相关概念

- 博主参考的详细文章
- **CPU利用率** CPU工作时间 / 总时间
- **吞吐量** 单位时间内完成进程的数量 完成作业量 / 总时间
- **周转时间** 进程到达->进程完成 (作业完成时间点-作业提交时间点)
    - 平均周转时间
    - **平均带权周转时间**(主要用于表征满意度) 周转时间 / 实际进行时间, 显然, 这 ≥ 1
- **响应时间** 单个进程到达->被处理
- **等待时间** 所有进程的相应时间的总和
    - 平均等待时间
- **抢占式调度**
    - 当前运行的进程可能被系统中断, 转为就绪态
- **非抢占式调度**
    - only进程主动释放CPU

分级调度

> 三个调度：
> • 长程调度决定把后备队列中的哪些作业调入内存，创建进程并加入相应的队列中
> • 中程调度按一定的算法将“外存”中已具备条件的进程换入内存中，而将内存中处于阻塞状态的某些进程换出至外存
> • 短程调度决定从就绪队列中哪一个进程应先获得处理器，并将处理机分配给选定的进程

长程调度是“作业-进程”之间的调度，中程调度是“挂起-激活”之间的调度，短程调度是“就绪-运行”之间的调度

长程调度

长程调度（也叫高级调度、作业调度）

- 长程调度决定哪一个程序何时可以进入到系统中处理
- 决策1：何时创建一个新进程。由要求的系统并发度驱动。创建的进程越多，每个进程可以执行的时间百分比就越小
- 决策2：加入哪一个新进程。基于简单的先来先服务原则、基于管理的系统性能的工具等（优先级、期待执行时间和I/O需求）
- 执行的频率最低

中程调度

中程调度（也叫中级调度）

- 为提高系统吞吐量和内存利用率而引入的内外存对换功能（换出　时，进程为挂起状态），主要涉及内存管理与扩充
- 将进程的部分或全部加载到内存中
- 换入决策基于管理多道程序并发程度的要求
- 执行的频率比长程调度要频繁些

短程调度

短程调度（也叫低级调度、进程调度）

- 执行得最频繁，要求在实现时达到高效率
- 短程调度程序也称作分派程序
- 主要任务：按照某种策略和方法选取一个处于就绪状态的进程占用处理机，包括
  - 保存处理机现场信息
  - 按某种算法选取进程
  - 把处理机分配给进程
- 短程调度的三个基本机制：排队器、分派器、上下文切换机制

引起调度的事件

1. 时钟中断（例如时间片用完）
2. I/O中断
3. 操作系统调用
4. 信号（例如在信号量上的wait操作，使进程阻塞）
5. 抢占方式下，就绪队列中出现某更高优先权的进程

常见的调度算法

详细参考

先来先服务(FCFS)

- 按照进程就绪的先后顺序使用CPU
- **特点**：公平，实现简单(FIFO队列)。
- **护航效果:** 长进程后面的短进程需要等待很长时间，不利于用户体验

优先级调度

- 静态优先级
  - 优先级确定后, 不会再变化
- 动态优先级
  - 优先级与占有CPU时间的长短成反比, 与在就绪队列中等待的时间长短成正比
  - 按理说, 等待很久的进程的优先级会慢慢升高, 确保大概率不会出现饥饿
  - 优先级数字越小, 意味着优先级越高, 越先被调度

    给每个作业一个优先级，优先级越高越紧迫，应该先执行。

    通常：系统进程优先级高于用户进程；前台进程优先级高于后台进程；操作系统更偏好 I/O型进程。

    特点：实现简单，但不公平，可能导致优先级低的进程产生饥饿现象；

    可能产生优先级反转问题（基于优先级的抢占式算法），即一个低优先级进程持有一个高优先级进程所需要的资源，使得高优先级进程等待低优先级进程运行。

最短作业优先(SJF)

- 非抢占式
  - 具有最短完成时间的进程优先执行，非抢占
  - 如果时间相同, 那么按FCFS, 先来先服务
  - **特点:**吞吐量提高, 但是对长进程不利, 并且事先知道执行时间很困难
  - 常用于长程调度, 不常用于进程调度
- 抢占式(最短剩余时间优先)
  - 当一个新就绪的进程比当前运行进程具有更短完成时间时，系统抢占当前进程，选择新就绪的进程执行
  - 有最短的平均周转时间，但不公平，源源不断的短任务到来，可能使长的任务长时间得不到运行，从而产生 “饥饿”现象

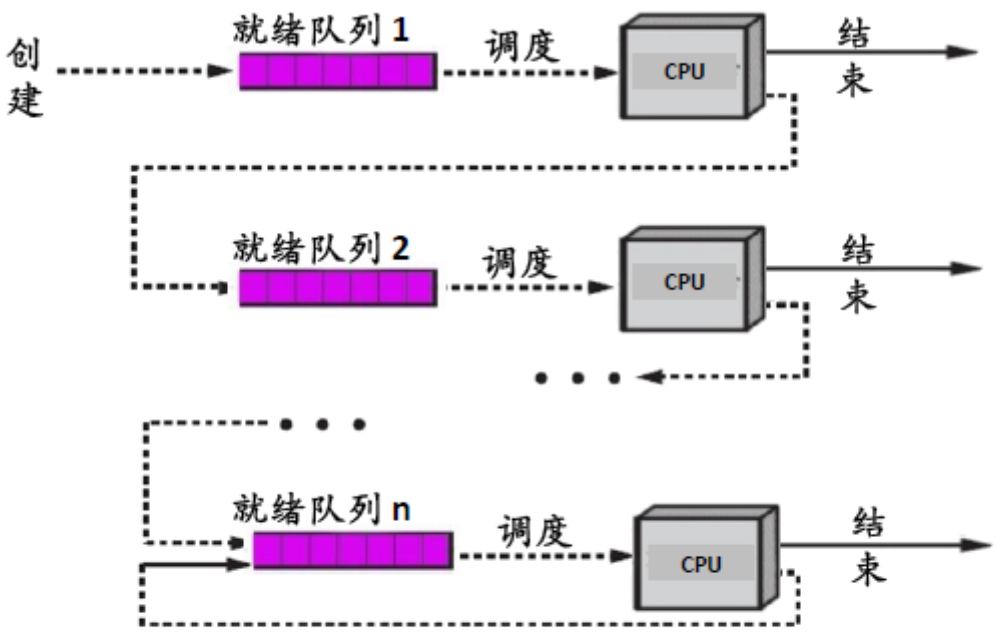
最高响应比优先(HRRN)

- 是一个综合算法，调度时，首先计算每个进程的响应比R，之后总是选择R最高的进程执行。
- **响应比R** = 周转时间 / 处理时间 =（处理时间 + 等待时间）/ 处理时间 = 1 +（等待时间 / 处理时间)
- **特点**：折中权衡
- **注意**: 每次运行完一个进程后, 所有进程的等待时间会更新, 要重新计算R

轮转(*Round Robin——RR*)

- 每个进程被分配一个时间片，允许该进程在该时间段运行，如果在时间片结束时该进程还在运行，则剥夺CPU并分配给另一个进程，如果该进程在时间片结束前阻塞或结束，则CPU立即进行切换。
- 特点：公平；有利于交互式计算，响应时间快；由于进程切换，时间片轮转算法要花费较高的开销；对进程表中不同进程的大小差异较大的有利，而对进程都是相同大小的不利

多级反馈轮转



队列模型示意

- 设置多个就绪队列，并为各个队列赋予不同的优先级。
- 第一个队列的优先级最高，依次递减优先级。对于各个队列进程执行时间片的大小也不同，优先级越高的队列，分配到的时间片越少。
- 当第一级队列为空时，再第二级队列进行调度，依次类推，各级队列按照时间片轮转方式进行调度。
- 当一个新进程创建后，首先把它放入第一队列的末尾。按照FCFS原则排队等待调度。当轮到该进程执行时，如它在该时间片完成，便可准备撤离系统，如果它在一个时间片结束时尚未完成，则调度程序便将该进程转入第二队列的末尾，再同样地按照FCFS原则等待调度执行。依次类推。
- **特点**：更偏好I/O型进程，对CPU型进程不太友好。

总结

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

原理

1. **非对称多处理(AM)**
   - 让一个主处理器处理所有调度决定、I/O处理以及其他系统活动，其他处理器只执行用户代码
   - 只有一个处理器访问系统数据结构，减少了数据共享的需要
2. **对称多处理(SMP)**
   - 每个处理器自我调度
   - 所有进程可能处在一个共同的就绪队列中，也可能每个处理器都有自己私有的就绪队列。无论怎样，每个处理器的调度程序都检查**共同就绪队列**，以便选择执行一个进程

处理器亲和性

> 当一个进程运行在一个特定的处理器上时会缓存该进程的一些数据和上下文，如果进程被迁移到另外一个处理器上，那么上一个处理器上缓存的数据设为无效，第二个处理器缓存应刷新。由于这些操作代价很高， 所以大多数SMP系统试图**避免进程从一个处理器迁移另一个处理器。**这叫做处理器的亲和性。

软亲和性

- 当一个操作系统试图保持进程运行在同一处理器上，但是这个进程也可以迁移到其他进程上

硬亲和性

- 操作系统提供系统调用来支持使某个进程运行在某个处理器子集中

负载均衡

> 对于SMP系统，重要的是保持所有处理器的负载平衡，以便充分利用多处理器的优点. 否则, 一个或多个处理器总是很闲, 其他处理器反而处于高负载状态

保持负载均衡的方法

1. 推迁移:

   一个特定的任务周期性地检查每个处理器的负载，如果发现不平衡，那么通过将进程从超载处理器推到空闲或不太忙的处理器，从而平均分配负载

2. 拉迁移

   空闲处理器从一个忙的处理器上拉一个等待任务

弊端

负载均衡往往会抵消处理器亲和性的好处。