有效的低功耗编译优化方法:部件使用局部化*

易会战+, 杨学军

(国防科学技术大学 计算机学院,湖南 长沙 410073)

An Effective Method of Low-Power Compilation Optimization: Localizing the Use of System Units

YI Hui-Zhan+, YANG Xue-Jun

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: huizhanyi@yahoo.com.cn, http://www.nudt.edu.cn

Received 2003-09-12; Accepted 2004-07-06

Yi HZ, Yang XJ. An effective method of low-power compilation optimization: Localizing the use of system units. *Journal of Software*, 2004,15(10):1451~1460.

http://www.jos.org.cn/1000-9825/15/1451.htm

Abstract: Recently, many research activities have targeted at reducing energy consumption of computer systems through software optimizations. By using the functions of dynamic voltage scaling (DVS) and turning off unused system units (TOSU), some software techniques can be used to reduce the wasted energy consumption, and one of the techniques is the compiler-directed method. Many hardware details are complex and fuzzy about DVS and TOSU, and so the paper has established an analytic model for compilation. Based on analyzing the model, the paper gives the idea about localizing the use of system units. If the systems have functions of DVS and TOSU, localizing the use of system units is an effective method of low-power compilation optimization.

Key words: compilation optimization; low-power; localizing; dynamic voltage scaling; turning off unused system units

摘 要: 使用软件技术优化系统能量正得到更多的关注.利用系统的动态电压缩放和功能部件关闭的功能为减少冗余能量消耗提供了优化的新途径,而编译指导的动态电压缩放(dynamic voltage scaling,简称 DVS)和功能部件关闭(turning off unused system units,简称 TOSU)是软件优化方法之一.DVS 或 TOSU 涉及到很多技术细节.抽象出可以用于编译研究的分析模型,根据对模型的研究,提出了部件使用局部化的概念.部件使用局部化在存在 DVS 和 TOSU 的技术支持下,是有效的低功耗编译优化方法.

关键词: 编译优化;低功耗;局部化;动态电压缩放;功能部件关闭

中图法分类号: TP314 **文献标识码**: A

* Supported by the National High-Tech Research and Development Plan of China under Grant No.2002AA1Z2105 (国家高技术研究发展计划(863))

作者简介: **易会战**(1976－),男,河北肃宁人,博士生,主要研究领域为低功耗编译优化,计算机体系结构;**杨学军**(1963－),男,博士,教授,主要研究领域为并行体系结构,并行编译,并行操作系统.

计算机系统的能量消耗越来越成为系统发展需要解决的问题之一.嵌入式系统和移动设备很早就面临能量消耗问题[1,2],在这样的系统中,电池的供电时间是衡量系统的重要参数.

高性能计算机系统和大型服务器产生了大量的能量消耗.高性能计算机系统往往由大量的处理节点、大规模的存储系统和 I/O 设备构成,消耗了大量能量[3].

能量优化对保证系统可靠运行有重要作用.为了跟踪 Moore 定律,计算机产业的支撑技术——互补金属氧化物半导体(CMOS)的工艺不断缩放,芯片的能量密度越来越大[4],这增加了封装和制冷的代价,能量问题成为芯片设计必须解决的问题.大型服务器系统的电力和制冷问题经常是许多结点失效的原因之一[5].

线路层、门层、体系结构层开发了很多方法减少芯片内部的冗余能量消耗,这包括选择适当的晶体管大小,选择低功耗的逻辑风格、多阈值电压[6]、动态电压缩放技术[7]、冗余部件关闭技术[8].动态电压缩放和冗余部件关闭技术是在系统的较高层次完成的节能技术,可以通过硬件方法来实现,同时通过软件来完成部件的管理.

早期的软件低功耗管理主要由操作系统完成,操作系统负责将空转的 CPU、内存、磁盘系统等关闭以达到节能的目的[9,10].随着硬件技术的发展,产生了一些细粒度的节能技术,例如,关闭 cache 行,关闭多 bank 内存系统,关闭某些 CPU 部件.完成这些细粒度的控制需要对程序本身的特点有更全面而具体的了解,所以编译指导的低功耗技术得以研究,例如,编译指导的动态电压缩放技术[11]、编译指导的 cache 能量优化[12]、编译指导的适应性处理节点策略[13]、编译指导的内存系统管理策略[14].

尽管如此,人们仍然很难清晰地说出低功耗编译优化技术存在哪些指导性的优化方向,已经出现的一些优化方法也很难形成一个完整的体系,本文试图解决上述问题.

本文第 1 节对芯片的能量消耗和硬件层为软件层提供的一些能量优化途径进行回顾和分析.第 2 节给出进行能量有效性研究的尺度.第 3 节抽象出在存在动态电压缩放和显示关闭功能部件支持下的分析模型.第 4 节根据模型对编译指导的低功耗优化方法进行分析,指出开发部件使用局部化的重要作用.最后是结论和未来的工作.

1 芯片系统的能量消耗和硬件层为软件提供的优化途径

CMOS 技术是计算机系统的基础,其线路的能量消耗可以分为动态能量消耗、短线能量消耗和静态能量消耗这 3 部分.

静态 CMOS 反向器的动态能量消耗公式为

$$P_d = \alpha f C V_{dd}^2 \tag{1}$$

其中 V_{dd} 为CMOS供电电压,C 为门输出结点的总电容,f 为时钟频率,α 为输入结点每个时钟周期的变化概率.

短线能量消耗是当静态 CMOS 门转换时,在短时间内,两个晶体管同时导电产生.这期间,在 V_{dd} 和地面之间会存在短时间的电流.随着工艺技术的发展,短线电流呈现下降趋势.

理想的 CMOS 线路的静态能量消耗等于 0.实际上,通过晶体管反向有偏二极管形成的结,在源极或漏极和基底之间总是存在泄漏电流的情况.静态能量消耗为

$$P_s = I_{leakage} V_{dd} \tag{2}$$

一般来说,系统中的动态能量消耗占主要部分,随着工艺的缩放,泄漏电流的比例逐渐增大.如果不使用任何泄漏控制机制,未来的工艺中动态能量消耗和静态能量消耗比例基本相当.

计算机系统是由软件和硬件组成的系统,低功耗问题必须从软件和硬件两方面综合考虑.下面对一些可能用于软件优化的技术进行回顾和总结.

计算机系统包括中心处理器、主存和 I/O 设备.一般来说,磁盘设备的能量消耗要比主存和处理器的功耗大几个量级,低功耗的系统往往不使用磁盘系统.内存系统 DRAM 的能量消耗是处理器能量消耗的几十倍到几百倍[15].处理器内部的动态能量消耗又由时钟系统、数据路径、存储系统和控制 I/O 等组成.

下面是一些重要的体系结构层低功耗技术.

(1) 动态电压缩放(dynamic voltage scaling,简称 DVS)——降低电压

系统的动态功耗和电压成二次方关系,降低供应电压可以降低系统的动态功耗,动态电压缩放在系统运行时动态改变电压.一般可以设置几个离散电压值,软件可以根据需求在几个电压值之间进行动态调整.实用的处理器包括 Transmeta Crusoe,Intel Xscale 和 AMD K6 III+.电压切换存在一定的能量开销和时间开销.

(2) 时钟门(clock gating)——减少切换电容

时钟系统的能量消耗占 CPU 总功耗的很大一部分,减少时钟系统的切换电容对总功耗有很大的作用.一种实际有效的方法是划分时钟网络,在每个周期只允许必要的部分进行切换.这通过时钟门来实现.

使用时钟门关闭的部件一般不能及时恢复正常状态,并且时钟系统可能产生小故障,这是使用时钟门存在的问题.如何有效地使用时钟门关闭功能部件,如何及时地将关闭的功能部件恢复到正常状态以降低性能损失是软件需要解决的问题.

(3) 存储系统——减少切换电容

CPU 内部的 cache,TLB,分支缓存占能量消耗的很大部分,DRAM 的功耗又是 CPU 的几十倍,磁盘设备更是重要的能量消耗源.低功耗的存储系统对降低系统功耗有很大作用.除了传统的多运行模式磁盘、内存系统以外,很多新的硬件技术用来解决存储系统的运行时功耗:

新的 cache 技术.处理器的发展集成了越来越大的芯片内 cache,大的 cache 造成了大量的能量消耗.在保持程序性能的前提下,功耗最优的 cache 大小和结构随着负载的变化而变化.于是产生了可重配置的 cache[16]和动态关闭 cache 行的 cache[17],这些 cache 设计的主要目的是减少动态切换的电容量,降低功耗.

多 bank 的内存结构.为了降低访存的切换电容量,将存储结构划分为多个 bank,每次只访问部分部件,不使用的内存 bank 可以关闭.

这些动态的存储系统部件为存储系统的能量优化提出了新问题,如动态 cache 结构下,如何有效利用 cache,保证性能并提高能量效率?采用什么样的方法进行 cache 数据的映射?基于分页的操作系统如何有效利用多 bank 的内存系统?程序如何有效地局部化,利用多个内存 bank 降低功耗?

(4) 编码和缓存——减少切换因子

应用中很多计算存在重复部分,可以在功能部件中增加 cache,将计算的结果保存.如果又有同样操作数的计算,则直接使用原来的值.这种方法减少了切换活动,降低了功耗[18].

有些计算使用的操作数不需要很高的精度,低位部分就足够了,这样可以通过一些技术监测冗余的高位部分,避免高位部分的计算以降低功耗[19].

(5) 泄漏能量减少技术

泄漏能量消耗是今后工艺发展面临的重要问题之一,泄漏控制的主要方式[20]有:

A. 输入向量控制(IVC)

许多研究使用模型和算法估计电路的实际和最大泄漏能量,得到的结果表明,输入模式对线路的泄漏行为有很大影响,这是晶体管栈效应的结果.因为晶体管栈中设备的状态由对应的输入决定,这又对应于部件的输入信号.IVC 的目标是找到一种输入模式,该模式最大化栈中晶体管关闭的数目,最小化泄漏电流.

B. 增加阈值电压(MTCMOS,BBC)

这种方案需要一些工艺支持,能够改变某些晶体管的阈值电压.当转入低功耗模式时,通过提高晶体管源极到体的电压,单元中的晶体管阈值电压动态增加.高阈值电压导致泄漏能量降低,同时保持内存单元的状态.尽管内存单元的泄漏降低,增加线路供应电压增加了功耗,转换开销很大.

C. 关闭供应电压(power supply gating,简称 PSG)

能量供应门关闭能量供应,结果空转的部件不再消耗泄漏电流.能量供应门可以使用睡眠晶体管实现,一般可以每个门有一个睡眠晶体管,也可以实现更大粒度的管理.能量供应门对减少泄漏很有效,但会丢失存储单元中的内容.

D. 动态电压缩放

缩放内存单元的电压为 1.5 倍的阈值电压,这样内存单元的内容可以保留,同时由于在高性能工艺中的短信道效果,泄漏会随着电压缩放显著减少.对于 0.07μm 工艺,睡眠电压近似为 0.3V.动态电压缩放方法的能量状

态转换很快,转换延迟 0.28ns.该方法被使用在 drowsy cache[21]中.

减少泄漏能量的技术是今后低功耗研究的热点,上述各种泄漏控制方式适用于体系结构层,这为编译指导控制泄漏能量消耗提供了可行的途径.

2 能量有效性尺度

在实时系统中,需要的操作数是一定的,更大的吞吐量对系统没有什么作用,这时每个操作使用的能量是有效的尺度,

$$Metric_{Fix}=\frac{Power}{Throughput}=\frac{Energy}{Operation} \tag{3}$$

能量延迟乘积由 Ricardo[22]提出,用来反映能量有效的处理器性能,

$$Metric_{energy*delay}=\frac{Energy}{Operation}\cdot\frac{t}{Operation}=\frac{Energy/t}{Operation^2/t^2}=\frac{Power}{Throughput^2} \tag{4}$$

能量延迟乘积越小,说明在相等的吞吐量下每个操作的能量越低,在固定的能量下,吞吐量越大.

Martin[23]认为,能量延迟乘积是不恰当的,因为能量近似为 $E=\frac{Cz\times V^2}{2}$,延迟为 $t=\frac{Cz}{Ku\times V}$,二者的乘积受电压变化的影响.他认为,任何随能量和延迟单调递增且对电压为不变量的尺度都可以用作衡量系统能量有效性的尺度,所以应该使用 Et^2 作为能量有效性的尺度.

3 低功耗的分析模型

编译技术生成节能的代码在 20 世纪 90 年代就已经有很多研究[24,25].传统的优化技术对低功耗优化并不是很有效,研究的结果往往是,性能最好的代码就是能量最有效的代码[26,27],只有一些面向减少内存访问的优化对能量是有效的[28].一些新技术的发展和新的体系结构的提出为编译技术提供了新的优化途径,例如,动态电压缩放技术可以有效利用程序运行期间各种部件使用不均衡的特点,通过降低电压减少功耗;时钟门技术可以动态关闭系统中的空转部件,这使得系统功耗进一步降低;而对泄漏问题的研究也产生了一些关闭系统部件降低泄漏的方法.编译技术采用各种方式来利用这些新技术,例如,Chung-Hsing Hsu[29]使用编译方法标记 CPU 可以降频运行的程序区域,对这些区域进行降低电压运行;H.S. Kim[30]基于程序基本块的 IPC 数动态调整系统中功能部件数目;V. Delaluz[31]对多 bank 内存的编译指导关闭内存 bank 进行了研究.

上述研究的一个共同点是利用了程序运行时对资源利用的不均衡性,使用一些方法对空转的部件采用节能策略进行处理.Jason Casmira[32]对这种开发的可能性进行了研究,发现程序运行期间存在大量的可松弛部分(slack),可以用来减少能量消耗.Jason Casmira 没有对各种编译策略或者什么样的编译策略会对程序的可松弛部分产生影响以及产生什么样的影响进行分析.

下面我们抽象出存在动态电压缩放或显示关闭功能部件功能等节能方法支持下的分析模型,并基于这一模型分析可能的编译优化方法和策略.

对于系统中的一个功能部件或者设备 c (c 代表各种可以独立控制的部件,这可能是一个运算部件,也可能是一个 cache 行或者内存 bank),它可以处于不同的功耗等级和性能等级.设功耗等级为

$$P_0^c,P_1^c,...,P_{(n-1)}^c,$$

性能等级为

$$S_0^c,S_1^c,...,S_{(n-1)}^c.$$

一般来说,不同功耗等级的功耗不同,功耗越低的功耗等级性能越低.对于处于不可用功耗等级的部件(例如处于关闭状态),令其相应的性能等级值为 0.

不同功耗等级间的切换一般存在一定的能量开销和性能损失,令功耗等级 $i\sim j$ 的切换能量开销为 $E_{i\to j}^c$,性能损失为 $t_{i\to j}^c$.

假定对设备 c,在系统运行期间采用了某种节能策略,那么对于运行在系统上的指令序列 K,在运行期间的不同时段 c 处于不同的功耗等级,设 k 时段使用 c 运行的指令数为 I_k^c,该时段大小为 t_k^c,

$$t_k^c=\begin{cases}\dfrac{I_k^c}{S^c_{x_k^c}}, & \text{如果} S^c_{x_k^c}\neq 0\\ \text{处于功耗等级} P^c_{x_k^c}\text{的时间}, & \text{如果} S^c_{x_k^c}=0\end{cases},$$

其中 x_k^c 代表 k 时段 c 的功耗等级数,公式中处于功耗等级 $P^c_{x_k^c}$ 的时间与指令序列中使用其他部件的指令相关,此时部件 c 没有被任何指令使用.

不考虑各种功耗模式切换的开销,指令序列 K 在运行期间对部件 c 的能量消耗为

$$E^c_{comp}=\sum_k t_k^c\cdot P^c_{x_k^c} \tag{5}$$

K 的运行时间为

$$t^c_{comp}=\sum_k t_k^c \tag{6}$$

将切换的能量开销和性能损失考虑进来,指令序列 K 在运行期间对部件 c 的能量消耗为

$$E^c=\sum_k t_k^c P^c_{x_k^c}+\sum_k E^c_{x_k^c\to x_{k+1}^c}=E^c_{comp}+E^c_{switch} \tag{7}$$

运行时间为

$$t=\sum_k (t_k^c+t^c_{x_k^c\to x_{k+1}^c})=\sum_k t_k^c+\sum_k t^c_{x_k^c\to x_{k+1}^c}=t^c_{comp}+t^c_{switch} \tag{8}$$

系统中一般存在多种类型的多个部件和设备,设部件为 c_l,每个部件都满足式(5)~式(8).指令序列 K 的总能量消耗为

$$E=\sum_l E^{c_l} \tag{9}$$

运行时间为 t.

对于存在时间限制的应用,$Metric_{Fix}$ 是衡量能量有效性的较好的尺度.而对于可以进行时间和能量取舍的应用,必须同时考虑完成应用消耗的能量和使用的时间,过高的能量消耗或者过长的时间使用都是不能接受的.能量延迟乘积尺度和 Martin 的 Et^2 尺度都考虑了这两方面的因素,但是对能量和时间的重视程度不同,Et^2 尺度体现了对时间更高的重视.实际上,

$$Metric_{energy*delay^x}=\frac{Energy}{Operation}\cdot\left(\frac{t}{Operation}\right)^x=\frac{Energy/t}{(Operation/t)^{x+1}}=\frac{Power}{Throughput^{x+1}} \tag{10}$$

综合了上述尺度,体现了对能量和时间的权衡.当 $x=1$ 时,上式就是能量延迟乘积尺度,当 $x=2$ 时就是 Et^2 尺度. $x\ (>0)$的取值越大,对时间越重视. $Metric_{energy*delay^x}$ 越小,说明系统在相等的吞吐量下每个操作的能量越低,在固定的能量下,吞吐量越大.

设指令序列 K 的操作个数为 O_K,则

$$Metric_{energy*delay^x}=\frac{E/t}{(O_K/t)^{x+1}}=\frac{Et^x}{(O_K)^{x+1}} \tag{11}$$

如果程序运行在一个可以调整电压和频率的系统上,假设动态功耗占统治地位,对 $Metric_{energy*delay^x}$ 的公式进行展开,

$$Metric_{energy*delay^x}=\frac{Et^x}{(O_K)^{x+1}}=\frac{\alpha CfV_{dd}^2\cdot\left(Cyclenum\cdot\dfrac{1}{f}\right)^{x+1}}{(O_K)^{x+1}}=\alpha Cf^{-x}V_{dd}^2\left(\frac{Cyclenum}{O_K}\right)^{x+1} \tag{12}$$

其中 $Cyclenum$ 为程序运行的周期数.

从式(12)可以看出,对于上述系统,为了提高效率,根据 x 的大小,应该使用使得 $f^{-x}V_{dd}^2$ 最小的系统配置.在以往的一些动态电压缩放的研究[10,33]中,假定性能和系统的时钟频率成正比,功耗与 CfV_{dd}^2 成正比.

所以,如果电压不变,频率变化,每个操作所使用的能量不变.要提高能量效率,必须降低电压运行.而使用 $Metric_{energy*delay^x}$ 尺度,结论是使得 $f^{-x}V_{dd}^2$ 最小.

$$E=\alpha CfV_{dd}^2t=\alpha CfV_{dd}^2\cdot\left(Cyclenum\cdot\frac{1}{f}\right)=\alpha CV_{dd}^2Cyclenum \tag{13}$$

$Cyclenum$ 与系统的电压和频率无关.要提高系统能量的有效性,要尽量减小 $Cyclenum$,这需要使得程序在系统上能够尽量平滑地运行,减小程序的驻留.这实际上用于衡量系统执行程序的有效性,很多传统的编译优化技术用于减少程序驻留,加快程序执行速度,与提高系统的能量有效性是一致的.对于 O_K,在一定时间内运行的程序操作个数越大,系统的吞吐量越大,则系统的能量效率越高.

如果考虑将式(11)用于衡量一个应用或者编译优化技术在某个系统上的能量性能的有效性,我们必须对其进行相应的修改.应用可能采用不同的算法,O_K 可能发生变化.如果使用 O_K 较小的算法,我们期望的结果是式(11)较小,也就是说,我们的算法更有效.但是 O_K 减小,式(11)可能增大,与我们的期望不符合.

究其原因,$Metric_{energy*delay^x}$ 用于衡量处理器或者系统的有效性,在一定时间内的吞吐量越大,说明系统的能量性能越有效.对于衡量应用的能量性能效果而言,我们希望使用尽量少的能量和尽量短的时间完成任务,不管操作数的多少,所以应当丢弃式(11)中的 O_K 项,使用 Et^x 作为应用的能量时间权衡尺度.同样,x(>0)的取值越大,对时间(性能)越重视.Et^x 应用尺度体现了用最少的能量和时间来完成任务.

4 开发部件使用的局部性

根据 Et^x 尺度,我们对提高应用的能量有效性进行分析.

4.1 Et^x尺度中固定时间t

在使用动态电压缩放和显示关闭功能部件的算法中,固定程序运行时间 t,尽量减少能量消耗 E 的研究很多;在实时嵌入式系统中,系统的完成更是有固定的时间要求,需要在指定时间内完成给定任务.

计算机系统的各种部件速度一般是不平衡的,例如,磁盘与内存间速度的不平衡,内存与处理器间速度的不平衡.降低非关键路径上的部件运行速度可以使得系统达到平衡.利用能量公式(9),系统的总能量消耗是各个部件能量消耗的和,在系统运行期间,只有部分部件处于关键路径上,它们的执行直接决定了程序的运行时间,而其他部件则处于非关键路径,它们可以减慢速度,同时降低电压,减少系统功耗.

另外,程序的需求和系统的部件数量也不可能达到完全的平衡.现代处理器提供了大量部件用于开发指令级并行度,而程序的运行特点是经常变化的.所以,经常是过多的部件数造成资源的浪费.根据式(9)和式(7),关闭部分不用的部件是节能的良好策略,这减小了 l 的数量.

另外一个问题是,如何调整 CPU 的速度保证在满足给定时间限制条件下完成任务并最小化能量消耗,这种情况最适合实时系统的需求.DK. Shin[34],提出使用静态方法分析程序的最差运行时间,设置 CPU 的相应速度,然后根据程序运行的具体路径进行动态电压调整.

4.2 最小化Et^x尺度

固定时间有时是不必要的,有些情况下是不现实的,这大大限制了对优化的灵活性,使用能量和时间权衡是必要的.这主要因为两种原因,一个是对能量的优化过程可能涉及到任务的重新调度或者指令的重新调度,甚至使用一些高层的优化过程,这可能造成程序执行时间的变化,很多工作对一些优化方法对程序的时间和能量变化进行过研究[25].另外,动态电压缩放和功能部件关闭的一些新的节能策略不可避免地引入了能量开销和性能损失,这也是程序执行时间发生变化的原因之一.

针对使用动态电压缩放和功能部件关闭方法进行的优化,下面使用我们的模型进行分析,研究一些可能的改进 Et^x 的方法.

根据能量公式和时间公式,应用尺度

$$Et^x=\sum_l(E^{c_l}_{comp}+E^{c_l}_{switch})t^x=\sum_l\left(\sum_k t^{c_l}_k P^{c_l}_{x^{c_l}_k}+\sum_k E^{c_l}_{x^{c_l}_k\to x^{c_l}_{k+1}}\right)t^x.$$

我们发现,如果切换的次数 k 和各个切换等级没有变化,功能部件数 l 一定,$\sum_l\left(\sum_k t^{c_l}_k P^{c_l}_{x^{c_l}_k}+\sum_k E^{c_l}_{x^{c_l}_k\to x^{c_l}_{k+1}}\right)$不变,而

$$t=\sum_k(t^c_k+t^c_{x^c_k\to x^c_{k+1}})=\sum_k t^c_k+\sum_k t^c_{x^c_k\to x^c_{k+1}}=t^c_{comp}+t^c_{switch}.$$

对于部件 c,如果它在某个时间段 t^c_k 不处于程序执行的关键路径上,在下一个时间段 t^c_{k+1} 处于关键路径上,那么 $P^c_{x^c_k}$ 到 $P^c_{x^c_{k+1}}$ 的切换造成的时间开销延长了程序执行时间 t.我们可以提前进行 $P^c_{x^c_k}$ 到 $P^c_{x^c_{k+1}}$ 的切换,这样可以减小或消除这次切换的时间开销.对于功耗等级切换时间开销较大的应用,这样的优化效果很明显,WP. Liao[35]讨论了上述方法的硬件实现和编译实现方法.

另外一种方法是减小功耗等级的切换次数 k.k 的减小有两个优点,一是降低了切换的能量开销和性能损失,二是增大了各个时间段的间隔,使得我们可以更有效地开发低功耗模式.

减小系统中使用的部件数 l 是又一种重要的方法.因为程序运行一般不会全部使用系统的资源,这是系统资源固定和应用需求变化间的矛盾,可以通过运行时关闭部件,减小 l 以达到供需的平衡.

4.3 开发部件使用的局部性

根据对模型的分析,我们发现减少参数 k 和 l 对提高应用能量的有效性有着重要作用,二者共同的特点是要求对部件使用局部化,所以本文提出了部件使用局部化的新概念.

系统中的指令类型是多种多样的,每种指令使用的功能部件或设备都是不同的,以往的任务调度和指令调度策略很少考虑到设备类型的因素.在新的低功耗技术支持下的系统,这些可能是关键的因素.程序执行期间对设备的使用是很复杂的,它可能随时都有启动设备的需求,如果这些设备被过于频繁地访问,考虑到节能策略的时间开销和能量损失,不是任何情况下使用节能方法都会得到收益,尽量集中一类部件或者一个部件的使用,最大化部件使用的间隔具有重要意义,这就是部件使用的局部化.

根据这一概念,我们总结出一些方法用于低功耗编译优化,这些方法可能是早有研究的,也可能是尚未考虑过的.

(1) 处理器部件类型局部化和设置恰当的部件数量(减小 k 和 l)

处理器中存在多种类型的处理部件,程序运行期间,可能使用不同类型的部件.如果在执行指令期间,把使用同种类型部件的指令尽量集中在一起,就可以更有效地使用节能策略.

例如,区分使用整数部件和浮点部件的指令.假定有指令序列 $i1,f2,i3,f4$,其中 i 代表整数指令,f 代表浮点指令.如果直接使用上面的调度方法,浮点指令和整数指令相互间隔,我们在运行期间需要对整数部件和浮点部件各进行两次能量等级切换.如果指令不相关,可以调整指令序列为 $i1,i3,f2,f4$,这样,部件只需要各切换一次,并且增大了切换间隔的时间(指令数).

除了集中对部件的使用,还需要设置程序运行需要的恰当的部件数目.在程序运行期间,各种部件类型使用的数目是不同的,并不是总处于供需的平衡状态.我们要使用有效的方法平衡程序需求和部件数量.例如,系统中的取指队列、保留站队列等结构的数量并不总是适当的,我们可以采取一定措施在程序运行期间关闭部分部件.过去在这方面有一些研究工作,例如,W.Zhang 等人[36]提出,在存在多种能量模式的 IALU 部件的 VLIW 处理器中,采用编译技术分析程序的特性,将非关键路径上的指令调度到慢速 IALU,以减少系统动态功耗,利用输入向量控制和能量门技术适时关闭和打开部件减少泄漏能量消耗;H.S.Kim 等人[30]采用编译技术在循环粒度中分析程序的 IPC 值,基于分析结果,动态分配激活的 IALU 数量,降低系统泄漏能量消耗.

(2) Cache 使用的局部化和设置适当的 cache 行数目(减小 l)

Cache 是处理器中能量消耗比例很高的部分,cache 使用的优化对优化系统能量消耗十分重要,需要对 cache 的使用进行局部化.这不单纯是根据 cache 的大小进行一些数据的局部性优化,而是假定在执行期间程序应当

使用尽量小的 cache 行数目,然后将剩余的 cache 行关闭.这与传统的基于 cache 的局部性优化存在差别,一些原来的基于循环的优化方法需要一定的修改才能得到更有效的优化效果.

设置好 cache 的使用数目,还要考虑 cache 行适时关闭和激活的策略,必须有恰当的 cache 管理方式才能减少引入的时间和能量开销.

对于 cache 的优化也存在一些研究,例如,W.Zhang 等人[12]研究了基于循环的粒度,在每个循环结束时用编译方法插入指令关闭指令 cache 行,减少泄漏能量消耗.

(3) 内存使用的局部化(减小 l)

内存的功耗占系统功耗的很大比例.内存系统能量的优化在很大程度上依赖于程序使用的局部性,对于多带的内存系统和多芯片系统的局部性优化需要进一步研究.例如,V.Delaluz 等人[14]对多运行模式、多带的内存系统进行了分析,使用软件策略优化程序的内存布局,降低存储器的活动.

(4) I/O 使用的局部化(减小 k)

对数据输入输出依赖很强的程序可以考虑进行 I/O 使用的局部化.I/O 局部化的可能方式是采用缓存方法,将大量小规模的 I/O 集中起来,这延长了两组 I/O 的时间,然后可能使用更积极有效的 I/O 节能策略.但是,这也可能造成内存需求的增加,必须考虑整体的能量消耗,确定能量的最大收益.

Taliver Heath 等人[37]在应用层和编译层对非交互式应用程序中的 I/O 访问时间进行变换,最大限度地使用内存空间缓存将来使用的数据,通过延长 I/O 的访问间隔来提高磁盘设备可以进行能量管理的比率.

(5) 多任务多设备的调度(减小 k)

对于运行在多任务环境、存在多种设备的系统,如何使用节能技术?任务的调度需要考虑对部件的使用情况,将集中使用某些部件的任务同时调度,这一般不会影响系统的吞吐量,并可以采取一些更有效的节能技术.

例如,YH.Lu 等人[38]研究了多任务多设备情况下,调度设备的使用,延长设备的空转时间,提高了能量管理的效率.

5 结论和以后的工作

在动态电压缩放和功能部件关闭技术支持下,开发部件使用的局部性是有效的编译优化技术,是重要的优化方向.

本文建立了编译的分析模型,根据对模型的分析,提出了部件使用局部性的概念.本文回顾了一些以往的开发部件使用局部性的工作,同时指出一些尚未进行研究的可能优化途径.这些方法往往在优化能量的同时产生了一些时间开销,需要使用 Et^x 尺度进行权衡,这些方法的有效性需要进一步实验验证.我们将对一些传统的优化方法进行分析,研究其对开发部件使用局部性的作用,发展有效的面向部件使用局部化的低功耗的编译优化技术.

致谢 在此,我们对审稿人给予的建议表示感谢.

References:

[1] Li ZY, Wang C, Xu R. Computation offloading to save energy on handheld devices: A partition scheme. In: ACM, ed. Proc. of the Int'l Conf. on Compilers, Architecture, and Synthesis for Embedded Systems. New York: ACM Press, 2001. 238~246.

[2] Stemm M, Katz RH. Measuring and reducing energy consumption of network interfaces in hand-held devices. IEEE Trans. on Communications, 1997,E80-B(8):1125~1131.

[3] Chase JS, Anderson DC, Thakar PN, Vahdat AM. Managing energy and server resources in hosting centers. In: ACM, ed. Proc. of the 18th ACM Symp. on Operating Systems Principles. New York: ACM Press, 2001. 1~14.

[4] Duarte D, Narayanan V, Irwin MJ. Impact of technology scaling in the clock system power. In: IEEE, ed. Proc. of the IEEE Computer Society Annual Symp. on VLSI (ISVLSI 2002). Washington: IEEE Computer Society Press, 2002. 1~6.

[5] Elnozahy M, Kistler M, Rajamony R. Energy conservation policies for Web servers. In: USENIX, ed. Proc. of the 4th USENIX Symp. on Internet Technologies and Systems. New York: The Advanced Computing Systems Association, 2003. 1~14.

[6] Psilogeorgopoulos M, Munteanu M, Chuang T-S. Ivey PA, Seed L. Contemporary techniques for lower power circuit design, PREST Deliverable D2.1. The University of Sheffield, 1998. 1~91. http: //www.shef.ac.uk/eee/esg/ lowpower/pdf-papers/prestd2. 1.pdf

[7] Klaiber A. The technology behind Crusoe™ processors—Low-Power x86-Compatible processors implemented with code morphingTM software. Transmeta Corporation, 2000. 1~18.

[8] Rele S, Pande S, Onder S, Gupta R. Optimizing static power dissipation by functional units in superscalar processor. In: Rele S, ed. Proc. of the 33rd Annual IEEE/ACM Int'l Symp. on Microarchitecture. Grenoble: Springer-Verlag, 2000. 261~275.

[9] Swaminathany V, Chakrabartyy K, Iyenga SS. Dynamic I/O power management for hard real-time systems. In: ACM, ed. Int'l Symp. on Software/Hardware Co-Design (CODES 2001). New York: ACM Press, 2001. 1~6.

[10] Weiser W, Welch B, Demers A, Shenker S. Scheduling for reduced CPU energy. In: USENIX, ed. Proc. of the 1st USENIX Symp. on Operating Systems Design and Implementation. New York: The Advanced Computing Systems Association, 1994. 1~11.

[11] Azevedo A, Issenin I, Cornea R. Profile-Based dynamic voltage scheduling using program checkpoints. In: Carlos K, ed. Proc. of the Conf. on Design, Automation and Test in Europe. Washington: IEEE Computer Society Press, 2002. 1~8.

[12] Zhang W, Hu JS, Degalahal V, Kandemir M, Vijaykrishnan N, Irwin MJ. Compiler-Directed instruction cache leakage optimization. In: IEEE ed. Proc. of the 35th Annual Int'l Symp. on Microarchitecture (MICRO-35). Washington: IEEE Computer Society Press, 2002. 208~218.

[13] Kadayif I, Kandemir M, Karakoy M. An energy saving strategy based on adaptive loop parallelization. In: ACM, ed. Proc. of the Design Automation Conf. (DAC 2002). New York: ACM Press, 2002. 1~6.

[14] Delaluz V, Kandemir M, Vijaykrishnan N, Sivasubramaniam A, Irwin MJ. DRAM energy management using software and hardware directed power mode control. In: IEEE ed. Proc. of the Int'l Symp. on High-Performance Computer Architecture. Washington: IEEE Computer Society Press, 2001. 1~8.

[15] Vijaykrishnan N, Kandemir M, Irwin MJ, Kim HS, Ye W. Energy-Driven integrated hardware-software optimization using simplePower. In: ACM ed. Proc. of the 27th Annual Int'l Symp. on Computer Architecture (ISCA 2000). New York: ACM Press, 2000. 95~106.

[16] Zhang CJ, Vahid F, Najjar W. A highly configurable cache architecture for embedded systems. In: ACM ed. Proc. of the 30th Annual Int'l Symp. on Computer Architecture. New York: ACM Press, 2003. 1~11.

[17] Witchel E, Asanović K. The span cache: Software controlled tag checks and cache line size. In: IEEE ed. Proc. of the 28th ISCA, Workshop on Complexity-Effective Design. Washington: IEEE Computer Society Press, 2001. 1~12.

[18] Gandhi KR, Mahapatra NR. A detailed study of hardware techniques that dynamically exploit frequent operands to reduce power consumption in integer function units. In: ACM ed. Proc. of the 2nd Annual Workshop on Duplicating, Deconstructing, and Debunking. New York: ACM Press, 2003. 1~9.

[19] Brooks D, Martonosi M. Dynamically exploiting narrow width operands to improve processor power and performance. In: IEEE ed. Proc. of the 5th Int'l Symp. on High Performance Computer Architecture. Washington: IEEE Computer Society Press, 1999. 1~10.

[20] Duarte D, Tsai Y-F, Vijaykrishnan N, Irwin MJ. Evaluating run-time techniques for leakage power reduction. In: IEEE ed. Proc. of the 7th ASPDAC/ 15th Int'l Conf. on VLSI Design. Washington: IEEE Computer Society Press, 2002. 1~8.

[21] Flautner K, Kim NS, Martin S, Blaauw D, Mudge T. Drowsy caches: Simple techniques for reducing leakage power. In: IEEE, ed. Proc. of the 29th Annual Int'l Symp. on Computer Architecture. Washington: IEEE Computer Society Press, 2002. 148~157.

[22] Gonzalez R, Horowitz M. Energy dissipation in general purpose microprocessors. IEEE Journal of Solid-State Circuits, 1996,31(9): 1277~1284.

[23] Martin AJ. Towards an energy complexity of computation. Information Processing Letters, 2001,77(2~4):181~187.

[24] Benini L, De Micheli G. System-Level power optimization: Techniques and tools. ACM Trans. on Design Automation of Electronic Systems (TODAES), 2000,5(2):115~192.

[25] Tiwari V, Malik S, Wolfe A. Compilation techniques for low energy: An overview. In: IEEE, ed. IEEE Symp. on Low-Power Electronics. Washington: IEEE Computer Society Press, 1994. 1~3.

[26] Seng JS, Tullsen DM. The effect of compiler optimizations on pentium 4 power consumption. In: IEEE, ed. Proc. of the 7th Annual Workshop on Interaction between Compilers and Computer Architectures. Washington: IEEE Computer Society Press, 2003. 1~6.

[27] Tiwari V, Malik S, Wolfe A, Lee MT-C. Instruction level power analysis and optimization of software. Journal of VLSI Signal Processing Systems, 1996,13(2):1~18.

[28] Kim HS, Irwin MJ, Vijaykrishnan MJ, Kandemir M. Effect of compiler optimizations on memory energy. In: IEEE ed. IEEE Workshop on Signal Processing Systems. Washington: IEEE Computer Society Press, 2000. 663~672.

[29] Hsu C-H, Kremer U. Compiler-Directed dynamic voltage scaling based on program regions. Technical Report, DCS-TR461: Rutgers University, 2001. 1~9.

[30] Kim HS, Vijaykrishnan N, Kandemir M, Irwin MJ. Adapting instruction level parallelism for optimizing leakage in VLIW architectures. In: ACM, ed. Proc. of the Languages, Compilers, and Tools for Embedded Systems (LCTES 2003). New York: ACM Press, 2003. 275~283.

[31] Delaluz V, Kandemir M, Vijaykrishnan N, Irwin MJ. Energy-Oriented compiler optimizations for partitioned memory architectures. In: ACM, ed. Proc. of the Int'l Conf. on Compilers, Architectures, and Synthesis for Embedded Systems (CASES 2000). New York: ACM Press, 2000. 1~10.

[32] Casmira J, Grunwald D. Dynamic instruction scheduling slack. In: Proc. of the 2000 KoolChips Workshop, Held in Conjunction with MICRO 2000 Monterey. 2000. 1~7.

[33] Govil K, Chan E, Wasserman H. Comparing algorithms for dynamic speed-setting of a low-power CPU. In: ACM, ed. Proc. of the 1st ACM Int'l Conf. on Mobile Computing and Networking. New York: ACM Press, 1995. 13~25.

[34] Shin DK, Kim JH, Lee SS. Low-Energy intra-task voltage scheduling using static timing analysis. In: ACM, ed. Proc. of the 38th Design Automation Conf. New York: ACM Press, 2001. 1~6.

[35] Liao WP, He L. Power modeling and reduction of VLIW processors. In: Kluwer, ed. Proc. of the Int'l Conf. on Parallel Architectures and Compilation Techniques, Workshop on Compilers and Operating Systems for Low Power. Norwell: Kluwer Academic Publishers, 2001. 155~171.

[36] Zhang W, Vijaykrishnan N, Kandemir M, Irwin MJ, Duarte MJ, Tsai Y-T. Exploiting VLIW schedule slacks for dynamic and leakage energy reduction. In: IEEE, ed. Proc. of the 34th Annual Int'l Symp. on Microarchitecture (MICRO-34). Washington: IEEE Computer Society, 2001. 102~113.

[37] Heath T, Pinheiro E, Hom J, Kremer U, Bianchini R. Code transformations for energy-efficient device management. IEEE Trans. on Computers, 2004,53(8):974~987.

[38] Lu Y-H, Benini L, De Micheli G. Low-Power task scheduling for multiple devices. In: ACM, ed. Proc. of the Int'l Workshop on Hardware/Software Codesign. New York: ACM Press, 2000. 39~43.