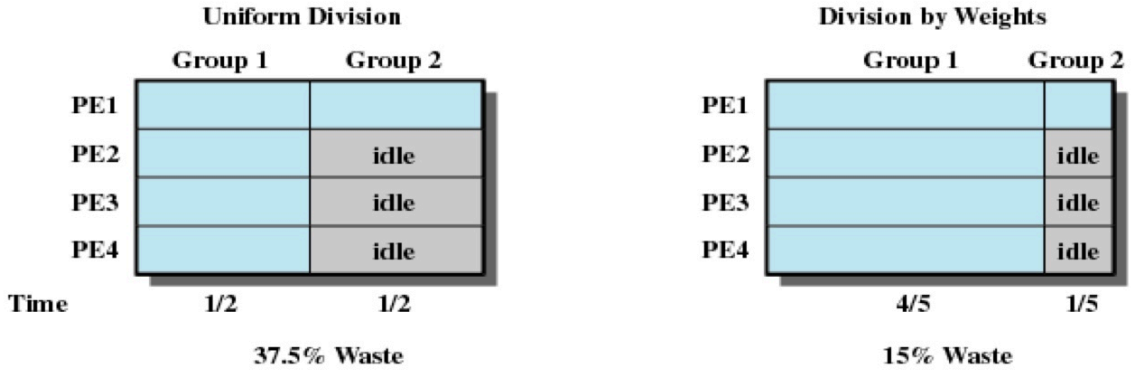
第十章 多处理机和实时调度

1. 与单处理机调度的区别
    - 注重**整体运行效率**（而不是个别处理机的利用率）
    - 更**多样的调度算法**
    - 多处理机访问 OS 数据结构时的**互斥**（对于共享内存系统）
    - **调度单位**广泛采用线程
2. 无约束性并行
    - 完全独立的应用或作业
    - 进程之间没有同步问题
    - Example is time-sharing system
3. 粗粒度和非常粗粒度的并行
    - 在很高级别上的进程间同步
    - 多道程序设计的单处理器上的并发进程运行有很好的效果
    - Can be supported on a multiprocessor with little change
4. 中等粒度的并行
    - 每个应用本身由一组线程组成
    - 线程间经常交互
5. 细粒度的并行
    - 高度并行的应用
    - 是一个特殊领域：有很多不同方法
6. 调度的考虑
    - 为进程选择处理器
    - 在每个处理器上使用多道程序设计
    - 实际的进程分派
7. 处理器的选择
    - 把处理器看成一个资源池，并按照要求把进程分派到相应的处理器
    - 静态分派：总分配在一个处理器上
        - 为每个处理器单独维护一个短程队列
        - 低开销
        - 可能造成负载不平衡
    - 全局队列
        - Schedule to any available processor
        - 当处理器很多时会产生性能瓶颈
    - 主从结构
        - 内核始终运行在一个特殊的处理器上

        - 主处理器负责调度
        - 从处理器向主处理器发送服务请求
        - 缺陷
            - 主处理器崩溃整个系统崩溃
            - 主处理器可能会成为系统性能瓶颈
    - 对等结构
        - 操作系统能运行在任何一个处理器上
        - 每个处理器单独进行调度
        - 使操作系统变得复杂
        - Make sure two processors do not choose the same process
8. 进程调度
    - 多优先级队列
    - 所有队列送进相同的处理器池中
9. 线程调度
    - 一个应用程序可以作为一组线程来执行
    - 线程在同一个地址空间中并发地执行
    - 如果线程同时运行在不同的处理器上，将获得非常高的效率
10. 多处理器线程调度
    - 加载共享
        - 线程不是分配到一个固定的处理器，而是维护 一个全局队列
    - 组调度
        - 一组相关线程同时调度到一组处理器上执行
    - 专用处理器分配
        - 线程指定到处理器
    - 动态调度
        - 执行过程中，进程的线程数目可变
11. 加载共享
    - 负载平均地分布在每个处理器上
    - 不需要集中调度器
    - 使用全局队列
12. 加载共享的缺点
    - 全局队列需要互斥访问
        - May be a bottleneck when more than one processor looks for work at the same time
    - 被强占的线程可能在其他的处理器上恢复执行
        - Cache use is less efficient
    - 如果所有线程被看做一个公共的线程池，那么一个程序的所有线程很难同时获得处理器
13. 组调度
    - 同时在一组处理器上调度一组线程

- Useful for applications where performance severely degrades when any part of the application is not running
- Threads often need to synchronize with each other

Scheduling Groups



14. 专用处理器分配
    - 把应用程序指定到固定的处理器
    - Some processors may be idle
    - No multiprogramming of processors
    - 合理性
        - 当有数十个或上百个处理器时，处理器使用率要求不像单处理器情况
        - 在一个程序的生命周期避免处理器的切换会加快程序的速度

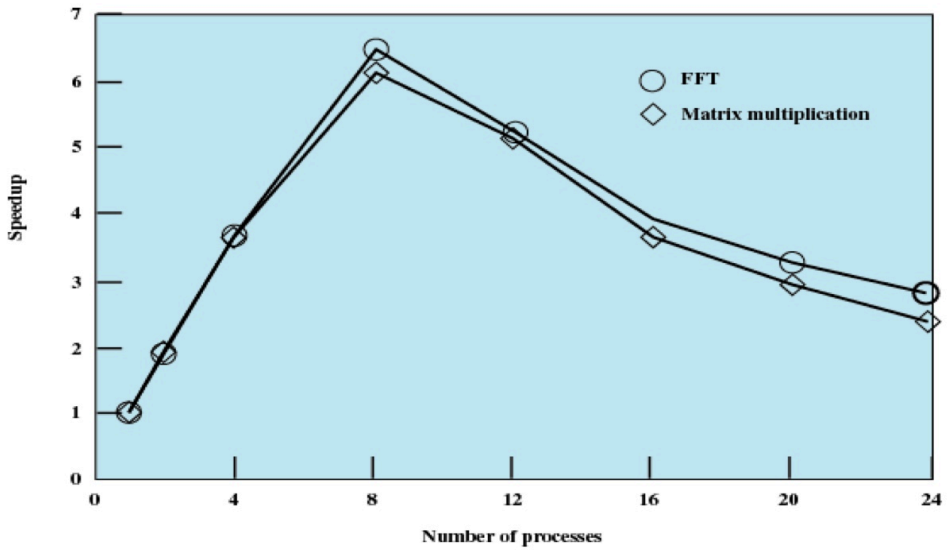


Figure 10.3 Application Speedup as a Function of Number of Processes [TUCK89]

15. 动态调度

- Number of threads in a process are altered dynamically by the application
- 操作系统通过调整负载情况来提高使用率
  - 分派空闲处理器
  - 新到达的作业从占有多个处理器的作业中分出一个处理器进行分派
  - 得到处理器前始终保持请求
  - 给当前没有处理器的作业分配一个处理器

16. **实时系统**
    - 系统的正确性不仅依赖于程序的运行结果， 同时也依赖于返回结果的时间
    - Tasks or processes attempt to control or react to events that take place in the outside world
    - These events occur in “real time” and tasks must be able to keep up with them
17. **实时操作系统的特点**
    - Deterministic—— 可确定性
      - 可以按照固定的、事先预定的时间或者时间间隔执行操作
      - 取决于系统的响应速度，其次取决于系统是否有足够的能力在要求的时间内处理所有的请求
    - Responsiveness—— 可响应性
      - 可确定性关心系统获知中断前的时间延迟
      - 可响应性关心系统为中断提供服务的时间
    - 用户控制
      - 用户确定优先级
      - Specify paging
      - What processes must always reside in main memory
      - Disks algorithms to use
      - Rights of processes
    - 可靠性
      - 性能的减低或损失会造成灾难性后果
    - Fail-soft operation—— 故障弱化操作
      - 系统在发生故障时尽可能多地保存其性能和数据的能力
      - 稳定性
18. 实时操作系统的典型特征
    - 快速的进程或线程切换
    - 尺寸小
    - 快速响应外部中断的能力
    - 进程通信的多任务工具
      - 信号量、事件
    - 可以快速收集数据的特殊顺序文件
    - 基于优先级的预先调度
    - 禁止中断的间隔最小化

- 任务延迟固定的时间
- 警报和超时

19. **实时调度**
    - 静态表驱动
        - Determines at run time when a task begins execution
    - 静态优先级驱动抢占法
        - Traditional priority-driven scheduler is used
    - 基于动态规划调度
        - Feasibility determined at run time
    - Dynamic best effort—— 动态尽力
        - No feasibility analysis is performed
20. **Deadline Scheduling**
    - 时限调度
    - Real-time applications are not concerned with speed but with completing tasks
    - Information used
        - Ready time
        - Starting deadline
        - Completion deadline
        - Processing time
        - Resource requirements
        - Priority
        - Subtask scheduler
    - 举例 Two Task

Table 10.2 Execution Profile of Two Periodic Tasks

Process	Arrival Time	Execution Time	Ending Deadline
A(1)	0	10	20
A(2)	20	10	40
A(3)	40	10	60
A(4)	60	10	80
A(5)	80	10	100
• • •	• • •	• • •	• • •
B(1)	0	25	50
B(2)	50	25	100
• • •	• • •	• • •	• • •

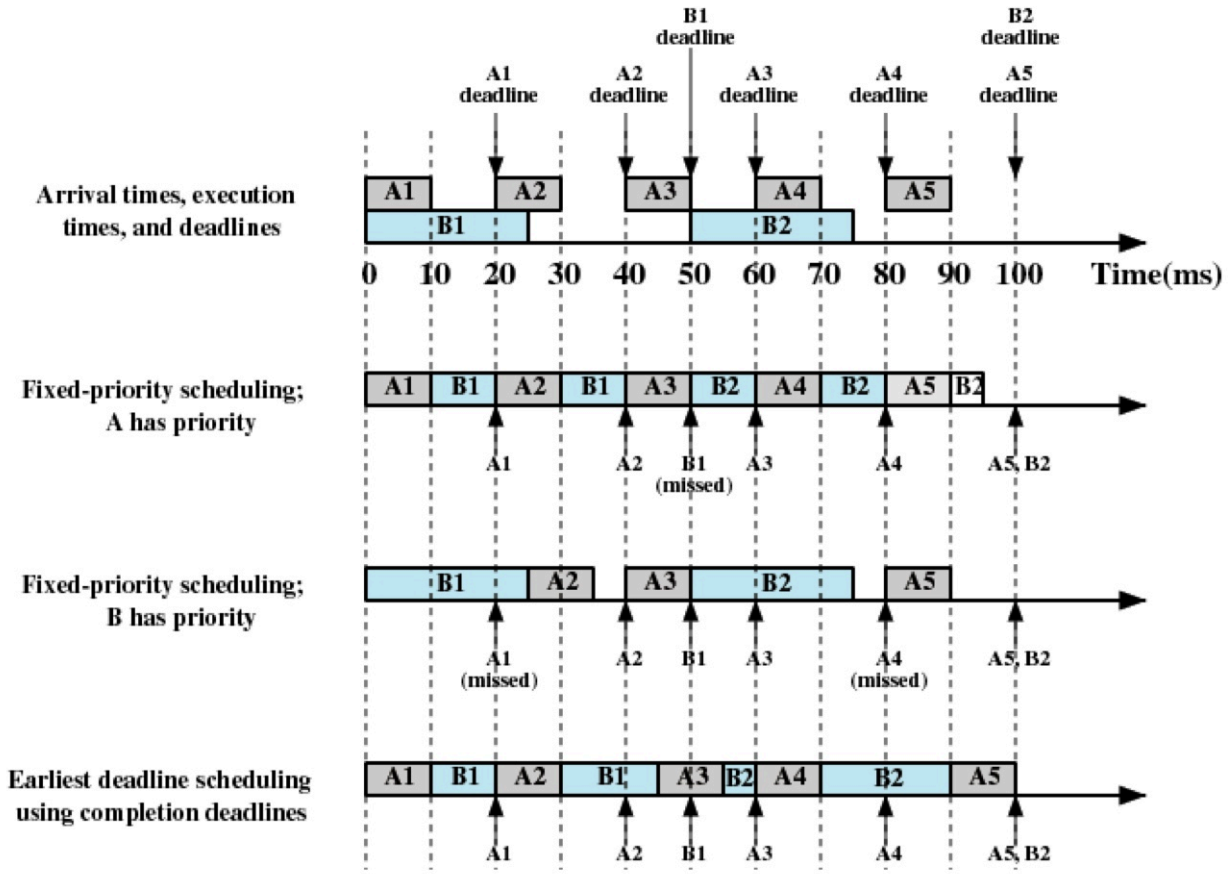


Figure 10.5 Scheduling of Periodic Real-time Tasks with Completion Deadlines (based on Table 10.2)

Table 10.3 Execution Profile of Five Aperiodic Tasks

Process	Arrival Time	Execution Time	Starting Deadline
A	10	20	110
B	20	20	20
C	40	20	50
D	50	20	90
E	60	20	70

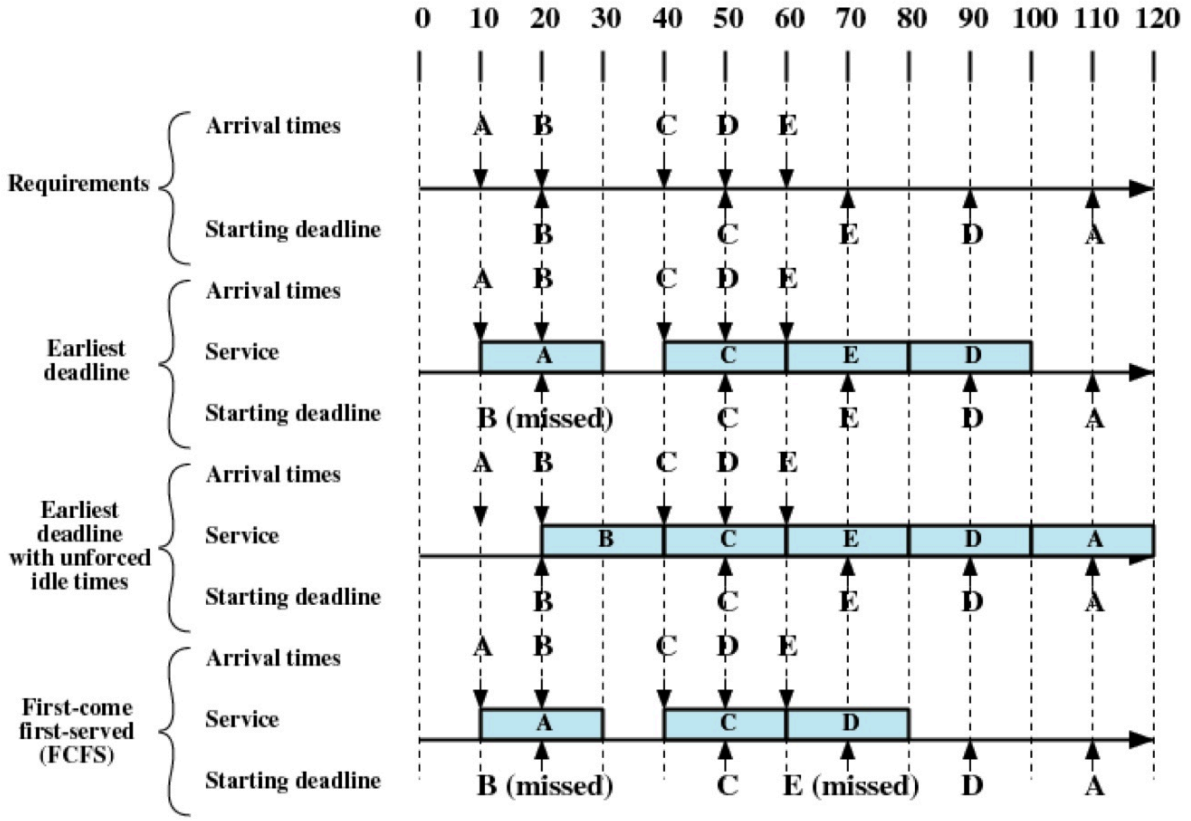
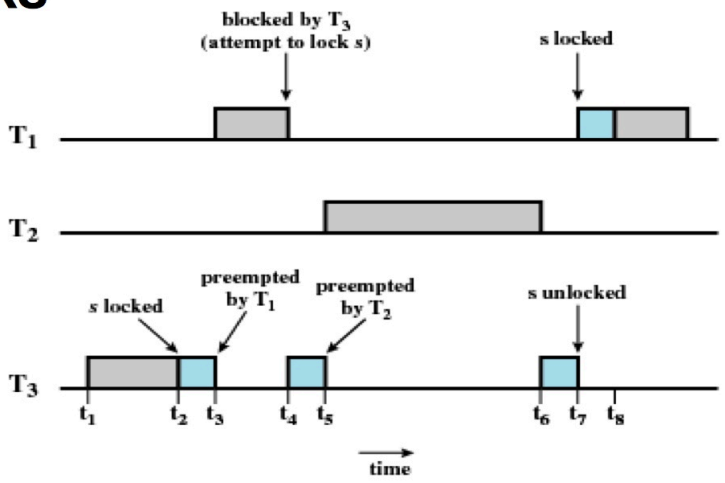


Figure 10.6 Scheduling of Aperiodic Real-time Tasks with Starting Deadlines

21. **Rate Monotonic Scheduling**
    - 速率单调调度 —— 周期性任务
    - Assigns priorities to tasks on the basis of their periods
    - Highest-priority task is the one with the shortest period
22. **Priority Inversion**
    - 优先级逆转
    - Can occur in any priority-based preemptive scheduling scheme
    - Occurs when circumstances within the system force a higher priority task to wait for a lower priority task

Unbounded Priority Inversion

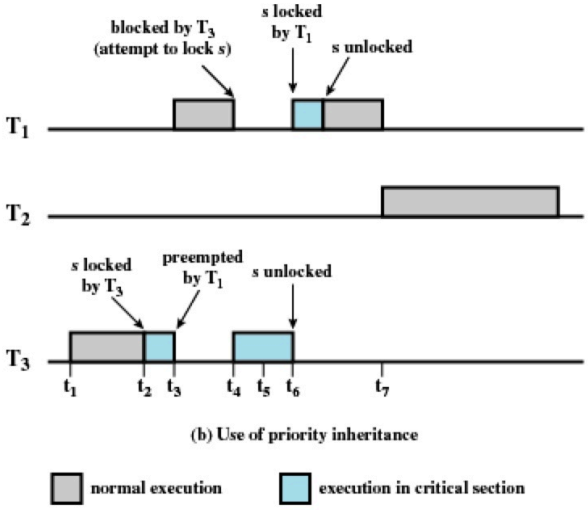
- Duration of a priority inversion depends on unpredictable actions of other unrelated tasks



(a) Unbounded priority inversion

Priority Inheritance

- Lower-priority task inherits the priority of any higher priority task pending on a resource they share



(b) Use of priority inheritance