编译方向探索

1 编译的本质与核心任务

编译的本质在于将程序中蕴含的计算过程转换为一种更适合分析、优化和执行的表达形式。无论是传统编译器还是深度学习编译器，其核心目标都是构建从“源代码描述”到“高效执行计划”的映射。因此，理解编译的根本任务，需要从对计算的重新表达与形式化建模两方面展开。

首先，编译是对计算的重新解释。源程序以人类友好的方式描述计算逻辑，而编译器需要从语言细节中抽取语义的本质，将其转换为在结构上更明确、在语义上更纯净的表示形式，使得程序行为能够被系统化地理解、分解与优化。其次，编译离不开对计算的形式化建模。通过构建抽象语法树（AST）、中间表示（IR）、控制流图（CFG）、静态单赋值形式（SSA）等形式结构，编译器得以从数学上严格刻画程序的执行语义，使控制流、数据流与优化机会清晰可见，并为后续的性能提升奠定基础。

编译工作的根本在于通过“语义重写”与“形式化建模”这两种互补方式，将高层计算意图转化为可优化、可验证、可执行的形式。正是在这一总框架下，现代编译技术得以不断扩展，从传统程序编译走向深度学习系统编译，实现更高效、更智能的计算表达与执行。

2 传统编译器与深度学习编译器

2.1 传统编译

传统编译器（如 GCC、LLVM）是现代软件生态的基础设施，其目标是将通用编程语言（如 C/C++、Rust、Go）编译为跨平台、高效且正确的机器代码。传统编译器普遍采用“语言前端—通用 IR—后端代码生成”的分层架构，其中 LLVM IR 的出现尤为重要。LLVM IR 作为高度抽象、稳定且可优化的中间表示，使得编程语言开发者无需为多种硬件架构独立实现后端，只需将语言语义映射到 LLVM IR，随后由 LLVM 的优化器和代码生成器负责将 IR 转换为针对不同 CPU 架构（如 x86、ARM、RISC-V）的高性能可执行代码。这种分层设计显著降低了编译器开发复杂度，并推动了现代语言生态的繁荣。

2.2 深度学习编译器

随着深度学习模型规模急剧增长以及硬件迅速异构化，传统编译技术已无法满足 AI 推理与训练的性能需求。深度学习框架（如 PyTorch、TensorFlow、ONNX）通常以计算图形式表达模型，底层则由多样化的硬件平台承载，包括 CPU、GPU、NPU、DSP、TPU、FPGA 等。这些硬件在并行结构、存储体系、指令集以及执行模型上存在巨大差异，使得硬件厂商长期依赖手工编写的高性能算子库（如 cuDNN、MKL、ACL、MIOpen）来维持性能。这种模式本质上类似“无统一 IR 的早期编译时代”，导致重复开发、强耦合和性能不可移植等问题。

深度学习编译器（如 TVM、XLA、TensorRT、AITemplate）正是为了解决这一困境而生。它们通过构建多层次 IR（graph IR、tensor IR、loop IR）来统一表达深度学习模型的结构与执行语义，并通过算子融合、数据布局转换、内存调度、并行化策略等优化手段提升整体性能。与传统编译器不同，深度学习编译器需要面对巨大的张量调度空间，因此普遍采用自动调优（auto-tuning）与机器学习驱动的性能模型（ML cost model）来搜索最优执行策略，从而实现跨硬件平台的性能可移植性。

此外，新一代深度学习编译器引入了诸如 MLIR、TensorIR、Relax 等高级 IR，以支持动态形状、控制流、混合精度、图级优化及硬件特化转换。同时，像 Triton 这样的编程系统提供了更灵活的 GPU kernel 编写方式，使开发者能够以 Python DSL 的形式直接描述高性能张量内核。

3 现存技术瓶颈

传统编译器（如 GCC、LLVM）在过去几十年引领了通用软件系统的发展，但随着现代计算架构和应用需求不断演进，其体系逐渐暴露出若干结构性瓶颈。

随着现代处理器体系结构持续演进，其硬件复杂度呈指数级增长，包括深度乱序流水线、多级缓存体系、复杂的向量化指令扩展以及多核与 NUMA 架构等特性，使得传统编译器长期以来依赖的手写启发式规则和模式匹配策略逐渐难以应对不断提升的优化需求。同时，传统编译器的核心中间表示（如 LLVM IR）主要面向标量和控制流抽象，缺乏对高维张量、数据布局、显式内存层级以及专用计算单元等现代计算特征的表达能力，导致大量高层语义在编译阶段被丢失，阻碍了跨算子优化和硬件特化优化的实施。更为关键的是，随着 GPU、NPU、TPU、DSP、FPGA 等多种加速器的普及，计算平台呈现出前所未有的异构化，其执行模型、并行粒度、存储体系和指令语义迥异，而传统编译器缺乏统一抽象层来覆盖这些硬件差异，使得编译结果既难以复用，也无法实现性能可移植性。在硬件复杂度、抽象能力不足与异构架构爆炸式增长的共同作用下，传统编译体系逐渐暴露出结构性瓶颈，难以满足现代高性能计算尤其是深度学习任务的需求。

随着深度学习模型规模的持续扩大以及 Transformer、LLM 等新型结构的快速演进，深度学习编译器在优化能力与系统抽象上逐渐显露出多方面的瓶颈。首先，尽管深度学习编译器旨在自动生成高性能算子内核，但其优化流程仍高度依赖人工定义的调度空间与规则模板，难以完全摆脱手工调优；与此同时，张量计算所带来的维度组合、内存访问模式和并行策略共同构成了指数级膨胀的调度搜索空间，使得自动调优成本极高且不稳定。其次，近年来大规模模型的动态结构——包括自注意力机制、动态形状、稀疏性以及 KV Cache 等——显著增加了计算图的复杂度，使传统以静态图、固定 shape 和规则算子为假设的编译优化策略难以奏效。此外，深度学习生态长期呈现碎片化特征：不同硬件厂商维护私有 IR、手写 kernel 和独立工具链，缺乏统一抽象层，使得编译器难以在异构平台上实现一致的性能表现。伴随模型创新周期不断缩短，编译器在新算子、新模式和新硬件出现后往往难以及时支持，进一步加剧了编译系统的维护负担。最终，由于缺乏对底层硬件特性（如 Tensor Core、warp 调度、片上 SRAM）的深度刻画，自动生成的内核在主流平台上仍难以达到 cuDNN、cuBLAS、OneDNN 等厂商深度优化库的顶尖性能水平。上述问题共同表明，深度学习编译器仍处于快速发展阶段，其自动化能力、抽象能力与性能可移植性距离理想目标依然存在显著差距。

4 编译技术的未来发展方向

针对深度学习编译器仍面临的手动调优依赖、调度空间爆炸、动态图难优化、生态碎片化以及性能难以匹配厂商手写库等问题，未来的研究可以从以下几个方向推进。首先，如何构建更强大的

自动调优机制仍然至关重要，包括探索基于强化学习、大模型驱动的性能预测器以及跨形状、跨模型的迁移学习式 cost model，以减少人工规则依赖并在大规模搜索空间中实现更高效的最优调度发现。其次，需要设计具备更高抽象能力的新型 IR，以更自然地表达动态形状、稀疏模式、注意力结构和阶段性计算行为，使优化器能够捕获更多全局语义并支持跨算子、跨层次的统一优化；相关工作包括 MLIR 多层次 IR、TVM Relax 以及更通用的 tensor program IR。再次，为解决硬件碎片化问题，研究者需要推动通用硬件抽象层的发展，实现 GPU、NPU、TPU 等加速器之间的语义统一，使编译器能够生成硬件无关的优化策略，并在下沉阶段进行自动特化。此外，异构系统上的深度学习运行时需要进一步智能化与统一化，特别是在支持动态 shape 的即时编译（JIT）、KV Cache 管理、跨 GPU 流调度、多设备通信和 pipeline 并行方面。与此同时，随着 LLM 在代码生成、模型推断以及性能预测方面的能力提升，利用大模型增强编译器的自动化水平将成为重要趋势，例如自动生成优化 pass、自动生成 kernel 模板、自动构建搜索空间甚至“端到端可学习的编译器”。最后，如何在保持编译器可扩展性与工程可维护性的前提下构建面向未来的统一编译基础设施，也是推动行业走向长期标准化与性能可移植性的关键方向。