文章编号：2095-2163(2022)10-0049-08 中图分类号：TP332 文献标志码：A

基于 ANN 的异构多核动态映射方法

李 阳[1]，李建华[1,2]，周义涛[1]，赵玉来[1]，胡海洋[1]
（1 合肥工业大学 计算机与信息学院，合肥 230601；2 合肥工业大学 情感计算与先进智能机器安徽省重点实验室，合肥 230601）

摘 要：异构多核架构逐渐成为多核处理器的主流平台，并且不同的应用场景对异构多核架构有不同的需求。通过集成不同类型的处理核，异构多核平台具有更好的灵活性，不同的应用程序可以根据需求动态地选择不同的核心组合来高效地处理任务。然而，随着核的类型、数量和支持的频率级别的增加，如何高效地调度和管理异构多核平台是当前的一个重要的挑战。现有的调度算法多是基于内核利用率的简单启发式算法，难以利用异构多核的特性获得最优的性能。本文提出了一种基于 ANN 的动态调度算法，能更好地发挥异构多核平台不同核心的优势来提升应用程序的执行效率。本文所提出的调度算法将应用程序的执行划分为等长的间隔，在相邻的间隔间动态调整映射策略。该调度算法由 2 个模块构成：首先根据不同核心在不同时刻的处理特征，利用神经网络预测应用程序下一阶段在该类型核心上运行的 *IPC*。其次，所提出的调度算法会根据预测的 *IPC* 值确定应用程序下一阶段被映射的核心类型，动态调整映射。实验表明，本文提出的方法相比于传统的轮询调度算法（Round Robin Scheduler，RRS）可以将每个时钟周期执行的指令条数（Instruction Per Cycle，*IPC*）平均提高 63%。

关键词：神经网络；动态映射；性能预测；异构多核

Heterogeneous multi-core dynamic mapping method based on neural network

LI YANG[1]，LI Jianhua[1,2]，ZHOU Yitao[1]，ZHAO Yulai[1]，HU Haiyang[1]
（1 School of Computer Science and Information Engineering，Hefei University of Technology，Hefei 230601，China；
2 Anhui Province Key Laboratory of Affective Computing and Advanced Intelligent Machine，
Hefei University of Technology，Hefei 230601，China）

【Abstract】 Heterogeneous multi-core platforms have gradually become the mainstream platforms for multi-core processors，and different application scenarios have different requirements for heterogeneous multi-core architectures. By integrating different types of processing cores，heterogeneous multi-core platforms are more flexible，and different applications can dynamically select different core combinations to handle tasks based on their needs. However，as the type，number，and frequency levels of the support increase，managing these platforms efficiently becomes challenging. Most of the existing scheduling algorithms rely on simple heuristics based on kernel utilization，and it is difficult to obtain optimal performance by taking advantage of heterogeneous multi-core features. In this paper，an ANN-based dynamic scheduling algorithm is proposed，which can better take advantages of different cores of heterogeneous multi-core platforms to improve the processing efficiency of applications. The dynamic mapping algorithm proposed in this paper consists of two modules：the algorithm divides the execution of the application into equal-length phases，and dynamically adjusts the mapping strategy between adjacent phases. First，according to the processing characteristics of different cores at different times，neural networks are used to predict the *IPC* running on that type of core in the next stage of the application. The core types of the application to be mapped in the next stage are then determined based on the *IPC* values by using the algorithm presented in this article. Experiments show that average instruction per clock cycle（*IPC*）of the proposed method is about 63% higher than that of the traditional Round Robin Scheduler（RRS）.

【Key words】 neural network；dynamic mapping；performance prediction；heterogeneous multi-core system

0 引 言

随着应用程序对计算能力需求的逐渐增长，以及半导体技术的逐步演进，多核处理器的核心数量一直在增加。异构多核架构通过在同一平台上集成多种类型的处理核，使得应用程序可以根据自身需求选择不同的核心组合来更加高能效地完成任务。因此，异构多核平台相比于同构多核平台拥有更高的灵活性，并且可以更合理地利用片上资源。目前，异构多核平台已经成为了当今嵌入式系统的主流解

基金项目：安徽省重点研究与开发计划（202004d07020004）；安徽省自然科学基金项目（2108085MF203）。

作者简介：李 阳（1996-），男，硕士研究生，主要研究方向：嵌入式系统与片上系统、机器学习；李建华（1985-），男，博士，副教授，主要研究方向：计算机体系结构、非易失性存储器。

通讯作者：李建华 Email：jhli@hfut.edu.cn

收稿日期：2022-03-17

决方案。充分发挥异构多核系统的优势需要高效的程序映射方法。

程序映射可以分为静态映射和动态映射。其中,动态程序映射可根据程序的阶段性特征以及不同类型处理核的特点进行映射的动态调整,能更充分地发挥平台的性能优势。静态程序映射是在设计时进行的,因此可以使用计算密集型搜索方法或精确的系统级映射模拟来寻找最佳或近最佳的映射解决方案。但是,通常情况下却无法处理(未知)动态工作负载。一个程序在其生命周期中可执行数百万、数十亿条、甚至更多的指令,并且其运行时的特征也在不断地变化。一个高效的映射算法应该能够根据这些变化的特征预测程序不同时刻在不同类型的核心上运行可获得的性能,以便获得更好的能效比。

近些年来,人工神经网络(Artificial Neural Network,ANN),是人工智能领域的研究热点之一。ANN 是由多个具有层次和连接关系的神经元相互连接而成,一个完整的人工神经网络是由输入层、输出层和隐藏层三部分组成。其中,输入层负责接收和传递信息,不进行任何计算。隐藏层位于输入层和输出层之间,可以与输入层和输出层相连,不同的隐藏层间也可以互相连接,这些连接都被赋予了一个数值权重,该权重与相应的输入参数相乘后,再与神经元其他传入连接的结果相加。接下去,这些总和被传递到神经元的激活函数中,通常选用的是一个线性函数。如此处理后的神经元输出将继续输入到下一层神经元或输出神经元。总地来说,ANN 可以通过合理的网络结构拟合任意的非线性方程,因此ANN 在数值预测和分类输出方面都有着巨大的潜力。

ANN 正在被广泛地应用于各种领域,例如情感分析[1]、预测估计[2]、生物医学[3]等。虽然 ANN 的应用受到越来越多的关注,但是到目前为止,将其应用于多核异构平台映射算法上的研究却仍处于起步阶段。为此,本文提出了一种基于 ANN 的异构多核动态映射方法 AbDM。

AbDM 的主要目标是最大化异构多核平台性能,在给定多个应用程序和一个具有多种不同类型核心的异构平台上,通过动态调整应用到核心的映射来实现目标。首先,使用基于 ANN 的性能预测器根据一段时间内的执行参数,预测每个应用下一阶段在不同类型核心上的性能。随后,重映射检测算法判断是否需要执行重映射。最后,若需要重映射,则由线程到核心类型匹配算法利用获得的性能值,确定应用到核心的映射。

本文的主要贡献如下:

(1)提出了一种线程到核心类型的匹配算法,在充分发挥平台性能优势的同时节省线程到核心类型绑定的时间。

(2)提出了一种重映射检测算法,当线程执行阶段发生变化时触发重映射过程,从而避免过于频繁的重映射,减少重新映射的时间和成本。

(3)提出了一种基于神经网络的性能预测器,将运行参数作为模型输入,预测性能。

实验结果证实本文提出的 AbDM 和轮询调度算法(RRS)相比平均每个时钟周期执行的指令条数(*IPC*)提高了 63%左右。

本文其余部分组织如下:第一节介绍了相关研究工作;第二节阐述了本文提出的 AbDM 的具体实现;第三节对本文方法进行了实验验证和分析;第四节则总结了本文工作并展望了未来的研究方向。

1 相关工作

异构多核平台上的应用映射在开发计算资源的多样性方面发挥着关键作用,对此已经有数十年的研究积累[4]。一般情况下对于异构多核映射问题大都采用一个线程映射到一个处理核心的模式[5]。当前针对异构多核架构映射问题的研究可以分为 3 类:离线、在线和混合(离线分析后在线使用)方法。为了找到近似最优的映射解决方案,现有研究大多采用基于搜索的方法,结合一些分析来评估考虑的映射与设计需求之间的关系。

对于离线方法,在离线状态下充分探索这些不同场景的资源分配,从而为每个场景指定一个最优或接近最优的资源分配方案[6]。例如,文献[7]提出了一种静态调度方法,这是一种基于种群的新算法,可以在运行时动态切换探索性和开发性搜索模式。静态调度器可以执行任务映射、调度和电压缩放。在文献[8]中,提出了一种算法来优化异构上应用程序的整体运行时间多核处理器。该算法将模拟退火和剪枝策略的组合,用于搜索应用程序和平台之间的最佳映射。上述离线方法通常使用离线分析来确定映射方案,虽然简单、但会产生额外的分析开销,并且没有充分考虑在线程序在执行过程中的动态变化。

与离线方法相比,在线资源管理缺少离线预处理阶段。一般每个应用程序占用的资源都是在运行时动态调整的。这种方法的资源分配计划是在运行

时计算出来的，方法的目的则是为了在短时间内得到一个比较好的资源分配计划。例如，文献[9]中提出的方法首先根据性能要求和资源可用性选择线程到核心的映射。然后，通过调整电压频率（$V-f$）水平来应用在线自适应，以在不牺牲应用性能的情况下实现能量优化。研究可知，在文献[10]中已有学者提出了一种基于机器学习的方法来增加异构多核平台上的平均故障工作量，最终结果表明该方法的平均工作量增幅可以高达 19.4%。在文献[11]中，还提出了一种不需要任何自身信息就能够在线对 OpenCL 应用进行功耗优化的一种自适应方法。

混合资源管理方法使用离线分析结果来做出运行时资源管理决策。第一步是在设计时进行全面的离线分析，并将分析结果集成到运行时控制器中，实现运行时的资源分配。例如文献[12]提出的方法，首先使用缓存分区技术来限制共享缓存的不可预测性，其次探讨核心频率的影响和缓存分区（CPs）对任务执行周期的影响及其对执行时间和能耗的影响。此外，文献[13]又提出了一种自适应映射方法。首先，使用性能监控计数器进行在线数据收集，然后使用离线训练性能预测模型来应用不同类型的数据。预测应用程序的性能，最后评估和选择资源组合（处理核心的数量和类型）。仍要提到的是，该方法还设置了一个资源管理器来监控应用程序性能、工作量，以及应用程序完成和新应用程序的到达情况，再以此来调整映射。

离线方法是设计时进行的，能够找到最优或者接近最优的映射方案，但是通常却无法处理动态工作负载问题。而在线方法能够解决动态工作负载问题，但是由于运行时计算能力有限，可能最终获得的映射方案不是最优方案或者接近最优的映射方案。混合资源管理方法结合了离线和在线的优点，可以在离线的时候对不同的应用场景进行充分的分析，也可以在运行时根据不同的执行场景动态地调整映射方案。所以目前混合资源管理方法获得了十分广泛的应用。

综上，本文采用混合资源分配方案来实现资源分配，可以充分考虑不同类型线程的运行时特性和不同内核的处理能力，从而最大限度地发挥平台的处理能力。

2　基于 ANN 的异构多核动态映射方法

本节主要介绍文中所提出的 AbDM 方案，图 1 给出了其总体框架。给定一组线程和一个异构多核平台，映射器首先为每个线程随机分配一个空闲核心。同时，收集必要的运行时信息，用来进行性能预测。然后，重映射检测器根据预测所得的性能判断是否需要执行重映射。在需要执行重映射的情况下，由线程到核心类型匹配算法，将线程和核心进行匹配，从而实现 IPC 优化。

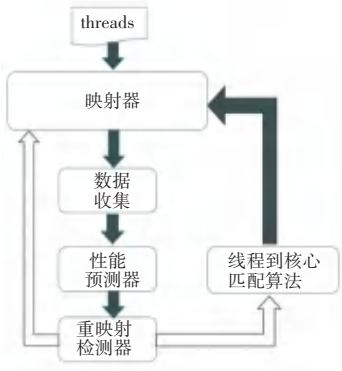

图 1　AbDM 总体框架

Fig. 1　Overview of the proposed approach, AbDM

2.1　问题定义

在本文中，通过 n 个线程 $T=(t_1,t_2,\cdots,t_n)$ 在 $H\times K$ 的多核平台上执行，表示为 $NoC=\{C_{1,1},\cdots,C_{h,k},\cdots,C_{H,K}\}$，其中 $C_{h,k}$ 表示核心位于 $2D$ 网格的第 h 行和第 k 列。每个核 $C_{h,k}$ 属于 m 种核类型的一种、表示为 $C=\{c_1,c_2,\cdots,c_m\}$，并且可以独立地执行 DVFS（Dynamic Voltage and Frequency Scaling）。本文所提出的 AbDM 方案使用每个时钟周期所执行的指令数、即指令数/周期（Instructions Per Cycle，*IPC*）作为性能指标。

由于程序运行在其生命周期中经常会发生一些阶段性的变化，并且不同的阶段有着不同的运行时的特征。所以，需要周期性地收集程序运行过程中相关特征数据。

2.2　运行时参数收集

AbDM 需要周期性地收集必要的运行时参数数据，以进行动态的重映射决策。AbDM 收集的参数如下：

（1）用于监视在特定内核上运行线程的性能的 *IPC* 值。

（2）缓存缺失，包括：一级数据缓存命中缺失率、二级缓存命中缺失率、三级缓存命中缺失率，以及读内存指令、写内存指令、浮点加法指令、浮点减法指令、浮点乘法指令和浮点除法指令。

2.3　性能预测器

为了将线程分配给异构多核系统中的特定核以最大化性能，必须了解每个线程在各种类型的核上

的执行情况[14]。实现这一目标的一种方法是在所有类型的内核上在线执行线程,并相应地采取最佳分配。这显然并不适用于在线映射方法,因为需要用到复杂的运行时迁移和性能度量。另一种方法是通过只在一种核心类型上运行(参见文献[7])来获得不同类型核心的线程性能。这种方法需要在设计时或运行时构建性能预测模型。由于运行时的学习模型可能会使动态映射方法不可扩展,并且还将引入太多的开销[15]。类似于文献[5,13]中的做法,本文选择在设计时使用机器学习技术为所有类型的核构建预测模型。

2.3.1 参数的选取

应用在某类核心上执行的 *IPC* 不仅取决于当前执行的类型,还和硬件资源相关。因此,需要将一些应用统计信息及执行参数作为预测器的输入参数。根据文献[16-17]中的方案所选择的参数和属性,本文选取了 9 个与 *IPC* 相关的参数来构建 ANN 学习模型。参数分别是缓存丢失率(一级数据缓存命中缺失率、二级缓存命中缺失率、三级缓存命中缺失率),以及读内存指令、写内存指令、浮点加法指令、浮点减法指令、浮点乘法指令和浮点除法指令。

2.3.2 参数的预处理

由于模拟直接收集的输入参数的统计量可能具有不同的尺度(例如,缓存失效率低于 1,而指令数可达数千万),因此需要对输入参数做预处理,以确保输入参数具有相同的尺度。另一方面,为了使用统计信息作为不同核心类型的性能预测的输入,统计信息应该尽可能通用。将统计数据规范化为比率可以确保神经网络输入的一致性,当使用从小核心收集的数据进行训练的时候,但却可能是使用从大核心收集的输入统计数据进行预测的。因此,本文将不同类型的指令数转换为指令比。

2.3.3 模型结构

在本文的 ANN 网络的输入层是由 9 个神经元组成,用来接收输入参数。一般来说,随着神经网络的隐藏层和隐藏层的节点数的增加可以降低网络误差、提高精度,但也会使网络复杂化,从而增加了网络的训练时间和出现“过拟合”的倾向。通过仿真实验,最终确定了性能预测器所使用的神经网络模型隐藏层数为 27,并且使用 *ReLU* 作为激活函数。在损失函数和学习率方面,本文选用了最常用的均方误差(Mean Squared Error, *MSE*)损失函数来对性能预测器进行训练,同时通过观察损失函数和学习率的关系,最终将学习率确定为 0.001。

2.4 重映射检测器

一个程序在其生命周期中可执行数百万甚至数十亿条指令,运行中的行为通常经历阶段变化[18]。在不同的阶段,可以观察到不同的执行特征[19]。因此,不能在每个信息收集阶段进行重新映射,而是只会在检测到某些阶段变化时进行重新映射,用以降低重新映射的成本。

在论文中定义一个 *IPC* 变化率 ΔIPC,当 ΔIPC 变化超过一定范围时就会触发重映射,从而调用后面的线程到核心类型匹配算法,反之本阶段将不进行重映射。ΔIPC 定义公式如下,

$$\Delta IPC = \frac{IPC(j+1) - IPC(j)}{IPC(j)} \tag{1}$$

其中,$IPC(j)$ 是 j 阶段的 *IPC*,$IPC(j+1)$ 是 $j+1$ 阶段的预测 *IPC*。

2.5 线程到核心类型匹配算法

本文的目的是获得线程到核心的最佳映射,从而充分发挥异构多核平台的性能优势。本文所提出的线程到核心类型的匹配算法,以性能预测器所预测的下一阶段的各个线程到各种类型核心的 *IPC* 值为基础,来确定线程拟将映射的具体核心类型。从而获得线程到核心的最优匹配,使得异构多核平台性能得以充分发挥,总的 *IPC* 值达到了最大化。

本文中定义线程 i 在核心类型 j 上的 *IPC* 值为 $IPC(i/j)$,并且 j 类型核心的数量为 $|j|$。线程与核心类型的绑定表示为 B。B 是一个一维向量,其中第 i 个位置的数 j 表示线程 i 映射到类型为 j 的核心上。

线程到核心类型的匹配算法可分为 4 步:

Step 1 初始化 t 中的所有线程到核心类型的绑定,使得初始绑定为空(这里设为 X)。

Step 2 对于所有类型的核心,根据性能预测器预测所得的 *IPC* 值,将对应的线程进行排序。例如对于核心类型 j,预测所有线程下一阶段在此类核心上的 *IPC* 值。然后根据预测映射所得的 *IPC* 值,为对应的线程进行排序(*IPC* 值越大的线程越靠前),得到序列 $TQ(TQ\_c_1(\cdots,t_k,\cdots),TQ\_c_2,\cdots,TQ\_c_j,\cdots)$。

Step 3 根据每种核心类型的处理能力,对核心类型进行排序,得到 $Q\_c(\cdots c_j\cdots)$。

Step 4 为了最大限度地提高系统性能,需要优先选择使用更强大的核心。因此,本算法的核心思想是根据线程处理能力的顺序(由 Q_c 表示)将线程分配给内核。首先依据序列 Q_c 中核心的顺序,依次查看该核心对应的 TQ 序列中的前 $|c_j|$ 个

线程的分配情况。若当前线程未被分配，将线程分配到当前核心。反之，若已经分配，则将线程映射到所得 *IPC* 较大的核心上。映射后获得的 *IPC* 值较小的核心对应的 *TQ* 序列上空出的核心位置，由对应的 *TQ* 序列第 $|c_j|$ 开始的第一个未被分配的线程补上。研发得到伪代码详见如下。

输入：

Thread set $T=\{t_1,t_2,\dots,t_n\}$,

Core Type set $C=\{c_1,c_2,\dots,c_n\}$

Estimated *IPC* values of each t_i on each c_j denoted by $IPC(t_i/c_j)$

输出：

thread to core type binding $B:T->C$

$TQ(TQ\_c_1, TQ\_c_2,\dots,TQ\_c_j,\dots)$

$TQ\_c_j:(\dots,t\_k,\dots)$

$Q\_c:(\dots,c_j,\dots)$

$|c_j|$：核心类型 j 的核心数

//Phase 1：Initialization

1：for t_i in T

2：　$B(t_i)=X$// 给 B 置初值 X

3：end

//Phase2：根据映射所得的 *IPC* 值，将对应的线程进行排序（*IPC* 值越大的线程越靠前）

4：for $j\leftarrow 1$ to m

5：　sort T according to $IPC(t_i/c_j)$ => obtain queue TQ_c_j

6：end

//Phase3：根据每种核心类型的处理能力，对核心类型进行排序（由强到弱）

7：sort c_j according to Processing Power =>obtain Q_C

//Phase4：线程到核心类型匹配

8：for c_j in Q_C from 1 to m

9：　for (l from 1 to $|c_j|$)

10：　　if ($B(TQ\_c_j[l]==X)$)

11：　　$B(B(TQ\_c_j[l]))=c_j$

12：　　break

13：　　else if ($IPC(TQ\_c_j[l]/B(TQ\_c_j[l]))<IPC(TQ\_c_j[l]/c_j)$)

14：　　$remove<=B(TQ\_c_j[l])$

15：　　$B(TQ\_c_j[l])<=c_j$

16：　　从 $TQ(remove)$ 队列中第 $|c_j|$ 开始向后找第一个未被分配的，

17：　　若有记为 t_p，那么把 t_p 分配到这个 *remove* 核心类型上，即

18：　　$B(t\_p)=remove$，反之则此类核心空出一个位置

19：　　然后将 $TQ(remove)$ 的 $B(TQ\_c_j[l])$ 移除

20：　　break

21：　　else

22：　　前面已分配的大，保留前面的分配

23：　　从 $TQ(c_j)$ 队列中第 $|c_j|$ 开始向后找第一个未被分配的，

24：　　若有记为 t_p，那么把 t_p 分配到这个核心类型上，即

25：　　$B(t\_p)=c_j$，反之则此类核心空出一个位置

26：　　然后将 $TQ(c_j)$ 队列中的 $B(TQ\_c_j[l])$ 移除

27：　　end

28：　end

29：end

3　实验评估

为了评估 AbDM 的性能，本文基于 Sniper 模拟器搭建了一个异构多核平台，用来模拟不同应用在实际运行中的行为。并且，将 AbDM 与常见的轮询调度算法（Round Robin Scheduler，RRS）[20] 进行对比，用来分析 AbDM 方法的优势。

轮询调度主要是通过交换在大核上执行的线程和在小内核上执行的线程进而实现对性能的优化。之所以选择轮询调度进行对比，是因为轮询调度算法可以在 Linux 调度程序和公平感知调度程序上为单线程和多线程工作负载提供性能改进[21]。

3.1　实验设置

实验使用模拟器 Sniper 搭建了一个 4 行 4 列的异构多核平台。该平台由 2 种处理能力不同的核心组成，这 2 种核心称为大核和小核，大核与小核各占两行，具体如图 2 所示。并且本文中的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，发射宽度均为 4，2 种类型的处理核具有相同的缓存架构。其中，L_1 Cache 为 256 KB，L_2 Cache 为 512 KB，共享的 L_3 Cache 大小设置为 8 MB。大核拥有 128 的指令窗口大小和长度为 48 的读取队列，而小核的指令窗口大小为 16，读取队列长度为 6。频率调节的步长为 0.2 GHz，对于每个频率，电压的设置参照文献[22]。

图 2 Sniper 模拟器布置的 16 个核心的系统平面图

Fig. 2 Floorplan of a 16-core system simulated in Sniper

从基准测试 SPLASH-2 中选择了 10 种不同的应用,并且将这些应用组成不同的分组,将分组内的所有程序在搭建的模拟平台上并发执行,用来模拟现实世界的程序执行情况。具体的程序组合见表 1。

表 1 应用程序组合

Tab. 1 Combination of programs

编号	程序组合
1	cholesky / ocean / ftt / volrend
2	barnes / radix / lu / radiosity
3	cholesky / ocean / ftt / volrend / fmm
4	barnes / radix / lu / radiosity/ raytrace
5	cholesky / ocean / ftt / volrend / fmm /raytrace
6	barnes / radix / lu / radiosity / raytrace / fmm
7	cholesky / ocean / ftt / volrend / fmm /raytrace /radiosity
8	barnes / radix / lu / radiosity / raytrace / fmm / lu

3.2 实验结果与分析

为了更好地评估本文提出的方法,实验分别对性能预测器、应用到核心的匹配算法以及 AbDM 整体进行评估。从局部和整体两个方面对本文提出的算法进行仿真测试。

3.2.1 性能预测器的评估

针对 2 种类型核的 16 核平台,实验中搭建了 2 种人工神经网络模型。一种是根据小核的线程执行信息来预测大核的 *IPC*,另一种是根据大核的线程执行信息来预测小核的 *IPC*。根据小核的线程执行信息来预测大核的 *IPC* 的模型称为 $model_1$,根据大核的线程执行信息来预测小核的 *IPC* 的模型称为 $model_2$。2 种神经网络模型都是使用上述的不同应用组合所收集的数据集训练得到的,数据集的收集周期为 30 ms,即每隔 30 ms 收集一次用以训练性能预测器的参数的数值。

为了更好地评估 AbDM 的性能,实验用 10 个 splash-2 基准应用作为 $model_1$ 和 $model_2$ 模型评估的测试集。图 3 给出了 $model_1$ 和 $model_2$ 预测不同应用 *IPC* 的误差。从图 3 中可以看出,2 个预测模型在应用 barnes 上均取得最小误差,同时在应用 volread 上的效果也最不理想,但是即使在最差情况 *MSE* 也仅为 0.31,并且 $model_1$ 和 $model_2$ 的平均 *MSE* 仅为 0.15 和 0.17。由实验数据还可以看出,总体上,大核性能预测器和小核性能预测器均具有较好的预测效果。

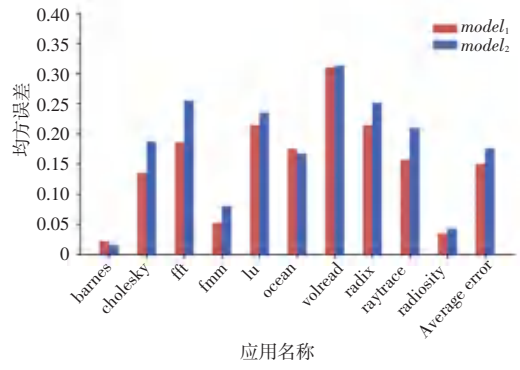

图 3 预测器误差

Fig. 3 *MSE* of ANN predictor

3.2.2 线程到核心类型匹配算法的评估

本部分通过将线程到核心类型匹配算法与穷举算法进行比较,来评估算法的性能。具体的比较有 2 方面。一方面是线程到核心类型匹配所需的时间,另一方面是匹配后所获得的 *IPC* 值。

通过设置不同的线程数、核心类型数以及每种核心的数量,来对比本文提出的线程到核心类型匹配算法与穷举法的性能。例如表 2 中,组合 1 表示的是 4 个线程运行于拥有 2 种不同类型核心的异构平台上,并且 2 种类型的核心数量都是 4 个。对于穷举算法来说,算法考虑了所有线程到核心类型匹配的可能性,故毋庸置疑可知,穷举算法获得的 *IPC* 值是最大值。接下去,由表 3 的实验结果可知,随着测试配置越来越复杂,穷举法所耗费的时间增长速度远远高于本文提出的方法,并且在配置 6 的时候,穷举法没有时间结果,是由于运行时间超过了 0.5 h,为此停止了实验。在总的 *IPC* 值方面,本文提出的方法可以接近于最优值。能够非常有效地获得接近最优的解。

表 2 异构多核配置

Tab. 2 Heterogeneous multi-core configuration

Config No.	threads	Core types	cores
1	4	2	(4,4)
2	8	2	(6,6)
3	12	3	(4,6,8)
4	12	4	(4,6,6,8)
5	16	3	(4,6,8)
6	16	5	(4,4,6,6,8)
7	20	4	(4,6,6,8)
8	20	6	(4,4,6,6,8,8)

表 3　本文提出的方法与遍历法的运行时间

Tab. 3　Running time of the method proposed in this article vs. the traversal method　　ms

Config No.	本文提出的方法	遍历	Config No.	本文提出的方法	遍历
1	0.61	0.46	5	0.64	4.8×10^5
2	0.49	7.04	6	0.79	–
3	0.64	1.5×10^4	7	0.73	–
4	0.71	3.6×10^5	8	0.81	–

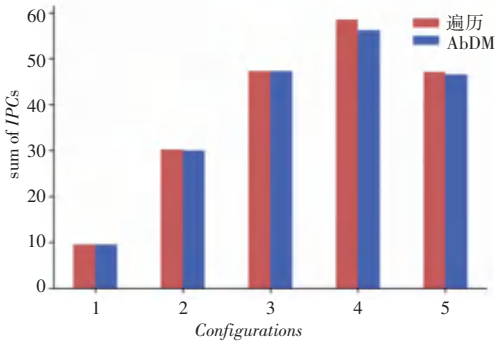

图 4　AbDM 与遍历法所获得的 *IPC* 对比

Fig. 4　Comparison of *IPC* obtained by AbDM vs. *IPC* obtained by the traversal method

3.2.3　对基于神经网络的异构多核动态映射方法的评估

为了评估 AbDM 与现有技术相比的优劣，实验中使用 RR 作为基准方法，并对 8 个应用程序组合进行实验，以观察本文提出的方法与 RRS 比较所实现的 *IPC* 加速比。

AbDM 和轮询调度的加速比如图 5 所示。图 5 中，横坐标表示不同的程序组合所对应的编号，纵坐标表示本文提出的方法与 RR 比较所获得的加速比。从图 5 中可以看到，AbDM 在所有的 8 种实验配置下效果均优于轮询调度算法，并且平均吞吐量提高了 63%。这是因为 AbDM 考虑到了程序生命周期中的阶段性变化，为不同阶段的程序选择适合当前阶段的核心类型，从而最大限度地发挥异构多核平台的性能优势。另一方面，AbDM 执行更少的重映射计算，每隔一个特定的周期进行一次重映射检测，当且仅当满足重映射条件是触发重映射。

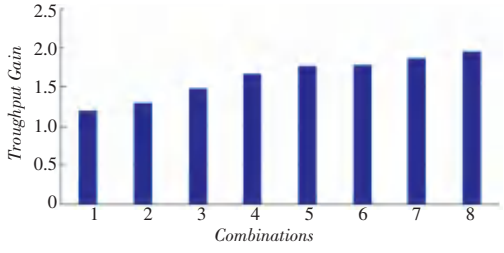

图 5　AbDM 和轮询调度的加速比

Fig. 5　Acceleration ratio of the proposed method and RRS

4　结束语

本文提出了一种基于神经网络的异构多核动态映射算法。该方法首先根据执行阶段不同线程的特点，通过人工神经网络预测不同类型核上各线程下一阶段的 *IPC* 值。然后，重映射算法确定是否需要重映射，并在必要时执行重映射算法。反之，若无需重映射，则保持当前映射不变。仿真实验结果表明本文提出的方法相对于传统的轮询算法优势明显，平均吞吐量提高了 63%。

此外随着处理核数量的增加，处理器中晶体管密度同步提高，现代多核处理器具有更高的功率密度，进而导致芯片温度也有同步升高，最终会对系统性能造成一定的负面影响[23]。所以在未来的工作中，会考虑引入一个新的参考因素热安全。在保证热安全的前提下，最大化异构多核平台的性能。

参考文献

[1] 林巧民，齐柱柱. 基于 HMM 和 ANN 混合模型的语音情感识别研究[J]. 计算机技术与发展，2018，28(10)：74-78.

[2] 倪晓兰. 滑坡趋势的 EMD-ANN 方法预测[J/OL]. 宜宾学院学报：1-11[2022-03-09]. http://kns.cnki.net/kcms/detail/51.1630.Z.20220307.1531.002.html.

[3] KRENEK J, KUCA K, BARTUSKOVA A, et al. Artificial neural networks in biomedicine applications [M]//WONG W. Proceedings of the 4th International Conference on Computer Engineering and Networks. Lecture Notes in Electrical Engineering. Cham：Springer, 2015，355：133-139.

[4] SINGH A K, SHAFIQUE M, KUMAR A, et al. Mapping on multi/many-core systems：Survey of current and emerging trends [C]//2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX, USA：IEEE, 2013：1-10.

[5] RAPP M, PATHANIA A, MITRA T, et al. Prediction-based task migration on s-nuca many-core[C]// 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). Florence, Italy：IEEE，2019：1579-1582.

[6] QUAN Wei, PIMENTEL A D. A scenario-based run-time task mapping algorithm for mpsocs[C]// 2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC). Austin, TX：IEEE, 2013：1-6.

[7] FANG Juan, ZONG Huan, ZHAO Haoyan, et al. Intelligent mapping method for power consumption and delay optimization based on heterogeneous NoC platform[J]. Electronics, 2019, 8(8)：912.

[8] ORSILA H, KANGAS T, SALMINEN E, et al. Automated memory-aware application distribution for multi-processor system-on-chips[J]. Journal of Systems Architecture, 2007, 53(11)：795-815.

[9] REDDY B K, SINGH A K, BISWAS D , et al. Inter-cluster thread-to-core mapping and DVFS on heterogeneous multi-cores [J]. IEEE Transactions on Multi-Scale Computing Systems, 2017, 4(3)：369-382.

[10] TONETTO R B, ROCHA H M D A, NAZAR GL, et al. A machine learning approach for reliability - aware application mapping for heterogeneous multicores[C]// 2020 57th ACM/IEEE Design Automation Conference (DAC). IEEE, 2020:1-6.
[11] ANGIOLETTI D, BERTANI F, BOLCHINI C, et al. A runtime resource management policy for OpenCL workloads on heterogeneous multicores[C]// 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE). Florence, Italy: IEEE, 2019:1385-1390.
[12] SHEIKH S Z, PASHA M A. Energy-Efficient Cache-Aware Scheduling on Heterogeneous Multicore System [J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 33 (1): 206-217.
[13] BASIREDDY K R, SINGH A, AI-HASHIMI, et al. AdaMD: adaptive mapping and DVFS for energy-efficient heterogeneous multicores[J]. IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 2019, 39 (10): 2206-2217.
[14] PRICOPI M, MUTHUKARUPPAN T S, VENKATARAMANI V, et al. Power-performance modeling on asymmetric multi-cores [C]// 2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES). Montreal, QC, Canada: IEEE, 2013:1-10.
[15] UJJWAL G, PATIL A C, BHAT G, et al. Dypo: Dynamic pareto-optimal configuration selection for heterogeneous mpsocs [J]. ACM Transactions on Embedded Computing Systems (TECS), 2017, 16(5s): 123:1-123:20.
[16] NEMIROVSKYD, ARKOSE T, MARKOVIC N, et al. A machine learning approach for performance prediction and scheduling on heterogeneous CPUs[C]//2017 29th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). Campinas, Brazil: IEEE, 2017:121-128.
[17] NEMIROVSKY D, ARKOSE T, MARKOVIC N, et al. A general guide to applying machine learning to computer architecture [J]. Supercomputing Frontiers and Innovations: An International Journal, 2018,5 (1): 95-115.
[18] KUMAR R, TULLSEN D M, RANGANATHAN P, et al. Single-ISA heterogeneous multi-core architectures for multithreaded workload performance [C]// Proceedings 31st Annual International Symposium on Computer Architecture. Munich, Germany: IEEE, 2004:64-75.
[19] SHEN Xipeng, ZHONG Yutao, DING Chen. Locality phase prediction[J]. ACM SIGPLAN Notices, 2004, 39 (11): 165-176.
[20] MARKOVIC N, NEMIROVSKY D, MILUTINOVIC V, et al. Hardware round-Robin scheduler for single-ISA asymmetric multi-core[M]// TRÄFF J, HUNOLD S, VERSACI F. Euro-Par 2015: Parallel Processing. Euro-Par 2015. Lecture Notes in Computer Science(). Berlin: Springer, 2015, 9233:122-134.
[21] CRAEYNESTK V, AKRAM S, HEIRMAN W, et al. Fairness-aware scheduling on single-ISA heterogeneous multi-cores[C]// Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques. Edinburgh, UK: IEEE, 2013:177-187.
[22] GRENAT A, PANT S, RACHALA R , et al. 5.6 adaptive clocking system for improved power efficiency in a 28 nm x86-64 microprocessor[J]. 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). San Francisco: IEEE, 2014:106-107.
[23] ANSARI M, PASANDIDEH M, SABER-LATIBARI, et al. Meeting thermal safe power in fault-tolerant heterogeneous embedded systems[J]. IEEE Embedded Systems Letters, 2020, 12 (1): 29-32.

(上接第 48 页)

[3] 韩宝玲,赵锐,罗庆生,等. 基于粒子群算法的四足机器人静步态优化方法[J]. 北京理工大学学报,2017,37(05):461-465.
[4] 易静,李光,薛晨慷,等. 四足机器人 Trot 步态偏航分析及其控制[J]. 机械与电子,2022,40(03):58-64.
[5] 于权伟,李光,谢楚政,等. 改进混沌麻雀搜索算法及其在冗余机械臂逆运动学求解中的应用[J/OL]. 机械科学与技术:1-7[2021-11-23]. https://doi.org/10.13433/j.cnki.1003-8728.20200624.
[6] WANG W, XU Y, HAN X, et al. A study of function-based wind profiles based on least squares method: A case in the suburbs of Hohhot[J]. Energy Reports, 2022, 8: 4303-4318.
[7] MA Zihan, LIANG Yanbin, TIAN Hua. Research on gait planning algorithm of quadruped robot based on central pattern generator [C]//2020 39th Chinese Control Conference (CCC). Shenyang, China: IEEE, 2020: 3948-3953.
[8] 游洋威. 基于三维简化模型的四足机器人 Trot 步态运动控制研究[D]. 哈尔滨:哈尔滨工业大学,2013.
[9] 李华师. 四足机器人仿生运动控制理论与方法的研究[D]. 北京:北京理工大学,2014.
[10] 唐锴,吴焕龙,张良安,等. 基于改进粒子群算法的四足机器人机体尺寸及质心位置优化[J]. 机械传动,2021,45(07):67-73.
[11] LIU Guiyun, SHU Cong, LIANG Zhongwei, et al. A modified sparrow search algorithm with application in 3d route planning for UAV[J]. Sensors, 2021, 21(4): 1224.
[12] JIA Jianfang, YUAN Shufang, SHI Yuanhao, et al. Improved sparrow search algorithm optimization deep extreme learning machine for lithium-ion battery state-of-health prediction[J]. Iscience, 2022, 25(4): 103988.
[13] XUE Jiankai, SHEN Bo. A novel swarm intelligence optimization approach: Sparrow search algorithm [J]. Systems Science & Control Engineering, 2020, 8(1): 22-34.
[14] KIMURA H, SHIMOYAMA I, MIURA H. Dynamics in the dynamic walk of a quadruped robot [J]. Advanced Robotics, 1989, 4(3): 283-301.
[15] XU R W, HSIEH K C, CHAN U H, et al. Analytical review on developing progress of the quadruped robot industry and gaits research [C]//2022 8th International Conference on Automation, Robotics and Applications (ICARA). Prague: IEEE, 2022:1-8.