链滴

一次查找分子级 Bug 的经历，过程太酸爽了

作者：ShowMeBug
原文链接：https://ld246.com/article/1684117365925
来源网站：链滴

“Debugging is like trying to find a needle in a haystack, except the needle is also made of ha .”Debug调试就像是在大片的干草堆中找针一样，只不过针也是由干草制成的。

在软件开发的世界里，偶尔会出现一些非常隐蔽的 Bug，这时候工程师们像探险家一样，需要深入代的丛林，寻找隐藏在其中的“幽灵宝藏”。前段时间，我和我的团队也踏上了这样一段刺激、有趣的险之旅。

最近繁忙的工作告一段落，我总算轻松下来了，想趁这个机会，跟大家分享我们的这次“旅途”。

01 引子

我是 ShowMeBug 的 CEO 李亚飞，是一个古老的 Ruby 工程师。由于2019年招聘工程师的噩梦经，我立志打造一个真实模拟工作场景的 IDE，用来终结八股文、算法横行的技术招聘时代。

这个云上的 IDE 引擎，我称之为轻协同 IDE 引擎——因为它不是为了繁杂重度的工作场景准备的，**是适应于大部分人的习惯、能快速上手熟悉、加载速度快、能协同（面试用）、低延迟感，让用户感非常友好。**

多环境启动与切换

为了达成秒级启动环境的性能要求，我们设计了一套精巧的分布式文件系统架构，其核心是一个可以间复制大量小文件的写时复制（COW）技术。IO吞吐能达到几万人同时在线，性能绝对是它的一大势。

我们对此信心满满，然而没想到，很快就翻车了。

02 探险启程

2023年1月，北方已经白雪皑皑，而深圳却仍难以感受到冬天的寒意。

我和我的团队在几次打开文件树的某个文件时，会显得有点慢——当时没有人在意，按照常规思路，网速”背了这个锅。事后我们复盘才发现，这个看似微不足道的小问题，其实正是我们开始这次探险旅的起点。

1月底，南方的寒意缓缓侵入。这时候我们的轻协同 IDE 引擎已经开始陆续支持了 Vue2、Vue3、React、Django、Rails 等框架环境，一开始表现都很棒，加载和启动速度都很快。但是，跑了一段时间，**们开始察觉，线上环境就出现个别环境（ Rails 环境）启动要 20-30s 才能完成。**

虽然其他环境仍然保持了极快的加载和启动速度，但敏锐的第六感告诉我，不行，这一定有什么猫腻如果不立即行动，势必会对用户体验带来很不好的影响。于是，我开始安排团队排查眼前这个不起眼问题，我们的探险之旅正式开始。

03 初露希望

湿冷的冬季，夜已深，我和我们的团队依旧坐在电脑前苦苦探索，瑟瑟发抖。

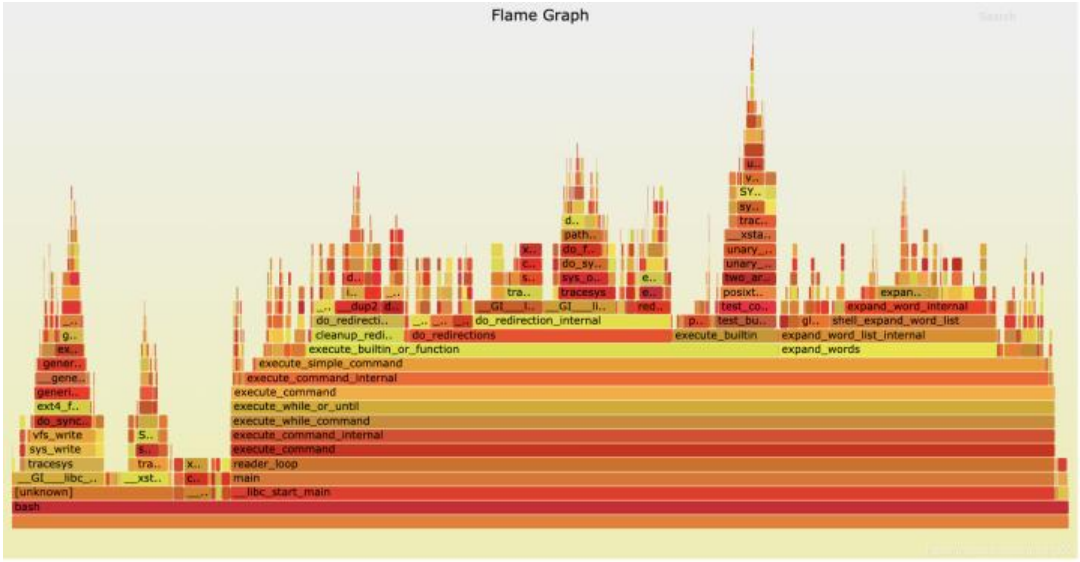
探险之旅的第一站，就是老大难的问题：定位Bug。目前只有某一个环境启动很慢，其他的环境都表不错。大家想了很多办法都没有想明白为什么，甚至怀疑这个环境的模板是不是有问题——但把代码在本地启动，最多就2秒。

哎，太诡异了。我们在这里卡了至少一周时间，不断追踪代码，分析日志文件，尝试各种方案，都没弄清楚一个正常的程序启动为什么会慢。我们一度陷入了疲惫和焦虑的情绪中。

“Debug 是种信仰，只有坚信自己能找到 Bug，才有可能找到 Bug。”

软件开发界一直有一个低级Bug定律：所有诡异的问题都来自一个低级原因。在这“山重水复疑无路之际，我们决定重新审视我们的探险路径：为什么只有 Rails 更慢，其他并不慢？会不会只是一个非微小的原因而导致？

这时候，恰好有一个架构师朋友来访，向我们建议，可以用 perf 火焰图分析看看Rails的启动过程。



perf火焰图实例

当我们用 perf来分析时，惊讶地发现：原来 Rails 的启动要加载更多的文件！紧接着，我们又重用了一个文件读写监控的工具：fatrace，通过它，我们看到 Rails 每次启动需要读写至少 5000 个文，但其他框架并不需要。

这才让我们突然意识到，会不会是文件系统读写速度不及预期，导致了启动变慢。

04 Bug现身

为了搞清楚是不是文件系统读写速度的问题，我们急需一个测试IO抖动的脚本。团队初步估算一下，好这个脚本需要一些时间。

这时候，夜已深了，研发同学已经陆续下班了。我猛然想到，不是还有ChatGPT吗？让它帮忙写一个试。

Bash Copy Caption

```bash
#!/bin/bash

while true
do
    start_time=$(date +%s.%N)
    time cat /root/lyftest/tmp/test_file.txt
    end_time=$(date +%s.%N)
    io_time=$(echo "$end_time - $start_time" | bc)
    #echo "test..."
    if (( $(echo "$io_time > 0.010" | bc -l) )); then # 10ms
        echo "IO操作时间为 $io_time, 时间戳为 $(date '+%Y-%m-%d %H:%M:%S.%N')" >> test_io.txt
    fi
    sleep 0.5
done
```

IO抖动脚本

Cool，几乎不需要改动就能用，把代码扔在服务器开跑，一测，果然发现问题：**每一次文件读写都需 10-20 ms 才能完成。**实际上，一个优秀的磁盘IO读写时延应该在亚毫级，但这里至少慢了50倍。

Bingo，如同“幽灵宝藏”一般的分子级Bug逐渐显现，**问题的根因已经确认：过慢的磁盘IO读写引了一系列操作变慢，进而导致启动时间变得非常慢。**

更庆幸的是，它还让我们发现了偶尔打开文件树变慢的根本原因，这也是整个系统并发能力下降的罪祸首。

05 迷雾追因

看到这里，大家可能会问，这套分布式文件系统莫非一直这么慢，你们为什么在之前没有发现？

非也，早在项目开始的时候，这里的时延是比较良好的，大家没有特别注意这个 IOPS 性能指标，直我们后面才留意到，系统运行超过一个月时，IO 读写时延很容易就进入到卡顿的状态，表现就是文系统所在主机 CPU 忽高忽低，重启就会临时恢复。

此时，探险之旅还没结束。毕竟，这个“幽灵宝藏”周围依旧笼罩着一层迷雾。

我们继续用 fatrace（监控谁在读写哪个IO） 监控线上各个候选人答题目录的 IO 读写情况，好家伙我们发现了一个意外的情况：**几乎每一秒都有一次全量的文件 stats 操作（这是一个检测文件是否有性变化的 IO 操作）！**

也就是说，比如有 1000 个候选人正在各自的 IDE 中编码，每个候选人平均有 300 个文件，就会出每秒 30 万的 IO 操作数！

我们赶紧去查资料，根据研究数据显示，一个普通的 SSD 盘的 IOPS 最高也就到 2-3 万 。于是，我重新测试了自己分布式文件系统的 IOPS 能力，结果发现也是 2-3 万 。

那这肯定远远达不到我们理想中的能力级别。

这时，问题更加明确：某种未知的原因导致了大量的 IOPS 的需求，引发了 IO 读写时延变长，慢了约几十倍。

06 接近尾声

我和我的团队继续深究下去，问题已经变得非常明确了：

原来，早在去年 12月，我们上线一个监听文件增删的变化来通知各端刷新的功能。

最开始我们采用事件监听（fswatch event），因为跨了主机，所以存在1-2s的延迟。研发同学将其为轮询实现的方案，进而引发了每秒扫描目录的 stats 行为。

当在百人以下访问时，IOPS 没有破万，还足够应对。但一旦访问量上千，便会引发 IO 变慢，进而致系统出现各种异常：间歇导致某些关键接口 QPS 变低，进而引发系统抖动。

随着“幽灵宝藏”显露真身，这次分子级Bug的探险之旅也已经接近尾声。团队大呼：这过程实在太爽了！

07 技术无止境

每一个程序员在成长路上，都需要与Bug作充足的对抗，要么你勇于探索，深入代码的丛林，快速定，挖到越来越丰富的“宝藏”，然后尽情汲取到顶级的知识，最终成为高手；或者被它打趴下，花费量时间都找不到问题的根源，成为芸芸众生中的一人。

当然，程序员的世界中，不单单是Debug。

当我毕业5年之后，开始意识到**技术的真正价值是解决真正的社会问题。**前文中我提到：我发现技术聘真是一个极其痛苦的事：特别花面试官的时间，却又无法有效分析出候选人的技术能力，所以我创了ShowMeBug，**用模拟实战的编程环境，解决科学评估人才的难题。**

这个轻协同 IDE 技术从零开发，支持协同文件树、完全自定义的文件编辑器、协同的控制台(Consol）与终端（Shell），甚至直接支持 Ctrl+P 的文件树搜索，不仅易于使用，又强大有力。

但是这还不够。要知道，追求技术精进是我们技术人的毕生追求。技术越厉害，意味着我们的参数性越好。对于这个轻协同IDE，我们追求三个零：**零配置、零启动、零延迟。**其中，零启动就是本文所求的极限：**以最快的速度启动环境和切换环境。**

因此，探险之旅结束后，我们进一步改进了此文件系统，设定 raid 的多磁盘冗余，采用高性能SSD，时重新制定了新磁盘架构参数，优化相关代码，最终大幅提升了分布式文件系统的稳定性与并发能力。

截止本文结尾，ShowMeBug启动环境的平均速度为 1.3 秒，切换环境速度进入到亚秒级，仅需要 70ms。目前在全球范围的技术能力评估赛道（TSA）中，具备1-2年的领先性。

08 后记

正当我打算结束本文时，我们内部的产品吐槽群信息闪烁，点开一看：嚯，我们又发现了新Bug。

立夏已至，我们的探险之旅又即将开始。