Curve

Curve核心组件之 MDS

D I G I T A L S A I L

陈威

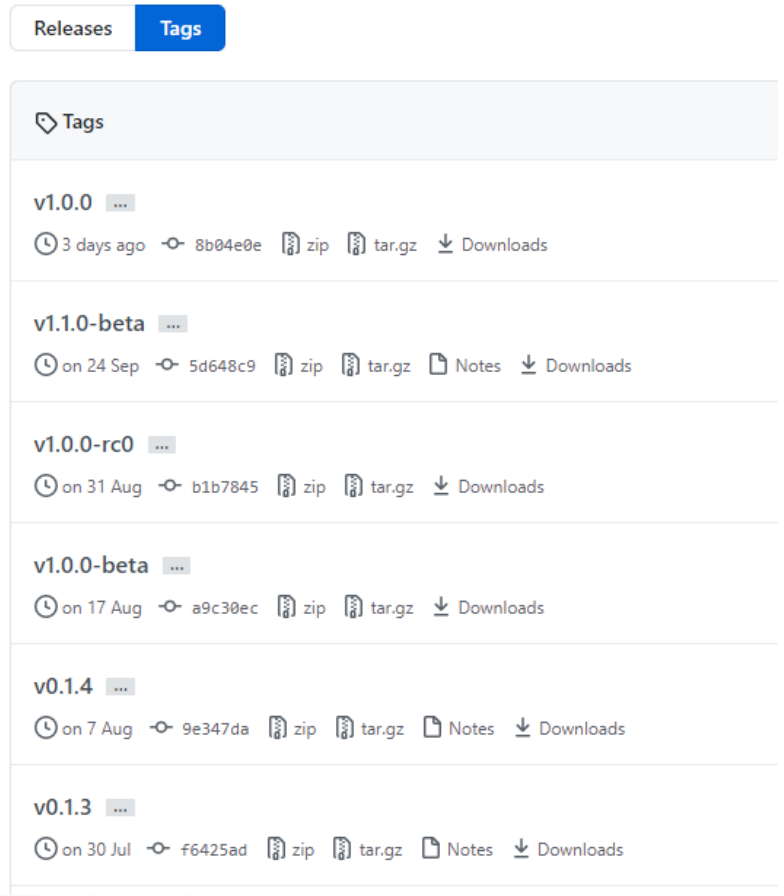
网易数帆存储团队

概述

Curve

Curve 是高性能、高可用、高可靠的分布式存储系统

- 高性能、低延迟
- 可支撑储场景：块存储、对象存储、云原生数据库、EC等
- 当前实现了高性能块存储，对接OpenStack和 K8s
  网易内部线上无故障稳定运行一年多
- 已开源
  - github主页： https://opencurve.github.io/
  - github代码仓库： https://github.com/opencurve/curve



Curve

DIGITAL SAIL

目录

基本架构

- 元数据节点 MDS

  管理元数据信息
  收集集群状态信息，自动调度

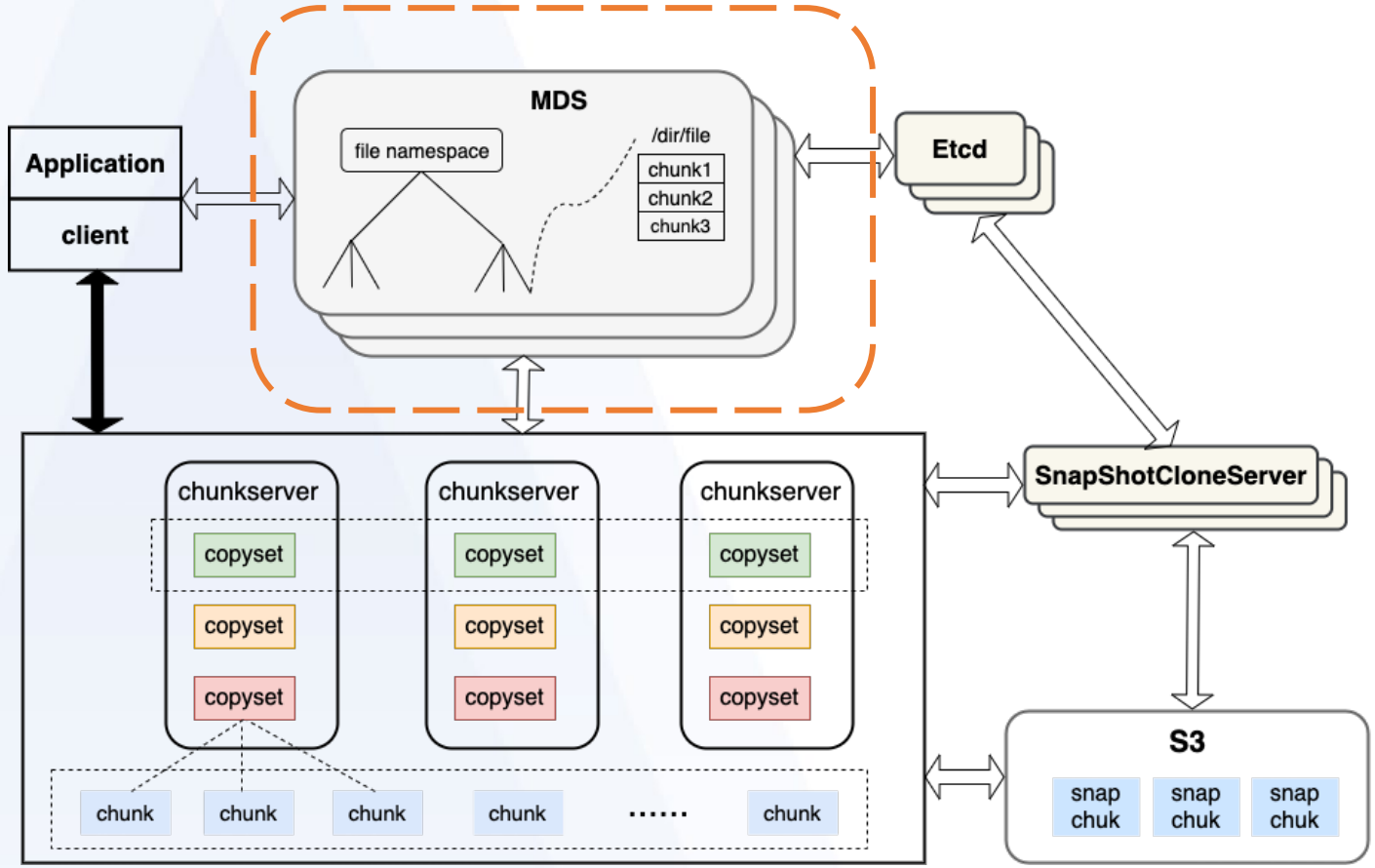
- 数据节点 Chunkserver

  数据存储
  副本一致性

- 客户端 Client

  对元数据增删改查
  对数据增删改查

- 快照克隆服务器



MDS各个组件

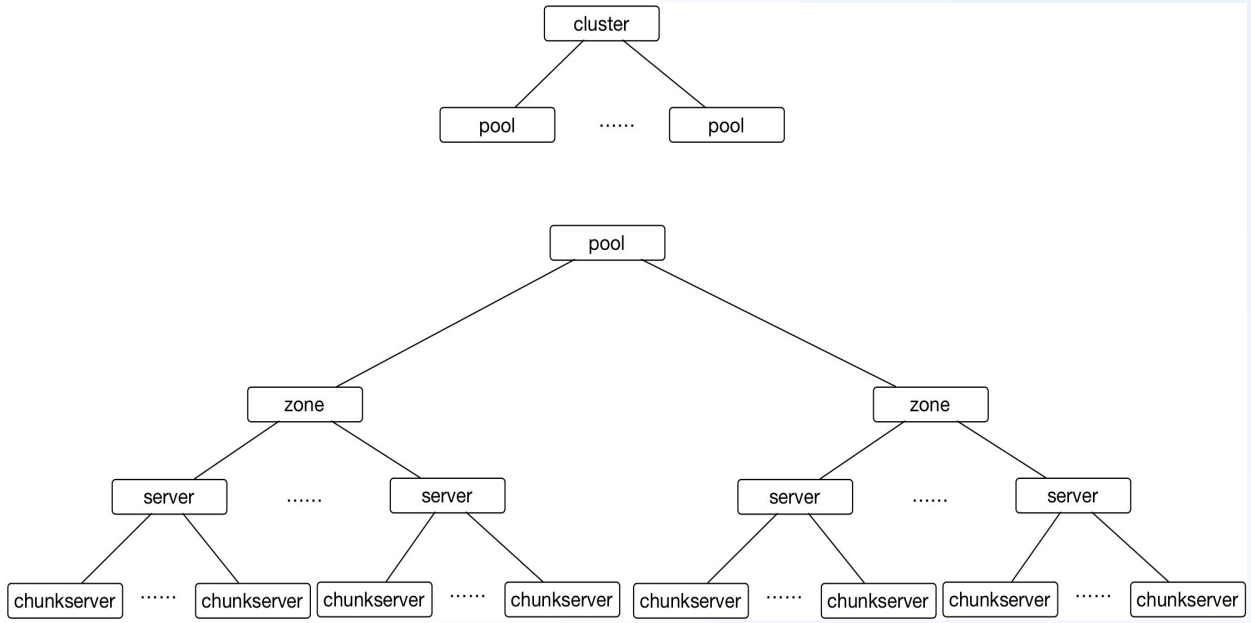
MDS是中心节点，负责元数据管理、集群状态收集与调度。MDS包含以下几个部分：

- Topology: 管理集群的 topo 元数据信息。
- Nameserver: 管理文件的元数据信息。
- Copyset: 副本放置策略。
- Heartbeat: 心跳模块。跟chunkserver进行交互，收集chunkserver上的负载信息、copyset信息等。
- Scheduler: 调度模块。用于自动容错和负载均衡。

TOPOLOGY

topology用于管理和组织机器，利用底层机器的放置、网络的规划以面向业务提供如下功能和非功能需求。

1. **故障域的隔离**：比如副本的放置分布在不同机器，不同机架，或是不同的交换机下面。
2. **隔离和共享**：不同用户的数据可以实现固定物理资源的隔离和共享。



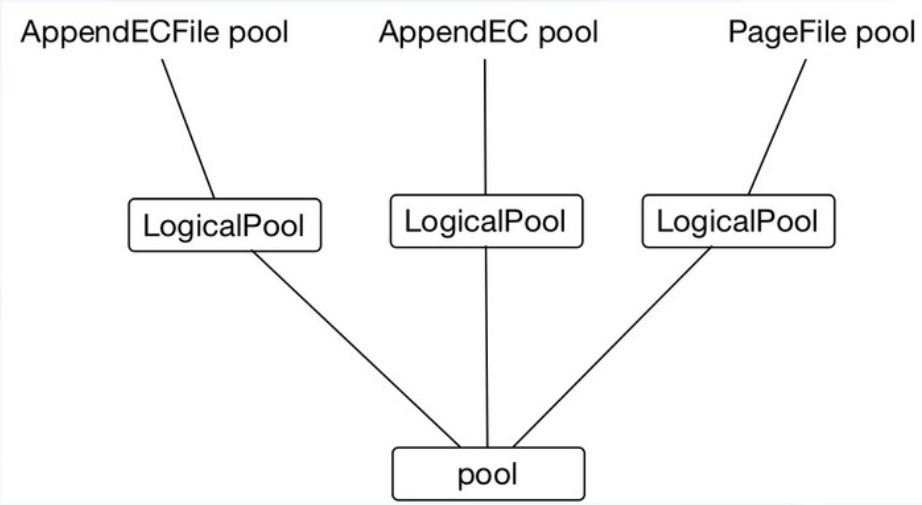
- **pool:** 用于实现对机器资源进行物理隔离，server不能跨Pool交互。运维上，建议以pool为单元进行物理资源的扩容。
- **zone:** 故障隔离的基本单元，一般来说属于不同zone的机器至少是部署在不同的机架，一个server必须归属于一个zone。
- **server:** 用于抽象描述一台物理服务器，chunkserver必须归属一个于server。
- **Chunkserver:** 用于抽象描述物理服务器上的一块物理磁盘(SSD)，chunkserver以一块磁盘作为最小的服务单元。

TOPOLOGY

curve在上物理pool之上又引入逻辑pool的概念，以实现统一存储系统的需求，即在单个存储系统中多副本PageFile支持块设备、三副本AppendFile（待开发）支持在线对象存储、AppendECFile（待开发）支持近线对象存储可以共存。

如上所示LogicalPool与pool为多对一的关系，一个物理pool可以存放各种类型的file。当然由于curve支持多个pool，可以选择一个logicalPool独享一个pool。

通过结合curve的用户系统，LogicalPool可以通过配置限定特定user使用的方式，实现多个租户数据物理隔离（待开发）。



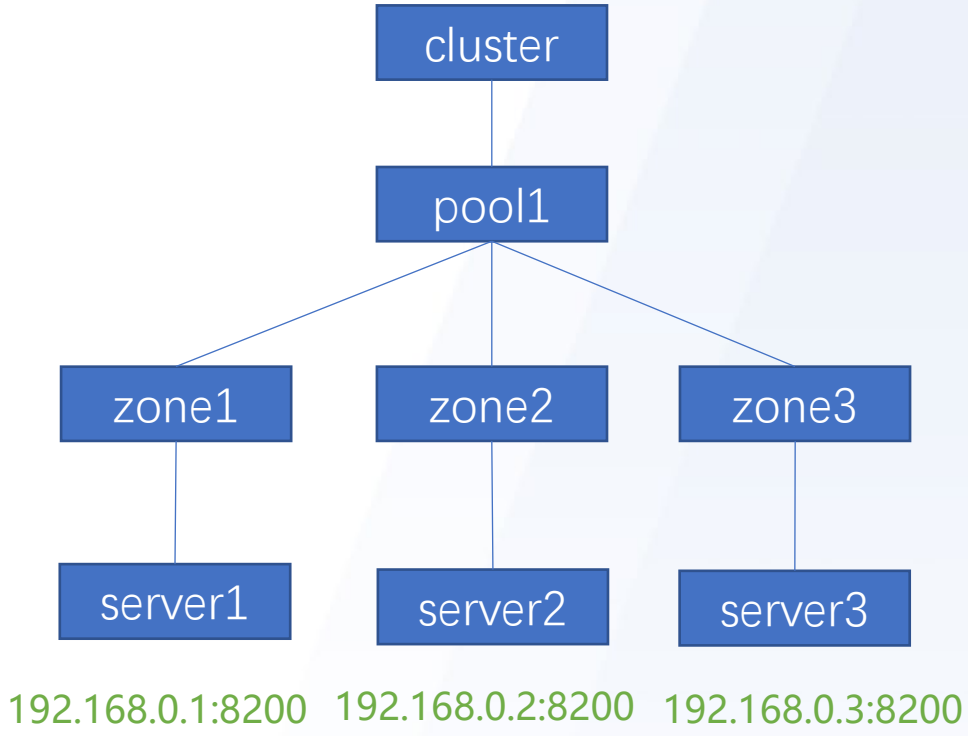
TOPOLOGY

Curve

Topology的实际例子，右侧是topo配置文件：

集群有一个物理pool，由3个zone组成，每个zone有1台server。

在物理pool上，还创建了一个逻辑pool，逻辑pool使用3个zone，采用3副本，有100个copyset。



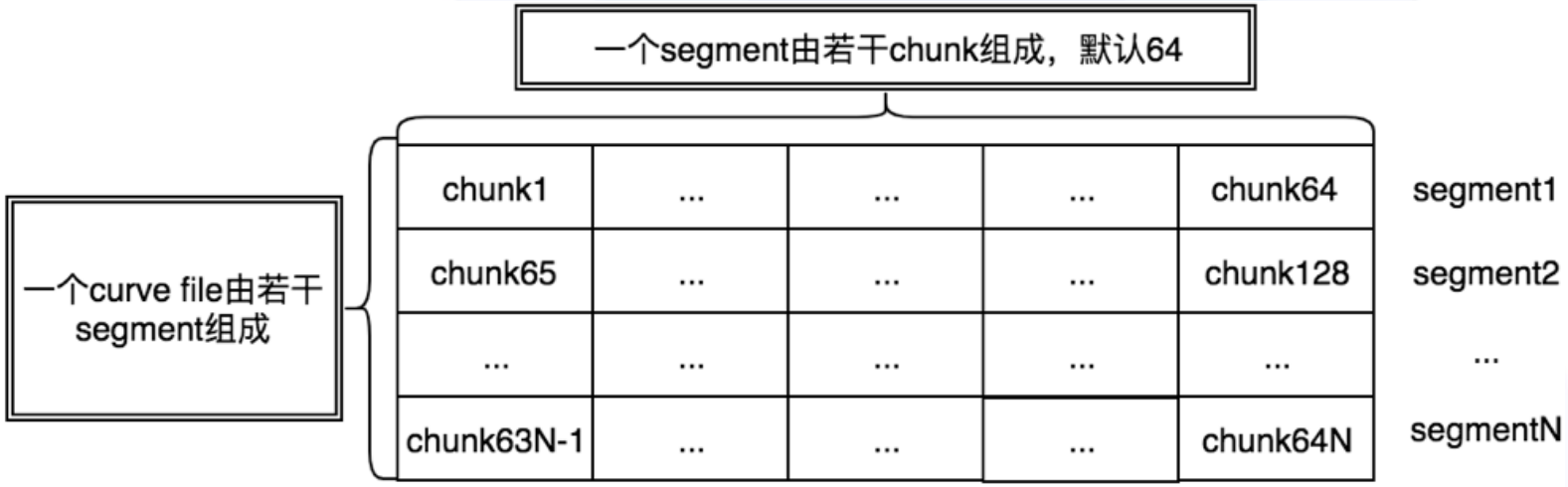

```
cluster_map:
  servers:
    - name: server1
      internalip: 192.168.0.1
      internalport: 8200
      externalip:  192.168.0.1
      externalport: 8200
      zone: zone1
      physicalpool: pool1
    - name: server2
      internalip:  192.168.0.2
      internalport: 8200
      externalip:  192.168.0.2
      externalport: 8200
      zone: zone2
      physicalpool: pool1
    - name: server3
      internalip:  192.168.0.3
      internalport: 8200
      externalip:  192.168.0.3
      externalport: 8200
      zone: zone3
      physicalpool: pool1
  logicalpools:
    - name: logicalPool1
      physicalpool: pool1
      type: 0
      replicasnum: 3
      copysetnum: 100
      zonenum: 3
      scatterwidth: 0
```

NAMESERVER

NameServer管理namespace元数据信息，包括（更具体的信息可以查看curve/proto/nameserver2.proto）：

- **FileInfo**: 文件的信息。
- **PageFileSegment**: segment是给文件分配空间的最小单位 。
- **PageFileChunkInfo**: chunk是数据分片的最小单元。

segment 和 chunk的关系如下图:



NAMESERVER

Namespace的文件的目录层次关系如右图。

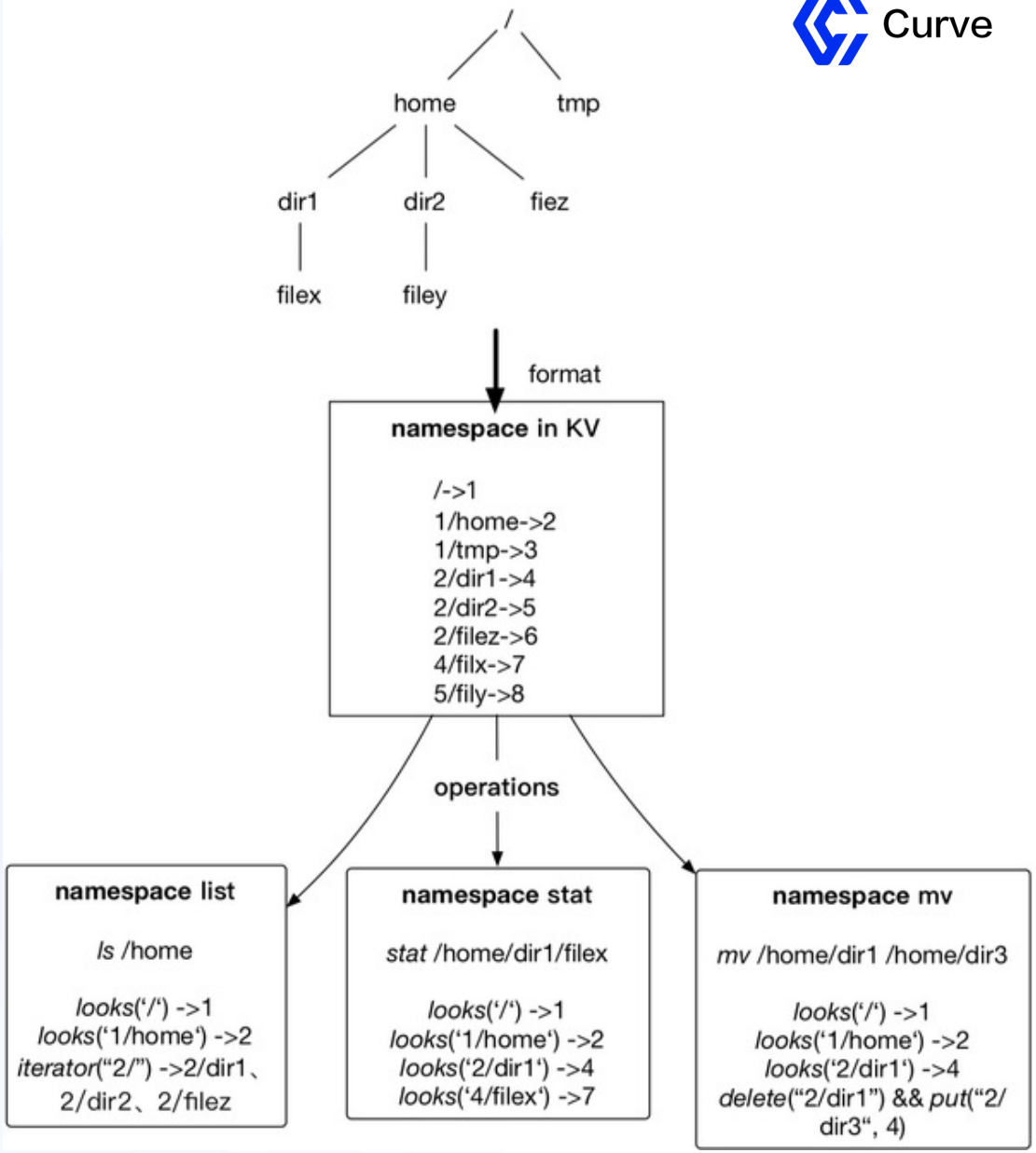
文件的元数据以KV的方式存储。

- Key：ParentID + “/”+ BaseName；
- Value：自身的文件ID。

这种方式可以很好地平衡几个需求：

- **文件列目录**：列出目录下的所有文件和目录
- **文件查找**：查找一个具体的文件
- **目录重命名**：对一个目录/文件进行重命名

当前元数据信息编码之后存储在 etcd 中。



COPYSET

Curve

Curve系统中数据分片的最小单位称之为Chunk。在大规模的存储容量下，会产生大量的Chunk，如此众多的Chunk，会对元数据的存储、管理产生一定压力。因此引入CopySet的概念，CopySet类似于ceph的pg。CopySet可以理解为一组复制组，这组复制组的成员关系完全一样。CopySet的概念在文献「Copysets: Reducing the Frequency of Data Loss in Cloud Storage」提出。

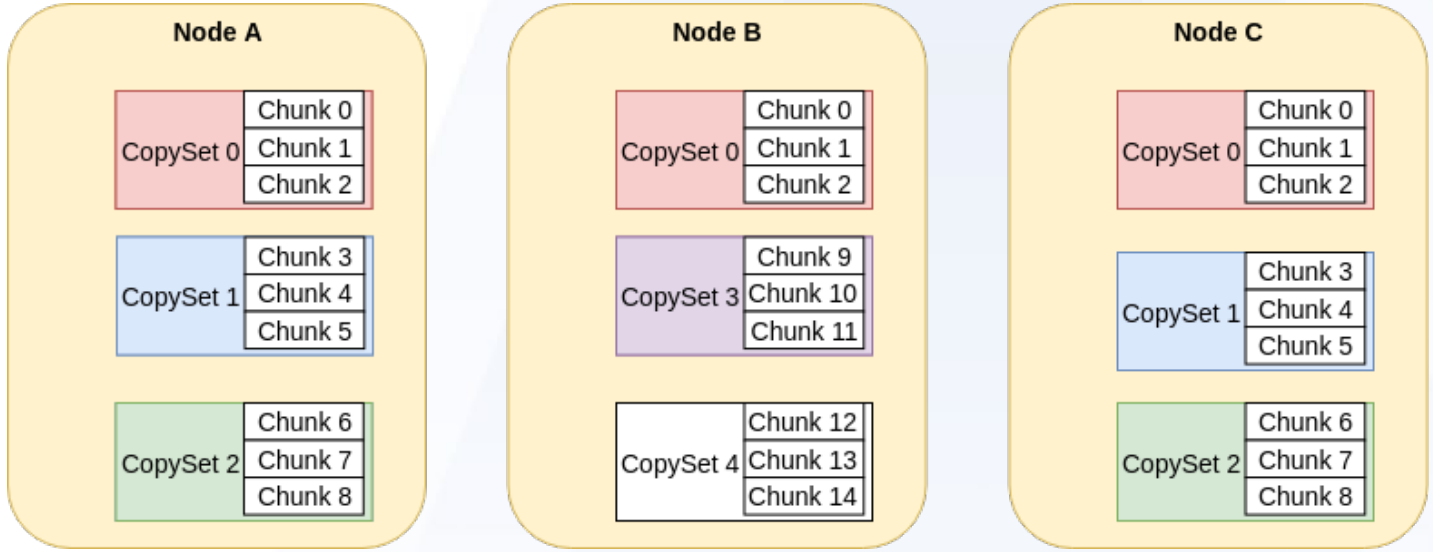
在 Curve 系统引入 CopySet 有几个目的：

1. **减少元数据量**：如果为每个Chunk去保存复制组成员关系，需要至少 ChunkID+3×NodeID=20 个byte，而如果在Chunk到复制组之间引入一个CopySet，每个Chunk可以用ChunkID+CopySetID=12个byte。
2. **减少复制组数量**：如果一个数据节点存在 256K个复制组，复制组的内存资源占用将会非常恐怖；复制组之间的通信将会非常复杂，例如复制组内Primary给Secondary定期发送心跳进行探活，在256K个复制组的情况下，心跳的流量将会非常大；而引入CopySet的概念之后，可以以CopySet的粒度进行探活、配置变更，降低开销。
3. **提高数据可靠性**：在数据复制组过度打散的情况下，在发生多个节点同时故障的情况下，数据的可靠性会受到影响。引入CopySet，可提高分布式存储系统中的数据持久性，降低数据丢失的概率。

COPYSET

ChunkServer，Copyset和Chunk三者之间的关系如下图：

Mds在分配空间时，轮流在不同的copyset中分配，每次从copyset中分配1个chunk，这个chunk用copysetId:chunkId来唯一标识。



COPYSET

Curve

Copyset的生成策略：Source code : curve/src/mds/copyset/

```
bool GenCopyset(const ClusterInfo& cluster, int numCopysets, std::vector<Copyset>* out);
```

例如要在(zone1 zone2 zone3 zone4)中创建8个copyset:

round1:

zone1	zone2	zone3	zone4
cs1	cs4	cs7	cs10
cs2	cs5	cs8	cs11
cs3	cs6	cs9	cs12

copyset-1: (cs1, cs4, cs7)
copyset-2: (cs10, cs2, cs5)
copyset-3: (cs8, cs11, cs3)
copyset-4: (cs6, cs9, cs12)

round2:

zone1	zone2	zone3	zone4
cs2	cs6	cs7	cs11
cs3	cs4	cs9	cs10
cs1	cs5	cs8	cs12

copyset-5: (cs2, cs6, cs7)
copyset-6: (cs11, cs3, cs4)
copyset-7: (cs9, cs10, cs1)
copyset-8: (cs5, cs8, cs12)

result:
cs1 [copyset-1, copyset-7]
cs2 [copyset-2, copyset-5]
cs3 [copyset-3, copyset-6]
cs4 [copyset-1, copyset-6]
cs5 [copyset-2, copyset-8]
cs6 [copyset-4, copyset-5]
cs7 [copyset-1, copyset-5]
cs8 [copyset-3, copyset-8]
cs9 [copyset-4, copyset-7]
cs10[copyset-2, copyset-7]
cs11[copyset-3, copyset-6]
cs12[copyset-4, copyset-8]

COPYSET

1. 用户发起请求(fd, offset, length)；
2. Client 向 mds 查询请求的元数据，

   并缓存到本地，请求转换为对 chunk 的请求

(CopysetId ,chunkId, offset in chunk, length in chunk);

3. Client 向 chunkserver 查询 chunk 所在的

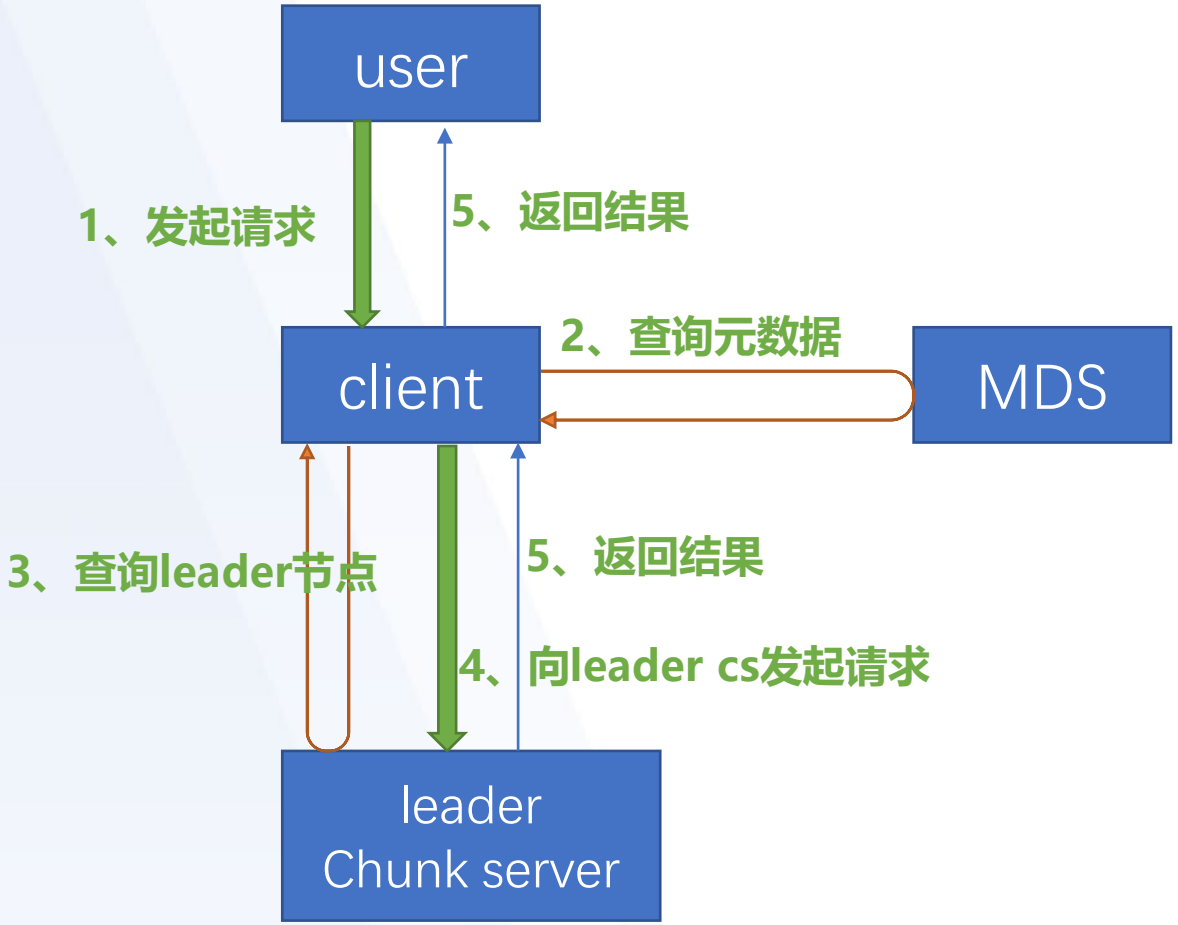
   copyset的leader Chunkserver节点；

4. Client 向 leader 发送读写请求client (IP, port,

   CopysetId, chunkId, offset in chunk, length in chunk),

   Chunkserver 完成后通知;

5. Client通知用户请求完成。



HEARTBEAT

心跳用于中心节点和数据节点的数据交互，详细功能如下：

- 通过chunkserver的定期心跳，检测chunkserver的在线状态（online, unstable, offline）
- 记录chunkserver定期上报的状态信息（磁盘容量，磁盘负载，copyset负载等），以提供运维工具查看上述状态信息。
- 通过上述信息的定期更新，作为schedule 模块进行均衡及配置变更的依据
- 通过chunkserver定期上报copyset的copyset的epoch， 检测chunkserver的copyset与mds差异，同步两者的copyset信息
- 支持配置变更功能，在心跳回复报文中下发mds发起的配置变更命令，并在后续心跳中获取配置变更进度。

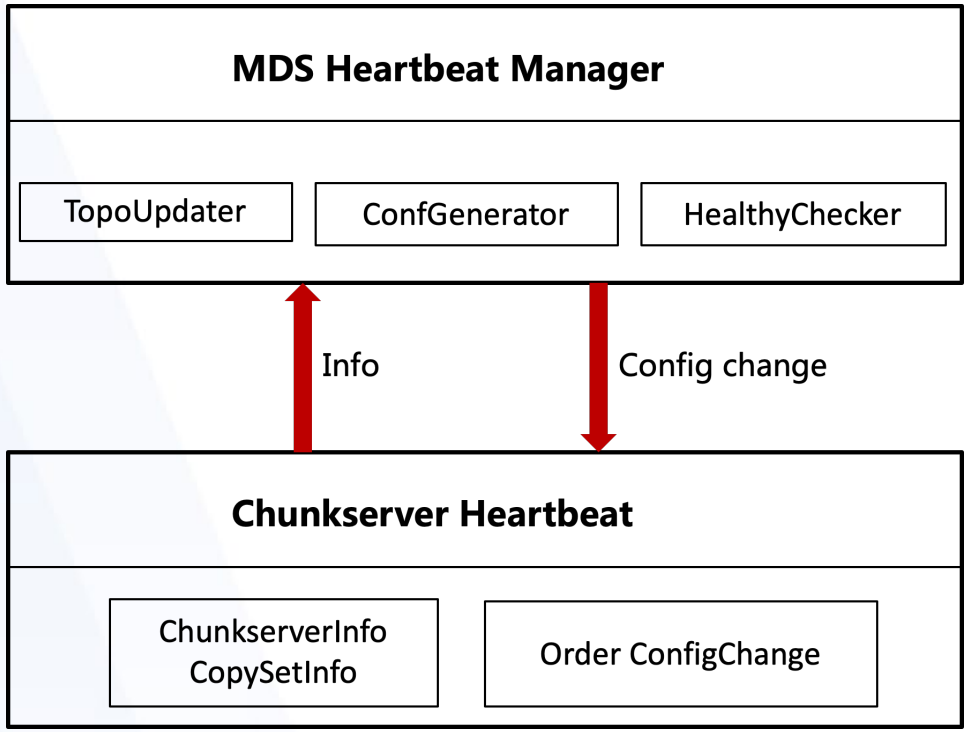
HEARTBEAT

MDS端： mds 端的心跳主要由三个部分组成：

- **TopoUpdate**r: 根据 chunkserver 上报的 copyset 信息更新拓扑中的信息。
- **ConfGenerator**: 将当前上报的 copyset 信息提交给调度模块，获取该 copyset 上可能需要执行的任务。
- **HealthyChecker**: 检查集群中的 chunkserver 在当前时间点距离上一次心跳的时间，根据这个时间差更新chunkserver状态。

Chunkserver端： chunkserver 端的心跳由两个部分组成：

- **ChunkServerInfo/CopySetInfo**：获取当前 chunkserver 上的 copyset 信息上报给 MDS。
- **Order ConfigChange**: 将 MDS 下发的任务提交给对应的 对应模块执行。

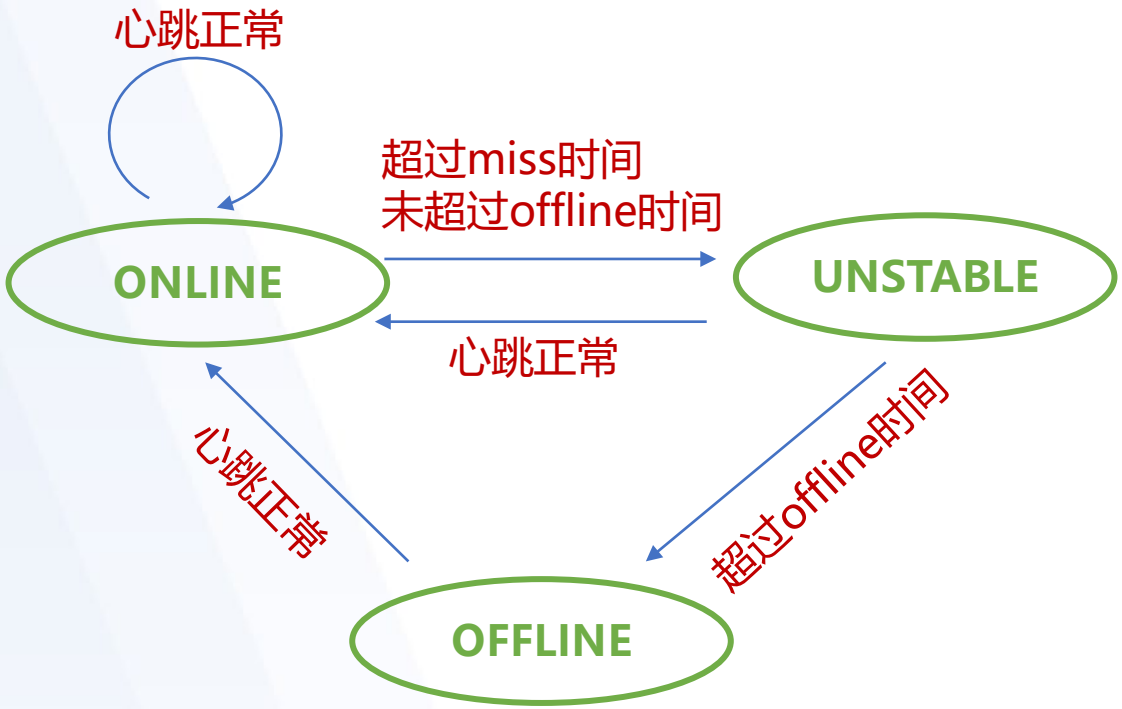


HEARTBEAT

Curve

Chunk server的状态更新：

- **Online**: chunk server在线，正常服务。
- **Unstable**: chunk server一段时间没收到心跳（默认30s），但是还没有到达offline的时间（默认30min），chunkserver状态改为unstable状态，打印一条warning日志。
- **Offline** :chunk server超过offline的时间没有收到心跳（默认30min），chunkserver状态改为offline，打印一条error日志。调度模块感知到offline状态，触发chunk server的recover修复。



SCHEDULE

Curve

Schedule（系统调度）是为了实现系统的自动容错和负载均衡，这两个功能是分布式存储系统的核心问题，也是 curve 是否能上生产环境的决定因素之一。

- **自动容错**保证常见异常（如坏盘、机器宕机）导致的数据丢失不依赖人工处理，可以自动修复。
- **负载均衡**和资源均衡保证集群中的磁盘、cpu、内存等资源的利用率最大化。

SCHEDULE

Curve

Schdedule的具体实现

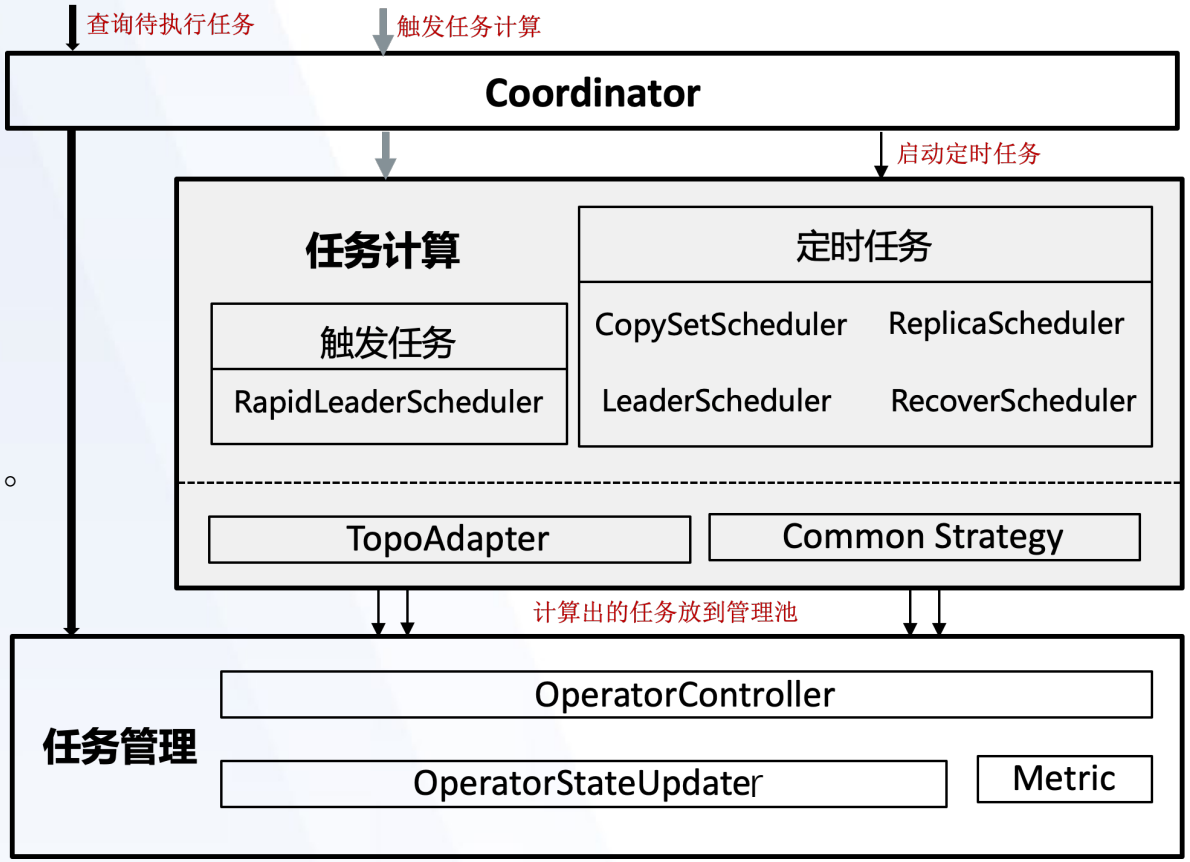
Coordinator: 调度模块的对外接口。心跳会将chunkserver上报上来的copyset信息提交给Coordinator，内部根据该信息判断当前copyset是否有配置变更任务执行，如果有任务则下发。

任务计算: 任务计算模块包含了多个定时任务 和 触发任务。

- **定时任务**由调度模块定时触发。
- **触发任务**由外部触发，管理员通过工具触发。
- **TopoAdapter** 用于获取Topology中调度需要使用的数据。
- **Common Strategy** 是通用的副本添加和移除策略。

任务管理: 任务管理模块用于管理计算模块产生的任务。

- **operatorController** 是任务集合，用于存放和获取任务;
- **operatorStateUpdater** 根据上报的copyset信息更新状态;
- **Metric**用于统计不同任务个数。



SCHEDULE

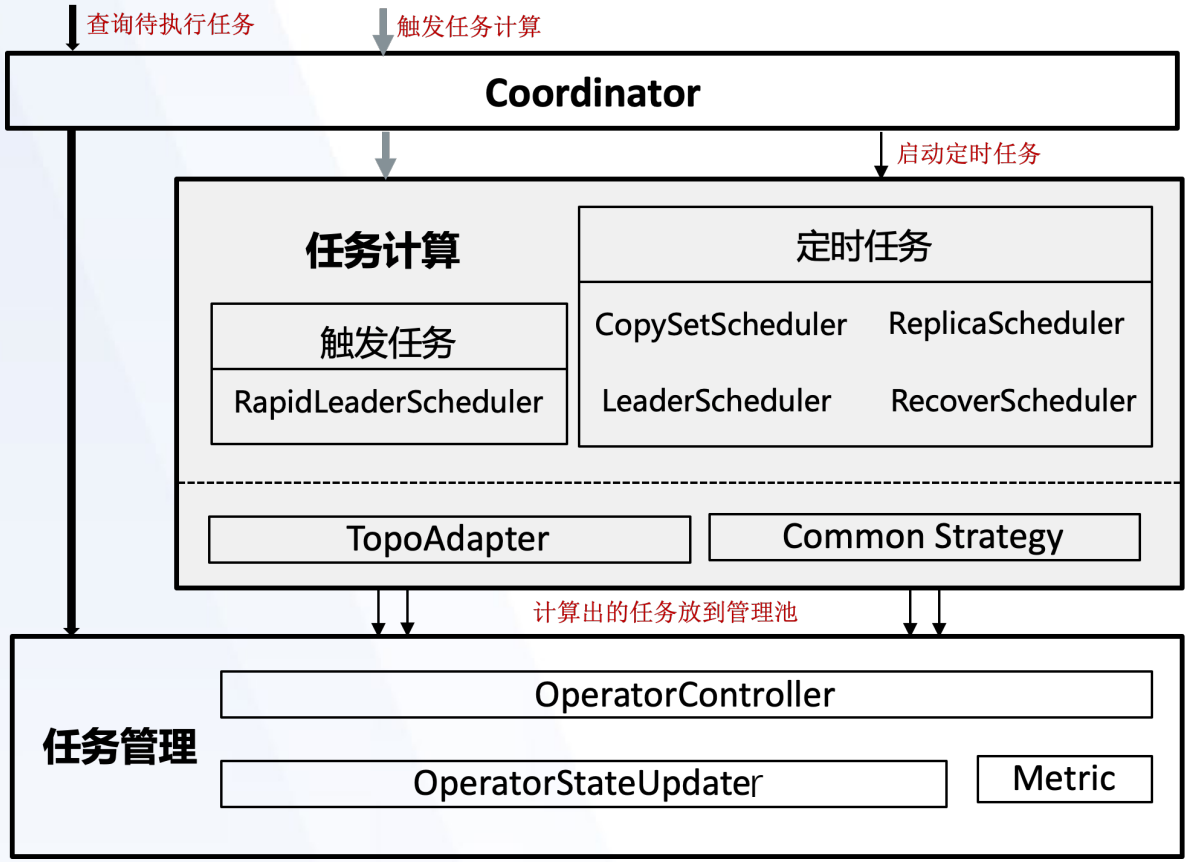
Curve

定时任务：

- **CopySetScheduler** 是copyset均衡调度器，根据集群中copyset的分布情况生成copyset迁移任务；
- **LeaderScheduler** 是leader均衡调度器，根据集群中leader的分布情况生成leader变更任务；
- **ReplicaScheduler** 是副本数量调度器，根据当前copyset的副本数生成副本增删任务；
- **RecoverScheduler** 是恢复调度器，根据当前copyset副本的存活状态生成迁移任务。

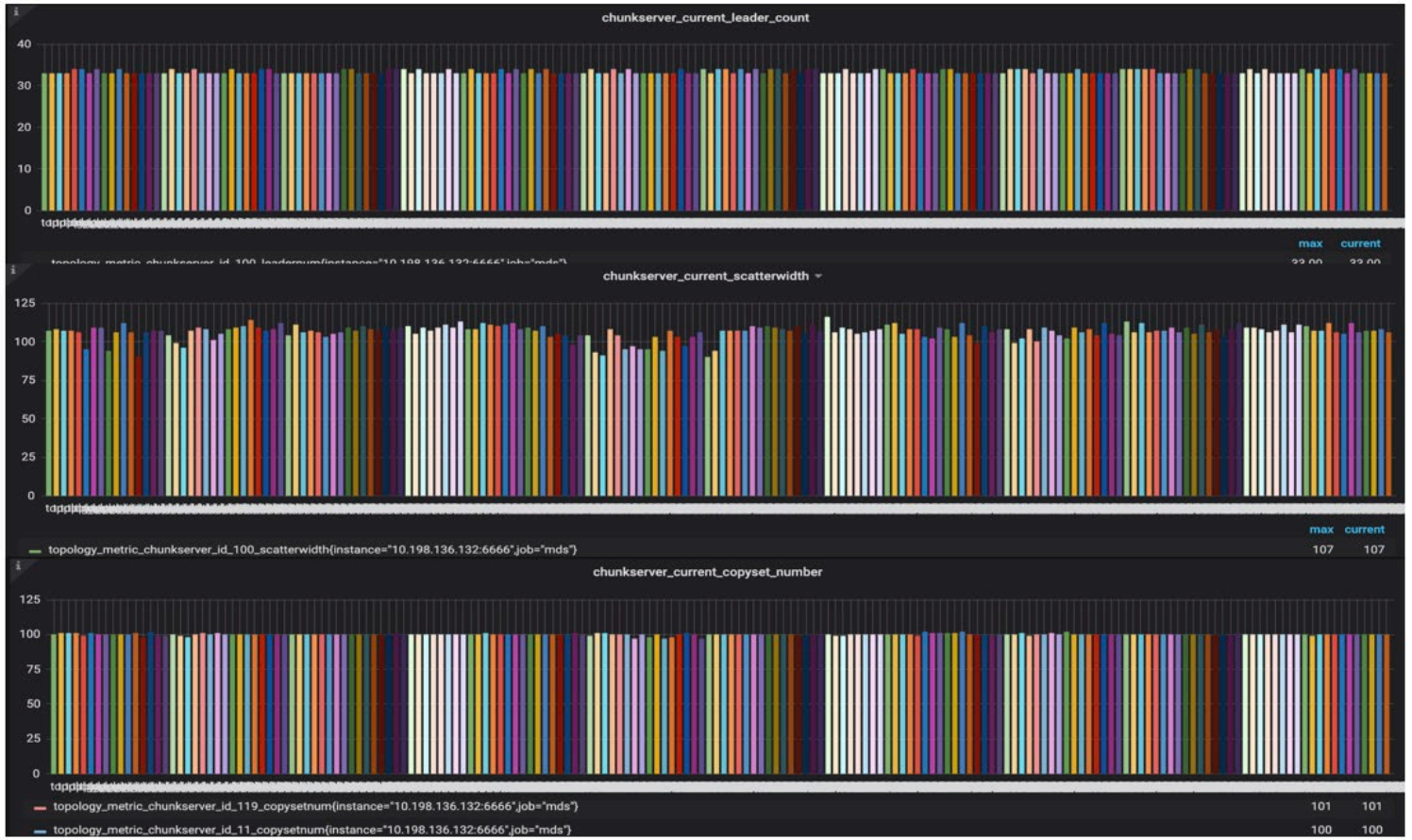
触发任务：

- **RapidLeaderScheduler** 是快速leader均衡器，由外部触发，一次生成多个leader变更任务，使得集群的leader尽快达到均衡状态。



SCHEDULE

- 集群负载和资源均衡
  - leader
    copyset
    scatter-width
  - 无需人工干预
  - 对io影响几乎无影响



欢迎大家参与CURVE项目！

- github主页： https://opencurve.github.io/
- github代码仓库： https://github.com/opencurve/curve
- 技术讲座直播：https://live.bilibili.com/22585337
- 系列讲座合集：https://space.bilibili.com/700847536/channel/detail?cid=153949

THANK YOU

D I G I T A L S A I L

扫码即可关注