

MEILI INC.

从Linux系统内核层面 来解决实际问题的实战经验

by 亚方

meili™ MEILI INC.

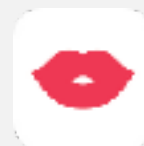


背景介绍

蘑菇街业务快速增长，随之而来服务的稳定性挑战也越来越大，对于复杂稳定性问题的定位，除了从开发人员的视角来分析，还需要能够从系统、内核的视角来分析，这对一些疑难问题、关键问题的解决来说是很有帮助，甚至是关键的。在定位和解决完这些疑难问题后，我们还构建了一套问题分析和定位平台，帮助业务方实现 devops 自助化。

meili Meili Inc.

MEILI INC.

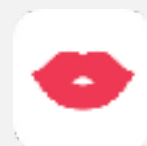


主要内容

- 常见问题的不同分析思路
- 涉及到的内核基本知识
 - Linux系统为CentOS-6/CentOS-7
 - 对于的内核分别是kernel-2.6.32/kernel-3.10.0

meili Meili Inc.

MEILI INC.

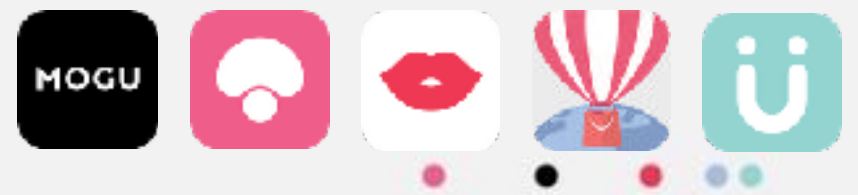


稳定性问题概述

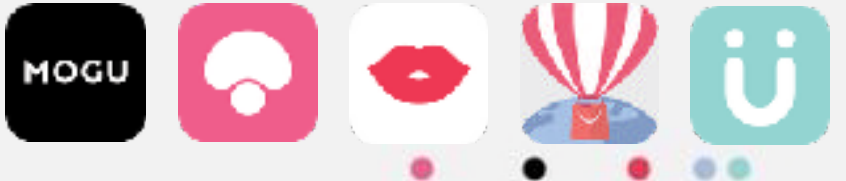
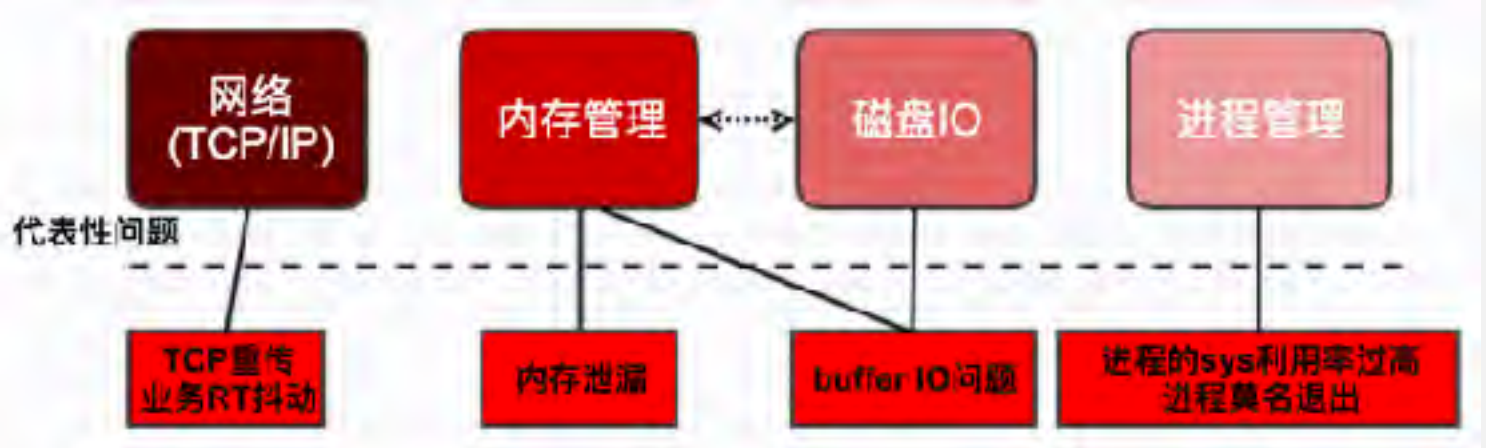


说明:

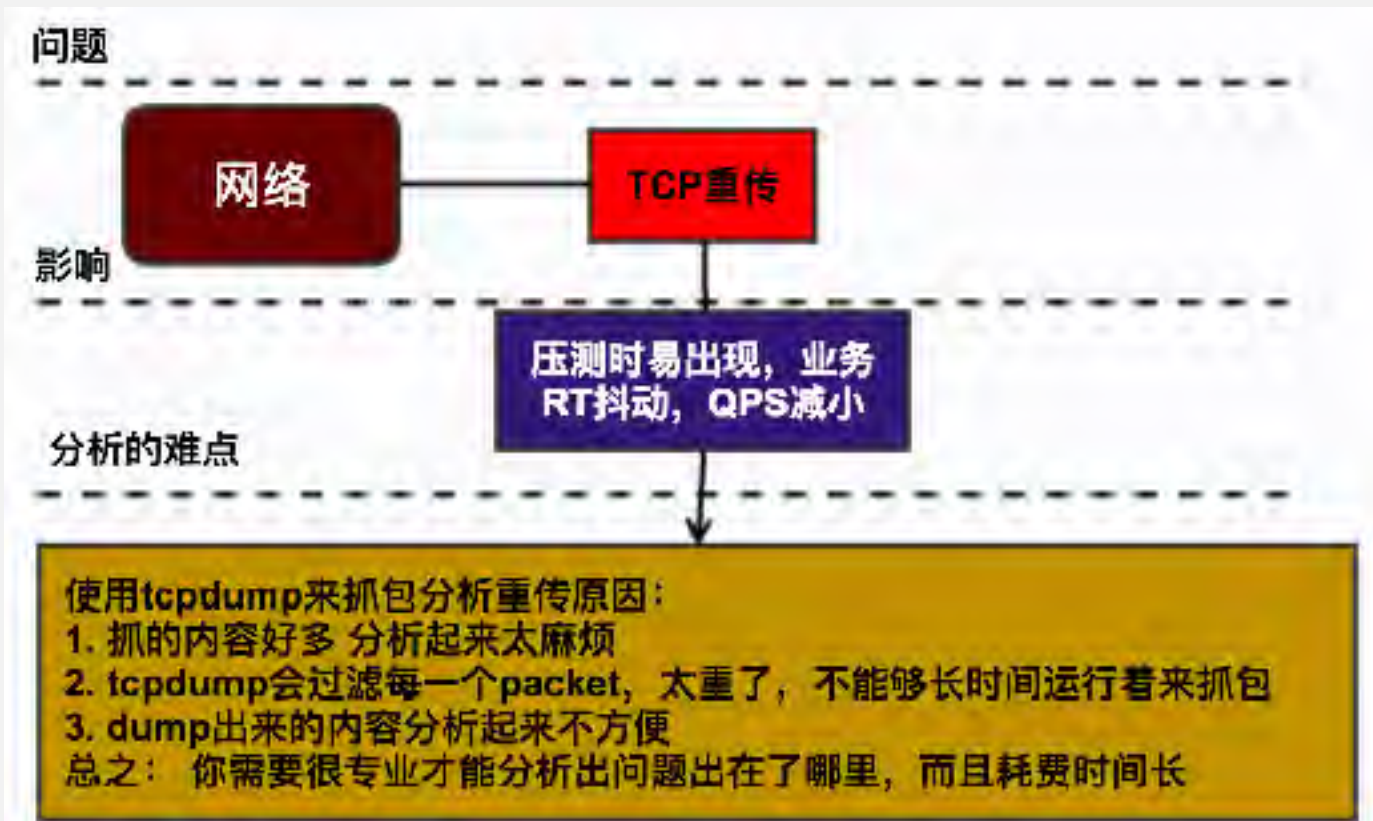
- 1. 一个问题往往牵涉到操作系统的很多模块，我们以问题的主要解决点来做划分
- 2. 磁盘IO的问题往往伴随着内存管理的问题



代表性问题

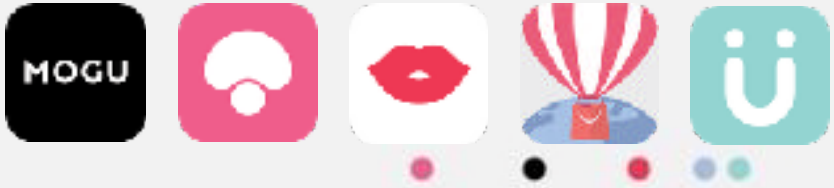


网络部分代表性问题分析：TCP重传问题

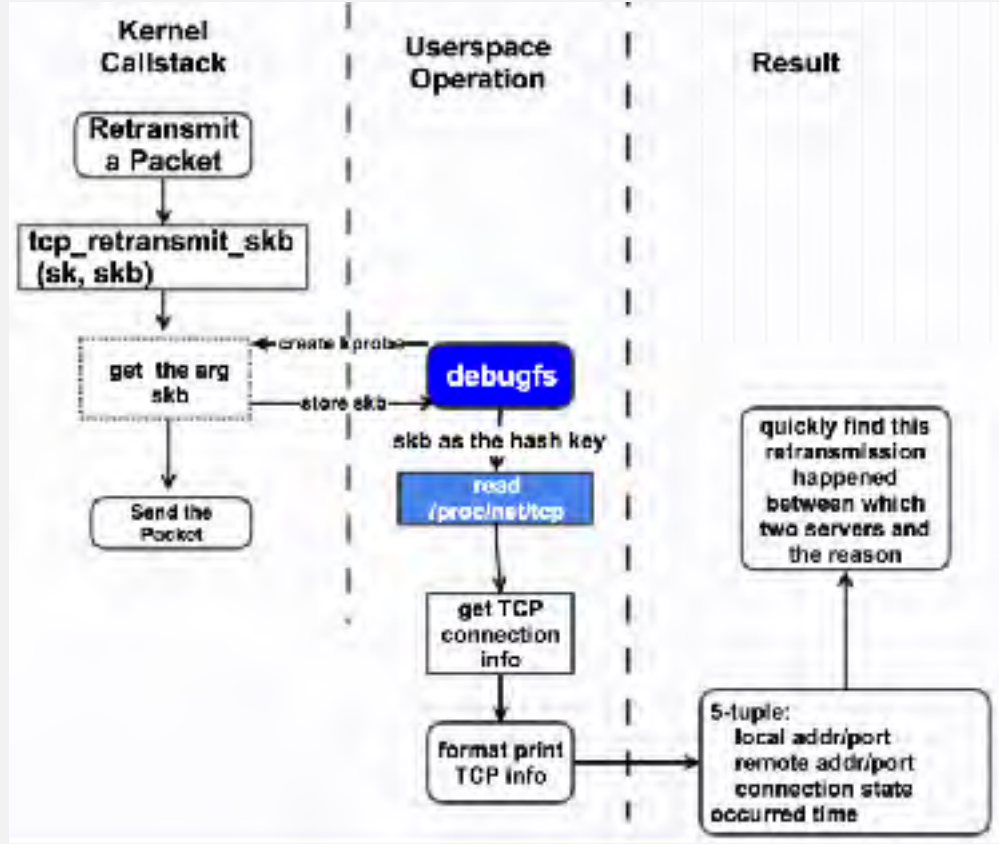


小改进： 如何高效的分析tcp重传问题

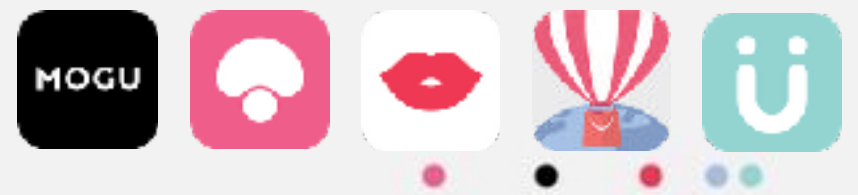
| | <i>tcpdump</i> | <i>tcpretrans</i> | 改进 <i>tcpretrans</i> |
|----|--|--|---|
| 特征 | <p>优势:</p> <ul style="list-style-type: none">1. 抓取的信息全面 <p>劣势:</p> <ul style="list-style-type: none">1. 内容多,难分析2. 重量级, 比较消耗系统资源3. <i>dump</i>出来的内容分析不方便, 需要借助 <i>wireshark</i>分析 | <p>优势:</p> <ul style="list-style-type: none">1. 轻量级, 对系统性消耗小, 可以一直运行着2. 信息简介明了, 便于快速分析 <p>劣势:</p> <ul style="list-style-type: none">1. 需要掌握内核知识来使用它2. 需要一些额外的繁琐步骤来设置3. 对于发生重传后连接存在时间小于1s的显示不了 | <p>优势:</p> <ul style="list-style-type: none">1. 易用, 不需要内核专业知识, 只需要一条命令就可以2. 对于重传后存在时间小于1s的连接也可以显示出来 |



改进后的tcpretrans原理



MEILI INC.



实时打印内容示例

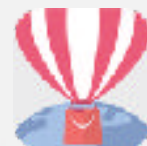
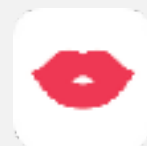
问题分析实例：

```
$ tcpdumps
TIME      PID      LADDR:LPORT      -- RADDR:RPORT      STATE
11:36:35  0        10.11.11.13:48696  R> 10.19.11.26:9092  SYN_SENT
11:36:38  0        10.11.11.13:42320  R> 10.17.107.18:9092  SYN_SENT
11:36:39  0        10.11.11.13:44844  R> 10.19.11.27:9092  SYN_SENT
11:36:40  0        10.11.11.13:42320  R> 10.17.107.18:9092  SYN_SENT
11:36:40  0        10.11.11.13:44844  R> 10.19.11.27:9092  SYN_SENT
11:36:41  0        10.11.11.13:48696  R> 10.19.11.26:9092  SYN_SENT
11:36:44  0        10.11.11.13:42320  R> 10.17.107.18:9092  SYN_SENT
11:36:49  0        10.11.11.13:48696  R> 10.19.11.26:9092  SYN_SENT
11:36:52  0        10.11.11.13:44844  R> 10.19.11.27:9092  SYN_SENT
11:36:55  0        10.11.11.13:59785  R> 10.19.11.29:9092  SYN_SENT
11:36:57  0        10.11.11.13:59785  R> 10.19.11.29:9092  SYN_SENT
11:37:01  0        10.11.11.13:59785  R> 10.19.11.29:9092  SYN_SENT
```

可以看到是9092这个端口存在问题，然后分析该端口对应的服务即可



MEILI INC.

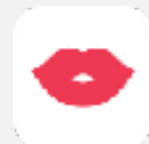


tcpretrans上线后的效果

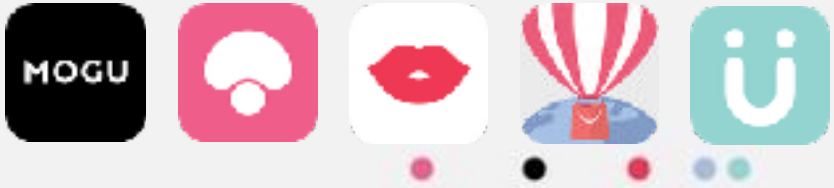
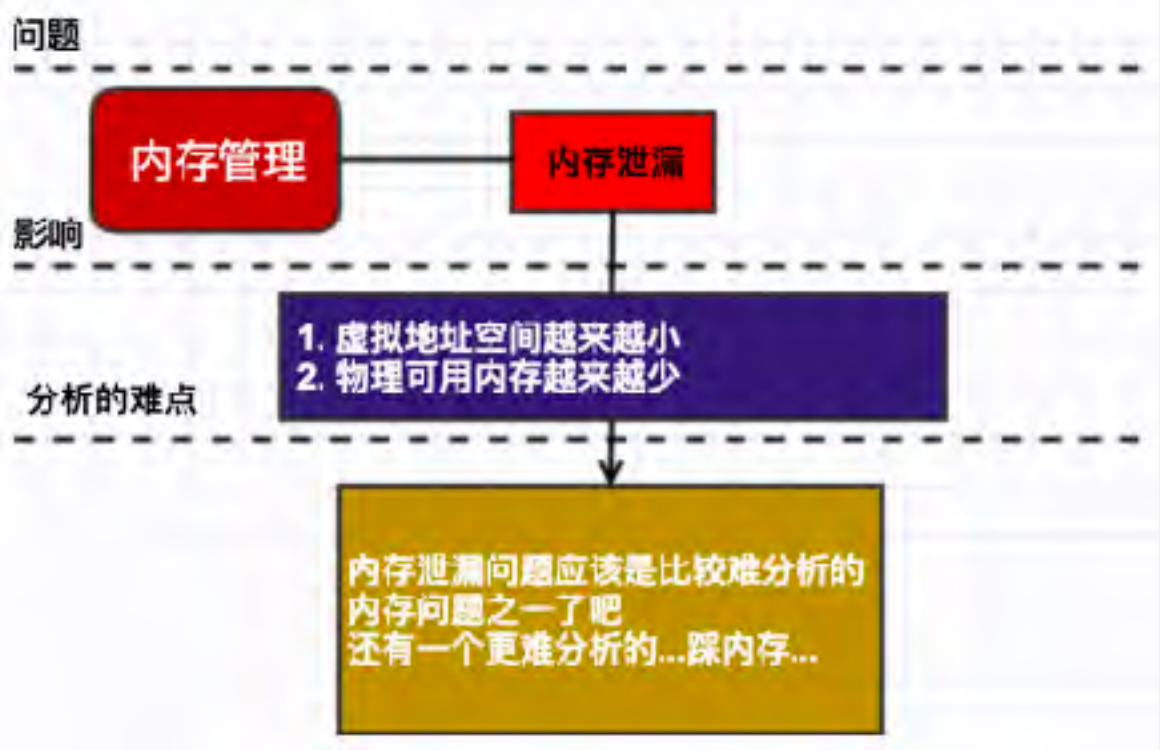
- 现在已经被运维以及业务人员大量使用，在遇到TCP重传时，他们首先想到的是用这个工具来抓取下信息。
- 在遇到TCP重传变高时，快速的帮助我们指明正确的前进方向
- 运用它帮助很多业务方分析了问题

meili Meili Inc.

MEILI INC.

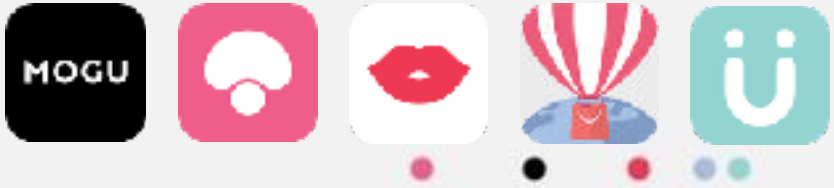


内存管理代表性问题：内存泄漏



内存泄漏问题的分析方法

| 分析方法 | 分析工具以及适用场景 |
|--------------------|---|
| 在编译阶段去分析 | Google的sanitizers是其中的代表: <i>Asan, Tsan, Msan</i> 内存泄漏问题的本质方法都是做在编译及以前的 |
| 在运行时去做分析 | <i>Valgrind</i> 是其中的代表 他的一个要求是需要使用 <i>valgrind</i> 来重新执行应用程序去做跟踪分析 |
| 在运行时不中断应用程序的执行去做分析 | 这是一个难题,业界并没有现成的成熟方案 从技术手段而言,这也是可以做到的,只是是否有价值 |

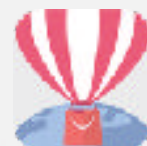
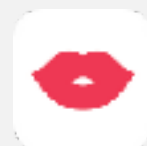


内存泄漏问题有危害的根源在哪里？

- 长运行daemon
任务起来之后除非去停止他否则就一直运行着
- 处理request式
只运行较短时间，内存泄漏也不会有啥问题
- 防病易 治病难！

meili Meili Inc.

MEILI INC.

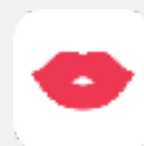


案例：在运行时不中断任务去分析内存泄漏

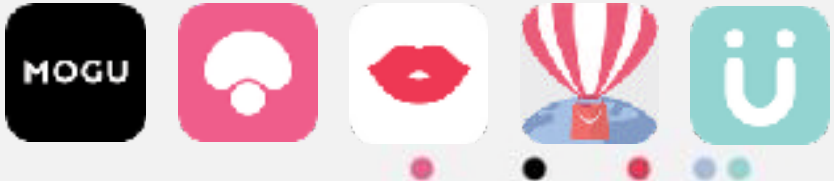
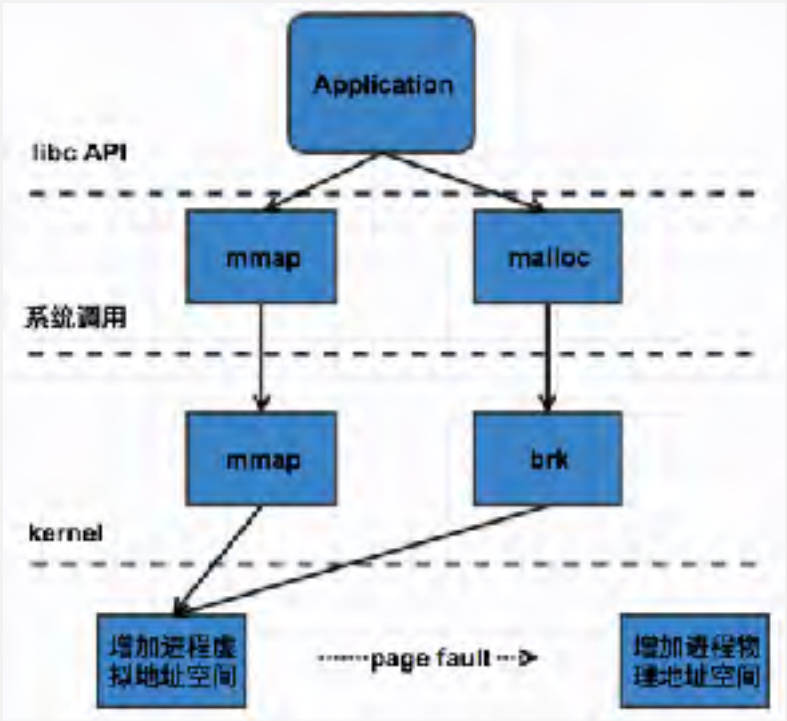
- 线上几个机器出现了虚拟地址空间快被耗尽的情况
- 这种问题很难复现，需要抓取线上机器的现场信息来分析

meili Meili Inc.

MEILI INC.

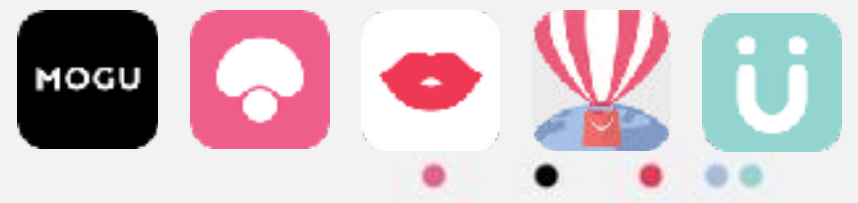
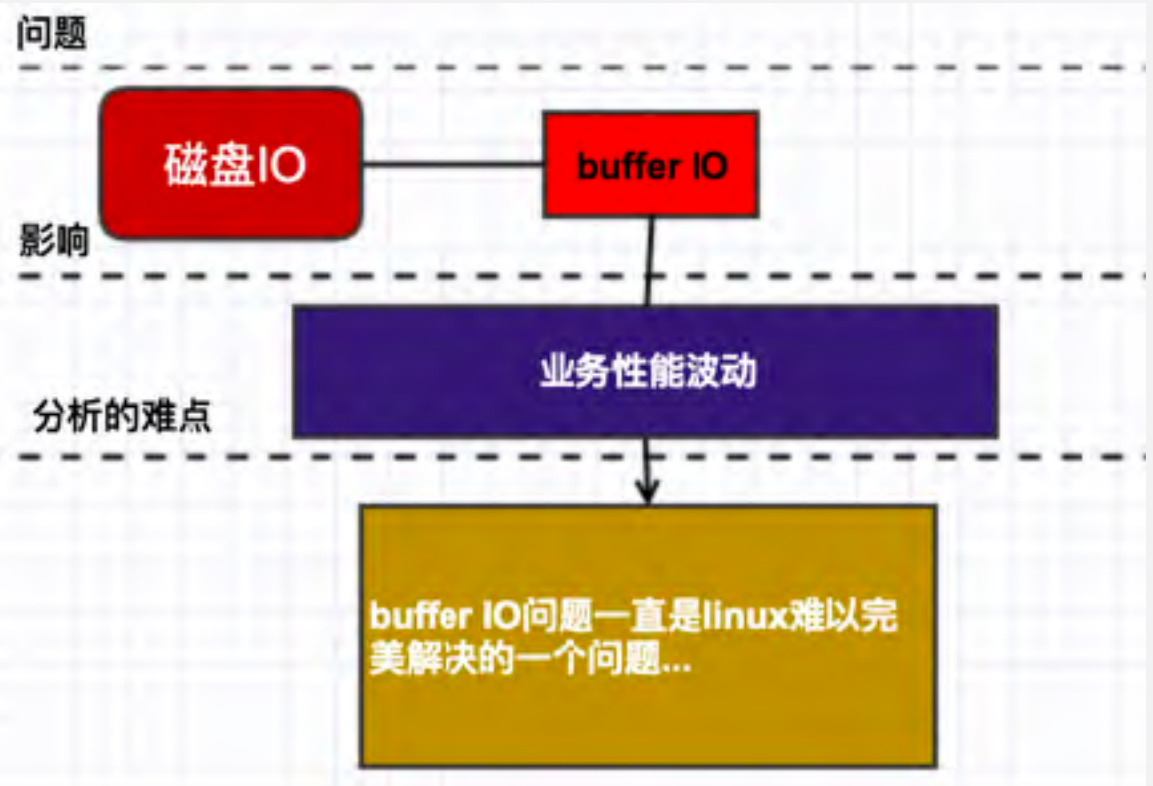


应用申请内存

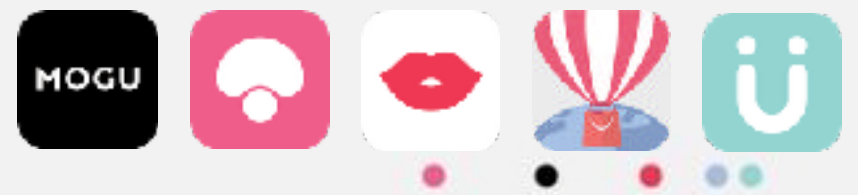
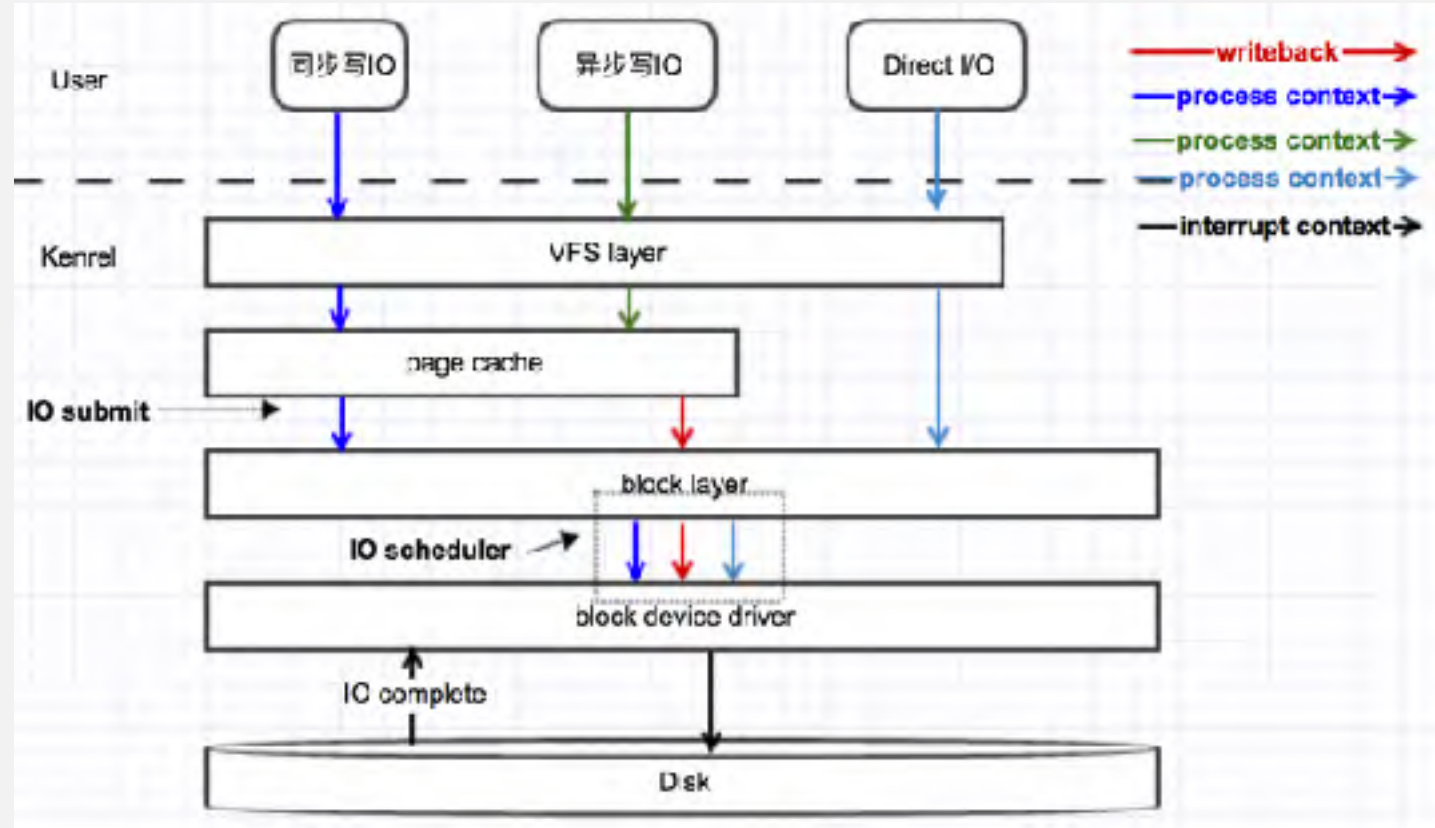




磁盘IO代表性问题：buffer IO



buffer IO与writeback

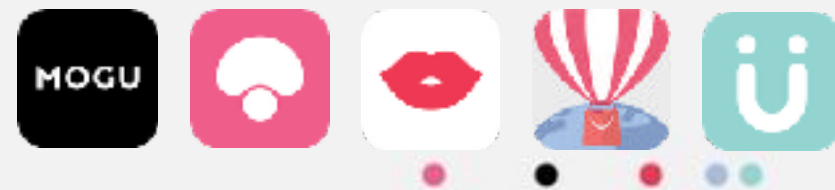


writeback的失控

- 容易引起的问题
 - 内存页回收时的dirty pages
 - 这是CentOS-6上很容易出现的问题,所以基于CentOS-6来讲
 - 异步写IO的突发导致IO行为不稳定
 - 这个问题一直困扰着Linux开发人员吧

meili Meili Inc.

MEILI INC.

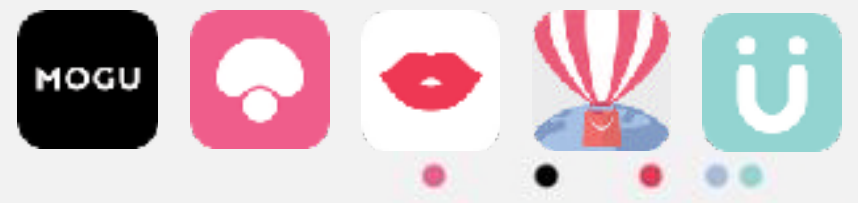
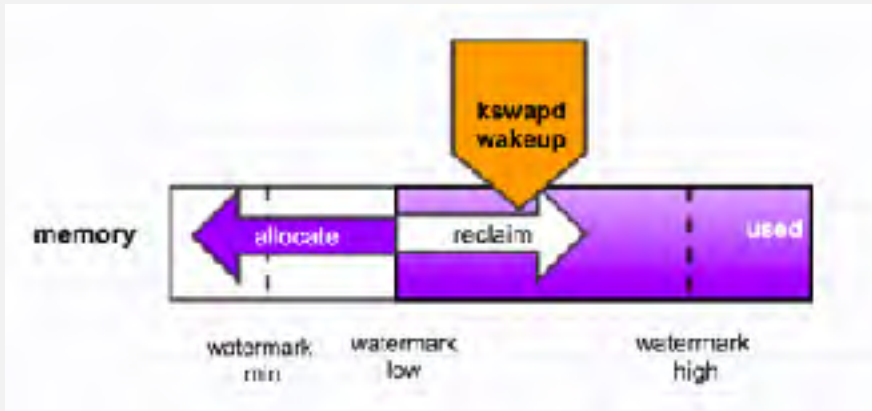
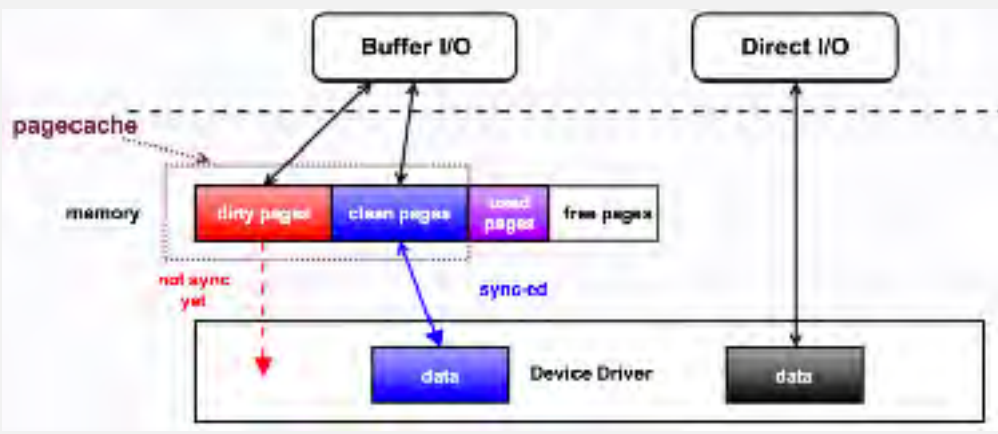


内存页回收时的dirty pages

- 关于dirty page

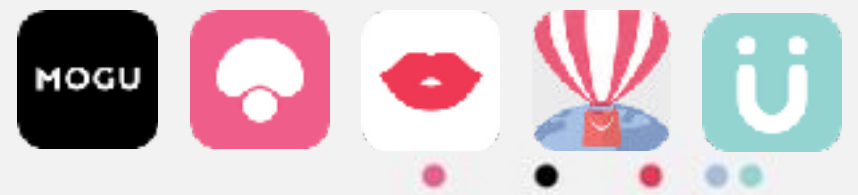
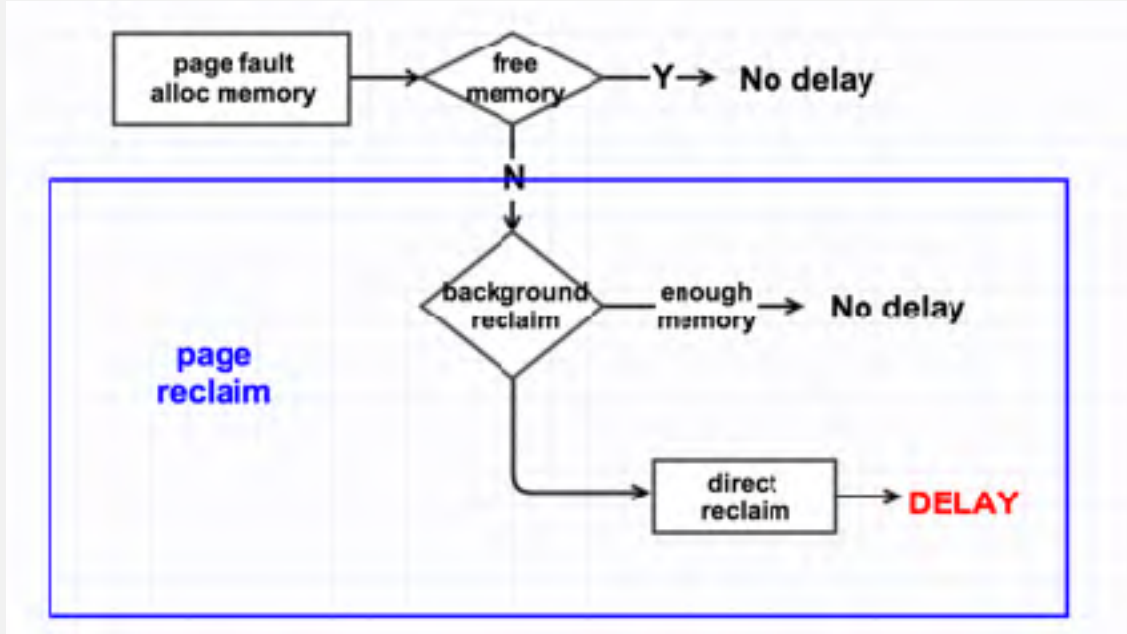
- 关于内存页回收

- 进程在分配内存(物理内存)时, 如果系统free pages不足, 就会去回收pagecache(包括clean page和dirty page)



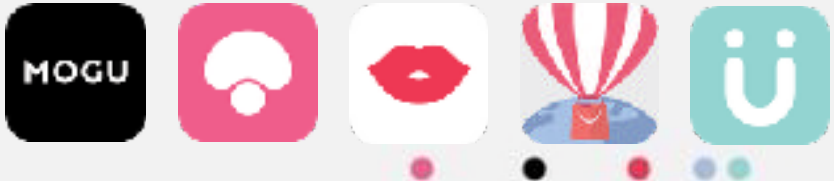
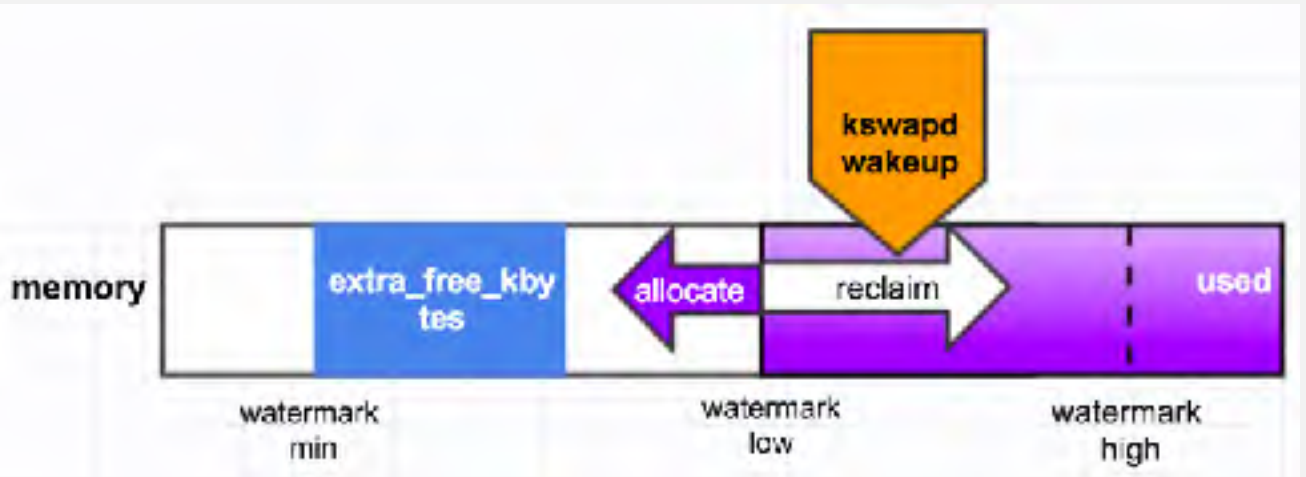
然后问题就来了...

- dirty page要写入disk，导致内存分配很慢

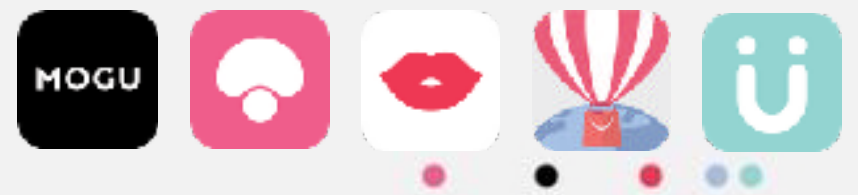
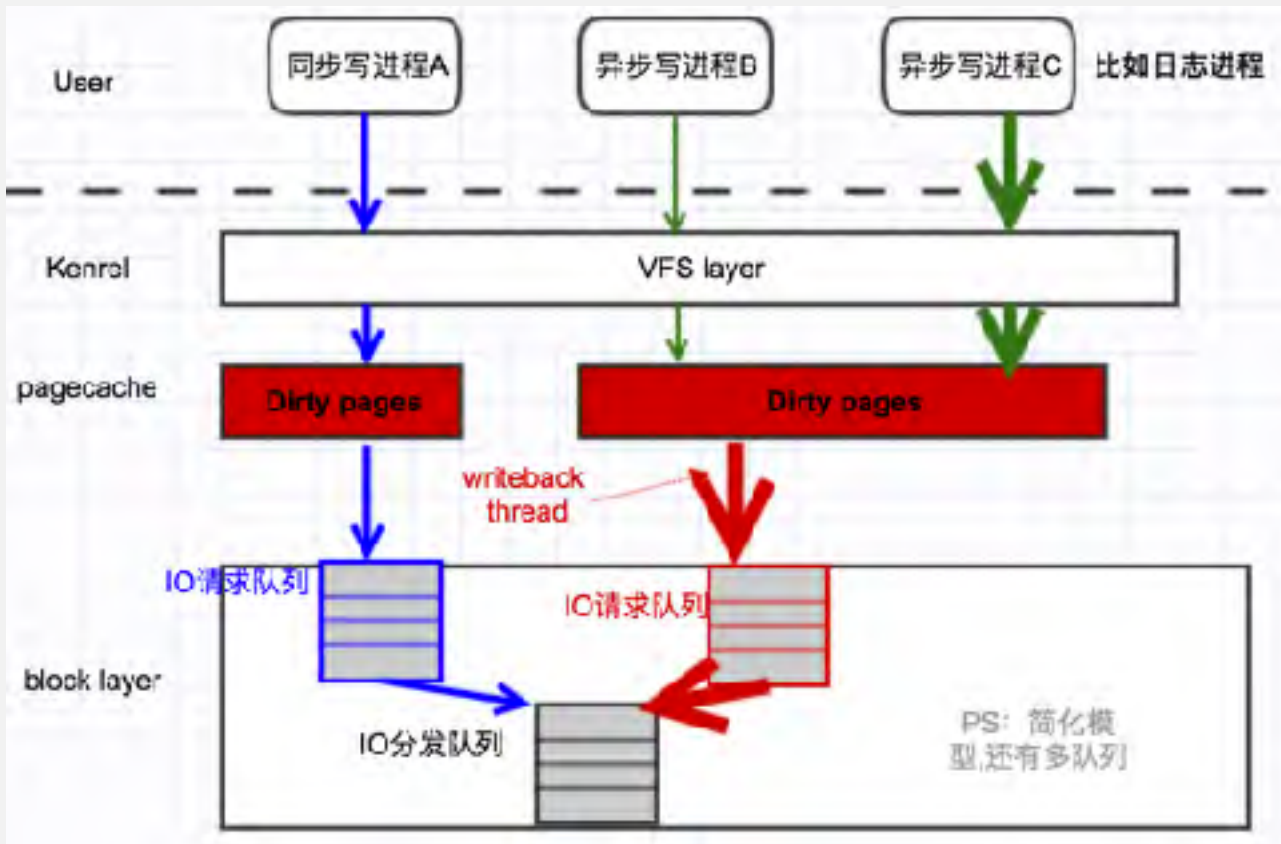


不优雅但有效的解决方案

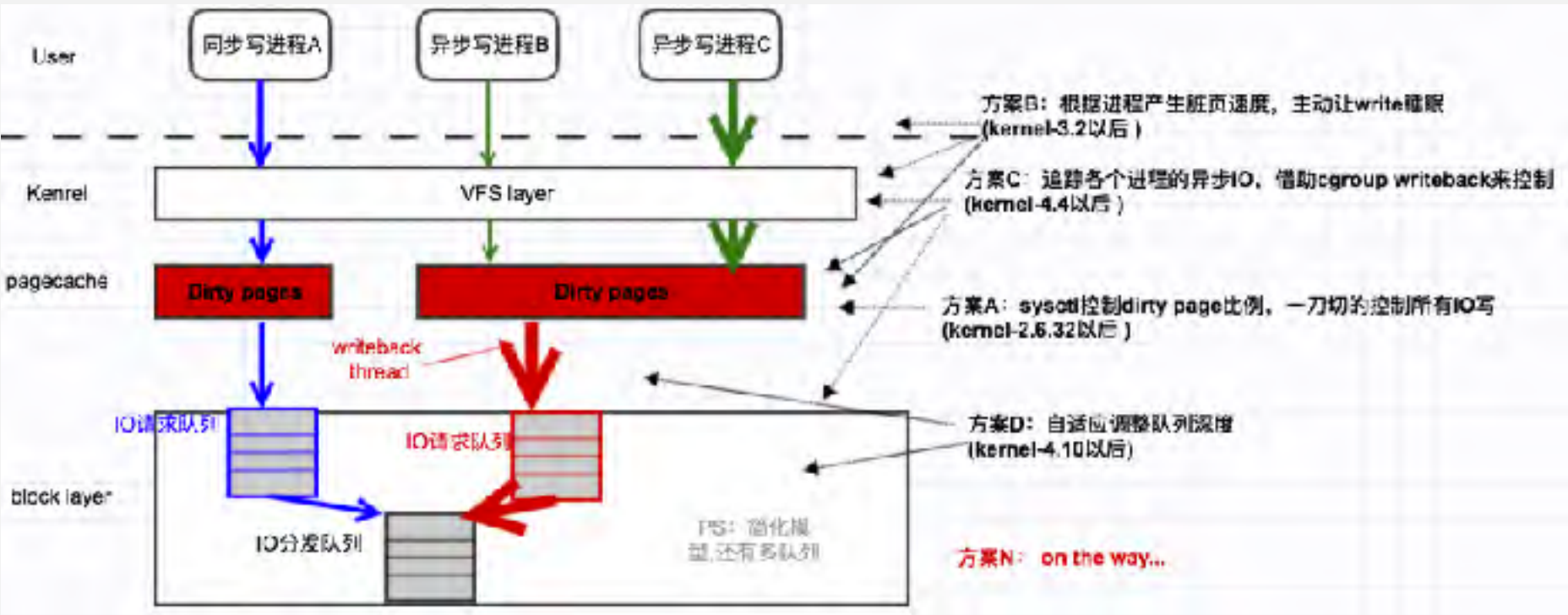
- 调整内存水位，限制page cache大小，确保系统有足够free page



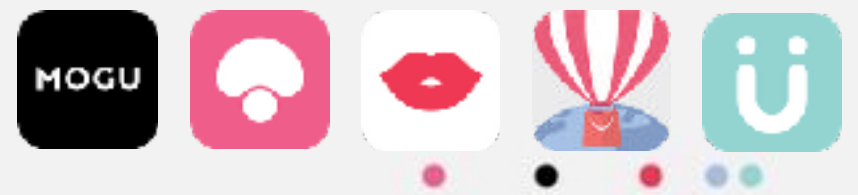
异步写IO的突发： 一个很棘手的问题



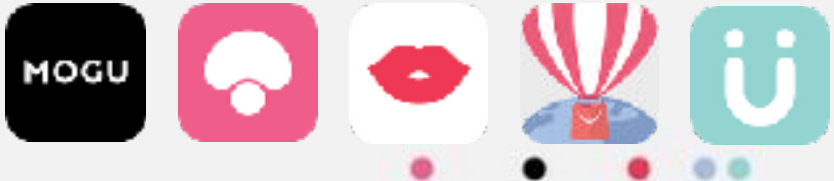
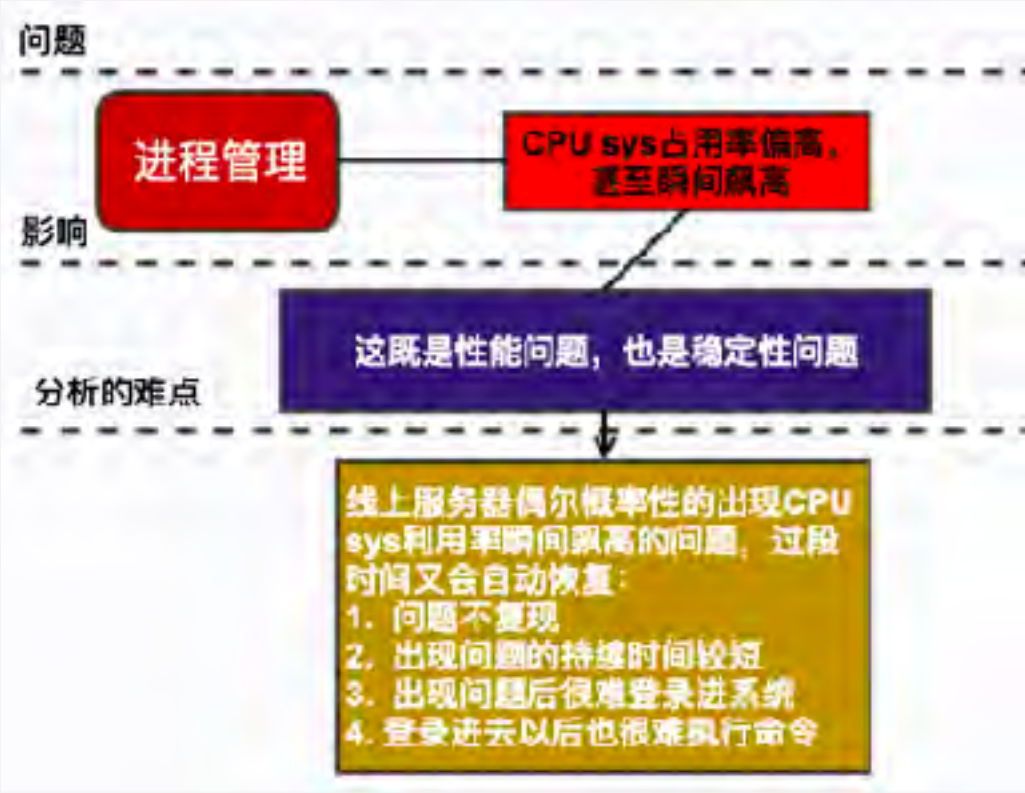
一些解决方案



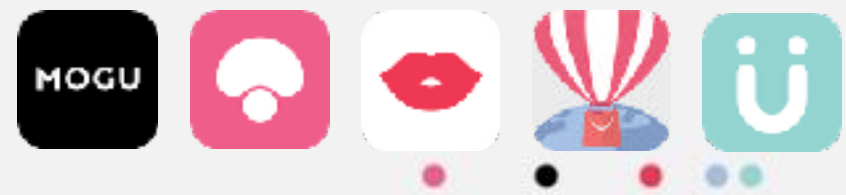
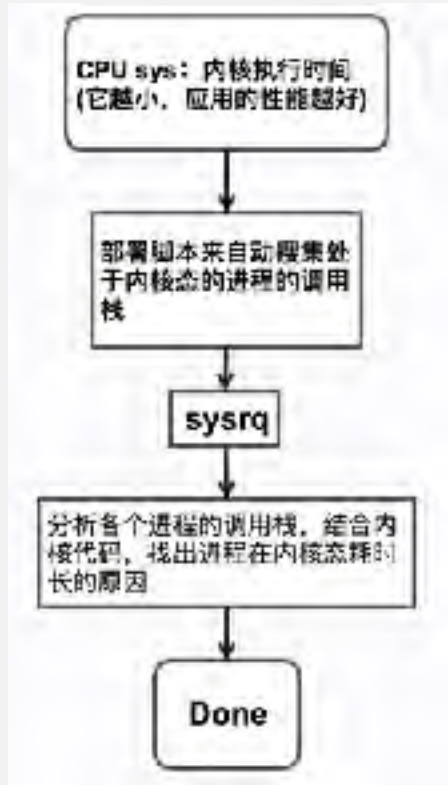
MEILI INC.



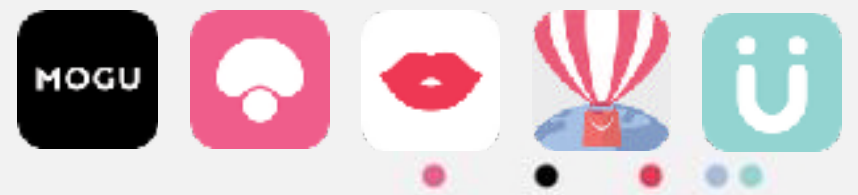
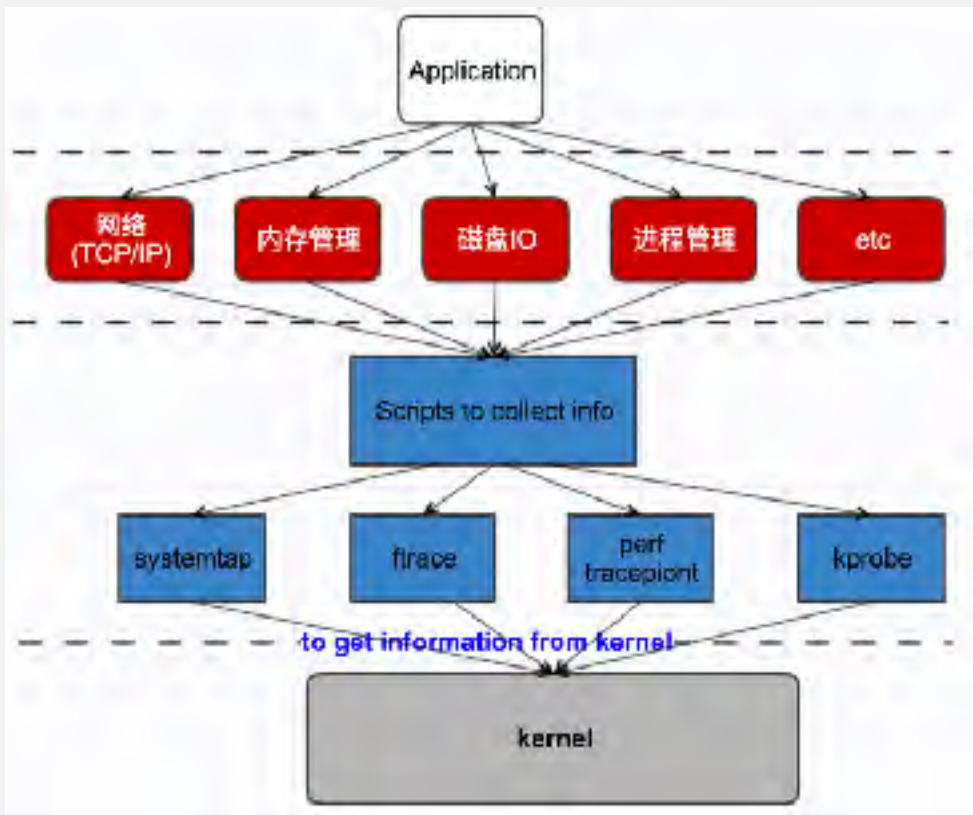
进程管理代表性问题：CPU sys利用率高



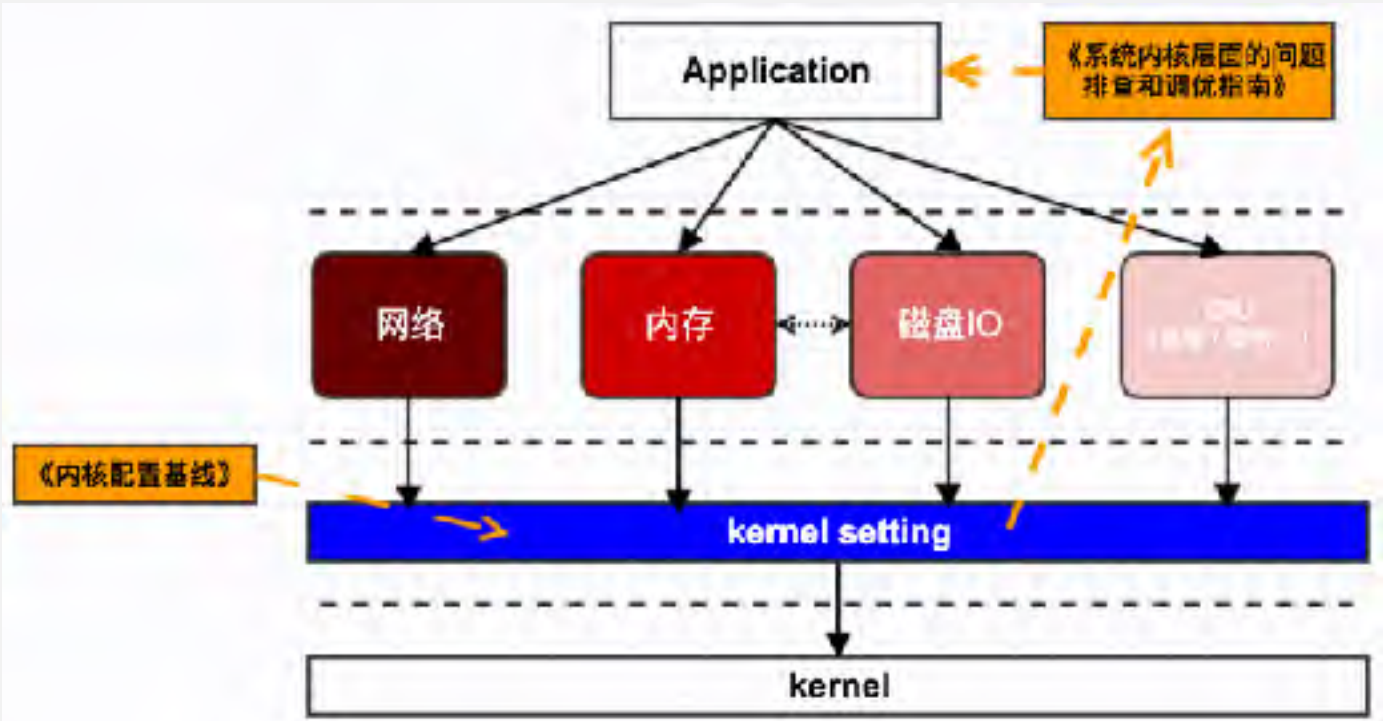
sys瞬间飙高又恢复的问题分析



对于这些问题的故障现场收集方法做一个总结

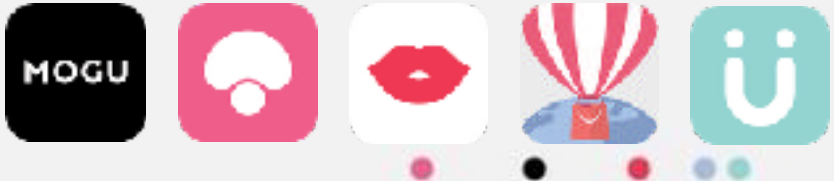


解决这些问题的一些沉淀

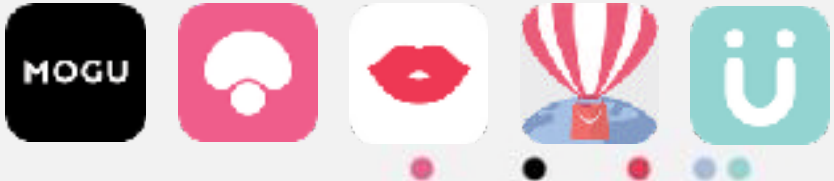
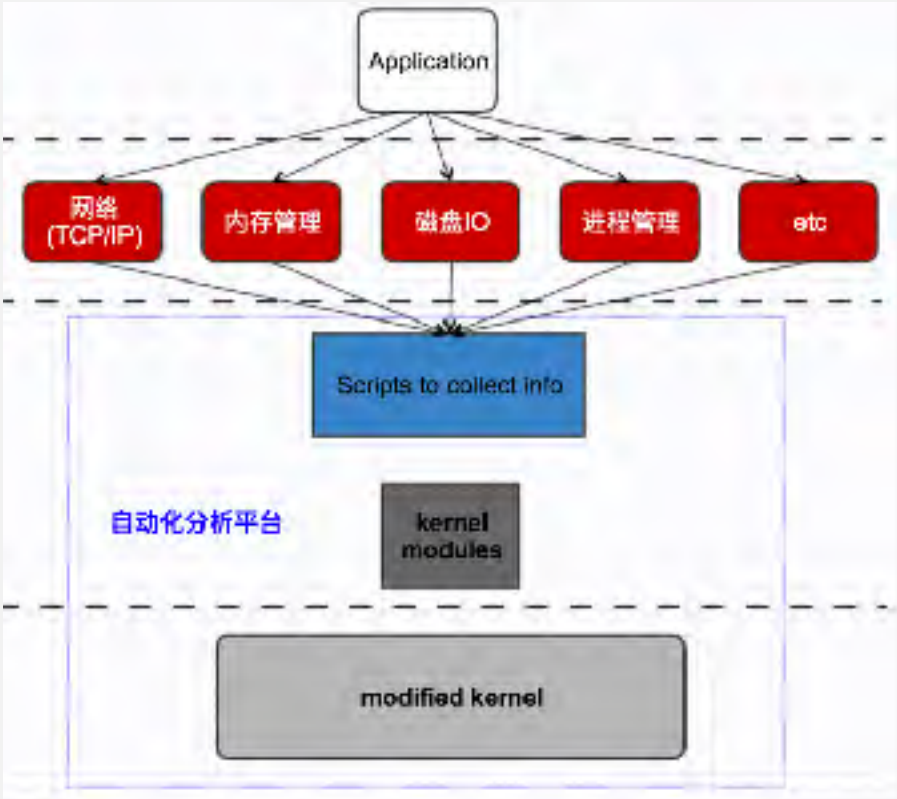


meili Meili Inc.

MEILI INC.



未来的规划...





THANKS