[7장] 분산 시스템을 위한 유일 ID 생성기 설계

가상 면접 사례로 배우는 대규모 시스템 설계 기초

이민석 / unchaptered

https://inblog.ai/monthly-cs
https://github.com/monthly-cs/2024-03-system-design-interview-1

문제 이해 및 설계 범위 확정

- **분산 시스템에서 사용 가능한 형태일 것 (auto_increment는 사용 불가)**
- **주요 제약 조건**
  - ID는 **유일**해야 함
  - ID는 **정렬**가능해야 함
  - ID는 **숫자**로만 구성되어야 함
  - ID는 **64비트**로 표현될 수 있어야 함
  - ID는 **초당 10,000개 이상** 생성되어야 함
  - Next ID는 Prev ID보다 항상 크지만, 그 차이가 1은 아님

개략적인 설계안 제시 및 동의 구하기

다중 마스터 복제
(Multi-Master Replication)

데이터베이스의 auto_increment를 **개별 서버**에서 활용

UUID
(Universally unique Identifier)

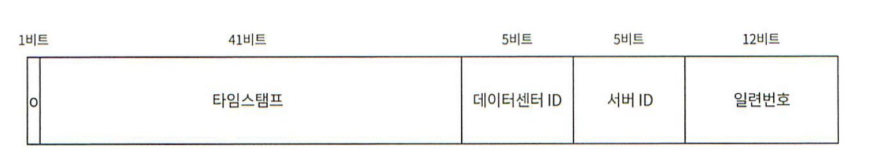
컴퓨터 시스템에 저장되는 정보를 유일하게 식별하기 위한 **128 비트** 짜리 수
→ 09c9e62-50b4-468d-bf8a-c07e1040bfb2

티켓 서버
(Ticket Server)

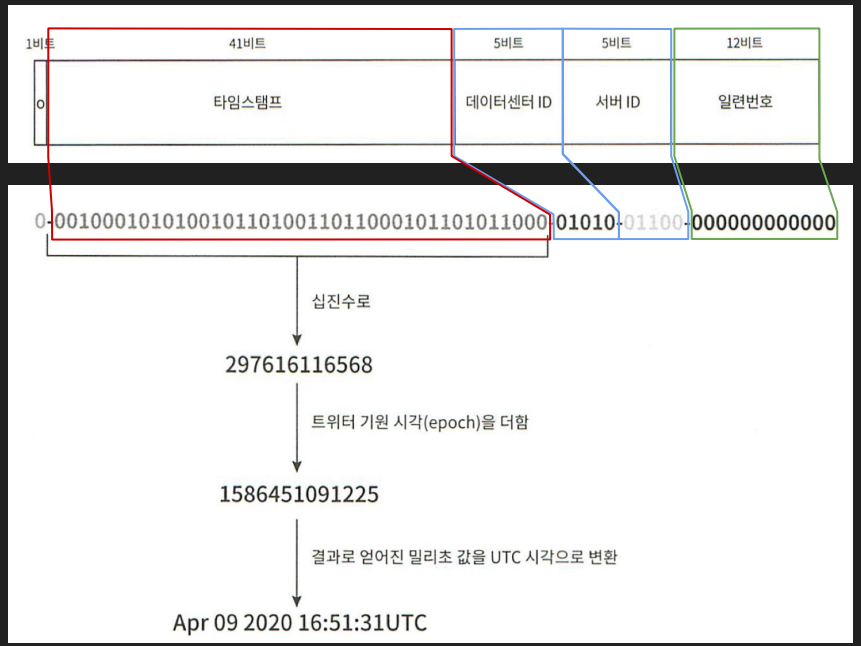
데이터베이스의 auto_increment를 **티켓 서버**에서만 발행

트위터 스노우 플레이크
(Twitter Snowflake)

분할 정복(Divide and Conquer)에 따라, ID를 여러 절로 분할하여 생성



트위터 스노우 플레이크 IDs (Twitter Snowflake IDs)



타임스탬프(Timestamp)

2^41 - 1 = 2,199,023,255,551 ms
≒ 2,199,023,255 s
≒ 610,839 h
≒ 25,451 d
≒ 69.73 y

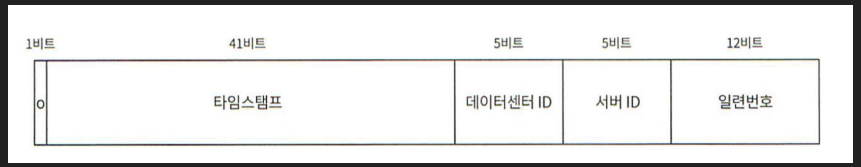
데이터센터/서버 ID(DataCenter/Server ID)

2^5 - 1 = 31

일련번호(Sequence ID)

2^12 = 4,096

트위터 스노우 플레이크 IDs (Twitter Snowflake IDs)



타임스탬프(Timestamp)

2^41 - 1 = 2,199,023,255,551 ms
≒ 2,199,023,255 s
≒ 610,839 h
≒ 25,451 d
≒ 69.73 y

데이터센터/서버 ID(DataCenter/Server ID)

2^5 - 1 = 31

일련번호(Sequence ID)

2^12 = 4,096

단일 데이터센터 + 단일 서버일 경우, ID 중복 상황

1밀리초(1/1000초) 동안, **4096 + N개**의 ID 생성 요청이 된 경우

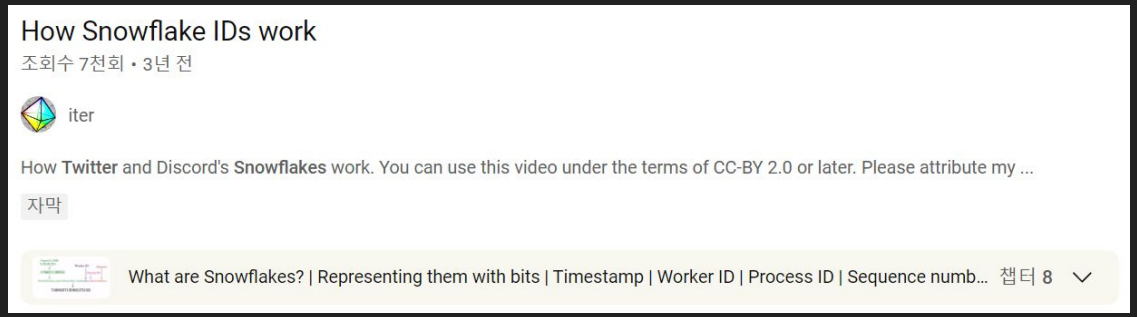
→ 초당 4,096,000 + N개 이상의 ID 생성 요청이 들어와야, 중복이 된다.

단일 데이터센터 + 다수 서버(M)일 경우, ID 중복 상황

1밀리초(1/1000초) 동안, **4096*M + N개**의 ID 생성 요청이 된 경우

→ 초당 4,096,000 * M + N개 이상의 ID 생성 요청이 들어와야, 중복이 된다.
→ 4개 서버 및 10개 프로세스인 경우, 163,840,000 + N개의 요청이 분기점

Twitter Snowflake IDs Work



https://youtu.be/aLYKd7h7vgY?si=8OP9oKe4SeLrzBdW

더 나아가서…

- **서버 동기화(Clock Synchronization)**
  서버 별로 다른 시계를 쓸 경우, Network Time Protocol 사용

- **길이 최적화(Length Optimization)**
  동시성이 낮고 수명이 길다면, 일련번호(Sequence ID)를 N바이트 줄이고 타임스탬프(Timestamp)를 N바이트 늘리자

- **고가용성(High Availability)**
  ID 생성기에 대해서는 높은 수준의 가용성이 보장

- **재해 복구 전략(Disaster Recovery Strategy)**
  ID 생성기가 SPoF로 작용하지 않으려면, 물리적 장애에 대한 Active-Active 재해복구 전략이 필요하다.
  Active-Active Multi Site 전략을 사용하는 것이 어떨까?

감사합니다.