计算机科学
COMPUTER SCIENCE

基于训练集聚类选择优化的CPU功耗建模精度提升方法

李泽锴, 钟佳卿, 冯绍骏, 陈娟, 邓荣宇, 徐涛, 谭政源, 周柯杏, 朱鹏志, 马兆阳

引用本文

李泽锴, 钟佳卿, 冯绍骏, 陈娟, 邓荣宇, 徐涛, 谭政源, 周柯杏, 朱鹏志, 马兆阳. 基于训练集聚类选择优化的CPU功耗建模精度提升方法[J]. 计算机科学, 2024, 51(9): 59-70.

LI Zekai, ZHONG Jiaqing, FENG Shaojun, CHEN Juan, DENG Rongyu, XU Tao, TAN Zhengyuan, ZHOU Kexing, ZHU Pengzhi, MA Zhaoyang. CPU Power Modeling Accuracy Improvement Method Based on Training Set Clustering Selection [J]. Computer Science, 2024, 51(9): 59-70.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231100015

基于训练集聚类选择优化的CPU功耗建模精度提升方法

李泽锴　钟佳卿　冯绍骏　陈　娟　邓荣宇　徐　涛　谭政源　周柯杏　朱鹏志　马兆阳
国防科技大学计算机学院　长沙 410073
(zekaili@nudt.edu.cn)

摘　要　建立高精度、低开销的CPU功耗模型对于计算机系统的功耗管理与功耗优化至关重要。一般认为训练集规模越大，CPU功耗模型精度越高。但有研究发现增大训练集规模不一定会提高功耗建模精度，有时甚至会导致精度下降，因此，如何选择功耗模型训练集以保证CPU功耗模型精度达到要求具有重要意义。文中提出一种基于聚类的训练集选择优化算法来解决上述问题，在有效保证CPU功耗建模精度的同时降低了CPU功耗建模的开销。该算法首先通过主成分分析将基于PMC的程序特征转换为 p 维向量特征空间，然后根据找到的最优聚类数按照程序特征对程序进行聚类，从每个聚类簇中选出代表程序；最后根据“单聚类簇内代表性最强原则”与“多聚类簇间代表程序数最少原则”形成最优训练集，模型精度相比Baseline精度有明显提高。在x86和ARM两类处理器平台上分别采用线性功耗建模和神经网络功耗建模两种方式，对算法进行了实验评估，实验结果表明所提算法的功耗建模精度有效显著提升。

关键词：CP功耗建模；训练集选择；主成分分析；K-means聚类
中图分类号　TP302

CPU Power Modeling Accuracy Improvement Method Based on Training Set Clustering Selection

LI Zekai, ZHONG Jiaqing, FENG Shaojun, CHEN Juan, DENG Rongyu, XU Tao, TAN Zhengyuan, ZHOU Kexing, ZHU Pengzhi and MA Zhaoyang
College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China

Abstract　Building a high-precision and low-cost CPU power model is crucial for power management and power optimization of computer systems. It is generally believed that the larger the size of the training set, the higher the accuracy of the CPU power model. However, some studies have found that increasing the size of the training set may not necessarily improve the accuracy of power modeling, or even sometimes leading to a decrease in accuracy. Therefore, it is necessary to screen the training set of the power model to ensure that the accuracy of the CPU power model does not decrease while achieving a low-cost target for model training. This paper proposes an optimization algorithm for training set selection based on clustering. It first converts PMC-based program features into a p-dimension vector feature space through principal component analysis (PCA), then clusters the programs according to the optimal number of clusters found, and selects representative programs from each cluster. Finally, according to the principle of selecting the strongest representative program within a single cluster and selecting the least number of representative programs among multiple clusters, a low-cost training set is achieved for a significant reduction in training overhead without loss of modeling accuracy. Experimental evaluation of the algorithm is conducted on both x86 and ARM-based processor platforms using linear power modeling and neural network power modeling, and the experimental results validate the effectiveness of the algorithm. These results indicate a significant improvement in CPU power consumption model accuracy.

Keywords　CPU power modeling, Training set selection, Principal component analysis, K-means clustering

1　引言

对于处理器实时功耗建模及预测，精度要求越高，实时功耗预测的难度就越大[1-2]。在进行高精度、低开销的CPU实时功耗建模过程中，选择合适的训练集非常重要，因为它对功耗模型精度及开销均会产生影响[3]。

一般认为建模训练集规模越大则功耗模型精度越高，但事实是，增加训练集数量并不一定能保证获得更高的CPU功耗模型精度，甚至还会出现精度下降的情况。这是因为高精度的CPU功耗模型要能反映各种情况下CPU的实时功耗状态，要求训练程序特征必须足够多样，能够反映尽可能多的程序特征。但是，即使训练集包含的样本功耗状态足够多样，如果每种状态下只有少量样本，或者样本总数较多，但在各种状态下的分布不均匀，都可能影响模型精度。在实际模型训练过程中，往往难以达到包含足够的多种状态并且每种状态下训练样本足够多的要求，因此训练集的选择对提高CPU功耗建模精度显得尤为重要。在保证精度相同的情况下，希望选择尽可能少的训练程序以降低模型训练开销。

研究训练集的选择对CPU功耗建模精度的影响存在挑战，在建模过程中，大多数研究人员不是根据理论分析而是根据

到稿日期：2023-11-01　返修日期：2023-12-19
通信作者：陈娟(juanchen@nudt.edu.cn)

经验选取训练集，有时会导致 CPU 功耗建模精度不高，尤其是神经网络这种对样本数量有一定要求的模型，样本数量和分布对功耗建模精度会比较敏感。目前存在各种类型的 CPU 功耗模型，包括线性模型和非线性模型，其中线性模型包括经典回归模型[4]和基于主成分分析等统计学手段的模型[5]，而非线性模型近年来又新出现一些机器学习方法[6]、神经网络[7]、深度神经网络[8]等。这些模型在选择训练集时通常不会进行量化分析，往往认为功耗建模精度只依赖于模型本身。而本文实验发现，有时训练集的选择对 CPU 功耗建模精度的影响很大。对于提高 CPU 功耗模型精度而言，训练集并不是越多越好。近年来一些研究采用自适应反馈机制，通过动态功耗建模方法来提高 CPU 功耗建模精度[9]，这也反映了程序特征的差异性、多样性对 CPU 功耗建模精度的影响。

如何选择训练集，以减少随意选取训练集对 CPU 实时功耗建模精度带来的不良影响？本文关注以下两个问题：

1）如何分析训练集分布特点对 CPU 实时功耗建模精度的影响？

2）如何选取合适的训练集来保证 CPU 功耗建模精度？

为了回答问题 1，本文从训练集在样本空间分布特点的角度分析不同训练集对功耗建模精度的影响，包括训练集与预测集之间的相似度以及训练集分布均匀程度对功耗建模精度的影响（详见第 3 章）。

为了回答问题 2，本文提出了“基于 K-means 聚类的训练集选择优化算法”，该算法先通过主成分分析（Principal Component Analysis，PCA）将基于 PMC 的程序特征转换为 p 维向量特征空间；然后根据找到的最优聚类数按照程序特征对程序进行聚类，从每个聚类簇中选出代表程序；最后根据“单聚类簇内代表性最强原则”与“多聚类簇间代表程序数最少原则”得到最优训练集，提高了 CPU 功耗建模精度，同时降低了 CPU 功耗模型的训练开销（详见第 4 章）。

本文在 x86 和 ARM 两类处理器平台上，对基于 K-means 聚类的训练集选择优化算法的有效性进行了验证，采用了线性功耗建模、神经网络功耗建模两种方式。候选训练集包括 SPEC2016[10]，HPCC[11]等。实验结果表明，本文提出的算法得到的 CPU 功耗模型与 Baseline（详见第 5.3 节）相比，MRE 在 x86 平台上平均降低了 15.65%，在 ARM 平台上平均降低了 36.25%。

本文第 2 章介绍了本文的研究动机，列举了影响 CPU 功耗建模精度的因素；第 3 章详细分析了不同训练集选择对 CPU 功耗建模精度的影响；第 4 章介绍了基于 K-means 聚类的训练集选择优化算法；第 5 章全面分析了实验结果；第 6 章介绍了相关工作；最后总结全文。

2 研究动机

一个 CPU 功耗模型的精度受到训练集和模型本身两方面的影响，在不改变模型的前提下，训练集的选择就成为影响功耗建模精度的另一个重要因素。

在进行 CPU 实时功耗建模过程中，我们发现，使用更多的训练集不一定会提高功耗模型精度。如表 1 所列，分别选用线性模型和神经网络模型进行 CPU 功耗建模，当训练集规模（训练集中程序数量）为 6 时，CPU 功耗建模的平均相对误差比规模为 2 和 4 时更大。出现这种现象的原因是，当使用更多的程序加入训练集时，虽然程序特征的数量更多，但是训练程序在特征上可能不具备功耗建模精度所要求的模型特征代表性、分布均匀性等，因此可能会产生过拟合等现象，功耗建模精度反而下降。这里使用的是 Intel Xeon Gold 6226R 处理器平台，待选的训练基准测试集为 SPEC2016[10]，HPCC[11]，PARSEC[12]，GRAPH500[13]，SMG2000[14]，HPCG[15]，HPL-AI[16]测试集。采样实时 PMC 数据进行功耗建模。

表 1 不同规模训练集上线性/非线性功耗建模精度对比

Table 1 Comparison of linear/nonlinear power consumption modeling accuracy on training sets of different sizes

训练集规模（程序数量）	平均相对误差（MRE）（线性模型）/%	平均相对误差（MRE）（神经网络模型）/%
2	4.387	4.113
4	4.318	3.675
6	4.457	4.564

因此，需要分析训练集的选择对 CPU 功耗模型精度的影响，从而选择合适的训练集来保证 CPU 功耗建模精度，并在一定程度上，降低功耗建模开销。Eriksson 等[17]在关于 QSAR（Quantitative Structure-Activity Relationship）模型建模时部分模型可靠度低的调研中发现，训练集的选取对模型准确度影响较大。选择不同的训练集最多可以造成 10% 的 RMSEP（均方根误差）差距（0.55 vs 0.6）。在模型训练中，Eriksson 等发现输入数据冗余且难以分析。他们首先使用 PCA 对输入进行降维，得到两个主成分，然后使用这两个主成分对输入数据进行分析，最后根据不同的选取原则，选择 5 种不同的训练集并对它们的模型精度进行对比。结果表明，使用提供的 20% 的训练集就可以得到一个精度足够高的模型。

3 分析不同训练集对功耗建模精度的影响

CPU 功耗建模时通常会考虑线性模型、非线性模型等，近年来也出现了采用神经网络进行 CPU 功耗建模。本文考虑了线性模型和神经网络模型两种。

尽管线性模型相比非线性模型有时难以达到高精度的功耗建模要求，但线性模型建立简单且训练速度快，因此本文选用了多元线性回归模型[18]对 CPU 功耗进行建模，并选用了 5 个与功耗相关性较大的 PMC（性能监控事件）作为线性模型的项。

本文选用的是 Gutierrez 等[18]提出的神经网络模型，它同样也是基于 PMC 建模的。网络包含 5 个输入节点（每个节点代表一个 PMC 事件）、2 个隐藏层（分别具有 5 个和 3 个节点），以及 1 个输出节点，输出预测得到 CPU 功耗。隐藏层节点间采用 sigmoid 函数连接。

本文使用一个 $p \times t$ 数组来表示一段时间 t 内的程序特征，其中 p 表示有 p 组程序特征（这里用 PMC），t 表示时间长度。如果选择的训练集的程序特征不能覆盖整个特征空间，那么将很难达到较高的 CPU 功耗模型精度。

由于 p 个程序特征之间可能存在一定的相关性，直接用 p 维向量进行建模可能会影响精度，因此本文采用 PCA[19]将 p 维向量降为 p' 维向量，以消除程序特征之间的相关性，减少

对功耗建模精度的影响。下文中 p 维向量均表示通过 PCA 降维之后的 p' 维向量，但为了符号的通用性，仍记为 p 维向量。

定义程序 i 和程序 j 的特征差异距离 $d_{i,j}$。使用一个 t 维向量表示一个程序特征在运行时间 t 内的读数。这样一个程序的 p 个程序特征就是 p 个 t 维向量。对于一个程序特征 $k(1\leqslant k\leqslant p)$，可以计算出程序 i,j 在第 k 个特征上的 t 维向量欧氏距离（记为 $dist_{i,j}^{k}$）。那么程序 i,j 的特征差异距离 $d_{i,j}=\frac{\sum_{k=1}^{p} dist_{i,j}^{k}}{p}$。定义 $d_{i,j}$ 的阈值为 d_{th}。如果 $d_{i,j}$ 大于该阈值，则认为两个程序的差异性大；否则认为它们的差异性小。d_{th} 的值根据实验经验设定，规定 $d_{i,j}$ 前 40% 小的程序对为差异性小的程序对，根据表 2 中任意两个程序之间的特征差异距离 $d_{i,j}$，设定阈值 $d_{\mathrm{th}}=1\times10^{-6}$。

表 2　任意两个程序 i,j 之间的特征差异距离 $d_{i,j}$

Table 2　Feature difference distance $d_{i,j}$ between any two programs i,j

程序 i	程序 j						
	Graph500	HPCC	HPCG	HPL-AI	PARSEC	SMG2000	SPEC
Graph500	0.00						
HPCC	7.72×10^{-7}	0.00					
HPCG	1.25×10^{-6}	8.67×10^{-7}	0.00				
HPL-AI	4.86×10^{-6}	4.49×10^{-6}	5.23×10^{-6}	0.00			
PARSEC	1.28×10^{-6}	8.24×10^{-7}	1.18×10^{-7}	5.14×10^{-6}	0.00		
SMG2000	1.37×10^{-6}	1.38×10^{-6}	7.12×10^{-7}	5.86×10^{-6}	8.29×10^{-7}	0.00	
SPEC	2.62×10^{-6}	2.37×10^{-6}	1.50×10^{-6}	6.57×10^{-6}	1.55×10^{-6}	1.31×10^{-6}	0.00

3.1　训练集与预测集之间的相似度对精度的影响

用 $S_{\mathcal{T},\mathcal{P}}$ 表示训练集 $\mathcal{T}$ 与预测集 $\mathcal{P}$ 之间的特征相似度，$S'_{\mathcal{T},i}$ 表示训练集 $\mathcal{T}$ 与一个预测集程序 i 之间的特征相似度。同时，为了对 $d_{i,j}$ 进行归一化，令 R 表示覆盖所有程序点[1)] 的最小矩形的对角线长。

$$S'_{\mathcal{T},i}=\max_{\forall j\in\mathcal{T}}\left\{1-\frac{d_{i,j}}{R}\right\} \tag{1}$$

$$S_{\mathcal{T},i}=\min_{\forall i\in\mathcal{P}}\{S'_{\mathcal{T},i}\} \tag{2}$$

同时用 S_{th} 表示 $S_{\mathcal{T},\mathcal{P}}$ 的阈值。如果 $S_{\mathcal{T},\mathcal{P}}<S_{\mathrm{th}}$，则认为训练集与预测集的特征相似度较小，否则认为它们的相似度较大。$S_{\mathcal{T},\mathcal{P}}$ 越大，期望的功耗建模精度越高。与 d_{th} 类似，根据实验经验，设定 $S_{\mathrm{th}}=0.7$。

下面对比在 $S_{\mathcal{T},\mathcal{P}}$ 不同取值下功耗模型精度的差异。现构造两组训练集和预测集，第一组选择 PARSEC+SMG2000 作为训练集，HPCC+HPCG 作为预测集，此时 $S_{\mathcal{T},\mathcal{P}}=0.794$，大于 $S_{\mathrm{th}}=0.7$，代表训练集与预测集相似度大的情况；第二组选择 HPCC+SPEC 作为训练集，SMG2000+HPCG 作为预测集，此时 $S_{\mathcal{T},\mathcal{P}}=0.019$，远小于 $S_{\mathrm{th}}=0.7$，代表训练集与预测集相似度小的情况。从表 3 中可以看出，相似度大的组平均相对误差为 2.8% 和 4.1%（分别使用线性模型和神经网络模型，下同），远低于相似度小的组的 22.6% 和 9.5%，验证了训练集和预测集之间的相似度越大，功耗模型预测精度越高。

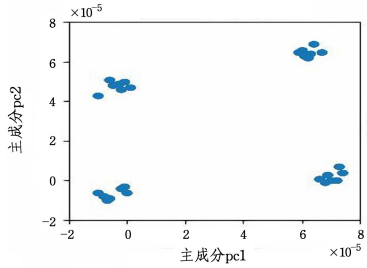
表 3　训练集 T 与预测集 P 之间的特征相似度 $S_{\mathcal{T},\mathcal{P}}$ 对功耗模型精度的影响

Table 3　Influence of feature similarity between training set T and prediction set P on the accuracy of power model

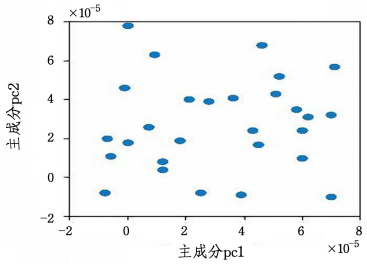
不同相似度	不同模型	
	平均相对误差(MRE)（线性模型）/%	平均相对误差(MRE)（神经网络模型）/%
相似度大	2.8	4.1
相似度小	22.6	9.5

3.2　训练集分布均匀程度对功耗建模精度的影响

训练集分布均匀程度越好，功耗建模精度就越高。如果训练集多数程序间 $d_{i,j}\leqslant d_{\mathrm{th}}$，在样本空间中分布就集中，从而导致训练集与大多数预测集之间的相似度小，模型预测精度低，泛化能力弱。因此，为提高模型预测精度，增强泛化能力，应选用分布均匀的训练集训练模型。图 1 给出了一组训练集分布均匀程度的例子，左侧图程序集中分布在 4 个区域，分布均匀程度差，这可能导致模型对这部分样本空间过度训练，造成过拟合，从而降低功耗模型精度；在右侧图中，多数程序之间 $d_{i,j}>d_{\mathrm{th}}$，训练集分布均匀程度优。



(a)训练集分布均匀程度差



(b)训练集分布均匀程度优

图 1　一组训练集分布均匀程度的例子

Fig. 1　Example of distribution uniformity of training set

下面对比使用不同分布均匀程度的训练集建模的功耗模型精度差异。构造两组相同规模的训练集：第一组选择 SMG2000+HPL-AI+HPCC+SPEC2016 作为训练集；第二组选择 PARSEC+HPCG+HPCC+SPEC2016 作为训练集。第一组训练集中，任意两程序之间的特征差异距离 $d_{i,j}>d_{\mathrm{th}}$，分布均匀程度优。第二组训练集中，PARSEC 与 HPCC，

1) 程序点指特征向量经过 PCA 降维后在平面上对应的点

HPCG之间均有$d_{i,j} \leqslant d_{th}$，程序间差异性小，分布均匀程度差。对两组训练集进行功耗建模之后，分别对所有程序组成的预测集进行预测。为保证训练集和预测集不重复，这里仅随机选出程序10%的数据作为训练集，剩余的90%数据作为预测集。从表4可以看出，第一组训练集的平均相对误差为4.57%和2.76%（分别为线性模型和神经网络模型，下同），第二组平均训练集的相对误差为28.39%和4.34%，这表明训练集分布均匀程度越好，功耗模型预测精度越高。

表4 训练集分布均匀程度对功耗模型精度的影响

Table 4 Influence of distribution uniformity of training set on the accuracy of power model

分布均匀程度	不同模型	
	平均相对误差(MRE)(线性模型)/%	平均相对误差(MRE)(神经网络模型)/%
优	4.57	2.76
差	28.39	4.34

4 算法描述

基于第3章不同训练集对功耗建模精度影响的分析，本章进一步研究如何选择训练集来保证功耗建模精度不受影响，且尽可能减少训练时间开销。本文提出了基于*K*-means聚类的训练集选择优化算法，简称训练集选择算法。图2给出了训练集选择算法的流程图，具体如算法1所示。

算法1 基于*K*-means聚类的训练集选择优化算法

输入：N个程序运行时读取的实时PMC数据P_i(i=1,2,3,…,N)，初始聚类数k_0(默认为2)

输出：训练集T^*，满足$M \leqslant |T^*| \leqslant N$，其中M为最优聚类数

1. 对数据进行预处理，得到每个程序对应的特征向量Vec_i。
2. 令$k=k_0$。
3. 对所得特征向量集合按k类进行K-means聚类。
4. 根据CH指标对聚类进行评估，得到对应的指标值ch_k。
5. 枚举$k=(k_0+1)\sim N$，并重复步骤3—步骤4。
6. 比较所有的ch_k($k_0 \leqslant k \leqslant N$)，选出其中的最小值，将该值对应的k赋值给M。
7. 对特征向量集合按M类进行K-means聚类。
8. 按照“单聚类簇内代表性最强原则”，从上一步聚类结果的每一簇(共M簇)中选出1—2个程序作为代表程序，构成该簇的代表程序集。
9. 按照“多聚类间代表程序数最少原则”，对代表程序集进行筛选，去掉冗余的代表程序(基于聚类的冗余代表程序迭代筛选方法)，得到训练集T^*，有$M \leqslant |T^*| \leqslant N$。

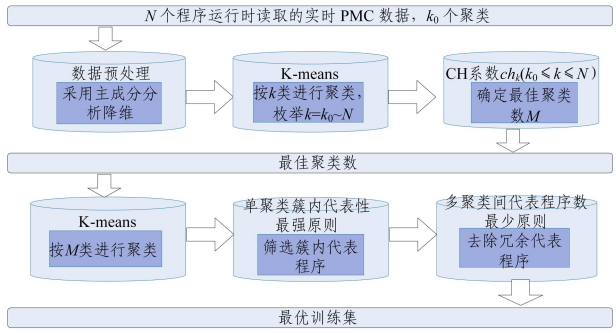


图2 基于*K*-means聚类的训练集选择优化算法流程图

Fig. 2 Flowchart of training set selection optimization algorithm based on *K*-means clustering

当训练集$\mathcal{T}$中一定数量的程序之间的特征差异距离$d_{i,j}$小于阈值d_{th}时，容易出现过拟合，进而导致功耗建模精度受影响。为了减少训练集选择不当带来的精度下降，采用聚类方法将所有待选择的训练程序分为多种类型，其中每一类中任意两个程序之间的特征差异距离$d_{i,j}$小于阈值d_{th}。从同一类中只选择少量(1个或2个)代表程序，以减少训练集中出现大量特征相近程序的情况，并减少功耗建模训练时间开销。图3给出了ARM处理器平台上基于聚类的冗余代表程序迭代筛选方法的执行过程示例。

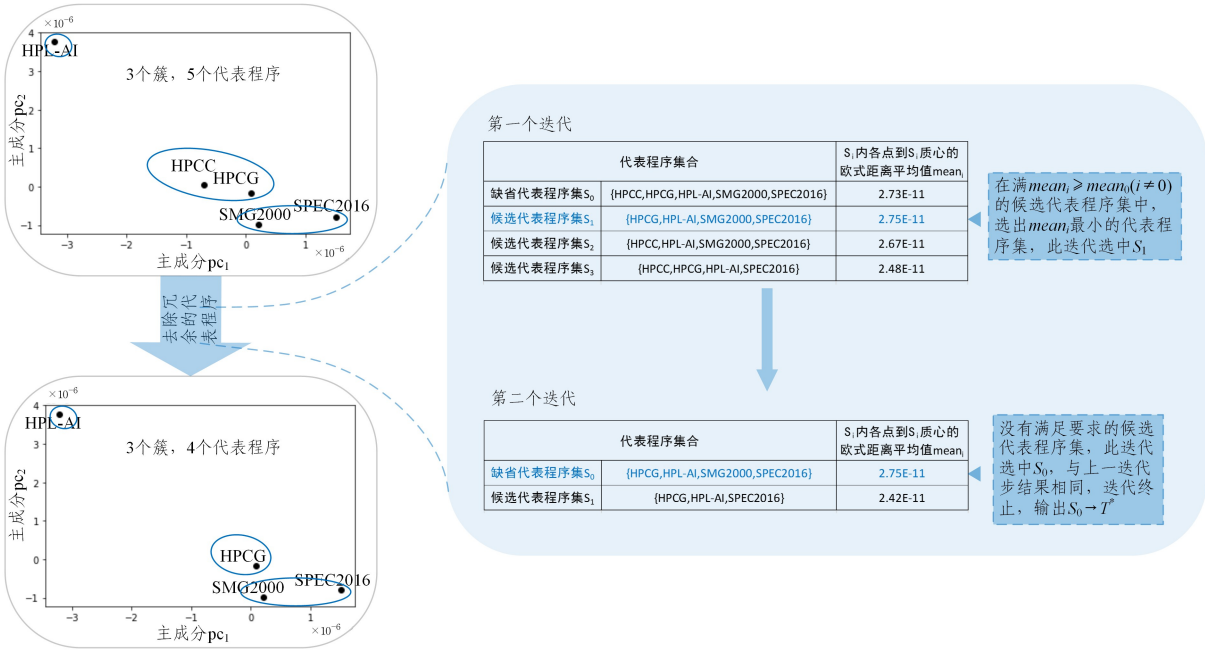


图3 ARM处理器平台上基于聚类的冗余代表程序迭代筛选方法的执行过程示例

Fig. 3 Example of redundant representative program iterative screening method based on clustering on ARM platform

算法的输入是 N 个 $p\times t$ 大小的矩阵 $\boldsymbol{P}_i(i=1,2,\cdots,N)$ 和初始聚类数 k_0（默认为 2），矩阵 $\boldsymbol{P}_i$ 表示第 i 个程序的 p 个程序特征在运行时间 t 内的读数。算法的输出是在最优聚类数 $M(M\leqslant N)$ 下的优化训练集 T^*。

训练集选择算法主要涉及以下 3 个方面：

1）数据预处理（对应算法步骤 1）。通过 PCA 降维将 n 个程序的程序特征降为 $p(p=2)$ 维向量，降维后的 p 维程序特征向量记为 $\boldsymbol{Vec}_i(i=1,2,\cdots,N)$。

2）寻找最优聚类数（对应算法步骤 2－步骤 6）。因为 K-means 聚类的最优聚类数 M 依赖于实际的训练集程序特征，所以算法遍历了所有可能的聚类数，选择 CH[20] 指标最优的聚类数作为 M 的值。CH 指标是一种常用的 K-means 聚类效果评估标准，其计算式为：

$$s=\frac{tr(B_k)(n-k)}{r(W_k)(k-1)} \tag{3}$$

$$\boldsymbol{B}_k=\sum_{q=1}^{k}n_q(c_q-c_e)(c_q-c_e)^{\mathrm{T}} \tag{4}$$

$$\boldsymbol{W}_k=\sum_{q=1}^{k}\sum_{x\in C_q}(x-c_q)(x-c_q)^{\mathrm{T}} \tag{5}$$

其中，tr 表示矩阵的迹，n 表示数据规模（程序数量），k 表示聚类数，c_q 表示类 q 的中心点，c_e 表示数据集的中心点，n_q 表示类 q 中的数据规模，C_q 表示类 q 的数据集合，$\boldsymbol{B}_k$ 为类间离差矩阵，$\boldsymbol{W}_k$ 为类内离差矩阵。

3）训练集生成（对应算法步骤 7－步骤 9）。对特征向量集合按 M 类进行 K-means 聚类之后，从局部和全局两个角度，分别根据“单聚类簇内代表性最强原则”和“多聚类间代表程序数最少原则”，分两步得到最优训练集 T^*，具体如下：

（1）用“单聚类簇内代表性最强原则”来选择每个聚类簇的代表程序集。为了保证每个聚类簇选出的代表程序能够真正“代表”该簇的所有程序（指功耗建模精度与 Baseline 相当），首先选择距离该簇质心（原质心）最近的程序作为初始代表程序；然后将初始代表程序与该簇内所有其他程序一一构成代表程序对，计算每个代表程序对的新质心（为这两个程序的中心位置），由此得到一组新质心；最后，从这组新质心中选出一个新质心，满足选出的新质心与所有程序的距离几何平均值 d_1 和原质心与所有程序的距离几何平均值 d_2 偏差比最小，且小于给定阈值。如果能找到满足条件的新质心，则将对应代表程序对的 2 个程序构成该簇的代表程序集，否则该簇的代表程序集仅包含初始代表程序。最终，对于 M 个聚类簇，选出大小为 $n(M\leqslant n\leqslant 2M)$ 的代表程序 S_0。这里的阈值根据实验统计经验获得，在第 5.1 节中定义了两个平台（x86 处理器平台、ARM 处理器平台）的阈值均为 5%。

（2）用“多聚类簇间代表程序数最少原则”来去除冗余代表程序（基于聚类的冗余代表程序迭代筛选方法）。为尽量减少功耗模型训练开销，我们在上一步的基础上筛选冗余的代表程序，基本思想是迭代地逐步去除冗余程序。在每一个迭代步中，首先寻找每两个簇中距离最近的两个代表程序构成重叠区间，然后每次删除重叠区间的一个代表程序形成一个候选代表程序集。因此，按照组合数计算，对于 M 个聚类簇，总共有 C_M^2 个重叠区间，有 $2C_M^2=M(M-1)$ 个候选代表程序集。这里用 S_j 表示候选代表程序集，其中 $j=1,2,3,\cdots,2C_M^2$。S_0 表示缺省代表程序集，即上一个迭代步得出的输出（在第一个迭代步中 S_0 为步骤（1）中得出的代表程序集）。对于每个代表程序集 S_j，计算 S_j 内各点到 S_j 质心的欧氏距离的平均值 $mean_j$，然后从候选代表程序集 S_j 中选出一个满足（$mean_j\geqslant mean_0$）且（$mean_j-mean_0$）最小的候选代表程序集 S_k。若没有满足条件的 j，则令 $S_k=S_0$。本迭代步结束，S_k 为本迭代步输出。

当一个迭代步的输出 S_k 与上一迭代步的输出结果相等时，迭代终止，本迭代步的输出 S_k 即为算法的输出训练集 T^*。

在数据预处理中提到的主成分分析[19]是一种统计方法。通过正交变换将一类可能存在相关性的变量转换为一类线性不相关的变量，转换后的这类变量被称作主成分。PCA 降维过程如下：

1）对 N 个程序对应的矩阵 $\boldsymbol{P}_i$ 求出协方差矩阵 $\boldsymbol{C}_i$；

2）对于每个协方差矩阵 $\boldsymbol{C}_i$，求出特征值 E_j 和对应的特征向量 $\boldsymbol{Vec}_j$；

3）对于每个程序，选出 $\boldsymbol{C}_i$ 最大的前两个特征值对应的两个特征向量组成映射矩阵 $\boldsymbol{A}_i$；

4）对于每个程序，使用映射矩阵 $\boldsymbol{A}_i$ 对 $\boldsymbol{P}_i$ 进行映射，得到两个主成分 pc_1^i，pc_2^i；

5）将 N 个程序对应 t 个时刻求出的主成分分别求出平均值，作为 N 个程序的程序特征向量 $\boldsymbol{Vec}_i=(pc_1^i,pc_2^i)(i=1,2,3,\cdots,N)$。

5 实验验证

5.1 实验环境

本文的实验是在两个平台上进行的，平台 A 和平台 B 分别代表 ARM 处理器平台和 x86 处理器平台，具体配置如表 5 所列。实验所涉及的 7 个程序代表 7 个基准测试集（见表 6）。实验中使用的线性模型为多元线性回归模型[18]；神经网络模型为文献[18]中提到的一种多层前馈结构神经网络模型，网络包含两个隐藏层，分别具有 5 个和 3 个节点，隐藏层节点之间采用 sigmoid 函数连接。本文选用 5 个与功耗相关性较大的 PMC（branch-misses，cpu-cycles，instructions，L1-dcache-loads，L1-dcache-load-misses）对线性模型和神经网络模型进行建模。

表 5 平台硬件环境

Table 5 Hardware environment of two platforms

硬件	平台	
	平台 A	平台 B
CPU	ARM 处理器	Intel Xeon Gold 6226R
DRAM/GB	64	64

表 6 使用的测试集、程序个数

Table 6 Test set used in experiment and number of program

测试集	程序个数
SPEC2016[10]	43
HPCC[11]	5
PARSEC[12]	12
Graph500[13]	2
SMG2000[14]	1
HPCG[15]	1
HPL-AI[16]	1

5.2 总体结果及分析

本节给出训练集选择算法的总体结果，对比的基准程序（Baseline）是基于特征差异阈值的分类法进行数据集筛选。

实验结果包含在 x86 处理器平台和 ARM 处理器平台上数据预处理后的程序特征向量、算法聚类结果和代表程序选择结果，以及分别采用线性功耗模型与神经网络功耗模型的功耗建模精度和功耗建模开销。其中，功耗建模开销用功耗建模所需的时间来度量，主要包括训练集数据采集和数据筛选花费的时间，以及模型训练消耗的时间。

5.2.1 总体结果

表 7 中列出了经过数据预处理后每个程序对应的特征向量$\boldsymbol{Vec}_i$；同时，表 8 和表 9 中列出了算法的聚类结果和代表程序选择结果。图 4 和图 5 中给出了训练集选择算法的精度提升效果[1)]。其中 Baseline 选择了 PARSEC，SMG2000，Graph500 作为训练集。

表 7 不同处理器平台上数据预处理得到的向量结果

Table 7 Vector results obtained from data preprocessing on different processor platforms

Benchmark	x86 预处理向量	ARM 预处理向量
Graph500	$(1.926\times10^{-6}, 9.068\times10^{-7})$	$(-1.104\times10^{-6}, -6.149\times10^{-7})$
HPCG	$(2.176\times10^{-6}, -1.791\times10^{-5})$	$(1.004\times10^{-7}, -2.823\times10^{-7})$
HPCC	$(3.022\times10^{-5}, -6.304\times10^{-6})$	$(-7.027\times10^{-7}, 4.392\times10^{-8})$
PARSEC	$(-2.481\times10^{-5}, 3.502\times10^{-7})$	$(9.411\times10^{-8}, -1.647\times10^{-7})$
HPL-AI	$(1.656\times10^{-5}, -3.161\times10^{-5})$	$(-3.212\times10^{-6}, 3.765\times10^{-6})$
SPEC2016	$(-8.905\times10^{-6}, 7.564\times10^{-5})$	$(1.506\times10^{-6}, -8.031\times10^{-7})$
SMG2000	$(-2.378\times10^{-5}, -1.148\times10^{-5})$	$(2.133\times10^{-7}, -9.85\times10^{-7})$

表 8 x86 处理器平台聚类结果和代表程序选择结果

Table 8 Clustering results on x86 processor platform and representative program selection results

聚类结果		代表程序
簇编号	簇中所有程序	
第 1 簇	HPCC	HPCC
第 2 簇	SPEC2016	SPEC2016
第 3 簇	Graph500，PARSEC，SMG2000	SMG2000
第 4 簇	HPCG，HPL-AI	HPCG

表 9 ARM 处理器平台聚类结果和代表程序选择结果

Table 9 Clustering results on ARM processor platform and representative program selection results

聚类结果		代表程序
簇编号	簇中所有程序	
第 1 簇	HPCC，HPCG，Graph500，PARSEC	HPCG
第 2 簇	HPL-AI	HPL-AI
第 3 簇	SPEC2016，SMG2000	SMG2000

结果显示，与 Baseline 相比，在 x86 处理器平台上，训练集选择算法使 CPU 功耗模型的 MRE 分别降低了 5.55% 和 28.31%（分别使用线性模型和神经网络模型，下同），平均 MRE 降低了 15.65%，功耗模型训练开销降低明显，线性模型和神经网络模型建模开销均降低达 26.03%；在 ARM 处理器平台上，训练集选择算法的 MRE 分别降低了 41.79% 和 21.42%，平均 MRE 降低了 36.25%。

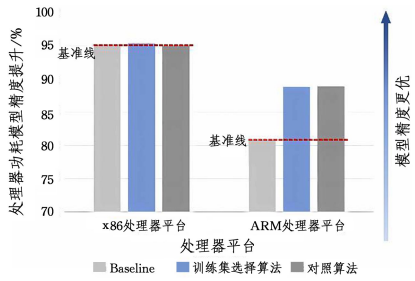


图 4 训练集选择优化算法与 Baseline 的精度对比（线性模型）

Fig. 4 Accuracy comparison between training set selection optimization algorithm and Baseline(linear model)

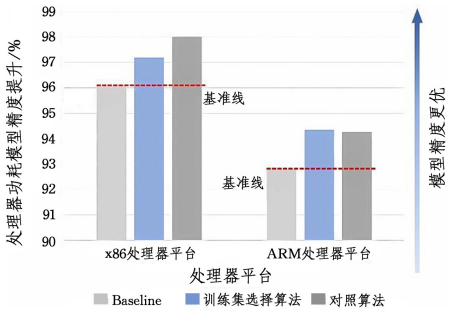


图 5 训练集选择优化算法与 Baseline 的精度对比（神经网络模型）

Fig. 5 Accuracy comparison between training set selection optimization algorithm and Baseline(neural network model)

表 10 和表 11 列出了训练集选择算法和 Baseline 的功耗建模开销，结果表明训练集选择算法和对照算法均能有效降低功耗建模开销。表 12 和表 13 列出了训练集选择算法和对照算法的功耗模型精度，结果显示，在 x86 处理器平台上，对照算法在降低功耗建模开销的同时可能会导致功耗模型精度降低，而训练集选择算法能在降低功耗建模开销的同时保证功耗模型精度。Baseline 算法以及对照算法将在第 5.3 节中进行介绍。

表 10 训练集选择优化算法以及对照算法应用于线性模型的功耗建模开销

Table 10 Modeling consumption obtained by optimization algorithm for training set selection and contrast algorithm applied to linear model

(s)

算法	平台	
	功耗建模开销（x86 处理器平台）	功耗建模开销（ARM 处理器平台）
Baseline	76531.64	5012.38
训练集选择优化算法	56609.82	33564.52
对照算法	98977.42	47287.68

1) 精度定义为：精度 $=1-MRE$

表 11 训练集选择优化算法以及对照算法应用于神经网络模型的功耗建模开销

Table 11 Modeling consumption obtained by optimization algorithm for training set selection and contrast algorithm applied to neural network model

(s)

算法	平台	
	功耗建模开销(x86 处理器平台)	功耗建模开销(ARM 处理器平台)
Baseline	77132.74	5618.69
训练集选择优化算法	57058.25	33931.67
对照算法	99567.83	47792.45

表 12 训练集选择优化算法以及对照算法应用于简单线性模型得到的平均相对误差

Table 12 MRE obtained by optimization algorithm for training setselection and contrast algorithm applied to linear model

(%)

算法	平台	
	平均相对误差(MRE)(x86 处理器平台)	平均相对误差(MRE)(ARM 处理器平台)
Baseline	4.917	19.224
训练集选择优化算法	4.644	11.190
对照算法	5.022	11.143

表 13 训练集选择优化算法以及对照算法应用于神经网络模型的平均相对误差

Table 13 MRE obtained by optimization algorithm for training set selection and contrast algorithm applied to neural network model

(%)

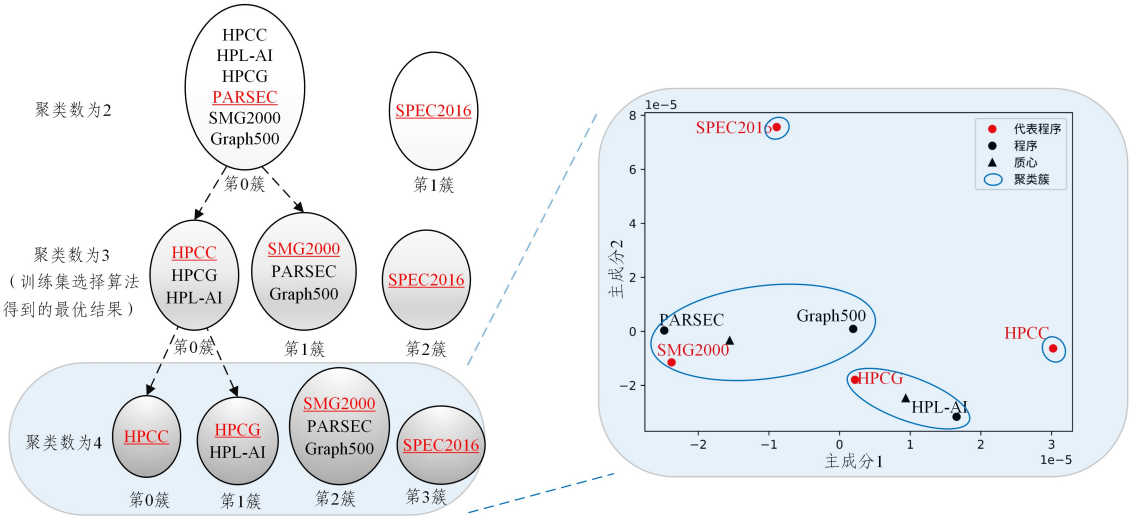
算法	平台	
	平均相对误差(MRE)(x86 处理器平台)	平均相对误差(MRE)(ARM 处理器平台)
Baseline	3.920	7.193
训练集选择优化算法	2.810	5.652
对照算法	1.985	5.726

5.2.2 各种参数变化下的结果分析

1)聚类数分析

图 6 和图 7 分别给出了在 x86 处理器平台和 ARM 处理器平台上不同聚类数下的聚类结果。表 14 和表 15 中列出了不同聚类数 k 的 CPU 功耗建模开销对比，结果表明在 x86 处理器平台和 ARM 平台上功耗建模开销均随聚类数 k 的增大而降低。

表 16 和表 17 列出了不同聚类数 k 的 CPU 功耗建模精度对比结果。在 x86 处理器平台上，使用线性模型进行功耗建模，使用最优聚类数 $k=4$ 训练得到的 CPU 功耗模型 MRE 为 4.644%，相比聚类数 $k=2$ 和 $k=3$ 的 MRE 分别降低 0.194%和 0.002%；使用神经网络模型进行功耗建模，使用最优聚类数 $k=4$ 训练得到的功耗模型 MRE 为 2.810%，相比聚类数 $k=2$ 和 $k=3$ 的 MRE 分别降低 1.474%和 0.137%。在 ARM 处理器平台上，使用线性模型进行功耗建模，使用最优聚类数 $k=4$ 训练得到的功耗模型 MRE 为 11.190%，相比聚类数 $k=2$ 的 MRE 降低 5.308%，与使用聚类数 $k=3$ 的结果相同；使用神经网络模型进行功耗建模，使用最优聚类数 $k=4$ 训练得到的功耗模型 MRE 为 5.652%，相比聚类数 $k=2$ 和 $k=3$ 的 MRE 分别降低 4.961%和 1.539%。这验证了训练集选择算法中最优聚类数选择方法的有效性。



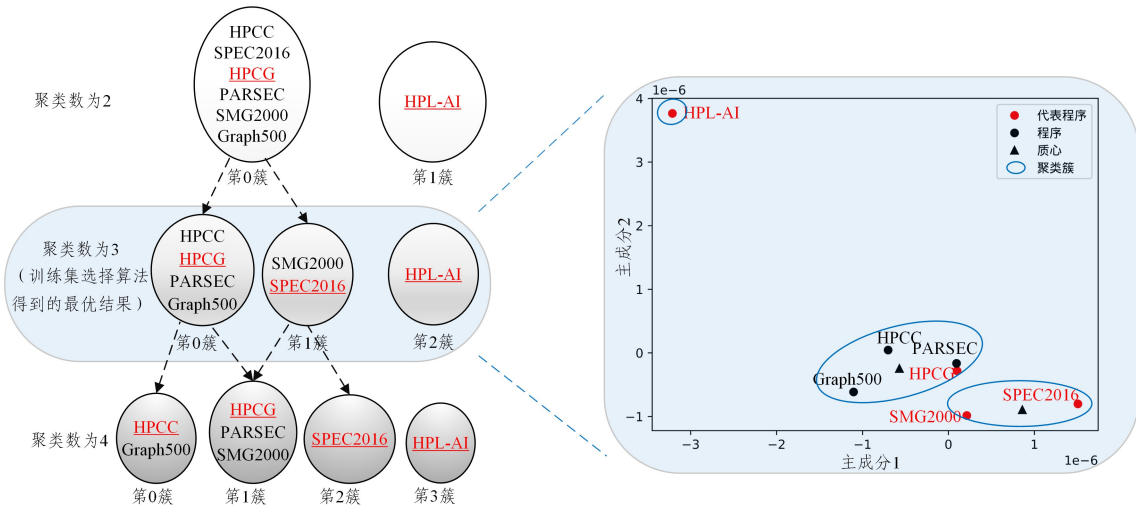
注：最优聚类数为 4，训练集为 {HPCC，HPCG，SMG2000，SPEC2016}。

(a)3 种聚类数下的分类及代表程序(标注下划线)选择结果

(b) 聚类数为 4 时的聚类结果及质心位置

图 6 x86 处理器平台上基于 K-means 聚类的训练集选择优化算法结果

Fig. 6 Results of optimization algorithm for training set selection based on K-means clustering on x86 platform



注：最优聚类数为 3，训练集为｛HPCG，SPEC2016，HPL-AI｝。

（a）3 种聚类数下的分类及代表程序（标注下划线）选择结果　　（b）聚类数为 3 时的聚类结果及质心位置

图 7　ARM 处理器平台上基于 *K*-means 聚类的训练集选择优化算法结果

Fig. 7　Results of optimization algorithm for training set selection based on *K*-means clustering on ARM platform

表 14　*k* 的不同取值对线性模型功耗建模开销的影响

Table 14　Effect of different *k* on consumption of power modeling using linear model

（s）

k	平台	
	功耗建模开销（x86 处理器平台）	功耗建模开销（ARM 处理器平台）
k=2	23953.62	7953.42
k=3	55225.74	20534.27
k=4	60706.39	37188.45

表 15　*k* 的不同取值对神经网络模型功耗建模开销的影响

Table 15　Effect of different *k* on consumption of power modeling using neural network model

（s）

k	平台	
	功耗建模开销（x86 处理器平台）	功耗建模开销（ARM 处理器平台）
k=2	24253.52	8456.26
k=3	55721.63	21031.95
k=4	61302.58	43192.85

表 16　*k* 的不同取值对线性模型功耗建模误差的影响

Table 16　Effect of different *k* on MRE of power modeling using linear model

（%）

k	平台	
	平均相对误差（MRE）（x86 处理器平台）	平均相对误差（MRE）（ARM 处理器平台）
k=2	4.838	16.498
k=3	4.646	11.190
k=4	4.644	11.190

表 17　*k* 的不同取值对神经网络模型功耗建模误差的影响

Table 17　Effect of different *k* on MRE of power modeling using neural network model

（%）

k	平台	
	平均相对误差（MRE）（x86 处理器平台）	平均相对误差（MRE）（ARM 处理器平台）
k=2	4.284	10.613
k=3	2.947	7.191
k=4	2.810	5.652

2）代表程序选择效果分析

为了验证训练集选择算法的代表程序选择效果，我们用随机选择法进行对比。随机选择法指从每一类中随机选择代表程序组成训练集的方式。

表 18 和表 19 列出了训练集选择算法和随机选择法建模精度对比结果。

表 18　x86 平台四聚类下不同代表性程序选择对功耗建模精度的影响

Table 18　Effect of different representative programs on power modeling accuracy under four clusters on x86 platform

（%）

选择的代表性程序	采用线性模型得到的 MRE	采用神经网络模型得到的 MRE
Graph500，HPCG，HPCC，SPEC2016	46.764	5.741
Graph500，HPL-AI，HPCC，SPEC2016	39.886	4.344
PARSEC，HPCG，HPCC，SPEC2016	28.385	8.455
PARSEC，HPL-AI，HPCC，SPEC2016	28.137	9.742
SMG2000，HPL-AI，HPCC，SPEC2016	4.567	2.762
SMG2000，HPCG，HPCC，SPEC2016（训练集选择算法选出的训练集）	4.644	2.810

表 19　ARM 处理器平台三聚类下不同代表性程序选择对功耗建模精度的影响

Table 19　Effect of different representative programs on power modeling accuracy under three clusters on ARM platform

（%）

选择的代表性程序	采用线性模型得到的 MRE	采用神经网络模型得到的 MRE
PARSEC，SMG2000，HPL-AI	25.914	9.752
HPCC，SMG2000，HPL-AI	17.974	7.823
Graph500，SMG2000，HPL-AI	17.346	10.646
HPCC，SPEC2016，HPL-AI	11.231	6.376
PARSEC，SPEC2016，HPL-AI	14.672	6.146
HPCG，SPEC2016，HPL-AI	14.161	6.060
Graph500，SPEC2016，HPL-AI	14.066	5.955
HPCG，SMG2000，HPL-AI（训练集选择算法选出的训练集）	11.190	5.652

从表中可以看出，相比随机选择法，在 x86 处理器平台

上，训练集选择算法平均相对误差平均减小24.882% 和3.399%（分别为线性模型和神经网络模型，下同）；在 ARM 处理器平台，训练集选择算法平均相对误差平均减小 5.291%和 1.885%。这验证了训练集选择算法的步骤 8－步骤 9 中代表程序选择方法的有效性。

5.3 进一步实验的结果

5.3.1 Baseline 算法和对照实验

本小节设计了 Baseline 算法和对照实验算法，其与训练集选择算法的不同主要体现在分类方法上。通过设计一个采用不同分类方法的 Baseline 算法，来进一步验证训练集选择算法中 K-means 分类方法部分的有效性。Baseline 算法步骤描述如下：

1）数据处理（同算法 1 步骤 1）；

2）采用基于特征差异阈值的分类法；

3）代表程序选择（同算法 1 步骤 7－步骤 9）。

算法中的基于特征差异阈值的分类法是将两两特征差异距离$d_{i,j}$小于阈值 d_{th}的程序分为一类；如果一个程序与所有程序特征差异距离$d_{i,j}$均大于阈值d_{th}，则将其单独分为一类。

而对照算法中，我们采用了模型训练中常用的训练集选择方法，即不进行数据集筛选，将收集到的所有数据集作为训练集，尽可能地扩大训练集规模。

表 12 和表 13 列出了训练集选择算法和 Baseline 算法的精度；表 20 和表 21 列出了 Baseline 算法在 x86 处理器平台和 ARM 处理器平台的最优聚类数下的聚类结果。在 x86 处理器平台上，相比 Baseline，对照算法采用线性模型时精度降低了 0.105%，采用神经网络模型时精度提升了1.935%，说明 Baseline 算法难以保证 CPU 功耗模型精度不降低。在 ARM 处理器平台上，相比 Baseline，对照算法采用线性模型时精度提升了 8.081%，采用神经网络模型时精度提升了 1.467%。两个处理器平台上的对照实验结果进一步表明本文提出的基于 K-means 聚类的训练集选择优化算法在保证功耗模型精度上是非常有效的。

表 20 x86 处理器平台上 Baseline 算法聚类结果

Table 20 Clustering results of Baseline algorithm on x86 platform

聚类结果		代表程序
簇编号	簇中所有程序	
第 1 簇	HPCC，PARSEC	PARSEC
第 2 簇	HPL-AI，HPCG，SPEC2016，SMG2000	SMG2000
第 3 簇	Graph500	Graph500

表 21 ARM 处理器平台上 Baseline 算法聚类结果

Table 21 Clustering results of Baseline algorithm on ARM platform

聚类结果		代表程序
簇编号	簇中所有程序	
第 1 簇	HPCC，PARSEC，SPEC2016，SMG2000	PARSEC
第 2 簇	Graph500，HPL-AI，HPCG，PARSEC	HPL-AI

5.3.2 权衡精度及训练开销的效果显著

为综合评估训练集选择算法在精度和开销两个方面的建模效果，本文引入了一种评估得分。该方法对各个实验组的误差率和功耗建模开销进行了统一的度量，随后根据精度和开销的相对重要性，赋予它们相应的权重。评估得分 score 计算步骤如下：

1）数据归一化

应用公式$x_i'=\frac{x_{\max}-x_i}{x_{\max}-x_{\min}}$，其中$x_i'$，$x_i$，$x_{\max}$，$x_{\min}$分别为同一平台下第 i 组 MRE 归一化后的值、最大值、最小值，以及不同实验组的原始数值，功耗建模开销归一化的值 y 也用同样的公式处理。Baseline 算法实验组训练集选择优化算法实验组，对照组分别为第 1，2，3 组，i 取值为 1，2，3。

2）设置权重精度-开销权重组合 $weight_1$-$weight_2$

3）计算不同模型下的评估得分

本文进行了 4 组实验，分别令精度-开销权重组合 $weigh_1$-$weight_2$为 50%-50%，60%-40%，70%-30%和 80%-20%，尽可能覆盖多种情况，以研究不同权重下的评估得分 score，具体如图 8－图 11 所示。

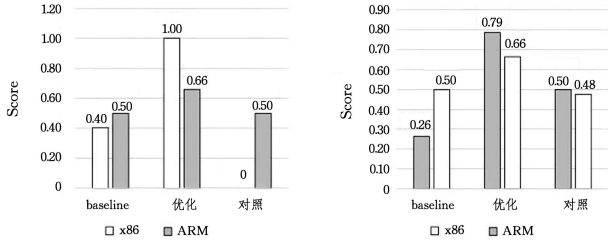
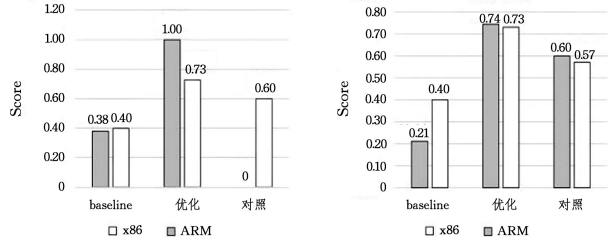


图 8 精度-开销权重组合 $weight_1$-$weight_2$为 50%-50%时的评估得分

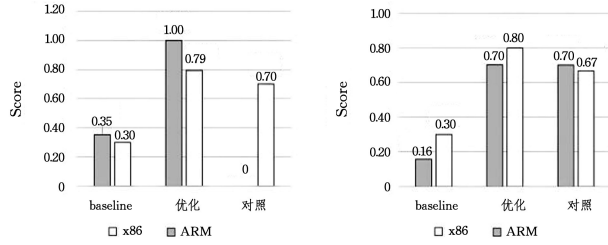
Fig. 8 Evaluation score when precision-cost weight combination $weight_1$-$weight_2$ is 50%-50%



(a)线性模型 (b)神经网络模型

图 9 精度-开销权重组合 $weight_1$-$weight_2$为 60%-40%时的评估得分

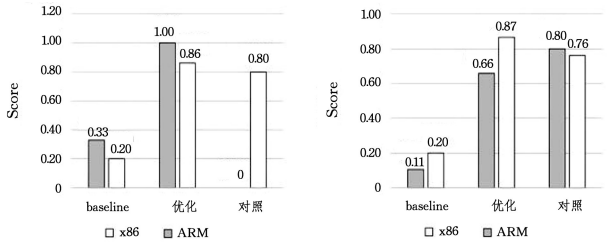
Fig. 9 Evaluation score when precision-cost weight combination $weight_1$-$weight_2$ is 60%-40%



(a)线性模型 (b)神经网络模型

图 10 精度-开销权重组合 $weight_1$-$weight_2$ 为 70%-30%时的评估得分

Fig. 10 Evaluation score when precision-cost weight combination $weight_1$-$weight_2$ is 70%-30%



(a)线性模型 (b)神经网络模型

图 11 精度-开销权重组合 $weight_1$-$weight_2$ 为 80%-20%时的评估得分

Fig. 11 Evaluation score when precision-cost weight combination $weight_1$-$weight_2$ is 80%-20%

5.3.3 与其他聚类算法的对比

为验证 K-means 聚类在训练集选择算法中的效果，本小节将算法中使用的聚类算法替换为凝聚层次聚类算法(Agglomerative Hierarchical Clustering，AHC)[21]，以对比两种聚类算法的聚类效果。AHC 算法是一种常用的采用自底向上聚类策略的层次聚类算法，它将数据集中每个样本看作一个初始聚类簇，迭代地找出距离最近的两个聚类簇并进行合并，直到达到预设的聚类簇个数。两个聚类簇 C_i，C_j 之间的距离通常为平均距离。

$$d_{\text{avg}}(C_i,C_j)=\frac{1}{|C_i||C_j|}\sum_{x\in C_i,z\in C_j}dist(x,z) \tag{6}$$

其中，$dist(x,z)$ 为 x 与 z 的欧氏距离。

实验结果表明，不同的聚类算法会对算法的优化效果产生较大影响。表 22 和表 23 中分别列出了在 x86 处理器平台和 ARM 处理器平台上使用层次聚类算法的聚类结果和代表程序选择结果。在 x86 处理器平台上，AHC 算法选出的代表程序为 HPCC+HPCG+PARSEC+SEPC2016，线性模型和神经网络模型的 MRE 分别为 5.41%和 2.74%；在 ARM 处理器平台上，层级聚类算法选出的代表程序为 HPCG+PARSEC+SMG2000，线性模型和神经网络模型的 MRE 分别为 25.914%和 9.752%。对比 5.2.1 节中使用 K-means 聚类的总体结果，在 x86 处理器平台上，使用 AHC 算法线性模型的 MRE 提升了 16.49%，神经网络模型的 MRE 降低了 2.67%；在 ARM 处理器平台上，线性模型和神经网络模型的 MRE 分别提升了 131.58%和 72.541%。结果表明，AHC 算法在 x86 处理器平台上精度提升效果与 K-means 聚类相近，但在 ARM 处理器平台上精度提升效果远远低于 K-means 聚类。这表明选用不同的聚类算法会对算法的优化效果产生较大影响，AHC 算法并不能很好地适应算法中的聚类需求。

表 22 x86 处理器平台 AHC 算法聚类结果和代表程序选择结果

Table 22 AHC algorithm clustering results and representative program selection results on x86 platform

聚类结果		代表程序
簇编号	簇中所有程序	
第 1 簇	HPCC，HPL-AI	HPCC
第 2 簇	HPCG，SMG2000	HPCG
第 3 簇	Graph500，PARSEC	PARSEC
第 4 簇	SPEC2016	SPEC2016

表 23 ARM 处理器平台 AHC 算法聚类结果和代表程序选择结果

Table 23 AHC algorithm clustering results and representative program selection results on ARM platform

聚类结果		代表程序
簇编号	簇中所有程序	
第 1 簇	HPCC，HPCG，Graph500，PARSEC，HPL-AI	HPL-AI
第 2 簇	PARSEC	PARSEC
第 3 簇	SMG2000	SMG2000

6 相关工作

对处理器进行功耗建模一直以来都不乏相关研究，近年来处理器功耗建模精度提升方面尤其受到关注。Xie 等[22]提出一种自动对每周期功耗进行建模的框架 APOLLO，并在微处理器上进行了集成和实现。他们的研究发现，在采用更低时间分辨率的情况下，模型精度还会进一步提升。Nikov 等[23]则提出一种基于事件的处理器功耗模型，能够实现在不支持 PMU 的平台上进行处理器功耗建模。该模型需要在两个平台上同时运行相同的 Benchmark，一个是支持 PMU 和功耗测量的平台；另一个是不支持 PMU 但支持功耗测量的平台。通过在第一类平台上的测量分析来建立功耗与 PMU 之间的对应关系，并根据两个平台上时间戳的同步对应关系对第一类平台的 PMU 和第二类平台上的功耗值做匹配，从而建立功耗模型。最终该方法获得了较高的功耗模型精度，平均误差小于 2%。

功耗建模的文章中很少涉及如何选择建模训练集的工作，例如表 22 中列举的工作，都只是在研究中介绍了建模所选用的训练集，没有给出选择这些训练集的方法。还有一些功耗建模方法在选择训练集时没有选择标准的测试程序，而是自己设计训练集，同样也没有给出这样做的原因，例如 Xie 等[22]在进行功耗建模时选择使用自己设计的 microbench 作为训练集。

表 24 功耗建模相关文献及其使用的 Benchmark

Table 24 Related literatures on power consumption modeling and the Benchmarks they used

来源	Benchmark			
IPDPS12[24]	caret	nnet		
IPDPS05[25]	SPEC	FT	NPB	SP
PACT02[26]	SPEC	Postgres		
IPDPS16[27]	eq3dyna			
ISLPED05[28]	SPEC			
DAC00[29]	MPEG2			
IGCCP13[30]	HPL			
SoutheastCon10[31]	NAS	BT	SP	
IJPP17[32]	SPEC	MiBench	SD-VBS	
ISLPED01[33]	SPEC	FP		

PCA 作为一种常见的统计学手段，已被大量应用于降低建立模型的开销，进一步还会采用分类方法选取训练集。Marbach 等[19]针对芯片设计中流片验证阶段花费时间长的问题，提出可以通过研究训练集的特点，减少部分训练集输入。首先收集这些训练集的部分运行时参数，如平均 cache 命中率、分支预测准确率等，然后使用 PCA 技术对收集到的数据进行降维分析，将每个程序看作一个由若干主成分组成的向量，并通过这些向量间的距离定义程序之间的相似度。

后续实验验证了相似度高的程序在多项验证指标上具有相似的特点。Eriksson 等[17]针对 QSAR 模型的建模中部分模型可靠度低的问题，发现训练集的选取对该模型的准确度影响较大。输入数据包括分子质量、摩尔折射率等，冗余且难以分析。他们首先使用 PCA 对输入进行降维，得到两个主成分，然后使用这两个主成分对输入数据进行分析。根据不同的选取原则，选择了 5 种不同的训练集，并对比了模型精度。结果表明，使用全部输入的 20%作为训练集就可以得到一个精度足够高的模型。

先前已有的工作缺乏有效的分类方法。Marbach 等[19]通过设立相似度的阈值，将两两程序之间距离低于阈值的若干程序分为一类，并在每类中选取代表程序，由这些代表程序组合得到的新训练集就可以在芯片验证中达到和原训练集相同的效果。本文的实验结果表明，在 CPU 功耗建模当中，相比使用阈值的分类法，使用聚类算法能够得到更高的功耗建模精度。

结束语 目前已有多种功耗模型被用于 CPU 功耗建模，但针对如何选取适合的模型训练集以避免精度下降的研究较少。本文针对功耗建模中训练集选择不当导致的 CPU 功耗建模精度不理想的问题，分析了不同训练集对功耗建模精度的影响，并提出了基于 K-means 聚类的训练集选择优化算法。该算法通过聚类方法选出了单聚类簇中最具代表性的程序，同时通过基于聚类的冗余代表程序迭代筛选方法尽可能去除了多聚类簇间冗余的训练程序，实现了在保证模型训练精度不降低的情况下最小化训练开销。最终在 x86 处理器和 ARM 处理器两类平台上验证了所提算法的有效性。

参 考 文 献

[1] CHEN J，ZHOU W H，DONG Y，et al. Analyzing time-dimension communication characterizations for representative scientific applications on super computer systems[J]. Frontiers of Computer Science，2019，13(6)：1228-1242.

[2] CHEN J，QI X，WU F，et al. More Bang for Your Buck：Boosting Performance with Capped Power Consumption[J]. Tsinghua Science and Technology，2021，26(3)：370-383.

[3] WANG Z，TANG Y，CHEN J，et al. Energy wall for exascale supercomputing[J]. Computing and Informatics，2016，35(4)：941-962.

[4] TIWARI A，LAURENZANO M A，CARRINGTON L，et al. Modeling Power and Energy Usage of HPC Kernels[C]// IEEE International Parallel & Distributed Processing Symposium Workshops & Phd Forum. IEEE，2012.

[5] ZHOU Z，ABAWAJY J H，LI F，et al. Fine-Grained Energy Consumption Model of Servers Based on Task Characteristics in Cloud Data Center[J]. IEEE Access，2018，6：27080-27090.

[6] VON KISTOWSKI J，GROHMANN J，SCHMITT N，et al. Predicting server power consumption from standard rating results[C]// International Conference on Performance Engineering. 2019.

[7] LIN W，WU G，WANG X，et al. An artificial neural network approach to power consumption model construction for servers in cloud data centers[J]. IEEE Transactions on Sustainable Computing，2019，5(3)：329-340.

[8] LI Y，HU H，WEN Y，et al. Learning-based power prediction for data centre operations via deep neural networks[C]// the 5th International Workshop. ACM，2016.

[9] ZHANG Y，DONG Y，CHEN J，et al. Pmc-based dynamic adaptive cpu and dram power modeling[C]// International Conference on Algorithm and Architectures for Parrallel Processing. 2020：92-111.

[10] SPEC. Standard performance evaluation corporation [EB/OL]. 2017. http://www. spec. org/cpu2017/.

[11] LUSZCZEK P R，BAILEY D H，DONGARRA J J，et al. S12-the hpc challenge(hpcc) benchmark suite[Z]. 2006.

[12] PARSEC. The princeton application repository for shared- memory computers[EB/OL]. https://parsec. oden. utexas. edu/.

[13] GRAPH500. Graph500[EB/OL]. https://graph500. org/.

[14] BRIAN C. The smg2000 benchmark code [EB/OL]. https://asc. llnl. gov/computing _ resources/purple/archive/benchmarks/smg/.

[15] HPCG. High performance conjugate gradients [EB/OL]. https://www. hpcg-benchmark. org/.

[16] JACK D. Hpl-ai mixed-precision benchmark[EB/OL]. https://icl. bitbucket. io/hpl-ai/.

[17] ERIKSSON L，JOHANSSON E，WOLD S. On the selection of the training set in environmental qsar analysis when compounds are clustered[J]. Journal of Chemometrics，2000，14：599-616.

[18] GUTIERREZ M，TAMIR D，QASEM A. Evaluating Neural Network Methods for PMC-based CPU Power Prediction[C]// International Multi-Conference on Computing in the Global Information Technology. 2015.

[19] MARBACH M，ONDUSKO R，RAMACHANDRAN R P，et al. Neural network classifiers and Principal Component Analysis for blind signal to noise ratio estimation of speech signals[C]// 2009 IEEE International Symposium on Circuits and Systems (ISCAS). 2009：97-100.

[20] CALIŃ SKI T，HARABASZ J. A dendrite method for cluster analysis[J]. Communications in Statistics-theory and Methods，1974，3：1-27.

[21] ZHANG F L，ZHOU H C，ZHANG J J，et al. Protocol classification algorithm based on improved cohesive hierarchical clustering [J]. Computer Engineering and Science，2017，39(4)：796-803.

[22] XIE Z Y，XU X Q，WALKER M，et al. Apollo：An automated power modeling framework for run-time power introspection in high-volume commercial microprocessors[C]// 54th Annual IEEE/ACM International Symposium on Microarchitecture. 2021：1-14.

[23] NIKOV K，MARTINEZ M，CHAMSKI Z，et al. Robust and accurate fine-grain power models for embedded systems with no on-chip pmu[J]. arXiv. 2106. 00565，2021.

[24] TIWARI A，LAURENZANO M A，CARRINGTON L，et al. Modeling Power and Energy Usage of HPC Kernels[C]// IEEE International Parallel & Distributed Processing Symposium Workshops & Phd Forum. IEEE，2012.

[25] FENG X，GE R，CAMERON K W. Power and Energy Profiling

of Scientific Applications on Distributed Systems[C]// 19th IEEE International Parallel and Distributed Processing Symposium. 2005.

[26] CONTRERAS G, MARTONOSI M. Power prediction for intel xscale processors using performance monitoring unit events [C]// Proceedings of the 2005 International Symposium on Low Power Electronics and Design. 2005: 221-226.

[27] WU X, TAYLOR V. Utilizing Hardware Performance Counters to Model and Optimize the Energy and Performance of Large Scale Scientific Applications on Power-Aware Supercomputers [C] // IEEE International Parallel & Distributed Processing Symposium Workshops. IEEE, 2016.

[28] BIRCHER W L, VALLURI M, LAW J, et al. Runtime identification ofmicroprocessor energy saving opportunities[C]// International Symposium on Low Power Electronics & Design. IEEE, 2005.

[29] QU G, KAWABE N, USAMI K, et al. Function-level power estimation methodology for microprocessors[C]// Proceedings of the 37th Annual Design Automation Conference. 2000: 810-813.

[30] LAROS J H, POKORNY P, DEBONIS D, et al. Powerinsight—a commodity power measurement capability[C]// 2013 International Green Computing Conference Proceedings(IGCC). 2013: 1-6.

[31] BEDARD D, LIM M Y, FOWLER R, et al. Powermon: Fine-grained and integrated power monitoring for commodity computer systems[C]// IEEE Southeastcon. 2010: 479-484.

[32] ZHENG X, JOHN L K, GERSTLAUER A, et al. Lacross: Learning-based analytical cross-platform performance and power prediction[J]. International Journal of Parallel Programming, 2017, 45(6): 1488-1514.

[33] JOSEPH R, MARTONOSI M. Run-time power estimation in high performance microprocessors[C]// International Symposium on Low Power Electronics Design. 2001.

LI Zekai, born in 2003, undergraduate. His main research interests include high performance computing and so on.

CHEN Juan, born in 1980, Ph.D, professor. Her main research interests include high-performance computing, low-power compiler and power management, etc.

(责任编辑:杨雪敏)