influxdata

Benchmarking InfluxDB vs. OpenTSDB for Time Series Data, Metrics & Management

AN INFLUXDATA TECHNICAL PAPER

External Contributors

Vlasta Hajek Senior Software Engineer, Bonitoo

Tomas Klapka DevOps Engineer, Bonitoo

Ivan Kudibal Engineering Manager, Bonitoo

January 2018 (Revision 2)

Introduction

In this technical paper, we'll compare the performance and features of InfluxDB and OpenTSDB for common time series workloads, specifically looking at the rates of data ingestion, on-disk data compression, and query performance. We'll also look at a feature comparison and the resulting time required to build a complete time series solution with each tool.

Our goal with this benchmarking test was to create a consistent, up-to-date comparison that reflects the latest developments in both InfluxDB and OpenTSDB. Periodically, we'll re-run these benchmarks and update this document with our findings. All of the code for these benchmarks is available on GitHub. Feel free to open up issues or pull requests on that repository or if you have any questions, comments, or suggestions.

This comparison should prove valuable to developers and architects evaluating the suitability of these technologies for their use case, especially those building DevOps Monitoring (Infrastructure Monitoring, Application Monitoring, Cloud Monitoring), IoT Monitoring, and Real-Time Analytics applications.

Why Time Series?

Time series data has historically been associated with applications in finance. However, as developers and businesses move to instrument more in their servers, applications, network and the physical world, time series is becoming the de facto standard for how to think about storing, retrieving, and mining this data for real-time and historical insight. To learn more about why you should insist on using a purpose-built, time series backend versus attempting to retrofit a document, full-text, or RDBMS to satisfy your use case, check out the "Why Time-Series Matters for Metrics, Real-Time and IoT/Sensor Data" technical paper.

What is Time Series Data?

Time series data is nothing more than a sequence of values, typically consisting of successive measurements made from the same source over a time interval. Put another way, if you were to plot your values on a graph, one of your axes would always be time. For example, time series data may be produced by sensors like weather stations or RFIDs, IT infrastructure components like apps, servers, and network switches or by stock trading systems.

Time Series Databases are optimized for the collection, storage, retrieval and processing of time series data; nothing more, nothing less. Compare this to document databases optimized for storing JSON documents, search databases optimized for full-text searches or traditional relational databases optimized for the tabular storage of related data in rows and columns.

Baron Schwartz has outlined some of the typical characteristics of a purpose-built Time Series Database. These include:

About InfluxDB

InfluxDB Version Tested: v1.4.2

InfluxDB is an open-source Time Series Database written in Go. At its core is a custom-built storage engine called the Time-Structured Merge (TSM) Tree, which is optimized for time series data. Controlled by a custom SQL-like query language named InfluxQL, InfluxDB provides out-of-the-box support for mathematical and statistical functions across time ranges and is perfect for custom monitoring and metrics collection, real-time analytics, plus IoT and sensor data workloads.

About OpenTSDB

OpenTSDB Version Tested: v2.3.0

OpenTSDB is a scalable, distributed Time Series Database written in Java and built on top of HBase. It was originally authored by Benoît Sigoure at StumbleUpon beginning in 2010 and open-sourced under LGPL.

As opposed to InfluxDB, OpenTSDB is a Time Series Database that depends on 3rd party standalone products. It relies upon HBase as its data storage layer, so the OpenTSDB Time Series Daemons (TSDs in OpenTSDB parlance) effectively provide the functionality of a query engine with no shared state between instances. This can require a significant amount of additional operational cost and overhead to manage in a production deployment.

- 90+% of the database's workload is a high volume of high-frequency writes
- Writes are typically appends to existing measurements over time
- These writes are typically done in a sequential order, for example: every second or every minute
- If a Time Series Database gets constrained for resources, it is typically because it is I/O bound
- Updates to correct or modify individual values already written are rare
- Deleting data is almost always done across large time ranges (days, months or years), rarely if ever to a specific point
- Queries issued to the database are typically sequential per-series, in some form of sort order with perhaps a time-based operator or function applied
- Issuing queries that perform concurrent reads or reads of multiple series are common

Comparison At-a-Glance

	InfluxDB	OpenTSDB
Description	Database designed for time series, events and metrics data management	Distributed, scalable Time Series Database running on top of HBase
Website	https://influxdata.com/	http://opentsdb.net/
GitHub	https://github.com/influxdata/influxdb	https://github.com/OpenTSDB/opentsdb
Documentation	https://docs.influxdata.com/influxdb/latest/	http://opentsdb.net/manual.html
Initial Release	2013	2010
Latest Release	v1.4.2, November 2017	v2.3.0, December 2016
License	Open Source, MIT	Open Source, LGPL
Language	Go	Java
Operating Systems	Linux, OS X	Linux, OS X, Windows
Data Access APIs	HTTP Line Protocol, JSON, UDP	HTTP, Telnet
Schema	Schema-free	Fixed schema

Overview

Before we dig into the benchmarks themselves, it's important to note that comparing InfluxDB to OpenTSDB directly requires a little explanation of OpenTSDB. Because OpenTSDB requires HBase for its storage layer, it cannot operate as a standalone system, unlike InfluxDB. Generally, running HBase requires a significant amount of operational overhead and itself contains a number of subsystems that are all required. We'll discuss these features in more detail towards the end of this paper.

In building a representative benchmark suite, we identified the most commonly evaluated characteristics for working with time series data. As we'll describe in additional detail below, we looked at performance across three vectors:

1. **Data ingest performance -** measured in values per second (per node)
2. **On-disk storage requirements -** measured in bytes
3. **Mean query response time -** measured in milliseconds

CONCLUSION:

> ***InfluxDB outperformed OpenTSDB in all three categories by a significant margin.***

It's important to point out that working with OpenTSDB requires a significant amount of setup, configuration, and tuning to achieve optimal write and query performance. We consulted all of the available OpenTSDB documentation and made choices to maximize OpenTSDB performance, but these choices will likely vary depending on your use case and dataset. To provide equal comparison to OpenTSDB, the tests were run in a single host deployment. It's an unusual type of deployment, as OpenTSDB is mostly configured in cluster environment, but it provides comparable results.

The Dataset

For this benchmark, we focused on a dataset that models a common DevOps monitoring and metrics use case, where a fleet of servers are periodically reporting system and application metrics at a regular time interval. We sampled 100 values across 9 subsystems (CPU, memory, disk, disk I/O, kernel, network, Redis, PostgreSQL, and Nginx) every 10 seconds. For the key comparisons, we looked at a dataset that represents 100 servers over a 24-hour period, which represents a relatively modest deployment. We also provided some color about how these comparisons scale with a larger dataset, both in duration and number of servers.

Overview of the parameters for the sample dataset

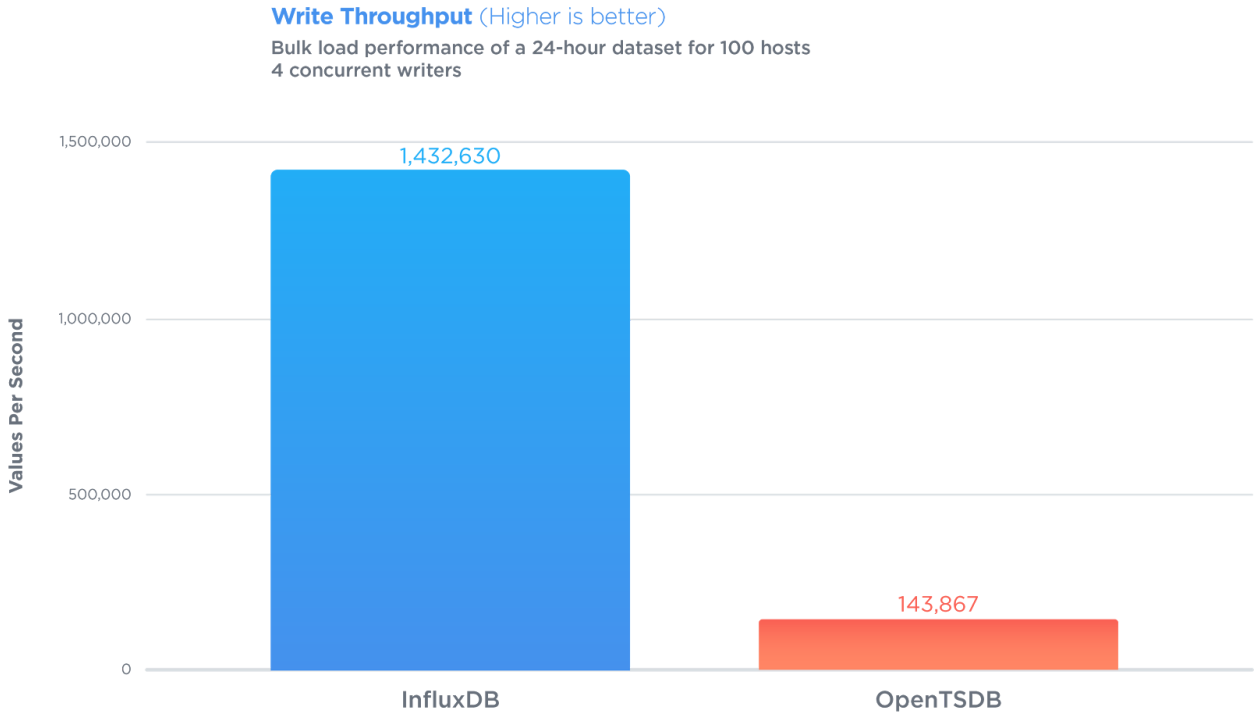
Number of Servers	100
Values measured per Server	100
Measurement Interval	10s

Dataset Duration(s)	24h
Total values in dataset	87,264,000

This is only a subset of the entire benchmark suite, but it's a representative example. At the end of this paper, we will discuss other variables and their impacts on performance. If you're interested in additional detail, you can read more about the testing methodology on GitHub.

Write Performance

To test write performance, we concurrently batch loaded the 24-hour dataset with 4 worker threads (to be able to compare to the other databases tests). We found that the average throughput of OpenTSDB was **143,867 values per second**. The same dataset loaded into InfluxDB at a rate of **1,432,630 values per second** , which corresponds to approximately **9.96x faster ingestion** by InfluxDB.



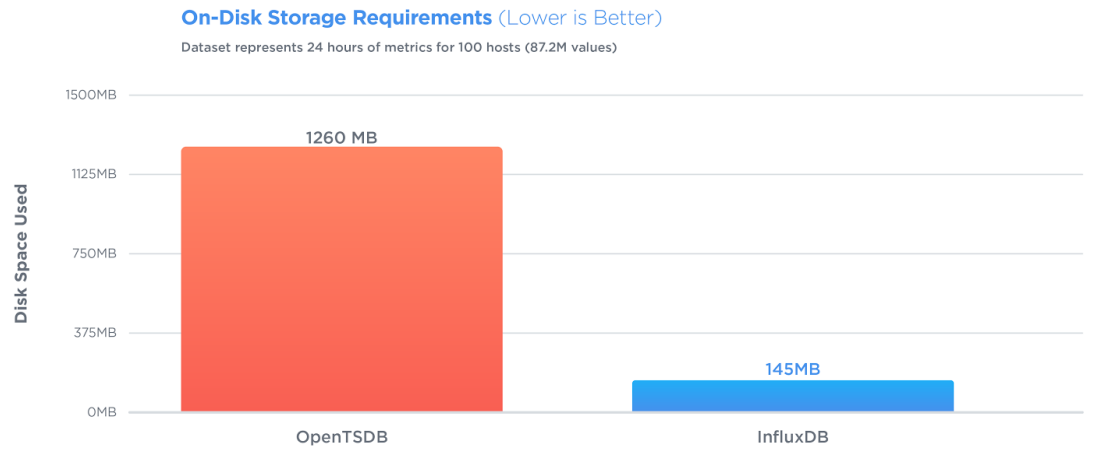
In our testing, we found that write throughput stays relatively consistent across longer datasets (i.e. 48 hours, 72 hours, 96 hours) for both databases. This is what we would expect when looking at systems that are designed to handle time series data.

CONCLUSION:

InfluxDB outperformed OpenTSDB by 9.96x when evaluating data ingestion performance.

On-Disk Storage Requirements

For the same 24-hour dataset outlined above, we looked at the amount of disk space used after writing all values and allowing each database's native compaction process to finish. We found that the dataset required **1.23 GB** for OpenTSDB. The same dataset required only **145 MB** for InfluxDB, corresponding to **8.69x** better compression by InfluxDB. This results in approximately 1.71.4 bytes per value for InfluxDB and 15.1 bytes per value for OpenTSDB.



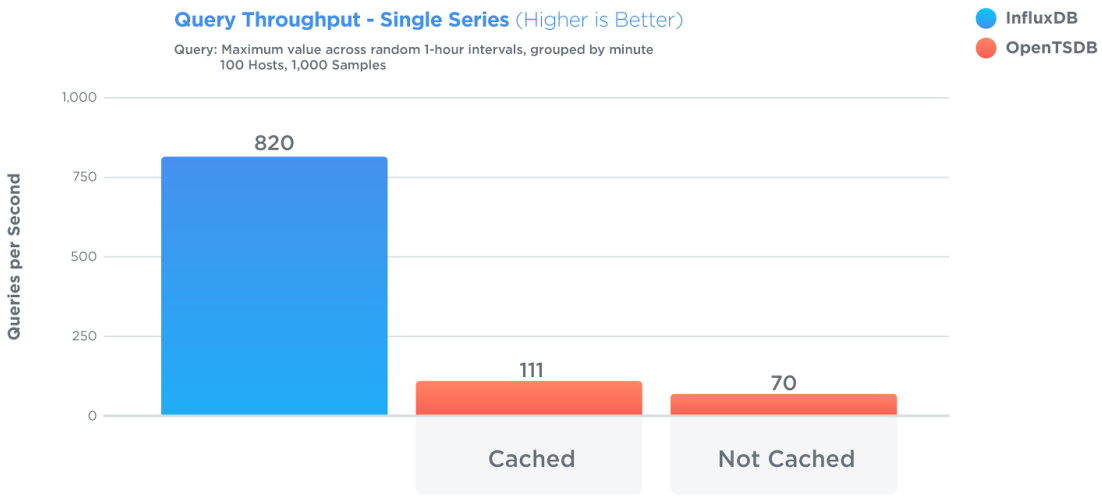
CONCLUSION:

InfluxDB outperformed OpenTSDB by delivering 8.69x better on-disk compression.

Query Performance

To test query performance, we chose a query that aggregates data for a single server over a random 1-hour period of time, grouped into one-minute intervals, potentially representing a single line on a visualization—a common DevOps monitoring and metrics function. Querying an individual time series is common for many IoT use cases as well.

To reduce variability, the query times were averaged over 1,000 runs. With concurrent for 4 worker threads, we found that the mean query response time for OpenTSDB was 9 ms (111.15 queries/sec). The same query took an average of 1.22ms (820.47 queries/sec) on InfluxDB, demonstrating approximately 7.38x higher query performance than OpenTSDB.



To give the best light on OpenTSDB we used response time of cached queries (the second run is faster than the response time of query from fresh data). Average query response time of the first run took 14.33 ms (69.77 queries/sec).

In contrast, InfluxDB response times remain same across the query executions.

When we consider the architecture of OpenTSDB, these results aren't terribly surprising. Because each query to OpenTSDB then has to reach out to the HBase for retrieving data before processing it to provide a response, there will always be an additional latency component, which appears to lead to about a four-fold decrease in query throughput.

CONCLUSION:

> ***InfluxDB outperformed OpenTSDB by providing approximately 7.38x faster query performance.***

Testing Hardware

All of the tests performed were conducted on two dedicated machines running Ubuntu 16.04 LTS, on AWS infrastructure node image c4.4xlarge: Intel Xeon E5-2666 v3 2.9GHz,

16 vCPU, 30GB RAM, 1x EBS Provisioned 6000 IOPS SSD 120GB. For each test, one machine served as the data load and query client, and the other ran only the database server. Both machines were connected within a 1.29 Gbits/sec network.

User Experience Comparison

Overview

OpenTSDB is a time series data store that uses Apache HBase as its storage backend. OpenTSDB is designed to ingest service telemetry data from a compute cluster, and to generate visualizations in a web GUI with gnuplot. It also has a query API.

Running OpenTSDB requires deploying the following components, which are usually located on separate servers:

1. OpenTSDB Time Series Daemons (TSDs),
2. Apache HBase Master nodes,
3. Apache HBase Data nodes,
4. Apache ZooKeeper daemons (for coordinating HBase nodes),
5. Apache Hadoop HDFS, and
6. Apache ZooKeeper daemons (for coordinating Hadoop nodes).

Each of these six components is required to run OpenTSDB in production. Users of OpenTSDB must understand the internals of OpenTSDB and HBase to make effective use of OpenTSDB. (HDFS is fully abstracted by HBase, but system administrators must additionally understand how to run HDFS in production.)

Typical OpenTSDB users are in a situation where another team manages the HBase and HDFS deployment.

InfluxDB is also a special-purpose Time Series Database. However, it is self-contained, and requires no support services to be deployed in production.

Mental Models

Both InfluxDB and OpenTSDB are Time Series Databases, and their mental models for data are similar. In both databases, the notion of a "value at a single point in time" is a first-class concept. In both databases, each value belongs to exactly one measurement, and each value may have tags associated with it.

OpenTSDB supports up to millisecond resolution, and InfluxDB supports nanosecond resolution. This becomes increasingly important as sub-millisecond operations become more common, and additionally allows the freedom to accurately store timestamps for events that may occur in close temporal proximity to one another.

One caveat about OpenTSDB is that it is primarily designed for generating dashboard graphs, not for satisfying arbitrary queries nor for storing data exactly. This has implications for how it should be used (more detail is given in the "Correctness" section).

Correctness

Data Types

OpenTSDB represents all values internally as 32-bit floats. Therefore, exact integer values cannot be represented. The impact of this is application-dependent: when using OpenTSDB as a source of truth, this may cause data quality issues.

In contrast, InfluxDB uses strongly-typed numeric values, and supports both 64-bit floats and 64-bit integers.

HBase Caveats

To be close as possible when comparing to InfluxDB deployed on a single node we decided to configure the OpenTSDB as a Single Node Cluster. Based on Apache HBase 1.2.6 (Pseudo-Distributed Local Install) and Hadoop 2.7.4 Single Node Cluster.

In contrast, InfluxDB is designed to be efficient even when running on one server, and requires no up-front configuration to be performant.

OpenTSDB Caveats

We found a number of surprises when researching OpenTSDB. Any OpenTSDB user or administrator should be aware of the following:

- OpenTSDB requires every value to have at least one tag (http://opentsdb.net/docs/build/html/user_guide/writing.html#naming-schema).
- OpenTSDB performs its own compactions, outside of HBase (which also does compactions).
- OpenTSDB does not use backpressure, causing high-variance write latency: https://groups.google.com/forum/#!topic/opentsdb/73SQbMogyu0

- With OpenTSDB compactions enabled, OpenTSDB servers can stop responding to writes, even under light load.
- Large OpenTSDB queries can cause out-of-memory errors and crash the service.
- The OpenTSDB schema requires HBase tables that must be set up by an HBase administrator.
- OpenTSDB has an endpoint (enabled by default) to manage the server. This means that any user who can connect to OpenTSDB can (accidentally or intentionally) shut down the service. The route is `http://<host>:4242/diediedie`
- There are reports on the OpenTSDB mailing list of OpenTSDB servers entering infinite loops. This is due to an opaque misconfiguration issue.
- OpenTSDB has a variable delay in how long it takes for writes to be reflected in query results (https://groups.google.com/forum/#!topic/opentsdb/S3WLQl7kQ_U).

Configuration

OpenTSDB 2.3.0 requires significant configuration and tweaking to begin loading and querying time series data.

Here is a non-exhaustive list of issues we found while setting up and tuning OpenTSDB, as well as our experiences with HBase and HDFS. It is in approximately chronological order.

- We had to to enable "chunks" for POST requests (`tsd.http.request.enable_chunked` from http://opentsdb.net/docs/build/html/api_http/put.html)
- We had to increase **`tsd.http.request.max_chunk`** for better performance using bigger batches (100 points ~ 29KB in body size, default is 4KB) (
- We had to set **`tsd.core.auto_create_metrics = true`** so that metrics could be created automatically, superseding our shell scripts for metric creation.
- We had to edit both XML and plain text files to configure OpenTSDB.
- We had to set **`tsd.core.uid.random_metrics = true`** to try to relieve metadata write coordination issues, because we saw OpenTSDB errors about conflicting UIDs for the same metric.
- We had to disable OpenTSDB compactions (setting **`tsd.storage.enable_compaction = false`**) to make write performance predictable and to prevent the TSD servers from timing out.
- We had to increase max tag size (**tsd.storage.max_tags**). Default value is 8.

In contrast, InfluxDB requires minimal configuration and is ready out-of-the-box to performantly store and serve time series data.

Documentation

OpenTSDB's documentation is excellent at explaining the underlying data model and the way OpenTSDB and HBase interact. The documentation is much sparser at explaining how to productionize an OpenTSDB deployment. Ostensibly, this is because each OpenTSDB deployment requires tuning HBase, a process that is application-dependent.

User Experience Conclusion

OpenTSDB and InfluxDB are both Time Series Databases. However, OpenTSDB requires multiple orders of magnitude, more administration overhead to run in production, and requires many application-dependent tweaks before it is performant. Even then, it was not entirely correct when running ad-hoc queries over our data. InfluxDB is simpler to set up, easier to use effectively, and returns exactly correct query results.

Summary

In the course of this benchmarking paper we looked at the performance of InfluxDB and OpenTSDB performance across three vectors:

- Data ingest performance - measured in values per second per node
- On-disk storage requirements - measured in Gigabytes
- Mean query response time - measured in milliseconds

The benchmarking tests and resulting data demonstrated that InfluxDB outperformed OpenTSDB across all three tests by a significant margin. Specifically:

- InfluxDB outperformed OpenTSDB by 9.96x for data ingestion
- InfluxDB outperformed OpenTSDB by delivering 8.69x better compression
- InfluxDB outperformed OpenTSDB by up to 7.38x when measuring query performance

We discussed this in depth above, but it’s important to remember that in order to get started with OpenTSDB, a significant amount of up-front work is required to get a functional HBase cluster running, which then requires even more configuration and tuning to achieve optimal performance. InfluxDB, on the other hand, is ready to use for time series workloads out of the box with no additional configuration.

In conclusion, we highly encourage developers and architects to run these benchmarks themselves to independently verify the results on their hardware and datasets of choice. However, for those looking for a valid starting point on which technology will give better time series data ingestion, compression and query performance “out-of-the-box”, InfluxDB is the clear winner across all these dimensions, especially when the datasets become larger and the system runs over a longer period of time.

What’s Next?

InfluxDB Documentation, Downloads & Guides

- 1.4.2 Download
- 1.4 Installation Guide
- 1.4 Getting Started
- 1.4 Schema Design
- 1.4 Line Protocol Reference
- 1.4 Key Concepts

Benchmarking Resources

- Benchmarking Code, Methodology and Documentation on GitHub

Have Questions or Need Help?I

- Community
- Technical Support
- Virtual Training