# 出了问题不要靠猜

LI Daobing <lidaobing@gmail.com>

七牛云存储

2013-10-16 上海

- 李道兵 <lidaobing@gmail.com>
- 先后任职于金山，盛大，GIGA循绿，七牛
- 现在在七牛做首席架构师
- 关注领域: 服务端安全, 质量保障, 运维自动化, DevOps
- Github: http://github.com/lidaobing
- SpeckerDeck: https://speakerdeck.com/lidaobing

# 目录

- 从一个 Bug 讲起
- 我看到的乱象
- 我推荐的解决方案

# 一个奇怪的 Bug

- 网盘的上传服务
  - Windows 下 IE, Chrome 浏览器下正常
  - Linux 下 Firefox, Chrome 浏览器下正常
  - Windows 下的 Firefox 不正常, 上传失败。

# 如何定位Bug

# 如何定位Bug

- 首先查看浏览器的控制台，观察是否有错误(都没有)

# 如何定位Bug

- 首先查看浏览器的控制台，观察是否有错误(都没有)
- 用 Wireshark 截获请求包，发现失败的案例服务端返回412(通常为200)

# 如何定位Bug

- 首先查看浏览器的控制台，观察是否有错误(都没有)
- 用 Wireshark 截获请求包，发现失败的案例服务端返回412(通常为200)
- 对比 HTTP Request，发现失败案例的 HTTP 头有“Content-length”，而其他情况下为“Content-Length”

# 如何定位Bug

- 首先查看浏览器的控制台，观察是否有错误(都没有)
- 用 Wireshark 截获请求包，发现失败的案例服务端返回412(通常为200)
- 对比 HTTP Request，发现失败案例的 HTTP 头有“Content-length”，而其他情况下为“Content-Length”
- 用 telnet 发送不同的请求来验证结论

# 如何定位Bug

# 如何定位Bug

- 确认原因是服务端对 Content-Length 的大小写敏感，与标准不符，造成了这个 Bug

# 如何定位Bug

- 确认原因是服务端对 Content-Length 的大小写敏感，与标准不符，造成了这个 Bug
- 通知服务端开发人员尽快修复

# 好Bug/坏Bug

# 好Bug/坏Bug

- 能再现的 Bug 就是好 Bug

# 好Bug/坏Bug

- 能再现的 Bug 就是好 Bug
- 如果不能再现，能拿到异常栈或者详细日志的也是好Bug

# 好Bug/坏Bug

- 能再现的 Bug 就是好 Bug
- 如果不能再现，能拿到异常栈或者详细日志的也是好Bug
- 对于坏 Bug, 他的最大用处就是督促你补充日志

# 另一个奇怪的Bug

- 从公司访问网站有12s延迟
- 直接 ping 响应很快
- 从其他网络来源访问没有问题
- 监控没有问题, 也没有客户来抱怨此事

# 先猜猜看

# 先猜猜看

- 公司路由出问题了？
  - 重启一下路由试试看？
  - 但为什么只有访问我们自己网站会出事？

# 先猜猜看

- 公司路由出问题了？
  - 重启一下路由试试看？
  - 但为什么只有访问我们自己网站会出事？
- 人品问题？
  - 做程序员还是唯物一点比较好

# 先猜猜看

- 公司路由出问题了？
  - 重启一下路由试试看？
  - 但为什么只有访问我们自己网站会出事？
- 人品问题？
  - 做程序员还是唯物一点比较好
- 不管，反正没人抱怨这事
  - 99% 的用户会遇到问题时不会报告

# 分析

# 分析

- tcpdump抓包
  - 客户端: 发了6个 SYN 包， 到最后一个才收到 SYN/ACK
  - 服务端: 确实收到了6个 SYN 包, 而且确实前面5个 SYN 包都没有回复
  - 从其他网络访问， 确实在首个 SYN 包之后返回

# 分析

- tcpdump抓包
  - 客户端: 发了6个 SYN 包， 到最后一个才收到 SYN/ACK
  - 服务端: 确实收到了6个 SYN 包, 而且确实前面5个 SYN 包都没有回复
  - 从其他网络访问， 确实在首个 SYN 包之后返回
- 结论: 问题跟路由/线路无关， 确实是服务端的问题

# 分析

# 分析

- 再重新分析 tcpdump 的记录， 发现前5个 SYN 包带了 timestamp, 第6个没带
  - 尝试关掉客户端TCP timestamp, 瞬间响应
  - 尝试关掉服务端TCP timestamp, 瞬间响应

# 分析

- 再重新分析 tcpdump 的记录， 发现前5个 SYN 包带了 timestamp, 第6个没带
  - 尝试关掉客户端TCP timestamp, 瞬间响应
  - 尝试关掉服务端TCP timestamp, 瞬间响应
- 结论: Bug 与 timestamp 相关， 但仍然无法解释为什么其他网络没问题

# 解決方案A

# 解决方案A

- 关掉服务端的TCP timestamp
  - TCP timestamp 会影响传输速度，但影响不大

# 解决方案A

- 关掉服务端的TCP timestamp
  - TCP timestamp 会影响传输速度，但影响不大
- 疑问仍然存在，为什么服务器不响应某些 timestamp 包, 有时候又能正常响应

# 分析

# 分析

- 看内核源码 (CTO 亲自操刀)
  - 入口机器有大量的反向代理，端口经常不够用，所以开启了TIME_WAIT 端口重用
  - 开启了 TIME_WAIT 端口重用后，服务端要求同一个IP的 SYN 包 timestamp 必须是顺序的
  - 办公网络是内网, 并且对我们官网访问很频繁，导致故障发生

# 分析

- 看内核源码 (CTO 亲自操刀)
  - 入口机器有大量的反向代理，端口经常不够用，所以开启了TIME_WAIT 端口重用
    - 开启了 TIME_WAIT 端口重用后，服务端要求同一个IP的 SYN 包 timestamp 必须是顺序的
    - 办公网络是内网, 并且对我们官网访问很频繁，导致故障发生
- 看来找到真实的原因了, 有没有更好地解决方案？

# 解决方案B

- nginx 高版本已经支持 keep-alive
- 升级 nginx, 启用 keep-alive, 降低端口占用
- 关闭端口重用，并加强端口数的监控和报警

# 我所看到的乱象

- 用寻找 workaround 代替解决问题

# 我所看到的乱象

- 用寻找 workaround 代替解决问题
  - 换个浏览器吧

# 我所看到的乱象

- 用寻找 workaround 代替解决问题
  - 换个浏览器吧
  - 重启, 清 cookie, 清 cache, 重新登录

# 我所看到的乱象

- 用寻找 workaround 代替解决问题
  - 换个浏览器吧
  - 重启, 清 cookie, 清 cache, 重新登录
  - 现场被破坏得干干净净

# 我所看到的乱象

- 用寻找 workaround 代替解决问题
  - 换个浏览器吧
  - 重启, 清 cookie, 清 cache, 重新登录
  - 现场被破坏得干干净净
  - 对于那些不易再现的 Bug, 就损失了一次修复的机会

# 我所看到的乱象

- 尝试用试错法解决所有问题

# 我所看到的乱象

- 尝试用试错法解决所有问题
- 升级依赖包, 升级插件, ...

# 我所看到的乱象

- 尝试用试错法解决所有问题
  - 升级依赖包, 升级插件, ...
  - 随便猜一个原因, 改两句代码, 然后重新测试

# 我所看到的乱象

- 缺少反省

# 我所看到的乱象

- 缺少反省
  - 寻找到 workaround 便认为问题解决了

# 我所看到的乱象

- 缺少反省
  - 寻找到 workaround 便认为问题解决了
  - 没有用测试固化 Bug, 容易产生回归Bug

# 我所看到的乱象

- 缺少反省
  - 寻找到 workaround 便认为问题解决了
  - 没有用测试固化 Bug, 容易产生回归Bug
  - 一个 Bug 可能在多处出现，没有尝试搜索其他有 Bug 的地方

# 我所看到的乱象

- 缺少反省
  - 寻找到 workaround 便认为问题解决了
  - 没有用测试固化 Bug, 容易产生回归Bug
  - 一个 Bug 可能在多处出现，没有尝试搜索其他有 Bug 的地方
  - 没有通过补充日志等手段来降低日后定位Bug的难度

# 我所看到的乱象

- 海恩法则: 每一起严重事故的背后，必然有29次轻微事故和300起未遂先兆以及1000起事故隐患

# 我们推荐的方法

- 出了问题不要靠猜
  - 浏览器
  - 服务端日志
  - 抓包工具

# 浏览器看什么？

# 浏览器看什么？

- 是否有 JS 错误?

# 浏览器看什么？

- 是否有 JS 错误?
- 网络请求是否发出？

# 浏览器看什么？

- 是否有 JS 错误?
- 网络请求是否发出？
- 发出的请求是否正确： URL, 方法, 参数, Accept, Cookie

# 浏览器看什么？

- 是否有 JS 错误?
- 网络请求是否发出？
- 发出的请求是否正确： URL, 方法, 参数, Accept, Cookie
- 期望的返回值是什么？

# 浏览器看什么？

- 是否有 JS 错误?
- 网络请求是否发出？
- 发出的请求是否正确： URL, 方法, 参数, Accept, Cookie
- 期望的返回值是什么？
- Request-Id

# Request-Id

- 对每次请求产生一个唯一的Id

# Request-Id

- 对每次请求产生一个唯一的Id
- 该 Id 以 HTTP Response Header 的方式发送到客户端

# Request-Id

- 对每次请求产生一个唯一的Id
- 该 Id 以 HTTP Response Header 的方式发送到客户端
- 可以在 nginx 层面实现或者在业务逻辑层面实现

# 服务端日志看什么？

- 四要素

# 服务端日志看什么？

- 四要素
  - 时间: 开始时间，总耗时

# 服务端日志看什么？

- 四要素
  - 时间: 开始时间，　总耗时
  - 谁: 用户Id, Session-Id, Request-Id

# 服务端日志看什么？

- 四要素
  - 时间: 开始时间，　总耗时
  - 谁: 用户Id, Session-Id, Request-Id
  - 做什么: URL, 方法，　XHR?, format, 参数(注意保护密码)

# 服务端日志看什么？

- 四要素
  - 时间: 开始时间，　总耗时
  - 谁: 用户Id, Session-Id, Request-Id
  - 做什么: URL, 方法，　XHR?, format, 参数(注意保护密码)
  - 结果是什么？

# 服务端日志看什么？

- 对其他服务的请求
  - 邮件，短信，其他服务...
  - 记录请求详情和耗时

但是经常没有足够的日志

# wireshark tcpdump

- 抓包工具
- wireshark: 有界面
- tcpdump: sudo tcpdump -n -s 4096 -w 1.log port 80

# 其他工具

- strace/dtruss: 系统调用跟踪工具
- lsof: 列出文件打开情况
- valgrind: 查内存泄露
- ltrace: 查询库调用

# 总结

- 我们推崇通过一种系统的方法来分析问题，寻找问题的根源
- 我们反对只靠试错法来解决问题
- 能再现的 Bug 是好 Bug，如果不能再现，也要拿到对应的网络请求和日志
- 如果这次解决不了 Bug，那么就改善你的日志，确保下次 Bug 出现的时候能解决他

# 我们团队

# Q&A