LAMP 시스템 조율, Part 3: MySQL 조율

서버 조율 팁으로 MySQL 서버에 날개를 달자

난이도 : 중급

Sean A. Walberg, 선임 네트워크 엔지니어

옮긴이: 박재호 이해영 dwkorea@kr.ibm.com

2008 년 5 월 06 일

LAMP(Linux®, Apache, MySQL, PHP/Perl) 아키텍처를 활용하는 응용 프로그램은 끊임없이 개발되고 배포되고 있습니다. 하지만 때로 서버 관리자는 다른 사람이 작성했다는 이유만으로 응용 프로그램 자체에 대한 통제권이 거의 없습니다. 기사 셋으로 이뤄진 이번 연재물은 응용 프로그램 성능을 향상시킬 서버 환경 설정 항목을 다룹니다. 연재 마지막인 세 번째 기사에서는 최대 성능을 발휘하도록 데이터베이스 계층을 조율하는 데 초점을 맞춥니다.

MySQL 조율에 대해

MySQL 서버를 빠르게 하기 위한 방법은 세 가지가 있는데, 효율이 낮은 쪽에서 높아지는 쪽으로 나열하면 다음과 같다.

1. 하드웨어로 문제를 푼다.
2. MySQL 프로세스 설정을 조율한다.
3. 질의를 최적화한다.

하드웨어로 문제를 푸는 방법이 가장 먼저 떠오른다. 특히 데이터베이스가 자원을 잡아먹는 괴물이라는 사실을 감안하면 말이다. 하지만 이 해법에는 한계가 있다. 현실을 고려할 때 CPU나 디스크 속력은 두 배로, 메모리 용량은 네 배에서 여덟 배 정도만 늘일 수 있다.

DB2®로 이주
MySQL에서 IBM® DB2®로 이주하는 명쾌하고 비용이 들지 않는 방법을 찾고 싶은가? "MySQL 또는 PostgreSQL에서 DB2 Express-C로의 마이그레이션 (영문)" 기사에서는 이주 도구를 활용해 쉽게 이전하는 방법을 보여준다. 공짜 DB2 Express-C를 내려받아 지금 바로 시도해보자.

두 번째로 좋은 방법은 `mysqld`라는 MySQL 서버 조율이다. 이 프로세스 조율은 올바른 위치에 메모리를 할당하고 어떤 부하가 걸릴지 `mysqld`에 알려주는 조정 기법을 의미한다. 디스크 속력을 좀 더 빠르게 만드는 대신, 필요한 디스크 접근 횟수를 줄이는 편이 유리하다. 비슷하게, MySQL 프로세스가 올바르게 동작하도록 만드는 조율은 개발자가 임시 디스크 테이블과 파일 여닫기 같은 배경 작업에 신경을 쓰는 대신 질의에 대한 서비스에 좀 더 많은 시간을 보낼 수 있음을 의미한다. `mysqld` 조율은 이번 기사에서 주로 다룰 내용이다.

최고로 좋은 방법은 질의 최적화다. 이는 적절한 색인을 테이블에 만들어 놓고, MySQL의 장점을 최대로 활용하는 방향으로 질의를 작성하는 조율 기법을 의미한다. 이번 기사에서 질의 조율을 다루지는 않지만(이 주제로 책을 써도 되겠다), `mysqld` 환경 설정을 변경해 조율이 필요한 질의를 보고하도록 만든다.

중요한 조율 순서를 제시하긴 했지만, 그렇다고 해서 적절히 조율을 마친 질의를 위해 하드웨어나 `mysqld` 설정을 무시하라는 말은 아니다. 느린 기계는 느린 기계일 뿐이며,

제대로 작성한 질의를 돌리더라도 부하가 걸려 실패하는 경우를 목격했는데, mysqld가 질의를 서비스하는 대신 바쁘게 움직이느라 시간을 소모하고 있었기 때문이었다.

느린 질의 기록

SQL 서버에서 자료 테이블은 디스크에 위치한다. 색인은 전체 테이블을 찾지 않고서 서버가 테이블에서 자료 열을 찾아내도록 도와준다. 전체 테이블을 찾을 때 테이블 탐색을 수행한다고 부른다. 종종 테이블에서 일부만 원하는 경우가 있는데, 전체 테이블 탐색은 디스크 I/O와 시간을 상당히 많이 소비한다. 이런 문제는 테이블 조인 과정에서 복합적으로 나타나는데, 양쪽 테이블에 들어있는 열을 하나씩 비교해야 하기 때문이다.

물론 테이블 탐색이 항상 두통거리만은 아니다. 종종 전체 테이블을 읽는 경우가 일부만 읽는 경우보다 더 효과적인 경우도 있다(이런 결정을 내리려면 질의 계획이라는 작업을 거쳐야 한다). 색인을 비효율적으로 사용하거나 전혀 색인을 사용하지 않으면 질의가 느려지며, 테이블 크기가 증가하면서 서버에 부하가 걸리면 이런 문제점은 더욱 두드러진다. 실행을 위해 주어진 시간보다 더 오래 걸리는 질의는 느린 질의라고 부른다.

mysqld 환경 설정에서 느린 질의 로그라고 적절히 이름이 붙은 느린 질의 기록을 활성화할 수 있다. 관리자는 이 로그 파일을 살펴 응용 프로그램에서 어느 곳을 추가로 조사할지 결정한다. Listing 1은 느린 질의 로그를 활성화하기 위해 my.cnf에 필요한 환경 설정을 보여준다.

Listing 1. MySQL 느린 질의 로그 활성

```
[mysqld]
; 느린 질의 로그를 활성화한다. 기본은 10초다.
log-slow-queries
; 5초 이상 걸리는 질의를 기록한다.
long_query_time = 5
; long_query_time보다 적게 걸릴 경우 색인을 사용하지 않는 질의를 기록한다.
; MySQL 4.1 이상 버전에만 통한다
log-queries-not-using-indexes
```

이와 같은 세 가지 설정을 함께 사용하면, 5초 이상 지속되는 질의나 색인을 사용하지 않는 질의를 기록한다. log-queries-not-using-indexes에 대한 경고가 하나 있다. 반드시 MySQL 4.1 이상 버전을 사용해야만 한다. 느린 질의 로그는 MySQL 자료 디렉터리에 들어 있으며, 파일 형식은 hostname-slow.log이다. 다른 이름이나 경로를 사용한다면, my.cnf에서 log-slow-queries = /new/path/to/file을 지정하자.

느린 질의 로그를 읽으려면 mysqldumpslow 명령을 내린다. 로그 파일 경로를 지정하는 방법으로 느린 질의를 발견 순서에 따라 정렬한 목록을 얻는다. 도움을 주는 기능 한 가지는 mysqldumpslow가 결과를 비교하기 앞서 사용자 정의 자료를 제거하므로 동일한 질의로 여러 번 수행해도 하나로 센다. 이는 대다수 작업에 필요한 질의를 찾아내는 데 도움을 준다.

질의 캐시

대다수 LAMP 응용 프로그램은 데이터베이스에 상당히 의존하며 동일한 질의를 여러 번 반복한다. 질의를 만들 때마다 데이터베이스는 똑같은 작업을 해야만 한다. 즉 질의를 해석해, 실행 방법을 결정하고, 디스크에서 정보를 메모리에 올리고, 클라이언트에 이를 반환한다. MySQL은 질의 캐시라는 기능을 사용해서 메모리에 질의 결과를 저장하며 필요할 때 찾아쓴다. 여러 인스턴스에서 이런 캐시는 극적으로 성능을 높힌다. 하지만 질의 캐시는 기본적으로 비활성화되어 있다는 사실을 염두에 두자.

`query_cache_size = 32M`를 /etc/my.conf에 추가하면 질의 캐시로 32MB를 잡는다.

질의 캐시 감시

질의 캐시를 활성화한 다음에, 효율적으로 사용하고 있는지 이해하는 과정이 중요하다. MySQL은 여러 변수를 사용해서 캐시에서 어떤 일이 벌어지는지 감시하도록 만든다. Listing 2는 캐시 상태를 보여준다.

Listing 2. 질의 캐시 통계 출력

```
mysql> SHOW STATUS LIKE 'qcache%';
+-------------------------+------------+
| Variable_name           | Value      |
+-------------------------+------------+
| Qcache_free_blocks      | 5216       |
| Qcache_free_memory      | 14640664   |
| Qcache_hits             | 2581646882 |
| Qcache_inserts          | 360210964  |
| Qcache_lowmem_prunes    | 281680433  |
| Qcache_not_cached       | 79740667   |
| Qcache_queries_in_cache | 16927      |
| Qcache_total_blocks     | 47042      |
+-------------------------+------------+
8 rows in set (0.00 sec)
```

각 항목을 분리하면 표 1과 같다.

표 1. MySQL 질의 캐시 변수

변수 이름	설명
`Qcache_free_blocks`	캐시에 있는 연속적인 메모리 블록 숫자. 높은 숫자는 단편화가 일어난 징표다. `FLUSH QUERY CACHE`는 캐시 조각을 모아 자유 블록 하나로 만든다.
`Qcache_free_memory`	캐시에 있는 자유 메모리
`Qcache_hits`	캐시에서 질의를 가져올 때마다 값이 증가한다.

Qcache_inserts	질의가 들어올 때마다 증가한다. inserts를 hits로 나누면 비적중률을, 1에서 비적중률을 빼면 적중률을 구할 수 있다. 직전 에제에서 대략 질의 중 87%를 캐시에서 가져왔다.
Qcache_lowmem_prunes	캐시를 위한 메모리가 부족해져 더 많은 질의를 위한 공간을 확보하기 위해 정리되어야 하는 횟수. 이 숫자를 계속해서 살펴보는데, 증가 추세에 있다면 단편화가 심각하거나 메모리가 부족하다는 징표다(위에서 언급한 free_blocks와 free_memory를 살펴본다).
Qcache_not_cached	일반적으로 SELECT 구문이 아니기 때문에 캐시 후보에서 제외된 질의 숫자
Qcache_queries_in_cache	현재 캐시되어 있는 질의 숫자(응답 숫자 포함)
Qcache_total_blocks	캐시에 있는 블록 숫자

종종 이런 값의 변화 추이를 살펴보면 캐시를 효율적으로 사용하는지 파악하는 데 도움을 준다. FLUSH STATUS는 몇몇 카운터를 초기화하므로 서버가 동작 중에 있을 경우 도움이 된다.

모든 내용을 캐시하도록 과도하게 큰 캐시를 잡고 싶은 유혹이 든다. mysqld는 메모리 부족으로 인한 정리 작업과 같은 캐시 관리 작업도 해야 하므로, 여기에만 신경을 쓸 경우 서버가 꼼짝달싹하지 못한다. 일반적인 규칙을 설명하자면 FLUSH QUERY CACHE가 오래 걸린다면 캐시가 너무 큰 상황이다.

제약 가하기

시스템 부하가 자원 부족으로 이어지지 않도록 mysqld에 몇 가지 제약을 가해야 한다. Listing 3은 my.cnf에서 몇 가지 중요한 자원 관련 설정을 보여준다.

Listing 3. MySQL 자원 설정

```
set-variable=max_connections=500
set-variable=wait_timeout=10
max_connect_errors = 100
```

최대 접속은 첫째 행에서 다룬다. 아파치가 사용하는 MaxClients와 같이, 서비스가 가능한 접속 수만 허용한다. 지금까지 서버가 처리한 최대 접속 수를 확인하려면 SHOW STATUS LIKE 'max_used_connections' 명령을 내린다.

둘째 행은 mysqld가 10초 이상 쉬고 있는 접속을 끊어버리도록 만든다. LAMP 응용 프로그램에서 데이터베이스 접속은 일반적으로 웹 서버가 요청을 처리하는 동안에만

이뤄진다. 종종 부하가 걸린 상태에서 연결이 일시 정지된 상황에서 접속 테이블 공간을 차지하는 경우가 있다. 활성 사용자가 많거나 데이터베이스에 영속적인 접속이 이뤄지고 있다면, 이 값을 낮춰잡는 정책은 바람직하지 않다.

마지막 행은 안전 벨트다. 호스트에 서버 접속 관련 문제가 생겨 너무 많이 요청을 취소한다면, `FLUSH HOSTS`를 수행할 때까지 호스트는 잠겨버린다. 기본적으로 열 번 정도 실패하면 잠겨버리도록 설명한다. 이 값을 100으로 바꾸면 문제가 무엇이든 복구할 시간을 서버에 충분히 준다. 더 높은 값으로 설정하더라도 그다지 도움을 주지 않는 이유는 서버가 한 번에 100번 연결해도 실패한다면, 이후 계속 시도하더라도 연결에 성공할 가능성이 희박하기 때문이다.

버퍼와 캐시

MySQL은 100개가 넘는 조율 설정값을 지원한다. 하지만 천만다행으로 이 중에서 몇 가지만 알면 충분하다. 설정을 올바르게 하려면, `SHOW STATUS` 명령을 통해 상태 변수를 살펴보고, 이를 통해 `mysqld`가 원하는 방식으로 움직이는지 파악한다. 시스템에 존재하는 메모리 자원을 넘어서 버퍼와 캐시를 할당할 수 없기에 종종 조율 과정에서 타협이 필요하다.

MySQL 조율값은 `mysqld` 프로세스 전체나 개별 클라이언트 세션에 대해 설정이 가능하다.

서버 단위 설정

각 테이블은 디스크에 파일 형태로 저장되며, 테이블을 읽기 위해서는 파일을 열어야 한다. 파일 읽기 과정을 빠르게 하기 위해, `mysqld`는 /etc/mysqld.conf에 지정된 숫자(`table_cache`)만큼 열린 파일을 캐시한다. Listing 4는 열린 테이블에 대한 활동 상황을 출력하는 방법을 보여준다.

Listing 4. 열린 테이블 활동 상황 출력

```
mysql> SHOW STATUS LIKE 'open%tables';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| Open_tables   | 5000  |
| Opened_tables | 195   |
+---------------+-------+
2 rows in set (0.00 sec)
```

Listing 4는 현재 테이블 5000개가 열려있으며, 테이블 195개가 열려야 했음을 보여준다. 캐시에 유효한 파일 기술자가 없기 때문에 이런 현상이 일어난다(직전에 통계를 초기화했으므로 5000개 열린 테이블 중에 단지 195개만 열렸다고 기록이 남는다). `SHOW STATUS` 명령을 다시 실행할 때 `Opened_tables`가 급격하게 올라가면, 캐시 적중률이 떨어진 상황이다. `table_cache` 설정값보다 `Open_tables` 설정값이 훨씬 낮으면, 캐시를 너무 크게 잡았다(물론 여유있게 설정하는 방식은 나쁘지 않다). 예를 들어, `table_cache = 5000`으로 테이블 캐시 값을 조정한다.

테이블 캐시와 마찬가지로 스레드를 위한 캐시도 있다. mysqld는 접속을 받을 때 필요한 스레드를 만든다. 바쁜 서버에서 접속이 빠르게 연결되었다 끊어지면, 초기 접속 속력을 높아기 위해 나중에 사용할 요량으로 스레드를 캐시한다.

Listing 5는 충분한 스레드가 캐시되었는지 살펴보는 방법을 보여준다.

Listing 5. 스레드 사용량 통계 보기

```
mysql> SHOW STATUS LIKE 'threads%';
+-------------------+--------+
| Variable_name     | Value  |
+-------------------+--------+
| Threads_cached    | 27     |
| Threads_connected | 15     |
| Threads_created   | 838610 |
| Threads_running   | 3      |
+-------------------+--------+
4 rows in set (0.00 sec)
```

여기서 가장 중요한 값은 Threads_created로 mysqld가 새로운 스레드를 생성할 때마다 하나씩 증가한다. 연속적으로 SHOW STATUS 명령을 내릴 때, 이 숫자가 급격하게 올라가면 스레드 캐시 수치를 높여야 한다. 예를 들어, my.cnf에서 thread_cache = 40을 설정하면 된다.

키 버퍼는 MyISAM 테이블을 위한 색인 블록을 저장한다. 이상적으로 이런 블록에 대한 요청은 디스크가 아니라 메모리에서 일어나야 한다. Listing 6은 메모리와 디스크에서 얼마나 많은 블록을 읽는지 확인하는 방법을 보여준다.

Listing 6. 키 효율성 확인

```
mysql> show status like '%key_read%';
+-------------------+-----------+
| Variable_name     | Value     |
+-------------------+-----------+
| Key_read_requests | 163554268 |
| Key_reads         | 98247     |
+-------------------+-----------+
2 rows in set (0.00 sec)
```

Key_reads는 디스크에서 요청한 숫자이며, Key_read_requests는 전체 숫자다. Key_reads를 Key_read_requests로 나누면 비적중률이 나온다. Listing 6을 보면 1000개 요청 당 0.6개가 적중하지 않았다. 1000개 요청 당 1개 이상 적중하지 않는다면 키 버퍼를 늘려야 한다. 예를 들어, key_buffer = 384M를 지정하면 버퍼를 384MB로 늘인다.

임시 테이블은 GROUP BY 절과 같이 추가 처리가 필요할 때 임시로 자료를 저장할 곳으로, 좀 더 고급 질의에서 사용된다. 이상적으로 이런 테이블은 메모리에 생성하지만, 임시 테이블이 너무 커질 경우 디스크에 써야 한다. Listing 7은 임시 테이블 생성과 관

련한 통계를 보여준다.

Listing 7. 임시 테이블 사용량 보기

```
mysql> SHOW STATUS LIKE 'created_tmp%';
+-------------------------+-------+
| Variable_name           | Value |
+-------------------------+-------+
| Created_tmp_disk_tables | 30660 |
| Created_tmp_files       | 2     |
| Created_tmp_tables      | 32912 |
+-------------------------+-------+
3 rows in set (0.00 sec)
```

임시 테이블을 사용하면 `Created_tmp_tables`가 증가한다. 디스크 기반 테이블을 사용하면 `Created_tmp_disk_tables`가 증가한다. 이 비율을 정확하고 빠르게 결정하지 못하는 이유는 질의에 의존하기 때문이다. 시차를 두고 `Created_tmp_disk_tables`를 관찰하면 생성된 디스크 테이블 비율을 알 수 있고, 설정 값이 유효한지 살펴볼 수 있다. `tmp_table_size`와 `max_heap_table_size` 둘 다 임시 테이블 최대 크기를 제어하므로, my.cnf에서 양쪽 설정을 모두 확인해야 한다.

세션 단위 설정

이어지는 설정은 세션 단위다. 이 값을 설정할 때 신경을 써야 하는 이유는 잠재적인 접속 숫자에 설정값이 곱해지므로 메모리 사용량이 늘어나기 때문이다. 코드에서 해당 세션 값을 변경하거나 my.cnf에서 모든 세션 값을 변경할 수 있다.

MySQL이 정렬 작업을 수행할 때, 디스크에서 읽는 열을 저장하기 위한 정렬 버퍼를 할당한다. 정렬할 자료 크기가 너무 크다면, 디스크에 임시 파일로 자료를 저장하고, 다시 한번 정렬해야 한다. `sort_merge_passes` 상태값이 높으면, 디스크 활동량이 많다는 증거다. Listing 8은 정렬 관련 상태 카운터 몇 가지를 보여준다.

Listing 8. 정렬 통계 보기

```
mysql> SHOW STATUS LIKE "sort%";
+-------------------+---------+
| Variable_name     | Value   |
+-------------------+---------+
| Sort_merge_passes | 1       |
| Sort_range        | 79192   |
| Sort_rows         | 2066532 |
| Sort_scan         | 44006   |
+-------------------+---------+
4 rows in set (0.00 sec)
```

`sort_merge_passes`가 높다면, `sort_buffer_size` 쪽에 관심을 기울여야 한다. 예를 들어, `sort_buffer_size = 4M`를 지정하면, 정렬 버퍼를 4MB로 늘인다.

MySQL은 또한 테이블을 읽기 위한 메모리를 할당한다. 이상적으로 보면 색인은 필요

한 열에서만 읽도록 충분한 정보를 제공하지만, (자료 특성 때문이나 설계 잘못으로 인해) 읽어야 할 테이블이 많은 질의도 있기 마련이다. 이런 행동 양식을 이해하려면, (색인으로 직접 접근하는 대신) 테이블에서 다음 열을 직접 읽어야 하는 숫자와 `SELECT` 문 개수를 알아야 한다. 이렇게 하려면 Listing 9에서 소개하는 명령을 내린다.

Listing 9. 테이블 탐색 비율 확인

```
mysql> SHOW STATUS LIKE "com_select";
+---------------+--------+
| Variable_name | Value  |
+---------------+--------+
| Com_select    | 318243 |
+---------------+--------+
1 row in set (0.00 sec)

mysql> SHOW STATUS LIKE "handler_read_rnd_next";
+-----------------------+-----------+
| Variable_name         | Value     |
+-----------------------+-----------+
| Handler_read_rnd_next | 165959471 |
+-----------------------+-----------+
1 row in set (0.00 sec)
```

`Handler_read_rnd_next` / `Com_select`는 테이블 탐색 비율을 보여주는데, Listing 9에서는 521:1이다. 4000이 넘어가면, `read_buffer_size = 4M`와 같이 `read_buffer_size` 값이 충분히 크게 설정되어 있는지 확인한다. 이 값이 8M 이상으로 커진다면, 개발자에게 질의 조율이 필요하다고 알려주자!

세 가지 필수 도구

구체적인 설정값을 파고 들 때, `SHOW STATUS` 명령이 유용하긴 하지만, `mysqld`에서 제공하는 방대한 자료를 해석하는 과정에 도움을 주는 몇 가지 도구가 필요하다. 세 가지 필수 도구를 찾아내었는데, 참고자료 절에 정리해놓았다.

대다수 시스템 관리자는 `top` 명령에 익숙하다. `top`은 태스크가 소비하는 CPU와 메모리를 주기적으로 갱신하면서 보여준다. `mytop`은 `top`을 모델로 만든 프로그램으로 현재 동작 중인 질의와 연결된 모든 클라이언트를 보여준다. `mytop`은 또한 키 버퍼와 질의 캐시 효율성에 대한 실시간 및 과거 자료를 제공하며, 실행 중인 질의 통계도 보여준다. 상황 파악에 유용한 도구이며, 10초 내로 서버 상태와 문제를 초래하는 모든 접속을 표시할 수 있다.

`mysqlard`는 MySQL 서버에 접속하는 데몬으로, 5초마다 자료를 수집해 라운드 로빈으로 동작하는 데이터베이스 뒷단에 저장한다. 웹 페이지는 테이블 캐시 사용량, 키 효율성, 접속된 클라이언트, 임시 테이블 사용량과 같은 자료를 출력한다. `mytop`은 서버 상태를 스냅 사진으로 찍어주며, `mysqlard`는 장기간에 걸친 서버 상태를 보여준다. 보너스로, `mysqlard`는 수집한 몇몇 정보를 활용해 서버 조율 방법을 제안한다.

SHOW STATUS 정보를 수집하는 또 다른 도구는 mysqlreport다. 이 도구가 mysqlard가 보고하는 내용보다 훨씬 자세한 보고 내역을 제공하는 이유는 서버의 모든 측면을 분석하기 때문이다. mysqlreport가 서버 조율에 뛰어난 이유는 상태 변수를 적절히 계산해 수정이 필요한 내용을 알려주기 때문이다.

요약

MySQL 조율 기본기를 설명하는 이 기사로 LAMP 컴포넌트 조율을 다루는 연재물을 마무리한다. 조율은 대부분 동작 원리를 이해하고 적절하게 동작하는지 확인하고 조정하고, 다시 평가하는 작업이다. 리눅스, 아파치, PHP, MySQL로 대표되는 각 컴포넌트마다 각자 요구 사항이 존재한다. 개별적으로 컴포넌트를 이해하고 있으면, 응용 프로그램을 느리게 만드는 병목을 제거하는 데 도움이 된다.

참고자료

교육

- "MySQL 또는 PostgreSQL에서 DB2 Express-C로의 마이그레이션 (영문)"(developerWorks, 2006년 6월)은 MySQL에서 DB2 Express-C로 이주하는 손쉬운 방법을 설명한다.
- IBM은 또한 DB2 Express-C로 이주하려는 MySQL 관리자를 돕기 위해 ""Leveraging MySQL skills to learn DB2 Express"(developerWorks, 2006년 2월)과 기타 연재물을 제공한다.
- "Using MySQL in a federated database environment"(developerWorks, 2004년 12월)은 WebSphere®에서 MySQL 데이터베이스에 들어있는 자료를 접근하는 방법을 설명하는 튜토리얼이다.
- SHOW VARIABLES와 SHOW STATUS는 MySQL 문서에 잘 정의되어 있다.
- 블로그를 좋아한다면 MySQL 성능 블로그, Xaprb, MySQL DBA가 읽을 만하다.
- developerWorks 아키텍처 영역을 보면 아키텍처 부문에서 기술을 발전시키도록 도와주는 참고자료가 나온다. 적절한 아키텍처 개발이 LAMP 응용 프로그램 확장성을 높이는 열쇠다.
- developerWorks에서 리눅스 영역을 보면, 리눅스 튜토리얼과 지난 달에 가장 인기 있었던 기사와 튜토리얼을 포함한 리눅스 개발자용 참고 자료가 나온다.
- developerWorks 기술 행사와 웹 캐스트를 놓치지 말기 바란다.

제품 및 기술 얻기

- 지금부터 3년 전에 나왔음에도 불구하고 High Performance MySQL는 여전히 가치 있는 책이다. 저자는 또한 MySQL에 대한 다양한 기사를 제공하는 웹 사이트를 운영한다.

- mytop은 정확하게 그 순간 MySQL 서버에서 어떤 일이 일어나는지를 말해주며, 몇몇 핵심 통계 자료를 제공한다. 데이터베이스 문제가 발생했을 때 처음 사용하는 프로그램이다.

- mysqlard는 MySQL 서버에서 성능 지표를 그래프로 보여주며, 조율에 대한 조언도 한다.

- mysqlreport는 필수 도구다. 이 도구는 여러분을 대신해 `SHOW STATUS` 값을 분석한다.

- phpMyAdmin에 대한 링크 없이는 MySQL 기사가 끝나지 않는다. 상태 변수에 대한 몇 가지 해석과 더불어 관리를 쉽게 만드는 기능을 제공한다.

- DB2®, Lotus®, Rational®, Tivoli®, WebSphere®와 같은 최신 IBM 평가판 소프트웨어를 포함하는 두 장짜리 DVD 세트인 SEK for Linux를 주문하자.

- IBM 평가판 소프트웨어: developerWorks에서 직접 내려 받아 다음번 리눅스 프로젝트에 활용하자.

토론

- 블로그, 포럼, 포드캐스트, 새로운 developerWorks space에 있는 새로운 공동체 토픽을 통해 developerWorks 공동체에 참여한다.

필자소개

Sean Walberg는 1994년 이래로 학계, 회사, 인터넷 서비스 제공 업체 환경을 두루 거치며 리눅스와 유닉스 분야에서 일해왔다. Walberg는 여러 해 동안 시스템 관리 서적을 집필해왔다.

이 문서 북마킹 하기

mar.gar.in naver eolin del.icio.us