Distributed Email Service

xxx

April 2023

What is the same as class design:

- protocols
- API functions
- send
- receive

what is different from class design:

- data storage
- scaling the system
- search email in your own office

1 Step 1

Understand the Problem and Establish Design Scope

- send and receive email
- fetch emails actively
- search emails by keywords etc
- spam detection

Traffic estimation:
1 billion DAU
send out 10 mails, receive 40 emails, with 50KB metadata size each

$$QPS = \frac{1billion * 10}{10^5 s} = 100,000 \tag{1}$$

store received emails size:

$$1billion * 40 * 365days * 50kB = 730PB \tag{2}$$

2 Step 2

Propose High-Level Design and Get Buy-in
email basics:

- SMTP: a sending protocol
- POP: Post Office Protocol, once send a whole mail body to user, delete it from server
- IMAP: internet mail access protocol
  only downloads header when connection is slow
  does not delete mail from server

DNS:
server size is large, decide which server to use
defined by priority number
So:
send protocol SMTP, get protocol IMAP/POP
API design:

- POST/v1/messages
- GET/v1/folders
- $GET/v1/folders/: folder\_id/messages$
- $GET/v1/messages/: message\_id$

distributed mail server architecture:
two ways of connecting to server:

- web servers: user send get request to server
- real-time servers: push email to users in real-time

Storage layer:

- metadata DB
- attachment storage
- distributed cache: users get recent mails
- search storage: has its own indexing and search storage

email sending flow:
webmail -¿ load balancer -¿ web server -¿ outgoing queue -¿ SMTP -¿ internet
email receiving flow internet -¿ load balancer -¿ incoming queue -¿ mail processing (spam etc) -¿ real-time server / web server -¿ webmail

3 Step 3

choose database: data consistency, reduce disk I/O
need a lot read / write
write: send, delete causes re-indexing

- relational database: not suitable for this large mail situation
- distributed object store (s3): no good indexing
- NoSQL, e.g. Google bigTable

Process spam:

- dedicated IP
- classify email
- email sender reputation
- ban spam
- feedback processing

Search:
characteristics:

- scope: user's own mail box
- sorting: by time, content etc
- indexing should be near real time

Possible method: elastic Search or LSM
Scalability:
different data center in different countries