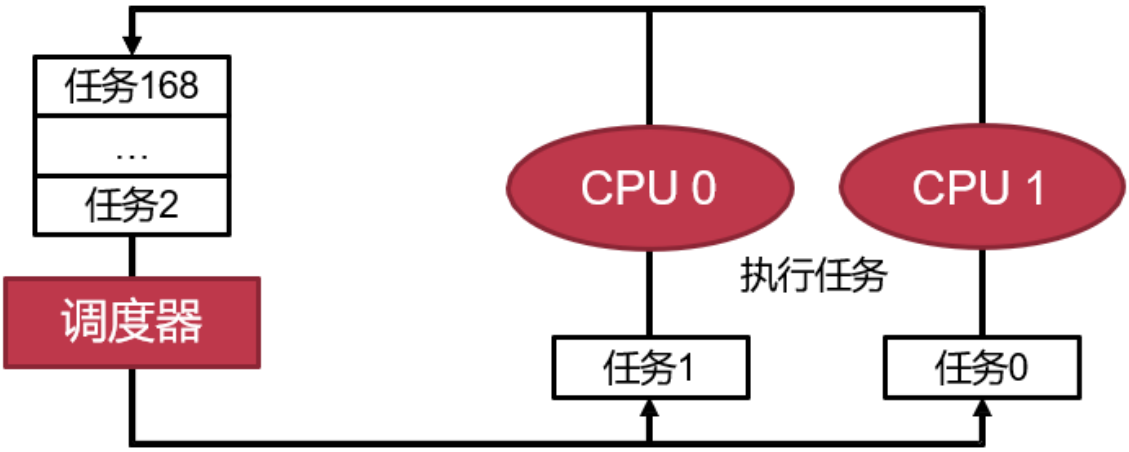
进程调度

Overview



为了在有限的资源下通过对多个程序执行过程的管理，满足系统和应用的指标，调度器需要对线程进行调度(Scheduling)。

> 线程是调度器的调度对象，但在Linux等OS常用任务(job/task)描述线程。

执行停止执行条件

- 时间片结束
- 出现I/O请求
- 主动停止或进入睡眠
- 被系统打断（抢占式调度）

调度决策

1. 下一个执行的任务
2. 执行该任务的CPU核心
3. 执行的时长，即时间片

调度指标

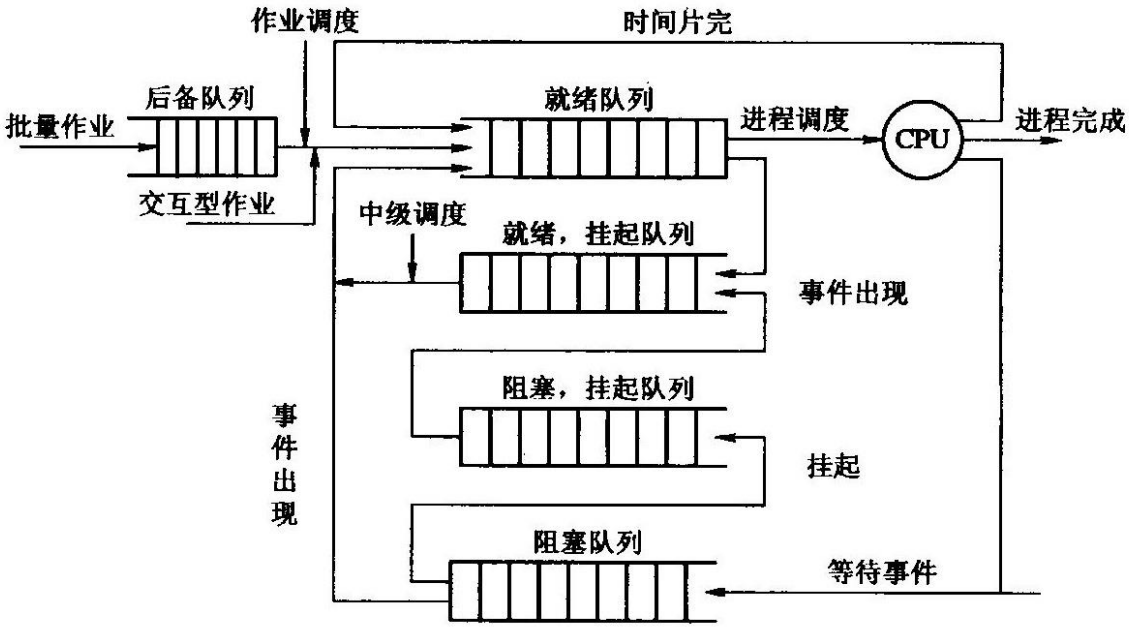
Scheduling Criteria（调度指标）

- **周转时间**：任务第一次进入系统到执行结束的时间
- **响应时间**：任务第一次进入系统到第一次给用户输出的时间
- **实时性**：在任务的截止时间内完成任务
- **公平性**：每个任务都应该有机会执行，不能饿死
- 吞吐率：单位时间内处理的任务数量
- 开销低：调度器是为了优化系统，而非制造性能BUG
- 可扩展：随着任务数量增加，仍能正常工作

调度的挑战

- 缺少信息
- 线程/任务间的复杂交互
- 调度目标多样性：不同的系统可能关注不一样的调度指标

三级调度



具有三级调度时的调度队列模型

进程调度可以分为三类：

- Long-Term Scheduler
- Short-Term Scheduler
- Medium-Term Scheduler

Long-Term Scheduler

也称为 job scheduler.
goal: **regulates the program and select process from the queue and loads them into memory for execution.**
负责将处于新生状态的进程转为就绪状态，限制系统被短期调度管理的进程数量。

Short-Term Scheduler

也称为 CPU scheduler.
goal:This helps you to select from a group of processes that are ready to execute and allocates CPU to one of them.
负责进程在就绪状态、运行状态和阻塞状态的转换。

Medium-Term Scheduler

中期调度是换页机制的一部分，能够降低进程占用的内存。
机制：将被短期调度器管理的处于就绪状态或阻塞状态的进程挂起，转为**挂起就绪**状态或**挂起阻塞**状态，并在适当时机重新激活被挂机的进程。

批处理任务调度

批处理任务调度算法的关注指标主要是**周转时间**。

FCFS

先到先得(First Come First Served, FCFS)，也称为先进先出(FIFO)策略。

- 特点：简单、直观，有利于长任务。

- 问题： 平均周转周期长、响应时间长

SJF

最短任务优先(Shortest Job First, SJF)，该策略会选择**运行时间最短**的任务执行。

- 特点：**平均周转时间短**
- 问题：
  - 必须预知任务的运行时间
  - 严重依赖于任务到达时间点：若短任务迟来则不会执行（非抢占式调度）
  - 不公平，长任务容易饿死

STCF

最短完成时间任务优先(Shortest Time-to-Completion First, STCF)，也称为最短剩余时间优先 shortest remaining time next（SRTN）。该策略会选择**剩余运行时间最短**的任务执行，且为**抢占式调度**(Preemptive Scheduling)，可以看出SJF的抢占式版本，解决了迟来的短任务无法受益问题。

> 抢占式调度: 在任务到达系统时也会进行调度，有可能中断当前执行的任务转而执行其他任务。

- 特点：**平均周转时间短**
- 问题：不公平，长任务容易饿死

交互式任务调度

RR

时间片轮转(Round Robin , RR)：为了定时响应用户，为任务设置时间片，限定任务每次执行时间。

- 特点：平均**响应时间短**
- 问题
  - **周转时间长**，尤其是任务运行时间相似的场景
  - 需要选取时间片大小，过小的时间片会导致大量的**上下文切换开销。**

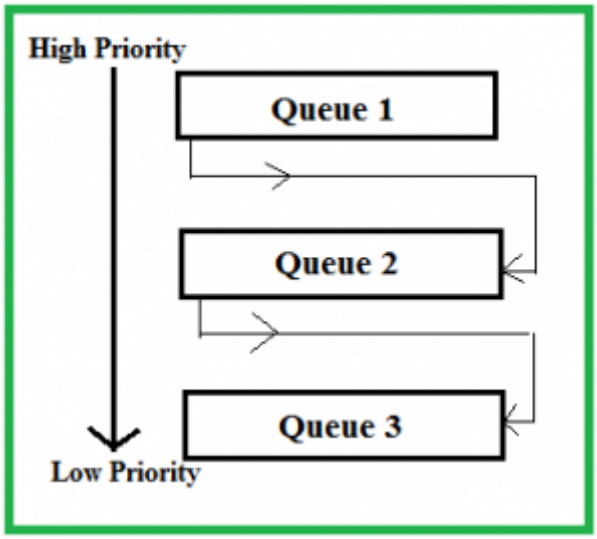
优先级调度

通过为每个任务指定一个**优先级(Priority)**，调度器坑确定哪些任务应该优先执行。实际上，FCFS、STCF、STCF、RR也有优先级。
Multi-level Queue(多级队列)属于静态优先级调度策略，严重依赖于预设的优先级进行调度、

注意事项：

- 提高I/O密集型任务的优先级，避免造成I/O利用率低下
- 低优先级任务饥饿问题：需要为等待时间过长的任务提高优先级
- 优先级反转

MLFQ



多级反馈队列(Multi-Level Feedback Queue, MLFQ)，属于经典的基于多级队列的动态优先级调度。

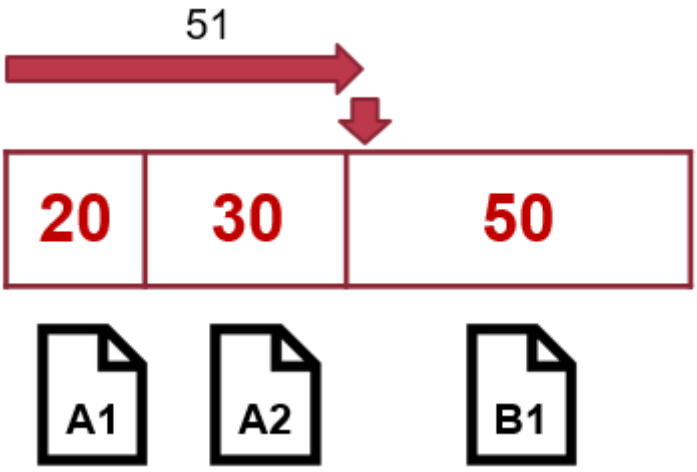
we arrive at the first two basic rules for MLFQ:
• Rule 1: If Priority(A) > Priority(B), A runs (B doesn't).
• Rule 2: If Priority(A) = Priority(B), A & B run in RR.优先级越低，时间片越长。
• Rule 3: When a job enters the system, it is placed at the highest priority (the topmost queue).
动态调节机制：
• **Penaly**: If a job uses up an entire **time slice while running**, its **priority is reduced** (i.e., it moves down one queue).
• **Boost**：**After some time period** S, move all the jobs in the system to the **topmost queue**.
Penaly可以提高短任务（同时包括I/O密集型任务）的优先级，从而提高平均周转时间。
Boost可以防止长任务饥饿。
问题：需要调整许多参数，如优先级队列的数量、时间片大小及最大运行时间，Boost的周期等等。

公平共享调度

公平共享调度（Fair-Share Scheduling）考虑用户占用的资源比例：用**份额**（share）量化每个任务对CPU时间的使用，明确任务应该使用的**系统资源比例**。

Lottery Scheduling

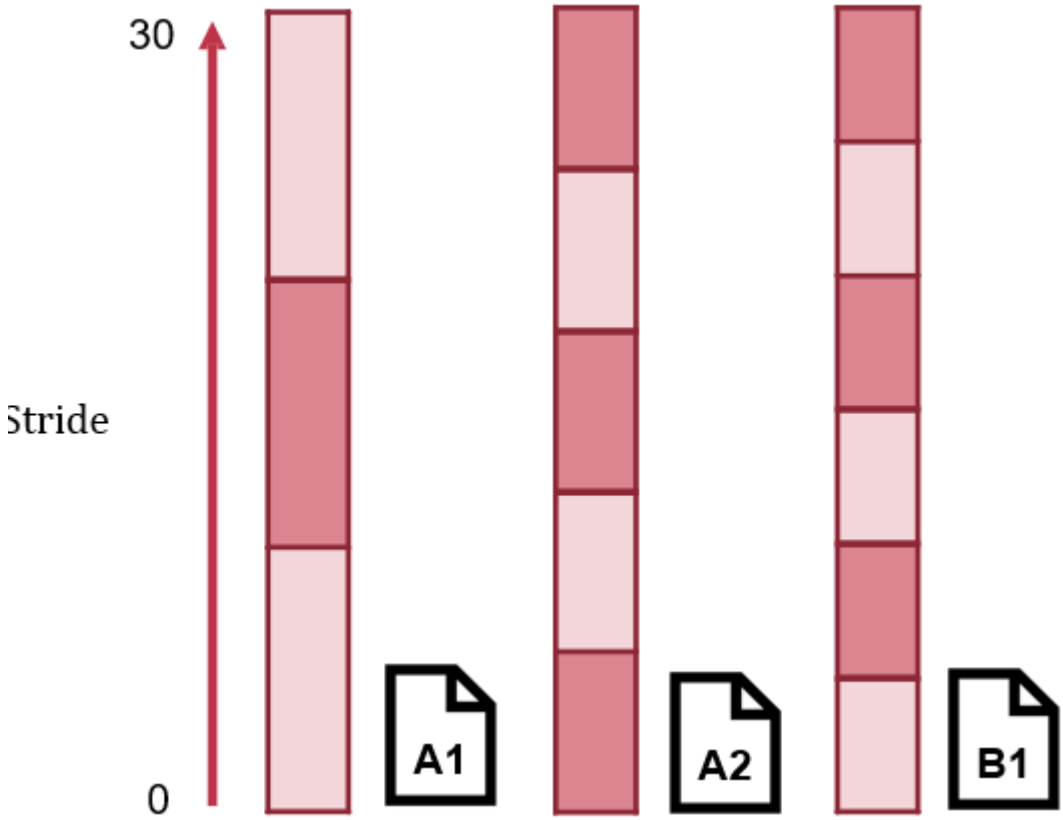
Lottery Scheduling（彩票调度）：根据每个任务的份额比例分配每个任务的彩票数量，任务拥有的彩票数量越多，越有可能被调度：每次调度根据生成随机数确定任务是否被调度。



Ticket Transfer(彩票转让)：与优先级反转类似。

Lottery Scheduling的随机化实现简单，但会带来不确定性，在调度次数较少时将达不到预期效果（即频率和概率在次数较少的情况下下相差过大）。

Stride Scheduling



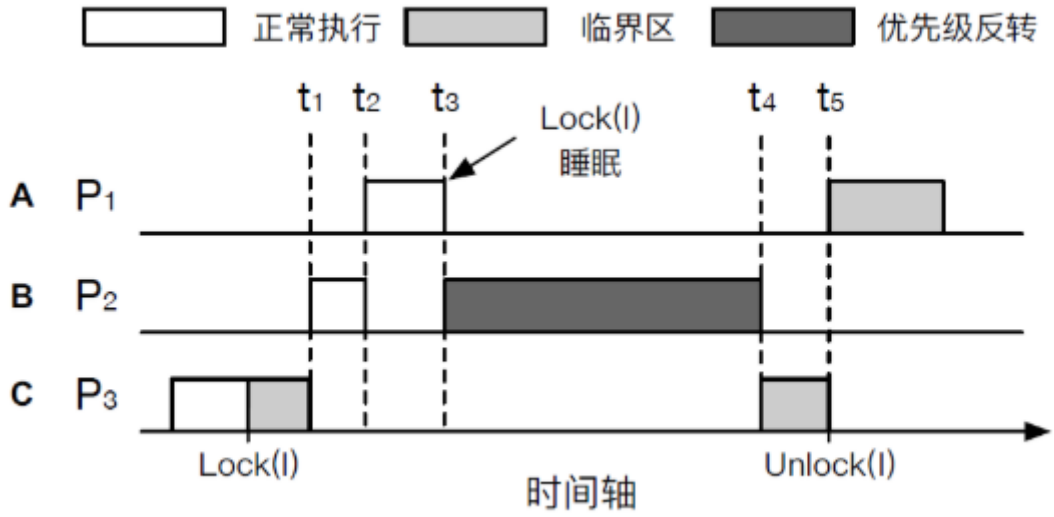
Stride Scheduling(步幅调度)：**确定性**版本的Lottery Scheduling，核心是**步幅（Stride）**：任务一次执行增加的虚拟时间，和份额成反比。
机制：每次调度选择当前虚拟时间最小的任务调度并增加虚拟时间（即步幅）大小。

> 新的任务进入时虚拟时间应该设置成当前所有任务的最小虚拟时间，防止新的任务长时间占有CPU。

优先级反转

线程执行顺序违反预先优先级。

产生原因



高优先级A线程抢占了低优先级C线程并尝试获取被C线程持有的资源，导致A线程阻塞，另一个比A线程优先级低的线程B却可以正常执行。

- 操作系统：基于优先级调度
- 锁：按照锁使用的策略进行调度

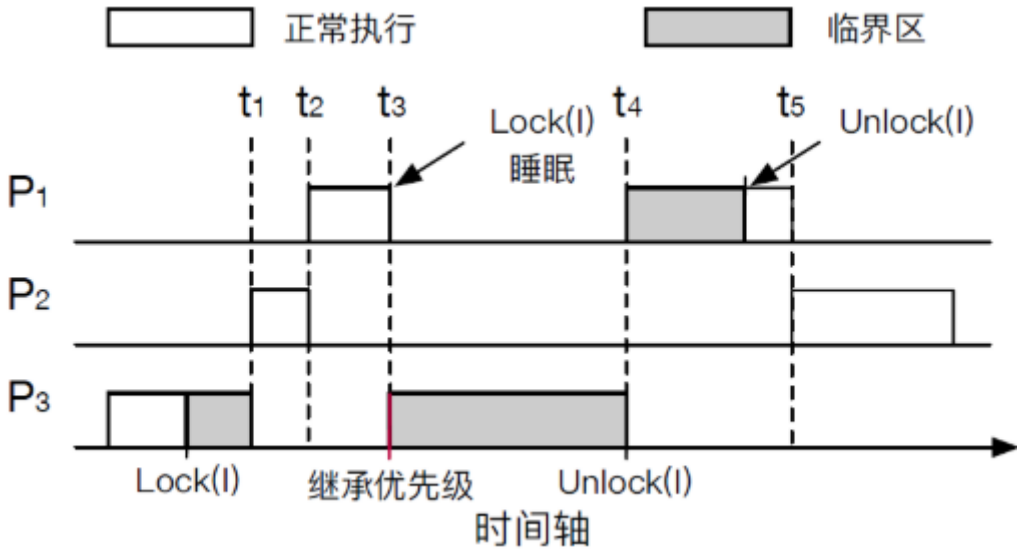
根本原因：双重调度不协调

解决方案

不可打断临界区协议

不可打断临界区协议(Non-preemptive Critical Sections, NCP)：进入临界区后不允许其他进程打断：禁止操作系统调度。

优先级继承协议

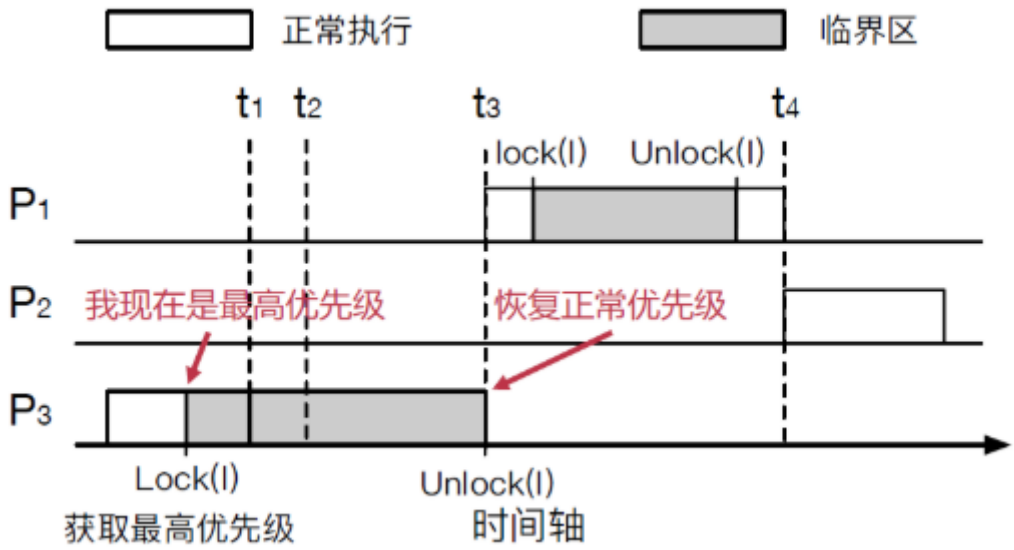


优先级继承协议(Priority Inheritance Protocol, PIP)：将被阻塞的高优先级任务的优先级转移给占用资源的低优先级任务。

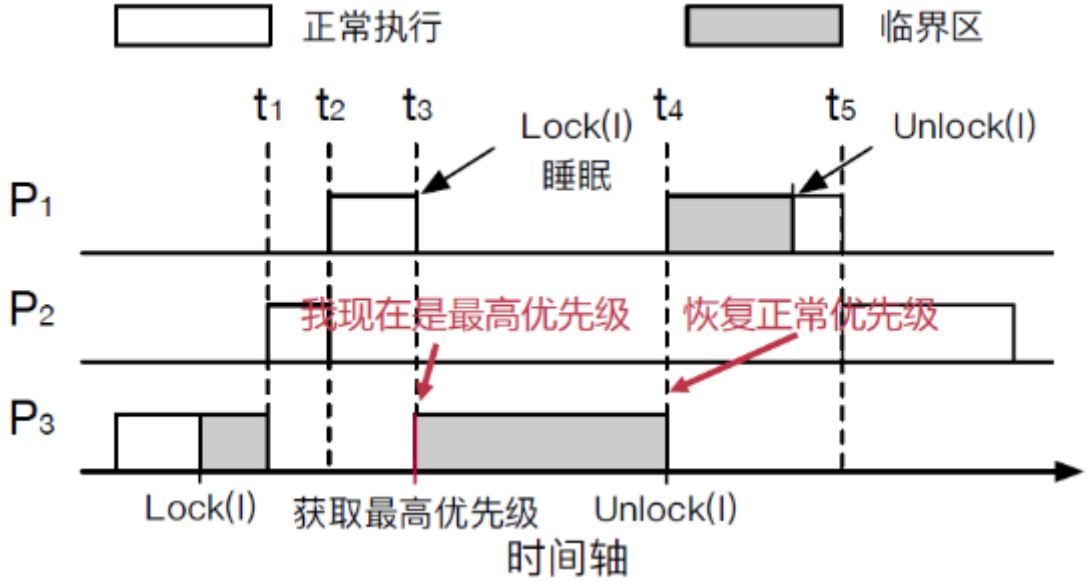
优先级置顶协议

优先级置顶协议(Priority Ceiling Protocols, PCP)：将获取锁的线程的优先级设置为该锁的所有竞争线程的最高优先级。
根据获取锁的时间分类，该协议可分为：

- 即时优先级置顶协议(Immediate Priority Ceiling Protocols, IPCP)：在获取锁后**立刻**提升优先级。



- 原生优先级置顶协议(Original Priority Ceiling Protocols, OPCP)：在**检测到高优先级竞争该锁**时再提升优先级。



实时调度

对于**周期任务**，主要有两种调度算法：

- 速率单调
- 最早截止时间优先

速率单调

速率单调（Rate-Monotonic, RM）：**速率**越高，则优先级越高。

> 速率：任务周期的倒数

在基于静态优先级的实时调度策略中，RM策略是最优的。但在一组任务的CPU利用率U不大于1的情况下，不一定能够满足这组任务的截止时间要求。

最早截止时间优先

最早截止时间优先(Earliest Deadline First, EDF)：根据任务的**截止时间**动态调整任务的优先级。
在CPU利用率U不大于1的情况下，EDF一定能够满足这组任务的截止时间要求，是在U不大于1的情况下的最优实时调度策略。
若U大于1，则会引发多米诺效应：EDF没有任务的运行时间信息，无法提前拒绝任务，也没有中断机制调度其他任务。

Reference

- Multilevel Feedback Queues (MLFQ) - LASS-PDF
- 上海交通大学SE-315 · 操作系统（2020春）