基于模拟器的嵌入式操作系统能耗估算与分析

赵 霞[1,2],郭 耀[1],雷志勇[1],陈向群[1]
(1. 北京大学信息科学技术学院软件研究所,高可信软件技术教育部重点实验室,北京 100871;
2. 北京工商大学计算机学院,北京 100037)

摘 要: 随着嵌入式系统低能耗技术研究的深入,软件对系统能耗的影响越来越受到人们的关注,并向着定量分析方向发展. 本文提出一种嵌入式操作系统能耗量化分析方法,通过模拟运行嵌入式操作系统和应用软件,利用微体系结构能耗模型估算单时钟周期指令能耗. 提出基于软件功能结构的操作系统内核能耗估算模型并估算分析内核执行路径、服务、例程、原子函数的能耗,发现操作系统内核中显著影响系统能耗的关键软件模块及其特征. 实验结果表明,本方法可以从很大程度上提高嵌入式操作系统能耗估算和分析的准确性,估算结果有助于嵌入式操作系统能耗的量化分析和操作系统及应用程序的能耗优化设计.

关键词: 嵌入式操作系统; 能耗估算与分析; 量化方法
中图分类号: TP393 **文献标识码**: A **文章编号**: 0372-2112 (2008) 02-0209-07

Estimation and Analysis of Embedded Operating System Energy Consumption

ZHAO Xia[1,2], GUO Yao[1], LEI Zhi-Yong[1], CHEN Xiang-qun[1]
(1. *Institute of Software, School of Electronics Engineering and Computer Science, Peking University, Key laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, Beijing* 100871, *China*;
2. *College of Computer Science and Technology, Beijing Technology and Business University, Beijing* 100037, *China*)

Abstract: With the progress of low-power research on embedded systems, the estimation and analysis of the energy consumption of operating system becomes a hot topic. This paper presents a quantitative approach to estimate and analyze the energy consumption of embedded operating system (EOS). In this approach, EOS and applications are executed on two cooperated simulators. Instruction execution energy is estimated using a cycle-accurate micro-architecture power model. We propose an OS energy consumption estimation model based on software functionality and structure. The approach can calculate the energy consumption for functions, routines, services and kernel execution paths in an EOS, and can identify the key modules and factors impacting the system energy consumption. Our experiments show that the proposed approach improves the accuracy and efficiency of energy estimation of EOS significantly. The estimation results can be used to quantitatively analyze and optimize the energy consumption of EOS and applications.

Key words: embedded operating system; energy consumption estimation and analysis; quantitative approach

1 引言

随着嵌入式系统处理器复杂度提高和软件功能日益复杂,系统功耗不断增加. 不仅硬件的设计实现对系统功耗起决定作用,系统中的软件,特别是操作系统的算法、结构等设计实现也越来越显著地影响系统能耗[1,2]. 为了分析影响系统能耗的关键软件设计要素,评估软件优化对系统能耗的影响,以及为动态电源管理决策提供支持,需要在软硬件设计早期阶段从软件角度对计算机系统中的各个组成部分进行能耗分析与评估. 计算机系统中软件能耗模型及评估方法的研究[3~5,8~15],成为近年来低能耗研究领域的新热点. 操作系统对系统能耗的影响以及如何优化操作系统提高系统能耗效率的问题也成为其中备受重视的问题[6,7,16,18,19].

功耗和能耗评估原本是硬件领域的概念,根据部件物理结构和电路参数等建立能耗模型[9]进行评估是常用方法之一. 1994 年,Tiwari 等人认为仅从硬件结构角度评估系统能耗无法满足系统级能耗评估的需求,提出软件能耗的概念. 软件能耗是指令在处理器上执行过程

收稿日期:2007-05-14;修回日期:2007-08-21
基金项目:国家 863 高技术研究发展计划(No. 2007AA01Z462, No. 2007AA010304)

中系统部件消耗的能量[2]. 研究软件能耗的目的是通过把硬件能耗映射到软件结构和软件特征上,来研究软件成分对处理器功耗和能耗的影响,并利用软件设计优化技术提高系统能耗效率.

基于能耗模型和模拟器的方法是目前软件能耗估算的一种主流方法,它能够在软硬件系统开发早期阶段进行软件能耗估算和分析,不干扰软件系统的运行,不需要修改操作系统和应用程序代码. 微体系结构能耗模型[9]可以精确地估算指令执行过程中处理器各部件的能耗,比其他模型[2,10]更易于提高软件能耗估算的准确度. 操作系统能耗估算比用户态软件能耗估算更为复杂,需要全系统模拟器支持操作系统模拟运行. 现有的基于 SPARCv9 体系结构[11]和 MIPS R10000 体系结构[12]的全系统模拟器,分别针对 Solaris 操作系统和 SGI IRIX5.3 操作系统进行能耗分析. 这些方法针对服务器硬件和软件,其中的能耗估算方法和工具无法直接用于嵌入式操作系统能耗估算研究. Austin 等人在 SimpleScaler[13]的基础上开发的 PowerAnaylzer 能耗模型[14]和估算工具[15]虽然针对嵌入式处理器体系结构,可以估算用户程序能耗;但不能用于操作系统能耗估算. 钟伟军[16]等人提出支持嵌入式操作系统的 ARM 能耗模拟器,模拟估算每个并发运行程序的能耗,但不能用于分析嵌入式操作系统内核能耗. 因此,目前尚未见到能够细化分析嵌入式操作系统内核函数、例程等粒度的,基于模拟器和微体系结构能耗模型的能耗估算方法,也未见到对嵌入式操作系统内核能耗的深入分析.

本文提出的嵌入式操作系统能耗估算方法借鉴了 Austin 等人的微体系结构能耗模型,通过集成全系统指令模拟器和微体系结构能耗模拟器支持嵌入式操作系统的模拟运行. 提出基于软件功能结构的操作系统内核能耗估算模型,并估算内核执行路径、服务、例程、原子函数的能耗. 在此基础上,对内核能耗进行了深入的分析,发现操作系统内核中显著影响系统能耗的关键软件模块及其特征,并对软件能耗优化设计方法进行讨论.

2 嵌入式操作系统能耗估算方法

本文提出的能耗估算方法由三部分组成:操作系统目标代码的模拟执行;指令流的单时钟周期能耗的估算,不同粒度的操作系统内核模块能耗的估算. 分别由全系统指令级模拟器,微体系结构能耗模拟器和内核能耗估算器来实现,如图 1 所示.

全系统指令级模拟器模拟嵌入式微处理器指令集的指令功能和外围系统硬件,支持嵌入式操作系统模拟运行. 模拟器输入是嵌入式操作系统的可执行目标映像文件和包含应用程序的根文件系统,输出是动态执行的指令流地址和指令码给微体系结构能耗模拟器. 微体系结构能耗模拟器模拟处理器内部的流水线结构和组成部件的操作;按照各部件的能耗模型统计指令流执行过程中每个时钟周期微处理器各部件的能耗(称为单时钟周期能耗),发送到例程/服务能耗估算器. 内核能耗估算器合成操作系统内核中各种粒度代码模块的能耗. 本文提出基于软件功能结构分析的操作系统内核能耗估算模型:把运行时的操作系统内核看作由原子函数、例程和服务组成的软件实体和动态执行路径构成的逻辑实体,估算不同粒度的内核实体能耗.

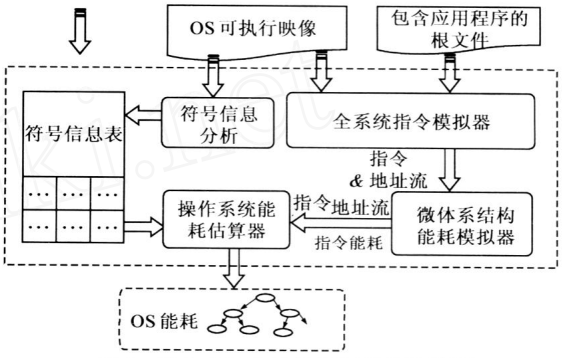

图 1 嵌入式操作系统能耗估算结构图

3 单时钟周期能耗估算与能耗模型

微体系结构能耗模拟器每个时钟周期从缓冲区中取一条指令,分析该指令经过各段流水线时操作的微体系结构部件(如 ALU、Cache、TLB 等部件),按照微体系结构能耗模型计算被访问部件的能耗. 能耗模型考虑连续指令执行过程中分支延迟、流水线停顿、控制流预测失效和 Cache 失效等对能耗的影响,并根据操作系统用户态到内核态转换调节微体系结构部件的能耗. 每个时钟周期累计该周期内流水线上所有被访问部件的能耗和访存能耗,作为该时钟周期内的处理器能耗——单时钟周期能耗.

微体系结构部件可以抽象为 Datapath、Cache、Clock 和 Memory 四类和其他逻辑单元. 其中,Datapath,Cache 和 Clock 部件用基于 CMOS 电路的功耗计算公式. 设 V 表示供电电压,f 表示微处理器时钟频率,C 为部件所有门输出结点电容之和,α 是介于 0 - 1 之间的因子,表示门翻转的概率,I_{sort} 表示短路电流,I_{leak} 表示泄漏电流. 基于 CMOS 技术的半导体器件功耗计算公式为[17]:

$$P = \alpha CV^2 f + t\alpha VI_{short} + VI_{leak} \quad (1)$$

设 N 是访问部件所用时钟周期数,则能耗的计算公式为[17]:

$$E = N \times P/f \quad (2)$$

Memory 部件的能耗与活动状态的电流和访存时钟

周期数相关,I_{active}是处于活动状态时的电流,N 是一次访存的时钟周期数.其能耗计算公式如下[18]:

$$E = N \times V \times I_{active} / f \quad (3)$$

根据以上部件能耗模型计算指令经过每个流水段时各部件能耗获得单时钟周期能耗 e_i,计算公式是:

$$e_i = e_{datapath} + e_{cache} + e_{sram} + e_{clock} + e_{misc} \quad (4)$$

其中,$e_{datapath}$是在处理器流水线中数据路径能耗,e_{cache}是 Cache 访问能耗,e_{sram}是 TLB 操作能耗,e_{clock}是时钟电路能耗,e_{misc}是其他逻辑单元的能耗.

4　操作系统内核的能耗估算

基于软件功能结构分析的操作系统内核能耗估算模型把运行时的操作系统内核看作由原子函数、例程和服务组成的软件实体和动态执行路径构成的逻辑实体.原子函数由连续指令序列构成,不再调用子函数,是操作系统能耗估算的最小单位.例程是若干相互调用的函数集合.服务是通过硬件陷入机制开始执行,并且具有特定函数入口的例程.服务有三种类型:系统调用、异常服务和中断服务.内核执行路径表示一个从用户态进入内核开始,到返回用户态结束的完整执行过程,是服务和例程的集合.

4.1　原子函数的能耗估算

原子函数的能耗估算用指令地址标识单时钟周期能耗,累计原子函数中所有指令对应的单时钟周期能耗,作为该函数的能耗.

本文利用求和运算的可组合性并通过微体系结构模拟器处理,保证用函数所属的指令流的单时钟周期能耗之和来计算函数能耗的准确性.单时钟周期能耗不同于单条指令能耗:后者指该指令经过各段流水线部件时的能耗之和;而单时钟周期能耗是在同一个时钟周期里各段流水线部件的能耗之和,与相互关联的指令流有关.在指令顺序执行情况下,相邻指令均属于同一函数.在发生程序异常或中断时,模拟器执行完前面的指令并刷新流水线.当遇到函数调用和返回时,指令流的地址变为不连续指令,预取失效,模拟器刷新流水线,下一个函数的指令进入流水线之前,属于上一个函数的指令对应的能耗已经统计并输出,流水线停顿等的能耗算在上一个函数中.

估算函数能耗的另一个问题是判断指令是否属于某函数,关键是识别函数调用和返回.StrongARM 体系结构平台上的嵌入式操作系统代码,利用堆栈操作、改程序计数器、中断异常等方式改变指令流进入或离开一个函数,识别指令码方法不可行.我们构造函数符号信息表和地址匹配方法来正确识别函数调用与返回,并通过构造函数调用树来识别更复杂的程序结构.

4.2　例程和服务的能耗估算

例程是具有调用关系的函数集合,用函数调用树来表达,例程能耗由该节点的子孙节点能耗递归计算得到,即调用树根节点的能耗,记为 $E_{Routine} = E_{f_0}$.设例程的函数调用树由节点 $f_0, f_1, \cdots, f_n$ 组成,每个节点对应一个函数,f_0 是根节点.函数 f_i 的能耗为 E_{f_i},属于该函数的指令集合为 I_{f_i},其子函数为 $f_{j1}, f_{j2}, \cdots, f_{jm}$,($i < j_1 < j_2 < \cdots < j_m < n$),指令的单时钟周期能耗为 e_i;则函数调用树上任意节点的能耗,是其子函数能耗与直接属于函数的指令能耗之和:

$$E_{f_i} = \begin{cases} \sum_{i \in I_{f_i}} e_i, & \text{不调用子函数} \\ \sum_{i \in I_{f_i}} e_i + \sum_{k=1}^{m} E_{f_{jk}}, & \text{调用子函数} \end{cases} \quad (5)$$

构造函数调用树的过程如图 2 所示.

```
repeat:
    从缓冲队列中获得指令的地址和能耗;
    查找指令所属函数,将名称赋给 func;
    IF (系统调用或异常入口) THEN 创建新树;
    IF (返回用户态) THEN 结束旧树建新树;
    cur_node 指向当前节点;
    IF (func 是 cur_node) THEN
      把指令能耗累加到当前节点上; CONTINUE;
    沿父指针和子指针查找,返回 NULL 或节点;
    IF(找到节点) THEN
        pre_node 指向当前节点;
        cur_node 指向找到节点;
        把 pre_node 能耗加到 cur_node 上;
        CONTINUE;
    为当前函数创建新节点作为当前节点子节点;
```

图 2　内核能耗估算算法

内核服务通过硬件陷入机制从特定程序入口被调用执行,通过识别特定入口地址,从指令流中发现内核服务,并为内核服务建立函数调用树,按照式(5)估算其能耗.

4.3　执行路径的能耗估算

已有的能耗分析方法[18,19]认为一个系统调用的能耗是从系统调用进入内核开始到返回用户态的软件能耗.实际上,从系统调用进入内核到返回用户态过程中,会随机性地存在中断服务等与系统调用无直接关系的内核例程/服务.因此,把这个过程的能耗全部作为系统调用的能耗会造成估算错误.

本文把从进入内核开始到返回用户态过程执行的服务和例程的集合定义为内核执行路径,并提出两种内核服务类型.显式服务定义为用户显式调用的系统调用服务,且仅包括实现该服务功能的例程,例如 sys_

read ,sys_fork 等;隐式服务定义为对应用程序透明的,不属于当前执行逻辑的内核服务,例如中断、异常处理等.

由此,内核执行路径能耗计算分为两种情况:(1)单一执行路径:是不被中断的显式系统调用服务执行路径,其能耗为该显式服务的能耗;(2)复合执行路径:当显式服务被中断时,执行路径对应于该路径上出现的显式服务、中断服务、进程调度例程等多棵子树构成的函数调用树,其能耗为所有子树能耗的和. 综上所述,执行路径能耗计算公式为:

$$E_{execpath}=\begin{cases} E_{explict\_service}, & \text{单一执行路径} \\ E_{explict\_service}+E_{implict\_service}+E_{Routine}, & \text{复合执行路径} \end{cases} \quad (6)$$

5 操作系统内核能耗分析

本文按照上述方法实现了能耗估算工具模拟 StrongARM-1100 处理器嵌入式硬件平台,对 ARMLinux 2.4.18 内核进行能耗分析. 工具包括经过改进的全系统指令模拟器 Skyeye[20] 和微体系结构能耗模拟器 PowerAnalyzer[21],以及内核能耗估算器. StrongARM 处理器的微体系结构参数见表 1. 测试程序集包括嵌入式系统工具包 busybox 和自己开发的消息通信测试程序(见表 2).

表 1 StrongARM-1100 处理器微体系结构参数

Parameter	Value
Feature Size	0.18μm
V_{dd}	1.5V
Frequency	200MHz
Fetch/ Issue/ Retire Width	2(inorder)
RUU/ LSQ size	4
L1 I-Cache	16KB(32B cache line ,2-way assoc.)
L1 D-Cache	8KB(32B cache line ,32-way assoc.)
TLB (full assoc) entries	32

表 2 部分测试程序

功能类型	交互类测试程序	非交互类测试程序
文件操作	cat ,ls ,ln ,echo ,more ,du	rm ,mount ,cp ,gzip ,gunzip ,tar
网络命令	ifconfig ,ftpget ,telnet	netstat
进程用户管理	ps ,whoami	kill ,chgrp ,
消息通信程序	Msg client	Msg server

能耗估算分析工具实时估算操作系统内核能耗,以层次列表形式输出函数调用树及其每个节点能耗(见图 3),每条记录包括例程或函数名称、微体系结构能耗以及内存访问能耗.

```
sys_read(micro :17.005 ,mem :0.000)
    fget(micro :8.527 ,mem :0.000)
    generic_file_read(micro :3.336 ,mem :0.000)
    fput(micro :1.130 ,mem :0.000)
```

(a) sys_read 在 generic_file_read 执行路径上的能耗

```
sys_read(micro :523.169 ,mem :0.000)
    fget(micro :3.106 ,mem :0.000)
    proc_file_read(micro :428.426 ,mem :0.000)
        ...
        free_pages(micro :8.810 ,mem :0.000)
    fput(micro :1.084 ,mem :0.000)
)
```

(b) sys_read 在 proc_file_read 执行路径上的能耗

```
sys_read(micro :14962.155 ,mem :164.220)
  fget(micro :3.106 ,mem :0.000)
  tty_read(micro :14953.539 ,mem :139.230)
      ...
      cpu_idle(micro :14924.231 ,mem :42.840)
      _irq_svc(micro :31.205 ,mem :0.000)
  fput(micro :1.132 ,mem :0.000)
)
```

(c) sys_read 在 tty_read 执行路径上的能耗

图 3 不同执行路径下的例程能耗

5.1 不同类型程序下服务调用次数及能耗特征

从软件能耗分析的角度,本文把测试程序划分为非交互型和交互型两类(如表 2 所示). 交互型程序在执行过程中要等待用户输入或者向屏幕输出;非交互型程序开始执行后不进行用户交互.

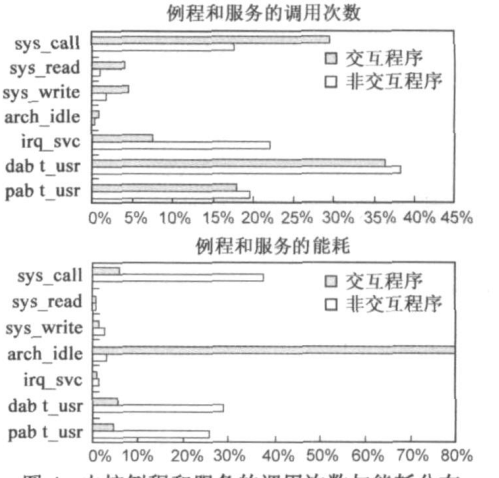

图 4 内核例程和服务的调用次数与能耗分布

从图 4 可见两类程序对应的内核能耗特征如下:

(1)交互型程序的处理器空闲(arch_idle)能耗所占比例显著高于非交互性程序,前者占总能耗的 81 %;后者占总能耗的 3 %. 人机交互造成大量的处理器空闲造成能耗浪费.

(2)缺页异常处理(dabt_usr)和预取异常处理(pabt_usr)执行次数和能耗所占比例显著. 在所有的内核服务中,缺页和预取异常处理执行次数最多,在交互程序中占总数的 54 %,在非交互程序中占总数的 57 %. 原因是在 StrongARM 平台上的 ARMLinux 2.4 采用虚拟存储器

技术,通过频繁的缺页异常和预取异常处理把代码和数据加载到内存.

(3)系统调用次数与能耗在两类应用程序中有较大差别.在交互程序中,系统调用次数之和占总数的37%,比非交互程序中所占比例高17%,但能耗之和所占比例只有8%,比后者低34%.可见,非交互程序中系统调用能耗是较主要部分,由于空闲能耗比例较大,所以交互程序中系统调用能耗不太显著.

综上所述,交互式系统中处理器空闲是能耗主要来源,而非交互程序中的异常处理是能耗主要来源,优化这部分内核例程和服务是降低内核能耗的途径之一.另外,通过改进设备I/O例程功能和内核调度策略,或通过动态功耗管理和电压/频率调节减少处理器空闲和高功耗状态也可以达到降低系统能耗的目的.

5.2　例程/服务的功能与能耗关系分析

分析结果表明,内核例程/服务能耗与其功能的复杂度有较大关联.表3给出部分内核例程/服务能耗估算结果的统计和分类.

表3　部分例程与服务的能耗

类别	例程/服务名	执行次数	能耗(uJ)			
			平均值	最低值	最高值	标准偏差
ID操作	sys_getegid	25	4.99	3.52	9.51	1.70
	sys_geteuid	20	3.12	2.143	7.59	1.91
	sys_getgid	20	2.17	1.95	2.71	0.18
	sys_getpgrp	19	3.03	2.15	4.97	0.97
	sys_getpid	19	1.55	0.20	2.17	0.41
进程控制	_switch_to	110	1.24	1.08	1.78	0.16
	_tasklet_schedule	69	1.09	0.93	1.66	0.19
	_wake_up	427	2.78	0.20	5.97	0.77
	wake_up_process	30	5.48	3.04	8.57	1.30
	schedule	113	8.77	1.04	28.67	7.80
内存访问	sys_brk	147	12.21	4.45	29.60	7.00
	sys_fork	24	89.40	73.10	113.73	11.04
	sys_execv	24	453.31	372.00	1219.60	164.83
	sys_exit	21	138.34	115.81	188.50	19.09
	sys_clone	6	43.88	25.18	72.39	16.24
文件系统	sys_stat64	77	53.34	20.25	243.36	29.11
	sys_open	44	71.63	29.2	445.94	62.105
	sys_write	291	80.97	38.72	204.94	32.71
	sys_read	266	44.74	8.10	355.00	40.78
	sys_close	66	78.44	8.02	330.58	93.59
IO读写	tty_read	197	23.74	12.47	35.474	4.64
	tty_write	291	71.88	31.46	197.78	32.31
异常中断处理	_dabt_svc	39	46.04	32.98	165.15	20.53
	_pabt_user	1211	34.27	191.27	402.69	37.36
	_irq_svc	244	34.12	9.50	86.57	12.41
	do_timer	84	16.79	6.62	32.05	4.70

根据功能和能耗特征,可以把内核例程/服务分为ID操作、进程控制、内存访问、文件系统、I/O读写、异常中断处理六大类.其能耗特征如下:

(1)功能简单的例程能耗低,能耗值稳定.由原子函数组成的ID操作服务,如sys_getpid等读取进程信息,能耗平均值不超过5uJ.部分内核例程,如进程上下文切换(_switch_to)、唤醒进程(wake_up_process)能耗平均值不超过10uJ,标准偏差不超过2.

(2)功能复杂的服务和内核执行路径在不同情境下能耗差别大,如进程管理、内存管理、文件系统服务.从图3的函数调用树及能耗估算结果可见,系统调用引发的内核执行路径在不同情况下代码执行量和能耗差别很大,值得仔细分析.

(3)需要关注单次执行能耗显著和总能耗显著的内核例程/服务.进程管理相关的服务有较多的内存和外存操作,单次执行能耗较大.如sys_execv加载新程序执行,能耗开销为453.31uJ.又如sys_fork进程创建服务,能耗为89.40uJ;线程创建服务sys_clone能耗为43.88uJ,如果用内核线程代替进程来实现操作系统并发操作,可减少进程管理带来的能耗、时间和内存开销.另外,通过减少这类例程/服务的调用次数或者优化这些服务/例程是降低系统的能耗的重要途径.

5.3　内核执行路径中的隐式服务能耗

我们对由系统调用引发的内核执行路径能耗进行深入分析,发现异常、中断、进程调度等内核服务/例程等隐式能耗是内核能耗中不可忽视的部分,虽然不被应用程序所感知,但由于其随机性和频繁性,会对系统调用能耗估算带来误差.目前尚未看到相关研究涉及该问题.

表4　sys_read执行路径中显式和隐式服务能耗试验结果

内核执行路径	序号	总能耗	显式调用(uJ)	Tty中断服务(uJ)	进程调度(uJ)	Idle进程(uJ)	Timer中断	显式服务能耗所占比例
tty_read异步读	1	1705	41	130	10	1476	48	2.40%
	2	30603	20	226	9	30301	47	0.06%
	3	15689	30	169	4	15435	51	0.19%
generic_file_read	1	123	78	45	0	0	0	63.41%
	2	137	85	52	0	0	0	62.04%
	3	40	40	0	0	0	0	100%
	4	13	13	0	0	0	0	100%

以sys_read为例,表4给出系统调用sys_read陷入内核后,执行路径上显式服务和隐式服务执行情况及其能耗.表中列出两种执行实例,一种是通过tty_read异步读用户从键盘输入的字符.内核执行路径包括tty设备驱动程序等待输入,进程进等待队列,执行空闲进程,用户输入引发中断处理,最终把输入数据复制到用户缓冲区.此情况下,执行了tty中断、进程调度、Timer中断等隐式服务,特别是Idle进程消耗了大量的时间和能量,显式服务的能耗所占的比例不到3%.另一种generic_file_read读取文件,调用数据缺页异常服务把文件从Flash读到内存,显式服务所占的比例为62.04%

～63.41 %.

5.4 关于操作系统内核能耗优化问题的讨论

从上文分析结果可以看出,操作系统内核能耗具有一定的规律与特征,操作系统内核对系统能耗的影响与以下几方面因素有关:

(1) 应用程序对内核服务的调用次数决定内核对系统能耗影响所占的比例;

(2) 不同功能的内核服务例程,由于访问资源和算法不同,能耗差异较大;

(3) 由用户交互造成的处理器空闲等待是交互系统中处理器能耗主要来源。

根据以上因素的分析,可以从以下几方面优化设计来降低系统能耗:

(1) 优化应用程序结构和流程设计,减少对能耗显著例程/服务的调用次数,或改进进程间通信方式;

(2) 优化操作系统的进程管理、异常和预取处理等例程/服务,如减少进程调度次数,或减少不必要的中断和异常处理;

(3) 改进 I/O 例程/服务功能与设计和内核调度策略,或通过动态电压/频率调节减慢程序执行速度,减少处理器空闲。

此外,也可以根据本方法得出的能耗估算结果设计能耗估算宏模型[12,22],利用宏模型来评估和优化操作系统和应用软件设计。

6 结论

目前没有适用的测量方法来分析操作系统内核的软件能耗,而低能耗设计领域又非常需要能够评价和分析嵌入式系统软件能耗的手段. 本文提出了一种基于模拟器的估算嵌入式操作系统能耗的方法,并对嵌入式操作系统进行了能耗估算和分析. 本文的贡献在于:(1) 用基于微体系结构能耗模型的模拟器估算操作系统能耗,达到时钟周期精度. (2) 基于指令流地址分析过滤方法,有效判别函数调用关系;(3) 提出基于软件功能结构的操作系统内核能耗估算模型,实现对操作系统内核函数、例程、服务和内核执行路径等多种粒度能耗分析. (4) 通过分析发现影响内核能耗的隐含和随机因素,对正确估算应用程序和操作系统能耗有重要意义. 该方法有助于提高嵌入式操作系统内核能耗分析和优化的准确性,有效支持操作系统及应用软件的能耗评估与优化设计. 本方法独立于具体的处理器类型和操作系统,可以与其他指令级模拟器和微体系结构能耗模拟器集成为新的操作系统能耗估算工具.

参考文献:

[1] 李允,罗蕾,熊光泽. 面向普适计算的自适应技术研究[J]. 电子学报. 2004,32(5):740 - 744.
Li Y,Luo L,Xiong G. The Adaptive Technology for Pervasive Computing[J]. Acta Electronica Sinica,2004,32(5):740 - 744. (in Chinese)

[2] Tiwari V,malik S,wolfe A. Power analysis of embedded software:A first step towards software power minimization[J]. IEEE Transactions on Very Large Scale Integration. 1994,2(4):437 - 444.

[3] Hu C,Jimenez D A,Kremer U. Toward an evaluation infrastructure for power and energy optimizations[A]. Proceedings 19th IEEE International Parallel and Distributed Processing Symposium[C]. Los Alamitos,California:Ieee computer society Press,2005. 8 - 14.

[4] Chandra K,Selim G. A run-time,feedback-based energy estimation model For embedded devices[A]. Proceedings of the 4th international conference Hardware/software codesign and system synthesis[C]. Seoul,Korea,2006.

[5] Dunkels A,Sterlind F,Tsiftes N,et al. Software-based on-line energy estimation for sensor nodes[A]. In Proceedings of the Fourth Workshop on Embedded Networked Sensors (Emnets IV)[C]. New York:ACM Press,2007. 23 - 27.

[6] Banerjee K S,Agu E. PowerSpy:fine-grained software energy profiling for mobile devices[A]. International Conference on Wireless Networks,Communications and Mobile Computing[C]. Los Alamitos,California:Ieee computer society Press,2005. 1136 - 1141.

[7] Li T,John L K. Operating system power minimization through run-time processor resource adaptation[A]. Microprocessors and Microsystems[C]. Guildford,Surrey,England:IPC Science and Technology Press,2006. 173 - 224.

[8] 粟雅娟,魏少军. 针对分布式系统的快速能耗估计方法及应用[J]. 电子学报. 2005,33(9):1706 - 1709.
Su Yajuan,Wei Shao Jun. A fast energy-aware design technique for distributed real-time systems[J]. Acta Electronica Sinica,2005,33(9):1706 - 1709. (in Chinese)

[9] Brooks D,Tiwari V,Martonosi M. Wattch:A framework for architectural-level power analysis and optimizations[A]. 27th Annual International Symposium on Computer Architecture[C]. 2000.

[10] Tan T K,Raghunathan A,Jha N K. Software architecture transformations:a new approach to low energy embedded software[A]. Design,Automation and Test in Europe Conference and Exhibition (DATE'03)[C]. 2003. 11046.

[11] Chen J,Dubois M,Stenstrom P. Integrating complete-system and user-level performance/power simulators:the SimWattch approach[A]. IEEE International Symposium on Performance Analysis of Systems and Software[C]. Los Alamitos,California:Ieee computer society Press,2003. 1 - 10.

[12] Gurumurthi S,Sivasubramaniam A,Irwin M J,et al. Using

complete machine simulation for software power estimation: The softWatt approach[A]. Proceedings of the International Symposium on High Performance Computer Architecture[C]. Cambridge, MA: Morgan Kaufmann Publishers, 2002. 141 - 150.

[13] Austin T M. A User's and Hacker's Guide to the SimpleScalar Architectural Research Tool set[EB/OL]. http://www.cs.virginia.edu/~skadron/cs654/slides, 2003-08-20.

[14] Mudge T, Austin T. sim-panalyzer2.0-Reference Menual[Z]. University of Michigan, University of Colorado, 2002.

[15] Mudge T, Austin T, Grunwald D. The PowerAnalyzer project[EB/OL]. http://www.eecs.umich.edu/~tnm/power/. 2003 - 09 - 02.

[16] 钟伟军,刘明业. 支持嵌入式操作系统的 ARM 能耗模拟器设计[J]. 计算机应用研究. 2006(4):219 - 221.
Zhong W, Liu M. Design of ARM energy consumption simulator supporting embedded operating system[J]. Application Research of Computers. 2006, (4):219 - 221. (in Chinese)

[17] Chandrakasan A P, Sheng S, Brodersen R W. Low-power CMOS digital design[J]. IEEE Journal of Solid-State Circuits. 1992, 27(4):473 - 484.

[18] Dick R P, Lakshminarayana G, Raghunathan A, et al. Power analysis of embedded operating systems[A]. Proceedings of the 37th Conference on Design Automation[C]. Los Alamitos, California: Ieee computer society Press, 2000. 312 - 315.

[19] Acquaviva A, Benini L, Ricco B. Energy characterization of embedded real-time operating systems[A]. Workshop on Compilers and Operating Systems for Low Power, COLP'01[C]. New York: ACM Press, 2001. 13 - 18.

[20] 陈渝. 源码开放的嵌入式系统软件分析与实践——基于 SkyEye 和 ARM 开发平台[M]. 北京:北京航空航天大学出版社, 2004.
Chen Y. The Analysis and Practice on Open Source Embedded System Software——Based on SkyEye and ARM Developing Platform[M]. Beijing: Beihang University Press, 2004. (in Chinese)

[21] Mudge T, Austin T, Grunwald D. PowerAnalyzer for Pocket Computers. [EB/OL]. http://www.eecs.umich.edu/~tnm/power/, 2002-02-10.

作者简介:

赵　霞　女,1972 年生于甘肃迭部,北京大学信息科学技术学院博士生,北京工商大学副教授. 研究方向为系统软件、低能耗软件设计与软件工程.
E-mail: zhaoxia@os.pku.edu.cn

郭　耀　男,1976 年生于山西襄汾,北京大学信息科学技术学院讲师,2007 年获得美国马萨诸塞大学计算机工程博士学位. 研究方向为系统软件、低功耗系统设计、编译技术、软件工程等.

雷志勇　男,1982 年生于湖南,硕士研究生. 研究方向为系统软件.

陈向群　女,1961 年生于北京,北京大学信息科学技术学院教授. 研究方向为系统软件与软件工程.