《页面置换算法（下）》微课脚本设计

|  |  |
| --- | --- |
| 本节微课名称 | 页面置换算法（下） |
| 知识点描述 | 页面置换算法的相关原理 |
| 知识点来源 | 《现代操作系统》第三章 第4节 |
| 教学类型 | 讲授-启发-讨论-问答 |
| 设计思路 | 对于大部分学生来说如果单是通过对书本上的内容来对该章节的知识进行学习的话会觉得有些枯燥乏味。而如果是通过看视频的方式，能更灵活直观地表述出需要表达的内容，通过文字、图片和动画特效等，更加能够获取学生的兴趣，从而可以对页面置换算法的知识点有更深入的了解与认知。 |

教学过程：

|  |  |  |  |
| --- | --- | --- | --- |
|  | 内容 | 画面 | 时间 |
| 一、片头 | 老师出场，介绍本节课知识点：   1. 最近最少使用页面置换算法 2. 用软件模拟LRU | 同学们好，今天我给大家带来的是现代操作系统原理课的“页面置换算法（下）”，这节课我们将学习以下内容（列出本次微课的知识点提纲） |  |
| 二、正文讲解 | 1.最近最少使用页面置换算法  引出算法；设计思想；性能好，开销大；算法的一种硬件实现；动画演示算法的运作过程 | 对最优算法的一个很好的近似是基于这样的观察：在前面几条指令中频繁使用的页面很可能在后面的几条指令中还会被使用。反过来说，已经很久没有使用的页面很有可能在未来较长的一段时间内仍然不会被使用。这个思想提示了一个可实现的算法：最近最少使用页面置换算法（LRU）。（引出算法）  它的设计思想是：选择最后一次访问时间距离当前时间最长的一页并置换，即置换未使用时间最长的一页。（设计思想）  这个算法的性能虽然是最接近最优算法的，但是实现起来会带来很大的开销。为了完全实现这个算法，需要在内存中维护一个所有页面的链表，最近最多使用的页面放在表头，最近最少使用的页面放在表尾。困难的是在每次访问内存时都必须要更新整个链表。在链表中找到一个页面，删除它，然后把它移动到表头是一个非常费时的操作，即使使用硬件实现也一样费时。（性能好，开销大）  然而，还是有一些使用特殊硬件实现这个算法的方法。下面将介绍最近最少使用算法的一种硬件实现，当然这个方法首先需要硬件的支持。它的思想是：给在页框中的所有页面设置一个矩阵，当某一个页面被访问时，就把这个页面在矩阵中对应的一行设置为1，再把对应的一列设置为0。假设页面访问顺序为：0 1 2 3 2 1 0 3 2 3。当系统访问0号页面时，将对应的一行设置成1，再将对应的一列设置成0。当系统访问1号页面时，将对应的一行设置成1，再将对应的一列设置成0。当所有页面都被设置完成后，如果这时需要进行页面置换，则淘汰在对应的一行中值最小的页面，这个页面就是最近最少使用的页面。所以1号页面将被淘汰。（算法的一种硬件实现）  下面我们通过一个例子来演示最近最少使用页面置换算法的运作过程和计算运作过程中的缺页次数。（动画演示算法的运作过程）    系统给某进程分配3个页框，初始为空。  进程执行时，页面走向序列为：2 3 2 1 5 2 4 5 3 2 5 2  系统首先访问2号页面，缺页一次，将2号页面调入页框。  然后访问3号页面，缺页一次，将3号页面调入页框。  然后访问2号页面，没有缺页，无需操作。  然后访问1号页面，缺页一次，将1号页面调入页框。  然后访问5号页面，缺页一次，由于页框中3号页面最后一次访问时间距离当前时间最长，所以淘汰3号页面并将5号页面调入页框。  然后访问2号页面，没有缺页，无需操作。  然后访问4号页面，缺页一次，由于页框中1号页面最后一次访问时间距离当前时间最长，所以淘汰1号页面并将4号页面调入页框。  然后访问5号页面，没有缺页，无需操作。  然后访问3号页面，缺页一次，由于页框中2号页面最后一次访问时间距离当前时间最长，所以淘汰2号页面并将3号页面调入页框。  然后访问2号页面，缺页一次，由于页框中4号页面最后一次访问时间距离当前时间最长，所以淘汰4号页面并将2号页面调入页框。  然后访问5号页面，没有缺页，无需操作。  然后访问2号页面，没有缺页，无需操作。  所以进程运用最近最少使用页面置换算法在运作过程中一共有7次缺页。 |  |
|  | 2.用软件模拟LRU  引出用软件模拟LRU；NFU设计思想和具体做法；NFU的主要问题；AGING介绍；AGING如何工作；AGING和LRU的第一个区别；AGING和LRU的第二个区别 | 最近最少使用算法虽然在理论上可以使用一些特殊硬件来实现，但是只有非常少的计算机拥有这种硬件。因此，需要一个能用软件实现的解决方案。（引出用软件模拟LRU）  有一种可能的方案称为最不常用页面置换算法（NFU）。它的设计思想是：选择访问次数最少的页面进行置换。该算法将每个页面与一个软件计数器相关联，计数器的初值为0。每次时钟中断时，操作系统扫描内存中所有的页面，将每个页面的R位的值加到它的计数器上，这个值可能是0或1。当发生缺页中断时，则置换计数器值最小的页面。（NFU设计思想和具体做法）  最不常用算法的主要问题是它从来不忘记任何事情。假设有这样一种情况：内存中有页面A和页面B两个页面，页面A在经过四次扫描后的计数器的值为1 1 0 0，而页面B在经过四次扫描后的计数器的值为0 0 1 1。虽然在这四次扫描中这两个页面的访问次数相同，且页面A的未使用时间比页面B更长，但是页面A的计数器的值比页面B更高。如果这时发生缺页中断需要淘汰一个页面，则页面B将会被淘汰。因此，这个算法可能会导致操作系统置换有用的页面而不是最近未使用的页面。（NFU的主要问题）  幸运的是只需对最不常用算法做一个小小的修改就能使它很好地模拟最近最少使用算法。其修改分为两部分：首先，在R位被加进之前先将计数器右移一位；其次，将R位加到计数器的最左端的位而不是最右端的位。修改以后的算法称为老化页面置换算法（AGING），下面将解释它是如何工作的。（AGING介绍）  假设在第一个时钟滴答后，页面0~5的R位值分别为1 0 1 0 1 1。换句话说，在时钟滴答0到时钟滴答1期间，访问了页面0、2、4、5，它们的R位设置为1，而其他页面的R位仍然是0。对应的6个计数器在经过移位并把R位插入其左端后的值如图a所示，图中后面的4列是在下4个时钟滴答后的6个计数器的值。当某时刻发生缺页中断时，将置换计数器值最小的页面。（AGING如何工作）  该算法与最近最少使用算法有两个区别。如图e中的页面3和页面5，它们都是连续两个时钟滴答没有被访问过了，而在两个时钟滴答之前的时钟滴答中它们都被访问过。根据最近最少使用算法，如果必须置换一个页面，则应该在这两个页面中选择一个。然而现在的问题是，我们不知道在时钟滴答1到时钟滴答2期间它们中的哪一个页面是后被访问到的。因为在每个时钟滴答中只记录了一位，所以无法区分在一个时钟滴答中哪个页面在较早的时间被访问以及哪个页面在较晚的时间被访问，因此，我们能做的就是置换页面3，原因是页面5在更往前的两个时钟滴答中也被访问过而页面3没有。（AGING与LRU的第一个区别）  第二个区别是老化算法的计数器只有有限位数，在本例中是8位，这就限制了其对以往页面的记录，如果两个页面的计数器都是0，我们只能在这两个页面中随机选一个进行置换。实际上，有可能其中一个页面上次被访问是在9个时钟滴答以前，另一个页面是在1000个时钟滴答以前，而我们却无法看到这些。但是在实践中，如果一个时钟滴答是20ms，那么8位一般是够用的，假如一个页面已经有160ms没有被访问过了，那么这个页面很可能并不重要。（AGING与LRU的第二个区别） |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  | 以上就是本节课的全部内容了，同学们掌握好这节课的知识了吗？下面我们来做几道题巩固一下吧。 |  |
| 三、结尾 | 本节课结束 通过几道小问题回顾今天所学知识 | 转到互动答题 |  |