



华泰证券开源中间件应用部署及实践

杨军

A horizontal banner with a red left half and a grey right half. The red half contains the "HUATAI SECURITIES" logo in white. The grey half features a pattern of white-outlined diamonds.

HUATAI
SECURITIES

目录

华泰证券集团简介

Redis Cluster应用实践


Kafka 应用实践

Zookeeper应用实践

华泰证券



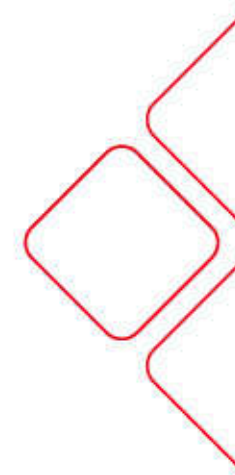
- 全国领先的大型综合性证券集团，具有庞大的客户基础、领先的互联网平台及敏捷协同的全业务链体系，股票代码601688
- 主营业务：经纪及财富管理、投资银行、投资及交易、资产管理、海外业务
- 经纪业务全市场第一（8.37%）

- 
- 用户数1400万
 - 日活用户峰值500万，手机端占比50%以上
 - 系统峰值交易金额3500亿/天，2015年全年交易额35万亿；“淘宝2015年双11交易额912亿”

分布式统一缓存选型考量



Redis Cluster



开源社区集群解决方案，支持分布式集群部署模式。通过节点与slot映射，方便集群动态扩展

01

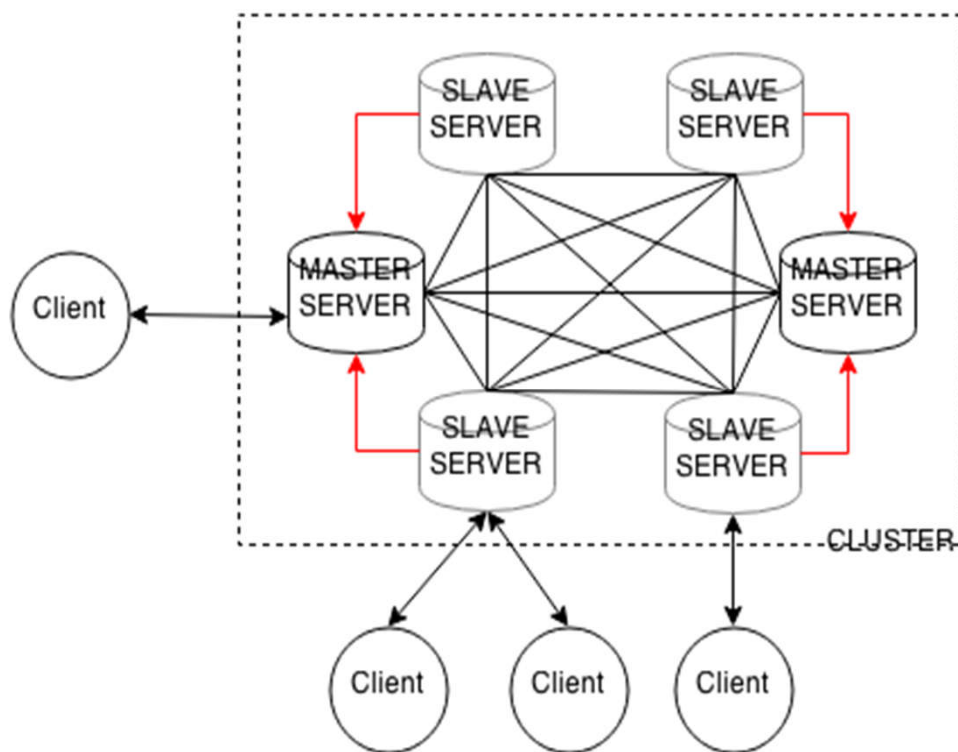
节点支持主备模式，故障自动发现与切换，保障系统高可用；支持数据持久化。

02

无中心架构，客户端与Redis节点直连，不需要proxy，性能更优

03

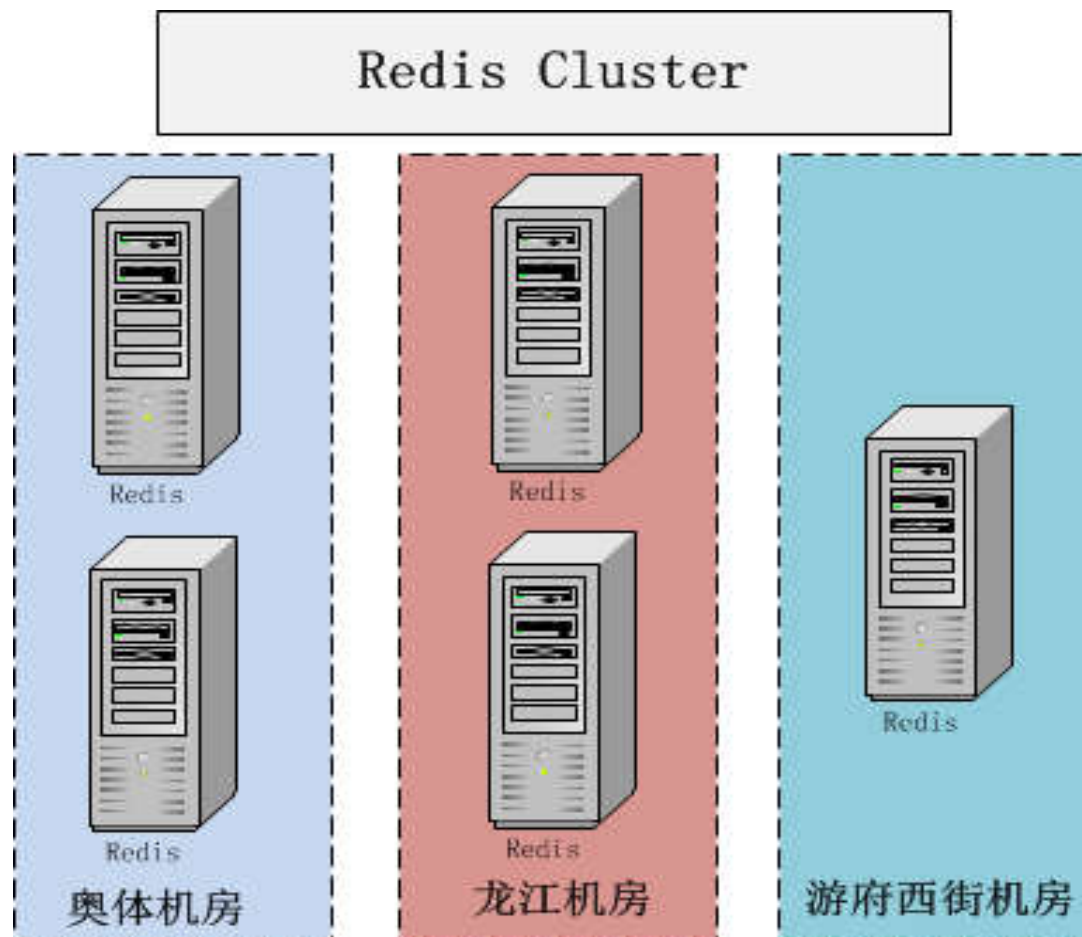
Redis Cluster架构



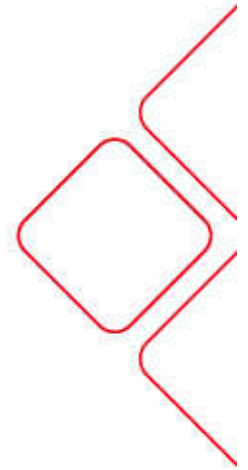
Redis Cluster特点:

- 去中心化: 节点间通过Gossip协议通信
- 客户端可以向任一节点发送请求, 再由其重定向到正确节点请求数据
- 集群通过slot预分配槽位; 增加节点不需要重启服务, 方便扩展

Redis Cluster部署



Redis Cluster部署



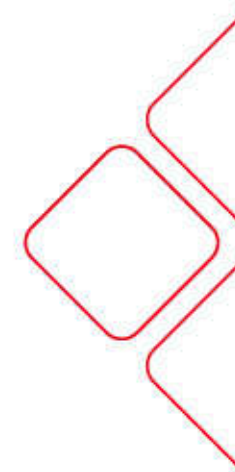
为什么要跨三机房部署？

- ◆ Redis Cluster节点fail是通过集群中超过半数的节点检测失效时才生效。单机房/两机房部署不能解决机房Crash时的高可用
- ◆ 没有主从复制的节点一旦故障，将导致整个集群不可用。跨机房部署通过配置节点M/S跨机房解决节点可用性

跨机房部署要求：

- ◆ 基础设施：网络高带宽、低延迟、稳定
 - 高带宽：主/从同步
 - 低延迟：解决客户端重定向查询的性能问题
 - 稳定：减少主/从节点切换。
(奥体/龙江间，网络延迟约0.3ms；与游府西街的网络延迟在0.8ms。游府西街部署一个slot，主要承担仲裁节点作用)

Redis Cluster应用实践



问题1：部分备节点异常crash，抛出如有图类似堆栈错误信息，且无法通过重启恢复。

定位：与jira #3343问题相似
<https://github.com/antirez/redis/issues/3343> 且Redis Cluster大版本(3.2.0)一致。list更新时索引未同步bug

解决： merge
<https://github.com/antirez/redis/commit/704196790ea9fd9ef4b556837baec417e1989c42d>
代码，暂时解决问题

----- FAST MEMORY TEST -----

```
27648:S 24 Jun 12:16:32.267 # Bio thread for job type #0 terminated
27648:S 24 Jun 12:16:32.267 # Bio thread for job type #1 terminated
*** Preparing to test memory region 722000 (114688 bytes)
*** Preparing to test memory region 1c75000 (135168 bytes)
*** Preparing to test memory region 7fd6dbe00000 (3072327680 bytes)
*** Preparing to test memory region 7fd7931ff000 (8388608 bytes)
*** Preparing to test memory region 7fd793a00000 (8388608 bytes)
*** Preparing to test memory region 7fd794200000 (2097152 bytes)
*** Preparing to test memory region 7fd79aa00000 (2097152 bytes)
*** Preparing to test memory region 7fd79b0ac000 (20480 bytes)
*** Preparing to test memory region 7fd79b2c9000 (16384 bytes)
*** Preparing to test memory region 7fd79b9c7000 (16384 bytes)
*** Preparing to test memory region 7fd79b9f1000 (4096 bytes)
*** Preparing to test memory region 7fd79b9f2000 (4096 bytes)
*** Preparing to test memory region 7fd79b9f5000 (4096 bytes)
```

.O.O.O.O.O.O.O.O.O.O.O

Fast memory test PASSED, however your memory can still be broken. Please run a memory test for several hours if possible.

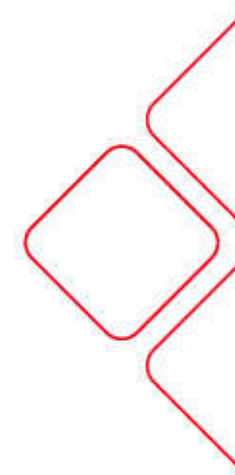
=== REDIS BUG REPORT END. Make sure to include from START to END. ===

Please report the crash by opening an issue on github:

<http://github.com/antirez/redis/issues>

Suspect RAM error? Use redis-server --test-memory to verify it.

Redis Cluster应用实践(续)



问题2：节点Master/Slave切换频繁

定位：通过监控，发现M/S切换时间点与节点产生dump文件时间吻合

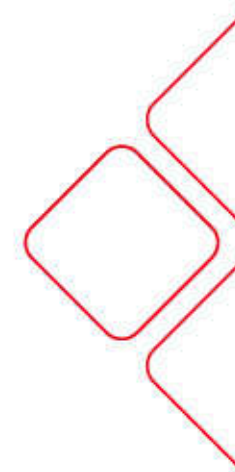
解决：

- 关闭所有主节点dump，开启AOF持久化；
- 降低备节点dump频率

```
save 900 1  
save 300 10  
save 60 100000
```

- 加强监控，且部署takeover脚本；主备发生切换时，通过脚本refine切回，确保主节点运行在关闭了dump节点上

Redis Cluster应用实践(续)



问题3：应用频繁报JedisClusterMaxRedirectionsException异常

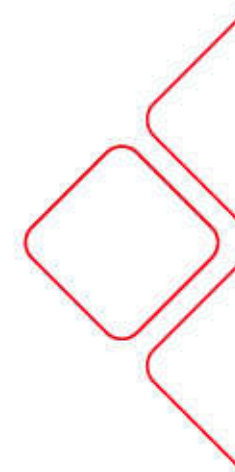
定位：

- 抛出异常的任务多为时间复杂度为 $O(N)$ 操作，如smembers...且集合成员较多(>10k)
- JedisCluster客户端三次任务执行command超时或失败抛出

解决：

- 减少 $O(N)$ 复杂度的操作，如必须使用业务在缓存时做数据拆分（如股票按市场拆分）
- 禁止使用keys
- 修改command重试参数redis.maxRedirections

Redis Cluster应用实践(续)



问题4: JedisCluster客户端pub/sub接口调用时，集群节点流量异常；正常节点流量<10MB，开启pub/sub模式，节点流量飙升至60MB以上

定位: 客户端接口暂不支持。业务暂时规避pub/sub接口调用

```
/* -----  
 * CLUSTER Pub/Sub support  
 *  
 * For now we do very little, just propagating PUBLISH messages across the whole  
 * cluster. In the future we'll try to get smarter and avoiding propagating those  
 * messages to hosts without receives for a given channel.  
 * ----- */  
void clusterPropagatePublish(robj *channel, robj *message) {  
    clusterSendPublish(NULL, channel, message);  
}
```

其他使用注意事项



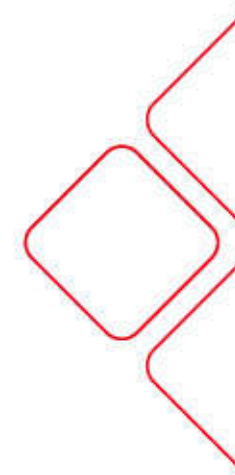
Redis Cluster:

- 多keys不能跨节点操作;
- 不支持Multi/Exec;
- Redis主备非强一致, 切换存在丢数风险;
- value大小对性能影响很大; 生产value大小建议不超过1k。

Jedis Cluster:

- 创建JedisCluster连接时, 连接池配置关闭testOnBorrow和testOnReturn, 只需开启testWhileIdle。

多租户



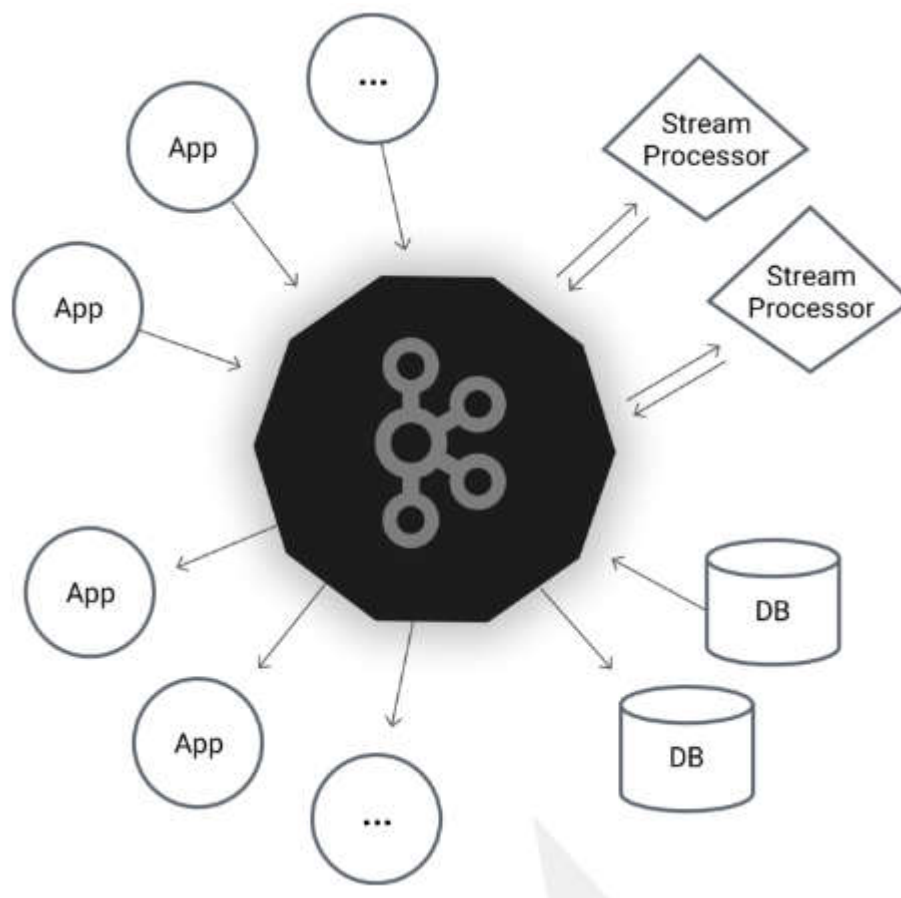
VS



运维管理
资源隔离
.....



分布式消息中间件选型



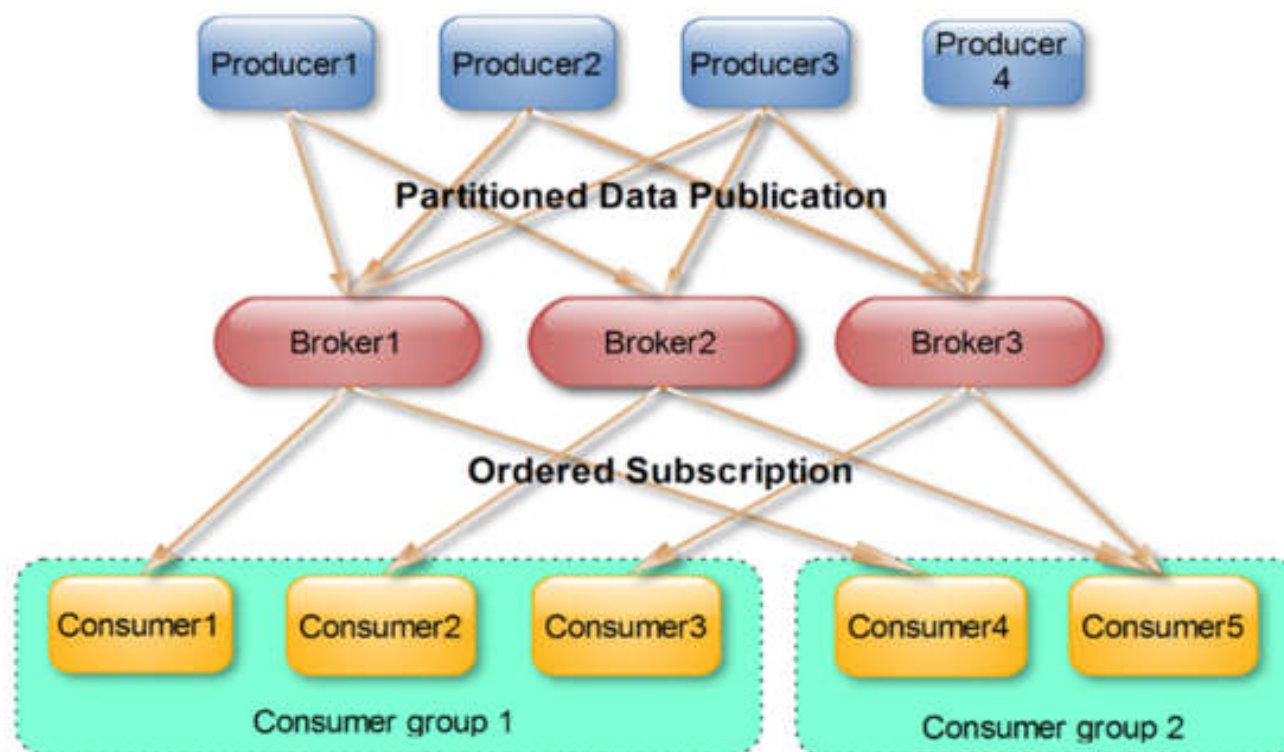
分布式消息中间件选型



互联网公司消息中间件解决方案的事实标准

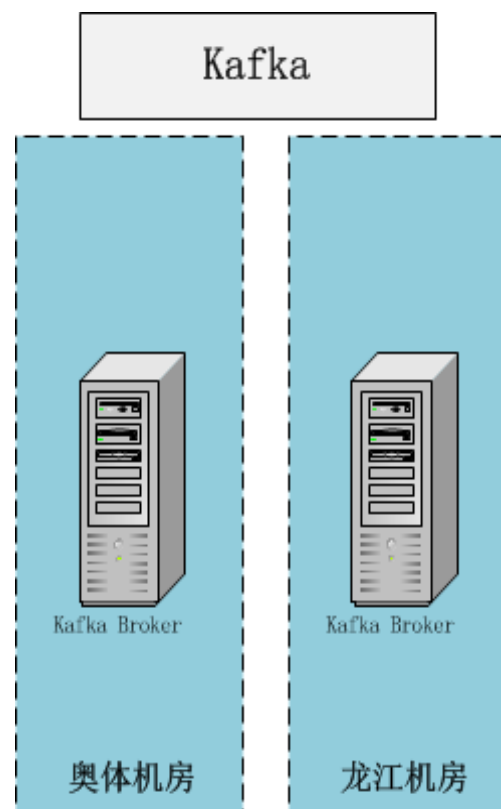
- 高吞吐
- 支持消息高堆积
- 快速持久化
- 完全的分布式系统，通过Zookeeper自动实现复杂负载均衡
- 支持消息复制，数据更安全
- 组件无状态，支持fast Recovery
- 同时支持广播和pub/sub模式
- 轻量（对比RabbitMQ、ActiveMQ）

Kafka系统拓扑

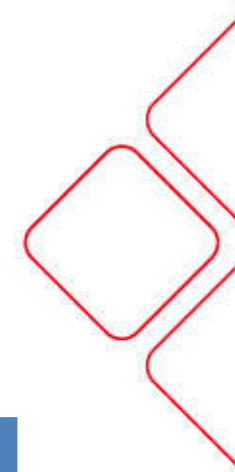


Kafka集群部署

- Kafka集群通过Zookeeper自动实现复杂负载均衡
- 两机房部署可保证集群跨机房双活



Kafka应用实践



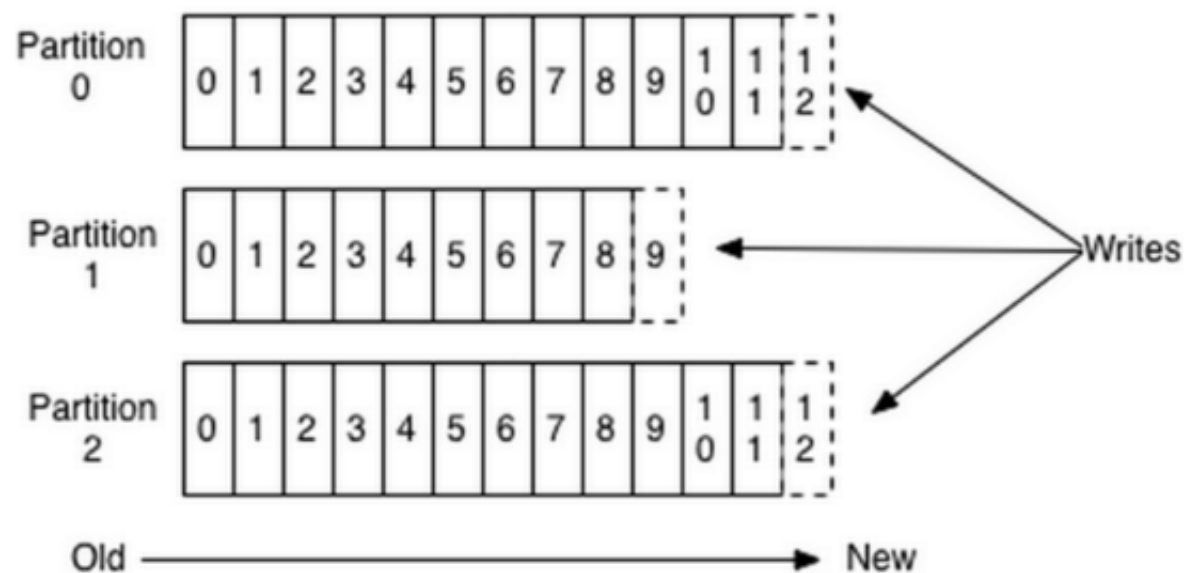
经验1：大量**topic/partition**会降低**kafka**吞吐性能

Topic数量	发送端并发数	发送端RT/ms	发送端TPS	消费端TPS
64	800	5	13.6w	13.6w
128	256	23	8500	8500
256	256	133	2215	2352

Kafka每个分区对应一个物理文件，**topic**增加，消息落地会加剧磁盘IO竞争，进而影响系统吞吐！

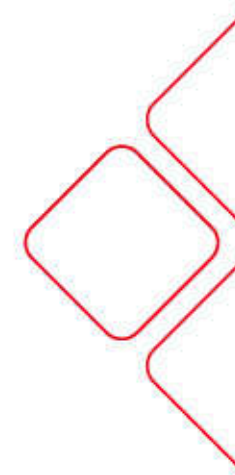
Kafka应用实践(续)

经验2：合理设置消息**key**值，保证消息处理有序



Topic在逻辑上可以被认为是一个queue；每个topic有1个或多个分区。同一topic内数据消费不保证有序，但分区内消息有序被消费。

Kafka应用实践(续)



经验3：如何保证消息可靠

发送端：kafka支持两种传输可靠性机制：

- At most once: 消息可能会丢，但绝不会重复传输；
- At least once: 消息绝不会丢，但可能会重复传输。

Kafka发送不支持幂等，如果需要消息发送且仅发送一次，就需要从业务逻辑层上保证。系统默认At least Once (`request.required.acks = 1`)；可通过设置`request.required.acks = 0`配置消息发送变成At most Once

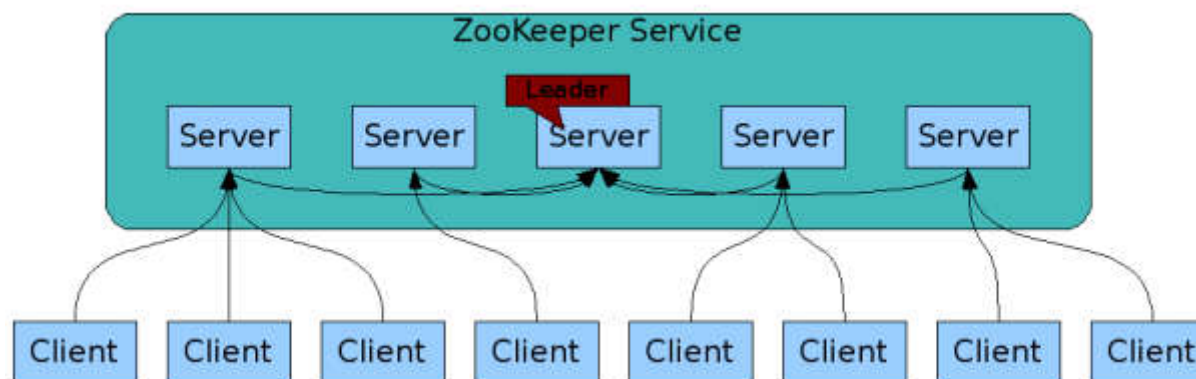
!!!Important: 重要数据配置`request.required.acks = all`

消费端：通过High/Low Level API可以保证消息两个层次的可靠性：At least once和Exactly once

Broker:

- ISR & WAL
- 对于内存数据，通过配置flush配置参数`log.flush.interval.messages`和`log.flush.interval.ms`降低broker crash丢数据风险。

分布式应用协调服务

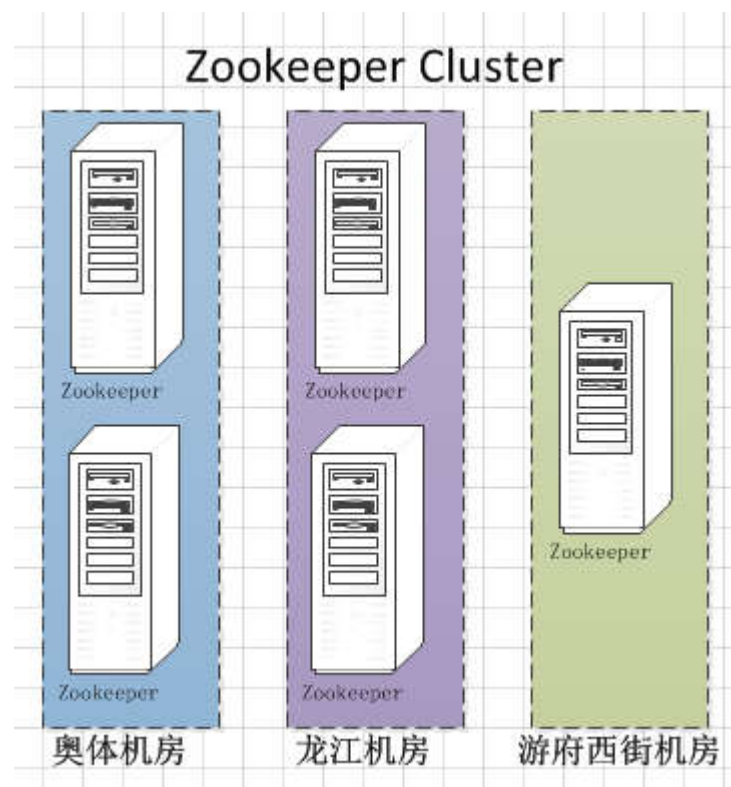


Zookeeper特性

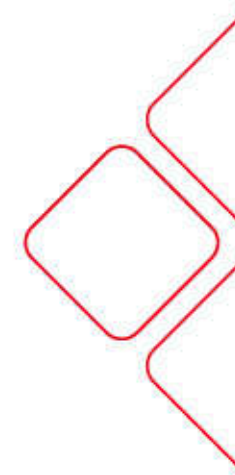
- 顺序一致性：同一客户端发起任务，严格有序
- 操作原子性：集群所有节点事务一致
- 可靠：稳定
- 实时性：通过顺序一致性和数据最终一致性提供准实时服务
- 单一视图：连接任意一台server，查询结果统一

Zookeeper集群部署

- zookeeper通过Zab(Paxos)协议通信，保证集群间数据一致性
- Zookeeper是分布式应用的核心组件，三机房部署进一步提升集群可用性
- 扩机房部署对网络基础设施要求较高



Zookeeper集群部署(续)



三机房部署节点计算:

➤ 复杂

假定机器总数是N, 三个机房分别部署N1, N2, N3个实例节点

✓ $N1 = (N-1)/2$

✓ $N2 = \lfloor (N-N1)/2 \rfloor$

✓ $N3 = N - N1 - N2$

➤ 简单

根据网络情况选择网络最差的, 部署一个实例, 其他两个机房分别部署 $(N-1)/2$

异地多机房部署, 可以考虑使用observe模式。

Zookeeper应用实践



经验1: Znode watch网络异常断开重连后，需要重新添加watch

Znode watch是一次性监听行为；

- 处理程序如需持续监听znode变更，变更处理后重新添加watch
- 网络中断，捕获异常，重新连接后添加znode watch

经验2: Zookeeper不要存储大量数据

Zookeeper设计初衷是协调服务

- 单个Znode数据大小不超过1M
- 大量数据存储涉及数据落地持久化以及集群间内部传播，影响Zookeeper性能。

开源中间件在华泰的业务应用



移动CRM

涨乐财付通

行情中心





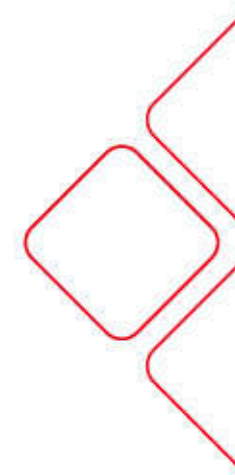


统一配置
中心
DisConf

Dubbo微
服务框架

Kafka/Sto
rm集群管
理

分布式锁



谢 谢 ！