GIGAOM

William McKnight, Jake Dolezal
Jun 10, 2021

The Cost Savings of Replacing Kafka with Pulsar

Product Evaluation: DataStax Luna Streaming, a production-ready distribution of Apache Pulsar

The Cost Savings of Replacing Kafka with Pulsar

Product Evaluation: DataStax Luna Streaming, a production-ready distribution of Apache Pulsar

Table of Contents

1. Summary

With machine learning progressing into the enterprise and bringing with it the ability to utilize every bit of data possible, it is essential to process an increasing amount of data in real-time.

There are a number of applications being developed that make autonomous decisions about where data is produced, consumed, analyzed, and reacted to in real-time. The technology is making pragmatic, tactical decisions on its own as a result.

However, if data is not captured within a certain window of time, its value is lost and the decision or action that needs to take place as a result never occurs.

There are, fortunately, technologies designed to handle large volumes of time-sensitive, streaming data. Known by names like streaming, messaging, live feeds, real-time, and event-driven, this category of data needs special attention because delayed processing can negatively affect its value. A sudden price change, a critical threshold met, an anomaly detected, a sensor reading changing rapidly, an outlier in a log file—any of these can be of immense value to a decision-maker or a process, but only if alerted in time to affect the outcome.

The focus of this report is on real-time data and how autonomous systems can be fed at scale while producing reliable performance. To shed light on this challenge, we assess and benchmark two leading streaming data technologies—Apache Kafka^{TM} and Apache Pulsar^{TM}. Both solutions process massive amounts of streaming data generated from social media, logging systems, clickstreams, Internet-of-Things devices, and more. However, they also differ in important ways, including throughput and cost, which we uncover in our hands-on testing.

Our findings? In the three real-world scenarios we devised, we found that the production-ready distribution of Pulsar from DataStax with administration and monitoring tools and 24/7 enterprise support known as Luna Streaming produced higher average throughput and lower costs in all the testing workloads. Kafka and Pulsar are both streaming solutions. But with Kafka, you'll be challenged with throughput, which will result in a potential 81% greater overall cost. The results offer a compelling case for Pulsar/Luna Streaming to support organizations' growing streaming needs.

2. Differences Between Pulsar and Kafka

Inside Kafka

The most popular streaming data processor is the Apache Kafka project. Created by LinkedIn, Kafka was open-sourced and graduated from the Apache Incubator in late 2012. Kafka is a distributed publish-subscribe messaging system that maintains feeds of messages in groups, known as topics. Publishers write data to topics and subscribers read from topics. Kafka is a distributed system, where topics are partitioned and replicated across multiple nodes in the cluster.

Within Kafka, messages are simple byte arrays that can store objects in virtually any format—most often strings or JSON. A message is simply an array of bytes to Kafka. Kafka attaches a key to each message, so that all messages with the same key will arrive together within the same partition or be delivered to the same subscriber.

Kafka treats each topic partition as a log file where the messages are ordered by a unique offset. Kafka clients are users of the system, either producers or consumers. To be efficient and scalable, Kafka relies on consumers to track their location within each log; this design choice allows it to support a large number of users and retain huge amounts of data with little overhead.

Kafka has enterprise-supported deployments with Confluent and others. Cloudera provides a Kafka platform, as a part of Cloudera DataFlow, with ecosystem components and some productivity-boosting innovations for a complete Kafka implementation. This configuration includes shared compliance-ready security, governance, lineage, and control in one application across multiple environments.

Kafka is optimized for high volume and high performance, but there are some architectural drawbacks to the technology.

One of the biggest—and unfortunately very foundational—issues is the platform's monolithic architecture, which tightly couples the message-broker compute capabilities with the topic storage and persistence capabilities. Its partitions are forever tied to its nodes, which hinders customers' ability to lower costs with less expensive resources over time.

Kafka nodes can't easily be added or removed, which means customers frequently must size for peak loads.

In addition, Kafka was built fundamentally as a single-tenant architecture. There's no easy way to logically separate resources for different users of Kafka. You can't easily give users of a certain business unit free rein to manage just their resources in Kafka without introducing risks that those users may be able to impact others on the platform.

This fact, combined with the limits that Kafka has on the number of topics that can be handled by a cluster, almost always leads organizations to spin up a new Kafka cluster for each new business unit or department that wants to adopt it.

Now, take the inefficient resource utilization problem and multiply it by the number of teams using Kafka. With each cluster needing to be monitored, maintained, and administered, the administrative overhead and resulting costs get large as you scale Kafka.

Inside Pulsar

Pulsar takes a different tack to this problem.

Originally developed and utilized at Yahoo, Pulsar began its incubation at Apache in late 2016 and follows the same publisher-subscriber model (pub-sub) as Kafka. It also has the same producers, topics, and consumers. Pulsar uses built-in multi-datacenter replication and is architected for multi-tenancy, leveraging the concepts of tenants and namespaces. There is a hierarchy of a Pulsar cluster containing multiple tenants, which contain different namespaces that contain any number of topics. Pulsar also supports queuing, which allows for different use cases, as well as a migration path from legacy message queuing systems.

A tenant could represent all the messaging for a particular team, application, product vertical, and the like. Namespaces is the administrative unit where security, replication, configurations, and subscriptions are managed. At the topic level, messages are partitioned and managed by brokers using a user-defined routing policy—such as single, round robin, hash, or custom—thus granting further transparency and control in the process.

Additionally, Pulsar has a Kafka API compatibility interface—making it easier to port existing Kafka applications.

A key distinction of Pulsar over Kafka is how it handles message durability. Pulsar ensures that message data is never lost under any circumstances. Pulsar achieves this with Apache BookKeeper to provide low-latency persistent storage. When a Pulsar broker receives messages, it sends the message data to several BookKeeper nodes, which push the data into a write-ahead log and into memory even before an acknowledgement is sent. In the event of a hardware failure, power outage, network problem, or other disruption, Pulsar messages are safe-kept in permanent storage.

Pulsar has a decoupled architecture. This means that Pulsar can use high-performance local storage optimized for fast reads and writes to ingest messages, but it can also offload historical data to much cheaper options such as AWS S3.

With Pulsar decoupling the persistence and broker layers, you can easily scale the size of both independently. (Again, this includes cost-effective cloud storage options as well as local storage.) Pulsar is inherently multi-tenant, which means that when a new business unit or department wants to use Pulsar, you simply add a new logical tenant to the system. This means you can support a greater number of business units with a smaller number of clusters.

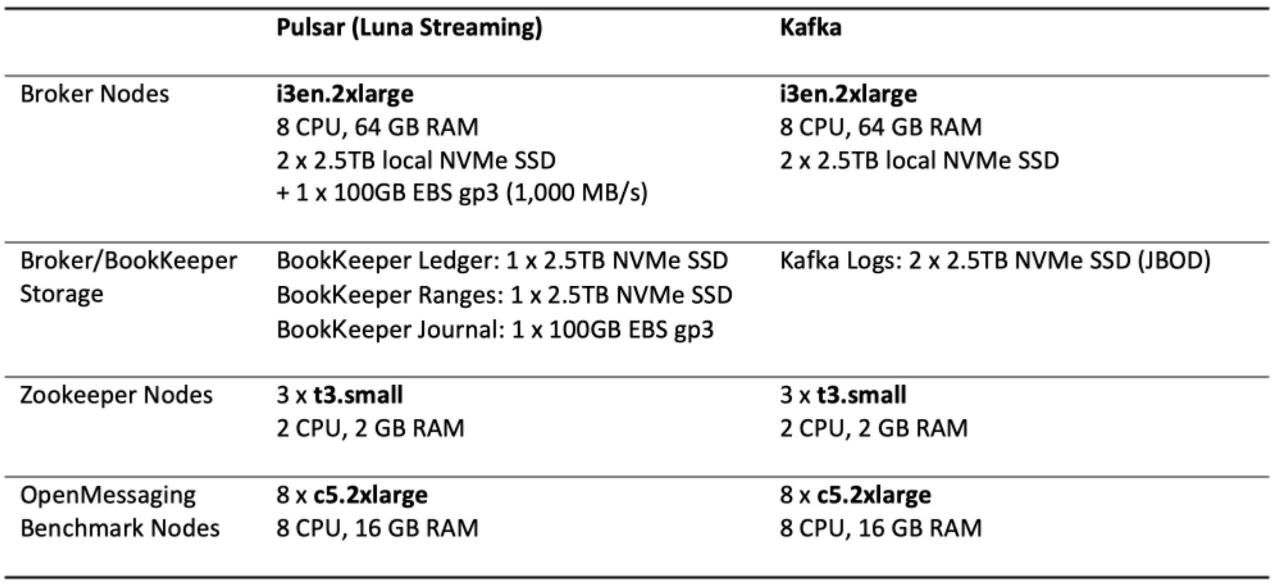
Pulsar architecture is built for the streaming enterprise future. It has decoupled storage and compute, on-demand and efficient scaling, a complete feature set, and multi-region and multi-tenant that are core to the platform. It is cloud-native and has tiered storage capabilities to enable message retention on less expensive stores.

3. Performance Test

For our performance testing, we deployed open-source Pulsar (Luna Streaming) and Apache Kafka clusters with Kubernetes. Our Kubernetes cluster was deployed onto Amazon Web Services EC2 instances using kOps. For the Luna Streaming deployment, we used the Datastax Quick Start Helm Charts. For the Kafka deployment, we used Strimzi.

We also employed the same hardware configuration of i3en.2xlarge EC2 instance types for Kafka brokers and to co-locate the Pulsar brokers and BookKeeper nodes to take advantage of the two large (2.5TB), fast, locally-attached NVMe solid-state drives. For Kafka, we spanned the persistent volume storage across both disks using JBOD. For Pulsar, we created persistent volumes and used one of the local drives for the BookKeeper ledger and the other for the ranges. For the BookKeeper journal, we provisioned a 100GB gp3 AWS Elastic Band Storage (EBS) volume with 4,000 IOPS and 1,000 MB/s throughput. Other than taking advantage of this storage configuration for both platforms, we performed no other specific tuning of either platform and preferred instead to go with their "out-of-the-box" configurations as they were deployed via their respective Docker images and Helm charts.

Table 1 shows a comparison in the two platforms and how we configured them.
Table 1. Platform Configurations Compared



	Pulsar (Luna Streaming)	**Kafka**
Broker Nodes	**i3en.2xlarge** 8 CPU, 64 GB RAM 2 x 2.5TB local NVMe SSD + 1 x 100GB EBS gp3 (1,000 MB/s)	**i3en.2xlarge** 8 CPU, 64 GB RAM 2 x 2.5TB local NVMe SSD
Broker/BookKeeper Storage	BookKeeper Ledger: 1 x 2.5TB NVMe SSD BookKeeper Ranges: 1 x 2.5TB NVMe SSD BookKeeper Journal: 1 x 100GB EBS gp3	Kafka Logs: 2 x 2.5TB NVMe SSD (JBOD)
Zookeeper Nodes	3 x **t3.small** 2 CPU, 2 GB RAM	3 x **t3.small** 2 CPU, 2 GB RAM
OpenMessaging Benchmark Nodes	8 x **c5.2xlarge** 8 CPU, 16 GB RAM	8 x **c5.2xlarge** 8 CPU, 16 GB RAM

To conduct the tests, we used the OpenMessaging Benchmark (OMB) test harness. We used the Confluent fork of the OpenMessaging Benchmark on GitHub, because Confluent published some fixes to the original harness to improve the performance of Kafka and Pulsar (Luna Streaming) it used in some of its testing in 2020.

Performance Test Results

Our performance testing revealed Pulsar (Luna Streaming) had a higher average throughput in all the OMB testing workloads we performed.

The following chart shows how we arrived at these results. With a three-broker node configuration, Luna Streaming achieved an average of 104 MB/s throughput, while Kafka only had an average of 65 MB/s. With six broker nodes, Luna averaged 172 MB/s to Kafka's 125 MB/s. With nine broker nodes each, Luna averaged 306 MB/s, while Kafka only averaged 210 MB/s. Since Luna Streaming showed higher throughput in all the head-to-head broker count tests we ran, we sought to answer another question: What would the equivalent Kafka broker node count be compared to Luna Streaming?

We continued to add node groups of three to the Kafka configuration until we achieved better throughput than Luna Streaming's nine-broker node performance. This occurred at 15 Kafka broker nodes. Now we had the information we needed to make our approximation.

Using the chart below and extrapolating trend lines, we found the following in terms of broker node equivalence:

3 Luna Streaming broker nodes ≅ 5 Kafka broker nodes
6 Luna Streaming broker nodes ≅ 8 Kafka broker nodes
9 Luna Streaming broker nodes ≅ 14 Kafka broker nodes

The chart in **Figure 1** shows the results. Our goal was to build a cost scenario around throughput performance needs. While this is certainly just an approximation of performance equivalence and not exact, it still sufficed because you obviously would not configure a fractional broker node—even in a containerized environment.

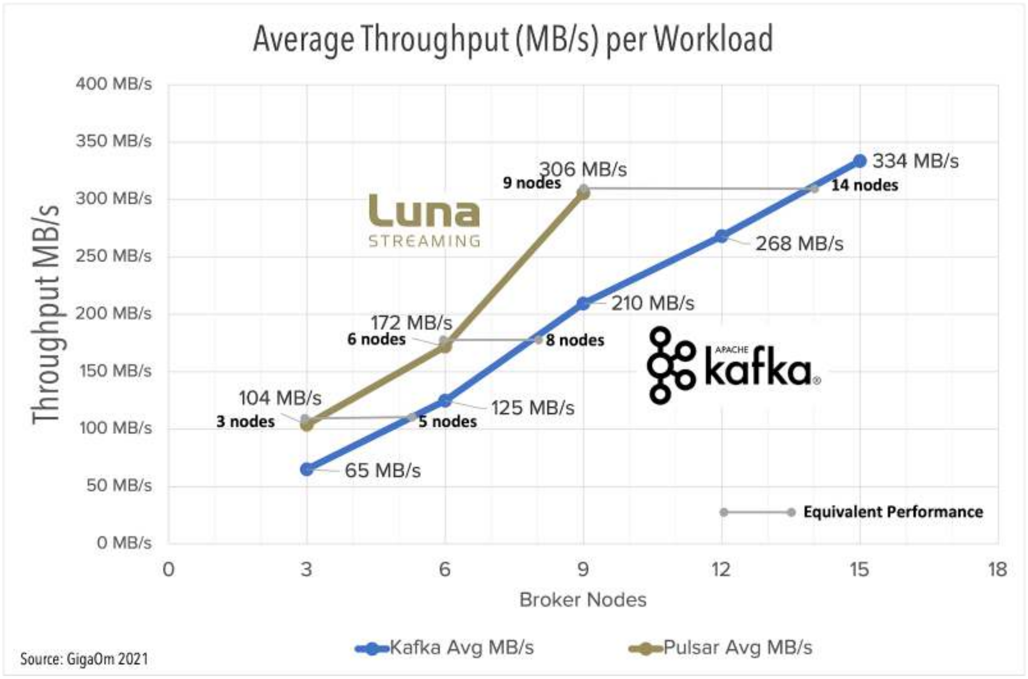


Figure 1. Broker Node Equivalence

Note: For Luna Streaming, we co-located the Pulsar broker pod and BookKeeper pod on the same i3en.2xlarge node.

4. Cost Scenarios

While these architectural and performance results are insightful, what does it all mean in terms of the bottom-line costs of deploying Pulsar (Luna Streaming) versus Kafka? We found three scenarios where choosing Luna Streaming over Kafka would produce significant cost savings.

For our cost scenarios, we consider the following AWS infrastructure costs (US West 2 region, taken at the time of testing). **Table 2** shows the breakout:

Table 2. Comparing Cost and Configuration

Broker Node	i3en.2xlarge	$0.904 per hour per node
ZooKeeper	t3.small	$0.0208 per hour per node
BookKeeper Journal Disk (Pulsar only)	gp3 100GB-4,000 IOPS-1,000 MB/s	$576 per year per node

NOTE: With the EC2 i3en.2xlarge nodes, the locally-attached NVMe SSD disks we used for broker/ BookKeeper storage are included in the hourly instance costs.

Cost Scenario 1: High Throughput

In our first scenario, we cover enterprises with high-throughput needs. By “high throughput,” we refer to enterprises that need to process from 100 MB/s to 300+ MB/s—either due to a high number of messages per second (100,000+ msg/s), relatively large per-message payload sizes (1 KB and up), or a combination of both. For this cost scenario, we can draw directly from our performance test results, again, where we found:

3 Luna Streaming broker nodes ≅ 5 Kafka broker nodes

6 Luna Streaming broker nodes ≅ 8 Kafka broker nodes

9 Luna Streaming broker nodes ≅ 14 Kafka broker nodes

Table 3 considers an enterprise’s streaming data needs over time using a “small” cluster (3x Luna or 5x Kafka) in Year 1, a “medium” cluster in Year 2 (6x Luna or 8x Kafka), and a “large” cluster (9x Luna or 14x Kafka) in Year 3.

Table 3. High-Throughput Cost Scenario

~100 MB/s Throughput	Brokers	Per Year	Savings
Luna Streaming	3	$26,031.74	35%
Kafka	5	$40,141.82	
~175 MB/s Throughput	Brokers	Per Year	**Savings**
Luna Streaming	6	$51,516.86	19%
Kafka	8	$63,898.94	
~300 MB/s Throughput	Brokers	Per Year	**Savings**
Luna Streaming	9	$77,001.98	31%
Kafka	14	$111,413.18	

Using the node equivalencies found in our testing above, Pulsar (Luna Streaming) produces significant infrastructure cost savings over Kafka, ranging from 19% to 35%.

Cost Scenario 2: High Complexity

For our second cost scenario, we considered another architectural difference between Luna Streaming and Kafka that may also have significant cost implications. Instead of raw throughput requirements, some companies may need to support complex environments with a high number of topics and partitions to handle the wide range of needs across the enterprise.

For example, imagine the not too uncommon use case of 10,000 total partitions across a large number of topics. This happens when an enterprise uses its messaging platform across multiple lines of business and for many different processes, operations, and applications. As an organization gets more sophisticated with its messaging uses, it will require more topics. As topics grow, they require more and partitions. Thus, over time the partition counts can become quite high.

Kafka has limitations when it comes to the number of partitions it can feasibly handle per broker. According to Confluent, the partition count per broker should not exceed 100 x *b* x *r*, where *b* is the number of brokers in a Kafka cluster and *r* is the replication factor. In our test case of 3 broker nodes for Kafka and Pulsar (Luna Streaming), this would be 100 x 3 brokers x 3 replication factor to yield 900 partitions per broker. The way Pulsar is architected, it does not matter how many partitions or topics you have per cluster. Therefore, to achieve 10,000 partitions, you would need at least 12 Kafka brokers, compared to the 3 Pulsar broker nodes.

This cost scenario would look like **Table 4**:

Table 4. High Complexity Cost Scenario

10,000+ partitions	Brokers	Per Year	**Savings**
Luna Streaming	3	$26,031.74	**3.7x**
Kafka	12	$95,575.10	**73%**

Using the scenario of 10,000 partitions across all the messaging platform's topics, an enterprise employing Kafka would spend 3.7x more than the same enterprise working with Luna Streaming. From the perspective of the enterprise using Luna Streaming over Kafka, the difference yields a substantial cost savings of 73%.

Cost Scenario 3: High Data Volume

For our third scenario, we consider yet another architectural difference between Pulsar and Kafka that can lead to significant cost savings. Consider an enterprise that processes a large amount of message data per year—upwards of 10TB.

Pulsar has the ability to offload ledger data to cold storage (such as Amazon S3), a feature that Kafka lacks. If the Kafka logs fill up the available disk space on the broker nodes, the only solution is to either expand the disks or add more nodes. If additional nodes are required (as they would be in our performance testing, because the local NVMe SSDs we used on the i3en.2xlarge EC2 instances are fixed), then the Kafka architecture requires that the data be redistributed. This is a very CPU-intensive and long-running process, depending on the amount of data that must be redistributed. Thus, an enterprise processing large amounts of message data per year must also consider its long term storage solution—if it chooses Luna Streaming, it's easy: simply offload old messages to S3 or another low-cost storage solution.

For the calculation in Table 5, we considered the storage accumulation rate of 10TB per year and a Kafka replication factor of 3. In this case, we need six Kafka nodes compared to three for Luna Streaming. Note again that the i3en.2xlarge instances we chose have 5TB of combined NVMe SSD storage capacity. For Luna Streaming, we also figured in the cost of AWS S3 Standard storage rate of $0.023 GB-month. We calculated these figures to reflect the addition of six more Kafka broker nodes and re-distribution of the data at the end of each year.

Table 5. High Data Volume Cost Scenario

Year 1 (10TB stored)	Brokers	Per Year	Savings
Luna Streaming	3	$26,031.74	**1.8x**
Kafka	6	$48,060.86	**46%**
Year 2 (20TB stored)	Brokers	Per Year	**Savings**
Luna Streaming + S3 Storage	3	$26,267.26	**3.6x**
Kafka	12	$95,575.10	**73%**
Year 3 (30TB stored)	Brokers	Per Year	**Savings**
Luna Streaming + S3 Storage	3	$26,502.78	**5.4x**
Kafka	18	$143,089.34	**81%**

In the scenario of 10TB of message data processed per year, this would result in Luna Streaming producing infrastructure cost savings over Kafka of between 46% and 81% per year. Put another way, an enterprise using Kafka could expect to spend 1.8x to as much as 5.4x more than one using Luna Streaming.

Cost Differences

Ultimately we arrived at our final cost differences for the study. **Figure 2** shows the total for both deployments—Kafka and Luna Streaming—over a three-year span. Overall the cost differences were 81%, which we also broke down by category (**Figure 3**).

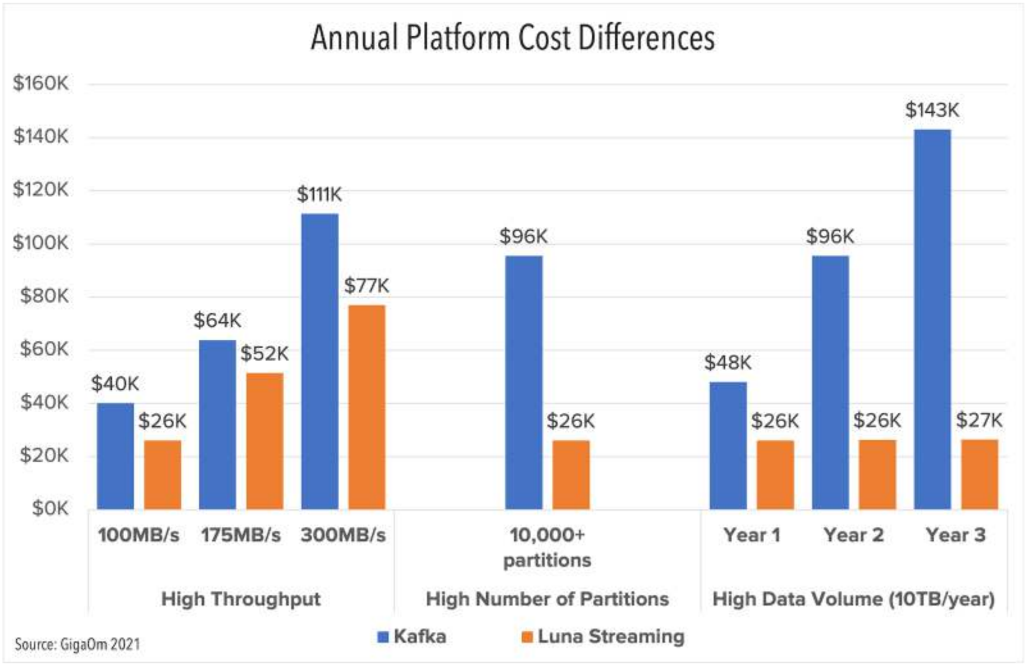


Figure 2: Kafka and Luna Streaming Cost Differences

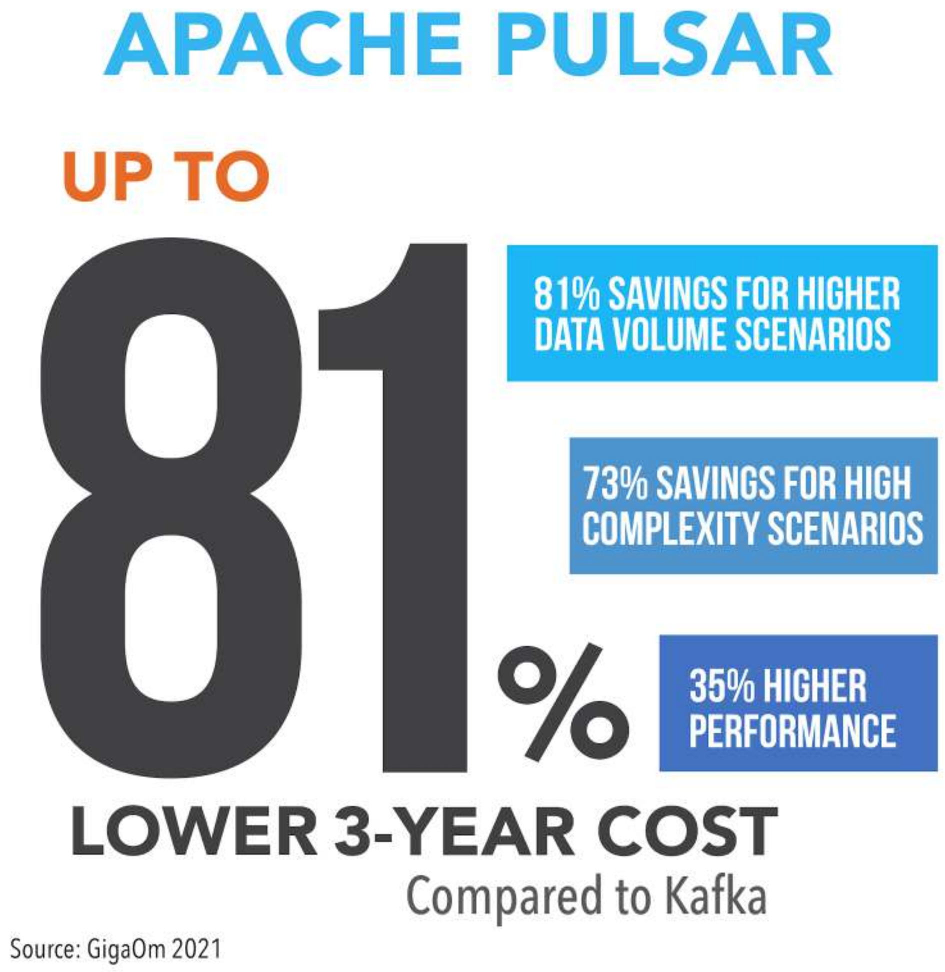


Figure 3: Apache Pulsar Compared to Kafka in Cost

5. Conclusion

For our performance testing, we deployed Luna Streaming (Pulsar) and Apache Kafka clusters with Kubernetes deployed onto Amazon Web Services EC2 instances. Other than taking advantage of storage configuration options for both platforms, we went with their “out-of-the-box” configurations.

With a three-broker node configuration, Luna Streaming achieved an average of 104 MB/s throughput, while Kafka averaged just 65 MB/s. With six broker nodes, Luna averaged 172 MB/s to Kafka’s 125 MB/s. With nine broker nodes each, Luna averaged 306 MB/s, while Kafka only averaged 210 MB/s.

In terms of broker node counts to achieve a near-parity of performance between the two platforms, we found that:

- Three Luna Streaming broker nodes was about equivalent to five Kafka broker nodes
- Six Luna Streaming broker nodes was about equivalent to eight Kafka broker nodes
- Nine Luna Streaming broker nodes was about equivalent to fourteen Kafka broker nodes.

In our three real-world scenarios we found that choosing Luna Streaming over Kafka yields significant cost savings. In a High Throughput test, using the node equivalences, we found between a 19% and 35% savings in infrastructure costs by using Luna Streaming instead of Kafka. In a High Complexity test, using a scenario of 10,000 partitions across all the messaging platform’s topics, we found a 73% savings in infrastructure costs by using Luna Streaming instead of Kafka. Finally, using a High Data Volume test with 10TB of message data processed per year resulted in between 46% and 81% savings per year in infrastructure costs by using Luna Streaming instead of Kafka.

Our performance testing revealed Luna Streaming had a higher average throughput and lower costs in all the testing workloads we performed. We believe this offers a compelling case for Lunar Streaming (Pulsar) to support your growing streaming needs. Kafka and Pulsar are both streaming solutions. But with Kafka, you’ll be challenged with throughput, which will result in a potential 81% greater overall cost.

6. About DataStax

DataStax delivers an open, multi-cloud data stack for modern data applications. The company's marquee offering is DataStax Astra, the industry's first and only open, multi-cloud serverless database. Built on proven Apache Cassandra™, Apache Pulsar™ streaming, and the Stargate open source API platform, Astra delivers Cassandra-as-a-service and Pulsar-as-a-service with an unprecedented combination of pay-as-you-go data, simplified operations, and the freedom of multi-cloud and open source.

With DataStax, any developer or enterprise can now use data, both in motion and at rest, at massive scale, with 100 percent uptime, for lower cost. Today, nearly 500 of the world's most demanding enterprises and half of the Fortune 100 rely on DataStax to power modern data apps, including The Home Depot, T-Mobile, US Bank, and Intuit.

7. Disclaimer

Cost is important but it is only one criterion for a platform selection. This test is a point-in-time check into specific costs. There are numerous other factors to consider in selection across factors of Performance, Administration, Features and Functionality, Workload Management, User Interface, Scalability, Vendor, Reliability, and numerous other criteria. It is also our experience that costs change over time and are competitively different for different workloads. Also, a cost leader can hit up against the point of diminishing returns and viable contenders can quickly close the gap.

GigaOm runs all of its tests to strict ethical standards. The results of the report are the objective results of the application of tests to the simulations described in the report. The report clearly defines the selected criteria and process used to establish the field test. The report also clearly states the tools and workloads used. The reader is left to determine for themselves how to qualify the information for their individual needs. The report does not make any claim regarding the third-party certification and presents the objective results received from the application of the process to the criteria as described in the report. The report strictly measures costs and does not purport to evaluate other factors that potential customers may find relevant when making a purchase decision.

This is a sponsored report. DataStax chose the competitors, the test, and the Luna Streaming configuration was the default provisioned by DataStax. GigaOm chose the most compatible configurations for Kafka and ran the tests. Choosing compatible configurations is subject to judgment. We have attempted to describe our decisions in this paper.

8 About William McKnight

William McKnight is a former Fortune 50 technology executive and database engineer. An Ernst & Young Entrepreneur of the Year finalist and frequent best practices judge, he helps enterprise clients with action plans, architectures, strategies, and technology tools to manage information.

Currently, William is an analyst for GigaOm Research who takes corporate information and turns it into a bottom-line-enhancing asset. He has worked with Dong Energy, France Telecom, Pfizer, Samba Bank, ScotiaBank, Teva Pharmaceuticals, and Verizon, among many others. William focuses on delivering business value and solving business problems utilizing proven approaches in information management.

9 About Jake Dolezal

Jake Dolezal is a contributing analyst at GigaOm. He has two decades of experience in the information management field, with expertise in analytics, data warehousing, master data management, data governance, business intelligence, statistics, data modeling and integration, and visualization. Jake has solved technical problems across a broad range of industries, including healthcare, education, government, manufacturing, engineering, hospitality, and restaurants. He has a doctorate in information management from Syracuse University.

10. About GigaOm

GigaOm provides technical, operational, and business advice for IT's strategic digital enterprise and business initiatives. Enterprise business leaders, CIOs, and technology organizations partner with GigaOm for practical, actionable, strategic, and visionary advice for modernizing and transforming their business. GigaOm's advice empowers enterprises to successfully compete in an increasingly complicated business atmosphere that requires a solid understanding of constantly changing customer demands.

GigaOm works directly with enterprises both inside and outside of the IT organization to apply proven research and methodologies designed to avoid pitfalls and roadblocks while balancing risk and innovation. Research methodologies include but are not limited to adoption and benchmarking surveys, use cases, interviews, ROI/TCO, market landscapes, strategic trends, and technical benchmarks. Our analysts possess 20+ years of experience advising a spectrum of clients from early adopters to mainstream enterprises.

GigaOm's perspective is that of the unbiased enterprise practitioner. Through this perspective, GigaOm connects with engaged and loyal subscribers on a deep and meaningful level.

11. Copyright