Building Log Solutions with GridDB

March 20, 2018
Revision 0.05

Table of Contents

List of Figures

Executive Summary

As applications have scaled to handle millions of requests per day, the underlying infrastructure required to monitor those applications has also needed to scale. With many options available to collect, store, analyse, and visualize log data, this white paper demonstrates how to effectively use the GridDB database to build logging solutions.

Introduction

GridDB is an innovative, high performance NoSQL database developed by Toshiba Digital Solutions Corporation whose characteristics make it ideal for storing log data. This whitepaper aims to demonstrate how to build log solutions using GridDB, going from collection to visualization.

Log solutions can store, visualize, and report many different types of application specific data. These logs may include system resource utilization (such as CPU, Memory, or Disk usage percentage), application profiling data (to determine what parts of application use more system resources) or application events such a history of what web pages were accessed or of errors that occurred.

A particular log solution is built of several components, first the *log agent* that reads or generates the *metric* that is then stored in the *data store*, in this case GridDB. The reporting mechanism either generates reports that may be used by some other business process within the organization or the visualization software builds easy to understand graphs, tables, or other visual representations of the log data. When required, the Log solutions alerting system can create tickets in an organization’s issue tracker or notify system administrators that a problem has occurred via SMS, email or other means.

This white paper looks at how GridDB can be used with each component that may be required in a log solution, generating reports with simple and easy-to-use queries, log data can be visualized with Grafana. Built-in trigger functions enable GridDB to integrate with existing ticketing or issue tracking systems to make sysadmins aware of any issues.

Beyond the system components, both GridDB’s unique Key-Container architecture and native Time Series data support will be examined to show how GridDB’s core architecture is ideal for storing both small and large amounts of log data.

Data Sources

Log data can come from many sources, including: system calls that return information about the system, parsing application log files, or by the application itself directly inserting data into the database. These values are called *metrics*.

The *agent* -- or software that processes the data -- sources and either inserts the metrics directly into GridDB or pushes them further into the log-gathering pipeline.

The scope, scale, and system architecture of the application whose metrics are being logged will determine if the agent will directly insert the data into the database or push them into a log-gathering pipeline.

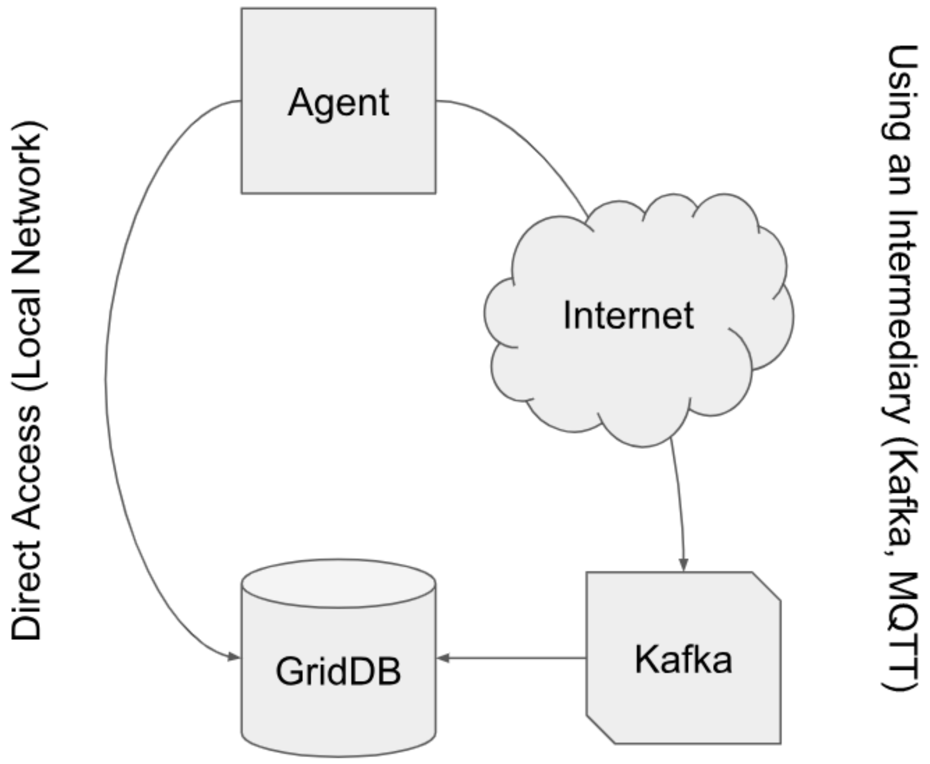

Direct Access (Local Network)
Agent
Internet
GridDB
Kafka
Using an Intermediary (Kafka, MQTT)

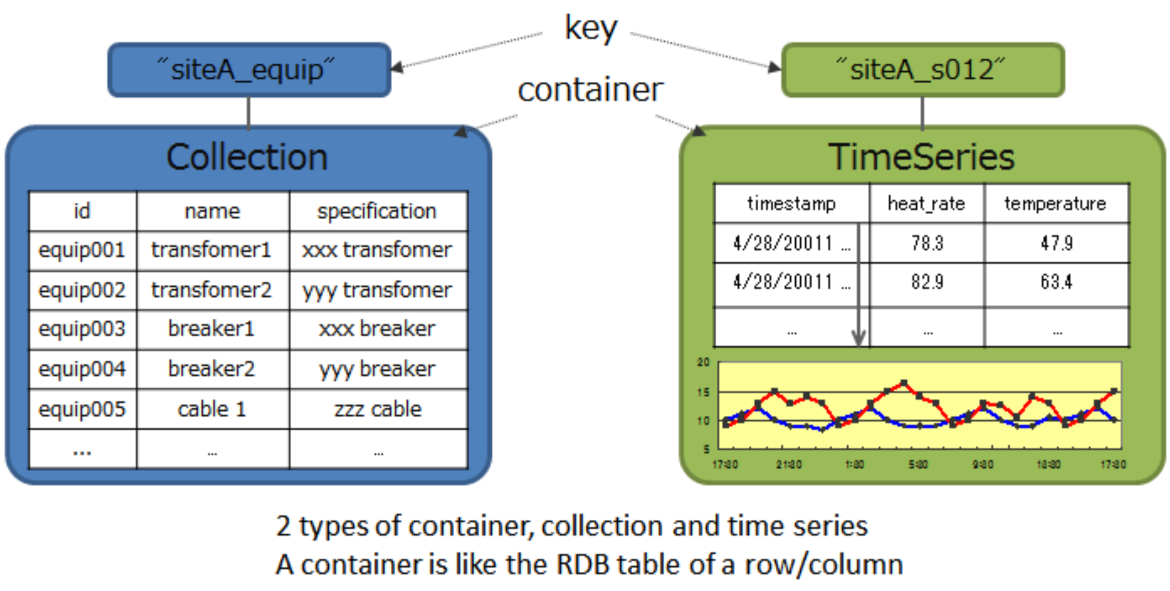
Figure 1: Log Agent Access Methods

One common method for building an Internet-scale log pipeline is to send the logged metrics encapsulated in Message Pack or JSON over MQTT to Kafka. From there, the metrics can be ingested into GridDB.

Containers

One of the most interesting features of GridDB is its Key-Container architecture (as compared to Key-Column [SQL, Cassandra], Key-Document [MongoDB], or Key-Value

[Redis]). The Key-Container model gives the developer the flexibility to structure their data in a myriad of different ways, allowing the ability to enable the most efficient way for querying.



id	name	specification
equip001	transfomer1	xxx transfomer
equip002	transfomer2	yyy transfomer
equip003	breaker1	xxx breaker
equip004	breaker2	yyy breaker
equip005	cable 1	zzz cable
...	...	...

timestamp	heat_rate	temperature
4/28/20011 ...	78.3	47.9
4/28/20011 ...	82.9	63.4
...	...	...

2 types of container, collection and time series
A container is like the RDB table of a row/column

Figure 2: GridDB Containers

For example, if you were interested in measuring memory usage between hosts, you would use one container per host, but if you were interested in measuring how long a request takes for different functions, a developer would use one container per function.

Queries

To build useful reports, a strong query language is required that is able to collect minimum, maximums, averages, and other statistical information about the metrics collected. GridDB uses the TQL query language which is easily understandable and familiar to anyone who has worked with SQL.

```
SELECT * WHERE timestamp > TIMESTAMP('2011-01-01T00:00:00Z') ORDER BY
timestamp LIMIT 100
```

Developers can use several aggregation functions that can calculate the MIN, MAX, AVG, SUM, COUNT, VARIANCE, and STDDEV of a column in a given query.

Instead of executing multiple queries to a container, a consolidated query can be executed once to fetch multiple aggregates of the logged metric. This is useful for acquiring aggregate results such as the daily minimum, maximum, and average values of a particular metric.

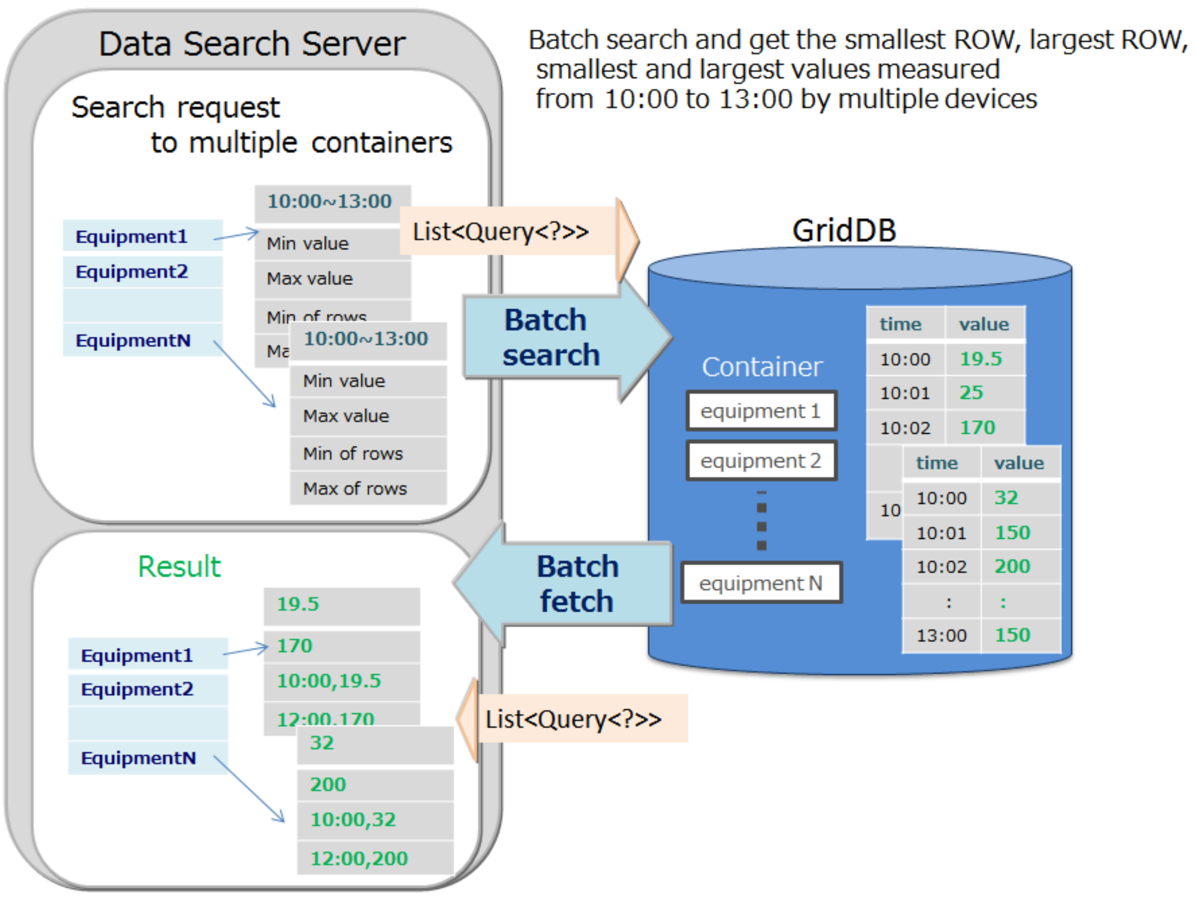


Figure 3: GridDB Multi Query

Time Series Data

As system metrics are taken at a given time and events happen at a specific time, log data is inherently TimeSeries data with timestamps attached to all rows. GridDB was designed with TimeSeries data in mind and packs several features that assist in building log solutions.

The TIME_SAMPLING function is useful for data where events are stored at a high-frequency but the particular query only requires low frequency data. For example, data is stored at one second intervals but only a sampling at ten-minute intervals is required for a particular query.

```
SELECT TIME_SAMPLING(voltage103, TIMESTAMP('2011-07-01T00:00:00Z'),
TIMESTAMP('2011-07-02T00:00:00Z'), 10, MINUTE)
```

The TIME_AVG function provides a convenient method for calculating the time weighted averages of a column even if the intervals between records are irregular.

```
SELECT TIME_AVG(voltage103) FROM plant1
  WHERE TIMESTAMP('2018-07-01T00:00:00Z') <= timestamp
  AND timestamp < TIMESTAMP('2018-08-01T00:00:00Z')
```

GridDB has two features that will help lower the amount of storage required for Time Series, the first is Time Series compression, which can reduce storage required by up to 300%. With the automatic expiration function, Time Series data can be automatically removed from the database when it reaches a certain age using following function:

```
TimeSeriesProperties().setRowExpiration(int elapsedTime, TimeUnit
timeUnit)
```

Alerts

The ability for the database to trigger Alerts is important within a Log Solution to notify system administrators of potential problems such as a server's disk filling up, repeated unsuccessful login attempts, or long request times.

You can create alerts using GridDB's trigger function. This will send an automatic notification using either Java Messaging Service (JMS) or a REST API call when an operation (add/update or delete) is carried out on the row data of a container that matches the given set of conditions. Event notifications can be received without the need to poll and monitor database updates in the application system.

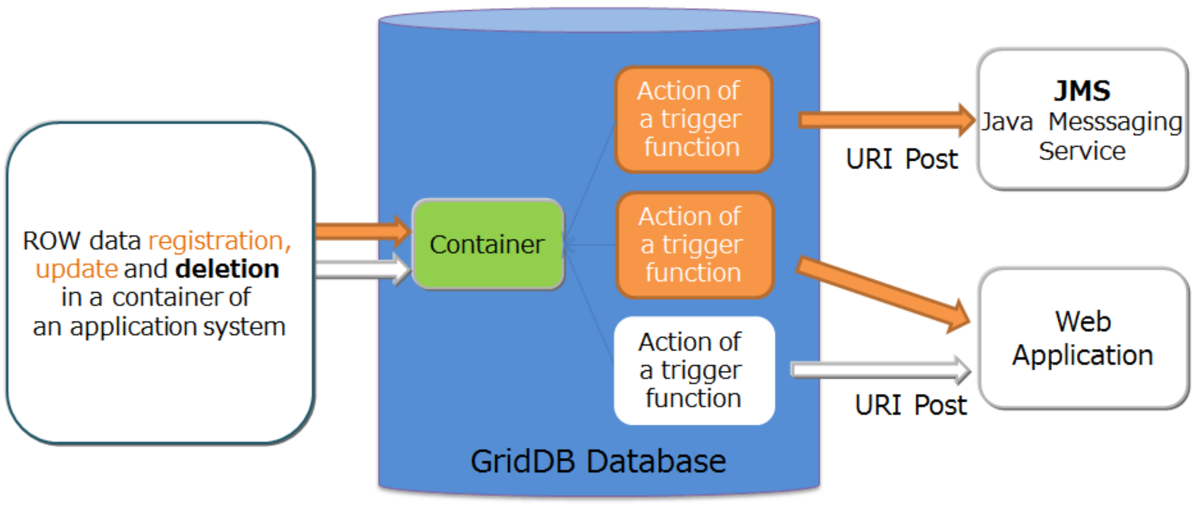

ROW data registration, update and deletion in a container of an application system
Container
Action of a trigger function
Action of a trigger function
Action of a trigger function
GridDB Database
URI Post
JMS Java Messaging Service
Web Application
URI Post

Figure 4: Trigger Functionality

Multiple triggers can be set in a single container and additionally, when a trigger occurs, the application can also be notified of the data that triggered the alert.

Visualization

While there are a multitude of different ways to visualize data in GridDB, using Grafana has the best support and two different methods of accessibility available; the GridDB Grafana plugin which uses the GridDB WebAPI or a custom connector that implements the API endpoints required by SimpleJSON datasource plugin and allows customized queries to be performed.

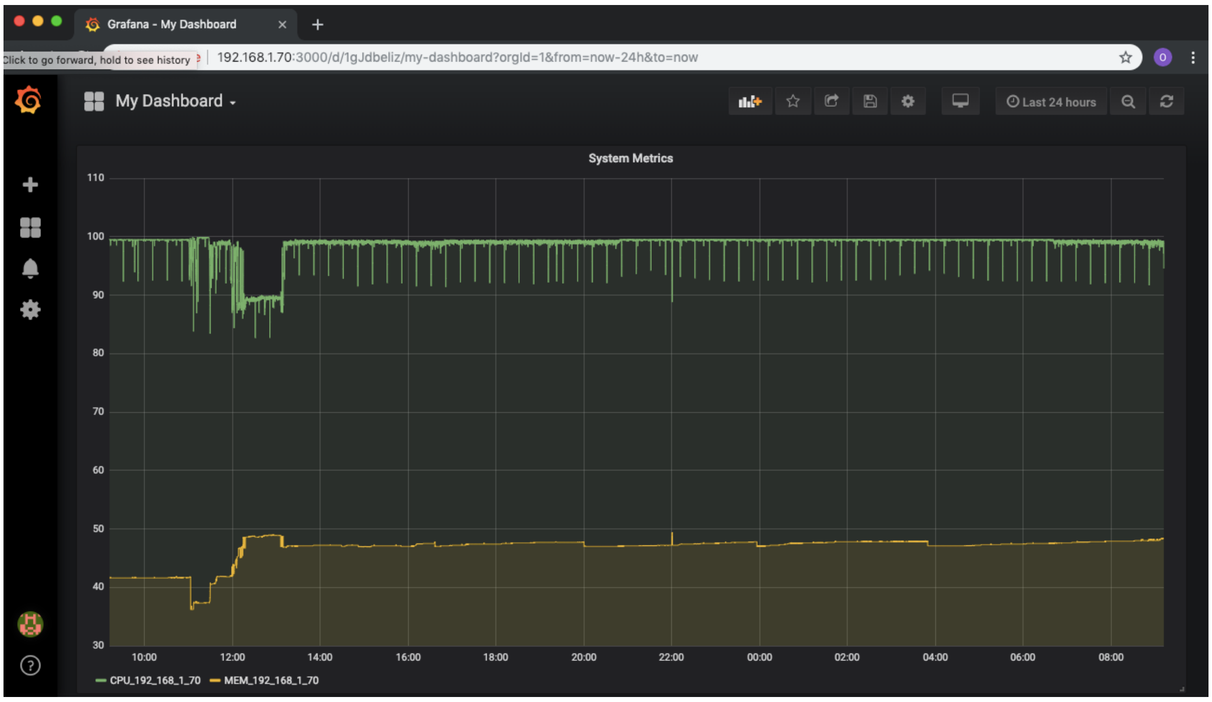


Figure 5: Grafana Screenshot

The GridDB Grafana plugin is best used when simply displaying metrics while building and using GridDB Grafana JSON connector is useful when more complex querying is required such as when building a dashboard of response times for certain conditions.

Performance

When building a log solution, often the computational cost of the log framework is just as intensive as the application it is monitoring. GridDB features a hybrid in-memory architecture, parallel query processing and minimal overhead that produces incredible performance.

As Fixstars has demonstrated in previous benchmarks against other databases such as Cassandra, InfluxDB, and MariaDB (MySQL), GridDB's higher performance means it requires a fraction of the resources as its competitors.

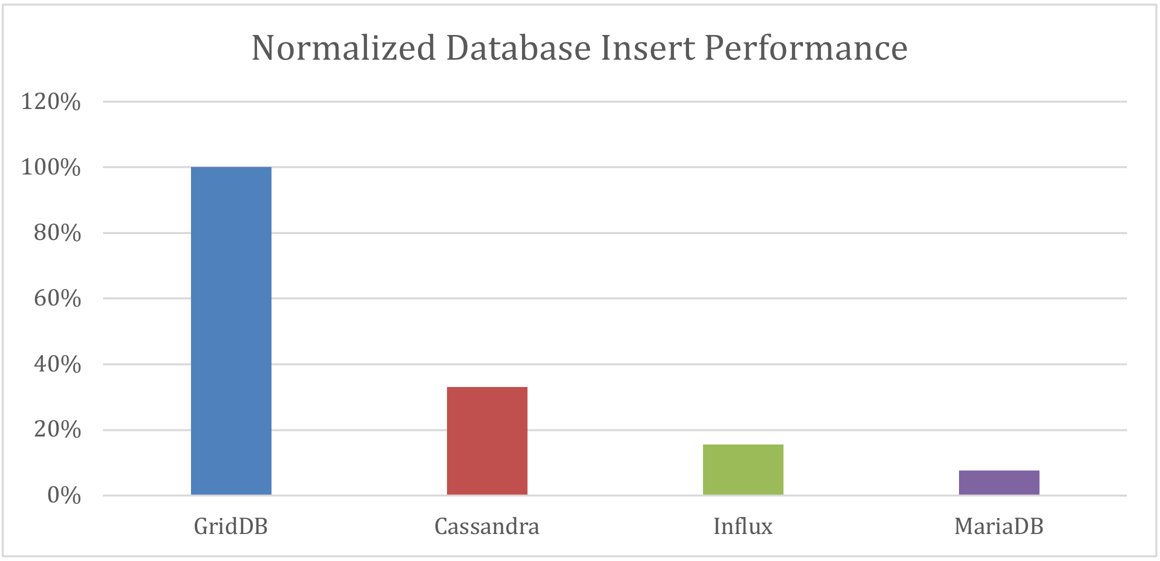


Figure 6: Write Performance Comparison

The above and below graphs represent the relative performance of GridDB and Cassandra, InfluxDB, and MariaDB using their specific test workload. For benchmarking Cassandra, the comparison[1] is using YCSB Insert and Workload B, while the InfluxDB benchmark[2] used YCSB-TS's Insert and Workload A and the MariaDB comparison[3] used an in-house developed workload that mimicked a common transactional insert with aggregation queries. All of the shown benchmarks are using one database system and one application system.

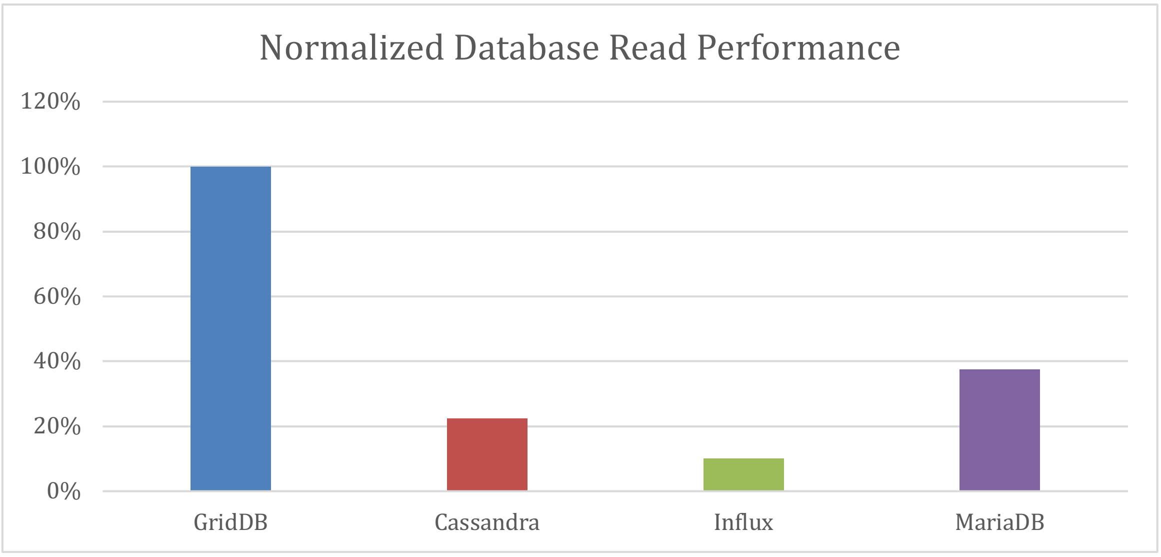


Figure 7: Read Performance Comparison

[1] https://griddb.net/en/docs/Fixstars_NoSQL_Benchmarks.pdf
[2] https://griddb.net/en/docs/TimeSeries_Database_Benchmark_GridDB_InfluxDB.pdf
[3] https://griddb.net/en/docs/Benchmarking_Application_GridDB_MariaDB.pdf

Conclusion

As this whitepaper demonstrates, GridDB is well suited to building a Log Solution with extremely high performance. The upgraded performance brought on by GridDB allows for reduced operational costs and also a combination of features and capabilities that enable integration with other system components, which includes, most notably, integration with the very popular visualization tool Grafana.