# 架构实战营模块8 - 第5课

# 常见集群算法解析



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 Gossip 协议的基本实现和技术本质
2. 掌握 Bully 算法的基本原理
3. 掌握 Raft 算法的技术本质和应用

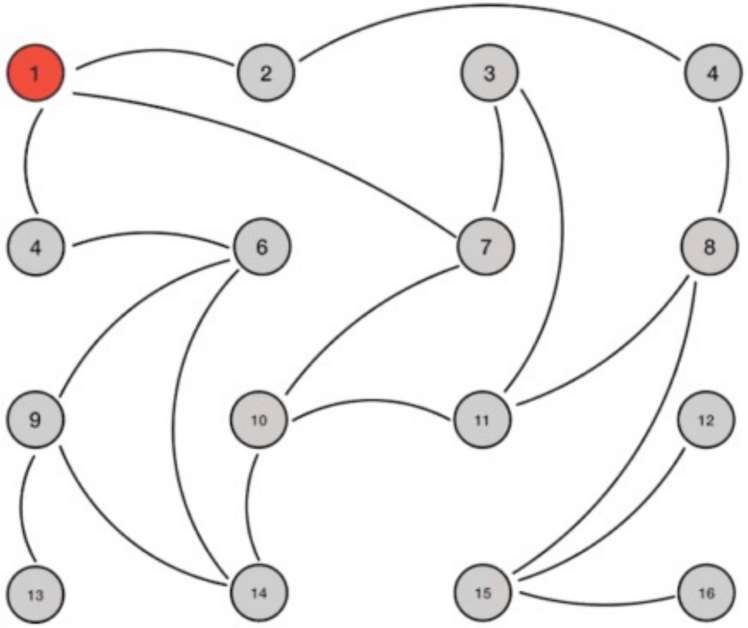


最强的不一定是最好的！

# 目录

# 1. Gossip 协议

# Gossip 协议简介




【定义】
Gossip protocol，又叫 Epidemic Protocol （流行病协议），也叫“流言算法”、“疫情传播算法”等，其名称已经形象的说明了算法的原理和工作方式。
学习链接

【应用场景】
1. 分布式网络，无集中管理节点；
2. 节点间点对点传播信息。

【典型应用】
1. P2P；
2. Bitcoin；
3. Apache Cassandra、Redis cluster。

开源实现：memberlist

# Gossip 协议优缺点

极客时间 训练营

优点

简单

1. 扩展性：网络节点可任意增加和修改；
2. 容错性：无中心节点，任意节点宕机不影响协议运行；
3. 去中心化：任意节点都可以发送消息。

缺点

最终一致性

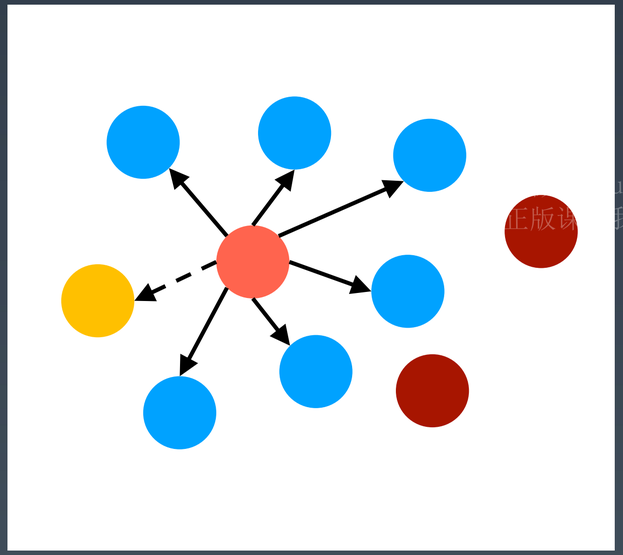
1. 需要花费一定时间达到最终一致性；
2. 消息冗余；
3. 不适合超大规模集群（超过1000）；
4. 恶意节点传播垃圾信息。



在达到最终一致性之前，集群状态不一致怎么办？

# Gossip 模式1 - Direct Mail

直邮模式：
通知所有邻居更新信息，邻居节点收到消息后不会转发。



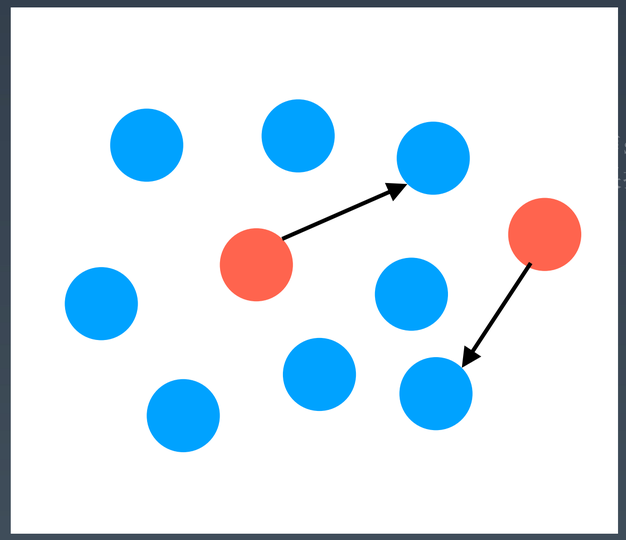
【优点】
简单。

【缺点】
1. 难以达到最终一致性
 - 节点消息可能丢失（图中黄色节点）；
 - 节点可能并没有连接（图中深红色节点）。
2. 容错性低
 - 需要缓存发送失败的消息。
3. 种子节点压力大

【应用场景】
社交网络（朋友的朋友并不一定是你的朋友）。

# Gossip 模式2 - Anti-Entropy

反熵模式：
集群中的节点，每隔一段时间随机选择1个节点，互相交换所有数据，然后进行同步，消除数据不一致。



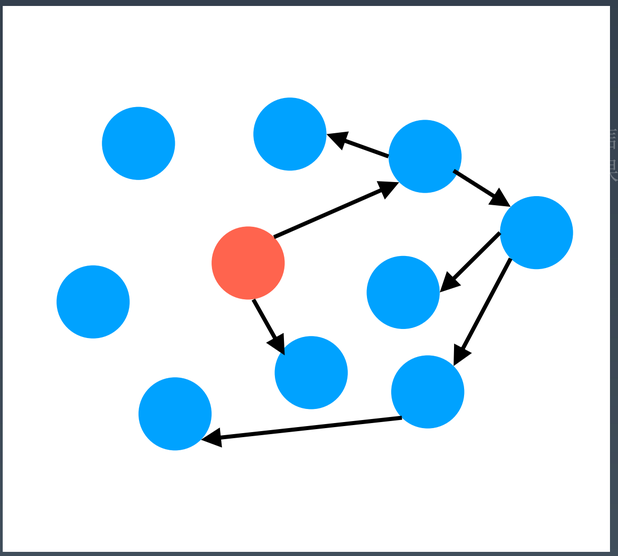
【优点】
最终一致性。

【缺点】
1. 信息同步的成本高
 - checksum；
 - updated list.
2. 达到最终一致性的耗时较长

【应用场景】
节点数量不多，实现最终一致性，例如存储系统多副本一致性。

# Gossip 模式3 - Rumor mongering

谣言传播：
收到更新消息后，自己成为“受感染节点”，周期性的传播更新消息，如果发现其它节点已经知道了消息，则按照一定概率将自己变为 removed，不再传播消息。



【优点】
1. 最终一致性；
2. 传播信息少；
3. 达到一致性所需时间少。

【缺点】
1. 有一定概率可能不一致；
2. 节点数量不能太多，Redis 官方文档最大1000。参考链接

【应用场景】
节点经常变化的集群。

# 2. Bully 选举算法

# Bully 算法简介

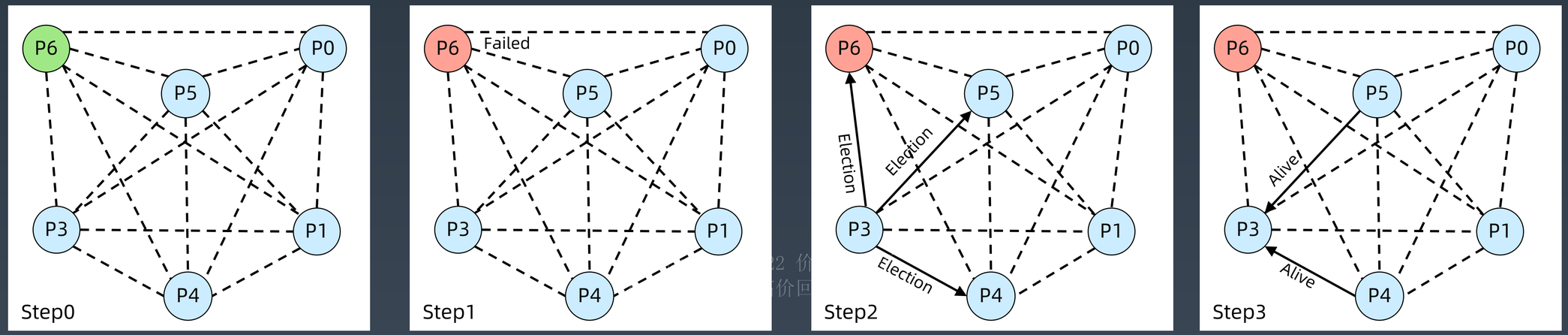
极客时间 训练营

【Bully 算法】
当一个进程发现协调者（或 Leader）不再响应请求时，就判定其出现故障，于是它就发起选举，选出新的协调者，即当前活动进程中进程号最大者。
Bully 的中文意思是“霸凌”，但实际实现时，找最小的节点也可以，关键点是“最”。

【关键假设】
1. 系统是同步的；
2. 进程在任何时候都可能失败，包括算法在执行的过程中；
3. 进程失败后停止工作，重启后重新工作；
4. 有失败监控者，它可以发现失败的进程；
5. 进程之间消息传递是可靠的；
6. 每一个进程都知道自己和其他每一个进程的 ID 以及地址。

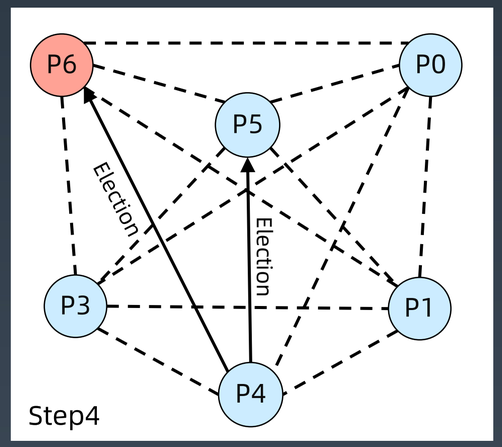
参考链接

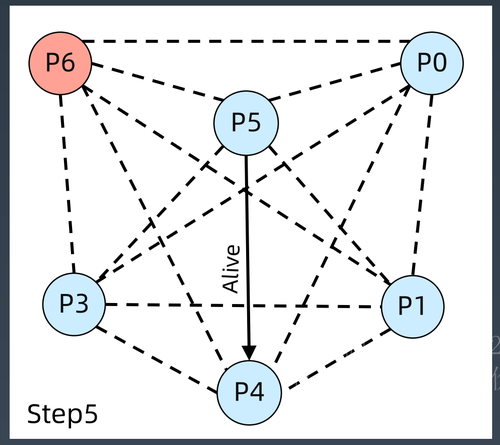
# Bully 算法选举过程(1/2)

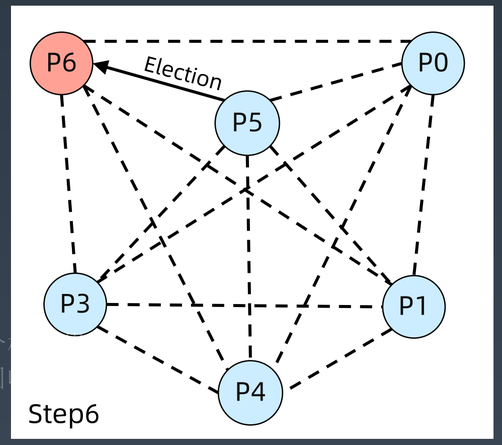


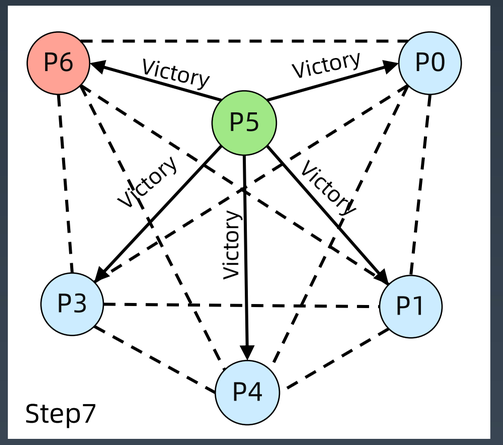

step0：初始状态，P6 为 Leader；
step1：P6 故障；
step2：P3 检测到 P6 故障，发起选举，向 P4、P5、P6 发送 Election 消息；
step3：P4、P5 回复 P3 Alive 消息，说明自己还活着，P3 退出选举，等待 Victory 消息。

# Bully 算法选举过程(2/2)




step4：P4 向 P5、P6 发送 Election 消息；
step5：P5 回复 P4 Alive 消息，P4 退出选举，等待 Victory 消息；
step6：P5 向 P6 发送 Election 消息；
step7：P5 未收到 Alive 消息，成功当选，向所有节点发送 Victory 消息，选举结束。

为什么还要向 P6 发送消息？

# 3. Raft 选举算法

# Raft 算法简介

极客时间 训练营

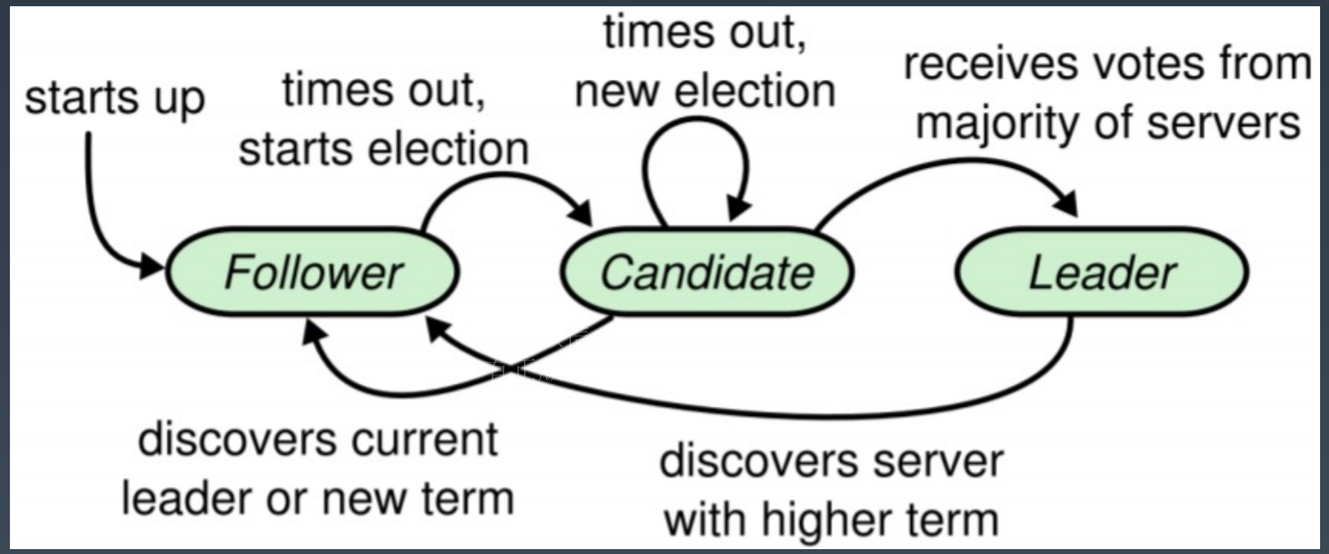
【Raft 算法】
Raft is a consensus algorithm that is designed to be easy to understand. It's equivalent to Paxos in fault-tolerance and performance. The difference is that it's decomposed into relatively independent subproblems, and it cleanly addresses all major pieces needed for practical systems.

【关键点】
1. 容易理解（Raft 作者说几个研究人员研究了1年还不是很明白 Paxos）；
2. 算法明确划分为选举、复制、安全三个子问题。

“There is only one consensus protocol, and that’s Paxos” - all other approaches are just broken versions of Paxos.
—— Mike Burrows, inventor of the Chubby service at Google

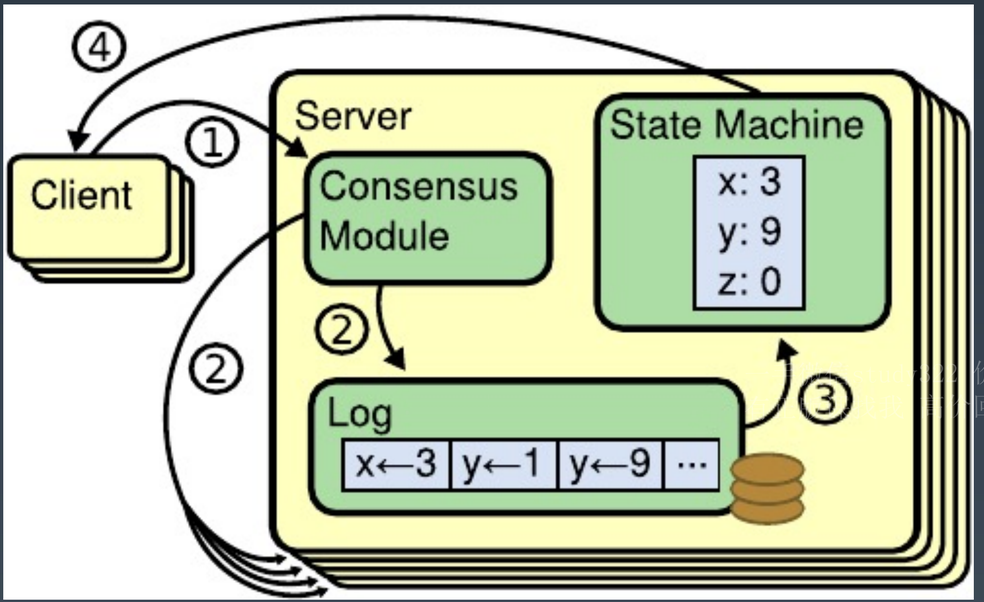
官网链接

# Raft 实现1 - leader 选举

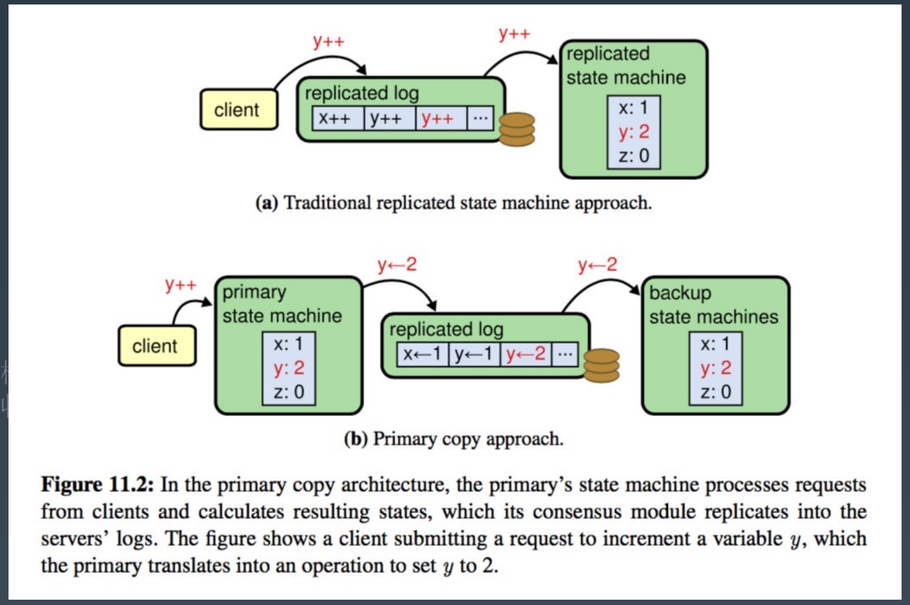



1. The client of the application makes requests only to and gets responses only from the leader server. etcd的实现
2. only be available when a leader has been successfully elected and is alive.

# Raft 实现2 - 日志复制




Replicated state machine：复制状态机，复制的是日志而不是数据；
典型代表：Raft，参考论文。




**(a)** Traditional replicated state machine approach.

**(b)** Primary copy approach.

**Figure 11.2:** In the primary copy architecture, the primary's state machine processes requests from clients and calculates resulting states, which its consensus module replicates into the servers' logs. The figure shows a client submitting a request to increment a variable $y$, which the primary translates into an operation to set $y$ to 2.

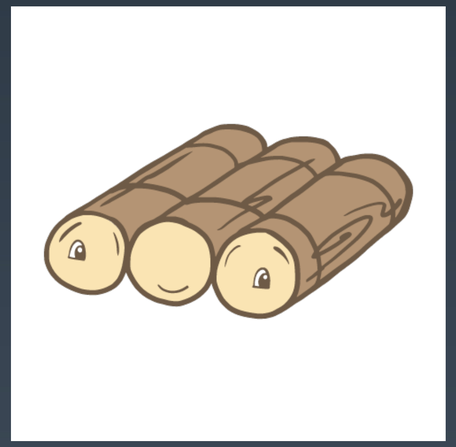
Primary-backup systems：主备复制，复制的是执行后的数据；
典型代表：ZooKeeper 的 ZAB，参考论文。

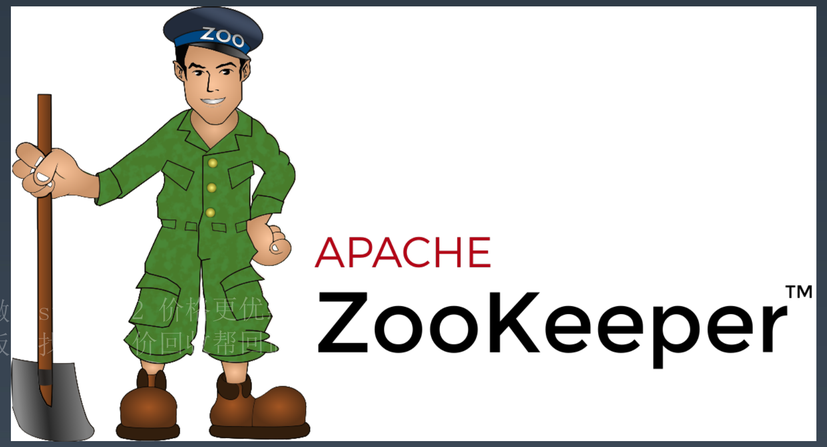
# Raft vs ZAB vs Paxos

极客时间 训练营

| | Raft | ZAB | Paxos |
|---|---|---|---|
| 分布式一致性 | 弱于 Paxos | 弱于 Paxos | 最强 |
| 复制方式 | Replicated state machine | Primary-backup | 每次都投票 |
| 读写方式 | 读写 Leader，Follower 不接受请求 | 读写 Leader，读 Follower | 任意节点都可以读写 |
| 是否有Leader | 是，强 Leader | 是，强 Leader | 无，部分变种有 Leader，但只是协调作用 |
| 复杂度 | 比 Paxos 简单 | 比 Paxos 简单 | 复杂 |
| 特殊场景 | 选举期间不能服务 | 选举期间不能服务 | Livelock，参考链接 |

# Raft vs ZooKeeper








1. 如果你想内嵌分布式选举或者一致性功能，或者基于业务特性做一些小调整，选择 Raft，例如 MongoDB、etcd 等；
2. 如果你想实现分布式选举或者一致性，但是不想自己去实现协议代码，选择 ZooKeeper，例如 HDFS、Cassandra 等；
3. 如果你不确定，请选择 ZooKeeper。

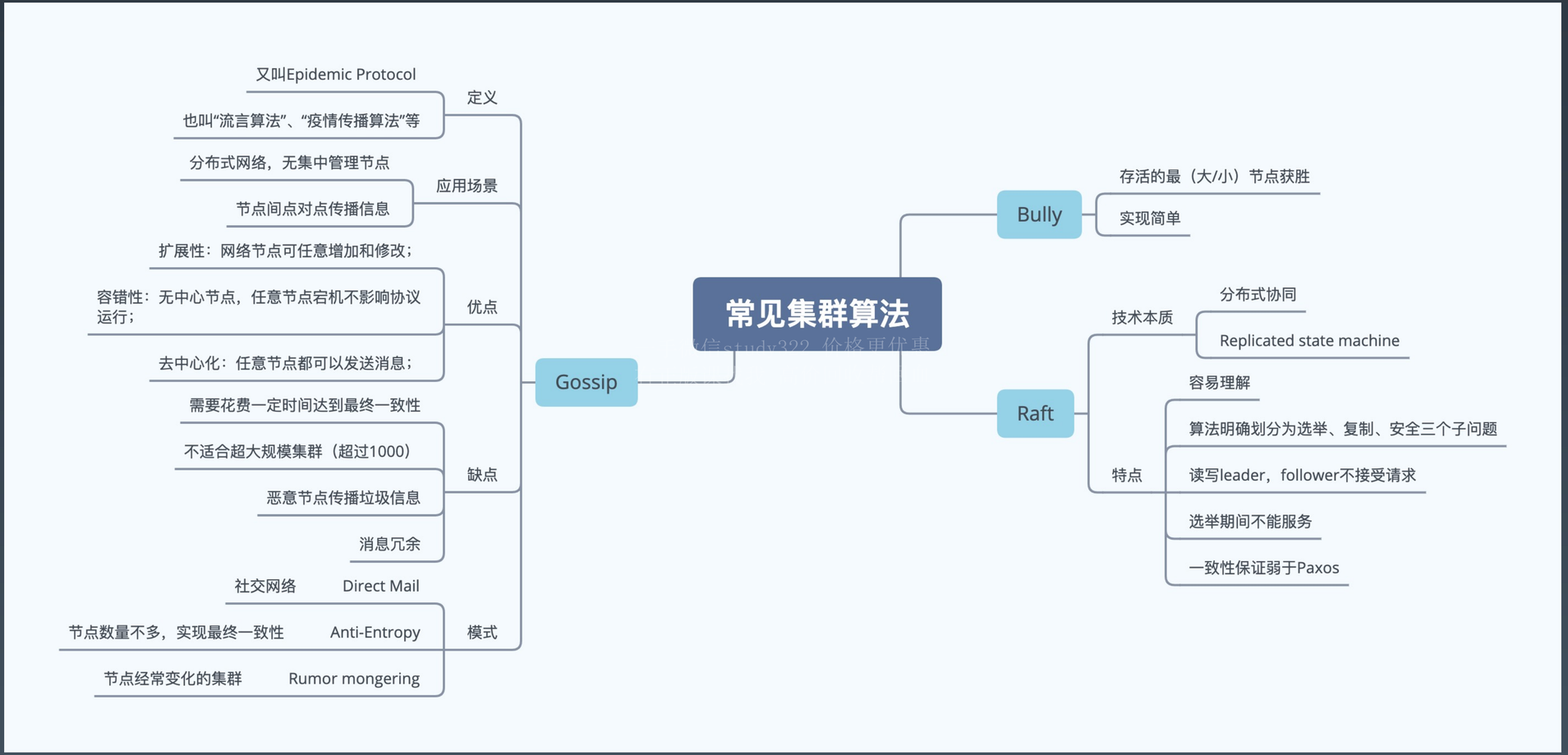
# Raft 的资源

极客时间 训练营

| Stars | Name | Primary Authors | Language | License | Replication? | Persistence? | Changes? | Compaction? |
|---|---|---|---|---|---|---|---|---|
| 9,373★ | TiKV | Jay, ngaut, siddontang, tiancaiamao | Rust | Apache-2.0 | Yes | Yes | Yes | Yes |
| 36,059★ | etcd/raft | Blake Mizerany, Xiang Li, Yicheng Qin | Go | Apache-2.0 | Yes | Yes | Yes | Yes |
| 4,759★ | hashicorp/raft | Armon Dadgar | Go | MPL-2.0 | Yes | Yes | Yes | Yes |
| 2,668★ | braft | Zhangyi Chen, Yao Wang | C++ | Apache-2.0 | Yes | Yes | Yes | Yes |
| 3,643★ | dragonboat | Lei Ni | Go | Apache-2.0 | Yes | Yes | Yes | Yes |
| 24,717★ | RethinkDB | | C++ | Apache-2.0 | Yes | Yes | Yes | Yes |
| 2,336★ | SOFAJRaft | Boyan, Jiachun | Java | Apache-2.0 | Yes | Yes | Yes | Yes |
| 1,405★ | Kudu | David Alves, Todd Lipcon, Mike Percy | C++ | Apache-2.0 | Yes | Yes | Yes | Yes |
| 2,249★ | go-raft | Ben Johnson , Xiang Li, CoreOS) | Go | MIT | Yes | Yes | No | Yes |
| 4,369★ | hazelcast-raft | Mehmet Dogan, Ensar Basri Kahveci | Java | Apache-2.0 | Yes | Yes | Yes | Yes |

详细参考官方信息：Where can I get Raft ?

# 本节思维导图


常见集群算法
Gossip
定义
又叫Epidemic Protocol
也叫“流言算法”、“疫情传播算法”等
应用场景
分布式网络，无集中管理节点
节点间点对点传播信息
优点
扩展性：网络节点可任意增加和修改；
容错性：无中心节点，任意节点宕机不影响协议运行；
去中心化：任意节点都可以发送消息；
缺点
需要花费一定时间达到最终一致性
不适合超大规模集群（超过1000）
恶意节点传播垃圾信息
消息冗余
模式
社交网络
Direct Mail
节点数量不多，实现最终一致性
Anti-Entropy
节点经常变化的集群
Rumor mongering
Bully
存活的最（大/小）节点获胜
实现简单
Raft
技术本质
分布式协同
Replicated state machine
特点
容易理解
算法明确划分为选举、复制、安全三个子问题
读写leader，follower不接受请求
选举期间不能服务
一致性保证弱于Paxos

# 随堂测验

极客时间 训练营

## 【判断题】

1. Gossip 协议实现简单，适合对一致性要求不高的集群。
2. Gossip 的反熵模式可以保证集群最终一致性，因此应该优先采用。
3. Bully 算法必须挑选集群中节点 ID 最大的作为 Leader。
4. Raft 算法一致性强度不如 Paxos，但是实现要简单。
5. Raft 算法和 ZAB 算法的 Follower 都支持处理客户端请求。



## 【思考题】

为什么 Paxos 是最好分布式协同算法，但应用却不广？

# Q&A

极客时间 训练营

# 茶歇时间

极客时间 训练营

八卦，趣闻，内幕……

# THANKS