FC SAN

Enterprise applications

NetApp
May 09, 2024

目录

FC SAN

Oracle数据库I/O的LUN对齐

LUN对齐是指针对底层文件系统布局优化I/O。

在ONTAP系统上、存储以4 KB为单位进行组织。一个数据库或文件系统的8 KB块应正好映射到两个4 KB块。如果LUN配置错误使对齐在任一方向上移动1 KB、则每个8 KB块将位于三个不同的4 KB存储块上、而不是两个。这种安排会增加发生原因延迟、并在存储系统中执行发生原因额外的I/O。

对齐也会影响LVM架构。如果在整个驱动器设备上定义了逻辑卷组中的物理卷(不创建分区)、则LUN上的第一个4 KB块与存储系统上的第一个4 KB块对齐。这是正确的对齐方式。分区会出现问题、因为它们会移动操作系统使用LUN的起始位置。只要偏移量以4 KB的整数单位移动、LUN就会对齐。

在Linux环境中、在整个驱动器设备上构建逻辑卷组。如果需要分区、请运行以检查对齐情况 `fdisk -u` 并验证每个分区的起始位置是否为八的倍数。这意味着分区从八个512字节扇区的倍数开始、即4 KB。

另请参见一节中有关数据压缩块对齐的讨论 "效率"。与8 KB压缩块边界对齐的任何布局也与4 KB边界对齐。

未对齐警告

数据库重做/事务日志记录通常会生成未对齐的I/O、此I/O可能会导致发生原因发出有关ONTAP上LUN错位的警告、从而使人产生误解。

日志记录会使用不同大小的写入顺序写入日志文件。不与4 KB边界对齐的日志写入操作通常不会出现发生原因性能问题、因为下一个日志写入操作会完成块。因此、ONTAP几乎能够将所有写入作为完整的4 KB块进行处理、即使某些4 KB块中的数据是在两个单独的操作中写入的。

使用等实用程序验证对齐情况 `sio` 或 `dd` 可以按定义的块大小生成I/O。可以使用查看存储系统上的I/O对齐统计信息 `stats` 命令：请参见 "WAFL对齐验证" 有关详细信息 …

Solaris环境中的对齐更为复杂。请参见 "ONTAP SAN 主机配置" 有关详细信息 …

小心
在Solaris x86环境中，请格外注意正确对齐，因为大多数配置都有多个分区层。Solaris x86分区片通常位于标准主引导记录分区表之上。

Oracle数据库LUN大小调整和LUN计数

要获得Oracle数据库的最佳性能和易管理性、选择最佳LUN大小和要使用的LUN数量至关重要。

LUN是ONTAP上的一个虚拟化对象、位于托管聚合中的所有驱动器上。因此、LUN的性能不受其大小的影响、因为无论选择何种大小、LUN都会利用聚合的全部性能潜能。

为了方便起见、客户可能希望使用特定大小的LUN。例如、如果数据库是基于LVM或Oracle ASM磁盘组构建的、其中每个磁盘组包含两个1 TB的LUN、则该磁盘组必须以1 TB为增量进行增长。最好使用八个500 GB的LUN来构建磁盘组、以便可以以较小的增量来增加磁盘组。

建议不要建立通用标准LUN大小、因为这样做会使易管理性复杂化。例如、如果数据库或数据存储库的大小介于1 TB到2 TB之间、则100 GB的标准LUN大小可能效果良好、但20 TB的数据库或数据存储库需要200个LUN。这意味着、服务器重新启动时间会更长、需要在各种用户界面中管理更多对象、SnapCenter等产品必须对许多对象执行发现。使用更少、更大的LUN可避免此类问题。

- LUN计数比LUN大小更重要。
- LUN大小主要由LUN计数要求控制。
- 避免创建超出所需数量的LUN。

LUN计数

与LUN大小不同、LUN计数会影响性能。应用程序性能通常取决于通过SCSI层执行并行I/O的能力。因此、两个LUN的性能优于一个LUN。使用Veritas VLVM、Linux LVM2或Oracle ASM等LVM是提高并行性的最简单方法。

虽然对随机I/O非常繁重的100% SSD环境进行的测试表明、LUN数量最多可增加到64个、但一般来说、NetApp客户从LUN数量增加到16个以上所获得的优势微乎其微。

- NetApp建议*：

通常、四到十六个LUN足以满足任何给定数据库工作负载的I/O需求。由于主机SCSI实施的限制、如果LUN数量少于四个、则可能会造成性能限制。

Oracle数据库LUN放置

数据库LUN在ONTAP卷中的最佳放置位置主要取决于各种ONTAP功能的使用方式。

Volumes

首次接触ONTAP的客户通常会感到困惑的一点是、FlexVol的使用、通常简称为"卷"。

卷不是LUN。这些术语与许多其他供应商产品(包括云提供商)同义。ONTAP卷只是管理容器。它们不会自行提供数据、也不会占用空间。它们是文件或LUN的容器、旨在提高和简化易管理性、尤其是大规模管理。

卷和LUN

相关LUN通常位于同一个卷中。例如、需要10个LUN的数据库通常会将所有10个LUN放置在同一个卷上。

- 采用1：1的LUN与卷比率(即每个卷一个LUN)是一种*不*正式的最佳实践。
- 而是应将卷视为工作负载或数据集的容器。每个卷可能有一个LUN、也可能有多个LUN。正确的问题解答取决于易管理性要求。
- 将LUN分散在不必要数量的卷上可能会导致额外开销和操作计划问题、例如快照操作、UI中显示的对象数量过多、并导致在达到LUN限制之前达到平台卷限制。

卷、LUN和快照

Snapshot策略和计划放置在卷上、而不是LUN上。如果包含10个LUN的数据集位于同一个卷中、则这些LUN只需要一个Snapshot策略。

此外、在一个卷中将给定数据集的所有相关LUN同位可实现原子快照操作。例如、如果基础LUN都位于一个卷上、则驻留在10个LUN上的数据库或包含10个不同操作系统的基于VMware的应用程序环境可以作为一个一致的对象进行保护。如果将它们放置在不同的卷上、则快照可能会(也可能不会)完全同步、即使是同时计划的也是如此。

在某些情况下、由于恢复要求、可能需要将一组相关LUN拆分为两个不同的卷。例如、一个数据库可能有四个用于数据文件的LUN和两个用于日志的LUN。在这种情况下、最好使用包含4个LUN的数据文件卷和包含2个LUN的日志卷。原因是独立可恢复性。例如、可以有选择地将数据文件卷还原到先前的状态、这意味着所有四个LUN都将还原到快照的状态、而日志卷及其关键数据不会受到影响。

卷、LUN和SnapMirror

SnapMirror策略和操作与快照操作一样、在卷上执行、而不是在LUN上执行。

通过在一个卷中将相关LUN同位、您可以创建一个SnapMirror关系、并通过一次更新来更新所有包含的数据。与快照一样、更新也是一项原子操作。保证SnapMirror目标具有源LUN的单个时间点副本。如果LUN分布在多个卷上、则这些副本之间可能一致、也可能不一致。

卷、LUN和QoS

虽然可以有选择地将QoS应用于各个LUN、但在卷级别设置QoS通常更容易。例如、给定ESX服务器中子系统使用的所有LUN都可以放置在一个卷上、然后应用ONTAP自适应QoS策略。因此、会产生一个可自行扩展的每TB IOPS限制、用于对所有LUN执行适用场景操作。

同样、如果数据库需要10万次IOPS并占用10个LUN、则在单个卷上设置一个10万次IOPS限制比在每个LUN上设置10个单独的10万次IOPS限制更容易。

多卷布局

在某些情况下、在多个卷之间分布LUN可能会很有用。主要原因是控制器条带化。例如、一个HA存储系统可能托管一个数据库、其中需要每个控制器的全部处理和缓存潜力。在这种情况下、典型的设计是、将一半的LUN放置在控制器1上的一个卷中、而将另一半LUN放置在控制器2上的一个卷中。

同样、控制器条带化也可用于负载平衡。如果HA系统托管100个数据库、每个数据库包含10个LUN、则可以设计该系统、其中每个数据库在两个控制器中的每个控制器上都接收一个5 LUN卷。这样、在配置更多数据库时、可以保证每个控制器的负载对称。

但是、这些示例均不涉及卷与LUN的比例为1：1。我们的目标仍然是通过在卷中主机代管相关LUN来优化易管理性。

例如、LUN与卷的比例为1：1就意味着容器化、在容器化中、每个LUN可能真正代表一个工作负载、需要逐个进行管理。在这种情况下、1：1的比例可能是最佳的。

Oracle数据库LUN大小调整和基于LVM的大小调整

当基于SAN的文件系统达到其容量限制时、可通过两种方法增加可用空间：

- 增加LUN的大小
- 将LUN添加到现有卷组并增加包含的逻辑卷

虽然可以选择调整LUN大小来增加容量、但通常最好使用LVM、包括Oracle ASM。存在LVM的一个主要原因是

避免调整LUN大小。通过LVM、多个LUN会绑定到一个虚拟存储池中。从该池中划分出来的逻辑卷由LVM管理、并且可以轻松调整大小。另一个优势是、通过在所有可用LUN之间分布给定逻辑卷、可以避免特定驱动器上出现热点。通常、可以通过使用卷管理器将逻辑卷的底层块区重新定位到新LUN来执行透明迁移。

使用Oracle数据库进行LVM条带化

LVM条带化是指在多个LUN之间分布数据。结果是、许多数据库的性能显著提高。

在闪存驱动器时代之前、条带化用于帮助克服旋转驱动器的性能限制。例如、如果操作系统需要执行1 MB的读取操作、则从单个驱动器读取1 MB的数据将需要大量的驱动器磁头查找和读取、因为1 MB的传输速度较慢。如果在8个LUN上对1 MB的数据进行条带化、则操作系统可以问题描述并行执行8个128 K读取操作、从而减少完成1 MB传输所需的时间。

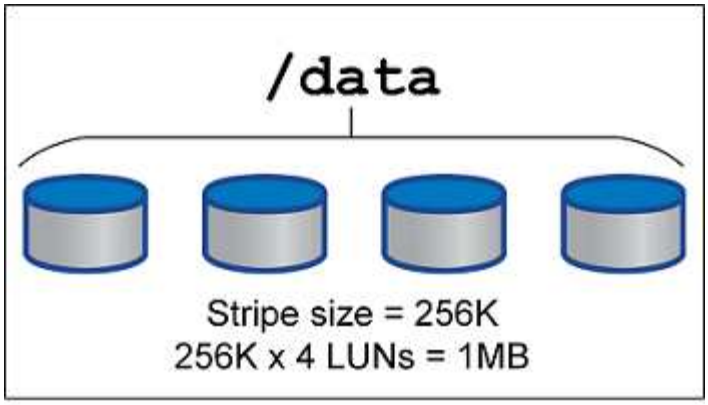
使用旋转驱动器进行条带化更为困难、因为必须事先知道I/O模式。如果条带化未正确调整为真正的I/O模式、则条带化配置可能会损害性能。使用Oracle数据库、尤其是使用全闪存配置时、条带化更易于配置、并且经验证可显著提高性能。

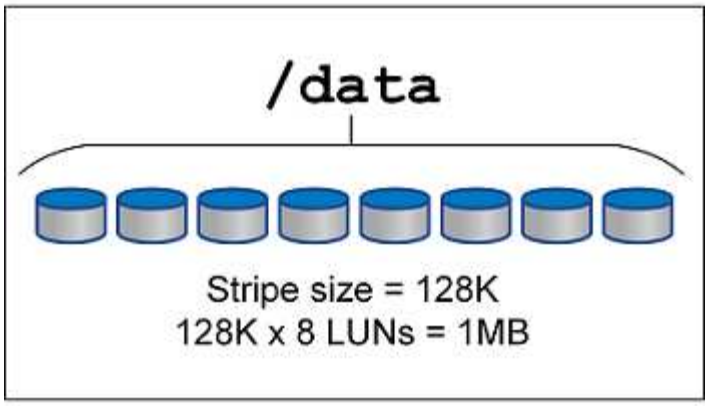
默认情况下、逻辑卷管理器(例如Oracle ASM)会进行条带化、但本机操作系统LVM则不会进行条带化。其中一些会将多个LUN绑定在一起、形成一个串联设备、从而导致数据文件只存在于一个LUN设备上。这会导致热点。其他LVM实施默认使用分布式块区。这与条带化类似、但更粗。卷组中的LUN会被划分为多个大块、称为块区、通常以MB为单位进行测量、然后逻辑卷会分布在这些块区中。结果是、文件的随机I/O应在各个LUN之间分布良好、但顺序I/O操作的效率不如所能达到的高。

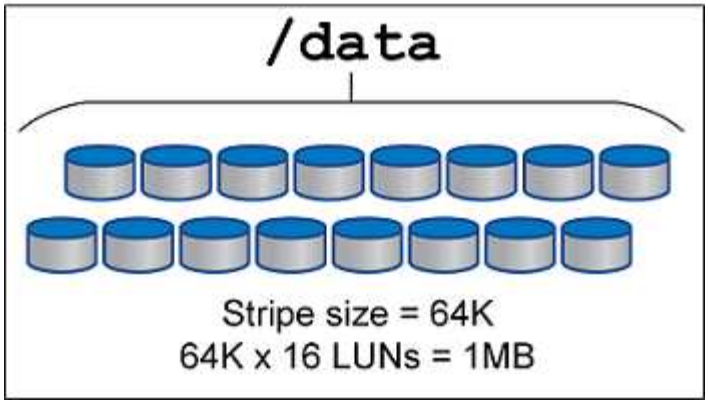
性能密集型应用程序I/O几乎始终为(a)基本块大小单位或(b) 1兆字节。

条带化配置的主要目标是确保单文件I/O可作为一个单元执行、多块I/O (大小应为1 MB)可在条带化卷中的所有LUN之间均匀并行。这意味着条带大小不能小于数据库块大小、条带大小乘以LUN数量应为1 MB。

下图显示了三个可能的条带大小和宽度调整选项。选择LUN数量是为了满足上述性能要求、但在所有情况下、单个条带内的总数据均为1 MB。


/data
Stripe size = 256K
256K x 4 LUNs = 1MB


/data
Stripe size = 128K
128K x 8 LUNs = 1MB


/data
Stripe size = 64K
64K x 16 LUNs = 1MB