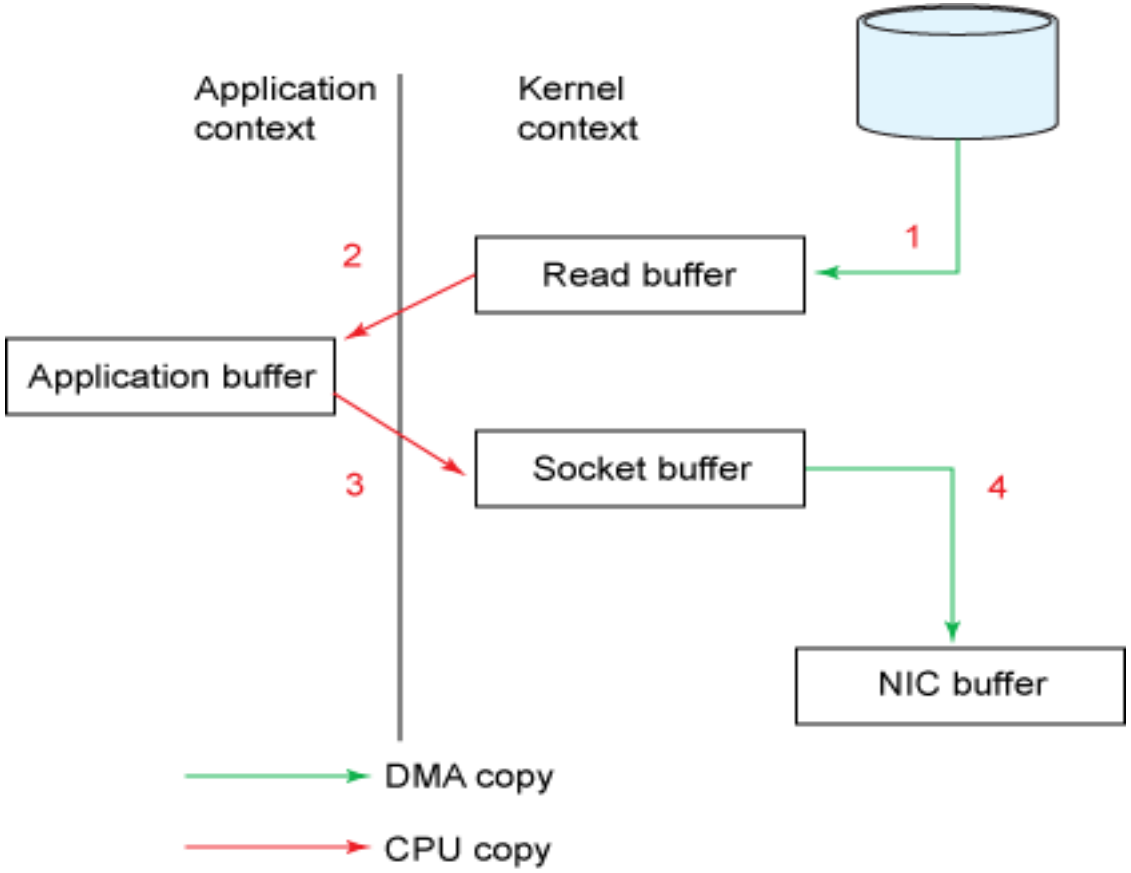
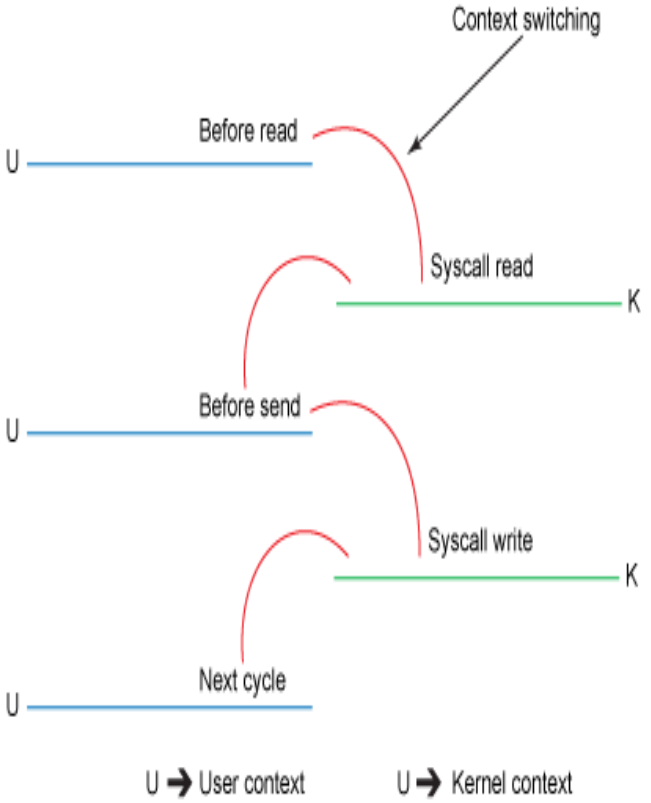
Kafka Topics Architecture

No Zero Copy

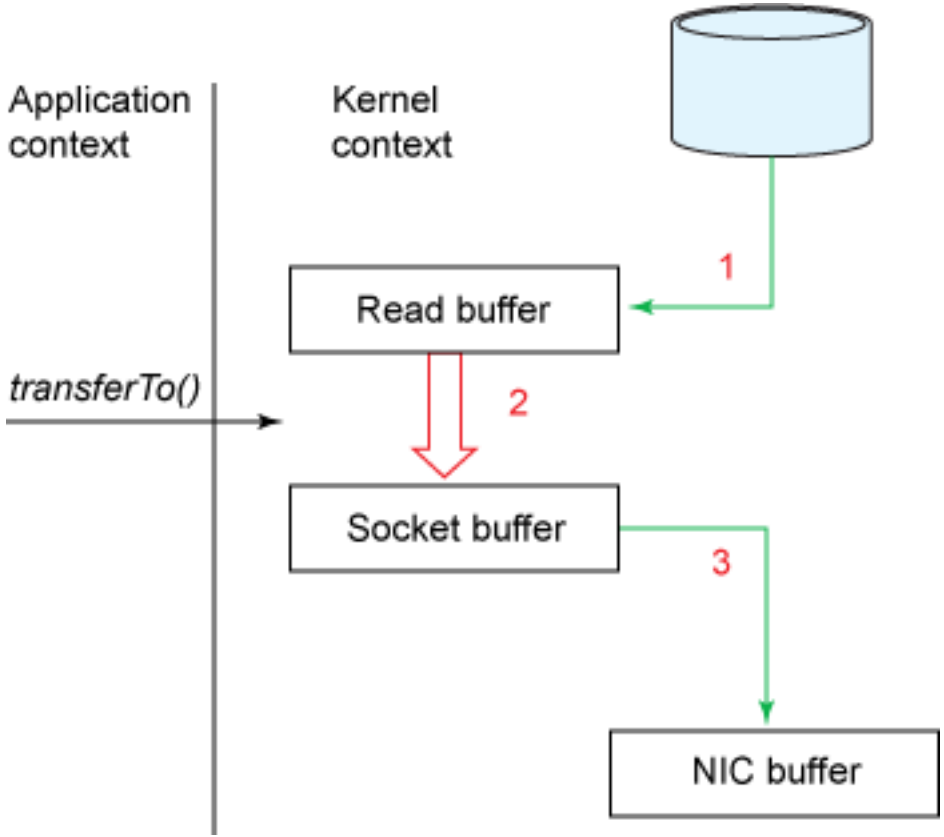
Tos

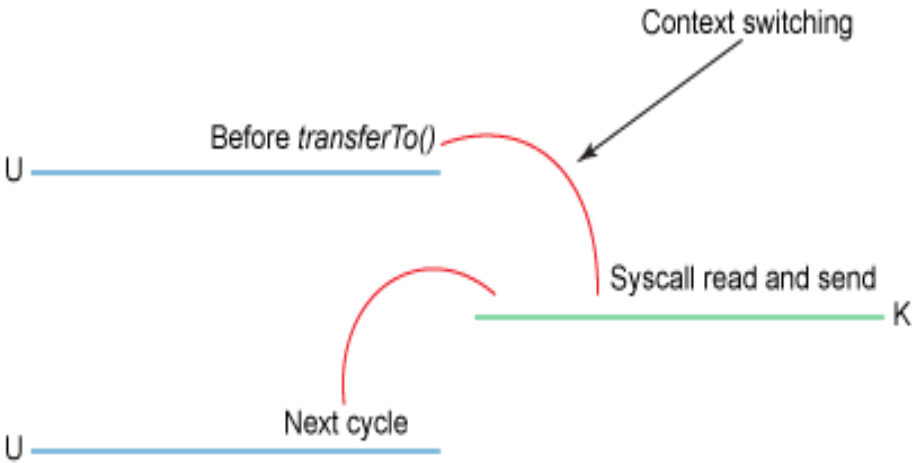




Zero Copy

Tos

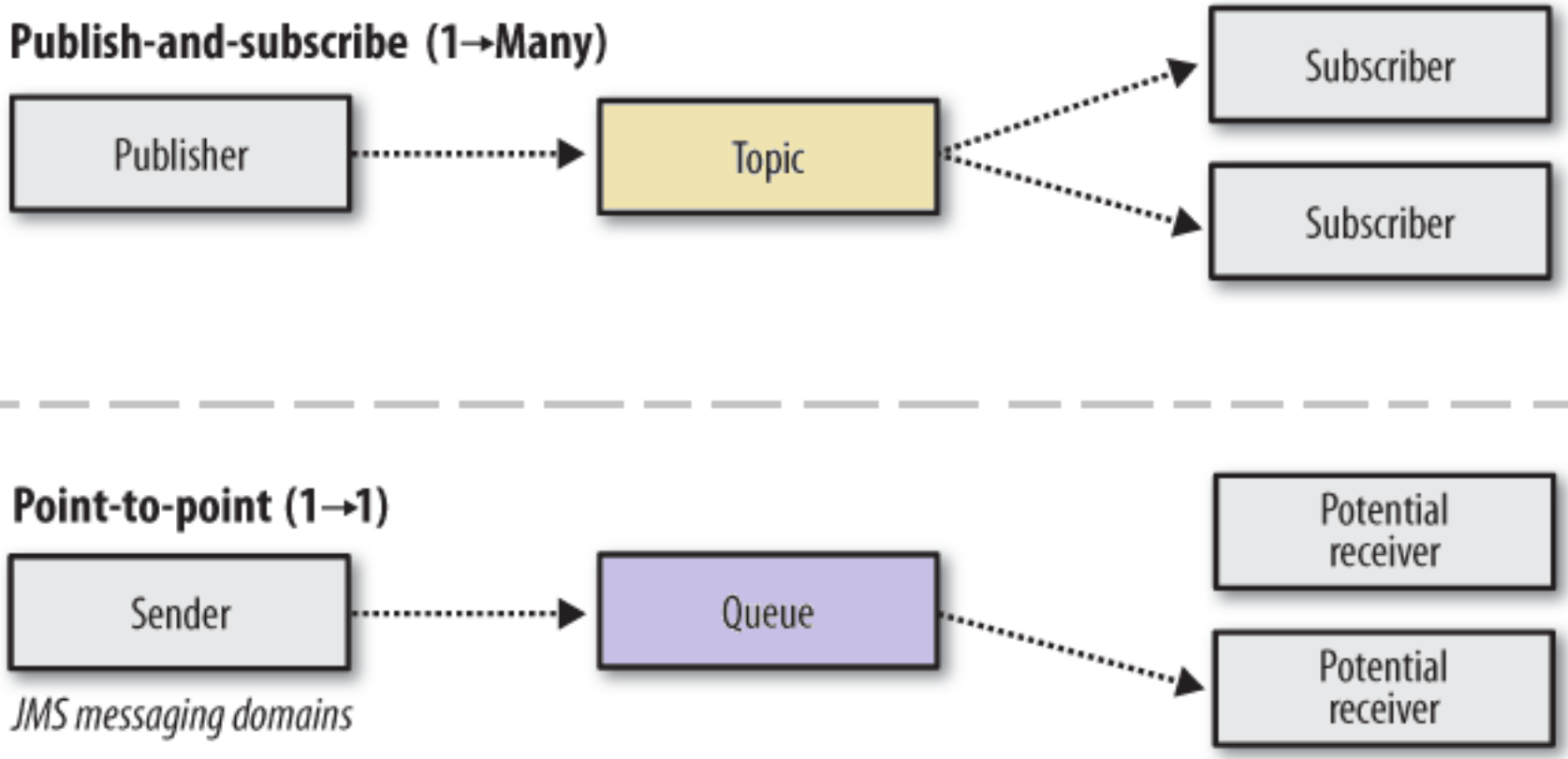




Kafka versus MOM

Messaging - Architecture

Tos



JMS messaging domains

Kafka vs JMS vs RabbitMQ Messaging

Tos

- Is Kafka a Queue or a Pub/Sub/Topic?
  - Yes
- Kafka is like a Queue per consumer group
  - Kafka is a queue system per consumer in consumer group so load balancing like JMS, RabbitMQ queue
- Kafka is like Topics in JMS, RabbitMQ, MOM
  - Topic/pub/sub by offering Consumer Groups which act like subscriptions
  - Broadcast to multiple consumer groups
- MOM = JMS, ActiveMQ, RabbitMQ, IBM MQ Series, Tibco, etc.

Kafka vs MOM

Tos

- By design, Kafka is better suited for scale than traditional MOM systems due to partition topic log
  - Load divided among Consumers for read by partition
  - Handles parallel consumers better than traditional MOM
- Also by moving location (partition offset) in log to client/consumer side of equation instead of the broker, less tracking required by Broker and more flexible consumers
- Kafka written with mechanical sympathy, modern hardware, cloud in mind
  - Disks are faster
  - Servers have tons of system memory
  - Easier to spin up servers for scale out

Kafka Fundamentals

Tos

- ***Records*** have a ***key (optional)***, ***value*** and ***timestamp; Immutable***
- ***Topic*** a stream of records (“/orders”, “/user-signups”), feed name
  - ***Log*** topic storage on disk
  - ***Partition*** / Segments (parts of Topic Log)

Kafka Design Motivation Scale

Tos

- To scale Kafka is
  - distributed,
  - supports sharding
  - load balancing
- Scaling needs inspired Kafka partitioning and consumer model
- Kafka scales writes and reads with partitioned, distributed, commit logs

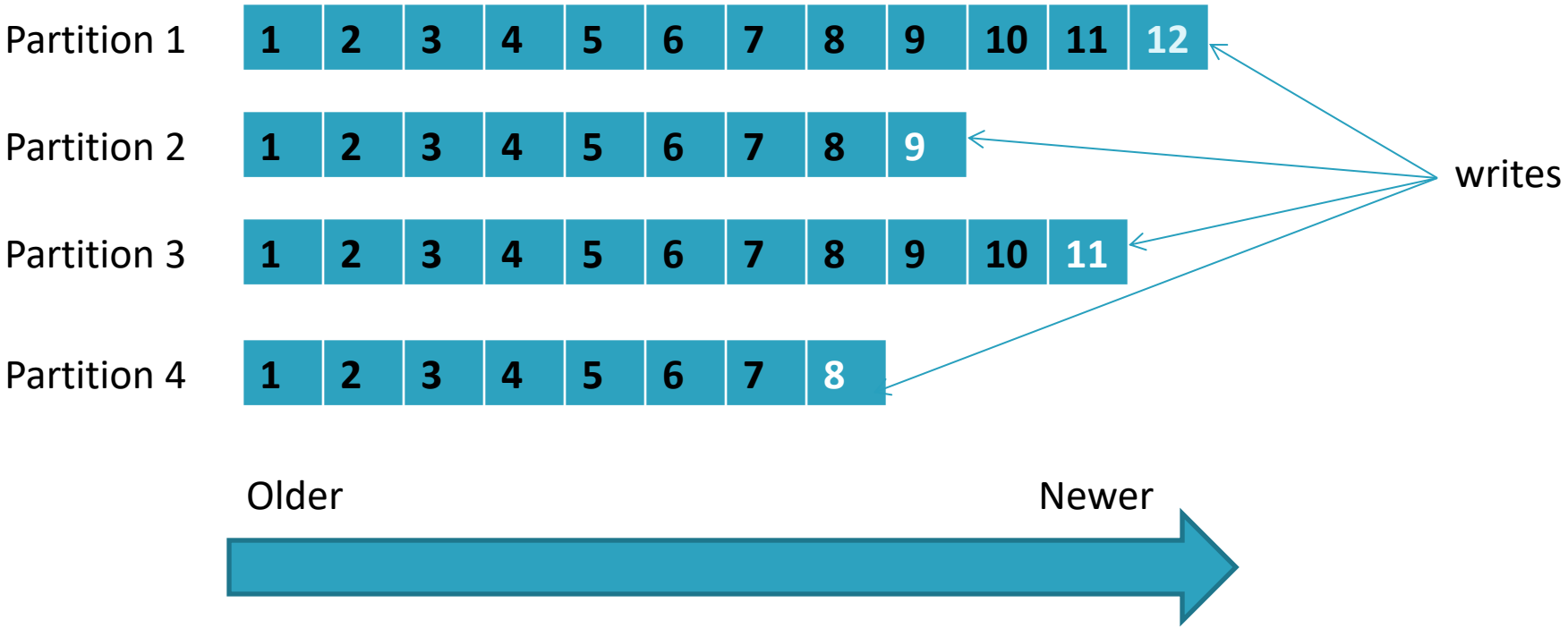
Persistence: Embrace Filesystem

Tos

- Kafka relies heavily on filesystem for storing and caching messages/records
- Disk performance of hard drives performance of sequential writes is fast
  - JBOD with six 7200rpm SATA RAID-5 array clocks at 600MB/sec
  - Heavily optimized by operating systems
- Ton of cache: Operating systems use available of main memory for disk caching
- JVM GC overhead is high for caching objects | OS file caches are almost free
- Kafka greatly simplifies code for cache coherence by using OS page cache
- Kafka disk does sequential reads easily optimized by OS page cache

Kafka Topic Partitions Layout

Tos



Topics, Logs, Partitions

Tos

- Kafka ***Topic*** is a stream of records
- ***Topics*** stored in log
- ***Log*** broken up into ***partitions*** and ***segments***
- ***Topic*** is a category or stream name or feed
- Topics are pub/sub
  - Can have zero or many subscribers - consumer groups
- ***Topics*** are broken up and spread by partitions for speed and size

Topic Partitions

Tos

- ***Topics*** are broken up into ***partitions***
- ***Partitions*** decided usually by key of record
  - Key of record determines which partition
- ***Partitions*** are used to scale Kafka across many servers
  - Record sent to correct partition by key
- ***Partitions*** are used to facilitate parallel consumers
  - Records are consumed in parallel up to the number of partitions
- Order guaranteed per partition
- Partitions can be ***replicated*** to multiple brokers

Topic Partition Log

Tos

- ***Order*** is maintained only in a single ***partition***
  - ***Partition*** is ordered, immutable sequence of records that is continually appended to—a structured commit ***log***
- ***Records*** in partitions are assigned ***sequential id*** number called the *offset*
- ***Offset*** identifies each record within the partition
- ***Topic Partitions*** allow Kafka log to scale beyond a size that will fit on a single server
  - Topic partition must fit on servers that host it
  - topic can span many partitions hosted on many servers

Topic Parallelism and consumers

Tos

- Topic Partitions are unit of ***parallelism***
  partition can only be used by one consumer in group at a time
- Consumers can run in their own process or their own thread
- If a consumer stops, Kafka spreads partitions across remaining consumer in group
- #of Consumers you can run per Consumer Group limited by #of Partitions
- Consumers getting assigned partition aids in efficient message consumption tracking

Kafka Scale and Speed

Tos

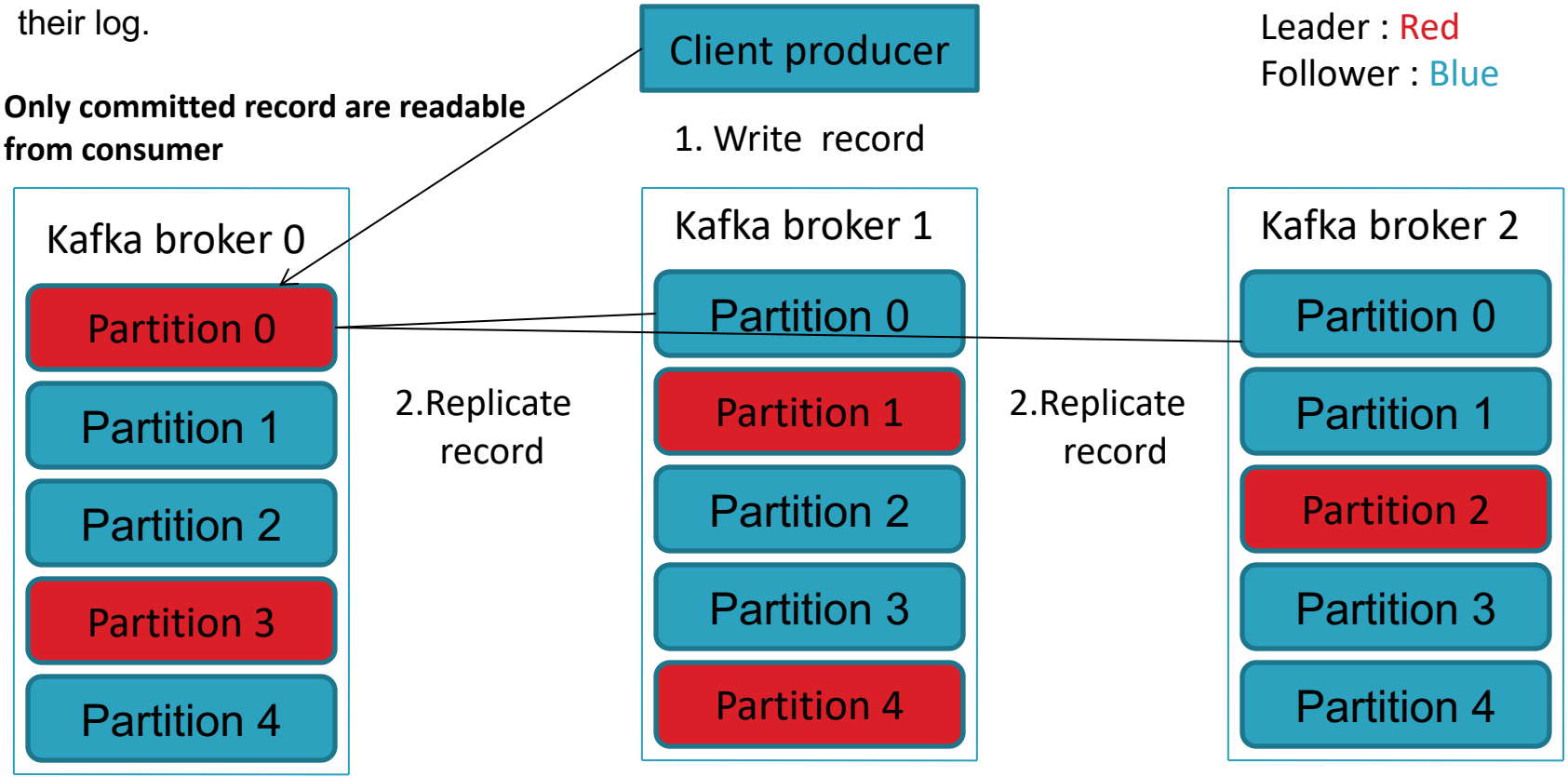
- How can Kafka scale if multiple producers and consumers read/write to same Kafka Topic log?
- Writes fast: Sequential writes to file system are ***fast*** (700 MB or more a second)
- Scales writes and reads by ***sharding:***
  - Topic logs into ***Partitions*** (parts of a Topic log)
  - Topics logs can be split into multiple Partitions ***different machines/different disks***
  - Multiple Producers can write to different Partitions of the same Topic
  - Multiple Consumers Groups can read from different partitions

Kafka Replication to Partition 0

Tos

Record is considered “committed”. When all ISRs for partition wrote to their log.

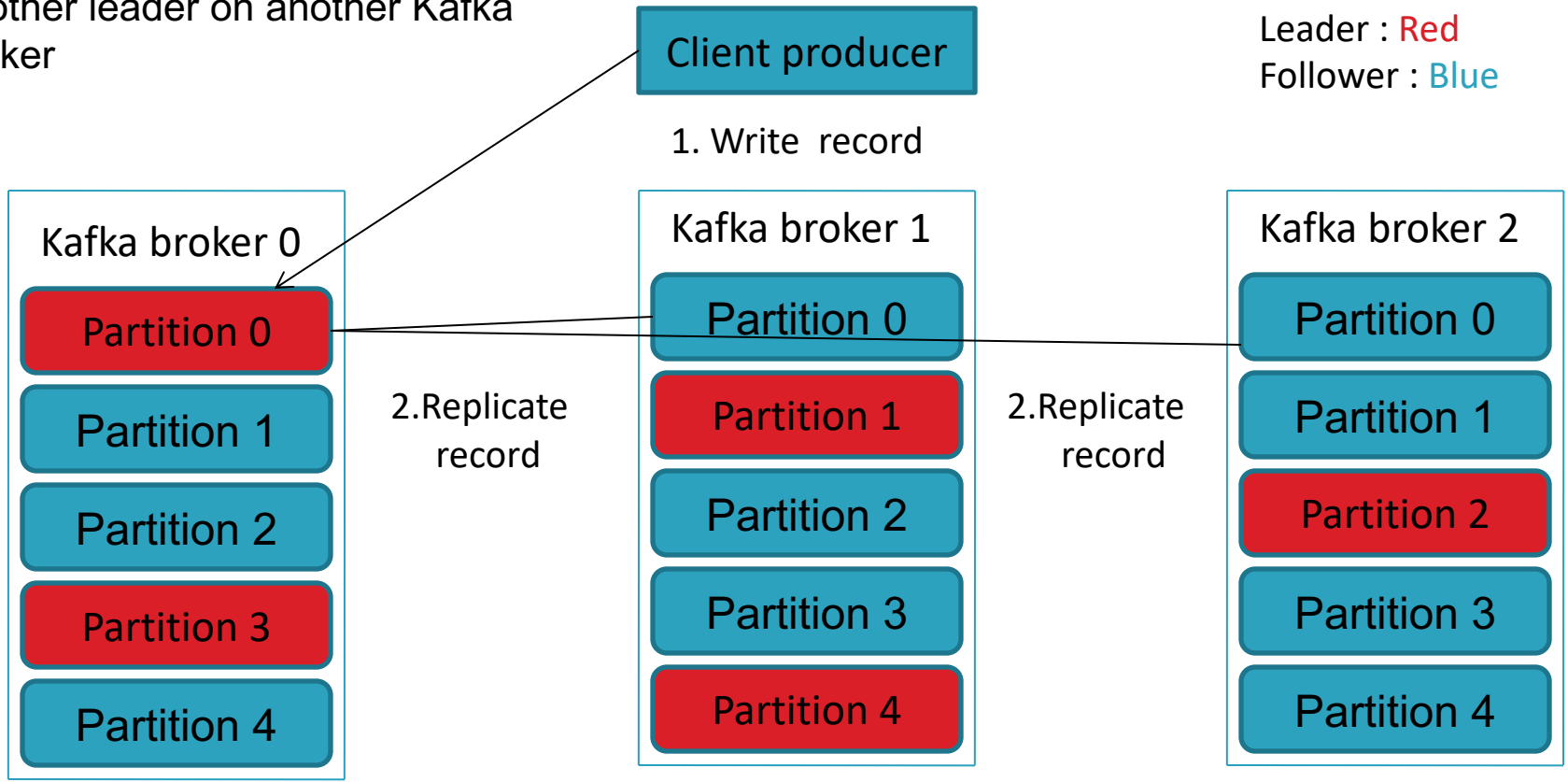
Only committed record are readable from consumer



Kafka Replication to Partition 1

Tos

Another partition can be owned by another leader on another Kafka broker



Replication: Kafka Partition Distribution

Tos

- Each partition has ***leader server*** and zero or more ***follower servers***
  - ***Leader*** handles all read and write requests for partition
  - ***Followers*** replicate leader, and take over if leader dies
  - Used for parallel consumer handling within a group
- Partitions of log are distributed over the servers in the Kafka cluster with each server handling data and requests for a share of partitions
- Each partition can be replicated across a configurable number of Kafka servers - Used for fault tolerance

Replication: Kafka Partition leader

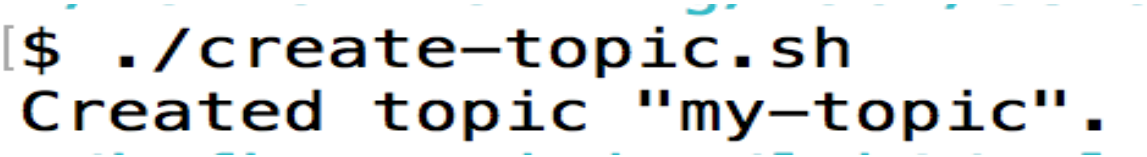
Tos

- One node/partition's replicas is chosen as ***leader***
- Leader handles all reads and writes of Records for partition
- Writes to partition are ***replicated*** to ***followers*** (node/partition pair)
- An ***follower*** that is ***in-sync*** is called an ***ISR (in-sync replica)***
- If a partition leader fails, one ISR is chosen as new leader

Create Kafka Topic

Tos

#/opt/kafka/bin/kafka-topics.sh --create –bootstrap-server localhost:9020 \
--replication-factor 1 --partitions 12 --topic my-topic



List Topics

Tos

#/opt/kafka/bin/kafka-topics.sh --list --bootstrap-server localhost:9020

```
~/kafka-training/lab1/solution
$ ./list-topics.sh
__consumer_offsets
_schemas
my-example-topic
my-example-topic2
my-topic
new-employees
```

Lab - Basic Kafka Operations - CLI (Topic) – 30 Mins