CS291A: Scalable Internet Services

gTrack: Track Prices of Games on Steam

Nazmus Saquib Udit Paul Alex Ermakov

Graduate Students
Department of Computer Science
University of California Santa Barbara

December 8, 2018

Outline

Outline

Introduction

1. gTrack is a website built using Ruby on Rails.
2. gTrack is designed for users to get information related to the games available on Steam.
3. Logged in users can comment and express their like or dislike about any game.
4. gTrack users are presented with a highly specialized search feature.

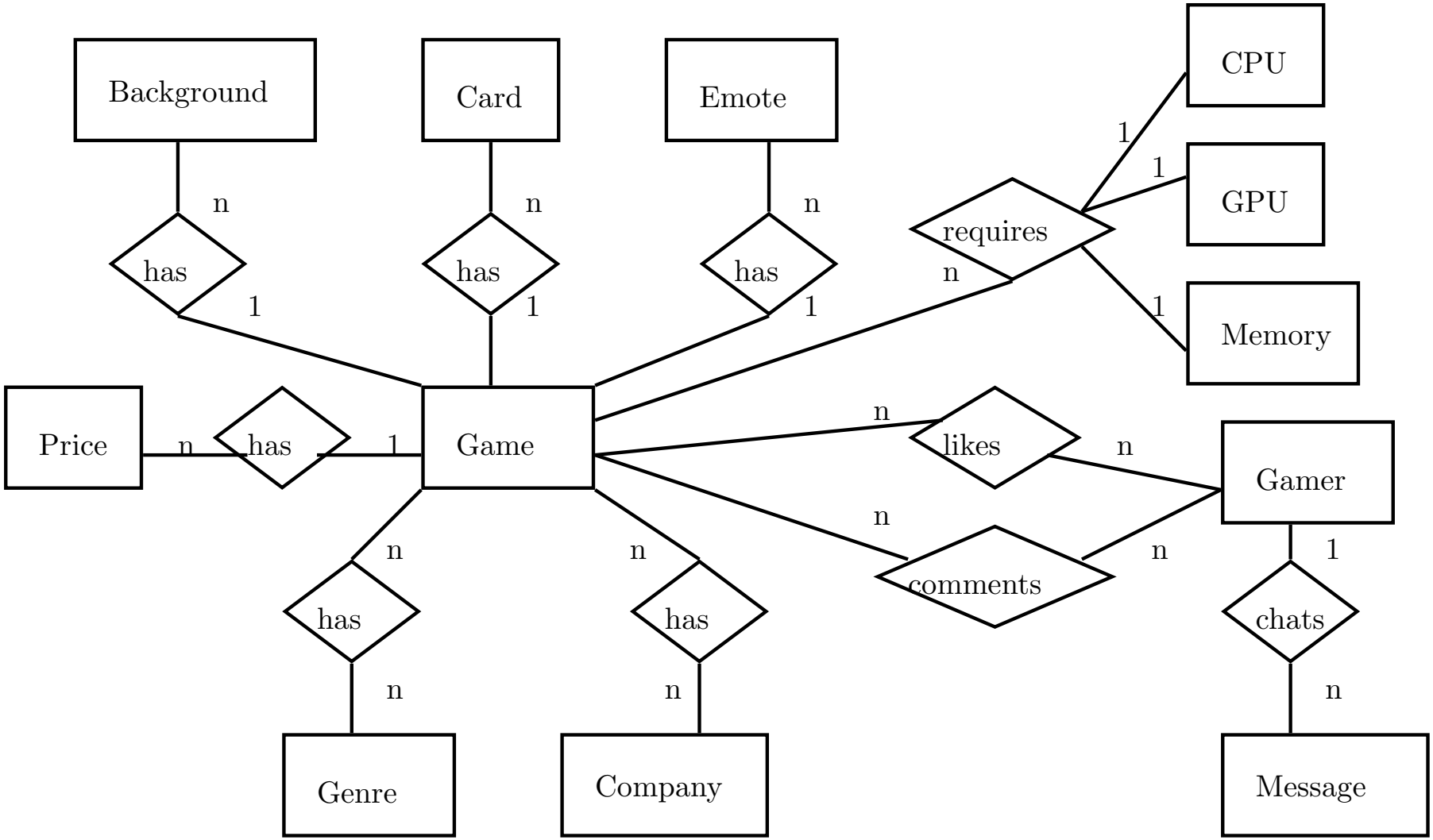
Outline

Motivation

1. Steam is the largest PC game distribution platform, yet its search functionalities are inadequate in meeting specialized queries.
2. Items such as emotes, cards and background to a game are not presented in an organised manner in Steam.
3. The games available on Steam do not have their price histories.

Outline

Entity Relationship Diagram



Overview of Seed Data

- In total **349 MB** worth of data
- Major tables:
  - 15450 games
  - 775510 comments (50 comments/game on average)
  - 436322 price history (28 histories/game on average)
  - 3095 backgrounds, 9391 cards, 3662 emotes

Outline

Test set-up

1. User arrival rates were modelled in 8 phases.
2. The work flow consisted of 4 distinct sessions with various probabilities.
3. Interspersed waiting within sessions.
4. Specialized tests were set up to test caching.

Outline

Results

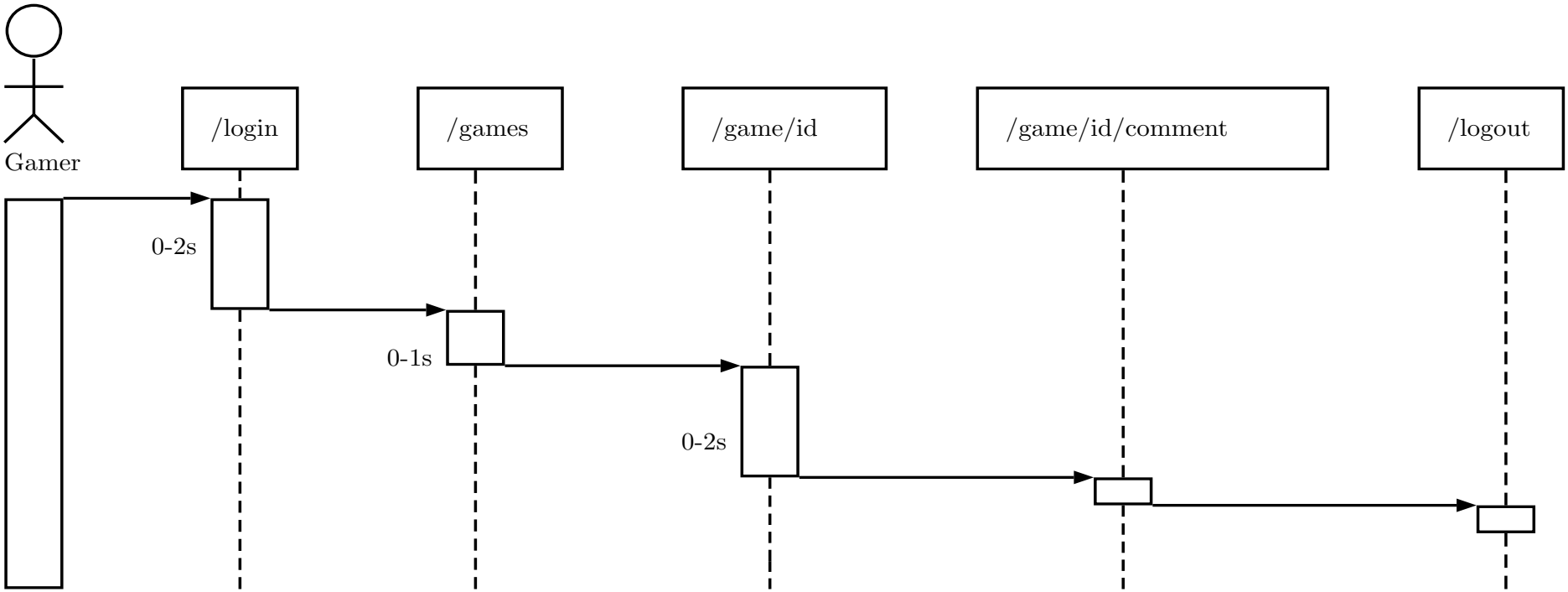


Figure: First Session

Outline

Session 2

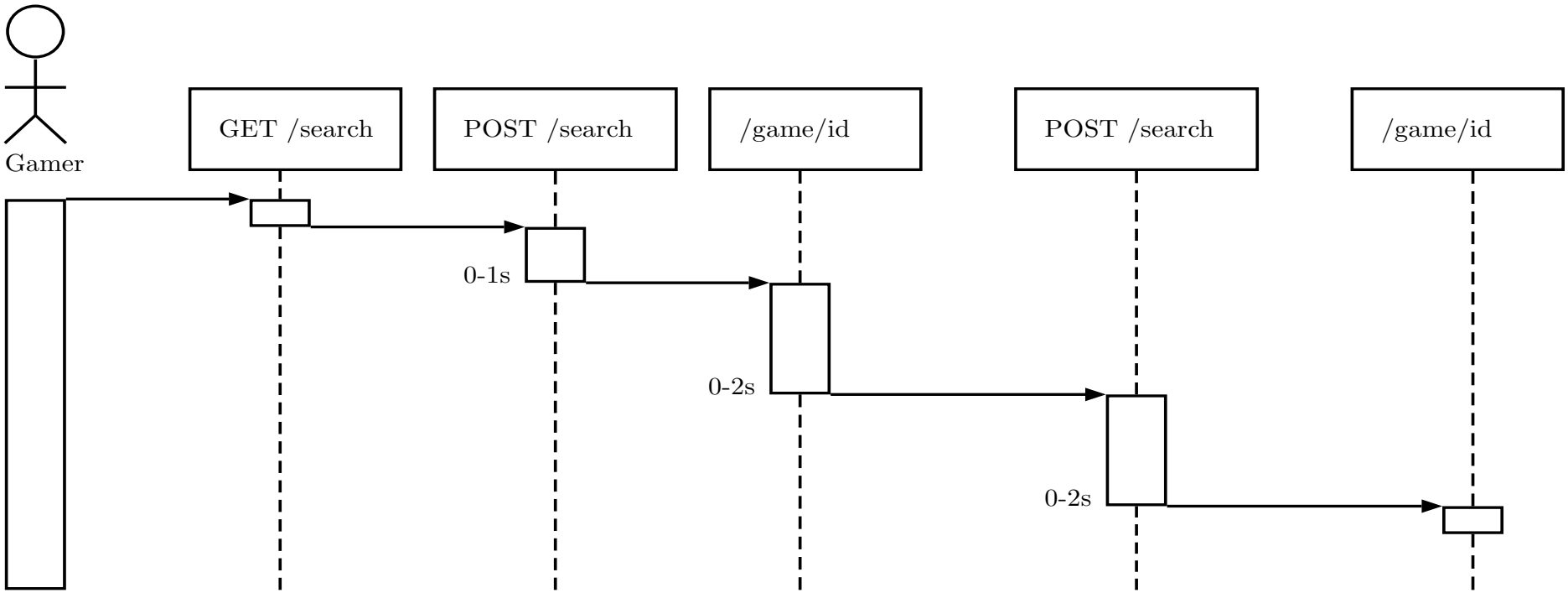


Figure: Second Session

Outline

Session 3

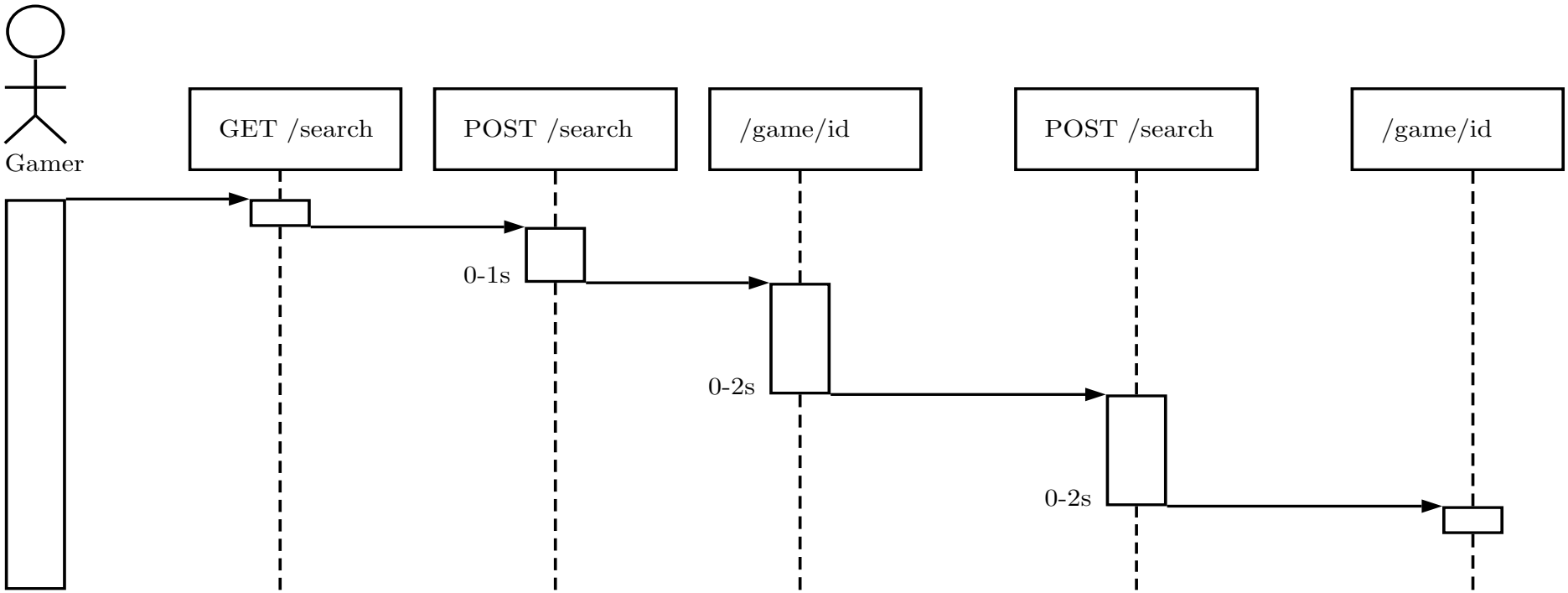


Figure: Third Session

Outline

Session 4

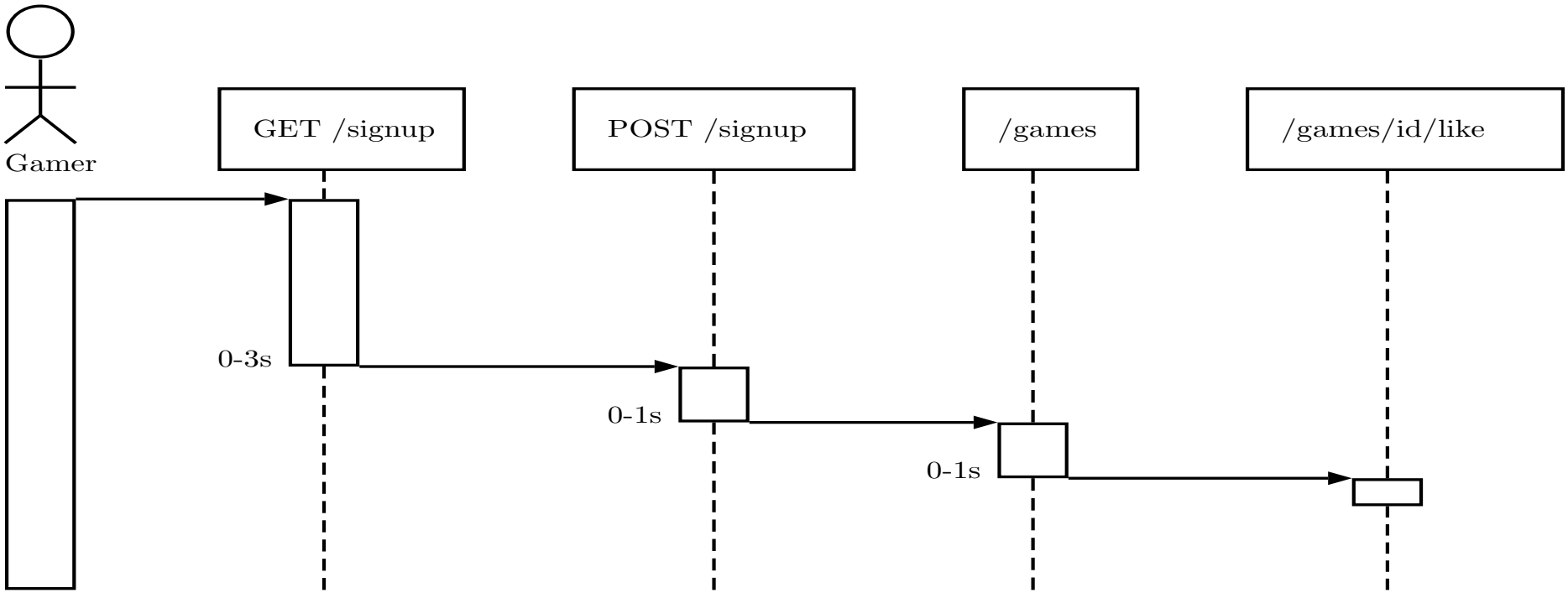
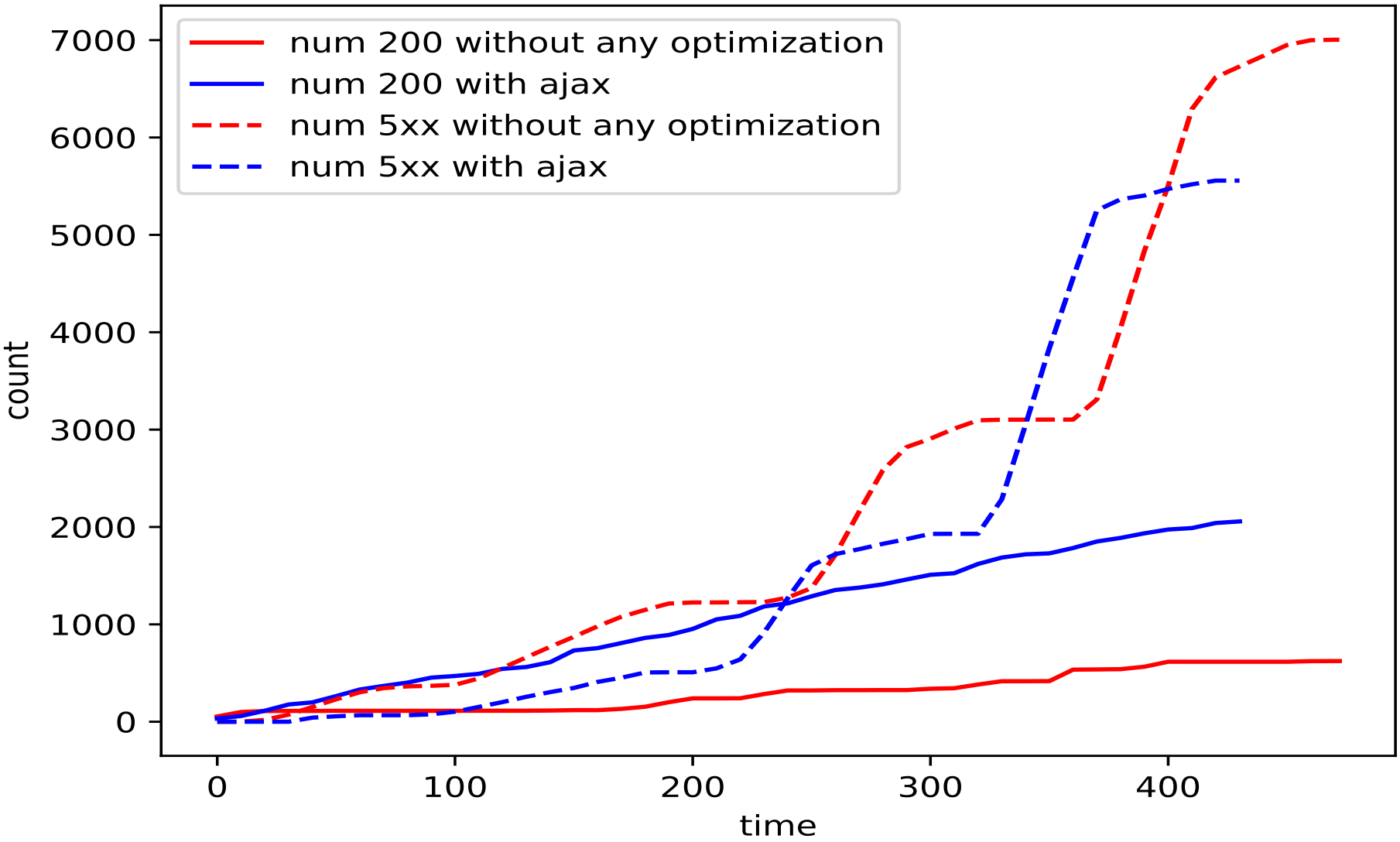


Figure: Fourth Session

Outline

Optimization 1: AJAX



Outline

Optimization 2: Indexing

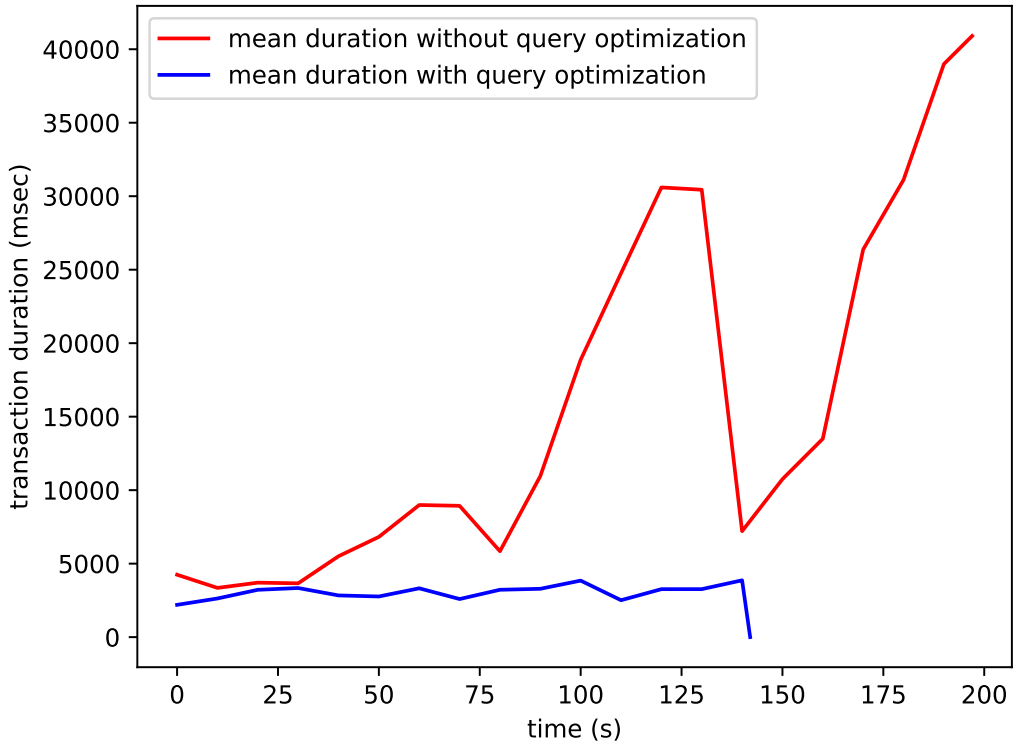


Figure: Mean duration for index page transaction without and with indexing.

Outline

Optimization 3: Caching

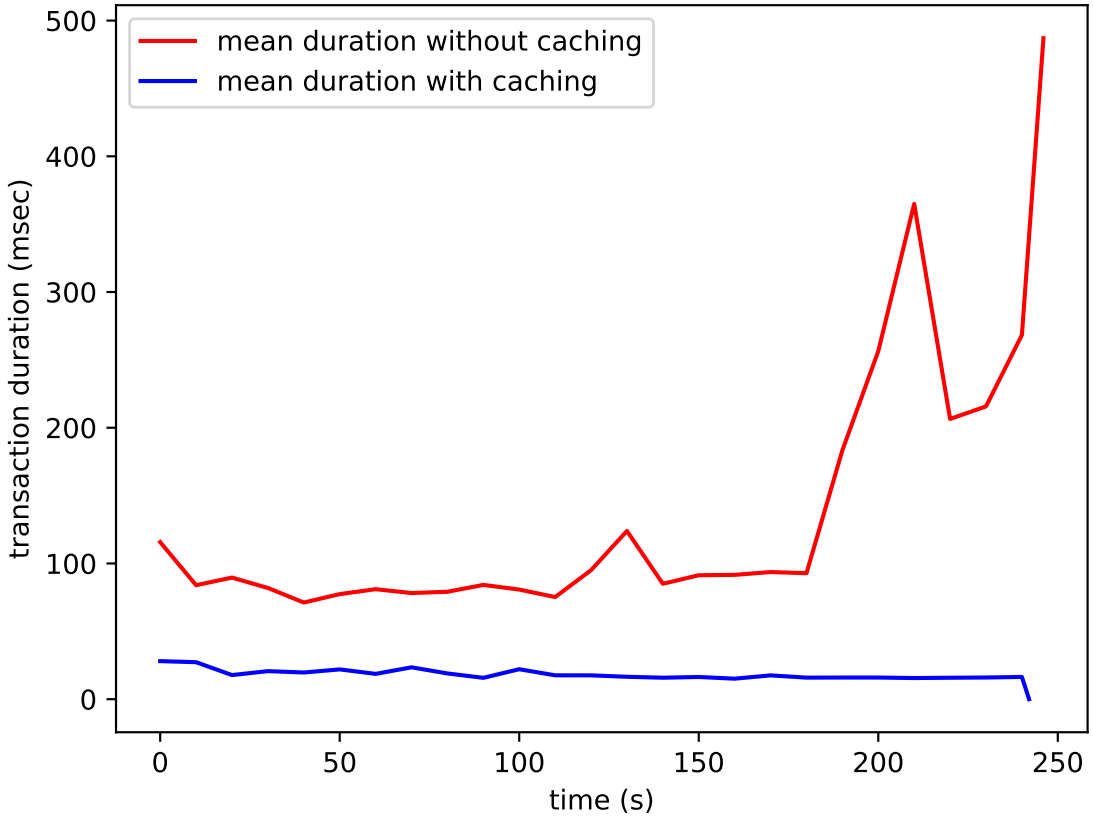


Figure: Mean duration for system requirement search transaction with and without caching.

Outline

Horizontal and Vertical Scaling

1. The website was load tested with various hardware configuration.
2. It was detected very early that the major bottleneck lay with the database.
3. The app server used was c5 with various database servers.

Outline

16 users/second arrival rate

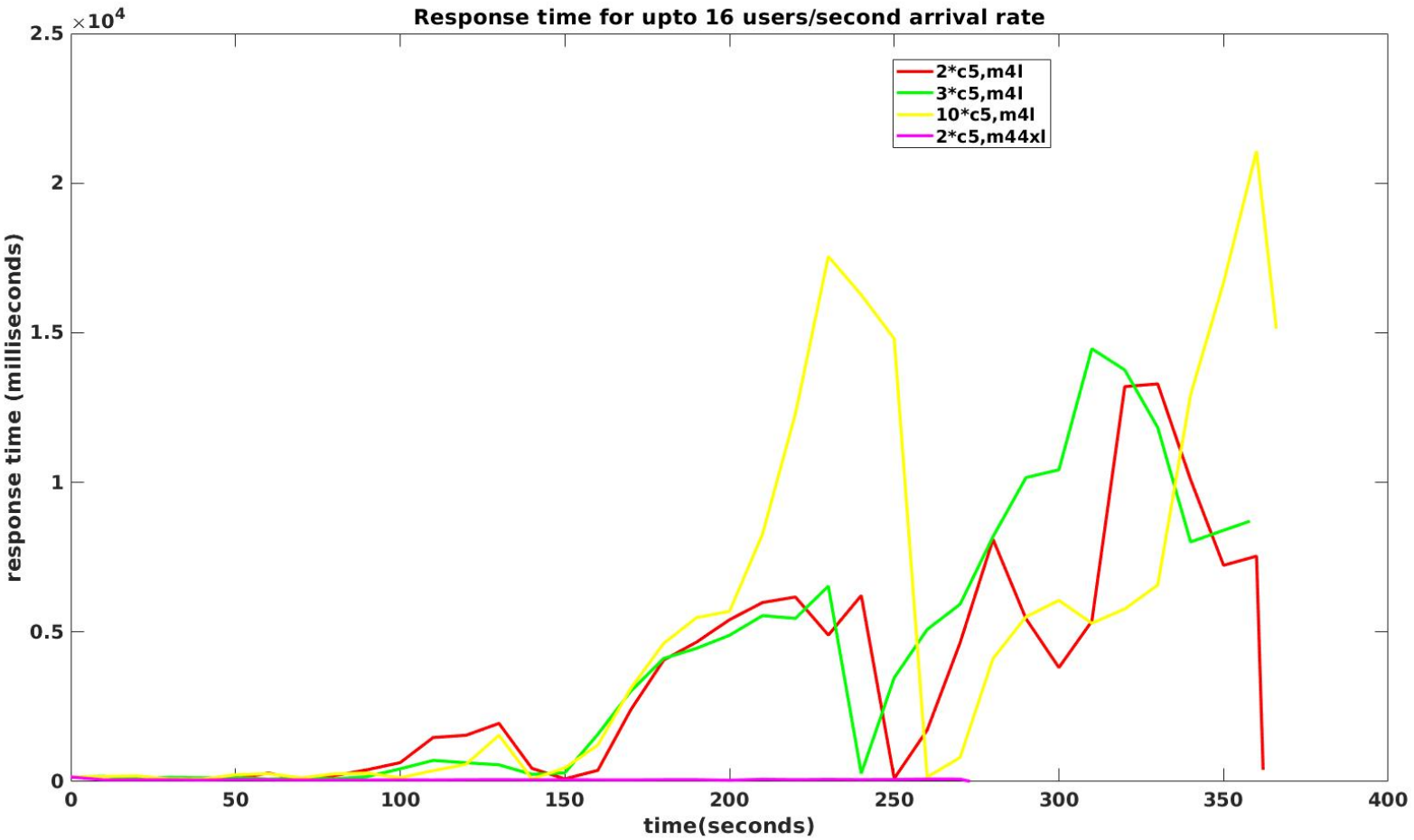


Figure: Mean response time while handling up to 16 users/second

Outline

32 users/second arrival rate

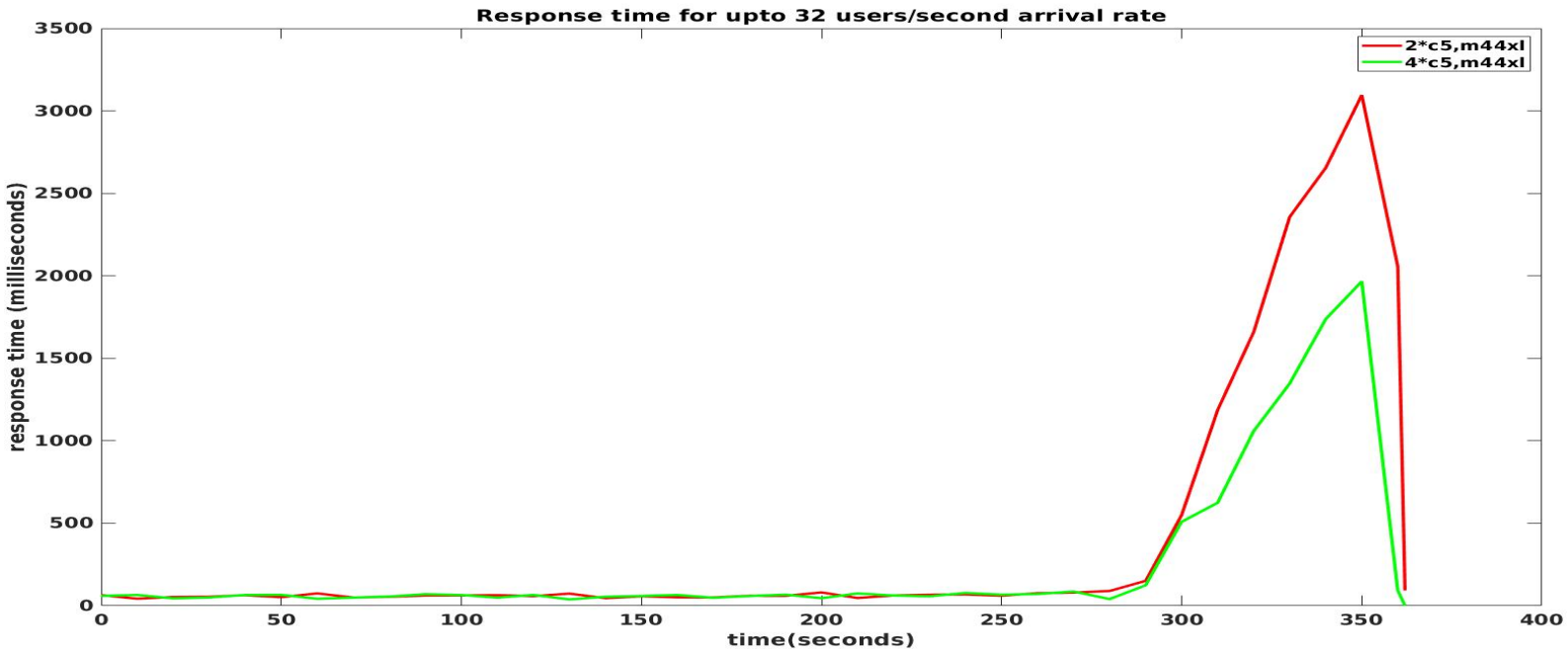


Figure: Mean response time while handling up to 32 users/second

Outline

64 users/second arrival rate

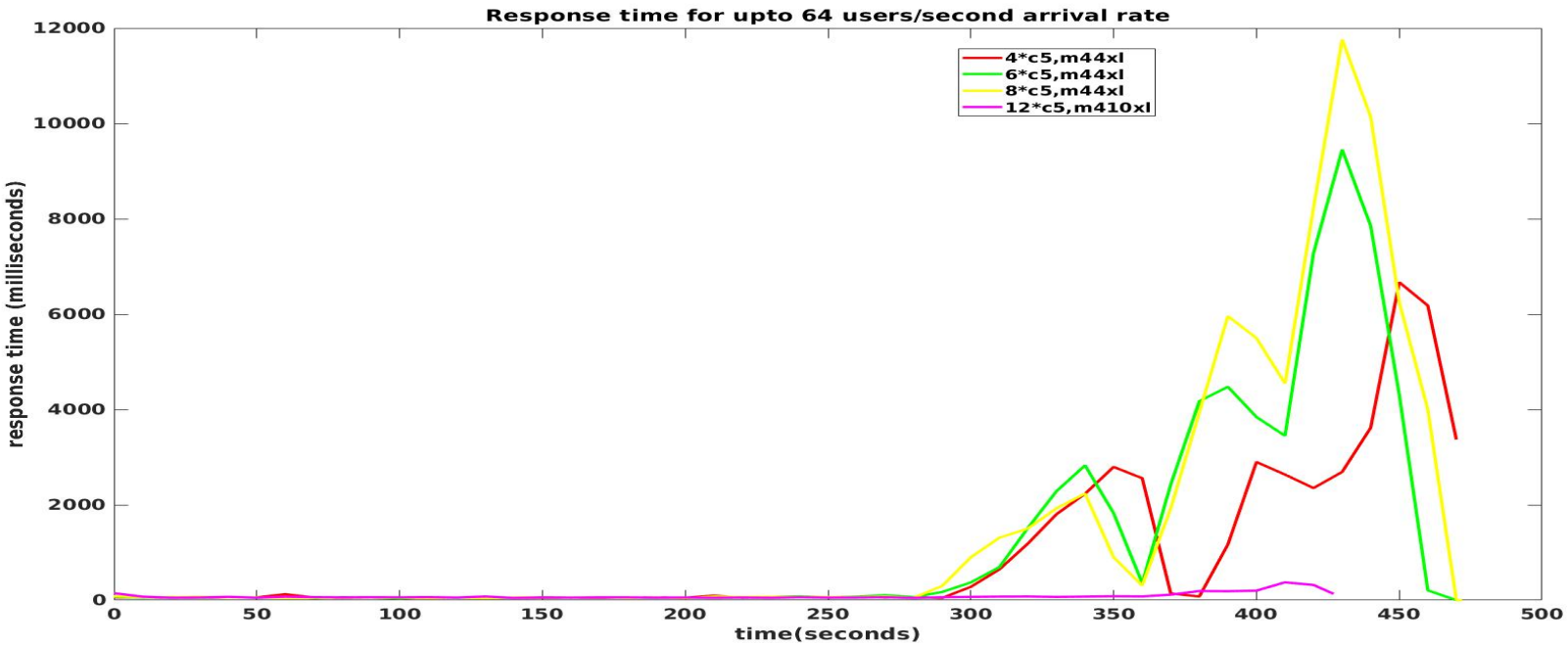


Figure: Mean response time while handling up to 64 users/second

Outline

Conclusion

1.