ch6 CPU调度

基本概念

通过多道程序提高cpu的利用率

CPU-I/O区间周期

进程执行由CPU执行和I/O等待周期组成

I/O约束程序：大量短CPU区间

CPU约束程序：少量长CPU区间

CPU调度程序

负责选择

CPU空闲时从内存中就绪可执行的进程中选择一个，为其分配CPU

就绪队列中的记录通常是PCB

CPU调度决策在 4 种情况下发生：

1运行到等待，2运行到就绪，3等待到就绪，4进程终止

抢占式调度：在4种情况下都调度

非抢占式调度：只在1，4情况下调度

分派程序

负责移交控制权

将cpu控制权交给CPU调度程序选择的进程

功能：1.切换上下文，2.切换到用户模式，3. 跳转到用户程序的合适程序以重新启动这个程序

分派延迟：切换进程时，停止一个进程，启动另一个进程花的时间

调度准则

General Criteria

CPU使用率

吞吐量：单位时间内完成进程的数量

周转时间：运行进程花费的时间。从等待进入内存到进程结束

等待时间：就绪队列中等待时间之和

响应时间：开始响应花费的时间。从等待进入内存到产生第一响应的时间。（周转时间受输出设备速度的限制）

Optimization Criteria

最大CPU使用率

最大吞吐量

最小周转时间

最小等待时间

最小响应时间

> 本书标准：最小化平均**等待时间**

调度算法

FCFS (First-Come, First-Served ）（先到先服务调度）

计算平均等待时间

护航效果：小进程等待大进程

非抢占的 -> 对于分时系统比较麻烦

SJF (Shortest-Job-First ）（最短作业优先调度）

每个进程与**下一个CPU区间段**相关联（不是总长度）

最佳

困难：如何知道下一个CPU请求的长度 -> 常用于长期调度，不适合短期调度

抢占或非抢占

- 非抢占式的
- 抢占式的：SRTF最短剩余时间有限

Priority Scheduling（优先级调度）

每个进程与一个优先权关联，相同优先权的按照FCFS

SJF是优先级调度的一个特例

> 本书中用小的数字表示高优先级

抢占或非抢占

问题：饥饿（低优先级的进程无穷堵塞）

解决方案：老化（增加长时间等待进程的优先级）

RR (Round Robin)（轮转法调度）

类似FCFS，但加入了**抢占**

专门为**分时系统**设计

就绪队列：循环FIFO队列

设置时间片间隔，每次给进程分配不超过一个时间片间隔的CPU

时间片结束但没运行完的进程会加到循环队列的尾部

这种方法的平均等待时间非常长（注意计算）（更长的周转时间，更短的响应时间）

性能依赖于时间片的大小

太大会变成FCFS 太小会增加上下文切换的开销

Multilevel Queue（多级队列调度）

就绪队列分成多个独立队列，每个队列由自己的调度算法

进程被**永久地**分到一个队列

队列之间要有调度，通常采用固定优先级可抢占

通常前台队列比后台队列优先级高

具体方案：

- 队列之间具有绝对的优先级，只有高优先级队列空，才运行低优先级队列
- 给队列分配时间片

Multilevel Feedback Queue（多级反馈队列）

多级队列不允许进程在队列间移动 -> 优点 低调度开销 缺点 不灵活

多级反馈队列允许进程在队列间移动：

升级到高优先级队列 / 降级到低优先级队列

多处理器调度

负载分配

如何把公共就绪队列中的进程分配给处理器？两种方案：

1. 每个处理器自我调度，自己去取。这种方法必须做好数据一致性，因为访问同一个共

享存储区。难实现。
2. 由一个处理器负责分派。主从结构

实时调度

- 硬实时

  资源预约

- 软实时

  要求：

  - 系统有优先权调度，实时队列优先权最高，且这种情况不会随时间发生改变。
  - 小的分派延迟。

优先级倒置 -> 解决：优先级继承

线程调度

调度方法

- 本地调度

  线程库如何决定将哪个线程放入可用的LWP。

- 全局调度

  内核如何决定下一个运行哪个内核线程。

Solaris 2 调度

Java线程调度

算法评估

解决的问题：在有了前面的评价准则之后，如何评估到底哪种调度算法最好

确定性建模

假设特例，以点概面

排队模型

利用统计学原理求平均

模拟

编写模拟程序

实现

真实地实现出来