计算机科学
COMPUTER SCIENCE

面向粒子输运程序加速的体系结构设计

傅思清，黎铁军，张建民

引用本文

傅思清，黎铁军，张建民. 面向粒子输运程序加速的体系结构设计[J]. 计算机科学, 2022, 49(6): 81-88.

FU Si-qing, LI Tie-jun, ZHANG Jian-min. Architecture Design for Particle Transport Code Acceleration[J]. Computer Science, 2022, 49(6): 81-88.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210600179

面向粒子输运程序加速的体系结构设计

傅思清　黎铁军　张建民
国防科技大学计算机学院　长沙 410073
(fusiqingnudt@nudt.edu.cn)

摘　要　粒子输运的随机模拟方法通常用于求解大量运动状态中粒子的特征量。粒子输运问题广泛出现在医学、天体物理和核物理领域,当前粒子输运随机模拟求解方法的主要挑战是计算机能够支撑的模拟样本数、模拟时间尺度与研究人员研究实际问题的需求之间的差距。处理器性能的发展随着工艺尺寸进步的停滞进入了新的历史阶段,复杂的片上结构的集成已经不符合现今的要求。面向粒子输运程序,文中开展了一系列体系结构设计工作,通过分析和利用程序的并行性和访存特点,设计了精简内核和可重配置缓存来加速程序。通过模拟器验证,文中提出的体系结构相比传统乱序架构获得了4.45倍性能功耗比优势以及2.78倍性能面积比优势,这为进一步研究大规模众核粒子输运加速器奠定了基础。

关键词:粒子输运;蒙特卡洛;体系结构;加速器;流水线
中图法分类号　TP302

Architecture Design for Particle Transport Code Acceleration

FU Si-qing,LI Tie-jun and ZHANG Jian-min
School of Computer,National University of Defense Technology,Changsha 410073,China

Abstract　The stochastic simulation method of particle transport is usually used to solve the characteristic quantity of a large number of moving particles. Particle transport problems are widely found in the fields of medicine,astrophysics and nuclear physics. The main challenge of current stochastic simulation methods for particle transport is the gap between the number of simulation samples supported by computers,the simulation timescale,and researchers' needs to study practical problems. Since the development of processor performance has entered a new historical stage with the stagnation of process size progress,the integration of complex on-chip structures no longer meets the current requirements. For particle transport programs,this paper carries out a series of architecture design works. By analyzing and using the parallelism and access characteristics of the program,simplified kernel and reconfigurable cache are designed to speed up the program. Experiments show that compared to the traditional architecture composed of multiple out-of-order cores,this architecture can obtain more than 4.5x in performance per watt and 2.78x in performance per area,which lays a foundation for the further study of large-scale many-nucleus particle transport accelerator.

Keywords　Particle transport,Monte Carlo,Architecture,Accelerator,Pipeline

1　引言

粒子输运理论源于Boltzman方程,它是由Boltman导出的反映微观粒子在介质中迁移守恒关系的微分-积分方程。随着中子物理学和计算机科学的发展,人们开始使用计算机求解粒子输运方程,以解决粒子运动的相关问题,也使得粒子输运方程的数值求解方法得到了广泛的发展运用。例如,在医学领域[1],通过研究光子、电子以及质子等微观粒子在人体组织中的输运来进行癌症的放射性治疗;在核反应堆设计中[2],通过研究中子输运过程来计算堆芯的中子分布以进行屏蔽设计;在天体物理学中[3],通过研究天体中的辐射输运来推断天体的结构、组成以及产生辐射能源的反应。

粒子输运的数值求解方法分为确定性方法和随机模拟方法,后者是目前最广泛使用的求解方法。粒子输运的随机模拟方法即蒙特卡洛方法(Monte Carlo,MC),其在求解问题时需要建立对系统中每个粒子的跟踪,在时间变化中模拟每个粒子随机的运动和反应,通过对大量粒子运动历史的跟踪,抽样出充分的随机实验值,并以此采用统计方法得出相应特征值的估计量,并将其作为问题的解。

粒子输运的随机模拟求解涉及大量跟踪和计算,但随着计算机技术的快速发展,MC方法特有的高并行性得到了利用。在运动历史的跟踪中大量粒子的运动相互独立,相互

到稿日期:2021-06-22　返修日期:2022-02-15
基金项目:国家重点研发计划(2018YFB0204301)
This work was supported by the National Key Research and Development Program of China(2018YFB0204301).
通信作者:黎铁军(tjli@nudt.edu.cn)

之间的影响非常小，为线程间的并行提供了机会。当前 MC 方法的主要挑战是计算机能够支撑的模拟样本数、模拟时间尺度以及研究人员研究实际问题的需求之间的差距。早期的研究[4-7]大多是针对 MC 方法并行化的实现，之后[8-10]的研究开始致力于使用 FPGA 加速 MC 程序，但 FPGA 的高性能背后是以过高的专用性为代价的。

随着摩尔定律的终结或者说放缓[11]，为了在满足功耗要求的同时提高性能，由众多功能单一的内核构成的众核处理器成为了新的选择，这些单一内核依托所支持的应用服务进行定制，频率较低，数量众多，又能保证良好的性能。例如，Intel MIC 协处理器[12]由简化的通用 X86 核构成，其核数为 50～62；与 MIC 具有同等级别双精度浮点峰值性能的 NVIDIA Tesla K20C GPU[13]由数量众多的简单计算核构成，核数达到了 2496；日本推出的 PEZY-SC[14]由 2 个 ARM 控制核和 1024 个 RISC 逻辑核构成；而 Google 公司推出的面向神经网络加速的 TPU 的运算部件包含 65536 个 8 位矩阵乘单元阵列，它们都根据各自的支撑对象定制了相应的简化内核。

与 MC 方法加速相关的研究由来已久，由于 MC 方法在不同应用上具有相似的本质，这里的相关工作不局限于对粒子输运的研究，任何使用 MC 方法的模拟包括动力学模拟和期权定价等都考虑在内。一批国内外的工作通过研究算法和体系结构间的相互适应来加速 MC 应用。Kowalski 等[15]对粒子输运中的表面跟踪算法进行了修改，优化了距离数据的缓存策略，使它们能在附近的跟踪中被反复使用，使得运行时间得到 7%～20%的优化，但这一工作仍然是利用现有硬件特性而进行的加速研究，并未对体系结构进行调整。Brugger 等[16]的应用目标是期权定价应用，他们在 CPU 和 FPGA 异构系统上对算法不同部分的需求进行在线动态重配置，这种混合架构比高端英特尔 CPU 快了近 2 个数量级。类似的工作包括 Zhang 等[17]的研究，而他们通过 GPU 和 Xeon Phi 异构得到优于基于 GPU 计算期权定价的解决方案。国内方面，Li 等[18]优化了基于 CPU/Matrix2000 异构协同计算的并行程序，在天河 2A 系统上测试后拓展到 45 万核时的并行效率保持在 5 万核时的 22.54%。但总体来说，国内外从体系结构角度研究粒子输运加速的相关工作还并不丰富。

本文针对粒子输运的随机模拟应用，从体系结构角度探索和定制体系结构，为提出高性能低功耗众核领域专用的处理器提供了帮助，而之前从体系结构角度出发进行的蒙特卡洛加速研究较少。本文的主要贡献如下：

(1)描述了粒子输运的随机模拟程序的程序特征，能够有效帮助设计高效硬件与其匹配。

(2)设计了在性能功耗比和性能面积比上具有优势的专用体系结构，为进一步开发大规模专用计算机提供了指导。

本文第 2 节主要介绍了文章的背景、动机以及一些相关工作；第 3 节主要描述了体系结构设计，包括用作对比的传统的多核处理器(Chip Multiprocessors，CMP)，以及本文的粒子输运专用体系结构；第 4 节主要描述了实验配置；第 5 节进行了实验并对结果进行了分析；最后总结全文。

2 蒙特卡洛粒子输运程序

本文面向的应用为 MC 方法求解的粒子输运程序，本节通过介绍 MC 方法和粒子输运程序的发展，结合处理器性能发展的现状，来介绍文章的基本背景和相关领域的研究现状。

2.1 蒙特卡罗方法

MC 方法又称随机模拟法，由于其计算方法的随机性，得名于摩纳哥著名赌城蒙特卡洛，可以说其是在计算机对粒子行为进行随机模拟的过程中发展起来的一种计算方法。MC 方法通过随机抽样模拟求解数值积分和数值微分，其特点在于用随机数进行抽样，用统计平均给出估计量均值。

随机模拟的基本思想可以参考投针求 π 的实验。在图 1 所示的单位正方形内均匀投针，投针的过程即进行随机模拟的过程。接下来进行统计抽样，我们跟踪针投过的模拟事件后的结果，从而得到命中圆的数量，则针命中内圆的概率为：

$$P=\frac{Area\ of\ inscribed\ circle}{Area\ of\ units\ quare}=\frac{\pi}{4}\sim\frac{M}{N} \tag{1}$$

其中，N 为投针总数，M 为命中圆的针数。显然 N 越大，π 的值越精确。上述实验过程在计算机中的实现依赖于两组随机数。在 x 轴(0,1)区间和 y 轴(0,1)区间使用伪随机数发生器产生随机数 γ_1 和 γ_2，判断不等式(2)是否成立。

$$(\gamma_1-1/2)^2+(\gamma_2-1/2)^2\leqslant\frac{1}{4} \tag{2}$$

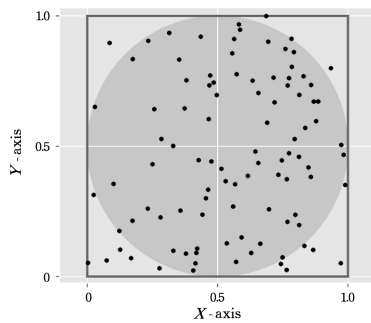


图 1 MC 方法投针求 π

Fig. 1 Schematic diagram of π calculation by MC method

通过算法 1 在计算机中模拟这个简单的 MC 方法。

算法 1 The Monte Carlo method to find π

Input: ϵ_0, ΔN

Output: π

1. initialize: Set N=0, M=0, n=1
2. while $\epsilon=\sqrt{\frac{N-M}{NM}}\leqslant\epsilon_0$ do
3. N=N+ΔN
4. while n≤N do
5. Generating random numbers λ_1, λ_2
6. if $(\lambda_1-1/2)^2+(\lambda_2-1/2)^2\leqslant\frac{1}{4}$ then N=N+1
7. end if
8. n=n+1
9. end while
10. π=4M/N
11. end while
12. Print π

前文只讨论了估计量的求解，而估计量来自随机数，又涉及随机数的产生和抽样。本文不讨论随机数的产生过程，只通过随机数抽样示范 MC 方法的基本思想。由于 MC 方法擅

于求解一类不便于直接确切求解的问题,其在金融工程学、宏观经济学、生物医学和计算物理学中均有广泛应用。本文以计算物理学中的粒子输运应用为负载设计适用的体系结构,但 MC 方法的计算特点决定了这类问题有相似的特性表征。

2.2 粒子输运程序

使用 MC 方法求解粒子输运问题,建立粒子库,在一定几何尺度内跟踪所有粒子,并对其依概率随机执行各种反应和分支事件,并在达到时间步长终点时抽样统计值,从而得到需要的特征值。

例如,使能量为 E 的中子与 235U 核发生反应,如果碰撞后可能发生的核反应包括裂变(f)、散射(el)和吸收(c),则其概率反应分别为 $P_f=\Sigma_f/\Sigma_t$,$P_{el}=\Sigma_{el}/\Sigma_t$,$P_c=\Sigma_c/\Sigma_t$。通过算法 2 确定碰撞后发生的反应类型。

算法 2 Monte Carlo method to determine the reaction type for a collision

```
Input:P_el,P_c,E
Output:Event
1. Generating random numbers ξ
2. P_el(E)→P
3. if ξ<P then Event=scattering
4. else
5.    P+P_c(E)→P
6.    if ξ<P then Event=absorbing
7.    else
8.       Event=Fissioning
9.    end if
10. end if
11. Print Event
```

对于每一个粒子,计算机并行地执行跟踪,MC 方法求解粒子输运过程忠实地反映了物理过程,如果程序细致地体现了各种分支事件,且依赖可靠的横截面数据,则随机模拟求解的数值将比较可靠。

本文使用的负载程序为 Quicksilver[19],它是 Mercury 的代理程序,开发目的是作为原型测试和研究 Mercury 的潜在编程模型和设计选项。作为代理程序,Quicksilver 准确捕获父程序 Mercury 的控制流: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模式。

Quicksilver 在每个时间步长内依次执行 cycleInit(),cycleTracking()和 cycleFinalize()这 3 个函数,程序在 3 个阶段分别初始化粒子信息、跟踪粒子执行输运过程以及更新数据并释放缓存。cycleTracking()为主要的跟踪阶段,算法 3 给出了 cycleTracking()函数的结构,本文着重讨论的也是这部分的性能表现。函数并行地跟踪每个粒子,对它们执行不同事件,在跟踪一个粒子的过程中,首先调用 Segment_Outcome()函数,以决定进入的 3 种事件中的分支,然后分别执行穿越、碰撞或直接进入终结,最后根据事件中是否吸收逃逸来决定是否继续循环处理。在调用碰撞事件后,控制流随机进入吸收、散射、裂变事件,这里只简单介绍了 Quicksilver 程序的控制流和分支结构,不对具体代码块的硬件负载展开赘述。

算法 3 Basic structure of cycleTracking() function

```
Input:Particle bank
1. for all particles do
2.   while ! absorbed or ! incensus or ! escaped do
3.     segmentoutcome=Segment_Outcome()
4.     if segmentoutcome==facet then
5.       facet_event()
6.     end if
7.     if segmentoutcome==collision then
8.       collision_event()
9.     end if
10.    if segmentoutcome==census then
11.      census_event()
12.    end if
13.  end while
14. end for
```

2.3 粒子输运随机模拟求解的发展与性能机会

由于粒子输运问题研究的重要性以及 MC 方法的广泛应用,已经有相当多成熟通用的 MC 程序,如美国洛斯阿拉莫斯国家实验室研制的 MCNP 程序[20]以及全开源的 OpenMC 程序[21]。这些 MC 程序在各自面向的应用场景中建模粒子传输和横截面数据,精确模拟粒子输运过程。通过圆周率的生成模拟可以清晰地看到,每一次随机抽样的过程是相互独立的,这体现了 MC 程序天然的并行性。为了利用和提高 MC 程序的并行性,亦有相当的工作致力于 MC 程序并行化的优化。

Tyagi 等[4]将用于放射剂量计算的 MC 程序 DPM 使用 MPI 并行化,在 24 个 CPU 组成的簇上实现 23.07 的加速比。类似的早期成果还包括 Wagner 等[5]对 MCNP-4A 版本在 MPI 上的并行化。随着研究的不断深入,简单的并行化依然不能满足大规模粒子的模拟需要。Alguacil 等[6]对 MCNP 的访存特性进行了研究,评估了程序与几何相关的内存管理,确定了内存瓶颈的根源是编码对内存分配空间的高估,经过修改显著提升了代码性能。Anderson 等[7]在 Tesla K20 上探索 MC 程序的大规模并行,相比单个 Intel Xeon E5540 CPU 内核达到了 148 倍的提升。

除在通用 CPU 核和众核处理器上加速 MC 程序外,基于 FPGA 的加速也被广泛研究。Luu 等[8]将 PDT(Photo Dynamic Therapy,PDT)算法在 Stratix III FPGA 上进行了硬件实现,相比主频 3GHz 的 Intel Xeon 处理器上的软件计算,该硬件设计获得了 28 倍的计算加速比,而计算过程所消耗的功耗只是软件计算的 1/716。Whitton 等[9]也将 PDT 算法在 FPGA 上进行了硬件实现,并且获得了 20.7 倍的计算加速比。Gokhale 等[10]对热辐射算法在 Virtex IIFPGA 上进行了硬件实现,相比主频为 3GHz 的 Intel Xeon 处理器上的软件计算,该硬件设计获得了 10.4 倍的计算加速比。然而,FPGA 加速的高专用性并不适于为此构建昂贵的专用计算机,而从体系结构角度出发的 MC 程序加速研究比较缺乏。

后摩尔时代处理器性能提升已经不再从“底部空间”的器件小型化中收益,更多的好处将来源于软件、算法和硬件体系结构等“顶部空间”[22]。比较传统的做法是根据硬件来定制软件,如在算法上充分利用并行处理器和矢量单元,而对给定算法定制加速硬件的进展并不大。在这方面可以简化硬件结

构，更有效地分配晶体管，如增加并行单元数量，以提高高并行问题的效率，再如领域专用化，删去不需要的处理器功能，添加更多自定义的处理器加速单元。本文将在体系结构层次为粒子输运随机模拟程序的性能提升寻找机会。

3 体系结构设计

本节介绍了本文相关的体系结构，其中包括目前传统的以乱序内核为主要部分的传统多核体系结构，以及本文设计的针对 MC 粒子输运程序进行了优化和定制的专用体系结构。

3.1 通用多核体系结构

图 2 给出了一个传统的多核体系结构模型，整个体系结构的中心是 4 个乱序通用内核（General-purpose Processing Elements，GPEs），复杂的内核通过乱序发射获得了更好的 CPI，但也付出了面积和功耗的代价。4 个乱序内核独占各自的一级缓存，并共享一个二级缓存。在更大规模的体系结构中，它们可以共同组成一个簇，并通过交叉开关或其他互联结构连接到 DRAM 内存。MC 粒子输运应用程序对通信的依赖相对较小，因此在体系结构规模不大的情况下，本文直接将最后一级缓存（Last Level Cache，LLC）连接到 DRAM 内存。

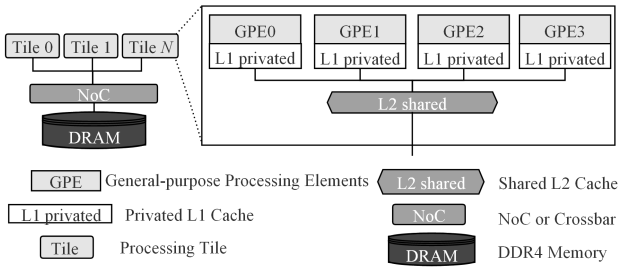


图 2 通用多核体系结构

Fig. 2 General multi-core architecture

3.2 粒子输运专用体系结构

我们在 gem5[23] 模拟器中建模简化的内核。MinorCPU 作为一个全面而详细的顺序 CPU 模型，适合作为建模的基础，在此之上我们建模完整的体系结构，如图 3 所示。相比通用多核体系结构的 GPE，我们为专用架构设计了蒙特卡洛专用内核（Monte Carlo Processors，MCPs），并根据粒子输运不同阶段的需求设计了使用数据旁路对二级缓存的越过机制。内核设计将在本节讨论，二级缓存旁路及效果将在实验部分详细描述。

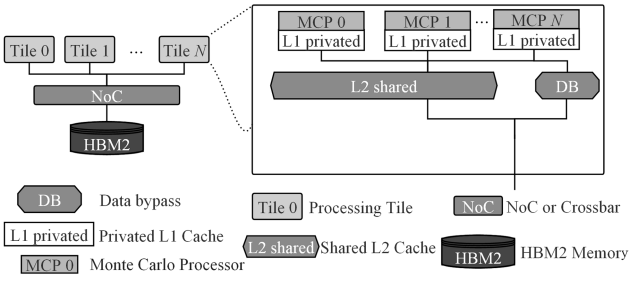


图 3 粒子输运专用体系结构

Fig. 3 Special architecture for particle transport

3.2.1 流水线

顺序流水线提供了 4 个指令阶段，即 Fetch1，Fetch2，Decode 和 Execute，如图 4 所示。表 1 列出了流水线各阶段的一些重要参数。

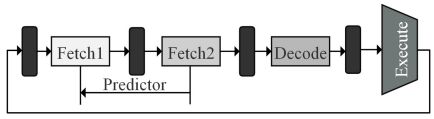


图 4 四级指令流水线

Fig. 4 Pipeline of four stage instruction

表 1 流水线中的重要参数

Table 1 Important parameters in pipeline

Instruction stage	Parameter	Default Value
Fetch1	fetch1FetchLimit	1
	fetch1LineSnapWidth	0
	fetch1LineWidth	0
	fetch1ToFetch2ForwardDelay	1
	fetch1ToFetch2BackwardDelay	0
Fetch2	fetch2InputBufferSize	2
	fetch2ToDecodeForwardDelay	1
Decode	decodeInputBufferSize	3
	decodeToExecuteForwardDelay	1
	decodeInputWidth	1
Execute	executeInputWidth	1
	executeInputBufferSize	7
	executeIssueLimit	1
	executeMemoryIssueLimit	1
	executeCommitLimit	1
	executeMemoryCommitLimit	1

流水线在 Fetch1 阶段从 Icache 提取缓存行，并将其传递到 Fetch2 分解为指令。分支预测阶段在 Fetch2 实现，预测的分支指令反向输出到 Fetch1。由于蒙特卡洛输运涉及大量随机分支过程，因此不同种类的分支预测器不会有明显的性能差异，但考虑到粒子循环跟踪过程中依然依赖分支预测部件，我们使用简单的双模预测器就可以在设计上满足性能和面积的折中要求。Decode 阶段将指令译码成操作，并将其在 Execute 阶段执行，后者更多地涉及内存访问和功能单元。在流水阶段设计上，我们着重关注的是各级输入宽度以及缓冲区大小。为了精简单线程的资源占用，我们使用了较小的输入宽度和缓冲区大小，以最大限度地在保持功能的同时获得面积和功耗的收益。另外，对于涉及访存的 Execute 阶段，每个周期执行的发射的指令数、可发射访存指令数、可提交指令数以及内存引用数都下降到可接受的最低水平，因为粒子随机跟踪过程不是访存密集的，满足线程零碎的内存访问即可。

MinorCPU 的流水结构分为 4 个指令阶段，而流水深度通过对指令延迟的比例调整模拟实现。现代处理器往往具有极深的流水级数，这样做的好处是可以追求更高的主频，从而追求高性能，但带来的负面作用是面积和功耗上的过度开销。极简主义的体系结构面向单核的小面积低功耗，因此采取最浅的流水级数，保持 4 级流水执行指令阶段，通过浅流水获得功耗收益。

3.2.2 指令执行部件

Gem5 中支持配置多种功能单元，它们支持各自的操作类，用于执行相应的指令。同时各类操作的操作延迟也可以相应调整。图 5 给出了几个 gem5 中的内核模型使用功能单

元情况。Cortex-A15 作为高性能大核，对发射时容易遇到部件占用的功能部件做了冗余配置，配置了更多的 SimleALU 和 FP 部件。而 Cortex-A7 内核更强调低功耗，因此相比大核减少了冗余的 FP 部件。相比两种公版架构，我们进一步减少了蒙特卡洛输运设计的专用精简内核的执行单元，完全不进行冗余配置。虽然这可能带来发射等待等性能损耗，但我们期待更大的面积、功耗收益，这一点也将在实验中得到证实。

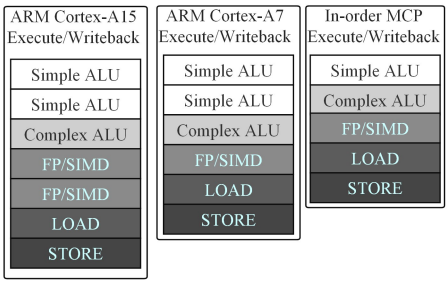


图 5 gem5 中几种内核模型使用功能单元的情况

Fig. 5 Some core model using functional unit in gem5

另外，对于 ARMv8 指令集而言，相当的指令在专用领域计算机中是不需要的，裁剪部分冗余指令同样会带来面积上的显著收益。但在性能模拟的过程中，gem5 只在功能上模拟了各种指令的执行，并不设涉及面积建模，而 McPAT[24] 也没有相应机制予以支持。因此，本文特别指出这部分内容可以成为一个重要的提升点，但在本文实验中并不体现，故在结合指令集简化后，本文的结论只会更加优异。

3.2.3 *存储层次结构*

缓存大小占据了片上面积的重要部分，同时也显著地影响了性能表现。在实验中两级缓存在粒子跟踪阶段体现出了较高的命中率，显然二级缓存对性能起到了极大的提升作用，因此必须在粒子跟踪阶段保持适配的二级缓存层次。

通常缓存可以配置预取器来提高对局部性的利用，从细粒度上看，MC 输运的执行随机分支较多，一级数据预取并不能涵盖这些分支，因此不适合配置预取器，但二级预取由于粒子的循环跟踪，应当是必要的。因此，对一级数据缓存不配置预取器，对二级缓存配置预取器。

此外，对于粒子输运程序的不同阶段，对于缓存的需求实际上是不同的。我们将在实验中详细描述 cycleInit()和 cycleFinalize()阶段与 cycleTracking()函数不同的访存特性，并讨论何时使用数据旁路越过二级缓存。因此，在存储层次上我们支持使用数据旁路来将一级缓存直接连接到内存。

在内存方面，传统体系结构和专用体系结构分别选用了 DDR4_2400_16x4 和我们在 gem5 中建模的 HBM2_2000_4H_1x128 模型[25]，后者使用 8 个内存控制器模拟 8 通道，满足 MC 粒子输运在初始化和终结阶段的高带宽访存行为对内存性能提出的要求。

4 实验设置

本节介绍了实验相应的配置，由底至上包括主机硬件配置、gem5 模拟器配置、模拟器开启的系统配置以及负载输入等。

4.1 模拟器和硬件设置

为了评估本文的专用体系结构的面积和运行 MC 输运程序的性能功耗表现，我们使用 gem5 模拟器(版本 20.01)结合 McPAT 模拟器(版本 1.3)建模了整个系统，使用 gem5 统计数据和配置为依据生成 McPAT 的输入文件。运行模拟器的机器使用的处理器为 AMD Ryzen5 PRO 2600 Six-Core Processor，机器运行操作系统为 Ubuntu 20.04 LTS。在实验中主要使用 FS 模式运行，FS 模式启动的镜像安装操作系统版本为 Ubuntu16.04 LTS，Quicksilver 程序使用 qemu 在镜像中编译，编译选项使用 openMP 并行。两种体系结构共同的建模参数按表 2 中指定执行。对于通用多核体系结构，我们使用模拟器中的 Cortex-A15 高性能乱序内核模型，性能和功耗模拟使用 28 nm 工艺尺寸。

表 2 模拟器建模体系结构的重要配置

Table 2 Important configuration of the architecture modeled by simulator

Unite	Configuration
Processing Element	2.2 GHz click speed，4 PEs per tile active in experiment
L1 cache	32 kB，cache line size 64 icache：2-way set-associative，privated，dcache：4-way set-associative，privated
L2 cache	256 kB or 512 kB each tile，8-way set-associative，shared replacement policy is RandomRP
Main Memory	4 GB

4.2 负载输入设置

Quicksilver 支持从命令行或者输入文件接受输入参数，MC 输运问题十分多样，因此 Quicksilver 也尽可能灵活地支持丰富的输入选项。表 3 列出了一些重要的输入选项和我们使用的输入规模值，其中 dt 表示时间步长，也就是一次模拟直到终结的时间尺度。过短的数值将导致大量粒子在跟踪时直接进入终结。X,Y,Z 代表了模拟的几何尺寸，Quicksilver 程序将粒子输运的几何范围简化成一个平行六面体，对几何尺度和粒子数的适当规划能够在尽可能快速运行模拟器的同时得到具有代表性的结果。模拟使用材料的截面数据由吸收、散射以及裂变各自的比例和横截面数据提供。截面比例反映了几种碰撞事件发生的概率，而 $nuBar$ 为裂变瞬发中子数，表示裂变事件产生粒子的情况。真实条件下的截面数据随能量变化，在 Quicksilver 中将其简化为一个由 5 个变量控制的 4 次多项式。Quicksilver 程序作为一个代表程序被包含在基准程序组 Coral2 之中，由于输入参数设置应当符合物理规律，因此以基准程序输入为基础，结合基准程序对输入规模的建议采取了表 3 所列的输入。

表 3 Quicksilver 程序的输入规模

Table 3 Input size for Quicksilver program

Parameter	Values
Size of simulation(cm)，X	16
Size of simulation(cm)，Y	16
Size of simulation(cm)，Z	16
Number of particles	163840
CrossSection ratio，Absorption	0.04
CrossSection ratio，Fission	0.05
CrossSection ratio，Scattering	1
$nuBar$	1.6
dt/s	2×10^{-9}

5 实验与结果

依照第 4 节的实验配置，我们进行了一系列实验，表征了

MC 粒子输运程序的特征，并对专用体系结构的性能进行了验证。

5.1 粒子输运程序特征

实验的这一部分以通用多核体系结构为依托，运行 Quicksilver 程序以体现粒子输运随机模拟程序的基本特征。在实验结果中将 cycleInit()，cycleTracking()和 cycleFinalize()作为感兴趣区域(Region of Interest，ROI)。

5.1.1 粒子规模对程序的影响

Quicksilver 在每个时间步长内依次执行 cycleInit()，cycleTracking()和 cycleFinalize()这 3 个函数，3 个函数的执行时间在不同粒子规模输入下的占比如图 6 所示。程序运行在搭载 AMD Ryzen5 PRO 2600 Six-Core Processor 的机器上，运行操作系统为 Ubuntu 20.04 LTS。程序在 3 个阶段分别初始化粒子信息、跟踪粒子执行输运过程以及更新数据并释放缓存。可以看到，随着粒子规模的增大，cycleTracking()占用程序时间逐渐缩短，但在实验范围内依然大于 95%。同时，由于 Quicksilver 程序只对 cycleTracking()函数进行并行化处理，本文着重研究该函数的性能，在 5.4 节中对其余两个部分进行探讨。

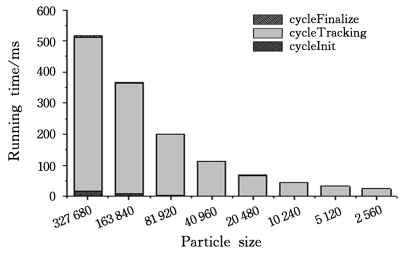


图 6 不同输入规模下粒子输运程序 3 个函数的运行时间

Fig. 6 Running time of three functions in particle transport code with different input scales

5.1.2 访存特性

基于建模的通用多核体系结构，我们在其中运行 Quicksilver 程序，观察它的访存特性。

图 7 给出了指令缓存、数据缓存和二级缓存在 3 个程序阶段的失效率。对于指令缓存而言，3 个函数阶段都表现出了良好的局部性。

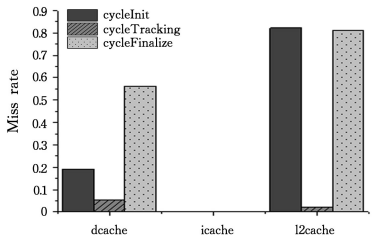


图 7 在通用多核体系结构上运行粒子输运程序的各级缓存失效率

Fig. 7 Cache miss rates of particle transport codes running on a general multi-core architecture

对于数据缓存而言，由于 cycleTracking()函数中随机分支的影响以及 cycleInit()函数中粒子库的初始化和读取，两个阶段的失效率分别达到了 19%和 56%，但二级缓存出现的状况更加显著。二级缓存数据再次突出了 cycleInit()和 cycleFinalize()阶段真正的访存特点，其尺寸显然不能满足粒子输运模拟所需的大量粒子信息和截面数据存储。然而，这些访存特点也给后续优化这两个程序阶段提供了机会。关注核心的跟踪函数，cycleTracking()函数拥有远高于其他两个阶段的指令执行规模，但访存需求远不如其余两个阶段，这一特点为我们精简片上缓存提供了机会，对于片上面积占比极大的二级缓存，我们的小尺寸配置也与访存特点相适应。

5.2 单核性能损失

基于之前介绍过的专用体系结构的各种结构和配置，我们验证其相比通用多核体系结构的性能下降情况。我们使用 gem5 模拟器，按照之前约定的配置建模了两种体系结构，前者作为高性能大核的代表，后者体现了针对此类问题进行简化设计的体系结构原型。我们对运行的 Quicksilver 程序截取循环中的每一次粒子跟踪，并将其作为 ROI，分别截取了 180 次跟踪作为性能的对比数据。图 8 给出了两种体系结构运行 MC 粒子输运的性能对比。

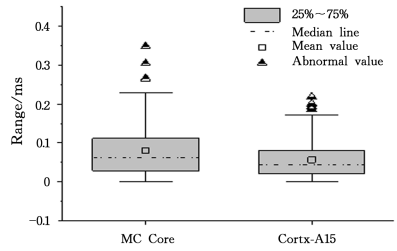


图 8 两种体系结构每次循环的运行时间对比

Fig. 8 Running time comparison of two architectures for loop tracking particles

可以看到，专用体系结构执行对粒子跟踪的运算时耗费了更多的时间。经过统计，整体而言，由于乱序发射被调整成顺序发射，同时减少了执行部件以及简化了各指令阶段发射宽度和缓冲区大小，极简主义体系结构相比高性能大核将获得 41%的性能下降。令人欣慰的事，这一数值完全是可接受的。Endo 等[26]指出，使用通用负载进行测试，同一时期乱序内核比顺序内核获得了约 3 倍的性能提升。而在我们的实验中，正是由于 MC 程序独特的复杂分支，使得乱序内核在分支预测过程中常常受挫，不能很好地发挥其优势。在此情况下，我们开始重点关注极简主义体系结构带来的功耗和面积优势。

5.3 专用体系结构的功耗和面积收益

依据 gem5 建模的架构配置和运行模拟的统计输出，我们在 McPAT 中建模了两种处理器。两者的功耗和面积对比如图 9 所示。

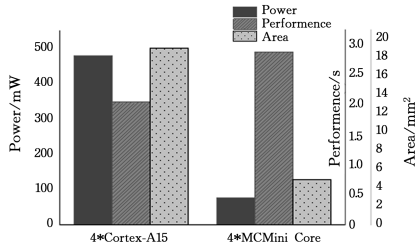


图 9 两种体系结构性能、功耗和面积对比

Fig. 9 Performance, power and area comparison of two architectures

通过之前描述的一系列简化，简化核面积只有乱序核的 1/4，而功耗更是降到了 1/6。在此情况下，综合之前性能下降的情况，专用体系结构也将获得 4.45 倍性能功耗比优势以

及 2.78 倍性能面积比优势。在扩大并行度之后，相应的优势将转化到性能上。

5.4 可重配置结构

本文主要讨论函数主要部分 cycleTracking()函数的优化，但 5.1.2 节中 cycleInit()和 cycleFinalize()函数运行时二级缓存的高失效率令人关注，为此我们设计了二级缓存旁路，可以适时地越过二级缓存直接使一级缓存连接内存。

实验时我们测试了输入规模为 163840 和 1310720 这两组规模，规模的扩大使得两个函数运行时 LLC 的失效率更高。图 10 给出了测试的结果，纵坐标体现了两级缓存结构相比一级结构的优势，若值为负，则说明数据旁路绕过二级缓存后有更好的性能。在 163840 规模时，二级缓存配置仍然具有优势，但当规模扩大到 1310720 时，更简化的缓存配置具有优势，cycleInit()和 cycleFinalize()函数在性能上分别因此获益 4.6%和 18%。因此，数据旁路的开启与输入规模相关，较大规模的模拟将倾向于使用更简化的存储层次结构，更细节的切换策略和方式将在今后的工作中进一步研究。

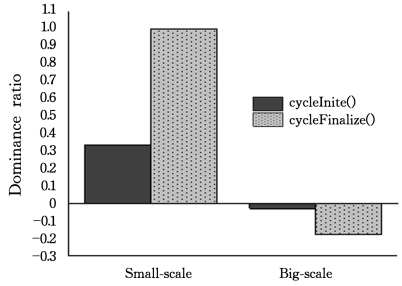


图 10 使用不同规模输入对比二级缓存数据旁路的影响

Fig. 10 Comparison of impact of L2 cache data bypass with different size inputs

结束语 本文面向粒子输运随机模拟应用设计了一种专用体系结构，通过简化内核以提高并行度以及缓存动态优化策略，提高了核心程序运算的性能功耗比和性能面积比。我们使用 gem5 和 McPAT 模拟器作为分析工具，通过合适的建模与仿真表明，本文中的体系结构设计可以将 MC 粒子输运代理程序 Quicksilver 核心跟踪阶段的性能功耗比提升 345%，性能面积比提高 178%。本文从实践上探索，以简化核为基础加速具有高并行性和访存局部性的应用程序，为计算机在底部空间逐渐枯竭的今天探索了性能的发展方向。未来我们将对粒子输运程序的专用体系结构展开进一步研究，包括指令集设计、加速部件定制、性能建模等。

参 考 文 献

[1] YANI S, BUDIANSAH I, RHANI M F, et al. Monte Carlo Model and Output Factors of Elekta InfinityTM 6 and 10 MV Photon Beam [J]. Reports of Practical Oncology and Radiotherapy, 2020, 25(4): 470-478.

[2] FRIDMAN E, HUO X K. Dynamic simulation of the CEFR control rod drop experiments with the Monte Carlo code Serpent [J]. Annals of Nuclear Energy, 2020, 148: 107707-107719.

[3] MAGDZIARZ P, ZDZIARSKI A A. Angle-dependent Compton reflection of X-rays and gamma-rays [J]. Monthly Notices of the Royal Astronomical Society, 1995, 273(3): 837-848.

[4] TYAGI N, BOSE A, CHETTY I J. Implementation of the DPM Monte Carlo code on a parallel architecture for treatment planning applications [J]. Medical physics, 2004, 31(9): 2721-2725.

[5] WAGNER J C, HAGHIGHAT A J. Parallel MCNP Monte Carlo transport calculations with MPI [J]. Transactions of the American Nuclear Society, 1996, 75(9): 338-339.

[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] ANDERSON J A, JANKOWSKI E, GRUBB T L, et al. Massively parallel Monte Carlo for many-particle simulations on GPUs [J]. Journal of Computational Physics, 2013, 254: 27-38.

[8] LUU J, REDMOND K, LO W, et al. FPGA-based Monte Carlo computation of light absorption for photodynamic cancer therapy [C]// 2009 17th IEEE Symposium on Field Programmable Custom Computing Machines. IEEE, 2009: 157-164.

[9] WHITTON K, HU X S, CEDRIC X Y, et al. An FPGA solution for radiation dose calculation [C] // 2006 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines. IEEE, 2006: 227-236.

[10] GOKHALE M, FRIGO J, AHRENS C, et al. Monte carlo radiative heat transfer simulation on a reconfigurable computer[C]// International Conference on Field Programmable Logic and Applications. Springer, 2004: 95-104.

[11] PEPER F. The End of Moore's Law: Opportunities for Natural Computing? [J]. New Generation Computing, 2017, 35(3): 253-269.

[12] WANG Y, BRUN E, MALVAGI F, et al. Competing Energy Lookup Algorithms in Monte Carlo Neutron Transport Calculations and Their Optimization on CPU and Intel MIC Architectures [J]. Procedia Computer Science, 2016, 80: 484-495.

[13] LUJAN P, HALYO V, HUNT A, et al. GPU Enhancement of the Trigger to Extend Physics Reach at the Large Hadron Collider [J]. Journal of Instrumentation, 2013, 8(10): 14214-14247.

[14] AOYAMA T, ISHIKAWA K I, KIMURA Y, et al. First Application of Lattice QCD to Pezy-SC Processor [J]. Procedia Computer Science, 2016, 80: 1418-1427.

[15] 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.

[16] BRUGGER C, DE SCHRYVER C, WEHN N. Hyper: A runtime reconfigurable architecture for monte carlo option pricing in the heston model[C]// 2014 24th International Conference on Field Programmable Logic and Applications(FPL). IEEE, 2014: 1-8.

[17] 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(9): 3715-3737.

[18] LI B, LIU J. Heterogeneous cooperative coputing of particle

transport based on Monte Carlo method on the tianhe 2A system [J]. Computer Engineering and Science, 2020, 42(11): 1922-1928.

[19] RICHARDS D F, BLEILE R C, BRANTLEY P S, et al. Quicksilver: a proxy app for the Monte Carlo transport code mercury [C]// 2017 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2017: 866-873.

[20] GOORLEY J T, JAMES M R, BOOTH T E, et al. Initial MCNP6 release overview-MCNP6 version 1.0 [R/OL]. Los Alamos National Lab. (LANL), Los Alamos, NM (United States). https://doi.org/10.2172/1086758.

[21] ROMANO P K, HORELIK N E, HERMAN B R, et al. OpenMC: A state-of-the-art Monte Carlo code for research and development[C]// SNA + MC 2013-Joint International Conference on Supercomputing in Nuclear Applications + Monte Carlo. EDP Sciences, 2014: 92-97.

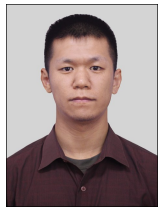
[22] LEISERSON C E, THOMPSON N C, EMER J S, et al. There's plenty of room at the Top: What will drive computer performance after Moore's law? [J/OL]. Science, 2020, 368(6495). https://www.science.org/doi/10.1126/science.aam9744.

[23] BINKERT N, BECKMANN B, BLACK G, et al. The gem5 simulator [J]. ACM SIGARCH Computer Architecture News, 2011, 39(2): 1-7.

[24] 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.

[25] SOHN K, YUN W J, OH R, et al. A 1.2 V 20 nm 307 GB/s HBM DRAM with at-speed wafer-level IO test scheme and adaptive refresh considering temperature distribution [J]. IEEE Journal of Solid-State Circuits, 2016, 52(1): 250-260.

[26] ENDO F A, COUROUSSÉ D, CHARLES H P. 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.



FU Si-qing, born in 1996, postgraduate. His main research interests include computer architecture and high performance computing.

LI Tie-jun, born in 1977, Ph.D, researcher, Ph.D supervisor, is a member of China Computer Federation. His main research interests include high performance computing and so on.

(责任编辑:柯颖)