基于协处理器和动态时间片 RM 调度算法研究

张学军,周 浩,严金童,鲁 友
(南京邮电大学 电子科学与工程学院,江苏 南京 210003)

摘 要: 协处理器与主处理器结合形成的多核心处理系统具有低功耗和高性能的优点,由此形成的异构并行架构可以有效地减轻主处理器的负担,增强系统的实时性和有效性。为了充分利用多核心处理器平台的优势,提高并改进实时系统的调度效率与实现方法,结合单调速率调度算法和时间片轮转算法的优点,提出一种基于动态时间片的单调速率(RM)实时调度算法,并将新的算法应用于无线抄表系统中。实验结果表明,所提出的新算法在复杂通信环境下调度用时接近于理想时间,优于传统实时调度算法。

关键词: 实时调度算法; 协处理器; 动态时间片; 无线抄表系统
中图分类号: TP393 **文献标识码:** A **文章编号:** 1673-629X(2015)03-0188-05
doi: 10.3969/j.issn.1673-629X.2015.03.043

Research on RM Scheduling Algorithm Based on Coprocessor and Dynamic Time Slice

ZHANG Xue-jun, ZHOU Hao, YAN Jin-tong, LU You
(School of Electronic Science and Engineering, Nanjing University of Posts and Telecommunications, Nanjing 210003, China)

Abstract: The coprocessor and the main processor are combined to form the multi-core processor system, which has the advantages of low power consumption and high performance. The heterogeneous parallel architecture can effectively reduce the burden of the main processor, enhancing timeliness and effectiveness of the system. In order to make full use of the advantages of multi-core platform, improve the scheduling efficiency and implementation method of real-time system, a new Rate Monotonic (RM) real-time scheduling algorithm based on dynamic time slice is proposed. It combines the advantages of speed-adjustment scheduling algorithm and the round-robin scheduling algorithm. The new algorithm is applied to the wireless meter reading system. Experimental results show that the proposed algorithm makes the time of scheduling in complex communication environment obviously close to the ideal time, which is superior to the traditional algorithm.

Key words: real-time scheduling algorithms; coprocessor; dynamic time slice; wireless meter system

0 引 言

随着科技的发展,广泛应用于工业控制领域的实时系统越来越深入复杂民用领域,这也对处理器的调度和处理能力提出了较高的要求。

传统的单核心处理器与系统调度算法已经不能满足复杂应用的需求。多核心处理结构和与之适配的多核心调度算法能为复杂的应用提供保障。

文中提出了一种基于动态时间片的单调速率实时调度算法,并将其应用于无线抄表系统中,结果表明该算法优于传统实时调度算法。

1 协处理器与实时调度算法

所谓协处理器,就是在主处理器旁增设一个效能较低的处理器,两者之间通过高效的通信网络相连接,主处理器和协处理器可以并行地执行任务。为了提高系统效率,使要执行的任务能在尽量短的时间内完成,合理地安排任务执行顺序和给任务安排合适的处理器是要考虑的关键因素。

在多处理器的并行处理环境的模型中,每个任务只能分配给一个处理器。在实际应用中,这种关系可以用有向无环图(Directed Acyclic Graph, DAG)来表

收稿日期: 2014-05-07 **修回日期:** 2014-08-11 **网络出版时间:** 2015-01-20
基金项目: 江苏省高校自然科学基金(08KJB510015); 华为高校科技基金(YJCB2008039WL)
作者简介: 张学军(1969-),男,教授,博士,研究方向为认知网络频谱感知、无线射频识别技术、嵌入式电子系统设计等; 周 浩(1990-),男,硕士研究生,研究方向为嵌入式电子系统设计等。
网络出版地址: http://www.cnki.net/kcms/detail/61.1450.TP.20150120.2202.037.html

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

示: 有向无环图中的每个节点代表一个任务，节点的权值是这个任务在不同处理器上执行所需时间的平均值[1]。

多处理器有局部调度和全局调度两类实时调度方法。局部调度将任务拆分成进程，将特定的进程分配给特定处理器，调度是局部化的、进程化的。全局调度将所有等待调度的任务组成全局等待队列，等待处理器调度，处理器允许调度任务时，选择优先级最高的任务执行，调度是全局化的、任务化的[2]。

1.1　局部调度算法改进

1.1.1　问题模型

假设任务集: $T_S = \{T_1, T_2, \cdots, T_n\}$，处理核心 O_1 和 O_2，两者之间的传输带宽为 B。在对任务集中的任务进行先后排序的时候，需要使用全局调度算法。当决定具体的任务排序后，对每个任务的子进程进行处理器分配的时候，需要使用局部调度算法。这里首先对局部调度算法进行改进。

任务 T_i 由进程 Q_j 和进程 Q_k 两个先后进程构成。Q_j 是 Q_k 的前置进程，即 Q_j 必须在 Q_k 执行之前执行完毕，并将 Q_k 所需要的数据 $D(j,k)$ 及时传送给它。在这个过程中，共产生了以下几个不可预知的时间量: Q_j 的执行时间 T_j，数据的传输时间 T_t(代价)，Q_k 的执行时间 T_k。则在有向图中，节点 T_i 的值 C_i 通过以下公式计算:

$$C_i = T_j + T_t + T_k \tag{1}$$

$$T_t = D(j,k)/B \tag{2}$$

$$C_i = T_j + D(j,k)/B + T_k \tag{3}$$

由此可知，每个任务 T_i 在两个处理器的平均处理时间为:

$$W_i = C_i/2 \tag{4}$$

显而易见，T_t 已经由硬件决定，改变的可能性不大，唯有降低 T_j 和 T_k 是提高系统性能的关键。另外，以上分析均是假定进程数据传输完毕后，后置进程 Q_k 立即执行[1]。

图 1 为局部调度示意图。

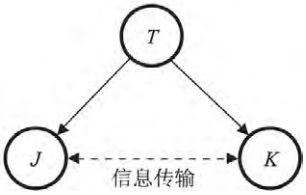

图 1　局部调度示意图

图 1 中的每个任务都拥有以下两个性质:

(1) 设定任务 T_i 的最早开始时间为 EST(i)(Earliest Start Time)，表示一个任务开始执行时间的下界。

$$\mathrm{EST}(0) = 0 \tag{5}$$

$$\mathrm{EST}(i) = \mathrm{EST}(i-1) + C(i-1) = \mathrm{EST}(i-1) + 2W(i-1) \tag{6}$$

(2) 设定任务 T_i 的最晚开始时间为 LST(i)(Last Start Time)，表示最后一个任务结束完成时，任意一个任务开始执行时间的上界。假设有 n 个任务。

$$\mathrm{LST}(n) = \mathrm{EST}(n) \tag{7}$$

1.1.2　算法步骤

以 ARM® Cortex - M4 和 Cortex - M0 双核架构的异构并行微控制器为例，核心 Cortex - M4 性能强劲，对于大数据特别是 DSP 信号的处理尤为突出，Cortex - M0 处理能耗非常低、适用于调度和 IO 数据传输。新算法有两个阶段: C_i 权值计算与子进程分配。新算法的描述如下:

步骤 1: 由式(3)计算当前任务的 C_i 值，此时将任务分拆成前置进程 A 和后置进程 B;

步骤 2: 对于需要执行时间较长、执行条件复杂的后置进程 B，分配到 M4 内核的就绪队列，进程 A 分配给 M0 内核的执行队列;

步骤 3: 进程 A 执行完毕后，M0 内核将数据传送到 M4 内核，此时 M0 内核开始执行下一个任务的进程 A，而 M4 在执行上一个任务的进程 B。

在算法执行过程中，M0 内核的任务主要是执行前置任务并负责数据的传输，将繁重的任务交给 M4 内核，并且在 M4 内核执行时，已经为其准备好了下一个任务的前置数据，充分利用了协处理器的优越性。

1.2　全局调度算法改进

RM 调度算法是一种典型的静态优先级调度算法，它根据任务执行周期的长短来决定调度优先级，执行周期小的任务具有较高的优先级。这种调度方式主要适合周期性任务，对于突发性任务，无法预先估计执行时间，因而会导致系统的执行效率很低。对于给定任务集合$\{S_1, S_2, \cdots, S_n\}$，调度器的利用率必须满足

$$\frac{C_1}{T_1} + \frac{C_2}{T_2} + \cdots + \frac{C_n}{T_n} \leqslant n(2^{1/n} - 1) = L(n) \tag{8}$$

式中，C_n 表示任务 S_n 的执行时间; T_n 表示任务 S_n 的执行周期; $L(n)$ 表示调度器利用率的最下界。当 $n \to \infty$ 时，$L(n) \to \ln 2 \approx 0.693$。整个任务集的负载小于 $L(n)$ 时，RM 算法调度是可行的[3]。

引用 RM 实时调度算法的思想，用时越短的任务优先级越高，但由于没有时间片或其他终止策略的设置，如果一个实时任务由于某种自身原因或者环境原因超出了自己的执行时间，任务却没有完成，则该任务将保持当前的高优先级执行下去，直到执行完毕。这样就会顺延给后续任务，形成多米诺效应，造成多个任务超出截止时间。在实际情况中特别是无线通信传输中很容易发生过载时，导致控制器调度工作大增，性能

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

退化[4]。

结合单调速率调度算法和时间片轮转算法的优点，对系统实时调度算法进行改进，提出了一种基于动态时间片的单调速率实时调度算法（Rate Monotonic real - time scheduling algorithm based on Dynamic Time Slice，DTS - RM）。

以无线抄表系统的实际应用为例。节点所处环境信噪比高，则传输用时较短；节点所处的环境信噪比低，则传输用时较长。因此，将每个传输任务作为一个任务，信噪比高，则设置该任务优先级高，反之，则设置低的优先级[5]。DTS - RM 算法步骤为：

步骤 1：注册。

当手持机进入小区后，首先对小区所有设备发送广播帧，提醒各个仪表设备进行注册。手持机发送广播后，可能会有多个仪表同时向手持机发送注册帧。为了减低碰撞的发生，采用经典的 ALOHA 算法。ALOHA 算法是一种随机接入型的算法[6-7]，其工作原理：手持机发送广播帧，接收到广播帧的仪表产生注册帧，并立即发送到无线信道中；若在规定时间内收到手持机的确认应答，表示注册成功，否则执行重发策略，即随机退避若干个时序后再向手持机发送注册帧，直到注册成功。

步骤 2：数据传输。

（1）设置多个隔离的就绪队列，依据不同的优先级，将实时任务分配到几个不同级别的队列中。

（2）各个队列按照级别先后进入处理机，高优先级的队列，任务的初始时间片设置的较短，低优先级的队列，任务的初始时间片设置的较长。如图 2 所示，具体设置在 2.1 节中详细介绍。

（3）第一个任务时间片是初始时间片，队列中剩余的每个任务的时间片动态调整。若在时间片内，前一个任务完成，则将接下来任务的时间片减小 0.05 s，以此类推，渐次减小。若一旦遇到某个任务在时间片内无法完成，时间片的设置开始每次增大 0.05 s，直到任务能够完成。

（4）如果队列上的某个实时任务运行超过分配运行时间，若继续运行下去，有可能造成整个队列的延时，则放弃该任务，并记录其 ID。

（5）在所有队列均执行完毕后，对传输失败的任务依据记录的 ID 依次进行轮询通信。

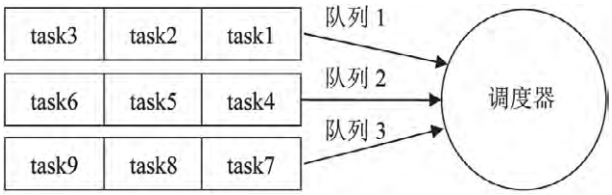

图 2 基于时间片的 RM 实时调度算法队列示意图

2 算法实现

2.1 变量的设置

将该改进型实时调度算法应用到实际系统中，需要在系统代码中增加队列优先级和任务时间片等变量。在无线抄表系统中，优先级由该仪表所处的无线环境的好坏决定，量化到数值上就是信噪比。由于该系统采用的无线通信芯片能够测算当前的传输环境的信噪比并形成数值，可在仪表的注册帧中加入 3 bit 位的信号强度位。将 000 到 111 生成如表 1 所示的量化等级表[8]。

表 1 量化等级表

优先级	量化等级	信号强度（dBmW）
1	000（0）	-66 ~ 无穷大
2	001（1）	-70 ~ -67
3	010（2）	-74 ~ -71
4	011（3）	-78 ~ -75
5	100（4）	-82 ~ -79
6	101（5）	-86 ~ -83
7	110（6）	-90 ~ -87
8	111（7）	无穷小 ~ -90

手持机在接收各个仪表设备的注册帧后，可以根据注册帧中含有的信号强度量化值，将所有仪表分成多个就绪队列，其中信号强度高的仪表队列被赋予高优先级，信号强度低的仪表队列被赋予低优先级。

每个队列的初始时间片 T 与该序列的优先级 P 的关系为：

$$T = kP(k\text{ 为常数}) \tag{9}$$

经过大量的实际测试，发现在信号强度正常的情况下，每个任务传输完毕需要大约 0.8 s。设优先级 P 的取值从 1 到 8，则 k 取 0.25 比较合适。

2.2 数据帧的设置

在该调度算法中，一个队列对应多个任务，系统要构造一个结构体 OS_line，用来控制队列，称作队列控制块[9]，主要包含下列元素：

（1）OSPrio：队列的优先级，占 3 bits。

（2）line_Stat：队列的状态，如果该队列处于就绪状态，那么这个变量就置为就绪，占 2 bits。

（3）Count：当前优先级下含有的任务数，占 9 bits。

（4）Tcount：当前队列的初始时间片大小，占 3 bits。

当一个队列得到调度，进入 CPU 执行后，任务按照时间片轮转调度，同时时间片做动态调整。因此系统中还需要加入一个结构体 OS_task，用来控制任务，称作任务控制块，主要包含下列元素：

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

(1) task_ID: 当前任务的 ID,占 10 bits。

(2) task_Stat: 当前任务的状态,如果该状态处于就绪状态,那么这个变量就置为就绪,占 2 bits。

(3) task_Tcount: 当前任务的时间片大小,占 3 bits。

3　改进算法在无线抄表系统中的应用

3.1　无线抄表系统硬件设计

无线抄表系统的硬件主要包括计量表数据发射装置和手持式数据接收机两部分。二者硬件结构基本一致,由 LPC43XX 系列双核微控制、SI4432 无线发射、SPI 总线等核心部分构成,如图 3 所示。

图 3　硬件示意图

LPC43XX 是全球首款采用 ARM® Cortex™ - M4 和 Cortex™ - M0 双核架构的异构并行微控制器。LPC43XX 系列 ARM 控制器为 DSP 和 MCU 应用开发提供了单一的架构和环境。Cortex - M0 子系统处理器可分担 Cortex - M4 处理器大量数据传输和 I/O 处理任务,减小 Cortex - M4 内核的带宽占用。

SI4432 无线收发芯片提供的频率范围为 240 MHz 到 930 MHz,可调输出功率达 +20 dBm,支持调频扩频,并提供了自动唤醒定时器、信道强度评估、低电量检测器、64 字节发射/接收、自动数据包处理等功能,同时还支持 SPI 总线连接。

3.2　μC/OS - II 实时内核的移植

μC/OS - II 内核是一个开源、抢占式的内核[10],它包含了操作系统的任务调度、任务管理、时间管理、内存管理、定时管理以及任务通信与同步等基本的特性[11]。其优先级抢占策略,能够保证系统的实时性。μC/OS - II 内核中,一般只需要移植 os_cpu.h、os_cpu_a.asm 和 os_cpu_c.c 文件,大大降低了系统移植难度。μC/OS 的定时中断确保了系统的实时性,每个时钟节拍到来就会产生一次定时中断,中断后进行任务调度,运行就绪表中优先级最高的任务。系统架构如图 4 所示[12]。

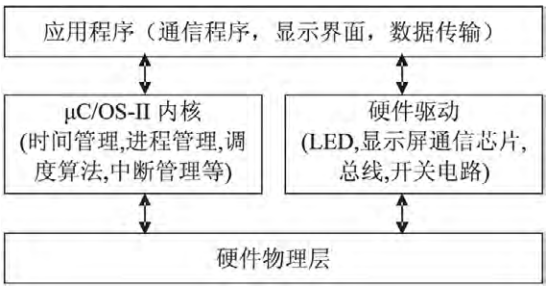

图 4　系统架构

3.3　测试分析

在不同的信噪比环境下,对仪表与手持机单独通信成功所需要的时间进行了多次测试,实验数据如表 2 所示。

表 2　在不同信噪比下仪表与手持机通信时间

信噪比/dB	优先级	平均完成时间/s
-65	0	0.2
-73	2	0.35
-81	4	0.6
-88	6	0.7
-92	7	NA

注: NA 表示多次未成功通信。

由表 2 可见,随着信噪比的减小,通信用时逐渐变长,在低信噪比时,完成所需时间陡然变大,在信噪比接近 -100 dB 时,出现了失败通信。因此,选取前 4 个信噪比环境,分别在 0.2 s,0.35 s,0.6 s,0.7 s 数据周围,随机选取 6 个数值,共生成 24 个数据,排成 4 个队列的仿真数据。

为了模拟真实通信环境中随机出现的通信延时与干扰,在测试程序中人为添加干扰延时,信噪比较低时,干扰延时发生的概率加大,如表 3 所示。

表 3　仿真数据

队列	6 个任务完成时间/s					
队列 1	0.29	0.24	0.22	0.19	0.24	0.28
队列 2	0.31	0.33	0.36	0.38	0.40	0.44
队列 3	0.55	0.55	0.66	0.65	0.71	0.53
队列 4	0.70	0.67	0.73	0.66	0.69	0.71

本次测试中,全部任务完成最少需要 12.5 s 左右。将此 24 个任务打乱,形成 5 种不同的顺序方案,导入基于动态时间片的 RM 实时调度程序和传统的 RM 实时调度程序中,以此模拟现实通信中,全部数据点均发起通信的情况。统计 5 次完成所有任务需要的时间,曲线如图 5 所示。传统 RM 实时调度算法 5 次测试用时分别为: 15.50 s,14.80 s,15.15 s,15.30 s,14.98 s,平均用时 15.146 s。基于动态时间片的 RM 调度算法 5 次测试用时分别为: 14.12 s,15.05 s,14.95 s,14.65 s,13.95 s,平均用时 14.544 s,比传统 RM 算法的性能平均提高了约 4%。从图 5 可以看出,5 次不同的任务排序,基于动态时间片的 RM 实时调度算法完成的时间各不相同,说明不同的任务排序对完成时间是有影响的。但在 5 次测试中,基于动态时间片的 RM 实时调度算法的完成时间最差仅比最少时间多 2.55 s。

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

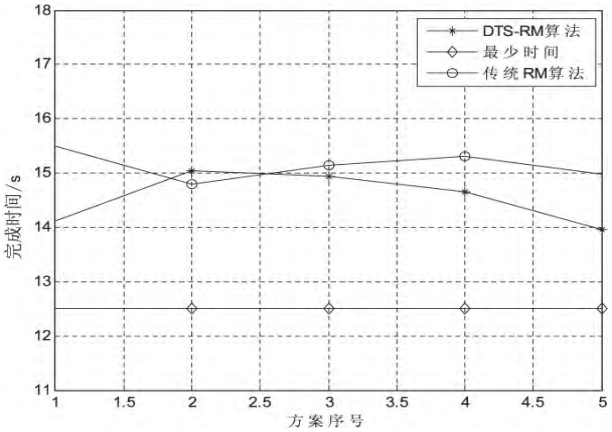

图 5 模拟测试

为了模拟实际通信环境中随机出现的通信延时与干扰，在测试程序中人为添加干扰延时，信噪比较低时，干扰延时发生的概率加大[13-14]。为了对比明显，将第 2 次测试中的干扰延时出现的次数减少，使之明显低于现实通信环境中的干扰。测试表明，改进的算法未发生作用且由于时间片的设置，用时较长，所以在图 5 中第 2 次测试时改进算法用时较传统 RM 算法略长。在第 3、4、5 次测试中，将干扰延时发生的概率再次调整到正常范围以内。

测试结果表明，基于动态时间片的 RM 实时调度算法的性能优于传统 RM 实时调度算法。传统的 RM 实时调度程序在运行程序时，虽然在优先级比较高即环境信噪比比较好时性能很高，但一旦遇到低信噪比任务，任务完成很容易超时，导致后续任务不断延期，性能退化快。

4 结束语

文中提出了基于协处理器与动态时间片的 RM 实时调度算法，该算法对多核平台上的实时任务，运用全局调度与局部调度的混合调度，并且引入时间片轮转算法。

实验结果表明，该算法在运用到无线抄表系统后，表现出更加有效率的调度性能。当然，该算法还有比较大的提升空间，例如在对时间片的处理上可以使用更负责的算法，双核处理器的功能也有待开发，等等。

参考文献:

[1] 蒋韵联，孙广中，许胤龙. 并行异构系统中的一种高效任务调度算法[J]. 计算机工程，2007，33(11)：39－41.

[2] 张惠娟，翟鸿鸣，周利华. 多处理器系统的实时调度算法研究[J]. 计算机工程与设计，2004，25(8)：1233－1235.

[3] Liu C L，Layland J W. Scheduling algorithms for multiprogramming in a hard－real－time environment[J]. Journal of the ACM，1973，20(1)：46－61.

[4] 邢群科，郝红卫，温天江. 两种经典实时调度算法的研究与实现[J]. 计算机工程与设计，2006，27(1)：117－119.

[5] 唐毓毅，朱怡安，黄姝娟，等. 一种有约束关系的实时周期任务调度算法研究[J]. 计算机技术与发展，2013，23(7)：1－5.

[6] Zhen Bin，Kobayashi M，Shimizu M. Framed ALOHA for multiple RFID objects identification[J]. IEICE Transactions on Communications，2005，E88－B(3)：991－999.

[7] Lee S R，Joo S D，Lee C W. An enhanced dynamic framed slotted ALOHA algorithm for RFID tag identification[C]//Proc of the second international conference on mobile and ubiquitous systems: networking and services. [s. l.]: IEEE，2005: 166－172.

[8] 田 冲，周井泉. 一种基于加权公平队列调度的改进型算法[J]. 计算机技术与发展，2013，23(6)：71－73.

[9] 吕 方，崔慧敏，霍 玮，等. 面向并发性能下降的调度策略的综述[J]. 计算机研究与发展，2014，51(1)：17－30.

[10] 任 哲. 嵌入式实时操作系统 μC/OS－II 原理及应用[M]. 第 2 版. 北京: 北京航空航天大学出版社，2009: 12－18.

[11] 陈是知. μC/OS－II 内核分析、移植与驱动程序开发[M]. 北京: 人民邮电出版社，2007: 20－25.

[12] Ravat B，Diwan A，Reddy P S K. Development of an arm based modbus RTU to TCP/IP protocol converter using μC/OS II[J]. Networking and Communication Engineering，2013，5(6)：271－274.

[13] Rezaeian A，Naghibzadeh M，Neamatollahi P. Scheduling hard real－time tasks on multi－core using intelligent rate－monotonic[C]//Proc of 3rd international conference on computer and knowledge engineering. Mashhad: IEEE，2013: 449－453.

[14] 宋 杰，檀林欣，曹竹冬，等. 一种新型的实时调度算法[J]. 计算机技术与发展，2010，20(12)：73－76.

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