

Contents

1 Glow	1
1.1 Goals	1
1.2 High Level IR : target independent optimization	1
1.3 Low-level IR	1
1.3.1 An example of IR format function	2

1 Glow

1.1 Goals

Consume a neural network computation graph, optimize it, and code generation for it for a diverse set of backends.

- Glow is a machine learning compiler for heterogeneous hardware.
- Glow lowers the traditional neural network dataflow graph into a two-phase **strongly-typed** IR:
 1. the high-level IR allowing domain-specific optimization.
 - kernel fusion
 2. the low-level instruction-based address-only IR to perform memory-related optimizations
 - instruction scheduling
 - static memory allocation
 - copy elimination

1.2 High Level IR : target independent optimization

- High-level IR is Dataflow node based representation (Tensors and operators).
- The graph is strongly-typed (Each tensor has a known data type).

Glow graph is structured as a module

- Variable, Function, Node are concepts from the implementations.
- 1. Variables: persistent tensors for learnable parameter (global tensors).
- 2. A module contains multiple functions (functions are a set of sequentially executed operators in essence).
 - for training tasks, there will be forward functions, backward functions, optimization functions.
 - Glow functions contain “nodes” that represent the different operations of a neural network.
- A function contains multiple nodes (nodes are operators in essence).
- Nodes inside functions are able to reference variables which are owned by the module.

Node lowering

- breaks the high-level operator nodes into low-level linear algebra operator nodes.
 - It gives me the feeling that these low-level linear algebra operator nodes are very HLO primitives in XLA.
- the new graph may affect instruction scheduling.

1.3 Low-level IR

- Low-level IR is a instruction based representation.
 - Glow use a **self-defined IR**, not directly use LLVM IR.
- One-to-many translation: each high-level node is translated into one or more instructions.
- Memory is added at the low-level IR.
 - In-place memory transformation for elementwise computation.
- The IR is strongly typed. Each operand has known parameter type.
- Device dependent optimization.
 - define hardware specific DMA instruction.
 - * implement a instruction scheduling to hidden memory latency.

1.3.1 An example of IR format function

- A function in IR format has two parts:
 1. declare
 - declare several memory regions that live throughout the lifetime of the program (like global variable in C++)
 - memory region in the declare part is GLOBAL.
 2. program
 - a list of instructions
 - memory region in the program part is LOCAL.
- Memory region is strongly-typed.
- Each operand is annotated with one of the qualifiers: @in (the buffer is read from), @out(the buffer is written to), @inout (the buffer is both read from and written to)
 - copy elimination
 - buffer sharing
 - keep the memory buffer (not deleted) of forward computation, so that they can be reused in backward computation