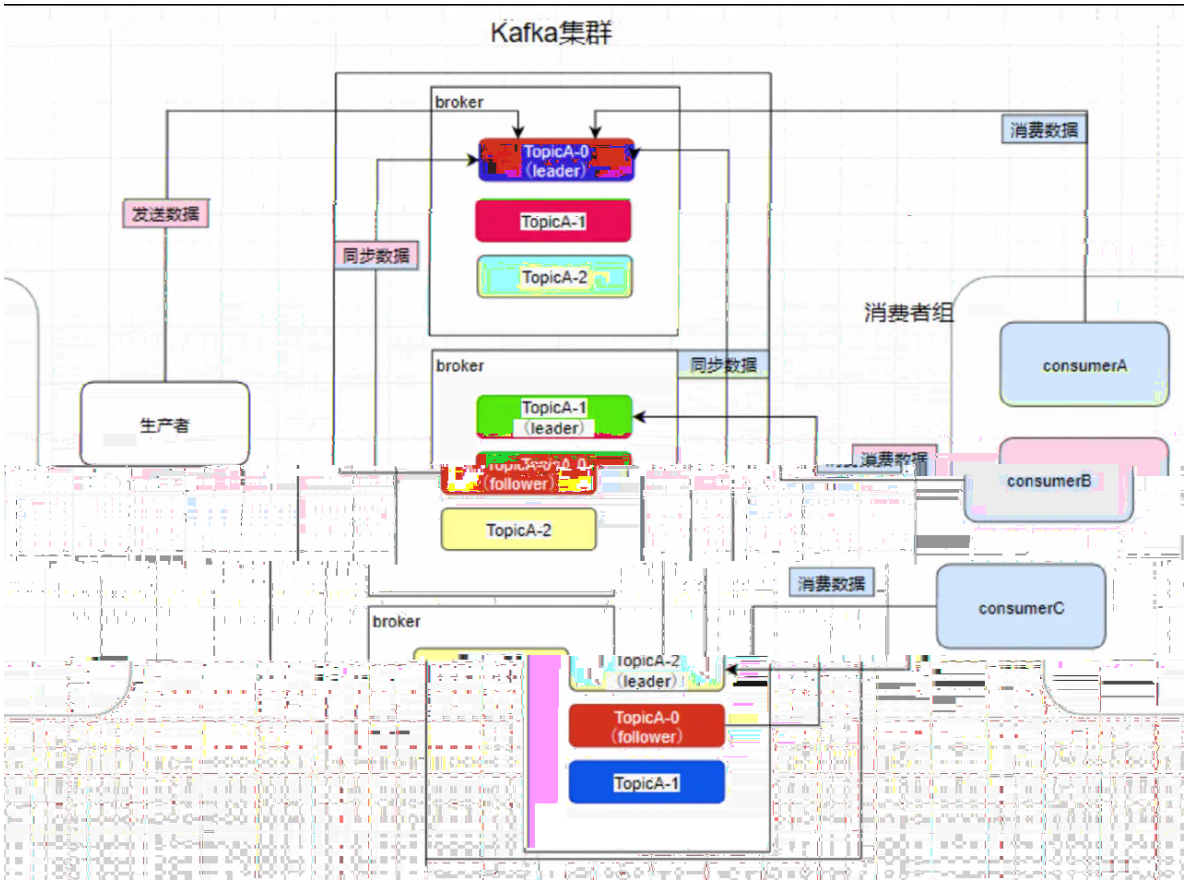
简介

> Apache Kafka - , , . Kafka . Kafka , . Kafka ZooKeeper . Apache Storm Spark , .
>
> Kafka ,
>
> Zookeeper

架构



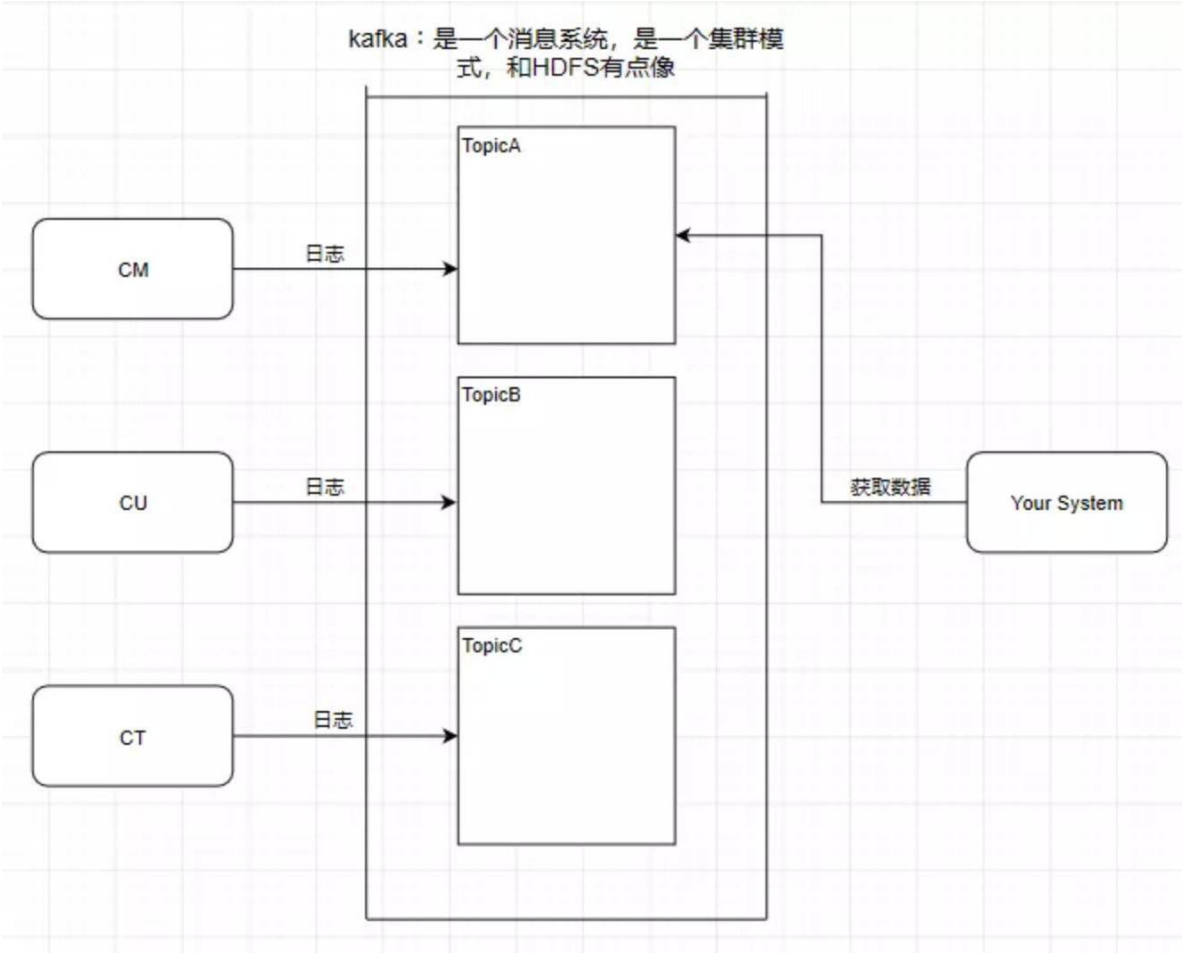
基础概念

Broker

> Server. Topic , Partition, Replica. Producer Consumer
>
> : Controller, Kafka Zookeeper
> Broker . Controller. ( ),
> , ,

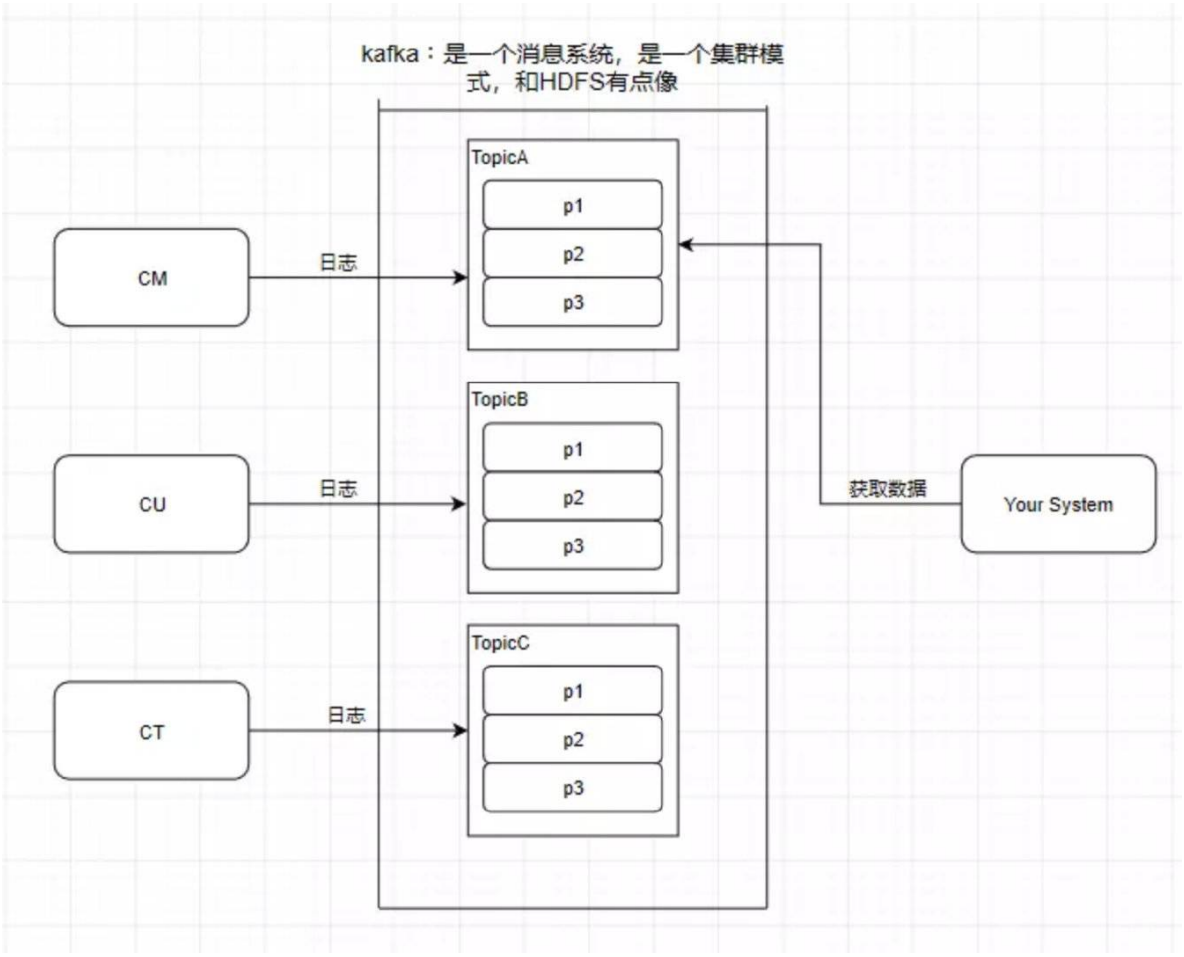
Topic

> MQ Queue. Kafka Topic Partition (Broker)



Partition (分区)

> MQ Queue ,
>
> Producer Message, key partition hash,
>
> Consumer Group Consumer .



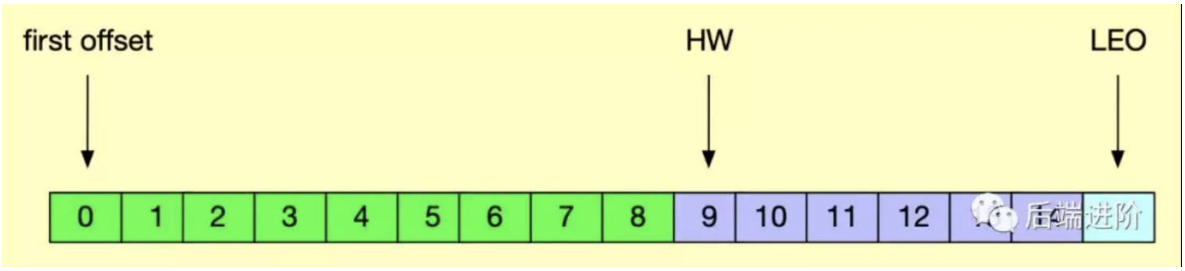
Replica (备份)

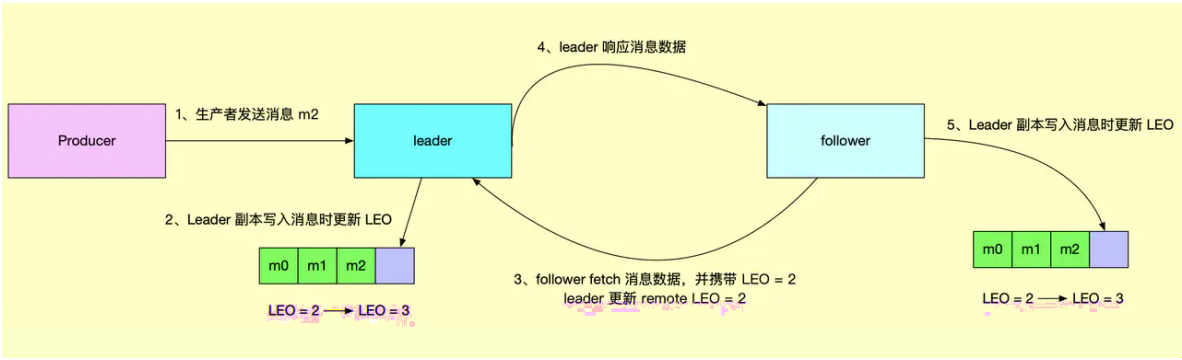
> Partition . Replica Broker
>
> Бeader: Replica , Partition Leader (Broker). Producer , Leader . Consumer Leader Follower: Replica , . Leader pull ( )

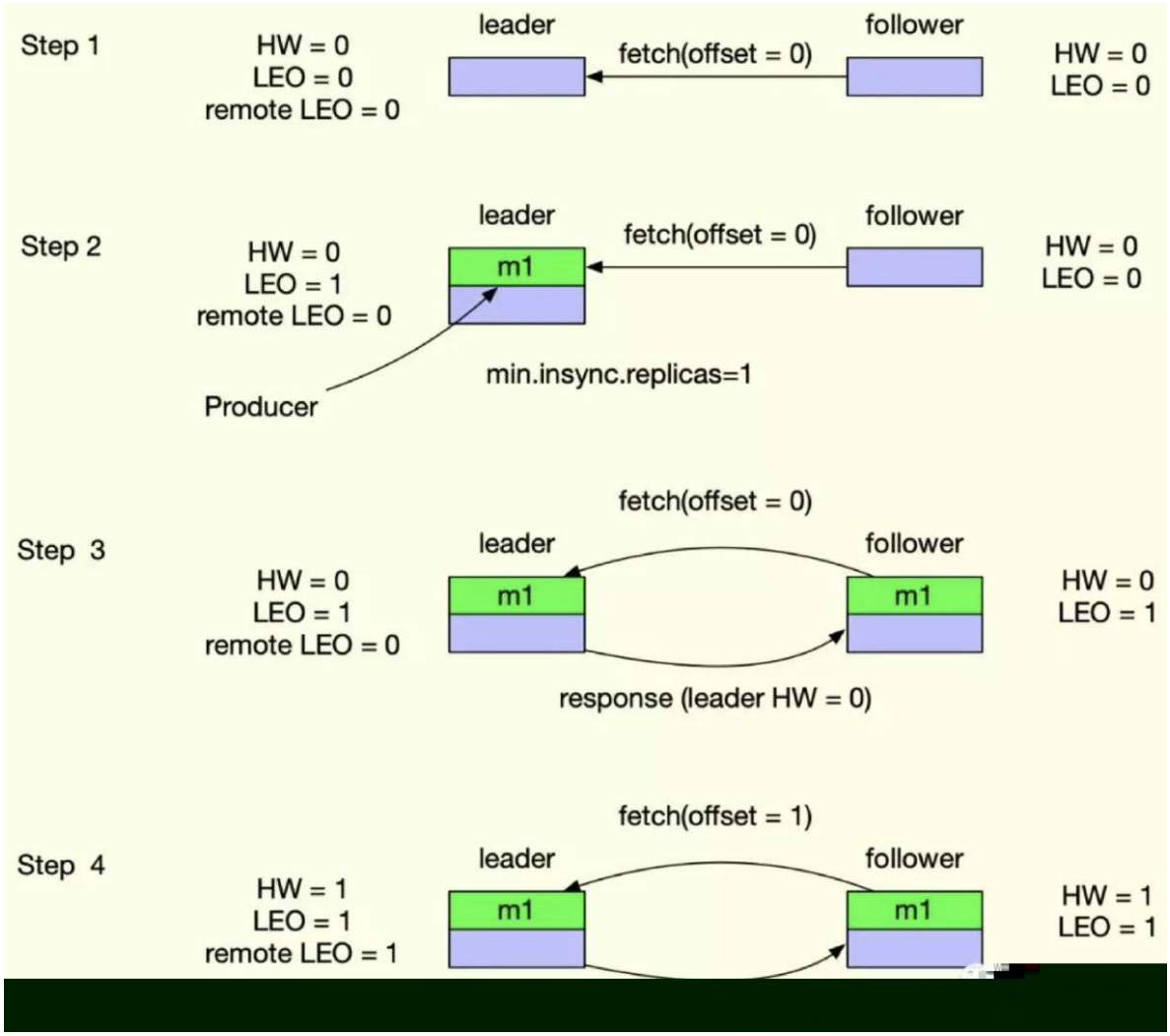
```
rerplica.lag.time.max.ms=10000
# 如果leader发现flower超过10秒没有向它发起fech请求，那么leader考虑这个flower是不是程序出了点问题
# 或者资源紧张调度不过来，它太慢了，不希望它拖慢后面的进度，就把它从ISR中移除.

rerplica.lag.max.messages=4000
# 相差4000条就移除
# flower慢的时候，保证高可用性，同时满足这两个条件后又加入ISR中，
# 在可用性与一致性做了动态平衡    亮点

min.insync.replicas=1
# 需要保证ISR中至少有多少个replica
```

水印备份机制

> LEO (last end offset): , , , LEO Leader LEO , LEO , remote LEO . Follower fetch LEO, Leader LEO HW
>
> HW (high watermark): , , HW LEO , HW " " " " ,







Message

> MQ Queue Message.

Producer

> MQ . Broker push ( )

数据一致性保证 (消息不丢失)

```
request.required.asks=0
# 0:相当于异步的，不需要leader给予回复，producer立即返回，发送就是成功，
那么发送消息网络超时或broker crash(1.Partition的Leader还没有commit消息 2.Leader与
Follower数据不同步)，既有可能丢失也可能会重发
# 1：当leader接收到消息之后发送ack，丢会重发，丢的概率很小
# -1：当所有的follower都同步消息成功后发送ack. 不会丢失消息
```

Consumer

> MQ . Broker pull ( ) , 100ms . Consumer Partition
>
> : ack :

Consumer Group

> Kafka , Topic , Topic Consumer Group
> Consumer , Message Consumer
>
> : Consumer Group Consumer Partition , Partition

分片规则

> Kafka Replica : RangeAssignor RoundRobinAssignor
>
> RangeAssignor:
>
> 1. Broker(n Broker) Partition
> 2. i Partition (i mod n) Broker
> 3. i Partition j Replica ((i + j) mod n) Broker

Rebalance (重平衡)

> Rebalance , Consumer Group consumer ,
> Topic
>
> Rebalance , Consumer Group , Rebalance

Coordinator

> Group Coordinator , Broker Group
> Coordinator Group Meta , Partition Offset
> Kafka Topic(__consumer_offsets) Kafka 0.9 Zookeeper Partition
> offset (consumers/{group}/offsets/{topic}/{partition}), Zookeeper
> , 0.9 Topic Partition offset

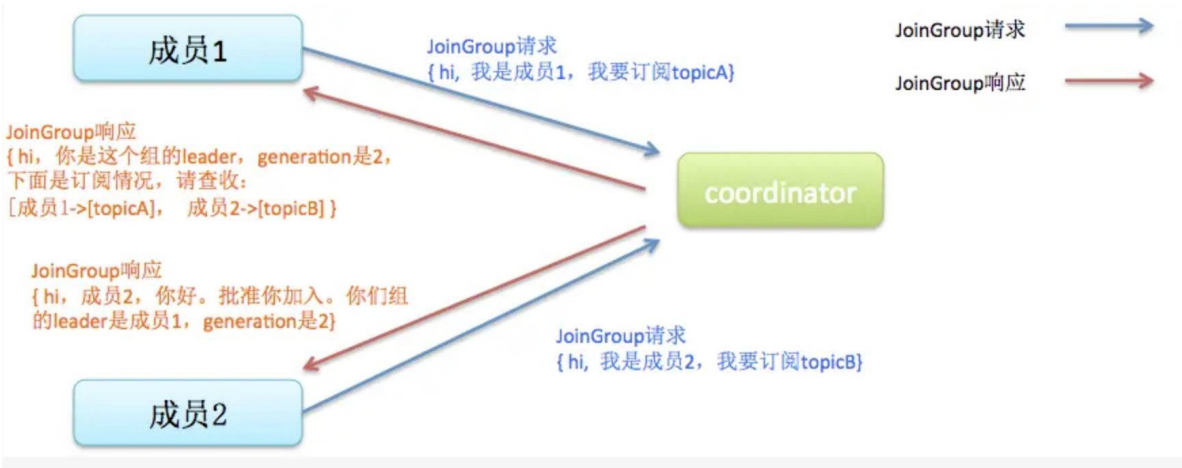
触发条件

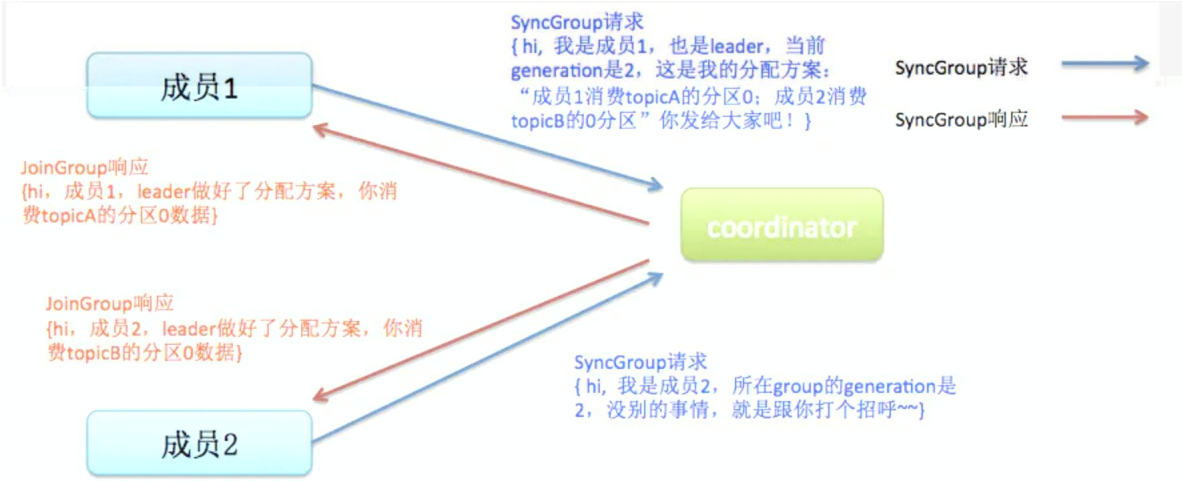
> 1.
>
> 1.
> 2.
> 3. . GC, Group
> Coordinator ,
>
> 2. Topic Consumer Group
> 3. Topic

Rebalace 流程

> Rebalance Join Sync
>
> 1. Join: . , Coordinator JoinGroup ,
> . JoinGroup , Coordinator
> Consumer Leader , Consumer Leader
> Consumer Leader Coordinator . Consumer Leader
>
> 2. Sync: Consumer Leader , Consumer Topic
> Partition. , Leader SyncGroup Coordinator,
> Leader SyncGroup , . Coordinator
> SyncGroup Response Consumer.

: Kafka Rebalance





如何避免 Rebalance

2 3, . 1 1 3 , 3

```
# 心跳相关
session.timeout.ms = 6s
heartbeat.interval.ms = 2s

# 消费时间
max.poll.interval.ms
```

日志索引

Kafka TB , : .

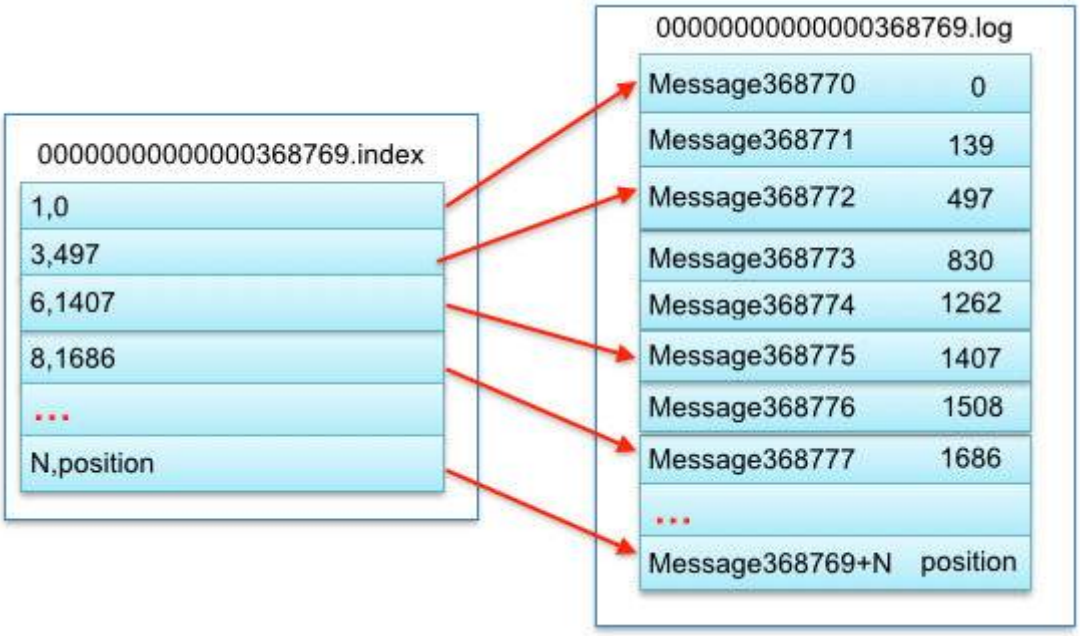
Kafka (1G) , , , , , . Kafka , offset

partition log (1G), log index offset Message :

1. offset (: 368773), offset index (368769.index)

2. (368773 - 368769 = 4) index (368769.index) offset log (3, 497)
3. (3, 497), (368773 830)

```
00000000000000000000.index
00000000000000000000.log
00000000000000368769.index
00000000000000368769.log
00000000000000737337.index
00000000000000737337.log
00000000000001105814.index
00000000000001105814.log
.....
```



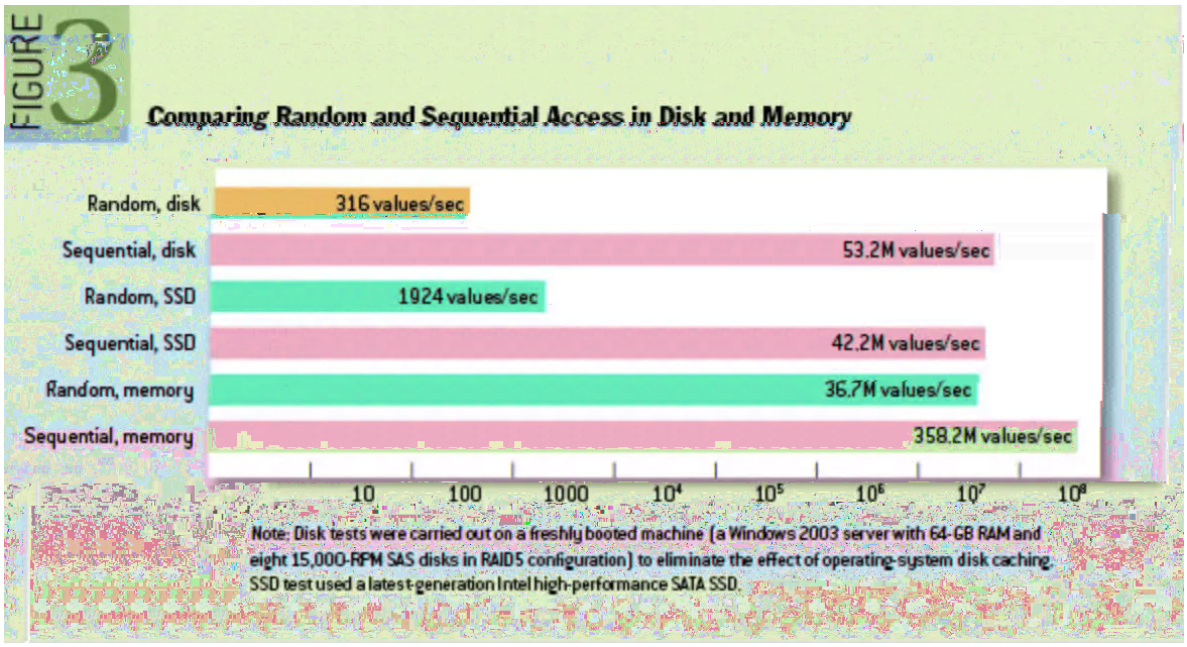
高性能, 高吞吐

分区的原因

> MQ Queue, ,
> . , , , .
> , . Kafka Partition, Kafka
> Partition . Topic Partition,
> . Java 1.7 ConcurrentHashMap , , ,

顺序写

> , .



批发送

> I/O . Kafka ,
> Overhead, . Kafka 0.82 , send
> ( )

```
# 批量发送的基本单位, 默认是16384Bytes, 即16kB
batch.size

# 延迟时间
linger.ms

# 两者满足其一便发送
```

数据压缩

> , . Batch
> ,

> Broker Consumer
>
> : Producer Broker, , Broker Consumer ,

Page Cache & MMap

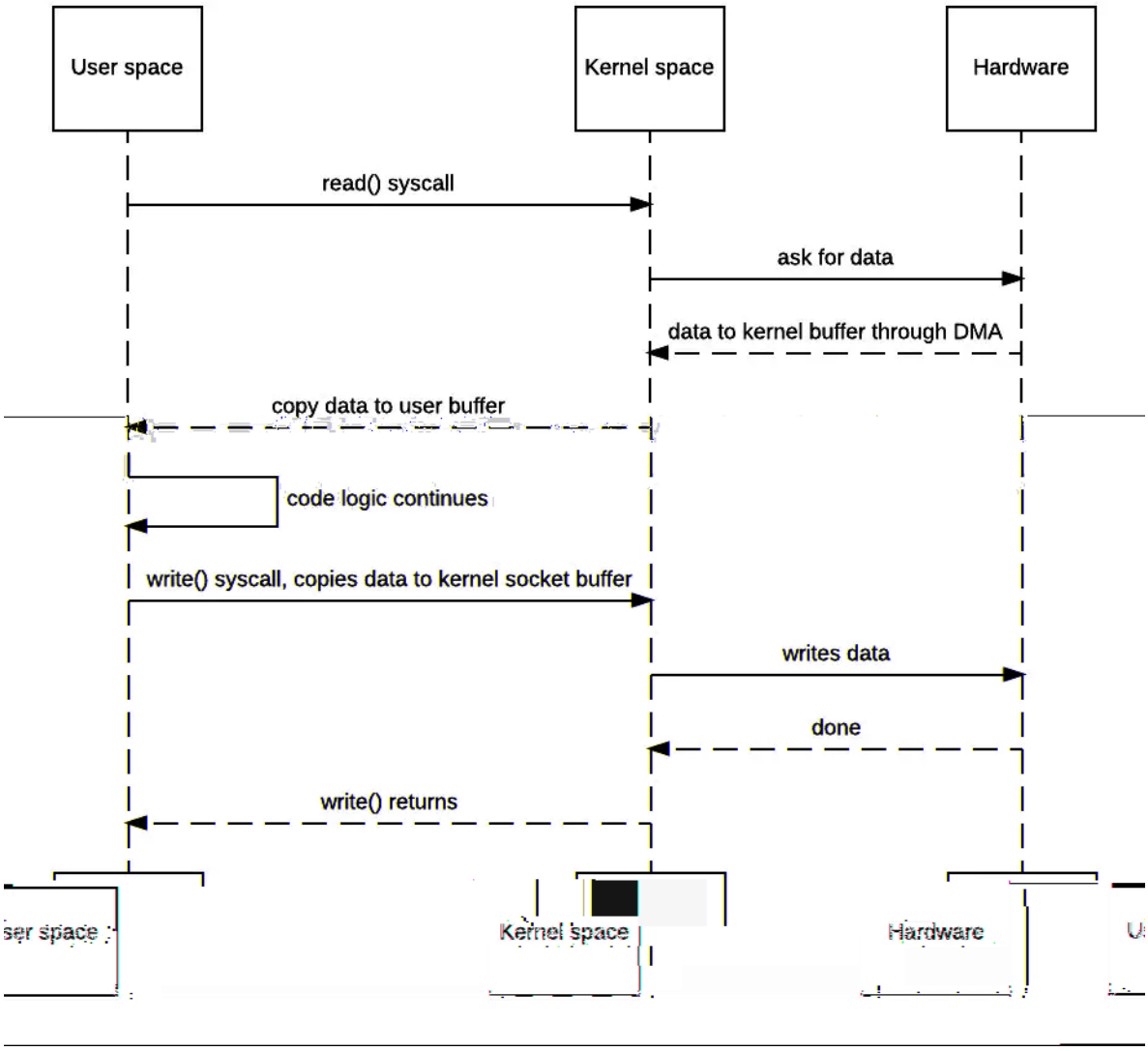
> MMap Page Cache, Page Cache , MMap Page Cache. MMap , Page Cache ,

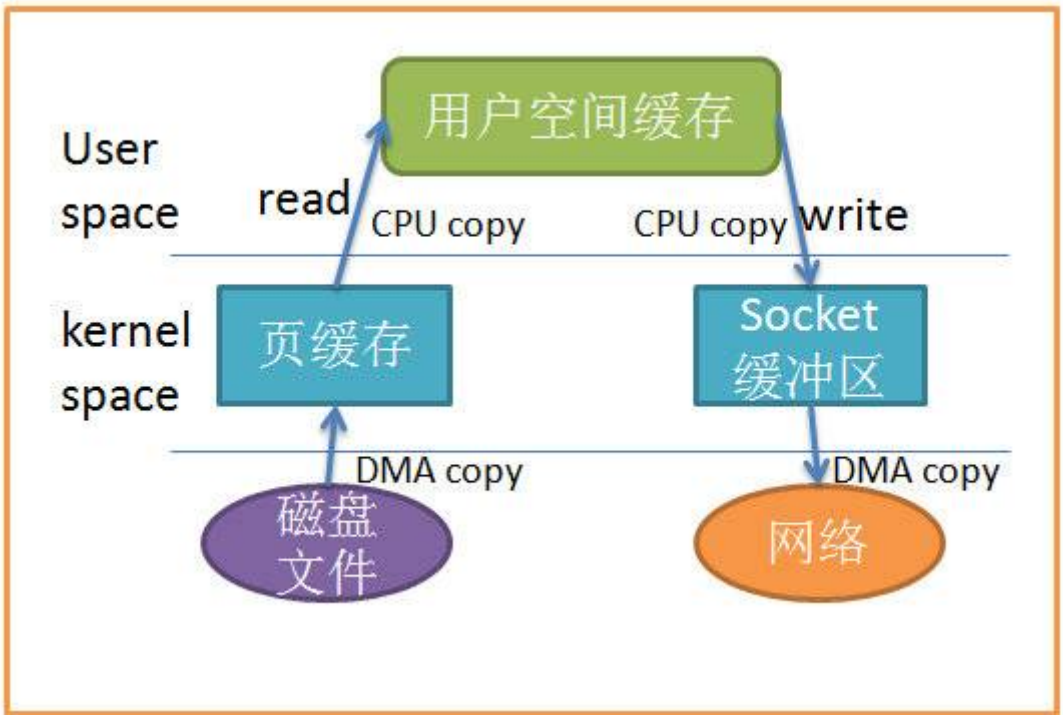
Page Cache

> page cache,
> page cache , page cache (
> fsync), ,
>
> : page cache flusher sync()/fsync()
> ,

MMap (Memory Mapped Files, 内存映射文件)

> MMap . Java MappedByteBuffer
>
> :
>
> 1. JVM OS read() , ,
> 2. ( ) (DMA)
> 3. , read() ,
> 4. JVM OS write() , ,
> 5. Socket
> 6. Socket DMA ( ) , write() ,
>
> : , , ,
> : , , DMA
> ,
>
> , , .

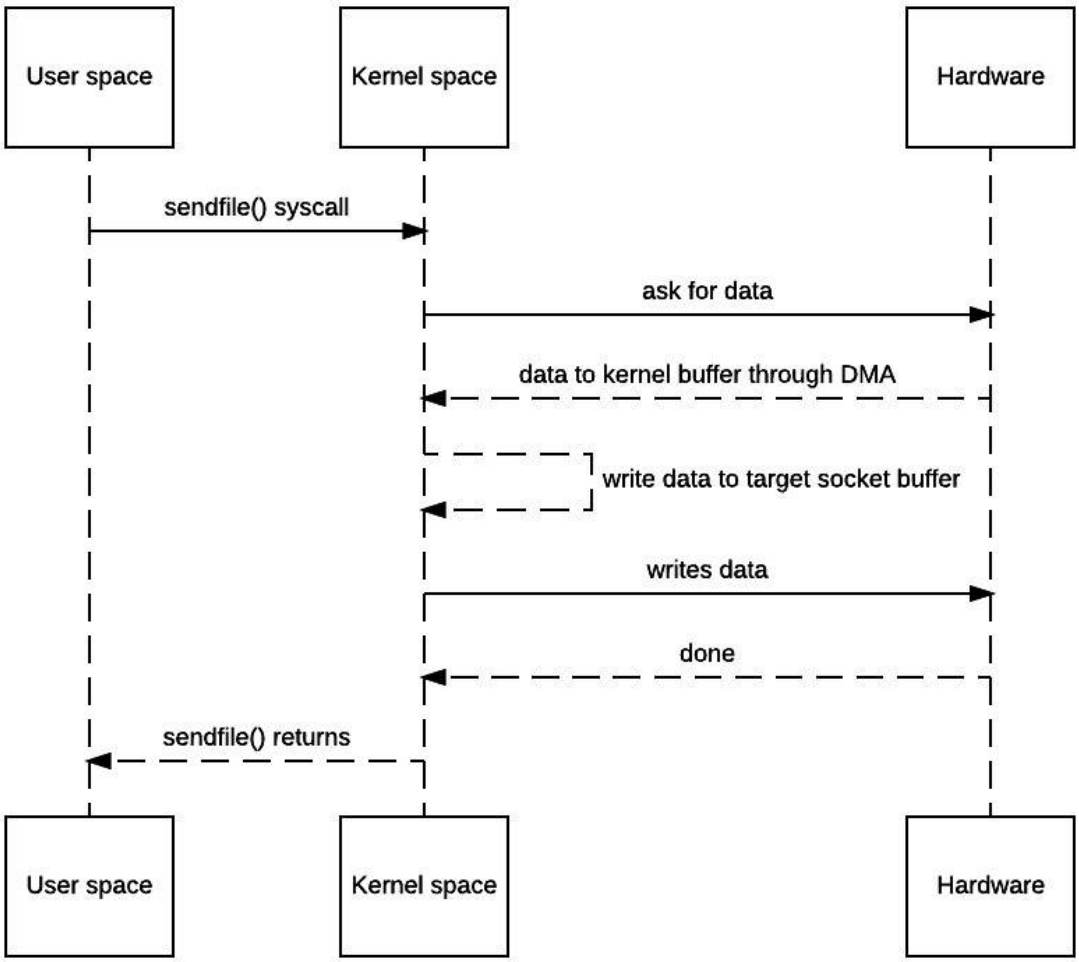




mmap

> 1. JVM OS sendfile()
> 2. ( ) (DMA)

> 3. Socket DMA ( ) , write() ,
>
> , Socket ,
> ,



Kafka 的运用

> Kafka , Page Cache , Page Cache.
> Kafka , IO.
> . Page Cache , ,

Kafka , page cache?

1. JVM , object overhead
2. JVM , GC , GC ,
3. ,