百度云 Cockroach DB

CockroachDB 中国社区大会

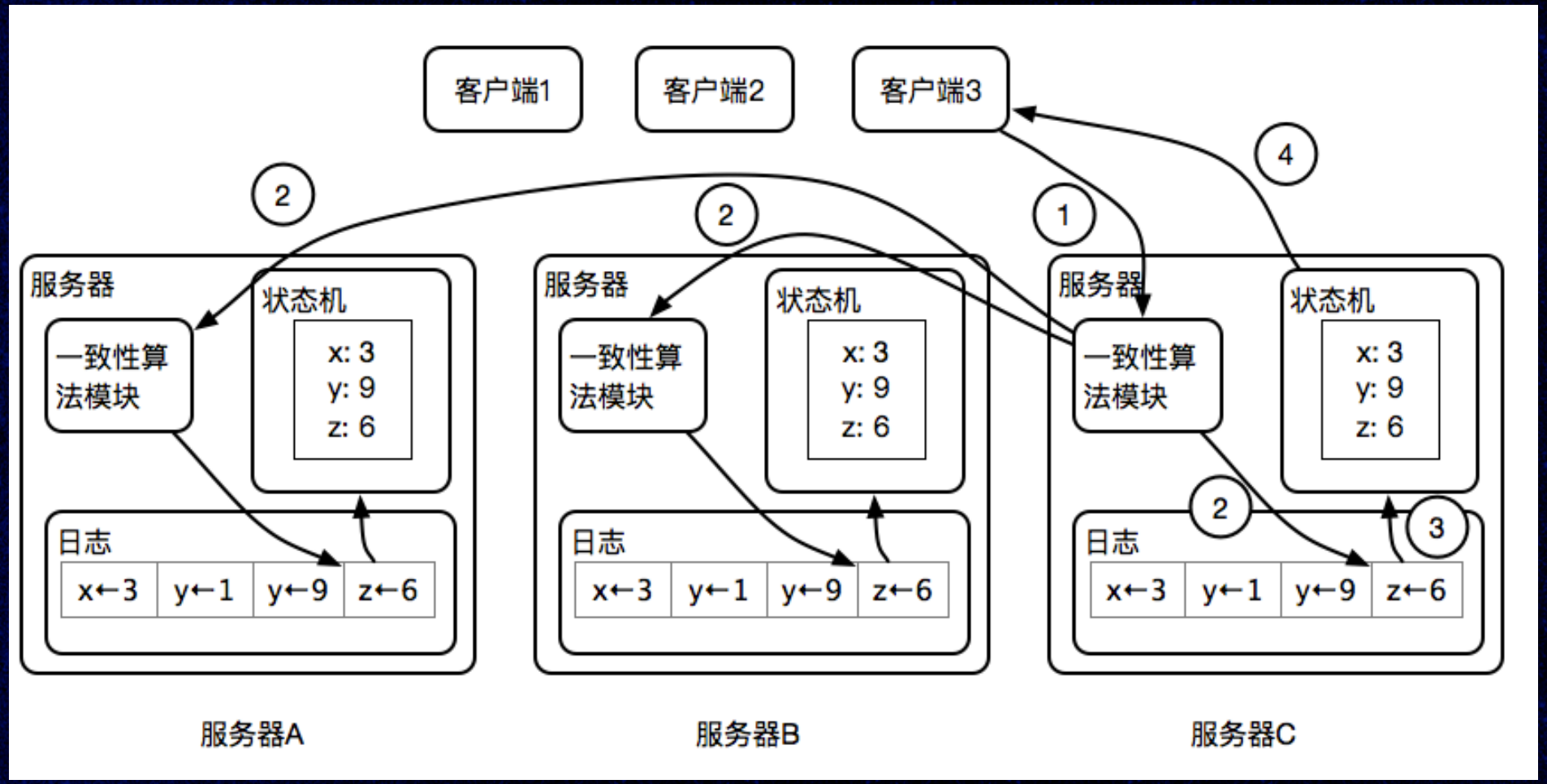
Raft inside CockroachDB

么敬国
yaojingguo@xdf.cn

大纲

- Raft简介
- Etcd Raft简介
- CockroachDB的Raft集成
- CockroachDB的Raft优化

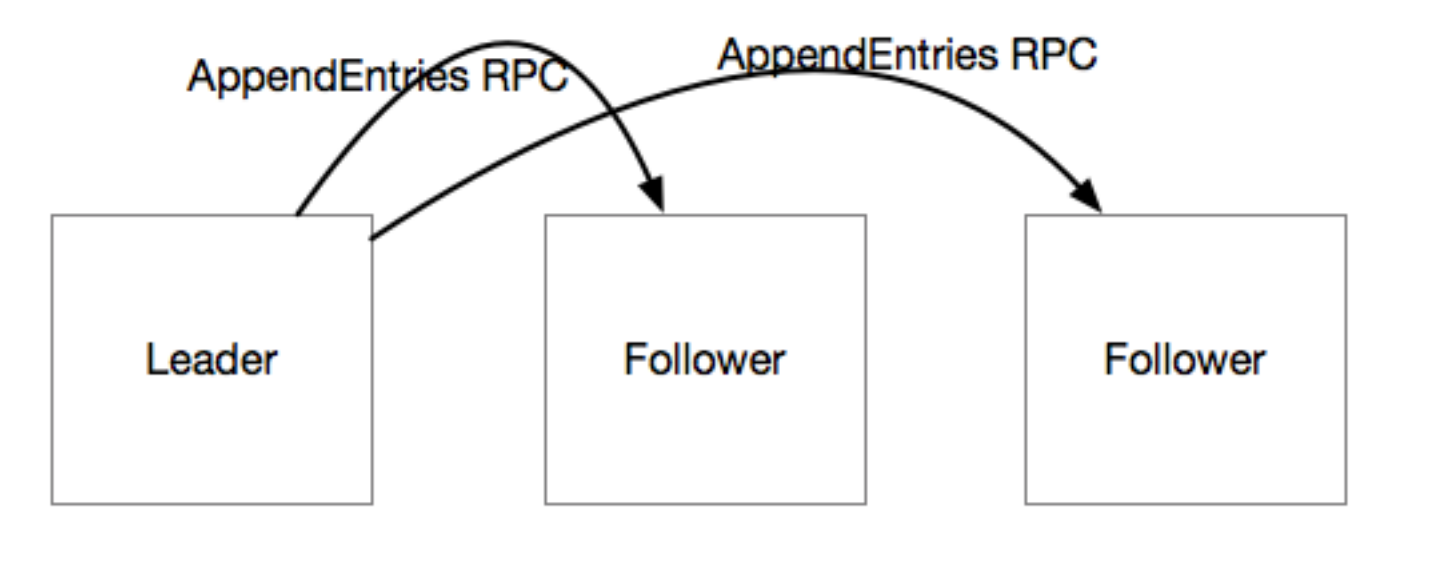
复制状态机


客户端1
客户端2
客户端3
服务器
一致性算法模块
状态机
x: 3
y: 9
z: 6
日志
x←3
y←1
y←9
z←6
服务器A
服务器B
服务器C

Raft基本概念

Raft算法是目前广泛使用的分布式数据一致性算法。一个Raft集群包括若干节点。节点可以处于以下三种状态：leader 、follower 和candidate。Leader处理来自客户端的请求。Follower不会主动发起任何操作，只会被动的接收leader和candidate的请求。Raft论文提到了3个最主要的RPC： AppendEntries、RequestVote和InstallSnapshot。

- AppendEntries：在一个Raft集群中有Leader的情况下，Leader不断的发送AppendEntries RPC给follower。



Raft RPC

- RequestVote： 如果follower在一个election timeout周期内没有收到心跳信息，就认为目前集群中没有leader。此时follower会发送RequestVote 给集群中其他的节点。
- InstallSnapshot：Snapshot一方面可以回收不断增长的日志存储空间，一方面可以快速更新一个follower的状态。

Etcd简介

- Etcd是一个高可用的key-value存储，使用Raft算法实现整个系统的高可用。
- Etcd的Raft算法实现可以当做一个库被其他系统使用。 Etcd的Raft库是目前使用最为广泛的Raft算法实现。
- 有两种接入Etcd Raft库的方式：
  - Node接口
  - RawNode接口

Node接口

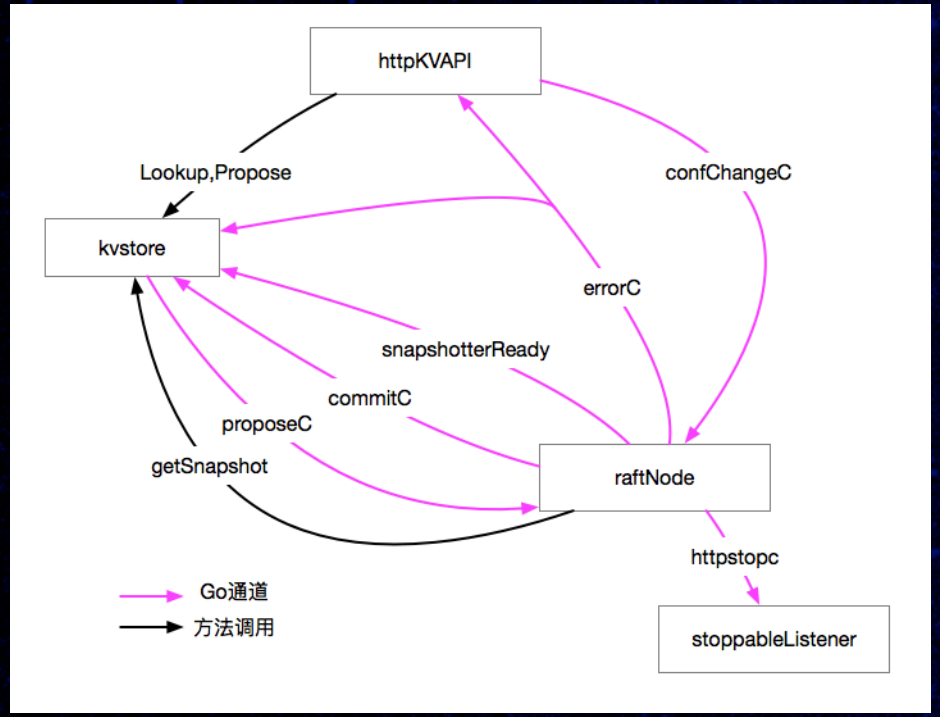
Node接口使用Go通道做了并发控制，使用起来相对简单，只需要对以下三部分部分进行配置：

- Transporter interface: 用来进行Raft节点间网络通信。Etcd提供了一个基于HTTP协议的实现rafthttp.Transport。
- Storage inferface: 用来存储Raft日志和Snapshot。 Etcd提供了一个基于内存的实现raft. MemoryStorage 。因为MemoryStorage是基于内存的，需要和wal.WAL配合使用。 wal.WAL是一个write ahead log。
- Snapshotter struct: 用来保存Snapshot。

raftexample

Raftexample是一个展示如何使用Node接口的例子，实现了一个暴露HTTP接口的高可用key-value存储。

- **proposeC**: kvstore用来把key-value更新发送给Raft。
- **commitC**: Raft用来把已经提交的key-value更新发送给kvstore。
- getSnapShot: kvstore把Snapshot发送给Raft。
- snapshotterReady: Raft用来把Snapshot发送给kvstore。



RawNode struct

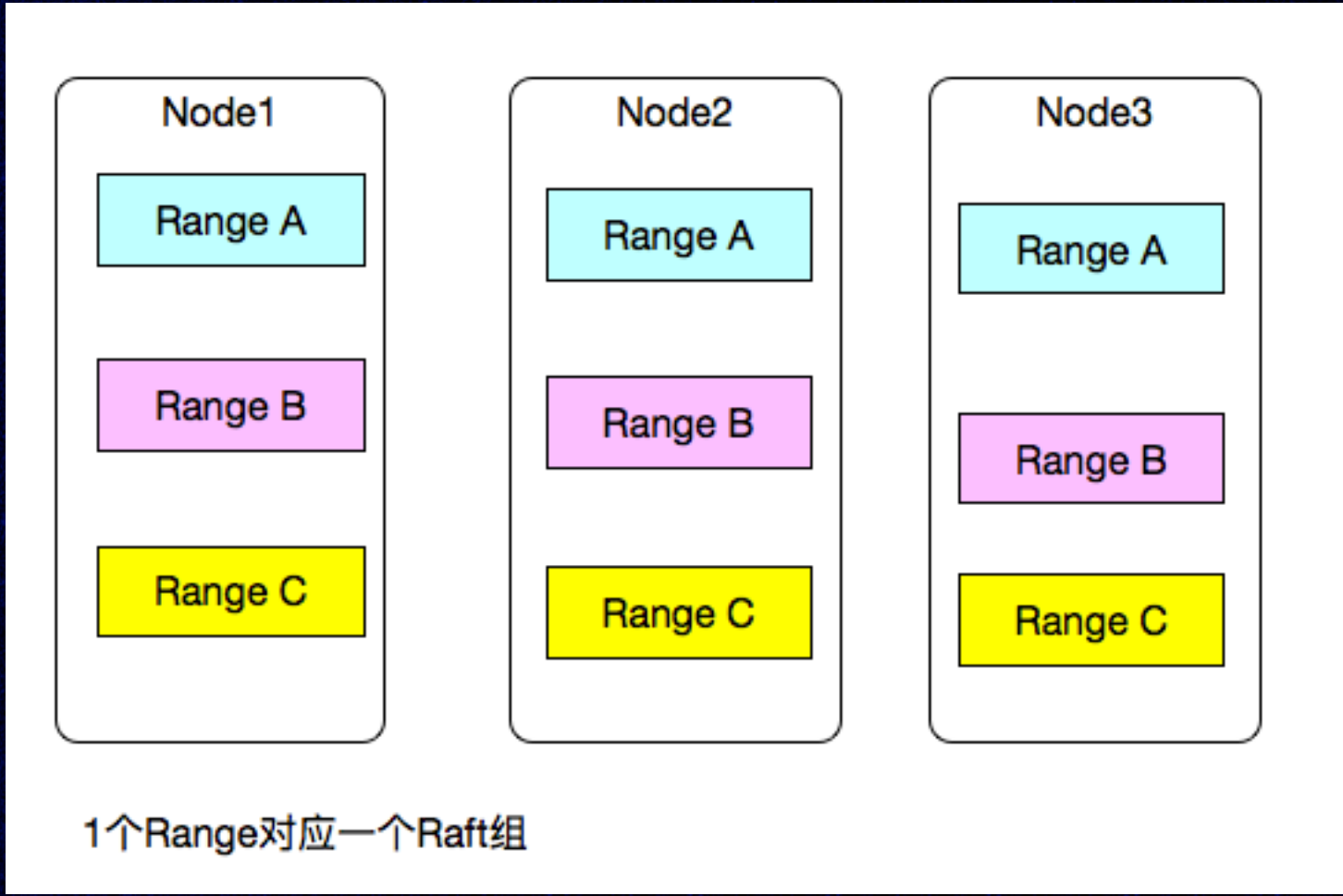
RawNode实现了Node接口的部分方法，没有使用Go通道来进行并发控制，由使用方自己来做并发控制。
CockroachDB使用到的RawNode方法：

- **Step()**:把通过gRpc接收的Raft消息发送给Raft。
- **HasReady()**：检查是否有Ready需要处理。
- **Ready()**：返回需要处理的Ready。
- **Advance()**:通知Raft已经处理完了Ready信息。
- **Propose()**：让Raft发送AppendEntries RPC。
- **Campaign()**：当前节点进入candidate状态，发起投票。
- **Tick()**：Raft的逻辑时钟数+1。
- **TransferLeader()**：迁移leader角色。
- **Status()**：返回Raft组的当前状态。
- **WithProgress()**：用一个回调函数遍历Raft组中各个节点的Progress。
- **ReportUnreachable()**：汇报最近一次消息发送不能触达的节点。

CockroachDB的Raft集成

- Storage: 基于RocksDB的实现。因为RocksDB本身就是durable的，不需要额外的WAL。
- Snapshot: 使用的RocksDB的GetSnapShot API标记SnapShot。然后把SnapShot上的key-value使用gRpc的steaming机制发送给follower。
- Transport: 没有使用Raft本身的Transporter接口，完全是自己基于gRpc的实现。

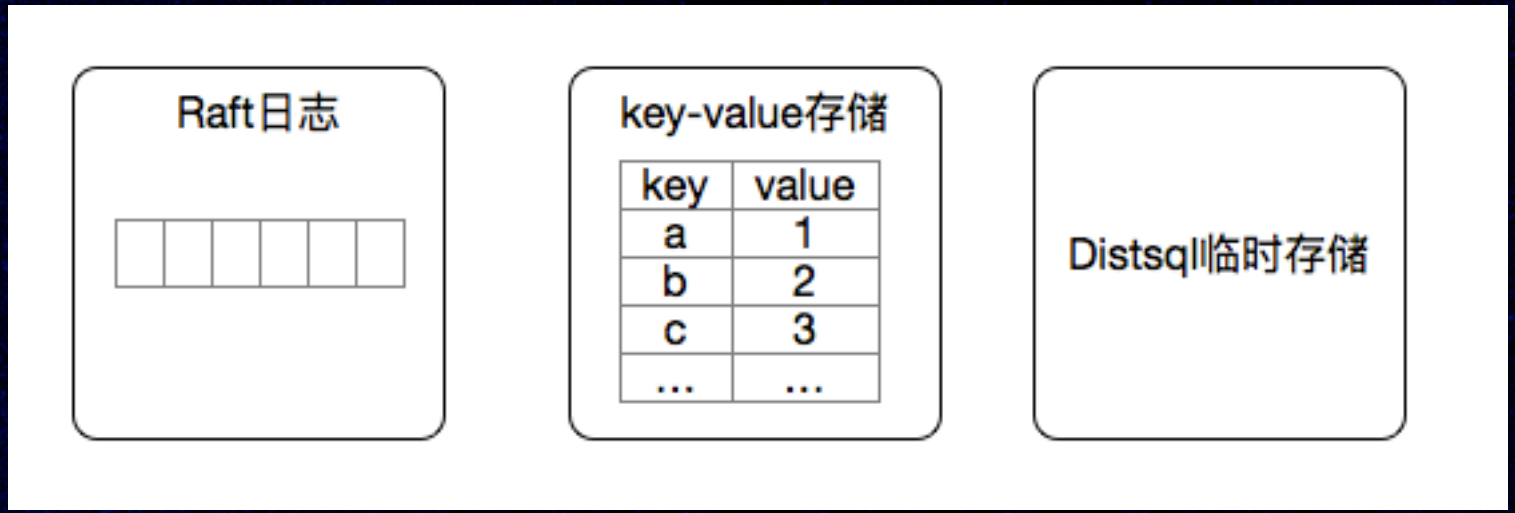
CockroachDB复制存储层



CockroachDB存储层

CockroachDB存储层由3个RocksDB组成：

- 一个用来保存Raft日志。
- 一个用来保存Key-value对，对应Raft算法中提到的state machine。
- 一个用来保存Distsql运行中使用的临时数据。



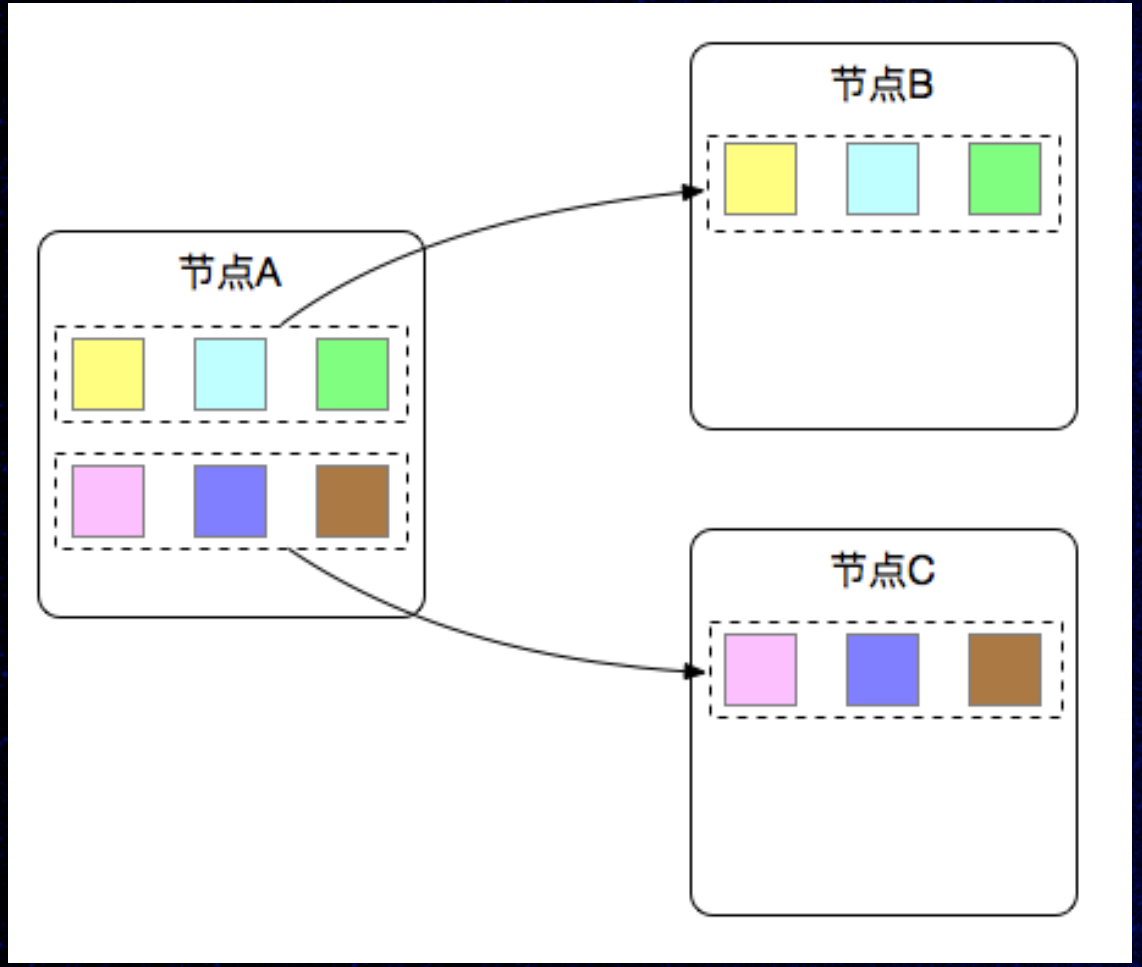
CockroachDB的Raft优化

1. MultiRaft
2. Quiescent Raft
3. Proposer Evaluated KV
4. Raft Proposal Sideloading for SSTable ingestion

MultiRaft

Raft算法使用心跳来维持Leader的角色。像Etcd这样的系统，整个系统只有一个Raft组，这些定期发送的心跳不会消耗太多计算资源。但是CockroachDB的一个节点上就可能有成千上万的Raft组，心跳会消耗大量计算资源。

MultiRaft就是把需要发送一个节点的心跳信息合并到一起，作为一个RPC请求发送给Raft组的其他副本，从而减少计算资源的消耗。

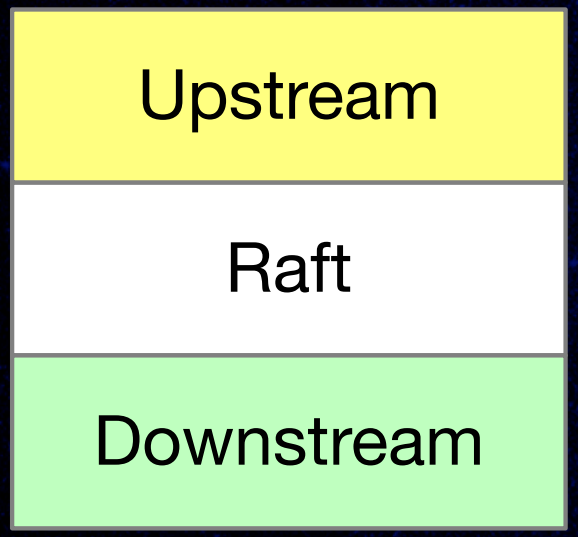


静默Raft

- CockroachDB延迟加载各个Range的Raft组。
- 静默Range
  - 如果一个Range的Raft组长时间没有事情可做，可以把Raft组转换到静默状态。一个静默的Raft组不会有任何网络传输。此时Raft组的Tick处于停止状态，所以不需要发送MsgHeartbeat来维持leader状态。
  - 一个Range的主副本通过把一条特殊的MsgHeartbeat消息来进行Raft静默。
  - 发送到到任何一个静默Raft副本会唤醒整个Raft组。

RFC: Proposer Evaluated KV

主副本是唯一向Raft提交命令的副本。 但是每个副本要单独应用这些命令。但是因为命令的执行在不同的副本上都会产生的结果，所以在其他副本上的执行是在做无用功。这个RFC主副本上执行的结果保存下来，然后在应用Raft日志命令时直接应用到其他副本。把大量Raft的downstream计算，移到Raft的Upstream。



举例

- Split操作：Split一个Range要进行大量的统计数据计算。通过这个RFC，这些统计计算只需要在主副本上执行一次，然后在应用Raft命令到各个副本时直接使用这些统计数据。
- Put操作：即使是一个简单的Put操作也需要进行一些时间戳和并发控制处理。通过这个RFC，这些处理只需要在主副本上执行一次，在应用Raft命令到各个副本时涉及的只是RocksDB的简单写操作。

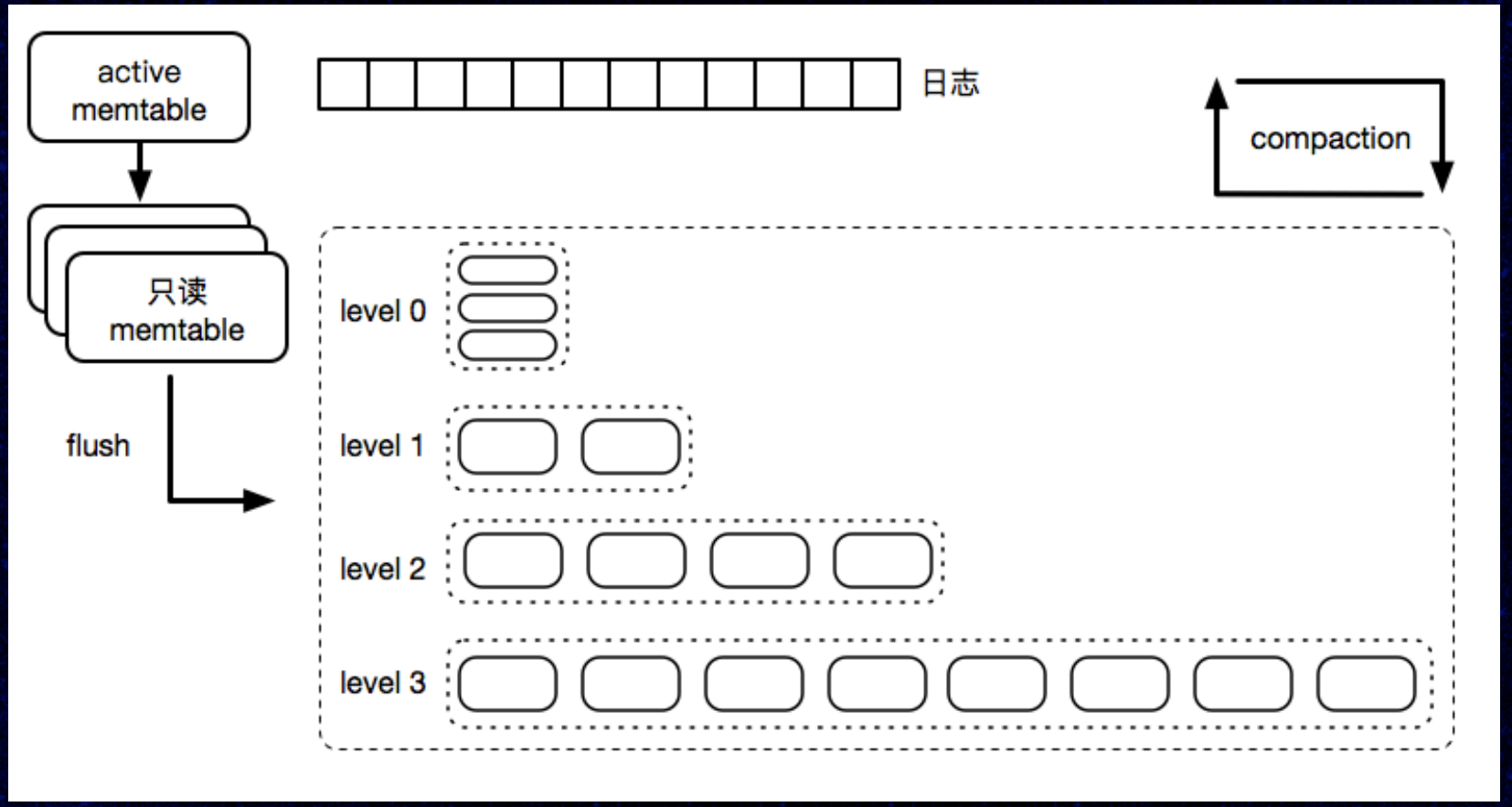
实现RESTORE命令的笨方法

使用Raft把WriteBatch分发各个Raft副本，这种办法有以下两个问题：

- 把WriteBatch写入Raft日志会涉及写入放大。
- 把WriteBatch应用到Key-value存储也会涉及写入放大。

理想的办法是直接把要恢复的数据直接加载到Key-value存储的RocksDB。

SSTable Ingestion to RocksDB



RFC: Raft Proposal Sideloading for SSTable ingestion

在一个节点收到SSTable ingestion命令时，不把SSTable放到Raft日志里面。一个Raft副本在接到SSTable ingestion命令做以下两件事情：

- 把命令里面的SSTable文件内容保存本地的文件系统。
- 把命令里面的metadata保存到Raft日志。

在应用SSTable ingestion命令时，依据Raft日志里面的metadata直接把上面保存的文件加载到RocksDB。

参考

- https://github.com/etcd-io/etcd/blob/master/contrib/raftexample/doc.go
- https://github.com/etcd-io/etcd/tree/master/contrib/raftexample
- https://github.com/cockroachdb/cockroach/blob/master/docs/RFCS/20160420_proposer_evaluated_kv.md
- https://github.com/cockroachdb/cockroach/blob/master/docs/RFCS/20170601_raft_sstable_sideloading.md
- https://www.cockroachlabs.com/blog/scaling-raft/
- https://github.com/cockroachdb/cockroach/pull/9315
- https://github.com/cockroachdb/cockroach/blob/master/docs/design.md
- https://github.com/cockroachdb/cockroach

Thanks

2019年1月12日

百度云 Cockroach DB