整机系统实时功率剖析与建模

杨良怀[1,2] 朱红燕[1]
(浙江工业大学计算机科学与技术学院 杭州 310023)[1]
(浙江省可视媒体智能处理技术研究重点实验室 杭州 310023)[2]

摘 要 功率剖析与建模是功率感知DBMS的基础。将主要硬件(处理器与磁盘)的利用率作为系统功率的指示器,根据资源利用率实时地估计系统功率,构建整机系统的实时功率模型。在多核架构下,整体CPU的活动信息会掩盖单个核的使用情况,因此,从执行核粒度考察执行频率与利用率的综合影响,采用多元线性回归方法拟合执行核的执行频率和利用率、磁盘的利用率和系统功率之间的关系。实验结果显示,所构模型平均相对误差小于12%,且占用系统资源较少,从而不会影响其他应用程序的执行,具有较好的应用价值。

关键词 功率建模,功率剖析,软功率计,多元回归
中图法分类号 TP315 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.09.005

Whole System Realtime Power Profiling and Modeling

YANG Liang-huai[1,2] ZHU Hong-yan[1]
(School of Computer Science and Technology, Zhejiang University of Technology, Hangzhou 310023, China)[1]
(Key Laboratory of Visual Media Intelligent Process Technology of Zhejiang Province, Hangzhou 310023, China)[2]

Abstract Power profiling and modeling are fundament to power aware DBMS. This paper proposed a component-level power modeling method by exploiting the utilizations of the main components (CPU and disk) as the indicators of the system power. To cope with the multi-core architecture, we investigated into the individual cores instead of the monolithic CPU that may cover up some essential details. And we fitted the relationship between the utilizations and power by using the multiple linear regression method. Experimental results show that the average relative error of the power model is less than 12%, and it is lightweight without incurring too much additional cost on other applications.

Keywords Power modeling, Power profiling, Soft-power-meter, Multiple regression

1 引言

计算高性能一直以来都是指引计算机领域硬件和软件发展的方向标。著名的摩尔定律指出:集成电路上可容纳的晶体管数量每18个月增加一倍,从而性能提升一倍。然而随着对信息处理能力的需求以及芯片集成密度的不断增加,计算所需的能量消耗也随之增长,出现了所谓的"功耗摩尔定律":单一节点上的计算能耗每18个月翻一番[1]。

能效计算成为当今计算机领域面临的一个迫切问题。ECOSystem[2]是第一个将能量作为一种资源加以管理的系统。2008年5月,一群有影响力的数据库界研究人员在加州伯克利Claremont度假胜地一起讨论数据库领域的状况并提出一些关于未来的研究方向的建议[3],指出在核心数据库引擎方面的重要研究议题之一是设计具有功率感知的DBMS,它可以限制能耗而不牺牲可扩展性。

功耗感知DBMS的研究以系统功率的剖析与建模为前提。本文从组件层次探讨如何根据硬件的使用情况来预测系统的实时功率。迄今,基于硬件的功率剖析方法[4]因代价昂贵、不灵活等缺陷难以被广泛使用。而有些基于软件的功率剖析策略[2]只能提供脱机统计信息而非实时的系统信息,很难在运行时根据系统的运行状态来应用能量调节策略;Economou[8]通过建模方式实时预测系统功率,但没有针对多核架构以及多执行频率进行考虑。针对现有工作的不足,本文根据实时收集的组件性能统计信息来估计系统的即时功率,基于组件利用率与功率之间呈正相关的事实,采用多元线性回归方法拟合它们之间的关系,构建组件级的功率预测模型——"软功率计"。收集的信息包括CPU利用率及其执行频率、磁盘利用率。多核体系架构成为计算机芯片发展的趋势,从整体处理器的活动信息中难以捕捉到单个核的执行情况,尤其是核异构的架构体系,在设计时就决定了每个核不同的处理能力。考察每个执行核的使用情况,更能有效且准确地反映系统的真实状态,提高模型的准确性。为此,本文从执行核粒度来建模,获取单个核的执行频率和利用率。实验结果显示,"软功率计"的平均相对误差小于12%,但有时因某些进程或系统活动的影响,绝对误差可能会超过20%。同时,"软功率计"消耗较少系统资源,对其他应用程序干扰较

到稿日期:2013-11-19 返修日期:2013-02-16 本文受国家基金项目(61070042),浙江省基金项目(LY13F020026,LY14F020017)资助。
杨良怀(1967-),男,博士,教授,主要研究方向为数据库系统,E-mail:yanglh@zjut.edu.cn;朱红燕(1988-),女,硕士生,主要研究方向为数据库系统。

少，可以应用到功率感知 DBMS 之中。

本文第 2 节介绍功率建模的相关工作；第 3 节和第 4 节分别论述功率模型的构建和实现；第 5 节给出实验结果，并对其进行分析；最后对全文进行总结。

2 相关工作

实时功率估计是功率感知 DBMS 的基础。目前的功率剖析技术可分为硬件测量技术和软件估计技术[10]。仪器测量具有测量精确的优点，但由于价格昂贵等因素而无法大规模地部署在数据中心，这种方法可用于验证或评估模型的有效性。软件预测在准确性上不如仪器测量，但能够提供细粒度的联机功率信息，具有经济灵活的优点。下面重点讨论软件功率预测技术。

Bellosa[5]首次提出使用系统中的硬件性能计数器来预测系统的功率。若能获知每种事件的功率，则可以建立模型计算系统的功率；实验发现 4 种性能事件（整数操作、浮点数操作、二级寻址和内存访问）与系统功率的关系最为紧密，且 4 种操作的操作量与功率近似呈线性关系。事实上，系统中的性能计数器以及能同时监控的性能事件都很有限，但是随着硬件制作工艺的改进，这些问题将得到改善。

相继地，Li[6]也提出了相应功率模型，认为系统的功率由各个例行程序的功率累加而成。其实验发现例行程序的功率与 IPC（每周期指令数）指数近似呈线性关系，从而构建了例行程序的功率模型，并在此基础上得到系统的能量模型、所有活动的例行程序能耗求和即系统的整体能耗。实验结果显示系统功率预测误差在 20%在 50%之间。系统中的例行程序众多，要得到高准确度的模型需考虑所有例行程序，从而使模型过于复杂，这就需要额外的计算资源来维护相关的数据信息，不利于实时的功率剖析。

不同于以往基于性能计数器的功率建模工作，Bircher 等[7]指出与处理器相关的性能事件不仅是处理器功率的有效指示因子，而且与其余设备的功率也存在极大关联，文中重点考察了 5 大设备，包括处理器、内存、芯片组、I/O 子系统和磁盘。通过分析 5 种设备之间的信息传递关系，选择了 6 种性能事件，针对每种设备挑选最为相关的事件作为模型的输入参数。最后通过一组工作负载来验证这组模型的准确性，结果表明 5 种模型的预测误差均控制在 18%以内。

Economou[8]针对服务器环境提出了 Mantis 框架，实现了系统实时功率的剖析与建模。将处理器、磁盘、内存和网络的相关性能指数作为预测系统功率的依据，建立功率与硬件性能指数之间的线性回归模型。Mantis 框架分别对两种服务器结构（Blade，Itanium）构建了功率模型，并用 5 种不同的应用程序来验证模型的准确性，结果显示模型的平均误差范围为 0～15%。

Do 等[9]提出了针对 Linux 系统的进程级能量模型 pTop，为 3 种主要硬件（处理器、磁盘和无线网卡）分别构建能量模型。根据处理器的活动时间、不同执行频率上的运行时间以及执行频率之间的切换次数构建处理器的能量模型；根据单位时间内磁盘/网卡的读写量来估算磁盘/网卡的能耗。而后，根据每个活动进程对各个硬件的占用比例来划分硬件的总能耗，得到进程的能耗模型。pTop 的目的在于计算一段时间内每个活动进程的能耗，硬件的额定功率作为已知条件被写入配置文件。然而，系统的状态是实时变化的，简单地给定处理器的额定功率或将磁盘、网卡的读写功率和读写速率看作常数自然会降低模型的准确性。

在 pTop 的基础上，Shi 等[10]提出了运行于 Windows 平台上的 pTopW 模型。同样地，pTopW 先计算相关组件的能耗，而后根据进程对组件的占用比率来分摊组件的总能耗，各硬件的能耗配额累加即得到进程的能耗模型。pTopW 改进了前一版的不足，在模型中考虑内存的影响，并实现能耗感知框架“EnergyGuard”，旨在识别能耗异常的应用程序。其实验分别验证了组件模型的有效性，虽然准确性不高，但大多数情况下能预测功率的变化趋势。

李等[11]将 CPU 负载、I/O 负载和屏幕亮度以 10%为增量划分为 10 个区间，借助功率分析仪得到 3 种变量在每个区间的基准能耗。而后，实时获取被测机这 3 种参数的变化值，基于基准能耗数据利用分段线性加权拟合方法来估算系统的实时功率。对比实验表明功率估计系统具有较高的准确性，但并没有将执行频率的影响考虑在内；而我们的实验结果表明，系统功率与执行频率呈正相关。

Economou[8]与李[11]的研究与本文的工作最为相关，模型设计思想相似，即利用组件级的性能指数来预测系统的实时功率。本文所使用的输入参数与其不同，包括 CPU 核的利用率及其相应执行频率、磁盘利用率，暂不考虑网络通信功率。多核架构是未来系统发展的必然趋势，多核处理器比之于单核高时钟频率处理器节能[12,13]，在单个芯片上集成了几十、上百、乃至上千个处理单元，之间通过高速片上网络通信。Laudon[12]指出多核处理器是服务器群突破“功障”（power wall）的一个必然选择，多核处理器比相同性能的单核处理器节能，有更好的性能/瓦（即能效）。从核粒度来考察更能反映系统的真实运行状态，本文将考察每个执行核的利用率及其执行频率，并将其作为模型的输入参数。

3 实时功率预测模型

3.1 问题描述

对于台式机，处理器、磁盘和内存是主要的耗能部件。(1)CPU：由 CPU 功率公式 $P_{cpu} \approx cv^2 f$ 可知，处理器的功率与执行频率以及电压的平方成正比。给定执行频率，在采样周期内电压稳定的情况下，处理器的利用率越高，其瞬时功率越高，则系统功率也越高。由此可知，系统功率正比于处理器的执行频率和利用率。处理器各个核的使用情况存在差异，本文显式计算每个核的利用率。(2)磁盘：磁盘功率分别由电机转动所需的功率、寻道功率和接口与控制逻辑所需的功率 3 者组成[14]。即使磁盘没有读写任务，主轴电机和控制逻辑单元仍需持续耗电来维持磁盘的运转，这部分功耗被视为常数。磁盘空闲时磁头不寻道，而一旦有读写操作，且磁头不在所需的磁道上时，磁头便开始寻道，从而产生寻道功耗，这部分功耗随任务量的不同而不同，即磁盘的动态功率不可预知，与磁盘的使用程度有关。一般地，磁盘的利用率越高，其功率也越高。(3)内存：Linux 系统中，/proc/meminfo 文件虽记录了内存的性能信息，但并没有给出内存的读写活动时间，从而难以计算内存的利用率。目前，本文在模型中尚未显式地把内存

部分的功率加入到模型构造之中，而将其作为系统背景功率（非时变）来处理。在将来进一步的研究中，拟从缓存缺失而导致内存访问这一角度来计算内存的活动时间，在模型中反映内存使用率对内存动态功率的贡献。

如何根据执行核的执行频率和利用率、磁盘的利用率作为预测系统功率的参数，建立系统功率预测模型是本文的议题。

3.2 功率模型（理论）

系统功率由静态功率和动态功率两部分构成。系统功率记为 P，静态功率即系统无负载时消耗的功率记为 P_{static}，动态功率则是维持硬件的运转以完成任务所需的额外功率，记为 P_{dyn}，单位皆为瓦特（w）。

$$P=P_{static}+P_{dyn} \tag{1}$$

在一段时间 $t(t>0)$ 内，系统消耗的能量为 E，可视为静态能耗与动态能耗之和，而系统的动态能耗由各个组件的能耗构成。

$$E=E_{static}+E_{dyn} \tag{2}$$

$$E=E_{static}+E_{cpu}+E_{disk} \tag{3}$$

在式(3)中，E_{cpu} 和 E_{disk} 分别表示 cpu 和磁盘的能耗。假设在时间 t 内，cpu 执行时间为 t_1，磁盘工作时间为 t_2，根据能量与功率之间的关系，得到式(4)：

$$P\cdot t=P_{static}\cdot t+P_{cpu}\cdot t_1+P_{disk}\cdot t_2 \tag{4}$$

等式两边同除时间 t，得到式(5)：

$$P=P_{static}+\frac{P_{cpu}\cdot t_1}{t}+\frac{P_{disk}\cdot t_2}{t} \tag{5}$$

根据组件的利用率定义，$u_{cpu}=t_1/t$ 表示 cpu 在时间 t 内的使用率，$u_{disk}=t_2/t$ 表示磁盘的利用率。代入式(5)得到：

$$P=P_{static}+P_{cpu}\cdot u_{cpu}+P_{disk}\cdot u_{disk} \tag{6}$$

据处理器功率与频率成正比，将式(6)中的 P_{cpu} 由 αf 代替，f 为频率。另外，P_{disk} 替换成字符 β，用 C 取代静态功率 P_{static}，得到功率模型式(7)：

$$P=C+\alpha\cdot f\cdot u_{cpu}+\beta\cdot u_{disk} \tag{7}$$

对于 CMP 多核处理器，假设有 n 个核，各个核记为 c_i；在按需运行模式（Ondemand）下，运行时每个核对应特定的执行频率，记为 $f_i(i\in[0,n-1])$ 最终得到如下模型原型：

$$P=C+\sum_{i=0}^{n-1}\alpha_i f_{ci}\cdot u_{ci}+\beta\cdot u_{disk} \tag{8}$$

其中，C 为待测机的静态功率，是常数项；f_{ci} 表示系统中某一执行核的执行频率，单位 GHz；u_{ci} 表示第 i 个执行核的利用率。假设各个核的执行能力等价，则可以认为每个核所对应的系数 α_i 相等，记为 α。式(8)即变为 $P=C+\alpha\sum_{i=0}^{n-1}f_{ci}\cdot u_{ci}+\beta\cdot u_{disk}$，系数 α 与 β 由多元线性回归模型训练得到。

4 系统实现

4.1 实验系统

我们采用 C++语言在 Linux 下实现了“软功率计”框架。采用杭州远方光电信息有限公司的 PF9808B 数字功耗仪，搭建瞬时功率建模的实验平台，如图 1 所示。数字功耗仪每秒 1 次采样被测系统的整机功率，另一台监控机通过串口读取相应采样值。为排除功耗仪采样程序的干扰，采用双机通信的方式进行实验。表 1 是被测系统的硬件参数。

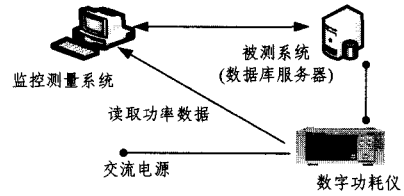


图 1 功耗测试环境示意图

表 1 实验设备规格

组件	型号	功率
操作系统	Ubuntu10.04，内核版本：2.6.33	—
CPU	Intel(R) Core(TM)i5-2400 3.1GHz	95
内存	记忆科技 DDR3 1600MHz 2G×2	3
硬盘	西数 WD5000AAKX/7200RPM	7

4.2 数据采集

Linux 系统中，所有硬件活动信息均被记录在/proc 文件夹下。/proc 是基于内存的一个伪文件系统。实验系统从该文件系统中获取各个处理器和磁盘的活动状态信息，据此计算采样周期内组件的利用率，而每个采样点的系统即时功率由功耗仪提供。

4.2.1 核利用率与频率

/proc/stat 文件记录了 CPU 的活动信息，其记录值是从系统启动开始累计到当前时刻的统计数值，具体参数说明如表 2 所列。

表 2 处理器性能统计数值

参数	参数说明
user	从系统启动开始累计到当前时刻，用户态的 CPU 时间（单位：jiffies），不包含 nice 值为负的进程
nice	从系统启动开始累计到当前时刻，nice 值为负的进程占用的 CPU 时间
system	从系统启动开始累计到当前时刻的内核态运行时间
idle	从系统启动开始累计到当前时刻，除硬盘等待时间以外其他等待时间
iowait	从系统启动开始累计到当前时刻，硬盘 IO 等待时间
irq	从系统启动开始累计到当前时刻，硬中断时间
softirq	从系统启动开始累计到当前时刻，软中断时间

核利用率为处理核执行非系统空闲进程的时间与其总的执行时间的比值。获取两个时刻的统计数值，便可求得采样周期内的每个处理核的利用率。t_1 和 t_2 为两次采样时刻（$t_2>t_1$），x_1 表示 t_1 时刻所获取的统计值，x_2 表示 t_2 时刻获取的统计值，x 取自表 2 中的参数列表。

$$tot\_time_1=user_1+system_1+nice_1+idle_1+iowait_1+irq_1+softirq_1 \tag{9}$$

$$tot\_time_2=user_2+system_2+nice_2+idle_2+iowait_2+irq_2+softirq_2 \tag{10}$$

$$run\_time_1=user_1+system_1+nice_1 \tag{11}$$

$$run\_time_2=user_2+system_2+nice_2 \tag{12}$$

$$U_c=(run\_time_2-run\_time_1)*100/(tot\_time_2-tot\_time_1) \tag{13}$$

其中，U_c 表示处理核的利用率。

由于处理器的功率与执行频率成正比，因此频率也是影响系统功率的重要因素，且两者呈正相关的关系。系统中所有处理核的即时执行频率被分别记录在/sys/devices/system/cpu/cpu$x$/cpufreq/scaling_cur_freq 这一文件中，其中 x

代表处理核的编号。

4.2.2 *磁盘利用率*

/proc/diskstats 文件记录了磁盘相关的活动信息，实验需要 tot_ticks 指标，它表示从系统启动到当前时刻磁盘的活动时间。同样，采样获取两个时刻的统计值，两个时刻的总活动时间之差与采样周期的比值便是所需的磁盘利用率。即 t_1 时刻获取的值为 tot_ticks_1，t_2 时刻获取的值为 tot_ticks_2（$t_2 > t_1$），则磁盘利用率的计算公式如下所示：

$$U_{disk} = (tot\_ticks_2 - tot\_ticks_1)/interval \quad (14)$$

式中，U_{disk} 表示磁盘的利用率，$interval$ 表示采样周期，在实验过程中，采样周期为一秒，即 $interval$ 为 1。

4.2.3 *即时功率*

为了排除功耗仪采样程序的干扰，采用双机通信的方式进行实验。由一台 PC 机专门负责收集从串口获取的功率值，以等同于串口读取速率的频率记录下每个采样时刻对应的功率值。应方便数据处理的需求，将测试端测得的实时功率值通过网络传输至被测试端，执行核的执行频率、利用率、磁盘的利用率以及对应时刻的功率值皆被记录在同一个文件中。

4.3 实验负载

由 3.2 节的分析可知，处理核的个数、执行频率和使用率，以及磁盘的读写活动均能引起系统功率的变化。对于一般的计算机系统，其能耗构成中，处理器能耗占绝大部分。因此，将重点控制处理核的使用率和执行个数，利用文件的读写操作控制磁盘的使用，但不深入考察磁盘的利用率变化产生的影响。此外如无特殊说明，实验程序运行在按需模式下，处理核的执行频率将由操作系统根据运行环境自主调节。

为使功率模型具有普适性，在单核模式（同一时刻，用户态只有一个核处于运行态）下，设计了 8 种负载，每种负载产生特定范围的核利用率，见表 3 所列。负载的设计较为简单，利用循环结构控制执行时间，以产生合适的记录条数（实验中记录条数为 20～40），在循环中，随机数运算将产生执行核的利用率，文件读写操作将产生磁盘的利用率。而后通过控制文件读写的粒度来影响核的利用率，如在循环中，每循环 100 次输出随机数或者输出一段数据，形成负载——load3。前 7 种负载均通过如上介绍的方法获得，最后一种负载则通过冒泡排序算法获得，旨在使执行核达到较高的利用率。同样通过循环来控制记录数，在循环中，对生成的 100 个随机数（0～100）进行排序，并将排序结果输出。

表 3 单线程测试负载

负载	核利用率范围(%)	平均利用率(%)
load1	<20	15
load2	(20,30)	30
load3	(30,40)	40
load4	(40,50)	50
load5	(50,60)	60
load6	(60,70)	70
load7	(70,90)	80
load8	(90,100)	95

虽然基于 CMP 的多核 CPU 中各个核是独立运行的，但核之间仍可能会存在共享资源所带来的干扰，引起利用率发生不同于单核运行时的变化。在多核运行模式（同一时刻，用户态有多个核处于运行态）下，设计的 5 种测试负载如表 4 所列。负载设计思想与单核环境负载设计相似，根据与线程绑定的执行核的编号来控制输出文件的命名，从而避免执行核因读写同一文件造成的干扰。负载所对应的利用率为多个核执行相同负载时计算所得。

表 4 多线程测试负载

负载	核利用率范围(%)	平均利用率(%)
multiLoad1	0	0
multiLoad2	<50	30
multiLoad3	<80	60
multiLoad4	<100	90
multiLoad5	100	100

5 性能评价

本节阐述功率模型构建以及功率模型验证两方面的实验结果。

5.1 功率模型构建

当系统处于空闲态时，测得其平均功率为 38.5W，即静态功率。在单核运行模式下，设计综合负载 1，包含表 3 中所有负载，并在 4 个执行核上分别运行一次（核与线程绑定），得到单核实验数据。在多核运行模式下，为使单个核的利用率范围（0～100%）等可能出现，设计综合负载 2，使得表 4 所列的 5 种负载等概率（即概率为 0.2）执行，利用随机数（0～1）来决定执行何种负载。更改系统配置（此处特指执行核个数），分别得到双核、三核以及四核实验数据。

汇总所得到的 4 组数据，采用多元线性回归方法来拟合 4 个核的执行频率和利用率、磁盘利用率与系统功率之间的关系，得到的模型如式(15)所示：

$$P = 45.126 + 0.0479 \cdot f_{c0} \cdot u_{c0} + 0.0479 \cdot f_{c1} \cdot u_{c1} + 0.0488 \cdot f_{c2} \cdot u_{c2} + 0.0488 \cdot f_{c3} \cdot u_{c3} + 0.0168 \cdot u_{disk} \quad (15)$$

公式中参数的含义见上文所述。得到的模型与 3.2 节讨论的理论模型相符合，4 个核相对应的系数比较接近，印证了基于 CMP 架构的系统中核的执行能力是等价的设计；其次，执行核的系数大于磁盘的对应系数，符合 CPU 的功率比重大于磁盘的功率比重的事实；最后，理想情况下，常数项应等于静态功率（38.5W），但拟合结果为 45.126，与真实值的误差为 17%。因测试负载是人为设计的，且无法预知操作系统的活动，故无法考察所有可能的情况。另外，处理器和磁盘等硬件在处理任务过程中的等待状态下，其功率应高于纯粹的静态功率。因此，模型中的常数项应略高于静态功率。

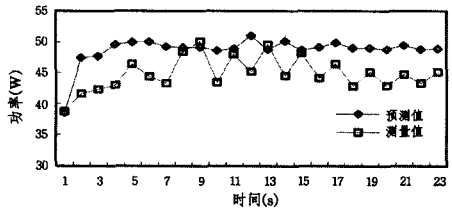
5.2 模型验证

为验证模型的准确性，本文实现 4 种连接算法，分别为块嵌套循环连接算法（BNL）、排序归并连接算法（SMJ）、Grace 哈希连接算法（GHJ）、混合哈希连接算法（HHJ）。测试数据采用 TPC-H 测试基准中的 customer 和 orders 基本表（两表之间只有一个主键/外键连接）。在单核运行模式下，表的规模 *Scale* 取为 3，连接元组数为 4500000 条。分别运行 4 种连接程序，并绘制功率图谱，如图 2 所示。

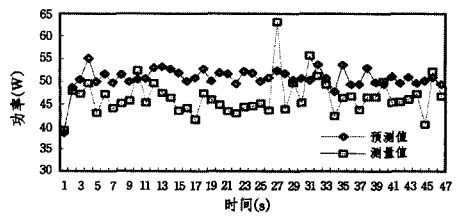
由图 2 可知，4 种连接算法产生的功率变化幅度都比较小，基本维持在 45W 左右。分析运行结果发现，4 种连接算法都充分地利用了磁盘 I/O 资源，而 CPU 的利用率较低，平均仅为 15%，所以系统的功率普遍较低。经计算，在 4 种算

法的测试下，模型的平均相对误差分别为 9.11%、11.12%、9.76%和 9.18%。

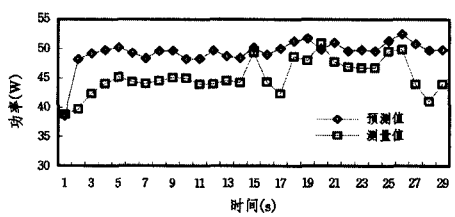
为了检验所建功率模型能否适用于多线程环境，将 4 个连接算法分别按多进程方式运行，并绘制如图 3 所示的功率图谱。在多进程运行模式下，表的生成规模 $S=1$，连接元组数为 1500000 条。



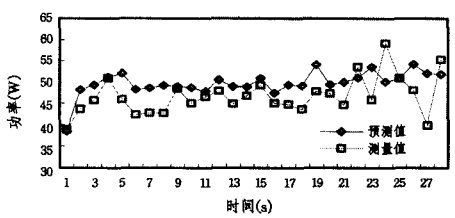
(a) BNL



(b) SMJ



(c) GHJ



(d) HHJ

图 2　功率模型预测

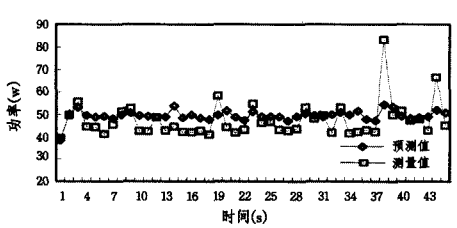


图 3　多核环境下功率模型预测

由图可知，多核多进程运行时，系统功率的变化与单进程环境下的变化总体相差不大、波动小，在 40W～60W 之间变化。原因是处理器的处理速率远高于磁盘的读写速率，造成处理器等待，处于低利用率状态。模型的预测平均相对误差为 11.21%。

5.3　CPU 频率参数对模型准确性的影响

本节考察 CPU 频率因素对模型的影响。为此，构建输入参数仅有 CPU 整体的利用率和磁盘利用率的功率模型，称为 PM2，它不考虑处理器的执行频率。为方便叙述，将前面的模型记为 PM1。采用与前面完全一致的实验方法来构建新的功率模型，得到 PM2 模型，如式(16)所示：

$$PM2=46.8936+0.5313\cdot U_{cpu}+0.0122\cdot U_{disk} \quad (16)$$

式中，PM2、U_{cpu} 和 U_{disk} 分别表示系统功率、CPU 利用率和磁盘的利用率。

实验负载包括上文所提的 4 种连接算法、Linux 系统的开源杀毒软件 ClamAV[15] 以及冒泡排序算法 BUBBLE。6 种应用程序的核利用率、CPU 利用率和磁盘的利用率见表 5，大体上覆盖了从低到高的范围。应用程序的运行可认为是单线程操作，测试机共有 4 个核，当单核满负载(利用率为 100%)时，CPU 整体的利用率为 25%。

表 5　应用程序的利用率

负载	单核利用率(%)	CPU 利用率(%)	磁盘利用率(%)
BNL	15	7	99
SMJ	30	10	98
GHJ	23	8.5	95
HHJ	24	8.4	93
ClamAV	77	20	43
BUBBLE	94	24	40

分别运行这组测试用例，对比两种模型的预测误差，得到图 4 所示结果。由图 4 可知，在 6 种测试环境下，PM1 的预测相对误差为 10%左右，不超过 12%，而 PM2 的相对误差则在 15%左右。除了 BUBBLE 测试程序，PM1 的预测误差稍高于 PM2 的误差，其他程序的测试结果显示 PM1 的预测效果均优于 PM2 的预测效果。这表明设计的“软功率计”能够较好地预测和反映真实环境的功率变化，也可发现在模型中考虑执行频率之后，大多数环境下，能够降低预测误差，提高准确性。

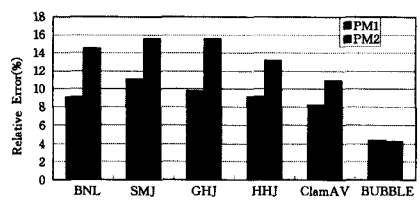


图 4　功率模型的平均相对误差

结束语　本文提出了一种通用的根据 CPU 各个核的利用率及其相应执行频率、磁盘利用率来预测系统功率的方法，构建了功率预测模型，实现了“软功率计”。实验结果表明，该模型的平均相对误差小于 12%，但有时因不可预知因素(如底层操作系统进程等任务)的影响，个别点绝对误差可能会超过 20%。软功率计仅仅在输出日志信息时占用少量 IO 资源，额外运行负荷小，不会影响其他应用程序的执行性能。

在进一步的工作中，将考察内存利用率对提高模型准确性的作用；同时，根据系统中各个活动进程对各硬件的占用比例分摊系统功率，构建进程级的功率模型。

参 考 文 献

[1] Wu C F. Making a case for efficient supercomputing [J]. Queue, 2003, 1(7): 54-64

[2] Heng Z, Ellis C S, Lebeck A R, et al. ECOSystem: managing energy as a first class operating system resource[J]. SIGARCH Comput. Archi. News, 2002, 30(5): 123-132

[3] Agrawal R, Ailamaki A, Bernstein P A, et al. The Claremont re-

port on database research[J]. SIGMOD Record,2008,37(3):9-19

[4] Flinn J, Satyanarayanan M. Energy-aware adaptation for mobile applications[C]// Proceedings of the ACM symposium on Operating systems principles. 1999:48-63

[5] Bellosa F. The benefits of event-driven energy accounting in power-sensitive systems[C]// Proceedings of the Workshop on ACM SIGOPS. 2000:37-42

[6] Li T, John L K. Run-time modeling and estimation of operating system power consumption[C]// Proceedings of the ACM International Conference on Measurement and Modeling of Computer Systems. 2003:160-171

[7] Bircher W L, John LK. Complete System Power Estimation: A Trickle-Down Approach Based on Performance Events[C]// Performance Analysis of Systems & Software. 2007:158-168

[8] Economou D, Rivoire S, Kozyrakis C. Full-system power analysis and modeling for server environments[C]// Workshop on Modeling, Benchmarking and Simulation. 2006

[9] Do T, Rawshdeh S, Shi W S. pTop: A process-level power profiling tool[C]// Proceedings of the Workshop on Power Aware Computing and Systems. 2009

[10] Chen H, Li Y, Shi W S. Fine-grained power management using process-level profiling[J]. Sustainable Computing: Informatics and Systems, 2012, 2(1):33-42

[11] 李晓,李战怀,刘文洁,等. 计算机系统实时估算方法:中国, CN102221874 A [P]. http://www. google. com/patents/CN102221874A? cl=zh, 2011-10-19

[12] Laudon J. Performance/Watt: The New Server Focus[J]. SIGARCH Computer Architecture News, 2005, 33(4):5-13

[13] Geer D. Chip makers turn to multicore processors[J]. Computer & Processing, 2005, 38(5):11-13

[14] Bostoen T, Mullender S, Berbers Y. Power-Reduction Techniques for Data-Center Storage Systems[J]. ACM Comput. Surv., 2013, 45(3):1-38

[15] Clam AntiVirus[OL]. http://www. clamav. net/

(上接第 27 页)

结束语 概念模型验证是建模与仿真校核、验证与确认中最重要、最困难的问题之一,对于提前修正概念模型的错误,降低仿真开发的风险和成本,提高仿真系统的可信性具有重要意义。然而现有概念模型验证方法,要么过于依靠专家的知识经验,主观性强,要么验证过程复杂繁琐,成本高。本文采用本体论和语义网技术,提出了一种基于本体与规则推理的装备保障仿真概念模型语义验证方法。该方法主要是通过模型的描述形式转换、模型验证规则构建、模型的格式转换、语义推理等环节实现。其实质是将基于 UML 描述的概念模型转换为本体推理机可读的 OWL 描述的语义模型,并将专家的验证知识、验证经验以及领域知识提炼为验证规则,在计算机上实现概念模型语义验证的自动化。最后通过实例验证了本文所提方法的有效性,它可以减少专家验证的主观性和不确定性,降低形式化验证方法的复杂性,提高验证效率。

参 考 文 献

[1] Robert G S. Verification and Validation of Simullation Models [C]// Proceedings of the 2010 Winter Simulation Conference. 2010:173-176

[2] 王勇,杨明. 复杂仿真系统概念模型评估技术研究[J]. 系统仿真学报,2008,20(24):6808-6810

[3] 樊浩,黄树彩. 基于 Petri 网的概念模型验证方法研究[J]. 计算机应用研究,2010,27(3):999-1002

[4] Ynag Hui-zhen, Hao Li-li. Colored Petri Nets-based Formal Modeling and Validation for Federation Conceptual Model[J]. Journal of System Simulation, 2012, 24(7):1361-1365

[5] 岳增坤,陈炜. 基于 xUML 的 C4ISR 系统可执行对象模型设计[J]. 系统仿真学报,2009,21(8):2190-2194

[6] 何晓晔,徐培德,沙基昌. 任务空间概念模型轻量级形式化校核方法初探[J]. 系统仿真学报,2006,18(5):1108-1109

[7] 骆翔宇,苏开乐,顾明. 一种求解认知难题的模型检查方法[J]. 计算机学报,2010,33(3):406-414

[8] 夏薇,姚益平,慕晓冬. 基于 TLA 的事件图模型形式化验证方法[J]. 计算机应用研究,2011,28(11):406-414

[9] 杨斌,齐玉东,孟凡磊,等. 本体在概念建模中的应用研究[J]. 计算机技术与发展,2011,5(21):246-249

[10] 齐玉东,杨斌,郭天杰. 军事概念模型中的部分-整体关系的表达与推理[J]. 系统仿真学报,2010,22(11):2537-2541

[11] 唐见兵. 作战仿真系统可信性研究[D]. 长沙:国防科学技术大学,2009

[12] 王智学,董庆超. 基于 UML 模型的 C4ISR 系统能力需求分析与验证[J]. 系统工程与电子技术,2009,31(9):1561-1564

[13] 刘忠,钱猛. 基于语义推理的作战计划验证方法[J]. 系统工程与电子技术,2010,5(32)

[14] 钱猛,刘忠. 使用本体和 SWRL 验证作战计划的方法[J]. 计算机工程与应用,2009,45(8):208-212

[15] 刘振中,刘勇. 基于 UML 类图的 OWL 本体映射方法[J]. 计算机应用研究,2009,35(13):40-43

[16] 王翀,何克清,刘进. 基于 OWL 元模型的本体建模研究[J]. 武汉大学学报,2004,50(5):581-585

[17] 郁书好,苏守宝,刘仁金. UML 和 OWL 在本体建模中的比较研究[J]. 计算机技术与发展,2007,17(1):155-157

[18] James C. XSL Transformations (XSLT) Version 2.0 [EB/OL]. http://www. w3. org /TR/xslt2. 0, 2009-10-10

[19] W3C. SWRL: A Semantic Web Rule Language Combining OWL and RuleML [EB/OL]. http://www. daml. org/rul es/proposal, 2012-1-12

[20] Sandia National Laboratories. Jess, the Rule Engine for the Java Platform [EB/OL]. http://Herzberg. ca. sandia. gov/jess, 2012-1-12