

1. 现代计算机系统需要存储层级的原因:

① 为了提高性价比减少成本, CPU内存, 存储容量很小. 基本以KB级别存储, MB级别的CPU内存比较少见, 出于经济成本的考虑.

② 提高存储器的访问速度和容量, 通过缓存等层级存储的设置, 加快CPU对存储器中一些高频访问的指令和数据访问速度.

2. 页式虚拟存储中, 页过大导致:

存储层级设置可以降低成本, 将占字节大的存储部分单独独立出来使得成本降低, 存储容量提升.

① 由页过大, 每一页大小和对应物理地址相同. 存入cache后占用内存巨大.

② 需要大片大片的连续内存空间与之建立起映射关系, 很奢侈.
条件

页过小导致: ① 页表规模增大, 占用更多的内存空间

② 页表规模增大, 页表查找时间增加.

3. 1) 位0 'V' 比特表示 PTE 是否有效. 如果 $V=0$, 就不需要关心 PTE 的其他位, 可以被软件自由使用

位 'R' 'W' 'X' 为权限相关的比特位, 分别指明 page 是否可读, 读写, 可执行

当 X/W/R 都为 0 时, 表示此 PTE 指向下一级的 page table, 否则它就是 leaf PTE, 可写页面也必须标记为可读的.

位 'U' 表示 u-mode 是否允许访问该页面.

位 'G' 表示一个全局映射

位 'A' 和位 'D': 每个 leaf page 包含 A 和 D 比特位

A 比特位表示自上次清除后, 虚拟 page 已经被读取. 写入或 fetched.

D 比特位表示自上次清除后, 虚拟 page 已经被写入.

2) 如果用户进程能够自由修改自己的页表, 可能会发生页表中那 8 位关于权限设置和 leaf page 所属状态的标记各位数受到篡改, 可能会导致无法预料的权限异常和状态异常, 产生错误.

3) 一个 X/W/R 位全部为 0 的有效页表条目有什么意义?

表示此 PTE 指向下一级的 page table, 否则就是 leaf PTE, (可写页面也必须标记为可读的)

4. 1) PMP 控制寄存器中的 X/W/R 位也为对应的执行、写、读的权限. 为 1 时权限存在, 0 时无权限. 对于 $R=0, W=1$ 情况与实际不符, 该权限对应的是文件权限, 限制某地址范围内的代码执行, 读、写操作. 作为保留, 以便未来某些状况下使用

2) A: 地址匹配模式, 限制访问某个地址范围

L: 锁定模式, 锁定某个 PMP 入口, 使其不可修改.

5. 1) 一个页的大小为 $2^{12}B$, 每个页表条目使用 8 字节空间. 内存系统按字节寻址,

则有 $\frac{2^{64}}{2^{12}} = 2^{52}$ 个表项, 每个页表条目占 2^3 字节空间

共需要 $2^{52} \cdot 2^3 = 2^{55}$ 字节的存储空间用于存储页表

2) 48 位虚拟地址空间时有 $\frac{2^{48}}{2^{12}} = 2^{36}$ 个表项, 每个页表条目占 8 字节空间,

则空间为 2^{39} 字节
所需

3) ① 多级页表可以使页表在内存中离散存储, 通过增加索引来定位到具体的项

② 节省页表内存. 通过不为进程实际使用的那些虚拟地址内存区请求页表来减少内存使用量. 无需一次性开辟一大块连续内存空间存放页表.