뉴스 피드 시스템 설계

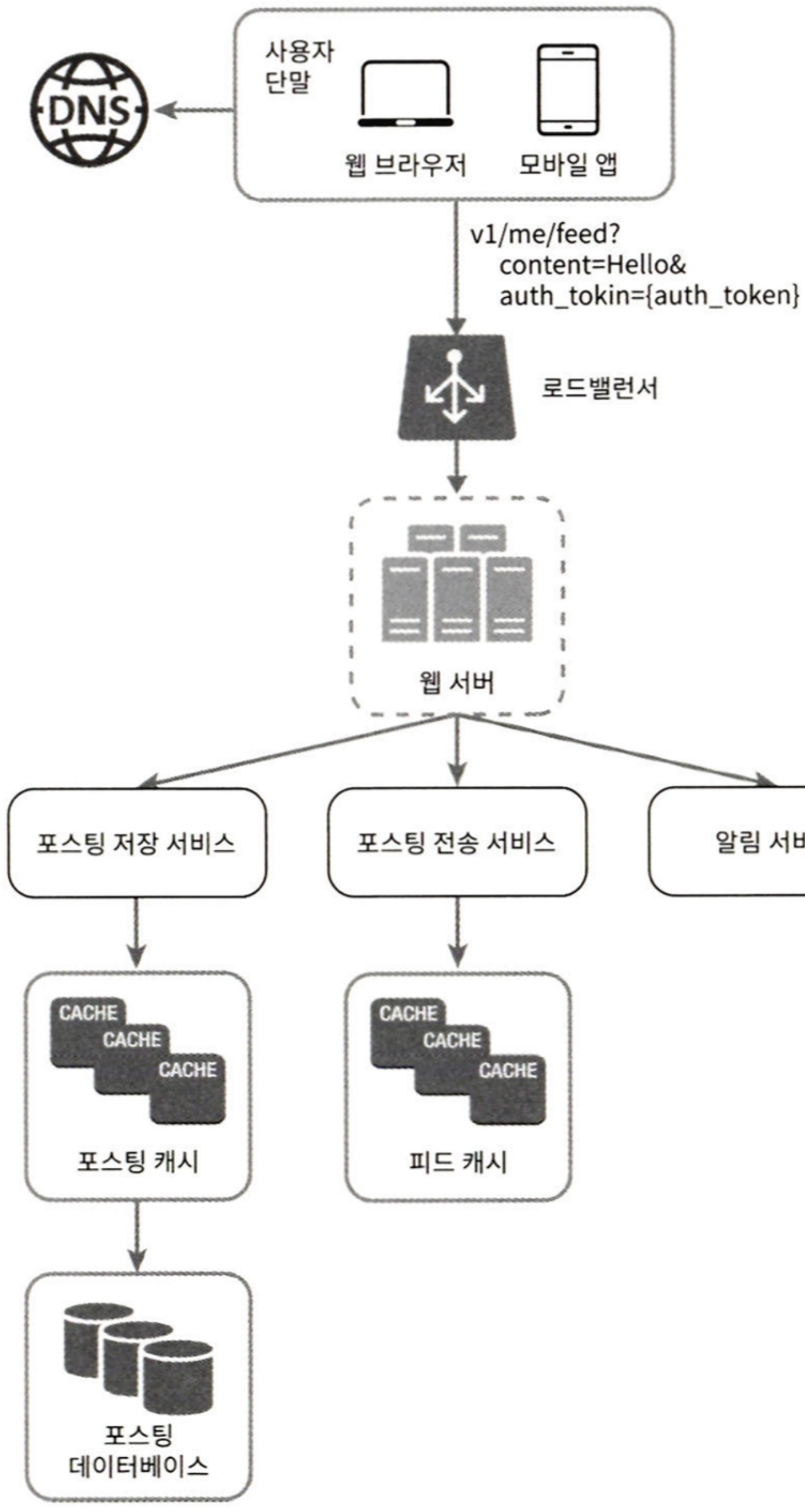
개략적 추정

- 모바일, 웹 둘 다 지원
- 뉴스 피드 페이지에 새로운 피드를 올릴 수 있어야 함. 친구들이 올리는 피드를 볼 수도 있어야 함.
- 시간은 역순
- 최대 5000명의 친구
- 매일 1000만명 사용
- 이미지, 비디오 등의 미디어 파일도 포함

개략적 설계안 제시 및 동의 구하기

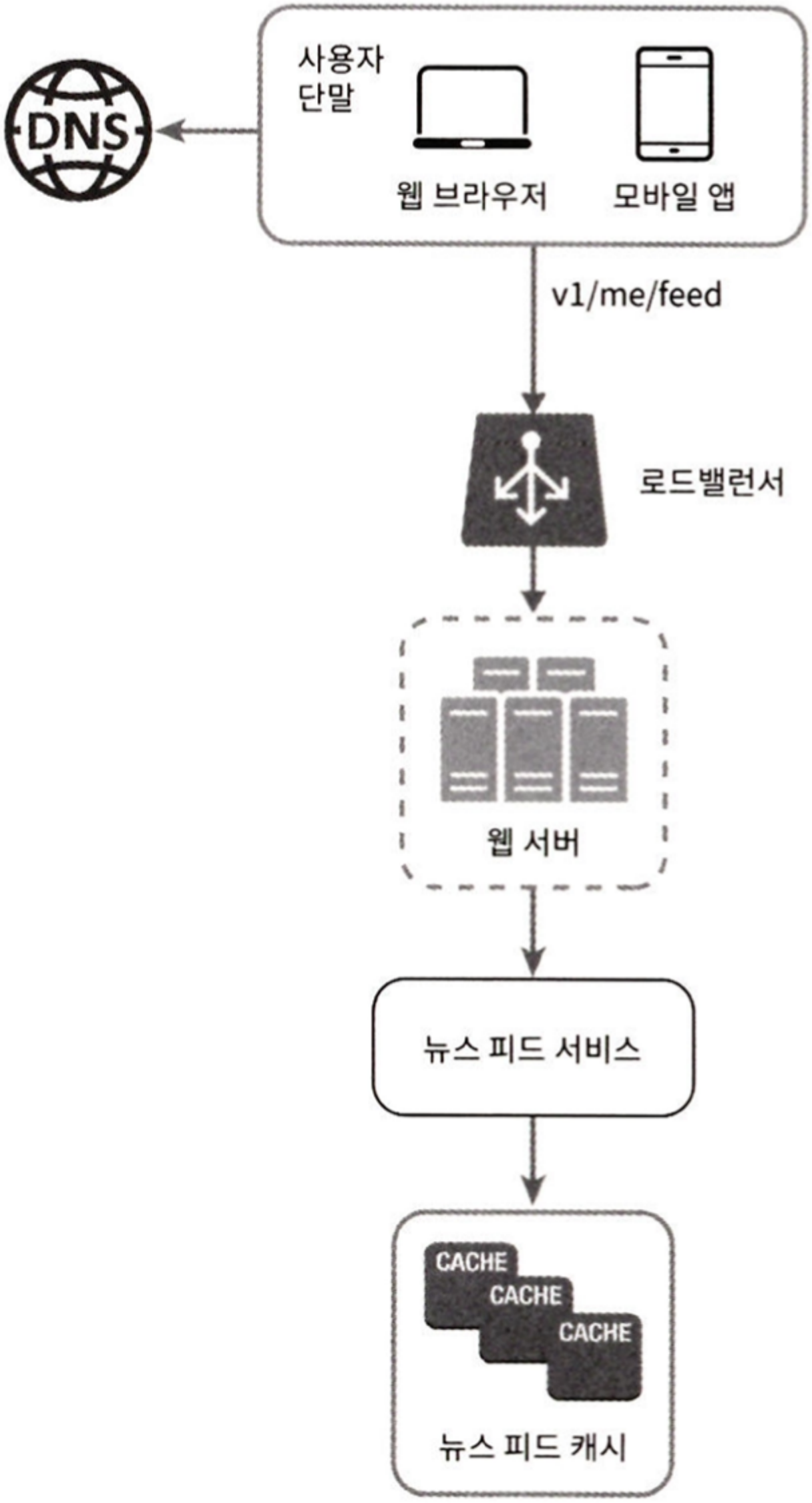
뉴스 피드 API

- 피드 발행
  - POST /v1/me/feed


사용자
단말
DNS
웹 브라우저
모바일 앱
v1/me/feed?
content=Hello&
auth_tokin={auth_token}
로드밸런서
웹 서버
포스팅 저장 서비스
포스팅 전송 서비스
알림 서비스
CACHE
CACHE
CACHE
포스팅 캐시
CACHE
CACHE
CACHE
피드 캐시
포스팅
데이터베이스

- 사용자

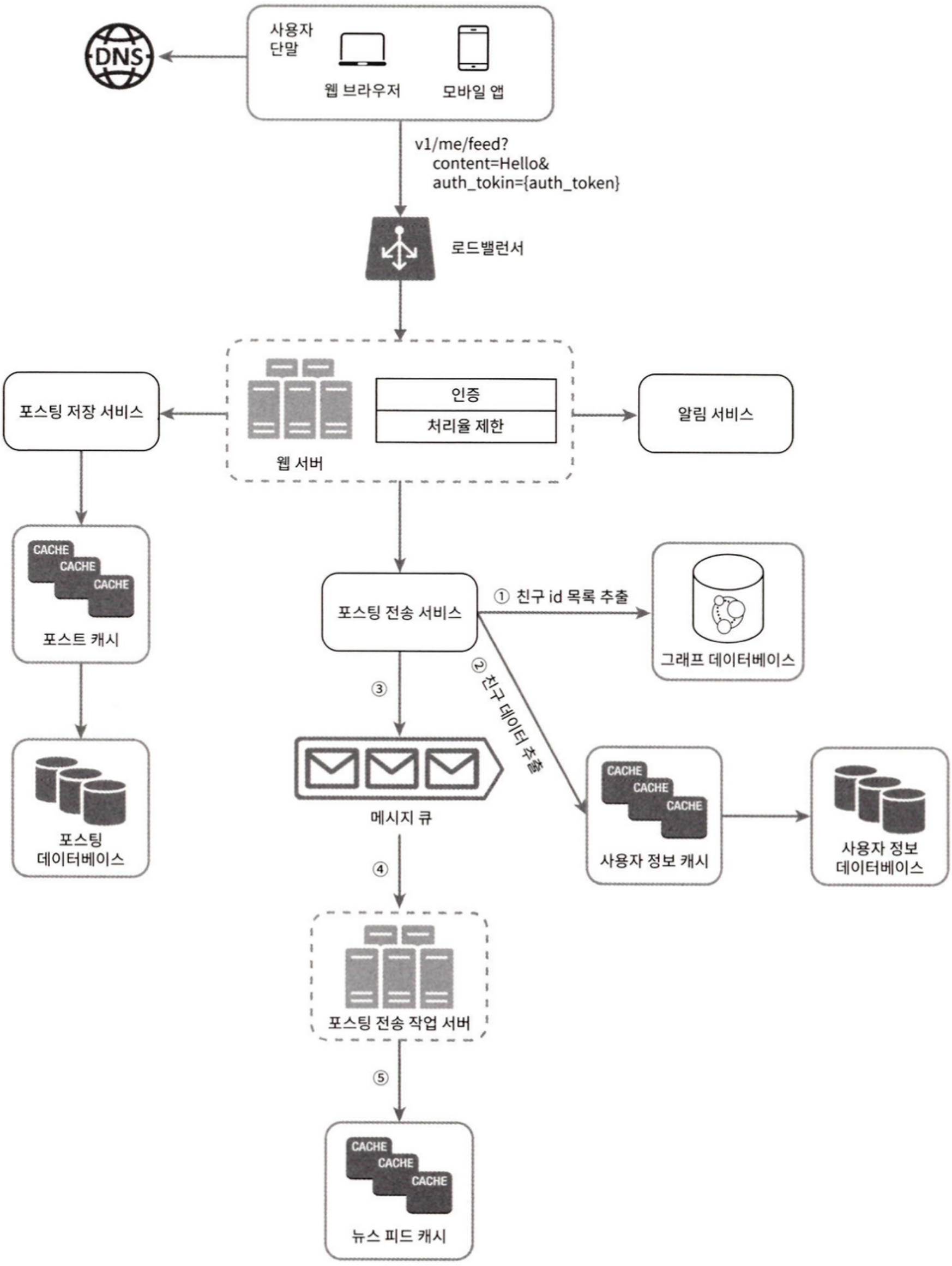
        - 피드 발생 API를 사용하여 새 포스팅을 올린다.
    - 로드밸런서
        - 트래픽을 웹 서버들로 분산한다.
    - 웹서버
        - HTTP 요청을 내부 서비스로 중계한다.
    - 포스팅 저장 서비스
        - 새 포스팅을 데이터베이스와 캐시에 저장한다.
    - 포스팅 전송 서비스
        - 새 포스팅을 친구의 뉴스 피드에 푸시
        - 캐시에 보관하여 빠르게 읽을 수 있도록 한다.
    - 알림 서비스
        - 친구들에게 새 포스팅에 대한 알림을 보낸다.
- 피드 읽기



- 사용자
  - 피드 읽기 API를 호출한다.
- 로드밸런서
  - 트래픽을 웹 서버들로 분산한다.
- 웹서버

        - 트래픽을 뉴스 피드 서비스로 보낸다.
    - 뉴스 피드 서비스
        - 캐시에서 뉴스 피드를 가져온다.
    - 뉴스 피드 캐시
        - 뉴스 피드를 렌더링할 때 필요한 ID를 보관한다.

상세 설계



웹 서버

- 클라이언트와 통신
- 인증
  - Authorization 헤더에 토큰을 포함하는 사용자만 포스팅 가능
- 처리율 제한 (rate limiter)

    - 스팸 차단, 유해한 콘텐츠가 자주 올라오는 것을 방지

팬아웃 서비스

- 팬아웃(fanout)
    - 어떤 사용자의 새 포스팅을 그 사용자와 친구 관계에 있는 모든 사용자에게 전달하는 과정
- 쓰기 시점 팬아웃 (fanout-on-write or push model)
- 읽기 시점 팬아웃 (fanout-on-read or pull model)

쓰기 시점 팬아웃 fanout-on-write (push) model

- 새로운 포스팅을 기록하는 시점에 뉴스 피드 갱신
- 즉포스팅이 완료되면 바로 해당 사용자의 캐시에 해당 포스팅 기록
- 장점
    - 뉴스 피드가 실시간으로 갱신, 친구 목록에 있는 사용자에게 즉시 전송
    - 새 포스팅이 기록되는 순간에 뉴스 피드가 이미 갱신되므로, 뉴스 피드를 읽는 데 드는 시간이 짧아짐
- 단점
    - 친구가 많은 사용자의 경우 친구 목록을 가져오고 그 목록에 있는 사용자 모두의 뉴스 피드를 갱신하는 데 많은 시간이 소요될 수 있음
    - 서비스를 자주 이용하지 않는 사용자의 피드까지 갱신해야 하므로 컴퓨팅 자원이 낭비됨

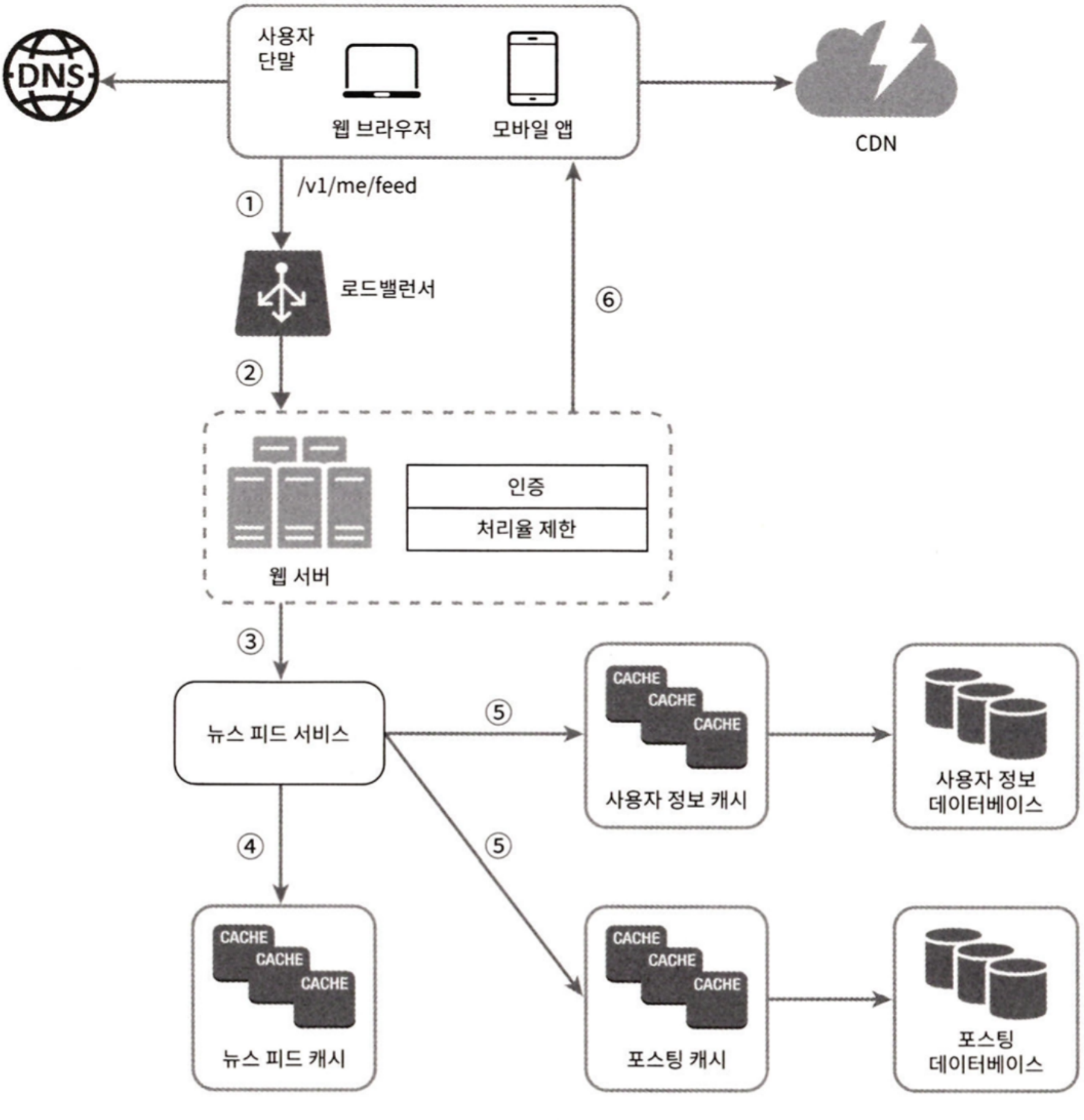
읽기 시점 팬아웃 fanout-on-read (pull) model

- 피드를 읽어야 하는 시점에 뉴스 피드를 갱신
- 요청 기반 (on-demand) 모델
- 사용자가 본인 홈페이지나 타임라인은 로딩하는 시점에 새로운 포스트를 가져오게 됨
- 장점
    - 비활성화된 사용자, 또는 서비스에 거의 로그인하지 않는 사용자의 경우에는 이 모델이 유리함.
    - 로그인 전까지는 어떤 컴퓨팅 자원도 소모하지 않음
    - 데이터를 친구 각각에 푸시하는 작업이 필요 없으므로 hotkey 문제 발생 X
- 단점
    - 뉴스 피드를 읽는데 많은 시간이 소요될 수 있음

타협안

- 뉴스 피드를 빠르게 가져오는 것은 아주 중요.
    - 대부분의 사용자는 푸시 모델을 사용함
- 친구나 팔로워가 아주 많은 사용자의 경우
    - 팔로워로 하여금 해당 사용자의 포스팅을 필요할 때 가져가도록 읽기시점 팬아웃모델 사용
- 안정 해시를 통해 요청과 데이터를 보다 고르게 분산하여 핫키 문제 완화

피드 읽기 흐름 상세 설계



1. 사용자가 뉴스 피드를 읽으려고 요청
2. 로드밸런서가 요청을 웹 서버 가운데 하나로 로드밸런싱 진행
3. 웹 서버는 뉴스 피드 서비스 호출
4. 뉴스 피드 서비스는 뉴스 피드 캐시에서 ID 목록을 획득
5. 뉴스 피드에서 표시할 사용자 이름, 사용자 사진, 포스팅 콘텐츠, 이미지 등을 사용자 캐시와 포스팅 캐시에서 가져와 완전한 뉴스 피드를 만듦
6. 생성된 뉴스 피드를 JSON 형태로 클라이언트에게 전송, 클라이언트는 해당 피드를 렌더링 함

캐시 구조

- 뉴스 피드
  - 뉴스 피드의 id 보관
- 콘텐츠
  - 일반, 인기 콘텐츠를 분리하여 보관
- 소셜 그래프
  - 사용자간의 관계를 보관
- 행동
  - 사용자 행위 관계를 분리하여 보관
- 횟수
  - 행동의 횟수를 분리하여 보관

마무리

- 데이터베이스 규모 확장
  - 스케일 업 vs 아웃
  - RDBMS vs NoSQL
  - master - slave 다중화
  - 읽기 전용 레플리카
  - 일관성 모델
  - 샤딩
- 추가 논의 주제
  - Stateless 하게 웹 계층 운영
  - 데이터 캐싱을 가능한 많이 하는 방법
  - 여러 데이터 센터를 지원할 방법
  - 메시지 큐를 사용하여 결합도 낮추기
  - 메트릭 모니터링