

1. ① 访问速度: 不同存储介质的访问速度存在差异. 寄存器和缓存是速度快的存储设备, 但容量小, 而内存和磁盘的容量较大但访问速度较慢. 使用多个层级的存储设备, 可根据数据的访问频率和速度要求, 将数据存储在适当的层级中, 提高对常用数据访问能力.
- ② 成本: 高速存储设备成本较高, 而容量大的存储设备成本相对较低. 保证性能的同时, 降低了成本.
- ③ 容量需求: 现代计算机处理的数据量大, 需要更大的存储量. 使用多个层级的存储设备, 可提供足够的容量满足不同的应用需求.
- ④ 可扩展性: 存储层级使得计算机系统存储容量可以根据需要进行扩展. 例如, 可以通过增加缓存的大小或添加更多的磁盘来扩展存储容量.

2. 过大: 1. 进程只用到存储较小数据块, 每个页中还存在未被利用的空间, 占用内存空间, 降低内存利用率.

2. 每个进程所需页表项数量也会增加, 页表大小增加, 增加了页表维护开销.

过小: 1. 数据被分散存放不同页中, 造成大量外部碎片, 降低内存利用率.

2. 所需页表项数量增加, 进而增加了页表维护开销.

3.



3. (1) 位0: 有效位. 用于指示页表条目是否有效.

位1: 读权限位. 指示对于该页表条目所映射的页面是否有读权限

位2: 写权限位. 对于该页表条目所映射的页面是否有写权限

位3: 执行权限位. 所映射的页面是否有执行权限

位4: 用户位. 页面是否为用户态可访问

位5: 全局位. 用于指示该页表条目是否为全局映射.

位6: 访问位. 指示所映射的页面是否被访问过

位7: 脏位. 指示页面是否被写入过.

(2) ① 用户进程可通过修改页表来越权访问其他进程的内存空间, 导致数据泄漏或篡改

② 导致内存管理程序错误和崩溃

③ 内存管理的混乱, 内存分配和释放出现问题.

④ 共享资源冲突.

(3) 意味着该页表不可执行代码, 不可写入数据, 也无法读取其中内容.

4. (1) 进一步细化对物理内存的访问权限控制, 确保只有被授权的程序可以执行, 读取或写入特定物理内存区域.

(2) L位: Lock位. 控制对于PMP寄存器写入权限

A位: Address位. 确定PMP寄存器地址范围



5. (1)  $2^{64}$  个可能的虚拟地址.

$$\text{则 } 8 \times \frac{2^{64}}{4 \times 1024} = 2^{55} \text{ Byte}$$

需要  $2^{55}$  个字节.

(2) 若限制仅使用 48 位虚拟地址空间.

$$8 \times \frac{2^{48}}{4 \times 1024} = 2^{39} \text{ Byte}$$

需要  $2^{39}$  个字节

- (3)
1. 减小页表的大小, 在大型内存系统中, 页表的大小可能非常庞大, 多级页表可以通过层级结构来组织页表. 将大的页表拆分成多个小的页表,
  2. 惰性加载. 采用分级加载. 程序运行中, 当页面真正被访问时才会加载相应的页表项. 意味着只有访问的页面需要加载到内存中. 节省存储开销, 加速访问速度.
  3. 局部性原理. 程序执行过程中, 可能只会访问到一部分连续虚拟页面. 通过多级页表, 使得常用的页面在更高级的页表层次中, 而不常用页面则在最低级页表层次中. 可减少整体页表大小, 提高内存访问效率.