

## 基于 big. LITTLE 架构的多目标功耗自适应控制方法

冯国富<sup>1,2,3</sup>, 舒玉娟<sup>1,3</sup>, 陈 明<sup>1,3</sup>, 董立夫<sup>1,3</sup>

(1. 上海海洋大学 信息学院, 上海 201306; 2. 江苏中洋集团股份有限公司, 江苏 南通 226600;

3. 农业部渔业信息重点实验室, 上海 201306)

**摘 要:** 针对移动计算系统功耗约束条件时常变动, 以及动态电压频率调节无法有效克服静态功耗导致的能量损失等问题, 提出一种多目标功耗自适应控制方法。根据实时功耗约束制定调核策略, 确定处理器核类型及数量, 结合操作系统线程亲和性、进程迁移与处理器热插拔完成处理器核的开启、关闭及负荷管理, 实现功耗自适应。在典型多核应用 MapReduce 模型 Phoenix 与可变形部件模型上的实验结果表明, 该方法能够按需调度核类型及数量来完成计算任务, 与传统功率恒定系统相比, 执行时间与能耗平均减少 60.91% 和 48.54%, 有效提高目标系统能效。

**关键词:** big. LITTLE 架构; 功耗约束; 静态功耗; 多核处理器; 自适应控制

开放科学(资源服务)标志码(OSID):



**中文引用格式:** 冯国富, 舒玉娟, 陈明, 等. 基于 big. LITTLE 架构的多目标功耗自适应控制方法[J]. 计算机工程, 2019, 45(7): 60-65.

**英文引用格式:** FENG Guofu, SHU Yajuan, CHEN Ming, et al. Multi-objective self-adaptive power consumption control method based on big. LITTLE architecture[J]. Computer Engineering, 2019, 45(7): 60-65.

## Multi-objective Self-adaptive Power Consumption Control Method Based on big. LITTLE Architecture

FENG Guofu<sup>1,2,3</sup>, SHU Yajuan<sup>1,3</sup>, CHEN Ming<sup>1,3</sup>, DONG Lifu<sup>1,3</sup>

(1. College of Information Technology, Shanghai Ocean University, Shanghai 201306, China;

2. Jiangsu Zhongyang Group Co., Ltd., Nantong, Jiangsu 226600, China;

3. Key Laboratory of Fishery Information under Ministry of Agriculture, Shanghai 201306, China)

**[Abstract]** To address the frequent changes of power consumption constraints in mobile computing systems and the energy loss caused by the inability of Dynamic Voltage and Frequency Scaling (DVFS) to effectively overcome the static power consumption, this paper proposes a multi-objective self-adaptive power consumption control method. According to the real-time power consumption constraints, this method formulates the core adjustment strategy to determine the type and number of processor cores, and combines Operating System (OS) thread affinity, process migration and CPU hot-plugging to complete the opening and closing of cores and load management, achieving self-adaptive power consumption. Experimental results on the typical multi-core application MapReduce model Phoenix and Deformable Parts Model (DPM) show that the proposed method can schedule the cores of the suitable type and quantity to complete calculation tasks on demand. Compared with the traditional constant-power system, the execution time and energy consumption are reduced by 60.91% and 48.54% on average, which means the proposed method effectively improves the energy efficiency of the target system.

**[Key words]** big. LITTLE architecture; power consumption constraint; static power consumption; multi-core processor; self-adaptive control

**DOI:** 10.19678/j.issn.1000-3428.0052735

**基金项目:** 上海市科技创新行动计划项目“小龙虾生态化智能化设施养殖关键技术研究与应用”(16391902900)。

**作者简介:** 冯国富(1971—), 男, 副教授, 主研方向为高性能计算机系统结构、物联网、嵌入式技术; 舒玉娟, 硕士研究生; 陈 明, 教授、博士生导师; 董立夫, 硕士研究生。

**收稿日期:** 2018-09-25

**修回日期:** 2018-10-24

**E-mail:** yjshush@163.com

## 0 概述

目前,用户计算目标随多核处理器架构的日益普及和系统复杂性的不断提高而逐渐多元化,计算系统不再单纯追求高性能,计算过程中的功耗问题越来越得到重视。大量研究关注于如何在满足系统最大功耗约束条件<sup>[1-2]</sup>的同时,优化系统的执行性能<sup>[3-4]</sup>;另有研究则把实时功耗约束条件下特定任务能耗最小化作为研究重点<sup>[5-7]</sup>。移动计算系统功耗约束不仅受用户主观性能需求变化影响,还受客观供电条件的制约。电池能量变化、外部电源的有效性、新能源能量收集输出功率不稳定<sup>[8-9]</sup>等复杂条件使系统供电具有不确定性或间歇性<sup>[10]</sup>,导致功耗约束条件也具有一定的波动性和不确定性。

相对硬件层面的发展,功耗控制软件技术相对滞后,但多核系统能效的高低,较大程度上由系统架构上资源管理软件决定<sup>[11]</sup>。目前,绝大部分软件层面功耗控制研究主要关注处理器动态功耗,随着集成电路制造工艺进入纳米阶段,处理器静态功耗相比动态功耗,在总功耗中所占比重越来越大<sup>[5]</sup>。因此,静态功耗成为功耗管理的一个重点与难点。

文献[12]提出一种根据工作负载调节处理器核数量的静态功耗节能实时调度方法。文献[5]考虑处理器切换开销情况,根据状态切换时间和能量开销来确定最优调度序列。但对于同构系统,单纯依靠软件来调整工作单元数量的静态功耗管理,调度粒度太大,不易做到功耗管理的精细化。

非对称异构多核技术(如 big. LITTLE<sup>[13]</sup>)采用高性能大核和低功耗小核相组合的灵活功率配置,为在调整处理器核数量基础上精细化管理功耗提供了硬件技术基础。文献[14]在 ARM big. LITTLE 平台上,将动态电压频率调节(Dynamic Voltage and Frequency Scaling, DVFS)、负载均衡和任务迁移等各种电源管理技术整合到一个统一框架,设计并实现了基于价格理论的异构核电源管理框架。

本文基于 big. LITTLE 架构,提出一种多目标功耗自适应控制方法。利用异构核的灵活配置,在满足系统实时功耗约束条件下,以性能或能耗为目标,充分挖掘系统的能效。通过分析不同处理器核实时性能、功耗以及能耗间的关系,根据实时功耗约束条件、任务负载情况及调度目标,制定调核策略,以确定处理器核类型及数量。

## 1 处理器功耗模型与问题定义

### 1.1 处理器功耗模型

假设非对称多处理器系统由  $M$  类处理器核构成,记为  $T = \{T_0, T_1, \dots, T_{M-1}\}$ 。由文献[15]可知,电容切换活动产生的动态功耗和泄露电流产生的静态功耗是处理器功耗的两大来源。本文主要针对静态功

耗,通过使处理器核进入睡眠状态(又称关闭)或从睡眠状态返回活跃状态(又称打开)的方式,调节处理器功耗。当处理器核处于睡眠状态时,对应处理器功耗可忽略不计,近似为 0。假设基础功耗记为  $P_{base}$ ,第  $i$  类处理器核  $T_i$  处于活跃状态的数量记为  $N_i$ ,功耗记为  $P_i$ ,那么处理器的总功耗如式(1)所示。

$$P = P_{base} + \sum_{i=0}^{M-1} N_i P_i \quad (1)$$

其中,  $N_i$  不能全为 0。

以 big. LITTLE 非对称多处理器系统为例,  $M=2$ , 设高效能小核为  $T_0$ , 处于活跃状态的数量为  $N_0$ ; 高性能大核为  $T_1$ , 处于活跃状态的数量为  $N_1$ 。在通常情况下主核(编号为 0 的处理器核)不能够关闭,永远处于活跃状态,除非关机,则由式(1)可得处理器功耗如式(2)所示。

$$P = P_{base} + N_0 P_0 + N_1 P_1 \quad (2)$$

其中,当主核为  $T_0$  时,  $N_0$  不能为 0,当主核为  $T_1$  时,  $N_1$  不能为 0。

另外,根据文献[16]可知,能耗作为计算机系统一段时间内总的能量消耗,由时间  $t$  和功耗  $P$  2 个变量因素共同决定,如式(3)所示。

$$E = \int_t^{t+\Delta t} P dt \quad (3)$$

由于 big. LITTLE 架构中异构处理器具有不同的指令执行速度<sup>[4]</sup>,因此根据处于活跃状态的处理器核类型及数量,总执行速度如式(4)所示。

$$V = \frac{N_0 V_0 + N_1 V_1}{N_0 + N_1} \quad (4)$$

其中,  $V_0$  表示高效能小核的执行速度,  $V_1$  表示高性能大核的执行速度。

### 1.2 功耗管理问题定义

在满足功耗约束条件下,从 2 个角度出发权衡功耗管理方案中性能与能耗之间的矛盾。

1) 在满足功耗约束条件下,从性能优先角度出发,功耗管理问题可描述为:在满足  $P < P_{constraint}$  条件下,最大化执行速度  $V$ 。该问题可归纳为典型的规划问题,约束条件由式(2)、式(4)以及约束功耗  $P_{constraint}$  构成,目标是最大执行速度  $V_{max}$ ,并推算出  $V_{max}$  对应的处理器核资源组合。由于执行速度不易直接测量,执行速度最大化问题可转化为在任务量一定的情况下,最小化执行时间  $T$ 。

2) 在满足功耗约束条件下,从节能优先角度出发,功耗管理问题可描述为:在满足  $P < P_{constraint}$  条件下,最小化能耗  $E$ 。同样该问题也可归纳为典型的规划问题,约束条件由式(4)以及约束功耗  $P_{constraint}$  构成,目标是最小能耗  $E_{min}$ ,并推算出  $E_{min}$  对应的处理器核资源组合。

## 2 功耗自适应方法

图 1 为本文功耗自适应方法的整体架构。该

架构主要由模块①调核策略管理器、模块②核资源管理器以及模块③线程与负荷管理器 3 个部分组成。任务分解与工作者线程 (Worker) 创建由应用本身实现。

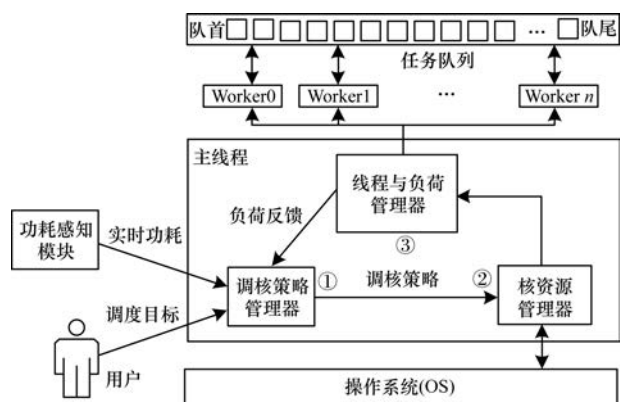


图 1 功耗自适应方法架构

1) 调核策略管理器根据实时功耗约束条件、任务负载情况及调度目标制定相应调核策略并传送给核资源管理器。调核策略既考虑不同处理器核资源与系统实时功耗、性能以及能耗的关系,同时也兼顾实时约束功耗,避免目标系统由于得不到足够外部能源供给而无法正常运行。

2) 在操作系统 CPU 热插拔机制上构建核资源管理器,用于根据调核策略通过操作系统 (OS) 提供的接口实现自动调核。由于异构处理器核具有不同的执行速度和功耗开销,因此选择合适的处理器核类型及数量对系统整体性能以及能耗至关重要。

3) 引入线程与负荷管理器负责管理工作者线程,并以共享内存、信号量等方式作为与主线程的通信机制。

## 2.1 调核策略管理器

调核策略根据实时功耗约束条件、任务负载情况及调度目标,动态确定活跃处理器核组合,使得处理器在满足功耗约束条件下达到性能最优或最大化节能,即在满足  $P < P_{\text{constraint}}$  条件下,最小化执行时间  $T$  或最小化能耗  $E$ 。

图 2 以性能优先为例描述调核策略。

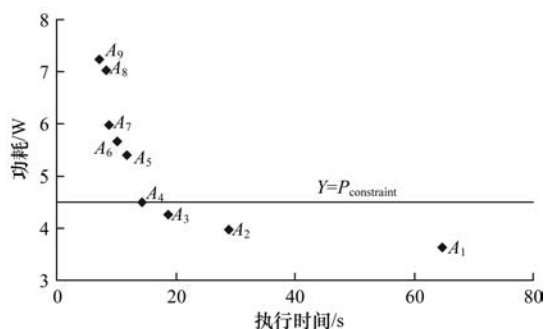


图 2 调核策略规划示意图

图 2 按处理器功耗“升序”排列,给不同处理器核类型及数量组合进行标序,并采用离散点  $\{A_1(T_1, P_1), A_2(T_2, P_2), \dots, A_n(T_n, P_n)\}$  表示不同处理器核组合  $n$  下  $T_n$  与  $P_n$  的关系。直线  $Y = P_{\text{constraint}}$  表示功耗约束条件,该直线根据约束功耗的变化而上下移动,那么约束条件由离散点  $\{A_1, A_2, \dots, A_n\}$ 、以及直线  $Y = P_{\text{constraint}}$  构成,直线  $Y$  以下区域的离散点为可行域。如图 2 所示,离散点  $A_4$  为最优解,即在当前功耗约束条件下,核资源管理器应按离散点  $A_4$  所对应的处理器核组合调整。

在系统运行中,一旦功耗约束条件  $P_{\text{constraint}}$  发生改变,且任务队列中存在未完成任务,按上述调核策略规划法所求得的有效处理器组合对系统核资源作出调整。

为避免因外部功耗约束条件波动过快造成频繁执行热插拔操作的问题,采用域值缓冲带方法加以解决。具体方法如下:当  $P_{\text{constraint}}$  提高时,只有  $P_{\text{constraint}}$  满足式(5),且连续 3 个采样周期都发出处理器调整请求时,才调整当前处理器核组合,否则保持不变;反之,当  $P_{\text{constraint}}$  降低时,只要  $P_{\text{constraint}}$  满足式(6),就对当前处理器核组合进行调整。

$$P_{\text{constraint}} \geq P_{\text{current}} + (P_{\text{current}+1} - P_{\text{current}}) \times 120\% \quad (5)$$

$$P_{\text{constraint}} < P_{\text{current}} - (P_{\text{current}} - P_{\text{current}-1}) \times 110\% \quad (6)$$

其中,  $P_{\text{current}}$  表示当前处理器组合下处理器功耗需求,  $P_{\text{current}+1}$  表示下一处理器组合下功耗需求,  $P_{\text{current}-1}$  表示上一处理器组合下功耗需求。

考虑到  $P_{\text{constraint}}$  具有不确定性,在图 1 所示的功耗感知模块中,创建线程负责实时感知约束功耗,线程入口地址为 `receive_power_func()`。随后主线程循环利用 `start` 信号量来启动该线程,即 `sem_post(&start)`。针对不同应用场景以及功耗约束条件变动情况,可对线程调度周期进行适当调整。本文设定调度周期为 1 s,该值取决于实验经验值。

## 2.2 核资源管理器

核资源管理器负责根据调核策略实现功耗自适应调核,利用 CPU 热插拔动态调整处理器核组合。

CPU 在热移除时,系统首先通过 `cpu_down()` 迫使处理器核执行 `idle` 进程,并将进程的优先级设为最高;其次将处理器核上所有进程以及处理器核上的中断、定时器、中断 `bottom half`、`tasklet` 都被迁移至其他处理器核,OS 负责完成残留任务的迁移;调用 `cpu_disable()`、`cpu_dead()` 清理体系结构相关工作,从而使该处理器核不会运行其他进程。

若需要将编号为  $i$  的 CPU 热移除,则具体实现方法是 `fputs("0", "/sys/devices/system/cpu/cpui/online")`。

CPU 热插入相对比较简单,系统无需额外配置即可识别到新插入的处理器核,由线程管理器唤醒相关线程并交由该处理器核运行,从任务队列中获取新任务并执行。若需要将编号为  $j$  的 CPU 热插入,则具体实现方法是 `fputs("1","/sys/devices/system/cpu/cpupj/online")`。

### 2.3 线程管理器

线程管理器负责根据处理器核资源的变动情况,动态控制每个工作者线程(Worker)的工作状态,并结合 OS 进程迁移与线程亲和性管理处理器负荷。

在系统初始时为每个应用程序创建与实际处理器数量相一致的 Worker,并根据处理器核资源的初始状态,控制 Worker 的工作状态。将处于活跃状态的处理器核相应的 Worker 设置线程亲和性,依次从任务队列中获取任务并执行;另外,处于睡眠状态的处理器核相应的 Worker 不设置线程亲和性,并且休眠该 Worker。

当处理器核资源发生变化时,基于以下规则管理 Worker:当执行 CPU 热移除操作,使某个处理器核由活跃状态转变为睡眠状态时,则相应的 Worker 由 OS 迁移至其他活跃处理器核上继续运行,以清空该 Worker 上正在执行的任务,然后休眠该 Worker,使其不能够继续从任务队列中获取任务并执行;当执行 CPU 热插入操作,使某个处理器核由睡眠状态转变为活跃状态时,则唤醒相应的 Worker,并调用函数 `sched_setaffinity()` 设置线程亲和性,Worker 重新从任务队列中获取任务并执行,设置亲和性,减少工作者线程在处理器核之间频繁迁移,提高 CPU cache 命中率,使性能进一步提高。

另外,在共享存储的多核系统架构下,多个线程

同时操作同一个共享变量需加锁操作,Phoenix 采用 `x86_64` 内联汇编的方式实现原子操作。因此,Phoenix 仅支持 `x86_64` 多核架构,不支持本文的 ARM 架构。针对该问题,本文利用 Linux 下与架构无关的 gcc 内置原子操作函数 `sync_fetch_and_add()` 来实现原子操作, gcc4.4 及以上版本提供该函数。实验结果表明,对于 `x86_64` 多核架构,采用 Linux 下 gcc 内置原子操作函数与采用的内联汇编方式相比,程序运行结果相同,并且运行时间和功耗也几乎相同。

### 3 实验结果与分析

本文以 64 位四 Cortex-A53 小核和双 Cortex-A72 大核的 RK3399 作为实验平台,并以 Linux 的命名习惯 `cpu0`、`cpu1`、`cpu2`、`cpu3` 代表 4 个小核;`cpu4`、`cpu5` 代表 2 个大核。操作系统采用 Ubuntu (Linux 4.4.52), gcc-4.8.5。

实验通过 HP-9800 功耗监测仪测量系统功耗,并通过 COM3 串行接口获取外部功耗约束输入。在实验中所有功耗数据不包含处理器冷却风扇功耗,每组实验执行 20 次并求平均值。

以共享存储 MapReduce 实现 Phoenix<sup>[17]</sup> 以及目标检测算法 DPM<sup>[18-19]</sup> 2 个典型多核并行应用为实验示例。表 1 以 Phoenix 提供的 Word\_count 处理 200 MB 数据,以及 DPM 对 640 像素 × 425 像素的图像进行目标匹配为例,列出不同处理器核组合下测试用例的执行时间、功耗和能耗。为方便分析,按系统功耗“升序”排列。在表 1 中,  $N_{\text{current}}$  表示系统当前处理器核数,  $T_n$ 、 $P_n$  和  $E_n$  分别表示第  $n$  组处理器组合下系统执行时间、功耗和能耗。根据 RK3399 特点,系统启动处理器核组合共 12 种。

表 1 不同处理器核组合情况下测试结果

$N_{\text{current}} =$ $N_{\text{little}} + N_{\text{big}}$	Word_count				DPM			
	序号 $n$	$T_n/s$	$P_n/W$	$E_n/J$	序号 $n$	$T_n/s$	$P_n/W$	$E_n/J$
1 = 1 + 0	1	64.85	3.63	235.41	1	37.85	3.35	126.80
2 = 2 + 0	2	29.00	3.97	115.13	2	19.04	3.64	69.31
3 = 3 + 0	3	18.81	4.26	80.13	3	12.38	3.93	48.65
4 = 4 + 0	4	14.34	4.50	64.53	4	9.14	4.18	38.21
2 = 1 + 1	—	17.38	5.13	89.16	5	7.89	5.52	43.55
3 = 2 + 1	5	11.76	5.39	63.39	6	6.58	5.83	38.36
4 = 3 + 1	6	10.28	5.67	58.29	7	5.68	6.12	34.76
5 = 4 + 1	7	8.79	5.97	52.48	8	4.91	6.41	31.47
3 = 1 + 2	—	11.54	6.31	72.82	9	4.52	7.73	34.94
4 = 2 + 2	—	9.68	6.57	63.60	10	4.04	8.08	32.64
5 = 3 + 2	8	8.21	7.03	57.72	11	3.69	8.30	30.63
6 = 4 + 2	9	7.19	7.25	52.13	12	3.47	8.61	29.88

通过分析表 1 的数据可知,采用 Phoenix 中的 Word\_count,  $P_{base} \approx 3.63 \text{ W}$ 、 $P_{little} \approx 0.28 \text{ W}$ 、 $P_{big} \approx 1.26 \text{ W}$ 。采用 DPM 用例,  $P_{base} \approx 3.35 \text{ W}$ 、 $P_{little} \approx 0.29 \text{ W}$ 、 $P_{big} \approx 2.20 \text{ W}$ 。从高效的角度出发,当系统功耗递增时,执行时间应呈递减趋势。显然,测试用例为 Word\_count,当处理器核组合“ $N_{current} = N_{little} + N_{big}$ ”为“ $2 = 1 + 1$ ”及“ $3 = 1 + 2$ ”“ $4 = 2 + 2$ ”时,不满足这一原则,也即这 3 组处理器组合无意义。为了方便表述,将表 1 中有效处理器组合分别进行编号,并将此作为后续调核标准。

为验证本文方法的有效性,以能量收集方式供电功率波动为例,模拟 3 种不同外部功耗约束条件,并将性能优先作为调度目标验证功耗自适应方法的有效性。

约束条件 A:假设约束功耗的初始值为  $P_1$ ,且每隔 1 s 增加 0.3 W,持续 13 s;之后每隔 1 s 减少 0.3 W,持续 13 s,约束功耗依此循环变化。

约束条件 B:假设约束功耗的初始值为  $P_7$ ,且每隔 2 s 减少 0.5 W,持续 8 s;之后每隔 2 s 增加 0.5 W,持续 8 s,约束功耗依此循环变化。

约束条件 C:假设约束功耗的初始值为  $P_9$ ,且每隔 1 s 减少 0.5 W,持续 4 s;之后每隔 3 s 增加 1 W,持续 6 s,约束功耗依此循环变化。

上述  $P_1$ 、 $P_7$ 、 $P_9$  具体数值见表 1。

由于传统多线程运行时系统未考虑外部功耗约束,为满足功耗约束条件,实验通过手动设定处理器

的开启,在系统运行之前初始化处理器核资源。下面为 2 种手动设定方式。

手动方式 1:由于主核除非关机永远处于活跃状态,约束功耗必须能够满足主核功耗需求,因此只将主核初始化为活跃状态,其他处理器核均设为睡眠状态。

手动方式 2:从最大化利用资源的角度出发,选择与最低约束功耗相匹配的处理器核组合初始化处理器。

在上述 3 种功耗约束条件下,当测试用例为 Word\_count 时,最低约束功耗分别 3.63 W、3.97 W 及 5.25 W,按表 1 处理器组合序号 1、2、4 初始化处理器核组合;当测试用例为 DPM 时,最低约束功耗分别为 3.35 W、4.12 W 及 5.73 W,按表 1 处理器组合序号 1、3、5 初始化处理器核组合。然而,由于功耗约束条件具有不确定性,通常无法预测最低约束功耗,因此该设定方式可行性不高,一旦供电不足,系统将会断电停机。

表 2 列出 Word\_count 数据集大小分别为 10 MB、50 MB 及 100 MB 时的系统执行时间及能耗。由表 2 可知,在功耗约束条件不确定情况下,本文方法执行时间与能耗均小于手动方式 1、手动方式 2,并且随数据量的增大,其能效优势逐步体现。同样,从表 2 中 DPM 在不同调核方式下执行时间与能耗对比可知,本文方法明显优于手动方式。

表 2 不同调核方式执行时间与能耗对比

功耗约束条件	调核方式	Word_count		DPM	
		$T_n/s$	$E_n/W$	$T_n/s$	$E_n/W$
A	手动方式 1	4.27, 8.90, 40.48	14.73, 31.06, 143.39	37.85	126.79
	手动方式 2	4.27, 8.90, 40.48	14.73, 31.06, 143.39	37.85	126.79
	本文方法	3.05, 6.56, 13.13	11.10, 25.85, 58.61	10.07	49.29
B	手动方式 1	4.27, 8.90, 40.48	14.73, 31.06, 143.39	37.85	126.79
	手动方式 2	2.83, 5.97, 19.84	10.83, 22.81, 76.58	12.38	48.65
	本文方法	2.39, 4.39, 13.81	10.37, 19.18, 64.22	9.49	46.08
C	手动方式 1	4.27, 8.90, 40.48	14.73, 31.06, 143.39	37.85	126.79
	手动方式 2	1.21, 3.25, 9.59	5.31, 14.20, 42.10	7.89	43.55
	本文方法	1.38, 2.72, 7.65	6.08, 12.43, 38.32	5.42	35.65

图 3、图 4 分别为 3 种方法的执行时间和能耗对比结果。

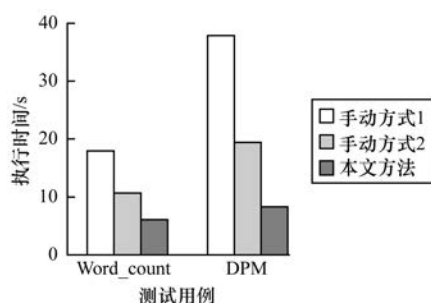


图 3 3 种方法执行时间对比结果

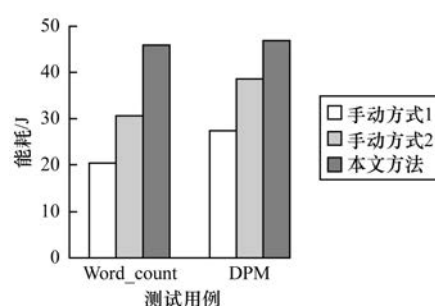


图 4 3 种方法能耗对比结果

由图 3、图 4 可知,当测试用例为 Word\_count 时,与手动方式 1 相比,本文方法执行时间平均减少

了65.77%,能耗平均减少了56.63%;与手动方式2相比,本文方法执行时间平均减少了42.80%,能耗平均减少了31.81%。当测试用例为DPM时,与手动方式1相比,本文方法执行时间平均减少了78.02%,能耗平均减少了65.56%;与手动方式2相比,本文方法执行时间平均减少了57.05%,能耗平均减少了40.18%。

实验结果表明,本文提出的功耗自适应方法不仅能够满足处理器实时功耗约束条件,具有比传统方法更优的安全性,且由于能够及时感知外部功耗约束变化,并能利用新增外部能量投入新的计算资源,使典型应用执行时间与能耗在测试案例中平均减少60.91%和48.54%,达到充分利用系统资源,提高系统能效的目的。

#### 4 结束语

由于移动计算系统受用户主观性能需求或客观供电条件制约,本文基于big. LITTLE架构,提出一种多目标功耗自适应方法。该方法结合OS线程亲和性、进程迁移与处理器热插拔完成处理器核的开启、关闭及负荷管理,以实现功耗自适应。实验结果表明,本文方法在满足处理器实时功耗约束条件以及目标系统需求的同时,能够有效减少执行时间,降低系统能耗。但本文方法没有考虑电压对静态功耗的影响,下一步对功耗管理进行研究,并将big. LITTLE架构与当前最新的多核架构进行对比分析,提高系统能效。

#### 参考文献

- [1] MA Kai, LI Xue, CHEN Ming, et al. Scalable power control for many-core architectures running multi-threaded applications [J]. ACM SIGARCH Computer Architecture News, 2011, 39(3): 449-460.
- [2] WINTER J A, ALBONESI H, SHOEMAKER C A. Scalable thread scheduling and global power management for heterogeneous many-core architectures [C]//Proceedings of the 19th International Conference on Parallel Architecture and Compilation Techniques. Vienna, Austria: [s. n.], 2010: 29-40.
- [3] WANG Zhuowei, ZHAO Wuqing, WANG Hao, et al. Three-level performance optimization for heterogeneous systems based on software prefetching under power constraints [J]. Future Generation Computer Systems, 2018, 86(1): 51-58.
- [4] 王桂彬, 杜静, 唐滔. 一种面向异构并行系统的最大功耗管理方法 [J]. 软件学报, 2013, 24(10): 2460-2472.
- [5] 张冬松, 吴飞, 陈芳园, 等. 开销敏感的多处理器最优节能实时调度算法 [J]. 计算机学报, 2012, 35(6): 1297-1312.
- [6] LU Jun, QIU Qinru. Scheduling and mapping of periodic tasks on multi-core embedded systems with energy harvesting [C]//Proceedings of Green Computing Conference & Workshops. Washington D. C., USA: IEEE Press, 2011: 1-6.
- [7] SALAMY H. Task scheduling on multi-core embedded systems under power and thermal constraints [J]. International Journal of Electronics, 2015, 102(12): 2075-2091.
- [8] 徐雷钧, 白雪, 潘祎雯, 等. 传感器节点自主供电的环境混合能量收集系统设计 [J]. 农业工程学报, 2017, 33(8): 147-152.
- [9] 张慧妍, 李爽, 于家斌, 等. 基于模糊整数规划的水质浮标光伏/蓄电池动力源配置优化 [J]. 农业工程学报, 2015, 31(19): 183-189.
- [10] 王俊, 蔡兴国, 季峰, 等. 考虑新能源发电不确定性的可用输电能力风险效益评估 [J]. 电力系统自动化, 2012, 36(14): 108-112.
- [11] 王静莲, 龚斌, 刘弘, 等. 支持绿色异构计算的能效感知调度模型与算法 [J]. 软件学报, 2016, 27(9): 2414-2425.
- [12] LI Kong. Improving multi-core server performance and reducing energy consumption by workload dependent dynamic power management [J]. IEEE Transactions on Cloud Computing, 2016, 4(2): 122-137.
- [13] GREENHALGH P. big. LITTLE processing with ARM Cortex-A15 and Cortex-A7: improving energy efficiency in high performance mobile platforms [EB/OL]. [2018-08-08]. [http://www.arm.com/files/downloads/big\\_LITTLE\\_Final\\_Final](http://www.arm.com/files/downloads/big_LITTLE_Final_Final).
- [14] MUTHUKARUPPAN T S, PATHANIA A, MITRA T. Price theory based power management for heterogeneous multi-cores [J]. ACM SIGPLAN Notices, 2014, 49(4): 161-176.
- [15] JEJURIKAR R, PEREIRA C, GUPTA R. Leakage aware dynamic voltage scaling for real-time embedded systems [C]//Proceedings of the 41st Design Automation Conference. San Diego, USA: [s. n.], 2004: 275-280.
- [16] 林闯, 田源, 姚敏. 绿色网络和绿色评价: 节能机制、模型和评价 [J]. 计算机学报, 2011, 34(4): 593-612.
- [17] RANGER C, RAGHURAMAN R, PENMETSA A, et al. Evaluating MapReduce for multi-core and multiprocessor systems [C]//Proceedings of IEEE International Symposium on High Performance Computer Architecture. Washington D. C., USA: IEEE Press, 2007: 13-24.
- [18] FELZENSZWALB P, MCALLESTER D, RAMANAN D. A discriminately trained, multiscale, deformable part model [C]//Proceedings of 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. Anchorage, USA: [s. n.], 2008: 1-8.
- [19] 俞先国. 改进的可变形部件模型及其在行人检测中的应用 [D]. 长沙: 国防科学技术大学, 2013.