Server Side Refactoring@AWS
by
Triton Ho@hypebeast.com

About hypebeast.com

- fashion website
- 3.7M monthly unique visitor
- Co-related website: hbx.com
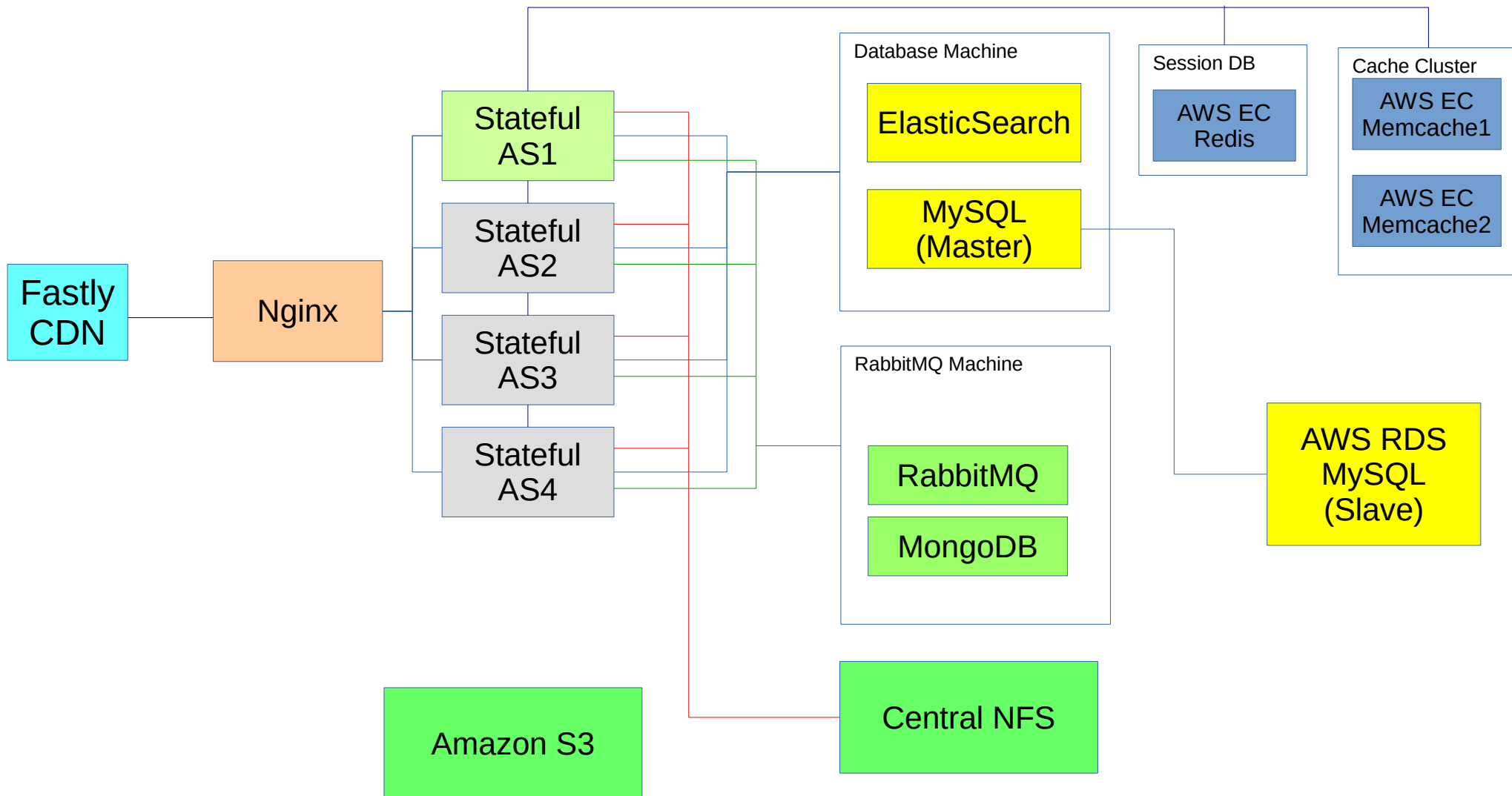- unpredictable traffic burst

Technical challenges

- High traffic! High traffic! High traffic!
- Lack of manpower
  - 猫の手も借りたい
- Brittle system
- Dogs in the office

Problems

- Multiple Single-Point-of-Failure
- Multiple performance bottleneck
- Hard for system upgrade
- Capacity Limit
- Incompatible components

Hypebeast.com@2015

Fastly CDN — **Nginx**

Stateful AS1
Stateful AS2
Stateful AS3
Stateful AS4

Database Machine
ElasticSearch
MySQL (Master)

Session DB
AWS EC Redis

Cache Cluster
AWS EC Memcache1
AWS EC Memcache2

RabbitMQ Machine
RabbitMQ
MongoDB

AWS RDS MySQL (Slave)

Amazon S3

Central NFS

Multiple Stage of Revamp

- Add kubernates
- Migrate MySQL to Mariadb@RDS
- Migrate NFS to AWS EFS
- Migrate Mongodb, RabbitMQ to kubernetes
- Migrate ElasticSearch to AWS ES
- Migrate LoadBalancer to AWS ELB
- Bug fix on AS, become Stateless
- Migrate Memcache from AWS EC to AS localhost

kubernates

- Motivation: too many system component
  - Example: Microsoft chatbot, multiple Wordpress
  - PHP 5.5, 5.6, 7.0……
  - Node.JS 4.0, 6.0……

Our Requirement on k8s

- As a platform to host various dockerized component

- Auto container restart after crash

- Store environment variable

- Easy service endpoint setup

- Allow Internal Service endpoint

- Deployment management

Why Not AWS ECS

- Troublesome and unmanagable
- Shattered documentation and components
  - AWS Route 53?
  - AWS CodeDeploy?
- Not a better solution than k8s

Deployment on k8s

- "kube deploy projectX" on slack
- HUBOT run deploy.sh
- cd ~/projects/projectX *&& git pull*
- Build docker image
- Upload the docker image to AWS ECR
- Rolling deployment

Internal Service on k8s

- k8s has own Amazon VPC.
- Thus, VPC peering connection is needed.
- Add pcx-XXXXXXXX to your VPC Routing Table
- For each Internal Service
  - Create k8s services(type = NodePort).
  - Create a AWS Internal ELB
  - Add the Internal ELB to k8s Auto Scaling Group

Afterthought on k8s + docker

- Costed one month to study + minesweeping + deploy.

- 10.0.0.0/8 is hardcoded as pod's IP.
  - Collides with AWS RDS default IP range.

- Default instance type m3.medium is too small

- For new service:
  - >=1 days to prepare the DockerFile
  - 2-3 hours to setup on k8s

- Private library repo caused great trouble in DockerFile

Mariadb@RDS

- Motivation:
  - Mariadb is more advanced
  - Replication/Failover is not easy and error-prone
    - Setup is easy, but automation is hard.

- Self Hosted instance is hard for applying patch, instance scaling

Near-Zero Migration To RDS

- mysqldump the production mysql
- Create the RDS instance(with Multi-AZ) and then import the data
- Setup the replication from mysql to mariadb
  - mysql.rds_start_replication on RDS
- Switch the production connection from mySQL to mariadb
- Stop the replication
  - mysql.rds_stop_replication on RDS

Afterthought on RDS

- Highly vendor lock-in.
  - No external replication
- Multi-AZ
  - Robbery, 2x price but no readonly-slave
  - After instance-type change, the avaliability-zone change too
    - EXTRA 1MS DOES MATTER FOR BAD CODE
- No c4 instance, bad for MySQL and mariadb
- In config, no CPUNum variable

Amazon EFS

- Motivation:
  - Need a flexible size NFS
  - Need a global storage for wordpress

Afterthought on AWS EFS

- Just a NFS

- On Ubuntu 16.04.1, failed to automount

- No monitoring

- Poor performance for hosting php code
    - Any one particular file can have up to 87 locks across all users of the file system.

Mongodb+RabbitMQ@k8s

- Motivation:
  - Used by side project only, low usage
  - Switch off the EC2 instance

Afterthought on k8s

- Amazon internal-ELB has idle timeout 60s
  - Our mongodb / MQ connection raise exception every 60 sec.
  - Increase the idle time to 3600 sec
- EBS is used for data storage.
- Why not AWS SQS:
  - We need delayed Messaging.

AWS ES

- Motivation:
  - Lack of Manpower
    - 猫の手も借りたい
  - Scalability and HA

Afterthought on AWS ES

- WTF, no security group and security group

- For hypebeast.com, t2.medium is better

From nginx to AWS ELB

- Motivation:
  - Central Nginx become bottleneck
  - Central Nginx is the single point of failure

- Original:
  - Two nginx tier
  - One central nginx for LB(ip_hash) and url rewrite
  - Nginx on each application server for PHP

Afterthought on AWS ELB

- Stickly loadBalancer will override X-Forwarded-Proto

    – Serious problem for CDN

- Non Stickly loadBalancer raise bug in our code.

- Shit……

Revamp on AS

- Motivation:
  - Make the AS become stateless
  - Increase performance by upgrade OS version and PHP version
  - Reduce Network latency by using localhost memcache

Afterthought on AWS ElasticCache

- Just waste your money

- Application server is CPU-bound, Cache server is RAM-bound

  - Combining them is perfect

- Using localhost memcache, the avg time on cache decreased from 30ms to 8ms

Conclusion

- AWS is good for startup to save manpower

- But AWS feature is not enough for advanced usage

  - Especially you have technical debt

- If you just need VM, gCloud may be much better.