DOI: 10.13382/j.jemi.B2002911

一种多核系统任务扰动迭代算法*

张多利[1,2] 廖金月[1,2] 罗 乐[1,2] 倪 伟[1,2] 宋宇鲲[1,2]
(1. 合肥工业大学 微电子设计研究所 合肥 230009; 2. 教育部 IC 设计网上合作研究中心 合肥 230009)

摘 要:任务调度问题是多核处理器相关技术的一个重要组成部分。基于列表的调度算法因其低复杂度和高效率得到广泛关注,但确定任务优先级列表方法的单一性使得算法对解空间搜索不够,易陷入局部最优。为此,提出一种基于任务扰动的迭代型列表调度算法(task perturbation iteration algorithm, TPIA)。该算法通过选取任务扰动因子按照一定扰动策略进行调度列表迭代,对迭代后的列表进行贪心选择,生成更优的调度列表序列以得到更好的调度结果。通过实例和随机有向无环图(DAG)有限集对算法进行验证,结果表明算法能有效改善调度解,调度性能提升平均可达 16.51%,适宜处理大规模、高出入度的复杂 DAG 图;针对 TPIA 算法在低任务总数高通讯开销情况下性能有所下降的问题,对平均任务节点数 130 以下的任务图进行分组测试,获得了对应的 CCR 上界值及其变化趋势。

关键词: 静态任务;调度算法;扰动因子;扰动策略;搜索空间
中图分类号: TN401 **文献标识码**: A **国家标准学科分类代码**: 520.1040

Task perturbation iteration algorithm on multicore systems

Zhang Duoli[1,2] Liao Jinyue[1,2] Luo Le[1,2] Ni Wei[1,2] Song Yukun[1,2]
(1. Institute of VLSI Design, Hefei University of Technology, Hefei 230009, China;
2. IC Design Web-cooperation Research Center of MOE, Hefei 230009, China)

Abstract: Task scheduling is an important part of multi-core processor technology. List-based scheduling algorithms are widely concerned because of their low complexity and high efficiency, but the singleness of the task priority list method makes the algorithm not enough to search the solution space, and it is easy to fall into the local optimal. Therefore, proposes an iterative list scheduling algorithm (TPIA) based on task perturbation. The algorithm selects the task disturbance factor to iterate the scheduling list according to a certain disturbance strategy, greedily selects the iterated list, and generates a better scheduling list sequence to obtain better scheduling results. The thesis validates the algorithm through examples and a limited set of random DAG graphs. The results show that the algorithm can effectively improve the scheduling solution, and the scheduling performance can be improved by an average of 16.51%. It is suitable for processing large-scale and high-entry complex DAG graphs. When the total number of tasks is low and the communication overhead is high, the performance is reduced. The task graphs with an average task node number of less than 130 are grouped and tested to obtain the corresponding upper CCR and its change trend.

Keywords: static task; scheduling; turmoil element; perturbation strategy; search space

0 引 言

多核系统的发展对任务并行执行提出了更高的要求,使得任务调度逐渐成为影响系统性能的关键因素,因此任务调度算法研究变得越发重要。

任务调度的目的是合理分配任务到处理器核心上计算以得到最短的总任务执行时间或其他性能优化[1-5]。由于任务调度问题的 NPC(non-deterministic complete)特性,调度算法无法在多项式时间内给出任务调度的最优

收稿日期: 2020-01-10 Received Date: 2020-01-10
*基金项目:国家自然科学基金(61874156)资助项目

解,为此,众多研究者提出多种调度算法以期能在多项式时间内逼近任务调度最优解,上述研究工作可分为基于列表的调度算法(改进 HEFT[9]、ALS[10]),使用加权算法衡量任务节点的重要程度[6-13];基于任务复制的调度算法(IREA[14]、CDLOS[20]、CPTD[15]),以任务的冗余执行来改善任务间的数据依赖关系;智能搜索算法,通过模拟自然界已有的寻优现象[16-18],从解空间中搜索最优解;以及基于聚簇的调度算法[19-20]。

基于列表的调度算法具有较低的算法复杂度和良好的调度结果,得到广泛应用。改进 HEFT 算法使用 BL(bottom level)和 TL(top level)权值对任务排序,在一定程度上反映了任务间的重要程度,但是依靠 BL/TL 权值能获取的列表极其有限,改进的 HEFT 算法搜索空间狭小,难以获得理想调度效果;对任务间空闲时间段采取插入技术进行处理,但由于适用插入技术的节点有限,所以调度效率提高空间有限。ALS 算法在每一个任务调度完成之后,都会重新计算余下所有任务的优先级,通过对任务优先级的动态更新,ALS 算法能够更好地保证任务的调度优先级和任务重要程度间的一致性,然而启发式原则的优先级计算方法难以完全反映任务的重要程度,ALS 算法仍有陷入局部最优风险。

针对上述算法的缺陷,本文在 SLS 算法的基础上提出任务扰动迭代算法(task perturbation iteration algorithm, TPIA),TPIA 算法以初始调度列表为基础,通过任务扰动策略对调度列表进行不断地寻优迭代,扩大解空间搜索范围;同时,使用插入技术,在任务分配阶段进一步优化列表调度结果。TPIA 具有算法复杂度低,调度结果均较优。

1　相关模型

在任务调度问题中,有向无环图(directed acyclic graph, DAG)常用来对实际问题进行抽象建模,如图 1 所示。

DAG 图用 $G=(V,E,W,C)$ 表示,其中 V 是指图中所有节点的集合,节点代表任务;E 指图中的所有边的集合,边代表任务之间的通信;W 表示所有节点的权重,也就是任务的执行时间;C 表示所有边的权重,边权重表示任务在不同处理核上执行时的通信时间,当任务在同一处理核上执行时,不需要通信,默认通信时间为 0。

在 DAG 图中有 2 类特殊的点,称为源点和汇点,设有 2 个任务节点 $n_i,n_j \in V$, e_{ij} 表示 n_i 到 n_j 的边(通信),那么满足条件 $(n_x \in V \& e_{xi} \in V)$ 的节点 n_x 称为 n_i 的前驱节点,用 $pred(n_i)=\{n_x \in V:e_{xi} \in V\}$ 表示节点 n_i 的前驱节点集合,如 $pred(n_i)$ 为空集,则表示 n_i 为源点;满足条件 $(n_x \in V \& e_{ix} \in V)$ 的节点 n_x 称为 n_i 的后继节点,用 $succ(n_i)=\{n_x \in V:e_{ix} \in V\}$ 表示节点 n_i 的后继节点集合,如 $succ(n_i)$ 为空集,则表示节点 n_i 为汇点。

为反映任务通信时间和执行时间的关系,本文引入 CCR(communication to computation ratio)的概念,即任务通信时间平均值和任务执行时间平均值的比值,CCR 越大,则表示任务图中通信时间占比越高,反之,则表示通信时间占比越小。

DAG 中任务之间的关联程度,一般由入度和出度决定,入度指的是任务节点的前继任务个数,出度指的是任务节点的后继任务个数,出度和入度越高则代表任务之间关联度越高。

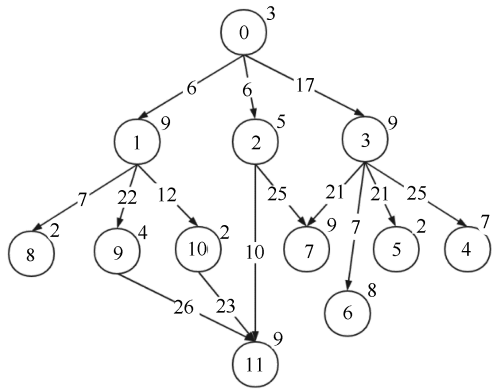

图 1　一个实际程序映射而来的 DAG 图

Fig. 1　A DAG diagram mapped from an actual program

2　任务扰动迭代算法

2.1　初始调度列表生成

TPIA 算法是一种列表调度算法,因此,首先需要计算 DAG 图中的各个任务节点权重,以生成初始调度列表。

对于节点权重的计算,目前常用的权重计算策略有 2 种,即 TL 和 BL。节点的 TL 是指从任意一个源点到该节点的最长路径的长度,但不包括该节点的权重;节点的 BL 是指从该节点到任意一个汇点的最长路径的长度,并且包括该节点的权重。式(1)和(2)定义了 TL 和 BL 的计算方法。

$$\begin{cases} 0, & n_i \text{ 是源点} \\ \max\limits_{n_k \in pred(n_i)} \{tl(n_k)+w(n_i)+c(e_{k,i})\}, & \text{其他} \end{cases} \tag{1}$$

$$\begin{cases} w(n_i), & n_i \text{ 是汇点} \\ \max\limits_{n_k \in suc(n_i)} \{bl(n_k)+w(n_i)+c(e_{k,i})\}, & \text{其他} \end{cases} \tag{2}$$

任务节点权重获得方式会影响任务在调度列表内的次序,导致调度结果差异。因此,选择一个较好的任务权

重计算策略非常重要,文献[11]指出采用 BL 策略得到的初始调度列表,调度性能优于 TL 方法,因此,本文提出的 TPIA 算法首先对 DAG 图中每一个任务节点进行 BL 权值计算,再按照 BL 值进行降序排列,得到初始调度列表。

2.2　扰动因子

基于任务节点 BL 权值排序的初始调度列表并不能完整反映任务的重要程度,所以,本文提出的 TPIA 算法通过在局部范围内随机对调度列表顺序进行重新调整,扩大解空间的搜索范围,寻找更优的调度列表。

对原有列表的局部顺序进行随机调整过程相当于在原列表中引入一个微小的扰动。具体操作如下:首先从调度列表中随机选中一个任务,该任务在列表中的位置将发生移动,且移动方向和步长具有不确定性和随机性,因此称被选中的任务为扰动因子。

由于调度列表中存在边界和任务前后继关系的限制,以下任务禁止选为扰动因子:

$$index(TE)=0\&\text{card}\{n_x \mid n_x\in V, pred(n_x)=\varnothing\}=1 \tag{3}$$

$$index(TE)=card(v)-1\&\text{card}\{n_x \mid n_x\in V, succ(n_x)=\varnothing\}=1 \tag{4}$$

$$SL(index(TE)-1)\in pred(TE)\&SL(index(TE)+1)\in succ(TE) \tag{5}$$

式中:$index(TE)$ 表示扰动因子 TE 在调度列表 SL 中的索引,默认索引从 0 开始;$SL(index(TE)-1)$ 表示调度列表中索引为 $index(TE)-1$ 的元素。

完成扰动因子选择后的操作是移动扰动因子,包括确定移动方向和确定扰动距离 2 个操作。

2.3　扰动方向

调度列表是 1 个一维矢量,故扰动因子只能向 2 个方向移动,因此,扰动方向定义如下:扰动方向只有 2 种可能,即向左或向右。

受扰动因子位置和前后继关系的限制,某些位置的扰动因子只有一个确定的扰动方向。当扰动因子处于调度列表的首位或末位时,由于扰动因子不能向列表外移动,因此扰动因子只能向列表的另一端移动;由于任务具有前驱后继的约束,因此在调度列表中,具有前驱后继的任务其相对顺序不可改变。针对以上特殊情况,对扰动方向添加如下约束:

$$index(TE)=0, O\neq Left \tag{6}$$

$$index(TE)=card(V)-1, O\neq Right \tag{7}$$

$$SL(index(TE)-1)\in pred(TE), O\neq Left \tag{8}$$

$$SL(index(TE)+1)\in succ(TE), O\neq Right \tag{9}$$

2.4　扰动距离

确定扰动方向后,下一步是选择合适的扰动距离进行任务的移动。本文定义扰动距离为任务经过移动后的位置与原位置索引的差值的绝对值,单位是任务,即移动多少个任务代表扰动距离为多少。

任务间存在前后继关系,所以扰动因子在调度列表内的移动要严格遵守与前驱任务和后继任务之间的相对顺序,因此,扰动距离应当满足如下约束:

$$D\leqslant \max_{n_x\in V}(card(succ(n_x))), O=Left \tag{10}$$

$$pred(TE)\cap SL(index(TE)-D, index(TE))\neq(\varnothing)\ \$\ index(TE)-D\geqslant 0, O=Right \tag{11}$$

$$succ(TE)\cap SL(index(TE), index(TE)+D)\neq\varnothing\&index(TE)-D\leqslant card(V), O=Left \tag{12}$$

其中,$SL(index(TE), index(TE)+D)$ 表示索引区间为 $(index(TE), index(TE)+D)$ 的所有调度列表元素。

2.5　调度列表更新

得到扰动因子、扰动方向以及扰动距离后,即可遵循所用方法产生新的调度列表,整个操作过程如图 2 所示,图 2(a)中任务 9 是被选中的扰动因子,扰动方向为右,扰动距离为 2,根据上述条件对调度列表进行更新,得到新的调度列表如图 2(b)所示。

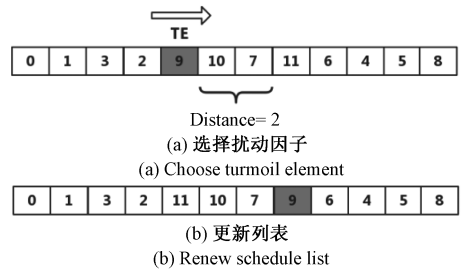

图 2　任务扰动因子移动示意

Fig. 2　Task turmoil element movement

2.6　任务扰动迭代具体实施

综上,本文提出的 TPIA 算法具体步骤如下。

Input:　bl_schedule_list

Output:　best_schedule_list

Initial best_schedule_list = bl_schedule_list

for count in range(1, 30×*TASK_NUM*)

TE = random of schedule_list' s element

O = random of(Right, Left)

D = random of range(1, max_of_DAG_task_fanout)

new_schedule_list = SLU(best_schedule_list, TE, O, D)

If *SL(new_shedule_list) < SL(best_schedule_list)*

Best_schedule_list = new_schedule_list

end for

* *TE(Turmoil Element)*,　*O(orientation)*,

D(distance), *SLU(schedule list updata)*,

SL(Schedule Length)

以图 1 所示任务图为例的 TPIA 算法的具体流程

如下。

1)BL 值计算,得到降序排列的 *bl_schedule_list* = {0, 1, 3, 2, 9, 10, 7, 11, 6, 4, 5, 8},将 *best_schedule_list* 初始化为 *bl_schedule_list*;

2)设定任务扰动迭代的次数为 30 倍任务数,此例即为 360 次;

3)在调度列表中随机选中任务 9 为扰动因子;

4)随机选中右为扰动方向;

5)本例中任务节点最大的扇出为 4,故在 1 到 4 中随机选中 2 为扰动距离;

6)将调度列表按照步骤 3)~5)的要求更新为{0, 1, 3, 2, 10, 7, 9, 11, 6, 4, 5, 8};

7)为新生成的调度列表{0, 1, 3, 2, 10, 7, 9, 11, 6, 4, 5, 8}按照最早执行的贪心策略分配处理器核,得到调度长度为 55,而原调度列表的调度长度为 58,可见新产生的调度列表调度结果小于现有调度列表,进入步骤 8);

8)将迭代产生的调度列表{0, 1, 3, 2, 10, 7, 9, 11, 6, 4, 5, 8}赋值给 *best_schedule_list*;

9)进入下一轮迭代;

10)经过多轮扰动迭代,如图 3 所示,最终得到的调度列表(*best_schedule_list*)为{0, 2, 3, 1, 9, 10, 7, 11, 6, 4, 5, 8},相较于初始调度列表,TPIA 算法优化后的调度列表调度长度为 35,调度效率提升 39.7%。

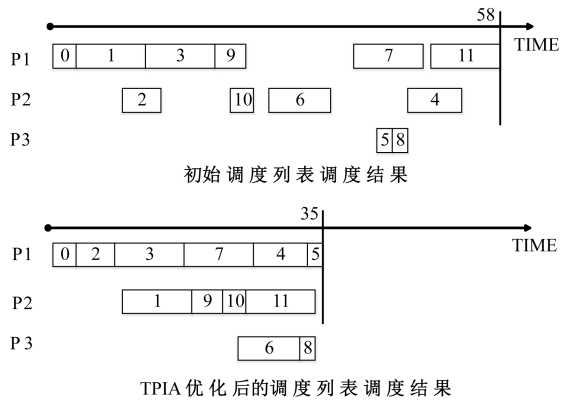

图 3　调度结果对比

Fig. 3　Comparison of scheduling results

2.7　任务分配

在任务分配阶段,使用区间插入技术进一步压缩任务调度时长。每次为当前待执行的任务分配处理器核时,遍寻全部处理器空闲时间片,将任务插入到最早可容纳当前任务的空闲时间片上,任务节点 n_i 的开始时间如下:

$$ST(n_i)=\min_{n\in NP}\begin{pmatrix}\max(valid(p_n)),\\ EFT(n_{pred})+c(e_{pred,i})\end{pmatrix} \tag{13}$$

2.8　时间复杂度分析

SLS 算法的时间复杂度为 $O(V^2)$, TPIA 算法在 SLS 算法的基础上对调度列表进行扰动迭代,总迭代次数 $C = p \times V(p = 30)$, 因此 TPIA 算法的复杂度为 $O(30 \times V \times V^2)$, 化简后为 $O(30V^3)$,因此本算法的复杂度为 $O(V^3)$。

3　实验

分别使用已有算法的实例 DAG 图及随机 DAG 图进行实验,以评测 TPIA 算法的性能。

3.1　已有算法 DAG 图测试

首先对文献[14-15]中对应的 DAG 图使用 TPIA 算法进行调度,得到结果与原算法的调度结果进行比较,结果如表 1 所示,可见 TPIA 算法在调度长度上均小于原文献所提出算法,具有更高调度效率。

表 1　调度结果对比

Table 1　Scheduling result comparison

图例	算法	时间	内核个数
文献[14]	HEFT 改进算法[14]	18	2
图 1	TPIA	17	2
文献[15]	高级列表调度算法[15]	17	4
图 1(a)	TPIA	16	4

3.2　实例任务图

距离多普勒(range-Doppler, RD)算法是雷达成像中最为直观和经典的算法。雷达成像中使用 RD 算法实现方位估计工作的任务图如图 4 所示。对此任务图在并行计算平台上的执行过程使用 SLS 算法和 TPIA 算法分别进行调度,表 2 结果表明经 TPIA 算法调度调整了任务执行的优先级列表,新列表的调度长度为 38 579,低于 SLS 算法调度结果。TPIA 算法可以减少任务执行时间,加快雷达成像进程。

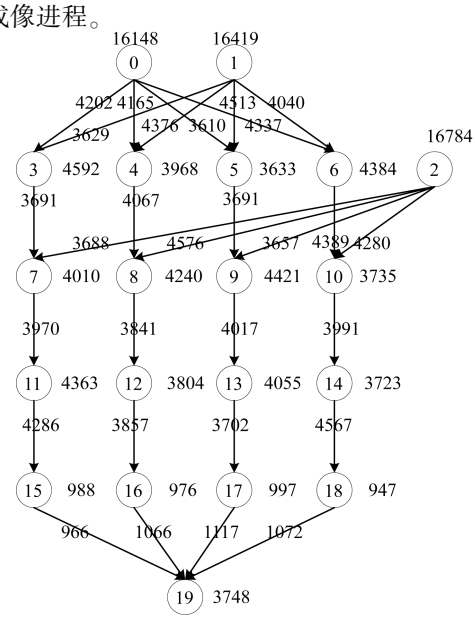

图 4　距离多普勒方位估计任务图

Fig. 4　Task graph of range Doppler bearing estimation

表 2　实例任务图调度结果对比表

Table 2　Example task graph scheduling result comparison

算法	调度列表	时间
SLS	{0,1,2,3,6,4,5,7,9,10,8,11,14,13,12,17,16,18,15,19}	38 715
TPIA	{0,1,2,5,3,6,4,7,9,10,8,14,13,12,11,17,16,18,15,19}	38 597

3.3　随机 DAG 任务图测试

为验证算法在普遍情况下的调度性能，本文使用 TGFF 随机生成 64 组任务图，包含节点数、平均出入度、CCR 三个约束条件。其中，任务节点数取值为 50、100、150、200；平均出度和平均入度相同，取值范围是 2、3、4、5；CCR 取值有 0.1、1、5、10。每组包括 100 个任务图，故本测试集合共包含 6 400 个随机 DAG 任务图。

WPTS 算法[15]是一种包含任务聚簇和加权任务复制的新型混合任务调度算法。本文选择此算法作为比较对象，使用增强比（improve rate，IR）表示 TPIA 算法对比 WPTS 算法的效率提升，对于给定任务图 G：

$$IR = (WPTS(G) - TPIA(G))/WPTS(G) \tag{14}$$

经实验获得 IR 的测试结果如图 5 所示。

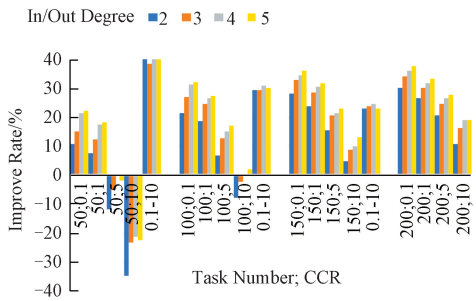

图 5　不同条件下 IR 变化

Fig. 5　IR change under different conditions

从图 5 可以看出，64 组实验中，*IR* 获得正值的次数为 55 次，即在绝大部分情况下，TPIA 算法都能获得较好的调度性能。其中 *IR* 的最大值为 37.90%，最小值为 −35.09%，平均值为 16.51%，不考虑性能降低的情况下，平均性能提升可达 21.02%。

图 5 中随着任务图的某一约束条件改变，TPIA 算法相对 WPTS 算法的增强比 IR 具有较为有规律的变化趋势，归纳总结如下。

1）仅改变任务图出入度时，*IR* 随着出入度的增大而增大，最大可增加 11.71%。这表明 TPIA 算法的性能随着出入度的增加有所提升，算法适合处理关联复杂的任务。

2）仅改变任务总数时，*IR* 随着任务数量的增加而明显增大，*IR* 的增加范围为 14.78%～46.01%。任务节点总数的增加将引起 TPIA 算法性能的显著提升，TPIA 算法更适合处理规模较大的任务图。

3）仅改变 CCR 时，随着 CCR 的增加，*IR* 有所下降，且当任务规模较小时，*IR* 可为负值。表明当通信开销持续增加，TPIA 算法将逐渐失去性能优势。

为确定 TPIA 算法适用范围，本文对不同任务规模及出入度下的任务图进行实验，得到图 6 所示的 CCR 上界，实验数据保留两位有效数字。使用 TPIA 算法处理特定规模和出入度下的任务时，CCR 应不超过实验获得的上界值。

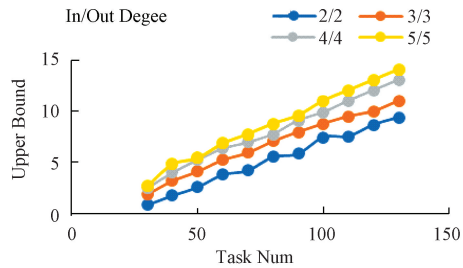

图 6　TPIA 算法在不同任务规模下 CCR 上界

Fig. 6　CCR upper bound of TPIA algorithm under different task scales

4　结　论

本文针对列表调度算法提出一种改进的任务扰动迭代列表算法。TPIA 算法从 SLS 算法产生的初始调度列表出发，通过选择任务扰动因子进行随机扰动，有效扩大了对任务调度解空间的搜索范围，能够更好地产生更优调度列表，得到较短的任务调度长度。

为验证本算法的性能，本文横向对比了 3 种任务调度算法的具体实例，均得到更短的调度长度且内核使用效率高；同时本文使用随机 DAG 任务图有限集对算法进行测试，结果表明本算法对任务关联程度高，在任务规模大、通信代价低的情况下，算法调度性能提升更显著，相对 WPTS 算法性能平均提升可达 16.51%。TPIA 算法实现简单，除了在低任务总数高通信代价的少数不利情形下，均能收获良好的调度性能。

参考文献

[1] 吴小东，韩建军，王天江. 一种基于 VFD 多核系统的硬实时任务节能调度算法[J]. 计算机研究与发展，2012，49(5)：1018-1027.

WU X D，HAN J J，WANG T J. Energy-aware scheduling of hard real-time tasks in vfd-based multi-core systems [J]. Journal of Computer Research and Development，2012，49(5)：1018-1027.

[2] CHETTO M. Optimal scheduling for real-time jobs in energy harvesting computing systems [J]. IEEE Transactions on Emerging Topics in Computing, 2014, 2(2): 122-133.

[3] FARAGARDI H R, SHOJAEE R, YAZDANI N. Reliability-aware task allocation in distributed computing systems using hybrid simulated annealing and tabu search[C]. 14th IEEE International Conference on High Performance Computing and Communication & IEEE 9TH International Conference on Embedded software and Systems, 2012: 1088-1095

[4] 彭浩,韩江洪,陆阳,等. 副版本延迟的多处理器全局实时容错调度算法[J]. 电子测量与仪器学报,2015, 29(9):1302-1309.
PENG H,HAO J H,LU Y, et al. Global real-time fault tolerant scheduling algorithm for backup delay multiprocessor[J]. Journal of Electronic Measurement and Instrumentation,2015,29(9):1302-1309.

[5] TANG Z, QI L, CHENG Z, et al. An energy-efficient task scheduling algorithm in DVFS-enabled cloud environment [J]. Journal of Grid Computing, 2016, 14(1):55-74.

[6] BITTENCOURT L F,SAKELLARIOU R, MADEIRA E R M. DAG scheduling using a look ahead variant of the heterogeneous earliest finish time algorithm [C]. 18th Euromicro Conference on Parallel,2010: 27-34.

[7] TABAKE K, CAMBAZOGLU B B, AYKANAT C. Improving the performance of independent task assignment heuristics minmin, maxmin and sufferage [J]. IEEE Transactions on Parallel and Distributed Systems, 2014, 25(5):1244-1256.

[8] 宋宇鲲,韦龙龙,张多利. 多核系统静态任务调度的启发式算法[J]. 电子测量与仪器学报,2018,32(5): 134-141.
SONG Y K, WEI L L, ZHANG D L. et al. Heuristic algorithm for static tasks scheduling of multicore system[J]. Journal of Electronic Measurement and Instrumentation, 2018,32(5):134-141.

[9] 李东生,李龙,王骁,等. 基于多核系统 NOC 架构的静态列表调度算法[J]. 计算机应用研究,2016,33(4): 1054-1057.
LI D SH, LI L, WANG J, et al. Static list scheduling algorithm based on multi-core system NOC architecture[J]. Application Research of Computers, 2016,33(4):1054-1057.

[10] 穆鹏程,JEAN FRANCOIS N,MICKAEL R,等. 并行嵌入式系统中具有通信竞争任务调度问题的高级列表调度方法[J]. 中国科学(信息科学),2011,41(3): 349-36.
MU P CH, JEAN FRANCOIS N, MICKAEL R, et al. Advanced list scheduling heuristic for task scheduling with communication contention for parallel embedded systems[J]. Science China Information Sciences, 2011, 41 (3): 349-364.

[11] SHEN Y, QI D. Study on static task scheduling based on heterogeneous multi-core processor [C]. International Conference on Computer Network, Electronic and Automation, 2017: 180-182.

[12] 赵茂行,徐德刚,苏志芳,等. 面向异构多核平台的两段式多任务调度算法研究[J]. 控制工程, 2018, 25(12): 2140-2146.
ZHAO M H, XU D G, SU ZH F, et al. Two-stage multi-task scheduling algorithm for heterogeneous multi-core processor platform [J]. Control Engineering of China, 2018, 25(12): 2140-2146.

[13] 周超群,周亦敏. 一种改进的基于复制的异构多核任务调度算法[J]. 电子科技,2017,30(6): 57-62.
ZHOU CH Q,ZHOU Y M. An improved replication based heterogeneous multi-core task scheduling algorithm [J]. Electronic Science & Technology,2017,30 (6): 57-62.

[14] 谢志强,韩英杰,齐永红,等. 基于关键路径和任务复制的多核调度算法[J]. 国防科技大学学报,2014, 36(1):172-177.
XIE ZH Q, HAN Y J, QI Y H, et al. Multi-core scheduling algorithm based on critical path and task replication[J]. Journal of National University of Defense Technology, 2014,36(1):172-177.

[15] 李静梅,张博,王雪. 基于粒子群优化的异构多处理器任务调度算法[J]. 计算机应用研究,2012,29(10): 3621-3624.
LI J M, ZHANG B, WANG X. Heterogeneous multiprocessor task scheduling algorithm based on particle swarm optimization [J]. Application Research of Computers, 2012, 29(10):3621-3624.

[16] LEE J, RAMANATHAN S, PHAN K, et al. MC-fluid: Multi-core fluid-based mixed-criticality scheduling[J]. IEEE Transactions on Computers,2018,67(4):469-483.

[17] LU H, LIU J, NIU R, et al. Fitness distance analysis for parallel genetic algorithm in the test task scheduling problem [C]. Soft Computing, 2014, 18 (12): 2385-2396.

[18] ALI H G, SAROIT I A, KOTB A, et al. Grouped tasks scheduling algorithm based on QoS in cloudcomputing network [J]. Egyptian Informatics Journal, 2017, 18(1): 11-19.

[19] 李静梅,李静,丁楠. 基于异构多核处理器的高效任务

调度算法[J]. 高技术通讯,2012,22(3): 225-230.
LI J M, LI J, DING N. An efficient task scheduling algorithm for heterogeneous multicore processors [J]. High Technology Letters, 2012,22(3): 225-230.

[20] YOOSEFI A, NAJI H R. A clustering algorithm for communication-aware scheduling of task graphs on multi-core reconfigurable systems [J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28 (10): 2718-2732.

作者简介

张多利,1999 年于合肥工业大学获得学士学位,2005 年于合肥工业大学获得博士学位,现为合肥工业大学研究员、硕士生导师,主要研究方向为多核处理器体系结构与设计方法、多媒体专用集成电路设计实现等。

E-mail:zhangduoli@ hfut. edu. cn

Zhang Duoli received B. Sc. degree from Hefei University of Technology in 1999 and Ph. D. degree from Hefei University of Technology in 2005. He is currently a researcher and M. Sc. supervisor at Hefei University of Technology. His main research interests include multi-core processor architecture and design methods, multimedia-specific integration circuit design and realization.

廖金月,2020 年毕业于合肥工业大学,主要研究方向为任务调度。

E-mail:liaojinyue@ foxmail. com

Liao Jinyue graduated from Hefei University of Technology in 2020. Her main research interests include task scheduling.

罗乐,2019 年于合肥工业大学获得硕士学位,主要研究方向为任务调度。

E-mail:connectluole@ 163. com

Luo Le received M. Sc. degree from Hefei University of Technology in 2019. His main research interests include task scheduling.

倪伟,1999 年于合肥工业大学获得学士学位,2004 年于合肥工业大学获得博士学位,现为合肥工业大学副教授、硕士生导师,主要研究方向为数字集成电路设计、可重构计算、嵌入式系统等。

E-mail:ni. wei@ hfut. edu. cn

Ni Wei received B. Sc. degree from Hefei University of Technology in 1999 and Ph. D. degree from Hefei University of Technology in 2004. He is currently an associate professor and a M. Sc. supervisor at Hefei University of Technology. His main research interests include digital integrated circuit design, reconfigurable computing and embedding System.

宋宇鲲(通信作者),2007 年于合肥工业大学获得博士学位,现为合肥工业大学副研究员、硕士生导师,主要研究方向为片上网络优化、多核系统设计、数字信号处理的 VLSI 实现等。

E-mail:songyukun@ hfut. edu. cn

Song Yukun (Corresponding author) received Ph. D. from Hefei University of Technology in 2007. Now he is an associate researcher and M. Sc. supervisor at Hefei University of Technology. His main research interests include on-chip network optimization, multi-core system design and VLSI implementation of digital signal processing.