Notification Systems

Primary → high level design (go from 1000 → 1B users! NoSQL/SQL, caching, Msg Q, ...)
scaling for massive load
infrastructure

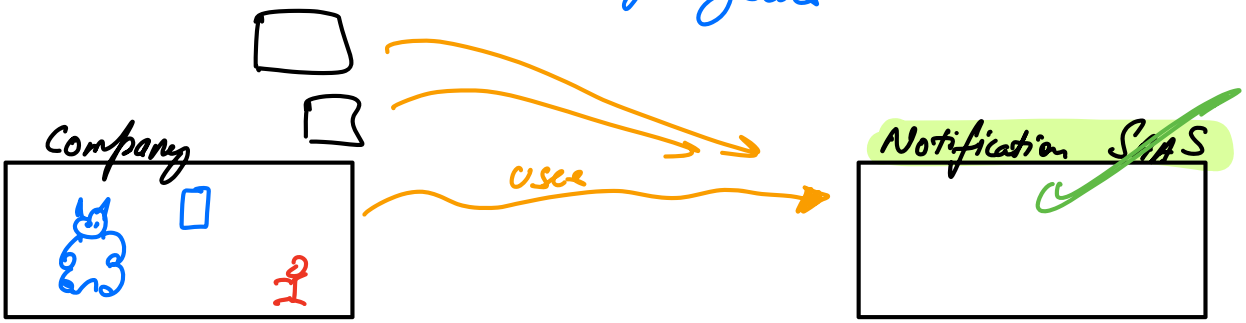
v. little → low level design (good, extensible, maintable code)
db schema / tables / attributes
code

no → code

Requirements

Functional Requirements
with code for / expose APIs for

Non functional Requirements
other design goals



- not a notification SAAS
- they have a separate (not related to notification) product
- they still need notifications

- enable notifications from businesses to users

You

- build your own notification SAAS
- a senior architect / CTO considering to build notifications in house
  - do not → plan on a 2nd party!
- just to learn HLD

5% of your entire company's productivity.

Functional Requirements

- ✓ clients should be able to send notifications to users
  company / backend code
- ✓ client should be able to send multiple types of notifications
  - SMS
  - Email
  - In-app
  - whatsapp

  plug-gable / extensible

  we should be able to very easily add a new notification type / channel in the future!
- ✓ client should be able to send bulk notifications
  - examples → Email campaign
    - celebrity problem (twitter)
- X • should support multiple clients (multi-tenancy)

✗ • client base limits

∴ it's a SCALE

how many notifications in total { • VIP clients (with more notifications)
• Pay as you go (0.001$ / notification)

also, rate limiters

✓ • users should be able to waive notifications

✓ • take into account user preferences

- Max 10 of notifications / day
- Select channels → Email ✓
  → SMS ✗
  → In-app ✓
  → whatsapp ✗
- Select types
  → turn off promotional
  → turn off notification about delivery
  → payment

✓ • Observability

- how many notification are being delivered

delivery confirmation

Google → ≈ 10^{+}M servers

- how many errors are raised per channel
- backups & logs

✓ channel restrictions

Compliance

Client → user

channel
SMS / Email / ...

✓ priority (categories / notification)

OTP / recovery / transactional / promotional / ...

(≤ 10 sec)

(hours latency is okay!)

Non-functional

✓ • low latency

✓ • reliability

guaranteed (almost) delivery of notifications

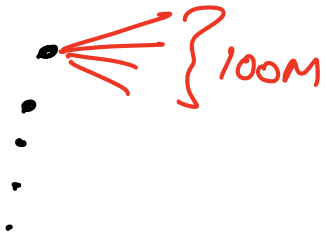
retries / observability / success callbacks

✓ • massive scale

• ~~Security~~ Not discussed!

∵ you should not build security systems unless you are an expert!

Elon Musk → 100M followers



Youtube

avg user has 100 subscriptions.

2B+ users on Youtube

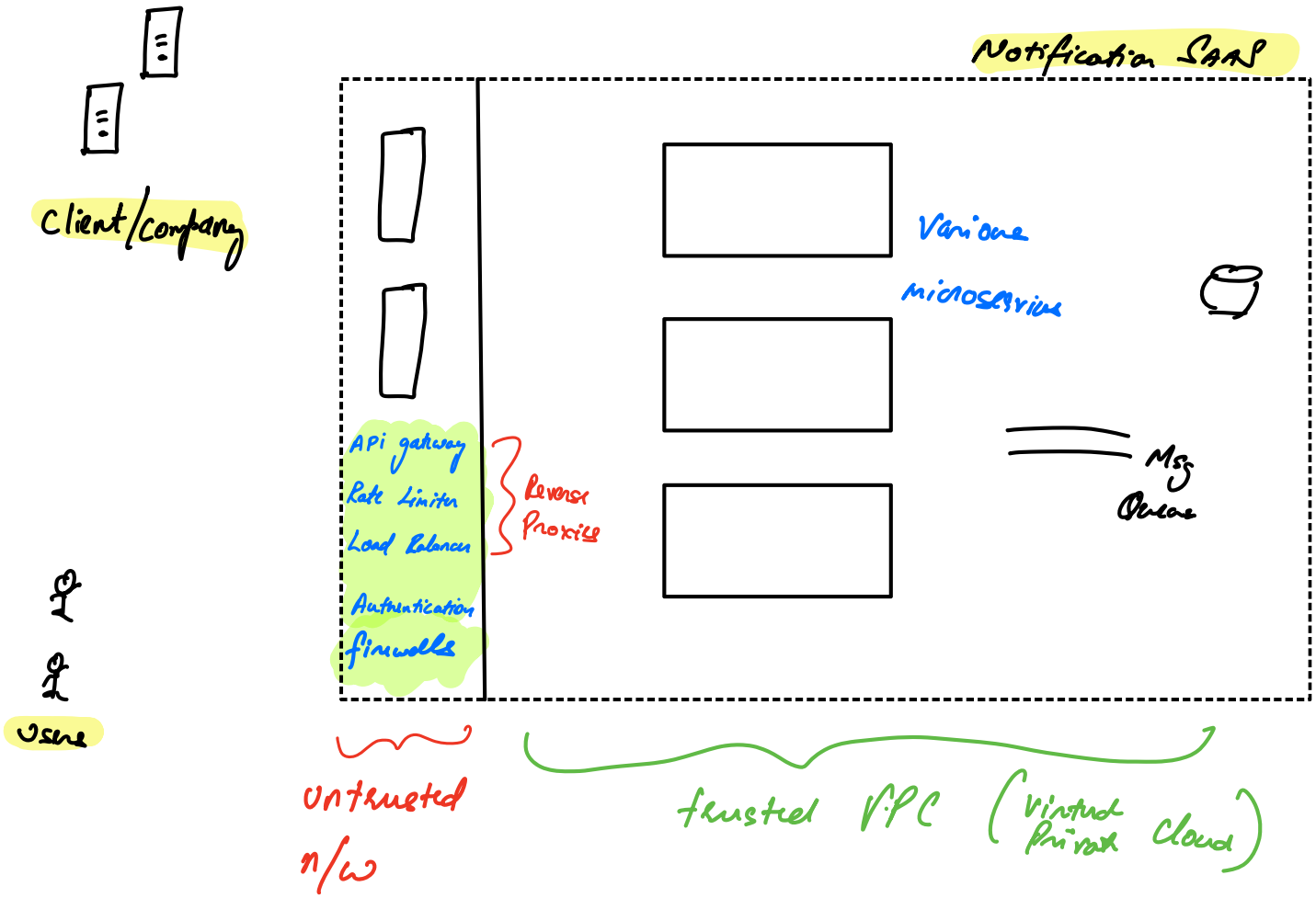
100M videos/day.

You will need to send trillion/quadrillion 3 notification/day.

Now you must
subscribe + Bell icon!

Instagram /...

Basics → typical infrastructure



HLD Basics

- Networking basics → DNS / CDN / IP / OSI
- Database basics → ACID / Normalization / schema design / indexes
- Load Balancing & consistent Hashing
- Sharding (Horizontal vs Vertical partition)
- Replication (Master slave / Multi-master)
- CAP & PACELC theorem
  - immediate consistency
  - eventual consistency
  - data loss
  - tunable consistency
- Caching
  - CDNs / browser cache
  - global / local / distributed
  - in-memory (Redis)
  - Invalidation (TTL / write through / write around / write back)
  - Eviction (LRU, FIFO, LFU, ..)
- API gateway / LB / Auth / Rate Limiter / Reverse Proxy / VPC / subnetting
- SQL vs NoSQL
  - Key-value (Redis / memcached / DynamoDB)
  - document (Mongo / Cockroach)
  - Column (Cassandra / Hbase / ... Scylla)

- File (S3 / GitLFs / HDFs)

- Internals of dbs → LSM Trees / bloom filters / sparse index / sharding / db orchestration / data replication

- Message Queues (Kafka / SQS / ...)
  Event driven architecture
  Pub / sub

- Search → Elastic search

- Microservices
  distributed transactions (2 Phase Commit / Saga Pattern / Choreography / Orchestration / correlation ID / compensating transactions)
  Observability

Sending Notifications + Incorporate various channels

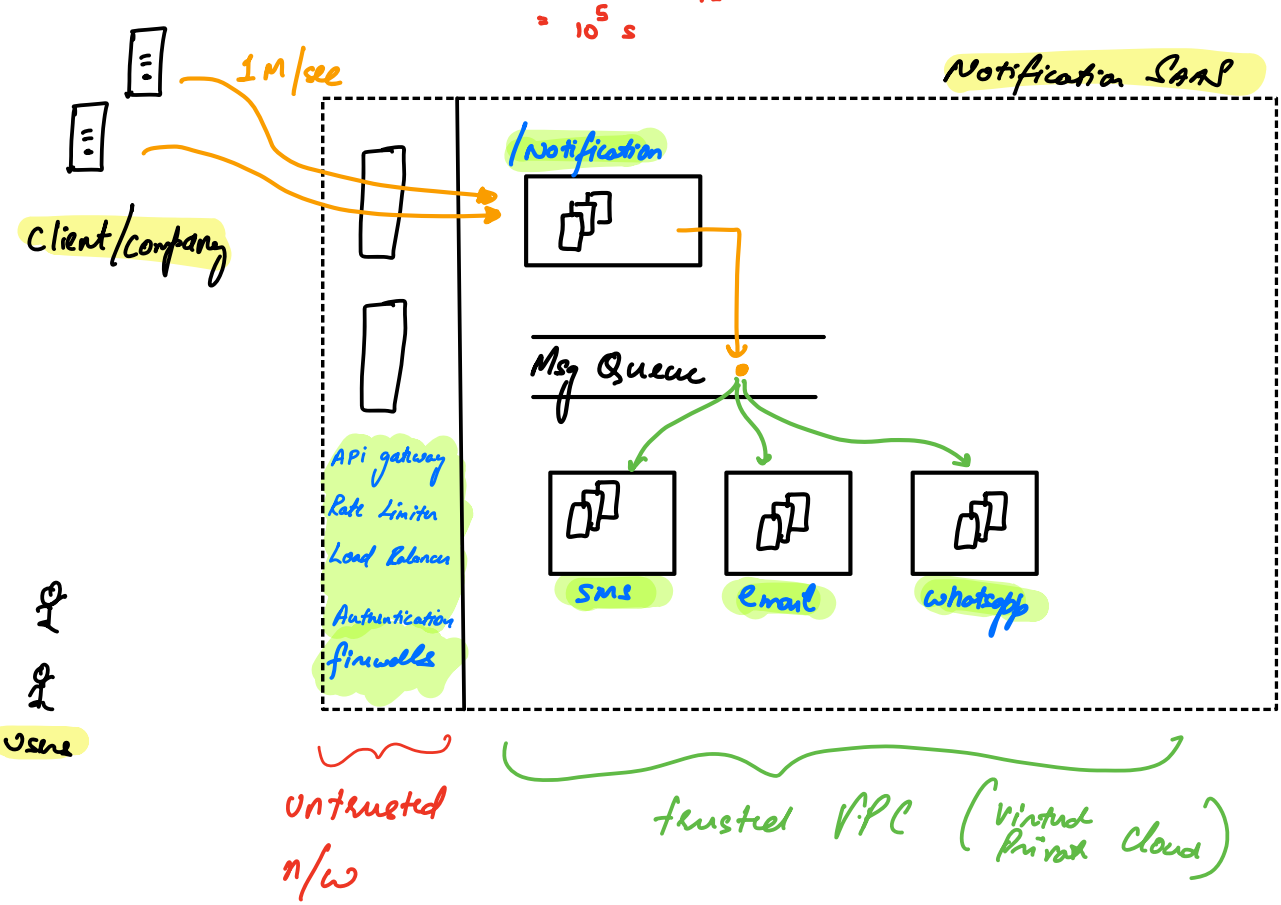
total (normal + bulk) we have 2B users

each recieves 50 notifications / day

Rate = 2B users * 50 notifi. / user / day = $\frac{2 \times 10^9 \times 50}{10^5}$ notifications / sec = 10^6

= 1M notifications / s

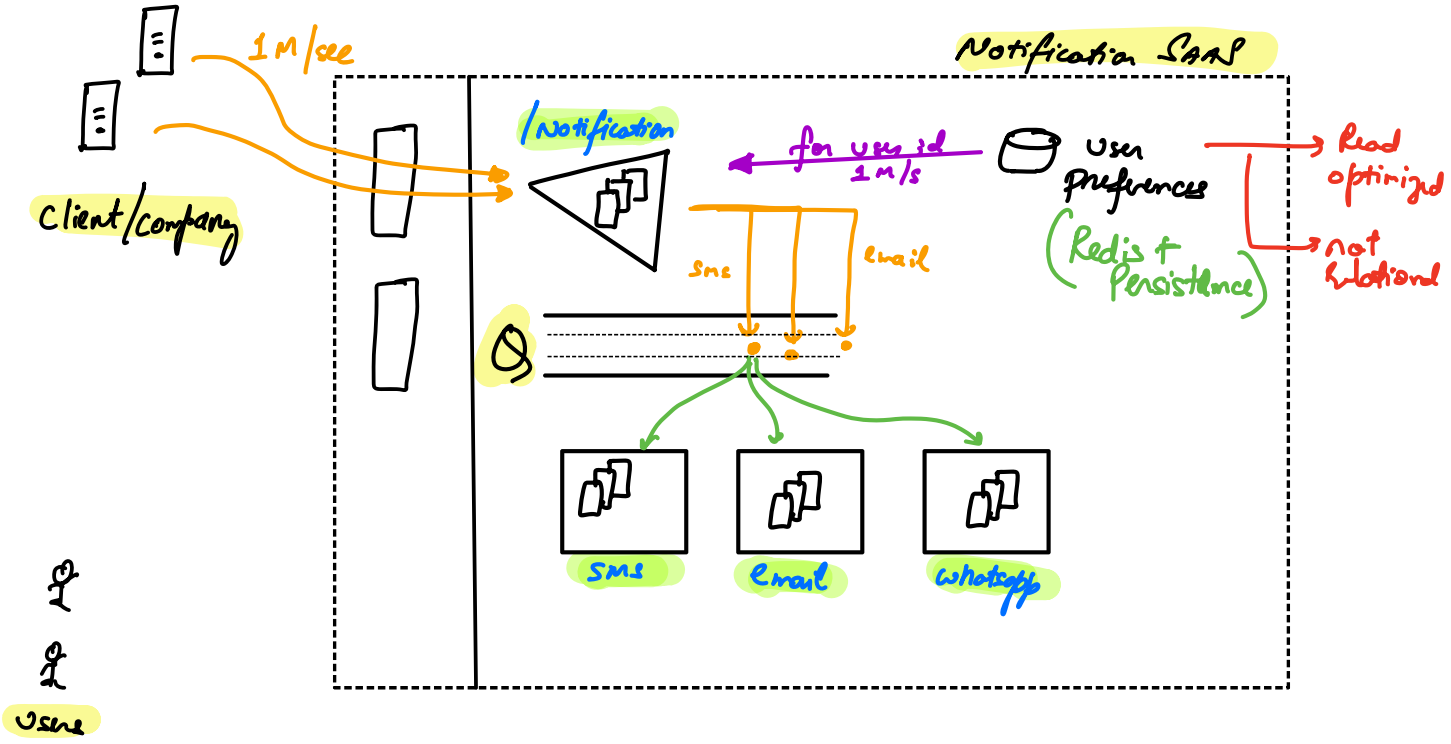
86400 s = 60 * 60 * 24 s = 10^5 s



Client API request to send Notification

```
{
  user_id : 2136,
  type : 'OTP'        promotional / urgent / ..
  channels : {
      sms : { msg : 'Hi ...', attached : 'URL' }
      email : { subject : '...', body : '...', attach ... }
  }
  inapp : { msg : '...', action : 'open (navigation)' }
}
```

In db maintain user id → Phone/email/...

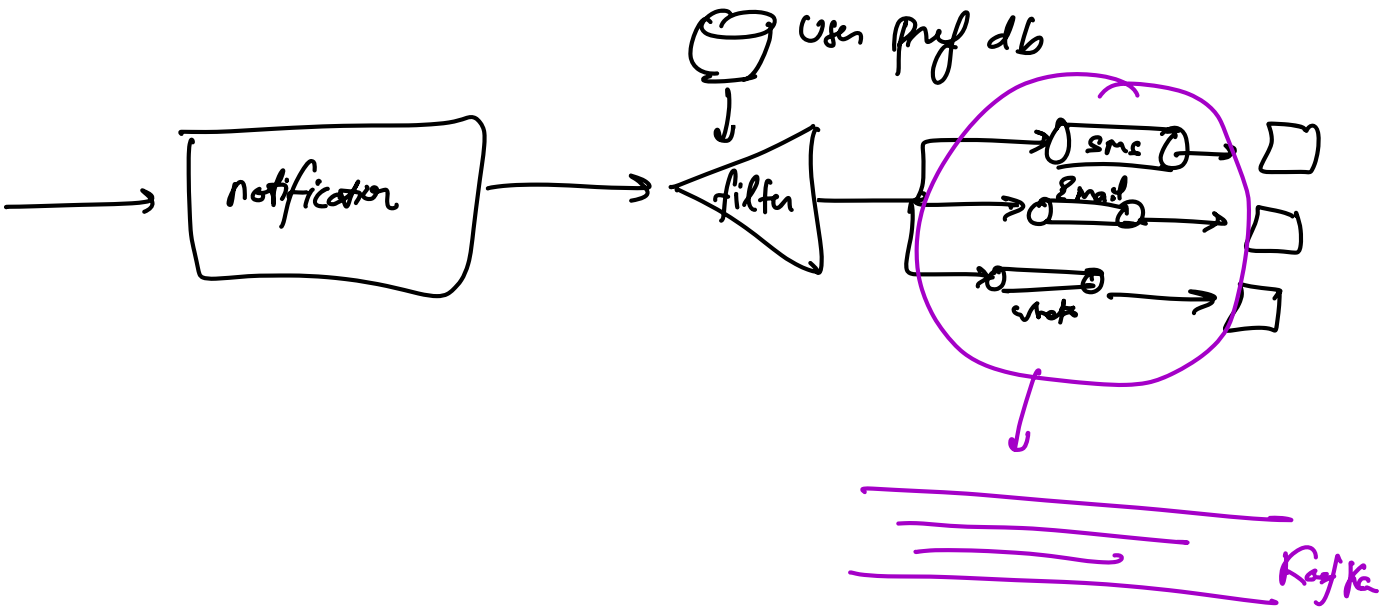


Notification fan out service

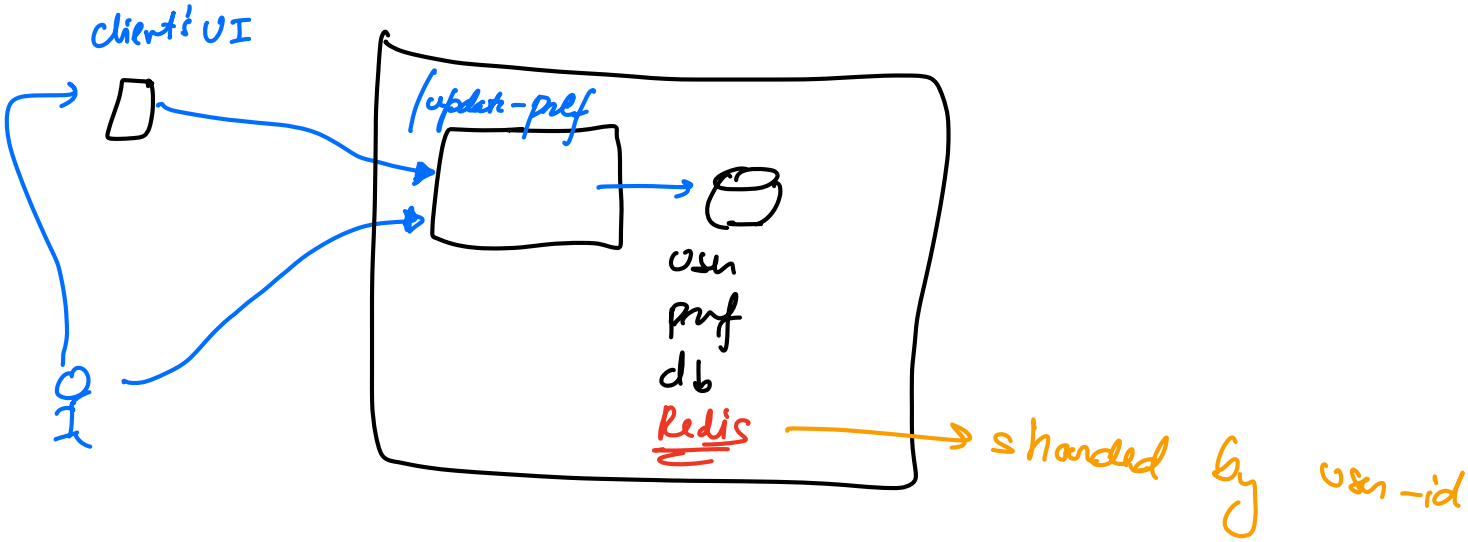
- takes multi-channel API req from client
- check user preference
- drops a msg in Q for each enabled channel.

Msg Q is partitioned primarily by channel/topic
sms/email/whatsapp

Persisted Q → should not lose msgs



User Preferences



2 B users 10 channels

user : id : channel : type : Yes / No / max per day

10 B 4 B → 14 B / entry

2 B * 10 * 14 = 280 B bytes = 280 GB of data

as of 2024, max system config?

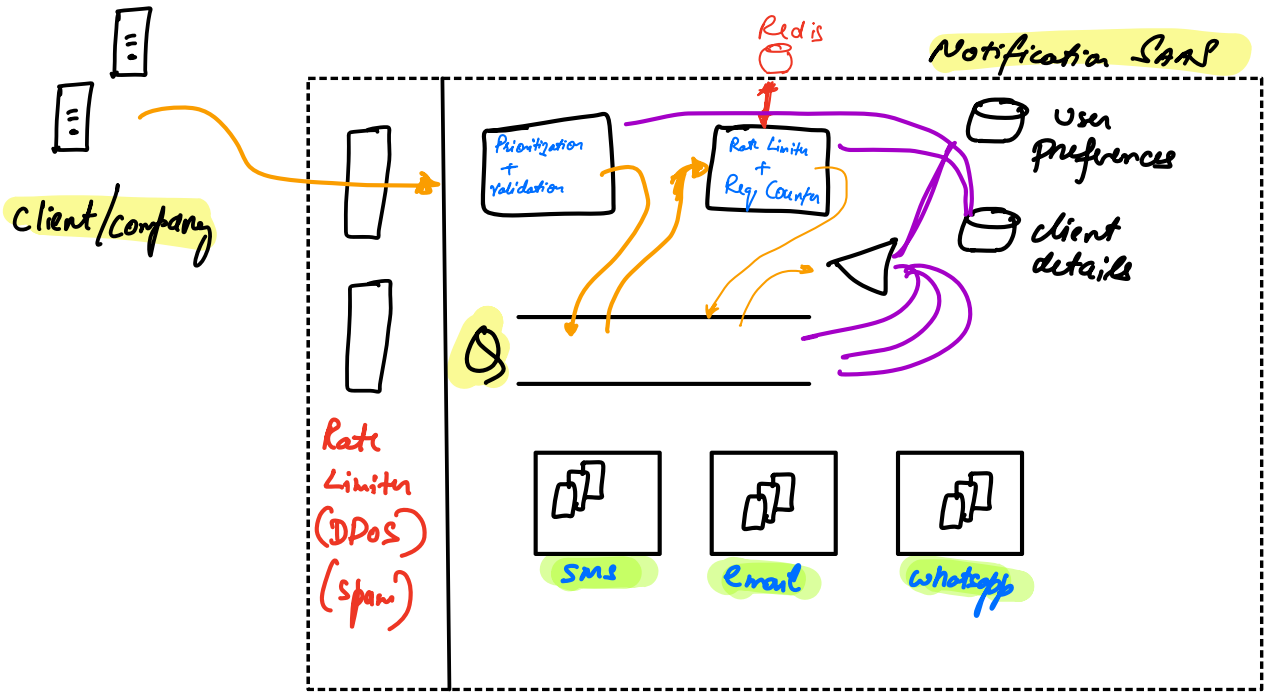
RAM 12TB

CPU 300 cores / 600 threads

Disk 2PB

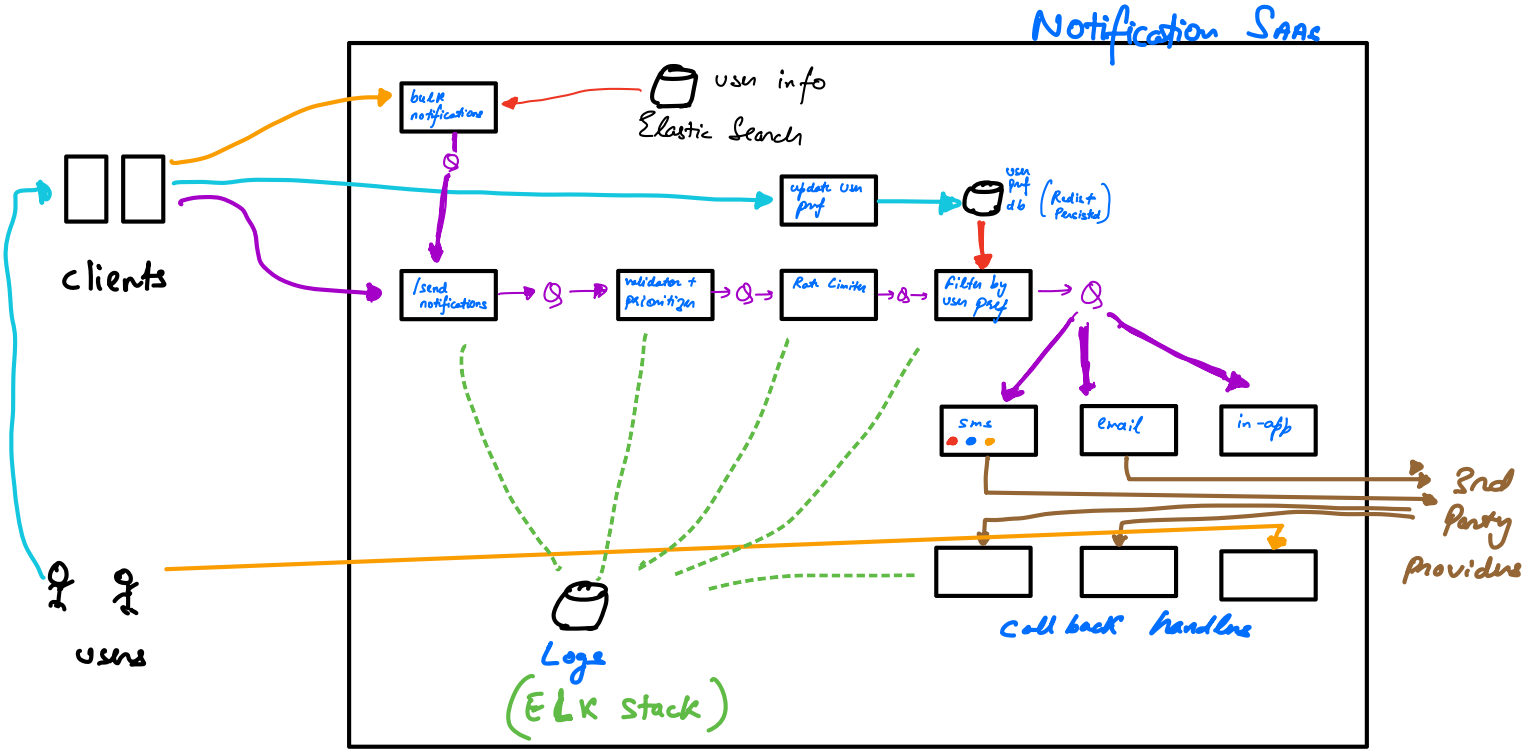
N/W 10Tbps

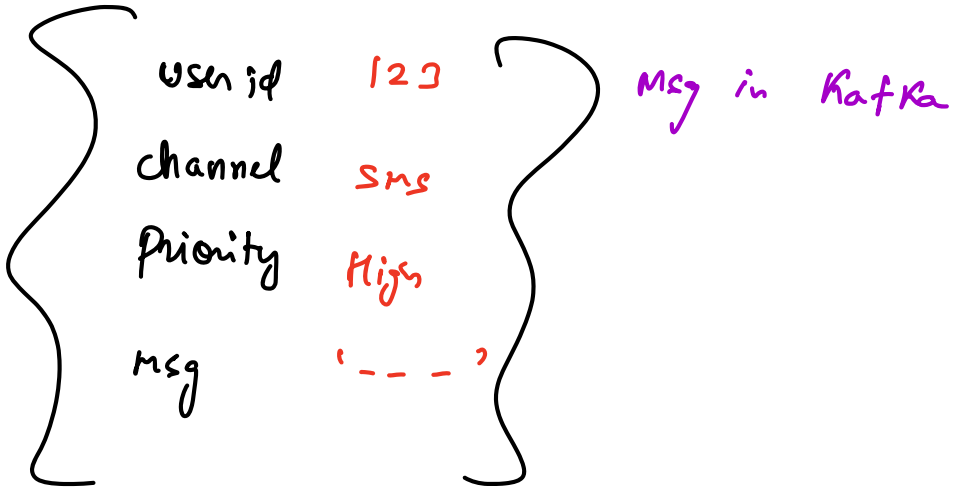
Including Limits



Notification service also applies rate limiting based on

- client profile (VIP/amount/..)
- User preference.


Notification SAAS
usr info
Elastic Search
bulk notifications
update usr prof
usr prof db (Redis + Persisted)
clients
/send notifications
validator + prioritizer
Rate Limiter
Filter by usr pref
sms
email
in-app
3rd Party Providers
users
Logs
(ELK stack)
call back handlers


usr id 123
channel sms
priority High
msg '---'
Msg in Kafka