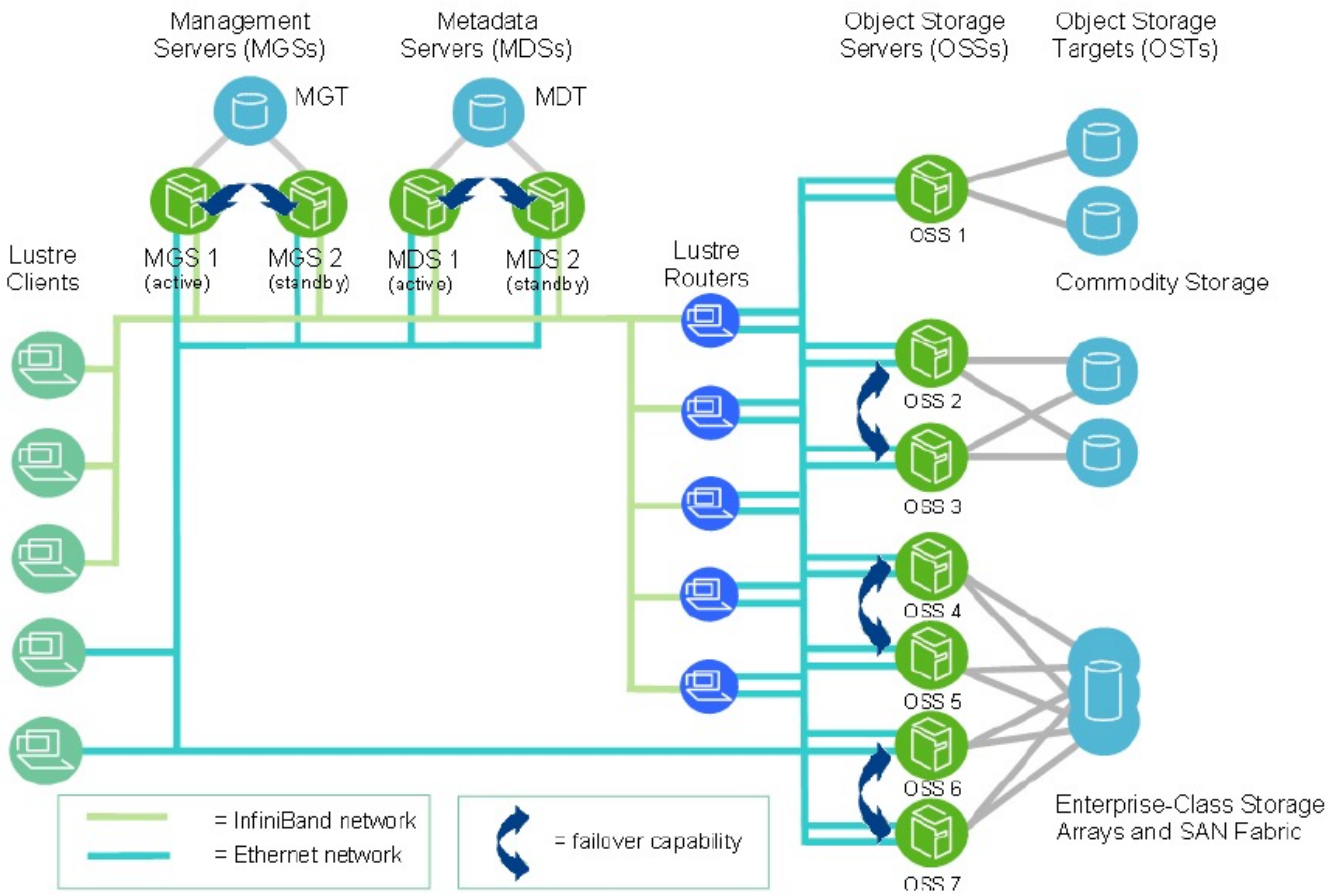
超算中的存储系统

龙汀汀

概述



Lustre分布式文件系统集群的架构

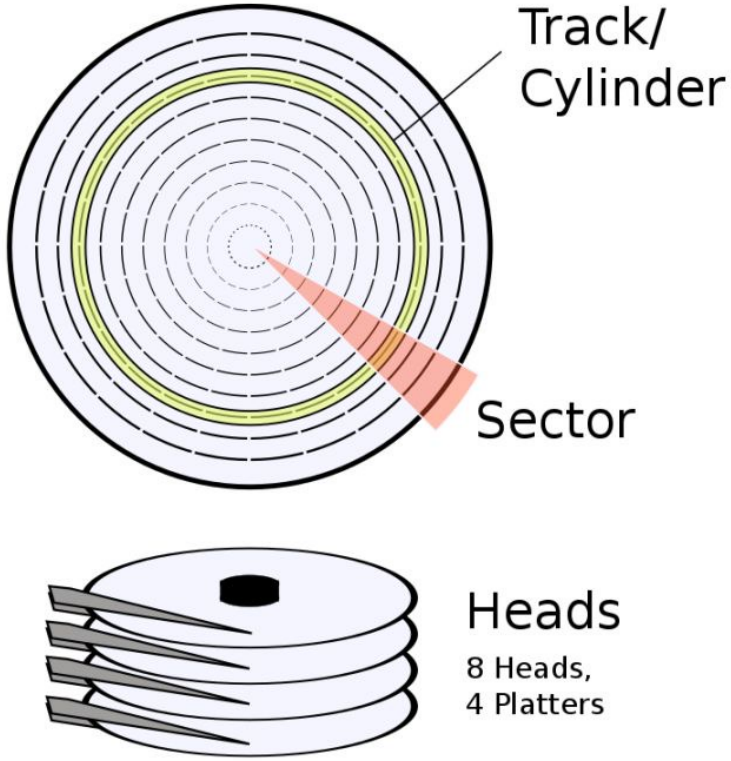
Lustre Client
网络
Lustre Server
后端文件系统(ZFS)
硬件(HBA卡、磁盘等)

Lustre的软件栈

存储性能的评价维度

- 时延：一次IO操作的耗时
- 随机IO吞吐：多个并行的随机IO所能达到的读写速率
- 顺序IO吞吐：顺序IO所能达到的读写速率

机械硬盘的构造

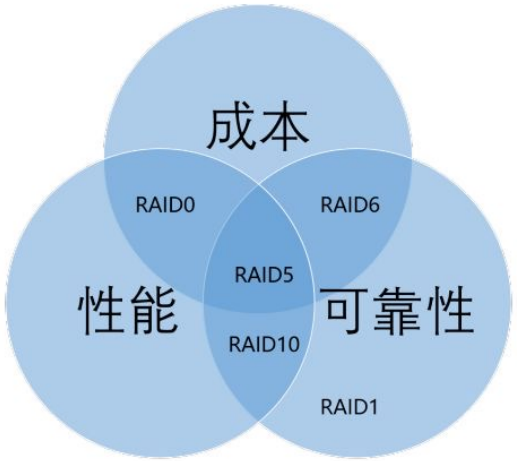


机械硬盘一次IO的时间包括：

- 寻道时间(磁头移动到对应磁道)，一般是10ms级别
- 旋转时间(磁片旋转到对应位置)，取决于磁盘转速，7200rpm的平均时间(转半圈)是4.2ms
- 数据传输时间(读写和传输磁道上的数据)

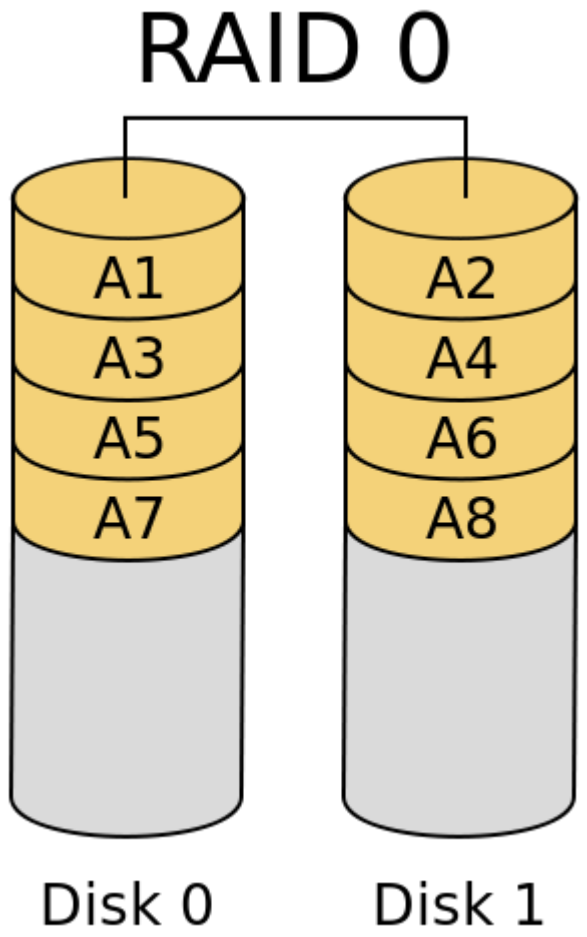
RAID

- Redundant Array of Independent/Inexpensive Disks
- 独立/廉价冗余磁盘阵列
- 用一定的方式把多个相对便宜的硬盘组合起来，在成本、性能和可靠性之间取得平衡
- 是现代存储集群中管理磁盘的基础技术



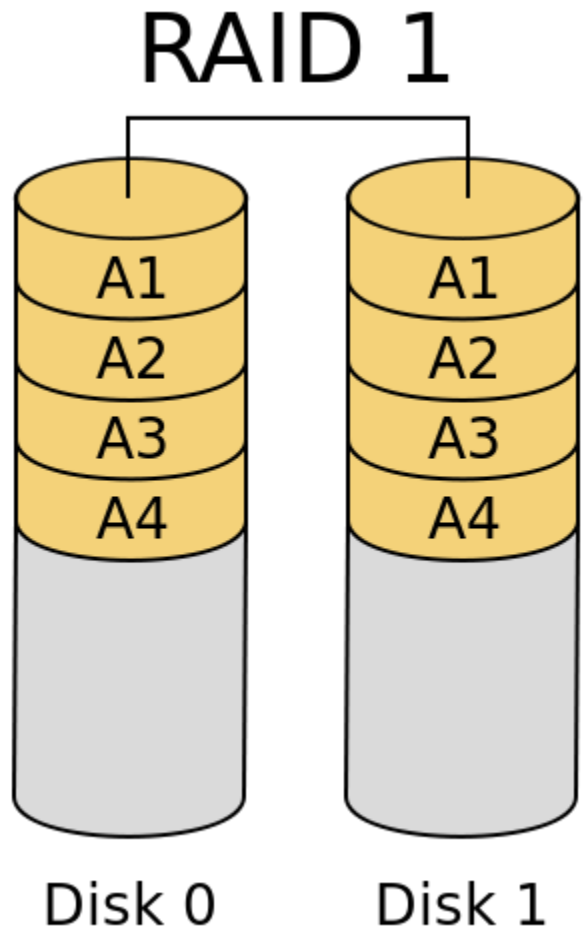
RAID0

- 数据分散存储在所有磁盘中
- N倍IO吞吐
- 可靠性差，无法容忍磁盘故障
- 条带化(一行表示一个条带)



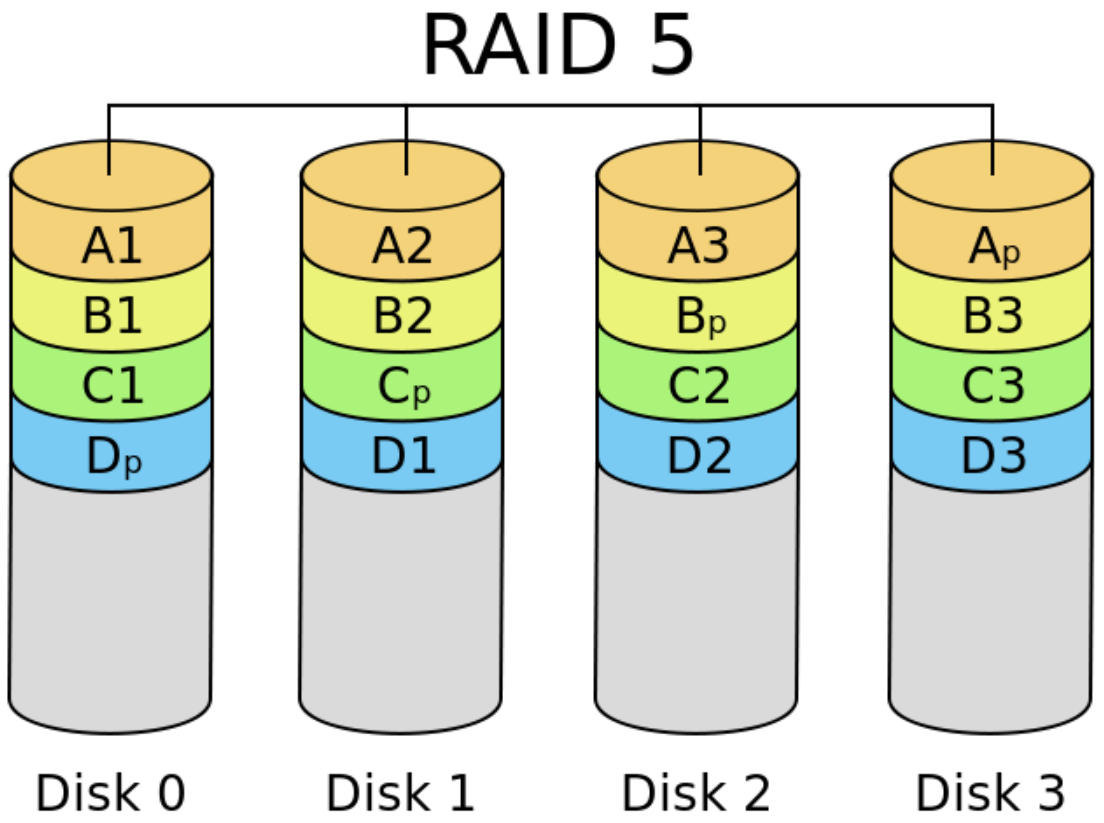
RAID1

- 2块或更多磁盘，每个磁盘保存一样的数据
- 可靠性高，可容忍N-1个磁盘同时故障
- N倍读取吞吐
- 写入速率等于单个磁盘，或者更慢



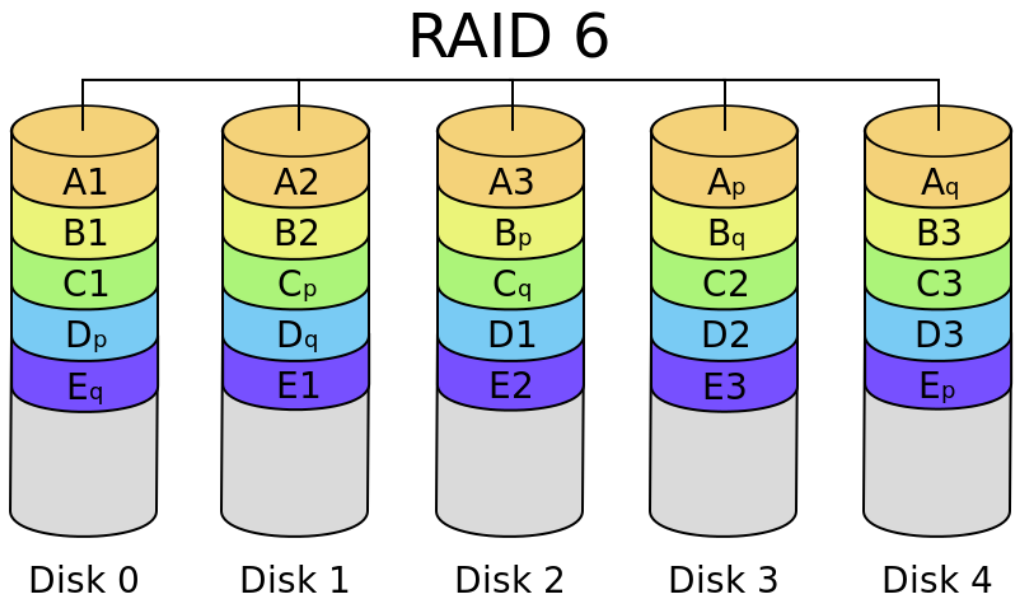
RAID5

- 3块或更多磁盘，1个条带中有1个校验块
- 可容忍1个磁盘故障
- 读写性能均有接近N倍提升

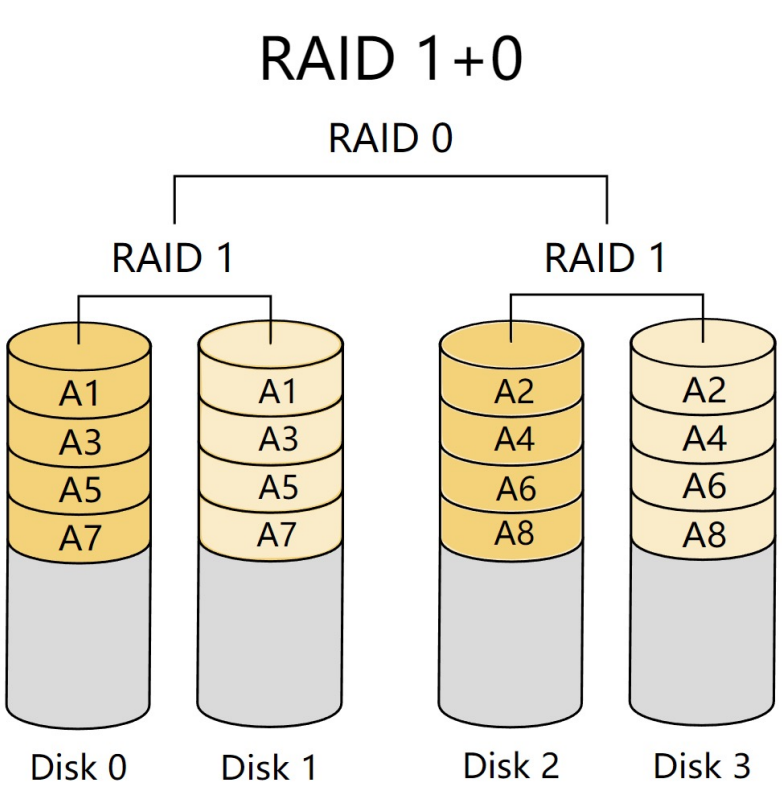


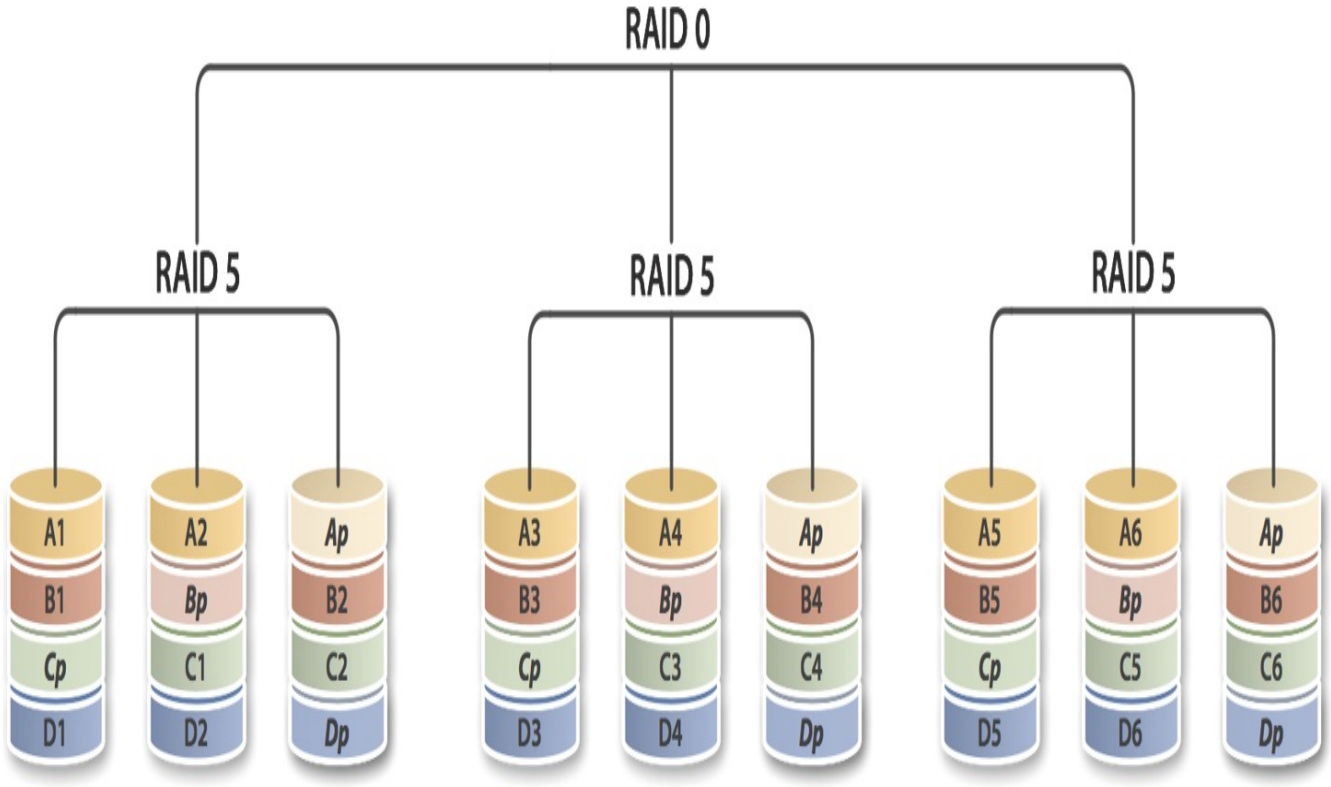
RAID6

- 4块或更多磁盘，1个条带中有2个校验块
- 可容忍2个磁盘故障
- 读写性能均有所提高



嵌套raid级别


RAID 1+0
RAID 0
RAID 1
RAID 1
A1
A3
A5
A7
A1
A3
A5
A7
A2
A4
A6
A8
A2
A4
A6
A8
Disk 0
Disk 1
Disk 2
Disk 3


RAID 0
RAID 5
RAID 5
RAID 5
A1
B1
Cp
D1
A2
Bp
C1
D2
Ap
B2
C2
Dp
A3
B3
Cp
D3
A4
Bp
C3
D4
Ap
B4
C4
Dp
A5
B5
Cp
D5
A6
Bp
C5
D6
Ap
B6
C6
Dp

Raid的性能

- 大体上说，读性能：raid0>raid1>raid5>raid6，写性能：raid0>raid5>raid6>raid1
- 嵌套raid的性能特征更复杂，取决于磁盘数量和raid组合方式。
- 嵌套raid中，通常来说简单地增加磁盘数量就能提高性能，无论是增加group内磁盘数量或是增加group数量。但磁盘数增加到一定规模后性能提升缓慢甚至无提升。

OpenZFS文件系统

- OpenZFS是一个开源的单机文件系统，可以作为Lustre分布式文件系统的后端
- 提供了软RAID(包括raid0，raid1，raid5/6/7，以及嵌套raid)、压缩、加密、快照、去重等功能。

OpenZFS的基本概念

- vdev，virtual device。vdev即磁盘，或者能当磁盘用的东西。包括：磁盘、文件、raidz、mirror（即raid1）、缓存/日志设备等。raidz/mirror类型的vdev不能嵌套，即不能对raidz/mirror进一步做raidz/mirror
- pool：对vdev做条带化后得到的存储池
- dataset：类似于分区。创建了一个pool后，默认会创建一个dataset。可以后续手动添加dataset。与一般的分区的差别在于，zfs不预设每个dataset的容量，dataset之间共享整个pool。建在同一个pool上的不同dataset可以设置不同的优化参数，来应对不同的场景。

https://openzfs.github.io/openzfs-docs/man/7/zpoolconcepts.7.html

OpenZFS中的raid

- OpenZFS提供了基本的raid0和raid1，以及raid5/6/7的改进版本，即raidz1/2/3。Raidz比传统raid更加复杂。
- Raidz的改进包括：
  - 可变条带长度，减少碎片，代价是降低了性能
  - 有额外的校验数据，可以纠错（纠静默数据错误），代价是额外的空间和性能开销；而传统raid的校验和只能在磁盘损坏时恢复数据，无法纠错
  - 通过写时复制来解决write hole问题，并提供快照功能

OpenZFS-可变条带长度

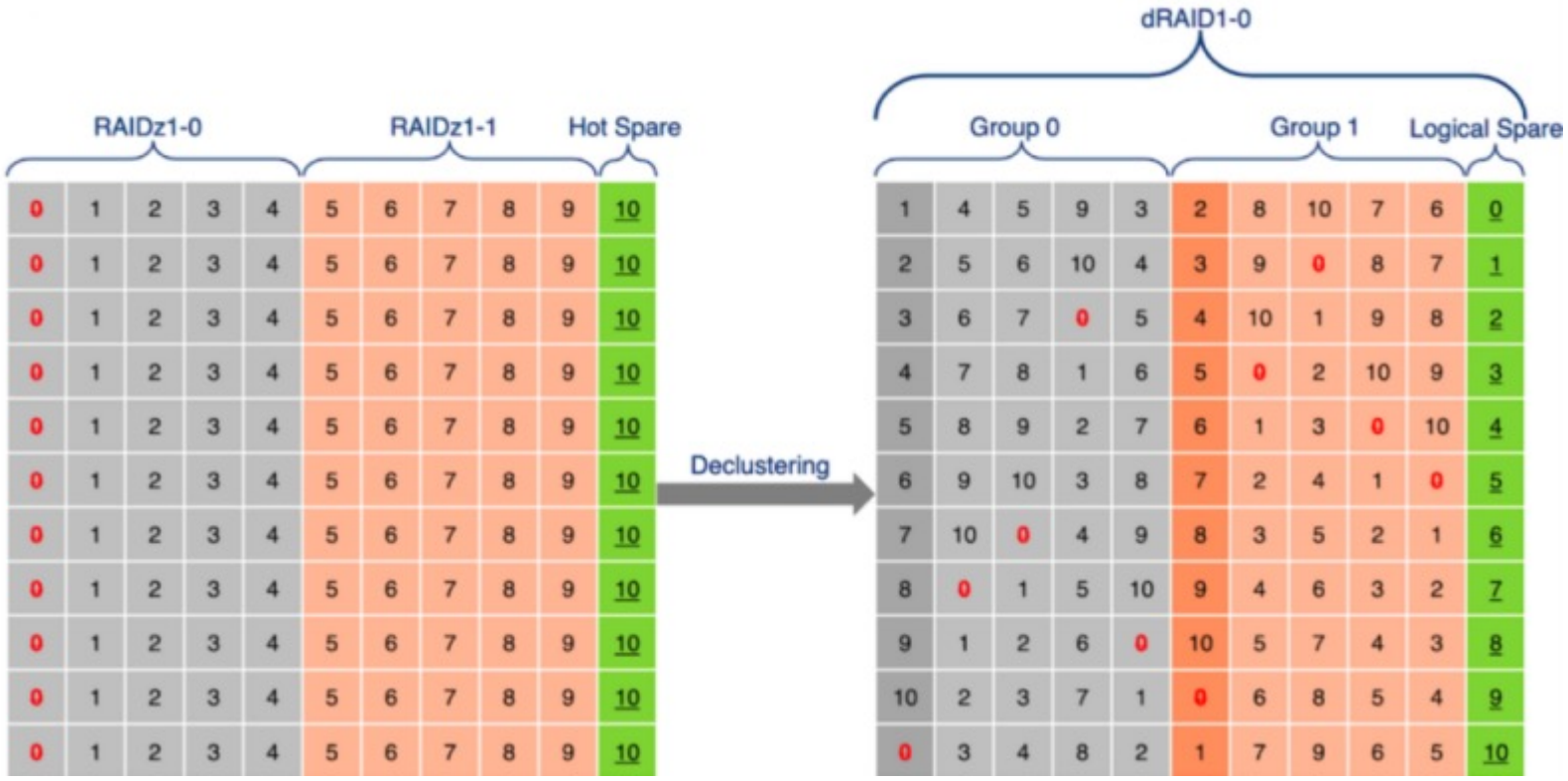
RAID5			
Disk 1	Disk 2	Disk 3	Disk 4
1	2	3	P
5	6	P	4
9	P	7	8
P	10	11	12

RAIDZ			
Disk 1	Disk 2	Disk 3	Disk 4
P	1	P	2
P	3	4	X
P	5	6	7
8	P	9	10

- 使用额外的元数据记录每个数据块的长度和位置；
- 空间利用率更高，减少碎片；但也需要空间存放元数据；
- 降低了性能，因为IO操作时需要读写元数据；
- 通过元数据追踪已用/空闲的磁盘块，重建时只需考虑占用的块；当已用空间不多时，可以大幅提高重建速度；但如果磁盘利用率较高，则适得其反

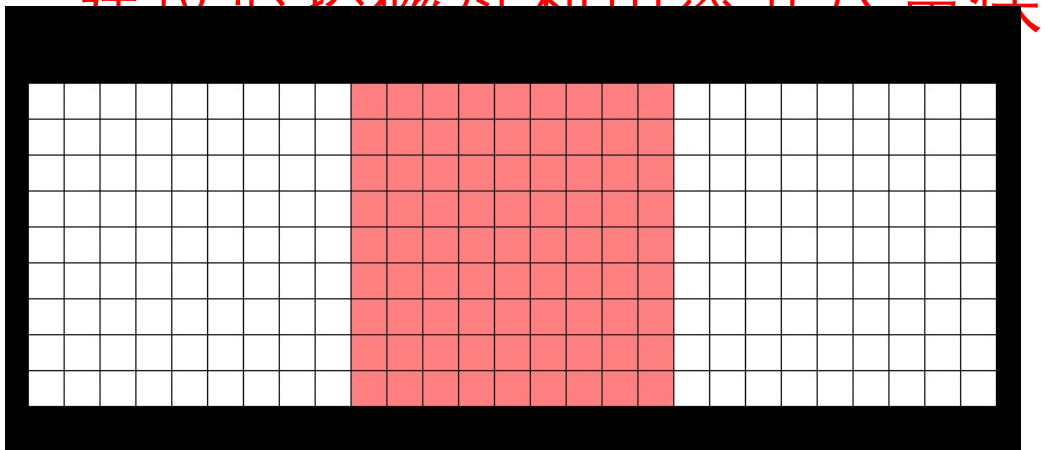
OpenZFS-draid

- Draid是对raidz的改进，以性能为代价提高了重建速度

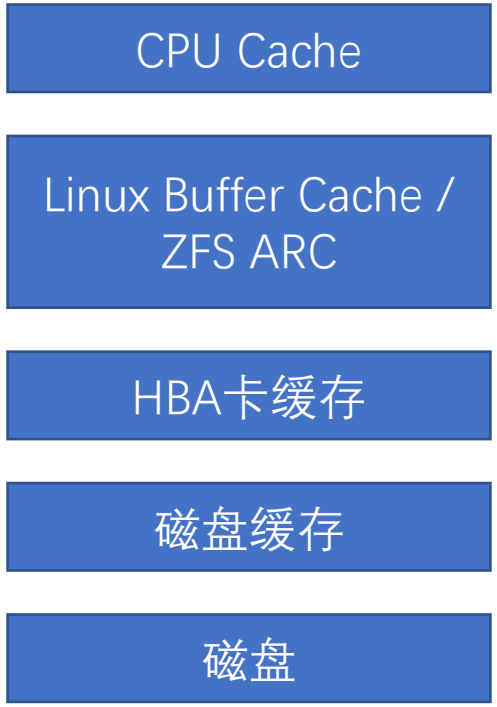


OpenZFS-写时复制

- ZFS采用了写时复制(copy on write)技术，每次写入数据块时不是覆盖原数据，而是寻找一个空闲块写入新数据
- 可以快速创建快照，并定期备份
- 解决write hole的问题（raid5/6/7阵列中，写入条带时断电，导致数据和校验块不一致）
- 代价是容易导致磁盘碎片，在磁盘利用率较高时会显著降低性能，建议监控磁盘利用率并尽量保持在80%以下



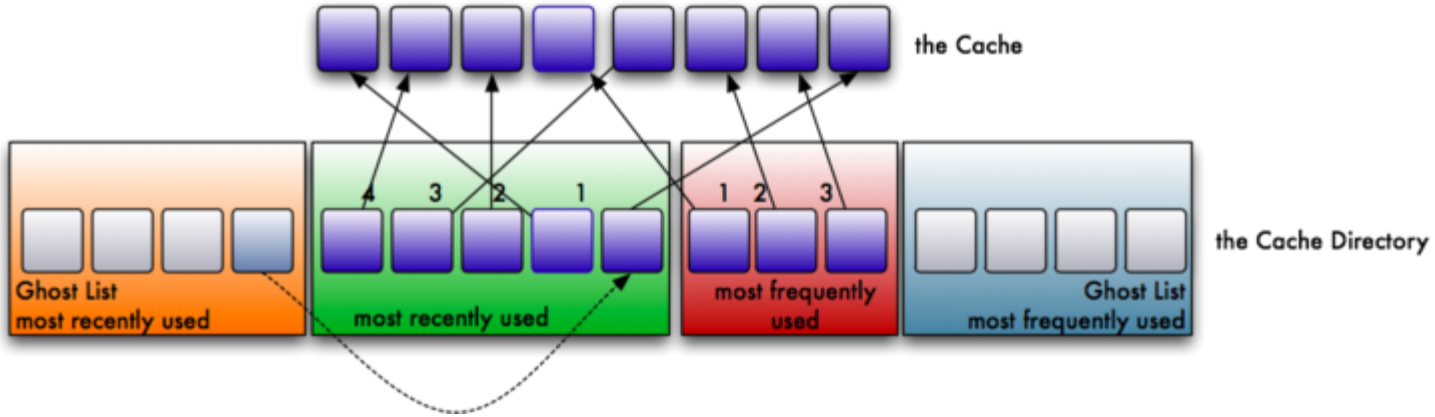
OpenZFS中的存储层次结构



- 在存储系统中，影响性能的不仅仅是磁盘和RAID，作用更大的实际上是各级缓存
- 存储系统的整体性能取决于各级缓存的性能和命中率
- 主要关注内存这一级别。在该级别，ext4等文件系统使用Linux Buffer Cache作为缓存，但ZFS使用了效率更高的ZFS ARC缓存

ZFS ARC

- ZFS ARC是zfs文件系统所使用的读缓存(不适用于写入操作)
- Adjustable Replacement Cache
- 结合了LRU(Least Recently Used)和LFU(Least Frequently Used)两种cache替换算法，根据工作负载动态调整两类cache的比例，从而比Linux Buffer Cache的单纯的LRU更好。
- ARC默认最高占系统内存的1/2，可通过参数调整，Lustre建议3/4



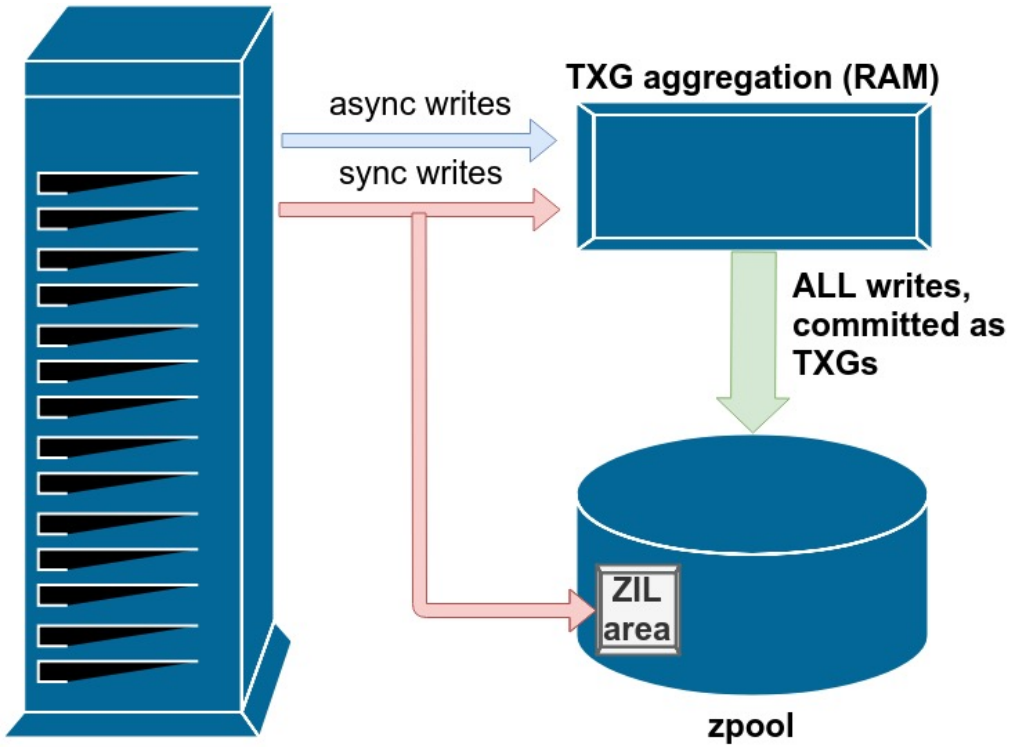
L2 ARC

- 可以使用高速硬盘(例如高速ssd等)来扩展ARC，作为L2 ARC

OpenZFS-写缓冲与写日志

- 写入分为同步写入和异步写入，通过在open函数中通过flags参数来指定
  - 同步写入：程序调用write函数写入数据，数据落盘后write函数返回，程序继续运行
  - 异步写入：程序调用write函数写入数据，文件系统将数据写入内存缓冲后立即返回，程序继续运行；文件系统在适当时机将内存缓冲区落盘
  - 异步写入可以利用内存缓冲，性能更好，但存在断电丢失数据的风险

OpenZFS-写缓冲与写日志

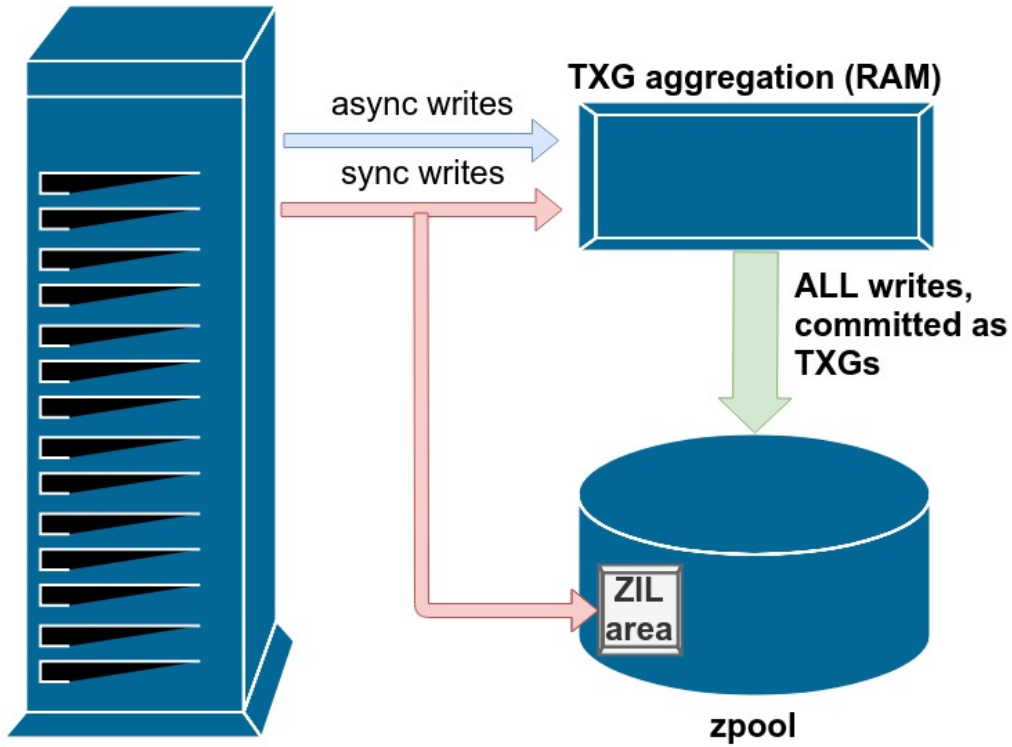


Normal operation - without SLOG

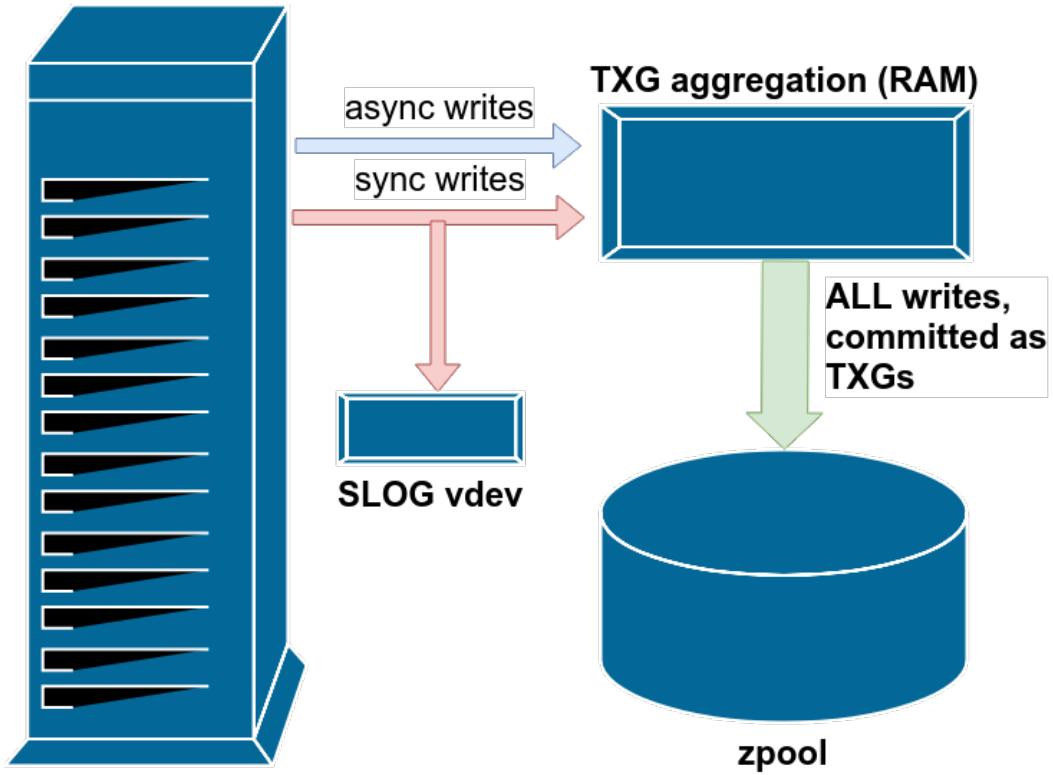
https://jrs-s.net/2019/05/02/zfs-sync-async-zil-slog/

- 这张图说明了zfs里的同步写入和异步写入的区别
- 异步写入(async writes)中，数据被写入TXG(位于内存)中，并在稍后刷入磁盘
- 同步写入(sync writes)中，数据被写入TXG(位于内存)和ZIL(位于磁盘)中；zfs稍后将TXG刷入磁盘并删除ZIL中的相应数据；这会导致一次同步写入被放大为两次磁盘写IO(一次zil，一次正式写入)；对于较大的数据，zfs在写入zil时仅写入指针，同时将数据直接写入存储池，从而缓解该问题

OpenZFS-写缓冲与写日志



Normal operation - without SLOG



Normal operation - with SLOG vdev

可以配置一个SLOG设备来存储ZIL，从而提高同步写入的性能，同时避免双写带来的写入放大问题

OpenZFS中的缓存/缓冲/日志概念总结

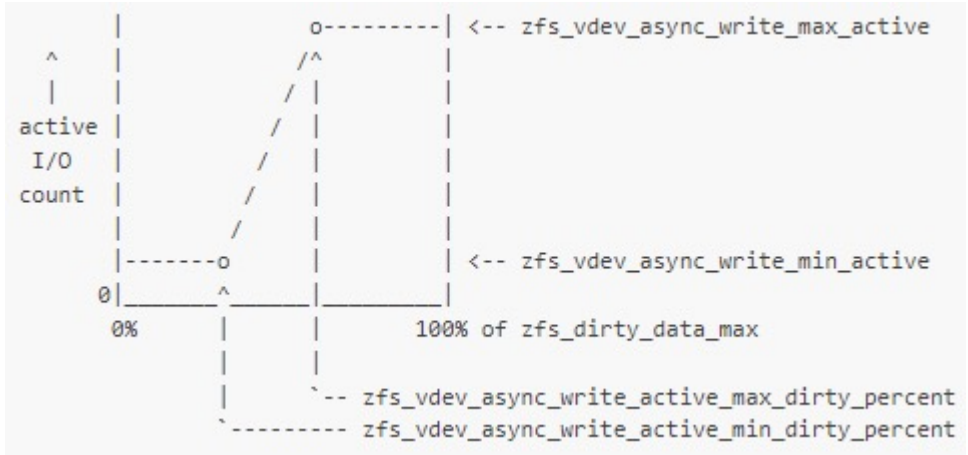
- ARC-读缓存，位于内存，用于同步读取（读取都是同步的）
- TXG-写缓冲，位于内存，用于同步/异步写入
- ZIL-同步写日志，位于磁盘，用于同步写入；如果没有配置SLOG的话就和普通磁盘数据存在一起
- SLOG-用于存储ZIL的设备，通常使用高速SSD等

ZFS IO调度器

- IO调度器用于确定IO操作发出的时间和顺序。在并发场景中，通过调整同步/异步读写，以及纠错等任务的线程数来获得最佳的吞吐和延迟
- 调高某类IO的max_active将提高此类操作的最大线程数，通常能提高性能（达到性能上限后不再提高)，但会导致其他操作的延迟增加。例如，调高scrub read 的max_active可以加快重建和纠错速度，但是会影响其他作业。

I/O Class	Min Active Parameter	Max Active Parameter
sync read	zfs_vdev_sync_read_min_active	zfs_vdev_sync_read_max_active
sync write	zfs_vdev_sync_write_min_active	zfs_vdev_sync_write_max_active
async read	zfs_vdev_async_read_min_active	zfs_vdev_async_read_max_active
async write	zfs_vdev_async_write_min_active	zfs_vdev_async_write_max_active
scrub read	zfs_vdev_scrub_min_active	zfs_vdev_scrub_max_active

ZFS IO调度器-异步写



- 异步写的调度更复杂。ZFS提供了5个参数来控制异步写的调度，用于控制脏数据的最大容量(即写缓冲的大小)，以及脏数据量与并行IO操作数的分段函数。
- 调整这些参数可以权衡写缓冲的容量、异步写入中落盘的时机等等

ZFS中的其他模块参数

- min_active、max_active等参数都是模块参数，即全局的参数（区别于某个虚拟设备、某个分区的参数）
- zfetch_max_distance-预取长度
- zfs_arc_max-arc读缓存容量
- zfs_vdev_aggregation_limit-最终向设备发出IO操作前，允许聚合的最大数据量；调高后能将更多相邻小IO合并成为一个大的IO，降低IOPS，代价是更高的CPU性能开销
- 。。。。。。
- 官方文档：https://openzfs.github.io/openzfs-docs/Performance%20and%20Tuning/Module%20Parameters.html

ZFS中的pool和dataset参数

- ZFS可以为不同的pool和dataset设置参数
- ashift-扇区大小，应当大于等于底层磁盘的扇区大小
- compression-压缩算法，默认不压缩，建议开启并使用lz4压缩算法
- dedup-是否开启重复数据删除，默认不开启。开启后对内存的需求很大，实践经验表明约需要磁盘容量的5‰的内存，内存溢出后性能将会急剧下降。
- 。。。
- https://openzfs.github.io/openzfs-docs/man/7/zpoolprops.7.html
- https://openzfs.github.io/openzfs-docs/man/7/zfsprops.7.html

ZFS的性能分析

- 使用zpool iostat命令，可以显示各类操作的延迟、排队情况等，协助定位性能的瓶颈
- https://openzfs.github.io/openzfs-docs/man/8/zpool-iostat.8.html
- 使用BPF等工具来进行更深入的追踪和分析
- https://www.brendangregg.com/linuxperf.html

ZFS的维护

- 关注ZFS的磁盘空间使用率，尽量保持在80%以下甚至更低，否则会由于碎片化导致性能显著下降，以及重建速度变慢
- 为pool设置热备盘并开启autoreplace，在磁盘故障时自动进行热备替换并邮件提醒，缩短raid降级的窗口期
- 定期使用scrub命令来纠磁盘静默错误
- 定期使用快照和备份
- 定期使用zpool status监控pool的健康状态并邮件提醒

存储系统的Benchmark工具

- VDBench-裸盘性能
- IOzone-文件系统性能，衡量简单应用场景下的性能
- IO500-结合Mdtest、IOR等测试工具，是HPC存储系统基准测试的权威