

面向动态可重构系统的低能耗调度算法

敬 超¹, 朱燕民^{1,2}, 李明禄^{1,2}

(1 上海交通大学 计算机科学与工程系, 上海 200240; 2 上海市可扩展计算与系统重点实验室, 上海 200240)

摘 要: 本文主要研究了动态可重构系统的能耗最优化问题, 提出了一种基于蚁群算法的低能耗调度算法 (ASR)。该算法以蚁群算法的概率状态转移规则为中心, 通过设计合理的启发函数 (即任务放置策略) 达到最小化系统能耗目的。通过实验模拟, 与蛮力算法比较后发现, 采用提出的 ASR 调度算法, 不仅使得系统损失的总能耗比最优能耗高 3%, 而且算法复杂度低, 运行速度快。

关键词: 动态可重构系统; 调度算法; 低能耗; 变形; 蚁群算法

中图分类号: TP316.4

文献标识码: A

文章编号: 1000-7180(2012)09-0184-05

Energy-efficient Scheduling on Dynamically Reconfigurable Systems

JING Chao¹, ZHU Yan-min^{1,2}, LI Ming-lu^{1,2}

(1 Department of Computer Science and Engineering, Shanghai Jiaotong University, Shanghai 200240, China;

2 Shanghai Key Laboratory of Scalable Computing and Systems, Shanghai 200240, China)

Abstract: This paper studies the crucial problem of energy-efficiency scheduling on reconfigurable systems, present an Ant Colony Optimization based energy-efficient scheduling algorithm called ASR for reconfigurable system. We design a heuristic function (task placement strategy) to serve as the center for probabilistic state transition rule. Based on this strategy, the overall energy consumption is reduced. Finally, we conduct comprehensive trace-driven simulation experiments to evaluate our algorithm and the results show that our energy efficient scheduling algorithm successfully processes all tasks without violating deadline requirements, and the proposed algorithm consumes 3% up to that of the brute-force algorithm.

Key words: dynamically reconfigurable system; scheduling algorithm; energy efficiency; transformable; ant colony optimization algorithm

1 引言

可重构系统能耗问题的研究主要集中于两个方面: 体系结构级和系统级。体系结构级解决能耗问题的方法, 主要通过设计硬件系统来达到低能耗目的。而对于系统级, 主要采用调度技术解决能耗问题。文献[1]提出了一种后放置方法减少漏电损耗。方法的局限性在于仅可用在存在约束的任务调度。文献[2]针对任务分区在给定和未给定的前提下, 分别提出了最优和逼近算法达到系统能耗最优化的目标, 但

是忽略了可重构系统的重构开销。文献[3]考虑了重构开销, 但是以提高系统性能为目的。

本文研究的问题是在给定时间约束的条件下, 最优化重构系统的能耗。与以往的研究不同, 除了考虑可重构系统的特点外, 根据任务资源的需求的多样性, 提出了一种可变形的任务模型。模型的特点是分配的资源数目与任务运行时间成反比, 即分配的资源越多运行时间越短, 如图 1 所示。其次, 结合新任务模型和系统特征, 将问题进行了形式化描述, 并分析了问题的复杂性。接着, 提出了一种快速、复杂

收稿日期: 2012-02-12; 修回日期: 2012-03-12

基金项目: 国家自然科学基金项目 (61170238, 60903190); 国家“八六三”计划项目 (2009AA012201); 博士点基金 (20100073120021)

度低的能耗调度算法(ASR),该算法以蚁群算法的概率状态转移规则为核心,结合系统特征设计任务放置策略,从而达到降低系统能耗目的.最后,通过小规模数据实验模拟,将ASR与蛮力方法比较,结果表明算法消耗的能量仅比蛮力算法高3%.

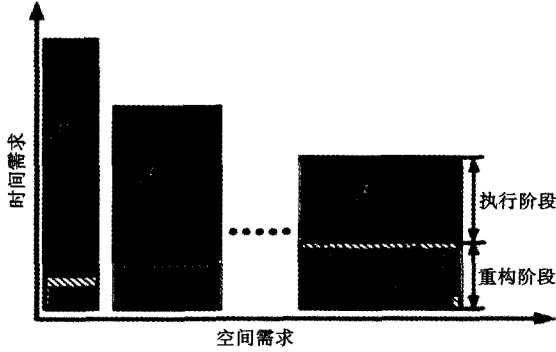


图1 变形任务模型时空图

2 相关工作

可重构系统的节能研究工作主要是通过改变系统体系结构或电路实现系统节能的目标.这方面的工作主要有:F. Li [4]等人出了一种灵活的FPGA体系评估架构(fpgaEVA-LP)来分析基于LUT体系结构的能耗.文献[5]设计了一种采用dual-V_{dd}/dual-V_t的FPGA电路有效的降低了动态能耗.还提出了基于功耗感知的CAD算法解决任务放置问题.在文献[6]中,由于FPGA系统中互连组件的重要性,提出了一种V_{dd}可编程互联结构降低功耗.这些方法都是通过改变已有系统硬件结构来达到能耗最优化目的.

另一组研究工作则是侧重于系统级,通过调度技术来降低系统能耗.这方面的工作有[1]提出一种后放置策略降低系统漏点损耗.然而方法只适用于具备依赖约束关系的任务集,而对于无约束关系的任务该方法则不适用.文献[2]则是对给定任务分区和未给定分区,分别提出了最优化能耗算法和逼近算法.方法的应用表明调度技术的使用同样也可以达到降低可重构系统能耗的目的.

在现有的能耗优化问题研究中,与可重构系统能耗优化最接近的研究就是经典的多核能耗优化.文献[7]的遗传算法是最为常用的一种方法,通过自然物种基因遗传原理,得到问题的“最优”解.但是,在该算法中,基因要经历变异过程,使得“好”的基因无法保留,而且还扩大了搜索空间加大算法的复杂性.因此,经典的遗传算法也不适用.

3 问题建模

3.1 系统模型

系统模型采用部分动态可重构系统.为简化问题的复杂性,假设系统中的逻辑资源成一维模型组成.系统中共有 K 个可重构逻辑块,逻辑块集合可以表示为 $B = \{B_1, \dots, B_K\}$.系统的可重构端口有限,任务必须依次重构至系统才可执行.

应用由 N 个独立的、非抢占的任务组成,任务集可表示为: $T = \{T_1, \dots, T_N\}$.应用的完成必须遵从统一的时限约束,每个任务主要由两个元组构成:工作周期和候选块.

- 工作周期:任务的完成所需要经历的周期数,用 $C = \{C_1, \dots, C_N\}$ 表示.工作周期的运行完成需经历重构阶段和执行阶段;

- 候选块:任务有 M 个候选块供选择.候选块由不同数目的、连续的逻辑块组成.对于任务 T_i 候选块可以表示为: $\{F_{(i,1)}, \dots, F_{(i,M)}\}$.根据可重构系统的高度可并行能力,假设任务为可变形模型,即分配候选块内逻辑块数目越多任务的运行时间越短.

3.2 能耗模型

对于CMOS电路,动态功率的消耗可以通过如下的公式计算:

$$P(S) = K_{ef} \times V_{DD}^2 \times S \quad (1)$$

其中 K_{ef} , V_{DD} , S 分别表示有效负载容量,供电电压,系统运行速率.

根据文献[2],可以将系统运行速率定义为:

$$S = k_h \times (V_{DD} - V_T)^{\epsilon} / V_{DD} \quad (2)$$

从已有研究中,已知 $\epsilon = 2$, $V_T = 0$,因此从(1), (2)可以推出系统功耗计算公式:

$$P(S) = \mu \cdot S^3 \quad (3)$$

其中 $\mu = K_{ef} / kh^2$.因此,系统的功耗主要与 S 有关.

在能耗模型中,假设从小到大依次有 N_f 个离散的速率级别,级别的选择会影响所有的任务运行速率,因此, T_i 选用候选块 $F_{(i,j)}$ 以速率 f 运行后产生的能耗为

$$E_{(i,j)} = \mu \cdot (S_{(i,j)}^f)^2 \cdot C_i \quad (4)$$

所以,应用完成后所有任务消耗的系统能量为

$$\sum_{i=1}^N E_{(i,j)} = \sum_{i=1}^N \mu_{(i,j)} \cdot (S_{(i,j)}^f)^2 \cdot C_i \quad (5)$$

其中, $\forall j \in [1, M]$, $f \in [1, N_f]$.

3.3 问题定义

本文主要研究线下最优化能耗问题,应用开始

运行后,任务的完成时间必须满足时限 D 的前提,最小化系统总体能耗,问题的形式化定义如下

$$\begin{aligned} \min \left(\sum_{i=1}^N E_{(i,j)} \right) &= \sum_{i=1}^N \mu_{(i,j)} \cdot (S_{(i,j)}^f)^2 \cdot C_i \\ \text{s. t. } L &\leq D \end{aligned} \quad (6)$$

其中 L 表示最后一个任务完成后总运行时间。

3.4 复杂性分析

定理 1 可重构系统的能耗最优化问题为 NPC 问题。

证明:要证提出的问题是 NPC 问题,分两个步骤:首先,证其为 NP 问题. 给定问题的上界,猜一个可能的解 E_b ,在多项式时间内验证该解, E_b 小于等于上界,该问题属于 NP 问题。

接着要证明问题为 NPC 必须将一个已知的 NPC 归约于该问题. 将经典的二维装箱问题归约至该问题,构造实例输入为 N 个任务,候选块个数为 1,单级系统速率,则可将二维装箱问题在多项式时间内归约至要证明问题。

因此,定理 1 为 NPC 问题,证毕。

4 算法设计

4.1 算法基本思想

文中提出了基于蚁群算法的低能耗调度方法 (ASR)。由于蚁群算法在时间复杂度的优越性,所以算法的主要思想基于蚁群算法. 在满足时限的前提下,以最优化系统能耗为目标来确定任务的调度方案. ASR 主要分为以下三步:

- 为了满足任务的时限需求,将可重构系统的速率设置为最大;
- 利用蚁群算法中的启发函数和转移规则,每只蚂蚁逐步形成的调度方案. 接着,信息激素根据全局信息激素准则挥发,重复该步骤,达到终止条件,得到最佳的任务调度方案(最短的总运行时间);
- 基于调度方案,在满足时限的前提下,动态调节系统的运行速率,从而达到降低系统的能耗目的,并最终得到最优解。

4.2 算法设计细节

首先,算法初始化生成 N 只蚂蚁,用 a_k 表示第 k 只蚂蚁. $J_{ak}(R)$ 表示 a_k 在第 R 轮位于某节点后下一步还可以转移的节点. 节点表示候选块被分配给任务,如节点 (i, j) 表示任务 T_i 分配给候选块 $F(i, j)$. 在 ASR 中,一旦满足了概率状态转移规则,蚂蚁做出转移决策,表明任务选了某候选块后被放置于系统. 接着, W 和 $J_{ak}(R)$ 被更新,其中 W 用来保存

蚂蚁得到的解, $L(R)$ 用来保存每轮的最佳调度方案. 算法重复运行最大轮次 $EC_{\max}$ 。

下面分别定义了算法中的概率状态转移规则和全局挥发准则:

$$Pr_{(i,j) \rightarrow (i',j')}^{ak} = \frac{[\phi_{(i',j')}^{ak}]^{\alpha} \cdot [\eta_{(i',j')}^{ak}]^{\beta}}{\sum_{(i',j') \in J_{ak}(R)} [\phi_{(i',j')}^{ak}]^{\alpha} \cdot [\eta_{(i',j')}^{ak}]^{\beta}} \quad (7)$$

式(7)为概率状态转移规则,其中 ϕ 表示 a_k 从节点 (i, j) 到 (i', j') 的信息激素; η 为辅助 a_k 状态转移决策的启发函数,下一节将详细描述相应的策略 FFIT; Pr 表示 a_k 从节点 (i, j) 到 (i', j') 的转移概率; α, β 分别表示信息激素和启发函数在转移决策时的重要性。

$$\phi(n, m) \leftarrow (1 - \rho)\phi(n, m) + Q/Pr_{(s, sj) \rightarrow (e, ej)}^{ak} \quad (8)$$

式(8)为全局信息激素更新规则,当所有蚂蚁都形成各自的调度方案后,信息激素的挥发采用当前最优策略, V 表示蚂蚁 a_k 形成的最优调度方案起点为 (si, sj) 终点为 (ei, ej) ,其对应的信息激素也会被更新. ρ 表示挥发速率,速率越大算法收敛得越快。

4.3 启发函数

本节提出了左底首次满足紧凑任务放置策略 (First-fit bottom-left task placement algorithm, FFIT) 来实现启发函数。

该任务放置策略的设计作用不仅是为蚂蚁做出状态转移提供参考,还可以提高系统资源的利用率. 可重构系统中逻辑资源的利用率对任务的完成时间起着非常重要的作用,如果逻辑资源利用低,则会延长任务的完成时间. 如图 2 所示, X, Y 轴分别表示可用资源和完成时间,由于任务 T_1, T_2 放置不佳,造成资源利用率过低,所以 T_3 到来后即使空闲资

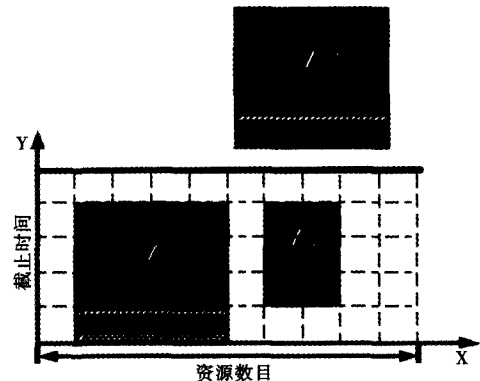


图 2 可重构系统资源利用问题示意图

源足够,也无法满足时限的需求。

左底首次满足紧凑放置策略可以解决该问题。它的基本思想:在满足系统约束的前提下,以左对齐的方式将任务放置于最低层,这样做的好处可以使任务放置后,提高资源利用率,缩短任务完成时间。

4.4 算法分析

算法1中 N 只蚂蚁的一轮的算法复杂度为 $O(MN^3)$ 。如果最大轮次为 $EC_{\max}$,算法的时间复杂度为 $O(EC_{\max} \cdot MN^3)$ 。

算法1 ASR

```

输入:  $T$ : 任务集合;
       $F$ :  $M$ 个候选块;
       $D$ : 任务截止时间;
       $S$ :  $N$ 个系统速率级;
输出: 最小能耗 $E$ 
      调度方案Schedule

Begin
  初始化产生大小为 $N$ 的蚂蚁集合;
  其中系统速率设为最大 $N_r$ ;
  For  $R$ : =1 to  $EC_{\max}$  Do
    将蚂蚁放置各个节点;
     $k$ : =1,  $W$ : =0
    While  $k \leq N$  Do
      蚂蚁 $ak$ 遍历完剩余节点后,
      将结果保存至 $W(ak)$ ;
       $k++$ 
    End
    选出最佳的解保存至 $N(R)$ ;
    采用最优解更新由式(8)全局更新;
  End
  将各个轮次中最优分配方案保存,并记录下 $L$ ;
  While  $L \leq D$  Do
    逐个降低速率,降低全局能耗;
  End
End
输出结果: 分配方案Schedule,  $E$ 

```

算法2中时间复杂度为 $O(N \cdot PO)$ 。其中 PO 表示可以放置点集合大小。

算法2 FFIT

```

输入:  $T$ : 任务集合 $N_r$ ;
       $B$ : 可重构逻辑资源集合
输出: 任务完成时间 $Y$ 

Begin
  初始化  $i$ : =1, 生成ListTable;
  将蚂蚁放置各个节点;
  While  $i \leq N_r$  Do
    For  $j$ : =1 to  $PO$  Do
      If  $T$ 符合放置在 $j$  Then
        更新集合 $B$ 中用逻辑资源数目;
        将当前系统可用资源数目及可放置点,
        保存于ListTable;
        退出For循环;
      Else
        Break
      End
    End
     $i++$ 
  End
  根据ListTable计算机完成时间结果存于 $Y$ 
End
输出结果: 任务完成时间 $Y$ 

```

5 实验模拟

5.1 实验设置

可重构系统中包含的资源个数分别为100和200两类,每类对应的任务集合大小分别为6,8,10,每个任务对应的候选块集合大小为三组。工作周期数目服从随机均匀分布,见表1所示。

表1 实验参数设置

资源数目	工作周期		候选块
	可重构周期数	执行周期数	连续逻辑块数目
100	[1, 5]	[5, 20]	[20, 80]
200	[1, 10]	[10, 30]	[80, 120]

假设可调节的速率级别数目为10,算法用C#实现,模拟平台为Dell PC机,系统Windows XP Service Pack2,双核3.00G Hz,内存2 GB。

5.2 算法比较

5.2.1 蛮力算法

蛮力算法(Brute-Force algorithm, BF)通过暴力计算枚举所有低能耗调度方案,从中获得最优方案。算法复杂度为 $O(N! \cdot M^N)$ 。

5.2.2 爬山算法

爬山算法(Hill Climbing based algorithm, HCA)是一种常用的启发式算法,它主要是通过反复迭代,局部交换的方式得到最优解。爬山算法的时间复杂度为 $O(MN^3)$ 。

5.3 实验结果分析

图3将ASR分别与BF和HCA比较。与HCA相比,ASR分别节约了6%和11.88%的能耗。这是因为提出算法采用了全局最优化的而HCA得到是局部最优。同时,与BF相比,ASR算法多消耗了12.67%和3%的能耗。图4将三种算法的最短完成时间进行了比较,ASR得到的最短完成时间最接近最优值。当任务完成时间越短,节约的能耗就越多。

表2表明了三种算法运行时间开销。尽管ASR需要的时间开销大于HCA,但是其降低的能耗要优于HCA。表3说明了算法挥发速率与最大轮次的关系,即挥发速率越大,收敛越快轮次越少。同时也表明了ASR是一个复杂度低的算法。

表2 三种算法的时间开销

算法	资源个数	任务数目	时间开销
BF	200	10	≥ 10 小时
HCA	200	10	2.43 秒
ASR	200	10	31.23 秒

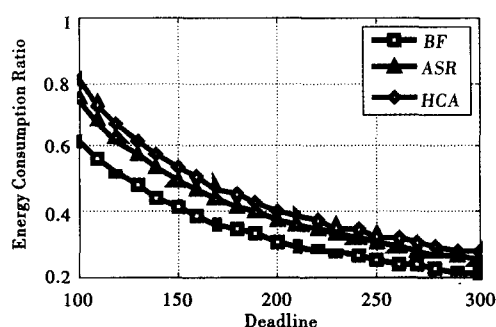
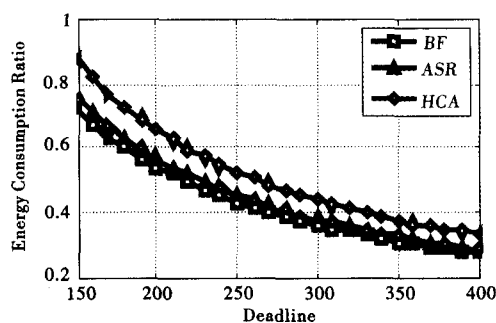
(a) $K=100$ (b) $K=200$

图3 三种算法能耗比较示意图

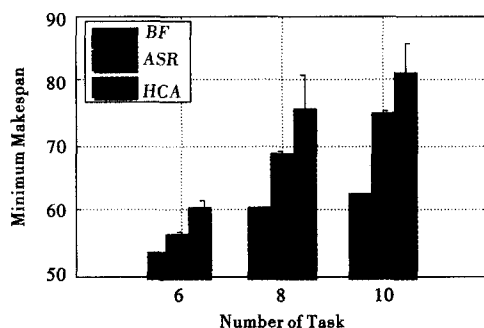
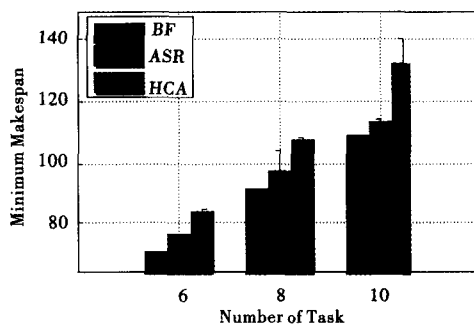
(a) $K=100$ (b) $K=200$

图4 三种算法最短完成时间比较示意图

表3 挥发速率与最大轮次的关系

挥发速率	0.1	0.3	0.5	0.7	0.9
最大轮次	65	19	9	4	2
完成时间	106	106	113	113	113

6 结束语

本文基于可重构系统的高度并行能力,任务对资源需求的多样性,首先提出了一种新的变形任务模型。接着,面向可重构系统中的挑战,如重构开销、重构互斥性等,以最小化能耗为目标,提出了复杂度低、快速的基于蚁群算法低能耗调度算法。实验结果表明,采用 ASR 调度算法后系统的能耗降低,仅比采用蛮力算法得到的最优能耗高 3%。

参考文献:

- [1] C F Li. Post-placement leakage optimization for partially dynamically reconfigurable FPGAs[C]// Proc International Symposium on Low Power Electronics and Design (ISLPED) Portland, Oregon, USA: IEEE, 2007.
- [2] Perng N C. Energy-efficient scheduling on multi-context FPGA's[C]// Proc International Symposium on Circuits and Systems (ISCAS) Island of KOS, Greece: IEEE, 2006.

- [3] Angermeier J. Heuristics for scheduling reconfigurable devices with consideration of reconfiguration overheads [C]// Proc IEEE Reconfigurable Architecture Workshop (RAW) on IPDPS miami, Florida, USA: IEEE, 2008.
- [4] Li F. Architecture evaluation for power-efficient FPGAs[C]// Proc ACM International Symposium on Field-Programmable Gate Arrays (FPGA) Monterey, CA, USA: ACM, 2003.
- [5] Li F. Low-power FPGA using pre-defined dual-Vdd/dual-Vt fabrics[C]// Proc ACM/SIGDA International Symposium on Field Programmable Gate Arrays (FPGA) Monterey, CA, USA: ACM, 2004:42-50.
- [6] Lin Y. Power modeling and architecture evaluation for FPGA with novel circuits for Vdd programmability[C]// Proc ACM International Symposium on Field-Programmable Gate Arrays (FPGA) Monterey, CA, USA: ACM, 2005.
- [7] Hou E S H. A genetic algorithm for multiprocessor scheduling[M]. Transactions on Parallel and Distributed Systems (TPDS), 1994,5(2):113-120.

作者简介:

敬超男, (1983—), 博士生. 研究方向为高性能计算、云计算。