Instructor: https://www.linkedin.com/in/agarwalpragy
Class Duration: 2.5 hours
15 mins break midway

Agenda

1. **Understand Scalability** - what is it, and why should we care?
    a. What challenges occur when going from 1000 users to Billion + users
    b. How do large systems manage such scale
2. **Approaching a High Level Design problem**
    a. Problem Statement
    b. Requirements Gathering
        i. Functional
        ii. Non-functional
    c. Scale Estimation
    d. System Design
3. **Notification Systems in depth**

Non-agenda

1. No code will be written
2. No systems will be deployed

A Taste of High Level Design

Google - Staff Engineer

- **Location:** India - Hyderabad, Bengaluru, Pune, ..
- **Salary:** up to 3 Cr per annum!
- **Experience:** 10+ years of experience

HLD Interview Question

Given a file containing strings, sort the strings in dictionary order.

Example

input

```
cat, dog, apple, laptop, class, high, level, design
```

output

```
apple, cat, class, design, dog, high, level
```

Just use the built-in sorting function!

```
with open('data.txt', 'r') as f:
    lines = f.readlines()

sort(lines)
```

Catch: there is 50 petabytes of data

- **Bit:** fundamental unit of information `(0 / 1)`
- **Byte:** **8** bits
- **Kilobyte:** **1000** bytes = **10**^{3} bytes
- **Kibbibyte:** **1024** bytes = **2**^{10} bytes
- **Megabyte:** **1,000,000** bytes = **10**^{6} bytes
- **Gigabyte:** **10**^{9} bytes
- **Terabyte:** **10**^{12} bytes
- **Petabyte:** **10**^{15} bytes

50 Petabytes = 50,000,000 GB of data!

Q: Will 50PB fit in your RAM?

Absolutely no!

Q: Will 50PB fit in your HDD?

Actually not even the HDD is large enough.
No!

Q: How will this data be stored?

Distributed Computing.

Thousands of servers distributed across the globe.
This data will be stored in a ***sharded+replicated*** manner across those servers.

Q: What can go wrong?

You can't even open the file directly. You have to talk to those thousands of servers.

1. Memory limits of individual machines (bypassed by distributing)
2. Network issues
   a. Slow network causes delays
   b. Wire is cut - network is down
   c. Machine itself might be down
3. Maintenance issues
   a. Hardware problem
   b. Deployment ongoing
4. Security issues
   a. Malicious actors
5. DDoS attacks
6. Heterogeneity - different machines have different software/hardware configurations

Despite all of these challenges you have to solve the task!

"Simple tasks become extremely difficult at scale"

High Level Design

- Advanced topic of software engineering
- how to go from 1000 users to a billion users
- what unique challenges arise at massive scale, and how to deal with them.

Approaching a High Level Design Problem

DSA Interviews

- very well defined
- know exactly what is being asked
- you have access to example testcases (you know what the correct answer is)
- you also know the size of the inputs (array length is 10^5)
  - you have an idea of the kind of solution that will work
  - ~~$O(n^2)$~~
  - `O(n log n)`

Design Interviews

- extremely vague
- "design twitter"
- you have to understand what the interviewer wants
- understand the problem
- understand the core challenges

- focus on the key tasks
- deliver the design
- all this in 45 mins

The 5-step approach

(20-30 mins)

1. Problem Statement
2. Functional Requirements
3. Non-functional Requirements
4. Scale Estimation

only then can we actually

(25-15 mins)

5. System Design

Solution (system design) comes at last. First you've to understand the problem!

Problem Statement

" Design a notification system "

Company: *teddy.com*
sells soft toys online (e-commerce store)

Company: *scaler.com*
teach advanced software courses (ed-tech company)

Q. Do these companies need to send notifications to its users?

Yes!

- ***teddy.com***
  - order updates
    - payment success / failure
    - order is shipped
    - delivery notifications
  - product updates

        - out of stock
        - back in stock
        - new product launches (in-app)
    - marketing updates
        - sale
        - discount coupon
    - auth pipeline
        - OTP   (sms/email)
        - password reset notifications (email)
- ***scaler.com***
    - auth pipeline
        - OTP
        - password reset
    - event reminders
        - broadcasts: notify all the people enrolled in today's masterclass

Every large scale company would need to send notifications to connect deeply with their users.

Q. Is building the software-tech for large scale notification systems the "core" business of this company?

Is sending notifications why "scaler.com" exists?
Is sending notifications why "teddy.com" exists?

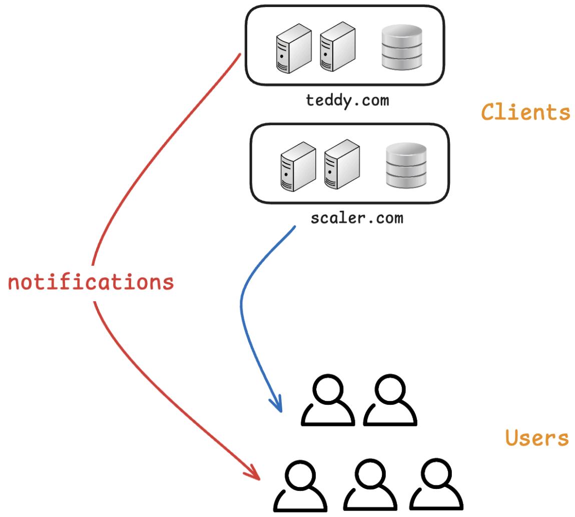
No.

The core business of these companies is different.
Notifications are just something that we need. ***It's a dependency.***

It is difficult even for tech companies to build robust dependencies like notification systems.
Such companies make use of other SaaS products

- notification systems
- chat applications
    - purchase a license for slack/MS teams/ ..
- payment gateway
    - integrate razorpay / stripe

Software as a Service (SaaS)

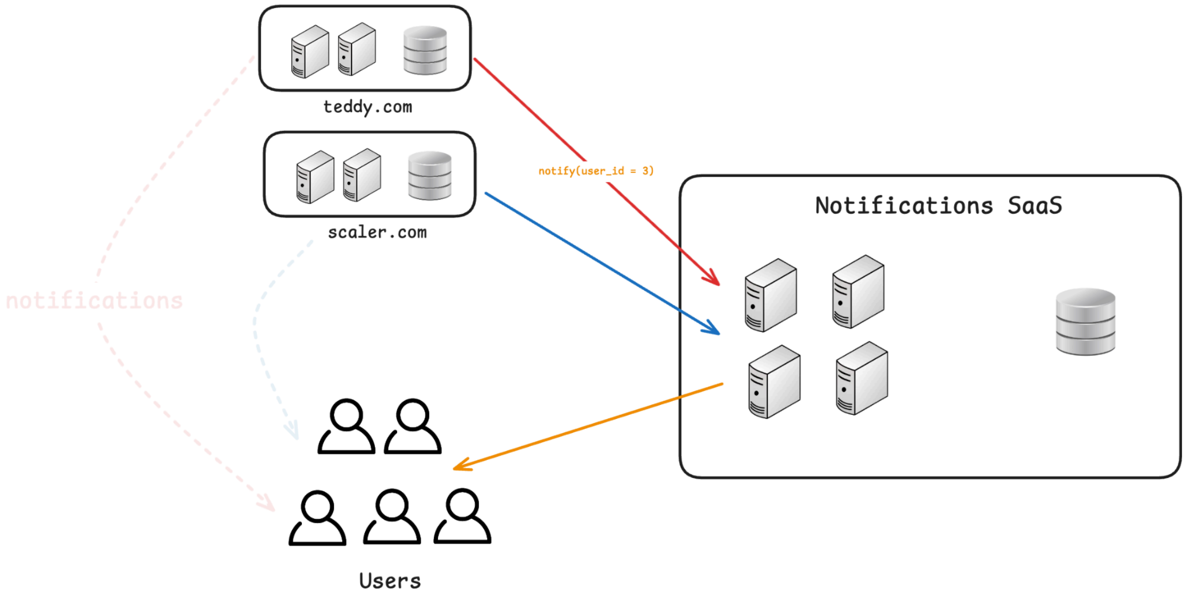


Clients (teddy.com) want to send notifications to their users. But they don't have the tech infra to support that.

Company: notifications.dev
Core business: enable other companies to send notification to their users

This is a hard problem.

We will be building a **" Notification SaaS "**



Functional Requirements

" features/functionality exposed to the users "

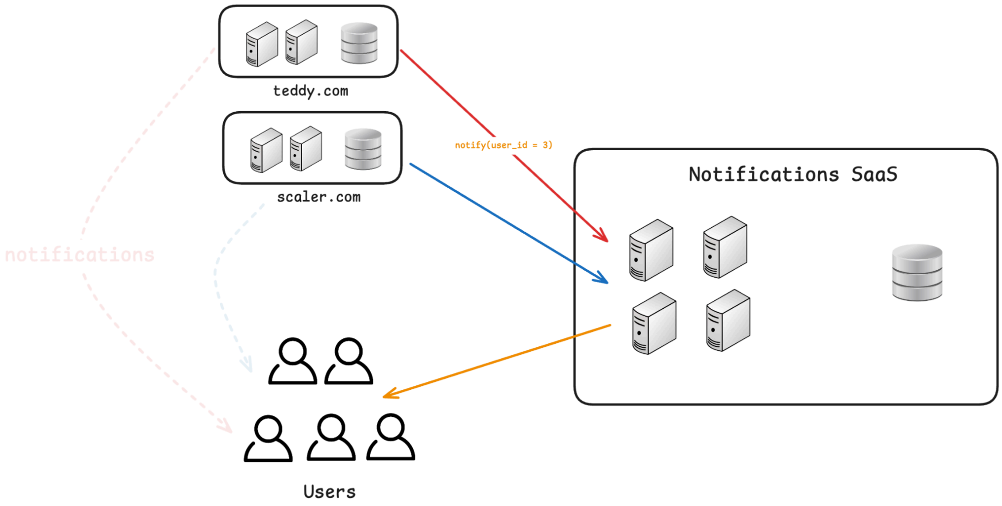
Minimal Viable Product (MVP)

proof of concept / blueprint / prototype

- **minimal:** no fancy features (minimal set of requirements needed to be viable)
- **viable:** it should be usable (not broken, should demonstrate the core concept)
- **product:** solves a genuine problem for someone

When you're asked a question, if you jump to an answer, you've the wrong answer!
The response to any question should involve thinking.

Actors: who will be using this product?



1. **Clients**
   organisations like *teddy.com* or *scaler.com*
   clients will send notifications to **their** users via **our** API

2. **Users**
   people (human beings) that will be receiving these notifications

API: Application Programming Interface
Inputs & Outputs of a system - how can an external/internal entity interact with the system

Whenever I ask people to think of requirements - they start participating in a Race - they think that the more requirements you say, the better you will do in interview.
This is false!

In reality, you only want a small set of requirements.

Non-Requirements

1. ~~Good & intuitive User Interface~~
    - will this affect our backend design? No
    - does this UI have anything to do with the "Scalability" part? No
2. ~~Account Management~~
    - every system needs
        i. registration
        ii. login
        iii. logout
        iv. password update / password recovery
    - but this won't effect our scalability

these are common requirements that every web service has.

API: Application-Program *Interface*
what are the inputs, and what are the outputs

Thinking about the API for the requirement will help us understand the requirement better.

Client Requirements

1. **A client should be able to send notifications to its users**

```
notify(client_id    [uuid]    supplied via API key
       user_id      [uuid]
       message      [json]
      ): ack / error
```

Note: I've not specified the protocol (REST/RPC/websocket/..)

i. bad idea to specify the exact phone number of the user in the request
    - because it can change
        - sms: phone number
        - email: email id
        - slack notifications: slack handle
        - whatsapp notifications: ...

    instead, just use `user_id`
    the clients will maintain details about the users
    user_id => phone/email/…

ii. bad idea to determine the exact message schema right now
    - the schema might change - we need to keep it flexible

    instead, just keep it as a **schemaless json object**

Question: If we use `user_id` in the notify() API, then how will our backend know what's the email/phone corresponding to this user_id?

2. **The clients should be able to store** some **data about their users in the Notification SaaS database**
   so that we can map `user_id` to phone/email/...

   Sensitive Data? We will not sharing any sensitive data about our users - just enough data to send notifications easily. Apart from that, these companies have binding legal contracts.

   CRUD api for managing user data
   create/read/update/delete

3. **Clients should be able to send notifications via multiple channels**

   i. SMS / Email / Push notifications
   ii. **these channels should be extensible/pluggable**
   tomorrow, if I decide to add a new channel (whatsapp notifications), then I should be able to do that without having to redesign the backend architecture
   iii. we should be able to freely add/remove channels

   " If we have to redesign the system every time the requirements change, it's a bad design! "
   Your designs should be robust - they should be flexible to incorporate changing requirements!

4. **Clients should be able to broadcast a notification to a large group of users**

   for example, when Scaler wants to send masterclass reminders, we want to notify all people (>10,000) that are enrolled in the upcoming masterclass

Bad designs

- ~~List of user_ids~~

```
broadcast(client_id   [uuid],
          user_ids  list[uuid],
          message
         ): ack / failure
```

List of user ids can get extremely large!

Elon Musk has 200 million followers on Twitter
Whenever Elon tweets, we must broadcast the notification to 200M users.

size of `user_ids list[uuid]`? **1.6 GB**

```
[0a8772-1238abc33-1238abc33-1238abc33,
0a8772-1238abc33-1238abc33-1238abc33,
0a8772-1238abc33-1238abc33-1238abc33, ...]
```

```
8 bytes / id  * 200 million id
= 8 bytes * 200 million
= 1.6 GB
```

- ~~User groups~~

  first, create multiple groups
  Group of all followers of Elon Musk. Group of all followers of Selena Gomez,
  ..
  Then use `group_id` to broadcast

  issues:
  - Not every client will have the same "group" structure
  - create a new group for every twitter user

  groups will also not work

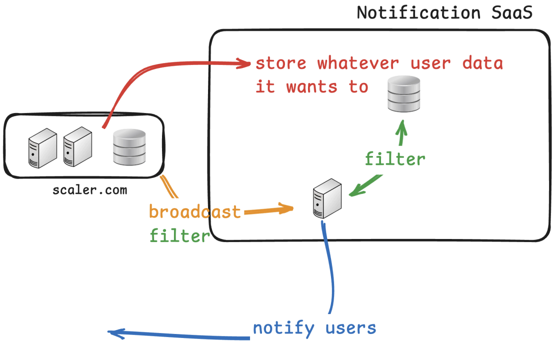
good design

```
broadcast(client_id     [uuid],
          filter_query  [string]
          message       [json]
         ): ack / failure
```

The `filter_query` will be executed on the user data that the client has provided us with via the user management API.

Using the `filter_query`, we will find a list of users (that match the query) to broadcast the notification to.

example:
filter = `"all users that follow [user_id=1234]"`



User Requirements

1. **User should be able to receive notifications**

    a. **SMS / Whatsapp / Email:** our backend systems don't have to do anything - this will be handled by 3rd parties - these will be just be external API calls

    b. **in-app notifications:**
    when the app user opens the app and connects to wifi, the app will call our backend API to establish a 2 way persistent connection (eg. websocket connection)
    ```
    connect(user_id): connection
    ```

2. **Users should be able to control the types of notifications that they receive.**

    User should be able to enable/disable notifications:

    a. wrt **channels**: allow SMS but no emails
    b. wrt **priority**: allow OTPs not no marketing
    c. wrt **company**: allow from Scaler but not from Insurance company

    ```
    setPreferences(user_id        [uuid]
                   preferences    [json]
                  ) : success / failure
    ```

Future Scope

Need to complete the design during the 45 min interview. Don't bite more than you can chew - keep the requirements minimal.

- ~~security~~
    - do NOT build security unless you're a security expert
    - 99% of all security holes happen because non-experts try to do security!
- delivery acknowledgement (blue tick / double tick)

- compliance

Non-Functional Requirements

" Requirements that are not for our users, but for our backend "

Common

- Security
- Observability
- Low Latency
- Consistency / Availability
- ...

Specific to Notifications

1. **Priority**
    - Prioritize OTPs / Financial notifications over marketing notifications
    - *does this mean marketing notifications will always be slow?*
      No.
      When the system is working fine, all notifications will have low latency
      When the system is overwhelmed, then we will deliver OTPs before we deliver marketing notifications

2. **Rate Limiting**
    - DDoS protection
      Distributed Denial of Attack

    - Fair Usage Policy (FUP)
      prevents some users to hogging up all the resources

    - enforcing User Tiers
        i. free user: can only send 5 notifications / day
        ii. basic user: paid (5$) can send 5000 notifications / day
        iii. pro user: paid (100$) can send 50,000 notifications / day
        iv. enterprise: paid (contact sales) unlimited notifications

Scale Estimation

Why do we care about the scale?

Designing a system for 100 users is very different from designing a system for 1 billion users.
The scale determines the design!

Goals

1. # requests / sec
2. size of data
3. peak load

To get a rough estimate of these values. We don't have to be exact.
" Back of the envelope calculations / guesstimations "

Assumptions

Monthly Active Users (MAU)
we're building for web scale / planet scale
`1 billion users`

Average number of notifications that a active user receives / day
`20 notifications / user / day`

Pareto principle: 80-20 rule.

" 20% of your users will generate 80% of your traffic "

Only 20% of our users will be receiving notifications on any given day.
Only 20% of our users are active on any given day

Estimates

Average Load = 40,000 notifications / second

```
= (20 notifications/user/day) * (20% of 1 billion users)
= (20 notifications/user/day) * (2 * 10⁸ users)
= 4 * 10⁹ notifications / day
= (4 * 10⁹ notifications) / (24 * 60 * 60 seconds)
= (4 * 10⁹ notifications) / (≈10⁵ seconds)
= 40,000 notifications / second
```

```
1 day = 24 * 60 * 60 seconds = 86,400 seconds ≈ 10⁵ seconds
```

Peak Load = 2,000,000 notifications / second

During festivals / global events (covid) the number of notifications can suddenly peak

assume that the peak load is **50x** the average load

```
= 50 * 40,000 notifications / second
= 2 million notifications / second
```

quick 15 mins break.

System Design

Heads-up!

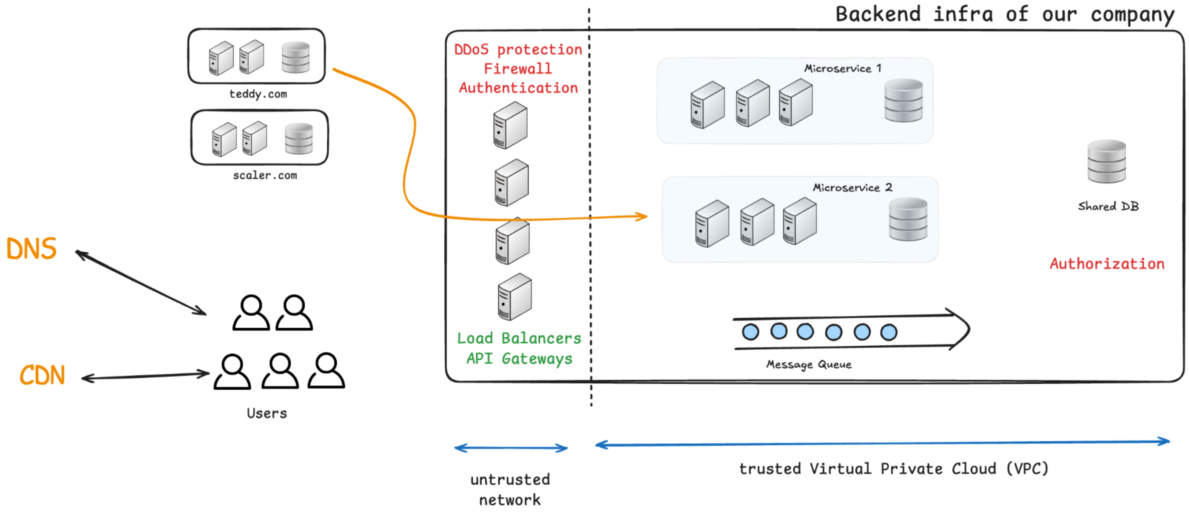
I will be using a bunch of jargon / terminology that might new to you.
Please do NOT get overwhelmed.
Just try to understand that idea behind what is being discussed.

Either ask in chat / just Google the specific term.
It is okay to not understand HLD to a 100% today.

Default Design

1. Untrusted N/W Layer:
    a. houses any security measures we have - rate limiters for DDoS prevention, firewalls, authentication
    b. houses Load Balancer & API gateways
2. Trusted VPC
    a. all of our actual systems and databases are inside this

Any communication with the outside world should pass through the untrusted layer



Basic Terminology

- **DNS:** Domain Name Service. Given a domain name (pragy.dev) it will tell you ip-address of the server
- **CDN:** Content Delivery Network. Responsible for serving most-static content from a server near to the user (edge-node)
- **API Gateway:** responsible for routing the request to the correct microservice
  if I'm sending an email, the request should be send to the gmail API and not the google maps API
- **Load Balancer:** responsible for equal distribution of the load across the servers
  load = both the number of requests and the amount of data
  back in 2020, google had 10 million $^{+}$ servers in total!

- **Auth**
  - **Authentication**
    am I who I claim to be?
    only at the untrusted layer

  - **Authorization**
    do I have the permission to access this resource?
    happens in every service

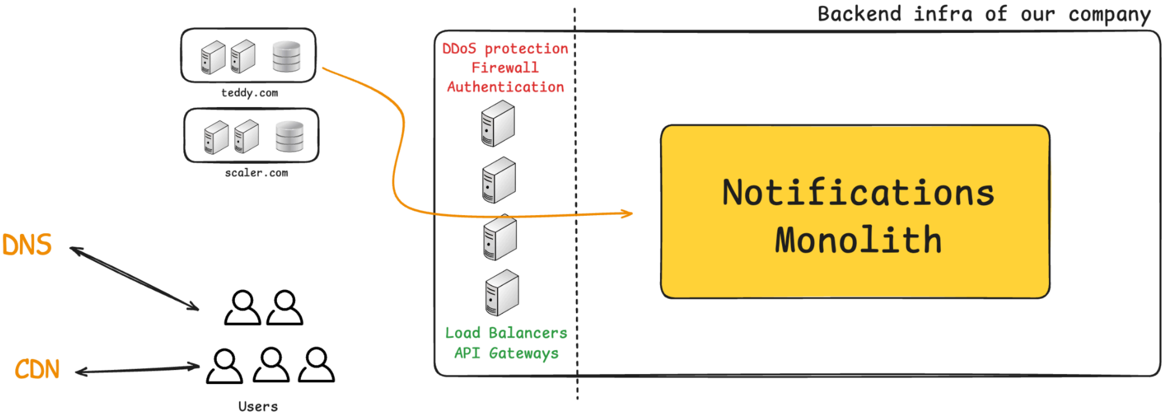
Notification Service

We have a lot of features - channels, priority, rate limits, broadcast, ...

Q. Should all the responsibility be in the notification service?

No!

Monolith



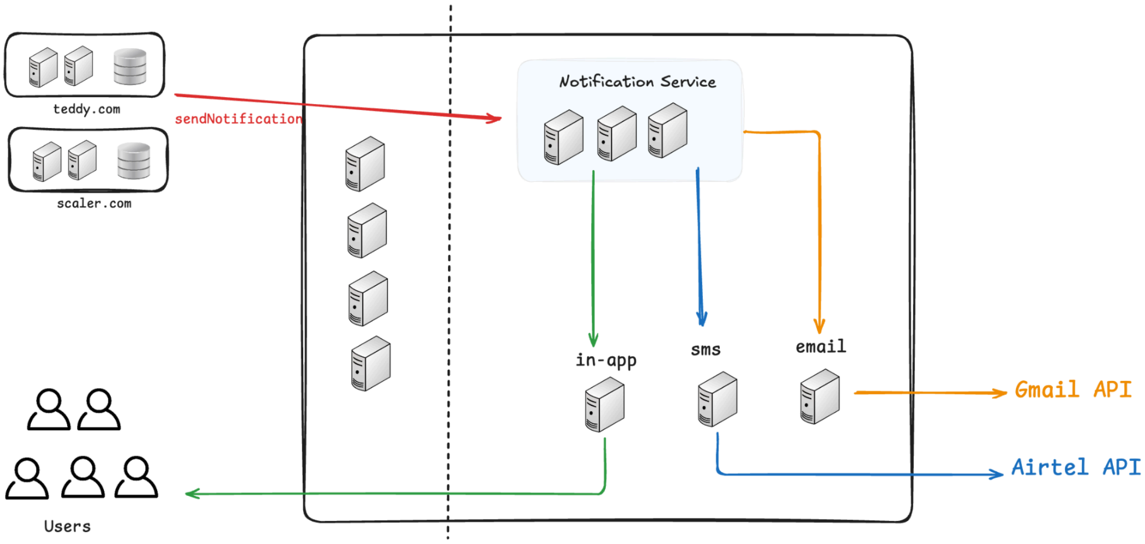
At high scale, monolith are bad

1. can't scale the servers independently
    a. send 100,000 SMS / second, but only 200 slack notifications / second
2. codebase becomes a *"**big ball of mud**"*

Both monolithic and microservices have pros & cons (won't be discussing in depth)

Microservices

We will have different services for different channels.
The notification service will receive a request and will forward it to the appropriate channels



Q: How should the client communicate with the Notification Service?

1. how to enable / disable various channels (SMS / Email ..)

2. how to encode the notification priority
3. the content of the notification

```
notify(client_id    [uuid]  (supplied via API key)
       user_id      [uuid]
       message      [json]
      ): ack / error
```

```
{
    msg_id:  uuid
    type:    'OTP' / 'Financial' / 'Critical' / 'Marketing'
    (priority will be decided based on the notification type)
    channels: {
        'sms': {content: ...}
        'email': {subject: '...'   body: '...' }
        'in-app': {message: ...   icon: ...  action: ...}
    }
}
```

For example, if Scaler wants to send an OTP to Apurva, then..

```
{
    msg_id: 'a3b6-1234-..'
    type: 'OTP'
    channels: {
        'sms': {content: 'Please login using the OTP 1234. This OTP will be valid
fo next 5 minutes'},
        'whatsapp': {content: '...', attachment_url: '...'}
    }
}
```

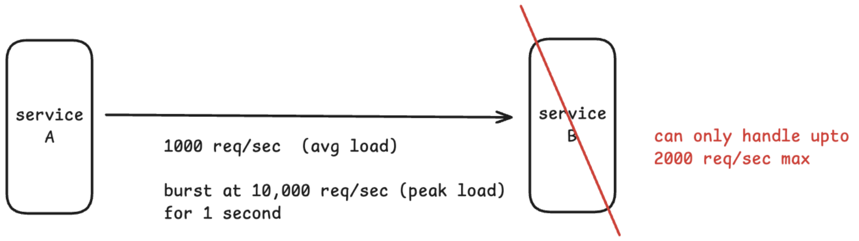
A simple REST API will do for notify(...)

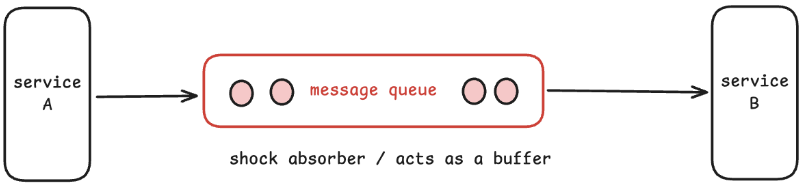
advanced: this REST API endpoint should be idempotent!
Calling the API with the same payload (client_id,user_id,message_id) multiple times should only send the notification once!

Q: How should the communication happen b/w the notification service and the various services for specific channels?

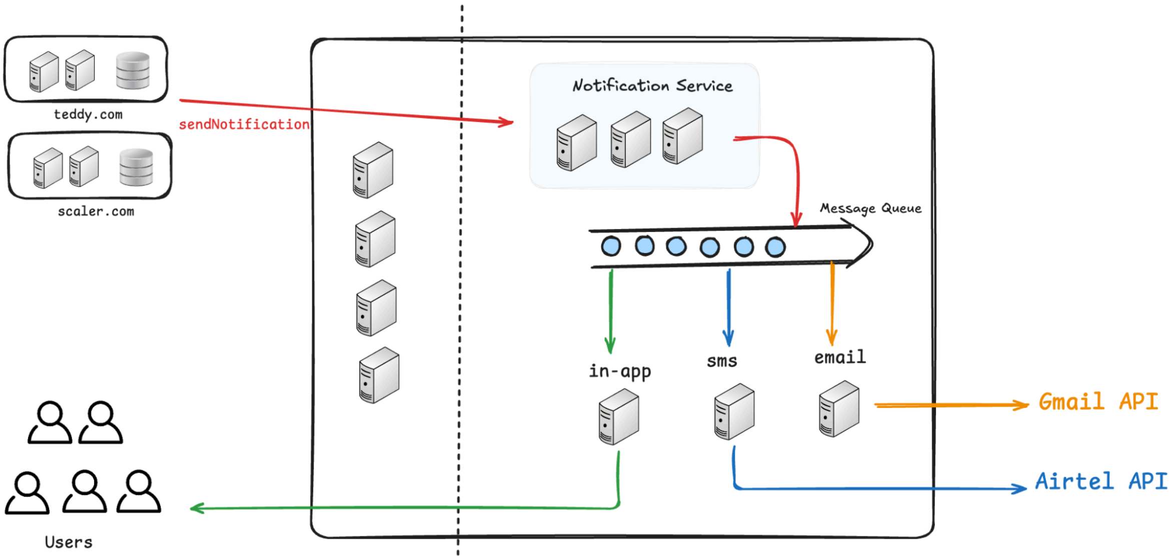
1. Sync (API Call)
    - REST
    - SOAP
    - Remote Procedure Call (RPC)
    - Inter Process Communication (IPC) (very very rare - requires all the microservices to be inside the same server)
2. Async (Message Passing / Pub-Sub)
    - Messaging Queue (Kafka)

We will use an async communication protocol (pub-sub).
Messaging queues like Kafka have a lot of advantages.


service A
1000 req/sec (avg load)
burst at 10,000 req/sec (peak load) for 1 second
service B
can only handle upto 2000 req/sec max



Use a **persistent message** queue (like Kafka) to hold the messages while the various channel APIs are handling them



Choosing the correct Database

1. **SQL (Relational Databases)**
   a. *PostgreSQL, MySQL, Oracle DB, IBM DB2, Sqlite, MSSQL, ...*
   b. **Strengths**

        i. ACID transactions
        ii. Relations & Joins (complex schema)
        iii. Rigid Schema (structured data)
        iv. Indexing
        v. Rich & Complex querying
    c. **Weaknesses**
        i. Do not scale well
            1. no built-in support for sharding
            2. sharding negates most of the strengths
        ii. Difficulty with unstructured data

2. **(no-sql) Key-Value**
    a. *Redis, Firestore, DynamoDB, Memcached*
    b. **Strengths**
        i. very fast (usually in-memory, with optional disk persistence)
        ii. auto-sharding - scale well
        iii. very simple
    c. **Weaknesses**
        i. no rigid schema
        ii. no joins / relations
        iii. no complex query (get/set)
        iv. no search
        v. no indexes
        vi. no transactions

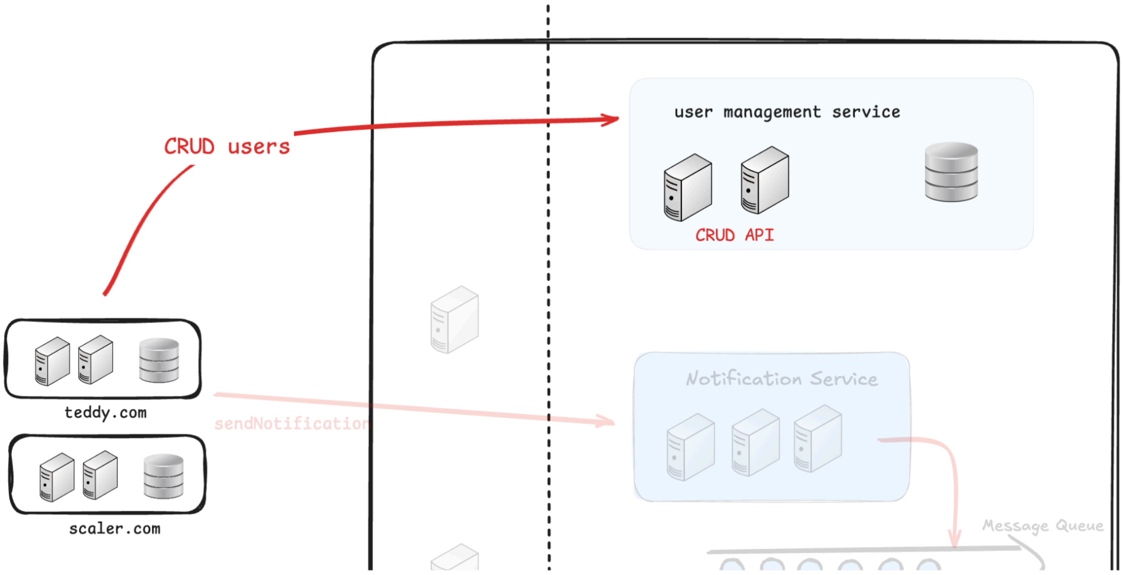
3. **(no-sql) Document**
    a. *MongoDB, ElasticSearch, Couchbase, CockroachDB*
    b. **Strengths**
        i. support semi/unstructured data (JSON)
        ii. support full text search
        iii. index on any top level attribute
    c. **Weaknesses**
        i. no relations / joins
        ii. only local indexes (not global)
        iii. no transactions

4. **Column Family**
    a. *Cassandra, ScyllaDB, HBase, BigTable*
    b. **Strengths**
        i. fast aggregate queries over a single column
        ii. semi-structured data
        iii. very fast inserts (LSM trees)
        iv. support time based pagination efficiently
    c. **Weaknesses**
        i. no joins
        ii. reading a row is slow (because column major storage)
        iii. design upfront

5. **Graph (rarely used)**
6. **Vector (rarely used)**
7. **Object / File Stores**
8. **...**

User Management Service

The clients should be able to add/manage data about their users in the notification saas database



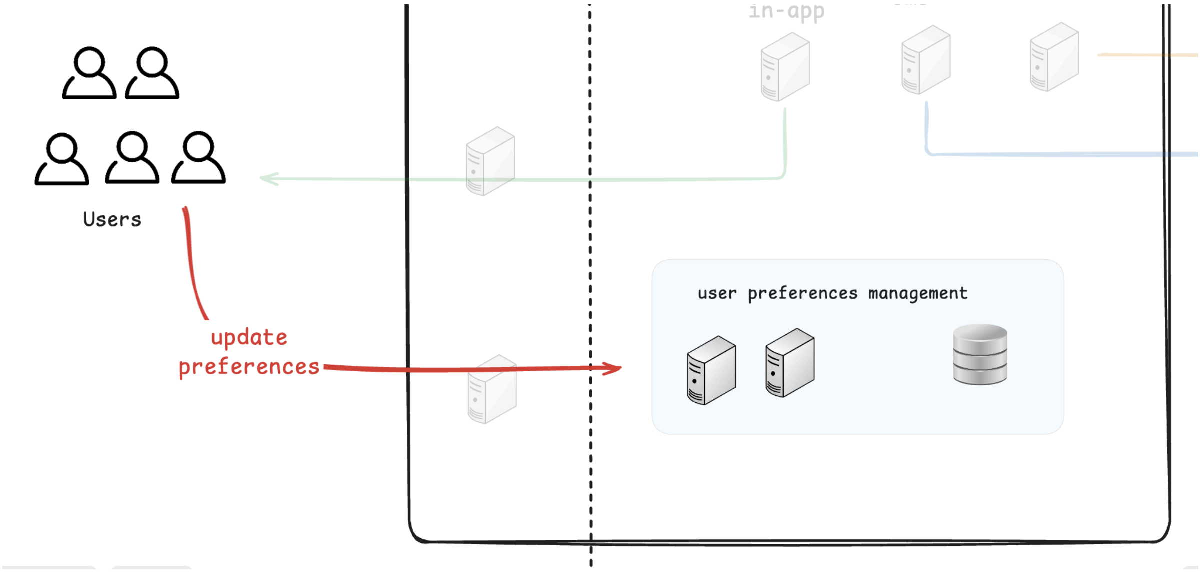
Q: What is the ideal database for the User Management Service (for storing the user data)?

What do we need?

1. Do we know the exact schema of the user's data?
   No. Different clients will have different schemas.
   So we need schemaless
2. Do we need search?
   Yes. We need powerful search for broadcast filtering
3. Complex joins?
   No. We don't even know the schema - so no possibility of joins
4. Do we require transactions?
   No

Ideal Database: Document database with powerful search capabilities – **ElasticSearch**

User Preferences Service



Q: what's the ideal database for the user preferences service?

What do we need?

1. very fast querying (because for every notification that needs to be sent, we have to check the user preferences)

   since we're sending notifications at 2 million / second, this database also needs to be queried at 2 million reads / second

2. complex schema? No

   ```
       sms: true
       email: false
   ```

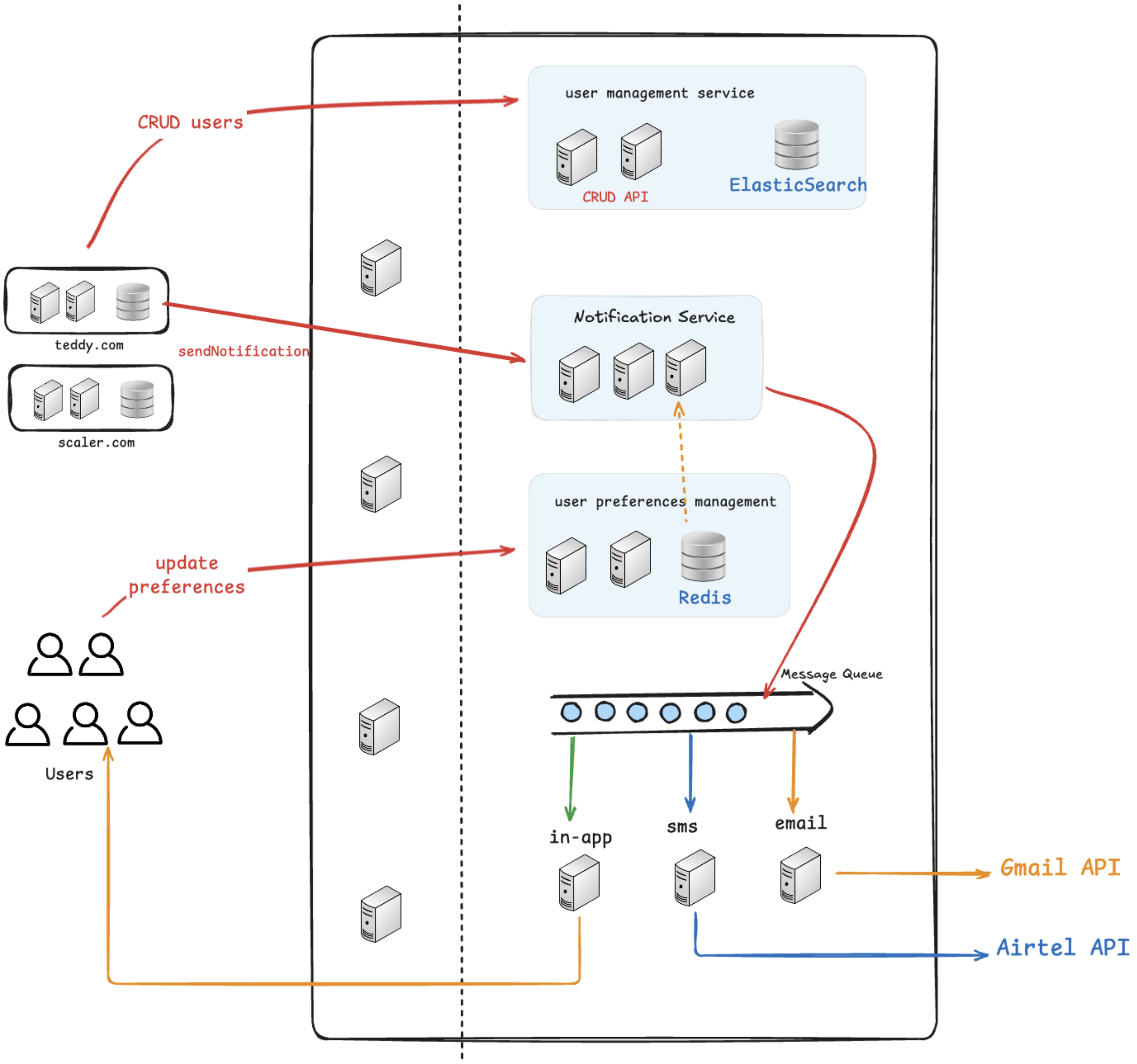
   Can be a simple key-value thing

3. We need persistence, because we don't want to lose the preference data

Ideal Database: A key-value database that support periodic disk persistence – Redis (+persistence mode)

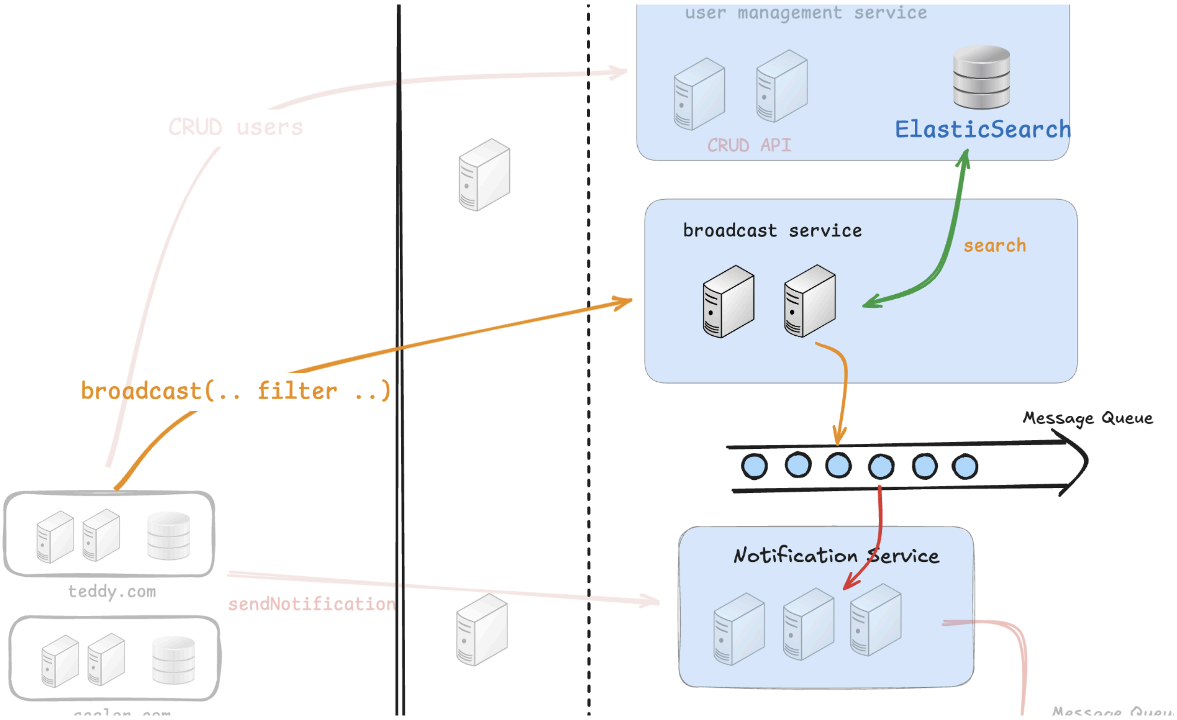
How large is the user preference data?
1 billion users. Each user's preference data is 100 bytes in size
Total data = `1 billion * 100 bytes` = 100GB

Design so far..



Broadcast Notifications

1. Client makes a `broadcast(... filter …)` API call
2. The broadcast service use this `filter` and it will search through the user data stored in elasticsearch (management by the UMS)
3. Elasticsearch will return the list of users
    a. it won't return 200M items in a single response
    b. instead, the responses will be paginated
4. Broadcast service will send these notifications batch by batch via the notification service – it will drop messages in the queue for each user that needs to be notified



Resources - Notification Systems

1. Ultimate Guide on Notification Services - A Technical Overview (2024)
2. Notification Service Design | The Ultimate Guide with Diagrams
3. Building a Notification Service: Top Technical Mistakes
4. Top Non-Technical Mistakes With Your Notification Service

QnA

The user list of scaler will have to be kept syncing everyday...how will that be handled?

Yes, the user list will have to be kept in-sync!

Inside Scaler's backend, in the user signup flow, there will be a "hook"

- whenever a user signs up, (async) make a call to notifications.dev API to add this user's information

When we are doing Scale Estimation in an interview are we expected to calculate all the numbers? and how to make the average and peak assumptions?

Expected to calculate, for each API/microservice that exists in the requirements

1. number of requests
    a. avg/peak
2. amount of data

Avg/Peak - depends on the application

Eg, IRCTC (railway ticket booking) - when tatkal window opens, the load can be 100x of the average load
Big Billion Day / Prime Day - iphone sales

is it Okay to design simple a monolith block first and then change arch based on scale requirement?

If you're a CTO / developer for a small company, and you're building the "product"/"startup" right now..

you won't have a lot of users
even in the near future (next 2-3 years), you will not have 10 million users

You should start with a Monolith - because it easy to build, and it will be cheaper to deploy

If you're a developer working at a large company - Google

whatever product you're building, that product will receive billions of users from day 1

You should start with microservices even on day 1

Practicality beats purity!

Hi pragy if we are using websockets & pub/sub how are we will manage that large scale

In-app notifications will happen via websocket connections

The pub-sub architecture might use persistent websocket connections.

Can you actually handle 1 billion users with websockets?

Q: How many websocket connections can a single server handle?

Whatsapp can handle 2.2 million connections on 1 server (back in 2012)

How many servers to handle 1 billion users?

20% are active on a given day => 200M users
200M / 2.2M => 100 servers