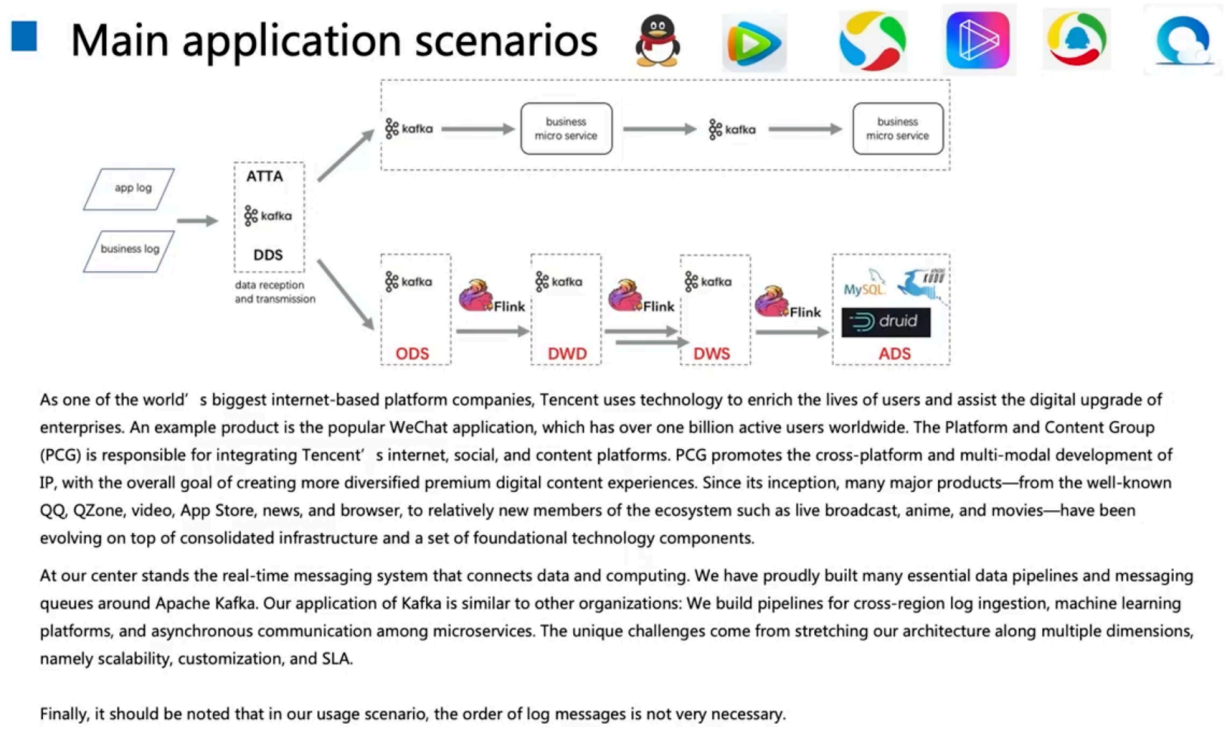
Tencent's Kafka

Overall structure



As one of the world' s biggest internet-based platform companies, Tencent uses technology to enrich the lives of users and assist the digital upgrade of enterprises. An example product is the popular WeChat application, which has over one billion active users worldwide. The Platform and Content Group (PCG) is responsible for integrating Tencent' s internet, social, and content platforms. PCG promotes the cross-platform and multi-modal development of IP, with the overall goal of creating more diversified premium digital content experiences. Since its inception, many major products—from the well-known QQ, QZone, video, App Store, news, and browser, to relatively new members of the ecosystem such as live broadcast, anime, and movies—have been evolving on top of consolidated infrastructure and a set of foundational technology components.

At our center stands the real-time messaging system that connects data and computing. We have proudly built many essential data pipelines and messaging queues around Apache Kafka. Our application of Kafka is similar to other organizations: We build pipelines for cross-region log ingestion, machine learning platforms, and asynchronous communication among microservices. The unique challenges come from stretching our architecture along multiple dimensions, namely scalability, customization, and SLA.

Finally, it should be noted that in our usage scenario, the order of log messages is not very necessary.

Overall, the key concept of Tencent's logic is to get the logs and push them into two directions. One direction is to using these logs or events to drive its own business service. Another way is to store these information into Tencent's own data base.

Here are some corresponding phrases of the acronyms.

ODS: operational data store

DWD: Data Warehouse Design

DWS: Data Warehouse System

ADS: Alternate data stream

Some detail

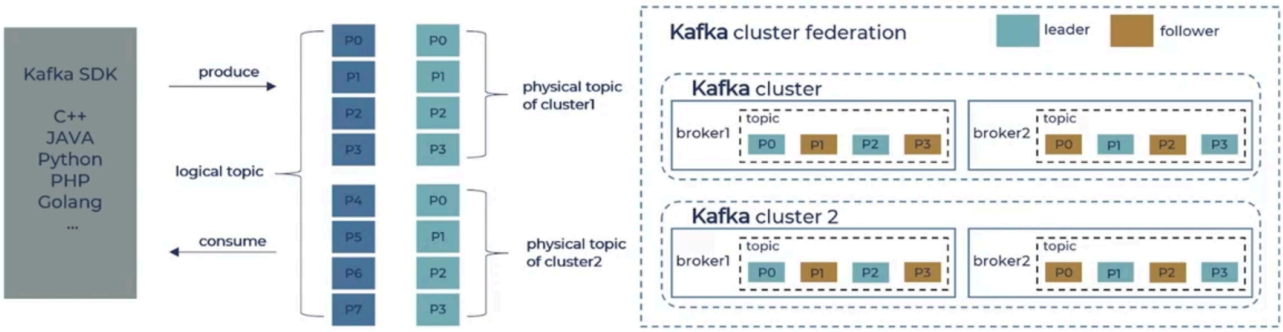
Tencent's demand

- Support for very large-scale clusters
- Topic partition needs to support lossless scaling down
- Low latency and high availability

Original federal Kafka design

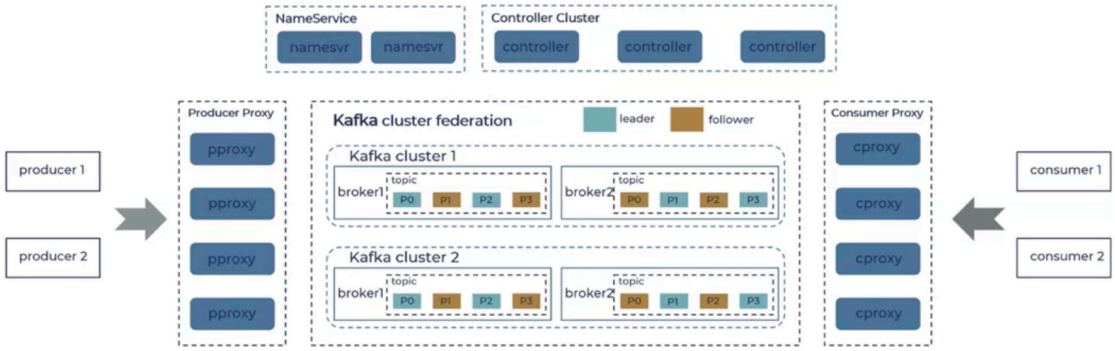
Architecture evolution process in using Apache kafka

first federal kafka design



Example: A logical topic with eight partitions (P0–P7) is distributed to two physical clusters each with four partitions (P0–P3)

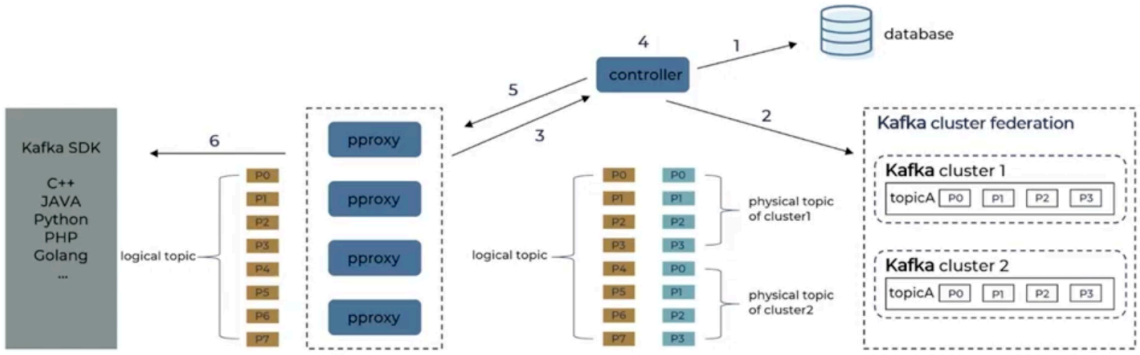
Key components and workflows



On the one hand We add Two proxy services, one for the producer (pproxy) and another for the consumer (cproxy), implement the core protocols of the Kafka broker. They are also responsible for mapping logical topics to their physical incarnation. The application uses the same Kafka SDK to connect directly to the proxy, which acts as a broker.

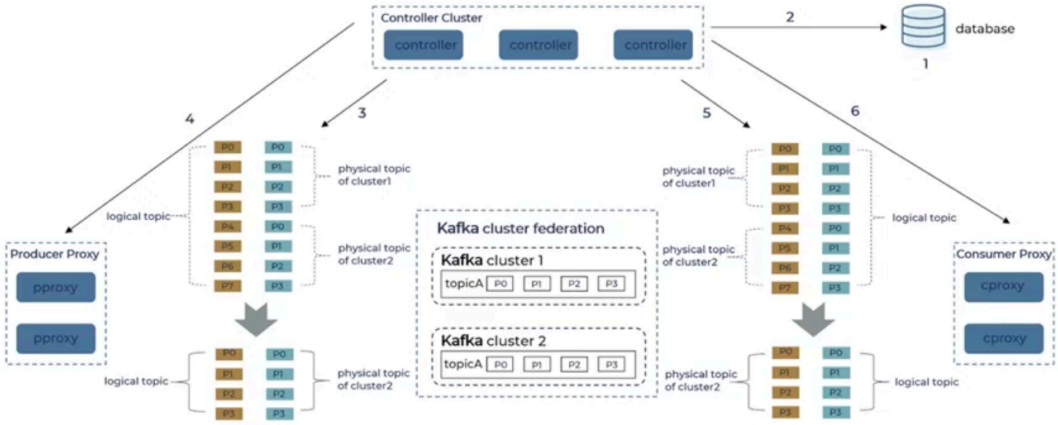
On the other hand We extract the logic of the Kafka controller node (such as topic metadata) into a separate service, which is also called "the controller," but it is different from Kafka' s own controller functionality. This service is responsible for collecting the Metadata of physical clusters, composing the partition information logical topics, and then publishing it to the proxy brokers.

Cluster provisioning workflow



1. Controller reads the mapping between logical topic and physical topic from the config database.
2. Controller scans metadata of all physical topics from each Kafka cluster.
3. Controller finds the list of available pproxy from heartbeat message.
4. Controller composes the metadata for the logical topics.
5. Controller pushes both the topic mapping and topic metadata to proxy brokers.
6. The metadata is sent to the Kafka SDK.

Cluster decommission workflow



1. Controller marks the cluster (cluster 1) to be decommissioned in read-only mode in the config database.
2. Controller picks up this state change when it fetches the metadata from the database.
3. Controller reconstructs metadata for producers, removing cluster 1 from the physical clusters.
4. Controller pushes a new version of producer metadata to the proxy. At this moment, data stops being written to cluster 1 , while the consuming side is not impacted.
5. While all topics in the cluster are expired, the controller reconstructs the consumer metadata again and removes the physical topics no longer valid.
6. Consumer proxy loads the new metadata and now when the new consumer arrives, it will no longer pick up cluster 1. This completes the decommission operation.

Advantages and drawbacks of Cluster federation

Advantages

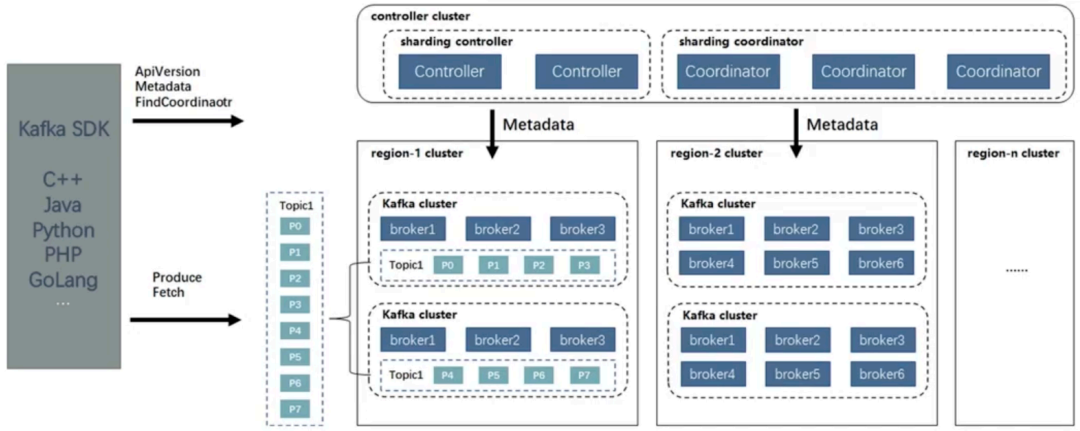
- The topic partition can achieve lossless expansion and contraction
- Disaster tolerance can be achieved at the sub-cluster level

Drawbacks

- Cannot support a large number of topics
- Proxy service needs to be implied to all Kafka protocols (this is a quiet huge job.) and hard to add new feature from the Apache Kafka community.
- Latency, due to the proxy.

Tencent's Kafka

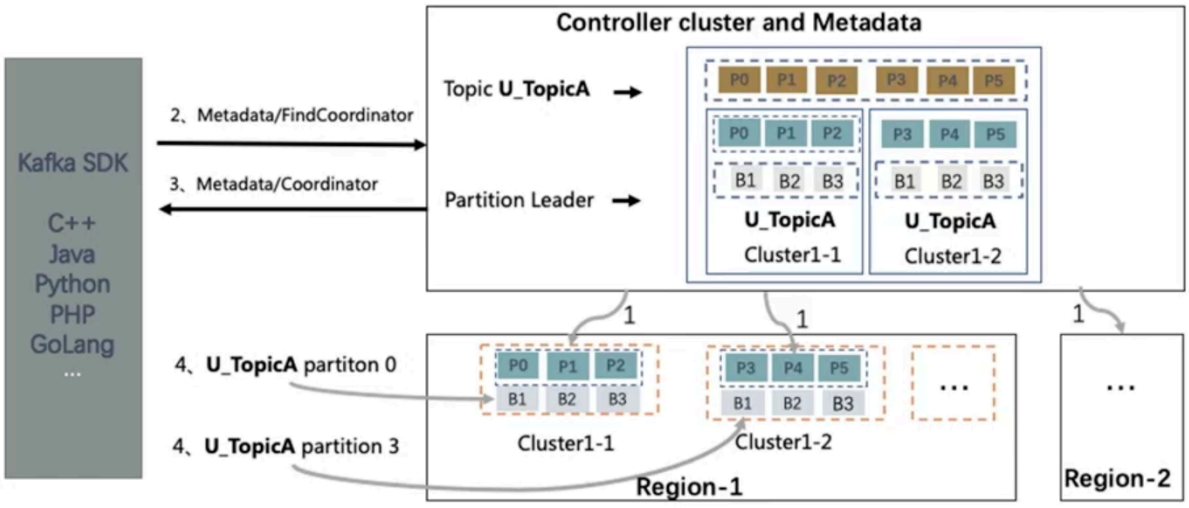
The improved version of the Cluster Federation（support a large number of topics）



In order to make kafka sdk get a merged metadata, we added a new module "Controller cluster", which is a complete Kafka cluster with zookeeper, but there is no broker role on it. On this "Controller cluster", we have mounted multiple kafka clusters with Brokers, and divided these kafka clusters into regions. In operation, we ensure that a topic partition will be allocated to all sub-regions in a region, but cross-region are not allowed. **Therefore, this improved federated cluster can achieve the purpose of expanding the partitions by expanding the sub-clusters under the group where the topic is located when the number of topic partitions is insufficient. When the number of topics in the federated cluster is insufficient, the purpose of expanding the topic can be achieved by expanding the group.**

"Controller cluster" will obtain the metadata of all sub-clusters, and sort and merge the metadata of the same topic to obtain a complete metadata and send it to kafka SDK. so that the message production without transaction is already available Up. Since there is no correlation between the sub-clusters and a consumer group tends to subscribe to multiple topics, the management of consumer groups and transactions must be implemented on the "Controller Cluster".

How does the new Cluster Federation work ?



Produce/Fetch request for partition 0 and partition 3 of **U_TopicA**

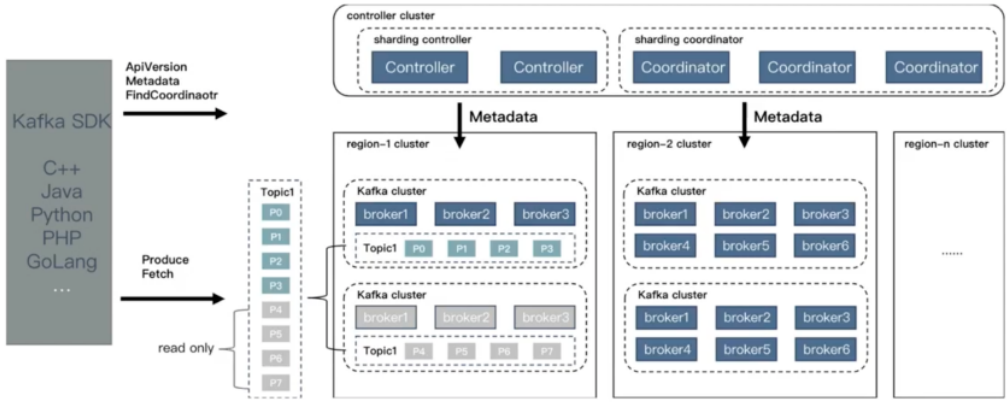
1、After the "controller Cluster" is started, the metadata of all sub-clusters will be merged and sorted by Topic

2、The SDK obtains metadata from the "Controller Cluster" cluster

3、"Controller Cluster" will return the merged topic metadata

4、The SDK normally reads and writes messages based on metadata, but Metadata Update and FindCoordiantor will only request from the " controller Cluster".

Improve the availability of cluster services (1)

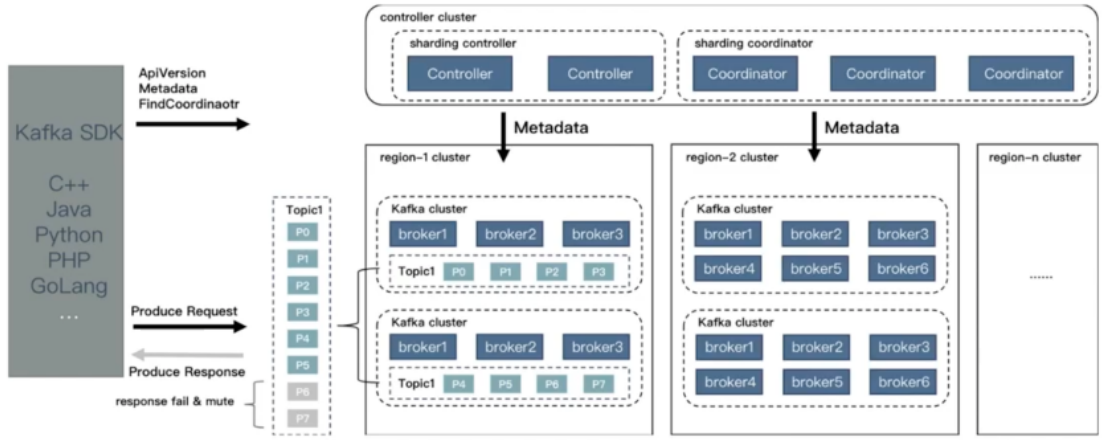


1、when Controller cluster discovers failures in sub-clusters, it will inhibit reading and writing to the partitions on those sub-clusters

2、when Kafka sdk discovers the status of the partitions is "inhibit writing", it will stop writing to those partitions

change the partition status and the ability to scaling down the partition. We have submitted KIP-694 about it.

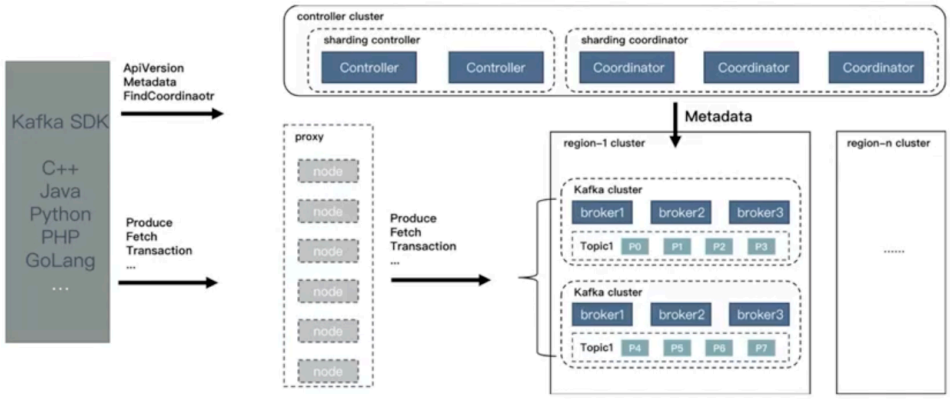
Improve the availability of cluster services (2)



when the kafka sdk discovers slow partition or high failure rate, it will mute these partitions

the ability of kafka sdk muting the abnormal partitions , we have submitted kip-693 about it

Incorporate new features of the Apache Kafka community



Due to the existence of the proxy service, we cannot integrate the new features of the community well. , So we have removed the proxy service in the new architecture.
Now kafka SDK can obtain the aggregated metadata, so it can directly initiate a request to the real Broker.

When we designed the federation mode, kafka 2.8.0 was not released, so our overall design idea is to be better compatible with the features of KIP 500 in the future, such as the process can be split into different roles, and Controller unified management metadata.

Operational data at Tencent

	Apache kafka	Tencent's customized kafka
Read and write recovery time after cluster failure	Depends on the developer's time to deal with the failure	The client will stop writing to the abnormal sub-cluster, and the recovery time is <1min
Number of Topics in a single cluster	Before version 2.8.0, due to the performance bottleneck of zookeeper, the number of topic partition could not meet our needs	The number of topic partition in a cluster can support up to 10,000 topics * 10,000 partitions
Machine recycling and partition shrinking after traffic peaks	Machine recycling can be achieved through rebalance, but partition scaling down cannot be supported	We can achieve business-insensitive partition scaling down and recycling machines without balancing

My personal view about Tencent's Kafka

So, this is a quiet stright forward way to achieve the needs of Tencent's own business. Personally, I think the most interesting part of this solution is the way they handle the management of Kafka cluster. As they said, the using a new Kafka cluster to manage the remain parts of Kafka clusters. This is great solution while do not replace zookeeper. The idea behind this behavior is some sort of "scaling". And it is very similar to the lever cache strategy in computer system. When the number of partitions reachs a certain number, it is very struggle for zookeeper to manage these brokers and manage whole Kafka system. However, zookeeper does give us a lot of convenient functions and it works petty well when the size of Kafka is not too big. So, the key step of Tencent's solution is using a modified Kafka cluster which do not have broker on it, manages the metadata. And to release the load of zookeeper, they decide to group all clusters by region. Any cross-region topic partition is not banned.

Also, Tencent's engineers decide to remove the proxy layer. This action makes it easier for Kafka to integrate some new feature of the community. But I am not sure about how they achieve this. For next step, Tencent may continue to contribute on KIP-500 to replace zookeeper and I will do some study about it.

KIP-500, it is the document about the replacement of zookeeper in Kafka and I found some great reasons about why zookeeper cannot fit today's requirement.

> ## Metadata as an Event Log
>
> We often talk about the benefits of managing state as a stream of events. A single number, the offset, describes a consumer's position in the stream. Multiple consumers can quickly catch up to the latest state simply by replaying all the events newer than their current offset. The log establishes a clear ordering between events, and ensures that the consumers always move along a single timeline.
>
> **However, although our users enjoy these benefits, Kafka itself has been left out.** We treat changes to metadata as isolated changes with no relationship to each other. When the controller pushes out state change notifications (such as LeaderAndIsrRequest) to other brokers in the cluster, it is possible for brokers to get some of the changes, but not all. Although the controller retries several

> times, it eventually give up. This can leave brokers in a divergent state.
>
> **Worse still, although ZooKeeper is the store of record, the state in ZooKeeper often doesn't match the state that is held in memory in the controller.** For example, when a partition leader changes its ISR in ZK, the controller will typically not learn about these changes for many seconds. There is no generic way for the controller to follow the ZooKeeper event log. Although the controller can set one-shot watches, the number of watches is limited for performance reasons. When a watch triggers, it doesn't tell the controller the current state-- only that the state has changed. By the time the controller re-reads the znode and sets up a new watch, the state may have changed from what it was when the watch originally fired. If there is no watch set, the controller may not learn about the change at all. In some cases, restarting the controller is the only way to resolve the discrepancy.
>
> Rather than being stored in a separate system, metadata should be stored in Kafka itself. This will avoid all the problems associated with discrepancies between the controller state and the Zookeeper state. Rather than pushing out notifications to brokers, brokers should simply consume metadata events from the event log. This ensures that metadata changes will always arrive in the same order. Brokers will be able to store metadata locally in a file. When they start up, they will only need to read what has changed from the controller, not the full state. This will let us support more partitions with less CPU consumption.

There are some deployment and configuration improvements after removing zookeeper.

> ## Simpler Deployment and Configuration
>
> ZooKeeper is a separate system, with its own configuration file syntax, management tools, and deployment patterns. This means that system administrators need to learn how to manage and deploy two separate distributed systems in order to deploy Kafka. This can be a daunting task for administrators, especially if they are not very familiar with deploying Java services. Unifying the system would greatly improve the "day one" experience of running Kafka, and help broaden its adoption.
>
> Because the Kafka and ZooKeeper configurations are separate, it is easy to make mistakes. For example, administrators may set up SASL on Kafka, and incorrectly think that they have secured all of the data traveling over the network. In fact, it is also necessary to configure security in the separate, external ZooKeeper system in order to do this. Unifying the two systems would give a uniform security configuration model.
>
> Finally, in the future we may want to support a single-node Kafka mode. This would be useful for people who wanted to quickly test out Kafka without starting multiple daemons. Removing the ZooKeeper dependency makes this possible.

Definition

Data warehouse:

> In computing, a **data warehouse** (**DW** or **DWH**), also known as an **enterprise data warehouse** (**EDW**), is a system used for reporting and data analysis and is considered a core component of business intelligence.[1] DWs are central repositories of integrated data from one or more disparate sources. They store current and historical data in one single place[2] that are used for creating analytical reports for workers throughout the enterprise.[3]

Apache Flink:

> **Apache Flink** is an open-source, unified stream-processing and batch-processing framework developed by the Apache Software Foundation. The core of Apache Flink is a distributed streaming data-flow engine written in Java and Scala.[2][3] Flink executes arbitrary dataflow programs in a data-parallel and pipelined (hence task parallel) manner.[4] Flink's pipelined runtime system enables the execution of bulk/batch and stream processing programs.[5][6] Furthermore, Flink's runtime supports the execution of iterative algorithms natively.[7]
>
> Flink provides a high-throughput, low-latency streaming engine[8] as well as support for event-time processing and state management. Flink applications are fault-tolerant in the event of machine failure and support exactly-once semantics.[9] Programs can be written in Java, Scala,[10] Python,[11] and SQL[12] and are automatically compiled and optimized[13] into dataflow programs that are executed in a cluster or cloud environment.[14]
>
> Flink does not provide its own data-storage system, but provides data-source and sink connectors to systems such as Amazon Kinesis, Apache Kafka, HDFS, Apache Cassandra, and ElasticSearch.[15]

Reference

https://en.wikipedia.org/wiki/Data_warehouse

https://en.wikipedia.org/wiki/Apache_Flink

https://cwiki.apache.org/confluence/display/KAFKA/KIP-500%3A+Replace+ZooKeeper+with+a+Self-Managed+Metadata+Quorum#KIP500:ReplaceZooKeeperwithaSelfManagedMetadataQuorum-MetadataasanEventLog

https://kafkasummit.io/session-virtual/?v26dd132ae80017cdaf764437c30ebe6f10c1b1eeaab01165e44366654b368dfaeab6baf7e386a642ecb238989334530e=FFBAA5A18A1382F8FE99CE8FAA83B7B551583065287F0669212F8B00119C37FAA9EE6FB616077BAEAC2EC227B4DF7372&fromFollowed (You need to have an account to access this link. The name of this presentation is "**Enhancing Apache Kafka for Large Scale Real-Time Data Pipeline at Tencent**" and the speaker's pronunciation is really really bad.)