When to use RabbitMQ over Kafka? [closed]

Asked 7 years, 10 months ago Modified 2 months ago

Viewed 275k times

557

I've been asked to evaluate RabbitMQ instead of Kafka but found it hard to find a situation where a message queue is more suitable than Kafka. Does anyone know use cases where a message queue fits better in terms of throughput, durability, latency, or ease-of-use?

apache-kafka rabbitmq message-queue

Share

Improve this question

Follow

edited Sep 23, 2020 at 0:12

Nikhil Banka
91 ● 1 ● 13

asked Feb 10, 2017 at 4:30

Joe
13k ● 39 ● 120 ● 199

12 primarily opinion-based,Many good questions generate some degree of opinion based on expert experience, but answers to this question will tend to be almost entirely based on opinions, rather than facts, references, or specific expertise. – VedantK Feb 10, 2017 at 8:32

3 @Guillaume That's not necessarily true. There a clients for many languages available for Kafka: cwiki.apache.org/confluence/display/KAFKA/Clients Furthermore, Confluent offers many high performant open source Kafka clients in other languages. Check out "Confluent Open Source" offer: confluent.io/product/compare – Matthias J. Sax Feb 10, 2017 at 18:54

3 @MatthiasJ.Sax Both RabbitMQ and kafka have a wealth of clients in many languages, but my point was about official clients. In the link you gave it is written black on white: *we are maintaining all but the jvm client external to the main code base*. Regarding confluent, I am indeed a big user, but the additional clients are through the language agnostic rest API, which although quite awesome does not have the same throughput as the official java client. – Guillaume Feb 10, 2017 at 19:17

3 @Guillaume For "random" open source clients from the community I agree; not all a high performance (it pretty hard to write a good client) -- that why I put "That's not **necessarily** true." ;) However, Confluent's provided C/C++ and Python clients are high throughput and as efficient as the AK Java clients... – Matthias J. Sax Feb 10, 2017 at 20:02

4 I would recommend reading this blog: jack-vanlightly.com/blog/2017/12/4/… – roottraveller Aug 21, 2019 at 6:49

16 Answers

Sorted by: Highest score (default)

918

RabbitMQ is a solid, general-purpose **message broker** that supports several protocols such as AMQP (1.0 and 0.9.1), MQTT, STOMP, etc. It can handle high throughput. A common use case for RabbitMQ is to handle background jobs or long-running task, such as file scanning, image scaling or PDF conversion. RabbitMQ is also used between microservices, where it serves as a means of communicating between applications, avoiding bottlenecks passing messages.

Update since the release of RabbitMQ Streams: RabbitMQ stands out for offering various queue types, each designed for specific messaging needs. Traditionally, all RabbitMQ's queues would delete messages once consumed and acknowledged, which made them unsuitable for long-term storage or replaying messages - but that changed with RabbitMQ v3.9 when another queue type was introduced, Stream Queues.

Stream Queues are persistent and replicated, and like traditional queues, they buffer messages from producers for consumers. Under the hood, Streams model an append-only log that's immutable. In this context, messages written to a Stream can't be erased; they can only be read. To read messages from a Stream in RabbitMQ, one or more consumers subscribe to it and read the same message as many times as they want. This functionality mirrors Kafka's behavior, making RabbitMQ's stream queues attractive for users seeking Kafka-like features in a RabbitMQ environment.

Kafka is a message bus optimized for **high-throughput ingestion data streams** and replay. Use Kafka when you need to move a large amount of data, process data in real-time, or analyze data over a time period. In other words, where data need to be collected, stored, and handled. An example is when you want to track user activity on a webshop and generate suggested items to buy. Another example is data analysis for tracking, ingestion, logging or security.

Kafka and RabbitMQ Streams can be seen as **durable message brokers** where applications can process and re-process streamed data on disk. With both Kafka and RabbitMQ Streams, the data sent is stored until a specified retention period has passed, either a period of time or a size limit. The message stays in the queue until the retention period/size limit is exceeded, meaning the message is not removed once it’s consumed. Instead, it can be replayed or consumed multiple times, which is a setting that can be adjusted.

If you are planning to use replay in RabbitMQ or Apache Kafka, ensure that you are using it correctly and for the correct reason! Replaying an event multiple times that should just happen a single time; e.g. if you happen to save a customer order multiple times, is not ideal in most usage scenarios. Where a replay does come in handy is when you have a bug in the consumer that requires deploying a newer version, and you need to re-processing some or all of the messages.

Kafka has a very simple routing approach. RabbitMQ has other options if you need to route your messages in a more complex ways, to your consumers. Use Kafka if you need to support batch consumers that could be offline or consumers that want messages at low latency.

In order to understand how to read data from Kafka, we first need to understand its consumers and consumer groups. Partitions allow you to parallelize a topic by splitting the data across multiple nodes. Each record in a partition is assigned and identified by its unique offset. This offset points to the record in a partition. In the latest version of Kafka, Kafka maintains a numerical offset for each record in a partition. A consumer in Kafka can either automatically commit offsets periodically, or it can choose to control this committed position manually. RabbitMQ will keep all states about consumed/acknowledged/unacknowledged messages. I find Kafka more complex to understand than the case of RabbitMQ, where the message is simply removed from the queue once it's acked. RabbitMQ does also require less recourses the Kafka and works well for most use cases.

RabbitMQ's queues are fastest when they're empty, while Kafka retains large amounts of data with very little overhead - Kafka is designed for holding and distributing large volumes of messages.

Kafka is built from the ground up with horizontal scaling (scale by adding more machines) in mind, while RabbitMQ is mostly designed for vertical scaling (scale by adding more power).

RabbitMQ has a built-in user-friendly interface that lets you monitor and handle your RabbitMQ server from a web browser. Among other things, queues, connections, channels, exchanges, users and user permissions can be handled - created, deleted and listed in the browser and you can monitor message rates and send/receive messages manually. Kafka has a number of open-source tools, and also some commercial ones, offering the administration and monitoring functionalities. I would say that it's easier/gets faster to get a good understanding of RabbitMQ.

In general, if you want a traditional pub-sub message broker, the obvious choice is RabbitMQ (classic or quorum queues), as it will most probably scale more than you will ever need it to scale. I would have chosen

RabbitMQ if my requirements were simple enough to deal with system communication through channels/queues. I would also have chosen it for streaming of data if there is not an insane amount of data to stream. You can also get started over a day.

There are two main situations where I would choose RabbitMQ; For long-running tasks, when I need to run reliable background jobs. And for communication and integration within, and between applications, i.e as middleman between microservices; where a system simply needs to notify another part of the system to start to work on a task, like ordering handling in a webshop (order placed, update order status, send order, payment, etc.).

*In general, if you want a framework for storing, reading (re-reading, replay), and analyzing a huge amount of streaming data, use Apache Kafka.** It's ideal for systems that are audited or those that need to store lots of messages permanently. These can also be broken down into two main use cases for analyzing data (tracking, ingestion, logging, security etc.) or real-time processing.

More reading, use cases and some comparison data can be found here: https://www.cloudamqp.com/blog/2019-12-12-when-to-use-rabbitmq-or-apache-kafka.html

Also recommending the industry paper: "Kafka versus RabbitMQ: A comparative study of two industry reference publish/subscribe implementations": http://dl.acm.org/citation.cfm?id=3093908

I do work at a company providing both Apache Kafka and RabbitMQ as a Service.

Share Improve this answer Follow

edited Oct 17 at 9:16

answered Feb 10, 2017 at 8:10

Lovisa Johansson
10.4k ● 1 ● 19 ● 21

---

45 What does "high-ingress" mean? – Martin Thoma Jul 4, 2017 at 6:56

44 high-ingress = high-throughput ingestion – jbustamovej Jul 9, 2017 at 13:32

8 I question your point about RabbitMQ "mostly designed for vertical scaling". How so... – Ryan.Bartsch Jul 19, 2018 at 10:14

59 Horizontal scaling (scale by adding more machines) does not give you a better performance in RabbitMQ. Best performance is received when you do vertical scaling (scale by adding more power). I know this because I have been working with thousands of RabbitMQ clusters for many years now. You can do horizontal scaling in Rabbit, but that means that you also set up clustering between your nodes, which will slow down your setup. I wrote a guide about best practice for high performance vs high availability in RabbitMQ: cloudamqp.com/blog/2017-12-29-part1-rabbitmq-best-practice.html – Lovisa Johansson Sep 6, 2018 at 8:05

9 "...while Kafka doesn't, it assumes the consumer keep tracks of what's been consumed and not." This is incorrect. Kafka keeps track of the messages consumed by each individual consumer. – jucardi Feb 7, 2019 at 0:56

66

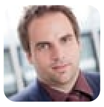
I hear this question every week... While RabbitMQ (like IBM MQ or JMS or other messaging solutions in general) is used for traditional messaging, Apache Kafka is used as streaming platform (messaging + distributed storage + processing of data). Both are built for different use cases.

You can use Kafka for "traditional messaging", but not use MQ for Kafka-specific scenarios.

The article “**Apache Kafka vs. Enterprise Service Bus (ESB)—Friends, Enemies, or Frenemies?** (https://www.confluent.io/blog/apache-kafka-vs-enterprise-service-bus-esb-friends-enemies-or-frenemies/)” discusses why Kafka is not competitive but complementary to integration and messaging solutions (including RabbitMQ) and how to integrate both.

Share Improve this answer Follow

answered Jul 27, 2018 at 7:17

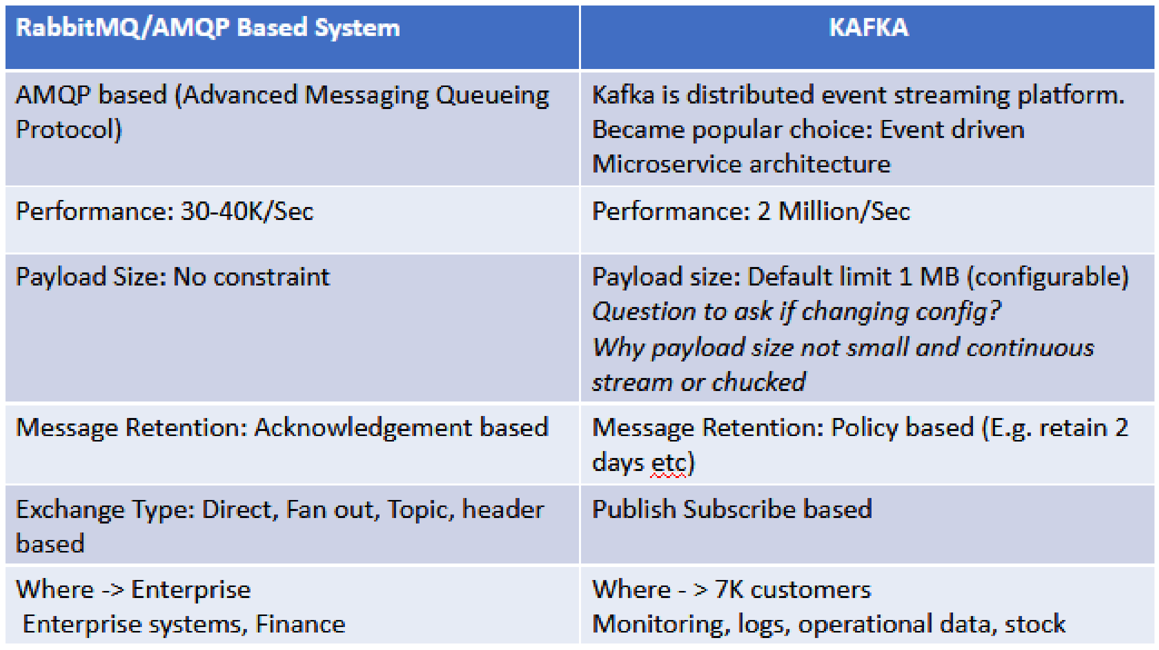


Kai Wähner
5,430 ● 4 ● 37 ● 33

It says it's complementary to an already existing MQ and ESB solutions (because rebuilding is probably difficult), but that newer solutions are all Kafka. – Desperado Aug 27, 2022 at 5:04

62

5 Major differences between Kafka and RabbitMQ, customer who are using them:



RabbitMQ/AMQP Based System	KAFKA
AMQP based (Advanced Messaging Queueing Protocol)	Kafka is distributed event streaming platform. Became popular choice: Event driven Microservice architecture
Performance: 30-40K/Sec	Performance: 2 Million/Sec
Payload Size: No constraint	Payload size: Default limit 1 MB (configurable) *Question to ask if changing config? Why payload size not small and continuous stream or chucked*
Message Retention: Acknowledgement based	Message Retention: Policy based (E.g. retain 2 days etc)
Exchange Type: Direct, Fan out, Topic, header based	Publish Subscribe based
Where -> Enterprise Enterprise systems, Finance	Where - > 7K customers Monitoring, logs, operational data, stock

Which messaging system to choose or should we change our existing messaging system?

There is no one answer to above question. One possible approach to review when you have to decide which messaging system or should you change existing system is to "Evaluate scope and cost"

Share Improve this answer

Follow

answered May 9, 2019 at 5:28

Shishir

863 ● 7 ● 7

17 Where is your source for this information? I don't agree with your answer regarding performance in RabbitMQ - that depends on the number of queues, connections etc.
– Lovisa Johansson May 9, 2019 at 14:09

Correct. But average variance range is similar as stated above. There are scenario where it does better or worse than above mentioned range. Refer Rabbitmq blog. Latest data points might have changed rabbitmq.com/blog/2012/04/25/…
– Shishir May 10, 2019 at 15:18

@Shishir - Could you share more details/links that explain the different message exchange types - direct, fan out, pub/sub etc? These sound to be helpful in determining the right messaging platform for given requirements. Thanks – Andy Dufresne May 13, 2019 at 5:35

2 @Shishir a link from 2012, might have changed, yes. – Lovisa Johansson Nov 21, 2019 at 9:39

2 @AndyDufresne, a bit late, but here is a link: [cloudamqp.com/blog/…]() – Lovisa Johansson Nov 21, 2019 at 9:39

54

One critical difference that you guys forgot is RabbitMQ is push based messaging system whereas Kafka is pull based messaging system. This is important in the scenario where messaging system has to satisfy disparate types of consumers with different processing capabilities. With Pull based system the consumer can consume based on their capability where push systems will push the messages irrespective of the state of consumer thereby putting consumer at high risk.

Share Improve this answer Follow

answered Jun 3, 2019 at 4:39

kanishka vatsa
2,294 ● 20 ● 8

24 You can achieve both pull and push with RabbitMQ – Nikolas Mar 4, 2020 at 11:39

32

I know it's a bit late and maybe you already, indirectly, said it, but again, Kafka is not a queue at all, it's a log (as someone said above, poll based).

To make it simple, the most obvious use case when you should prefer RabbitMQ (or any queue techno) over Kafka is the following one :

You have multiple consumers consuming from a queue and whenever there is a new message in the queue and an available consumer, you want this message to be processed. If you look closely at how Kafka works, you'll notice it does not know how to do that, because of partition scaling, you'll have a consumer dedicated to a partition and you'll get into starvation issue. Issue that is easily avoided by using simple queue techno. You can think of using a thread that will dispatch the different messages from same partition, but again, Kafka does not have any selective acknowledgment mechanisms.

The most you could do is doing as those guys and try to transform Kafka as a queue : https://github.com/softwaremill/kmq

Yannick

Share Improve this answer Follow

edited Aug 20, 2019 at 20:08
tjg184
4,658 ● 1 ● 31 ● 56

answered Jul 4, 2019 at 19:01

Yannick
1,398 ● 3 ● 16 ● 26

This is one of the main reasons we decided to use RabbitMQ instead of kafka in our microservice based system. It was very important for use to be flexible in increasing and decreasing consumers according to the message income rate, in RabbitMQ this is easy you simply start more consumers, no repartition is needed like in kafka. – Siraf Apr 7 at 11:07

31

RabbitMQ is a traditional general purpose message broker. It enables web servers to respond to requests quickly and deliver messages to multiple services. Publishers are able to publish messages and make them available to queues, so that consumers can retrieve them. The communication can be either asynchronous or synchronous.

On the other hand, **Apache Kafka** is not *just* a message broker. It was initially designed and implemented by LinkedIn in order to serve as a message queue. Since 2011, Kafka has been open sourced and quickly evolved into a distributed streaming platform, which is used for the implementation of real-time data pipelines and streaming applications.

> It is horizontally scalable, fault-tolerant, wicked fast, and runs in production in thousands of companies.

Modern organisations have various data pipelines that facilitate the communication between systems or services. Things get a bit more complicated when a reasonable number of services needs to communicate with each other at real time.

The architecture becomes complex since various integrations are required in order to enable the inter-communication of these services. More precisely, for an architecture that encompasses m source and n target services, n x m distinct integrations need to be written. Also, every integration comes with a different specification, meaning that one might require a different protocol (HTTP, TCP, JDBC, etc.) or a different data representation (Binary, Apache Avro, JSON, etc.), making things even more challenging. Furthermore, source services might address increased load from connections that could potentially impact latency.

Apache Kafka leads to more simple and manageable architectures, by decoupling data pipelines. Kafka acts as a high-throughput distributed system where source services push streams of data, making them available for target services to pull them at real-time.

Also, a lot of open-source and enterprise-level User Interfaces for managing Kafka Clusters are available now. For more details refer to my articles **Overview of UI monitoring tools for Apache Kafka clusters** and **Why Apache Kafka?**

The decision of whether to go for RabbitMQ or Kafka is dependent to the requirements of your project. In general, if you want a simple/traditional pub-sub message broker then go for RabbitMQ. If you want to build an event-driven architecture on top of which your organisation will be acting on events at real-time, then go for Apache Kafka as it provides more functionality for this architectural type (for example Kafka Streams or ksqlDB).

Share Improve this answer Follow

edited Apr 10, 2020 at 21:36

answered Apr 7, 2019 at 13:47

Giorgos Myrianthous
39.6k ● 21 ● 152 ● 170

19

Use RabbitMQ when:

- You don't have to handle with Bigdata and you prefer a convenient in-built UI for monitoring
- No need of automatically replicable queues
- No multi subscribers for the messages- Since unlike Kafka which is a log, RabbitMQ is a queue and messages are removed once consumed and acknowledgment arrived
- If you have the requirements to use Wildcards and regex for messages
- If defining message priority is important

In Short: RabbitMQ is good for simple use cases, with low traffic of data, with the benefit of priority queue and flexible routing options. For massive data and high throughput use Kafka.

Share Improve this answer Follow

answered Aug 24, 2019 at 21:44

Anjali Shyamsundar
555 ●1 ●5 ●14

7 Multi subscribers is handled fine, not in a single queue but fanning out to multiple and potentially dynamic queues. Rabbit is certainly not just for 'simple use cases' it's for a completely different paragdim but no less complex than large data sets that need retaining for long periods. Can you expand on the message priority part? – Owen Feb 13, 2020 at 6:59

19

I'll provide an objective answer based on my experience with both, I'll also skip the theory behind them, assuming you already know it and/or other answers has already provided enough.

RabbitMQ: I'd pick this one if my requirements are simple enough to deal with system communication through channels/queues, retention and streaming is not a requirement. For e.g. When the manufacture system built the asset it notifies the agreement system to configure the contracts and so on.

Kafka: Event sourcing requirement mainly, when you might need to deal with streams (sometimes infinite), huge amount of data at once properly balanced, replay offsets in order to ensure a given state and so on. Keep in mind that this architecture brings more complexity as well, since it includes concepts such as topics/partitions/brokers/tombstone messages, etc. as a first class importance.

Share Improve this answer Follow

edited Aug 27, 2023 at 23:13

answered Jun 20, 2019 at 11:34

irobson
865 ● 8 ● 20

RabbitMQ introduced the streams feature in v3.9.0 [rabbitmq.com/docs/streams]() which brings kafka like queues with streaming semantics to RabbitMQ – Siraf Apr 8 at 11:21

11

If you have complex routing needs and want a built-in GUI to monitor the broker, then RabbitMQ might be best for your application. Otherwise, if you're looking for a message broker to handle high throughput and provide access to stream history, Kafka is the likely the better choice.

Share Improve this answer Follow

answered Jun 9, 2020 at 19:06

Maria Hatfield
111 ● 1 ● 2

2 [+1] Good explanation, I am sure you have been using them in your projects, could you name some that have used either of them in mounting application message systems?
– GingerHead Jun 9, 2020 at 19:42

3 @GingerHead We worked with a radio company that used RabbitMQ for their GUI and ease of setup. It was great for developers to easily check on the status of their microservices. The same company also used Kafka for high-volume streams of data that needed to have retention time of over three days. If you are interested in reading more about the differences between the two technologies here is an article I wrote on the topic: [Kafka vs. RabbitMQ article]().
– Maria Hatfield Jun 11, 2020 at 16:39

7

Scaling both is hard in a distributed fault tolerant way but I'd make a case that it's much harder at massive scale with RabbitMQ. It's not trivial to understand Shovel, Federation, Mirrored Msg Queues, ACK, Mem issues, Fault tollerance etc. Not to say you won't also have specific issues with Zookeeper etc on Kafka but there are less moving parts to manage. That said, you get a Polyglot exchange with RMQ which you don't with Kafka. If you want streaming, use Kafka. If you want simple IoT or similar high volume packet delivery, use Kafka. It's about smart consumers. If you want msg flexibility and higher reliability with higher costs and possibly some complexity, use RMQ.

Share Improve this answer Follow

answered Apr 6, 2019 at 18:11

user3919920
365 ● 3 ● 8

4 I don't agree how you infer RMQ has "some complexity" as if to say Kafka has less complexity. – Cory Robinson Sep 5, 2019 at 0:15

7

The short answer is "message acknowledgements". RabbitMQ can be configured to require message acknowledgements. If a receiver fails the message goes back on the queue and another receiver can try again. While you can accomplish this in Kafka with your own code, it works with RabbitMQ out of the box.

In my experience, if you have an application that has requirements to query a stream of information, Kafka and KSql are your best bet. If you want a queueing system you are better off with RabbitMQ.

Share Improve this answer Follow

answered Jun 8, 2020 at 12:08

GlennSills
4,157 ● 27 ● 29

5

Technically, Kafka offers a huge superset of features when compared to the set of features offered by Rabbit MQ.

If the question is

Is Rabbit MQ technically better than Kafka?

then the answer is

No.

However, if the question is

Is Rabbit MQ better than Kafka from a business perspective?

then, the answer is

Probably 'Yes', in some business scenarios

Rabbit MQ can be better than Kafka, from a business perspective, for the following reasons:

1. Maintenance of legacy applications that depend on Rabbit MQ
2. Staff training cost and steep learning curve required for implementing Kafka
3. Infrastructure cost for Kafka is higher than that for Rabbit MQ.
4. Troubleshooting problems in Kafka implementation is difficult when compared to that in Rabbit MQ implementation.
    - A Rabbit MQ Developer can easily maintain and support applications that use Rabbit MQ.
    - The same is not true with Kafka. Experience with just Kafka development is not sufficient to

maintain and support applications that use Kafka. The support personnel require other skills like zoo-keeper, networking, disk storage too.

Share Improve this answer Follow

answered Jun 27, 2020 at 9:44

Gopinath
4,897 ● 1 ● 16 ● 19

4

The only benefit that I can think of is Transactional feature, rest all can be done by using Kafka

Share Improve this answer Follow

answered Mar 15, 2019 at 23:22

RB7
475 ● 7 ● 10

6 Kafka has transactions – OneCricketeer Jan 1, 2020 at 12:28

1

Apache Kafka is a popular choice for powering data pipelines. Apache kafka added kafka stream to support popular etl use cases. KSQL makes it simple to transform data within the pipeline, readying messages to cleanly land in another system. KSQL is the streaming SQL engine for Apache Kafka. It provides an easy-to-use yet powerful interactive SQL interface for stream processing on Kafka, without the need to write code in a programming language such as Java or Python. KSQL is scalable, elastic, fault-tolerant, and real-time. It supports a wide range of streaming operations, including data filtering, transformations, aggregations, joins, windowing, and sessionization.

https://docs.confluent.io/current/ksql/docs/index.html

Rabbitmq is not a popular choice for etl systems rather for those systems where it requires simple messaging systems with less throughput.

Share Improve this answer Follow

answered Jan 2, 2020 at 15:45

Salona Sinha
21 ● 2

1

I realize that this is an old question, but one scenario where RabbitMQ might be a better choice is when dealing with data redaction.

With RabbitMQ, by default once the message has been consumed, it's deleted. With Kafka, by default, messages are kept for a week. It's common to set this to a much longer time, or even to never delete them.

While both products can be configured to retain (or not retain) messages, if CCPA or GDPR compliance is a concern, I'd go with RabbitMQ.

Share Improve this answer Follow

answered Feb 1, 2020 at 2:59

Merkle Groot
906 ● 5 ● 10

-4

The most voted answer covers most part but I would like to high light use case point of view. Can kafka do that rabbit mq can do, answer is yes but can rabbit mq do everything that kafka does, the answer is no.

The thing that rabbit mq cannot do that makes kafka apart, is distributed message processing. With this now read back the most voted answer and it will make more sense.

To elaborate, take a use case where you need to create a messaging system that has super high throughput for example "likes" in facebook and You have chosen rabbit mq for that. You created an exchange and queue and a consumer where all publishers (in this case FB users) can publish 'likes' messages. Since your throughput is high, you will create multiple threads in consumer to process messages in parallel but you still bounded by the hardware capacity of the machine where consumer is running. Assuming that one consumer is not sufficient to process all messages - what would you do?

- Can you add one more consumer to queue - no you cant do that.
- Can you create a new queue and bind that queue to exchange that publishes 'likes' message, answer is no cause you will have messages processed twice.

That is the core problem that kafka solves. It lets you create distributed partitions (Queue in rabbit mq) and distributed consumer that talk to each other. That ensures your messages in a topic get processed by consumers distributed in various nodes (Machines).

Kafka brokers ensure that messages get load balanced across all partitions of that topic. Consumer group make sure that all consumer talk to each other and message does not get processed twice.

But in real life you will not face this problem unless your throughput is seriously high because rabbit mq can also process data very fast even with one consumer.

Share Improve this answer Follow

edited Jan 2, 2021 at 14:18
Gajus
73.5k ● 80 ● 294 ● 469

answered Jun 8, 2020 at 20:32
Amit Sharma
41 ● 1 ● 2

---

5 "*...Can you add one more consumer to queue - no you can't do that....*", why can't we add more than one consumer to the same queue in rabbitmq? RabbitMQ says we can here clearly. The messages are delivered to multiple consumers in a round-robin way. – SkrewEverything Aug 16, 2020 at 17:13

5 @SkrewEverything you absolutely can. This entire answer is based on a wrongful assumption that you cannot. – Gajus Jan 2, 2021 at 14:20

4 Rabbitmq official website -> tutorial number 2 (workers) contradicts you – Martin Jan 30, 2021 at 19:33

1 You can increase and decrease the number of RabbitMQ consumers anytime without doing anything in RabbitMQ's

queue! This is one of the main reasons we used RabbitMQ instead of kafka! – Siraf Apr 7 at 11:19