淘宝网 Taobao.com 淘我喜欢！

淘宝消息中间件技术演变

张乐伟（韩彰）

2013-7-14

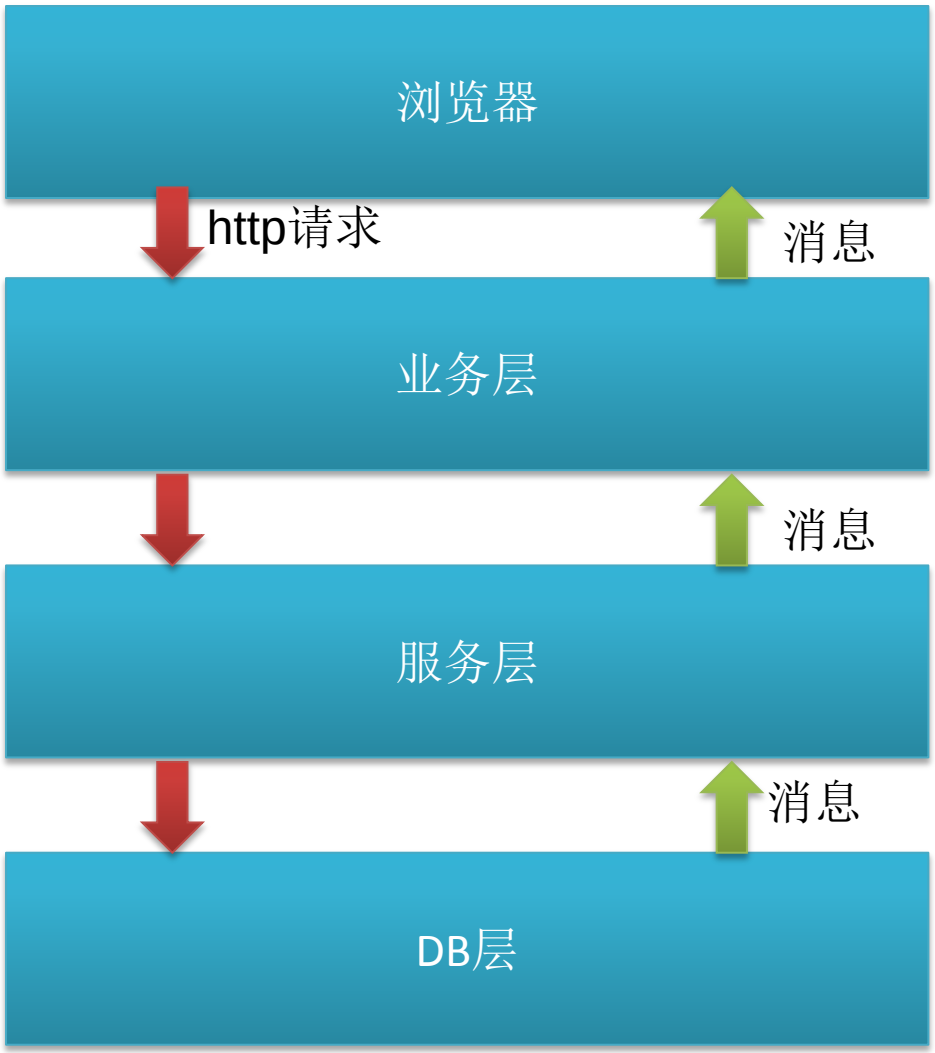
消息中间件技术演变

- 消息系统应用场景及现状
- 消息系统问题分析
- Metaq1.0的提出及存在的问题
- Metaq2.0的产生及相关功能介绍
- Metaq3.0 功能特点

消息系统应用场景

- 异步解耦
- 排队模型
- 流计算
- 流控型
- 重复消费
- 事务消息
- 顺序消息
- 定时投递
- 广播消息

消息系统应用场景

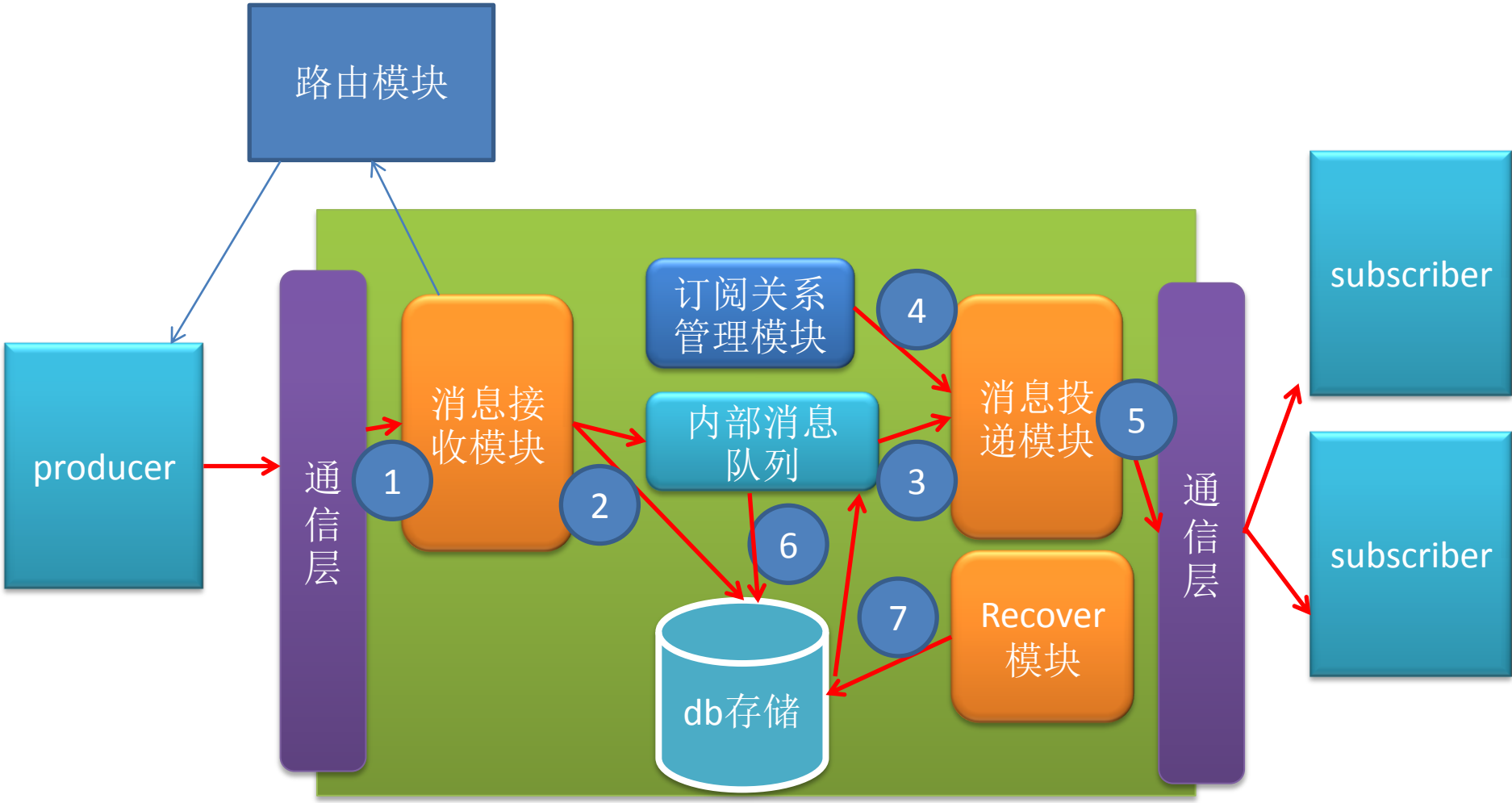


现状

- 接入系统500多个
- 每天消息量：30多亿接收，100多亿投递
- 12年双11，消息量5倍
- 13年双11？
- 堆积常发生
- 性能需要提升
- 接收投递比很多（1:15）

消息系统概述

- 基于pub-sub模型消息系统

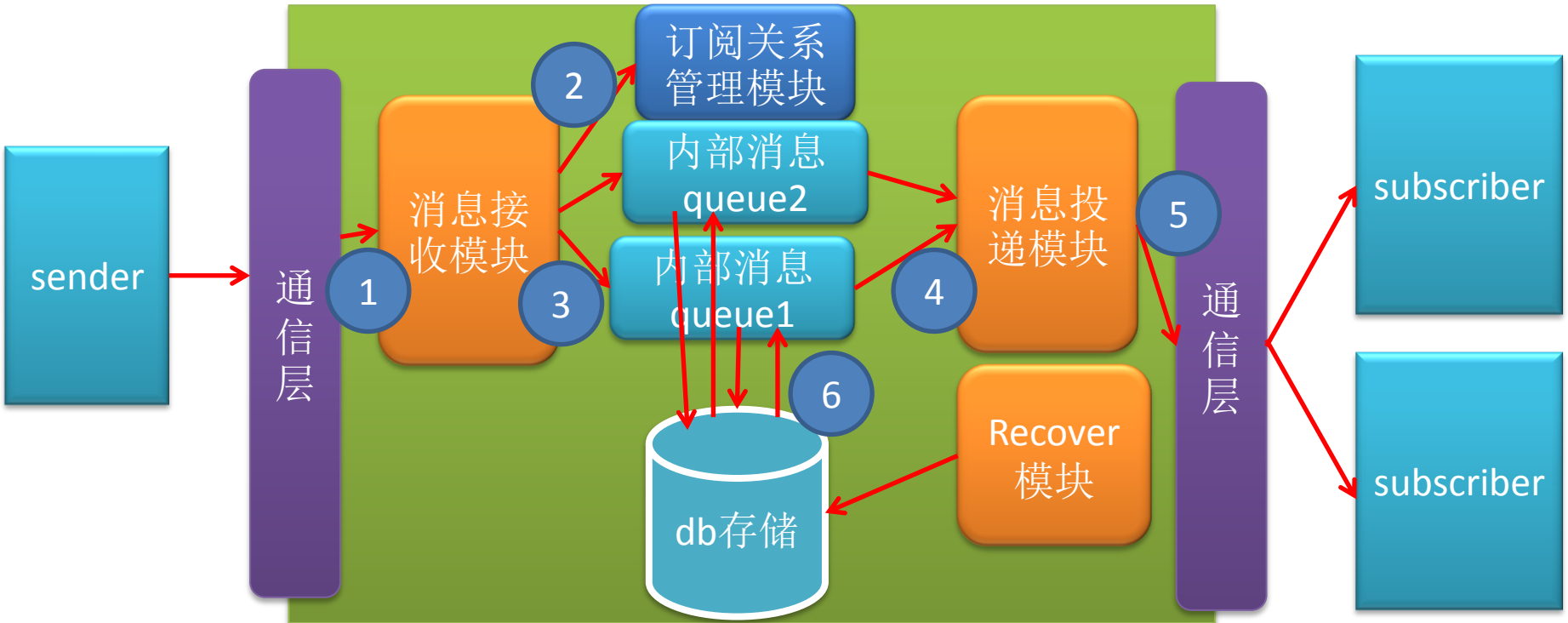


存在的问题

- 堆积
  - 堆积的产生：发送方（上游）与接收方（下游）能力不匹配，或者接收方（下游）异常
  - 堆积对自身的影响：堆积将导致存储数据积压，整个集群性能严重下降。
  - 堆积对业务方的影响：由于某个或者某几个接收方堆积，将影响其他接收方消息的实时性。
- 性能

堆积对业务方影响

- 本质
  - 基于topic的模型，非queue模型，即整个系统共享一个queue
- 基于queue模型消息系统



性能

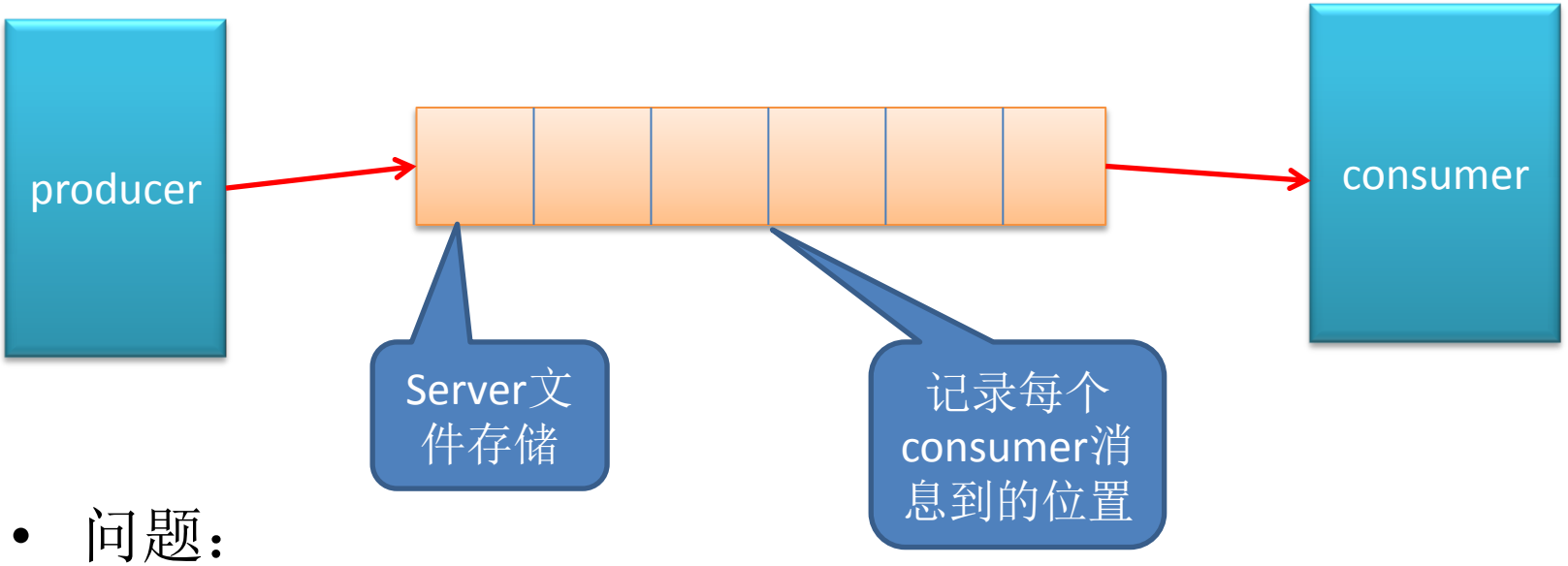
- 消息系统特点
  - 存储+推送（拉取）
  - 消息存储时间短
- 存储
  - Insert
  - Delete
  - Update
  - Select
  - 大部分情况下只是使用了数据库机器的内存
  - Mysql是否适合做消息系统的存储？

Innodb特点

- 存储模型：B+树
  - 随机写（分裂，合并）
  - 随机读
  - 适合于读多，写少
  - 特别适合于范围查找
- 存储类型
  - Undolog
  - Data
    - 数据
    - 索引
  - Redolog
    - Insert，delete等
  - Binlog(可以关闭）

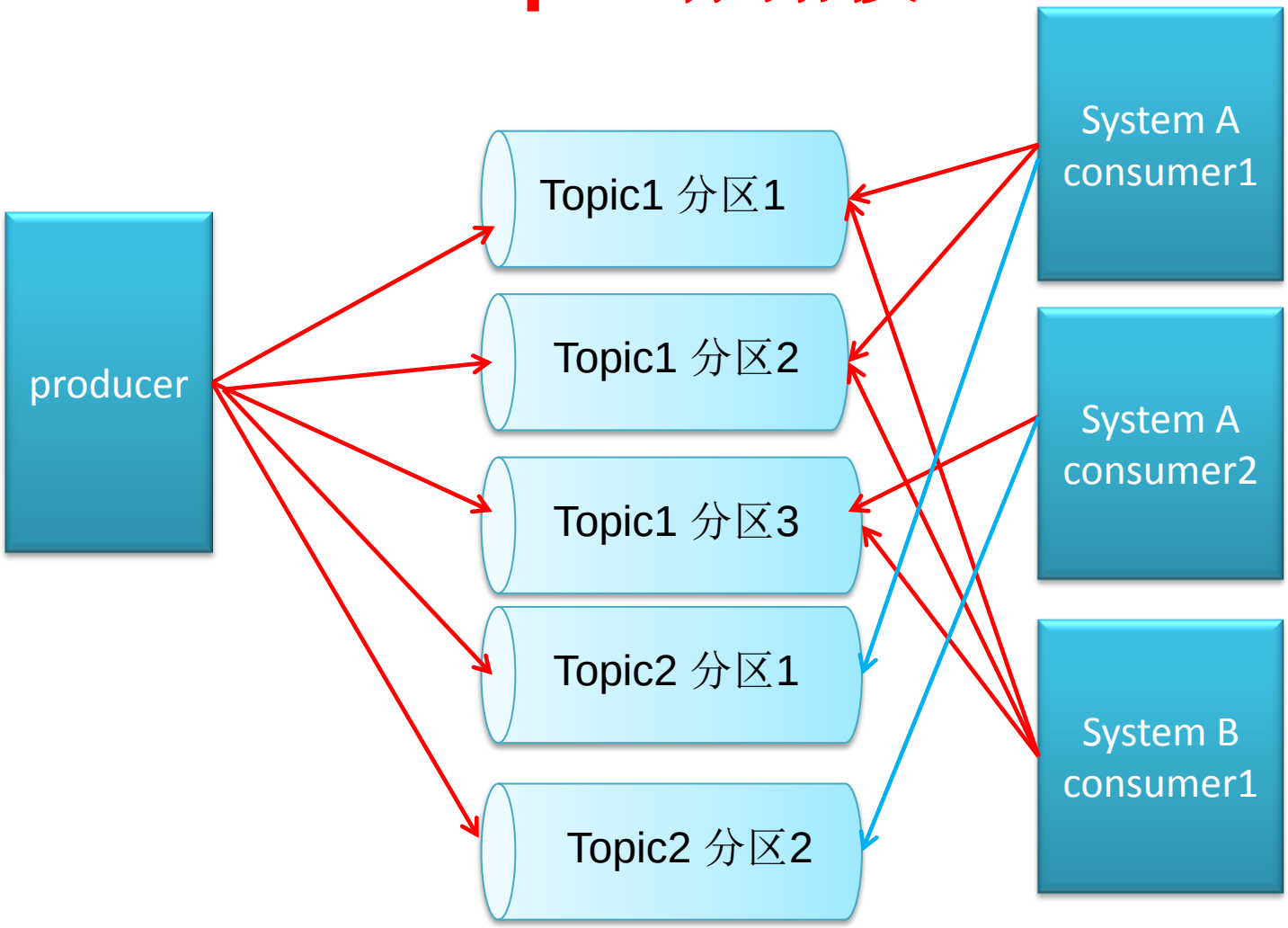
消息系统存储

- 顺序写？
- 顺序读？



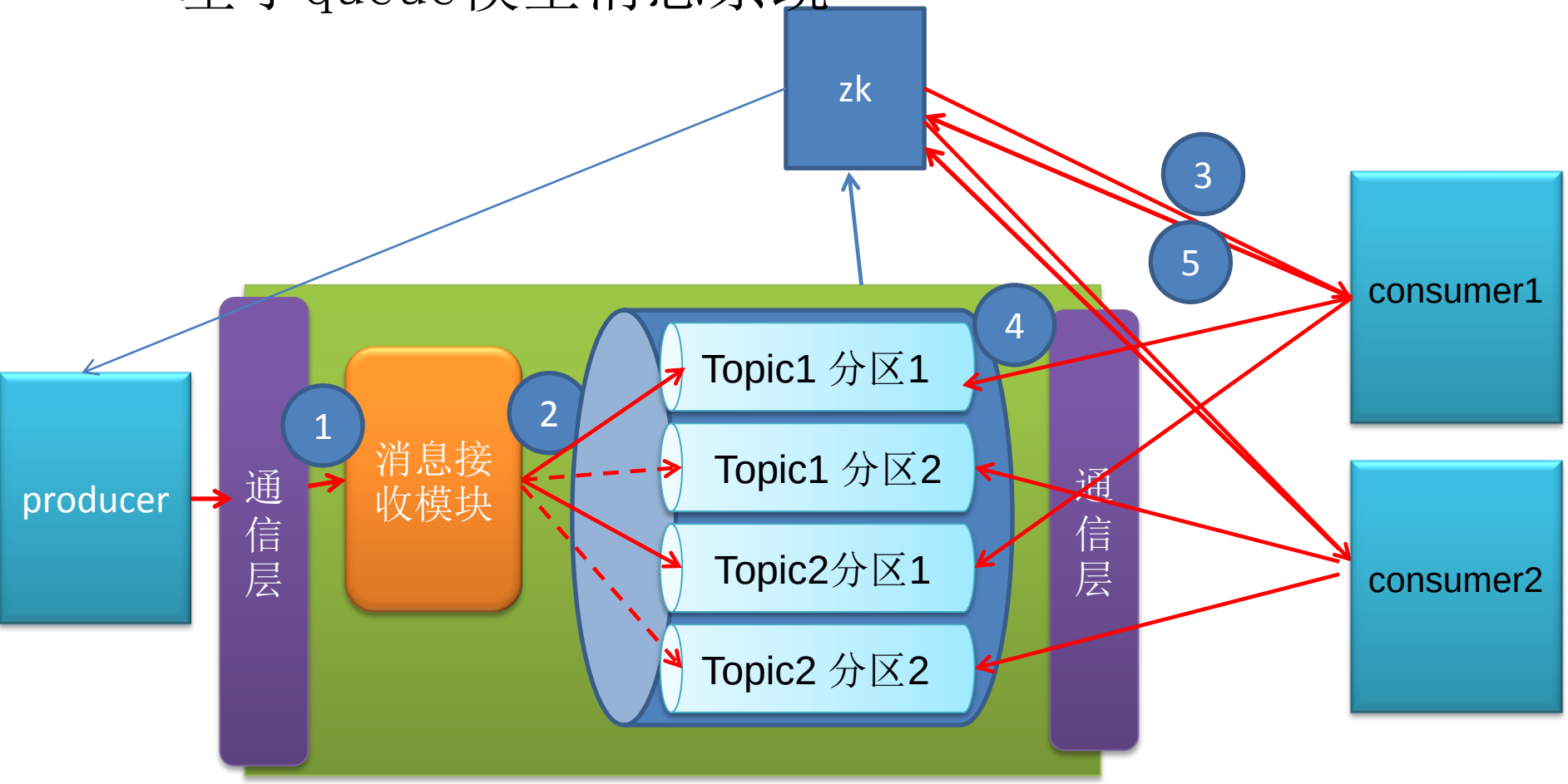
- 问题：
  - 多个topic，consumer group如何处理
  - 一对多消息如何处理

metaq1.0存储模型



metaq1.0整体结构

- 基于queue模型消息系统



推拉模型区别

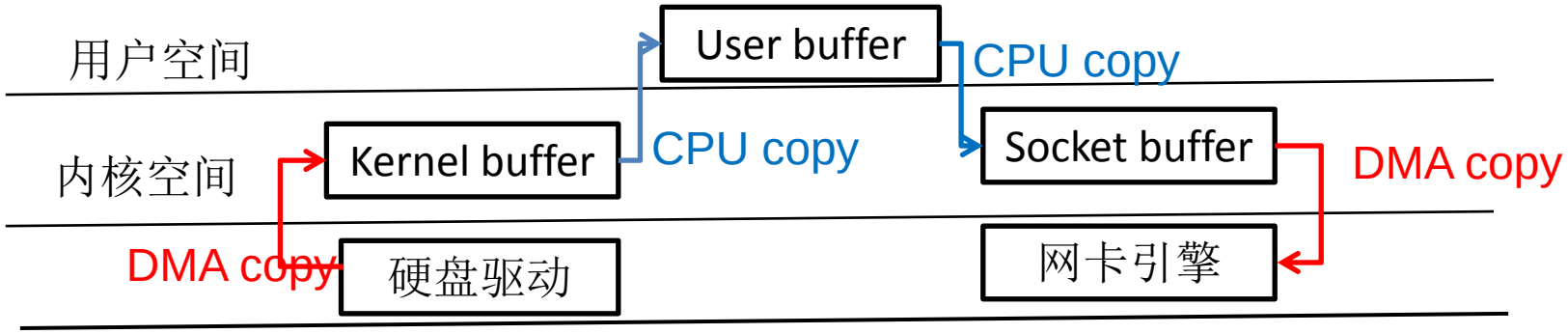
- 推模型优势
  - Consumer扩容方便，不受分区数限制
  - 支持复杂的过滤（因为订阅关系在server端）
  - 一个分区可以为多个consumer服务，分区数不会是瓶颈
- 拉模型优势
  - Consumer端控制方便，便于业务方根据自己需要去获取消息
  - 顺序消息支持方便
  - 允许长时间消费一条消息

Metaq 1.0特点

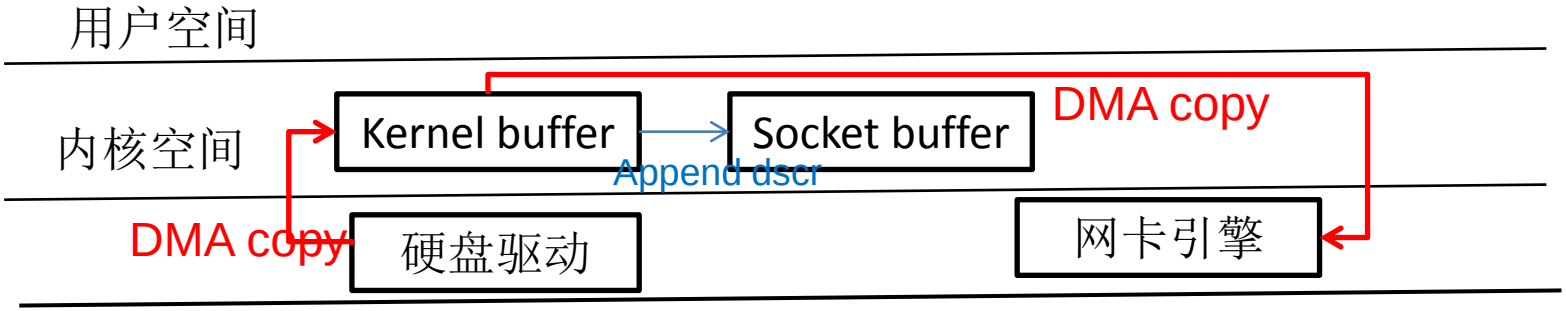
- 解决问题
  - 性能
  - 顺序消息
  - 不同consumer group互相影响问题
  - 堆积能力
  - 内存瓶颈：sendfile

zero-copy 技术

- Read, write

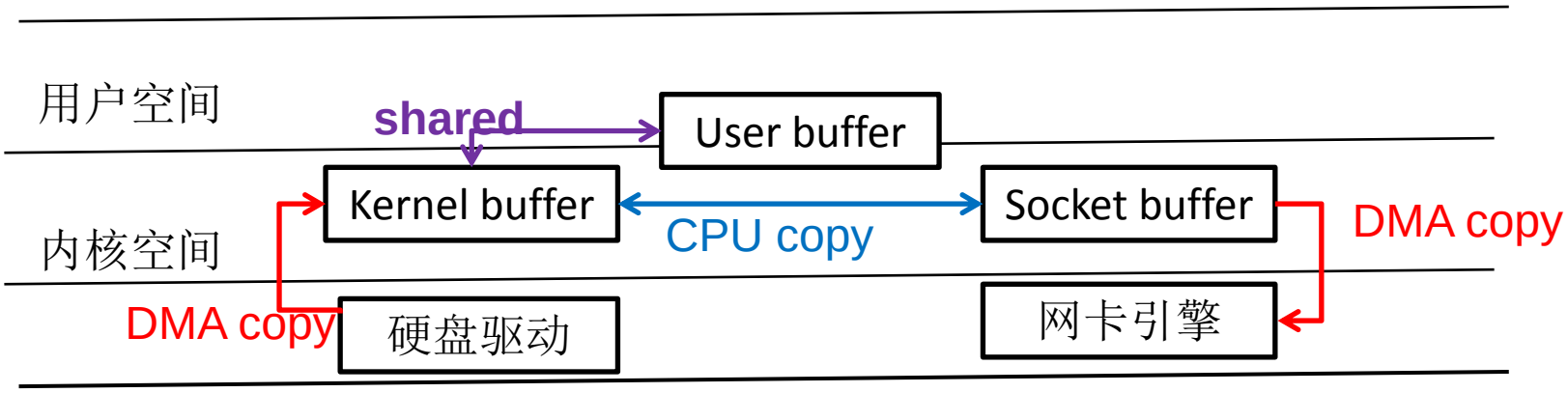


- Sendfile



zero-copy 技术

- Mmap, write



- 消息系统中使用优势
  - 减少上下文切换
  - 减少jvm内存使用（再也不怕投递比了）
- 缺点
  - 需要指定待传输消息在文件中的位置及长度等（即无法将消息反序列化到jvm中进行查找过滤）

metaq1.0存在的问题

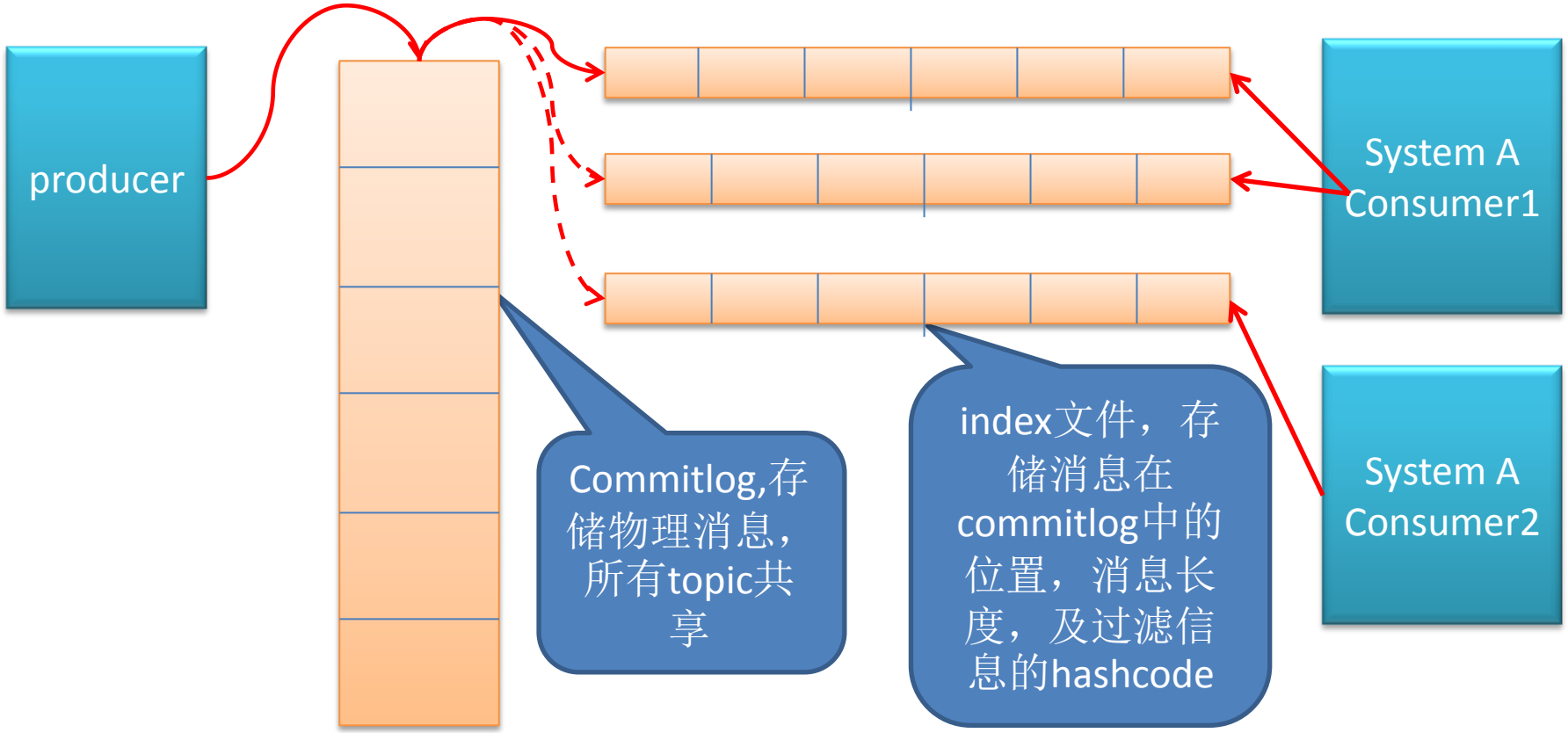
- 分区瓶颈
  - 分区数即可支持消费方集群的规模，与consumer机器数对应
  - 不同topic需要不同分区
- 性能
  - 分区数增加（文件数）：顺序写，顺序读----→随机写，随机读
  - 分区数超过30，性能直线下降
- Client复杂
  - 依赖zk，争抢分区，复杂，经常出各种问题，大量消费者场景无法支持，如forest，6000个消费者。
- 过滤
  - 针对某个topic下的消息过滤比较困难，因为利用了sendfile，无法反序列化到JVM中过滤

metaq1.0存在的问题

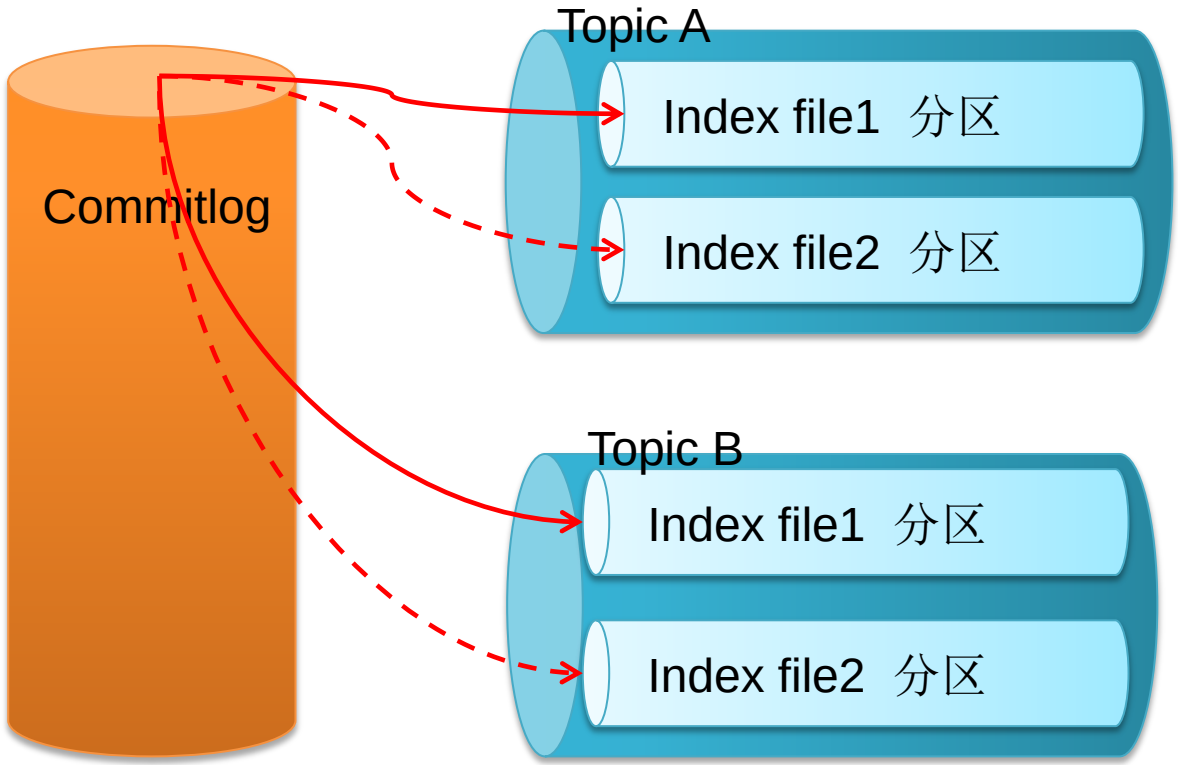
- 内存
  - 内存无法完全利用
- 无法获取落后（等待）的消息数
  - 由于消息大小不确定，无法获取落后消息数（某些场景下需要）

Metaq 2.0

- 存储模型



Metaq2.0 目录结构



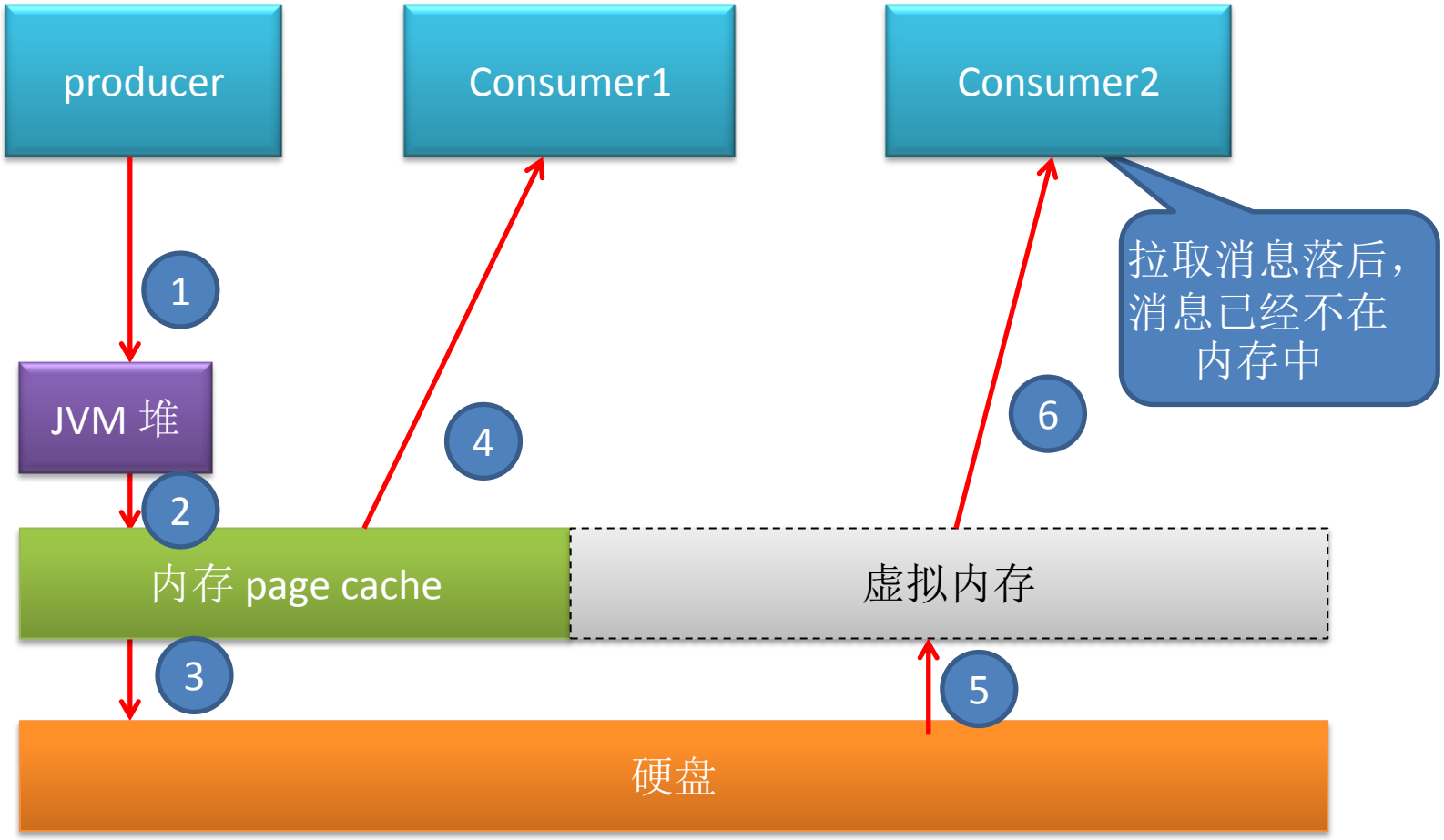
- 文件目录
  - Commitlog，所有topic一个commitlog文件，一个文件1G分割文件
  - Index文件，一个topic一个或多个index文件（分区数），不同topic不同index文件，且index文件数（分区数）大于等于consumer机器数

metaq2.0

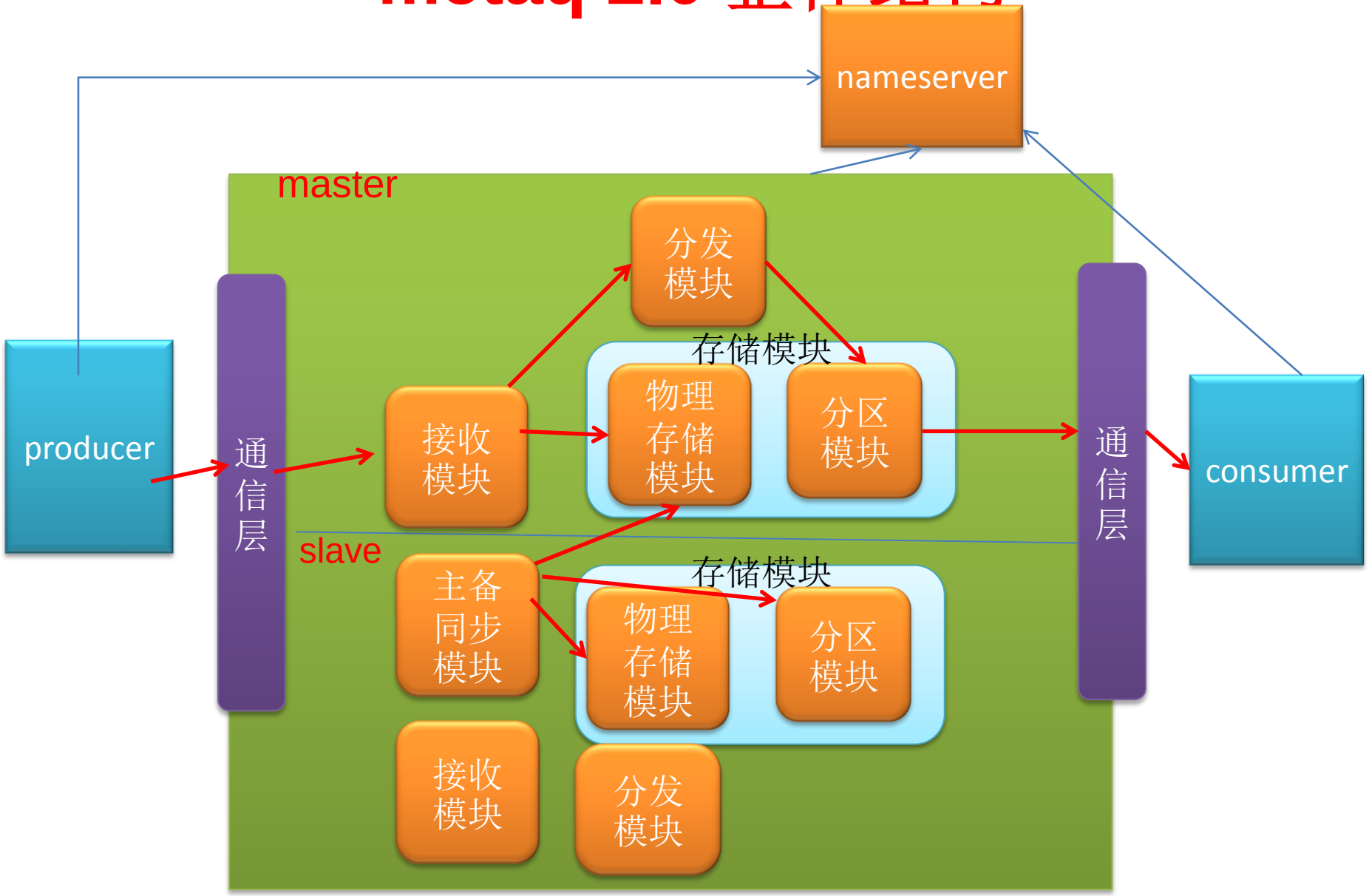
- 刷盘模式
  - Commitlog同步刷盘，groupcommit方式顺序刷盘
  - Index file 异步循环刷盘
- 内存模型
  - 利用mmap，映射到pagecache，充分利用内存
- 消息读取
  - 从index file中获取消息的位置，长度信息，然后从commitlog中读取，不需要再到JVM中
- 分区数
  - 单机可以支持上万分区数
- 消息过滤
  - 在index file中对过滤的消息比较hashcode，存在误差
- 获取落后（等待）的消息数
  - 由于index file每条消息定长，所以能计算消息数，及落后消息数
  - 可以支持消息回溯，用于消费已经消费过的消息

Metaq 2.0 数据流动

- 消息如何在JVM堆，内存，磁盘上流动



metaq 2.0 整体结构



metaq2.0功能特点

- 高性能
- 支持大量分区
- 支持消息过滤
- 支持消息堆积
  - 消息堆积下会有性能下降
- 天然具有限流功能
- 顺序消息支持
- Client简化
  - 分区根据一定的算法分配，同时考虑机房优先，不再依赖于zk的争抢

metaq2.0性能数据

以下表格场景基于这个前提：

1、CPU 16Core Intel(R) Xeon(R) CPU L5630 @ 2.13GHz

2、Memory 24G

3、Disk RAID SAS 15000 1.4T

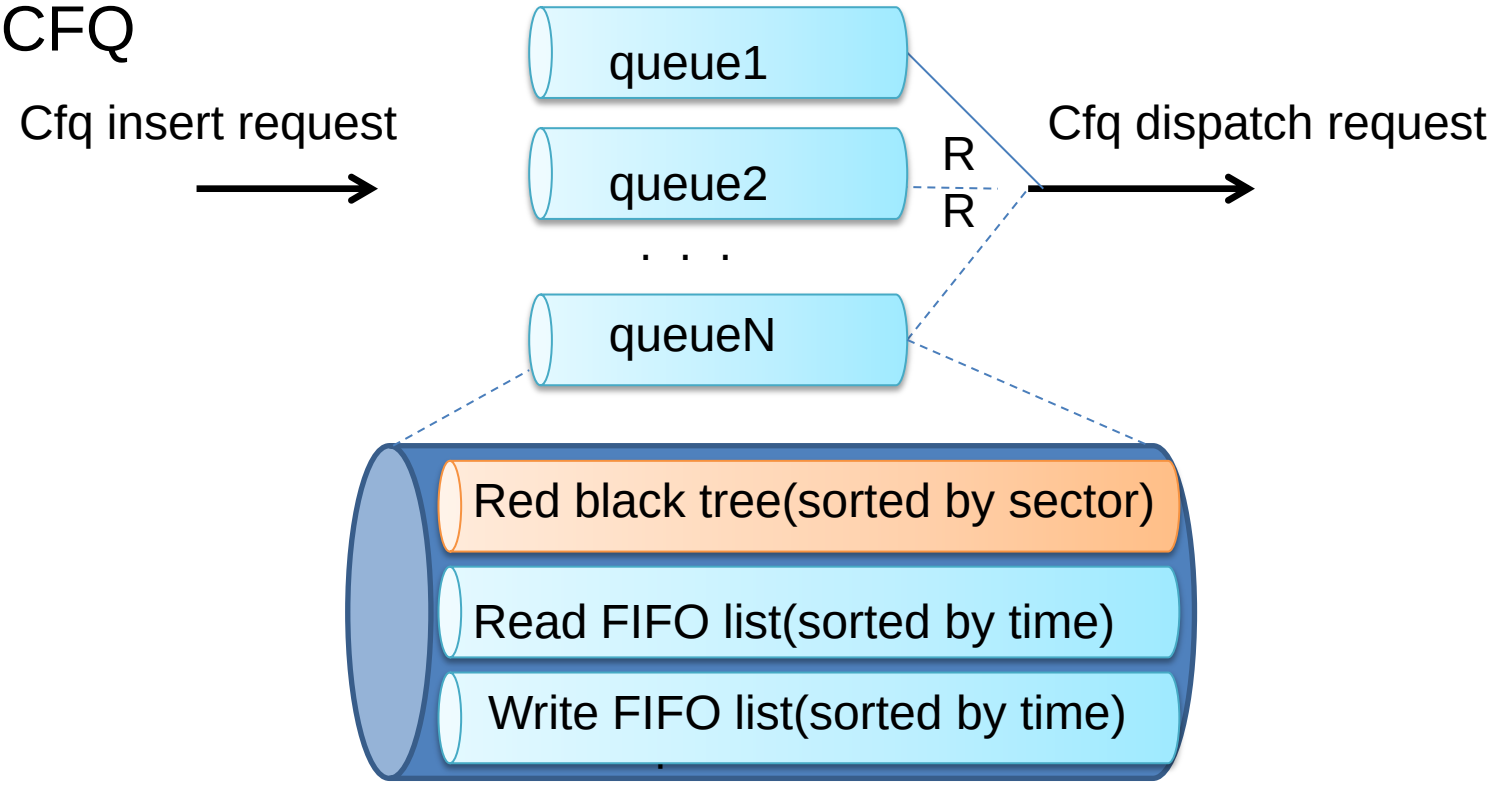
4、Linux 2.6.32 ext4

磁盘类型	刷盘策略	分区数	消息大小	发送耗时	发送消息TPS	订阅消息TPS	LOAD	IOWAIT	NETIN	NETOUT
RAID SAS 15000	异步	1024	128	4.1	5.3万	5.3万	2.7	0.83	14M	27M
RAID SAS 15000	异步	1024	256	5	4.6万	4.6万	3	1.23	19M	28M
RAID SAS 15000	异步	1024	2k	3.66	3.3万	3.3万	3.4	1.54	69M	75M
RAID SAS 15000	异步	1024	4k	5	2.9万	2.9万	4	2.39	117M	122M
RAID SAS 15000	异步	10240	512	5.3	3.6万	3.6万	3.3	1.14	18M	18M
RAID SAS 15000	同步	1024	128	2.9	3.9万	3.9万	2.9	1	11M	11M
RAID SAS 15000	同步	1024	256	3	3.8万	3.8万	3	1	16M	16M
RAID SAS 15000	同步	1024	2k	4.31	2.7万	2.7万	3	1.5	58M	58M
RAID SAS 15000	同步	1024	2k	4.31	2.7万	5.8万	3	1.5	58M	122M
RAID SAS 15000	同步	1024	4k	7.5	2.2万	2.2万	3	1.87	91M	94M
RAID SAS 15000	同步	1024	4k	7.5	2.2万	3.2万	3	1.87	91M	125M
RAID SAS 15000	同步	10240	512	7	2.8万	2.8万	3.8	3.4	16M	16M
RAID SAS 15000	**同步**	**1024**	**4k**	**65**	**0.1万**	**0.15万**	**7.7**	**57.4**	**5M**	**7M**

- 堆积压测
  - 先堆积500G数据，确保1024个分区全部有consumer拉取，consumer起1024个线程拉取，且全部拉到磁盘上

linux IO调度

- CFQ，noop，deadline，anticipatory
- CFQ



linux IO调度

- deadline

- deadline

RAID SAS 15000	同步	1024	4k	5.3	0.8万	1万	10	4	25M	45M

metaq3.0功能特点

- 支持事务
  - 最终一致，client需要check接口
- 消息实时性
  - 推拉结合
- 主备同步双写
  - 主备同时写，写完返回，主写存储，备写失败，用户自己选择。
- 主备自动切换
  - 主不可用，自动切换到备
- 定时延迟消息
- 单队列并行消费
- 消费失败延时重试

淘宝网 Taobao.com 淘我喜欢！

淘宝中间件期待各路英雄加盟！

hanzhang@taobao.com

微博：@淘宝韩彰