CASE STUDY

Heap

Heap Takes Aim at the Enterprise with Real-time Performance and Big-Time Cost Reduction via SingleStore

$2M
annual cost reduction

10X
Can support 10X more data

60X
faster ingest

-25%COGS
Reduced cost of goods sold by 25%

<1sec
Sub-second average query duration

3M1B
3M events/sec; 1B events/mo.

-$5M2mo
$5M est. lost revenue in 2mo. on previous DB

"We were able to build more confidence in our ability to provide a platform we could grow with that could operate cost-effectively at scale. SingleStore struck the right balance of performance, cost, expressiveness, and partnership."

Molly Shelestak
Principal Product Manager, Heap

Heap is a digital insights platform that offers complete understanding of customers' digital journeys. With Heap, companies can use quantitative and qualitative insights to make data-informed decisions on how to improve their products, and thus quickly improve conversion, retention, and customer delight.

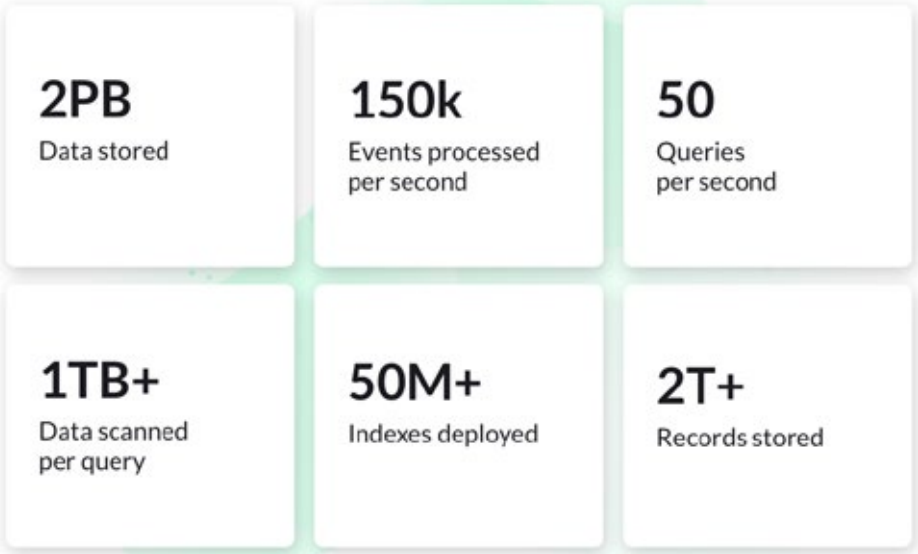

IMAGE | Heap by the Numbers

The platform provides interactive behavioral analytics, session replay, and managed ETL to the data warehouses of its customers including Crunchbase, E*TRADE, esurance, Freshworks, HelloSign, Lending Club, Northwestern Mutual, Redfin, and many more. "Unlike traditional analytics, where you have to make a tracking plan for specified interactions and require an engineer's time and expertise to implement that plan, Heap collects all of that data for you automatically," said James Katz, Staff Software Engineer, Heap.

For example, if you want to see how logins are changing over time, you can isolate login events — events that have a click on a button with text login — and with Heap you can quickly access the complete historical data set on it.

SingleStore

Challenges/Goals

Heap adds a very simple tag to its customers' web and mobile presence to track all customer behavior, and this creates petabytes of data. Heap supports true complex analytics queries; a query may scan billions of data records. The team runs a high compression ratio, 80-90%. In short, Heap is a demanding data product.

Heap was built on PostgreSQL with a CitusData layer on top of it to provide distributed PostgreSQL. "I want to be clear: we love PostgreSQL, but fundamentally we ran into problems with our legacy architecture because Postgres and Citus are designed to serve as transactional databases," said Katz, whereas Heap's core value is providing rapid analytical insights to its customers. The team worked around this with indexing strategies, but faced technical and business limitations for which there were no real workarounds:

- Coupled Storage and Compute = Limited Analytics. PostgreSQL couples storage and compute, which means, for example, that an operational workload running in the background must run on the same hardware and in the same database as queries its customers are running that are production-critical and customer experience-impact ing. As a result, for years Heap has not built new features and in fact, entire classes of analyses, such as account-based analysis, that it wanted to build. That has led to it losing existing revenue through customer churn and what it calculates as $5 million in new revenue in two recent quarters alone, as its data limitations prevented it from pursuing enterprise customers.

- Not Optimized for Shuffling A Lot of Data. The system was not optimized for shuffling large amounts of data, so the team built the system in a way that simply omitted features because those features would have required large shuffles.

- Indexing Helps Read/Hurts Write/Increases Cost. The team developed a unique indexing approach as a coping strategy, but as Katz noted, "Indexing is great on the Read path but comes at at a cost on the Write path: Write throughput has been a challenge the whole seven years I've been at Heap," and with tens of millions of indexes running across Heap, keeping them all up to date has been expensive.

- High Maintenance Effort and Cost. Heap has been spending almost $3 million per year on infrastructure costs alone, and that, coupled with high maintenance requirements, meant that its cost of goods sold (COGS) was hurting margins.

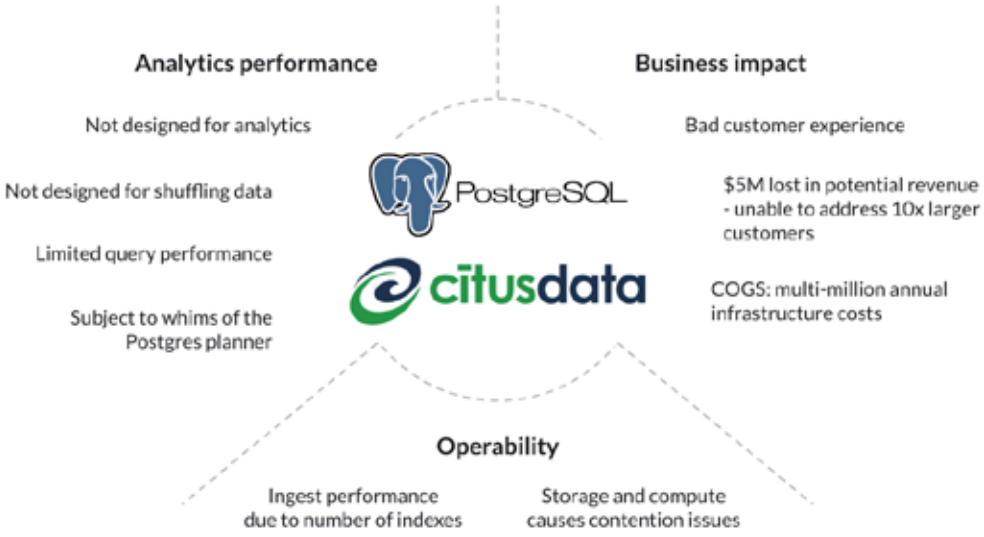

IMAGE | Heap's Core Drivers Toward a New Solution

Heap needed a next-generation database to meet its SLAs, support large enterprise customers, scale as they grow, and partner with them. Heap's data costs were growing linearly with event volume and size, and that does not translate directly into added value or revenue, so the company needed a cost-effective solution.

Technology Requirements

Heap wanted a more modern, columnar database that decoupled storage from compute, one designed for high-throughput analytics.

Core Specifications

- Data Type and Size: large, sparse JSON objects; 2+ petabytes
- Ingest: 500,000 rows per second
- 10X Scale Support: ability to support customers at 10X the existing size limitations; this translates into one billion events per month for a given environment and three million events per second at peak load
- Shuffle Ops-Ready: to execute those large-scale shuffle operations, resolve user identity at query time, and perform other non-user-based analyses such as account-based analysis, without negatively impacting overall system performance
- GDPR Compliant: Heap is moving into the EU market, with a new data center in-region, and the new system had to be GDPR-compliant to quickly provide data portability, data erasure/right to be forgotten requests, and more

Query Requirements

- Query Speeds: 3-9 seconds on funnel queries
- End-to-End Latency: from data creation to queryability in <5 minutes
- Concurrency: 40 concurrent queries at peak
- P95 Response: 9000ms (9sec) required, 3000ms (3sec) target
- 10X Scale Concurrency: 100 concurrent queries at peak
- 10X Scale P95 Response: 15000ms (15sec) required, 9000ms (9sec) target

"Expressiveness was key, to handle all of those complex queries and also to enable us to build out new features," said Molly Shelestak, Principal Product Manager, Heap. "Developer experience was essential: we wanted to make it easy for any developer to write new types of analyses."

Performance is challenging as you collect more and more data, and Heap needed to be able to provide sub-second query response even as it scaled the system to support its planned 10X data growth.

Step 1:

```
WITH step0 AS (
   SELECT user_id, min(time), 0 AS
funnel_step
   FROM etrade_full
   WHERE time BETWEEN '2020-03-01' AND
'2020-03-30'
   and event_0 = 1
   GROUP BY user_id
),
```

Step 2:

```
step1 AS (
      SELECT user_id, time, 1 AS funnel_step
      FROM etrade_full
      WHERE time BETWEEN '2020-03-01' AND
'2020-03-30'
      AND event_1 = 1
   ),
   step2 AS (
      SELECT user_id, time, 2 AS funnel_step
      FROM etrade_full
      WHERE time BETWEEN '2020-03-01' AND
'2020-03-30'
      and event_2 = 1
   ),
   step3 AS (
      SELECT user_id, time, 3 AS funnel_step
      FROM etrade_full
      WHERE time BETWEEN '2020-03-01' AND
'2020-03-30'
      and event_3 = 1
   ),
   step4 AS (
      SELECT user_id, time, 4 AS funnel_step
      FROM etrade_full
      WHERE time BETWEEN '2020-03-01' AND
'2020-03-30'
      and event_4 = 1
   )
```

Step 3:

```
SELECT
   count(DISTINCT s0.user_id) AS step0,
   count(DISTINCT s1.user_id) AS step1,
   count(DISTINCT s2.user_id) AS step2,
   count(DISTINCT s3.user_id) AS step3,
   count(DISTINCT s4.user_id) AS step4
FROM step0 s0
LEFT JOIN with(no_movement=true) step1 s1 ON
(s0.user_id = s1.user_id)
LEFT JOIN with(no_movement=true) step2 s2 ON
(s1.user_id = s2.user_id AND s1.time IS NOT NULL
AND s1.time < s2.time)
LEFT JOIN with(no_movement=true) step3 s3 ON
(s2.user_id = s3.user_id AND s2.time IS NOT NULL
AND s2.time < s3.time)
LEFT JOIN with(no_movement=true) step4 s4 ON
(s3.user_id = s4.user_id AND s3.time IS NOT NULL
AND s3.time < s4.time);
```

IMAGE | Example of a Funnel Query in Heap

Why SingleStore

Heap undertook an extremely thorough evaluation of a dozen databases. A number of them were knocked out immediately because like PostgreSQL, they couple compute and storage. “Others solely offer a managed service,” said Shelestak, and Heap needs to be able to host its own solutions. Some fell short because they cannot support performant shuffle operations; others because they offer no UDF support.

“A lot of OLAP data warehouses are designed with traditional OLAP workloads in mind, where everything is denormalized into a single table and you can follow the MapReduce paradigm. If you can confine your workloads to these types of queries, these systems can do ok,” said Katz, “but they are not capable of supporting distributed joins, and/or they deliver poor performance when it comes to things like repartitioning.”

Since complex analyses can be tough to express performantly in SQL, Heap needed the ability to put procedural code in the database and run it from a SQL interface. Some analytics systems couldn’t do that at all, or could but it was a huge hassle to try to implement.

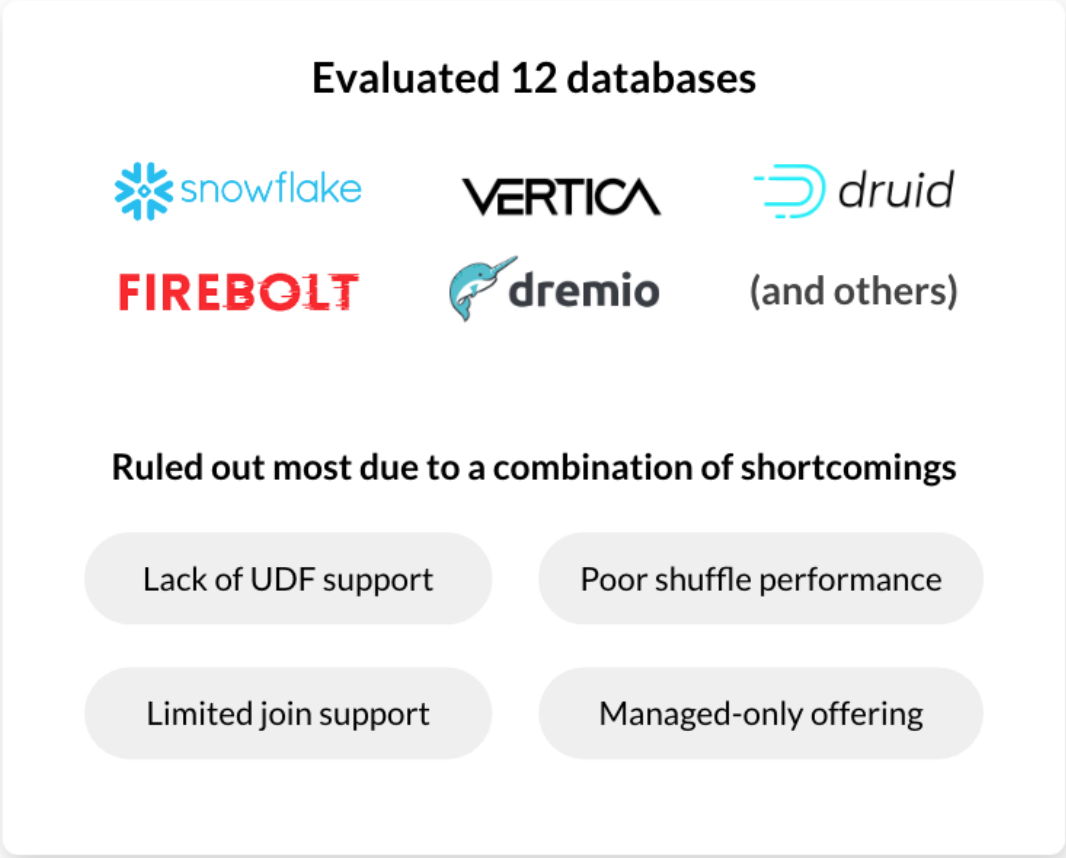

IMAGE | Heap's Database Selection Process

There was also the issue of cost.

"We store a ton of data, our AWS bill is probably our single biggest cost component, and the pricing of some of these other tools, particularly managed services, was going to hurt margin and force us to raise prices," explained Katz.

And Then There Were Two: SingleStore and Snowflake

When the dust cleared, only two remained: SingleStore and Snowflake. Snowflake performed well on many queries, but its support for JSON was not strong enough and its ingest was not fast enough. Meanwhile, "SingleStore's performance shone through in the PoC," said Katz.

The extensive POC was led on the SingleStore side by Sarung Tripathi, Global Head of Presales, and Cezary Lazowski, Manager, Solution Engineering, and many Product and Engineering team leaders including Eric Hanson, Director of Product Management, worked alongside the Heap team to stand up its build. "SingleStore was a partner throughout the whole process, starting with the Slack channel, config suggestions, weekly calls, and more," said Katz.

"SingleStore delivered 10X better performance than our current system," added Shelestak, "and with SingleStore we can execute distributed joins so we can resolve identity on Read and a range of non-user analyses."

SingleStore Ticked All the Tech Boxes...

In the words of the Heap team, SingleStore ticked all the boxes, including:

- Columnstore, which is the default table type in SingleStoreDB.
- Decoupled storage and compute with its Unlimited Storage (Bottomless)
- Fast ingest
- Low latency analytical queries
- Seekable JSON support for analytics on sparse objects
- Support for 10X data size

"SingleStore had performed better than competitive databases in previous testing," said Katz. "Its underlying primitives are blazing-fast. Then with Bottomless Storage it solved our main structural issue" (of coupled storage and compute).

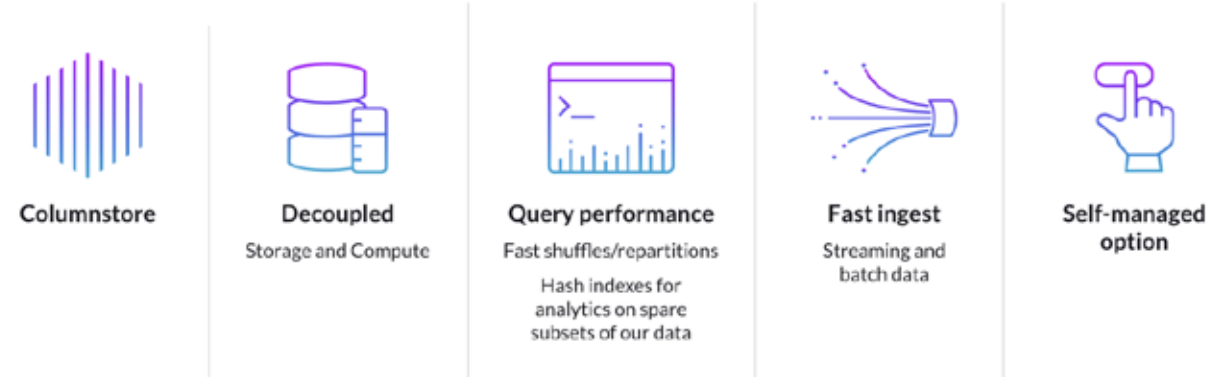

IMAGE | Why Heap Chose SingleStore

"SingleStore had performed better than competitive databases in previous testing. Its underlying primitives are blazing-fast. Then with Bottomless Storage it solved our main structural issue" (of coupled storage and compute). — *James Katz, Staff Software Engineer, Heap*

...and Closed the Deal with Cost-Effective Performance

Heap found Snowflake too expensive. Its pricing model was opaque and it was hard for the team to get an idea of what cost would look like. "We had little control over those costs, so we didn't feel confident we could build a cost-effective solution right out of the gate," said Shelestak, much less at (10X) scale.

"We were able to build more confidence in our ability to provide a platform we could grow with that could operate cost-effectively at scale for those large-data-volume customers," she added. "In the end, SingleStore struck the right balance of performance, cost, expressiveness, and partnership."

Solution

Heap is now built on SingleStoreDB Self-Managed running on AWS. SingleStore is the query layer on top of Heap's data system and the data lives in SingleStore. When a customer runs a query from Heap, the App sends a JSON payload to the App Server, the App Server compiles to SQL and fires it off to SingleStore, the results come back, the App server post-processes and sends it back to the Front End.

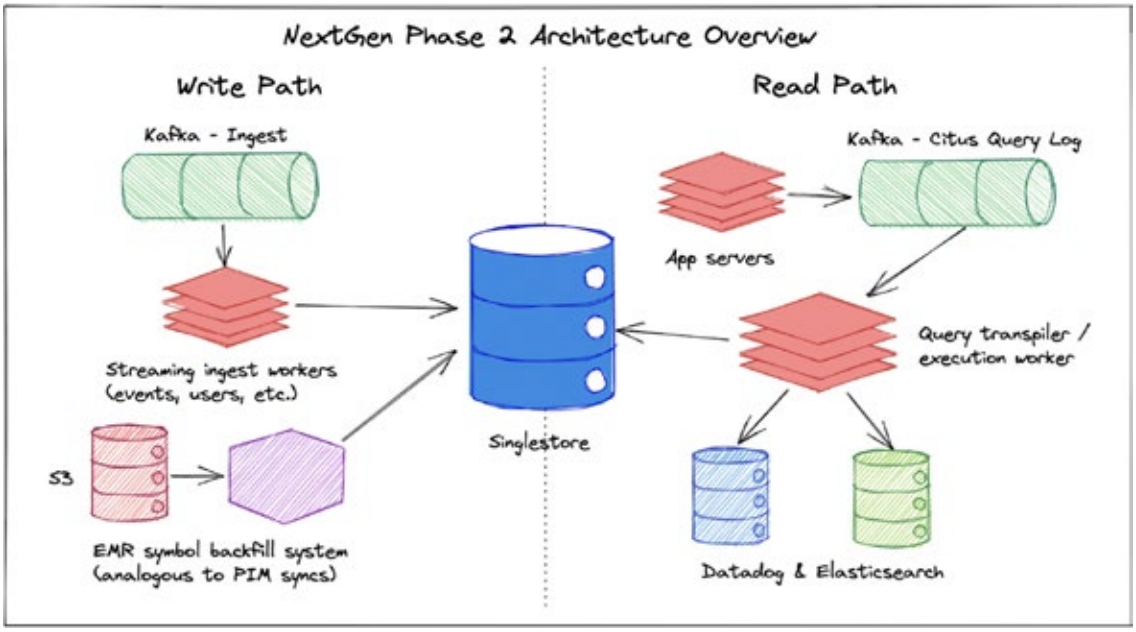

IMAGE | Heap NextGen Phase 2 Architecture Overview

As shown in Heap's High-Level End State Architecture, Heap is temporarily running a bifurcated setup as it closes in on fully migrating to SingleStore. Before, PostgreSQL was both the Query Layer and Fundamental Durability Layer; now a shared Data Lake is the shared Durability Layer and SingleStore is the Query Layer serving all interactive queries because they must run fast.

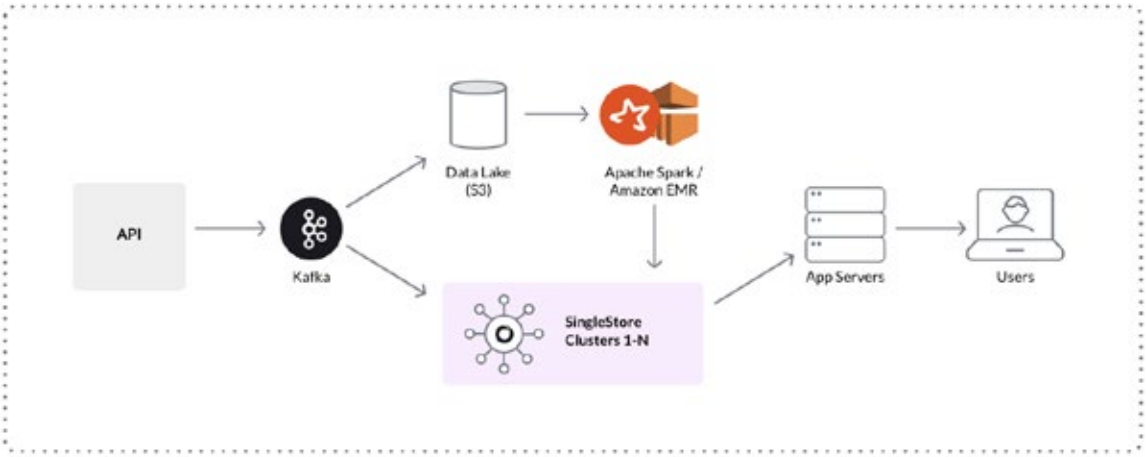

IMAGE | Heap's High-Level End State Architecture

Customers create some definitions, e.g., "This particular slice of my data is interesting," that Heap cannot serve quickly with hash indexes because they require Regex or something else that would run slowly. So Katz and team are now routing those requests through a Materialization system that "kicks off a Spark job to get all the relevant records from our Data Lake and ram them into SingleStore," he said, to benefit from SingleStore's blazing-fast ingest throughput. "This also illustrates the versatility of being able to pull data from both Kafka and AWS S3."

Heap is migrating its multi-petabyte data set from its Data Lake into SingleStore, cutting over customer queries that are still Data Lake-based. After a few more months in this dual mode, the team expects to be totally migrated to SingleStore. "SingleStore has made it super-easy to switch to a multi-cluster model," explained Katz. With PostgreSQL, Heap has had one big cluster that served nearly all of its customers, but switching to multi-cluster:

- "Limits the blast radius if one customer is running a lot of really expensive queries and causing contention issues," he added, and
- Accelerates Heap's expansion to the EU by letting it create clusters in-region for customers who need to store their data solely in the EU

The Heap team noted another SingleStore feature that makes its design work: "SingleStore positions its hash indexes as a transactional feature, but they are incredibly useful for us on the analytical side," Katz pointed out. "If a customer says they need to find, for example, Checkout events: if we have to scan 50 billion records it won't meet our performance requirements, so in this case we only scan the subset of rows matching that index."

"SingleStore positions its hash indexes as a transactional feature, but they are incredibly useful for us on the analytical side." — *James Katz, Staff Software Engineer, Heap*

Heap commended SingleStore for its ongoing partnership from day one, providing guidance during the initial phase, helping debug issues, being on call to triage issues during off-hours. Now in the data migration phase, "They continue as incredible partners, including building features to enhance some of our features including our JSON workloads," said Katz. "SingleStore Support has been great, for example, when we ask, 'We have migration coming up, can you have someone around later,' and they've been there for us."

Heap has deployed SingleStore on a 40-node cluster of i4i.8XL (32 vCPU 256 GB RAM) machines custom-designed by AWS to provide high I/O performance, low latency, minimal latency variability, and security with always-on encryption.

Outcomes

As shown in the adjacent table and discussed throughout this story, Heap's expertise and savvy as a data innovator is now delivering success after success:

Technical impact	**Fast data ingest** latency from 1 hour to 1 minute during peak hours	**Low latency queries** average query duration under 1 second	**Improved operability** decoupled storage / compute simplifies operations
Business impact	**New product innovation** unlocked 10+ new query types	**25% COGS reduction** saving millions on our analytics system	**Unlocked new TAM** now able to sell to 10x larger customers

IMAGE | Heap Results with SingleStore

- $2 million in annual cost-savings
- The ability to support 10X more data so it can now safely and effectively sell to the large enterprise customers that were beyond its reach before
- A 25% reduction in cost of goods sold, which, like the $2 million, goes directly to the bottom line
- 60X faster data ingest during peak hours, from 1 hour to 1 minute
- Sub-second average query duration
- Successfully handling three million events per second and a billion events per month

Supporting 10X more data also means Heap can be confident that its days of losing millions in potential revenue every few months because its legacy infrastructure wasn't up to the job are over. Heap has future-proofed its platform with scalability and greater predictability.

"Teams are already unblocked and working on new types of analyses to support our customers," said Shelestak. "We can now provide a cost-effective solution for our larger customers, a new COGs model, and new pricing: across the board we're doing a lot better with SingleStore." Heap is also looking at SingleStore as a replacement for its existing Apache Flink solution for stream and batch processing.

Heap is recognized globally for its quality of insights that help companies optimize their products, and it now has a data infrastructure that supports its enterprise vision. Its customers, who can now access the real-time insights they need to compete in today's economy are happier, and that means higher NPS Scores for Heap. As Heap's high-performance reputation precedes it in the market, that also improves close rates for the Heap sales force, which is valuable at any time but particularly with Heap making its move into the EU market and supporting that new data center.

SingleStore Fast Fact

Heap is one of the core customers that inspired SingleStore to double/triple down on JSON. The JSON enhancements in the SingleStore 8.0 Release were based in part on Heap's workload.

COMING SOON Watch James Katz describe how Heap is succeeding with SingleStore at the launch of our SingleStore 8.1 Release.

SingleStore is helping companies compete and win across every vertical. **Learn More >**