MySQL

Enterprise applications

NetApp
October 14, 2024

目录

MySQL

基于ONTAP的MySQL数据库

MySQL及其变体(包括MariaDB和Percona MySQL)是世界上最受欢迎的数据库。

此ONTAP和MySQL数据库文档将取代先前发布的_TR-4722：《基于ONTAP的MySQL数据库最佳实践》。_

ONTAP是MySQL数据库的理想平台、因为ONTAP实际上是为数据库设计的。随机IO延迟优化、高级服务质量(Quality of Service、QoS)和基本FlexClone功能等众多功能专为满足数据库工作负载的需求而创建。

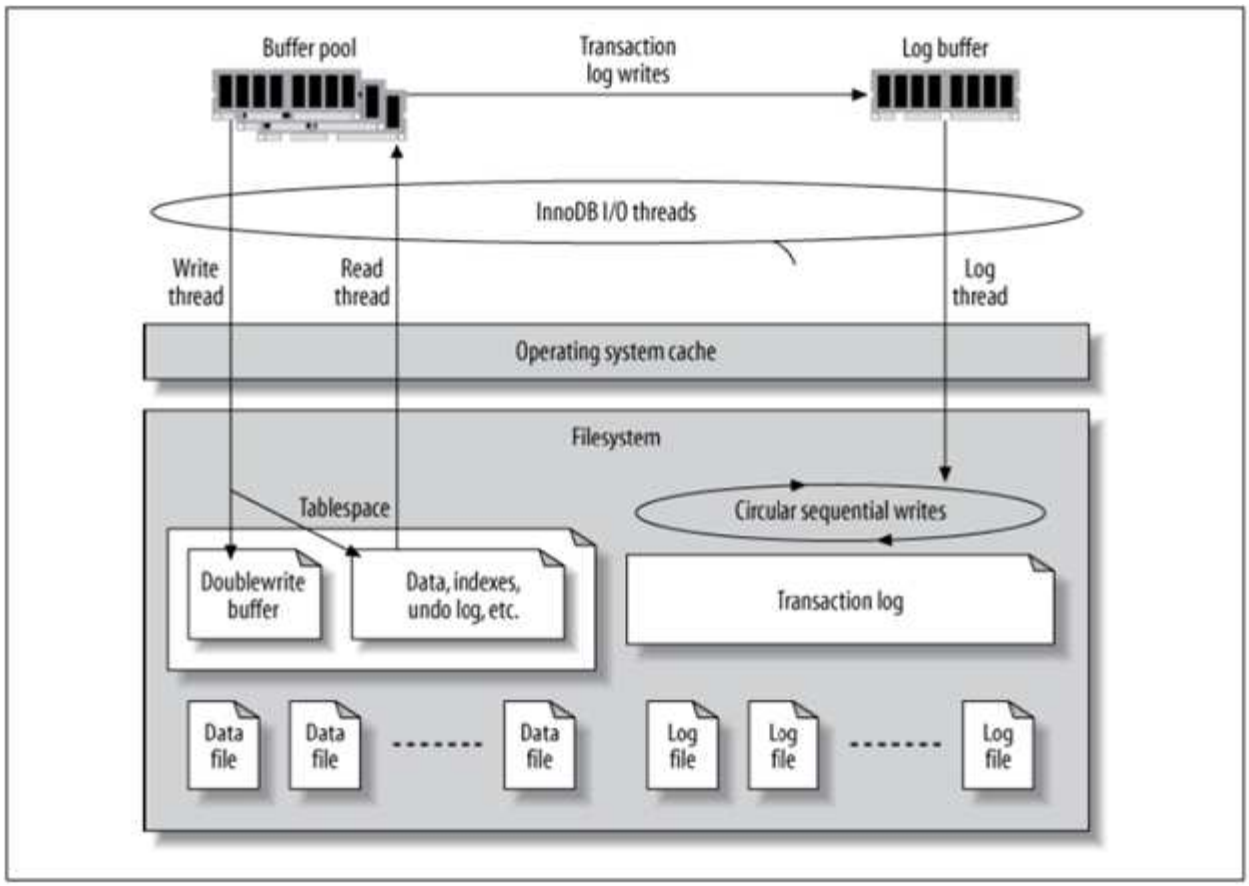
无中断升级(包括更换存储)等其他功能可确保关键数据库始终可用。您还可以通过MetroCluster为大型环境实现即时灾难恢复、或者使用SnapMirror主动同步功能选择数据库。

最重要的是、ONTAP提供无与伦比的性能、能够根据您的独特需求调整解决方案的大小。我们的高端系统可以提供超过100万次IOPS、延迟以微秒为单位、但如果您只需要10万次IOPS、则可以使用仍运行完全相同存储操作系统的较小控制器来调整存储解决方案的大小。

数据库配置

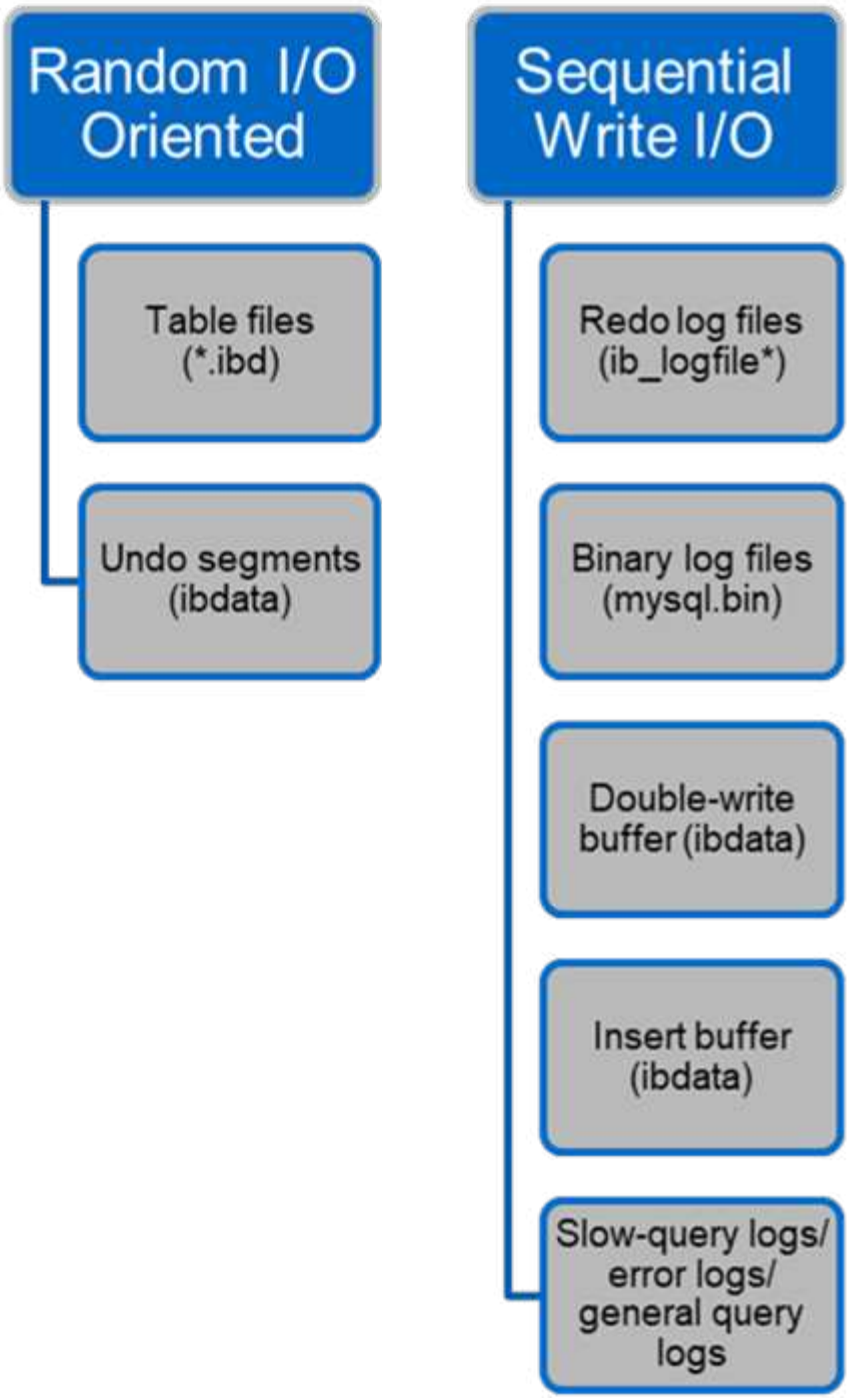
MySQL和InnoDB

InnoDB充当存储和MySQL服务器之间的中间层、它将数据存储到驱动器。



MySQL I/O分为两类：

- 随机文件I/O
- 顺序文件I/O



数据文件会随机读取和覆盖、从而导致IOPS较高。因此、建议使用SSD存储。

重做日志文件和二进制日志文件是事务日志。它们会按顺序写入、因此、您可以通过写入缓存在HDD上获得良好的性能。恢复时会发生顺序读取、但很少会导致性能问题、因为日志文件大小通常小于数据文件、顺序读取比随机读取(发生在数据文件上)速度快。

双写缓冲区是InnoDB的一项特殊功能。InnoDB首先将已刷新的页面写入双写入缓冲区、然后将这些页面写入数据文件上的正确位置。此过程可防止页面损坏。如果没有双写入缓冲区、则在向驱动器写入的过程中发生电源故障时、页面可能会损坏。由于写入双写入缓冲区是顺序的、因此针对HDD进行了高度优化。恢复时进行顺序读取。

由于ONTAP NVRAM已提供写入保护、因此不需要双写缓冲。MySQL有一个参数、`skip_innodb_doublewrite`，禁用双写缓冲区。此功能可以显著提高性能。

插入缓冲区也是InnoDB的一项特殊功能。如果内存中不存在非唯一的二级索引块、则InnoDB会将条目插入到插入缓冲区中、以避免随机I/O操作。插入缓冲区会定期合并到数据库中的二级索引树中。插入缓冲区通过将I/O请求合并到同一个块来减少I/O操作的数量；随机I/O操作可以是顺序的。插入缓冲区也针对HDD进行了高度优化。正常操作期间会同时进行顺序写入和读取。

撤消段是随机I/O导向的。为了保证多版本并发(MVCC)，InnoDB必须在撤消段中注册旧图像。从撤消段读取以前的图像需要随机读取。如果您运行具有可重复读取的长事务(例如、myq_dump—单个事务)或运行长查询、则可能会发生随机读取。因此、在这种情况下、最好在SSD上存储撤消区块。如果只运行短事务或查询、则随机读取不是问题描述。

*由于InnoDB I/O的特性、NetApp建议*采用以下存储设计布局。

- 一个卷、用于存储面向MySQL的随机和顺序I/O文件
- 另一个卷、用于存储MySQL的纯顺序I/O导向文件

此布局还有助于您设计数据保护策略和策略。

MySQL配置参数

NetApp建议使用一些重要的MySQL配置参数来获得最佳性能。

Parameters	值
InnoDB_log_file_size	256 M
INNODB_FLOG_AT_TRx_Commit	2.
InnoDB_doublewite	0
INNODB_FLUG_METHOD	同步
InnoDB_buffer_pool_size	11g
InnoDB_IO_Capacity	8192.
InnoDB_buffer_pool_instances	8.
InnoDB_LRU_SCAN_深度	8192.
open_file_Limit	65535

要设置本节中所述的参数、必须在MySQL配置文件(my.cnf)中进行更改。NetApp最佳实践是内部测试的结果。

InnoDB_log_file_size

为InnoDB日志文件大小选择合适的大小对于写入操作以及在服务器崩溃后有适当的恢复时间非常重要。

由于登录到该文件的事务太多、因此日志文件大小对于写入操作非常重要。修改记录后、所做的更改不会立即写入表空间。而是将更改记录在日志文件的末尾、并将页面标记为脏。InnoDB使用其日志将随机I/O转换为顺序I/O

日志已满时、脏页会按顺序写出到表空间中、以释放日志文件中的空间。例如、假设服务器在事务处理期间崩溃、而写入操作仅记录在日志文件中。在服务器恢复运行之前、它必须经过恢复阶段、在此阶段中、系统将重写记录在日志文件中的更改。日志文件中的条目越多、服务器恢复所需的时间就越长。

在此示例中、日志文件大小会影响恢复时间和写入性能。在为日志文件大小选择正确的数字时、请平衡恢复时间与写入性能。通常、128 M到512 M之间的值都很好。

INNODB_FLOG_AT_TRx_Commit

数据发生更改时、更改不会立即写入存储。

相反、数据会记录在日志缓冲区中、日志缓冲区是InnoDB分配给缓冲区的内存的一部分、用于缓冲日志文件中记录的更改。在提交事务时、缓冲区已满时或每秒一次(以先发生的事件为准)、InnoDB会将缓冲区转至日志文件。用于控制此过程的配置变量为InnoDB_F冲_LOG_AT_TRx_commit。值选项包括：

- 设置时 `innodb_flush_log_trx_at_commit=0`，InnoDB会将修改后的数据(位于InnoDB缓冲池中)写入日志文件(ib_logfile)，并每秒刷新日志文件(写入存储)。但是、提交事务时、它不会执行任何操作。如果发生电源故障或系统崩溃、则未转储的数据都不可恢复、因为它不会写入日志文件或驱动器。
- 设置时 `innodb_flush_log_trx_commit=1`，则InnoDB会将日志缓冲区写入事务日志，并将每个事务转储到持久存储。例如、对于所有事务提交、InnoDB先写入日志、然后再写入存储。存储速度较慢会对性能产生负面影响；例如、每秒InnoDB事务数会减少。
- 设置时 `innodb_flush_log_trx_commit=2`，InnoDB会在每次提交时将日志缓冲区写入日志文件；但是，它不会将数据写入存储。InnoDB每秒刷新一次数据。即使出现电源故障或系统崩溃、选项2数据也会显示在日志文件中、并且可以恢复。

如果性能是主要目标、请将该值设置为2。由于InnoDB每秒向驱动器写入一次、而不是每次提交事务、因此性能会显著提高。如果发生电源故障或崩溃、可以从事务日志中恢复数据。

如果数据安全是主要目标、请将该值设置为1、以便每次提交事务时、InnoDB都会转至驱动器。但是、性能可能会受到影响。

* NetApp建议*将InnoDB_F冲_LOG_TRx_commit值设置为2以提高性能。

InnoDB_doublewite

时间 `innodb_doublewrite` 处于启用状态(默认值)时、InnoDB会将所有数据存储两次：首先存储到双写入缓冲区、然后存储到实际数据文件。

您可以使用关闭此参数 `--skip-innodb_doublewrite` 适用于基准测试、或者您更关心最高性能而非数据完整性或可能出现的故障。InnoDB使用一种称为双写的文件转储技术。在将页面写入数据文件之前、InnoDB会将其写入一个称为双写入缓冲区的连续区域。在完成对双写缓冲区的写入和刷新后、InnoDB会将页面写入到其在数据文件中的正确位置。如果操作系统或myqld进程在页面写入期间崩溃、则InnoDB稍后可以在崩溃恢复期间从双写入缓冲区中找到一个良好的页面副本。

* NetApp建议*禁用双写缓冲区。ONTAP NVRAM具有相同的功能。双重缓冲会不必要地损害性能。

InnoDB_buffer_pool_size

InnoDB缓冲池是任何调整活动最重要的部分。

InnoDB在很大程度上依赖于缓冲区池来缓存索引并对数据、自适应哈希索引、插入缓冲区以及内部使用的许多其他数据结构进行分段。缓冲池还会缓冲对数据的更改、以便不必立即对存储执行写入操作、从而提高性能。缓冲池是InnoDB不可或缺的一部分、必须相应地调整其大小。设置缓冲池大小时、请考虑以下因素：

- 对于仅限InnoDB的专用计算机、请将缓冲池大小设置为可用RAM的80%或更多。
- 如果它不是MySQL专用服务器、请将大小设置为RAM的50%。

INNODB_FLUG_METHOD

InnoDB_flush_方法 参数用于指定InnoDB如何打开和刷新日志和数据文件。

优化

在InnoDB优化中、如果适用、设置此参数会调整数据库性能。

以下选项用于通过InnoDB转储文件：

- `fsync`。InnoDB使用 `fsync()` 系统调用以刷新数据和日志文件。此选项为默认设置。
- `O_DSYNC`。InnoDB使用 `O_DSYNC` 选项打开并刷新日志文件，并使用fsync()刷新数据文件。InnoDB不使用 `O_DSYNC` 这是因为在许多UNIX版本上都存在问题。
- `O_DIRECT`。InnoDB使用 `O_DIRECT` 选项(或 `directio()` 在Solaris上)以打开数据文件并使用 `fsync()` 刷新数据和日志文件。此选项在某些版本的GNU或Linux、FreEBSD,和Solaris上可用。
- `O_DIRECT_NO_FSYNC`。InnoDB使用 `O_DIRECT` 选项；但是、它会跳过 `fsync()` 系统调用。此选项不适用于某些类型的文件系统(例如XFS)。如果不确定文件系统是否需要 `fsync()` 系统调用(例如、要保留所有文件元数据)、请使用 `O_DIRECT` 选项。

观察结果

在NetApp实验室测试中、 `fsync` NFS和SAN上使用了默认选项、与相比、这是一个非常好的性能提升 `O_DIRECT`。将刷新方法用作时 `O_DIRECT` 通过使用ONTAP、我们发现客户端会以串行方式在4096块的边界处写入大量单字节写入。这些写入会增加网络延迟并降低性能。

InnoDB_IO_Capacity

在InnoDB插件中、从MySQL 5.7中添加了一个名为InnoDB_io_Capacity的新参数。

它控制InnoDB执行的最大IOPS数(包括异常页面的转储速率以及插入缓冲区[ibuf]批大小)。InnoDB_IO_Capacity参数可通过InnoDB后台任务(例如从缓冲区池转储页面以及合并更改缓冲区中的数据)设置IOPS上限。

将InnoDB_IO_Capacity参数设置为系统每秒可执行的大约I/O操作数。理想情况下、应尽可能降低设置值、但不要太低、以免后台活动减慢。如果设置值过高、则会从缓冲池中删除数据、并且插入缓冲区的速度过快、无法缓存、因此无法提供显著优势。

* NetApp建议*如果通过NFS使用此设置、则分析IOPS (Sysbench/FIO)的测试结果并相应地设置参数。除非您在InnoDB缓冲池中看到的已修改或脏页面多于所需数量、否则请使用可能的最小值进行转储和清除以保持最新。

请勿使用极值(如20、000或更多)、除非您已证明较低的值不足以满足工作负载要求。

InnoDB_IO_Capacity参数用于调节转储速率和相关I/O

如果将此参数或InnoDB_IO_capity_max参数设置得过高、并因过早转储而浪费I/O操作、可能会严重影响性能。

InnoDB_LRU_SCAN_深度

- `innodb_lru_scan_depth` 参数会影响InnoDB缓冲池的刷新操作的算法和启发式。

调整I/O密集型工作负载的性能专家主要关注此参数。对于每个缓冲池实例、此参数用于指定页面清理程序线程应继续扫描的最近最少使用(Least Recently Used、LRU)页面列表中的下限、以查找要刷新的脏页面。此后台操作每秒执行一次。

您可以向上或向下调整该值、以最大程度地减少可用页数。不要将该值设置得比所需值高得多、因为扫描会产生显著的性能成本。此外、请考虑在更改缓冲池实例数时调整此参数、因为 `innodb_lru_scan_depth * innodb_buffer_pool_instances` 定义页面清理程序线程每秒执行的工作量。

小于默认值的设置适用于大多数工作负载。只有在典型工作负载下具有备用I/O容量时、才应考虑增加此值。相反、如果写入密集型工作负载使I/O容量饱和、请减小该值、尤其是在缓冲池较大的情况下。

open_file_极限 值

- `open_file_limits` 参数用于确定操作系统允许myqld打开的文件数。

运行时此参数的值是系统允许的实际值、可能与服务器启动时指定的值不同。在MySQL无法更改打开文件数量的系统上、此值为0。有效 `open_files_limit` 值基于系统启动时指定的值(如果有)和的值 `max_connections` 和 `table_open_cache` 使用以下公式：

- 10 + `max_connections` + (`table_open_cache` x 2)
- `max_connections` x 5.
- 操作系统限制(如果为正数)
- 如果操作系统限制为无限大： `open_files_limit` 值在启动时指定；如果无、则为5、000

服务器将尝试使用这四个值中的最大值来获取文件描述符数量。如果无法获取多个描述符、则服务器会尝试获取系统允许的多个描述符。

主机配置

MySQL容器化

MySQL数据库容器化正在变得越来越普遍。

底层容器管理几乎始终通过Docker来执行。OpenShift和Kubnetes等容器管理平台使大型容器环境的管理变得更加简单。容器化的优势包括降低成本、因为无需为虚拟机管理程序授予许可证。此外、容器还允许多个数据库彼此隔离运行、同时共享同一个底层内核和操作系统。容器只需几微秒即可完成配置。

NetApp提供了Asta三项功能来提供高级存储管理功能。例如、Asta三元数据可以使在Kubnetes中创建的容器自动在适当的层上配置其存储、应用导出策略、设置快照策略、甚至可以将一个容器克隆到另一个容器。对于追加

信息，请参见 "Astra Trident 文档"。

MySQL和NFSv3插槽表

Linux上的NFSv3性能取决于名为的参数 `tcp_max_slot_table_entries`。

TCP插槽表相当于主机总线适配器(Host Bus Adapter、HBA)队列深度的NFSv3。这些表可控制任何时候都可以处理的NFS操作的数量。默认值通常为16、该值太低、无法实现最佳性能。在较新的Linux内核上会出现相反的问题、这会自动将TCP插槽表限制增加到使NFS服务器充满请求的级别。

为了获得最佳性能并防止出现性能问题、请调整控制TCP插槽表的内核参数。

运行 `sysctl -a | grep tcp.*.slot_table` 命令、并观察以下参数：

```
# sysctl -a | grep tcp.*.slot_table
sunrpc.tcp_max_slot_table_entries = 128
sunrpc.tcp_slot_table_entries = 128
```

所有Linux系统都应包括 `sunrpc.tcp_slot_table_entries`，但只有部分包括 `sunrpc.tcp_max_slot_table_entries`。它们都应设置为128。

小心
如果未设置这些参数、可能会对性能产生显著影响。在某些情况下、性能会受到限制、因为Linux操作系统发出的I/O不足在其他情况下、随着Linux操作系统尝试问题描述的I/O数超过可处理的I/O数、I/O时间会增加。

I/O计划程序和MySQL

Linux内核允许对块设备的I/O计划方式进行低级控制。

各种Linux发行版上的默认值差别很大。MySQL建议您使用 `NOOP` 或 `deadline` 在Linux上具有本机异步I/O (AIO)的I/O计划程序。一般来说、使用NoOps时、NetApp客户和内部测试的结果会更好。

MySQL的InnoDB存储引擎使用Linux上的异步I/O子系统(本机AIO)对数据文件页面执行预读和写入请求。此行为由控制 `innodb_use_native_aio` 配置选项、默认情况下处于启用状态。使用本机AIO时、I/O计划程序的类型对I/O性能的影响更大。执行基准测试、确定哪个I/O计划程序可以为您的工作负载和环境提供最佳结果。

有关配置I/O计划程序的说明、请参见相关的Linux和MySQL文档。

MySQL文件描述符

要运行、MySQL服务器需要文件描述符、而默认值是不够的。

它使用这些表来打开新连接、将表存储在缓存中、创建临时表以解决复杂的查询、以及访问永久性查询。如果myqld无法在需要时打开新文件、则它可能会停止正常运行。此问题描述的一个常见现象是错误24："打开的文件太多"。 可以同时打开的文件描述符myqld的数量由定义 `open_files_limit` 在配置文件中设置的选项 (`/etc/my.cnf`）。但是 `open_files_limit` 也取决于操作系统的限制。这种依赖关系使变量的设置变得更加复杂。

MySQL无法设置它 `open_files_limit` 选项高于下指定的值 `ulimit 'open files'`。因此、您需要在操作系统级别明确设置这些限制、以使MySQL能够根据需要打开文件。有两种方法可在Linux中检查文件限制：

- 。 `ulimit` Command快速为您提供有关允许或锁定的参数的详细问题描述。运行此命令所做的更改不是永久的、将在系统重新启动后擦除。
- 对进行的更改 `/etc/security/limit.conf` 文件是永久文件、不受系统重新启动的影响。

确保更改用户myql的硬限制和软限制。以下摘录来自此配置：

```
mysql hard nofile 65535
mysql soft nofile 65353
```

同时、在中更新相同的配置 `my.cnf` 以充分利用打开文件的限制。

存储配置

使用**NFS**的**MySQL**

MySQL文档建议您在NAS部署中使用NFSv4。

ONTAP NFS传输大小

默认情况下、ONTAP会将NFS IO大小限制为64K。MySQL数据库的随机IO使用的块大小要小得多、远远低于64K的最大值。大型块IO通常会并行运行、因此最大64K也不是一个限制。

在某些工作负载中、最大64K会产生限制。特别是、如果数据库执行的IO数量较少而规模较大、则单线程操作(如完整表扫描备份操作)将更快、更高效地运行。具有数据库工作负载的ONTAP的最佳IO处理大小为256 K。下面列出的适用于特定操作系统的NFS挂载选项已相应地从64K更新为256K。

给定ONTAP SVM的最大传输大小可按如下方式进行更改：

```
Cluster01::> set advanced

Warning: These advanced commands are potentially dangerous; use them only
when directed to do so by NetApp personnel.

Do you want to continue? {y|n}: y

Cluster01::*> nfs server modify -vserver vserver1 -tcp-max-xfer-size
262144
```

请勿将ONTAP上允许的最大传输大小减小到低于当前挂载的NFS文件系统的rsize/wsize值。在某些操作系统中、这可能会导致挂起甚至数据损坏。例如、如果NFS客户端当前设置为rsize/wsize 65536,则ONTAP最大传输大小可以在65536- 1048576之间进行调整,但不会产生任何影响,因为客户端本身是有限的。将最大传输大小减小至65536,可能会损坏可用性或数据。

- NetApp建议使用*

设置以下NFSv4 fstab (/etc/fstab)设置：

```
nfs4 rw,
hard,nointr,bg,vers=4,proto=tcp,noatime,rsize=262144,wsize=262144
```

NFSv3的一个常见问题描述是断电后锁定的InnoDB日志文件。使用时间或切换日志文件解决了此问题描述。但是、NFSv4具有锁定操作、并可跟踪打开的文件和委派。

采用**SAN**的**MySQL**

使用通常的双卷模式为MySQL配置SAN有两种选择。

只要I/O和容量需求不超过单个LUN文件系统的限制、就可以将较小的数据库放置在一对标准LUN上。例如、需要大约2K随机IOPS的数据库可以托管在单个LUN上的单个文件系统上。同样、大小仅为100 GB的数据库可以容纳在一个LUN上、而不会产生管理问题。

大型数据库需要多个LUN。例如、需要100K IOPS的数据库最有可能至少需要八个LUN。由于驱动器的SCSI通道数量不足、单个LUN将成为瓶颈。同样、在一个10 TB LUN上管理一个10 TB数据库也很困难。逻辑卷管理器旨在将多个LUN的性能和容量功能绑定在一起、以提高性能和易管理性。

在这两种情况下、一对ONTAP卷都应足以满足要求。在简单的配置中、数据文件LUN会像日志LUN一样放置在一个专用卷中。使用逻辑卷管理器配置时、数据文件卷组中的所有LUN都将位于一个专用卷中、而日志卷组的LUN将位于另一个专用卷中。

*MySQL建议*在SAN上部署NetApp时使用两个文件系统：

- 第一个文件系统存储所有MySQL数据、包括表空间、数据和索引。
- 第二个文件系统存储所有日志(二进制日志、慢速日志和事务日志)。

以这种方式分隔数据有多种原因、包括：

- 数据文件和日志文件的I/O模式不同。如果将它们分开、则可以使用QoS控制提供更多选项。
- 要充分利用Snapshot技术、需要能够独立还原数据文件。将数据文件与日志文件相结合会影响数据文件还原。
- NetApp SnapMirror技术可用于为数据库提供简单的低RPO灾难恢复功能；但是、它需要为数据文件和日志制定不同的复制计划。

使用此基本的双卷布局可使解决方案适应未来需要、以便在需要时可以使用所有ONTAP功能。

- NetApp建议*使用ext4文件系统格式化驱动器，因为它具有以下功能：
- 日志文件系统(jfs)中使用的块管理功能的扩展方法以及扩展文件系统(xfs)的延迟分配功能。
- ext4允许文件系统最多包含1个外部字节(2^60字节)、文件最多包含16个TB (16 * 2^40字节)。相比之下、ext3文件系统仅支持最大文件系统大小16 TB和最大文件大小2 TB。
- 在ext4文件系统中、多块分配(mbALLO每次 操作)会为一个文件分配多个块、而不是像ext3那样逐个分配。此配置可减少多次调用块分配器的开销、并优化内存分配。
- 虽然XFS是许多Linux分发版的默认设置、但它管理元数据的方式不同、不适用于某些MySQL配置。

- NetApp建议*在mkfs实用程序中使用4k块大小选项、以便与现有块LUN大小保持一致。

```
mkfs.ext4 -b 4096
```

NetApp LUN将数据存储在4 KB物理块中、从而生成八个512字节逻辑块。

如果未设置相同的块大小、I/O将无法与物理块正确对齐、并且可能会在RAID组中的两个不同驱动器中写入数据、从而导致延迟。

请务必对齐I/O、以实现顺畅的读/写操作。但是、如果I/O从逻辑块开始、而逻辑块不是物理块的起始位置、则表示I/O未对齐。只有当I/O操作从逻辑块(即物理块中的第一个逻辑块)开始时、才会对齐。

版权信息

商标信息