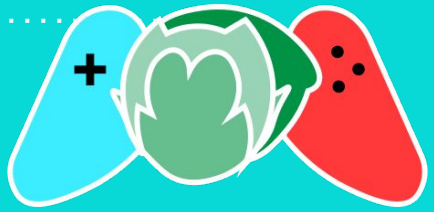


LinksIn

Amy Tu
Junhong Wang
Omar Tleimat
Cheuk Yin Phipson Lee

1. Overview
2. Load Testing
3. Future Development
4. Conclusion

1

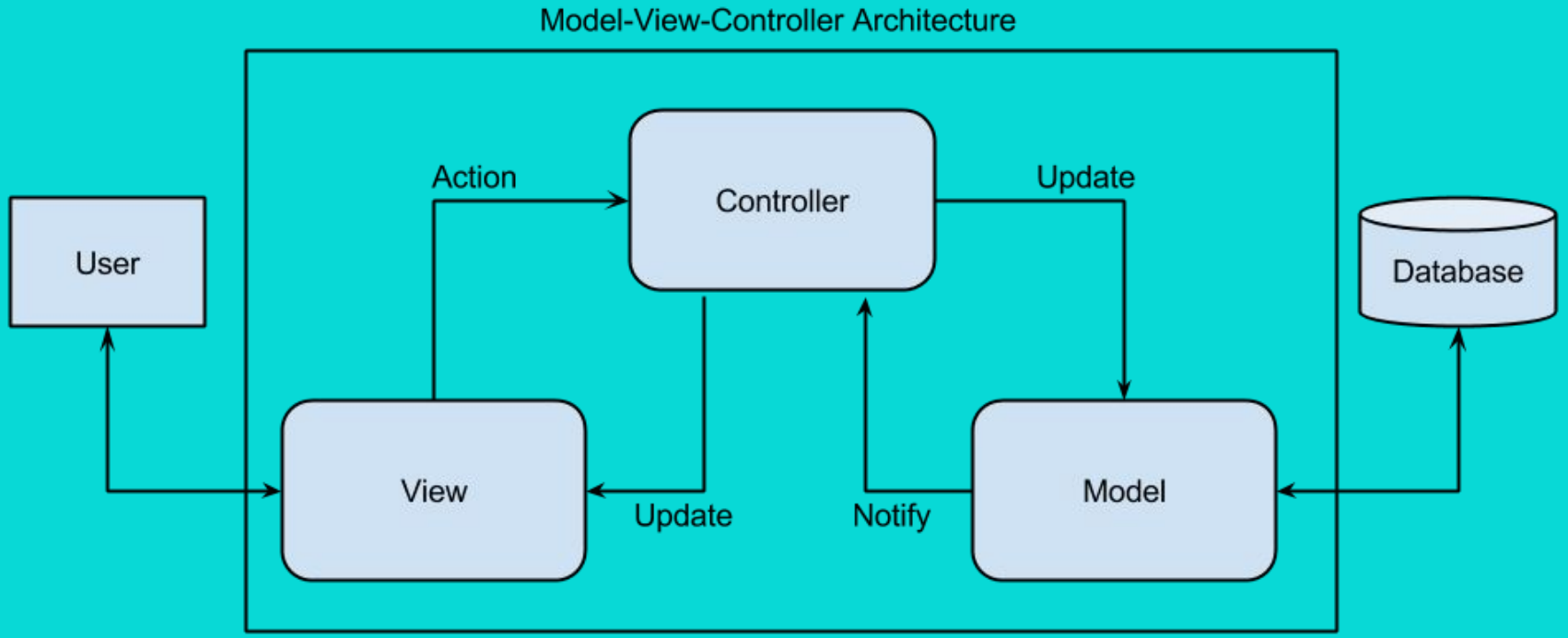
Project Overview

Demo project, application architecture

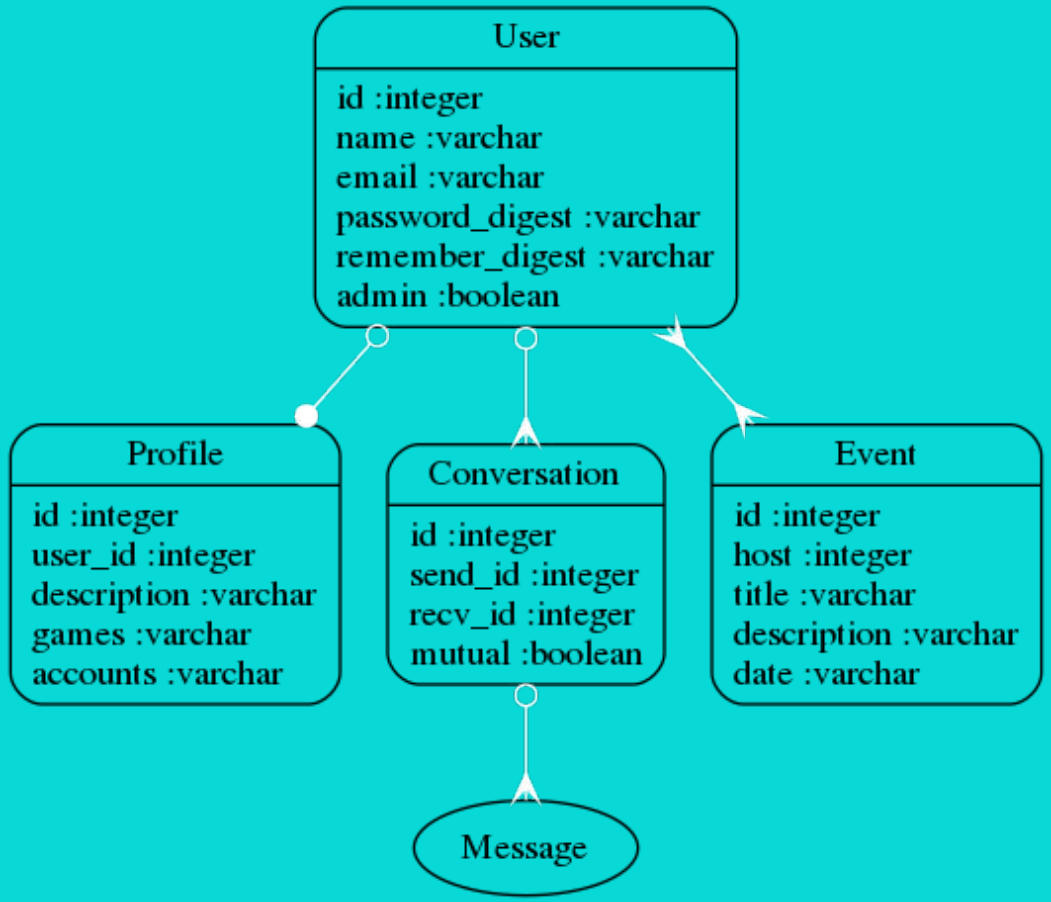

Demo

Application Architecture

Model-View-Controller (MVC)

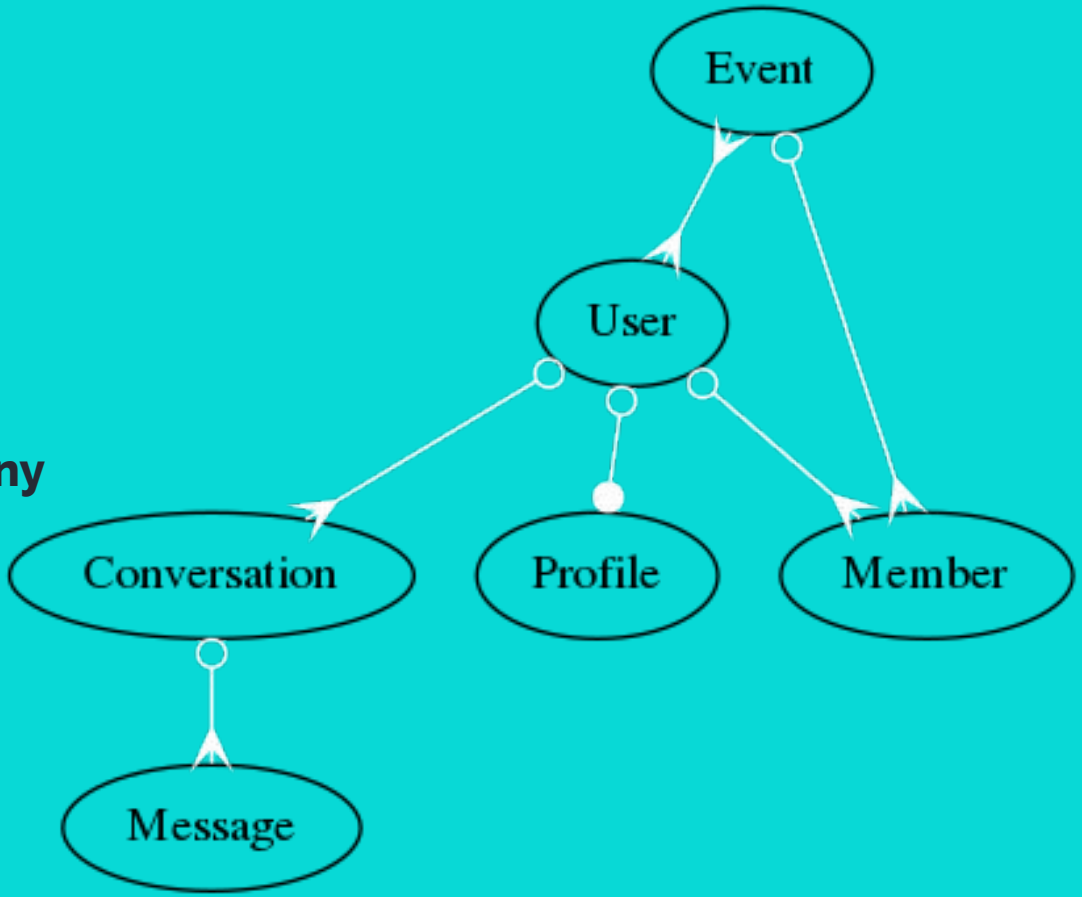


UML Diagram


User
id :integer
name :varchar
email :varchar
password_digest :varchar
remember_digest :varchar
admin :boolean
Profile
id :integer
user_id :integer
description :varchar
games :varchar
accounts :varchar
Conversation
id :integer
send_id :integer
recv_id :integer
mutual :boolean
Event
id :integer
host :integer
title :varchar
description :varchar
date :varchar
Message

Design Decisions

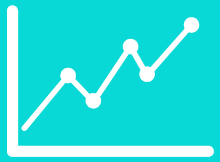
has_many, through =>
vs
has_and_belongs_to_many



2

Experiment & Results

Critical path, optimization before and afters, future optimizations



Test Phases

Test Phases

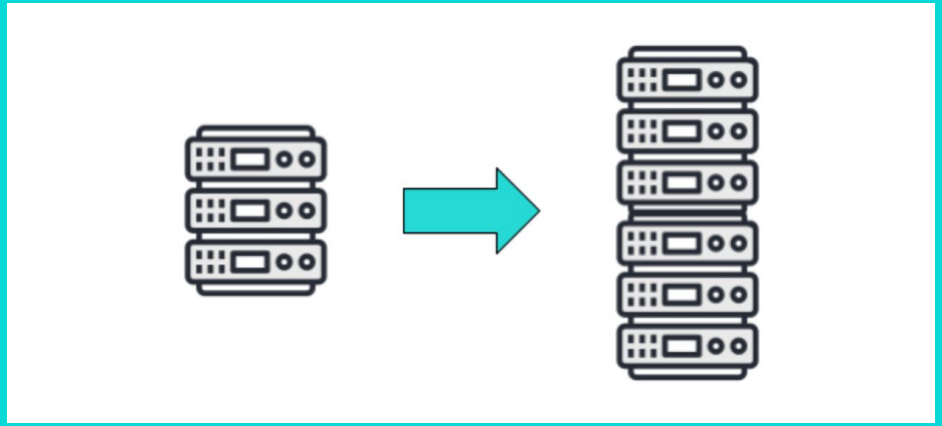
Phases	Users/sec
1	1
2	1.5
3	2
4	4
5	6
6	10
7	16
8	20
9	25
10	35
11	45
12	55

Critical Path

Critical Path

Step	Operation
1	Go to home page
2	Go to login page
3	Fill out login form
4	Log in and redirected to home page
5	Swipe right on user 1 and redirect to messages
6	Messages to user 1
7	Go to home page
8	Repeat step 5 - 7 for user 2, 3, 4, and 5
9	Go to events page
10	Go to event creation page
11	Create an event

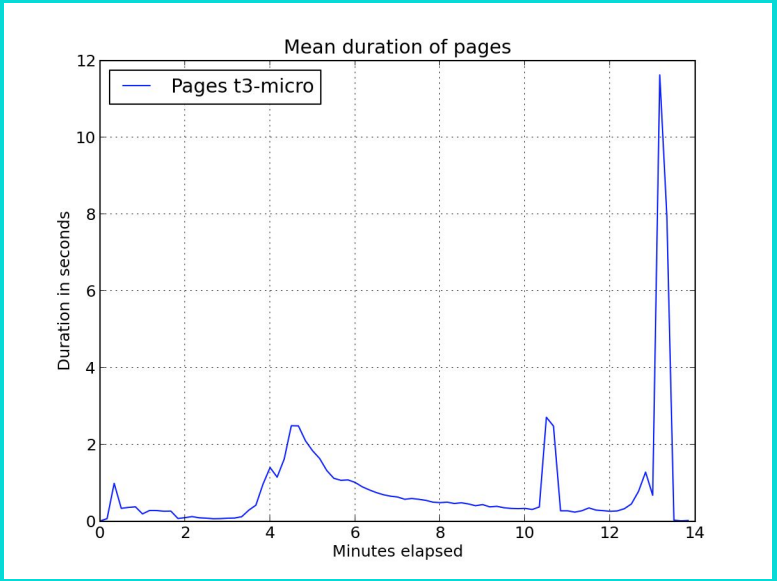
Vertical Scaling

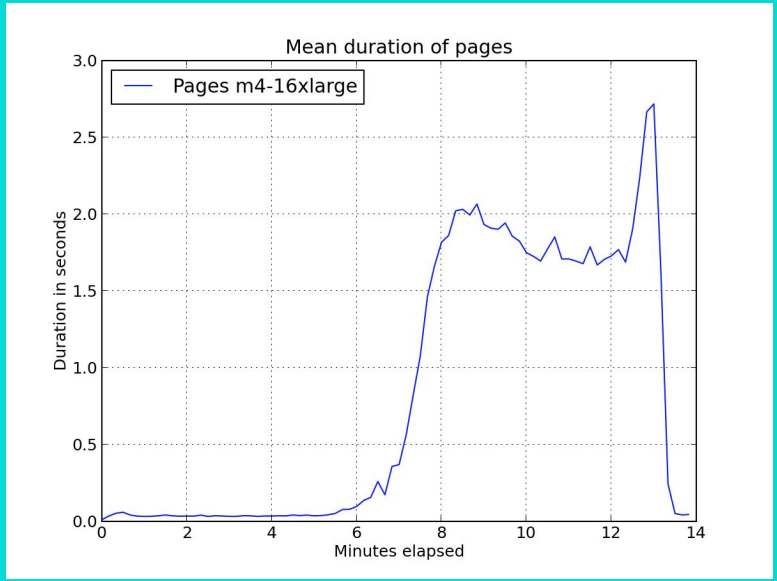


Vertical Scaling

Instance	vCPU	Memory (GiB)	Price
t3.micro	2	1	$7.4/month
t3.large	2	8	$60.0/month
t3.xlarge	4	16	$119.8/month
t3.2xlarge	8	32	$239.6/month
m4.4xlarge	16	64	$576/month
m4.16xlarge	64	256	$2304/month

Vertical Scaling





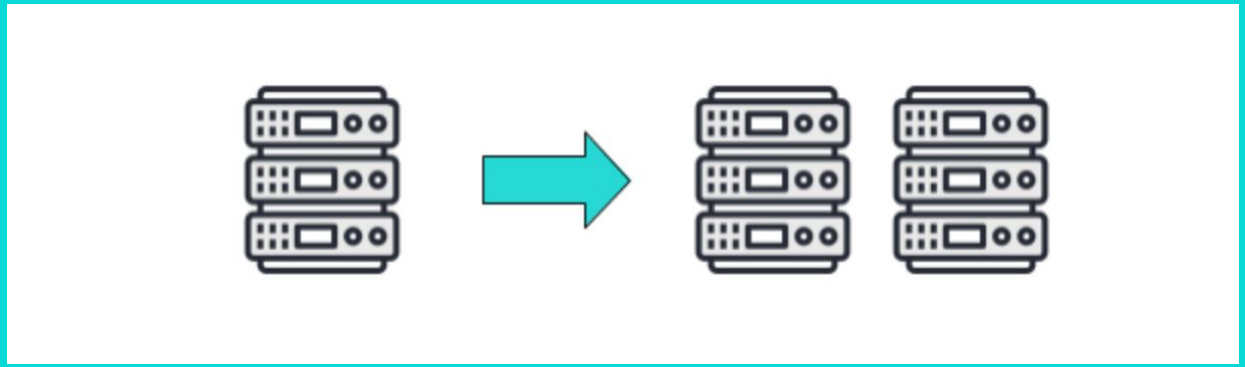
t3-micro

m4-16xlarge

Vertical Scaling

Instance	Price	Max Phase Handled
t3.micro	$7.4/month	3 (2 users/sec)
t3.large	$60.0/month	3 (2 users/sec)
t3.xlarge	$119.8/month	5 (6 users/sec)
t3.2xlarge	$239.6/month	5 (6 users/sec)
m4.4xlarge	$576/month	5 (6 users/sec)
m4.16xlarge	$2304/month	6 (10 users/sec)

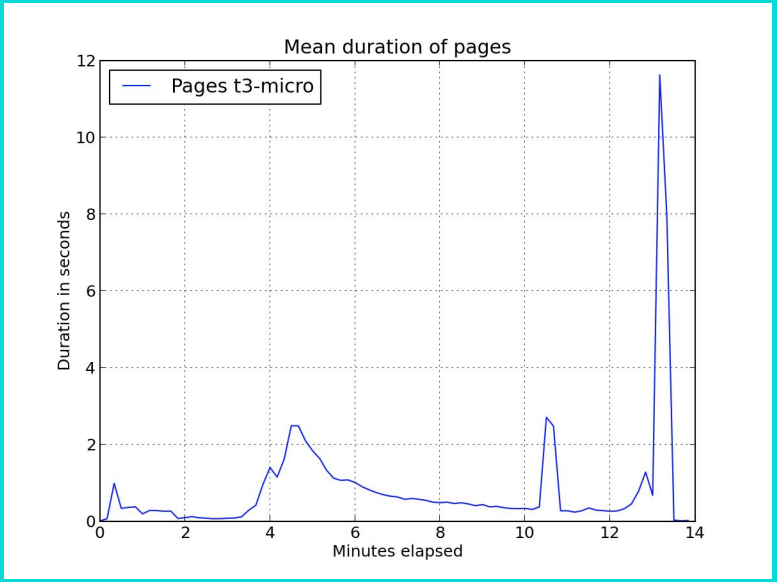
Horizontal Scaling

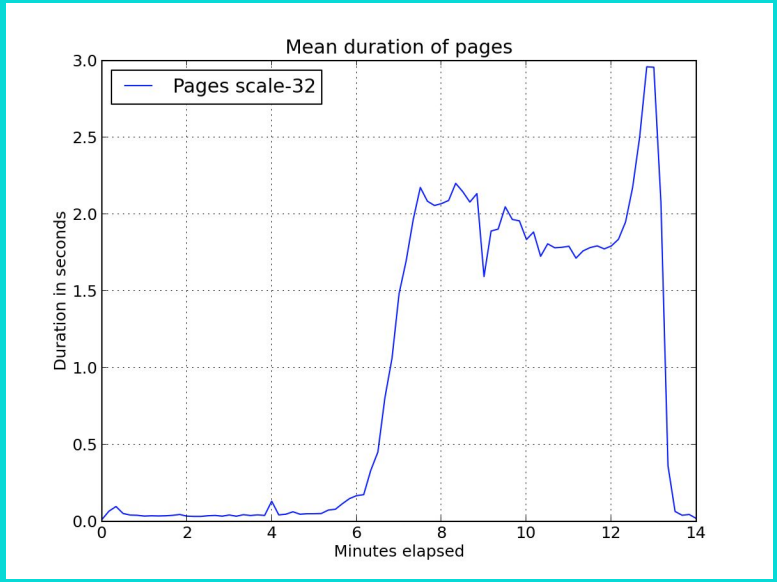


Horizontal Scaling

Instance	vCPU	Memory (GiB)	Price
t3.micro x 1	2 x 1	1 x 1	$7.49/month
t3.micro x 2	2 x 2	1 x 2	$14.98/month
t3.micro x 4	2 x 4	1 x 4	$30.00/month
t3.micro x 8	2 x 8	1 x 8	$59.90/month
t3.micro x 16	2 x 16	1 x 16	$119.81/month
t3.micro x 32	2 x 32	1 x 32	$239.62/month

Horizontal Scaling





t3-micro x 1

t3-micro x 32

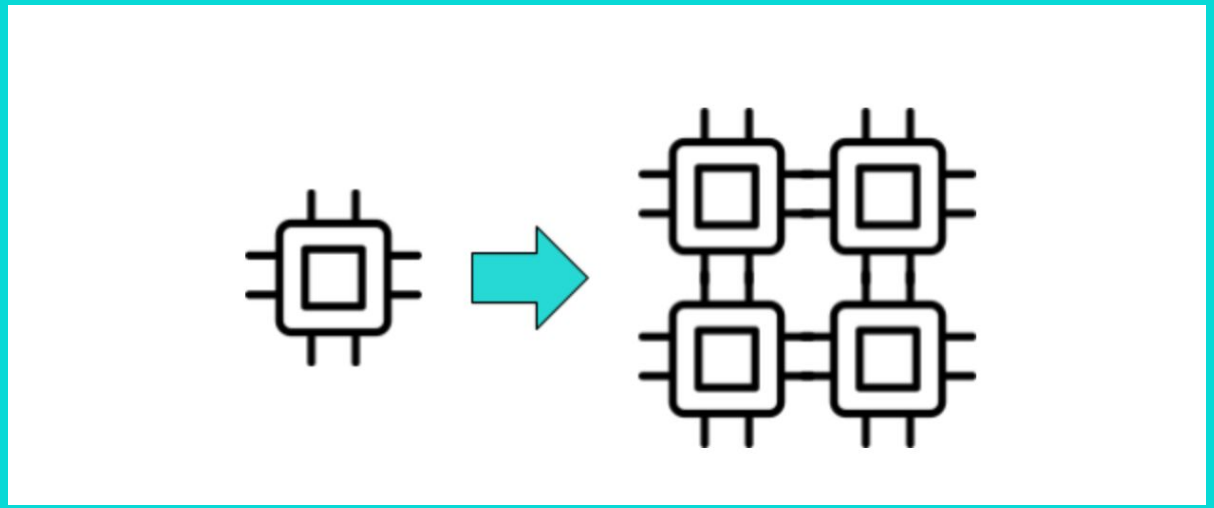
Horizontal Scaling

Instance	Price	Max Phase Handled
t3.micro x 1	$7.49/month	3 (2 users/sec)
t3.micro x 2	$14.98/month	5 (6 users/sec)
t3.micro x 4	$30.00/month	5 (6 users/sec)
t3.micro x 8	$59.90/month	5 (6 users/sec)
t3.micro x 16	$119.81/month	5 (6 users/sec)
t3.micro x 32	$239.62/month	6 (10 users/sec)

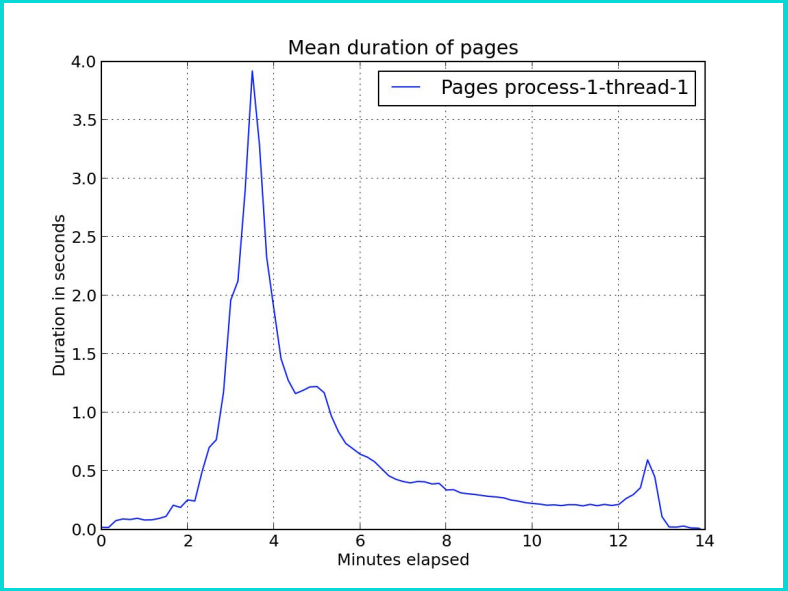
Horizontal Scaling

Save $2064.38/month

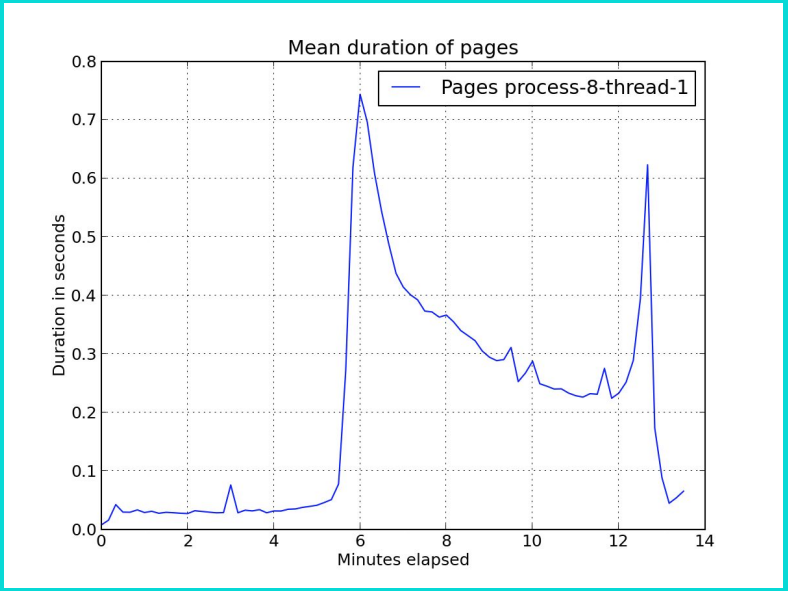
Processes/Threads



Processes/Threads

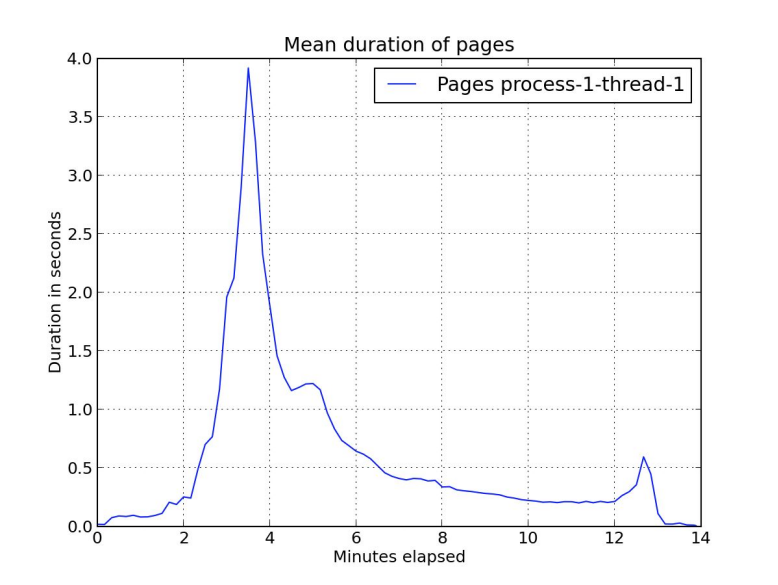


process x 1
thread x 1

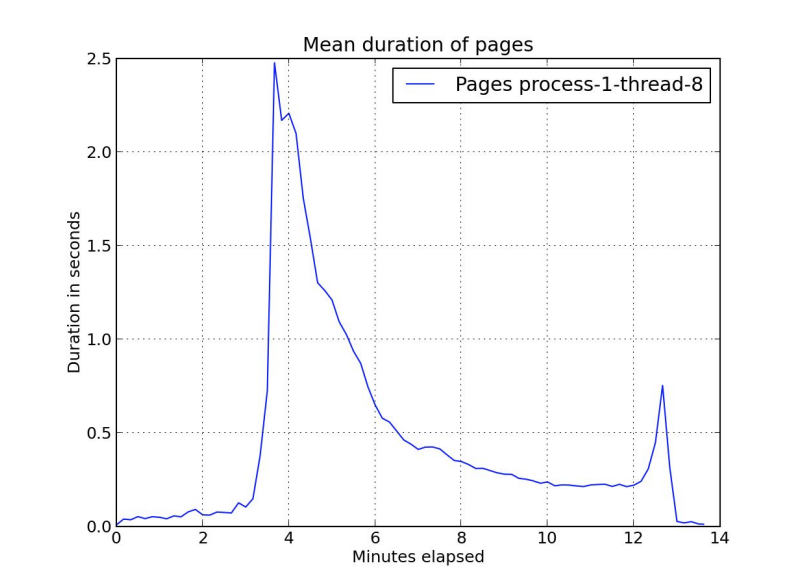


process x 8
thread x 1

Processes/Threads



process x 1
thread x 1

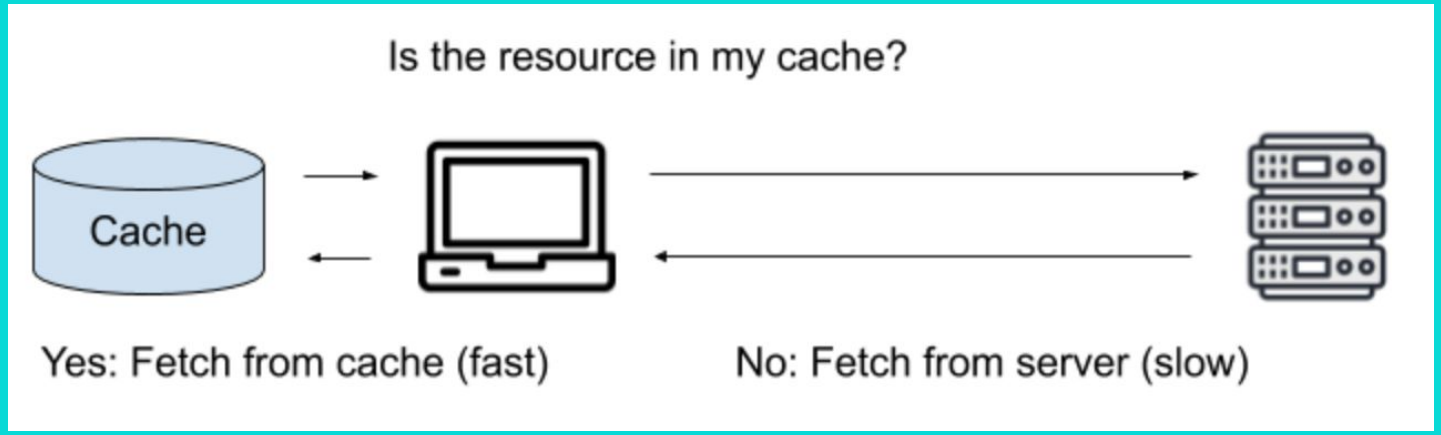


process x 1
thread x 8

Processes/Threads

Instance	Max Phase Handled
process x 1 & thread x 1	1 (1 user/sec)
process x 8 & thread x 1	5 (6 users/sec)
process x 1 & thread x 8	3 (2 users/sec)

Client-side Caching



Client-side caching

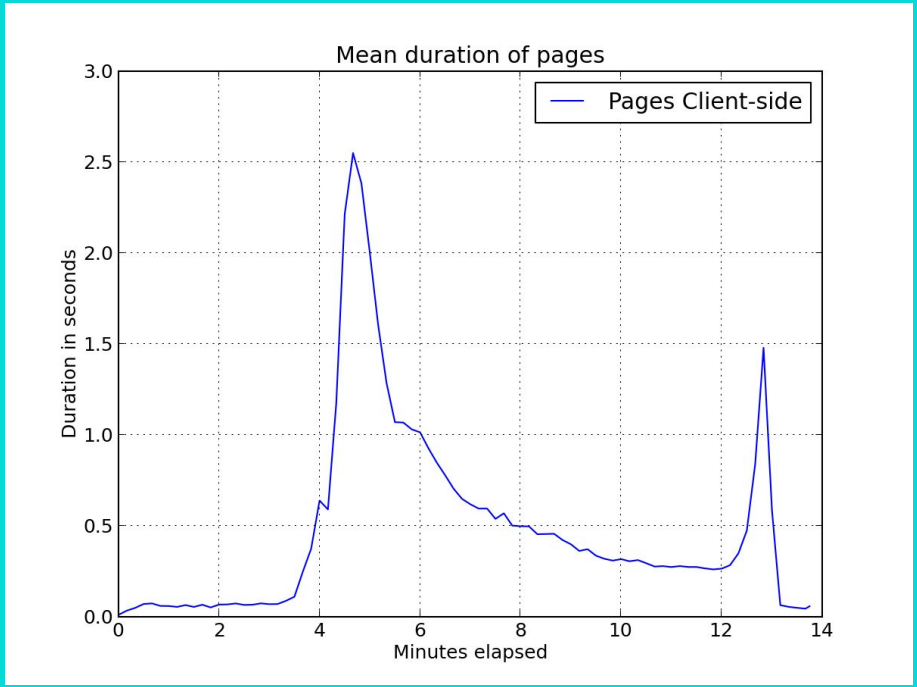
user_controller.rb

```
if stale?([@user, @user.updated_at, @user.events])
     @user = User.find(params[:id])
end
```

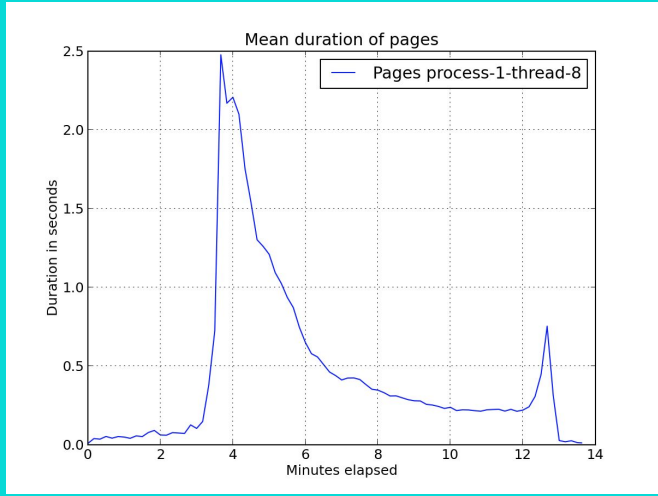
event_controller.rb

```
if stale?([@event, @event.updated_at])
     @event = Event.find(params[:id])
end
```

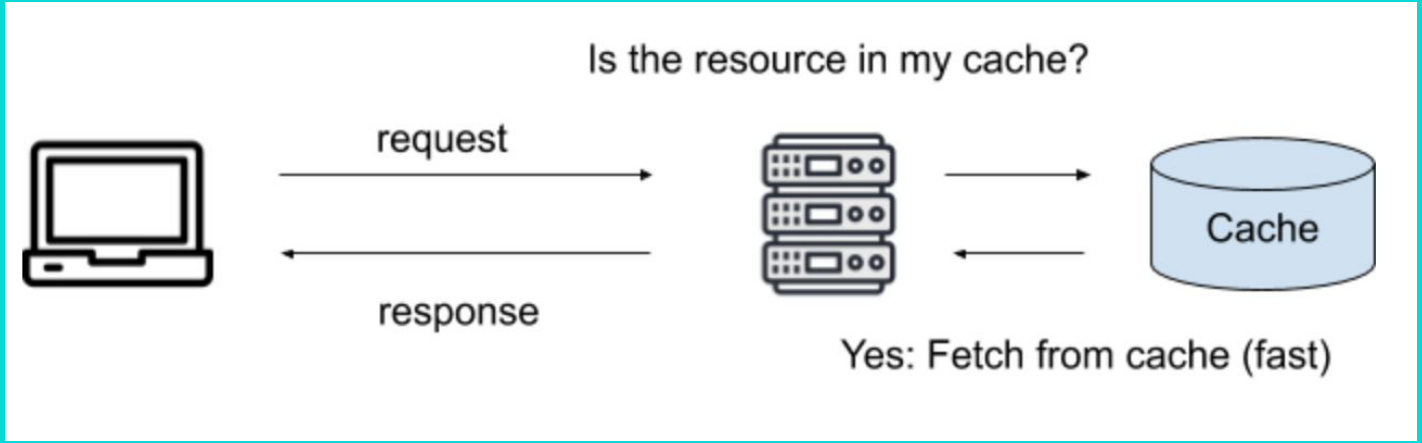
Client-side caching



Still fails at Phase 3



Server-side Caching



Server-side caching

```
<% cache(Event.cache_key_for_eventList) do %>
  ...
  <% @user.events.each do |event| %>
    <% cache(Event.cache_key_for_event(event))
do %>
      ...
    <% end %>
  <% end %>
<% end %>
```

_eventcart.html.erb

```
def Event.cache_key_for_event(e)

"event-#{e.id}-#{e.updated_at}-#{e.member
s.count}"
end

def Event.cache_key_for_eventList

"eventList-#{Event.maximum(:updated_at)}-
#{Member.maximum(:updated_at)}"
end
```

event.rb

Server-side caching

```
<div class="LI_GamerCardContainer">
...
   <% @userlist.each do |user| %>
   <% cache(User.cache_key_for_user(x)) do %>
      ...
      <% end %>
   <% end %>
 </div>
```

```
def User.cache_key_for_user(x)
   "user-#{x.id}-#{x.updated_at}-#{x.events.count}"
end
```

Users

Server-side caching

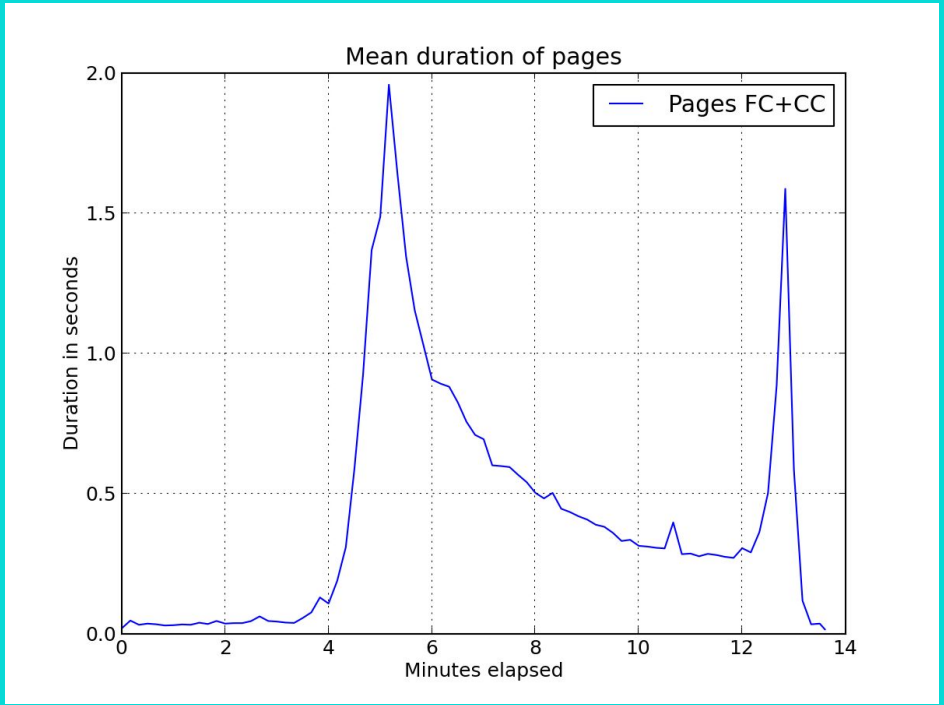
```
<div class="columns" >
 <div class="column is-3" style="overflow: auto;">
   <% @conversations.each do |conversation| %>
     <% cache(Conversation.cache_key_for_conversation(conversation)) do %>
        ...
       <% end %>
   <% end %>
     ...
   <div style="height: 90%; overflow-y: scroll" id="message-window">
     <% @messages.reverse.each do |message| %>
       <% cache(Message.cache_key_for_message(message)) do %>
       ...
       <% end %>
     <% end %>
   </div>
        ...
</div>
```

Conversations

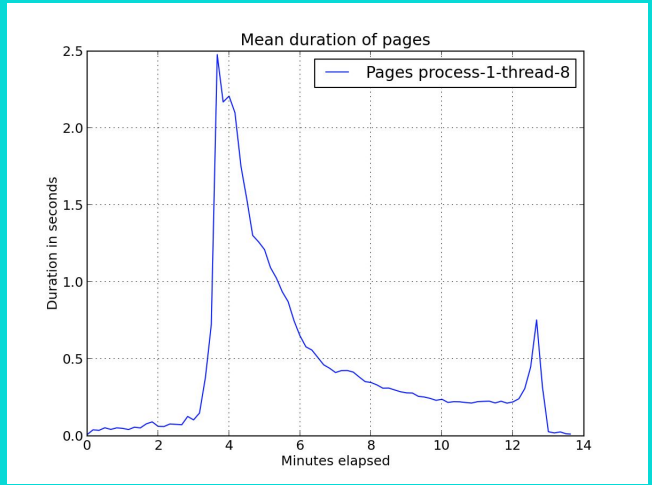
```
def
Conversation.cache_key_
for_conversation(x)

"conversation-#{x.send_
id}-#{x.recv_id}-#{x.up
dated_at}"
end
```

```
def
Message.cache_key_for_m
essage(x)

"message-#{x.user_id}-#
{x.conversation_id}-#{x
.created_at}"
end
```

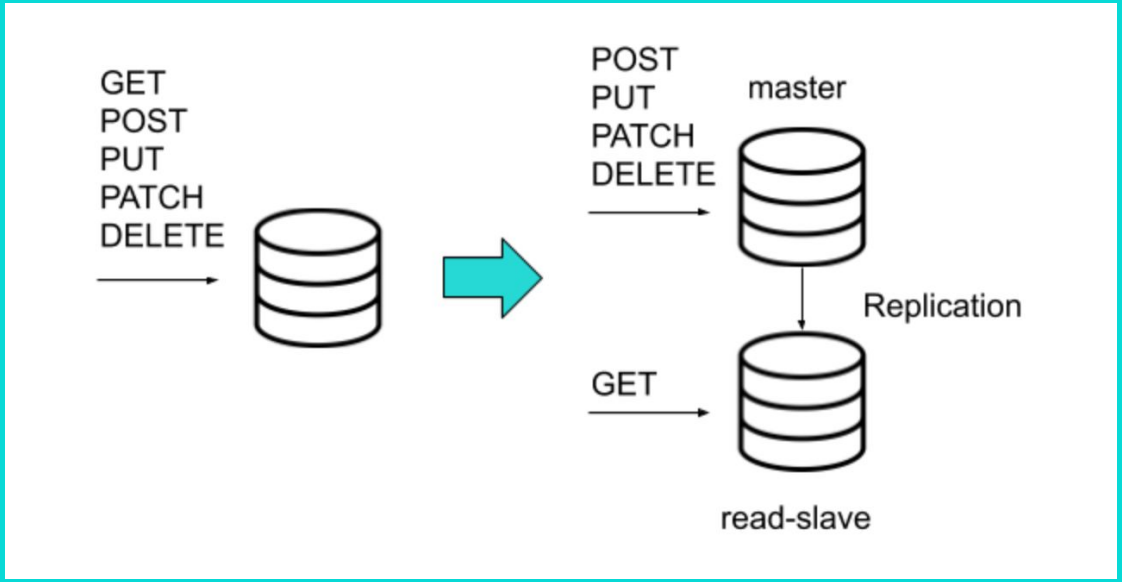
Server-side caching



Fails at Phase 4 (originally Phase 3)



Read-slave


GET
POST
PUT
PATCH
DELETE
POST
PUT
PATCH
DELETE
master
Replication
GET
read-slave

Read-slave

```
# config/database.yml
...

production:
  primary:
    adapter: postgresql
    database: <%= ENV['RDS_DB_NAME'] %>
    encoding: UTF8
    host: <%= ENV['RDS_HOSTNAME'] %>
    password: <%= ENV['RDS_PASSWORD'] %>
    port: <%= ENV['RDS_PORT'] %>
    username: <%= ENV['RDS_USERNAME'] %>
  primary_replica:
    adapter: postgresql
    database: <%= ENV['RDS_DB_NAME'] %>
    encoding: UTF8
    host: ""
    password: <%= ENV['RDS_PASSWORD'] %>
    port: <%= ENV['RDS_PORT'] %>
    username: <%= ENV['RDS_USERNAME'] %>
    replica: true
```

Read-slave

```ruby
# app/models/application_record.rb
class ApplicationRecord < ActiveRecord::Base
  self.abstract_class = true
  connects_to database: { writing: :primary, reading: :primary_replica }
end
```

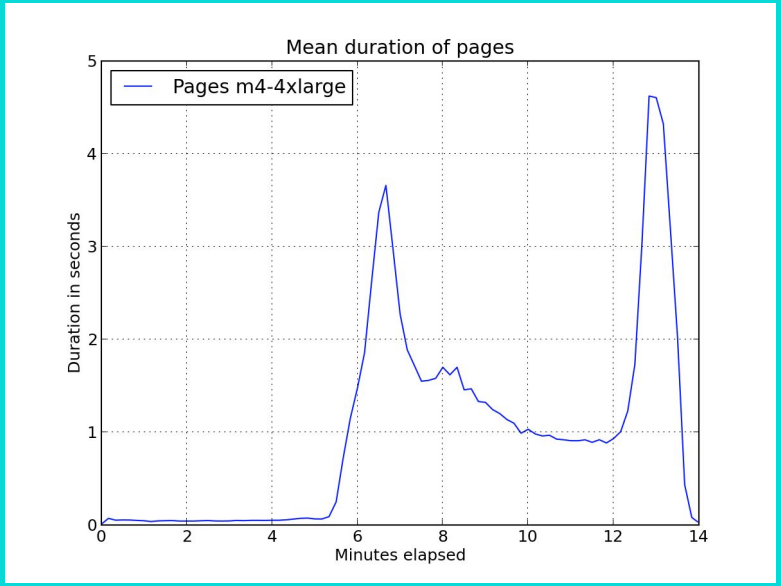
Read-slave

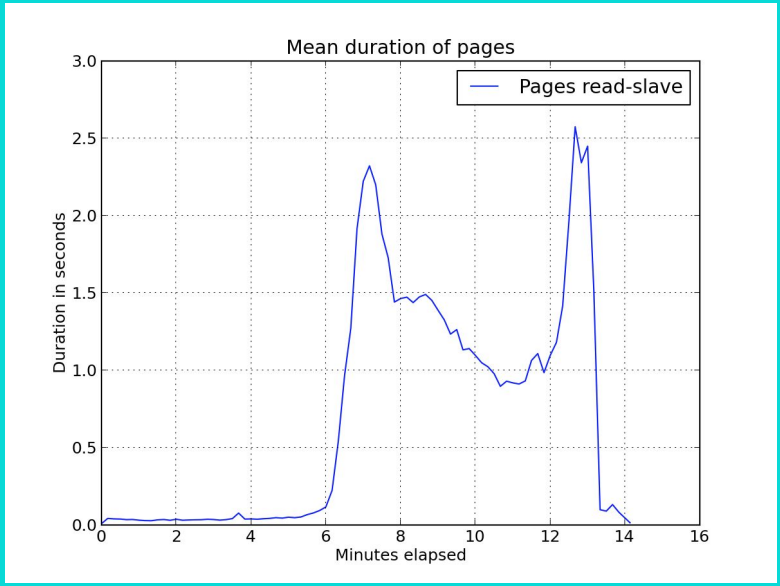
```ruby
# config/application.rb
module Linksin
  class Application < Rails::Application

  ...

  config.active_record.database_selector =
    { delay: 2.seconds }

  config.active_record.database_resolver =
    ActiveRecord::Middleware::DatabaseSelector::Resolver

  config.active_record.database_resolver_context =
    ActiveRecord::Middleware::DatabaseSelector::Resolver::Session

  end
end
```

Read-slave

```
# config/database.yml
...

production:
  ...
  primary_replica:
    ...
    host: "linksin.civkgqjsrtyg.us-west-2.rds.amazonaws.com"
    ...
```

Read-slave





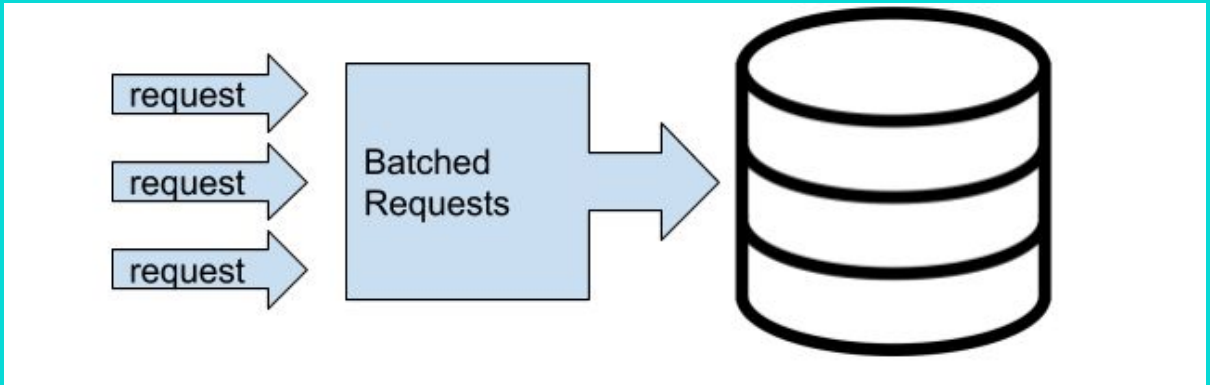
w/o read-slave

w/ read-slave

Read-slave

Instance	Max Phase Handled
w/o read-slave	5 (6 users/sec)
w/ read-slave	6 (10 users/sec)

SQL Optimization



SQL Optimization

- Used eager loading to circumvent N+1 query problem

```
Rendering conversations/matches.html.erb within layouts/application
Conversation Load (0.2ms)  SELECT "conversations".* FROM "conversations" WHERE ((conversations.send_id = 1 OR conversations.recv_id = 1))
↳ app/views/conversations/matches.html.erb:3
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 2], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 2], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 4], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 4], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 11], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 11], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 6], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.2ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 6], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 8], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 8], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 12], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 12], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 15], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 15], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
CACHE User Load (0.0ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:5
User Load (0.1ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 5], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:6
Profile Load (0.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" = ? LIMIT ?  [["user_id", 5], ["LIMIT", 1]]
↳ app/views/conversations/matches.html.erb:20
```

Eager Loading

```
Processing by ConversationsController#matches as HTML
  User Load (0.6ms)  SELECT "users".* FROM "users" WHERE "users"."id" = ? LIMIT ?  [["id", 1], ["LIMIT", 1]]
  ↳ app/helpers/sessions_helper.rb:18:in `current_user'
  Rendering conversations/matches.html.erb within layouts/application
  Conversation Load (0.7ms)  SELECT "conversations".* FROM "conversations" WHERE ((conversations.send_id = 1 OR conversations.recv_id = 1))
  ↳ app/views/conversations/matches.html.erb:3
  User Load (0.2ms)  SELECT "users".* FROM "users" WHERE "users"."id" IN (?, ?)  [["id", 2], ["id", 1]]
  ↳ app/views/conversations/matches.html.erb:3
  Profile Load (1.1ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" IN (?, ?)  [["user_id", 1], ["user_id", 2]]
  ↳ app/views/conversations/matches.html.erb:3
  User Load (0.2ms)  SELECT "users".* FROM "users" WHERE "users"."id" IN (?, ?, ?, ?, ?, ?, ?, ?)  [["id", 1], ["id", 4], ["id", 11], ["id", 6], ["id", 8], ["id", 12], ["id", 15], ["id", 5]]
  ↳ app/views/conversations/matches.html.erb:3
  Profile Load (0.2ms)  SELECT "profiles".* FROM "profiles" WHERE "profiles"."user_id" IN (?, ?, ?, ?, ?, ?, ?, ?)  [["user_id", 1], ["user_id", 4], ["user_id", 5], ["user_id", 6], ["user_id", 8], ["user_id", 11], ["user_id", 12], ["user_id", 15]]
  ↳ app/views/conversations/matches.html.erb:3
```

SQL Optimization

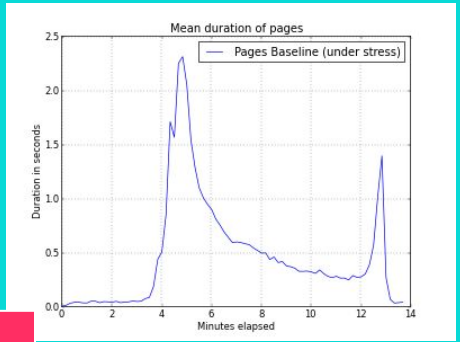
- Attempted (keyword: attempted) to speed up user matching with a custom query

```
def show
  @user = User.find(params[:id])
  @userlist = randomShow(@user).select do
|user|
    conversation =
Conversation.between(user.id, @user.id)
    conversation.empty?
  end
end
```

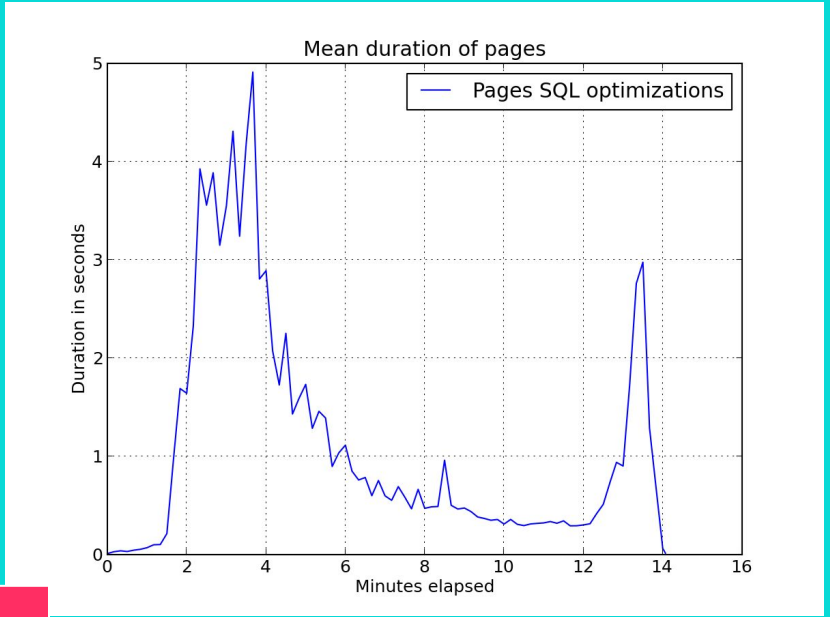
```
def randomShow(user)
  User.find_by_sql("SELECT * FROM users u
                    WHERE ((u.id != #{user.id}) AND
                    NOT EXISTS
                    ( SELECT u.id
                      FROM conversations c
                      WHERE (c.send_id = u.id) OR
                      (c.recv_id = u.id)
                    ))
                    ORDER BY RANDOM()
                    LIMIT 20;")
end
```

SQL Optimization

- Performance actually decreased!
  - NOT EXISTS runs the sub-query for each id → scales horribly
  - NOT IN should have been used instead as the sub-queries are only run once



Before



After

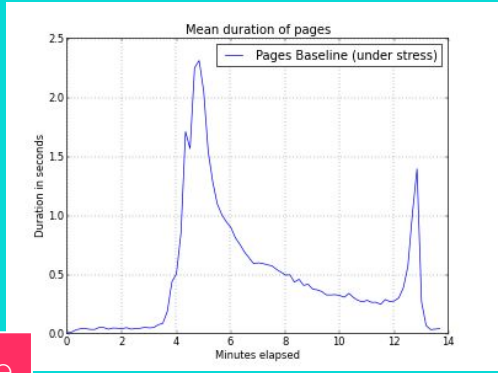
SQL Optimization

Instance	Max Phase Handled
w/o sql changes	3 (2 users/sec)
w/ sql changes	1 (1 users/sec)

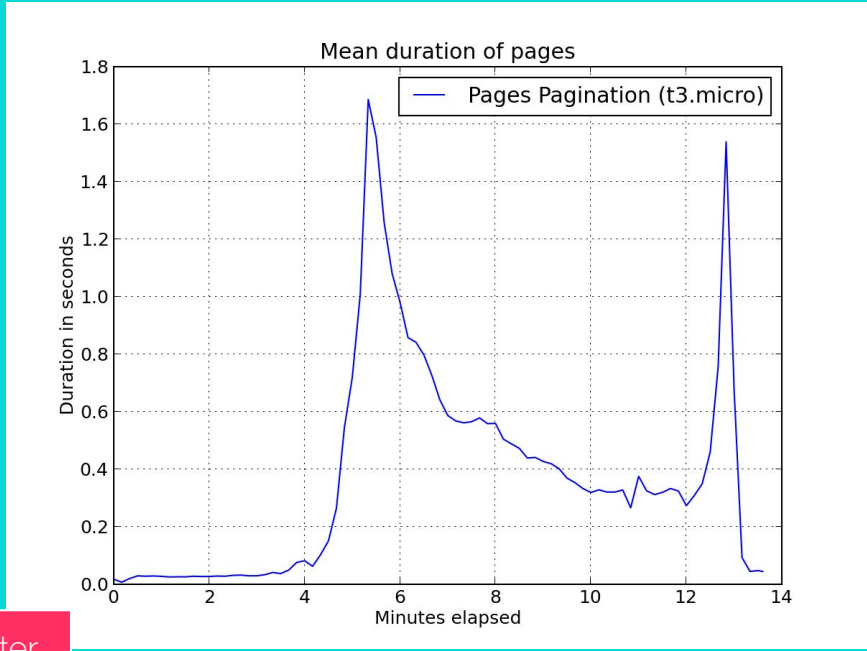
Pagination

Pagination

- Switched from will_paginate to pagy for countless pagination
- Used pagination on user for matching → load more users via ajax when down to two cards



Before



After

Pagination

Instance	Max Phase Handled
w/o pagination	3 (2 users/sec)
w/ pagination	4 (4 users/sec)

3

Future Development

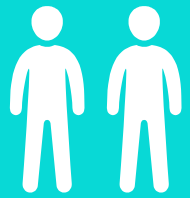
Database sharding, more features

- ❑ Database Sharding
- ❑ Improve Events
- ❑ Real-time messaging
- ❑ Notifications
- ❑ Linking external accounts
- ❑ Combine optimizations

4

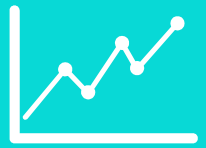
Conclusions & Lessons Learned

Agile Development

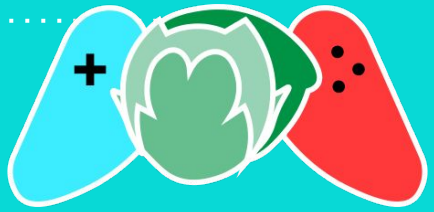


Pair Programming

Scalable Web Service



Load Testing



Thanks!