

大规模redis集群的服务治理之路

高嵩

About me

- 11年工作经验，2011年加入优酷，曾任高级工程师、技术专家，现任高级技术经理
- 目前在大优酷数据战略团队主要负责分布式缓存、实时计算平台的搭建与优化
- 对分布式存储、流计算、高并发高可用系统有浓厚兴趣，热爱分享与交流
- 技术博客：<http://blueswind8306.iteye.com/>



高嵩

北京 昌平



扫一扫上面的二维码图案，加我微信

目录

➤ Redis Cluster介绍

- ✓ Redis Cluster特性&集群架构
- ✓ 服务端分片
- ✓ 客户端请求&请求重定向
- ✓ Failover

➤ 运维经验


- ✓ 集群扩容
- ✓ 手动failover
- ✓ 集群迁移



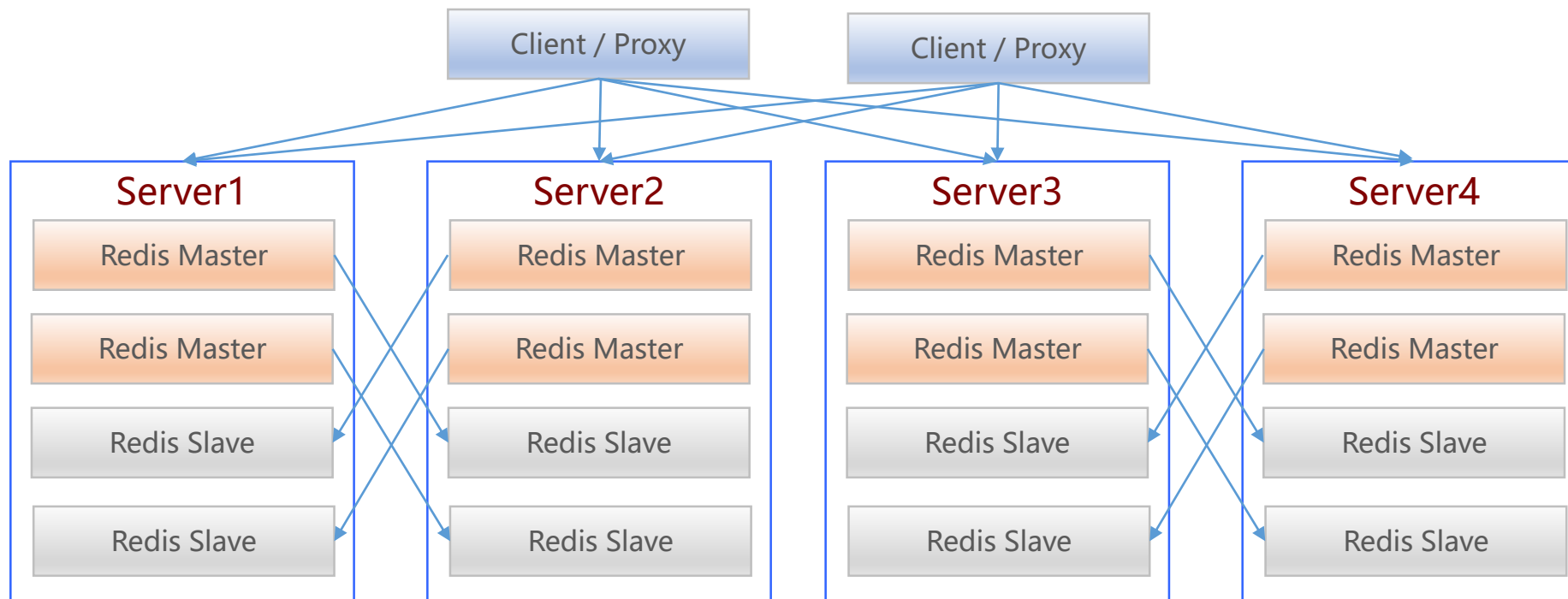
Redis Cluster介绍



■ Redis Cluster特性

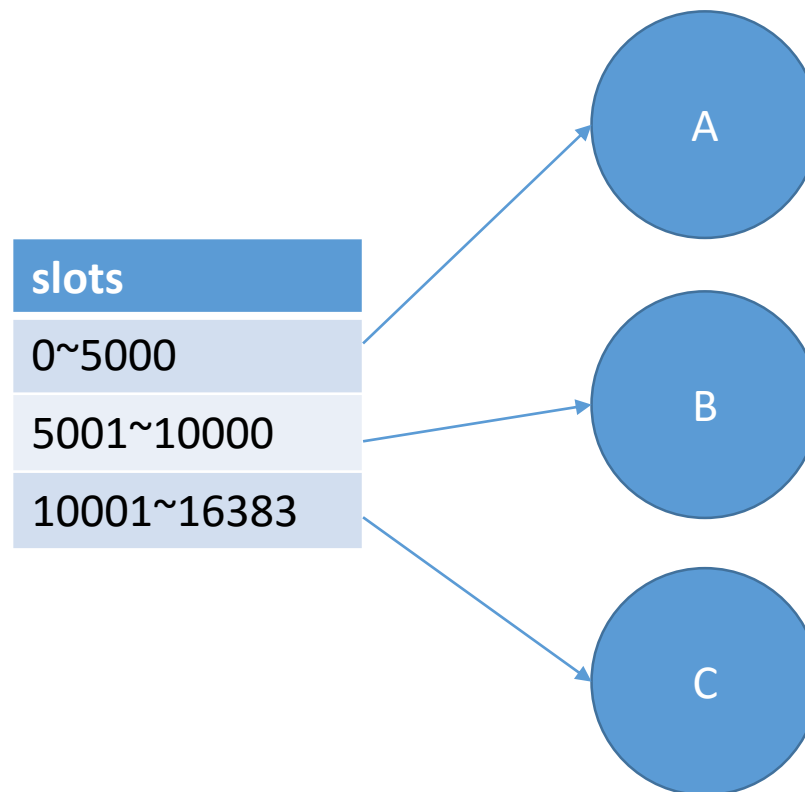
- 支持string/hash/list/set/sortedset/hyperloglog/geo/pubsub等**大部分**单机Redis功能
 - 去中心化的分布式集群
 - 主从全量/增量同步
 - 服务端分片
 - 节点水平伸缩，扩容/缩容对调用方透明
 - 自动failover/failback
- 

Redis Cluster 集群架构



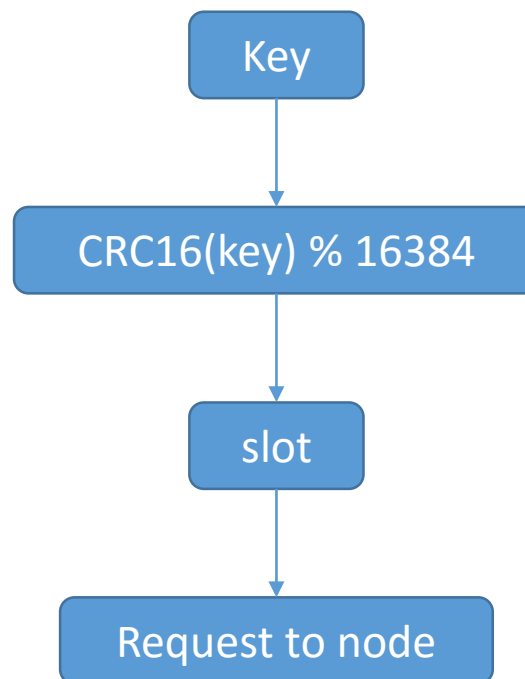
■ 服务端分片

- 数据分为固定的16384个槽 (slot)
- 每个node负责一部分slot的数据存储
- 每个node有整个集群的slot->node映射关系
- 集群初始化时确定slot->node的关系
- 扩容/缩容通过slot迁移完成



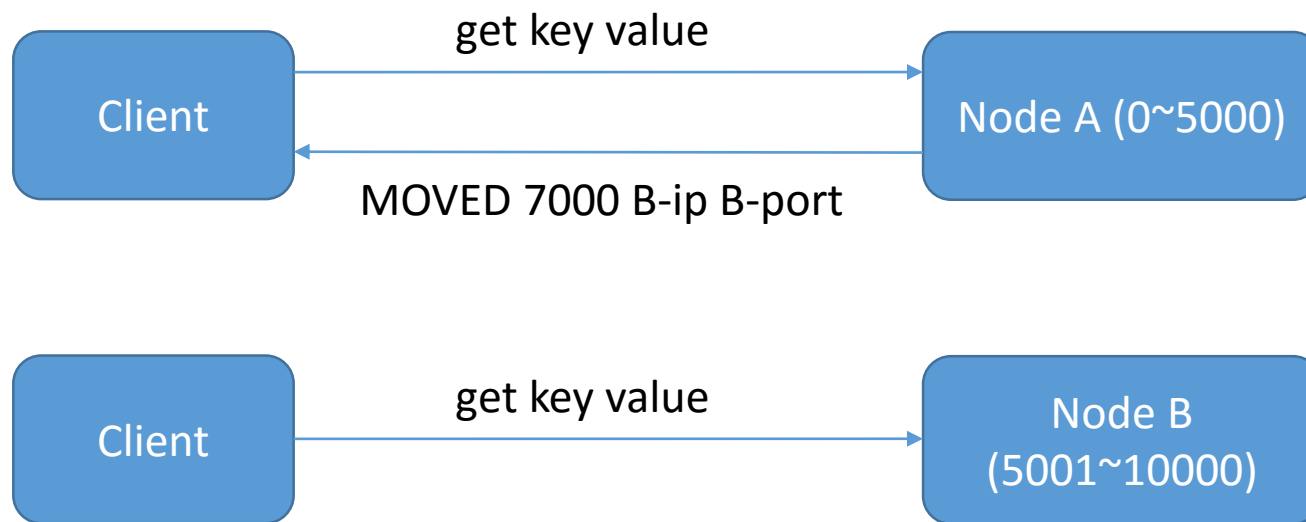
客户端请求

- 客户端缓存slot->node的映射关系
- 请求一个key时，在本地先算出key对应的slot，再根据slot->node的对应关系找到node
- 如果服务端的映射关系和客户端不一致怎么办？



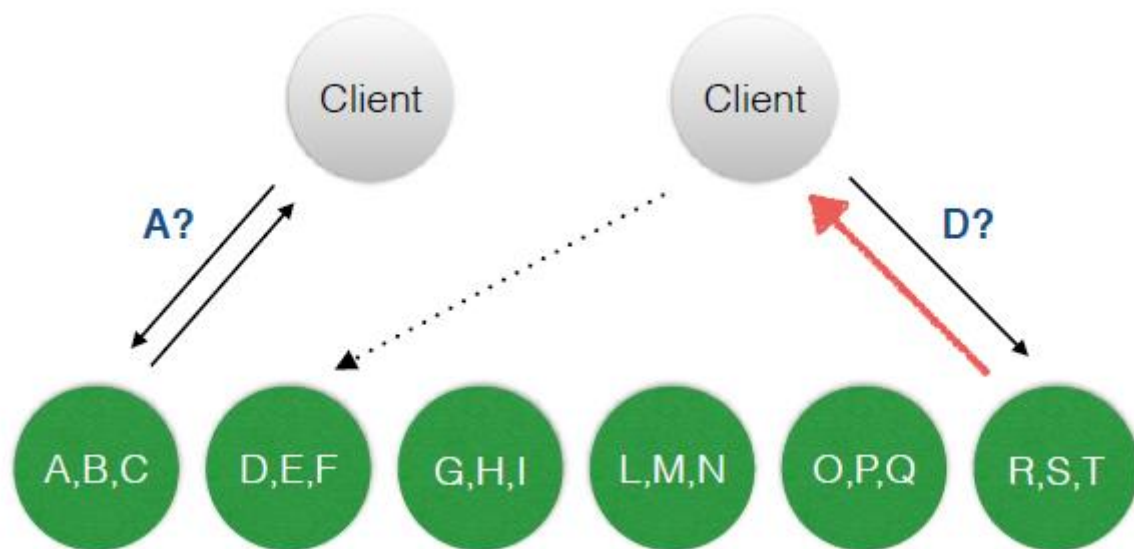
请求重定向 – MOVED

- 客户端请求的key所对应的槽不在Node A
- Node A会返回一个长期重定向错误 (MOVED)
- 客户端根据MOVED重定向信息，访问Node B
- 客户端重新缓存slot->node的映射关系



请求重定向

- 解决了客户端与服务端的一致性问题
- 集群状态变化（扩容/缩容）对客户端请求不会造成影响
- 支持多次跳转



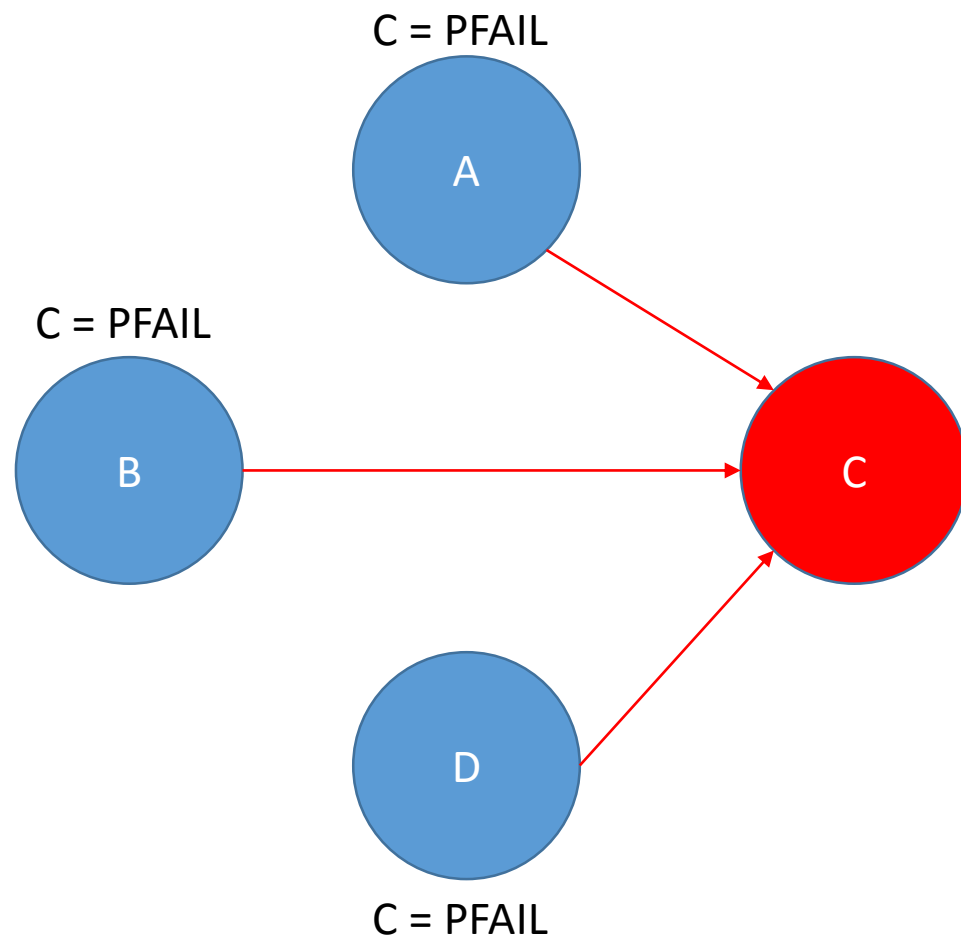
Failover

➤ Fail判定

➤ Leader选举

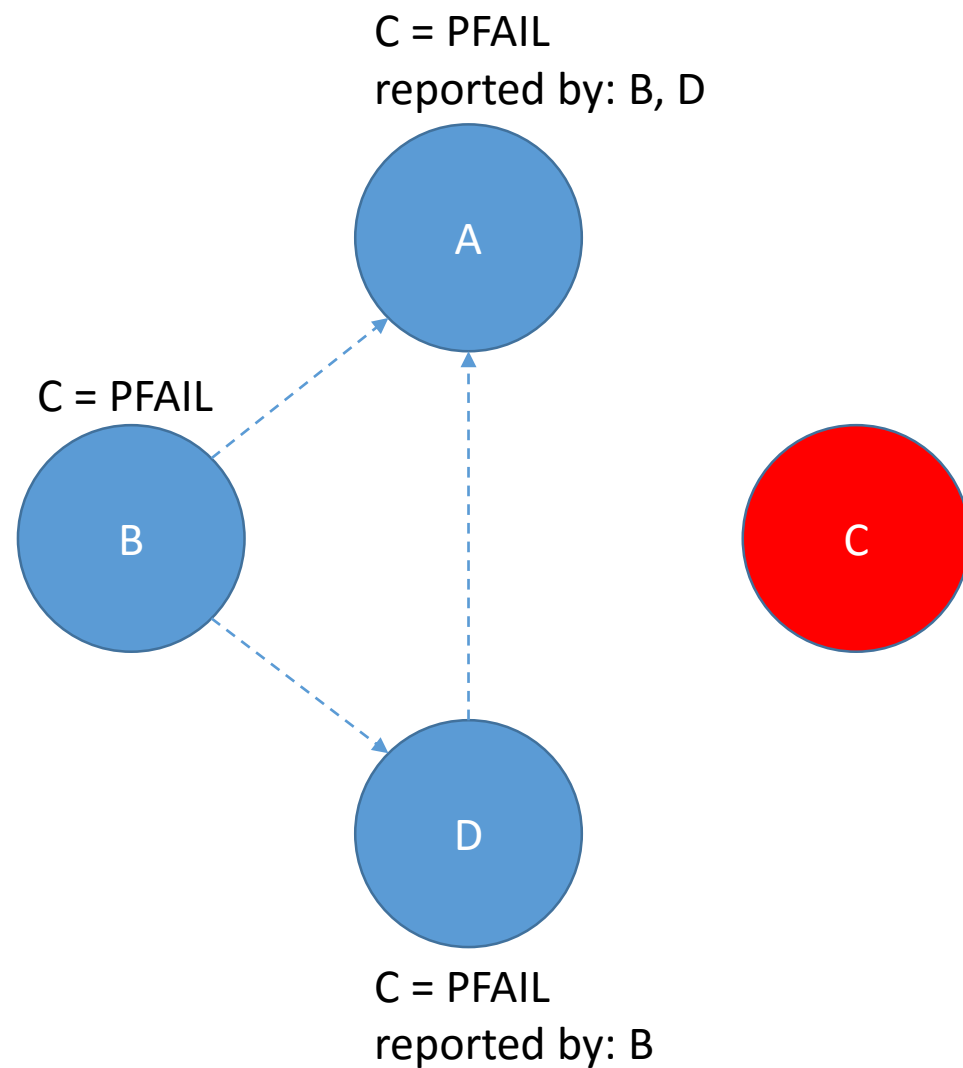
Fail判定

- 集群通过心跳检测节点是否下线
- 下线状态分为PFAIL (possible failure) 和FAIL
- A、B、D节点都发现C节点下线 (PFAIL)



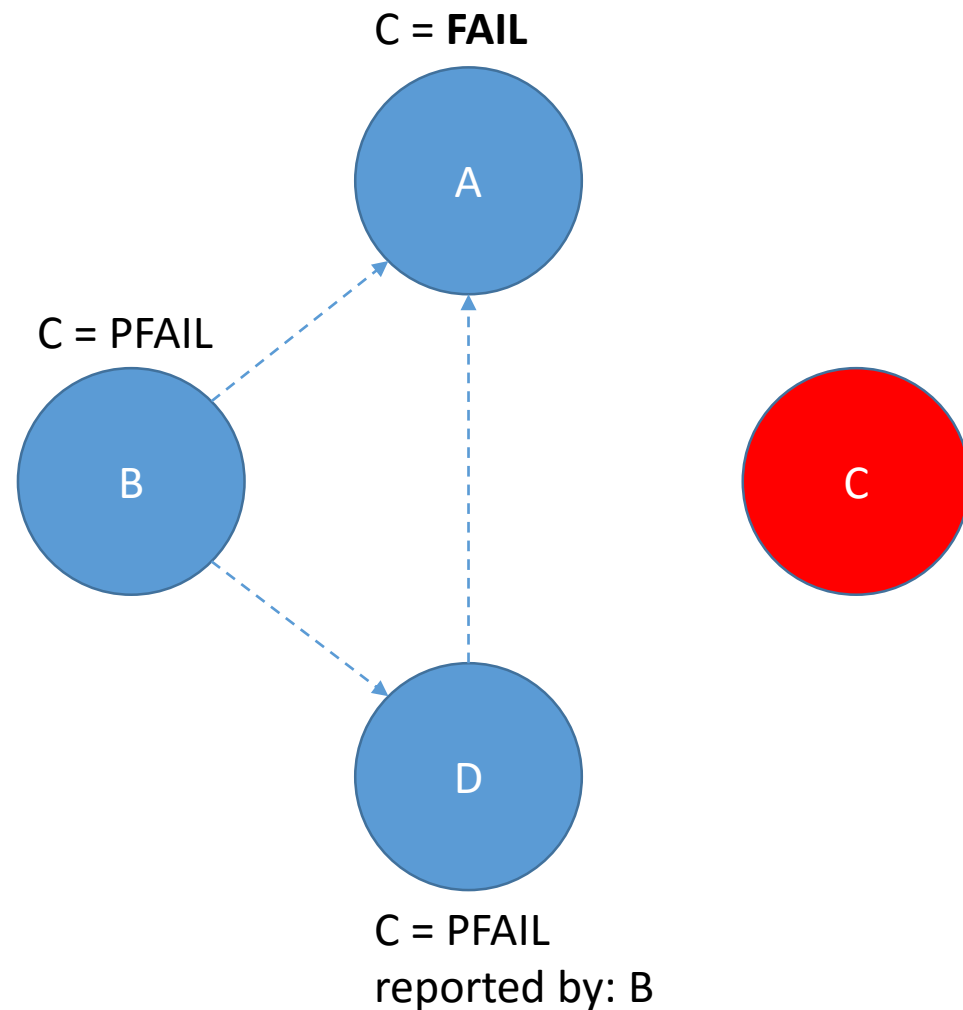
Fail判定

- 集群通过心跳检测节点是否下线
- 下线状态分为PFAIL (possible failure) 和FAIL
- A、B、D节点都发现C节点下线 (PFAIL)
- A节点首先收到B、D节点关于C节点的PFAIL传播消息



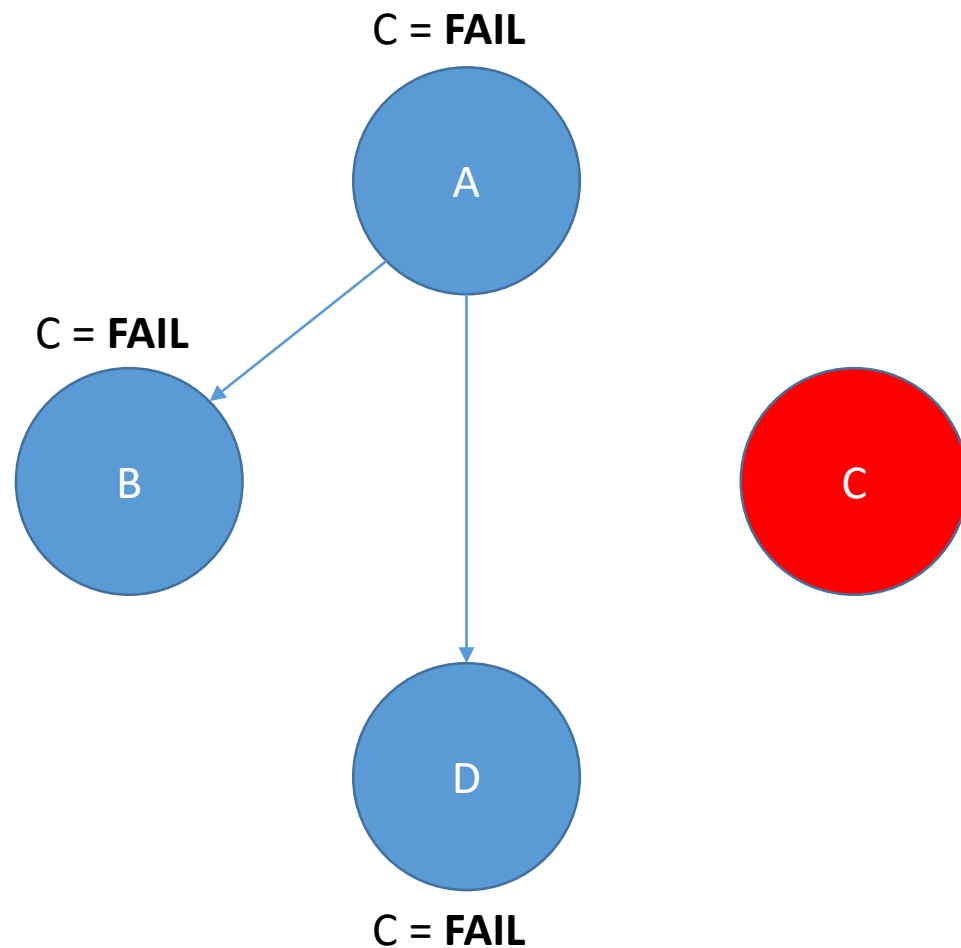
Fail判定

- 集群通过心跳检测节点是否下线
- 下线状态分为PFAIL (possible failure) 和FAIL
- A、B、D节点都发现C节点下线 (PFAIL)
- A节点首先收到B、D节点关于C节点的PFAIL传播消息
- 此时A节点发现集群中大多数节点都将C标记为PFAIL , 所以A将C标记为FAIL



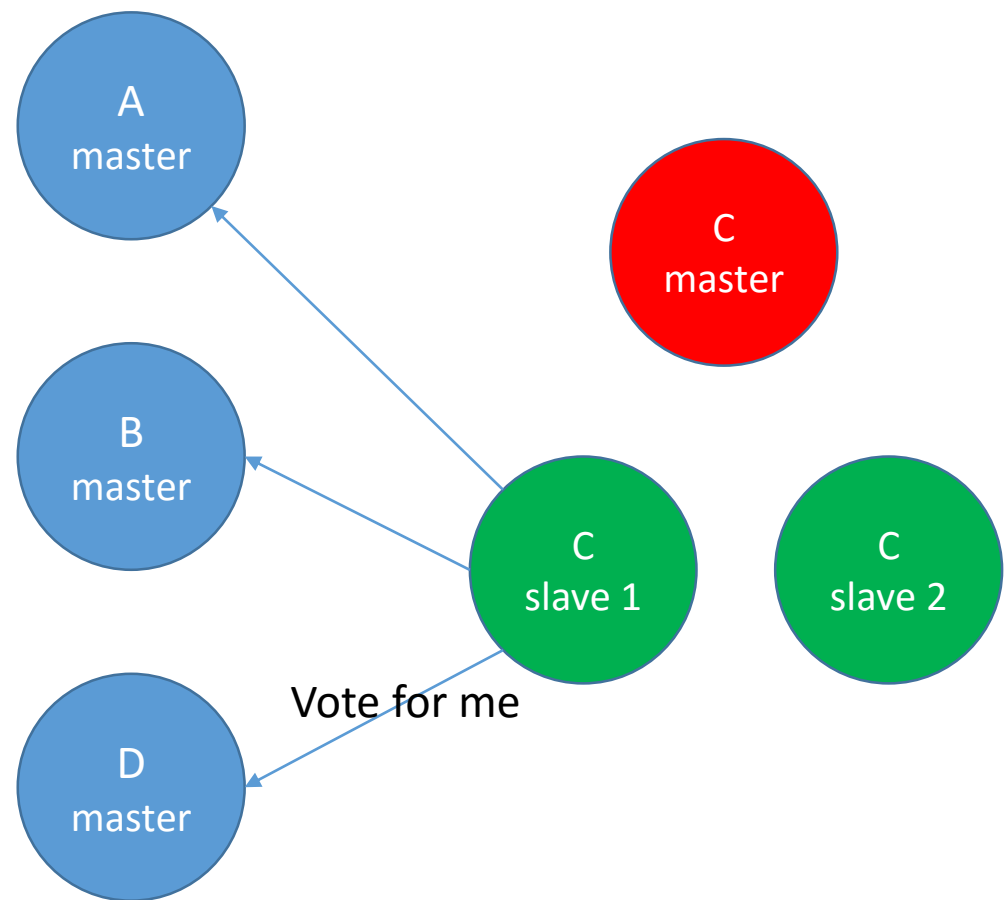
Fail判定

- 集群通过心跳检测节点是否下线
- 下线状态分为PFAIL (possible failure) 和FAIL
- A、B、D节点都发现C节点下线 (PFAIL)
- A节点首先收到B、D节点关于C节点的PFAIL传播消息
- 此时A节点发现集群中大多数节点都将C标记为PFAIL , 所以A将C标记为FAIL
- A节点马上广播FAIL消息给集群中的其它节点



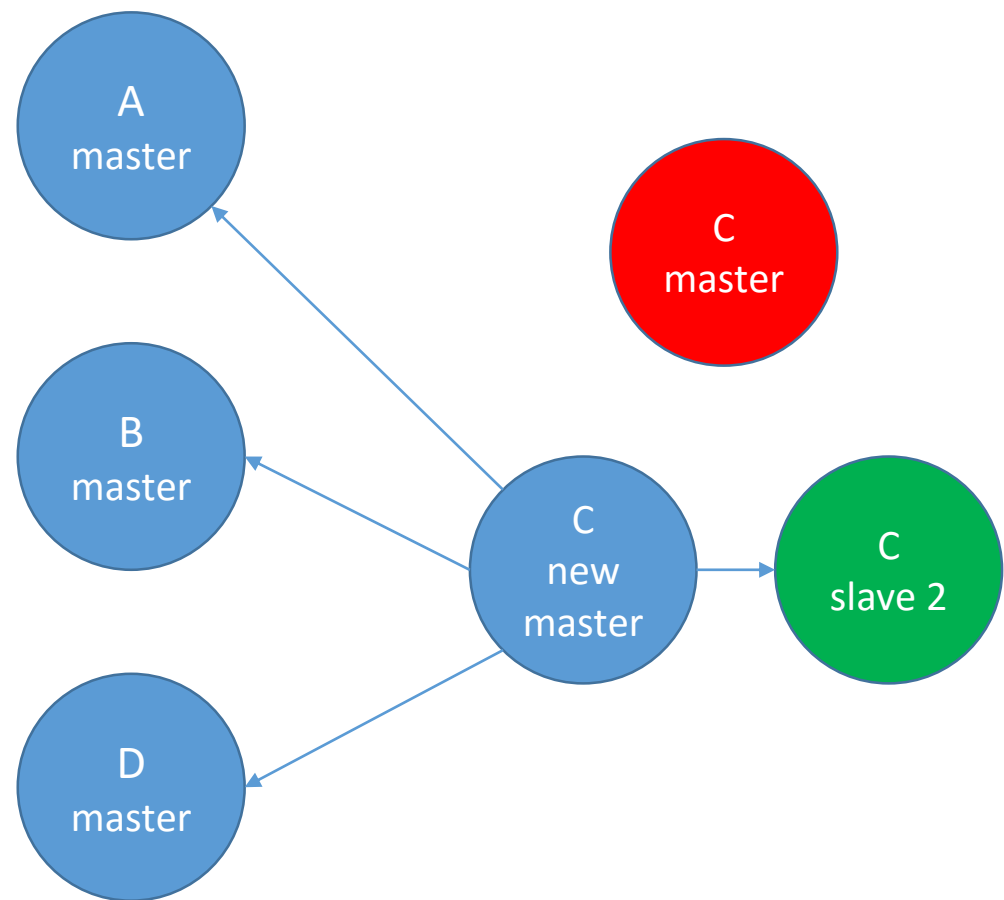
Leader选举

- 近似Raft算法
- Slave与FAIL master的offset越小说明数据越相近，也越有机会得到投票



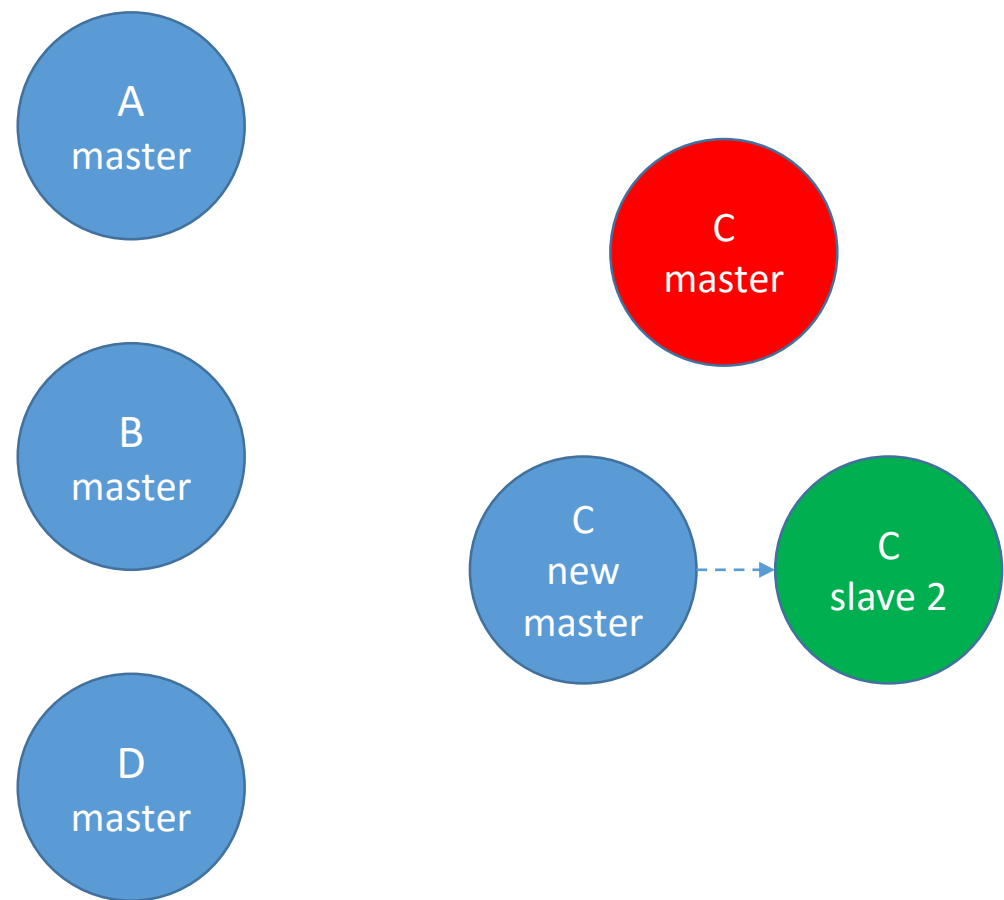
Leader选举

- 近似Raft算法
- Slave与FAIL master的offset越小说明数据越相近，也越有机会得到投票
- 一旦获得大多数节点的投票，从节点将晋升为新的主节点
- 它将接管FAIL节点所负责的所有slot，并将晋升信息马上广播给集群所有节点



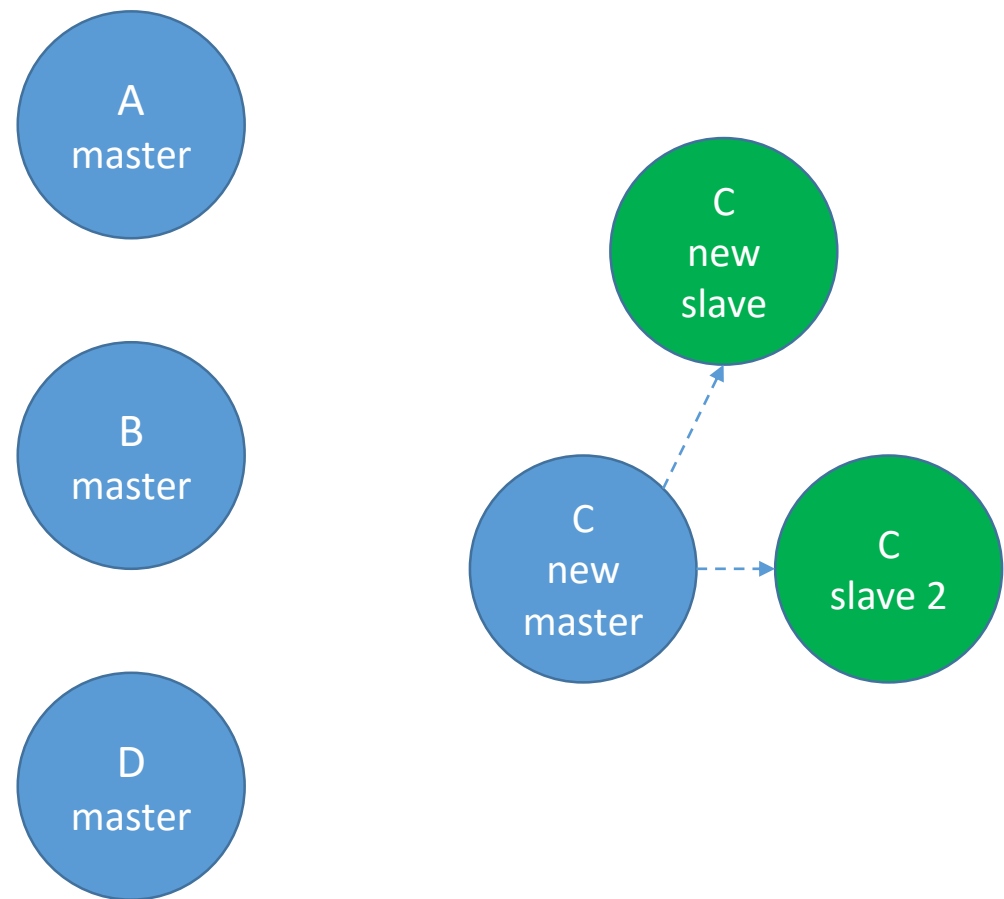
Leader选举

- 近似Raft算法
- Slave与FAIL master的offset越小说明数据越相近，也越有机会得到投票
- 一旦获得大多数节点的投票，从节点将晋升为新的主节点
- 它将接管FAIL节点所负责的所有slot，并将晋升信息马上广播给集群所有节点
- FAIL节点的所有从节点将重新同步新主节点的数据



Leader选举

- 近似Raft算法
- Slave与FAIL master的offset越小说明数据越相近，也越有机会得到投票
- 一旦获得大多数节点的投票，从节点将晋升为新的主节点
- 它将接管FAIL节点所负责的所有slot，并将晋升信息马上广播给集群所有节点
- FAIL节点的所有从节点将重新同步新主节点的数据
- 当FAIL节点重新上线，将调整状态为从节点





运维经验



■ 集群扩容

➤ 将新实例加入集群

- ✓ `redis-trib.rb add-node 新masterIP:port 现有节点IP:port`
- ✓ `redis-trib.rb add-node --slave --master-id <master-id> 新slaveIP:port 现有节点IP:port`

➤ 扩容目标

- ✓ 在key分布大致均衡的前提下，尽量使每个实例的slot个数一致

集群扩容

Node A	
0	
1	
2	
3	

Node B	
4	
5	
6	
7	
8	

Node C	

集群扩容

- 按照slot个数排序
- 算出平均slot数

Node B	Node A	Node C
4	0	
5	1	
6	2	
7	3	
8		

集群扩容

➤ 按照slot个数排序

➤ 算出平均slot数

➤ 将大于平均值的节点的槽迁移到新节点

✓ `redis-trib.rb reshard --from <node-id> --to <node-id> --slots <numslots> --yes`
`<host>:<port>`

➤ Redis 3.2开始，redis-trib.rb支持rebalance功能了！

✓ 支持权重、阈值配置

✓ 对于新扩容节点需要增加 `--use-empty-masters` 参数

✓ [demo](#)

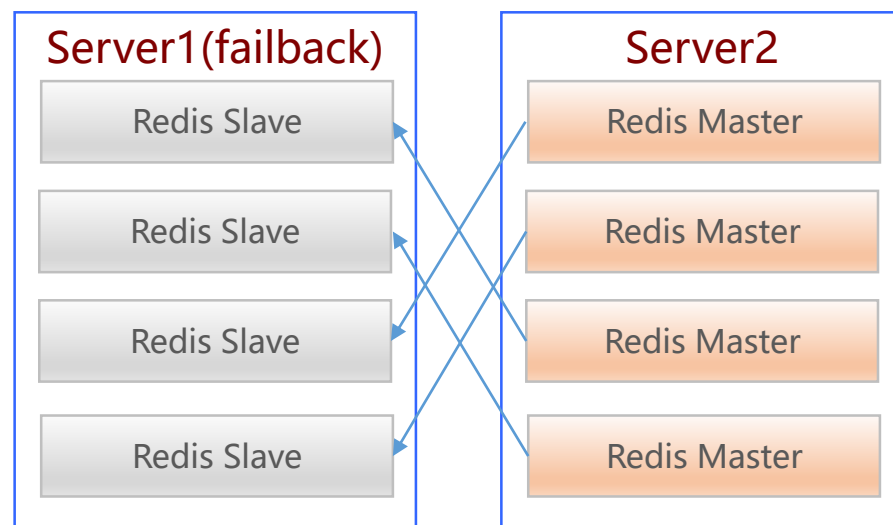
✓ [说明文档](#)

Node B	Node A	Node C
6	1	4
7	2	5
8	3	0

■ 手动failover

➤ 命令：cluster failover

➤ 场景1：节点故障恢复后，需要恢复之前的主从关系

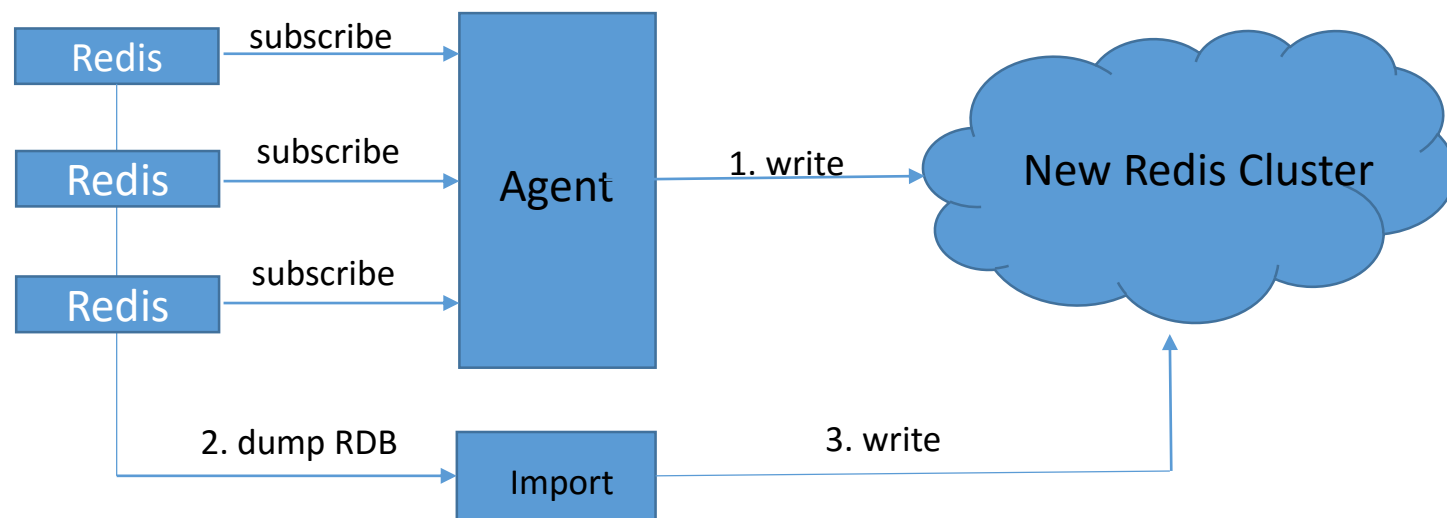


■ 手动failover

- 场景2：集群子版本升级
- 步骤1：关闭从节点->升级->重启从节点
 - ✓ 注意：重启后需要等待主从重新同步完成
- 步骤2：对新版本实例做手动failover->新主晋升完成
- 重复以上两步，直到集群所有实例升级完成
- 注意：此操作可能会有新老版本兼容性风险，必须充分测试！

集群迁移

➤ 场景：Redis单机版向集群版迁移



■ 集群迁移

➤ Agent

- ✓ 利用keyspace notifications订阅单机版Redis的所有写事件
- ✓ 将订阅到的事件同步给Redis Cluster
- ✓ 如果Agent挂了，需要重启Agent，并且重新生成对应实例的RDB+import重新导入

➤ Import

- ✓ 对于目标集群中已经存在的key
 - 读出源集群key对应的value信息、以及TTL信息
 - 将key、value、TTL信息写入目标集群

■ 集群迁移

- **Import导入完成后，做好两边的数据比对**
- **客户端一次性切写到Redis Cluster**
 - ✓ 切写后考虑写请求回流，防止需要回滚的情况
- **客户端读再逐渐切到Redis Cluster**



本PPT来自Redis交流群第一次线下活动

视频在<http://www.meipai.com/media/733309504?from=groupmessage>

欢迎交流redis的开发和运维，群主每日精选一篇redis有关的文章在群里分享。 所有每日分享的文章都记录在以下
repo：<https://github.com/tao12345666333/redis-articles>

欢迎加群交流，请加我微信，二维码如下

THANKS