基本问题：在电脑面前什么都会，过三天完全忘记，然后被问的哑口无言，原来平时学看的都是无效操作

- ✓ cpu为什么高（说）
- ✓ 系统为这么慢（学）
- ✓ 内存为这么大（斗）
- ✓ crash怎么解决（唱）

- ✓ 服务发现与在线扩容
- ✓ 一致性，主节点挂点 和重启 保证数据一致性

一般思路 cpu高（为什么高 原理还是不清楚）

top 查看cpu指标： **1** us，2 sy，4 wa， 3 hi，si（不是重点，谁都可以看）

1. perf寻找热点函数：（和高什么关系）（重点 这个可以控制）

   a 是否可以c++内敛，常量折叠， b 缓存命中率 for循环优化 c 内存对齐 。

2. 特定功能的内核线程高不好解决，但能看问题原因，还是检查自己程序，不能归咎系统机制不好。

   检查系统调用函数（strace）和中断类型（`watch -d cat /proc/softirqs`）

3. 检查系统设计：线程进程设置比例，甚至协程代替（重点 这个可以控制。减少cs）

4. 网络io 检查一下：网络模型 epoll ，回到step 3 （后台服务都是io类型，次要重点）

   滑动窗口与带宽瓶颈（外界因素非重点，回到 1 2 3上）

   磁盘io：是否启用缓存（一般不占用过多cpu，引起变慢。这个不是重点，但是属于cpu的指标。liunx优化专栏还例子 证明。）

一般思路 cpu高（为什么高 原理还是不清楚）

遗漏：

cpu 高判断标准：cpu 基本概念 什么是正常的。？？

其实真正情况一片黑，无思下手

- **编译器优化**：很多编译器都会提供优化选项，适当开启它们，在编译阶段你就可以获得编译器的帮助，来提升性能。比如， gcc 就提供了优化选项 -O2，开启后会自动对应用程序的代码进行优化。
- **算法优化**：使用复杂度更低的算法，可以显著加快处理速度。比如，在数据比较大的情况下，可以用 O(nlogn) 的排序算法（如快排、归并排序等），代替 O(n^2) 的排序算法（如冒泡、插入排序等）。
- **异步处理**：使用异步处理，可以避免程序因为等待某个资源而一直阻塞，从而提升程序的并发处理能力。比如，把轮询替换为事件通知，就可以避免轮询耗费 CPU 的问题。
- **多线程代替多进程**：前面讲过，相对于进程的上下文切换，线程的上下文切换并不切换进程地址空间，因此可以降低上下文切换的成本。
- **善用缓存**：经常访问的数据或者计算过程中的步骤，可以放到内存中缓存起来，这样在下次用时就能直接从内存中获取，加快程序的处理速度。

怎么评估性能优化的效果?

首先，来看第一个问题，怎么评估性能优化的效果。

我们解决性能问题的目的，自然是想得到一个性能提升的效果。为了评估这个效果，我们需要对系统的性能指标进行量化，并且要分别测试出优化前、后的性能指标，用前后指标的变化来对比呈现效果。我把这个方法叫做性能评估“三步走”。

1. 确定性能的量化指标。
2. 测试优化前的性能指标。
3. 测试优化后的性能指标。

先看第一步，性能的量化指标有很多，比如 CPU 使用率、应用程序的吞吐量、客户端请求的延迟等，都可以评估性能。那我们应该选择什么指标来评估呢?

我的建议是**不要局限在单一维度的指标上**，你至少要从应用程序和系统资源这两个维度，分别选择不同的指标。比如，以 Web 应用为例：

例子：

系统为什么这么慢：准备

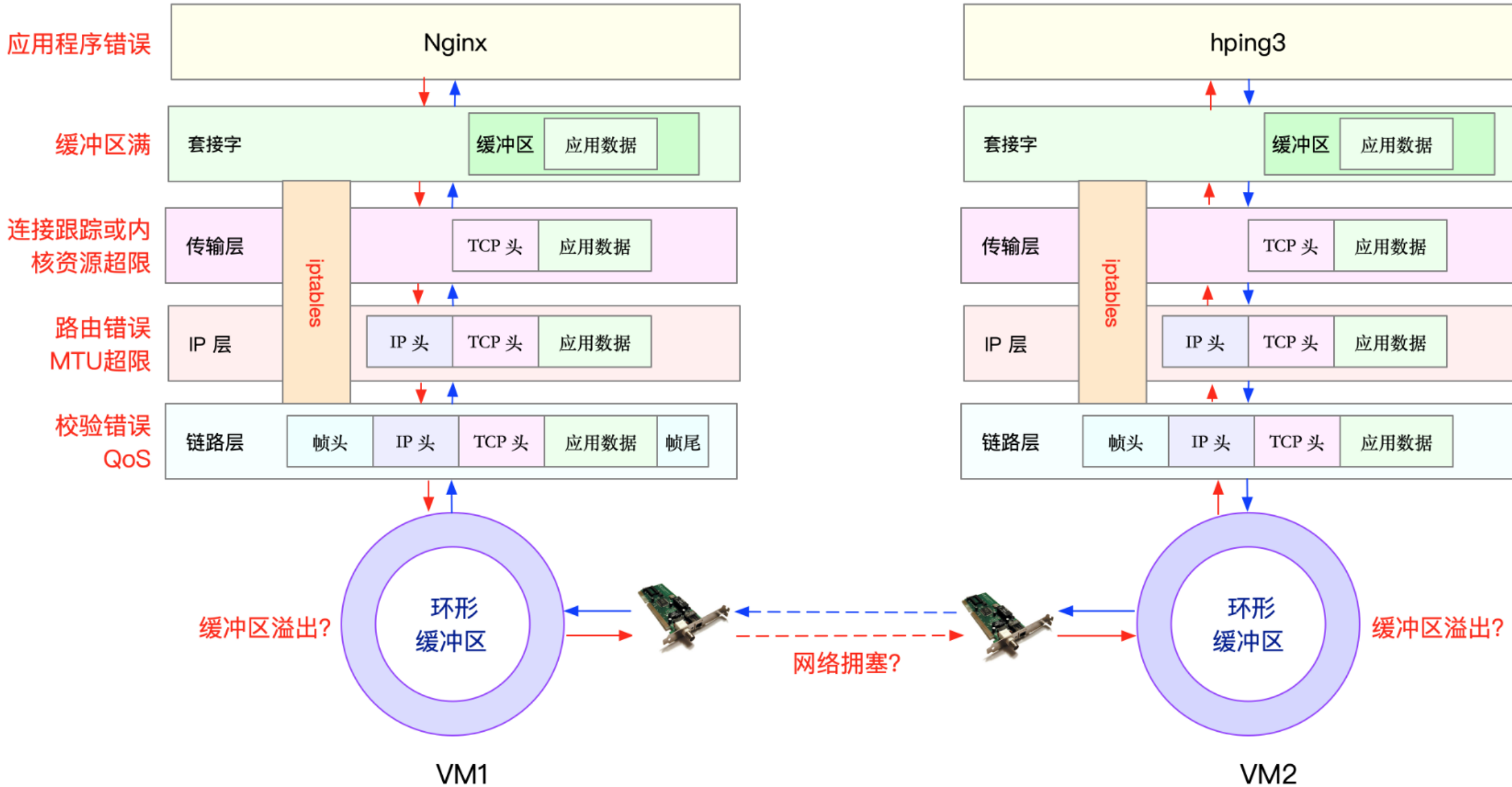
- 系统的生命周期大致大致可以按照顺序来看

  从客户端到服务器，然后在服务器上进行解析，执行 并返回结果给客户端。

- 使用 一个profiler。*Use a profiler.*
- 查看程序执行时的汇编码。*View the assembly code when the program is executed.*

如果你觉得需要程序有更好的执行速度，那么，最基本的方法就是使用一个profiler和愿意去查看一下其汇编代码以找到程序的瓶颈

只有找到了程序的瓶颈，此时才是真正在思考如何去改进的时候


应用程序错误
Nginx
缓冲区满
套接字
缓冲区
应用数据
连接跟踪或内核资源超限
传输层
iptables
TCP 头
应用数据
路由错误
MTU超限
IP 层
IP 头
TCP 头
应用数据
校验错误
QoS
链路层
帧头
IP 头
TCP 头
应用数据
帧尾
缓冲区溢出?
环形缓冲区
网络拥塞?
VM1
hping3
应用程序
套接字
缓冲区
应用数据
缓冲区满
传输层
iptables
TCP 头
应用数据
连接跟踪
核资源超
IP 层
IP 头
TCP 头
应用数据
路由错误
MTU超限
链路层
帧头
IP 头
TCP 头
应用数据
校验错误
QoS
环形缓冲区
缓冲区溢出?
VM2

进程内存为什么高

- 虚拟内存占用高一定问题吗？

原理： unallocated(不占用空间)，cahed（物理内存） uncached（磁盘）

mac top

PhysMem: 16G used (2566M wired), 320M unused.

内存为什么这么高？

- 内存分配与回收 【这个有c++demo例子，网上有，以前动手练习过。现在想不起来了】

malloc 和free会导致内存泄漏吗？不会

如果小于128k 直接通过mmap方式申请和释放不会有问题。

如果小于128k，会按照brk方式申请顺序申请。

如果申请连续空间需要**额外8字节记录大小**（对齐自然的）

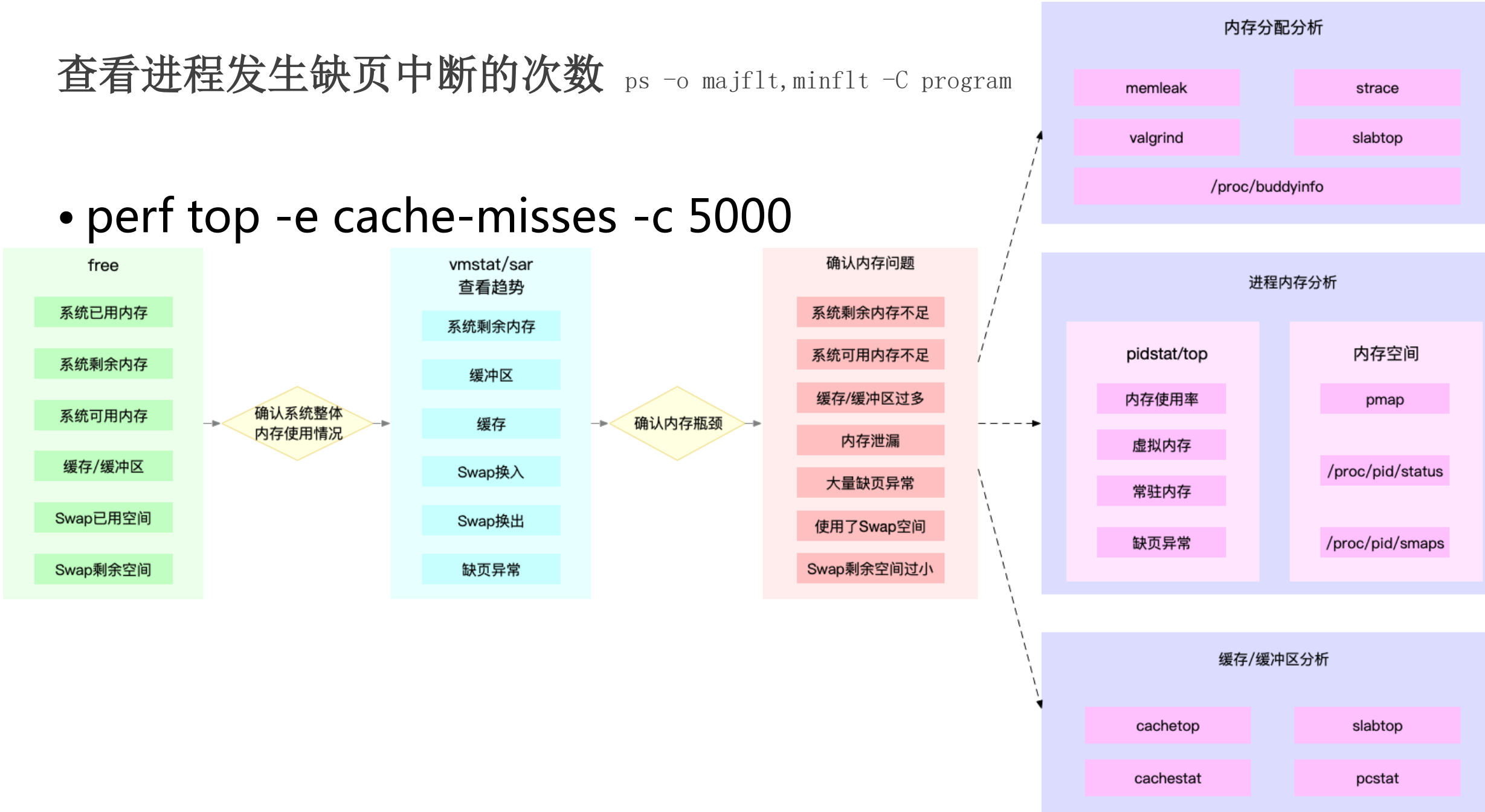
释放时候，需要低地址先释放，低地址不释放空间永远被占用

重复利用时候造成内存碎片。

不考虑多线程问题，从一片连续地址分配上角度考虑

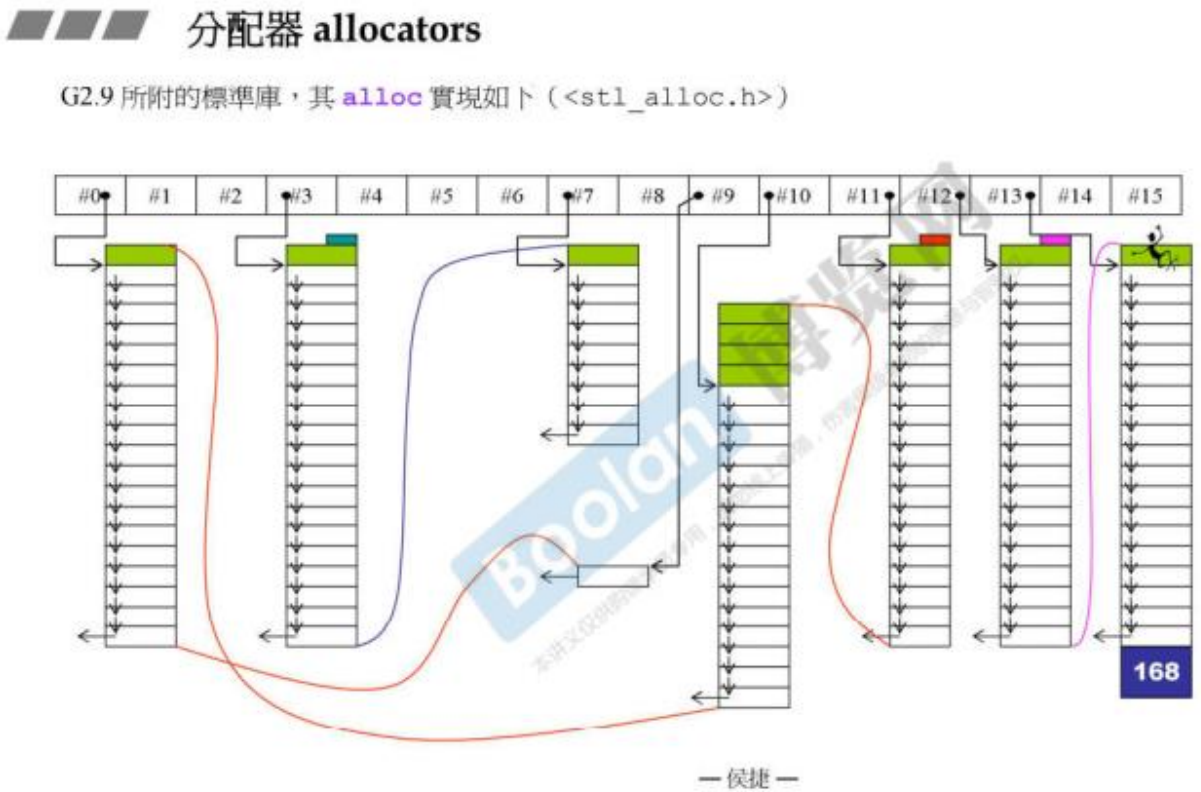
查看进程发生缺页中断的次数 `ps -o majflt,minflt -C program`

- perf top -e cache-misses -c 5000



内存为什么这么高？

task2:tcmalloc 优缺点。



内存：valgrind

- 阅读使用手册 FQA

- Your program is then run on a synthetic CPU provided by the Valgrind core

如何解决crash（1）问题

- tail -f valgrind.log

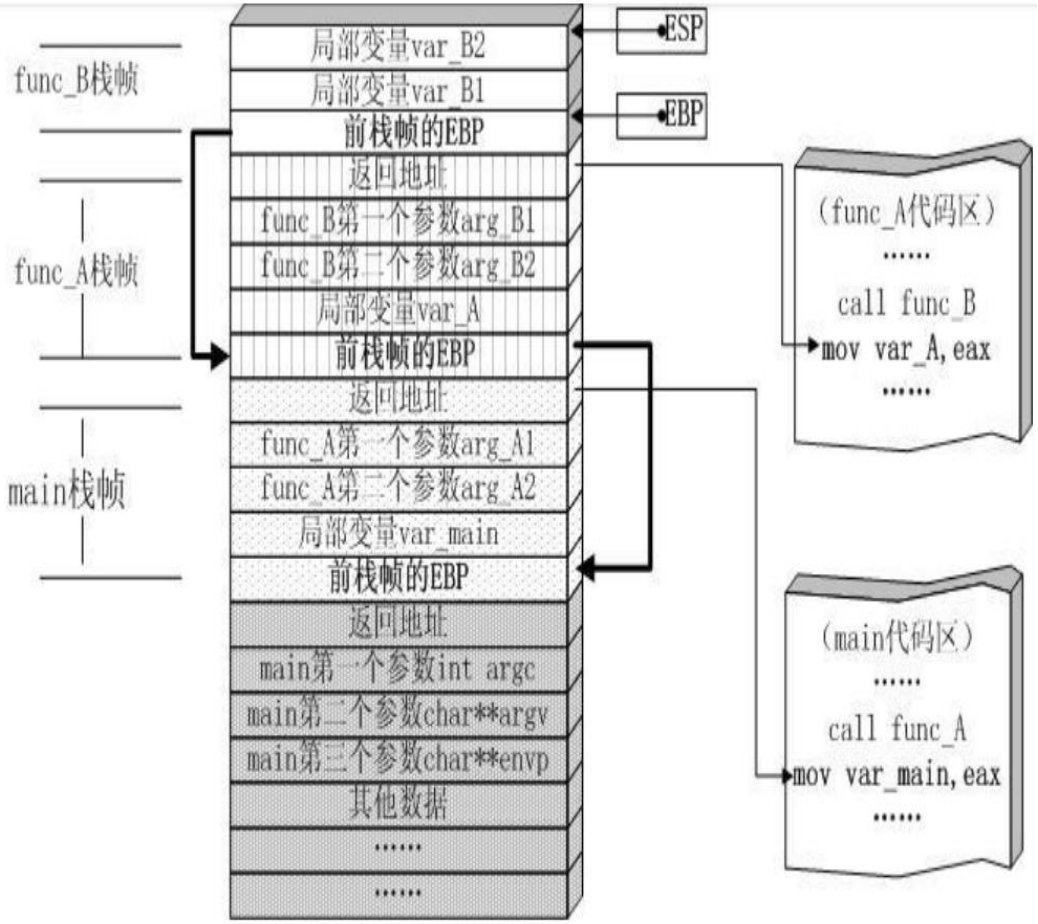
==18072== Invalid write of size 8

==18072== at 0x7A9A2C: ??? (in /home/work/proxy/gcache-proxy)

- (gdb) bt //打印堆栈信息
- #0 0x0000000000472b1b in ?? ()
- #1 0x0000000000000006 in ?? () at /usr/lib/gcc/x86_64-redhat-linux/4.4.7/../../../../include/c++/4.4.7/bits/stl_vector.h:131
- #2 0x0000000000fb0490 in ?? ()
- #3 0x00007f6ff00c1370 in ?? ()
- #4 0x000000000042a8f0 in std::_Rb_tree<std::basic_string<char

如何解决crash（2）准备 汇编与函数堆栈 的关系

- https://godbolt.org/z/dcvvhM
- https://godbolt.org/z/8sPPPP
- 、



如何解决crash（3）如何查看一个corrupt stack

- https://izualzhy.cn/why-the-code-stack-is-overflow
- https://sourceware.org/gdb/onlinedocs/gdb/Memory.html
- https://www.jianshu.com/p/0299f56edab5
- https://xz.aliyun.com/t/2554
- http://blog.sina.com.cn/s/blog_72ef7bea0102w1c6.html

服务发现与在线扩容

✓ 把IP和端口看成一个记录，需要一个高可用服务存储这些数据，而不是在客户端记录

这样 msyql ，redis etc zk 这样工具联系在一起了。