星载异构计算环境下的能耗优化任务调度

安建峰[1],游红俊[2],赵伟勋[1],刘咪咪[1],张盛兵[1]
(1.西北工业大学 计算机学院,陕西 西安 710129;2.上海航天电子技术研究所,上海 201109)

摘　要: 针对航空航天探测等要求高可靠性、强实时性、低功耗的尖端领域,基于新型的异构多核计算平台对全迁移策略的支持,提出了能耗优化的实时任务调度算法。算法主要分为2部分:负载分配和任务调度。负载分配确定每个任务在不同的处理器集群上工作量的分配比例,并对系统的能耗进行优化。任务调度在负载分配的基础上,采用时间片划分的思想对已分配的任务进行合理调度,保证每个任务都能满足系统约束条件。将此算法与现有的SA算法和Hetero-Split算法进行对比,实验结果显示:此算法比Hetero-Split算法在寻求可行性任务调度方案方面相当,但强于SA算法;在能耗比方面,此算法比Hetero-Split算法在系统总能耗方面有大幅度的降低,降低比率约为23%～24%。

关键词: 二型异构; 多核处理器; 实时系统; 任务调度算法; 全迁移
中图分类号: TP 301.6　　**文献标志码:** A　　**DOI:** 10.19328/j.cnki.2096-8655.2021.04.006

Real-Time Task Scheduling for Space-Oriented Computing on Heterogeneous System

AN Jianfeng[1], YOU Hongjun[2], ZHAO Weixun[1], LIU Mimi[1], ZHANG Shengbing[1]
(1.School of Computer Science, Northwestern Polytechnic University, Xi'an 710129, Shaanxi, China;
2.Shanghai Aerospace Electronic Technology Institute, Shanghai 201109, China)

Abstract: Aiming at the cutting-edge fields such as aerospace exploration which require high reliability, hard real-time performance, and low power consumption, a real-time task scheduling algorithm for energy optimization is proposed based on the two-type heterogeneous platform which supports full migration strategy. The algorithm is mainly divided into two parts, i.e., load distribution and task scheduling. Load distribution, named EB-Split, determines the allocation of workload for each task on different processor clusters, and optimizes the energy consumption of the system. Task scheduling is based on the load distribution. It uses the concept of time-slicing to schedule the distributed tasks aptly, ensuring the system constraints. The EB-Split algorithm is compared with the existing simulated annealing (SA) algorithm and Hetero-Split algorithm through experiments. The results show that the EB-Split algorithm is equivalent with the Hetero-Split algorithm while stronger than the SA algorithm in seeking a feasible task scheduling scheme. For energy consumption, the EB-Split algorithm is much better than the Hetero-Split algorithm. Compared with the Hetero-Split algorithm, the EB-Split algorithm can reduce the total energy consumption of the system with a reduction ratio of 23%～24%.

Key words: two-type heterogeneous; multiprocessor; real-time system; task scheduling algorithm; full migration

0　引言

在诸如航天探测系统等尖端领域里,系统对安全性、可靠性、实时性以及功耗都有较高的要求,其中,安全性和可靠性一般通过硬件设备进行保障,实时性及功耗往往通过合理的任务调度方案来保证。随着近些年异构多核处理器的发展,在异构平台上,如何处理上述问题成为一种研究方向。

异构多核处理器由不同类型的处理器核组成,

收稿日期: 2021-03-19; **修回日期:** 2021-07-02
基金项目: 上海航天科技创新基金(SAST2016088)
作者简介: 安建峰(1977—),男,博士,副教授,主要研究方向为计算机系统结构。

相比较单核、同构多核处理器具有独特的优势，能够更有针对性地处理特定的事件。常见的异构多核处理器有 CPU 和 GPU，或 FPGA 的集成、ARM 的 BIG.LITTLE 架构等，根据处理器核的类型数量称之为二型异构多核处理器。在异构处理器中，研究最为广泛的就是 ARM BIG.LITTLE 架构处理器下的任务调度算法[1-3]。

异构多核处理器的任务调度已经被证明是 NP 完全问题，目前还没有算法可以在多项式时间内求得最优解，现有算法大多使用启发式的算法求得近似解[4]。异构调度模式一般分为 3 种类型：不迁移、同构核间迁移以及全局核间迁移。

1）不迁移[5-7]是指当任务分配到一个处理器核后只在该处理器核上执行；

2）同构核间迁移是指任务在执行期间可以迁移到同种类型的处理器核上执行；

3）全局核间迁移是指一个任务在执行期间可以迁移到任意类型处理器核上执行。

RARAVI 等[8]提出了用整数线性规划（Integer Linear Programming，ILP）的方法来解决同构核间迁移模式下的任务调度问题，并且提出了一种按分类进行分配的近似算法（Sort and Assign，SA）。BARUAH[9]对全局核间迁移模式下周期性任务调度的可行性进行了分析和研究，通过一组线性规划（Linear Programming，LP）约束判定对于给定处理器平台和任务集是否存在可行的调度方案。CHWA 等[10]提出了一种最优的全局核间迁移模式下的周期性任务调度算法，然而其未考虑功耗问题。李延祺等[11]提出一系列基于计算概率的建模方法，用来解决星载实时嵌入式系统中，对于具有数据依赖的非周期性任务的处理器和电压分配相关问题，在所有时间约束下具有更好的能效。杨亚琪等[12]提出一种异构多核下兼顾应用公平性和能耗的调度方法。

本文的主要工作是以二型异构多处理器为调度平台，将调度算法分为负载分配及任务调度 2 个步骤：

1）负载分配算法 EB-Split 是在 CHWA 等[10]提出的全迁移算法（Hetero-Split）基础上加入能耗优化的思想；

2）任务调度算法 BI-Wrap 则根据负载分配的结果合理安排任务时间片，保证调度的可行性。

实验表明：本算法在不损失 Hetero-Split 算法所能寻找到的可行方案数的条件下，能耗降低约 23%～24%。

1　系统模型构建

基于二型异构多核系统平台 π，该平台包含 2 个不同类型的处理器集群 π_1、π_2，每个集群 π_k 由 m_k 个完全相同的 type-k（k=1，2）类型的处理器组成。构建一个由 n 个独立的周期性实时任务 τ_i 组成的任务集 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$，其中，每一个任务 $\tau_i\in\tau$ 由 6 元组（C_i，T_i，r_i^1，r_i^2，p_i^1，p_i^2）标识，6 元组中：C_i 为任务 τ_i 的工作负载；T_i 为任务 τ_i 的周期也就是任务的相对截止时间；r_i^1、r_i^2 分别为任务 τ_i 在 type-1 和 type-2 类型处理器核上的运行速率；p_i^1、p_i^2 分别为任务 τ_i 在 type-1 和 type-2 类型处理器核上的功耗，并对于所有任务 $\tau_i\in\tau$ 的 r_i^1 和 r_i^2、p_i^1 和 p_i^2 都可以有任意给定的正实数值。设 x_i^1、x_i^2 分别为任务 τ_i 的工作负载 C_i 在 type-1 和 type-2 类型的处理器集群上的分配比例，则有：$x_i^1+x_i^2=1$。任务 τ_i 在 type-k 类型处理器上运行时的资源利用率或者资源占有率为 u_i^k，则有

$$u_i^k=\frac{t_i^k}{T_i}=\frac{x_i^k\cdot C_i}{r_i^k}\cdot\frac{1}{T_i} \tag{1}$$

特殊地，当 $x_i^1=1$ 时，称 u_i^1 是任务 τ_i 在 type-1 处理器集群上最大资源利用率并改记作 $u_i^{1,\max}$；同理，$u_i^{2,\max}$ 表示当 $x_i^2=1$ 时 u_i^2 的值。易得

$$u_i^k=x_i^k\cdot u_i^{k,\max} \tag{2}$$

约定每一个任务都独立且可抢占，可以在处理器集群内部，也可以跨集群进行任务的迁移，假定这些迁移代价可以接受。每一个任务都会被重复地调用，每一个此类调用称为一个作业或者任务实例。

调度遵从非并行执行（No Parallel Execution，NPE）限制：

1）每个处理器在任一时刻只能运行一个任务；

2）一个任务实例任一时刻只能运行在一个处理器上。

为了方便调度算法的设计，将调度方案分为 2 个子部分：负载分配和任务调度。

1）负载分配问题决定了在保证实时性的条件下，每个任务在每个类型的处理器核上该分配多少子任务，同时尽可能降低系统能耗；

2）调度生成问题在解决负载分配问题的基础上，生成一个合理的调度方式使得所有任务都可以满足截止期约束条件。

2 负载分配算法 EB-Split

基于能耗优化的二型异构负载分配算法 EB-Split 是在文献[10]中的分配算法的基础上加入能耗的因素,调整任务分配,使得在保证可调度性的条件下,尽量减小系统能耗。

2.1 可行性条件

为了保证任务的截止期约束,任务 τ_i 应该分配给 type-1 类型和 type-2 类型处理器集群的最小任务分配量分别记作 lo_i^1 和 lo_i^2。任务 τ_i 在两类集群上的剩余工作负载分别记作 y_i^1 和 y_i^2,即

$$x_i^1 = y_i^1 + lo_i^1 \tag{3}$$

$$x_i^2 = y_i^2 + lo_i^2 \tag{4}$$

为使给每个处理器集群分配的任务可调度,以下条件必须满足:

$$\overline{C_1}: \forall \tau_i \in \tau, y_i^1 + y_i^2 = 1 - lo_i^1 - lo_i^2 \tag{5}$$

$$\overline{C_3}: \sum\nolimits_{\tau_i \in \tau} y_i^1 \cdot u_i^{1,\max} \leqslant m_1 - \sum\nolimits_i lo_i^1 \cdot u_i^{1,\max} \tag{6}$$

$$\overline{C_4}: \sum\nolimits_{\tau_i \in \tau} y_i^2 \cdot u_i^{2,\max} \leqslant m_2 - \sum\nolimits_i lo_i^2 \cdot u_i^{2,\max} \tag{7}$$

$$\overline{C_5}: \forall \tau_i \in \tau, 0 \leqslant y_i^1, y_i^2 \leqslant 1 - lo_i^1 - lo_i^2 \tag{8}$$

综上,负载分配算法 Hetero-Split 理论上保证了只要存在则必然能够找到可行的负载分配方案[10]。

2.2 基于能耗的分配调整

对于给定的处理器平台和任务集,理论上可能存在多种满足要求的任务调度方案。设 SUR 代表 Surplus。

定义 type-1 和 type-2 类型处理器集群执行能力剩余量分别为 S_{UR1} 和 S_{UR2},则由约束条件 C_3 和 C_4 可得

$$S_{\mathrm{UR1}} = m_1 - \sum_{\tau_i \in \tau} u_i^1 \tag{9}$$

$$S_{\mathrm{UR2}} = m_2 - \sum_{\tau_i \in \tau} u_i^2 \tag{10}$$

能耗优化算法的思想就是在保证 S_{UR1} 和 S_{UR2} 都非负的前提下,调整 y_i^1 和 y_i^2 的大小,尽可能地降低系统总能耗。

2.3 能耗优化目标

系统总能耗 E 由各任务在 2 种类型处理器集群上的能耗之和得到,因此,在给定时间片长度 L 下,该时间片内的系统总能耗的计算公式如下:

$$E = \sum_{\tau_i \in \tau} \left(t_i^1 \times p_i^1 + t_i^2 \times p_i^2 \right) \times \frac{L}{T_i} = L \times \sum\nolimits_{\tau_i \in \tau} \left(x_i^1 \times \frac{p_i^1}{r_i^1} + x_i^2 \times \frac{p_i^2}{r_i^2} \right) \times \frac{C_i}{T_i} \tag{11}$$

由式(11)右侧化简可以看出,对于任务 τ_i 而言:当 $\frac{p_i^1}{r_i^1}$ 比 $\frac{p_i^2}{r_i^2}$ 小(大)时,将其以更大的比例分配到 type-1(type-2)类型的处理器群上可以减小系统总能耗。

2.4 能耗优化分析

根据 $\frac{p_i^1}{r_i^1}$ 与 $\frac{p_i^2}{r_i^2}$ 的大小关系,从整个任务集 τ 划分出 W_1 和 W_2 两个子集:

$$W_1 = \left\{ \tau_i \mid \tau_i \in \tau, \frac{p_i^1}{r_i^1} > \frac{p_i^2}{r_i^2}, y_i^1 > 0 \right\} \tag{12}$$

$$W_2 = \left\{ \tau_i \mid \tau_i \in \tau, \frac{p_i^1}{r_i^1} \leqslant \frac{p_i^2}{r_i^2}, y_i^2 > 0 \right\} \tag{13}$$

式中:W_1 为那些已经分配到 type-1 类型的处理器群上($y_i^1 > 0$),但是从减小系统能耗的角度则更适合分配到 type-2 类型处理器群上的任务;W_2 同理。

首先考虑从 W_1 中选取任务 τ_i,将其以更大的比例重新分配到 type-2 类型的处理器群上。用 Δy_i^1 表示在分配调整的过程中任务 τ_i 在 2 种类型处理器上的分配比例变化,即重新分配后有

$$y_i^1 = y_i^1 - \Delta y_i^1 \tag{14}$$

$$y_i^2 = y_i^2 + \Delta y_i^1 \tag{15}$$

设 ΔU_2 为 S_{UR2} 在该过程中的减小量,ΔE 为系统总能耗 E 在该过程中的减小量,则由式(2)可得

$$\Delta U_2 = \Delta u_i^2 = \Delta y_i^1 \times u_i^{2,\max} \tag{16}$$

由式(11)可得

$$\Delta E = L \times \Delta y_i^1 \times C_i \times \left(\frac{p_i^1}{r_i^1} - \frac{p_i^2}{r_i^2} \right) \times \frac{1}{T_i} \tag{17}$$

因此,由式(16)以及式(17)可得

$$\frac{\Delta E}{\Delta U_2} = L \times r_i^2 \times \left(\frac{p_i^1}{r_i^1} - \frac{p_i^2}{r_i^2} \right) \tag{18}$$

在不考虑能耗的任务分配方案下,S_{UR2} 为定值,所以为最大限度地减小系统的总能耗,从 W_1 中选取任务的规则就是每次选取对应 $\Delta E/\Delta U_2$ 值即 $r_i^2 \times \left(\frac{p_i^1}{r_i^1} - \frac{p_i^2}{r_i^2} \right)$ 值最大的任务,以一定的比例 $k_i\left(k_i \leqslant y_i^1\right)$ 重新调整到 type-2 类型处理器核上;从 W_2 中选取任务的方法同理。

2.5　分配方法

对于 W_1 和 W_2 中元素存在与否,分为以下 4 种情况:

1) W_1 和 W_2 均为空:表示已经不存在待调整的任务,分配方案结束;

2) W_1 非空,W_2 为空:选取 W_1 中的第一个任务 τ_i,以比例 k_i 重新调整到 type-2 类型处理器核上;

3) W_1 非空,W_2 为空:与 2)同理;

4) W_1 和 W_2 均非空。

选取 W_1 中的第一个任务 τ_i,以比例 k_i 重新调整到 type-2 类型处理器核上;选取 W_2 中的第一个任务 τ_j,以比例 k_j 重新调整到 type-1 类型处理器核上。为使 S_{UR1} 和 S_{UR2} 都非负,需要保证满足以下几个条件:

$$\mathrm{CC}:\begin{cases} u_j^{1,\max}\times k_j - u_i^{1,\max}\times k_i \leqslant S_{\mathrm{UR1}} \\ u_i^{2,\max}\times k_i - u_j^{2,\max}\times k_j \leqslant S_{\mathrm{UR2}} \\ 0\leqslant k_i \leqslant y_i^1 \\ 0\leqslant k_j \leqslant y_j^2 \end{cases} \tag{19}$$

在此过程中,系统总能耗的减小量 $\triangle E$ 为

$$\triangle E = L\times k_i\times\left(\frac{p_i^1}{r_i^1}-\frac{p_i^2}{r_i^2}\right)\times\frac{C_i}{T_i}+ L\times k_j\times\left(\frac{p_j^2}{r_j^2}-\frac{p_j^1}{r_j^1}\right)\times\frac{C_j}{T_j} \tag{20}$$

为在满足约束条件 CC 的情况下使得 $\triangle E$ 最大,利用二元线性规划,易求得 k_i 和 k_j,根据 k_i 和 k_j 值做相应的分配比例调整。

3　任务调度算法 BI-Wrap

为保证在 EB-Split 算法下所有的任务都满足截止期需求且不违背 NPE 原则,还需设计一种合适的任务调度方法,决定在某一时刻哪个任务具体在哪个处理器核上执行。

3.1　时间片划分

LEVIN 和 FUNK 证明了对于一个多核处理器上的任务调度问题,如果参与调度的任务具有 2 个或者 2 个以上的不同的截止期,那么就无法找到合适的任务调度方案。而如果任务集中的所有任务都具有相同的截止期,那么就可以找到可行的任务调度方案[13]。

因此,在二型异构多核平台上,要保证任务集中的所有任务都具有相同的截止期,可采用时间片划分[14-16]的思想,将整个时间轴划分为多个时间片,每个时间片分配相应的任务工作量,即同一个时间片内部所要调度的任务均共享同一个截止期。

3.2　时间片划分方法

每个任务 τ_i 在执行过程中会产生多个周期性的作业,每个作业的释放周期为相对截止时间 T_i,故任务 τ_i 所释放的第 k 个作业的绝对截止时间为 $k\times T_i$。

将所有任务 $\tau_i\in\tau$ 的所有绝对截止时间呈现于同一时间轴上,按照绝对时间的先后顺序定义为时间节点 $t_j(j=0,1,\cdots,j)$,其中,$t_0=0$ 表示时间轴的起始节点,也是任务调度执行的开始时间。用 σ_j 表示时间片划分完成后的第 j 个时间片,用 L_j 表示第 j 个时间片的长度,则时间片划分的结果为

$$\sigma_j=\left[t_{j-1},t_j\right),L_j=t_j-t_{j-1} \tag{21}$$

3.3　时间片上的任务工作量安排

在长度为 L_j 的时间片 σ_j 内,根据等比例划分的方法,任务 τ_i 在 type-1 类型处理器上的工作量安排为

$$\frac{x_i^1\times C_i}{T_i}\times L_j = u_i^1\times L_j\times r_i^1 \tag{22}$$

在 type-2 类型处理器上的工作量安排与 type-1 同理。

3.4　BI-Wrap 调度算法

BI-Wrap 调度算法首先将给定的任务集划分为 4 个子集:E、M、SP_1、SP_2。其中,集合 E 囊括同时分配到两类处理器群上($x_i^1>0$ 且 $x_i^2>0$)且满足 $u_i^1+u_i^2=1$ 的任务;集合 M 包含同时分配到两类处理器群上($x_i^1>0$ 且 $x_i^2>0$)而且满足 $u_i^1+u_i^2<1$ 的任务;集合 SP_1 中是仅分配到 type-1 类型处理器群上($x_i^1=1,x_i^2=0$)的任务;集合 SP_2 中为仅分配到 type-2 类型处理器群上($x_i^1=0,x_i^2=1$)的任务。

对于那些同时分配到 type-1 及 type-2 类型处理器群上($x_i^1>0$ 且 $x_i^2>0$)的任务,由文献[16]可知,$u_i^1+u_i^2=1$ 成立当且仅当 $x_i^k=lo_i^k$ 满足,因此,其中至多存在 2 个任务满足 $u_i^1+u_i^2<1$,其他任务都满足 $u_i^1+u_i^2=1$,故集合 M 中至多存在 2 个任务,可用

τ_a、τ_b表示。

定义 4 个队列：P_1、R_1、P_2、R_2，将集合$\{E, M, SP_1, SP_2\}$中的任务按照下列规则添加到这 4 个队列中：

$$P_1 = (\text{all tasks in } E, \tau_a, \text{all tasks in } SP_1) \quad (23)$$

$$R_1 = (\tau_b) \quad (24)$$

$$R_2 = (\text{all tasks in } E, \tau_a, \text{all tasks in } SP_2) \quad (25)$$

$$P_2 = (\tau_b) \quad (26)$$

在 type-1 类型处理器群以及 type-2 类型处理器群上分别对分配好的任务进行调度，调度规则如下：

1）在 type-1 类型的处理器上，从 core 1 开始，按照处理器核编号递增且时间递增的方式对队列 P_1 中的任务依次进行调度。

2）在 type-1 类型的处理器上，从 core m_1 开始，按照处理器核编号递减且时间递减的方式对队列 R_1 中的任务依次排列在时间片上，调度时按时间递增方向进行。

3）在 type-2 类型的处理器上，从 core 1 开始，按照处理器核编号递增且时间递增的方式对队列 P_2 中的任务依次进行调度。

4）在 type-2 类型的处理器上，从 core m_2 开始，按照处理器核编号递减且时间递减的方式对队列 R_2 中的任务依次排列在时间片上，调度时按时间递增方向进行。

3.5 调度举例

任务集的划分结果为

$$E = \{\tau_5, \tau_7\} \quad (27)$$

$$M = \{\tau_3, \tau_6\} \quad (28)$$

$$SP_1 = \{\tau_2, \tau_4, \tau_1\} \quad (29)$$

$$SP_2 = \varnothing \quad (30)$$

由调度规则可得

$$P_1 = (\tau_5, \tau_7, \tau_3, \tau_2, \tau_4, \tau_1) \quad (31)$$

$$R_1 = (\tau_6) \quad (32)$$

$$R_2 = (\tau_5, \tau_7, \tau_3) \quad (33)$$

$$P_2 = (\tau_6) \quad (34)$$

应用 BI-Wrap 算法中的调度规则在所划分的第 1 个时间片($\sigma_1 = [0, 60)$, $L_1 = 60$)上对上述 4 个队列中的任务实例进行调度，调度结果如图 1 所示。

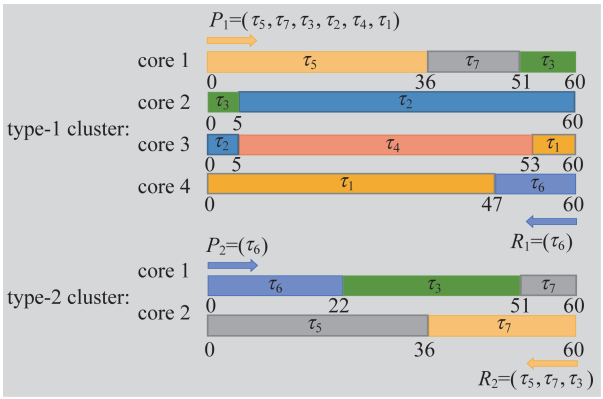

图 1 BI-Wrap 调度算法实例

Fig.1 Examples of the BI-Wrap schedule algorithm

如图 1 所示，采用 BI-Wrap 算法进行任务调度可保证在第 1 个时间片内分配的任务工作量都能在时间片截止之前完成，且所有任务在调度过程中都满足非并行执行性。

3.6 BI-Wrap 算法的时间复杂度

BI-Wrap 算法首先将整个任务集划分为 4 个子集，这一过程的时间复杂度为 $O(n)$，然后，BI-Wrap 算法对这 4 个子集中的任务均按照既定的规则进行顺序调度，这一过程的时间复杂度也为 $O(n)$。因此，整个 BI-Wrap 算法的时间复杂度为 $O(n)$。

4 仿真试验及结果分析

4.1 仿真实验

在 Windows 平台下用 C++语言模拟算法的调度过程，对 SA 算法、Hetero-Split 算法及 EB-Split 算法(使用 BI-Wrap 算法进行调度)分别进行了模拟调度。

对算法的输入数据，设置处理器集群(m_1, m_2)的个数平均至少大于 2，任务集 τ 中的每个任务(τ_i)在不同处理器集群上的参数元组(C_i, T_i, r_i^1, r_i^2, p_i^1, p_i^2)值均由随机函数生成。实验每次测试的任务集数目为 10 000。根据分析，实验将首先执行算法的负载分配以追求低能耗的目标，然后，按照 BI-Wrap 算法保证分配好的负载能够进行调度。

首先测试得到 3 种算法各自满足系统约束的可行任务调度方案数见表 1。

表 1　各算法所求的可行方案数

Tab.1　Numbers of feasible schemes for each algorithm

测试组数	SA 可行方案数/个	Hetero-Split 可行方案数/个	EB-Split 可行方案数/个
10 000	8 755	9 098	9 098
10 000	8 701	9 026	9 026
10 000	8 684	9 018	9 018
10 000	8 720	9 053	9 053
10 000	8 687	9 057	9 057

由实验结果可知，EB-Split 算法不损失 Hetero-Split 算法求得的可行方案数，且两者均优于 SA 算法。然后对比 EB-Split 算法（使用 BI-Wrap 算法进行调度）与 Hetero-Split 算法（正常调度）的平均系统能耗（分别用 E1_avg 和 E2_avg 表示），每次测试得到的系统能耗按式（11）计算而得。并计算 EB-Split 算法相对于 Hetero-Split 算法的能耗降低百分比 $R=(E1\_avg-E2\_avg)/E1\_avg$。实验结果如表 2、图 2 所示。

表 2　Hetero-Split 与 EB-Split 算法的能耗比

Tab.2　Energy consumption ratios of the Hetero-Split and EB-Split algorithms

测试组数	Hetero-Split 平均能耗/J	EB-Split 平均能耗/J	能耗降低比例/%
10 000	129.00	99.08	23.20
10 000	130.01	99.63	23.37
10 000	130.63	100.54	23.04
10 000	129.51	99.51	23.16
10 000	129.07	99.33	23.05

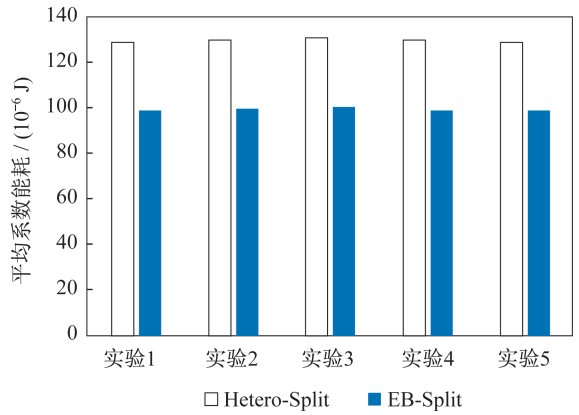

图 2　Hetero-Split 与 EB-Split 的能耗对比

Fig.2　Energy consumption of the Hetero-Split and EB-Split algorithms

由实验结果可知，EB-Split 算法相较于 Hetero-Split 算法在系统总能耗方面约降低 23%～24%。

4.2　仿真结果分析

由实验结果可知，EB-Split 算法与 Hetero-Split 算法在寻求可行性方案上性能相当且优于 SA 算法。Hetero-Split 算法已被证明为可行方案数最优的[10]，故本文所设计的 EB-Split 算法也满足可行任一方案的最优性。此外，在能耗方面，EB-Split 算法相较于 Hetero-Split 算法在系统总能耗方面有大幅度降低，降低比率约为 23%～24%。

5　结束语

异构多核处理器上实时任务调度算法的设计向来是难点问题，尤其在航天探测等对系统各方面性能指标要求都比较高的尖端领域。本文设计的基于全迁移的、加入能耗约束的实时任务调度算法，在保证系统实时性的条件下对能耗进行了优化，在总体性能方面优于目前的较出色的 SA 算法和 Hetero-Split 算法。

由于本文未考虑并行、迁移开销、任务相关性等方面的问题，因此，还有许多工作需进一步研究，且该任务调度算法目前虽然在能耗方面有较好的性能，但是并没有达到在满足实时约束条件下的最佳能耗。

参考文献

[1] CATALÁN S, RODRÍGUEZ-SÁNCHEZ R, QUINTANA-ORTÍ E S, et al. Static versus dynamic task scheduling of the lu factorization on ARM big. LITTLE architectures [C]// IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). 2017: 733-742.

[2] LI X F, CHEN G K, WEN W. Energy-efficient execution for repetitive app usages on big. LITTLE architectures [C]// ACM Proceedings of the 54th Annual Design Automation Conference. 2017: 1-6.

[3] KIM M, KIM K, GERACI J R, et al. Utilization-aware load balancing for the energy efficient operation of the big.LITTLE processor [C]// IEEE Proceedings of the Conference on Design, Automation & Test in Europe. 2014: 1-4.

[4] 石祥龙. 基于异构多核处理器的静态任务调度算法研究[D]. 南京：南京邮电大学，2015.

[5] ANDERSSON B, RARAVI G. Scheduling constrained-deadline parallel tasks on two-type heterogeneous multiprocessors [C]// ACM Proceedings of the 24th International Conference on Real-Time Networks and Systems. 2016: 247-256.

[6] RARAVI G, ANDERSSON B, BLETSAS K. Assigning real-time tasks on heterogeneous multiprocessors with two unrelated types of processors [J]. Real-Time Systems, 2013, 49(1): 29-72.

[7] HAN J J, GONG S L, WANG Z J, et al. Blocking-aware partitioned real-time scheduling for uniform heterogeneous multicore platforms [J]. ACM Transactions on Embedded Computing Systems, 2020, 19(1):1-25.

[8] RARAVI G, ANDERSSON B, NELIS V, et al. Task assignment algorithms for two-type heterogeneous multiprocessors [J]. Multiprocessor and Mixed-Criticality Scheduling, 2014, 50(1): 87-141.

[9] BARUAH S. Feasibility analysis of preemptive real-time systems upon heterogeneous multiprocessor platforms [C]// IEEE International Real-Time Systems Symposium. 2004: 37-46.

[10] CHWA H S, SEO J, LEE J, et al. Optimal real-time scheduling on two-type heterogeneous multicore platforms [C]// IEEE Real-Time Systems Symposium. 2015: 119-129.

[11] 李延祺,任海,白亮,等.一种面向节能的星载实时任务动态调度算法研究[J].上海航天,2019,36(3):82-89.

[12] 杨亚琪,栾钟治,杨海龙,等.异构多核下兼顾应用公平性和能耗的调度方法研究[J].计算机工程与科学,2016,38(5):848-856.

[13] LEVIN G, FUNK S, SADOWSKI C, et al. DP-FAIR: a simple model for understanding optimal multiprocessor scheduling [C]// IEEE Euro micro Conference on Real-Time Systems. 2010: 3-13.

[14] ZHU D K, MOSSE D, MELHEM R G. Multiple-resource periodic scheduling problem: how much fairness is necessary [C]// IEEE International Real-Time Systems Symposium (RTSS). 2003: 142-151.

[15] ANDERSSON B, BLETSAS K. Sporadic multiprocessor scheduling with few preemptions [C]// IEEE Euro micro Conference on Real-Time Systems. 2008: 243-252.

[16] UMARANI S G, GEETHA R. Task scheduling using ant colony optimization in multicore architectures: a survey [J]. Soft Computing, 2018, 22(15): 5179-5196.

(上接第 18 页)

[12] BERGER R, CHADWICK S, CHAN E, et al. Quad-core radiation-hardened system-on-chip power architecture processor [C]// IEEE Aerospace Conference. 2015:1-12.

[13] DOYLE R J. Reinventing the role of computing in space [EB/OL]. (2019-07-14) [2021-02-21]. https://www.caltech.edu/research/jpl.

[14] WESLEY P GABRIEL M, MONTGOMERY G, et al. High-performance spaceflight computing (HPSC) program overview [C]// NASA Space Computing & Connected Enterprise Resiliency Conference (SCCERC). 2018: 20180003537.

[15] Defense media network. Defense advanced research projects agency 1958-2018 [EB/OL]. (2018-10-26) [2021-02-21]. https://www. defensemedianetwork. com/ blog/ darpa-defense-advanced-research-projects-agency-1958-2018/.

[16] 康小录,张岩.空间电推进技术应用现状与发展趋势[J].上海航天,2019,36(6):28-38.

[17] 陈子健,秦伟,吴新科.基于GaN HEMT和平面变压器的高效率谐振DC/DC变换器[J].上海航天,2020,37(2):139-145.