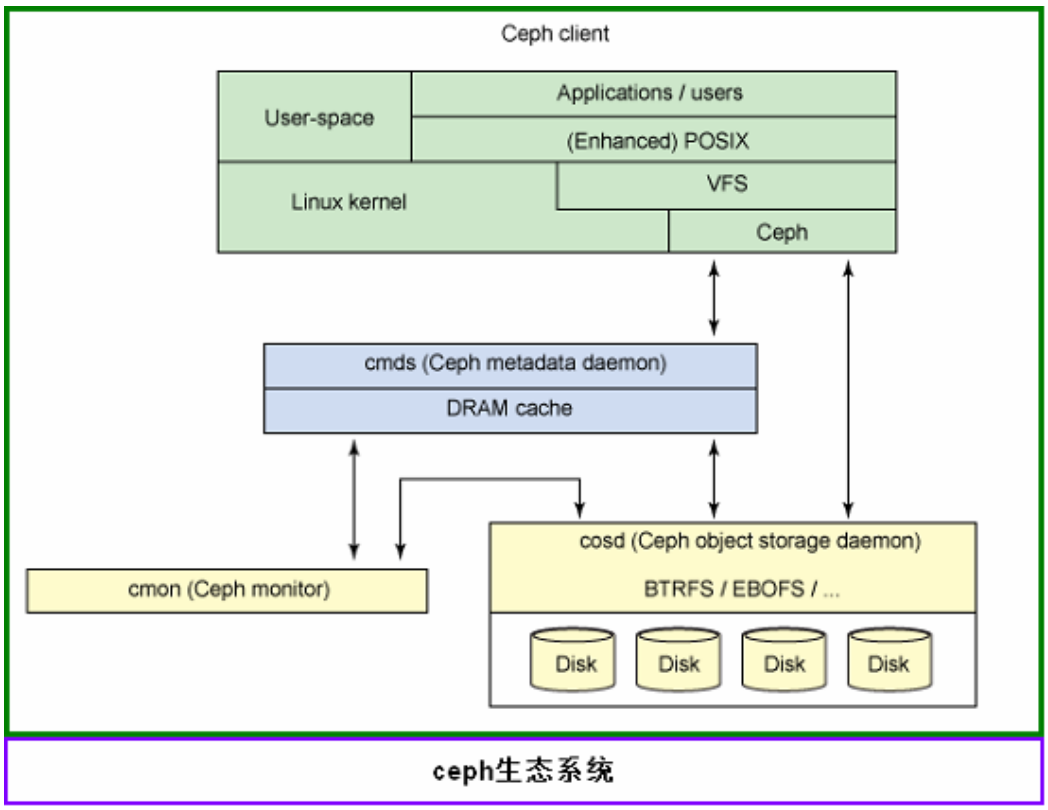
ceph基础原理梳理

---

ceph生态系统

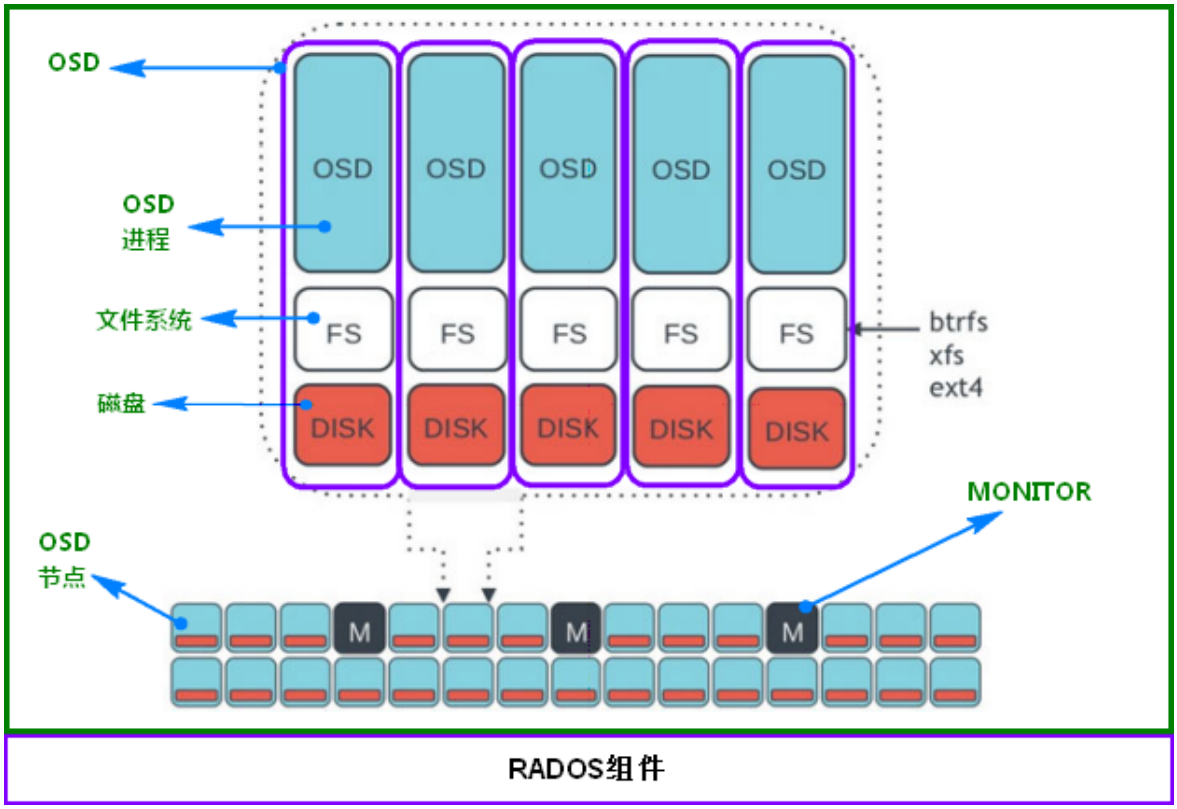


- client客户端：是cephFS文件系统的使用者
- mds元数据父进程：用于提供ceph集群中的元数据服务
- osd对象存储服务进程：用于提供数据和元数据的实际存储空间

- monitor监控进程：用于提供集群的管理功能

OSD结构

OSD主要功能：负责完成数据存储功能



1. OSD节点 ： 用于指定完成OSD功能的物理主机节点

2. OSD daemon ： 负责完成OSD的所有逻辑功能，包括当前OSD与monitor和其他OSD(事实上是其他OSD的deamon)通信以维护更新系统状态，与其他OSD共同完成数据的存储和维护，与client通信完成各种数据对象操作等等.

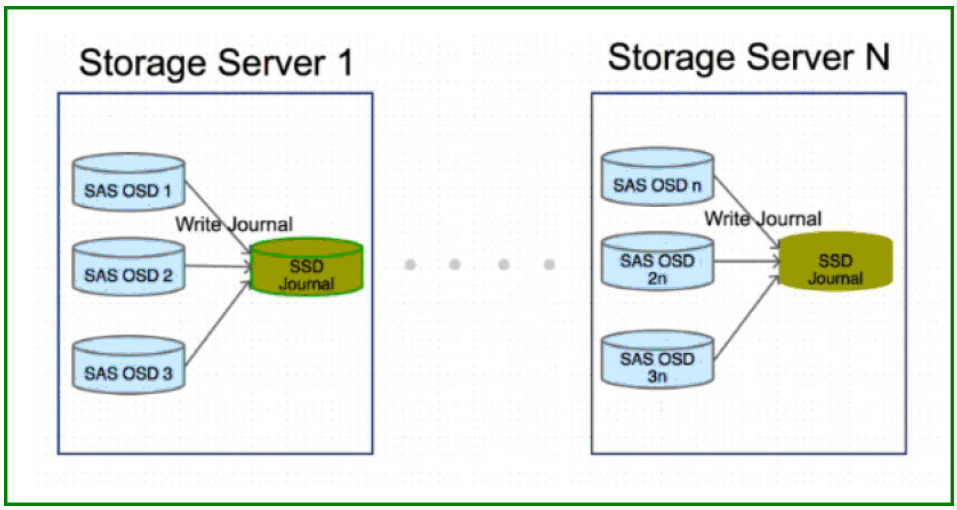
3. FS ： 在物理磁盘上创建的文件系统，用于为上层的OSD daemon提供存储数据的文件系统

4. DISK ： 可以看作是单块物理磁盘或物理磁盘中的一个分区

平时不加特指的OSD则表示该OSD为OSD daemon，FS和DISK三个部分组成的集合

ssd磁盘在ceph集群中的应用场景

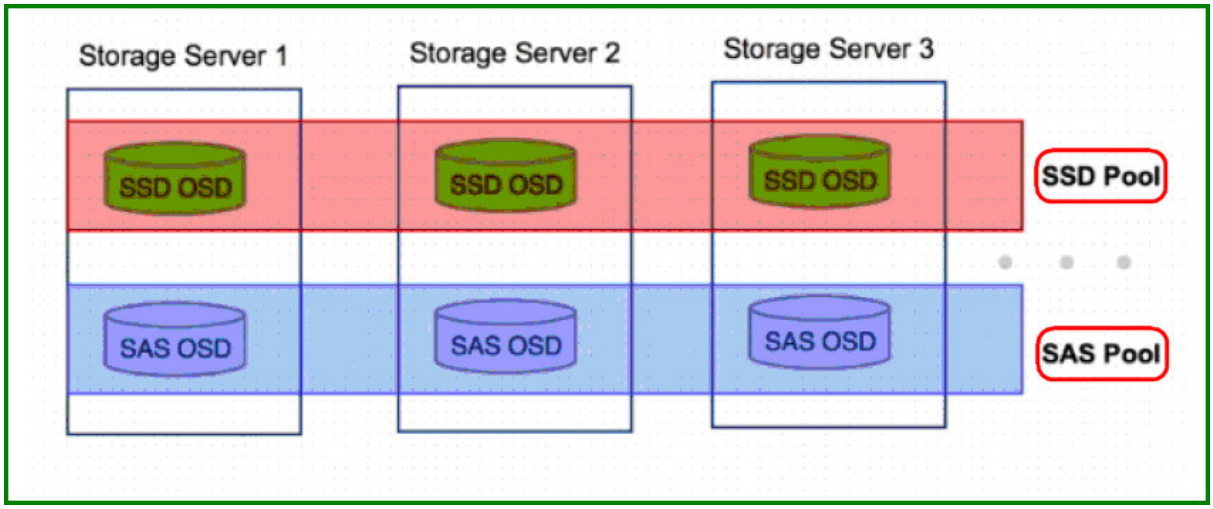
1. ssd作为日志盘使用



实现方式：osd日志是大量小数据块和随机IO写操作，可以提高ceph的整体性能

特性：不能完全发挥ssd的优势，不建议选择这种使用方法

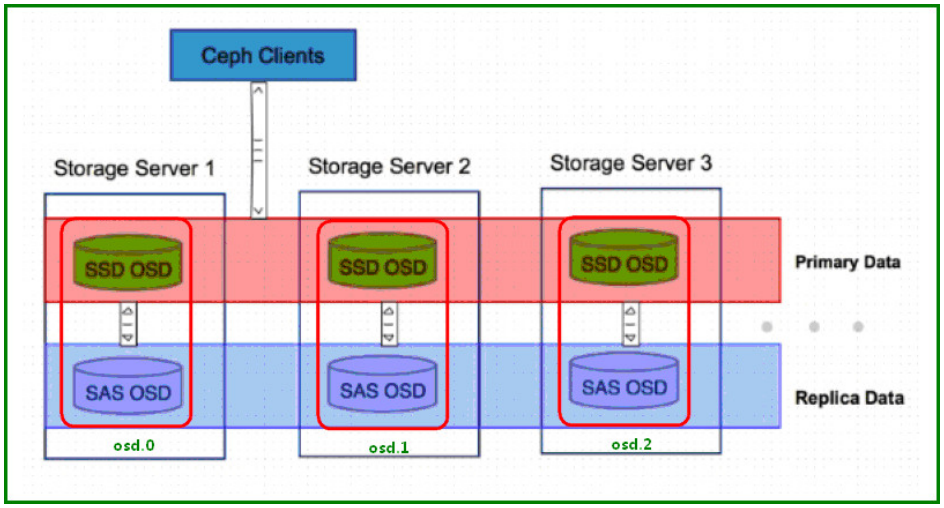
2. 与sas，sata混合使用，使用ssd作为单独的pool



实现方式：可以根据数据传输需求实现存储控制：对传输速度要求高的使用ssd pool，对传输速度要求低的使用sata或sas pool

特性：配置较为简单，能够对业务进行层级划分，公司中就是使用的这种方法

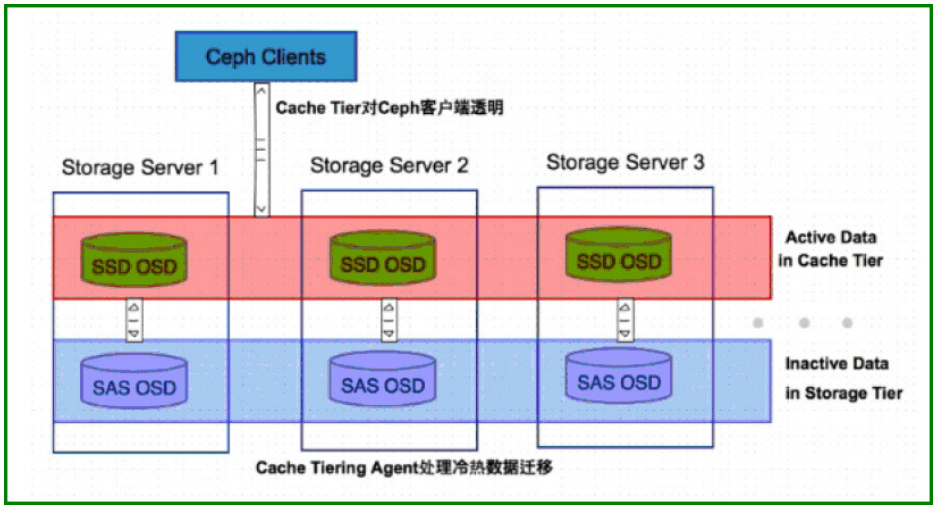
3. 将ssd与sata或sas构成同一个pool，通过配置crush算法实现将ssd中的osd作为primary osd



实现方式：通过配置crush算法将多副本中的主数据存储在位于ssd磁盘上的osd中

特性：可以大幅度提高读操作性能，ssd，sas或sata的部署位置都是规定好的，不能实现灵活配置

4.ssd作为ceph cache tiering结构中的cache层级



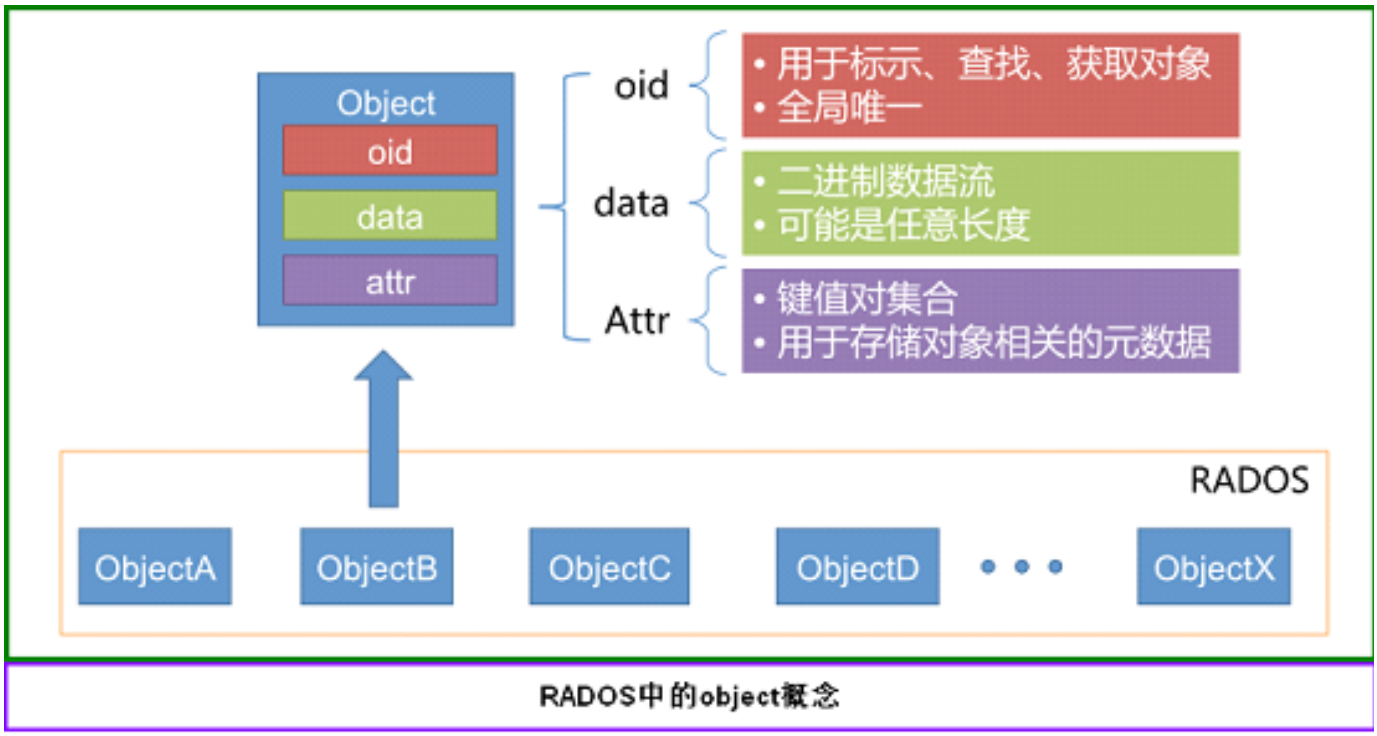
实现方式：使用ssd类型的磁盘构成一个pool用于完成cache缓存功能，使用sas或sata的磁盘构成的pool用于完成storage存储功能

特性：最为灵活的实现方式，性价比最高

---

object概念

ceph中的object概念



RADOS能够实现object对象存储功能

对象存储功能：能够实现将数据通路(数据读或写)和控制通路(元数据)分离，并且基于对象存储设备(Object-based Storage Device，OSD)构建存储系统，每个对象存储设备具有一定的智能，能够自动管理其上的数据分布.

US(个人理解)：RADOS是一个object对象存储系统，RADOS会将所有数据作为object进行处理，object在RADOS中都有指定的存储空间，因此一个大块data数据，RADOS会将该data划分为多个object，每个object存储在RADOS中表现形式对应为一个file文件

object组成部分

1. oid：用于标识每个object，每个object的oid都是全局唯一的，可以通过oid对对应object进行全局寻址
2. data：用于存储每个object中的数据
3. attr：用于存储每个object的元数据(时间戳，属主属组等信息)，元数据信息是通过key-value键值对方式进行存储的

---

client中的数据data在ceph集群中OSD节点中的读写过程

该过程主要分为两个阶段：

I. client需要通过相关信息，在本地计算出data需要存储的OSD节点的位置；

II. client中的数据在OSD节点上执行读写操作

I. client中的数据映射至指定OSD的过程

该过程主要通过两个步骤实现

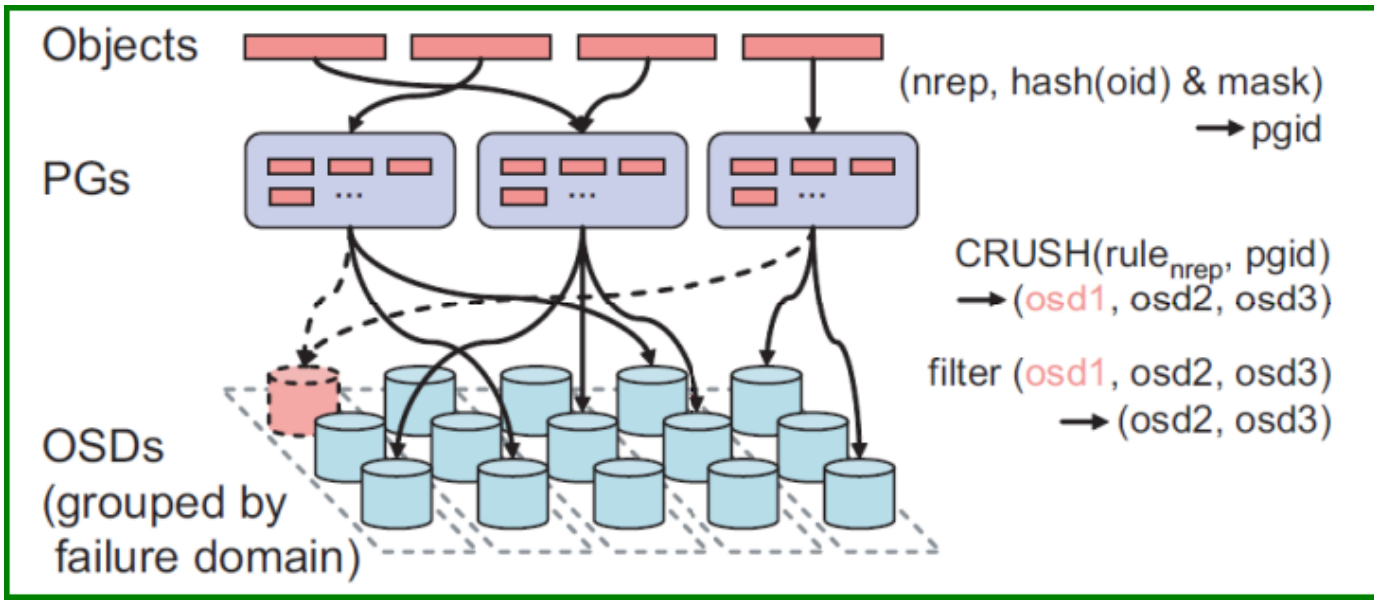
1 client取得ceph cluster信息过程

2. client在本地计算得到OSD位置过程

1. client取得ceph cluster信息过程

client会通过socket与ceph集群中mon节点进行通信，从mon节点获取cluster map数据，client通过cluster map在本地通过计算得到要存储数据的ceph集群中的OSD位置

2. client在本地计算得到OSD位置过程



2-1. data映射为object

过程 ： 将用户存储的data映射为RADOS中能够处理的object，实质就是将单个data分割成多个大小相同的object，这多个object在RADOS中变现为file文件格式

映射方法 ： 通过data的ino和non映射为object的oid

其中：

ino 是未进行分割操作的data的元数据，可以看作是data的唯一id；

ono 是该data进行分割操作后生成的某个object的序号；

每个object通过唯一的 oid 进行标识，每个oid是通过线性映射生成的，

每个data的ino应该保证惟一性，否则分割生成的object则响应无法保证惟一性

优点：让大小不同的data转换成多个大小相同的object，并通过RADOS进行管理；
对单个data进行串行处理可以转换成RADOS中对对应object的并行处理

2-2. object映射到PG

过程：将每个object独立映射到一个PG中去

映射方法 ： hash(oid) && mask --> pgid

US(个人理解) ：

1. 使用ceph指定的静态哈希函数对object的oid进行计算得到对应的哈希值hash(oid)，哈希值hash(oid)是一个近似均匀分布的伪随机值；

2. hash(oid)与mask进行按位与运算，得到最终的PG序号pgid；其中mask=m-1，m是给定的PG的总数，m应该为2的整数次幂

优点 ：

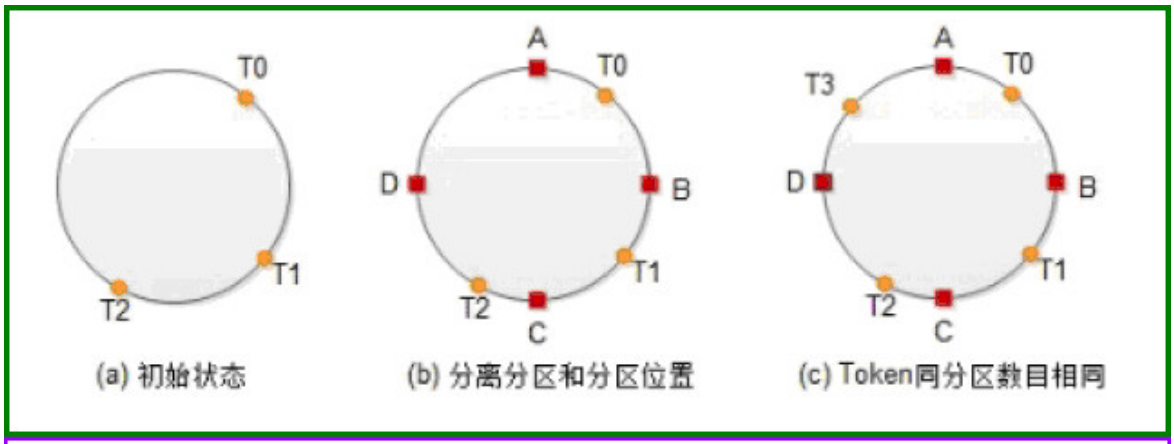
基于这一机制，当有大量object和大量PG时，RADOS能够保证object和PG之间的近似均匀映射. 又因为object是由file切分而来，大部分object的size相同，

这一映射最终保证了，各个PG中存储的object的总数据量近似均匀

2-3. PG映射到OSD

过程：将作为object的逻辑组织单元的PG映射到数据的实际存储单元ODS中

PG映射至单个osd



pg映射至单个osd原理

PG映射至单个osd的实质：用于完成类似一致性hash的过程

a. 保证全部PG能够均匀映射(分布)至各个osd上

b. 保证当某个osd宕机后，在PG迁移至其他osd的过程能够平滑进行，而不会产生较大的偏移

PG映射至多个osd

映射方法：CRUSH(pgid) --> (osd1, osd2, osd3)；filter(osd1, osd2, osd3) --> (osd2, osd3)

a. RADOS使用一种名为CRUSH的算法，将pgid代入其中CRUSH算法表达式中，然后根据事先定义的rule算法规则计算得到一组n个OSD，这n个OSD共同负责存储和维护一个PG中的所有object；n在生产环境中通常为3，用于表示一个object的存储个数

b. filter过滤器会根据这n个OSD的当前状态，从这n个OSD中选择当前可用的OSD来存储object

具体到每个OSD，则由其上运行的OSD deamon负责执行映射到本地的object在本地文件系统中的存储，访问，元数据维护等操作.

对本次映射过程的影响因素：

a. 当系统中的OSD状态，数量发生变化时，cluster map可能发生变化，而这种变化将会影响到PG与OSD之间的映射.

b. rule算法规则

只有当系统状态(cluster map)和runle算法规则不发生变化时，PG和ODS之间的映射关系才是固定不变的.

实际使用过程中，rule算法规则已经配置通常不会改变，系统状态(cluster map)的改变通常是由于设备损坏或存储集群规模发生改变造成的

US(个人理解)：

1. object映射到PG过程中，使用hash算法对每个object的oid运算，得到的hash(oid)来实现每个object映射到指定PG中

该过程中是基于每个object的oid进行映射，因此可以实现对每个object映射到对应的PG中，该过程可以看作是一个动态映射过程

2. PG映射到OSD过程中，使用CRUSH算法对每个PG的pgid和事先定义的rule来计算得出多个OSD

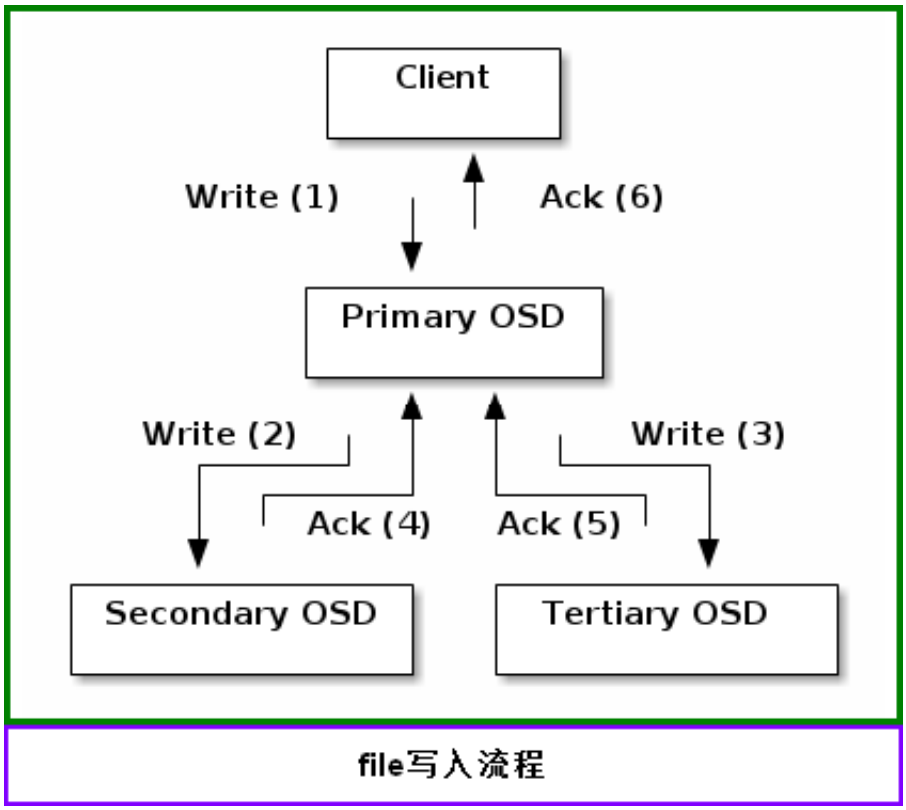
该过程中式基于每个PG的pgid进行映射，因此可以实现将已映射到PG中的object映射到固定对应的OSD中，该过程可以看作是一个静态映射过程

II. client的数据data在OSD节点上进行读写操作

前提条件 ：

1. 用户写入file较小，只能被划分为1个object
2. 系统中1个PG会被映射到3个OSD上

client写入object过程



1. client通过在本地计算，得到了存储指定object的3个OSD节点位置

这三个OSD具有各自不同的序号，序号最靠前的OSD就是该组中的primary OSD，其他的两个依次为secondary OSD和tertiary OSD

2. client会直接和primary OSD进行通信，发起write操作
3. primary OSD收到来自client的write操作后，primary OSD本身分别向secondary OSD和tertiary OSD发起write操作
4. secondary OSD和tertiary OSD完成write操作后，它们会向primary OSD发送对于write操作的确认信息
5. primary OSD收到它们的确认信息后，primary OSD本身就会执行完成write操作，prymary OSD再向client发送write操作的确认信息

write流程的优点：

1. write写入流程本质上可以实现数据写入的可靠性，尽量避免数据丢失
2. client只需要向primary OSD发送write操作，可以降低client和ceph集群之间的数据传输

write流程的缺点：

primary OSD只有收到其他OSD的确认信息后，才会向client发送确认信息，因此会产生较长时间的延迟

产生了ceph对client的两段式确认：

a. 当各个OSD将数据写入各自的内存缓冲区后，ceph就会向client发送一次确认，可以保证client可以及时向下执行队列中的命令

b. 当各个OSD将数据写入磁盘后，ceph会向client发送一个最终确认信号，client收到最终确认信号后会根据需要删除本地数据

从OSD的角度来看，由于同一个OSD在不同PG中的排列次序不同，因此同一个OSD的工作压力可能会分摊，避免单个OSD成为性能瓶颈

client读取object流程

1. client需要完成相同的寻址过程，来获取存储指定object的3个OSD位置

2. 读取数据时，client只需要与primary OSD进行通信来读取指定object，而不需要与其他两个OSD进行通信

CRUSH算法

ceph中的CRUSH算法用于实现根据PG的pgid来映射到多个OSD

CRUSH算法包括从cluster map和rule两个部分

cluster map

1. cluster map组成部分

cluster map用于描述ceph集群的拓扑结构，可以看作是整个cluster集群位置关系，可以看作是cluster的地图，cluster map由5张表组成：monitor map，OSD map，PG map，CRUSH map，MDS map

1-1. moitor map 包括每个cluster的fsid，position，name address，port四种主要信息，还会额外提供current epoch信息

可以使用命令 # ceph mon dump 来查看monitor map

1-2. OSD map 包含信息：cluster fsid，该map的时间戳信息，pool列表，replica size，PG numbers，OSD列表，
每个OSD信息(osd状态和osd权重)

osd在osd map中的状态变化：

osd启动过程：osd向mongitor申请加入ceph集群 --> osd置为in状态 --> monitor将该osd置为up状态 -->
协商成功后新加入的osd与其他osd进行通信，并接受monitor分配来的pg

osd宕机过程：与宕机osd通过heartbeat进行通信的其他osd将宕机osd置为down状态 --> 该osd上分配的primary PG会以replica
PG的身份分配至新的osd上(原有的replica PG会在对应的osd上升级为primary PG)

可以使用命令 # ceph osd map 来查看OSD map信息

1-3. PG map 包含信息：PG version，该map的时间戳信息，OSD map的最新版本，full ratios，以及每个PG的详细信息：pgid，PG运行状态，
PG执行状态，PG状态，每个pool的数据使用信息

PG状态在PG map中的状态变化：osd发生故障 --> monitor将宕机osd上角色为primary PG对应的replica PG升级为primary PG -->
原来处于primary角色的PG会处于degrade状态 --> (宕机的)osd置为out状态 --> PG会以replica PG的角色分配至新的osd上

1-4. CRUSH map 包含信息：存储设备列表，failure domain失效域，rules算法规则

可以使用命令 来查看CRUSH map信息

1-5. MDS map 包含信息：MDS map epoch，元数据信息池，元数据服务器列表，元数据服务器状态

可以使用命令 # ceoh mds dump 来查看MDS map信息

每个map会维护它本身的迭代版本和操作记录，monitor节点会保存cluster map的最新副本

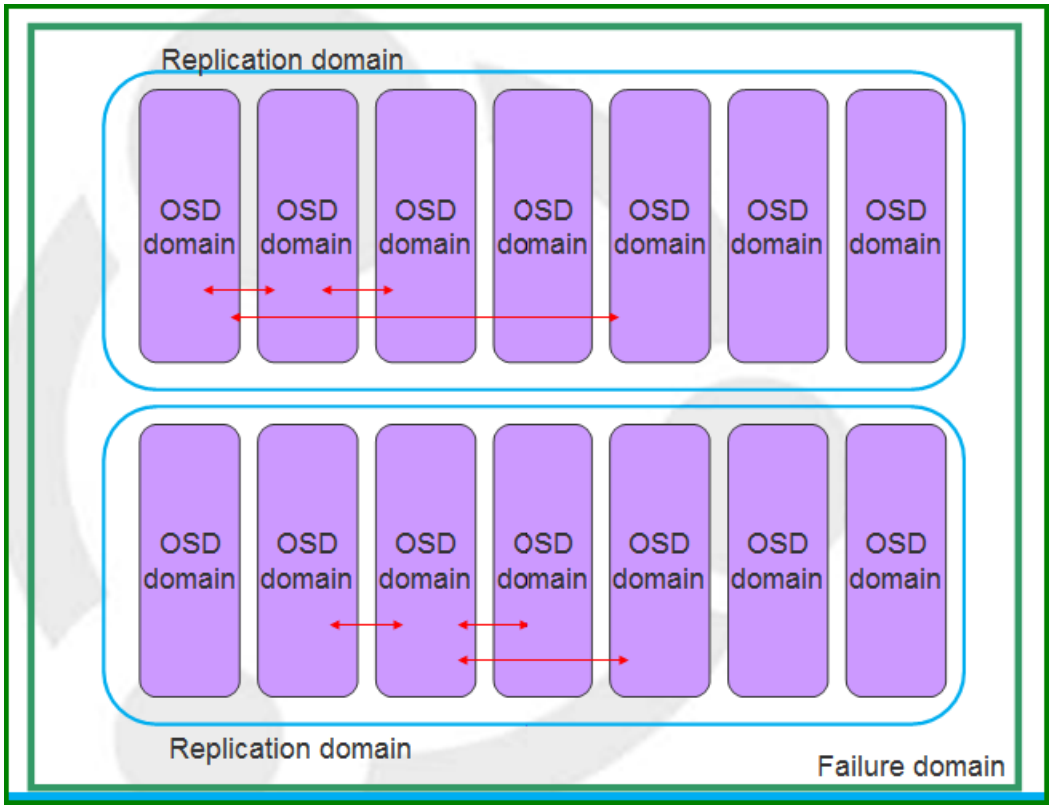
2. cluster map类型

2-1. 物理结构cluster map：可以用于描述实际物理设备的映射关系

层级关系：数据中心 --> 机房 --> 机架 --> 物理主机 --> osd

2-2. 逻辑结构cluster map ：可以用于描述集群在逻辑层级的映射关系

层级关系: Failure-domain --> replica-domain --> osd-domain --> osd



一个failure domain中可以含有多个replicat domain, 每个replica domain中可以含有多个osd domain

多个replica domain之间不能进行数据复制, 单个replica domain内的多个osd domain之间可以进行数据复制

rules

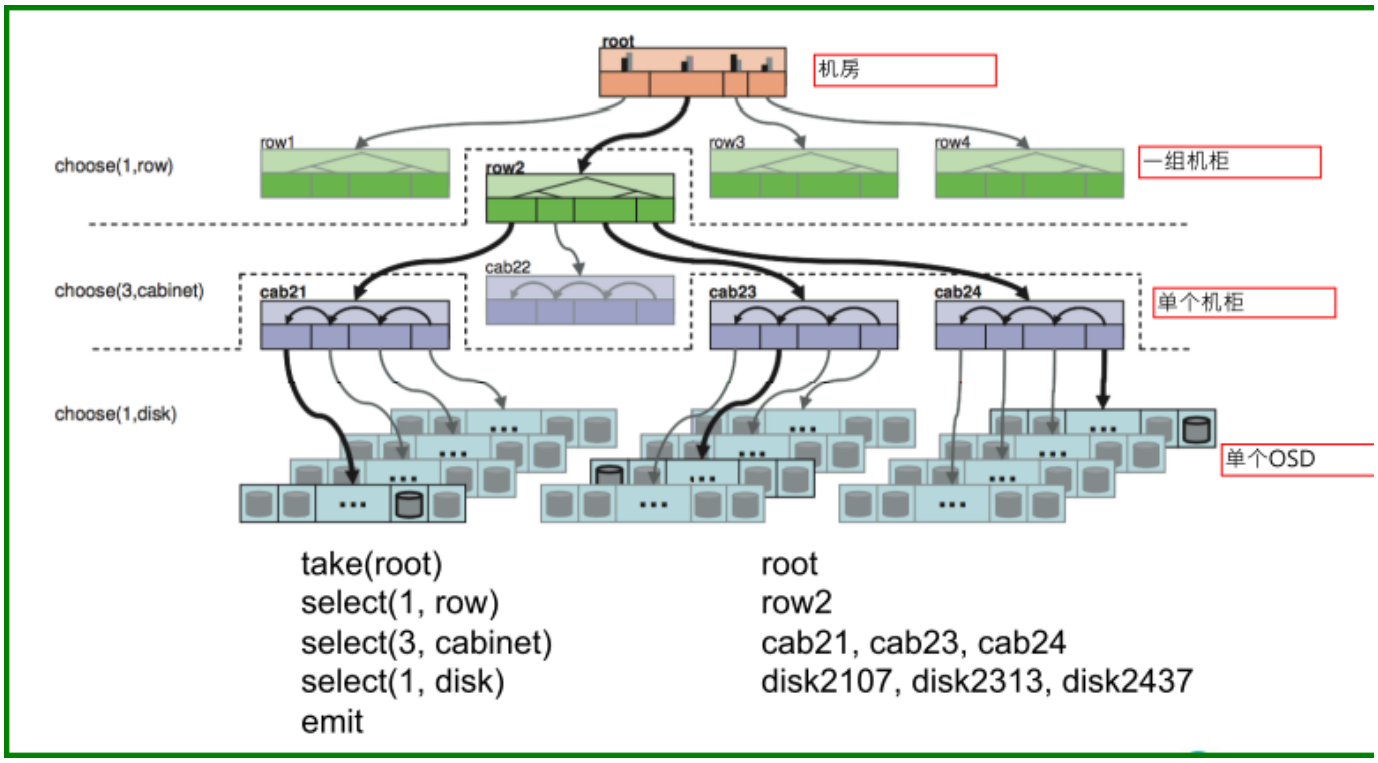
rules用于定义CRUSH算法的具体规则, rule通过重复执行take(bucketID)和select(n,bucket type)两类操作来选取保存的副本位置

1. take(bucketID): 用于从指定的bucketID中选取副本位置, EX: 可以指定从某台机架上选取副本位置, 实现将不同副本隔离在不同故障域

2. select(n, bucketType): 用于在给定的bucket下选取n个类型为bucketType的bucket, 选取bucket主要考虑层级结构中的节点容量, 以及节点离线或加入集群时的数据迁移量

3. emit: 用于表示算法执行完成, 执行退出

CRUSH算法过程实例

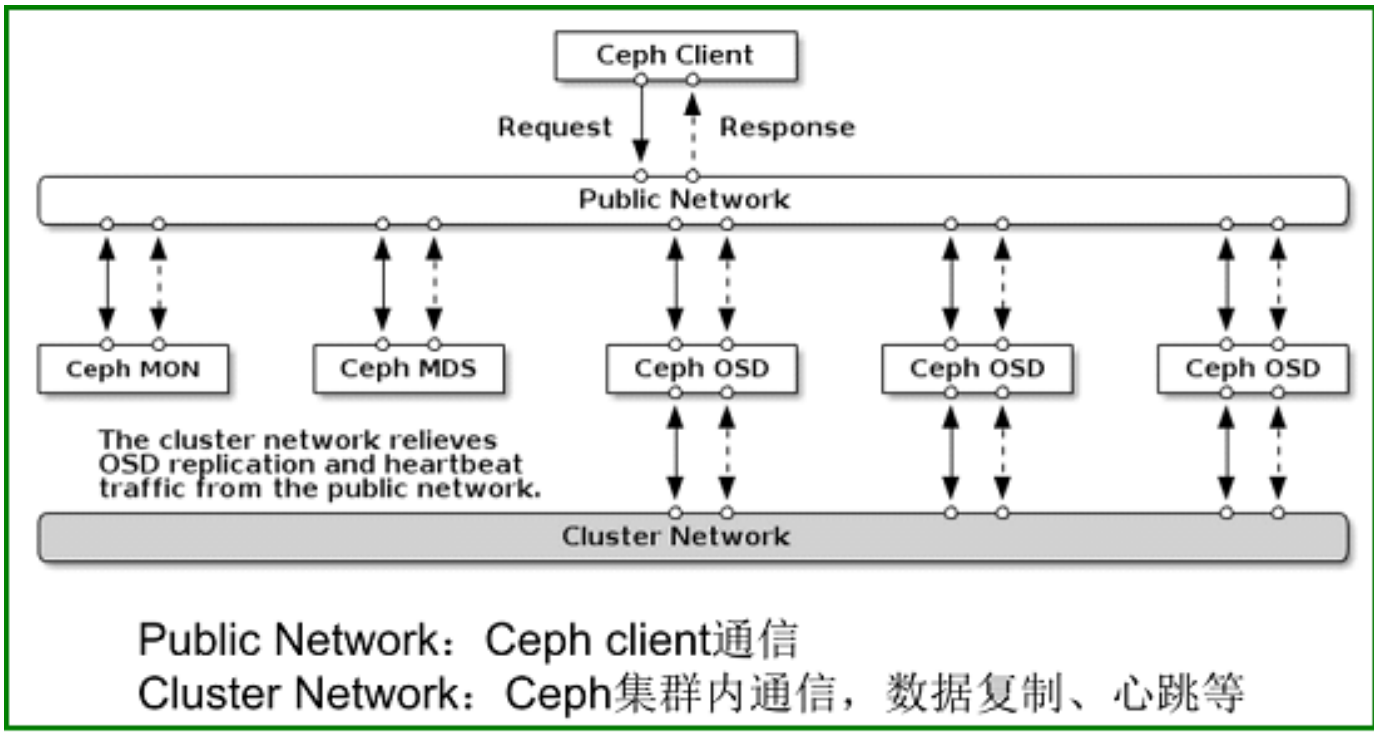


1. take(root)：用于选择机房
2. choos(1,row)：用于从选定的机房中的多组机柜中选择一组机柜
3. choose(3, cabinet)：用于从选定的一组机柜中选择3个机柜
4. choose(1, disk)：用于从选定的3个机柜中分别选择3个OSD来存储数据
5. emit：算法执行完成

---

ceph网络

ceph网络逻辑架构

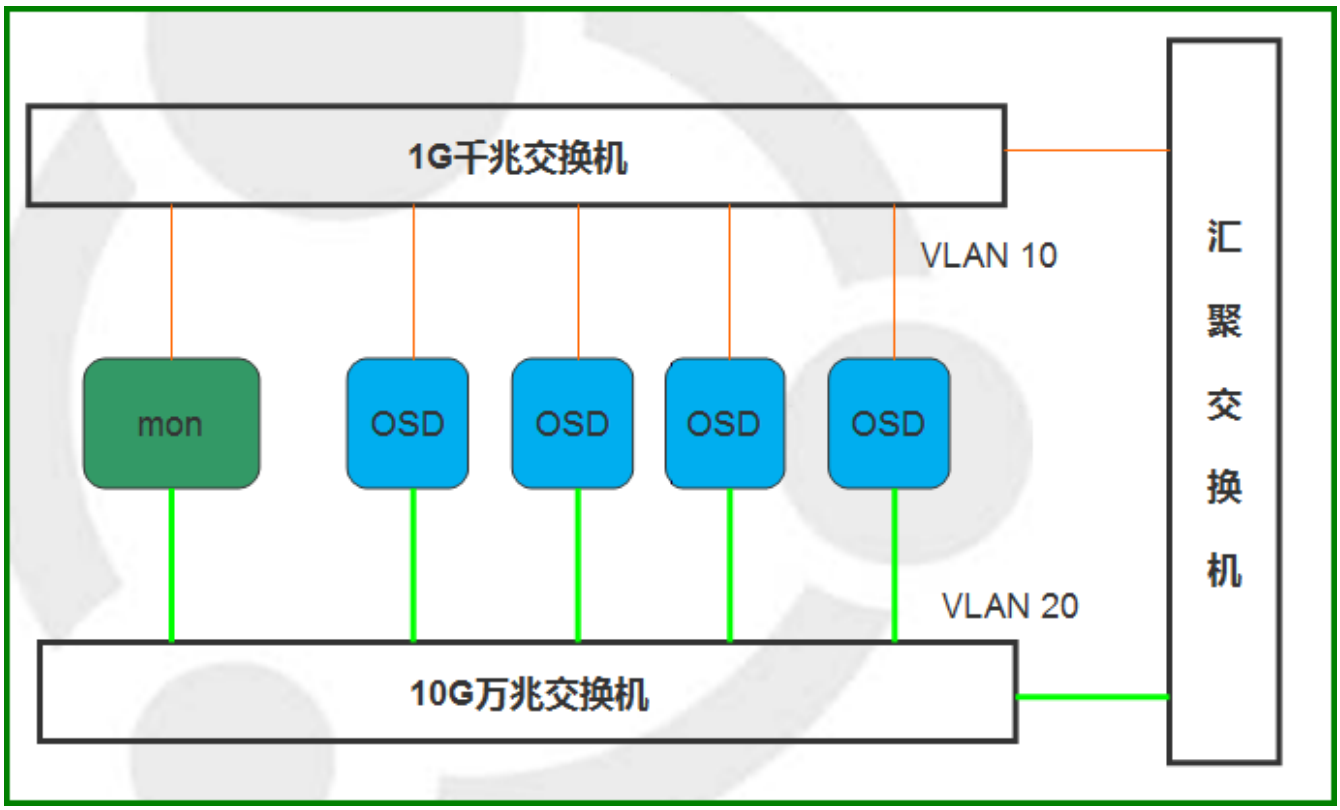


ceph集群中根据传输的数据类型，将网络分为两种类型Public Network和Cluster Network

1. Public Network ：用于承载client和ceph cluster通信数据，主要包括client和mon之间关于cluster map的数据通信，client和OSD之间对于数据读写操作的数据通信

2. Cluster Network ：用于承载ceph cluster内部通信数据，主要包括：不同OSD节点之间的数据复制，不同OSD节点之间与mon和OSD之间的心跳信息

ceph网络物理拓扑



ceph集群根据物理网络连接类型，可以将物理网络分为管理网络和数据网络

物理网络拓扑中通过VLAN方式对管理网络和数据网络进行子网隔离

1. 管理网络 ： 使用1G千兆网络接口，用于client和mon之间关于cluster map的数据通信

2. 数据网络 ： 使用10G万兆网络接口，用于client和OSD之间关于object读写操作的数据通信，OSD节点之间的数据复制，不同OSD节点之间心跳信息，OSD与mon之间心跳信息，ceph集群内内部其他数据通信

US(个人理解)：

Public Network，Cluster Network 与 管理网络，数据网络这两组概念不是一一对应的

这两组概念的关系为：

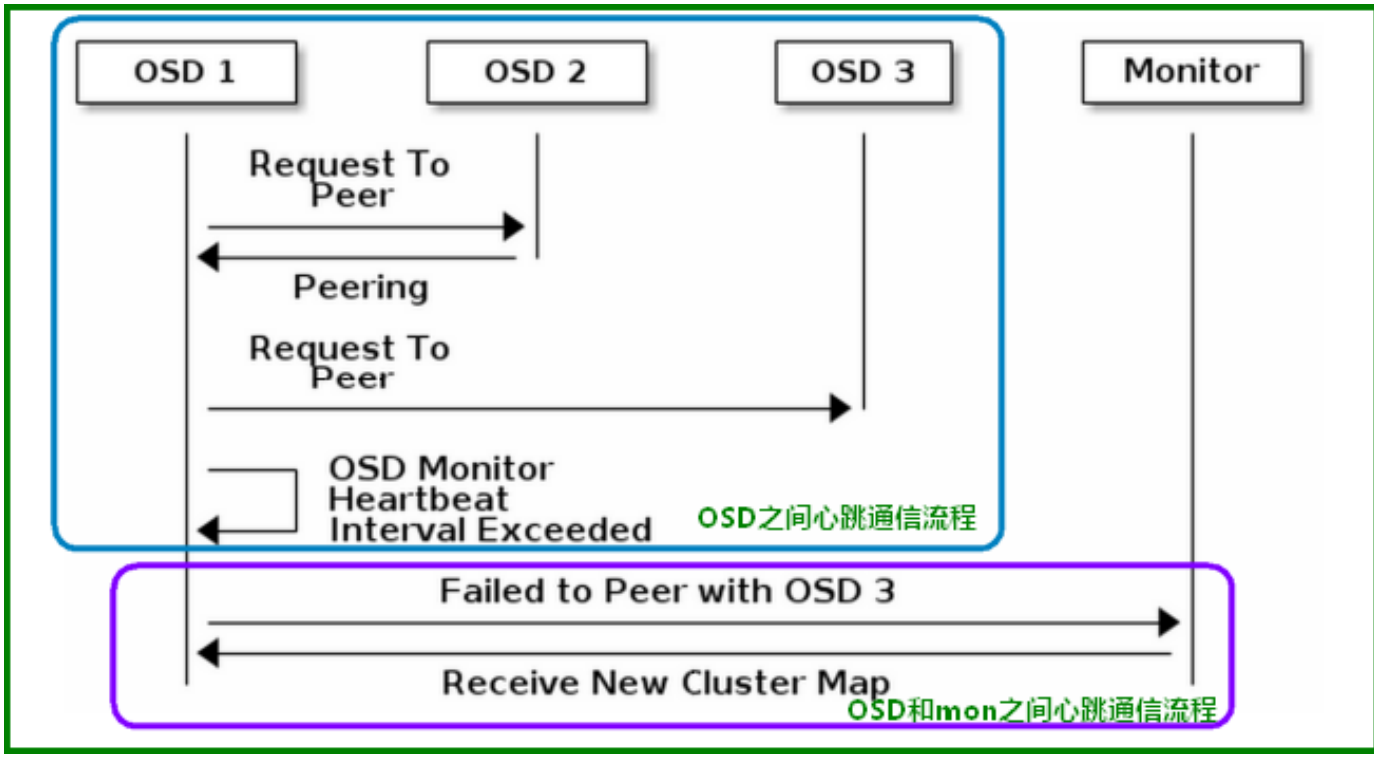
管理网络只承载了Public Network中的client与mon的数据通信部分

数据网络承载了Public Netwrok中的client与OSD的数据通信部分和Cluster Network中的全部数据通信部分

---

ceph集群心跳实现方式

不同OSD节点之间心跳通信



不同osd节点会与就近的osd节点进行心跳方式的通信，用来确认对端节点的状态是up的

当osd节点超出指定时间间隔没有收到对端osd节点的应答报文，此时该osd节点会认为对端osd节点的状态时down的，

当前osd节点会向monitor节点发送信息报告指定osd节点的状态时down的，

monitor节点会根据收到对指定osd节点状态为down的状态信息，来将该osd节点标记为down

osd节点间心跳信息的相关参数：

osd_heartbeat_interval=3 ： osd节点之间正常心跳通信的时间间隔为3s

osd_heartbeat_grace=7 ： osd节点将对端osd节点标记为down状态时等待的最大时间间隔为7s

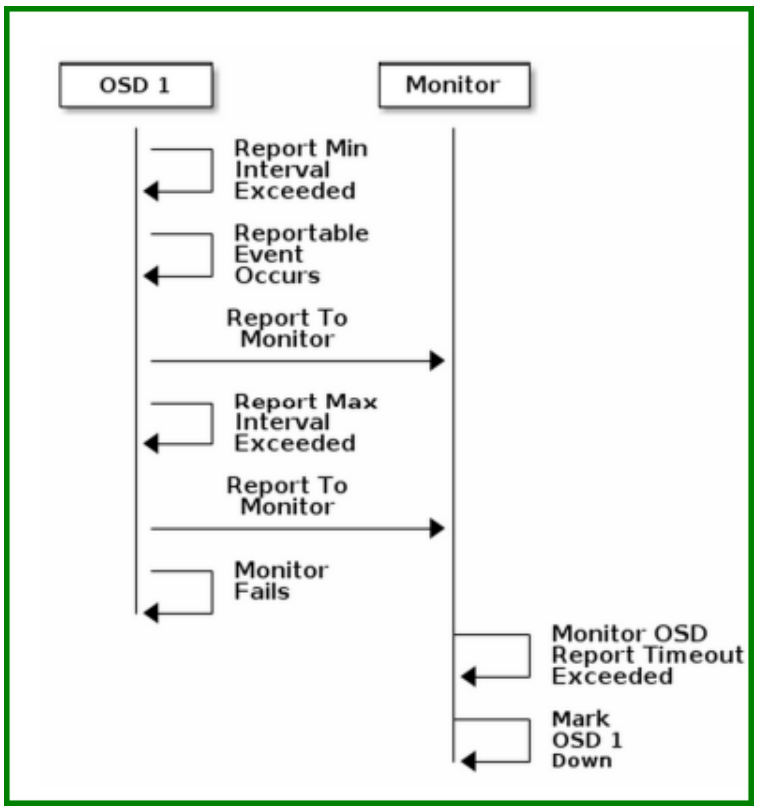
monitor节点判断osd节点状态的心跳信息参数：

mon_osd_min_down_reports=3 ： 当monitor节点收到最少3个报文时，会将指定osd节点标记为down

mon_osd_min_down_reporters=1 ： 当monitor节点收到最少来自1个osd节点的报文时，会将指定osd节点标记为down

monitor节点将指定osd节点标记为down状态时，需要同时满足以上两种条件

OSD与monitor之间进行心跳通信



osd节点会向monitor节点之间进行心跳通信，来向monitor节点表示osd节点本身的状态

osd节点向monitor节点发送心跳信息的通信方式有两种类型：主动心跳和被动心跳

1. 主动心跳 ： 当osd有事件需要向monitor发送通知情况下，osd会以大于最小报告时间间隔向monitor发送心跳信息

2. 被动心跳 ： 当osd没有事件需要先monitor发送通知情况下，osd会以小于最大报告时间间隔向monitor发送心跳信息

monitor端在超出最大报告时间间隔的情况下，没有收到指定osd节点发送的心跳信息，会将对应osd节点标记为down

osd节点和monitor节点发送心跳信息的参数

osd_mon_report_interval_max=120 ： osd节点向monitor节点发送心跳信息的最大时间间隔为120s

osd_mon_report_interval_min=5 ： osd节点向monitor节点发送心跳信息的最小时间间隔为5s

monitor节点接收osd心跳信息的参数

mon_osd_report_timeout=900 ： monitor节点允许等待osd节点发送心跳信息的最大等待时间间隔为900s