

面向 PIM 异构系统的 MTTF 感知的可靠性任务调度

庞德松, 梁华国

(合肥工业大学 电子科学与应用物理学院, 安徽 合肥 230009)

摘 要: 随着图形计算和数据分析等访存密集型应用的繁荣, 内存内计算(PIM)被认为是解决日益严重的“存储墙”最可行的解决方案. 这种 PIM 系统高可靠性(保证系统长时间高性能地正常运行)在大数据时代显得尤为重要. 然而, 现有的基于异构系统的优化系统老化的任务调度算法, 没有考虑 PIM+CPU 系统的硬件特性, 如访存竞争等, 如果直接应用在 PIM+CPU 系统中不能达到较好的优化效果. 因此, 我们考虑 PIM+CPU 异构系统的特性, 对该系统建立了一个老化可靠性模型, 并提出了一个基于 MTTF 的任务调度算法 PIM-MTTF 来平衡整个系统的 MTTF. 实验结果表明, 与现有的异构系统调度的研究工作相比, 该方法能够将 PIM+CPU 系统的平均 MTTF 差异值平均降低 9.1%, 系统性能平均提升 13.1%.

关键词: 异构系统; 可靠性; 任务调度; 内存内计算

中图分类号: TP306+.3

文献标识码: A

文章编号: 1000—7180(2018)12—0038—06

MTTF-Aware Reliability Task Scheduling for PIM-based Heterogeneous Computing System

PANG De-song, LIANG Hua-guo

(School of Electronic Science & Applied Physics, Hefei University of Technology, Hefei 230009, China)

Abstract: Processing-in-Memory (PIM) has been recognized as the most feasible solution to resolve the ever-aggravating memory wall especially as the boom of memory-intensive scale-out workloads such as graph computing and data analytics. The high reliability of this PIM system (ensuring that the system operates normally for long periods of high performance) is particularly important in the era of big data. However, existing aging-award task scheduling algorithms for heterogeneous systems do not consider hardware characteristics (memory interference etc.) in PIM+CPU system, if it is directly applied in the PIM + CPU system, a better optimization effect cannot be achieved. Therefore, we built an aging reliability model for PIM + CPU heterogeneous systems and proposed an MTTF-based task scheduling algorithm, PIM-MTTF, to balance the MTTF of the entire system. Experimental results show that, compared to the traditional scheduling algorithm for heterogeneous system, the proposed method is able to reduce MTTF variation over 9.1% on average and improvement of the system performance by 13.1% on average for PIM+CPU system.

Key words: heterogeneous system; reliability; task scheduling; PIM

1 引言

随着现代社会进入大数据时代, 各种数据密集型应用呈现井喷式发展, “存储墙”问题变得日益尖锐, 内存计算(Process in Memory, PIM)被认为是

有效的解决“存储墙”瓶颈的新型计算架构. PIM 将计算单元与存储单元紧密地耦合在一起, 从而消除访存带宽瓶颈的限制以及计算单元与存储单元的片间数据迁移所产生的开销, 具有极高的存储密度以及带宽(最大可达 320 GB/s). 这种 PIM 系统高可

靠性(保证系统长时间高性能地正常运行)在大数据时代显得尤为重要。

虽然研究异构系统的高可靠性的工作有很多,但是,考虑到新型的 PIM+CPU 系统所具有的硬件特性,如高带宽、低延迟等特性,的研究工作还没有。我们发现,直接将现有的任务调度方法应用在 PIM+CPU 系统,并不能达到最佳的效果。因此,我们为 PIM+CPU 异构系统建立了一个老化可靠性模型,并提出了一个基于 MTTF 的任务调度算法 PIM-MTTF 来平衡整个系统的 MTTF。

2 相关工作

随着集成电路特征尺寸越来越小,电路的老化问题对电路的可靠性产生越来越重要的影响。针对电路老化导致的异构系统的可靠性问题,目前已经有很多相关的学术著作提出方法来解决多核异构平台的系统可靠性问题。

为了提高整个异构系统的可靠性,一些研究利用热感知任务调度算法来限制核心之间的峰值温度或热梯度。文献[1]针对多处理器系统中泄漏功率与温度的关系,提出了一种功耗约束技术的热感知任务调度算法以降低功耗;还有利用 NBTI 感知算法来平衡多核处理器平台的工作负载,以及利用复制任务调度算法来减少软错误。考虑到磨损机制的共同相关性,文献[2]提出 EM 是主导老化效应,而 TDDB 和 NBTI 由于它们的相互作用对系统级可靠性影响不大。考虑到电迁移老化效应下的 MTTF 模型,文献[3]提出了一种基于 DVFS 的方法来提高嵌入式实时系统的可靠性。然而,不同于传统的异构系统,PIM+CPU 异构系统所特有的硬件特性对系统的性能具有重要的影响,而很少有研究考虑了各个处理器 MTTF 之间的差异对 PIM+CPU 异构系统可靠性的影响。

3 PIM+CPU 异构系统可靠性模型

异构系统整体可靠性受到诸如自身结构、软错误、磨损机制等因素的影响。本文主要研究老化对系统可靠性的影响,通过任务调度来优化异构系统整体的 MTTF。

3.1 功耗模型

在 PIM+多核 CPU 系统中,单个处理单元的瞬时功耗被划分为两个部分,分别为瞬时泄漏功耗和瞬时动态功耗:

$$P = P_{\text{dynamic}} + P_{\text{leakage}} \quad (1)$$

式中, P 是处理单元总的瞬时功耗; P_{dynamic} 是瞬时动

态功耗; P_{leakage} 是瞬时泄漏功耗。考虑到 PIM+CPU 系统经常处在运行状态,那么单个处理单元的瞬时功耗主要来源于动态功耗,即处理单元的瞬时功耗近似等于瞬时动态功耗,所以处理单元 i 的瞬时功耗如公式(2)所示:

$$P_i \approx P_{\text{dynamic}} \quad (2)$$

而瞬时动态功耗与处理单元的电压以及频率有关,如公式(3)所示:

$$P_i \propto V_{\text{dd}}(i)^2 \cdot f_i \quad (3)$$

式中, $V_{\text{dd}}(i)$ 表示处理单元 i 的电压; f_i 表示处理单元 i 的时钟频率。

3.2 温度模型

文献[4]指出,处理单元的温度收到多种因素影响,包括处理单元本身的平均功耗、任务运行时间以及环境温度。处理单元的累积温度受到动态功耗的影响,而处理单元的负载时间,即处理单元运行任务的时间,决定动态功耗。平均功耗是指处理单元在一段时间内产生的能耗除以时间的值。 P_{avg} 表示执行单个任务的平均功耗,对于单个任务, P_{avg} 可以近似的等于 P_i 。当运行特定任务时,处理器的绝对温度可以按公式(4)计算:

$$T = T_a + P_{\text{avg}} \cdot R \cdot (1 - e^{\frac{-\Delta t}{RC}}) \quad (4)$$

式中, Δt 表示负载时间; T_a 是环境温度; R 和 C 分别表示热电阻和电容。

3.3 MTTF 模型

在 PIM+CPU 异构多核平台中,我们采用 black 模型[5]来评估电迁移效应下单处理器的 MTTF。处理器 i 的 MTTF 如公式(5)所示:

$$\text{MTTF}_i \propto (V_{\text{dd}}(i) \cdot f_i \cdot p_i)^{-2} \cdot e^{\frac{E_a}{k \cdot T_i}} \quad (5)$$

式中, $V_{\text{dd}}(i)$ 表示处理单元 i 的电压; f_i 表示处理单元 i 的时钟频率; p_i 是处理单元 i 内晶体管的跳变概率; T_i 是处理单元 i 的温度; k 是玻尔兹曼常数; E_a 是活化能,也是常数。

根据 Black 模型可知,在 PIM+CPU 异构系统中,在其他因素相同的情况下,同一个任务在具有不同频率的处理单元上运行,会使处理单元之间的 MTTF 产生差异。因此,在 PIM+CPU 异构系统的任务执行期间,不仅要考虑单个处理单元的可靠性,还要考虑整个系统中所有处理单元之间的 MTTF 差异,以保障整个系统的可靠性。

3.4 优化目标模型

从上文的 MTTF 模型可以看出,处理单元的电压和频率对可靠性具有重要的影响作用,同时处理单元

的动态功耗也受到频率和电压的影响, 公式(5)给出了单个处理单元的 MTTF 计算模型. 对于 PIM+CPU 异构系统, 在每个处理单元的可靠性受到保障的前提下, 为了提升整个系统的可靠性, 需要将每个处理单元之间可靠性差异作为考虑的重要因素. 系统 MTTF 差异值的计算方法是系统所有处理单元 MTTF 值的平方差, 如公式(6)所示, 通过该差异值来衡量 PIM+CPU 异构系统的处理单元之间可靠性差异.

$$\text{VARIANCE}_{\text{MTTF}} = \frac{1}{n} \sum_{i=1}^n (\text{MTTF}_i - \text{MTTF}_{\text{avg}})^2 \quad (6)$$

在公式(6)中, MTTF_i 表示第 i 个处理单元的 MTTF; MTTF_{avg} 表示 PIM+CPU 异构系统中所有处理单元 MTTF 的算术平均值. 因此, 任务调度的优化目标就是最小化系统的 MTTF 差异值, 即:

$$\min(\text{VARIANCE}_{\text{MTTF}}) \quad (7)$$

4 访存干扰模型

图 1 所示一个典型的 PIM+CPU 异构系统. 与传统异构系统不同的是, PIM+CPU 异构系统中包含了一个计算型内存, 它的功能既包括传统的内存的功能又包括运算的功能, 在传统的内存读写命令的基础上, 还具有一套完备的指令集, 可以直接在内存芯片的内部完成以往需要在 CPU 中完成的功能. CPU 中执行任务的内存地址同过虚拟地址映射到 DRAM 阵列的物理地址, PIM 通过 DRAM 内的缓存行(cache line)直接访问 DRAM 阵列, 所以整个 CPU 与 PIM 共享 DRAM 内存, 这就带来 PIM 与 CPU 直接访问 DRAM 的干扰竞争问题. 不同的是, CPU 是通过系统总线访问 DRAM 内存, 而 PIM 是通过芯片内部的缓存行直接访问 DRAM. PIM 与 CPU 通过共享 DRAM(内存)的方式形成了一个完整的异构并行系统. 另外, PIM+CPU 异构系统中具有一个将内存与计算逻辑紧密耦合的 PIM 运算单元, 并且该 PIM 运算核心具有近数据计算的优势, 非常适合于执行访存密集型应用.

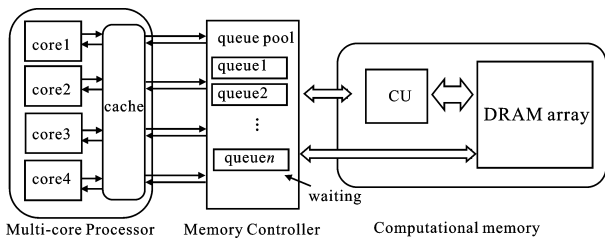


图 1 PIM+CPU 异构系统模型

在 PIM 与 CPU 混合的系统中, 访存干扰延迟通常包括以下两种情况: (1) 来自 CPU 的不同核同时访存时在访存请求队列排队等待的延迟; (2) 来自 CPU 和 PIM 的访存请求同时访问内存中的同一块区域所产生的竞争延迟.

4.1 CPU 之间的访存干扰延迟

多核 CPU 中并行访存请求的排队延迟本质上是一个排队问题. 假设处理单元与内存之间具有理想的互连网络, 并且访存具有随机性, 文献[6]提出了一个开放并且闭合的排队模型, 它表明存储器访问延迟是由内存中 bank 预留时间和存储器利用率的乘积给出的, 存储器利用率由 bank 预订时间内到达 bank 的平均访问请求数算出. 我们采用该排队模型来计算排队访存延迟:

$$D = \frac{u_0 - 1}{2} \cdot t_a + \sqrt{\frac{(u_0 - 1)^2}{4} + \gamma \cdot \frac{u_0 \cdot t_{\text{mem}}}{2 \cdot t_a}} \cdot t_a \quad (8)$$

$$u_0 = \frac{m \cdot t_{\text{mem}}}{\text{bank} \cdot t_a} \quad (9)$$

在公式(8)中, t_a 表示在没有访存干扰情况下平均每次访存之间的间隔时间, t_{mem} 表示内存的时钟周期时间, γ 表示在闭合系统中由于反馈所产生的修正因数, u_0 表示内存的利用率, 它可以通过公式(9)进行计算, 公式(9)中的 m 和 bank 分别表示处理单元的总数量以及内存中 bank 的数量. 当考虑访存请求的排队延迟时, 实际访存带宽 B_{actual} 可以通过公式(10)获得, 其中 B 代表理想的访存带宽, 并且 p 代表理想的平均存储器访问延迟:

$$B_{\text{actual}} = \frac{B}{1 + \frac{D}{p}} \quad (10)$$

4.2 CPU 与 PIM 之间的访存干扰延迟

对于 PIM 和 CPU 之间的访存竞争, 我们采用先来先服务的模型来评估访存竞争所产生的延迟. $\text{delay}_{\text{pim}}(t_i)$ 表示任务 t_i 在 PIM 上执行时因访存竞争所产生的延迟, $\text{delay}_{\text{cpu}}(t_i)$ 表示任务 t_i 在 CPU 上执行时因访存竞争所产生的延迟.

假设在 PIM+CPU 异构系统中有 C 个 CPU 核心和一个单独的 PIM 核心, 并且任务 $t_i (i = 1, 2, \dots, K)$ 被分配给处理单元 $p_j (j = 1, 2, \dots, C, C+1)$. K 表示任务图中的任务总数. p_{C+1} 是指 PIM. 当任务分配给 PIM 时, 延迟主要是内存冲突. 当任务分配给 CPU 时, 延迟包括排队延迟和冲突延迟. 在这种情况下, 可以采用公式(11) 计算在处理单元 p_j 上执行的任务的存储器访问延迟:

$$\text{delay}(t_i, p_j) = \begin{cases} \text{delay}_{\text{pim}}(t_i), & j = C + 1 \\ D \cdot \text{Inst}_{\text{num}} \cdot P_{\text{cache_miss}} + \text{delay}_{\text{cpu}}(t_i), & \text{others} \end{cases} \quad (11)$$

4.3 任务间通信模型

任务之间的通信开销与任务被分配的处理单元高度相关,所以任务间的通信延迟模型必须考虑任务位置.对于具有先后优先级关系并且相邻的任务,它们可以分配给 CPU 或 PIM,因此有四种可能的组合,并且通信延迟模型基本上是具有四个分段的精确函数.

我们假设 $C_{m,i}$ 表示 t_m 和 t_i 之间的通信成本,而这个成本主要取决于任务间的数据通信的开销,公式(12)给出了 $C_{m,i}$ 的算方法.当两个任务都在 CPU 上时,通信数据在缓存中有部分,其余部分在内存中.存储在缓存中的数据可以立即传输,这种情况下的通信延迟基本上是由内存数据传输到缓存引起的,这就是内存数据量除以可通过内存数据获得的内存带宽公式(10).当任务 t_m 在 CPU 中并且 t_i 在 PIM 中时,保存在缓存中的数据需要首先写回内存当中,而内存部分可以通过高带宽低延迟的接口传输到 PIM.对于其余两个任务位置场景,通信成本完全由内存中数据移动引起,并且可以通过公式中详述的数据传输量和通信带宽容易地进行计算.

$$C(t_m, t_i) = \begin{cases} T_{\text{init}} + \frac{\text{cache}_{m,i}}{B_{\text{actual}}(t_m)} + \frac{\text{data}_{m,i}}{B_{\text{pim}}(t_i)}, & t_m \text{ in CPU}, t_i \text{ in PIM} \\ T_{\text{init}} + \frac{\text{data}_{m,i}}{B_{\text{actual}}(t_i)}, & t_m \text{ in PIM}, t_i \text{ in CPU} \\ T_{\text{init}} + \frac{\text{data}_{m,i} - \text{cache}_{m,i}}{B_{\text{actual}}(t_i)}, & t_m \text{ and } t_i \text{ both in CP} \\ T_{\text{init}} + \frac{\text{data}_{m,i}}{B_{\text{pim}}(t_i)}, & t_m \text{ and } t_i \text{ both in PI} \end{cases} \quad (12)$$

式中, T_{init} 是内存访问的初始建立时间; $B_{\text{actual}}(t_i)$ 是任务 t_i 在运行时的实际带宽,参考公式(10). $\text{cache}_{m,i}$ 是 cache 中缓存的 t_m 和 t_i 之间的通信数据. $\text{data}_{m,i}$ 是缓存在内存中的 t_m 和 t_i 之间的通信数据.

4.4 PIM-MTTF 任务调度算法

PIM-MTTF 算法是为了优化 PIM+CPU 异构系统 MTTF 差异值的任务调度算法. PIM-MTTF 算法总体可以分为两大步骤,首先根据任务中任务之间的约束关系以及任务的运行参数对任务排序生成一个任务队列,第二步是依次从任务队列中取出任务,计算该任务在每个处理单元上执行后的每个处理单元的 MTTF 值(这里同时考虑访存竞争对任

务执行时间的影响),并根据这个值为该任务选择最优的处理单元.

4.4.1 任务图和任务队列

对于任意一个任务图,其中可能存在多个处在同一层级(可以同时执行)的任务,但是其中某些任务可能位于任务图执行的关键路径上,如果它们被排在同一层级任务的后面,整个应用程序的结束时间将会更长.因此,我们倾向于将关键路径上的任务分配给更高优先级,而具有更高权重值的任务将会提前执行.每项任务的权重值定义如公式(13)所示:

$$\text{priority}(t_i) = \overline{w}_i + \max_{t_j \in \text{succ}(t_i)} (\overline{C}_{i,j} + \text{priority}(t_j)) \quad (13)$$

$$\overline{w}_i = \frac{1}{m} \sum_{j=1}^m w_{i,j} \quad (14)$$

式中,任务 t_j 是任务 t_i 的后继任务; $\overline{w}_i$ 是任务 t_i 在所有处理单元上运行时间的平均值; $\overline{C}_{i,j}$ 是任务 t_i 和 t_j 的平均通信时间,等于公式(12)中四种情况的平均值; $\text{succ}(t_i)$ 表示任务 t_i 在任务图中的所有后继任务.从权重值的公式可以看出,每个任务的权重值依赖于该任务的后继任务,所以对对应一个给定的任务图,任务的权重值是从任务图的出口任务反向到入口任务依次递归计算计算.另外,出口任务 t_{exit} 的权重值为该任务的平均运行时间 $\overline{w}_{\text{exit}}$.

完成对所有任务权重值计算后,按照每个任务的权重值降序标准进行排序并生成任务队列.如果存在多个任务的权重值相等,为了简单起见,则随机确定这些任务的先后顺序.因此,在这种方式生成的任务队列中,排在前面的任务的权重值必定不会小于任意一个排在后面的任务的权重值,这就确保了对于任意一个任务图,任务队列中任务的先后顺序必定满足任务图中任务之间的先后关系约束.

4.4.2 处理单元选择策略

PIM-MTTF 任务调度算法的目标是通过减少不同处理单元之间的可靠性差异来提升 PIM+CPU 异构系统的可靠性.由于异构多核系统的任务调度是一个 NP 完全问题,所以采用贪心策略将任务分配给处理单元.对于任务队列中排在第一个的任务,首先根据每个处理单元的负载时间计算 MTTF 值,然后选择具有最大 MTTF 值的处理单元执行该任务,最后根据任务的运行时间,这个运行时间不仅包含该任务的预期运行时间,还包括该任务由于访存竞争所产生的竞争延迟时间,来更新该任务的处理单元的负载时间.完整的 PIM-MTTF 算法如下:

算法 1:PIM-MTTF 算法

```
Input:任务图
for  $i=1,2,\dots,n$  do
    根据目标系统计算任务  $i$  的平均运行时间
end for
将任务图中的出口任务放入队列 queue 中
while queue 不为空 do
    取出 queue 中第一个任务  $t$ 
    计算任务  $t$  的优先级权值
    将任务  $t$  的所有前驱任务依次放入队列 queue 中
end while
根据任务权值大小生成任务队列 task_queue
while task_queue 不为空 do
    取出 task_queue 中第一个任务  $t$ 
    for  $j=1,2,\dots,m$  do
        根据负载时间计算处理单元  $j$  的 MTTF
    end for
    将任务  $t$  分配到具有最大 MTTF 值的处理单元  $k$  上
    计算任务  $t$  在处理单元  $k$  上执行所产生的访存竞争延迟
    更新处理单元  $k$  的负载时间
end while
```

5 实验与结果

本部分将 PIM-MTTF 算法与经典的 HEFT^[7] 算法以及 MILP-based 算法进行对比. HEFT 算法是异构系统中著名的列表任务调度算法,它可以有效地提高异构系统的性能;MILP-based 算法是文献[8]中提出的平衡系统的 MTTF 算法的改进版本,该算法可以有效地提升系统整体的可靠性. 在实验中,我们从系统 MTTF 差异值、性能以及负载时间三个方面对 PIM-MTTF 算法进行评估.

5.1 实验参数

实验调度平台有一个 4 核 CPU 与一个 HMC 组成,每个处理单元的主频均为 1 GHz. 我们选择 Multi2Sim^[9],它是 CPU 和 GPU 的模拟器,并对其进行大量修改以模拟多核 CPU 和 PIM 体系结构. 另外,我们使用 HMC^[10]作为我们的计算内存模型. 我们使用 McPAT^[11]来评估系统的能耗. 实验采用的测试程序以及不同输入数据量的大小如表 1 所示.

5.2 系统 MTTF 差异值

在实验中,为了评估 PIM-MTTF 算法的有效性,采用 HEFT 算法以及 MILP-based 任务调度算法作为对比. 图 2 给出了在三种不同输入数据集的情况下归一化的系统 MTTF 差异值,归一化的基准值是采用 HEFT 调度算法的系统 MTTF 差异值. 实验结果证明了 PIM-MTTF 算法的有效性. 在

表 1 不同应用的输入数据量大小

应用程序	输入数据量大小		
	Small	Medium	Large
PageRank	2 M	16 M	118 M
Breadth First Search(BFS)	1 M	4 M	16 M
K-means	1 M	8 M	50 M
Sparse Matrix-Vector Multiply(SPMV)	11 M	44 M	136 M
Connected Component(CC)	1 M	5 M	31 M
Topological Sorting(TS)	2 M	10 M	43 M

large 输入数据集情况下,采用 PIM-MTTF 算法的平均 MTTF 差异值比 HEFT 平均降低 57.8%,尤其是对于 PageRank、BFS 以及 CC,系统的 MTTF 差异值得到非常显著的降低. 即使在 small 输入数据集情况下,PIM-MTTF 仍可平均降低 21.3%的 MTTF 差异值. 此外,与 MILP-based 算法相比,在 large 输入数据集下,PIM-MTTF 平均降低 9.1%的系统 MTTF 差异值.

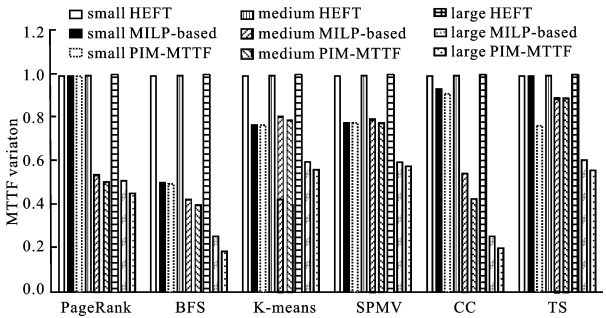


图 2 三种任务调度算法的 MTTF 差异值

5.3 性能提升

图 3 所示为目标系统在三种任务调度算法下不同测试程序的归一化的运行时间,归一化的参考基准是采用 HEFT 算法的运行时间. 从图中可以看出,相比较与 HEFT 算法,PIM-MTTF 算法能为 PIM+CPU 系统带来平均 13.1%的性能提升,这是因为尽管 PIM-MTTF 算法旨在减少系统的 MTTF 差异值,但同时也考虑到内存访问问题(如内存访问竞争),从而减少系统的内存干扰. 减少的内存干扰使得系统的性能得到改善. 而 PIM-MTTF 算法与 MILP-based 算法相比,在性能上比较没有区别.

5.4 负载时间

图 4 给出了在使用 PIM-MTTF 算法情况下目标系统中 4 个 CPU 核以及一个 PIM 归一化的负载时间,归一化的参考基准是 CPU 核 1 的负载时间. 如图所示,不同处理单元负载时间的平均差异为 1.9%,最大为 9.9%,可以看出每个处理单元的负载时间存在的差异比较小. 这是因为 PIM-MTTF 算法可以很好地处理每个计算单元之间的差异,并

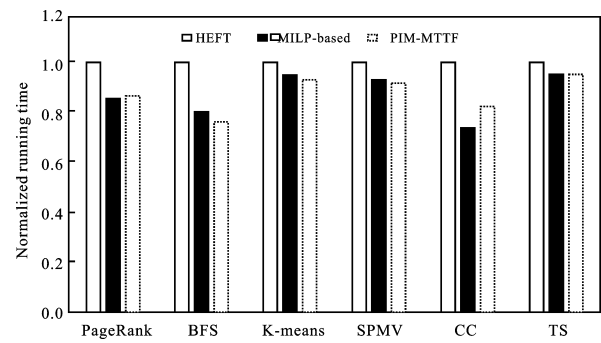


图 3 不同调度算法下的系统性能

且使每个处理的负载时间得到均衡。

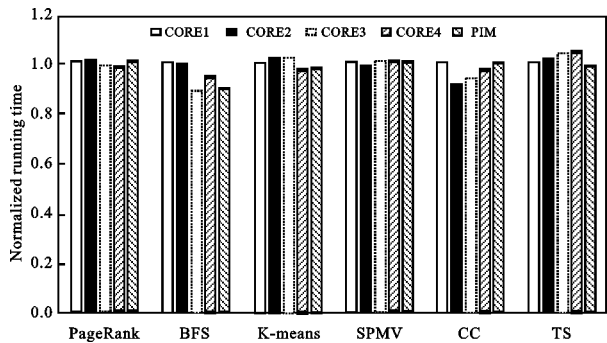


图 4 Large 数据量下的归一化的负载时间

6 结束语

本文针对新型的 PIM+CPU 异构计算系统的可靠性问题建立了形式化的任务调度模型,同时提出了基于 MTTF 的任务调度算法—PIM-MTTF 来平衡 PIM + CPU 异构系统的 MTTF. 由于 PIM 硬件结构的特殊性,我们也考虑了 CPU 与 PIM 之间的访存竞争、带宽竞争等问题. 实验结果显示,相比较与经典的异构系统任务调度算法 HEFT, PIM-MTTF 算法平均降低了 57.8% 系统 MTTF 差异值,而相比与平衡系统 MTTF 的 MILP-based 算法, PIM-MTTF 算法平均降低了 9.1% 系统 MTTF 差异值,证明了本模型和算法的实用性.

参考文献:

[1] Yuan L, Leventhal S, Qu G. Temperature-aware leakage minimization technique for real-time systems[C]//Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design. San Jose, CA, USA, ACM, 2006: 761-764.

[2] Wu K C, Lee M C, Marculescu D, et al. Mitigating lifetime underestimation: a system-level approach considering temperature variations and correlations between failure mechanisms[C]//Design, Automation &

Test in Europe Conference & Exhibition (DATE). Dresden, Germany, 2012: 1269-1274.

[3] Kim T, Zheng B, Chen H B, et al. Lifetime optimization for real-time embedded systems considering electromigration effects[C]//Computer-Aided Design (IC-CAD), 2014 IEEE/ACM International Conference on. Hong Kong, China, IEEE, 2014: 434-439.

[4] 刘勇攀, 杨华中, 汪蕙. 一种热感集成电路功耗模型及其对动态电压调整技术的影响[J]. 半导体学报, 2007, 28(4): 530-536.

[5] Huang L, Yuan F, Xu Q. On task allocation and scheduling for lifetime extension of platform-based MPSoC designs [J]. IEEE Transactions on Parallel and Distributed Systems, 2011, 22(12): 2088-2099.

[6] Bucher I Y, Calahan D A. Access conflicts in multi-processor memories queueing models and simulation studies[J]. ACM SIGARCH Computer Architecture News, 1990, 18(3b): 428-438.

[7] Topcuoglu H, Hariri S, Wu M. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE transactions on parallel and distributed systems, 2002, 13(3): 260-274.

[8] Liang H, Dai Y, Yi M, et al. MTTF-aware reliability task scheduling for heterogeneous multicore system [C]//International Conference on Algorithms and Architectures for Parallel Processing. Berlin, Springer, Cham, 2015: 716-727.

[9] Ubal R, Sahuquillo J, Petit S, et al. Multi2sim: A simulation framework to evaluate multicore-multithread processors[C]//IEEE 19th International Symposium on Computer Architecture and High Performance computing. Piscataway, NJ, USA, 2007: 62-68.

[10] Nai L, Kim H. Instruction offloading with hmc 2.0 standard: A case study for graph traversals[C]//Proceedings of the 2015 International Symposium on Memory Systems. Portland, Oregon, ACM, 2015: 258-261.

[11] Li S, Ahn J H, Strong R D, et al. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures[C]//Micro-architecture, MICRO-42, 42nd Annual IEEE/ACM International Symposium on. New York, NY, USA, IEEE, 2009: 469-480.

作者简介:

庞德松 男,(1993-),硕士研究生. 研究方为计算机系统任务调度. E-mail:1078472788@qq.com.

梁华国 男,(1959-),博士,教授. 研究方向为嵌入式系统综合与测试、数字系统设计自动化.