# 实时调度算法研究*

王志平** 熊光泽
(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】 对实时调度进行了讨论；研究了单处理器下的经典调度算法：单调速率调度、最早死线调度和最短空闲时间优先调度；分析了多处理器系统中的典型调度算法；对分布式系统中的两种调度算法：广义单调速率调度和分布式风车调度做了简要论述；指出实时调度研究策略方向。

关 键 词 实时调度； 实时调度算法； 单调速率调度； 最早死线优先； 风车调度
中图分类号 TP301.6

在实时系统中，实时调度是一个被广泛研究的题目。调度的实质是资源的分配，而实时调度强调的是任务的时间约束。实时系统的基本问题就是要保证系统中的任务满足其时间要求，从而保证系统的实时性。

## 1 实时调度分类

实时调度的分类有多种方式。根据建立调度表和可调度性分析是脱机还是联机实现，实时调度可分为静态调度和动态调度；按系统分类，可分为单处理器调度、集中式多处理器调度和分布式调度；按任务是否可抢占，可分为抢占调度和不可抢占调度；而按实时性要求，又可以分为硬实时调度和软实时调度。下面分别按照单处理器、多处理器和分布式调度对实时调度算法进行讨论。

## 2 单处理器实时调度

问题描述：假设一任务集 $S=\{\tau_1, \tau_2,\cdots, \tau_n\}$，周期分别是 $T_1, T_2,\cdots,T_n$，执行时间为 $c_1, c_2,\cdots,c_n$，死线为 $D_1, D_2,\cdots,D_n$，$D_i=T_i$。任务 $\tau_i$ 可以被抢占。

CPU 利用率用 $U=\sum_{i=1}^{n}(c_i/T_i)$ 来表示。对于单处理器，$U\leqslant 1$ 是 $S$ 可调度的前提条件，否则 $S$ 不可调度。

### 2.1 RMS

任务按单调速率优先级分配(RMPA)的调度算法称为单调速率调度(RMS)。RMPA 是指任务的优先级按任务周期 $T$ 来分配。周期短的任务优先级高，周期长的任务优先级则低。RMS 算法有如下定理：

**定理 1** $n$ 个独立的周期性任务可以被 RMPA 调度，如果 $U\leqslant n(2^{1/n}-1)$。

定理 1 中不考虑 $n=1$ 的情况。RMS 是单处理器下的最优静态调度算法[1]。

### 2.2 EDF

最早死线优先算法(EDF)也称为死线驱动调度算法(DDS)，是一种动态调度算法。EDF 指在调度时刻，任务的优先级根据任务的死线动态分配。死线越短，优先级越高。EDF 有如下定理：

**定理 2** 如果一任务集按 EDF 算法可调度，当且仅当 $U\leqslant 1$。

### 2.3 LLF

最短空闲时间优先算法(LLF)也是一种动态调度算法。LLF 指在调度时刻，任务的优先级根据

1999年10月13日收稿
* 电子部预研基金资助项目
** 男 30岁 博士生

任务的空闲时间动态分配。空闲时间越短，优先级越高。空闲时间=死线-任务剩余执行时间。LLF可调度条件与EDF相同(定理2)。

理论上，EDF和LLF算法都是单处理器下的最优调度算法。但由于EDF和LLF在每个调度时刻都要计算任务死线或空闲时间，并根据计算结果改变任务的优先级，因此开销大、不易实现，其应用受到一定的限制。

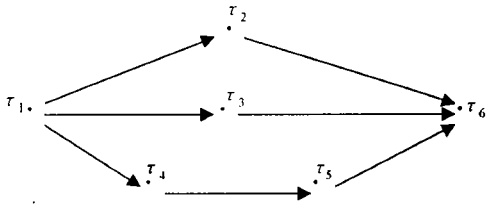


图1 任务集$S$的无环计算图

## 3 多处理器实时调度

### 3.1 静态调度

问题描述：一组具有优先关系的任务在$m$个处理器上运行，任务优先关系用“$\prec$”表示，即如果两个任务($\tau_1$, $\tau_2$)存在优先关系$\tau_1 \prec \tau_2$，则$\tau_1$必须在$\tau_2$开始运行之前完成。任务优先关系可用一个无环图来表示，称为计算图$G$，如图1所示。表示任务集$S=\{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5\}$中任务存在优先关系$\prec = \{(\tau_1, \tau_2), (\tau_1, \tau_3), (\tau_1, \tau_4), (\tau_2, \tau_6), (\tau_3, \tau_6), (\tau_4, \tau_5), (\tau_5, \tau_6)\}$。多处理器静态调度就是要找出长度最短的调度表。

规定如下三种调度规范：1) 基本(或非抢占)调度BS。任务在执行过程中不能被打断。2) 可抢占调度PS，任务可被抢占，这里的抢占不必是基于优先级的。3) 广义调度GS。GS是一个理论上的概念，允许一个处理器可在同一时刻执行多个任务。事实上，每个处理器在某一时刻最多执行一个任务。GS基于的前提为：在给定的时间段，处理器可以将其计算能力的一部分$\alpha(0 \leqslant \alpha \leqslant 1)$分配给任一任务。如果一个任务在某一处理器上花费了$c$个时间单位，而处理器给这个任务分配的计算能力为$\alpha$，则这个任务在处理器上实际运行的时间长度应为$c/\alpha$个时间单位。GS要求：给定$k$个相同的处理器，$\sum \alpha \leqslant k$；同一个任务不可以同时在多于一个处理器上并行执行。

定义 $C_A(G,k)$表示调度规范$A$(BS、PS或GS)下，$k$个处理器的计算图$G$的最小计算时间。

**定理3** 可抢占调度和广义调度的最小计算时间相等即对任一$k$个处理器的计算图$G$，有

$$C_{GS}(G,k)= C_{PS}(G,k)$$

BS不能得到最短调度表；GS虽然可以得到最短调度表，但只是理论结果；由定理3可知，PS和GS具有相同的最小计算时间，而PS在实际中是可实现的。直接求PS调度表比较困难，因此，PS调度表生成过程一般为：1) 根据任务间的优先关系，画出计算图$G$；2) 按$G$生成GS调度表；3) 采用某种方法将GS调度表转换成PS调度表。

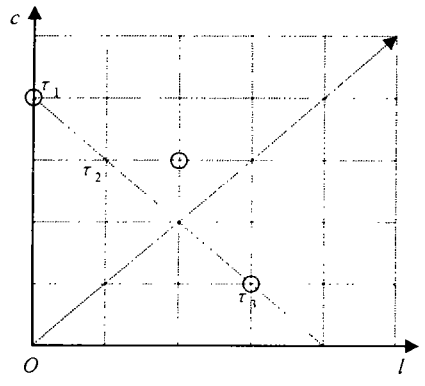


图2 $l$-$c$空间$t$时刻的任务表示

### 3.2 动态调度

问题描述：到达时间不确定而计算时间$c$和死线$D$已知的$n$个任务，运行在$m$个处理器上，$n$不确定，动态调度的目标是使系统能够对变化的环境作出迅速的反应。

#### 3.2.1 $l$-$c$空间

调度任务的状态可以由$l$-$c$空间表示，如图2所示。圆圈表示任务在$l$-$c$空间的位置，即$t$时刻的状态。横坐标表示任务的空闲时间$l(t)$，纵坐标表示任务的剩余计算时间$c(t)$。任务空闲时间$l(t) = D - c(t)$。图中虚线间隔表示一个时间单位，如$\tau_2$的当前时刻的剩余计算时间为3，空闲时间为2，因此死线为3+2=5。

每隔一个时间单位，$l$-$c$空间将刷新一次。任务在$l$-$c$空间的位置变化具有不同的含义：

1) 任务执行：任务向下移动，$c(t)$变小；

2) 任务不执行：任务向左移动，$l(t)$变小；

3) 任务未到达：任务不动。有些任务的计算时间和空闲时间在任务未到达前就已确定，这些任务在 $l$-$c$ 空间预留了位置，但只有到达后才被激活；

4) 新任务到达：根据到达任务的计算时间 $c$ 和空闲时间 $l$，设置其在 $l$-$c$ 空间的坐标位置；

5) 任务执行完毕：任务到达 $l$ 轴，此时 $c(t) = 0$；

6) 任务运行超时失败：任务落在 $c$ 轴左边，此时 $l(t) < 0$。

任务死线属性也在 $l$-$c$ 空间表示出来。由于 $D = l(t) + c(t)$，因此相同死线的任务都位于 125°角的同一直线上，而死线沿 45°角递增。如图 2 所示，$\tau_1$和 $\tau_3$的死线相等，而 $\tau_2$的死线比 $\tau_1$和 $\tau_3$的长。

3.2.2　调度算法

在 $l$-$c$ 空间内，任务按 EDF 或 LLF 算法调度。EDF 或 LLF 在单处理器下是最优调度算法，但在多处理器下则不是[2]。对多处理器动态实时调度，有：

**定理 4**　在多处理器下，如果任务计算时间、死线或到达时间不能预先确定，则最优调度算法不存在。

定理 4 表明基于 $l$-$c$ 空间的多处理器动态调度算法没有最优解。由于该算法不能保证所有任务的死线，因此属于软实时调度。

## 4　分布式实时调度

分布式实时调度算法可以分为两类：1) 以 RMS 为基础的广义 RMS 调度；2) 以风车调度 Sr 为基础的 DSr 调度[3]。

### 4.1　GRMS

GRMS 将 RMS 用于分布式实时系统，对 RMS 的一些基本概念如可调度性、可抢占等进行了扩展，同时还引入一些新的概念，如系统一致性等。

GRMS 算法的实现步骤为：

1) 将子任务分配到各个结点；

2) 将分布式任务的端-端死线分配给每个子任务；

3) 同步每个子任务的周期；

4) 每个结点按 RM 或死线 RM 来调度；5) 进行可调度性检验。

### 4.2　DSr

在一些实时系统中，任务必须以距离约束的方式执行。距离约束是指：同一任务两次相继完成的时间间隔应该总是小于或等于某一段时间，这样的实时系统称为距离约束任务系统(DCTS)。

问题描述：令 $X=\{X_i\}$是一分布式系统的事务(分布式任务)集，$1 \leqslant i \leqslant n$，系统的结点数为 $m$，$\tau_{ij}$是运行在结点 $N_j$上的事务 $X_i$的任务，$\tau_{ij}$的距离约束为 $d_i$，执行时间为 $c_{ij}$。不失一般性，设 $d_i \leqslant d_{i+1}$。另外，假设所有事务都有同样的执行结点顺序，这里的事务由运行在不同结点的一组连续任务组成。

DSr 算法产生多点风车调度周期，可以保证分布式实时系统中任务调度无抖动[4]。DSr 有如下定理：

**定理 5**　给定一个分布式 DCTS 集 $X$，$n$ 个事务运行在 $m$ 个结点上。设 $\rho_j(X) = \sum_{i=1}^{n} c_{ij}/d_i$ 为结点 $N_j$的密度，如果对于 $\forall j$，$\rho_j(\tau) \leqslant 2^{1/(n-1)}$，则 $X$ 用 DSr 可调度。

## 5　结　论

以往对实时调度算法研究主要集中在硬实时、静态调度，并且无论是单处理器调度还是分布式

调度，一般是以 RMS 算法为基础；而动态调度则以 EDF、LLF 为主。随着实时系统朝着开放、分布和多媒体发展，实时应用灵活多变，实时调度算法的研究将主要面向分布、动态、弱实时以及混合调度。

## 参 考 文 献

1 Liu C, Layland J. Scheduling algorithms for multiprogramming in real-time environment. Journal of ACM, 1973, 20(1): 46~61

2 Muntz R R, Coffman E G. Preemptive scheduling of real-time tasks in multiprocessor systems. Journal of the ACM, 1970, 17(2): 324~338

3 Han C C, Lin K J. Scheduling distance-constrained real-time tasks. Proceedings IEEE Real-Time Systems Symposium, NewYork, 1992: 300~308

4 Hsueh C, Lin K J, Fan N. Distributed pinwheel scheduling with end-to-end timing constraints, Proceedings IEEE Real-Time Systems Symposium, 1995: 171~181

5 Sha L, Rajkumar R, Sathaye S. Generalized rate-monotonic scheduling theory: a framework for developing real-time systems. Proceedings of the IEEE, 1994, 82(1): 68~82

6 Nissanke N. Realtime systems. New Jersey: Prentice Hall, 1997: 235~309

# Study of Real-time Scheduling Algorithms

Wang Zhiping Xiong Guangze

(College of Computer Science and Engineering, UEST of China Chengdu 610054)

**Abstract** This paper discusses real-time scheduling. Classic real-time scheduling algorithms in uni--processor systems and two typical multi-processor scheduling algorithms are investigated. Distributed real-time scheduling algorithms are analyzed briefly.

**Key words** real-time scheduling; real-time scheduling algorithms; rate monotonic scheduling; earliest deadline first; pinwheel scheduling