_load,host=serverB,region=Wigner
>

> select * from "cpu_load" where time > now() - 1m;
name: cpu_load
time                host    region value
----                ----    ------ -----
1505840898274149470 serverA Meyrin 2.75
1505840902907684665 serverB Wigner 0.57
```

Retention policy

- How long InfluxDB keeps data (DURATION)
- How many copies of those data are stored in the cluster (REPLICATION)

```
show retention policies
name    duration shardGroupDuration replicaN default
----    -------- ------------------ -------- -------
autogen 0s       168h0m0s           1        true
```

Using InfluxDB

InfluxDB Interfaces

HTTP API

Endpoint	Description
/ping	Check the status of your InfluxDB instance and your version of InfluxDB
/query	Query data and manage databases, retention policies, and users
/write	Write data to a pre-existing database

Multiple API client libraries available

- Go, Python, Java, JavaScript, Perl, .Net, etc.

Example

- Use the HTTP API to see the database version

```
curl -i https://dbod-DBNAME.cern.ch:DBPORT/ping

HTTP/1.1 204 No Content
Content-Type: application/json
Request-Id: c4443ff6-9d7d-11e7-82dd-000000000000
X-Influxdb-Version: 1.3.0
Date: Tue, 19 Sep 2017 21:01:54 GMT
```

Command Line Interface

- Interactive shell for the HTTP API

```
influx -ssl \
-host dbod-DBNAME.cern.ch \
-port DBPORT \
-username 'username' \
-password ''

password:
Connected to https://dbod-DBNAME.cern.ch:DBPORT version 1.3.0
InfluxDB shell version: 1.3.0
> set password for "admin" = 'NewPasswordHere'
>
```

Writing data

Syntax

```
measurement[,tag_key1=tag_value1...] field_key=field_value[,field_key2=
    ↪ field_value2] [timestamp]
```

HTTP write

```
curl --user "username:password" -i \
-XPOST 'https://dbod-DBNAME.cern.ch:DBPORT/write?db=mydb' \
--data-binary \
'cpu_load,host=serverA,region=Meyrin value=0.49 1505844394686769520'
```

Datatypes

Measurements, tag keys, tag values and field keys are strings

Field values can be strings, floats, integers or booleans

Timestamps are UNIX timestamps with precision up to nanoseconds (default)

```
measurement[,tag_key1=tag_value1...] field_key=field_value[,field_key2
↪ =field_value2] [timestamp]
```

Querying data

- SQL-like query language

```
SELECT COUNT(value)
FROM cpu_load
WHERE time > now() - 2h
    AND time < now() - 60m
    AND host='serverA'
GROUP BY time(1h);

name: cpu_load
time                  count
----                  -----
2017-09-19T16:00:00Z 17
2017-09-19T17:00:00Z 2
>
```

Querying data

- Using the HTTP API

```
curl --user "username:password" -i \
-G  'https://DBNAME:DBPORT/query?pretty=true' \
--data-urlencode "db=mydb" \
--data-urlencode \
"q=select count(value)
from cpu_load
where time > now() - 2h
and time < now() - 60m
and host='serverA'
group by time(1h)"
```

Querying data

- Using the HTTP API

```
{"results": [{
    "statement_id": 0,
    "series": [{"name": "cpu_load",
           "columns":   ["time"                  ,"count"],
            "values": [ ["2017-09-19T16:00:00Z" ,17],
                        ["2017-09-19T17:00:00Z" , 2]
                      ]
               }
]}]}
```

InfluxQL Functions

Aggregations	Selectors	Transformations	Predictor
COUNT	BOTTOM	CEILING	HOLT_WINTERS
DISTINCT	FIRST	CUMULATIVE_SUM	
INTEGRAL	LAST	DERIVATIVE	
MEAN	MAX	DIFFERENCE	
MEDIAN	MIN	ELAPSED	
MODE	PERCENTILE	FLOOR	
SPREAD	SAMPLE	HISTOGRAM	
STDDEV	TOP	MOVING_AVERAGE	
SUM		NON_NEGATIVE_DERIVATIVE	
		NON_NEGATIVE_DIFFERENCE	

Sampling

Continuous Queries (CQ)

- Query that runs automatically and periodically
- Store query results in a specified measurement.
- Require a function in the select clause and must include a "group by time()"

```
CREATE CONTINUOUS QUERY "mycq" ON "mydb"
BEGIN
SELECT min("temperature")
INTO "min_temperature"
FROM "cooling_system"
GROUP BY time(30m)
END
```

Continuous Queries are not concurrent

Continues Queries are single thread

- All CQ run sequentially in the instance

Issue observed

- CQ takes longer than its interval
- Multiple CQs with different intervals, longer CQs delay the rest of CQs

No fix available now

- Review CQ execution time
- CQ output for real time or low latency uses is not recommended

Concurrent CQs might be added in a future release

- https://github.com/influxdata/influxdb/issues/8545

Continuous Query Statistics (if enabled)

- available in the `cq_query` measurement of the `_internal` monitor database
- `db, cq, durationNS, startTime, endTime, pointsWrittenOK`

Compactions concurrency

What are compactions ?

- recurring processes the run automatically in the database
- migrate data stored in a write-optimized format into a more read-optimized format

There are multiple level compactions

- some of them can take some time depending on the amount of data

Compactions generate a lot of IO activity

- It generates a new copy of the data compacted and then deletes the data not compacted

Performance issues when there are concurrent compactions under high load instances

We cannot control when the compactions are run but...

- New option to limit the concurrent compactions from 1.3 release
- `max-concurrent-compactions`

Conclusions

Most use cases work fine with single instance

Project continues very active and evolving fast

InfluxDB fully integrated in the CERN Database on Demand platform

Currently running 65 InfluxDB instances

Upgrade to version 1.3.5 being prepared

Thank you

Special thanks to Antonio Romero Marin
for the work on the InfluxDB pilot project.
(and for the slides)

home.cern