Docker 碎碎念

2015-08-14 by Kelvin Peng

别问我为啥还不更新伊朗的攻略(´◔ ‸◔')

Docker 1.8 也发布了，我们呢在经过漫长的开发和不能说的五千字之后， Eru 平台目前承当了公司呃，5%左右的流量，目前来看姿势良好（我才不会说 Redis Cluster 总请求数已经破千亿这个事），来总开源贡献的 Open-Falcon 也被我们调教得自带 API 光环的酸爽。肥六的 Flat UI 已经写得超神，DNS CronJob 玩得飞起，我就只能没事有事优化下各组件消耗和 review 代码了。

但是……关键就是这个但是，每次整合 Docker 的时候各种想吐槽，当然如果是玩 Docker 公司自己一套工具就算了，自研轮子真是一把辛酸一把泪。

Stats

1.5 之前，Container 的 Stats 需要读取 Cgroups 计数器来获得，Google 家的 cAdvisor 原本采用的 libcontainer 中的一个 Public Struct，我们按照这个思路在 Agent 也实现了这么一堆东西。结果没过几个月，libcontainer 从一堆重复结构体重构成了依然依然重复结构体，好吧你们重构能力不行或者说历史原因都能忍，公开接口换什么换！

所以 Google 家一怒之下自己搞了个 vendor，我特么硬是看了一周代码硬吃了这波高维攻击……

然后脑残的 1.5 来了，自带 Stats 接口让人眼前一亮，但是！卧槽这是哪个脑残设计的 1s 返回一次 Metrics 的逻辑啊。关键是从 API 来说只能起一个 Goroutine 一直 1s 处理一次数据直到 Container 挂掉。中途退出，主动退出，Step 控制什么都

没有，真是服了这群 doge 了，这 API 从工程上来说就是不合格的。复杂的不说，1 秒一次的 Metrics 在高压力多容器环境下，对 CPU 网卡的消耗肉眼都能从监控上看到，我都不知道谁同意的这种无厘头 PR。

哦对了，关于这个 API 可用性，一直到 1.7 版才修复，中间 1.6 一如既往的不可用 PR#10766。我们现在用 1.7 + Agent 控制 Step，用 consistent hash 保证对 open-falcon transfer 的可写性，这才满足了现有体系的需求。

日志

1.6 之前，docker 的日志处理简直脑残到极点，json-file 本地不做切割一个文件写到死还不能换目录，线上业务分分钟教硬盘做人有木有。当年洪荒年代各种方案用来搞这个幺蛾子，比如容器内开 syslog 做 forward，对不起，您的容器违反了单一容器单一进程原则打破了树桩结构，卒。再比如暴露本机 syslog 进容器，对不起单点负载过高拖垮宿主机，卒。最后我们采用了 attach forward stdout 和 stderr 的方式解决了这个问题。

然后 1.6 发布了，终于有了原生的 syslog 支持，然而并没有什么卵用。成熟的平台里面用 attach 可以做到按照不同来路均衡日志后端，还可以加入一系列的 meta info 来增强日志可读性，关键是还不会破坏容器本身自身的隔离性，而且在 agent 端想怎么 buffer 就怎么 buffer，想起几个连接写就起几个连接写，想用 tcp 就 tcp 想用 udp 就能 udp，恩我们就是这么做的。所以原生的 syslog 在这个版本里面怎么看都觉得只是把以前用 syslog 的 trick 勉强的做了一个 api 罢了。

到昨天发布 1.8 后（我才懒得吐槽一天后就发布了 1.8.1），官方终于把 log 进行的大量的扩展，Fluentd，GELF，syslog-facility 等，从生产的角度来说应该是够用了，但是对于已有的老司机而言，和 attach 方案并没太多的优势，所以还是洗洗睡吧。顶多就是新平台能用这些 feature 更容易收集日志罢了。

网络

不得不说 libnetwork 这个坑，其实从 docker 出生起就埋下了。1.7 甚至 1.8 之前，我个人对这个方向上面的看法是完全悲观的，也就是说，没法用。第三方 overlay

方案怎么看都比 docker 目前集成和实现的方案更加的灵活可靠。1.7 说 libnetwork 跟着发布，然后呢，毛都没有。甚至到了 1.8，libnetwork 的 docs 还在 experimental 底下。同时期 plugin 的另外一个方向 volume 已经进 master 发布了，高下立断。

从我自己的经验来讲，网络应该是 docker 平台所面临最大的问题或者说是最高的门槛，光是纯粹的系统工程师并没有什么卵用，还得各个部分如 SA 的深度参与才能比较好的解决。从目前我们采取的 Agent 来控制 container 的方案来看，就现在 plugin networking 即便能用，我们也不会用，可扩展性和定制性实在太差，对于混合 SDN 的支持基本是没有的（好吧虽然我们也在做 macvlan <-> calico 的事情）。当然了，也不能否认 docker 在这方面的努力，对于新平台而言，尝（diao）试（jin）一（keng）下（li）还是不错的嘛……

至于 1.9 行不行，只能到时候再看了，不过看着 libnetwork 到 docker 1.9 才发布 0.5，个人依然谨慎悲观。

存储

对于有状态的服务而言，容器内存储永远是悬在头顶上的利剑，说不定哪天就连带数据一起丢了。我觉得 docker 官方推崇的什么微服务啦，拆分啊，其实大多数情况下是不现实的，又不是每个公司都是刚创业，历史负担摆在这里了。如果 mount 到外部就还是会出现管理上的问题，当然打破了隔离安全也无从谈起了。我们这边比较依赖应用方自己做迁移逻辑，比如 redis 的容器在下线钱要做迁移 slot 什么的，但长远来看，对业务方要求还是蛮高的。

1.8 之后 volume plugin 终于转正，我还是比较看好的。当然了，对于 volume 的选择，又是掉进另一个坑的节奏，不过短期来看，在有状态的 container 集群管理方面，这个 plugin 的提供了一个新的途径。另外对比网络而言，这货的靠谱程度高多了……

资源

到目前这个地步，CPU_SHARE CPU_SET 已经完全足够了，CPU_QUOTA 我个

人觉得对 Public 的编排平台而言可能会有效果，但对内而言，意义不是很大。Memory 这块 docker 除非实现自身的 Soft limit，否则目前 hard limit 并没有什么卵用。我们用 Agent 采取旁路监控的方式半自动的实现了 Soft limit，虽然在时效性上不如腾讯改内核的实现，但也足够使用了。

目前头疼的可能是网络和磁盘 IO 的限制。前者虽然我们做得到对单一容器的流量计数，但是做不到限流，怎么和路由结合做 QoS 还是说继续现在的人工 QoS 的模式，在未来一切皆有可能。至于后者，docker 官方你先赶紧实现对 Block IO Stats 的支持，等有数据了再去决策吧。

曾经我对我组的人说过 1.7 升级完之后半年之后不会动线上服务器，可是尼玛一个月后的 1.7.1 是什么鬼。目前新的 1.8 对我的吸引力还是太小，反正 golang 的 docker-client 实现质量也不高，等等 1.9 再看了。现有的平台已经完全足够支撑起我们的服务，所以给肥六加薪了！想要来做国内自主的调度编排平台的小伙伴们，简历不要停啊！

#docker #work