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。

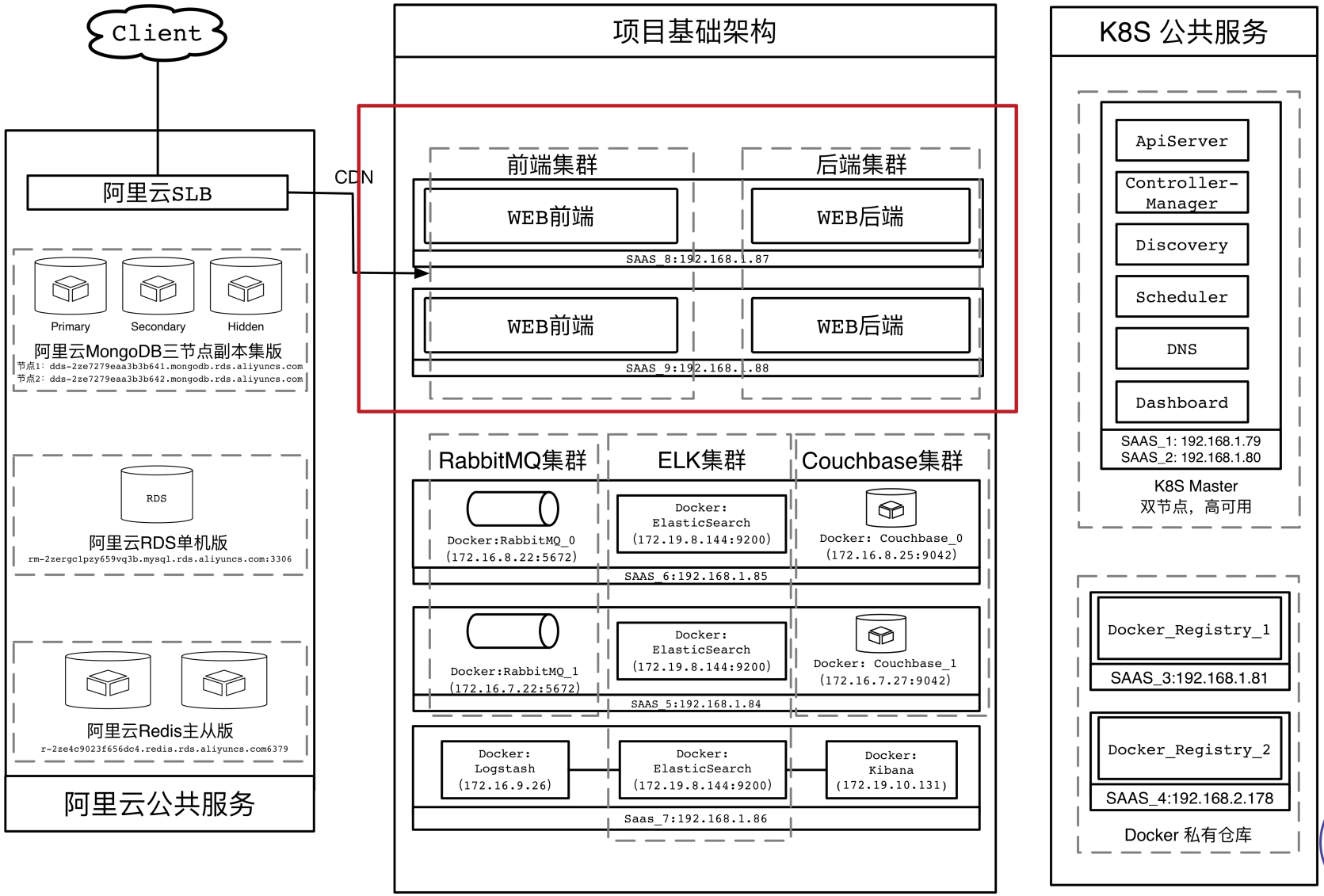
研究目的

- 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-2zergc1pzy659vq3b.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

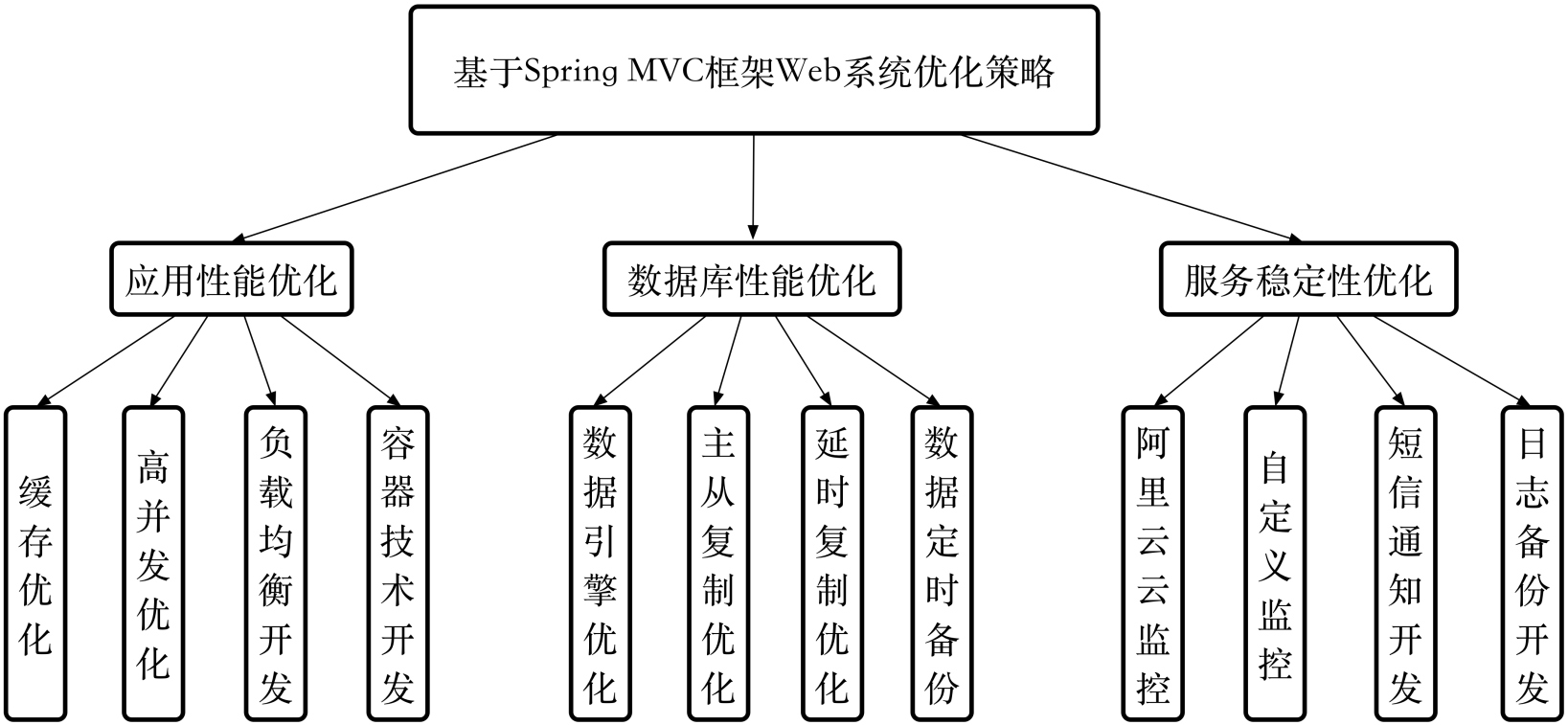
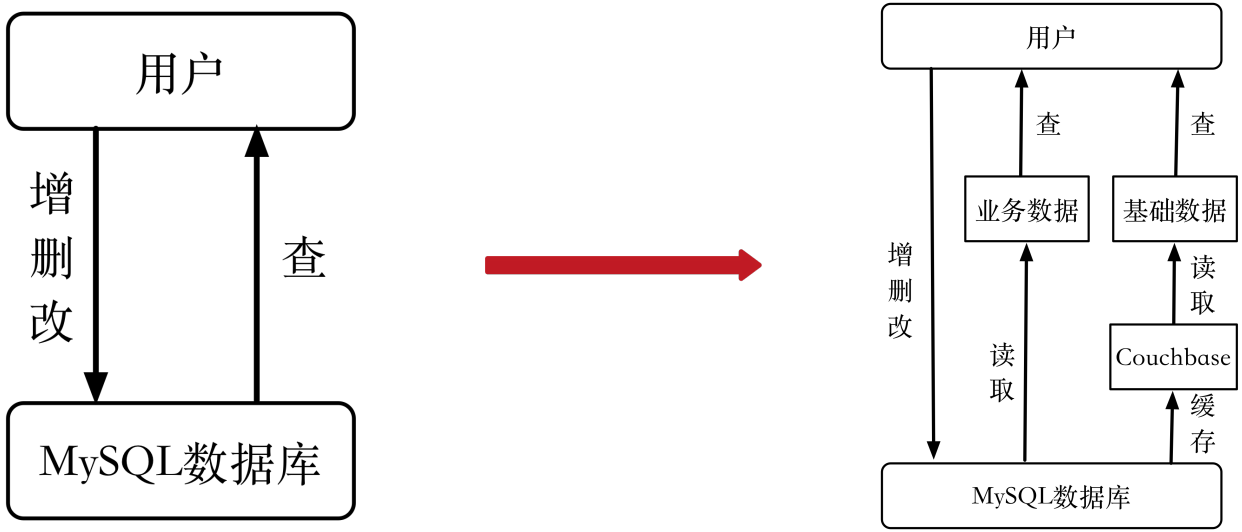


图: 平台功能

一、应用性能优化

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[1]



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

一、应用性能优化

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[7]：

- 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	-

一、应用性能优化

3. 容器化开发部署优化优化

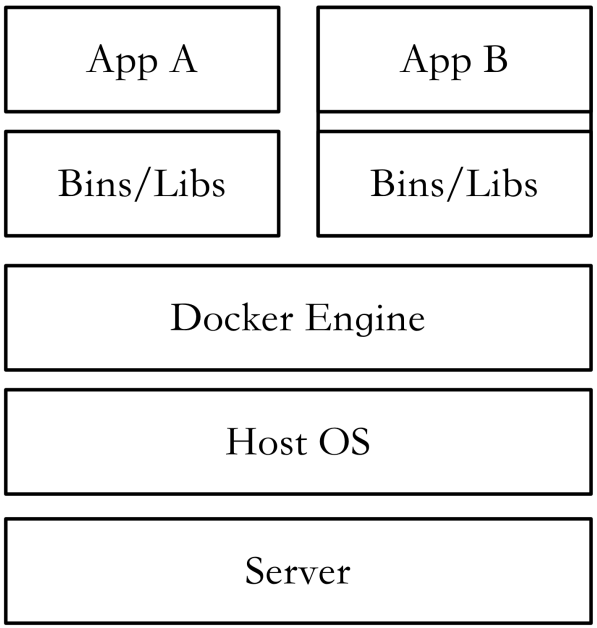
随着业务的扩展，目前一个 WEB 应用所使用的必要服务器数量已经扩展到了 7 个节点:

1. 应用服务节点 *2
2. 数据库节点 *2
3. ELK 日志服务及其他服务节点 *3

如何快速部署一个新的服务节点必然是运维人员在新增节点时必须要面对的问题[3]。

容器化是解决快速部署新节点和持续交付新服务的最优解决方案，他避免了每次安装软件的繁琐操作也节省了配置新的服务的各种配置成本。通过容器化，能够使开发运维的成本降低 40% 以上。

一、应用性能优化



3. 容器化开发部署优化优化

同传统虚拟机技术，容器技术的优势包括[2]：

1. 更高效的利用系统资源
2. 更快速的启动时间
3. 一致的运行环境
4. 更轻松的迁移、维护和扩展

表：同传统虚拟机技术对比

对比参数	容器技术	虚拟机技术
启动时间	秒级	分钟级
硬盘使用	一般为 MB	一般为 GB
应用性能	接近原生	弱于
系统支持量	单机支持上千个容器	一般几十个

二、数据库性能优化

1. InnoDB 引擎优化

虽然 MySQL 数据库支持的存储引擎有很多，但是常用的引擎主要是默认的 MyISAM 和使用最多的 InnoDB[6]。

表: MySQL 数据库存储引擎

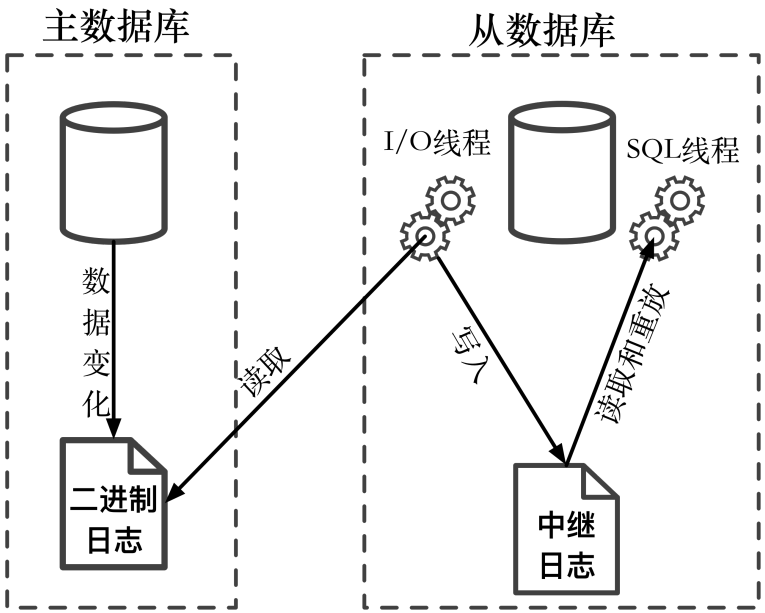
存储引擎	描述
InnoDB	支持事务和行级锁，是 Mysql 上唯一一个提供了外键约束的引擎
MyISAM	基于 ISAM 存储引擎，常用的引擎，但是不支持事务、行级锁、而且崩溃后不能保证完全恢复。

二、数据库性能优化

1. InnoDB 引擎优化

1. 内存利用方面，调整内存分配以及数据库缓冲池的数量等，保证 MySQL 在高 IO 负载时有非常稳定的吞吐[4]
2. IO 控制方面，通过控制线程的并发数和数据的压缩比保证系统正常运行的情况下数据库有更高的效率
3. 日志控制方面，因为日志文件的大小和实时写入量对于系统的性能来所有很大的影响，在默认情况下，对数据库进行高并发读写，吞吐量在 70req/sec 左右，优化参数后吞吐量可以提升到 300req/sec.

二、数据库性能优化



2. 数据库复制结构开发

开发数据库复制的目的是为了保证当一个节点损坏时，另一个数据库节点能够正常提供服务，当两个节点损坏时，可以通过延时复制的节点以及数据库 binlog 日志尽可能的恢复平台数据。复制的过程主要设计为三步[5].

三、服务器稳定性优化

1. 应用监控

应用监控主要是对 Tomcat 健康状态的监控，根据监控状态判断自动操作和手动操作，如果是通过重启服务能够解决的异常则会自动的远程重启服务。

表: Tocmat 状态

状态	描述
active	运行正常
failed	启动失败
unknown	未知错误

三、服务器稳定性优化

为了准确的判断服务异常错误信息，针对服务的不同状态规定不同的状态码，方便运维人员快速判断异常情况。

表: 自定义返回状态码

状态码	描述
100	表示 WEB 访问正常
101	表示 WEB 应用异常，需要查看日志
102	Tomcat 服务停止，需要重启
103	服务不存在，需要查看日志
104	服务器连接失败，需要查看是网络问题还是服务器异常
105	服务重启
106	其他异常
200	重启成功
201	重启失败

三、服务器稳定性优化

通过开发监控和重启的 python 脚本以及短信通知的 python 脚本实现具体操作，通过 shell 调用脚本实现日志记录和其他扩展。

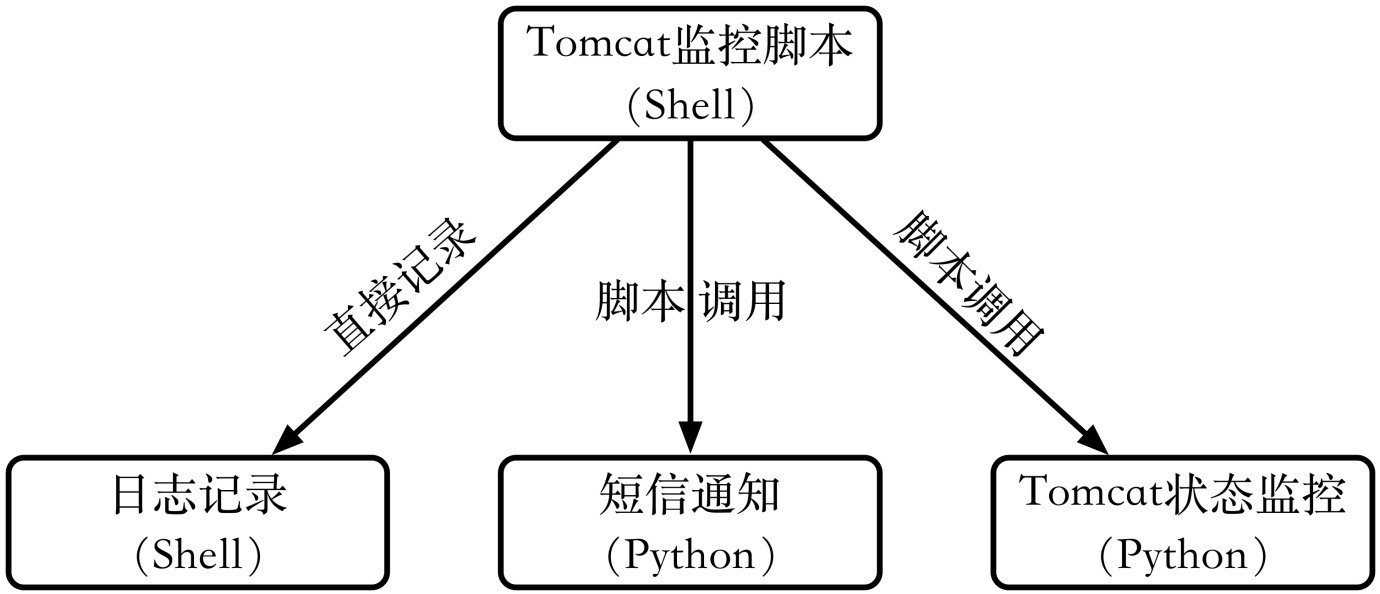


图: tomcat 监控框架

三、服务器稳定性优化

2. 数据健康监控

- 为了保证数据库的正常运行，通过开发 shell 脚本实现 mysql 的服务状态监控，当一个节点出现异常时，会通过 API 动态调整阿里云数据库负载均衡的权重，保证正常的节点能持续提供服务，同时对异常节点进行恢复处理。

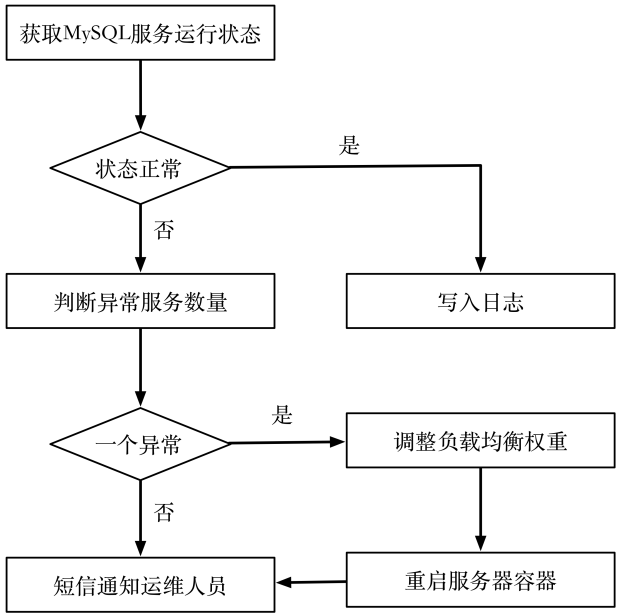


图: mysql 健康监控

三、服务器稳定性优化

2. 数据同步监控

- 因为两个数据库节点进行双主复制，当一个损坏时另外一个提供服务，这就要求两个数据库的内容保持一致，因此需要实时监控数据库的同步状态，保证两个库的数据一致。

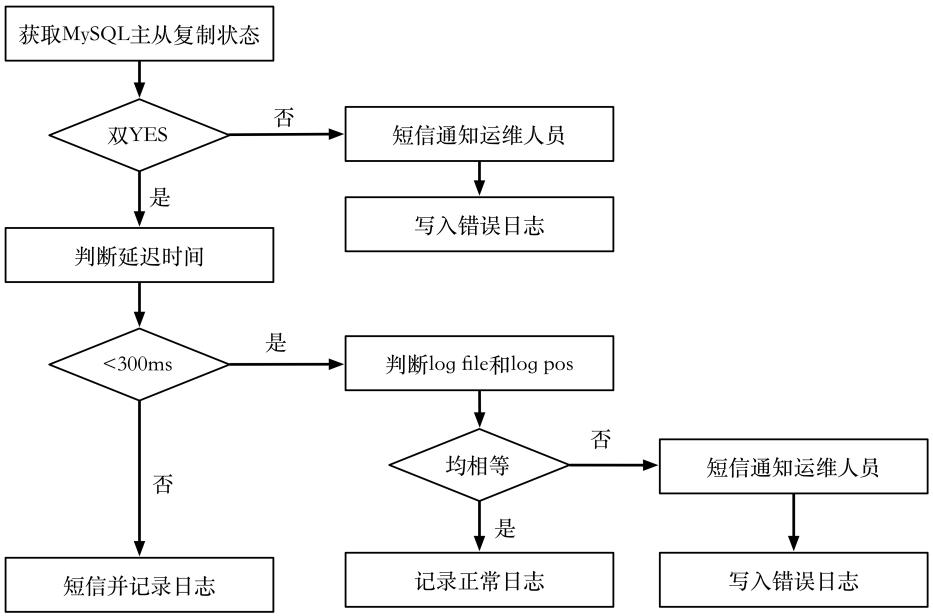


图: mysql 同步监控

三、服务器稳定性优化

3. 心跳监听开发

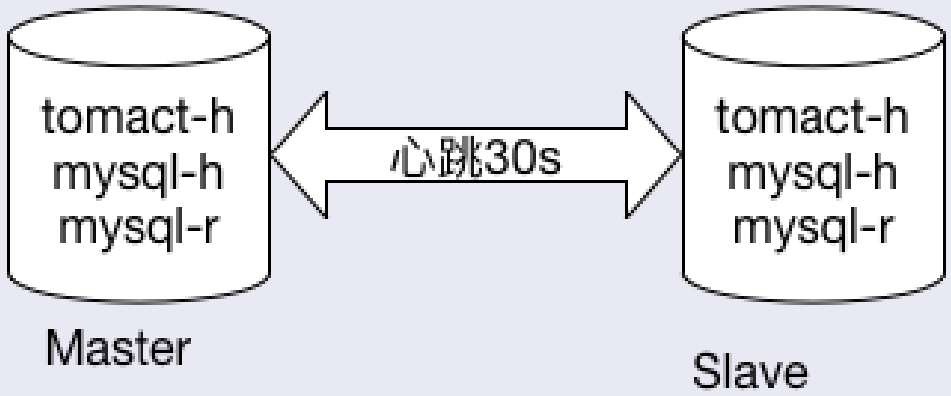


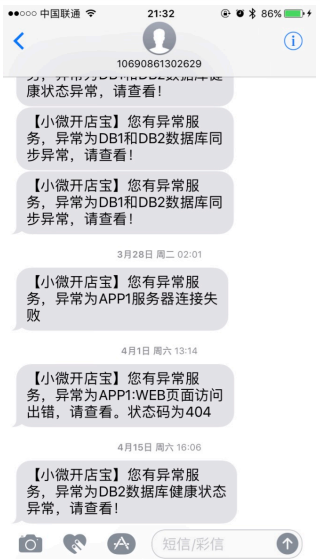
图: 心跳监听模型

Heartbeat 是一款开源的提供高可用 (Highly-Available) 服务的软件，通过 Heart- beat 可以将资源 (IP 及程序服务等资源) 从一台已经故障的计算机快速转移到另一台正常运转的机器上继续提供服务，称之为高可用服务[8]。

三、服务器稳定性优化

4. 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	操作
tomcat_prod_app2_log	112.219MB	2016-11-17 09:16:24	2017-05-23 02:00:01	查看 | 删除
tomcat_prod_app1_log	52.983MB	2016-11-15 17:43:50	2017-05-23 02:00:01	查看 | 删除
tomcat_test_log	158.903MB	2016-11-17 09:16:18	2017-05-23 02:00:01	查看 | 删除

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，目前应用服务较优化前相比，单个服务器的负载达到了比较好的程度，应用的访问速度提升了接近 20%, 新版的部署时间节省了 50%。

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，但是目前的 WEB 应用系统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;

References I

[1] Martin C Brown. Developing with Couchbase Server. O'Reilly, 2013.

[2] Ryan Chamberlain and Jennifer Schommer. Using docker to support reproducible research. DOI: http://dx. doi. org/10.6084/m9. figshare, 1101910, 2014.

[3] John Fink. Docker: a software as a service, operating system-level virtualization framework. Code4Lib Journal, 25, 2014.

[4] Peter Frühwirt, Marcus Huber, Martin Mulazzani, and Edgar R Weippl. Innodb database forensics. In Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, pages 1028–1036. IEEE, 2010.

[5] 秦金. 分布式 mysql 副本控制的研究与实现. Master's thesis, 北京邮电大学, 2013.

[6] 胡雯 and 李燕. Mysql 数据库存储引擎探析. 软件导刊, 11(12):129–131, 2012.

[7] 蒋文旭 and 辛阳. 大型高并发 web 应用系统架构分析与设计. PhD thesis, 北京: 北京邮电大学, 2012.

[8] 郭绪晶. 服务器集群系统高可用模块设计与实现. PhD thesis, 万方数据资源系统, 2012.