7️⃣

7장. 분산 시스템을 위한 유일 ID 생성기 설계

문제 이해 및 설계 범위 확정

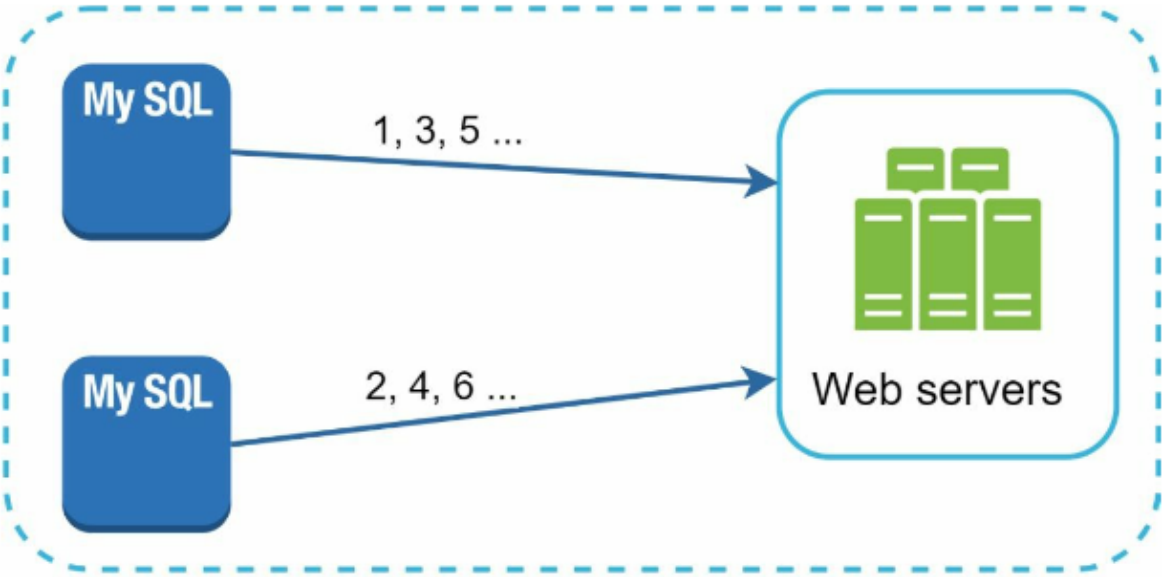
이번 챕터에서는 분산 시스템에서 유일한 ID를 생성하는 방법에 대해 공부해본다. 요구사항을 정리하면 아래와 같다.

- ID는 유일해야한다.
- ID는 숫자로만 구성되어야 한다.
- ID는 64비트로 표현될 수 있는 값이어야 한다.
- ID는 발급 날짜에 따라 정렬 가능해야 한다.
- 초당 10,000개의 ID를 만들 수 있어야 한다.

개략적 설계안 제시 및 동의 구하기

다중 마스터 복제

- db의 auto_increment 기능을 활용하여, 현재 사용중인 데이터베이스 서버의 수 k 만큼씩 증가시킴
- 서버의 수를 증가시킴으로써 초당 생성가능한 ID 수를 늘릴 수 있음
    - 이를 통해 규모 확장성 문제를 어느 정도 해결 가능



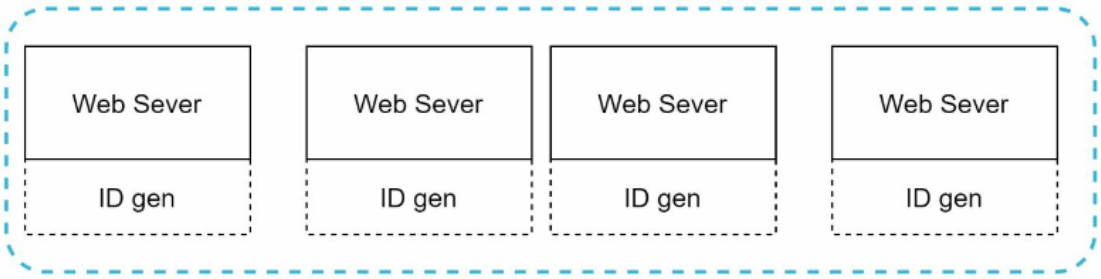
단점

- 여러 데이터 센터에 걸쳐 규모를 늘리기 어렵다
- ID의 유일성은 보장되지만, 그 값이 시간 흐름에 맞춰 커지도록 보장할 수 없다.
- 서버를 추가하거나 삭제할 때도 잘 동작하도록 만들기 어렵다

UUID

> 컴퓨터 시스템에 저장되는 정보를 유일하게 식별하기위한 128bit 크기의 수

- 충돌 가능성은 지극히 낮으며, 초당 10억개의 UUID를 100년동안 계속 만들지 않는 이상 중복의 가능성이 50% 이하이다
- 서버 간 조율 없이 ID 생성이 가능하다



장점

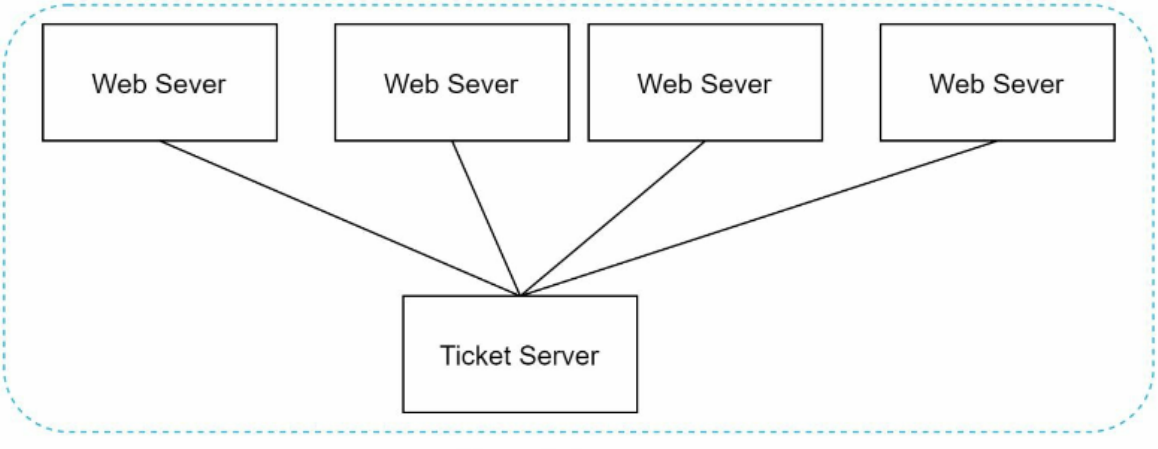
- UUID 생성은 단순하며, 동기화 이슈도 없다
- 각 서버가 알아서 ID를 만드는 구조이므로, 규모 확장도 쉽다

단점

- ID가 128bit로 길다
- 시간순으로 정렬할 수 없다
- 숫자가 아닌 값이 ID에 포함될 수 있다

티켓 서버

> auto increment 기능을 가진 하나의 서버에서 ID를 return 해주는 방식



장점

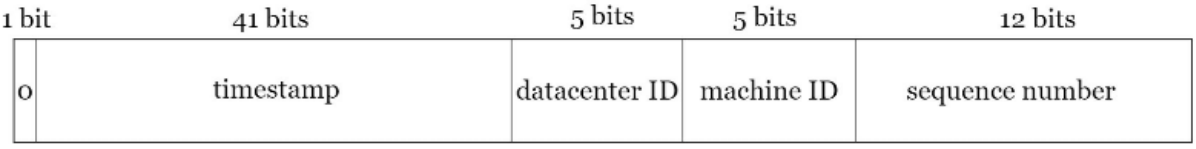
- 유일성이 보장되는 오직 숫자로만 구성된 ID를 쉽게 만들 수 있다
- 구현하기 쉽고, 중소 규모의 애플리케이션에 적합

단점

- SPOF가 될 수 있다
- SPOF를 피하기 위해 여러 서버를 준비한다면, 동기화의 문제가 생긴다

트위터 스노우플레이크 접근법

> 하나의 id를 여러 섹션으로 나누어 의미를 부여하는 방식



- 사인(sign) 비트 : 1비트를 할당한다. 음수와 양수를 구별하는데 사용한다. 현재는 필요없지만 추후를 위해 남겨둔다.
- 타임스탬프(timestamp) : 41비트를 할당한다. 기원 시각 (epoch) 이후로 몇 밀리초가 경과했는지를 나타내는 값이다.
- 데이터센터 ID : 5비트를 할당한다. 32개의 데이터센터를 지원가능.
- 서버 ID : 5비트를 할당한다. 각 데이터센터당 32개의 서버를 지원가능.
- 일련번호 : 12비트를 할당한다. 각 서버에서는 ID를 생성할 때마다, 일련번호를 1만큼 증가시키고 1밀리초가 경과할 때 마다 0으로 초기화시킨다.

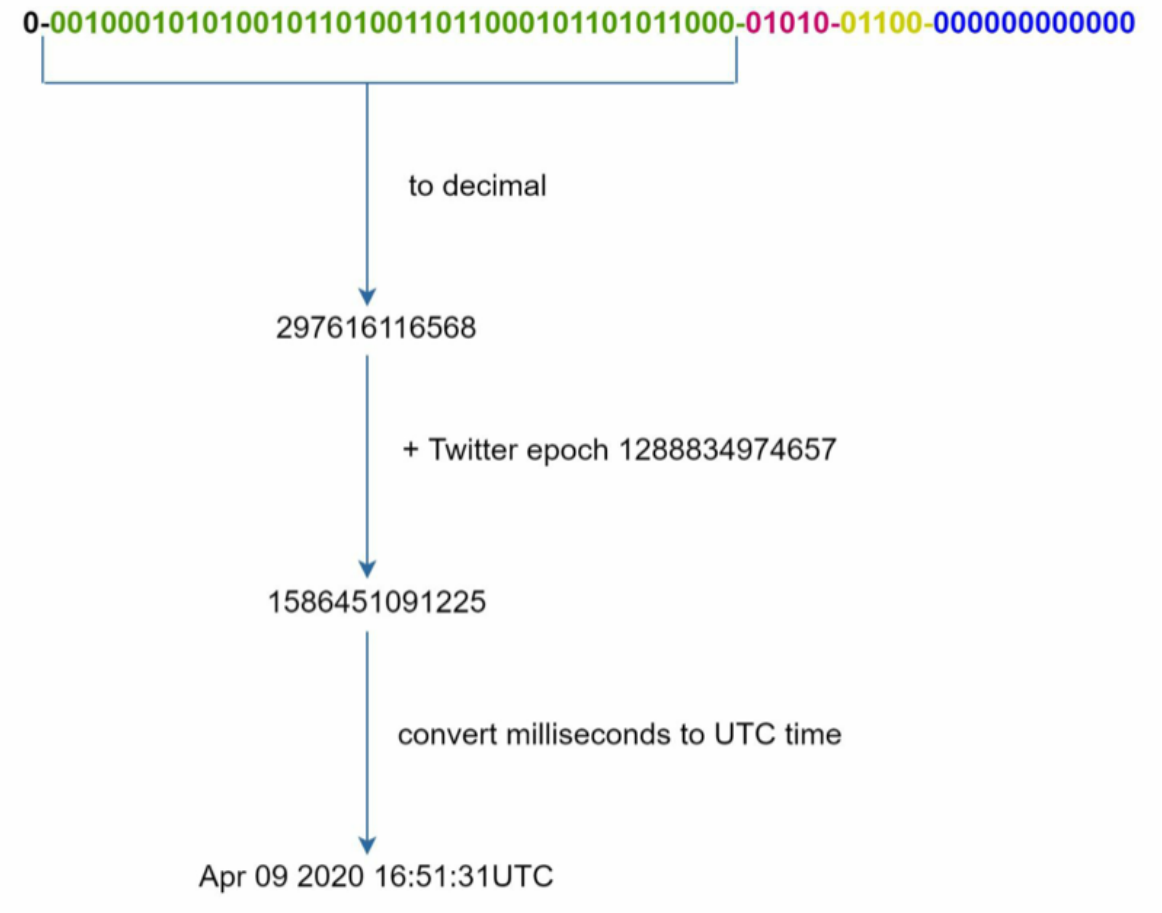
상세 설계

트위터 스노우프레이크 접근법을 사용하여 상세 설계 진행

- 데이터 센터 ID와 서버 ID는 시스템 시작시 결정되며, 운영되는 동안 변경되지 않는다
- 데이터 센터나 서버 ID를 잘못 변경하면, 충돌이 발생할 수 있으므로 신중히 변경해야한다
- 타임 스탬프나 일련번호는 ID 생성기가 동작하는 동안 만들어지는 값이다

타임 스탬프

- 타임스탬프는 41bit를 차지하고 있음
- 시간이 흐름에 따라 점점 큰 값을 가지기 때문에 정렬 가능
- 아래는 UTC 시각을 추출하는 예제



- 41bit는 최대 69년을 표현할 수 있으므로, 69년이 지난다면 기원 시각 또는 ID 체계를 변경해야한다

일련 번호

- 일련번호는 12bit이기 때문에, 4096개를 만들 수 있다
- 밀리초동안 하나 이상의 ID를 만드는 경우에만, 0보다 큰 값을 갖게될 것이다

마무리

스노플레이크 방식은 모든 요구사항을 만족하면서, 분산 환경에서 규모 확장이 가능했기 때문에 선택했다.

아래 사항들을 추가로 살펴보면 좋을 것 이다.

- 시계 동기화

    - 이번 설계를 진행하면서 우리는 ID 생성 서버들이 전부 같은 시계를 사용한다고 가정하였다
    - 이런 가정은 하나의 서버가 여러 코어에서 실행될 경우 유효하지 않을 수 있다
    - 여러 서버가 물리적으로 독립된 여러 장비에서 실행되는 경우에도 마찬가지
    - 이러한 문제를 해결하기 위해 NTP (Network Time Protocol) 이 가장 보편적인 해결 수단이니 참고해보자
- 각 절(section)의 길이 최적화
    - 동시성이 낮고, 수명이 긴 애플리케이션이라면 일련 번호 절의 길이를 줄이고 타임스탬의 길이를 늘리는 것이 좋을 것이다
- 고가용성
    - ID 생성기는 필수이므로, 높은 고가용성을 지원해야한다

참고 링크

https://velog.io/@kshired/가상-면접-사례로-배우는-대규모-시스템-설계-기초-분산-시스템을-위한-유일-ID-생성기-설계

https://hudi.blog/system-design-interview-alex-xu-7/