From Dandelion to Tree: Scaling Slack

Bing Wei
Infra@Slack
Hey team, hoping to have that proposal ready for the Alaska clients by 3pm today, how are we doing? I can chip in wherever needed!

Carl Benting
I’m just about finished putting together the estimate portion of it, I could use some feedback. Here’s the google doc I’m working on... docs.google.com/bin/proposal

Q3 OOH — Cost Estimate
Google Drive Document

The numbers look pretty good, I tweaked a few things, but we’re good to go!

Reena Baines
I’m just wrapping up the sketches, I’ll post them here once I’m done!
From supporting *small* teams

To serving *gigantic* organizations of hundreds of thousands of users
Biggest Teams

2015
8,000 users

2016
26,000 users
2015
8,000 users

2016
26,000 users

today
266,000 users
Challenge

Slowness Connecting to Slack
User Connect Flow in 2015

1. https://slack.com/api/rtm.start

2. HTTP response:
   a snapshot of the team

3. Long-lived WebSocket connection
   real time events

Client gets eventually consistent snapshot of the team
User Connect Flow in 2015

1. https://slack.com/api/rtm.start

2. HTTP response:
   a snapshot of the team

3. Long-lived WebSocket connection
   real time events
User Connect Flow in 2015

1. https://slack.com/api/rtm.start
2. HTTP response: 
   a snapshot of the team
3. Long-lived WebSocket connection with real time events

It grows with the team size!!
Megabytes of data
## Team Snapshot Size

<table>
<thead>
<tr>
<th>Number of users</th>
<th>Number of channels</th>
<th>Snapshot size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td>10</td>
<td>200K</td>
</tr>
<tr>
<td>500</td>
<td>200</td>
<td>2.5M</td>
</tr>
<tr>
<td>3,000</td>
<td>7,000</td>
<td>20M</td>
</tr>
<tr>
<td>30,000</td>
<td>1,000</td>
<td>60M</td>
</tr>
</tbody>
</table>
Client Lazy Loading

Download less data at boot time

Lazy loading on demand
Client Lazy Loading

**Client**

Rewrite data model layer to load data asynchronously.

**Server**

Fulfill client requests with low latency.
Flannel: An Edge Cache Service

A query engine backed by cache on edge locations
Staged Development

Stage 1: “A man in the middle” + Just-in-time Annotation

Goal: Deploy as fast as possible
with minimal dependency on other architectural changes

Stage 2: Pub/Sub model to update cache

Goal: Optimize for performance
Flannel V1: “A Man in the Middle”

Client

WebSocket

chat messages & events
sync to the client

Real Time Message API
Flannel V1: “A Man in the Middle”

Use real time events to update cache
E.g. user creation, channel creation, user joins a channel, etc.
Flannel V1: Just-In-Time Annotation

Client A
user Alice + msg

Client B
msg

Client C
msg

Cache

msg from Alice

Real Time Message API

Heuristic: client A may not have user Alice object in its local cache
Flannel Deployment

Client → GeoDNS

Edge Region B

HAProxy

Team Affinity

Edge Region A

HAProxy

Team Affinity

Edge Region C

HAProxy

Team Affinity
Areas for Improvement after Stage 1

- Client A
- Client B
- Client C

Cache updates tied to WebSocket
Cold Cache/Tail latency

Duplicated Msg

Real Time Message API
Flannel V2: Pub/Sub for Cache Updates

Client → WebSocket → Cache → WebSocket → Cache

Pub/Sub for cache updates

Real Time Message API
events reduce by 500
Flannel Query Latency
FAQs

● Why not use Memcache?

● How is Pub/Sub implemented?

● How do you deal with reconnect storm?
2017-10-31 Slack is down

**Thundering Herd:** A Gateway Server deployment exercises a bug in Flannel, causing ~40% of the Flannel fleet to crash and restart. The number of connected clients dropped from 3.2m to 1.6m between 15:50 and 15:52. This thundering herd
We Used to Have Rate Limits

- Hard to get right
  (Specially hard when incident happens)
- Manually tuned => Slow to take effect
Failures are inevitable, how do we
1. Avoid cascading failures
2. Minimize recovery time?
Automatic Resilience Measures

Admission Control
Automatic Resilience Measures

Circuit Breaker
In Progress

Auto Scaling

Regional Failover
What I learned

- Innovative solutions
- Incremental changes
- Learn by doing
Thank You!

@bingw11