实时系统调度算法综述*

邱相存　臧　洌　杨　丹　董国良
（南京航空航天大学计算机科学与技术学院　南京　210016）

摘　要　在多道程序环境下，主存中有多个进程，其数目往往多于处理机数目。操作系统通过处理机调度程序，按照某种调度算法动态地把处理机分配给就绪队列中的一个进程，使之执行。处理机是重要的计算机资源，提高处理机的利用率及改善系统性能（吞吐量、响应时间），很大程度上取决于处理机调度性能的好坏，因而操作系统的调度算法是非常重要的。通过研究基本的操作系统作业（进程）调度算法，详尽分析和对比这些调度算法的优势和劣势。最后对新兴的实时系统研究现状进行介绍和展望，为以后实时系统调度算法研究提供了有效的参考价值。

关键词　进程；处理机调度；调度算法；实时系统调度

中图分类号　TP316　　**DOI**：10.3969/j.issn1672-9722.2014.12.008

Survey of Scheduling Algorithm in the Real-time System

QIU Xiangcun　ZANG Lie　YANG Dan　DONG Guoliang
(College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing　210016)

Abstract　In a multiprogramming environment, it has multiple processes in main memory, which is often more than the number of processor. Operating system schedules procedures through the processor, according to some scheduling algorithms dynamically distributing the processor to a process from ready queue, and causes it to perform. Processor is an important resource of computer, improving the processor utilization and the system performance(throughput, response time) greatly depend on the stand or fall of processor scheduling performance, so operating system scheduling algorithm is very important. By studying the basic operation system(process) scheduling algorithm, the experimental results detailedly analyze and compare the advantages and disadvantages of the scheduling algorithm. Finally, the new research about real-time system is introduced and predicted. It is valuable for real-time system in the scheduling algorithm research.

Key Words　process, processor scheduling, scheduling algorithm, real-time system scheduling

Class Number　TP316

1　引言

操作系统[1]是管理计算机系统的全部硬件资源包括软件资源以及数据资源；控制程序运行；改善人机界面；为其他应用软件提供支持等，使计算机系统所有资源最大限度地发挥作用，为用户提供方便的、有效的、友善的服务界面。操作系统的形成迄今已有 50 多年的时间。在上世纪 50 年代出现了单道批处理操作系统；60 年代中期产生了多道程序批处理系统；不久后又出现了基于多道程序的分时系统，与此同时也诞生了用于工业控制和武器控制的实时操作系统。20 世纪 80 年代开始至 21 世纪初，是微型机、多处理机和计算机网络高速发展的年代，同时也是微机 OS、多处理机 OS[2]和网络 OS 以及分布式 OS 的形成和大发展的年代。

调度算法作为操作系统的核心研究领域，关乎着一个操作系统性能[3]的好坏。由于大数据、云计算等新兴技术的发展，传统的调度策略不能有效地解决计算性能的要求问题，因此众多的研究学者和科研机构、学术期刊、顶级会议涌现了大批的研究成果。然而国内外综述的研究文献总结得不够全面，本文介绍了几个典型的调度策略，以及实时系

* 收稿日期：2014 年 6 月 3 日，修回日期：2014 年 7 月 23 日
作者简介：邱相存，男，硕士，研究方向：软件工程，实时与嵌入式软件。臧洌，女，副教授，研究方向：软件工程、网络安全。杨丹，女，硕士，研究方向：软件工程。董国良，男，硕士，研究方向：计算机科学与技术。

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

统的调度,试图较为全面地综述近年来的最新研究成果和研究方向,期望对此领域的研究有所帮助。

2 调度方式和调度算法的若干准则

在一个操作系统的设计中,应如何选择调度方式和算法,在很大程度上取决于操作系统的类型及其目标。例如,在批处理系统、分时系统和实时系统中,通常采用不同的调度方式和算法。选择调度方式和算法的准则,有的是面向用户的,有的是面向系统的。

2.1 面向用户的准则

这是为了满足用户的需求所应遵循的一些准则。其中,比较重要的几点如下:

1)周转时间短。通常把周转时间的长短作为评价批处理系统的性能、选择作业调度方式与算法的重要准则之一。其中周转时间,是指从作业被提交给系统开始,到作业完成为止的这段时间间隔。它包括四部分时间:作业在外存后备队列上等待(作业)调度的时间,进程在就绪队列上等待进程调度的时间,进程在CPU上执行的时间,以及进程等待I/O操作完成的时间。其中后三项在一个作业的整个处理过程中可能会发生多次。

对每个用户来说,都希望自己作业的周转时间最短。但作为计算机系统的管理者,则总是希望能使平均周转时间最短,这不仅会有效地提高系统资源的利用率,而且还可使大多数用户都感到满意。可把平均周转时间描述为

$$T=\frac{1}{n}\left[\sum_{i=1}^{n}T_i\right]$$

作业的周转时间 T 与系统为它提供服务的时间 T_s 之比,即 $W=T/T_s$,称为带权周转时间,而平均带权周转时间则可以表示为

$$W=\frac{1}{n}\left[\sum_{i=1}^{n}\frac{T_i}{T_s}\right]$$

2)响应时间快。常把响应时间的长短用来评价分时系统的性能,这是选择分时系统中进程调度算法的重要准则之一。其中响应时间,是从用户通过键盘提交一个请求开始,直至系统首次产生响应为止的时间,或者说,指导屏幕上显示出结果为止的一段时间间隔。它包括三部分时间:从键盘输入的请求信息传送到处理机的时间,处理机对请求信息进行处理的时间,以及将所形成的响应信息回送到终端显示器的时间。

3)截止时间的保证。这是评价实时系统性能的重要指标,因而是选择实时调度算法的重要准则。所谓截止时间,是指某任务必须开始执行的最迟时间,或必须完成的最迟时间。对于严格的实时系统,其调度方式和调度算法必须保证这一点,否则将可能造成难以预测的后果。

4)优先权准则。在批处理、分时和实时系统中选择调度算法时,都可以遵循优先权原则,以便让某些紧急的作业能得到及时处理。在要求严格的场合,往往还须选择抢占式调度方式,才能保证紧急作业得到及时处理。

2.2 面向系统的准则

这是为了满足系统要求而应遵循的一些准则。其中,较重要的几点如下:

1)系统吞吐量高。这是用于评价批处理系统性能的另一个重要指标,因而是选择批处理作业调度的重要准则。由于吞吐量是指在单位时间内系统所完成的作业数,因而它与批处理作业的平均长度具有密切关系。对于大型作业,一般吞吐量约为每小时一道作业;对于中、小型作业,其吞吐量则可能达到数十道作业之多。作业调度的方式和算法对吞吐量的大小也将产生较大影响。事实上,对于同一批作业,若采用了较好的调度方式和算法,则可显著地提高系统的吞吐量。

2)处理机的利用率好。对于大、中型多用户系统,由于CPU价格十分昂贵,致使处理机的利用率成为衡量系统性能的十分重要的指标;而调度方式和算法对处理机的利用率起着十分重要的作用。在实际系统中,CPU的利用率一般在40%(系统负荷较轻)到90%之间。在大、中型系统中,在选择调度方式和算法时,应考虑到这一准则。但对于单用户微机或某些实时系统,此准则就不那么重要了。

3)各类资源的平衡利用。在大、中型系统中,不仅要使处理机的利用率高,而且还应能有效地利用其他各类资源,如内存、外存和I/O设备等。选择适当的调度方式和算法可以保持系统中各类资源都处于忙碌状态。但对于微型机和某些实时系统而言,该准则并不重要。

3 调度算法

调度算法是指根据系统的资源分配策略所规定的资源分配算法。对于不同的系统和系统目标,通常采用不同的调度算法,例如,在批处理系统中,为了照顾为数众多的短作业,应采用短作业优先的调度算法;在分时系统中,为了保证系统具有合理的响应时间,应采用轮转法进行调度。目前存在的

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

多种调度算法中，有的算法适用于作业调度，有的算法适用于进程调度；但也有些调度算法既可以用于作业调度，也可用于进程调度。

3.1　先来先服务调度算法

先来先服务[4](FCFS)调度算法是一种最简单的调度算法，该算法既可以作用于作业调度，也可以用于进程调度。当在作业调度中采用该算法时，每次调度都是从后备作业队列中选择一个或多个最先进入该队列的作业，将它们调入内存，为它们分配资源、创建进程，然后放入就绪队列[5]。在进程调度中采用 FCFS 算法时，则每次调度是从就绪队列中选择一个最先进入该队列的进程，为之分配处理机，使之投入运行。该进程一直运行到完成或者发生某事件而阻塞后才放弃处理机。

FCFS 算法比较有利于长作业(进程)，而不利于短作业(进程)。下表列出了 A、B、C、D 四个作业分别到达系统的时间、要求服务的时间、开始执行的时间以及各自完成时间，并计算出各自的周转时间和带权周转时间。如表 1 所示。

表 1　任务集合 S1(单位：毫秒(ms))

进程名	到达时间	服务时间	开始执行时间	完成时间	周转时间	带权周转时间
A	0	1	0	1	1	1
B	1	100	1	101	100	1
C	2	1	101	102	100	100
D	3	100	102	202	199	1.99

从表 1 可以看出，其中短作业 C 的带权周转时间为 100；而长作业 D 的带权周转时间仅为 1.99。据此可知，FCFS 调度算法有利于 CPU 繁忙型的作业，而不利于 I/O 繁忙型的作业(进程)。CPU 繁忙型作业是指该类作业需要大量的 CPU 时间进行计算，而很少请求 I/O。通常的科学计算属于 CPU 繁忙型作业。I/O 繁忙型作业是指 CPU 进行处理时需频繁地请求 I/O。目前大多数事务处理都属于 I/O 繁忙型作业。

3.2　短作业(进程)优先调度算法

短作业(进程)优先调度算法[6] SJ(P)F，是对短作业或者短进程优先调度的算法。它们可以分别用于作业调度和进程调度。短作业优先(SJF)的调度算法是从后备队列中选择一个或者若干个估计运行时间最短的作业，将它们调入内存运行。而短进程优先(SPF)调度算法则是从就绪队列中选出一个估计运行时间最短的进程，将处理机分配给它，使它立即执行并一直执行到完成，或发生某事件而被阻塞放弃处理机时再重新调度。

我们利用 SJ(P)F 算法进行调度，下表列出了 A、B、C、D 四个作业分别到达系统的时间、要求服务的时间、开始执行的时间以及各自完成时间，并计算出各自的周转时间和带权周转时间。如表 2 所示。

表 2　任务集合 S2(单位：毫秒(ms))

进程名	到达时间	服务时间	开始执行时间	完成时间	周转时间	带权周转时间
A	0	4	0	4	4	1
B	1	3	6	9	8	2.67
C	2	5	9	14	12	2.4
D	3	2	4	6	4	3

从表 2 可以看出，SJ(P)F 调度算法对服务时间较短的作业进程非常有利。例如，短作业进程 D 在 3 的时候进入就绪队列，但是由于其服务时间是剩余三个进程(B、C、D)中服务时间最短的，因此优先得到处理器。

在此，通过一个例子来说明采用 FCFS 算法和 SJ(P)F 算法的调度性能。如表 3 所示有五个进程 A、B、C、D、E。从每个进程的完成时间中减去其到达时间，即得到周转时间，进而可以算出每个进程的带权周转时间。

表 3　任务集合 S(单位：毫秒(ms))

进程名	A	B	C	D	E
到达时间	0	1	2	3	4
服务时间	4	3	5	2	4

根据表 3，得到两种调度算法下，进程 A、B、C、D、E 的完成时间、周转时间和带权周转时间。

由图 1～图 3 可以看出，采用 SJ(P)F 算法后，不论是平均周转时间还是平均带权周转时间，都有较明显的改善，尤其是对短作业 D，其周转时间由原来的(FCFS 调度算法时)11 降为 3；而平均带权周转时间是从 5.5 降到 1.5。这说明 SJF 调度算法能有效地降低作业的平均等待时间，提高系统吞吐量。

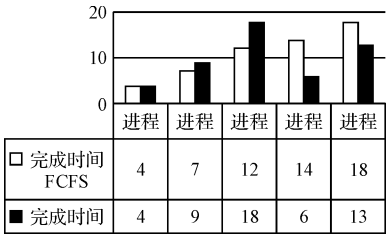


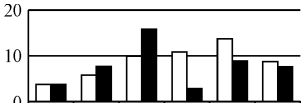
图 1　完成时间对比图

SJ(P)F 调度算法也存在一定的缺点：

1) 该算法对长作业不利，如作业 C 的周转时间由 10 增至 16，其带权周转时间由 2 增至 3.1。因此，如果有一长作业(进程)进入系统的后备队列

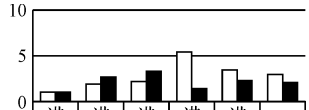
?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

(就绪队列),由于调度程序总是优先调度短作业(进程),将导致长作业(进程)长期得不到调度。



	进程A	进程B	进程C	进程D	进程E	平均
□ 周转时间 FCFS	4	6	10	11	14	9
■ 周转时间 SJ(P)F	4	8	16	3	9	8

图2 周转时间对比图



	进程A	进程B	进程C	进程D	进程E	平均
□ 带权周转时间 FCFS	1	2	2	5.5	3.5	2.8
■ 带权周转时间 SJ(P)F	1	2.67	3.2	1.5	2.25	2.1

图3 带权周转时间对比图

2)该算法完全未考虑作业的紧急程度,因而不能保证紧迫性作业(进程)会被及时处理。

3)由于作业(进程)的长短只是根据用户提供的估计执行时间而定的,而用户又可能有意或无意地缩短其作业的估计运行时间,致使该算法不一定能真正做到短作业优先调度。

3.3 优先权优先调度算法

为了照顾紧迫型作业,使之在进入系统后便获得优先处理,引入了最高优先权优先(FPF)调度算法[7],此算法不仅能用于批处理系统中的作业调度,也可以作为各种操作系统的进程调度。当该算法用于作业调度时,系统将从后备队列中选择若干个优先权最高的作业装入内存。进程调度时,该算法是把处理机分配给就绪队列中优先权最高的进程。

该算法又可分为非抢占式优先权算法和抢占式优先权算法。其中,非抢占式优先权调度算法,系统一旦把处理机分配给就绪队列中优先权最高的进程后,该进程就一直执行下去,直到完成;或因发生某事件该进程放弃处理机时,系统才可以将处理机重新分配给另一优先权最高的进程。这种调度算法主要用于批处理系统中,也可以用于某些实时性要求不严的实时系统中。而抢占式优先权调度算法,系统同样是把优先权分配给优先权最高的进程,使之执行。但在执行期间,只要又出现了另一个优先权更高的进程,进程调度程序就立即停止当前进程的执行,重新将处理机分配给新到的优先权最高的进程。这种调度算法能更好地满足紧迫作业的要求,常用于要求比较严格的实时系统中,以及对性能要求较高的批处理和分时系统中。优先权分为静态优先权和动态优先权。

3.3.1 静态优先权

静态优先权是在创建进程时确定的,且在进程的整个运行期间保持不变。

确定进程优先权的依据主要有以下三个方面:

1)进程类型。一般地,系统进程(如接收进程、对换进程、磁盘I/O进程)的优先权高于一般用户进程的优先权。

2)进程对资源的需求。如进程的估计执行时间以及内存需要量的多少,对这些要求少的进程应赋予较高的优先权。

3)用户要求。根据用户进程的紧迫程度以及用户所付费用的多少来确定优先权的。

静态优先级简单易行,系统开销小,但是不够精确,很可能出现优先级低的作业长期没有被调度的情况。因此,在系统性能要求不高的系统中才使用静态优先权。

3.3.2 动态优先权

动态优先权[8]是在创建进程时所赋予的优先权,是可以随进程的推进或随其等待时间的增加而改变,以便获得更好的性能。例如,可以规定,在就绪队列中的进程随着其等待时间的增长,其优先权以速率 b 提高。若所有的进程都具有相同的优先权初值,则是最先进入到就绪队列的进程将因其动态优先权变得最高而优先得到处理机,此即FCFS算法。若所有的就绪进程具有各不相同的优先权初值,那么,对于优先权初值低的进程,在等待足够时间后,其优先权便可能升为最高,从而获得处理机。当采用抢占式优先权调度算法时,如果再规定当前进程的优先权以速率 c 下降,则可防止一个长作业长期垄断处理机。

3.4 最高响应比优先调度算法

最高响应比优先法(HRN)是对FCFS方式和SJF方式的一种综合平衡。FCFS方式只考虑每个作业的等待时间而未考虑执行时间的长短,而SJF方式只考虑执行时间而未考虑等待时间的长短。因此,这两种调度算法在某些极端情况下会带来某些不便。如果能为每个作业引入前面所述的动态优先权,并使作业的优先级随着等待时间的增加而以速率 a 提高,则长作业在等待一定时间后,必然有机会分配到处理机。HRN调度策略同时考虑每

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

个作业的等待时间长短和估计需要的执行时间长短，从中选出响应比最高的作业投入使用。

响应比 R 定义如下：

$$R=(W+T)/T=1+W/T$$

其中 T 为该作业估计需要的执行时间，W 为作业在后备状态队列中的等待时间。每当进行作业调度时，系统计算每个作业的响应比，选择其中 R 最大者投入执行。这样，即使是长作业，随着等待时间的增加，W/T 也随着增加，也有机会获得调度执行。同时根据上述公式可以看出：

1）如果作业的等待时间相同，则要求服务的时间越短，其优先级越高，因而该算法有利于短作业。

2）当要求服务的时间相同时，作业的优先权取决于其等待时间，等待时间越长，其优先级越高，因而它实现的是先来先服务。

3）对于长作业，作业的优先级可以随等待时间的增加而提高，当其等待时间足够长时，其优先级便可升到很高，从而获得处理机。

总之，该算法既照顾了短作业，又考虑了作业到达的先后次序，不会使长作业长期得不到服务。因此，该算法是介于 FCFS 和 SJF 之间的一种折中算法。然而，在利用该算法时，由于长作业也有机会投入运行，在同一时间内处理的作业数显然要少于 SJF 算法，从而采用 HRN 方法时其吞吐量将小于采用 SJF 方法时的吞吐量。另外，在利用该算法时，每要进行调度之前，都须先做响应比的计算，系统开销也要相应增加。

3.5　轮转调度算法

轮转法(RR)是系统将所有的就绪进程按先来先服务的原则排成一个队列，每次调度时，把 CPU 分配给队首进程，并令其执行一个时间片。时间片的大小从几毫秒到几百毫秒。当执行的时间片用完时，由一个计时器发出时钟中断请求，调度程序便据此信号来停止该进程的执行，并将它送往就绪队列的末尾；然后，再把处理机分配给就绪队列中新的队首进程，同时也让它执行一个时间片。这样就可以保证就绪队列中的所有进程在一给定的时间内均能获得一时间片的处理机执行时间。总之，系统能在给定时间内响应所有用户的请求。

在轮转调度算法中，时间片的大小对系统性能有很大的影响：过长使得每个进程都能在一个时间片完成，时间片轮转算法就退化为 FCFS 算法，无法满足交互式用户的需求；过短将有利于短作业，因为它能较快地完成，但会频繁地发生中断、进程上下文的切换，从而增加系统的开销。一个较为可取的大小是，时间片略大于一次典型交互所需要的时间。一般对时间片的确定要从下面几个方面考虑：

1）对响应时间的要求。T(响应时间)$=N$(进程数目)$*q$(时间片)。

2）就绪进程的数目。数目越多，时间片越小。

3）系统的处理能力。应当使用户输入通常在一个时间片内能处理完，否则响应时间，平均周转时间和平均带权周转时间延长。

表 4 给出了时间片分别为 $q=1$ 和 $q=4$ 时，任务集 S 的 A、B、C、D、E 五个进程的运行情况。

表 4　任务集合 S(单位：毫秒(ms))

进程名	A	B	C	D	E
到达时间	0	1	2	3	4
服务时间	4	3	5	2	4

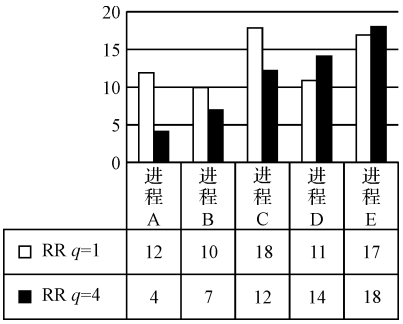


图 4　不同时间片完成时间对比图

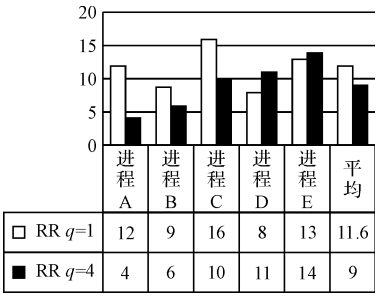


图 5　不同时间片周转时间对比图

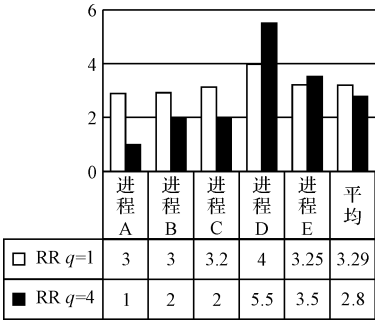


图 6　不同时间片带权周转时间对比图

由图 4～图 6 可知，对于任务集 S 的五个进程 A、B、C、D、E，时间片 $q=4$ 时平均周转时间和平均

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

带权周转时间比时间片 $q=1$ 时效果更好。这说明选择好的时间片对系统的性能至关重要。

3.6 多级反馈队列调度算法

多级反馈队列[9~10](MLFQ)调度算法结合了先来先服务(FIFO)、轮转调度法(RR)、优先级算法和短作业优先(SJF)算法。该算法有多个队列,同一个队列中的进程优先级相同,不同队列中进程的优先级不同;最高优先级上的进程运行一个时间片,次高级上的进程运行两个时间片,再下一级运行4个时间片,以此类推;每当一个进程在一个优先级队列中用完它的时间片后,就下移一级,进入另一个队列;在低优先级队列中等待时间过长的进程,将移入高优先级队列;调度程序在将进程从等待操作中释放后提高该进程的优先级。由于基于多级反馈队列法能够较好地满足全部进程的CPU运行时间,因此该方法具有很大的潜力。

多级反馈队列将所有可运行的任务都分配一个调度优先级,这个优先级决定了它们将放入哪个运行队列中,在选择一个新的任务运行时,系统从最高优先级向最低优先级搜索运行队列,并选出第一个非空队列中的第一个任务,如果一个队列中有多个任务,则系统按先来先服务原则分配处理机,即系统分配以相同的时间量,按照它们在队列中的顺序运行它们,如果一个任务被封锁,它不被放回任何运行队列中。如果一个进程用完分配给它的时间段,则把它放回它原来队列的末端,并选出该队列前端的进程投入运行。时间片越短,交互响应越好。然而,较长的时间片提供了较高的系统总体吞吐量,因为系统减少了用于环境切换的开销,并且处理器的快速缓存将较少刷新。

其模型如图7所示。

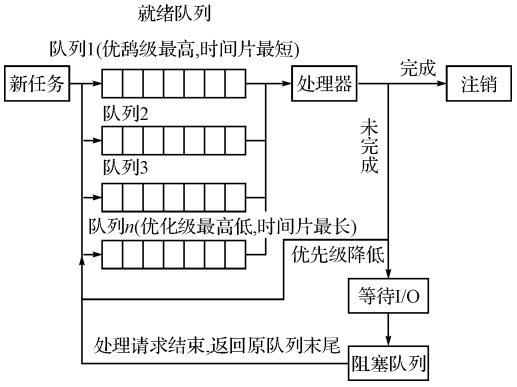


图7 多级队列反馈调度算法

对图7有以下说明:

· 各级就绪队列中的任务具有不同的时间片,优先级最高的第一级队列中执行时间片最小,随着队列的级别增加,其任务的优先级降低了,但是时间片的大小却增加了。队列提高一级,其时间片增加一倍。

· 各级队列均按先来先服务原则排序。

· 当第 $n-1$ 级任务就绪队列为空后调度程序才去调度第 n 级就绪队列中的任务,调度方法相同。所有就绪队列中优先级最低级别队列中的任务是采用时间片轮转调度。

· 当比运行任务更高级别的队列中到达一个新的任务时,它将抢占先运行任务的处理机,而被抢占的任务回到其原队列末尾。

多级反馈队列调度算法具有更好的性能,能更好地满足各种类型用户的需要。

1) 终端型作业用户。由于终端型作业用户所提交的作业大多属于交互作业,作业通常较小,系统只要能使这些作业(进程)在第一队列所规定的时间片内完成,便可使终端型作业用户都感到满意。

2) 短批处理作业用户。对于很短的批处理型作业,开始时像终端型作业一样,如果仅在第一队列中执行一个时间片即可完成,便可获得与终端型作业一样的响应时间。对于稍长的作业,通常也只需要在第二队列和第三队列各执行一个时间片即可完成,其周转时间仍然较短。

3) 长批处理作业用户。对于长作业,它将依次在第 $1,2,\cdots,n$ 个队列中运行,然后再按轮转方式运行,用户不必担心其作业长期得不到处理。

4 实时系统调度

实时系统[11]在国外研究时间比较早。最近几年,国内很多研究所和高校也参与其中,这些研究机构大都集中在硬实时系统领域和实时数据库领域。如实时智能技术的研究主要由中国科学院软件研究所带头;在分布式实时系统及实时调度算法容错性的研发方面,国防科技大学具有优势;而对实时数据库的探索方面华中科技大学则起到了非常大的推动作用。

调度是在可调度性分析的基础上建立调度表然后调度优先权最高的任务执行。如果调度表是预先计算好的,即脱机实现的,称为静态调度;调度表是在线动态计算得到的称为动态调度。将动态调度和静态调度结合在一起的调度称为混合调度。

1) 静态调度

静态调度的典型算法有先进先出算法(FIFO)、

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

单调速率调度算法(RM)、单调时限算法(DM)等。静态调度有以下特点:运行额外开销小、调度的稳定性和可预测性好,分析任务的运行时间相对容易,而且便于在处理机、I/O 设备和通信设备上实现,因此在军事、航空航天、核电站控制、实时仿真、实时工业控制等实际工程中得到广泛应用。最早提出的典型的静态调度算法是 RM 调度算法。静态调度算法多用于周期任务系统或者可以将非周期任务和将闲散任务转化为周期任务的系统。静态调度算法有个明显的优点,不需占用过多的 CPU 时间,但它也有许多缺点:首先,对非周期任务处理的低效。静态调度算法适合周期任务控制领域系统,例如汽车自动控制系统,因为这些系统的任务执行情况和资源利用都是可以预先掌握的。基于这种调度机制的速度一般都很快,并且可以很容易达到调度独立性。然而,这种调度机制却不能保证对环境变化的灵活性。静态调度算法无法有效地保证非周期任务在恶劣环境条件下灵活的响应。对于非周期任务,静态策略也把它当成周期性任务处理,只是这时的周期是非周期任务可能的最短的周期,系统资源也就按照最短的周期进行分配,但大多数情况下,非周期任务并不是以最短的周期运行,这会造成资源的浪费。其次,不能有效处理每次调用的不同,因为任务的优先级一旦决定,运行时刻就不能改变了。而资源的分配必须根据最坏的情况考虑。如某一个操作一般需运行 5ms,但在某特定情况下需要运行 100ms,则静态调度策略考虑最坏的情况,认为这个操作需运行 100ms,这样也就会造成资源的浪费。一般来讲静态调度不能适应实时系统状态配置可变的特征,而且只有在低的资源利用率的情况下,才能提供资源的保证。

2) 动态调度

典型的动态调度算法有最早时限优先调度[14](EDF)、最小松弛优先调度算法(LLF)、最短处理时间优先算法(SPFT)等。动态调度可以在执行过程中依据系统的要求来随时进行调度策略的选择。在大量事件出现以后,依据任务的优先级采用调度算法来确定哪组任务启动或者就绪任务执行,该优先级可以静态分配也可以动态分配。对于 EDF 算法,该抢占式调度算法在单处理机环境下,最高优先级给予任务时限最小的任务。但其局限性是只考虑某任务的时限,而不考虑任务是否可以在这个时限中执行完。LLF 算法是将最高优先级分配给任务执行完毕到时限时间间隔最小的任务。SPFT 算法则是将最高级分配给执行时间最短的任务。动态调度算法都是灵活性很强的调度算法,并且可以推广到非周期任务和闲散任务。虽然动态调度比起静态调度来说更具一般性,但它存在开销大的缺点,这包括通讯延迟、额外内存及执行调度所需的时间。另外,动态调度有可能导致一个任务在处理机之间多次徘徊,消耗更多的时间。并且动态调度算法使用在分布式计算机控制系统中,针对具有预先约束条件的多任务调度、同步机制和互为限制的任务调度也存在缺陷。

3) 混合调度

混合调度结合静态和动态两种调度方式,使这两种调度长短互补,更好地满足系统实时要求,同时也降低系统的调度开销。针对静态调度和动态调度各自的缺点,可考虑把两者结合起来,利用它们各自的优点,形成静、动调度相结合的方案。如最紧急优先策略 MuF(Maximum urgency First):根据任务的紧急程度,对于每个任务分配一个静态优先级。这个静态优先级一旦规定下来,在运行时刻就不可再更改。静态优先级高的任务,它的整个优先级高;对于静态优先级相同的任务,再用松弛度分配其动态子优先级;如果子优先级仍然相同,则再进一步按照任务的重要性分配其静态子优先级。

4) 其他分类方法

根据实时系统是单处理器的还是分布式的,将实时调度可分为单处理器调度和分布式调度。单处理器调度主要是对独立任务集的调度。调度要考虑任务自身的特性,如任务周期、任务执行时间、时限及任务间的约束关系等。分布式调度通常采用综合调度的方法,需要将集中调度和有限的全局调度相组合。对于普通的分布式系统,要考虑任务分配与迁移等问题,但对于嵌入分布式系统,任务分配已经在设计阶段完成,一般也不允许任务迁移,因为它会带来很大的系统开销,不适用于面向应用的有严格时间要求的嵌入式实时系统中。此外,按任务是否可以被抢占,可以分为以下两类:可抢占调度与不可抢占调度。任务可抢占允许较大的调动空间,有利于满足更多任务的调度要求。但可抢占调度由于需要进行额外的任务切换,因而带来一定的运行开销。不可抢占调度不存在任务切换,因此没有运行开销。其优点还有可预测性好、不存在互斥访问共享资源问题等。但它的缺点是不灵活。

目前在实时系统调度算法的研究中,传统调度算法与数学控制理论[23]相结合的研究已经成为了

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved.　http://www.cnki.net

国内外研究者研究的热点,掀起了实时系统任务调度的又一次研究热潮。在相关研究[24~25]中,有效地利用了自适应实时调度技术来增强数字控制系统的性能。根据文献[26],研究人员提出了自适应处理器调度算法、视频流等的服务质量等级管理结构等概念。经过多年的总结验证,已经形成了一整套基于反馈机制的实时任务调度框架体系,为该领域的进一步发展奠定了基础。

5 结语

大数据、云计算等新兴技术的出现,要求操作系统在处理数据方面要求更高,以大数据处理为中心的实时调度算法是支撑计算机性能的基础技术,相关研究具有很重要的理论价值与现实意义。本文详细介绍操作系统的基本调度方式和调度算法,通过具体的实例对比六个基本调度方法,分析了它们在面向用户和面向系统两个准则上的性能差异,给出了各个调度算法的优势和劣势。针对目前新兴实时系统调度(特别是嵌入式系统调度)技术,本文详尽介绍了实时系统调度的最新进展情况,尤其是基于多级反馈队列的实时系统调度,在时间片大小和队列个数如何最优量上面有前景,结合数据控制理论和自适应技术将是今后实时系统研究的工作重点。

参考文献

[1] Tanenbaum A S. Modern operating systems[M]. New Jersey: Prentice Hall Press,2007.

[2] Boyd-Wickizer S, Chen H, Chen R, et al. Corey: An Operating System for Many Cores[C]//OSDI,2008.

[3] Soltesz S, P tzl H, Fiuczynski M E, et al. Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors [C]//ACM SIGOPS Operating Systems Review. ACM,2007.

[4] Sharma R, Soni V K, Mishra M K, et al. An Agent Based Dynamic Resource Scheduling Model with FCFS-Job Grouping Strategy in Grid Computing[C]//International conference on cluster and Grid computing systems(ICCGCS-2010), Italy, Rome,2010.

[5] Adan I, Weiss G. Exact FCFS matching rates for two infinite multitype sequences[J]. Operations Research, 2012,60(2):475-489.

[6] 金瑛棋,吴俊敏,赵小雨. 公平性考虑的短作业优先内存调度策略[J]. 计算机工程,2012,38(20):243-246.
JIN Yingqi, WU Junmin, ZHAO Xiaoyu. Fairness-considered Shortest Job First Strategy for Memory Scheduling[J]. Computer Engineering,2012,38(20): 243-246.

[7] 涂刚,阳富民,卢炎生. 基于动态优先级策略的最优软非周期任务调度算法[J]. 计算机研究与发展,2005,41(11):2026-2034.
TU Gang, YANG Fumin, LU Yansheng. An Optimal Scheduling Algorithm for Soft Aperiodic Tasks[J]. Journal of Computer Research and Development,2005, 41(11):2026-2034.

[8] 丛龙水. 动态优先级作业调度算法与实现[J]. 计算机工程与应用,2013,49(10):267-270.
CONG Longshui. Realization of dynamic priority scheduling algorithm[J]. Computer Engineering and Applications,2013,49(10):267-270.

[9] Parvar M R E, Parvar M E, Safari S. A Starvation Free IMLFQ Scheduling Algorithm Based on Neural Network[J]. International Journal of Computational Intelligence Research,2008,4(1).

[10] Hoganson K. Reducing MLFQ scheduling starvation with feedback and exponential averaging[J]. Journal of Computing Sciences in Colleges,2009,25(2):196-202.

[11] Laplante P A. Real-tiime Systems Design and Analysis[J]. 1993.

[12] Williams R. Real-time systems development[M]. Butterworth-Heinemann,2005.

[13] Wolf W. High-Performance Embedded Computing: Architectures, Applications, and Methodologies[M]. Morgan Kaufmann,2006.

[14] Zhu H., Hansen J. P., Lehoezky J. P., et al. OPtimal Partitioning for quantized EDF scheduling[C]// Proceedings of the 23rd IEEE Real-Time Systems Symposium. IEEE Computer Society,2002.

[15] Natale M. D., Meschi A. Scheduling messages with earliest deadline techniques[J]. The Journal of Real-Time System,2001,20(3):255-285.

[16] M. Caccamo, G. Buttazzo, D. C. Thomas. Efficient reclaiming in reservation based real-time systems with variable execution times[J]. IEEE Transactions on Computers,2005,54(2):198-213.

[17] Albers K, Slomka F. An event stream driven approximation for the analysis of real-time systems[C]//Real-Time Systems, 2004. ECRTS 2004. Proceedings. 16th Euromicro Conference on. IEEE,2004:187-195.

[18] 李江,常葆林. 嵌入式操作系统设计中的若干问题[J]. 计算机工程,2000,26(6):88-89.
LI Jiang. CHANG Baolin. Some Problems in the design of embedded operating system[J]. Computer Engineering,2000,26(6):88-89.

(下转第2266页)

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net

[3] 向德全,孙静娟.军队院校教学评估回顾及新一轮评价思考[J].中国高等教育评估,2010(3):24-27.
XIANG Dequan, SUN Jingjuan. Review and evaluation of a new round of thinking about evaluation of teaching in Military Academies[J]. Higher educational evaluation,2010(3):24-27.

[4] 洪庆根,刘昆福,黄友谊.关于教学工作评价建设常态化的思考[J].海军工程大学学报,2012(9):37-41.
[5] Marsh, H. W. SEEQ: A Reliable, valid, and useful instrument for collecting student's evaluations of university teaching[J]. British Journal of Educational Psychology,1982(1):77-95.

[6] Bloom, Madaus, Hastings.教育评价[M].邱渊,王钢,夏孝川等译.上海:华东师范大学出版社,1990:100-267.

[7] Berk, R. A. Survey of 12 strategies to measure teaching effectiveness[J]. International journal of teaching and learning in higher education,2005(1):48-62.

[8] 聂宏展,聂耸,乔怡,等.基于主成分分析法的输电规划方案综合决策[J].电网技术,2010,34(3):1-3.
NIE Hongzhan, NIE Song, QIAO Yi, et al. Comprehensive Decision-Making of Alternative Transmission Network Planning Based on Principal Component Analysis[J]. Power System Technology,2010,34(3):1-3.

[9] 朱胜伟,周德云,李兆强.基于改进主成分分析法的目标威胁评估[J].计算仿真,2010,27(3):1-3.
ZHU Shengwei, ZHOU Deyun, LI Zhaoqiang. Object Threat Evaluation Based on Improved Principal Components Analysis[J]. Computer Simulation,2010,27(3):1-3.

[10] 豆建斌,卢兴华,杨彭远,等.基于改进主成分分析法的装备水平评估研究[J].指挥控制与仿真,2011,33(1):71-73.
DOU Jianbing, LU Xinghua, YANG Pengyuan, et al. Research on Evaluation for the Standard of Military Equipment Readiness Based on Improved Principal Components Analysis Method[J]. Command Control & Simulation,2011,33(1):71-73.

[11] 胡运权.运筹学教程.[M].北京:清华大学出版社,2003:436-439.
HU Yunquan. Operational Research Tutorial[M]. Beijing: Tsinghua University Press,2003:436-439.

(上接第2258页)

[19] 毛佳,张振花,戴红.实时系统调度算法的优化设计[J].计算机工程与应用,2003,39(15):112-115
MAO Jia, ZHANG Zhenhua, DAI Hong. Optimum Design to Scheduling Algorithm for Real-Time Systems[J]. Computer Engineering and Applications, 2003,39(15):112-115.

[20] 张尧学,方存好,王勇.非精确计算中基于反馈的CPU在线调度算法[J].软件学报,2004,15(4):616-623.
ZHANG Yaoxue, FAN Cunhao, WANG Yong. A Feedback-Driven Online Scheduler for Processes with Imprecise Computing[J]. Journal of Software,2004, 15(4):616-623.

[21] 李建国,陈松乔,鲁志辉.实时异构系统的动态分批优化调度算法[J].计算机学报,2006,29(6):976-984
LI Jianguo, CHEN Songqiao, LU Zhihui. A dynamic Scheduling Algorithm Based on Group Optimization in Real-Time Heterogeneous System[J]. Chinese Journal of Computers,2006,29(6):976-984.

[22] 陈源龙,马培军,李东.硬实时系统中自适应反馈软件容错动态调度算法研究[J].宇航学报,2010,31(11):2591-2596
CHEN Yuanlong, MA Peijun, LI Dong. Dynamic Scheduling Algorithm with Adaptive Feedback Software Fault-Tolerance in Hard Real-Time System[J]. Journal of Astronautics,2010,31(11):2591-2596.

[23] 秦承刚,于东,吴文江.一种面向数控系统的动态反馈调度模型[J].小型微型计算机系统,2010,31(4):788-792
QIN Chenggang, YU Dong, WU Wenjiang. Dynamic Feedback-based Scheduling Model for CNC System [J]. Journal of Chinese Computer Systems,2010,31(4):788-792.

[24] Thidapat Chantem, Xiaobo Sharon Hu, Michael D. Lemmon. Generalized Elastic Scheduling for Real-Time Tasks[J]. IEEE Transactions on Computers, 2009,58(4):480-495.

[25] Qin C, Yu D, Wu W, et al. Control system and elastic scheduling co-design[C]//Information and Automation(ICIA), 2010 IEEE International Conference on. IEEE,2010:1142-1148.

[26] Baochun. Li, Klara. Nahrstedt. Impact of Control Theory on Qos A daptation in Distributed Middle ware Systems[C]//Proceedings of the 2001 American Control Conference(ACC 2001). Arlington, Virginia, 2001:25-27.

?1994-2015 China Academic Journal Electronic Publishing House. All rights reserved. http://www.cnki.net