·计算机系统前沿技术·　　文章编号：1000-3428(2023)12-0010-15　　文献标志码：A　　中图分类号：TP391

面向E级超算系统的众核片上存储层次研究

方燕飞，刘　齐，董恩铭，李雁冰，过　锋，王　谛，何王全，漆锋滨

（国家并行计算机工程技术研究中心，北京 100190）

摘　要：当前众核已成为构建高性能计算（HPC）超级计算机的主流微处理器架构，为HPC领域E级超算提供强大的算力。随着众核处理器片上集成的运算核心数量不断增加，众多核心对存储资源竞争愈加激烈，“访存墙”问题越来越突出。众核片上存储层次是缓解“访存墙”问题并帮助HPC应用更好地发挥众核处理器的计算优势以提升实际应用性能的重要结构。众核片上存储层次的设计对众核片上系统性能、功耗和面积具有重要影响，是众核结构设计中的重要环节，也是业界的研究热点。由于众核芯片发展历史和片上微体系结构设计技术的不同，以及所面向的应用领域需求不同等原因，目前的HPC主流众核片上存储层次结构并不单一，但从横向比较和各处理器自身纵向发展趋势，以及从HPC与数据科学、机器学习不断融合发展带来的应用需求变化来看，SPM+Cache的混合结构最可能成为今后HPC E级超算系统众核处理器片上存储层次设计的主流选择。在面向E级计算的软件和算法层面，开展针对众核存储层次特点的设计与优化，可以帮助HPC应用更好地发挥众核处理器的计算优势，从而有效提升实际应用性能，因此面向众核片上存储层次特点的软件及算法设计与优化技术也是业界的研究热点之一。首先按照不同的组织方式将片上存储层次分为多级Cache结构、SPM结构和SPM+Cache混合结构，并总结分析3种结构的优缺点。然后分析国际主流GPU、同构众核、国产众核等面向主流E级超算系统的众核处理器片上存储层次设计现状与发展趋势。最后从众核LLC管理与缓存一致性协议、SPM空间管理与数据移动优化、SPM+Cache混合结构的全局视角优化等角度综述国际上的存储层次设计与优化相关软硬件技术的研究现状。在此基础上，从软硬件及算法设计等不同角度展望了片上存储层次的未来研究方向。

关键词：E级超算；众核处理器；存储层次；高性能计算；便签式存储器；末级缓存

开放科学（资源服务）标志码（OSID）：

中文引用格式：方燕飞，刘齐，董恩铭，等. 面向E级超算系统的众核片上存储层次研究[J]. 计算机工程，2023，49(12)：10-24.

英文引用格式：FANG Y F，LIU Q，DONG E M，et al. Research on manycore on-chip storage hierarchy for exascale supercomputer systems[J].Computer Engineering，2023，49(12)：10-24.

Research on Manycore On-chip Storage Hierarchy for Exascale Supercomputer Systems

FANG Yanfei，LIU Qi，DONG Enming，LI Yanbing，GUO Feng，WANG Di，HE Wangquan，QI Fengbin

（National Research Center of Parallel Computer Engineering and Technology，Beijing 100190，China）

【Abstract】Manycore has become the mainstream processor architecture for building HPC supercomputer systems，providing powerful computing power for High Performance Computing（HPC）exascale supercomputers. With the increasing number of cores integrated on manycore processor chips，the competition for large-scale cores for memory resources has become more intense. Manycore on-chip memory hierarchy is an important structure that alleviates the "memory wall" problem，aids HPC applications better play the computing advantages of manycore processors，and improves the performance of practical applications. The design has a significant impact on the performance，power consumption，and area of an on-chip system. The design of a many-call on-chip memory hierarchy has a significant

基金项目：国家重点研发计划（2021FYB03011100）。

作者简介：方燕飞（1980—），女，高级工程师、硕士，主研方向为软件体系结构；刘　齐，工程师、硕士；董恩铭、李雁冰，工程师、博士；过　锋，副研究员、博士；王　谛，工程师、博士；何王全、漆锋滨，研究员、博士。

收稿日期：2022-12-16　　**修回日期：**2023-02-14　　**E-mail：**flyyaj@163.com

impact on the performance, power consumption, and area of manycore systems. It is an important part of the structural design of manycore systems and is a research interest in the industry. Owing to the differences in the development history of manycore chips, the design technology of on-chip microarchitecture, and the different requirements of the application fields, the current HPC mainstream manycore on-chip storage hierarchy is different; however, from the perspective of horizontal comparison and the vertical development trend of each processor, as well as from the changes in application requirements brought by the continuous integration and development of HPC, data science, and machine learning, the hybrid structure of the SPM+Cache would most likely become the mainstream choice for the on-chip storage hierarchy designs of manycore processors in HPC exascale supercomputer systems in the future. For exascale computing software and algorithms, the designs and optimization based on the characteristics of the manycore memory hierarchy can aid HPC applications benefit from the computing advantages of manycore processors, thus effectively improving the performance of practical applications. Therefore, software, algorithm design, and optimization technology for the characteristics of the manycore on-chip storage hierarchy is also a research interest in the industry. This study first partitioned the on-chip memory hierarchy into multilevel Cache, SPM, and SPM+Cache hybrid structures according to different organizations, and then summarized and analyzed the advantages and disadvantages of these structures. This study analyzed the current status and development trend of the memory hierarchy designs of the chips of mainstream exascale supercomputer systems, such as the international mainstream GPU, homogeneous manycore, and domestic manycore. In summary, the research status of software and hardware technologies is related to the design and optimization of the memory hierarchy from the manycore of the manycore LLC management and cache consistency protocol, SPM management and data movement optimization, and the global perspective optimization of the SPM+cache hybrid architecture. Thus, this study looks forward to the future research direction of on-chip memory hierarchy based on different perspectives, such as hardware, software, and algorithm designs.

【Key words】 Exascale supercomputer; manycore processor; storage hierarchy; High Performance Computing (HPC); Scratchpad Memory (SPM); Last Level Cache (LLC)

DOI: 10. 19678/j. issn. 1000-3428. 0066548

0　概述

随着计算机科学技术的高速发展，处理器计算能力不断提升，超级计算机系统规模不断扩大，高性能计算（High Performance Computing，HPC）开始迈入E级（Exascale，E为超级计算机运算单位，意为每秒百亿亿次）时代。在同一芯片内集成大量计算核心的众核处理器[1-4]凭借超高的性能功耗比和性能面积比成为超算领域的主流处理器架构，2022年12月第60届TOP500[1]榜单上的前10台超算系统均采用众核处理器为系统提供主要算力。

随着数据科学和机器学习方法在HPC应用领域的渗透和深度融合，HPC应用对超算系统的运算能力和密集数据处理能力都提出了越来越大的需求。而随着工艺和体系结构技术的发展，面向E级超算系统的众核处理器计算密度不断提升，片上集成的运算核心数量不断增加，众多核心对存储资源的竞争愈加剧烈，众核处理器面临的“访存墙”问题越来越突出。E级超算众核片上存储层次是缓解E级系统内众核处理器“访存墙”问题，帮助HPC应用更好地发挥E级系统众核处理器强大的计算优势，提升实际应用性能的重要结构。然而，片上存储器在整个处理器上的面积占比超过50%，功耗占比约为25%~45%[2]，而且存储容量非常有限。因此，如何充分利用E级超算系统众核片上面积，合理设计存储层次，并从面向E级计算的应用与软件层面最大限度地发挥众核片上存储层次的优势，是工业界和学术界的研究热点。

片上存储层次有多种组织方式，传统多核处理器通常采用多级高速缓存（Cache）结构，而嵌入式处理器通常采用便签式存储器（Scratchpad Memory，SPM）结构来缓解“访存墙”问题。Cache具有硬件自动管理、性能优化显著等优点，但随着晶体管数量的日益增长，Cache体系结构在现代众核处理器中变得更加复杂，在面积、性能、功耗等方面都面临挑战，而且Cache具有程序执行时间不可预知、软件不可控等缺点。SPM作为Cache的替代或补充，在能源效率、时间可控性和可扩展性等方面具有明显优势，但其缺点是编译器或程序员需要显式地控制数据布局与传输，对于软件优化和用户编程具有很大挑战。为了充分利用Cache和SPM的优势，在性能、功耗、

面积、可编程性等方面取得更好的平衡，面向E级超算的众核片上存储层次的结构变得比多核和嵌入式处理器的存储层次都更加丰富且复杂，软硬件及应用层面都有许多关键技术值得研究和突破。

本文按照不同的组织方式将片上存储层次分为多级Cache结构、SPM结构和SPM+Cache混合结构，并总结3种结构的优缺点。分析国际主流GPU、同构众核、国产众核等面向主流E级超算系统的众核处理器片上存储层次设计现状与发展趋势，并从众核共享末级Cache（Last Level Cache，LLC）管理与缓存一致性协议、SPM空间管理与数据移动优化、SPM+Cache混合结构的全局视角优化等角度综述国际上的存储层次设计与优化相关软硬件技术的研究现状。

1 众核片上存储层次特点分析

在过去的近二十年中，随着集成电路设计与制造技术的快速发展，处理器频率以每年50%~100%的速度增长，然而主存的速度增长每年只有6%~7%[5]，处理器与主存速度差异越来越大。为了缓解处理器和主存之间的速度差，在寄存器与主存之间引入了一种比主存速度更快、容量更小的静态随机存取存储器（SRAM）作为临时存储，形成片上层次化存储结构，也称作片上存储层次。E级超算系统采用包括寄存器、高速缓存、主存、外存的多级存储体系结构。众核处理器作为E级系统的计算节点，其片上存储结构是E级超算系统存储体系结构中的一个重要组成部分，是缓解众核处理器“访存墙”问题，帮助HPC应用更好地发挥众核处理器计算优势，提升实际应用性能的重要结构。在当前主流众核处理器中，片上存储层次形式多样，按照SRAM的组织方式可以分为多级Cache结构、SPM结构和SPM+Cache的混合结构等。

1.1 多级Cache结构

在通用处理器上，SRAM通常被组织为Cache。在通用多核处理器上常有3~4级Cache，不同层次的Cache，其容量、延时和功率不同，从主存到寄存器，越往上Cache的层次越低，容量越小，访问延时越低。比如IBM的Z15[6]，每个核心私有L1数据Cache和指令Cache，一个数据和指令共用的L2 Cache，所有核共享一个大容量的L3 Cache，此外，还有一个更大容量的片外L4 Cache。但随着片上核心数量的不断增加，受功耗面积等制约，多核Cache的层次在减少，比如下一代IBM Telum[7]，将取消L4和L3 Cache，物理上只支持L1和L2 Cache，同时借助更高级的技术实现逻辑上的共享L3 Cache。在当前主流的众核处理器上，由于片上核心数量众多，片上互连结构复杂，为更有效挥地发Cache的作用，众核大多采用2级Cache结构，如图1所示，各核心私有L1 Cache，L2 Cache作为末级Cache被所有核心共享，而且L2 Cache大多容量较大，由多个Bank组成，通过高速NoC互连实现核心间共享。

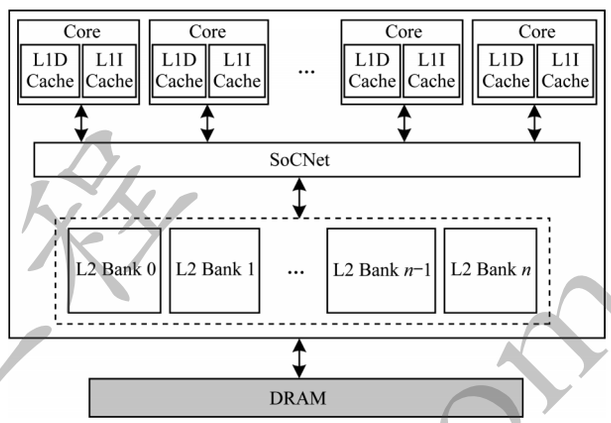

图1 众核处理器上的两级Cache存储结构

Fig.1 Two level Cache memory hierarchy on manycore processor

1.2 SPM结构

Cache的设计越来越复杂，且受到面积和功耗的限制，而与Cache相比，SPM的功耗和面积比Cache低40%以上[5]，因此很多众核处理器将片上SRAM组织成由软件管理的便签式存储器。例如早期由IBM、Sony和Toshiba共同开发的Cell[8]处理器，由1个跟Power处理器（PPE）兼容的主处理器和8个协处理器（即SPE）组成，其中每个SPE带有256 KB SPM。又如Adapteva Epiphany[9]的64核心众核处理器，每个计算核心带有软件管理的SPM。再如“神威·太湖之光”超算系统采用的申威高性能众核处理器SW26010[10]，由4个控制核心和256个从核心组成，每个从核心带有160 KB的SPM。通常，片上的多个计算核心之间通过片上网络（NoC）互连，每个核心除了可以访问本地SPM之外，还可以通过NoC访问远程的SPM，并且访问延迟与核心间的距离成正比。众核片上核心访问远程核心SPM通常有两种方式：一种是由底层体系结构支持的共享SPM方式；另一种是通过RMA等方式显式地访问远程核心的SPM。前者对用户透明，后者需要调用RMA接口显式编程。有的处理器上的核心还配备直接内存访问（DMA）引擎，用于在片外内存和本地SPM之间传输批量数据。图2显示了基于SPM的众核架构[11]存储结构示意图。每个核心包含一个统一的SPM用于指令和数据，每个SPM持有应用程序要使用的全局地址空间的不同部分。虽然SPM

可以同时作为代码和数据的片上缓存,但在高性能计算领域,为了减少对代码管理的复杂度,众核处理器通常会在片上提供指令Cache,而只有数据是采用SPM结构进行缓存,典型的如SW26010。本文后续讨论的众核片上存储层次也默认都是有指令Cache的。

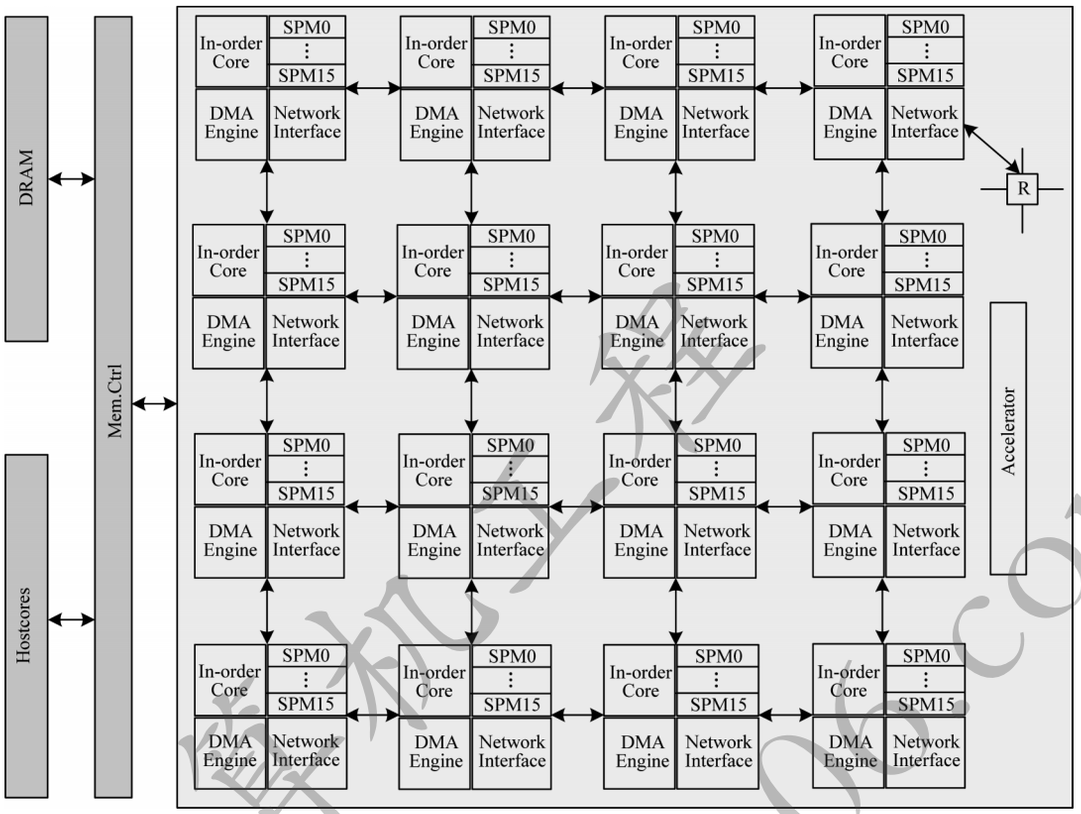

图2　众核处理器上的SPM存储结构

Fig.2　SPM storage structure on manycore processor

随着SPM逐渐被用在GPU、通用处理器及高端处理器上以代替一级数据Cache,在一些高性能计算机体系结构中,甚至采用多级软件管理存储层次代替传统多级Cache存储层次[12]。

1.3　SPM+Cache混合结构

由于传统的SPM控制器不包含任何辅助管理数据的逻辑电路,SPM中的所有数据必须经由软件显式管理,没有管理逻辑电路带来的额外代价,在相同容量下,SPM面积和单次访问的能耗都小于Cache。Cache由于在发生冲突等情况下访问时间会大幅增加,因此Cache具有不确定性的命中时间,SPM无法像Cache那样由硬件自动完成缓冲数据的换入换出操作,因此SPM不存在访问缺失的情况,始终能够保证一个很小的固定时钟周期数的访问时间,从而保证了程序的确定性。但是SPM的容量有限,而且由于不能自动完成数据从主存到SPM的映射,因此对编译器和用户编程带来了很大挑战。表1给出了Cache与SPM特点的对比,两者各有优缺点,而且优势互补。

表1　Cache与SPM的特点对比

Table 1　Comparison of features of Cache and SPM

方式与参数	Cache	SPM
管理方式	硬件管理	软件管理
寻址方式	复杂	简单
编程难度	低	高
访问命中速度	数拍	数拍
访问不命中速度	从数十到数百拍	—
相同容量面积	大	小
相同容量功耗	高	低
相同容量性能	易获得、难控制	为追求极致性能提供可能

为了更好地适应不同应用的需求,充分利用Cache与SPM的优势,越来越多的众核处理器片上存储结构开始采用SPM+Cache混合的方式来追求一种平衡设计。NVIDA公司面向通用计算领域的GPU上的存储层次就是SPM+Cache混合的典型代表,从其第二代GPU架构Fermi[13]开始,为了提升GPU的计算效率,引入了L1和L2 Cache,SPM结构的共享存储器(Shared Memory,SMEM)容量也变得

更大，图3给出了典型的Fermi架构的存储层次，每个SM有一个私有的L1 Cache和SMEM，容量分别为48 KB或16 KB，所有流式多处理器（SM）共享768 KB L2 Cache。自Fermi之后的每一代架构都继承了Cache+SPM的混合结构。NVIDIA GPU一直保留SMEM功能，由软件负责实现数据的读写和一致性管理，有利于CUDA多线程间数据重用，对于图形计算等数据流处理类应用可以挖掘极致的性能。而后引入的两级Cache结构，使GPU在HPC领域的适应性明显加强，没有使用SMEM的应用程序以及无法预知数据地址的算法，都可以从L1 Cache设计中显著获益。

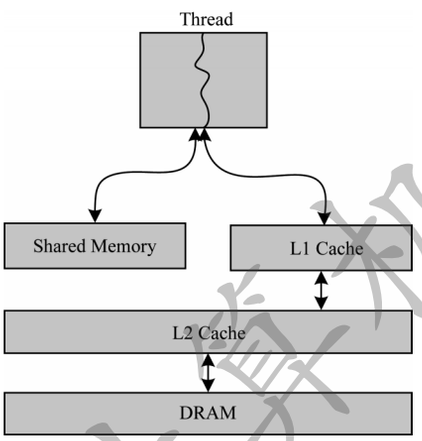

图3 Fermi存储层次示意图

Fig.3 Schematic diagram of Fermi storage hierarchy

1.4 众核片上存储层次结构类型总结

众核处理器系统根据其片上存储层次中是否包含SPM可以分为4类：1）没有SPM的纯Cache系统；2）只有SPM的不可Cache系统；3）拥有SPM和Cache的混合系统；4）SPM和Cache混合且容量可配置的系统，可以配置成纯SPM模式、纯Cache模式、或者SPM与Cache混合的模式，而且SPM与Cache的容量可以有多种组合配置。典型的混合存储结构为核心私有1级Cache和SPM（容量可配置），所有核心共享MB级大容量L2 Cache的结构。

片上存储层次的设计除了组织方式外，还有一个必须关注的物理部件，即片上高速互连网络NoC（如富士通A64FX[3]、申威众核处理器均设计了NoC）。片上高速网络增强了核间数据交互能力，间接提高了单核心Cache或者SPM的数据重用能力，降低了主存访问压力，是硬件高效组织共享末级Cache和共享SPM的关键，也是应用算法和系统软件充分共享利用片上资源的重要途径，因此，在片上存储层次的设计与使用中，必须考虑片上网络的结构特点。

2 面向E级超算的众核处理器片上存储层次

2.1 主流GPGPU存储层次现状

作为通用加速卡的GPGPU凭借其卓越的计算能力，已经在HPC领域扮演着越来越重要的角色。NVIDIA通用GPU作为最主流的通用众核加速卡，已经广泛应用于HPC领域，比如第60届Top500排名前10的超算中有5台都采用NVIDIA的GPU进行加速。而AMD和Intel的通用GPU也分别作为美国的3台E级系统上的加速卡为系统提供主要算力。下文将分别分析上述3个面向HPC的主流GPU架构存储层次现状及其发展趋势。

2.1.1 NVIDIA Hopper架构的存储层次

NVIDIA GPU的架构从Tesla到Hopper已经发展了9代，为了适应GPU计算能力的不断提升，其片上存储层次也在不断发展。自Fermi架构采用SPM+Cache混合的片上存储结构之后，后续的架构在存储层次上的设计没有较大的变革，其变化主要表现在各级存储在容量上的变化、SMEM与L1 Cache结合方式上的变化等。直至第7代Volta架构，其片上存储方式较前几代架构又有了比较重大的变化。Volta架构将数据Cache与SMEM从功能上做到一个存储体（memory block）中，使得两种缓存方式均达到了其访问的最好性能。SMEM与L1 Cache的总容量为128 KB，且容量可以有多种灵活的配置。第8代Ampere架构的片上存储方式继承了Volta架构的片上存储方式，但将存储空间总大小扩大至192 KB/SM（采用Ampere架构的典型GPU为A100[14]）。A100的L2 Cache容量扩展到40 MB，是V100的7倍，并且支持驻留优先级设置，可以提高持续访问数据驻留Cache的权利，从而减小DMA带宽需求。此外，Ampere增加了异步拷贝指令，可以直接从L2 Cache以及DRAM中批量传输数据到SMEM中，而在前一代架构Volta中，需要通过一个全局加载指令从L1 Cache到寄存器中，然后再用Store-Shared指令从寄存器中传到SMEM中，图4展示了Ampere与Volta的存储结构变化。从最新一代Hopper架构H100和Ampere架构A100的对比来看，最近的两代存储架构方面没有大的变化，只是H100的SLM容量增加到256 KB，且H100从硬件对TMA（Tensor Memory Accelerator）的功能上做了更多的支持。

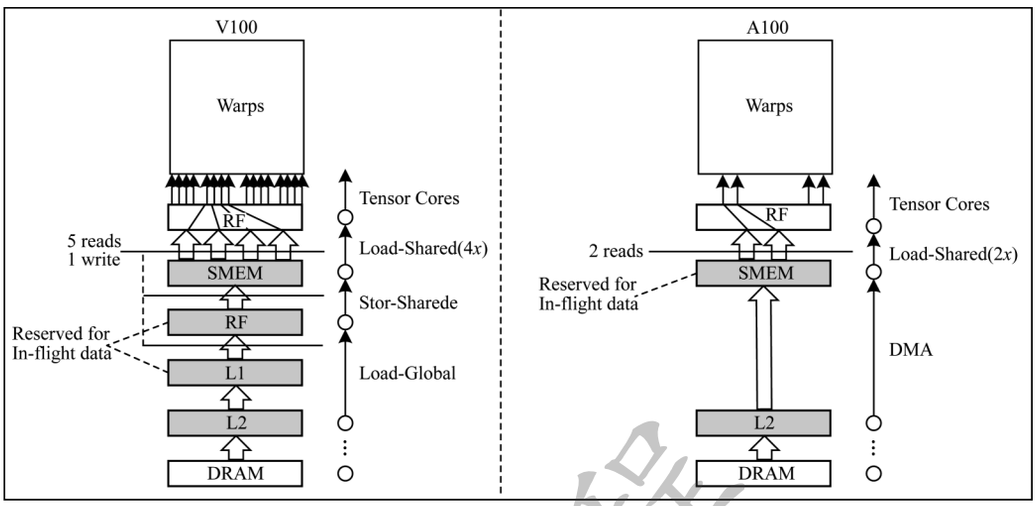

图4　Ampere与Volta的存储结构变化

Fig.4　Storage structure changes of Ampere and Volta

从NVIDIA 9代GPU架构的存储层次发展趋势来看,Cache+SPM的混合结构是其片上存储层次的一个创新解决方案,混合结构可以更好地兼顾面积、性能、功耗及好用性,以满足不同领域应用的多样化需求。对于某些应用,Cache仍然是保证系统通用性和应用性能的关键结构,而SPM结构为某些HPC应用挖掘极致性能提供了更多可能。对于有些更复杂的应用,则既需要SMEM也需要Cache,因此支持Cache与SMEM容量的灵活配置,可以进一步提高对不同类型应用的适应性。进一步地,为方便程序员或者编译系统挖掘应用数据访问特性,Hopper架构引入了线程块集群(Thread Block Cluster,TBC)和分布共享内存(Distributed Shared Memory,DSMEM)概念,同一线程块集群内的SM之间可以共享访问SPM,而无须通过Global Memory交互数据,从而提高数据交互效率。此外,硬件上提供了只读数据缓存的Read-Only Data Cache以及异步拷贝指令,都可以进一步降低访存时延、提升带宽利用率,虽然这些设计会增加软件管理和编程复杂度,但是为应用或编译系统充分挖掘片上缓冲性能提供了更多手段。

2.1.2　AMD CDNA2架构的存储层次

AMD CDNA2是AMD在2021年11月新发布的GPU架构,其设计目标是科学计算和机器学习领域的应用。最新TOP500排名第1的美国E级超算系统Frontier采用的加速卡AMD MI250X正是基于AMD CDNA2架构的AMD Instinct MI200构建的。AMD MI250X采用MCM技术封装了两个AMD Instinct MI200 GCD。

每个AMD Instinct MI200 GCD包含112个计算单元(Compute Unit,CU),最低级别的内存层次结构位于CU内部,如图5所示,每个CU包含一个16 KB L1数据Cache和64 KB SPM结构的本地共享存储器,另有一个4 KB共享SPM结构的全局共享存储器。整个GCD共享一个由32个Slice构成的16路组相连结构的L2 Cache,总容量为8 MB。

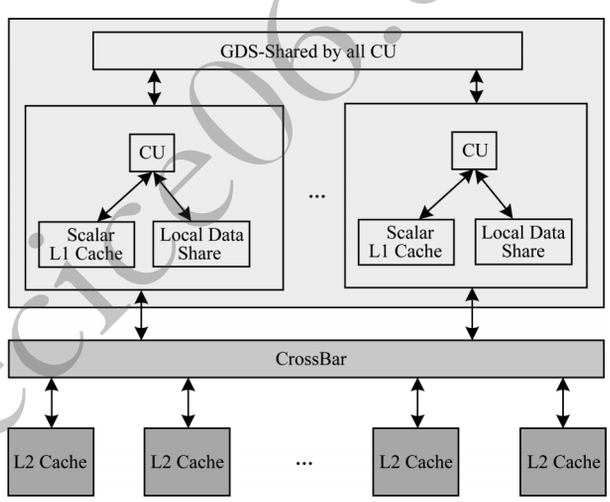

图5　AMD CDNA2片上存储层次

Fig.5　On-chip storage hierarchy of AMD CDNA2

2.1.3　Intel Xe架构的存储层次

美国E级计划中的Aurora超算系统采用Intel Xe架构的GPU,Xe架构是Intel继Gen 9和Gen 11之后的最新一代GPU架构。Gen 9和Gen 11都是采用3级Cache结构,在片上使用共享本地内存(Shared Local Memory,SLM),是一种SPM结构。作为一级数据缓冲,没有通用的L1数据Cache。Gen 11相对Gen 9在存储层次上的改进特点是将SLM带入了子Slice单元,使得SLM更接近执行单元(EU),以便在同时访问L3 Cache时减少数据的争用。SLM更接近EU也有助于减少延迟并提高效率,如图6所示。

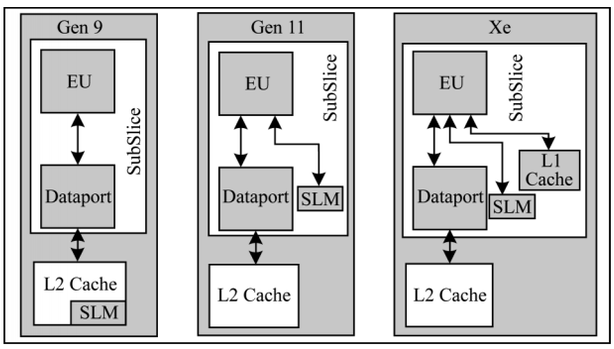

图6 Intel Xe，Gen 9，Gen 11的存储结构变化

Fig.6 Storage structure changes of Intel Xe，Gen 9，Gen 11

Intel的Xe架构是对前两代Gen 9和Gen 11架构上的进一步升级，Intel Xe架构不仅全面提升了执行单元的规模，从存储层次发展来看，Xe架构保留了Gen 11相对Gen 9在SLM上的改进，即将SLM带入SubSlice，并且Xe架构相对Gen 11进一步新增了L1数据Cache。Xe架构还支持端对端压缩，L2 Cache的容量也有大幅提升。目前发布的Intel Xe架构GPU Ponte Vecchio支持512 KB的SLM与L1 DCache动态可配的一级缓冲，二级Cache增加到64 MB。从不同代系的发展来看，在追求兼顾性能与好用性的目标中，Intel的GPU存储层次设计也逐步采用SPM+Cache的混合结构。

2.2 同构众核系统的片上存储层次

目前TOP500排名第2的日本的Fugaku/Post-K超算系统，采用的是富士通A64FX处理器[3]。

A64FX处理器采用同构众核处理器体系结构，片上集成52个同构核心，包括48个计算核心与4个辅助核心，如图7所示，所有核心分为4个组，每组13个核心，一组核心称为一个CMG。在A64FX处理器芯片中，多个CMG通过片上网络连接。富士通的A64FX处理器为了减小芯片大小，采用两级Cache的结构，每个核心私有64 KB一级Cache，一个CMG的所有核心共享一个8 MB二级Cache。A64FX采用类似于环形拓扑网络的片上网络，以支持CMG之间的内存一致性协议，从而可以使用多个CMG执行共享内存程序。

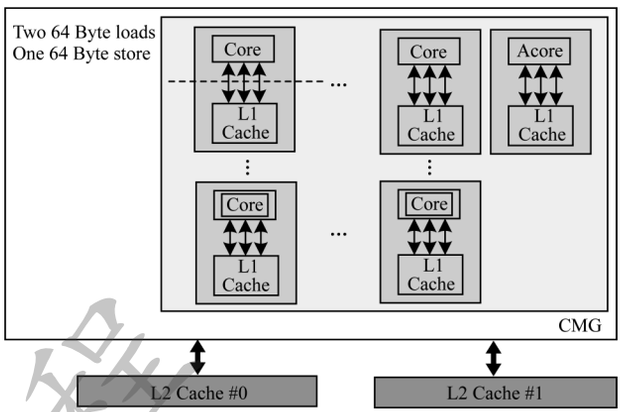

图7 A64FX片上存储层次

Fig.7 On-chip storage hierarchy of A64FX

富岳采用的A64FX存储层次设计是以应用和编程软件需求牵引的一种协同设计。比如其CMG内核心数量、一二级Cache的容量以及Cache line的大小等参数设置，都是在模拟富岳的重点目标应用程序性能的前提下给出的，在功率、面积和性能之间取得更好的权衡。为了提高目标应用程序的性能，采用增强型的缓存功能设计，比如采用组合聚集操作技术，使间接加载的数据吞吐量加倍，采用二级缓存数据重用优化技术提高缓存带宽。纯Cache结构具有更好的应用适应性，但同时也面临可扩展性问题。

2.3 新一代国产众核存储层次现状

2.3.1 MT-3000异构众核处理器存储层次

国防科技大学推出了一款自主设计的面向新一代天河超算系统的异构众核处理器MT-3000[12]。

MT-3000采用多区域加速架构，其结构如图8所示，CPU内包含16个通用处理器核心和4个加速域，采用混合存储器层次结构。

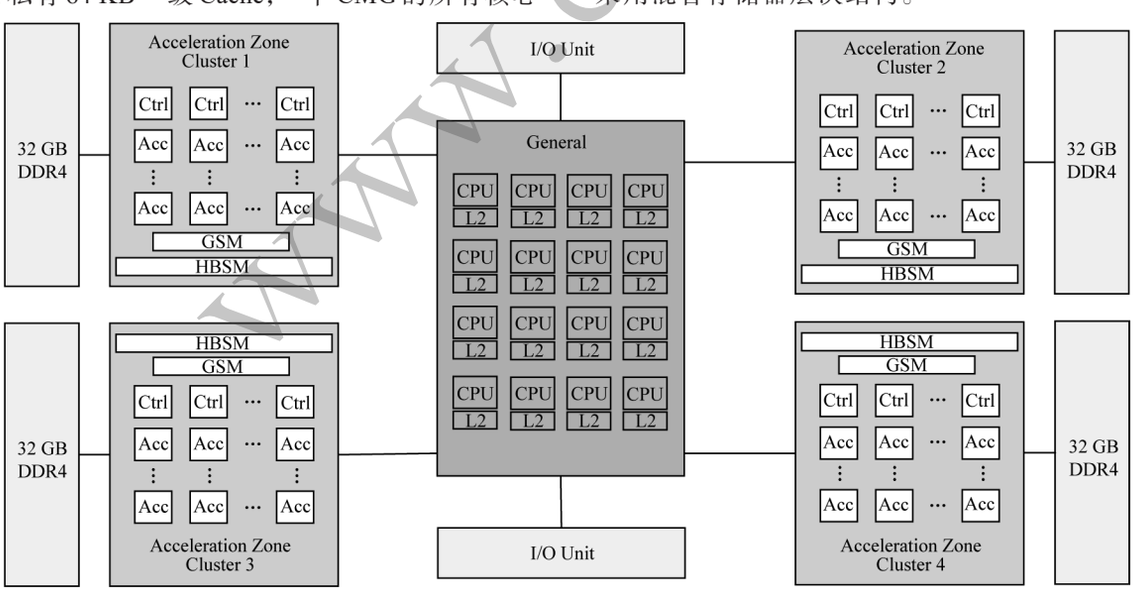

图8 GPDSP存储层次示意图

Fig.8 Schematic diagram of GPDSP storage hierarchy

MT-3000的每个通用CPU核含有支持高速缓存一致性的两级私有Cache,L1 Cache被L2 Cache包含,L2 Cache容量为512 KB。每个加速域包含24个控制核心和384个加速核心,采用两级SPM结构,其存储层次如图8所示,每个加速域含有1.5 MB的域内共享内存(DSM)和48 MB的高速共享内存(HBSM),其中DSM只能由加速域内的核心访问,而HBSM被本加速域的所有核心及通用域的16个CPU核共享。片上含有两级高速互连网络,每个加速域内采用全连接网络,而16个通用核心间通过二维Mesh网相连,16个通用核心可以访问4个加速域的HBSM。MT-3000加速域支持软件管理的垂直存储一致性,将加速域内的DSM(6 MB)、HBSM(共48 MB)、DDR4存储(共32 GB)三级存储架构,通过高速DMA和片上网络连接,提供了高带宽的数据传输和核间数据交互能力。

2.3.2　申威众核处理器存储层次

用于"神威·太湖之光"的申威26010众核处理器[10,15],片上采用纯数据SPM的结构。SW26010包含4个核组,每个核组有64个运算核心,每个运算核心私有128 KB SPM。而面向E级系统的新一代申威众核处理器[16]开始采用SPM+Cache混合的片上存储结构。该众核处理器包含6个核组,每个核组有64个运算核心,每个运算核心有256 KB SRAM,支持SPM与L1 Cache混合且容量可配置的使用方式,64个运算核心的SRAM还可以通过片上NoC互连组织成共享SPM空间,片上存储模型如图9所示。运算核心可以RMA或者远程load/store的方式访问同一核组内远程运算核心的SPM空间,也可以通过DMA将主存空间数据批量传输到私有或者共享的SPM空间上。

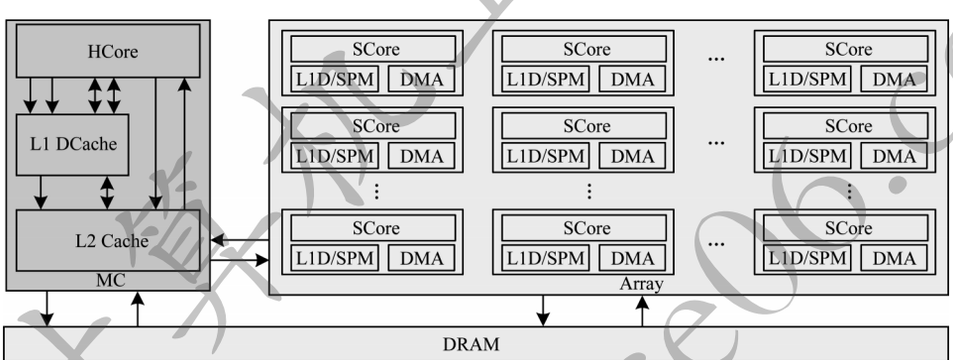

图9　新一代申威众核处理器存储层次

Fig.9　The storage hierarchy of the new generation of Shenwei manycore processors

为更好地适应不同数据访问特征的HPC应用的需求,提升主存数据的访问效率,系统支持主存空间访问可以配置成可Cache和不可Cache两种属性。但为了降低编程复杂度,这种属性通常对用户透明,用户程序主存数据默认都是可Cache的,编译器或者运行时可以利用主存不可Cache空间结合DMA进行访存优化,减少维护Cache一致性的开销。

2.4　面向E级超算的众核处理器片上存储层次结构类型总结

表2以众核计算核心的视角总结了各主流众核处理器的片上存储层次。富岳A64FX同构众核采用两级Cache结构,MT-3000异构融合架构中的加速域采用纯SPM结构,而NVIDIA、Intel、AMD的面向E级系统的GPU以及申威的最新一代异构众核处理器SW260x0均采用SPM+Cache混合的结构。可见目前的HPC主流众核片上存储层次结构并不单一,这是由多种因素导致的,不同处理器面向的应用领域需求不同,追求的设计目标不同,各自的技术积累也不同。但是,从横向的比较和各处理器自身纵向的发展趋势,以及从HPC与数据科学、机器学习不断融合发展带来的应用需求变化来看,SPM+Cache的混合结构最可能成为今后HPC众核处理器片上存储层次设计的主流选择。另外,从表2最后一列的分析可以看出,虽然主流GPU都采用了的SPM+Cache结构,但每一款GPU的SPM和Cache的功能细节也各有不同,例如NVIDIA H100的L1只能为Cache CUDA对应的Local空间的数据,Global空间数据只能缓存进L2 Cache。H100还支持同一TBC内SM远程访问共享SPM,新的存储层次特性也给HPC编程带来新挑战。而AMD MI250X GPU支持的CU私有SPM和L1 DCache是SIMD单元专用的,如果HPC程序员要深度优化程序性能,则必须要充分利用好这些存储结构特性。

表2 主流众核处理器片上存储层次

Table 2 On-chip storage hierarchy on mainstream manycore processor

型号架构	类型	私有SPM	共享SPM	L1 Dcache	L2 Dcache	特点分析
H100 (NVIDIA Hopper)	GPU	0~256 KB可配/SM	GPC内共享SPM	0~256 KB可配/SM	50 MB(LLC)	支持SPM+L1 DCache动态配置和大容量LLC，L1只能Cache Local空间的数据，Global空间数据只能缓存进L2 Cache，支持同一TBC内SM远程访问共享SPM，新的存储层次特性也给HPC编程优化带来新挑战
MI250X (AMD CDNA2)	GPU	64 KB(向量)/CU	4 KB(向量)/GCD	16 KB/CU	16 MB(LLC)	支持CU私有向量的SPM和L1 Dcach、GCD内共享SPM、LLC等多种功能的缓存，HPC应用需要相应的编程控制以获得性能增量
Ponte Vecchio (Intel Xe)	GPU	0~512 KB可配/Xe-core	无	0~512 KB可配/Xe-core	408 MB(LLC)	支持SPM+L1 DCache动态配置，兼顾性能与应用适应面，超大容量LLC提供更高访存带宽，但用好SPM仍然需要编程控制
A64FX	同构众核	无	无	64 KB	8 MB(LLC)	多级Cache，支持横向Cache一致性，应用适应性好，但硬件设计面临可扩性问题
MT-3000	异构众核	无	1.5 MB+48 MB/加速域	无	无	两级共享SPM，没有Cache，不支持统一的内存空间，4个加速域的数据交互需要借助通用核完成，异构编程控制较复杂
SW260x0	异构众核	32~160 KB可配/单核心	0~8 MB可配/单核心	32~128 KB可配/单核心	无	支持SPM+L1 DCache动态配置，没有LLC、L1 DCache可缓冲所有主存空间数据，但缓冲共享主存空间时需要用户保证共享一致性，支持SPM到SPM的批量数据传输是其一大特点，但需要用户显式编程控制

3 面向众核片上存储层次的优化技术

众核是构建E级超算系统的主要计算节点，为E级超算系统提供主要算力来源。如何充分发挥众核处理器片上众多核心的计算能力，一方面需要硬件层面开展众核片上存储结构设计，另一方面需要面向E级计算的应用和软件开展相关优化技术，以充分利用众核片上存储层次特点。当前面向E级的主流众核处理器采用的片上存储层次结构多样，各具优缺点，本文从多级Cache、SPM以及SPM+Cache混合3种结构分别探讨当前主流软硬件设计与优化技术。

3.1 多级Cache的优化

随着片上核心数量的不断增大，片上互连结构日趋复杂，要使Cache结构在片上众多核心竞争的环境下能继续高效工作，必须解决好两方面问题：共享末级Cache(LLC[11])的高效管理问题和Cache一致性问题。

3.1.1 LLC的管理策略

众核片上典型的Cache结构为：核心私有一级Cache，L2 Cache作为LLC由所有核心共享。在众核环境下LLC会同时接收来自于多个线程的访存请求，而线程间的数据访问特征有可能各不相同，大量并发线程间的相互干扰会影响LLC的使用效率。因此，如何管理片上LLC资源是影响众核处理器访存性能的关键因素。传统LLC的替换与插入策略的研究已经比较成熟，包括使用LRU作为替换策略的插

入策略、动态插入策略(DIP)、线程感知的DIP策略,还有后续提出的静态重引用间隔预测策略、动态重引用间隔预测策略和线程感知的动态重引用间隔预测策略。但是,这些传统的LLC管理策略在众核系统下已无法高效工作。文献[17-20]提出了在CMP的竞争线程之间动态划分共享LLC的策略,文献[21-22]则进一步考虑了多线程间的共享数据行为的影响,并将其融入到LLC的管理策略。上述技术大多针对同构众核系统,而文献[23-24]针对异构众核系统开展了LLC管理策略研究。文献[25]在前人基础上提出了适合片上异构众核处理器的动态LLC替换策略DIPP(Dynamic Insertion/Promotion Policy)。该替换策略核心思想是通过限制GPU核能够获取的Cache资源,达到降低程序间的线程干扰,降低程序的失效率和提升系统整体性能的目的。

随着CMP上处理器核心数量的不断增大,非一致Cache体系结构(NUCA)成为当前CMP上大容量低延迟Cache的主要组织结构。NUCA结构中的LLC由于容量较大,通常分为多个Cache存储体(bank),所有bank置于芯片中央,通过NoC在逻辑上组织成一个被所有核心共享的统一的末级Cache。NoC是组织LLC的重要互连方式。随着NoC规模的扩大,一方面,网络延迟正在成为缓存访问延迟的主要来源,另一方面,不同核心之间的通信距离和延迟差距正在增大。这种差距会严重导致网络延迟不平衡,加剧缓存访问延迟的不均匀程度,进而恶化系统性能。文献[26]分别针对无冲突延迟和竞争访问提出了非一致性存储映射和非一致链路分布的设计方法。文献[27]提出了一种新的面向公平性和位置感知的NUCA方案,以缓解网络延迟不平衡的问题,实现更统一的缓存访问。文献[28]提出一种位置敏感的LLC,通过感知Cache行位置和利用GPGPU的核间通信来缓解GPGPU片上网络瓶颈,减少访存延迟。文献[29]将LLC淘汰的Cache块保存在NoC的路由器上,当有核心请求这块数据时,由NoC直接响应这个请求。这些研究都充分利用了NoC的特点,为基于NoC的NUCA数据访问的不平衡性问题提供了多种解决方案。

3.1.2 众核片上缓存一致性协议

当多个处理器核的高速缓存保持从主存获取的同一数据对象的本地副本时,即使其中任何一个高速缓存修改了同一数据对象的值,也会导致高速缓存和共享内存之间共享数据的全局视图不一致,这个问题称为缓存一致性。在众核处理器中,为管理共享缓存中的读写操作,以便在所有处理器核心之间维护一致性,而执行的一组特定规则称为缓存一致性协议。缓存一致性协议可分为目录协议[30]和监听协议[31]两大类,基本作用是发现共享数据块的状态。传统的缓存一致性协议设计复杂,开销较大,无法满足众核处理器共享存储系统追求高效、低功耗、可扩展的设计目标。国际上有大量面向片上众核系统低开销可扩展的Cache一致性协议研究。文献[32]提出了可扩展到千核规模的融合一致性Cache,在统一物理内存的基础上,采用两级目录的融合一致性设计。文献[33]提出了SCD框架,它依赖于高效的高关联缓存实现一个可扩展到数千个内核的单级目录,精确地跟踪共享集,并且产生可忽略的目录引起的失效。国内也有许多研究者在Cache一致性协议功能扩展和性能优化等方面开展了研究。文献[34]针对共享存储系统提出一种没有目录和间接访问、没有众多一致性状态和竞争的简单高效的Cache一致性协议VISU,解决了制约目录协议可扩展性的目录开销问题。文献[35-36]提出了一种具有表达力的、区域高效的目录缓存设计,并分别在64、256核系统上进行了评估。文献[37-38]提出了基于时间的硬件一致性协议,即库缓存一致性(Library Cache Coherence,LCC),它通过暂停对缓存块的写入,直到它们被所有共享者自失效为止,从而实现顺序一致性。

当可伸缩一致性在通用芯片多处理器中得到广泛研究时,GPU体系结构具有一系列新的挑战。引入传统的目录协议会给现有的GPU应用程序增加不必要的一致性通信开销。此外,这些协议增加了GPU存储系统的验证复杂性。文献[39]描述了一个基于时间的GPU一致性框架,称为时间一致性(TC),它利用系统中的全局同步计数器来开发一个简化的GPU一致性协议。同步计数器使所有一致性转换(如缓存块失效)能够同步发生,从而消除所有一致性通信和协议争用,达到减少GPU一致性的开销的目的。文献[40]提出CPU-GPU硬件缓存一致性不要同时实现统一共享内存和高GPU性能的思想。文献[41]针对CPU-GPU系统开发了异构系统一致性协议(HSC),以缓解GPU内存请求的一致性带宽效应。文献[42]提出一种软件辅助硬件一致性(SAHC)来扩展Cache一致性以适应异构处理器。系统软件通常具有跨CPU和GPU共享数据模式的语义知识。这些高级知识可用于在异构处理器中跨面向吞吐量的GPU和对延迟敏感的CPU提供缓存一致性。SAHC提出了一种混合软硬件机制,该机制仅在需要时才使用硬件一致性,同时使用软件知识过滤掉大部分不必要的一致性通信。

从上述研究可以看出,众核上的Cache一致性协议设计工作围绕可扩展和低开销的设计目标,针对典型数据共享模式和众核处理器体系结构开展软硬件协同设计,提出的大多是一种有裁减的弱一致性协议。

3.2 SPM结构的优化

SPM结构的特点是完全将数据的布局与传输交给软件,面向SPM结构的众核程序的编程与优化,必须解决好程序数据的布局与传输两个关键环节。

3.2.1 基于SPM的堆栈空间管理

在只有SPM结构的众核系统上，软件必须显式地管理每个核心本地SPM的进出数据，包括全局数据、堆空间数据、栈空间数据的管理。当核心对应的所有数据都能在本地SPM上放下时，程序执行效率非常高。当核心程序对应的数据总量超出本地SPM容量时，必须进行显式数据管理才能使SPM发挥出较好的性能。对全局数据而言，用户通常可以将其分为两种，一种是可以完整放入SPM中的变量，另一种是无法放入SPM空间的全局数据（数据容量超过SPM空间，或者SPM空间已满），用户可以在程序需要数据之前通过DMA引入数据到SPM，在不再需要数据之后将其返回到全局内存。当程序的全局数组总容量超过SPM空间时，程序员需要考虑哪些数组布局在SPM，哪些数组布局在主存并通过DMA进行传输。对于C/C++程序来说，全局静态数据是可以在编译时确定容量的，而其栈空间或者堆空间数据其大小是可变的，并且与输入数据有关，用户通常没有有效的方式来管理堆数据和栈数据。由于堆栈数据的访问是程序中内存访问的重要部分，因此必须有高效的堆栈管理手段来发挥SPM的性能。

基于SPM的存储层次对程序员提出了很高的要求，需要对算法和片上存储层次特点都有非常准确的把握，而且程序实现起来工作量也很大。为此，国际上开展了许多面向软件管理的存储层次的编译优化研究，一类是面向SPM的静态分配技术[43-45]，通过采用启发式或者整数线性规划等算法，确定哪些数据布局进SPM，哪些放置在主存中，还有一类是动态分配技术[46-49]，包括软件Cache技术、基于图着色算法的SPM分配与自动DMA技术等。文献[50-54]专门针对众核系统面向SPM的堆栈空间管理开展了长期深入的研究。在栈空间管理方面，他们最初提出了局存容量受限的众核处理器上的栈数据管理的方案[50]，支持在SPM上的任意空间上管理任何任务的栈数据，并正确管理所有栈指针，但是其管理开销较高，并且管理没有得到优化。在此技术上，文献[54]对栈数据管理做了进一步优化，并减少其管理开销。在堆空间管理方面，文献[51-52]提出了一种半自动的、可扩展的堆数据管理框架，通过提供简单直观的编程接口来帮助用户实现自动管理堆数据，隐藏了软件缓存堆数据的复杂性。之后又在文献[53]中进一步提出一种全自动、高效的堆数据管理框架，通过一个编译和运行时系统，实现在有限的本地内存众核体系结构中自动管理无限大小的堆数据。文献[48]在上述框架基础上进一步开展了3种通用优化，使得堆管理框架更加高效。

3.2.2 基于SPM的片上数据移动

为了更好地发挥SPM的结构优势，HPC领域的主流众核系统片上存储结构设计通常会提供DMA来加强SPM与主存间数据的传输效率，同时利用NoC实现片上众多核心间SPM的互连，增大单核心可见的SPM容量。SPM的容量、DMA访存带宽、NoC通信带宽都是众核系统实现核心数据高效传输与访问的重要资源，复杂的众核应用需要通过对核心数据进行合理高效的划分、映射与传输，才能实现上述片上资源的高效率用。比如通过循环变换等技术可以增加SPM数据的重用率，减少数据进出SPM的频率。核心间基于NoC的数据传输带宽要比从主存到SPM的DMA带宽高很多，可以结合应用访存特征合理分配数据与任务，使得多核心间可通过远程内存访问（Remote Memory Access，RMA）命令或者RLD/RST共享SPM上的数据，发挥NoC的优势，提升SPM数据在片上的重用率，这些方法都可以降低应用程序对单个核心的SPM容量需求，缓解DMA访存压力。上述工作对用户编程和编译器优化提出了巨大挑战。国际上有许多工作围绕SPM结构特点开展高效性与好编程性的研究，从不同角度提出了许多面向SPM高效编程的优化方案、框架、运行时库及编译优化，为用户开发高效的SPM众核程序提供编程优化指导或者编译优化支持。

文献[55]针对SPM用于多线程应用程序时遇到的挑战，提出了协调数据管理（CDM），这是一个编译时框架，可以自动识别共享/私有变量，并将这些变量及这些变量的拷贝（如果需要）一起放置到合适的片内或片外存储器中，同时考虑NoC争用。另外，还开发了一个精确的整数线性规划（ILP）公式以及一个迭代的、可伸缩的算法，用于将多线程应用程序中的数据变量放置在多个核心SPM上。文献[56]提出了一个用于优化异构多核体系结构上SPM和主存之间的多线程数据传输的编译时框架MSDTM，该框架通过应用程序分析和依赖性检查来确定数据传输操作的分配，并通过所设计的性能模型来推导数据传输的最佳粒度。文献[57]提出一个名为UniSPM的框架，使用低开销的递归启发式算法来解决NP-hard的映射问题，为基于NoC的SPM多核多阶段多线程应用程序提供统一的线程和数据映射框架。文献[58]针对众核平台存储层次特点提出了一种基于运行支持库的OpenMP数组私有化的编译优化方法，对可重用数据进行私有化，充分利用有限的SPM资源减少DMA通信，以提升程序执行效率。文献[59]针对“神威·太湖之光”超算系统上支持DMA的SPM存储结构，提出了一种基于带宽感知的OpenCL程序循环平铺方法，对传统的仅带宽和仅容量考虑的循环平铺方法进行了改进，有效提升了带宽利用率和SPM重用。文献[60]基于SPM提出了一种新的GPU资源管理方法EXPARS，通过将寄存器文件扩展到SPM内存，通过未被充分利用的SPM来支持额外的寄存器分配，在逻辑上提供了一个更大的寄存器文件。文献[61]针对申威众核处

理器核心的SPM存储结构特点，开展了基础函数库的内存延迟优化，提出一种有效的自动数据转换方法和一种表查找方法来优化基础函数库的访存延迟。

3.3　SPM+Cache混合结构下的全视角优化

相比纯Cache或者纯SPM结构的存储层次，混合结构具有两者的优势。原本SPM结构下最棘手的程序堆栈空间管理问题在混合结构下可以得到很好的缓解——堆栈空间可以直接放在主存，基于Cache进行缓存。而原本引起严重Cache冲突的数据可以优先布局进SPM，或者利用DMA加SPM缓冲的方式避免数据进入Cache。因此，混合结构下存储层次的优化，重点在于如何联合Cache和SPM开展全视角优化以获得一加一大于二的性能。

国际上针对混合结构的片上存储层次优化开展了一系列研究。文献[62]利用DMA实现SPM数据的动态换入换出，从而避免了传统动态优化技术利用数据Cache实现数据搬运可能造成的对数据Cache的污染问题，并利用类似虚拟内存管理的思想，将部分动态申请且频繁引起数据Cache冲突的堆栈数据和堆数据的连续虚地址空间重定位到SPM中获得系统性能和能耗的收益。文献[63]研究了6种不同的SPM分配算法来优化混合SPM缓存结构的性能或能耗，并通过实验证明感知Cache的SPM分配比不感知Cache的SPM分配具有更好的性能或能耗的结论。文献[4]认为在混合体系结构中可将频繁使用的数据分配给SPM以进行快速检索来降低对缓存的访问频率，因此可将混合SPM+Cache的Cache行更主动地置于低功耗模式，在不显著降低性能的情况下减少更多的泄漏能量。文献[57]提出一种编译器技术将应用程序数据对象映射到SPM-Cache，以整个程序优化为目标，根据SPM和Cache大小以及应用程序的动态需求进行数据分配。文献[64]提出一种Cache和SPM参数可动态配置的可重构片上统一存储器（RcfgMem），通过动态调整Cache的关联度和SPM的容量等片上存储资源的参数，可以适应不同的应用程序或同一应用程序在不同阶段对存储层次的需求，在不损失系统性能的前提下达到降低系统能耗的目的。文献[65]通过一种软硬协同的方式支持SPM到主存的映射，实现SPM与主存间的自动隐式数据传输，减轻用户显式调用DMA传递数据的编程负担，并通过支持重映射和利用SPM压缩存储机制提升SPM空间的利用率，同时又避免了Cache结构下访问TLB所需的延迟开销和Cache不命中开销。

4　未来展望

在众核体系结构日益复杂的趋势下，众核片上存储层次的设计与优化使用面临许多关键技术挑战，需要开展软硬件协同设计。下文从硬件、软件和算法3个层面的未来发展进行展望：

1）硬件设计层面。随着E级系统众核处理器上集成的核心数量越来越多，片上互连能力越来越复杂，传统的多级Cache或者SPM将无法满足新兴众核结构上存储层次的发展需求，众核处理器片上存储层次会越来越丰富。从存储层次结构发展趋势上来看，Cache与SPM混合的结构因其更好的应用适应性而更可能成为未来众核片上存储层次的主流结构。而如何基于片上网络实现众核心间SPM的数据共享、数据批量传输以及可扩展的Cache横向一致性都将是未来片上存储结构的重要研究方向。总之，面向超大规模E级系统的众核处理器片上存储层次的设计需围绕系统的整体设计目标开展，未来非常有必要针对大量目标应用场景进行数据访问特征分析，并结合片外内存访问带宽、片上网络结构、片上可用面积、整体功耗控制等条件开展平衡设计。

2）软件设计层面。从编译优化角度来看，可以对应用程序核心数据容量、数据访问粒度和重用情况开展分析，并建立好的访存收益评估模型，为数据分配合理的实际地址空间（SPM或者主存）以及为主存数据访问选择高效的传输方式（DMA、RMA、Cache模式）；在编程语言设计角度，可以为专家级用户提供显式的存储层次描述手段，方便用户直接控制核心数据的布局与传输，或者指导编译器开展相应的编译优化与程序变换工作，从而提升众核片上存储层次的易编程性与高效性；在编程框架设计角度，可以针对领域应用特征，开发多平台统一的编程框架接口，底层针对不同的众核处理器片上存储层次特点，开展具体的优化实现，从而提升应用面向不同片上存储结构的性能可移植性；在运行时支撑库的角度，应该针对目标平台的存储层次特点开展领域典型算子库的设计与实现，为应用程序开发提供平台相关的高效API接口支撑。

3）算法与程序设计层面。就HPC用户层面而言，需要针对具体的目标平台上的存储层次特点，结合应用核心运算特征，开展高效的众核并行算法模型设计。并行算法模型需要根据应用的核心任务数据量和访问特征研究线程间无关并行、线程间流水并行、基于共享内存或片上通信的多线程协同并行、主从异步并行等方式，再基于具体的并行模式开展数据布局与传输方案设计。而应用程序级的优化则需要结合目标平台众核片上存储层次的具体特点，比如SPM的容量大小、共享范围、访问带宽和延时，再进一步结合程序核心循环的数据访问方式、数据量等开展数据布局优化、SPM数据重用优化、批量数据传输优化、基于SPM缓冲的数据传输与计算重叠优化等，从而使应用程序能够更好地发挥出存储层次的优势，提升应用效率。

5 结束语

本文阐述了3类常见的片上存储结构及特点，分析了国际主流GPU、同构众核及国产众核等面向主流E级超算系统的众核处理器片上存储层次设计现状与发展趋势，并总结了国际上的存储层次设计与优化相关软硬件技术的研究现状。后续将针对具体的应用领域，对应用获得的性能与片上存储结构类型及各种结构参数的定量关系方面开展研究，从而为众核片上存储层次的软硬件设计给出更具体的参考依据。

参考文献

[1] TOP500 LIST-NOVEMBER 2021[EB/OL]. [2022-11-10]. https://www.top500.org/lists/top500/list/2021/11/. 2021.

[2] BANAKAR R, STEINKE S, LEE B S, et al. Scratchpad memory: a design alternative for cache on-chip memory inembedded systems [C]//Proceedings of the 10th International Symposium on Hardware/Software Codesign. Washington D. C., USA: IEEE Press, 2002: 1587-1599.

[3] SATO M, KODAMA Y, TSUJI M, et al. Co-design for A64FX manycore processor and "Fugaku"[J]. IEEE Micro, 2022, 42(2): 26-34.

[4] WEN H, ZHANG W. Reducing cache leakage energy for hybrid SPM-cache architectures[C]//Proceedings of 2014 International Conference on Compilers, Architecture and Synthesis for Embedded Systems. New York, USA: ACM Press, 2014: 1-9.

[5] 刘霞. GX64-DSP片上标向量便签式存储器设计与实现[D]. 长沙: 国防科学技术大学, 2015.
LIU X. Design and implementation of GX64-DSP on-chip scalar vector scratch pad memory[D]. Changsha: National University of Defense Technology, 2015. (in Chinese)

[6] WOLPERT D, BERRY C, BELL B, et al. Cores, cache, content, and characterization: IBM's second generation 14-nm product, z15 [J]. IEEE Journal of Solid-State Circuits, 2021, 56(1): 98-111.

[7] IBM Corporation. IBM Telum processor [EB/OL]. [2022-11-10]. https://www.ibm.com/blog/ibm-telum-processor-the-next-gen-microprocessor-for-ibm-z-and-ibm-linuxone/.

[8] PHAM D, ASANO S, BOLLIGER M, et al. The design and implementation of a first-generation CELL processor[C]// Proceedings of IEEE International Solid-State Circuits Conference. Washington D. C., USA: IEEE Press, 2005: 125-136.

[9] Adapteva Inc. Epiphany technical reference documents [EB/OL]. [2022-11-10]. http://wwwadaptevacom/all-documents.

[10] 胡向东, 柯希明, 尹飞, 等. 高性能众核处理器申威26010[J]. 计算机研究与发展, 2021, 58(6): 1155-1165.
HU X D, KE X M, YIN F, et al. Shenwei-26010: a high-performance many-core processor[J]. Journal of Computer Research and Development, 2021, 58(6): 1155-1165. (in Chinese)

[11] 高珂, 陈荔城, 范东睿, 等. 多核系统共享内存资源分配和管理研究[J]. 计算机学报, 2015, 38(5): 1020-1034.
GAO K, CHEN L C, FAN D R, et al. Shared memory resources allocation and management research on multicore systems[J]. Chinese Journal of Computers, 2015, 38(5): 1020-1034. (in Chinese)

[12] LU K, WANG Y, GUO Y, et al. MT-3000: a heterogeneous multi-zone processor for HPC[EB/OL]. [2022-11-10]. https://doi-org-s.libyc.nudt.edu.cn:443/10.1007/s42514-022-00095-y.

[13] NVIDIA's next generation CUDA compute architecture: Fermi[EB/OL]. [2022-11-10]. http://wwwnvidiacom/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaperpdf, 2009.

[14] CHOQUETTE J, GANDHI W, GIROUX O, et al. NVIDIA A100 tensor core GPU: performance and innovation[J]. IEEE Micro, 2021, 41(2): 29-35.

[15] 高剑刚, 胡晋, 龚道永, 等. 神威·太湖之光可靠性及可用性设计与分析[J]. 计算机研究与发展, 2021, 58(12): 2696-2707.
GAO J G, HU J, GONG D Y, et al. Design and analysis of reliability and availability on Sunway TaihuLight[J]. Journal of Computer Research and Development, 2021, 58(12): 2696-2707. (in Chinese)

[16] LI F, LIU X, LIU Y, et al. SW_Qsim: a minimize-memory quantum simulator with high-performance on a new Sunway supercomputer[C]//Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis. New York, USA: ACM Press, 2021: 1-13.

[17] MANIKANTAN R, RAJAN K, GOVINDARAJAN R. Probabilistic shared cache management [C]//Proceedings of the 39th Annual International Symposium on Computer Architecture. Washington D. C., USA: IEEE Press, 2012: 428-439.

[18] QURESHI M, PATT Y. Utility-based cache partitioning: a low-overhead, high-performance, runtime mechanism to partition shared caches[C]//Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture. Washington D. C., USA: IEEE Press, 2006: 323-335.

[19] SANCHEZ D, KOZYRAKIS C. Scalable and efficient fine-grained cache partitioning with vantage[J]. IEEE Micro, 2012, 32(3): 26-37.

[20] XIE Y J, LOH G. PIPP: promotion/insertion pseudo-partitioning of multi-core shared caches[J]. Computer Architecture, 2009, 43: 556-567.

[21] CHEN Y, LI W L, KIM C, et al. Efficient shared cache management through sharing-aware replacement and streaming-aware insertion policy[C]//Proceedings of 2009 IEEE International Symposium on Parallel & Distributed Processing. Washington D. C., USA: IEEE Press, 2009: 157-166.

[22] NATARAJAN R, CHAUDHURI M. Characterizing multi-threaded applications for designing sharing-aware last-level cache replacement policies[C]//Proceedings of 2013 IEEE International Symposium on Workload Characterization. Washington D. C., USA: IEEE Press, 2013: 553-568.

[23] 刘士建. 基于CPU-GPU异构架构下Cache优化技术的研究[D]. 北京: 北京工业大学, 2018.
LIU S J. Research on Cache optimization technology based on CPU-GPU heterogeneous architecture[D]. Beijing: Beijing University of Technology, 2018. (in Chinese)

[24] 范清文. 异构多核下Cache替换算法的性能优化研究[D].

北京:北京工业大学,2017.
FAN Q W. Research on performance optimization of Cache replacement algorithm under heterogeneous multi-core[D]. Beijing: Beijing University of Technology, 2017. (in Chinese)
[25] 马虓. 片上异构多核处理器LLC替换策略研究[D]. 长沙:国防科学技术大学,2013.
MA X. Research on LLC replacement strategy of heterogeneous multi-core processor on chip[D]. Changsha: National University of Defense Technology, 2013. (in Chinese)
[26] 王子聪,陈小文,郭阳. 片上多核处理器Cache访问均衡性研究[J]. 计算机学报,2019,42(11):2403-2416.
WANG Z C, CHEN X W, GUO Y. Research on cache access equalization in chip multi-processor[J]. Chinese Journal of Computers, 2019, 42(11): 2403-2416. (in Chinese)
[27] WANG Z C, CHEN X W, LI C, et al. Fairness-oriented and location-aware NUCA for many-core SoC[C]//Proceedings of the 11th IEEE/ACM International Symposium on Networks-on-Chip. New York, USA: ACM Press, 2017: 1-8.
[28] ZHAO X, LIU Y X, ADILEH A, et al. LA-LLC: inter-core locality-aware last-level cache to exploit many-to-many traffic in GPGPUs [J]. IEEE Computer Architecture Letters, 2017, 16(1): 42-45.
[29] KUMAR A, DAS A, JOSE J. Reducing off-chip miss penalty by exploiting underutilised on-chip router buffers[C]//Proceedings of the 38th IEEE International Conference on Computer Design. Washington D. C., USA: IEEE Press, 2020: 532-546.
[30] ALVAREZ L, VILANOVA L, GONZALEZ M, et al. Hardware-software coherence protocol for the coexistence of caches and local memories[C]//Proceedings of 2012 International Conference for High Performance Computing, Networking, Storage and Analysis. New York, USA: ACM Press, 2012: 1-11.
[31] CARBALLEIRA F G, CARRETERO J, CALDERÓN A, et al. An adaptive cache coherence protocol specification for parallel input/output systems[J]. IEEE Transactions on Parallel & Distributed Systems, 2004, 15(6): 533-545.
[32] PEI S W, KIM M S, GAUDIOT J, et al. Fusion coherence: scalable cache coherence for heterogeneous kilo-core system[J]. Computer and Information Science, 2014, 58(12): 143-156.
[33] SÁNCHEZ D, KOZYRAKIS C. SCD: a scalable coherence directory with flexible sharer set encoding [J]. IEEE Computer Society, 2012, 44(5): 1-12.
[34] 何锡明,马胜,黄立波,等. 一种基于自更新的简单高效Cache一致性协议[J]. 计算机研究与发展,2019,56(4):719-729.
HE X M, MA S, HUANG L B, et al. A simple and efficient cache coherence protocol based on self-updating [J]. Journal of Computer Research and Development, 2019, 56(4): 719-729. (in Chinese)
[35] MEKKAT V, HOLEY A, YEW P C, et al. Building expressive, area-efficient coherence directories [C]// Proceedings of the 22nd International Conference on Parallel Architectures and Compilation Techniques. Washington D. C., USA: IEEE Press, 2013: 237-246.
[36] LIU P, HU Q, HUA X C. Adaptive coherence granularity for multi-socket systems [J]. IEEE Transactions on Computers, 2017, 66(8): 1302-1312.
[37] SHIM K S, CHO M H, LIS M, et al. Library cache coherence[C]//Proceedings of IEEE International Conference on Computer. Washington D. C., USA: IEEE Press, 2011: 457-465.
[38] LIS M, SHIM K S, CHO M H, et al. Memory coherence in the age of multicores[C]//Proceedings of the 29th IEEE International Conference on Computer Design. Washington D. C., USA: IEEE Press, 2011: 1-8.
[39] SINGH I, SHRIRAMAN A, FUNG W W L, et al. Cache coherence for GPU architectures[C]//Proceedings of the 19th IEEE International Symposium on High Performance Computer Architecture. Washington D. C., USA: IEEE Press, 2013: 332-345.
[40] AGARWAL N, NELLANS D, EBRAHIMI E, et al. Selective GPU caches to eliminate CPU-GPU HW cache coherence[C]//Proceedings of 2016 IEEE International Symposium on High Performance Computer Architecture. Washington D. C., USA: IEEE Press, 2016: 335-348.
[41] POWER J, BASU A, GU J L, et al. Heterogeneous system coherence for integrated CPU-GPU systems [C]// Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture. New York, USA: ACM Press, 2013: 457-467.
[42] BASU A, PUTHOOR S, CHE S, et al. Software assisted hardware cache coherence for heterogeneous processors[C]// Proceedings of the 2nd International Symposium on Memory Systems. New York, USA: ACM Press, 2016: 279-288.
[43] 刘勇,刘丽,何王全. 面向众核多级访存资源的静态数据布局优化模型[J]. 计算机应用与软件,2011,28(7):53-56.
LIU Y, LIU L, HE W Q. A static data placement optimisation model oriented towards multi-core hierarchical accessible resources [J]. Computer Applications and Software, 2011, 28(7): 53-56. (in Chinese)
[44] 胡志刚,石金锋,蒋湘涛. 针对能耗热点的SPM静态分配管理策略[J]. 计算机工程与应用,2010,46(3):58-61,75.
HU Z G, SHI J F, JIANG X T. Static allocation management strategy for SPM based on energy hotpot[J]. Computer Engineering and Applications, 2010, 46(3): 58-61, 75. (in Chinese)
[45] NGUYEN N, DOMINGUEZ A, BARUA R. Memory allocation for embedded systems with a compile-time-unknown scratch-pad size [J]. ACM Transactions on Embedded Computing Systems, 2021, 8(3): 221-235.
[46] 刘勇,陆林生,何王全. 一种简便的栈式片上内存动态管理方法[J]. 计算机工程与科学,2010,32(9):111-114.
LIU Y, LU L S, HE W Q. An easy stack-analogy on-chip memory dynamic allocation compilation technique[J]. Computer Engineering & Science, 2010, 32(9): 111-114. (in Chinese)
[47] 李嘉欣,邓宁. 一种基于访问计数的SPM管理策略[J]. 计算机工程,2013,39(9):109-113.
LI J X, DENG N. A scratch pad memory management strategy based on access counting [J]. Computer Engineering, 2013, 39(9): 109-113. (in Chinese)
[48] LIN J P, LU J, CAI J A, et al. Efficient heap data management on software managed manycore architectures [C]//

Proceedings of the 32nd International Conference on VLSI Design and 18th International Conference on Embedded Systems. Washington D. C., USA: IEEE Press, 2019: 112-125.

[49] UDAYAKUMARAN S, DOMINGUEZ A, BARUA R. Dynamic allocation for scratch-pad memory using compile-time decisions[J]. ACM Transactions on Embedded Computing Systems, 2006, 5(2): 472-511.

[50] BAI K, SHRIVASTAVA A, KUDCHADKER S. Stack data management for Limited Local Memory multi-core processors[C]//Proceedings of the 22nd IEEE International Conference on Application-specific Systems, Architectures and Processors. Washington D. C., USA: IEEE Press, 2011: 578-587.

[51] BAI K, SHRIVASTAVA A. Heap data management for limited local memory multi-core processors[C]// Proceedings of the 8th IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis. New York, USA: ACM Press, 2010: 317-326.

[52] BAI K, SHRIVASTAVA A. A software-only scheme for managing heap data on limited local memory multicore processors[J]. ACM Transactions on Embedded Computing Systems, 2013, 13(1): 332-345.

[53] BAI K, SHRIVASTAVA A. Automatic and efficient heap data management for limited local memory multicore architectures[C]//Proceedings of Design, Automation & Test in Europe Conference & Exhibition. Washington D. C., USA: IEEE Press, 2013: 593-598.

[54] LU J, BAI K, SHRIVASTAVA A. SSDM: smart stack data management for software managed multicores[C]// Proceedings of the 50th Annual Design Automation Conference. New York, USA: ACM Press, 2013: 1-8.

[55] VENKATARAMANI V, CHAN M C, MITRA T. Scratchpad-memory management for multi-threaded applications on many-core architectures[J]. ACM Transactions on Embedded Computing Systems, 2019, 18(1): 1-28.

[56] TAO X H, PANG J M, XU J L, et al. Compiler-directed scratchpad memory data transfer optimization for multithreaded applications on a heterogeneous many-core architecture[J]. The Journal of Supercomputing, 2021, 77(12): 14502-14524.

[57] CHAKRABORTY P, PANDA P R, SEN S. Partitioning and data mapping in reconfigurable cache and scratchpad memory: based architectures[J]. ACM Transactions on Design Automation of Electronic Systems, 2016, 22(1): 12-23.

[58] 李建江, 刘珍珍, 王珏. 基于IBM Cell多核平台的OpenMP数组私有化技术研究[J]. 计算机研究与发展, 2010, 47(8): 1434-1441.
LI J J, LIU Z Z, WANG J. Optimizing OpenMP by array privatization on the multi-core platform of IBM cell[J]. Journal of Computer Research and Development, 2010, 47(8): 1434-1441. (in Chinese)

[59] WU M C, LIU Y, CUI H M, et al. Bandwidth-aware loop tiling for DMA-supported scratchpad memory[C]// Proceedings of ACM International Conference on Parallel Architectures and Compilation Techniques. New York, USA: ACM Press, 2020: 97-109.

[60] YU C, BAI Y B, SUN Q X, et al. Improving thread-level parallelism in GPUs through expanding register file to scratchpad memory[J]. ACM Transactions on Architecture and Code Optimization, 2019, 15(4): 235-243.

[61] ZHOU B, HUANG Y Z, XU J C, et al. Memory latency optimizations for the elementary functions on the Sunway architecture[J]. The Journal of Supercomputing, 2019, 75(7): 3917-3944.

[62] 凌明. Cache/SPM共存架的动态存储布局优化技术研究[D]. 南京: 东南大学, 2011.
LING M. Research on dynamic storage layout optimization technology of Cache/SPM coexistence shelf[D]. Nanjing: Southeast University, 2011. (in Chinese)

[63] WU L, ZHANG W. Cache-aware SPM allocation algorithms for hybrid SPM-cache architectures[C]// Proceedings of the 16th International Symposium on Quality Electronic Design. Washington D. C., USA: IEEE Press, 2015: 432-441.

[64] 凌明, 张阳, 梅晨, 等. 一种面向能耗的可重构片上统一存储架构[J]. 东南大学学报(自然科学版), 2011, 41(6): 1137-1145.
LING M, ZHANG Y, MEI C, et al. Energy-oriented reconfigurable on-chip unified memory architecture[J]. Journal of Southeast University (Natural Science Edition), 2011, 41(6): 1137-1145. (in Chinese)

[65] KOMURAVELLI R, SINCLAIR M D, ALSOP J, et al. Stash: have your scratchpad and cache it too[C]// Proceedings of the 42nd Annual International Symposium on Computer Architecture. New York, USA: ACM Press, 2015: 707-719.

编辑 索书志