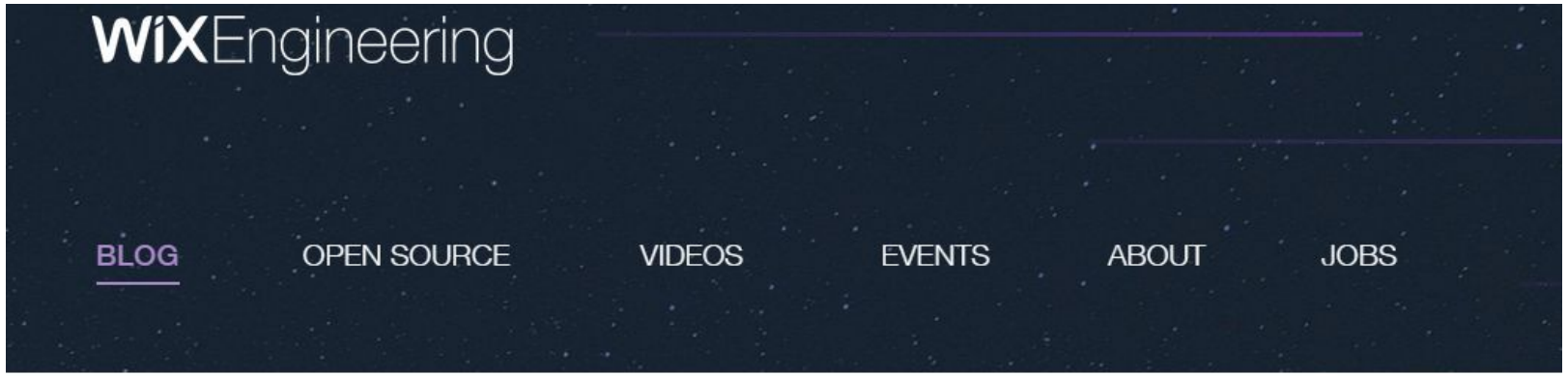
100M으로 스케일링
Key-Value 저장소로
MySQL을 사용 NoSQL
이상의 퍼포먼스를 내자

최흥배
https://jacking75.github.io/



Scaling to 100M: MySQL is a Better NoSQL

Yoav Abrahami | December 10th 2015 | MySQL

MySQL is a better NoSQL. When considering a NoSQL use case, such as key/value storage, MySQL makes more sense in terms of performance, ease of use, and stability. MySQL is a solid engine with lots of online material, ranging from operations and failure cases, to replication and different patterns of usage. For this reason, it has an advantage over newer NoSQL engines that are not as battle tested.

In recent years, NoSQL engines have become mainstream. Many developers look at NoSQL engines—such as MongoDB, Cassandra, Redis, or Hadoop—as their first choice for building applications, considering them a single family of products that

Search

Most Popular

1. Scaling to 100M: MySQL is a Better NoSQL
2. JavaScript: The Extra Good Parts
3. Testing Asynchronous Code

http://blog.wix.engineering/2015/12/10/scaling-to-100m-mysql-is-a-better-nosql/

MySQL은 NoSQL 보다 뛰어나다.

Key-Value 저장소라는 NoSQL의 활용 사례를 보면 퍼포먼스와 사용하기 쉬움, 안정성 면에서 MySQL 쪽이 합리적이다.

MySQL에는 운영 및 장애에 관한 것에서 리플리케이션이나 다른 사용 패턴까지 많은 온라인 자료가 있고 견실한 엔진이다.
이런 이유로 비교할 것도 없이, MySQL은 최근 NoSQL 엔진보다 뛰어나다라고 말할 수 있다.

현재는 NoSQL 엔진이 주류가 되고 있다. 많은 개발자가 MongoDB나 Cassandra, Redis, Hadoop 같은 NoSQL 엔진을 애플리케이션 구축의 첫 후보로 생각하고 이것들이 모든 옛날부터의 SQL 엔진을 웃도는 것으로 생각하고 있다.

NoSQL데이터베이스가 선택되는 이유에는 과대 광고나 "릴레이셔널 데이터베이스가 NoSQL과 같은 기능이 없다"라는 잘못된 억측이 요인에 있지만, 기술자는 자주 운영 비용과 안정성, 성숙도라는 우려 점을 간과하기 쉽다.

제약이나 다른 NoSQL(그리고 SQL) 엔진의 결함에 관한 자세한 정보는 Aphyr에 게재되고 있는 Jepsen 시리즈 기사를 봐라.

이 글은 왜 우리가 "Key-Value 활용 사례에서 대부분의 전용 NoSQL 엔진보다 MySQL을 사용하는 것이 좋다"라고 생각하고 있는지를 설명하고 가이드 라인도 제시한다.

Wix사이트의 해결

누군가가 Wix 사이트로의 링크를 클릭하면 브라우저는 Wix 서버로의 사이트 주소와 함께 HTTP 리퀘스트를 보낸다.

이것은 주소가 Wix 프리미엄 사이트의 커스텀 도메인(예를 들면 domain.com)이라도 Wix 도메인의 서브 도메인에 있는 프리 사이트(예를 들어 user.wix.com/site)이라도 발생한다.

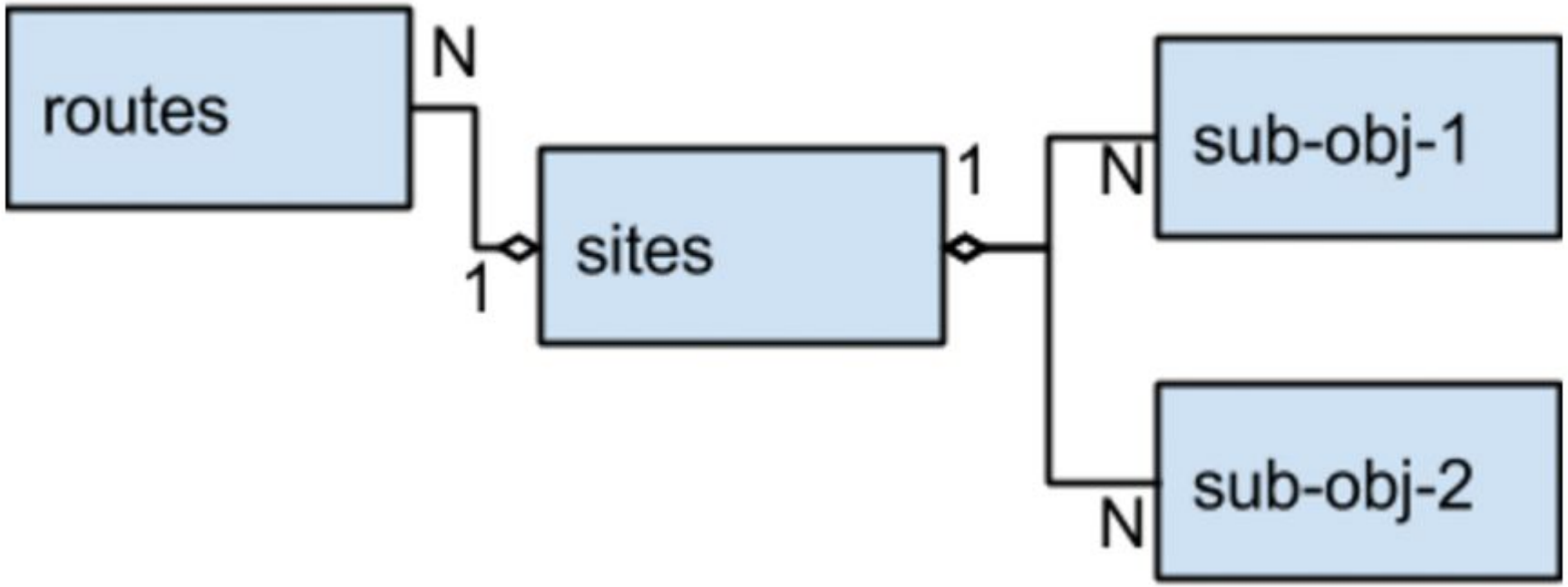
이 서버는 발송된 사이트 주소를 바탕으로 Key-Value 검색을 실시하는 것으로, 사이트의 URL을 해결해야 한다.

라우팅 테이블은 사이트 주소를 사이트 객체로 전환하는 데 쓰인다.

왜냐하면 사이트는 복수의 루트에 노출할 수 있고 관계는 다 대 일이기 때문이다. 사이트가 한번 발견되면 애플리케이션이 그것을 읽는다.

사이트 개체 자체는 사이트가 이용하는 다른 서비스인 자식 객체의 2개의 목록을 포함하는 복잡한 구조로 되어 있다.

다음은 표준적인 SQL과 정규화된 스키마를 가정한 우리 객체의 샘플 모델이다.



기존 정규화된 모델로 사이트를 갱신하는 경우 데이터 정합성이 유지되는 것을 보증하기 위해서 복수 테이블을 갱신하는 트랜잭션을 실시할 필요가 있다(트랜잭션은 영향을 받는 테이블에 대한 동시 기록/읽기를 방지하기 위해서 DB 락을 쓰는 것에 유의 하자). 이 모델에서는 각 테이블에 시리얼 키(자동 증가 번호), 외부 키, 그리고 루트 테이블 내의 URL 필드 인덱스를 갖게 된다.

그러나 정규화된 스키마에서의 모델링에는 몇 가지 문제가 있다.

- 락은 테이블에 대한 액세스를 제한하므로 높은 처리량의 활용 사례에서는 성능에 제한되는 것이 있다.
- 객체 읽기에서는 몇몇 SQL 쿼리(이 경우는 4개) 또는 JOIN을 일으킨다. 이것은 레이턴시와 밀접한 관계가 있다.
- 시리얼 키가 락을 강제하여 쓰기 산출량을 제한한다.

이들 문제는 결과적으로 MySQL(또는 다른 SQL 엔진)에서 얻어지는 처리량과 병행성에 제한을 받게 된다. 이러한 결점과 유스 케이스가 실제로는 key-value이라는 사실에서 많은 개발자는 안정성 및 정합성 및 유용성 등을 희생하더라도 더 나은 처리량과 병행성을 제공하는 NoSQL을 선택하고 있다.

Wix의 경우 우리는 MySQL을 Key-Value 저장소로 독창적으로 사용했을 때, 이것이 대부분의 NoSQL 엔진과 MySQL의 정규화된 데이터 모델(이런 경우) 보다 좋은 기능을 하는 것을 발견했다.

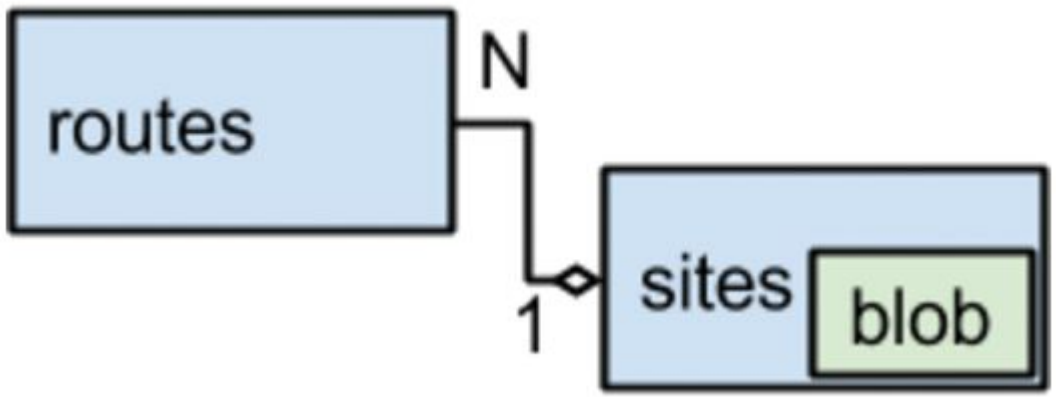
간단하게 MySQL을 NoSQL 엔진으로 사용해 보자.
우리의 현재 시스템에는 스케일링, 산출량, 병행성, 레이턴시 형태가 있고, 어떤 NoSQL 엔진에 대해서도 뛰어나다.

다음이 몇 가지 데이터이다.

- 3개의 데이터 센터에 대해서 액티브/액티브/액티브 구성을 설정한다.
- 스루풋은 200,000 RPM 정도 주문 크기.
- 10GB의 스토리지 사이즈에 100,000,000 레코드 정도의 라우팅 테이블.
- 200GB의 스토리지 사이즈에 100,000,000 레코드 정도의 사이트 테이블.
- 평균 1.0~1.5 밀리 초의 읽기 레이턴시 삭감(실제로는 1개의 데이터 센터에서는 0.2~0.3밀리 초).

유의할 것은 1.0밀리 초 정도의 레이턴시는 오픈 소스, 클라이드 베이스 어느 쪽에서도 대부분의 Key-Value 엔진보다 뛰어나다고 생각하고 있다. 그리고(베이직 SQL 엔진이라고 생각하고 있는) MySQL로 이를 실현하고 있습니다.

다음이 우리가 사용하고 있는 실제 스키마이다.



```
CREATE TABLE `routes` (
  `route` varchar(255) NOT NULL,
  `site_id` varchar(50) NOT NULL,
  `last_update_date` bigint NOT NULL,
  PRIMARY KEY (`key`),
  KEY (`site_id`)
)

CREATE TABLE `sites` (
  `site_id` varchar(50) NOT NULL,
  `owner_id` varchar(50) NOT NULL,
  `schema_version` varchar(10) NOT NULL DEFAULT '1.0',
  `site_data` text NOT NULL,
  `last_update_date` bigint NOT NULL,
  PRIMARY KEY (`site_id`)
) /*ENGINE=InnoDB DEFAULT CHARSET=utf8 ROW_FORMAT=COMPRESSED KEY_BLOCK_SIZE=16*/;
```

쿼리 조건으로 사용되지 않는 필드는 단일 BLOB 필드에 들어간다 (site_data 텍스트 필드).
여기에는 sub-obj 테이블과 객체 테이블 자체의 모든 필드도 포함된다.
또 우리는 시리얼 키를 사용하지 않고 클라이언트에서 생성된 GUID 값인 varchar(50)을 쓰는 것에 주목하자. 이에 대해서는 다음 섹션에서 자세히 설명한다.

다음은 우리가 쓰는 쿼리로 높은 처리량과 낮은 레이턴시를 가지고 있습다.

```
1. select * from sites where site_id = (
2.   select site_id from routes where route = ?
3. )
```

이것은 우선 루트 테이블 위에서 유니크한 인덱스에 의해서 쿼리를 실행한다. 이 쿼리는 1개의 결과만 반환한다. 그리고 프라이머리 키로 사이트를 찾는다.

그리고 다시 1개의 레코드를 찾는다. 이 중첩된 쿼리 구문은 양쪽의 SQL쿼리를 실행하기 위해서 데이터베이스를 한번만 왕복하는 것을 보증한다.

높은 트래픽·고 갱신 레이트 조건 하에서 평균 1밀리 세컨드 이하로 안정된 퍼포먼스를 발휘하고 있다. 갱신은 트랜잭션을 사용하지 않아도 대략 트랜잭션적으로 동작한다. 이것은 하나의 insert 스테이트먼트에 사이트 모두를 입력하고 루트가 입력될 때까지 쿼리에 등장하지 않기 때문이다. 그래서 처음 사이트를 입력하고 다음에 경로를 입력하면 사이트 테이블 내에 고아 자료가 있는 엣지 케이스라도 정합성이 유지되게 된다.

MySQL을 NoSQL 엔진으로 사용하기 위한 가이드 라인

지금까지의 예(그리고 Wix의 다른 경우)로부터 얻은 경험을 이용하여 MySQL을 NoSQL 엔진으로 사용하기 위한 가이드 라인 리스트를 간단하게 작성했다.

MySQL을 NoSQL 엔진으로 사용할 때 기억해 두면 좋은 것은 데이터베이스의 락이나 복잡한 쿼리 사용을 기피한다는 것이다.

- 락을 일으키는 트랜잭션을 사용하지 않고, 실용적인 트랜잭션을 사용한다.
- 시리얼 키를 사용하지 않는다. 시리얼 키는 락이나 복잡한 액티브/액티브 설정을 일으킨다.
- 클라이언트에서 생성한 유니크 키를 사용한다.우리는 GUID를 사용했다.

읽기에 최적화한 스키마를 설정할 때는 아래의 추가의 가이드 라인을 더 해라.

- 정규화 하지 않는다.
- 기존 필드만 색인한다. 인덱스에 필드가 필요 없는 경우, 단일 blob/텍스트 필드로 저장한다(JSON 및 XML 등).
- 외부 키를 사용하지 않는다.
- 쿼리에서 1줄만 읽어 올 수 있는 스키마를 설정한다.
- alter table 명령을 실행하지 않는다. Alter table 명령은 락이나 다운 타임을 일으키는 것으로, 라이브 마이레이션을 사용한다.

데이터를 쿼리 할 때는 아래를 의식하자.

- 프라이머리 키 또는 인덱스로 레코드를 쿼리 한다.
- JOIN은 사용하지 않는다.
- aggregation은 사용하지 않는다.
- 마스터 데이터 베이스 상이 아니라 레플리카 상에서 하우스 키핑 처리 (정기적으로 로그 파일을 삭제하는 것) 쿼리(BI나 데이터의 외삽 등)를 실행한다.

라이브 마이그레이션 및 실용적 트랜잭션에 대한 자세한 설명은 다른 블로그에서 소개할 예정이다.

정리

이 글에서 기억해 두었으면 하는 가장 중요한 것은 다른 생각도 할 수 있다는 것이다.

MySQL을 NoSQL 엔진으로 사용하는 것은 멋지지만 이는 원래 디자인된 사용 방법은 아니다. 여기에서 시연했듯이 이 예는 어디까지나 Key-Value접근을 위해 구축된 전용 NoSQL 엔진 대신 MySQL을 사용한다는 것이다.

Wix에서는 Key-Value 사례(그 외에도 공통)의 선택 중 하나로 MySQL 엔진을 선택 했다.
왜냐하면 사용하기 쉽고 조작이 쉽다는 멋진 생태계 때문이다.
또한, 대부분의 NoSQL 엔진에 못지않은 레이턴시와 처리량, 병행성 매트릭스를 제공하고 있다.