華中科技大學

数据中心技术课程综述报告

院　　系 计算机科学与计算学院

班　　级 硕 2207 班

学　　号 M202273800

姓　　名 张茂荣

2023 年 1 月 18 日

近内存计算体系结构研究进展与趋势

张茂荣 1)

1) 华中科技大学计算机科学与技术学院，武汉市 430074

摘 要 随着应用数据处理需求的激增，在传统冯·诺依曼(von Neumann)体系结构中，处理器到主存之间的总线数据传输逐渐成为瓶颈。不仅如此，近年来兴起的数据密集型应用，如神经网络和图计算等，呈现出较严重的数据局部性，缓存命中率低。在这些新兴数据密集型应用的处理过程中，中央处理器到主存间的数据传输量大，导致系统的性能不佳且能耗变高。针对传统冯·诺依曼体系结构的局限性，近内存计算体系结构通过赋予主存端一定的计算能力，以缓解因数据量大以及数据局部性差带来的总线拥堵和传输能耗高的问题。近内存计算的主要形式是以高带宽的连接方式将计算资源集成到主存单元中(近内存计算)，但它也有自己的优缺点和适用场景。本文首先介绍并分析了近内存计算的提出和兴起原因，然后从硬件和微体系结构方面介绍近内存计算技术，接着分析和总结了近内存计算所面临的挑战，最后介绍了近内存计算给目前流行的应用带来的机遇。

关键词 近内存计算；神经网络；图计算

Graph processing architecture research progress and trends

Maorong Zhang[1)]

1) Department of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074

Abstract With the explosive increase of processed data, data transmission through the bus between CPU and the main memory has become a bottleneck in the traditional von Neumann architecture. On top of this, popular data-intensive workloads, such as neural networks and graph computing applications, have poor data locality, which results in a substantial increase of the cache miss rate. Processing such popular data-intensive workloads hinders the entire system since the data transmission causes long latency and high energy consumption. Near-memory processing architecture greatly reduces this data transmission by equipping the main memory with computation ability, alleviating the problems of poor performance and high energy consumption caused by a large amount of data and a poor data locality. The main form of near-memory processing is to integrate computing resources into the main memory unit in the way of high bandwidth connection (near data computing), but it also has its own advantages and disadvantages and application scenarios. In this survey, the birth and development of near-memory processing is firstly introduced and discussed. Its techniques, ranging from hardware to micro-architecture, are then presented. Furthermore, the challenges faced by near-memory processing are analyzed. Finally, the opportunities that near-memory processing offers for popular applications are discussed.

Key words near-memory processing; neural network; graph computing

1 引言

近年来，应用数据呈现爆炸式增长，处理器和主存之间的带宽限制成为数据密集型应用的瓶颈。此外，目前流行的一些数据密集型应用，如神经网络应用和图计算应用，数据的局部性差。这会导致处理器片上缓存命中率降低，进而导致处理器和主存之间频繁地传输数据。这样的大量数据传输除了使得总线拥堵并影响性能，还会造成大量的能耗开

销。研究表明，两个浮点数在 CPU(central processing unit)和主存之间传输所需的能耗要比一次浮点数运算大两个数量级[1, 2]。在大数据系统中，能耗开销大会使得基于传统冯·诺依曼(von Neumann)结构的系统扩展性差，甚至无法支持大型的数据密集型应用。

近内存计算(near-memory processing，NMP)给总线上数据传输量过大的问题提供了一个根源性的解决方案。它通过赋予内存一部分计算能力，使其能直接处理一些形式单一且数据量大的计算(例如向量乘矩阵)。近内存计算的主要形式是以高带宽的连接方式将计算资源集成到主存单元中，它很大程度减少了中央处理器和主存之间的数据移动，从而达到提升系统性能并降低能耗的目的。数据表明，近年来提出的近内存计算架构相比于传统冯·诺依曼架构有着几十，上百，甚至上千倍的性能提升。尽管在性能和能耗方面优势明显，目前近内存计算的应用仍面临着诸多挑战。

然后，在近内存计算架构中，由于内存中用于数据处理的逻辑芯片计算能力较弱，架构设计者需要分析和提取出程序中适合放到内存中做计算的部分，其余留给中央处理器处理。其次，架构设计者还需要针对上层应用的特点，对近内存计算中的逻辑芯片进行精心设计以取得大幅性能提升。不仅如此，近内存计算还缺乏高效透明的系统级支持。虽然一些面向特定应用设计的近内存计算微体系结构提供了上层软件接口，但是程序员需要对底层充分了解才能够使用该近内存计算系统。也有研究者设计了对上层透明的近内存计算系统接口，但仅能适应特定的近内存计算结构，缺乏通用性。

由此可见，近内存计算技术虽然潜力大，但是其结构复杂，与上层应用关系紧密，应用到现有系统中仍面临着诸多挑战。因此，我们对现有的近内存计算研究进行综述，阐述近内存计算从硬件架构到软件系统支持的相关技术方法和研究进展。本文首先详细分析和介绍了近内存计算的兴起原因，然后介绍了近内存计算的形式以及目前已有的近内存计算微体系结构，接着分析和总结近内存计算所面临的挑战，最后介绍近内存计算给现在流行的应用带来的机遇。

2 近内存计算的提出与兴起

2.1 近内存计算的提出

20 世纪 70 年代，超级计算机中单指令多数据流的性能不佳。这是因为存在数据依赖的问题，导致单指令多数据流无法进行高效的数据并行。因此，研究者们提出近内存计算技术，希望在存储端加上处理单元，提前执行好部分计算以减少数据的依赖。

以 20 世纪提出的近内存计算微体系结构为例，该结构通过在存储阵列旁加了一些计算单元(例如 ALU)，用于支持存储阵列内部的数据处理。这种做法直观但并不实用。一方面，当时的技术能力还无法使计算单元和存储单元紧密地结合，所加的硬件资源占用了较大的存储片上面积，不能达到与其相匹配的计算资源利用率。另一方面，当时的应用所处理的数据量不大，不能使所加的计算单元满载。在当时的技术环境下，这种直接在存储内部加计算单元的方式不仅难以获得较高的资源利用率，而且不能达到占用面积小且能耗低的目的。故而，近内存计算在 20 世纪并没有兴起。

2.2 近内存计算的兴起

近内存计算真正兴起是在 2010 年后，数据呈现指数级暴增。该阶段数据驱动的应用发展迅猛，例如近年来流行的人工智能应用，需要大量的数据进行模型训练，推理时也需要处理大量的数据。以谷歌翻译为例，近年来处理数据量不断增长，目前一分钟需要翻译六千九百五十万个词。现代计算机中片上存储空间有限，对于大规模的数据处理需求，中央处理器将频繁地到主存取数据并把处理好的数据存回主存。数据显示，传输两个浮点数的能耗要比一次浮点运算大两个数量级[1, 2]。现在全球能源紧缺，降低能耗是设计现代计算机的一大要点。因此，研究者们重新考虑赋予内存一定的计算能力，从而减少数据移动，降低计算机系统运行能耗。

与此同时，新型存储器件迅猛发展，包括 3D 堆叠的存储器件，如 HMC(hybrid memory cube)/HBM(high bandwidth memory)和 3DXPoint，以及交叉栅栏式(crossbar)结构的非易失性存储器件，如 ReRAM 和 PCM。图 1 展示了一种 3D 堆叠的存储结构。其中，最底层为逻辑层，包含控制器和其他处理单元。逻辑层上面堆叠有 DRAM 存储层，他们通过高速地穿过硅片的通道(through silicon via，TSV)相连接。图 1 中的一个立方体(cube)包含了 16 个拱(vault)，每个拱的一层包含了两个阵列

(bank)。与传统的二维结构不同的是，该 3D 结构最底层的逻辑层可以放置计算单元，且可以通过超高速的 TSV 与上层的存储单元进行数据交互，从而使得计算和存储结合得更加紧密。

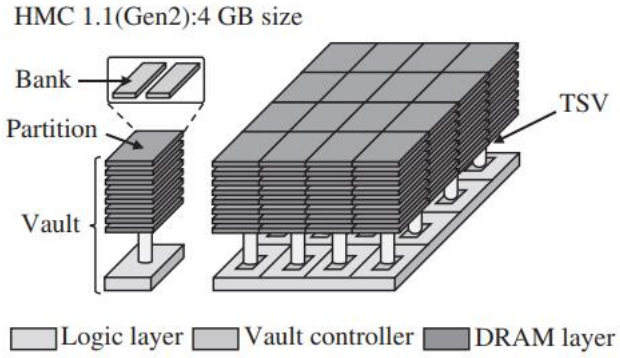

图 1 3D-stacked 存储结构

综上所述，数据驱动的应用迅猛发展以及数据量指数级暴增驱动了近内存计算的发展，并且新型存储器件的快速发展为近内存计算提供了技术保障。因此，近内存计算在 2010 年后兴起。

3 近内存计算架构与技术

近内存计算技术是一个宏观的概念，是将计算能力集成到内存中的技术统称。集成了近内存计算技术的计算机系统不仅能直接在内存中执行部分计算，还能支持传统以 CPU 为核心的应用程序的执行。区别于近内存计算，存算一体芯片将存储与计算相结合，是一种 ASIC(application-specific integrated circuit)芯片，常用于嵌入式设备中，针对一类特定的应用设计，不能处理其他应用程序。下面将从硬件结构和所支持的计算操作两个方面具体介绍近内存计算相关技术。

3.1 通用的近内存计算架构

通用的近内存计算架构方面代表性工作有：AMD Research 的 TOP-PIM[3]，Carnegie Mellon University 的 TOM[4]，University of Wisconsin-Madison 的 DRAMA[5]和 NDA[6]，Seoul National University 的 PEI[7]，IBM Research 的 AMC(active memory cube)[8]和基于多核 CPU 的近内存计算系统[9]，Stanford University 的 HRL[10]，Brown University 为近内存计算设计的并发数据结构[11]，Georgia Institute of Technology 的 AxRAM[12]，以及 Chinese Academy of Sciences 的 proPRAM[13]，具体如下。

TOP-PIM[3]提出了一种近内存计算的架构，存储单元不直接堆叠在中央处理器上，而是堆叠在内存处理器上，与中央处理器进行交互。TOP-PIM 在选择内存处理器时，考虑了能耗和热量的限制，充分分析大量应用在性能和能耗方面的特征。TOP-PIM 认为面向吞吐量的 GPU 核更适用于高带宽高数据并行的场景。实验表明，在 22nm 的工艺下，TOP-PIM 可以减少 76%的能耗，且仅带来 27%的性能损失。TOP-PIM 测试的不是某种特定类型的应用，无法对特定类型应用加速；另外，它将整个应用都放到近内存计算中执行，而近内存计算中的计算资源更适用于单指令多数据流的计算，难以支持复杂的多样的计算。因此，TOP-PIM 在性能上不如传统冯·诺依曼系统。

TOM[4]将代码分成多个块，通过编译器的静态分析，判断出适合放到近内存计算中执行的块代码，避免了把整个代码都放到近内存计算模块中执行。其大致结构与 TOP-PIM 相似，不同的是 TOM 支持各个 NDC cube 之间的通信。TOM 的提出是为了解决大数据时代 GPU 与主存之间带宽小的问题，除了通过编译器静态分析代码块并选择合适的代码块放到近内存计算中执行之外，TOM 还分析预测了哪些数据会被放到近内存计算中的代码块访问，并将这些数据放在相应代码块执行的 NDC cube 中，以此来减少各个 NDC cube 之间的通信。TOM 中的代码分析和数据映射都对上层透明，程序员可非常方便地使用近内存计算。实验显示，TOM 平均能提高 GPU 的主流应用 30%的性能。

DRAMA[5]和 NDA[6]与其他近内存计算架构不同，它们建立在商用的 DRAM 设备上，力求对现有商用 3D 堆叠的 DRAM 硬件(HMC 和 HBM)改动最小，并且对现在的存储系统架构改动最小。DRAMA 和 NDA 的结构，二者整体结构还是中央处理器通过总线和 DRAMDIMM 相连。不同的是，其中一部分 DRAMDIMM 使用了堆叠在 DRAM 上面的 CGRA(coarse-grained reconfigurable array)加速器，能够将数据密集型的操作放到此类近数据结构中进行处理。实验显示，这样的近数据处理结构能够带来 3 倍到 60 倍的性能提升，减少了 63%~ 96%的系统能耗。

PEI[7]提出了一套近内存计算和现有系统结合的软硬件接口。PEI 将近内存计算指令计算单元(PCU)放在每个主处理器核以及每个近内存计算核上，使得指令可在主处理器端或近内存计算端执

行；同时还加入了近内存计算管理单元(PMU)，与LLC(last level cache)以及主存控制器相互合作。它的主旨是使用能够计算的存储指令和特殊指令实现简单的近内存计算指令，供上层应用所用。在现有的顺序编程模型基础上，PEI 加入硬件单元来监测数据的局部性，根据数据的局部性自动决定哪些操作在近内存计算单元里执行。因此，近内存计算系统能和现有的编程模型、缓存冲突处理机制，以及虚拟内存管理很好地协作。PEI 能和现有的系统结构很好地协作。

AMC[8]是一个面向百亿兆级超级计算机(一秒钟执行 1018 运算)设计的近内存计算系统。AMC 在设计 NDC cube 的逻辑层时，全方面考虑了常用科学应用的计算需求和百亿兆级计算机系统的低功耗需求，力求最大化有限面积的利用率和能效比。同时，AMC 通过软硬件接口将硬件微结构暴露给上层软件，允许软件针对性能需求调整应用参数，支持软件根据操作系统特点配置和调度 AMC 中的资源。AMC 使用 OpenMP 4.0[29]作为近内存计算中节点粒度的编程接口，相应编译器编译好程序供 AMC 执行。实验显示，AMC 的内部带宽比传统中央处理器到主存的带宽大一个数量级；此外，AMC 比传统处理器计算效率高一个数量级，能以很高并行度执行负载的重要部分，其能耗比传统系统一半还少。AMC 验证了近内存计算比将存储移动到中央处理器端的模式效率更高。

Vermij 等[9]提出了一个基于多核 CPU 的近内存计算系统，以支持近内存计算和现有系统的结合。他们重新考虑了近内存计算中的关键问题，如数据冲突、数据排布、通信、地址映射和编程模型等，并实现了一个基于软件和硬件的模拟器。实验表明，他们所提出的系统在 Graph500 测试中与 CPU 系统相比，有 1.5 倍的性能提升。

HRL[10]针对 NDC cube 逻辑层设计时，片上功耗和面积限制与高带宽和充足计算量的矛盾，提出了可重构的逻辑阵列，能够灵活适用于大量的数据密集型应用。HRL 发现，ASIC 的设计性能好但缺少灵活性，无法支持大量应用。FPGA 和 CGRA 通过可编程性提供了充足的灵活度。但是 FPGA 是以比特为粒度的计算单元和互联单元可编程的硬件，片上面积开销大； CGRA 能支持复杂数据流的强大互联，能耗比 FPGA 更高，在特殊计算和不规则数据上灵活度低，性能不佳。因此，HRL 提出了混合可重构的逻辑阵列，用作 NDC cube 的逻辑层，包含了 FPGA 块和 CGRA 块。HRL 阵列结构，其中，FU(function unit)是用类 CGRA 块实现的，保证面积和能效比，能够支持数学运算和逻辑操作，包括加法、减法、乘法，和比较操作；CLB(configurable logic block)是用类 FPGA 块实现的，用来实现一些不规则的控制逻辑和特殊函数(比如神经网络中的激活函数)；OMB(output multiplexer block)是由多个多工器组成的，放置在靠近输出的位置，用来支持计算分支，例如树形、瀑布型，和并行型计算，是一种能同时节省片上面积和能耗开销的分支实现方式。另外，HRL 没有配置类似 FPGA 的 BRAM 缓存，因为近内存计算应用通常数据局部性差，大缓存是额外负担，不能提高系统的效率。如上所述，HRL 综合了 FPGA 能耗低和 CGRA 面积利用率高的优点，比基于 FPGA 的 NDC 系统性能高 2.2 倍，比基于 CGRA 的 NDC 系统性能高 1.7 倍。

Liu 等[11]提出了适应于近内存计算的并发数据结构。他们发现，现在的服务器集群中通常有几百个核，并发数据结构在吞吐和扩展性方面优于传统的顺序数据结构。因此，近内存计算系统如何支持并发数据结构，以及并发数据结构如何利用近内存计算的优势，成为研究重点。Liu 等发现，并发数据结构性能优于普通的近内存计算的数据结构；另外，利用如数据融合、数据分块、流水线等技术面向近内存计算设计的并发数据结构，性能优于传统 CPU 中的并发数据结构。他们设计的跳表结构实例，以及该结构在近内存计算系统中的映射。在这个例子中，跳表被分成 3 部分存储到 3 个拱中，每个部分都以哨兵点开头，且哨兵点在 CPU 端有一份拷贝。除此之外，他们还实现了针对近内存计算设计的链表，先进先出队列和管道，在性能上优于现有的面向传统 CPU 系统的相关数据结构。

Yazdanbakhsh 等[12]认为将加速器集成到 NDC cube 中极具挑战，需要加速器低能耗，占用面积小，还要支撑多样化的应用。他们提出 AxRAM，利用 GPU 应用的可近似性，将不同区域代码中的计算转化成乘加(multiply-accumulate operation，MAC)等近似算子。AxRAM 用单一的算子近似应用中的操作，从而使 NDC cube 中逻辑层设计简单、能耗低、占面积小。为了保持 GPU 的单指令多数据流的执行特性，近内存计算端用多个 MAC 近似单元绕 GPU 翻译并执行命令，这样做不需要改变 DRAM 的内部结构且不产生额外的内存开销。实验结果显示，与传统 GPU 系统相比，AxRAM 平均取得了 2.6 倍

的性能提升以及 13.3 倍的能耗节约，而只带来了 2.1%的面积开销。

proPRAM[13]是少数不基于 3D 堆叠结构的近内存计算架构。它充分利用了 NVM 中的基础逻辑单元实现近内存计算，例如数据比较写(data-comparison write，DCW)单元，翻转写(Flip-n-Write)单元等对 SET/RESET 操作非常关键的单元。proPRAM 对硬件结构改动微小，且改动对上层应用不可见，能很好地支持数据密集型操作，与 CPU 系统相比，在数据密集型应用上能取得 15 倍的能耗节约。

3.2 针对机器学习的近内存计算架构

针对机器学习的近内存计算架构代表性工作有：Georgia Institute of Technology 的 BSSync(bounded staled sync)[14]和 Neurocube[15]，Advanced Micro Devices 的 Co-ML[16]，具体如下。

BSSync[14]指出，在并行实现的机器学习应用中，原子操作用来保障无锁状态下算法的收敛，但带来很大的同步开销，且同步产生的通信延迟不与占比大的计算延迟重叠。BSSync 发现，在机器学习应用迭代收敛过程中，可以用未更新的中间数据进行计算，从而提出利用基于近内存计算的有边界一致性模型减少原子操作带来的延迟开销。BSSync 系统结构中，CPU 核里面增加了原子请求队列、控制寄存器以及区域表来实现边界一致性模型。实验显示，BSSync 比机器学习应用在传统冯·诺依曼系统中的异步并行的实现快 1.33 倍。

Neurocube[15]是一个针对神经网络计算设计的可编程、可扩展，且节能的近内存计算系统架构。Neurocube 架构的左边是普遍使用的 NDC cube 结构，右边是逻辑层设计。逻辑层采用了细粒度可编程的设计模型，以灵活支持神经网络计算。其中，每个 PE 有多个 MAC 单元支持神经网络中最常用的乘加操作，同时还有存储权值的寄存器和缓存以及相应的计数器。它首先将神经网络存储到 NDC cube 的存储单元中，包括每层数据、神经元状态、连接权值。当一个层处理好之后，与中央处理器交互一次，然后执行下一层。Neurocube 通过对逻辑层硬件、数据映射方式、片上互联，以及编程方式的精心设计，使得神经网络计算在 NDC cube 中能够高效执行。实验显示，相比于 GPU 系统，Neurocube 有 4 倍的每瓦计算效率提升，与 ASIC 系统相比，灵活性更好、扩展能力更强。

不同于针对机器学习设计的注重优化乘加(MAC)操作的近内存计算系统，Co-ML[16]提出，虽然包含 MAC 操作的卷积层等计算占整个机器学习过程的比例大，但这些计算是计算密集型的，数据复用性好，计算/字节比率高(即一个字节从内存中读出来之后用来计算的次数多)；事实上，机器学习过程中，约 32%的时间用于数据密集型计算，这些计算的计算/字节比率低。Co-ML 将这些低计算/字节比率的计算部分放在近内存计算端，把 MAC 等操作放在主处理器上做。实验显示，Co-ML 在机器学习的数据密集型计算上的加速达到了 20 倍，总体有 14%的性能提升。

3.3 针对图计算的近内存计算架构

针对图计算的近内存计算架构的代表性工作有：Seoul National University 的 Tesseract[17]和 Georgia Institute of Technology 的 GraphPIM[18]，具体如下。

Tesseract[17]是一个针对图计算的可编程的近内存计算系统架构，它综合了图计算的特点，重新考虑了逻辑单元和存储单元的集成方式。Tesseract 的系统结构中，左边是一个图计算在互联的 NDC cube 中执行的实例，中间是一个 NDC cube 内部的连接结构，右边是一个拱内部的逻辑层结构。Tesseract 逻辑层使用了顺序执行的计算核。Tesseract 还使用了可以隐藏远程访问延迟的消息传递机制，以及为图计算定制的预取硬件，和一系列支持这些操作的上层接口。实验显示，与传统的系统相比，Tesseract 能取得 87%的能耗节约。

GraphPIM[18]是一个针对图计算的，基于商用 HMC2.0 的近内存计算完整解决方案。GraphPIM 主要解决利用近内存计算运行图计算的两大挑战：(1) 应该把图计算应用的哪些部分放到 NDC cube 中执行；(2) 如何把部分计算放到 NDC cube 中执行，即如何提供中央处理器和近内存计算处理器的接口。GraphPIM 发现，在图计算中，原子操作是损害性能的主要原因，因此应该把原子操作放到内存端，避免不规则的数据访问带来的大通信开销。GraphPIM 使用现有的中央处理器指令，把中央处理器端的原子操作指令映射到近数据处理端，并使用无缓存的配置。在 GraphPIM 架构下，上层应用程序员无需额外的工作，现有的指令集结构也不需要改动，就可以在图计算的负载中使用近内存计算。GraphPIM 的系统结构中，GraphPIM 在图数据

管理和硬件支持方面做了改动。在虚拟地址空间中，GraphPIM 使用传统系统的指令绕过缓存来分配图数据；在硬件端，GraphPIM 在中央处理器上加了一个 POU(PIM offloading unit)用来决定哪些操作放到 NDC cube 中执行。此外，GraphPIM 充分分析了图计算应用特征，判断出哪些部分放到近数据处理端做会有性能提升。实验显示，GraphPIM 与传统用 HMC2.0 当做主存的冯·诺依曼系统结构相比，有 2.4 倍的性能提升和 37%的能耗节约。

3.4 针对垃圾回收的近内存计算架构

Sungkyunkwan University 的 Charon[19]是一个针对垃圾回收提出的近内存计算系统。Charon 发现了在大数据系统中垃圾回收的开销尤为严重：在存储密集型的应用里，垃圾回收几乎占了一半的执行时间。通过分析垃圾回收算法，Charon 发现，主要占 MinorGC 执行时间的操作是 Search，Copy 和 Scan&Push；主要占 MajorGC 执行时间的操作是 Scan&Push，Bitmap Count 和 Copy。不仅如此，这些操作需要通过多线程并行提高吞吐量，而且这些操作的空间局部性和时间局部性很差。传统冯·诺依曼系统结构在执行这些操作时效率低下，而近内存计算则非常适合。因此，Charon 把这些操作放到近内存计算端处理。Charon 的系统结构中，其中加了 Copy/Search，Bitmap Count 和 Scan&Push 三个处理单元，用来处理垃圾回收中对应的操作。这些单元直接与主存交互，避免了主处理器通过串行的低带宽读取数据，从而减少了数据传输的时间和能耗开销。实验显示，Charon 能取得 3.3 倍的性能提升和 60.7%的能耗节约。

3.5 近内存计算架构与技术小节

近内存计算中逻辑层的设计较为灵活，可以针对不同系统的需求设计通用的处理器或者专用的加速器。在设计针对通用应用的近内存计算系统时，由于放到内存端的通用处理器一般性能较弱，需要考虑自动化地分割应用程序的计算部分，把能从近内存计算中获益的部分放到内存中处理。在设计针对特定类型应用的近内存计算系统时，需要仔细分析应用特点，抽取算子，设计对应的数据流。除了逻辑层的设计，近内存计算系统结构设计还需要考虑：各个内存块之间的连接方式，包括通信方式和数据一致性协议、数据映射策略、与现有系统集成方式、软硬件接口设计。

4 近内存计算面临的挑战

近内存计算设计时需要考虑计算机中多个层级。近内存计算本身存在一些问题，与现有的系统相结合也会带来新的问题，每个具体问题都涉及某些层级。解决这些问题是近内存计算可以在计算机系统中发挥作用，提升系统性能和降低能耗的关键。本节将分析并总结近内存计算和存内计算本身及其与现有系统集成会带来的挑战。

4.1 高能耗问题

在近内存计算系统中，对程序模块的分割不当或者过度使用近内存计算来处理数据反而会增加系统能耗。该问题的产生通常与编译器的设计和内存控制器的设计相关。Kim 等[20]分析了集成有传统处理器和 NDC cube 的系统的能耗特征，发现 LLCMPKI(misses per kilo instructions)是一个决定应用是否能在近数据系统中高效执行的关键特征：通常高 MPKI 表示可从近内存计算系统中获得高的能耗节约。因此，他们认为应将高 MPKI 的计算放到 NDC cube 中做，把其余部分留在传统系统中做。

4.2 硬件配置问题

因为近内存计算的 NDC cube 中逻辑芯片通常使用 FPGA，CGRA，和 ASIC 处理器等非通用处理器，所以硬件配置与应用需求的匹配成为一个挑战。该问题通常与电路设计和器件选择强相关。Gao 等[10]提出了一种异构且可重构的逻辑阵列，同时利用 FGPA 和 CGRA。通过分析适合近内存计算处理的大量应用，结合两者长处，设计 FPGA 和 CGRA 以及逻辑层的多路选择器的个数，以满足近内存计算应用的高性能低能耗的需求。针对特定应用场景或者特定应用，如何配置近内存计算中的逻辑芯片部件仍有待探索和研究。

4.3 计算资源利用率问题

近内存计算的计算资源基本均匀分布在整个内存中，直接使用传统的数据存储方式会导致计算资源分配不均匀，计算核与其所需的数据相距远等问题，进而导致近内存计算的计算资源利用率低。该问题的产生通常与编译器设计和内存控制器设计相关。内存中数据的存储需要考虑到计算资源的利用率，尽量使所有计算资源能够时分复用，管道串行，要避免因计算资源集中在某一块使用而导致

长时间的数据等待。

对于特定的存内计算架构，通常采取设计合适的数据映射策略来提高计算资源的利用率的思路。而数据映射策略取决于不同的近内存计算系统架构和运行在其上的应用[7]。因此，在设计上层系统时，需要同时考虑两者来设计合适的数据和计算核的排布。

4.4 缺少灵活的系统支持

运行在近内存计算系统中的应用需要在中央处理器和近内存计算端来回切换，因此，相应的系统应支持这样的灵活切换。目前的近内存计算都是针对特定的硬件结构或特定应用提供的上层软件支持[7]，灵活性不够。

近内存计算运用到系统中时，还需要考虑近内存计算中的多任务调度以及近内存计算任务和传统访存任务的调度问题。首先，由于近内存计算的计算资源均匀分布在内存中，一个好的多任务调度策略能使计算资源利用率高，同时减少系统中的冲突和中断。其次，在不处理近内存计算任务时，近内存计算模块也可以用作普通存储模块，因此，需要一个近内存计算任务和传统任务的调度策略，来优化近内存计算和传统存储混合的系统结构性能。另外，近内存计算中有多个计算模块，各计算模块之间可能会共享数据，包括并发地对数据进行增删改查的操作。因此，近内存计算需要一个解决冲突问题的方案，以保证其数据的一致性。

以上问题都属于近内存计算的系统问题，通常与系统级的设计强相关，也与编译器和内存控制器相关。

5 近内存计算所提供的机遇

近内存计算技术在处理存储密集型、数据局部性差、且计算形式较为单一、易于并行的应用上优势明显。目前流行的 3 类应用：机器学习，图计算和基因工程，在传统系统中因总线带宽有限和片上片下数据移动太频繁而性能低且能耗高。近内存计算技术恰好弥补了传统计算机体系结构在这 3 种流行应用上的缺陷。本节主要介绍近内存计算给这 3 种应用带来的机遇。

5.1 机器学习

机器学习的核心是神经网络。神经网络类型很多，包括：卷积神经网络，主要用于图像处理；循环神经网络，主要用于自然语言处理；深度神经网络是卷积网络或者循环神经网络的强化版，有更多的网络层；生成对抗网络是无监督学习中的最受关注的网络类型；还有深度强化学习，是一种在线学习方式，基于环境的改变而调整自身的行为。这些网络各具特点，大的网络会对存储造成很大的压力，连接多的网络会给计算和传输造成很大的压力。他们的共同点在于都有很多向量乘矩阵操作，占到所有计算的 90%以上。因此基于 NVM 的存内计算适合用于支持神经网络计算。

5.2 图计算

随着互联网数据爆炸式的增长以及人们对数据单元之间关系的关注，图计算在新型应用中占有较大地位。图计算的应用包括：网络安全、社交媒体、网页评分和引用排序、自然语言处理、系统生物学、推荐系统等。由于图计算应用的数据局部性差且数据量很大，目前用来处理图计算的系统面临性能不佳且能耗过高的难题。

图计算中大量操作都可以转换成矩阵乘的形式，因此可以用基于 NVM 的存内计算来处理。综合考虑图数据的预处理，稀疏矩阵的分隔和映射，以及硬件控制和数据流设计，能够高效且低能耗地支持图计算应用。图计算也可用基于 HMC 的 NDC 来处理，通过分析现在图计算中哪些操作适合放到 HMC 中处理，然后有针对性地设计逻辑层和上层的指令集，能取得较高的性能提升和能耗节约。所以近内存计算给图计算的发展提供了很好机遇。

5.3 基因工程

近年来，随着基因工程的发展，生物学信息数据呈指数级增长。这种生物数据的暴增给现在诸如基因序列匹配的应用带来了很大的挑战。目前有一些针对基因序列查找的软件方面的加速方法，也有利用基因处理的高并行性的硬件加速方法。但是由于生物数据的量太过庞大，传统计算机系统片上片下数移动量太大，系统能耗是一个很大的问题。基于 RCAM 的存内计算能利用 CAM 的极速查找能力，提供很高的硬件并行度，同时在存内处理能降低数据移动的能耗，适用于大规模生物数据处理。因此，存内计算给基因工程的发展提供了机遇。

6 总结

在数据爆炸时代，近内存计算技术为解决传统冯·诺依曼架构中总线拥堵问题以及片上片下数据传输能耗过高问题提供了解决方案。近内存计算技术得益于新型 3D 堆叠技术和非易失存储技术的发展。近内存计算的主要形式是以高带宽的连接方式将计算资源集成到主存单元中，它很大程度减少了中央处理器和主存之间的数据移动，从而达到提升系统性能并降低能耗的目的。近内存计算技术通常依赖于 3D 堆叠的内存结构，存算依然分离，但计算部分到存储部分带宽较大，计算部分灵活性较高。近内存计算技术总体还处于发展初期，有一些硬件和软件支持上的问题。但是近内存计算的潜力很大，能够给目前流行的机器学习应用，图计算应用，和基因工程提供高效低能耗的系统结构支持。

参 考 文 献

[1] Song L, Qian X, Li H, et al Pipelayer: a pipelined reram-based accelerator for deep learning. In: Proceedings of 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). Austin: IEEE, 2017. 541–552

[2] Farmahini-Farahani A, Ahn J H, Morrow K, et al NDA: near-DRAM acceleration architecture leveraging commodity DRAM devices and standard memory modules. In: Proceedings of 2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA). Burlingame: IEEE, 2015. 283–295

[3] Zhang D, Jayasena N, Lyashevsky A, et al TOP-PIM: throughput-oriented programmable processing in memory. In: Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing. Vancouver: ACM, 2014. 85–98

[4] Hsieh K, Ebrahimi E, Kim G, et al Transparent offloading and mapping (TOM): enabling programmer-transparent near-data processing in GPU systems. In: Proceedings of ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA), 2016. 44: 204–216

[5] Farmahini-Farahani A, Ho Ahn J, Morrow K, et al DRAMA: an architecture for accelerated processing near memory. IEEE Comput Arch Lett, 2015, 14: 26–29

[6] Mutlu O, Ghose S, G´ omez-Luna J, et al Processing data where it makes sense: enabling in-memory computation. Microprocessors Microsyst, 2019, 67: 28–41

[7] Ahn J, Yoo S, Mutlu O, et al PIM-enabled instructions: a low-overhead, locality-aware processing-in-memory architecture. In: Proceedings of 2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA). Portland: IEEE, 2015. 336–348

[8] Nair R, Antao S F, Bertolli C, et al Active memory cube: a processing-in-memory architecture for exascale systems. IBM J Res Dev, 2015, 59: 1–14

[9] Vermij E, Hagleitner C, Fiorin L, et al An architecture for near-data processing systems. In: Proceedings of the ACM International Conference on Computing Frontiers. Como: ACM, 2016. 357–360

[10] Gao M, Kozyrakis C. HRL: efficient and flexible reconfigurable logic for near-data processing. In: Proceedings of 2016 IEEE International Symposium on High Performance Computer Architecture (HPCA). Barcelona: IEEE, 2016.126-137

[11] Liu Z, Calciu I, Herlihy M, et al Concurrent data structures for near-memory computing. In: Proceedings of the 29th ACM Symposium on Parallelism in Algorithms and Architectures. Washington: ACM, 2017. 235–245

[12] Yazdanbakhsh A, Song C, Sacks J, et al In-DRAM near-data approximate acceleration for GPUs. In: Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques. Limassol Cyprus: ACM, 2018. 34

[13] Wang Y, Han Y, Zhang L, et al ProPRAM: exploiting the transparent logic resources in non-volatile memory for near data computing. In: Proceedings of the 52nd Annual Design Automation Conference. San Francisco: ACM, 2015. 47

[14] Lee J H, Sim J, Kim H. SSync: processing near memory for machine learning workloads with bounded staleness consistency models. In: Proceedings of 2015 International Conference on Parallel Architecture and Compilation (PACT), San Francisco: IEEE, 2015. 241–252

[15] Kim D, Kung J, Chai S, et al Neurocube: a programmable digital neuromorphic architecture with high-density 3D memory. In: Proceedings of 2016 ACM/IEEE 43rd Annual International Symposium on Computer Architecture (ISCA). Seoul: IEEE, 2016. 380–392

[16] Aga S, Jayasena N, Ignatowski M. Co-ML: a case for collaborative ML acceleration using near-data processing. In: Proceedings of the International Symposium on Memory Systems. Alexandria: ACM, 2019. 506–517

[17] Ahn J, Hong S, Yoo S, et al A scalable processing-in-memory accelerator for parallel graph processing. SIGARCH Comput Archit News, 2016, 43: 105–117

[18] Nai L, Hadidi R, Sim J, et al GraphPIM: enabling instruction-level PIM offloading in graph computing frameworks. In: Proceedings of 2017 IEEE International Symposium on High Performance Computer Architecture (HPCA). Austin: IEEE, 2017. 457–468

[19] Jang J, Heo J, Lee Y, et al Charon: specialized near-memory processing architecture for clearing dead objects in memory. In: Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture. Columbus: ACM, 2019. 726–739

[20] Kim H, Kim H, Yalamanchili S, et al Understanding energy aspects of processing-near-memory for HPC workloads. In: Proceedings of the 2015 International Symposium on Memory Systems. Washington: ACM, 2015. 276–282