高技术通讯 2020 年 第30卷 第11期:1093 ~1103

doi:10.3772/j.issn.1002-0470.2020.11.001

NDP-Ledger:面向区块链应用的通用高通量加速架构①

安述倩②*** 李文明③* 范志华*** 吴海彬* 吴 萌* 王 达* 张 浩* 唐志敏***

(*计算机体系结构国家重点实验室(中国科学院计算技术研究所) 北京 100190)

(**中国科学院大学 北京 100049)

摘 要 区块链技术由于去中心化及不可篡改等特性,广泛应用于数字货币、支付交易等领域,其算法对计算能力和存储访问能力有较高要求,导致传统冯诺依曼结构在面向区块链应用时能效比较低。3D 堆叠存储器因可以缓解冯诺依曼结构的访存瓶颈,成为了热门研究方向之一。本文基于3D 堆叠存储器技术及数据流执行模式,提出了一种面向区块链应用的高通量近数据处理(NDP)架构,NDP-Ledger。本文深入分析和论证了区块链应用的计算特征及3D 堆叠存储技术在区块链应用中的适应性问题,并基于数据流执行模式设计了一种通用的高并发区块链加速体系结构,使区块链加速器在满足通用性的前提下提高处理性能。模拟评估结果表明,本文提出的区块链通用加速器结构,在典型区块链应用处理方面的性能优于主流的 CPU 和 GPU。

关键词 区块链;3D 堆叠存储;近数据处理(NDP);数据流;通用加速器结构

0 引 言

作为去中心化交易协议,区块链已广泛应用于众多领域,如数字货币、金融服务和物联网(Internet of things,IoT)[1-4]等,成为研究热点之一。最初基于区块链技术的数字货币应用,如比特币采用的典型的安全哈希算法(secure Hash algorithm,SHA-256)对计算资源需求较大,对存储容量和带宽的需求较小,这使 SHA-256 算法在图形处理器(graphics processing unit, GPU)上取得了优异的效率。但是由于 GPU 的通用性设计,SHA-256 并不能充分利用 GPU 上的各类计算资源,导致计算能效比并不理想[5]。另一方面,由于 SHA-256 算法固定,计算模式简单,定制化的 ASIC 芯片较 GPU 可以实现更高的性能和能效比[6],成为了当前数字货币领域的主要算力之一。但是 ASIC 芯片却违背了比特币设计的去中心化思想[7],为缓解这种冲突,研究人员提出了众多新的数字货币应用,如莱特币(litecoin, LTC)[8]、以太坊(Ethereum)[9]、Dash 币[10]、Zcash[11]、Equihash 算法[12]、Bytom [13]等,算法变得更加复杂,并且需要更大的存储容量和更高的访存带宽。这些数字货币的发展方向是寻找反中心化算法以实现区块链处理器的公平性,导致越来越多的区块链算法增加了对存储、内存带宽的需求,并降低了对计算资源的需求,这使中央处理器(central processing unit, CPU)、GPU 甚至 ASIC 的效率越来越低。同时,区块链算法迅速演变,而 ASIC 只能针对一类区块链算法进行加速,无法实现通用性的加速。基于以上分析,需要找到相对通用的处理架构,满足高能效的区块链算法处理。

近年来,近数据处理(near-data-processing,NDP)

① 国家自然科学基金(61732018, 61872335, 61802367, 61672499),国家重点研发计划(2018YFB1003501),中国科学院国际伙伴计划(171111KYSB20170032)和计算机体系结构国家重点实验室创新(CARCH3303, CARCH3407, CARCH3502, CARCH3505)资助项目。

② 女,1985 年生,博士生;研究方向:计算机系统结构;E-mail:anshuqian@ict.ac.cn

③ 通信作者,E-mail: liwenming@ict.ac.cn

(收稿日期:2019-11-06)

为需要大存储容量和高存储访问带宽的应用提供了一种有效的解决方案。3D 堆叠存储技术提供了一种通过降低数据搬运成本来提高计算系统的性能和降低功耗的方法。此外,为了进一步提高大规模并行执行应用程序的效率,数据流执行模式已在许多处理器中得到广泛应用。数据流执行模式具有天然的并行特征,并通过降低访存来缓解“存储墙”问题。

通过分析不同类型的区块链应用程序的特点以及 3D 堆叠存储技术特征,本文提出了一种基于 3D 堆叠存储器和数据流执行模式的面向区块链应用处理的通用数据处理架构,NDP-Ledger。本文的主要贡献包括:

(1)分析并对比了不同区块链应用程序的算法特征,并基于算法特征,提出了通用 NDP 加速器架构设计 NDP-Ledger。

(2)提出了基于 3D 堆叠存储器的面向区块链应用的通用 NDP 加速架构。

(3)设计并实现了粗粒度的数据流执行模型,提高数据处理的并行度,高效地控制内存以及阵列之间的数据流动。

(4)实验结果表明,本文提出的通用加速处理架构与通用多核 CPU 相比,性能提升了 39.98 倍;与通用 GPU 相比,性能提升了 1.8 倍。

1 相关工作

近几年,区块链技术在学术界和工业界都获得了极大的关注,基于区块链协议的应用越来越多地服务于国民经济的发展,包括金融服务、公证、智能合约、IoT 以及数字货币。此类应用对处理器的性能和功耗提出了更高的要求,低效的 CPU 无法满足此类应用的需求。因为具有更多的计算资源和更高的并行度,GPU 一度成为继 CPU 之后的首选区块链处理架构。为了追求更高的收益,针对一种或几种区块链应用的专用 ASIC 加速器逐渐替代了 GPU,例如针对比特币设计的 AvalonMiner [14]等。然而,由于新算法和新应用的不断涌现,这些 ASIC 无法满足不断演化的算法需求,仅能支持一种或几种算法的高效执行。另外,越来越多的算法不适合在 ASIC 上进行计算,这给 ASIC 设计带来了更大的挑战。

随着 3D 封装技术的发展,NDP 为具有简单计算模式和大存储容量需求的应用程序提供了新的高效解决方案。Kang 等人[15]将计算单元、有限状态机和其他控制逻辑集成到普通动态随机存储器(dynamic random access memory, DRAM)中构成了 3D 堆叠内存,用来加速用于 DNA 比对的 Blast 算法。Pugsley 等人[16]使用了低能耗的指令内核,例如 Cortex A5,与混合内存立方体(hybrid memory cube, HMC)一起实现了高能效的新一代数据中心(new data center, NDC)架构。Nair 等人[17]在 HMC 的逻辑层上实现了复杂的运算部件,用于加速科学计算应用处理。Ahn 等人[18]提出了一种称为 PEI(PIM-enabled instructions)的通用 NDP 体系结构,该体系结构实现了一种可基于数据的位置实现内存处理的机制。与之前的工作类似,Santos 等人[19]在 HMC 的逻辑层中放置了可重构处理单元,可以灵活处理不同类型的数据。文献[20]利用现场可编程门阵列(field programmable gate array, FPGA)和粗粒度可重构阵列(coarse-grained reconfigurable array, CGRA)的优势来实现高能效和高灵活性的 NDP 系统。同时,由于图计算的大量离散存储访问的特征,研究人员提出了基于 HMC 的高效架构用于图计算处理[21-23]。在其他相关工作中,HMC 用于字符串处理[24]和矩阵中的乘加运算[25],这些算法具有简单的计算模式和海量的数据集。综上所述,随着 3D 堆叠技术的发展,NDP 已成为解决内存密集型应用程序的最有前景的方法之一。

另一方面,随着应用程序(如人工智能(artificial intelligence, AI)、IoT 和网络服务)所需要处理的数据量的不断增长,数据流体系结构展现出比传统冯诺依曼体系结构具有更好的执行并行度和执行性能。数据流体系结构的优点主要来自 2 个方面:首先是数据流执行模式的天然并行性,一旦数据准备好就触发指令,无需考虑指令和数据的依赖性;第二是内存访问量的减少,相当一部分数据在片上流动,缓解了传统冯诺依曼结构中的“存储墙”的问题。

已有的诸多研究工作已经证明了数据流体系结构在许多应用程序中的优势。例如,Wave Computing 提出的数据流处理器(DPU)[26]实现了高性能处理 AI 应用程序,NeuFlow 处理器[27]是为视觉处理设计的数据流架构。其他典型的数据流体系结构包括 TeraFlux[28]、Runnemede[29]、TRIPS[30]和 WaveScalar[31],在特定应用中,均体现出数据流结构对于冯诺依曼体系结构的巨大优势。

基于以上分析,本文探索了基于 3D 堆叠内存和数据流控制机制的区块链应用通用加速结构设计思路,以实现更为高效的区块链通用处理架构。

2 区块链应用分析

去中心化及不可篡改是区块链的核心思想,分布式账本是由不断产生的区块组成,这些区块在整个区块链网络上共享给不同的组织,以确保分类账的公平性、开放性和安全性。区块链网络中的每个节点都接收区块头,并在处理后返回计算结果。该过程由多种算法组成,用以证明工作已经完成。随着越来越多不同需求的逐渐演变,基于区块链应用的算法变得越来越复杂,算法的核心对于计算资源要求不高,但是越来越依赖于存储器来减少 ASIC 对于去中心化思想的影响。在本节中,首先根据算法采用的核心思想总结不同算法的特征和具体的资源需求,然后根据不同的算法特征以及各种计算、存储需求,分析适用于不同区块链应用程序的硬件架构设计,并结合新兴的存储技术,分析未来的结构设计的可能性。

2.1 哈希相关的算法

比特币作为区块链技术的第 1 个著名应用,利用了加密算法 SHA-256,该算法执行定点模式的哈希运算,包括 6 个逻辑运算功能,如表 1 所示。在大多数的哈希算法中,移位、与、或、异或、非运算是最常见而且是主要的运算。这些操作很简单,对计算资源的复杂性要求不高,但是需要重复执行多次,这对于结构复杂而功能强大的 CPU 及 GPU 来说是一种资源浪费。

表 1 SHA-256 算法的逻辑运算符和功能描述

操作符	操作	功能
$\oplus$	按位异或	$Ch(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z)$
$\wedge$	按位与	$Maj(x, y, z) = (x \wedge y) \oplus (\neg x \wedge z) \oplus (\neg y \wedge z)$
$\neg$	按位补码	$\sum_0(x) = S^2(x) \oplus S^{13}(x) \oplus S^{22}(x)$
R^n	右移 n 位(SHR)	$\sum_1(x) = S^6(x) \oplus S^{11}(x) \oplus S^{25}(x)$
S^n	向右旋转 n 位	$\sigma_0(x) = S^7(x) \oplus S^{18}(x) \oplus R^3(x)$
+	字符串连接(附加)	$\sigma_1(x) = S^{17}(x) \oplus S^{19}(x) \oplus R^{10}(x)$

在比特币之后,研究人员提出了使用更复杂的哈希运算或不同哈希运算组合的数字货币。因此,更加复杂和需要更多软硬件资源的工作量证明(proof of work, PoW)算法被提出,如 Quark 应用了 9 种哈希函数,包括 BLAKE, BMW, GROESTL, JH, KECCAK 和 SKEIN 等等。DASH 结合了 11 种哈希加密算法,命名为 X11。相应地,X13 和 X15 算法也随之被提出。几乎所有方法都基于现有的加密算法进行了升级,这些算法强调简单的逻辑运算而不是复杂的科学计算,这使得针对此类算法的 ASIC 的研发相对容易。

2.2 内存依赖型算法

因为纯哈希算法易于 ASIC 加速,这违背了区块链去中心化的原则,因此,一些更平等的共识算法被逐渐提出。例如,Ethash 是对 Dagger-Hashimoto 进行优化,该算法的执行很大程度依赖存储容量和访存带宽,其数据量的大小超过 1 GB,并且每年增加 7 GB,与内存需求相比,对 CPU 的计算能力的需求并不是那么强烈。另一个典型的应用是 Zcash,它使用被称为生日悖论的 Equihash 作为其共识算法。Equihash 是一种依赖于内存的算法,每个线程至少需要 1 GB 的内存容量。还有许多其他共识算法,所

有这些算法都强调带宽和内存容量,而不是计算性能。因此,针对这类应用,NDP 架构具有很大的优势。

2.3　其他典型算法

有些研究人员认为 ASIC 仅支持哈希类运算,造成了资源浪费,因此并不认可这种做法。使用现有的计算资源来处理区块链应用程序,同时进行一些其他有效计算,这方面是有意义的。例如,tensority 是基于张量计算模型的 ASIC 友好共识算法,其中矩阵和张量运算被引入到哈希过程中。因此,这些处理机还可以用于 AI 硬件加速服务、并行计算,并产生额外的社会效益。另一个例子是 Primecoin,它是世界上第 1 个以科学计算为设计目标的共识算法。在 Primecoin 中,PoW 不仅为网络提供了安全性和生成区块的功能,而且还生成了数学研究人员感兴趣的特殊形式的质数链。

以上为代表性的区块链应用程序。当然,还有许多其他的区块链应用程序使用了复杂且有意义的共识算法,这些算法面向不同的应用领域,都可以归为上述类别。

2.4　设计面向区块链应用的高效结构

大多数区块链应用程序的特点是逻辑操作简单,内存容量需求大和带宽需求高。因此,要为区块链应用开发更加通用和高效的处理架构,应该充分考虑并利用这些特征。随着 3D 封装技术的发展,3D 堆叠存储器技术已经成熟。已有研究证明了 3D 堆叠存储器[25-30]的优势,它通过减少数据搬运距离并提供高带宽来提高计算机的性能。3D 堆叠存储器,如 HMC 已经成为比传统存储器体系结构更具吸引力的设计方案。本文以 HMC 为例,研究 3D 堆叠存储器在区块链应用场景中的效果。

如图 1 所示,在 HMC 中,逻辑层(logic die)和多层 DRAM 堆叠在芯片上,使用硅通孔(through silicon via,TSV)技术以进行层间通信。典型的 HMC 由 32 个库(vault)组成,每个库在垂直方向上由几个可以独立访问的堆叠随机存取存储(random access memory,RAM)块组成,以此实现更短的存储访问距离和更高的内存带宽。与传统的内存系统相比,HMC 具有更高的访存性能和更低的能耗损失。逻辑层可以通过集成运算部件来就近处理存储器中的数据,添加了运算部件的 HMC 可以像 ASIC 一样作为高能效的加速架构。

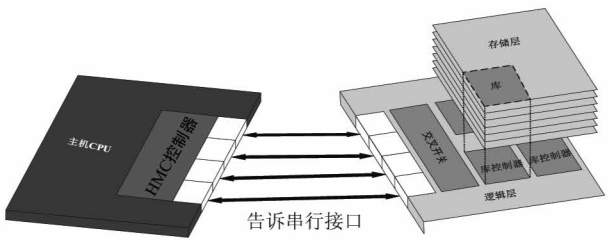


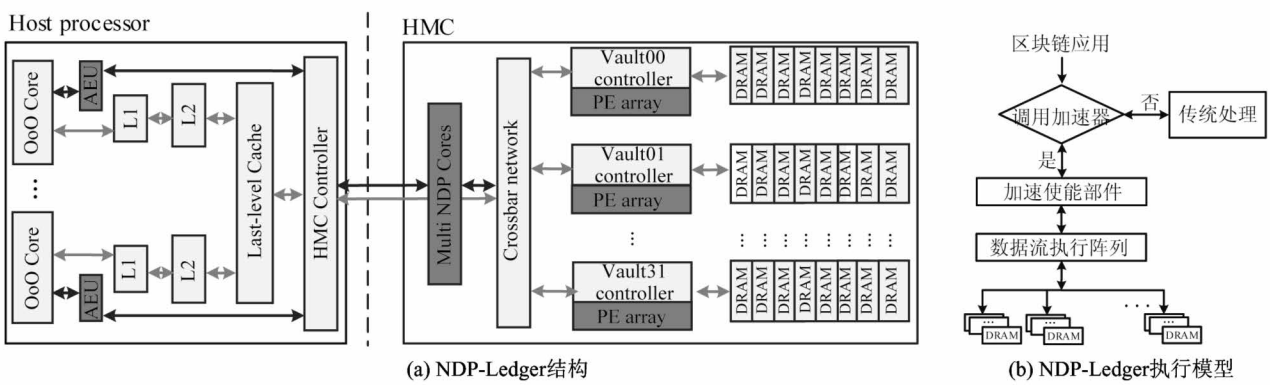
图 1　HMC 存储结构示意图

3　NDP-Ledger 架构设计

本文提出了一种基于 3D 堆叠存储器和数据流执行模式的近数据处理架构 NDP- Ledger,以加速各种区块链应用程序的执行。本节将讨论 NDP- Ledger 的结构设计、编程模型和数据流控制机制。

3.1　NDP-Ledger 体系结构概述

NDP-Ledger 的总体架构如图 2 所示。在主机端,设置了 1 个加速使能单元(acceleration enable unit,AEU)。AEU用于监视主机处理器的行为并决定



(a) NDP-Ledger结构　　(b) NDP-Ledger执行模型

图 2　NDP-Ledger 体系结构图

是否激活 NDP-Ledger。用户使用专门的编程接口来控制 AEU。在收到启动 NDP-Ledger 加速区块链算法的 CALL 之后,CALL 将绕过缓存直接发送到 HMC 控制器的 NDP 处理内核,NDP 处理器核解析指令并运行,指令执行结束后,收集结果并处理,并将最终结果发送回主机端。在 AEU 未使能状态下,传统的内存访问系统仍可按其普通方式工作。当选择 NDP-Ledger 模式时,数据将直接在 HMC 中被处理。在数据一致性方面,为了保持高速缓存和 HMC 之间的数据一致性,2 种方式可以选择。第 1 个方法是当命令从 AEU 发送到 HMC 控制器时,控制器需要广播到所有缓存,无效掉 HMC 端需要由内核处理的所有数据。第 2 个方法是高速缓存绕过策略,即需要由处理器核处理的数据应标记为非高速缓存存储数据。在 NDP-Ledger 中,选择第 2 种方法,即绕过缓存策略,以避免维护缓存一致性带来开销。图 2(b)显示了 NDP-Ledger 的执行模型。

图 3 显示了 NDP-Ledger 的详细加速架构设计。如图 3(a)所示,在 HMC 的逻辑层中的每个 vault 控制器上实现加速处理单元阵列,由 16 个处理单元(process element,PE)组成。所有运算单元都由高速总线连接。如图 3(b)所示,PE 之间由 1 个 mesh 结构的片上网络连接。4 个 PE 由同一个路由器连接,并且可以在数据流执行模型中作为 1 个工作组。共享路由器可提高 PE 组内部的通信效率。图 3(c)展示了 PE 的结构,它由指令缓冲区、上下文操作数、流水线和路由器组成。本文采用粗粒度的数据流执行模型,指令块(而不是指令)作为最小的调度单元。粗粒度的数据流执行模型可以简化控制逻辑,并且流水线设计仍使用控制流执行模型。图 3(d)展示了流水线的结构。区块链应用程序中大部分的运算为简单的逻辑运算,因此,PE 中配置了 2 个逻辑单元(logic unit,LU)来增强逻辑运算能力,此外还包含 1 个算术单元,1 个浮点单元和 1 个 LD/ST 单元。与传统处理器流水线不同,在数据流执行模式中,增加了 FLOW 单元来支持 PE 之间的数据流动。FLOW 单元由特殊指令控制,用来将数据从操作数缓冲区复制到其他 PE 中。该指令在源代码的编译阶段生成。数据流映射决定了指令块和 PE 之间的映射关系。为了提高执行的并行性,PE 采用 SIMD 执行模式。

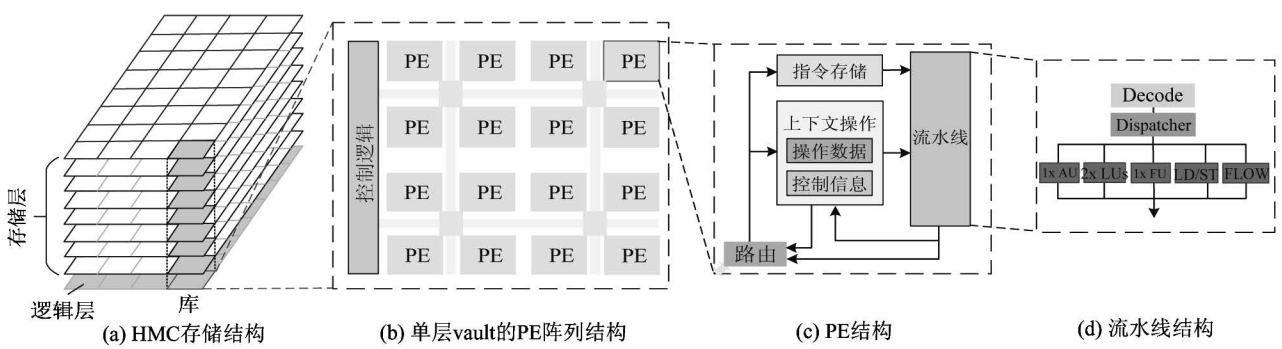


图 3　NDP-Ledger 加速架构示意图

3.2　编程和指令映射

在编程模型的设计上,实现了类似于 CUDA 的编程模型作为用户调用运行内核算法的 NDP-Ledger 加速器的接口。加速单元仅用于处理算法的内核部分,而其他功能,如通过 Internet 与其他节点进行通信等功能,在主处理器上执行。

本文实现了粗粒度的数据流执行模型,其中指令块是最小的调度单元。指令块通常包含多条指令,用于实现特定功能。比如 *for*( )、*while*( ) 或者公式等(例如比特币应用中的 *Maj*( )、*Ch*( )、Σ0( )、σ1( )),其中一个或者多个的功能实现可以编写在 1 个指令块中,称之为执行块(execution block,Exeblock)。

下面以比特币为例来详细说明 Exeblock 和映射机制的实现。如图 4 所示,根据上面介绍的规则,将该算法编程为几个Exeblock(白色上层框),灰色

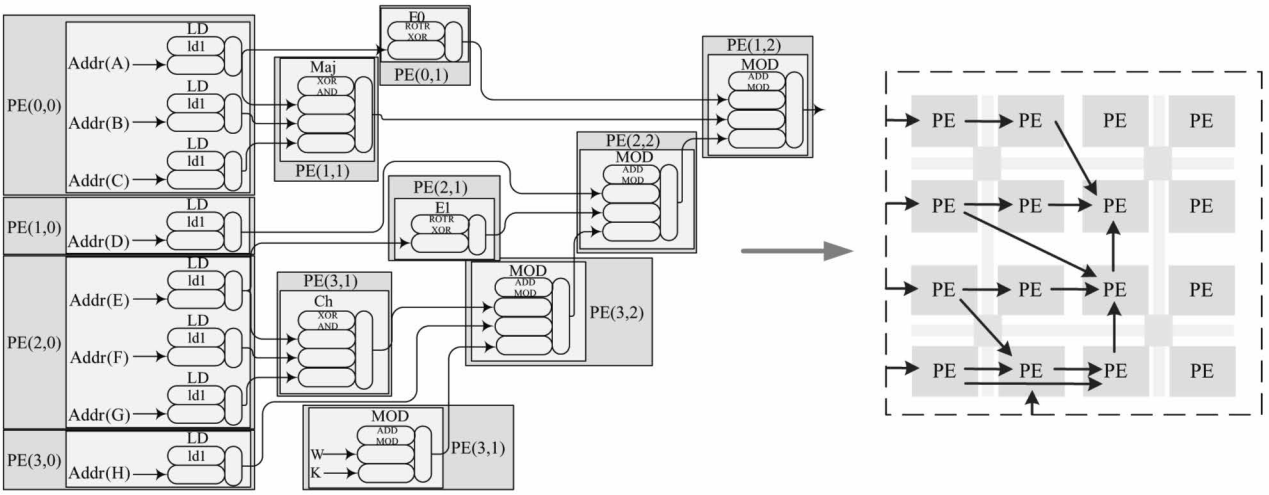


图 4　比特币算法到 PE 阵列的映射方式以及数据流图

底框显示运行 Exeblock 的 PE。右侧部分显示数据的流向。在当前映射方式中，选择边缘 PE 加载数据并将数据搬运到它们的右侧的 PE 中以进行下一步计算。空闲 PE 可以处理另一个比特币任务，为了充分利用计算资源，多个任务可以同时映射到 PE 阵列上运行。

在进行 Exeblock 划分时，按照图 5 所示的方法，针对算法进行解析。根据 SHA-256 的算法特点，首先将算法的执行过程划分为“报文预处理”、“加载 Hash 初值”、“Hash 运算”等 3 大部分，针对每一部分，根据其所需要执行的具体功能，按照指令数均衡的原则，划分指令块，并利用 PE 的任务级并行以及 SIMD 特性，挖掘算法的并行性；根据指令块之间的数据依赖关系，实现算法的依赖关系图，最终形成数据流图。图 4 是针对 SHA-256 算法的宏观解析，具体到实际的指令块，编译器通过检索实际的数据流依赖关系，会形成更加复杂的数据流图。

除了 Exeblock 的划分，数据流执行模式中最重要的映射机制，是将 Exeblock 放到指定 PE 上的策略，以往研究人员针对映射机制研究已经有很多工作，如面向科学计算的映射算法[32]、考虑负载均衡的映射算法 TBIM[33]（topology-based instruction mapping）等。TBIM 算法如图 6 所示，将 Exeblock 根据数据依赖关系形成数据流图（dataflow graph），作为映射算法输入之一，并且将 PE 阵列以及拓扑关系作为映射算法第2个输入，通过衡量各个PE上定点

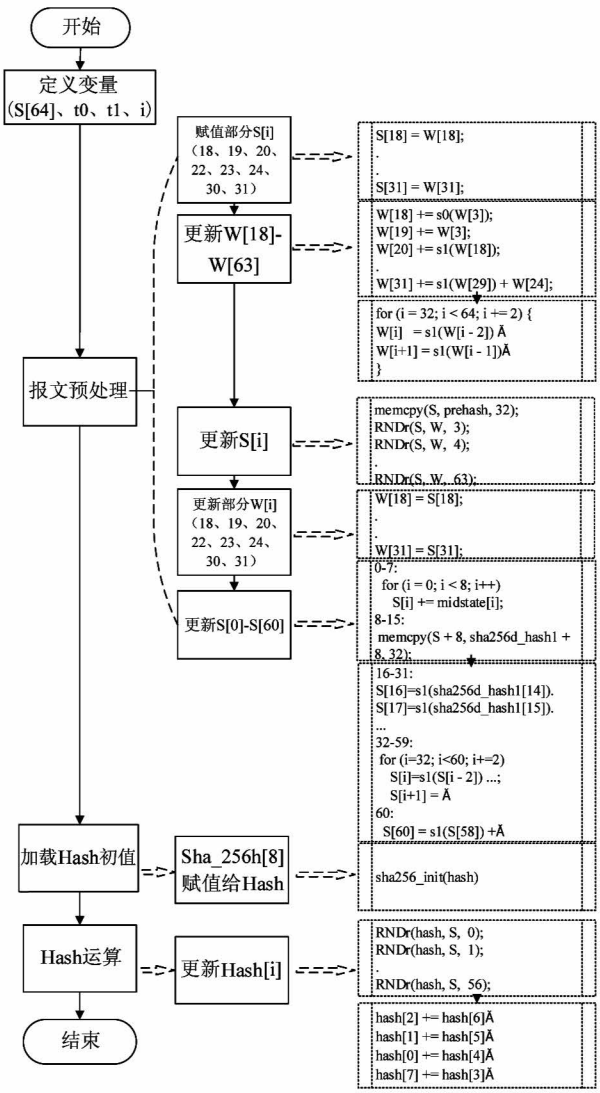


图 5　SHA-256 算法数据流图解析过程

```
Input: G - dataflow graph, A - array of PEs
Output: M - a mapping G -> A
1: (L, NL) = divideByLoop(G)
2: if L not null
3:     {FL, IL} = divideByInstType(L)
4: end if
5: if NL not null
6:     {FNL, INL} = divideByInstType(NL)
7: end if
8: for each set s in {FL, IL, FNL, INL}
9:     if s is not null
10:        SC = groupByTopo(s, A, M)
11:        PEC = clusterByPE(SC, A, M)
12:        for each cluster c in PEC
13:            M + = clusterMap(c, A, M)
14:        end for
15:    end if
16: end for
```

图 6　TBIM 算法[33]

浮点负载以及网络传输代价的大小,逐个检索各个指令块的最佳映射位置,获得最优的映射结果。

3.3　粗粒度的数据流控制机制

在本文实现的粗粒度数据流模型中,最小调度单位是 Exeblock,其中包含一行或几行程序代码,这些代码可以实现用户确定的一个或多个功能。一旦所有数据到达,就可以触发 Exeblock 执行。如图 7 所示,一个应用程序可以分为几个任务,每个任务由几个 Exeblock 组成。可以根据数据量的大小和同时运行的任务数将 Exeblock 执行流图映射到内部 PE 级别、PE 阵列级别或跨 PE 阵列级别。在图中,有 2 个要处理的应用程序,每个应用程序包含几个任务。图中显示了只有 1 个 PE 阵列的可能映射结果。数据流图映射可能跨 PE 阵列,多个 PE 阵列共同在一个应用程序或任务上工作。数据流映射是在编程时确定的,并将配置信息发送到每个 PE 阵列的控制逻辑,如图 4 所示。为了使数据流图的 Exeblock 更有效地工作,本文开发了一种双向 ack 机制来控制 Exeblock 之间的交互。每个 Exeblock 都有 1 个三位的状态字,该状态字指示与执行块相关的运行状态。三位的状态字用来控制上游和下游 Exeblock 的执行,如图 8 所示。Ack、active 和 done 是 3 个控制信号,用于维持数据流图的执行。Ack 用于将自身节点的完成消息通知上游节点。Active 负责启动下游节点。Done 信号标记节点所有循环的结束。所有 Exeblock 均受 HMC 内存中 NDP 内核的控制。这样的设计方案使得 PE 支持任务级并行,同时可以并行执行一个任务的不同迭代。图 8 展示了 5 个 Exeblock 的执行示例。图中标记了控制步骤的详细说明。每个 Exeblock 都有 upstream、downstream、enable 三位状态字,分别用于表示当前节点需要回应

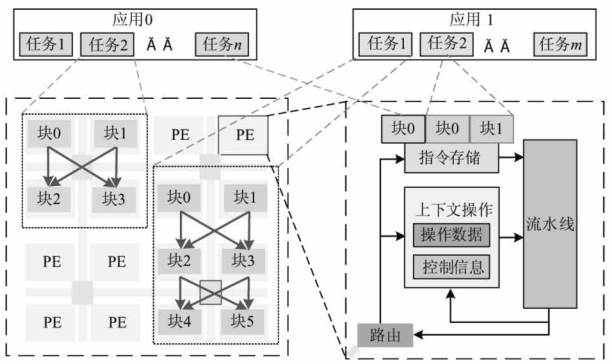


图 7　PE 阵列和 PE 单元上的映射示例图

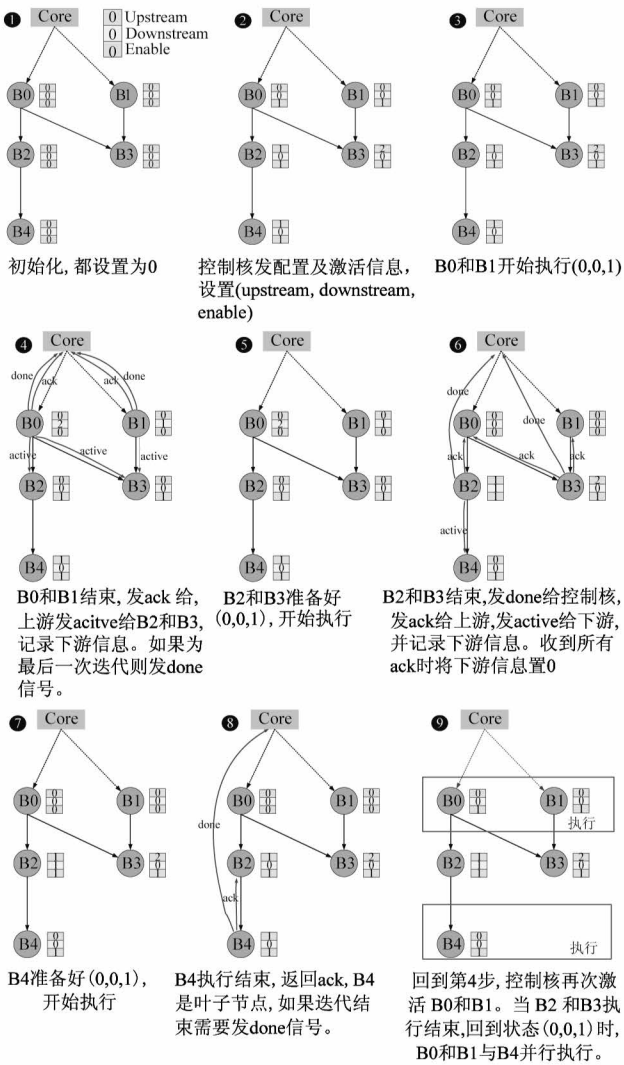


图 8　粗粒度数据流的控制机制示意图

的上游个数、需要激活的下游个数以及是否已经被控制部件激活。Exeblock 可以执行的条件是当前被激活的节点需要发送 ack 和 active 的上下游节点数都为 0,即在上次迭代执行已经结束的情况下,才可以进入下一次执行。图中 B0、B1 为根节点,无父节点,B2、B3 分别为其子节点,B3 和 B4 为叶子节点。Enable 主要用来控制根节点是否可以正常执行,根节点收到控制核的激活信号后,处于可执行状态,执行结束后需要将 enable 归 0,避免在子节点当前迭代未结束之前再次执行;中间各子节点包括叶子节点,主要通过判断 upstream 来确定是否可以正常执行。运行过程中,中间节点的 enable 信号一直处于激活状态,后续的各次迭代收到上游节点的激活信息后进行开始执行。控制核需要收到所有节点发出的 done 信号之后,才会激活任务的下一次迭代,如图 8 第 9 步所示,B0 和 B4 同时在执行任务的不同迭代,实现任务级并行。

4 实验与评估

为了验证 NDP-Ledger 的性能,对本文所提出的架构进行了模拟仿真,并将其与 Intel CPU 和 NVIDIA GPU 进行了比较。

4.1 实验设置

本实验将 CasHMC[34] 集成到 Gem5 [35] 中来模拟 NDP-Ledger 的架构设计。Gem5 是一个完整的系统模拟器。CasHMC 是一个实现了 HMC 存储的周期精确的模拟器。实验中,采用了 1 个 Cube,存储空间大小为 8 GB。每个 Cube 包含 32 个 vaults,每个 vault 分为 16 个 banks,其中 32 个 vaults 分别对应 32 个 PE 阵列结构。本文在 HMC 内存中添加了 4 个 ARM1176 作为多 NDP 处理器核来控制 PE 阵列,ARM1176 用作加速器的 MCU,它负责调度任务到加速器并与主机处理器进行通信。为了提高并行计算能力,在实验中,每个 PE 单元包含 4 组算术部件,并以 SIMD 方式执行。本文使用 CACTI 对寄存器的功耗和面积、配置缓冲区以及其他部件进行建模。使用通用服务器级 Intel E5-2697 v3 和高性能服务器加速卡 NVIDIA TITAN XP 作为对比平台,详细配置如表 2 所示。NDP-Ledger 的功耗和面积参数为 32 nm 工艺下参数。

表 2 CPU、GPU 和 NDP-Ledger 参数对比

项目	CPU	GPU	NDP-Ledger
SIMD 数	224	3 840	2 048
频率	3.6 GHz	1.5 GHz	1.5 GHz (NDP cores) 1.5 GHz (PE arrays)
面积	456.12 mm^2	471 mm^2	<303.36 mm^2
功耗	145 W	250 W	<116 W
存储	L1: 14×64 kB L2: 14×256 kB LLC: 35 MB	GDDR5X 547.7 GB/s	4 links per package, 120 GB/s per link 480 GB/s
	DRAM: 16 GB	12 GB	8 GB

HMC 逻辑芯片的功耗和面积数据来源于 Micron 公司公布的数据[36]。四个 NDP 核心的总功率为 0.5 W,HMC 为 90 W[16],每个 PE 阵列的总功率为 0.8 W,32 个 PE 阵列的总功率为 25.6 W。NDP-Ledger 的整体功率为 116 W。NDP-Ledger 的面积等于 HMC 控制逻辑部分的面积。从表中可以看出,NDP-Ledger 的面积和功率都小于 CPU 和 GPU。

本文选择了具有代表性的区块链应用的算法作为基准测试程序,如表 3 所示。算法中的主要操作步骤已在表中列出。这些运算主要与移位运算、逻辑运算和矩阵运算有关。费马小定理是搜索素数 Cunningham 链的关键操作。

4.2 性能评估与分析

在实验中,通过编程来充分利用 CPU 和 GPU

的硬件资源。为了排除主机系统对 CPU、GPU 和 NDP-Ledger 的影响,仅使用这 3 个平台执行区块链应用程序的核心算法部分。

表 3　基准测试程序说明

算法		核心操作
SHA256	*S*1	$\sigma_0(x)=S^7(x)\oplus S^{18}(x)\oplus R^3(x)$
	*S*2	$\sigma_1(x)=S^{17}(x)\oplus S^{19}(x)\oplus R^{10}(x)$
	*S*3	$\Sigma_0(x)=S^2(x)\oplus S^{13}(x)\oplus S^{22}(x)$
	*S*4	$Maj(x,y,z)=(x\wedge y)\oplus(\neg x\wedge z)\oplus(\neg y\wedge z)$
	*S*5	$\Sigma_1(x)=S^6(x)\oplus S^{11}(x)\oplus S^{25}(x)$
	*S*6	$Ch(x,y,z)=(x\wedge y)\oplus(\neg x\wedge z)$
	*S*7	$CAT(h)=h0+h1+h2+h3+h4+h5+h6+h7$
Scrypt	*S*1	PBKDF2-HMAC-SHA256
	*S*2	$X[i]\hat{}=(((x)<<(y))\mid((x)>>(32-(y))))$
Tensority	*S*1	SHA256
	*S*2	Scrypt
	*S*3	Matrix Construct
	*S*4	Matrix Multiplication: $Ma\times Mb^T$
	*S*5	FNV: Hash matrix
	*S*6	SHA256
Equihash	*S*1	$Xi_1\oplus Xi_2\oplus\cdots\oplus Xi_2^k=0$
Fermat's little theorem	*S*1	$a^{(p-1)}\bmod p=1$

图 9 显示了 CPU、GPU 和 NDP-Ledger 的性能比较。结果表明,NDP-Ledger 的平均运算速度是 E5-2697 v3 的 39.98 倍,是 NVIDIA TITAN XP 平均速度的 1.8 倍。最好的加速结果为 Scrypt 算法,因为该算法具有简单的逻辑运算和访存密集型的特点。

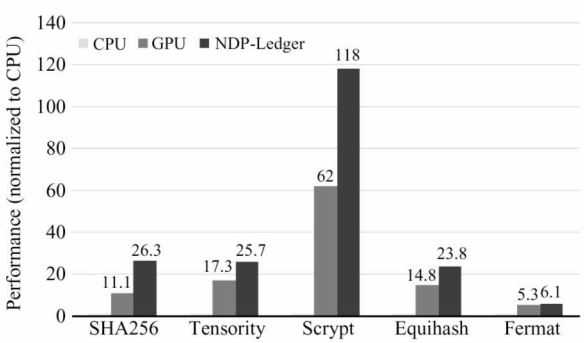


图 9　CPU、GPU 和 NDP-Ledger 性能对比图

图 10 显示了 3 种结构的能效对比。由于 NDP-Ledger 所拥有的热设计功耗(thermal design power, TDP)比 GPU 少,因此在能效上取得了更好的结果。结果表明,NDP-Ledger 和 GPU 相比,能效比是其3.9 倍;和 CPU 相比,能效比是其 49.97 倍。

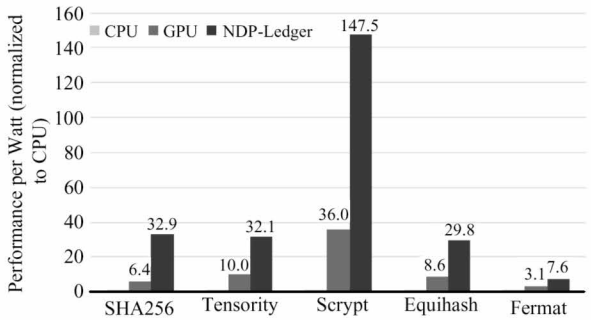


图 10　CPU、GPU 和 NDP-Ledger 功效对比图

在性能密度方面,NDP-Ledger 由于其 3D 堆叠存储技术减小了总面积,从而提高了单位面积的性能,如图 11 所示。集成 PE 阵列的逻辑层,其面积等于 HMC 本身面积。实验结果表明,NDP-Ledger 的单位面积计算密度是 E5-2697 v3 的 60.11 倍,是 NVIDIA TITAN XP 的 2.81 倍。

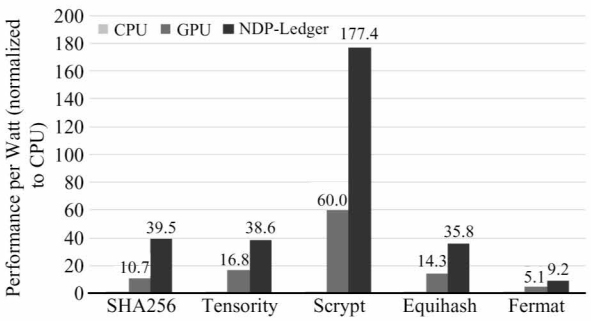


图 11　CPU、GPU 和 NDP-Ledger 性能密度对比图

5　结　论

随着区块链技术的飞速发展,区块链应用越来越丰富,其核心算法对处理器的计算性能和存储访问性能要求不断变化,目前缺乏可以高效处理区块链应用的通用结构。针对这一现状,本文提出了基于 3D 堆叠存储的具有数据流执行模式的通用 NDP 区块链加速结构设计,将 PE 阵列单元与 3D 堆叠存储集成在一起,以提高性能密度并减少数据传输距

离。实验表明,本文提出的 NDP-Ledger 提高了区块链应用的处理性能和能效比。同时,提出了一种面向粗粒度数据流执行模型的高效编程和控制行机制,提高了程序执行的并发度,减少了存储器访问的需求。评估结果表明,NDP-Ledger 处理性能是主流 CPU 的 39.98 倍,是主流 GPU 的 1.8 倍,验证了本文提出的结构的有效性。

本文提出的通用性加速结构 NDP-Ledger 结合了 3D 堆栈存储以及数据流执行模型,下一步的研究工作需要针对这两方面结合来进行。针对数据流执行模型的编译技术研究,进一步减少对存储访问的依赖;针对存储层次结构的优化研究,进一步提升存储的效率。这两方面对 NDP-Ledger 的结构及性能提升有着重要影响,也是基于本文提出的结构未来要开展的工作。

参考文献

[1] Dorri A, Kanhere S S, Jurdak R, et al. Blockchain for IoT security and privacy: the case study of a smart home [C]// IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), Kona, USA, 2017:618-623

[2] Konstantinos C, Michael D. Blockchains and smart contracts for the Internet of Things[J]. *IEEE Access*, 2016, 4: 2292-2303

[3] Sarah U. Blockchain beyond Bitcoin[J]. *Communications of the ACM*, 2016,59(11):15-17

[4] 陈伟利,郑子彬. 区块链数据分析:现状、趋势与挑战[J]. 计算机研究与发展, 2018,55(9): 1853-1870

[5] Shin M, Hiroki M. Accelerating blockchain search of full nodes using GPUs[C]// Euromicro International Conference on Parallel, Cambridge, UK, 2018:244-248

[6] Michael B T. The evolution of bitcoin hardware[J]. *IEEE Computer*, 2017, 50(9): 58-66

[7] Beikverdi A, Song J. Trend of centralization in bitcoin's distributed network[C]// IEEE/ACIS 16th International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), Takamatsu, Japan, 2015:1-6

[8] JoE L. Is Litecoin the future of cryptocurrency. [EB/OL]. https://www. investopedia. com/news/litecoin-future-cryptocurrency/:Litecoin, 2017

[9] Gavin W. Ethereum: a secure decentralized generalized transaction ledger. [EB/OL]. http://gavwood. com/paper. pdf: Ethereum, 2014

[10] Nathaniel L. Digital is the Cash. [EB/OL]. https://media. dash. org/wp-content/uploads/Digital _ is _ the _ Cash-Nathaniel _ Luz. pdf:DASH, 2019

[11] Eli B S, Alessandro C, Christina G, et al. Zerocash: decentralized anonymous payments from bitcoin[C]// IEEE Symposium on Security and Privacy (SP), Berkeley, USA, 2014:459-474

[12] Alex B, Dmitry K. Equihash: asymmetric proof-of-work based on the generalized birthday problem[J]. *Ledger*, 2017, 2: 1-30

[13] Bytom F. Bytom: an interoperation protocol for diversified byte assets [EB/OL]. https://bytom. io/Bytom-Technical-White-Paper-EN. pdf: Bytom Foundation, 2018

[14] Michael B T. The evolution of Bitcoin hardware. [EB/OL]. https://cseweb. ucsd. edu/ ~ mbtaylor/papers/Taylor _ Bitcoin _ IEEE _ Computer _ 2017. pdf : University of Washington, 2017

[15] Kang J Y, Gupta S, Gaudiot J L, et al. An efficient PIM (processor-in-memory) architecture for BLAST[C]// Conference Record of the 38th Asilomar Conference on Signals, Systems and Computers (ASILOMAR), Pacific Grove, USA,2004:503-507

[16] Pugsley S H, Jestes J, Zhang H H, et al. NDC: analyzing the impact of 3D-stacked memory + logic devices on map reduce workloads[C]// 2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Monterey, USA, 2014:190-200

[17] Nair R, Antao S, Bertolli C, et al. Active memory cube: a processing-in-memory architecture for exascale systems [J]. *IBM Journal of Research and Development*, 2015, 59(2-3):1-14

[18] Ahn J, Yoo S, Mutlu O, et al. PIM-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture[C]// Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, USA, 2015:336-348

[19] Santos P C, Oliveira G F, Tome D G, et al. Operand size reconfiguration for big data processing in memory[C]// Design, Automation and Test in Europe Conference and Exhibition (DATE), Lausanne, Switzerland, 2017: 710-715

[20] Gao M Y, Kozyrakis C. HRL: efficient and flexible reconfigurable logic for near-data processing[C]// IEEE International Symposium on High Performance Computer Architecture (HPCA), Barcelona, Spain, 2016:126-137

[21] Nai L F, Hadidi R, Sim J, et al. Graph PIM: enabling instruction-level PIM offloading in graph computing frameworks[C]//2017 IEEE International Symposium on High Performance Computer Architecture (HPCA), Austin, USA,2017:457-468

[22] Ahn J, Hong S, Yoo S, et al. A scalable processing-in-memory accelerator for parallel graph processing[C]// 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA), Portland, USA,

2015:105-117

[23] Khoram S, Zhang J L, Strange M, et al. Accelerating Large-Scale Graph Analytics with FPGA and HMC[C]// 2017 IEEE 25th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), Napa, USA,2017: 82-82

[24] Li W M, Ye X C, Wang D, et al. PIM-WEAVER: a high energy-efficient, general-purpose acceleration architecture for string operations in big data processing[J]. *Sustainable Computing: Informatics and Systems*, 2019, 21:129-142

[25] Jeon D I, Park K B, Chung K S. HMC-MAC: processing-in memory architecture for multiply-accumulate operations with hybrid memory cube[J]. *IEEE Computer Architecture Letters*, 2018,17(1): 5-8

[26] Niccol C. A dataflow processing chip for training deep neural networks[C]// 2017 29th Symposium on High Perfomance Chips, Cupertino, USA, 2017 :1-11

[27] Pham P H, Jelaca D, Farabet C, et al. NeuFlow: dataflow vision processing system-on-a-chip[C]//2012 IEEE 55th International Midwest Symposium on Circuits and Systems (MWSCAS), Boise, USA, 2012:1044-1047

[28] Solinas M, Badia R M, Bodin F, et al. The TERAFLUX project: exploiting the dataflow paradigm innext generation teradevices[C]// 2013 Euromicro Conference on Digital System Design, Los Alamitos, USA, 2013:272-279

[29] Carter N P, Agrawal A, Borkar S, et al. Runnemede: an architecture for ubiquitous high-performance computing[C]//19th IEEE International Symposium on High Performance Computer Architecture, Shenzhen, China, 2013:198-209

[30] Burger D, Keckler S W, McKinley K S, et al. Scaling to the end of silicon with EDGE architectures[J]. *IEEE Computer*, 2004,37(7): 44-55

[31] Swanson S, Schwerin A, Mercaldi M, et al. The wave scalar architecture[J]. *ACM Transactions on Computer Systems*, 2007, 25(2): 4:1-4:54

[32] 申小伟, 叶笑春, 王达,等. 一种面向科学计算的数据流优化方法[J]. 计算机学报, 2017,40(9):223-238

[33] Shen X W, Ye X C, Tan X, et al. POSTER: an optimization of dataflow architectures for scientific applications[C]// Proceedings of the 2016 International Conference on Parallel Architectures and Compilation, Haifa, Israel, 2016:441-442

[34] Jeon D I, Chung K S. CasHMC: A Cycle-Accurate Simulator for Hybrid Memory Cube[J]. *IEEE Computer Architecture Letters*, 2017,16(1): 10-13

[35] Binket N L, Beckmann B M, Black G, et al. The gem5 simulator[J]. *SIGARCH Computer Architecture News*, 2011,39(2): 1-7

[36] Pawlowski J T. Hybrid memory cube (HMC)[C]//2011 IEEE Hot Chips 23 Symposium (HCS), Stanford, USA, 2011:1-24

NDP-Ledger: a high-throughput general-purpose acceleration architecture for blockchain applications

An Shuqian[*,**], Li Wenming[*], Fan Zhihua[*,**], Wu Haibin[*], Wu Meng[*], Wang Da[*], Zhang Hao[*], Tang Zhimin[*,**]

([*] State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100190)

([**] University of Chinese Academy of Sciences, Beijing 100049)

Abstract

Blockchain technology is widely used in the fields of digital currency, payment due to its features of decentralization and tamper-proof. The algorithms require high computing power and storage access capacity, resulting in low energy efficiency of traditional von Neumann structures in blockchain applications. The 3D stacked memory, which alleviates the bottleneck of von Neumann structure, has become one of the hot research fields. Based on the 3D stacked memory technology and data flow execution mode, a high-throughput near-data-processing (NDP) architecture for blockchain applications (NDP-Ledger) is proposeed. It analyzes and demonstrates the computing characteristics of blockchain applications and the adaptability of 3D stacked storage technology in blockchain applications. A general-purpose high-concurrency blockchain acceleration architecture is designed based on the dataflow execution mode, which can improve the processing performance on the premise of meeting the universality. The evaluation results show that the general-purpose acceleration architecture of the blockchain proposed has better performance in typical blockchain application processing than CPUs and GPUs.

Key words: blockchain, 3D stacked memory, near-data-processing (NDP), dataflow, general-purpose acceleration architecture