OSDI 2022 | 速来围观！微软亚洲研究院计算机系统领域最新论文！

2022-07-12 | 作者：微软亚洲研究院

编者按：OSDI（Operating Systems Design and Implementation）是计算机系统领域最顶级的学术会议之一，汇集了全球计算机科学家们对于计算机系统的前瞻性思考。第16届 OSDI 于2022年7月11日至13日召开，本次会议共有253篇论文投稿，接收49篇，接收率为19.4%。本文中，我们将分享微软亚洲研究院被 OSDI 2022收录的3篇论文，希望可以帮助大家了解计算机系统领域的前沿趋势。欢迎感兴趣的读者阅读论文原文。

Roller: **快速高效的深度学习算子编译器**

ROLLER: Fast and Efficient Tensor Compilation for Deep Learning

Hongyu Zhu[†◇*] Ruofan Wu[‡◇*] Yijia Diao[$◇*] Shanbin Ke[¶◇*] Haoyu Li[§◇*] Chen Zhang[£◇*]
Jilong Xue[◇] Lingxiao Ma[◇] Yuqing Xia[◇] Wei Cui[◇] Fan Yang[◇] Mao Yang[◇]
Lidong Zhou[◇] Asaf Cidon[§] Gennady Pekhimenko[†]
[†]*University of Toronto* [‡]*Renmin University of China* [$]*Shanghai Jiao Tong University*
[¶]*UCSD* [§]*Columbia University* [£]*Tsinghua University* [◇]*Microsoft Research*

论文链接：https://www.usenix.org/conference/osdi22/presentation/zhu
代码地址：https://github.com/microsoft/nnfusion/

随着模型规模的不断增长，深度学习对算力的需求也与日俱增。当前，深度学习硬件加速器（如 GPU、TPU 等）主要依赖算子加速库来支持深度学习应用。然而，随着不断出现的新模型和新硬件类型，业界对快速、高效地开发新算子有了更高的要求。算子编译器（Tensor Compiler）作为一种新途径，提供了将算子自动编译成对应加速器内核代码的能力。

目前，深度学习模型的算子通常被实现成为多重循环嵌套的计算。为了实现对这种算子程序性能的进一步提升，研究员们通常需要对已实现的多重循环计算进行循环展开、合并、分块、缓存使用、改变并行度等调整。这实质上是一个组合优化问题，搜索空间巨大，所以主流的算子编译器往往不得不借助机器学习的方法在这个巨大的搜索空间中找出较优的程序实现。而这种搜索过程往往需要数千甚至上万步的探索，导致编译一个模型经常需要耗费数小时甚至数天的时间。这种问题在新型的硬件和计算量巨大的算子上特别严重。因此，目前主流的基于机器学习的算子程序优化方法极大地影响了深度学习在实际场景中的部署效率。随着近年来深度学习模型规模的快速增长，预计这种情况将进一步恶化。

微软亚洲研究院联合多伦多大学等多所高校在对深度学习计算中用到的大量算子进行编译、优化和性能分析后，发现了一个有意思的现象：尽管每个算子的优化选项成千上万，但能达到最大性能的程序，其配置往往与硬件参数匹配，从而能够更充分地利用硬件资源。基于这样的观察，研究员们提出了一个新的算子编译器 Roller。为了更好地匹配硬件参数，Roller 将算子的计算过程建模成基于数据块（tile）的“数据处理流水线”，即将不同大小的数据块从多级内存结构中搬运到计算核心处理并逐级写回。为了使整个流水线的吞吐最大化，Roller 要求每一级中数据块的设置都必须充分匹配硬件的参数设置，比如内存的访问宽度、计算核的并行度等等。这样的约束不仅保证了计算程序在整个流水线中的每一级都充分发挥了硬件的最大性能，同时也简化了对算子程序的性能建模，大大减小了搜索空间，从而避免了在实际硬件加速器上耗时地进行大量尝试和性能测量。因此，这样一种“白盒”的内核程序构建方法从本质上可以大大缩减算子编译时间。

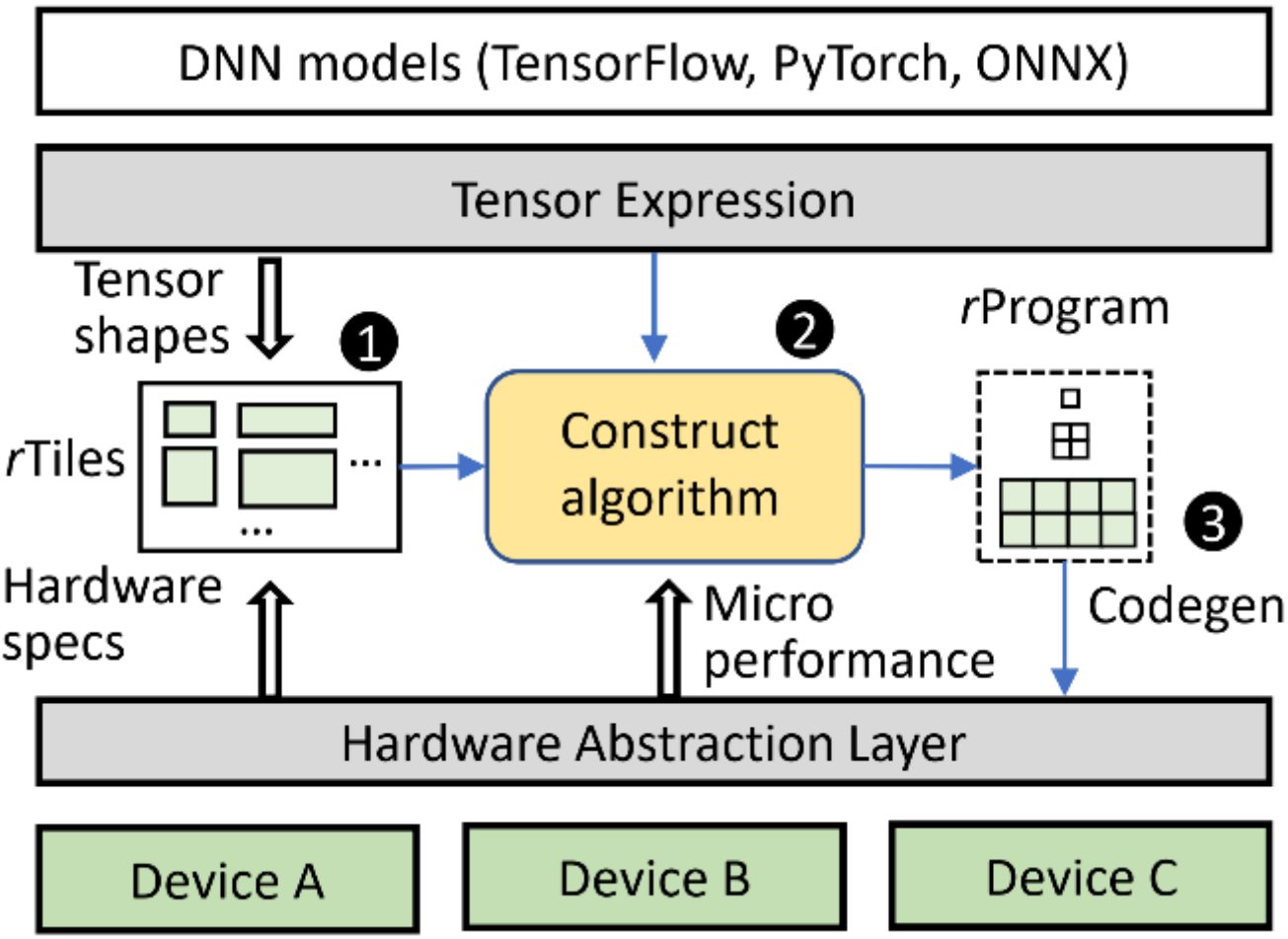

图1：Roller 编译器的架构

更重要的是，Roller 的整个设计是构建在一套通用硬件抽象上的，它可以很好地适配到多种主流加速器设备上。通过在 NVIDIA GPU、AMD GPU 和 Graphcore IPU 上进行实验评测，研究员们发现相比于当前算子编译器（如 TVM 和 Ansor）和算子库（如 cuBLAS、cuDNN 等），Roller 的编译方法可以生成性能相近，甚至更好的内核代码。例如，研究员们在对不同类型的模型中常用到的一百多个算子进行编译后，Roller 产生的内核代码中分别有59.7%和73.1%的代码要优于 NVIDIA 和 AMD 的算子库，有54.6%和58.8%的代码优于 TVM 或 Ansor 在这两种 GPU 上生成程序的性能。值得注意的是，Roller 的算子编译时间也从小时级别降低到了秒级别，编译时间减少了三个数量级！

研究员们认为 Roller 这样的系统可以给硬件加速器提供更高效快速的软件生态支持。对于缺少成熟算子库的新型硬件厂商，Roller 还提供了一个可以快速生成算子库的机会，从而获得以较快速度赶上领先硬件厂商的机会。Roller 的“白盒”编译方法也给针对特定硬件快速进行深度学习模型优化和部署带来了新的机遇。微软亚洲研究院正在基于 Roller 进一步完善深度学习模型编译栈，并展开更多关于模型优化、场景部署和新型硬件支持的研究。

RESIN: **一种处理云服务中内存泄漏的整体方案**

RESIN: A Holistic Service for Dealing with Memory Leaks in Production Cloud Infrastructure

Chang Lou[1], Cong Chen[2], Peng Huang[1], Yingnong Dang[2], Si Qin[3], Xinsheng Yang[4], Xukun Li[2], Qingwei Lin[3], Murali Chintalapati[2]

[1]Johns Hopkins University [2]Microsoft Azure [3]Microsoft Research [4]Meta

论文链接：https://www.usenix.org/conference/osdi22/presentation/lou-chang-resin

云计算的基础设施包含许多复杂的软件组件，因此会遇到内存泄漏的问题。在云服务中，一旦某个进程发生内存泄露，就会使得其部署的大量机器内存不足以及无辜进程被杀，最终导致大量用户的虚拟机性能下降、崩溃及重启，严重影响用户的体验并可能给用户造成重大经济损失。

然而，在云系统中，内存泄漏问题很难处理，现有解决方案或是准确性不高，或是会产生很高的开销。其原因，一方面是由于许多内存泄漏在极少数情况下才会被缓慢触发，因此在开发部署过程中很容易躲过测试和检测；另一方面，在检测到内存泄漏后，相关问题很难离线复现，这会使得开发人员很难找到泄漏的根本原因。为了解决上述问题，微软亚洲研究院与微软 Azure 团队以及霍普金斯大学的科研人员，一起提出了一种处理云服务中内存泄漏的整体方案 RESIN。

RESIN 采用集中式方法，无需访问组件的源代码，也不需要大量的检测或重新编译。RESIN 对每个主机都使用了一个监控代理（Monitor Agent），该代理通过利用底层操作系统功能来收集内存使用数据，因此它支持包括内核（Kernel）在内的所有组件。收集好的数据随后会被加载到远程服务用于进一步的数据分析，从而最大限度地减少主机的开销。通过聚合来自不同主机的数据，RESIN 可以运行更复杂的分析来捕获复杂的泄漏。对于内存泄漏问题，RESIN 进行了多级分解和处理。它会首先执行轻量级泄漏检测，并在需要确认和诊断时触发更深入的动态检查。这种分治法（Divide-and-Conquer）使 RESIN 能够同时实现低开销、高精度和可扩展性。

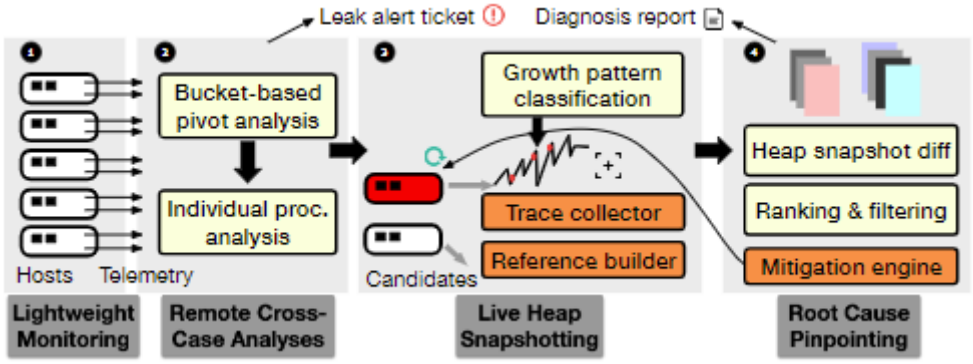

图2：RESIN 系统的工作流程

目前，RESIN 已经在微软 Azure 中运行3年，具有高准确性与低开销等优势。由于内存不足导致的虚拟机重启有效地减少了41倍。

SparTA: 基于张量稀疏属性的深度学习模型稀疏化编译框架

SparTA: Deep-Learning Model Sparsity via Tensor-with-Sparsity-Attribute

Ningxin Zheng[1*], Bin Lin[1,2*], Quanlu Zhang[1], Lingxiao Ma[1], Yuqing Yang[1], Fan Yang[1], Yang Wang[1], Mao Yang[1], Lidong Zhou[1]

[1]Microsoft Research, [2]Tsinghua University

论文链接：https://www.usenix.org/conference/osdi22/presentation/zheng-ningxin
项目代码地址：https://github.com/microsoft/SparTA.git
论文结果复现代码地址：https://github.com/microsoft/nni/tree/sparta_artifact/sparta

随着深度学习的快速发展，深度学习模型的规模呈现出指数级增长的态势。单个模型具有高达万亿级别的参数量，这远远超出了硬件加速器运算能力的增长速度。同时，端侧设备由于其运算能力和功耗的限制，对深度学习模型大小和推理延迟有近乎严苛的要求。因此，探索深度学习模型中的稀疏性以及对稀疏模型做有效的加速成为深度学习模型发展和落地的关键因素。然而，目前稀疏模型在系统支持方面还存在诸多不足，阻碍了模型在稀疏化上的探索。

这些不足具体表现为（i）由于针对特殊的模型稀疏模式构建高效的代码非常困难，通常需要系统程序员的深度参与，深度学习研究员在模型稀疏化的研究过程中通常使用代理指标（Proxy metrics，如 FLOPs、位宽）来估算加速效果。可是代理指标并不能准确反映模型稀疏化带来的加速效果，其预测的效果有时甚至和实际情况相去甚远；（ii）当前稀疏优化的大多工作都只集中在单个算子，忽视了一个稀疏化算子在整个深度学习模型中可能产生的连带影响，可将稀疏化传导到模型中的其他算子；（iii）当前针对某个具体模型的稀疏化优化方案很难被重用，难以和其它相关技术组合起来迁移到其他深度学习模型上去，例如，在被剪枝的算子和被量化的算子上做的优化技术很难直接迁移到一个既被剪枝又被量化的算子上。

为了解决模型稀疏化过程中遇到的上述挑战，微软亚洲研究院提出了 SparTA——一个专门优化稀疏深度学习模型的端到端编译框架。SparTA 将深度学习模型中张量的稀疏属性（由剪枝和量化产生）作为整个编译框架中的核心抽象 TeSA (Tensor with Sparsity Attribute)，并围绕 TeSA 构建出面向稀疏模型的全套编译优化技术。

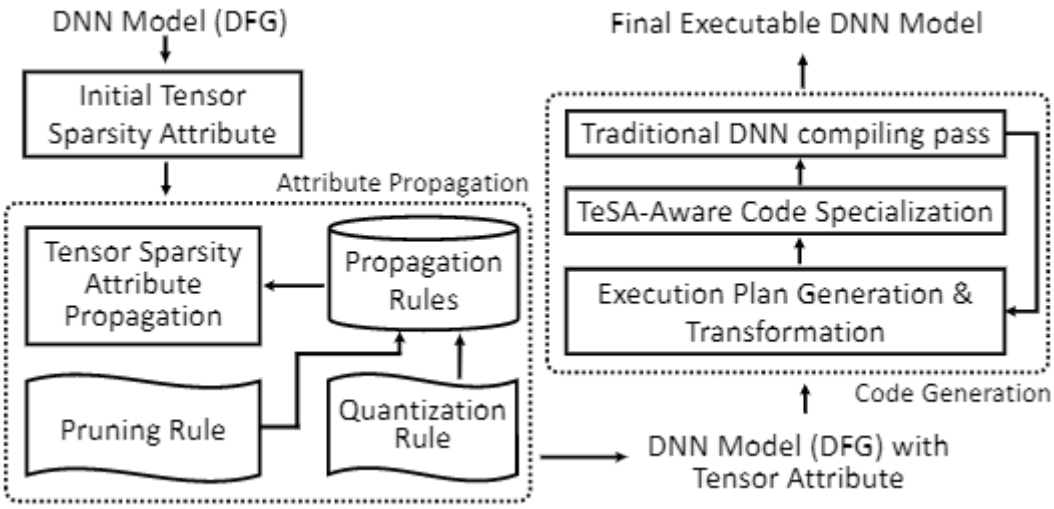

图3：SparTA 的系统架构

在使用 SparTA 时，用户先使用 TeSA 来标注深度学习模型中某些张量的稀疏样式（Sparsity pattern），然后通过 SparTA 提出的三个核心技术来对模型进行端到端地优化：一是张量的稀疏属性在整个数据流图中的传播，由 Tensor Algebra 和 Tensor Scrambling 技术自动完成；二是对稀疏算子进行优化变换（Transformation），变换为更易于加速、计算效率更高的稀疏样式，这种变换使得不同的优化技术可以被有机地结合起来；三是做针对稀疏张量的代码特化（Code Specialization），将稀疏样式硬编码到算子的代码中，删除死代码（Dead Code），并且针对给定加速器特化使用其专有硬件（如 sparse tensor core）。

通过全面的测试，SparTA 展示出了相比于已有工具高达平均8.4倍的加速效果。SparTA 不但可以对稠密模型稀疏化以后进行优化，而且还能用来加速一开始就采用特定稀疏样式设计的大型预训练模型。在这方面，SparTA 已经被用来优化微软亚洲研究院研发的 NUWA 预训练视频生成模型。SparTA 对 NUWA 模型中提出的 3DNA 稀疏化注意力机制达到了2倍以上的加速。微软亚洲研究院目前正在对 SparTA 进行代码重整和优化，提高易用性，并将尽快开源，以促进深度学习模型稀疏化的研究和实用化。

标签

系统与网络　顶会聚焦