Onnx-mlir: MLIR-based Compiler for ONNX Models The Latest Status

Tung D. Le, Tong Chen, Ettore Tiotto, Haruki Imai, Yasushi Negishi, Kevin O'Brien, Kiyokuni Kawachiya, Alexandre E Eichenberger

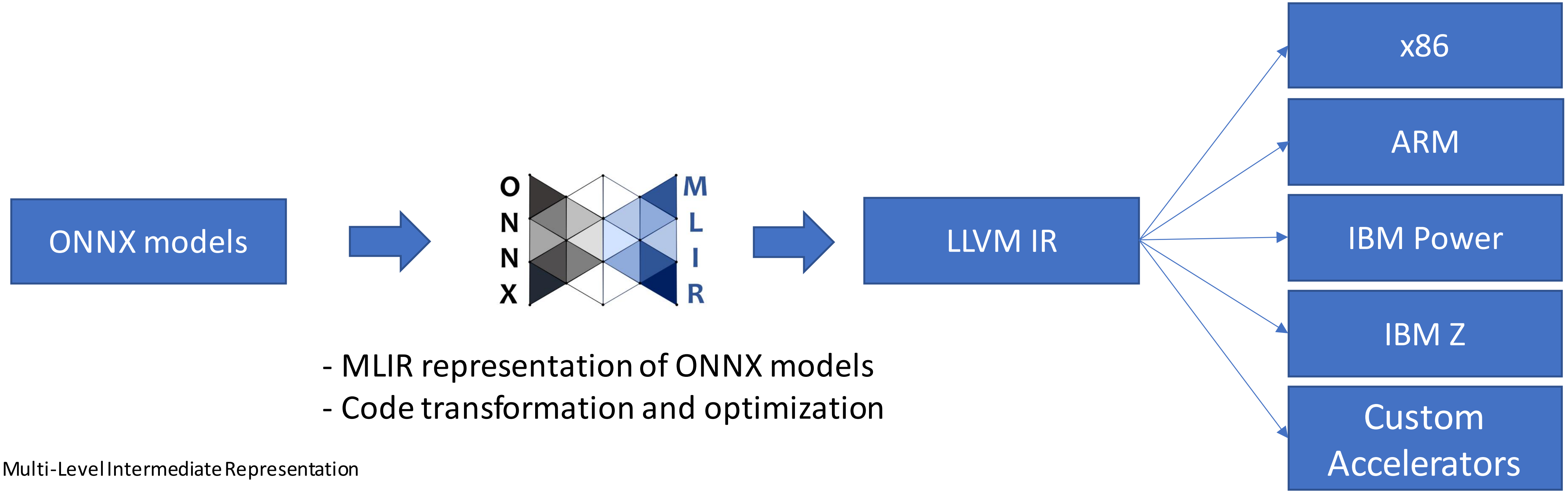
IBM Research

https://github.com/onnx/onnx-mlir

Presenting the work of many people!

What is onnx-mlir?

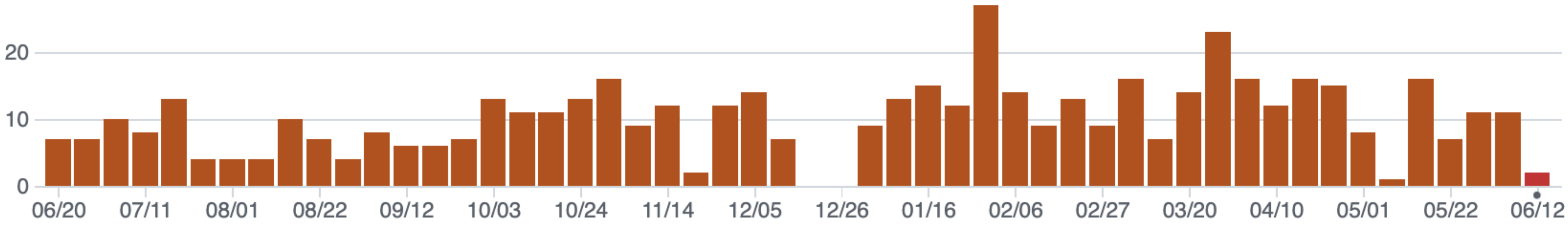
- Compile an ONNX model to an optimized binary using
  - MLIR* to perform high-level optimization transformations
  - LLVM to perform low-level optimizations and code generation



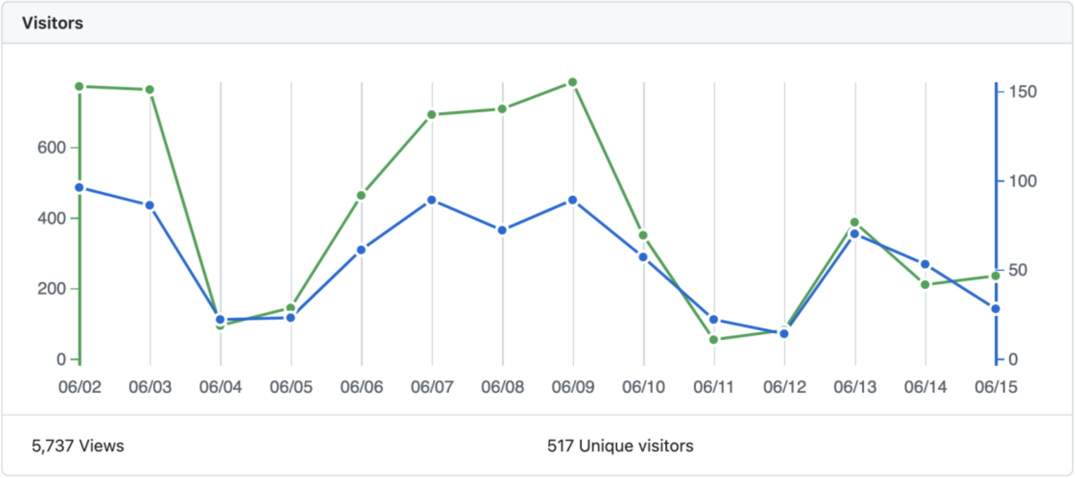
- MLIR representation of ONNX models
- Code transformation and optimization

*MLIR: Multi-Level Intermediate Representation

Young but active project

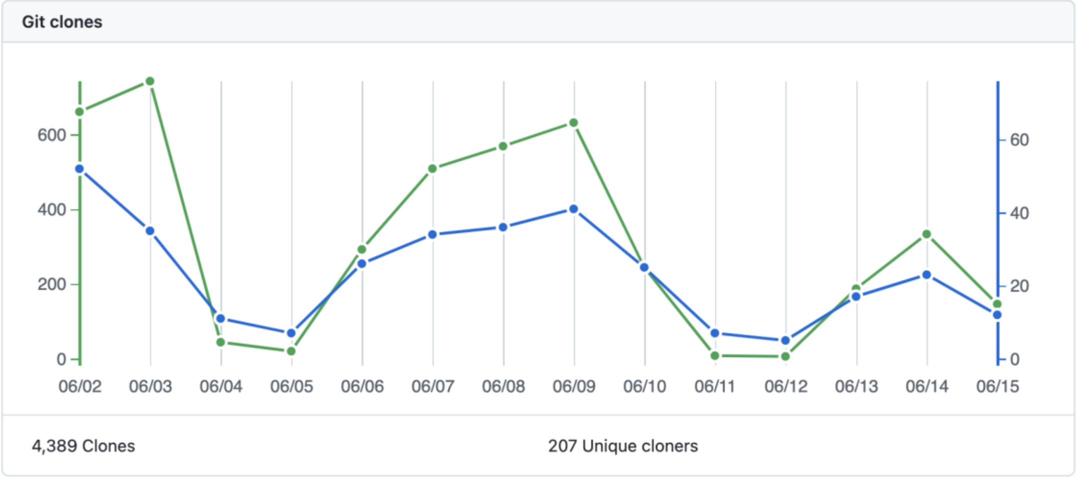


Commits per week over the last year



Visitors

5,737 Views 517 Unique visitors



Git clones

4,389 Clones 207 Unique cloners

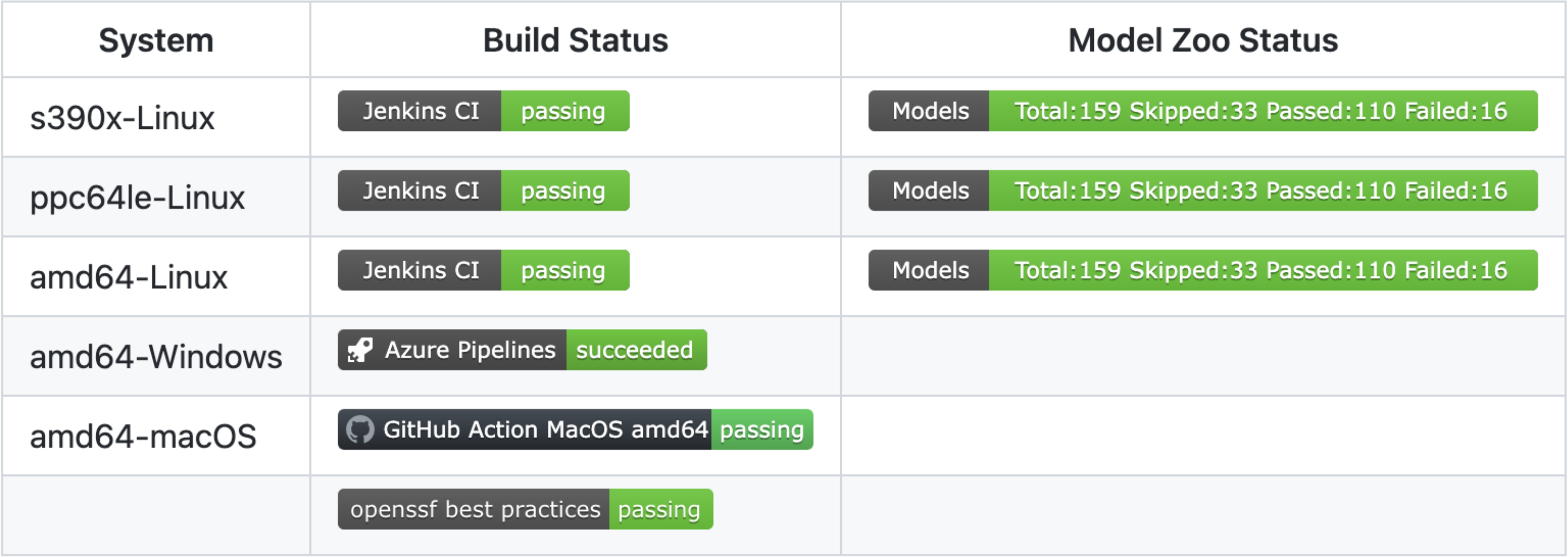
- Contributions and support from IBM, Microsoft, Arm, Facebook, and others

Design goals

- A reference ONNX dialect in MLIR
- Easy to write optimizations for CPU and custom accelerators
  - From high-level (e.g., graph level) to low-level (e.g., instruction level)
- Easy to deploy
  - Stand-alone driver and runtime support in Python/C/C++/Java
  - Integration into other MLIR-based compilers
- Continuously tested
  - Unit tests and ONNX model zoo
  - x86, Power, z/Architecture
  - Windows, Linux, z/OS, macOS
  - Python/C++/Java

Onnx-mlir in practice

- Tested with many models in the ONNX model zoo
- Deployed in IBM Watson Machine Learning for z/OS (WMLz)
- One of the inference engines in BentoML framework



System	Build Status	Model Zoo Status
s390x-Linux	Jenkins CI passing	Models Total:159 Skipped:33 Passed:110 Failed:16
ppc64le-Linux	Jenkins CI passing	Models Total:159 Skipped:33 Passed:110 Failed:16
amd64-Linux	Jenkins CI passing	Models Total:159 Skipped:33 Passed:110 Failed:16
amd64-Windows	Azure Pipelines succeeded	
amd64-macOS	GitHub Action MacOS amd64 passing	
	openssf best practices passing	

Online scoring services in IBM WMLz

BENTOML

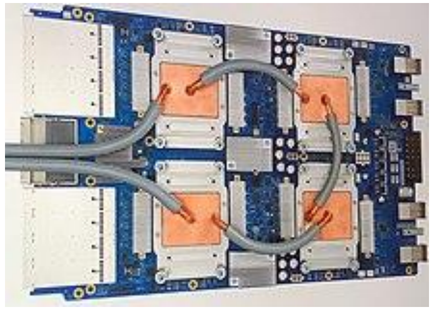
The Unified Model Serving Framework

Recent developments in onnx-mlir

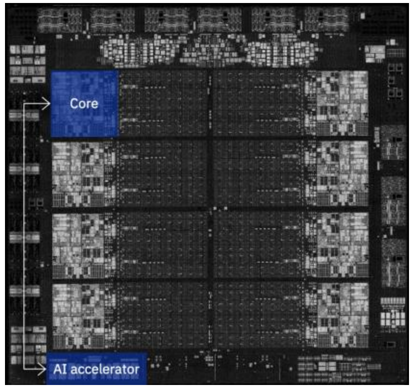
- A framework for supporting custom accelerators
  - Easy to offload ONNX operators to accelerators
  - Custom optimizations for custom accelerators

- We have demonstrated the framework for IBM on-chip low-latency AI accelerator introduced in IBM z16.

NVIDIA GPU

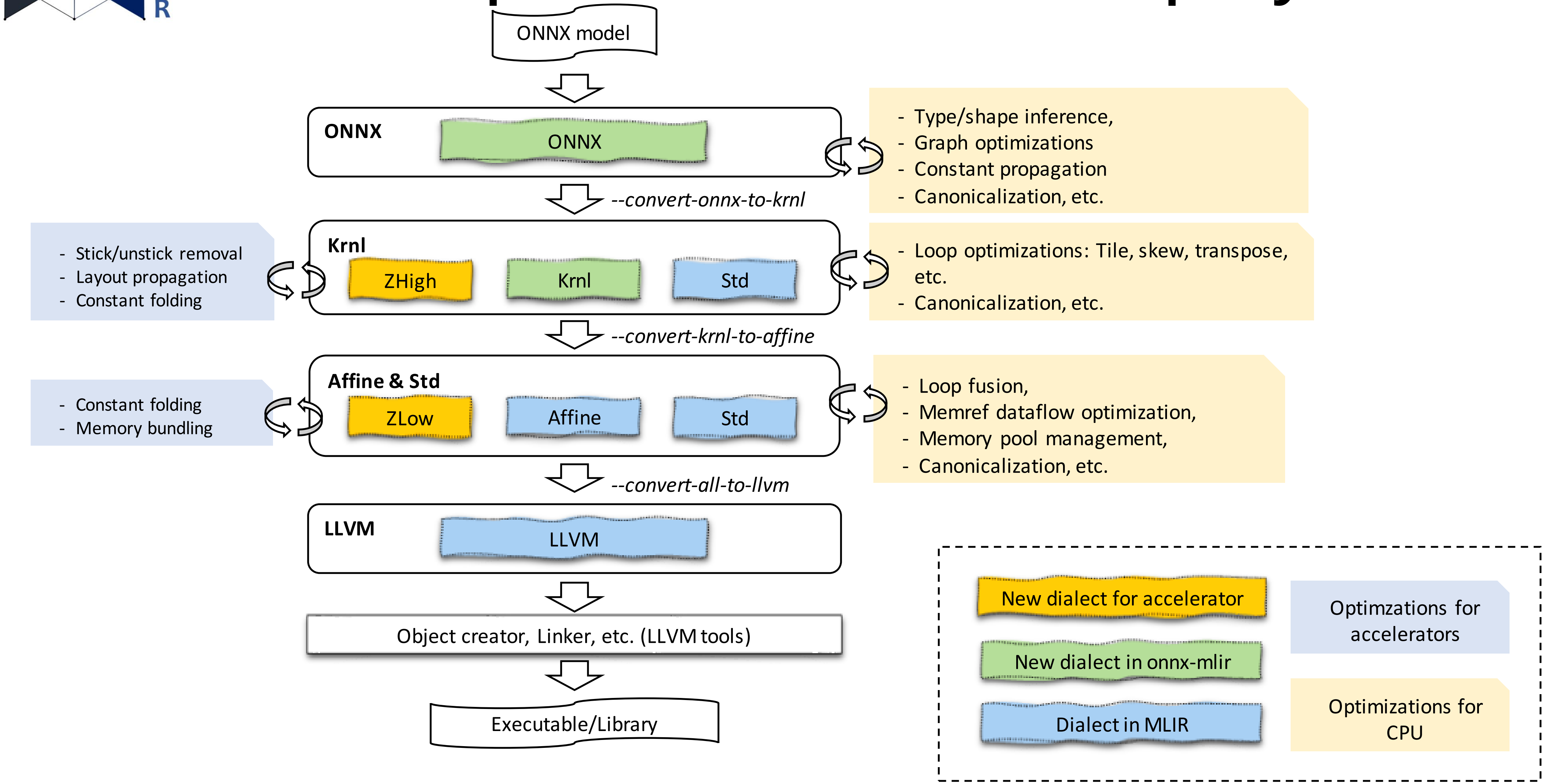


Google TPU



IBM on-chip AI accelerator
in Telum processor

How optimizations are deployed?



ONNX model

ONNX

ONNX

- Type/shape inference,
- Graph optimizations
- Constant propagation
- Canonicalization, etc.

--convert-onnx-to-krnl

- Stick/unstick removal
- Layout propagation
- Constant folding

Krnl

ZHigh

Krnl

Std

- Loop optimizations: Tile, skew, transpose, etc.
- Canonicalization, etc.

--convert-krnl-to-affine

- Constant folding
- Memory bundling

Affine & Std

ZLow

Affine

Std

- Loop fusion,
- Memref dataflow optimization,
- Memory pool management,
- Canonicalization, etc.

--convert-all-to-llvm

LLVM

LLVM

Object creator, Linker, etc. (LLVM tools)

Executable/Library

New dialect for accelerator

New dialect in onnx-mlir

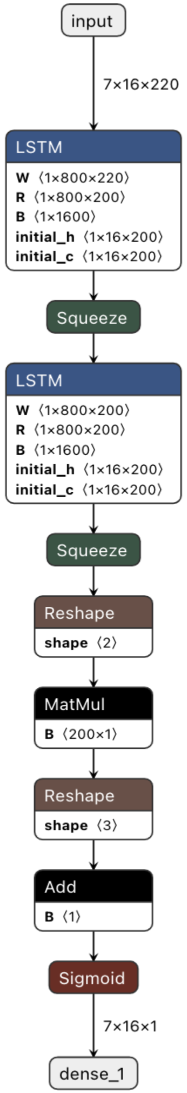
Dialect in MLIR

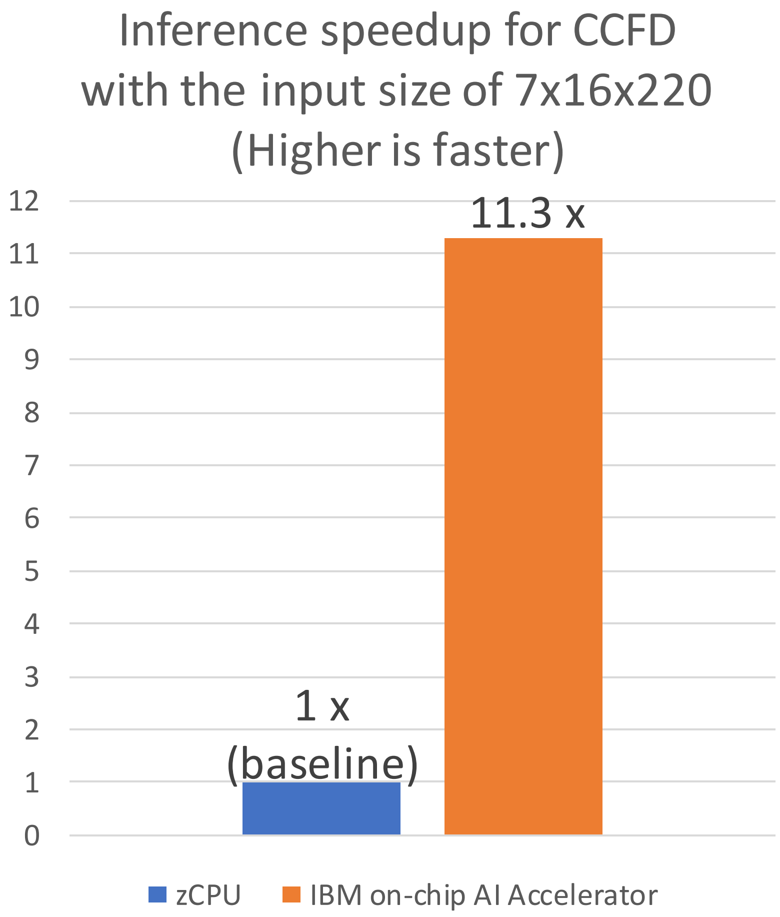
Optimzations for accelerators

Optimizations for CPU

Example: Credit Card Fraud Detection (CCFD)

CCFD model: https://github.com/IBM/ai-on-z-fraud-detectior




Inference speedup for CCFD with the input size of 7x16x220 (Higher is faster)

Real-time detection of fraudulent transactions

Result here is unoffical and does not represent any IBM product

Summary

- Onnx-mlir is an open-source compiler for ONNX models
  - Easy to do optimizations and support new accelerators
- Call for contribution
  - Cool compiler technologies, e.g., AI for compilers, etc.
  - More architectures of interest
- Some areas of interest in near future
  - Optimize operators: Conv, Pooling, Reduction, etc.
  - Support ONNX machine learning operators
  - Support other accelerators, e.g., GPGPU

We truly thank all contributors to onnx-mlir!