ONNX Compiler SIG

Compiler SIG Goals

- Shape ONNX specification to make it implementer friendly
  - Unambiguous
  - Lean
  - Documented

- Build shared ONNX compiler infrastructure
  - onnx-mlir
  - shape inference

Compiler SIG is an Active Community

Companies involved

- ByteDance, AMD, ARM, Groq, IBM, Microsoft, NVIDIA

Monthly Compiler SIG meetings

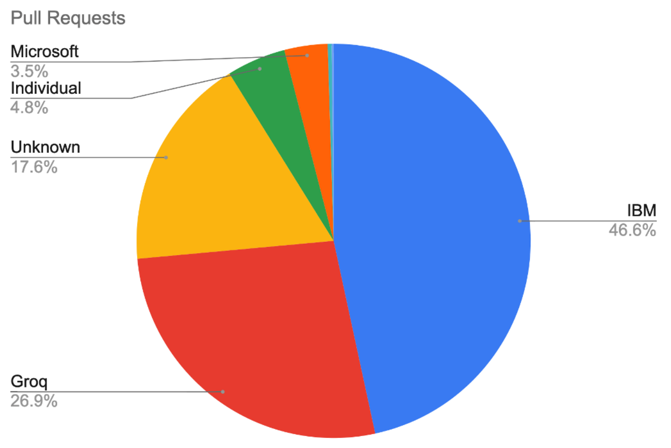
- 1st Tuesday of the month, 8-9pm EST

Weekly ONNX-MLIR meetings

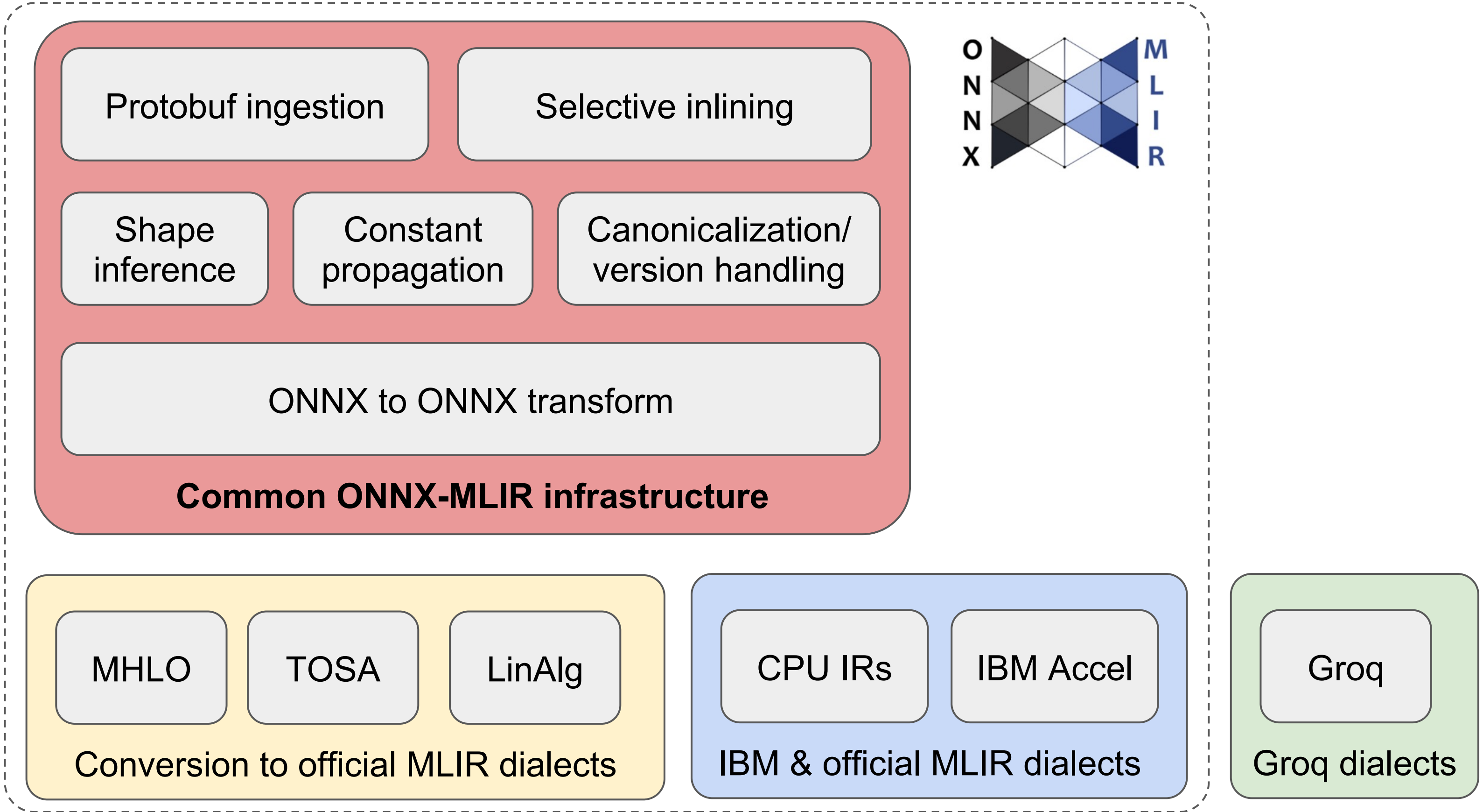
- Tuesday @ Asia & Europe friendly times

Statistics

- 621 PR by 49 developers in the last 12 months



ONNX-MLIR Infrastructure



Protobuf ingestion

Selective inlining

Shape inference

Constant propagation

Canonicalization/ version handling

ONNX to ONNX transform

Common ONNX-MLIR infrastructure

MHLO

TOSA

LinAlg

Conversion to official MLIR dialects

CPU IRs

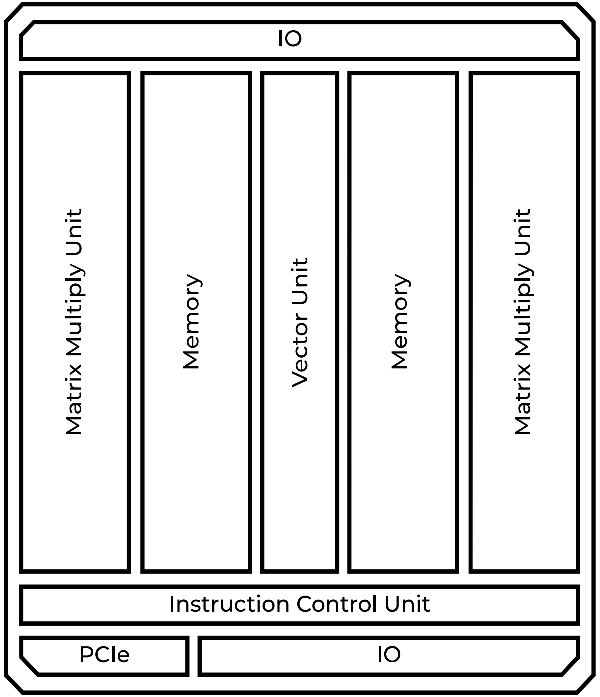
IBM Accel

IBM & official MLIR dialects

Groq

Groq dialects

Groq case study



IO

Matrix Multiply Unit

Memory

Vector Unit

Memory

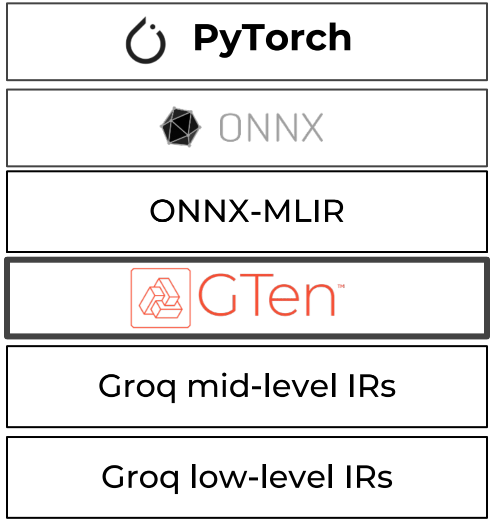
Matrix Multiply Unit

Instruction Control Unit

PCIe

IO

Groq's TSP



PyTorch

ONNX

ONNX-MLIR

GTen™

Groq mid-level IRs

Groq low-level IRs

IBM Case study

Offers for IBM Z servers:

- Single binary with CPU and Accelerator code with no external dependences
- Optimized usage of AI accelerator
- Minimized data movement / reorganization between CPU and AI accelerator

Offers to community:

- CPU code that can run on any LLVM supported platform (Mac/Windows/Linux)
- Template for accelerators that other company may reuse

Aspirations of the Compiler SIG

More proactive role for operations

- Making sure new operators are consistent and unambiguous

Making current ONNX compiler infrastructure more attractive

- Increasing synergy of current infrastructure to better serve users

Reaching out to other Deep Learning compiler platforms