시스템 설계 면접 공략법

신입일때 (전사공통)

회사 문화와 정책에 따라 다르다.

이부분은 현 스터디와 무관하므로 노코멘트

경력일때 (팀실무자 채용)

다음과 같이 나눠서 면접 방식 및 난의도를 구성한다.

- 3년차 이하 (주니어)
- 3년차 ~ 9년차 (가장 선호하는 중니어?)
- 10년차 이상 (시니어)

경력일때 (팀실무자 채용)

경력사항 질문

- 프로젝트에서 진짜로 내가 한것은 무엇인가?
  (명확한 RNR – Role & Responsibility)
- 나는 그문제 해결을 위해서 구체적으로 무엇을 했는가?
  (기술경험 확인)
- 그 상황에서 이러한 문제가 추가로 발생한다면 어떻게 해결하겠는가?
  (누적된 경험을 기반으로 순간적인 문제해결능력 확인)

컴퓨터 사이언스 질문

- 전반적인 전산학 관련 질문 (기본기 확인)
- 특정 기술셋에 대한 질문 (우리 회사의 기술셋과 핏이 맞는지 확인)
- 최신 기술 트랜드 질문 (계속 학습을 하고 있는지 확인)

코딩테스트

- 기본적인 알고리즘 구현능력 (경력은 상대적으로 낮은 비중)
- 코딩스타일 확인
- 테스트구현 및 부하테스트 경험 확인
- Git 활용 능력 및 PR 기반의 협업 경험 확인
- 코드리뷰 경험 확인

가상의 시스템 설계

- 특정 분야의 시스템 아키텍처를 디자인하고 구현 할 수 있는 능력이 있는지 확인
- 구간별 SPOF (single point of failure) 회피 능력 확인
- 서비스 장애에 따른 트러블슈팅 능력 확인
- 서비스 규모에 따른 아키텍처 구현 능력 확인
  (오버엔지니어링 방지)
- 비용 (각종 리소스)을 최소화한 아키텍처 구성 능력 확인
  (효율적인 Trade-Off)

경력일때 (팀실무자 채용)

경력사항 질문

- 프로젝트에서 진짜로 내가 한것은 무엇인가?
  (명확한 RNR
- 나는 그문제 ...가?
  (기술경험 확
- 그 상황에서 ...게 해결하겠는가?
  (누적된 경험을 기반으로 분산적인 문제해결능력 확인)

포장 가능
입개발 가능

컴퓨터 사이언스 질문

- 전반적인 전
- 특정 기술셋 ...이 맞는지 확인)
- 최신 기술 트 ...확인)

입개발 가능

코딩테스트

- 기본적인 알고리즘 구현능력 (경력은 상대적으로 낮은 비중)
- 코딩스타
- 테스트구
- Git 활용
- 코드리뷰

학습에 의한 통과 가능

가상의 시스템 설계

- 특정 분야의 시스템 아키텍처를 디자인하고 구현 할 수 있는
  능력이 있
- 구간별 SP ...확인
- 서비스 장
- 서비스 규
  (오버엔지
- 비용 (각종 리소스)을 최소화한 아키텍처 구성 능력 확인
  (효율적인 Trade-Off)

정답이 없음
경험이 중요함

가상의 시스템 설계

- **시스템 설계 면접은 당황스러울 때가 많다.**
  - 널리 알려진 제품 X를 설계해 보라
  - 모호하고 범위도 지나치게 넓다.

- **문제를 정확히 이해하고 구체적으로 설계범위를 확정해야 한다.**
  - 구체적으로 어떤 기능을 구현해야 하는가?
  - 사용자 수는 얼마인가?
  - 서비스 규모는 얼마나 빨리 커지리라 예상하는가? (6개월, 1년, 3년 뒤의 규모는 얼마로 예상되는가?)

- **시스템 설계 면접은 면접관과 면접자가 모호한 문제를 풀기위해 협력하여 그 해결책을 찾아가는 과정이다.**
  - 이 문제의 대부분은 정해진 정답이 없다.
  - 면접자의 설계 능력을 시연하는 자리이다.
  - 설계 과정에서 내린 다양한 결정들에 대해 면접관을 설득하는 자리이다.
  - 또한, 면접관의 피드백을 건설적인 방식으로 수용할 수 있음을 선보이는 자리이다.

시스템 설계를 위한
개략적인 규모 추정

> ***개략적인 규모 추정(back-of-the-envelope estimation)****은 보편적으로 통용되는 성능 수치상에서 사고 실험(thought experiments)를 행하여 추정치를 계산하는 행위로서, 어떤 설계가 요구사항에 부합 할 것인지 보기 위한 것이다.*

미국의 프로그래머이자 컴퓨터 과학자. 1999년, 구글에 입사한 이래로 수 많은 인재들이 모이는 구글에서도 손꼽히는 전설적인 프로그래머 중 한 명으로 통한다. 구글리서치[2]와 구글AI[3]의 SVP를 맡으며 실질적 총괄자 역할을 맡고 있으며 시니어 펠로우 직급이기도 하다.[4] 2023년에 구글 딥마인드[5]와 구글브레인 팀이 통합된 이후에는 구글의 수석과학자로 임명되며 인공지능 전반을 책임지고 있다.[6]

그의 행적은 그야말로 구글의 엔지니어링 그 자체라고 할 수 있다. 텐서플로우, 맵리듀스, 빅테이블, 스패너 등의 핵심적인 프로젝트를 진두지휘했으며, 구글 검색, 구글 번역, 구글 광고의 초창기 버전 역시 그의 손에서 태어났다. 그리고 2012년 경 딥러닝의 잠재성을 파악한 뒤 앤드류 응 등과 함께 전 세계 탑 레벨의 인공지능 인재들을 끌어모아 일찍이 구글 브레인을 설립하는 등 구글이 AI 업계를 선도하는 데에 큰 역할을 했다. 현 시점 임원급의 엔지니어 상당수는 그가 데려온 사람들이기도 하다.

이렇게 놀라운 능력에 존경을 보내고자 구글러들은 척 노리스에 대한 사실들을 패러디해 제프딘에 대한 사실들을 정리하기에 이러렀을 정도다. 아래는 그 목록 중 몇 가지이다.

- 제프 딘의 PIN은 원주율의 마지막 네 자리이다.[7]
- 2002년 인덱싱 서버가 죽었을 때, 2시간 동안 제프 딘이 직접 유저의 검색에 응대했다. 그 시간 동안 검색 품질 평가 지수가 5점 올랐다고 한다.
- 구글 인사 시스템의 최대 레벨이 10일 때 제프 딘은 11로 승진했다. (실제 일어난 일)
- 제프 딘은 코드를 커밋하기 전에 컴파일과 실행을 해보는데, 이는 그저 CPU와 컴파일러에 버그가 있나를 확인하기 위함일 뿐이다.
- 제프 딘이 스탠퍼드 대학교에 강의를 갔을 때 사람들이 너무나도 많이 몰려와서 도널드 커누스 교수조차 땅바닥에 앉아서 봐야 했다. (실제 일어난 일)
- 제프 딘의 이력서에는 그가 아직 하지 않은 일들이 적혀 있다. 그 편이 더 짧기 때문이다.
- 컴파일러는 제프 딘에게 경고를 하지 않는다. 제프 딘이 컴파일러에게 경고를 한다.
- 신이 "빛이 있으라"라 말했다. 제프 딘은 코드 리뷰를 해주기 위해 그 옆에 있었다.
- 어느 날 구글 서버들의 생산성이 갑자기 두 배로 늘어났다. 서버를 모니터링하던 직원이 그 이유를 궁금해했는데, 제프 딘의 책상 옆을 지나다 그 이유를 알게 되었다. 그의 키보드 USB가 1.0에서 2.0으로 업그레이드된 것이다.[8]

https://namu.wiki/w/%EC%A0%9C%ED%94%84%20%EB%94%98

규모 확장성을 표현하는 수치들

개요

- 어떤 설계가 요구사항에 부합하는지 고민하기 위해 시스템 용량이나 성능 요구사항을 개략적으로 추정해야 한다.
- 개략적인 규모추정 (back of the envelope estimation)을 위해서는 규모 확장성을 표현하는 수치들에 익숙해져야 한다.

보편적으로 사용되는 항목

- 2의 제곱수
- 응답지연 (Latency)
- 고가용성 (HA - High Ability)
- 처리율 (TPS – Transaction per second, QPS – Query per second)

2의 제곱수

2의 x 제곱	근사치	이름	축약형
10	1천(thousand)	1킬로바이트(Kilobyte)	1KB
20	1백만(million)	1메가바이트(Megabyte)	1MB
30	10억(billion)	1기가바이트(Gigabyte)	1GB
40	1조(trillion)	1테라바이트(Terabyte)	1TB
50	1000조(quadrillion)	1페타바이트(Petabyte)	1PB

표 2-1

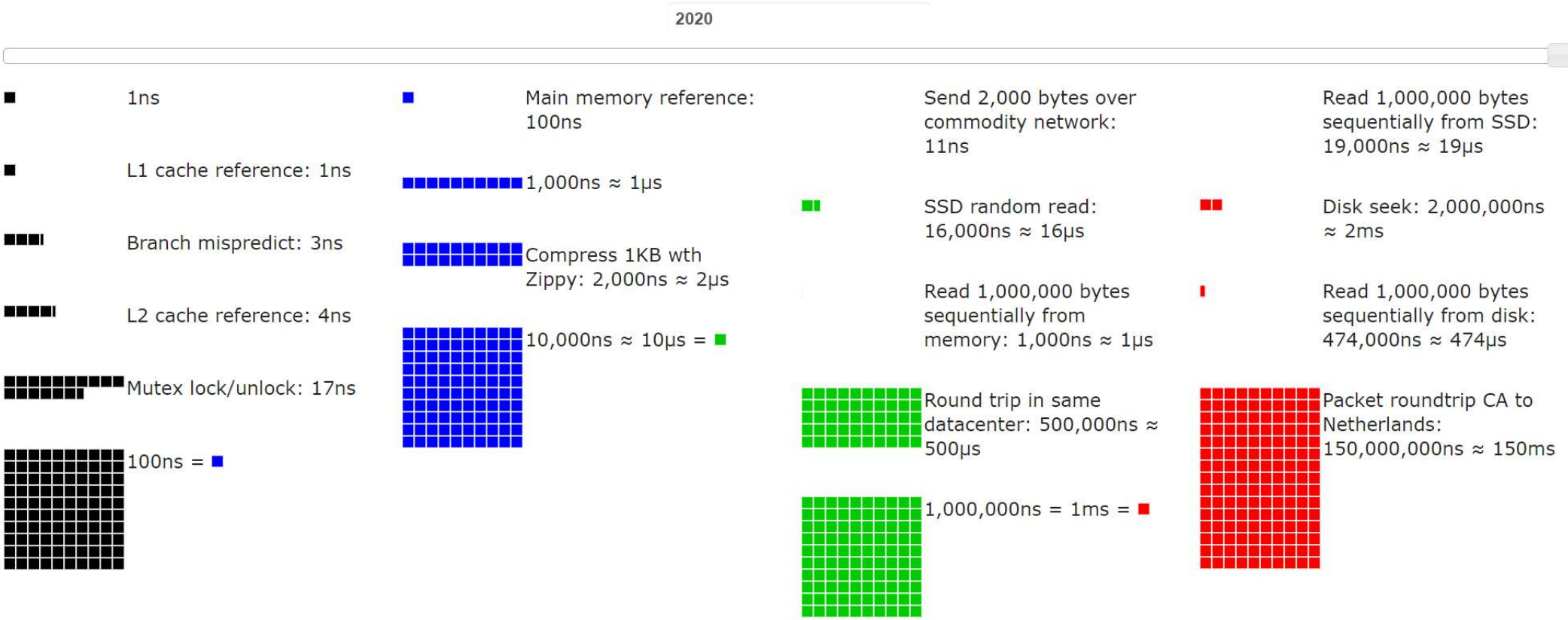
응답지연 값

- 1ns = 10^-9로서 1나노초인 10억분의 1초
- 1μs = 10^-6로서 1마이크로초인100만분의 1초
- 1ms = 10^-3로서 1밀리초인1000분의 1초

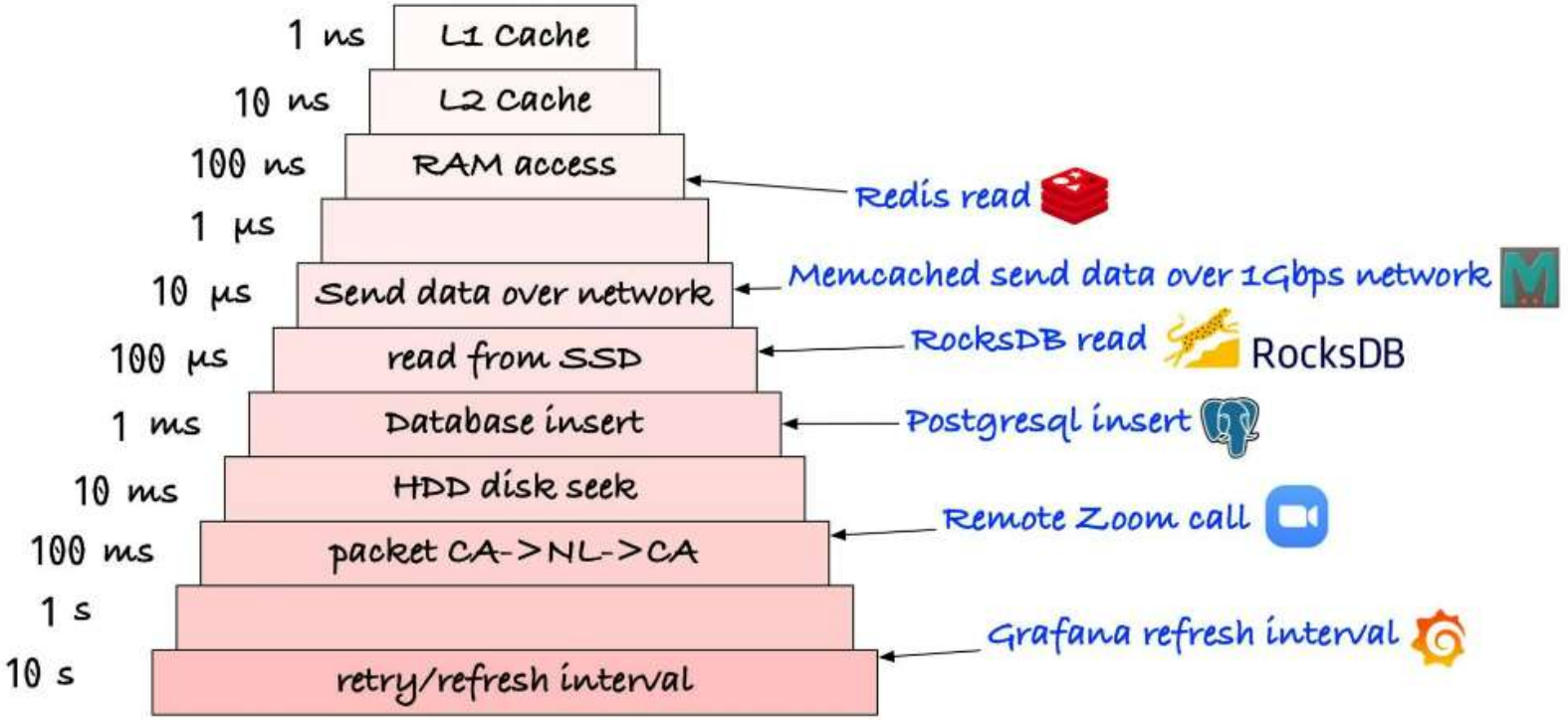
연산명	시간
L1 캐시 참조	0.5ns
분기 예측 오류(branch mispredict)	5ns
L2 캐시 참조	7ns
뮤텍스(mutex) 락/언락	100ns
주 메모리 참조	100ns
Zippy로 1 KB 압축	10,000ns = 10μs
1 Gbps 네트워크로 2 KB 전송	20,000ns = 20μs
메모리에서 1 MB 순차적으로 read	250,000ns = 250μs
같은 데이터 센터 내에서의 메시지 왕복 지연시간	500,000ns = 500μs
디스크 탐색(seek)	10,000,000ns = 10ms
네트워크에서 1 MB 순차적으로 read	10,000,000ns = 10ms
디스크에서 1 MB 순차적으로 read	30,000,000ns = 30ms
한 패킷의 CA(캘리포니아)로부터 네덜란드까지의 왕복 지연시간	150,000,000ns = 150ms

- 메모리는 빠르지만 디스크는 아직도 느리다.
- 디스크 탐색(seek)은 가능한 한 피하도록
- 단순한 압축 알고리즘은 빠르다.
- 데이터를 인터넷으로 전송하기 전에 가능하면 압축하기
- 데이터 센터는 보통 여러 지역에 분산되어 있고, 센터들 간에 데이터 전송에는 시간이 오래 걸린다.

응답지연 값


2020
1ns
L1 cache reference: 1ns
Branch mispredict: 3ns
L2 cache reference: 4ns
Mutex lock/unlock: 17ns
100ns =
Main memory reference: 100ns
1,000ns ≈ 1µs
Compress 1KB wth Zippy: 2,000ns ≈ 2µs
10,000ns ≈ 10µs =
Send 2,000 bytes over commodity network: 11ns
SSD random read: 16,000ns ≈ 16µs
Read 1,000,000 bytes sequentially from memory: 1,000ns ≈ 1µs
Round trip in same datacenter: 500,000ns ≈ 500µs
1,000,000ns = 1ms =
Read 1,000,000 bytes sequentially from SSD: 19,000ns ≈ 19µs
Disk seek: 2,000,000ns ≈ 2ms
Read 1,000,000 bytes sequentially from disk: 474,000ns ≈ 474µs
Packet roundtrip CA to Netherlands: 150,000,000ns ≈ 150ms

응답지연 값


1 ns
L1 Cache
10 ns
L2 Cache
100 ns
RAM access
Redis read
1 µs
10 µs
Send data over network
Memcached send data over 1Gbps network
100 µs
read from SSD
RocksDB read
RocksDB
1 ms
Database insert
Postgresql insert
10 ms
HDD disk seek
100 ms
packet CA->NL->CA
Remote Zoom call
1 s
10 s
retry/refresh interval
Grafana refresh interval

가용성에 관계된 수치들

고가용성

- 고가용성은 시스템이 오랜 시간 동안 지속적으로 중단 없이 운영될 수 있는 능력을 지칭하는 용어이다.
- 가용성은 퍼센트(%)로 표현하는데, 100%는 시스템이 한 번도 중단되 적이 없었음을 의미한다.
- 대부분은 서비스는 99%에서 100% 사이의 값을 갖는다.

SLA

- SLA (Service Level Agreement)는 서비스 사업자 (Service Provider)가 보편적으로 사용하는 용어이다.
- 사업자와 고객사이에 맺어진 합의를 의미한다.
- 이 합의에는 서비스 사업자가 제공하는 서비스의 가용시간 (uptime)이 공식적으로 기술되어 있다.
- 가용시간은 관습적으로 숫자9의 개수와 시스템 장애시간 (downtime) 사이의 관계이다.

가용률	하루당 장애시간	주당 장애시간	개월당 장애시간	연간 장애시간
99%	14.40분	1.68시간	7.31시간	3.65일
99.9%	1.44분	10.08분	43.83분	8.77시간
99.99%	8.64초	1.01분	4.38분	52.60분
99.999%	864.00밀리초	6.05초	26.30초	5.26분
99.9999%	86.40밀리초	604.80밀리초	2.63초	31.56초

클라우드 SLA

가용률	하루당 장애시간	주당 장애시간	개월당 장애시간	연간 장애시간
99%	14.40분	1.68시간	7.31시간	3.65일
99.9%	1.44분	10.08분	43.83분	8.77시간
99.99%	8.64초	1.01분	4.38분	52.60분
99.999%	864.00밀리초	6.05초	26.30초	5.26분
99.9999%	86.40밀리초	604.80밀리초	2.63초	31.56초

AWS 99.99%
https://aws.amazon.com/ko/ec2/sla/historical/

Azure 99.99%
https://azure.microsoft.com/en-us/explore/global-infrastructure/availability-zones

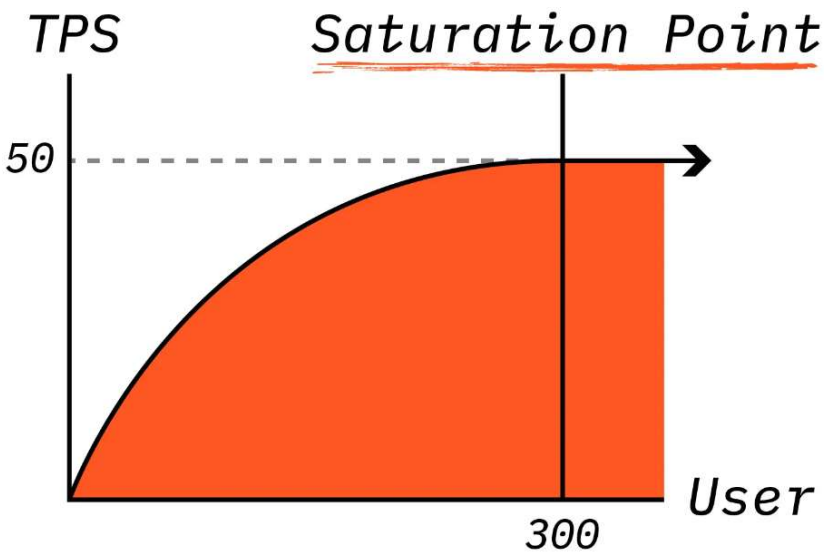
GCP 99.95%
https://cloud.google.com/run/sla

NAVER CLOUD 99.90%
https://www.ncloud.com/policy/sla/svc

처리율

TPS

- TPS (Transaction per second)는 초당 트랜잭션의 개수를 의미한다.
- 일정기간 (1분, 1시간)을 정하고 그동안 실행된 트랜잭션의 수를 구해서 1초 구간의 값으로 변경
- 서비스에 사용자가 지속적으로 늘어 날 경우 특정 지점부터 TPS가 더이상 증가하지 않는 상황이 발생한다.
- TPS와 동시 접속자를 미리 계산해보면 서비스의 성능을 예측할 수 있다.



- 이 시스템의 한계는 TPS 50이다.
- 300명 이하의 사용자일때는 모두가 서비스를 쾌적하게 사용할 수 있다.
- 그래프상 사용자가 300명일 경우 TPS 50에 도달한다.
- 서비스가 성장하여 사용자가 300명을 넘어가기 시작하면 다수의 응답시간이 느려 질 것이다.

규모 추정 예제

가정

- 월간 능동 사용자(monthly active user)의 3억(300milion) 명이다.
- 50%의 사용자가 트위터를 매일 사용
- 평균적으로 각 사용자는 매일 2건의 트윗을 업로드
- 미디어를 포함하는 트윗은 10% 정도
- 데이터는 5년간 보관

추정

QPS 추정치

- 일간 능동 사용자(Daily Active User, DAU) = 3억 x 50% = 1.5억(150 milion)
- QPS = 1.5억 x 2트윗 / 24시간 / 3600 초 = 약 3500
- 최대 QPS(Peek QPS) = 2 * QPS = 약 7000

미디어 저장을 위한 저장소 요구량

- 평균 트윗 크기
  - tweet_id에 64바이트
  - 텍스트에 140바이트
  - 미디어에 1MB
- 미디어 저장서 요구량: 1.5억 x 2 x 10% x 1 MB = 30TB/일
- 5년간 미디어를 보관하기 위한 저장소 요구량: 30TB x 365 x 5 = 약 55PB