doi:10.11887/j.cn.202502014 http://journal.nudt.edu.cn

符合粒子输运模拟的专用加速器体系结构

张建民,刘津津*,许炜康,黎铁军
(国防科技大学 计算机学院,湖南 长沙 410073)

摘 要:粒子输运模拟是高性能计算机的主要应用,对于其日益增长的计算规模需求,通用微处理器由于其单核结构复杂,无法适应程序特征,难以获得较高的性能功耗比。因此,对求解粒子输运非确定性数值模拟的程序特征进行提取与分析;基于算法特征,对开源微处理器内核架构进行定制设计,包括加速器流水线结构、分支预测部件、多级 Cache 层次与主存设计,构建一种符合粒子输运程序特征的专用加速器体系结构。在业界通用体系结构模拟器上运行粒子输运程序的模拟结果表明,与 ARM Cortex-A15 相比,所提出的专用加速器体系结构在同等功耗下可获得4.6倍的性能提升,在同等面积下可获得3.2倍的性能提升。

关键词:粒子输运模拟;专用加速器;程序特征;分支预测;多级 Cache
中图分类号:TP303 **文献标志码**:A **文章编号**:1001-2486(2025)02-155-10

论文拓展

Specific accelerator architecture conforming to particle transport simulation

ZHANG Jianmin, LIU Jinjin, XU Weikang, LI Tiejun*
(College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China)

Abstract: Particle transport simulation is one of the main applications of high performance computers. But facing to its fast growing compute requirements, the general-purpose microprocessors cannot adapt to the particle transport program features, owing to the complexity architecture of its single core, and then it is difficult to obtain high ratio of performance and power. Therefore, the program features of the particle transport non-deterministic numerical simulation were extracted and analyzed. Based on the characteristics of the algorithm, the architecture of open-source microprocessor core was designed, including pipeline structure, branch prediction unit, multi-level Cache hierarchy and main memory design. A specific accelerator architecture was designed in accordance to the particle transport program features. The simulation results of running the particle transport program on the general architecture simulator show that, as compared with ARM Cortex-A15, the proposed specific accelerator can achieve 4.6 times performance improvement under the same power consumption, and 3.2 times under the same area.

Keywords: particle transport simulation; specific accelerator; program feature; branch prediction; multi-level Cache

粒子输运模拟是高性能并行计算机的主要应用之一,粒子输运理论目前已经被广泛应用在许多不同的物理和工程领域,包括核数值模拟[1]、核反应堆设计[2]、医学放射性治疗[3]、天体物理等领域[4]。对粒子输运方程的数值求解,最常见的是蒙特卡罗(Monte Carlo,MC)方法。基于MC方法的粒子输运程序可以真实地模拟粒子在不同材料介质中输运的过程和求解各种复杂几何条件及截面复杂变化情况的问题。

由于应用规模越来越大,粒子输运问题的数值求解计算量迅速增加,以及特定环境下求解的实时性需求,粒子输运模拟的高效求解方法受到研究者的重视。Liu 等[5]研究粒子传输的扫描调度算法,提出了一种并行空间域分解算法,在天河二号高达16 384个处理器上具有良好的拓展性。Alguacil 等[6]对粒子输运程序的访存特性进行研究,对其代码进行修改,优化后程序初始化时间减少了20%至60%。Leppänen 等[7]加速了粒子输运程序中裂变源的收敛,通过从改进的源猜测开始模拟,减少了所需的非活动周期数。Kowalski 等[8]对表面跟

收稿日期:2022-12-26
基金项目:国家重点研发计划资助项目(2022YFB2803405);国家自然科学基金资助项目(62072464,U19A2062)
第一作者:张建民(1979—),男,山西平遥人,副研究员,博士,硕士生导师,E-mail:jmzhang@nudt.edu.cn
***通信作者**:刘津津(1998—),女,广东汕头人,硕士研究生,E-mail:liujinjin@nudt.du.cn
引用格式:张建民, 刘津津, 许炜康, 等. 符合粒子输运模拟的专用加速器体系结构[J]. 国防科技大学学报, 2025, 47(2): 155-164.
Citation: ZHANG J M, LIU J J, XU W K, et al. Specific accelerator architecture conforming to particle transport simulation[J]. Journal of National University of Defense technology, 2025, 47(2): 155-164.

踪算法进行了优化,相比标准方法,该算法可获得7%至21%的加速比。Zhang等[9]实现了CPU和GPU的异构混合加速,实验表明该架构对算法的性能加速可达2倍。Sweezy[10]将中子注量估计器卸载到GPU;与八核CPU相比,移植到GPU可获得4~5倍的加速。Tithi等[11]通过将Quicksilver[12]移植到英特尔新架构可编程集成统一内存架构(programmable integrated unified memory architecture, PIUMA)上,使8核PIUMA架构的性能达到28核CLX 8280的2倍。Fu等[13]面向MC程序在gem5中配置高级精简指令(advanced RISC machines, ARM)内核,在4核精简ARM内核下实现了4.45倍的功耗性能优势和2.78倍面积性能优势。Gorshkov等[14]在Intel MIC处理器上加速了光子在复杂几何介质中迁移的MC模拟算法,通过改变存储光子轨迹的数据结构来减少内存使用。

在处理器体系结构性能评估方面,Endo等[15]在gem5[16]上模拟了Cortex-A8和Cortex-A9处理器,通过和真实硬件比较评估模拟的准确性。他们还基于gem5和McPAT[17]对非对称嵌入式核的性能、功耗和面积进行评估[18]。Walker等[19]提出了一种方法评估CPU性能模型中的误差源;通过实验对Cortex-A7和Cortex-A15进行误差源分析,平均绝对百分比误差和平均百分比误差分别提高了18%和10%。Semakin[20]通过在gem5中实现CPU模型并采用了2~32个核运行程序,得到每个核数下的加速比及扩展效率。

近年来,由于半导体工艺技术的不断进步,微处理器依赖加深流水线深度,提高频率,实现了性能的不断提升。然而,由于功耗和散热的影响,多数微处理器在性能提升的同时碰上了“功耗墙”问题。尤其是面对粒子输运数值模拟的精度、规模与复杂性急剧增加的现状,传统处理器由于其内核结构较复杂,无法适应粒子输运程序特征,因此难以获得较高的性能功耗比。为实现粒子输运问题的高效求解,构建面向粒子输运模拟的专用加速器体系结构成为解决其计算效率、功耗和面积的一个新的技术途径。研究的主要目标是实现性能功耗比以及性能面积比更优的小核,使得众核加速器能够在同等功耗或面积下实现更多的核,从而达到总体性能最佳。

因此,本文对符合粒子输运模拟的专用加速器体系结构展开研究。基于业界通用体系结构模拟器gem5,对求解粒子输运模拟的MC程序特征进行提取与分析。基于算法特征,对开源通用微处理器内核架构进行定制设计,通过在体系结构模拟器上的实验探索加速器流水线架构、分支预测部件、多级Cache结构与主存设计,最终构建一种符合粒子输运程序特征的专用加速器体系结构。基于gem5与McPAT运行粒子输运程序的模拟结果表明,与ARM Cortex-A15相比,所提出的专用加速器体系结构在同等功耗下可获得4.6倍的性能提升,在同等面积下可获得3.2倍的性能提升。

1 粒子输运模拟程序特征分析

Quicksilver程序是Mercury[21]的代理程序。Mercury是美国劳伦斯利弗莫尔国家实验室开发的粒子输运程序,但代码不公开。Quicksilver是Mercury多个特征的模型,可精确模拟随机采样而产生的分支,以及与读取横截面表关联的内存访问等操作。

Quicksilver程序的伪代码如算法1所示。其中for循环内的三个函数是程序的主体。cycleInit()函数的功能是负责在每个循环的时间步长开始时建立目标粒子数。cycleTracking()函数是程序的主要运算部分,它跟踪每个粒子运动过程,计算粒子运动的距离,直到粒子被吸收或者逃逸,涉及大量的分支和计算。第三个cycleFinalize()函数统计每个时间步长的各类计算数据信息。

算法1 粒子输运程序 Quicksilver

Alg. 1 Particle transport simulation program Quicksilver

输入:循环长度和粒子数

输出:碰撞反应结果

```
int main()
{Parameters reading;
 Configures printing;
 system initialization;
 for(int i=0;i<n Steps;++i)
 {cycleInit();
  cycleTracking();
  cycleFinalize();}
 mpiFinalize;}
cycleInit(){
  source in particles;
  population control;}
cycleTracking(){
  for allparticles{
    do{compute distance to census;
       compute distance to facet;
       compute distance to reaction;
       do segment with shortest distance;
       increment tallies;}
       until census, absorbed, escaped;}}
cycleFinalize(){
    reduce all tallies;
    Free memory;}
```

基于 gem5 全系统运行 Quicksilver 程序,在粒子规模数为 100 000 的情况下,统计三个函数的平均运行时间。其中 cycleTracking( ) 函数是程序最耗时部分,约占程序总时长的 97.9%,并且随着粒子规模数的增大,运行时间也显著增加。cycleInit( ) 函数运行时间占比很小,约占总时长的 2.0%。cycleFinalize( ) 函数模拟不同规模粒子时运行时间基本上没有变化并且所用时间占比也极小,约占 0.1%。因此将 cycleTracking( ) 函数作为加速器体系结构设计的主要优化目标。

为了分析三个函数的访存情况,在 gem5 中模拟了 100 000 粒子规模,对访存带宽进行了统计,得到图 1。cycleInit( ) 和 cycleFinalize( ) 有更高的访存带宽需求,但仅是瞬时带宽高,访存需求集中。跟踪整个程序运行时发现,cycleTracking( ) 函数会占用大量的存储空间,符合基于历史的 MC 程序在初始化完粒子库前后的访存特性。

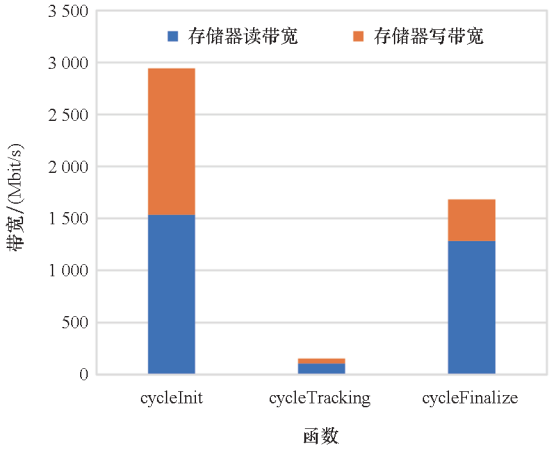

图 1　三个主要函数的访存带宽

Fig. 1　Memory access bandwidth of three main functions

由于粒子输运程序包含了大量的分支操作,因此通过实验分析分支预测部件在程序中的执行情况。在 gem5 的全系统中设置了一个动态分支预测器,运行粒子规模为 100 000 的 Quicksilver 程序,然后统计三个主要函数的分支预测操作数的占比情况。其中 Quicksilver 程序中的大量分支运算都集中在 cycleTracking( ) 函数部分,占比约为 96.2%,而 cycleInit( ) 和 cycleFinalize( ) 的分支预测操作次数分别仅占约整个程序的 1.7% 和 2.1%。

通过上述实验,得到粒子输运程序的一些特性:第一,粒子跟踪阶段运行时间占比高;第二,程序指令规模较小,访存需求集中;第三,分支操作多,随机性突出。这些程序特征将为粒子输运模拟加速器体系结构设计提供重要的指导。

2　专用加速器体系结构设计

基于 Quicksilver 的程序特征分析结论,针对开源微处理器第五代精简指令集(reduced instruction set computing-Ⅴ,RISC-Ⅴ)的内核与存储体系结构的各种配置进行实验与分析,设计符合粒子输运模拟的专用加速器体系结构。

2.1　流水线设计

根据第 1 节程序特征分析得到的结论,粒子跟踪阶段的运行时间占比最高,因此,对函数 cycleTracking( ) 的代码进行了分析。发现它指令规模较小,没有包含复杂的诸如向量操作的指令,指令类型少,并行度高,因此基于标量 RISC-Ⅴ指令集流水线架构,专用加速器内核采用轻量化的小核设计方案,通过较小的面积和功耗获得相对可观的性能,并且能够满足未来众核体系结构的需求,发掘粒子输运程序的高并行性。

在 gem5 中,MinorCPU 是顺序执行的 CPU 模型,有 4 级浅流水线以及可配置的数据结构和执行操作,且兼容 RISC-Ⅴ指令集,因此采用 MinorCPU 模型作为专用加速器的流水线基础架构,图 2 给出了其流水线示意图,其中取指 2 到取指 1 之间有一个反向缓冲器支持分支预测。

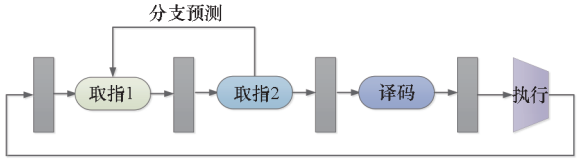

图 2　专用加速器的流水线示意图

Fig. 2　Schematic diagram of the specific accelerator pipeline

1)取指 1:取指操作分为 2 段,其中取指 1 段从指令 Cache 中取出 Cache 行,并将其传输到取指 2 段。通过配置一次读取指令的数目、取指延迟和分支预测反向路径的延迟进行定制设计。

2)取指 2:取指 2 段在接收到指令 Cache 行后,将其存入缓冲器,拆分成独立的指令后,发送到译码段。同时完成分支预测操作,并将分支预测结果返回到取指 1 段。根据程序特征,本段的分支预测部件是流水线设计的重点。

3)译码:译码段将指令解析为微操作。在定制设计中,主要配置输入缓冲器的大小、译码段的延迟和译码宽度。

4)执行:执行段主要完成指令微操作的执行和访存操作。定制设计包括执行段的输入宽度、

输入缓冲器大小和访存发送的操作数。

除流水线架构设计之外,流水线的频率也是影响其性能的重要因素。因此,在 gem5 中实现上述流水线设计,在 0.5 ~ 2.5 GHz 不同的频率下运行粒子输运模拟程序,进行了性能统计与分析。根据实验结果,在步长都为 0.5 GHz 的情况下,当频率在 1 ~ 2 GHz 时,程序运行时间减小的斜率更大。而当频率超过 2 GHz 时,频率继续提升会加深流水线,设计难度会急剧增加,流水线栈也极易成为关键路径,并且由于集成电路的功耗是随主频的提高而增长,因此提升主频所获得的收益也会越来越小。综上所述,专用加速器内核设计应采用 1 ~ 2 GHz 的主频。

2.2 分支预测部件设计

根据第 1 节程序特征分析得到的结论,粒子输运程序中存在大量随机性突出的分支操作,且主要集中于 cycleTracking( ) 函数,占比超过 96%,因此如何优化分支操作是提升程序执行性能的关键因素。分支预测是目前 CPU 广泛采用的一项重要动态执行技术,目的是为了提高 CPU 的指令执行效率,避免因分支指令导致的流水线停顿。动态分支预测技术是依据已执行指令的历史信息和当前分支指令的信息进行综合预测的方法,以更优的预测率占据了处理器设计的主流。

在专用加速器内核中分别集成了 4 种动态分支预测器:Bi-Mode、Tournament、标签几何历史长度预测器(tagged geometric history length predictor, TAGE)、多视角感知预测器(multiperspective perceptron predictor with TAGE, MPPwT)。在 gem5 上运行程序进行了实验,对加速器内核的程序运行时间和分支目标缓存(branch target buffer, BTB)的命中率进行了统计与分析。

通过实验发现,Bi-Mode、Tournament、TAGE 分支预测器在 Quicksilver 程序运行时间和 BTB 命中率方面表现相差不大,但都比多视角感知预测器更优;其中 TAGE 的运行时间最少,但仅比 Bi-Mode 快了 3%。Bi-Mode 预测器是一种相对简单的动态分支预测技术,其结构如图 3 所示。Bi-Mode 的模式历史表(pattern history table, PHT)分成了两个部分,根据程序运行状态来动态选择合适的一个表来进行预测,该方法可以保留基于全局历史的预测优点,同时减少破坏性混叠,提高预测的准确性。因此,综合考虑分支预测器的设计实现复杂度与硬件开销,专用加速器中分支预测部件采用分支预测效果较好且结构最简单的 Bi-Mode 分支预测器。

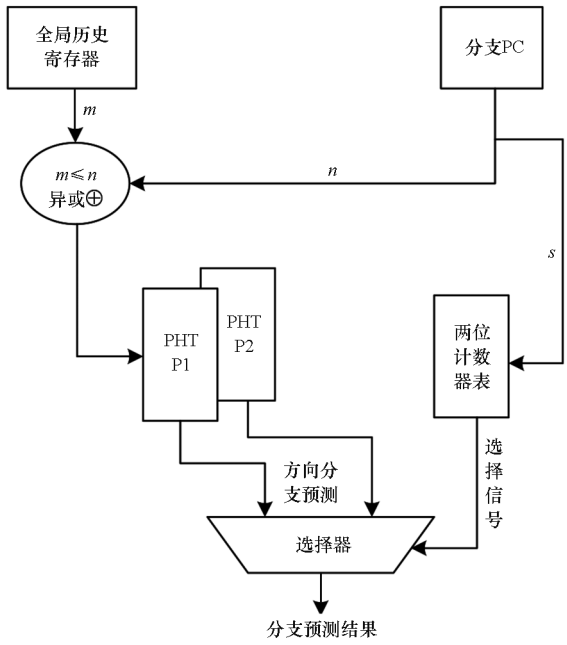

图 3 专用加速器的 Bi-Mode 分支预测器

Fig. 3 Bi-Mode branch predictor of the specific accelerator

2.3 多级 Cache 设计

根据第 1 节程序分析的结论,粒子输运程序的访存需求集中,cycleInit( ) 和 cycleFinalize( ) 的瞬时带宽更高,表明程序在初始化粒子库前后的访存需求大,因此对于 Cache 容量以及主存带宽的需求较高,以免出现访存瓶颈。缓存(Cache)是影响处理器程序执行性能的关键因素,因此对专用加速器的多级 Cache 架构进行设计。由于加速器内核是基于开源指令集 RISC-V 的轻量化内核设计,因此 Cache 层次采用 2 级 Cache 架构,并且限制容量,避免过大的存储器面积导致芯片面积增加,功耗提升。一级 Cache 分为 L1 数据 Cache 和 L1 指令 Cache,且最大容量分别定义为 64 KB 和 128 KB;而二级 Cache 为指令与数据共享,最大容量定义为 2 MB,L2 Cache 与主存进行数据交换,专用加速器的存储层次结构如图 4 所示。

首先,对加速器中的 L1 指令 Cache 的容量进行设计。加速器其他部件都相同,仅将 L1 指令 Cache 容量设置为 16 KB、32 KB 和 64 KB。在 gem5 中运行粒子输运程序,对运行时间进行了统计,同时对程序的三个主要函数 cycleInit( )、cycleTracking( )、cycleFinalize( ) 的 L1 指令 Cache 的失效率进行统计。根据实验结果,三个函数的 L1 指令 Cache 的失效率都低于 0.4%,命中率很高,但是 64 KB 的 L1 指令 Cache 的失效率降低曲

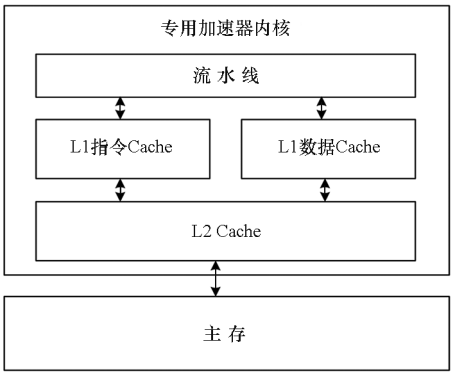

图 4　专用加速器的存储层次结构

Fig. 4　Memory hierarchy of the specific accelerator

线的斜率更高。因此,面向粒子输运模拟的专用加速器选择 64 KB 的 L1 指令 Cache。

其次,对专用加速器中的 L1 数据 Cache 容量进行设计。加速器其他部件都保持不变,将 L1 数据 Cache 容量设置为 16 KB、32 KB、64 KB 和 128 KB,在 gem5 中运行 Quicksilver 程序,对运行时间和 Cache 失效率进行了统计。根据实验结果,随着 L1 数据 Cache 的容量提升,程序运行时间明显减小。并且更大容量的 L1 数据 Cache 也能降低 cycleTracking( ) 函数的失效率。因此,专用加速器选择 128 KB 的 L1 数据 Cache。

最后,对专用加速器的 L2 Cache 进行设计。在 gem5 中,保持加速器其他部件相同,对 256 KB、512 KB、1 MB、2 MB 容量的 L2 Cache 进行实验,统计了粒子输运程序的运行时间和三个主要函数的失效率。从实验结果看,程序运行时间随 L2 Cache 容量的增加而明显地减少。并且随着 L2 Cache 的容量增加,三个函数的失效率都单调下降。综上所述,在面向粒子输运模拟的专用加速器中使用 2 MB 容量的 L2 Cache 设计。

2.4　主存设计

在架构设计中,主存的容量和访问速度也是影响程序执行性能的关键因素。在专用加速器主存容量设计中,对容量为 512 MB、1 GB、2 GB、4 GB的 DDR4_2400_8x8 内存芯片进行了程序性能测试,粒子数规模为 100 000。根据实验结果发现,在改变其主存容量大小时,程序运行时间并未发生改变,表明在当前粒子规模下,程序所需内存应小于 512 MB。在体系结构设计时,需要结合粒子规模数以及整个计算机系统成本、功耗与复杂度等因素,并基于专用加速器的轻量化设计思路,加速器设计选择 512 MB 的主存容量。

主存的访问速度也是影响粒子输运程序执行效率的重要因素。根据应用广泛程度,选取 LPDDR5_6400_1x16_8B_BL32、DDR4_2400_8x8、DDR4_2400_16x4 与 DDR4_2400_4x16 四种主存类型进行实验,其中三种 DDR4 的交换带宽和主频都相同,仅是数据线与地址线的位数不同。

将容量均设置为 512 MB,专用加速器的主存分别配置为这四种类型进行实验,统计了程序的运行时间。根据实验结果,三种 DDR4 内存类型的运行时间明显低于 LPDDR5;而三种交换带宽相同的 DDR4 内存的运行时间总体差异不大,其中 DDR4_2400_16x4 的运行时间最小。因此,专用加速器的主存类型选择程序执行性能最优的 DDR4_2400_16x4。

3　实验评测与分析

在 gem5 中运行粒子输运程序模拟 100 000 个粒子的碰撞过程,对专用加速器各个主要部件的不同配置进行了性能测试,是完成加速器体系结构设计的基础。并且和主流 ARM 微处理器在性能、功耗与面积方面进行了实验对比与分析。

3.1　加速器主频实验结果与分析

将加速器的主频分别设置为 0.5 GHz、1 GHz、1.5 GHz、2 GHz、2.5 GHz 五个频率,对 Quicksilver 程序的运行时间进行了统计,得到如图 5所示的结果。

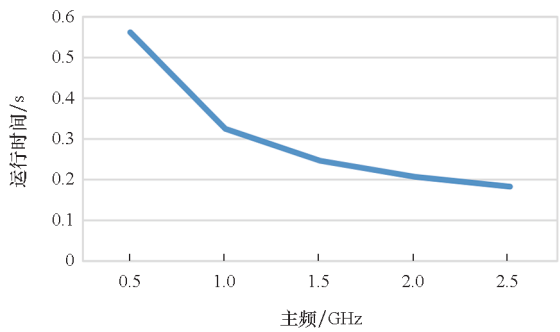

图 5　加速器在不同主频下程序运行时间

Fig. 5　Runtime at different main frequencies of accelerator

根据图 5,随着加速器主频的提升,程序的运行时间单调下降。由于频率增量相同,那么曲线减小的斜率越大,表明性能增加越显著。在主频为 1 ~ 2 GHz 时,程序运行时间减小的趋势明显;但随着频率增加,尤其是在 2 GHz 之后,程序运行时间减小的趋势变缓。

3.2　分支预测部件实验结果与分析

基于 gem5 进行粒子输运模拟时,将专用加速器内核集成 Bi-Mode、Tournament、TAGE、

MPPwT 分支预测器进行实验。对 Quicksilver 程序的执行时间进行了统计,得到图 6。其中,Bi-Mode、Tournament、TAGE 分支预测器在程序运行时间上相差不大,TAGE 的运行时间最优,但仅比 Bi-Mode 快了约 3.4%;MPPwT 最慢,比 TAGE 的运行时间多约 15.6%。

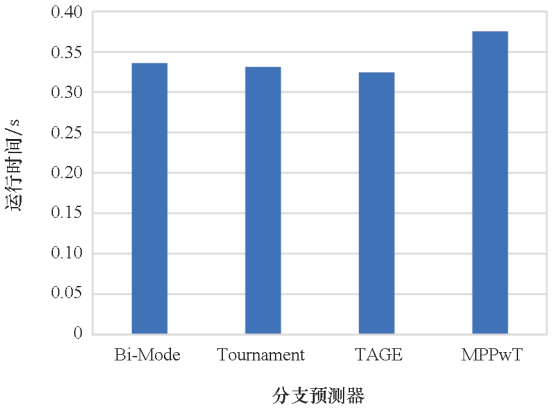

图 6 配置 4 种分支预测器的程序执行时间

Fig. 6 Program runtime at 4 branch predictors of accelerator

集成 4 种分支预测器,将分支目标缓冲器 BTB 命中率进行了统计,得到图 7。其中,Bi-Mode、Tournament、TAGE 分支预测器的 BTB 命中率相差不大,均在 99.5% 以上;MPPwT 的 BTB 命中率为 92.35%,与其他三个存在明显的差距。由于程序是对粒子运动的历史情况进行模拟,因此基于历史的分支预测器都取得了良好的效果。

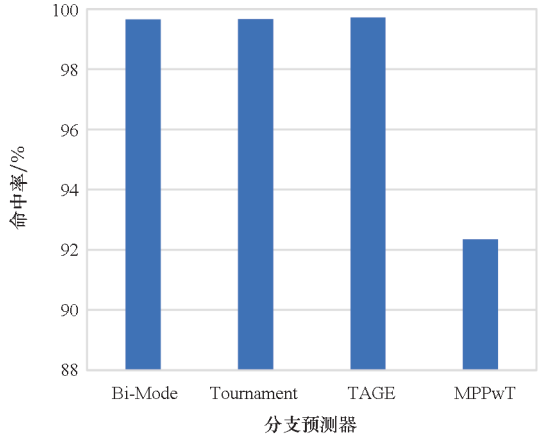

图 7 加速器采用四种分支预测器的 BTB 命中率

Fig. 7 BTB hit ratio of accelerator with 4 branch predictors

3.3 多级 Cache 实验结果与分析

3.3.1 L1 指令 Cache

在 gem5 中运行 Quicksilver 程序,通过实验对比专用加速器中不同容量 L1 指令 Cache 对程序运行性能的影响。将 L1 指令 Cache 容量设置为 16 KB、32 KB 和 64 KB,程序运行时间的统计结果如图 8 所示。其中,32 KB 的 L1 指令 Cache 比 16 KB 的运行时间少约 0.3%,64 KB 比 32 KB 少约 1.1%,性能提升的斜率增加。

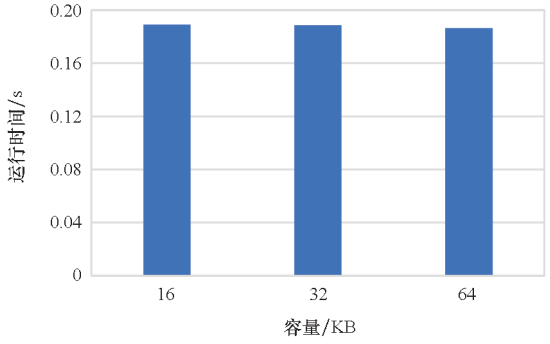

图 8 设置不同 L1 指令 Cache 容量时程序运行时间

Fig. 8 Program runtime under different capacities of L1 instruction Cache

并且对程序中三个主要函数运行时 L1 指令 Cache 失效率进行了统计,得到图 9。其中,三个函数的失效率都小于 0.40%,命中率高,主要原因是程序规模小,尤其 cycleInit( ) 和 cycleFinalize( ) 中的指令很少,而 cycleTracking( ) 中主要是循环操作,在程序运行初期载入 L1 指令 Cache 中后,指令失效的情况较少。并且,相比从 16 KB 到 32 KB,32 KB至64 KB 的 L1 指令 Cache 的失效率降低的曲线斜率更高,表明就目前测试的 L1 指令 Cache 容量而言,容量越大,程序运行的性能越高。

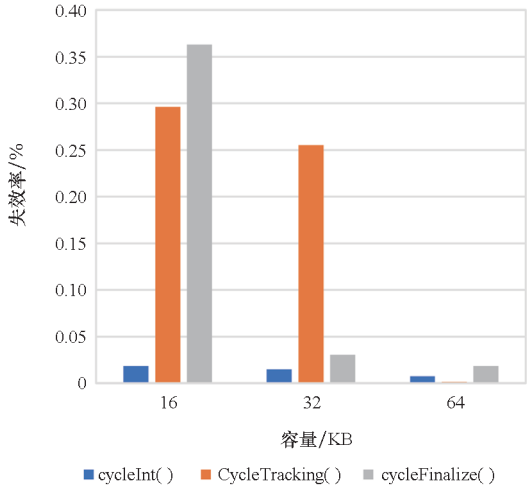

图 9 三种主要函数的 L1 指令 Cache 失效率

Fig. 9 Miss rate of L1 instruction Cache for three main functions

3.3.2 L1 数据 Cache

通过实验分析专用加速器中 L1 数据 Cache 容量对粒子输运程序性能的影响。将 L1 数据 Cache 的容量设置为 16 KB、32 KB、64 KB、128 KB,对 Quicksilver 程序运行时间进行统计,得

到图 10。其中,随着 L1 数据 Cache 的容量增加,程序运行时间明显减小。通过实验统计了三个函数的运行时间,cycleInit( ) 和 cycleFinalize( ) 函数的运行时间变化不大,而 cycleTracking( ) 函数的运行时间在减小,其原因在于程序的主要运算过程,例如粒子的追踪过程以及访问截面数据都是在 cycleTracking( ) 函数中实现,因此它对于 L1 数据 Cache 的容量更为敏感。并且统计了程序中三个主要函数运行时 L1 数据 Cache 的失效率,如图 11所示。其中,随着 L1 数据 Cache 容量增加,cycleInit( ) 和 cycleFinalize( ) 的失效率降低并不明显,而 cycleTracking( ) 下降更显著。主要原因是 cycleInit( ) 和 cycleFinalize( ) 仅是突发访存带宽要求高,访存总次数却低于 cycleTracking( )。而 L1 数据 Cache 的失效率都在 14% 以下,其原因在于访存过程主要是访问反应截面数据库,与模拟的粒子数规模直接相关。因此,更大容量的 L1 数据 Cache 可降低 cycleTracking( ) 函数的失效率,从而减少程序的整体运行时间。

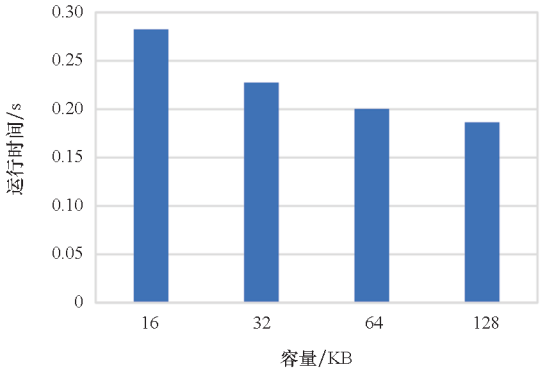

图 10　设置不同 L1 数据 Cache 容量时程序运行时间

Fig. 10　Program runtime under different capacities of L1 data Cache

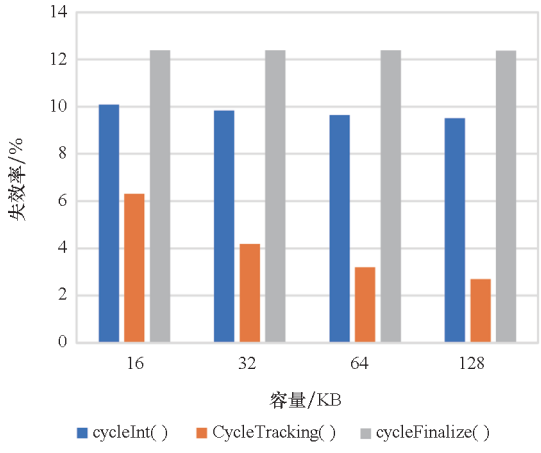

图 11　三个主要函数的 L1 数据 Cache 失效率

Fig. 11　Miss rate of L1 data Cache for three main functions

3.3.3　L2 Cache

通过实验分析专用加速器中 L2 Cache 容量对程序性能的影响。将 L2 Cache 设置为 256 KB、512 KB、1 MB、2 MB 容量进行测试,统计了程序的运行时间,得到图 12。其中,程序的总体运行时间随 L2 Cache 容量的增加而明显减少;通过进一步实验发现,程序中三个主要函数的运行时间也都单调降低。并且分别统计了三个函数运行时 L2 Cache 的失效率,如图 13 所示。其中,随着 L2 Cache 容量增加,三个函数的失效率都在下降,而 cycleTracking( ) 失效率降低的比例更高。其原因是,cycleInit( ) 和 cycleFinalize( ) 函数仅是完成粒子初始化和计算数据收集功能,对 L2 Cache 容量变化不敏感,而访问反应截面数据等大量访存操作主要集中于 cycleTracking( ) 函数,因此随着 L2 Cache 容量增加,该函数的失效率降低更加显著。

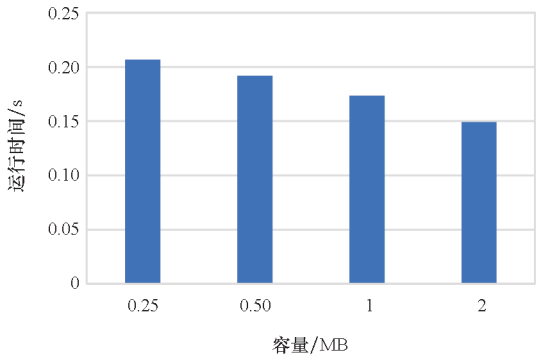

图 12　设置不同 L2 Cache 容量时程序运行时间

Fig. 12　Program runtime at different capacities of L2 Cache

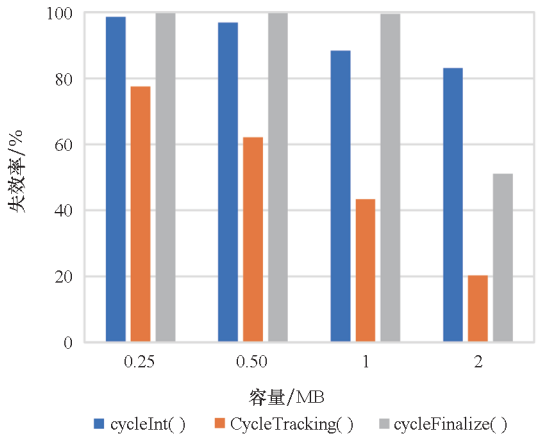

图 13　三个主要函数的 L2 Cache 失效率

Fig. 13　Miss rate of L2 Cache for three main functions

3.4　主存实验结果与分析

通过实验分析不同主存容量对粒子输运程序运行性能的影响,在 gem5 中分别对 512 MB、1 GB、2 GB、4 GB 容量的 DDR4_2400_8x8 内存进

行程序性能测试,统计了程序运行时间,如图 14 所示。其中,不同主存容量下程序的运行时间相同,其原因是在粒子规模数为 100 000,程序运行实际所需要的内存小于 512 MB,因此无法体现内存容量对加速器性能的影响。

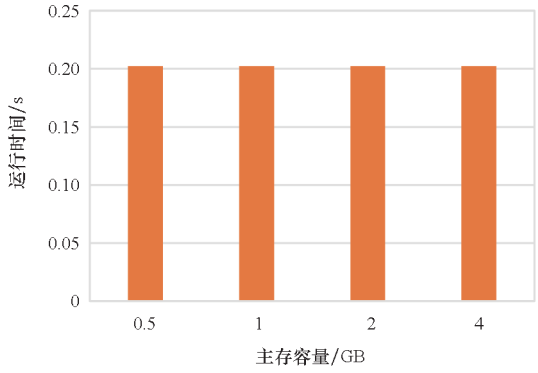

图 14 不同主存容量下程序运行时间

Fig. 14 Program runtime at different capacities of main memory

通过实验分析不同主存类型对程序性能的影响,选择 DDR4_2400_8x8、DDR4_2400_16x4、DDR4_2400_4x16、LPDDR5_6400_1x16_8B_BL32(LPDDR5)四种常见的主存芯片进行实验,在容量均为 512 MB 的情况下,统计程序的运行时间,得到图 15。其中,DDR4 内存的运行时间明显低于 LPDDR5。三种带宽相同的 DDR4 内存的运行时间差异不大,但 DDR4_2400_16x4 的运行时间最小。

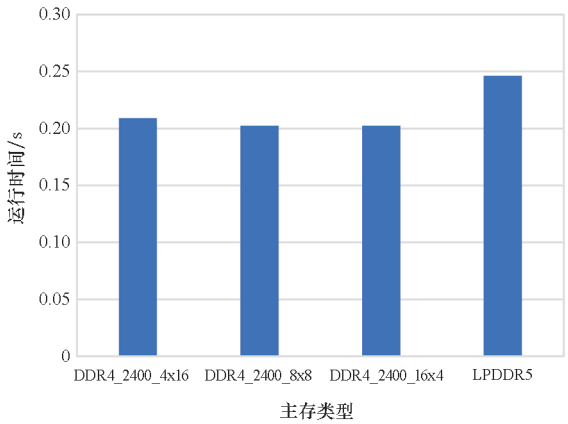

图 15 四种主存类型的程序运行时间

Fig. 15 Program runtime of 4 main memory types

3.5 与 ARM 微处理器的对比与分析

为了评估专用加速器的性能、面积和功耗,在 gem5 中构建了一个主流商用微处理器 ARM Cortex – A15 与其进行对比实验。Cortex – A15 广泛地应用于各种研究和商业领域,它包含一个 15 级高性能流水线、支持乱序执行的内核,兼容 ARMv7 指令集,支持数字信号处理和单指令多数据流指令扩展。

Cortex – A15 和专用加速器一种面各蒙特卡罗专用微处理器(a specific microprocessor architecture for Monte Carlo, ASMAMC)的存储层次配置相同,都采用了 64 KB 的 L1 指令 Cache 和 128 KB 的 L1 数据 Cache、2 MB 的共享 L2 Cache 以及 512 MB 的 DDR4 主存。在 gem5 中运行 Quicksilver 程序,粒子数规模为 100 000,统计 Cortex – A15 和专用加速器的运行时间,如图 16 和图 17 的右侧纵坐标所示。

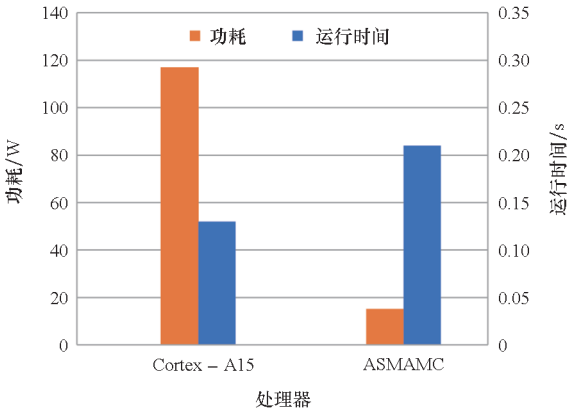

图 16 专用加速器与 Cortex – A15 性能与功耗对比

Fig. 16 Comparison of performance and power between the specific accelerator and Cortex – A15

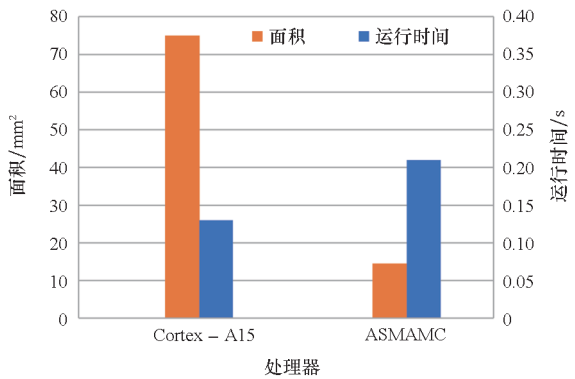

图 17 专用加速器与 Cortex – A15 性能与面积对比

Fig. 17 Comparison of performance and area between the specific accelerator and Cortex – A15

采用 McPAT 模拟器评估功耗和面积,它被广泛应用于处理器体系结构探索的研究领域。将程序在 gem5 中运行产生的统计信息与配置作为 McPAT 的输入文件,为专用加速器与 Cortex – A15 在 McPAT 中建模,其中工艺制程设置为 28 nm,分别得到 Cortex – A15 与专用加速器的功耗信息,如图 16 所示;Cortex – A15 与专用加速器的面积信息如图 17 所示。

从图 16 和图 17 得到,与 Cortex – A15 处理器相比,ASMAMC 的程序运行时间更长,性能下降约 39%;但是 ASMAMC 的功耗和面积则大幅降低。通过比较二者的性能/功耗,专用加速器获得 4.6 倍的优势;分析性能/面积,专用加速器得到 3.2 倍的优势。这主要是由于,Cortex – A15 采用乱序执行的大核架构,包含更长的流水线、更复杂的分支预测部件、多级分支预测表、强大的浮点处理单元以及双指令发射端口与执行逻辑。因此 Cortex – A15 能够获得更高的性能,但其代价是这些逻辑会占用更大的面积与更高的功耗。而专用加速器是基于 RISC-Ⅴ指令集,采用了顺序发射内核和简洁的分支预测部件,流水线较短,并且减少了指令发射宽度和浮点执行逻辑。所以,尽管专用加速器的程序执行性能出现了一定程度的下降,但是在面积和功耗方面获得了显著的优势,并且综合来看,单位面积的性能和单位功耗的性能方面都得到了数倍的提升。而且,专用加速器采用精简内核的设计,由于面积小、功耗低,更易于实现众核架构,更适合粒子输运程序粒子历史之间无相关性、并行度高、负载均衡、通信很少的特性。

4　结论

针对通用处理器无法适应粒子输运程序特征,难以在计算效率、功耗和面积方面取得均衡的问题,对面向粒子输运模拟的专用加速器体系结构展开研究。首先对求解粒子输运数值模拟的 MC 程序特征进行提取与分析。基于算法特征,对专用加速器体系结构进行定制设计。实验结果表明,与 ARM Cortex – A15 相比,专用加速器在同等功耗下可获得 4.6 倍的性能提升,在同等面积下可获得 3.2 倍的性能提升。

专用加速器的优点是易于定制化指令与部件,充分发挥专用部件的优势,获得较高的性能功耗比和性能面积比;内核采用通用处理器指令集,应用软件不需要移植;可将内核尽量简化,符合粒子输运程序并行化程度高的特点。但其缺点是通用性有所欠缺。因此,未来的研究工作将在专用体系结构基础上继续实验并开发更多应用的计算类型,例如偏微分方程求解、稀疏矩阵计算等。

参考文献(References)

[1] 林啸. 共振磁扰动对 EAST 中粒子轨道行为影响的数值模拟[D]. 杭州: 浙江大学, 2021.
LIN X. Simulation of particle orbit with resonant magnetic perturbations in EAST Tokamak[D]. Hangzhou: Zhejiang University, 2021. (in Chinese)

[2] 周培德, 薛小刚, 吴明宇, 等. 钠冷快堆先进建模与高性能粒子输运数值模拟进展[J]. 原子能科学技术, 2022, 56(11): 2395 – 2407.
ZHOU P D, XUE X G, WU M Y, et al. Progress in advanced modeling and numerical simulation of high-performance particle transport for sodium-cooled fast reactor[J]. Atomic Energy Science and Technology, 2022, 56(11): 2395 – 2407. (in Chinese)

[3] 张庆华, 刘志强, 郭典, 等. 基于 TPS 和蒙特卡罗计算的^{12}C 肿瘤放射治疗 In-beam PET 剂量监测研究[J]. 原子核物理评论, 2022, 39(3): 359 – 366.
ZHANG Q H, LIU Z Q, GUO D, et al. In-beam PET dose monitoring study of ^{12}C tumor radiotherapy based on TPS and Monte Carlo calculation[J]. Nuclear Physics Review, 2022, 39(3): 359 – 366. (in Chinese)

[4] 徐涵, 卓红斌, 杨晓虎, 等. 超热电子在稠密等离子体中输运的混合粒子模拟方法[J]. 计算物理, 2017, 34(5): 505 – 525.
XU H, ZHUO H B, YANG X H, et al. Hybrid particle-in-cell/fluid model for hot electron transport in dense plasmas[J]. Chinese Journal of Computational Physics, 2017, 34(5): 505 – 525. (in Chinese)

[5] LIU J, CHI L H, LIN W Q, et al. Parallel S_n sweep scheduling algorithm on unstructured grids for multigroup time-dependent particle transport equations[J]. Nuclear Science and Engineering, 2016, 184(4): 527 – 536.

[6] ALGUACIL J, SAUVAN P, JUAREZ R, et al. Assessment and optimization of MCNP memory management for detailed geometry of nuclear fusion facilities[J]. Fusion Engineering and Design, 2018, 136: 386 – 389.

[7] LEPPÄNEN J. Acceleration of fission source convergence in the Serpent 2 Monte Carlo code using a response matrix based solution for the initial source distribution[J]. Annals of Nuclear Energy, 2019, 128: 63 – 68.

[8] KOWALSKI M A, COSGROVE P M. Acceleration of surface tracking in Monte Carlo transport via distance caching[J]. Annals of Nuclear Energy, 2021, 152: 108002.

[9] ZHANG S, WANG Z, PENG Y, et al. Mapping of option pricing algorithms onto heterogeneous many-core architectures[J]. The Journal of Supercomputing, 2017, 73: 3715 – 3737.

[10] SWEEZY J E. A Monte Carlo volumetric-ray-casting estimator for global fluence tallies on GPUs[J]. Journal of Computational Physics, 2018, 372: 426 – 445.

[11] TITHI J J, PETRINI F, RICHARDS D F. Lessons learned from accelerating quicksilver on programmable integrated unified memory architecture (PIUMA) and how that′s different from CPU[C]// Proceedings of the High Performance Computing, 2021: 38 – 56.

[12] RICHARDS D F, BLEILE R C, BRANTLEY P S, et al. Quicksilver: a proxy app for the Monte Carlo transport code mercury[C]//Proceedings of the 2017 IEEE International

Conference on Cluster Computing (CLUSTER), 2017: 866 – 873.

[13] FU S Q, LI T J, ZHANG J M, et al. MiniMCTAD: minimalist Monte Carlo transport architecture design[C]// Proceedings of the 2021 IEEE International Conference on Parallel & Distributed Processing with Applications, 2021: 1 – 10.

[14] GORSHKOV A V, KIRILLIN M Y. Acceleration of Monte Carlo simulation of photon migration in complex heterogeneous media using Intel many-integrated core architecture[J]. Journal of Biomedical Optics, 2015, 20(8): 85002.

[15] ENDO F A, COUROUSSÉ D, CHARLES H P. Micro-architectural simulation of in-order and out-of-order ARM microprocessors with gem5[C]//Proceedings of the 2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV), 2014: 266 – 273.

[16] QURESHI Y M, SIMON W A, ZAPATER M, et al. Gem5-X: a many-core heterogeneous simulation platform for architectural exploration and optimization[J]. ACM Transactions on Architecture and Code Optimization, 2021, 18(4): 1 – 27.

[17] LI S, AHN J H, STRONG R D, et al. The McPAT framework for multicore and manycore architectures: simultaneously modeling power, area, and timing[J]. ACM Transactions on Architecture and Code Optimization, 2013, 10(1): 1 – 29.

[18] ENDO F A, COUROUSSÉ D, CHARLES H P, et al. Micro-architectural simulation of embedded core heterogeneity with gem5 and McPAT[C]//Proceedings of the 2015 Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools, 2015: 1 – 6.

[19] WALKER M, BISCHOFF S, DIESTELHORST S, et al. Hardware-validated CPU performance and energy modelling[C]//Proceedings of the 2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2018: 44 – 53.

[20] SEMAKIN A N. Simulation of a multi-core computer system in the gem5 simulator[C]//Proceedings of the AIP Conference, 2020.

[21] POZULP M, BECK B, BLEILE R, et al. Status of Mercury and imp: two Monte Carlo transport codes developed using shared infrastructure at Lawrence Livermore National Laboratory[J]. European Physics Journal-Nuclear Sciences & Technologies, 2024, 10: 19.1 – 19.7.