Stream Native

Pulsar vs. Kafka: A More Accurate Perspective from Use Cases and Community to Features and Performance

Table of Contents

Part 1: A Perspective on Pulsar Adoption, Use Cases, Differentiators, and Community

Introduction to Part 1

Data is transforming the business landscape with major industry leaders like Amazon, Uber, and Netflix demonstrating how access to real-time data, data messaging, and processing capabilities can translate to better products and customer experiences, disrupt entire industries, and generate billions in revenue. This need for real-time insights across industries is driving adoption and innovation in the messaging space.

As companies look to adopt real-time streaming solutions for new, innovative applications and to improve their existing systems, business leaders are seeking to better understand the respective advantages and disadvantages associated with the top technologies in the space, namely Pulsar, Kafka, and RabbitMQ.

Today, companies' messaging needs are increasingly complex and many organizations require a more comprehensive solution than RabbitMQ or Kafka can provide on their own. While RabbitMQ is best suited for message queueing and Kafka can manage data pipelines, Pulsar can accomplish both.

Companies that have a need for both types of messaging are increasingly choosing Pulsar for its flexibility, scalability, and ability to simplify operations by delivering multiple messaging functions on the same platform. Pulsar provides unique, sought-after capabilities, such as unified messaging and the ability to build streaming-first applications, which are powering some of today's most advanced companies.

However, because Pulsar is a younger technology, some are less familiar with its capabilities. In this post, we will address some common misconceptions about Pulsar and show Pulsar's growing popularity as evidenced by rapid growth in its adoption, an increase in the number and variety of use cases, and its ever-expanding community. We will also address the risks associated with adopting a new technology and explain why maintaining the status quo presents the risk of being left behind in a quickly changing landscape.

We have chosen to frame our discussion around commonly asked questions.

#1: How mature is Pulsar's technology and has it been tested in real-world applications?

To provide some insight into Pulsar's maturity and real-world use cases, we'll start with a brief background on its origin and development.

Pulsar's development began within Yahoo in 2012. It was committed to open source in 2016 and became a top-level Apache project in 2018. It has enterprise support from StreamNative. Pulsar enjoys several advantages as a newer entrant into the messaging space. Specifically, its developers at Yahoo had worked on Kafka and other traditional messaging technologies previously and knew the shortcomings associated with these platforms first-hand. As a result, they designed Pulsar with some distinct advantages that make it easier to operate as well as to provide features - such as unified messaging and tiered storage - which introduce new capabilities that are well-suited for emerging use cases.

By comparison, Kafka originated within LinkedIn. It was committed to open source in 2011 and became a top-level Apache project in 2012. As the first major event-streaming platform on the market, it is widely recognized and widely adopted. Kafka receives enterprise support from a number of companies, including Confluent. Compared to Pulsar, Kafka is a more mature technology that is popular, has a bigger community, and a more advanced ecosystem.

Even though it's a younger technology, Pulsar has seen tremendous growth, particularly over the past 18 months. It has been adopted by a growing list of global media companies, technology companies, and financial institutions. Below are examples of significant enterprise-level use cases that illustrate Pulsar's ability to handle mission-critical applications.

Tencent Builds Their Payment Platform on Pulsar

Tencent's adoption of Pulsar for their transactional billing system, Midas, demonstrates Pulsar's ability to handle mission-critical applications and provides compelling evidence that the technology has been rigorously tested and performs well in demanding environments. Midas operates at a massive scale, processing more than 10 billion transactions and 10+ TBs of data daily. The billing system is a critical piece of infrastructure for a company with over $50 billion in annual revenue.

Five Years of Success at Verizon Media

Verizon Media provides another compelling use case, having successfully operated Pulsar in production for over five years. Verizon Media, via its acquisition of Yahoo, is the original developer of Pulsar. In their recent *Pulsar Summit talk*, Joe Francis and Ludwig Pummer of Verizon Media described Pulsar as a "battle-tested" system that is being used throughout the Verizon Media landscape. They shared that Pulsar routinely handles up to 3 million write requests/second on more than 2.8 million distinct topics. Pulsar has satisfied Verizon Media's need for a low-latency, highly available system that can be scaled easily and has the ability to support a business that operates across six global data centers.

Splunk Adopts Pulsar for Their Data Stream Processor

Another key adoption story comes from Splunk, a company that has used Kafka in production environments for years. During a recent Pulsar Summit talk, "*Why Splunk Chose Pulsar*," Karthik Ramasamy shared Splunk's reasons for choosing Pulsar to power its next-generation analytics product, Splunk DSP, which handles billions of events per day. Ramasamy explained that Pulsar was able to meet 18 key requirements and cited its ease of scalability, lower operating costs, better performance, and strong open-source community as major factors in their decision to adopt Pulsar.

The above use cases clearly demonstrate that Pulsar is a powerful solution that many industry leaders are choosing to power critical business infrastructure. Although Kafka is more mature and more widely used, Pulsar's rapid rate of adoption is evidence of its strong capabilities and readiness for mission-critical use cases.

#2: What are the key differences between the competing technologies, and what business advantages are associated with each?

While major technology and media companies, such as Uber and Netflix, have been able to successfully build unified batch and stream processing and streaming-first applications to power their real-time data needs, most companies lack the vast engineering and financial resources these applications typically require. However, Pulsar offers advanced messaging capabilities that enable companies to overcome many of these challenges.

Below, we highlight three unique capabilities - some current and others still in development - that distinctly set Pulsar apart from its competitors.

Unified Messaging Model

Two of the most common types of messaging used today are application messaging (traditional queuing systems) and data pipelines. Application messaging is used to enable asynchronous communications (often developed on platforms such as RabbitMQ, AMQP, JMS, among others), while data pipelines are used to move high volumes of data between different systems (such as Apache Kafka or AWS Kinesis). Because these two types of messaging are performed on different systems and serve different functions, companies often need to operate both. Developing and managing separate systems is not only expensive and complex, but can also make it difficult to integrate systems and centralize data.

Pulsar's core technology gives users the ability both to deploy it as a traditional queuing system and use it in data pipelines, uniquely positioning Pulsar as the ideal platform to provide unified messaging capabilities. Unified messaging makes it easier for organizations to capture and distribute their data, which facilitates the use of real-time data to drive business innovation.

Pulsar also recently added tools - Kafka-on-Pulsar (KoP) and AMQP-on-Pulsar (AoP) - that make it even easier for companies to leverage these unified messaging capabilities. (We discuss KoP and AoP in more detail below.)

Batch and Event-Stream Storage

Because companies today need to be able to make timely decisions and react to change quickly, the need for real-time, meaningful data has never been more critical. At the same time, it is crucial to be able to integrate and understand large amounts of historical data in order to gain a complete picture of a business.

Traditional Big Data systems (such as Hadoop) facilitate decision-making by allowing organizations to analyze massive historical data sets. However, as these systems can take minutes, hours, or even days to process data, they struggle to integrate real-time data and the results they produce are often of limited value.

Stream processors, such as Kafka Streams, are adept at processing streaming data and computing answers closer to real-time, but are not a good fit for processing large historical datasets. Many organizations need to run both batch and streaming data processors in order to gain the insights they need for their business. However, maintaining multiple systems is expensive and each system has its own respective challenges.

More recently, systems have been developed which can do both batch and stream processing. Apache Flink is one example. Currently, Flink is used for stream processing with both Kafka and Pulsar. However, Flink's batch capabilities are not particularly compatible with Kafka as Kafka is only able to deliver data in streams, making it too slow for most batch workloads.

By contrast, Pulsar's tiered storage model provides the batch storage capabilities needed to support batch processing in Flink. In the near future, Flink's batch processing capabilities will be integrated with Pulsar, enabling companies to query both historical and real-time data quickly and more easily, unlocking a unique competitive advantage.

"Streaming-First" Applications

Web application development is in the midst of a major transformation as companies look to develop more sophisticated web applications. The traditional application model that pairs a single monolithic application with a large SQL database is giving way to applications composed of many, smaller components, or "microservices."

Many organizations are now adopting microservices because they offer greater flexibility to meet changing business needs and help facilitate development across growing engineering teams. However, microservices introduce new challenges, such as the need to enable communication among various components and keep them synchronized.

With a newer microservices technique called “event sourcing,” applications produce and broadcast streams of events into a shared messaging system which captures the event history in a centralized log. This improves the flow of data and helps keep applications in sync.

But event sourcing can be difficult to implement as it requires both traditional messaging capabilities and the ability to store event history for long periods of time. While Kafka is capable of storing streams of events for days or weeks, event sourcing typically requires longer retention times. This added challenge often requires users to build multiple tiers of Kafka clusters to manage the growth of event data, plus additional systems to manage and track data collectively.

By contrast, Pulsar’s unified messaging model is a natural fit, as it can easily distribute events to other components and effectively store event streams for indefinite periods of time. This unique design feature makes Pulsar especially attractive to companies looking to acquire dynamic, streaming-first capabilities.

While unified messaging, combined batch and event-streaming storage, and a “streaming-first” approach might be feasible to achieve with other systems, these features would be complex to implement and would require a great deal of effort and investment. In contrast, Pulsar’s design includes all of these features, enabling users to adapt to the changing technology landscape easily and with far less complexity.

#3: Does Pulsar have the community and enterprise support it needs to continue to develop and garner further adoption?

A snapshot comparison of the Pulsar and Kakfa communities today reflects that Kafka’s is larger overall, with more Slack users and more stack overflow questions. While Pulsar's community is currently smaller, it is highly engaged and rapidly growing. Below are some highlights of its recent momentum.

Pulsar's First Global Summit

In June, *Pulsar held its first global event* - the *Pulsar Summit Virtual Conference 2020*. The event featured more than 30 speaker sessions from Pulsar's top contributors, thought leaders, and developers. We heard real-world Pulsar adoption stories and received insights from companies such as Verizon Media, Splunk, Iterable, and OVHcloud.

With more than 600 sign-ups - including attendees from top internet, technology, and financial institutions such as Google, Microsoft, AMEX, Salesforce, Disney, and Paypal - the event revealed a highly engaged and global Pulsar community and demonstrated that interest in Pulsar is burgeoning.

In fact, the global Pulsar community subsequently asked us to host dedicated regional events in Asia and Europe soon. To meet this growing demand, we have scheduled Pulsar Summit Asia 2020 in October and are currently planning Pulsar Summit Europe.

Community Support - Training and Events

In addition to facilitating large, widely attended summits, the Pulsar community is focusing on interactive training and online events. For example, earlier this year, the community, led by StreamNative, launched a weekly *live-streaming, interactive tutorial* called TGIP (Thank Goodness It's Pulsar) that provides technology updates and hands-on tutorials highlighting various operational aspects. TGIP sessions are available on YouTube and StreamNative.io and are helping to augment Pulsar's growing knowledge base.

In 2020, the Pulsar community also launched *monthly webinars* to share best practices, new use cases, and technology updates. Recent webinars have been hosted by strategic commercial and open-source partners such as OVHCloud, Overstock, and Nutanix. On July 28th, StreamNative will be hosting *Operating Pulsar in Production* as a panel discussion with additional participants from Verizon Media and Splunk.

Pulsar's ecosystem has further evolved with the expansion of professional training, which is available through StreamNative and other partners. In fact, Pulsar and Kafka expert Jesse Anderson recently led an in-depth training session on *Developing Pulsar Functions*. Professional training sessions help to enlarge the pool of Pulsar-trained engineers and allow Pulsar users to accelerate their messaging and streaming platform development initiatives.

In addition, an increase in the *publication of white papers* is helping to expand Pulsar's knowledge base.

Committed community partners have also contributed to key project advancements. Below, we look at two recent product launches.

OVHCloud Helps Companies Move from Kafka to Pulsar

In March 2020, OVHCloud and StreamNative launched Kafka-on-Pulsar (KoP), the result of the two companies working closely in partnership. KoP enables Kafka users to migrate their existing Kafka applications and services to Pulsar without modifying the code. Although only recently released, KoP has already been adopted by several organizations and is being used in production environments. Moreover, KoP's availability is helping to expand Pulsar's adoption.

China Mobile Helps Companies Move from RabbitMQ to Pulsar

In June 2020, China Mobile and StreamNative announced the launch of another major platform upgrade, AMQP on Pulsar (AoP). Similar to KoP, AoP allows organizations currently using RabbitMQ (or other AMQP message brokers) to migrate existing applications and services to Pulsar without code modification. Again, this is a key initiative that will help drive the adoption and usage of Pulsar.

The events and initiatives described above illustrate the Pulsar community's firm commitment to education and ecosystem development. More importantly, they demonstrate the momentum and growth we can expect in the future.

Conclusion

In today's ever-changing business landscape, access to data can unlock innovative business opportunities, define new categories, and propel companies ahead of the competition. As a result, organizations are increasingly seeking to leverage their data and the insights that can be gained from it to develop competitive advantages, and they are seeking new technologies to help them achieve these goals.

In this post, we set out to address some common business concerns organizations face when evaluating a new technology. These include the technology's proven capabilities, its ability to enable in-demand business use cases, and, in the case of open-source technologies, the size and level of engagement within the project's community.

The Tencent, Verizon Media, and Splunk use cases described earlier demonstrate Pulsar's ability to deliver mission-critical applications in the real world. Beyond its proven capabilities, Pulsar's ability to deliver unified messaging and streaming-first applications provides a marked advantage by enabling organizations to build disruptive, competitive technologies without requiring extensive resources. Pulsar's integration with Flink, which is currently in development, will provide yet another competitive advantage: the ability to perform both batch and stream processing on the same platform.

While the Pulsar community and a few other key areas, such as documentation, are still small, their growth has increased considerably in the past 18 months. Pulsar's highly engaged and quickly growing community and ecosystem are committed to contributing to the ongoing expansion of Pulsar's knowledge base and training materials, while also accelerating the development of key capabilities.

Disruption can happen quickly and organizations evaluating any technology need to consider not only the strengths and weaknesses it has today, but also how the technology will continue to grow and evolve to meet business needs in the future. The combination of Pulsar's enhanced messaging offering and unique capabilities make it a strong alternative that should be considered by any company looking to develop real-time data streaming capabilities.

Part 2: A Perspective on Performance, Architecture, and Features

Introduction to Part 2

The shift to real-time streaming technologies has bolstered the adoption of Pulsar and there has been a marked increase in both the interest and adoption of Pulsar in 2020. With Pulsar being sought out by companies developing messaging and event-streaming applications — from Fortune 100 companies to forward-thinking start-ups — and so much growth around the Pulsar project, it has garnered a lot of recent press and attention.

Pulsar offers many advantages that make it an attractive choice for companies seeking to adopt a unified messaging and event streaming platform. Compared with Kafka, Pulsar is more resilient and less complex to operate and scale and it covers all the fundamentals necessary for building event streaming applications and incorporates many built-in features with a rich set of tools.

In Part 2, we compare Pulsar and Kafka in terms of performance, architecture, and features to help you make an informed decision.

The shift to real-time streaming technologies has bolstered the adoption of Pulsar and there has been a marked increase in both the interest and adoption of Pulsar in 2020. With Pulsar being sought out by companies developing messaging and event-streaming applications — from Fortune 100 companies to forward-thinking start-ups — and so much growth around the Pulsar project, it has garnered a lot of recent press and attention.

For the most part, the recent press and articles have helped to provide valuable education and transparency into Pulsar's use cases and capabilities. Companies such as Verizon Media, Iterable, Nutanix, and Overstock.com, are just a handful of companies who have recently presented their Pulsar use cases and shared insights into how they are leveraging Pulsar to achieve their business goals.

However, not all recent press has been entirely accurate and we have received a number of requests from the Pulsar community to address a recent Confluent blog comparing Kafka, Pulsar, and RabbitMQ. We appreciate that Pulsar is a quickly growing and evolving technology and we would like to take this opportunity to provide a deep dive into Pulsar's capabilities.

In Part 2, we will leverage in-depth knowledge of the Pulsar technology, community, and ecosystem to provide a more balanced and holistic picture of the event-streaming landscape.

General

Components

Pulsar is composed of 3 main components: a broker, which is a stateless service that clients connect to for core messaging, and two stateful services, Apache BookKeeper and Apache ZooKeeper. BookKeeper nodes (bookies) store the actual messages and cursor positions while ZooKeeper is used strictly for metadata storage by both brokers and bookies. Additionally, BookKeeper leverages RocksDB as an embedded database, which is used to store internal indices, but it is not managed independently of BookKeeper.

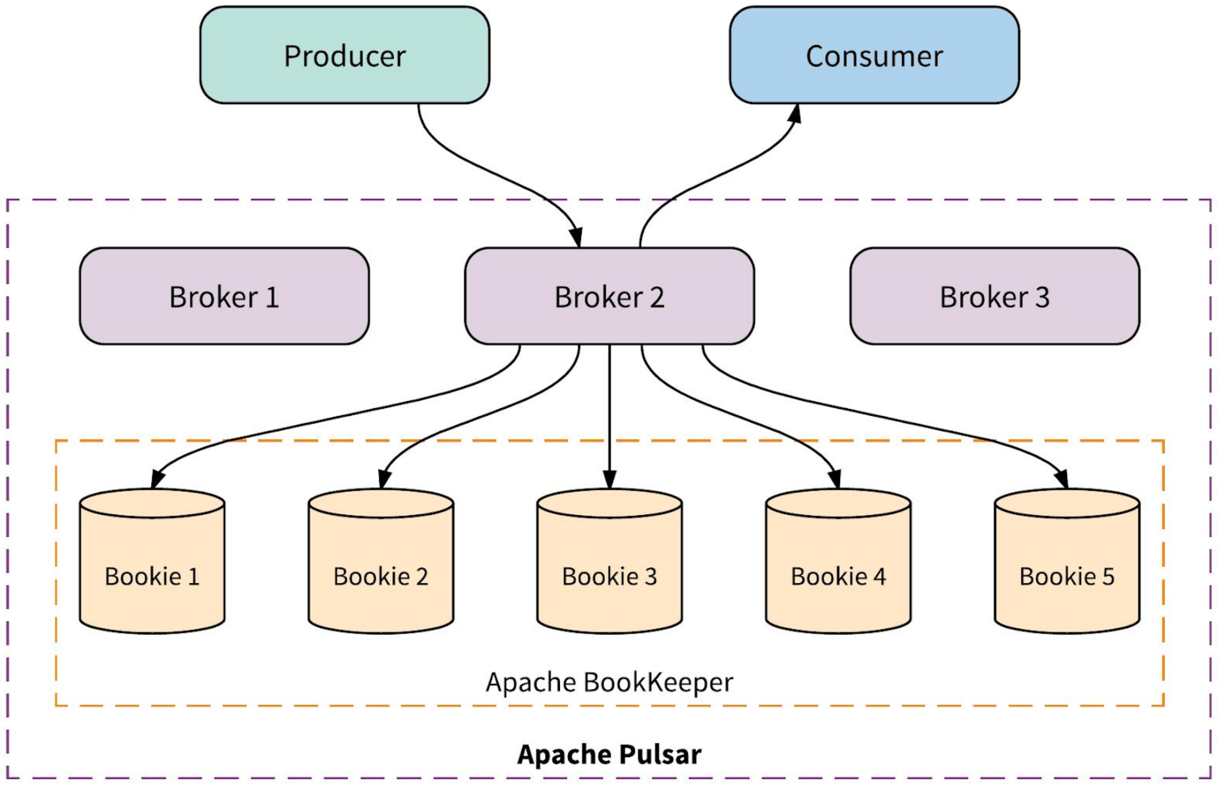

Figure 1: Pulsar layered architecture

Unlike Kafka, which employs a monolithic architecture model that tightly couples serving and storage, Pulsar leverages a multi-layer design which allows it to manage these functions in separate layers. Pulsar's broker performs computing on one layer and the bookie manages stateful storage on another.

While, on the surface, it may seem like Pulsar's architecture is more complicated compared with Kafka's, the reality is more nuanced. Architectural decisions come with trade-offs and Pulsar's inclusion of BookKeeper enables it to provide more flexible scalability, lower operational burden, faster, and more consistent performance. We will talk in more detail about each of these benefits later on.

Storage Architecture

The architectural differences in Pulsar also extend to how Pulsar stores data. Pulsar breaks topic partitions into segments and then distributes the segments across the storage nodes in Apache BookKeeper to get better performance, scalability, and availability.

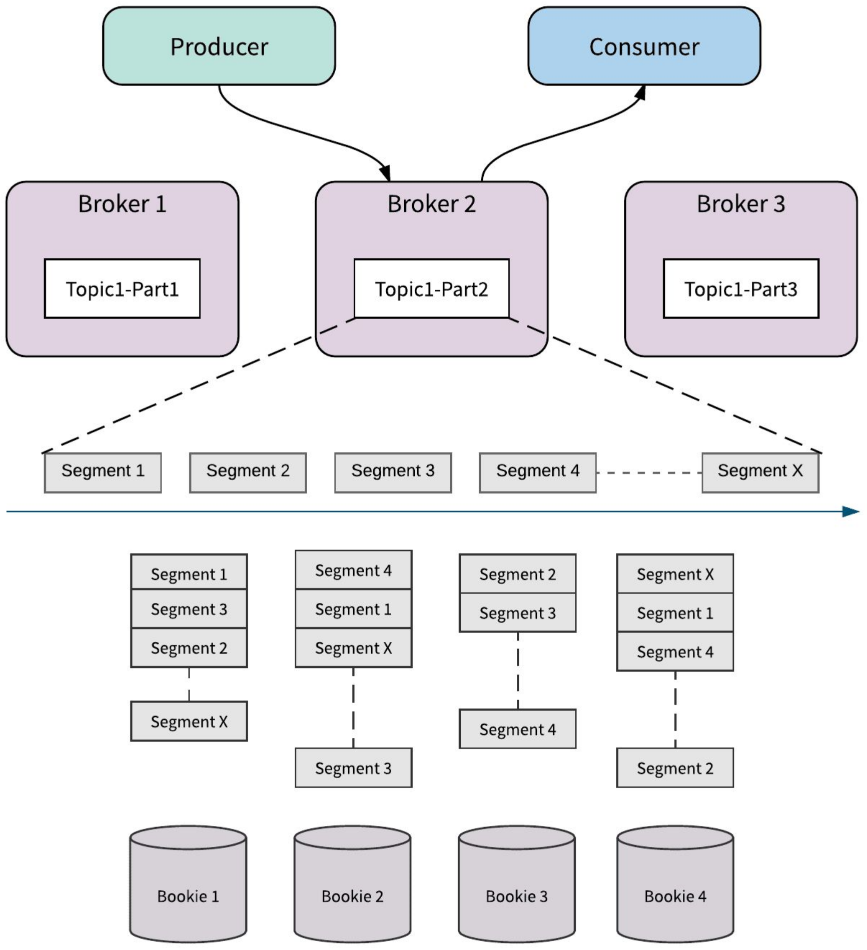

Figure 2: Pulsar storage architecture

Pulsar's infinite distributed log is segment centric and implemented by leveraging scale-out log storage (via Apache BookKeeper) with built-in tiered storage support which enables segments to be distributed evenly across storage nodes. Because the data associated with any given topic is not tied to any specific storage node, it is easy to replace nodes and to scale up or down. Moreover, the smallest or slowest node in the cluster cannot impose any storage or bandwidth limitations.

Pulsar's partition-rebalance-free architecture ensures instant scalability and higher availability. Both of these factors are extremely important and make Pulsar well-suited for building mission-critical services such as billing platforms for financial use cases, transaction processing systems for e-commerce and retailers, and real-time risk control systems for financial institutions.

By leveraging the powerful Netty framework, data is zero-copied when it is transferred from producers to brokers to bookies. This works extremely well for all streaming use cases because the data is transferred directly over the network or to disk without any performance penalties.

Message Consumption

Pulsar's consumption model takes a streaming-pull approach. This is an enhanced version of long-polling as it eliminates the wait time between individual calls and requests and provides bi-directional message streaming. The streaming-pull model enables Pulsar to achieve lower end-to-end latency than any other existing long-polling-based messaging solutions, such as Kafka.

Ease of Use

Operational Simplicity

When evaluating the operational simplicity for a given technology, it's important to consider not only the initial set-up but also its long-term maintenance and scalability. Helpful questions to consider include:

- How quickly and simply can you scale your cluster to keep up with your business growth?
- Does your cluster provide out-of-the-box features for multi-tenancy that map well to multiple teams and users?
- Will the operational tasks, such as replacing hardware, require maintenance that potentially can impact the availability and reliability of your business?
- Can your system easily replicate data for geographic redundancy or different access patterns

Long-time Kafka users will know these are not easy questions to answer when operating Kafka. Most of these tasks require a suite of tools external to Kafka, such as cruise control for managing rebalancing of clusters and Kafka mirror-maker/replicator for any replication needs.

Many organizations also develop tooling for provisioning and managing multiple distinct clusters as Kafka can be difficult to share across teams. These types of tools are critical to run Kafka at scale successfully but also add to its complexity. The most capable tools for managing Kafka clusters have been developed as proprietary, closed source tooling. It is no surprise that Kafka's complex overhead and operations have pushed many businesses to use Confluent.

By contrast, Pulsar's goal is to streamline operations and scalability. Below we respond to the same questions with respect to Pulsar's capabilities:

- How quickly and simply can you scale your cluster to keep up with your business growth?

  New compute and storage capacity is automatically and immediately utilized with Pulsar's automatic load balancing. This allows migrating topics to equalize load among brokers and new bookie nodes immediately receiving write traffic for new segments, with no manual rebalancing or broker management required.

- Does your cluster provide out-of-the-box features for multi-tenancy that map well to multiple teams and users?

  Pulsar provides a hierarchical structure of tenants and namespaces which map logically to organizations and teams, with these same constructs allowing for simple ACLs, quotas, self-service controls, and even resources isolation to allow cluster operators to confidently manage shared clusters.

- Will the operational tasks, such as replacing hardware, require maintenance that potentially can impact the availability and reliability of your business?

  The stateless broker of Pulsar is able to be replaced easily, as there is no risk of data loss. Bookie nodes will automatically replicate any under-replicated segments of data and tools for decommissioning and replacing nodes is built-in and easily automatable.

- Can your system easily replicate data for geographic redundancy or different access patterns?

  Pulsar has built-in replication, which can be used to seamlessly span geographic regions or replicate data to additional clusters for other purposes (disaster recovery, analytics, and so on.)

In comparison to Kafka, Pulsar's batteries included approach provides a more complete solution to the real-world problems of streaming data. With this added perspective, the overall simplicity of use favors Pulsar as it offers a more complete core feature set and allows operators and developers to focus on the core needs of their business.

Documentation and Learning

As Pulsar is a newer technology than Kafka, the ecosystem is not as developed and Pulsar is still building out its documentation and training resources. However, this has been a major area of growth for Pulsar and great strides have been made over the past 18 months. Here are some of the most notable accomplishments:

- Pulsar Summit Virtual Conference 2020, the first global summit for Pulsar, featured 36 sessions with speakers from 25+ organizations and attracted 600+ attendees sign-ups.
- 50+ videos and training decks created in 2020 alone.
- Weekly Pulsar live-streaming and interactive tutorials.
- Professional training with industry-leading trainers.
- Monthly webinars with strategic commercial partners.
- Recent white papers from Tuya, OVHCloud, Tencent, Yahoo!Japan, and more

For more on Pulsar documentation and training, visit StreamNative's Resources page.

Enterprise Support

Kafka and Pulsar both have enterprise-grade support offerings. Kafka has enterprise-grade support offerings from multiple large vendors, including Confluent. Pulsar has enterprise-grade support from StreamNative, a newer entrant on the scene. StreamNative offers fully managed Pulsar services for enterprises as well as enterprise-grade support for Pulsar.

StreamNative has a fast-growing and highly-experienced team with deep roots in the messaging and event-streaming space. StreamNative was founded by the core team of Pulsar and BookKeeper. In just a few short years, StreamNative has helped to significantly grow the Pulsar ecosystem — more on this in our next post — including garnering the support of committed strategic partners who are helping to further Pulsar development to meet the needs of a wide number of use cases.

Some major recent developments include the launch of Kafka-on-Pulsar, or KoP, which was launched in March 2020 by OVHCloud and StreamNative. By adding the KoP protocol handler to an existing Pulsar cluster, you can now migrate existing Kafka applications and services to Pulsar without modifying the code. In June 2020, China Mobile and StreamNative announced the launch of another major platform upgrade, AMQP on Pulsar (AoP). This enables RabbitMQ applications to leverage Pulsar's powerful features, such as infinite event stream retention with Apache BookKeeper and tiered storage. We will talk about each of these in more detail in our next post.

Integrations

Alongside the rapid growth in the number of Pulsar adoptions, we have seen the Pulsar community develop into a large, highly-engaged, and global user community. This active Pulsar community has played a key role in driving growth in the number of integrations in the ecosystem. In just the past six months, the number of officially supported connectors in the Pulsar ecosystem has grown tremendously.

To further support this community effort, StreamNative recently launched StreamNative Hub, which provides a convenient central location where users can find and download integrations. This resource will help accelerate the growth of Pulsar's connector and plug-in ecosystem.

The Pulsar community has also been actively working with other communities on integrating with their projects. For example, Pulsar has been working closely with the Flink community on developing the Pulsar-Flink Connector as part of FLIP-72. Pulsar-Spark Connector provides developers the capability of using Apache Spark to process events in Apache Pulsar.

SkyWalking Pulsar Plugin integrates Apache SkyWalking with Apache Pulsar, allowing people to trace Pulsar messages using SkyWalking. These are just a few examples of a large collection of integrations the Pulsar community is currently working on.

Client Library Diversity

Pulsar currently supports 7 languages officially, compared with Kafka's 1 language. While the Confluent post reported that Kafka currently supports 22 languages, it is important to note that most of the 22 languages Confluent referred to are not official clients, and many are no longer actively maintained. At last count, the Apache Kafka project had only one officially released client, compared with the seven officially supported by Apache Pulsar:

- Java
- C
- C++
- Python
- Go
- .NET
- Node

Pulsar also supports a rapidly growing list of community developed clients, which includes the following:

- Rust
- Scala
- Ruby
- Erlang

Performance and Availability

Throughput, Latency, and Scale

Both Pulsar and Kafka have successfully been leveraged in a number of enterprise use cases and each system has its advantages, with both systems being capable of handling large amounts of traffic with similar amounts of hardware. One common misconception of Pulsar is that because it has more components, it must require more servers to achieve the same performance. While this may be true in some hardware configurations, in many configurations Pulsar can get more from the same resources.

As an example, Splunk recently shared that one of the reasons they choose Pulsar over Kafka is that Pulsar is 1.5x - 2x lower in CAPEX cost with 5x - 50x improvement in latency and 2x - 3x lower in OPEX due to layered architecture (from slide 34). They found this was due to Pulsar being better able to utilize disk IO with lower CPU utilization and better control over memory.

More generally, companies such as Tencent have chosen Pulsar in large part due to its performance attributes. As discussed in a recent whitepaper Tencent's billing platform, which serves over a million merchants and manages 30 billion escrow accounts, is currently using Pulsar to process hundreds of millions of dollars in revenue per day. Tencent chose Pulsar over Kafka for its predictable low latency, stronger consistency, and durability guarantees.

Ordering Guarantees

Apache Pulsar offers four distinct subscription modes. The four modes and their associated ordering guarantees are described below. An individual application's ordering and consumption scalability requirements determine which subscription mode is appropriate for that application.

- Both the Exclusive and Failover subscription modes provide very strong ordering guarantees at a partition level even when consuming a topic in parallel across many consumers.
- Shared mode allows you to scale the number of consumers beyond the number of partitions, thus making this mode well-suited for worker queue use cases.
- Key_Shared mode combines the advantages of the other subscription modes. It allows scaling the number of consumers beyond the number of partitions and provides a strong ordering guarantee at a key level.

For more information about Pulsar's subscription types and their associated ordering guarantees, see subscriptions.

Feature

Built-In Stream Processing

Pulsar and Kafka have two different goals when it comes to built-in stream processing. Pulsar integrates with Flink and Spark, two mature, full-fledged stream processing frameworks, for more complex stream processing needs and developed Pulsar Functions to focus on lightweight computation. Kafka developed Kafka Streams with the goal of providing a full-fledged stream processing engine.

As a result, Kafka Streams is more complex. Users need to figure out where and how to run the KStreams application and it is unnecessarily complicated for most lightweight computing use cases.

Pulsar Functions, on the other hand, makes lightweight computing use cases easy to implement and enables developers to create complex processing logic without deploying a separate neighboring system. Additionally, it provides language-native and easy-to-use API. Developers don't have to learn a complicated API in order to start writing event streaming applications.

A Pulsar Improvement Proposal (PIP) was recently submitted to the Pulsar project to introduce Function Mesh. Function Mesh is a serverless event-streaming framework that combines multiple Pulsar Functions together to facilitate building complex event-streaming applications.

Exactly-Once Processing

Pulsar currently supports exactly-once producers via broker-side deduplication and we are happy to share a major upgrade is presently in development and will be available soon!

Support for transactional message streaming started in PIP-31 and is currently in development. This feature will improve Pulsar's message delivery semantics and processing guarantees. With transactional streaming, each message is written or processed exactly once with no duplication or data loss, even when a broker or function instance fails. Transactional messaging not only makes it easier to write applications using Pulsar or Pulsar Functions, but it also expands the scope of the use cases that Pulsar can support. We are making rapid progress on this feature and it will be included in Pulsar 2.7.0 which is scheduled for release in September 2020.

Topic (Log) Compaction

Pulsar was designed to provide users a choice of formats for consuming data. Applications can choose to consume either raw data or compacted data, as appropriate. By doing this, Pulsar allows for non-compacted data to have a retention policy, keeping control over unbounded growth, but still allowing periodic compaction to generate the most recent materialized view around. The built-in tiered storage feature also allows Pulsar to offload the non-compacted data from BookKeeper to cloud storage and makes it much cheaper to store events for a much longer period.

Unlike Pulsar, Kafka does not offer users the option to consume raw data. Kafka removes raw data immediately after it is compacted.

Use Case

Event Streaming

Pulsar was originally developed as a unified pub/sub messaging platform in Yahoo! (known as Cloud Messaging). However, Pulsar has grown beyond a messaging platform and become a unified messaging and event streaming platform. Pulsar includes a complete set of tools as part of the platform, to provide all the fundamentals necessary for building event streaming applications. Pulsar encompasses the following event streaming capabilities:

- Infinite event stream storage makes it possible to store events at scale by leveraging scale-out log storage (via Apache BookKeeper) with built-in tiered storage support to cost-effective systems like S3, HDFS, and so on.
- Unified pub/sub messaging model allows developers to add messaging to their applications easily. This model can be scaled both based on traffic and on the user's needs.
- Protocol handler framework and protocol compatibility with Kafka (via Kafka-on-Pulsar and AMQP (via AMQP-on-Pulsar) allow applications to produce and consume events from anywhere using any existing protocols.
- Pulsar IO provides a set of connectors integrating larger ecosystems, allowing users to ingest data from external systems without writing any code.
- Integration with Flink enables comprehensive event processing.
- Pulsar Functions offers a lightweight serverless framework for processing events as they arrive.

- Integration with Presto (Pulsar SQL) allows data scientists and developers to use ANSI-compliant SQL to gain insights into their data and business.

Message Routing

Pulsar provides comprehensive routing capabilities through Pulsar IO, Pulsar Functions, and Pulsar Protocol Handler. Pulsar's routing capabilities include content-based routing, message transformation, and message enrichment.

Pulsar has more robust routing capabilities compared with Kafka. Pulsar provides a flexible deployment model for connectors and functions. These can be run within a broker, allowing for easy deployment. Alternatively, they can be run in a dedicated pool of nodes (similar to Kafka Streams) which allows for massive scale-out. Pulsar also integrates natively with Kubernetes. In addition, Pulsar can be configured to schedule function and connector workloads as pods, thus fully leveraging the elasticity of Kubernetes.

Message Queuing

As noted above, Pulsar was originally developed as a unified pub/sub messaging platform. The Pulsar team learned a lot of the pros and cons of operating existing open-source messaging systems and applied their experiences to designing Pulsar's unified messaging model. The Pulsar messaging API combines both queueing and streaming capabilities. It not only allows implementing a worker queue that delivers messages round-robin to competing consumers (via Shared subscription) but also supports event streaming by delivering messages based on the order of messages in a partition (via Failover subscription) or a key range (via Key_Shared subscription). Developers are able to build both messaging and event streaming applications on the same set of data without duplicating it to different siloed systems.

Additionally, The Pulsar community is also working on bringing the native support of different messaging protocols (such as AoP and KoP) to Apache Pulsar to extend Pulsar's messaging capabilities.

Conclusion

This is a very exhilarating time marked by tremendous growth and change in the Pulsar community. Pulsar's ecosystem is developing and expanding as its technology continues to evolve and new use cases are added.

Pulsar offers many advantages that make it an attractive choice for companies seeking to adopt a unified messaging and event streaming platform. compared with Kafka, Pulsar is more resilient and less complex to operate and scale.

Like any new technology, it can take time to roll-out and adopt, however, Pulsar provides a turnkey solution that is ready for production upon installation with lower ongoing maintenance costs. Pulsar covers all the fundamentals necessary for building event streaming applications and incorporates many built-in features, including a rich set of tools. Pulsar's tools are available for immediate use and do not require additional installation steps.

At StreamNative, we are continuously working on developing new features and enhancements to strengthen Pulsar's capabilities and grow the community. We are making excellent progress on the ambitious goals we have set for 2020 and look forward to releasing Pulsar 2.7.0 in September.

Learn More About Pulsar

We encourage you to sign up for the *Pulsar Newsletter* to stay up-to-date on upcoming events and technology updates. If you would like to chat with current Pulsar users, you can join the *Pulsar Slack Channel*.

Special Thanks

We would like to thank the many members of the Pulsar community who contributed to this article - especially, Jerry Peng, Jesse Anderson, Joe Francis, Matteo Merli, Sanjeev Kulkarni, and Addison Higham.

Links & Resources

- For more on Pulsar documentation and training, visit *StreamNative's Resources page*.
- You can also access *recent white papers* from Tuya, OVHCloud, Tencent, Yahoo!Japan, and more.

Do you have any questions? Please join the Pulsar Slack Channel.

About the Author

Carolyn King is an experienced product marketing leader with a focus on SaaS-based and emerging technologies, including Real-Time Streaming, AI and Ad Tech. She has led marketing at a number of VC-backed companies and has successfully driven the development and execution of go-to-market strategies. Carolyn holds an MBA from UCLA Anderson and a BA in Business-Economics from UCLA. You can follow her on twitter.

Addison Higham is a senior software developer at StreamNative. He has been focusing on micro-services, distributed systems, big data, and cloud architecture with extensive experience in various big data Apache projects, such as Pulsar, Flink, and Spark. You can follow him on twitter.

Sijie Guo is the co-founder and CEO of StreamNative, which provides a cloud-native event streaming platform powered by Apache Pulsar. Sijie has worked on messaging and streaming data technologies for more than a decade. Prior to StreamNative, Sijie cofounded Streamlio, a company focused on real-time solutions. At Twitter, Sijie was the tech lead for the messaging infrastructure group, where he co-created DistributedLog and Twitter EventBus. Prior to that, he worked on the push notification infrastructure at Yahoo!, where he was one of the original developers of BookKeeper and Pulsar. He is also the VP of Apache BookKeeper and PMC member of Apache Pulsar. You can follow him on twitter.

About StreamNative

Founded by the original developers of Apache Pulsar and Apache BookKeeper, StreamNative provides a cloud-native event streaming platform powered by Apache Pulsar that enables companies to access enterprise data as real-time event streams.

For more information about StreamNative, refer to the following channels.

- StreamNative Website: https://streamnative.io
- StreamNative Twitter: https://twitter.com/streamnativeio