软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2017,28(4):845−859 [doi: 10.13328/j.cnki.jos.005194]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

数据中心中 DVFS 对程序性能影响模型的设计*

李登辉[1,2], 赵家程[1,2], 崔慧敏[1], 冯晓兵[1]

[1](计算机体系结构国家重点实验室(中国科学院 计算技术研究所),北京 100190)

[2](中国科学院大学 计算机控制与工程学院,北京 100049)

通讯作者: 崔慧敏, E-mail: cuihm@ict.ac.cn

摘 要: 数据中心以可接受的成本,承载着超大规模的互联网应用.数据中心的能源消耗直接影响着数据中心的一次性建造成本和长期维护成本,是数据中心总体持有成本的重要组成部分.现代的数据中心普遍采用动态电压频率调节(dynamic voltage frequency scaling,简称DVFS)来提升单节点的能耗表现.但是,DVFS这一类机制同时影响到应用的能源消耗和性能,而这一问题尚未被深入探索.专注于 DVFS 机制对应用程序性能的影响,提出了一个分析模型用来量化地刻画应用程序的性能与处理器频率之间的关系,可以预测程序在任意频率下的性能.具体来说,依据执行时访问内存子系统资源的不同,把程序的指令分为两部分——片上指令和片外指令,并分别独立建模.片上指令是指仅需访问片上资源就可以完成执行的指令,其执行时间与处理器频率呈线性关系;片外指令是指需要访问主存的指令,其执行时间与处理器频率无关.通过上述划分和对每一部分执行时间的分别建模,可以获得应用程序的执行时间与处理器频率之间的量化模型.使用两个不同的平台和 SPEC 2006 中的所有标准程序验证该模型,平均误差不超过 1.34%.

关键词: DVFS;数据中心;能耗;频率;性能预测模型

中图法分类号: TP316

中文引用格式: 李登辉,赵家程,崔慧敏,冯晓兵.数据中心中DVFS对程序性能影响模型的设计.软件学报,2017,28(4):845−859. http:// www.jos.org.cn/1000-9825/5194.htm

英文引用格式: Li DH, Zhao JC, Cui HM, Feng XB. Modeling the impact of DVFS on performance of applications in datacenter. Ruan Jian Xue Bao/Journal of Software, 2017,28(4):845−859 (in Chinese). http://www.jos.org.cn/1000-9825/5194.htm

Modeling the Impact of DVFS on Performance of Applications in Datacenter

LI Deng-Hui[1,2], ZHAO Jia-Cheng[1,2], CUI Hui-Min[1], FENG Xiao-Bing[1]

[1](State Key Laboratory of Computer Architecture (Institute of Computing Technology, The Chinese Academy of Sciences), Beijing 100190, China)

[2](School of Computer and Control Engineering, University of Chinese Academy of Sciences, Beijing 100049, China)

Abstract: Datacenters are built to house massive internet services at an affordable price. Both Op-ex (long-time operational expenditure) and Cap-ex (one-time construction costs) are directly impacted by datacenter power consumption. Thus, DVFS (dynamic voltage and frequency scaling) is widely adopted to improve per node energy efficiency. However, it is well known but has not yet been fully explored that such schemes affect an application's power consumption and performance simultaneously. This paper focuses on the impact of DVFS

* 基金项目: 国家重点基础研究发展计划(973)(2016YFB1000402); 国家高技术研究发展计划(863)(2015AA015306); 国家自然科学基金(61672492, 61432016, 61402445, 61521092)

Foundation item: National Basic Research Program of China (973) (2016YFB1000402); National High Technology Research and Development Program of China (863) (2015AA015306); National Natural Science Foundation of China (61672492, 61432016, 61402445, 61521092)

收稿时间: 2016-06-19; 修改时间: 2016-09-08; 采用时间: 2016-11-26; jos 在线出版时间: 2017-01-24

CNKI 网络优先出版: 2017-02-20 13:43:26, http://www.cnki.net/kcms/detail/11.2560.TP.20170220.1343.001.html

on performance of an application and proposes an analytical model to quantitatively characterize the relationship between an application's performance and a processor's frequency, which can be leveraged to predict the performance of an application at any frequency. Specifically, according to different memory subsystem resources accessed, instructions of an application are divided into two parts: on-chip instructions and off-chip instructions, which can be modeled independently. On-Chip instructions refer to instructions which only access on-chip resources, and their execution time is frequency-relevant and can be modeled using a linear function. Off-chip instructions stand for instructions accessing the main memory, and their execution time is dominated by memory access latency and is frequency-irrelevant. By the division and modeling of the two parts, a quantitative model can be obtained between the execution time of an application and frequency of a processor. Evaluations using two different platforms and all benchmarks of SPEC 2006 show that the derived models are very precise, with average prediction error less than 1.34%.

Key words: DVFS; datacenter; energy cost; frequency; performance prediction model

计算领域正在迈入云计算的时代,由数千至数十万台服务器组成的数据中心逐步成为核心的计算平台,开始承载日益增多的应用.而与此同时,数据中心的整体持有成本(total cost of ownership,简称 TCO)也在不断上升[1].因此,如何降低数据中心的 TCO,成为学术界和工业界都密切关注的一个关键问题,给研究人员带来了新的挑战.

现有的研究表明:超过 15%的数据中心成本来自于数据中心服务器的能源消耗,而如果将散热等成本考虑进去,这一比例甚至可以超过 40%[2].这意味着数据中心的能源效率,是影响数据中心 TCO 的重要因素[3].而数据中心由数千至数十万的服务器节点组成,如何提高每一个节点的能效表现则至关重要.单节点的能耗极大程度上依赖于现代处理器提供的能源管理支持,而动态电压频率调节(dynamic voltage frequency scaling,简称 DVFS)是其中最重要的组成部分.DVFS 诞生在 20 世纪 90 年代[4],提供了一套根据当前系统的工作负载情况动态调节处理器电压和频率来节省能耗的机制[5−9],可以实现功耗和性能的平衡,并普遍被现代处理器采用.

数据中心的能耗是当前学术界和工业界共同关注的热点问题,借助于 DVFS 机制,为了提升数据中心的能耗表现,针对不同的应用类型(交互式应用[10]、延时敏感型的应用[11−14]),不同的优化目标(性能-功耗[10]、QoS-功耗[11−14]),从软件栈的不同层次(运行时环境[10−14]、编译[15,16]),国内外研究人员展开了诸多深入的研究[17−21].首先,从高效地利用 DVFS 机制减少能耗的角度出发,如基于 DVFS 机制进行任务调度来实现节能的目的[22−27];从虚拟化的节能算法、基于主机关闭开启的节能算法[28−32],还有从其他各个方面来探索数据中心中的节能方案[33−36].进一步地,研究人员开始在预测模型的指导下,更加高效地调节处理器频率[10−12,15,16].但这些方案中,当处理器处于工作状态时,仍然是以最高的频率、最耗能的方式在运行,欠缺对应用程序性能的量化研究,从而不能精确地实现性能和功耗的共同优化.另一方面,研究者们进行了大量的研究来提升处理器的能耗表现,如:Femal[37]介绍了一种非统一的能耗分配方案,动态地分配服务器之间的功率来提高数据中心中的吞吐量;Gandhi[38]通过寻找功率和频率的关系建立功率分配方案优化地来提高数据中心的性能,注重于提高性能,缩短响应时间;Pack & Cap[39]主要是针对多线程的并行应用程序,利用 DVFS 机制设计的优化方案来在限定的功率下节省能耗.但是,DVFS 机制同时影响处理器的功耗性能,这些研究都专注于借助 DVFS 机制提高处理器的能耗表现,欠缺对应用程序性能的量化研究,从而不能精确地实现性能和功耗的协同优化.在能耗与频率关系的量化研究方面,Mauro 等人[40]给出了计算模型公式来刻画能耗与应用程序 CPU 占用时间及频率之间的关系,但是该研究针对的是能耗随频率的变化情况,并没有关注频率变化对应用程序性能上的影响,并不能在节省能耗的同时,对应用程序应该在怎样的频率下运行才算合适给出指导方案;Shoaib 等人的研究[41]给出了一个量化的模型来反映应用程序的性能和频率的关系,但是该研究是从程序语言的角度出发,分析了 Java 应用程序的代码特征建立的预测模型,并不能精确到底层的 CPU 和访存行为的预测,有很大的局限性,故而误差比较大.

而当涉及到程序的性能与频率之间关系的模型时,研究人员已经注意到,简单的线性模型不能精确地刻画执行时间与频率的关系;并已经意识到,需要将访存指令与计算指令分开考虑.而不同的研究人员对此有不同的模型假设,如:文献[11]仅假设频率越高程序的性能越好,并没有构建量化的模型;文献[15,16]以编译手段为基础,在静态代码分析的基础上构建了程序执行时间与频率之间关系的模型;文献[10]仅以运行时获得的多个频率下的执行时间为基础,使用回归分析的手段构建两者的模型.本文从程序本身的运行时特征出发去构建程序执行

时间与频率之间关系的模型,仅需在一个频率下的一次执行,无需多个频率下的先验知识,可以适应在数据中心中 on-the-fly 的部署.

本文专注于探讨应用程序的性能对 DVFS 机制的敏感性,着眼于 DVFS 机制对程序性能的影响,提出并构建了一个精确的分析模型来刻画应用程序的性能与处理器频率的关系,可以精确地预测应用程序在不同的频率下的性能,其主要内容包括:(1) 根据运行时指令的完成是否需要访问主存,将指令分为片上完成指令和片外完成指令两部分,并分别构建了其执行时间与处理器频率之间关系的理论模型,从而可以获得程序的执行时间与频率的关系;(2) 提出一种实用的方法在主流处理器上获得上述理论模型;(3) 通过实验验证了上述模型和方法的有效性,平均误差不超过 1.34%.可以精确地反映程序的性能与处理器频率之间的关系,为数据中心中选择合适的频率运行该应用程序提供依据,进一步地,可以协助实现性能和功耗的协同优化.

本文第 1 节介绍我们的研究动机.第 2 节介绍理论模型及其实际构建方法.第 3 节是实验部分,验证模型和方法的精确性.第 4 节总结全文.

1 研究动机

正如前文所阐述的,DVFS 机制带来的频率改变将会显著影响程序的性能.在本节,我们将用典型程序在主流平台上的实验结果说明:(1) 频率直接影响程序的性能,且不能用简单的线性模型刻画;(2) 应用程序对频率变化的敏感性不同.

1.1 频率对程序性能的影响

众所周知,处理器的频率变化将直接影响程序的执行时间.直观上,程序的执行时间可以认为是与处理器频率线性相关的,但是我们通过实验发现,简单的线性模型并不能反映出真实的性能变化.我们使用了 SPEC CPU 2006 中的应用 429.mcf 在一个主流的 Intel 六核平台(Intel Xeon E5-2620@2.00GHz)上进行了实验.具体来说,我们设定处理器工作在不同的频率之下并测试程序相应的执行时间,计算出在各个频率下归一化后的程序性能.图 1 展示的是 429.mcf 的测试结果,其中,横轴表示处理器的工作频率,纵轴表示归一化后的程序的性能,程序的性能由下述公式计算:

$$P_f = \frac{1}{t_f} \tag{1.1}$$

程序归一化的性能由下述公式计算:

$$P_{f\text{-}Normalized} = \frac{P_f}{P_0} = \frac{t_0}{t_f} \tag{1.2}$$

其中,t_0 为程序在该平台最低频率 f_0(1.2GHz)下程序运行的时间,t_f 为频率 f 下程序的运行时间,P_0 为最低频率(1.2GHz)下程序的性能,P_f 为频率 f 下程序的性能,$P_{f\text{-}Normalied}$ 是归一化后程序的性能.

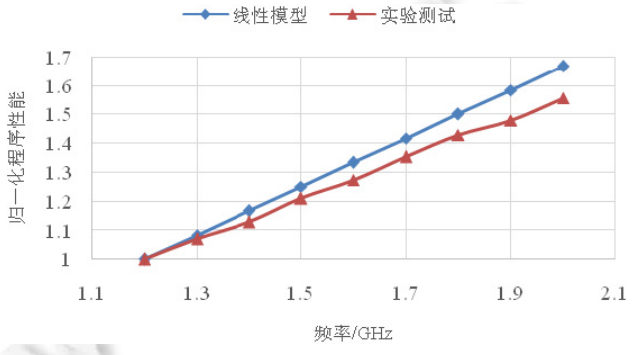


Fig.1 Correlationbetween performance of 429.mcf and CPU's frequency

图 1 429.mcf 性能与频率之间的关系

图 1 中,横轴是处理器的工作频率,纵轴是归一化后的程序性能.其中,三角形的点是实际测得的性能,方形的点是使用简单线性模型估计得来的.线性模型的归一化性能由下述公式计算得到:

$$t_f=\#cycles\cdot T_f \tag{1.3}$$

$$P^{predicted}_{f\text{-}Normalised}=\frac{t_0}{t_f}=\frac{\#cycles\cdot T_0}{\#cycles\cdot T_f}=\frac{f}{f_0} \tag{1.4}$$

其中,t_f为频率f下程序的运行时间,$\#cycles$指的是程序执行的时钟周期数,T_f指的是频率f下一个时钟周期的时间,$P^{predicted}_{f\text{-}Normalised}$指的是简单的线性模型预测的归一化后的程序性能,$t_0$为程序在该平台最低频率$f_0$(1.2GHz)下程序运行的时间,$T_0$指的是最低频率$f_0$(1.2GHz)下一个时钟周期的时间.在简单的线性模型中,程序执行总的时钟周期数不随频率变化,因此,程序的执行时间可由公式(1.3)直接计算.公式(1.4)表明,预测的程序的归一化性能与处理器频率是简单的线性关系.

从图 1 中可以观察到:程序的归一化性能随着频率的增加而增大,具有极强的相关性,符合我们直观上的认识.但是简单的线性模型并不能描述这种变化,以频率为 2.0GHz 为例,简单线性模型预测的归一化性能为 1.67,而实际的性能为 1.55,误差高达 7.10%.而同时我们也注意到:随着频率调节范围的扩大,程序执行时间的变化幅度同样增大.以图 1 中的 429.mcf 为例,随着频率从 1.2GHz 增加到 2.0GHz,程序的归一化性能提高了 55%.而随着处理器技术的发展,现代芯片的频率调节范围将会逐步扩大,所以功耗控制导致应用程序的性能变化将会更加突出.

1.2 应用程序对频率的敏感性

上文阐释了程序的性能对处理器频率有极强的敏感性,本节将进一步利用实验数据说明这种敏感性并不唯一.也就是说,不同的应用程序对频率有着不同的敏感性.

图 2 反映了这种敏感性造成的性能变化之间的差距.这是在 Intel Xeon CPU E5-2620@2.00GHz 上 SPEC CPU 2006 中不同的 5 个应用在不同频率下所反映出的归一化性能.该平台上的 CPU 频率变化范围是 1.20GHz~2.00GHz,频率变化的步长为 100MHz.图 2 中,横轴是处理器的频率,纵轴是归一化的程序性能,由公式(1.2)计算得来,不同颜色的线代表不同程序的性能随着频率变化的变化曲线.从实验结果可以看出,不同的程序受到频率变化带来的影响不一样.当频率提高时,各个程序性能的提升程度各不相同.如图 2 所示:当频率为 2.0GHz 时,相对于频率为 1.2GHz 时,处理器的频率提升了 0.8GHz,453.povray 的性能提升了 67.5%,而 471.omnetpp 仅提升了 51.6%.由此可以发现,不同的程序对频率有着不同的敏感性.为了探讨频率对程序的影响情况,我们将需要寻找合适的程序特征来刻画应用程序的性能变化与处理器频率之间的关系.

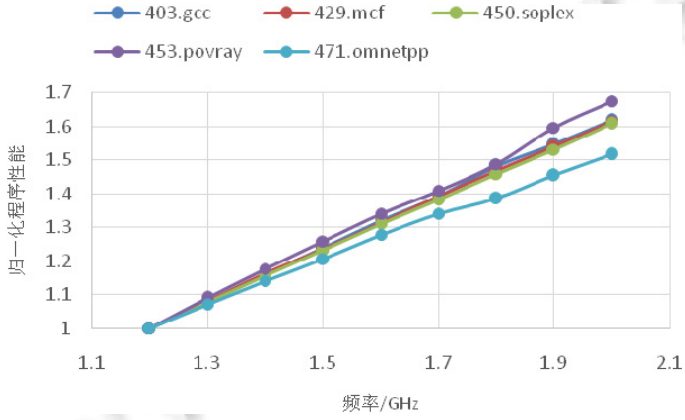


Fig.2 Sensitivity of applications to CPU's frequency

图 2 应用程序对频率的敏感性

2 建立性能预测模型

本节的主要内容是从体系结构的角度出发,从理论上分析程序的执行过程,并针对程序的执行过程构建一个量化的模型来刻画程序的执行时间与频率变化的关系,并提出在当前主流硬件平台上可行的方法来获取相应的模型参数,并最终建立程序的性能预测模型.

2.1 模型化程序的执行过程

本节,我们将首先从现代处理器典型的层次存储体系出发分析应用程序的执行过程,并以此为基础,提出一个简化的模型来表示应用程序对应的执行时间,并进一步地分析执行时间各组成部分与处理器频率的关系.

如图 3 所示,为了实现成本、速度和容量的折衷,现代处理器普遍具有层次化的存储结构,越靠近 CPU 的部分,访问延迟越低,而同时容量也越少,如图 3 所示的寄存器和各级缓存;而越远离 CPU 的部分,延迟较高,但是容量也相对较大,如图 3 所示的内存部分.

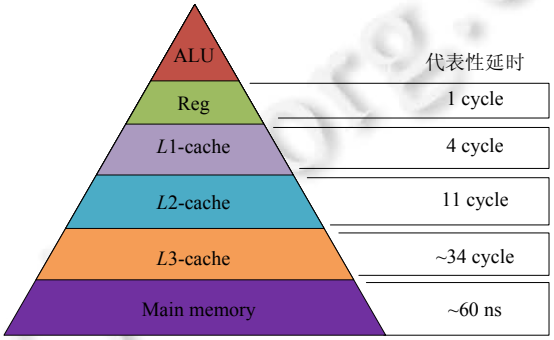


Fig.3 Memory hierarchy and representative latencies of modern CPU architecture[42]

图 3 现代处理的层次存储架构和典型的访问延迟[42]

现代处理器具有包括超标量、多发射等在内的复杂体系结构特征,这使得建立一个精确的模型描述程序的执行过程是不可能完成的任务.而这种层级存储结构具有一个非常典型的特征,访存延迟是逐级单调增大,这种鲜明的结构特征引导我们建立了一个简化的模型来描述程序的执行过程.具体来说,我们假设所有的指令都是串行执行,那么我们可以依据程序指令访问的延迟最大的资源对指令进行分类,而依据我们的串行假设,程序的执行时间可以由如公式(2.1)表示.

$$T = T_{reg} + \sum T_{Li} + T_{memory} \tag{2.1}$$

其中,T 代表程序总的执行时间,T_{reg} 代表仅需访问寄存器资源的指令所需要的时间,T_{Li} 代表需访问第 i 级 cache 指令的执行时间,$T_{menmory}$ 代表需要访问内存指令的执行时间.这样的模型化简使得程序的执行时间与存储器的层次结构关联起来,使得我们进一步地分析程序的执行时间与频率关系成为可能,我们将用实验结果验证这样化简的模型仍旧是足够精确的.接下来,探讨访问不同层级的资源对应的执行时间与处理器频率的关系.

2.2 建立预测模型

本节将分析公式(2.1)中各部分时间与处理器频率的关系,并建立最终的理论预测模型.

T_{reg}:处理器访问寄存器的速度是固定的周期数,而运算单元所需要的周期数同样是固定的.这意味着:当 CPU 的频率 f 发生变化时,时钟周期 CLK=1/f 随之变化.而访问寄存器的延迟也是同步变化的,这意味着指令的执行时间也将同步变化.于是我们假设在某个频率下得到了所有仅需访问寄存器的指令所需要的运算单元和访问寄存器的周期数,则可以将仅需访问寄存器的指令的执行时间在任意频率下的执行时间表示为

$$T_{reg} = T_{ALU} + T_{reg\text{-}latency} = \#cycles_{ALU} \cdot \frac{1}{f} + \#cycles_{reg} \cdot \frac{1}{f} = I_{reg} \cdot CPI_{reg} \cdot \frac{1}{f} \tag{2.2}$$

其中,T_{reg} 表示仅需访问寄存器的指令的执行时间;T_{ALU} 表示这些指令执行过程中运算单元 ALU 的执行时间;$T_{reg\text{-}latency}$ 表示寄存器的访问造成的时间开销;$\#cycles_{ALU}$ 表示仅访问寄存器的指令 ALU 执行所需时钟周期数;$\#cycles_{reg}$ 表示仅访问寄存器的指令所需时钟周期数;I_{reg} 表示仅需访问寄存器的指令的数目;CPI_{reg} 表示平均每条仅访问寄存器的指令所需平均时钟周期数,包含访问时延和运算时间,这个参数是为了更方便地表达程序的执行时间而引入的,只能通过计算获得,无法直接测试得到;f 表示 CPU 的频率.

T_{Li}:处理器访问各级缓存的延迟同样是固定的周期数.对于这些需要访问缓存的指令,由于我们假设的机器模型是顺序串行的,则程序必须要等待数据返回才能继续,执行时间需要根据数据返回的延迟来决定.但是,由于缓存访问的延迟同样由处理器的频率决定,所以和仅需访问寄存器的指令一样,其执行时间将会同步变化.对第 i 级缓存,其延时可由下列公式计算:

$$T_{Li}=T_{ALU}+T_{reg\text{-}latency}+T_{cache\text{-}latency}=\#cycles_{ALU}\cdot\frac{1}{f}+\#cycles_{reg}\cdot\frac{1}{f}+\#cycles_{cache}\cdot\frac{1}{f}=I_{Li}\cdot CPI_{Li}\cdot\frac{1}{f} \tag{2.3}$$

其中,T_{Li} 表示需要访问第 i 级缓存的指令执行所需要的时间,T_{ALU} 表示这些指令执行过程中运算单元 ALU 的执行时间,$T_{reg\text{-}latency}$ 表示寄存器的访问造成的时间开销,$T_{cache\text{-}latency}$ 表示这些指令访问第 i 级缓存的延时,$\#cycles_{ALU}$ 表示这部分指令执行过程中 ALU 执行所需要的时钟周期数,$\#cycles_{reg}$ 表示这部分指令执行过程中访问寄存器所需要的时钟周期数,$\#cycles_{cache}$ 表示这些指令访问到第 i 级缓存过程中所需要的时钟周期数,I_{Li} 表示第 i 级缓存命中的指令数目,CPI_{Li} 表示平均每条缓存命中指令所需要的时钟周期数,f 表示 CPU 的频率.

T_{memory}:程序访问内存的延迟由内存参数决定,与 CPU 的频率无关,那么对于缓存未命中的指令而言,由于内存的延迟非常大(如图 3 所示,若频率为 2.0GHz,大约是寄存器访问延迟的 120 倍),程序的执行时间主要由访存的延迟决定,ALU 部分的时间可以忽略.从而我们可以用下述公式计算 T_{memory}:

$$T_{memory}=I_{memory}\cdot t_{mem\text{-}latency} \tag{2.4}$$

其中,T_{memory} 表示需要访问内存的指令所花费的时间,I_{memory} 表示需要访问内存的指令数目,$t_{mem\text{-}latency}$ 表示每次访问内存的延迟.

综上所述,程序总的执行时间可以表示为

$$\begin{aligned}T&=I_{reg}\cdot CPI_{reg}\cdot\frac{1}{f}+\sum I_{Li}\cdot CPI_{Li}\cdot\frac{1}{f}+I_{memory}\cdot t_{mem\text{-}latency}\\&=(I_{reg}\cdot CPI_{reg}+\sum I_{Li}\cdot CPI_{Li})\cdot\frac{1}{f}+I_{memory}\cdot t_{mem\text{-}latency}\\&=(I_{reg}+\sum I_{Li})\cdot CPI_{on\text{-}chip}\cdot\frac{1}{f}+I_{memory}\cdot t_{mem\text{-}latency}\end{aligned} \tag{2.5}$$

这里,把计算指令和缓存命中指令各自平均每条指令的时钟周期数 CPI_{reg} 和 CPI_{Li} 合并为所有在芯片中完成指令的平均时钟周期数 $CPI_{on\text{-}chip}$.

由此可以发现,公式(2.1)的各部分时间可以分为两类:一类是与处理器频率相关的 T_{reg} 和 T_{Li},这类时间所执行的指令都在芯片上完成,故称为片上完成指令;另一类是与频率无关的 T_{memory},这类时间所执行的指令都在芯片外完成,称为片下完成指令.那么,程序的执行时间可以表示为

$$T=T_{on\text{-}chip}+T_{off\text{-}chip} \tag{2.6}$$

其中,T 表示程序总的执行时间,$T_{on\text{-}chip}$ 表示片上完成指令执行的时间,$T_{off\text{-}chip}$ 表示片外完成指令执行的时间.于是,从这种执行时间的分类出发,我们可以把程序的所有指令进行分类,如图 4 所示,根据执行时间与频率是否相关分为片上完成指令和片下完成指令.

现在,不妨设一个程序总的指令数为 I,最后一级缓存缺失的比例为 r,那么 $T_{off\text{-}chip}$ 可由下列公式计算得到:

$$T_{off\text{-}chip}=I\cdot r\cdot t_{off\text{-}chip} \tag{2.7}$$

其中,$t_{off\text{-}chip}$ 为一条片下完成的指令平均完成的时间,与频率无关,由内存本身的参数确定.

同理,$T_{on\text{-}chip}$ 可以由下列公式计算得到:

$$T_{on\text{-}chip}=I\cdot(1-r)\cdot t_{on\text{-}chip} \tag{2.8}$$

其中,$t_{on\text{-}chip}$ 为一条片上完成指令平均完成的时间.$t_{on\text{-}chip}$ 是以时钟周期为单位的,完全由频率决定,则 $t_{on\text{-}chip}$ 的计算公式如下:

$$t_{on\text{-}chip} = CPI_{on\text{-}chip} \cdot \frac{1}{f} \tag{2.9}$$

由公式(2.6)~公式(2.9)可以得到程序运行的总时间 T 的计算公式为

$$T = I \cdot r \cdot t_{off\text{-}chip} + I \cdot (1-r) \cdot CPI_{on\text{-}chip} \cdot \frac{1}{f} \tag{2.10}$$

根据这个模型,对任意一个应用程序,只需要获取程序的指令数 I、缓存缺失的比例 r、程序片上完成指令平均的 $CPI_{on\text{-}chip}$ 以及该平台下的访存指令完成时间 $t_{off\text{-}chip}$ 这些特征,就可以得到程序在任意频率 f 下的性能.

从该模型可以看出,片外完成指令所需总时间是一个定值,所以,当程序缓存缺失的比例 r 越大时,程序随频率变化的执行时间就越短,性能的提升幅度就越小.

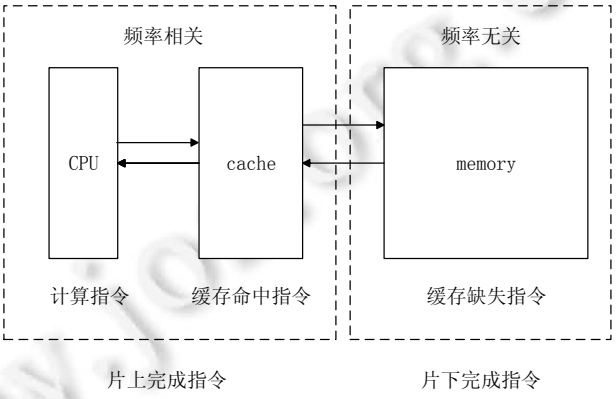


Fig.4 On-Chip instructions and off-chip instructions of an application: Two independent parts

图 4 程序指令可以分为独立的片上指令和片外指令两部分

2.3 获得模型参数

上文给出了理论上程序的性能模型,本节将给出一种可行的方法来获得模型的所有参数.具体来说,本文采用在一个基准频率下运行一遍的方式获取模型的所有参数,据此预测程序在其他频率下的执行时间,并使用第 1 节中的公式(1.2)来计算程序的归一化性能.如公式(2.10)所示,其中,I,r,$t_{off\text{-}chip}$,$CPI_{on\text{-}chip}$ 是模型所需参数,下面我们研究如何获得这些参数.

I,r:I 表示的是程序的指令总数,r 表示的是最后一级缓存未命中的指令占总指令数目的比例.对于这两个参数,我们都可以借助于 profile 来获得.具体来说,VTune[43]可以通过监测 *INST_RETIRED.ANY* 这个硬件事件得到一个程序所执行的所有指令数 I,而片外完成的指令数则相当于缓存缺失的指令数,亦即 $L3$ 缓存缺失的次数,由此我们可以通过监测 *MEM_LOAD_RETIRED.LLC_MISS* 这个硬件事件得到 $L3$ 缓存缺失的次数 n,以此计算出一个程序片下完成指令的比例 $r = \frac{n}{I}$.值得注意的是:在不同的硬件平台下,获取程序的特征的硬件事件不一定相同,在选择硬件事件时应仔细查阅相关资料获取相对应的硬件事件.

$t_{off\text{-}chip}$:$t_{off\text{-}chip}$ 反映了访问一次内存的平均时间,由内存本身的性质确定,不随 CPU 的频率变化而变化,我们可以用内存延迟测试工具 Memory Latency Checker[44]对平台进行测试获取.

$CPI_{on\text{-}chip}$:$CPI_{on\text{-}chip}$ 指的是片上完成指令的平均 CPI,现有的 profile 工具很难直接测试出来,但是可以发现:在公式(2.10)中,如果在一次测试中,在基准频率 f_0 下得到了程序的执行时间 T 以及其他的相关参数 I,r,$t_{off\text{-}chip}$,那么我们可以通过公式(2.10)计算出该程序的 $CPI_{on\text{-}chip}$ 数值,计算公式如下:

$$CPI_{on\text{-}chip}=\frac{T_0-I\cdot r\cdot t_{off\text{-}chip}}{I\cdot(1-r)\cdot\frac{1}{f_0}} \tag{2.11}$$

其中,T_0表示程序总的执行时间,I表示的是程序的指令总数,r表示的是最后一级缓存未命中的指令占总指令数目的比例,$t_{off\text{-}chip}$表示访问一次内存的平均时间,f_0是测试的基准频率.可以看出:$CPI_{on\text{-}chip}$对于特定的程序和系统是一个定值,不会随频率而发生变化.

于是在基准频率f_0下,对某个程序运行一遍,就可以抓取到对应的程序特征,从而获得相关的参数,进而建立起程序的性能预测模型.

综上,我们的性能模型为

$$T=I\cdot r\cdot t_{off\text{-}chip}+I\cdot(1-r)\cdot CPI_{on\text{-}chip}\cdot\frac{1}{f} \tag{2.12}$$

其中,T表示程序总的执行时间,I表示的是程序的指令总数,r表示的是最后一级缓存未命中的指令占总指令数目的比例,$t_{off\text{-}chip}$表示访问一次内存的平均时间,$CPI_{on\text{-}chip}$指的是片上完成指令的平均CPI,f是CPU的频率.模型中的参数I,r是在基准频率f_0下测试得到的;$t_{off\text{-}chip}$是一个平台相关的参数,用Memory Latency Checker测试得到;$CPI_{on\text{-}chip}$则由公式(2.11)计算得到.

由此可知:程序的性能与频率f之间并不是简单的线性关系,只有其中片上指令的执行时间与频率f呈线性关系.

3 模型验证

在我们的实验平台上,使用上述的建模方法,针对SPEC CPU 2006中的29个应用程序得到了它们的性能预测模型.本节首先介绍我们的实验平台,然后对所获得的模型进行验证并分析模型的误差,并且为了验证模型的通用性,在其他实验平台上对该模型做出了实验验证.

3.1 实验平台简介

我们的实验平台包含一个Intel XeonCPU E5-2620@2.00GHz的CPU芯片,含有6个计算核心,每个核心含有两个线程,带有32KB私有的D-cache和32KB私有的I-cache,256KB的私有*L2*缓存以及15MB的共享*L3*缓存,还有16GB的DDR3内存.实验平台参数见表1.

Table 1 Experimental environment parameters

表1 实验环境参数

参数名称	参数值
处理器	Intel Xeon CPU E5-2620
处理器频率变化范围	1.2GHz–2.0GHz
操作系统	Linux 2.6.32
编译器	GCC 4.4.7
基准程序	SPEC CPU 2006

3.2 获取实例模型

公式(2.12)给出了应用程序的性能预测模型,我们使用前一节提到的实验方法,为SPEC CPU 2006中的应用构建了其对应的性能模型.

针对任意一个给定的应用程序*A*,在基准频率f_0下运行一遍,利用VTune监测工具记录下*A*的*INST_RETIRED.ANY*和*MEM_LOAD_RETIRED.LLC_MISS*这两个硬件事件的值以及程序*A*的运行时间,再用Memory Latency Checker测出该硬件平台下的$t_{off\text{-}chip}$值,即可得到上述模型中的各项参数值,代入后就可以得到程序*A*在该平台下的性能预测函数.为了更加清晰地展现预测过程,我们以429.mcf为例.

在我们的实验平台下,测出$t_{off\text{-}chip}$的值为91ns.以f_0=1.2GHz为基准频率,在此频率下利用VTune监测工具

测得各参数的值.表 2 给出了实验得到的数据及相关的参数值.

Table 2 Model parameters for 429.mcf

表 2 429.mcf 的模型参数

参数名称	参数值
INST_RETIRED.ANY	284121×2000000
MEM_LOAD_RETIRED.LLC_MISS	83842×10000
t_0 (s)	552.27
I(指令总数)	284121×2000000
n(缓存缺失次数)	83842×10000
$r=n/I$(片外完成指令比例)	0.001 475 463
$CPI_{on\text{-}chip}$(片上完成指令平均 CPI)	1.006 6

代入模型并由公式(1.2)可以得到归一化的程序性能为

$$P=\frac{552.27}{\frac{571.1685}{f}+76.2962}.$$

这就是 429.mcf 在该平台下的性能预测模型,利用这个模型,我们可以预测任意频率下的执行时间和归一化性能.

表 3 给出了在该平台下,6 个 SPEC CPU 2006 中的测试程序的性能预测模型,表中的平均误差是指根据该预测模型得到的各个频率下的归一化执行效率与实验测得的归一化执行效率相比的平均误差.可以看到,误差都非常小,表示该模型的预测十分精确.

Table 3 Performance prediction model for applications in SPEC CPU 2006

表 3 SPEC CPU 2006 中的应用程序的性能预测模型

测试程序	预测函数	平均误差(%)
403.gcc	$P=\frac{670.70}{\frac{737.5826}{f}+56.0478}$	1.57
444.namd	$P=\frac{1252.05}{\frac{1501.9}{f}+0.5032}$	0.14
450.soplex	$P=\frac{274.56}{\frac{286.0825}{f}+36.1579}$	2.64
453.povray	$P=\frac{598.30}{\frac{442.8976}{f}+0.00091}$	0.67
471.omnetpp	$P=\frac{586.42}{\frac{477.6534}{f}+188.3755}$	5.67
483.xalancbmk	$P=\frac{615.98}{\frac{724.3608}{f}+12.3460}$	0.52

3.3 预测模型预测精度

本节使用典型的 SPEC 2006 应用来验证我们所提模型的精确性,并探讨了这一预测模型与最后一级缓存缺失率的关系.实验结果表明:这一模型非常精准,预测平均误差为 1.34%.进一步地,又探讨了共享缓存缺失率与应用对频率变化的敏感程度之间的关系.

图 5 给出了 429.mcf 在各个频率下预测的相对于基准频率 1.2GHz 的归一化性能和实验测得的归一化性能.图 5 中横轴代表不同的处理器频率,纵轴代表归一化的程序性能,绿色柱子是测得的程序性能,蓝色柱子是我们模型预测的结果,黄色柱子是简单线性模型的预测结果.从图中可以看出:预测结果与实验结果非常接近,相比

于简单线性模型有非常大的进步.如图中 1.8GHz,实验测得的归一化性能为 1.429,预测性能为 1.432,误差仅为 0.28%;简单线性模型为 1.5,误差高达 5.68%.本文的预测模型对于这一应用程序在所有频率下的平均误差仅为 2.55%.

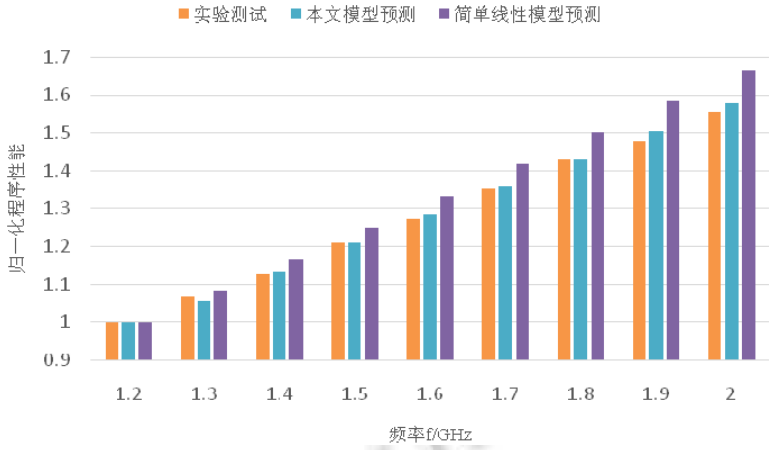


Fig.5 Prediction precision for benchmark 429.mcf

图 5 429.mcf 的实验测试性能与预测性能对比

为了验证模型的准确性,我们对 SPEC CPU 2006 中的所有测试程序都进行了预测精度的检查,结果如图 6 所示.图 6 给出了表 3 中的 6 个应用程序的预测函数曲线与测试得到的性能曲线,在每个子图中:横轴表示处理器的主频,单位是 GHz;纵轴代表根据第 1 节的公式(1.2)和第 2 节的公式(2.12)计算得到的归一化后的程序性能.

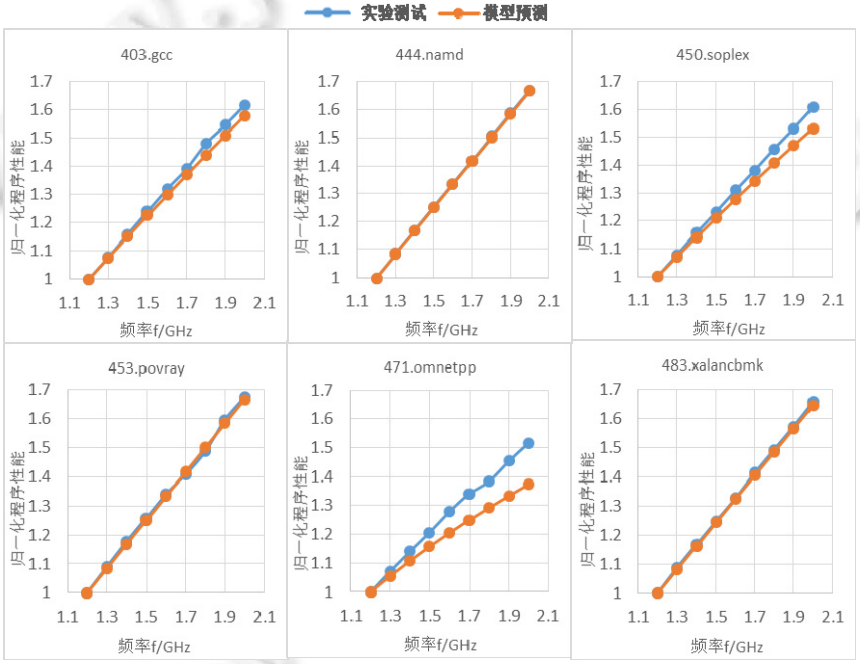


Fig.6 Prediction precision for six SPEC CPU 2006 applications listed in Table 3

图 6 表 3 中 6 个 SPEC CPU 2006 应用的预测精度

从图 6 可以看出,每个程序对应的两条曲线的重合度非常高,说明我们的预测十分接近实际运行结果.实验

结果表明:预测模型与实验测试得到的归一化性能相比的误差范围是 0.02%~7.62%,平均误差为 1.34%.

图 7 探讨了应用程序对频率变化的敏感性与共享缓存未命中率的关系.图 7 中给出了各个程序的性能预测曲线,横轴代表处理器的主频,单位是 GHz;纵轴代表根据第 1 节的公式(1.2)和第 2 节的公式(2.12)计算得到的归一化后的程序性能.图 7 反映了程序归一化后的程序性能和频率之间的对应关系以及对应的缓存缺失指令比例 r,可以看出:r 越大,其曲线的斜率就越小,性能的提升随着频率的变化幅度就越小.这说明,程序的性能随着频率的增大而提升的幅度与缓存缺失指令的比例有着负相关的关系,与理论模型中的分析互相印证,验证了模型中使用缓存缺失指令比例 r 作为描述程序性能对频率敏感性的程序特征的正确性.

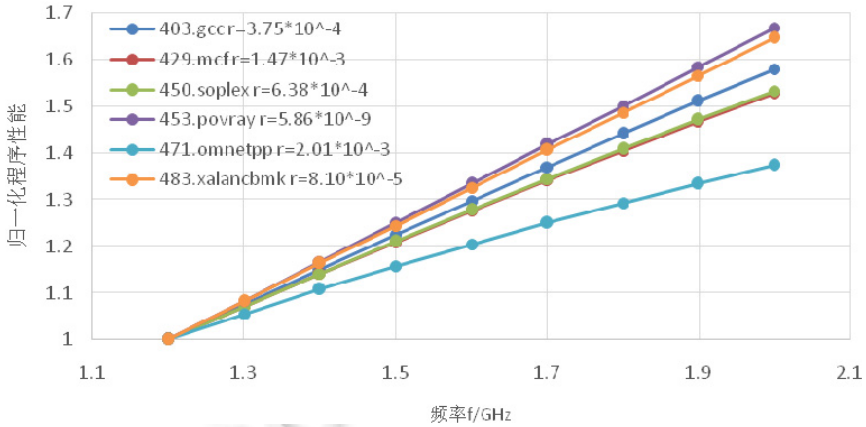


Fig.7 Relationship between performance prediction model and cache missrate

图 7 性能预测曲线与缓存缺失比例的关系

3.4 预测误差分析

尽管我们的预测模型与实验测得的结果相差很小,但仍然有一些误差,最大的平均误差超过了 5.0%.而在某个应用程序的某个频率下,它的误差可能更高,比如 459.GemsFDTD 在频率为 2.0GHz 下的实验,测得的归一化执行效率与预测的归一化执行效率相比误差达到了 9.55%,这是我们的实验数据中最大的误差值.

造成误差的原因主要有两个:一个是本文在建立抽象模型时没有考虑到在现行的复杂的体系结构层次中存在着普遍的并行性,比如内存级并行等各种复杂的并行设计中的多发射和超标量,这些复杂的结构使得本文的模型在许多细节的刻画上并不十分精确,只是在整体上给出了一个大致的预测,从而导致预测模型具有一定的系统误差;另一个是获取的模型参数不够精确,在模型中,访问内存的时间 $t_{off\text{-}chip}$ 使用 Memory Latency Checker 对平台进行测试而得到,实际上这是一个测试得到的平均值,每次访存的时间实际上都是不确定的,很难得到一个十分精确的预测值,所以这种做法会对模型带来一些误差.另外,我们使用硬件事件 *MEM_LOAD_RETIRED.LLC_MISS* 作为访问内存的次数,然而这个时间其实表示的仅仅是到内存中取数据的指令数,而内存访问还应包括数据写回内存的次数.尽管大部分的写回操作并不影响程序的执行时间,但在读操作与写操作有依赖的情况下仍然会有部分的误差,所以我们在建立模型的过程中使用 *MEM_LOAD_RETIRED.LLC_MISS* 所计算的片外完成指令比例仅仅是一个估计值,并不能完全准确地体现程序的访存次数.这些问题导致了预测模型会存在一些实验误差.

3.5 模型的通用性

为了确定本文中建立的程序性能预测模型对于大部分的硬件平台都广泛适用,我们在另外两个不同的硬件平台上用 SPEC 2006 的应用程序进行了相同的测试;为了确定该模型对于数据中心中不同类型的各种应用程序都能广泛使用,我们在同一平台下用 PARSEC 中的应用程序进行了测试,实验结果都比较符合模型的预测.

3.5.1 不同的硬件平台

为了测试不同的 CPU 硬件平台下该模型的适用性,本文选择了一个同样是 Intel 的 CPU 硬件平台和一个 AMD 的 CPU 硬件平台下进行实验测试.

第 1 个平台使用的是 Intel Xeon CPU E5645@2.40GHz 的 CPU 芯片,含有 2 个物理 CPU,每个 CPU 含有 6 个计算核心,每个核心含有 1 个线程,带有 64KB 的 *L*1 缓存,其中包括 32KB 私有的 D-cache 和 32KB 私有的 I-cache,256KB 的私有 *L*2 缓存,以及 12MB 的共享 *L*3 缓存,还有 8GB 的 DDR3 内存.在该平台上,对 SPEC 2006 中的 29 个应用程序进行了性能预测,并对这 29 个应用程序在各个频率下的性能进行了测试,用预测的性能与实际运行得到的性能相比,得到了在该平台下各个程序的预测误差,误差范围在 0.03%~6.67%,平均误差为 1.23%.而直接用第 2 节提到的简单线性模型预测的平均误差为 7.73%,最大误差甚至高达 23.5%,从而表明本文模型的准确性.

第 2 个平台使用的是 Quad-Core AMD Opteron Processor 8374 HE 的 CPU 芯片,含有 4 个物理 CPU,每个 CPU 含有 4 个计算核心,每个核心含有 1 个线程,带有 64KB 私有的 D-cache 和 64KB 私有的 I-cache,512KB 的私有 *L*2 缓存以及 6MB 的共享 *L*3 缓存,还有 64GB 的 DDR3 内存.在该平台上,对 SPEC 2006 中的部分应用程序进行了性能预测,并且对这些应用程序在各个频率下的性能进行了测试,用模型预测的性能与实际运行得到的性能相比,得到了在该平台下各个程序的预测误差,平均误差为 2.78%.而简单线性模型的平均误差高达 8.5%,同样说明了本文模型的准确性.

在这两个不同平台下的扩展实验的结果都比较符合模型的预期效果,并且预测精度远高于简单线性模型的预测精度,验证了该预测模型的平台可扩展性.

3.5.2 不同的应用程序

PARSEC[45]是普林斯顿大学在 2008 年提出的新的性能测试程序集,最新的研究成果表明[46],它们代表了一类典型的数据中心应用,本文选取了 PARSEC 中的 canneal,fluidanimate,facesim,streamcluster[47,48]这 4 个应用程序,在第 3 节提到的 Intel CPU 硬件平台下对该模型进行了验证.用模型预测的结果与各个频率下测试出来的实际性能相对比,得到了在该平台下各个程序的预测误差,平均误差为 2.99%,验证了该模型对于不同类型的应用程序都适用这一结论.

4 总结与展望

本文从当前体系结构逐级增大的访问延迟这一特征入手,依据访问资源的层次不同,将程序的指令分成了片上指令和片外指令两个部分,其中,片上指令的完成时间和处理器频率呈线性关系,片外指令的完成时间与频率无关,并据此提出了新的量化模型来预测应用程序在任意频率下的性能.实验结果表明:我们的预测模型相当精准,平均预测误差小于 1.34%,为性能和功耗的协同优化提供了空间.

该模型的建立,使得精确的预测程序性能成为可能,也为数据中心功耗和性能的优化提供了更多的空间.比如,数据中心的很多应用具有服务质量的要求,在我们所提模型的指导下,就可以选择满足程序的性能需求的最低频率,从而优化数据中心的能耗表现.

关于未来的工作,我们一方面将进一步扩大模型的适用范围,如数据中心中典型的多应用混合场景和处理器的 Turbo Boost 技术结合时的性能变化预测、内存的 DVFS 机制和不同的处理器体系结构对预测模型的影响等.另一方面,我们将扩大研究内容,以研究更多因素对程序性能的直接影响,如网络 I/O、磁盘 I/O 等.

References:

[1] Patterson MK, Costello D, Grimm P, Loeffler M. Data center TCO: A comparison of high-density and low-density spaces. In: Proc. of the Thermal Challenges in Next Generation Electronic Systems (THERMES 2007). Santa Fe, 2007. 42–49.

[2] Greenberg A, Hamilton J, Maltz DA, Patel P. The cost of a cloud: Research problems in data center networks. ACM SIGCOMM Computer Communication Review, 2009,39(1):68–73. [doi: 10.1145/1496091.1496103]

[3] Govindan S, Sivasubramaniam A, Urgaonkar B. Benefits and limitations of tapping into stored energy for datacenters. In: Proc. of the 38th Annual Int'l Symp. on Computer architecture (ISCA 2011). New York: IEEE, 2011. 341−351. [doi: 10.1145/2024723.2000105]

[4] Macken P, Degrauwe M, Van Paemel M, Oguey H. A voltage reduction technique for digital systems. In: Proc. of the 37th IEEE Int'l Solid-State Circuits Conf. on Digest of Technical Papers (ISSCC). New York: IEEE, 1990. 238−239. [doi: 10.1109/ISSCC.1990.110213]

[5] Isci C, Buyuktosunoglu A, Cher CY, Bose P, Martonosi M. An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. In: Proc. of the 39th Annual IEEE/ACM Int'l Symp. on Microarchitecture. New York: IEEE, 2006. 347−358. [doi: 10.1109/MICRO.2006.8]

[6] Kim W, Gupta MS, Wei GY, Brooks D. System level analysis of fast, per-core DVFS using on-chip switching regulators. In: Proc. of the 2008 IEEE 14th Int'l Symp. on High Performance Computer Architecture (HPCA 2008). New York: IEEE, 2008. 123−134. [doi: 10.1109/HPCA.2008.4658633]

[7] Semeraro G, Magklis G, Balasubramonian R, Albonesi DH, Dwarkadas S, Scott ML. Energy-Efficient processor design using multiple clock domains with dynamic voltage and frequency scaling. In: Proc. of the 8th Int'l Symp. on High Performance Computer Architecture (HPCA-8 2002). New York: IEEE, 2002. 29−40. [doi: 10.1109/HPCA.2002.995696]

[8] Wu Q, Juang P, Martonosi M, Clark DW. Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors. In: Proc. of the 11th Int'l Symp. on High-Performance Computer Architecture (HPCA-11 2005). New York: IEEE, 2005. 178−189. [doi: 10.1109/HPCA.2005.43]

[9] Lee WY, Ko YW, Lee H, Kim H. Energy-Efficient scheduling of a real-time task on DVFS-enabled multi-cores. In: Proc. of the 2009 Int'l Conf. on Hybrid Information Technology (ICHIT 2009). New York: ACM Press, 2009. 273−277. [doi: 10.1145/1644993.1645046]

[10] Lo D, Song T, Suh GE. Prediction-Guided performance-energy trade-off for interactive applications. In: Proc. of the 48th Int'l Symp. on Microarchitecture (MICRO 48). New York: ACM Press, 2015. 508−520. [doi: 10.1145/2830772.2830776]

[11] Zhu H, Erez M. Dirigent: Enforcing QoS for latency-critical tasks on shared multicore systems. In: Proc. of the 21st Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2016). New York: ACM Press, 2016. 33−47. [doi: 10.1145/2872362.2872394]

[12] Hsu CH, Zhang Y, Laurenzano MA, Meisner D. Adrenaline: Pinpointing and reining in tail queries with quick voltage boosting. In: Proc. of the 2015 IEEE 21st Int'l Symp. on High Performance Computer Architecture (HPCA 2015). New York: IEEE, 2015. 271−282. [doi: 10.1109/HPCA.2015.7056039]

[13] Vamanan B, Sohail HB, Hasan J, Vijaykumar TN. Timetrader: Exploiting latency tail to save datacenter energy for online search. In: Proc. of the 48th Int'l Symp. on Microarchitecture (MICRO 48). New York: ACM Press, 2015. 585−597. [doi: 10.1145/2830772.2830779]

[14] Lo D, Cheng L, Govindaraju R, Ranganathan P, Kozyrakis C. Heracles: Improving resource efficiency at scale. ACM Sigarch Computer Architecture News, 2015,43(3):450−462. [doi: 10.1145/2872887.2749475]

[15] Xie F, Martonosi M, Malik S. Compile-Time dynamic voltage scaling settings: Opportunities and limits. In: Proc. of the ACM SIGPLAN 2003 Conf. on Programming Language Design and Implementation (PLDI 2003). New York: ACM, 2003. 49−62. [doi: 10.1145/780822.781138]

[16] Wu Q, Martonosi M, Clark DW, Jin L. A dynamic compilation framework for controlling microprocessor energy and performance. In: Proc. of the 38th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO 38). New York: IEEE Computer Society, 2005. 271−282. [doi: 10.1109/MICRO.2005.7]

[17] Wang YH, Wang KL, Sun XK, Zhang DS, Wu F. Research and implementation of DVFS energy saving technologies based on CPUfreq. Computer Measurement & Control, 2016,24(2):151−154 (in Chinese with English abstract).

[18] Miftakhutdinov R, Ebrahimi E, Patt YN. Predicting performance impact of DVFS for realistic memory systems. In: Proc. of the 45th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO 45). New York: IEEE Computer Society, 2012. 155−165. [doi: 10.1109/MICRO.2012.23]

[19] Keramidas G, Spiliopoulos V, Kaxiras S. Interval-Based models for run-time DVFS orchestration in superscalar processors. In: Proc. of the 7th ACM Int'l Conf. on Computing Frontiers (CF 2010). New York: ACM Press, 2010. 287−296. [doi: 10.1145/1787275.1787338]

[20] Liu NT, Weng Y, Lin Y, Zhang WR,Wei ZL, Shao K. Dynamic power management scheme based on software behavior prediction. Computer Engineering, 2015,41(6):269−273, 279 (in Chinese with English abstract).

[21] Hu ZG, Ouyang C, Yan CK. Resource load balancing method for energy-consumption reducingin cloud environment. Computer Engineering, 2012,38(5):53−55 (in Chinese with English abstract).

[22] Goh LK, Veeravalli B, Viswanathan S. Design of fast and efficient energy-aware gradient-based scheduling algorithms heterogeneous embedded multiprocessor systems. IEEE Trans. on Parallel and Distributed Systems, 2009,20(1):1−12. [doi: 10.1109/TPDS.2008.55]

[23] Kim KH, Beloglazov A, Buyya R. Power-Aware provisioning of virtual machines for real-time cloud services. Concurrency and Computation: Practice and Experience, 2011,23(13):1491−1505. [doi: 10.1002/cpe.1712]

[24] Von Laszewski G, Wang L, Younge AJ, He X. Power-Aware scheduling of virtual machines in DVFS-enabled clusters. In: Proc. of the 2009 IEEE Int'l Conf. on Cluster Computing and Workshops (CLUSTER 2009). New York: IEEE, 2009. 1−10. [doi: 10.1109/CLUSTR.2009.5289182]

[25] Wang L, Von Laszewski G, Dayal J, Wang F. Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS. In: Proc. of the 10th IEEE/ACM Int'l Conf. on Cluster Cloud and Grid Computing (CCGrid 2010). New York: IEEE, 2010. 368−377. [doi: 10.1109/CCGRID.2010.19]

[26] Zhang XQ, He ZT, Li CL, Zhang HX, Qian QF. Research on energy saving algorithm of datacenter in cloud computing system. Application Research of Computers, 2013,30(4):961−964 (in Chinese with English abstract).

[27] Chen Y, Jia GY, Li X, Zhang HP. Task behavior based on DVFS mechanism. Computer Systems and Applications, 2013,22(10): 1−7 (in Chinese with English abstract).

[28] Beloglazov A, Buyya R. Energy efficient allocation of virtual machines in cloud data centers. In: Proc. of the 10th IEEE/ACM Int'l Conf. on Cluster Cloud and Grid Computing (CCGrid 2010). New York: IEEE, 2010. 577−578. [doi: 10.1109/CCGRID.2010.45]

[29] Nathuji R, Schwan K. VirtualPower: Coordinated power management in virtualized enterprise systems. In: Proc. of the 21st ACM SIGOPS Symp. on Operating Systems Principles (SOSP 2007). New York: ACM SIGOPS Operating Systems Review, 2007. 265−278. [doi: 10.1145/1323293.1294287]

[30] Stoess J, Lang C, Bellosa F. Energy management for hypervisor-based virtual machines. In: Proc. of the USENIX Annual Technical Conf. New York: IEEE, 2007. 1−14.

[31] Van HN, Tran FD, Menaud JM. Performance and power management for cloud infrastructures. In: Proc. of the 2010 IEEE 3rd Int'l Conf. on Cloud Computing (CLOUD). New York: IEEE, 2010. 329−336. [doi: 10.1109/CLOUD.2010.25]

[32] Zeng ZB, Xu L. Energy efficiency virtual resource allocation stratage for cloud computing. Computer Systems and Applications, 2011,20(12):55−60 (in Chinese with English abstract).

[33] Liu PC. Research on virtual machine live migration in cloud computing [MS. Thesis]. Shanghai: Fudan University, 2009 (in Chinese with English abstract).

[34] Wu Q, Xiong GZ. Adaptive dynamic power management for non-stationary self-similar requests. Ruan Jian Xue Bao/Journal of Software, 2005,16(8):1499−1505 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1499.htm

[35] Liu H. Research on WMSNs key technologies in energy conservationin cloud computing center [Ph.D. Thesis]. Dalian: Dalian University of Technology, 2011 (in Chinese with English abstract).

[36] Tan YM, Zeng GS, Wang W. Policy of energy optimal management for cloud computing platform with stochastic tasks. Ruan Jian Xue Bao/Journal of Software, 2012,23(2):266−278 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/4143.htm [doi: 10.3724/SP.J.1001.2012.04143]

[37] Femal ME, Freeh VW. Boosting data center performance through non-uniform power allocation. In: Proc. of the 2nd Int'l Conf. on Autonomic Computing (ICAC 2005). New York: IEEE, 2005. 250−261. [doi: 10.1109/ICAC.2005.17]

[38] Gandhi A, Harchol-Balter M, Das R, Lefurgy C. Optimal power allocation in server farms. ACM SIGMETRICS Performance Evaluation Review, 2009,37(1):157−168. [doi: 10.1145/2492101.1555368]

[39] Cochran R, Hankendi C, Coskun AK, Reda S. Pack & Cap: Adaptive DVFS and thread packing under power caps. In: Proc. of the 44th Annual IEEE/ACM Int'l Symp. on Microarchitecture (MICRO-44 2011). New York: ACM Press, 2011. 175−185. [doi: 10.1145/2155620.2155641]

[40] Rossi FD, Storch M, de Oliveira I, De Rose CAF. Modeling power consumption for DVFS policies. In: Proc. of the 2015 IEEE Int'l Symp. on Circuits and Systems (ISCAS). New York: IEEE, 2015. 1879−1882. [doi: 10.1109/ISCAS.2015.7169024]

[41] Akram S, Sartor JB, Eeckhout L. DVFS performance prediction for managed multithreaded applications. In: Proc. of the 2016 IEEE Int'l Symp. on Performance Analysis of Systems and Software (ISPASS). New York: IEEE, 2016. 12−23. [doi: 10.1109/ISPASS.2016.7482070]

[42] Intel 64 and IA-32 architectures optimization reference manual. 2016, Chapter 2, 2.2.4. http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf

[43] Intel. Intel VTune amplifier XE. 2013. https://software.intel.com/en-us/intel-vtune-amplifier-xe

[44] Intel. Intel memory latency checker v3.1. 2016. https://software.intel.com/en-us/articles/intelr-memory-latency-checker

[45] Bienia C, Kumar S, Singh JP, Li K. The PARSEC benchmark suite: Characterization and architectural implications. In: Proc. of the 17th Int'l Conf. on Parallel Architectures and Compilation Techniques (PACT 2008). New York: ACM Press, 2008. 72−81. [doi: 10.1145/1454115.1454128]

[46] Zhu H, Erez M. Dirigent: Enforcing QoS for latency-critical tasks on shared multicore systems. In: Proc. of the 21st Int'l Conf. on Architectural Support for Programming Languages and Operating Systems (ASPLOS 2016). New York: ACM Press, 2016. 33−47. [doi: 10.1145/2872362.2872394]

[47] Wang W, Dey T, Mars J, Tang LJ, Davidson JW, Soffa ML. Performance analysis of thread mappings with a holistic view of the hardware resources. In: Proc. of the 2012 IEEE Int'l Symp. on Performance Analysis of Systems & Software (ISPASS). New York: IEEE, 2012. 156−167. [doi: 10.1109/ISPASS.2012.6189222]

[48] Park H, Baek S, Choi J, Lee D, Noh SH. Regularities considered harmful: Forcing randomness to memory accesses to reduce row buffer conflicts for multi-core, multi-bank systems. ACM SIGARCH Computer Architecture News, 2013,41(1):181−192. [doi: 10.1145/2499368.2451137]

附中文参考文献:

[17] 王益涵,王凯林,孙宪坤,张冬松,吴飞.基于 CPUfreq 的 DVFS 节能技术的研究与实现.计算机测量与控制,2016,24(2):151−154.

[20] 刘念唐,翁宇,林雨,张文睿,韦志磊,邵堃.基于软件行为预测的动态电源管理方案.计算机工程,2015,41(6):269−273,279.

[21] 胡志刚,欧阳晟,阎朝坤.云环境下面向能耗降低的资源负载均衡方法.计算机工程,2012,38(5):53−55.

[26] 张小庆,贺忠堂,李春林,张恒喜,钱琼芬.云计算系统中数据中心的节能算法研究.计算机应用研究,2013,30(4):961−964.

[27] 陈云,贾刚勇,李曦,张海鹏.基于任务行为分析的 DVFS 机制.计算机系统应用,2013,22(10):1−7.

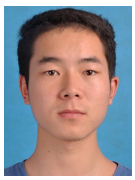
[32] 曾智斌,许力.云计算中高能效的虚拟资源分配策略.计算机系统应用,2011,20(12):55−60.

[33] 刘鹏程.云计算中虚拟机动态迁移的研究[硕士学位论文].上海:复旦大学,2009.

[34] 吴琦,熊光泽.非平稳自相似业务下自适应动态功耗管理.软件学报,2005,16(8):1499−1505. http://www.jos.org.cn/1000-9825/16/1499.htm

[35] 刘航.WMSNs 在云计算中心节能减排中的关键技术研究[博士学位论文].大连:大连理工大学,2011.

[36] 谭一鸣,曾国荪,王伟.随机任务在云计算平台中能耗的优化管理方法.软件学报,2012,23(2):266−278. http://www.jos.org.cn/1000-9825/4143.htm [doi:10.3724/SP.J.1001.2012.04143]



李登辉(1993－),男,湖北荆门人,学士,主要研究领域为并行计算,并行编译,并行编程环境.

崔慧敏(1979－),女,博士,副研究员,CCF 专业会员,主要研究领域为并行计算,并行编译,并行编程.

赵家程(1989－),男,学士,CCF 学生会员,主要研究领域为并行计算,并行编译,并行编程环境.

冯晓兵(1969－),男,博士,教授,博士生导师,CCF 杰出会员,主要研究领域为先进编译技术及相关工具环境.