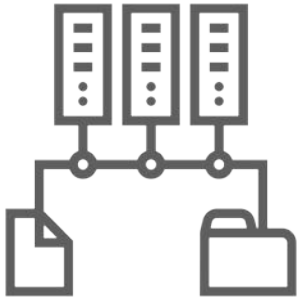
Winter://Dev.Camp 최종 발표

Healer : 조영호 홍성문 정명지

목차 Contents

프로젝트 목표

프로젝트 목표



대용량 서버 아키텍처

Micro Service Architecture의 이해
Scale Out 가능한 구조의 설계 및 구현

채팅 및 메신저 구조 이해

실시간 채팅

완성도 있는 프로젝트

메신저 기능에 대한 완전한 서비스 제공

프로젝트 기획

프로젝트 기획

Web kakao

발자취..

메신저..?

파일전송

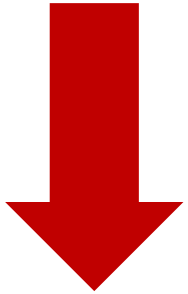
인증

채팅

채팅방

친구

채팅방, 유저 관리, 파일...



CRUD의 반복..!

채팅은?

실시간 대화

푸시 알림

채팅


지속적인 트래픽 소모
Polling
느린 반응성
실시간 채팅의 구현
Web Socket
실시간 반응
연결 유지 및 관리

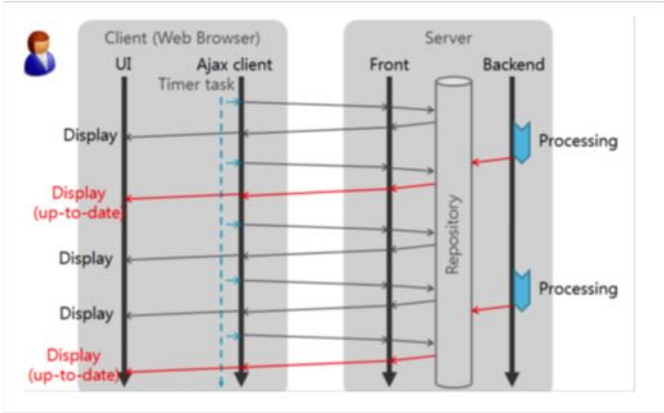
채팅

푸시알람

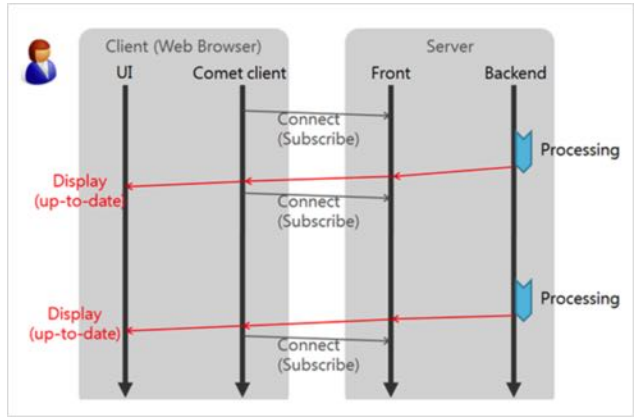
WebSocket?

채팅방의 증가에 따른 부하

채팅



“ 비동기 응답의
관리 이슈 ”



Polling

채팅방의 증가에 따른 부하

Long Polling

비동기 응답을 통한 트래픽 낭비의 최소화

Polling에 비해 빠른 반응성

기능 및 시연

Healer: Kakao

localhost

kakao

hsm63746244@gmail.com

••••

LOG IN

SIGN UP

Reset password

Healer: Kakao

localhost

Friends

조영호
상태메세지 입니다.

홍성문
홍성문의 상태메세지

정명지
Healer팀의 WebKakao

테스터1
상태 메세지6

테스ㅌ
상태 메세지4

테스터2
상태 메세지5

테스
상태 메세지6

채팅 서버

WebSocket 채팅서버

고민

진짜 실시간 채팅 - 제약 : 브라우저

실시간 읽은 사람 수 확인하기

Scale Out !!

WebSocket 채팅서버

분석

WebSocket

진짜 실시간
연결 이후 TCP통신
최소화된 패킷

Polling

Fake 실시간..
연속되는 HTTP콜
계속되는 요청
→ **서버부담**

WebSocket 채팅서버

분석

연결 지향적

WebSocket

진짜 실시간
연결 이후 TCP통신
최소화된 패킷

“

여러 대의 서버가 있을때
누가 어디에 연결되어 있는지 모르더라도
채팅메시지를 전달할 방법이 없을까?

”

WebSocket 채팅서버

결론

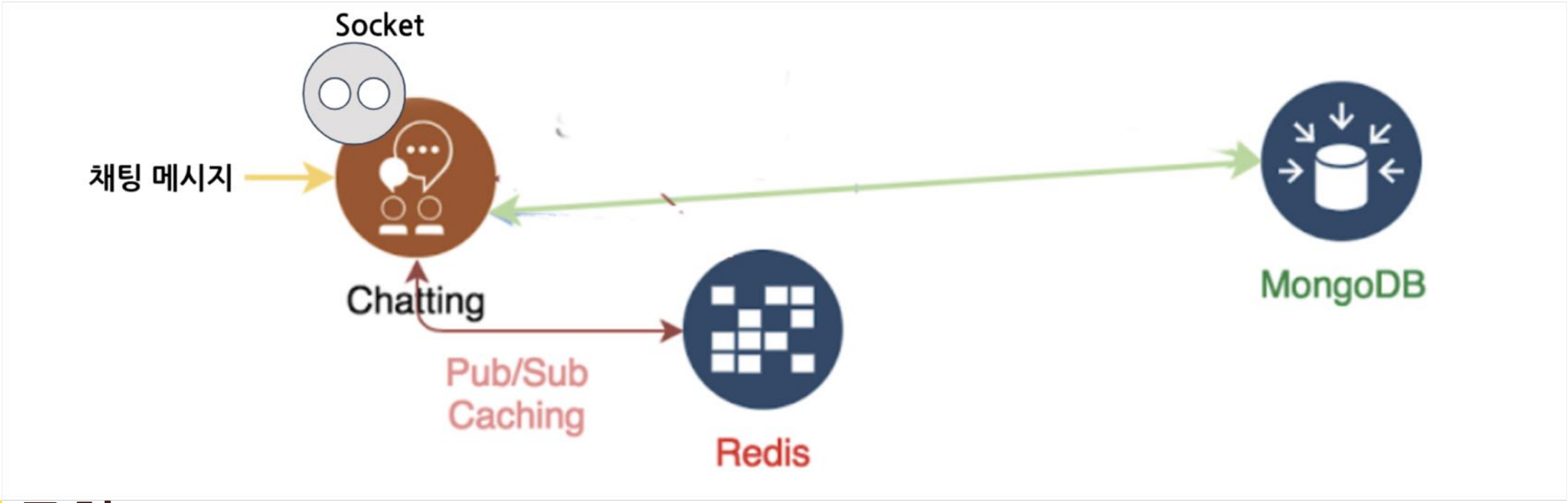
+

redis

Pub/Sub

연결에 대한 의존을 줄임 → Scale Out!

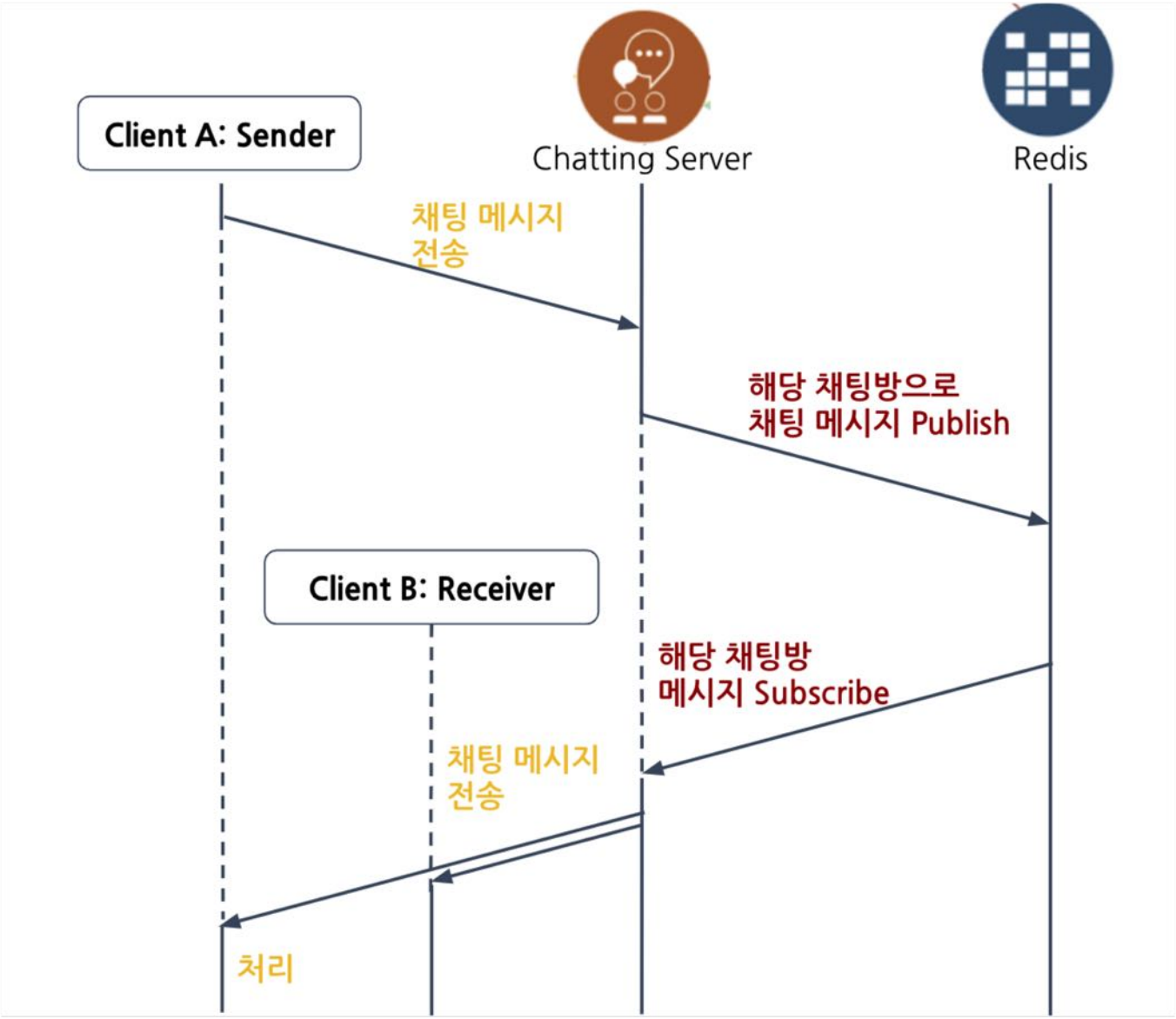
WebSocket 채팅서버



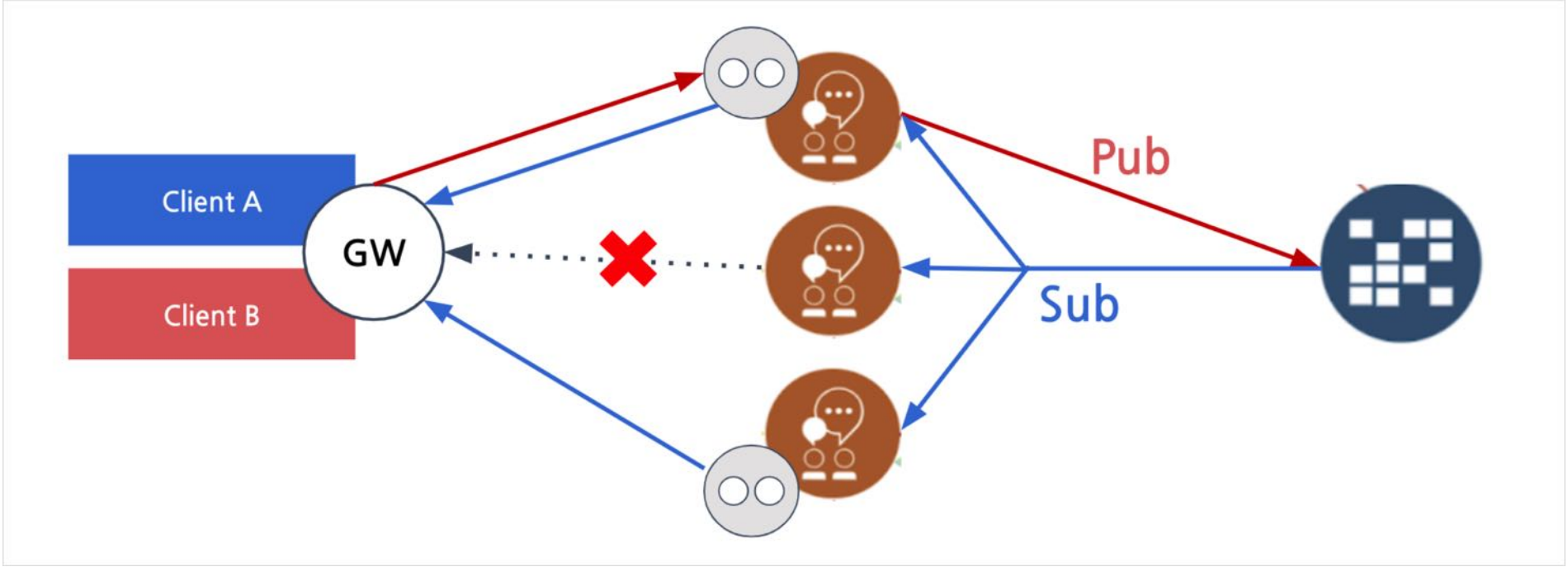
구현

접속(Subscribe)
실시간 메시지 전송 / 실시간 읽음 확인
접속 시, 현재 참여자 정보 전송

WebSocket 채팅서버



WebSocket + Redis Pub/Sub



장점

실제 연결된 클라이언트에게만 메세지를 전달하기 때문에
SCALE OUT하기 좋습니다.

STOMP ??

고민

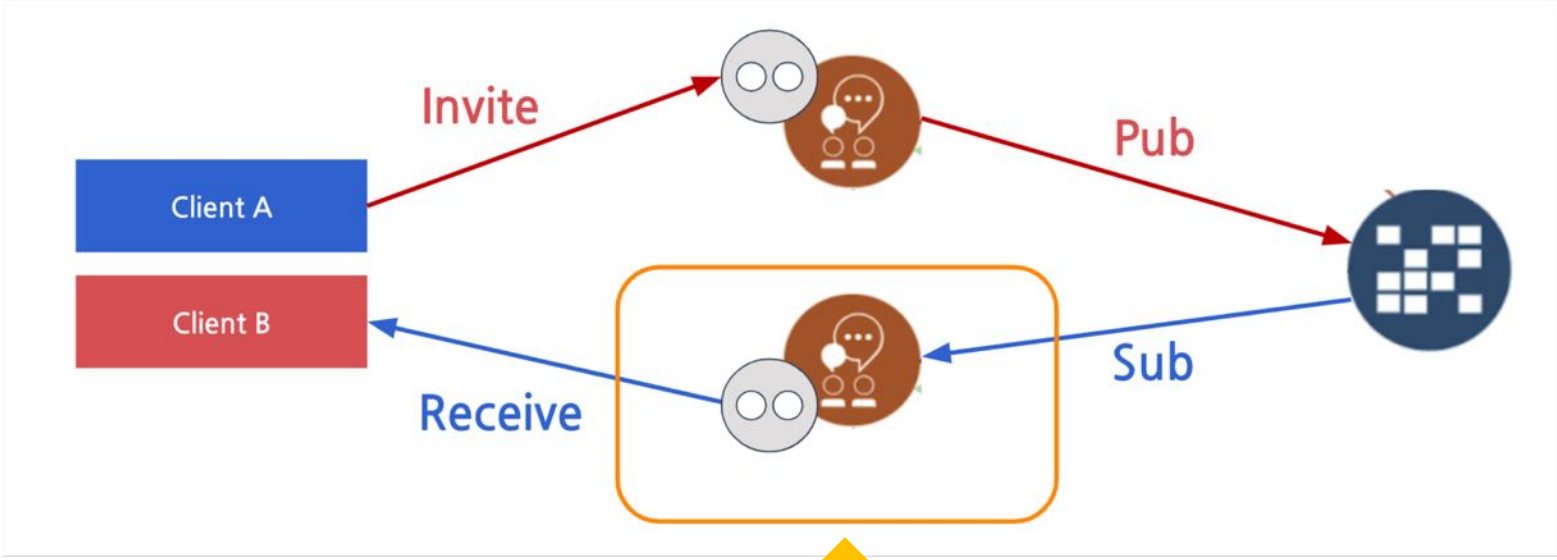
“

실시간 채팅중이지 않을 때에도
내가 속한 **모든 채팅방을 감시(Subscribe)**하고 있어야 할까?

”

WebSocket : 새로운 채팅방 초대받는다면

분석

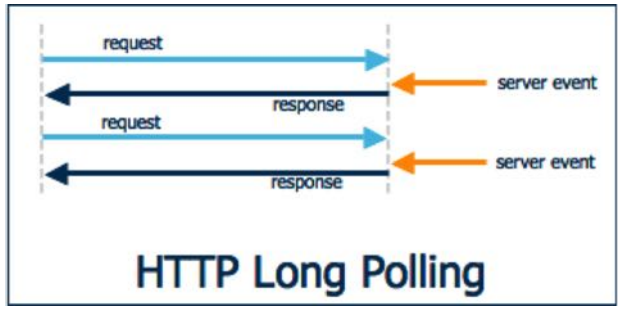


초대 전용 Topic을 모든 사용자가 sub해야한다.

WebSocket : 새로운 채팅방 초대받는다면

결론

+

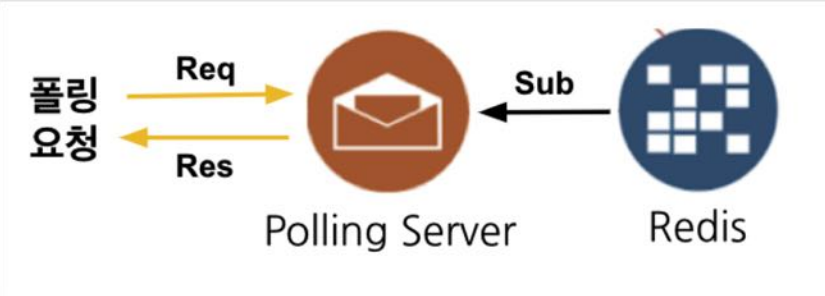


실시간 채팅

채팅중이지 않을 때,
모든 방에 대한 새 채팅 푸쉬

트래픽 분산

Long Polling 채팅서버



목적

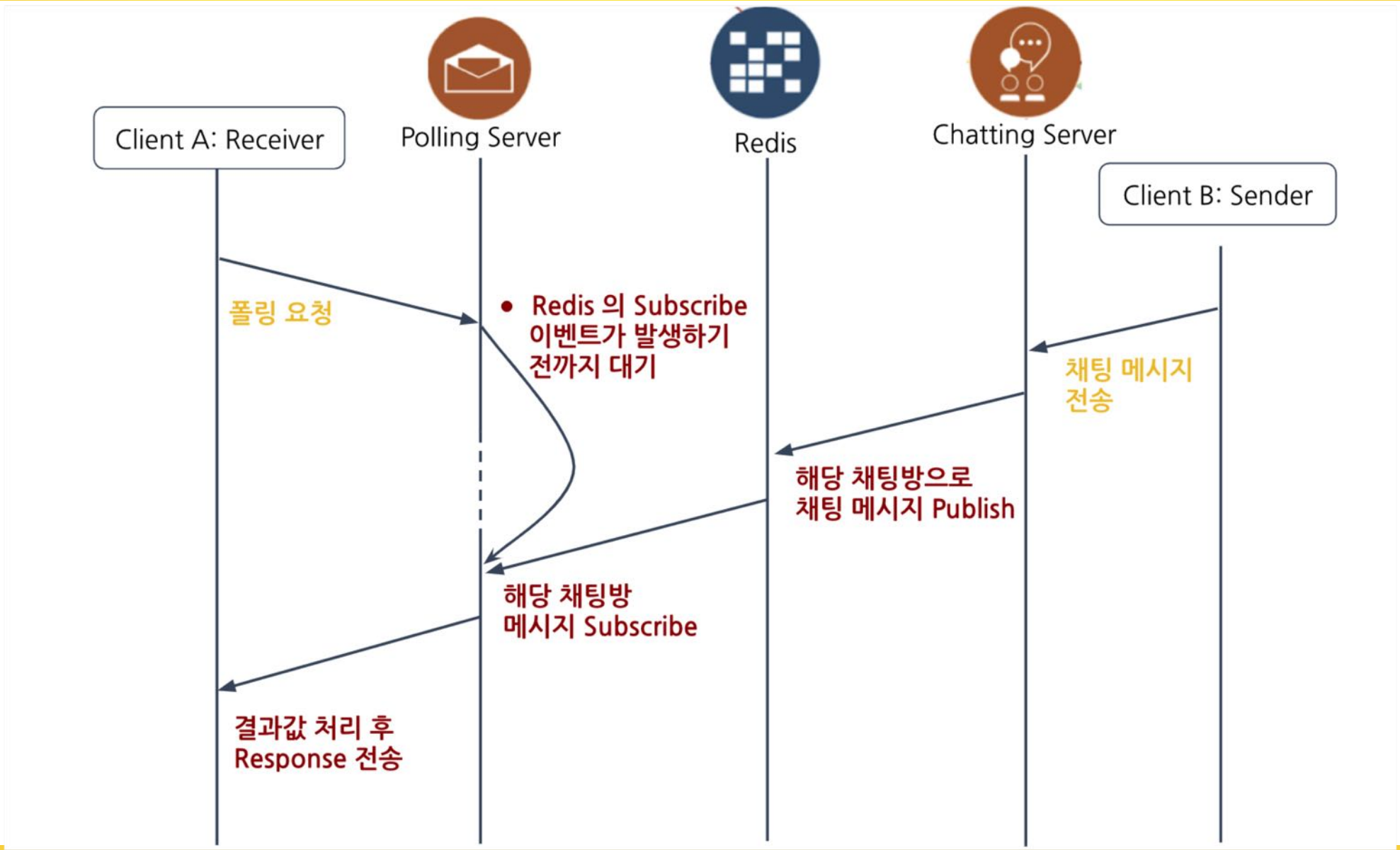
채팅방 리스트에서의 새로운 메시지 수신
초대 수신

장점

초대받은 사람의 Request Object를 찾아 채팅방 초대메시지를 보내고
그 사람에게만 Response를 전달하여 서버 부담을 최소화!

➡ **비동기 응답 객체 하나**를 통해 효과적으로 관리

Long Polling 채팅서버



실시간 읽기가 가능할까?

STOMP를 최대한 활용

Connect
Subscribe
Unsubscribe
Disconnect

발 생 시 점

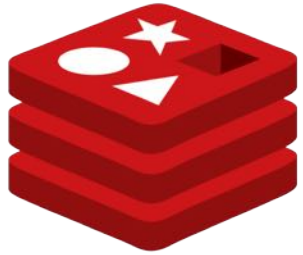
로그인시

채팅방 입장시
채팅방 퇴장시
로그아웃, 세션 종료시

Stomp 활용

입장 / 퇴장시
채팅방에 메시지 전송!

접속 중이지 않은 사람은?



redis

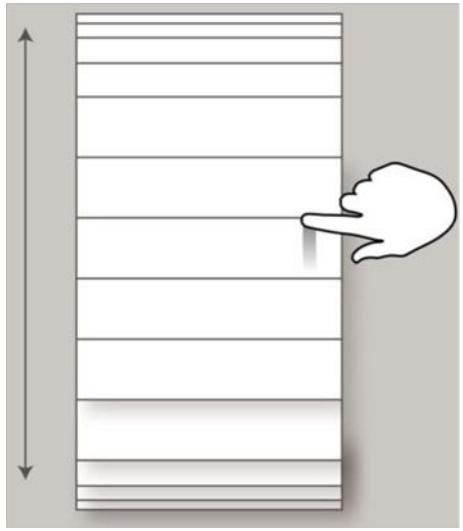
: 사용자별 어디까지 읽었는지 메시지 인덱스 저장!

메시지는
어디에 저장할까?

채팅메시지는 어디에 저장할까?

고민

웹브라우저에
모든 메시지를 저장할 순 없다!

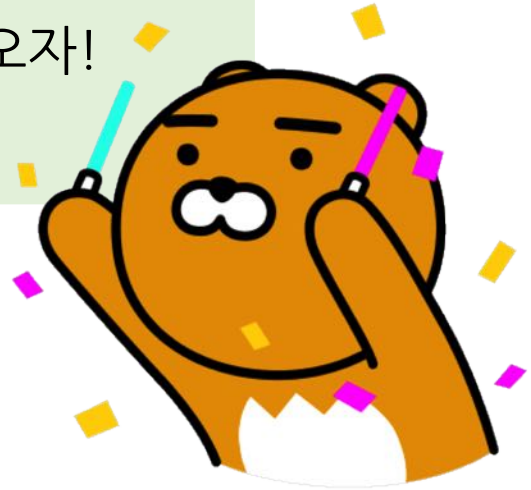


스크롤로
과거 메시지 조회

목표

“ 채팅 메시지 ”

최대한 효율적이고 빠르게 가져오자!



효과적인 저장방법

분석

1. Row에 하나의 메시지를 넣으면
   여러 개의 메시지 가져오기 불리함
2. 잦은 DB Insert로 인한 성능 하락 예상

효과적인 저장방법

분석

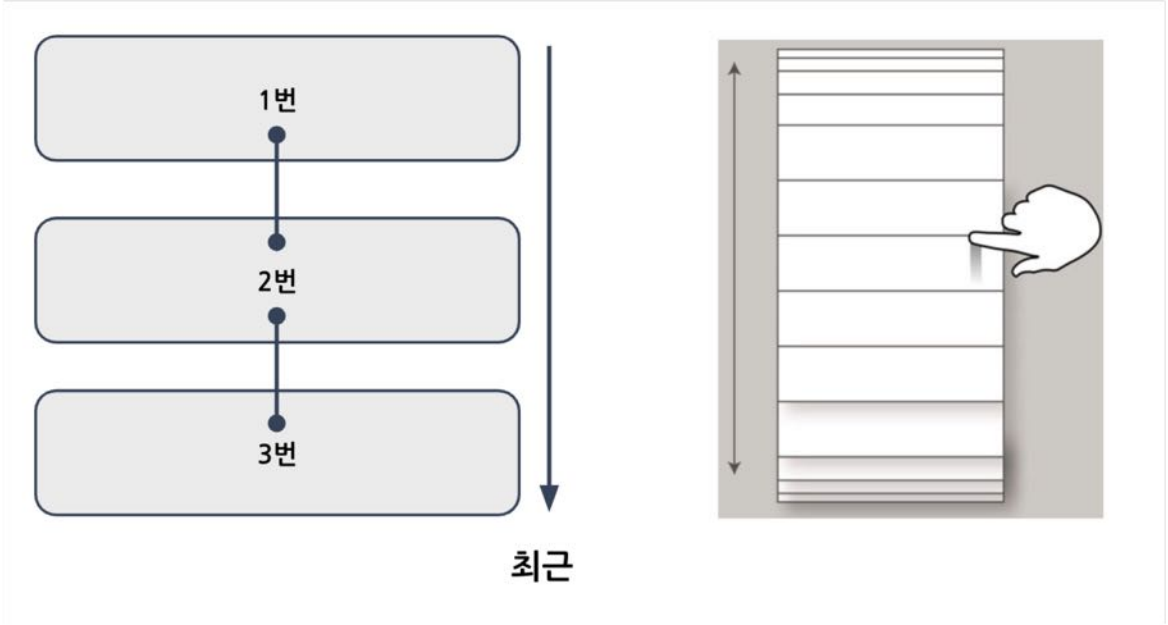
선 택

1. 메시지들을 덩어리로 묶어서 저장 가능!
2. 잦은 DB Insert에 최적화

효과적인 저장방법

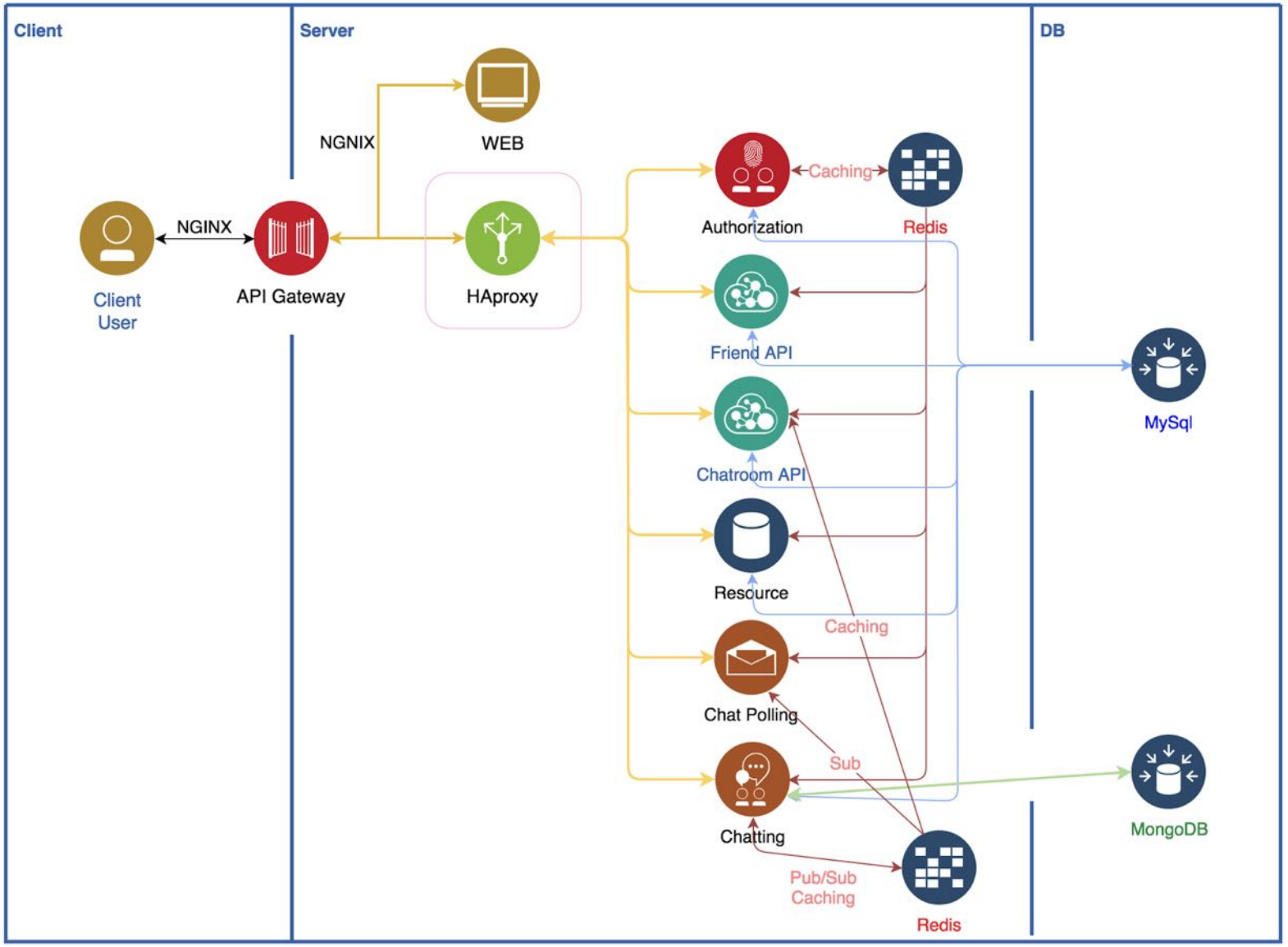
결론

“ 메시지를 덩어리로 넣어 ”
시간순으로 연결하자!



아키텍처

아키텍처 구조


Client
Server
DB
NGNIX
WEB
NGINX
Client
User
API Gateway
HAproxy
Caching
Authorization
Redis
Friend API
MySql
Chatroom API
Resource
Caching
Chat Polling
Sub
Chatting
MongoDB
Pub/Sub
Caching
Redis

프로젝트를 마치고..

캠프 소감

Healer 홍성문

프로젝트를 시작하며 Front End에 대한 두려움이 있었지만 좋은 팀원을 만나 서로에게 많은 것들을 배울 수 있는 좋은 기회가 되었다. 약 두 달간 짧게 진행되었던 프로젝트이지만 정말 재미있게 진행하였고 경험이 되었다.

React와 CSS에 좀더 친숙해지는 시간이었고,
처음 개발해보는 Spring Boot에 조금 자신감이 생기는 계기
설계와 성능에 중점을 둔 진짜 개발자가 되어가는 과정을 겪었다.

Healer 조영호

Healer 정명지

아키텍처의 전반적인 이해와 설계를 통해서 서버는 API를 짜는 것이 다가 아님을 다시 한번 느낄 수 있는 기회였다. 앞으로 서버를 만들 때, 아키텍처를 고려하면서 설계할 것 같다. 그리고, 대단한 팀원들을 만나서 많이 배울 수 있는 시간이었다.

감사합니다 ☺
모두들 수고 많으셨어요!