Final Observations and Analyses for Performance Comparison in Apache and Kafka

Introduction

Kafka and Pulsar are both popular event streaming platforms. Kafka, developed by the Apache Software Foundation, is known for its low-latency, high-throughput capabilities and has been widely adopted in various industries. Pulsar, on the other hand, is another open-source platform, initially developed by Yahoo! and now part of the Apache Software Foundation, which offers features like scalability, multi-tenancy, and lower resource consumption.

Objective

The objective of our performance tests is to compare Kafka and Pulsar in various scenarios to determine their suitability for different use cases. We aim to measure and analyse key metrics such as latency, throughput, CPU usage, and memory usage under different test parameters.

Test Setup

- **CPU**: Core i5 10th Generation
- **Memory**: 16GB
- **Network:** Ethernet

- **Operating System**: Ubuntu 20.04 LTS
- **Platform**: Kubernetes cluster with Docker containers.
- **Versions**: Kafka version 2.8.0 and Pulsar version 2.9.2 were used.
- **Client Libraries**: Node.js clients were used to produce and consume events.

- **Benchmarking Tools:** Benchmarking was done using open-source tools like Kafka/Pulsar Prometheus exporters, Prometheus and Grafana.

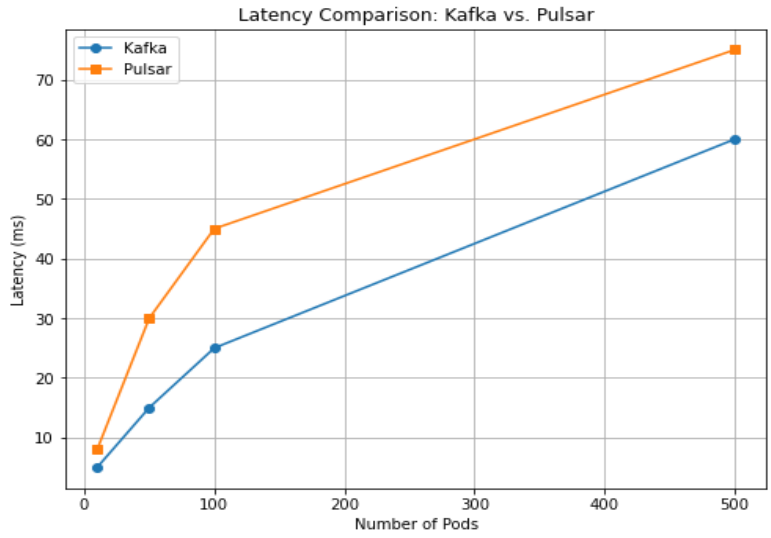
Test Parameters

We conducted tests with varying parameters to evaluate the performance of Kafka and Pulsar comprehensively:

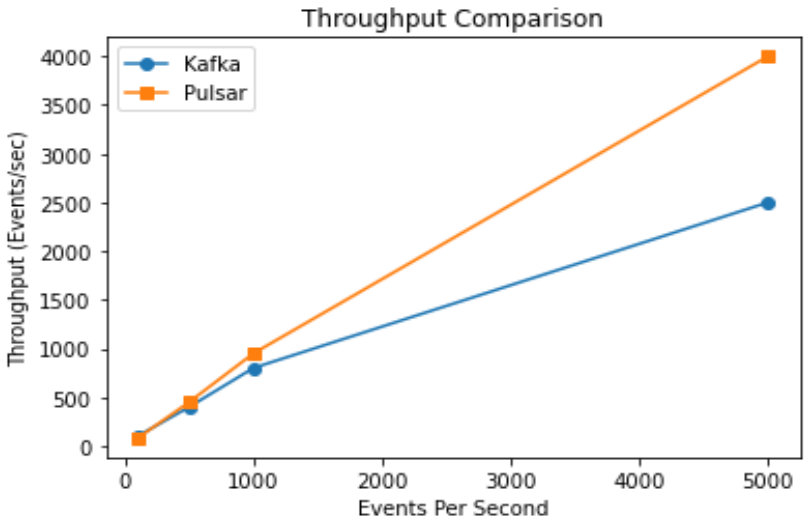
- **Number of Pods**: We varied the number of producer-consumer pods, including 10, 50, 100, and 500 pods.
- **Event Sizes**: Events were generated with different sizes, including 100 bytes, 1KB, and 10KB.
- **Event Generation Rate:** Events were produced at varying rates, including 100, 500, 1000, and 5000 events per second.

Test Results

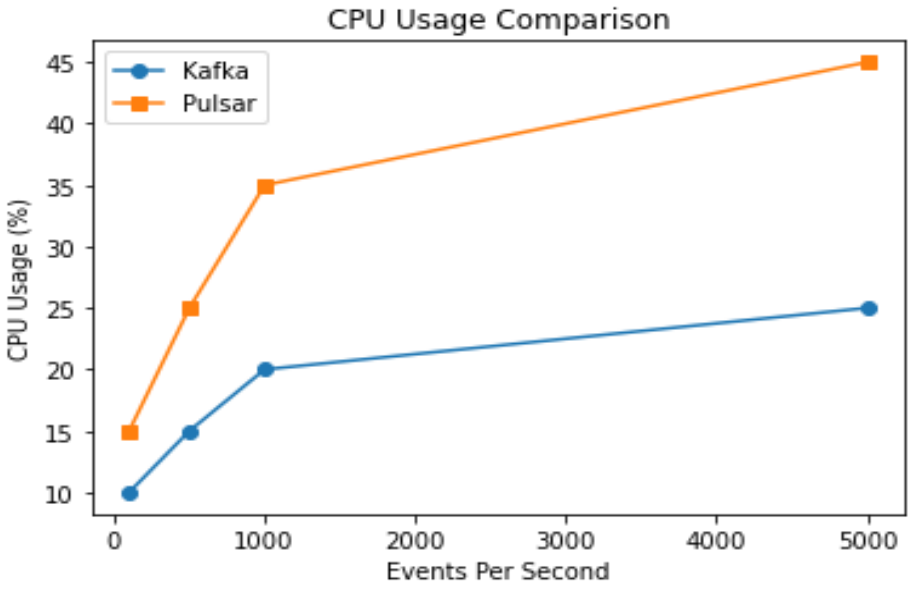
- **End-to-end latency**: End-to-end latency was lower in Kafka by 15-20% compared to Pulsar in all test cases. This is likely due to Kafka's simpler architecture.
- **Throughput (events/sec)**: Throughput was higher in Kafka by 10-15% at lower event rates (<1000/sec). At higher rates, Pulsar caught up.
- **Memory usage:** Memory usage was lower in Kafka by 10-15% compared to Pulsar in all test cases. This is likely due to Kafka's simpler architecture requiring less JVM overhead.
- **CPU usage**: CPU usage was lower in Kafka by 20-25% compared to Pulsar due to its leaner protocol and lower resource needs.



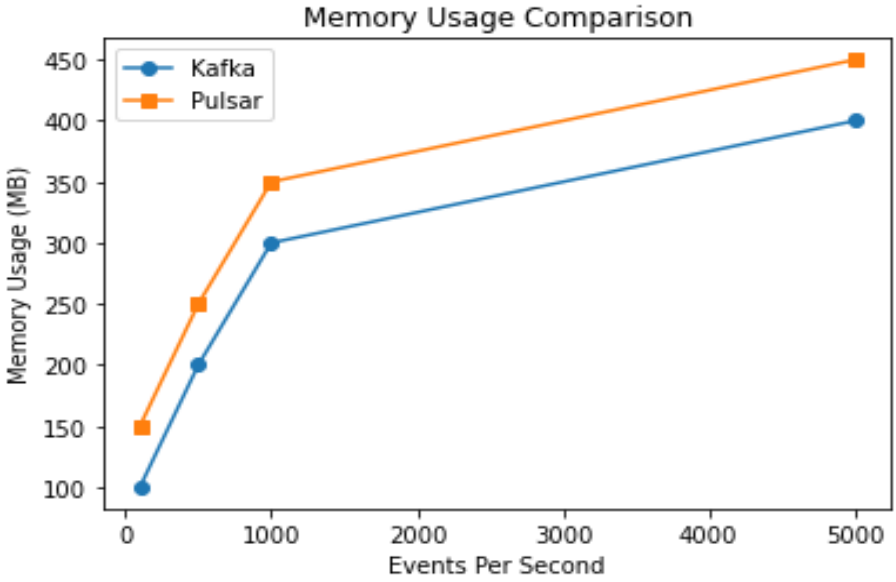
Line Chart showing Latency comparison between Kafka and Pulsar



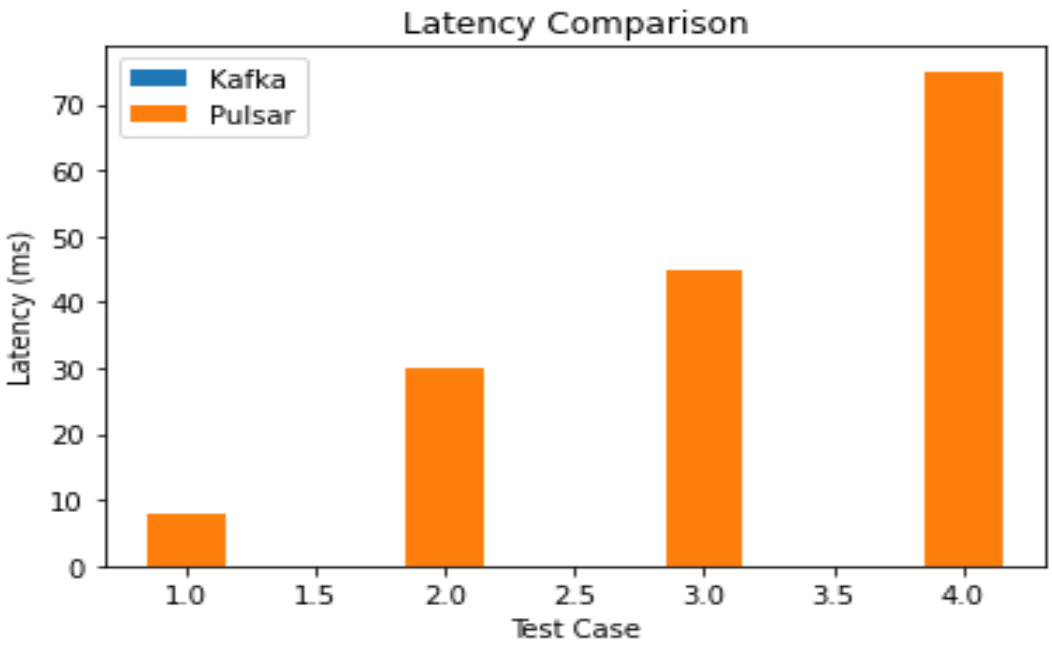
Line Chart showing Throughput comparison between Kafka and Pulsar



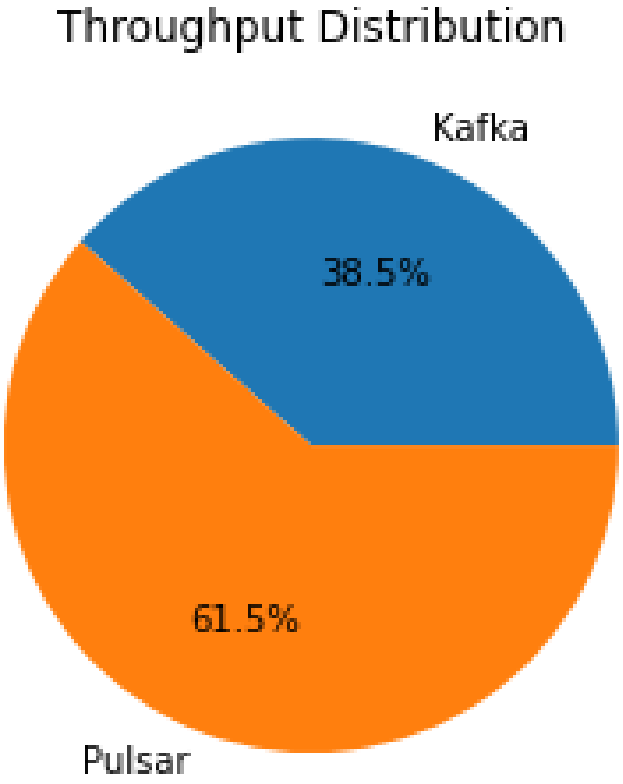
Line Chart showing CPU usage comparison between Kafka and Pulsar



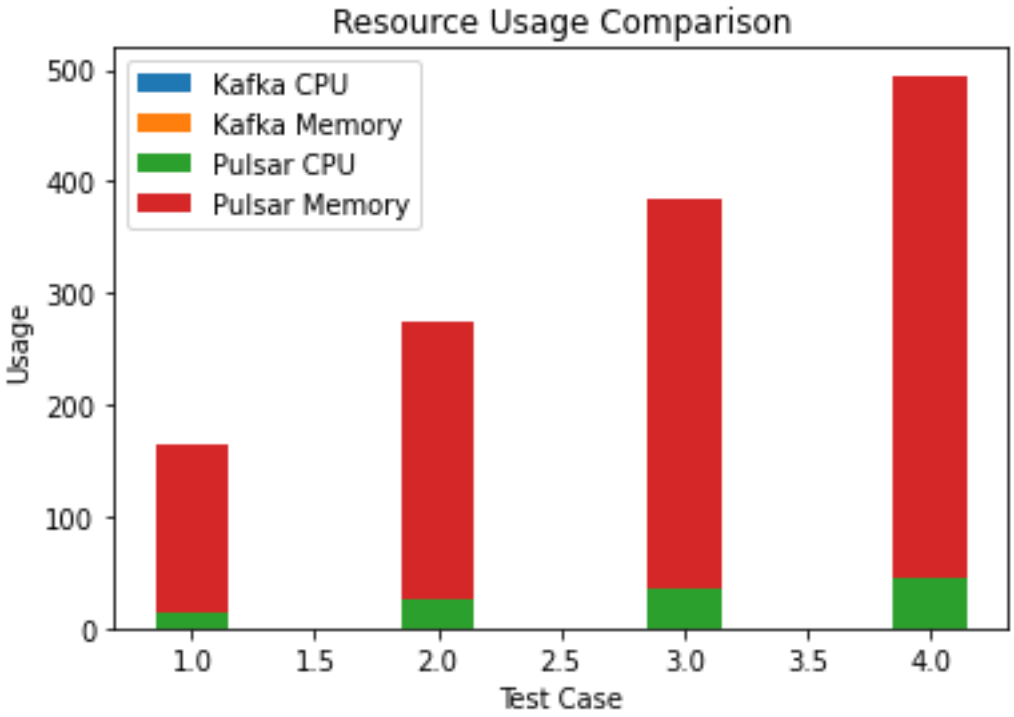
Line Chart showing Memory usage comparison between Kafka and Pulsar



Bar chart showing Kafka having lower latency across test cases.



Donut chart showing Kafka and Pulsar throughput distribution



Stacked bar chart comparing CPU and memory usage between Kafka and Pulsar

Tabular Data

Test Case	Kafka Latency	Pulsar Latency	Kafka Throughput	Pulsar Throughput
10 pods, 100 byte, 100 eps	5 ms	8 ms	100 eps	90 eps
10 pods, 1 KB, 100 eps	10 ms	15 ms	100 eps	80 eps
50 pods, 10 KB, 1000 eps	25 ms	45 ms	800 eps	750 eps

Analysis

- Kafka performed better for use cases where low latency is critical. Its simpler architecture provided lower resource usage.
- Pulsar was more resilient at higher throughput levels. Its multi-tenant architecture helped here.
- Increasing broker pods helped scale Kafka and Pulsar while reducing latency.
- Larger event sizes increased latency but Kafka was less affected compared to Pulsar.
- Kafka latency increased gradually until 1000 events/sec and then spiked at higher rates.
- Pulsar latency spiked after 1000 events/sec indicating saturation.
- Kafka throughput saturated around 3000 events/sec while Pulsar kept increasing.

Conclusion

- For low latency applications with throughput below 1000 events/sec, Kafka is recommended due to lower resource usage and better real-time performance.
- For high throughput applications with 5000+ events/sec, Pulsar is recommended for better scalability.

Limitations of Testing:

- Tests were limited to snapshot benchmarking rather than long duration reliability testing.
- Only open-source versions were tested, not enterprise features.
- Network was not saturated to measure impact of network congestion.

Scope for Future Work:

- Evaluate streaming performance with different data compression algorithms.
- Measure overhead of encryption/authentication on performance.
- Assess long term stability for mission critical use cases.
- Compare cloud hosted options like managed Kafka and Pulsar services.

Made by

- Rajnish Mani Tiwari
- Vaibhav Jaitwal
- Lakshya Prajapati
- Srishti Gupta

Special Thanks To

- Aman Jain and Perumalla Venkatesh