第09章：虚拟内存

假装我们有用不完的物理内存

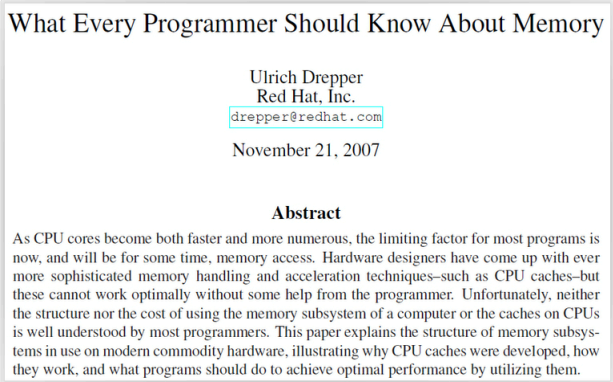
视频解说

导读

面试官：请解释一下什么是虚拟内存? 有些面试官喜欢在面试的时候考察一下，或多或少能够反映出候选人的计算机基础素养，实际上内存管理（包含虚拟内存）的议题非常复杂，不是三言两语能讲清楚的，但至少你要做到心中有数，有宏观的整体把握，不至于一问三不知。

推荐你认真阅读一下Ulrich Drepper撰写的长达114页的经典论文：What Every Programmer Should Know About Memory，如果你实在没有耐心看完它，或者想了解其中的重点内容，那么也可以通过观看我自己录制的小视频来了解其中的重点内容：

每个程序员都应该知道的内存知识 (第3部分：虚拟内存)



What Every Programmer Should Know About Memory

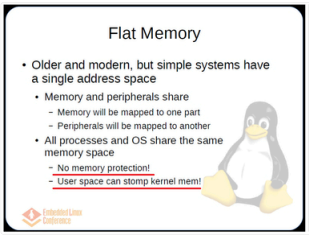
Ulrich Drepper
Red Hat, Inc.
drepper@redhat.com

November 21, 2007

Abstract

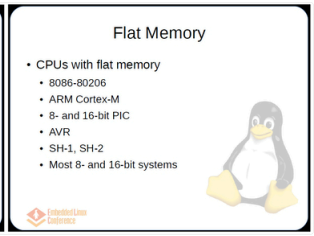
As CPU cores become both faster and more numerous, the limiting factor for most programs is now, and will be for some time, memory access. Hardware designers have come up with ever more sophisticated memory handling and acceleration techniques–such as CPU caches–but these cannot work optimally without some help from the programmer. Unfortunately, neither the structure nor the cost of using the memory subsystem of a computer or the caches on CPUs is well understood by most programmers. This paper explains the structure of memory subsystems in use on modern commodity hardware, illustrating why CPU caches were developed, how they work, and what programs should do to achieve optimal performance by utilizing them.

虚拟内存属于 **操作系统 + 硬件** 范畴，首先，并非所有系统都使用虚拟内存，简单的（或内存受限的）系统就有可能使用单一物理内存地址空间。



Flat Memory

- Older and modern, but simple systems have a single address space
  - Memory and peripherals share
    - Memory will be mapped to one part
    - Peripherals will be mapped to another
  - All processes and OS share the same memory space
    - No memory protection!
    - User space can stomp kernel mem!



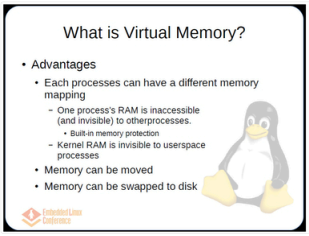
Flat Memory

- CPUs with flat memory
  - 8086-80206
  - ARM Cortex-M
  - 8- and 16-bit PIC
  - AVR
  - SH-1, SH-2
  - Most 8- and 16-bit systems

物理内存有什么问题？ 1. 内存空间不够 2. 产生内存碎片 3. 没有内存保护

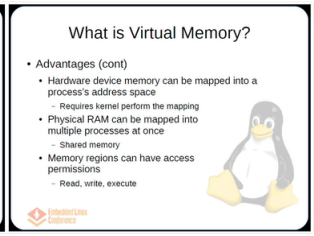
（虚拟内存闪亮登场，本质就是增加一个中间层，计算机科学的任何问题都可以通过增加间接层解决！）

虚拟内存有什么优点? 1. 可以使用磁盘交换空间 2. 虚拟地址到物理地址映射灵活 3. 进程地址空间隔离



What is Virtual Memory?

- Advantages
  - Each processes can have a different memory mapping
    - One process's RAM is inaccessible (and invisible) to otherprocesses.
      - Built-in memory protection
    - Kernel RAM is invisible to userspace processes
  - Memory can be moved
  - Memory can be swapped to disk



What is Virtual Memory?

- Advantages (cont)
  - Hardware device memory can be mapped into a process's address space
    - Requires kernel perform the mapping
  - Physical RAM can be mapped into multiple processes at once
    - Shared memory
  - Memory regions can have access permissions
    - Read, write, execute

相关的一些基本概念：

虚拟地址到物理地址的转换涉及哪些硬件/软件/过程？ （MMU & TLB & Page Fault & Lazy allocation）

Lazy allocation：当程序向操作系统申请动态内存时，系统会调用相应的函数分配内存，但是这种分配并不是实时的，内核会去修改页表 Page Table，但是只有当用户真的开始使用这块内存时，才会分配物理页面（通过Page Fault 来触发）。

Linux Kernel Space (kmalloc, vmalloc) **& User Space** (malloc)

动态内存管理是很重要的功能，内存一直都是很宝贵的资源，一个好的内存管理策略可以极大地提升系统性能，就用户空间的动态内存管理而言，主要的实现有：ptmalloc，tcmalloc，jemalloc ...

`GLIBC malloc` 内部通过 `brk` 或 `mmap` 系统调用向内核申请堆内存

==> glibc源码一瞥 （相关的补充参考资料： Understanding glibc malloc）

目前glibc的实现是ptmalloc2，它使用 chunk 作为内存管理的基本单元（Allocated chunk，Free chunk，Top chunk，Last Remainder chunk），采用边界标记法，用户 free 掉的内存并不是都会马上归还给系统，ptmalloc 会统一管理空闲的 chunk，当用户进行下一次分配请求时，ptmalloc 会首先试图在空闲的 chunk 中挑选一块给用户，这样就避免了频繁的系统调用，降低了内存分配的开销，ptmalloc 将相似大小的 chunk 用链表链接起来，这样的一个链表被称为一个 bin，ptmalloc 按照大小维护了多个类型的bin（fast bin，small bin，large bin，unsorted bin，除了fast bin是单向链表之外，其他的都是双向链表）... 另外我们还可以通过mallopt()来改变相关的内存分配行为的参数。

*** **案例参考（遇到的问题 & 解决的方法）**：阿里（华庭=庄明强）：Glibc内存管理ptmalloc源代码分析

学习方式

CMU教授的视频教程 - Lecture17：虚拟内存概念

CMU教授的视频教程 - Lecture18：虚拟内存系统

CMU教授的视频教程 - Lecture19：动态内存分配（基础）

CMU教授的视频教程 - Lecture20：动态内存分配（进阶）

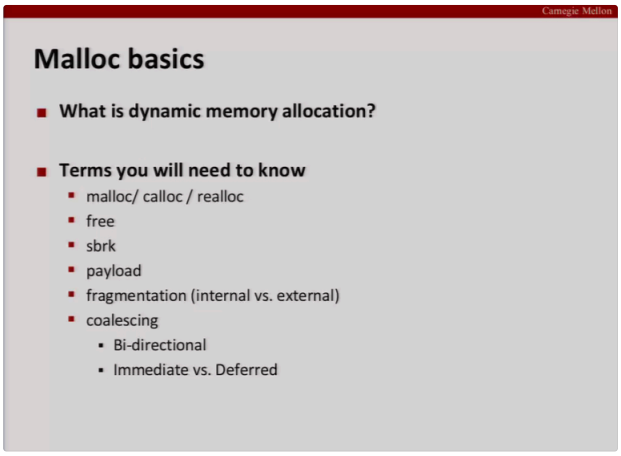
实验解读

自己动手写一个内存分配器（Memory Allocator）== 绝非易事！

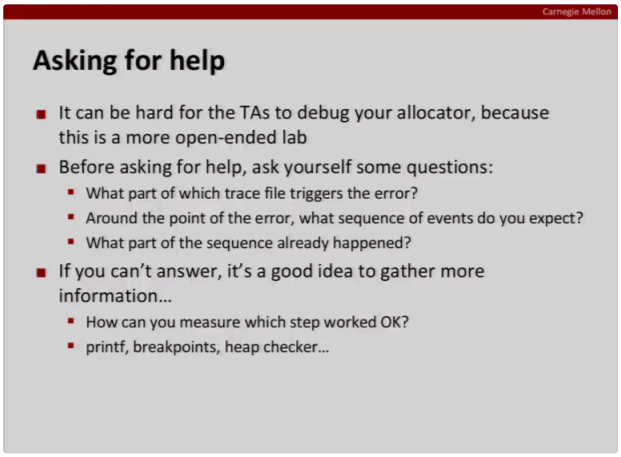
- *Malloc Lab [Updated 9/2/14]* (README, Writeup, Release Notes, Self-Study Handout)

  Students implement their own versions of `malloc`, `free`, and `realloc`. This lab gives students a clear understanding of data layout and organization, and requires them to evaluate different trade-offs between space and time efficiency. One of our favorite labs. When students finish this one, they really understand pointers!

CMU助教的视频：Malloc Lab （其他的补充教程：Malloc tutorial）



CMU助教的视频：Debugging Malloc Lab （补充：如何以聪明的方式提问（Eric Steven Raymond））



延伸阅读

- Linux专题介绍：内存管理

Previous
第08章：异常控制流

Next
第10章：系统级I/O

Last updated 3 years ago