Project 4 Virtual Memory 设计文档

中国科学院大学
段江飞
2018 年 12 月 5 日星期三

1. 内存管理设计

（1）你设计的页表项包含哪些内容？页表本身使用什么数据结构保存？

页表项的数据结构

```
    typedef struct pte {
    uint32_t ptentry;
    uint32_t sd_sect;
} pte_t;
```

sd_sect 是该页表项在 sd 卡中对应的位置，ptentry 的结构如下：

```
//Page table entry flags
// 31                          12 11 9  8   7   6   5
// +--------------20--------------+---+---+---+---+---+------+
// |       Page Table Index         |  C | D | V | G |   |
// +--------------------------------+---+---+---+---+---+------+
```

页表项的位主要是前面的页框号和后面的 D 和 V 位，D 位表示 Dirty，是否可写，V 表示 Valid，是否对应有页框。为了方便对 TLB 表项的设置，在页表项中添加了 C 位和 G 位，含义和 TLB 表项中的 C(Cache Coherency)和 G(Global)一样。

页表本身用一个数组来存储，在任务三种，目前设计是每个 PCB 对应一个页表，这样导致页表占的内存很大，加载的很慢，所以测试时，加载的内存很小。为了完成 bonus，我可能会修改数据结构，改为总只有一个页表。

（2）任务 1 和任务 2 中各自初始化了多少个页表项，以及使用了多少个物理页框保存页表？

任务 1 和任务 2 均初始化了全部的页表项(524288 项)，用了物理页框是 4096 个，对应 16M 到 32M 的地址空间。

（3）任务 2 和任务 3 中，进程的用户态栈的起始地址各是多少，栈空间各是多大？

进程用户栈的起始地址是 0xffc，栈空间大小为一个页的大小，即 4KB。

（4） 任务 1 和任务 2 中你设计的操作系统实际通过页表可以访问到的物理内存有多大？

任务一和二中的页框初始化了 16M，但任务一中由于 TLB 表项只有 32 项，可以访问到的物理内存大小为 256KB，任务二所能访问到的物理内存大小为 16MB。

（5）TLB miss 何时发生？在任务 2 中，你处理 TLB miss 的流程是怎样的？

当访问的虚拟地址对应的页表项不在 TLB 中，或者对应的页表项无效，发生 TLB miss。

任务中的 TLB miss 发生在访问的虚拟地址对应的页表项不在 TLB 中，处理就是直接将对应的页表项写入 TLB 表项。在任务三种的 TLB miss 处理流程如下：

TLB miss 处理流程：

首先，利用 tlbp 判断需要 refill 还是 invalid。

如果是 refill，就直接设置相应的 CP0 寄存器，然后写入 TLB。

如果是 invalid，则去查询相应的页表项是否有物理页框对应，如果有，则直接将 TLB 设置为有效，如果没有，则需要分配物理页框，如果物理页框没有了，需要进行页替换。

2. 缺页处理设计

（1） **何时会发生缺页处理？你设计的缺页处理流程是怎样的，此处的物理页分配策略是什么？**

缺页处理发生在 TLB 无效，且对应的页表项无效时。

缺页处理：将物理内存用链表组织起来，一个 freelist 和一个 busylist，分配物理页时，就查询 freelist，选最前面的结点对应的物理页框，和页表项建立映射。此处，物理页分配策略是 FIFO。

（2） **你设计中哪些页属于 pinning pages？你实现的页替换策略是怎样的？**

目前设计的页中还没有 pinning pages，这也导致了在下面 bonus 的处理中，会出现换用户栈的情况，然后就一直没有调出来。

页替换的策略是很 naïve 的，利用 FIFO。

3. Bonus 设计

（1）Bonus 中你设计的操作系统通过页表访问的可用物理内存是多少？何时会触发 swap 操作？swap 操作是由专门的进程完成么？

Bonus 中可用物理内存是 16MB 到 32MB 共 16MB 大小。在发生 page fault 时，如果没有可用的物理页框，就需要除法 swap 操作。

（2）你设计的页替换策略是怎样的，有什么优势和不足么？

页替换利用 FIFO，优势就是实现简单，不足就是可能替换高频率使用的页面，导致发生例外次数提高，降低效率。

（3）你设计的测试用例是怎样的？

测试用例：仍然是原本的测试，只是减少物理页框数，设置可用物理页框为 4 个，然后执行任务。

（4）设计或实现过程中遇到的问题和得到的经验（如果有的话可以写下来，不是必需项）

最关键的问题：没有固定页，导致会出现换用户栈的情况，由于 swap 是一个单独的进程，在换页的时候，TLB 表项并没有破坏，页表项和物理页框直接的映射还保持着，这样占用这个页框的进程还能正常运行，在换页完成之后，这个映射消失，这时候如果发生中断，就会导致原本的任务栈没有了，然后发生例外，然后在 swap，导致很严重的后果。

另外，还有一些 bug 没有找出来，进行修改后，会重新检查。

4. 关键函数功能

1. freerange：初始化的时候，用于初始化物理页框，加入 freelist
2. do_TLB_Refill：TLB 例外处理函数