

11장 - 뉴스 피드 시스템 설계

≡ Tags

[1. 문제 이해 및 설계 범위 확정](#)

[2. 개략적 설계안 제시 및 동의 구하기](#)

[1. 피드 발행 \(feed publishing\)](#)

[2. 뉴스 피드 생성\(news feed building\)](#)

[뉴스피드 API](#)

[피드 발행 개략적 형태](#)

[뉴스 피드 생성](#)

[상세 설계](#)

[피드 발행 흐름 상세 설계](#)

[웹 서버](#)

[포스팅 전송\(팬아웃\) 서비스](#)

[두 가지의 방법을 결합하여 장점을 취하고 단점을 버려보자](#)

[캐시 구조](#)

[마무리](#)

[데이터 베이스 규모 확장](#)

[논의해볼 주제](#)

1. 문제 이해 및 설계 범위 확정

1. 모바일 앱, 웹 지원 여부
2. 새로운 피드를 올릴 수 있어야함
3. 기준이 필요
 - 토픽 점수
 - 시간 흐름 역순
4. 방문자의 규모 측정이 필요함

2. 개략적 설계안 제시 및 동의 구하기

1. 피드 발행 (feed publishing)

- 사용자가 스토리를 포스팅하면 해당 데이터를 캐시와 데이터베이스에 기록하는 역할
- 새 포스팅은 친구의 뉴스 피드에도 전송된다.

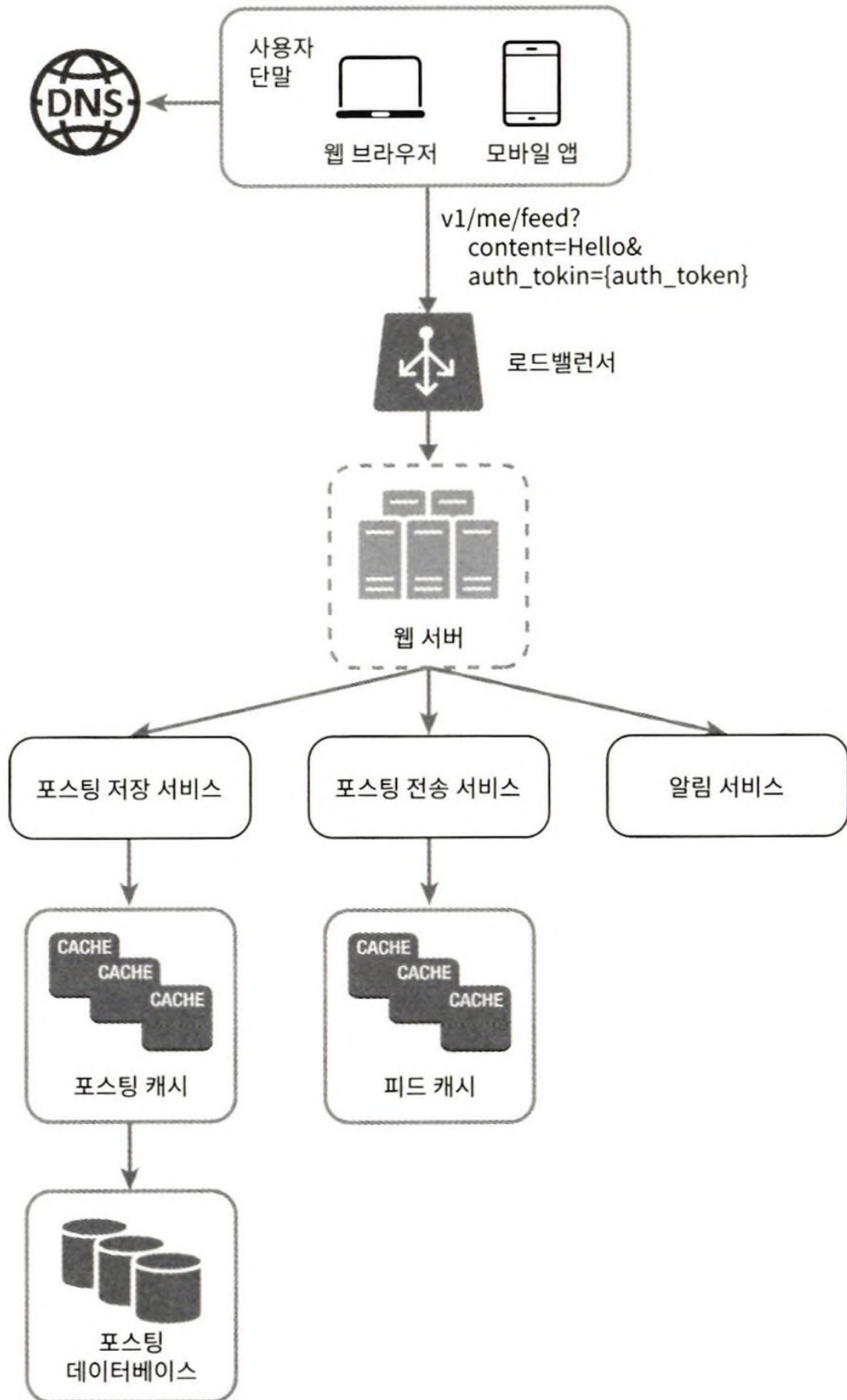
2. 뉴스 피드 생성(news feed building)

- 지면 관계상 뉴스 피드는 모든 친구의 포스팅을 시간 흐름 역순으로 모아서 만든다고 가정한다.

뉴스피드 API

- HTTP 프로토콜 기반이고, 상태 정보를 업데이트하거나, 뉴스 피드를 가져오거나, 친구를 추가하는 등의 다양한 작업을 수행하는데 사용
- 피드 발행 API
 - `POST /v1/me/feed`
 - 인자 :
 - 바디(body) → 포스팅 내용에 해당.
 - Authorization 헤더 : API 호출을 인증하기 위해 사용
- 피드 읽기 API
 - `GET /v1/me/feed`
 - 인자 :
 - Authorization 헤더 : API 호출을 인증하기 위해 사용

피드 발행 개략적 형태



- 사용자 : 모바일 앱이나 브라우저에서 새 포스팅을 올리는 주체
- 로드밸런서(Load balancer): 트래픽을 웹 서버들로 분산한다.
- 웹 서버: HTTP 요청을 내부 서비스로 중계하는 역할을 담당한다.
- 포스팅 저장 서비스(post Service): 새 포스팅을 데이터베이스와 캐시에 저장한다.
- 포스팅 전송 서비스(fanout service): 새 포스팅을 친구의 뉴스 피드에 푸시(push)한다. 뉴스 피드 데이터는 캐시에 보관하여 빠르게 읽어갈 수 있도록 한다.
- 알림 서비스(notification service): 친구들에게 새 포스팅이 올라왔음을 알리거나, 푸시 알림을 보내는 역할을 담당한다.

뉴스 피드 생성

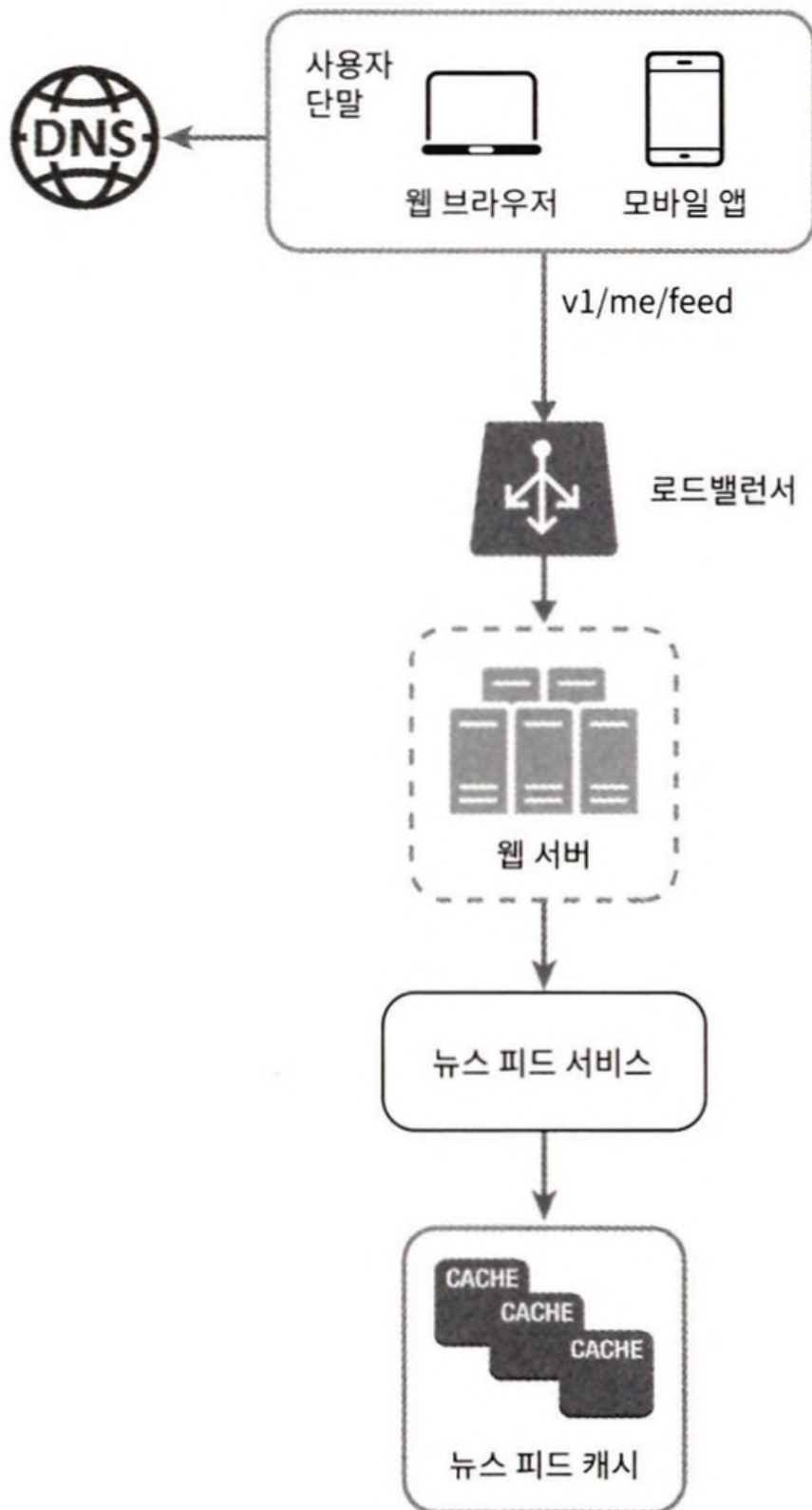


그림 11-3

- 사용자: 뉴스 피드를 읽는 주체
- 로드 밸런서: 트래픽을 웹 서버들로 분산
- 웹 서버: 트래픽을 뉴스 피드 서비스로 보낸다.
- 뉴스 피드 서비스(news feed service): 캐시에서 뉴스 피드를 가져오는 서비스다.
- 뉴스 피드 캐시(news feed cache): 뉴스 피드를 렌더링할 때 필요한 피드 ID를 보관한다.

상세 설계

피드 발행 흐름 상세 설계

- 컴포넌트는 개략적 설계안에서 다룬 정도로 충분할 것이다.
- 웹 서버와 포스팅 전송 서비스(fanout-service)에 초점을 맞추었다.

웹 서버

- 웹 서버는 클라이언트와 통신할 뿐 아니라 인증이나 처리율 제한 등의 기능도 수행한다.
- 올바른 인증 토큰을 Authorization 헤더에 넣고 API를 호출하는 사용자만 포스팅을 할 수 있어야 한다.
- 스팸을 막고 유해한 콘텐츠가 자주 올라오는 것을 방지하기 위해서 특정 기간 동안 한 사용자가 올릴 수 있는 포스팅의 수에 제한을 두어야 한다.

포스팅 전송(팬아웃) 서비스

- 포스팅 전송, 즉 팬아웃(fanout)은 어떤 사용자의 새 포스팅을 그 사용자와 친구 관계에 있는 모든 사용자에게 전달하는 과정이다.
- 팬아웃에는 두 가지 모델이 있는데 하나는 쓰기 시점에 팬아웃(fanout-on-write)하는 모델이고(push 모델이라고도 함)
- 읽기 시점에는 팬아웃(fanout-onread)하는 모델이다(pull 모델이라고도 함)

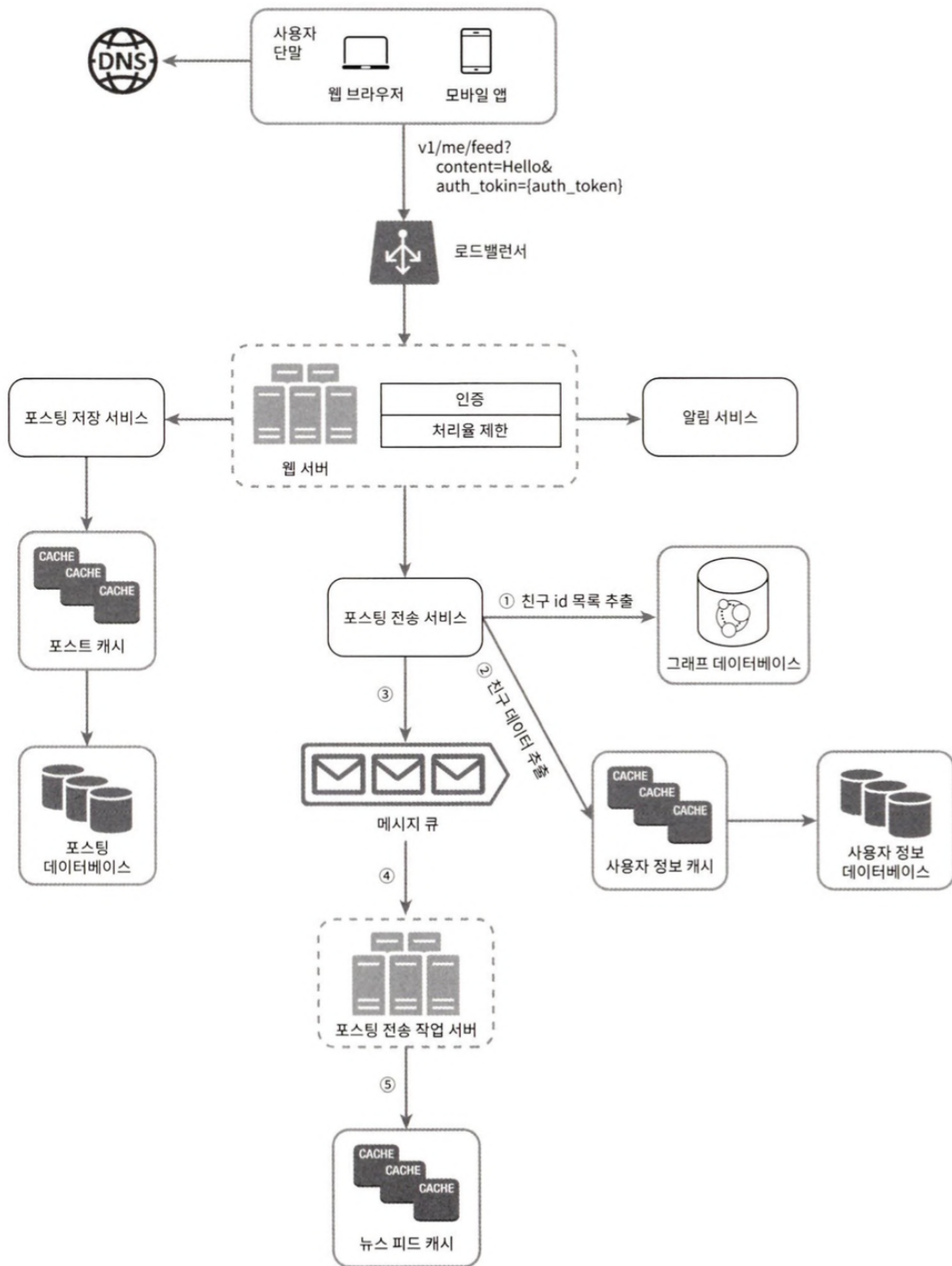


그림 11-4

• 쓰기 시점에 팬아웃 하는 모델

- 새로운 포스팅을 기록하는 시점에 뉴스 피드를 갱신하게 된다.

- 포스팅이 완료되면 바로 해당 사용자의 캐시에 해당 포스팅을 기록하는 것
- 장점:
 - 뉴스 피드가 실시간으로 갱신되며 친구 목록에 있는 사용자에게 즉시 전송 된다.
 - 새 포스팅을 기록하는 순간에 뉴스 피드가 이미 갱신되므로(pre-computed) 뉴스피드를 읽는 데 드는 시간이 짧아진다.
- 단점:
 - 친구가 많은 사용자의 경우 친구 목록을 가져오고 그 목록에 있는 사용자 모두의 뉴스 피드를 갱신하는 데 많은 시간이 소요될 수 있다. 핫키(hotkey) 라고 부르는 문제이다.
 - 서비스를 자주 이용하지 않는 사용자의 피드까지 갱신해야 하므로 컴퓨팅 자원이 낭비된다.

• 읽기 시점에 팬아웃하는 모델

- 피드를 읽어야 하는 시점에 뉴스 피드를 갱신한다.
- 따라서 요청 기반(on-demand) 모델이다.
- 사용자가 본임 홈페이지나 타임라인을 로딩하는 시점에 새로운 포스트를 가져오게 된다.
- 장점
 - 비활성화된 사용자, 또는 서비스에 거의 로그인하지 않는 사용자의 경우는 이 모델이 유리하다.
 - 데이터를 친구 각각에 푸시하는 작업이 필요 없으므로 핫키 문제도 생기지 않는다.
- 단점
 - 뉴스 피드를 읽는 데 많은 시간이 소요될 수 있다.

두 가지의 방법을 결합하여 장점을 취하고 단점을 버려보자

- 친구나 팔로어(follower)가 아주 많은 사용자의 경우에 팔로어로 하여금 해당 사용자의 포스팅을 읽은 시점에 팬 아웃을 이용하여 시스템 과부하를 방지할 것이다.
- 아울러 안정 해시(consistent hashing)를 통해 요청과 데이터를 보다 고르게 분산하여 핫키 문제를 줄여볼 것이다.

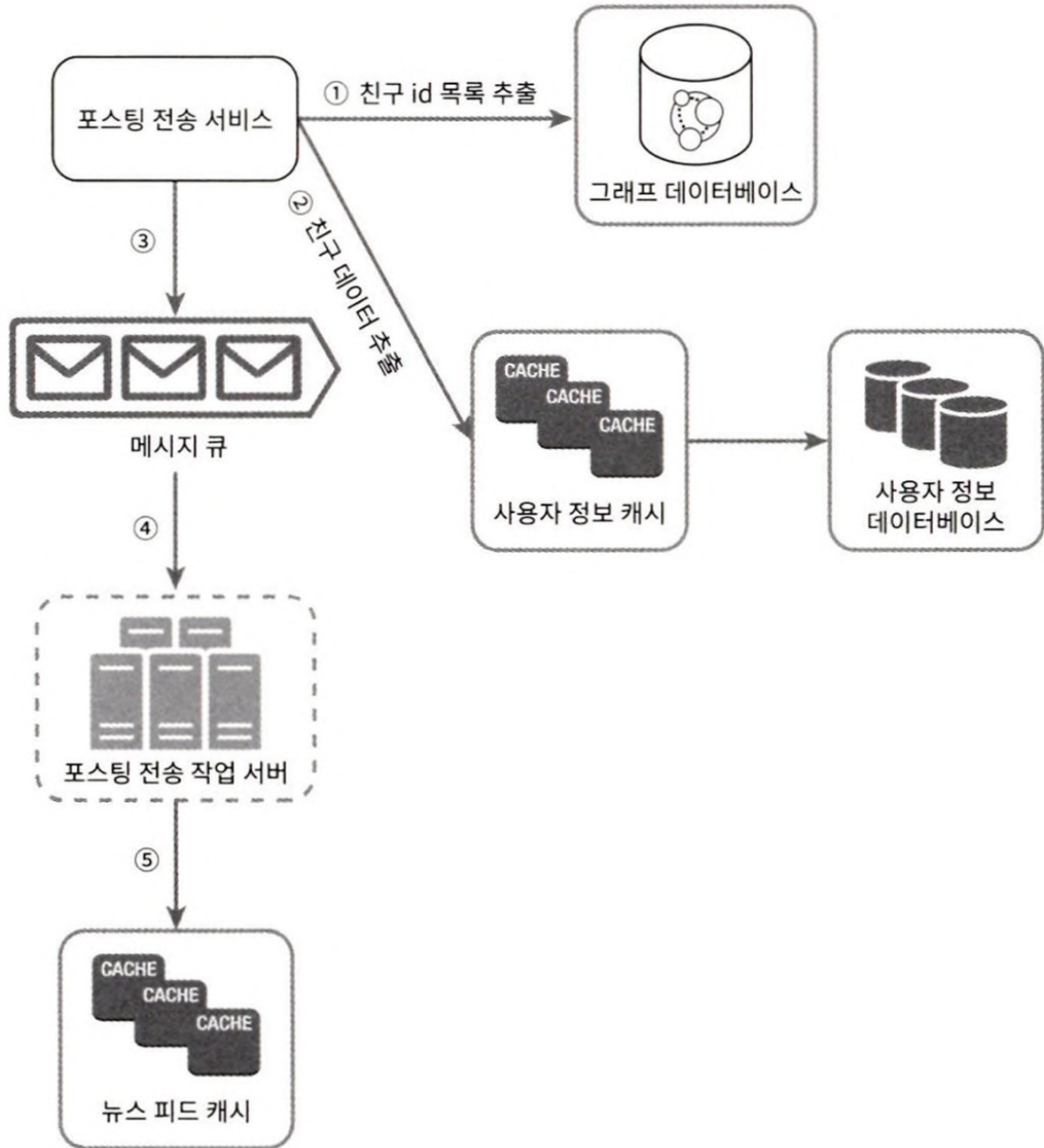


그림 11-5

• 팬아웃 서비스는 다음과 같이 동작함

1. 그래프 데이터베이스에 친구 ID 목록을 가져온다. (그래프 데이터베이스는 친구 관계나 친구 추천을 관리하기 적합)
2. 사용자 정보 캐시에서 친구들의 정보를 가져온다.
 - a. 사용자 설정에 따라 친구 가운데 일부를 걸러낸다. (mute한 사용자를 걸러냄)
 - b. 친구 관계는 유지하지만 새로운 뉴스 피드에 보이지 않아야 한다.
3. 친구 목록과 새 스토리의 포스팅 ID를 메시지 큐에 넣는다.

4. 팬아웃 작업 서버가 메세지 큐에서 데이터를 꺼내어 뉴스 피드 데이터를 뉴스 피드 캐시에 넣는다.

- 뉴스피드 캐시는 <post_id, user_id>의 순서쌍을 보관하는 매핑 테이블이다.
- 새로운 포스팅이 만들어질 때마다 이 캐시 매핑테이블에 레코드들이 추가될 것이다.
- 사용자 정보와 포스팅 정보 전부를 이 테이블에 저장하지 않는 이유는 메모리 요구량이 지나치게 늘 수 있기 때문이다.
- 메모리 크기를 적정 수준으로 유지하기 위해서, 이 캐시의 크기에 제한을 두며, 해당 값을 조정이 가능하도록 한다.

피드 읽기 흐름 상세 설계

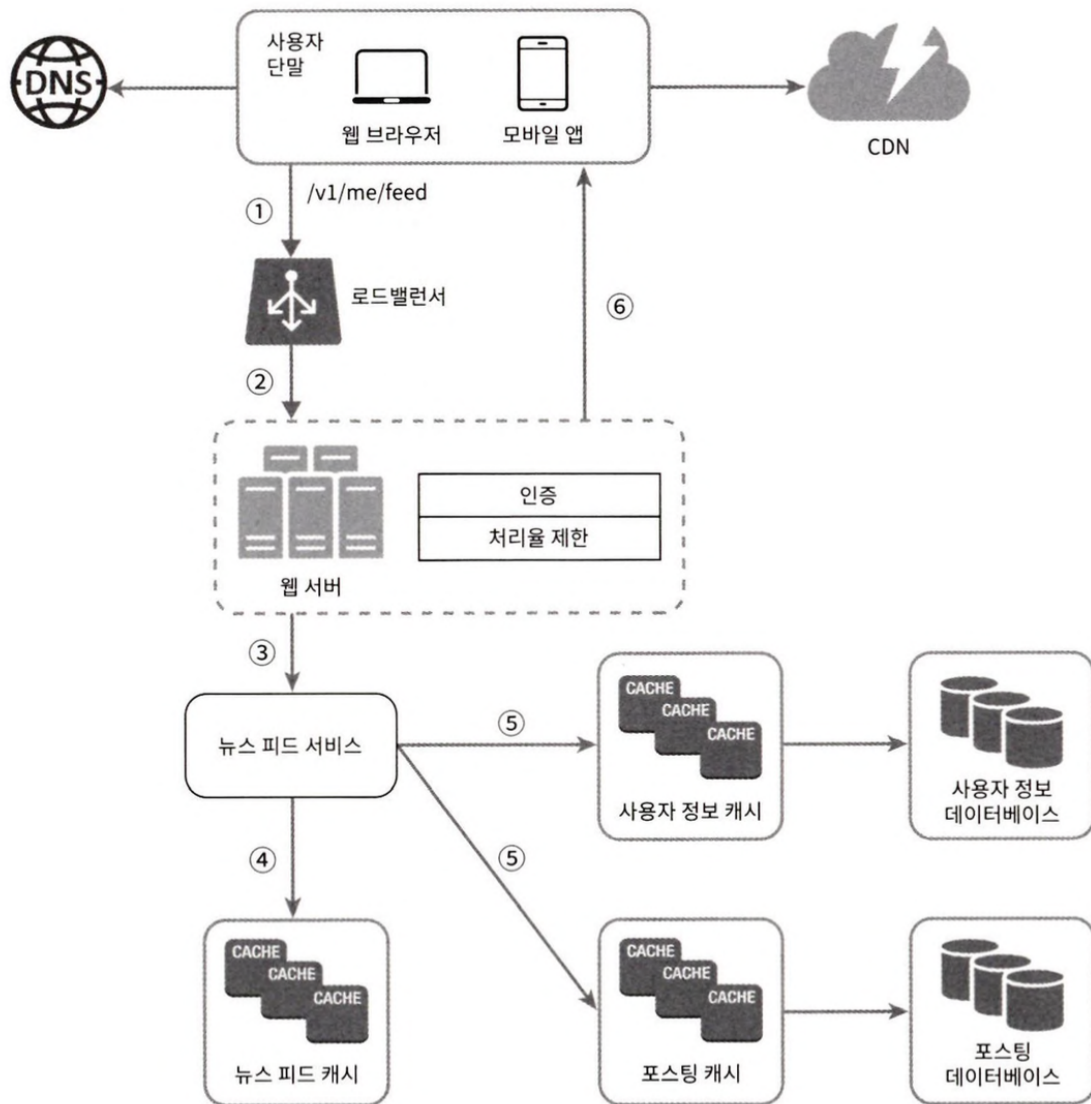


그림 11-7

- 이미지나 비디오 같은 미디어 콘텐츠는 CDN에 저장하여 빨리 읽어갈 수 있도록 하였다.
- 뉴스피드를 읽어 가는 단계 이다.
 1. 사용자가 뉴스 피드를 읽으려는 요청을 보낸다. → `/v1/me/feed`
 2. 로드밸런서가 요청을 웹 서버 가운데 하나로 보낸다.
 3. 웹 서버는 피드를 가져오기 위해 뉴스 피드 서비스를 호출한다.
 4. 뉴스 피드 서비스는 뉴스 피드 캐시에서 포스팅ID 목록을 가져온다.
 5. 뉴스 피드에 표시할 사용자 이름, 사용자 사진, 포스팅 콘텐츠, 이미지 등을 사용자 캐시와 포스팅 캐시에서 가져와 완전한 뉴스 피드를 만든다.

6. 생성된 뉴스 피드를 JSON 형태로 클라이언트에게 보낸다. 클라이언트는 해당 피드를 렌더링한다.

캐시 구조

- 캐시는 뉴스 피드 시스템의 핵심 컴포넌트다.



그림 11-8

- 뉴스 피드: 뉴스 피드의 ID를 보관한다.
- 콘텐츠: 포스팅 데이터를 보관한다. 인기 콘텐츠는 따로 보관한다.
- 소셜 그래프: 사용자 간 관계 정보를 보관한다.
- 행동(action): 포스팅에 대한 사용자의 행위에 관한 정보를 보관한다. 포스팅에 대한 ‘좋아요’, 답글 등등이 해당함
- 횟수(counter): ‘좋아요’ 횟수, 응답 수, 팔로어 수, 팔로잉 수 등의 정보를 보관한다.

마무리

- 다루면 좋을만한 주제는 아래와 같다.

데이터 베이스 규모 확장

- 수직적 규모 확장 vs 수평적 규모 확장
- SQL vs NoSQL
- 주-부(master-slave) 다중화
- 복제본(replica)에 대한 읽기 연산
- 일관성 모델(consistency model)
- 데이터베이스 샤딩(sharding)

논의해볼 주제

- 웹 계층(web tier)을 무상태로 운영
- 가능한 많은 데이터를 캐시할 방법
- 여러 데이터 센터를 지원할 방법
- 메세지 큐를 사용하여 컴포넌트 사이의 결합도 낮추기
- 핵심 메트릭(key metric)에 대한 모니터링
 - 트래픽이 몰리는 시간대의 QPS(Queries per Second), 사용자가 뉴스 피드를 새로고침(refresh) 할 때의 지연시간