一种能耗约束的多核系统任务调度算法

谢 盈[1,2*]，陈建英[1,2]，吴尽昭[3]，丁旭阳[4]

(1. 西南民族大学计算机系统国家民委重点实验室 成都 610041；2. 西南民族大学计算机科学与技术学院 成都 610041；
3. 广西民族大学广西混杂计算与集成电路设计分析重点实验室 南宁 530006；
4. 电子科技大学计算机科学与工程学院 成都 611731)

【摘要】不同的任务调度算法将任务分配在不同的处理单元，会产生不同的能耗。在基于片上网络的多核系统中，将任务分摊到所有处理器核能提高系统利用率，但导致大量的簇间通信，增加数据传输能耗。本文在对系统能耗进行建模的基础上，提出一种能耗约束的任务调度算法。该算法结合任务间依赖关系，动态计算任务分配时产生通信开销最小的簇和系统利用率最大的簇，通过计算在相应簇上产生的系统能耗，选择产生系统能耗小的分配方案以平衡系统利用率和簇间通信开销。仿真实验结果证明，算法在减少了簇间通信开销并提高系统利用率的同时，降低了系统能耗。

关 键 词 通信开销； 能耗约束； 片上网络； 系统利用率
中图分类号 TP301 **文献标志码** A doi:10.3969/j.issn.1001-0548.2019.02.014

A Energy-Constraint Task Scheduling Algorithm on Multi-Processor System

XIE Ying[1,2*], CHEN Jian-ying[1,2], WU Jin-zhao[3], and DING Xu-yang[4]

(1. The Key Laboratory for Computer Systems of State Ethnic Affairs Commission, Southwest Minzu University Chengdu 610041;
2. School of Computer Science and Technology, Southwest Minzu University Chengdu 610041;
3. Guangxi Key Laboratory of Hybrid Computational and IC Design Analysis, Guangxi University for Nationalities Nanning 530006;
4. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 611731)

Abstract Different task scheduling algorithms assign tasks to different processors and result in different energy consumptions. In NoC-based (network on chip-based) multi-processor systems, distributing tasks to all processor elements could improve system utilization, but causing a large amount of inter-cluster communication and increasing data transmission energy consumption. Based on modeling the energy consumption model, this paper proposes an energy constraints task scheduling algorithm. Combining with the dependencies between tasks, the proposed algorithm dynamically calculates the cluster with minimal communication overhead and the cluster with maximal system utilization. Though calculating the energy consumption of corresponding clusters, the proposed algorithm chooses the plan which generates smaller energy consumption in order to balance system utilization and inter-cluster communication overhead. Simulation results show that the proposed algorithm can reduce the energy consumption, lower the inter-cluster communication overhead and increase the system utilization.

Key words communication overhead; energy-constraint; NoC; system utilization

随着片上处理单元数量不断增加，片上网络(network on chips, NoC)所占芯片面积的比例不断提高，在提供高扩展性、高带宽、低延迟等性能的同时限制了多核系统性能的进一步提高[1-2]。特别是针对通信密集型应用和在电池供电的设备中，NoC带来的能耗开销已成为系统能耗的重要组成部分。

合适的任务调度算法能显著降低基于NoC的多核系统的整体能耗，并在一定程度上提高系统的利用率。在基于NoC的多核系统任务调度算法中，文献[3]提出了一种位置感知的实时任务调度算法，利用处理器核的物理位置将任务调度在相邻的处理器核中以降低核间的通信距离，从而达到节能的目的，但是该算法没

收稿日期：2017 – 12 – 29；修回日期：2018 – 06 –19
基金项目：国家社科基金重大项目(17ZDA160)；国家自然科学基金(11461006, 61772006)；广西科技计划(AB17129012)；广西科技重大专项(AA17204096)；广西科技基地和人才专项(2016AD05050)
作者简介：谢盈(1984 –)，女，博士生，主要从事嵌入式系统与形式化方法的研究. E-mail: xieying33@163.com

有考虑在能耗约束下的系统利用率问题。文献[4]针对软实时任务提出了一种能耗约束的任务调度算法，利用动态电压/频率调节技术(dynamic voltage frequency scaling, DVFS)实现软实时任务的节能调度，但没有考虑数据在NoC延迟及传输过程中产生的能耗。文献[5]提出了一个多核处理器时间/事件触发通信系统模型，并将该模型映射为混合整数线性规划问题，用于将任务分配给处理单元。模型将任务间的依赖关系、片上路由间的有限连通性和系统能耗作为约束条件，在给定的多核架构上优化应用任务的计算和通信延迟。文献[6]提出一种非线性规划算法，为采用DVFS技术的多核系统提供一种连续频率模型，用于为每个任务分配最优的频率和通信链路以降低总能耗，但算法只关注独立任务，没有考虑依赖任务的分配。文献[7]结合动态功耗管理(dynamic power management, DPM)技术和DVFS技术，以系统能耗为约束对系统任务进行调度，该方法不基于任何先验知识，在任务到达后再获取任务具体属性，没有充分考虑任务间的依赖关系及系统利用率。

本文针对基于NoC的多核系统提出一种能耗约束的任务调度算法ECTSA(energy-constraint task scheduling algorithm on NoC-based multi-processor system)。该算法在对系统能耗进行建模的基础上，结合任务间前驱后续关系，动态计算任务分配时产生通信开销最小的簇和系统利用率最大的簇，引入能耗作为约束因子，通过计算在相应簇上产生的系统能耗，选择产生系统能耗小的分配方案在通信开销最小化和系统利用率最大化间进行适应性选择。

1 系统模型

1.1 目标平台模型

目标平台采用 $M\times N\times C$ 的NoC结构。其中，路由器和链路构成的网络规模为 $M\times N$，每个路由器通过网络接口连接一个含有 p 个处理单元的簇，$0<p\leqslant C$。一个 $3\times3\times2$ 的NoC平台如图1所示。

$M\times N\times C$ 的NoC目标平台可表示为：

$$\text{Cluster}=\{\text{Clu}^1,\text{Clu}^2,\cdots,\text{Clu}^{M\times N}\}$$

式中，$\text{Clu}^i=(S^i,P^i,Q^i)$ 表示第i个簇，簇 Clu^i 的计算能力抽象为 S^i，$P^i=\{p_1^i,p_2^i,\cdots,p_{c_i}^i\}$ 表示簇 Clu^i 中的 c_i 个处理单元集合，p_j^i 表示簇 Clu^i 中的第 j 个处理单元，目标平台处理单元集合为 $\bigcup_{i=1}^{M\times N}\bigcup_{j=1}^{c_i}p_j^i$，$Q^i=\{q_1^i,q_2^i,\cdots,q_{c_i}^i\}$ 是簇 Clu^i 中各处理单元对应的等待队列的集合，q_j^i 表示处理单元 p_j^i 上排队等待执行的任务队列，$|q_j^i|$ 为队列长度。

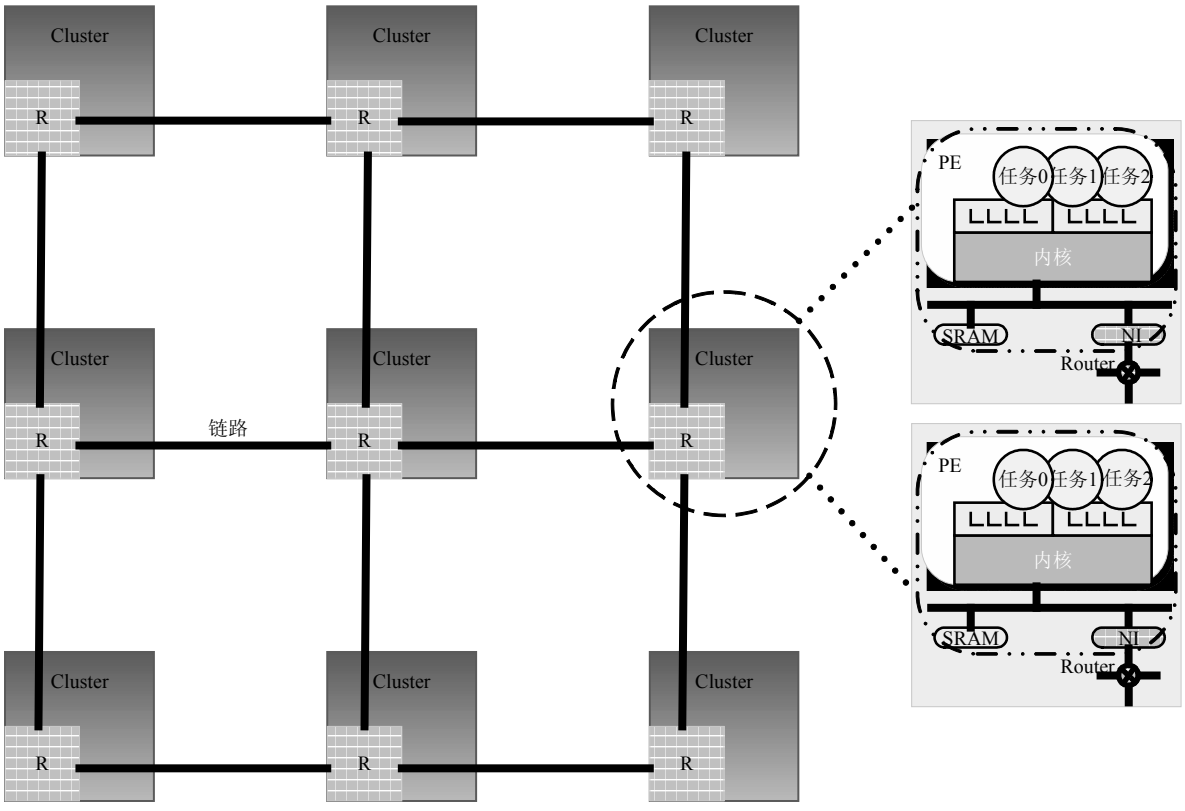

图1 3×3×2 的NoC的平台示意图

1.2 任务模型

并行依赖任务集(parallel dependent tasks, PDT)由二元组 $\mathrm{PDT}=(T,E)$ 描述的DAG图表示。其中 $T=\{T_0,T_1,\cdots,T_n\}$ 表示任务节点集合，T_0 为虚拟起始节点；$E=\{e_{i,j}\mid T_i\to T_j,i\leqslant n,j\leqslant n\}$ 是节点间的有约束关系的边集合，$e_{i,j}$ 表示任务 T_i 是任务 T_j 的直接前驱，任务 T_j 需要等待其所有前驱任务都运行结束后才能开始运行。$\mathrm{Calc}(T_i)$ 表示任务 T_i 的计算量，$\mathrm{Comm}(e_{i,j})$ 表示边 $e_{i,j}$ 上的通信数据量。$\mathrm{pred}(T_i)$ 表示任务 T_i 的直接前驱结点集。具有前驱后续关系的任务序列称为路径，路径的权重 W 定义为路径上所有任务节点的计算量和节点间边的通信数据量之和。关键路径 $\mathrm{CP}_{i,j}$ 是任务节点 T_i 到 T_j 的所有路径中权重最大的路径。

1.3 能耗模型

$M\times N\times C$ 的NoC目标平台中，每个簇上采用独立的DVFS技术动态调节处理单元供电电压/时钟频率。大多数采用DVFS技术的处理器功耗近似表示为静态功耗和动态功耗之和：

$$\mathrm{Power}=P_{\mathrm{dynamic}}+P_{\mathrm{static}} \tag{1}$$

式中，静态功耗由漏电电流造成，跟电路逻辑器件的出厂状态有关，是电路稳定状态下的功耗，可设为常量 θ。动态功耗是电路充放电引起的功耗，由电路的负载电容 C_L、供电电压 V 和时钟频率 f 决定[8]：

$$P_{\mathrm{dynamic}}=C_LV^2f \tag{2}$$

当前待调度任务 T_i 被分配到簇 Clu^k 中产生的计算能耗 $E_{i,k}^{(\mathrm{calc})}$ 表示为：

$$\begin{aligned}E_{i,k}^{(\mathrm{calc})}&=\int_0^{t_i}P_{\mathrm{dynamic}}^{(\mathrm{Clu}^k)}\mathrm{d}t_i+\theta^{(\mathrm{Clu}^k)}t_i=\\&\int_0^{t_i}(C_L^{(\mathrm{Clu}^k)}V^{(\mathrm{Clu}^k)2}f^{(\mathrm{Clu}^k)})\,\mathrm{d}t_i+\theta^{(\mathrm{Clu}^k)}t_i\end{aligned} \tag{3}$$

式中，$t_i=\mathrm{Calc}(T_i)/S^k$ 是簇 Clu^k 处理单元 p_j^k 执行任务 T_i 所需时间；$C_L^{(\mathrm{Clu}^k)}$，$V^{(\mathrm{Clu}^k)}$，$f^{(\mathrm{Clu}^k)}$ 是簇 Clu^k 处理单元的负载电容、供电电压和时钟频率。

虫孔交换机制能大大降低了报文的传输延迟，是NoC中最常用的路由交换方法[9]。在虫孔机制中，一个微片flit从簇 Clu^{k1} 到簇 Clu^{k2} 的传输过程中产生的能耗 $E_{k1,k2}^{(\mathrm{flit})}$ 包括传输路径上所有路由器产生的能耗和传输链路上产生的能耗：

$$\begin{cases}E_{\mathrm{router}}=\displaystyle\int_0^{t^{(r)}}(C_L^{(r)}V^{(r)2}f^{(r)})\,\mathrm{d}t^{(r)}+\theta^{(r)}t^{(r)}\\E_{\mathrm{link}}=\displaystyle\int_0^{t^{(l)}}(C_L^{(l)}V^{(l)2}f^{(l)})\,\mathrm{d}t^{(l)}+\theta^{(l)}t^{(l)}\\E_{k1,k2}^{(\mathrm{flit})}=(M_{k1,k2}+1)E_{\mathrm{router}}+M_{k1,k2}E_{\mathrm{link}}\end{cases} \tag{4}$$

式中，$M_{k1,k2}$ 是簇 Clu^{k1} 到簇 Clu^{k2} 的曼哈顿距离；$C_L^{(r)}$、$V^{(r)}$、$f^{(r)}$ 及 $t^{(r)}$ 是路由器的负载电容、供电电压、频率及一个flit在路由器上的传输时间；$C_L^{(l)}$、$V^{(l)}$、$f^{(l)}$ 及 $t^{(l)}$ 是两个路由器之间的链路的负载电容、供电电压、频率及一个flit在两个路由器之间的链路上的传输时间。

任务以指数分布延迟到达主节点的任务队列。主节点按照任务调度算法将任务分配到合适的计算节点。令本文主节点位于簇 $\mathrm{Clu}^{k'}$。将任务 T_i 分配到簇 Clu^k 上产生的通信数据量为 $\mathrm{Comm}_i^{\mathrm{Clu}^k}$。令一个flit的位宽为 B，则将任务 T_i 从主节点所在簇 $\mathrm{Clu}^{k'}$ 分配到簇 Clu^k 产生的传输能耗 $E_{i,k}^{(\mathrm{comm})}$ 为：

$$E_{i,k}^{(\mathrm{comm})}=\frac{\mathrm{Comm}_i^{\mathrm{Clu}^k}}{B}E_{k',k}^{(\mathrm{flit})} \tag{5}$$

式中，$\mathrm{Comm}_i^{\mathrm{Clu}^k}\in\{\mathrm{Comm}(e_{j,i})\mid e_{j,i}\in E,T_j\in\mathrm{pred}(T_i)\}$。

根据以上分析，系统能耗包括处理单元产生的计算能耗和数据传输过程中路由器和链路产生的传输能耗。由式(3)和式(5)，将当前待调度任务 T_i 分配到簇 Clu^k 中产生的总能耗为：

$$E_{i,k}=E_{i,k}^{(\mathrm{calc})}+E_{i,k}^{(\mathrm{comm})} \tag{6}$$

2 ECTSA算法

从减小簇间通信开销的角度出发，将所有任务都调度到同一个簇中，簇间通信开销达到最小；从提高系统利用率的角度出发，将任务分摊到多个处理单元，让每个处理器单元都有充分的任务执行以加强系统的数据处理能力，从而提高系统利用率。对于依赖任务而言，最小化簇间通信开销和最大化系统利用率是反相关的两个目标，ECTSA算法引入能耗作为决策因子，通过动态计算任务分配在相应簇上产生的系统能耗，选择产生系统能耗小的分配方案以平衡系统利用率和簇间通信开销。

2.1 最小化通信开销

在任务调度过程中，已调度的任务和未调度的任务都会和当前待调度任务产生数据交互关系。在图2所示的PDT中，若当前待调度任务为 T_6，则PDT被 T_6 划分为3个子集P、F和O。其中子集P为已经调度了的任务，子集F为 T_6 及 T_6 的直接和间接后续任务，剩余的任务构成子集O。

T_i 为当前待调度任务，从考虑最小化簇间通信开销的角度出发，T_i 应该被划分到和其具有最大通信数据量的直接前驱 T_r 所在的簇。子集F中的任务为 T_i 的后续任务，故 T_i 的调度结果会直接影响子集F

中的所有任务。子集O中的任务都是未调度任务，且不会受到T_i调度结果的影响，因此在对T_i进行调度时不考虑子集O中的任务。通过以上分析，在对当前待调度任务T_i进行调度时，最小化当前簇间通信开销的问题就转化为最小化当前子集P与子集F的簇间通信开销。

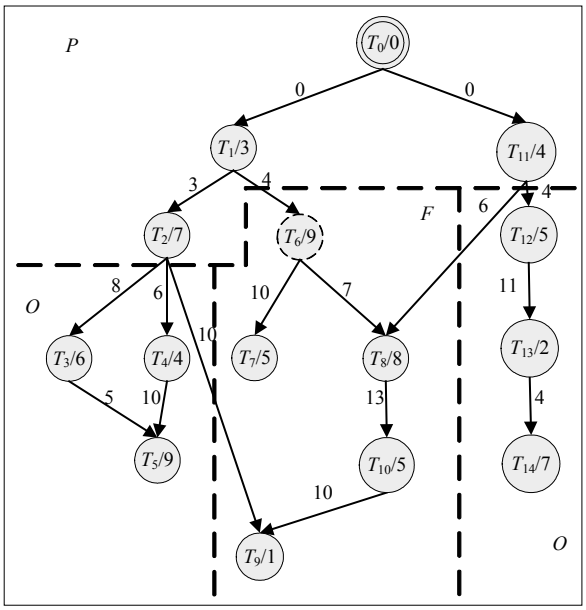

图2 并行依赖任务集PDT的划分

为了量化子集P与子集F的簇间通信开销，ECTSA算法为每个任务T_i引入通信负载数组$\mathrm{CommLoad}_i[R]$，$R=M\times N$为目标结构中簇的数目。其中，分量$\mathrm{CommLoad}_i[k]$表示当前待分配任务T_i对应的子集P到子集F的所有数据交互中，在簇Clu^k上产生的数据交互量：

$$\mathrm{CommLoad}_i[k]=\sum_{e_{r,s}}\delta\max(W(\mathrm{CP}_{r,s}),W(\mathrm{CP}_{i,s})) \quad (7)$$

式中，$T_r\in P$；$T_s\in F$；$e_{r,s}$表示任务T_i对应子集P到子集F的所有数据交互，若T_r被调度在簇Clu^k上，则$\delta=1$，否则$\delta=0$。

2.2 最大化系统利用率

算法通过将任务分配给目标平台中利用率最低的簇执行从而最大化系统利用率。为了量化簇利用率，对当前待调度任务T_i，ECTSA算法引入计算负载数组$\mathrm{CalcLoad}_i[R]$，$R=M\times N$为目标结构中簇的数目。其中，分量$\mathrm{CalcLoad}_i[k]$表示对于当前待分配任务T_i，簇Clu^k内所有处理单元等待队列上的任务的计算负载之和：

$$\mathrm{CalcLoad}_i[k]=\sum_{r=1}^{c_k}\sum_{s=1}^{|q_r^k|}\mathrm{Calc}(T_s) \quad (8)$$

式中，$T_s\in P$是已经调度到簇Clu^k内处理单元的任务；c_k是簇Clu^k中处理单元的个数；q_r^k是处理单元p_r^k所维护的等待队列。

2.3 能耗约束调度

图3给出了针对基于NoC的多核系统的能耗约束的任务调度算法ECTSA的总体流程图。

1) 由于任务间的数据依赖关系，任务要等到其所有直接前驱都执行结束后才能开始执行，因此算法采用可调度任务列表STL来存放当前所有可被调度的任务。

2) STL中的任务往往不止一个，算法优先调度在PDT中关键路径上的任务；当有多个关键路径上的任务时选择主优先级高的任务调度；否则，选择次优先级高的任务调度。

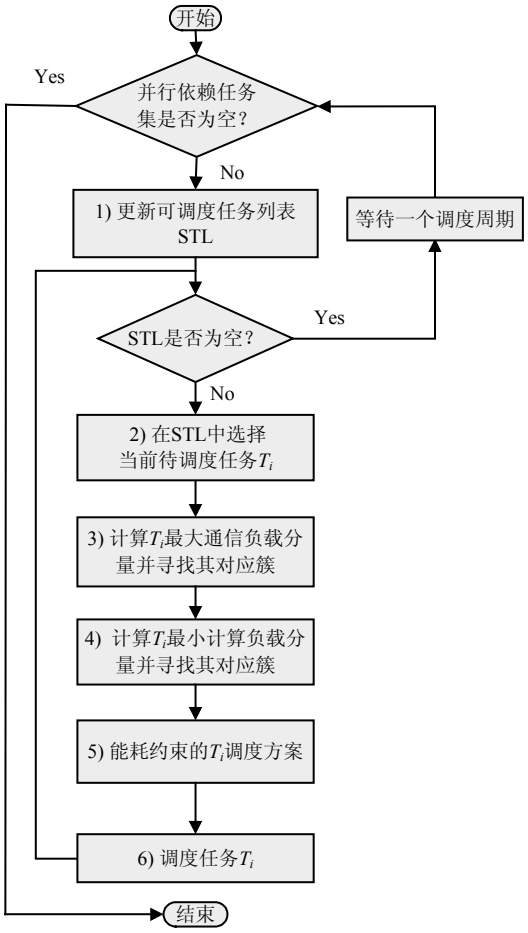

图3 ECTSA算法流程图

任务T_i的主优先级定义为：

$$\mathrm{MP}(T_i)=\begin{cases}1 & \mathrm{pred}(T_i)=\varnothing\\ \max\limits_{T_j\in\mathrm{pred}(T_i)}(\mathrm{MP}(T_j))+1 & \text{其他}\end{cases}$$

任务主优先级越大说明该任务离叶子节点的距离越远，后续任务对当前任务的数据依赖越大，应

优先调度。

任务 T_i 的次优先级为定义为最晚启动时间 $\mathrm{LST}(T_i)$ 和最早启动时间 $\mathrm{EST}(T_i)$ 之差的倒数：

$$\mathrm{SP}(T_i)=\frac{1}{\mathrm{LST}(T_i)-\mathrm{EST}(T_i)}$$

$$\mathrm{EST}(T_i)=\begin{cases}0 & \mathrm{pred}(T_i)=\varnothing\\ \max\limits_{T_j\in\mathrm{pred}(T_i)}\{\mathrm{ECT}(T_j)+\mathrm{Comm}(e_{j,i})/S^L\} & \text{其他}\end{cases}$$

$$\mathrm{ECT}(T_i)=\mathrm{EST}(T_i)+\mathrm{Calc}(T_i)/S^k\quad(\psi_{i,k}=1)$$

$$\mathrm{LST}(T_i)=\mathrm{LCT}(T_i)-\mathrm{Calc}(T_i)/S^k\quad(\psi_{i,k}=1)$$

式中，S^L 为链路通信速率；$\psi_{i,k}=1$ 表示任务 T_i 分配在簇 Clu^k 上；$\mathrm{ECT}(T_i)$ 和 $\mathrm{LCT}(T_i)$ 分别表示任务的最早完成时间和最晚完成时间。

最晚启动时间和最早启动时间越接近的任务，紧迫程度越大，该类任务对PDT调度长度的影响也越大，需要尽可能早地进行调度。

3) 利用当前待调度任务子集P、子集F间的数据交互计算当前待调度任务 T_i 通信负载数组 $\mathrm{CommLoad}_i[R]$，找出最大通信负载分量所对应的簇 Clu^{k1}。

4) 计算当前待调度任务 T_i 的计算负载数组 $\mathrm{CalcLoad}_i[R]$，找出最小计算负载分量所对应的簇 Clu^{k2}。

5) 计算任务 T_i 分配到簇 Clu^{k1} 产生的系统能耗 $E_{i,k1}$，分配到簇 Clu^{k2} 产生的系统能耗 $E_{i,k2}$，以能耗作为决策因子，选择产生较小系统能耗的分配方案，即选择 $\min(E_{i,k1},E_{i,k2})$ 所对应的簇。

6) 将任务 T_i 分配到 $\min(E_{i,k1},E_{i,k2})$ 所对应的簇上，若可调度任务列表STL不为空，转步骤2)，否则，等待一个调度周期后转步骤1)。

3 实　验

本节从系统利用率、簇间数据交互次数、系统能耗三个维度将ECTSA算法GEDF-OLEASA算法进行了对比。

仿真实验使用Noxim[10]模拟 $4\times4\times3$ 的NoC多核系统。任务集由TGFF[11]生成，TGFF能对包括任务数、任务计算开销、任务间依赖关系及任务间通信数据量等属性进行模拟。Noxim中的流量表由TGFF模拟的任务间通信数据量构成。

仿真实验中，片上网络采用 $4\times4\times3$ 的目标平台，片上网络频率为100 MHz，链路带宽为6 400 Mbs^{-1}，路由算法为确定性XY路由算法。TGFF[11]产生任务节点规模分别为25、50、75、100、125、150的并行依赖任务集PDT；处理单元工作电压 $V\in[1.5,2.5]$ V，阈值电压 $V_{\mathrm{th}}\in[0.3,0.6]$ V，静态功耗 $\theta\in[0.16,0.45]$ W。

图4为PDT规模为25、50、75、100、125、150时在同一目标平台下ECTSA算法和GEDF-OLEASA算法的簇间数据交互次数对比图。从图4可以看出，通过同时考虑已调度任务和会受到调度结果影响的未调度任务，ECTSA算法能够明显降低任务调度过程中的数据交互次数，降低簇间通信开销。

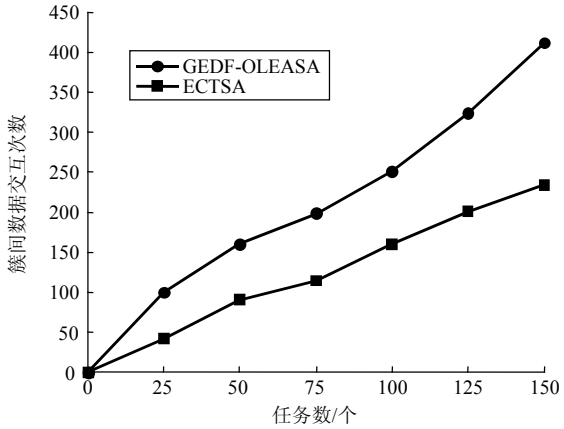

图4　不同任务数的簇间数据交互次数

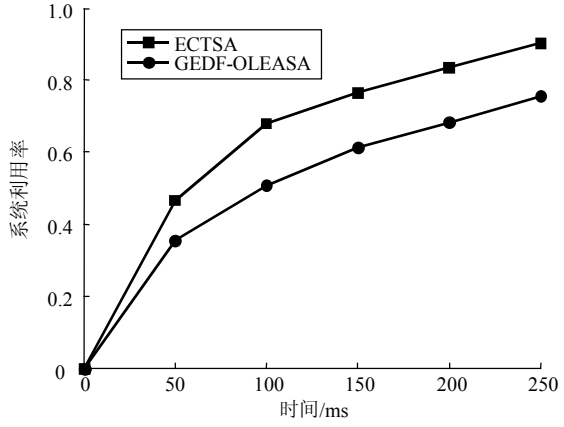

图5　系统利用率比较

图5为PDT规模为100时在同一目标平台下ECTSA算法和GEDF-OLEASA算法的系统利用率对比图。从图5可以看出ECTSA算法对应系统利用率相对提高，增强了系统的数据处理能力。也说明ECTSA算法通过将任务分配给利用率最低的簇以提高系统整体利用率是可行的。

为了对比不同算法所产生的系统能耗，引入任务平均调度长度(average scheduling length, ASL) 与任务最坏情况下调度长度(worst scheduling length, WSL)之比，刻画任务运行时属性。系统能耗以GEDF-OLEASA的能耗作为标准进行归一化处理。

图6为PDT规模为100个任务时，ECTSA算法和GEDF-OLEASA在不同ASL/WSL下系统能耗的比较。当ASL远低于WSL时，ECTSA算法比GEDF-OLEASA算法节能达到20%以上，随着ASL/WSL比值的增大，ECTSA算法的节能效果有所降低，但始终优于GEDF-OLEASA算法。

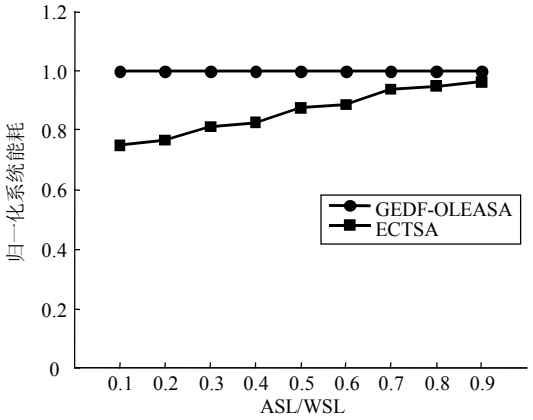

图6 不同ASL/WSL下系统能耗比较

4 结 束 语

本文提出了一种能耗约束的基于NoC的多核系统任务调度算法，该算法在对目标平台系统能耗进行建模的基础上，动态计算任务分配在最大通信负载分量所对应的簇上产生的系统能耗及任务分配在最小计算负载分量所对应的簇上产生的系统能耗，并选择产生系统能耗小的分配方案以平衡系统利用率和处理单元间通信开销。仿真实验结果证明，本文所提出的ECTSA算法相较于GEDF-OLEASA算法在减少了核间通信开销并提高系统利用率的同时，降低了系统能耗。

本文研究工作得到西南民族大学中央高校基本科研业务费(2015NZYQN28)的资助，在此表示感谢。

参 考 文 献

[1] SHACHAM A, BERGMAN K, CARLONI L P. Photonic networks-on-chip for future generations of chip multiprocessors[J]. IEEE Transactions on Computers, 2008, 57(9): 1246-1260.

[2] BENINI L, MICHELI G D. Networks on chips: a new SoC paradigm[J]. Computer, 2002, 35(1): 70-78.

[3] HU W, TANG X, XIE B, et al. An efficient power-aware optimization for task scheduling on NoC-based many-core system[C]//IEEE International Conference on Computer and Information Technology. [S.l.]: IEEE Computer Society, 2010: 171-178.

[4] BAUTISTA D, SAHUQUILLO J, HASSAN H, et al. A simple power-aware scheduling for multicore systems when running real-time applications[C]//2008 IEEE International Symposium on Parallel and Distributed Processing. [S.l.]: IEEE, 2008: 1-7.

[5] MURSHED A, OBERMAISSER R, AHMADIAN H, et al. Scheduling and allocation of time-triggered and event-triggered services for multi-core processors with networks-on-a-chip[C]//2015 IEEE 13th International Conference on Industrial Informatics (INDIN). [S.l.]: IEEE, 2015: 1424-1431.

[6] ISHAK S A, WU H, TARIQ U U. Energy-aware task scheduling on heterogeneous NoC-Based MPSoCs[C]// IEEE International Conference on Computer Design. [S.l.]: IEEE Computer Society, 2017: 165-168.

[7] 张冬松, 吴彤, 陈芳园, 等. 多核系统中基于Global EDF的在线节能实时调度算法[J]. 软件学报, 2012, 23(4): 996-1009.
ZHANG Dong-song, WU Tong, CHEN Fang-yuan, et al. Global EDF-based on-line energy-aware real-time scheduling algorithm in multi-core systems[J]. Journal of Software, 2012, 23(4):996-1009.

[8] KIM W, GUPTA M S, WEI G Y, et al. System level analysis of fast, per-core DVFS using on-chip switching regulators[C]//IEEE 14th Int'l Symp on High Performance Computer Architecture. Los Alamitos: IEEE, 2008: 123-134.

[9] HAN J J, LIN M, ZHU D, et al. Contention-aware energy management scheme for noc-based multicore real-time systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2015, 26(3): 691-701.

[10] Sourceforge. Noxim: Network-on-chip simulator[EB/OL]. [2017-03-09]. http://sourceforge.net/projects/noxim.

[11] DICK R P, RHODES D L, WOLF W. TGFF: Task graphs for free[C]//Proceedings of the 6th International Workshop on Hardware/Software Codesign. Seattle, USA: [s.n.], 1988: 97-101.

编 辑 蒋 晓