PostgreSQL

Enterprise applications

NetApp
January 02, 2026

목차

PostgreSQL

개요

PostgreSQL에는 PostgreSQL, PostgreSQL Plus 및 EDB Postgres Advanced Server(EPAS)가 포함된 변종이 함께 제공됩니다. PostgreSQL은 일반적으로 다중 계층 애플리케이션을 위한 백엔드 데이터베이스로 구축됩니다. 일반적인 미들웨어 패키지(PHP, Java, Python, Tcl/Tk, ODBC, 또한 JDBC는 오픈 소스 데이터베이스 관리 시스템에서 널리 사용되고 있습니다. ONTAP는 PostgreSQL 데이터베이스를 실행할 때 탁월한 선택으로 신뢰성, 고성능 및 효율적인 데이터 관리 기능을 제공합니다.

ONTAP 및 PostgreSQL 데이터베이스에 대한 이 문서는 이전에 게시된 _TR-4770:PostgreSQL 데이터베이스를 ONTAP 모범 사례에 대한 데이터베이스로 대체합니다

데이터가 기하급수적으로 늘어남에 따라 데이터 관리가 기업의 복잡성이 더욱더 복잡해지고 있습니다. 이러한 복잡성으로 인해 라이센스, 운영, 지원 및 유지 관리 비용이 증가합니다. 전체 TCO를 절감하려면 안정적인 고성능 백엔드 스토리지를 사용하여 상용 데이터베이스에서 오픈 소스 데이터베이스로 전환하는 것을 고려해 보십시오.

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

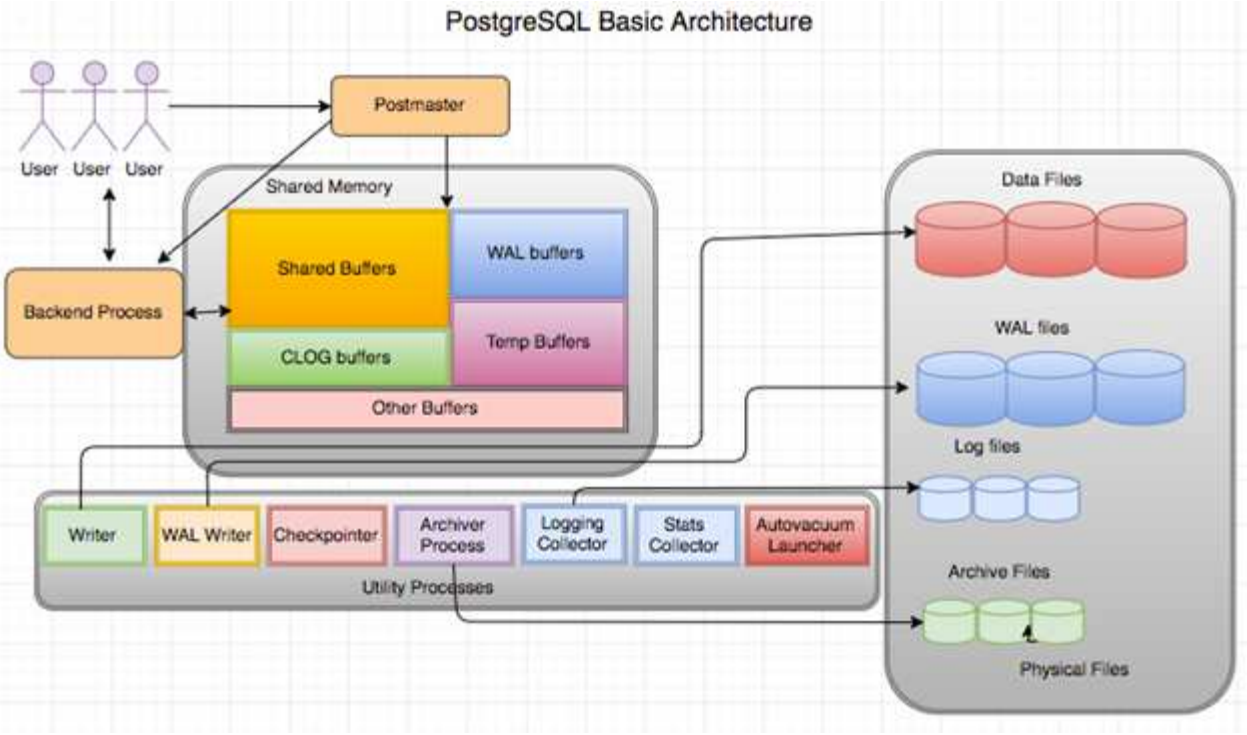
무중단 업그레이드(스토리지 교체 포함)와 같은 추가 기능을 통해 중요 데이터베이스의 가용성을 보장합니다. MetroCluster를 통해 대규모 환경에서 즉시 재해 복구를 수행하거나 SnapMirror 활성 동기화를 사용하여 데이터베이스를 선택할 수도 있습니다.

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, 동일한 스토리지 운영 체제를 실행하는 작은 컨트롤러로 스토리지 솔루션을 적정 크기로 조정할 수 있습니다.

데이터베이스 구성

있습니다

PostgreSQL은 클라이언트 및 서버 아키텍처를 기반으로 하는 RDBMS입니다. PostgreSQL 인스턴스는 데이터베이스 클러스터라고 하며, 이 클러스터는 서버 컬렉션이 아닌 데이터베이스의 모음입니다.



PostgreSQL 데이터베이스에는 포스트마스터, 프런트 엔드(클라이언트) 및 백 엔드의 세 가지 주요 요소가 있습니다 클라이언트는 IP 프로토콜 및 연결할 데이터베이스와 같은 정보를 사용하여 Postmaster에 요청을 보냅니다. 포스트마스터는 연결을 인증하고 추가 통신을 위해 백 엔드 프로세스에 전달합니다. 백 엔드 프로세스는 쿼리를 실행하고 결과를 프런트 엔드(클라이언트)로 직접 보냅니다.

PostgreSQL 인스턴스는 다중 스레드 모델 대신 다중 프로세스 모델을 기반으로 합니다. 이 프로세스는 서로 다른 작업에 대해 여러 프로세스를 생성하며 각 프로세스에는 고유한 기능이 있습니다. 주요 프로세스에는 클라이언트 프로세스, Wal 작성기 프로세스, 백그라운드 작성기 프로세스 및 체크포인트 프로세스가 포함됩니다.

- 클라이언트(포그라운드) 프로세스가 PostgreSQL 인스턴스에 읽기 또는 쓰기 요청을 보내면 디스크에 직접 데이터를 읽거나 쓰지 않습니다. 먼저 공유 버퍼와 Wal(Write-Ahead Logging) 버퍼에 데이터를 버퍼링합니다.
- Wal 작성기 프로세스는 Wal 로그에 쓸 공유 버퍼 및 Wal 버퍼의 내용을 조작합니다. WAL 로그는 일반적으로 PostgreSQL의 트랜잭션 로그이며 순차적으로 기록됩니다. 따라서 데이터베이스의 응답 시간을 향상시키기 위해 PostgreSQL은 먼저 트랜잭션 로그에 기록하고 클라이언트를 확인합니다.
- 데이터베이스를 일관된 상태로 만들기 위해 백그라운드 작성기 프로세스는 공유 버퍼에 더티 페이지가 있는지 주기적으로 검사합니다. 그런 다음 데이터를 NetApp 볼륨 또는 LUN에 저장된 데이터 파일로 플러싱합니다.
- 또한 체크포인터 프로세스는 백그라운드 프로세스보다 빈도가 낮아 주기적으로 실행되며 버퍼가 수정되지 않습니다. NetApp 디스크에 저장된 Wal 로그 끝에 체크포인트 레코드를 쓰고 플러시하도록 Wal 작성기 프로세스에 신호를 보냅니다. 또한 백그라운드 작성기 프로세스에 모든 더티 페이지를 기록하고 디스크에 플러시하도록 신호를 보냅니다.

초기화 매개 변수

를 사용하여 새 데이터베이스 클러스터를 생성합니다 `initdb` 프로그램. AN `initdb` 스크립트는 클러스터를 정의하는 데이터 파일, 시스템 테이블 및 템플릿 데이터베이스(template0 및 Template1)를 생성합니다.

템플릿 데이터베이스는 재고 데이터베이스를 나타냅니다. 시스템 테이블, 표준 보기, 함수 및 데이터 형식에 대한 정의가 포함되어 있습니다. `pgdata` 에 대한 인수 역할을 합니다 `initdb` 데이터베이스 클러스터의 위치를 지정하는

스크립트입니다.

PostgreSQL의 모든 데이터베이스 개체는 해당 OID에 의해 내부적으로 관리됩니다. 테이블 및 인덱스도 개별 OID에 의해 관리됩니다. 데이터베이스 개체와 해당 OID 간의 관계는 개체 유형에 따라 적절한 시스템 카탈로그 테이블에 저장됩니다. 예를 들어 데이터베이스 및 힙 테이블의 OID는 에 저장됩니다 `pg_database` 및 ``pg_class. PostgreSQL 클라이언트에서 쿼리를 실행하여 OID를 확인할 수 있습니다.

모든 데이터베이스에는 1GB로 제한된 고유한 개별 테이블 및 인덱스 파일이 있습니다. 각 테이블에는 각각 접미사가 붙은 두 개의 관련 파일이 있습니다 `_fsm` 및 `_vm`. 이를 여유 공간 지도와 가시성 지도라고 합니다. 이러한 파일은 사용 가능한 공간 용량에 대한 정보를 저장하며 테이블 파일의 각 페이지를 볼 수 있습니다. 인덱스에는 개별 가용 공간 맵만 있으며 가시성 맵은 없습니다.

를 클릭합니다 `pg_xlog/pg_wal` 디렉토리에는 미리 쓰기 로그가 포함됩니다. 미리 쓰기 로그는 데이터베이스의 안정성과 성능을 개선하는 데 사용됩니다. 테이블의 행을 업데이트할 때마다 PostgreSQL은 먼저 미리 쓰기 로그에 변경 내용을 쓰고 나중에 실제 데이터 페이지에 대한 수정 내용을 디스크에 씁니다. 를 클릭합니다 `pg_xlog` 디렉터리에는 일반적으로 여러 파일이 포함되어 있지만 initdb는 첫 번째 파일만 만듭니다. 필요에 따라 추가 파일이 추가됩니다. 각 xlog 파일의 길이는 16MB입니다.

설정

성능을 향상시킬 수 있는 PostgreSQL 튜닝 구성에는 여러 가지가 있습니다.

가장 일반적으로 사용되는 매개 변수는 다음과 같습니다.

- `max_connections = <num>`: 한 번에 가질 최대 데이터베이스 연결 수입니다. 이 매개 변수를 사용하여 디스크로의 스와핑을 제한하고 성능을 중단합니다. 응용 프로그램 요구 사항에 따라 연결 풀 설정에 대해 이 매개 변수를 조정할 수도 있습니다.
- `shared_buffers = <num>`: 데이터베이스 서버의 성능을 개선하는 가장 간단한 방법입니다. 최신 하드웨어의 기본값은 LOW 입니다. 배포 중에 시스템에서 사용 가능한 RAM의 약 25%로 설정됩니다. 이 매개 변수 설정은 특정 데이터베이스 인스턴스에서 작동하는 방식에 따라 달라집니다. 시행 착오로 값을 늘리거나 줄여야 할 수도 있습니다. 그러나 높게 설정하면 성능이 저하될 수 있습니다.
- `effective_cache_size = <num>`: 이 값은 PostgreSQL의 최적화 프로그램에 PostgreSQL이 데이터를 캐싱하는 데 사용할 수 있는 메모리 양을 알려주고 인덱스를 사용할지 여부를 결정하는 데 도움이 됩니다. 값이 클수록 인덱스를 사용할 가능성이 높아집니다. 이 매개 변수는 에 할당된 메모리 크기로 설정해야 합니다 `shared_buffers` 사용 가능한 OS 캐시 양에 더합니다. 이 값은 전체 시스템 메모리의 50%를 초과하는 경우가 많습니다.
- `work_mem = <num>`: 이 매개변수는 정렬 작업 및 해시 테이블에 사용할 메모리 양을 제어합니다. 응용 프로그램에서 무거운 정렬을 수행하는 경우 메모리 양을 늘려야 할 수 있지만 주의해야 합니다. 이는 시스템 차원 매개 변수가 아니라 작업당 매개 변수입니다. 복합 쿼리에 여러 개의 정렬 작업이 있는 경우 여러 개의 work_mem 메모리 유닛을 사용하며 여러 백 엔드에서 동시에 이 작업을 수행할 수 있습니다. 값이 너무 큰 경우 이 쿼리로 인해 데이터베이스 서버가 스왑될 수 있습니다. 이전 버전의 PostgreSQL에서는 이 옵션을 sort_mem이라고 했습니다.
- `fsync = <boolean> (on or off)`: 이 매개변수는 트랜잭션이 커밋되기 전에 fsync()를 사용하여 모든 Wal 페이지를 디스크에 동기화할지 여부를 결정합니다. 이 기능을 끄면 쓰기 성능이 향상될 수 있으며 이 기능을 켜면 시스템이 충돌할 때 손상 위험으로부터 보호됩니다.
- `checkpoint_timeout`: 체크포인트 프로세스는 커밋된 데이터를 디스크로 플러시합니다. 여기에는 디스크에서 많은 양의 읽기/쓰기 작업이 사용됩니다. 이 값은 초 단위로 설정되고 값이 낮을수록 충돌 복구 시간이 줄어들고 값을 늘리면 체크포인트 호출을 줄여 시스템 리소스의 부하가 감소할 수 있습니다. 애플리케이션 중요도, 사용량, 데이터베이스 가용성에 따라 checkpoint_timeout 값을 설정합니다.
- `commit_delay = <num>` 및 `commit_siblings = <num>`: 이러한 옵션은 한 번에 커밋되는 여러 트랜잭션을 작성함으로써 성능을 향상시키는 데 사용됩니다. 트랜잭션이 커밋되는 즉시 여러 개의 commit_siblings

개체가 활성 상태인 경우 서버는 commit_delay microseconds가 한 번에 여러 트랜잭션을 커밋할 때까지 기다립니다.

- `max_worker_processes / max_parallel_workers`: 프로세스에 대한 최적의 작업자 수를 구성합니다. max_parallel_workers는 사용 가능한 CPU 수에 해당합니다. 응용 프로그램 설계에 따라 병렬 작업의 경우 쿼리에 필요한 작업자의 수가 줄어들 수 있습니다. 두 매개 변수의 값을 동일하게 유지하되 테스트 후 값을 조정하는 것이 좋습니다.
- `random_page_cost = <num>`: 이 값은 PostgreSQL이 비순차적 디스크 읽기를 보는 방식을 제어합니다. 값이 높을수록 PostgreSQL은 인덱스 검사 대신 순차 검사를 사용할 가능성이 높으므로 서버에 빠른 디스크가 있음을 나타냅니다. 계획 기반 최적화, 진공 청소, 쿼리 또는 스키마를 변경하는 인덱싱 등의 다른 옵션을 평가한 후 이 설정을 수정합니다.
- `effective_io_concurrency = <num>`: 이 매개변수는 PostgreSQL이 동시에 실행하려고 시도하는 동시 디스크 I/O 작업의 수를 설정합니다. 이 값을 높이면 개별 PostgreSQL 세션이 병렬로 시작하려고 하는 입출력 작업 수가 증가합니다. 허용되는 범위는 1에서 1,000까지이며, 비동기 I/O 요청 발급을 비활성화하려면 0입니다. 현재 이 설정은 비트맵 힙 스캔에만 영향을 줍니다. SSD(Solid-State Drive)와 기타 메모리 기반 스토리지(NVMe)는 동시 요청을 많이 처리할 수 있으므로 수백 개의 요청이 최고의 가치를 실현할 수 있습니다.

PostgreSQL 구성 매개 변수의 전체 목록은 PostgreSQL 설명서를 참조하십시오.

토스트

TOAST는 특대형 특성 저장 기술을 의미합니다. PostgreSQL은 고정된 페이지 크기(일반적으로 8KB)를 사용하며 튜플이 여러 페이지에 걸쳐 있을 수 없습니다. 따라서 큰 필드 값을 직접 저장할 수 없습니다. 이 크기를 초과하는 행을 저장하려고 할 때 TOAST는 큰 열의 데이터를 작은 "조각"으로 나눈 다음 TOAST 테이블에 저장합니다.

토스트 속성의 큰 값은 결과 집합이 클라이언트로 전송될 때만(선택한 경우) 당겨집니다. TOAST(Out-of-Line Storage)가 없을 때보다 테이블 자체가 훨씬 작고 공유 버퍼 캐시에 더 많은 행을 저장할 수 있습니다.

진공

일반적인 PostgreSQL 작업에서는 업데이트로 삭제되거나 폐기된 튜플은 테이블에서 물리적으로 제거되지 않으며 진공이 실행될 때까지 그대로 유지됩니다. 따라서, 특히 자주 업데이트되는 테이블에서 정기적으로 진공을 실행해야 합니다. 그런 다음 디스크 공간 중단을 방지하기 위해 새 행에서 재사용할 수 있도록 해당 공간을 재확보해야 합니다. 그러나 운영 체제에 공간을 반환하지 않습니다.

페이지 내의 여유 공간은 조각화되지 않습니다. Vacuum은 전체 블록을 다시 쓰므로 나머지 행을 효율적으로 압축하고 페이지에 연속된 단일 여유 공간 블록을 남깁니다.

반면, VACUM FULL은 데드 공간 없이 완전히 새로운 버전의 테이블 파일을 작성하여 테이블을 적극적으로 압축합니다. 이렇게 하면 테이블 크기가 최소화되지만 시간이 오래 걸릴 수 있습니다. 또한 작업이 완료될 때까지 테이블의 새 복사본을 위한 추가 디스크 공간이 필요합니다. 일상적인 진공의 목적은 진공이 완전히 작동하지 않도록 하는 것입니다. 이 프로세스는 테이블을 최소 크기로 유지할 뿐만 아니라 디스크 공간의 안정적 상태 사용도 유지합니다.

테이블스페이스

데이터베이스 클러스터가 초기화될 때 두 개의 테이블스페이스가 자동으로 생성됩니다.

를 클릭합니다 `pg_global` 테이블스페이스는 공유 시스템 카탈로그에 사용됩니다. 를 클릭합니다 `pg_default` 테이블스페이스는 Template1 및 template0 데이터베이스의 기본 테이블스페이스입니다. 클러스터가 초기화된 파티션이나 볼륨에 공간이 부족하여 확장할 수 없는 경우 다른 파티션에 테이블스페이스를 생성하여 시스템을 재구성할 때까지 사용할 수 있습니다.

자주 사용되는 인덱스는 솔리드 스테이트 디바이스처럼 빠른 고가용성 디스크에 배치할 수 있습니다. 또한 드물게 사용되거나 성능이 중요하지 않은 아카이브 데이터를 저장하는 테이블을 SAS 또는 SATA 드라이브와 같이 더 저렴하고 느린 디스크 시스템에 저장할 수 있습니다.

테이블스페이스는 데이터베이스 클러스터의 일부이며 데이터 파일의 자동 모음으로 취급할 수 없습니다. 기본 데이터 디렉토리에 포함된 메타데이터에 따라 달라지므로 다른 데이터베이스 클러스터에 연결하거나 개별적으로 백업할 수 없습니다. 마찬가지로 파일 삭제, 디스크 오류 등을 통해 테이블스페이스를 잃으면 데이터베이스 클러스터를 읽을 수 없거나 시작할 수 없게 됩니다. RAM 디스크와 같은 임시 파일 시스템에 테이블스페이스를 배치하면 전체 클러스터의 안정성이 저하됩니다.

생성된 후 요청 사용자에게 충분한 권한이 있는 경우 모든 데이터베이스에서 테이블스페이스를 사용할 수 있습니다. PostgreSQL은 심볼 링크를 사용하여 테이블스페이스의 구현을 간소화합니다. PostgreSQL은 에 행을 추가합니다 `pg_tablespace` 테이블(클러스터 전체 테이블) 및 새 OID(개체 식별자)를 해당 행에 할당합니다. 마지막으로, 서버는 OID를 사용하여 클러스터와 지정된 디렉토리 사이에 심볼 링크를 생성합니다. 디렉터리 `$PGDATA/pg_tblspc` 클러스터에 정의되어 있지 않은 각 테이블스페이스를 가리키는 심볼 링크를 포함합니다.

스토리지 구성

NFS 를 참조하십시오

PostgreSQL 데이터베이스는 NFSv3 또는 NFSv4 파일 시스템에서 호스팅할 수 있습니다. 가장 좋은 옵션은 데이터베이스 외부의 요인에 따라 달라집니다.

예를 들어 특정 클러스터 환경에서는 NFSv4 잠금 동작이 더 바람직할 수 있습니다. (를 참조하십시오 "여기" 참조)

그렇지 않으면 성능을 포함하여 데이터베이스 기능이 거의 동일해야 합니다. 유일한 요구 사항은 를 사용하는 것입니다 `hard` 마운트 옵션. 소프트 타임아웃이 복구할 수 없는 입출력 오류를 생성하지 않도록 하기 위해 필요합니다.

NFSv4를 프로토콜로 선택하는 경우 NetApp에서는 NFSv4.1을 사용할 것을 권장합니다. NFSv4.1에서는 NFSv4 프로토콜의 몇 가지 기능이 개선되어 NFSv4.0에 대한 복구 성능이 향상되었습니다.

일반 데이터베이스 워크로드에 다음 마운트 옵션을 사용하십시오.

```
rw,hard,nointr,bg,vers=[3|4],proto=tcp,rsize=65536,wsize=65536
```

순차적 IO가 많이 필요할 경우 다음 섹션에서 설명하는 대로 NFS 전송 크기를 늘릴 수 있습니다.

NFS 전송 크기

기본적으로 ONTAP에서는 NFS I/O 크기를 64K로 제한합니다.

대부분의 애플리케이션 및 데이터베이스에서 랜덤 I/O는 최대 64K 미만으로 훨씬 작은 블록 크기를 사용합니다. 대규모 블록 I/O는 일반적으로 병렬화되므로 최대 64K 역시 최대 대역폭을 확보하는 데 제한이 없습니다.

일부 워크로드는 최대 64K로 인해 제한이 발생합니다. 특히, 데이터베이스가 적은 수의 입출력을 수행할 수 있는 경우 백업 또는 복구 작업 또는 데이터베이스 전체 테이블 스캔과 같은 단일 스레드 작업이 보다 빠르고 효율적으로 실행됩니다. ONTAP의 최적의 I/O 처리 크기는 256K입니다.

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
Cluster01::*>
```

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 제한되기 때문에 ONTAP 최대 전송 크기를 65536에서 1048576 사이에서 아무 영향 없이 조정할 수 있습니다. 최대 전송 크기를 65536 미만으로 줄이면 가용성 또는 데이터가 손상될 수 있습니다.

ONTAP 레벨에서 전송 크기를 늘리면 다음과 같은 마운트 옵션이 사용됩니다.

```
rw,hard,nointr,bg,vers=[3|4],proto=tcp,rsize=262144,wsize=262144
```

NFSv3 TCP 슬롯 테이블

NFSv3을 Linux와 함께 사용할 경우에는 TCP 슬롯 테이블을 올바르게 설정하는 것이 중요합니다.

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.

산

SAN을 사용하는 PostgreSQL 데이터베이스는 일반적으로 xfs 파일 시스템에서 호스팅되지만

다른 데이터베이스는 OS 공급업체에서 지원하는 경우 사용할 수 있습니다

단일 LUN은 일반적으로 최대 100K IOPS를 지원할 수 있지만, IO 집약적인 데이터베이스는 일반적으로 LVM을 스트라이핑과 함께 사용해야 합니다.

LVM 스트라이핑

플래시 드라이브의 시대 이전에는 스트라이핑이 회전식 드라이브의 성능 제한을 극복하는 데 사용되었습니다. 예를 들어, OS에서 1MB 읽기 작업을 수행해야 하는 경우 1MB가 느리게 전송되기 때문에 단일 드라이브에서 1MB의 데이터를 읽으려면 많은 드라이브 헤드가 필요합니다. 이 1MB의 데이터가 8개의 LUN에 스트라이핑된 경우 운영 체제에서는 8개의 128K의 읽기 작업을 병렬로 실행하고 1MB 전송을 완료하는 데 필요한 시간을 줄일 수 있습니다.

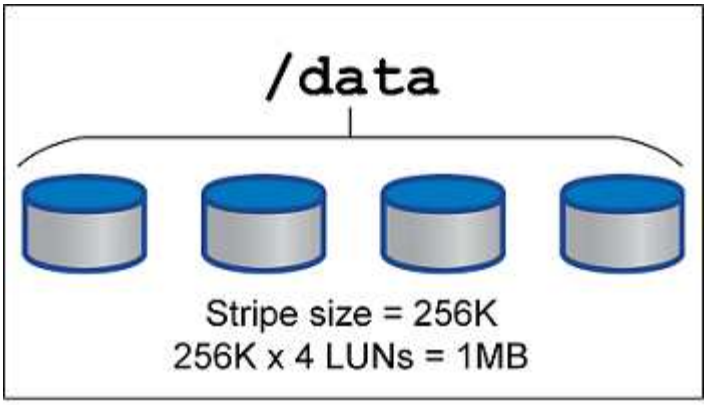
회전식 드라이브를 사용한 스트라이핑은 I/O 패턴을 사전에 알고 있어야 하므로 더 어려웠습니다. 스트라이핑이 실제 I/O 패턴에 맞게 올바르게 조정되지 않은 경우 스트라이핑된 구성이 성능을 저하시킬 수 있습니다. Oracle 데이터베이스, 특히 All-Flash 구성을 사용하면 스트라이핑이 훨씬 쉽게 구성되고 성능이 크게 향상된다는 사실이 입증되었습니다.

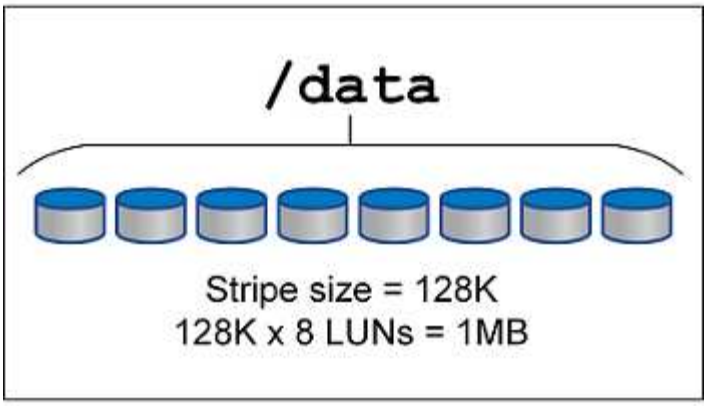
기본적으로 Oracle ASM 스트라이프와 같은 논리적 볼륨 관리자는 있지만 기본 OS LVM은 그렇지 않습니다. 이 중 일부는 여러 LUN을 연결된 장치로 연결하므로 하나의 LUN 디바이스와 하나의 LUN 디바이스에 데이터 파일이 존재합니다. 이로 인해 핫스팟이 발생합니다. 다른 LVM 구현은 기본적으로 분산 익스텐트로 설정됩니다. 이는 스트라이핑과 비슷하지만 더 거칠습니다. 볼륨 그룹의 LUN은 익스텐트라고 하는 큰 조각으로 분할되며 일반적으로 메가바이트 단위로 측정되며 논리적 볼륨은 이러한 익스텐트에 분산됩니다. 그 결과 파일에 대한 랜덤 I/O가 LUN 전체에 분산되어야 하지만 순차적 I/O 작업의 효율성이 최대한 높지는 않습니다.

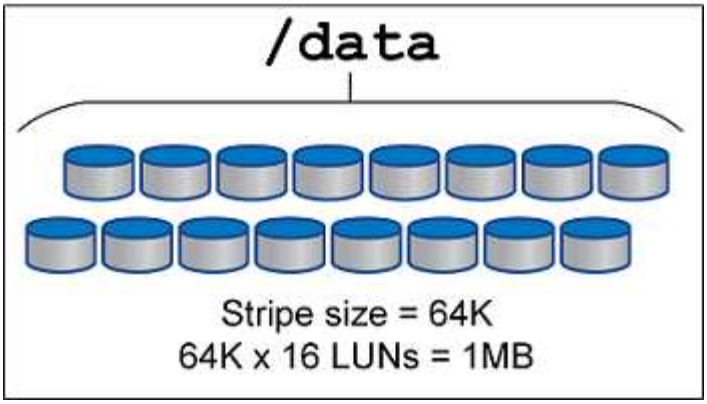
높은 성능을 필요로 하는 애플리케이션 I/O는 거의 항상 (a) 기본 블록 크기 단위 또는 (b) 1MB입니다.

스트라이핑 구성의 기본적인 목표는 단일 파일 I/O를 단일 유닛으로 수행하고, 1MB 크기여야 하는 다중 블록 I/O를 스트라이핑 볼륨의 모든 LUN에 걸쳐 균등하게 병렬 처리할 수 있도록 지원하는 것입니다. 즉, 스트라이프 크기가 데이터베이스 블록 크기보다 작아서는 안 되며 스트라이프 크기를 LUN 수를 곱한 크기가 1MB여야 합니다.

다음 그림에서는 스트라이프 크기 및 폭 조정에 사용할 수 있는 세 가지 옵션을 보여 줍니다. 위에서 설명한 대로 성능 요구 사항을 충족하기 위해 LUN 수를 선택하지만 모든 경우에 단일 스트라이프의 총 데이터는 1MB입니다.







데이터 보호

기본 **Ddta** 보호

스토리지 설계의 주요 측면 중 하나는 PostgreSQL 볼륨을 보호하는 것입니다. 고객은 덤프 방식을 사용하거나 파일 시스템 백업을 사용하여 PostgreSQL 데이터베이스를 보호할 수 있습니다. 이 섹션에서는 개별 데이터베이스 또는 전체 클러스터를 백업하는 다양한 접근 방식에 대해 설명합니다.

PostgreSQL 데이터를 백업하는 방법에는 세 가지가 있습니다.

- SQL Server 덤프
- 파일 시스템 레벨 백업
- 지속적인 아카이빙

SQL Server 덤프 방법의 기본 개념은 SQL Server 명령을 사용하여 파일을 생성하는 것입니다. 이 명령은 서버에 반환될 때 덤프 시점의 데이터베이스를 다시 만들 수 있습니다. PostgreSQL은 유틸리티 프로그램을 제공합니다 `pg_dump` 및 `pg_dump_all` 개별 및 클러스터 수준 백업을 생성하는 데 사용됩니다. 이러한 덤프는 논리적이며 Wal 재생에 사용할 수 있는 충분한 정보가 포함되어 있지 않습니다.

다른 백업 전략은 파일 시스템 수준 백업을 사용하는 것입니다. 이 백업을 사용하면 관리자가 PostgreSQL에서 데이터베이스에 데이터를 저장하는 데 사용하는 파일을 직접 복사할 수 있습니다. 이 방법은 오프라인 모드에서 수행됩니다. 데이터베이스나 클러스터를 종료해야 합니다. 또 다른 대안은 을 사용하는 것입니다 `pg_basebackup` PostgreSQL 데이터베이스의 핫 스트리밍 백업을 실행합니다.

스냅샷 수

PostgreSQL을 사용한 스냅샷 기반 백업에는 전체 또는 시점 복구를 제공하기 위해 데이터 파일, Wal 파일 및 보관된 Wal 파일에 대한 스냅샷 구성이 필요합니다.

PostgreSQL 데이터베이스의 경우 스냅샷을 사용하는 평균 백업 시간은 몇 초에서 몇 분 사이입니다. 이 백업 속도는 보다 60 ~ 100배 빠릅니다 `pg_basebackup` 다양한 파일 시스템 기반 백업 방법을 알아봅니다.

NetApp 스토리지의 스냅샷은 충돌 시에도 정합성이 보장되는 스냅샷과 애플리케이션 정합성을 유지할 수 있습니다. 장애 발생 시 정합성이 보장되는 스냅샷은 데이터베이스를 중지하지 않고 스토리지에 생성되지만 애플리케이션 정합성이 보장되는 스냅샷은 데이터베이스가 백업 모드에 있는 동안 생성됩니다. 또한 NetApp는 후속 스냅샷이 증분식 영구 백업되도록 하여 스토리지 절감 및 네트워크 효율성을 높입니다.

스냅샷은 빠르고 시스템 성능에 영향을 미치지 않으므로 다른 스트리밍 백업 기술과 마찬가지로 일일 백업을 하나씩 생성하는 대신 매일 여러 스냅샷을 예약할 수 있습니다. 복원 및 복구 작업이 필요한 경우 시스템 가동 중지 시간은 다음과 같은 두 가지 주요 기능으로 줄어듭니다.

- NetApp SnapRestore 데이터 복구 기술은 복원 작업이 몇 초 안에 실행된다는 것을 의미합니다.
- 공격적 RPO(복구 시점 목표)는 적용해야 하는 데이터베이스 로그 수가 적고 전달 복구도 가속화된다는 것을 의미합니다.

PostgreSQL을 백업하려면 (consistency-group) Wal 및 보관된 로그를 사용하여 데이터 볼륨을 동시에 보호해야 합니다. Snapshot 기술을 사용하여 Wal 파일을 복사하는 동안 를 실행해야 합니다 `pg_stop` 보관해야 하는 모든 Wal 항목을 플러시합니다. 복구 중에 Wal 항목을 플러시하는 경우 데이터베이스를 중지하거나, 기존 데이터 디렉토리를 마운트 해제 또는 삭제하고, 스토리지에서 SnapRestore 작업만 수행하면 됩니다. 복구가 완료된 후 시스템을 마운트하여 현재 상태로 되돌릴 수 있습니다. 시점 복구의 경우 Wal 및 아카이브 로그를 복원할 수도 있습니다. 그러면 PostgreSQL이 가장 일관된 지점을 결정하고 자동으로 복구합니다.

정합성 보장 그룹은 ONTAP의 기능이며 단일 인스턴스 또는 여러 테이블스페이스가 있는 데이터베이스에 여러 개의 볼륨이 마운트된 경우 권장됩니다. 정합성 보장 그룹 스냅샷은 모든 볼륨이 함께 그룹화되고 보호되도록 합니다. 일관성 그룹은 ONTAP System Manager에서 효율적으로 관리할 수 있으며 일관성 그룹을 클론 복제하여 테스트 또는 개발 목적으로 데이터베이스의 인스턴스 복사본을 생성할 수도 있습니다.

데이터 보호 소프트웨어

Snapshot 및 NetApp FlexClone 기술과 함께 PostgreSQL 데이터베이스용 NetApp SnapCenter 플러그인을 사용하면 다음과 같은 이점을 얻을 수 있습니다.

- 신속한 백업 및 복원:
- 공간 효율적인 클론 복제:

- 빠르고 효율적인 재해 복구 시스템을 구축할 수 있는 능력.

다음과 같은 상황에서 Veeam Software 및 Commvault와 같은 NetApp 프리미엄 백업 파트너를 선호할 수 있습니다.

- 이기종 환경에서 워크로드 관리
- 장기간 보존을 위해 클라우드 또는 테이프에 백업 저장
- 다양한 OS 버전 및 유형 지원

PostgreSQL용 SnapCenter 플러그인은 커뮤니티 지원 플러그인이며 설치 및 설명서는 NetApp 자동화 스토어에서 사용할 수 있습니다. 사용자는 SnapCenter를 통해 원격으로 데이터베이스를 백업하고, 데이터를 클론 복제하고, 복원할 수 있습니다.