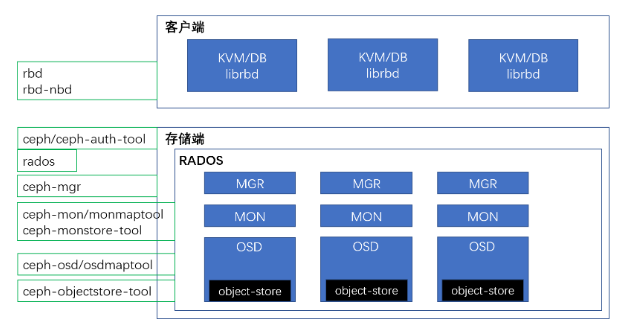
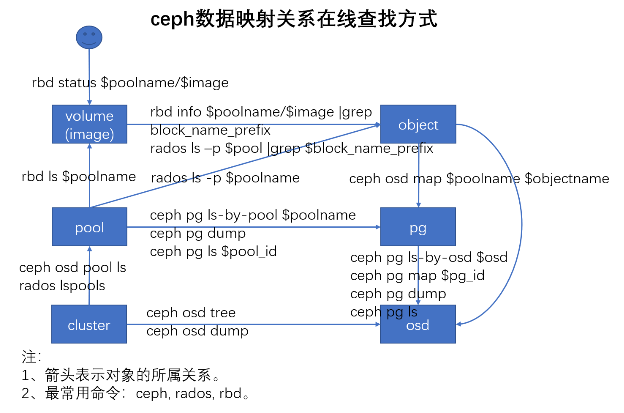
**第六章 Ceph集群维护**

**Ø Ceph常用管理命令**

ceph相关命令很多，各个层次、各个组件有不用的命令工具，以Ceph-RBD为例，总体如下图所示。



图一，Ceph-RBD各个层次常用命令工具



图二，ceph数据映射关系在线查找方式

下面是以表格形式列举了Ceph相关命令的介绍。

|  |  |  |
| --- | --- | --- |
| **分类** | **常用命令** | **介绍** |
| ceph | ceph -s, ceph status，ceph health detail | 查看集群状态 |
| ceph osd tree | 查看集群的拓扑结构 |
| ceph df, ceph osd df | 查看集群、osd的容量等信息 |
| ceph pg dump | 查看所有的PG状态等信息 |
| ceph osd pool ls [detail] | 查看集群的pool信息 |
| ceph osd create, ceph osd  rm $osdid | osd的创建和删除 |
| ceph osd crush rule ls|dump | 查看集群的crush rule信息 |
| ceph mgr module ls | 查看mgr组件的所有模块 |
| ceph auth list | 查看集群的认证信息 |
| ceph config show | 查看集群的配置，N版本开始可以通过ceph config统一进行ceph配置管理 |
| rados | rados lspools | 查看集群的pool信息 |
| rados get <obj\_name> <out\_file> | 查看单个object |
| rados put <obj\_name> <in\_file> | 写入单个object |
| rbd | rbd create $image\_name -p $pool --size $size --image-format 2[1] | 创建卷 |
| rbd ls $pool | 列举 $pool 下面的卷 |
| rbd info -p $pool $image | 查询 $pool 下面卷 $image 的详细信息 |
| rbd du $pool/$image | 查询卷的实际使用容量 |
| rbd status $pool/$iamge | 查询卷状态 |
| rbd snap list $pool/$image | 查询一个卷对应的快照 |
| rbd export $pool/$image $image\_file  rbd import $image\_file $pool/$image | 导出导入卷，可以用在卷离线迁移等场景，注意import的目的卷不能已经存在 |

表一 ceph相关命令

ceph相关命令很多，这里只是列举了部分高频命令，帮助你快速地掌握基本操作。更多请参考ceph -h、rados -h、rbd -h等帮助手册。

**Ø Ceph Dashboard**

主要进行14.2.22版本dashboard部署和功能介绍。

**dashboard部署**

1. 安装包。对于14.2.22版本，yum源可能没有ceph-mgr-dashboard安装包，需要单独下载并安装，可以使用如下ceph官方源：

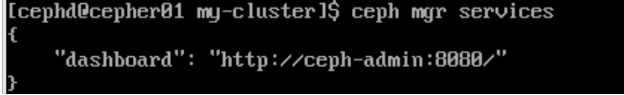
<https://download.ceph.com/rpm-nautilus/el7/noarch/>

1. 社区默认没有开启dashboard功能，需要手动开启，并且需要强制开启：

ceph mgr module enable dashboard --force

1. 过几秒钟可以查到mgr服务有了dashboard，说明开启成功

ceph mgr services



1. nautilus用新的命令创建用户：

echo “ceph” > password.txt

ceph dashboard ac-user-create admin administrator -i password.txt

1. 注意确保mgr主节点上的firewalld服务被关掉，或者设置打开端口

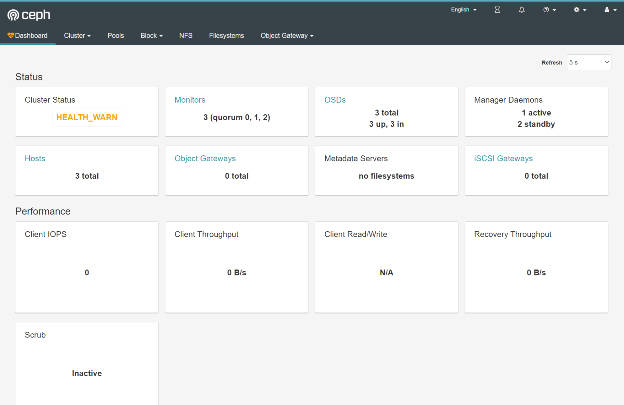
sudo systemctl stop firewalld && sudo systemctl disable firewalld

或者

firewall-cmd --add-port=8443/tcp --permanent

firewall-cmd --reload

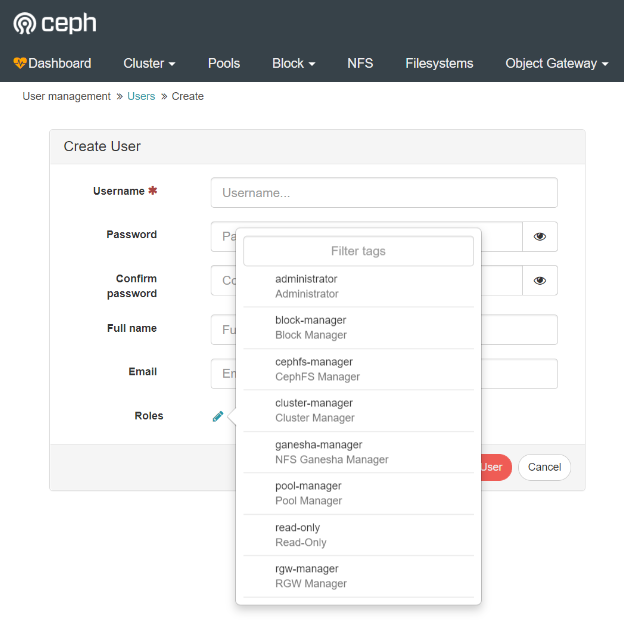
1. 通过上面ceph mgr services输出的url，或者ip:port访问ceph dashboard页面

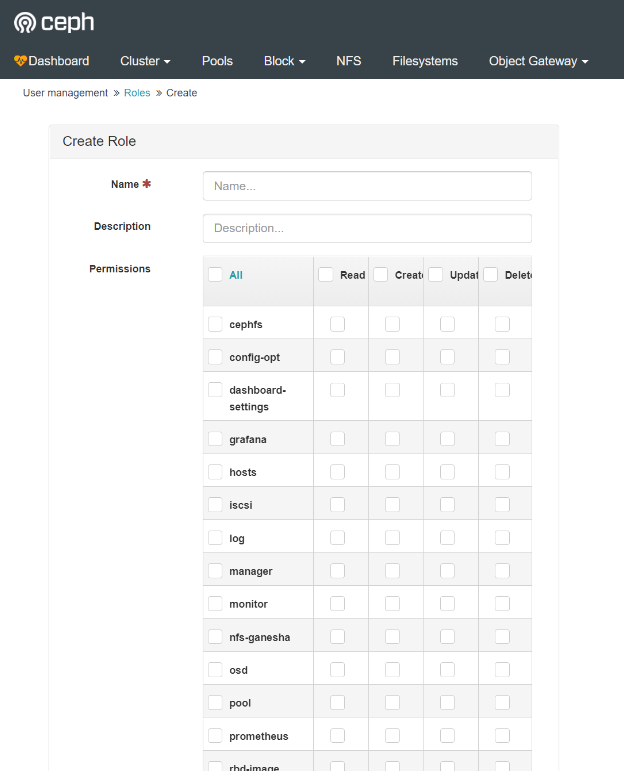


**dashboard介绍**

nautilus版本的dashboard支持如下功能：

1. Dashboard健康大盘，可以方便地一览集群情况，以卡片的形式展示了集群的状态、性能和容量信息
2. 组件状态和配置信息，包括Hosts、Monitor、OSD、Configuration、CRUSH map、manager等，并且支持集群flag配置修改，组件配置修改等
3. 存储池pool信息增删改查
4. 块存储信息，包括卷、快照、iSCSI
5. 文件存储信息，包括NFS和Filesystems
6. 对象存储信息，包括用户和桶
7. 支持丰富的用户和角色管理，基于RBAC：
   1. 可以给创建的用户配置不同角色，内置的角色包括：块管理员、文件管理员、对象管理员、集群管理员、超级管理员等等。
   2. 进一步可以进行自定义角色，通过矩阵的形式可以给所有组件配置CURD分离的权限，拥有很好的灵活性。





1. 支持dashboard的swagger api，可以很方便的管理接口，地址为：http://${ip}:${port}/docs
2. 监控数据查看，可以通过启用prometheus和grafana来查看，需要做一些配置、安装工作，这里不详细介绍
3. 其他功能在右上角的顶部菜单栏可以看到，包括：后台任务管理、通知等

总体来说，ceph dashboard的功能性和易用性得到了很大的提升，除了丰富的信息查询，还支持常用的运维操作。在此基础上对接了第三方的prometheus和grafana，可以完成监控、报警、监控图表展示等功能。

**Ø 扩容及缩容**

**扩容**

**准备工作**

服务器、网络等

1.0 名词解释

|  |  |  |
| --- | --- | --- |
| 中文全称 | 英文简称 | 解释及作用 |
| 管理ip | Admin ip | 用于主机登录管理,访问yum源/dns/ntp/监控等公共服务。 |
| 存储ip | Public ip | 客户端主机通过存储ip来访问存储集群。 |
| 集群ip | Cluster ip | Osd节点之间同步数据，mon节点可以不配置该ip. |
| Mon节点 | Monitor node | Mon节点一般为单数，3个或者5个，看具体集群规模，规模小的集群可以和osd节点共用。 |
| Osd节点 | Object storage device | OSD所在的主机 |

1.1 整理当前ceph存储集群主机信息列表,示例:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sn | hostname | adminIP | publicIP | clusterIP | type |
| xxxxx1 | host1 | 192.168.1.1 | 192.168.122.1 | 192.168.62.1 | mon+osd |
| xxxxx2 | host2 | 192.168.1.2 | 192.168.122.2 | 192.168.62.2 | mon+osd |
| xxxxx3 | host3 | 192.168.1.3 | 192.168.122.3 | 192.168.62.3 | mon+osd |
| xxxxx4 | host4 | 192.168.1.4 | 192.168.122.4 | 192.168.62.4 | osd |

......

规划整理扩容节点信息,示例(当前假设扩容2个节点):

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| sn | hostname | adminIP | publicIP | clusterIP | type |
| xxxxxn | hostn | 192.168.1.61 | 192.168.122.61 | 192.168.62.61 | osd |
| xxxxxm | hostm | 192.168.1.62 | 192.168.122.62 | 192.168.62.62 | osd |

子网掩码/网关及vlanID,示例:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | netmask:255.255.255.0  gw:192.168.1.254  vlanID:2 | netmask:255.255.255.0  gw:192.168.122.254  vlanID:4 | netmask:255.255.255.0  gw:192.168.62.254  vlanID:6 |  |

**扩容前检查**

检查checklist以及推荐配置

因涉及到多台主机进行操作,建议使用批量操作工具,如ansible/saltstack/puppet等。

1. 给扩容节点配置好各网段ip及根据环境配置路由并验证各扩容主机与原集群主机间网络联通是否正常(通常,集群节点需要关闭防火墙)并确保避免IP冲突

sudo systemctl disable firewalld

1. 配置扩容节点yum源/ntp/dns/磁盘raid级别/raid缓存策略以及安装相关工具包(如Megacli或storcli以及一些日常运维工具或脚本)等,并确保扩容节点能正常访问公共服务(如yum源/ntp/dns)
2. 通常扩容节点的相关公共配置(如yum源/dns/ntp/系统版本/内核版本/磁盘是否做raid/raid级别/raid缓存策略等)需要和原集群主机配置保持一致
3. 关闭主机默认开启的文件扫描建立索引定时任务,原因是osd节点上有较多磁盘且磁盘容量较大,存储的文件可以能多,导致每次系统扫描文件建立索引都会消耗大量系统资源.

mv /etc/cron.daily/mlocate /home/

注意: 1.根据具体环境配置路由,路由配置不正常导致的后果是严重的,如扩容完成后集群内部数据无法向扩容节点进行数据迁移无法完成balance, 或是客户端无法访问扩容节点数据导致客户端io请求阻塞等.

2.配置ntp后要即时进行时间同步并更新同步到系统硬件时钟(hwclock -w),集群主机间时间不同步会导致集群节点间的认证失败.导致数据集群内部数据通信失败

**扩容变更**

扩容变更实施步骤

|  |  |
| --- | --- |
| 1.对各扩容节点安装ceph相关rpm包 | yum install ceph ceph-debuginfo -y |
| 2.从原集群主机上获取ceph集群配置及keyring文件到扩容节点 | scp /etc/ceph/{ceph.confceph.client.admin.keyring} root@${扩容节点}:/etc/ceph/ |
| 3.在依次在各扩容节点创建osd实例 | ceph osd crush add-bucket `hostname -s` host |
| num\_id\_osd=`ceph osd create` |
| mkdir -p /var/lib/ceph/osd/ceph-${num\_id\_osd} |
| mkfs.xfs /dev/${osd-dev} -f |
| mount /dev/${osd-dev} /var/lib/ceph/osd/ceph-${num\_id\_osd} |
| ceph-osd -i $num\_id\_osd --mkfs --mkkey |
| ceph auth add osd.${num\_id\_osd} osd &apos;allow \*&apos; mon &apos;allow rwx&apos; -i /var/lib/ceph/osd/ceph-${num\_id\_osd}/keyring |
| ceph osd crush add osd.${num\_id\_osd}  ${reweight}   host=`hostname -s` |
| chown ceph:ceph /var/lib/ceph/osd/ceph-${num\_id\_osd} -R |
| systemctl start ceph-osd@${num\_id\_osd} |
| 4.若crush有人为调整为多层级,则根据具体环境调整扩容节点在crush中的位置开始数据迁移完成数据再平衡 |  |

**缩容**

缩容通常值缩容集群容量,主要针对提供osd服务节点

缩容变更实施步骤

1.停止缩容节点上ceph相关服务进程,

systemctl stop ceph.target

2.依次将缩容节点上的所有osd踢出集群:

ceph osd out osd.$id

3.删除已踢出集群osd:

ceph osd rm osd.$id; ceph auth del osd.$id

4.将已踢出集群的osd移除crush:

ceph osd crush remove osd.$id

5.当缩容节点上osd均已从crush中移除后,将缩容节点从crush中移除:

ceph osd crush remove <缩容节点>

6.清除/etc/ceph/ceph.conf 配置文件中与缩容节点相关信息(如 osd相关信息),并将更新后的/etc/ceph/ceph.conf 配置文件同步更新到集群存量主机

7.其他清理操作,如卸载ceph相关软件/监控信息更新/缩容osd磁盘数据清除/ip回收/集群信息更新等

**Ø 数据均衡**

**为什么要做数据均衡**

ceph的CRUSH算法中的bucket选择算法在保障了ceph灵活满足各种功能需求的同时，一定程度上达到了较好的pg均衡效果。但是在实际部署、运维过程中，又会有各种情况会导致较大程度的容量和负载不均。其中包括如下场景：

1. 部署初始配置规划不合理，主要是PG数、pool数等
2. 集群节点数少
3. 运维过程操作多，扩容、缩容、增加pool、节点/磁盘维护中的退出和加入等操作
4. 后期硬件迭代导致的异构，一个集群存在异构的不同容量的磁盘

而因为数据不均，一方面，使用率最高的osd会成为系统的瓶颈，导致集群整体容量使用率不高，另一方面，数据不均也会导致负载不均，从而容易引发集群不稳定。数据均衡直接影响集群稳定和成本，是ceph管理员需要掌握的基本操作。

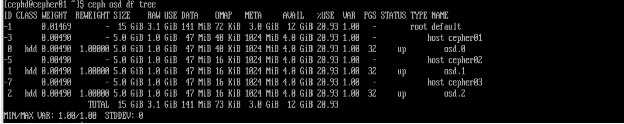
**Ceph数据均衡的几种模式**

出现了数据不均衡，我们要如何处理。这里有3种方式，reweight、upmap和balancer。

需要注意的是，Ceph的数据映射都是以PG为单位的，所以所有涉及数据迁移的地方，都是以PG为对象进行的。

**reweight方式**

通过Ceph osd df tree我们可以看到Ceph有一个weight值和一个reweight值。



图三，ceph osd df tree输出结果

其中reweight值表示迁移权重，可以用来调整以实现数据均衡，这个取值是0~1，越大表示数据量越多，初始部署的时候就是1，所以就不能继续调大。weight值表示初始的容量权重，通常和osd对应的磁盘容量对应，如4T磁盘，weight值就是4。

调整单个osd的权重进行均衡的操作如下：

ceph osd reweight $osd\_num $reweight

对于需要调整的osd列表，通常需要很多次这样的调整，为了减小工作量，Ceph也提供了批量调整的方式。批量调整有两种模式，一种是按照osd空间利用率（reweight-by-utilization）,另外一种是按照PG在OSD上的分布（reweight-by-pg）。此外，为了变更前期了解数据迁移量，评估影响，还可以通过加“test-”的方式查看本次要迁移的信息。

以reweight-by-utilization为例，操作如下：

ceph osd test-reweight-by-utilization [$overload] [$max\_change] [$max\_osds] [—no-increasing]

ceph osd reweight-by-utilization [$overload] [$max\_change] [$max\_osds] [—no-increasing]

其中后面4个是**可选**参数，各个参数的含义如下：

|  |  |
| --- | --- |
| 参数 | 含义 |
| overload | 整型，取值范围：>=100,默认值是120。  表示对超出集群平均使用率一定程度的osd进行调整，默认的120表示对超出20%的进行调整 |
| max\_change | 浮点类型，趋势范围：[0,1]，默认值是0.5。  表示每个osd的reweight值调整的最大幅度的限制 |
| max\_osds | 整型，默认值是4，表示每次至多调整的osd个数 |
| —no-increasing | 如果带上，表示不允许reweight上调，如果不带，表示允许reweight上调，但是不超过1 |

**upmap方式**

upmap，是一种对pg进行重映射的方式，从社区的luminous v12.2.x版本开始支持。这种方式可以对个别pg或者整个集群部分pg进行很精确的重映射，从而可以实现很好的均衡效果。但需要注意的是，这种方式要求客户端必须能够理解upmap结构（osdmap中的），所以内核态或者用户态的客户端都需要是同样新的版本，大于luminous v12.2.x，否则会出问题。

操作方式如下：

1. 导出当前pool的osdmap

ceph osd getmap -o curr\_osdmap

1. 查看当前pool的pg分布情况

osdmaptool --test-map-pgs --pool $pool\_id ./curr\_osdmap

1. 使用导出的osdmap计算pg的移动

osdmaptool curr\_osdmap --upmap afterupmap --upmap-pool $pool\_name --upmap-max 100 --upmap-deviation 0.01

这里的upmap-max是最大计算次数，upmap-deviation是调整后的最大误差目标，最后结果写afterupmap文件

1. 查看准备进行的重映射情况

vim afterupmap

1. 执行重映射

source afterupmap

执行完成，相关pg进入backfill状态，带remapped完成，即可验证pg分布情况

1. 导出并查看新的pg分布情况

ceph osd getmap -o new\_osdmap

osdmaptool --test-map-pgs --pool $pool\_id ./new\_osdmap

**balancer**

balancer是终极利器，从Luminous版本引入，实现了自动化的数据均衡。

概念解析

|  |  |
| --- | --- |
| 概念 | 解释 |
| optimize | 一次优化方案生产操作，产生一个优化计划 |
| plan | 一个优化计划，根据当前集群情况的某一次优化方案 |
| eval | 评估一个优化计划 |
| mode | 具体优化的模式，包括none/crush-compat/upmap，N版本默认为空，需要设置 |
| execute | 一个优化计划的一次执行，进行实际的数据迁移 |

可以针对整个集群进行调整、单个pool进行调整或者某一些pool进行调整，下面以整个集群的调整方式为例。

1. 查看balancer模块的状态

ceph balancer status

1. 设定数据平衡的模式

ceph balancer mode none/crush-compat/upmap

1. 设置一个任务计划

ceph balancer optimize plan1

1. 评估一个任务计划

ceph balancer eval plan1

1. 执行一个计划

ceph balancer execute plan1

1. 删除所有计划

ceph balancer reset

1. 启用/停用balancer的自动优化功能

ceph balancer on

ceph balancer off

1. 启用自动优化功能之后，默认情况下，balancer以60s为周期、以整个集群为粒度自动化执行数据平衡。这种默认配置可能会对延迟敏感的业务造成影响，所以可以采用调整执行的时间段、每次的调整步长等方式来降低影响。例如

ceph config set mgr.${mgrName} mgr/balancer/begin\_time 0000

ceph config set mgr.${mgrName} mgr/balancer/end\_time 0600

ceph config set mgr.${mgrName} mgr/balancer/crush\_compat\_step 0.4

1. 通过如下命令可以查到所有的balancer模块相关的配置key

ceph config ls |grep balancer

1. 需要注意的是，L版本用的ceph config-key命令从M版本开始即废弃了，统一使用ceph config命令来实现参数的修改。

**几种数据均衡方式对比**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 方式 | 版本要求 | 均衡效果 | 影响大小 | 数据迁移量 | 自动化程度 |
| reweight | 无 | 较差 | 批量调整影响较大 | 批量调整数据迁移量大 | 手动 |
| upmap | 客户端：L及以上  服务端：L及以上 | 很好 | 根据映射改变程度 | 根据映射改变程度 | 手动 |
| balancer | crush-compat模式对版本无要求，upmap模式同样对版本有如上要求 | 很好 | 灵活调整 | 灵活调整 | 全自动 |

**Ø 常见故障及处理方法**

**背景知识**

**硬盘**

这里介绍下硬盘类型和每种硬盘的特点，监控和运维，其他。

1. 硬盘类型和特点

硬盘按照接口协议类型和介质类型可以分为很多种。如下：

|  |  |  |  |
| --- | --- | --- | --- |
| 接口协议  介质 | SATA | SAS | NVME |
| HDD | SATA HDD | SAS HDD | - |
| SSD | SATA SSD | SAS SSD | NVME SSD |

1. 对于硬盘的接口协议，NVMe好于SAS，SAS好于SATA。简单地说，从SATA到SAS再到NVMe是命令队列深度和命令队列数量的很大提升，从而带来IO性能的大幅提升。
2. 对于HDD和SSD的区别，HDD是传统的机型硬盘，机械臂的寻道需要花费数毫秒的时间，所以数据读写延迟较高。而SSD是通过芯片来制造的，读写闪存颗粒，读写延迟很低。
3. 对于SSD的存储介质又分为多种，包括SLC、MLC、TLC、QLC，每个闪存颗粒分别可以存1bit、2bit、3bit、4bit。所以对于这4种闪存颗粒，容量递增，成本递减，但是由于每个颗粒的可擦写次数变少了，导致4种闪存颗粒寿命递减，使用场景后者也更加偏向于读。

2. 监控和运维

1. 通过安装通用的smartmontools安装包，我们可以通过smartctl工具进行硬盘信息的查看。
2. 对于配置了RAID的硬盘，我们可以使用通用的storcli/megacli工具很方便地查看adapter card、vd、pd健康状态信息，以及阵列卡的日志等。
3. 对于NVMe盘，通过安装nvme-cli安装包，我们可以使用nvme命令来进行NVMe盘的管理。
4. 对于Linux系统工具，我们常用的包括：df、du、lsblk、iostat、iotop、lsof等。

**阵列卡**

存储节点根据选型可以配置阵列卡配置RAID，这可以利用阵列卡的缓存和条带化达到加速效果。对于存储节点OSD磁盘，通常配置单盘RAID0。当然相比没有阵列卡，我们引入了阵列卡可能出现的问题，比如初期的选型需要考虑阵列卡缓存和带宽，避免成为主机的瓶颈，建设阶段需要设置合理的RAID配置（读写缓存等），运维阶段需要处理可能的阵列卡本身、阵列卡缓存、阵列卡电池或电容故障。

**硬盘故障处理**

一. osd数据盘故障处理:通常osd数据盘故障会导致osd进程写osd磁盘失败并导致osd进程中止,处理办法通常根据osd故障盘更换时间远近分不同处理办法:

1.若能尽快安排故障盘更换,则故障盘更换前不需要把故障osd给out出集群,避免了数据向其他正常osd迁移,更换osd故障盘后只对故障盘osd进行数据回填

2.若不能尽快安排故障盘更换,则故障盘更换前需要故障osd给out出集群,使故障osd数据迁移到其他正常osd磁盘,恢复集群多副本高可用的健康状态,避免因长时间未对故障osd磁盘维护的过程中出现其他故障,增加集群不可用风险

待办：换盘步骤

二. osd节点系统盘故障,正常情况下系统盘通常都会由两块磁盘做raid1, 当其中一块磁盘出现故障时,系统虽然还能运行,但偶尔也会因为其中一块系统盘故障导致整台主机负载飙高并影响到该主机上所有osd进程运行(通常表现为该主机上osd进程打印心跳超时日志等),导致集群出现block,因此需要将该主机上的osd给stop并尽快安排维护在线维护系统故障盘.若因各种情况实在不能尽快维护的,也需要把该主机上的所有osd给stop并out出集群.

**服务器故障处理**

a. RAID/HBA卡

1.RAID/HBA故障导致其上磁盘无法读写的导致主机宕机的,尽快安排停机维护,若不能尽快安排维护的需把该主机上osd给out出集群

2.RAID/HBA故障出现reset情况会导致其上osd写磁盘异常而中止会重启,通常阵列卡发生reset频率不高,但最好还是尽快安排停机维护

b. CPU、内存：

1.CPU、内存故障通常都会打印相关日志,在登录主机带外远程管理卡界面也会显式出相关硬件异常,

c. 其他硬件(主板/电源/风扇等)故障

这些硬件故障基本都需要停机维护,原则基本和上述硬件故障维护差不多,需要注意的是主板停机更换会伴随着主机硬件时钟回归到原始状态(和当前实际时间相差甚远),维护完成并开机启动后系统时钟同步完成之均用的是硬件时钟,会导致该主机上的osd进程因和集群时间偏离过大,出现该主机上osd相关认证失败,导致集群osd无法完成数据同步,集群状态异常,因此开机后需要在系统时间同步到ntp服务器时钟后再拉起osd.

**网络故障处理**

a. 端口故障：bond、交换机端口

b. 时延类：闪断、丢包、错包、拥塞等，不同网络平面

c. 防火墙：部分端口不通、单通

d. 网络分区

**其他故障处理**

**故障一:一个monitor down问题处理**

**现象**

当一个monitor down之后，可能出现*ceph –s* 在数秒内hang住的情况，然后输出集群状态信息中表明一个monitor down

**解决过程**

1. 尝试拉起mon进程

*service ceph start mon.a 或者 systemctl start ceph-mon@a* (对于REHL7)

1. 若启动失败，检查mon map是否完整。

*ll /var/lib/ceph/mon/ceph-a/*

*从unable to read magic*等日志可以看出mon map有损坏。

若mon map损坏，则需要重新生成mon map，再启动mon进程。

1. 在有mon quorum的情况下，从quorum生成monmap：

*ceph mon getmap -o /tmp/monmap*

1. 生成mon keyring

方法一：正常mon上复制 /var/lib/ceph/mon/ceph-b/keyring 到故障mon节点上/tmp/ceph.mon.keyring

 方法二：ceph auth get mon. -o /tmp/ceph.mon.keyring

1. 添加client.admin到ceph.mon.keyring：

*ceph-authtool  /tmp/ceph.mon.keyring  –import-keyring  /etc/ceph/ceph.admin.keyring*

最终内容如下：

*cat /tmp/ceph.mon.keyring*

[mon.]

        key = xxxxYHNYgUWTCxAApBP5QgdpC5YiiiIRaRxxxx==

        caps mon = "allow \*"

[client.admin]

        key = yyyyYHNY4Az2GhAAEXcC7WAdTM1YGORrzKyyyy==

        auid = 0

        caps mds = "allow"

        caps mon = "allow \*"

        caps osd = "allow \*"

1. 删除已有的故障monitor：

*ceph mon remove mon.a*

备份原有的mon数据到/tmp目录下：

*cp –r /var/lib/ceph/mon/ceph-a /tmp*

删除原有目录下的mon数据

*cd /var/lib/ceph/mon/ceph-a*

*rm -rf ./\**

1. 根据monitor map和keyring重建monitor：

*sudo -u ceph ceph-mon [--cluster {cluster-name}] --mkfs -i a --monmap /tmp/monmap --keyring /tmp/ceph.mon.keyring*

*例如：*

*ceph-mon --mkfs -i a --monmap /tmp/monmap --keyring /tmp/ceph.mon.keyring*

*chown ceph:ceph -R  /var/lib/ceph/mon/*

1. 启动mon daemon

REHL6：*service ceph start mon.a*

REHL7：*systemctl start ceph-mon@a*

1. 查看集群状态，mon恢复正常。

*ceph -s*

**故障二:两个monitor同时down问题处理**

**现象**

查看ceph集群状态，已经不能得到正常的输出

**解决过程**

**场景一：**

对down掉的mon进程进行拉起，若拉起成功，则mon服务可用性恢复，后续分析mon down的原因。

执行以下命令进行拉起：

*systemctl start ceph-mon@mon1*

检查mon进程是否正常启动：

*ps –ef |grep ceph-mon*

*ceph –s*

**场景二：**

2个mon进程同时down，重启无效，monmap损坏。

解决方法：

将现存的mon上的monmap提取出来，注入到monmap损坏的mon上重启。

需要先stop活着的mon进程，进行extract monmap，然后将monmap拷贝到其他两个节点。再在两个节点上分别进行inject monmap并拉起mon进程，最终3个mon都正常可用。

具体操作如下：

首先，从monmap最新(leader)的节点上提取monmap的前提是先stop这个mon，否则因为有锁而会出错。如下：

*ceph-mon -i b --extract-monmap /tmp/monmap*

IO error: lock /var/lib/ceph/mon/ceph-b/store.db/LOCK: Resource temporarily unavailable

error opening mon data directory at &apos;/var/lib/ceph/mon/ceph-b&apos;: (22) Invalid argument

先停止mon进程

*systemctl stop ceph-mon@b*

然后进行monmap的提取，并启动mon.b

*ceph-mon -i b --extract-monmap /tmp/monmap*

*systemctl start ceph-mon@b*

*ps -ef |grep ceph-mon*

ceph 6372 1 0 17:28 ? 00:00:00 /usr/bin/ceph-mon -f --cluster ceph --id b --setuser ceph --setgroup ceph

root      6504  2235  0 17:28 pts/0    00:00:00 grep --color=auto ceph-mon

将提取的monmap拷贝到mon-a和mon-c主机

*scp /tmp/monmap mon-a:/tmp/monmap*

*scp /tmp/monmap mon-c:/tmp/monmap*

分别在mon-a和mon-c上注入拷贝过来的monmap

*ceph-mon -i a --inject-monmap /tmp/monmap*

*systemctl start ceph-mon@a*

*ceph-mon -i c --inject-monmap /tmp/monmap*

*systemctl start ceph-mon@c*

**场景三：**

在恶劣情况下，mon进程down，重启无效，monmap损坏，并且mon所在主机已经不能正常工作，或者不能启动，并且短时间主机无法修复。

解决方法：

需要将集群的mon总数降为1，在将mon降为单点的情况下，保证可用性，服务正常。后续再主机可用或者有新机器可用的情况下在添加mon。

假设仅存活mon.b。

停止mon.b进程，然后进行monmap的提取

*systemctl stop ceph-mon@b*

*ceph-mon -i b --extract-monmap /tmp/monmap*

从monmap中删除mon.a mon.c，减少mon总数

*monmaptool /tmp/monmap --rm a*

monmaptool: monmap file /tmp/monmap

monmaptool: removing a

monmaptool: writing epoch 3 to /tmp/monmap (2 monitors)

*monmaptool /tmp/monmap --rm c*

monmaptool: monmap file /tmp/monmap

monmaptool: removing c

monmaptool: writing epoch 3 to /tmp/monmap (1 monitors)

将删除过mon的monmap注入本地，然后启动mon.b

*ceph-mon -i b --inject-monmap /tmp/monmap*

*systemctl start ceph-mon@b*

*ps -ef |grep ceph-mon*

ceph 6372 1 0 17:28 ? 00:00:00 /usr/bin/ceph-mon -f --cluster ceph --id b --setuser ceph --setgroup ceph

root      6504  2235  0 17:28 pts/0    00:00:00 grep --color=auto ceph-mon

**故障三:不一致的PG**

如果集群状态显示pg inconsistent，表明三副本数据中存在不一致

1.查找不一致的PG

执行 :ceph health detail

HEALTH\_ERR 2 pgs inconsistent; 2 scrub errors; noout,nodeep-scrub flag(s) set

pg 1.88e is active+clean+inconsistent, acting [35,40,122]

pg 2.502 is active+clean+inconsistent, acting [65,35,113]

2 scrub errors

详细不一致信息可以通过查看monitor中的/var/log/ceph/ceph.log日志查看不一致的日志，比如：

xxxx-01-09 15:11:28.720762 osd.35 3.1.23.54:6822/76521 3838 : cluster [ERR] 1.88e shard 35: soid 87b1f88e/rbd\_data.144ee524639756.00000000000003e4/head//1 missing attr \_, missing attr snapset

xxxx-01-09 15:11:31.873166 osd.35 3.1.23.54:6822/76521 3839 : cluster [ERR] 1.88e scrub 0 missing, 1 inconsistent objects

其中1.88e 代表的PGID ， shard 35 代表的是OSD.35,   87b1f88e 代表不一致的对象的hash值， rbd\_data.144ee524639756.00000000000003e4/head//1 代表不一致的对象；

2.确定不一致对象的位置

根据这些信息，我们就能查找到不一致的对象存储的位置，对象存储在OSD.35中，/var/lib/ceph/osd/ceph-35/current ，所属PG为1.88e, /var/lib/ceph/osd/ceph-35/current /1.88e\_head，对象的hash值为87b1f88e，reverse hash值为e88f1b78，不一致的对象可能存在的位置有多种可能，/var/lib/ceph/osd/ceph-35/current /1.88e\_head/DIR\_E/ DIR\_8/ DIR\_8/  [ DIR\_F/]  |   [ DIR\_F/DIR\_1/]  |  [ DIR\_F/DIR\_1/ DIR\_B/]，最终我们是在/var/lib/ceph/osd/ceph-35/current /1.88e\_head/DIR\_E/ DIR\_8/ DIR\_8/DIR\_F/ 这个路径下找到了该对象，

rbd\\udata.144ee524639756.00000000000003e4\_\_head\_87B1F88E\_\_1

查看该对象的扩展属性，确实是空的

attr -l rbd\\udata.144ee524639756.00000000000003e4\_\_head\_87B1F88E\_\_1

3.将不一致对象拷贝到tmp目录下，(-a 参数保留文件的扩展属性信息)

cp –a rbd\\udata.144ee524639756.00000000000003e4\_\_head\_87B1F88E\_\_1 /tmp

4.修复不一致的对象

ceph pg repair 1.88e

5.确认已修复

ceph –s确保pg inconsistent状态消失

**故障四:osd所在磁盘文件系统错误**

文件系统报错xfs\_log\_forec:error -5 returned

1. 如果OSD没有停止，停止该OSD服务

systemctl stop ceph-osd@${id}

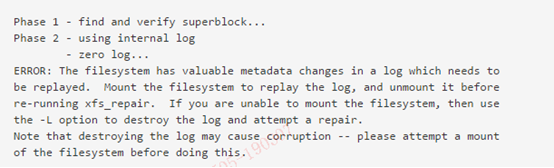
2. 卸载osd对应的目录：

umount /var/lib/ceph/osd/ceph-${id}

3. 使用xfs\_repair尝试修复OSD对应的设备：

xfs\_repair /dev/xxx

如果提示如下信息，继续下面的操作，否则跳到下个步骤4



！！！**注意：**不要使用xfs\_repair –L去修复，-L修复会丢失数据。

出现这种情况，重新mount 该osd，

mount –o rw,noatime,attr2,inode64,noquota /dev/xxx  /var/lib/ceph/osd/ceph-${id}

触发xfs replay journal

再次卸载osd对应的目录：

umount /var/lib/ceph/osd/ceph-${id}

再次执行xfs\_repair，如果执行仍然报错，重建该OSD；

4.重启该OSD服务

systemctl start ceph-osd@${id}

**Ø ceph多地容灾介绍**

xx

**Ø ceph集群性能分析**

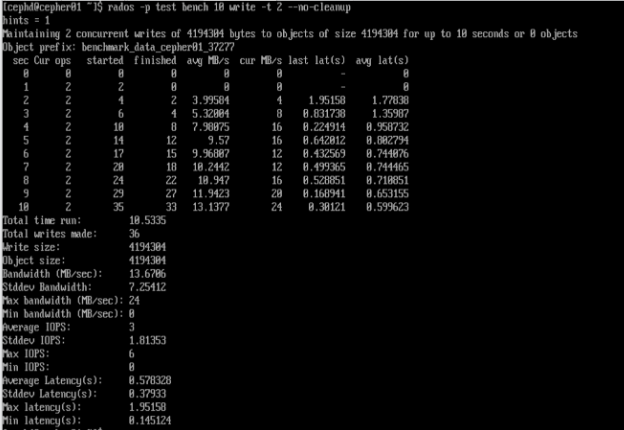
**性能预估**

**基准性能测试**

通过rados bench可以测试rados集群的性能，例如：

进行集群顺序写测试，对象为test pool，并发度2，不清理数据（有的时候出于接下来的测试需要可以不清理测试数据）

rados -p test bench 10 write -t 2 --no-cleanup



rados bench结果图

使用rbd bench简单测试单个卷的性能。

rbd bench test/testimg —io-type write



rbd bench 结果图

**第十章 Crush原理及设计**

**Ø Crush算法介绍**

crush是Ceph核心设计之一，在Sage最初的博士论文里面就出现了。

对于分布式存储系统，首先要解决集群部署之初时**静态**的数据路由、数据容量和负载的均衡问题。随后在运维阶段，无法避免的需要对磁盘、节点进行剔除、加入、缩容或者扩容，这里就需要支持**动态**的数据路由、数据容量和负载的均衡，这也是更加重要的需求。毕竟磁盘、节点的故障不可避免，业务的需求不断变化，与此同时还要保障生产的稳定可用、成本可控。

传统的存储系统设计，直接用静态映射，每次数据IO都要查询一次映射数据metadata，这种设计容易使metadata server成为单点或者性能瓶颈。而ceph的设计目标是支持超大规模、无单点。CRUSH算法，每次数据IO只要在客户端“算一算”就可以直接和对应的节点进行数据IO。

针对指定的PG，有个对应的pg\_id，crush算法的以pg\_id、cluster map和placement rule作为输入。cluster map通常不变，在故障时才会更新，placement rule基本不会改变（特殊需求需要变更除外），所以对于指定PG的每次crush计算，获得的结果总是一致的。另外，因为哈希函数是伪随机的，crush选择每个目标存储对象概率相对独立，从而可以保证数据在整个集群中均匀分布。

**Ø Crush数据结构和算法原理**

**概念介绍**

|  |  |
| --- | --- |
| 概念 | 介绍 |
| cluster map | 集群的元数据信息，通常是monitor维护，包括：monitor map、osd map、pg map、crush map等 |
| crush map | 集群的树状结构信息 |
| bucket | 集群树状结构的中间节点，如rack、host等等 |
| root | 集群树状结构的根节点，一般名为default |

**Crush map**

Ceph在设计之初就考虑了大规模的支持和故障容忍，所以就有多层级化的集群拓扑结构。包括：root -> region -> datacenter -> room -> pod -> pdu -> row -> rack -> chassis -> host -> osd 这11层，其中root是根节点，root、host、osd一般是必备的，用户可以根据实际需要进行配置。通常在生产环境中不会配置所有层级，但是推荐至少配置rack或者host级别的故障域，以避免机柜或者主机级别故障造成的业务中断。

**Crush副本选择过程**

在crush map的基础上，ceph通过设置好的crush rule（placement rule）来进行副本选择和数据分布。如下图所示，主要分为：tack，select，emit 3个过程，下面详细说明：

1. tack：选定根节点。一般就是选择root，集群默认的名字是root层级名字为default。
2. select：选择bucket。即按照集群的树状层次结构依次选择各层级的bucket，这个过程是最重要的过程，过程决定了最终数据的均衡情况和集群故障域。对于整个路径的选择算法有2种，分别为firstn和indep。这2个算法都是实现的深度优先遍历算法，区别在选不出指定数量的异常情况下给出的结果不同，firstn的结果会去除为空的节点，而indep会按顺序给出带空值的结果。例如一次选择过程得出的结果如下：

* firstn: [2, 5, 8]
* indep: [2, 5, CRUSH\_ITEM\_NONE, 8]

type字段决定了故障域配置，默认是host，即以host作为故障域，集群运行过程中，单个节点的故障不会导致集群不可用（除去无冗余的情况）。对于重要生产环境的配置，我们可以调整为以rack或者其他更高层级为故障域，以实现更高数据可用性和持久性。

对于每个bucket也有自己的选择算法，这个选择算法可以很大程度影响均衡度和运维过程中的数据平衡效率。后面再作单独介绍

select过程可能会出现冲突、过载等异常情况，稍后单独说明。

1. emit：输出结果。



副本策略默认crush rule

**选择过程的异常处理**

1. 冲突

因为选择的过程有随机因子，所以可能会出现一个osd被选中2次的情况，这种情况就是冲突。

1. OSD过载（失效）

即选出的结果出现较大程度不均衡的情况。

1. 处理方式

Crush会自动处理过程中出现的这些异常，主要通过重试来解决。crush算法的输入包括待选择对象的特征值和随机因子，对象特征值是不变的，通过调整随机因子可以实现异常情况下的重试。为了避免死循环或者执行效率低，有choose\_local\_retries和choose\_total\_retries这2个参数控制局部重试次数和全局重试次数。

**Crush的bucket选择算法**

最初Sage Weil的设计有4种选择算法，unique、list、tree和straw，8年后又加了一个straw2算法。

CRUSH的几种bucket选择算法对比：

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 算法  对比项 | unique | list | tree | straw | straw2 |
| 时间复杂度 | *O(1)* | *O(N)* | *O(log(N))* | *O(N)* | *O(N)* |
| 数据迁移量 | 多 | 多 | 较多 | 少 | 最少 |

从表中可以看出，综合来看，straw2算法更有优势。Straw2算法是在straw算法基础上的改进版，为了解决straw算法在实际生产中仍然有较多的不相关的osd之间数据迁移问题。并且从L版本开始，straw2是默认的bucket选择算法。

**Ø Crush实践以及生产环境案例分析**

**梳理不同案例的crush配置**

通常没有什么特别需求时可不用进行特别的crush配置,当有特殊需求时可人为手动配置crush.特殊需求包括对crush进行不同逻辑域划分:

需求一:osd对应的磁盘若对应不同磁盘类型如HDD/SSD等,当需要不同性能级别存储需求时,可将使用SSD磁盘的osd划分到一个逻辑域,对应使用HDD磁盘的osd划分到另一个逻辑域.具体需求可以是:1.使用SSD磁盘的osd所在逻辑域提供高性能型存储,HDD磁盘的osd所在逻辑域提供容量型存储;2.使用SSD磁盘的osd所在逻辑域用于存储元数据,使用HDD磁盘存储实际数据

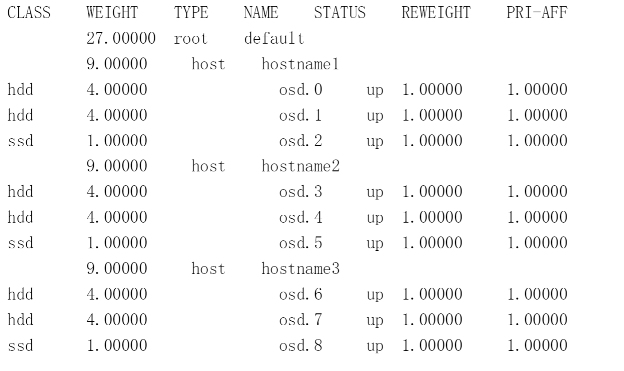
需求二:可能与磁盘介质类型无关,但需要将集群存储的数据进行物理隔离,比如用于生产的数据存储在集群的一部分osd节点上,用于测试的数据存储在集群的另外一个部分osd节点上.两个隔离域互不影响.

操作:

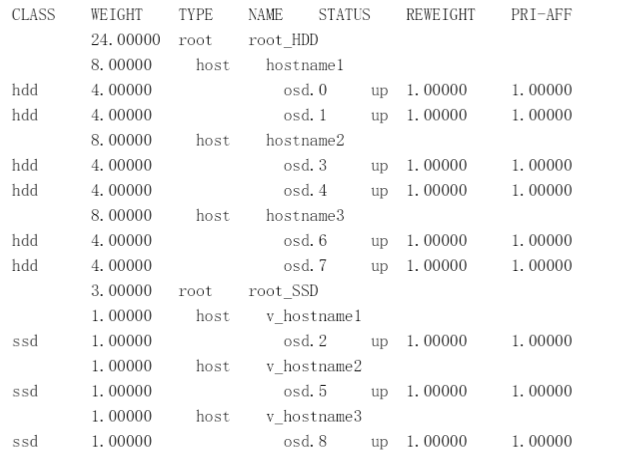
划分多个逻辑域通常可以在crush中创建对应个数的顶级bucket,并将不同逻辑域的osd节点移到对应的bucket下,并配置对应的crush rule以指导数据存储到不同顶级bucket下的osd磁盘,实现数据存储按逻辑域隔离:

例如:

通常未经人为调整部署后集群的ceph osd tree 结果如下:



当需要按osd磁盘类型创建不同隔离域时,需要人为手动调整crush结构,示例如下:



上述crush相关调整命令如下:

完成crush结构调整后还需要配合 crush rule才能指导pool数据存储到对应隔离域,分别创建各隔离域对应的crush rule,如HDD\_ruleset和SSD\_ruleset,命令如下:

ceph osd crush rule create-simple HDD\_ruleset root\_HDD host

ceph osd crush rule create-simple SSD\_ruleset root\_SSD host

此后创建存储池时指定对应crush rule 则应用不同crush rule的存储池则存储到不同的隔离域,存储池创建示例:

ceph osd pool create SSD\_pool1 <pg\_num> <pgp\_num> SSD\_ruleset

ceph osd pool create HDD\_pool1 <pg\_num> <pgp\_num> HDD\_ruleset

上述为原理演示,只做简单crush 结构调整和简单crush rule 配置,实际环境集群可能不止三个节点,crush 结构更庞大**层级**也可能更多,crush rule 花样也更多,请根据具体环境与需求具体配置

增加rack隔离说明: