多级反馈队列调度算法
各种调度算法比较
多处理器调度算法

下面我们首先介绍，多级反馈队列调度算法 然后对前面介绍的各种调度算法进行比较

多级反馈队列调度算法(1/3)

- Multilevel Feedback
- 是UNIX的一个分支BSD（加州大学伯克利分校开发和发布的）5.3版所采用的调度算法
- 是一个综合调度算法

折衷权衡
tradeoff

因素之后进行折中权衡的一个结果 下面我们介绍

多级反馈队列调度算法(2/3)

- 设置多个就绪队列，第一级队列优先级最高
- 给不同就绪队列中的进程分配长度不同的时间片，第一级队列时间片最小；随着队列优先级别的降低，时间片增大
- 当第一级队列为空时，在第二级队列调度，以此类推
- 各级队列按照*时间片轮转方式*进行调度
- 当一个新创建进程就绪后，进入第一级队列
- 进程用完时间片而放弃CPU，进入下一级就绪队列
- 由于阻塞而放弃CPU的进程进入相应的等待队列，一旦等待的事件发生，该进程回到原来一级就绪队列（？）

队首 or 队尾？

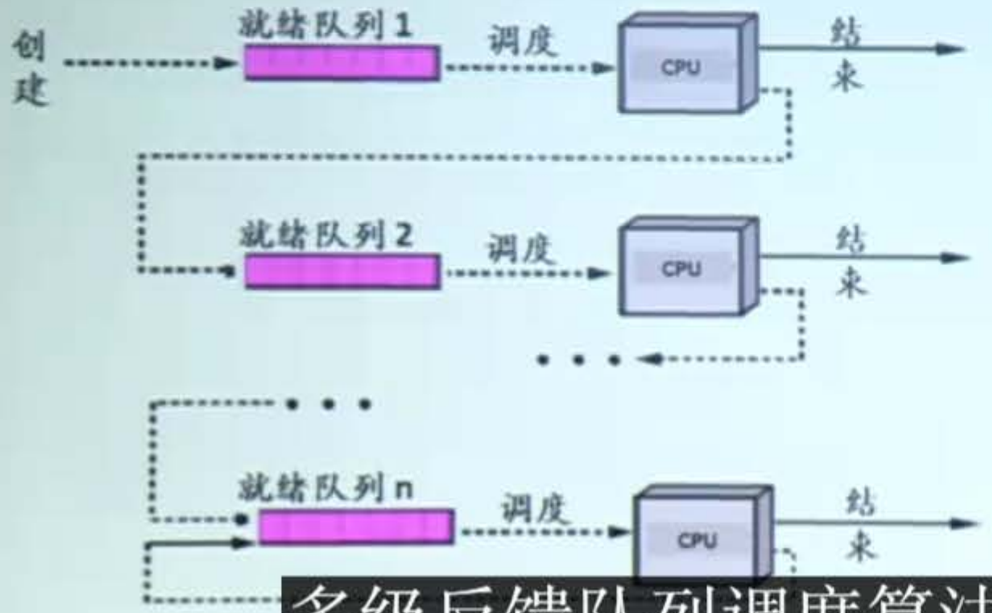
再次被调度上CPU时对时间片的处理？

非抢占式

多级反馈队列调度算法(3/3)

若允许抢占

- 当有一个优先级更高的进程就绪时，可以抢占CPU
  被抢占的进程回到原来一级就绪队列末尾（或者？）



队列模型示意

多级反馈队列调度算法对 I/O 型进程更偏好一点

小结：各种调度算法的比较

调度算法	占用CPU方式	吞吐量	响应时间	开销	对进程的影响	饥饿问题
FCFS	非抢占	不强调	可能很慢，特别是当进程的执行时间差别很大时	最小	对短进程不利；对I/O型的进程不利	无
Round Robin	抢占(时间片用完时)	若时间片小，吞吐量会很低	为短进程提供好的响应时间	最小	公平对待	无
SJF	非抢占	高	为短进程提供好的响应时间	可能较大	对长进程不利	可能
SRTN	抢占(到达时)	高	提供好的响应时间	可能较大	对长进程不利	可能
HRRN	非抢占	高	提供好的响应时间	可能较大	很好的平衡	无
Feedback	抢占(时间片用完时)	不强调	不强调	可能较大	对I/O型进程有利	可能

多处理器调度算法需要考虑的问题

我们简要介绍一下在设计多处理器调度算法时 需要考虑的几个问题

多处理器调度算法设计

- 不仅要决定选择哪一个进程执行
  - 还需要决定在哪一个CPU上执行
- 要考虑进程在多个CPU之间迁移时的开销
  - 高速缓存失效、TLB失效
  - 尽可能使进程总是在同一个CPU上执行
    - 如果每个进程可以调度到所有CPU上，假如进程上次在CPU1上执行，本次被调度到CPU2，则会增加高速缓存失效、TLB失效；如果每个进程尽量调度到指定的CPU上，各种失效就会减少
- 考虑负载均衡问题

都保持忙碌，那么这就 是一个负载均衡的问题

0:00 下面我们首先介绍，多级反馈队列调度算法 然后对前面介绍的各种调度算法进行比较 之后呢，我们简单讨论一下 在设计多处理器调度算法时所要考虑的几个问题 多级反馈队列调度算法 是 UNIX 的一个分支，BSD 5.3 版所采用的调度算法 它是在前面各种调度算法的基础之上 提出的一个综合的调度算法，是在考虑了各种 因素之后进行折中权衡的一个结果 下面我们介绍 一下多级反馈队列调度算法的基本思想 就绪队列设置成多个 其中第一级队列的优先级最高 依次从高到低，系统 给不同的就绪队列分配了长度不同的时间片 第一级队列优先级最高 但分配给它的时间片最小 随着队列优先级的不断降低 分配给队列的时间片就越大 比如说第一级队列分配给它一个单位的话 第二级队列就可以分配成两倍的时间片 那么第三级可以分配四倍的时间片，这就是 级别越低，时间片越大 在进行调度的时候 首先从优先级高的队列进行调度 如果第一级队列没有进程了 那么系统会从第二级队列进行调度，以此类推 每一个队列都是按照时间片轮转的方式进行调度 新创建的进程 就绪的时候呢都进入第一级队列 如果一个被调度上 CPU 的进程用完了时间片 而放弃了 CPU，那么它就进入下一集就绪队列 也就是说，它被降级了 那么如果一个被调度上 CPU 的进程 由于等待 I/O 而被阻塞，进入了等待队列 当等待的事件发生后，这个进程从等待队列 回到原来一级就绪队列。 那么这里头 我们可以根据不同的情况来设计不同的方案 以体现系统对这一类进程的偏好程度，比如说 这个进程是回到原来一级就绪队列的队首呢 还是队尾？如果回到队首，说明系统对这类进程更加友好 另外，当这个进程再度被调度上 CPU 之后 是让它运行完剩余的时间片呢 还是重新给它分配一个完整的时间片让它去运行？ 也体现了系统对这类进程的偏好程度 那么我们现在所看到的 多级反馈队列调度算法呢是一个非抢占式的 如果允许抢占呢 也就是说当有一个更高优先级的进程就绪的时候 可以抢占正在运行进程的 CPU 那么被抢占的进程呢 会回到原来一级就绪队列的末尾 当然也可以有不同的设计方案 比如说回到原来一级就绪队列的队首 当这个进程再度被调度上 CPU 时呢 可以运行完它刚才剩余的时间片 也可以重新给它一个完整的新的时间片让它运行 因此呢又派生出不同的设计方案 下面我们来看一下这张图 它反映了一个进程在 队列里头的一些迁移活动 当创建一个新的进程时 所有的进程都进入第一级队列 如果是 I/O 型的进程，那么它可能 被调度上 CPU 之后很短时间就去等待 I/O 当它从等待队列又回到就绪队列的时候 由于我们让它回到原来一级就绪队列，所以它呢优先级没有降低 被调度上 CPU

的机会很多。 但是对于 CPU 型的进程 它被调度上 CPU，用完了一个时间片之后 它就会回到下一级队列 那么如果每次都用完了它的时间片，它就会降级 可能一个 CPU 型的进程就慢慢降到了 优先级最低的这个队列里头，因此我们可以看到 通过这样一个调度算法 就可以慢慢地区分出来哪些进程是 CPU 型进程 哪些进程是 I/O 型进程，很显然 多级反馈队列调度算法对 I/O 型进程更偏好一点 对 CPU 型进程呢不太有利 但是呢它也做了一些弥补，比如说 优先级高的队列时间片短 而优先级低的队列时间片会很大 所以当低优先级的 CPU 型进程被调度上 CPU 之后，它可以运行更长的时间 那这里呢也是一种平衡的结果。下面 我们对前面介绍的各种调度算法 做一下小结。 在占用 CPU 的方式上 先来先服务，短作业优先 最高响应比优先调度算法是非抢占式的调度策略 最短剩余时间优先 则是一个抢占式的调度策略，而时间片轮转 多级反馈队列调度算法 则是允许在时间片到的时候可以进行抢占 在追求调度算法指标上 我们来看一下短作业优先 最短剩余时间优先和最高响应比优先 这三个调度算法都可以带来比较高的吞吐量 而 时间片轮转调度算法 如果时间片很小，那么它的吞吐量就很低 先来先服务调度算法 还有多级反馈队列调度算法对吞吐量这个指标 并不强调，在响应时间方面 时间片轮转，短作业优先两个调度算法 对短的作业可以提供很快的响应时间 最短剩余时间优先 和最高响应比优先调度算法呢，也可以提供很好的响应时间 而对于先来先服务调度算法 当不同的进程 它们的时间有很大差别的时候，对于某些进程 它的响应时间会很慢，下面我们来看一下 调度算法本身所带来的开销 先来先服务和时间片轮转调度算法 因为实现比较简单，所以开销比较小 而其它四种调度算法它们的开销可能很大 比如说最高响应比优先调度算法 它要计算每一个进程的响应比 才能选择响应比最高的那个进程。 所以计算响应比需要花时间 多级反馈队列调度算法需要维护多个就绪队列，这也需要花时间 接着我们来看一下不同的调度算法 对进程的影响。 时间片轮转调度算法 公平地对待每一个进程。 最高响应比优先调度算法 则是在先就绪的进程和短进程之间做了一个很好的平衡 而来先来先服务调度算法 对长进程之后的短进程，或者是 I/O 型进程 是不利的。 那么最

短作业优先 和最短剩余时间优先的调度算法对长进程不利 会导致长进程产生饥饿现象 多级反馈队列调度算法 对于 I/O 型进程是偏好的，也就是说对 I/O 型进程有利 而对 CPU 型进程不利 会导致 CPU 型进程产生饥饿 所以没有一种调度策略是完美的 也不可能有一种调度策略对各种类型的进程都能够照顾到 因此在设计调度策略的时候，我们应该 考虑综合性的因素，下面 我们简要介绍一下在设计多处理器调度算法时需要考虑的几个问题 在设计多处理器调度算法时 我们不仅要决定选择哪一个进程执行 而且还要确定这个被选中的进程在哪一个 CPU 上执行 另外我们必须要考虑 进程在多个 CPU 之间迁移时所带来的高速缓存 TLB 失效的开销 如果一个进程 之前在 CPU1 上执行，后来又被调度到了 CPU2 上执行 那么高速缓存 TLB 失效就会增加 而如果这个进程每次都被指定到 同一个 CPU 上执行，那么就会减少各种 失效。 因此应该尽可能地使进程总在同一个 CPU 上执行 另外呢，我们还应该考虑到一个负载均衡的问题 因为有多个 CPU，那么不可能让某些 CPU 非常地忙碌 而其它 CPU 很空闲，所以 我们要通过调度使得 所有的 CPU 都保持忙碌，那么这就 是一个负载均衡的问题