Project 4 Virtual Memory设计文档

中国科学院大学

蔡洛姗

2020/12/14

# 内存管理设计

1. 你设计的页表是几级页表，最大能索引到多大的物理空间？页表项的数据结构是什么？页表本身的数据结构是什么？ 你设计的页表需要几个物理页框保存？

RISCV的SV39模式支持三级页表，最大能索引到4KB一页的空间，不过它也可以调整为二级页表，索引到2MB的空间，我们可用的物理地址范围为0x50000000-0x60000000。RISC-V页表项是直接存放在指定位置的内存中的，以硬件定义好的格式存放。一项占64位，包括Reserved、PPN[2]、PPN[1]、PPN[0]、RSW、D、A、G、U、X、W、R、V位，其中R、W、X 代表可读可写可执行；G代表是否全局映射；A和D代表Access和Dirty，用于表明页面是否被访问或者写入过；。V代表该页表项是否有效。页表本身是占用一块连续物理空间的内存，对于内核和每个用户进程，分别需要页目录的一个页4KB，然后根据需要分配对应的二级页表和三级页表完成映射。

# 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？

由于我在初始化内存时为所有内核地址都建立了虚实映射的页表，在创建用户进程的时候为载入的ELF文件代码以及用户栈对应的一页建立了虚实映射的页表，因此缺页发生的情况可能有：用户进程访问到除此之外的其它用户地址，软件访问的虚拟地址尚未建立虚实地址映射；一个已建立好的虚实映射，但物理页框没有在物理内存中而是被换出到了磁盘上。

缺页处理流程是：首先由于无法在页表中找到对应的物理地址，硬件会触发缺页异常，cause寄存器的值为f。跳入例外处理入口后，根据cause寄存器的异常号跳转到缺页处理函数handle\_page\_fault，触发缺页的地址会被存放在CSR 的staval 寄存器中作为参数传入。然后在这个程序中建立一个从虚拟页面到物理页面的映射并将它加入页表即可。

1. 你使用什么数据结构管理物理页框，管理多少物理页框（例如管理哪些地址范围内的物理页框）？在缺页分配时，按照什么策略或原则进行物理页框分配？

我使用了一个单调递增的指针来指向可用的物理地址0xffffffc051001000 - 0xffffffc060000000，然后用一个链表freelist来存放回收的物理页框。每次缺页分配时，先查看空闲链表中是否有可用的页框，如果有则先分配，如果没有则指针值加上一页大小，将这一页分配出去。

（3）你的设计中是否有pinned的物理页框？若有，具体是保存什么内容的物理页框？

有，用来保存内核页表。

# C-Core设计（做C-Core的同学需要写）

请至少包含以下内容

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

（2）你设计的页替换策略是怎样的，有什么优势和不足么？

（3）你设计的测试用例是怎样的？

（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）

# 关键函数功能

/\* RISC-V 初始化建立内核页表，开启虚存机制 \*/

// using 2MB large page

**void** **map\_page**(**uint64\_t** va, **uint64\_t** pa, PTE \*pgdir)

{

// TODO: map va to pa

**uint64\_t** second\_pgdir = alloc\_page();

**uint64\_t** pgdir\_ppn = (second\_pgdir/4096) << 10;

\*pgdir = pgdir\_ppn | \_PAGE\_PRESENT | \_PAGE\_GLOBAL | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

**uint64\_t** vpn1 = 0;

**uint64\_t** second\_ppn;

**while**(vpn1>=0 && vpn1<=**0x1ff**){

second\_ppn = (pa/4096) << 10;

\*(PTE \*)second\_pgdir = second\_ppn | \_PAGE\_PRESENT | \_PAGE\_READ | \_PAGE\_WRITE | \_PAGE\_EXEC | \_PAGE\_GLOBAL | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

vpn1 += 1;

second\_pgdir += 8;

pa += LARGE\_PAGE\_SIZE; //2MB

}

}

**void** **enable\_vm**()

{

set\_satp(SATP\_MODE\_SV39, 0, PGDIR\_PA/4096);

local\_flush\_tlb\_all();

}

**void** **setup\_vm**()

{

// TODO:

clear\_pgdir(PGDIR\_PA);

//temporary mapping 2MB for 0x50201000;

**uint64\_t** pa = **0x50200000**; //vpn2=1;vpn1=129;

**uintptr\_t** pgdir = PGDIR\_PA+8;

**uint64\_t** second\_pgdir = alloc\_page() + 129\*8;

\*(PTE \*)pgdir = ((second\_pgdir/4096)<<10) | \_PAGE\_PRESENT | \_PAGE\_GLOBAL | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

\*(PTE \*)second\_pgdir = ((pa/4096)<<10) | \_PAGE\_PRESENT | \_PAGE\_READ | \_PAGE\_WRITE | \_PAGE\_EXEC | \_PAGE\_GLOBAL | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

**uint64\_t** va = **0xffffffc000000000**;

pgdir = PGDIR\_PA + **0x800**; //kernel:+2KB

**while**(va<=**0xffffffffffffffff** && va>=**0xffffffc000000000**){

pa = (va << 26) >> 26;

map\_page(va, pa, (PTE \*)pgdir);

va += **0x40000000**;

pgdir += 8;

}

// enable virtual memory

enable\_vm();

}

/\*\*\*\*\*\*\*\*\*\*\* start here \*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**int** **boot\_kernel**(**unsigned** **long** mhartid)

{

**if** (mhartid == 0) {

setup\_vm();

// load kernel

start\_kernel =

(kernel\_entry\_t)load\_elf(\_elf\_main, \_length\_main,

PGDIR\_PA, directmap);

} **else** {

set\_satp(SATP\_MODE\_SV39, 0, PGDIR\_PA/4096);

}

start\_kernel(mhartid);

**return** 0;

}

/\* 按需分配，建立用户页表 \*/

**uintptr\_t** **alloc\_page\_helper**(**uintptr\_t** va, **uintptr\_t** pgdir)

{

**uint64\_t** vpn2 = va >> 30;

**uint64\_t** vpn1 = (va << 34) >> 55;

**uint64\_t** vpn0 = (va << 43) >> 55;

**uint64\_t** offset = (va << 52) >> 52;

**uint64\_t** second\_pgdir;

**uint64\_t** third\_pgdir;

**uint64\_t** pgdir\_entry = \*(PTE \*)(pgdir+vpn2\*8);

**if**((pgdir\_entry%2)==0){ //invalid

second\_pgdir = allocPage(); //virtual addr

**uint64\_t** pgdir\_ppn = ((second\_pgdir-**0xffffffc000000000**)/4096)<<10;

\*(PTE \*)(pgdir+vpn2\*8) = pgdir\_ppn | \_PAGE\_PRESENT | \_PAGE\_USER | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

}**else**{

second\_pgdir = (pgdir\_entry >> 10) \* 4096 + **0xffffffc000000000**;

}

**uint64\_t** pgdir2\_entry = \*(PTE \*)(second\_pgdir+vpn1\*8);

**if**((pgdir2\_entry%2)==0){ //invalid

third\_pgdir = allocPage();

**uint64\_t** second\_ppn = ((third\_pgdir-**0xffffffc000000000**)/4096) << 10;

\*(PTE \*)(second\_pgdir+vpn1\*8) = second\_ppn | \_PAGE\_PRESENT | \_PAGE\_USER | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

}**else**{

third\_pgdir = (pgdir2\_entry >> 10) \* 4096 + **0xffffffc000000000**;

}

**uint64\_t** pa = allocPage();

**uint64\_t** third\_ppn = ((pa-**0xffffffc000000000**)/4096) << 10;

\*(PTE \*)(third\_pgdir+vpn0\*8) = third\_ppn | \_PAGE\_PRESENT | \_PAGE\_USER | \_PAGE\_READ | \_PAGE\_WRITE | \_PAGE\_EXEC | \_PAGE\_ACCESSED | \_PAGE\_DIRTY;

**return** pa;

}