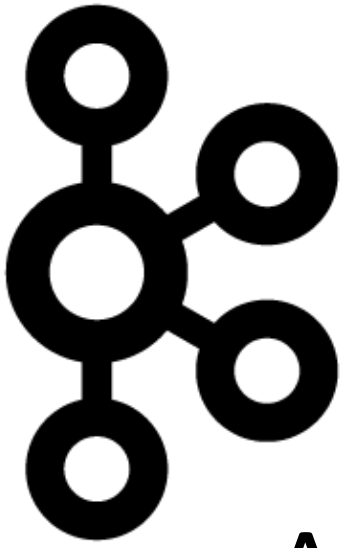


kafka

Apache Kafka

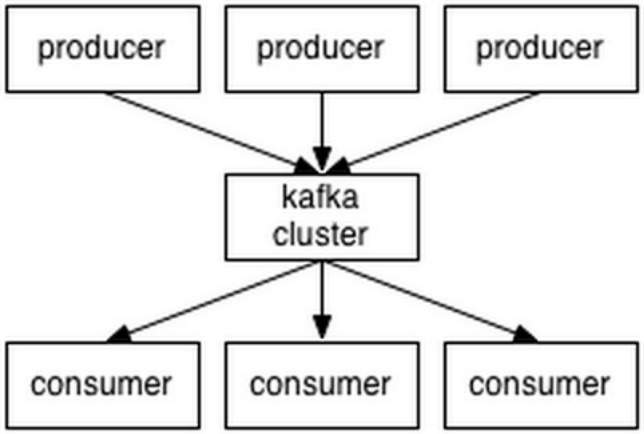
Slides modified from
CMSC 491: Hadoop-Based Distributed Computing
Spring 2016 Adam Shook

Overview

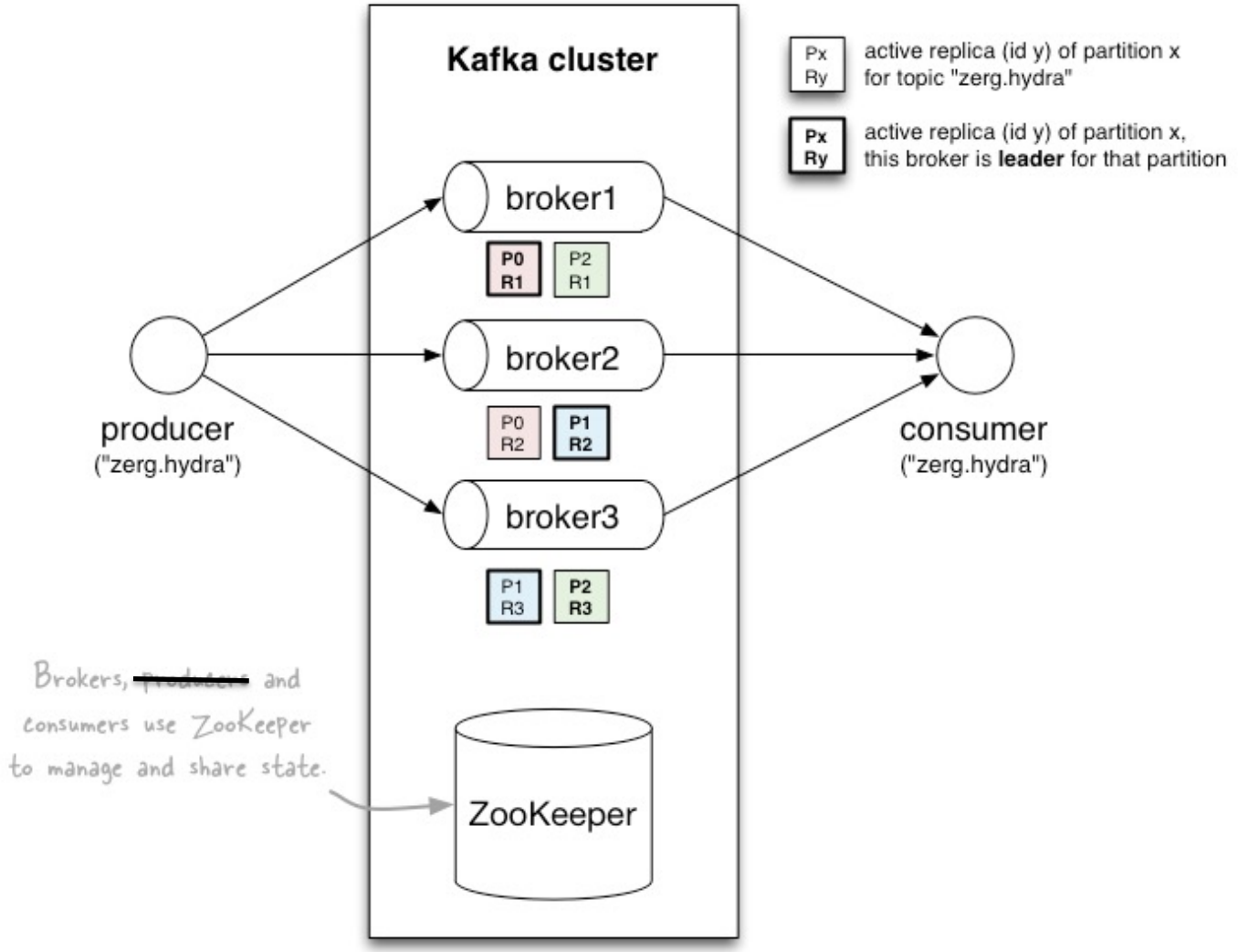
- Kafka is a “publish-subscribe messaging rethought as a distributed commit log”
- Fast
- Scalable
- Durable
- Distributed

A first look

- The who is who
  - **Producers** write data to **brokers**.
  - **Consumers** read data from **brokers**.
  - All this is distributed, including
    - Producers
    - Kafka brokers and storage
    - Consumers
- The data
  - Data is stored in **topics**.
  - **Topics** are split into **partitions**, which are **replicated**.
  - Brokers run on different servers and manage data on the servers



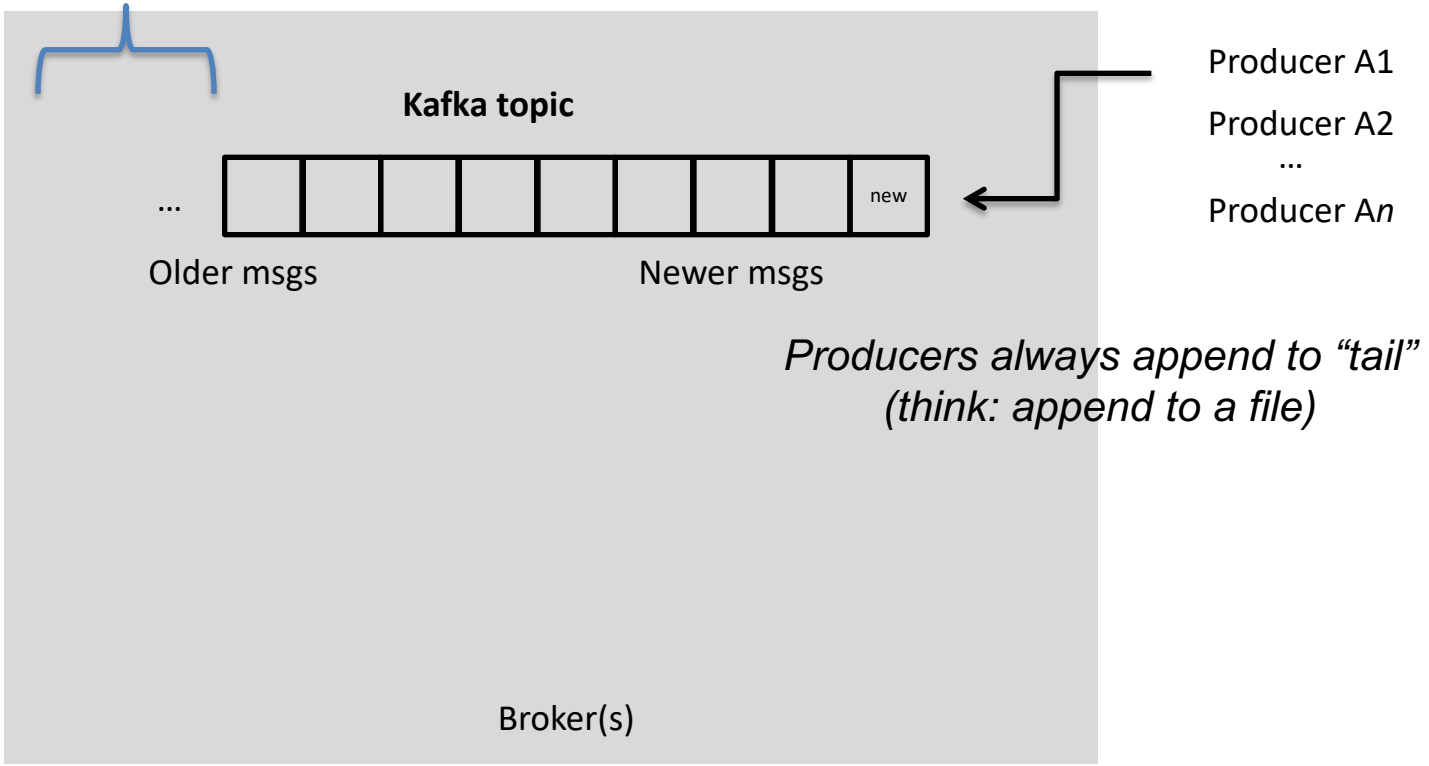
A first look


Kafka cluster
Px Ry
active replica (id y) of partition x for topic "zerg.hydra"
Px Ry
active replica (id y) of partition x, this broker is leader for that partition
broker1
P0 R1
P2 R1
broker2
P0 R2
P1 R2
broker3
P1 R3
P2 R3
producer ("zerg.hydra")
consumer ("zerg.hydra")
Brokers, ~~producers~~ and consumers use ZooKeeper to manage and share state.
ZooKeeper

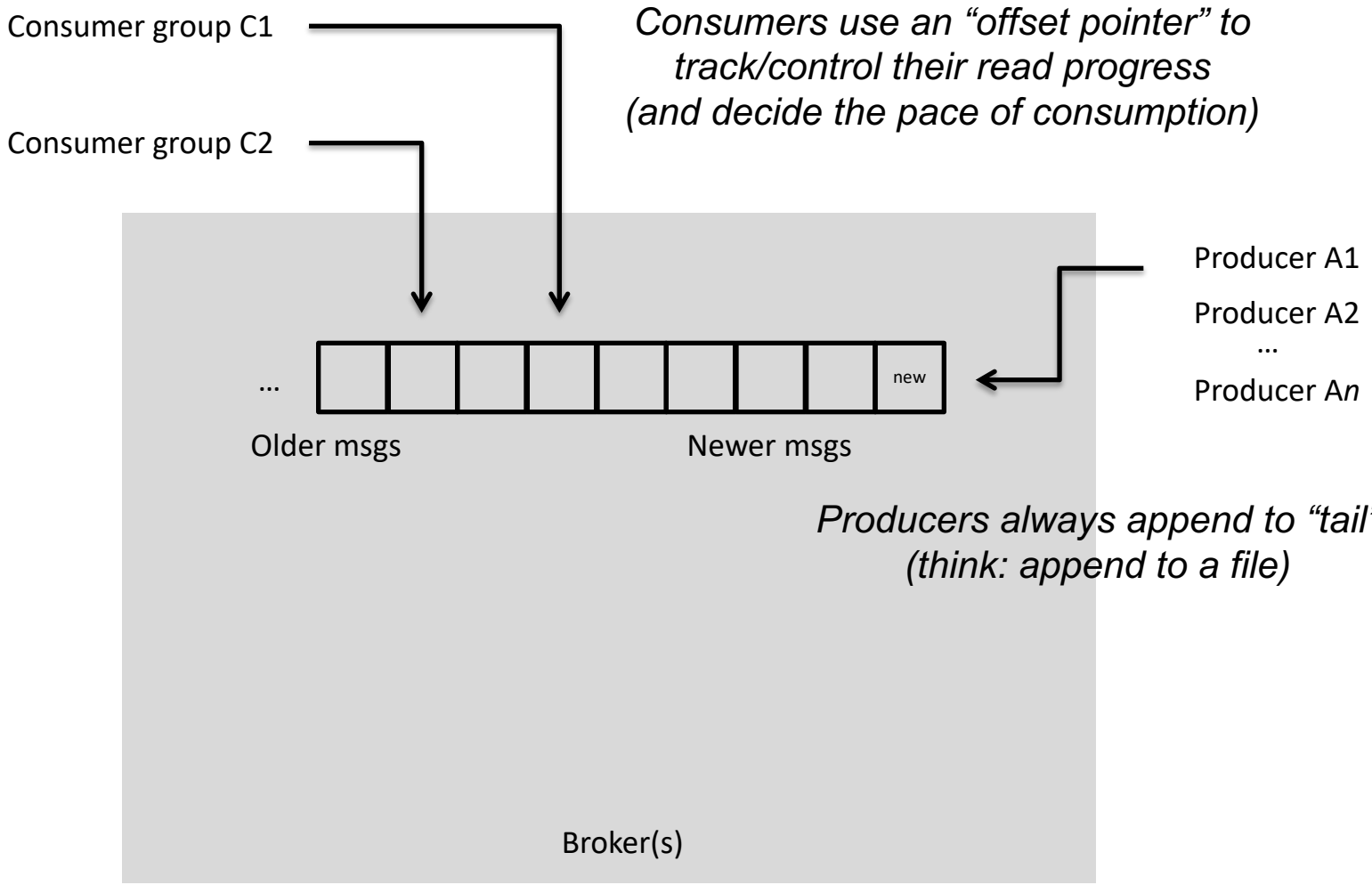
Topics

- **Topic:** feed name to which messages are published
  - Example: “zerg.hydra”

Kafka prunes “head” based on age or max size or “key”

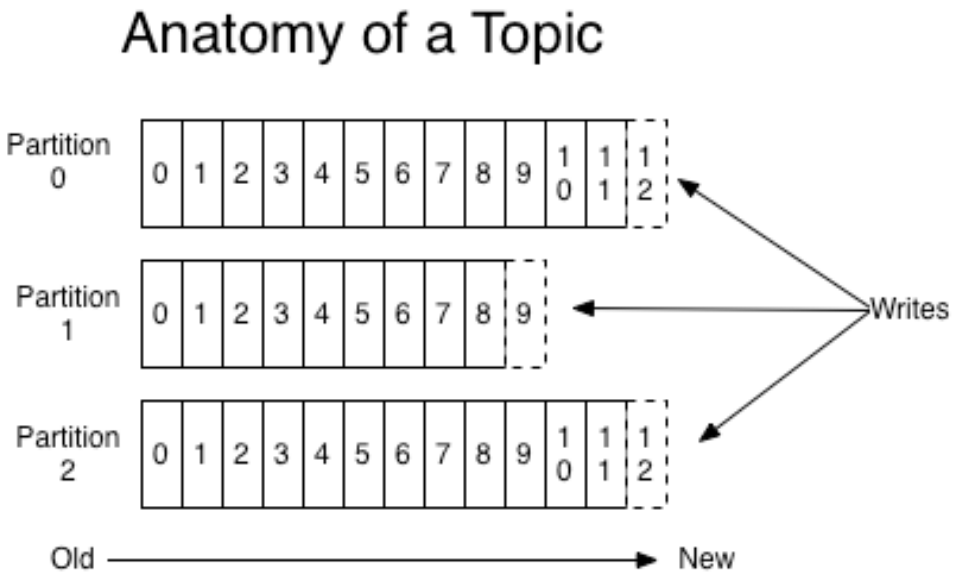


Topics


Consumer group C1
Consumer group C2
Consumers use an "offset pointer" to track/control their read progress (and decide the pace of consumption)
Producer A1
Producer A2
…
Producer An
…
new
Older msgs
Newer msgs
Producers always append to "tail" (think: append to a file)
Broker(s)

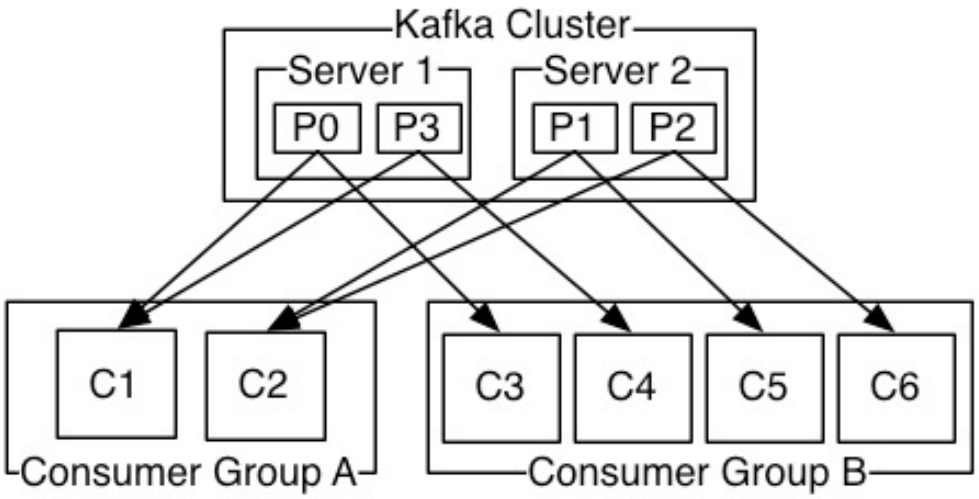
Partitions

- A topic consists of **partitions.**
- Partition: **ordered + immutable** sequence of messages that is continually appended to



Partitions

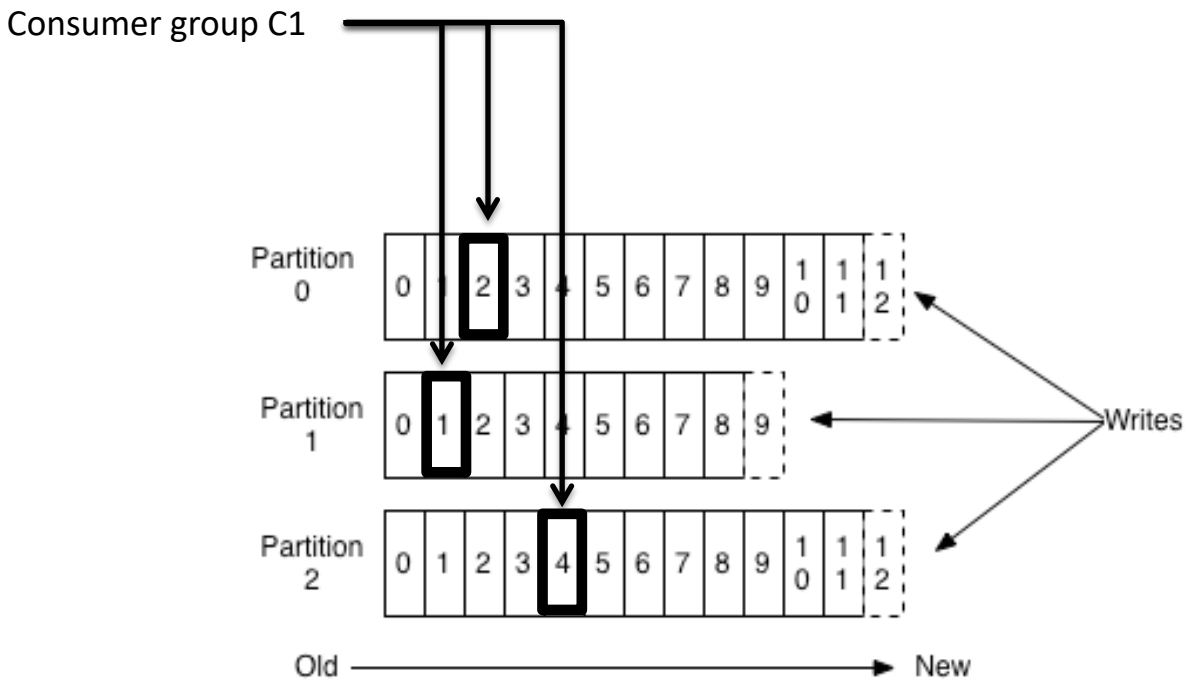
- #partitions of a topic is configurable
- #partitions determines **max** consumer (group) parallelism



  - Consumer group A, with 2 consumers, reads from a 4-partition topic
  - Consumer group B, with 4 consumers, reads from the same topic

Partition offsets

- **Offset**: messages in the partitions are each assigned a unique (per partition) and sequential id called the *offset*
  - Consumers track their pointers via *(offset, partition, topic)* tuples



Replicas of a partition

- **Replicas:** “backups” of a partition
  - They exist solely to prevent data loss.
  - Replicas are never read from, never written to.
    - They do NOT help to increase producer or consumer parallelism!
  - Kafka tolerates *(numReplicas - 1)* dead brokers before losing data
    - LinkedIn: numReplicas == 2 → 1 broker can die

Brokers and Zookeeper

- Apache ZooKeeper is a high-reliability service for coordination of high-reliability distributed applications
- ZooKeeper
  - handles the leadership election of Kafka brokers
  - who the preferred leader node is for a given topic/partition pair
  - And other tasks to ensure system runs reliably despite failures of servers