Lecture 13 虚拟存储器

1. 虚拟存储器

介绍

构造虚拟存储器主要有两个动机

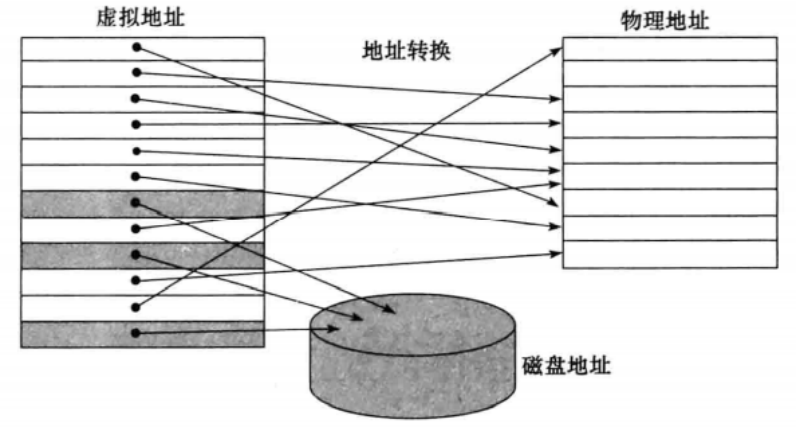
- 允许云计算在多个虚拟机之间有效而安全地共享存储器
    - 我们希望将每个程序都编译到它自己的地址空间 address space，存储器只能有该程序访问的独立的一连串地址
    - 虚拟存储器实现程序地址空间到物理地址 physical address 之间的转换
- 消除一个小而受限的存储容量对程序设计造成的影响
    - 允许单用户程序使用超过 main memory 的容量

尽管虚拟存储器和 cache 的工作一样，可以把 main memory 看成 cache ，用于与二级存储器（disk）等的交互，CPU 硬件和操作系统共同管理，将虚拟地址转换为物理地址

- 处理器产生一个虚拟地址 virtual address
- 结合软硬件转换成一个物理地址 physical address
- 访问主存

处于历史原因，在虚拟存储器中

- block 被称为 page
- miss 被称为 page fault，指访问的 page 不在 main memory 中

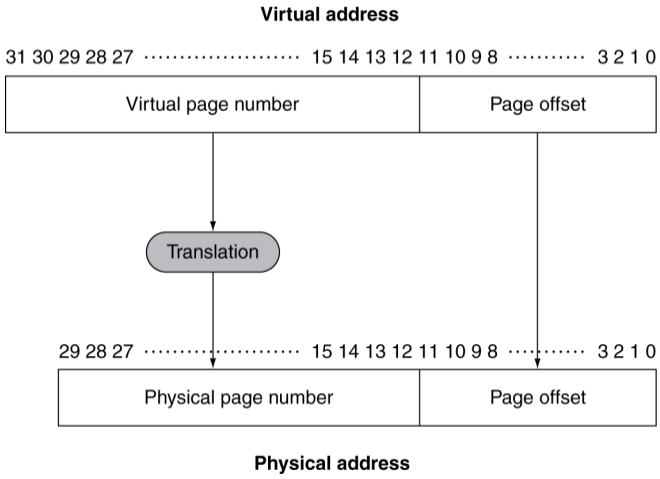


- 虚拟存储器和物理存储器都被划分成 page
- 虚拟 page 可能在 main memory 中，可能在 disk 中
- 物理 page 可以被两个指向相同物理地址的虚拟地址共享

在虚拟存储器中，地址被划分为

- virtual page number：有几个 page
- page offset：描述一个 page 的大小

转换的效果如下，物理地址的 page offset 和虚拟地址的 page offset 是一样的



Page fault

缺失在虚拟存储器中通常被称为 page fault，一次 page fault 需要花费数百万个时钟周期，因此设计的时候应该尽可能避免 page fault

- page 足够大，目前典型的 page 大小为 4~16 KiB
- 虚拟存储器中页是以全相联 full associative
- 设计一些算法
- 采用写回 write back 机制

如果发生 page fault

- 操作系统获得控制权
- 从下一级存储器 disk 中找到该 page
- 将它放入 main memory 中

2. Page 的存放和查找

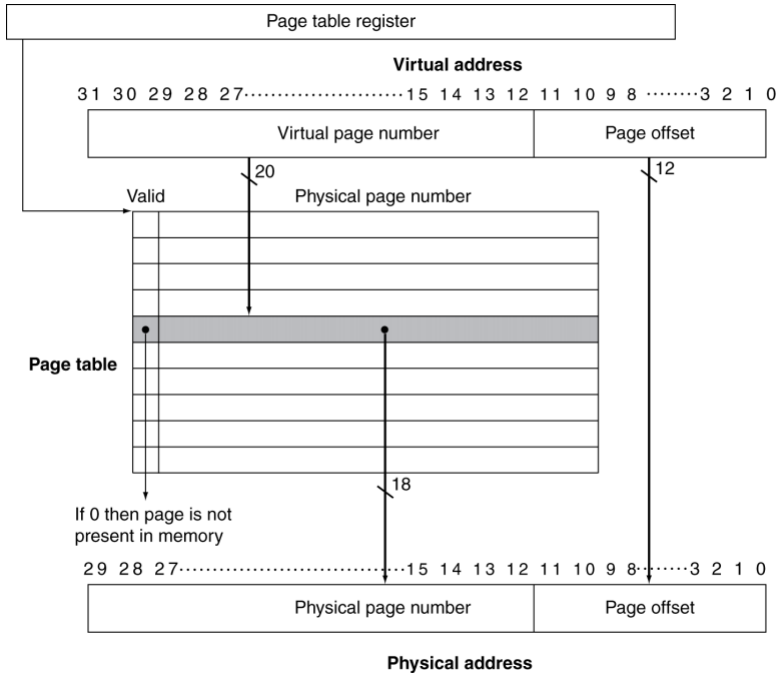
Page table

我们使用一个索引存储器的表来定位 page，这种结构被称为 page table，相当于把一个 virtual address 映射到一个 physical address 上

- page table 被放在 main memory 中，它使用 virtual address 作为 index，找到相应的 physical address
    - page table 中的每一项被称为 Page Table Entry（PTE）
- 每个程序都有自己的 page table，用来将程序的 virtual address space 映射在 main memory 上
- 为了找到 page table，硬件包含一个指向 page table 首地址的存储器，我们称为 page table register

Page Table Entry（PTE）

下图表明了 page table 中如何从虚拟地址指向物理地址的



Page Table Entry（PTE）有几个部分

- Valid bit（1 bit）
    - 1 代表有效，说明存的是 main memory 的 physical address
    - 0 代表无效，说明存的是 disk 的 physical address，即发生了一次 page fault

- Dirty bit（1 bit）
    - 由于使用写回机制，表示数据 main memory 中是否已经发生了替换
    - 当该 page 被替换出 main memory 的时候再复制到 disk 中去
- Reference bit / Use bit（1 bit）
    - 表示是否为最近访问
    - 当某一 page 被访问时置 1
    - 使用 LRU 原则进行替换
    - 操作系统定期将 reference bit 清零，然后再重新记录
- Physical page number：存的是 physical address

上图假设 virtual address 的长度为 32 bit，一个 page 的大小是 2^{12} byte， physical address 的长度为 30 bit，它忽略了 Dirty bit 和 Reference bit

3. 加速地址转换 TLB

TLB 转换逻辑

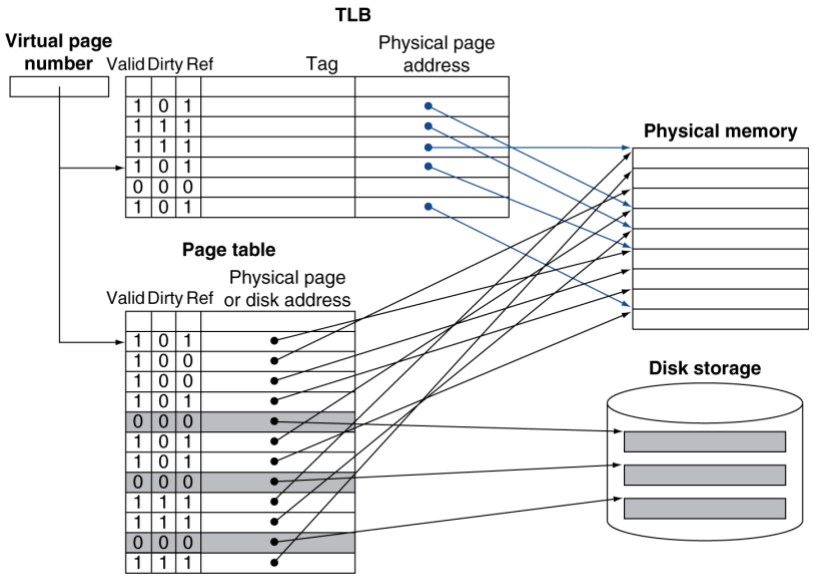
由于 page table 被放在 main memory 中，因此程序每次至少需要访问两次 main memory

- 第一次去 page table 找到物理地址
- 第二次才能从 main memory 中获得数据

现代处理器都包含一个特殊的 cache 来记录最近使用过的地址变换，这个特殊的地转换 cache 通常称为**快表 Transalation-Lookaside Buffer（TLB）**

- 是用于记录最近地址的映射信息的高速缓存，可以避免每次都要访问 page table

下图是一个使用 TLB 的逻辑，流程如下

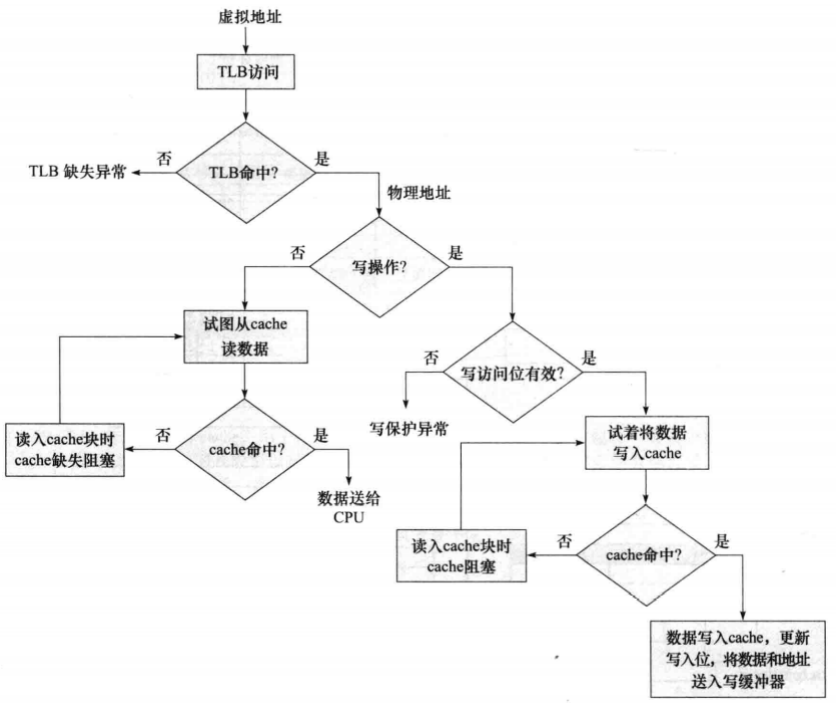



```
访问 TLB
if(命中){
    Physical page number 形成物理地址
    Reference bit 置位
    if(是写操作){
        Dirty bit 置位
    }
    考虑可能存在写回的问题

}else{
    if(page 在 main memory 中){
        处理器可以将 page table 中指定的 PTE 放入 TLB 中
    }
    if(page 不在 main memory 中){
        发生 page fault
        处理器调用操作系统异常处理
    }

    从 TLB 中选择一项进行替换
    Reference bit 置位
    if(是写操作){
        Dirty bit 置位
    }
    考虑可能存在写回的问题
}
```

存储器访问逻辑

事实上， cache 中存储的地址和 memory 中一样，都是 physical address，也就是说 TLB 或者 Page table 会将 virtual address 映射成 physical address 后，再进访问

存储器访问的逻辑如下



4. 虚拟存储器中的保护

因为虚拟存储器允许多个进程共享一个 main memory，它必须为这些进程和操作系统提供存储保护

- 每个进程有它自己的虚拟地址空间，操作系统管理 page table
- 每一个用户不可以修改 page table 的映射，这一操作必须由操作系统完成
- 当进程希望以受限的方式共享信息时，操作系统必须协助它们

为了使操作系统能够保护虚拟存储系统，硬件至少具有下面的基本能力

- 特权管理 Privileged supervisor mode，又叫 kernel mode：运行操作系统进程的模式和
- 系统调用异常 System call exception：处理器在用户模式和管理模式相互切换
    - 它使用一些特权指令 Privileged instructions
- Page table 和其他只能在管理模式 supervisor mode 下访问的状态信息

5. 存储器总结

在存储器层次结构的每一层，我们需要注意

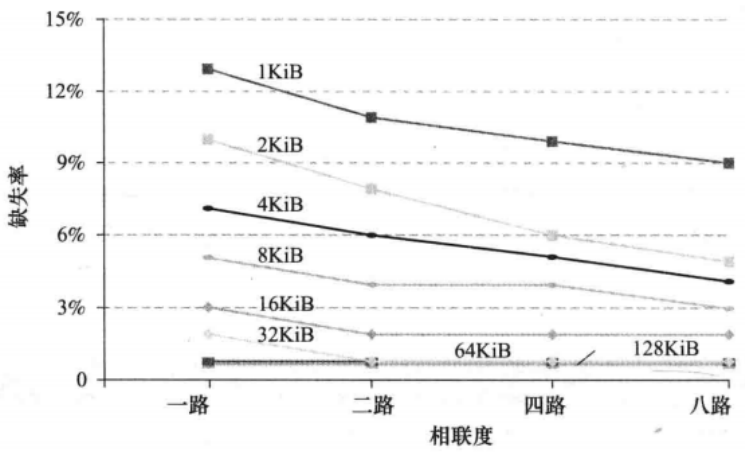
- Block placement
  block / page 是如何放在存储器里的
- Finding a block
  如何找到 block / page
- Replacement on a miss
  当发生了一次 miss / page fault 如何替换
- Write policy
  写的策略是什么

Block Placement

存储器的放置结构

- 直接映射
- n 路组相联
- 全相联

较高的结合性会减少 miss rate，但是会增加复杂度，开销和 access time



Finding a Block

Associativity	Location method	Tag comparisons
Direct mapped	Index	1
n-way set associative	Set index, then search entries within the set	n
Fully associative	Search all entries	#entries
	Full lookup table	0

虚拟存储器

- 全相联，使用查找表 Page Table
- 减少 miss rate

Cache 和 TLB

- 组相联
- 部分 cache 使用直接映射

Replacement

替换方法

- LRU
- 随机

虚拟存储器

- 近似的 LRU 算法

Cache 和 TLB

- 在相连度不低的层次结构中实现 LRU 的代价太高了，一般只使用近似的 LRU
- 也可以采用随机替换，更容易实现

Write Policy

写直达

- 同时更新上层和下层
- 简化替换，但可能需要写缓冲区

写回

- 只更新上层
- 当发生替换时，更新较低级别

虚拟存储器和 Cache 都使用写回机制

Miss 的来源

Miss 的种类

强制缺失 Compulsory miss，又叫 cold start miss

- 不得不发生的 miss，比如刚刚开机的时候的时候，main memory 里面什么都没有

容量缺失 Capacity miss

- cache 的大小有限，只能采用替换策略

冲突缺失 Conflict miss，又叫 collision misses

- 在非全相联的设计中，放置的位置有的冲突

设计方法

设计更改	对MISS RATE的影响	反面影响
增加 cache 的大小	减少 capacity miss	可能增加 access time
增加结合性 associativity	减少 conflict miss	可能增加 access time
增大 block size	减少 compulsory miss （一次搬的东西增加）	增加 miss penalty 如果 block size 过大，可能会增加 miss rate