11장 뉴스피드, 12장 채팅 시스템 설계

11장 뉴스피드

친구 id 목록 추출할 때 그래프 데이터베이스 사용. 그래프 데이터베이스의 장점은? 왜 쓰나

- 가장 큰 장점은 인덱스를 이용하지 않아도 연결된 노드를 찾는 것이 빠르다는 것이다.
    - 노드와 노드간의 관계를 이용해 인접한 관계를 찾는 기능인데 index free adjacency라고 합니다. 이처럼 관계를 이용한 정보를 탐색하는데 강력하고, 관계형 데이터베이스보다 관계를 표현하는데 있어 더 직관적이며, 왜곡 없이 표현할 수 있다.
    - RDBMS에서 10개 이상의 테이블을 조인하게 되면 테이블의 사이즈, 데이터양, 조인 순서들 많은 부분을 고려하여도, 성능저하가 발생하는 것을 막을수가 없지만, GraphDB는 이런 복잡한 연산을 처리하는데 적합한 그래프 이론을 알고리즘으로 채택하고 있다.
- 다른 NoSQL과 마찬가지로 스키마가 없는 구조로 되어 있으며, 반정규화된 데이터를 처리하는데 적합하다.
    - 노드는 RDBMS의 테이블과 비교할 수 있는데, 노드는 테이블이 행/열의 데이터를 가지고 있는 것과 같은 속성을 가지고 있다.
    - GraphDB에서 관계는 항상 시작과 끝점이 있는 방향을 가지고 있는데 , 자체 참조가 가능하여 시작과 끝이 동일 노드가 될 수도 있습니다. 관계는 명시적이며, 노드 처럼 속성을 가질수도 있습니다.
- 4~6개 이상의 테이블을 이용하는 **복잡한 질의**를 해야한다면 GraphDB가 좋은 대안이 될수 있다.
- 서로 다른 데이터들이 어떻게 연관되어 있는지 확인하는 경우에도 GraphDB는 강점을 보인다. 시작과 끝이 되는 노드만 알려주고 그래프 이론을 적용하기만 하면 되기 때문.
- 친구 관계나 친구 추천을 관리하기 적합함

뉴스 피드를 발행 과정

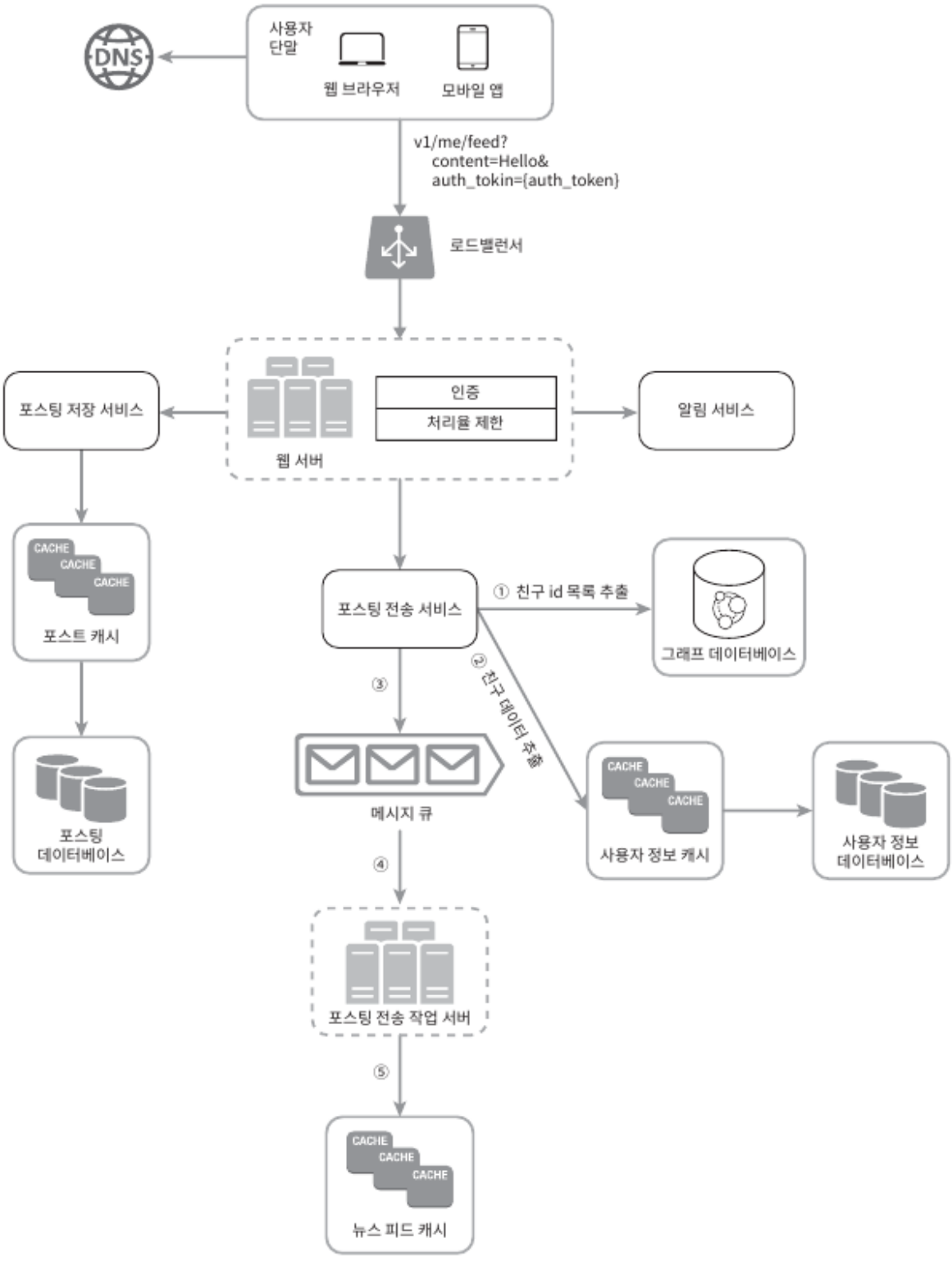

DNS
사용자
단말
웹 브라우저
모바일 앱
v1/me/feed?
content=Hello&
auth_tokin={auth_token}
로드밸런서
포스팅 저장 서비스
인증
처리율 제한
웹 서버
알림 서비스
CACHE
CACHE
CACHE
포스트 캐시
포스팅 전송 서비스
① 친구 id 목록 추출
그래프 데이터베이스
③
② 친구 데이터 추출
포스팅
데이터베이스
메시지 큐
CACHE
CACHE
CACHE
사용자 정보 캐시
사용자 정보
데이터베이스
④
포스팅 전송 작업 서버
⑤
CACHE
CACHE
CACHE
뉴스 피드 캐시

그림 11-4

1. 그래프 데이터베이스에서 친구 ID 목록을 가져온다.
2. 사용자 정보 캐시에서 친구들의 정보를 가져온다. mute 처리되어있다면 친구 일부를 걸러낸다.
3. 친구 목록과 새 스토리의 포스팅 ID를 메시지 큐에 넣는다.
4. 팬아웃 작업 서버가 메시지 큐에서 데이터를 꺼내어 뉴스 피드 데이터를 뉴스 피드 캐시에 넣는다.
    a. 뉴스 피드 캐시는 <포스팅 ID, 사용자 ID>의 순서쌍을 보관하는 매핑 테이블이다.
    b. 따라서 새로운 포스팅이 만들어질 때마다 레코드가 추가된다.
    c. 메모리 요구량이 지나치게 늘어나는 것을 방지하고 적정 수준으로 유지하기 위해 캐시의 크기에 제한을 둔다.
    d. 대부분의 사용자가 보려 하는 것은 최신 스토리이기 때문에 캐시 미스가 일어날 확률은 낮다.

> 1. **쓰기 시점에 팬아웃하는 모델**: 새로운 포스팅을 기록하는 시점에 뉴스 피드를 갱신. 포스팅이 완료되면 바로 해당 사용자의 캐시에 포스팅을 기록
>
> 장점
> - 뉴스피드가 실시간으로 갱신되며, 친구 목록에 있는 사용자에게 즉시 전송됨.
> - 새 포스팅이 기록되는 순간에 뉴스 피드가 이미 갱신되므로 뉴스 피드를 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 수도 있다. → **핫키**(hotkey)라고 부르는 문제다.
> - 서비스를 자주 이용하지 않는 사용자의 피드까지 갱신해야 하므로 컴퓨팅 자원이 낭비된다.

> 2. **읽기 시점에 팬아웃하는 모델**: 피드를 읽어야 하는 시점에 뉴스 피드를 갱신한다.
> 요청 기반(on-demand)모델이다. 사용자가 본인 홈페이지나 타임라인을 로딩하는 시점에 새로운 포스트를 가져오게 된다.
>
> 장점
> - 비활성화된 사용자, 또는 서비스에 거의 로그인하지 않는 사용자의 경우에는 이 모델이 유리하다. 로그인하기까지는 어떤 컴퓨팅 자원도 소모하지 않는다.
> - 데이터를 친구 각각에 푸시하는 작업이 필요 없으므로 핫키 문제도 발생하지 않는다.
>
> 단점
> - 뉴스 피드를 읽는 데 많은 시간이 소요될 수 있다.

이 책에서는 두 가지 방법을 결합하여 장점은 취하고 단점은 버리는 전략을 취했다.

- 뉴스 피드를 빠르게 가져오는 것은 아주 중요하므로 대부분의 사용자에 대해서는 푸시 모델을 사용한다.
- 팔로어가 아주 많은 사용자의 경우에는 팔로어로 하여금 해당 사용자의 포스팅을 필요할 때 가져가도록 하는 풀 모델을 사용하여 시스템 과부하를 방지한다.
- 안정 해시(consistent hashing)을 통해 요청과 데이터를 보다 고르게 분산하여 핫키 문제를 줄여본다.

뉴스 피드를 읽는 과정

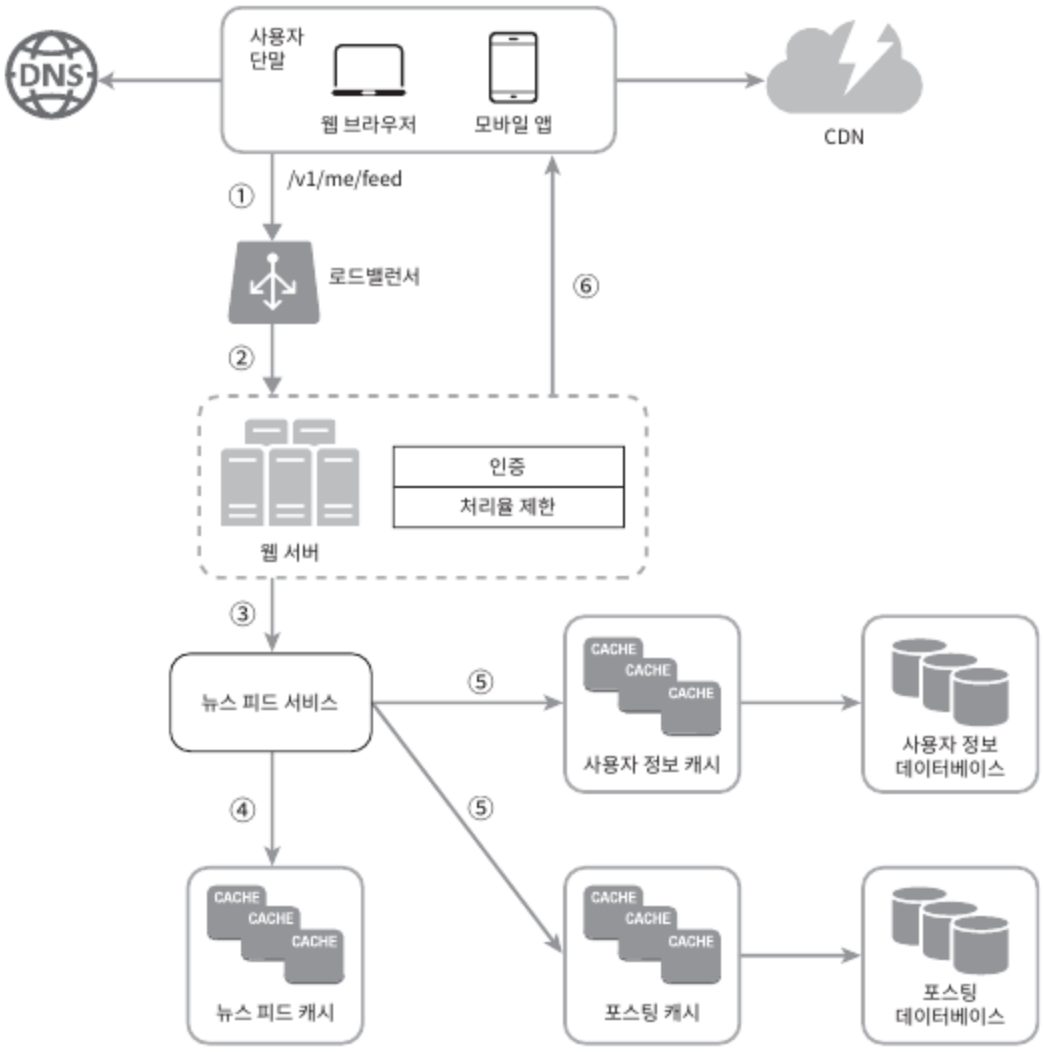


그림 11-7

12장 채팅 시스템

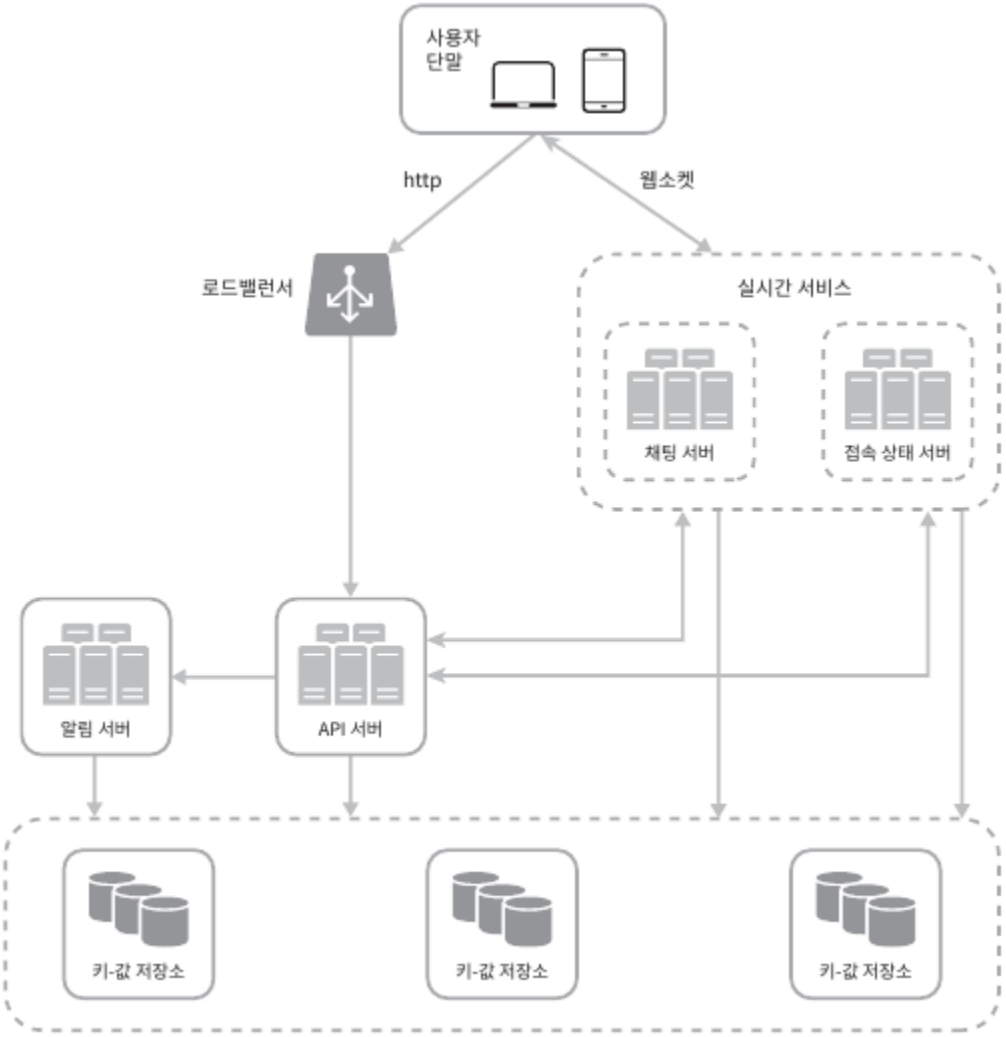
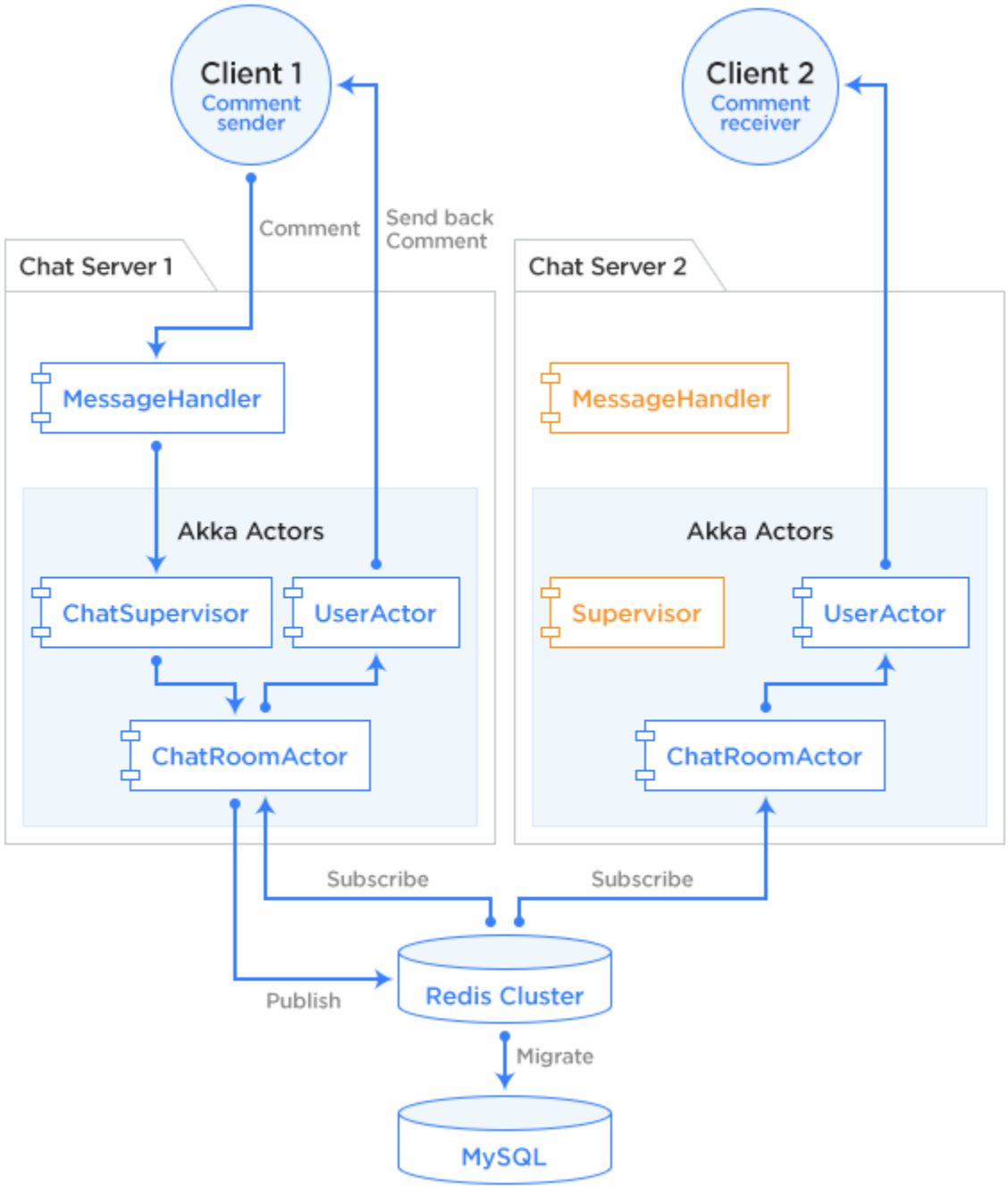
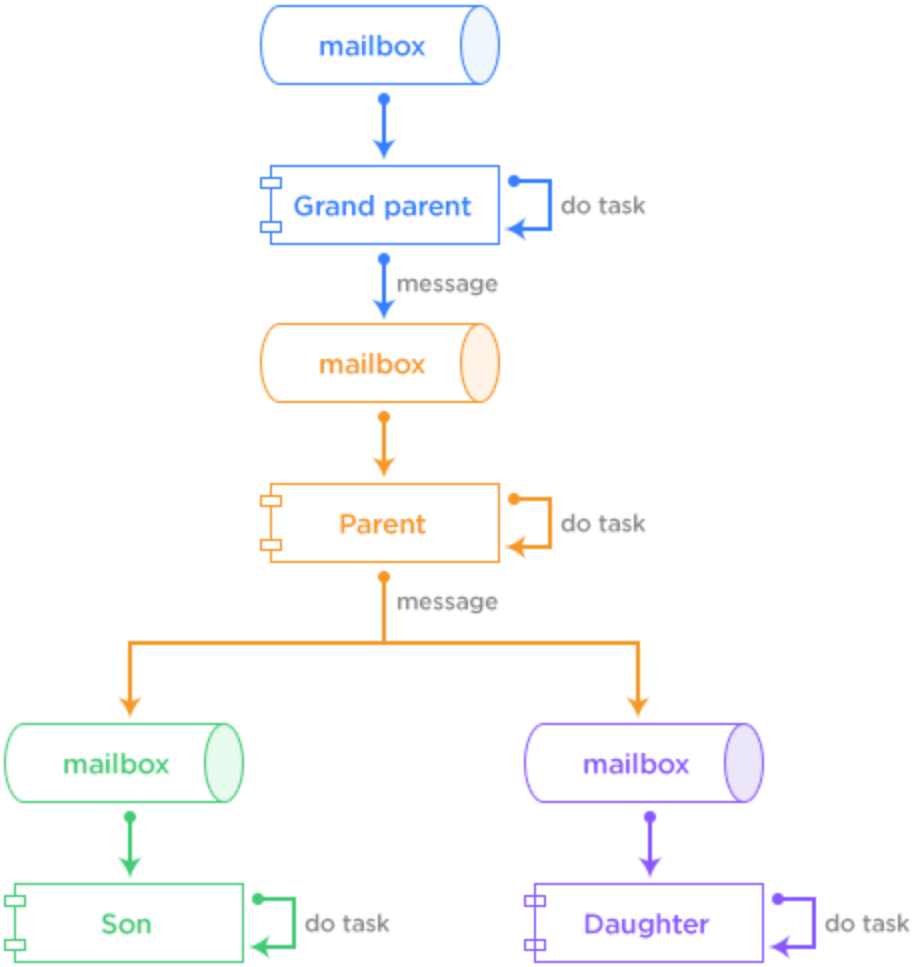


그림 12-8

LINE 채팅 아키텍처


Client 1
Comment sender
Client 2
Comment receiver
Comment
Send back Comment
Chat Server 1
Chat Server 2
MessageHandler
MessageHandler
Akka Actors
Akka Actors
ChatSupervisor
UserActor
Supervisor
UserActor
ChatRoomActor
ChatRoomActor
Subscribe
Subscribe
Publish
Redis Cluster
Migrate
MySQL



참고

- https://rastalion.me/graph-db-그래프-데이터베이스/
- https://engineering.linecorp.com/ko/blog/the-architecture-behind-chatting-on-line-live/