0-1 Million in 46 Days
Scaling a Facebook Application in Rails

Ikai Lan
Linkedin
Hi! I’m Ikai Lan

Ikai Lan
Loquacious Problem Solver/Causer
San Francisco Bay Area

Current
- Software Engineer at LinkedIn

Past
- Engineering Consultant at CollegeCarrot
- Systems Engineer at Citrix Systems
- Systems Administrator at San Jose State University

Education
- San Jose State University

Recommended
3 people have recommended Ikai
1 manager, 2 co-workers

Connections
172 connections

Industry
Computer Software

Public Profile
http://www.linkedin.com/in/ikailan
What this talk is not

- One-size-fits all solution to scaling problems
- Example source code
- An in-depth discussion of Ruby on Rails
What this talk is

- A look into the challenges we faced with scaling a rapid growth web application
This is a detective story.
Rather than telling you best practices
We’ll talk about how we came to define them.
Bumper Sticker?

- Investigation of the Facebook platform
- Investigation of Ruby on Rails as a serious development platform
Reid Hoffman’s wager
1 million users in 45 days!
Personal Badging

- Create and display badges
- Share badges
- Find badges
= Bumper Sticker!

Over 17 million Facebook users use Bumper Sticker to express themselves, entertain, and inspire one another by creating and sharing stickers. Browse through millions of stickers and add them to your profile or even stick your friends! Can't find a sticker?}

Go to this Application
Remove this Application

This application cannot be added to your Pages. Learn more.

Edit Application
Edit Admins
Send an Update to Fans
Promote Application with an Ad
Become a Fan
Block Application
View Insights
Share +
Success!
... 1 million in 46 days
Trouble in Paradise
Throw hardware at it!
More hardware!
This is not working.
Current state of affairs

- 7 million adds
- 20 million pageviews
- 3 million timeouts a day
Strategy

- Analyze MySQL slow queries log
- Application log
D’oh! No logging!
Logging -> Syslog

- Rails does not buffer writes to filesystem
Clue 1: Slaves ran several hours behind Master
Clue 2: Updates to add count

- One by one updates
Counter Cache

- SQL Updates not batched
- Very inefficient!
Result: less slow queries
Lesson 1: Buffer everything!
Still had lots of timeouts!
Clue 3: 80% of traffic to static content
Next attempt: edge caching!
Edge caching using fb:ref

- Facebook mechanism for storing data
Fb:ref on the load balancer!

- F5 BigIP set to conditionally respond to static URIs with fb:ref
- Background process to constantly update fb:ref in Facebook cache
Result: Static pages = perfect
Lesson 2: Edge cache when possible
But we were still serving millions of timeouts!
Clue 4: Most CPU time accepting stickers

- Request taking up to 10 seconds during peak
Slow database writes

```
Terminal — ssh — 129x43

# Query time: 7 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972922;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(16443515, 7842583, 6444045, NULL, '2008-09-02 04:48:37', 0);
# User@Host: bumper[bumper] @ [10.17.86.10]
# Query time: 9 Lock time: 0 Rows sent: 0 Rows examined: 0
SELECT * FROM stickers_users WHERE (stickers_users.sticker_id = 16631416 AND stickers_users.user_id = 5229062) LIMIT 1;
# User@Host: bumper[bumper] @ [10.17.86.10]
# Query time: 7 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972923;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(16443515, 61988872, 422082, NULL, '2008-09-09 04:48:37', 0);
# User@Host: bumper[bumper] @ [10.17.86.137]
# Query time: 7 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972924;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(16443515, 39943372, 74613552, NULL, '2008-09-09 04:48:37', 0);
# User@Host: bumper[bumper] @ [10.17.86.107]
# Query time: 7 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972925;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(166212737, 79975957, 85060827, NULL, '2008-09-02 04:48:37', 0);
# User@Host: bumper[bumper] @ [10.17.86.137]
# Query time: 7 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972926;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(166212737, 17106912, 7704382, NULL, '2008-09-02 04:48:37', 0);
# User@Host: bumper[bumper] @ [10.17.178.83]
# Query time: 6 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972927;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(166352684, 84511672, 7704382, NULL, '2008-09-02 04:48:37', 0);
# User@Host: bumper[bumper] @ [10.17.178.83]
# Query time: 6 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972928;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(166352684, 53887962, 58872912, NULL, '2008-09-02 04:48:38', 0);
# User@Host: bumper[bumper] @ [10.12.43.230]
# Query time: 6 Lock time: 0 Rows sent: 0 Rows examined: 0
SET Insert id: 966972929;
INSERT INTO stickers_users (sticker_id, user_id, invitor_id, position, created_at, hidden)
VALUES(166352684, 61009031, 31811134, NULL, '2008-09-02 04:48:38', 0);
```
1st attempt: Background process writes
Result: Timeouts dropped significantly!
Success ... ?
Application would not seem responsive to user actions and seem out of sync.
2nd attempt: Two step write
Step 1: Synchronous write to MemCache

Step 2: Asynchronous write to database (queue)
Result: Success!
Lesson 3: Use asynchronous processing when possible
Lesson 3.1: Keeping data in sync is not trivial!
Lesson 3.2: Lack of concurrency in Rails presents problem earlier
Lesson 3.3: Very few problems can be reduced to a linear equation
QED: Rails can scale.

\[
\text{Need } x_0 \in \mathbb{R}^n \ \text{ s.t. } A x = 0, \ \text{so } x_0 \text{ satisfies } a + \square \Rightarrow x_0 \text{ s.t. } r_1, r_2, \ldots, r_n
\]

\[
\text{Now prove } x' \in \mathbb{R}^n \ \text{s.t. } C x' = 0, \ \text{so } x' = \square \Rightarrow r_m = r_n' a \Rightarrow r_2 = r_2' \Rightarrow r_1 = r_1' - ar_2'
\]
Latest stats

- 16 million installs
- 30 million pageviews a day
- Average response time of <80ms
Scaling has a price!
Complexity
No new functionality!
Key takeaways

- Make long running tasks asynchronous
- Use edge caching
- Buffer long operations when possible
The bigger picture

- Language choice is less important than architecture
- Be creative – brute force doesn’t work
- Be ready to be wrong
Questions?
Thanks for coming!

- ilan@linkedin.com
- http://www.linkedin.com/in/ikailan