Scaling Pinterest

Marty Weiner
Level 83 Interwebz Geek
Evolution
Growth
March 2010

- RackSpace
- 1 small Web Engine
- 1 small MySQL DB
- 1 Engineer + 2 Founders
Growth

March 2010
Scaling Pinterest

January 2011

- Amazon EC2 + S3 + CloudFront
- 1 NGinX, 4 Web Engines
- 1 MySQL DB + 1 Read Slave
- 1 Task Queue + 2 Task Processors
- 1 MongoDB
- 2 Engineers + 2 Founders
Scaling Pinterest

September 2011

- Amazon EC2 + S3 + CloudFront
- 2 NGINX, 16 Web Engines + 2 API Engines
- 5 Functionally Sharded MySQL DB + 9 read slaves
- 4 Cassandra Nodes
- 15 Membase Nodes (3 separate clusters)
- 8 Memcache Nodes
- 10 Redis Nodes
- 3 Task Routers + 4 Task Processors
- 4 Elastic Search Nodes
- 3 Mongo Clusters
- 3 Engineers (8 Total)

Page views per day

It will fail. Keep it simple.
If you’re the biggest user of a technology, the challenges will be greatly amplified
Scaling Pinterest

January 2012

Growth
Growth

April 2012

- Amazon EC2 + S3 + Edge Cast
- 12 Engineers
- 135 Web Engines + 75 API Engines
  - 1 Data Infrastructure
  - 10 Service Instances
  - 1 Ops
- 80 MySQL DBs (m1.xlarge) + 1 slave
  - 2 Mobile
    - each
  - 8 Generalists
- 110 Redis Instances
- 10 Non-Engineers
- 60 Memcache Instances
- 2 Redis Task Manager + 60 Task Processors
- 3rd party sharded Solr

Page views per day

Growth
April 2013

- 65+ Engineers
- Amazon EC2 + S3 + Edge Cast
- 7 Data Infrastructure + Science
- 400+ Web Engines + 400+ API Engines
- 7 Search and Discovery Engines
- 9 Business and Platform
- 70+ MySQL DBs (hi.4xlarge on SSDs)
  - 6 Spam, Abuse, Security
  - 1 slave each
- 9 Web
- 100+ Redis Instances
- 9 Mobile
- 230+ Memcache Instances
- 2 growth
- 10 Redis Task Manager + 500 Task Processors
- 6 Ops
- 65+ Engineers (130+ total)
- 65+ Non-Engineers
- 6 services (60 instances)
- Sharded Solr
- 20 HBase
- 12 Kafka + Azkabhan
- 8 Zookeeper Instances
- 12 Varnish

Page views per day

April 2012

April 2013
Technologies
Arch Overview

Scaling Pinterest

ELB

Routing & Filtering
(Varnish)

API
(Python)

Web App
(Python / JS / HTML)

Task Processing
(PinLater)

Pin Images
(S3)

All connection pairings managed by ZooKeeper

MySQL Service
(Java/Finagle)

Memcache Mux
(Nutcracker)

Follower Service
(Python/Thrift)

Feed Service
(Python/Thrift)

Search Service
(Python/Thrift)

Spam Service
(Python/Thrift)

Sharded
MySQL

Memcache

Redis

HBase (Zen)

Puppet

StatsD

CDN

Scaling Pinterest
Data Pipeline

- Web App (Python)
- API App (Python)
- Task Processing
- Kafka
- Secor
- Spam Processing
- Pinball
- S3
- Qubole
- Redshift
Our MySQL Sharding?

http://www.infoq.com/presentations/Pinterest
Questions to ask

- Does it meet your needs?
- How mature is the product?
- Is it commonly used? Can you hire people who have used it?
- Is the community active?
- How robust is it to failure?
- How well does it scale? Will you be the biggest user?
- Does it have a good debugging tools? Profiler? Backup software?
- Is the cost justified?
Maturity = \frac{\text{Blood and Sweat}}{\text{Complexity}}
Choosing Your Tech

Questions to ask

• Does it meet your needs?
• How mature is the product?
• Is it commonly used? Can you hire people who have used it?
• Is the community active?
• How robust is it to failure?
• How well does it scale? Will you be the biggest user?
• Does it have a good debugging tools? Profiler? Backup software?
• Is the cost justified?
Hosting

Why Amazon Web Services (AWS)?

- Variety of servers running Linux
- Very good peripherals: load balancing, DNS, map reduce, basic security, and more
- Good reliability
- Very active dev community
- Not cheap, but...
- New instances ready in seconds
Hosting

AWS Usage

- Route 53 for DNS
- ELB for 1st tier load balance
- EC2 Ubuntu Linux
  - Varnish layer
  - All web, API, background appliances
  - All services
  - All databases and caches
- S3 for images, logs
Why Python?

- Extremely mature
- Well known and well liked
- Solid active community
- Very good libraries specifically targeted to web development
- Effective rapid prototyping
- Open Source

Some Java and Go...

- Faster, lower variance response time
Python Usage

• All web backend, API, and related business logic

• Most services

Java and Go Usage

• Varnish plugins

• Search indexers

• High frequency services (e.g., MySQL service)
Production Data

Why MySQL and Memcache?

- Extremely mature
- Well known and well liked
- (MySQL) Rarely catastrophic loss of data
- Response time to request rate increases linearly
- Very good software support: XtraBackup, Innotop, Maatkit
- Solid active community
- Open Source
**Production Data**

**MySQL and Memcache Usage**

- Storage / Caching of core data
- Users, boards, pins, comments, domains
- Mappings (e.g., users to boards, user likes, repin info)
- Legal compliance data
Production Data

Why Redis?

- Well known and well liked
- Active community
- Consistently good performance
- Variety of convenient and efficient data structures
- 3 Flavors of Persistence: Now, Snapshot, Never
- Open Source
Production Data

Redis Usage

- Follower data
- Configurations
- Public feed pin IDs
- Caching of various core mappings (e.g., board to pins)
Production Data

Why HBase?

- Small, but growing loyal community
- Difficult to hire for, but...
- Non-volatile, $O(1)$, extremely fast and efficient storage
- Strong Hadoop integration
- Consistently good performance
- Used by Facebook (bigger than us)
- Seems to work well
- Open Source
HBase Usage

- User feeds (pin IDs are pushed to feeds)
- Rich pin details
- Spam features
- User relationships to pins
Production Data

What happened to Cassandra, Mongo, ES, and Membase?

• Does it meet your needs?
• How mature is the product?
• Is it commonly used? Can you hire people who have used it?
• Is the community active? Can you get help?
• How robust is it to failure?
• How well does it scale? Will you be the biggest user?
• Does it have a good debugging tools? Profiler? Backup software?
• Is the cost justified?
A 2nd chance...
Stuff we could have done better

- Logging on day 1 (StatsD, Kafka, Map Reduce)
- Log every request, event, signup
- Basic analytics
- Recovery from data corruption or failure
- Alerting on day 1
A 2nd Chance

Stuff we could have done better

- Shard our MySQL storage much earlier
- Once you start relying on read slaves, start the timebomb countdown
- We also fell into the NoSQL trap (Membase, Cassandra, Mongo, etc)
- Pyres for background tasks day 1
- Hire technical operations eng earlier
- Chef / Puppet earlier
- Unit testing earlier (Jenkins for builds)
A 2nd Chance

Stuff we could have done better

• A/B testing earlier
• Decider on top of Zookeeper WATCH
• Progressive roll out
• Kill switches
Looking Forward

- Beyond 400 Pinployees
- Continually improve Pinner experience
  - Help Pinners discover more of the things they love
- Build better and faster
- Continually improve collaboration and build bigger, better, faster products
Have fun
No Seriously, Have fun
Thanks!

Questions?

marty@pinterest.com
pinterest.com/martaaay