DOI：10.7652/xjtuxb201506012

片上网络异构多核系统任务调度与映射

杨鹏飞，王泉
（西安电子科技大学计算机学院，710071，西安）

摘要：针对传统任务模型包含有效信息少，任务调度算法效率低、效果差的问题，设计了新的任务模型，提出了一种改进的粒子群算法(optimized particle swarm optimization，oPSO)。新模型增加了对任务类型及任务间迁移成本、计算单元类型及其运行成本等特性的描述。通过分析任务调度问题的需求，制定了 oPSO 算法的编解码方案，设定了算法各个关键部分参数及计算方法，并解决了粒子群算法(PSO)在任务调度前期收敛速度过快、后期易陷入局部最优的问题。在不同任务规模下分别对遗传算法(GA)、PSO 以及 oPSO 算法进行调度仿真对比，当 IP 核数目为 100 左右时，oPSO 算法较 GA 算法和 PSO 算法运行时间至少缩短 10%，系统功耗至少降低 15%，实验结果表明：oPSO 算法调度效果明显优于其他算法，且各节点上功耗更为均衡，适用于解决任务调度问题。

关键词：多核系统；片上网络；任务划分；IP 映射
中图分类号：TN409 **文献标志码**：A **文章编号**：0253-987X(2015)06-0072-05

An Effective Scheduling and Mapping Algorithm of Tasks for Heterogeneous NoC-Based MPSoC

YANG Pengfei, WANG Quan
(School of Computer, Xidian University, Xi'an 710071, China)

Abstract: A new task model is designed and an optimized particle swarm optimization (oPSO) algorithm is proposed to solve the problem that the traditional task DAG model contains less information and the existing task scheduling algorithms based on the model are of inefficiency. The new model adds the description of task type and some real inter-task relations such as transfer cost of the task, the type of processing element (PE) and its running cost. After the requirements of task scheduling and mapping are analyzed, a new scheme of coding and decoding is formulated, key parameters of the algorithm and their calculation are proposed, and the shortcomings of the particle swarm optimization algorithm such as poor local search capacity in the early period and being easily trapped into local optima in the late period of the algorithm are overcome. Simulations and comparisons with the GA and PSO algorithms under different IP scales show that when the number of IPs is about 100, the execution time and the power consumption of the oPSO algorithm reduce at least 10% and 15%, respectively, the scheduling effect of oPSO is much better than those of other algorithms, and the energy consumption on each IPs is balanced. Thus it can be concluded that the proposed algorithm is applicable for the solution of task scheduling.

Keywords: multiprocessor system-on-chip; network on chip; task scheduling; IP mapping

收稿日期：2014-10-21。 作者简介：杨鹏飞(1985—)，男，博士生；王泉(通信作者)，男，教授，博士生导师。 基金项目：国家自然科学基金资助项目(61474087)。
网络出版时间：2015-04-29 网络出版地址：http://www.cnki.net/kcms/detail/61.1069.T.20150429.1437.002.html

利用片上网络连接的系统级芯片(system-on-chip，SOC)成为新一代复杂计算体系结构的必然选择[1]，但是也给系统设计带来很多新的问题与挑战，任务的调度映射问题成为研究重点，决定了任务在体系结构上的实现方式、处理性能和效率。调度问题的通用解决思路是将任务抽象成某种形式的模型来设计调度算法，根据子任务之间的信息，以整个系统运行时间最短为目标对任务进行分配与调度[2-4]，这其中涉及任务模型的设计以及调度算法的选择应用两个关键部分。

任务模型方面，最为普遍的是有向无环图(directed acyclic graph，DAG)模型。DAG 模型将任务间的关系简单抽象为前后调用关系、数据通信量以及其在不同计算单元上的运算时间，而任务间的相互依赖关系复杂，其他影响因素例如任务类型、其对应的计算单元的种类、子任务在不同计算单元上的运行成本等，DAG 模型并没有全面包含，且模型较少地考虑任务在不同计算单元间的迁移成本。

调度算法方面，从进行调度决策的时机来看，可以分为静态调度和动态调度。静态任务调度是由编译器在编译时进行调度决策。例如基于列表的算法(list-based)[3-4]、聚类算法[5-7]和基于副本的算法(duplication based algorithms)[6-7]。静态调度模型有一些缺点，如模型只是基于对处理器间通信和执行时间的近似估计，会产生不好的调度结果。动态任务调度是根据系统中实际任务运行时的情况实时将其调度到相应的计算单元上执行，并满足系统所要求的各项约束指标，例如基于遗传算法[8]和蚁群[9-10]的启发式任务调度、基于任务池库的动态调度算法[11]、粒子群优化算法(particle swarm optimization，PSO)[12]以及基于实时性约束的动态调度算法[13]等。在实际应用中，这些算法本身的问题容易导致运行过程中产生种种弊端。例如遗传算法参数较多、编程实现比较复杂，且易陷入局部最优；蚁群算法前期不能很好地覆盖全部集合；粒子群优化算法较遗传算法更加方便高效，但是其初期收敛速度过快，易陷入局部最优，后期的局部搜索能力较差，收敛速度缓慢。以上种种问题导致应用传统算法进行调度计算都会导致结果与最优值间存在差距。

此外，目前的调度算法多数较少或者没有综合考虑功耗均衡、资源占用等其他因素，没有考虑片上网络拓扑的异构性，且大多忽略任务间的通信延时，算法的实用性较差。在项目组之前的研究中，针对片上网络规模大，片上可调用资源相对任务规模有较大盈余的情况，将调度方案分为任务划分与调度和 IP 映射两个部分完成[14]，但是 IP 映射算法在片上网络规模不大，任务运行需调用全部片上资源的情况下成为多余运算，浪费了系统资源。

本文重新设计了任务模型，使其能真实反映子任务间调度与制约关系。适应性的设计改进粒子群优化算法，使其适用于异构多核任务调度映射问题，并克服了算法本身容易陷入局部最优、后期局部搜索能力差的缺陷。利用其将一个大的任务按照通信量及调用关系分割成数个具有高并行性、粒度大小合适的子任务，结合任务性质分配到相应的计算单元上，在任务执行时间短、占用系统资源少、功耗均衡的前提下，进一步降低整个系统的通信延时。

1　相关模型

算法设计了两个模型，任务调度模型和通信核图。任务调度模型将任务抽象为一个五元组

$$O_{\mathrm{DAG}}=(V,E,S,U,C) \tag{1}$$

式中：V 表示任务节点集，即顶点 $v\in V$ 表示一个子任务；E 表示边集，边 $e_{ij}\in E$ 表示任务 v_i 到 v_j 存在数据通信，方向表示数据传递方向；S 表示任务类型，其与计算单元的类型对应，即任务只能被分配到与之类型匹配的计算单元上执行，可以通过矩阵 $\boldsymbol{D}=\{d_{a,b}\}$ 表示，元素 $d_{a,b}=\infty$ 表示任务 v_a 不适宜在计算单元 p_b 上执行，$d_{a,b}=t$ 表示 v_a 可以在 p_b 上执行，执行时间为 t。元素 $U_r\in U$ 表示第 r 种计算单元的单位时间运行成本。元素 $C_{ij}\in C$ 表示子任务 v_i 通过边 e_{ij} 到 v_j 的迁移成本，当 v_i 到 v_j 分配到同一个计算单元上时，C_{ij} 为 0。

子任务映射到计算单元上，会形成具有通信关系的通信核图，可以抽象为一个三元组

$$C_{\mathrm{DAG}}=(P,R,T) \tag{2}$$

式中：P 表示通信核图中的计算单元集合，其中顶点 $p_\eta(r)\in P$，η 是计算单元的唯一识别编码，r 为类型编码；R 表示边集，边 $r_{xy}\in R$ 表示计算单元 p_x 与 p_y 之间存在数据交换；T 为计算单元间的通信开销，T_{xy} 表示计算单元 p_x 与 p_y 之间的通信总量。

2　调度算法

粒子群算法不能直接用于解决任务调度映射问题，必须进行适应性的优化设计。

2.1　编解码

子任务和计算单元的数目经常是不同的，设一个任务中包含 N 个子任务，系统中有 M 个可利用的计

算单元,这些计算单元可分为 m 种,按子任务数与计算单元数的大小关系,编码分为以下 3 种情况。

(1)计算单元数等于子任务数。粒子编码长度等于子任务数,假设 $N=M=10$,粒子(2,3,5,1,10,8,6,9,7,4)即是一个可行的调度方案,见表 1。

表 1 M 等于 N 时编码示例

子任务号	1	2	3	4	5	6	7	8	9	10
计算单元号	2	3	5	1	10	8	6	9	7	4

(2)计算单元数大于子任务数。粒子编码长度等于计算单元的数目,编码时首先添加 $M-N$ 个虚拟子任务,这些子任务在任意计算单元上的执行时间均为 0,它们与其他 N 个子任务间的通信量为 0。假设 $M=10$、$N=6$,粒子编码示例见表 2。

表 2 M 大于 N 时编码示例

编号类型	实际子任务						虚拟子任务			
子任务号	1	2	3	4	5	6	7	8	9	10
计算单元号	2	3	5	1	10	8	6	9	7	4

(3)计算单元数小于子任务数。首先采用线性聚簇的思想,将通信量大且能在同一个计算单元上执行的子任务合并,直到合并后的子任务数等于计算单元数。对于 O_{DAG},首先将所有子任务中相邻的有通信关系且其对应计算单元类型有交集的子任务按照通信量的大小排序,然后按照通信量,从大到小进行合并,直到子任务数等于计算单元数。假设 $M=6$、$N=10$,粒子编码示例见表 3。

表 3 M 小于 N 时编码示例

合并前子任务号	1,5	6	10	2	3,7,8	4,9
合并后子任务号	1	2	3	4	5	6
计算单元号	3	5	1	4	6	2

通过以上处理,计算单元数等于子任务数,编码方式等同于第一种描述情况。

解码方面,算法结束后,会得到一个最优的计算单元序列,每个计算单元所在的位置序列即代表其需要执行的子任务编号。对于计算单元数目大于子任务数的情况,由于虚拟子任务是不占用运算资源和通信资源的,所以运行虚拟子任务的计算单元实际是不被调用的;对于计算单元数小于子任务数的情况,计算单元的位置序列对应的是合并后的子任务编号,只需将合并后的子任务编号与合并前的编号对应起来,即可实现解码操作。

由前面的问题模型可知,每个子任务在不同计算单元上的执行时间是已知的,每个计算单元上任务运行的时间为

$$T_{\mathrm{sub}}^{x}=\sum_{i=1}^{q}T_{i,x} \tag{3}$$

式中:$T_{i,x}$ 表示子任务 v_i 在第 x 个计算单元上的运行时间;q 表示分配到第 x 个计算单元上的子任务的数目。用 k 表示此次调度算法中可调用的计算单元的总数目,则任务总完成时间为

$$T_{\mathrm{all}}=\max_{x=1}^{k}\{T_{\mathrm{sub}}^{x}\} \tag{4}$$

完成任务的总运行成本为

$$C_{\mathrm{run}}=\sum_{x=1}^{k}T_{\mathrm{sub}}^{x}U_x \tag{5}$$

假设分配到第 x 个计算单元上的任务集合为 V_x,分配到第 y 个计算单元上的任务集合为 V_y,则计算单元 p_x 与 p_y 之间的任务迁移成本为

$$C_{\mathrm{tr}}^{xy}=\sum_{\forall i,j}C_{ij}\quad(v_i\in V_x,v_j\in V_y) \tag{6}$$

整个任务的迁移成本为

$$C_{\mathrm{tr}}=\sum_{\forall x\neq y}C_{\mathrm{tr}}^{xy} \tag{7}$$

拓扑上,计算单元 p_x 与 p_y 的通信成本为

$$C_{\mathrm{com}}=\sum_{\forall T_{xy}\in T}T_{xy}D(L(p_x),L(p_y)) \tag{8}$$

式中:T_{xy} 是 p_x 与 p_y 之间的通信总量;$D(L(p_x),L(p_y))$ 是 p_x 与 p_y 在拓扑上的映射位置之间的曼哈顿距离。

2.2 初始化及适应度函数

设种群规模为 S、子任务数为 N、计算单元的总数为 M、计算单元种类数为 m,则种群的初始化描述为:随机产生 S 个粒子,第 s 个粒子的位置由向量 $\boldsymbol{x}_s=(x_{s1},x_{s2},\cdots,x_{sn})$ 表示,其中 $1\leqslant s\leqslant S$,$1\leqslant n\leqslant N$,$x_{sn}(1\leqslant x_{sn}\leqslant M)$ 表示在第 s 个粒子中任务 s 被分配到第 x_{sn} 个计算单元上运行;粒子速度由向量 $\boldsymbol{v}_s=(v_{s1},v_{s2},\cdots,v_{sn})$ 表示,其中 $1\leqslant s\leqslant S$,$1\leqslant n\leqslant N$,$-M\leqslant v_{sn}\leqslant M$。时间的适应度函数为

$$F_{\mathrm{time}}^{s}=1/T_{\mathrm{all}}^{s}\quad(1\leqslant s\leqslant S) \tag{9}$$

式中:T_{all}^{s} 表示第 s 个粒子的对应映射方案的任务总完成时间。成本的适应度函数为

$$F_{\mathrm{cos}}^{s}=1/(C_{\mathrm{run}}^{s}+C_{\mathrm{tr}}^{s})\quad(1\leqslant s\leqslant S) \tag{10}$$

式中:C_{run}^{s} 和 C_{tr}^{s} 分别表示第 s 个粒子的对应映射方案的任务总运行成本和迁移成本。根据子任务分配到计算单元所在的位置,定义通信成本适应度函数为

$$F_{\mathrm{com}}^{s}=1/C_{\mathrm{com}}^{s}\quad(1\leqslant s\leqslant S) \tag{11}$$

式中:C_{com}^{s} 表示第 s 个粒子的对应映射方案的任务总通信成本。粒子总的适应度函数为

$$F = \alpha F^{s}_{\text{time}} + \beta F^{s}_{\text{cos}} + \delta F^{s}_{\text{com}} \quad (12)$$

算法选择总适应度高的粒子，为进化出下一代优秀的粒子提供优良基础。α、β 和 δ 为权重系数，表示在粒子选择时更侧重于哪方面的性能。

2.3　位置及速度更新

在每一次迭代中，粒子依据自身的历史最优位置和整个群体的最优位置来更新速度和位置。第 s 个粒子经历过的最好位置记为 $B^{s}_{p}=(B^{s1}_{p}, B^{s2}_{p}, \cdots, B^{sn}_{p})$，在整个群体中，所有粒子经历过的最好位置记为 $B^{s}_{a}=(B^{s1}_{a}, B^{s2}_{a}, \cdots, B^{sn}_{a})$。用 g 表示当前算法的迭代次数，粒子根据下式更新自己的速度和位置

$$v^{g+1}_{s} = wv^{g}_{s} + c_1 r_1 (B^{s}_{p} - x^{g}_{s}) + c_2 r_2 (B^{s}_{a} - x^{g}_{s}) \quad (13)$$

$$x^{g+1}_{s} = x^{g}_{s} + v^{g}_{s} \quad (14)$$

式中：w 用来控制粒子历史速度对当前速度的影响程度，对于平衡算法全局与局部搜索能力有很大作用。w 较大时算法具有较强的全局搜索能力，w 较小时算法有利于局部搜索，采用如下线性递减惯性权重

$$w(g) = w_{\text{s}}(w_{\text{s}} - w_{\text{e}})(G - g)/G \quad (15)$$

式中：w_{s}、w_{e} 分别为初始惯性权重和迭代至最大次数 G 时的惯性权重。w 随着迭代次数变化，使得算法在开始时搜索较大区域，较快地确定最优解的大致位置，之后粒子速度减慢，开始细致的局部搜索，避免算法陷入局部最优出现早熟现象，并在迭代后期仍然能够保持较强的搜索能力，提高了算法性能。

2.4　算法流程

步骤 1　按照上文描述，对粒子群位置和速度进行随机初始化；

步骤 2　计算每个粒子的适应度 F，设置粒子的 B_p 和 B_a；

步骤 3　如果 B_p 和 B_a 在多次迭代中始终保持不变，或者算法达到最大迭代次数，输出最优解，算法结束，否则转至步骤 4；

步骤 4　对每个粒子根据式(13)、(14)更新粒子的速度和位置；

步骤 5　转至步骤 3。

3　对比实验及分析

本文从两方面对所设计的调度映射方案进行对比和评价，一是算法本身的执行速度，通过对相同规模的任务分别按照遗传算法(genetic algorithm, GA)、PSO 算法及本文改进的粒子群算法(optimized particle swarm optimization, oPSO)进行运算。这部分在 Matlab2012b 上完成，软件环境是惠普的 Z800 工作站，CPU 为 Xeon E5530，主频为 2.4 GHz，内存为 8 GB，操作系统为 64 位 Windows XP 系统，实验中所需不同规模的任务由任务产生工具(task graph for free, TGFF)[15]产生，粒子数 S 设置为 100，迭代次数均为 200 次，群体最优值的停滞代数设为 5，α、β 和 δ 均设置为 1，加速因子 $c_1=c_2=2$，不同任务规模下算法运行时间 t 对比如图 1 所示。

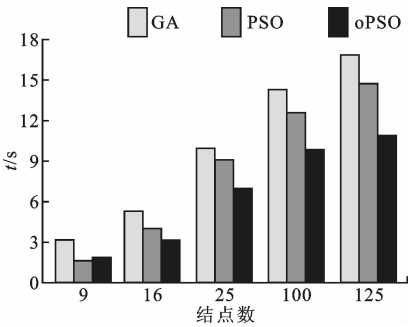


图 1　算法执行时间对比

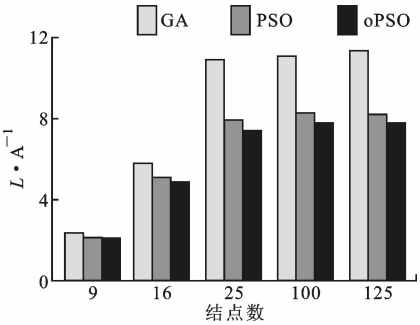
当任务规模较小时，oPSO 相比 PSO 算法优势不大，运行时间甚至会略大于 PSO 算法，这是因为 oPSO 在进行粒子更新操作时多了对权值 w 的更新计算，增加了运算步骤和运行时间，算法优势随任务规模的扩大逐渐明显。我们统计了 200 次迭代中得到最优解的次数，GA 算法是 43 次，PSO 算法是 69 次，而 oPSO 算法能达到 131 次，运算结果明显优于其他两种算法。

对比不同映射结果在运行过程中产生的包平均延时 L 和系统功耗 W。实验运行平台参数同上，片上网络拓扑采用常用的 Mesh 结构，路由算法使用 XY 路由算法，对比结果如图 2 所示，当任务规模较小时，3 种算法调度效果性能差异不大，随着任务规模不断扩大(子任务数为 100 时)，无论是包平均延时还是系统功耗，oPSO 算法的调度效果都要明显优于其他两种算法。

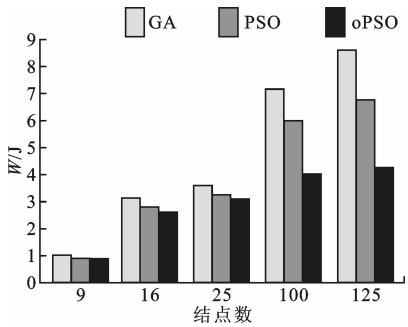
同时，实验记录了片上网络上每个交换节点处的功率情况，图 3 是节点规模为 100 的情况下，分别执行 3 种算法时各个节点上的功率分布，可以看到，oPSO 算法能将任务较为均衡地映射在整个系统上，不会出现过度繁忙或空闲的节点或区域，功耗较为均衡。

4　结　论

本文分析了片上网络系统任务调度映射中的关键问题，设计了新任务模型，使其能够更加真实地反映子任务间的制约关系；以单位时间运行成本作为衡量标准，算法调度结果更加高效、实用；调度目标

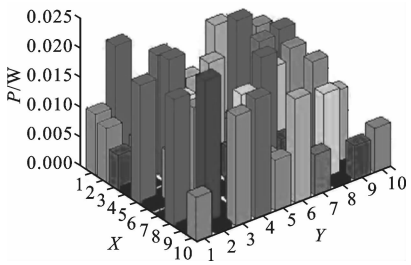


(a)包平均延时对比

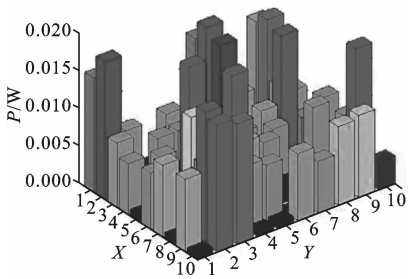


(b)系统功耗对比

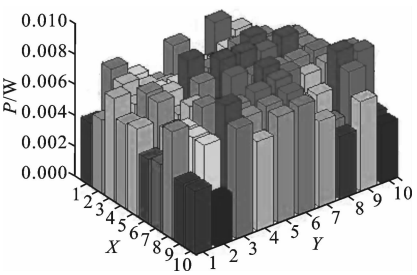
图 2 算法执行效果对比



(a)GA 算法节点功率分布



(b)PSO 算法节点功率分布



(c)oPSO 算法节点功率分布

图 3 交换节点上的功率分布比较

不仅考虑任务总的完成时间,同时也考虑时间成本和资源成本,从而达到系统性能的综合优化;改进了 PSO 算法,使其适于解决大规模任务调度问题,克服了算法前期搜素过快,容易出现早熟现象,后期搜索能力下降,易陷入局部最优的缺点。

参考文献:

[1] ADDO-QUAYE C. Thermal-aware mapping and placement for 3-D NoC designs [C]// Proceedings of the 2005 IEEE International SOC Conference. Piscataway, NJ, USA: IEEE, 2005: 25-28.

[2] SINGH A K, WU Jigang, PRAKASH A, et al. Mapping algorithms for NoC-based heterogeneous MPSoC platforms [C]// Proceedings of the 12th Euromicro Conference on Digital System Design, Architectures, Methods and Tools. Piscataway, NJ, USA: IEEE, 2009: 133-140.

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

[4] DAOUD M I, KHARMA N. Efficient compile-time task scheduling for heterogeneous distributed computing systems [C]// Proceedings of the 12th International Conference on Parallel and Distributed Systems. Piscataway, NJ, USA: IEEE Computer Society, 2006: 1-9.

[5] WU Minyou, GAJSKI D D. Hypertool: a programming aid for message-passing systems [J]. IEEE Transactions on Parallel and Distributed Systems, 1990, 1(3): 330-343.

[6] CHUNG Y C, RANKA S. Applications and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed memory multiprocessors [C]// Proceedings of the Supercomputing '92. Piscataway, NJ, USA: IEEE, 1992: 512-521.

[7] ANMAD I, KWORK Y K. A new approach to scheduling parallel programs using task duplication [C]// Proceedings of the International Conference on Parallel Processing. Piscataway, NJ, USA: IEEE, 1994: 47-51.

[8] SAYUTI M N S M, INDRUSIAK L S. Real-time low-power task mapping in networks-on-chip [C]// Proceedings of the 2013 IEEE Computer Society Annual Symposium on VLSI. Piscataway, NJ, USA: IEEE, 2013: 14-19.

(下转第 125 页)