1️⃣

231018 11. 뉴스 피드 시스템 설계

홈 페이지 중앙에 지속적으로 업데이트되는 스토리

사용자 상태 정보 업데이트, 사진, 비디오, 링크, 앱 활동

팔로하는 사람들, 페이지, 그룹으로부터 나오는 좋아요

페이스북 뉴스 피드 설계

인스타그램 피드 설계

트위터 타임라인 설계

▼ 문제 이해 및 설계 범위 확정

- 사용자는 뉴스 피드 페이지에 새로운 스토리를 올릴 수 있어야 함
- 친구들이 올린 스토리를 볼 수 있어야 함
- 시간 흐름 역순으로 표시되어야 함
- 한 명의 사용자는 최대 5000명의 친구를 가질 수 있음
- 매일 천만 명이 방문한다고 사정
- 스토리에는 이미지나 비디오 등의 미디어 파일이 포함될 수 있음

▼ 개략적 설계안 제시 및 동의 구하기

1. 피드 발행
    - 사용자가 스토리를 포스팅하면 해당 데이터를 캐시와 DB에 기록
        - 새 포스팅은 친구의 뉴스 피드에도 전송
2. 뉴스 피드 생성
    - 모든 친구의 포스팅을 시간 흐름 역순으로 모아서 만든다고 가정

▼ 뉴스 피드 API

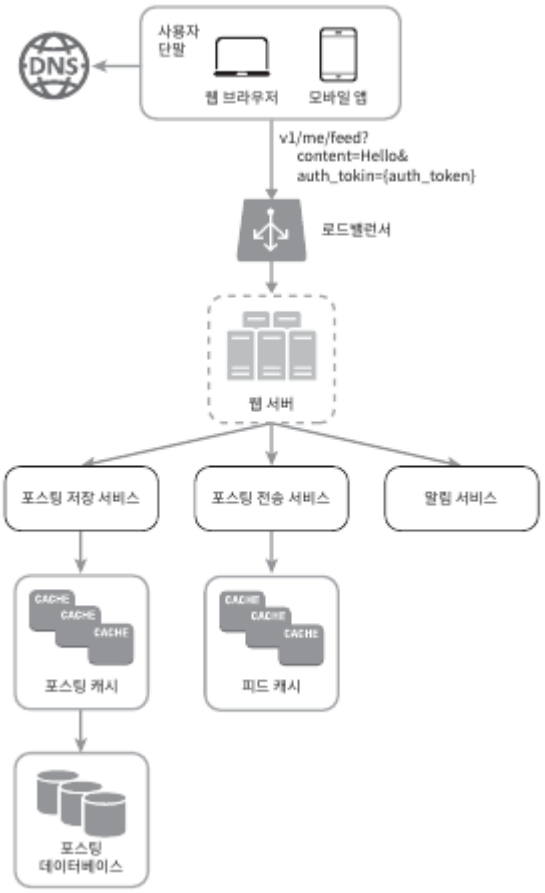
피드 발행 API

- 새 스토리를 포스팅하기 위한 API
- HTTP POST
- 인자
  - body
    - 포스팅 내용에 해당
  - Authorization 헤더
    - API 호출을 인증하기 위해 사용

피드 읽기 API

- 뉴스 피드를 가져오는 API
- 인자
  - Authorization 헤더
    - API 호출을 인증하기 위해 사용

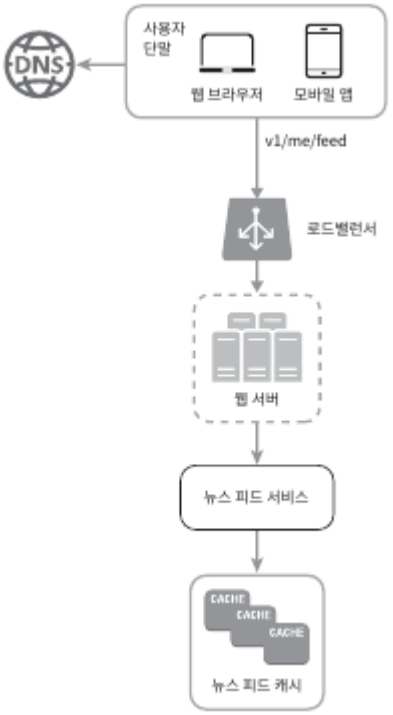
▼ 피드 발행



- 사용자
  - 모바일 앱이나 브라우저에서 새 포스팅을 올리는 주체
- 로드밸런서
  - 트래픽을 웹 서버들로 분산
- 웹 서버
  - HTTP 요청을 내부 서비스로 중계
- 포스팅 저장 서비스
  - 새 포스팅을 DB와 캐시에 저장
- 포스팅 전송 서비스
  - 새 포스팅을 친구의 뉴스 피드에 푸시
  - 뉴스 피드 데이터는 캐시에 보관하여 빠르게 읽어갈 수 있도록 함
- 알림 서비스
  - 친구들에게 새 포스팅이 올라왔음을 알림

- 푸시 알림을 보냄

▼ 뉴스 피드 생성



- 사용자
  - 뉴스 피드를 읽는 주체
- 로드밸런서
  - 트래픽을 웹 서버들로 분산
- 웹 서버
  - 트래픽을 뉴스 피드 서비스로 전송
- 뉴스 피드 서비스
  - 캐시에서 뉴스 피드를 가져오는 서비스
- 뉴스 피드 캐시
  - 뉴스 피드를 렌더링할 때 필요한 피드 ID를 보관

▼ 상세 설계

▼ 피드 발행 흐름 설계

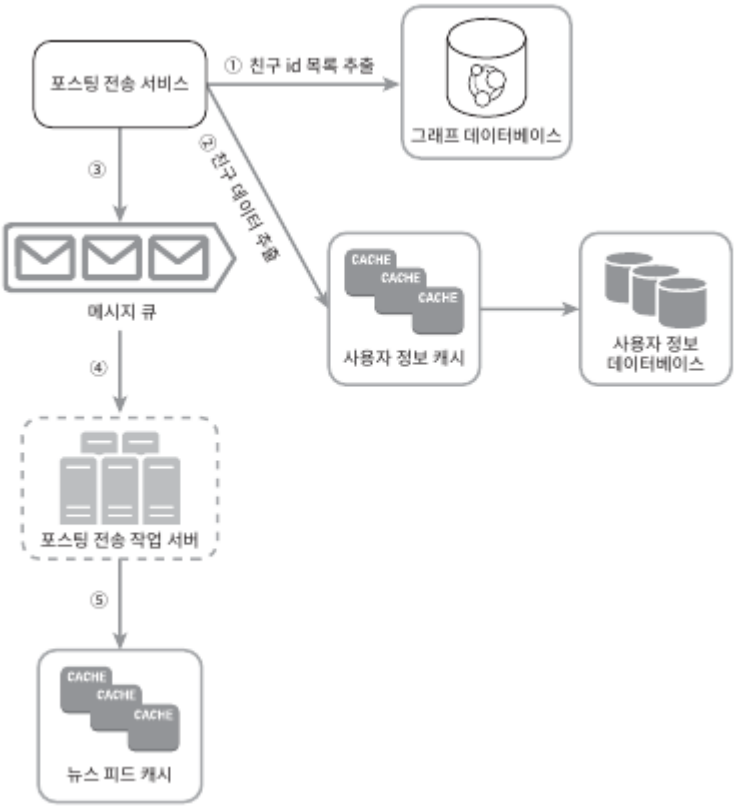
웹 서버

- 클라이언트와 통신

- 인증
- 처리율 제한

포스팅 전송(팬아웃) 서비스

- 어떤 사용자의 새 포스팅을 그 사용자와 친구 관계에 있는 모든 사용자에게 전달하는 과정
- 쓰기 시점에 팬아웃하는 모델: push
  - 새로운 포스팅을 기록하는 시점에 뉴스피드를 갱신
  - 포스팅이 완료되면 바로 해당 사용자의 캐시에 해당 포스팅을 기록
  - 장점
    - 뉴스 피드가 실시간으로 갱신되며 친구 목록에 있는 사용자에게 즉시 전송됨
    - 포스팅이 쓰이는 시점에 전송되므로 뉴스 피드를 읽는 데 드는 시간이 짧아짐
  - 단점
    - 친구가 많은 사용자의 경우 뉴스 피드 갱신에 많은 시간이 소요될 수 있음: 핫키 이슈
    - 서비스를 자주 이용하지 않는 사용자의 피드까지 갱신됨: 컴퓨팅 자원 낭비
- 읽기 시점에 팬아웃하는 모델: pull
  - 피드를 읽어야 하는 시점에 뉴스 피드를 갱신
  - 요청 기반 모델
  - 장점
    - 비활성화된 사용자 또는 거의 로그인하지 않는 사용자의 경우에 유리
    - 로그인하기까지는 어떤 컴퓨팅 자원도 소모하지 않음
    - 데이터를 친구 각각에 푸시하는 작업이 필요 없으므로 핫키 문제 없음
  - 단점
    - 뉴스 피드를 읽는데 많은 시간이 소요
- 두 가지 방법을 결합

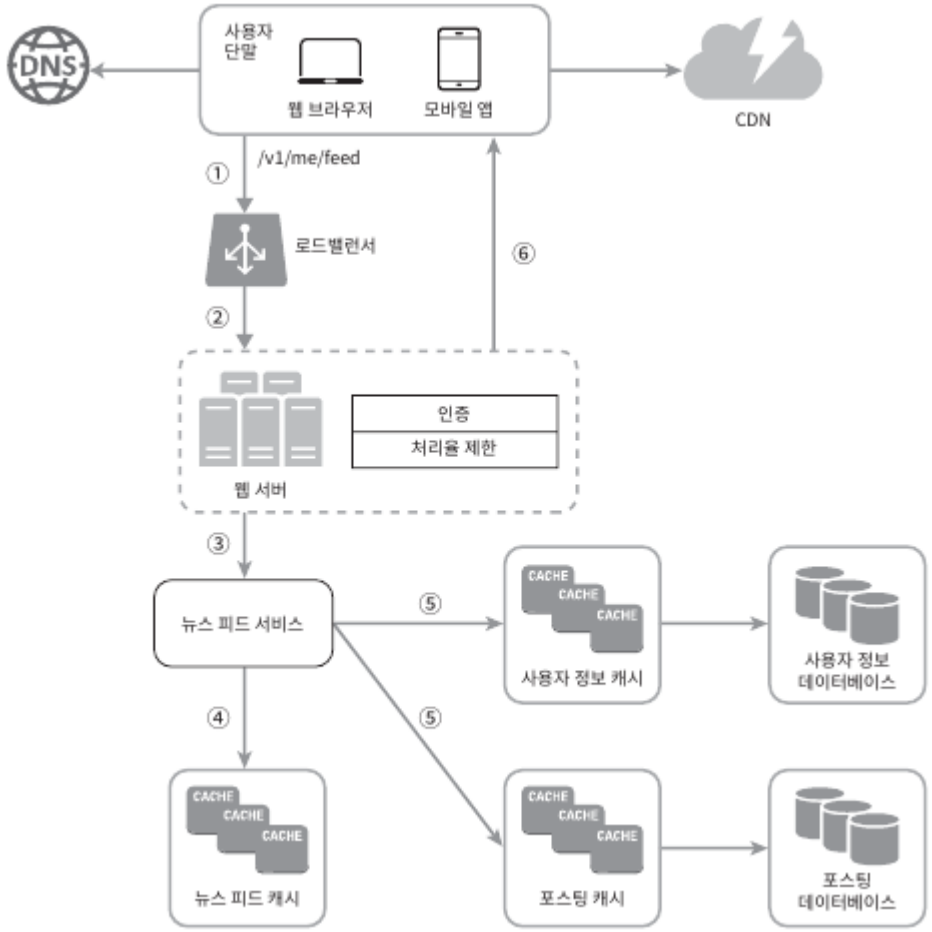


- 뉴스 피드를 빠르게 가져올 수 있도록 하는 것은 아주 중요하므로 대부분의 사용자에 대해서는 푸시 모델을 사용
- 친구가 많은 사용자는 해당 사용자의 포스팅을 필요할 때 가져가도록 하는 풀 모델을 사용하여 시스템 과부하 방지
- 안정 해시를 통해 요청과 데이터를 보다 고르게 분산하여 핫키 문제 최대한 해결
- 동작 방법
  1. 그래프 DB에서 친구 ID 목록 조회
  2. 사용자 정보 캐시에서 친구들 정보 조회
  3. 사용자 설정에 따라 일부 걸러냄
     - 피드 업데이트 무시하기로 한 친구
     - 새 포스팅이 일부 친구에게만 보여지도록 설정한 경우
  4. 친구 목록과 새 스토리의 포스팅 ID를 메시지 큐에 넣음
  5. 팬아웃 작업 서버가 메시지 큐에서 데이터를 꺼내어 뉴스 피드 데이터를 뉴스 피드 캐시에 넣음

- 뉴스 피드 캐시는 <포스팅 ID, 사용자 ID> 순서쌍을 보관하는 매핑 테이블

6. 새로운 포스팅이 만들어질 때마다 이 캐시에 레코드들이 추가
    - 사용자 정보와 포스팅 정보 전부를 이 테이블에 저장하지 않는 이유는 메모리 요구량이 지나치게 늘어날 수 있기 때문
    - 메모리 크기를 적정 수준으로 유지하기 위해 이 캐시의 크기에 제한을 두며 해당 값은 조정이 가능하도록 함

▼ 피드 읽기 흐름 설계



1. 사용자가 뉴스 피드를 읽으려는 요청 전송
2. 로드밸런서가 요청을 웹 서버 가운데 하나로 전송
3. 웹 서버는 피드를 가져오기 위해 뉴스 피드 서비스를 호출
4. 뉴스 피드 서비스는 뉴스 피드 캐시에서 포스팅 ID 목록 조회

5. 뉴스 피드에 표시할 사용자 이름, 사진, 콘텐츠 등을 사용자 캐시와 포스팅 캐시에서 가져와 완전한 뉴스 피드 생성
6. 생성된 뉴스 피드를 JSON 형태로 클라이언트에 전송
7. 클라이언트는 해당 피드를 렌더링

▼ 캐시 구조

- 뉴스 피드 시스템의 핵심 컴포넌트
- 뉴스 피드
  - 뉴스 피드의 ID를 보관
- 콘텐츠
  - 포스팅 데이터를 보관
  - 인기 콘텐츠는 따로 보관
- 소셜 그래프
  - 사용자 간 관계 정보 보관
- 행동
  - 포스팅에 대한 사용자의 행위에 관한 정보를 보관
  - 포스팅에 대한 '좋아요', 답글 등등이 이에 해당
- 횟수
  - '좋아요' 횟수, 응답 수, 팔로어 수, 팔로잉 수 등의 정보를 보관

▼ 마무리

데이터베이스 규모 확장

- 수직적 규모 확장 vs 수평적 규모 확장
- SQL vs NoSQL
- 주-부 데이터베이스 다중화
- 복제본에 대한 읽기 연산
- 일관성 모델
- 데이터베이스 샤딩

이 외 주제

- 웹 계층을 무상태로 운영하기
- 가능한 한 많은 데이터를 캐시할 방법
- 여러 데이터 센터를 지원할 방법
- 메시지 큐를 사용하여 컴포넌트 사이의 결합도 낮추기
- 핵심 메트릭에 대한 모니터링
    - 트래픽이 몰리는 시간대의 QPS, 사용자 피드를 새로고침할 때의 지연시간

▼ 토론

게시글 내용이 바뀔 경우에는 어떻게 처리해야 할까