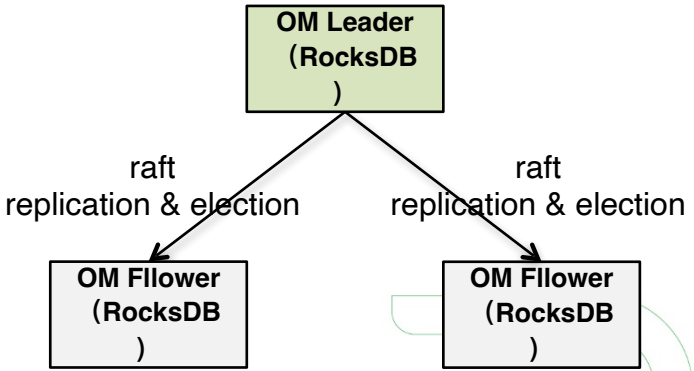
OM改造 分享

OM元信息扩展与OM HA稳定性改进

问题：

- om元信息存储在本机的rocksdb中，单个bucket的**元信息上限受限于单机磁盘的大小**，无法满足海量数据规模的元信息存储；
- 元数据基于raft复制到follower节点，需要引入snapshot等功能保证元数据在所有om节点的一致性，**复杂度偏高**

原生OM HA架构：

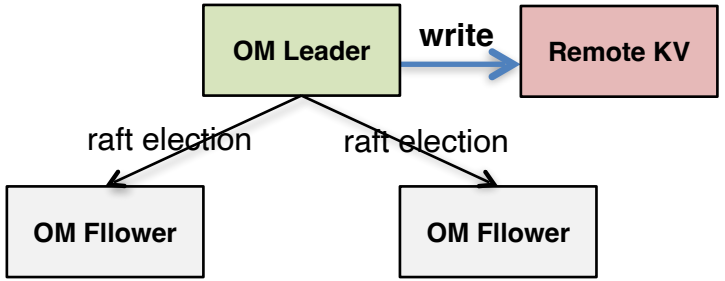


OM元信息扩展与OM HA稳定性改进

解决办法：

- 将om元信息存储在分布式kv中
- 改造om ha元信息复制部分，leader不再向follower同步数据
- 去掉复制snapshot的逻辑，元数据的一致性取决于分布式kv

新OM HA架构



OM元信息扩展 - 元信息写分布式kv

分布式kv的选型要满足以下要求：

- om的元信息通过double buffer flush将多次db操作合并为一次，数据库要**支持多表多行batch的原子性**
- 类似rocksdb，数据要以**key有序存储**，便于实现s3的object list等操作
- 规模可以水平扩展，依赖组件少，内部掌握程度高

使用Apache Cassandra：

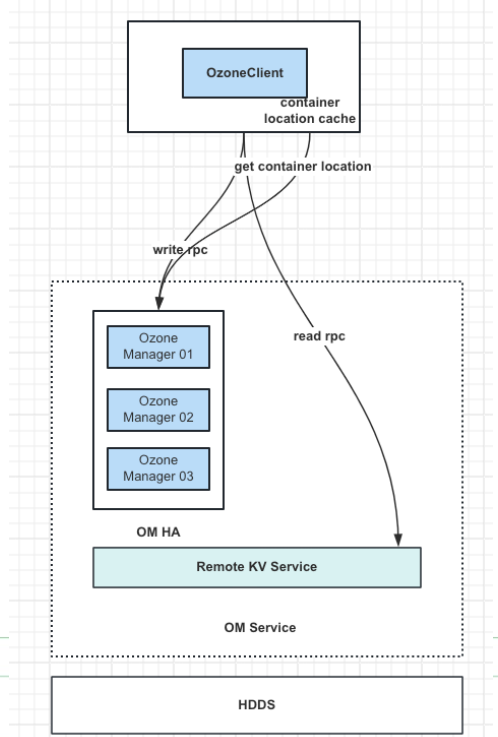
- 多种数据分区策略，支持ByteOrderedPartitioner与rocksdb排序规则相同
- 支持多表多行batch操作的原子性
- 无中心节点架构，可调节的一致性模型，规模可水平扩展，360内部掌握程度高
- 支持多数据中心部署，可保证元数据的异地容灾
- 方便与计算引擎结合，进行分布式的元信息分析（如spark-cassandra-connector）

OM元信息扩展 - observer read （元数据读写分离）

由于已经将元数据提取值分布式kv存储，不存在follower节点上的数据与leader上滞后或不一致的问题，可以实现读写分离，使用客户端直接连接分布式kv读取元信息。

- 客户端识别读请求
  - List/Head Object 直接从kv读取元信息
  - Get Object 从kv读取元信息+ container location从本地cache获取，未获取到则从om端获取

可解决元数据集中从om读取的问题，占用om的处理能力，过多的大文件的元数据获取，很容易使om gc产生压力。



表拆分

- 例如 openkey table 和 multirt info table，delete table 为临时表，有大量的写动作并立即删除
- 独立一个cassandra集群存储
- 避免大量compact 引起集群的时延不稳定，还可以分档访问压力

- 例如key table等其他表，单独一个cassandra集群

Cassandra instance1

"deletedTable";
"openKeyTable";
"multipartInfoTable";
"openFileTable";
"deletedDirectoryTable";

Cassandra instance2

"userTable";
"volumeTable";
"bucketTable";
"keyTable";
"s3SecretTable";
"dTokenTable";
"prefixTable";
"directoryTable";
"fileTable";
"transactionInfoTable";
"metaTable";
"tenantAccessIdTable";
"principalToAccessIdsTable";
"tenantStateTable";
"snapshotInfoTable";
"snapshotRenamedTable";
"compactionLogTable";

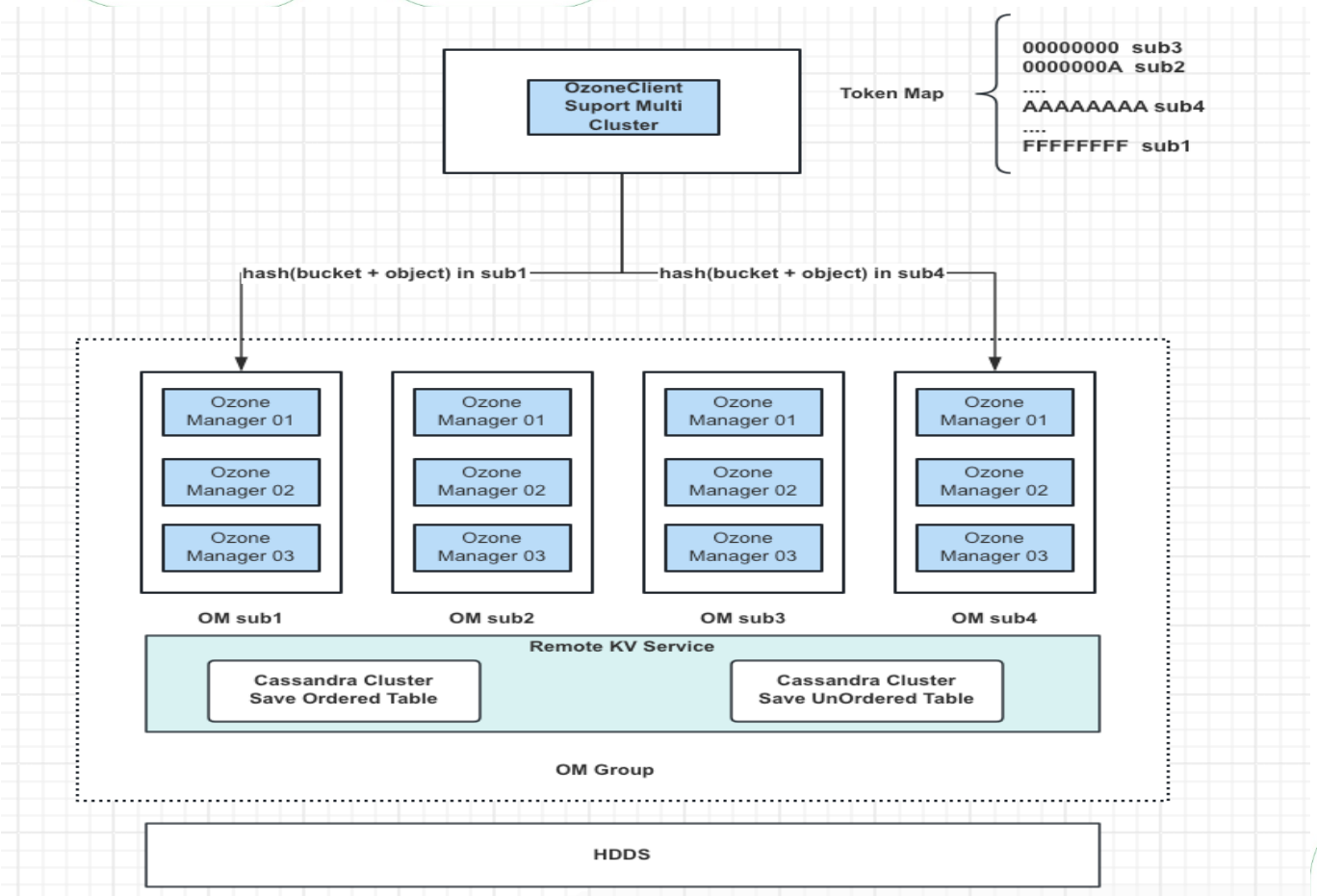
table cache 延时清理

写操作流程

- 客户端发送 createkey 请求到om端
- 传输数据
- 客户端发送 commitkey 请求到om端

每一个table，都会有一个table cache，将openkey table的table cache 配置延时清理，因为createkey请求中会写openkey，而在commitkey请求时会读取openkey，可以很大限度命中table cache，减少一次openkey的查数据库请求。

进一步扩展 客户端连接多组om，突破om单节点瓶颈



Thanks