#ASLI ENGINEERING

Himeji - Central Authorization at Airbnb

SWIPE

BY

ARPIT BHAYANI

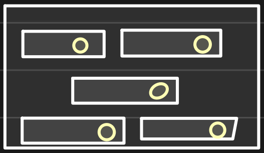
Central Authorization Service @ Airbnb

Only checking for authencation is not enough
we need granular access control for
defining who can do what on the platform

eg: can user 1 edit post p1 ?
can user 2 access wifi info of property p2 ?
can user 3 read file present in folder shared with group g3 ?

Authorization in monolith

Things are simpler when it is monolith
all the checks are well within the codebase
and part of the business logic



We need a central auth service when we adopt microservices ...

Himeji - Central authorization @ Airbnb

Authorization is modelled on

Entity	Relation	Principal
↓	↓	↓
post: 123: comment	WRITE	U:123

user 123 has write privileges on post: 123 's comment

This information is stored in database as a tuple

entity # relation @ principal

Entity is a three-fold info : type : id : part ↙ optional

post : 123 : comment # WRITE @ user : 123

How are the rules defined and configured ?

Writing one tuple for each permission for each entity will explode data

Hence, we need to leverage rules, transitivity and set theory

eg: WRITE → read and write } on a listing
OWNER → read, write and owner privileges } on airbnb

Hence, we define relation on listing that defines access hierarchy as

```
LISTING :
  # WRITE :      ← while checking 'WRITE' relation on a listing
                   check 'UNION' of WRITE or OWNER relation
    union :
      - # WRITE
      - # OWNER
  # READ :       ← while checking 'READ' reation on a listing
    union :        check 'UNION' of READ and WRITE relation
      - # READ         ( transitively OWNER too )
      - # WRITE
```

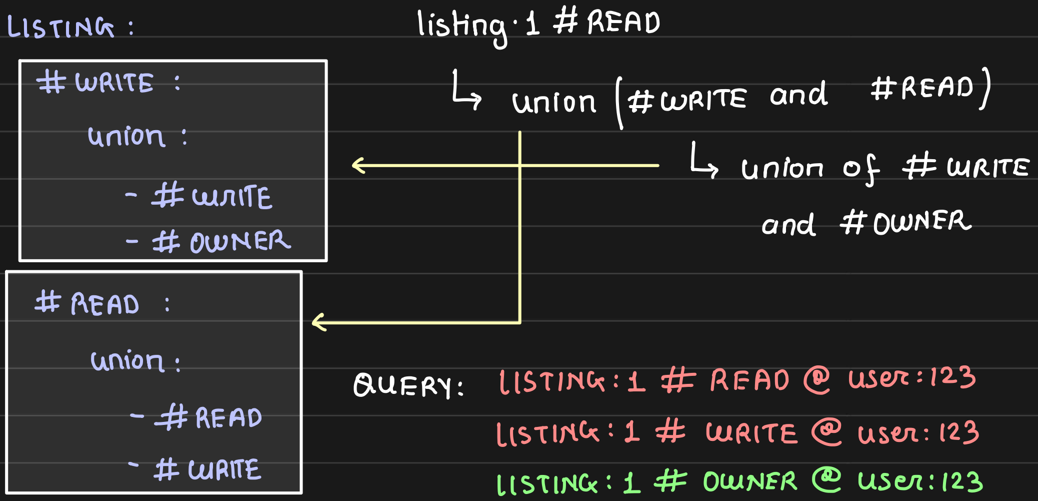
Say, user:123 is owner of listing:1, the database will have one entry

listing:1 # OWNER @ user:123

Say, we want to check

check (listing:1, READ , user:123) ↗ should return True / False

Because of the rule we defined



LISTING :

# WRITE :
union :
- # WRITE
- # OWNER

# READ :
union :
- # READ
- # WRITE

listing:1 # READ

↳ union (# WRITE and # READ)

↳ union of # WRITE and # OWNER

QUERY: LISTING:1 # READ @ user:123

LISTING:1 # WRITE @ user:123

LISTING:1 # OWNER @ user:123

Because our DB contains entry for listing:1 # OWNER @ user:123 the evaluation of

check (listing:1, READ , user:123) → TRUE

but what if access to something depends on the existence of some other entity?

Say, we want to allow people to read location if they made reservation

```
LISTING :        ← allow reading location of listing
  LOCATION :
                        If user is the owner of the listing, or
    '#READ' :  ↙
                                ↓
      union :
                                                         For the entity in question
        - #OWNER                                                   ↓
        - LISTING : $id # RESERVATION @
                    ↑
                                   Reference ( Reservation : $rid # GUEST )
```

Say, we have following entries in database

listing: 1 # owner @ user·123

listing: 1 # reservation @ ref (reservation: 500)

reservation : 500 # guest @ user: 456

Say, we want to check

check (listing: 1: location, READ, user: 456) ↙ should return True / False

Query: listing: 1 # owner @ user: 456

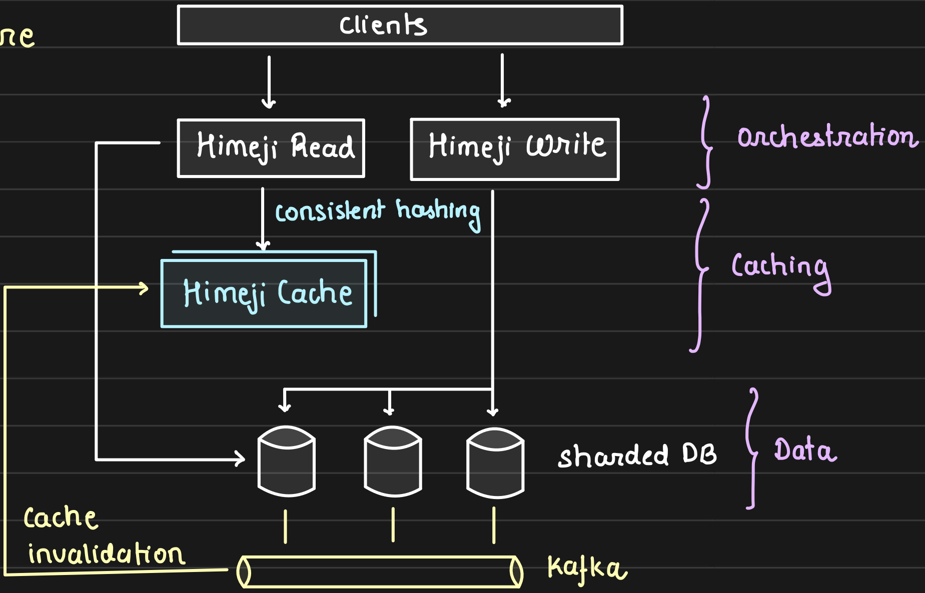
listing: 1 # reservation

Then: matched references

reservation: 500 # guest @ user: 456

ARPIT BHAYANI

Architecture



Orchestration layer

- receives request from client
- forwards request to cache using consistent hashing
- computes response as per config and responds

Caching layer (98% hit rate)

- sharded and replicated
- Consistent hashing determines data ownership

Data layer

- logically sharded persistent database
- mutations in data invalidates the cache

ARPIT BHAYANI