基于 GINI 指数分类的嵌入式 CPU 功耗预测方法

王 海 高 岭 宋振孝 戴小平 卢怡杰

(西北大学信息科学与技术学院 西安 710069)

摘 要 文中提出了一种基于嵌入式系统 CPU 功耗预测并对其进行低功耗优化的方法.引入 GINI 指数的构建训练分类器,利用 PowerTop 工具对系统 CPU 进行监测,并以此作为训练数据集,将该分类器封装到系统中,对嵌入式系统的 CPU 频率、电压及所处的状态进行预测.通过仿真实验表明,该方法在系统负载较小的情况下,对嵌入式 CPU 功耗的优化的效果更好.

关键词 GINI 指数;功耗预测;机器学习;CPU 功耗;移动互联网

中图法分类号 TP3 **DOI 号** 10.3724/SP.J.1016.2015.00397

A Method of the Power Consumption Prediction of Embedded CPU Based on the GINI Index Classification Method

WANG Hai GAO Ling SONG Zhen-Xiao DAI Xiao-Ping LU Yi-Jie

(*School of Information Science & Technology, Northwest University, Xi'an* 710069)

Abstract In this paper, a method to predict and optimise the embedded CPU power consumption is put forward. In the method, GINI index method is introduced to construct the classifier. PowerTop tools monitor the CPU and collect the training data set, then encapsulate the classifier to the system as prediction module to predict the CPU frequency, voltage and the state of the system CPU. Through the simulation experiment shows that the method under the condition of the system load is small, the optimization of embedded CPU power optimization effect is better.

Keywords GINI index; power prediction; machine learning; CPU power consumption; mobile Internet

1 引 言

近年来,互联网已扩展到以智能手机作为代表性接入设备的移动时代.根据 2013 年 2 月 14 日全球技术研究和咨询公司据分析机构 Gartner 发布的新闻公告显示,虽然 2012 年全球手机销量为 17.5 亿部,比 2011 年下降 1.7%,但 2012 年智能手机的销量创记录的达到 2.07 亿,比上年同期增长 38.3%,预计 2013 年智能设备(智能手机及平板电脑)的销量将会突破十亿大关.其次,伴随智能终端数量的快速增长,移动应用的深度与广度也得到进一步的发展,并几乎涵盖了传统网络应用的所有层面.现在的问题是移动应用增加了,移动终端的功能增加了,但移动终端的供电能力却没有同比例的先行增长,特别是在保证移动重点性能的前提下如何来降低功耗的问题没有得到有效的解决.因此要使得移动应用进一步深入普及,迫切需要解决"性能-功耗"优化问

收稿日期:2013-12-18;最终修改稿收到日期:2014-11-26.本课题得到国家自然科学基金(61373176)、教育部博士点基金(20116101110016)、陕西省自然科学基金(2012JQ8047)资助.**王 海**,男,1977 年生,博士,讲师,主要研究方向为移动网络管理、服务计算.E-mail:hwang@nwu.edu.cn.**高 岭**,男,1964 年生,博士,教授,博士生导师,中国计算机学会(CCF)高级会员,主要研究领域为计算机网络、服务计算.**宋振孝**,男,1986 年生,硕士研究生,主要研究方向为机器学习、功耗优化.**戴小平**,男,1983 年生,博士,主要研究方向为移动计算、机器学习.**卢怡杰**,男,1984 年生,硕士研究生,主要研究方向为机器学习、功耗优化.

题,以有效延长移动设备的待机时间及有效工作时间.如何更好的控制嵌入式软件功耗成为当今技术发展的一大热点问题.

高功耗的嵌入式设备不仅会降低设备性能指标,还会增加系统的不稳定性,电池温度每上升10℃,系统失效率将提高1倍[1].降低嵌入式设备功耗研究包括硬件低功耗设计和软件低功耗设计.

嵌入式系统的重要特点之一就是工作负载的不均匀性以及动态变化性,可以通过动态关闭设备或者动态调节处理器的工作电压来取得系统性能和功耗之间的平衡.而且由于系统支持ACPI[高级配置和电源管理接口],易于将低功耗设计策略嵌入到系统内核中.

鉴于以上原因,本文提出了一种基于GINI指数分类方法,将学习到的分类器作为预测模块,插入到已有的系统对CPU所处状态进行预测.以达到最大化的降低系统的功耗,提高其性能的目的.

2 相关工作

目前大量的优化方法集中在硬件级[1-4]、编译级[5]、系统级[6]等方面,还没有从根本上实现降低功耗的预期目标.

在对嵌入式系统功耗的研究中,基于不同的设计目标提出了很多的方法,例如文献[7]提出了基于CSP的构件化嵌入式软件能耗分析与评估研究方法,文献[8]提出了基于模拟器的嵌入式操作系统能耗估算与分析.文献[9]提出的基于并行度分析模型的GPU功耗优化技术,这些多数都是从指令级角度及硬件方面考虑的[10-13].但是从对系统CPU所处的状态进行功耗优化来考虑,这些方法都有其一定的使用局限性[14].

3 基于GINI指数的分类预测方法

分类是机器学习和数据挖掘的一种重要方法.分类是通过对已有标记数据集的学习,构建具有一定泛化能力的分类器,分类器能把未知数据映射到分类器可区分的给定类别.已有的分类方法有:决策树、贝叶斯网络、遗传算法、粗糙集、k-最临近法等等[15].

在已有分类方法中,决策树方法具有计算量小、易于理解、同时适应处理连续和离散数据,能够明确的给出最具区分能力的属性等优点,受到重视和被广泛的应用[16].本文提出的基于GINI指数的分类预测方法正是决策树的一种.

3.1 决策树算法

决策树方法通过对实例进行归纳学习,从无次序、无规则的实例集合构建起一个使用树型结构来表示的分类器.

构建决策树的分成两个阶段:决策树生成与决策树修剪.因为决策树创建相对复杂程度更高,因此我们讨论重点落在树的创建上.

决策树的建立是一个递归的过程,构造树的通用过程如下:

(1)从训练样本中选取有代表性的单个结点,开始构造树.

(2)如果所有样本都属于该结点类.则用该结点的类标号标记此树,称该结点为叶子结点.

(3)否则以信息熵的信息增益作为度量,选择分类能力最大的结点属性作为决策树的当前结点.

(4)针对每一个训练属性的值构建一个分支,并据此分支值对样本进行划分.

(5)算法递归形成各个划分上的判定树.如果某个样本已经出现在一个构造过的节点上,则该节点所有后代都不再进行考虑.

算法终止条件:

①样本的所有给定结点都属于同一类.

②可以进一步划分样本的属性已经不存在.出现这种情况后,就会使用一种称为多数表决的方法,将剩余的样本结点转换成树叶,以样本中元组个数所属类别最多的那个类别作为标记.

③如果某一分枝没有样本,则以样本中的多数类创建一个树叶.

属性选择度量有信息增益(information gain)、GINI指数等.

在树的构建过程中,有两个关键问题:

(1)分割点选取;

(2)如何分割数据.

决策树算法在对数据进行分类的时候,选取哪个属性作为分割点是一个重要的问题.可以用来作为判定分割属性的方法有诸如ID3,C4.5算法中根据信息熵原理总结的信息增益与信息增益率,CART、SLIQ以及SPRINT算法中的GINI指数等.虽然ID3与C4.5算法在对训练样本集的学习中能够挖掘尽可能多的信息,但是由于利用了信息熵的原理,因而其生成的决策树分支较多,规模较大,算法的可伸缩性与并行性较差.而使用GINI指

数来作为判定分割点的标准，则简化了生成决策树的规模，提高了决策树生成的效率. 并且在算法的可伸缩性与并行性方面的性能也有较大提高.

因此本文利用 GINI 指数来判定分割点，降低了计算任务的复杂性，从而达到减少在计算方面的开销的目的.

3.2　基于 GINI 指数的分类方法

分裂指数是用来度量属性分裂规则优劣程度的一个量度，即我们用来判定选择哪一个属性作为创建决策树的分割点的依据. 基尼指数(GINI Coefficient)是意大利经济学家基尼(Corrado Gini，1884～1965)于 1912 年提出的用于定量度量的确收入差异程度的指标.

当我们在选择分割属性的时候遇到离散型属性，首先将其属性分组，然后根据各属性的取值将其排序，以方便计算.

如果我们在选择分割属性的时候遇到连续型属性，首先对该连续型属性的值进行排序，得到一组有序的数据集，然后对该数据集使用一个称为游标的标尺来标记，将其分为两个部分，这样就相当于将其离散化成两个值. 分别计算这两部分数据的 GINI 指数，然后将游标向后移动一位，继续对数据进行划分，再分别计算这两部分数据的 GINI 指数，直到将数据划分完毕，对所计算的 GINI 指数进行比较，选择最小的那个作为这组数据的分裂指标[17]. 计算具体 GINI 指数的方法如下：

设数据集 S 有 n 条记录，分别属于 c 个互不相关的类，则

$$\mathrm{Gini}(S)=1-\sum_{j=1}^{c}P_j^2,$$

其中 $P_j=m/n$，m 为 S 中属于类 j 的记录个数.

如果集合 S 划分为两个子集 S_1，S_2，分别对应 n_1，n_2 条记录，那么这个分割的 GINI 指数就是

$$\mathrm{Gini}_{\mathrm{split}}(S)=\frac{n_1}{n}\mathrm{Gini}(S_1)+\frac{n_2}{n}\mathrm{Gini}(S_2).$$

若这个值越小，则这个分裂规则越好.

例如，现在有 n 个连续型属性，首先将此 n 个属性进行排序，对于排序完成的数据集，我们用一个称为游标的标尺来对数据进行划分. 这个游标主要是用来对我们所要处理的数据进行标识. 当我们进行数据处理时，移动游标，分别将数据划分为游标左端与右端两部分(Cbelow-Cabove)，即 1-(n－1)，2-(n－2)，3-(n－3)…(n－2)－2，(n－1)－1，并且这些数据都是按照从大到小的顺序排列的. 分别根据不同的 Cbelow 与 Cabove，计算出：

$\mathrm{Gini}_{\mathrm{split}}(S_1)=\mathrm{Gini}(S_1(1))+\mathrm{Gini}(S_1(n-1))$，

$\mathrm{Gini}_{\mathrm{split}}(S_2)=\mathrm{Gini}(S_2(2))+\mathrm{Gini}(S_2(n-2))$

…

$\mathrm{Gini}_{\mathrm{split}}(S_{n-1})=\mathrm{Gini}(S_{n-1}(n-1))+\mathrm{Gini}(S_{n-1}(1))$.

选择这些 Ginisplit 中的最小值，并以最小值的 Cbelow 与 Cabove 作为分割标准[18].

基于 GINI 指数的分类方法通过划分连续型属性值，将其离散化为两个值，大大简化了计算复杂度，提高了效率，是一种简单有效的方法，且其效果经过证明也是非常理想的.

4　Linux 内核中 CPU 的工作状态

基于 Linux 的嵌入式系统中，CPU 的工作功耗是整个系统中至关紧要的部分，如何降低 CPU 的工作功耗，关系到整个系统节能工作的成败. 对于处理器功耗的控制的一种手段就是降低 CPU 的工作频率以及使处理器长期处在无任务调度的睡眠状态下. 在 Linux 系统中通过 ACPI 可以实现对处理器的各种工作状态的控制，并且通过 CPUfreq 子系统可以实现对处理器动态频率的控制.

4.1　ACPI 中处理器的状态

ACPI 即 Advanced Configuration and Power Management Interface，高级配置电源管理接口[19]. 1997 年由 Intel、Microsoft、Toshiba 所共同制定提供操作系统应用程序管理所有电源管理接口. 在处理器电源管理中，它定义了处理器的 4 个状态：$C0$、$C1$、$C2$、$C3$，当处理器处于 $C1$、$C2$、$C3$ 态时，比 $C0$ 态能耗低，散热也低，处理器处于睡眠态，不执行指令，也就是所谓的低功耗工作状态.

(1) $C0$ 态

这是处理器正常工作状态，可以执行指令，也是正常能耗状态.

(2) $C1$ 态

通过“HLT”指令进入，所有处理器都支持此状态. 从 $C1$ 到 $C0$ 的唤醒时间最短，任何 CPU 唤醒事件，例如处理器产生中断，必须返回 $C0$ 态.

(3) $C2$ 态

处理器时钟和 I/O 缓冲停止，它比 $C1$ 态能耗低. 可通过 P-LV3 寄存器进入此状态. 从 $C2$ 态进入 $C0$ 态需要至少 100 ns. 在此状态时，任何 CPU 唤醒事件，也都需要返回 $C0$ 态.

(4) C3 态

与 C2 态类似,总线和 PLL 均锁定,内存关闭,保持 CACHE,因此能耗比较低,但 C2 态大,唤醒需一定时间. 在此状态时,任何 CPU 唤醒事件,也都返回 C0 状态.

CPU 的睡眠程度越深,能耗越低,因此为了节能需要尽可能让 CPU 更多处于 C1、C2、C3 等状态. 由于任务的不确定性导致 CPU 在此 4 种状态间频频切换,会产生大量因切换而带来的能耗开销. 对这些开销进行控制,可以更好的控制 CPU 的功耗.

处理器的电源状态如图 1 所示.

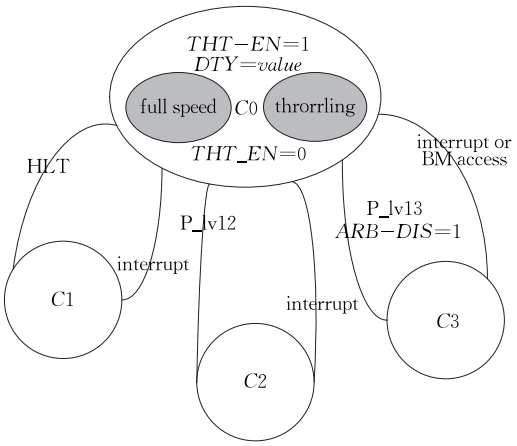

图 1 处理器电源状态

4.2 CPUfreq 子系统

2.6.10 Linux 以上所支持的 CPUfreq 子系统可以动态地调节处理器的频率. 可以用 CPUfreq 子系统和内核调控器来修改处理器的频率,改善系统的效率. 当处理器以较低的时钟速度运行时,它们消耗的电能和产生的热量也相对较少. CPUfreq 结构利用调控器来为系统设置静态或动态电源策略. 动态调控器可以根据 CPU 利用率来调整 CPU 频率,从而有助于节省电能,且不会牺牲性能.

图 2 显示了 CPU 频率,电压和功耗之间的关系. 所谓的 CPU 性能状态(P-state),是与 CPU 频率和电压相关的运转状态. C-state 对应相应的处理器工作状态,CPU 在不同状态之间进行切换的时候也会产生额外的功耗,如果 CPU 在进入一个休眠状态后需要立即唤醒,那么由此产生的功耗比处于休眠状态节省的功耗还要多,因此便达不到节省系统功耗的目的. 如何让 CPU 更多的处于低电压低频率的工作状态并且又不影响系统的整体性能即避免产生额外的功耗便是整个问题的关键. 这里我们提出了基于 GINI 指数的分类预测方法,对系统任务进行分类预测,从而使 CPU 更多的处于低电压低频率的工作状态,达到降低系统功耗的目的.

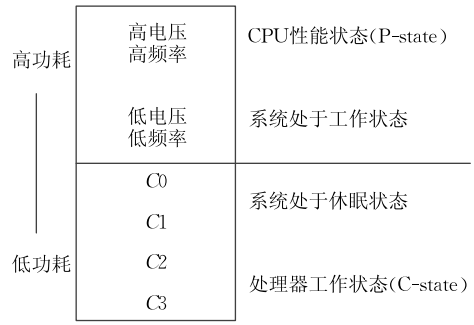

图 2 CPU 频率、电压与功耗关系

5 能耗预测模块的构造

建立能耗预测模块的过程可以分为 3 个阶段:

(1) 通过系统监测获取训练数据集;

(2) 使用基于 GINI 指数的分类方法通过训练集生成分类器;

(3) 能耗预测.

5.1 训练数据采集

本文通过 PowerTop 记录系统运行的各种参数以得到能耗预测模块所需的训练数据.

5.2 分类器训练

传统的动态电压调节算法都是按一定周期去检测 CPU 使用率,以此为根据动态调节 CPU 电压及频率进行"性能-功耗"优化. 动态调节 CPU 电压和频率一般分为两类:任务外(intra-task)方式和任务内(inter-task)方式.

这里主要介绍任务内(inter-task)方式.

根据依据的信息不同,任务内(inter-task)方式可分为 3 类:

(1) 基于任务最后期限作为 CPU 调节的依据,根据最坏情况下任务需要的时间,计算下一个时间段的大小. 由于难以获得任务的最后期限信息,因此此方法可行性不高.

(2) 实时检测任务执行过程,例如创建、删除、切换及系统调用等,据此作为调节 CPU 的依据. 检测任务的这些特征信息会带来额外的消耗,大约占 1%～4%.

(3) 按一定周期监测 CPU 使用率,通过对下一个时间段 CPU 使用率进行预测来调节 CPU. 这种方式实用性强,且实现简单.

在本文中,我们从另一个角度去考虑,每个任务

在执行的时候不是一直在占用 CPU，而是会不断唤醒 CPU，因为任务在执行的时候需要准备数据并做出相应的处理，然后当这些处理完毕的时候，才会去中断 CPU 的休眠状态，让 CPU 执行. 然后再做相应的准备工作，反复占用 CPU 运行，直到运行完毕.

首先，根据所获取信息，判断一个任务是属于 CPU 密集型任务(C)还是 I/O 密集型任务(IO)，不同类型的任务需要不同的运行策略，非 CPU 密集型任务可以暂时不去唤醒 CPU，而是等待合适的时间再去唤醒 CPU，这样可以让 CPU 在更多的时间内处于低电压低功耗的休眠的状态，然后根据任务单位时间内唤醒 CPU 的次数，计算其处于 $C0$ 状态(运行态)以及 $C1$，$C2$，$C3$ 等状态(低功耗态)的比例，并根据其不同的比例把任务分成高功耗任务与低功耗任务. 对于高功耗任务，我们让其更多的时间处于 CPU 较高频率与电压状态，即尽快执行完毕. 对于低功耗任务，我们则根据系统情况，让其更多的处于低电压低频率状态，从而达到降低系统功耗的目的.

根据所获得的历史数据，更新所有进程在已执行时间内的 CPU 使用率，设 Q 为最小时间段长度，每隔 Q 的时间后产生一个时钟中断. 进程 i 的 CPU 使用率为 e_i/I，其中 e_i 为该进程的执行时间，I 为计时器的间隔时间长度. 进程的执行时间需要根据系统最佳性能 进行标准化. 例如，假定 CPU 频率最大值为 600 MHz，某个进程在 CPU 频率为 300 MHz 时执行了 10 ms，标准化后其执行时间为 5 ms. 通过正在进行的优先级及时间片预测下一阶段将要运行的进程，算法进一步预测下一个周期 CPU 可以使用的频率，并根据系统所获取的数据，预测最佳的系统模式，这样做既保证进程的执行，又节约系统功耗.

5.3　预测算法

能耗预测算法过程如下：

(1) 首先分析任务运行中的参数值，包括任务类型、需唤醒 CPU 次数等；

(2) 根据分析得到的数值，使用分类器对任务所属类型进行分类；

(3) 根据(2)中分类结果，判断任务执行时合适的唤醒 CPU 时间以及预测下个任务执行时 CPU 所处的最佳状态，将结果返回给用户.

预测过程的算法如下.

算法 1.　预测过程算法.

输入：将在 CPU 中运行的任务样本 X

输出：任务类别 S

//输入为系统将要运行的任务样本 X

//输出为 x 的预测判断结果

//S 为判断任务运行中处理器所处的状态，分为 $S0$、$S1$、$S2$、$S3$($S0$ 中处理器 80%的时间处于 $C0$ 状态，并将处理器电压频率调至最高，其他时间处于 $C1$ 等低功耗状态. $S1$ 中处理器 60%的时间处于 $C0$ 状态，并将处理器电压频率调至比在 $C0$ 状态时低一些，$S2$ 中处理器 40%的时间处于 $C0$ 状态，并将处理器电压频率调至比在 $C1$ 状态时低一些，$S3$ 中处理器 20%的时间处于 $C0$ 状态，并将处理器电压频率调至最低)

1. $S=0$；

2. 从数据库表中取出任务 X 的数据；

3. 根据获得的任务 X 的运行状态参数，按照已训练好的分类器预测其运行中需唤醒 CPU 的次数，与外设的交互时间以及在处理器中的运行时间；

4. 根据预测的结果给出任务 X 运行时 CPU 的状态，S 的类别的建议；

5. 将预测的结果返回给系统；

6. 结束；

7. 返回最终的结果.

算法在判断每个任务所属的类型及其在运行过程中所需要唤醒的等 CPU 的次数以及由此引起的 CPU 的状态的变化方面具有简单、有效且差错率小的特点. 最终结果反馈给用户即系统，使系统确定其 CPU 可以处于哪一个低功耗的工作状态且不牺牲其性能，从而达到降低系统功耗的目的.

5.4　自动更新

能耗预测模块持续记录数据，定期重新生成分类器. 通过保持训练数据的时效性来确保预测的准确.

6　仿真实验分析

为了展示我们在系统电源管理上的优化效果，在这里通过模拟单一任务在单处理器中按照分类策略预测算法与无分类策略预测算法运行的情况，并进一步对两种运行方式的结果进行对比分析，可以发现我们提出的分类预测方法在系统能耗的节省方面达到了预期效果.

6.1　实验准备

本实验使用 MatLab 进行系统能耗仿真，使用 WEKA 分析仿真数据，仿真系统在单核情况下运行单任务. 假设没有任何能耗优化策略情况下系统的能耗为 1，即作为比较的基准. 通过系统仿真，观察

并记录系统在没有应用任何策略的情况下的运行情况,然后再观察并记录在应用了分类策略的情况下的系统运行情况,最后将运行结果进行汇总为对比分析做准备.

6.2 实验数据分析

获取训练数据与分类器训练的主要任务就是确定要研究的任务类型及其在系统运行中的能耗情况.

根据前面所提出的算法的需要,需要模拟程序运行过程中使处理器处于工作的时间,与外设进行数据交互的时间、唤醒 CPU 的次数.根据最后模拟的运行结果判断任务运行过程中处理器所应处的工作模式类型.

根据对已有研究成果的学习,系统任务的运行情况是一个随机的过程,对任务运行时具体的实验数据的提取存在一定的困难,因此在这里对数据需要进行一定的假设.利用 PowerTop 对系统运行的情况进行监控,可以发现任务在运行的过程中,前面提到的 3 个指标大体上服从一种平均分布,我们假设要模拟的实验数据服从平均分布情况,对其数据在一定范围内进行限制设置.

在仿真实验中,实验条件设置如下:

(1)任务在处理器中的运行时间($T1$)限制为 1～100,单位为毫秒(ms).

(2)任务与外设的交互时间($T2$)限制为 1～200,单位为毫秒(ms).

(3)任务唤醒 CPU 的次数($R1$)限制为 1～50,单位为次/毫秒(次/ms).

6.3 实验结果与分析

用 MatLab 随机生成两千组数据作为我们模拟的任务,每组数据包含 3 个数,分别代表处理器运行时间、与外设交互时间以及唤醒 CPU 次数,在这里我们不考虑其他因素对任务实际运行中的能耗影响.为了得到最终需要的分类结果,在这里我们为 $T1$,$T2$,$R1$ 分别赋予一个影响因子,考虑到对系统能耗的影响不同,分别赋予的因子为 0.8,0.2,0.05,结合影响因子计算出每个任务产生的系统能耗.

表 1 给出了模拟的任务数据,最后一项系统功耗是结合影响因子及处理器工作时处于不同的电压频率计算得出.通过表 1 的数据我们可以看到,当任务在处理器中运行的时间接近,而与外设的交互时间不一样时,例如任务 1 与任务 2,他们与外设的交互时间相差很大,但是最终消耗的系统能耗确相差不大,再看任务 1 与任务 1998,它们在运行过程中与外设交互的时间相差无几,但是在处理器中的运行时间相差很多,最终消耗的系统能耗相差了 3 倍之多,这说明与外设交互时消耗的能耗占整个任务运行期间消耗的系统能耗比例很低,唤醒 CPU 产生的系统功耗也只占总的系统能耗的一小部分,大部分的能耗由任务在处理器中运行时产生.图 3 反应了任务在没有应用任何运行策略时所产生的系统能耗的曲线图.在这里为了绘图的方便,我们选取前 100 个任务作为样本,使大家可以清晰的看到运行效果.

表 1 系统任务的 $T1$,$T2$,$R1$ 与功耗数据

任务号	处理器运行时间/ms	与外设交互时间/ms	唤醒 CPU 次数/(次/ms)	系统功耗
1	74.4867	85.4124	12	77.2525
2	74.2469	195.5538	4	98.7558
3	12.3860	83.7768	33	28.3499
4	3.4776	128.4898	11	29.0554
5	75.7838	16.9126	35	65.5721
6	83.7006	0.7718	42	69.2208
⋮	⋮	⋮	⋮	⋮
1998	5.3000	84.0857	13	21.7433
1999	97.1591	1.9955	11	78.6729
2000	15.0096	129.6011	36	39.7747

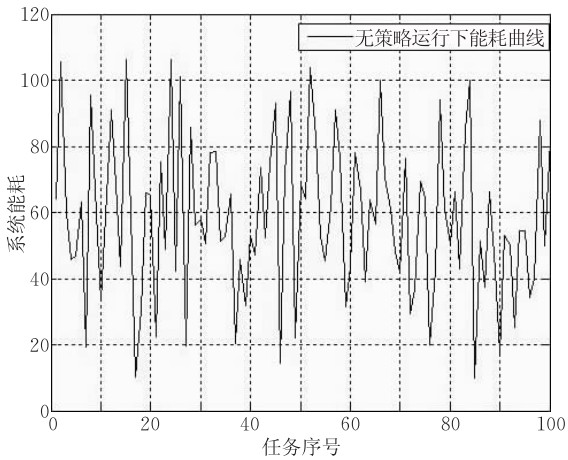

图 3 任务在没有应用任何策略的系统能耗

接下来使用 WEKA 来分类数据达到模拟分类器的效果,首先将数据按类别排序分别计算其 GINI 指数,计算方法按照前面提出的算法进行.按照计算所得的 GINI 指数对任务进行分类,对于高功耗的任务,使其处于分类结果中的 $S0$,然后依次按照计算得出的功耗数值分为 $S1$,$S2$,$S3$,将数据代入 WEKA 中,分类效果及分类树情况分别如图 4 与图 5 所示.

通过实验可以看到,我们所提出的分类算法在类型分类的正确率上达到了 96%,为接下来的系统预测算法提供了有效性的保证.

为了验证所提出的算法的节能效果,在这里我们将前面所选取的 100 个任务样本在应用分类算法之后重新进行运行分析,结果如图 6 所示.

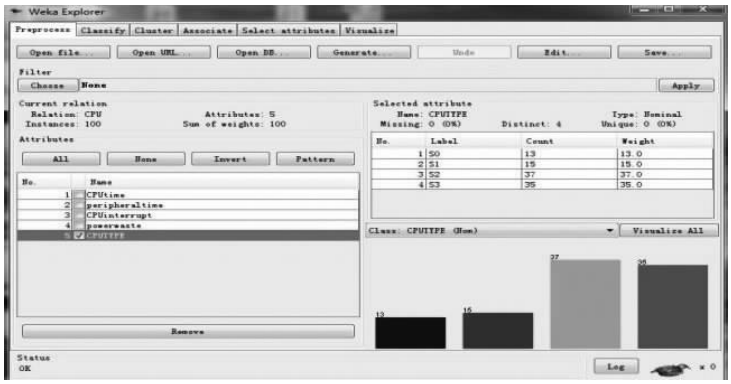

图 4　WEKA 数据分类效果

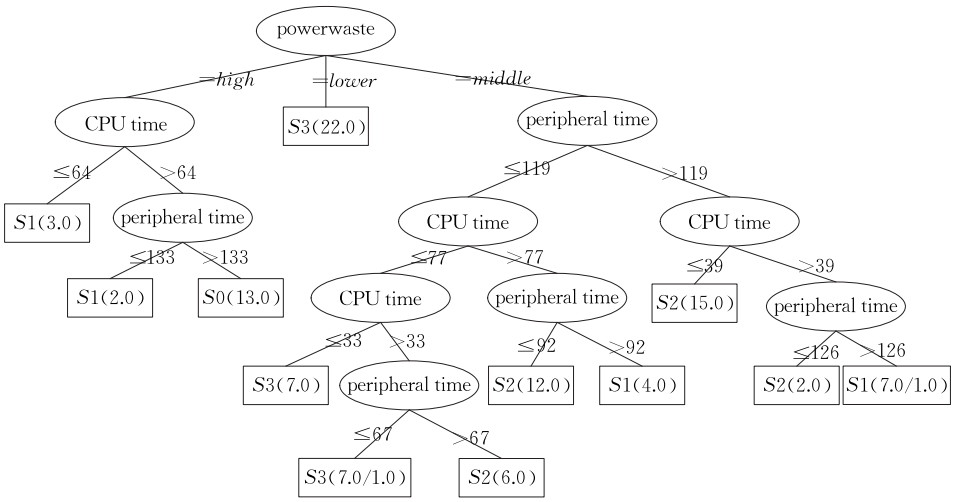

图 5　WEKA 分类树结果

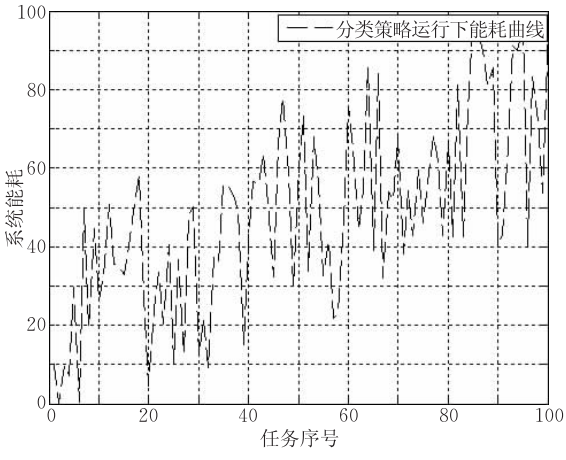

图 6　前 100 个样本的应用分类策略的运行效果

从图 6 可以看出，应用了分类策略之后，相对于系统运行在没有应用任何策略时，系统首先会将功耗较低的任务按照优先级尽量放在最前面执行，以使系统可以在不影响性能的前提前尽量减少系统功耗，而把高功耗的任务集中到一起，使系统在性能最优且尽量可以完成任务的前提下减少系统功耗.

对图 6 进行具体分析，系统开始的时候会利用模拟的分类器，首先根据获取的运行信息对的系统接下来将要运行的任务进行分类，判断其所属功耗类型，然后将其提交给系统并对系统的运行状态给出建议. 系统按照预测算法给出的预测结果建议进行运行，首先让更多的在分类的时候被划归于 S3 状态的任务先执行，在图中表现为一开始的系统处于消耗较低能耗的状态运行，此时 CPU 的电压频率会维持在一个较低的水平，因为此时运行的任务对系统的性能要求不高，所以总体上看并没有降低系统性能. 接下来对于能耗比较大的任务，系统会逐步提高其 CPU 的电压频率以满足任务的需求，并在此基础上尽可能的将系统的运行结束时间提前，从而达到功耗优化的目的. 与没有应用任何策略的运行结果相比较，应用分类策略运行时节省的系统能耗达到了 32.61%. 这在很大程度了减少了系统的能耗，达到了系统功耗的优化效果.

为了让大家有一个直观的观察效果，在这里我们将两者运行结果合并到一起，如图 7，从图中可以更加直观的看到我们所提出的算法的节能效果. 虽然算法在执行计算的时候会相对增加约 2%的系统消耗，

但是与取得的节能效果相比,增加的开销可以忽略.

为了进一步验证算法的有效性,我们从所模拟的实验数据中取出 8 组,每组 100 个,分别进行无任何策略的运行,加入分类算法策略的运行,将最终的结果进行对比分析,更加说明了算法的有效性.图 8～图 15 展示了我们所进行的 8 次模拟实验结果,从图中可以很直观地看到系统的节能效果.

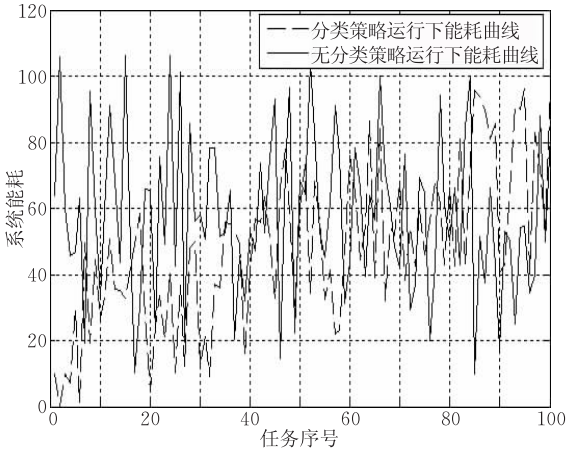

图 7 两种运行结果的比较

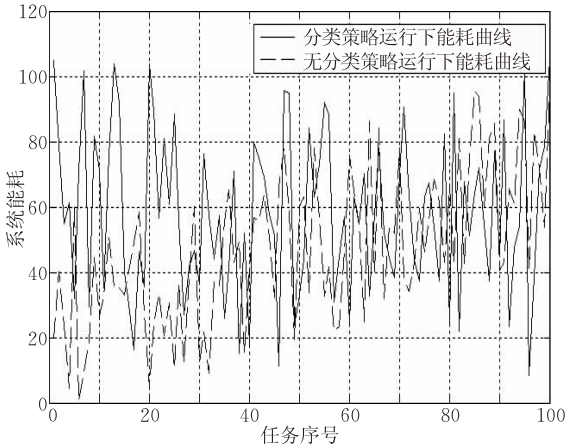

图 8 第 1 组两种运行结果的比较

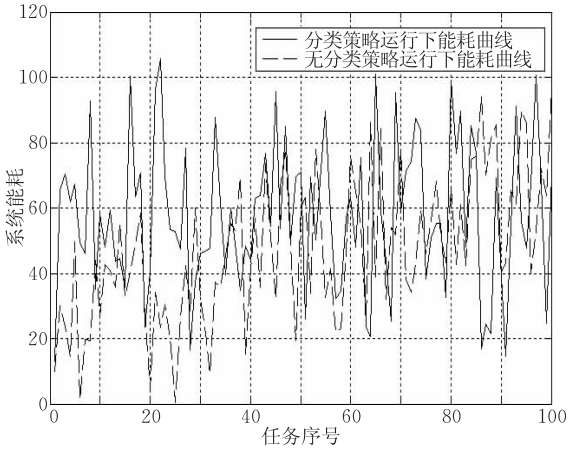

图 9 第 2 组两种运行结果的比较

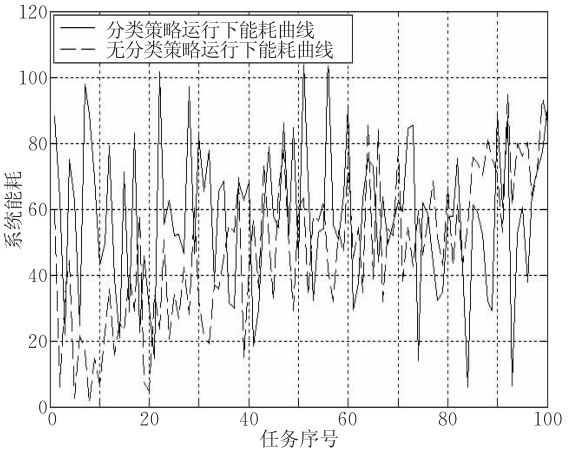

图 10 第 3 组两种运行结果的比较

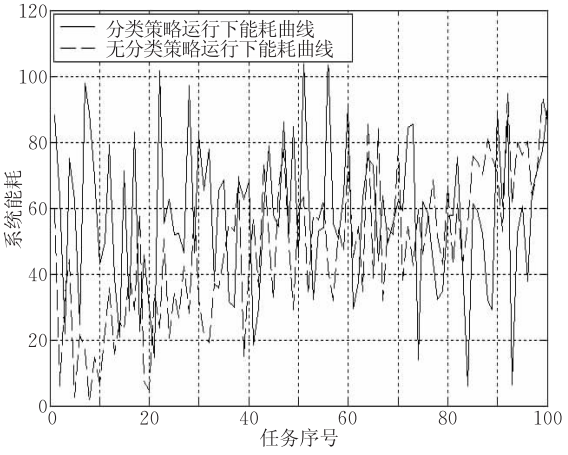

图 11 第 4 组两种运行结果的比较

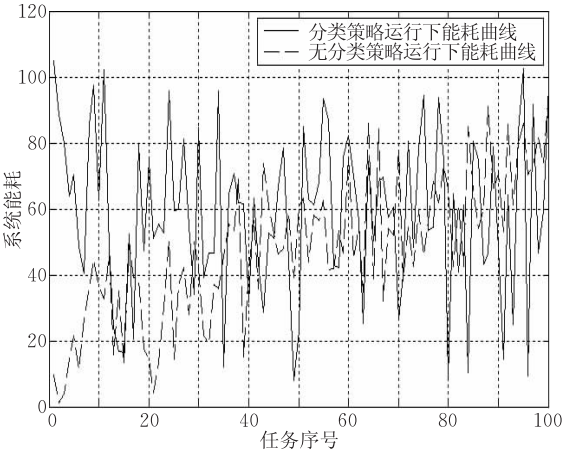

图 12 第 5 组两种运行结果的比较

对于 8 组实验的模拟结果,将在无策略运行下的能耗与在分类策略运行下的能耗进行统计分析,如表 2 所示.通过计算对比,8 组实验的系统功耗平均降低了 30.71%,基于这样一个实验事实,可以得到这样的结论:我们所提出的算法在系统功耗的优化方面确实取得了预期效果.

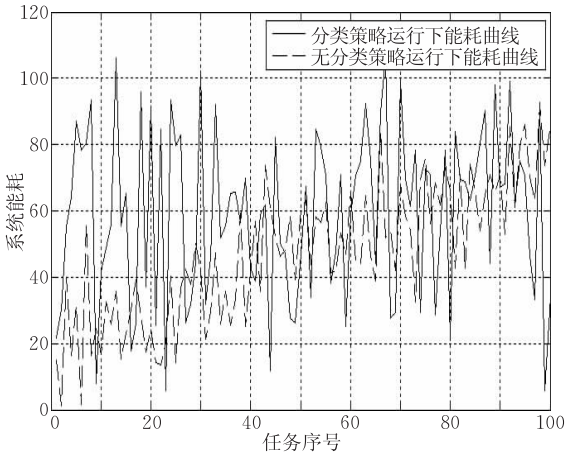

图 13　第 6 组两种运行结果的比较

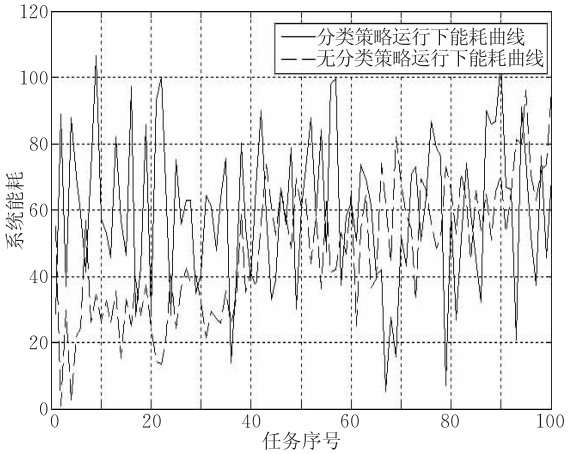

图 14　第 7 组两种运行结果的比较

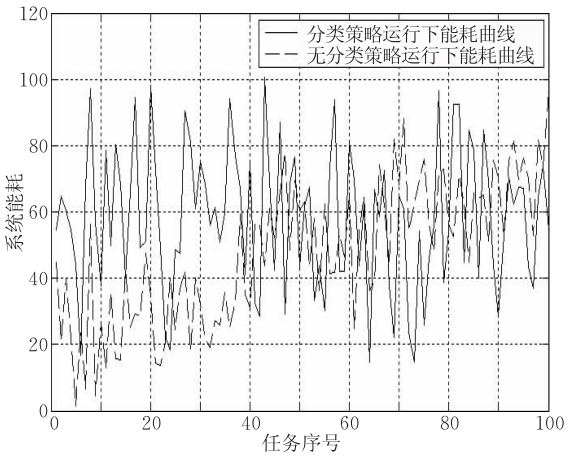

图 15　第 8 组两种运行结果的比较

表 2　8 组实验结果对比分析

实验组号	无策略运行下总能耗	分类策略运行下总能耗	节省的能耗	节省能耗比例/%
1	5826.6130	4134.1914	1692.4216	29.00
2	5788.0931	4025.7370	1762.3561	30.45
3	5428.4704	3861.7826	1566.6878	28.86
4	5734.1359	3906.4581	1827.6778	31.87
5	5870.8026	3751.2837	2119.5189	36.10
6	5941.0933	4083.9075	1857.1858	31.26
7	5960.0446	4126.3711	1833.6735	30.77
8	5832.8641	4234.5447	1598.3194	27.40

在这里我们还应该考虑到一个系统运行时间的问题，相比较于没有应用分类策略的系统运行时间，应用了分类策略之后，系统在运行时间上会有一定的增加，这是由于对系统的运行状态进行调整之后，有些任务在运行的过程中，处理器的状态会被调整到一个较低的电压及频率上，那么运行时间不可避免的要出现增加，图 16 显示模拟的 20 组实验中无分类策略运行与有分类策略运行时系统的能耗情况. 从图中我们可以看到，与没有运行任何策略的情况相比，应用了 GINI 指数分类策略之后，系统在能耗方面有了显著的降低了.

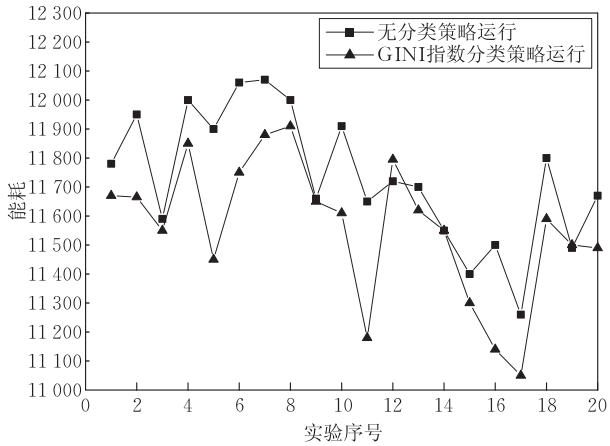

图 16　20 组实验不同运行策略时间开销图

实验的结果与理论计算的结果基本吻合，从实验模拟与理论计算两个方面都验证了结果的有效性. 虽然有时候在某些任务的运行中出现了系统能耗的增加，但是相对于系统整个运行过程中节省的能耗，增加的能耗完全在可以接受的范围.

7　结　论

本文将机器学习的理论应用于嵌入式 CPU 能耗的预测，提出的基于 GINI 指数分类策略的功耗优化预测方法，可以更好的对系统状态进行预测，对于降低系统功耗有明显的效果. 我们下一步的工作包括：进一步细化完善能耗的分析与评估的方法体系开发嵌入式系统的软件建模与验证的原型工具，并将继续加强对预测算法的优化，进一步探索更好的分类模型.

致　谢　匿名审稿专家对本文提出了宝贵修改意见，在此表示由衷的感谢！

参 考 文 献

[1] Feng W. Making a case for efficient supercomputing. ACM Queue, 2003, 1(7): 54-64

[2] Hao Jie, Peng Si-Long. Power driven circiut netlist multilevel partitioning algorithm. Journal of Computer Aided Design & Computer Graphics, 2009, 21(2): 190-195

[3] Wang Jiang-An, Zhuang Yi-Qi, Jin Zhao, Li Di. Optimization of embedded SRAM for low power and testing. Computer Science, 2010, 37(7): 301-303(in Chinese)
(王江安, 庄奕琪, 靳钊, 李迪. 嵌入式 SRAM 的低功耗优化及测试. 计算机科学, 2010, 37(7): 301-303)

[4] Ge Rong, Feng Xizhou, Cameron K W. Performance-constrained distributed DVS scheduling for scientific applications on power aware clusters//Proceedings of the 2005 ACM/IEEE Conference on Supercomputing. Seattle, USA, 2005: 34-44

[5] Saxe E. Power efficient software. Communications of the ACM, 2010, 53(2): 44-48

[6] Jian Da-Sheng, Li Xi, Wang Ai-Feng, Lei Ting. Optimization of system power based on dynamic voltage scaling. Computer Engineering, 2006, 32(1): 248-251(in Chinese)
(简大圣, 李曦, 王爱峰, 雷霆. 基于动态电压调节技术的系统功耗优化. 计算机工程, 2006, 32(1): 248-251)

[7] Zhang Teng-Teng, Wu Xiao, Li Chang-De, Dong Yun-Wei. On energy-consumption analysis and evaluation for component-based embedded system with CSP. Chinese Journal of Computers, 2009, 32(9): 1876-1883(in Chinese)
(张滕滕, 吴晓, 李长德, 董云卫. 基于 CSP 的构件化嵌入式软件能耗分析与评估研究方法. 计算机学报, 2009, 32(9): 1876-1883)

[8] Zhao Xia, Guo Yao, Lei Zhi-Yong, Chen Xiang-Qun. Estimation and analysis of embedded operating system energe consumption. Acta Electronica Sinica, 2008, 36(2): 209-215(in Chinese)
(赵霞, 郭耀, 雷志勇, 陈向群. 基于模拟器的嵌入式操作系统能耗估算与分析. 电子学报, 2008, 36(2): 209-215)

[9] Lin Yi-Song, Yang Xue-Jun, Tang Tao, Wang Gui-Bin, Xu Xin-Hai. A GPU low-power optimization based on parallelism analysis model. Chinese Journal of Computers, 2011, 34(4): 705-716(in Chinese)
(林一松, 杨学军, 唐滔, 王桂彬, 徐新海. 一种基于并行度分析模型的 GPU 功耗优化技术. 计算机学报, 2011, 34(4): 705-716)

[10] Takizawa H, Satol K, Kobayashi H. SPRAT: Runtime processor selection for energy-aware computing//Proceedings of the 3rd InternationaI Workshop on Automatic Performance Tuning. Tsukuba, Japan, 2008: 386-393

[11] Yang Qiao, Wu Xiao-Bo, Zhao Meng-Lian, Xu Jian. Method for systematic power consumption modeling and optimization of pipeline ADC. Journal of Zhejiang University (Science Edition), 2012, 39(2): 171-176(in Chinese)
(杨巧, 吴晓波, 赵梦恋, 徐建. 流水线模数转换器系统功耗建模与优化方法. 浙江大学学报(理学版), 2012, 39(2): 171-176)

[12] Liao Hai-Yan, Fan Ming-Ming. Testing and optimization of power consumption on source code and algorithm level. Microcontrollers & Embedded Systems, 2010, 1(1): 11-14(in Chinese)
(廖海艳, 范明明. 源码级和算法级的功耗测试与优化. 单片机与嵌入式系统应用, 2010, 1(1): 11-14)

[13] Luo Gang, Guo Bing, Shen Yan, Liao Hai-Yan, Ren Lei. Analysis and optimization method of energy consumption characteristics in embedded software based on source-code and algorithm level. Chinese Journal of Computers, 2009, 32(9): 1869-1875(in Chinese)
(罗刚, 郭兵, 沈艳, 廖海艳, 任磊. 源程序级和算法级嵌入式软件功耗特性的分析与优化方法研究. 计算机学报, 2009, 32(9): 1869-1875)

[14] Zhao Tie-Zhu, Dong Shou-Bin, Verdi March, et al. Predicting the parallel file system performance via machine learning. Journal of Copmuter Research and Development, 2011, 48(7): 1202-1215(in Chinese)
(赵铁柱, 董守斌, Verdi March 等. 基于机器学习的并行文件系统性能预测. 计算机研究与发展, 2011, 48(7): 1202-1215)

[15] Su Jin-Shu, Zhang Bo-Feng, Xu Xin. Advances in machine learning based text categorization. Journal of Software, 2006, 17(9): 1848-1859(in Chinese)
(苏金树, 张博锋, 徐昕. 基于机器学习的文本分类技术研究进展. 软件学报, 2006, 17(9): 1848-1859)

[16] Gey S, Nedelec E. Model selection for CART regression trees. IEEE Transactions on Information Theory, 2005, 51(2): 658-670

[17] Wei Hong-Ning. Study on the parallelism of decision tree classification based on SPRINT. Computer Application, 2005, 25(1): 39-41(in Chinese)
(魏红宁. 基于 SPRINT 方法的并行决策树分类研究. 计算机应用, 2005, 25(1): 39-41)

[18] Zhao Rui-Rui. Design and Implementation of Decision Tree Based on WEKA [Ph. D. dissertation]. Central South University, Changsha, 2007(in Chinese)
(赵蕊蕊. 基于 WEKA 平台的决策树算法设计与实现[博士学位论文]. 中南大学, 长沙, 2007)

[19] Lakkas G, Duduman B. The ACPI advantage for powering future-generation computers. Electrical Design News, 2001, 46(21): 91-92; 94-100

WANG Hai, born in 1977, Ph. D., lecturer. His research interests include mobile network management and services computing.

GAO Ling, born in 1964, Ph. D., professor, Ph. D. supervisor. His research interests include computer network and services computing.

SONG Zhen-Xiao, born in 1986, M. S. candidate. His research interests include machine learning and power optimization.

DAI Xiao-Ping, born in 1983, Ph. D. candidate. His research interests include mobile computing and machine learning.

LU Yi-Jie, born in 1984, M. S. His research interests include machine learning and power optimization.

Background

The work is supported by the National Natural Science Foundation of China under Grant No. 61373173 and the National Research Foundation for the Doctoral Program of Higher Education of China under Grant No. 20116101110016 and the Nature Science Foundation of Shaanxi Province under Grant No. 2012JQ8047. Because most of embedded systems are energy-constrained systems, the energy problem has become a hot topic in the field of embedded systems. At present, researchers have carried out a lot of works on energy consumption of embedded systems, however, most of which are based on instruction level or hardware, and few researchers focus on software architecture.

In this paper, the method to forecast and reduce the embedded CPU power consumption had been put forward. In this method, a training set is collected using the result of monitoring the system CPU by PowerTop, GINI index method is introduced to train the classifier, and then the trained classifier is packaged to a prediction module and embedded into the system to predict the frequency, voltage and the pattern of the system CPU. The test results show that the method for the system load is smaller, more conducive to the CPU power control.