

Cloud Time Series Database Evaluation

GridDB Cloud vs. InfluxDB Cloud vs. MongoDB Atlas

Version 1.0

2026

Fixstars Solutions, Inc.



[Executive Summary..... 3](#)

[Pricing Structures..... 4](#)

[Summary of Pay-As-You-Go Pricing..... 5](#)

[Test Environment and Methodology..... 5](#)

[Database Tiers and Cardinality Constraints..... 5](#)

[Data Modelling..... 5](#)

[Load Performance \(Data Ingestion\)..... 7](#)

[Why Batch Size Matters in Production..... 8](#)

[Why Workers Matter in Production..... 9](#)

[Query Performance Evaluation..... 9](#)

[Query Results: GridDB Cloud vs. InfluxDB Cloud..... 10](#)

[Section 1 Conclusion..... 11](#)

[Pricing Structures..... 12](#)

[GridDB Cloud — Fixed Monthly Commitment \(Shared Instance\)..... 12](#)

[MongoDB Atlas M40 Dedicated \(Azure\)..... 12](#)

[Summary of Fixed-Price Tier Pricing..... 13](#)

[Test Environment and Methodology..... 14](#)

[Data Modelling..... 15](#)

[Load Performance \(Data Ingestion\)..... 16](#)

[Query Performance Evaluation..... 17](#)

[Query Results: GridDB Cloud vs. MongoDB Atlas M40..... 18](#)

[Section 2 Conclusion..... 19](#)

[Overall Conclusion..... 20](#)

Executive Summary

This white paper presents a comprehensive performance and cost evaluation of three managed cloud time series databases: GridDB Cloud, InfluxDB Cloud, and MongoDB Atlas. Using the standardized Time Series Benchmark Suite (TSBS), we assess data ingestion throughput and query execution speeds across real-world IoT and DevOps workload patterns.

A critical finding of this evaluation is that no single pricing tier tells the whole story. GridDB Cloud's flexible offering spans two distinct purchasing models—pay-as-you-go and fixed monthly commitment—each suited to different operational profiles. This paper is therefore structured in two sections:

- Section 1 — Pay-As-You-Go Tier: GridDB Cloud vs. InfluxDB Cloud. These two platforms share an identical pricing structure, making performance and architectural reliability the decisive factors.
- Section 2 — Fixed-Price / Enterprise Tier: GridDB Cloud (Fixed Monthly Commitment, \$520/mo) vs. MongoDB Atlas M40 Dedicated (\$927.10/mo). At higher scale, a predictable flat-rate plan becomes attractive, and this section evaluates whether MongoDB's M40 tier can justify its higher price tag against GridDB's fixed offering.

Across both scenarios, GridDB Cloud demonstrates superior performance—scaling further under concurrency, sustaining higher ingest rates, and maintaining consistent low-latency query execution. The choice between GridDB's two plans ultimately comes down to your workload profile: bursty and variable, or heavy and continuous.

Database Backgrounds

- GridDB Cloud is a fully managed, highly scalable time series database that utilizes a unique Key-Container data model, optimizing data isolation and fast querying for individual devices. It is available on the Azure Marketplace in both pay-as-you-go and fixed monthly commitment tiers.
- InfluxDB Cloud is a purpose-built time series platform delivered as a fully managed cloud service, known for its rich ecosystem and specialized Line Protocol for metric ingestion.
- MongoDB Atlas is a fully managed cloud database with specialized time series collections, bringing time series capabilities to its widely used document model.

The Time Series Benchmark Suite (TSBS) is a standard set of applications for generating data and queries to evaluate time series databases. For this evaluation, the cpu-only subset of the TSBS DevOps use case was utilized—tracking 10 CPU metrics such as `usage_user` and `usage_system` across a set of monitored hosts.

Section 1: Pay-As-You-Go — GridDB Cloud vs. InfluxDB Cloud

This section compares GridDB Cloud and InfluxDB Cloud under their respective pay-as-you-go tiers. Because both platforms share an identical pricing model, this evaluation focuses on where they diverge: performance, architectural constraints, and operational reliability under load.

Pricing Structures

Both GridDB Cloud and InfluxDB Cloud are available on the Microsoft Azure Marketplace under pay-as-you-go tiers with the following identical rate structure:

Pricing Component	GridDB Cloud (PAYG)	InfluxDB Cloud (PAYG)
Base Compute	\$0.00 / month (Shared vCPU/RAM, 4 vCPU / 16 GB memory)	\$0.00 / month
Data In (Ingestion)	\$0.0025 per MB	\$0.0025 per MB
Data Out (Egress)	\$0.09 per GB	\$0.09 per GB
Storage	\$0.002 per GB-hour	\$0.002 per GB-hour
Query Count	\$0.012 per 100 requests	\$0.012 per 100 query and task runs

GridDB Pay As You Go Marketplace Pricing

Plan	Description	Price/billing frequency	Contract duration	Total for term
Pay-As-You-Go Get it now	This is a GridDB Cloud pay-as-you-go plan. This plan has the following features: <ul style="list-style-type: none"> Shared vCPU Shared RAM Up to 100GB Storage Up to 10,000 DB requests per 10 minutes 	\$0.00/month Plus: Request: \$0.012 100 requests Storage: \$0.002 1gb/1hour Data Out: \$0.09 1gb Data In: \$0.0025 1mb	1-month subscription	\$0.00 for 1 month

InfluxDB Pay As You Go Marketplace Pricing

Plan	Description	Price/billing frequency	Contract duration	Total for term
Usage-based plan Get it now	With a usage-based pricing model, InfluxDB Cloud users are charged based on the work performed. Because of this, users no longer have to allocate or size a server as a prerequisite for running workloads. The usage-based plan flexibly scales to any sized workload with fair pricing that helps avoid over-provisioning as your workload expands or contracts.	\$0.00/month Plus: Data Out: \$0.09 gb Query Count: \$0.012 100 query and task runs Storage: \$0.002 gb-hour Data In: \$0.0025 mb	1-month subscription	\$0.00 for 1 month
		\$0.00/year Plus: Data Out: \$0.09 gb Query Count: \$0.012 100 query and task runs Storage: \$0.002 gb-hour Data In: \$0.0025 mb	1-year subscription	\$0.00 for 1 year

Summary of Pay-As-You-Go Pricing

Since pricing is identical, the total cost for any given workload is the same on both platforms. The table below illustrates representative costs:

IoT Workload Size	GridDB Cloud / InfluxDB (Pay-As-You-Go)
Small Prototype(10 GB Ingested & Stored)	~\$40.20 / month(Ingest: \$25.60 + Storage: \$14.60)
Heavy Workload(100 GB Ingested & Stored)	~\$402.00 / month(Ingest: \$256.00 + Storage: \$146.00)

The key takeaway: at the pay-as-you-go tier, price is not the differentiator. Performance, stability, and architectural scalability are.

Test Environment and Methodology

To conduct a fair and rigorous evaluation across different cloud providers, the testing infrastructure and benchmarking software required specific configurations.

The application server driving the benchmarks was a Microsoft Azure Standard F2s v2 instance (2 vCPUs, 4 GiB memory) located in the India region. The database cluster for GridDB Cloud was provisioned in Singapore, connected via Azure VNet peering to minimize latency and simulate a secure enterprise architecture. InfluxDB Cloud was provisioned in Amsterdam due to regional availability constraints; communication was limited to their standard Web API as VNet peering is not available.

Due to architectural differences between the databases, customized TSBS implementations were used:

- GridDB Cloud: A custom Java implementation of the TSBS Go code was developed. Java serves as the native interface for GridDB, bypassing the overhead of Go drivers to accurately measure peak ingestion and query performance.
- InfluxDB Cloud: Testing utilized QuestDB's optimized TSBS fork for InfluxDB.

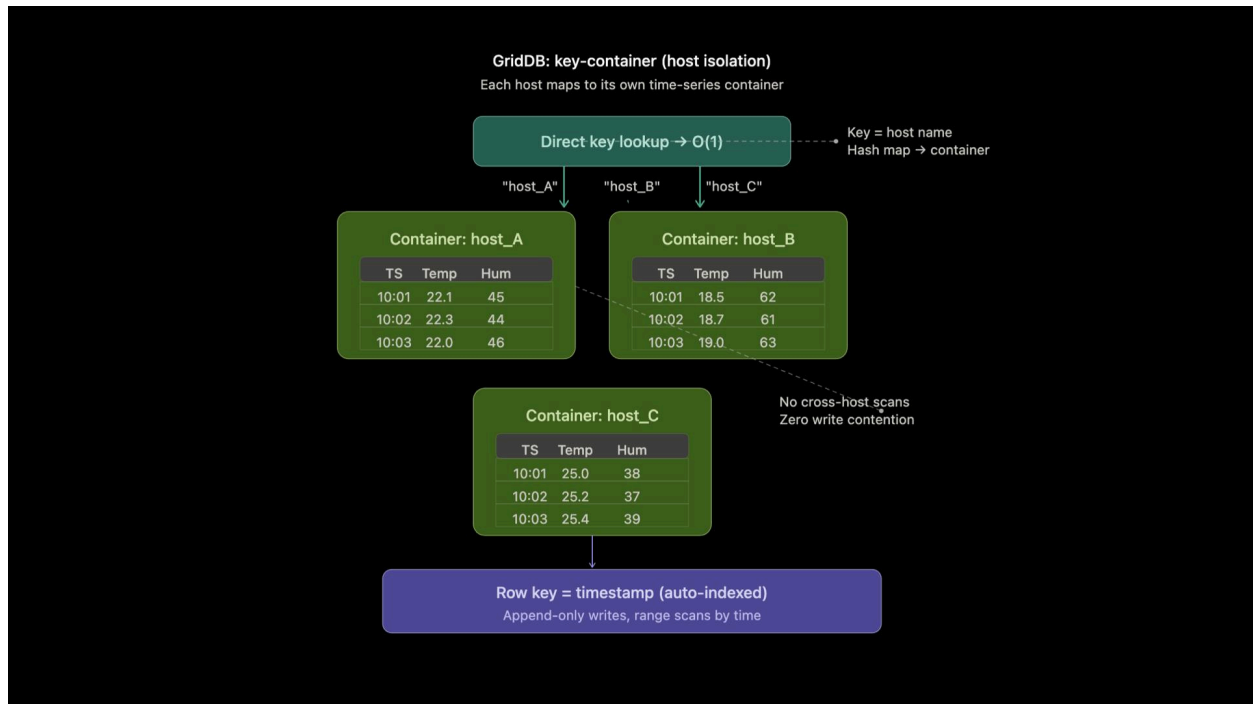
Database Tiers and Cardinality Constraints

An important constraint in this section is that InfluxDB Cloud's pay-as-you-go tier enforces strict rate limits based on cardinality (the number of unique time series or devices being tracked). While typical evaluations might scale to 100,000 hosts, this benchmark was strictly limited to 20 hosts. Exceeding this threshold triggered immediate throttling and rate-limit errors on InfluxDB, which would have compromised the fairness of the comparison.

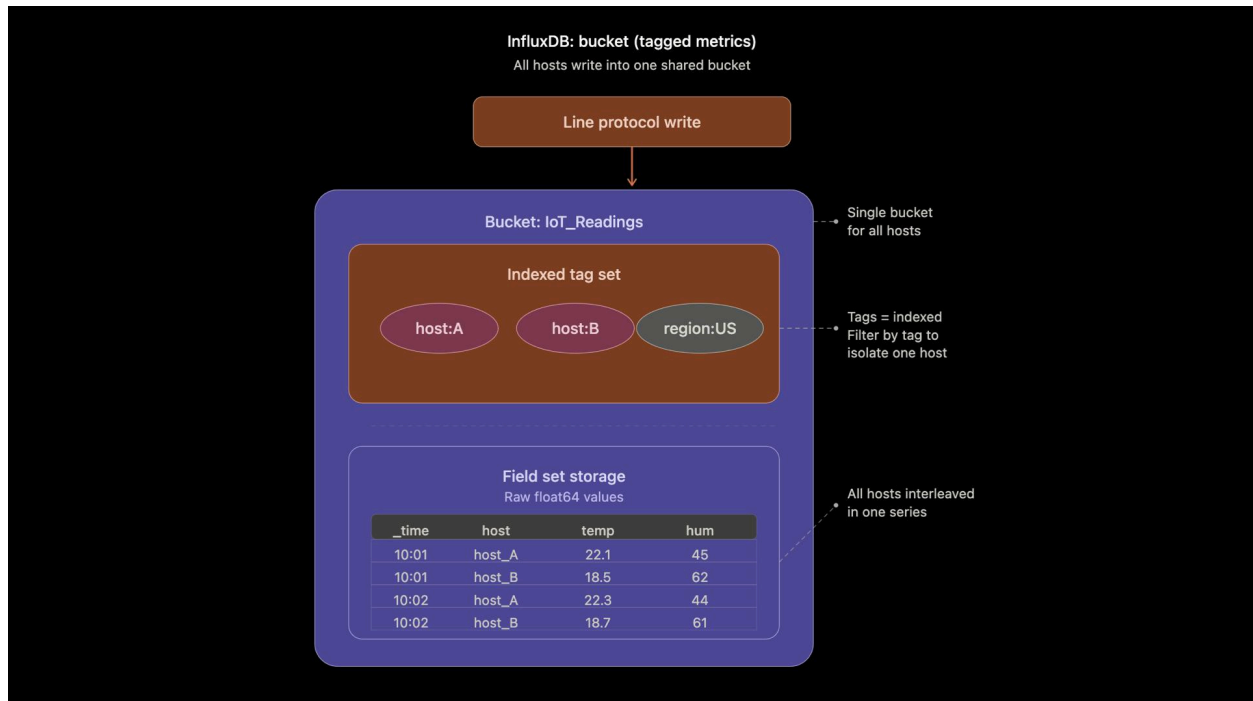
Data Modelling

Benchmarking these databases presents a unique challenge, as their underlying architectures and data models are significantly different.

- GridDB Cloud (Key-Container Model): GridDB assigns a separate container (table) to each individual host or device. This provides exceptional isolation and makes querying a single container extremely fast—the database never needs to scan a global table to retrieve device-specific data.



- InfluxDB Cloud (Bucket and Line Protocol): InfluxDB uses a structure based on buckets, measurements, tag sets (indexed metadata such as hostnames), and field sets (the actual metric values). Data is ingested using InfluxDB’s specialized Line Protocol, optimized for timestamped metrics but enforcing a strict separation between indexed tags and unindexed fields.



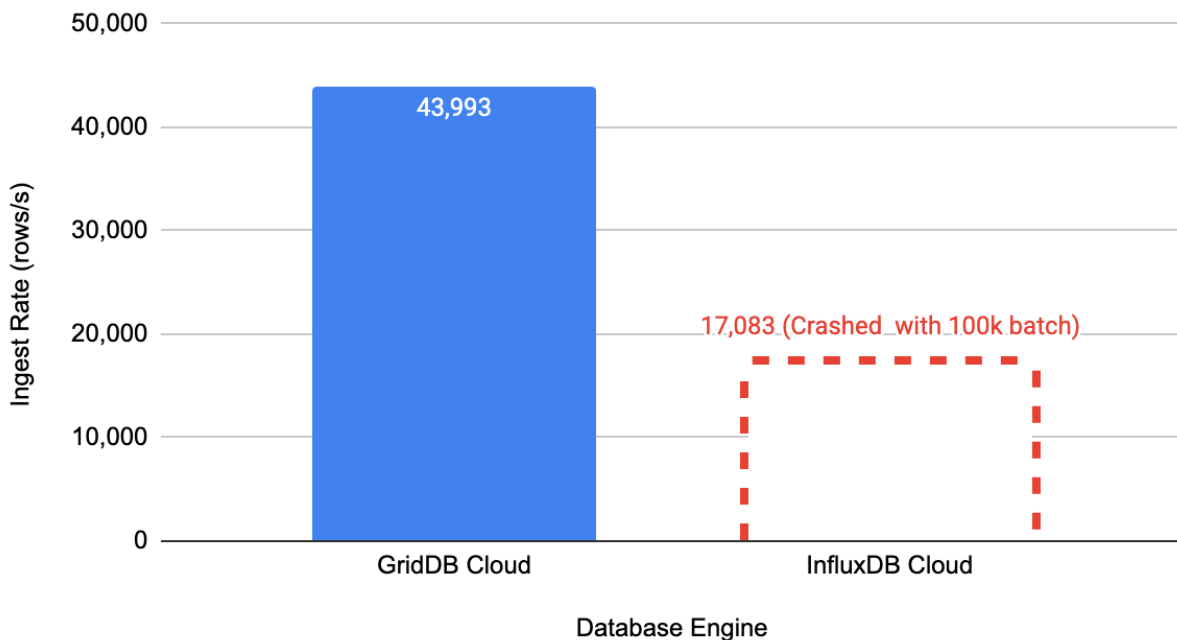
Load Performance (Data Ingestion)

Data ingestion was measured in rows per second and metrics per second. Varying batch sizes were tested to find the optimal ingestion rate for each platform, as network overhead and API limitations heavily influence cloud database performance.

Starting with a batch size of 100,000 rows per transaction: InfluxDB Cloud crashed immediately, throwing 429 Too Many Requests errors. GridDB Cloud successfully completed the load, peaking at 43,993 rows/sec—demonstrating exceptional resilience to large payload sizes.

Database Engine	Workers	Batch Size	Ingest Rate (Rows/sec)	Ingest Rate (Metrics/sec)
GridDB Cloud	2	100,000	43,993	439,928
InfluxDB Cloud	2	100,000	N/A	N/A

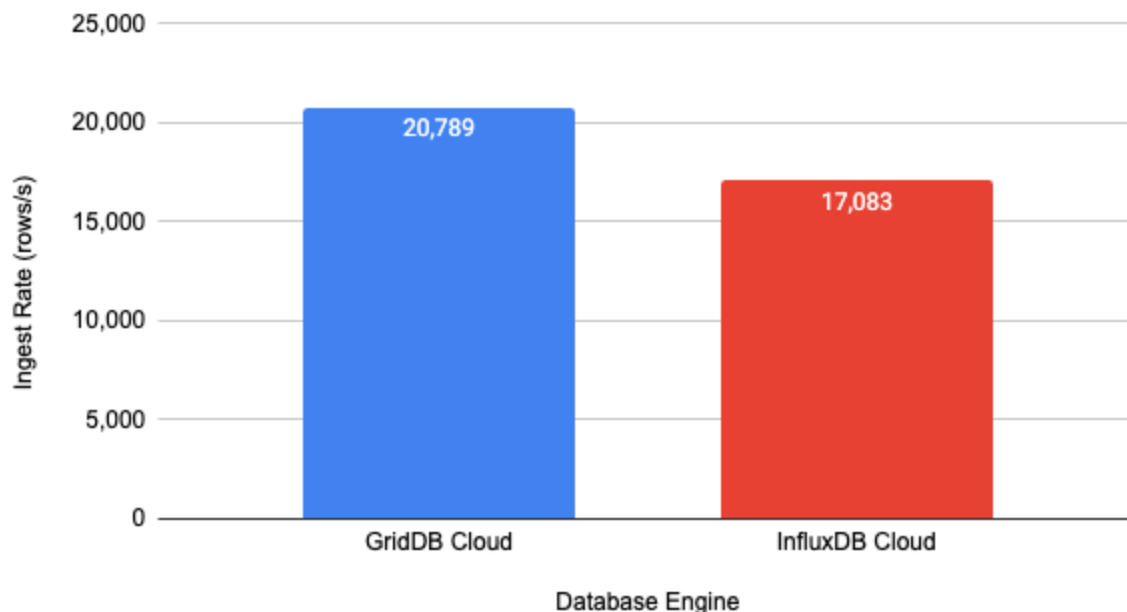
Ingest Rate (rows/s) Batch Size: 100k Rows (higher is better)



Dropping to a more realistic batch size of 20,000 rows per transaction, InfluxDB was able to succeed; here is the final comparison:

Database Engine	Workers	Batch Size	Ingest Rate (Rows/sec)	Ingest Rate (Metrics/sec)
GridDB Cloud	2	20,000	20,789	207,890
InfluxDB Cloud	2	20,000	17,083	170,830

Ingest Rate (rows/s) Batch Size: 20k Rows (higher is better)



As an experiment, more ingest configurations were attempted. The results showed that GridDB scales cleanly with additional concurrency — doubling workers from 2 to 4 (20k batch) raised throughput from ~21k to ~40k rows/sec. InfluxDB Cloud, by contrast, hits a hard concurrency ceiling: 4 workers consistently failed with 503 connection termination errors from the Influx write API. The maximum sustainable configuration was 3 workers at 20k batch size, yielding 22,510 rows/sec — less than half of GridDB’s 4-worker throughput, with no headroom to scale further.

Database	Max Workers	Batch	Ingest Rate (Rows/sec)
GridDB Cloud	4+ (tested)	20,000	39,968
InfluxDB Cloud	3 (hard ceiling)	20,000	22,510

Why Batch Size Matters in Production

In production environments, network I/O is expensive. Sending 100,000 rows in a single batch requires just one network round-trip, while the same data in 20,000-row increments requires five round-trips, multiplying latency and connection overhead.

When cloud databases enforce strict payload limits, engineering teams are forced to build and maintain complex buffering layers to prevent data loss. A database that natively supports massive batch ingestion allows for a simpler, more resilient architecture that absorbs sudden traffic spikes. This is particularly critical in two common modern architectures:

- High-Throughput Pipelines (e.g., Apache Kafka): Message brokers batch IoT sensor data before pushing to storage. If the downstream database limits batch sizes, Kafka must slow itself down—potentially falling behind the data stream.

- **Edge Connectivity:** In unstable network environments, a device may accumulate a massive backlog of data. If the cloud database cannot handle large burst ingests, data loss or crashes result.

Why Workers Matter in Production

In benchmarking terms, a "worker" is a concurrent client thread that opens its own connection to the database and pushes or pulls data in parallel with other workers. In a real production system, a worker maps directly to the things actually talking to your database: an ingestion service pod in Kubernetes, a Kafka consumer thread draining a partition, an IoT gateway flushing buffered sensor readings, or a backend API instance serving a live dashboard. Each one holds an open connection and competes for the database's CPU, memory, and lock structures.

Worker count is therefore a direct proxy for how many independent producers and consumers your database can sustain at once. A database that scales linearly with workers means you can scale your fleet horizontally — add more ingestion pods, more dashboard users, more edge devices — and trust that throughput will climb with you. A database that plateaus or crashes as workers increase forces architectural compromises: connection pooling layers, request queueing, backpressure logic, and ultimately a hard ceiling on how much real-world traffic you can absorb.

This matters acutely in production patterns:

- **Horizontal ingest fleets:** Modern IoT and observability pipelines fan out ingestion across many stateless workers (Kafka consumers, Lambda functions, container replicas) so that a single slow producer doesn't block the stream. If the database caps out at 3 workers — as InfluxDB Cloud did at this tier — your fleet's effective parallelism is capped at 3, no matter how many consumers you deploy.

Worker scalability, in short, sets the ceiling on how much production traffic your architecture can absorb before you have to engineer around the database rather than with it.

Query Performance Evaluation

Before results are presented, a critical anomaly must be noted: **InfluxDB Cloud exhibited a strict API rate limit across all queries.** Regardless of query complexity, InfluxDB was throttled to approximately 11.5 queries per second (QPS) with an artificial latency floor of roughly 173 ms. This is a significant operational constraint of the pay-as-you-go cloud tier compared to InfluxDB's self-hosted potential.

For this benchmark, a curated suite of five distinct TSBS query patterns was used to stress-test the databases across the complete spectrum of real-world data retrieval scenarios:

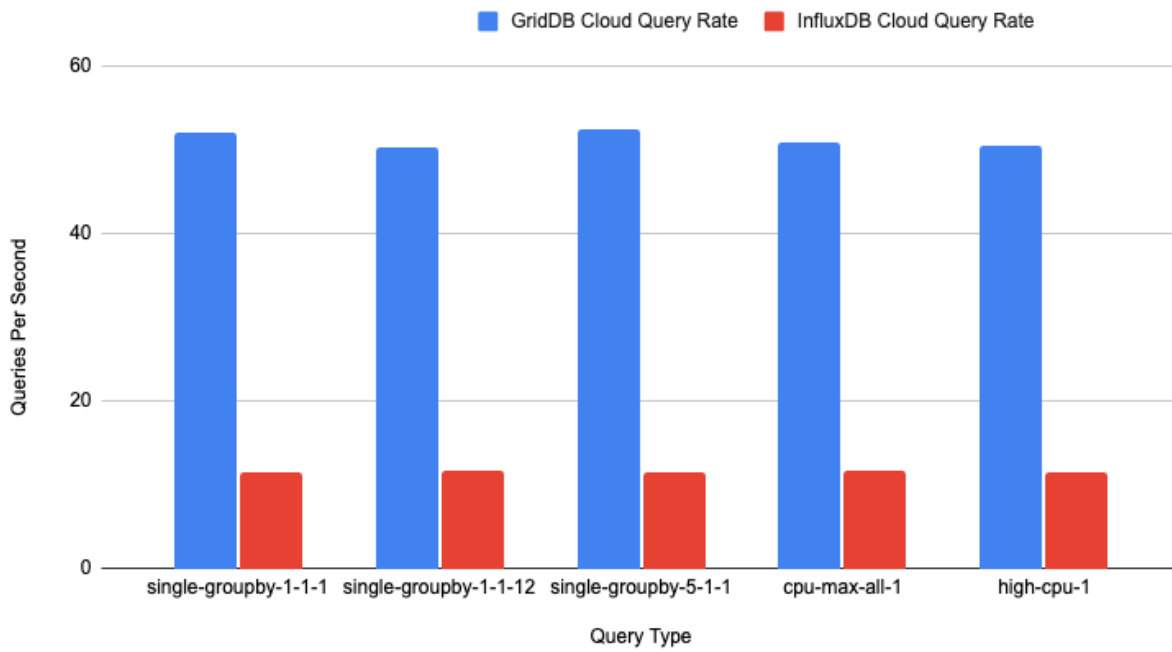
Query Name	Technical Behavior	Real-World IoT Equivalent
single-groupby-1-1-1	High-Resolution Aggregation: Groups a single metric into 1-minute buckets over a 1-hour window.	Minute-by-minute temperature averages for a single machine to diagnose micro-fluctuations.
single-groupby-1-1-12	Historical Trend Retrieval: Groups a single metric into 1-minute buckets over a 12-hour window.	Half-day operational trend line for gradual shifts in battery drain or heat generation.
single-groupby-5-1-1	Multi-Metric Aggregation: Groups 5 distinct metrics into 1-minute buckets over a 1-hour window.	Simultaneously tracking temperature, vibration, voltage, and pressure for a single asset.
cpu-max-all-1	Multi-Metric Peak Aggregation: Calculates the maximum peak for 5 distinct metrics grouped into 1-hour buckets.	Hourly summary report isolating the highest temperature and vibration recorded during a shift.
high-cpu-1	Threshold Filtering: Scans raw data for any single value exceeding a set limit (e.g., > 90.0).	Continuously scanning a sensor stream to identify when a turbine's RPM exceeds safe operational limits.

Query Results: GridDB Cloud vs. InfluxDB Cloud

Query Type	GridDB QPS	InfluxDB QPS	GridDB Latency	InfluxDB Latency	Winner
single-groupby-1-1-1	52.10	11.58	36.68 ms	172.43 ms	GridDB
single-groupby-1-1-12	50.26	11.60	37.56 ms	172.27 ms	GridDB
single-groupby-5-1-1	52.50	11.44	36.60 ms	174.70 ms	GridDB
cpu-max-all-1	50.85	11.62	39.95 ms	171.99 ms	GridDB
high-cpu-1	50.63	11.49	37.69 ms	173.99 ms	GridDB

Analysis: GridDB Cloud outperforms InfluxDB Cloud across every single query type, often by a factor of 4–5x in raw Queries Per Second (QPS) and with latency that is 4-5x lower. InfluxDB's artificial rate ceiling of ~11.5 QPS makes it impractical for any production IoT workload requiring real-time analytics at this pricing tier. GridDB's Key-Container model excels particularly on historical trend queries (single-groupby-1-1-12), where its ability to retrieve sequential data from a single container maintains consistent ~37 ms latency regardless of the time window being queried.

GridDB Cloud & InfluxDB Cloud (2 Worker) - Query Rate (higher is better)



Section 1 Conclusion

At the pay-as-you-go tier, GridDB Cloud and InfluxDB Cloud cost exactly the same. But they perform very differently. GridDB Cloud delivers 4-5x higher query throughput, 4-5x lower latency, and dramatically superior ingest resilience—handling batch sizes that cause InfluxDB to crash outright. For any IoT or DevOps workload at this pricing tier, GridDB Cloud is the clear choice.

Section 2: Fixed-Price Enterprise Tier — GridDB Cloud vs. MongoDB Atlas M40

As workloads grow heavier and more continuous, pay-as-you-go pricing becomes increasingly expensive. Both GridDB Cloud and MongoDB Atlas offer dedicated, fixed-price plans designed for organizations that need predictable billing alongside enterprise-grade capacity. This section compares:

- GridDB Cloud — Fixed Monthly Commitment (Shared Instance): \$520.00/month flat. Includes shared vCPU/RAM (equivalent to the PAYG tier hardware: 4 vCPU / 16 GB memory), up to 100 GB storage, and up to 10,000 DB requests per 10 minutes.
- MongoDB Atlas M40 Dedicated (Azure): ~\$927.10/month base. Includes dedicated compute with higher vCPU/RAM than MongoDB’s lower tiers, 64 GB included storage.

The central question is: does MongoDB’s M40 dedicated cluster—costing \$407.10/month more—offer enough performance to justify the premium over GridDB’s fixed plan?

Pricing Structures

GridDB Cloud — Fixed Monthly Commitment (Shared Instance)

- Base Compute: \$520.00 / month (fixed, all-inclusive)
- Shared vCPU / Shared RAM
- Up to 100 GB Storage included
- Up to 10,000 DB requests per 10 minutes
- Available on Microsoft Azure Marketplace

Plan	Description	Price/billing frequency	Contract duration	Total for term
Fixed Monthly Commitment Get it now	This is a Fixed Monthly commitment plan. This plan has the following features: <ul style="list-style-type: none"> ◦ Shared vCPU ◦ Shared RAM ◦ Up to 100GB Storage ◦ Up to 10,000 DB requests per 10 minutes 	\$520.00/month	1-month subscription	\$520.00 for 1 mont


MongoDB Atlas M40 Dedicated (Azure)

- Base Compute: ~\$1.27/hour (~\$927.10/month)
- Dedicated compute (higher vCPU/RAM than M10)
- 64 GB Storage included


- Unlimited queries up to hardware capacity
- Data In (Ingestion): \$0.09/GB
- Data Out (Egress): Standard Azure rates (~\$0.09/GB)
- Storage Overage: \$0.25/GB/month beyond included storage
- Bumping up the CPU tier from “Low” to “Normal” and bumping up the storage to match GridDB (100GB) increases the price from the base \$0.99/hr you see below up to \$1.27/hr which we used for all calculations.

Dedicated Cluster


Pay-as-you-go! Clusters are billed hourly with monthly invoices.



Amazon Web Services



Microsoft Azure



Google Cloud

Cluster Tier	Storage	RAM	vCPU	Base Price
M10	8–128 GB	2 GB	1 vCPU	\$0.08/hr
M20	16–256 GB	4 GB	1 vCPU	\$0.19/hr
M30	32–512 GB	8 GB	2 vCPUs	\$0.51/hr
M40	64 GB–1 TB	16 GB	4 vCPUs	\$0.99/hr

Summary of Fixed-Price Tier Pricing

Unlike the PAYG comparison, these two platforms have meaningfully different price points. The table below shows how costs compare across workload sizes, using GridDB’s pay-as-you-go rate (since GridDB’s fixed plan is most useful for bursty workloads, while PAYG costs scale with volume):

IoT Workload Size	GridDB Cloud (Best Plan)	MongoDB Atlas (M40 Dedicated)	Who Wins?
Small Prototype (5 GB Ingested & Stored, minimal queries)	~ \$20.10 / month(Pay-As-You-Go: Ingest \$12.80 + Storage \$7.30)	~ \$927.10 / month**(Base M40 Compute; 64GB Storage included)	GridDB — dramatically cheaper for low-volume workloads due to \$0 base compute.
Medium Workload (100 GB Ingested & Stored, low queries)	~ \$402.00 / month**(Pay-As-You-Go: Ingest \$256.00 + Storage \$146.00)	~ \$927.10 / month + Extras**(Base M40 + storage overage + hidden costs)	GridDB — still less than half the price of Atlas before even hitting the Fixed tier threshold.
Heavy Analysis Workload (100 GB Ingested & Stored, Heavy Queries e.g., Live Dashboards running 1.5M+ requests)	\$520.00 / month(Fixed Monthly Commitment — shields against high query volume that would push PAYG over \$580)	~ \$927.10 / month + Extras**(Fixed compute; storage overage and high data transfer/network costs add up)	GridDB Fixed Plan — ~\$400+/mo cheaper than M40, explicitly built to absorb high-frequency IoT queries without price spikes.

The key insight: GridDB Cloud wins on price at every stage of growth—not just the low end. For small and medium workloads, GridDB’s pay-as-you-go (PAYG) plan is a fraction of the cost of MongoDB Atlas’s **\$927.10/month** M40 base fee. As operational intensity grows—specifically when heavy query volumes from live dashboards or automated systems push PAYG costs toward the **\$520/month** threshold—organizations simply switch to GridDB’s Fixed Monthly Commitment. This locks in a flat \$520 rate, delivering massive IoT-scale performance while remaining over **\$400/month cheaper** than MongoDB M40. MongoDB’s pricing only appears competitive if mistakenly weighed against uncapped PAYG query costs; head-to-head against GridDB’s fixed plan, MongoDB M40 is vastly more expensive and slower across every benchmark.

Test Environment and Methodology

For this section, both databases were freshly wiped and re-provisioned before testing began. MongoDB Atlas was scaled up from the M10 tier (used in initial benchmarking) to the M40 dedicated cluster to provide a more equivalent hardware footprint to GridDB’s shared instance.

The application server remained the same Microsoft Azure Standard F2s v2 instance (2 vCPUs, 4 GiB memory) in the India region. Both GridDB Cloud and MongoDB Atlas were connected via Azure VNet peering and both clusters were provisioned in Singapore. Worker counts were significantly increased for this section to stress-test how each database scales under concurrent load—a critical capability for production IoT deployments.

The same customized TSBS implementations were used:

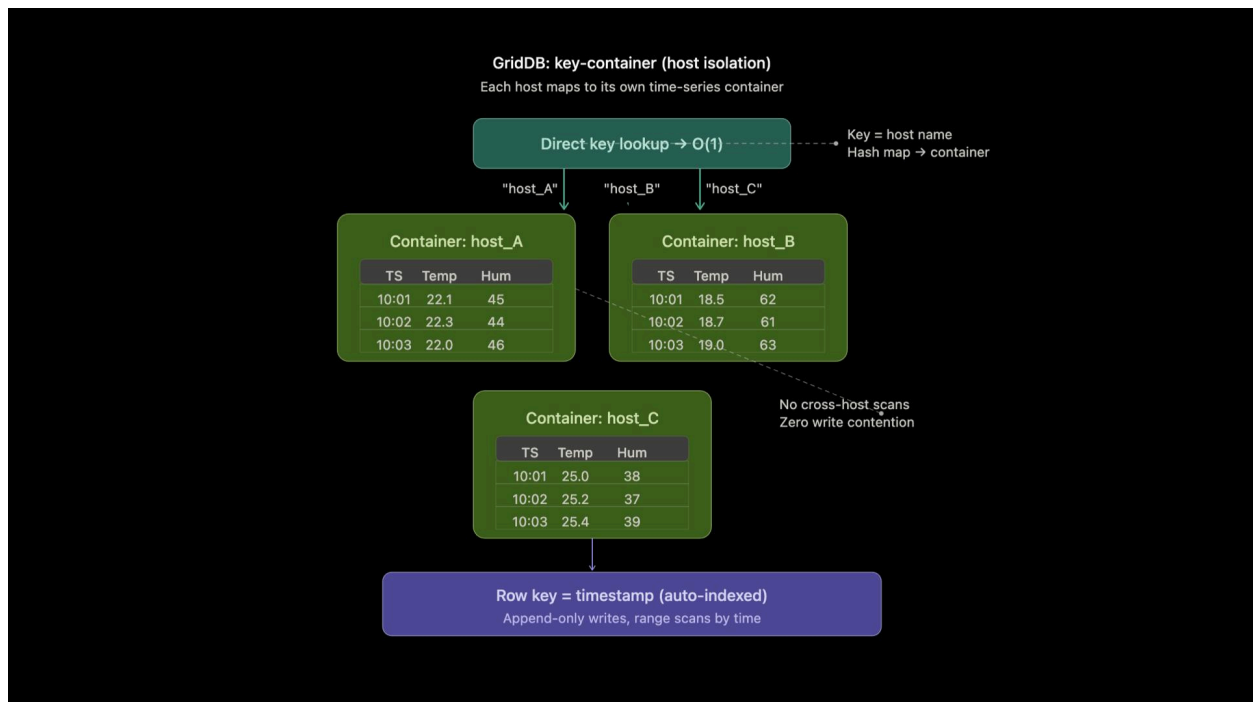
- GridDB Cloud: Custom Java implementation via the native GridDB Java client.
- MongoDB Atlas: MongoDB’s officially maintained TSBS fork, tailored for their time series collections.

Data Cardinality: To ensure a consistent baseline across the entire white paper, the dataset cardinality was maintained at 20 hosts for this section. While both GridDB Cloud and MongoDB Atlas M40 are capable of supporting significantly higher cardinality, holding this variable constant allows for a direct, "apples-to-apples" comparison against the baseline established in Section 1.

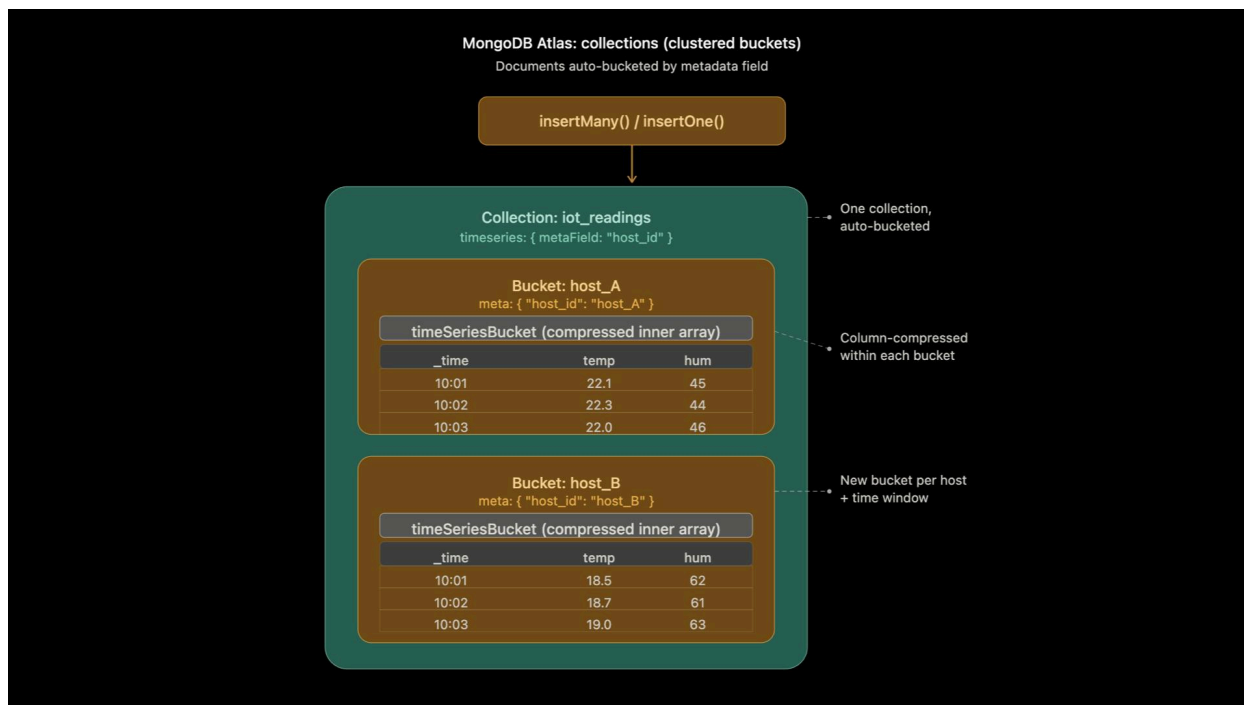
Data Modelling

The data model differences between GridDB and MongoDB are particularly relevant at higher concurrency levels:

- GridDB Cloud (Key-Container Model): Each host or device receives its own isolated container. Concurrent queries against multiple devices do not contend for the same table, allowing linear scaling as worker count increases.



- MongoDB Atlas (Time-Series Collections): MongoDB groups time-series data into clustered buckets based on time and a metadata field (e.g., host ID). While this improves compression and moderate-scale performance, the global collection structure means that high-concurrency queries must compete for shared index structures—contributing to the performance ceiling observed at 32+ workers.

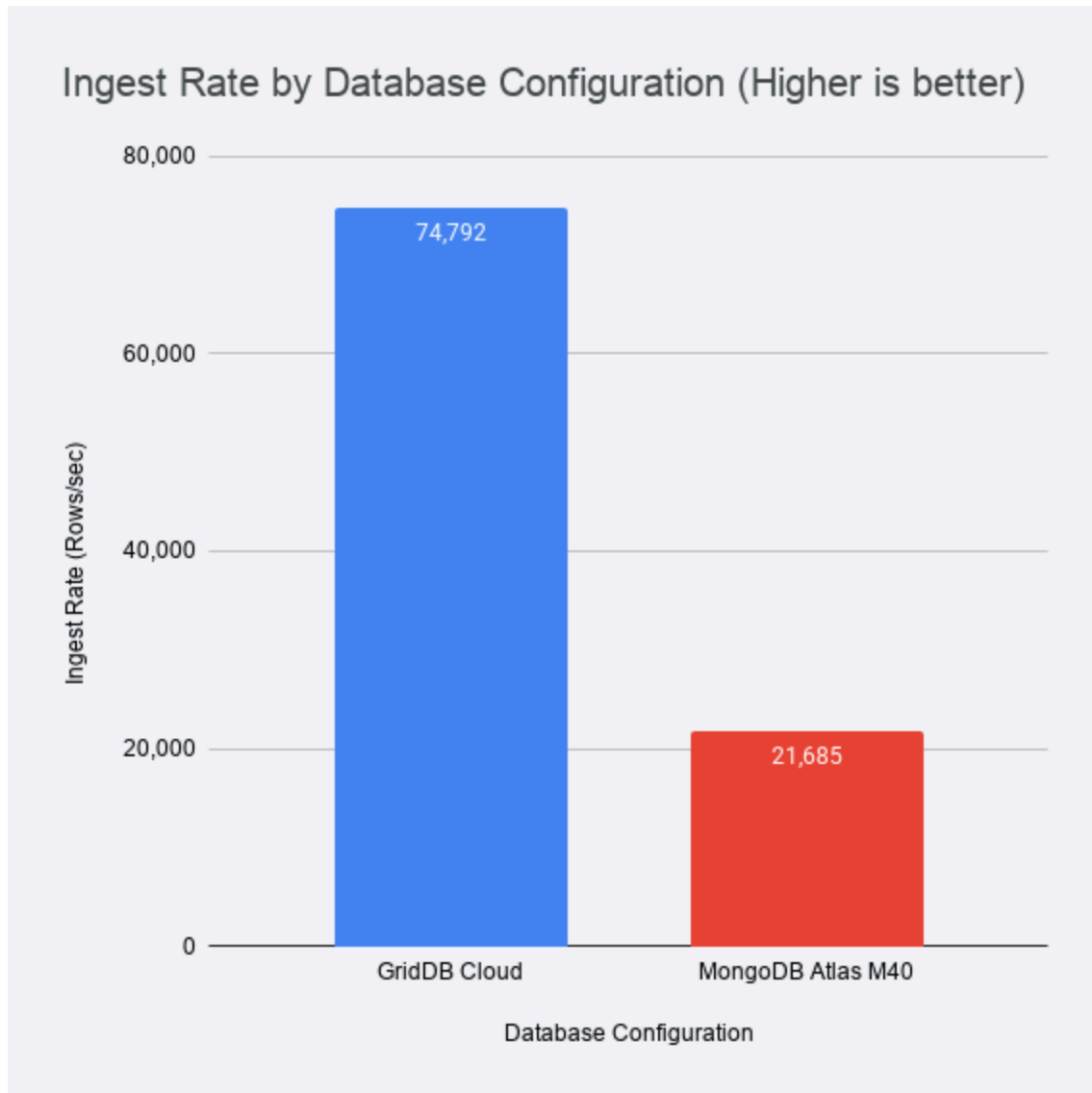


Load Performance (Data Ingestion)

Ingestion testing at this tier revealed a significant stability gap between the two platforms under high concurrency. To achieve comparable results, worker counts and batch sizes had to be adjusted differently for each database:

- **GridDB Cloud:** Workers were reduced to 16 (from higher counts tested) due to the VM running out of memory at peak throughput—a sign that GridDB was pushing the application server to its limits, not the database.
- **MongoDB Atlas M40:** To test at 16 workers, the batch size had to be drastically reduced to 5,000 rows. Attempting to ingest the standard 20,000-row batches at that concurrency level crashed the M40 cluster with out-of-memory errors. To establish an "apples-to-apples" baseline for how MongoDB handles the standard 20,000-row batch, concurrency had to be dropped all the way down to 2 workers. This is particularly notable given that the M40 is MongoDB's dedicated mid-tier cluster.

Database Engine	Workers	Batch Size	Ingest Rate (Rows/sec)	Ingest Rate (Metrics/sec)
GridDB Cloud (Java)	16	20,000	74,792	747,920
MongoDB Atlas M40	16	5,000	21,685	216,851
MongoDB Atlas M40	2	20,000	13,260	132,600



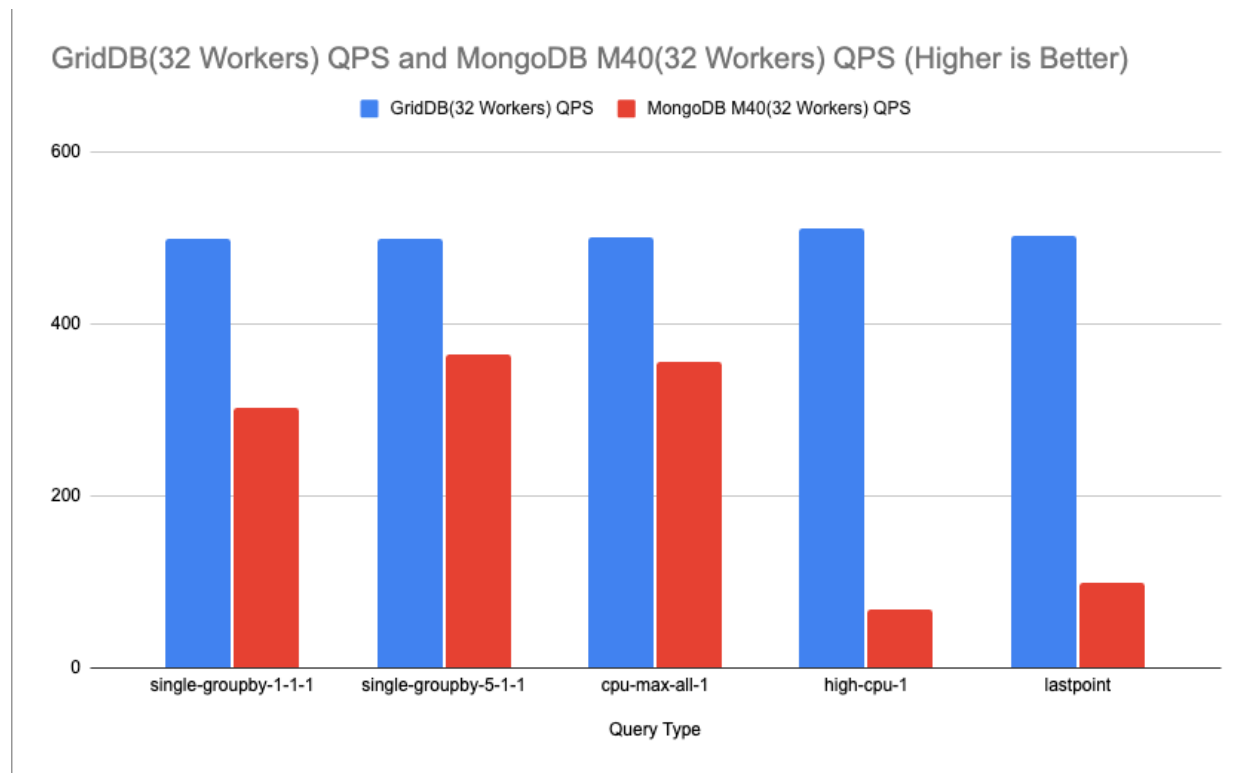
GridDB Cloud ingested data at 74,792 rows/sec—over 3.4x faster than MongoDB M40’s best result of 21,685 rows/sec. MongoDB’s instability at higher batch sizes (requiring a reduction to 5k rows to avoid crashing) reflects the same BSON format overhead issues observed with lower MongoDB tiers: verbose document encoding causes wire protocol messages to bloat in size, breaching hardcoded limits under concurrent high-volume loads.

Query Performance Evaluation

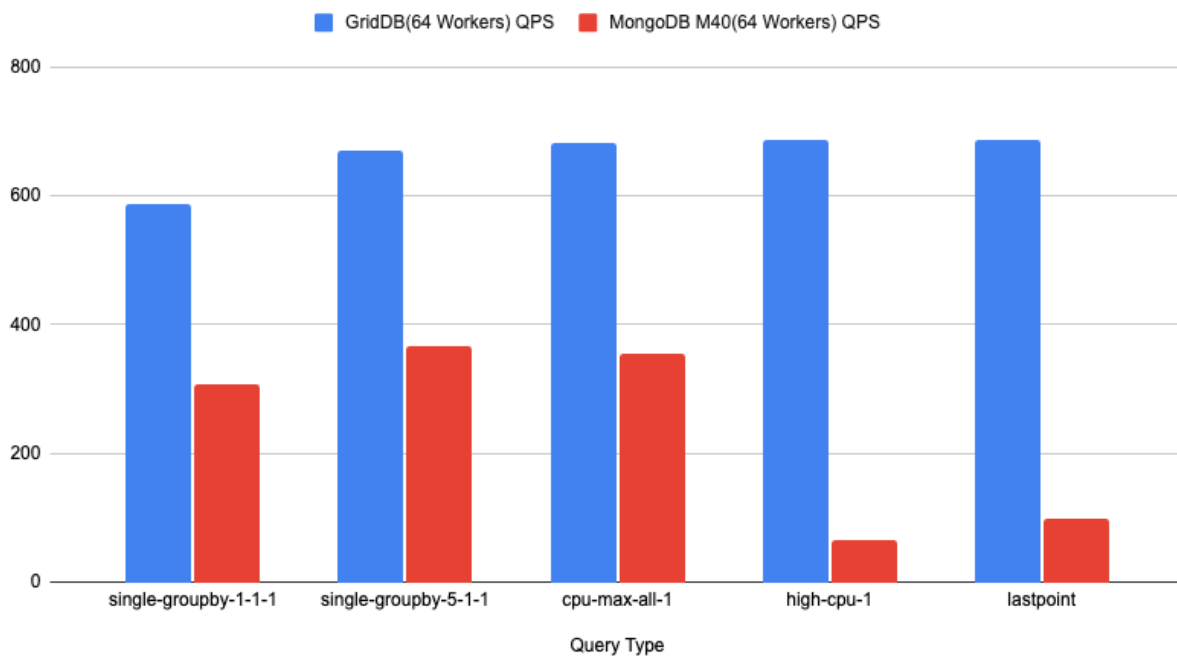
Query testing was conducted at significantly higher worker counts than in Section 1—32 and 64 concurrent workers—to evaluate how each database scales under enterprise-level concurrency. The same five curated TSBS query patterns were used (see Section 1 for query descriptions).

Query Results: GridDB Cloud vs. MongoDB Atlas M40

Query Type	MongoDB M40(32 Workers) QPS	GridDB(32 Workers) QPS	MongoDB M40(64 Workers) QPS	GridDB(64 Workers) QPS	GridDB Advantage (64 Workers)	Winner
single-groupby-1-1-1	302.41	498.50	308.38	586.17	~90% faster	GridDB
single-groupby-5-1-1	363.78	498.50	367.13	669.34	~82% faster	GridDB
cpu-max-all-1	355.01	501.76	354.27	682.59	~92% faster	GridDB
high-cpu-1	66.86	511.51	65.87	686.34	~10x faster	GridDB
lastpoint	98.88	502.26	98.03	685.87	~7x faster	GridDB



GridDB(64 Workers) QPS and MongoDB M40(64 Workers) QPS (Higher is Better)



Analysis: The results reveal two critical findings:

- GridDB scales with higher worker counts, MongoDB plateaus. As worker count increased from 32 to 64, GridDB's QPS continued to climb significantly across all query types—from 498 to 586 QPS on single-groupby-1-1-1, and from 511 to 686 QPS on high-cpu-1. MongoDB M40, by contrast, showed virtually no improvement between 32 and 64 workers, plateauing entirely. Scaling up MongoDB hardware did not unlock more concurrency.
- The gap widens on harder queries. On simple aggregation queries (single-groupby-1-1-1), MongoDB reaches 308 QPS at 64 workers—GridDB delivers 586 QPS, nearly double the throughput. On threshold-scanning queries (high-cpu-1) that require searching across a broader index, MongoDB drops to just 66 QPS while GridDB delivers 686 QPS—a 10x difference. This reflects the fundamental advantage of GridDB's Key-Container model: threshold queries are scoped to a single container rather than scanning a global collection.

Section 2 Conclusion

MongoDB Atlas M40 costs \$407.10/month more than GridDB Cloud's fixed plan—and delivers significantly worse performance across every benchmark. GridDB Cloud's fixed monthly commitment is the stronger choice for organizations needing enterprise-grade capacity with predictable billing.

For heavy, continuous workloads where pay-as-you-go costs would escalate, GridDB's Fixed Monthly Commitment at \$520/month is the obvious choice—not MongoDB M40. It caps costs at a flat rate that is over \$400/month cheaper than MongoDB M40, while maintaining every performance advantage demonstrated in this evaluation. There is no workload profile where MongoDB M40 wins on both price and performance against GridDB's fixed plan.

Overall Conclusion

This evaluation demonstrates that GridDB Cloud is the most capable and cost-flexible managed time series database across all scenarios tested. Its architecture—built around the Key-Container model—provides structural advantages that compound as workload complexity and concurrency increase.

	GridDB Cloud	InfluxDB Cloud	MongoDB Atlas M40
Pay-As-You-Go Price	\$0 base + usage	\$0 base + usage (identical)	N/A
Fixed Monthly Price	\$520/month	N/A	~\$927.70/month
Ingest Throughput	74,792 rows/sec	17,083 rows/sec	21,685 rows/sec
Query Scalability	Linear to 64+ workers	Rate-limited ~11.5 QPS	Plateaus at 32 workers
Batch Stability	100k row batches supported	Crashes at/above 100k rows	Crashes at/above 20k rows
Best For	All IoT / time series workloads	Low-frequency, low-volume monitoring	Teams already in MongoDB ecosystem

- GridDB Cloud (Pay-As-You-Go) is the clear recommendation for variable, bursty, and edge IoT workloads. Its \$0 base compute cost means you pay nothing when the system is idle, and its performance scales dramatically when you need it—outperforming InfluxDB at identical price points and MongoDB M40 at a fraction of the cost for low-to-medium data volumes.
- GridDB Cloud (Fixed Monthly Commitment) is the right choice for organizations with heavy, continuous workloads that require predictable billing. At \$520/month—\$407.10/month less than MongoDB M40—it delivers significantly superior performance, linear scalability to 64+ workers, and stable high-volume ingestion.
- InfluxDB Cloud’s pay-as-you-go tier suffers from severe API rate limiting that caps queries at ~11.5 QPS and restricts batch sizes to 5,000 rows. While its underlying engine is capable, these cloud-tier constraints make it difficult to recommend for production IoT workloads at this pricing tier.
- MongoDB Atlas M40 provides a familiar document-model entry point for teams already in the MongoDB ecosystem. However, its performance plateaus sharply under concurrent load, its ingest stability is poor at high batch sizes, and its \$927.10/month base cost is higher than GridDB’s fixed plan despite offering worse performance across every measured dimension.

The flexibility of GridDB Cloud’s dual pricing model—pay-as-you-go for lean and variable workloads, fixed commitment for heavy and continuous ones—means there is a GridDB Cloud option that outperforms the competition regardless of your operational profile.