Journal of Computer Applications
计算机应用, 2020, 40(10): 3081 - 3087
ISSN 1001-9081
CODEN JYIIDU
2020-10-10
http://www.joca.cn

文章编号:1001-9081(2020)10-3081-07
DOI:10.11772/j.issn.1001-9081.2020010118

基于机器学习的异构感知多核调度方法

安　鑫[1,2*],康　安[1,2],夏近伟[1,2],李建华[1,2],陈　田[1,2],任福继[1,2]
(1. 合肥工业大学 计算机与信息学院,合肥 230601;
2. 情感计算与先进智能机器安徽省重点实验室(合肥工业大学),合肥 230601)
(*通信作者电子邮箱 xin.an@hfut.edu.cn)

摘　要:异构多核处理器已成为现代嵌入式系统的主流解决方案,而好的在线映射或调度方法对其充分发挥高性能和低功耗的优势起着至关重要的作用。针对异构多核处理系统上的应用程序动态映射和调度问题,提出一种基于机器学习、能快速准确评估程序性能和程序行为阶段变化的检测技术来有效确定重映射时机从而最大化系统性能的映射和调度解决方案。该方案一方面通过合理选择处理核和程序运行时的静态和动态特征来有效感知异构处理所带来的计算能力和工作负载运行行为的差异,从而能够构建更加准确的预测模型;另一方面通过引入阶段检测来尽可能减少在线映射计算的次数,从而能够提供更加高效的调度方案。最后,在SPLASH-2数据集上验证了所提出调度方案的有效性。实验结果表明,与Linux默认的完全公平调度(CFS)方法相比,所提出的方法在系统计算性能方面提高了52%,在CPU资源利用率上提高了9.4%。这表明所提方法在系统计算性能和CPU资源利用率方面具备优良的性能,可以有效提升异构多核系统的应用动态映射和调度效果。

关键词:异构多核处理系统;动态映射和调度;机器学习;性能预测;阶段检测
中图分类号:TP302.7　　**文献标志码:**A

Heterogeneous sensing multi-core scheduling method based on machine learning

AN Xin[1,2*], KANG An[1,2], XIA Jinwei[1,2], LI Jianhua[1,2], CHEN Tian[1,2], REN Fuji[1,2]
(1. *School of Computer Science and Information Engineering, Hefei University of Technology, Hefei Anhui* 230601, *China*;
2. *Anhui Key Laboratory of Affective Computing and Advanced Intelligent Machine (Hefei University of Technology), Hefei Anhui* 230601, *China*)

Abstract: Heterogeneous multi-core processor is the mainstream solution for modern embedded systems now. Good online mapping or scheduling approaches play important roles in improving their advantages of high performance and low power consumption. To deal with the problem of dynamic mapping and scheduling of applications on heterogeneous multi-core processing systems, a dynamic mapping and scheduling solution was proposed to effectively determine remapping time in order to maximize the system performance by using the machine learning based detection technology of quickly and accurately evaluating program performance and program behavior phase change. In this solution, by carefully selecting the static and dynamic features of processing cores and programs to running to effectively detect the difference in computing power and workload running behaviors brought by heterogeneous processing, a more accurate prediction model was built. At the same time, by introducing phase detection technology, the number of online mapping computations was reduced as much as possible, so as to provide more efficient scheduling scheme. Finally, the effectiveness of the proposed scheduling scheme was verified on the SPLASH-2 dataset. Experimental results showed that, compared to the Completely Fair Scheduler (CFS) of Linux, the proposed method achieved about 52% computing performance gains and 9.4% improvement on CPU resource utilization rate. It shows that the proposed method has excellent performance in system computing performance and processor resource utilization, and can effectively improve the dynamic mapping and scheduling effect of applications of heterogeneous multi-core systems.

Key words: heterogeneous multi-core system; dynamic mapping and scheduling; machine learning; performance prediction; phase detection

收稿日期:2020-02-15;**修回日期:**2020-05-11;**录用日期:**2020-05-12。
基金项目:国家自然科学基金资助项目(U1613217);中央高校基本科研业务费专项资金资助项目(JZ2020YYPY0092)。

作者简介:安鑫(1987—),男,山东潍坊人,副教授,博士,CCF会员,主要研究方向:嵌入式系统、机器学习;　康安(1995—),男,河北邢台人,硕士研究生,主要研究方向:嵌入式软件;　夏近伟(1994—),男,安徽合肥人,硕士研究生,主要研究方向:嵌入式软件;　李建华(1985—),男,安徽肥西人,副教授,博士,主要研究方向:计算机体系结构、非易失性存储器;　陈田(1974—),女,安徽合肥人,副教授,博士,CCF高级会员,主要研究方向:超大规模集成电路/系统芯片低功耗测试、可穿戴计算;　任福继(1959—),男,四川南充人,教授,博士,主要研究方向:信号与信息处理、计算机视觉。

0 引言

自计算机诞生以来，现代计算机处理系统的发展开始呈现多元化的趋势，规模上实现了从小型的手持设备、笔记本电脑到集群服务器、数据中心的全覆盖。另一方面其应用领域也日趋多样化和复杂化，涵盖了多媒体、信息的加解密、网络服务和云同步、视觉和语音识别、自然语言处理等，这些不同类型的应用程序呈现出不同的程序特性和资源需求[1]。单核处理器受功耗、面积和成本等因素限制发展已经接近极限，为了应对当前应用的复杂性和对高性能、低功耗的需求，包含多个不同类型处理核的异构多核(heterogeneous multi-core)平台[2]已经成为主流的解决方案。

异构多核平台不同于传统的同构多核中处理核比较单一的情况，它具有不同类型的处理核，每种类型的处理核又具有不同的微架构，如：有些处理核是顺序流水线设计，有些是乱序流水线设计；有的处理核具有较大的缓存架构，有的处理核具有较小的缓存架构等。根据指令集架构和微架构的不同可以划分为单指令集异构多核平台和多指令集异构多核平台，其中：单指令集异构多核平台中的处理核具有相同的指令集架构，但具体的微架构实现不同；而多指令集异构多核平台中的处理核具有不同的指令集架构。单指令集异构多核平台主要将高性能高功耗的处理核和低性能低功耗的处理核进行集成，满足多样化的应用程序需求，由于处理核都为同一套指令集架构，因此无需额外的编程环境即可实现协同工作，程序执行点可以任意地进行迁移，如Arm的Big. Little架构将两种不同微架构的CPU(Cortex-A7和Cortex-A5)进行集成，满足了移动领域下用户对高性能和低功耗的需求[3]。多指令集异构多核平台主要以CPU-GPU为代表，CPU-GPU架构利用专门化的图形处理单元对可以高并发处理的程序片段提供加速支持，但该架构下不同类型的处理核之间的协同工作需要编程环境支持，不允许随意地迁移程序的执行点到不同类型的处理核上。本文主要针对单指令集架构的异构多核平台调度方法进行研究，因此下文中没有额外说明的异构多核平台皆为单指令集架构。

异构多核平台通常可以并发执行多个应用程序，并且每个应用程序的行为和资源需求往往也会随时间发生变化。为了适应这种动态执行场景并充分发挥出异构多核处理器的优势，需要解决的很重要的一个问题是如何根据系统需求对应用程序进行动态的资源映射和调度。一个好的异构调度策略需要能够感知异构处理器系统各个处理核之间的异构性和线程行为的不同特性，在对不同映射方案进行高效评估的基础上，动态地进行线程到处理核的映射。这种决定某个线程应该映射到哪个处理核的问题类似于机器学习技术已成功得到应用的推荐系统要解决的推荐问题(比如一些视频点播网站为用户推荐他们可能会喜欢的视频[4-5])。

以神经网络为代表的机器学习(Machine Learning, ML)和深度学习(Deep Learning, DL)技术目前已经在推荐系统、计算机视觉和自然语言处理等领域取得了巨大的成功。尽管机器学习类似于黑盒模型，所学到的输入数据和输出数据的关系通常难以被人们理解，但在输入数据足够多时，机器学习方法常常能提供出色的预测精度。目前已经不少工作尝试将ML引入到异构多核系统调度问题中并取得了不错的效果[6]。这方面的工作主要有两类：一类工作是仅针对如何对调度方案的性能或功耗等指标构建准确高效的预测模型[7-8]，这类工作需要结合其他在线搜索方法构成完整的动态映射和调度方案。另外一类是在第一类的基础上如何设计完整的线程或任务的动态映射和调度方法[9-10]。这类方法除了要构建准确高效的性能预测模型之外，还需要解决运行时随着线程行为等发生变化、何时对线程进行重新映射从而达到最优化运行的问题。目前大部分这类解决方案均基于固定的调度周期(如1 ms)进行映射选择计算，而应用程序的运行行为大多呈现出阶段变化的特征[11]，每个阶段的运行行为相对稳定，而且往往持续较长时间，因而，过于频繁的在线映射计算并不一定会改善当前的映射效果，反而会在一定程度上抵消整体映射和调度方案的效果。

本文提出了一种基于机器学习、能快速准确评估程序性能和程序行为阶段变化的检测技术来有效确定重映射时机，从而最大化系统性能的映射和调度解决方案。该方法通过合理选择处理核和线程运行时的静态和动态特征来有效感知异构处理所带来的处理核计算能力和工作负载运行行为的差异，从而构建更加准确的预测模型；通过引入阶段检测技术来尽可能减少在线映射计算的次数，从而提供更加高效的调度方案；通过与Linux系统使用的经典完全公平调度(Completely Fair Scheduler, CFS)方法进行对比，实验结果表明使用本文调度方法的计算性能能提高52%左右。

1 相关工作

为了充分利用异构多核系统的异构特性和优势来满足应用程序的多样化需求，如何设计和实现应用程序的合理映射和调度方案已经成为研究热点之一。由于调度问题是一个众所周知的NP-Hard问题，通过手动设计算法或者寻找满足所有给定约束条件的最优解非常困难并且十分耗时，因此，目前研究者们大多基于启发式算法[12-14]，在可接受的时间内寻找满足条件的一个近似最优解。文献[1]中作者从优化目标、分析模型、调度决策和算法评估四个方面对异构多核调度所面临的问题与挑战进行了总结和讨论，并对各种调度技术进行了概括。由于本文的关注点在于应用机器学习技术来解决异构多核系统中的任务映射及调度问题的工作，因此接下来主要讨论在进行应用程序映射核调度过程中，使用机器学习模型来优化解决方案的相关研究。

文献[12]中作者针对片上网络(Network-on-Chip, NoC)异构多核平台下的任务映射问题提出了一种基于人工神经网络(Artificial Neural Network, ANN)的解决方案。该方案首先探索任务节点映射到不同位置的处理核的所有可能，之后将对应的映射方式进行编码并交给ANN来预测，通过统计不同映射方案所能达到的最少执行时间来帮助设计者找到最优的任务映射方式。文献[13]则针对动态多核平台下的映射问题提出了一个基于机器学习方法的解决框架，分别使用K近邻和线性回归算法构建了一个线程数预测模型和一个处理核数量预测模型。在开始映射前，框架首先通过K近邻算法来预测应用程序需要划分成多少个线程，之后再利用线性回归算法来预测每个线程所需要的最佳处理内核数，从而找到最佳的映射方式。

在文献[14]中作者针对由CPU和GPU两类处理核组成的异构多核平台，首先对OpenCL程序的静态和动态特征进行了特征分析，之后选择了16种特征训练了一个可以预测应用程序在映射时需要映射的最佳处理核类型的支持向量机(Support Vector Machine, SVM)模型。在他们后续的工作中，对预测模型进行了进一步的优化，将更多类型的处理核因素纳入了模型的训练过程中，使用了多个SVM来训练预测模型，其中每种SVM对应一种类型的处理核，并结合优化目标

来确定程序在运行时所需要映射的处理核类型[15]。

文献[16]中详细评估分析了异构多核系统上的动态电压频率缩放行为,并利用凸优化方法来寻找性能与功率之间的关系,最终提出了一种可以预估线程在处理核上运行所产生功耗的预测模型,并基于该模型提出了一种可以优化系统运行时功耗的解决方案。文献[17]中将机器学习技术引入了单应用程序系统的能源优化探索领域。作者使用了5种机器学习模型来探索不同的配置选项,包括套接字分配、超线程使用和处理器DVFS(Dynamic Voltage and Frequency Scaling)级别等。该文献将调度计算的时间间隔设为5 s,并对调度间隔的大小设置进行了实验探索,实验结果表明越小的调度间隔越有利于把握程序行为变化,但会导致更多的调度计算开销。文献[18]中将异构平台的资源分配问题定义成机器学习问题,提出了一个基于ANN的动态资源管理器。该资源管理器在运行时监视每个应用程序的执行情况,利用细粒度的资源动态信息(如L2缓存空间、片外带宽、功率预算、在L1缓存中读/写命中以及读丢失/写丢失的数量等)预测每个应用程序在不同资源分配方案下的性能表现,从而动态调整资源分配策略,实现对缓存、内存和电压频率等资源的管理。文献[19]中同样利用ANN技术构建了一个调度模型来帮助优化系统最大的吞吐量。作者提出利用程序的行为特性和处理核微架构信息来为程序寻找最合适的处理核的思想,针对大核和小核两种类型的核心分别构建了两个ANN模型,利用程序的指令特征(如浮点运算指令、逻辑跳转指令等)和核心微架构参数(如各级缓存的命中丢失率)来训练ANN模型。作者设定两种ANN模型每隔1 ms运行一次,预测所有线程在大核(小核)的性能结果,并根据预测结果找出使整个异构平台可以达到最高IPC(Instructions Per Cycle)的调度方案。可以看出这些方法往往将调度间隔设成了固定的大小,没有充分考虑程序运行行为的阶段性变化,过于频繁的调度计算无疑会影响整体系统运行效率。

相对于其他方法,本文在权衡机器学习模型预测准确性、复杂度和在线预测模型开销的基础上,选择了具有一个隐藏层的ANN作为异构多核系统的性能预测模型。此外为了尽可能降低在线计算的次数,本文引入了阶段检测[18]的思想,通过感知程序线程行为是否发生足够明显的变化来决定是否进行重调度计算,从而尽可能地降低在线调度计算的额外开销,进一步提升系统的性能。

2　基于机器学习的异构多核系统调度方法

本章介绍本文针对异构多核系统所提出的动态映射和调度方法,对于给定的若干个独立线程,调度目标是通过对线程-处理核的动态映射实现异构多核系统的性能最大化。在本文中,计算性能通过每个时钟周期所运行的指令数即IPC值来衡量,并假设每个核上同一时刻只能运行一个线程[19]。图1给出了该方法的整体框架,由图可以看出整个异构多核调度方法的输入是一组处于就绪状态的待调度多线程队列,输出是线程到处理核的映射方案,主要由检测应用程序行为阶段变化的阶段检测器、基于ANN的系统性能预测器和映射方案计算部件三部分构成。具体工作分三个阶段:

1)在每个系统调度周期或者时间片收集异构多核平台上所运行的各个处理核和线程的运行信息(包括IPC值等相关信息),并将IPC值传递给阶段检测器。

2)阶段检测器通过对比每个处理核在当前时间片和上一时间片的IPC值来确定其所运行的线程是否发生了行为变化。如果行为变化足够明显,那么调度方法就进入重映射计算阶段来进行新的映射计算,来找到可以使系统性能最大化的新映射方案。

3)在重映射阶段,首先性能预测器基于采集到的线程当前运行信息来对其在不同类型核上的运行性能(IPC值)进行预测,然后映射计算部件根据这些预测值来决定当前时刻最优的映射方案即线程与处理核的最优映射关系。

此外,图1中的映射器用于具体部署映射计算部件得到的映射方案。

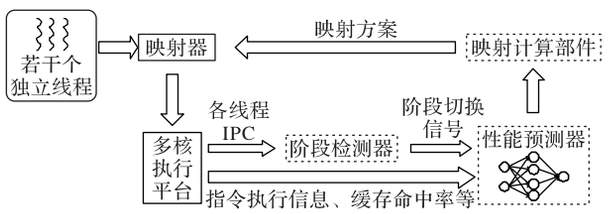

图1　整体框架

Fig. 1　Overall framework

2.1　阶段检测

一个程序在其生命周期中可以执行百亿条指令,在执行过程中其行为往往会发生阶段性的变化并且许多行为会重复出现[11]。程序在不同阶段一般具有不同的行为特征,表现为在不同阶段执行指令的类型和数量不同,cache缺失率、指令级并行度和系统资源利用率不同等[20]。

图2展示了运行SPEC2006基准程序测试集的gcc程序时其IPC值随时间片变化的情况。图2中的横坐标表示时间片的编号,每个时间片对应的纵坐标表示gcc程序在该时间片的IPC平均值。IPC值相对稳定的时间片区间可以看作一个程序稳定运行的阶段,从图2可以看出程序在运行时其IPC值会随着时间发生阶段性的变化,这种阶段性变化会在相邻两个时间片的IPC发生明显变化时出现。此外,还可以看到每个阶段横跨的时间有长有短,有的阶段维持的时间跨度甚至多达成百上千个时间片。由于在每个阶段程序行为相对稳定,因此就可以只在阶段发生切换时对现有的系统映射进行调整计算并寻找下一个阶段的最优映射方案,从而大大减少映射计算的次数,有效地降低在线计算的时间开销。

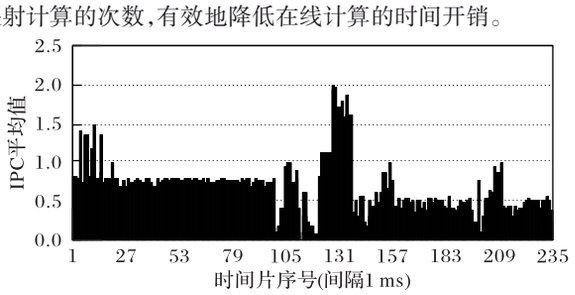

图2　gcc程序的IPC变化情况

Fig. 2　IPC change of gcc program

由于阶段切换时相邻两个时间的IPC变化较为明显,因而通过检测IPC变化幅度就可以检测阶段是否发生切换。为此本文通过在每个处理核上设置一个阶段检测器来完成阶段检测,工作原理是比较在相邻两个时间片所采集到的处理核上所运行线程的IPC值的变化幅度与所设阈值的大小,如果变化幅度高于所设阈值则认为阶段发生切换。IPC波动幅度δ_{ipc}的计算公式为:,

$$\delta_{ipc}=\frac{\left|IPC_j-IPC_{j-1}\right|}{IPC_j} \tag{1}$$

其中：IPC_j为当前时间片IPC值的大小；IPC_{j-1}是上一时间片IPC值的大小。

可以看出，能否找到发生阶段切换的最佳阈值，对于精确捕捉程序运行行为发生变化的时机至关重要。阈值的最优值往往需要根据具体的异构多核平台配置以及对运行的应用程序的实际执行情况进行探索来获得，文献[21]对不同阈值设置方法的检测效果进行了详细的探索和讨论，并给出了参考阈值，故本文参考该文献的设置方法，为检测器设置了大小为50%的全局阈值$\delta_{\max}$。类似地，在时间片大小的设置上本文也参考了其他相关方法[22]，将时间片设置为1 ms。

2.2 构建ANN预测模型

为了更加直观地阐述本文方法，假定异构处理器平台由两类处理核（即大核和小核）构成，由更多类型的处理核构成的异构平台可以相似地进行处理。本文选择采用ANN来为两类处理核分别构建性能预测器，预测器的输入是一组包含了线程运行信息和微架构参数信息的特征值集合，输出是当前线程在两类处理核上运行时能达到的IPC值。

2.2.1 模型参数选择

一个预测特定处理核上线程IPC值的机器学习模型的准确性很大程度上取决于输入参数是否能够准确捕捉到程序的行为特性（如浮点计算数量、访存行为、分支行为等）以及程序运行时所使用的处理核上各种资源的多少（如CPU资源、Cache资源等）。本文对各个输入参数利用皮尔逊相关系数（Pearson Correlation Coefficient，PCC）进行相关性分析[23]，利用相关系数来寻找参数与性能指标IPC是否有相关性，从而建立性能预测模型。在经过相关性分析后，本文从18种参数中选取了和IPC值关系较密切的10个指标来构建ANN性能预测器。这10个指标包括4种与处理核心微架构相关的参数，即L1-D、L1-I、L2和L3 Cache命中缺失率，以及6种与程序行为变化相关的参数，即浮点加法、减法、乘法、除法、跳转指令和读写内存指令。

2.2.2 ANN预测模型

除了输入参数本身以外，模型复杂度的高低和运行模型所产生的硬件开销等也是评估模型好坏的重要指标。由于ANN属于计算密集型算法，随着网络层数的增多，网络结构会越复杂，进而带来更多的硬件和在线预测开销等问题。因此，在综合权衡模型时间开销和精确度以后，本文最终使用了三层的神经网络来构建ANN性能预测器。表1给出了本文最终确定使用的网络模型的各项参数，接下来分别介绍这些参数的选择依据和考虑。

表1 人工神经网络参数设置

Tab. 1 Parameter setting of artificial neural network

参数	设置
输入节点个数	10
隐藏层节点个数	25
输出层节点个数	1
激活函数	ReLU函数
初始化学习率	0.000 1
损失函数	均方误差
训练方式	随机梯度下降（SGD）

ANN的输入层由10个神经元节点组成，输入层的10个节点分别负责接收10个输入参数，在计算完毕后，输入节点会将计算结果通过激活函数传递到隐藏层。理论上隐藏层节点的个数越多，ANN性能预测器在训练数据集上的预测效果就越好，但隐藏层节点个数超过一定数量会导致过拟合现象的出现。本文参考文献[24]中所提出的经验公式来确立隐藏层节点的个数，即采用经验公式$2n+m$，其中n为输入节点的数量，m为0~10的正整数。在对m的数值进一步进行实验分析后，选择使ANN性能预测器在训练集上获得最好效果的m值（即5），因此最终本文选择的隐藏层节点个数为25。隐藏层的每个神经元节点会接收所有的输入节点结果，在对其完成计算后通过激活函数传递到输出层。输出层利用所有隐藏神经元节点的计算结果得到最后的输出值（IPC）。

在激活函数的使用上本文选择了ReLU（Rectified Linear Unit）函数，该函数不含任何复杂的运算（如指数级运算），只拥有很小的计算量，可以最大限度地降低异构多核调度中产生的在线预测时间开销。同时，ReLU函数在训练过程中也可以有效避免梯度饱和问题，防止训练失败的情况出现。详细的ANN连接结构如图3所示。

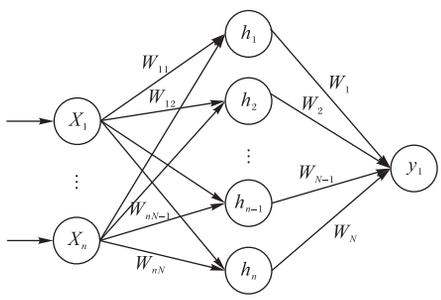

图3 人工神经网络模型

Fig. 3 Artificial neural network model

ANN性能预测器属于机器学习中的回归模型，因此本文使用回归模型常用的均方误差（Mean Squared Error，MSE）损失函数来对ANN性能预测器进行训练。使用MSE函数训练ANN性能预测器时，整个ANN的梯度会随MSE值的增大而增大，而MSE值趋于0时网络的梯度则会减小，因此采用固定的学习率即可保证整个网络可以有效地收敛，并在训练结束时取得良好的预测效果，所以本文将学习率设置为固定的0.001。

2.2.3 数据集获取和预处理

本文使用Sniper工具来获取用于训练大核和小核ANN性能预测器的两个数据集。为此，本文分别建立了两个独立的处理核平台，其中训练大核性能预测器的处理核平台只拥有一个大核，另一个用于训练小核性能预测器的处理核平台只拥有一个小核。通过在两个平台上完整执行基准测试集SPLASH-2中所有应用，并收集每个时间片的相关输入参数和输出标签等数据来获得原始数据集。

由于收集到的原始数据中各输入特征的单位或数量级不一致且相差较大（如每个时间片程序执行指令条数多达成百上千条，而cache命中缺失率常常远低于1），因此直接用原始数据对程序的IPC值进行预测的话，数量级较大的各种指令特征会严重影响预测效果。此外，程序每个时间片在大核上执行的各类型指令的条数与小核上执行的指令条数有时也会有1~2个数量级的差别，所以在大（小）核上收集的输入参数难以直接用来预测线程在小（大）核上的IPC值。因此本文对收集到的原始数据作了以下标准化处理：

1）针对6种表示程序行为的指令特征，本文用不同类型的指令条数占总指令条数的比率来替代原来的指令数量。

2）针对全部10种参数，本文对其进行了Z-Score归一化处

理以使数据集更加符合正态分布。具体计算公式如下:

$$\hat{x}_i = \frac{x_i - \mu_\beta}{\sqrt{\sigma_\beta^2 + \varepsilon}} \tag{2}$$

其中:μ_β和σ_β^2分别是训练集的均值和方差;ε是一个用来避免σ_β^2为0的常量,本次实验中将ε值设置为0.001。

2.2.4　预测器评估

在利用Sniper和基准测试集SPLASH-2得到所需要的原始数据集后,本文利用该数据集来对ANN性能预测器进行训练和评估。首先,将经过预处理后的原始数据集按照机器学习模型训练和评估常用的划分方法将数据集以7:3的比例进行划分,其中的70%用来训练模型,剩下的30%作为测试集来评估ANN性能预测器在未知数据上的预测效果。

图4显示了本文训练的两个ANN性能预测器在SPLASH-2基准程序测试集上的表现情况,MSE值越低表示ANN性能预测器的效果越好。从图4可以看出:小核性能预测器和大核性能预测器在barnes上表现最好,MSE值分别为0.010 8和0.013 7;而在volrend上两者表现最差,分别为0.750 6和0.936 4。总体上,在8个基准测试程序上两者分别达到了0.155 4和0.228 4的平均MSE值,均具有较好的预测效果。

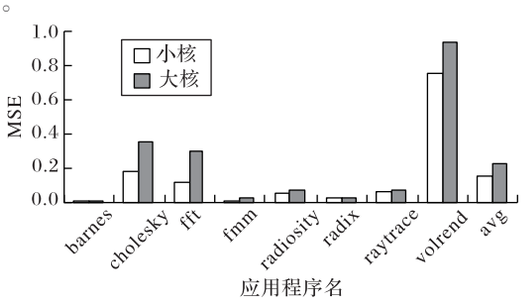

图4　ANN预测器的均方误差

Fig. 4　MSE of ANN predictor

2.2.5　映射方案计算部件

由于ANN性能预测器的输出结果是IPC值,并不是一个完整的映射方案,因此本文设计了一个映射方案计算部件来找出最优的映射方案。该部件的主要功能是利用ANN性能预测器给出各个线程在大(小)处理核上的IPC_{big}(IPC_{small})来对所有可行的映射方案进行评估,找出使所有线程总IPC_{sum}的最大的映射方案。

假设异构多核平台需要映射的线程数量为M,其中N个需要映射到大核上,另外$M-N$个线程映射到小核上,则存在的映射方案数量一共为C_M^N。映射方案计算部件每次选择N个线程的IPC_{big}和另外$M-N$个线程的IPC_{small}相加得到本次映射的IPC_{sum},在得到C_M^N个IPC_{sum}后,完成对所有可行映射方案的探索,并将IPC_{sum}值最大的那种映射方案输出给映射器。映射器在得到映射计算部件给出的映射方案后完成线程到处理核的部署。

3　实验评估

3.1　实验设置

本文选择采用业界常用的Sniper[24]作为数据采集和实验评估的工具,Sniper是一个用于处理器架构设计验证的模拟器,既可以用来模拟同构多核架构系统,也可以模拟异构多核架构系统的运行。该模拟器主要支持x86指令集的处理核,目前最新版本也加入了对RISC-V指令集的支持。为了实现良好的仿真效果和更快的仿真速度,Sniper选择了基于时间间隔模拟的方式来获得处理核运行时产生的各项指标并对设计方案进行评估。

由于本文主要针对单指令集架构下的异构多核平台,即同一平台下的多个处理核具有相同的指令集架构但各自有不同的微架构(如处理器主频、流水线深度、缓存系统差异等)的处理核,因此在本次实验中,本文基于Sniper搭建了一个由大核和小核两种类型的处理核组成的异构多核平台,其中大核指微架构的实现更加复杂,性能更高的处理核,而小核指的是微架构的实现相对简单,功耗较低的处理核。该平台是一个典型的四核异构非对称多核处理器,由1个大核和3个小核组成,详细配置如表2所示。1大3小的四核架构是移动平台领域常见的处理器配置方式:1个主打高性能的大核能在可接受面积和功耗的前提下提供最高的单核性能,而3个小核则可以在做到在更小的面积和功耗下提供一定的多核性能,从而在多个常见场景下拥有性能和功耗的优势。此外,由于cache大小、时钟频率等参数既影响程序的性能执行结果又影响了资源争用和线程同步等情况,为了精确验证程序在不同核心上执行的性能差异原因,本文采用了相同的时钟频率和三级cache缓存架构,其中L1 Cache为256 KB,L2 Cache为512 KB,共享的L3 Cache大小设置为8 MB。大核和小核的指令集架构均基于Intel Nehalem x86架构,且处理核的主频均为2.66 GHz。在核心的微架构配置上,本文为大核和小核均设置了大小为4的发射宽度,但是大核拥有128的指令窗口大小和长度为48的读取队列,与之对应的小核的指令窗口大小和队列长度分别为16和6。

表2　异构多核处理平台配置

Tab. 2　Heterogenous multi-core processor configurations

处理核	指令集架构	流水线类型	流水线深度	发射宽度	指令窗口	读取队列	时钟频率/GHz	L1 Cache/KB	L2 Cache/KB	L3 Cache/MB	分支预测/KB
核0	x86	乱序	15~24	4	128	48	2.66	256	512	8	2
核1	x86	顺序	8~12	4	16	6	2.66	256	512	8	2
核2	x86	顺序	8~12	4	16	6	2.66	256	512	8	2
核3	x86	顺序	8~12	4	16	6	2.66	256	512	8	2

本文使用SPLASH-2基准测试集作为实验用的程序集合。SPLASH-2基准测试集由一系列多线程程序组成,多应用于高性能计算和图形多媒体程序的测试。同时,Sniper集成了SPLASH-2基准测试集的已编译版本,可以对其下所有应用程序在实际硬件平台上的运行情况进行准确模拟,因此利用Sniper和SPLASH-2基准测试集可以全面综合地对异构调度方法进行评估,并测试调度效果。在对比实验中,本文选择了Linux默认的CFS调度器,该调度器已被广泛应用在各种多核处理系统中。

3.2　实验结果

在实际的调度场景中,调度器需要面临各种已知或未知类型的程序并对其进行调度。由于基准测试集中包含的程序

涵盖了大多数的程序类型，因此为了对调度器进行全面的评估本文将SPLASH-2基准测试集进行了如表3所示的分组。分组原则为使每组程序中既包含MSE值较低的程序，也包含MSE值较高的程序，并通过进行多组实验来尽可能充分地模拟调度器在真实环境下的复杂调度场景。为了保证实验评估结果的准确性，本文在Sniper下分别实现了异构调度器和CFS调度器，其中CFS调度器的实现方式参考了Linux Kernel 2.6.33所实现的版本[25]。在执行方式上，每组的4个多线程程序会在4个处理核上循环执行直到该组的所有程序的所有线程都至少执行完一遍为止，从而尽可能真实地模拟现实场景下的程序执行行为。

表3 实验用的程序组合

Tab. 3 Combination of programs in experiments

编号	程序组合
1	barnes/ cholesky / fft / fmm
2	volrend / fmm/ radix / fft
3	fft / raytrace / volrend / fmm
4	cholesky /fmm / raytrace / barnes
5	fmm / radiosity / radix / raytrace
6	radix /barnes/ raytrace / cholesky
7	raytrace /radiosity/cholesky / fft

图5给出了使用本文的异构调度方法在Sniper搭建的四核异构平台中分别运行表3中各组应用程序的实验结果，其中横坐标表示不同的程序组合编号，纵坐标是本文提出的异构调度方法与Linux默认的CFS相比所实现的IPC加速比。从图5可以看出，本文所构建的调度器相比CFS在组合2上加速比最低，在组合6上加速比最高，分别是1.12和2.49，在7种组合下平均实现了1.52的加速比。根据图4中ANN性能预测器在SPLASH-2基准测试程序上的表现可知，由于组合2含有MSE值最高的volrend和fft，因此整体的ANN性能预测器预测效果较其他组合相比最差，所以达到的IPC加速比最低。同理，组合6中各程序的MSE值整体比较接近，ANN性能预测器的整体预测效果较好，因而达到了最高的IPC加速比。

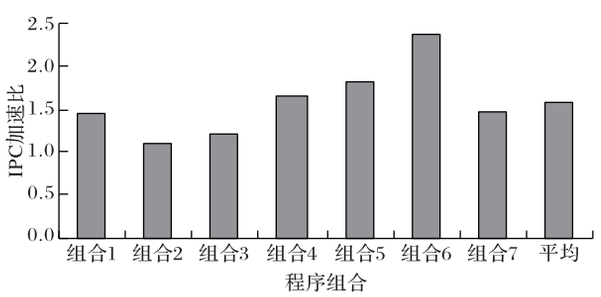

图5 本文异构调度方法的加速比

Fig. 5 Speedup ratio of proposed heterogenous scheduling method

图6给出了本文所提出的异构调度方法与CFS调度器在Sniper所搭建的四核异构平台中分别运行表3中各组应用程序的CPU资源利用率（CPU处于运行状态的时钟周期数与总的时钟周期数的比值）的情况，其中本文方法的平均CPU核资源利用率为93%，高于CFS的平均CPU核资源利用率（85%）。结果表明本文提出的异构调度器能够更加充分地利用CPU的资源。

通过实验数据可知，相对于利用虚拟运行时间大小来分配处理核资源的CFS调度算法，本文所提出的异构调度方法通过感知线程行为的变化和不同时刻下线程对处理核资源的需求，将线程及时分配到最适合其运行的处理核上，从而能够更好地发挥异构多核所带来的优势，获得更好的调度性能。

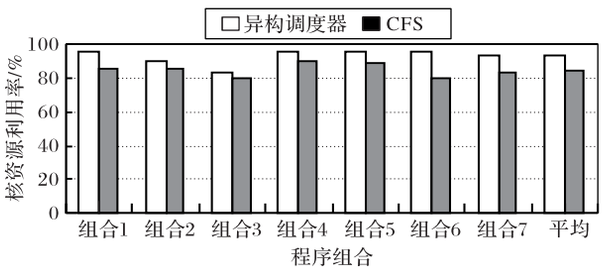

图6 本文异构调度方法和CFS的核资源利用率

Fig. 6 Core resource utilization rate of proposed heterogenous scheduling method and CFS

使用ANN性能预测器所带来的额外开销主要由预测器指令的执行时间开销和为了存储模型参数所带来的额外存储开销组成。在不考虑各级缓存命中缺失率、数据是否对齐等因素的影响下，一个ANN性能预测器每次执行所需要的浮点乘法指令条数约为250条、浮点加法指令为26条、访存相关的指令约为70条，而存储模型参数所需要的内存约为1 104 B。通过查询Intel指令手册[26]可知，理想情况下运行上述指令所需的总指令周期数最小为1 160，因此在4个处理核上分别运行4个ANN性能预测器所带来的总消耗为4 640个时钟周期，内存开销为552 KB。假设CPU时钟频率为1 GHz，当时间片为1 ms时，每个时间片有1×10^6时钟周期，运行ANN性能预测器的时间开销约为0.5%。而本文提出的方法整合了阶段检测器，此时ANN性能预测器往往需要经过往往成百上千个时间片才会被调用，因此引入ANN性能预测器所带来的实际额外时间开销会更低，甚至可以忽略。

综上，相对于经典的CFS调度器，本文所提出的调度程序引入了阶段检测机制和ANN性能预测器来帮助感知线程行为变化并为其选择更合适的处理核，最终在基本没有带来明显额外在线计算开销的前提下，获得了更好的性能收益。

4 结语

本文提出了一种基于阶段检测和机器学习预测模型的异构多核映射和调度方法：一方面通过阶段检测来感知线程的行为变化，按需进行重新映射和调度；另一方面通过采用机器学习技术来构建系统性能预测模型从而对程序在不同类型的处理核的IPC进行预测，并根据预测结果制定映射调度方案。实验结果表明，与Linux系统使用的CFS调度器相比，本文方法提升了52%的计算性能。

接下来的工作中，一方面希望将方法应用于具有更多处理核的异构多核处理平台上，为此需要研究能够高效地寻找最优或接近最优的搜索算法以提高该方法的可扩展性；另一方面由于目前的工作只考虑了性能优化，忽略了系统的能耗情况，因此也希望在当前的基础上引入对能耗指标的优化，从而实现异构多核平台的能效最大化。此外，采用更复杂的ANN模型来提高预测精度，以及更加全面地考虑资源竞争和线程同步等方面对调度方法设计的影响也是之后计划研究的方向。

参考文献（References）

[1] 赵姗，杨秋松，李明树. 性能非对称多核处理器下异构感知调度技术[J]. 软件学报，2019，30(4)：304-330.（ZHAO S，YANG Q S，LI M S. Heterogenity-aware scheduling research on performance asymmetric multicore processors[J]. Journal of Software，2019，30(4)：304-330.）

[2] BLAKE G, DRESLINSKI R G, MUDGE T. A survey of multicore processors[J]. IEEE Signal Processing Magazine, 2009, 26(6): 26-37.

[3] KAMDAR S, KAMDAR N. big. LITTLE architecture: heterogeneous multicore processing[J]. International Journal of Computer Applications, 2015, 119(1): 35-38.

[4] DAVIDSON J, LIEBALD B, LIU J, et al. The YouTube video recommendation system [C]// Proceedings of the 4th ACM Conference on Recommender Systems. New York: ACM, 2010: 293-296.

[5] WESTON J, MAKADIA A, YEE H. Label partitioning for sublinear ranking [C]// Proceedings of the 30th International Conference on Machine Learning. New York: JMLR. org, 2013: 181-189.

[6] 安鑫,张影,康安,等. 一种基于机器学习的异构多核处理器系统在线映射方法[J]. 计算机应用, 2019, 39(6): 1753-1759. (AN X, ZHANG Y, KANG A, et al. Machine learning based mapping approach for heterogeneous multi-processors system[J]. Journal of Computer Applications, 2019, 39(6): 1753-1759.)

[7] ZHANG Y, LAURENZANO M A, MARS J, et al. SMiTe: precise QoS prediction on real-system SMT processors to improve utilization in warehouse scale computers [C]// Proceedings of the 47th Annual IEEE/ACM International Symposium on Microarchitecture. Piscataway: IEEE, 2014: 406-418.

[8] MICHALSKA M, CASALE-BRUNET S, BEZATI E, et al. High-precision performance estimation for the design space exploration of dynamic dataflow programs[J]. IEEE Transactions on Multi-Scale Computing Systems, 2018, 4(2): 127-140.

[9] SAYADI H, PATEL N, SASAN A, et al. Machine learning-based approaches for energy-efficiency prediction and scheduling in composite cores architectures [C]// Proceedings of the 2017 IEEE International Conference on Computer Design. Piscataway: IEEE, 2017: 129-136.

[10] WANG L, LIU S, LU C, et al. Stable matching scheduler for single-ISA heterogeneous multi-core processors [C]// Proceedings of the 2015 International Workshop on Advanced Parallel Processing Technologies, LNCS 9231. Cham: Springer, 2015, 45-59.

[11] SHERWOOD T, PERELMAN E, HAMERLY G, et al. Discovering and exploiting program phases[J]. IEEE Micro, 2003, 23(6): 84-93.

[12] ROY P, ALAM M M U, DAS N. Heuristic based task scheduling in multiprocessor systems with genetic algorithm by choosing the eligible processor[J]. International Journal of Distributed and Parallel Systems, 2012, 3(4): 111-121.

[13] CHATTERJEE N, PAUL S, MUKHERJEE P, et al. Deadline and energy aware dynamic task mapping and scheduling for Network-on-Chip based multi-core platform [J]. Journal of Systems Architecture, 2017, 74: 61-77.

[14] GHARSELLAOUI H, KTATA I, KHARROUBI N, et al. Real-time reconfigurable scheduling of multiprocessor embedded systems using hybrid genetic based approach [C]// Proceedings of the IEEE/ACIS 14th International Conference on Information Systems. Piscataway: IEEE, 2015: 605-609.

[15] WEN Y, WANG Z, O' BOYLE M F P. Smart multi-task scheduling for OpenCL programs on CPU/GPU heterogeneous platforms [C]// Proceedings of the 21st International Conference on High Performance Computing. Piscataway: IEEE, 2014: 1-10.

[16] CAI E, JUAN D C, GARG S, et al. Learning-based power/performance optimization for many-core systems with extended-range voltage/frequency scaling [J]. IEEE Transactions on Computer Aided Design of Integrated Circuits and Systems, 2016, 35(8): 1318-1331.

[17] IMES C, HOFMEYR S, HOFFMANN H. Energy-efficient application resource scheduling using machine learning classifiers [C]// Proceedings of the 47th International Conference on Parallel Processing. New York: ACM, 2018: No. 45.

[18] SAWALHA L, BARNES R D. Phase-based scheduling and thread migration for heterogeneous multicore processors [C]// Proceedings of the 21st International Conference on Parallel Architectures and Compilation Techniques. Piscataway: IEEE, 2012: 493-493.

[19] RAPP M, PATHANIA A, MITRA T, et al. Prediction-based task migration on S-NUCA many-cores [C]// Proceedings of the 2019 Design, Automation and Test in Europe Conference and Exhibition. Piscataway: IEEE, 2019: 1579-1582.

[20] SHEN X, ZHONG Y, DING C. Locality phase prediction [C]// Proceedings of the 11th International Conference on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2004: 165-176.

[21] KUMAR R, TULLSEN D M, RANGANATHAN P, et al. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance [C]// Proceedings of the 31st International Symposium on Computer Architecture. Piscataway: IEEE, 2004: 64-75.

[22] VAN CRAEYNEST K, JALEEL A, EECKHOUT L, et al. Scheduling heterogeneous multi-cores through Performance Impact Estimation (PIE) [C]// Proceedings of the 39th Annual International Symposium on Computer Architecture. Piscataway: IEEE, 2012: 213-224.

[23] NEMIROVSKY D, ARKOSE T, MARKOVIC N, et al. A general guide to applying machine learning to computer architecture[J]. Supercomputing Frontiers and Innovations, 2018, 5(1): 95-115.

[24] CARLSON T E, HEIRMAN W, EYERMAN S, et al. An evaluation of high-level mechanistic core models [J]. ACM Transactions on Architecture and Code Optimization, 2014, 11(3): No. 28.

[25] JONES M T. Inside the Linux 2.6 completely fair scheduler: providing fair access to CPUs since 2.6.23[EB/OL]. [2019-09-19]. https://developer. ibm. com/tutorials/l-completely-fair-scheduler/.

[26] Intel. Intel 64 and IA-32 architectures optimization reference manual[EB/OL]. [2019-09-12]. https://www. intel. com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual. pdf.

This work is partially supported by the National Natural Science Foundation of China (U1613217), the Fundamental Research Funds for the Central Universities (JZ2020YYPY0092).

AN Xin, born in 1987, Ph. D., associate professor. His research interests include embedded systems, machine learning.

KANG An, born in 1995, M. S. candidate. His research interests include embedded software.

XIA Jinwei, born in 1994, M. S. candidate. His research interests include embedded software.

LI Jianhua, born in 1985, Ph. D., associate professor. His research interests include computer architecture, non-volatile memory.

CHEN Tian, born in 1974, Ph. D., associate professor. Her research interests include very large scale integrated circuit/system-on-chip low-power test, wearable computing.

REN Fuji, born in 1959, Ph. D., professor. His research interests include signal and information processing, computer vision.