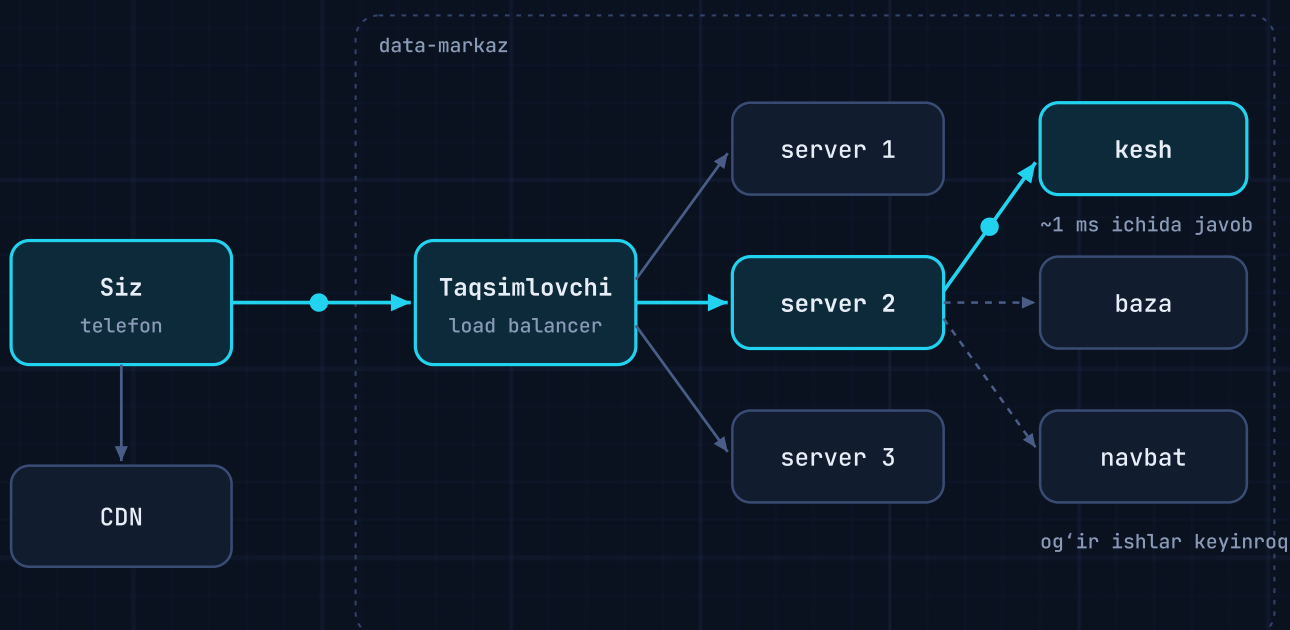


Tizim dizayni

shpargalka va 20 ta intervyu savoli

Katta saytlar ichida nima bo'lishini oddiy tilda tushuntiramiz: keshdan navbatgacha, raqamlardan intervyugacha. Har mavzu uchun hayotiy misol, sxema va amaliy maslahat bor.



Bitta so'rovning yo'li: telefoningizdan serverlarga va orqaga. Bularning hammasi ko'z ochib-yumguncha bo'ladi.

Bu qo'llanma kimga kerak

Instagram lentasini ochganingizda soniyaning bir qismida yuzlab rasm, layk va izoh keladi. Buning ortida bitta server emas, o'nlab qismlardan iborat tizim turadi. Tizim dizayni shu qismlarni to'g'ri tanlash va bir-biriga ulash ilmi.

Sizga foydali bo'ladi, agar

- backend dasturchi sifatida intervyuga tayyorlanayotgan bo'lsangiz;
- loyihangiz foydalanuvchi ko'paygan sari sekinlashayotgan bo'lsa;
- dasturchilar bilan bir tilda gaplashmoqchi bo'lgan menejer yoki asoschi bo'lsangiz.

Qanday o'qish kerak

- 3-varaqdagi katta xaritadan boshlang. Qolgan bo'limlar shu xaritaning bir qismini kattalashtiradi.
- Har bo'limda avval hayotiy misol, keyin sxema, oxirida maslahat. Sxemani qo'lda qayta chizib ko'ring: shunda yaxshiroq eslab qolasiz.
- Intervyudan bir kun oldin 11-varaqdagi shpargalka va 14-15-varaqdagi savollarni takrorlang.

Mundarija

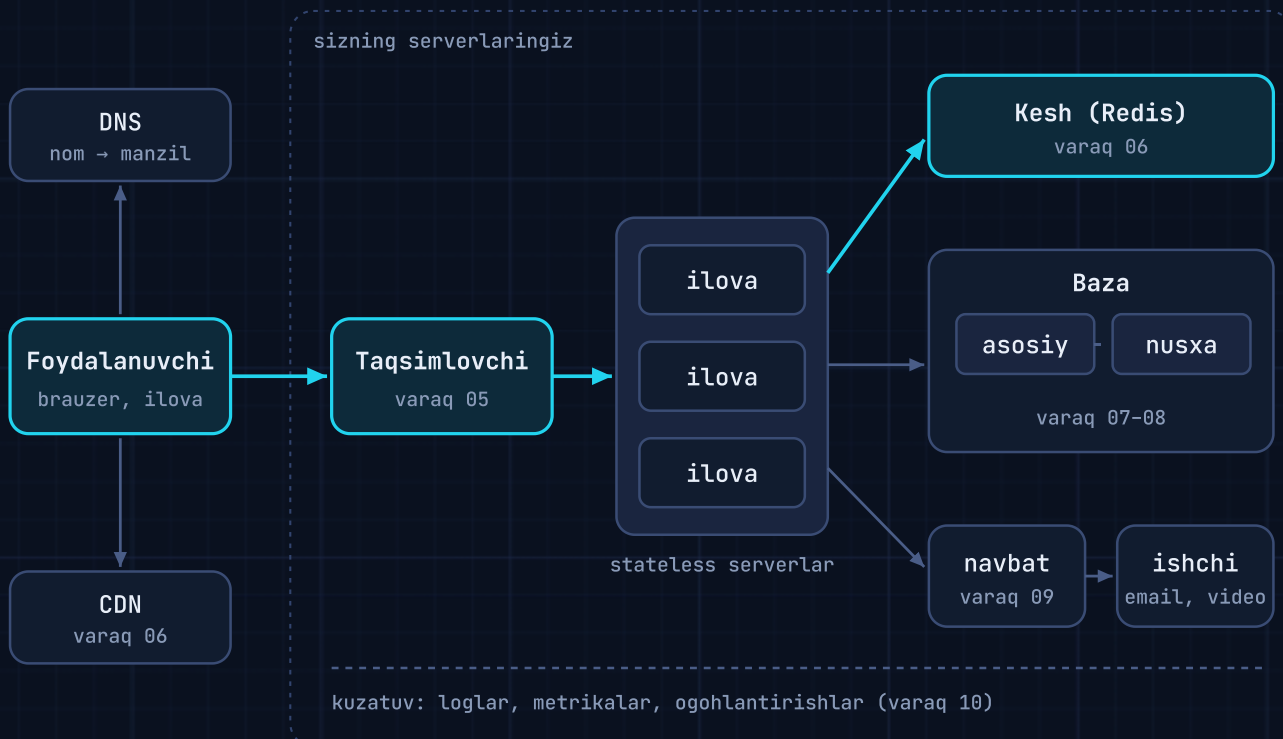
03	Katta xarita	umumiy ko'rinish
04	Raqamlar sezgisi	tezlik, ishonchlilik
05	Masshtablash	server qo'shish
06	Kesh	tez o'qish
07	Ma'lumotlar bazasi	indeks, sharding
08	Kelishuvlar	ACID, CAP
09	Navbat va hodisalar	asinxron ish
10	Ishonchlilik	buzilishga tayyorlanish
11	Shpargalka	muammo va yechim
12	Intervyu rejasi	45 daqiqa
13	Amaliy misol	havola qisqartirgich
14	20 ta savol	javob yo'nalishi bilan
16	Resurslar	kitob, kurs, blog
17	Davomi	Telegram kanal



Salom! Bu qo'llanmada architecuz videolaridagi eng muhim g'oyalarni bir joyga yig'dim va intervyuda so'raladigan savollarni qo'shdim. Bir o'qishda hammasini yodlash shart emas. Saqlab qo'ying va kerak bo'lganda qaytib oching.

Har bir katta sayt shu xaritaga o'xshaydi

Instagram, Uzum yoki Payme: nomlari va hajmi har xil, lekin ichidagi qismlar deyarli bir xil. Bu xaritani bilsangiz, istalgan tizimni tushuna boshlaysiz.



- **DNS** internetning telefon kitobi: instagram.com nomini server manziliga aylantiradi.
- **CDN** rasm va videolarni sizga eng yaqin shahardagi serverdan beradi.
- **Taqsimlovchi** so'rovlarni serverlar orasida teng bo'ladi va buzilgan serverni chetlab o'tadi.
- **Ilova serverlari** asosiy kod ishlaydigan joy. Ular hech narsani eslab qolmaydi, shuning uchun istalgancha qo'shish mumkin.
- **Kesh** tez-tez so'raladigan javoblar uchun tezkor xotira. Bazaga boradigan so'rovlarni kamaytiradi.
- **Baza** ma'lumotning asosiy, doimiy uyi. Asosiy nusxa yozadi, qolgan nusxalar o'qishga yordam beradi.
- **Navbat va ishchilar** shoshilinch bo'lmagan ishlarni foydalanuvchini kuttirmasdan keyinroq bajaradi.

Maslahat

Intervyuda istalgan masalani shu xaritadan boshlang. Avval oddiy variantni chizing (mijoz, server, baza), keyin muammo paydo bo'lgan joyga yangi qism qo'shing.



Raqamlar sezgisi

Yaxshi muhandis «bu tez» demaydi, «bu yuz baravar tezroq» deydi. Kompyuterdagi vaqtlarni his qilish uchun ularni odam vaqtiga o'tkazamiz: protsessor keshidan o'qish 1 soniya davom etadi, deb tasavvur qiling.

amal	haqiqiy vaqt	odam vaqtida
Protsessor keshi	0,5 ns	1 soniya
Operativ xotira (RAM)	100 ns	3 daqiqa
SSD diskdan o'qish	16 μ s	9 soat
Data-markaz ichida borib kelish	0,5 ms	12 kun
Oddiy disk (HDD) qidiruvi	10 ms	8 oy
Okean ortiga borib kelish	150 ms	9,5 yil

Yaxlitlangan qiymatlar; ustunlar logarifmik shkalada. Manba: Jeff Dean va Peter Norvig jadvali, Colin Scott yangilagan versiya.

Xulosa

Xotiradan o'qish tarmoqdan o'qishdan minglab marta tez. Kesh, CDN va «ma'lumotni foydalanuvchiga yaqin saqlash» g'oyalarining hammasi shu raqamlardan kelib chiqadi.

Chamalash uchun 4 qoida

- 1 sutka $\approx$ 86 400 soniya $\approx$ 100 ming. Demak **kuniga 1 mln so'rov $\approx$ soniyasiga 12.**
- Cho'qqi vaqtidagi yuklama odatda o'rtachadan 2–3 baravar ko'p. Tizimni cho'qqiga mo'ljallang.
- Ijtimoiy ilovalarda o'qish yozishdan 10 baravar va undan ham ko'p.
- 1 mln foydalanuvchi $\times$ 1 KB = 1 GB. Hajmni shunday qo'pol hisoblash yetarli.

«To'qqizlar» nimani anglatadi

Ishlash vaqti	Yiliga to'xtash
99%	3,65 kun
99,9%	8,8 soat
99,99%	53 daqiqa
99,999%	5 daqiqa

Har bir yangi «to'qqiz» narxni bir necha baravar oshiradi. Avval biznesga qanchasi kerakligini so'rang.

Mijoz ko'paydi. Endi nima qilamiz?

«Choyxonaga mehmon ko'paydi. Ikki yo'l bor: oshpazga kattaroq qozon olib berish yoki yana bir nechta oshpaz yollash.»

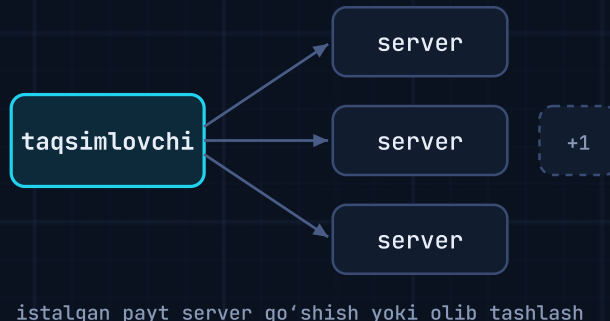
Vertikal: kattaroq qozon



Vertikal

- + oddiy: kodni o'zgartirish shart emas.
- chegarasi bor va qimmatlashib boradi.
- server bitta: u yiqilsa, hammasi yiqiladi.

Gorizontal: ko'proq oshpaz



Gorizontal

- + deyarli cheksiz o'sish, bitta server yiqilsa ham tizim ishlaydi.
- murakkabroq: taqsimlovchi va stateless kod kerak.

Stateless: serverlar almashtiriladigan bo'lsin

Agar server foydalanuvchi savatini o'z xotirasida saqlasa, keyingi so'rov boshqa serverga tushganda savat «yo'qoladi». Shuning uchun holat (sessiya, savat) serverdan tashqarida saqlanadi: Redisda, bazada yoki imzolangan tokenda. Shunda har qanday server har qanday so'rovga javob bera oladi.

Taqsimlovchi qanday tanlaydi

Usul	Qanday ishlaydi	Qachon mos
Round robin	Navbat bilan: 1, 2, 3, yana 1...	serverlar bir xil, so'rovlar yengil
Least connections	Eng kam band serverga beradi	so'rovlar uzun-qisqa aralash
IP hash	Bitta mijoz doim bitta serverga tushadi	eski, stateful ilovalar

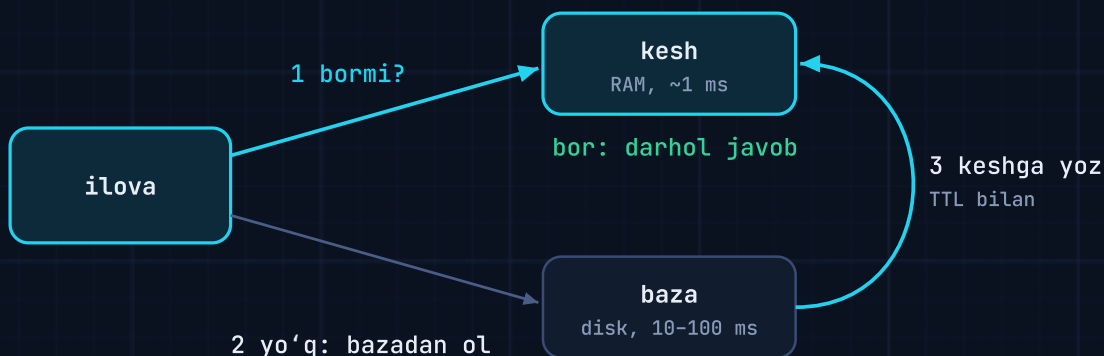
Taqsimlovchi har bir serverni bir necha soniyada tekshirib turadi (*health check*). Javob bermagan server ro'yxatdan vaqtincha chiqariladi.

Maslahat

Intervyuda «gorizontal masshtablayman» deyishdan oldin serverlarni stateless qiling. Aks holda yangi qo'shilgan server foydasiz bo'ladi.

Kesh: eng kerakli narsa qo'l ostida

«Oshxonada tuz va murch plita yonida turadi, qolgan zaxira esa omborda. Har safar omborga yugurmaysiz.»



Cache-aside, eng keng tarqalgan usul: ilova avval keshdan so'raydi, yo'q bo'lsa bazadan olib keshga ham yozib qo'yadi.

- **Hit ratio** so'rovlarning necha foizi keshdan javob olgani. 90%+ yaxshi natija.
- **TTL** yozuv qancha yashashi. Masalan, mahsulot narxi 5 daqiqa, profil rasmi 1 kun.
- **Jitter** TTLga tasodifiy bir necha soniya qo'shish. Shunda minglab yozuv bir paytda eskirib, bazaga birdan yopirilmaydi.
- **Eviction** xotira to'lsa nima o'chiriladi. Odatda LRU: eng uzoq vaqt so'ralmagan yozuv.
- **Invalidatsiya** ma'lumot o'zgarsa, keshdagi eski nusxani o'chirish. Eng qiyin qism shu.
- **CDN** ham kesh, faqat dunyo bo'ylab tarqalgan: rasm, video, JS va CSS fayllar uchun.

Tarix: Facebook, 2010-yil 23-sentabr

Facebook taxminan 2,5 soat ishlamay qoldi. Bitta sozlama qiymati xato yozilgan edi. Har bir server xatoni ko'rib, to'g'ri qiymatni bazadan so'ray boshladi. Minglab serverning so'rovi bazani bosib qoldi va kesh ham yordam bera olmadi.

Saboq: kesh ishlamay qolsa, uning ortidagi baza bu yuklamani ko'tara oladimi, oldindan o'ylang.

Manba: Facebook muhandislik blogi, «More details on today's outage», 2010.

Fakt

Netscape muhandisi Phil Karltonga nisbat beriladigan mashhur gap: «Informatikada faqat ikkita qiyin narsa bor: keshni yangilash va narsalarga nom qo'yish».

Baza sekinlashsa: uch qadam, shu tartibda

Arzon va oddiy yechimdan boshlang. Sharding kuchli, lekin murakkab: unga faqat oldingi qadamlar yetmaganda o'ting.

1

Indeks: kitob oxiridagi ko'rsatkich

Indekssiz baza 10 mln qatorni birma-bir ko'rib chiqadi. B-tree indeksi bilan kerakli qatorga 3–4 qadamda yetadi. Lekin har bir indeks yozishni biroz sekinlashtiradi va joy egallaydi: faqat **WHERE**, **JOIN** va **ORDER BY** ichida ishlatiladigan ustunlarga qo'ying.

2

Replikatsiya: bazaning nusxalari

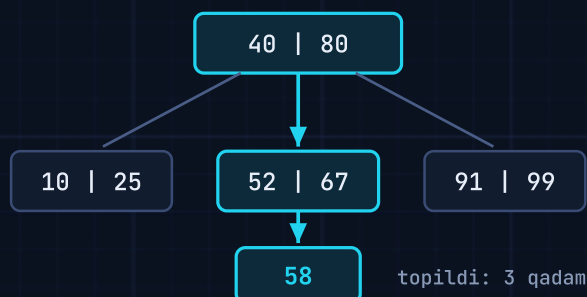
Asosiy baza yozadi, nusxalar o'qiydi. O'qish ko'p bo'lgan tizimlar uchun ajoyib. Nusxa biroz kechikadi (*replication lag*): yangi yozilgan ma'lumot bir lahzaga ko'rinmasligi mumkin.

3

Sharding: bazani bo'laklarga bo'lish

Har bir bo'lakda ma'lumotning bir qismi turadi. Masalan, foydalanuvchilar ID bo'yicha 4 ta serverga taqsimlanadi. Asosiy savol: qaysi kalit bo'yicha bo'lish (*shard key*).

B-tree: har qadamda qidiruv maydoni toraydi



Sharding: $user_id \% 4$



Maslahat

Yaxshi shard kaliti yuklamani teng bo'ladi va ko'p so'rovlarni bitta bo'lak ichida qoldiradi. Yomon misol: yaratilgan sana. Unda barcha yangi yozuvlar bitta «issiq» bo'lakka tushadi.

Tarix: B-tree, 1970-yillar

B-tree indeksini Boeing tadqiqot laboratoriyasida Rudolf Bayer va Edward McCreight ixtiro qilgan. Oradan 50 yildan ortiq vaqt o'tdi: PostgreSQL, MySQL va SQLite hali ham shu tuzilmadan foydalanadi. «B» harfi nimani anglatishini mualliflar hech qachon aniq aytmagan: Boeing, Bayer yoki *balanced* bo'lishi mumkin.



Mukammal tizim yo'q, faqat kelishuvlar bor

Tizim dizaynida har bir tanlov nimanidir beradi va nimanidir oladi. Ikki eng mashhur kelishuvni bank va doska misolida ko'ramiz.

ACID: pul o'tkazmasi yo'qolmasligi uchun

Ali Valiga 100 ming so'm o'tkazmoqda. Bu ikkita amal: Alidan yechish va Valiga qo'shish.

Xususiyat	Oddiy tilda	Bank misolida
Atomicity	Hammasi yoki hech narsa	Pul yechilib, o'tmay qolmaydi
Consistency	Qoidalar hech qachon buzilmaydi	Balans manfiy bo'lmaydi
Isolation	Bir vaqtdagi amallar bir-biriga xalaqit bermaydi	Ikki o'tkazma bitta pulni ikki marta ishlatmaydi
Durability	«Tasdiqlandi» degani doimiy	Tok o'chsa ham o'tkazma saqlanadi

CAP: tarmoq uzilganda nima muhimroq



Serverlar orasidagi aloqa uzilishi muqarrar. Shunda tizim tanlaydi: **aniqlik** (eskirgan javobdan ko'ra xato qaytaraman) yoki **ochiqlik** (har doim javob beraman, hatto biroz eski bo'lsa ham).

C bank balansi, chipta bron qilish, ombor qoldig'i.
A layklar soni, lenta, ko'rishlar soni.

Eventual consistency: «oxir-oqibat bir xil»

«Direktor yangilikni o'qituvchilar xonasidagi doskaga yozdi. Har bir sinf uni turli vaqtda eshitadi, lekin tushlikkacha hamma biladi.»

Siz layk bosdingiz, do'stingiz esa eski sonni ko'ryapti. Bu xato emas: nusxalar bir necha soniyada bir xil holatga keladi. Evaziga tizim tez va doim ochiq ishlaydi.

Fakt

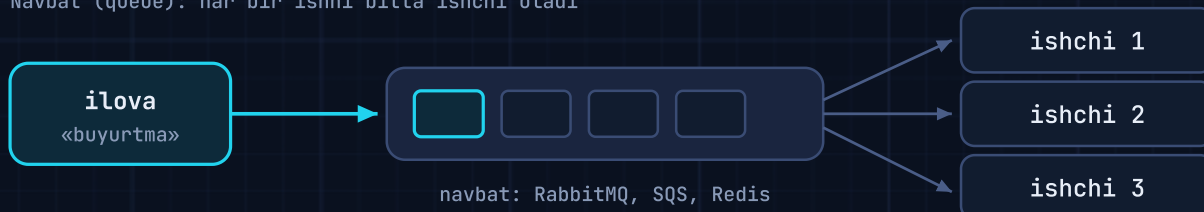
CAPni 2000-yilda Erik Bryuer gipoteza sifatida aytgan. 2002-yilda MIT olimlari Set Gilbert va Nensi Linch uni qat'iy isbotlagan.



Kutmang: ishni navbatga qo'ying

«Ofitsiant buyurtmani oshxonaga ilib qo'yadi va keyingi stolga o'tadi. Oshpaz pishirib bo'lguncha uning yonida kutib turmaydi.»

Navbat (queue): har bir ishni bitta ishchi oladi



Pub/Sub: bitta hodisani hamma obunachi oladi (fan-out)



Qachon kerak

- Foydalanuvchi kutishi shart bo'lmagan ishlar: email, video siqish, hisobot.
- Yuklama to'liqin bo'lib keladi: navbat uni yumshatadi, ishchilar o'z tezligida oladi.
- Bitta hodisaga bir nechta xizmat javob beradi: pub/sub ular orasidagi bog'liqlikni uzadi.

Nimaga ehtiyot bo'lish kerak

- Xabar **kamida bir marta** yetkaziladi, ya'ni ikki marta kelishi mumkin. Ishchi idempotent bo'lsin (10-varaq).
- Qayta-qayta xato bergan xabarlar alohida «o'lik xatlar» navbatiga (*dead letter queue*) o'tadi.
- Navbat uzunligini kuzating: u o'sib borsa, ishchilar ulgurmayapti.

Tarix: LinkedIn va Kafka

LinkedIn ichki tizimlari bir-biri bilan o'nlab alohida yo'llar orqali bog'lanib, chalkash to'rga aylangan edi. Muhandis Jey Kreps va hamkasblari hamma hodisalar oqadigan bitta «magistral» yaratdi va unga yozuvchi Frans Kafka nomini berdi. 2011-yilda loyiha ochiq kodga aylandi. Bugun LinkedIn Kafka orqali kuniga trillionlab xabar o'tkazadi.

Hamma narsa buziladi. Bunga tayyormisiz?

Amazon texnik direktori Verner Fogels shunday deydi: «Hamma narsa doim buziladi». Yaxshi tizim buzilmaydigan emas, buzilganda ham ishlashda davom etadigan tizim.

Timeout

Har bir tashqi so'rovga vaqt chegarasini qo'ying, masalan 2 soniya. Aks holda bitta sekin xizmat barcha oqimlarni egallab oladi va butun sayt qotadi.

Retry: qayta urinish, lekin oqilona



Har safar kutish ikki baravar oshadi (*exponential backoff*) va unga tasodifiy qo'shimcha (*jitter*) qo'shiladi, shunda minglab mijoz bir vaqtda qayta urinmaydi.

Circuit breaker: uydagi avtomat



Xizmat ketma-ket xato bersa, unga so'rov yuborish vaqtincha to'xtatiladi. Bir ozdan keyin bitta sinov so'rovi yuboriladi. O'tsa, hammasi yana ulanadi.

Rate limiting: metro turniketi

Bitta mijoz yoki IP uchun so'rovlar chegarasi, masalan daqiqasiga 100 ta. Mashhur algoritim *token bucket*: chelakka tangalar bir xil tezlikda tushadi, har so'rov bitta tanga oladi. Tanga tugasa, javob **429 Too Many Requests**.

Idempotentlik: «To'lash» tugmasini ikki marta bosish

Internet uzilib, ilova so'rovni qayta yubordi. Pul ikki marta yechilmasligi uchun har bir to'lov noyob kalit bilan keladi (**Idempotency-Key**). Server bu kalitni allaqachon ko'rgan bo'lsa, avvalgi javobni qaytaradi va amalni takrorlamaydi.

Tarix: Amazon S3, 2017-yil 28-fevral

Muhandis S3 hisob-kitob tizimidagi sekinlikni tuzatish uchun bir nechta serverni o'chirmoqchi edi. Buyruqdagi bitta xato tufayli rejadagidan ancha ko'p server o'chdi. AWSning AQSh sharqidagi mintaqasida S3 4 soatga yaqin ishlamadi va u bilan birga minglab saytlar ham to'xtadi.

Sabiq: xavfli buyruqlarga himoya qo'ying va tizimni qismlarga bo'ling. Shunda bitta xato hammasini yiqitmaydi.

Manba: AWS rasmiy hisoboti, «Summary of the Amazon S3 Service Disruption», 2017.

Shpargalka: muammo bo'lsa, qayerga qarash kerak

Chapda belgi, o'rtada birinchi va eng arzon qadam, o'ngda u yetmasa nima qilish. Bu varaqni saqlab qo'ying.

Muammo	Birinchi qadam	Keyingi qadam
Sahifa sekin, baza doim band	Kesh (Redis)	O'qish uchun baza nusxalari
Qidiruv yoki filtr sekin	Indeks qo'shish	To'liq matnli qidiruv (Elasticsearch)
Rasm va videolar sekin yuklanadi	CDN	Hajmni kichraytirish: WebP, AVIF
Bitta server yetmayapti	Stateless + taqsimlovchi	Avtomatik masshtablash
Og'ir ish foydalanuvchini kuttiradi	Navbat + ishchilar	Tayyor bo'lganda push xabar
Baza yozishga ulgurmayapti	Yozuvlarni guruhlab yozish	Sharding
Bitta hodisaga ko'p xizmat javob berishi kerak	Pub/Sub (fan-out)	Event-driven arxitektura
Bitta xizmat yiqilsa, hammasi yiqiladi	Timeout + circuit breaker	Qisman ishlash rejimi
Botlar va haddan tashqari so'rovlar	Rate limiting	CDN/WAF himoyasi
To'lov ikki marta yechilyapti	Idempotentlik kaliti	Tranzaksiya (ACID)
Foydalanuvchi o'z o'zgarishini ko'rmayapti	Yozgandan keyin asosiy bazadan o'qish	Kesh invalidatsiyasi
Hammasi bir vaqtda eskirib, baza qulayapti	TTLga jitter qo'shish	Bitta so'rov keshni yangilaydi, qolgani kutadi
Nima buzilganini bilmaymiz	Loglar va metrikalar	Treys va ogohlantirishlar

Oltin qoida

Avval o'lchang, keyin tuzating. Qayer sekinligini bilmay turib keshni yoki shardingni qo'shish vaqtni behuda sarflaydi.

Ikkinchi qoida

Eng oddiy ishlaydigan yechimni tanlang. Har bir yangi qism bu yana bitta buzilishi mumkin bo'lgan joy.



Tizim dizayni intervyusi: 45 daqiqalik reja

Intervyuer to'g'ri javobni emas, fikrlash yo'lingizni baholaydi. Bu masalada yagona to'g'ri javob yo'q. Muhimi tartib bilan, ovoz chiqarib fikrlash.

talablar	hisob	umumiy sxema	chuqurlashish	yakun
0	7	12	25	40 45 daq

- 1 Talablarni aniqlang (5–7 daqiqa)**
Darhol chizming. So'rang: asosiy funksiyalar qaysi? Nechta foydalanuvchi? O'qish ko'pmi yoki yozish? Nima muhimroq: tezlikmi yoki aniqlik? Funksional talablarni (nima qiladi) va nofunksional talablarni (qanchalik tez, ishonchli) alohida yozing.
- 2 Qo'pol hisob (3–5 daqiqa)**
Soniyasiga nechta so'rov, 5 yilda qancha joy, qancha trafik. Aniq son shart emas, tartibi muhim: 100 mi, 10 mingmi, 1 mln mi. 4-varaqdagi qoidalar shu yerda kerak bo'ladi.
- 3 Umumiy sxema (10–15 daqiqa)**
Asosiy API endpointlarini va ma'lumot modelini yozing. Keyin 3-varaqdagi xaritani oddiydan boshlab chizing: mijoz, taqsimlovchi, server, baza. Har bir qismni nega qo'shayotganingizni ayting.
- 4 Chuqurlashish (15 daqiqa)**
Eng qiyin 1–2 joyni tanlang: qayer birinchi bo'lib sinadi? Kesh, sharding, navbat yoki replikatsiya qo'shing va har birining kelishuvini ayting: «buni olamiz, buning evaziga bunisini yo'qotamiz».
- 5 Yakun (3–5 daqiqa)**
Yechimni qisqacha takrorlang. Zaif joylarni o'zingiz ayting va vaqt bo'lganda nimani yaxshilashingizni qo'shing: monitoring, xavfsizlik, xarajat.

Ko'p uchraydigan 5 xato

- Savol bermasdan darhol texnologiyalarni sanashni boshlash.
- Birinchi daqiqadanoq «Google darajasidagi» tizimni chizish. Oddiy yechimdan boshlang.
- Kelishuvlarni aytmaslik. «Kafka qo'yaman» emas, «Kafka qo'yaman, chunki...».
- Jim ishlash. Intervyuer fikringizni eshitmasa, uni baholay olmaydi.
- Ishora berilsa ham o'z variantida qolish. Ishora bu yordam.

Amaliy misol: havola qisqartirgich

Intervyudagi eng mashhur masala: uzun havolani bit.ly/aB3kZ9q kabi qisqa havolaga aylantiruvchi xizmat. 12-varaqdagi reja bo'yicha yechamiz.

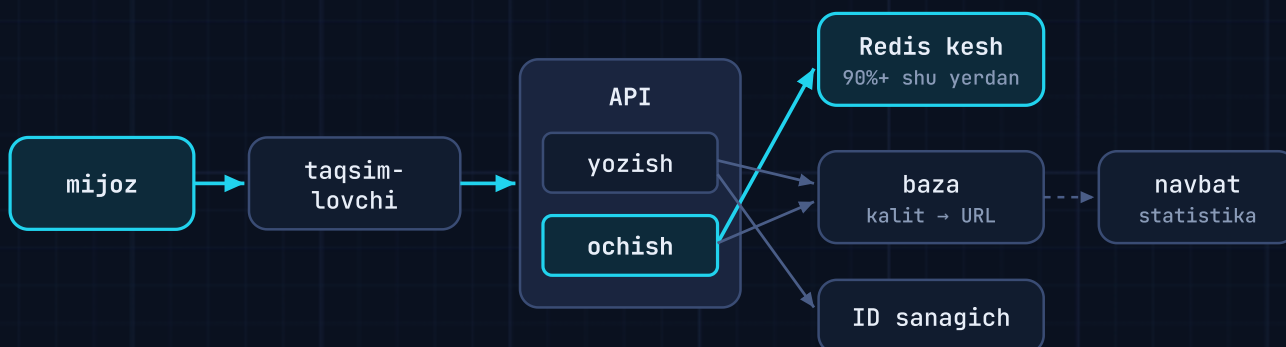
1 Talablar

Uzun havola beriladi, qisqasi qaytadi. Qisqa havola ochilganda foydalanuvchi asl manzilga yo'naltiriladi. Yo'naltirish juda tez va doim ishlashi kerak.

2 Hisob

Yangi havolalar	100 mln/kun $\approx$ 1 200/s
Ochishlar ($\times 10$)	$\approx$ 12 000/s
Bitta yozuv	$\approx$ 500 bayt
5 yilda joy	$\approx$ 90 TB

4 Sxema



5 Chuqurlashish

- **301 yoki 302?** 301 doimiy: brauzer uni eslab qoladi va serverga qayta kelmaydi. Bu tez, lekin bosishlarni sanab bo'lmaydi. Statistika kerak bo'lsa, 302.
- **O'qish yozishdan 10 baravar ko'p**, shuning uchun mashhur havolalar keshda turadi. Kesh xotirasi to'lganda LRU eng kam ochilganlarni chiqaradi.
- **Statistika** (kim, qachon, qayerdan bosdi) yo'naltirishni sekinlashtirmasligi uchun navbat orqali alohida yoziladi.
- **Bitta sanagich bitta zaif nuqta**. Har bir serverga oldindan 1000 talik ID diapazon beriladi va server ularni o'zi tarqatadi.

3 API va kalit

```
POST /api/links {"url": "..."}
→ {"short": "aB3kZ9q"}
GET /aB3kZ9q
→ 301/302, Location: asl manzil
```

Kalit: 7 belgi, base62 (a–z, A–Z, 0–9). $62^7 \approx 3,5$ trillion variant, bu ko'p yillarga yetadi. Kalitni tasodifiy yaratmasdan, o'sib boruvchi ID sanagichidan base62 ga o'tkazsak, takrorlanish bo'lmaydi.

20 ta intervyu savoli

Har bir savol ostida to'liq javob emas, javob qaysi yo'nalishda bo'lishi kerakligi berilgan. Avval o'zingiz ovoz chiqarib javob bering, keyin tekshiring.

01 Vertikal va gorizontal masshtablash farqi nima?

boshlang'ich

Kattaroq server yoki ko'proq server. Narx, chegara, bitta zaif nuqta. Gorizontal uchun stateless shart.

02 Stateless server nima va nega muhim?

boshlang'ich

Server so'rovlar orasida hech narsani eslab qolmaydi, holat Redisda yoki tokenda saqlanadi. Shuning uchun istalgan server istalgan so'rovga javob beradi.

03 Load balancer qanday ishlaydi?

boshlang'ich

Round robin, least connections, IP hash. Health check. L4 (TCP darajasi) va L7 (HTTP darajasi) farqi.

04 Kesh qayerlarda bo'lishi mumkin?

boshlang'ich

Brauzer, CDN, taqsimlovchi, ilova xotirasi, Redis, bazaning o'z keshi. Foydalanuvchiga qanchalik yaqin bo'lsa, shunchalik tez.

05 Cache-aside va write-through farqi?

o'rta

Cache-aside: ilova keshni o'zi to'ldiradi, faqat so'ralgan narsa keshga tushadi. Write-through: har bir yozuv bir vaqtda keshga ham yoziladi, kesh doim yangi, lekin yozish sekinroq.

06 Mashhur yozuv eskirsa, minglab so'rov bazaga yopiriladi. Nima qilasiz?

o'rta

Cache stampede. TTLga jitter, bitta so'rov keshni yangilaydi va qolganlari kutadi (lock), eskirishidan oldin fonda yangilash.

07 CDN qanday ishlaydi va qachon foyda bermaydi?

boshlang'ich

Foydalanuvchiga yaqin serverlarda fayl nusxalari. Statik kontent uchun ajoyib, har bir foydalanuvchi uchun alohida dinamik javoblar uchun foydasi kam.

08 Indeks nima va uni hamma ustunga qo'yish mumkinmi?

boshlang'ich

B-tree, qidiruv 3-4 qadamda. Har bir indeks yozishni sekinlashtiradi va joy oladi. Faqat so'rovlarda ishlatiladigan ustunlarga. [EXPLAIN](#) bilan tekshirish.

09 Replikatsiya va sharding farqi nima?

o'rta

Replikatsiya: bir xil ma'lumotning nusxalari, o'qish va ishonchlilik uchun. Sharding: ma'lumot bo'laklarga bo'linadi, yozish va hajm uchun. Ko'pincha ikkalasi birga ishlatiladi.

10 Shard kalitini qanday tanlaysiz?

yuqori

Yuklama teng tarqalishi, ko'p so'rovlar bitta bo'lak ichida qolishi. Hash va diapazon bo'yicha bo'lish. Issiq bo'lak muammosi. Yangi bo'lak qo'shish uchun consistent hashing.



11 ACID nima? Bank misolida tushuntiring. boshlang'ich

Atomicity, Consistency, Isolation, Durability. O'tkazma to'liq bajariladi yoki umuman bajarilmaydi, balans manfiy bo'lmaydi, tasdiqlangan o'tkazma yo'qolmaydi (8-varaq).

12 CAP teoremasi nimani aytadi? o'rta

Tarmoq uzilganda aniqlik yoki ochiqlik tanlanadi. Bank C ni tanlaydi, ijtimoiy lenta A ni. Aloqa uzilmagan paytda esa tezlik va aniqlik orasida tanlanadi.

13 SQL yoki NoSQL: qanday tanlaysiz? o'rta

Murakkab bog'lanishlar, tranzaksiyalar, aniq sxema bo'lsa SQL. Juda katta hajm, oddiy kalit bo'yicha so'rov, o'zgaruvchan tuzilma bo'lsa NoSQL. Ko'p loyihalar uchun PostgreSQL yaxshi boshlanish.

14 Navbat nima uchun kerak? boshlang'ich

Og'ir ishni foydalanuvchi so'rovidan ajratish, yuklama to'lqinlarini yumshatish, xizmatlarni bir-biridan mustaqil qilish. Ishchilar sonini alohida oshirish mumkin.

15 Navbat va pub/sub farqi nima? o'rta

Navbatda har bir xabarni bitta ishchi oladi. Pub/sub da bitta xabarni har bir obunachi oladi (fan-out). Kafka ikkalasini ham consumer group orqali qila oladi.

16 Idempotentlik nima? Misol keltiring. o'rta

Amalni bir necha marta bajarish natijani o'zgartirmaydi. To'lovda noyob kalit, ishchida «bu xabar allaqachon qayta ishlanganmi» tekshiruvi. **PUT** va **DELETE** idempotent, **POST** yo'q.

17 Retry da nega exponential backoff va jitter kerak? o'rta

Darhol qayta urinish yiqilgan xizmatni butunlay bo'g'adi. Kutishni oshirib borish unga tiklanishga vaqt beradi, jitter esa mijozlarni vaqt bo'yicha tarqatadi.

18 Rate limiterni qanday loyihalaysiz? yuqori

Token bucket yoki sliding window. Hisoblagichlar Redisda, kalit: foydalanuvchi + daqiqa. Bir nechta server uchun umumiy hisob. Javob **429** va **Retry-After** sarlavhasi.

19 Havola qisqartirgichni loyihalang. o'rta

To'liq yechim 13-varaqda. Asosiy nuqtalar: base62 kalit, o'qish uchun kesh, 301 va 302 farqi, statistika navbat orqali.

20 Telegramga o'xshash chat ilovasini loyihalang. yuqori

Doimiy ulanish (WebSocket), foydalanuvchi qaysi serverga ulanganini bilish, chat ichida xabarlar tartibi, oflayn foydalanuvchiga push, xabarlar uchun yozishga chidamli baza (Cassandra, ScyllaDB), katta guruhlarda fan-out.



Bitta savolga 10 daqiqa ajrating va javobni qog'ozga chizing. Haftasiga 3 ta savol. Bir oyda hammasi yodda qoladi.



Qayerdan davom ettirish kerak

Sohada tan olingan, o'zimiz tekshirgan manbalar. «Bepul onlayn» bo'limidagilarni bugunoq ochishingiz mumkin.

Kitoblar

- **Designing Data-Intensive Applications**, Martin Kleppmann. Bazalar, replikatsiya va taqsimlangan tizimlar haqidagi eng chuqur kitob.
- **System Design Interview**, 1 va 2-jild, Alex Xu. Intervyuga tayyorlanish uchun eng mashhur kitob, har bir masala sxema bilan.
- **Release It!**, Michael Nygard. Tizimlar amalda qanday buziladi: timeout, circuit breaker va boshqa himoya usullari.

Bepul onlayn

- **System Design Primer**: github.com/donnemartin/system-design-primer. GitHubdagi eng mashhur qo'llanmalardan biri, Anki kartalari bilan.
- **Amazon Builders' Library**: aws.amazon.com/builders-library. Amazon muhandislari timeout, retry va yuklama haqida yozgan maqolalar.
- **Latency numbers**: colin-scott.github.io. 4-varaqdagi raqamlarning interaktiv versiyasi.

Muhandislik bloglari

- **Netflix Tech Blog**: netflixtechblog.com. Millionlab tomoshabinga video qanday yetkaziladi.
- **Discord Blog**, «How Discord stores trillions of messages». Bazani ko'chirish haqidagi ajoyib haqiqiy hikoya.
- **Meta Engineering**: engineering.fb.com. Kesh va ma'lumotlar markazlari.
- **High Scalability**: highscalability.com. Mashhur saytlar arxitekturasining tahlillari.

Amaliyot uchun

- **Excalidraw**: excalidraw.com. Sxemalarni qo'lda chizilgandek chizish, intervyuda ham ko'p ishlatiladi.
- **Docker**: Redis, PostgreSQL va RabbitMQ ni o'z kompyuteringizda bir buyruq bilan ishga tushirib, sinab ko'ring.

4 haftalik reja

1-hafta

3–5-varaqlar.
Raqamlarni yodlang, katta xaritani 3 marta chizing.

2-hafta

6–8-varaqlar.
Docker'da Redis va PostgreSQL bilan sinab ko'ring.

3-hafta

9–10-varaqlar.
System Design Primer'dan 2 ta masala yeching.

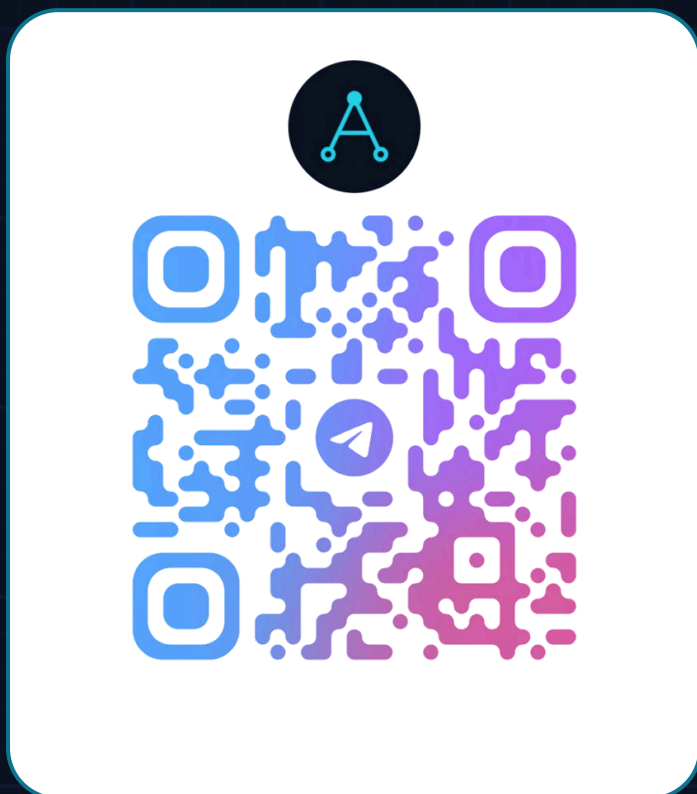
4-hafta

12–15-varaqlar.
Do'stingiz bilan 45 daqiqalik sinov intervyusi.



Bu faqat boshlanishi

Har bir mavzuni alohida, sxemalar va hayotiy hikoyalar bilan tushuntiramiz.
Yangi qo'llanmalar va videolar ortidagi to'liq sxemalar Telegram kanalimizda.



Telegram kanal

@architec_club

Kamerani QR kodga qarating yoki
qidiruvga **architec_club** deb yozing.

- videolardagi sxemalarning to'liq versiyasi;
- yangi bepul qo'llanmalar va shpargalkalar;
- savollaringizga javoblar.

Instagram

@architecuz

Har hafta yangi video: tizim dizayni va optimallashtirish oddiy tilda.



Qo'llanma foydali bo'lgan bo'lsa, uni intervyuga tayyorlanayotgan do'stingizga yuboring. Unga ham kerak bo'ladi.