

1. 存储系统的层次化主要是为了提高性价比。由于不同存储技术在技术指标和成本上的差异,存储采用多层的体系,其中最底层为持久化存储层,速度最慢容量最大,成本最低;越往上速度越快,容量越低,成本越高。这样就使得整体成本接近底层,性能接近最上层,容量和持久性等同持久化存储层

2. 过大的页: 当进程只需要使用很小的内存时,使用大页就会造成大量内存空间浪费
同时使页内碎片增加

过小的页: 虽然减少了内存碎片,但造成每进程占用较多的页面,从而导致页表过大,内存利用率降低

3. (1) D-Dirty: ^{为1}~~D为1~~D位表明该页已被写;为0时未被写/不可写若此时仍对该页进行写操作会触发 Page Fault (Store) 异常

A-Accessed: A为1时,表明该页可访问;为0时不可访问,否则触发 Page Fault 访问类型异常

Q-Global, 全局页面标识, ^{Q为1时}当前页面可供多进程共享; 为0时非共享, 进程ASID私有
U-User; U位为1时用户模式可访问; 为0时用户模式不可访问, 否则出现Page fault异常

X W R 分别为可执行、可写、可读标识: 都是1有效, 全为0时, 指针指向下一级页面表

V-Valid: 表明物理页在内存中是否被分配好, V为1时表明已分配好, 访问一个V=0的页面, 则将引发Page Fault异常

(2) 如果用户能自由修改自己的页表, 可能会使一些如用户模式不可访问或者不能写的^{一些}页表产生Page Fault异常

(3) XWR全为0表示指针指向下一级页表

判断当前工作模式下CPU对特定物理地址的可读、可写、可执行权限

4. 例: (1) X: X为1表明表项匹配地址可执行, 为0则不可执行

W: W为1表明PM表项匹配地址可写, 为0则不可写

R: R为1表明表项匹配地址可读, 为0则不可读

(2) L: Lock使能位; 为0则机器模式访问皆成功, 系统/用户模式则由R/W/X判定

为1则表项被锁住, 无法对相关表项进行修改

当配置TOR模式, 其前一个表项的地址寄存器也无法修改
所有模式根据R/W/X判定是否可访问

A: 表项的地址匹配模式: 00: OFF, 无效表项; 01: TOR, 使用相邻表项地址作为匹配区间

10: NAT, 区间大小为字节的匹配; 11: MAPOT, 区间大小为页匹配, 至少

4KB

5. 例: (1) 页内索引占12位, 剩5位, 所以需要 $2^{12} \times 8$ 字节, 即 2^{15} 字节空间

(2) 页内索引占12位, 剩36位, 所以要 $2^{36} \times 8 = 2^{39}$ 字节空间

(3) 因为多级表是一对多的关系, 每一级^例索引出的内容用来索引下一级的基址, 所以省去了大量未映射的页表项, 只保留如第一级表, 1个第二级或第三级在内存中, 从而大幅减小了空间