Preliminary draft 17:58 16 September 2018

16 September 2018
elenastefancova@gmail.com

Evaluation of the TimescaleDB PostgreSQL Time Series extension

ID: 17834

Elena Štefancová
Supervisors: Ignacio Coterillo Coz, Luca Magnoni
CERN, IT-DB-DBR, CH-1211 Geneva, Switzerland

Keywords: InfluxDB, PostgreSQL, PostgREST, Time series

Summary

The purpose of this project was to evaluate TimeScaleDB, an Open Source PostgreSQL extension which extends the functionality of the PostgreSQL database for hosting Time Series data in an optimum way while maintaining 100 % compatibility with the SQL standard and native PostgreSQL features such as replication.

Early results of the solution look very promising compared to alternatives like InfluxDB (currently used at CERN) with a considerable improvement in the use of resources under heavy activity loads while additional benefiting of transparently integrating with currently offered CERN IT solutions.

The technical core of the project would consist on deploying a setup capable of hosting the an activity load equivalent to current CERN Time Series use cases using TimescaleDB to better understand its feature set and resource requirements.

Contents

1 Introduction

Series of data instances indexed in time order are called time series and their properties usually contain components such as cyclical component, irregular component, trend component etc. The main advantages of using specific database approach for this type of data are scalability, usability and trade-offs.

The first step to evaluate TimescaleDB was to properly researched it and also the current solution of InfluxDB, with an emphasis on their difference.

Features of InfluxDB:

- Currently used in CERN (90 instances)
- Non-relational database
- Currently with problems on
  - Performance under medium/heavy activity read loads
  - Excessive resource use (Memory)
- Simple/high performing write and query HTTP APIs
- "SQL-like" query language (may be deprecated in the future...)

Features of proposed TimeScaleDB:

- Open Source PostgreSQL extension
- Database for hosting Time Series
- PostgreSQL is centrally supported at CERN
- *drop_chunks* command can be combined with an external tool for job scheduling, like crontab or systemd (similar functionality as InfluxDB Rentention Policies/CQs)
- All PostgreSQL functionality available:
  - Replication, partitioning, full SQL, etc.

To evaluate the solutions, the following procedure has been chosen:

1. Integration of TimescaleDB in DBOD
2. Sending data to TimescaleDB
   (a) PostgREST
   (b) Flume node
3. Comparision
   (a) Performance/Resource utilization
   (b) Integration with Grafana
      - Alarms

2 Current and proposed solution

At the moment (Figure 1), Flume InfluxDB sends data to InfluxDB database system and then the data are sent to Grafana (a open source software for time series analytics). The original proposed solution was based on was based on Flume using JDBC to connect directly to the TimeSeries Database. Unfortunately, this approach was not possible because of need to implement a middleware between Flume node and PostgreSQL.

The new solution (Figure 2) is suppose to send data from Flume PostgreSQL through PostgREST (RESTful API for Postgres database) to PostgreSQL database system and then to Grafana for analytics.

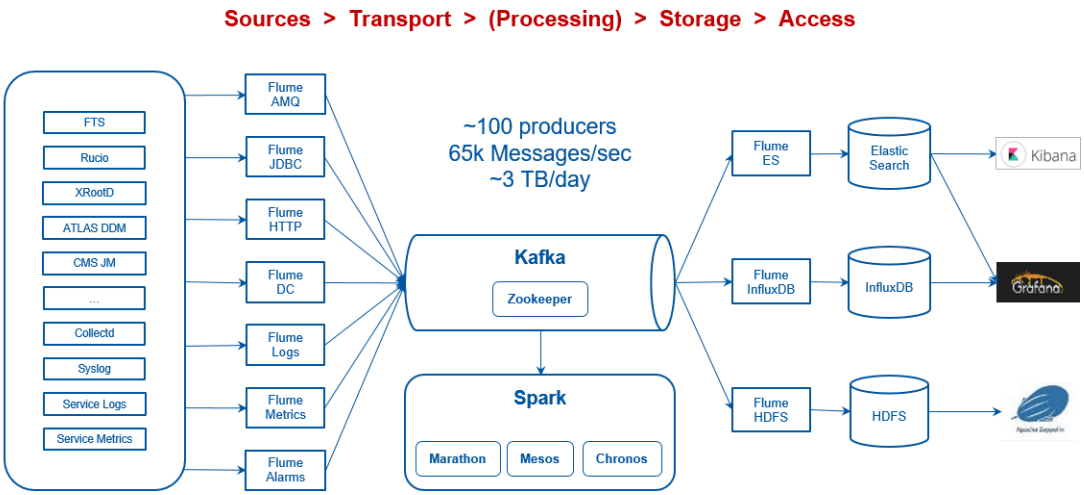

Figure 1: Current MONIT infrastructure.

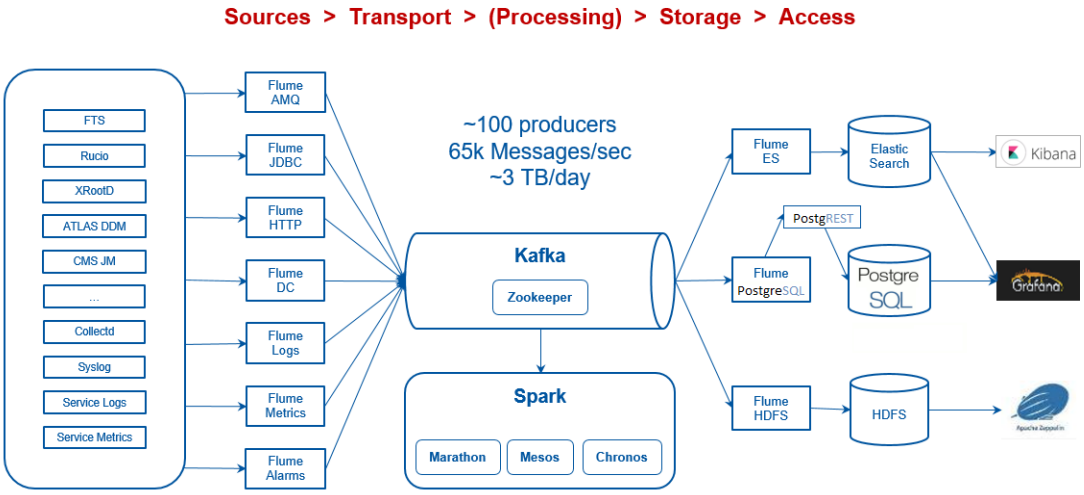

Figure 2: Time Series alternative flow.

3 Challenges

During the process multiple problems occured.

3.1 Configuration

The biggest problems were around installation and configuration of different solutions as PostgreSQL, PostgREST etc. Mostly the problems were caused by difficult authentication, e.g. PosgREST authentication is supported by tokens, and the most troubled was management of the premissions between different roles.

3.2 API calls to PostgreSQL

Another big challenge was that Flume is sending data in JSON format which is not native input format for PostgreSQL. The open source solution PostgREST was found, installed and configured. However, this approach brought another challenges in form of authentication setup (managment of roles in system).

3.3 Format of the JSON

Since the data from Flume node are coming in form of a JSON, our task was also to figure out what is the best praxis for storing it into PosgreSQL, e.g. how to upload the data from a JSON, what datatypes of instance attributes to use. As mentioned, for creating a REST API we used PostgREST and via remote procedure call (RPC).

After conversion from JSON to PosgreSQL firstly we stored everything as a TEXT column and through time chose formats which suit more the character of the data.

The main issue was that the timestamps in the given JSON were in epoch format which is not directly supported by PostgreSQL. The options were either store it as a double (but then we loose all the advantages of DataScale possibilities of time series) or convert it to proper timestamp format. We chose the second one (as a timestamp with time zone), but that required the creating of stored procedures on TimeScaleDB to convert data formats. Actually, more procedures like that were needed because the JSON contains inputs to more than one table.

Example of a given JSON

```
{
    "metadata":
    {
        "availability_zone":"unknown",
        "submitter_environment":"nova_master_3",
        "type":"cpu",
        "toplevel_hostgroup":"cloud_compute",
        "event_timestamp":1533309945000,
        "version":"3.0",
        "timestamp_format":"yyyy-MM-dd",
```

```
        "submitter_hostgroup":"cloud_compute/level2/kvm/
           gva_project_026",
        "type_prefix":"raw",
        "producer":"collectd",
        "_id":"43982369-079f-22c1-25c6-751ea20684d7",
        "timestamp":1533309945494
    },
    "data":
    {
        "time":1533309945.491,
        "interval":60.0,
        "host":"p06636624q04815.cern.ch",
        "plugin":"cpu",
        "plugin_instance":"",
        "type":"percent",
        "type_instance":"interrupt",
        "value":0.0,
        "dstype":"gauge"
    }
}
```


4 Future work

The configuration took more time than expected and for that reason the final stage of the project is still open.

At the moment, dedicated Flume node being set up by MONIT team and than the system is ready to be tested by receiving real data. Also, there are possibilities in tuning of the complete schema.

The last step is to evaluate resource utilization and functionality equivalence (Continuous Queries/Retention Policies, specific math operatores, etc.) after Grafana integration.