文章编号:1674-2974(2016)08-0151-06

一种面向动态异构多处理器的任务调度算法*

张海燕[1],刘 彦[2†],陈晓明[2],赵一弘[2]
(1. 湖南大学 电气与信息工程学院,湖南 长沙 410082;2. 湖南大学 信息科学与工程学院,湖南 长沙 410082)

摘 要:提出了基于遗传算法的面向动态异构多处理器的调度算法(Heterogeneous Scheduling Genetic Algorithm,HSGA),该算法利用连续的多个调度时间片完成遗传算法的迭代计算,在保证计算效率的同时获得较好的调度结果,从而为每个应用选择符合其计算特性的处理器内核.仿真实验表明,本文算法在4核、8核和16核的平台上相比较于经典的匈牙利算法 ED^2 仅分别增加了0.4%,1.1%和1.3%,新的调度算法相比于匈牙利算法和Local调度算法具有更好的调度效果及更好的动态适应性.

关键词:遗传算法;任务调度;功耗控制
中图分类号:TP316.4 **文献标识码**:A

An Improved Scheduling Algorithm for Dynamic Heterogeneous Chip Multicore Processors

ZHANG Hai-yan[1], LIU Yan[2†], CHEN Xiao-ming[2], ZHAO Yi-hong[2]
(1. College of Electrical and Information Engineering, Hunan Univ, Changsha, Hunan 410082, China;
2. College of Computer Science and Electronic Engineering, Hunan Univ, Changsha, Hunan 410082, China)

Abstract: This paper presented an improved scheduling algorithm for dynamic heterogeneous chip multicore processors(Heterogeneous Scheduling Genetic Algorithm, HSGA). The proposed scheduling algorithm uses time slices of OS scheduler to complete the iterative procedure of HSGA, which can obtain efficient task scheduling results and choose the best process core for each application task. The experiments using SESC simulator show that the ED^2s of the proposed algorithm are only 0.4%, 1.1% and 1.3% higher than those of a baseline classic Hungarian Algorithm with 4 cores, 8 cores and 16 cores chip multiprocessor respectively with random degradation. And the proposed algorithm can generate more stable and adaptive results for unpredictable heterogeneity, compared with Hungarian Algorithm and Local Search Algorithm.

Key words: genetic algorithms; task scheduling; power control

半导体技术的飞速发展使得设计者可以将更多晶体管或者处理器内核集成到一个单芯片上从而构成片上多处理器芯片(Chip Multiprocessor,CMP).多处理器芯片已经在服务器计算、桌面系统、甚至于嵌入式计算系统中占据了重要的地位,成为目前主流的处理器结构.多核处理器为计算系统带来高性

* 收稿日期:2015-04-03
基金项目:国家自然科学基金资助项目(61300037),National Natural Science Foundation of China(61300037)
作者简介:张海燕(1976—),女,山东临沂人,湖南大学工程师,硕士
† 通讯联系人,E-mail:liuyan@hnu.edu.cn

能的同时,也在芯片可靠性方面带来了新的挑战[1-2].

随着片上多处理器芯片的规模逐渐扩大,芯片制造和使用过程中的不可控因素造成的同构处理器性能和关键参数的异构性,成为体系结构和系统层面不可忽视的因素和挑战.就算是在单个晶圆内,由于生产工艺和流程的影响也可能导致各个处理器内核的功耗、最大工作频率等关键参数不同.在这种情况下,原本按照同构片上多处理器设计的 CMP 芯片可能具有异构性[3-4].大规模同构 CMP 芯片将面临众多原本应该性能一致的计算内核在功耗和性能方面表现出不一致的情况.如果芯片中某些组件或者电路出现了故障、性能下降与延迟,通过相关技术手段可以使出现性能变化的处理器内核降级使用[5].因此,原本同构的多核片上处理器 CMP 可能由于多种不可见的因素导致其片上多个处理器内核的性能与原有设计不同.相比原设计指标将存在多个降级使用的处理器内核,此情况称为片上多处理器的动态异构性.

本文主要考虑由于制造过程和使用阶段的不可见因素导致的芯片关键参数变化时多处理器的任务调度问题.对于按同构处理器设计的 CMP,若不考虑上述不可见因素带来的异构性而进行任务调度和分配显然难以得到优化的结果.本文提出一种基于遗传算法的动态异构多处理器调度算法(HSGA),在考虑同构 CMP 处理器出现内核降级使用的情况下,调整任务调度策略,在保证芯片总体功耗满足约束的条件下获得优化的性能.

1 相关研究

已有相关研究考虑同构多处理器降级的问题.文献[4]对 CMP 处理器的制造过程的可变性对不同处理器内核工作频率的影响进行了评估,他们认为由此带来的工作频率的差异达 20%,论文中提出了一系列电路级的方法降低不利影响.文献[6]为了达到 CMP 芯片功耗控制的目标,将功耗过高的处理器内核关闭.上述工作与本文的目标类似,但他们主要是从电路级考虑和解决动态异构性带来的问题,本文则主要从操作系统的任务调度层面考虑动态异构性给 CMP 性能带来的影响.

此外,很多工作针对多核处理器上的应用程序的特征进行优化.文献[7-8]主要基于 IPC(Instruction Per Clock)统计信息对应用程序行为进行分析从而找到更为有效的任务调度策略.在同构多处理器平台中还使用了任务迁移的技术提高调度效率.文献[9]提出了基于 CPI(Clock Per Instruction)栈信息的调度算法.上述工作中对于应用程序行为分析的部分可以作为本文工作的补充,但他们的工作主要基于内核数目少的多处理器芯片,没有考虑同构多处理芯片由于内核数量增加而对任务迁移和系统信息采集方面带来的限制.

多核处理器功耗管理吸引了众多研究者的关注[10-12].Ma 等人[10]的工作主要从多处理器芯片全局功耗控制入手,使用自动控制理论对 CMP 上的处理器内核进行分类,并确定各自的工作频率,所提方法展现了良好的效果和可扩展性.文献[11]与本文工作类似,在考虑制造过程异构性的情况下通过为每个处理器内核设定合理的工作频率来最优化芯片性能.文献[12]考虑了异构多处理器平台上的动态任务调度问题,并给出了 MTS 启发式方法来解决这个 NP 难问题.但上述工作的目标平台没有考虑多处理器芯片在使用过程中的故障导致处理器内核降级使用的情况.本文的工作在具备降级使用能力的动态异构多核处理器上,提出基于遗传算法的功耗敏感任务调度算法.

2 系统模型

2.1 系统结构与假设

本文研究的多核处理器 CMP 指单个芯片上集成了多个同构处理器内核,内核之间通过总线及共享内存进行通信的架构.考虑到制造和使用过程中不可预见的故障对处理器内核性能带来的影响,文中认为部分受影响的内核可以降级使用,即降低部分关键性能参数和指标但仍能正常操作.

本文主要探索在操作系统层面应用自调整的任务调度策略将任务调度到合适的、降级的处理器内核上执行,达到降低动态异构性对多处理器芯片计算性能影响的目的.多处理器任务调度问题是 NP 难问题,难以在多项式时间内找到最优解.考虑到实际多处理器芯片上优化目标和体系结构细节的复杂性,本文做了一些假设.首先,假设多处理器芯片上运行的任务之间是独立的,忽略任务间通信,并且只考虑单线程执行的情况.这个假设可以使在对任务运行状态采样更为准确的同时不失一般性.其次,假设平均分配外存访问带宽,忽略共享外存带宽占用的情况.简化共享外存的带宽分配策略有助于专注

于任务行为特征和调度问题的研究.

2.2　问题的描述

动态异构多核处理器任务调度的目标是在给定芯片总功耗预算之下获得最佳的性能,应用程序可以是多线程或者单线程程序,一般认为任务线程的数目与处理器数目相同.单处理器上的并发多线程程序超出了本文讨论范围.文中使用 $T_{i,j}$ 来表示任务 i 在处理器 j 上运行时的吞吐率,即单位时间内执行的指令数目.使用 $P_{i,j}$ 来表示任务 i 在处理器 j 上运行时所产生的功耗.这里使用 ED^2 来表示任务 i 在处理器 j 上执行时的开销[2].为了对比方便对不同处理器上的 ED^2 值进行了归一化处理.变量 $X_{i,j}$ 用于表示任务 i 是否分配到处理器 j 运行的状态.现有 m 个任务和 n 个处理器($m=n$),动态异构多核处理器任务调度问题可以使用整数线性规划对其进行建模,如式(1)和式(2)所示.

$$\max \sum_{i=1}^{m}\sum_{j=1}^{n} T_{i,j}x_{i,j}; \tag{1}$$

$$\text{s.t.}\begin{cases} \sum_{i=1}^{m}\sum_{j=1}^{n} P_{i,j}x_{i,j} \leqslant \text{Power}_{\text{budget}}, \\ \forall_i : \sum_{j=1}^{n} x_{i,j} = 1, \\ \forall_j : \sum_{i=1}^{m} x_{i,j} = 1, \\ \forall_{i,j} : x_{i,j} \in \{0,1\}. \end{cases} \tag{2}$$

式(1)表示本文的优化目标是最大化多处理器芯片的总吞吐率.式(2)表明使用 ILP 建模时给出的约束条件,即功耗低于功耗预算、一个任务只能分配给一个处理器内核以及单个处理器内核只能执行一个任务.

3　动态异构多处理器调度算法

基于第 2 节的问题描述,本文提出基于遗传算法的在线动态异构多处理器调度算法 HSGA.考虑遗传算法在解决组合优化问题时求解质量高和算法复杂度高的特点,本节从算法设计和算法执行框架两个部分描述所提动态异构多处理器调度算法.

3.1　算法设计

对于给定的 m 个片上多处理器,n 个任务/线程(其中 $m=n$),可设计如下算法:

1)设计编码.如果任务 i 分配给处理器 j,则 i 所对应的基因为 j,而染色体编码为 n 个任务的基因组合,长度为 n.如:染色体(1,3,2,4,5)表示分别把任务 1 至任务 5 分配给编号分别为 1,3,2,4,5 的处理器.

2)适应度函数.算法的目标是在芯片出现本文所关注的动态异构性导致处理器内核产生难以预计的功耗变化时进行调整,使总体功耗最小.适应度函数使用 ED^2 指标评估处理器功耗情况,可按公式(3)进行计算,表示该代染色体代表的所有处理器总开销越小,适应度越高.

$$\text{fitness} = \sum_{i=1}^{n} \text{energy}_i \times \text{delay}_i^2 . \tag{3}$$

3)具体算法流程.本文中使用基本的遗传算法,其流程如下:

Step1　编码,根据所求解的具体问题,本文采用实数编码;

Step2　确定个体的评价函数,使用公式(3)所示的适应度函数;

Step3　在开始调度之前的算法初始化周期中获取和产生初始群体;

Step4　将调度时间片平均等分成若干个周期,并将每个周期分为探索阶段和稳定阶段两个部分.前 1/10 的时间称为算法执行周期(探索阶段),其余的时间称为调度执行周期(稳定阶段);

Step5　开始探索阶段,使用适应度函数评价群体中的个体;

Step6　使用轮盘选择法,选择下一代群体.使用多点基因交换的方式产生新的个体,从上一代群体的个体中随机地选择进行基因交换构成下一代群体;

Step7　获得上一代和新的一代中适应值最小的个体.并将本轮适应度最大的个体替换成上一代适应度最小的个体;

Step8　开始稳定阶段,使用 Step7 中获得的个体来进行调度;

Step9　若满足停机条件则停机,否则转 Step5.

若 a 为种群大小,迭代次数为 b,则算法时间复杂度为 $O[b(a+m\times n)]$.当系统规模足够大时为 $O(bmn)$,搜索空间为 $O(ba)$.进行全搜索需要的时间,即搜索空间大小为 $O(m^2)$.

3.2　算法执行框架

1)实现片上多核处理器芯片的全局功耗管理要求芯片内部的各个处理器内核具有实时可调节运行频率的能力.目前 AMD 公司的 Opteron 系列多核芯片已具备类似功能,支持芯片全局功耗管理.

2)为了执行片上多核处理器芯片的全局功耗管

理的算法,还需要芯片内部具备对每个处理器内核或者分区域内核的实时功耗监测单元. 现有 Itanium 处理器已在芯片内部设置了单独的传感器用于监测各个处理器内核的功耗情况,Itanium 处理器独立的功耗管理单元消耗 0.5 W 左右的功耗,仅占用 5%左右的芯片面积[2],却给处理器的温度和功耗管理带来极大的便利.

3)执行本文算法需要芯片内部设置任务/线程调度器和功耗管理器. 这既可由单独的处理器内核负责,也可由操作系统层负责. 调度操作与功耗管理操作均由一个较短的采样周期和一个较长的稳定周期组成的时间片内进行. 在采样周期中,通过在一小段时间内运行不同的调度方案和功率配置方案,来评估应用程序和异构性能的计算内核的性能和功耗统计信息. 上述相关调度决定会在随后的稳定周期中保持,直到下一个时间片. 图 1 为算法执行时间图,假设线程调度时间片为 100 ms,功耗管理时间片为 10 ms [1-2].

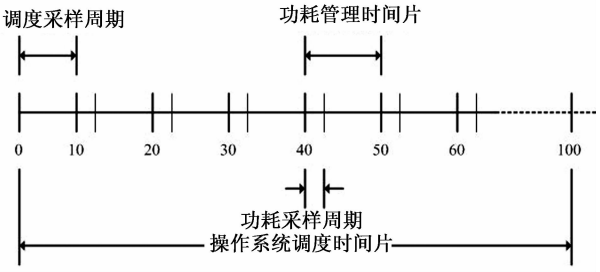

图 1 算法执行时间图

Fig. 1 Execution Chart

图 1 中,本文算法在调度采样周期获取处理器功耗开销数据(开销矩阵),获得处理器功耗参数后在功耗采样周期执行所提遗传算法对开销矩阵进行计算,并找到合适的调度方案,找到优化的调度方案后即可在后续的时间片的调度执行周期执行新的调度方案. 需要说明的是,本文所考虑的动态异构性对处理器内核性能的影响是偶发的、低频率的事件,因此对在线调度算法的实时性要求不高. 利用此特点,将遗传算法较高的计算开销分配到操作系统时间片内多个功耗采样周期中执行,一方面保证了基于遗传算法的调度方案的有效性,另一方面也使得算法的计算开销控制在可以接受的范围之内.

4 实验环境与结果分析

4.1 实验环境

本文使用与文献[2]类似的实验方法和平台. 文中主要使用 SESC 模拟器[1]模拟单个处理器. SESC 模拟器可以模拟不同体系结构的 CPU,并与能耗模型 Wattch,Cacti 和 Hotspot 配合进行功耗与温度调度方面的研究. 本文使用的测试集是 SPEC CPU2000,并在每个处理器内核上均只执行一个测试程序. 为了高效地对不同规模的片上多处理器结构进行模拟,本文使用与文献[2]类似的层次化结构组成多处理器模拟平台. 我们构建了一个由多个 SESC 模拟器构成的多处理器模拟环境来获取各个处理器性能和功耗方面的参数. 在此基础之上由一个上层框架负责信息统计、资源管理与调度决策. 通过对 SESC 模拟器的配置可以获得不同性能的、单个的处理器内核. 文中假设每个处理器具备相同的、静态分配的外存访问带宽. 为了便于实验和比较,选择如表 1 所示参数的处理器作为基准处理器内核.

表 1 基准处理器主要参数

Tab. 1 Parameters of baseline processor

部件	参数
Overall	MIPS-like out-of-order 2 GHz processor, 3-issue width
L1 I-Cache & L1 D-Cache	32 k, 8-way assoc., 2-ports, 1 cycle latency
L2 I-Cache & L2 D-Cache	32 k, 8-way assoc., 1-port, 2-banks,
Main memory	1 port, 200 cycle latency
ITLB & DTLB	64 * 8 size, 4-way assoc., 2-port, LRU-policy
Branch Predictor	Hybrid, 2 k-BTB, 2-way assoc., LRU-policy
Integer Functional Unit	2 ALUs, 2 mult/div units, 1 BJ unit, 59 registers
Floating Point Functional Unit	1 ALU, 1 mult/div unit, 59 registers

使用 8 个由表 1 所示主要参数的基准处理器组成标准的 8 核片上多处理器平台,每个单独的处理器是一个单线程、超标量、乱序执行的兼容 MIPS 指令集处理器. 通过对基准处理器的关键性能进行降级处理来对片上多处理器芯片所面临的动态异构性进行模拟. 动态异构性产生的故障可能会对 CPU 的各个方面带来不同的影响,文献[1-2]对此有较为详细的描述,这里不再详述. 本文分别采取 4 种 CPU 降级的策略如表 2 所示. 在 8 核片上多处理器的模拟器中,随机使用下面 4 种方法对同构处理器内核进行降级处理,从而模拟出具有动态异构特性的 8 核片上多处理器模拟平台.

表 2　处理器降级策略
Tab. 2　Degradation policy of processor

序号	降级部件
1	memeory_bus_delay+1, frequence * 0.8
2	frequence * 0.9, robsize/2
3	DL1/2, ALU/2
4	Frequence * 0.6

为了测试和评估本文所提算法的有效性,通过在 SPEC CPU2000 测试集中选取不同测试程序组合成不同的负载作为测试输入组合.

4.2　实验结果与讨论

本节对基于遗传算法的动态异构调度算法的实验结果进行分析和讨论.考虑到性能和功耗的平衡,在此选择 ED^2 指标作为主要评价参数[13].所有的测试数据以 ED^2 指标相对于匈牙利算法进行归一化后进行分析.

首先在 4 核异构多核处理器的环境下对算法的有效性进行评估,为了更好地进行算法实际效果的对比,此处选择以动态异构条件下调度效果较好、但时间成本很高的匈牙利算法[2]作为比较的基础.多处理器线程调度问题可以简化为经典的"指派问题",匈牙利算法解决此类问题的算法复杂度是 $O(n^3)$.Local 算法是文献[2]提出的面向动态异构多处理器的高效调度算法.通过对相邻的处理器进行线程"交换"来评估调度效果,若效果好则保留此调度方案,若效果不好则退回原分配方案,迭代进行.

图 2 为 Local 调度算法和本文所提遗传调度算法 HSGA 在 4 核动态异构多处理器条件下各个负载的 ED^2 值相对于匈牙利算法的逼近程度,其中"误差线"分别表示调度结果中的最好值和最坏值.由图 2 可知,本文所提遗传算法在 5 组随机组成的应用负载测试中都表现出比 Local 算法更好的性能.与匈牙利算法相比,所提遗传算法平均只增加了约 0.4%的 ED^2 值.值得注意的是,图 2 中"误差线"表示了在该组测试集测试过程中所产生的调度方案实际 ED^2 值的动态范围.由于我们从 SPEC2000 benchmark 测试集中随机选取测试程序组合成多线程测试负载,因此"误差线"在一定程度上反映了调度算法对于不同应用负载在整个测试周期中的动态适应性.所提遗传算法比 Local 算法表现出了更好的算法阶段行为适应性,也更适用于多核/众核处理器芯片的全局功耗控制调度.

为了进一步验证所提算法的有效性和可扩展性,论文在 8 核和 16 核环境下进行扩展实验对比,结果分别如图 3 和 4 所示.

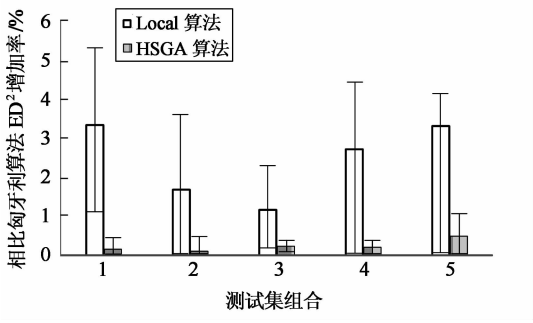

图 2　4 核调度结果
Fig. 2　Results of 4-core system

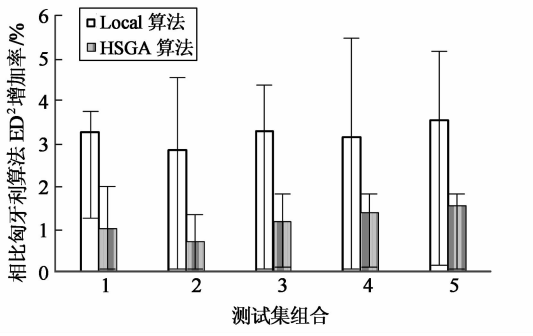

图 3　8 核调度结果
Fig. 3　Results of 8-core system

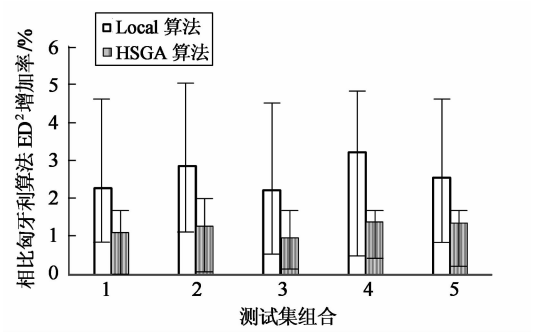

图 4　16 核调度结果
Fig. 4　Results of 16-core system

图 3 为 8 核多处理器上运行 SPEC2000 benchmark 测试集随机选取的任务负载进行测试的结果.在面临不可预知的动态异构性的情况下,Local 算法比匈牙利算法的 ED^2 增加 3%左右.本文 HSGA 遗传算法的 ED^2 仅仅增加了 1.1%左右,并依然展现出了较好的动态范围特性.图 4 为 16 核多处理器上运行 SPEC2000 benchmark 测试集的测试结果.处理器数量的增长给动态异构调度效果带来了一定

的影响,增加了算法搜索空间.但本文 HSGA 遗传算法在使用 16 核多处理器的情况下,整个应用负载 ED^2 值相比较匈牙利算法仅平均增长了 1.3%左右,展现了本文算法良好的扩展性.随着处理器数目的增加,传统匈牙利算法的复杂度将变得难以接受.本文算法考虑由于故障或者其他不可预见因素导致的多处理器动态异构性是对不同处理器结构一个偶发的影响,因此将传统遗传算法中较为复杂的算法迭代执行阶段分散到各个调度时间片执行,在不影响应用负载执行效率的情况下获得较好的线程调度效果.

5 结 论

随着单芯片上晶体管密度的不断提升,未来的片上多处理器芯片的规模将会越来越大.制造和使用过程中的不确定因素导致的可变性和故障将使得原本按照同构设计的处理器内核产生不可预见的异构性.与芯片设计时的静态异构性相比,片上多处理器不可预见性的动态异构性对软件系统的设计提出了新的挑战,即使得芯片在具备降级使用的条件时仍能获得可以接受的计算性能.

本文提出了一种基于遗传算法的面向不可预见动态异构性片上多处理器的调度算法 HSGA.当片上多处理器由于不可预见的因素导致部分处理器内核的工作频率或者性能出现变化时,本文的遗传算法会在调度时间片内对应用负载特征进行采样,并将传统遗传算法复杂的迭代过程分散到后续多个调度时间片执行,在保证计算效率的情况下提升了调度性能.文中基于 SESC 模拟器构建了多处理器环境,运行 SPEC2000 benchmark 进行了仿真实验.实验结果表明,所提遗传算法相比 Local 调度算法具有更好的调度效果和动态适应特性.下一步,我们将进一步改进算法执行效率,增加算法的可扩展性,并能适应更为复杂的应用负载.

参考文献

[1] TEODORESCU R, TORRELLAS J. Variation-aware application scheduling and power management for chip multiprocessors[C]//Proceedings of the International Symposium on Computer Architecture. Washington DC: IEEE Computer Society, 2008: 363-374.

[2] WINTER J A, ALBONESI D H. Scheduling algorithms for unpredictably heterogeneous CMP architecture[C]//Proceedings of the International Conference on Dependable System & Networks. Washington DC: IEEE Computer Society, 2008: 42-51.

[3] BORKAR S, KARNIK T, NARENDRA S, *et al*. Parameter variations and impact on circuits and microarchitecture[C]//Proceedings of the Design Automation Conference. Washington DC: IEEE Computer Society, 2003: 338-342.

[4] HUMENAY E, TARJAN D, SKADRON K. The impact of systematic process variations on symmetrical performance in chip multiprocessors[C]//Proceedings of the Conference on Design, Automation and Test in Europe. Washington DC: IEEE Computer Society, 2007: 1653-1658.

[5] SHIVAKUMAR P, KECKLER S W, MOORE C R, *et al*. Exploiting microarchitectural redundancy for defect tolerance[C]//Proceedings of the International Conference on Computer Design. Washington DC: IEEE Computer Society, 2003: 35-42.

[6] DONALD J, MARTONOSI M. Power efficiency for variation-tolerant multicore processors[C]//Proceedings of the International Symposium on Low Power Electronics and Design. New York: ACM, 2006: 304-309.

[7] KUMAR R, FARKAS K, JOUPPI N, *et al*. Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction[C]//Proceedings of IEEE/ACM International Symposium on Microarchitecture. Washington DC: IEEE Computer Society, 2003: 81-92.

[8] BECCHI M, CROWLEY P. Dynamic thread assignment on heterogeneous multiprocessor architectures[C]//Proceedings of the 3rd Conference on Computing Frontiers. New York: ACM, 2006: 29-40.

[9] KOUFATY D, REDDY D, HAHN S. Bias scheduling in heterogeneous multi-core architectures[C]//Proceedings of the 5th European Conference on Computer Systems. New York: ACM, 2010: 125-138.

[10] MA K, LI X, CHEN M, *et al*. Scalable power control for many-core architectures running multi-threaded applications[C]//Proceedings of the International Symposium on Computer Architecture. Washington DC: IEEE Computer Society, 2011: 449-460.

[11] WINTER J, ALBONESI D, SHOEMAKER C. Scalable thread scheduling and global power management for heterogeneous many-core architecture[C]//Proceedings of the International Conference on Parallel Architectures & Compilation Techniques. Washington DC: IEEE Computer Society, 2010: 29-40.

[12] LIU G, PARK J, MARCULESCU D. Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems[C]//Proceedings of the IEEE International Conference on Computer Design. Washington DC: IEEE Computer Society, 2013: 54-61.

[13] MARTIN A J. Towards an energy complexity of computation[J]. Information Processing Letters, 2001, 77(77): 181-187.