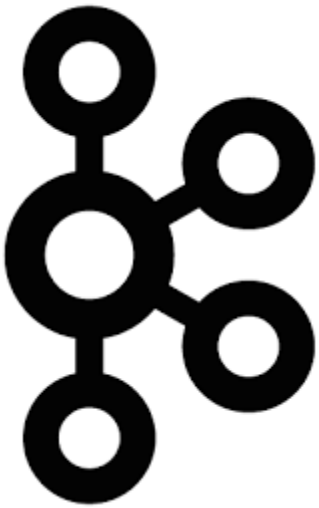
Discussão do artigo: "**Kafka: a Distributed Messaging System for Log Processing**" by Kreps, Jay, Neha Narkhede, and Jun Rao. NetDB 2011, pp. 1–7. 2011

Papers We Love - Brasília

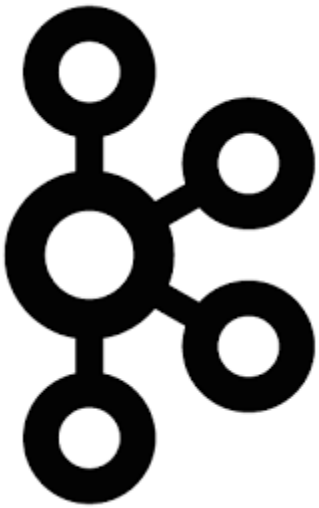
October 7, 2016

What is Kafka?



- A distributed messaging system for handling high amount of data in near real-time.

What is Kafka?



- A distributed messaging system for handling high amount of data in near real-time.
- It focuses on applications that require high-throughput data event processing.

What were some of the motivations to design Kafka

- Overkill of the available solutions (e.g., JMS)

What were some of the motivations to design Kafka

- Overkill of the available solutions (e.g., JMS)
- Lack of support to batch multiples messages into a single request

What were some of the motivations to design Kafka

- Overkill of the available solutions (e.g., JMS)
- Lack of support to batch multiples messages into a single request
- The difficult to partition and to store messages on multiple machines

What were some of the motivations to design Kafka

- Overkill of the available solutions (e.g., JMS)
- Lack of support to batch multiples messages into a single request
- The difficult to partition and to store messages on multiple machines
- The performance of existing messaging systems usually degrades when there are a high volume of message to be consumed

What is the architecture of Kafka?

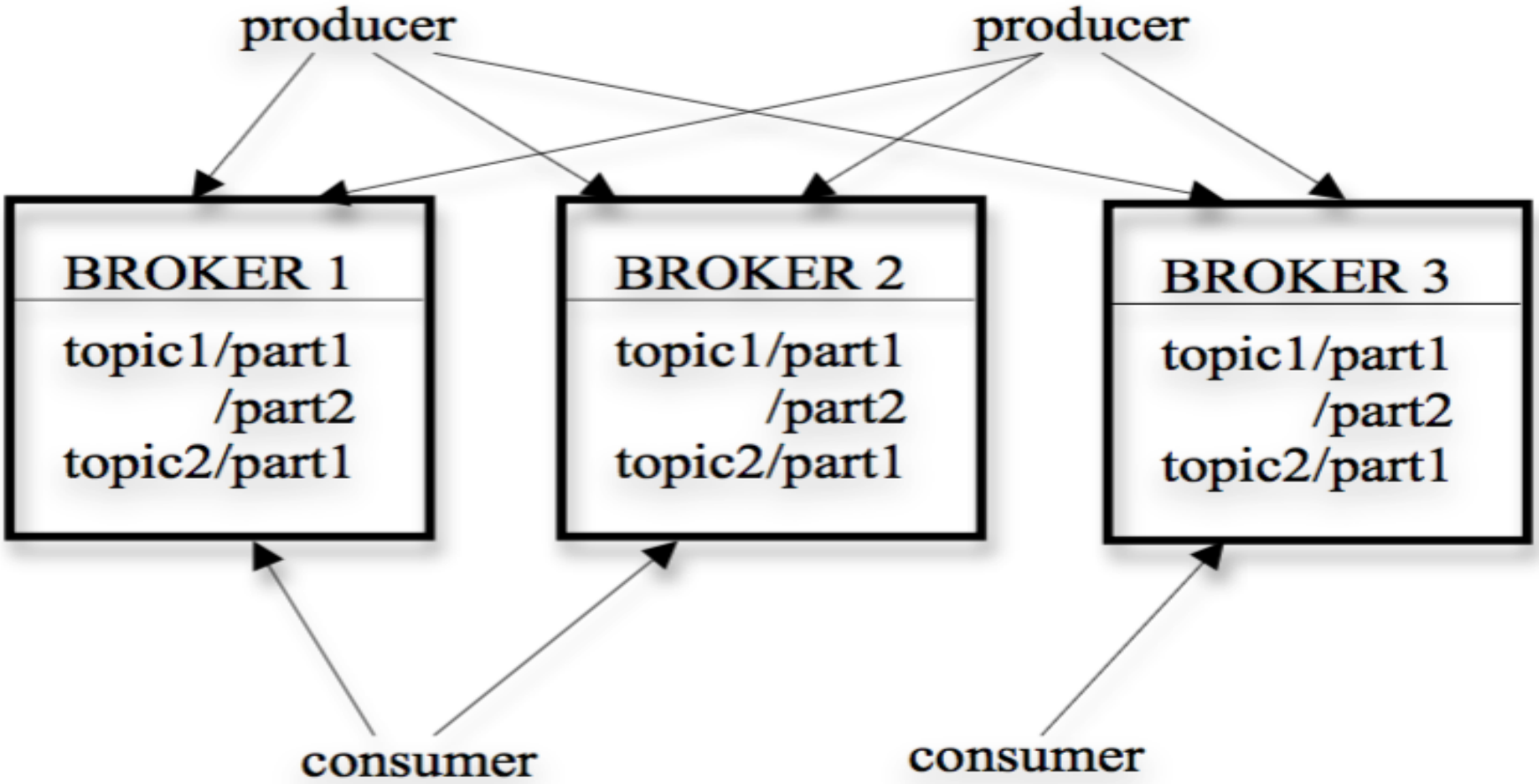


Figure 1: Kafka architecture

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.
- A **topic** is divided into **multiple partitions** stored across different **brokers**.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.
- A **topic** is divided into **multiple partitions** stored across different **brokers**.
- **Brokers** are servers where the published messages are stored.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.
- A **topic** is divided into **multiple partitions** stored across different **brokers**.
- **Brokers** are servers where the published messages are stored.
- A **consumer** can **subscribe** to **one** or **more** topics from the brokers.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.
- A **topic** is divided into **multiple partitions** stored across different **brokers**.
- **Brokers** are servers where the published messages are stored.
- A **consumer** can **subscribe** to **one** or **more** topics from the brokers.
- It is the **consumers** that **pull** the data from the brokers.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.
- A **topic** is divided into **multiple partitions** stored across different **brokers**.
- **Brokers** are servers where the published messages are stored.
- A **consumer** can **subscribe** to **one** or **more** topics from the brokers.
- It is the **consumers** that **pull** the data from the brokers.
- Consumers' are **blocked** until new messages are published to a topic.

What are the core concepts and design decisions of Kafka?

- A **publisher** can publish messages to a particular **topic**.
- A **topic** comprises a **stream of messages** of a particular type.
- A **topic** is divided into **multiple partitions** stored across different **brokers**.
- **Brokers** are servers where the published messages are stored.
- A **consumer** can **subscribe** to **one** or **more** topics from the brokers.
- It is the **consumers** that **pull** the data from the brokers.
- Consumers' are **blocked** until new messages are published to a topic.
- Kafka supports **point-to-point** and **publish/subscribe** delivery models.

Talk is cheap. Show me the code.

— Linus Torvalds

This is an example of the code of a producer

```
1  producer = new Producer (...)
2  message = new Message("Hello world!".getBytes())
3
4  set = new MessageSet(message)
5  producer.send("topic1", set)
```

This is an example of the code of a consumer

```
1    streams = new Consumer.createMessageStreams("topic1",
         1)
2    for (message: streams)
3      doSomethingWith(message)
```

What are the core concepts and design decisions of Kafka? (cont.)

- Each **pull request** from a consumer also **retrieves multiple messages** up to a certain size.

What are the core concepts and design decisions of Kafka? (cont.)

- Each **pull request** from a consumer also **retrieves multiple messages** up to a certain size.
- Kafka **does not cache the messages**, which helps on avoiding the garbage collector's overhead.

What are the core concepts and design decisions of Kafka? (cont.)

- Each **pull request** from a consumer also **retrieves multiple messages** up to a certain size.
- Kafka **does not cache the messages**, which helps on avoiding the garbage collector's overhead.
- **Brokers are stateless**, i.e., they do not keep a state describing the messages consumed by each consumer.

What are the core concepts and design decisions of Kafka? (cont.)

- Each **pull request** from a consumer also **retrieves multiple messages** up to a certain size.
- Kafka **does not cache the messages**, which helps on avoiding the garbage collector's overhead.
- **Brokers are stateless**, i.e., they do not keep a state describing the messages consumed by each consumer.
- Thus, it is the **consumer** that maintains the control (i.e., **offset**) of which message has be consumed.

What are the core concepts and design decisions of Kafka? (cont.)

- Each **pull request** from a consumer also **retrieves multiple messages** up to a certain size.
- Kafka **does not cache the messages**, which helps on avoiding the garbage collector's overhead.
- **Brokers are stateless**, i.e., they do not keep a state describing the messages consumed by each consumer.
- Thus, it is the **consumer** that maintains the control (i.e., **offset**) of which message has be consumed.
- Kafka uses **retention policy** to determine when a message must be deleted.

What are the core concepts and design decisions of Kafka? (cont.)

- Each **pull request** from a consumer also **retrieves multiple messages** up to a certain size.
- Kafka **does not cache the messages**, which helps on avoiding the garbage collector's overhead.
- **Brokers are stateless**, i.e., they do not keep a state describing the messages consumed by each consumer.
- Thus, it is the **consumer** that maintains the control (i.e., **offset**) of which message has be consumed.
- Kafka uses **retention policy** to determine when a message must be deleted.
- Implementing a **pull bash** approach makes easy to a consumer deliberately rewinds to an old offset and re-consume the data.

What are the core concepts and design decisions of Kafka? (cont.)

- Kafka **guarantees** that messages **from a partition** are delivered to a consumer **in order**. However, there is no guarantee on the ordering coming from various partitions.

What are the core concepts and design decisions of Kafka? (cont.)

- Kafka **guarantees** that messages **from a partition** are delivered to a consumer **in order**. However, there is no guarantee on the ordering coming from various partitions.
- Kafka stores a **cyclic redundancy check** (CRC) for each message in order **to avoid data corruption**.

These design decisions allow Kafka to have good performance

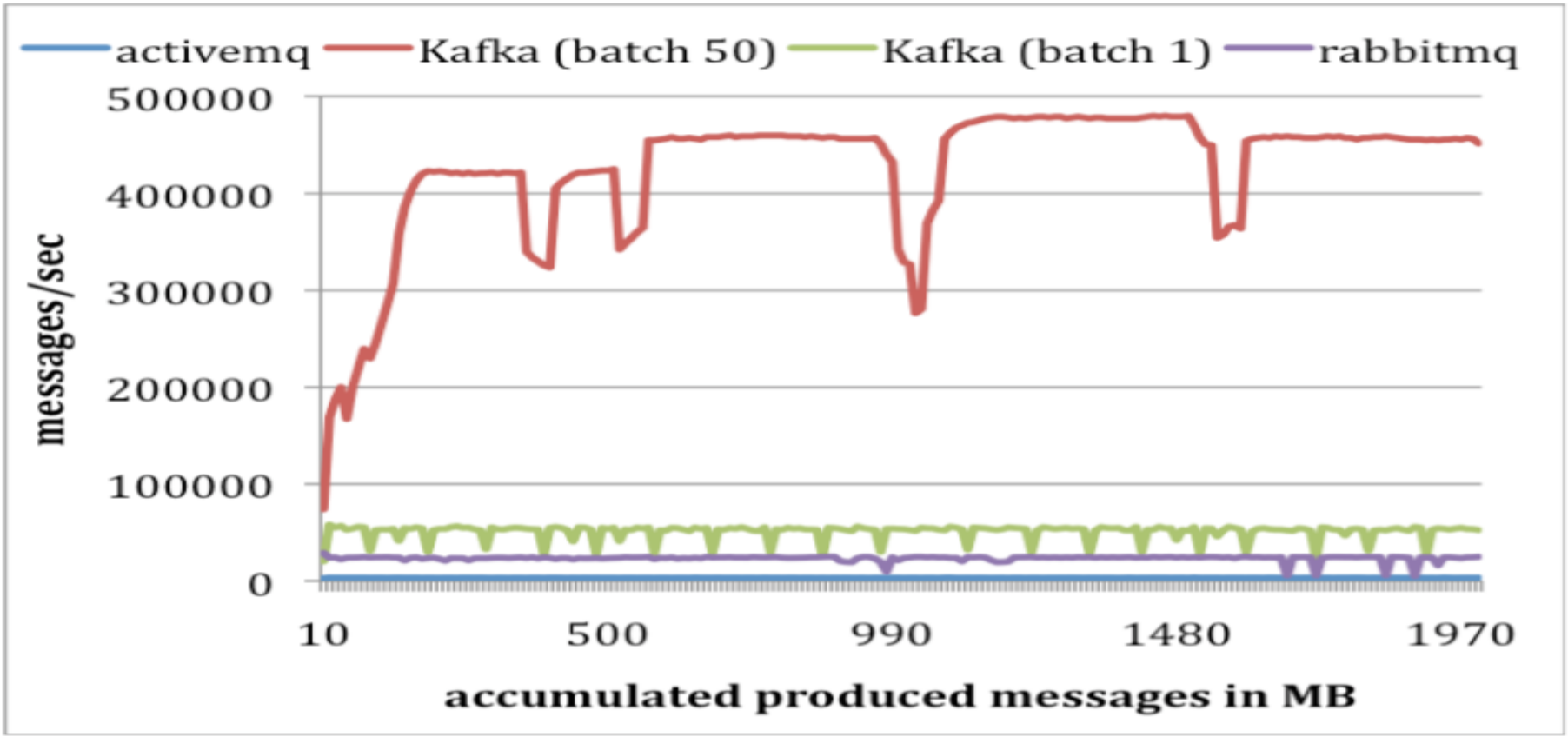


Figure 2: Produce performance

These design decisions allow Kafka to have good performance

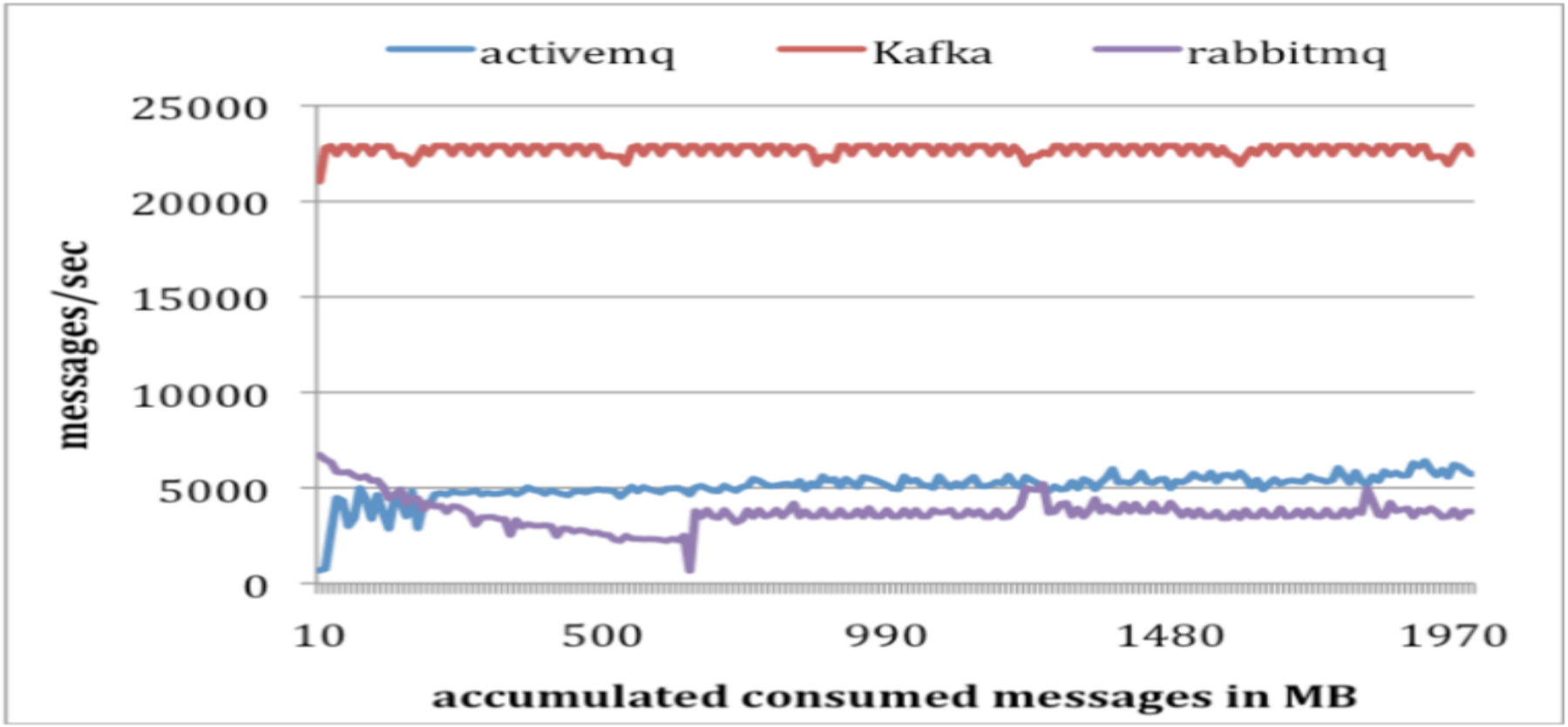


Figure 3: Consumer performance

That's all Folks!