

1 现代计算机系统需要存储层级的几个原因：

访问速度和容量需求：不同类型的存储器具有不同的访问速度和容量。寄存器是最快的存储器，但容量非常有限；高速缓存比主存快，但容量也较小；主存比辅助存储器快，但容量有限。通过在存储层级中使用不同速度和容量的存储器，可以满足不同的访问速度和容量需求。

成本效益：高速存储器通常比较昂贵，而辅助存储器则相对便宜。通过在存储层级中使用不同成本的存储器，可以在满足性能需求的同时降低成本。

局部性原理：在大多数计算任务中，存在着数据和指令的局部性原理。即计算机倾向于频繁地访问相邻的内存位置。因此，通过将最常用的数据和指令存储在较快的存储器中，可以提高系统的整体性能。

内存层次转换：存储层级还可以用于管理内存层次转换。当处理器需要访问数据或指令时，它首先查找最高层级的存储器。如果所需的数据不在当前层级中，就会向下一级存储器发出请求。这种层级结构可以减少处理器与辅助存储器之间的直接交互次数，从而提高系统效率。

2 过大的页大小：

内部碎片：过大的页大小可能导致内部碎片的问题。如果一个进程只需要访问页面中的一小部分数据，但整个页面都被加载到内存中，那么在页面内部会存在未使用的空间，称为内部碎片。

上下文切换开销：如果页大小很大，每次进行进程切换时需要保存和加载更多的页表项和页表，这会增加上下文切换的开销。

内存利用率下降：过大的页大小可能导致内存利用率下降。

过小的页大小：

外部碎片：过小的页大小可能导致外部碎片的问题。如果一个进程需要的内存大小超过了一个页的大小，但没有足够连续的空闲页可供分配，会导致外部碎片。这可能导致内存分配不连续，造成内存空间的浪费。

页表大小增加：如果页大小太小，系统需要为每个进程维护更多的页表项和页表，以管理虚拟地址和物理地址之间的映射关系。这会增加页表的大小和内存开销。

I/O 开销增加：过小的页大小可能会增加 I/O 开销。

3

1) 在 RISC-V 规范中, 位 7 至位 0 的功能如下:

位 0 (V): 有效位 (Valid), 指示该页表条目是否有效。如果该位为 1, 表示该条目有效; 如果为 0, 表示该条目无效。

位 1 (R): 读取权限 (Read), 指示对于该页的读取访问权限。如果该位为 1, 表示允许读取; 如果为 0, 表示禁止读取。

位 2 (W): 写入权限 (Write), 指示对于该页的写入访问权限。如果该位为 1, 表示允许写入; 如果为 0, 表示禁止写入。

位 3 (X): 执行权限 (Execute), 指示对于该页的执行访问权限。如果该位为 1, 表示允许执行; 如果为 0, 表示禁止执行。

位 4 (U): 用户模式权限 (User mode), 指示对于该页的用户模式访问权限。如果该位为 1, 表示用户模式可访问; 如果为 0, 表示用户模式禁止访问。

位 5 (G): 全局页 (Global), 表示该页在所有地址空间中是全局共享的。

位 6 (A): 访问位 (Accessed), 用于标识是否访问过该页。当处理器访问该页时, 硬件会将该位设置为 1, 以跟踪页面的访问情况。这可以用于页面置换算法或者性能统计等目的。

位 7 (D): 脏位 (Dirty), 用于标识页面是否被修改过。当处理器对该页进行写入操作时, 硬件会将该位设置为 1, 表示页面的内容已被修改。这可以用于页面置换算法或者写回操作等目的。

2) 如果用户进程能够自由修改自己的页表, 会导致安全性和稳定性问题。用户进程可能会修改页表中的权限位 (如读取、写入、执行权限), 使其可以访问其没有权限访问的内存区域。这可能导致未经授权的访问、数据损坏、信息泄露等安全问题。此外, 用户进程还可以修改有效位 (V) 来使无效的页表条目有效, 进而访问无效的页面。这可能导致页面不存在或者已被其他进程使用, 进而引发错误或冲突。

3) 在 RISC-V 的虚拟内存管理中, 一个 x/W/R 位全部为 0 的有效页表条目表示该页不可被访问 (禁止读取、写入和执行)。这样的页表条目可能表示一个未分配的虚拟页面, 或者是已分配但没有被加载到物理内存的页面。在访问这样的页时, 将会触发页错误异常, 并由操作系统来处理。

4

1) 在页表条目中已经存在 x/W/R 位的情况下, PMP 控制寄存器中的 X/W/R 位用于进一步限制对物理内存区域的访问权限。页表条目中的 x/W/R 位控制虚拟内存中的访问权限, 而 PMP 控制寄存器中的 X/W/R 位则控制对应的物理内存区域的访问权限。这样可以提供额外的安全性, 确保对物理内存的访问与虚拟内存中的访问权限相符合。

2) PMP 配置寄存器中的 L 位 (Lock) 和 A 位 (Address Matching) 有以下作用:

L 位: 当 L 位被设置为 1 时, 表示该 PMP 配置寄存器中的访问权限是锁定的, 不可被软件修改。这提供了一种机制, 使得对特定物理内存区域的访问权限无法被更改, 增加了系统的安全性和稳定性。

3) A 位: A 位用于指定 PMP 配置寄存器是否进行地址匹配。当 A 位被设置为 1 时, 表示该 PMP 配置寄存器的地址范围会与访问的物理地址进行匹配。如果匹配成功, 就会根据 PMP 控制寄存器中的权限位 (如 X/W/R 位) 进行访问权限的判断。如果 A 位被设置为 0, 表示该 PMP 配置寄存器将被忽略, 即不进行地址匹配, 直接根据其他配置寄存器的权限位进行访问权限的判断。

5

如果页大小为 4KB，每个页表条目使用 8 字节空间，内存系统按字节寻址，那么一个单级页表系统需要多大的空间用于存储页表？

$4\text{KB} / 8\text{B} = 512$ 个页表条目的地址

每个页表条目可以映射 $4\text{KB} / 8\text{B} = 512$ 个页面。

总的页表条目数 = 虚拟地址空间大小 / 页大小 = $(2^{64}) / (4\text{KB}) = 2^{52}$

$2^{52} * 8\text{B} = 2^{55}$ 字节

2)

$4\text{KB} / 8\text{B} = 512$ 个页面。

总的页表条目数 = 虚拟地址空间大小 / 页大小 = $(2^{48}) / (4\text{KB}) = 2^{36}$

一个单级页表系统需要的空间为 $2^{36} * 8\text{B} = 2^{39}$ 字节

3) 多级页表可以降低虚拟内存系统的实际页表存储开销的原因如下：

节省空间：多级页表将大的页表分割成更小的页表，只有在需要的时候才分配内存空间，从而节省了存储空间。只有被映射到物理内存的页表才会被加载，而不是一次性加载整个页表。

层次结构：多级页表使用层次结构，使得查找和管理页表更加高效。

惰性加载：多级页表支持惰性加载机制，只有当访问到相应的虚拟页时，才会进行页表的加载和映射。这样可以避免一次性加载整个页表，节省了内存和加载时间。