Process Historian with Cassandra: 24 hour challenge

***Abstract*—Process historian is a time-series database that stores readings from, for example, SCADA devices, IoT sensors, and the like. Such database should be scalable, durable (should have certain resilience to node failures) and support fast write operations. In this short document we describe our 24 hour challenge in building such database using Cassandra NoSQL, masterless database. We use Python Flask as a IoT facing web server. The web server implements simple REST API for the integration with IoT devices.**

I. Introduction

Process Historian is a must in modern IoT applications: such database is used for storing the time-series readings from various sensors. Process Historian should hold the following properties: (i) it should be durable to node failures, (ii) it should be scalable horizontally, (iii) it should guarantee fast writes to the disk, (iv) it should have clean design.

In what follows we present MVP for such database that uses Cassandara NoSQL database and Python Flask user facing web service. The solution is a result of 24 hours challenge. The source codes for our implementation can be found in [1].

In Figure 1 we show rather abstract architecture of our deployment. Thus in the setup we had 4 nodes deployed in the DigitalOcean cloud: (i) 3 nodes for Cassandra cluster; (ii) MySQL, Nginx, and single REST API server were deployed on single computing node; (iii) we had multiple data generators running on local machine.

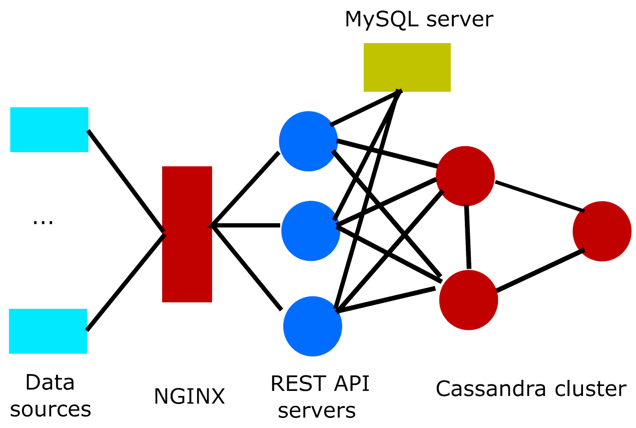

Fig. 1: Process Historian Architecture

The Cassandra row design is depicted in Figure 2. It is basically consists of a wide row (1 day long) and a composite partition key: tag plus date bucket.

The parameters for Cassandra clusters are the following (see Table I)

The remaining parameters where left unchanged and were configured during the release of Cassandra 4.1.9 by the developers.

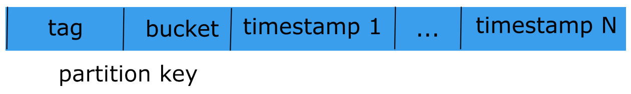

Fig. 2: Cassandra schema design

II. Data collection methodology

To generate the load we have implemented two different clients: (i) data writer and (ii) data fetcher. The writer was generating data for n tags and pushing into the cloud over the Internet in batches of size 1000 samples per batch. The writer was concurrently pushing data for n tags (one thread per tag). Here we were logging such parameters as batch start and end time, batch number and tag (this allowed us calculating the performance later on). Data fetcher was querying the data for m tags for 1 hour period. Here we logged start time, end time, iteration and tag name. Again such information allowed us to calculate the performance for various number of tags.

III. Data processing and basic results

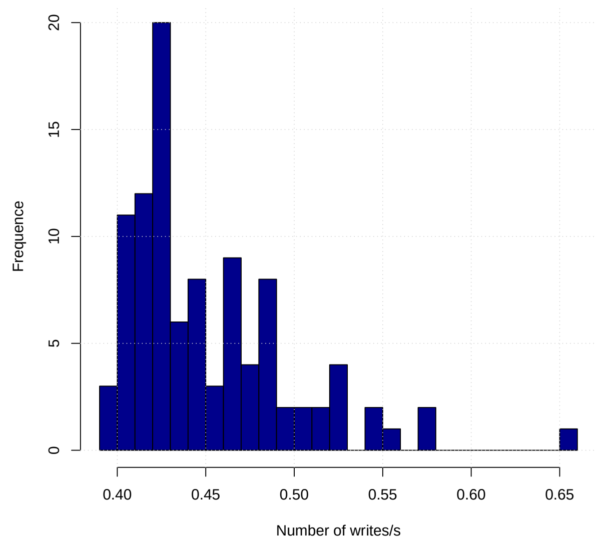

Fig. 3: Number of datapoints written per second

IV. Conclusions

In this short paper we have played a bit with the packets which were captured in a small enterprise. Our primary goal was to analyse the interactions of computers and build the statistics for the traffic which was captured for several hours. Our key findings are the following: (i) major traffic in the

TABLE I: Cassandra parameters

Parameter	**Value**
Number of nodes in the cluster	3
Replication factor	2
Node RAM size	2 GB
TTL	1 year
Number of seed nodes	1
Partitioner	uniform (Murmur3Partitioner)
Read and write consistency	quorum

network is HTTPS and HTTP, (ii) antivirus solutions consume considerable amount of bandwidth, (iii) computers in the network mainly interact with the default gateway and few servers, such as NFS server and mail server.

REFERENCES

[1] D. Kuptsov. Simple Process Historian database. https://github.com/dmitriykuptsov/process-historian-cassandra.