同构多核/众核处理器任务分配自适应模拟退火算法

闫 乔 覃志东 王绍宇 闫红曼
(东华大学计算机科学与技术学院 上海 201620)

摘 要 随着多核/众核处理器核心数快速增加,任务分配解空间急剧增大,降低近似解的相对偏差越来越难。提出一种自适应模拟退火算法,建立了模拟退火算法中参数与优化环境任务数和核心数的关系。核心数的增加不但可以有效降低近似解的相对偏差,而且使任务分配算法具有较高的环境自适应能力。与较近研究成果相比较,在16核心时,自适应模拟退火算法迭代次数增加41%,相对偏差降低86%。

关键词 众核处理器,模拟退火算法,任务分配
中图法分类号 TP311.52 **文献标识码** A

Adaptive Simulated Annealing Algorithm for Task Assignment on Homogeneous Multi/Many-core Processors

YAN Qiao QIN Zhi-dong WANG Shao-yu YAN Hong-man
(School of Computer Science and Technology, Donghua University, Shanghai 201620, China)

Abstract With rapid increasing of the number of cores in multi/many-core processors, the task assignment solution space increases sharply so that reducing the relative deviation of the approximate solution becomes more and more difficult. An adaptive simulated annealing algorithm was put forward by establishing the relationship of the algorithm parameters and the number of optimized environment tasks and cores. The increasing of core number can not only effectively reduce the relative deviation of the approximate solution, but also shows high adaptability for the environment. Experiments reveal that on the 16 cores platform, the adaptive simulated annealing algorithm iterations are increased by 41%, but the relative deviation is decreased by 86% versus the recent research results.

Keywords Many-core processor, Simulated annealing algorithm, Task assignment

1 引言

多核/众核任务分配与调度是公认的NP-Hard问题[1],加之任务执行时间等参数都是近似估计值,往往采用启发式或者近似算法求解这类问题。模拟退火算法(Simulated Annealing,SA),作为一种通用的智能启发式算法,以其计算过程简单、鲁棒性强,在多核/众核任务的分配与调度中应用更广泛[2-8]。

文献[4]针对同构平台DSP综合,比较了SA和启发式表调度算法:SA在核心数较少时,具有明显优势;但核心数增加时,其优化性能较差。文献[6]针对同构无通信代价任务,比较了SA算法和27种近似算法:SA算法具有一定的自适应性,但其自适应能力有待加强;相比混合式近似算法,其近似解的相对偏差有待改善。Orsila[7]针对这种不足,依据优化环境选取SA算法参数,使得算法具有较好的自适应能力,但其优化性能对任务量的大小比较敏感。

随着多核/众核处理器的核心数不断增加[9],多核/众核任务分配的解空间规模急剧增大,以往基于SA算法的任务分配算法不能很好地平衡迭代次数和近似解相对偏差的关系。Orsila[7]的工作代表着较新的研究成果,具有较好的自适应能力,但其对任务量的大小比较敏感。同时,同构多核/众核处理器以其较好的规整性和可扩展性,广泛应用于数据并行性领域[10-12],本文针对这一现状,提出一种基于同构多核/众核平台软件任务分配的自适应模拟退火(Adaptive Simulated Annealing,ASA)算法。当解空间随着核心数增加而急剧增大时,ASA算法能够在控制迭代次数快速增长的情况下,极大降低近似解的相对偏差,具有更高的自适应能力。

本文先对多核/众核软件综合方法的基本流程以及本文所采用的软件任务、硬件平台模型进行介绍;然后对基于其中核心环节——任务分配所提的ASA算法进行详细阐述;最后进行实验对比和分析。

2 多核/众核软件综合流程

多核/众核软件综合(Multicore/Manycore Sofeware Synthesis)方法是当前学术界和产业界研究的热点问题[14]。其主要环节如图1所示,包括建立任务软件与处理器逻辑模型、任务分配与调度、任务编译与平台绑定等。

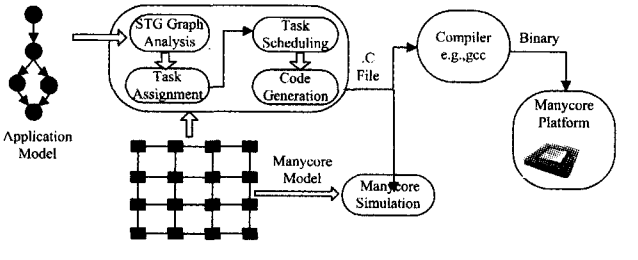


图1 多核/众核软件综合流程图

到稿日期:2013-07-20 返修日期:2013-11-03 本文受国家自然科学基金(60903160),中央高校基本科研业务费专项基金(11D11209)资助。
闫 乔(1986—),男,硕士生,主要研究方向为多核/众核软件综合方法,E-mail:yanqiao2435@163.com;覃志东 男,博士,副教授,主要研究方向为嵌入式与可编程系统;王绍宇 男,博士,讲师,主要研究方向为图像处理;闫红曼 女,博士,高级工程师,主要研究方向为嵌入式系统。

(1)任务软件与处理器抽象建模

任务软件建模就是针对具体应用软件抽取任务属性，如任务执行时间、依赖关系与通信量等，并对程序进行适当的并行转换以得到适于在众核体系结构上处理的并行任务模型。处理器抽象建模就是对硬件平台内核数、处理能力及互联结构进行抽象，是任务划分与调度的逻辑目标平台。

常采用的并行软件任务模型有：Standard Task Graph (STG)[13]、Synchronous DataFlow Graphs (SDF)[14]、Kahn Process Networks (KPN)[7]、Hierarchical Task Graph (HTG)[7]以及 Directed Acyclic Graph (DAG)[14]等。早稻田大学为多核/众核系统提供了一组 STG 基准测试软件模型库，并为每个 STG 图提供了在不同核心数下的最优分配解等相关数据。本文采用 STG 图作为软件任务抽象模型，如图 2 所示。一个 STG 可以用一个二元组 $G(V,E)$ 表示。其中，V 为顶点集合，$V=\{v_i, i=1,\cdots,N\}$；E 为边集合，$E=\{e_{ij}; 1\leqslant i<j\leqslant N\}$。$v_i$ 表示第 i 个节点的计算量，e_{ij} 表示任务 i 和任务 j 的优先次序，若其值为 1，则 i 必先于 j 处理；若为 0，则其无约束关系。

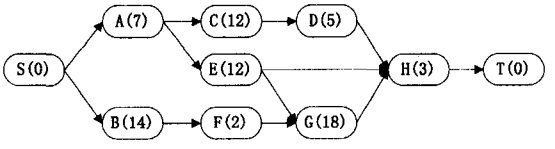


图 2 标准任务图(STG)

多核/众核处理器按连接方式可分为总线结构、交叉开关结构和 mesh 结构等。本文处理器模型采用 mesh 网络结构，用二元组表示为 $C=\{P,F\}$，$P=\{p_i; i=1,\cdots,M\}$，$F=\{f_i; i=1,\cdots,M\}$，p_i 代表第 i 个核心序号，f_i 代表第 i 个核心的每秒执行的指令数。由于是同构平台，因此各 f_i 的值都相等。

(2)任务分配与调度

任务分配是指根据并行任务模型将一个多任务软件划分为多个任务集合，建立各个任务集合与处理器内核的映射关系。任务调度是指确定各个内核上每个任务执行的先后次序。任务分配与调度是多核/众核软件综合的核心环节。

任务分配与调度的优化策略取决于具体的系统优化目标函数，如系统可靠性[2]、吞吐率[14]、能耗以及收敛时间[16]等。

为验证本文所提 ASA 算法任务分配的有效性，在任务分配后，需要采用常用的 B_level 调度策略[15]进行任务调度。

(3)任务编译与平台绑定

任务编译是指将分配和调度好的任务编译为可执行代码。而平台绑定是指将这些基于逻辑处理器分配和调度好的可执行代码映射到物理平台上，并建立通信路由。至此，整个多核/众核软件综合流程完成。

3 多核/众核任务分配的 ASA 算法

多核/众核并行任务的分配问题常采用诸如模拟退火算法(Simulated Annealing, SA)、遗传算法(Genetic Algorithm, GA)、禁忌搜索算法(Tabu Search, TS)、蚁群算法(Ant Colony, AC)、粒子群算法(Particle Swarm Optimization, PSO)等启发式或者近似算法求解。这些算法的优缺点如表 1 所列。

表 1 常用任务分配算法的优缺点比较

算法	优点	缺点
SA	实现简单，使用灵活，可收敛到最优解。	收敛速度和近似解质量依赖于很多参数的设定。
GA	具有大范围全局搜索能力，潜在的并行性、随机性，鲁棒性强。	易收敛到局部最优解，操作复杂，不易控制，难以处理非线性约束问题。
TS	利用已有知识，具有较高的求解质量和效率。	对初始解依赖性较强，易陷入局部最优解。
AC	鲁棒性和并行性较好，易于并行实现。	算法复杂，收敛速度慢，容易出现停滞现象。
PSO	个体可利用自身和群体经验，收敛速度快。	不善于处理离散的优化问题，容易陷入局部最优。

标准 SA 算法框架如图 3 所示。理论上可证明 SA 算法是能够以概率 1 收敛到全局最优解的。但标准 SA 算法收敛速度慢，执行时间长，优化性能依赖于参数的选取。本文在标准 SA 基础上，结合多核/众核平台的具体情形，利用目标优化环境相关信息，建立其与 SA 算法参数的关系，在解空间随着核心数增加而急剧增大的情况下，限制算法迭代次数的增长速度，提高算法近似解相对偏差的下降速度，以提高算法自适应能力。

```
SIMULATED-ANNEALING(S0)
    S <- S0
    C <- COST(S0)
    Sbest <- S
    Cbest <- C
    Rejects <- 0
    T <- T0
    k <- 0
    while (T > Tf)
    do Snew <- MOVE(S)
       Cnew <- COST(Snew)
        distC= Cnew - C
       r <- RANDOM()
       p <- PROB(distC,T)
       if (distC < 0 or r < p)
         then if (Cnew < Cbest)
                   then  Sbest <- Snew
                         Cbest <- Cnew
              S <- Snew
              C <- Cnew
              Rejects <- 0
       else   Rejects <- Rejects + 1
              if (Rejects >= Rejectsmax)
                 then break
       k <- k+1
       if (k == Rejectsmax)
         then T <- TEMPERATURE-COOLING(T0,tem_params)
              k <- 0
       i <- i+1
    return Sbest
```

图 3 标准模拟退火(SA)算法框架

本文所提 ASA 算法参数的选取依据如下：解空间仅由任务数和核心数决定，即 $O(M^N)$，其中任务数为 N，核心数为 M。ASA 算法的迭代次数与解的质量也主要由这两个因素决定，以下详细介绍这种 ASA 算法：

(1) 初始温度和 T_0 终止温度 T_f 的选取

Orsila[7]所提算法中，T_0 和 T_f 的设定皆依赖于任务量的大小，由于解空间规模只与任务数和核心数有关，因此在 ASA 算法中，T_0 和 T_f 的选取只依赖于任务数和核心数。在 SA 算法中，T_0 越高、T_f 越低，其迭代次数就越高，在这种情况下，虽然提高了寻找到最优解的概率；但是收敛速度比较

慢。在某些优化规模比较大的情况下，可能需要好几个小时甚至更长的时间才能得到优化结果。为提高优化速度和优化效果，我们充分利用优化环境信息，设定 T_0 和 T_f：

$$T_0=e^{a-\frac{1}{M}} \tag{1}$$

$$T_f=e^{b+\frac{1}{M+N}} \tag{2}$$

N 表示任务数，M 表示核心数。搜索空间随着任务数和核心数的增加而增大，所以设置初始温度 T_0 为核心数的增函数，终止温度 T_f 为任务数和核心数的减函数。a 和 b 是常数，且小于零，a 和 b 的值保证了一个安全的降温范围，在 ASA 算法中 $a=-4$，$b=-9$，优化算法具有较高的优化性能。

(2)每个温度级别的迭代次数 L

迭代次数 L 在 SA 算法分析中代表在不同的温度级别下马尔科夫链的长度。L 越高，总的迭代次数就越高。在搜索过程中，搜索空间随着核心数和任务数的增加而增大，相应的总的迭代次数就应该越高，Orsila 所提方法中，L 取值为：

$$L=(M-1)*N \tag{3}$$

此值是一个解空间的领域规模，考虑到 SA 算法本身的随机性，即在同一温度下，可能搜到相同的解，为提高解的近似度，本文设定 L：

$$L=M*N \tag{4}$$

M 和 N 的含义同式(1)，L 同样是任务数和核心数的增函数。

(3)连续拒绝次数的最大值 $Reject_{max}$

为提高优化速度，ASA 的结束条件有两种：一种是达到了结束温度 T_f，另一种是连续拒绝次数达到了 $Reject_{max}$。这个值越大，其优化度越高，相应的优化速度就越低。文献[7]中 $Reject_{max}$ 和 L 设定为 $Reject_{max}$ 相同的值，当温度 T 较高时，迭代的总次数较少，解的近似度较低；温度较低时，迭代的总次数较多，解的近似度较高。因此，为提高优化的收敛速度，$Reject_{max}$ 应随着温度降低而减小，本文设定：

$$Reject_{max}=L*(\frac{d*T}{T_0}) \tag{5}$$

T_0 为初始温度，T 为当前温度，d 是大于等于 1 的常数，d 值用于减缓 $Reject_{max}$ 的下降速度，以提高解的近似度。在我们的算法中，当 $d=3$ 时，具有最优效果。

(4)接受概率函数 ASA_prob

Orsila 算法接受概率函数为：

$$Orsila\_prob=\frac{1}{1+e^{(\frac{\Delta C}{0.5C_0T})}} \tag{6}$$

此形式的接受概率函数来源于生物上神经网络中的神经细胞的典型 S 状响应曲线。本文接受概率函数为：

$$ASA\_prob=e^{\frac{-\Delta C}{C_0T}} \tag{7}$$

虽然两种形式的优化效果相差不大，但是式(7)不依赖于手工参数的选取，提高了算法的适应能力。C_0 为初始温度 T_0 的初始代价，ΔC 表示当前代价与前一次迭代代价之差。ASA_prob 随着温度的降低而减小；且 ΔC 越大，ASA_prob 的值越小。

(5)降温函数 $Temperature\_cool\_new$

传统 SA 算法的降温函数一般为：

$$Temperature\_cool=T_0*q^i \tag{8}$$

为将每个温度下的迭代次数 L 融合进降温函数中，本文和文献[7]采用相同的降温函数：

$$Temperature\_cool\_new=T_0*q^{\lfloor\frac{i}{L}\rfloor} \tag{9}$$

其中，T_0 为初始温度，i 为迭代次数，L 为每个温度的迭代次数，q 为降温常数，设定为 0.95。

4 实验对比及结果分析

4.1 衡量标准

为了衡量实验结果，本文以近似解与最优解的相对偏差 R、加速比 $Speedup$ 及迭代次数作为算法的衡量标准。由于相对偏差和加速比的解在解空间中是相同的而且各个任务图的并行度不同，导致加速比相差比较大，因此采用相对偏差更方便对最优解进行衡量。在实验讨论中，多采用相对偏差。

定义 1(加速比 $Speedup$) 同一个多任务软件在单核处理器系统和多核处理器系统中运行消耗时间的比率，计算公式如下：

$$Speedup=\frac{T_s}{T_m} \tag{10}$$

T_s 为任务在一个核心上的执行时间，T_m 为采用分配算法后任务在多核/众核系统上的执行时间。

定义 2(相对偏差 R) 绝对偏差与最优解的比值，绝对偏差是指近似解与最优解的差值，计算公式如下：

$$R=\frac{Cost_{app}-Cost_{opt}}{Cost_{opt}} \tag{11}$$

$Cost_{app}$ 是用 ASA 任务分配算法计算出的近似解，$Cost_{opt}$ 是任务的最优解。相对偏差 R 值越小，表明近似解与最优解越接近，算法寻优能力越强。

定义 3(最大并行度 $Parallelism$) 同一时刻执行的任务数或者同一时刻工作的核心数，计算公式如下：

$$Parallelism=\frac{Task_{sum}}{Task_{cp}} \tag{12}$$

$Task_{sum}$ 表示任务图中所有任务的总计算量，$Task_{cp}$ 表示任务图中关键路径上的任务计算量。这个量越低，任务图的并行度越高。为提高精确度，我们对每个任务图分别在 4 核、8 核以及 16 核上各运行 10 次，取 10 次优化的平均值。

4.2 实验环境及实验方法

本文所有程序代码都采用 C 语言实现，程序运行环境为 VC6.0，实验主机安装的是 Windows XP 系统，CPU 为 Intel (R) Core(TM)2 Duo CPU E7500 @ 3.93Hz，内存 1GB。

实验时，需要在文献[13]下载包含 100 个节点的任务模型文件库，随机选取 10 个文件。每一个文件表示一个任务模型，文件中除构建任务模型所需的数据外，还包括一些其它信息，如任务总量、并行度以及关键路径长度等。在设计任务分配与调度算法代码时，需要首先利用任务模型文件构建 STG 任务图，并保存任务并行度，以供后续相对偏差的比较。然后设定处理器模型，包括处理器核心数、核心频率以及通信方式等。以 STG 任务图和处理器模型为输入，依次设计任务分配、任务调度和目标函数等代码模块。每一次任务随机分配的结果都需要目标函数的判定，在多次迭代后，最终会产生一个相对偏差较小的近似解。

4.3 新型算法的分配结果

表 2—表 4 是对 10 个任务图实验结果的平均统计。其中表 2 和表 3 分别是 Orsila 算法和本文 ASA 算法的实验结果，表明：随着核心数的增加，两种算法的迭代次数和加速比

均增加，而相对偏差在降低，说明这两种算法随着核心数的增加，优化能力都提高，具有环境自适应能力。表 4 是 ASA 算法与 Orsila 算法相比较的结果，由此表可以看出，与 Orsila 所提算法相比，随着核心数的增加，ASA 算法的迭代次数增加，但是相对偏差降低，特别是当核心数增加到 16 时，ASA 算法迭代次数增加 41%，而优化能力提高了 86%，表明随着核心数的增加，ASA 算法优化能力所提高的速度远远高于迭代次数增加所花代价，说明 ASA 更适合于众核系统。

表 2　Orsila 算法不同核心下实验平均值

核心数	4 核	8 核	16 核
迭代次数	26600.08	53200.05	106400.14
加速比	2.1514742	3.5221257	4.7214372
相对偏差	0.79491297	0.37243029	0.030616506

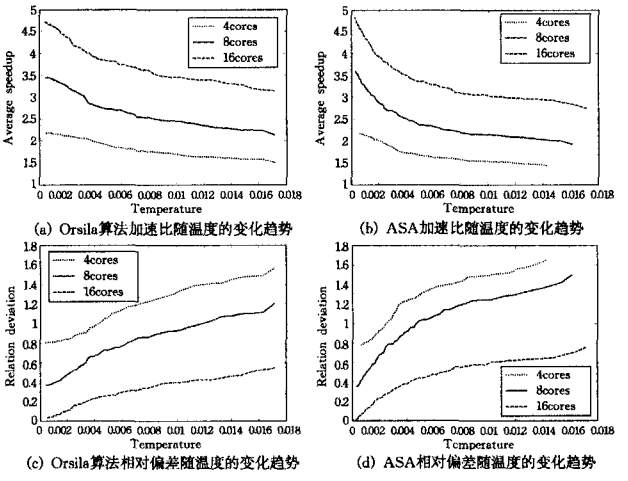
表 3　ASA 不同核心下实验平均值

核心数	4 核	8 核	16 核
迭代次数	28269.41	61431.63	150624.06
加速比	2.1638955	3.5664273	4.8286922
相对偏差	0.7826026	0.3536088	0.0096059

表 4　ASA 与 Orsila 算法实验平均值对比

核心数	4 核	8 核	16 核
迭代次数	0.06275657	0.15472880	0.415638198
加速比	0.00591365	0.01257808	0.022781587
相对偏差	−0.0154864	−0.0505369	−0.68625094

图 4(a)(b)和(c)(d)分别表示 Orsila 所提算法和 ASA 算法在不同核心数情况下的降温过程中加速比和相对偏差的变化。从这 4 幅子图可以看出，核心数增加，加速比增大，相对偏差降低，这反映出两种算法都具有环境自适应性。与传统 SA 不同，这两种算法在温度较低时，解的优化度提高比较快，正如图 4(a)(c)、图 4(b)(d)所示，在温度低于 0.006 时，两种算法加速比的增加以及相对偏差的降低都较快，但是 ASA 算法的变化速度明显高于 Orsila 所提算法，在降温结束时，ASA 算法的加速比和相对偏差都优于 Orsila 所提算法。

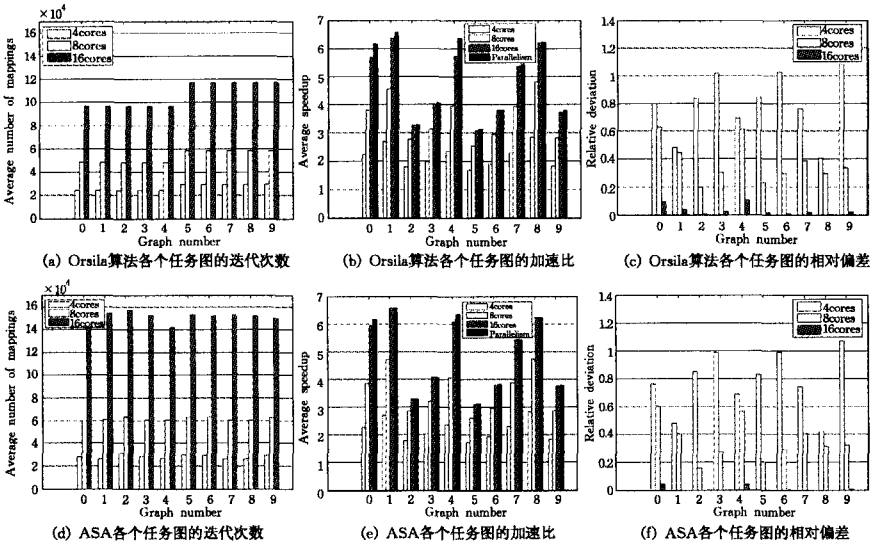


(a) Orsila算法加速比随温度的变化趋势　(b) ASA加速比随温度的变化趋势

(c) Orsila算法相对偏差随温度的变化趋势　(d) ASA相对偏差随温度的变化趋势

图 4　ASA 和 Orsila 算法降温过程中加速比和相对偏差变化

从图 4 还可以观察到，当核心数不同时，Orsila 所提算法的初始温度和结束温度几乎没有变化。而 ASA 算法的初始温度和结束温度不同，随着核心数增加，初始温度升高，结束温度降低。因为核心数增加，解空间规模增大，自适应提高初始温度和降低结束温度可以增加迭代次数，进而保证解的质量。这也是 ASA 算法较 Orsila 的一个显著改进点。相比结束温度，初始温度的变化比较大，这是因为其相对于结束温度的绝对值比较大。



(a) Orsila算法各个任务图的迭代次数　(b) Orsila算法各个任务图的加速比　(c) Orsila算法各个任务图的相对偏差

(d) ASA各个任务图的迭代次数　(e) ASA各个任务图的加速比　(f) ASA各个任务图的相对偏差

图 5　ASA 算法和 Orsila 算法迭代次数、加速比和相对偏差的对比

图 5(a)(c)(e)和图 5(b)(d)(f)分别是 Orsila 和 ASA 算法在不同核心数时，各个任务图的迭代次数、加速比以及相对偏差的实验统计结果。为降低实验误差，针对不同核心数、不同任务图，两种优化算法都分别运行 10 次，取平均值，以提高实验的准确度。通观 6 幅子图，可以看出，随着核心数的增加，迭代次数增加，加速比增加，相对偏差降低，这表明两种优化算法具有环境自适应能力。任务分配的解空间规模为 $O(N^M)$，N 为核心数，M 为任务数。要提高解的质量，就需要增加迭代次数，而核心数增加时，问题规模增大，所需要的迭代次数也需要增加。图 5(a)(d)和图 5(c)(f)显示，ASA 算法的迭代次数高于 Orsila 所提算法，相对偏差低于 Orsila 所提算法，且随着核心数的增加，迭代次数增长速度加快，相对偏差下降速度也加快。由表 4 可知，在 16 核心时，迭代次数平均增加 41%，相对偏差平均降低 86%。图 5(a)中前 5 个任务图的迭代次数几乎相同，后 5 个任务图的迭代次数也几乎相

（下转第 53 页）

cking and line balancing[C]//Proceedings of the IEEE International Conference on Robotics and Automation. 1992:1186-1192

[8] Ülker Ö, et al. A Grouping Genetic Algorithm Using Linear Linkage Encoding for Bin Packing[J]. Lecture Notes in Computer Science,2008,5199:1140-1149

[9] Emmerich M,et al. An EMO Algorithm Using the Hypervolume Measure as Selection Criterion[J]. Lecture Notes in Computer Science,2005,3410:62-76

[10] 方锦明. 云计算中基于 NSGA-II 的虚拟资源调度算法[J]. 计算机工程与设计,2012,33(4)

[11] Beume N,et al. SMS-EMOA:Multi-objective selection based on dominated hypervolume[J]. European Journal of Operational Research,2007,181(3):1653-1669

[12] Calheiros R N,Ranjan R,De Rose C A F,et al. CloudSim:A novel framework for modeling and simulation of cloud computing infrastructures and services[R]. GRIDS-TR-2009-1. Parkville, VIC:The University of Melbourne Australia,Grid Computing and Distributed Systems Laboratory,2009

(上接第 21 页)

同,而且后 5 个任务图的迭代次数要远高于前 5 个。这是因为前 5 个任务图的任务量范围是 1～10,而后 5 个任务量范围是 1～20。随着任务量的增加,Orsila 算法的迭代次数增加。ASA 算法的各任务的迭代次数只与核心数和任务数有关,所以各个任务图的迭代次数相差不大。对比图 5(d)(e)和图 5(f),ASA 算法的迭代次数在一定程度上反映了加速比和相对偏差,特别是当核心数为 16 时,任务 0 和任务 4 的迭代次数较少,加速比与并行度的差值较大,其相对偏差较高;其余任务的迭代次数较多,其相对偏差较少,其中任务 1、任务 2、任务 3、任务 7 和任务 8 均达到了最优解。这比较符合迭代次数与解的优化度之间的矛盾关系,Orsila 算法没有这一优势。

比较图 5(c)和图 5(f),ASA 算法在各个核心数各个任务图的相对偏差都低于 Orsila 算法,而且随着核心数的增加,相对偏差降低比较快,其中在 16 核心时,任务 1、任务 2、任务 3、任务 7 和任务 8 均达到了最优解,而 Orsila 算法均未能达到最优解,这表明 ASA 算法在核心数达到一定程度时,是可以寻到最优解的,这一性质对于某些要求最优解的应用领域非常重要。

结束语 为了平衡基于 SA 算法在多核/众核任务分配上的迭代次数与解精度的关系,本文在深入研究 SA 算法原理和多核/众核任务分配特点的基础上,建立初始温度、终止温度、每个温度级别的迭代次数、连续拒绝次数的最大值、接受概率函数、降温函数等 SA 算法参数与处理器内核数和任务数之间的优化关系,提出了一种 ASA 算法。本算法与已有的 Orsila 算法相比,能在处理器内核数增加的情况下,有效控制算法迭代次数增长速度,极大降低近似解的相对偏差,具有更好的环境自适应能力。

SA 算法虽然提出比较早,也较早应用在多核/众核任务分配中,但是因为算法本身的复杂性以及多核/众核平台的不断发展,我们还需要对模拟退火算法做进一步深入研究,使其能够适应更为复杂的环境,具有更高的优化效率和更快的收敛速度。

参 考 文 献

[1] Partitioning S V. scheduling parallel programs for execution on multiprocessors[M]. MIT Press,1989

[2] Huang L, Yuan F, Xu Q. On task allocation and scheduling for lifetime extension of platform based mpsoc designs[J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22 (12):2088-2099

[3] Kim J-K,Shivle S,Siegel H J,et al. Dynamically mapping tasks with priorities and multiple dead-lines in a heterogeneous environment[J]. ParallelDistrib. Comp. ,2007,67(2):154-169

[4] Koch P. Strategies for realistic and efficient static scheduling of data Independent algorithms onto multiple digital signal processors[R]. Technical report. The DSP Research Group, Institute for Electronic Systems, Aalborg University, Aalborg, Denmark, December 1995

[5] Ferrandi F,Pilato C,Sciuto D,et al. Mapping andscheduling of parallel C applications with ant colony optimization onto heterogeneous reconfigurable MPSoCs[C]//Design Automation Conference(ASP-DAC), 2010 15th Asia and South Pacific. 2010: 799-804

[6] Coroyer C, Liu Z. Effectiveness of heuristics andsimulated annealing for the scheduling of concu-rrent tasks an empirical comparison[M]//Rapport derecherche de l'INRIA Sophia Antipolis. 1991:452-463

[7] Heikki O. Optimizing algorithms for task graph mapping on multiprocessor system on chip SoCs[D]. Tampereen teknillinen yliopisto,Julkaisu Tampere University of Technology, Publication,2011

[8] 温平川,徐晓东. 并行遗传/模拟退火混合算法及其应用[J]. 计算机科学,2003,30(3):86-89

[9] 彭晓明,郭浩然,庞建民. 多核处理器——技术、趋势和挑战[J]. 计算机科学,2012,39(Z11):320-326

[10] Wentzlaff D,Griffin P,Hoffmann H,et al. On chip interconnection architecture of the tile processor[J]. IEEE MICRO,2007, 27(5):15-31

[11] Zhang Y P,Jeong T,Chen F,et al. A study of the on chip interconnection network for the ibm cyclops64 multicore architecture [C]//Proceedings of the 20th international conference on Parallel and distributed processing,ser. IPDPS'06. Washington,DC, USA:IEEE Computer Society,2006:64-74

[12] Fan Dong-rui, Yuan Nan, Zhang Jun-chao, et al. Godson-t: An efficient many-core architecture for parallel program executions [J]. Journal of ComputerScience and Technology,2009,24(6): 1061-1073

[13] Standard Task Graph Set [OL]. http://www.kasahara.elec.waseda.ac.jp/schedule/,2005-12-20

[14] Hashemi M. Automated Software synthesis for streaming applications on embedded manycore processors[D]. University of California,Davis,2011

[15] Kw Y K,Ahmad I. Static scheduling algorithms forallocating directed task graphs to multiprocessors[J]. ACM Comput. Surv. , 1999,31(4):406-471

[16] 王颖锋,刘志镜. 面向同构多核处理器的节能任务调度方法[J]. 计算机科学,2011,38(9):294-297