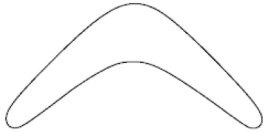
Boomerang

Scalable Internet Services



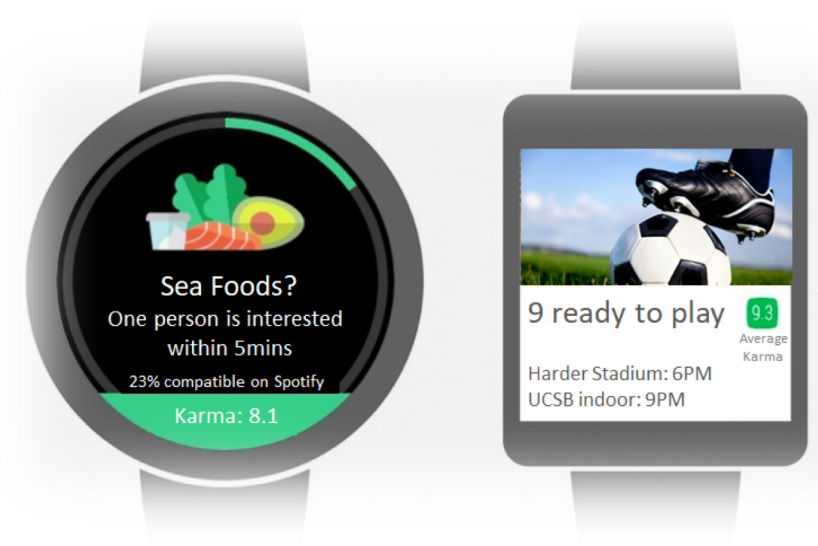
2017

Ali Abbasinasab

Boomerang

Mood, Match and Meet!

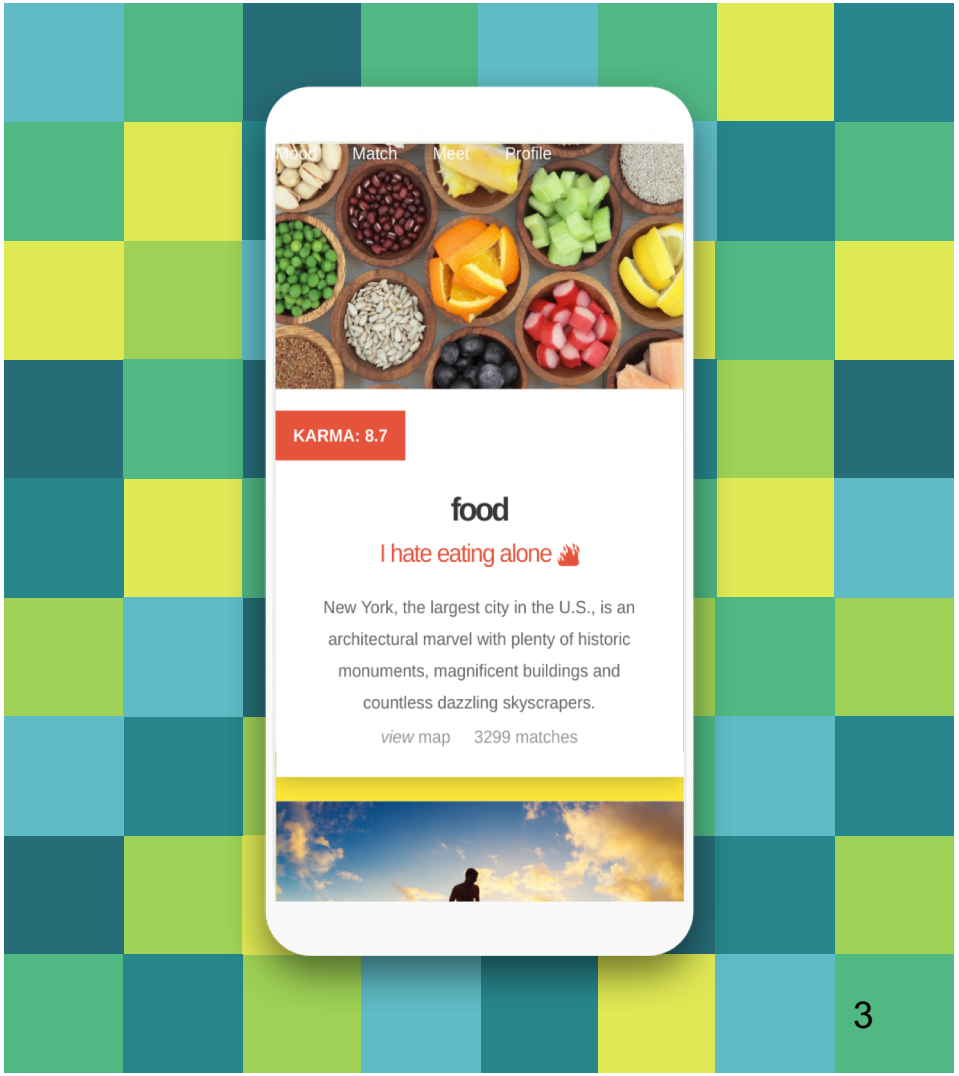
- JavaScript
- Bootstrap
- Nginx
- Ruby
- Rails
- PostgreSQL
- AWS EC2



GitHub: https://github.com/scalableinternetservices/Boomerang

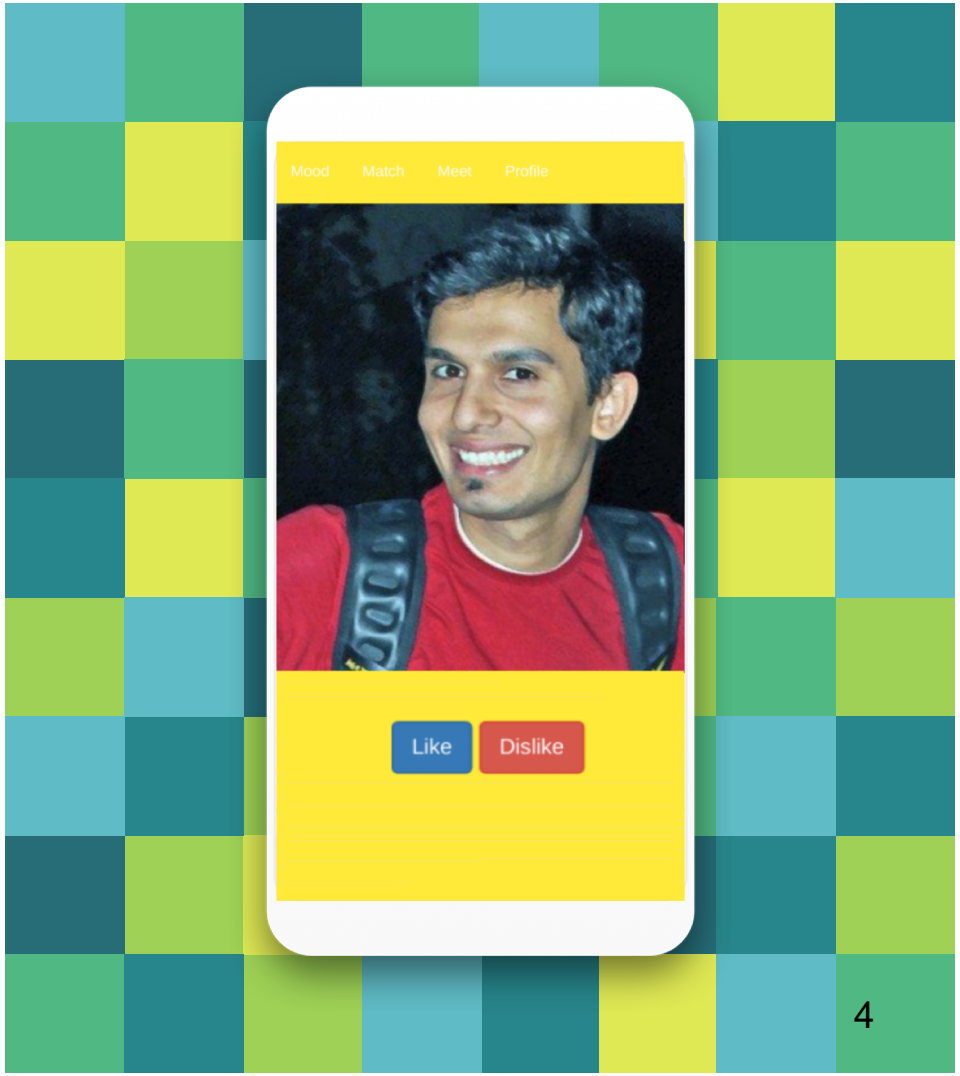
Boomerang

What mood are you in?



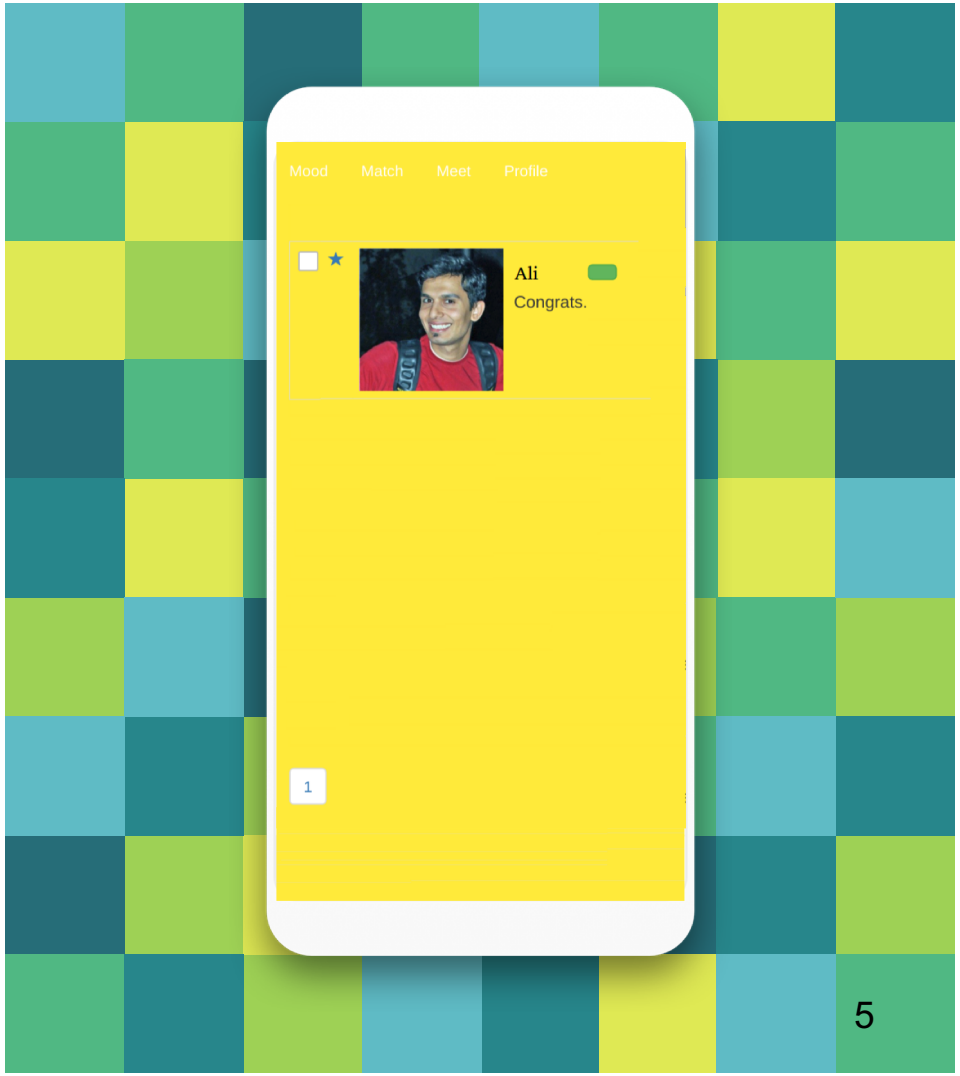
Boomerang

This person is in the same mood. Interested?

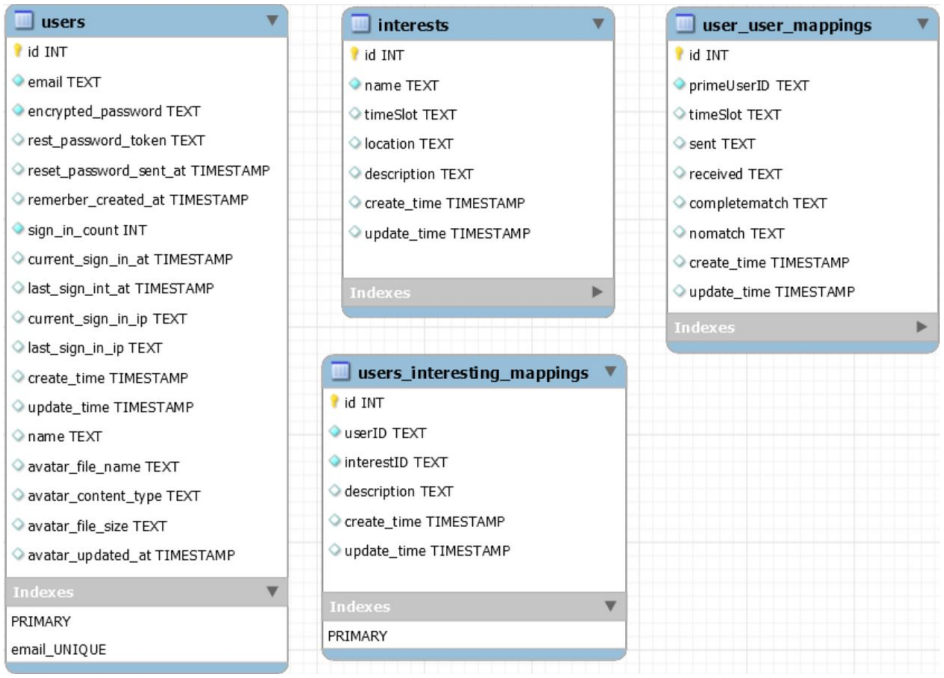


Boomerang

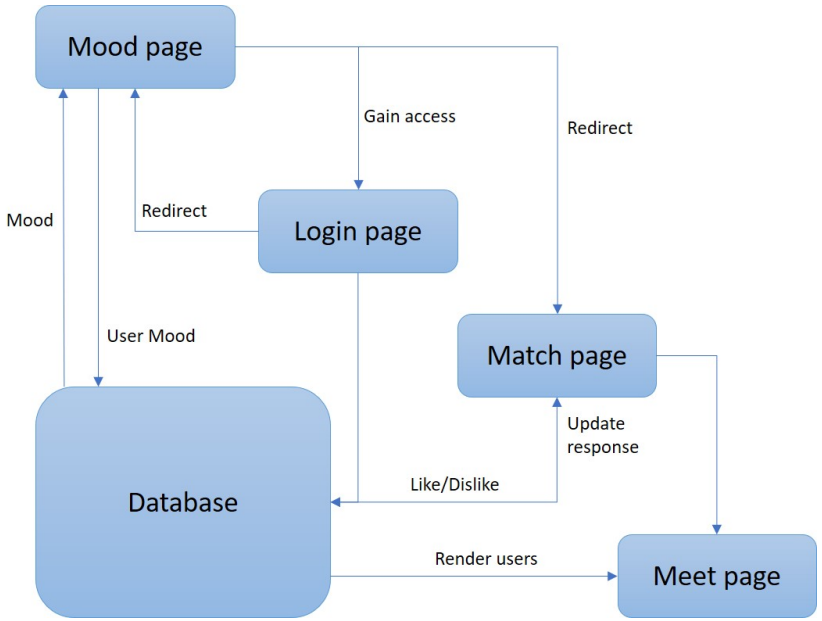
Matches!



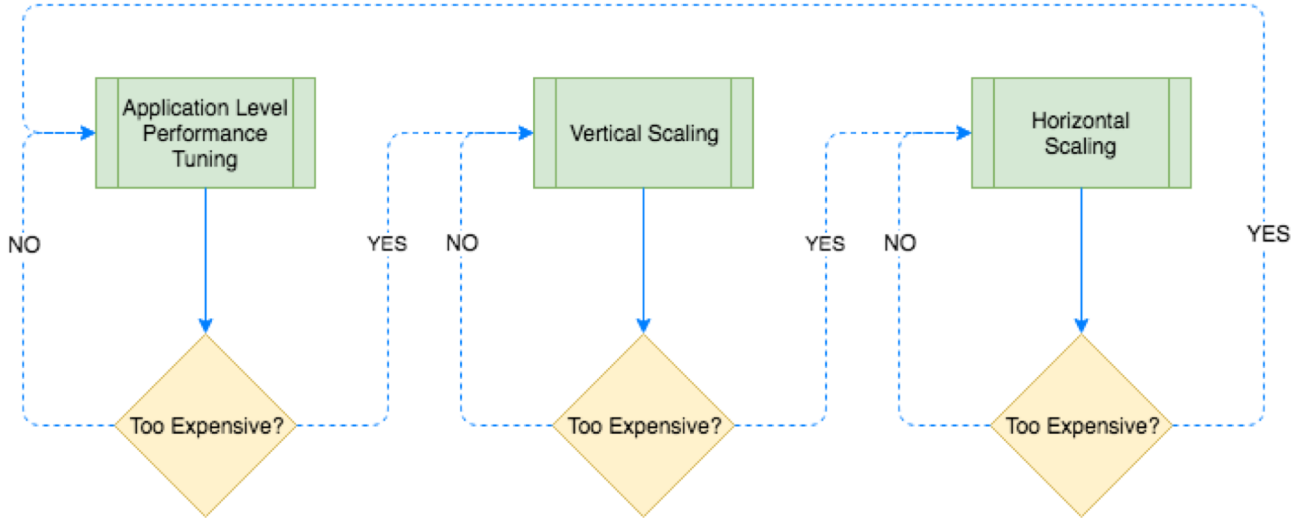
Models


users
id INT
email TEXT
encrypted_password TEXT
rest_password_token TEXT
reset_password_sent_at TIMESTAMP
remerber_created_at TIMESTAMP
sign_in_count INT
current_sign_in_at TIMESTAMP
last_sign_int_at TIMESTAMP
current_sign_in_ip TEXT
last_sign_in_ip TEXT
create_time TIMESTAMP
update_time TIMESTAMP
name TEXT
avatar_file_name TEXT
avatar_content_type TEXT
avatar_file_size TEXT
avatar_updated_at TIMESTAMP
Indexes
PRIMARY
email_UNIQUE
interests
id INT
name TEXT
timeSlot TEXT
location TEXT
description TEXT
create_time TIMESTAMP
update_time TIMESTAMP
Indexes
user_user_mappings
id INT
primeUserID TEXT
timeSlot TEXT
sent TEXT
received TEXT
completematch TEXT
nomatch TEXT
create_time TIMESTAMP
update_time TIMESTAMP
Indexes
users_interesting_mappings
id INT
userID TEXT
interestID TEXT
description TEXT
create_time TIMESTAMP
update_time TIMESTAMP
Indexes
PRIMARY

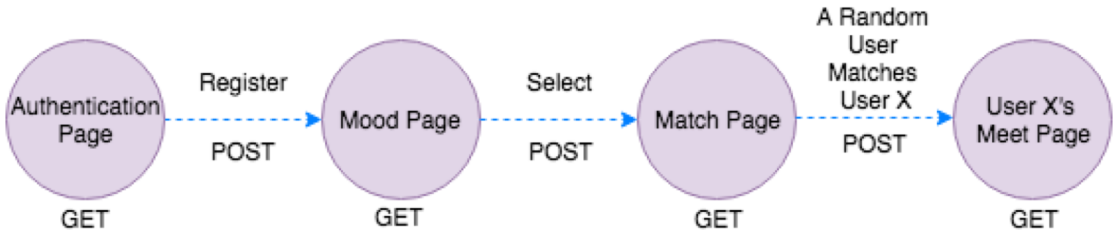
User Flow


Mood page
Gain access
Redirect
Mood
Redirect
Login page
User Mood
Match page
Update
response
Database
Like/Dislike
Render users
Meet page

High-Level Scalability Performance Strategy

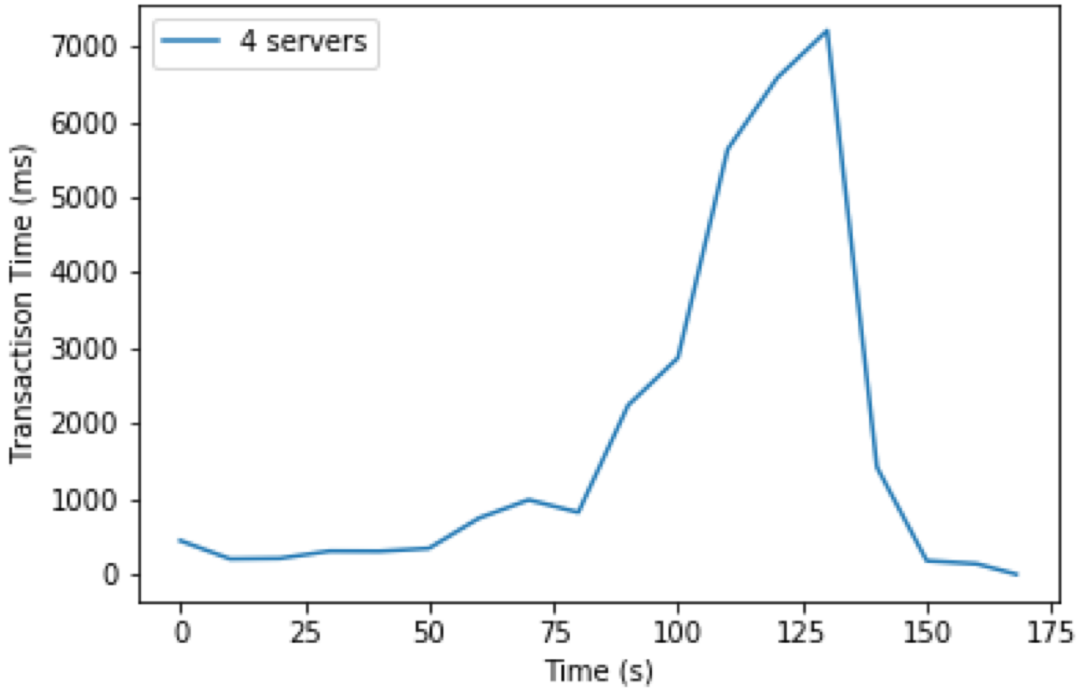


Load Test I


Authentication Page
GET
Register
POST
Mood Page
GET
Select
POST
Match Page
GET
A Random User Matches User X
POST
User X's Meet Page
GET

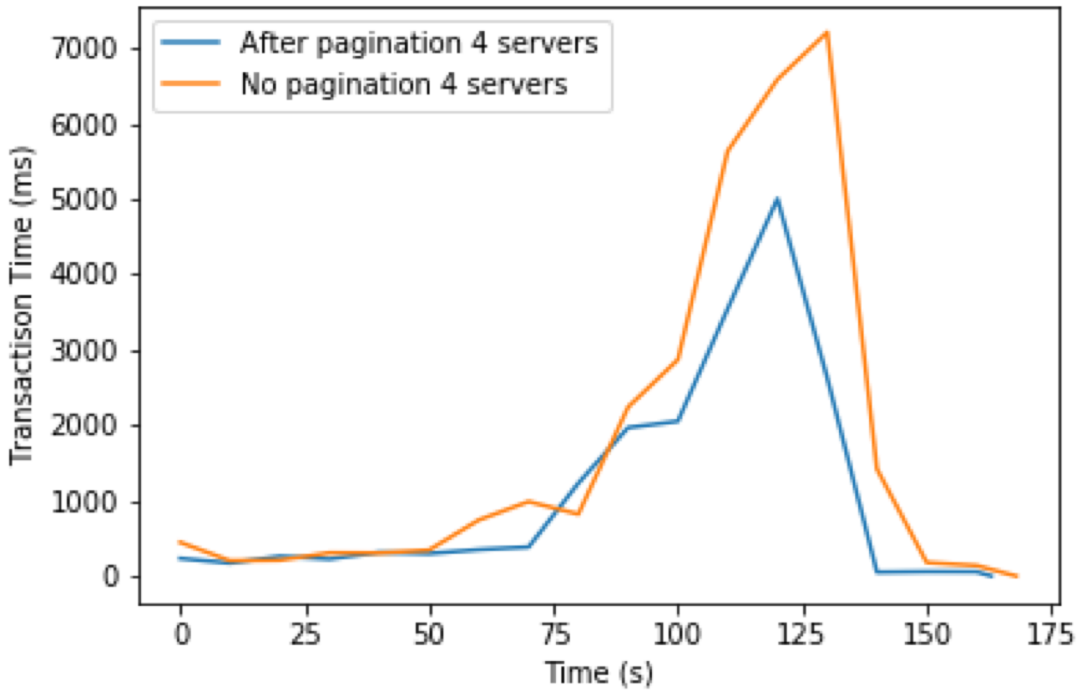
Before Optimization: 4 m3.medium servers,

Bottleneck after handing 6 user/sec



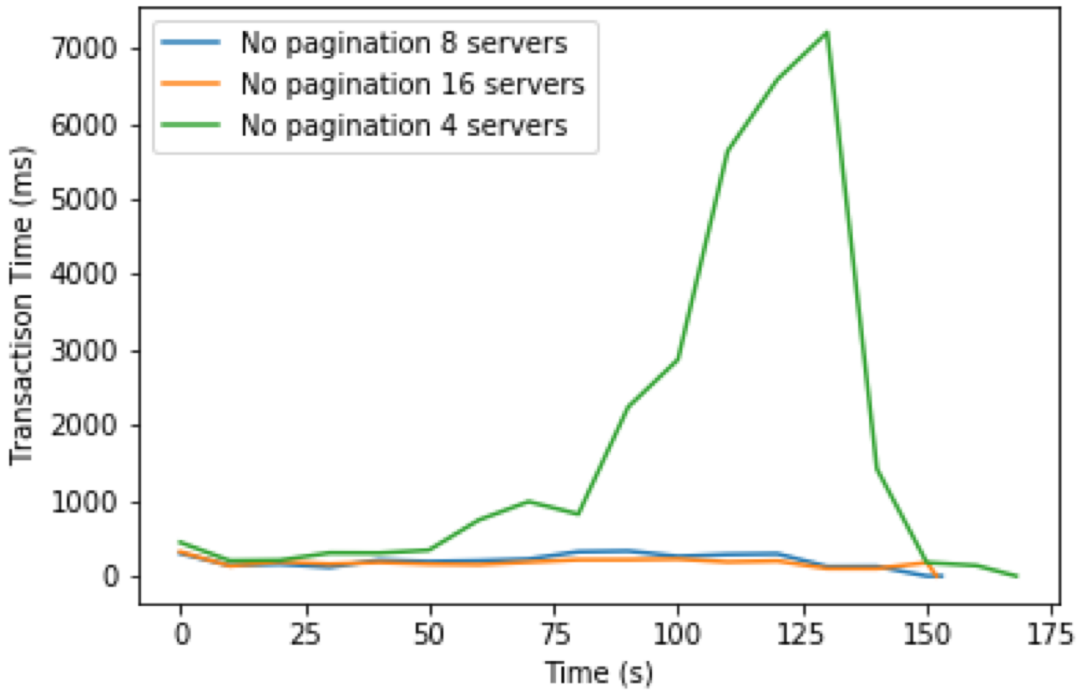
Pagination

Marginal improvement: 6 to 8 users/sec before the response time spikes

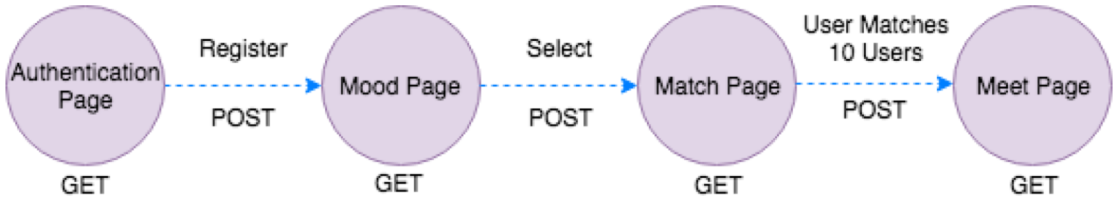


Horizontal Scaling: DB to 8 m3.medium

Bottleneck resolved



Load Test II


Authentication Page
GET
Register
POST
Mood Page
GET
Select
POST
Match Page
GET
User Matches 10 Users
POST
Meet Page
GET

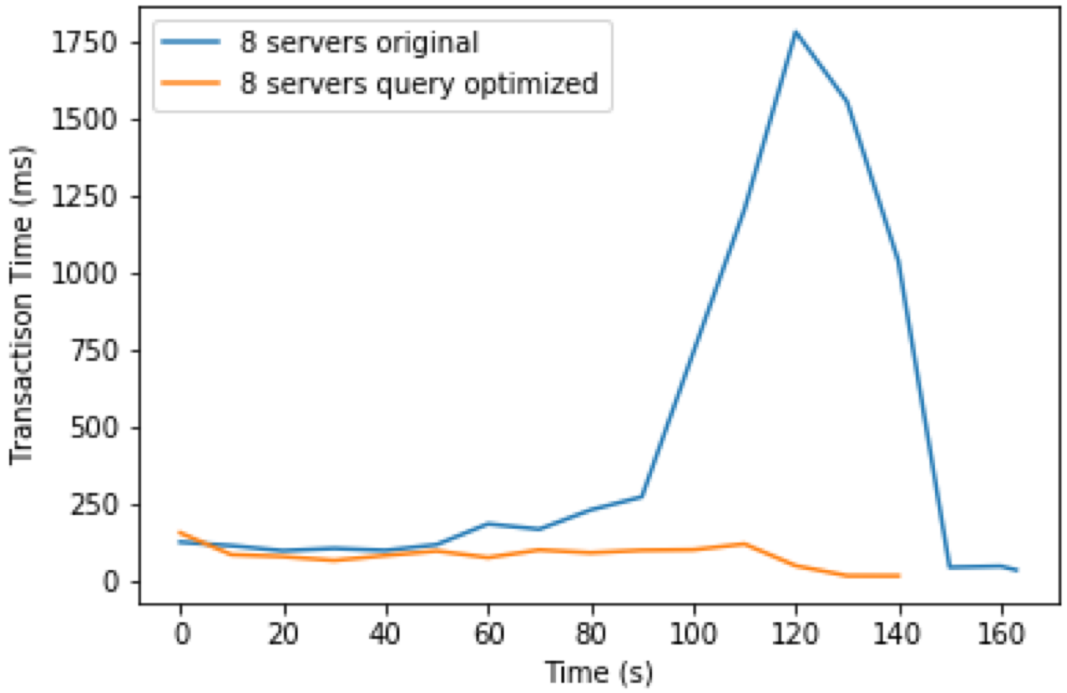
Diagnose

1.3s for a SingleSeconds SQL Query!

```
Started POST "/match/update" for 34.21
Processing by MatchController#updateRe
  Parameters: {"primuser"=>"433", "sec

  UserUserMapping Load (1.8ms)  SELECT
, ["LIMIT", 1]]
  UserUserMapping Load (15.3ms)  SELEC
], ["LIMIT", 1]]
  User Load (439.8ms)  SELECT  "users"

Completed 200 OK in 1277ms (Views: 0.6
  SQL (33.2ms)  UPDATE "user_user_mapp
rID", "433"]]
  SQL (34.9ms)  UPDATE "user_user_mapp

  UserUserMapping Load (0.9ms)  SELECT
, ["LIMIT", 1]]
  UserUserMapping Load (342.8ms)  SELE
"], ["LIMIT", 1]]
  UserInterestMapping Load (14.5ms)  S
t_mappings"."userID" != $2) ORDER BY RA

  UserInterestMapping Load (1335.5ms)
est_mappings"."userID" != $2) ORDER BY

  User Load (30.0ms)  SELECT  "users".

Completed 200 OK in 519ms (Views: 0.7m
  User Load (7.2ms)  SELECT  "users".*

Completed 200 OK in 1641ms (Views: 0.5
  UserInterestMapping Load (753.5ms)
st_mappings"."userID" != $2) ORDER BY R

  UserInterestMapping Load (397.1ms)
st_mappings"."userID" != $2) ORDER BY R
```

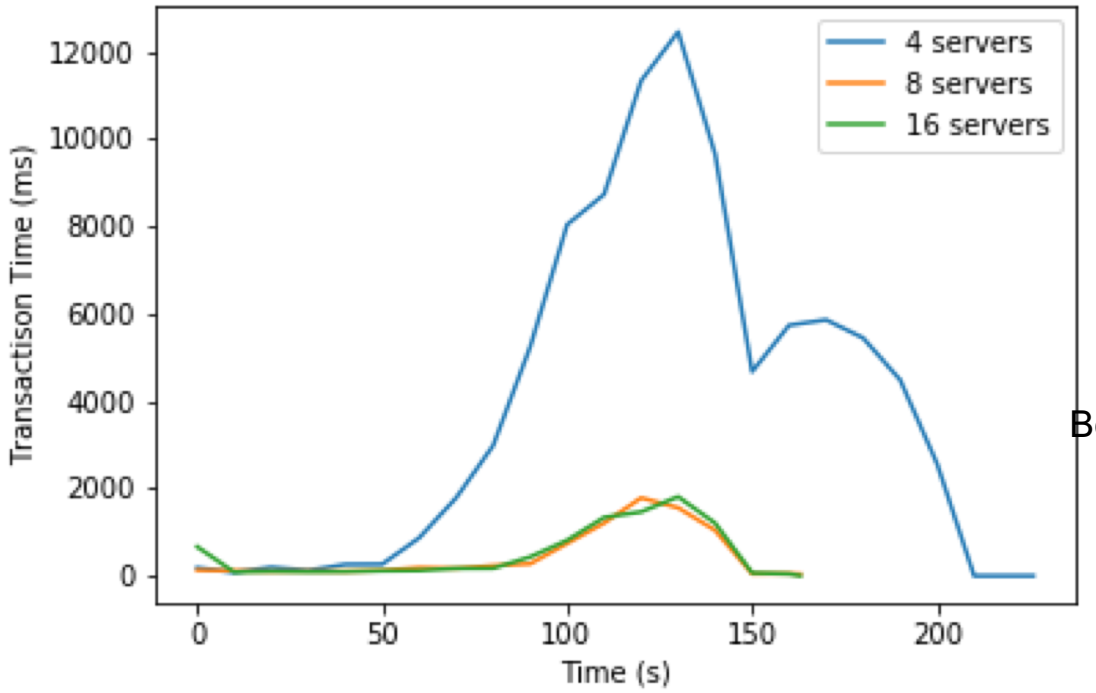
Query Optimization: avoid full-table scans

Bottleneck resolved for 8 m3.medium servers



Horizontal Scaling: DBto 8-16 m3.medium

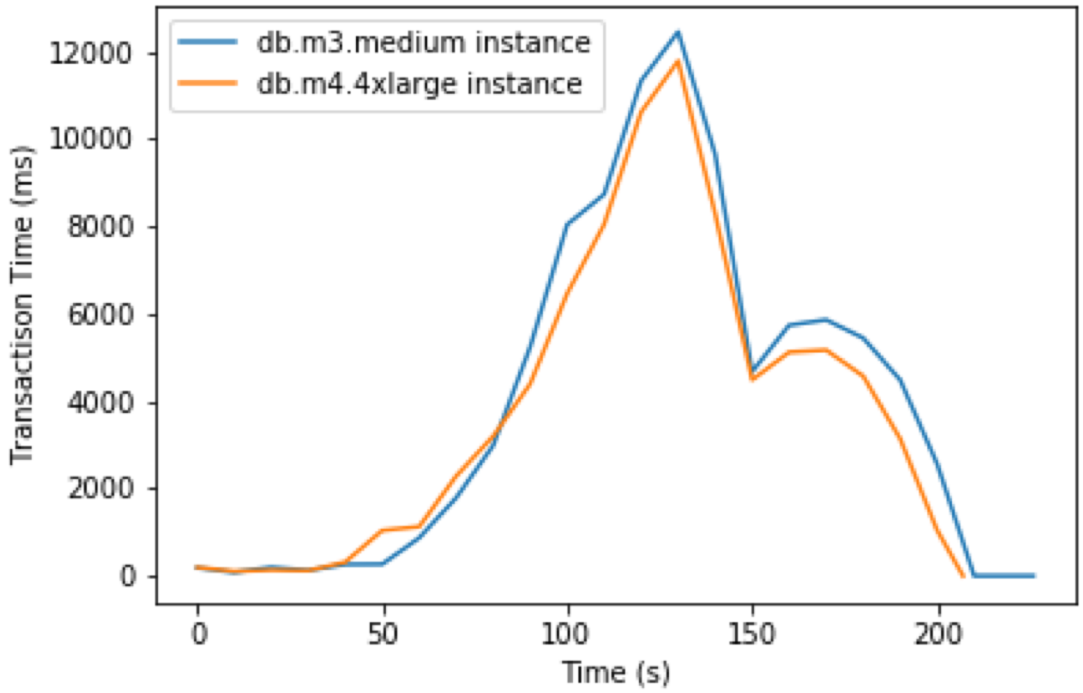
Bottleneck persists



Bottleneck persists

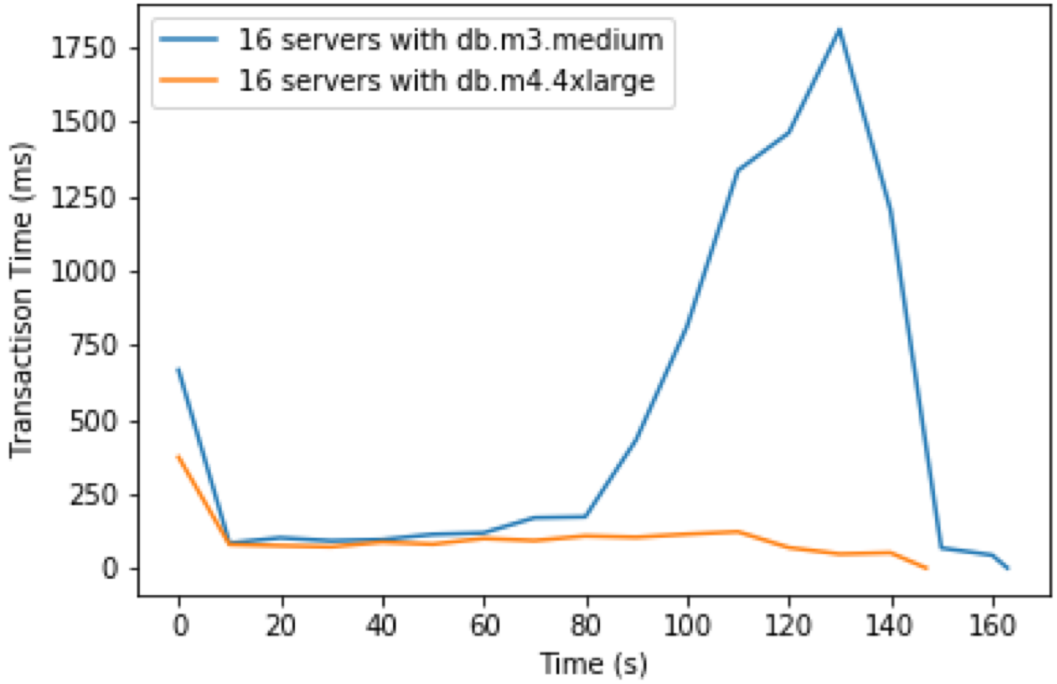
Vertical Scaling: db to 1 m4.4xlarge

Bottleneck persists



Mixed: 1 db.m4.4xlarge, 16 app.m3.medium

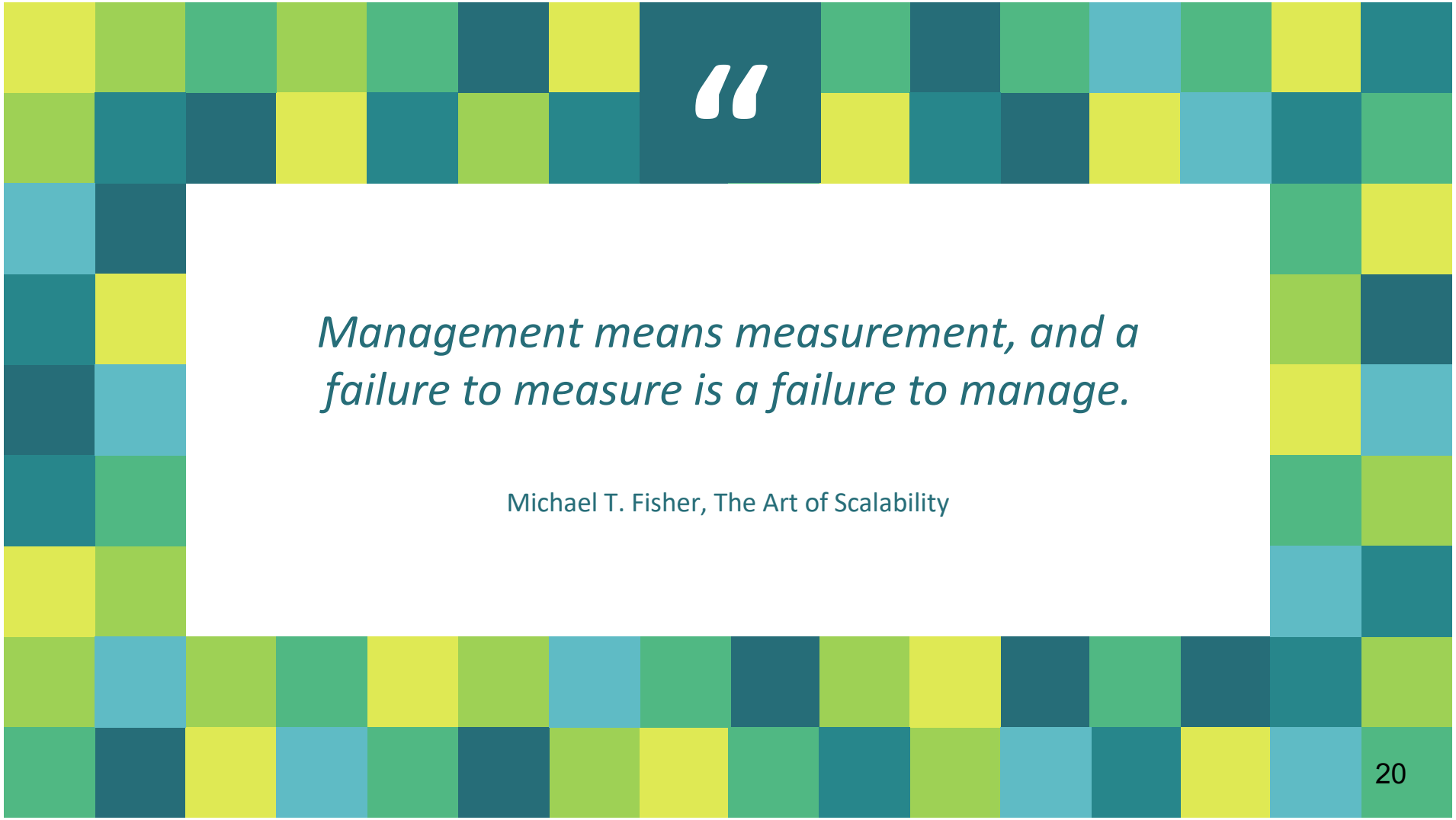
Bottleneck resolved (but expensive)



Conclusion

Requirement and Budget?

- Diagnose
- Code Optimization (Pagination, DB indexing, Caching, …)
- Load Balancing (Vertical and Horizontal Scaling


“
Management means measurement, and a failure to measure is a failure to manage.
Michael T. Fisher, The Art of Scalability

Thanks!

Ali Abbasinasab

@abbasinasab