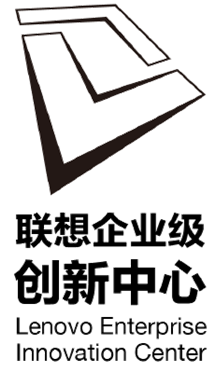


|  |
| --- |
|  |



**分布式存储Ceph**

**架构及性能调优实践**

作者：张云博、孙鑫

更新时间： 21 December 2016

目录

[一、Ceph基本介绍 2](#_Toc474767783)

[1.1 Ceph简介 2](#_Toc474767784)

[1.2 Ceph发展历程 3](#_Toc474767785)

[1.3 Ceph优缺点 4](#_Toc474767786)

[1.4 Ceph架构设计 4](#_Toc474767787)

[1.4.1 组件 4](#_Toc474767788)

[1.4.2 映射 6](#_Toc474767789)

[1.5 企业里的典型场景 8](#_Toc474767790)

[1.5.1 高性能场景 8](#_Toc474767791)

[1.5.2 通用场景 8](#_Toc474767792)

[1.5.3 大容量场景 8](#_Toc474767793)

[二、Ceph性能调优 8](#_Toc474767794)

[2.1 硬件选型 8](#_Toc474767797)

[2.1.1 CPU 9](#_Toc474767798)

[2.1.2 RAM内存 9](#_Toc474767799)

[2.1.3 数据存储 9](#_Toc474767800)

[2.1.4 网络 9](#_Toc474767801)

[2.1.5 硬盘 9](#_Toc474767802)

[2.1.6 Ceph OSD日志盘 9](#_Toc474767803)

[2.2 性能调优 10](#_Toc474767804)

[2.2.1 硬件优化 10](#_Toc474767805)

[2.2.2 软件层面 11](#_Toc474767806)

[2.2.3 Ceph层面优化 12](#_Toc474767807)

*东软集团有限公司*

# 一、Ceph基本介绍

## Ceph简介

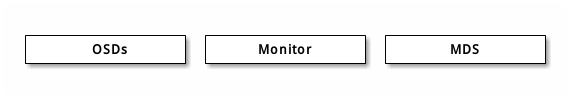
Ceph的官方网站Ceph.com上用如下这句话简明扼要地定义了Ceph：

“Ceph is a unified, distributed storage system designed for excellent performance, reliability and scalability.”

也即，Ceph是一种为优秀的性能、可靠性和可扩展性而设计的统一的、分布式的存储系统。应该说，这句话确实点出了Ceph的要义，可以作为理解Ceph系统设计思想和实现机制的基本出发点。在这个定义中，应当特别注意“存储系统”这个概念的两个修饰词，即“统一的”和“分布式的”。

具体而言，“统一的”意味着Ceph可以一套存储系统同时提供对象存储、块存储和文件系统存储三种功能，以便在满足不同应用需求的前提下简化部署和运维。而“分布式的”在Ceph系统中则意味着真正的无中心结构和没有理论上限的系统规模可扩展性。在实践当中，Ceph可以被部署于上千台服务器上。截至2013年3月初，Ceph在生产环境下部署的最大规模系统为Dreamhost公司的对象存储业务集群，其管理的物理存储容量为3PB

不管你是想为云平台提供Ceph 对象存储和/或 [Ceph 块设备](http://docs.ceph.org.cn/glossary/#term-38)，还是想部署一个 [Ceph 文件系统](http://docs.ceph.org.cn/glossary/#term-45)或者把 Ceph 作为他用，所有 [Ceph 存储集群](http://docs.ceph.org.cn/glossary/#term-21)的部署都始于部署一个个 [Ceph 节点](http://docs.ceph.org.cn/glossary/#term-13)、网络和 Ceph 存储集群。 Ceph 存储集群至少需要一个 Ceph Monitor 和两个 OSD 守护进程。而运行 Ceph 文件系统客户端时，则必须要有元数据服务器（ Metadata Server ）。



* **Ceph OSDs**: Ceph OSD 守护进程（ Ceph OSD ）的功能是存储数据，处理数据的复制、恢复、回填、再均衡，并通过检查其他OSD 守护进程的心跳来向 Ceph Monitors 提供一些监控信息。当 Ceph 存储集群设定为有2个副本时，至少需要2个 OSD 守护进程，集群才能达到 active+clean 状态（ Ceph 默认有3个副本，但你可以调整副本数）。
* **Monitors**: Ceph Monitor维护着展示集群状态的各种图表，包括监视器图、 OSD 图、归置组（ PG ）图、和 CRUSH 图。 Ceph 保存着发生在Monitors 、 OSD 和 PG上的每一次状态变更的历史信息（称为 epoch ）。
* **MDSs**: [Ceph 元数据服务器](http://docs.ceph.org.cn/glossary/#term-63)（ MDS ）为 [Ceph 文件系统](http://docs.ceph.org.cn/glossary/#term-45)存储元数据（也就是说，Ceph 块设备和 Ceph 对象存储不使用MDS ）。元数据服务器使得 POSIX 文件系统的用户们，可以在不对 Ceph 存储集群造成负担的前提下，执行诸如 ls、find 等基本命令。

Ceph 把客户端数据保存为存储池内的对象。通过使用 CRUSH 算法， Ceph 可以计算出哪个归置组（PG）应该持有指定的对象(Object)，然后进一步计算出哪个 OSD 守护进程持有该归置组。 CRUSH 算法使得 Ceph 存储集群能够动态地伸缩、再均衡和修复。

## Ceph发展历程

● 2004年06月：第一个commit

● 2006年11月：CRUSH 论文

● 2008年01月: v0.1版本

● 2010年12月：Qemu Block Device Driver for RBD

● 2012年02月：v0.42版本

● 2012年05月：Inktank成立

● 2012年12月：支持RBD Clone (format v2)

● 2013年03月：v0.58版本，Ceph Monitor的架构被重写

● 2013年05月：v0.61版本

● 2014年04月：Inktank被RedHat收购

● 2014年05月：v0.80版本

● 2015年02月 v0.87.1 Giant

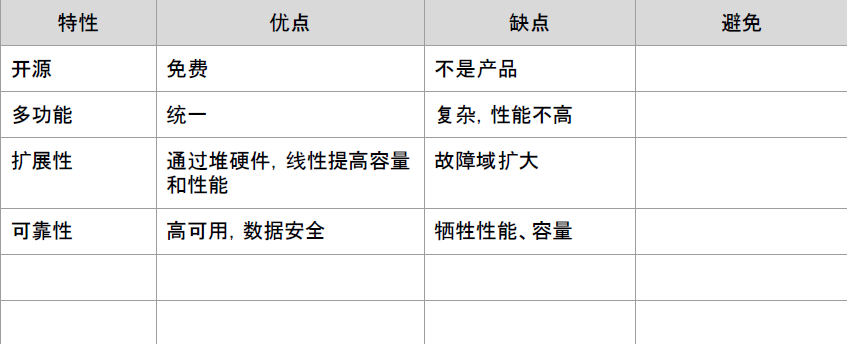
● 2015年04月 V0.94（LTS）Hammer

● 2015年05月 V9.00 Infernalis

● 2015年11月 V10.0.0 Jewel

● 2016年03月 V10.2.0 Jewel

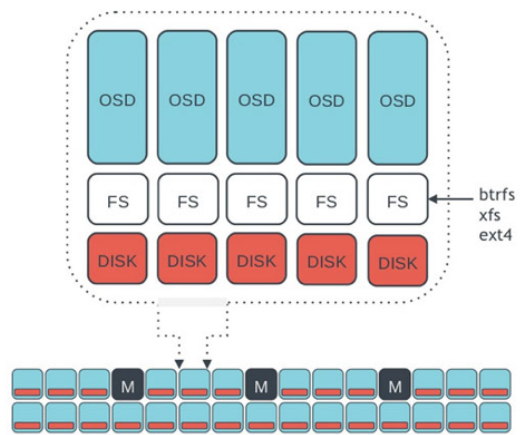
## Ceph优缺点



## Ceph架构设计

### 组件

**Ceph = 提供Block、File、Object接口的统一存储系统**

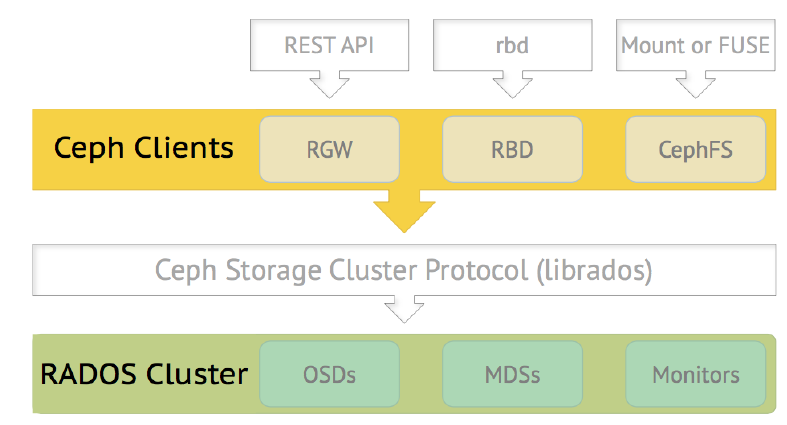
****

Ceph的底层是RADOS，它的意思是“A reliable, autonomous, distributed object storage”。 RADOS由两个组件组成：

* OSD： Object Storage Device，提供存储资源。
* Monitor：维护整个Ceph集群的全局状态。

RADOS具有很强的扩展性和可编程性，Ceph基于RADOS开发了  
Object Storage、Block Storage、FileSystem。Ceph另外两个组件是：

* MDS：用于保存CephFS的元数据。
* RADOS Gateway：对外提供REST接口，兼容S3和Swift的API。

****

### 映射

Ceph的命名空间是 (Pool, Object)，每个Object都会映射到一组OSD中(由这组OSD保存这个Object)：

**(Pool, Object) → (Pool, PG) → OSD set → Disk**

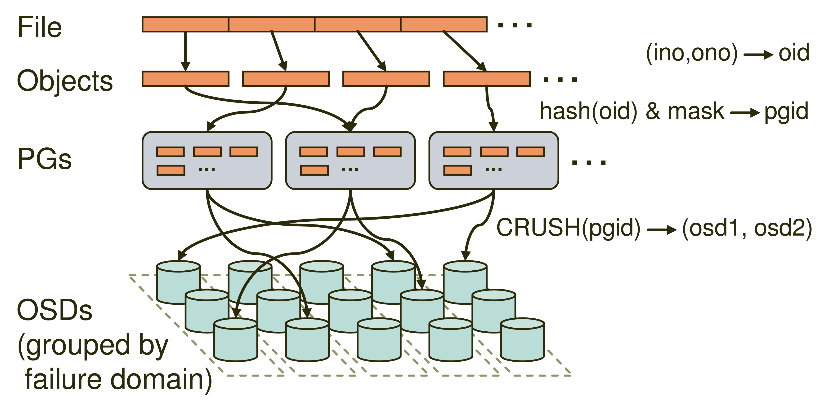
Ceph中Pools的属性有：

* Object的副本数
* Placement Groups的数量
* 所使用的CRUSH Ruleset

在Ceph中，Object先映射到PG(Placement Group)，再由PG映射到OSD set。每个Pool有多个PG，每个Object通过计算hash值并取模得到它所对应的PG。PG再映射到一组OSD（OSD的个数由Pool 的副本数决定），第一个OSD是Primary，剩下的都是Replicas。

数据映射(Data Placement)的方式决定了存储系统的性能和扩展性。(Pool, PG) → OSD set 的映射由四个因素决定：

* CRUSH算法：一种伪随机算法。
* OSD MAP：包含当前所有Pool的状态和所有OSD的状态。
* CRUSH MAP：包含当前磁盘、服务器、机架的层级结构。
* CRUSH Rules：数据映射的策略。这些策略可以灵活的设置object存放的区域。比如可以指定 pool1中所有objecst放置在机架1上，所有objects的第1个副本放置在机架1上的服务器A上，第2个副本分布在机架1上的服务器B上。 pool2中所有的object分布在机架2、3、4上，所有Object的第1个副本分布在机架2的服务器上，第2个副本分布在机架3的服 器上，第3个副本分布在机架4的服务器上。



这种数据映射的优点是：

* 把Object分成组，这降低了需要追踪和处理metadata的数量(在全局的层面上，我们不需要追踪和处理每个object的metadata和placement，只需要管理PG的metadata就可以了。PG的数量级远远低于object的数量级)。
* 增加PG的数量可以均衡每个OSD的负载，提高并行度。
* 分隔故障域，提高数据的可靠性。

## 企业里的典型场景

### 高性能场景

这种场景类型的亮点在于它在低TCO下每秒拥有最高的IOPS。典型的做法是使用包含了更快的SSD硬盘、PCIe SSD、NVMe做数据存储的高性能节点。通常用于块存储，也可以用在高IOPS的工作负载上。

### 通用场景

这种场景类型的亮点在于高吞吐量和每吞吐量的低功耗。通用的做法是拥有一个高带宽、物理隔离的双重网络，使用SSD和PCIe SSD做OSD日志盘。这种方法常用于块存储，如果你的应用场景需要高性能的对象存储和文件存储，也可以考虑使用。

### 大容量场景

这种场景类型的亮点在于数据中心每TB存储的低成本，以及机架单元物理空间的低成本。也被称为经济存储、廉价存储、存档/长期存储。通用的做法是使用插满机械硬盘的密集服务器，一般是12～20台服务器，每台服务器4～6TB的物理硬盘空间。通常用于低功耗、大存储容量的对象存储和文件存储。

# 二、Ceph性能调优



## 硬件选型

把握一个原则：Ceph的硬件选型需要根据存储需求和企业的使用场景来制定。

企业需要什么渴望什么：TCO低、高性能、高可靠

一般企业使用Ceph的历程：硬件选型－部署调优－性能测试－架构灾备设计－部分业务上线测试－运行维护（故障处理、预案演练等）。

### CPU

Ceph OSD运行RADOS服务，需要通过CRUSH来计算数据的存放位置，复制数据，以及维护Cluster Map的拷贝。通常建议每个OSD进程至少有一个CPU核。Metadata和Monitors？

### RAM内存

OSD在日常操作时不需要过多的内存（如每进程500MB）；但是，在执行恢复操作时，就需要大量的内存（如每进程每TB数据需要约1GB内存）。通常来说，内存越多越好。

### 数据存储

规划数据存储时要考虑成本和性能的权衡。进行系统操作，同时多个后台程序对单个驱动器进行读写操作会显著降低性能，也有文件系统的限制考虑。

### 网络

网卡能处理所有OSD硬盘总吞吐量，所以推荐最少安装两个千兆网卡，但是一般生产环境会建议部署万兆网卡。

### 硬盘

Ceph集群的性能很大程度上取决于存储介质的有效选择。应该在选择存储介质之前了解集群的工作负载和性能需求。

### Ceph OSD日志盘

如果工作环境是通用场景的需求，那么建议使用SSD做日志盘。使用SSD，可以减少访问时间，降低写延迟，大幅提升吞吐量。使用SSD做日志盘，可以对每个物理SSD创建多个逻辑分区，每个SSD逻辑分区（日志）映射到一个OSD数据盘。通常10～20GB日志大小足以满足大多数场景。

## 性能调优

### 硬件优化

#### Processor

ceph-osd进程在运行过程中会消耗CPU资源，所以一般会为每一个ceph-osd进程绑定一个CPU核上。

ceph-mon进程并不十分消耗CPU资源，所以不必为ceph-mon进程预留过多的CPU资源。

ceph-mds也是非常消耗CPU资源的，所以需要提供更多的CPU资源。

#### 内存

ceph-mon和ceph-mds需要2G内存，每个ceph-osd进程需要1G内存。

#### 网络

万兆网络现在基本上是跑Ceph必备的，网络规划上，也尽量考虑分离cilent和cluster网络。网络接口上可以使用bond来提供高可用或负载均衡。

#### SSD

SSD在ceph中的使用可以有几种架构

a、ssd作为Journal

b、ssd作为高速ssd pool(需要更改crushmap)

c、ssd做为tier pool

#### BIOS设置

a、 开启VT和HT，VH是虚拟化云平台必备的，HT是开启超线程单个处理器都能使用线程级并行计算。

b、关闭节能设置，可有一定的性能提升。

c、NUMA思路就是将内存和CPU分割为多个区域，每个区域叫做NODE,然后将NODE高速互联。 node内cpu与内存访问速度快于访问其他node的内存， NUMA可能会在某些情况下影响ceph-osd 。解决的方案，一种是通过BIOS关闭NUMA，另外一种就是通过cgroup将ceph-osd进程与某一个CPU Core以及同一NODE下的内存进行绑定。但是第二种看起来更麻烦，所以一般部署的时候可以在系统层面关闭NUMA。CentOS系统下，通过修改/etc/grub.conf文件，添加numa=off来关闭NUMA。

### 软件层面

1、 Kernel pid max 设置内核PID上限到最大值；

echo 4194303 > /proc/sys/kernel/pid\_max

2、 设置MTU，交换机端需要支持该功能，系统网卡设置才有效果；

配置文件追加MTU=9000

3、 read\_ahead, 通过数据预读并且记载到随机访问内存方式提高磁盘读操作；

echo "8192" > /sys/block/sda/queue/read\_ahead\_kb

4、 swappiness, 主要控制系统对swap的使用 ；

echo "vm.swappiness = 0"/etc/sysctl.conf ;  sysctl –p

5、 I/O Scheduler，SSD要用noop，SATA/SAS使用deadline；

echo "deadline" >/sys/block/sd[x]/queue/scheduler

echo "noop" >/sys/block/sd[x]/queue/scheduler

### Ceph层面优化

#### 配置文件的修改

[global]#全局设置

fsid = 88caa60a-e6d1-4590-a2b5-bd4e703e46d9   #集群标识ID

mon host = 10.0.1.21,10.0.1.22,10.0.1.23     #monitor IP 地址

auth cluster required = cephx         #集群认证

auth service required = cephx          #服务认证

auth client required = cephx          #客户端认证

osd pool default size = 2           #最小副本数

osd pool default min size = 1    #PG 处于 degraded 状态不影响其 IO 能力,min\_size是一个PG能接受IO的最小副本数

osd pool default pg num = 128         #pool的pg数量

osd pool default pgp num = 128        #pool的pgp数量

public network = 10.0.1.0/24                #公共网络(monitorIP段)

cluster network = 10.0.1.0/24               #集群网络

max open files = 131072      #默认0#如果设置了该选项，Ceph会设置系统的max open fds

mon initial members = controller1, controller2, compute01 #初始monitor (由创建monitor命令而定)

[mon]

mon data = /var/lib/ceph/mon/ceph-$id

mon clock drift allowed = 1    #默认值0.05#monitor间的clock drift

mon osd min down reporters = 13#默认值1#向monitor报告down的最小OSD数

mon osd down out interval = 600      #默认值300      #标记一个OSD状态为down和out之前ceph等待的秒数

[osd]

osd data = /var/lib/ceph/osd/ceph-$id

osd journal size = 20000 #默认5120           #osd journal大小

osd journal = /var/lib/ceph/osd/$cluster-$id/journal  #osd journal 位置

osd mkfs type = xfs             #格式化系统类型

osd mkfs options xfs = -f -i size=2048   #强制格式化

filestore xattr use omap = true     #默认false#为XATTRS使用object map，EXT4文件系统时使用，XFS或者btrfs也可以使用

filestore min sync interval = 10   #默认0.1#从日志到数据盘最小同步间隔(seconds)

filestore max sync interval = 15   #默认5#从日志到数据盘最大同步间隔(seconds)

filestore queue max ops = 25000    #默认500#数据盘最大接受的操作数

filestore queue max bytes = 1048576000       #默认100   #数据盘一次操作最大字节数(bytes

filestore queue committing max ops = 50000  #默认500     #数据盘能够commit的操作数

filestore queue committing max bytes = 10485760000  #默认100 #数据盘能够commit的最大字节数(bytes)

filestore split multiple = 8  #默认值2  #前一个子目录分裂成子目录中的文件的最大数量

filestore merge threshold = 40  #默认值10    #前一个子类目录中的文件合并到父类的最小数量

filestore fd cache size = 1024  #默认值128  #对象文件句柄缓存大小

journal max write bytes = 1073714824  #默认值1048560 #journal一次性写入的最大字节数(bytes)

journal max write entries = 10000  #默认值100  #journal一次性写入的最大记录数

journal queue max ops = 50000   #默认值50   #journal一次性最大在队列中的操作数

journal queue max bytes = 10485760000  #默认值33554432   #journal一次性最大在队列中的字节数(bytes)

osd max write size = 512  #默认值90   #OSD一次可写入的最大值(MB)

osd client message size cap = 2147483648  #默认值100    #客户端允许在内存中的最大数据(bytes)

osd deep scrub stride = 131072  #默认值524288    #在Deep Scrub时候允许读取的字节数(bytes)

osd op threads = 16  #默认值2     #并发文件系统操作数

osd disk threads = 4  #默认值1   #OSD密集型操作例如恢复和Scrubbing时的线程

osd map cache size = 1024  #默认值500   #保留OSD Map的缓存(MB)

osd map cache bl size = 128  #默认值50    #OSD进程在内存中的OSD Map缓存(MB)

osd mount options xfs = "rw,noexec,nodev,noatime,nodiratime,nobarrier" #默认值rw,noatime,inode64  #Ceph OSD xfs Mount选项

osd recovery op priority = 2  #默认值10       #恢复操作优先级，取值1-63，值越高占用资源越高

osd recovery max active = 10  #默认值15  #同一时间内活跃的恢复请求数

osd max backfills = 4   #默认值10   #一个OSD允许的最大backfills数

osd min pg log entries = 30000  #默认值3000  #修建PGLog是保留的最大PGLog数

osd max pg log entries = 100000  #默认值10000    #修建PGLog是保留的最大PGLog数

osd mon heartbeat interval = 40  #默认值30   #OSD ping一个monitor的时间间隔（默认30s）

ms dispatch throttle bytes = 1048576000  #默认值 104857600 #等待派遣的最大消息数

objecter inflight ops = 819200  #默认值1024      #客户端流控，允许的最大未发送io请求数，超过阀值会堵塞应用io，为0表示不受限

osd op log threshold = 50  #默认值5    #一次显示多少操作的log

osd crush chooseleaf type = 0  #默认值为1   #CRUSH规则用到chooseleaf时的bucket的类型

[client]

rbd cache = true  #默认值 true    #RBD缓存

rbd cache size = 335544320  #默认值33554432  #RBD缓存大小(bytes)

rbd cache max dirty = 134217728  #默认值25165824      #缓存为write-back时允许的最大dirty字节数(bytes)，如果为0，使用write-through

rbd cache max dirty age = 30  #默认值1      #在被刷新到存储盘前dirty数据存在缓存的时间(seconds)

rbd cache writethrough until flush = false  #默认值true  #该选项是为了兼容linux-2.6.32之前的virtio驱动，避免因为不发送flush请求，数据不回写 #设置该参数后，librbd会以writethrough的方式执行io，直到收到第一个flush请求，才切换为writeback方式。

rbd cache max dirty object = 2  #默认值0      #最大的Object对象数，默认为0，表示通过rbd cache size计算得到，librbd默认以4MB为单位对磁盘Image进行逻辑切分   #每个chunk对象抽象为一个Object；librbd中以Object为单位来管理缓存，增大该值可以提升性能

rbd cache target dirty = 235544320  #默认值16777216   #开始执行回写过程的脏数据大小，不能超过 rbd\_cache\_max\_dirty

#### PG Number

PG和PGP数量一定要根据OSD的数量进行调整，计算公式如下，但是最后算出的结果一定要接近或者等于一个2的指数。

Total PGs = (Total\_number\_of\_OSD \* 100) / max\_replication\_count

例如：

有100个osd，2副本，5个pool

Total PGs =100\*100/2=5000

每个pool 的PG=5000/5=1000，那么创建pool的时候就指定pg为1024

ceph osd pool create pool\_name 1024

#### 修改crush map

Crush map可以设置不同的osd对应到不同的pool，也可以修改每个osd的weight。

配置可参考：http://linuxnote.blog.51cto.com/9876511/1790758

#### 其他因素

ceph osd perf

通过osd perf可以提供磁盘latency的状况，如果延时过长，应该剔除osd。