第3讲
实时任务管理

安吉尧

目　录

3.1 实时任务的概念

实时系统所要响应和处理的外部事件有三种形式：

- 多个异步发生的外部事件
- 多个周期性发生的外部事件
- 上述两种情况的组合

如果为处理每一个外部事件而编制一个程序，这种程序的执行有两种方式。一种是*顺序执行*，一种是*并发执行*。

3.1 实时任务的概念

3.1.1 用顺序执行的程序实现实际应用系统

- 用一个事件中断处理程序，来捕捉外部所发生的事件，并把捕捉到的事件登记在一个先进先出(FIFO)的事件队列中
- 用一个控制循环不断地测试事件队列，若事件队列非空，取队列的第一个元素，转去处理相应事件的程序，并把该元素从队列中移去
- 执行结束，又回到循环的顶部，继续测试队列和取队列中事件进行处理。

这种顺序的处理方式,具有如下特点：

- 严格按照顺序执行，每一个操作都必须在一个操作结束之后进行
- 程序在运行过程中，独占系统资源，系统资源的状态只由程序本身确定，不受外界因素的影响

- 程序执行的结果与它的执行过程无关，不管它是连续不断地执行，还是曾经被事件中断处理程序所中断过，都不影响它的执行结果
- 如果程序执行时的初始条件相同，则最终结果也相同

以上特点说明了顺序程序的封闭性

——所谓封闭性，指程序一旦运行，其结果不受外界因素的影响

3.1 实时任务的概念

3.1.2 用并发执行的任务实现实时应用系统

随着计算机多重处理技术和多道程序的引入，就有可能使这些任务并发地执行，使得紧急的任务可以中断正在执行的任务，从而解决程序顺序执行所带来的问题。

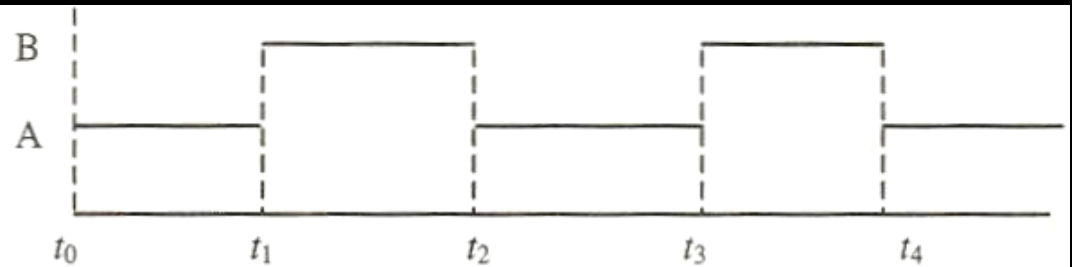


图 3.1　两个任务并发执行

假设,把处理事件A、B的两个任务，任务A、B的并发执行过程，如上图所示：

在时刻t1，紧急度较高的事件触发任务B执行，中断了A的执行CPU的控制权交给B，操作系统把A的执行状态保存下来，同理在t2、t3、t4时刻。

如上述可见，**任务的运行**具有如下几种基本特征：

- **动态特征**：任务是某种类型的一个活动，它由创建而产生，由调度而执行，得不到资源而暂停，最后由运行结束或撤消而消亡，因此，任务具有生命期，是动态产生和消亡的
- **并发特征**：若干个任务并发执行，互相竞争CPU的控制权，必须用某种调度算法来决定何时停止一个任务的工作，转而为另一个任务提供服务
- **独立特征**：任务是一个独立的运行单位，有自己必须执行的程序、自己的程序计数器、加工自己的数据、有自己的输入输出，它是系统进行资源分配和调度的独立单位

- **异步特征**：各个任务按照自己独立的速度向前推进，它们的执行是异步的，有多个任务经常需要同步地协调处理某个控制目标或工作对象。系统需要为它们提供一套通信及同步互斥机制
- **结构特征**：任务有其运行状态，必须为每一个任务配置一个任务控制块ＴＣＢ—任务所要执行的程序代码及任务所要加工处理的数据集组成

3.1 实时任务的概念

3.1.3 实时任务的分解

- 将一个实时应用问题分解为多个任务，可加快执行的速度，有效地利用系统资源
- 但是，过度的分解将使系统中有大量的任务，经常进行任务的切换，任务和任务之间，还需进行很多同步和互斥控制，将增加大量的系统服务工作，降低系统的速度和有效性
- 因此，把一个实时应用问题分解为若干个任务时，还必须进行各种综合平衡和折中，有时，把两个操作合并在一起处理，效果可能要好一些，有时，则必须分开处理，这都必须依赖于实时应用的特征

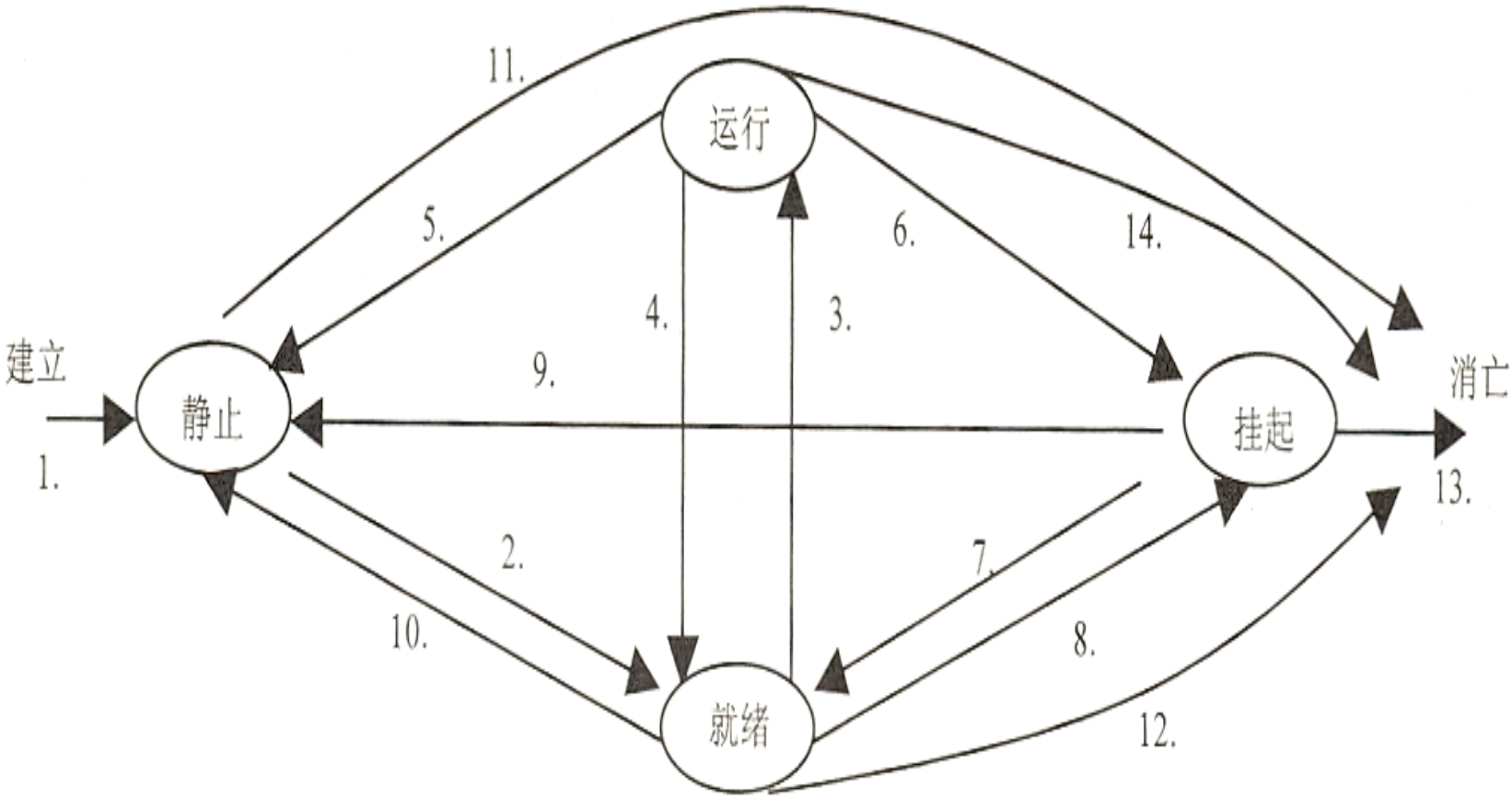
任务分解的准则：

- 1、时间原则
- 2、异步原则
- 3、优先性原则
- 4、清晰度和可维护性原则

3.1 实时任务的概念

3.1.4 实时任务的状态

- **静止状态**：任务建立之后，但尚未被启动，不具备运行条件，这时就认为任务处于静止状态
- **运行状态**：任务正在占用CPU运行
- **就绪状态**：任务已具备运行条件，准备运行，但CPU被别的任务所占有
- **挂起状态**：任务因某种原因，无法继续运行



1. create 2. start 3. schedule 4. schedule 5. exit, abort

6. suspend ,delay 7. active 8. suspend 9. 10. abort 11. 12. 13. 14. destroy

任务状态的转换情况

3.2 实时任务调度

实时任务的调度,就是当系统有多个任务时，如何确定由哪一个任务实际占有CPU。

实时任务调度牵涉到*调度的策略*和*调度的机制*问题，不同实时应用系统的用户，可能希望用不同的策略来进行调度，因此，对于实时操作系统来说，最好是把调度策略和调度机制分开。

实时任务调度策略：

- **单一速率调度算法**
- **截止时限最早优先算法**
- **可达截止时限最早优先算法**
- **最小裕度算法**
- **其他实时调度算法……**

Homework 2

Q: 还有哪些实时调度算法？

- 本讲目前所述调度算法描述均是**基于单处理器**情况
- 调度算法做出如下假设：
  - 抢占开销忽略
  - 忽略非处理器资源的要求
  - 任务之间没有优先约束

3.2 实时任务调度

3.2.1 速率单调算法

单一速率调度算法RMS（Rate-Monotonic Scheduling）是一个经典的算法，它是针对那些响应和处理周期性事件的实时任务的，任务的相对时限等于其周期

- 事先为每个实时任务分配一个与事件频率成正比的优先级，例如，周期为20ms的实时任务，优先级为50；而周期为100ms的实时任务，优先级为10，运行时，调度程序总是调度优先级最高的就绪任务
- 实现时，就绪队列中的所有任务按照优先级Priority排队，优先级最高的任务排在队首，当处于运行态的任务，由于某种原因被挂起时，只要把就绪队列的首元素从就绪队列中取下，使运行任务指针pRunTask指向该元素即可
- 如果是处于其他状态的任务变为就绪状态，而挂于就绪队列时，则必须对运行任务和就绪队列首元素的任务进行比较，优先级高的任务占有CPU

3.2 实时任务调度

3.2.2 截止时限最早优先算法

EDF（Earliest Deadline First）算法中截止时限最早的任务优先级最高，对于周期任务，其截止时限即为下一周期开始的时间，有时，把这种算法也称为时限驱动算法

就绪队列中的任务，按截止时限排序，截止时限早的任务排在队首，这个算法的处理，与单一速率调度算法类似，不同的是，现在是对截止时限进行判断，按截止时限最早优先策略处理。

本算法的缺点是：**已过或几乎要过截止时限的任务，仍然因优先级最高而得以运行，而这显然是不合适的**

3.2 实时任务调度

3.2.3 可达截止时限最早优先算法

本算法是对截止时限最早优先策略的改进，就绪队列的任务，仍然按照截止时限的顺序排队，但是在调度时超过截止时限的不予调度，如果记t为系统当前时间， E为任务估算执行时间，p为任务实际执行时间，d为截止时限。则：

$$d_1 = d - (t + E - p) \geq 0$$

表示该任务的截止时限是当前可达到的，于是，只要在调度时，按照上式计算被调度就绪任务的d_1，若大于 0 ，就进行调度，否则，就丢弃它。

在可达截止时限最早优先算法里，系统时钟管理部分中的时钟滴答中断处理程序，必须对运行任务的运行时间进行累计

空闲任务**IDLE**的截止时限DeadlineTime应置为无限大，而估算时间PredictedTime可为0，从而在进行任务调度时，可以保证就绪队列中至少有一个就绪任务，满足调度要求

3.2 实时任务调度

3.2.4 最小裕度算法

计算任务的富裕时间，称为裕度，裕度越小，优先级越高，以弥补上述情况的不足，即**LLF（least laxity first ）**

在最小裕度算法里，时钟的滴答中断，不但要累计运行任务的执行时间，还要对就绪队列上的任务的裕度进行累减

实际上，前一页公式中的d_1，便是这里所谓的裕度，由于正在运行的任务，其裕度不变，而就绪队列上的任务，其裕度随着时间的推移而减少，从而使得它们的优先权，动态地发生变化

3.2 实时任务调度

3.2.5 其他的实时调度算法

1、价值最高优先算法

在本算法里,每一个任务均有一个价值函数,价值最大的，优先级最高。例如，构造如下的价值函数：

$$V = C(W_1(t - t_s) - W_2 \times d + W_3 \times P - W_4 \times d_1)$$

其中，C为任务的紧急度，t为当前时间，ts为任务的启动时间，d为任务的截止时限，p为任务已执行的时间，d1为执行该任务的裕度，Wj为加权因子。

2、价值比最大优先算法

本算法里，定义一个价值比函数：

$$VD = \frac{v(t + E - p)}{E - p}$$

其中，v为类似上面所定义的价值函数，t为当前时间，E为估算执行时间，p为已执行时间，此时，VD值越大，优先级越高。

3.2 实时任务调度

3.2.6 实时任务的可调度性

假定，响应n个事件的n个任务Ti，各任务的执行时间为Ci，相应周期为Pi。在不考虑系统的其他辅助开销时，对于截止时限优先算法或最小裕度算法，应能满足下列条件：

$$\rho = \sum_{i=1}^{n} \frac{C_i}{P_i} \leqslant 1$$

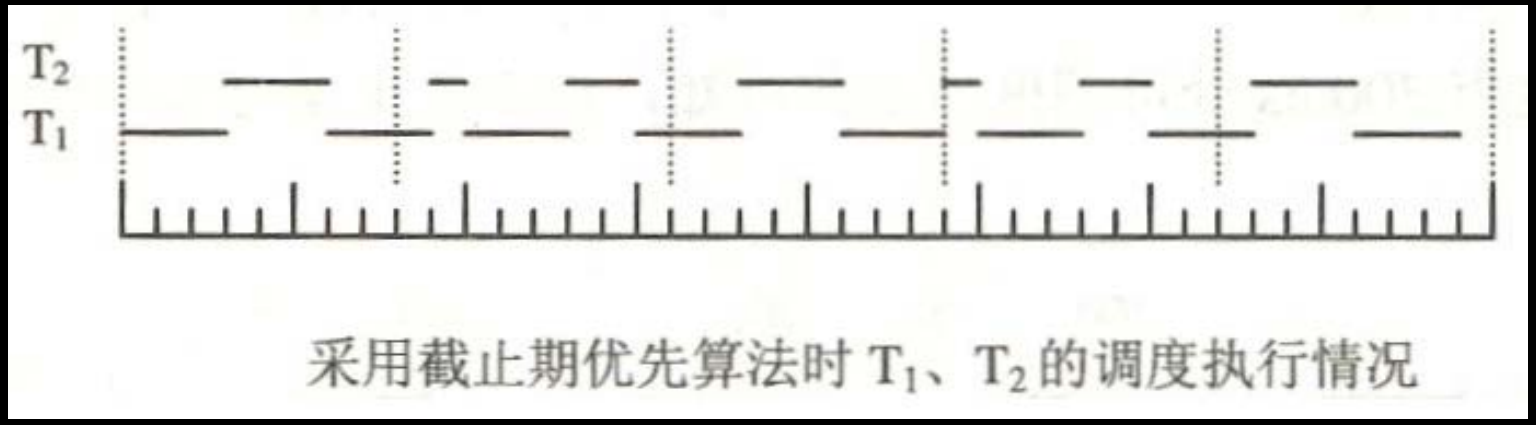
例如，有两个同时发生的周期任务T1和T2，其处理时间C1和C2分别为30ms和20ms，周期p1和p2分别是50ms和70ms，ρ=0.886<1，满足上述时间要求，任务T1在第0ms和第50ms被调度执行2次，而任务T2在第30ms处被调度执行一次，由此推算出，在50×70=3500ms中，任务T1被调度执行7次，任务T2被调度执行5次，每隔3500ms，发生一次相同的调度执行序列。**上述这样的任务集合，对于截止时限优先算法或最小裕度算法**，都是可调度的。

对于单一速率调度算法，任务集的可调度条件则为：

$$\rho = \sum_{i}^{n} \frac{C_i}{P_i} \leqslant n(\sqrt[n]{2} - 1)$$

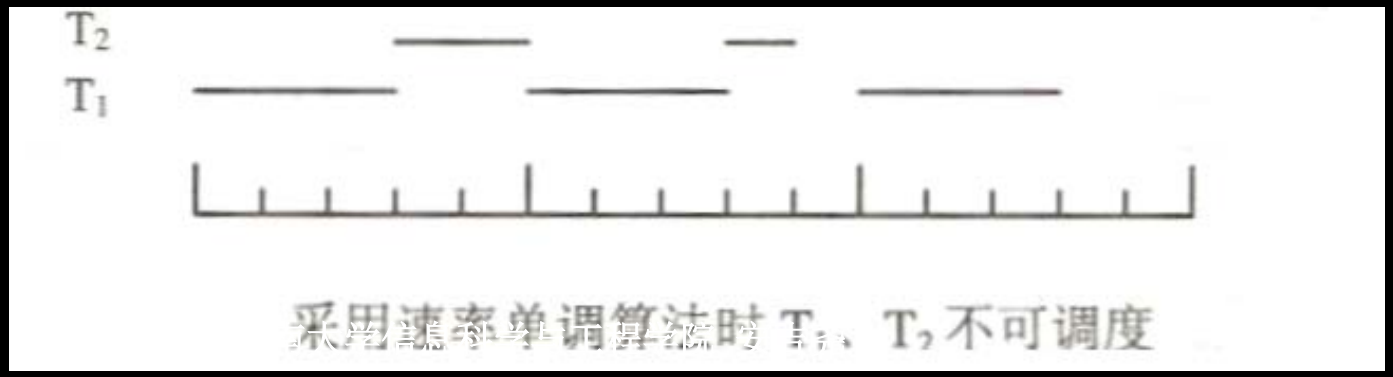
当n=2时， ρ 应小于0.828。例如，有两个周期任务T1和T2同时开始执行，其执行时间C1、C2和周期P1、P2分别为：C1=C2=30ms，P1=50ms，P2=80ms。

如果采用截止时限优先算法或最小裕度算法，则ρ=0.975，满足前述公式，这两个任务是可调度的，其调度执行情况如下图所示：



采用截止期优先算法时 T_1、T_2 的调度执行情况

如果改用优先级固定的单一速率调度算法，则ρ=0.975>0.828，不满足相应公式，这两个任务是不可调度的，调度情况如下图所示：



采用速率单调算法时 T_1、T_2 不可调度

对于上式，甚至有时ρ=0.88的一组任务，采用单一速率调度算法时，仍然是可调度的。例如，在前述例子中，如果把C2和P2分别改为20ms和70ms，而C1和P1不变，这时，ρ=0.886。采用速率单调算法，这两个任务是可以调度的。

在不满足单一调度算法可调度条件的情况下，采用单一速率调度算法，有些任务集仍有可能是可调度的，对于有n个任务的系统，为了检查其可调度性，必须检查其在P1×P2……×Pn时间里，任务的调度执行情况，而这是相当麻烦的。

Jie Wu提出了一个检查可调度性方法，指出：对一组任务，如果所有任务同时开始，每一个任务都能满足它的第一个截止时限，那么，对任意时间开始的组合里，所有任务都能满足其截止时限。

于是提出了**任务的调度点**这一概念。

对任务集的可调度性检查便归结为：在周期最长的任务的第一个调度点之前，如果存在其他某个任务的某个调度点，所有任务都可在这个调度点之前得到调度和执行，则该任务集是可调度的，否则，不可调度

天普大学~吴杰教授，IEEE Fellow

Jie Wu, Ph.D., FIEEE
Director of International Affairs, College of Science and Technology
Director of Center for Networked Computing (CNC)
Laura H. Carnell Professor, Department of Computer and Information Sciences
Temple University

Q1:什么是任务的调度点？
Theorm:可调度性判断原理。

例如，有三个任务T1、T2、T3，其执行时间和周期分别为：C1=40ms、C2=50ms、C3=80ms；P1=100ms，P2=150ms，P3=350ms，则ρ=0.962ms。周期最长的任务T3，其第一个调度点是在350ms处，在此之前的调度点有100ms(对T1)、150ms (对T2)、200ms (对T1)、300ms (对T1和T2)。

在100ms处的调度点：

C1+C2=90<100　　T1、T2可调度

C1+C2+C3=170>100　　T3有10ms的运行时间

在150ms处的调度点：

2C1+C2=130<150　　T1、T2可调度

2C1+C2+C3=210>150　　T3有20ms的运行时间

在200ms处的调度点：

2C1+2C2=180<200　　T1、T2可调度

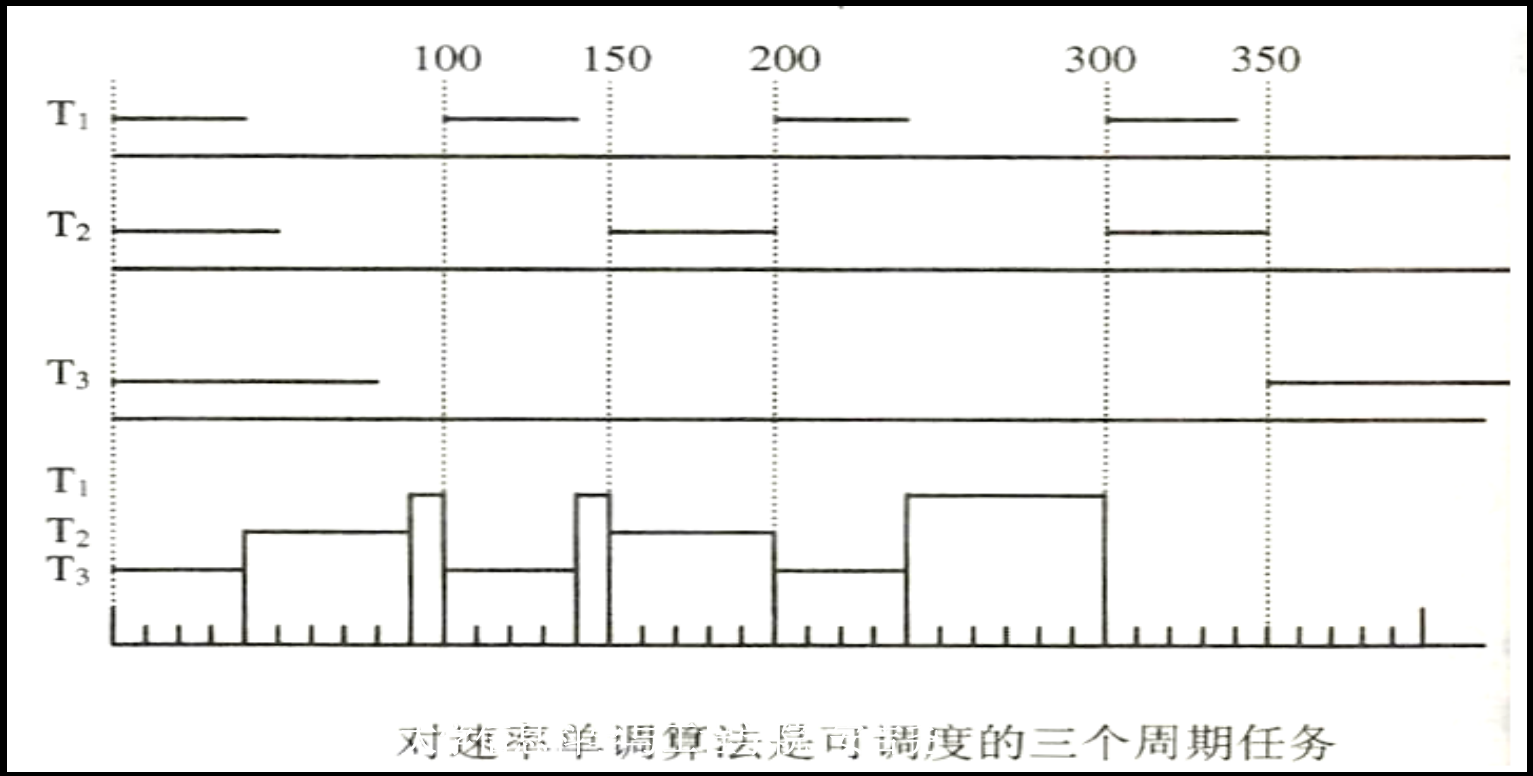
2C1+2C2+C3=260>200　　T3有20ms的运行时间

在300ms处的调度点：

3C1+2C2+C3=300　　T1、T2、T3可调度

T3全部运行结束

任务集在调度点300ms处可调度，由此可知，该任务集是可调度的，其调度顺序如下图所示：



对速率单调算法是可调度的三个周期任务

本讲小结

- 了解实时任务的特点，能描述实时任务
- 理解可调度性判断原理

Homework 3

流式大数据处理的三种Apache框架：Storm，Spark和Samza

Q1，各有什么特点？各有什么优缺点和差异性？

Q2，各有哪些公司产品代表？试举2-3个详细说明。

Q3，Storm –PK- Hadoop