架构实战营模块8 - 第4课

分片架构设计技巧

李运华

前阿里资深技术专家（P9）

教学目标

1. 学习 Elasticsearch 分片架构设计技巧
2. 学习 Redis cluster 架构设计技巧
3. 学习 MongoDB 和 HDFS 架构设计技巧

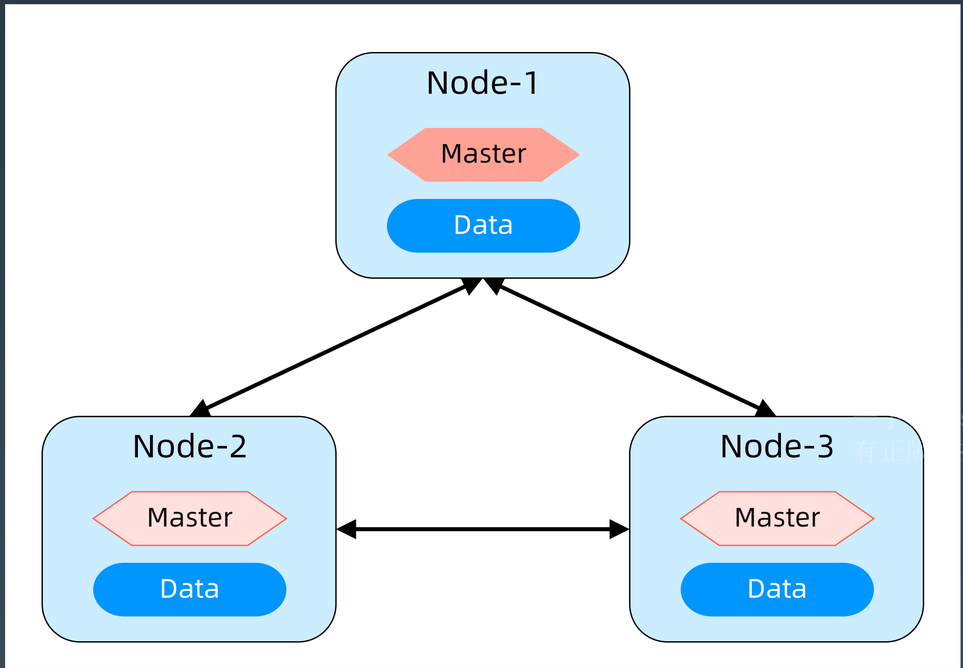
它山之石可以攻玉！

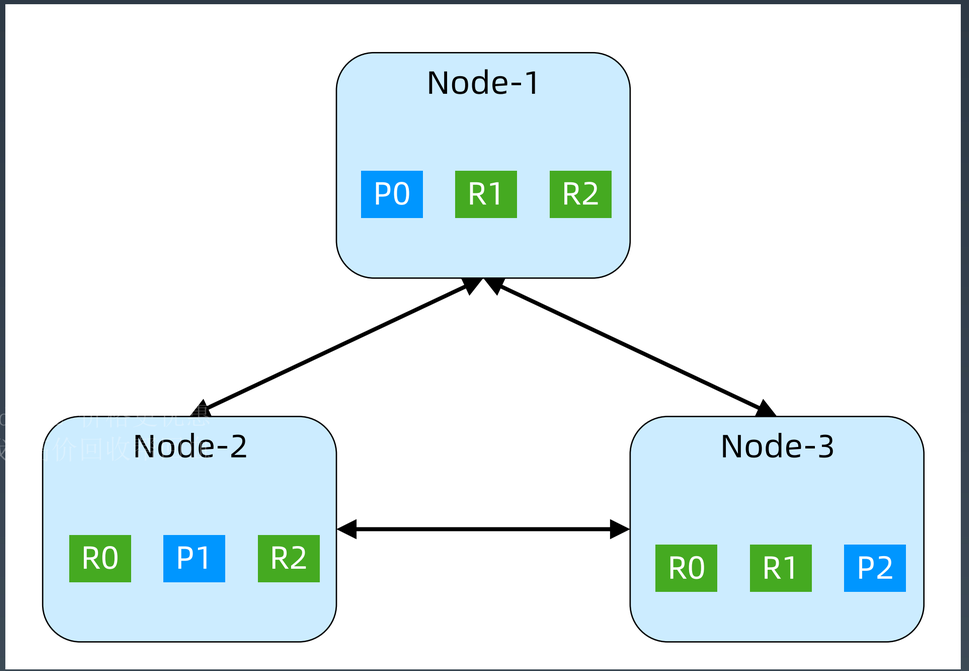
极客时间 训练营

目录

1. Elasticsearch 集群设计技巧

ES 的基本架构

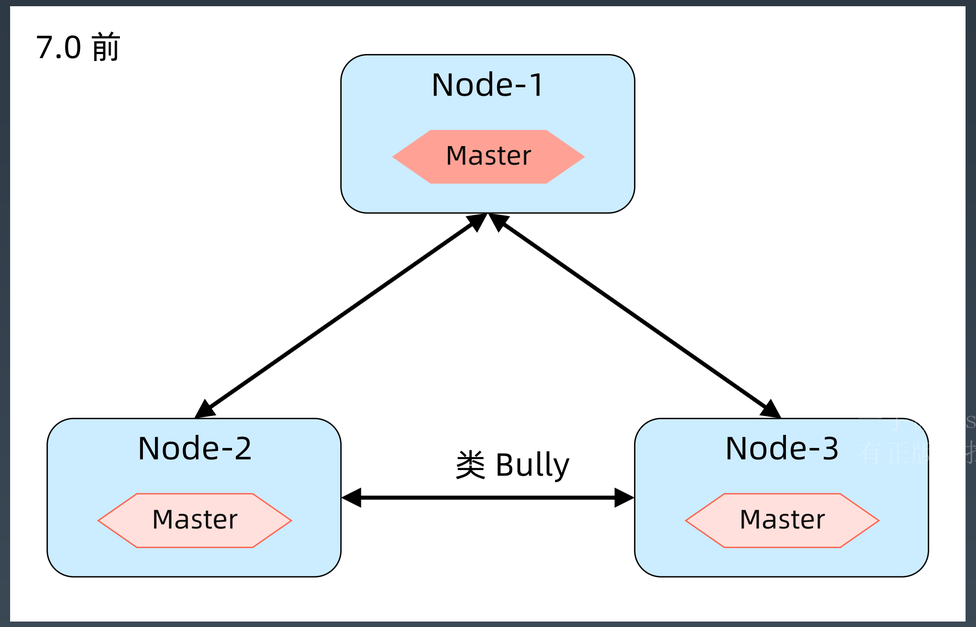


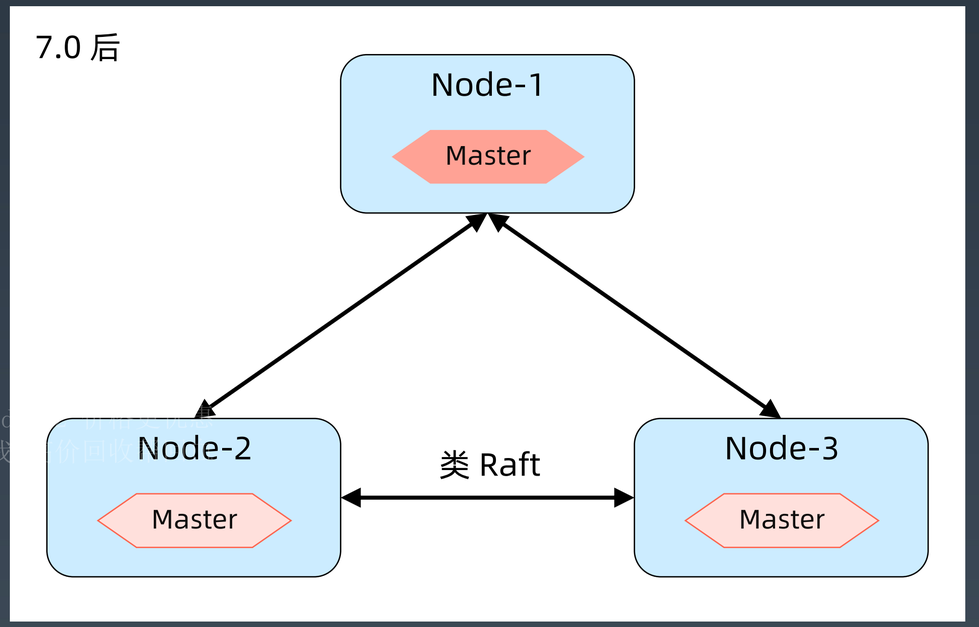


1. 节点可以配置为不同角色，通过选举 Master 管理集群；
2. Coordinating：协调节点，Master：管理节点，Data：数据存储节点， 更多节点类型和详细介绍请参考：官方文档。

数据是按照索引分片的，而不是按照节点分片，每个分片可以有多个副本来保证高可用，例如图中P0有2个R0副本。

ES 的选举

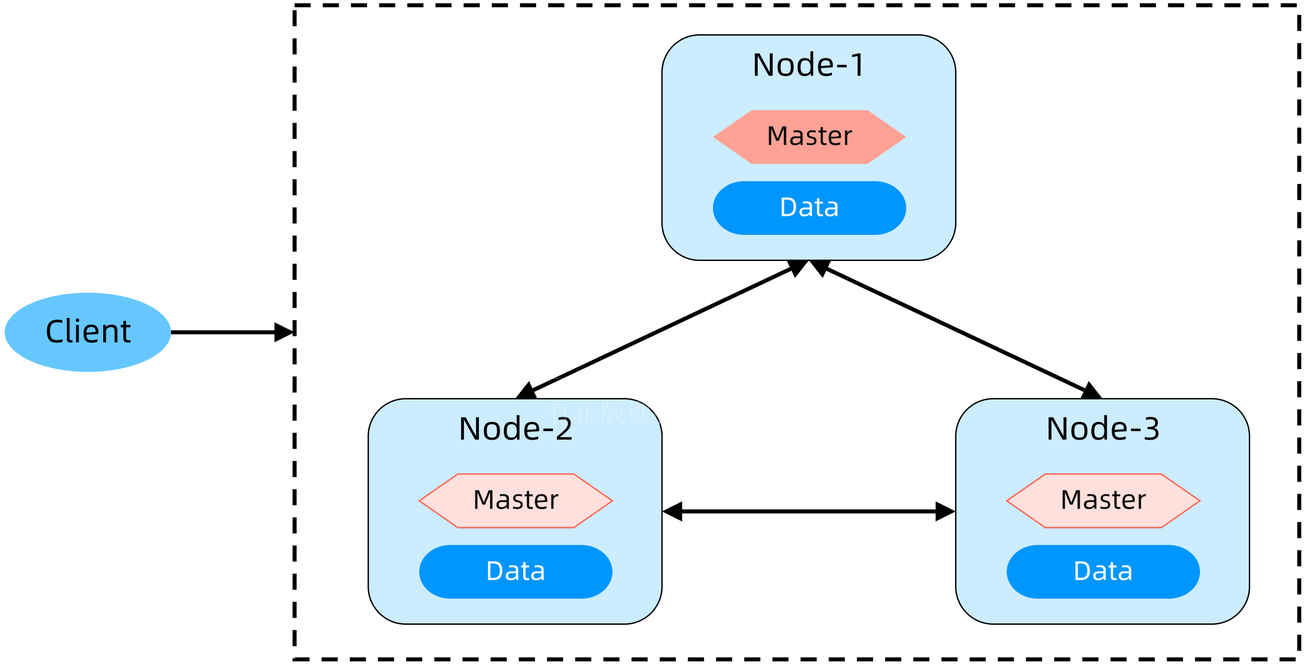




1. 先根据节点的 clusterStateVersion 比较，clusterStateVersion 越大，优先级越高；
2. clusterStateVersion 相同时，进入 compareNodes，其内部按照节点的 Id 比较(Id 为节点第一次启动时随机生成)。

Zen Discovery 采用了很多分布式共识算法中的想法，但只是有机地采用，并没有严格按照理论所规定的那个模型。7.0 是基于 Raft 但不是 Raft。[参考链接]

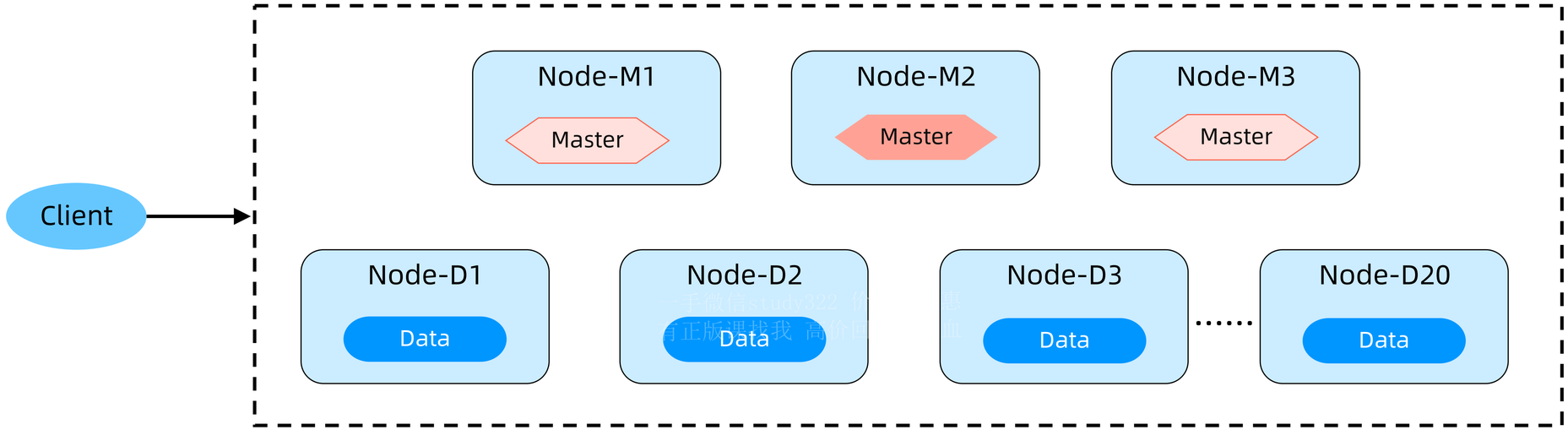
ES 的部署架构模式1 - Master 和 Data 混合部署



你能算出这种架构支持的数据量具体多大么？

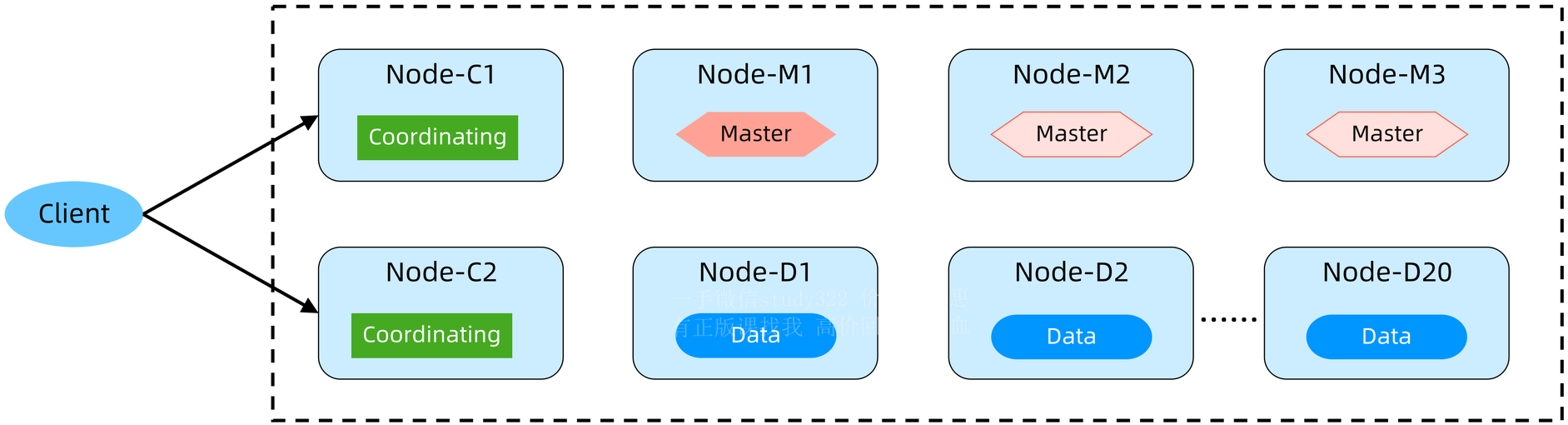
1. 节点同时配置为 Master 和 Data；
2. 每个节点都可以接收和处理客户端请求，写入操作会转发到数据主分片的 Node；
3. 适用于数据量不大的业务。

ES 的部署架构模式2 - Master 和 Data 分离部署



1. Master 节点和 Data 节点分离配置，Master 节点数量为3个或者5个，Data 节点数量可以是几十个；
2. Master 节点不处理读写请求，只负责集群管理，Data节点处理读写请求和数据存储；
3. 适用于数据量比较大的业务。

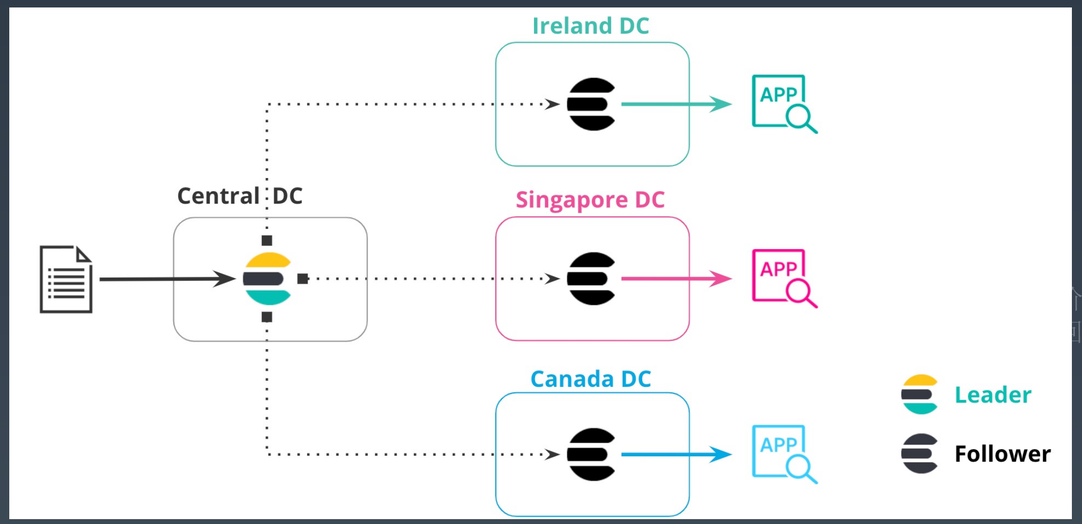
ES 的部署架构模式3 - Coordinating 分离部署

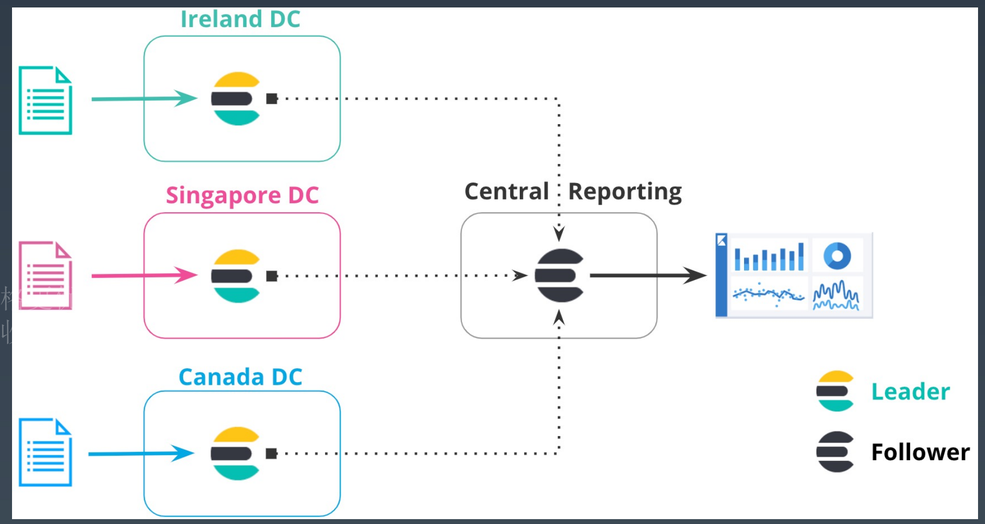


1. Master 节点数量为3个或者5个，Data 节点数量可以是几十个，Coordinating 节点有2个以上；
2. Master 节点不处理读写请求，只负责集群管理，Coordinating 负责读写聚合，Data 节点负责数据存储；
3. 适用于数据量比较大，读写请求比较复杂的业务。

架构模式2和架构模式3优选哪个？为什么？

ES 部署架构模式4 - Cross cluster replication



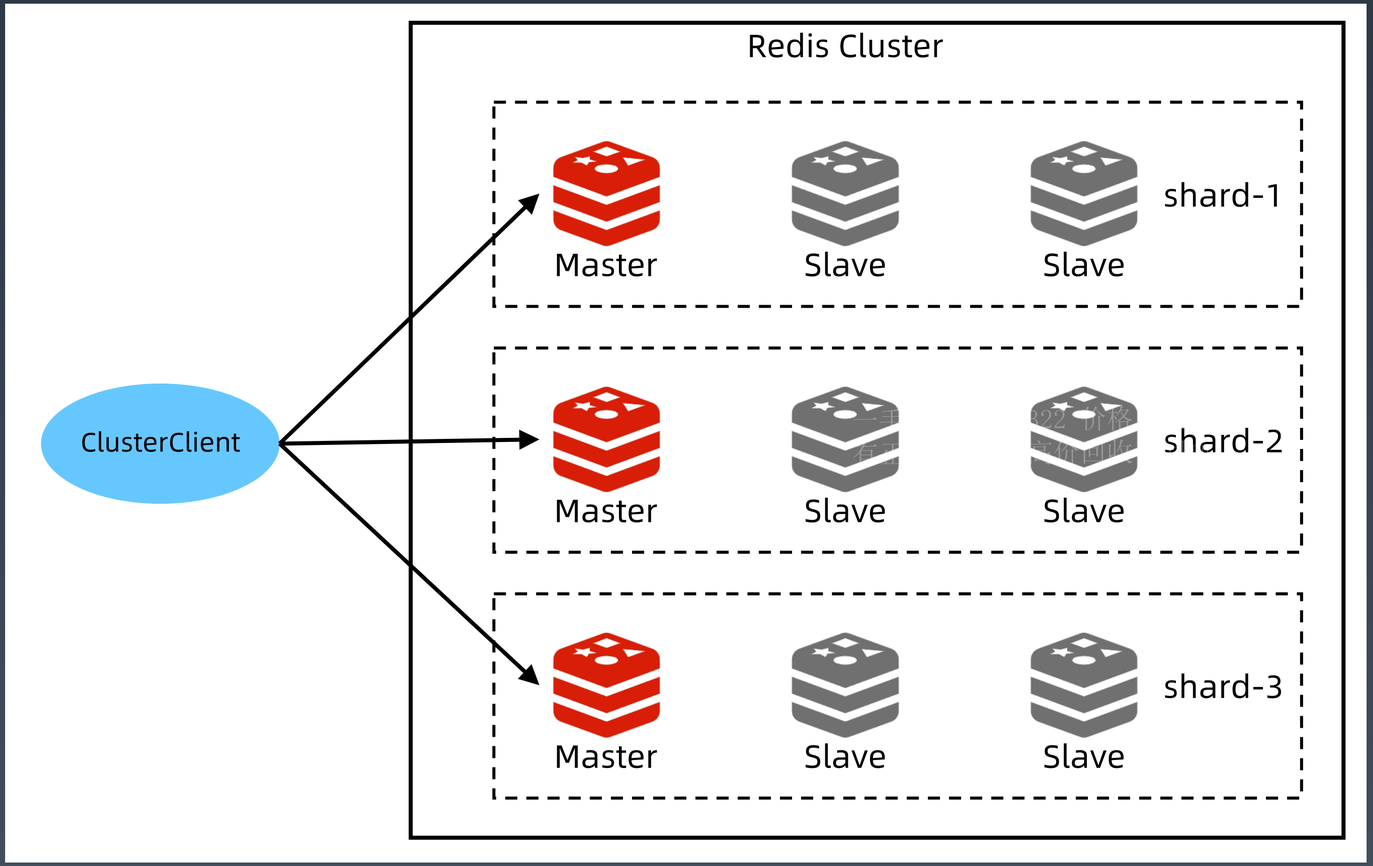


1. 配置两个集群为 Cross cluster replication，Leader 负责数据读写，Follower 复制数据，负责数据读取；
2. 适应场景：本地化、聚合存储。

学习链接

2. Redis cluster 设计分析

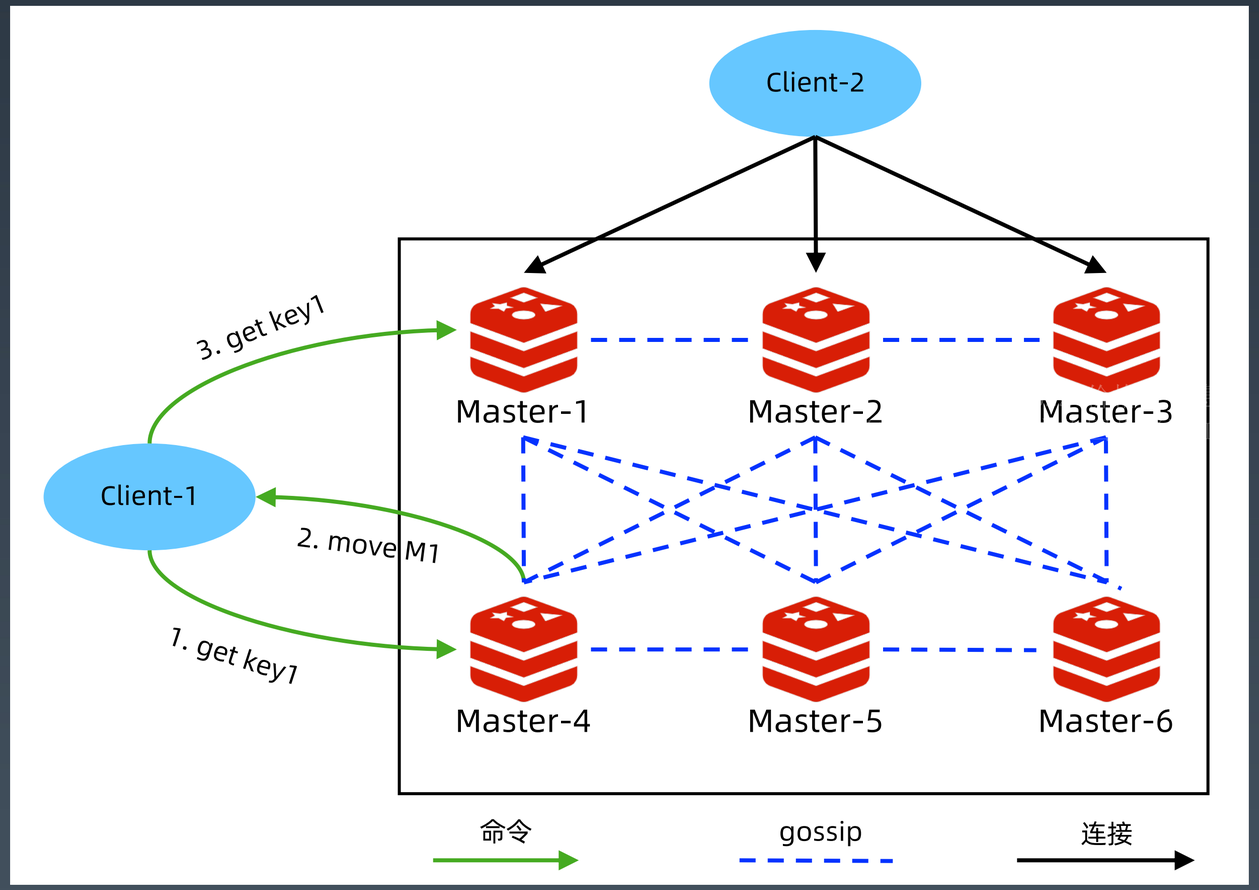
Redis Cluster 基本架构



1. Cluster 分为多个分片，不同分片保存不同数据；
2. 每个分片内部通过主备复制来保证可用性；
3. 分片内部自动实现 Master 选举，但不依赖 Sentinel，Cluster 本身具备分片选举的能力；
4. 客户端连接集群需要特定的实现，例如 jedisCluster，因为 Cluster 有特有的 Redis 命令。

学习链接：tutorial，specification，集群原理分析。

Redis 数据分布和路由

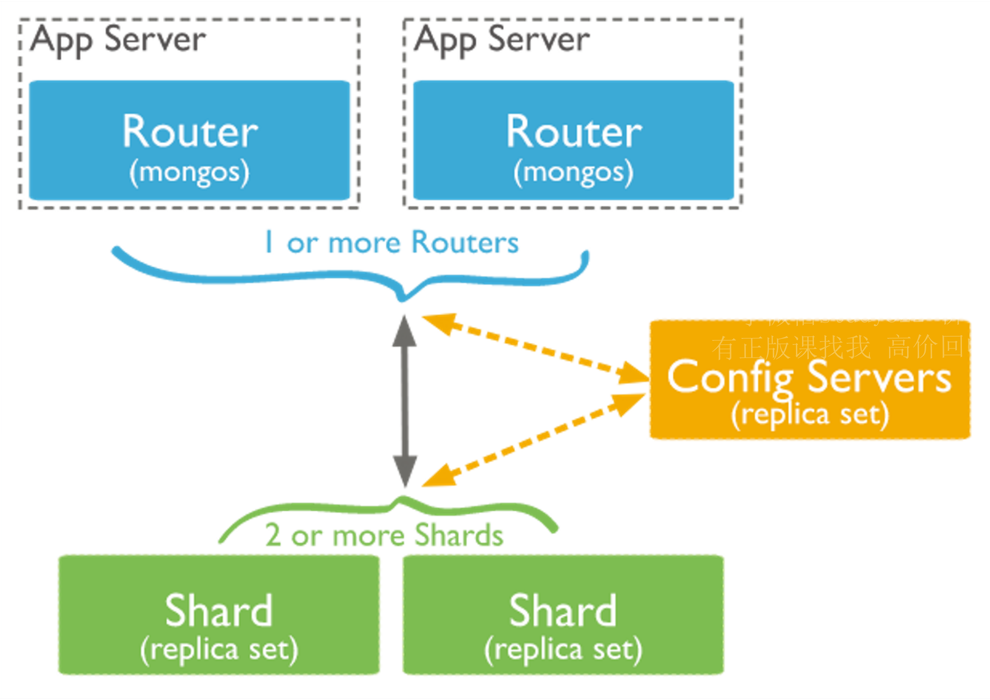


1. 所有 key 按照 hash 算法分为 16384个 槽位，然后将槽位分配给分片；
2. 节点之间通过 gossip 交换信息，节点变化的时候会自动更新集群信息；
3. 每个节点都有所有 key 的分布信息；
4. Client 连接任意节点，由节点用 move 指令来告诉实际的数据位置。

学习链接：Redis cluster specification

3. 其它分片集群设计分析

MongoDB sharding 架构



【mongos】

1. 独立部署的代理程序，应用程序请求发给 mongos；
2. 可以和应用程序部署在一起，也可以和 Shard 服务器部署在一起；
3. 为了提升性能，mongos 会缓存 Config Server 上保存的 cluster 配置信息；

【Config Server】

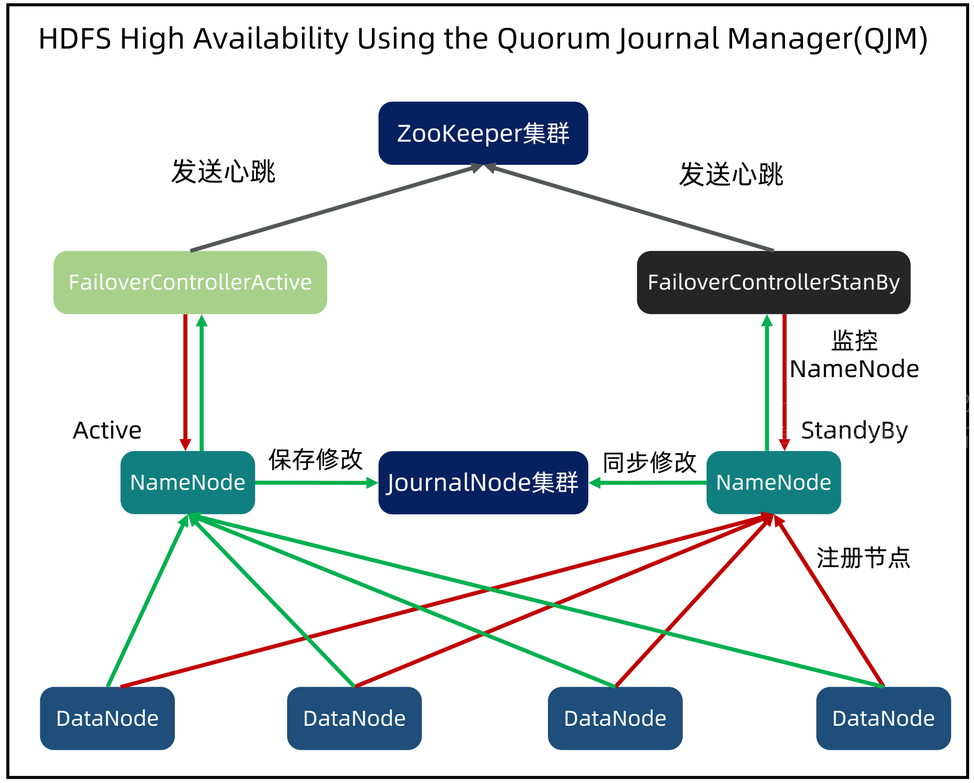
1. 存储集群的元数据；
2. 自身通过 replica set 保证高可用；
3. 当 Config Server 挂掉的时候，cluster 进入 read only。

【Shard】

1. 存储分片数据的服务器；
2. 自身通过 replica set 保证高可用，如果全部挂掉，分片就无访问了。

学习链接：官方文档

HDFS 架构



【NameNode】
集群管理节点，保存集群元数据，管理集群（平衡、分配等）。

【DataNode】
存储实际的数据，数据按照 block 存储。

【JournalNode】
1. 当 Active NameNode 修改集群状态后，会写日志到 JournalNode 集群里面；
2. StandBy NameNode 会监听 JournalNode，发生变化的时候会拉取日志；
3. JournalNode 至少3个，达到多数日志复制写入才算成功。

【FailoverController】
1. NameNode 节点内的一个独立进程，监控 NameNode 状态；
2. 依赖 ZooKeeper 做高可用。

链接：官方文档 ，学习链接

如果你来优化这个架构，可以怎么做？

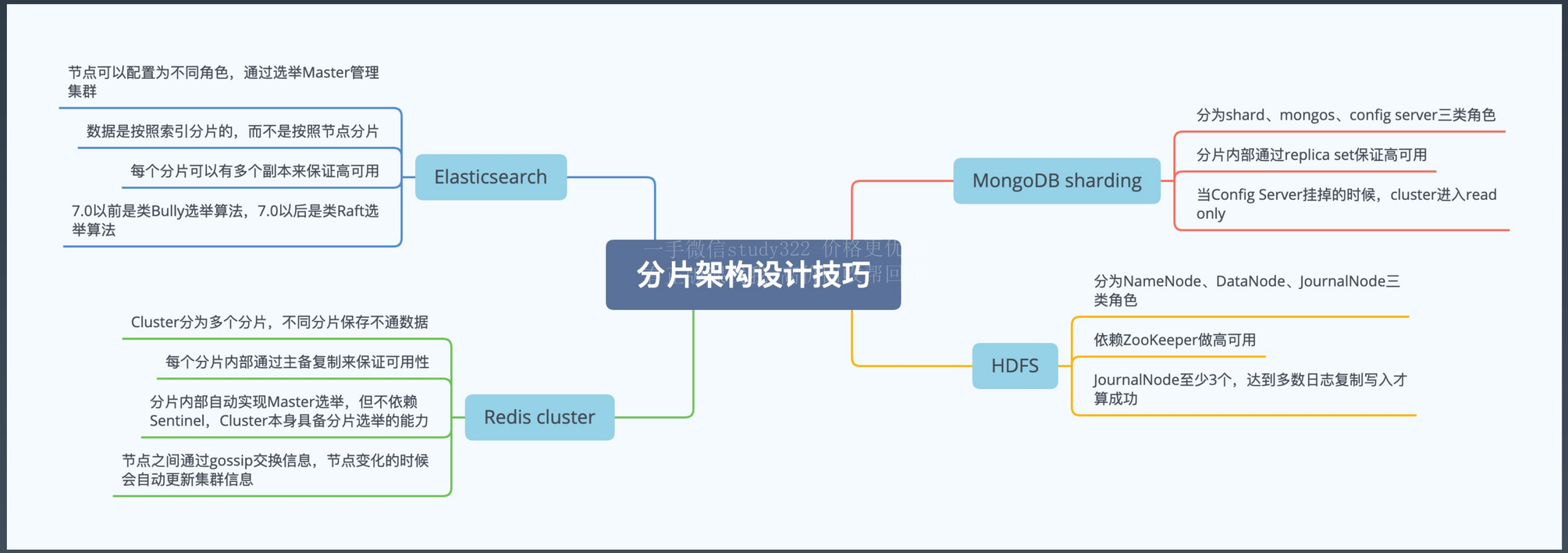
各个架构的简单分析对比

极客时间 训练营

维度	ES	Redis Cluster	MongoDB sharding	HDFS
实现复杂度	高，需要选举 Master 节点，且角色类型众多	高，需要 gossip 协议来交互信息	低，Config Server 管理集群	低，NameNode 管理集群
部署复杂度	中，需要根据业务规模采用不同的部署模式	低，只有一种模式，平滑伸缩	高，3类节点，且 Config server 和 Shard 要部署主备	高，部署 ZooKeeper、NameNode、DataNode、JournalNode
硬件成本	低	中，每个分片要部署 master 和 slave 节点	高，Config server 和 Shard 要部署主备	高，节点类型很多
支持集群规模	超大规模	中等规模，建议服务器数量100以内	超大规模	超大规模
适应场景	数据查询和分析	缓存	数据存储	数据存储

为什么 Redis Cluster 不适合超大规模集群？

本节思维导图


分片架构设计技巧
Elasticsearch
节点可以配置为不同角色，通过选举Master管理集群
数据是按照索引分片的，而不是按照节点分片
每个分片可以有多个副本来保证高可用
7.0以前是类Bully选举算法，7.0以后是类Raft选举算法
Redis cluster
Cluster分为多个分片，不同分片保存不通数据
每个分片内部通过主备复制来保证可用性
分片内部自动实现Master选举，但不依赖Sentinel，Cluster本身具备分片选举的能力
节点之间通过gossip交换信息，节点变化的时候会自动更新集群信息
MongoDB sharding
分为shard、mongos、config server三类角色
分片内部通过replica set保证高可用
当Config Server挂掉的时候，cluster进入read only
HDFS
分为NameNode、DataNode、JournalNode三类角色
依赖ZooKeeper做高可用
JournalNode至少3个，达到多数日志复制写入才算成功

随堂测验

极客时间 训练营

【判断题】

1. Elasticsearch 的节点角色包括 master、slave、data 等多种类型。
2. Elasticsearch 的部署架构可以灵活设计，支持不同业务场景。
3. Redis cluster 不适合超大规模集群。
4. MongoDB sharding 架构节点数量多，实现复杂。
5. HDFS 的 NameNode 和 MongoDB sharding 的 config sever 本质上都是集群管理。

【思考题】

HDFS 采取 JournalNode 这种模式的可能原因是什么？

Q&A

茶歇时间

八卦，趣闻，内幕……

THANKS