# 关于io优化

io优化的概念

读/写IO，最为常见说法，读IO，就是发指令，从磁盘读取某段扇区的内容。指令一般是通知磁盘开始扇区位置，然后给出需要从这个初始扇区往后读取的连续扇区个数，同时给出动作是读，还是写。磁盘收到这条指令，就会按照指令的要求，读或者写数据。控制器发出的这种指令＋数据，就是一次IO，读或者写。

大/小块IO，指控制器的指令中给出的连续读取扇区数目的多少，如果数目很大，比如128，64等等，就应该算是大块IO，如果很小，比如1， 4，8等等，就应该算是小块IO，大块和小块之间，没有明确的界限。

连续/随机IO，连续和随机，是指本次IO给出的初始扇区地址，和上一次IO的结束扇区地址，是不是完全连续的，或者相隔不多的，如果是，则本次IO应该算是一个连续IO，如果相差太大，则算一次随机IO。连续IO，因为本次初始扇区和上次结束扇区相隔很近，则磁头几乎不用换道或换道时间极短；如果相差太大，则磁头需要很长的换道时间，如果随机IO很多，导致磁头不停换道，效率大大降底。   
  
顺序/并发IO，这个的意思是，磁盘控制器每一次对磁盘组发出的指令套（指完成一个事物所需要的指令或者数据），是一条还是多条。如果是一条，则控制器缓存中的IO队列，只能一个一个的来，此时是顺序IO；如果控制器可以同时对磁盘组中的多块磁盘，同时发出指令套，则每次就可以执行多个IO，此时就是并发IO模式。并发IO模式提高了效率和速度。

IO并发几率。单盘，IO并发几率为0，因为一块磁盘同时只可以进行一次IO。对于raid0，2块盘情况下，条带深度比较大的时候（条带太小不能并发IO，下面会讲到），并发2个IO的几率为1/2。其他情况请自行运算。   
  
IOPS。一个IO所用的时间＝寻道时间＋数据传输时间。 IOPS＝IO并发系数/（寻道时间＋数据传输时间），由于寻道时间相对传输时间，大几个数量级，所以影响IOPS的关键因素，就是降底寻道时间，而在连续IO的情况下，寻道时间很短，仅在换磁道时候需要寻道。在这个前提下，传输时间越少，IOPS就越高。

每秒IO吞吐量。显然，每秒IO吞吐量＝IOPS乘以平均IO SIZE。 Io size越大，IOPS越高，每秒IO吞吐量就越高。设磁头每秒读写数据速度为V，V为定值。则IOPS＝IO并发系数/（寻道时间＋IO SIZE/V），代入，得每秒IO吞吐量＝IO并发系数乘IO SIZE乘V/（V乘寻道时间＋IO SIZE）。我们可以看出影响每秒IO吞吐量的最大因素，就是IO SIZE和寻道时间，IO SIZE越大，寻道时间越小，吞吐量越高。相比能显著影响IOPS的因素，只有一个，就是寻道时间。

那么我们该如何优化io呢

我们可以通过一下几点来优化io提升磁盘性能

实现数据的串联、条带化、镜像

串联（Concatenation）: 按顺序使用磁盘，一个磁盘使用完以后使用后续的磁盘

条带化（Striping）: 交替使用不同磁盘的空间。条带化使得IO操作可以并行，因此是提高IO性能的关键

镜像 （mirror）：当一个磁盘进行IO操作时，相同的操作发生在另外一个磁盘上。这样的功能为数据的安全性提供了支持.

实现以上功能

1. 我们可以为磁盘做RAID，Raid很基础，但是在存储系统中占据非常重要的地位，所有涉及存储的书籍都会提到RAID。RAID通过磁盘冗余的方式提高了可用性和可高性，一方面增加了数据读写速度，另一方面增加了数据的安全性。
2. 我们可以使用开源的存储工具来实现，比如gluesfs，ceph就可以很好的实现对象存储，块存储，文件系统，实现数据的镜像，条带化，串联
3. DAS，NAS，SAN

1．单台主机。在这种情况下，存储作为主机的一个或多个磁盘存在，这样局限性也是很明显的。由于受限于主机空间，一个主机只能装一块到几块硬盘，而硬盘空间时受限的，当磁盘满了以后，你不得不为主机更换更大空间的硬盘。

2 . 独立存储空间。为了解决空间的问题，于是考虑把磁盘独立出来，于是有了DAS（Direct Attached Storage），即：直连存储。DAS就是一组磁盘的集合体，数据读取和写入等也都是由主机来控制。但是，随之而来，DAS又面临了一个他无法解决的问题——存储空间的共享。接某个主机的JBOD（Just a Bunch Of Disks，磁盘组），只能这个主机使用，其他主机无法用。因此，如果DAS解决空间了，那么他无法解决的就是如果让空间能够在多个机器共享。因为DAS可以理解为与磁盘交互，DAS处理问题的层面相对更低。使用协议都是跟磁盘交互的协议

3 . 独立的存储网络。为了解决共享的问题，借鉴以太网的思想，于是有了SAN（Storage Area Network），即：存储网络。对于SAN网络，你能看到两个非常特点，一个就是光纤网络，另一个是光纤交换机。SAN网络由于不会之间跟磁盘交互，他考虑的更多是数据存取的问题，因此使用的协议相对DAS层面更高一些

当然，以上工具我们都学过

关于内核io优化

1.  page cache

linux操作系统默认情况下写都是有写缓存的，可以使用direct IO方式绕过操作系统的写缓存。当你写一串数据时，系统会开辟一块内存区域缓存这些数据，这块区域就是我们常说的page cache（操作系统的页缓存）。查看系统内存常用的命令有：vmstat、free、top等。

可以使用 cat /proc/meminfo 查看详细的内存使用情况

2. writeback

有了page cache就有了writeback写方式。一个写IO会先写入page cache，然后等待后台pdflush把page cache中脏数据刷入磁盘。如果在刷入磁盘之前系统断电，则page cache的数据丢失。所以对一些可靠性要求高的场景都会把这个写缓存禁掉。writeback写方式是linux操作系统提供的一种非常通用写模式。writeback提供了较好的吞吐量，有了缓存也缩短了IO响应时间。但它也有缺点：（1）断电可能丢数据（数据安全性）（2）对于像数据库这样自缓存的系统来说，多了一层IO缓存开销。因为数据库已经在应用层做了一层缓存。所以对于这样的应用可以用direct io方式，减少用户空间和page cache之间数据复制开销。（3）如果page cache过大，那么就会缓存太多的数据，当需要统一刷入磁盘的时候就会出现一个IO峰值和瓶颈，在这其间对用户的IO访问出现明显影响。如果想削平这个峰值可以把page cache容量设置小一点，让pdflush一段时间内较为平均的刷新dirty数据。

3. pdflush

pdflush是linux系统后台运行的一个线程，这个进程负责把page cahce中的dirty状态的数据定期的输入磁盘。一个系统中会运行很多这个pdflush。cat /proc/sys/vm/nr\_pdflush\_threads查看当前系统运行pdflush数量。当一段时间（一般是1s）没有任何的pdflush处于工作状态，系统会remove一个pdflush线程。pdflush最大和最小的数量是有配置的，但这些配置一般很少修改。