基于 SpringMVC 架构 WEB 应用系统优化研究

答辩人：武斌

导师：郑海永
电子系 中国海洋大学

Department electronic engineering
Ocean University Of China

2017 年 5 月 25 日

目录

来源及目的

题目来源

- 本人在海信参与的项目，负责 DevOps 的研究和实践，目的是实现产品的持续交付和快速部署。

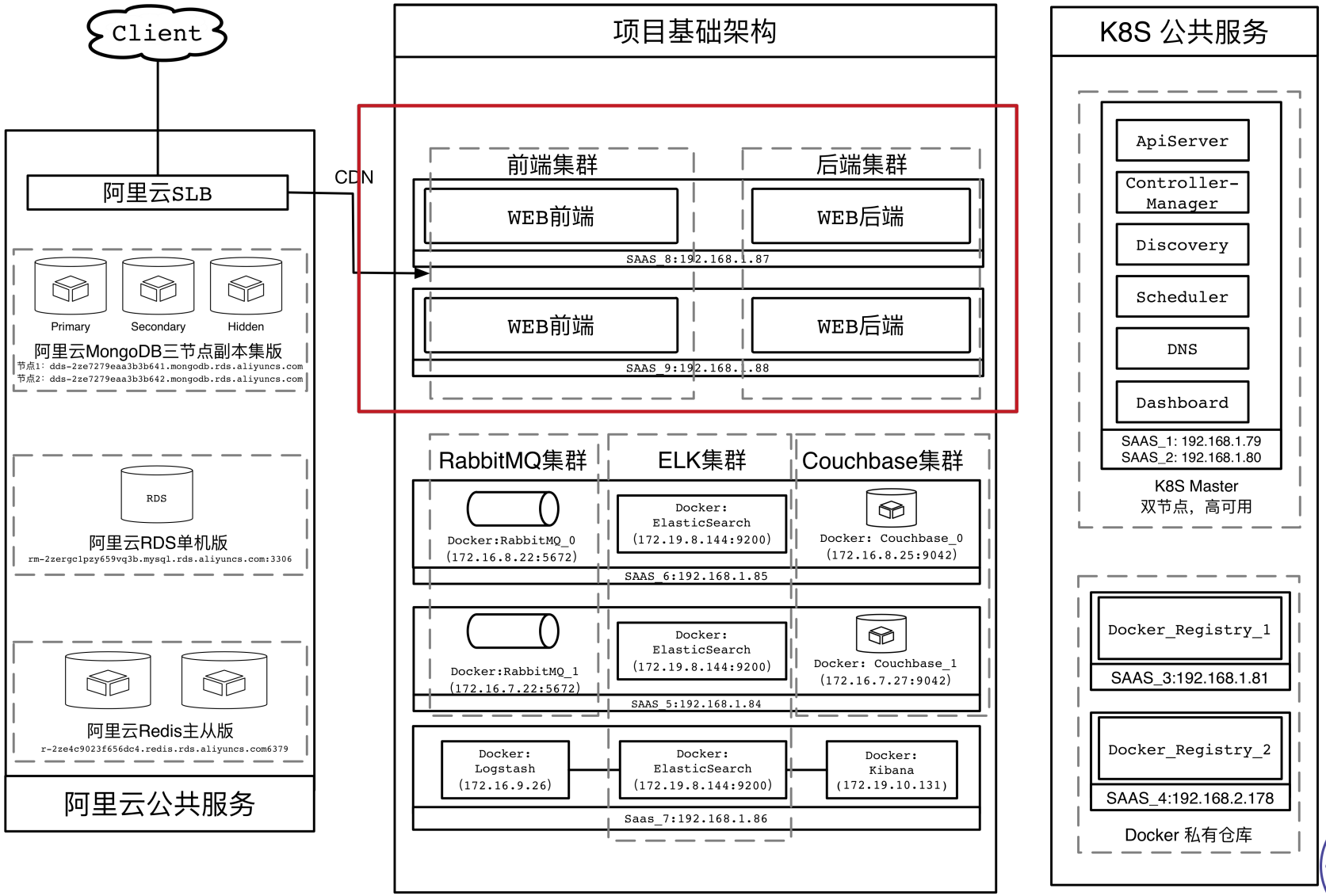
研究目的

- 研究系统开发过程中的代码交付、持续集成、服务器运维等方面的优化策略
- 基于现有的 WEB 项目研究一种切实可用的 WEB 平台开发运维优化方案，为有相同需求的研发人员提供借鉴和参考

方法及内容

1. 研究的基本框架
2. 应用性能优化
3. 数据库性能优化
4. 服务器稳定性优化

平台开发基本框架


Client
阿里云SLB
CDN
Primary
Secondary
Hidden
阿里云MongoDB三节点副本集版
节点1: dds-2ze7279eaa3b3b641.mongodb.rds.aliyuncs.com
节点2: dds-2ze7279eaa3b3b642.mongodb.rds.aliyuncs.com
RDS
阿里云RDS单机版
rm-2zergclpzy659vq3b.mysql.rds.aliyuncs.com:3306
阿里云Redis主从版
r-2ze4c9023f656dc4.redis.rds.aliyuncs.com6379
阿里云公共服务
项目基础架构
前端集群
后端集群
WEB前端
WEB后端
SAAS_8:192.168.1.87
WEB前端
WEB后端
SAAS_9:192.168.1.88
RabbitMQ集群
ELK集群
Couchbase集群
Docker:RabbitMQ_0
(172.16.8.22:5672)
Docker:
ElasticSearch
(172.19.8.144:9200)
Docker: Couchbase_0
(172.16.8.25:9042)
SAAS_6:192.168.1.85
Docker:RabbitMQ_1
(172.16.7.22:5672)
Docker:
ElasticSearch
(172.19.8.144:9200)
Docker: Couchbase_1
(172.16.7.27:9042)
SAAS_5:192.168.1.84
Docker:
Logstash
(172.16.9.26)
Docker:
ElasticSearch
(172.19.8.144:9200)
Docker:
Kibana
(172.19.10.131)
Saas_7:192.168.1.86
K8S 公共服务
ApiServer
Controller-Manager
Discovery
Scheduler
DNS
Dashboard
SAAS_1: 192.168.1.79
SAAS_2: 192.168.1.80
K8S Master
双节点，高可用
Docker_Registry_1
SAAS_3:192.168.1.81
Docker_Registry_2
SAAS_4:192.168.2.178
Docker 私有仓库

研究的主要内容

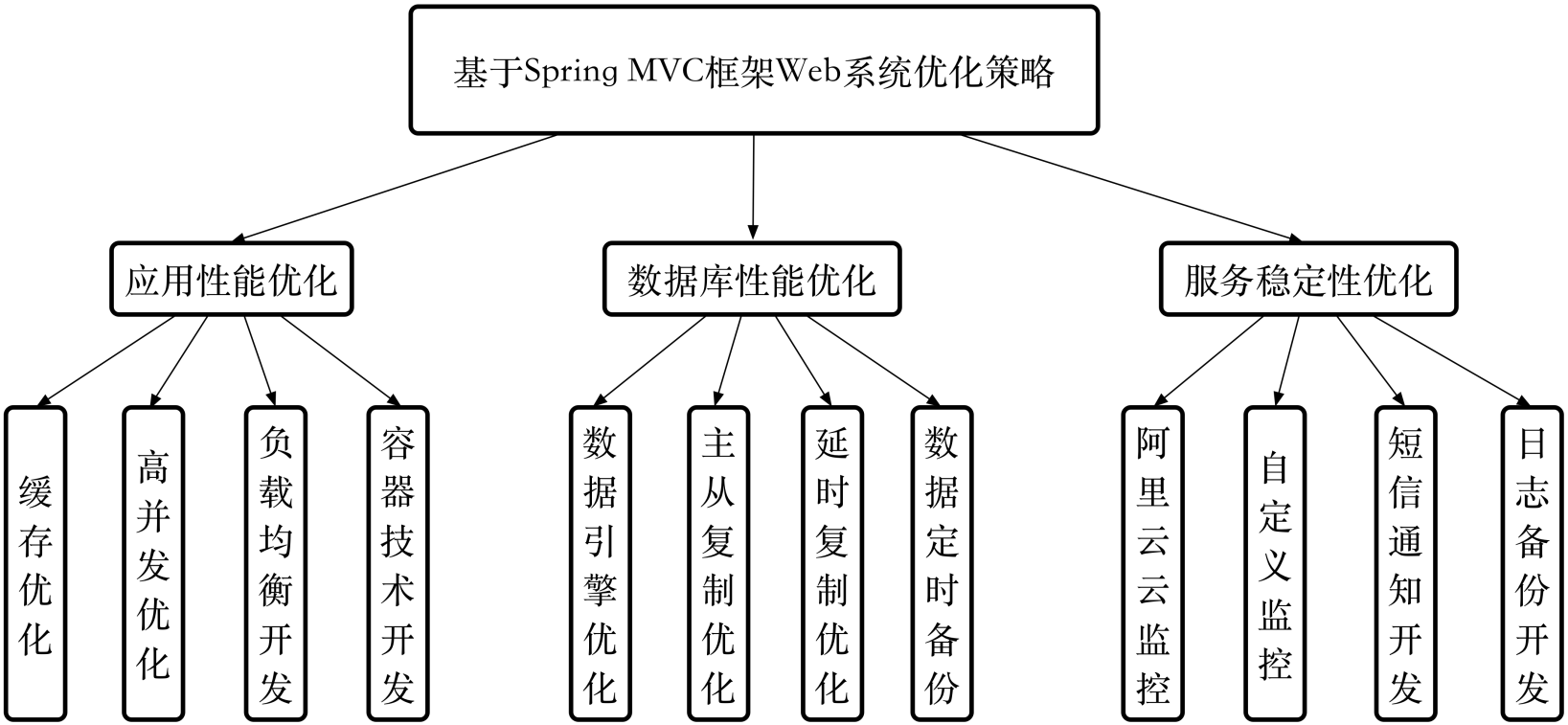
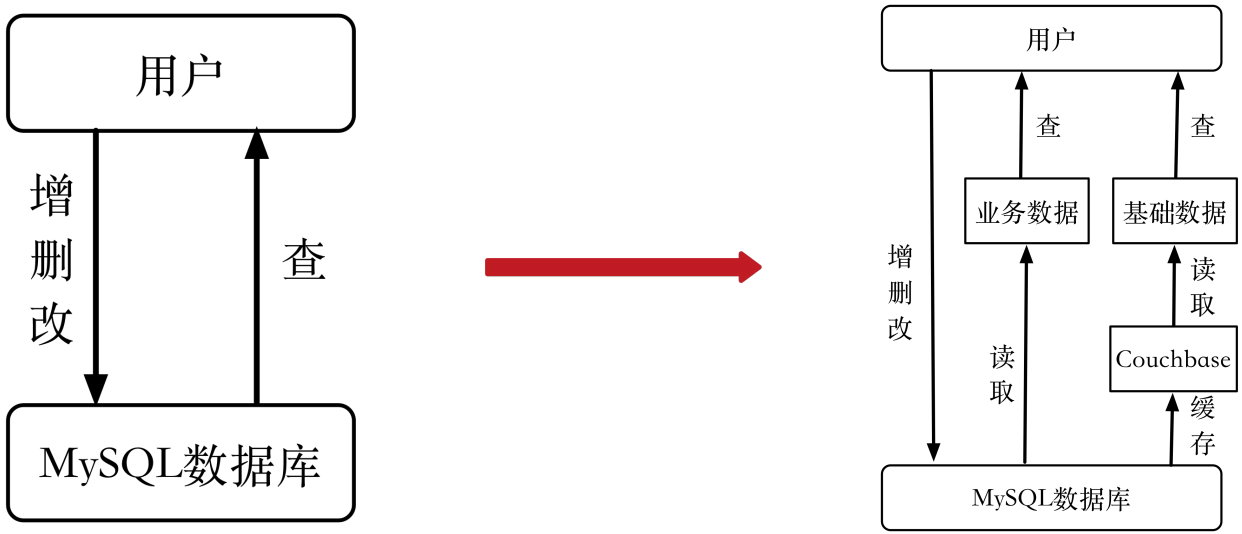


图: 平台功能

1. Couchbase 缓存优化

- 将项目的基础数据在首次访问时加载到缓存中



测试参数	关闭 Couchbase	开启 Couchbase
单次请求用时	0.670(s)	0.113(s)
请求失败次数	82	7

注: 10W 次请求，每次 100 个并发

2. Tomcat 高并发 APR 优化

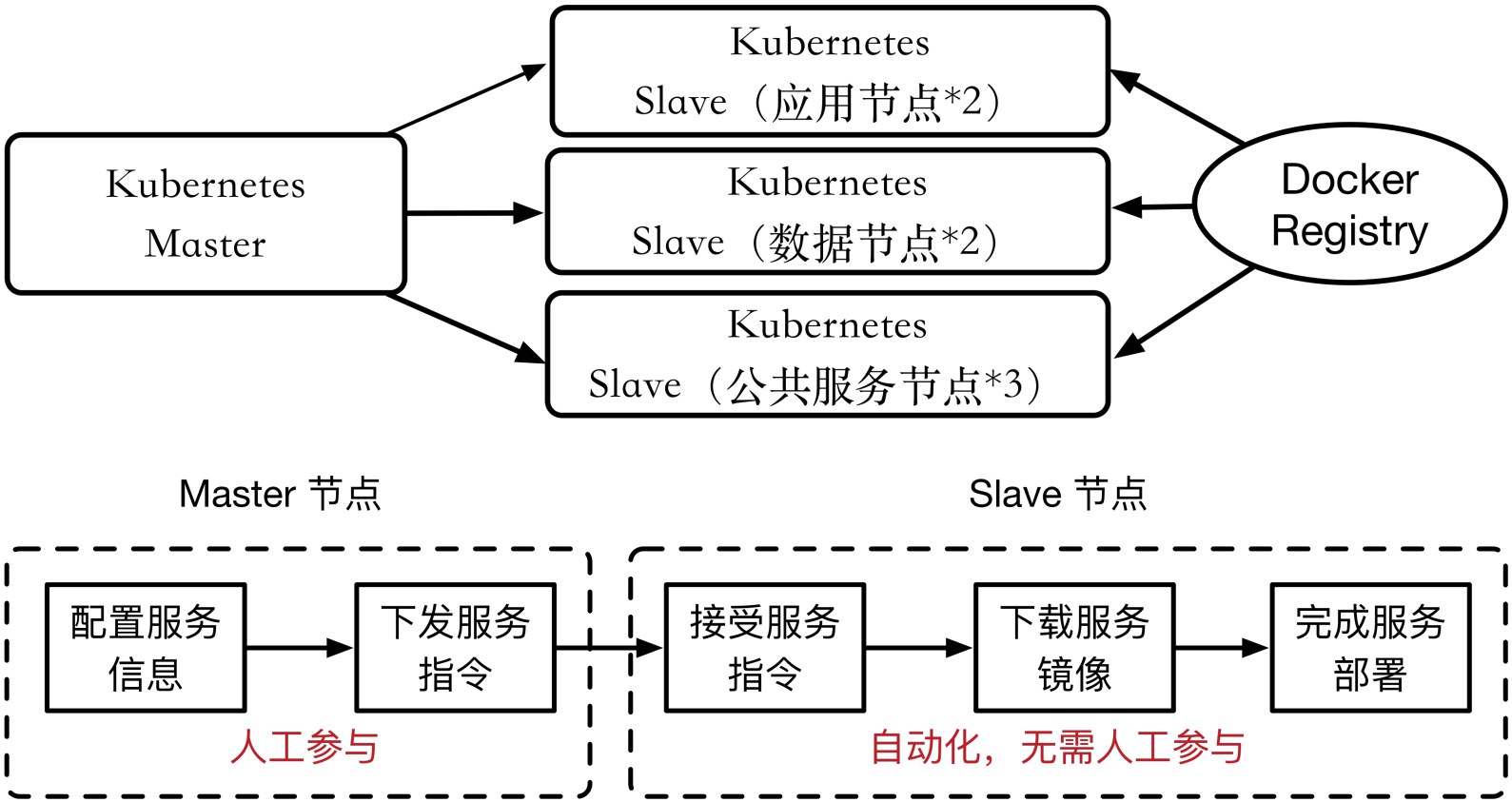
Tomcat 节约用户高并发的处理方式有 NIO 和 APR 两种方式:

- NIO 模式是一个基于缓冲区、并能提供非阻塞 I/O 操作的 Java API
- APR 模式是从操作系统级别解决异步 IO 问题，大幅度的提高服务器的处理和响应性能，是 Tomcat 运行高并发应用的首选模式。

请求次数	并发数量	APR 模式	NIO 模式
10000	10	6612.60	8255.48
50000	10	7480.96	8841.41
100000	10	6588.83	8355.85
10000	1000	6966.93	6155.21
50000	1000	9187.63	1957.35
100000	1000	7751.47	-

3. 容器化开发部署优化优化

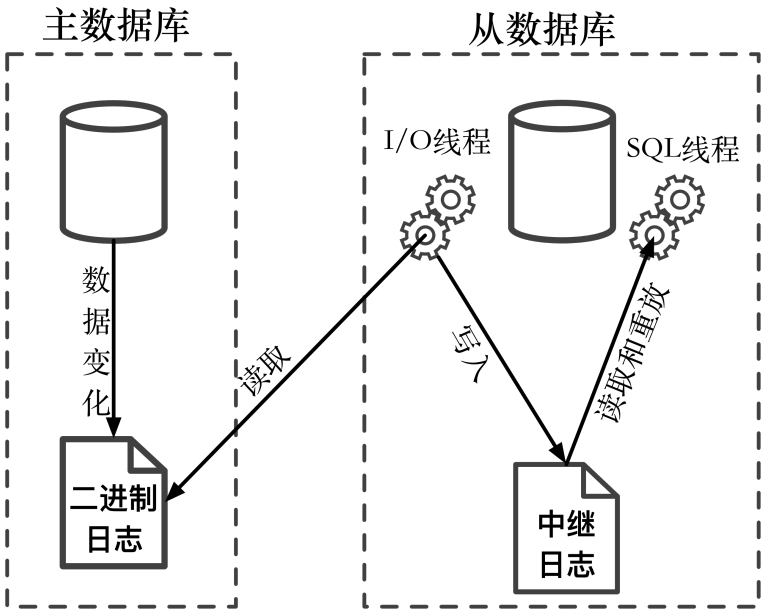
随着业务扩展，节点的快速部署和服务交付是性能优化过程中的重要工作。



一次部署三个服务的时间在 30 秒，传统部署方式平均时间在 15 分钟左右。

数据库复制结构开发

对于商业平台而已，保证数据的实时性和数据备份是应用优化的一个主要部分。




```
MySQL [(none)]> show slave status \G;
*************************** 1. row ***************************
               Slave_IO_State: Waiting for master to send event
                  Master_Host: 10.46.176.180
                  Master_User: backupmini
                  Master_Port: 3307
                Connect_Retry: 60
              Master_Log_File: db2.000001
          Read_Master_Log_Pos: 515277030
               Relay_Log_File: relay.000004
                Relay_Log_Pos: 1045549
        Relay_Master_Log_File: db2.000001
             Slave_IO_Running: Yes
            Slave_SQL_Running: Yes
```

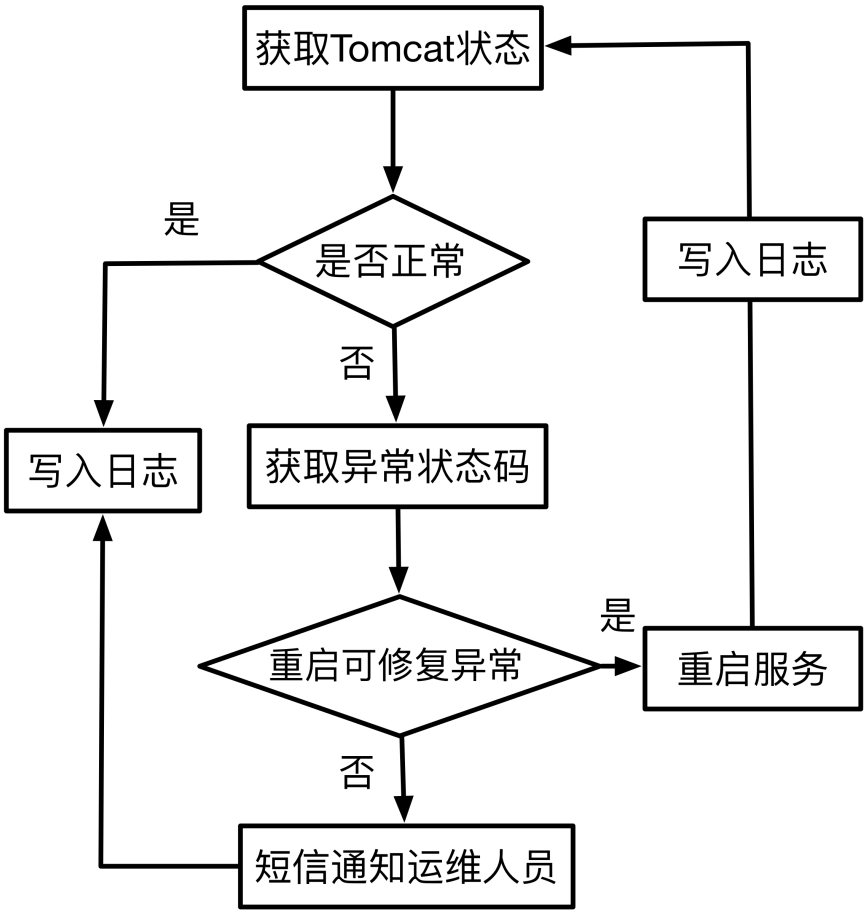


图: tomcat 健康监控

1. 应用监控

- 为保证应用能够持续正常的提供服务，需要对所有服务器下的应用服务进行监控

```
####2017-05-24 19:15:21####
--[OK]APP1 tomcat is RUNNING
--[OK]APP1 WEB is OK
--[OK]APP2 tomcat is RUNNING
--[OK]APP2 WEB is OK
####2017-05-24 19:15:52####
--[OK]APP1 tomcat is RUNNING
--[OK]APP1 WEB is OK
--[OK]APP2 tomcat is RUNNING
--[OK]APP2 WEB is OK
####2017-05-24 19:16:23####
--[OK]APP1 tomcat is RUNNING
--[OK]APP1 WEB is OK
--[OK]APP2 tomcat is RUNNING
--[OK]APP2 WEB is OK
```

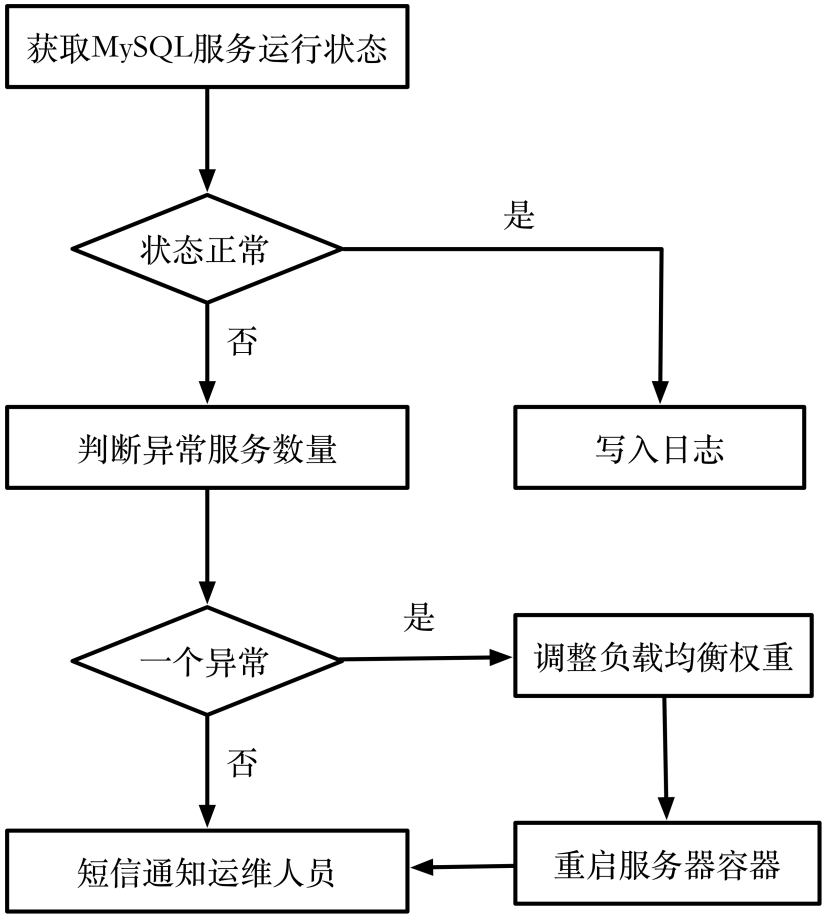


图: mysql 健康监控

2. 数据健康监控

- 在数据库层，首先要保证的是数据库能正常提供服务，所以需要对数据库的运行状态进行监控

```
2017-05-24 19:14:50     Mysql status Current OK!
  权重python脚本执行成功：权重无须修改，db1的权重为:100,
2017-05-24 19:15:21     Mysql status Current OK!
  权重python脚本执行成功：权重无须修改，db1的权重为:100,
2017-05-24 19:15:52     Mysql status Current OK!
  权重python脚本执行成功：权重无须修改，db1的权重为:100,
2017-05-24 19:16:23     Mysql status Current OK!
  权重python脚本执行成功：权重无须修改，db1的权重为:100,
2017-05-24 19:16:53     Mysql status Current OK!
  权重python脚本执行成功：权重无须修改，db1的权重为:100,
2017-05-24 19:17:24     Mysql status Current OK!
  权重python脚本执行成功：权重无须修改，db1的权重为:100,
```

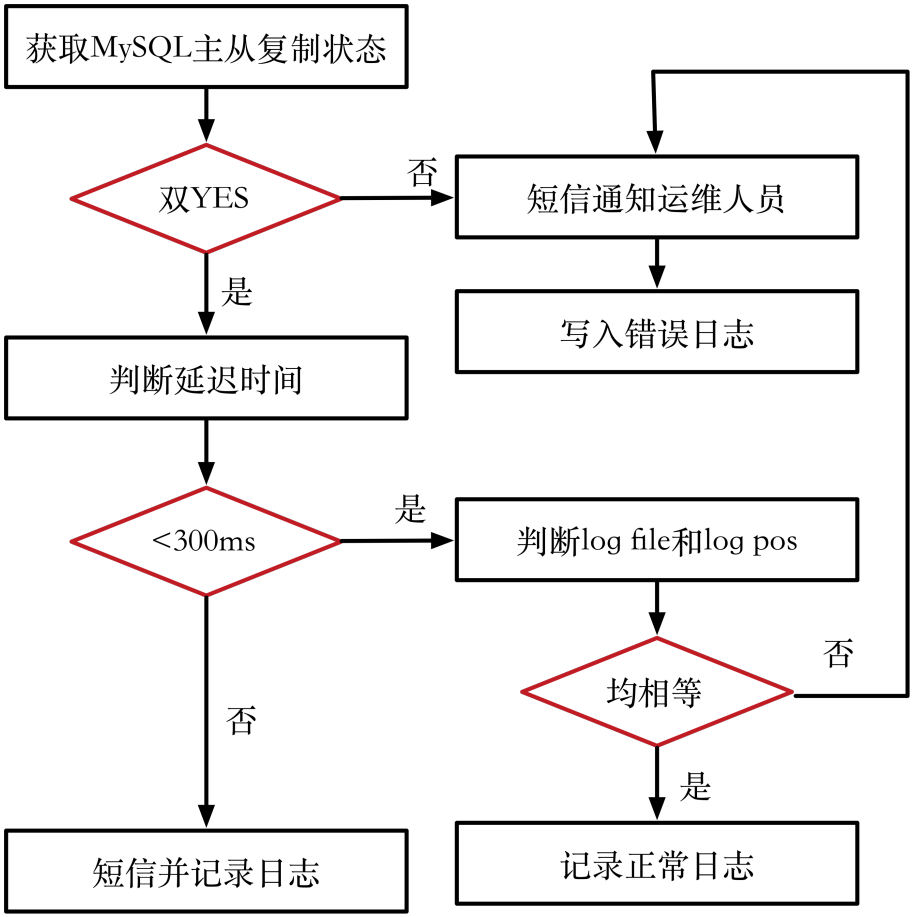


图: mysql 同步监控

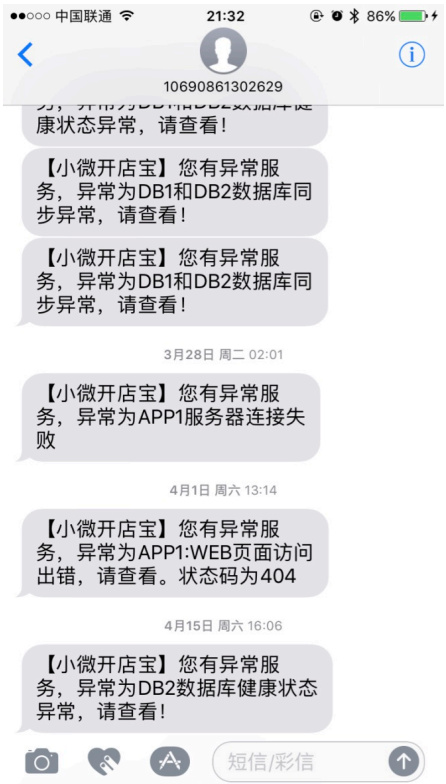
2. 数据同步监控

- 除了数据库的运行状态外，需要对数据库的主从同步状态进行监控，保证数据的完整性

```
2017-05-24 19:11:14     Mysql sync status Current OK!
2017-05-24 19:11:45     Mysql sync status Current OK!
2017-05-24 19:12:15     Mysql sync status Current OK!
2017-05-24 19:12:46     Mysql sync status Current OK!
2017-05-24 19:13:17     Mysql sync status Current OK!
2017-05-24 19:13:48     Mysql sync status Current OK!
2017-05-24 19:14:19     Mysql sync status Current OK!
2017-05-24 19:14:50     Mysql sync status Current OK!
2017-05-24 19:15:21     Mysql sync status Current OK!
2017-05-24 19:15:52     Mysql sync status Current OK!
2017-05-24 19:16:23     Mysql sync status Current OK!
2017-05-24 19:16:54     Mysql sync status Current OK!
2017-05-24 19:17:25     Mysql sync status Current OK!
2017-05-24 19:17:56     Mysql sync status Current OK!
2017-05-24 19:18:26     Mysql sync status Current OK!
```

3. 通知和备份方案设计

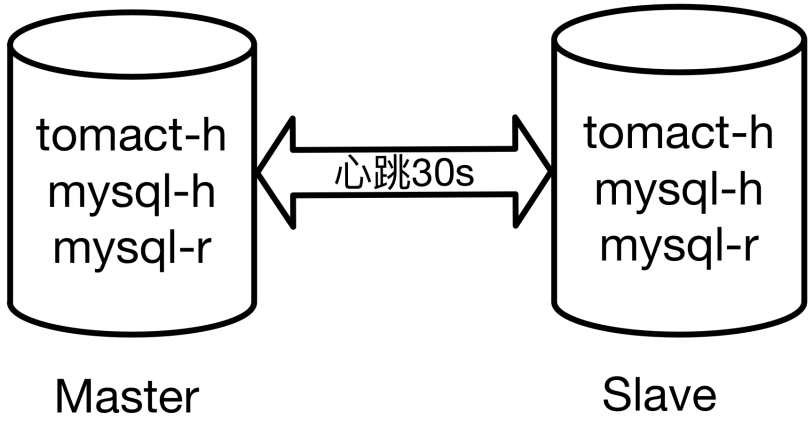
- 在第一时间通知运维人员服务器状态，开发短信通知脚本；
- 对异常进行细致的分析，准确定位异常原因，需要对所有的监控程序进行日志的保存和备份。



Vault名称	大小	创建时间	最后统计时间
tomcat_prod_app2_log	112.219MB	2016-11-17 09:16:24	2017-05-23 02:00:01
tomcat_prod_app1_log	52.983MB	2016-11-15 17:43:50	2017-05-23 02:00:01
tomcat_test_log	158.903MB	2016-11-17 09:16:18	2017-05-23 02:00:01

4. 心跳监听开发

为了保证监控服务的高可用，防止服务器崩溃导致监控停止的现象发生，需要在两台服务上部署监控服务并且配置心跳监听，保证同一时间只有一台机器上的监控服务运行




```
● heartbeat.service - Heartbeat High Availability Cluster Communication and Membership
   Loaded: loaded (/usr/lib/systemd/system/heartbeat.service; disabled; vendor preset: disabled)
   Active: active (running) since Tue 2017-03-28 08:57:21 CST; 1 months 27 days ago
 Main PID: 11344 (heartbeat)
   CGroup: /system.slice/heartbeat.service
           ├─11344 heartbeat: master control process
           ├─11350 heartbeat: FIFO reader
           ├─11351 heartbeat: write: bcast eth0
           ├─11352 heartbeat: read: bcast eth0
           ├─11353 heartbeat: write: ucast eth0
           ├─11354 heartbeat: read: ucast eth0
           ├─11381 heartbeat: master control process
           ├─11394 /bin/sh /usr/share/heartbeat/ResourceManager takegroup system_monito...
           ├─11430 /bin/bash /etc/ha.d/resource.d/system_monitor.sh start
           ├─32364 /bin/bash /etc/ha.d/resource.d/system_monitor.sh start
           └─32365 python /opt/sh/ssh/tomcat_fialover.py monitor 10.172.89.141
```

结论及展望

总结

通过应用、数据和服务器三个层级的优化策略研究，系统性能整体有了较大提升。

1. 高并发访问速度提升了 21%;
2. 应用部署时间节省 50%;
3. failover 故障恢复时间降低至 30s;
4. 服务器节点负载均保持在 85% 以下;

展望

虽然通过本论文的优化，WEB 应用的性能得到了较大的提升，但是依然有很大的改进空间：

1. 探索数据库读写分离方案，从数据库的输入输出两个方面进行优化，更好的保证数据的完整性;
2. 探索基于 kubernetes 的容器化集群部署方案以及监控方案;

谢谢！