**标题**

CMS: Computational CXL-Memory Solution for Memory-intensive Applications

CMS：用于加速**内存密集型应用程序**的**计算性的CXL-Memory**解决方案

**ABSTRACT**

CXL interface是最新的技术，通过在配置**异构设备**时提供一个**内存共享协议，**来有效地进行**内存扩展。**

但是，有限的**物理带宽**可能称为**新兴数据密集型应用程序**的重要瓶颈

在这项工作中，我们提出了一种新的基于CXL的**内存分解架构**，并进行了现实世界的原型演示，该架构利用near-data克服了CXL interface的带宽限制。

实验结果表明，与现有的CPU系统相比，我们的设计有更好的**性能功耗比**，比现有的系统更好，达到了1.9倍。

Compute express link (CXL), near-data-processing (NDP)

1. **INTRODUCTION**

大数据和人工智能兴起，为了满足内存需求，单纯扩大服务器和存储设备数量不是一个可扩展的解决方案。

内存分解是一种有前途的架构解决方案，它将**内存**从**计算节点**解耦。当遇到**用户应用程序**的**内存需求**时，它允许系统设计者扩展**独立于服务器的**额外的内存容量 。例如，在这个分解的服务器节点群中，内存压力大的服务器可以使用远离其他节点的节点的内存，。因此，这种方法可以比传统的专用CPU/内存体系结构更有效地管理资源。

最近在CXL上的努力是加速体系结构向内存分解的key推动者。CXL是一个**industry-supported(业界支持)**的**cache-coherent(缓存-相关)互联(interconnect)(CCI)**，使用一个**memory-semantic(内存-语义)协议**为各种各样的处理器来高效地扩张内存容量。与现有的**DDR interface**完全依赖于主机CPU不同，连接CXL的内存允许在**握手通信**在**存入内存**时**包含附加值。**

然而，在部署CXL内存时存在一个关键的技术挑战——与主机DDR的内存相比，CXL interfaced的带宽有限。

比较DDR和CXL内存的**带宽容量比，**,然而CXL内存的带宽非常有限，只有。在很多应用密集型的应用程序中，例如KNN(k-nearest neighbor search)，也就是我们的**系统**的**目标应用程序**中，贷款约束可能是限制系统性能的关键因素。

在本文中，提出了“CMS: Computational CXL-Memory Solution for Memory-intensive Applications”。CMS提供了一个快速的、可向外扩展和低成本的附加解决方案，提高了内存密集型应用程序的性能和电源效率。我们的CMS利用CXL interconnect(互连)来扩展需要大量数据的现代应用程序的内存容量。为了解决带宽挑战，我们采用了一种新颖的NDP(近数据处理)逻辑，最大限度地提高了内部带宽。我们的贡献总结如下:

1. 我们设计了一个CXL-based内存扩展原型，已经实际部署。它支持512GB的load/store空间
2. 我们在CXL内存中设计了一个NDP内核，通过减少**经过**CXL接口的**数据移动**来消除性能瓶颈。
3. 我们提出了一种load balancer(负载均衡器)，通过有效地交叉**内存通道**和**一个性能优化的MAC操作符that隐藏了累加器延迟**，来充分利用CMS的内部带宽
4. **2、RELATED WORK**

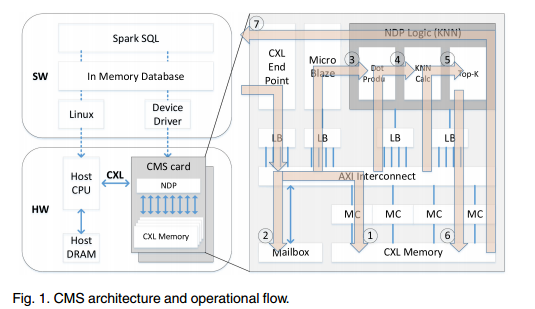
先前关于分解内存的一些研究，将capacity当做memory pool,允许它被多个机器分享。然而他们的工作依赖于RDMA over**无限带宽或以太网**，具有**有限的延迟**和**与主机相关的高开销**。相比之下，我们的工作利用新兴的CXL协议进行内存分解解决方案，在降低软件堆栈复杂性的同时实现高性能的内存共享(或池化)。

文献xxx、xxx提出了一种使用CXL存储设备的内存扩展系统IMDBMS (in -memory database management system)。作者小组成功展示了一个原型of CXL type 3 内存扩展设备 in E3.S 形式因子 using CXL.mem和CXL.io命令。然而，附加的cxl内存由于其物理限制，受到带宽的影响，已经在第一部分讨论过。我们的设计通过采用NDP逻辑来充分利用分解内存的内部带宽，从而克服了这一限制。

还有许多关于NDP技术的建议来解决处理器和内存之间的数据移动问题。例如，[17]、[18]中的工作分别卸载了内存数据库系统的**扫描操作**和推荐系统的**嵌入操作**，这两种操作都受到内存带宽的限制。它们将处理单元放入DIMM模块的缓冲区中，通过实现DIMM内部等级之间的并行性，增加了与DIMM接口相比的内部带宽。然而，RCD (Register Clock Driver)(寄存器时钟驱动程序)，which acts as a buffer for the RDIMM(Registered DIMM)，位于秩共享总线上，不能独立地在秩之间执行精确的并行操作[19]。相比之下，我们的工作支持可扩展的内部带宽，因为它可以根据内存节点中模块/ dimm的数量成比例地增加带宽。

1. **CMS ARCHITECTURE**

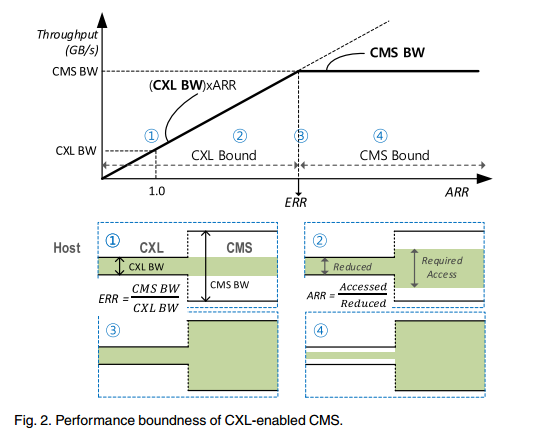
**3.1 Overall Architecture**



上图显示了整个系统，包括我们的CMS原型。它由主机CPU、DDR内存以及通过CXL接口连接的CMS原型组成。CMS原型包含一个CXL控制器、内部DDR内存和一个NDP引擎，该引擎允许CXL内存充分利用带宽，尽管CXL的带宽有限。

图1是我们CMS卡的详细框图。我们使用Xilinx Alveo U250对CMS卡进行了原型制作。我们将RTL和Xilinx(用于命令解析的软处理器)的MicroBlaze实现的卸载逻辑与CXL 2.0 IP进行了综合。Xilinx邮箱IP用于处理器间双向通信。CMS的操作流程如下。请注意，我们将KNN实现为初始示例，但是卸载逻辑可以灵活地扩展到其他应用程序。KNN操作的数据集最初已加载到CXL内存中。发出KNN搜索请求后，它将查询发送到CXL内存\_x0005\_①，其地址发送到邮箱②。然后，Micro Blaze解析查询\_x0005\_③，并在读取查询和数据④⑤的同时执行KNN操作。结果被写入内存中的预定地址⑥，这样主机就可以把它读回来\_x0005\_⑦

**3.2 Throughput Optimization Components吞吐量优化组件**

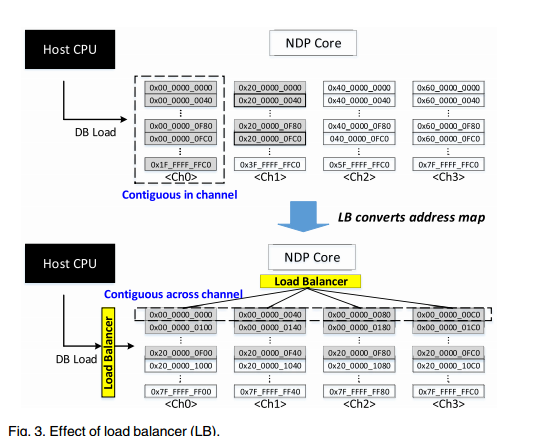


对于许多内存密集型工作负载，内存带宽是决定系统性能的关键因素[20]。如图2所示，在目标应用程序执行期间，限制性能的因素可能会随着数据缩减比(RR)的大小而变化。在本文中，我们以两种不同的方式定义了RR:application RR (ARR)和effective RR (ERR)。ARR是由应用程序算法决定的软件参数，定义为访问的数据大小与减少的数据大小的比率。ERR是硬件可以支持的缩减能力，定义为CMS BW(带宽)与CXL BW的比值。在ARR < ERR(图中①和②)区域，有效吞吐量可以表示为CXL BW \* ARR，表示性能受限于CXL带宽(CXL- bound)。而在ARR > ERR的区域(图2中④)， CMS处理的数据量大，经过CXL的数据量小。因此，性能受到CMS带宽(CMS- bound)的限制。对于许多内存密集型工作负载的情况，很明显我们的系统在cms绑定区域中运行，因为ARR大于ERR(例如，ERR 的 4.75 对比 ARR 的 80000，来自第 4 节中的 KNN 实验设置)。在这种情况下，通过最大化CMS BW的利用率并将ERR接近ARR来提高区域\_x0005\_4的上界对于性能至关重要。

为此，以下小节提出了CMS的两个关键组件，以提高CMS的BW利用率，从而提供高ERR。

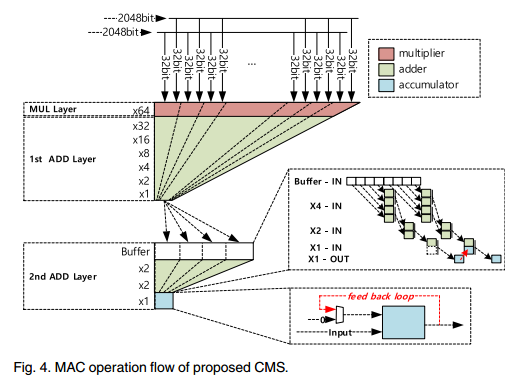
**3.2.1 Load Balancer**

使用 CXL 接口将 DDR 内存与 NDP 逻辑集成需要 CXL-AXI-DDR 接口的协议转换。在这里，我们发现了由于交错方法导致的性能下降点，尤其是在通过 CXL 接口将连续数据加载到 DDR 内存的多个通道时。 首先，在未应用交错的情况下，它会在一个通道中填充连续数据，然后依次填充其他通道。在这种情况下，NDP 只能使用单通道带宽，这使得所使用的总内存BW只有1/n(n=内存通道数)。其次，在通过更改 AXI 地址映射使用交错的情况下，由于 AXI 规范的地址边界约束，不允许少于 4KB 的交错。因此，NDP 在给定时间内仍然只使用 1/n 的带宽。 第三，主机 CPU 上的软件可以执行交错。在这种情况下，程序员应该手动重新排列用户空间中的数据，以便跨内存通道以交错方式存储表数据的连续内存地址。此外，它可能带来较高的软件开销，例如内存开销。



为了解决这个问题，我们设计了一种新的硬件逻辑，负载平衡器(load balancer LB)，它以host-transparently(主机透明)的方式在 64 字节内存访问单元中跨通道分发数据。如图3所示，我们的LB将每个通道的连续地址空间重新映射到跨通道。它通过以DDR访问粒度为单位交错每个通道来最大限度地提高内存带宽利用率。由于我们在硬件逻辑中实现了LB，因此减轻了主机CPU在内存映射中的负担，比涉及软件的数据管理具有更好的性能。我们在4.2节讨论LB效应的实验结果。

**3.2.2 MAC Block Optimization**

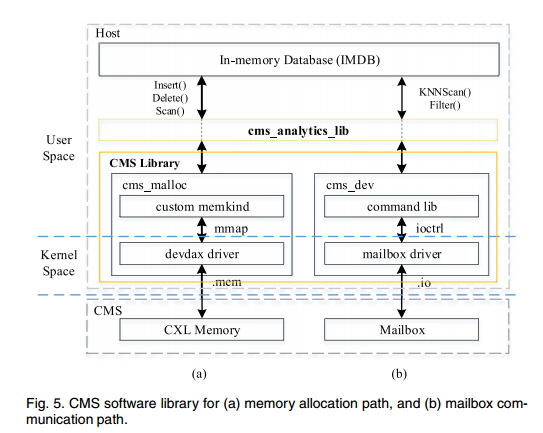


由于2048位数据通过前面描述的LB逻辑无缝地输入到NDP逻辑中，下一阶段的MAC运算符需要一种能够处理64x FP32数据(= 2048位)而不会降低性能的架构。然而，**反馈循环结构**固有的延迟是典型的MAC累加器关键的性能下降的因素。

我们提出的MAC架构有效地隐藏了累加器的延迟，以防止性能下降。图4是我们CMS的MAC操作流程。在我们目前的原型中，MAC操作符支持64 FP32执行。它通过三个主要步骤执行计算:MUL层、第一ADD层和第二ADD层。MUL层并行处理64个操作和64个数据。然后，两个ADD层执行级联加法，以获得64次执行的聚合结果。

在我们的原型设计过程中，我们发现当通过单个级联序列获得MAC操作结果时，由于最后一级的反馈环路阻塞，可能会出现称为吞吐量气泡的不希望出现的延迟。我们通过在第1层和第2层之间放置一个4深度的缓冲区来优化积累块，从而解决了这个问题。如图4所示，我们的设计缓冲了第一个ADD层的结果，第二个ADD层以x4为单位执行加法以获得最终的累积值。它成功地在缓冲时间内隐藏了最后一级反馈回路的运行时间，因此它可以无缝运行而不会阻塞输入。

**3.3 Software Architecture**



我们构建了基于 Apache Arrow [21] 的软件 API，名为 cms\_analytics\_lib，使我们的 CMS 设计能够灵活地适用于一般的数据分析平台。如图 5 所示，我们的 API 定义了 Insert、Delete、Scan 和 KNNScan 函数来与内存数据库进行通信。cms\_analytics\_lib 将 cms\_malloc 和 cms\_dev 堆栈作为下层，分别用于控制 CXL 内存和 NDP 单元。cms\_malloc 堆栈分配和释放内存，使用 CXL.mem 协议管理 CMS 内存中的相关元数据。cms\_dev 堆栈在通过 CXL.io 协议读写 CMS 的邮箱时控制 NDP 逻辑。我们实现了基于轮询而不是中断的邮箱读操作，以获得更高的性能。

1. **EVALUATION**

**4.1 Experimental Setup**

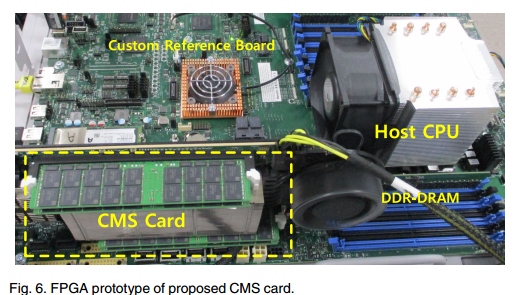


图6显示了我们的原型配置。我们使用 Xilinx Alveo U250 FPGA 和四通道 的DDR 内存（DDR4/2400Mbps/128GB）制作了 CMS 卡的原型。我们综合了在 RTL 中实现的 KNN 操作逻辑和 Xilinx 的 MicroBlaze 进行命令解析。为了评估所提出的CMS的有效性，我们构建了三种不同的设计配置:1)CPU\_only, 2) CME (CXL Memory Expansion)和3)CMS。CPU\_only是由一个CPU (Intel的未来至强处理器)和两个通道DDR内存(DDR5/4800Mbps/ 64GB)组成的传统系统机箱。我们使用VTS PM测量评估系统的功耗[22]。我们对KNN基准进行了实验[23]，其中特征向量是fp32类型特征的256维向量。对于KNN查询，每个系统将输入的特征向量与数百万个数据库特征的不同范围进行比较，并提取最相似的25个条目。

**4.2 Experimental Results and Analysis**

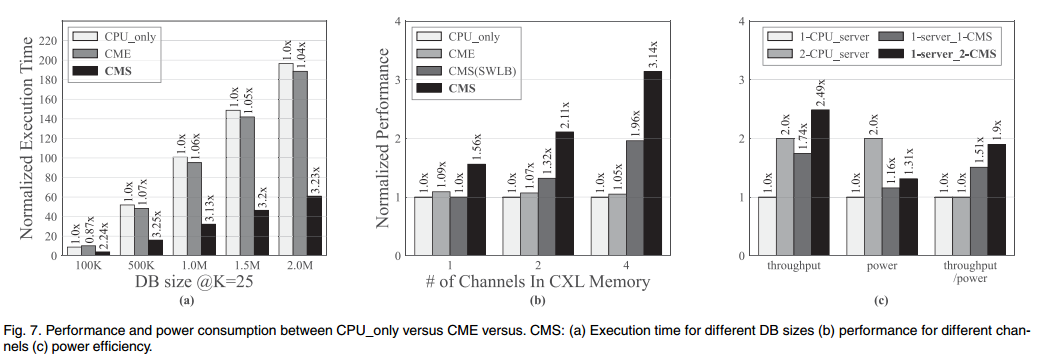


图 7a 显示了我们的 CMS 和两个参考系统( CPU\_only 和 CME )的执行时间，这些执行时间在 K=25 时的**不同数据库的大小都**归一化为 CPU\_only。结果表明，对于 2M 的数据库大小，CMS 的性能分别比 CPU\_only 和 CME 高出 3.23 倍和 3.1 倍。如第 3.2 节所述，由于在 CME 的情况下数据处理不会发生在 CXL 内存中（RR=1），因此 CXL 带宽完全限制了系统性能。但是，即使在这种情况下，我们的 CMS 也可以提高性能。这是因为 CMS 上的 NDP 代表主机访问和处理大量数据，从而显著减少了通过 CXL 传输的数据大小，即仅返回较小大小的计算结果。 接下来，我们验证 CMS 的可扩展性。图 7b 显示了随着 CXL 内存通道数的增加，性能如何变化。对于 CME，即使 CXL 内存的通道数从 1 CH（通道）增加到 4 CH，也几乎没有性能提升效果。相比之下，所提出的 CMS 的性能随着 CXL 内存的通道数（=带宽）的增加而按比例增加。这意味着有效吞吐量按比例增加，表明尽管 CXL 的带宽有限，但使用 NDP 的 CMS 可以充分利用内部内存带宽。 图 7b 还显示了 LB 组件对系统性能的影响。虽然 CMS 默认包含我们在第 3.2 节中描述的 LB 逻辑，但 CMS（SWLB）指的是主机在软件上执行 CXL 内存通道之间交错的情况。我们观察到基于软件的通道交错会给主机带来负担，导致 1.6 倍的性能下降。 图 7c 显示了所提出的 CMS（Scaleup）与一组 CPU 服务器（Scale-out）的功率效率。结果表明，CMS 可显著节省大规模功耗。例如，与两台单 CPU 服务器集群相比，配备两块 CMS 卡的单 CPU 服务器功耗降低 34%，吞吐量/功耗效率提高 90%。因此，我们得出结论，与最新的应用市场中存在内存扩展压力的现有服务器相​​比，CMS 是一种具有 TCO（Total cost of ownership总拥有成本）竞争力的有前途的解决方案。

1. **CONCLUSION**

在本研究中，我们提出了一种支持 NDP 的 CXL 内存，用于内存密集型应用，并进行了实际原型演示。 我们的设计通过采用优化的 NDP 核心并提供更大的内存节点容量来解决 CXL 中的数据移动瓶颈问题。实验结果表明，与传统 CPU 系统相比，我们的设计在执行时间上提高了 3.2 倍，节能 34%。