文章编号: 1001-0920 (2011) 11-1740-05

多处理器系统动态调度负载均衡节约算法

王遵彤, 李 彩, 吴启迪

(同济大学 电子与信息工程学院, 上海 201804)

摘 要: 将任务集与处理器处理能力之间的匹配关系作为研究调度算法性能的重要因素, 建立了相应的任务-处理器模型, 以描述多处理器系统的负载状况. 描述了多处理器系统任务可调度的必要条件, 设计实现了任务集的生成方法. 对节约算法进行改进, 提出了负载均衡的节约算法. 所提出的算法可在保证调度成功率的前提下, 缩短任务的平均响应时间和调度长度, 并均衡地提高处理器的利用率.

关键词: 多处理器系统; 可调度性; 动态调度; 负载均衡; 调度算法

中图分类号: TP273 **文献标识码:** A

Load-balancing thrift algorithm for dynamic scheduling of multi-processor systems

WANG Zun-tong, *LI Cai*, *WU Qi-di*

(College of Electrical and Information Engineering, Tongji University, Shanghai 201804, China. Correspondent: WANG Zun-tong, E-mail: wangzt@tongji.edu.cn)

Abstract: The performance of scheduling algorithm is affected remarkably by the processing capability of processors and the size of tasks. Therefore, a task-processor model is proposed to describe the load status of the multi-processor systems, which reflects whether the tasks to be processed exceed the processing capability of the systems. A prerequisite describing the schedulability of multi-processor systems is presented, and a method is designed to generate the random task set. Based on the thrift algorithm, load-balancing thrift algorithm is proposed, to schedule multi-processor systems, which can simultaneously increase the success rate of scheduling, reduce the average response time of tasks and schedule length, and improve the utility of the processors evenly.

Key words: multi-processor systems; schedulability; dynamic scheduling; load balancing; scheduling algorithm

1 引 言

日益复杂的多媒体、科学计算以及虚拟化等应用领域都需要更为强大的计算能力, 多处理器系统(如片上多处理器, CMP) 已经成为人们广泛接受的研发热点[1]. 其任务调度[2]就是通过调度算法将任务分派到不同的处理器执行, 以使系统的性能得到最大程度的优化. 实际上, 由于任务具有随机性, 不能找到全局最优算法来解决这类动态调度问题, 所以启发式算法得到了广泛使用, 如 Ramamritham 等人[3]基于启发式搜索, 提出了针对动态实时多处理器调度的近视算法 (MA); Manimaran 等人[4]在 MA 的基础上, 利用任务的并行性来提高调度算法的性能; 乔颖等人[5]在 MA 的基础上提出了节约算法 (TA); 李建国等人[6]则提出了为一批任务分配处理器的动态分批调度算法. 上述算法均以调度成功率为衡量标准来与其他算法进行比较和分析, 没有考虑算法对处理器利用率、任务响应时间等其他性能指标的影响, 而且未考虑任务数量对调度性能的影响.

本文将任务集与处理能力的匹配关系作为研究调度算法的重要因素, 首先对处理器-任务模型进行描述, 提出了可调度任务集的生成方法; 在节约算法[5]的基础上进行改进, 提出了负载均衡的节约算法, 并通过仿真比较了 2 种算法的调度成功率、平均响应时间、处理器利用率以及调度长度等指标.

2 多处理器系统动态调度模型

2.1 问题描述

多处理器系统动态调度是指在一个含有 m 个处

收稿日期: 2010-07-09; **修回日期:** 2011-01-08.

基金项目: 国家自然科学基金项目(60674076).

作者简介: 王遵彤(1965−), 男, 副教授, 博士, 从事复杂系统调度与优化、SoC 设计方法等研究; 李彩(1986−), 女, 硕士生, 从事CPU调度算法的研究.

理器及有限资源(如Cache等)的系统中, 为n个随机出现的任务分配具体的处理器及其他资源, 以保证其时间约束和资源约束同时得到满足[7]. 如果一个任务的时间约束和资源需求都得到满足, 则称该任务是可调度的; 如果一个任务集的所有任务在调度中都是可行的, 则称该调度对于此任务集是一个可行调度, 或称该任务集是可调度的[3-4].

多处理器系统的调度模型主要包括处理器模型、任务模型和调度算法3部分[8]. 处理器模型描述处理器结构和处理能力等信息, 任务模型描述调度等待处理的任务所需要的相关信息.

2.2 多处理器系统

设一个多处理器系统由m个处理器$(P_1, P_2, \cdots, P_m)$及K个资源$(\mathrm{Res}=r_1, \cdots, r_K)$组成, 处理能力$C_i\ (i=1,2,\cdots,m)$表示处理器$P_i$在单位时间内处理任务的能力, 多处理器系统的处理能力(TPC)为

$$\mathrm{TPC}=\sum_{i=1}^{m} C_i. \tag{1}$$

2.3 任 务 集

多处理器系统的任务集$S=\{s_1, s_2, \cdots, s_n\}$描述如下:

1) 任意$s_i \in S$可表示为一个多元组$s_i=\{at(i), st(i), pt(i), dt(i), et(i)\}, i=1,2,\cdots,n$. 其中: $at(i)$为任务s_i的到达时间; $st(i)$为任务s_i开始处理的时间; $pt(i)$为任务s_i所需的处理时间; $dt(i)$为任务s_i的截止期; $et(i)$为任务s_i对资源Res的需求, 有专用和共享2种使用方式, $et(i)=\{et(i)_1, \cdots, et(i)_k\}$, $k \in \{1,2,\cdots,K\}$, K为资源数.

2) 任务是不可抢占的, 且是相互独立的(任务间无先后顺序约束关系).

3) 任务不具有并行性, 即任务是不可再分的.

2.4 调 度 器

假定多处理器系统采用集中式调度机制, 由调度器将所有任务分配到各处理器执行; 每个处理器都有各自的调度队列, 处理器从该调度队列中选择任务进行处理; 调度器与各处理器之间的通信通过调度队列实现[9].

2.5 性能指标

对于多处理器系统, 可使用调度成功率、平均响应时间、处理器利用率、调度长度作为评价调度算法的性能指标.

1) 调度成功率(SR): 被算法调度成功的任务集数N'与被调度的任务集数N之比, 即

$$\mathrm{SR}=N'/N. \tag{2}$$

2) 平均响应时间(ART): 任务集中所有任务开始处理的时间与到达时间的平均差. 任务s_i的到达时间为$at(i)$, 开始处理的时间为$st(i)$, 则其响应时间为$rt(i)=st(i)-at(i)$. 设任务集中的任务数为n, 则任务的平均响应时间为

$$\mathrm{ART}=\sum_{i=1}^{n}\frac{rt(i)}{n}. \tag{3}$$

一个任务集的平均响应时间越小, 即等待时间越短, 调度算法的性能越好.

3) 调度长度(SL): 最早的任务到达时间与最后一个任务的完成时间之差. 假定多处理器系统是非抢占式的, 且任务s_i的处理时间$pt(i)$已知, 完成时间为$ft(i)=st(i)+pt(i)$, 则调度长度为

$$\mathrm{SL}=\max\{ft(i)\}-\min\{at(i)\}. \tag{4}$$

在调度成功的前提下, 较小的调度长度意味着任务被较早处理完毕, 且处理器的平均利用率较高.

4) 处理器利用率(UR): 处理器执行任务占用的时间和与调度长度之比. 设任务集S中有S_j个任务被处理器$P_j\ (j=1,2,\cdots,m)$响应, 则P_j的利用率为

$$\mathrm{UR}_j=\frac{\sum_{i=1}^{S_j} pt(i)}{\mathrm{SL}}, \tag{5}$$

而所有处理器的平均利用率(AUR)则可表示为

$$\mathrm{AUR}=\frac{\sum_{j=1}^{m}\mathrm{UR}_j}{\mathrm{SL}}. \tag{6}$$

一般情况下, 一个调度算法不能使上述指标同时达到最优. 对于实时性要求较高的系统, 调度成功率最重要, 应在保证调度成功率的前提下, 尽可能提高其他指标, 以使系统的整体性能得到优化.

3 任务集的生成

调度的目的是利用调度算法提高设备利用率, 降低任务响应时间和调度长度等. 但是, 一个系统的处理能力是一定的, 调度算法对系统性能的优化也是有限的: 当任务数量远低于系统的处理能力时, 调度算法可以轻易获得较好的性能; 当任务数量过多以至于远大于系统的处理能力时, 调度算法的效果会降低, 甚至出现调度不成功的情况. 因此, 需要描述系统处理能力与任务处理需求之间的关系及其对可调度性的影响, 为设计可行的调度算法提供依据.

3.1 任务集的可调度性

任务s_i的处理时间为$pt(i)$, m个处理器的总处理能力为TPC, 仿真时间为T, 系统的负载系数(LD_f)记作

$$\mathrm{LD}_f=\frac{\sum_{i=1}^{n} pt(i)}{\mathrm{TPC}\times T'}, \tag{7}$$

其中 $T'=\max\{at(i)+pt(i)\}-\min\{at(i)\}$. LD_f 反映了系统在仿真时间 T 内所有任务的总处理需求与系统总处理能力之比. 显然, 任务集可调度的必要条件是

$$\mathrm{LD}_f \leqslant 1. \tag{8}$$

本文研究处理能力相同的同构多处理器系统动态调度, 可认为 $C_1=C_2=\cdots=C_m=1$, 则系统的总处理能力为 $\mathrm{TPC}=m$. 因此, 任务集可调度的必要条件为

$$\frac{\sum_{i=1}^{n} pt(i)}{T'} \leqslant m. \tag{9}$$

3.2 任务集生成方法

任务集生成是指通过一系列的随机函数为 2.3 节所述的任务集中的每个任务 $s_i\ (i=1,2,\cdots,n)$ 产生相应的参数:

1) 到达时间 $at(i)$ 的生成. 设仿真时间为 T, 使用随机函数 $\mathrm{Random}[a,b]$ 产生任务 s_i 的到达时间, 有

$$at(i)=\mathrm{Random}\,[0,T]. \tag{10}$$

2) 处理时间 $pt(i)$ 的生成. 一般情况下, 任务的处理时间会在某个时间区内变化, 将其记为 $\min\_c$, $\max\_c$, 分别表示任务的最小和最大处理时间, 则

$$pt(i)=\mathrm{Random}\,[\min\_c,\max\_c]. \tag{11}$$

3) 截止期 $dt(i)$ 的生成. 任务 s_i 的最早完成时间为 $ft(i)=\max\{at(i)+pt(i)\}$, 而其截止期 $dt(i)\geqslant ft(i)$, 所以可在 $ft(i)$ 的基础上增加一个随机数得到任务的截止期. 设 $R\in[0,1)$ 为任务的可延迟度, 用来描述截止期的紧迫程度, R 越小, 任务越紧迫. 截止期生成如下:

$$dt(i)=\mathrm{Random}\,[ft(i),(1+R)\times ft(i)]. \tag{12}$$

4) 资源需求 $et(i)$ 的生成.

$$et(i)_k=\begin{cases}0,\ s_i \text{ 不访问资源 } r_j;\\ 1,\ s_i \text{ 以专用方式访问资源 } r_j;\\ 2,\ s_i \text{ 以共享方式访问资源 } r_j.\end{cases} \tag{13}$$

其中: $k\in\{1,2,\cdots,K\}$, K 为资源数, 利用随机函数生成 0, 1, 2, 为每个任务的资源需求赋值.

5) 将以上 n 个任务根据到达时间 $at(i)$ 按升序排列, 形成任务集 S.

6) 计算负载系数 LD_f, 若 $\mathrm{LD}_f\in(0.5,1)$, 则保留该任务集; 否则, 返回 1), 重新生成任务集.

4 负载均衡节约算法 (LBTA)

4.1 节约算法存在的问题分析

近视算法可能延迟后续任务的开始运行时间, 因而造成这些任务调度失败. 乔颖等人[5]基于近视算法提出了节约算法 (TA), 在满足任务截止期的处理器中, 选择一个最迟可运行的处理器, 从而增大了后续任务的可调度性. 但当任务出现的密度大, 运行时间很短而可延迟时间很长时, TA 会引起负载失衡, 使某个处理器频繁接受任务而其他处理器处于空闲状态. 以下的例子可以说明这种现象.

用 3.2 节方法生成任务集: 总仿真时间为 900 s, 任务数为 60, 任务的最大运行时间和最小运行时间分别设定为 60 s 和 30 s. 系统的处理器数为 4. 可计算系统的负载系数为 $\mathrm{LD}_f=0.753\,3<1$, 任务集满足可调度的必要条件. 设可行性检查窗口的大小为 5, 最大回溯数为 3, 使用节约算法得到 60 个任务在 4 个处理器上的分配和运行情况如图 1 所示.

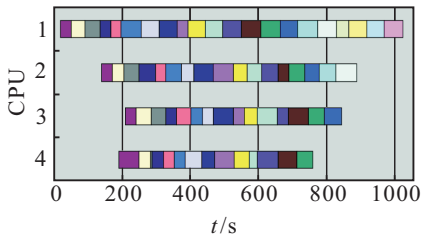

图 1 任务在各处理器上的运行情况

由图 1 可见, 处理器 1 满足任务的时间约束, 总是选择处理器 1 执行任务, 以使其他处理器最早可用时间提前, 从而使后续任务得以更早开始运行, 增加任务的可调度性. 但是, 这样会造成处理器 1 的负载过重, 其他处理器比较空闲. 实际上, 处理器的利用率不均衡会引起调度长度增加, 甚至调度不成功. 因此, 有必要对节约算法进行改进, 以使系统的性能指标进一步优化.

4.2 负载均衡的节约算法 LBTA

定义 1 对一个任务集的子集进行的调度称为局部调度. 若该子集扩展其补集内任一任务后仍为可行调度时, 则称该局部调度是强可行调度[9]. 当对任务集进行局部调度时, 采用固定大小的子集进行, 该子集称为调度窗口, 用 N 表示其大小, 即子集中的任务数.

定义 2 $\mathrm{RET}(k,s)$ 是资源 r_k 在共享方式下的最早可用时间, $\mathrm{RET}(k,e)$ 是资源 r_k 在专用方式下的最早可用时间.

定义 3 $\mathrm{EST}(i)$ 是 s_i 的最早可运行时间, 有

$$\mathrm{EST}(i)=\max\{at(i),\min(\mathrm{avtime}(p_j)),\max(\mathrm{RET}(k,u))\}. \tag{14}$$

其中: $\mathrm{avtime}(p_j)$ 是处理器 p 的最早可用时间; $\min(\mathrm{avtime}(p_j))$ 是系统中处理器的最早可用时间; $\max(\mathrm{RET}(k,u))$ 是任务 s_i 所需资源 k 的最早可用时

间. 当资源为共享访问方式时, $u=s$; 当资源为专用访问方式时, $u=e$.

定义 4　$\mathrm{gapt}\,(s_i,p)=dt(i)-\mathrm{avtime}\,(p_j)$ 表示任务 s_i 的截止期与处理器 p_j 的最早可用时间之差; 用 $\mathrm{avl}\,(s_ip_j)$ 表示任务 s_i 在处理器 p_j 上运行能否满足其截止期, 则

$$\mathrm{avl}\,(s_i,p_j)=\begin{cases}1,\ \mathrm{gapt}\,(s_i,p_j)\geqslant pt(i);\\ 0,\ \mathrm{gapt}\,(s_i,p_j)<pt(i).\end{cases}\tag{15}$$

定义 5　任务调度的目标函数为

$$H(s_i)=dt(i)+W_1\times\mathrm{EST}(i)+W_2\times pt(i),\tag{16}$$

其中 W_1 和 W_2 为权值, 分别表征任务的理想最早可用时间和运行时间对目标函数的影响程度.

设系统中每个资源对应一个入口, 采用与 TA[5] 相同的资源记录方式, 分别记录该资源的被访问任务数及访问模式 (专用或共享), 并用资源列表 ResList[ ] 记录所有资源的信息. 利用式 (15) 描述的目标函数, 可使处理器被选择的机会趋于均等, 从而得到负载均衡的节约算法 LBTA, 描述如下:

Step 1: 将每个处理器的任务队列按截止期非递减顺序排列. 开始时, 局部调度为空.

Step 2: 对调度窗口中的 N 个 (或少于 N 个) 任务进行调度可行性检查, 确定当前的局部调度是否为强可行的. 强可行的条件为: 对于任意 $s_i\in N$, 存在处理器 p_j 使得 $\mathrm{avl}\,(s_i,p_j)=1$, 若是, 则 feasible = ture; 否则, feasible = false.

Step 3: 若 feasible = ture, 则检查调度窗口中任务的目标函数 $H(s_i)=dt(i)+W_1\times\mathrm{EST}(i)+W_2\times ct(i)$, 选择 $H(s_i)$ 值最小的任务 s_i 扩充当前调度; 否则, 返回上一级, 并在此层的调度窗口中选择目标函数值次优的任务 s_j 扩展当前调度, 选择相应的处理器. 当有多于 1 个处理器可供选择时, 则随机选定其中 1 个.

Step 4: 将调度窗口向后移一个任务.

Step 5: 重复 Step 2 ~ Step 4 的操作, 直到满足以下条件中的任一个:

1) 得到一个完全的可行调度;

2) 已经达到最大的返回次数或 $H(s)$ 函数的最大估算值;

3) 达到最大回溯数;

4) 调度完成.

5　算法仿真与性能分析

利用 2.5 节给出的性能指标进行仿真, 以对 TA 和 LBTA 进行比较和分析.

5.1　调度成功率

首先比较 LBTA 算法与 TA 算法的调度成功率. 仿真使用的多核系统及其任务集描述如下:

1) 处理器数量为 4, 且每个处理器的处理能力相同; 系统资源数量为 2, 且每个资源只有一个实例.

2) 用 3.2 节方法生成任务集. 仿真时间为 900 s, 任务集中的任务数量为 60 ~ 80, 任务处理时间 $pt\in[30,60]$, 任务可延迟度 R 为 0.6.

3) 生成的任务集数量为 80, 每个任务集的负载系数需满足 $\mathrm{LD}_f\in[0.5,1)$.

使用上述 80 个任务集对 TA 和 LBTA 进行仿真, 图 2 所示为 2 种算法的调度成功率随调度窗口 K 变化的情况. 由图 2 可以看出, 随着调度窗口 K 的增大, 2 种算法的调度成功率都随之增加, 这是因为调度窗口增大会使算法在进行启发式搜索时的预见性增强. 在调度窗口相同时, LBTA 的调度成功率略高于 TA, 说明 LBTA 在保证调度成功率方面有更好的表现.

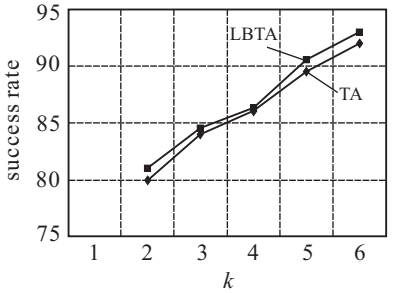

图 2　调度窗口对调度成功率的影响

5.2　其他性能指标

在上述 80 个任务集中选取一个包含 80 个任务的任务集 S, 系统的负载率为 $\mathrm{LD}_f=0.862<1$. 评价的性能指标包括任务的平均响应时间 ART, 调度长度 SL, 每个处理器的利用率 UR 及平均利用率 AUR, 仿真结果如表 1 所示.

表 1　2 种算法对系统性能的影响

算法	ART/s	SL/s	UR_1/%	UR_2/%	UR_3/%	UR_4/%	AUR/%
TA	30.54	1 134	96.20	82.54	65.34	51.30	73.90
LBTA	23.46	996	88.37	83.79	89.80	80.33	85.57

由表 1 可以看出:

1) LBTA 的任务平均响应时间为 23.46 s, 低于 TA 的任务平均响应时间 30.54 s.

2) LBTA 得到的调度长度为 996 s, 低于 TA 的调度长度 1 134 s.

3) LBTA 得到的处理器平均利用率为 85.57%, 且 4 个处理器的利用率比较接近, 均在 80% ~ 90% 之间; 而 TA 的处理器平均利用率 < 74%, 且 4 个处理器的利用率相差比较大, 最低的仅为 51.3%, 最高的则达 96.2%. 这是因为 LBTA 在为任务选择处理器时考虑了处理器当前的利用情况, 保证了负载均衡, 使得

系统的处理器有更高的平均利用率. 另一方面, 较高且均衡的处理器利用率意味着较短的调度长度, 使更多的任务较早得到处理, 因而平均响应时间得以降低. 2 种调度算法的仿真结果也证明了这一点.

由以上仿真结果可以看出, LBTA整体上比TA具有更好的性能指标.

6 结 论

随着多处理器系统逐渐获得更为广泛的应用, 其动态调度问题的研究也逐渐成为热点. 本文将任务集与处理器处理能力之间的匹配关系作为研究调度算法的重要因素, 描述了任务及处理器的一般模型, 分析了任务数量与系统处理能力之间的关系, 提出了可调度任务集的生成方法; 针对多处理器系统的动态调度, 对节约算法TA进行改进, 提出了负载均衡的节约算法LBTA. 仿真结果表明, 在调度成功率、平均响应时间、处理器利用率及调度长度等多个性能指标方面, LBTA均优于TA.

参考文献(References)

[1] Hammond L, Nayfeh B A, Olukotun K. A single-chip multiprocessor[J]. IEEE Computer, 1997, 30(9): 79-85.

[2] Mok A K. Fundamental design problems of distributed systems for the hard real-time environment[D]. Cambridge: Department of Electronic Engineering and Computer Sciences, MIT, 1983.

[3] Ramamritham K J, Stankovic A, Shiah P F. Efficient scheduling algorithms for real-time multiprocessor systems[J]. IEEE Trans on Parallel and Distributed Systems, 1990, 1(2): 184-194.

[4] Manimaran G, Murthy C S R. An efficient dynamic scheduling algorithm for multiprocessor real-time systems[J]. IEEE Trans on Parallel and Distributed Systems, 1998, 9(3): 312-319.

[5] 乔颖, 王宏安, 戴国忠. 一种新的实时多处理器系统的动态调度算法[J]. 软件学报, 2002, 13(1): 51-58.
(Qiao Y, Wang H A, Dai G Z. Developing a new dynamic scheduling algorithm for real-time multiprocessor systems[J]. J of Software, 2002, 13(1): 51-58.)

[6] 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): 976-983.
(Li J G , Chen S Q, Lu Z H. A Dynamic scheduling algorithm based on group optimization in real-time heterogeneous systems[J]. Chinese J of Computers, 2006, 29(6): 976-983.)

[7] 乔颖. 实时异构系统的集成动态调度算法[D]. 北京: 中国科学院软件研究所, 2001.
(Qiao Y. Study of integrated dynamic scheduling for real-time heterogeneous systems[D]. Beijing: Institute of Software, Chinese Academy of Sciences, 2001.)

[8] 吴佳俊. 多核多线程处理器上任务调度技术研究[D]. 北京: 中国科学院计算技术研究所, 2006.
(Wu J J. Task scheduling for multi-core and multi-thread processor[D]. Beijing: Institute of Computing Technology, Chinese Academy of Sciences, 2006.)

[9] 胡天一, 徐中伟. 多处理器并行EDPF优化实时调度算法[J]. 计算机工程与应用, 2007, 43(19): 35-38.
(Hu T Y, Xu Z W. Parallelized EDPF real-time optimization scheduling algorithm for multiprocessor[J]. Computer Engineering and Applications, 2007, 43(19): 35-38.)

(上接第1739页)

[4] 徐南荣, 卞南华. 红外辐射与制导[M]. 北京: 国防工业出版社, 1997: 334-338.
(Xu N R, Bian N H. Infrared radiation and guiding[M]. Beijing: National Defense Industry Press, 1997: 334-338.)

[5] Ferdowsi H, Maralani M, Jabehdar P, et al. Design of bearing-only vision-based tracking filters[J]. Optical Engineering, 2004, 23(2): 472-481.

[6] Julier S J, Uhlmann J K. A general method for approximating nonlinear transformations of probability distributions[EB/OL]. [1997-09-27]. http://www.robots.ox.ac.uk/-siju/work/publications/Unscented.zip.

[7] Julier S J, Uhlmann J K. A consistent, debiased method for converting between polar and Cartesian coordinate systems[C]. Proc of Aero Sense: The 11th Int Symposium on Aerospace/Defense Sensing Simulation and Controls. Orlando, 1997: 110-121.

[8] 潘泉, 杨峰, 叶亮, 等. 一类非线性滤波器——UKF综述[J]. 控制与决策, 2005, 20(5): 481-484.
(Pan Q, Yang F, Ye L, et al. Survey of a kind of nonlinear filters — UKF[J]. Control and Decision, 2005, 20(5): 481-489.)