极客时间训练营

架构实战营模块8 - 第2课

如何基于 ZooKeeper 实现高可用架构

李运华

前阿里资深技术专家（P9）

教学目标

极客时间 训练营

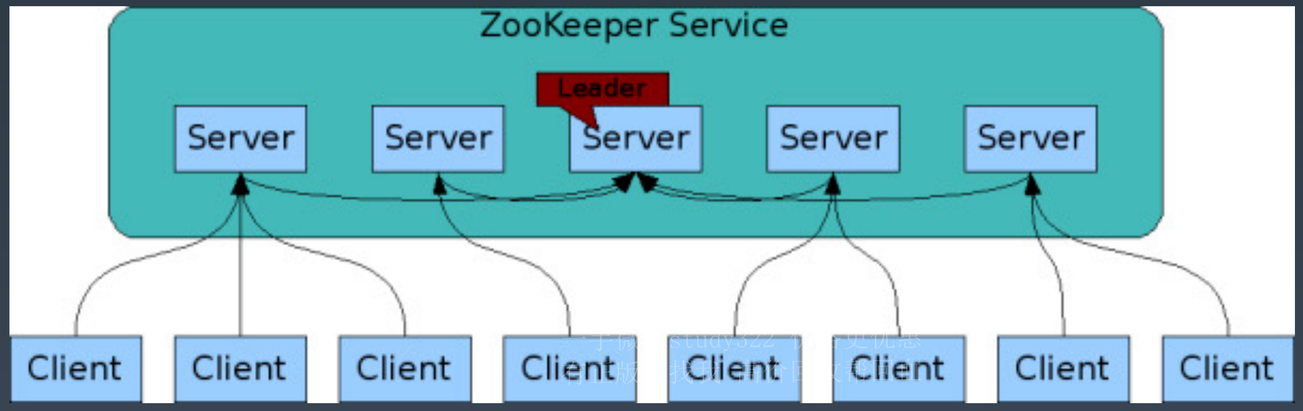
1. 掌握如何应用 ZooKeeper 实现高可用架构的具体技巧

不需要研究 ZooKeeper 源码，你也可以用好 ZooKeeper！

目录

1. ZooKeeper 高可用相关特性

ZooKeeper 介绍



ZooKeeper is a distributed, open-source coordination service for distributed applications.

It exposes a simple set of primitives that distributed applications can build upon to implement higher level services for synchronization, configuration maintenance, and groups and naming. 参考链接

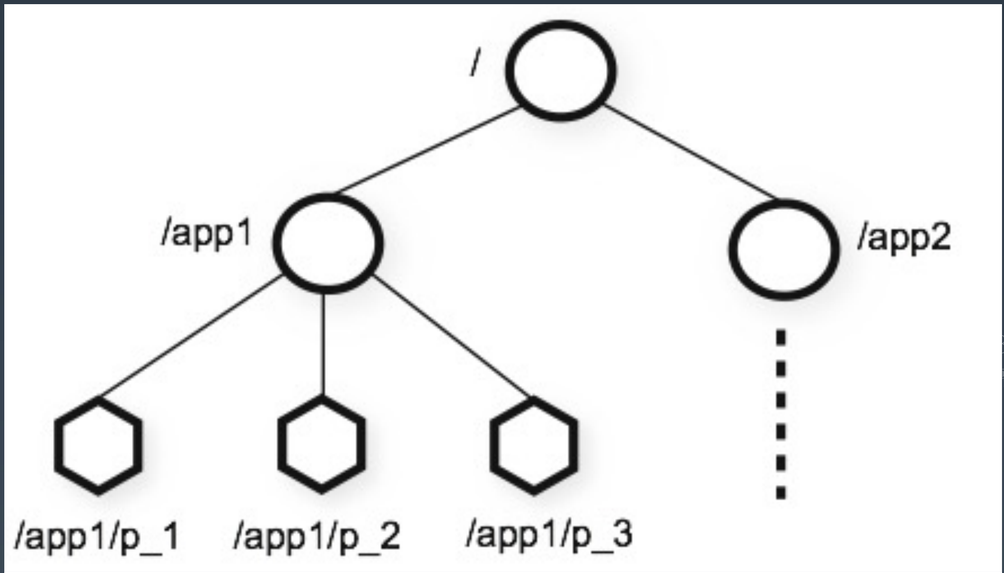
ZooKeeper 应用

极客时间 训练营

		Usage Patterns								
Project	**Consensus System**	**SR**	**LR**	**SS**	**BO**	**SD**	**GM**	**LE**	**MM**	**Q**
GFS	Chubby			✓				✓	✓	
Borg	Chubby/Paxos	✓				✓		✓		
Kubernetes	etcd						✓		✓	
Megastore	Paxos		✓							
Spanner	Paxos	✓								
Bigtable	Chubby						✓	✓	✓	
Hadoop/HDFS	ZooKeeper	✓						✓		
HBase	ZooKeeper	✓		✓			✓		✓	
Hive	ZooKeeper			✓					✓	
Configerator	Zeus								✓	
Cassandra	ZooKeeper					✓		✓	✓	
Accumulo	ZooKeeper		✓	✓					✓	
BookKeeper	ZooKeeper						✓		✓	
Hedwig	ZooKeeper						✓		✓	
Kafka	ZooKeeper						✓	✓	✓	
Solr	ZooKeeper							✓	✓	✓
Giraph	ZooKeeper		✓		✓				✓	
Hama	ZooKeeper				✓					
Mesos	ZooKeeper							✓		
CoreOS	etcd					✓				
OpenStack	ZooKeeper					✓				
Neo4j	ZooKeeper			✓				✓		

参考链接：Consensus in the Cloud: Paxos Systems Demystified

ZooKeeper 数据模型



【Watches】
Clients can set watches on znodes. Changes to that znode trigger the watch and then clear the watch. When a watch triggers, ZooKeeper sends the client a notification.

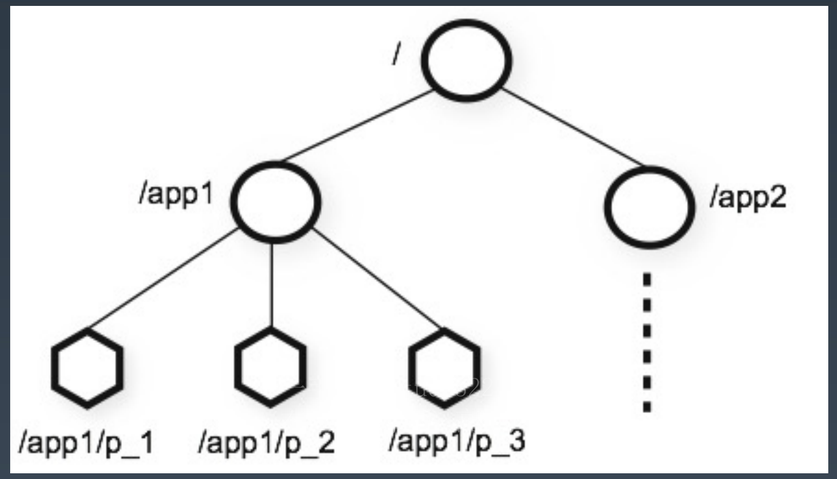
【Data Access】
The data stored at each znode in a namespace is read and written atomically.

【Ephemeral Nodes】
These znodes exists as long as the session that created the znode is active.

【Sequence Nodes】
When creating a znode you can also request that ZooKeeper append a monotonicly increasing counter to the end of path. This counter is unique to the parent znode.

参考链接

ZooKeeper 设计步骤



1. 设计 path

/app1/p_1

2. 选择 znode 类型

Ephemeral(临时、短命)
Sequence(有序)
Normal

3. 设计 znode 数据

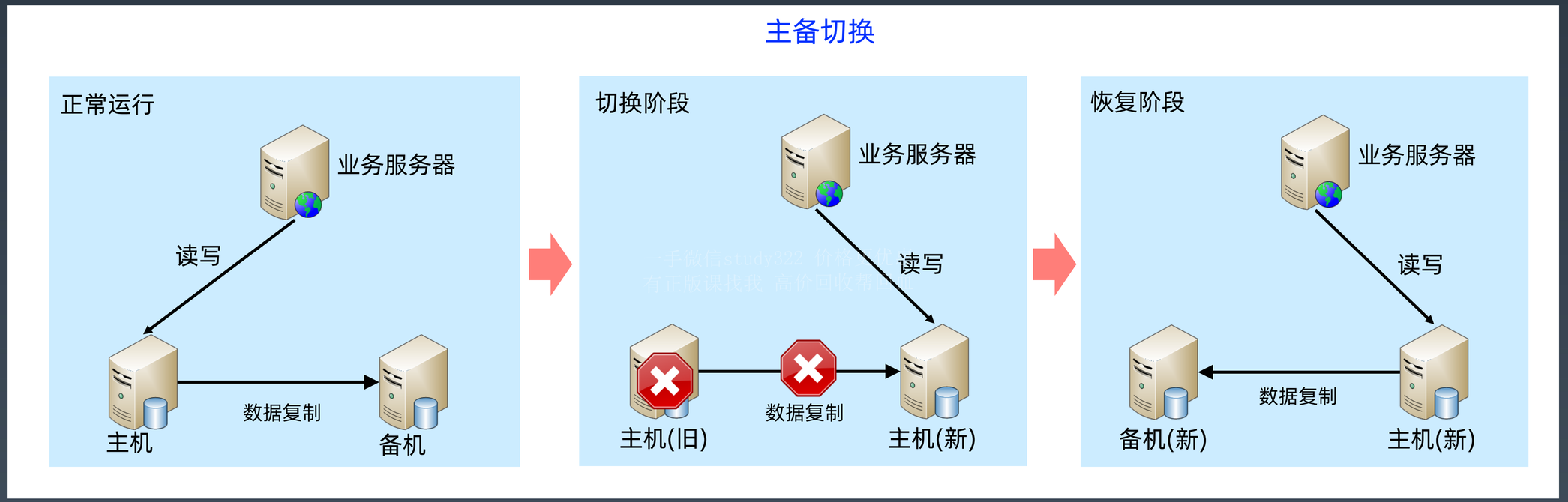
节点里面存放什么数据

4. 设计 Watch

Client 关注什么事件，事件发生后如何处理。

2. ZooKeeper 实现主备切换架构

主备切换基本架构


主备切换
正常运行
业务服务器
读写
主机
数据复制
备机
切换阶段
业务服务器
读写
主机(旧)
数据复制
主机(新)
恢复阶段
业务服务器
读写
备机(新)
数据复制
主机(新)

主备切换 ZooKeeper 方案

极客时间 训练营

1. 设计 Path

由于只有2个角色，因此直接设置两个 znode 即可：master、slave，样例：/com/taobao/book/operating/master，/com/taobao/book/operating/slave。

2. 选择节点类型

当 master 节点挂掉的时候，原来的 slave 升级为 master 节点，因此用 ephemeral 类型的 znode。

3. 设计节点数据

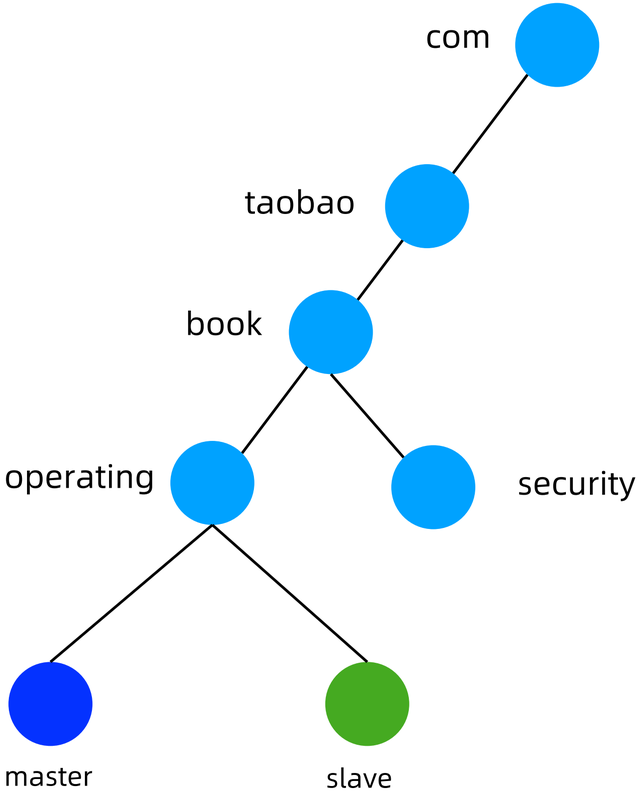
由于 slave 成为 master 后，会成为新的复制源，可能出现数据冲突，因此 slave 成为 master 后，节点写入成为 master 的时间，这样方便人工修复冲突数据。

4. 设计 Watch

1. 节点启动的时候，尝试创建 master znode，创建成功则切换为 master，否则创建 slave znode，成为 slave；
2. 如果 slave 节点收到 master znode 删除的事件，就自己去尝试创建 master znode，创建成功，则自己成为 master，删除自己创建的 slave znode。

参考代码

ZooKeeper 的目录



3. ZooKeeper 实现集群选举

集群选举方案1 - 最小节点获胜

极客时间 训练营

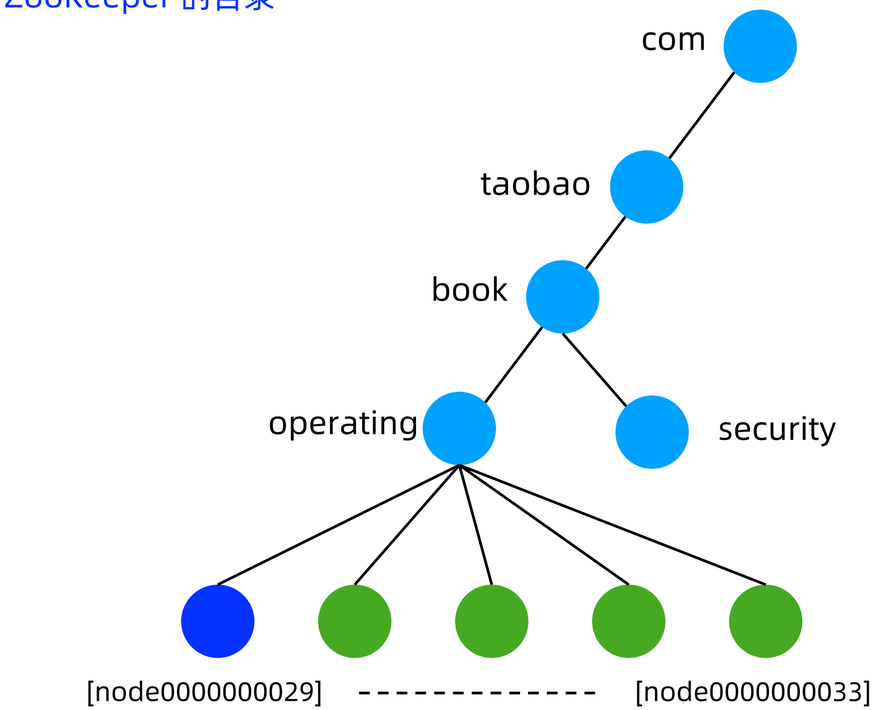
1. 设计 Path
集群共用父节点 parent znode，集群中的每个节点在 parent 目录下创建自己的 znode。

2. 选择节点类型
当 Leader 节点挂掉的时候，持有最小编号 znode 的集群节点成为新的 Leader，因此用 ephemeral_sequential 类型 znode。

3. 设计节点数据
可以根据业务需要灵活写入各种数据。

4. 设计 Watch
1. 节点启动或者重连后，在 parent 目录下创建自己的 ephemeral_sequntial znode；
2. 创建成功后扫描 parent 目录下所有 znode，如果自己的 znode 编号是最小的，则成为 Leader，否则 watch parent 目录；
3. 当 parent 目录有节点删除的时候，看看其编号是否正好比自己小1，如果是则自己成为 Leader，如果不是继续 watch。

ZooKeeper 的目录



Curator 的 LeaderLatch、LeaderSelector 采用这种策略，参考链接。

集群选举方案2 - 抢建唯一节点

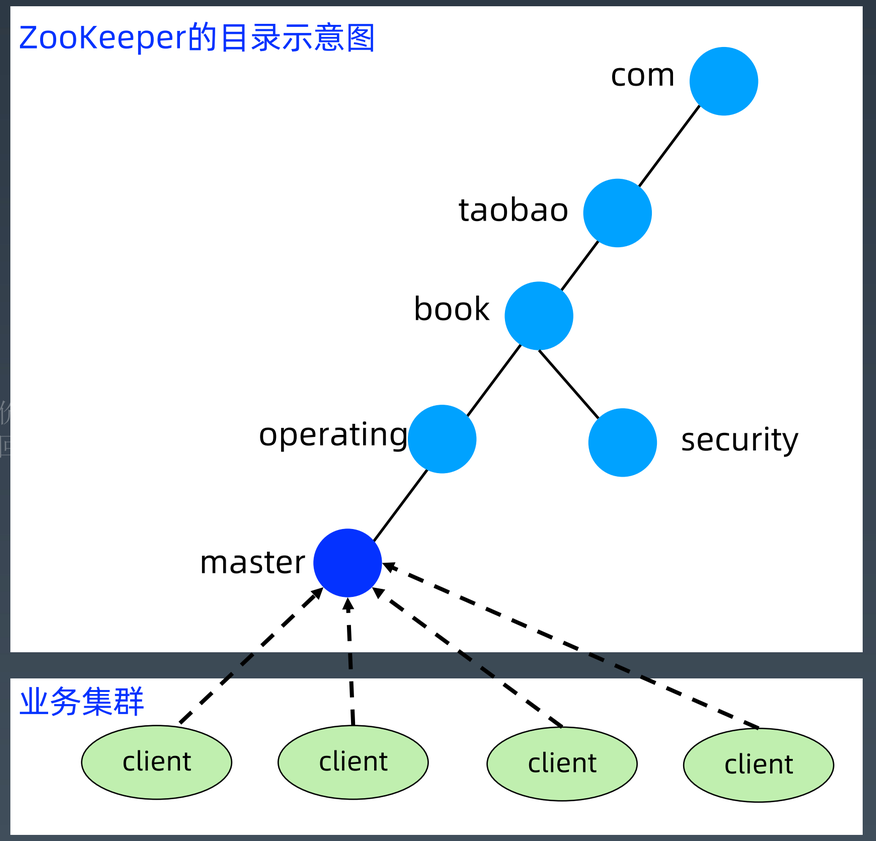
极客时间 训练营

1. 设计 Path
集群所有节点只有一个 Leader znode，本质上就是一个分布式锁。

2. 选择 znode 类型
当 Leader 节点挂掉的时候，剩余节点都来创建 Leader znode，看谁能最终抢到 Leader znode，因此用 ephemeral 类型。

3. 设计节点数据
可以根据业务需要灵活写入各种数据。

4. 设计 Watch
1. 节点启动或者重连后，尝试创建 Leader znode，尝试失败则 watch Leader znode；
2. 当收到 Leader znode 被删除的事件通知后，再次尝试创建 Leader znode，尝试成功则成为 Leader，失败则 watch Leader znode。



集群选举方案3 - 法官判决

极客时间 训练营

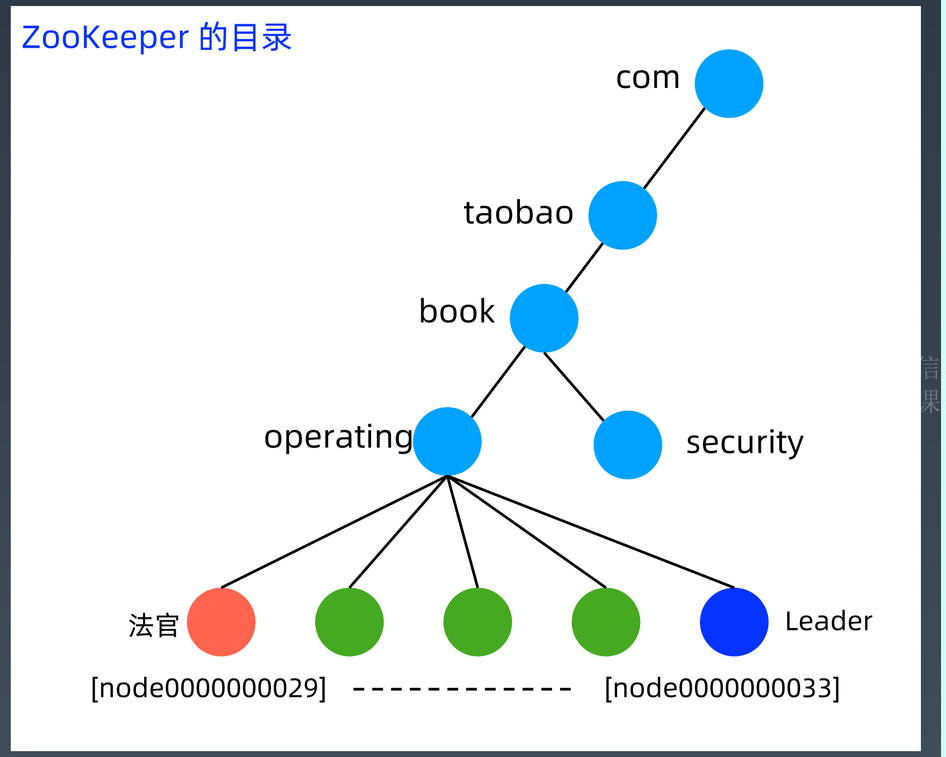
1. 设计 Path
集群共用父节点 parent znode，集群中的每个节点在 parent 目录下创建自己的 znode。

2. 选择节点类型
当 Leader 节点挂掉的时候，持有最小编号 znode 的集群节点成为“法官”，因此用 ephemeral_sequential 类型 znode。

3. 设计节点数据
可以根据业务需要灵活写入各种数据，例如写入当前存储的最新的数据对应的事务 ID。

4. 设计 Watch
见下一页。

集群选举方案3 - 方案示意



【节点设计】

- parent znode：图中的 operating，代表一个集群，选举结果写入到这里，例如：leader=server6。
- 法官 znode：图中的橙色 znode，最小的 znode，持有这个 znode 的节点负责选举算法/规则。
- 成员 znode：图中的绿色 znode，每个集群节点对应一个，在选举期间将选举需要的信息写入到自己的 znode。例如：Redis 存储集群，各个 slave 节点可以将自己存储的数据最新的 trxID 写入到 znode，然后法官节点将 trxID 最大的节点选为新的 Leader。

- Leader znode：图中的深蓝色 znode，集群里面只有一个，由法官选出来。

【设计 Watch 】

1. 节点启动或者重连后，在 parent 目录下创建自己的 ephemeral_sequntial znode，并 watch parent 目录；
2. 当 parent 目录有节点删除的时候，所有节点更新自己的 znode 里面和选举相关的数据；
3. “法官”节点读取所有 znode 的数据，根据规则或者算法选举新的 Leader，将选举结果写入 parent znode；
4. 所有节点 watch parent znode，收到变更通知的时候读取 parent znode 的数据，发现是自己则成为 Leader。

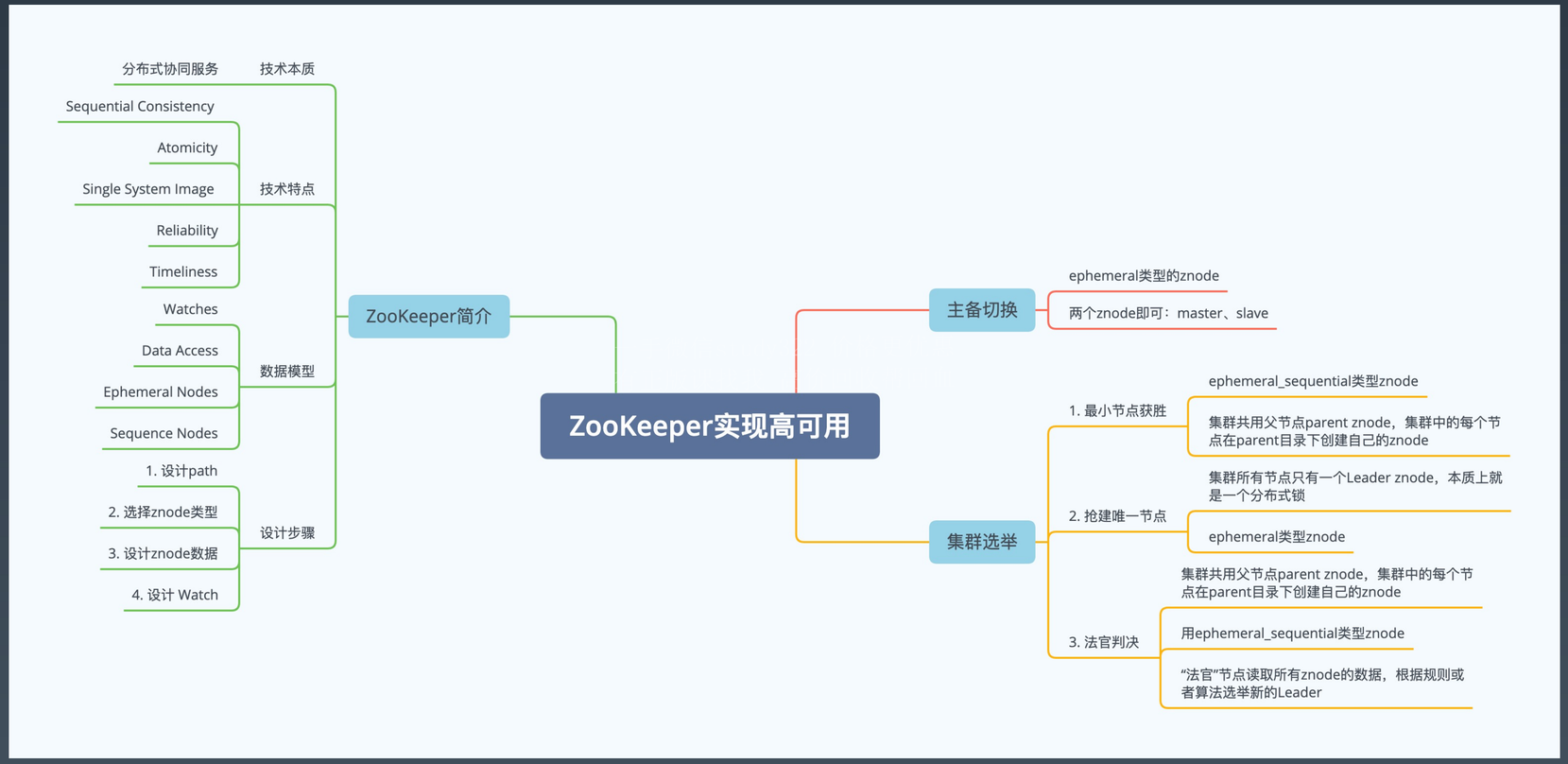
ZooKeeper 集群模式对比

极客时间 训练营

	实现复杂度	选举灵活性	应用场景
最小节点获胜	低	低	计算集群
抢建唯一节点	低	低	计算集群
法官判决	高	高，可以设计满足业务需求的复杂选举算法和规则	存储集群

为什么计算集群不用法官判决而存储集群用法官判决？

本节思维导图


ZooKeeper实现高可用
ZooKeeper简介
技术本质
分布式协同服务
技术特点
Sequential Consistency
Atomicity
Single System Image
Reliability
Timeliness
数据模型
Watches
Data Access
Ephemeral Nodes
Sequence Nodes
设计步骤
1. 设计path
2. 选择znode类型
3. 设计znode数据
4. 设计 Watch
主备切换
ephemeral类型的znode
两个znode即可：master、slave
集群选举
1. 最小节点获胜
ephemeral_sequential类型znode
集群共用父节点parent znode，集群中的每个节点在parent目录下创建自己的znode
2. 抢建唯一节点
集群所有节点只有一个Leader znode，本质上就是一个分布式锁
ephemeral类型znode
3. 法官判决
集群共用父节点parent znode，集群中的每个节点在parent目录下创建自己的znode
用ephemeral_sequential类型znode
“法官”节点读取所有znode的数据，根据规则或者算法选举新的Leader

随堂测验

极客时间 训练营

【判断题】

1. ZooKeeper 的分布式一致性协议是 ZAB，ZAB 就是 Paxos 算法的落地版本。
2. ZooKeeper 实现主备切换的时候，znode 的 path 可以固定为 master 和 slave。
3. 最小节点获胜的选举方案中，应该用 ephemeral 类型的 znode。
4. 法官判决选举方案中，法官不应该选自己成为 Leader。
5. 法官判断选举方案功能强大，无论计算集群还是存储集群都应该优先采用。

【思考题】

基于 ZooKeeper 的选举方案，对比 Redis Sentinel，功能上有什么差异？

Q&A

茶歇时间

极客时间 训练营

八卦，趣闻，内幕……

THANKS