AN INFLUXDATA TECHNICAL PAPER

Intro to the InfluxDB 3.0 Storage Engine

Anais Dotis-Georgiou
Lead Developer Advocate, InfluxData

MAY 2023

The never-ending quest to support large time series

One of the long-standing requests for InfluxDB is support for many types of time series data. What this means, more specifically, is support for high cardinality in the number of unique time series the database stores. Currently, customers with tens of millions of time series are looking to expand to hundreds of millions and even billions of unique time series.

The last iteration of the InfluxDB engine (TSI), focused on addressing ephemeral time series. We saw this often with use cases that tracked per-process or per-container metrics by putting their identifiers in tags. Ephemeral data, or data lasting for a brief period, became the norm as organizations deployed more IoT sensors and spun up containers with sophisticated orchestrators, such as Kubernetes.

Now, the focus of our users has started to shift to bringing in as much raw data as possible and then deriving their own insights from that raw data. The new storage engine represents the next phase of InfluxDB and its goal to support near-unlimited cardinality. We bring metrics, raw events, and tracing time series data into a single database core, allowing users to create time series on the fly from raw, high-precision event data.

What is the new InfluxDB Engine?

One datastore for all time series data (metrics, events, traces)

Users can write any event data with near infinite cardinality and slice-and-dice data on any dimension without sacrificing performance. This opens up use cases that rely on any combination of event, tracing, observability, and other ephemeral, extremely high cardinality data.

Unified query engine

The query engine underpinning InfluxDB 3.0 does not just support the ingestion of high cardinality data. We optimized InfluxDB 3.0 to query both “hot” data from in-memory cache and “cold” data from cloud object stores. Furthermore, queries that touch multiple time series are orders of magnitude faster in InfluxDB 3.0 than in previous versions of InfluxDB. Querying across ten series or 1 million series yields the same performance, making analytics across high-cardinality data possible.

Understands SQL and InfluxQL query languages

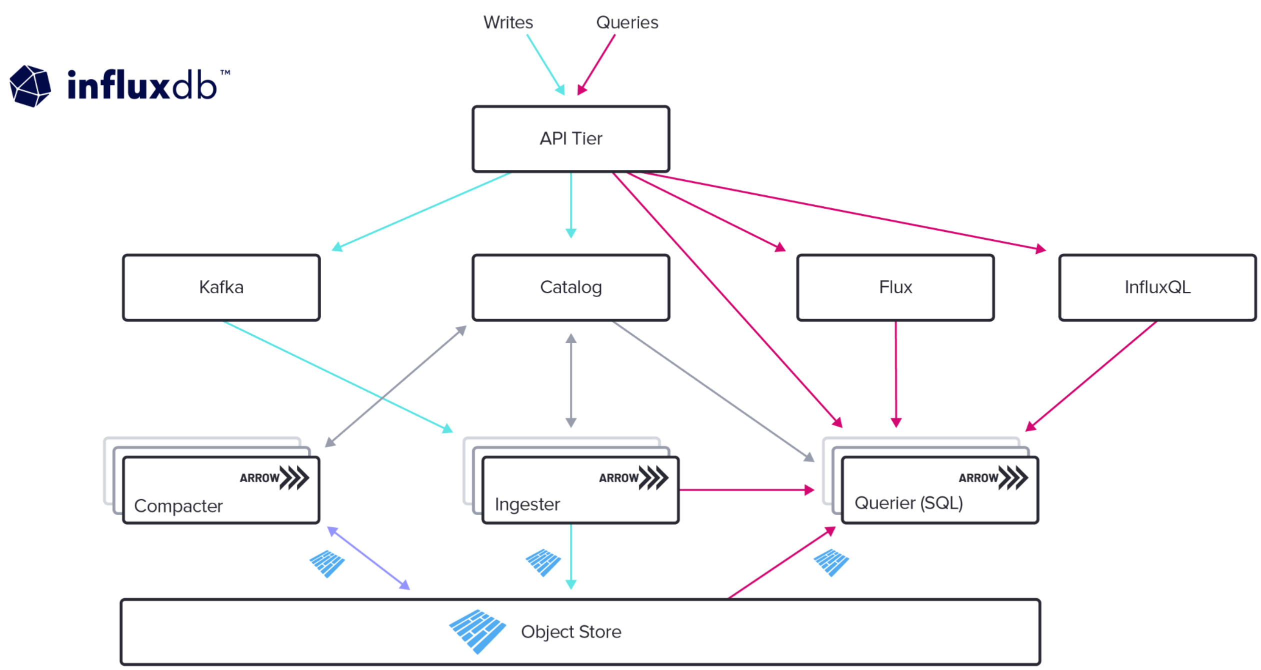
The InfluxDB 3.0 supports SQL queries natively, and offers continued support for InfluxQL. InfluxDB 3.0 also takes advantage of Arrow Flight SQL to offer compatibility with third-party tools like Grafana, PowerBI, and Tableau.

Parquet support

InfluxDB 3.0 persists data to disk as Parquet files, a columnar format that provides extremely high compression ratios. This enables users to store more, high precision data in less space. As a result, you have more data and spend less to keep it. Parquet is also an open storage format that many other services and ecosystems use. Therefore, Parquet files offer opportunities for extending the use and value of time series data through interoperability with Machine Learning tools like DataBricks, Amazon Forecast, H20.AI, and more.

Optimized for low latency queries

InfluxDB 3.0 leverages several open source technologies in the Apache Arrow project. These technologies allowed us to optimize InfluxDB for sub-second query responses using techniques such as vectorization, predicate pushdowns, aggregate pushdowns, parallelism, and more. Taken together, all this means that you can run analytics on the leading edge of data.



To understand how these technologies help build a new engine, we need to know what they are.

- **Rust** is a performant programming language that offers fine-grain memory management.
- **Apache Arrow** is a framework for defining in-memory columnar data.
- **Apache Parquet** is a column-oriented durable file format.
- **Arrow Flight** is a client-server framework designed to transport large datasets over network interfaces without significantly impacting performance.

- [Apache DataFusion]() is an extensible, in-memory query planning, optimization, and execution framework. It's written in Rust and uses Apache Arrow as its in-memory format.

Requirements for the new storage engine

To understand why we chose these technologies and what needs they fulfill, it's helpful to look at the various features and requirements we wanted InfluxDB 3.0 to have. The following table outlines several key requirements and goals for InfluxDB and which technologies are critical for achieving them.

	Requirement/Feature	Rust	Arrow	DataFusion	Parquet
1.	No limits on cardinality. Write any kind of event data and don't worry about what a tag or field is.	X	X	X	X
2.	Best-in-class performance on analytics queries in addition to our already well-served metrics queries.	X	X	X	X
3.	Separate compute from storage and tiered data storage. The DB should use cheaper object storage as its long-term durable store.			X	X
4.	Operator control over memory usage. The operator should be able to define how much memory is used for each buffering, caching, and query processing.			X	
5.	Bulk data import and export.				X

6.	Broader ecosystem compatibility. Where possible, we should aim to use and embrace emerging standards in the data and analytics ecosystem.	X	X	X	X
7.	Run at the edge and in the datacenter. Federated by design.	X			X

Cardinality and performance gains

One of the significant drivers for InfluxDB 3.0 is improving performance. Specifically, we want to enable InfluxDB to handle large datasets without sacrificing performance. In order to achieve this, we needed to solve the cardinality problem, which we did, using the Apache Arrow ecosystem.

Whereas previous versions of InfluxDB required bounded data for tag values, InfluxDB 3.0 doesn't need to differentiate between tag and field values. As a result, the InfluxDB 3.0 can handle nearly unlimited cardinality. InfluxDB has always been able to handle metrics really well, but the updates to InfluxDB 3.0 and the accompanying performance gains open up the number and types of use cases to include those that rely on real-time analysis of large, high cardinality datasets, like tracing and observability. Now users can write massive amounts of time series data, whether that's metrics, events, traces, or logs, to power these use cases without a drop in performance.

Furthermore, InfluxDB 3.0's support for high cardinality isn't just limited to data ingestion. Queries that touch multiple time series are orders of magnitude faster in InfluxDB 3.0 than in our previous versions of InfluxDB. So, whether you're querying across ten series or 1 million series, the database performs the same. This makes analytics across high cardinality data possible.

Conclusion

InfluxDB 3.0 delivers the capabilities and performance for large datasets that users want. As a purpose-built database for time series, InfluxDB 3.0 enables users to create value from leading edge data in real-time. Native SQL support improves developer productivity and reduces barriers to entry. Leveraging the Parquet data format delivers better data compression to lower costs and provides interoperability and extensibility with other popular ecosystems. Flight SQL enables integrations with key third party tools, like Grafana for visualization, to ensure you can get maximum value from your time series data. InfluxDB 3.0 takes time series to the next level, and we can't wait to see what you build with it.

InfluxDB documentation, downloads & guides

[Get InfluxDB]

[Try InfluxDB Cloud for Free]

[Get documentation]

[Additional tech papers]