BITS Pilani

Pilani Campus

Apache Kafka

Pravin Y Pawar
CSIS-WILP

Agenda

- ❑ Messaging Systems
- ❑ Kafka Architecture
- ❑ Kafka Fundamentals
- ❑ Kafka Workflow

Messaging System

- ❑ System to transfer data from one application to another
- ❑ Application focuses on its core features whereas messaging system takes care of data transfer / sharing

- ❑ Two types
  - ❑ Point to point
  - ❑ Pub-sub

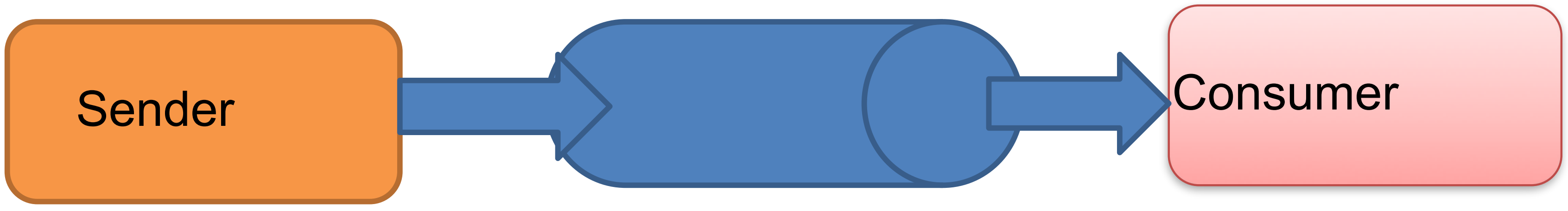
Point to Point messaging system

Messages are persisted in queue

One or more consumers can be connected to queue

But only one can consume the message

Message removed from queue,once any consumer reads it



Message Queue

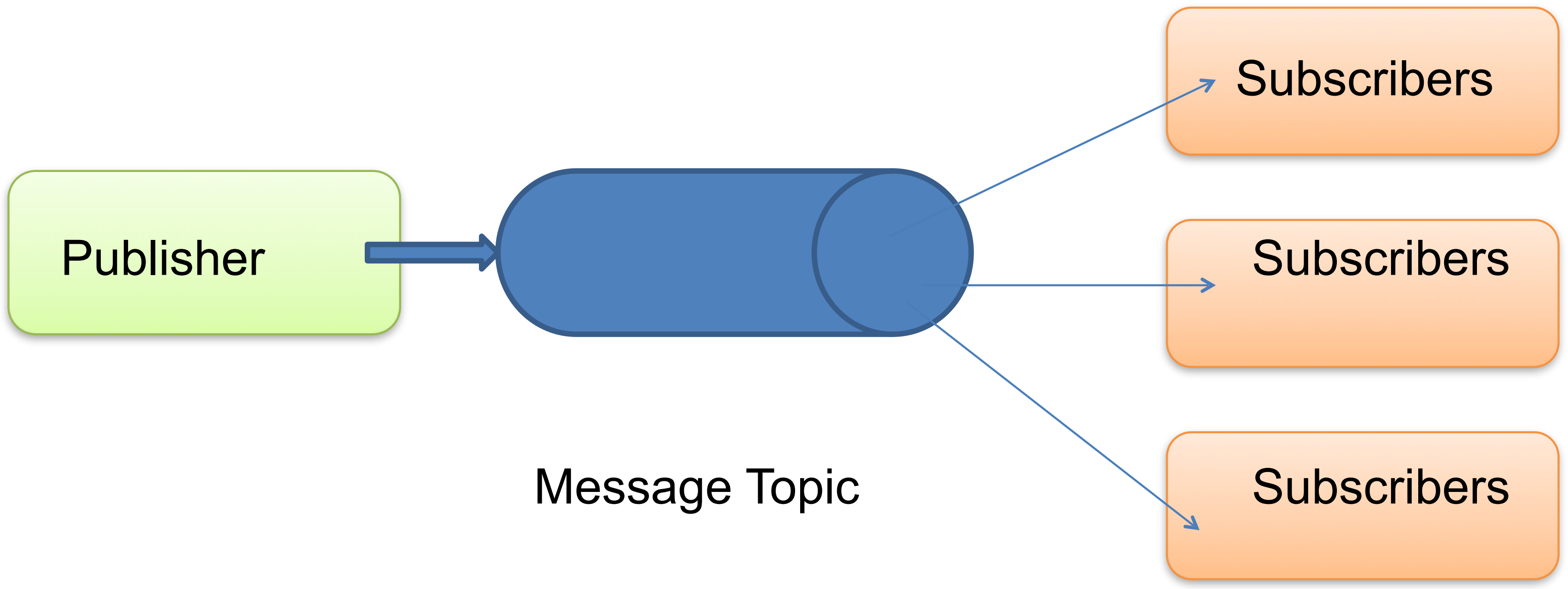
Pub-sub Messaging System

Messages are persisted in topic

Many consumers can consume messages at same time

Message producers are Publishers

Message consumers are Subscribers



Apache Kafka

- Distributed pub-sub and queuing messaging system
- Designed for high throughput distributed systems
  - Better throughput
  - In-built partitioning mechanism
  - Replicas
  - Fault Tolerant

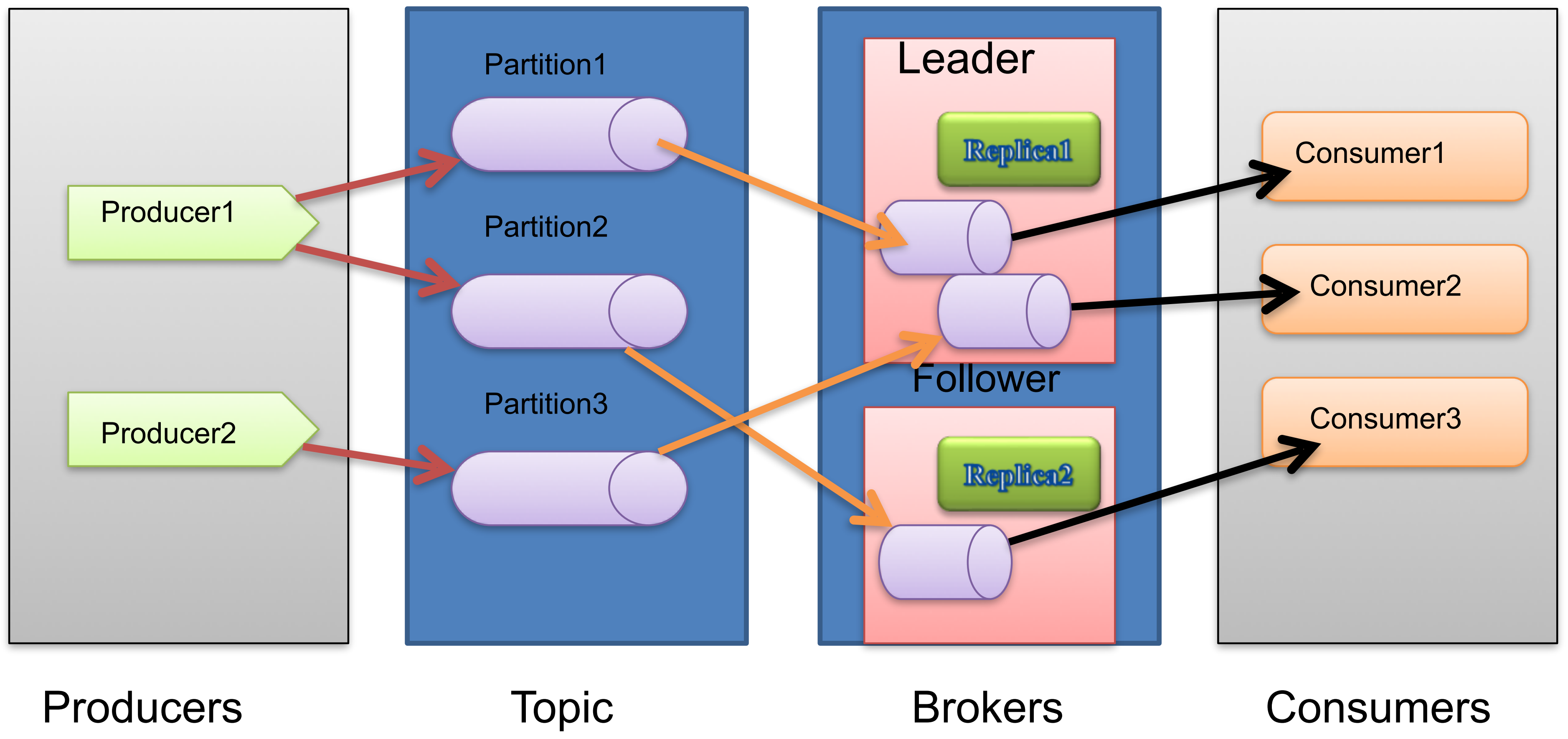
- Good fit for large scale data processing
- Suitable for both offline and online message transfer
- Messages are persisted on disk and replicated across cluster to prevent message loss
- Integrate with Storm and Spark easily

Kafka Benefits

- Reliability
  - distributed, partitioned, replicated and fault tolerant
- Scalability
  - scales easily without down time by addition of machines in cluster
- Durability
  - messages persists on disk as fast as possible
- Performance
  - has high throughput for both publishing and subscribing messages

Kafka Fundamentals

Kafka Architecture



Topics

- A stream of messages belonging to a particular category
  - Data is stored in topic
- Topics are split into partitions
  - For each topic, Kafka keeps a minimum of one partition
  - Each such partition contains messages in an immutable ordered sequence
- A partition is implemented as a set of segment files of equal sizes

Partition

- Topics may have many partitions, so it can handle an arbitrary amount of data.
  - Each partitioned message has a unique sequence id called as "offset".
- Replicas are nothing but "backups" of a partition.
- Replicas are never read or write data. They are used to prevent data loss.

Brokers

- ❑ Simple system responsible for maintaining the published data
  - ❑ Each broker may have zero or more partitions per topic
  - ❑ Used to maintain load balance
  - ❑ are stateless, so they use ZooKeeper for maintaining their cluster state

- ❑ If there are N partitions in a topic and N number of brokers
  - ✓ each broker will have one partition
- ❑ If there are N partitions in a topic and more than N brokers
  - ✓ the first N broker will have one partition and the next M broker will not have any partition for that particular topic
- ❑ If there are N partitions in a topic and less than N brokers
  - ✓ each broker will have one or more partition sharing among them

Producers

- The publisher of messages to one or more Kafka topics
- Sends data to Kafka brokers
- Every time a producer publishes a message to a broker, the broker simply appends the message to the last segment file
  - Actually, the message will be appended to a partition
- Producer can also send messages to a partition of their choice
- Kafka producer doesn’t wait for acknowledgements from the broker and sends messages as fast as the broker can handle.

Consumers

- Consumers read data from brokers.
- Consumers subscribes to one or more topics and consume published messages by pulling data from the brokers.
- Since Kafka brokers are stateless, which means that the consumer has to maintain how many messages have been consumed by using partition offset.
  - If the consumer acknowledges a particular message offset, it implies that the consumer has consumed all prior messages.
  - The consumers can rewind or skip to any point in a partition simply by supplying an offset value.
  - Consumer offset value is notified by ZooKeeper.

Kafka Cluster

- Kafka’s having more than one broker are called as Kafka cluster.
- A Kafka cluster can be expanded without downtime.
- These clusters are used to manage the persistence and replication of message data.

Leader and Follower node

- ❑ Leader
  - ✓ "Leader" is the node responsible for all reads and writes for the given partition.
  - ✓ Every partition has one server acting as a leader.

- ❑ Follower
  - ✓ Node which follows leader instructions are called as follower.
  - ✓ If the leader fails, one of the follower will automatically become the new leader.
  - ✓ A follower acts as normal consumer, pulls messages and updates its own data store.

ZooKeeper

- ZooKeeper is used for managing and coordinating Kafka broker.
- ZooKeeper service is mainly used to notify producer and consumer about the presence of any new broker in the Kafka system or failure of the broker in the Kafka system.
  - ✓ As per the notification received by the Zookeeper regarding presence or failure of the broker then producer and consumer takes decision and starts coordinating their task with some other broker.

Kafka Workflow

- Kafka is simply a collection of topics split into one or more partitions.
  - A Kafka partition is a linearly ordered sequence of messages, where each message is identified by their index .
  - All the data in a Kafka cluster is the disjointed union of partitions.
  - Incoming messages are written at the end of a partition and messages are sequentially read by consumers.
- Kafka provides both pub-sub and queue based messaging system in a fast, reliable, persisted, fault-tolerance and zero downtime manner.

Workflow of Pub-Sub Messaging

- ❑ Producers send message to a topic at regular intervals.
- ❑ Kafka broker stores all messages in the partitions configured for that particular topic. It ensures the messages are equally shared between partitions.
- ❑ Consumer subscribes to a specific topic.
- ❑ Once the consumer subscribes to a topic, Kafka will provide the current offset of the topic to the consumer and also saves the offset in the Zookeeper ensemble.
- ❑ Consumer will request the Kafka in a regular interval for new messages.
- ❑ Once Kafka receives the messages from producers, it forwards these messages to the consumers.
- ❑ Consumer will receive the message and process it.
- ❑ Once the messages are processed, consumer will send an acknowledgement to the Kafka broker.
- ❑ Once Kafka receives an acknowledgement, it changes the offset to the new value and updates it in the Zookeeper. Since offsets are maintained in the Zookeeper, the consumer can read next message correctly even during server outrages.
- ❑ This above flow will repeat until the consumer stops the request.
- ❑ Consumer has the option to rewind/skip to the desired offset of a topic at any time and read all the subsequent messages.

Reference

- Real-Time Analytics , Byron Ellis
  - https://kafka.apache.org/quickstart
  - https://kafka.apache.org/intro.html
  - http://tutorialspoint.com