MySQL

Enterprise applications

NetApp
January 02, 2026

목차

MySQL

개요

MySQL 및 MariaDB 및 Percona MySQL을 포함한 그 변종은 세계에서 가장 인기있는 데이터베이스입니다.

ONTAP 및 MySQL 데이터베이스에 대한 이 문서는 이전에 게시된 _TR-4722: ONTAP 모범 사례에 기반한 MySQL 데이터베이스를 대체합니다. _

ONTAP는 말 그대로 데이터베이스를 위해 설계되었기 때문에 ONTAP는 MySQL 데이터베이스에 이상적인 플랫폼입니다. 데이터베이스 워크로드의 요구사항을 해결하기 위해 랜덤 IO 지연 시간 최적화에서 고급 서비스 품질(QoS)과 같은 다양한 기능이 특별히 제작되었습니다.

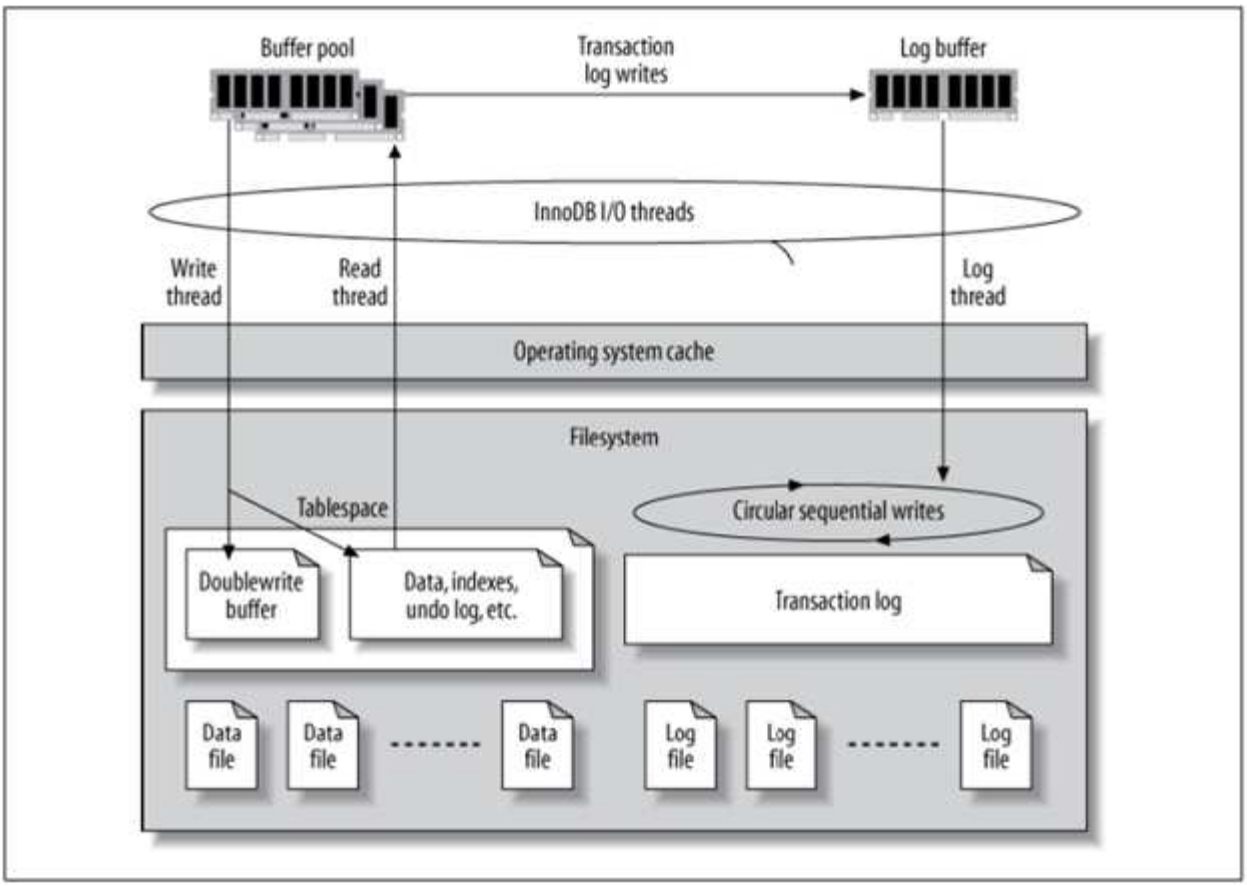
무중단 업그레이드(스토리지 교체 포함)와 같은 추가 기능을 통해 중요 데이터베이스의 가용성을 보장합니다. MetroCluster를 통해 대규모 환경에서 즉시 재해 복구를 수행하거나 SnapMirror 활성 동기화를 사용하여 데이터베이스를 선택할 수도 있습니다.

가장 중요한 것은 ONTAP가 고유한 요구사항에 맞게 솔루션을 사이징하는 능력과 함께 탁월한 성능을 제공한다는 것입니다. NetApp의 하이엔드 시스템은 마이크로초 단위의 지연 시간으로 1M IOPS 이상을 제공할 수 있지만, 100K IOPS만 필요한 경우 정확히 동일한 스토리지 운영 체제를 실행하는 작은 컨트롤러로 스토리지 솔루션을 적절한 크기로 조정할 수 있습니다.

데이터베이스 구성

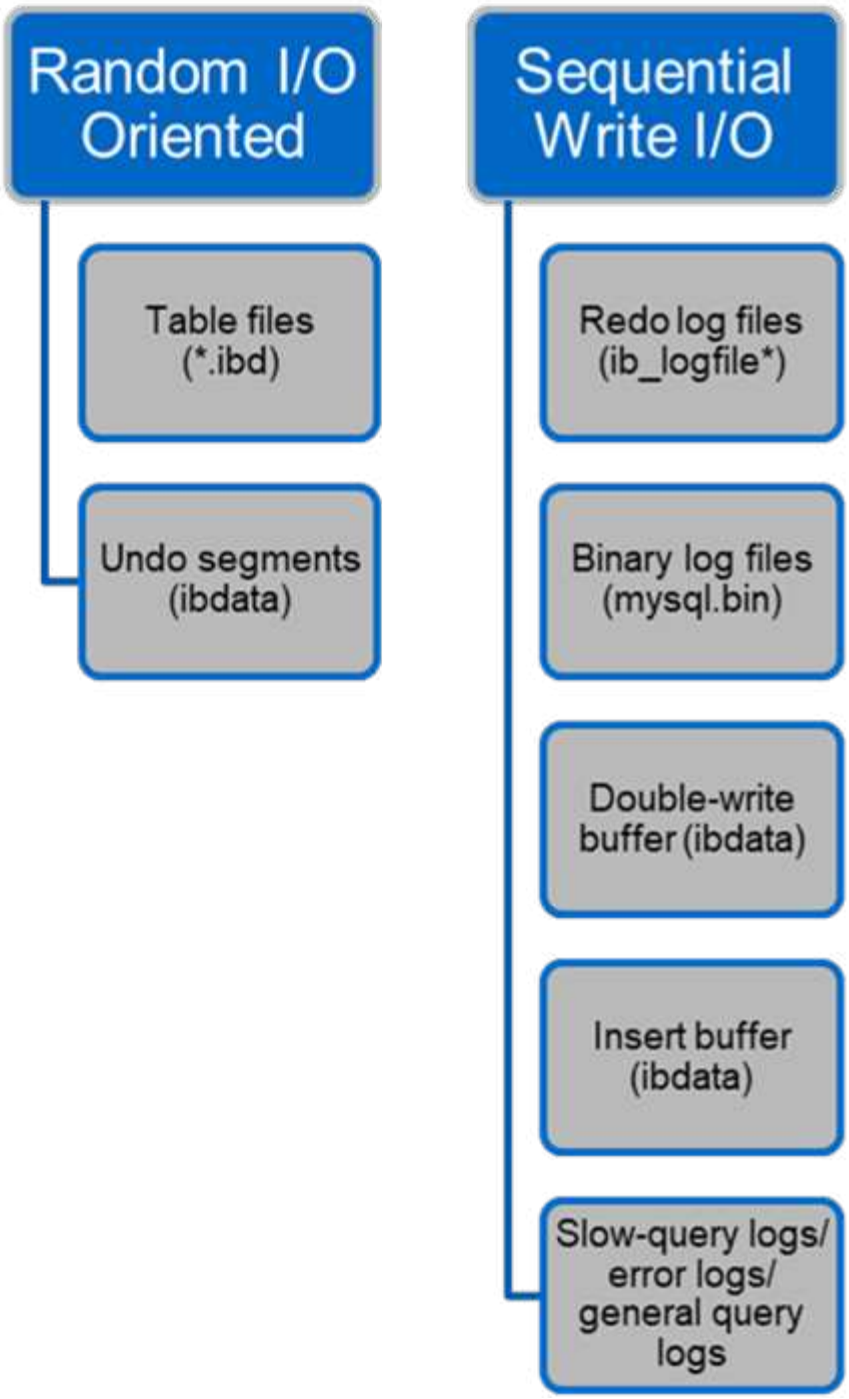
파일 구조

InnoDB는 스토리지와 MySQL 서버 사이의 중간 계층 역할을 하며 데이터를 드라이브에 저장합니다.



MySQL I/O는 두 가지 유형으로 분류됩니다.

- 랜덤 파일 I/O
- 순차적 파일 I/O



데이터 파일을 무작위로 읽고 덮어써서 IOPS가 높아집니다. 따라서 SSD 스토리지를 사용하는 것이 좋습니다.

재실행 로그 파일과 바이너리 로그 파일은 트랜잭션 로그입니다. 순차적으로 작성되므로 쓰기 캐시가 있는 HDD에서 우수한 성능을 얻을 수 있습니다. 순차적 읽기는 복구 시 발생하지만, 일반적으로 로그 파일 크기가 데이터 파일보다 작고 순차 읽기가 랜덤 읽기(데이터 파일에서 발생)보다 빠르므로 성능 문제가 발생하는 경우는 거의 없습니다.

이중 쓰기 버퍼는 InnoDB의 특별한 기능입니다. InnoDB는 먼저 플러시된 페이지를 이중 쓰기 버퍼에 쓴 다음 데이터 파일의 올바른 위치에 페이지를 씁니다. 이 프로세스는 페이지 손상을 방지합니다. 이중 쓰기 버퍼가 없으면 드라이브에 쓰기 프로세스 중에 전원 오류가 발생하면 페이지가 손상될 수 있습니다. 이중 쓰기 버퍼에 쓰는 작업이 순차적이기 때문에 HDD에 맞게 고도로 최적화되어 있습니다. 복구 시 순차적 읽기가 발생합니다.

ONTAP NVRAM은 이미 쓰기 보호를 제공하고 있기 때문에 이중 쓰기 버퍼링이 필요하지 않습니다. MySQL에는 매개 변수가 있습니다. `skip_innodb_doublewrite`이중 쓰기 버퍼를 사용하지 않도록 설정합니다. 이 기능은 성능을 크게 향상시킬 수 있습니다.

INSERT 버퍼는 InnoDB의 특별한 기능이기도 합니다. 고유하지 않은 보조 인덱스 블록이 메모리에 없으면 InnoDB는 임의의 I/O 작업을 방지하기 위해 INSERT 버퍼에 엔트리를 삽입합니다. 주기적으로 삽입 버퍼가 데이터베이스의 보조 인덱스 트리에 병합됩니다. 삽입 버퍼는 I/O 요청을 동일한 블록에 병합하여 I/O 작업 수를 줄입니다. 랜덤 I/O 작업은 순차적일 수 있습니다. 인서트 버퍼는 또한 HDD에 대해 고도로 최적화되어 있습니다. 순차적 쓰기와 읽기는 정상 작업 중에 발생합니다.

실행 취소 세그먼트는 임의 I/O 방향입니다. 다중 버전 동시성(MVCC)을 보장하려면 InnoDB가 실행 취소 세그먼트에 이전 영상을 등록해야 합니다. 실행 취소 세그먼트에서 이전 이미지를 읽으려면 임의 읽기가 필요합니다. 반복 가능한 읽기(예: mysqldump - 단일 트랜잭션)로 긴 트랜잭션을 실행하거나 긴 쿼리를 실행하면 임의 읽기가 발생할 수 있습니다. 따라서 SSD에 실행 취소 세그먼트를 저장하는 것이 더 좋습니다. 짧은 트랜잭션이나 쿼리만 실행할 경우 랜덤 읽기는 문제가 되지 않습니다.

*NetApp는 InnoDB I/O 특성으로 인해 다음과 같은 스토리지 디자인 레이아웃을 권장합니다.

- 단일 볼륨으로 MySQL의 랜덤 및 순차적 I/O 지향 파일을 저장합니다
- MySQL의 순수 순차 I/O 중심 파일을 저장하는 또 다른 볼륨입니다

또한 이 레이아웃은 데이터 보호 정책 및 전략을 설계하는 데 도움이 됩니다.

구성 매개 변수

NetApp은 최적의 성능을 얻기 위해 몇 가지 중요한 MySQL 구성 매개 변수를 권장합니다.

매개 변수	값
InnoDB_log_file_size입니다	256M
InnoDB_flush_log_at_TRx_commit입니다	2
inoDB_doublewrite입니다	0
InnoDB_flush_method 를 참조하십시오	fsync를 참조하십시오
InnoDB_buffer_pool_size입니다	11g
InnoDB_IO_capacity의 약어입니다	8192
InnoDB_buffer_pool_instances입니다	8
InnoDB_LRU_scan_depth 를 선택합니다	8192
open_file_limit를 선택합니다	65535

이 섹션에 설명된 매개 변수를 설정하려면 MySQL 구성 파일(my.cnf)에서 변경해야 합니다. NetApp 모범 사례는 사내에서 수행된 테스트의 결과입니다.

InnoDB_log_file_size입니다

InnoDB 로그 파일 크기에 적합한 크기를 선택하는 것은 쓰기 작업과 서버 충돌 후 적절한 복구 시간을 갖는 데 중요합니다.

많은 트랜잭션이 파일에 로그인되기 때문에 로그 파일 크기는 쓰기 작업에 중요합니다. 레코드가 수정되면 변경 내용이 테이블스페이스에 즉시 다시 기록되지 않습니다. 대신 변경 내용이 로그 파일 끝에 기록되고 페이지가 더티(dirty)로 표시됩니다. InnoDB는 로그를 사용하여 랜덤 I/O를 순차적 I/O로 변환합니다

로그가 가득 차면 로그 파일의 공간을 확보하기 위해 더티 페이지가 테이블스페이스에 순서대로 기록됩니다. 예를 들어 트랜잭션 중에 서버가 충돌하고 쓰기 작업이 로그 파일에만 기록된다고 가정합니다. 서버가 다시 가동되기 전에 로그 파일에 기록된 변경 내용이 재생되는 복구 단계를 거쳐야 합니다. 로그 파일에 있는 항목이 많을수록 서버가 복구하는 데 더 오래 걸립니다.

이 예에서 로그 파일 크기는 복구 시간과 쓰기 성능에 모두 영향을 줍니다. 로그 파일 크기에 적합한 숫자를 선택할 때는 복구 시간과 쓰기 성능의 균형을 맞춥니다. 일반적으로 128M과 512M 사이의 모든 것이 좋은 가치가 있습니다.

InnoDB_flush_log_at_TRx_commit입니다

데이터에 변경 사항이 있을 때 변경 사항이 스토리지에 즉시 기록되지 않습니다.

대신 로그 버퍼에 데이터가 기록됩니다. 로그 버퍼는 InnoDB가 로그 파일에 기록된 버퍼 변경 사항에 할당하는 메모리의 일부입니다. InnoDB는 트랜잭션이 커밋될 때, 버퍼가 가득 찰 때 또는 초당 한 번씩 이벤트가 먼저 발생하는 경우 버퍼를 로그 파일로 플러시합니다. 이 프로세스를 제어하는 구성 변수는 innodb_flush_log_at_TRx_commit입니다. 값 옵션은 다음과 같습니다.

- 를 설정합니다 `innodb_flush_log_trx_at_commit=0`, InnoDB는 InnoDB 버퍼 풀에 있는 수정된 데이터를 로그 파일(IB_LOGFILE)에 쓰고 로그 파일(스토리지에 쓰기)을 매초마다 플러시합니다. 그러나 트랜잭션이 커밋되면 아무 작업도 수행하지 않습니다. 전원 장애나 시스템 충돌이 발생한 경우 플러시되지 않은 데이터는 로그 파일에 기록되지 않으므로 복구할 수 없습니다.
- 를 설정합니다 `innodb_flush_log_trx_commit=1`, InnoDB 는 트랜잭션 로그에 로그 버퍼를 쓰고 모든 트랜잭션에 대해 내구성 있는 저장소로 플러시합니다. 예를 들어, 모든 트랜잭션 커밋에 대해 InnoDB는 로그에 쓴 다음 스토리지에 씁니다. 스토리지 속도가 느리면 성능에 부정적인 영향을 줍니다. 예를 들어 초당 InnoDB 트랜잭션 수가 줄어듭니다.
- 를 설정합니다 `innodb_flush_log_trx_commit=2`InnoDB 는 모든 커밋에서 로그 파일에 로그 버퍼를 쓰지만 스토리지에 데이터를 쓰지 않습니다. InnoDB는 1초에 한 번씩 데이터를 플러시합니다. 전원 장애 또는 시스템 충돌이 발생하더라도 로그 파일에서 옵션 2 데이터를 사용할 수 있으며 복구할 수 있습니다.

성과가 주요 목표인 경우 값을 2로 설정합니다. InnoDB는 모든 트랜잭션 커밋이 아니라 1초에 한 번씩 드라이브에 쓰기 때문에 성능이 크게 향상됩니다. 전원 장애 또는 충돌이 발생하면 트랜잭션 로그에서 데이터를 복구할 수 있습니다.

데이터 안전이 주요 목표인 경우 값을 1로 설정하여 모든 트랜잭션 커밋에 대해 InnoDB가 드라이브로 플러시합니다. 그러나 성능에 영향을 줄 수 있습니다.

* NetApp는 성능 향상을 위해 nodb_flush_log_TRx_commit 값을 2로 설정할 것을 권장합니다.

inoDB_doublewrite입니다

시기 `innodb_doublewrite` 이(기본값)을 사용하도록 설정하면 InnoDB는 모든 데이터를 두 번 저장합니다. 먼저 이중 쓰기 버퍼에 데이터를 저장한 다음 실제 데이터 파일에 저장합니다.

을 사용하여 이 매개 변수를 끌 수 있습니다 `--skip-innodb_doublewrite` 벤치마크용 또는 데이터 무결성 또는 가능한 오류보다 최고 성능에 더 관심이 있는 경우. InnoDB는 double-write라는 파일 플러시 기술을 사용합니다. InnoDB는 데이터 파일에 페이지를 쓰기 전에 이중 쓰기 버퍼라는 인접 영역에 페이지를 씁니다. 이중 쓰기 버퍼에 대한 쓰기 및 플러시가 완료된 후 InnoDB는 데이터 파일의 적절한 위치에 페이지를 씁니다. 페이지 쓰기 중에 운영 체제 또는

mysqld 프로세스가 충돌하는 경우 InnoDB는 나중에 충돌 복구 중에 이중 쓰기 버퍼에서 올바른 페이지 복사본을 찾을 수 있습니다.

* NetApp는 이중 쓰기 버퍼를 비활성화 * 할 것을 권장합니다. ONTAP NVRAM은 동일한 기능을 제공한다. 이중 버퍼링은 불필요하게 성능을 저하시킵니다.

innodb_buffer_pool_size입니다

InnoDB 버퍼 풀은 모든 튜닝 작업에서 가장 중요한 부분입니다.

InnoDB는 인덱스 캐싱 및 데이터 조정, 적응형 해시 인덱스, 삽입 버퍼 및 내부적으로 사용되는 기타 많은 데이터 구조를 위해 버퍼 풀에 크게 의존합니다. 버퍼 풀은 또한 쓰기 작업을 스토리지에 즉시 수행할 필요가 없도록 데이터의 변경 사항을 버퍼링하여 성능을 개선합니다. 버퍼 풀은 InnoDB의 필수 부분이며 그에 따라 크기를 조정해야 합니다. 버퍼 풀 크기를 설정할 때는 다음 요소를 고려하십시오.

- 전용 InnoDB 전용 시스템의 경우 버퍼 풀 크기를 사용 가능한 RAM의 80% 이상으로 설정합니다.
- MySQL 전용 서버가 아닌 경우 크기를 RAM의 50%로 설정합니다.

innodb_flush_method 를 참조하십시오

innodb_flush_method 매개 변수는 InnoDB가 로그 및 데이터 파일을 열고 플러시하는 방법을 지정합니다.

최적화

InnoDB 최적화에서 이 매개 변수를 설정하면 해당되는 경우 데이터베이스 성능이 조정됩니다.

다음 옵션은 InnoDB를 통해 파일을 플러시하는 것입니다.

- `fsync`. InnoDB는 를 사용합니다 `fsync()` 데이터 및 로그 파일을 모두 플러시하기 위한 시스템 호출입니다. 이 옵션이 기본 설정입니다.
- `O_DSYNC`. InnoDB는 를 사용합니다 `O_DSYNC` 로그 파일을 열고 플러시하는 옵션과 데이터 파일을 플러시하기 위한 fsync() 옵션을 선택합니다. InnoDB는 사용하지 않습니다 `O_DSYNC` 유닉스의 많은 변종에서 그것에 문제가 있기 때문에 직접.
- `O_DIRECT`. InnoDB는 를 사용합니다 `O_DIRECT` 옵션(또는 `directio()` Solaris의 경우)를 사용하여 데이터 파일을 열고 사용합니다 `fsync()` 데이터 및 로그 파일을 모두 플러시합니다. 이 옵션은 일부 GNU/Linux 버전, FreeBSD 및 Solaris에서 사용할 수 있습니다.
- `O_DIRECT_NO_FSYNC`. InnoDB는 를 사용합니다 `O_DIRECT` 입출력 플러싱 중에 옵션을 사용할 수 있지만 이 옵션은 를 건너뜁니다 `fsync()` 이후 시스템 호출 이 옵션은 일부 파일 시스템 유형(예: XFS)에는 적합하지 않습니다. 파일 시스템에 가 필요한지 확실하지 않은 경우 `fsync()` 예를 들어 모든 파일 메타데이터를 보존하려면 시스템 호출을 사용합니다 `O_DIRECT` 옵션을 선택합니다.

관찰

NetApp 랩 테스트에서 는 를 선택합니다 `fsync` 기본 옵션은 NFS 및 SAN에서 사용되었으며 에 비해 성능이 탁월했습니다 `O_DIRECT`. 으로 플러시 방법을 사용하는 동안 `O_DIRECT` ONTAP에서는 클라이언트가 4096 블록의 테두리에서 많은 싱글바이트 쓰기를 직렬 방식으로 기록하는 것을 확인했습니다. 이러한 쓰기는 네트워크에서 지연 시간이 증가하고 성능이 저하됩니다.

InnoDB_IO_capacity의 약어입니다

InnoDB 플러그인에서는 MySQL 5.7에서 innodb_io_capacity라는 새 매개 변수가 추가되었습니다.

InnoDB가 수행하는 최대 IOPS 수를 제어합니다(더티 페이지의 플러싱 비율과 삽입 버퍼[ibuf] 배치 크기 포함). innodb_io_capacity 매개 변수는 버퍼 풀에서 페이지를 플러시하거나 변경 버퍼에서 데이터를 병합하는 등 InnoDB 백그라운드 작업에 의한 IOPS의 상한을 설정합니다.

innodb_io_capacity 매개 변수를 시스템이 초당 수행할 수 있는 대략적인 입출력 작업 수로 설정합니다. 가장 좋은 방법은 설정을 가능한 낮게 유지하는 것이지만, 너무 낮게 설정하면 배경 활동이 느려지는 것이 좋습니다. 설정이 너무 높으면 버퍼 풀에서 데이터가 제거되고 캐싱을 위해 너무 빨리 버퍼를 삽입하여 상당한 이점을 얻을 수 있습니다.

* NetApp는 NFS에서 이 설정을 사용할 경우 IOPS(Sysbench/FiO)의 테스트 결과를 분석하고 그에 따라 매개변수를 설정할 것을 권장합니다. InnoDB 버퍼 풀에서 원하는 것보다 더 많은 수정 또는 더티 페이지가 표시되지 않는 한 플러싱과 퍼징에 가능한 가장 작은 값을 사용합니다.

작업량에 더 낮은 값이 충분하지 않다는 사실이 입증되지 않는 한 20,000개 이상의 극단적인 값을 사용하지 마십시오.

InnoDB_IO_CAPACITY 매개변수는 플러싱 속도 및 관련 I/O를 조절합니다

이 매개 변수 또는 innodb_io_capacity_max 매개 변수를 너무 높게 설정하여 성능을 심각하게 저하시킬 수 있습니다

InnoDB_LRU_scan_depth 를 선택합니다

를 클릭합니다 `innodb_lru_scan_depth` 매개 변수는 InnoDB 버퍼 풀에 대한 플러시 작업의 알고리즘 및 휴리스틱에 영향을 줍니다.

이 매개 변수는 I/O 집약적인 워크로드를 성능 전문가에 주로 적용됩니다. 각 버퍼 풀 인스턴스에 대해 이 매개 변수는 LRU(Least Recently Used) 페이지 목록에서 페이지 클리너 스레드가 계속 스캔해야 하는 범위를 지정하며 플러시할 더티 페이지를 찾습니다. 이 백그라운드 작업은 초당 한 번 수행됩니다.

값을 위 또는 아래로 조정하여 사용 가능한 페이지 수를 최소화할 수 있습니다. 스캔에 상당한 성능 비용이 들 수 있으므로 값을 필요 이상으로 설정하지 마십시오. 또한 버퍼 풀 인스턴스의 수를 변경할 때 이 매개 변수를 조정하는 것도 고려하십시오. 그 이유는 무엇입니까 `innodb_lru_scan_depth * innodb_buffer_pool_instances` 페이지 클리너 스레드에서 초당 수행하는 작업 양을 정의합니다.

기본값보다 작은 설정은 대부분의 워크로드에 적합합니다. 일반적인 작업 부하에서 여유 I/O 용량이 있는 경우에만 이 값을 높이는 것이 좋습니다. 반면, 쓰기 집약적 워크로드에서 I/O 용량이 포화되면 특히 버퍼 풀이 있는 경우 값을 줄이십시오.

Open_file_limits 를 참조하십시오

를 클릭합니다 `open_file_limits` 매개 변수는 운영 체제에서 mysqld를 열도록 허용하는 파일 수를 결정합니다.

런타임에 이 매개 변수의 값은 시스템에서 허용하는 실제 값이며 서버 시작 시 지정한 값과 다를 수 있습니다. MySQL이

열려 있는 파일 수를 변경할 수 없는 시스템의 값은 0입니다. 효과 `open_files_limit` 값은 시스템 시작 시 지정된 값(있는 경우)과 의 값을 기반으로 합니다 `max_connections` 및 `table_open_cache` 다음 수식을 사용하여 다음을 실행합니다.

- 10 이상 `max_connections` 를 누릅니다 (`table_open_cache` x 2)
- `max_connections` x 5
- 양수인 경우 운영 체제 제한
- 운영 체제 제한이 무한대인 경우: `open_files_limit` 시작 시 값이 지정되고 없는 경우 5,000입니다

서버는 이 네 개의 최대 값을 사용하여 파일 설명자의 수를 가져오려고 시도합니다. 이렇게 많은 설명자를 얻을 수 없는 경우 서버는 시스템에서 허용하는 수만큼 얻기를 시도합니다.

호스트 구성

컨테이너화

MySQL 데이터베이스의 컨테이너화가 갈수록 보편화되고 있습니다.

낮은 수준의 컨테이너 관리는 거의 항상 Docker를 통해 수행됩니다. OpenShift, Kubernetes와 같은 컨테이너 관리 플랫폼을 사용하면 대규모 컨테이너 환경을 더욱 간편하게 관리할 수 있습니다. 컨테이너화의 이점은 하이퍼바이저에 대한 라이센스를 부여할 필요가 없으므로 비용이 저렴합니다. 또한 컨테이너를 사용하면 여러 데이터베이스를 서로 격리하여 실행하면서 동일한 기본 커널과 운영 체제를 공유할 수 있습니다. 컨테이너는 마이크로초 단위로 프로비저닝할 수 있습니다.

NetApp은 Astra Trident를 제공하여 스토리지의 고급 관리 기능을 제공합니다. 예를 들어, Kubernetes에서 생성된 컨테이너에서 Astra Trident를 사용하면 해당 계층에 스토리지를 자동으로 프로비저닝하고, 엑스포트 정책을 적용하고, 스냅샷 정책을 설정하고, 심지어 컨테이너를 다른 컨테이너에 복제할 수 있습니다. 자세한 내용은 를 "Astra Trident 문서"참조하십시오.

NFSv3 슬롯 테이블

Linux에서 NFSv3 성능은 라는 매개 변수에 따라 다릅니다 `tcp_max_slot_table_entries`.

TCP 슬롯 테이블은 호스트 버스 어댑터(HBA) 큐 길이(queue depth)와 동등한 NFSv3의 기능입니다. 이들 테이블은 한 번에 수행될 수 있는 최대 NFS 작업의 수를 제어합니다. 기본값은 보통 16이며 최적의 성능을 발휘하기에 너무 낮습니다. 최신 Linux 커널에서는 반대의 문제가 발생하는데, 요청을 통해 NFS 서버를 포화시키는 수준까지 TCP 슬롯 테이블의 한계를 자동으로 높일 수 있습니다.

최적의 성능을 얻으면서 성능 문제를 방지하려면 TCP 슬롯 테이블을 제어하는 커널 매개 변수를 조정하십시오.

를 실행합니다 `sysctl -a | grep tcp.*.slot_table` 명령을 실행하고 다음 매개 변수를 관찰합니다.

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

모든 Linux 시스템에는 가 포함되어 있습니다 `sunrpc.tcp_slot_table_entries`그러나 일부에만 가 포함됩니다 `sunrpc.tcp_max_slot_table_entries`. 둘 다 128로 설정해야 합니다.

> 이러한 매개 변수를 설정하지 않으면 성능에 상당한 영향을 미칠 수 있습니다. 경우에 따라 Linux 운영 체제가 충분한 I/O를 실행하지 않기 때문에 성능이 제한됩니다 또는 Linux 운영 체제가 처리할 수 있는 것보다 더 많은 I/O를 발급하려고 하면 I/O 지연 시간이 늘어납니다.

I/O 스케줄러

Linux 커널은 블록 장치의 I/O를 스케줄링하여 낮은 레벨의 제어를 허용합니다.

Linux의 다양한 배포 버전에서의 기본값은 상당히 다릅니다. MySQL은 를 사용할 것을 권장합니다 `NOOP` 또는 a `deadline` Linux에서 네이티브 비동기식 I/O(AIO)를 사용하는 I/O 스케줄러. 일반적으로 NetApp 고객 및 내부 테스트가 NoOps를 통해 더 나은 결과를 나타냈습니다.

MySQL의 InnoDB 스토리지 엔진은 Linux의 비동기 I/O 하위 시스템(네이티브 AIO)을 사용하여 데이터 파일 페이지에 대한 읽기 및 쓰기 요청을 수행합니다. 이 동작은 에 의해 제어됩니다 `innodb_use_native_aio` 구성 옵션 - 기본적으로 활성화되어 있습니다. 기본 AIO를 사용할 경우 입출력 스케줄러 유형이 입출력 성능에 더 큰 영향을 미칩니다. 벤치마크를 수행하여 워크로드 및 환경에 가장 적합한 결과를 제공하는 I/O 스케줄러를 결정합니다.

I/O 스케줄러 구성에 관한 관련 Linux 및 MySQL 설명서를 참조하십시오.

파일 설명자

MySQL 서버를 실행하려면 파일 설명자가 필요하며 기본값이 충분하지 않습니다.

이를 사용하여 새 연결을 열고, 캐시에 테이블을 저장하고, 복잡한 쿼리를 해결하기 위한 임시 테이블을 만들고, 영구 테이블에 액세스합니다. 필요한 경우 mysqld가 새 파일을 열 수 없는 경우 제대로 작동하지 않을 수 있습니다. 이 문제의 일반적인 증상은 오류 24, "열려 있는 파일이 너무 많음"입니다. mysqld가 동시에 열 수 있는 파일 설명자의 수는 에 의해 정의됩니다 `open_files_limit` 구성 파일에 설정된 옵션입니다 (`/etc/my.cnf`)를 클릭합니다. 그러나 `open_files_limit` 또한 운영 체제의 제한에 따라 다릅니다. 이러한 의존성으로 인해 변수를 보다 복잡하게 설정할 수 있습니다.

MySQL을 설정할 수 없습니다 `open_files_limit` 에 지정된 값보다 높은 옵션 `ulimit 'open files'`. 따라서 필요에 따라 MySQL이 파일을 열 수 있도록 운영 체제 수준에서 이러한 제한을 명시적으로 설정해야 합니다. Linux에서 파일 제한을 확인하는 방법에는 두 가지가 있습니다.

- 를 클릭합니다 `ulimit` 명령어는 허용되거나 잠기는 매개변수에 대한 자세한 설명을 빠르게 제공합니다. 이 명령을 실행하여 변경한 사항은 영구적이지 않으며 시스템 재부팅 후 삭제됩니다.
- 의 변경 사항 `/etc/security/limit.conf` 파일은 영구적이며 시스템 재부팅의 영향을 받지 않습니다.

사용자 MySQL의 하드 제한과 소프트 제한값을 모두 변경해야 합니다. 다음 내용은 구성에서 발췌한 것입니다.

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

동시에 에서 동일한 설정을 업데이트합니다 `my.cnf` 열려 있는 파일 제한을 완전히 사용합니다.

스토리지 구성

NFS 를 참조하십시오

MySQL 설명서에서는 NAS 구축에 NFSv4를 사용할 것을 권장합니다.

ONTAP NFS 전송 크기

기본적으로 ONTAP는 NFS IO 크기를 64K로 제한합니다. MySQL 데이터베이스의 랜덤 IO는 최대 64K 이하라는 훨씬 더 작은 블록 크기를 사용합니다. 대형 블록 IO는 일반적으로 병렬화되므로 최대 64K 역시 제한이 없습니다.

일부 워크로드는 최대 64K로 인해 제한이 발생합니다. 특히, 전체 테이블 스캔 백업 작업과 같은 단일 스레드 작업은 데이터베이스가 더 적은 수의 입출력을 수행할 수 있는 경우 더 빠르고 효율적으로 실행됩니다. 데이터베이스 워크로드에서 ONTAP의 최적의 IO 처리 크기는 256K입니다. 아래에 나열된 특정 운영 체제에 대한 NFS 마운트 옵션이 그에 따라 64K에서 256K로 업데이트되었습니다.

특정 ONTAP SVM의 최대 전송 크기를 다음과 같이 변경할 수 있습니다.

```
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
```

ONTAP에서 허용되는 최대 전송 크기를 현재 마운트된 NFS 파일 시스템의 rsize/wsize 값보다 작게 줄이지 마십시오. 이로 인해 일부 운영 체제에서 중단되거나 심지어 데이터 손상이 발생할 수 있습니다. 예를 들어, NFS 클라이언트가 현재 rsize/wsize 65536으로 설정되어 있는 경우 클라이언트 자체가 제한되므로 영향을 미치지 않고 ONTAP 최대 전송 크기를 65536에서 1048576 사이에서 조정할 수 있습니다. 최대 전송 크기를 65536 미만으로 줄이면 가용성 또는 데이터가 손상될 수 있습니다.

NetApp 권장

다음 NFSv4 fstab(/etc/fstab) 설정 설정:

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

NFSv3의 일반적인 문제는 정전 후 잠긴 InnoDB 로그 파일이었습니다. 시간을 사용하거나 로그 파일을 전환하여 이 문제를 해결했습니다. 그러나 NFSv4에는 잠금 작업이 있으며 열려 있는 파일 및 위임을 추적합니다.

산

입출력 및 용량 요구 사항이 단일 LUN 파일 시스템의 제한 범위 내에 있는 경우 더 작은 데이터베이스를 한 쌍의 표준 LUN에 배치할 수 있습니다. 예를 들어, 약 2K 랜덤 IOPS가 필요한 데이터베이스는 단일 LUN의 단일 파일 시스템에서 호스팅될 수 있습니다. 마찬가지로, 크기가 100GB인 데이터베이스도 관리 문제를 일으키지 않고 단일 LUN에 적합합니다.

데이터베이스가 클수록 여러 개의 LUN이 필요합니다. 예를 들어, 100K IOPS가 필요한 데이터베이스에는 일반적으로 8개 이상의 LUN이 필요할 수 있습니다. 드라이브에 대한 SCSI 채널 수가 충분하지 않기 때문에 단일 LUN에 병목 현상이 발생합니다. 마찬가지로 단일 10TB LUN에서는 10TB 데이터베이스를 관리하기가 어렵습니다. 논리적 볼륨 관리자는 여러 LUN의 성능과 용량 기능을 함께 결합하여 성능과 관리 효율성을 높이도록 설계되었습니다.

두 경우 모두 한 쌍의 ONTAP 볼륨으로 충분합니다. 간단한 구성을 사용하면 데이터 파일 LUN이 로그 LUN과 마찬가지로 전용 볼륨에 배치됩니다. 논리적 볼륨 관리자를 구성하면 데이터 파일 볼륨 그룹의 모든 LUN이 전용 볼륨에 있고 로그 볼륨 그룹의 LUN은 두 번째 전용 볼륨에 있게 됩니다.

- NetApp는 * SAN에서 MySQL 배포를 위해 두 개의 파일 시스템을 사용할 것을 권장합니다.
- 첫 번째 파일 시스템은 테이블스페이스, 데이터 및 인덱스를 포함한 모든 MySQL 데이터를 저장합니다.
- 두 번째 파일 시스템은 모든 로그(바이너리 로그, 느린 로그 및 트랜잭션 로그)를 저장합니다.

이러한 방식으로 데이터를 분리해야 하는 이유는 다음과 같습니다.

- 데이터 파일과 로그 파일의 I/O 패턴은 다릅니다. 이들 포트를 분리하면 QoS 제어에서 더 많은 옵션을 사용할 수 있습니다.
- 스냅샷 기술을 최적으로 사용하려면 데이터 파일을 독립적으로 복원할 수 있는 기능이 필요합니다. 데이터 파일을 로그 파일과 함께 사용하면 데이터 파일 복구가 방해됩니다.
- NetApp SnapMirror 기술을 사용하여 데이터베이스에 단순한 RPO 재해 복구 기능을 제공할 수 있지만 데이터 파일 및 로그에 대해 서로 다른 복제 일정이 필요합니다.

필요한 경우 모든 ONTAP 기능을 사용할 수 있도록 미래에 대비한 솔루션을 이 기본적인 두 볼륨 레이아웃을 사용하십시오.

*NetApp는 다음과 같은 기능 때문에 ext4 파일 시스템으로 드라이브를 포맷할 것을 권장합니다.

- JFS(저널링 파일 시스템)에서 사용되는 블록 관리 기능과 XFS(확장 파일 시스템)의 지연 할당 기능에 대한 확장 접근 방식
- ext4는 최대 1개의 exbibyte(2의 60바이트)의 파일 시스템과 최대 16테비바이트(16*2 40바이트)의 파일을 허용합니다. 반대로 ext3 파일 시스템은 최대 파일 시스템 크기 16TB와 최대 파일 크기 2TB만 지원합니다.
- ext4 파일 시스템에서 다중 블록 할당(mballoc)은 ext3에서와 같이 파일에 대해 하나씩 할당하는 대신 단일 작업으로 파일에 대해 여러 블록을 할당합니다. 이 구성은 블록 할당자를 여러 번 호출하는 오버헤드를 줄이고 메모리 할당을 최적화합니다.
- XFS가 대부분의 Linux 배포판의 기본값이지만 메타데이터를 다르게 관리하므로 일부 MySQL 구성에 적합하지 않습니다.

- NetApp는 기존 블록 LUN 크기에 맞추기 위해 mkfs 유틸리티와 함께 4K 블록 크기 옵션을 사용할 것을 권장합니다.

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

NetApp LUN은 데이터를 4KB 물리적 블록에 저장하여 8개의 512바이트 논리적 블록을 생성합니다.

동일한 블록 크기를 설정하지 않을 경우 I/O가 물리적 블록과 올바르게 정렬되지 않고 RAID 그룹에 있는 두 개의 드라이브에 쓰므로 지연 시간이 발생합니다.

원활한 읽기/쓰기 작업을 위해 I/O를 맞추는 것이 중요합니다. 하지만 물리적 블록의 시작이 아닌 논리적 블록에서 I/O가 시작하면 I/O가 정렬 불량이 됩니다. I/O 작업은 물리적 블록의 첫 번째 논리적 블록인 논리적 블록에서 시작할 때만 정렬됩니다.

저작권 정보

상표 정보