·体系结构与软件技术· 文章编号：1000-3428(2021)07-0212-06 文献标志码：A 中图分类号：TP391

基于多核处理器的关联任务并行感知调度算法

梁秋玲[1]，张向利[2]，张红梅[2]，闫 坤[1]

（1.桂林电子科技大学 信息与通信学院，广西 桂林 541004；
2.桂林电子科技大学 信息与通信学院 认知无线电与信息处理省部共建教育部重点实验室，广西 桂林 541004）

摘 要：关联任务在多核处理器上并行调度所产生的通信时延，会对任务调度长度和处理器利用率造成负面影响，为了改善多核系统对关联任务的处理性能，针对关联任务在多核处理器上的调度特点，提出一种并行感知调度算法。计算各任务与终点间的最长路径值，按照该值的降序来分配任务调度次序，在分配处理器内核时兼顾关联度和任务最早可执行时间，设置最佳匹配评价函数。实验结果表明，与busHEFT和DTSV算法相比，该算法具有更短的任务调度时延、更少的通信量以及更高的处理器利用率。

关键词：多核系统；总线；关联任务；通信时延；任务调度

开放科学（资源服务）标志码（OSID）：

中文引用格式：梁秋玲，张向利，张红梅，等.基于多核处理器的关联任务并行感知调度算法[J].计算机工程，2021，47(7)：212-217.

英文引用格式：LIANG Q L，ZHANG X L，ZHANG H M，et al.Parallel perceptual scheduling algorithm for related tasks based on multi-core processors[J].Computer Engineering，2021，47(7)：212-217.

Parallel Perceptual Scheduling Algorithm for Related Tasks Based on Multi-Core Processors

LIANG Qiuling[1]，ZHANG Xiangli[2]，ZHANG Hongmei[2]，YAN Kun[1]

（1.School of Information and Communication，Guilin University of Electronic Technology，Guilin，Guangxi 541004，China；
2.Key Laboratory of Cognitive Radio and Information Processing，Ministry of Education，School of Information and Communication，Guilin University of Electronic Technology，Guilin，Guangxi 541004，China）

【Abstract】The communication delay caused by the parallel scheduling of related tasks on multi-core processors has a negative impact on the task scheduling length and processor utilization.In order to improve the performance of multi-core systems in processing related tasks，this paper proposes a parallel perceptual scheduling algorithm based on the scheduling characteristics of related tasks on multi-core processors.The algorithm calculates the value of the longest path from each task to the end point，and assigns the tasks in descending order of the calculated value.The relation and the earliest execution time are both considered when assigning processor cores，and the best matching evaluation function is created.Experimental results show that this algorithm has fewer task scheduling delay，less communication traffic and higher processor utilization compared with busHEFT，DTSV and other algorithms.

【Key words】multi-core system；bus；related task；communication delay；task scheduling

DOI：10. 19678/j. issn. 1000-3428. 0057225

0 概述

多核处理器的应用为实时系统高性能集成设计提供了有效途径[1]。多核处理器通常分为同构多核处理器和异构多核处理器。其中，异构多核处理器更有利于灵活配置系统资源，提高系统性能，降低系统功耗[2]。核间通信一直是多核系统设计过程中的重点问题，而基于共享总线的通信机制具有结构简单、易于实现的优点，能够有效实现核间的互联和通信[3]。但是，共享总线的通信方式效率较低，可扩展性较差，核间通信开

基金项目：广西云计算与大数据协同创新中心开放项目“大数据分布式存储与处理系统设计与实现”（YD1904）；广西研究生教育创新计划（2017YJCX22）。

作者简介：梁秋玲（1994—），女，硕士，主研方向为分布式计算；张向利（通信作者）、张红梅，教授、博士；闫 坤，副教授、博士。

收稿日期：2020-01-15 **修回日期：**2020-02-20 **E-mail：**1319786165@qq.com

销争用问题直接影响多核结构的性能[4-5]。因此，在多核系统应用中降低通信开销尤为重要。

多样化的数据流应用组成了结构各异的任务模型。在多核处理器系统研究中，常见的任务模型有fork/join模型[6]、同步并行模型[7]和DAG（Directed Acyclic Graph）模型[8]。随着智能设备的快速发展，关联任务随之产生，例如，一辆自动驾驶汽车具有并发的多种任务（速度监测、制动控制等），一个任务需要使用另一个任务的输出参数，从而导致任务间存在数据关联[9]。关联任务之间具有相互约束的特点，基于此，研究人员通常采用DAG模型对关联任务进行建模。与传统DAG任务不同，关联任务在多核处理器中执行时需要考虑通信时延。据统计，任务在多核系统中执行时，核间通信造成的通信时延是核内通信的3倍~5倍[10-11]。因此，如何设计任务的映射以充分利用多核处理器强大的计算能力，是关联任务调度算法设计过程中的关键问题。

近年来，有较多学者对关联任务调度算法进行了研究。文献[12]分析最坏情况下关联任务的关联数据流在总线上的执行时间以及系统调度方法，其优化了总线的调度策略。文献[13]通过模型检查器SPIN优化了静态任务和总线访问时间表，并提出一种有效的启发式算法。文献[14]提出一种基于动态规划的伪多项式时间算法求解任务的映射和执行顺序，通过提高任务的并行性来缩短任务调度时间，但其忽略了核间通信开销。文献[15]针对任务的周期特性，通过本地缓存区保存上一周期的关联数据使通信型任务和计算型任务能够重叠调度，从而最大程度地减少通信时延，同时，提出一种整数线性规划（ILP）方法对任务进行调度，但该方法比较复杂且仅适用于周期任务。文献[16]建立一种基于简单树（Simple-tree）的周期任务调度模型，其通过可延迟时间越短、任务越优先的调度方法改善资源利用率并提高任务的死限丢失率。文献[17]提出一种针对周期与非周期关联性任务的调度算法（DTSV），并根据任务相关性给出任务关联性评价函数，将强关联性任务分配到同一内核中，有效减少了核间通信，此外，该文结合轮询算法改善了负载均衡问题。

目前，针对基于总线的多核处理器上关联任务调度算法的研究取得了一定成果，但也存在一些问题，如算法多数是在同构多核处理器背景下展开分析，且所研究的任务结构简单，算法可扩展性较低。为此，本文提出一种异构多核处理器下的结构多样化关联任务调度算法。以计算所得任务的最长路径值为参照，合理分配任务调度的优先级。在处理器内核选择阶段，兼顾内核的可最早完成时间和任务分配到该内核的可减少通信延迟，设计任务与内核的最佳匹配评价函数。在任务调度期间，在内核和总线的空闲间隙插入合适的任务，以充分利用系统资源。

1　关联任务模型

关联任务模型结构如图1所示，研究人员通常将关联任务分为计算型任务和通信型任务两类，用符号<T,E>表示，T代表计算型任务集合，E代表通信型任务集合。图1中的节点即为一个计算型任务，记为T_i，表示第i个计算型任务，对应权值t_i表示任务计算时间。与当前节点直接相连的前一个节点称作前驱计算型任务，记为T_k，对应的连边称作前驱通信型任务，记为E_i^k，权值e_i^k表示通信时间；与当前节点直接相连的后一个节点称作后继计算型任务，记为T_j。为方便下文表述，将第一个计算型任务定义为源点任务，最后一个计算型任务定义为终点任务。每个计算型任务需要分配到处理器内核中执行，通信型任务则需在总线上执行。当具有数据关联性的计算型任务被分配在不同的内核时，任务间通信为核间通信，e_i^k不为零；计算型任务被分配到同一个内核时，任务间通信为核内通信，e_i^k很小，本文假定为0。

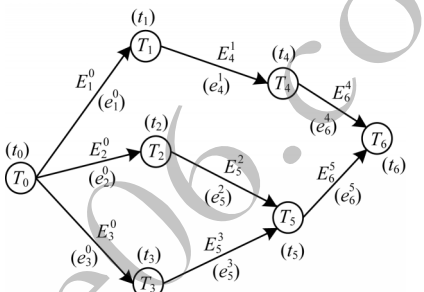

图1　关联任务模型结构

Fig.1　Related task model structure

2　算法设计及相关公式

异构多核处理器各个内核的功能、运算能力不同，导致同一个计算型任务分配到不同内核中执行所需的计算时间也不同。本文在分配计算型任务的调度优先级前，先计算任务在各个内核中执行所需的时间平均值，计算公式如下：

$$\overline{t}_i = T_{\text{avg}}(t(T_i, P_x)),\ x \in (1, 2, \cdots, M) \tag{1}$$

其中，$t(T_i, P_x)$表示T_i在第x个内核处理器中执行所需的时间，M为多核处理器系统的内核个数。

在异构多核系统中，只有芯片上集成的多核的功能配置与任务负载的并行特征匹配，才能有效提高计算效率[18]。根据任务图的模型结构，具有数据关联的各个任务存在严格的先后执行约束关系。因此，在任务调度优先级分配阶段，本文对HEFT算法进行改进，在计算每个计算型任务到终点任务的关键路径值时，考虑与之相关的通信型任务并将其定义为最长路径，按照最长路径值的降序来排列任务的调度次序。相关定义及公式如下：

定义1（计算型任务T_i的最长路径值$L(T_i)$）　$L(T_i)$表示在DAG表示的关联性任务图中，从计算型任务

T_i的最大前驱通信型任务到终点任务的路径中权值之和的最大值。$L(T_i)$的计算公式如下：

$$L(T_i)=\max(e_i^k)+\overline{t}_i+\max\{L(T_j)\}$$
$$T_j\in M_{\text{succ}}(T_i),T_k\in M_{\text{pred}}(T_i) \tag{2}$$

其中，$M_{\text{pred}}(T_i)$、$M_{\text{succ}}(T_i)$表示计算型任务T_i的前驱计算型任务集合和后继计算型任务集合。

高效的调度算法在进行任务调度时需要考虑应用程序的特点、行为和相关性能，然后将这些任务合理地分配到具有相应特性的不同类型的处理器核上[19]。因此，本文调度算法在内核选择阶段考虑任务在异构内核的最早执行时间和关联度。

将关联任务映射到多个内核中，定义$t_{\text{EST}}(T_i,P_x)$、$t_{\text{EFT}}(T_i,P_x)$分别为计算型任务T_i在内核P_x的最早可执行时间和最早完成时间。其中，源点任务T_{entry}的t_{EST}作为调度长度计算时的初始值，将其初始化为0，如下：

$$t_{\text{EST}}(T_{\text{entry}},P_x)=0 \tag{3}$$

根据初始化值，每个待调度计算型任务的t_{EST}与t_{EFT}，可以通过迭代计算处理器中已分配的计算型任务和总线上相关通信型任务的t_{EFT}获得，相关计算过程如下：

1）计算型任务的所有前驱通信型任务的最晚完成时间计算公式如下：

$$t_{\text{pred\_EFT}}(T_i)=\max(t_{\text{EFT}}(E_i^k,B_{\text{bus}})) \tag{4}$$

其中，$t_{\text{EFT}}(E_i^k,B_{\text{bus}})$表示通信型任务$E_i^k$在总线上的最早完成时间，其计算公式如式(5)所示：

$$t_{\text{EFT}}(E_i^k,B_{\text{bus}})=t_{\text{EST}}(E_i^k,B_{\text{bus}})+e_i^k \tag{5}$$

其中，$t_{\text{EST}}(E_i^k,B_{\text{bus}})$表示任务$E_i^k$在总线$B_{\text{bus}}$上的最早可执行时间，其通过迭代计算已调度前驱计算型任务和总线上通信型任务的t_{EFT}获得，如式(6)所示：

$$t_{\text{EST}}(E_i^k,B_{\text{bus}})=\max\{t_{\text{EFT}}(T_k,P_x),A_{\text{avail}}(B_{\text{bus}})\} \tag{6}$$

其中，$A_{\text{avail}}(B_{\text{bus}})$表示总线上最早可用空闲时隙的开始时间，其计算公式(式(7))需满足式(8)的条件。

$$A_{\text{avail}}(B_{\text{bus}})=\min\{t_{\text{EFT}}(E_m,B_{\text{bus}})\}$$
$$S_{\text{space}}(m,B_{\text{bus}})>e_i^k \tag{7}$$
$$S_{\text{space}}(m,B_{\text{bus}})=t_{\text{EST}}(E_{m+1},B_{\text{bus}})-t_{\text{EFT}}(E_m,B_{\text{bus}}) \tag{8}$$

其中，E_m和E_{m+1}代表总线上任意相邻的2个通信型任务，m为数量标记，$S_{\text{space}}(m,B_{\text{bus}})$为总线上的空闲时隙。

2）内核的最早可执行时间，即满足大于或等于任务在该内核的计算时间的最早空闲时隙，计算公式如式(9)、式(10)所示：

$$A_{\text{avail}}(P_x)=\min\{t_{\text{EFT}}(T_m,P_x)\}$$
$$S_{\text{space}}(m,P_x)\geqslant t(T_i,P_x) \tag{9}$$
$$S_{\text{space}}(m,P_x)=t_{\text{EST}}(T_{m+1},P_x)-t_{\text{EFT}}(T_m,P_x) \tag{10}$$

其中，$A_{\text{avail}}(P_x)$表示处理器P_x上最早可用空闲时隙的开始时间，式(9)需满足式(10)的条件。T_m和T_{m+1}分别代表同一个内核上任意相邻的2个计算型任务，$S_{\text{space}}(m,P_x)$为内核的空闲时隙。

3）计算型任务可最早执行时间，为计算型任务所有前驱通信型任务的最晚完成时间和内核最早可执行时间的最大值，如式(11)所示，计算型任务最早可完成时间如式(12)所示。

$$t_{\text{EST}}(T_i,P_x)=\max\{A_{\text{avail}}(P_x),t_{\text{pred\_EFT}}(T_i)\} \tag{11}$$
$$t_{\text{EFT}}(T_i,P_x)=t_{\text{EST}}(T_i,P_x)+t(T_i,P_x) \tag{12}$$

内核与待调度任务的关联度及其匹配评价函数的相关定义及公式如下：

定义2（任务关联度评价函数$R(T_i,P_x)$） $R(T_i,P_x)$为待分配任务T_i与分配到处理器P_x的任务因存在数据关联性而产生的通信时间总和。$R(T_i,P_x)$的计算公式如下：

$$R(T_i,P_x)=\sum_{k=1}^{n}e_i^k,T_k\in M_{\text{pred}}(T_i) \tag{13}$$

其中，n为处理器P_x中的所有计算型任务数量，n=0时表示处理器还没有分配计算型任务。

定义3（处理器匹配评价函数F） F为待分配任务T_i与处理器P_x的关联度与T_i在P_x上的最早可执行时间差值，F值越大，匹配值越大。F的计算公式如下：

$$F(T_i,P_x)=R(T_i,P_x)-t_{\text{EFT}}(T_i,P_x) \tag{14}$$

处理器匹配评价函数用于优化关联任务以通信时间换取计算时间的并行调度效果，降低多核系统的通信量。例如，若一个待分配任务在一个处理器中的最早完成时间只比另外一个处理器早一点，但另外一个处理器与该任务有更强的关联度，可减少更多的通信任务，本文算法则将任务分配到关联度强的处理器中以降低系统整体能耗。

3 关联任务调度算法实现

3.1 任务调度优先级分配

在关联任务中，计算型任务可以执行的必要条件是其所有前驱通信型任务执行完毕，本文算法通过以下步骤计算任务的调度顺序：

步骤1 计算各个计算型任务的最长路径值，按照最长路径值的降序进行排列，记为集合$T=\{T_1,T_2,\cdots,T_x,\cdots,T_y,\cdots,T_N\}$。

步骤2 按照排列顺序遍历集合T，若出现排列靠后的任务T_y是排列靠前任务T_x的前驱计算型任务，则将T_x取出并插入T_y之前，更新集合T；否则，集合T的元素顺序即为调度顺序。

3.2 处理器选择

在关联任务映射到对应的处理器内核时，为了降低通信量并充分利用多核系统强大的计算能力，本文算法通过内核的最佳匹配函数适当约束关联任

务调度的并行度,具体步骤如下:

步骤1 计算待调度任务在每个内核的最早可完成时间 $t_{EFT}(T_i,P_x)$。

步骤2 计算待调度任务与每个处理器的关联度 $R(T_i,P_x)$。

步骤3 计算待调度任务与处理器的匹配值 $F(T_i,P_x)$。

步骤4 选择 $F(T_i,P_x)$ 最大值对应的处理器。

4 实验结果与分析

4.1 任务集及其参数

文献[20]使用的DAG任务集产生方法所产生的任务集更具随机性、多样性和复杂性,因此,本文采用该方法生成实验任务集。文献[20]方法生成的任务模型的主要控制参数包括任务的深度 D_{depth}、单个分支节点的最大并行数 $M_{maxParBranches}$、并行因子 P_{par}、节点因子 P_{term}($P_{par}+P_{term}=1$)和随机因子 $P_{addProb}$。其中,P_{par} 为从同一源点出发的并行任务个数的概率参数,P_{term} 为不同任务汇集同一节点任务个数的概率参数,$P_{addProb}$ 为任务与任务间随机建立关联的概率参数。本文实验中关联任务产生器的各个参数值设置为:$D_{depth}=4$(4×2=8层),$P_{par}=0.8$,$P_{term}=0.2$,$P_{addProb}=0.2$,$M_{maxParBranches}=6$,任务模型的节点权值和边权值的取值范围均为10~100。

4.2 对比实验设计

本文从调度长度比(Scheduling Length Radio,SLR)[21]、处理器利用率和所产生的通信量3个方面来评估算法性能。HEFT是经典的DAG任务调度算法,在现实多种网络的任务调度中得到广泛应用,本文在对HEFT算法的处理过程中考虑通信任务的影响,使该算法适用于基于总线的多处理器环境下的关联任务调度,本文将其重命名为busHEFT,该算法与本文算法(busCDEFT)、DTSV算法[17]作为实验对比算法。

在任务调度过程中,算法具有良好可扩展性表现为其适用于结构多样化的任务集调度,且不会因为任务集或调度系统的一些参数变化而导致性能急剧下降。为证明本文算法具有良好的可扩展性,设计如下2组实验:第一组实验在不同数量的处理器下比较算法的SLR、处理器利用率和通信量3个指标结果,实验设置处理器个数分别为2、3、4、5、6,CCR(Communication to Computation Ratio)为2;第二组实验在不同CCR下比较算法的上述3个指标结果,其中,CCR最小值为0.5,最大值为6.0,步长为0.5,处理器内核数目为3。为了使实验更具代表性,在每次实验参数变化后都使用重新随机生成的任务集,2组实验使用的任务集数目均为500。

4.3 结果分析

第一组实验结果如图2~图4所示,从中可以看出,在不同数目的处理器条件下,与另外2种对比算法相比,本文算法具有更短的调度时延、更少的通信量和更高的资源利用率,这是因为DTSV算法虽然根据任务关联度评价函数将数据关联性大的任务放在同一个处理器上,在一定程度上减少了通信量,但是结合轮询算法分配其他关联度比较弱的任务以实现负载均衡的方法,所产生的通信量会随着处理器数量的增多而减少,同时任务调度时延也会增大,资源利用率降低,负载均衡的性能在异构处理器系统中较难保证,而本文算法和busHEFT算法在任务调度阶段考虑任务模型结构对调度次序的影响,从而能够合理分配调度优先级。在任务映射阶段,本文算法考虑了待调度任务在每个内核中的最早完成时间,使异构处理器核的运算能力与负载相互匹配,从而有效提高计算效率并缩短调度时延。此外,本文算法还兼顾了任务的关联度对通信时延的影响,算法在这样的前提条件下具有一定的并行感知调度性能,不会随着处理器的增多而任意并行调度,从而有效减少了任务通信量并提高了处理器利用率。

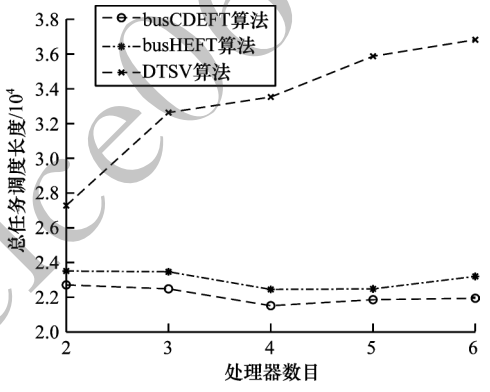

图2 不同处理器数目下3种算法的SLR性能比较

Fig.2 Comparison of SLR performance of three algorithms with different numbers of processors

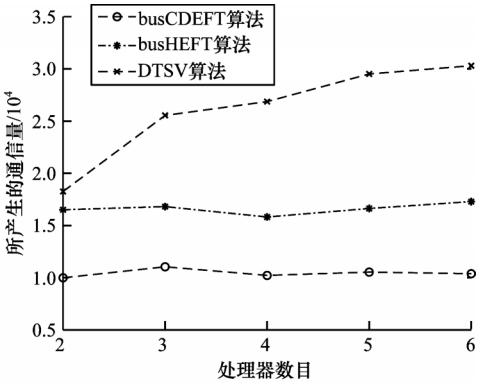

图3 不同处理器数目下3种算法所产生的通信量比较

Fig.3 Comparison of communication traffic of three algorithms with different numbers of processors

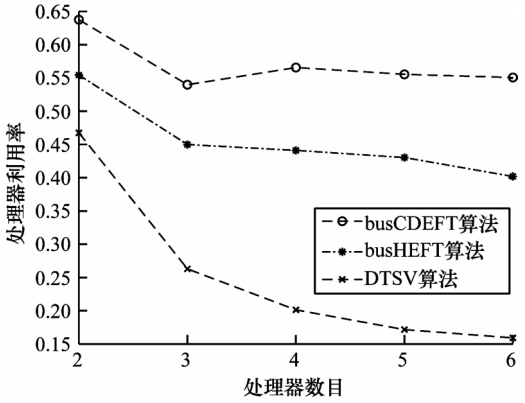

图4 不同处理器数目下3种算法的处理器利用率比较

Fig.4 Comparison of processor utilization of three algorithms with different numbers of processors

第二组实验结果如图5~图7所示,从中可以看出,在其他参数设置相同、任务集CCR值不同的条件下,本文算法同样表现出较好的性能,其任务调度时延更短,通信量更少,处理器利用率更高。因此,本文算法的适用范围较广,可扩展性较高。

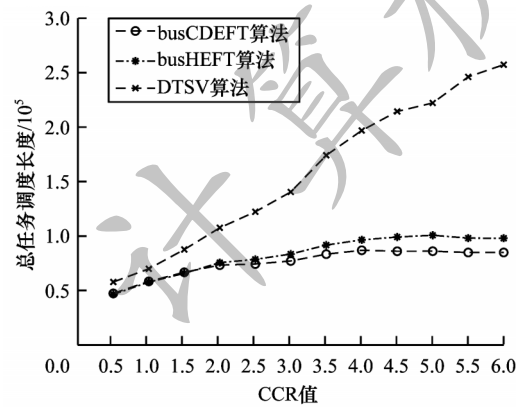

图5 不同CCR下3种算法的SLR性能比较

Fig.5 Comparison of SLR performance of three algorithms under different CCR

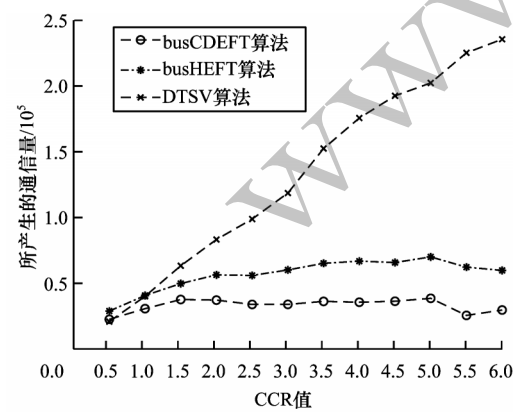

图6 不同CCR下3种算法所产生的通信量比较

Fig.6 Comparison of communication traffic of three algorithms under different CCR

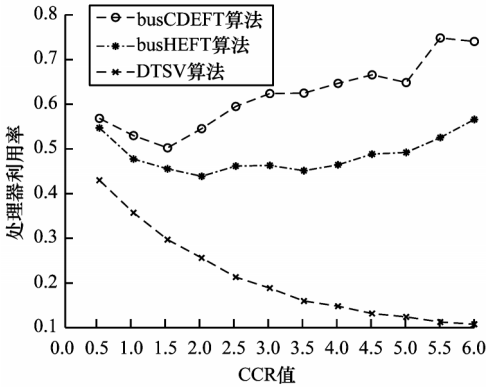

图7 不同CCR下3种算法的处理器利用率比较

Fig.7 Comparison of processor utilization of three algorithms under different CCR

5 结束语

本文提出一种关联任务调度算法,根据关联任务的结构特点计算各个计算型任务的最长路径值,以合理分配任务的调度顺序,同时考虑各个内核的运算性能,设计任务与处理器内核间的最佳匹配函数,从而提高系统的运行效率。实验结果表明,该算法在处理结构复杂的多样化关联任务集时表现出良好的任务调度性能。下一步将研究关联任务具有截止时间约束情况下调度算法的实时性和可靠性问题。

参考文献

[1] 陈刚,关楠,吕鸣松,等. 实时多核嵌入式系统研究综述[J]. 软件学报,2018,29(7):2152-2176.
CHEN G, GUAN N, LÜ M S, et al. State-of-the-art survey of real-time multicore system [J]. Journal of Software, 2018, 29(7): 2152-2176. (in Chinese)

[2] 周楠,胡娟,胡海明. 多核处理器发展趋势及关键技术[J]. 计算机工程与设计,2018,39(2):393-399.
ZHOU N, HU J, HU H M. Development trend and key techniques of multi-core processor[J]. Computer Engineering and Design, 2018, 39(2): 393-399. (in Chinese)

[3] 何军,王飙. 多核处理器的结构设计研究[J]. 计算机工程,2007,33(16):208-210.
HE J, WANG B. Research on architecture design of multi-core processor[J]. Computer Engineering, 2007, 33(16): 208-210. (in Chinese)

[4] 盛肖炜. 多核处理器内部核间通信研究[D]. 沈阳:沈阳理工大学,2013.
SHENG X W. Research on inter core communication in multi-core processors[D]. Shenyang: Shenyang Ligong University, 2013. (in Chinese)

[5] WANG T Y, NIU L W, REN S L, et al. Multi-core fixed-priority scheduling of real-time tasks with statistical deadline guarantee [C]//Proceedings of 2015 Design, Automation & Test in Europe Conference & Exhibition. New York, USA: ACM Press, 2015: 1335-1340.

[6] LAKSHMANAN K，KATO S，RAJKUMAR R. Scheduling parallel real-time tasks on multi-core processors [C]// Proceedings of the 31st IEEE Real-Time Systems Symposium. Washington D. C.，USA：IEEE Press，2010：259-268.

[7] SAIFULLAH A，AGRAWAL K，LU C，et al. Multi-core real-time scheduling for generalized parallel task models [C]//Proceedings of 2011 IEEE Real-Time Systems Symposium. Washington D. C.，USA：IEEE Press，2012：217-226.

[8] SANJOY B，VINCENZO B，ALBERTO M S，et al. A generalized parallel task model for recurrent real-time processes [C]//Proceedings of 2012 IEEE Real-Time Systems Symposium. Washington D. C.，USA：IEEE Press，2012：63-72.

[9] KARAGIANNIS G，ALTINTAS O，EKICI E，et al. Vehicular networking：a survey and tutorial on requirements，architectures，challenges，standards and solutions[J]. IEEE Communications Surveys & Tutorials，2011，13(4)：584-616.

[10] WANG Y，SHAO Z，CHAN H C B，et al. Memory-aware task scheduling with communication overhead minimization for streaming applications on bus-based multiprocessor system-on-chips [J]. IEEE Transactions on Parallel & Distributed Systems，2014，25(7)：1797-1807.

[11] 赵瑞姣，朱怡安，李联. 基于异构多核系统的混合关键任务调度算法[J]. 计算机工程，2018，44(2)：51-55.
ZHAO R J，ZHU Y A，LI L. Mixed-criticality task scheduling algorithm based on heterogeneous multi-core system[J]. Computer Engineering，2018，44(2)：51-55.(in Chinese)

[12] ROSEN J，ANDREI A，ELES P，et al. Bus access optimization for predictable implementation of real-time applications on multiprocessor systems-on-chip [C]// Proceedings of IEEE International Real-Time Systems Symposium. Washington D. C.，USA：IEEE Press，2007：1445-1462.

[13] GU Z H，HE X Q，YUAN M X. Optimization of static task and bus access schedules for time-triggered distributed embedded systems with model-checking[C]//Proceedings of the 44th Annual Design Automation Conference. Washington D. C.，USA：IEEE Press，2007：294-299.

[14] CHEN Y S，SHIH C S，KUO T W. Dynamic task scheduling and processing element allocation for multi-function SoCs[C]//Proceedings of the 13th IEEE Real-Time and Embedded Technology and Applications Symposium. Washington D. C.，USA：IEEE Press，2007：81-90.

[15] WANG Y，LIU D，QIN Z W，et al. Optimally removing intercore communication overhead for streaming applications on MPSoCs computers[J]. IEEE Transactions on Computers，2013，62(2)：336-350.

[16] 黄姝娟，朱怡安，李兵哲，等. 具有依赖关系的周期任务实时调度方法[J]. 计算机学报，2015，38(5)：999-1006.
HUANG S J，ZHU Y，LI B Z，et al. Real-time scheduling method for dependency period tasks[J]. Chinese Journal of Computers，2015，38(5)：999-1006.(in Chinese)

[17] 丁男，聂率航，许力，等. 面向车联网应用的数据关联性任务调度算法[J]. 计算机学报，2017，40(7)：1614-1625.
DING N，NIE S H，XU L，et al. Data related task scheduling for vehicular ad hoc networks[J]. Chinese Journal of Computers，2017，40(7)：1614-1625.(in Chinese)

[18] 任良育，赵成萍，严华. 基于任务复制与冗余消除的多核调度算法[J]. 计算机工程，2019，45(5)：59-65.
REN L Y，ZHAO C P，YAN H. Multi-core scheduling algorithm based on task duplication and redundancy elimination[J]. Computer Engineering，2019，45(5)：59-65.(in Chinese)

[19] HUNG S H，TU C H，YANG W L. A protable efficent intercore communication scheme for embedded multicore platforms[J]. Journal of System Architecture，2011，57(2)：193-205.

[20] MELANI A，BERTOGNA M，BONIFACI V，et al. Response-time analysis of conditional DAG tasks in multiprocessor systems[C]//Proceedings of 2015 Euromicro Conference on Real-Time Systems. Washington D. C.，USA：IEEE Press，2015：211-221.

[21] TOPCUOGLU H，HARIRI S，WU M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE Transactions on Parallel and Distributed Systems，2002，13(3)：260-274.

编辑　吴云芳