技术调研报告：异构融合计算技术工作原理深度解析

摘要

异构融合计算技术代表了当代高性能计算领域的核心发展方向，旨在通过整合不同类型的处理器（如CPU、GPU、FPGA、ASIC等）以应对日益增长的计算需求和能效挑战。本报告将深入剖析异构融合计算的六个关键层面：异构处理器协同、任务调度与划分、统一内存架构、高效互联通信、多架构编程模型以及软硬件协同优化。通过详细阐述各层面的工作原理、关键技术、具体事例及形象类比，本报告旨在为读者提供一个全面且深入的异构计算技术视图，揭示其如何通过深度融合实现性能飞跃，并展望其未来发展趋势。

1. 引言：异构融合计算的崛起

异构计算是一种在单一系统内集成并利用多种不同类型处理器或核心的计算范式，例如中央处理器（CPU）、图形处理器（GPU）、现场可编程门阵列（FPGA）和专用集成电路（ASIC）等 1。传统计算系统主要依赖单一类型的处理器，通常是CPU，但随着摩尔定律的物理极限逐渐显现，以及人工智能、机器学习和科学模拟等特定工作负载对计算效率和能效提出更高要求，异构计算已成为缓解算力瓶颈的关键策略 1。通过将任务动态分配给最适合的处理器单元，异构系统能够优化资源利用率，最大化计算性能，并有效降低延迟和功耗 1。

当前，行业正转向以异构为基础的技术新生态，主流芯片供应商正大力布局异构计算，试图构建更完整的生态系统 6。异构计算将工作负载根据其特性和需求进行更精细的拆分，然后逐步统一化和标准化处理 6。未来，异构计算的设计将根据不同的场景、数据种类以及对处理延迟和带宽的要求进行定制 6。本报告将围绕异构处理器协同、任务调度与划分、统一内存架构、高效互联通信、多架构编程模型以及软硬件协同优化这六个核心层面，对异构融合计算的具体工作原理进行深入探讨。

2. 异构处理器协同：各司其职，合力致远

异构处理器协同是异构计算的基石，它通过发挥不同处理器的独特优势，实现整体计算性能的显著提升。

CPU：通用任务的指挥官

中央处理器（CPU）是通用型处理器，擅长串行处理和复杂逻辑控制。在异构系统中，CPU通常扮演“指挥官”的角色，负责资源管理、任务分配以及对各异构加速单元所得结果进行后处理 1。它统筹整个计算系统的运作，确保不同类型的计算任务能够被准确地导向最合适的执行单元。可以将CPU好比一支交响乐队的指挥，负责统筹全局，将不同的乐章（任务）分配给最擅长演奏的乐器组（异构处理器），并确保它们协同演奏，最终呈现出完整的音乐作品。

GPU：并行计算的加速器

图形处理器（GPU）以其强大的并行处理架构而闻名，尤其擅长执行大规模简单重复的计算任务，如图形渲染、图像处理、模式识别、机器学习和科学模拟等 1。它通过同时执行数千个计算单元，实现对海量数据的快速处理，在单指令多数据（SIMD）和单指令多线程（SIMT）操作上表现卓越 2。GPU如同乐队中的小提琴组，每个小提琴手（处理单元）都执行着相似的动作（并行计算），但汇聚起来就能产生宏大而富有层次的声音，极大地提升了整体的“演奏速度”。

FPGA：可重构的定制专家

现场可编程门阵列（FPGA）是一种高度灵活和可重构的硬件平台，能够根据特定算法需求进行硬件级别的重新配置 1。它由逻辑块、可编程互连和输入/输出（I/O）块组成，可以实现流水线并行和数据并行 1。FPGA适用于实时信号处理、硬件加速机器学习算法以及定制硬件设计等任务 1。其优势在于无与伦比的灵活性和能效比，能够根据工作负载的特定需求进行优化，例如处理一个数据包的10个步骤，FPGA可以搭建一个10级流水线，每个数据包流经10级后即可完成处理 10。FPGA则像一个多功能定制工具箱，可以根据不同的任务需求，快速地重新组合内部的工具（逻辑电路），形成最适合当前任务的“专用工具”，从而实现高效且灵活的“工作”。

ASIC：极致效率的专精者

专用集成电路（ASIC）是为特定逻辑功能定制设计的芯片 8。它在特定任务上具有通用处理器无法比拟的性能、速度和低功耗优势 8。然而，ASIC的缺点是缺乏灵活性，无法轻易适应其他任务，一旦算法和流程变更，可能导致ASIC失去其价值 8。ASIC好比一个高度专业化的机器人，它被设计出来只做一件事情，并且能以极致的效率和精度完成这项任务，但无法胜任其他任何工作。

协同机制与案例

异构计算的核心在于利用每种处理器的独特优势，并动态地将任务分配给最适合的单元，以优化资源利用率和最小化延迟 1。一个典型的CPU+GPU+FPGA异构计算系统协同机制如下：CPU主机单元首先对任务进行预处理，并将其分类为需要高计算速度或图像处理的任务（第一类），以及需要低功耗、接口通信或数据加解密的任务（第二类） 7。随后，CPU通过OpenCL ICD Loader发现可用的GPU和FPGA加速单元，并将第一类任务分配给GPU进行并行处理，将第二类任务分配给FPGA进行串行或并行处理 7。GPU和FPGA通常都通过OpenCL语言进行编程 7。各加速单元完成任务后，通过PCIe等高速互联接口将计算结果传回CPU主机单元 7。最后，CPU主机单元读取、整合并对结果进行后处理，最终反馈给用户 7。这种机制充分发挥了CPU的管理优势、GPU的并行处理优势和FPGA的能效比与灵活性。

异构系统架构（HSA）是协同的强大推动者。HSA是一套跨厂商的规范，旨在将CPU和GPU集成到同一总线上，实现内存和任务共享 2。其主要目标是降低CPU、GPU及其他计算设备之间的通信延迟，并从程序员的角度提高兼容性，从而减轻程序员手动管理不同设备内存之间数据移动的复杂任务 2。HSA通过定义统一的虚拟地址空间，使计算设备能够共享页表，从而通过共享指针交换数据，实现零拷贝操作 2。这种架构使得任何加速器，例如图形处理器，都能够以与系统CPU相同的处理级别运行 2。

处理器从单一通用性向异构专业化演进的趋势，是摩尔定律在通用CPU性能提升上遭遇瓶颈后，对算力需求爆炸式增长的必然回应。这不仅仅是简单的“加法”，而是通过“术业有专攻”来突破传统架构的限制，实现更优的性能功耗比。当通用CPU的性能提升速度无法满足特定工作负载的需求时，引入专门的硬件加速器成为新的发展方向 1。这种专业化是应对物理限制和应用需求多样化的直接策略，是计算架构演进的深层驱动力。

异构处理器协同的真正挑战，已从如何独立优化每个处理器，转向如何高效地“编排”和“通信”这些特性迥异的单元。这意味着，互联通信和统一内存架构（后续章节将详述）不再仅仅是辅助功能，而是异构协同能否充分发挥潜力的决定性因素。如果各司其职的专家们无法顺畅交流，其合力就会大打折扣。因此，成功实现异构协同的关键在于构建高效、低延迟、高带宽的通信基础设施，并提供易于编程的抽象，以降低数据流动的管理负担，从而避免即使单个处理器性能再强，整体系统的优势也难以体现的情况。

下表对异构处理器的主要特性进行了对比：

表 1: 异构处理器特性对比

处理器类型

主要功能

核心优势

典型工作负载/应用场景

主要局限性

CPU

通用计算、逻辑控制、资源管理

灵活、擅长串行处理、分支逻辑

操作系统、通用应用、复杂控制

并行计算能力有限、能效相对较低

GPU

大规模并行计算、图形渲染

高吞吐量、擅长数据并行（SIMD/SIMT）

游戏、AI训练/推理、科学模拟、3D建模

串行处理能力有限、通用性不如CPU

FPGA

可重构硬件加速、定制逻辑

极高灵活性、能效比高、可现场重构

实时信号处理、硬件加速、网络处理、原型验证

设计复杂、开发周期长、频率相对较低

ASIC

特定功能硬件加速

极致性能、极低功耗、体积小

加密货币挖矿、AI推理专用芯片、数据中心特定加速

缺乏灵活性、无法重构、开发成本高、周期长

3. 任务调度与划分：智慧分配，效率为先

在异构计算环境中，如何高效地将计算任务分配给最合适的处理器并进行调度，是决定系统整体性能的关键。这涉及到复杂的权衡，以最小化总体执行时间、平衡负载并减少通信开销 12。

任务划分策略 (Task Partitioning Strategies)

在异构系统中，由于计算资源能力和通信开销的差异，实现高效的任务分配和负载均衡面临显著挑战 12。任务通常被建模为有向无环图（DAG），其中节点表示任务，边表示任务间的数据依赖关系 14。

任务划分策略主要分为静态、动态和混合方法。静态划分在任务执行前预先分配，优点是开销低，但缺点是缺乏对运行时变化的适应性 12。例如，递归二分法和图划分是常用的静态方法 12。动态划分根据系统运行时性能实时调整任务分配，优点是响应性强，但会引入同步和通信开销 12。工作窃取（work stealing）和主从（master-slave）模型是典型的动态策略 12。混合方法则结合了静态和动态方法的优点，通常是先进行初始静态分配，再根据运行时情况动态调整 12。自适应混合任务划分算法（AHTPA）通过对每个节点进行性能分析（如FLOPS、内存、I/O带宽、网络延迟），集成静态和动态策略以实现最优任务分配 12。例如，快手推荐系统采用计算与存储分离的架构模式，参数服务器负责存储和实时更新海量用户画像和模型参数，而计算任务则在异构计算单元上执行，这体现了高层次的任务逻辑划分 3。

任务调度算法 (Task Scheduling Algorithms)

调度算法旨在满足任务完成时间期望 17，最小化总执行时间，平衡负载，并减少通信开销 12。异构调度面临诸多挑战，包括节能 17、高调度成本 15、可扩展性 12，以及如何根据不同场景、数据种类和处理延迟、带宽要求进行精细化工作负载拆分和分配 6。

典型的调度算法包括列表调度算法，该算法为每个任务分配优先级，并按优先级顺序将任务调度到能最小化预定义成本函数的处理器上 15。其中，异构最早完成时间（HEFT）算法是一种静态调度算法，它在每一步选择具有最高“向上排名”（upward rank）的任务，并将其分配给能最小化其最早完成时间（Earliest Finish Time, EFT）的处理器 14。向上排名是任务到出口任务的最长路径长度，包括任务自身的计算成本 15。关键路径处理器（CPOP）算法是另一种静态调度算法，它通过计算任务的“向上排名”和“向下排名”（downward rank）之和来确定任务优先级。该算法旨在将所有关键任务（位于DAG关键路径上的任务）调度到单个处理器上，以最小化关键任务的总执行时间 15。针对动态环境，研究者提出了基于启发式和强化学习算法的动态调度方法 13。例如，针对Spark虚拟异构集群的新调度策略，在仿真和真机模拟环境下均显示出显著缩短运行时间并使节点分配更均匀的优势 21。

主流编程模型中的调度实践 (Scheduling Practices in Mainstream Programming Models)

OpenCL： OpenCL作为一种开放标准，提供了在多核CPU、GPU、FPGA等不同平台上的程序可移植性 22。其多任务调度框架能够根据设备实际运行状态和资源竞争情况，智能地确定最优任务分配 25。该框架通过预测内核在GPU上的加速比（基于静态代码结构和运行时输入数据大小），来优化任务优先级和调度 22。高加速比的内核任务被调度到GPU，低加速比的则调度到CPU。为避免额外开销，OpenCL调度器通常不将单个内核的工作拆分到多个设备上 22。

CUDA： CUDA编程模型是异构的，由CPU（主机）和GPU（设备）组成，两者拥有独立的内存空间 26。典型的CUDA程序执行流程包括：声明和分配主机与设备内存，初始化主机数据，数据从主机传输到设备，执行一个或多个内核，最后将结果从设备传输回主机 26。CUDA的线程调度以Warp（通常32个线程）为基本单位，由SM（Streaming Multiprocessor）的Warp调度器进行SIMD方式执行 28。CUDA支持动态并行性，允许在运行时创建和管理并行线程 29。提前调度（Ahead-of-Time, AoT）技术，如Nimble引擎所采用的，能够在GPU内核执行前完成调度过程，从而消除运行时调度开销，尤其适用于深度学习中包含大量小而短GPU任务的场景。它还能自动化流分配和同步过程 30。

SYCL： SYCL运行时利用乱序队列（out-of-order queues）根据数据依赖关系隐式构建任务执行图 31。在统一共享内存（USM）环境下，用户需要通过事件（events）显式构建任务间的依赖关系 31。syclFlow等高层编程接口允许用户定义任务依赖图，然后通过拓扑排序调度任务，并提交到SYCL队列，利用事件同步来管理依赖 31。

oneAPI (oneTBB)： oneAPI Threading Building Blocks (oneTBB) 提供了一个任务调度器，作为其算法模板和任务组的引擎 32。任务被定义为计算的量子，调度器将任务映射到工作线程，且这种映射是非抢占式的，即一旦线程开始运行任务，它将绑定到该任务直到完成 32。调度器会根据可用工作线程的数量动态调整实际并行度，因此不能保证所有潜在并行任务都实际并行执行 32。

异构系统中的任务调度复杂性源于需要在性能、能耗、延迟、吞吐量和负载均衡等相互冲突的目标之间进行权衡，并且要适应不同处理器的独特特性。这不仅仅是追求“快”，更是追求在给定任务和硬件条件下的“最优效率”。调度问题是一个多维度的优化挑战，其复杂性随系统规模和工作负载动态性而增加 12。因此，仅仅依靠简单的启发式规则已不足以实现最佳性能，系统需要更智能、更灵活的决策机制。向自适应、机器学习驱动和提前调度（AoT）等策略的转变，反映出业界认识到静态、一刀切的调度方法已无法满足动态、复杂工作负载的需求 13。

主流编程模型（OpenCL、CUDA、SYCL、oneAPI）在调度实践上的演进，揭示了将低级调度复杂性从开发者手中抽象出来的持续努力，同时仍保留对性能的控制。早期CUDA和OpenCL需要开发者手动管理内存和调度 26。SYCL和oneAPI则致力于提供更高级别的抽象（单源C++、USM、隐式依赖图），以简化编程 31。Nimble的AoT调度更是将调度决策前置 30。这种趋势表明，软件栈正在承担越来越多的优化责任，允许开发者专注于算法逻辑，而非底层的硬件细节。这种抽象和自动化对于异构计算的广泛采用至关重要，也预示着未来编译器和运行时系统将在任务放置和执行中扮演越来越智能的角色，可能利用AI/ML技术进行动态优化，从而弥合高级可编程性与硬件特定性能之间的差距。

下表对异构计算中典型的任务调度算法进行了对比：

表 2: 典型任务调度算法对比

调度算法

类型 (静态/动态/混合)

主要目标

核心机制/启发式

优点

缺点/复杂性

HEFT

静态

最小化总完成时间

基于向上排名，分配给最早完成的处理器

性能优异，调度时间相对较低

无法适应运行时变化，对任务图敏感

CPOP

静态

最小化关键路径执行时间

基于向上排名和向下排名，关键任务优先调度到单处理器

优化关键路径，提高并行度

调度时间可能较高，对非关键任务优化不足

动态启发式/ML

动态/混合

最小化总完成时间，负载均衡，节能

运行时性能预测，自适应调整，强化学习

适应性强，能处理动态负载和多目标优化

引入运行时开销，实现复杂，依赖准确预测

OpenCL调度框架

动态

最大化系统吞吐量，最小化平均周转时间

基于内核加速比预测和输入数据大小，动态分配CPU/GPU

提高异构平台利用率，避免不当任务分配

预测模型准确性影响性能，需提取静态代码特征

CUDA AoT调度 (Nimble)

提前/静态

消除运行时调度开销，加速小任务执行

预先分析计算图和输入形状，生成任务调度表

显著降低调度开销，提高吞吐量

主要适用于静态神经网络，不适用于动态图

SYCL运行时调度

动态

隐式任务图并行，简化依赖管理

乱序队列和事件同步，隐式数据依赖推断

简化编程模型，支持单源C++

USM环境下需显式管理事件，可能引入同步开销

oneTBB任务调度器

动态

并行化计算密集型工作负载

非抢占式任务映射到工作线程池

易于并行化CPU密集型任务

不保证实际并行执行，不适用于生产者-消费者模型

4. 统一内存架构：数据共享，消除壁垒

统一内存架构是异构计算中解决数据传输瓶颈和简化编程复杂性的关键技术。

核心理念与传统内存的挑战 (Core Concept & Challenges of Traditional Memory)

在传统异构系统中，CPU和GPU通常拥有独立的物理内存空间，例如CPU的RAM和GPU的显存 26。这意味着当CPU和GPU需要协同处理数据时，必须通过显式的数据拷贝操作（如cudaMemcpy）在它们各自的内存空间之间进行数据传输 26。这种显式拷贝会引入显著的延迟和带宽瓶颈，尤其对于处理大量数据的应用（如视频编辑、3D渲染、机器学习）而言，会严重影响系统性能和能效 27。

统一内存（Unified Memory, UM）旨在提供一个单一、连贯的内存地址空间，该空间可由系统中的任何CPU或GPU访问 1。它消除了程序员手动管理数据传输的需要，从而简化了编程模型。传统内存架构好比每个部门（CPU、GPU）都有自己的独立图书馆，部门间借书需要通过复杂的申请和搬运流程。而统一内存架构则像一个中央图书馆，所有部门都可以直接访问和共享同一批书籍，大大简化了知识（数据）的流通。

实现机制 (Implementation Mechanisms)

统一内存通过构建统一的虚拟地址空间来实现，允许处理器通过共享指针而非数据拷贝来交换数据 2。按需分页迁移（On-demand Paging Migration）是UM的核心机制之一。当某个处理器（如GPU）尝试访问其本地内存中不存在但属于统一内存空间的数据时，系统会自动触发页面错误（page fault），并将所需数据页从其当前位置（如CPU内存）迁移到访问该数据的处理器本地 43。

在NVIDIA CUDA中，通过cudaMallocManaged()函数或\_\_managed\_\_标识符来分配托管内存，使CPU和GPU都能直接访问。底层系统会根据需要自动执行数据拷贝和迁移 41。即使在物理内存分离的设备上，系统也会在后台进行必要的内存分配和数据拷贝 41。异构系统架构（HSA）明确要求计算设备共享页表，并通过定制的内存管理单元（MMU）实现统一的虚拟地址空间，从而支持设备间通过共享指针进行数据交换，实现零拷贝操作 2。Apple M1芯片是物理层面实现统一内存的典范，其CPU、GPU和神经网络引擎共享同一物理内存池，并能根据需求动态分配内存，从而实现极高的数据传输效率和低延迟 39。Intel oneAPI中的统一共享内存（Unified Shared Memory, USM）作为SYCL标准的一部分，通过增强现有的基于缓冲区的接口，提供了一种类似于CUDA统一内存的低级编程模型 36。

优势 (Advantages)

统一内存的引入带来了多方面优势。首先，它显著简化了编程模型，消除了手动管理CPU和GPU之间数据传输的繁琐和易错性，大大降低了开发复杂性，提高了程序员的生产力 41。其次，通过减少数据拷贝和降低数据传输延迟，UM显著提升了图形密集型和数据密集型应用的性能。数据可以动态地迁移到访问它的处理器本地，从而优化内存访问速度 39。此外，UM通过最小化冗余数据拷贝，释放更多内存用于更大的模型或数据集，并能根据CPU和GPU的需求动态分配内存，提高了内存使用效率 39。对于多GPU系统，统一内存简化了数据共享和同步，尤其有利于分布式深度学习训练，减少了多GPU训练设置中的瓶颈 45。最后，UM支持处理超出单个GPU物理显存容量的数据集，通过利用系统RAM来扩展GPU的可用内存，增强了可扩展性 45。

挑战与优化 (Challenges & Optimizations)

尽管统一内存简化了编程，但按需分页迁移可能引入性能开销。页面错误处理可能需要几十微秒，涉及TLB失效、数据迁移和页表更新，这可能导致应用程序某些部分的执行停止，影响整体性能 43。此外，确保CPU和GPU同时访问统一内存时的数据一致性是一个挑战，尤其是在计算能力较低的设备上 43。如果频繁发生页面错误，统一内存的性能可能变得不可预测 43。

针对这些挑战，研究者提出了多种优化策略。数据预取（Data Prefetching）和内存建议（Memory Advice）通过预先将数据移动到预期访问的处理器，减少页面错误和传输延迟 43。流关联（Stream Association）技术允许将内存分配与特定的CUDA流关联，以指示内核对数据的使用模式，从而实现数据传输优化和更大的并发性 43。此外，编译器技术也发挥着越来越重要的作用。例如，G10架构利用深度学习工作负载中张量行为的高度可预测性，通过编译器技术提前调度数据放置和迁移，从而在透明地扩展GPU内存容量的同时隐藏数据传输开销 49。

统一内存不仅仅是编程便利性功能，更是旨在缓解异构系统中不同内存空间造成的“内存墙”瓶颈的根本性架构变革。传统架构中独立内存和显式拷贝带来的问题，通过统一内存的自动化数据移动得到缓解 26。然而，这种自动化并非没有代价，页面错误处理可能导致性能下降 43。这表明，虽然抽象层降低了开发难度，但底层的性能行为仍然需要深入理解和优化，统一内存将复杂性从显式拷贝转移到了隐式管理和潜在的页面错误开销。

统一内存的持续演进（例如CUDA的流关联、Intel的USM、G10的编译器驱动预取）表明，仅仅拥有“统一地址空间”是不够的。真正的性能优化需要运行时和编译器内部更深层次的智能，能够“预测”并“主动管理”数据移动，有效地将潜在的瓶颈（页面错误）转化为隐藏的性能优化 43。这种趋势表明，统一内存的真正价值在于其背后的智能管理机制，它通过软件和硬件的紧密协同，将数据传输的复杂性内化，从而在不牺牲性能的前提下实现编程的简化。

下表对不同主流平台的统一内存实现进行了比较：

表 3: 统一内存模型对比

统一内存模型/平台

核心机制

数据传输方式

共享内存范围 (虚拟/物理)

主要优点

关键局限性/注意事项

NVIDIA CUDA Unified Memory

统一虚拟地址空间，按需分页迁移

自动/隐式

虚拟

简化编程，减少显式拷贝，支持超大模型

页面错误可能引入延迟，需要显式同步

AMD HSA Memory Model

统一虚拟地址空间，共享页表，定制MMU

自动/隐式

虚拟

降低通信延迟，简化编程，支持零拷贝

依赖硬件支持，需要操作系统内核和驱动支持

Intel oneAPI (SYCL USM)

统一共享内存，可与缓冲区模型结合

自动/隐式 (USM)

虚拟

跨平台，单源C++，提供灵活的内存管理选项

性能优化仍需关注底层细节，显式事件同步

Apple M1 Unified Memory

CPU/GPU/NPU共享同一物理内存池

物理共享，无需拷贝

物理

极致性能和能效，数据传输延迟极低

硬件紧耦合，不可升级，仅限Apple生态

5. 高效互联通信：高速公路，畅通无阻

高效的互联通信是异构计算系统发挥其潜力的关键，它确保了不同处理器之间的数据能够快速、流畅地交换。

互联通信的重要性 (Importance of Interconnect Communication)

互联通道是现代计算系统内部数据交换的生命线，连接着CPU、GPU、FPGA、存储等各种组件 50。随着计算需求的增加，尤其是复杂应用（如生成式AI）的崛起，互联通信的效率和性能变得至关重要，直接影响整体系统速度、可扩展性和可靠性 50。高效互联能够处理海量数据流，最大限度地减少瓶颈，并实现实时处理，从而有效缓解数据传输延迟 50。此外，互联技术还确保所有处理器都能访问最新数据，维持内存一致性 50。可以将高效互联通信好比一个城市的高速交通网络，它确保了城市不同区域（处理器）之间的人员和物资（数据）能够快速、顺畅地流动，避免了交通堵塞（数据传输瓶颈），从而保证了整个城市的正常高效运转。

关键互联技术 (Key Interconnect Technologies)

PCIe (PCI Express)： PCIe是一种标准化的、高速的串行扩展总线接口，广泛用于连接CPU、GPU、SSD等高带宽组件 7。CPU主机单元通常通过PCIe与GPU和FPGA异构加速单元进行通信 7。其可扩展架构支持多通道，PCIe 4.0每通道提供高达16 GT/s的数据传输速率 51。

NVLink (NVIDIA)： NVLink是NVIDIA开发的专有高速互联技术，旨在提供GPU之间的高带宽、低延迟通信，性能超越PCIe 51。NVLink 4.0可为每个GPU提供高达900 GB/s的双向带宽，对于多GPU数据吞吐量至关重要，尤其受益于深度学习任务中大型数据集在GPU间的分布式处理 51。它通过实现GPU到GPU的直接通信，避免了CPU的参与，从而降低了延迟并提高了系统性能 51。例如，NVIDIA DGX系统就使用NVLink连接GPU进行AI训练 51。

CXL (Compute Express Link)： CXL是一个开放的行业标准互联协议，旨在实现CPU、GPU、加速器和内存设备之间的缓存一致性内存共享 51。CXL利用PCIe的物理层，这意味着CXL设备可以与现有的PCIe基础设施兼容，并在此基础上提供额外的功能和能力 52。CXL引入了CXL.cache和CXL.mem协议，用于实现异构计算环境中的低延迟内存访问和动态内存池化与共享 51。CXL 3.0将带宽提升至PCIe Gen 6的两倍，并支持原子操作和增强安全性，适用于AI等高需求应用 52。

InfiniBand： InfiniBand是一种高性能网络标准，在大型分布式训练环境中至关重要。它提供服务器之间的低延迟和高吞吐量通信，确保处理器和数据流的有效协调，适用于资源密集型AI应用和HPC设置 50。它支持CPU-free的远程直接内存访问（RDMA），进一步降低了延迟 51。

其他互联技术： 随着800 GbE等标准的出现，以太网也提供了足够的带宽来支持AI工作负载，并与PCIe、NVLink、InfiniBand和CXL共存，主要用于系统间的数据传输 51。此外，通用芯片互联标准（UCIe）、高级接口总线（AIB）、Bunch of Wires (BoW) 和硅光子互联（Silicon Photonics Interconnects）等技术主要用于芯片内部或芯片间的高速数据传输，尤其在小芯片（chiplet）和异构集成设计中发挥关键作用，提供超高带宽和低延迟 55。

异构系统中的应用 (Applications in Heterogeneous Systems)

互联技术是实现异构系统中不同处理单元之间高效通信的基础 50。高性能互联技术支持更灵活、可扩展的计算架构，尤其在高性能CPU和GPU系统中 50。互联技术是小芯片架构的核心，允许将计算、内存、互联和I/O功能独立设计并集成到异构多芯片系统中，从而实现更高的集成度和性能 55。通过使用中介层（interposer）连接多个小芯片（2.5D集成）或通过硅通孔（TSV）实现垂直堆叠（3D堆叠），互联技术能够实现高互联密度和带宽，进一步提升系统性能 55。

挑战 (Challenges)

高效互联通信面临多重挑战。数据传输延迟或带宽不足会严重阻碍高需求应用的性能，特别是依赖快速数据处理的生成式AI模型 50。互联的硬件故障或数据损坏可能导致系统范围的中断，造成停机时间和经济损失 50。高速互联的特性使其成为数据拦截和侧信道攻击的潜在目标，存在敏感信息泄露的风险 50。此外，缺乏通用标准会使跨厂商小芯片集成和互操作性复杂化，尽管UCIe等标准正试图解决这一问题 55。

随着计算需求的不断升级，特别是数据密集型AI/ML工作负载的爆发，互联技术已从简单的“数据管道”演变为智能的、缓存一致的通信结构。从PCIe到NVLink和CXL的演进，标志着异构系统从通用总线架构向专业化、性能优化和内存一致性互联的转变，这对于释放异构系统的全部潜力至关重要。这种演进是由克服“数据移动瓶颈”的需求驱动的，该瓶颈正变得与“内存墙”同等关键 50。

CXL等标准的出现，以及UCIe在小芯片领域的推进，表明业界日益认识到开放、标准化和一致性互联对于未来异构系统可扩展性和互操作性的重要性 51。这预示着计算将走向更模块化和可组合的未来，不同厂商的组件可以无缝集成，从而促进创新，但也带来了系统设计和优化方面的新挑战。这种“标准化”和“开放性”的趋势是异构计算发展的重要信号，说明业界正努力打破专有壁垒，构建一个更灵活、更具互操作性的硬件生态系统，以支持“XPU战略”等更宏大的异构计算愿景 6。

下表对异构计算中关键互联技术进行了概览：

表 4: 关键互联技术概览

互联技术

关键特性

典型带宽/延迟

主要应用/使用场景

与其他技术的关系

PCIe

通用串行总线，点对点连接

高速，PCIe 4.0每通道16 GT/s

CPU与GPU/SSD等外设通信

CXL物理层基础

NVLink

NVIDIA专有，GPU间高速互联

超高带宽，NVLink 4.0达900 GB/s/GPU

多GPU深度学习训练，高性能计算

替代或补充PCIe，避免CPU参与

CXL

开放标准，缓存一致性内存共享

高速，CXL 3.0基于PCIe Gen 6

CPU/GPU/加速器内存池化，AI/HPC

基于PCIe物理层，扩展其功能

InfiniBand

高性能网络标准，RDMA

低延迟，高吞吐量

大型分布式训练集群，HPC

主要用于系统间通信，与PCIe/NVLink/CXL共存

6. 多架构编程模型：抽象统一，赋能开发

多架构编程模型是连接异构硬件与软件应用之间的桥梁，旨在降低开发复杂性，提高程序员生产力。

异构编程的挑战 (Challenges of Heterogeneous Programming)

异构计算产品涉及不同的系统架构、指令集和编程模型，这给软件开发者带来了巨大的难度 6。异构算力的融合使得编程模型与框架日趋复杂，迫切需要操作系统向应用提供统一、易用的编程抽象，以降低开发门槛 58。为了实现最佳性能，开发者往往需要手动管理数据移动、同步等低级细节，这牺牲了代码的可移植性，增加了复杂性，并要求开发者具备深厚的硬件知识 59。

内存层次结构是异构编程面临的硬件挑战之一。不同的计算核心（如GPU需要高带宽，传统CPU需要快速访问）对内存系统有不同的要求。设计一个能减少干扰并高效处理这些差异的内存层次结构是巨大挑战，且内存系统是功耗的重要来源 60。此外，如何连接不同的核心和内存模块，以及如何在不同核心之间分配工作负载以获得最佳性能和最低功耗，是贯穿整个计算堆栈的挑战 60。由于不同处理器能力的非对称性以及不透明的编程模型和操作系统抽象，异构系统在处理混合工作负载时，性能可预测性可能面临问题 4。异构编程的挑战好比让一支由不同语言使用者（不同架构）组成的团队，在没有统一翻译器和明确分工（编程模型）的情况下，共同完成一项复杂任务。

主流编程模型 (Mainstream Programming Models)

OpenCL： OpenCL是一个开放的行业标准，用于在各种现代CPU、GPU、DSP和FPGA等异构处理器上进行任务并行和数据并行计算 23。它提供了一套通用的语言、编程接口和硬件抽象，使开发者能够在异构环境中加速应用 23。OpenCL支持运行时编译，允许程序在目标硬件上本地运行，即使是原始开发者未曾接触过的平台 23。相较于OpenACC或CUDA，OpenCL暴露了更低的硬件控制层级，赋予程序员对并行化过程的完全控制，但这以牺牲部分易用性为代价 24。其内存模型定义了全局内存、常量内存、局部内存和私有内存四种类型 23。

CUDA (NVIDIA)： CUDA（Compute Unified Device Architecture）是NVIDIA开发的并行计算平台和编程模型，用于在NVIDIA GPU上进行通用计算 26。它通过C、C++、Fortran、Python、Julia和MATLAB等流行语言的扩展，使开发者能够利用GPU的强大并行处理能力加速应用 26。在CUDA模型中，CPU（主机）负责管理GPU（设备）内存并启动在设备上执行的内核函数 26。CUDA统一内存（Unified Memory, UM）的引入显著简化了内存管理，减少了显式数据传输的需求 27。NVIDIA提供了强大的CUDA生态系统，包括GPU加速库、编译器、开发工具和运行时库 61。

SYCL： SYCL是Khronos Group推出的一项免版税、跨平台抽象层，它基于OpenCL的底层概念、可移植性和效率，并允许使用完全标准的C++以“单源”风格编写异构处理器代码 34。SYCL支持单源开发，C++模板函数可以同时包含主机和设备代码，从而构建复杂的算法并在不同类型数据上重用 36。通过使用缓冲区（buffers）和访问器（accessors），SYCL减轻了程序员在主机和设备之间显式传输数据的负担；从SYCL 2020开始，也支持统一共享内存（USM）作为补充 36。SYCL是一个供应商中立的标准，在高性能计算（HPC）、人工智能和科学研究等领域获得了越来越多的采用 35。主要实现包括Intel的Data Parallel C++ (DPC++)、Codeplay的ComputeCpp（已弃用）、以及Codeplay和AMD/NVIDIA/Intel CPU支持的AdaptiveCpp 36。

oneAPI (Intel)： oneAPI是由Intel主导的开放、跨行业、基于标准、统一的、多架构、多供应商的编程模型，旨在为加速器架构提供通用的开发者体验 37。它基于SYCL规范，并以Data Parallel C++ (DPC++) 作为其核心编程语言 34。oneAPI的平台模型定义了主机和多个设备（CPU、加速器、计算单元、处理元素）之间的通信和协调 34。其执行模型规定了内核如何在设备上执行以及与主机交互，通过命令组提交到队列中 34。oneAPI还包括一套领域库（如Intel oneAPI Math Kernel Library），进一步简化了开发 64。

统一编程接口与单源编程 (Unified Programming Interfaces & Single-Source Programming)

统一编程接口和单源编程旨在降低软件开发者的难度，提高代码复用率，减少学习成本，从而提升开发效率和创新能力 6。单源编程允许开发者在同一个C++源文件中编写主机代码和设备代码，如SYCL和DPC++ 36。统一抽象将算法描述与底层硬件细节解耦，使开发者能够专注于算法本身，而无需过多关注特定硬件的实现细节 59。

编程模型演进趋势 (Evolutionary Trends in Programming Models)

编程模型正从早期需要手动管理大量底层细节的低级API（如CUDA、OpenCL）向更高级别的抽象（如SYCL、oneAPI）发展，以简化编程和提高可移植性 24。这种演进不仅追求代码的可移植性，更致力于实现“性能可移植性”，即代码在不同硬件上都能获得良好性能 65。未来的编程模型将越来越依赖于智能编译器优化和运行时系统，以自动处理任务放置、数据移动和同步，从而在不牺牲性能的前提下实现高级抽象 59。此外，针对特定应用领域，发展领域特定语言（DSLs）和框架，以引入领域知识，进一步解耦应用和架构优化，也是重要的发展方向 59。

异构硬件架构的激增，导致了异构计算领域的“软件危机”，即管理不同编程模型、指令集和内存层次结构的复杂性，极大地阻碍了开发者的生产力和代码可移植性 4。统一编程模型（如SYCL、oneAPI）的出现正是对此的直接回应，旨在抽象硬件细节，实现单源开发 34。然而，这种抽象往往伴随着对细粒度性能控制的权衡。这表明，虽然高级编程模型提供了便利，但要充分发挥异构硬件的潜力，仍然需要编译器和运行时系统在底层进行复杂的优化。

7. 软硬件协同优化：深度融合，极致性能

软硬件协同优化是异构计算实现极致性能和能效的关键，它要求硬件设计与软件需求紧密结合，共同应对计算挑战。

重要性 (Importance)

在异构计算中，软件和硬件的协同优化至关重要。仅凭硬件层面的突破不足以完全释放异构算力的潜力，必须通过“软硬兼施”才能更好地应对异构计算的需求 57。传统存储引擎中，压缩/解压、加密/解密等数据流处理任务通常由CPU直接完成，难以大规模并行处理，效率低下，CPU很容易成为性能瓶颈 68。通过软硬件协同优化，可以将CPU不擅长的计算任务卸载到其他异构计算硬件上，从而降低CPU负荷，提升系统整体性能 68。

对于移动设备而言，优化并非可选，而是将好想法转化为卓越执行的关键一步 5。异构计算旨在通过将计算任务发送到最适合的处理器来提高应用程序性能，同时改善散热和功耗效率 5。要实现更大的效益，需要从底层为异构计算设计的硬件架构，以及能够促进异构计算技术的软件栈 5。正是目的性构建的硬件与在更大系统抽象框架内提供细粒度控制的软件栈的结合，才能够实现异构计算所能提供的深度优化 5。

协同优化策略 (Co-optimization Strategies)

硬件设计为软件服务： 硬件设计需要考虑软件开发者的需求，降低异构资源的编程门槛。例如，高通骁龙移动平台的设计就旨在优化处理器之间的协同使用，如Hexagon DSP可以直接从传感器流式传输数据到DSP缓存，绕过DDR内存和相关的CPU数据传输周期 5。Adreno GPU支持64位虚拟寻址，允许共享虚拟内存（SVM），实现与Kryo CPU的高效协同处理 5。此外，ARM的Total Compute战略也采取整体方法进行SoC设计，确保其解决方案能够无缝、安全地处理日益复杂和计算密集型的工作负载 69。

编译器和运行时系统优化： 编译器和运行时系统在软硬件协同优化中扮演核心角色。它们负责将高级语言代码映射到异构硬件上，并进行设备特定的优化、任务划分、设备间通信和同步 59。例如，Liquid Metal等编译器和运行时系统旨在实现单语言编程异构计算平台，并在CPU和加速器（包括GPU和FPGA）上无缝协同执行程序 67。Intel oneAPI DPC++编译器通过其统一的、基于标准的编程模型，旨在为开发者提供跨架构的通用体验，并支持针对不同硬件的优化 38。NVIDIA CUDA Toolkit则提供了GPU加速库、编译器、调试和优化工具以及CUDA运行时，使开发者能够开发、优化和部署GPU加速应用 61。

领域特定优化： 针对特定应用领域（如AI/ML、图像处理、科学模拟），软硬件可以进行深度定制优化。例如，快手推荐系统基于Intel FPGA加速多元算力的成功尝试，证明了通过异构计算加速不同负载，能够显著提升推荐等场景下的系统吞吐与延时表现 57。

性能监控与调优： 持续的性能监控和调优是实现软硬件协同优化的重要环节。通过分析系统行为，识别瓶颈，并根据硬件特性调整软件策略，可以不断提升系统性能 5。

异构计算的性能提升不仅来源于单个处理器的进步，更关键在于不同类型处理器之间的高效协同，这正是软硬件协同优化的核心价值。硬件设计需要从一开始就考虑其与软件栈的协同作用，而软件则需要能够充分利用硬件的独特能力。这种协同是应对摩尔定律放缓和应用需求爆炸式增长的必然选择，它将计算系统推向一个更专业化、更高效的未来。

异构硬件架构的复杂性，使得传统上由软件或硬件单方面进行的优化变得不足。现在，硬件需要设计得更“软件友好”，提供更易于编程的接口和更透明的底层机制 66。同时，软件也需要更“硬件感知”，通过智能编译器和运行时系统，能够自动适配和利用不同硬件的特性 59。这种双向的适应和共同进化，是异构计算能否广泛普及和发挥其全部潜力的决定性因素。

8. 结论

异构融合计算技术通过整合CPU、GPU、FPGA、ASIC等多种处理器，实现了计算性能和能效的显著提升，已成为应对当代复杂计算挑战的核心范式。本报告从异构处理器协同、任务调度与划分、统一内存架构、高效互联通信、多架构编程模型以及软硬件协同优化六个层面深入剖析了其工作原理。

异构处理器协同的本质在于发挥各处理器的独特优势，并进行智能任务分配。CPU作为指挥官，GPU擅长并行，FPGA提供可重构性，ASIC则追求极致专精。这种协同模式的演进，体现了计算系统从通用性向专业化发展的趋势，以突破传统架构的性能瓶颈。其核心挑战已从独立优化转向高效的“编排”与“通信”，这强调了互联和内存架构的关键作用。

任务调度与划分在异构环境中至关重要，旨在平衡性能、能耗和负载。从静态到动态再到混合调度策略的演进，以及引入机器学习和提前调度技术，反映出业界对动态、复杂工作负载的适应性需求。主流编程模型在调度实践中不断将低级复杂性抽象化，预示着未来编译器和运行时系统将扮演更智能的角色，弥合高级可编程性与硬件特定性能之间的差距。

统一内存架构通过提供单一、连贯的内存地址空间，有效解决了传统内存架构中的数据传输瓶颈和编程复杂性。按需分页迁移和物理共享等实现机制，极大地简化了开发并提升了性能。然而，其性能优化仍需依赖运行时和编译器内部的智能，通过主动管理数据移动来隐藏潜在开销，推动内存管理向更智能化、透明化的方向发展。

高效互联通信是异构系统发挥潜力的生命线，确保数据在不同处理器之间快速、流畅交换。PCIe、NVLink、CXL和InfiniBand等关键互联技术，从通用总线向专业化、缓存一致性互联演进，这对于释放异构系统的全部潜力至关重要。CXL和UCIe等开放标准的出现，预示着未来计算将走向更模块化、可组合的集成模式。

多架构编程模型旨在降低异构硬件带来的开发复杂性。OpenCL、CUDA、SYCL和oneAPI等模型通过提供统一接口和单源编程能力，抽象了底层硬件细节。异构硬件架构的激增导致了“软件危机”，促使编程模型向更高层抽象发展，并更加依赖智能编译器和运行时系统进行优化。

最终，软硬件协同优化是异构计算实现极致性能和能效的关键。硬件设计需考虑软件需求，提供易用接口；软件则需“硬件感知”，通过智能编译器和运行时系统充分利用硬件特性。这种双向适应和共同进化，是异构计算能否广泛普及并发挥其全部潜力的决定性因素。

综上所述，异构融合计算技术是一个多层面、深度耦合的复杂系统工程。其未来发展将持续围绕处理器专业化、智能调度、统一内存、高速互联、抽象编程和软硬件协同优化这六个核心领域展开，共同推动计算能力迈向新的高度。

引用的著作

(PDF) The Future of Heterogeneous Computing: Integrating CPUs ..., 访问时间为 六月 2, 2025， https://www.researchgate.net/publication/389044641\_The\_Future\_of\_Heterogeneous\_Computing\_Integrating\_CPUs\_GPUs\_and\_FPGAs\_for\_High-Performance\_Applications

Heterogeneous System Architecture - Wikipedia, 访问时间为 六月 2, 2025， https://en.wikipedia.org/wiki/Heterogeneous\_System\_Architecture

快手携手英特尔通过LaoFe NDP 架构，加速异构计算提供多元算力, 访问时间为 六月 2, 2025， https://www.intel.cn/content/www/cn/zh/customer-spotlight/cases/heterogeneous-computing-power-laofe-ndp-architec.html

Heterogeneous computing - Wikipedia, 访问时间为 六月 2, 2025， https://en.wikipedia.org/wiki/Heterogeneous\_computing

Heterogeneous Computing: An Architecture and a Technique - Qualcomm, 访问时间为 六月 2, 2025， https://www.qualcomm.com/developer/blog/2017/03/heterogeneous-computing-architecture-and-technique

算力新生态，透视异构计算的机会和挑战 - 英特尔, 访问时间为 六月 2, 2025， https://www.intel.cn/content/www/cn/zh/cloud-computing/opportunities-challenges-heterogeneous-computing.html

CN107273331A - 一种基于cpu+gpu+fpga架构的异构计算系统和方法 ..., 访问时间为 六月 2, 2025， https://patents.google.com/patent/CN107273331A/zh

A Brief Introduction to CPU, GPU, ASIC, and FPGA | FS Community, 访问时间为 六月 2, 2025， https://community.fs.com/article/a-brief-introduction-to-cpu-gpu-asic-and-fpga.html

What is Heterogeneous Computing? - Supermicro, 访问时间为 六月 2, 2025， https://www.supermicro.com/en/glossary/heterogeneous-computing

FPGA, CPU, GPU, ASIC区别，FPGA为何这么牛- 设计与应用- 半导体芯科技, 访问时间为 六月 2, 2025， https://www.siscmag.com/news/show-4866.html

异构算力统一标识与服务白皮书, 访问时间为 六月 2, 2025， http://221.179.172.81/images/20210324/13331616553850222.pdf

ijrpr.com, 访问时间为 六月 2, 2025， https://ijrpr.com/uploads/V6ISSUE5/IJRPR46271.pdf

Research on computing task scheduling method for distributed heterogeneous parallel systems - PMC - PubMed Central, 访问时间为 六月 2, 2025， https://pmc.ncbi.nlm.nih.gov/articles/PMC11910613/

A Novel Task Scheduling Algorithm for Heterogeneous Computing - CiteSeerX, 访问时间为 六月 2, 2025， https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=f8f7375938e916c9fbae89d6ee133c8cd408e78d

disco.ethz.ch, 访问时间为 六月 2, 2025， https://disco.ethz.ch/courses/fs14/seminar/paper/Jochen/4.pdf

Performance Effective Task Scheduling Algorithm for Heterogeneous Computing System, 访问时间为 六月 2, 2025， https://par.cse.nsysu.edu.tw/resource/paper/2020/200908/Performance%20Effective%20Task%20Scheduling%20Algorithm%20for%20Heterogeneous.pdf

异构计算系统中弹性节能调度策略研究 - 计算机学报, 访问时间为 六月 2, 2025， http://cjc.ict.ac.cn/quanwenjiansuo/2012-6/zxm.pdf

面向CPU-GPU异构系统的数据分析负载均衡策略 - 计算机工程与科学, 访问时间为 六月 2, 2025， http://joces.nudt.edu.cn/CN/Y2019/V41/I03/417

(PDF) Performance-effective and low-complexity task scheduling for heterogeneous computing (2002) | Haluk Rahmi Topcuoglu | 3499 Citations - SciSpace, 访问时间为 六月 2, 2025， https://scispace.com/papers/performance-effective-and-low-complexity-task-scheduling-for-51tejcgo0m

Static task mapping for heterogeneous systems based on series-parallel decompositions - arXiv, 访问时间为 六月 2, 2025， https://arxiv.org/pdf/2502.19745

基于Spark的异构集群调度策略研究 - 汉斯出版社, 访问时间为 六月 2, 2025， https://www.hanspub.org/journal/paperinformation?paperid=19027

zwang4.github.io, 访问时间为 六月 2, 2025， https://zwang4.github.io/publications/hipc14.pdf

OpenCL: A Parallel Programming Standard for Heterogeneous Computing Systems - PMC, 访问时间为 六月 2, 2025， https://pmc.ncbi.nlm.nih.gov/articles/PMC2964860/

Introduction to OpenCL Programming (C/C++) - UL HPC Tutorials, 访问时间为 六月 2, 2025， https://ulhpc-tutorials.readthedocs.io/en/latest/gpu/opencl/

Multi-Task Scheduling Framework for OpenCL Programs on CPUs- GPUs Heterogeneous Platforms - SPIE Digital Library, 访问时间为 六月 2, 2025， https://www.spiedigitallibrary.org/conference-proceedings-of-spie/12167/121671Y/Multi-task-scheduling-framework-for-OpenCL-programs-on-CPUs-GPUs/10.1117/12.2628558.pdf

An Easy Introduction to CUDA C and C++ | NVIDIA Technical Blog, 访问时间为 六月 2, 2025， https://developer.nvidia.com/blog/easy-introduction-cuda-c-and-c/

Heterogeneous CUDA-programming model | Download Scientific Diagram - ResearchGate, 访问时间为 六月 2, 2025， https://www.researchgate.net/figure/Heterogeneous-CUDA-programming-model\_fig1\_301632677

How does CUDA thread scheduling work? - Massed Compute, 访问时间为 六月 2, 2025， https://massedcompute.com/faq-answers/?question=How%20does%20CUDA%20thread%20scheduling%20work?

What is the role of NVIDIA\'s CUDA architecture in supporting heterogeneous computing for large language models?, 访问时间为 六月 2, 2025， https://massedcompute.com/faq-answers/?question=What%20is%20the%20role%20of%20NVIDIA%27s%20CUDA%20architecture%20in%20supporting%20heterogeneous%20computing%20for%20large%20language%20models?

proceedings.nips.cc, 访问时间为 六月 2, 2025， https://proceedings.nips.cc/paper\_files/paper/2020/file/5f0ad4db43d8723d18169b2e4817a160-Paper.pdf

dian-lun-lin.github.io, 访问时间为 六月 2, 2025， https://dian-lun-lin.github.io/publications/2021-amte.pdf

Task Scheduler — oneAPI Specification 1.1-rev-1 documentation, 访问时间为 六月 2, 2025， https://oneapi-spec.uxlfoundation.org/specifications/oneapi/v1.1-rev-1/elements/onetbb/source/task\_scheduler

Task Scheduler — oneAPI Specification 1.3-rev-1 documentation, 访问时间为 六月 2, 2025， https://oneapi-spec.uxlfoundation.org/specifications/oneapi/v1.3-rev-1/elements/onetbb/source/task\_scheduler

Straightforward Heterogeneous Computing with the oneAPI Coexecutor Runtime - MDPI, 访问时间为 六月 2, 2025， https://www.mdpi.com/2079-9292/10/19/2386

A Decade of Heterogeneous C++ Compute Acceleration with SYCL - Khronos Blog, 访问时间为 六月 2, 2025， https://www.khronos.org/blog/a-decade-of-heterogeneous-c-compute-acceleration-with-sycl

SYCL - Wikipedia, 访问时间为 六月 2, 2025， https://en.wikipedia.org/wiki/SYCL

oneAPI Programming Model, 访问时间为 六月 2, 2025， https://oneapi.io/

oneAPI heterogeneous computing programming model PRC 2023, 访问时间为 六月 2, 2025， https://oneapi.io/event-sessions/oneapi-heterogeneous-computing-programming-model-prc-2023/

Understanding Unified Memory in Macs - Case Monkey, 访问时间为 六月 2, 2025， https://www.casemonkey.co.uk/blogs/blog/understanding-unified-memory-in-macs

UNIFIED MEMOR Y ARCHITECTURE - SGI Depot, 访问时间为 六月 2, 2025， http://www.sgidepot.co.uk/o2/1352.pdf

CUDA编程（八）统一内存 - 昇腾社区, 访问时间为 六月 2, 2025， https://www.hiascend.com/forum/thread-0229107180019590120-1-1.html

Unified Shared Memory | Intel Software - YouTube, 访问时间为 六月 2, 2025， https://www.youtube.com/watch?v=u1AR-AJSqlQ

附录N - CUDA 的统一内存- NVIDIA 技术博客, 访问时间为 六月 2, 2025， https://developer.nvidia.com/zh-cn/blog/cuda-unified-memory-introduction-cn/

What Is a Memory Bottleneck: Definition, Causes & Fixes - Sematext, 访问时间为 六月 2, 2025， https://sematext.com/glossary/memory-bottleneck/

What are the benefits of using NVIDIA's CUDA's unified memory model for large language models? - Massed Compute, 访问时间为 六月 2, 2025， https://massedcompute.com/faq-answers/?question=What%20are%20the%20benefits%20of%20using%20NVIDIA's%20CUDA's%20unified%20memory%20model%20for%20large%20language%20models?

What is Unified Memory on Mac and How Does It Work? - EMB Global, 访问时间为 六月 2, 2025， https://blog.emb.global/unified-memory-on-mac/

Unified memory management — HIP 6.3.42133 Documentation, 访问时间为 六月 2, 2025， https://rocm.docs.amd.com/projects/HIP/en/docs-6.3.1/how-to/hip\_runtime\_api/memory\_management/unified\_memory.html

Intel Unified Memory Framework 0.12.0 documentation, 访问时间为 六月 2, 2025， https://oneapi-src.github.io/unified-memory-framework/

How to Make Unified GPU Memory/Storage Architectures Truly Usable for AI? - Jian Huang, 访问时间为 六月 2, 2025， https://jianh.web.engr.illinois.edu/g10.html

Role of Interconnects in GenAI - Express Computer, 访问时间为 六月 2, 2025， https://www.expresscomputer.in/guest-blogs/role-of-interconnects-in-genai/122954/

What is Accelerated Computing? A Comprehensive Guide to the Future of Processing, 访问时间为 六月 2, 2025， https://www.ufispace.com/company/blog/7dbf0467-a09f-4703-8951-1f0411eb8628

CXL – GAMECHANGER FOR THE DATA CENTER - Dell Learning, 访问时间为 六月 2, 2025， https://learning.dell.com/content/dam/dell-emc/documents/en-us/2023KS\_Jaiswal-CXL\_Gamechanger\_for\_the\_Data\_Center.pdf

(PDF) The Hitchhiker's Guide to Programming and Optimizing CXL-Based Heterogeneous Systems - ResearchGate, 访问时间为 六月 2, 2025， https://www.researchgate.net/publication/385559955\_The\_Hitchhiker's\_Guide\_to\_Programming\_and\_Optimizing\_CXL-Based\_Heterogeneous\_Systems

Astera Labs Expands Collaboration with NVIDIA to Advance NVLink Fusion Ecosystem, 访问时间为 六月 2, 2025， https://www.asteralabs.com/news/astera-labs-expands-collaboration-with-nvidia-to-advance-nvlink-fusion-ecosystem/

Chiplets and Heterogeneous Integration: The Future of Semiconductor Design - ELE Times, 访问时间为 六月 2, 2025， https://www.eletimes.com/chiplets-and-heterogeneous-integration-the-future-of-semiconductor-design

3D Interconnect Architectures for Heterogeneous Technologies: Modeling and Optimization - Amazon.com, 访问时间为 六月 2, 2025， https://www.amazon.com/Interconnect-Architectures-Heterogeneous-Technologies-Optimization/dp/3030982319

异构计算的网，铺成智能时代的路 - 英特尔, 访问时间为 六月 2, 2025， https://www.intel.cn/content/www/cn/zh/cloud-computing/net-of-heterogeneous-computing-way-to-intelligence.html

模型原生操作系统：机遇、挑战与展望 - 安全内参, 访问时间为 六月 2, 2025， https://www.secrss.com/articles/76659

Hercules: A Compiler for Productive Programming of Heterogeneous Systems - arXiv, 访问时间为 六月 2, 2025， https://arxiv.org/html/2503.10855v1

Heterogeneous Computing - Communications of the ACM, 访问时间为 六月 2, 2025， https://cacm.acm.org/practice/heterogeneous-computing/

CUDA Zone - Library of Resources | NVIDIA Developer, 访问时间为 六月 2, 2025， https://developer.nvidia.com/cuda-zone

CUDA Toolkit - Free Tools and Training | NVIDIA Developer, 访问时间为 六月 2, 2025， https://developer.nvidia.com/cuda-toolkit

Understanding Portability of Automotive Workload: A Case Study with the Points-to-Image Kernel in SYCL on Heterogeneous Computing Platforms - OSTI.GOV, 访问时间为 六月 2, 2025， https://www.osti.gov/servlets/purl/1996701

Heterogeneous Programming Using oneAPI - Intel, 访问时间为 六月 2, 2025， https://www.intel.com/content/dam/develop/external/us/en/documents/parallel-universe-issue-39-1.pdf

Unified Programming Models for Heterogeneous High-Performance Computers - JCST, 访问时间为 六月 2, 2025， https://jcst.ict.ac.cn/EN/10.1007/s11390-023-2888-4

Enabling Heterogeneous Computing for Software Developers - eScholarship.org, 访问时间为 六月 2, 2025， https://escholarship.org/uc/item/4nw121pn

A compiler and runtime for heterogeneous computing | Request PDF - ResearchGate, 访问时间为 六月 2, 2025， https://www.researchgate.net/publication/261238296\_A\_compiler\_and\_runtime\_for\_heterogeneous\_computing

面向GoldenX 软硬协同优化的异构加速列式存储引擎研究 - 计算机学报, 访问时间为 六月 2, 2025， http://cjc.ict.ac.cn/online/bfpub/tyf-2021322110537.pdf

What is Heterogeneous Compute? - Arm, 访问时间为 六月 2, 2025， https://www.arm.com/glossary/heterogeneous-compute

Targeted Intel oneAPI DPC++ Compiler Optimization Rules Out 2k+ SPEC CPU Submissions - Reddit, 访问时间为 六月 2, 2025， https://www.reddit.com/r/intel/comments/1asa7q8/targeted\_intel\_oneapi\_dpc\_compiler\_optimization/