问题排查及系统优化

1.问题排查

1.1问题归类

- 逻辑缺陷：NPE、死循环、边界情况未覆盖
- 性能瓶颈：接口 RT 陡增、吞吐率上不去
- 内存异常：GC 卡顿、频繁 FGC、内存泄露、OOM
- 并发/分布式：存在竞争条件、时钟不同步
- 数据问题：出现脏数据、序列化失败
- 安全问题：DDoS 攻击、数据泄露
- 环境故障：宿主机宕机、网络不通、丢包
- 操作失误：配置推错、数据库误操作

1.2排查流程

1.2.1及时止损

- 新版发布：在发布新版本时，如果新版本有问题，首先考虑回滚
- 部分机器：部分机器出错时，考虑先把该部分机器的流量掐断
- 流量激增：考虑限流或增加节点数
- 下游依赖：降级预案
- 资源不足：已经运行稳定的应用突然中断，优先考虑内存资源不足，先重启，再排查是否内存泄漏（包括堆内和堆外）

1.2.2保留现场

- 日志：应用日志(kibana、adminserver、openshift内部日志)、中间件日志、内核日志
- Dump ：线程和堆dump
- 其它监控：自有监控系统、openshift自带监控

1.2.3定位问题

- 日志跟踪：根据日志提示排查问题
- 问题溯源：找到问题产生的根本原因
- 问题复现：在测试环境复现该问题，验证问题产生原因

- 新版本发布：是否最近有新版本发布
- 链路跟踪：zipkin

1.2.4解决问题

- 回归测试：避免产生新的问题
- 线上验证：持续观察，确保修复
- 问题归档：方便在出现类似的问题时，快速解决

1.3排查工具

1.3.1操作系统

- top、vmstat、iostat

1.3.2网络

- iftop、tcpdump、wireshark

1.3.3数据库

- awr(oracle)
- Promethus硬件性能查看

1.3.4应用

- jvm：jvisualvm、jprofiler、arthas、jstack、jmap
- kafka：http://10.1.18.254:8048/ke/account/signin
- zipkin：http://skywalking-ui-ca-opra-dev.apps.ocp.acca/　　根据环境调整namespace{ca-opra-dev}
- kibana：查看日志
- openshift：Promethus

2.系统优化

2.1性能

2.1.1性能指标

参考性能指标

- 吞吐率：单位时间内的处理能力
- 响应时间：一次请求从开始到结束的时间
- 伸缩性：通过水平扩展提高负载能力（包括应用及存储的扩展）

2.1.2性能分析

- 操作系统：cpu、内存、磁盘io
- 网络：流量
- 数据库：吞吐性能、慢sql
- 应用：jvm、kafka、skywalking、kibana

2.1.3优化原则

- 根据性能指标及场景进行优化
- 2/8原则

2.1.4优化方法

针对性能优化目标，优化方法可以考虑

- 集群
- 异步
- 批量
- 缓存
- 。。。

2.2稳定性

2.2.1指标

参考可用性指标

- 可用性

2.2.3优化原则

- 保证核心子系统系统可用性不低于99.90%
- 幂等性

2.2.4优化方法

- 集群
- 限流
- 熔断
- 降级
- 池化
- 超时
- 重试
- 。。。

2.3可维护性

2.3.1指标

参考可维护性指标

- 复杂度：业务边界清晰、架构职责单一、业务代码简洁等
- 可扩展：业务发生变更，代码易于扩展
- 可运维：方便水平扩展，完善的监控方案

2.3.2优化原则

- kiss：keep it simple stupid

2.3.3优化方法

- 界定边界：从业务层面划定好边界，合理拆分
- 定义规范：包括但不限于代码开发规范、第三方工具使用规范等
- 监控覆盖：完善的监控方案，方便发生问题时回溯