**Amazon S3-Storage系统中**

**EC编码的实现介绍**

## S3-Storage简介:

Amazon S3 提供了一个简单 Web 服务接口，可用于随时在 Web 上的任何位置存储和检索任何数量的数 据。单个对象最大5GB.(更大对象的MutiUpload不在我们讨论范围)

对于小于128Kb的数据采用6副本存储,大于128Kb采用 EC编码 9-18(冗余度同2副本) 存储.对于一份数据, 以一字节为单位, 编码成 18份数据, 每份数据长度为原数据长度的 1/9,任意读取9份数据 即可获得原数据.多余的9份冗余数据, 做修复用.

这里, 我们只介绍大于128kb数据的存储. rs编码部分的数据, S3存储结构结构分为KeyMap, ACCESS, Storage 三部分.通过ACCESS验证后, 数据写到 Storage中, Storage返回由vid和bid 组成的唯一编码.将<key, volume\_id/blob\_id> 保存在 keymap系统中. keymap系统也是kv系统.

## S3-Storage 集群

亚马逊集群(region)按地域划分, 一个集群内的机器物理位置在一个城市群中, 约在200公里范围内. 一个region内部的网络延时小于2-3ms, 一个集群分为3个AZ(类似mola3中zone的概念), 一份数据的18个编码部分,分布在3个AZ内, 每个AZ包含6个.这样可保证任意一个AZ挂断,读写服务都不受影响. 一个AZ内的机器规模可达到10w台, 一个集群的数据有1 EB以上

**机器配置:**

每个存储数据的单机由96块SATA磁盘组成, 使用AMD 8核处理器, 64G内存, 系统盘不提供数据服务 ,使用500G的SATA盘.数据盘使用大容量磁盘. 使用亚马逊自己研发的文件系统, 直接控制磁盘, 内部控制磁盘上同时的读请求个数. 机器间的网络协议也是自己开发的.

**关键概念:**

**colume:** colume一开始时就被分配. 每个colume 包括18个16GB的vlet组成,18个vlet固定属于一个唯一的colume id.

**Vlet:**Vlet是数据的容器, 每一份编码的数据, 无论大小 都存在与Vlet中, Vlet固定大小16G, 在创建时就占满磁盘空间, 再在内部管理数据(blob), 18个Vlet属于一组,里面的所有blob互相对应.一个Vlet有其内部的数据结构, 包括n个128KB的page,包括Vlet index(存储bid-到pageid信息).一个page中可能保存多条record,

**Blob:**blob是前端写入的数据. 每组blob对应colume内唯一的bid, blob在每个vlet中有1/18的数据, blob在每个vlet中数据为一个shard,shard由一些record组成.

图:colume的结构(包括18个Vlet)

## 各模块介绍:

图:S3-Storage中的主要模块

C3P0: C3P0负责分配cid-bid.也就是数据的写入标示,和写入位置.cid 并非 C3PO创建, 由 skywalker创建, C3PO 定期扫描所有 colume 的剩余空间. C3PO为分布式结构. 通过一致性哈希, 每个c3po 负责一定范围内的colume id的分片. C3PO 记录为 每个cid 分配的 bid, 和 colume 的已分配空间. 每次分配同一个 colume的 新 bid 加一. 来保证每次分配的bid的不同.C3P0负责一定范围的colume, 定期扫描全部r2d2来获取当前colum容量信息.

R2D2  R2D2是数据节点,下含vlet, 结构.Vlet管理负责内容的读写, r2d2 读写出错2次, 会汇报磁盘错误给 skywalker.由skywalker发起recover等操作. 对于drop磁盘等操作, r2d2也会汇报给skywalker.R2D2上部署有DFDD服务, 每个模块可以向DFDD获取服务再集群上的分布信息.全集群(30万台)服务器, DFDD通过gossip算法的机器变化收敛时间是15s.

skywalker  最开始时,所有colume/vlet都由 skywalker根据需要创建, skywalker会定期从r2d2上获取磁盘空间剩余信息.skywalker根据集群的写入速度, 不断创建新的colume/vlet来给c3po进行分配用.

根据r2d2 发送的drop 盘信息已经DFDD上检测到的机器离线 .skywalker将得知缺失vlet的colume, skywalker遍历全部colume, 如果有colume的vlete已经不足18个 需要recover ,创建一个新的 vlet, 然后通知mcp进行 recover vlet.并将元信息持久化到 EMC集群

BA BA是rs编码端, 以1字节为单位进行编码, 对数据分成18份. 流式切分. 以 65525byte(若干byte)进行切分. 如果18份 都成功后, 会向每个 vlet发生 commit信息(待确定,推测有), 完成后 返回前段成功. 如果18个超过 6个没有写成功. 无法完成此次写入, 则回滚此次写入的数据. 向每个vlet发送 回滚命令.

skynet 缓存元信息,从EMC中获取每个colume的18个vlet所在r2d2位置, 并缓存在内存中, 供查询用

EMC 基于 keymap集群的 可靠元信息存储模块.skywalker所有对元信息所做的修改会提交到EMC, 来持久化数据.

mcp 根据 skywalker的调度来 recover vlet, 每个vlet以blob为单位进行,blob的恢复交给 Tron去做.一个集群只有少量的mcp, 来管理所有的恢复任务,恢复任务交给相应的Tron去做.

Tron 根据 mcp的调度去进行恢复Vlet. 同时也会不断扫描每个 colume 上的所有 bid, 如果一个 vlet 缺少 bid, 会发起 recover任务.Tron与R2D2混部, 一个集群有上万台的Tron服务部署.

MM 进行rebalance, 标记过期的机器, 会 rebalance, 把A-> B copy到b上, 向 BA标记 AB 不可用, 然后 cp完成后 更改skywalker的元信息.

BV 垃圾信息GC模块, 定期扫描全部 的 cid-bid 内容 , 去 skymap中查询是否存在, 如果不存在进行删除.另外, BV也提供向keymap查询一个cid-bid是否存储的反查代理. 当BA去获取一个cid-bid时,获取失败会向BV查询此数据是在系统中丢失,还是被删除.

DS 删除中心, 对于BA发送的删除命令, 定期去做删除,两阶段提交的方式 删除所有18个 part. DS也是有多台,每个删除任务将随机发送给3台DS,DS将删除任务写到本地磁盘.3个任务被3台DS分别执行(有点像cm)

keymap 持久化数据. kv数据库, 但是可以velue项建立索引.keymap是6副本的小value kv存储服务.

## 基本读写流程

图:写流程

## 1写流程:

由 webserver 接收前段请求, 将请求发给 BA, BA 首先向 c3po 索取 一组 cid-bid, BA拿到cid-bid后会向skynet查询对应的r2d2的地址. 将数据以 65532Byte进行切分, 计算出18份数据, 写到r2d2中 r2d2 中 写以 page+record 为单位进行, 每次写入计入 page中的一个 新  record. .如果写失败的r2d2数量少于12个会认为写成功.稍后做恢复.

如果写失败, ba 会gc掉所有的数据. 返回前端失败. 如果成功, BA返回 WebService一组 cid-bid, ws把其写入到 keymap集群中.

图:读流程

## 读流程:

webservice 首先向keymap 查询一个 key的 cid-bid, 然后向skynet 查询得到 cid 对应的  vlet组,(18个r2d2的位置). 根据负载 向其中10个发起读请求.选择9个响应最快的节点去查询. 将需要读的 bid(如果是 range)则将相应的range传给 r2d2, r2d2 顺序将数据返回,

数据返回以page为单位, r2d2不会对page做解析, 而是返回给BA后, BA读到page中的数据, 按照 record的格式得到所需数据.每返回一个完整的编解码数据段, BA将真实数据计算出, 返回给 WS. 直至结束

图:删除流程

## 删除流程:

ws 向keymap读到 cid-bid, 向ba发起删除命令, BA将删除的cid-bid 交给 DS(delete sentre), DS 将任务持久化到本地硬盘, 定期的进行批量删除操作. 将要对一个blob删除的时候, 先向skynet查询 对应的 18个 r2d2 位置信息, 如果不满足18个 下一轮再删除. 满足的话 通过两阶段提交, 保证18个r2d2 的删除一致性.

在vlet层面: 因为一个page中可能含有多个bid中的数据, 所以一个page中相应的record被删除后, 实际会发生一次重写, 覆盖掉删除产生的空洞. 因为是128K的范围, 所有对io影响很小.

## recover流程:

所有的 colume, vlet一开始是由skywalker创建的, 所有vlet的变化必须经过 skywalker的允许.

图 recover一个Vlet

第一种recover情形:

skywalker 定期 向r2d2 获取所有的 r2d2 的实际容量, 实际位置. r2d2 挂断与否, 或者drop盘, dropdisk等信息,如果发现相应的 colume中少了 vlet, skywalker会创建一个 空的vlet, 然后将恢复vlet的任务交给mcp, mcp 再调度 tron进行实际的恢复, tron会读尚存在的副本, 进行编解码, 然后按照blob单位, 将vlet修复满 .

第二种recover情形:

Tron 会定期扫描所有的colume, 根据skynet提供的分布, 再向r2d2,获取 每个 vlet的 blob列表(md5), 如果一致继续下一个, 不一致会详细获取缺少的blob, 然后向缺少blob的vlet进行修复, 同上.

图:balance流程

## balance流程:

一些机器过保, 或者磁盘均衡操作, 并不需要计算编解码, 有MM进行, mm选择目的r2d2 和 源r2d2, 然后告诉 BA这两个节点在做 balance, 禁止读写. 在拷完后 , 对拷完解禁目的和源端 r2d2, 然后将修改提交给skywalker.skywalker也会将修改通过EMC持久化

## 垃圾回收:

时间集群可能产生一些垃圾数据, 比如写失败没有回滚掉的数据. 垃圾回收由BV进行.bv不断的list所有colume-bid, 然后向keymap中反向索引, 如果不存在. 那么将其删掉.

## 其他

相比我们保留原始数据的好处:

在做range读的时候, 即使是任意一个range内的数据挂了一块, 也不需要降级读, 读取所耗带宽一样.