ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.17, No.7, July 2006, pp.1650−1660
DOI: 10.1360/jos171650
© 2006 by *Journal of Software*. All rights reserved.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

能量受限的软件预取优化问题*

陈 娟+, 易会战, 董 勇, 杨学军

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Energy-Constrained Software Prefetching Optimization

CHEN Juan+, YI Hui-Zhan, DONG Yong, YANG Xue-Jun

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

+ Corresponding author: Phn: +86-731-4575810, E-mail: juanchen@nudt.edu.cn

Chen J, Yi HZ, Dong Y, Yang XJ. Energy-Constrained software prefetching optimization. *Journal of Software*, 2006,17(7):1650−1660. http://www.jos.org.cn/1000-9825/17/1650.htm

Abstract: In mobile and embedded devices, the energy supply is strictly constrained with the battery capacity and energy saving ability. In these energy-constrained settings, the available energy budget is not sufficient to meet the optimal performance objective. This paper presents an energy-constrained software prefetching optimization approach, which can obtain the optimal performance under the limited energy resource. The approach is based on DVS-enabled CPU and memory. Through inserting frequency-scaling instructions, CPU and memory frequencies are simultaneously adjusted to meet two performance objectives (one is the time; the other is the processor gain) under a given energy constraint. A detailed analytical model is built and then the effectiveness of the approach is validated by a set of array-intensive applications. Experimental results show that the approach is effective for energy-constrained prefetching optimization problem.

Key words: dynamic voltage scaling; energy-constrained; low power; performance optimization; software prefetching

摘 要: 在移动设备和嵌入式设备中,能量的供给是十分有限的,它受限于能量供给设备的容量和节电能力的大小.在能量受限的环境下,电池所提供的能量不足以使系统达到最优的性能目标.因此,提出了一种能量受限环境下最优化预取性能的方法.该方法通过软件控制的手段,能在有限的能量供给条件下达到最优的性能.该方法是基于动态频率可调的CPU和存储器的.根据CPU和存储器的忙闲情况,通过插入频率调节指令,指导调节CPU和存储器的频率,使得预取优化的两个性能指标(一是时间,二是处理器收益)在一定的能量约束条件下达到最优.对该问题建立了详细的模型及模拟环境,并通过一组以数组访问为主的测试程序验证了该方法的有效性.模拟结果表明,该方法对能量受限预取优化问题是有效的.

关键词: 动态电压调节;能量受限;低功耗;性能最优;软件预取
中图法分类号: O224 文献标识码: A

在嵌入式设备和移动设备中,能量的供给是有限的,这就使得能量有效性问题变得十分重要.除了尽量提高

* Received 2005-10-07; Accepted 2005-12-31

© 中国科学院软件研究所 http://www.jos.org.cn

电池容量以外,能量受限的优化问题逐渐成为被关心的重点[1−3].不同于能量最优化方法[4−7],能量受限的性能最优化问题把能量作为一个约束条件,将性能作为优化的目标,使移动设备在有限的能量供给条件下最大化其计算能力.

本文选择经过预取优化之后的代码段作为优化的对象,研究在能量受限条件下如何最优化预取性能的问题.软件预取优化作为一种有效的延迟容忍技术,通过将计算操作和访存操作尽量重叠来隐藏访存延迟,从而改进性能[8,9].要想获得完美的预取效果,必须将预取指令插入到正确的位置,这需要对程序进行正确的分析和计算.有些时候预取并不能达到最完美的效果,我们称它们为不完美的预取.不完美的预取分为两种情况:CPU-bound 和 memory-bound.CPU-bound 是指在 CPU 计算和访存操作重叠的区域里,CPU 计算的时间相对较长,占主导地位;memory-bound 是指访存的时间比 CPU 计算的时间长,占主导地位.在这两种情况下分别产生了存储器空转和 CPU 空转.我们通过降低 CPU 或存储器的频率来尽量消除存储器空转或 CPU 空转,从而降低能量消耗.后文图 1 对此给出了描述.

动态电压调节(dynamical voltage scaling,简称 DVS)根据应用的负载变化动态改变系统的运行电压和时钟频率,由于功耗和电压的二次方成正比,DVS 能够有效地节省能量.目前,大多数流行的低功耗处理器都支持 DVS 技术,例如,Intel 公司的 Xscale 处理器、Transmeta 公司的 Crusoe 处理器,等等.

关于 DVS 技术的研究已有很多,例如:Fen Xie 等人[10]利用动态电压调整方法建立了一个最大化能量节省的分析模型;Hsu[11,12]提出了一种编译技术来识别可以进行频率调节的程序段.在他们的电压/频率调整算法中,考虑了 memory-bound 条件下 CPU 频率调节的机会.Xiaobo Fan 等人[13]对 DVS 和功耗感知的存储系统之间的相互影响进行了深入的探讨,他们认为,最小化功耗条件下的电压/频率调节,应该包括存储系统的功耗节省.从他们对 MediaBench 的模拟实验可以看出,将动态电压调整方法和功耗感知的存储系统相结合,比单独考虑一种情况所获得的能量节约要大.例如:只使用动态电压调整所获得的能量节省仅有 39%,如果将功耗感知的存储系统相结合所获得的能量节省会达到 89%.Ricardo Bianchini 等人[14]对多处理机中的预取及多线程进行分析,得出了一些性能指标的评价方法,为本文建立预取优化的性能及能量模型提供了一定的参考.在我们以前的研究当中,文献[1]已经提出了一个能量受限的预取优化模型.本文的工作和文献[1]的不同之处在于:对原有的能量受限的预取优化模型进行了改进,不仅考虑了能量受限对执行时间的影响,同时也考虑了对处理器利用率改进的影响,使得问题更加完善.

我们以循环块为主要研究对象,分析了能量受限条件下预取优化问题.其中包括两个性能目标:执行时间最小和处理器收益最大.我们对各种预取优化可能产生的情况建立了分析模型,该模型通过动态调节 CPU 频率和存储器频率以获得能量约束条件下最优的性能目标,并通过一组模拟实验验证了该方法的有效性.

本文第 1 节给出我们的分析模型及求解方法.第 2 节对实验进行描述并给出模拟结果.第 3 节是工作总结.

1 能量受限的预取优化分析模型

我们考虑进行预取优化的循环段原型为 for (*i*=0; *i*<*N*; *i*++) *Compute*(*i*);

经过软件预取优化之后,循环段变成如下形式:

```
// iteration 0, prefetch b blocks
prefetch(0,b);
for (i=0;i<N–step;i+=step){
    // prefetch b blocks for iteration i+step to i+2*step–1
    prefetch(i+step,b);
    // Compute iteration i to i+step–1
    for (j=0; j<step; j++) Compute(i*step+j);
}
for (j=0; j<step; j++) Compute(i*step+j);
```

© 中国科学院软件研究所 http://www.jos.org.cn

主要的不同是:在循环迭代中插入了几条预取指令.在整个分析模型中,主要涉及的参数有:

- b:每条预取指令所预取的 cache 块数;
- N_b:每个循环迭代中预取指令的数目;
- E_b:预取每个 cache 块所消耗的能量开销;
- C_p:预取每个 cache 块所耗费的时钟周期数;
- C_m:每个迭代中由于 cache 失效所产生的访存延迟周期数;
- C_c:相邻的两条预取指令之间的 CPU 计算周期数.

为了获取下面的分析模型,我们先在程序、微体系结构以及电路实现方面进行一些假设:

- 程序的逻辑行为不会随着频率的改变而改变;
- 存储器和 CPU 是异步执行的,假定存储器的频率调整和 CPU 的频率调整是相互独立的;
- CPU 的频率是连续可调的,存储器的频率也是连续可调的;
- 不考虑频率转换之间的能量开销和延迟开销.

我们认为:前两个假设是真实、合理的,后两个假设是为了便于预取优化问题的分析.

图 1 阐述了预取循环段的执行情况.每条预取指令预取了下一个或下几个迭代中所要访问的数据,避免了因 cache 失效所产生的访存延迟.在图 1(a)中,*Pi* 表示预取指令操作,它计算所要预取的数据地址,例如,*P*1 预取在 *T*2 计算段中所需的数据,该数据在 *B* 点被访问;*Mi* 执行因 *Pi* 预取操作产生的访存操作.图 1(a)给出了完美的预取操作——存储访问完全与 CPU 计算重叠起来了,没有 CPU 空转或存储器空转;图 1(b)和图 1(c)给出了两种不完美的预取情况——CPU-bound 和 memory-bound.在这两种情况下,分别存在存储器空转和 CPU 空转,可以通过降低存储器频率和 CPU 频率的方法消除空转,从而降低能量消耗(*Pi* 表示预取第 *i*+1 个迭代中指令的时间开销;*Mi* 表示 *Pi* 预取所产生的访存开销;*Ti* 分别表示的是第 *i* 次迭代中计算部分的时间开销).

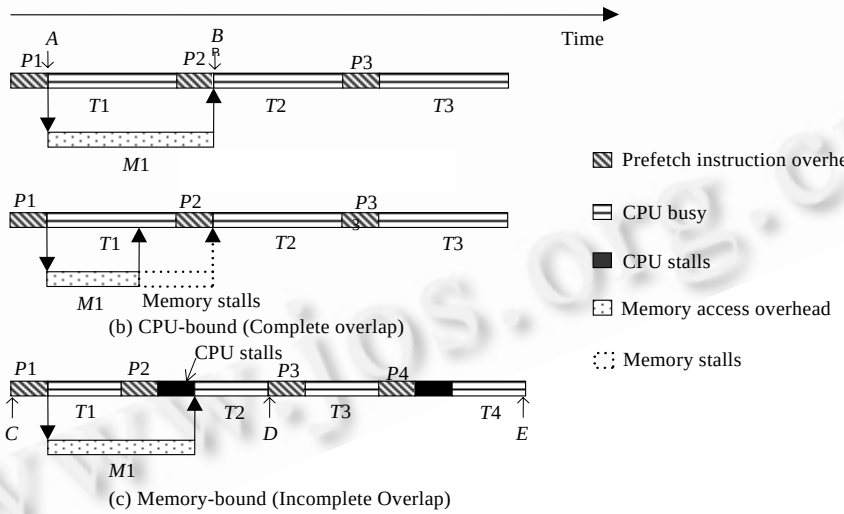


Fig.1 Illustration of software prefetching

图 1 预取优化的时序图

问题描述如下:给定一个经过预取优化的循环段 L 以及一个动态电压/频率可调的系统,找到最优的 CPU 频率 f_c 和存储器频率 f_m,使得程序在运行时达到最优的性能目标(执行时间最小或处理器收益最大),并且能量消耗不超过给定的能量预算值.

1.1 性能目标

1.1.1 最小化执行时间 T_{total}

我们先考虑第 1 个性能目标——最小化循环段的执行时间.统计循环段的执行时间需要分两种情况:一是

© 中国科学院软件研究所 http://www.jos.org.cn

CPU-bound 情况;二是 memory-bound 情况.

在第 1 种情况下,由于 CPU 计算时间足够长,将访存延迟完全隐藏了,因此执行时间 T_{total} 由预取数据地址计算时间和 CPU 计算时间构成.由于每条预取指令所取的 cache 块数为 b,每次迭代中预取指令数为 N_b,那么一次迭代中所要预取的 cache 块数就为 $b\cdot N_b$,又因为预取一个 cache 块所产生的时间开销为 C_p,那么预取数据地址计算的时间开销为$(b\cdot N_b\cdot C_p)/f_c$.这里,$f_c$ 表示 CPU 的频率.由于一次迭代中 CPU 计算部分的周期数为 C_c,很容易得到 CPU 计算部分的时间开销为 C_c/f_c.考虑 N 次迭代,因此,整个执行时间 T_{total} 就为

$$T_{total}=N\cdot\left(\frac{b\cdot N_b\cdot C_p}{f_c}+\frac{C_c}{f_c}\right)(cond1);$$

在第 2 种情况下,访存的时间较长,造成了 CPU 必须空转等待一段时间(如图 1(c)所示).所以,总的执行时间是受访存时间影响的.从图 1(c)来看,C 点到 D 点的间隔和 D 点到 E 点的间隔相等,且从 C 点到 D 点的执行时间占了整个执行时间的 2/N,容易得出从 C 点到 D 点的执行时间为 $\left(\frac{b\cdot N_b\cdot C_p}{f_c}+\frac{C_m}{f_m}+\frac{C_c}{f_c}\right)$,所以整个执行时间为

$$T_{total}=\frac{N}{2}\cdot\left(\frac{b\cdot N_b\cdot C_p}{f_c}+\frac{C_m}{f_m}+\frac{C_c}{f_c}\right)(cond2).$$

这里,$cond1$ 和 $cond2$ 分别表示 CPU-bound 条件和 memory-bound 条件,即

$$cond1\quad 代表\quad \frac{C_m}{f_m^0}\le\frac{b\cdot N_b\cdot C_p}{f_c^0}+\frac{C_c}{f_c^0},\qquad cond2\quad 代表\quad \frac{C_m}{f_m^0}>\frac{b\cdot N_b\cdot C_p}{f_c^0}+\frac{C_c}{f_c^0}$$

其中, f_c^0 和 f_m^0 分别表示初始的 CPU 频率和存储器频率.

综上所述,我们可以得到第 1 个性能目标——最小化执行时间.

$$\min\left(\frac{N}{2}\cdot\left(\frac{b\cdot N_b\cdot C_p+C_c}{f_c}+\frac{C_m}{f_m}\right)+\lambda\cdot\frac{N}{2}\cdot\left(\frac{b\cdot N_b\cdot C_p+C_c}{f_c}-\frac{C_m}{f_m}\right)\right)\tag{1}$$

这里, $\lambda=\begin{cases}1, & cond1\\ 0, & cond2\end{cases}$.

1.1.2 最大化处理器收益 G

定义 **1**. 处理器的利用率 U:处理器实际耗费在执行程序上的时间与程序运行总耗时之比,一般有 $0\le U\le 1$.

定义 **2**. 处理器收益 G:预取优化之后处理器利用率 U_p 与优化之前处理器利用率 U_{np} 之比.

第 2 个性能目标是最大化处理器收益 G.该收益越大,说明预取优化方法对提高处理器的利用率越有效.而处理器的利用率一般不能达到 100%.基于前面计算执行时间时的分析,很容易得到在两种不完美预取条件(CPU-bound 和 memory-bound)下,处理器的利用率分别为

$$U_p=\frac{C_c/f_c}{(b\cdot N_b\cdot C_p)/f_c+C_c/f_c}(cond1),\qquad U_p=\frac{2\cdot C_c/f_c}{(b\cdot N_b\cdot C_p+C_c)/f_c+C_m/f_m}(cond2).$$

未经过预取优化时,我们假定没有预取的数据的访问都会发生失效,此时,处理器的利用率为

$$U_{np}=\frac{C_c/f_c}{C_m/f_m+C_c/f_c}.$$

处理器收益 $G=U_p/U_{np}$,容易发现 G 一般大于 1,且在 memory-bound 时收益接近 2(这里,我们认为 $C_p<<C_c$).这主要是因为在 memory-bound 情况下,访存失效的开销很大,使得处理器利用率很低,预取提高了处理器的利用率.根据前面定义的二值函数λ,可以得到第 2 个最优化目标——最大化处理器收益:

$$\max\left((2-\lambda)\frac{\frac{C_m}{f_m}+\frac{C_c}{f_c}}{\frac{b\cdot N_b\cdot C_p+C_c}{f_c}+(1-\lambda)\cdot\frac{C_m}{f_m}}\right)\tag{2}$$

© 中国科学院软件研究所 http://www.jos.org.cn

1.2 能量约束

能量消耗 E_{total} 主要包括 3 部分*:预取指令计算数据地址消耗的能量 E_p、CPU 计算时消耗的能量 E_c 以及访存所消耗的能量 E_m.所以有 $E_{total}=E_p+E_c+E_m$.

定义 **3**. 能量预算 E_{budget}:系统可以提供给该程序的能量值.在能量受限环境中,能量预算可能小于能量上限 E_{bound}.

定义 **4**. 能量上限 E_{bound}:在没有能量约束条件下达到最优性能时所需耗费的最大能量值.

当能量预算为 E_{budget} 时,能量约束可以表示为 $E_p+E_c+E_m \le E_{budget}$.

E_p 表示预取指令计算地址所消耗的能量.我们假定预取一个 cache 块的能耗为 E_b,前面我们计算出一次迭代所预取的 cache 块数为 $b \cdot N_b$.因此,E_p 表达式为 $E_p=E_b \cdot b \cdot N_b \cdot N$.

E_c 表示 CPU 计算部分的能量消耗.假定 CPU 在频率为 f_c 时的功耗为 $P_c(f_c)$,前面已经假定了一次迭代内的 CPU 计算耗时为 C_c 个时钟周期.因此,E_c 表达式为 $E_c = P_c(f_c) \cdot \frac{C_c}{f_c} \cdot N$.

类似地,假定存储器在频率为 f_m 时的功耗为 $P_m(f_m)$,E_m 的表达式为 $E_m = P_m(f_m) \cdot \frac{C_m}{f_m} \cdot N$.

总的能量消耗为

$$E_{total} = E_b \cdot b \cdot N_b \cdot N + P_c(f_c) \cdot \frac{C_c}{f_c} \cdot N + P_m(f_m) \cdot \frac{C_m}{f_m} \cdot N \tag{3}$$

这样就得到了能量约束条件:

$$E_b \cdot b \cdot N_b \cdot N + P_c(f_c) \cdot \frac{C_c}{f_c} \cdot N + P_m(f_m) \cdot \frac{C_m}{f_m} \cdot N \le E_{budget} \tag{4}$$

在该能量约束条件中,存在两个功耗函数:$P_c(f_c)$和 $P_m(f_m)$,下面我们来确定这两个函数的值.由于功耗正比于电压的平方和频率的乘积,这里为了讨论问题方便,我们假定电压和频率呈线性关系.这是因为电压和频率满足公式(5)给出的关系,当β取一定值时,可以认为电压和频率呈线性关系.

$$f \propto \frac{(V - V_t)^{\beta}}{V} \tag{5}$$

这里,β是技术相关因子,对于不同的系统,β取值不同.一般来说,$\beta \in [1,2]$.当β取值为 2 时,可以近似认为频率正比于所提供电压.因此,我们假定功耗函数 $P_c(f_c)$和 $P_m(f_m)$的形式满足下面的两个公式:

$$P_c(f_c) = k_1 \cdot f_c^3 \tag{6}$$

$$P_m(f_m) = k_2 \cdot f_m^3 \tag{7}$$

参照 Transmeta 公司的 Crusoe 处理器 TM5700/5900[15]可以近似地得到 k_1,k_2 的值,分别为 7.72E-27 和 1.09 E-24.现将式(6)和式(7)代入式(3)和式(4),得到下面的能量计算公式及能量约束条件:

$$E_{total} = E_b \cdot b \cdot N_b \cdot N + k_1 \cdot f_c^2 \cdot C_c \cdot N + k_2 \cdot f_m^2 \cdot C_m \cdot N,$$

$$E_b \cdot b \cdot N_b \cdot N + k_1 \cdot f_c^2 \cdot C_c \cdot N + k_2 \cdot f_m^2 \cdot C_m \cdot N \le E_{budget}.$$

1.3 能量受限条件下的两个最优化问题

根据前面确定的两个最优化目标和能量受限条件,我们可以得出两个能量受限条件下的最优化问题:(1) 能量受限下的最小执行时间问题;(2) 能量受限下的最大处理器收益问题,如图 2 所示.

这两个优化问题有完全相同的约束条件,仅在目标函数上有所不同.第 3 个约束条件保证了在频率变化过程中不会改变程序所属类型的性质**.最后两个约束条件限定了 CPU 频率和存储器频率变化的范围, f_c', f_c'' 分别是 CPU 频率变化的下界和上界; f_m', f_m'' 分别是存储器频率变化的下界和上界.

---

* 这里考虑的能耗仅为功能部件及存储器的能量消耗,而忽略总线、时钟等其他能耗.

** 这里所说的程序所属类型是指 CPU-bound 或 memory-bound.

© 中国科学院软件研究所 http://www.jos.org.cn

$$\min\left(\frac{N}{2}\cdot\left(\frac{b\cdot N_b\cdot C_p+C_c}{f_c}+\frac{C_m}{f_m}\right)+\lambda\cdot\frac{N}{2}\left(\frac{b\cdot N_b\cdot C_p+C_c}{f_c}-\frac{C_m}{f_m}\right)\right)$$
$$E_b\cdot b\cdot N_b\cdot N+k_1\cdot f_c^2\cdot C_c\cdot N+k_2\cdot f_m^2\cdot C_m\cdot N\le E_{budget}$$
$$\lambda=\begin{cases}1, & cond1\\ 0, & cond2\end{cases}$$
$$\begin{cases}\dfrac{C_m}{f_m}\le\dfrac{b\cdot N_b\cdot C_p+C_c}{f_c}, & \text{if } \lambda=1\\ \dfrac{C_m}{f_m}>\dfrac{b\cdot N_b\cdot C_p+C_c}{f_c}, & \text{if } \lambda=0\end{cases}$$
$$f_c'\le f_c\le f_c''$$
$$f_m'\le f_m\le f_m''$$

(a) Problem 1: Minimize the execution time

(a) 优化问题 1:能量受限下的最小执行时间问题

$$\max\left((2-\lambda)\frac{\frac{C_m}{f_m}+\frac{C_c}{f_c}}{\frac{b\cdot N_b\cdot C_p+C_c}{f_c}+(1-\lambda)\cdot\frac{C_m}{f_m}}\right)$$
$$E_b\cdot b\cdot N_b\cdot N+k_1\cdot f_c^2\cdot C_c\cdot N+k_2\cdot f_m^2\cdot C_m\cdot N\le E_{budget}$$
$$\lambda=\begin{cases}1, & cond1\\ 0, & cond2\end{cases}$$
$$\begin{cases}\dfrac{C_m}{f_m}\le\dfrac{b\cdot N_b\cdot C_p+C_c}{f_c}, & \text{if } \lambda=1\\ \dfrac{C_m}{f_m}>\dfrac{b\cdot N_b\cdot C_p+C_c}{f_c}, & \text{if } \lambda=0\end{cases}$$
$$f_c'\le f_c\le f_c''$$
$$f_m'\le f_m\le f_m''$$

(b) Problem 2: Maximize processor gain

(b) 优化问题 2:能量受限下的最大处理器收益问题

Fig.2 Two performance optimizations with the energy constraint

图 2 能量受限条件下的两个性能最优问题

2 实验结果

这一节,我们选择一组以数组访问为主的例子来验证我们提出的能量受限条件下的性能最优化问题.首先,建立一个模拟环境;然后,在此环境下进行模拟,并对模拟结果进行分析.

2.1 模拟环境及benchmarks

我们建立的理想系统模型基于一种流行的低功耗处理器——Transmeta Crusoe™ 处理器 TM5700/TM5900.根据 Crusoe 处理器的 Data Book[15]的相关数据,我们获得了如表 1 所示的相关模拟平台的数据.

Table 1 Experimental parameters

表 1 系统参数以及和程序相关的参数

System parameter	Meaning	Value
E_b	Energy consumption by a cache block prefetched	10 pJ
b	The number of cache block prefetched by a prefetching instruction	4
C_p	The time consumed by a prefetched cache block	1 cycles
k_1	The coefficient for $P_c(f_c)=k_1\cdot f_c^3$	7.72E-27
k_2	The coefficient for $P_m(f_m)=k_2\cdot f_m^3$	1.09E-24
f_c^0	Initial CPU frequency	1000 MHz
f_m^0	Initial memory frequency	133 MHz
f_c'	The lower bound of continuous CPU frequency	433 MHz
f_c''	The upper bound of continuous CPU frequency	1000 MHz
f_m'	The lower bound of continuous memory frequency	83 MHz
f_m''	The upper bound of continuous memory frequency	133 MHz
Program parameter	**Meaning**	**Value**
N	Loop iteration counts including prefetching	Program specified
C_c	Computation time in one iteration (cycles)	Profiled data
C_m	Memory access time in one iteration (cycles)	Profiled data
N_b	The number of prefetching instructions in one iteration	Optimization specified
E_{bound}	The upper bound energy consumption	Program specified

为了获得表 1 中所示的程序参数,我们首先使用了一个修改的 SimpleScalar 工具集[16]对程序进行模拟.这个修改的 SimpleScalar 建模了一个 1GHz 4 路流出的动态调度处理器,并且,该模拟器建模了处理器的所有方面,包括指令取单元、分支预测器、寄存器重命名单元、功能单元流水和重排序缓冲.这个修改的 SimpleScalar 工具集由 Hameed[17]提供.该工具集修改的主要内容是在处理器的指令集中增加了预取指令.此外,该模拟器还对

© 中国科学院软件研究所 http://www.jos.org.cn

存储系统进行了详细的建模.其中的cache设计是:一个分离的8Kbyte直接映射$L1$ cache(cache块大小为32byte)和一个统一的256Kbyte 4路组相联$L2$ cache (cache块大小为64byte),像这样的cache配置能够满足我们的输入数据集.由于该模拟器没有对CPU频率和存储器频率进行调整,我们将SimpleScalar的模拟结合以MATLAB工具[18],完成对整个模型的模拟过程.

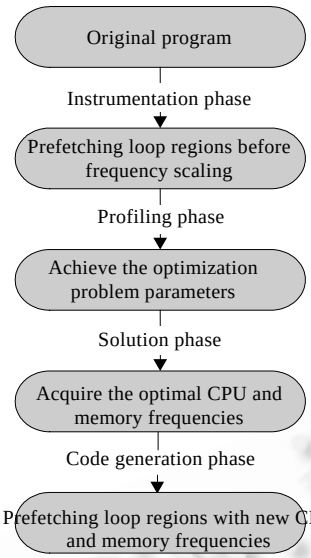


Fig.3 Framework of energy-constrained prefetching

图3 能量受限的预取优化问题编译框架

编译指导的能量受限预取优化的原型框架如图3所示.整个原型实现包括4个阶段:采集阶段、profile阶段、求解阶段和代码生成阶段.采集阶段完成从源程序中抽取出要优化的循环段,并进行预取优化;profile阶段使用SimpleScalar工具对预取优化循环段进行模拟执行,得出模型中所需的参数,例如C_c,C_m;求解阶段在我们的模拟平台上完成对最优CPU频率及存储器频率的求解;代码生成阶段中,根据上一阶段得到的最优CPU频率和存储器频率值,在源代码的相应位置处插入频率调整指令,获得最终的优化代码.

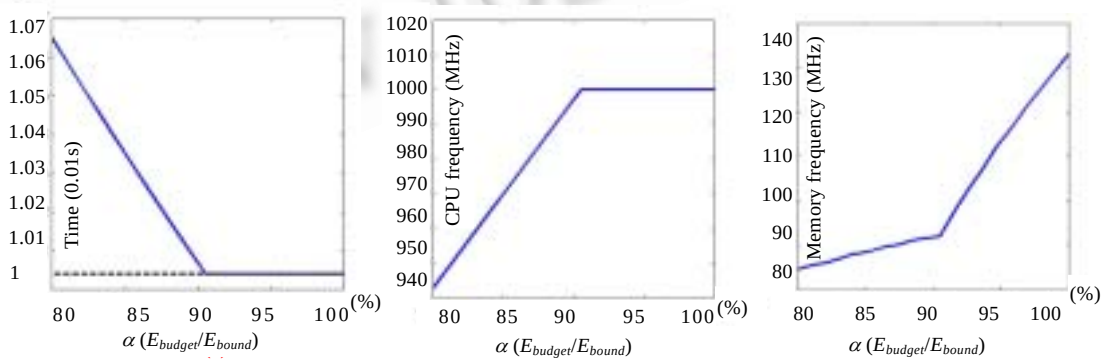
实验中,我们选择了一组以数组访问为主的应用来评价我们方法的有效性.这组测试程序包括Adi,2D Jacobi,Matmult,Stencil和Syr2k. Adi是从Livermore的核心基准测试程序中抽取的部分,包括6个数组;2D Jacobi执行了一个二维Jacobi松弛,包含2个数组;Matmult是两个矩阵相乘的运算,包含3个数组;Stencil是一个关于5点的模板计算,包含2个数组;Syr2k是Rank-2K更新计算程序,用来解带状对称矩阵的,包含3个数组.其中:Adi程序中数组的规模是1024×1024×3,其余程序中的数组规模均为1024×1024.

2.2 模拟结果

2.2.1 能量受限对执行时间的影响

首先分析一下不同的能量预算值E_{budget}对性能及频率调整的影响.在能量受限条件下,给定的能量预算值可能并没有达到能量上限E_{bound},此时,我们用表达式$E_{budget}=\alpha\cdot E_{bound}$来表示它们之间的关系.在能量受限的环境中,考虑$\alpha\leq 1$.$\alpha$越小,能量预算值越小,能量受限程度越大,可能的性能损失也越大.

图4和图5分别给出了在CPU-bound和memory-bound情况下性能损失情况及此时的CPU频率和存储器频率.



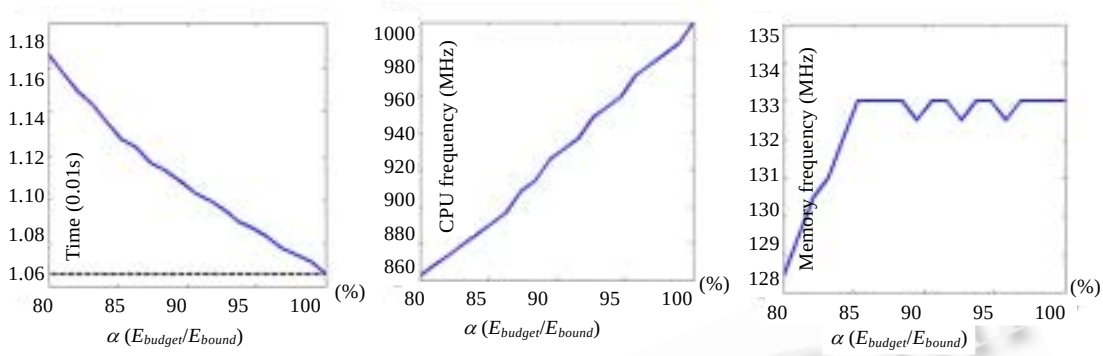
(a) Execution time variance (b) CPU frequency variance (c) Memory frequency variance

(a) 执行时间变化 (b) CPU频率变化 (c) 存储器频率变化

Fig.4 The impact of energy budget on the execution time for CPU-bound case (C_c=1000, C_m=90)

图4 CPU-Bound情况下不同的能量预算对执行时间的影响(C_c=1000,C_m=90)

© 中国科学院软件研究所 http://www.jos.org.cn



(a) Execution time variance (b) CPU frequency variance (c) Memory frequency variance

(a) 执行时间变化 (b) CPU频率变化 (c) 存储器频率变化

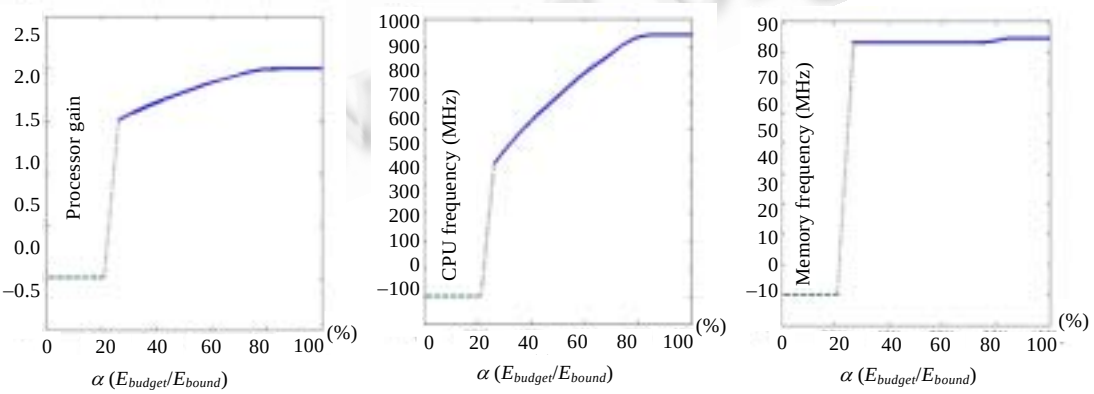
Fig.5 The impact of energy budget on the execution time for memory-bound case (C_c=1000, C_m=150)

图5 在memory-bound情况下不同的能量预算对执行时间的影响(C_c=1000,C_m=150)

在图 4(a)和图 5(a)中,虚线表示在没有能量约束条件下所获得的性能;实线表示在一定的能量预算下所获得的性能.可以看到:随着α值的增大,实线表示的性能越来越接近虚线表示的最优性能,且在图 4(a)中,当α增大到 91.1%时,性能已经到达没有能量约束时的最优值了.这说明:在 CPU-bound 情况下,可以在没有性能损失的情况下节约 8.9%的能量.注意:此时的 CPU 频率处于最高值,而存储器频率比最高值稍低.图 4(b)和图 4(c)分别给出了在同样条件下 CPU 频率和存储器频率变化趋势.图 5 给出的是 memory-bound 情况下的变化趋势图.在图 5(a)中,实线和虚线在α=1 时重合.这说明:在 memory-bound 情况下,不能在没有性能损失的条件下获得能量节约,即能量节约必然伴随着性能损失.

2.2.2 能量受限对处理器收益的影响

图 6 和图 7 分别给出了在两种情况下不同的能量预算对处理器收益的影响.在这两个图中,α的变化范围比图 4、图 5 要大,在 0~1 之间.可以看到:当α值低于 21.1%时,处理器获得的收益为 0;当α大于 26.3%时的值才是有意义的.我们用实线表示有意义的结果,而用虚线表示处理器收益为 0 时的情况,用点线将两部分连接起来.从模拟结果来看,memory-bound 情况下处理器收益几乎就等于 2,只在 CPU-bound 情况下处理器收益会随着α值的减小而有所降低.并且存储器频率在 memory-bound 情况下几乎稳定在最低的 83MHz 处,在 CPU-bound 情况下存储器频率产生微小变化.



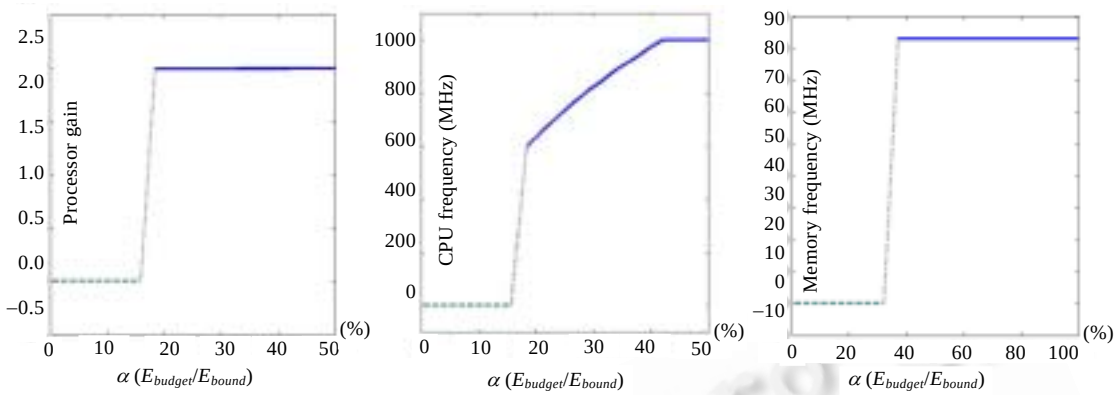
(a) Processor gain variance (b) CPU frequency variance (c) Memory frequency variance

(a) 处理器收益变化 (b) CPU频率变化 (c) 存储器频率变化

Fig.6 The impact of energy budget on the processor gain for CPU-bound case (C_c=1000, C_m=90)

图6 CPU-bound情况下不同的能量预算对处理器收益的影响(C_c=1000,C_m=90)

© 中国科学院软件研究所 http://www.jos.org.cn



(a) Processor gain variance (b) CPU frequency variance (c) Memory frequency variance

(a) 处理器收益变化 (b) CPU频率变化 (c) 存储器频率变化

Fig.7 The impact of energy budget on the processor gain for memory-bound case (C_c=1000, C_m=150)

图7 Memory-bound情况下不同的能量预算对处理器收益的影响(C_c=1000,C_m=150)

2.2.3 Benchmark 模拟结果

首先,我们给出每个测试程序在不同的能量受限程度下产生的性能损失,见表 2.其中能量受限程度用α值表示,且α越小说明能量约束越大.可以看到,对于 CPU-bound 情况(Adi),当 91.1%≤α≤100%时,没有性能损失,而此时 CPU 频率处于最高值,存储器频率低于最高值,这也验证了图 4(a)中出现的现象——CPU-bound 情况下可以通过适当降低存储器频率获得一部分能量节约,同时保证没有性能损失.

Table 2 Performance loss with the different degrees of energy constraint (α)

表2 在不同的能量受限程度(α)下的性能损失程度

Type	Program	α (%)	CPU frequency (MHz)	Memory frequency (MHz)	Performance loss (%)
CPU-bound	Adi	82.2	942	88.6	5.8
		86.7	971	88.6	2.9
		91.1	1 000	88.6	0
		95.6	1 000	111	0
		100	1 000	133	0
	2D Jacobi	82.2	898	122	10.2
		86.7	927	122	7.3
		91.1	942	127	5.8
		95.6	971	127	2.9
		100	1 000	133	0
Memory-bound	Matmult	82.2	898	122	9.3
		86.7	927	122	7.8
		91.1	942	127	5.0
		95.6	971	127	3.6
		100	1 000	133	0
	Stencil	82.2	782	133	6.3
		86.7	840	133	4.4
		91.1	898	133	2.7
		95.6	942	133	1.5
		100	1 000	133	0
	Syr2k	82.2	811	133	6.4
		86.7	869	133	4.2
		91.1	913	133	2.7
		95.6	956	133	1.3
		100	1 000	133	0

表 3 类似地给出了不同的能量受限条件下所获取的处理器收益大小,结果与图 6、图 7 的模拟结果相吻合.在 memory-bound 情况下,不同能量受限程度下,最优的处理器收益值都接近 2.只在 CPU-bound 情况下,当α值降低时,最优的处理器收益会有所降低(例如 Adi).对于存储器频率变化,在 memory-bound 情况下,始终保持在最低

© 中国科学院软件研究所 http://www.jos.org.cn

值处.相比之下,CPU 频率变化比存储器频率变化要大些,它随着α值的降低而减小.

Table 3 Processor gain with the different degree of energy constraint (α)

表 3 在不同的能量受限程度(α)下所获取的处理器收益

Type	Program	α (%)	CPU frequency (MHz)	Memory frequency (MHz)	Processor gain
CPU-bound	Adi	55.6	753	83	1.714 5
		66.7	840	83	1.797 6
		77.8	913	83	1.867 0
		88.9	985	83	1.936 3
		100	1 000	83	1.950 2
	2D Jacobi	55.6	724	94	1.988 0
		66.7	811	105	1.990 2
		77.8	855	111	1.991 1
		88.9	855	111	1.991 1
		100	855	111	1.991 1
Memory-bound	Matmult	55.6	782	83	1.992 0
		66.7	869	83	1.992 5
		77.8	942	83	1.992 8
		88.9	1 000	83	1.993 1
		100	1 000	83	1.993 1
	Stencil	55.6	855	83	1.989 5
		66.7	971	83	1.990 5
		77.8	1 000	83	1.990 8
		88.9	1 000	83	1.990 8
		100	1 000	83	1.990 8
	Syr2k	55.6	826	83	1.993 2
		66.7	942	83	1.993 9
		77.8	1 000	83	1.994 1
		88.9	1 000	83	1.994 1
		100	1 000	83	1.994 1

3 总 结

本文在以前工作的基础上,对能量受限的预取优化问题进行了较全面的分析,目的是为了解决在能量有限条件下如何通过调节 CPU 频率和存储器频率来满足给定的能量预算,使其在满足能量预算的条件下最大化性能目标.本文提出的性能目标包括两个:执行时间和处理器收益.我们建立了分析模型,并进行了模拟实验,得出了在每个性能目标下,不同的能量约束条件对性能的影响,以及达到最优性能时的 CPU 频率和存储器频率变化趋势.模拟结果表明:(1) 能量约束越大,执行时间越长,性能损失越大,相应的 CPU 频率和存储器频率都被调节到较低的水平;(2) CPU-bound 情况下,可以通过适当降低存储器频率获得一部分能量节约,同时保证没有性能损失;(3) 可以通过将存储器频率设置到最低,同时适当地降低 CPU 频率来获得较高的处理器收益(处理器收益接近 2).

References:

[1] Chen J, Dong Y, Yi HZ, Yang XJ. Energy-Constrained prefetching optimization in embedded applications. In: Yang LT, Amamiya M, Liu Z, Guo MY, Rammig FJ, eds. Proc. of the Embedded and Ubiquitous Computing—EUC 2005. LNCS 3824, Springer-Verlag, 2005. 267−280.

[2] AlEnawy TA, Aydin H. Energy-Constrained performance optimizations for real-time operating systems. In: Proc. of the Workshop on Compilers and Operating Systems for Low-Power (COLP 2003). New Orleans, 2003. http://cs.gmu.edu/~aydin/colp03.pdf

[3] Rusu C, Melhem R, Mossé D. Maximizing rewards for real-time applications with energy constraints. ACM Trans. on Embedded Computing Systems, 2003,2(4):537−559.

[4] Chen J, Dong Y, Yi HZ, Yang XJ. Compiler-Directed energy-aware prefetching optimization for embedded applications. In: Yang LT, Zhou XS, Zhao W, Wu ZH, Zhu YA, Lin M, eds. Proc. of the Embedded Software and Systems: 2nd Int'l Conf., ICESS 2005. LNCS 3820, Springer-Verlag, 2005. 230−243.

© 中国科学院软件研究所 http://www.jos.org.cn

[5] Zhao RC, Tang ZM, Zhang ZQ, Gao GR. Study on the low power technology of software pipeline. Journal of Software, 2003, 14(8):1357−1363 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1357.htm

[6] Yang HB, Govindarajan R, Gao GR, Cai G. Maximizing pipelined functional units usage for minimum power software pipelining. Technical Report, Delaware: CAPSL at the University of Delaware, 2001,41.

[7] Yang HB. Power-Aware compilation techniques for high performance processors [Ph.D. Thesis]. Delaware: University of Delaware, 2004.

[8] Mowry TC. Tolerating latency through software-controlled data prefetching [Ph.D. Thesis]. Stanford: Stanford University, 1994.

[9] Chen SM, Gibbons PB, Mowry TC. Improving index performance through prefetching. In: Proc. of the 2001 SIGMOD Int'l Conf. on Management of Data. New York: ACM Press, 2001. 235−246. http://citeseer.ist.psu.edu/chen01improving.html

[10] Xie F, Martonosi M, Malik S. Compile-Time dynamic voltage scaling settings: Opportunities and limits. In: Proc. of the ACM SIGPLAN 2005 Conf. on Programming Language Design and Implementation (PLDI 2003). New York: ACM Press, 2003. 49−62. http://parapet.ee.princeton.edu/papers/pldi081-xie.pdf

[11] Hsu CH, Kremer U, Hsiao M. Compiler-Directed dynamic voltage/frequency scheduling for energy reduction in microprocessors. In: Proc. of the Int'l Symp. on Low Power Electronics and Design (ISLPED 2001). New York: ACM Press, 2001. 275−278.

[12] Hsu CH. Compiler-Directed dynamic voltage and frequency scaling for CPU power and energy reduction [Ph.D. Thesis]. New Brunswick: State University of New Jersey, 2003.

[13] Fan XB, Ellis CS, Lebeck AR. The synergy between power-aware memory systems and processor voltage scaling. In: Proc. of the Workshop on Power-Aware Computer Systems (PACS 2003). 2003. 164−179. http://www.cs.duke.edu/~alvy/papers/dvs-pacs03.pdf

[14] Bianchini R, Lim BH. Evaluating the performance of multithreading and prefetching in multiprocessors. Journal of Parallel and Distributed Computing (JPDC), Special Issue on Multithreading for Multiprocessors, 1996,37(1):83−97.

[15] Crusoe processor model TM5700/TM5900 data book. 2004. http://www.transmeta.com/crusoe_docs/tm5900_databook_040204.pdf

[16] Burger D, Austin TM. The SimpleScalar tool set, Version 2.0. Technical Report, CS TR 1342, University of Wisconsin-Madison, 1997.

[17] Badawy AH, Aggarwal A, Yeung D, Tseng CW. The efficacy of software prefetching and locality optimizations on future memory systems. Journal of Instruction-Level Parallelism 6, 2004. http://www.jilp.org/vol6/

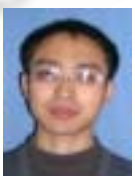
[18] http://www.mathworks.com/

附中文参考文献:

[5] 赵荣彩,唐志敏,张兆庆,Gao GR.软件流水的低功耗编译技术研究.软件学报,2003,14(8):1357−1363. http://www.jos.org.cn/1000-9825/14/1357.htm

陈娟(1980－),女,江西南昌人,博士生,主要研究领域为低功耗编译优化,并行算法.

易会战(1976－),男,博士生,主要研究领域为低功耗编译优化,计算机体系结构.



董勇(1980－),男,助理研究员,主要研究领域为并行文件系统,体系结构.

杨学军(1963－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为并行计算,体系结构.

© 中国科学院软件研究所 http://www.jos.org.cn