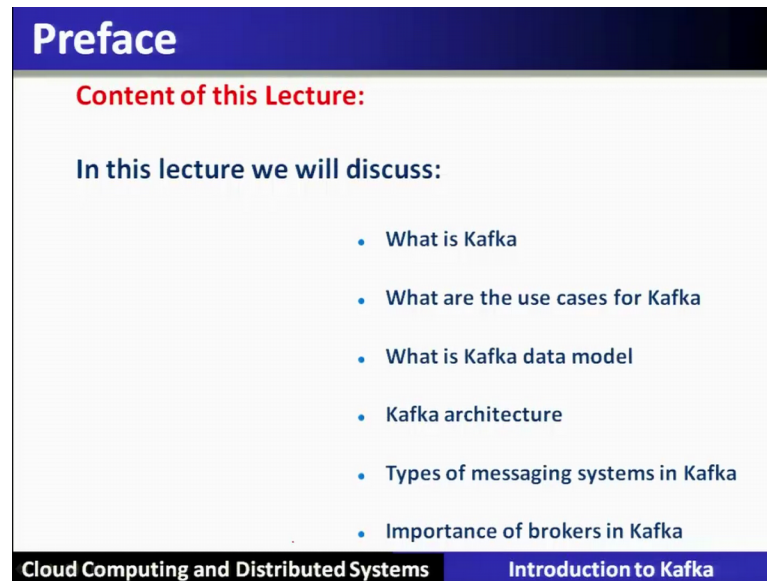


Cloud Computing and Distributed Systems
Dr. Rajiv Misra
Department of Computer Science and Engineering
Indian Institute of Technology, Patna

Lecture – 21
Introduction to Kafka

(Refer Slide Time: 00:15)



Preface

Content of this Lecture:

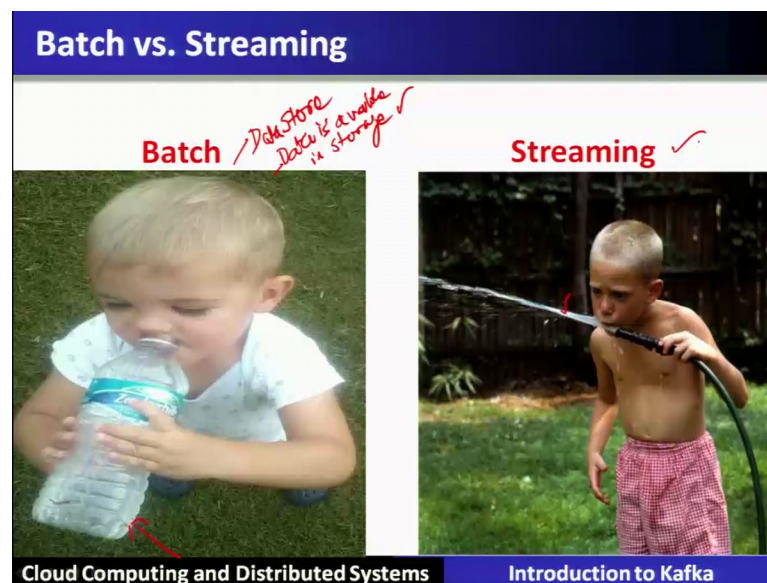
In this lecture we will discuss:

- What is Kafka
- What are the use cases for Kafka
- What is Kafka data model
- Kafka architecture
- Types of messaging systems in Kafka
- Importance of brokers in Kafka

Cloud Computing and Distributed Systems **Introduction to Kafka**

Introduction to Kafka. Preface content of this lecture. We will cover Apache, Kafka, its use cases; the data model of Kafka architecture, the type of messaging systems, importance of brokers in Kafka.

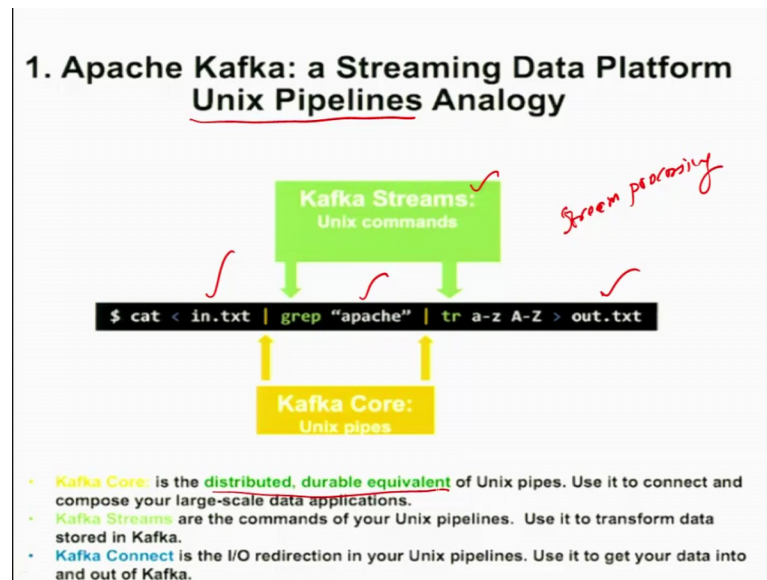
(Refer Slide Time: 00:39)



Let us understand what Kafka is used for so far. We have seen the databases. So, we can turn this particular data is stored as a batch processing application. Why because, data is available in the storage. Hence it is just like the water which is stored in a bottle and then it will be processed the data.

On the other hand, if the data is flowing and you want to tap the water out of that stream, then it is called a streaming. So, how are we processing the streaming kind of data? So, streaming in the sense data is flowing for example, in the network the data is flowing one; that means, in the internet lot of data is flowing all the time. In the real time if we analyze capture the data that is called time series data, and to process that data and store in the database or update in the database is some of the applications of streaming applications. So streaming is becoming a very powerful computing paradigm.

(Refer Slide Time: 02:20)



So, streaming is not new. So, streaming data platform was already there in the Unix pipeline for example, when we say out of a file data when we say grep Apache; that means, out of that particular file stream it will capture all those text. That means, which contains this particular word Apache, and it will be output. So, this is a kind of stream which was provided in the Unix command, but you know that Unix servers were running on one machine as a process.

So, it was a limited in the applications in the current scenario, in the current scenario the data is huge. So, this kind of streaming is required to be redefined, and Apache Kafka exactly does that that is called Kafka stream or a stream processing.

So, Kafka is a distributed equivalent of Unix pipes, which is used to connect and compose the large scale data applications that we will see in this part of the discussion.

(Refer Slide Time: 03:53)

Introduction: Apache Kafka

- **Kafka is a high-performance, real-time ^{Distributed} messaging system. It is an open source tool and is a part of Apache projects.**
- The characteristics of Kafka are:
 1. It is a distributed and partitioned messaging system. ✓
 2. It is highly fault-tolerant ✓
 3. It is highly scalable. ✓
 4. It can process and send millions of messages per second to several receivers.

Cloud Computing and Distributed Systems Introduction to Kafka

Apache Kafka introduction; so, Kafka is the high performance real time messaging system. In fact, it is a distributed messaging system. So, this particular Kafka is an open source and a part of Apache projects. The characteristics of the Kafka are it is a distributed partition messaging system. We will understand these terms in more details.

It is fault tolerant, highly scalable. It can process and send million of messages per second to several receivers. So, that scale this particular platform is going to support.

(Refer Slide Time: 04:49)

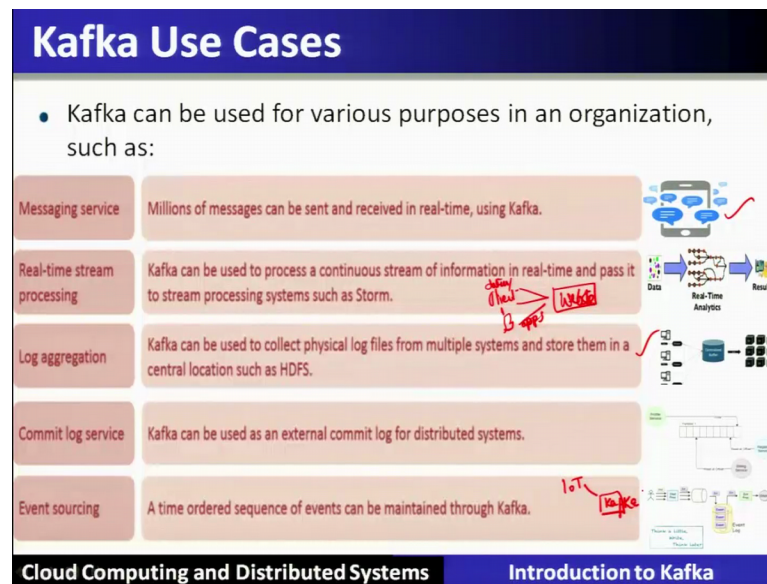
Kafka History

- **Apache Kafka was originally developed by LinkedIn and later, handed over to the open source community in early 2011.**
- It became a main Apache project in October, 2012. ✓
- A stable Apache Kafka version 0.8.2.0 was release in Feb, 2015.
- A stable Apache Kafka version 0.8.2.1 was released in May, 2015, which is the latest version.

Cloud Computing and Distributed Systems Introduction to Kafka

Let us trace through the brief history of Apache Kafka. So, Apache Kafka was originally developed by LinkedIn. And later handed over to the open source community in 2011. It became Apache project in October 2012. The stable version that is, 0.8.2.0 was released in February 2015. Stable version that is 0.8.2.0 version was released in May 2015. And this becomes continuous in the updates.

(Refer Slide Time: 05:40)



So, Kafka use cases, Kafka can be used for various purposes in any organizations such as it is messaging system billions of messages can be sent and received in the real time using Kafka. Messaging system we have already used in our mobile phones. Similarly, it is also used for real time stream processing. For example, your data, your website is being accessed through different means. For example, mobile apps can also use it through internet client can also use it. And also there are many other route from database it can also access.

So, there are many ways your website is accessing and you want to keep track of them in the real time. Hence, this will become a real time stream processing application. And Kafka can be used to process a continuous stream of information in the real time and pass it to the stream processing system such as storm. Another application is called log aggregation which often is used in any organization. So, Kafka can be used to collect the physical log files from multiple systems, and store them in a central location such as HDFS.

So, for example, there are different proxy servers. And these proxy servers are generating the physical log files. So, from multiple such systems a log can be physically collected. And is being aggregated and stored in HDFS in a real time. Similarly, commit log service. Kafka can be used as an external commit log for distributed system. Event sourcing that is time ordered sequence of events can be maintained through the Kafka that we will see the event sourcing. One such example is an IoT based events which are happening that can be captured using Kafka. And they can be tapped for maintaining the sequence of events into HDFS.

(Refer Slide Time: 08:34)

Apache Kafka: a Streaming Data Platform

Most of **what a business does can be thought as event streams**. They are in a

- **Retail system**: orders, shipments, returns, ...
- **Financial system**: stock ticks, orders, ...
- **Web site**: page views, clicks, searches, ...
- **IoT**: sensor readings, ...

and so on.

Cloud Computing and Distributed Systems Introduction to Kafka


So, most of the businesses can be thought of as event streams processing. For example, the businesses means it is handling the retail system that is various orders shipments returns financial systems, such as stock ticks orders, website uses such as page views clicks searches and monitoring through IoT that is called sensor reading and so on. This all kind of events are happening as the event stream in any business organization. To capture all of it, and get an aggregate view for any organization or for any customer or a user requires this kind of processing called streaming data application.

So, Kafka is the tool for doing this in an present scenario, the businesses are using this kind of event stream processing and Kafka will handle it.

(Refer Slide Time: 10:02)

Enter Kafka

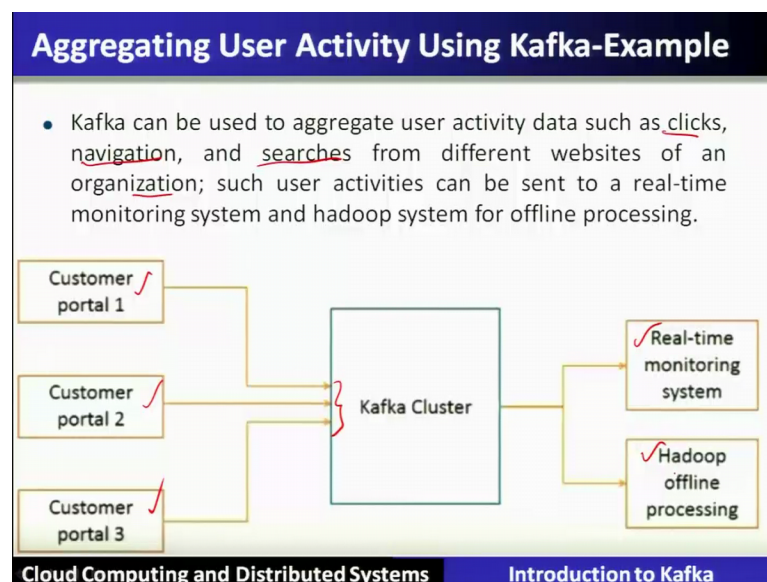
- Adopted at 1000s of companies worldwide



Cloud Computing and Distributed Systems Introduction to Kafka

Therefore, the Kafka is adopted more than thousands of companies worldwide some of the companies which are listed here are already using the Kafka as stream processing Netflix, EBay, airbnb then PayPal, Yahoo, Wikipedia, Salesforce and so on.

(Refer Slide Time: 10:28)

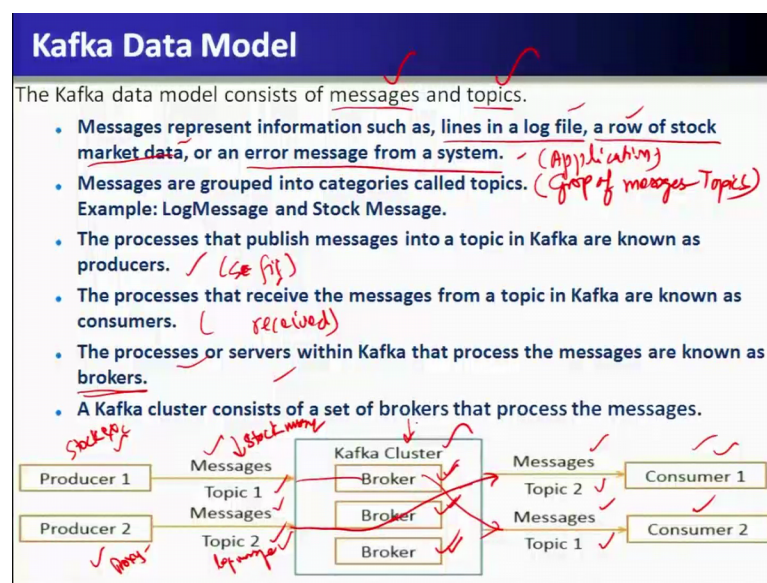


So, let us see how this aggregation of a particular user activity can be done in Kafka. Let us go through this simple example. So, Kafka can be used to aggregate the user activity data such as clicks navigation, searches from different websites of an organization. Such user activities can be sent to the real time monitoring system, and Hadoop for online

offline processing. So, here let us see that we have a customer portal 1, 2 and 3 who is accessing the upfront the websites and doing navigation having a clicks and search the websites.

All this particular events or activity data will be captured and will be sent to the Kafka Cluster. Kafka Cluster will then process these different streams and capture different type of topics for the real time monitoring, and also out of that some data will be stored in the databases through the Hadoop offline processing.

(Refer Slide Time: 12:06)



So, to do this let us go in more detail and see, what are the data models which are supported in the Kafka. So, the Kafka data model consists of the messages and the topics. So, messages represent the information's such as lines in a log file; a row in a row of a stock market data or an error message which is generated out of that particular system. So, these messages are generating or this is being generated out of different applications. And hence this different kind of information which we are seeing over here is dependent upon different applications.

But in Kafka it is all called as the messages as an abstraction. These messages are grouped into the categories which are called the topics. So, messages and topics we have just seen, the example is that the log message and the stock message is the example of kind of topics we will see in more detail about this. The processes that publish the messages into the topic in Kafka is known as the producer. Here in the figure below. You

can see there are 2 producers. They will produce the messages and also will be classified under the topic 1, another message which is produced by another producer which is classified as the topic number 2.

(Refer Slide Time: 14:37)

Topics

- **A topic** is a category of messages in Kafka. ✓
- **The producers** publish the messages into topics. ✓
- **The consumers** read the messages from topics. ✓
- **A topic** is divided into one or more partitions. ✓ (high throughput)
- **A partition** is also known as a commit log.
- **Each partition** contains an ordered set of messages.
- **Each message** is identified by its offset in the partition.
- **Messages** are added at one end of the partition and consumed at the other.

The diagram illustrates the Kafka architecture. It shows a 'Topic: simple' which is divided into two partitions: 'Partition 0' and 'Partition 1'. Each partition contains a sequence of messages identified by their offset (1, 2, 3, 4, 5, 6). A 'Writes' arrow points to the start of the message sequence in Partition 0. A 'Reads' arrow points away from the end of the message sequence in Partition 1. Handwritten red annotations include: 'copy' with an arrow from Partition 0 to Partition 1; 'offset' with arrows pointing to the message numbers; 'writer' and 'reader' with arrows pointing to the 'Writes' and 'Reads' respectively; and 'Bounded buffer problem - operating system' with an arrow pointing to the message sequence. The bottom of the slide has a blue bar with the text 'Cloud Computing and Distributed Systems' and 'Introduction to Kafka'.

So, the producer could be from by stock exchange, then the message will be the stock messages, and if this is the proxy server of some server or some data center, then here it will generate the log message. So, the processes that receive the message from the topic in the Kafka are known as the consumer. So, these messages will be received or also called as a consumed, they are called consumers who will consume it. So, these terms are used here in Kafka description. The processes or the servers within the Kafka that processes the message are known as Brokers. Here we are seeing here 3 different processes which process these messages and give it to the different customers who are interested in such topics.

For example, topic number 2 is of interest to the customer one. So, it will be sent to the customer number one after the processing through the Broker. So, the Broker will take the message from the producer of a particular topic process it and gives it to the consumer. Similarly, the message one will be handled by let us say a particular Broker and give it to the consumer number 2 who is interested in the message one. So, the processes or the servers within the Kafka that process the messages are known as the

Brokers. So, these Brokers are the processes and they run on the Cluster which is called a Kafka Cluster. So, Kafka Cluster consists of a set of Brokers that process the message.

Let us go in more detail of this data type which is called a topic. So, topic is a category of message in a Kafka. So, the producers publish the messages into the topics and consumer read the message of a particular topic in which he is of interest. So, topic is divided into one or more partitions, why for high throughput. Now partition is also known as commit log each partition contains an ordered set of messages. Each message is identified by its offset in the partition. Messages are added at one end of a partition and consumed at the other end of a partition.

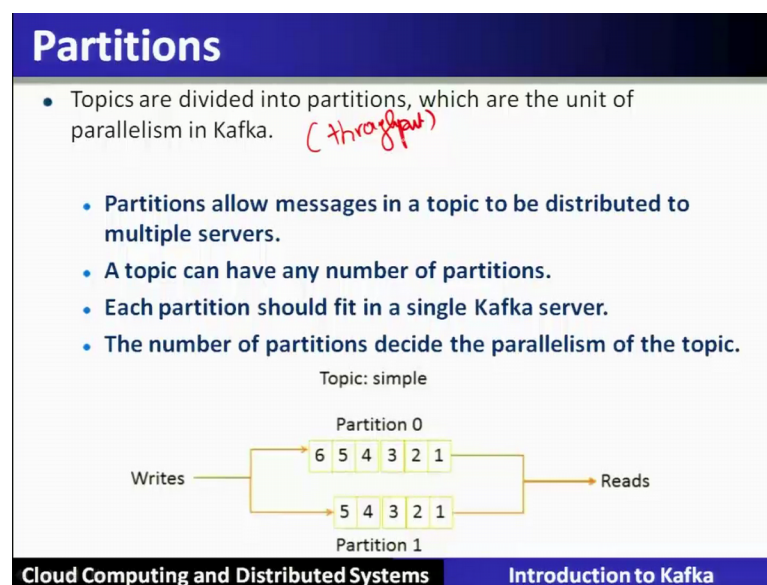
Let us understand this through a simple example. Before going into this example let us understand the concept of a reader and writer problem in a bounded buffer problem. That is called a bounded buffer problem. So, this particular problem you might have studied in operating system courses which are now generalized here in this particular form of messaging system. So, the writer will keep on writing in this particular form of a queue and reader will read in this particular queue and all the messages will be strictly in that particular order in which the writer writes.

Now, if writer is writing and reader is not reading, then eventually this particular bounded buffer will be full after sometime. And once the reader starts reading it, then it will be read. These particular elements are called offset. Now this particular single buffer is now generalized here in the Kafka messaging system which is maintained in the distributed system not in a single server. We will see this particular generalization and understand all the topics. What you mean by the topics?

So, a topic is now partitioned. So, messages are categorized into the partition which is holding the messages of a particular topic. So, this particular let us say that this is. So, this particular topic is divided into 2 different partitions, and the writer can write down for a particular message and according to the topic it will be classified in one of these particular partitions it will be written at the end. Similarly, the consumer will consume from the other end of the partition. So, each partition contains an ordered set of messages, ordered in the sense as they arrive ordering is defined here in the partition and it is strictly ordered.

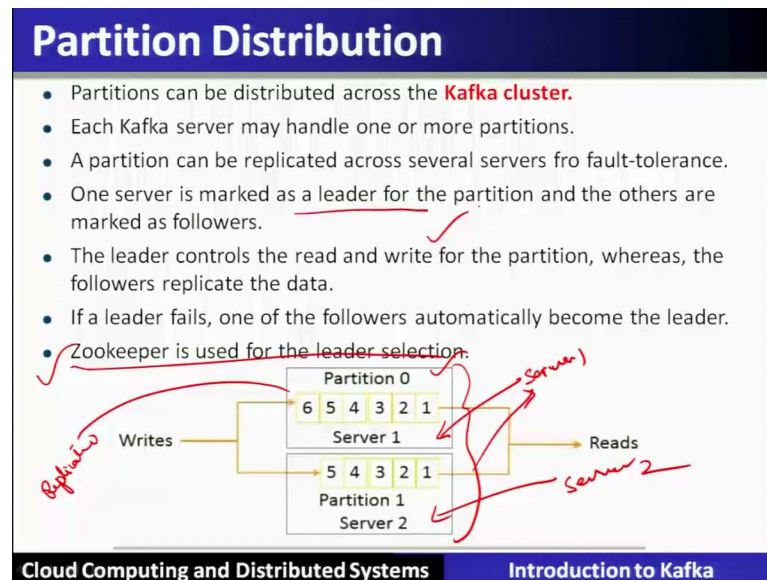
So, each message in the partition will be identified by an offset. Here the offset is offset number 4 of partition 0. And here this is the offset number 5 of the partition 1. So, if a new message will come to a partition 0. So, it will be finally, consumed here, and it will go to the offset 4 after the consumer consumes 2 more messages. So, this particular offset will know at which point of time which messages are available in to the partition. Again the messages are added at one end; that means, the writer will write down at this end, and the reader will read from the other end that is shown here in the picture; that is very simple let us go and understand about the partition.

(Refer Slide Time: 21:56)



So, the topics are divided into the partition which is the unit of parallelism in Kafka. So, parallelism is for enhancing the throughput by exploiting the parallelism. So, partition allows the messages in a particular topic to be distributed to the multiple servers. So, a topic can have any number of partitions, and each partition should fit in a single Kafka server. So, if the size increases, then you can have another partition and can be stored in another server. So, the number of servers can be scaled in a horizontal scaling to support many partitions here in Kafka. So, the number of partition decides the parallelism of the topic that we have already explained the partition distribution.

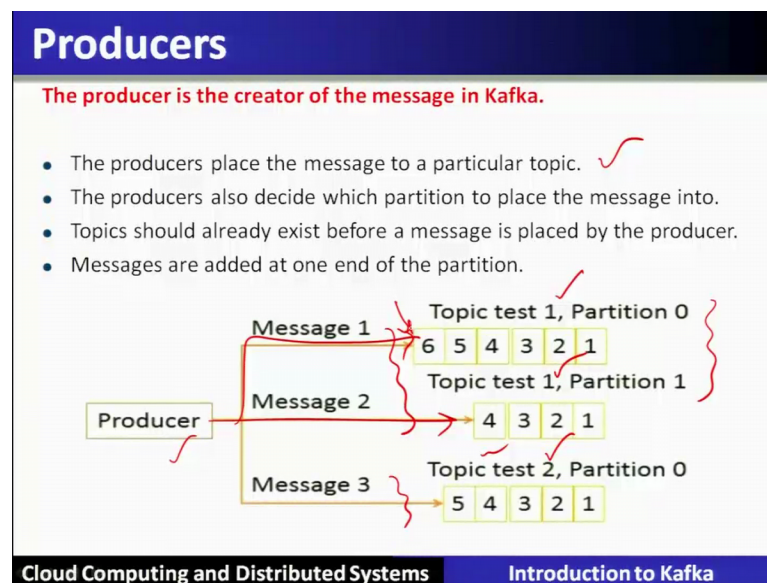
(Refer Slide Time: 23:02)



So, partitions can be distributed across the Kafka servers. Each server may handle one or more partition here we can see in the example that partition number 0 is stored in the server 1 and it has another server 2; which stores a different partition which is called partition 1. It is also possible that 2 different partitions can be stored in one server, or they can be separately stored. So, a partition can be replicated across several servers to for the fault tolerance. So, these petitions are replicated also if Hadoop is used then replication is by default 3.

If it is having more than one replicas, then one server is marked as the leader for the partition and others are the follower. And this leader election will be done through the Zookeeper. This particular leader controls the read and write for the partition where as the replicas will replicate the data. Hence a strict consistency is maintained why because the leader will control the rights also. So, if the leader fails one of the follower can automatically becomes the leader through the leader election.

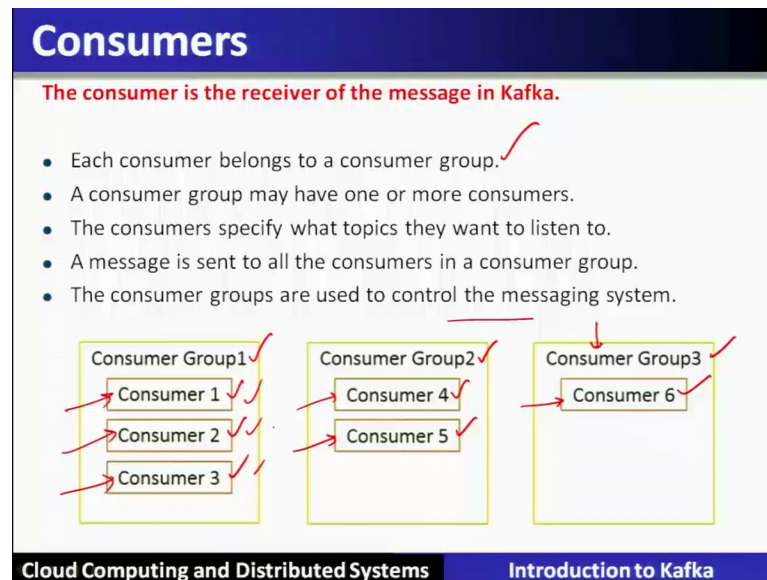
(Refer Slide Time: 25:03)



Let us understand about the producers. So, producer is the creator of the message in the Kafka. So, producer places the message to a particular topic. So, here we have shown the example of 3 different topic, topic which is called test one and another topic which is called test 2. So, here the producer produces 3 different types of messages, which are classified as the topic test one and another message which is classified as a topic test 2. So, therefore, the producers place the messages to a particular topic the producers also decide which partition to place the message into it.

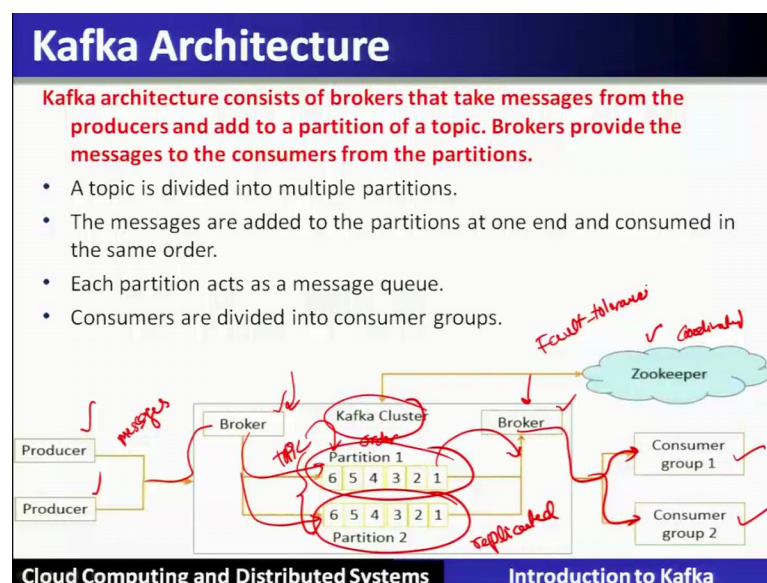
For example, message one and message 2 belongs to topic test one, but topic test one is maintained in 2 different partitions. So, the producer has to decide that this message will be placed in the partition 1. And message one will be maintained in a partition 0. Topic should already exist before the message is placed by the producer. So, messages are added at one end of the partition that we have already seen. So, the producer will write into the partition at one end of the partition.

(Refer Slide Time: 26:52)



Now, coming to the consumer, consumer is the receiver of the message in the Kafka. So, each consumer belongs to the consumer group. Here in this example we have shown 3 different consumer groups in the boxes. So, the consumer group may have one or more consumers. For example, consumer 1 has 3 different consumers. Consumer 2 has 2 consumers in his group. And consumer 3 has one consumer in the group the consumers specify what topic they want to listen to. So, the message is sent to all the consumers in a particular consumer group. The consumer group is used to control the messaging system. Let us see through more examples.

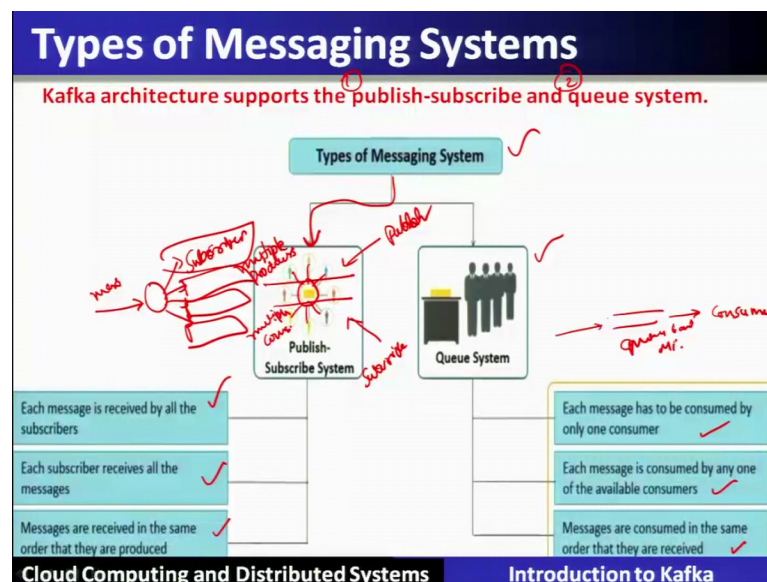
(Refer Slide Time: 27:47)



So, when a message comes they will be all the messages will be given to the consumer, but these consumers they also have to tell which kind of messages which topic, they are interested in hence once the consumer receives all the messages. It will control to give the topics which are of their interest to them. So, let us understand the Kafka architecture. So, Kafka architecture consists of the Brokers, these are the Brokers. That takes the message from the producer and adds to the partition of a topic. Brokers provide the messages to the consumer from the partition.

A topic is divided into multiple partitions. The messages are added to the partition at one end and consumed from the other end. In the same order each partition acts as a message queue, consumers are divided into the consumer groups.

(Refer Slide Time: 29:29)



So, let us understand the other component in the architecture. So, we have seen that the producer is producing the messages which through the Broker will be stored in these partitions. The Broker is the processes which are running in the Cluster which is called a Kafka Cluster.

Again these partitions also will be running either on the same server or multiple server and they are replicated also. Therefore, several machines are required to run this Kafka Cluster. And this all will be coordinated by the Zookeeper. So, therefore, Zookeeper will ensure the fault tolerance in the Kafka Cluster. The Kafka architecture supports 2 types

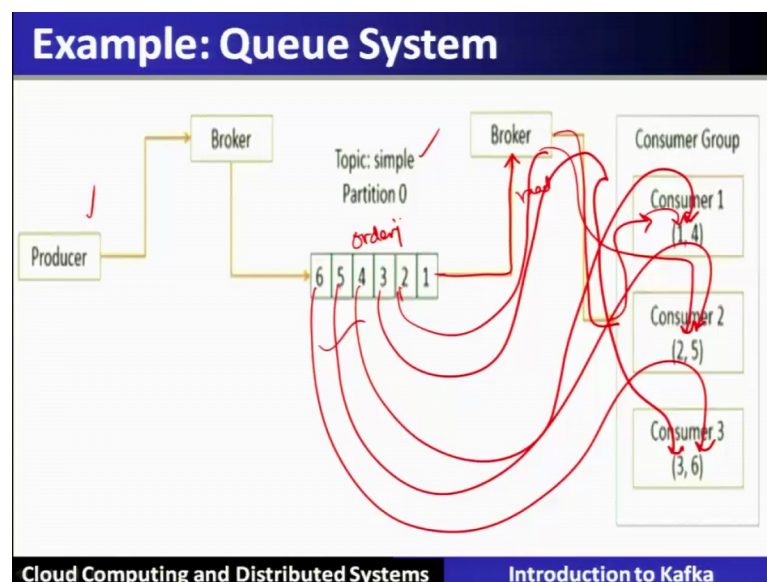
of messaging system. The first one is called publish subscribe messaging system. The other is called queue based messaging system.

So, there are 2 types the first type which is called publish subscribe messaging system. In this particular model, there is a producer which will produce there are several producers, which will be producing the messages. And these messages will be consumed by many consumers; so, multiple producer and multiple consumers. So, they are called publishers and the other system is called subscribe publish subscribe messaging system.

So, here each message in this system is received by all the subscribers. And each subscriber receives all the messages. And messages are received in that particular order that they are being produced. So, therefore, different subscriber will receive all the messages and they will do their applications as per the requirement.

On the other hand, there is another system which is called a queue system. Here the messages each message will be consumed by only one customer or by one consumer. So, each message is consumed by any one of the available consumers. And messages are consumed in the order they are received. So, that becomes a queue based messaging system. Let us see these systems in more details.

(Refer Slide Time: 33:42)



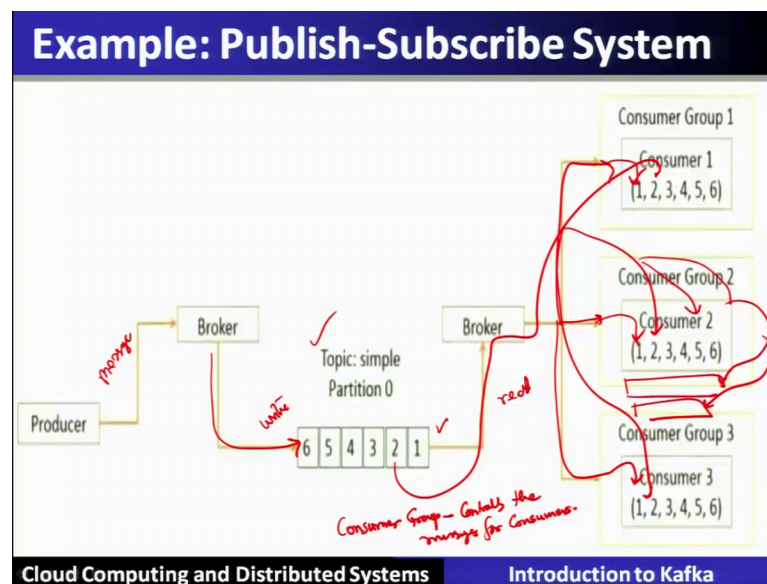
So, here the producer is producing the messages, and using the Broker the message will join in that particular partition of a particular topic let us say some topic. This particular

partition using Broker will read will consume the message that will it will read it, from the other end of the queue and it will give to exactly one consumer. So, here it is given to consumer number 1. It will not be given to all other consumer in a queue based messaging system. Similarly, the message number 2, through the Broker will be read and it will be given to the consumer number 2.

The message number 3 will be read through the Broker and it will be given to the consumer number 3 in this particular sequence. The message number 4 will be given to the consumer 1, message number 5 will be given to the consumer 2 and message number 6 will be given to the consumer number 3.

So, the ordering is strictly maintained and this is the method which is used here in the queue system.

(Refer Slide Time: 35:18)



On the other hand, in publish subscribe system let us see through an example. The producer is producing the messages which through the Broker will be written to the partition of a particular topic at one end of the partition. The message will be read from the other end of a partition, and through the Broker it will be given to all the consumers.

So, here you see that all the consumer will have the message number 1. Similarly, the message 2 will go through the through the Broker and will reach to all the consumers. Now if a particular consumer group has more than one consumer so, on arriving all these

messages at the consumer group this consumer group will decide which message to go to which of these consumers. So, consumer group here controls the messages for the consumers. In case every consumer group has only one consumer. So, it will be given all the messages to that consumer if more than one consumer is there in a particular group. So, it will be controlled through the consumer group and the messages will be delivered between them.

(Refer Slide Time: 37:17)

Brokers

Brokers are the Kafka processes that process the messages in Kafka.

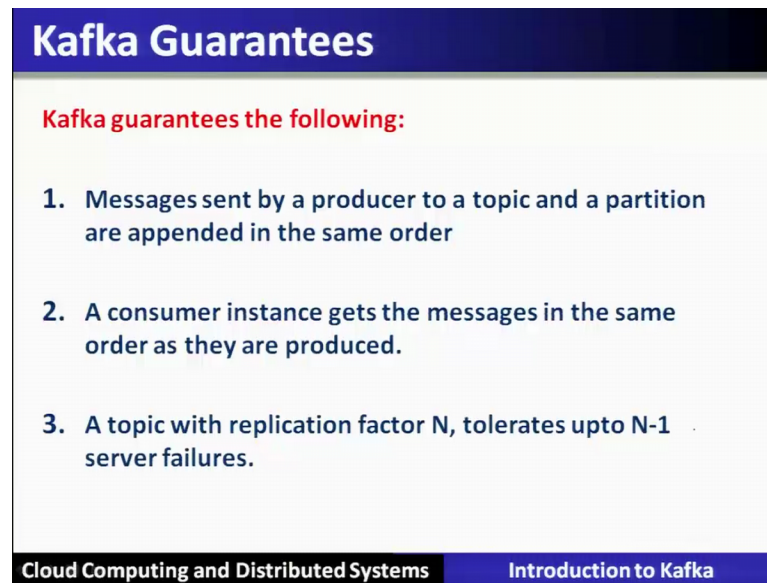
- Each machine in the cluster can run one broker.
- They coordinate among each other using Zookeeper.
- One broker acts as a leader for a partition and handles the delivery and persistence, where as, the others act as followers.

Cloud Computing and Distributed Systems Introduction to Kafka

The slide features a blue header with the title 'Brokers'. The main content area is white with a red definition and a bulleted list. A red checkmark is visible next to the last bullet point. The footer consists of two blue boxes with white text: 'Cloud Computing and Distributed Systems' and 'Introduction to Kafka'.

Now, comes the Broker. So, Brokers are the Kafka processes that process the messages in the Kafka messaging system. Each machine in the Cluster can run one Broker and they coordinate among each other using the Zookeeper. One Broker will act as the leader for the partition and handles the delivery and persistence; whereas the others act as the followers. Therefore, these Brokers are replicated and one of them is the leader which controls all the read and write to the partitions.

(Refer Slide Time: 38:13)

A presentation slide titled "Kafka Guarantees" with a dark blue header. The main content area is light gray and contains a red heading "Kafka guarantees the following:" followed by a numbered list of three points. The footer consists of two dark blue bars: "Cloud Computing and Distributed Systems" on the left and "Introduction to Kafka" on the right.

Kafka Guarantees

Kafka guarantees the following:

1. Messages sent by a producer to a topic and a partition are appended in the same order
2. A consumer instance gets the messages in the same order as they are produced.
3. A topic with replication factor N, tolerates upto N-1 server failures.

Cloud Computing and Distributed Systems Introduction to Kafka

Therefore, Kafka guarantees the following. First one is that the messages sent by the producer to a particular topic. And the partitions are appended in that particular order in which the messages are being sent by the producer. So, the order is maintained at any point of time, and that is guaranteed by the Kafka. Second guarantee which Kafka ensures is that the consumer instance gets the message in the same order they are being produced. So, a strictly order is maintained in the Kafka. So, a topic with the replication factor n tolerates up to n minus 1 different failure.

Now, let us see the replication in the Kafka.

(Refer Slide Time: 39:09)

Replication in Kafka

Kafka uses the primary-backup method of replication.

- One machine (one replica) is called a leader and is chosen as the primary; the remaining machines (replicas) are chosen as the followers and act as backups.
- The leader propagates the writes to the followers. ✓ *strict consistency*
- The leader waits until the writes are completed on all the replicas.
- If a replica is down, it is skipped for the write until it comes back. ✓
- If the leader fails, one of the followers will be chosen as the new leader; this mechanism can tolerate $n-1$ failures if the replication factor is 'n'. ✓

Cloud Computing and Distributed Systems Introduction to Kafka

So, Kafka uses primary backup method of replication; that means, one machine one replica is called the leader and is chosen as the primary the remaining machines are chosen as followers and act as a backup. The leader propagates the writes to the followers therefore, it ensures the strict consistency. The leader waits until the writes are completed on all the replicas if the replica is down it is skipped for write until it comes back. If the leader fails one of the followers will be chosen as the new leader and this mechanism can tolerate up to n minus 1 failure if the replication factor is n .

(Refer Slide Time: 40:00)

Persistence in Kafka

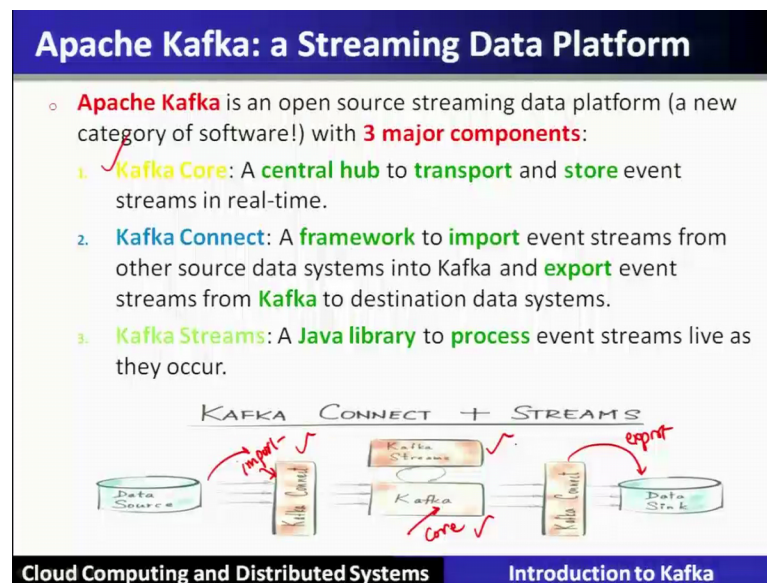
Kafka uses the Linux file system for persistence of messages

- Persistence ensures no messages are lost.
- Kafka relies on the file system page cache for fast reads and writes.
- All the data is immediately written to a file in file system.
- Messages are grouped as message sets for more efficient writes.
- Message sets can be compressed to reduce network bandwidth.
- A standardized binary message format is used among producers, brokers, and consumers to minimize data modification.

Cloud Computing and Distributed Systems Introduction to Kafka

Kafka uses the Linux file system for persistence of the messages. So, persistence ensure that no messages are lost Kafka relies on the file system page cache for fast read and write all the data are immediately written to a file system. Messages are grouped as message set for more efficient writes. And messages set can be imposed to reduce the network bandwidth. A standardized binary message format is used among the producers Brokers and consumers to minimize the data modifications.

(Refer Slide Time: 40:38)



In nutshell, Apache Kafka is a streaming data platform; which is an open source streaming data platform with 3 different major components. The first one is called Kafka core. This is the central hub to transport and store the event streams in a real time. That we have already seen.

Then second component is called Kafka connect. This framework is to import the event streams from other sources into the Kafka and export the events from the Kafka to the destination. So, there are 2 different connect. One is for import through the data sources. The other Kafka connect will export to the data sink. Then finally, we have seen the data streams that are called Kafka streams. So, Kafka streams are the java library to process the event streams live as they occur. So, that framework we have discussed here in this lecture.

(Refer Slide Time: 42:02)

Further Learning

- **Kafka Streams code examples**
 - Apache Kafka <https://github.com/apache/kafka/tree/trunk/streams/examples/src/main/java/org/apache/kafka/streams/examples>
 - Confluent <https://github.com/confluentinc/examples/tree/master/kafka-streams>
- **Source Code** <https://github.com/apache/kafka/tree/trunk/streams>
- **Kafka Streams Java docs** <http://docs.confluent.io/current/streams/javadocs/index.html>
- **First book on Kafka Streams (MEAP)**
 - Kafka Streams in Action <https://www.manning.com/books/kafka-streams-in-action>
- **Kafka Streams download**
 - Apache Kafka <https://kafka.apache.org/downloads>
 - Confluent Platform <http://www.confluent.io/download>

Cloud Computing and Distributed Systems Introduction to Kafka

Kafka streams code examples are given in these links and also different resources for further study is mentioned.

(Refer Slide Time: 42:19)

Conclusion

- Kafka is a high-performance, real-time messaging system.
- Kafka can be used as an external commit log for distributed systems.
- Kafka data model consists of messages and topics.
- Kafka architecture consists of brokers that take messages from the producers and add to a partition of a topics.
- Kafka architecture supports two types of messaging system called publish-subscribe and queue system.
- Brokers are the Kafka processes that process the messages in Kafka.

Cloud Computing and Distributed Systems Introduction to Kafka

Conclusion; so, Kafka is high performance real time distributed messaging system. This is highly reliable in the sense that it involves the Zookeeper to maintain the fault tolerance in the system. And also to ensure the horizontal scaling that means, it is highly scalable. We have seen that there are many applications in today's world where a company requires to understand the activity pattern of a particular user; that means, if he

is accessing different websites, what he is doing through the mouse click activities all these are captured through different data sources aggregated it.

And then it will do the real time analysis or it will store in the backup. So, this all is done through the Kafka messaging system. This is a distributed messaging system.

So, Kafka can be used as an external commit log for distributed systems also, this is another application. So, Kafka data model therefore, provides the abstraction of messages and topics. Kafka architecture consists of the Brokers they are nothing but the processors or processes which runs on Kafka Cluster. These processes will read the messages from the producers and maintained it into the partitions which also run on different servers in the Cluster. Kafka architecture also supports the consumers which can take or which can be modeled in 2 different manners one is in the form of queue; that means, exactly one of them can receive the message. The other is called publish subscribe model where in all the consumers will receive in the form of consumer groups, which in turn will divide among themselves according to the topic of interest.

So, publish subscribe model is more general where in different subscribe or a mess or a consumer groups, can be running different kind of applications for example, one such application could be to analyze this real time traffic and then extract the events and stored in the database, maybe (Refer Time: 45:39) database. The other one is it will analyze in the real time and take the action. There may be many different uses of this particular model of consumers.

So, Kafka architecture supports 2 types of messaging system that we have covered publish subscribe and queuing queue system. The Brokers are Kafka processors that process the messages in the Kafka, and they will run on the Cluster which is coordinated through the Zookeeper.

Thank you.