8.1 최적화 개요

데이터베이스의 성능은 테이블, 쿼리, 환경 설정 등 데이터베이스 레벨의 여러 요소에 따라 달라집니다. 이러한 소프트웨어 구조가 하드웨어 레벨의 CPU, I/O 작업으로 이어지며, 이를 최소화하고 최대한 효율적으로 만들어야 합니다. 데이터베이스 성능과 관련된 일을 함에 있어, 소프트웨어 측면에서 높은 수준의 규칙과 가이드라인을 배우고 벽걸이 시계 같은 것을 사용하여 성능을 측정하는 것부터 시작하게 될 것입니다. 전문가 수준이 되고 나면, 내부에서 일어나는 일에 대해 더 많이 배우고, CPU 사이클, I/O 작업 등으로 성능을 측정하기 시작할 것입니다.

일반 사용자는 기존의 소프트웨어 및 하드웨어 구성에서 최고의 데이터베이스 성능을 얻는 것을 목표로 합니다. 고급 사용자는 MySQL 소프트웨어 자체를 개선할 수 있는 기회를 찾거나, 자체 스토리지 엔진과 하드웨어 어플라이언스를 개발하여 MySQL 생태계를 확장할 수 있습니다.

데이터베이스 레벨에서의 최적화

데이터베이스 애플리케이션의 속도를 높이는 데 있어 가장 중요한 요소는 기본 설계입니다:

- 테이블이 제대로 구축되어 있습니까? 특히 컬럼에 적절한 데이터 타입을 사용했고, 각 테이블에는 작업 유형에 적합한 컬럼을 사용했습니까? 예를 들어, 빈번한 업데이트를 수행하는 애플리케이션은 대체로 소수의 컬럼을 가진 다수의 테이블을 사용하고, 대량의 데이터를 분석하는 애플리케이션은 대체로 소수의 컬럼을 가진 다수의 테이블을 사용합니다.
- 쿼리를 효율적으로 수행하기 위해 적절한인덱스가 설정되어 있습니까?
- 테이블마다 적절한 스토리지 엔진을 사용하고 있으며, 각 스토리지 엔진의 장점과 기능을 잘 활용하고 있습니까? 특히InnoDB와 같은 트랜잭션 스토리지 엔진 또는 MyISAM과 같은 비트랜잭션 스토리지 엔진의 선택은 성능과 확장성에 매우 중요할 수 있습니다.

> **참고사항**
> **InnoDB는 신규 테이블의 기본 스토리지 엔진입니다. 실제, InnoDB의 고급 성능 기능으로 인해 단순한 MyISAM 테이블보다 나은 성능을 발휘하는 경우가 많으며, 특히 사용량이 많은 데이터베이스에서 뛰어납니다.**

- 각 테이블은 적절한 행 서식을 사용하고 있습니까? 이 선택은 테이블에 사용되는 스토리지 엔진에 따라 달라집니다. 특히 압축된 테이블은 디스크 공간을 적게 사용하므로 데이터 읽기 및 쓰기에 필요한 디스크 I/O가 줄어듭니다. 압축은 InnoDB 테이블을 사용하는 모든 종류의 워크로드, MyISAM 테이블의 읽기 전용 워크로드에 사용할 수 있습니다.
- 애플리케이션에서 적절한잠금 전략을 사용하고 있습니까? 예를 들어, 데이터베이스 작업이 동시에 수행될 수 있도록 가능한 한 공유 액세스를 허용하거나, 중요한 작업이 최우선 순위를 차지할 수 있도록 배타적 액세스를 요청합니다. 여기서도 스토리지 엔진의 선택이 중요합니다. InnoDB 스토리지 엔진은 사용자가 관여하지 않고도 대부분의 잠금 문제를 처리하며, 덕분에 데이터베이스의 동시성을 향상시키고 코드에 대한 실험과 튜닝의 양을 줄일 수 있습니다.
- 캐싱에 사용되는 모든 메모리 영역의 크기가 올바르게 설정되어 있습니까? 자주 액세스하는 데이터를 보관할 수 있을 만큼 충분히 커야 하지만, 물리적 메모리를 과부하시켜 페이징을 유발할 정도로 크지는 않아야 합니다. 구성해야 할 주요 메모리 영역은 InnoDB 버퍼 풀과 MyISAM 키 캐시입니다.

하드웨어 레벨에서의 최적화

데이터베이스가 바빠지면 바쁠수록 모든 데이터베이스 애플리케이션은 결국 하드웨어의 한계에 도달하게 됩니다. DBA는 애플리케이션을 조정하거나 서버를 재구성하여 이러한 병목 현상을 피할 수 있는지 또는 추가 하드웨어 리소스가 필요한지 여부를 평가해야 합니다. 시스템 병목 현상은 일반적으로 다음과 같은 원인으로 인해 발생합니다:

- 디스크 탐색. 디스크가 데이터를 검색하는 데 시간이 걸립니다. 최신 디스크의 경우, 이 평균 시간은 일반적으로 10ms 미만이므로 이론적으로 1초당 약 100회 검색을 실행할 수 있습니다. 이 시간은 신형 디스크에서 서서히 개선되고 있지만, 하나의 테이블에 대해 최적화하는 것은 매우 어렵습니다. 탐색 시간을 최적화하는 방법은 데이터를 여러 개의 디스크에 분산시키는 것입니다.
- 디스크 읽기 및 쓰기. 디스크가 올바른 위치에 있다면, 이제 데이터를 읽거나 써야 합니다. 최신 디스크는 하나의 디스크에서 최소 10~20 MB/s의 처리량을 제공합니다. 여러 디스크에서 병렬로 읽을 수 있기 때문에 디스크 탐색보다는 더 쉽게 최적화할 수 있습니다.
- CPU 사이클. 데이터가 메인 메모리에 있는 경우, 결과를 얻기 위해 데이터를 처리해야 합니다. 메모리 용량에 비해 더 큰 테이블을 사용하는 것이 가장 일반적인 제한 요인이 됩니다. 그러나 작은 테이블의 경우 일반적으로 속도는 문제가 되지 않습니다.
- 메모리 대역폭. CPU가 CPU 캐시에 담을 수 있는 것보다 더 많은 데이터를 필요로 하는 경우, 메인 메모리 대역폭이 병목 지점이 될 수 있습니다. 이는 대부분의 시스템에서 흔치 않은 병목 현상이지만, 알아두어야 할 사항입니다.

이식성과 성능 사이의 균형화

이식 가능한 MySQL 프로그램에서 성능 지향적인 SQL 확장 기능을 사용하려면, SQL문 내에 MySQL 전용의 키워드들을 /*! */ 주석 구분 기호로 감쌉니다. 다른 SQL 서버는 주석으로 처리된 키워드를 무시합니다. 주석 작성에 대한 자세한 내용은 9.7절 '주석'을 참고합니다.

Revision #5
★Created 8 September 2023 05:41:58 by 신민항
✎Updated 9 September 2023 17:26:07 by 신민항