DOI：10.7652/xjtuxb201812011

采用机器学习算法的软件能耗感知模型及其应用

傅翠娇[1]，钱德沛[1,2]，栾钟治[1]
（1.北京航空航天大学计算机学院，100191，北京；2.中山大学数据科学与计算机学院，510275，广州）

摘要：针对软件开发人员不了解代码的能耗，或者即使能测得软件运行时的能耗但由于缺乏对影响能耗的各因素了解的问题，在对 4 种机器学习算法实验的基础上，选用岭回归算法建立了以软件性能事件为特征的软件能耗模型。以收集的软件能耗测量值和软件性能事件数据作为训练样本，利用岭回归机器学习算法建立平均能耗模型，用平均能耗模型来预测其他软件的平均能耗；并在此基础上，利用软件运行的实时特征提出了实时的能耗模型。实验结果表明：以测量的能耗为基准，采用所提出的平均能耗模型对软件能耗进行预测的误差率在 9%以内；与平均能耗模型相比，实时能耗模型的预测误差率更低。采用本文所提出的两种能耗模型，软件设计人员不需要对自己的软件进行测量和训练就可以较准确地估算出软件的能耗并能解释能耗产生的原因，最后根据预测结果对软件的能效进行优化。

关键词：机器学习；软件能耗模型；性能事件；能效
中图分类号：TP303 **文献标志码**：A **文章编号**：0253-987X(2018)12-0070-07

An Energy Consumption Model for Softwares Using Machine Learning and Its Applications

FU Cuijiao[1], QIAN Depei[1,2], LUAN Zhongzhi[1]
(1. School of Computer Science and Engineering, Beihang University, Beijing 100191, China;
2. School of Data and Computer Science, Sun Yat-Sen University, Guangzhou 510275, China)

Abstract: An energy consumption model for softwares using machine learning is proposed to address the problem that software developers either are usually not aware of the specific energy consumption of their codes, or may measure the energy consumption of the application softwares using some tools but do not understand the factors which affect energy consumption and the relationship among them. The Ridge machine learning algorithm is adopted based on testings of 4 different machine learning algorithms to establish the model for estimating the software energy consumption characterized by software performance events. First, the collected measurement values of energy consumptions and the data of the software performance events of a set of softwares are used as the training samples, and the machine learning algorithm is applied to build up the energy consumption model. Then the average energy consumption of any other software is estimated using this model. On this basis, a real-time program energy consumption model for the running programs is established. Experimental results show that the energy consumption prediction error given by the proposed average energy consumption model is less than 9%. A

收稿日期：2018-04-23。 作者简介：傅翠娇(1978—)，女，博士生；钱德沛(通信作者)，男，教授，博士生导师。 基金项目：自然科学基金资助项目(2017YFB0202202)。
网络出版时间：2018-10-25 网络出版地址：http://kns.cnki.net/kcms/detail/61.1069.T.20181023.1652.008.html

comparison with the average energy consumption model shows that the real-time program energy consumption model further reduces the prediction error. When the software designers use the proposed model to get the energy consumption of their codes, they are not required to measure or train their own software. Not only can the energy consumption of the software be accurately estimated, but also the causes of energy consumption can be explained, and the energy efficiency of the software can be optimized according to the prediction result.

Keywords: machine learning; software energy consumption model; performance event; energy efficiency

随着通信网络、个人计算机以及数据中心规模的快速增长，它们消耗的能量也越来越多，2012 年个人和商业计算机的能耗已超过 470 TW·h[1]。在计算机硬件方面已经采取了一些节能技术，主要有根据系统中的负载来调节操作状态的动态电源管理技术[2]和根据计算能力的不同需要来动态调节电压和频率的技术[3]。然而，在软件方面，软件开发者一般不确定一段代码的具体能耗和如何减少能耗。虽然借助于某些测量工具可以直接读出软件的能耗，但是它不能解释能耗变化的原因以及定位程序能耗较高的代码段。本文提出了一种基于机器学习算法的能耗模型，软件开发者可以通过该模型来评估代码的能耗，而不需要用测量的能耗数据训练自己的模型。该能耗模型是基于性能事件工具测量的性能数据建立的，这些性能事件本身可以反映出程序运行时的行为，所以可以解释能耗变化的原因。实验数据表明，以测量的能耗为基准，该模型的平均能耗预测误差小于 9%。程序员可以利用该模型对自己所编写的程序的能耗进行预测，而不需要专门对程序再进行能耗测量和训练；也可以利用该模型准确地估计一些重复使用率高的代码能耗，以便为优化这些代码提供依据。

1　软件能耗模型的研究现状

现有的关于软件能耗的研究工作主要包括：CPU 级、功能部件级、进程级、分布式系统的应用程序以及基于系统调用的能耗模型等。

CPU 的能耗是计算机系统能耗中的重要部分。CPU 级能耗模型的研究主要集中在周期级和指令级两个抽象的级别。周期级体系结构模拟器是在每一个执行周期通过激活微体系结构级单元或者块来估测模拟处理器的能耗[4]，David 等对处理器各部分分别进行模拟然后计算出最后的能耗值[5]。指令级的能耗模型是用来估计程序中所有指令执行的总能耗，Seo 等提出了基于 Java 编程语言的能耗模型[6]。

在通过收集和使用实时部件能耗信息建立功能部件级能耗模型方面，Chen 等提出了计算机系统主要功能部件的能耗模型[7]；Shye 等发现整个计算机能耗的主要部分是屏幕和 CPU 功耗[8]；Gurumurthi 等发现最耗能的部件是硬盘[9]；Zhang 等研究了部件利用率的能耗模型[10]。

进程级能耗模型是一种更加细粒度的方法，它根据实时的能耗信息作出节省能耗的关键决策。Chen 等把进程的能耗定义为进程对所有功能部件使用的能耗之和[7]。

Feng 等在文献[11]中提出了分布式系统应用程序的能耗测量和分析框架；Seo 等则提出了基于 Java 编程语言的分布式系统能耗模型[6]。

Pathak 等提出了基于系统调用的有限状态机能耗模型[12]，有限状态机的每个状态代表一个特定部件的功率使用模式。Aggarwal 等提出了基于系统调用计数的模型[13]，该模型不适用于新的应用软件，它只能预测能耗的变化，并不能给出具体的能耗情况。

2　软件能耗模型的建模准备

本文采用机器学习算法建立能耗模型，该模型以软件的性能事件为特征。在建立软件能耗模型的过程中，首先采集用作训练样本的性能和能耗数据，然后根据相应的算法进行特征选择，通过有监督的机器学习算法建立与性能相关的能耗模型，再根据模型预测测试程序的能耗，最后根据预测的能耗结果，通过修改程序代码来达到优化程序能效的目的。

2.1　性能事件

本文以性能事件作为模型的特征是因为性能事件与程序的行为和特征有关。Perf 工具[14]是 Linux 下的性能分析工具。它支持多种性能事件，主要包含 7 个硬件事件、8 个软件事件和 26 个硬件高速缓存事件。软件事件实际上是与硬件无关的内核计数

器。硬件事件和高速缓存事件依赖于 CPU 架构[4]。我们应用 Perf 工具采集了 41 个性能事件，这些性能事件是 cycles、instructions 等。

2.2 训练样本数据的收集

为了建立一个健壮的能耗模型揭示能耗与性能的关系，需要收集大量的训练数据(即软件性能数据与软件能耗数据)。为了方便数据的采集，本研究团队专门研制了 Marcher 服务器，该服务器提供了能耗采集的基本功能。我们在 Marcher 服务器上运行大量的样本软件，采集软件运行的性能和能耗相关的数据。软件的性能数据是利用 Perf 工具采集的程序运行过程中的数据。能耗数据是在 Marcher 服务器读取的运行程序的功耗数据。用作训练的程序主要是 C、C++、Python 以及 Java 程序，这些程序中有 100 多个程序是本研究团队自己编写的程序，有 300 多个程序来自编程任务网站[15]，此外还包括 OMP2012[16]、SPEC2006[17]、FFT 和 Nbody 等程序[18]。用做测试的程序是 Bhomp[18] 和 NPB3.0[19]。为了使测试集中于计算等基本的操作，去掉了程序中所有与输入输出相关的代码部分。

为了获取实验所需的训练样本数据，专门设计了脚本程序。在服务器上运行这些脚本程序，自动实时记录能耗数据。从可测功率的高性能计算机(Marcher 服务器)上实时读取程序运行时的功率数据并求其平均值，为了避免由于其他因素造成的干扰，每个程序运行 3 次，取 3 次功率数据的平均值。本文还设计了一些脚本用于记录从性能工具获取的性能事件数据，分别记录了多个点的性能数据。对于性能数据没有采取平均值而是采用了中位数。采用中位数是因为它不受极端数据的影响，可以避免数据相差较大。

2.3 影响程序能耗的运行特征选择

为了减少特征集中的特征数，需要进行特征选择。选择出的特征可以更好地解释模型并提高模型预测的准确性。

由于不同性能事件数据的取值是不同的，有的数值差距较大，所以将性能数据标准化到 0～1 的范围内。标准化的数据使机器学习算法运行的更快更准确[12]。

本文选用了 Perman 相关性分析、主成分分析法以及随机森林算法进行特征选择。根据机器学习算法和相关性分析算法选择了排名前 22 的特征，它们分别是：cycles、instructions、cache-references、cache-misses、bus-cycles、branches、branch-misses、ref-cycles、stalled-cycles-backend、stalled-cycles-frontend、L1-dcache-loads、L1-dcache-load-misses、dTLB-loads、LLC-loads、LLC-load-misses、LLC-stores、cpu-clock、cs、migrations、L1-dcache-prefetch-misses、branch-load-misses 和 branch-loads。其中随机森林算法在选择这 22 个特征时，预测的得分为 96.06，这些特征主要与系统时钟、访问高速缓存以及执行指令的顺序有关。

不同类型的软件可选择出不同的特征。首先选择采集自程序 Nbody 的数据来训练模型，只选择 LLC-stores 和 migrations 特征时，预测的得分可达 86.1，模型的系数是[16.861，71.254]，常数是 72.814。当选择 bus-cycles、branch-misses、ref-cycles、LLC-stores、LLC-prefetches 和 migrations 这 6 个特征时，预测得分高达 94.77，模型的系数是[−5.554，3.005，−5.545，17.113，6.026，72.042]，常数是 72.490。Nbody 程序是典型的并行程序。对于并行程序，在开始运行时需要把任务分派到各个处理器上，Migrations 参数反映了程序的并行执行程度。对于计算密集型的程序来说，由于存在较多的写回操作，因此 LLC-stores 反映了最后一级高速缓存写回的次数。

2.4 用于能耗建模的机器学习算法

本文分别采用了线性回归、随机森林、岭回归和套索回归 4 种不同的机器学习算法建立预测能耗的模型，之所以选择这些算法是因为它们使用简单、预测准确。

线性回归算法是用于预测分析的基本算法[20]。本文中因变量是 CPU 的能耗，自变量是程序的性能特征。随机森林算法是一种集成的回归学习算法，随机森林算法在进行样本训练时，分别训练一系列的决策树，通过合并各个树的预测结果来减少预测的方差，提高其在测试集上的性能表现。岭回归算法是简单线性回归的扩展。通过增加一个正则表达式来限制系数的大小以避免过度拟合，它本质上是一种修正的最小二乘法，只是增加了一个惩罚项。对一组样本数据集$[x_1^i, x_2^i, \cdots, x_n^i, y^i]$($i=1, \cdots, m$)，岭回归算法找到一个系数向量$\boldsymbol{\theta}=[\theta_1, \theta_2, \cdots, \theta_n]$，使误差平方和最小，得到最佳的预测功耗 $p=\boldsymbol{\theta x}$，可表示如下

$$\boldsymbol{\theta}=\operatorname{argmin}\left[\sum_{i=1}^{m}\left(y^i-\sum_{j=1}^{n}\theta_j x_j^i\right)^2+\lambda\sum_{j=1}^{n}\theta_j^2\right] \quad (1)$$

式中：m 是用于训练的程序数；n 是从性能特征中选择的特征数；$\boldsymbol{x}$ 为性能特征向量，$\boldsymbol{x}=[x_1, x_2, \cdots,$

$x_n]^{\mathrm{T}}$；y^i 是测量的功耗；惩罚项就是式(1)中包含 θ_j^2 的部分[21]。岭回归算法的弱点是它不减少具有零系数的特征。套索回归算法可以去掉一组高度相关的特征。岭回归和套索回归算法的区别是惩罚项不同。在套索回归算法中惩罚项就是把式(1)的 θ_j^2 部分替换成 $|\theta_j|$[21]。

2.5　预测误差率

本文将预测误差率定义为测量的功耗与预测的功耗差的绝对值除以测量的功耗

$$\varepsilon = |y - p| / y \tag{2}$$

式中：ε 为软件预测误差率；y 为测量的功耗；p 为预测的功耗。

2.6　软件分类方法

根据树型随机分类算法选择性能特征 cache_miss 对测试的程序进行分类。运行 Weka 软件的分类功能，根据实验数据得到 cache_miss 的阈值。在实验中，根据程序性能的 cache_miss 数值将软件分为两类，一类是 cache_miss 数值大于或等于 2×10^7 的软件，这类软件具有计算密集型和访存密集型的特征；一类是 cache_miss 数值小于 2×10^7 的软件。对于每个内存访问请求，处理器查找主缓存以检索数据，若找不到数据，则将其视为高速缓存缺失。cache_miss 是软件的一个重要性能指标，非常适合对软件进行分类，然后根据类别分别建立能耗模型。

3　软件能耗模型的建立与实验分析

本节首先介绍了实验所用服务器的配置和测试程序，然后验证了实验的预测准确率，并以预测准确率较高的岭回归算法建立了能耗模型，最后根据模型分析了能耗产生的原因。

3.1　Marcher 服务器系统配置

在美国国家科学基金会资助的可测功率的高性能计算机(代号 Marcher)上进行实验。每一个服务器包含两个英特尔 Xeon E5-2600 处理器，每个处理器有 16 个内核、一个 32 GB 的 DRAM、一个 K20 GPU 和一个英特尔 Xeon Phi 协处理器，以及硬盘和 SSD 的混合存储。在服务器上已开发了一个易于使用的 API，它可以从多个来源来读取功率数值，包括英特尔 RAPL 接口[21]、NVIDIA 管理库(NVML)接口[22]、英特尔 micaccess API[23]和电力数据采集卡(PODAC)。通过这个 API，可以很容易地在运行时收集所有主要部件(例如 CPU、内存、硬盘、GPU 和 Xeon Phi)的细粒度的功率数据。由于测试样本程序不使用 GPU 和 Xeon Phi，而磁盘的功率在大多数时候是相同的，所以在实验中只计算 CPU 功率数据。关于服务器系统更多的细节可参见文献[24]。

3.2　测试程序说明

在测试实验中使用的基准测试程序主要包括 NPB3.0 和 Bhomp，共有 27 个独立的程序运行形式。其中，程序 BT 的 S 类型的可执行程序的形式是“BT.S”，Bhomp 中的程序是不同的输入数据集下的 Bhomp 程序的独立运行结果。例如 Nhome(5)是指 Nhome(200 000，10，5)，其中 200 000、10、5 是程序运行时的输入参数，其他程序类似。这些程序主要是 CPU 计算密集型和内存访问密集型程序。

3.3　预测结果的验证

预测的误差率越小说明预测准确率越高。表 1 给出了不同机器学习算法下能耗测量数据和预测数据之间的差异。由实验结果得到，这些程序能耗预测的平均预测误差率为 6.80%。岭回归算法的平均预测误差率最小为 5.70%。随机森林算法、套索回归和线性回归算法的平均预测误差率分别为 6.5%、8.18%和 6.81%。可以看出，所得到的预测数据基本上是准确的。由于岭回归算法的误差最小，所以在建立能耗模型时选取岭回归算法。

3.4　软件能耗模型

本文首先建立了软件平均能耗模型，然后在此基础上建立了实时能耗模型，并对二者的预测误差率进行了比较。

3.4.1　平均能耗模型　通过对 4 种机器学习算法的实验比较，选择岭回归算法作为模型所用的算法，建立了预测平均功耗的模型 $p=\boldsymbol{c}\boldsymbol{x}+b$，其中 $\boldsymbol{c}$ 为模型的系数向量，$\boldsymbol{c}=[\boldsymbol{c}_1,\boldsymbol{c}_2,\cdots,\boldsymbol{c}_n]$；$b$ 为常数。程序能耗的公式是 $E=pt$，p 为 CPU 的功耗，t 为程序的运行时间。代入 p 得到能耗公式

$$E = (\boldsymbol{c}\boldsymbol{x} + b)t \tag{3}$$

当性能参数 cache_miss 值大于等于 2×10^7 时，能耗模型 $E_1=(\boldsymbol{c}_1\boldsymbol{x}+b_1)t$。当 cache_miss 值小于 2×10^7 时，能耗模型 $E_2=(\boldsymbol{c}_2\boldsymbol{x}+b_2)t$。根据岭回归机器学习算法可以得到模型的系数，其中

$b_1=109.74$

$b_2=63.2$

$\boldsymbol{c}_1=[-3.97,-2.64,-8.13,-3.41,3.76,-6.7,-0.97,3.77,-11.14,2.46,-8.7,0.74,-8.7,-9.64,3.12,-11.66,3.85,9.64,3.76,3.54,-0.94,-6.31]$

表 1 不同机器学习算法软件能耗的预测误差率

程序运行形式	预测误差率/%			
	随机森林算法	套索回归算法	线性回归算法	岭回归算法
BT. A	0.37	7.04	1.74	1.14
BT. B	8.67	1.63	4.87	1.30
BT. C	12.12	0.16	16.84	29.57
CG. A	12.43	1.36	7.56	7.42
CG. B	13.25	3.82	3.22	1.18
CG. C	0.00	1.97	9.28	10.05
FT. A	3.72	5.37	4.14	3.85
IS. A	5.93	9.44	0.19	0.42
IS. B	8.44	5.04	4.63	4.36
IS. C	8.64	8.06	1.99	1.57
LU. A	11.03	6.70	2.88	1.74
LU. B	13.52	5.07	10.08	9.66
LU. C	6.93	13.88	22.72	12.09
MG. A	2.44	6.69	2.49	1.68
MG. B	1.29	4.60	5.39	5.92
SP. C	23.13	2.38	9.82	0.44
Bhomp(5)	4.70	13.46	1.22	1.21
Bhomp(6)	10.32	4.19	6.58	6.70
Bhomp(7)	7.30	1.43	6.74	1.98
Bhomp(8)	4.25	6.06	1.66	0.00
Bhomp(9)	1.77	9.55	9.32	0.08
Bhomp(10)	0.05	11.56	4.18	0.03
Bhomp(12)	3.56	17.21	6.35	0.08
Bhomp(13)	0.98	14.71	8.85	0.08
Bhomp(14)	6.85	18.47	16.58	0.12
Bhomp(15)	7.77	20.67	8.70	0.07
Bhomp(16)	5.57	20.46	5.79	0.06
平均误差率/%	6.50	8.18	6.81	5.70

$\boldsymbol{c}_2=[2.45,-1.93,-1.64,1.51,2.44,-3.73,-0.78,2.44,0.1,8.2,1.3,-3.04,1.3,-2.81,1.55,1.69,2.45,2.34,21.51,-0.4,-0.8,-3.71]$

3.4.2 实时能耗模型 建立上述能耗模型时，选取性能数据中位数作为训练数据，虽然中位数可以规避数值差异较大带来的问题，但是它不能实时地反映系统的实际运行功率，所以需要建立实时的能耗模型。根据实时的性能信息实时预测出能耗的值。当 cache_miss 的值大于等于 2×10^7 时程序的 j 时刻(在 0.02 s 的一段时间内)的实时功耗模型为 $p_1=\boldsymbol{c}_1\boldsymbol{x}+b_1$；当 cache_miss 的值小于 2×10^7 时程序的功耗模型为 $p_2=\boldsymbol{c}_2\boldsymbol{x}+b_2$，对应 j 时刻的实时能耗模型分别为 E_{1j} 和 E_{2j}，即

$$E_{1j}=(\boldsymbol{c}_1\boldsymbol{x}_j+b_1)t/N,\quad E_{2j}=(\boldsymbol{c}_2\boldsymbol{x}_j+b_2)t/N \tag{4}$$

Perf 工具的采样频率为 N，程序运行的时间为 T，那么在时间 T 内，Perf 工具共采样了 TN 次。两类程序的平均能耗分别为 E_1 和 E_2，即

$$\left.\begin{aligned}E_1&=P_1T=\frac{\sum_{j=0}^{TN}P_{1j}}{NT}T=\frac{\sum_{j=0}^{TN}P_{1j}}{N}\\E_2&=P_2T=\frac{\sum_{j=0}^{TN}P_{2j}}{NT}T=\frac{\sum_{j=0}^{TN}P_{2j}}{N}\end{aligned}\right\} \tag{5}$$

表 2 给出了以 Bhomp 基准测试程序为测试程序的预测误差率结果，从表 2 可以看出，实时能耗模型预测误差率平均值仅为 0.79%，比平均能耗模型预测误差率降低了 17%。

表 2 不同应用程序的平均能耗模型和实时能耗模型预测误差率比较

程序运行形式	平均能耗模型预测误差率/%	实时能耗模型预测误差率/%
Nhome(5)	1.21	0.26
Nhome(6)	6.70	4.02
Nhome(7)	1.98	1.40
Nhome(8)	0.00	0.59
Nhome(9)	0.08	0.55
Nhome(10)	0.03	0.28
Nhome(12)	0.08	0.22
Nhome(13)	0.08	0.29
Nhome(14)	0.12	0.08
Nhome(15)	0.07	0.21
Nhome(16)	0.06	0.83
平均误差率/%	0.95	0.79

3.5 程序能耗的原因分析

从 3.4.1 节的能耗模型可以看出，和功耗相关的主要性能特征可以分为以下几类。

(1)与 CPU 的执行周期有关的参数为 cycles、bus_cycles、ref-cycles 以及 cpu-clock，这些都是和时间相关的参数。这些参数的值越大，程序的执行时间越长，CPU 的能耗也越高。

(2)与分支预测相关的参数为 branches、branch-misses、branch-loads 和 branch-load-misses。如果

分支预测正确，指令的执行沿最短时间的路线前进，程序的执行时间较短。但是如果这些数值变大，说明程序在分支预测上花费了更多的时间，会延长整个程序的执行时间。

(3)与高速缓存访问相关的参数为 cache-references、cache-misses、L1-dcache-loads、L1-dcache-load-misses、dTLB-loads、LLC-loads、LLC-load-misses、LLC-stores、L1-dcache-prefetch-misses。其中 cache-references 是高速缓存的访问次数，cache-misses 是高速缓存访问不命中的次数。一般假定所有的存储器停顿都是由高速缓存缺失引起的，把命中时钟周期数包含在 CPU 执行时钟周期数中。最后一级高速缓存的读写、TLB 的读取、L1 数据高速缓存的读取、读取失败的次数以及预测失败的次数都对高速缓存的缺失率起重要作用。CPU 频率越高，CPU 满载时的功率也越高，每次缺失会需要较多的时钟周期数，从而导致存储器停顿时间延长、性能降低、能耗增加。

(4)与程序的切换有关的参数为 cs、migrations。cs 是上下文切换的次数，migrations 是线程从一个 CPU 的核转移到另一个核上执行的次数。在计算量较大的情况下，CPU 会把一些计算任务转移到空闲的核上去执行。

(5)与程序执行指令条数相关的参数为 instructions。指令数是独立于硬件的，一般用来评价性能，在时钟周期不变的情况下，执行的指令越多，程序的执行时间越长。

通过性能参数对 CPU 的功耗建模，可以揭示功耗产生的原因。当通过调节 CPU 的功率来降低能耗时，从模型上可以看出调节功耗影响了程序的性能，进而延长了程序的执行时间，从而增加了程序能耗，所以不能为了降低能耗而盲目地降低或者增加 CPU 的功率。

4　基于能耗预测的能效优化

能效优化可以通过对采用不同方法编写的相同功能的程序对比实验来说明。实验中实际测量与预测的 CPU 功耗基本一致。实验中所有程序都运行了 3 次，实验数据取平均值。在第 1 组递归和非递归程序对比实验中，递归程序运行了 6.798 s，CPU 的功耗实际测量值为 378.994 W，预测值为 382.549 W；非递归程序运行了 0.201 s，CPU 的功耗实际测量值为 8.085 W，预测值为 7.444 W，可以看出递归程序的能耗较大，原因在于递归程序因为要保护现场占用了太多的资源，因此在编写程序时不建议采用递归方法。本实验的第 2 组数据是重载和未采用重载操作的程序。采用重载操作的程序运行了 58.419 s，CPU 的功耗测量值为 77.826 W，预测值为 78.257 W；而未采用重载操作的程序运行了 28.817 s，CPU 的功耗测量值为 80.522 W，预测值为 79.280 W，可以看出采用重载操作的程序运行时间较长，因为带有多个重载操作符的表达式可能会导致临时对象创建中间结果。通过两组程序的能耗对比，为程序员编写高能效的程序提供了依据。

5　结　论

本文提出了一种利用机器学习算法预测软件能耗的方法，并建立了以软件性能事件为特征的能耗模型。在实验中使用了 4 种机器学习算法，通过预测误差对比实验选取了最适合的岭回归机器学习算法。实验结果表明，以测量能耗为基准的能耗预测误差率低于 9%，并且根据模型更容易分析能耗产生的原因。基于模型的能耗预测方法避免了采用程序代码插桩方式测量能耗会影响程序性能的缺点。本文还提出了一种基于性能特征的软件分类方法，即根据 cache_miss 的阈值将软件分类并分别建模。本模型对计算密集型和内存访问密集型程序的能耗有更精确的预测。下一步我们将基于神经网络和深度学习等算法，结合更大规模的程序数据训练样本，建立更为准确的能耗模型。

参考文献：

[1] HEDDEGHEM W V, LAMBERT S, LANNOO B, et al. Trends in worldwide ICT electricity consumption from 2007 to 2012 [J]. Computer Communications. 2014, 50(1): 64-76.

[2] AHMAD W, OLZENSPIESP K F H, STOELINGA M, et al. Green computing: power optimisation of VFI-based real-time multiprocessor dataflow applications [C]// Proceedings of the 2015 Euromicro Conference on Digital System Design. Piscataway, NJ, USA: IEEE, 2015: 272-275.

[3] GE Rong, VOGT R, MAJUMDER J, et al. Effects of dynamic voltage and frequency scaling on a K20 GPU [C]// Proceedings of the 2013 42nd International Conference on Parallel Processing. Piscataway, NJ, USA: IEEE, 2013: 826-833.

[4] MARTONOSI M, TIWARI V, BROOKS D. Wattch: a framework for architectural-level power analysis and

optimization [C]// Proceedings of the 27th International Symposium on Computer Architecture. Piscataway, NJ, USA: IEEE, 2000: 83-94.

[5] BRANCO D. Impact of GCC optimization levels in energy consumption during C/C++ program execution [C]// Proceedings of the 13th IEEE International Scientific Conference on Informatics. Piscataway, NJ, USA: IEEE, 2015: 52-56.

[6] SEO C, MALEK S, MEDVIDOVIC N. Component-level energy consumption estimation for distributed java-based software systems [C]// Proceedings of the International Symposium on Component-Based Software Engineering. Berlin, Germany: Springer-Verlag, 2008: 97-113.

[7] CHEN Hui, LI Youhuizi, SHI Weisong. Fine-grained power management using process-level profiling [J]. Sustainable Computing: Informatics and Systems, 2012, 2(1): 33-42.

[8] SHYE A, SCHOLBROCK B, MEMIK G. Into the wild: studying real user activity patterns to guide power optimizations for mobile architectures [C]// Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture. Piscataway, NJ, USA: IEEE, 2009: 168-178.

[9] GURUMURTHI S, SIVASUBRAMANIAM A, IRWIN M J, et al. Using complete machine simulation for software power estimation: the SoftWatt approach [C]// Proceedings of the 8th International Symposium on High-Performance Computer Architecture. Piscataway, NJ, USA: IEEE, 2002: 141-150.

[10] ZHANG Lide, TIWANA B, DICK R P, et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones [C]// Proceedings of the 2010 IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. Piscataway, NJ, USA: IEEE, 2010: 105-114.

[11] GE Rong, FENG Xizhou. ETune: a power analysis framework for data-intensive computing [C]// Proceedings of the 41st International Conference on Parallel Processing Workshops. Piscataway, NJ, USA: IEEE, 2012: 254-261.

[12] PATHAK A, HU Y C, ZHANG M, et al. Fine-grained power modeling for smartphones using system call tracing [C]// Proceedings of the 6th Conference on Computer Systems. New York, USA: ACM, 2001: 153-168.

[13] AGGARWAL K, ZHANG Chenlei, CAMPBELL J C, et al. The power of system call traces: predicting the software energy consumption impact of changes [C]// Proceedings of the 24th International Conference on Computer Science and Software Engineering. New York, USA: ACM, 2014: 219-233.

[14] Anon. Perf [EB/OL]. (2015-05-05)[2017-06-05]. https://perf.wiki.kernel.org/index.php/Tutorial.

[15] Anon. Programming tasks [EB/OL]. (2015-09-29)[2017-06-05]. http://rosettacode.org/wiki/Category:Programming_Tasks.

[16] Anon. SPEC CPU 2006 [EB/OL]. (2008-05-22)[2017-06-05]. http://www.spec.org/cpu2006/.

[17] Anon. SPEC OMP 2012. [EB/OL]. (2015-05-11)[2017-06-05]. http://www.spec.org/omp2012/.

[18] ZECENA I, BURTSCHER M, JIN T D, et al. Evaluating the performance and energy efficiency of n-body codes on multi-core CPUs and GPUs [C]// Proceedings of the 2013 32nd IEEE International Performance Computing and Communications Conference. Piscataway, NJ, USA: IEEE, 2013: 1-8.

[19] Anon. NPB3.0 [EB/OL]. (2016-03-21)[2017-06-05]. https://www.nas.nasa.gov/publications/npb.html.

[20] HASTIE T, TIBSHIRANI R, FRIEDMAN J. Linear methods for regression in the elements of statistical learning: data mining, inference, and prediction [M]// Springer Series in Statistics. Berlin, Germany: Springer, 2008: 43-52.

[21] Anon. Running average power limit-rapl [EB/OL]. (2014-06-06)[2017-06-05]. https://01.org/zh/blogs/2014/running-average-power-limit-rapl.

[22] Anon. Nvidia management library NVML [EB/OL]. (2017-10-31)[2017-06-05]. https://developer.nvidia.com/nvidia-management-library-nvml.

[23] Anon. Intel® Xeon Phi™ coprocessor developer's quick start guide [EB/OL]. (2012-12-01)[2017-06-05]. https://software.intel.com/en-us/articles/intel-xeon-phi-coprocessor-developers-quick-start-guide.

[24] ZONG Ziliang, GE Rong, GU Qijun. Marcher: a heterogeneous system supporting energy-aware high performance computing and big data analytics [J]. Big Data Research, 2017, 8: 27-38.

(编辑 武红江)