TVM: Where are we going

Tianqi Chen

OctoML

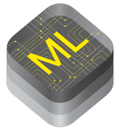
sampl

PAUL G. ALLEN SCHOOL
OF COMPUTER SCIENCE & ENGINEERING

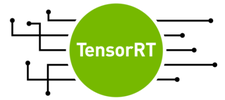
tvm.ai

Current Deep Learning Landscape

Frameworks and Inference engines



DL Compilers

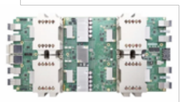

Kenrel Libraries

CuDNN NNPack MKL-DNN

Hand optimized

Hardware

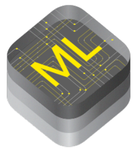



tvm

Open source, automated end-to-end optimization framework for deep learning.

TVM Stack



High-Level Differentiable IR

Tensor Expression and Optimization Search Space

LLVM, CUDA, Metal

VTA



Edge FPGA

Cloud FPGA

ASIC

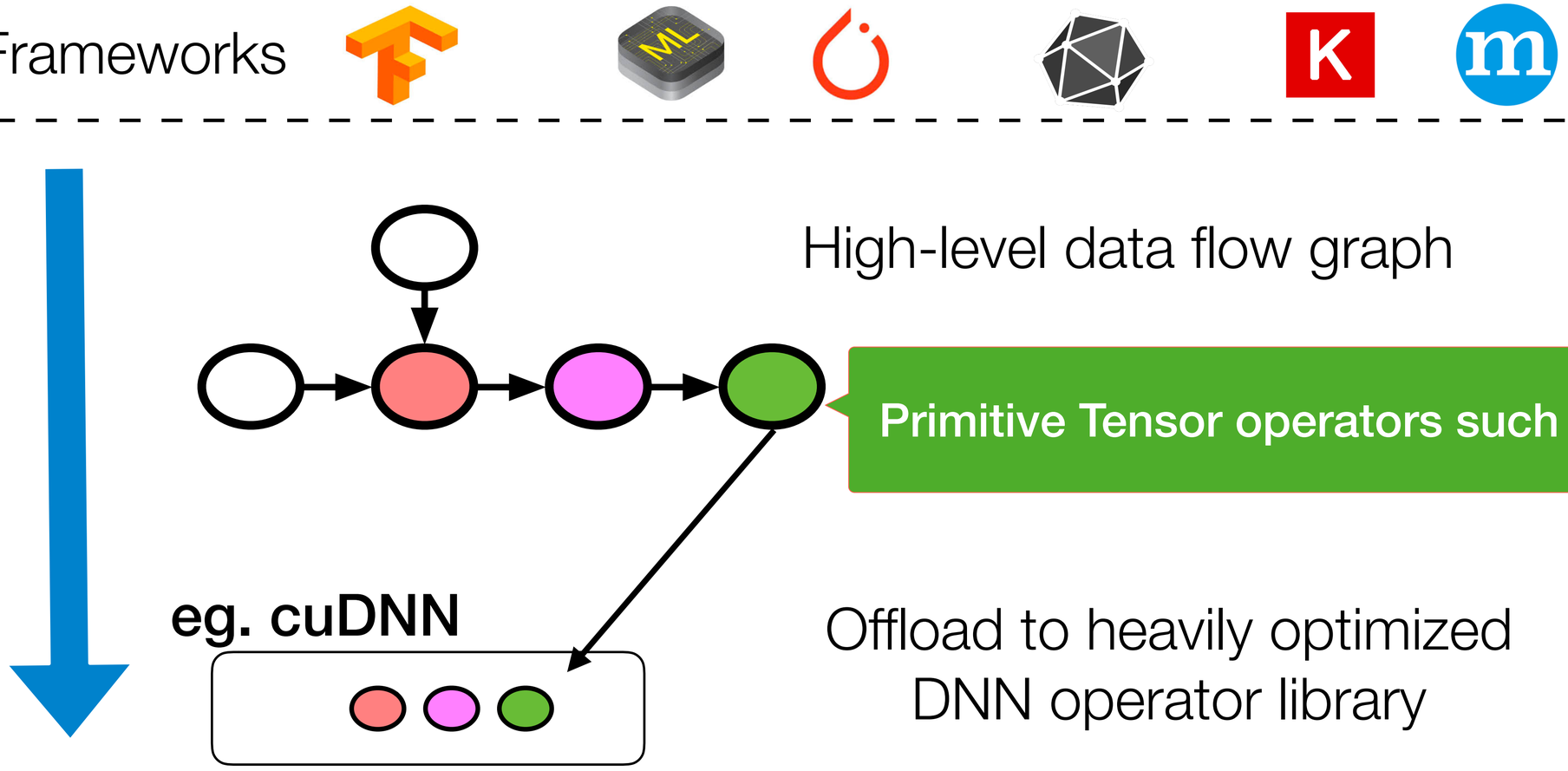
Optimization

AutoTVM

Device Fleet

Existing Deep Learning Frameworks

Frameworks

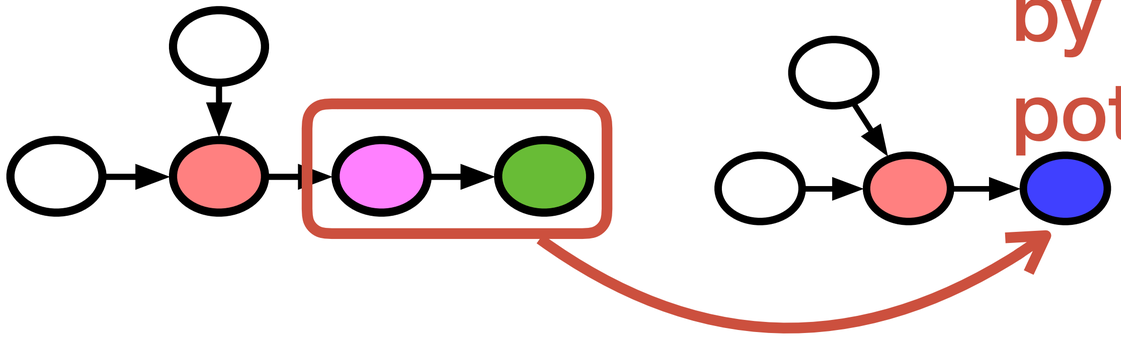


Hardware

NVIDIA

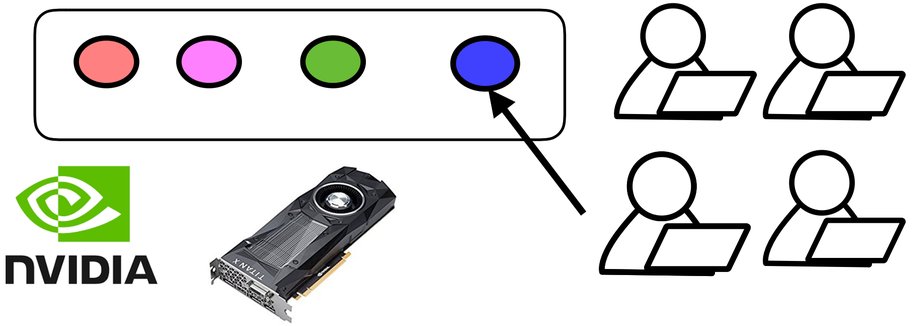
Limitations of Existing Approach

Frameworks



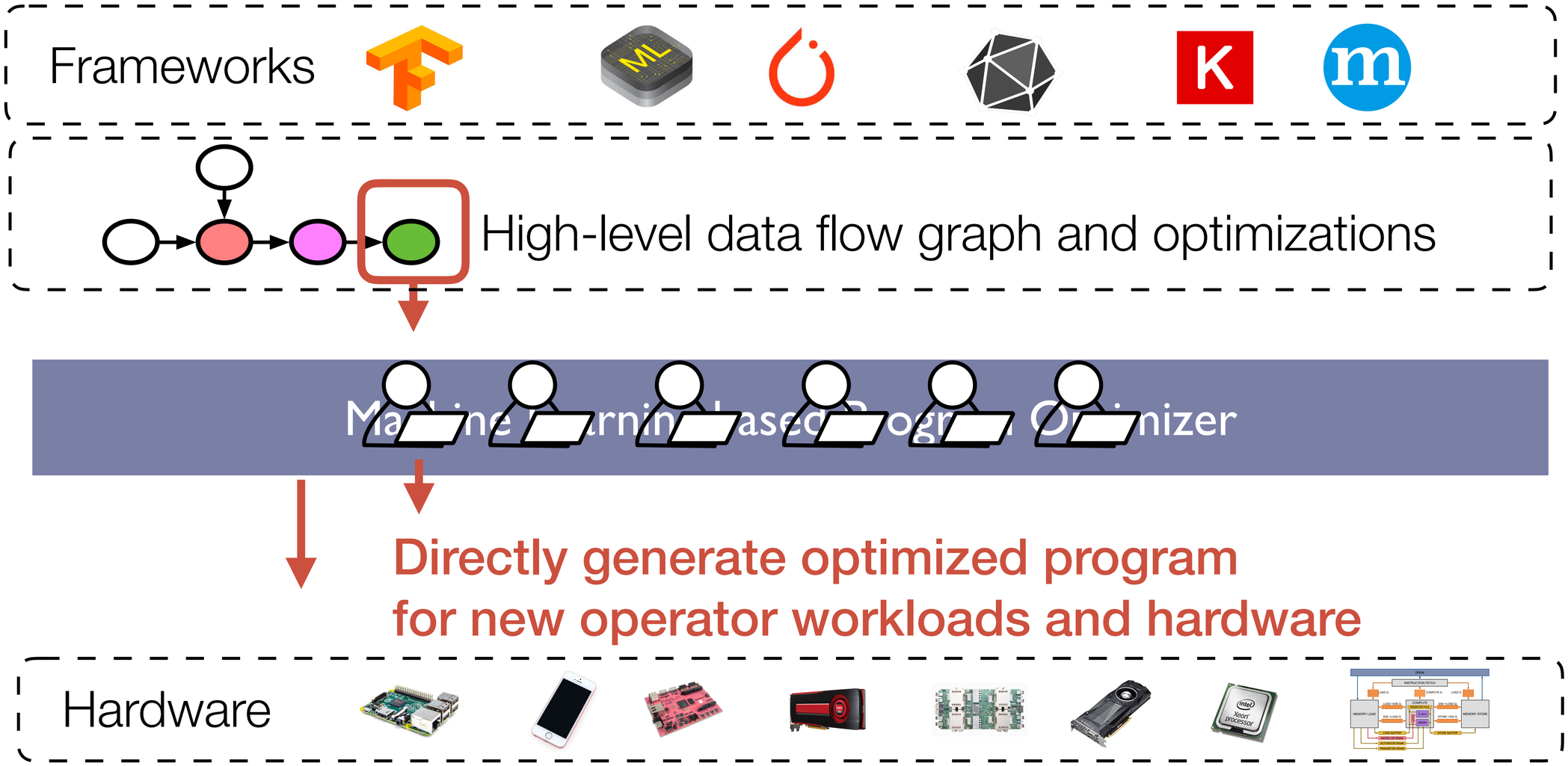
New operator introduced
by operator fusion optimization
potential benefit: 1.5x speedup

cuDNN



Engineering intensive

TVM: Learning-based Learning System



Why Automation is the Future

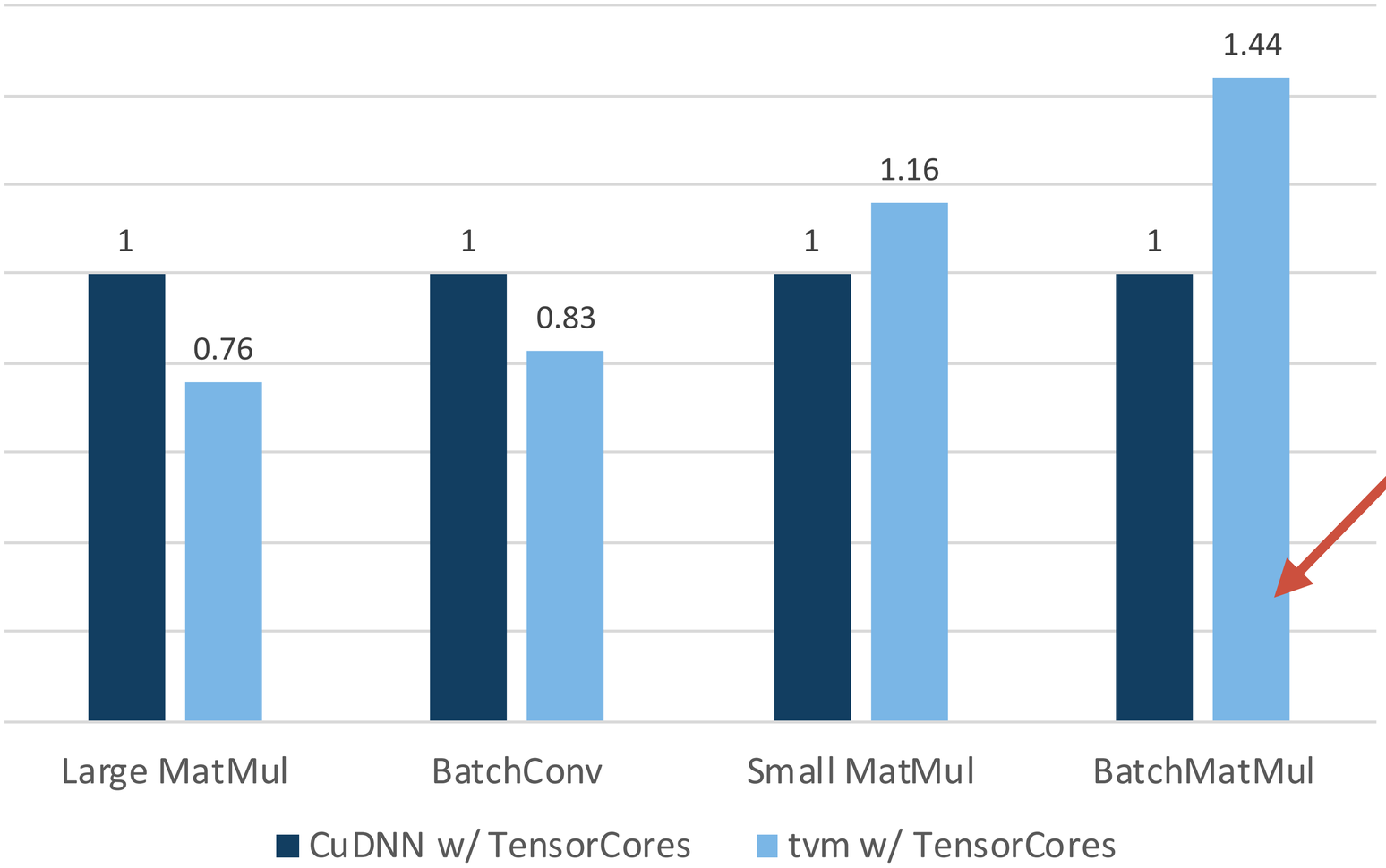
Clear winner on emerging models in product

Competitive on benchmarking type model

Quickly enables other optimizations: fusion, layout, parallelization

Portable performance across devices

Why Automation is the Future



1.4x better on emerging workloads Transformer related workloads

Credit: Siyuan Feng

Where are we going

Unified Runtime For Heterogeneous Devices

Runtime Module Interface

GetFunction(string) -> tvm::runtime::PackedFunc

SaveToBinary/LoadFromBinary

tvm::runtime::Module

Subclasses

NPUModule

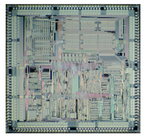
CUDAModule

TFModule

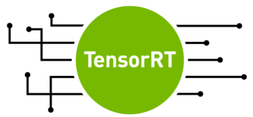
NPU Driver

CUDA Driver

Device Drivers



External Runtimes



Unified Runtime Benefit

Unified library packaging

```
mod.export_library("mylib.so")
```

Free API (Py/Java/Go)

```
lib = tvm.module.load("mylib.so")
func = lib["npufunction0"]
func(a, b)
```

Automatic RPC Support

```
remote = tvm.rpc.connect(board_url, port)
remote.upload("mylib.so")
remote_mod = remote.load_module(“mylib.so")
func = remote_mod[“npufunction0"]
func(remote_a, remote_b)
```

Virtual Machine: Supporting Dynamic Workload

Dynamic shape workloads

More runtime objects: Arrays, Tuples, Trees, ADTs

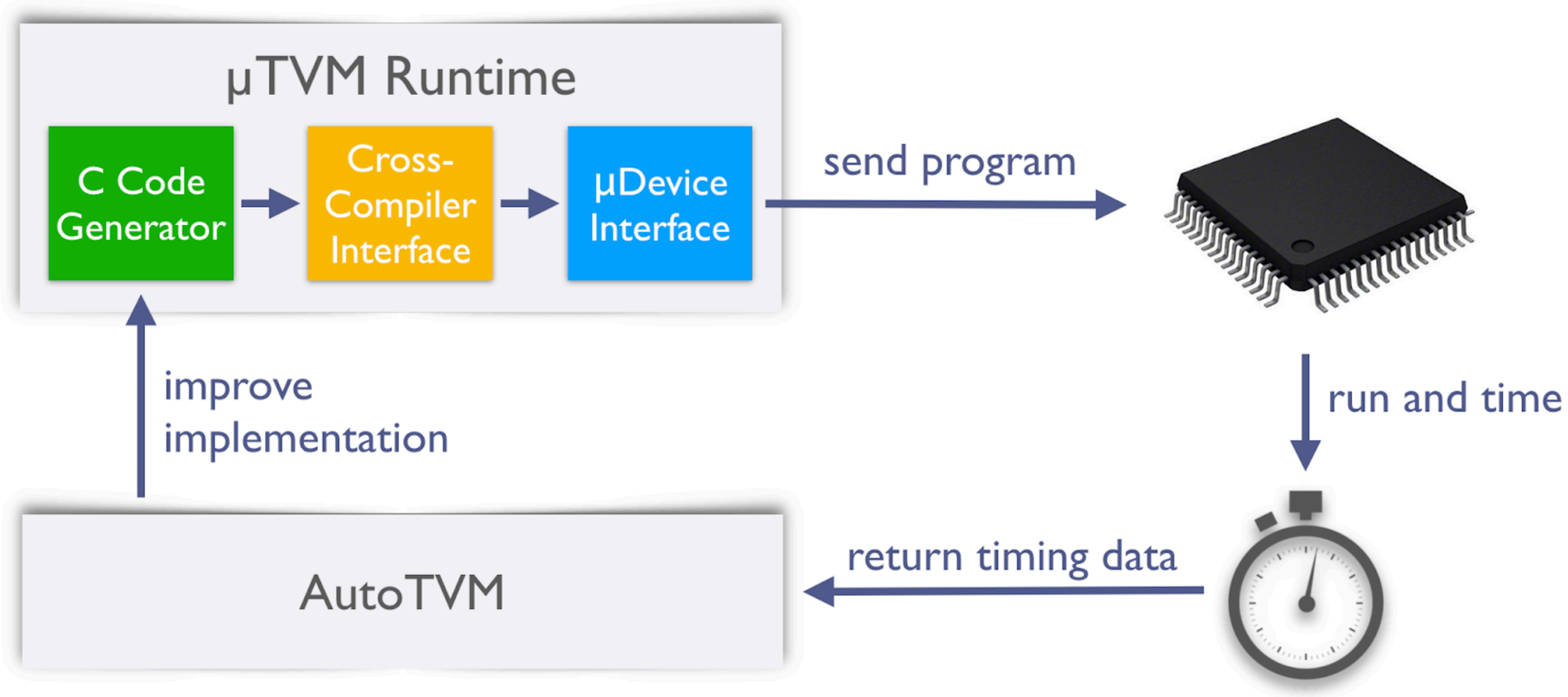
Minimum runtime for dynamic models

Credit: Jared Roesch, Haichen Shen et.al

uTVM: TVM on bare-metal Devices

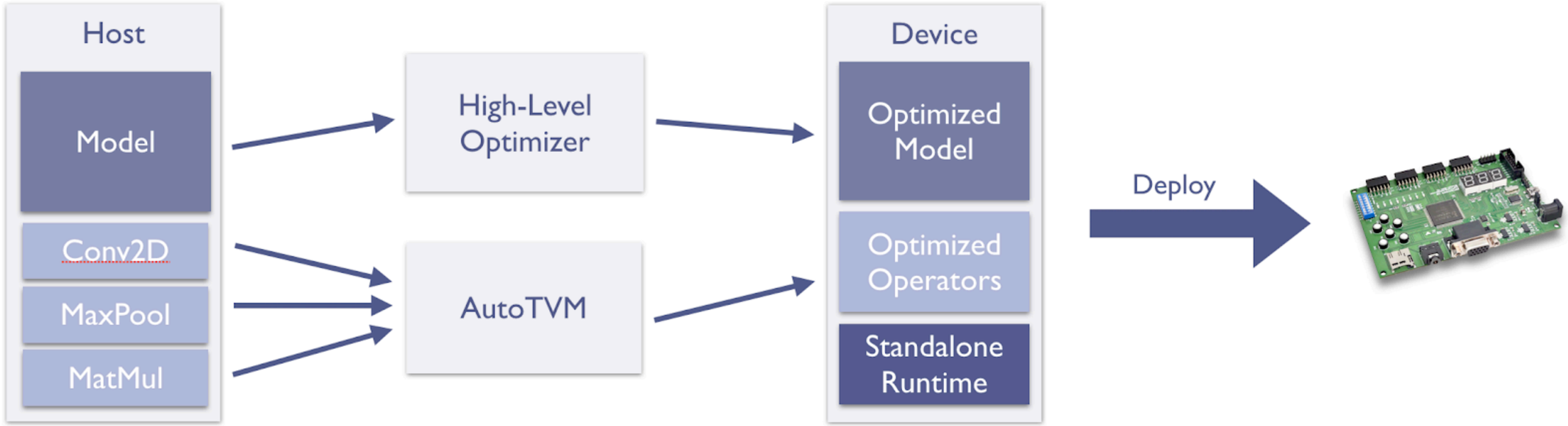
Support bare-metal J-TAG devices, **no OS is needed**

ARM Cortex-M
RISC-V



Credit: Logan Weber

uTVM upcoming: Self Hosted Runtime

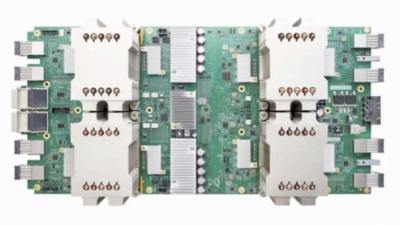


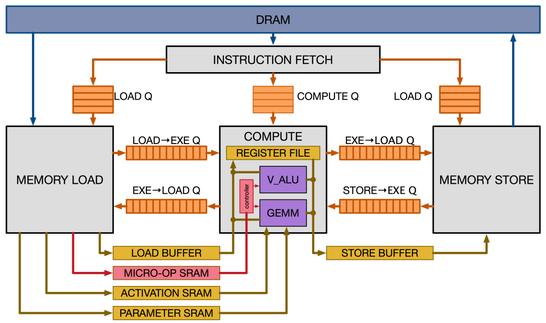
Credit: Logan Weber

Designed for Accelerators(NPU)

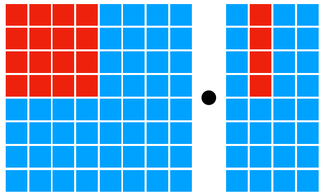
Search Space for TPU-like Specialized Accelerators

TPUs

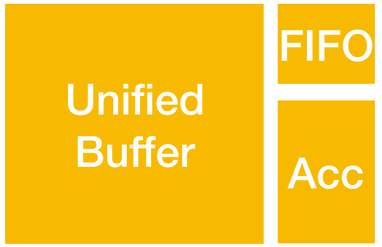




Tensor Compute Primitives

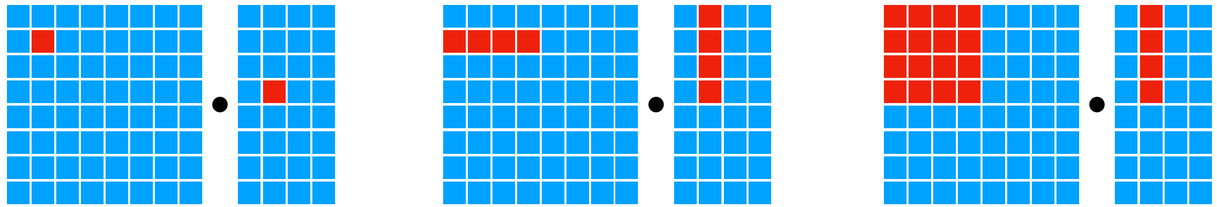


Explicitly Managed Memory Subsystem



Tensorization Challenge

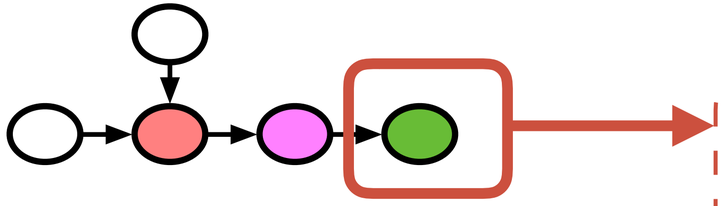
Compute primitives



scalar *vector* *tensor*

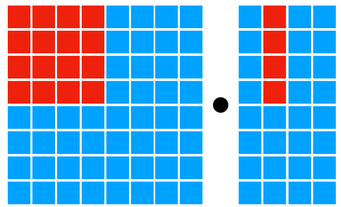
Challenge: Build systems to support emerging tensor instructions

Tensorization Challenge



Computation Specification (Tensor Expression)

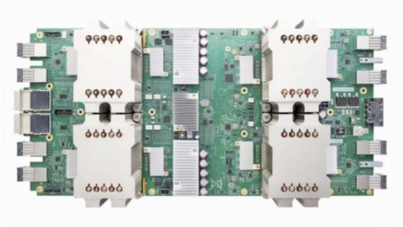
```
C = tvm.compute((m, n),
    lambda y, x: tvm.sum(A[k, y] * B[k, x], axis=k))
```

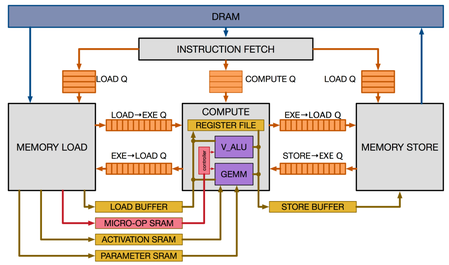


```
A = tvm.placeholder((8, 8))
B = tvm.placeholder((8,))
k = tvm.reduce_axis((0, 8))
C = tvm.compute((8, 8),
    lambda y, x: tvm.sum(A[k, y] * B[k], axis=k))
```

Tensorization

HW Interface Specification by Tensor Expression





VTA: Open & Flexible Deep Learning Accelerator

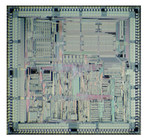
Current TVM Stack

VTA Runtime & JIT Compiler

VTA Hardware/Software Interface (ISA)

VTA MicroArchitecture

VTA Simulator



- Runtime JIT compile accelerator micro code
- Support heterogenous devices, 10x better than CPU on the same board.
- Move hardware complexity to software

compiler, driver,
hardware design
full stack open source

HW-SW Blueprint for Flexible Deep Learning Acceleration. **Moreau et al. IEEE Micro 2019.**

TSIM: Support for Future Hardware

Current TVM Stack

New NPU Runtime

TSIM Driver

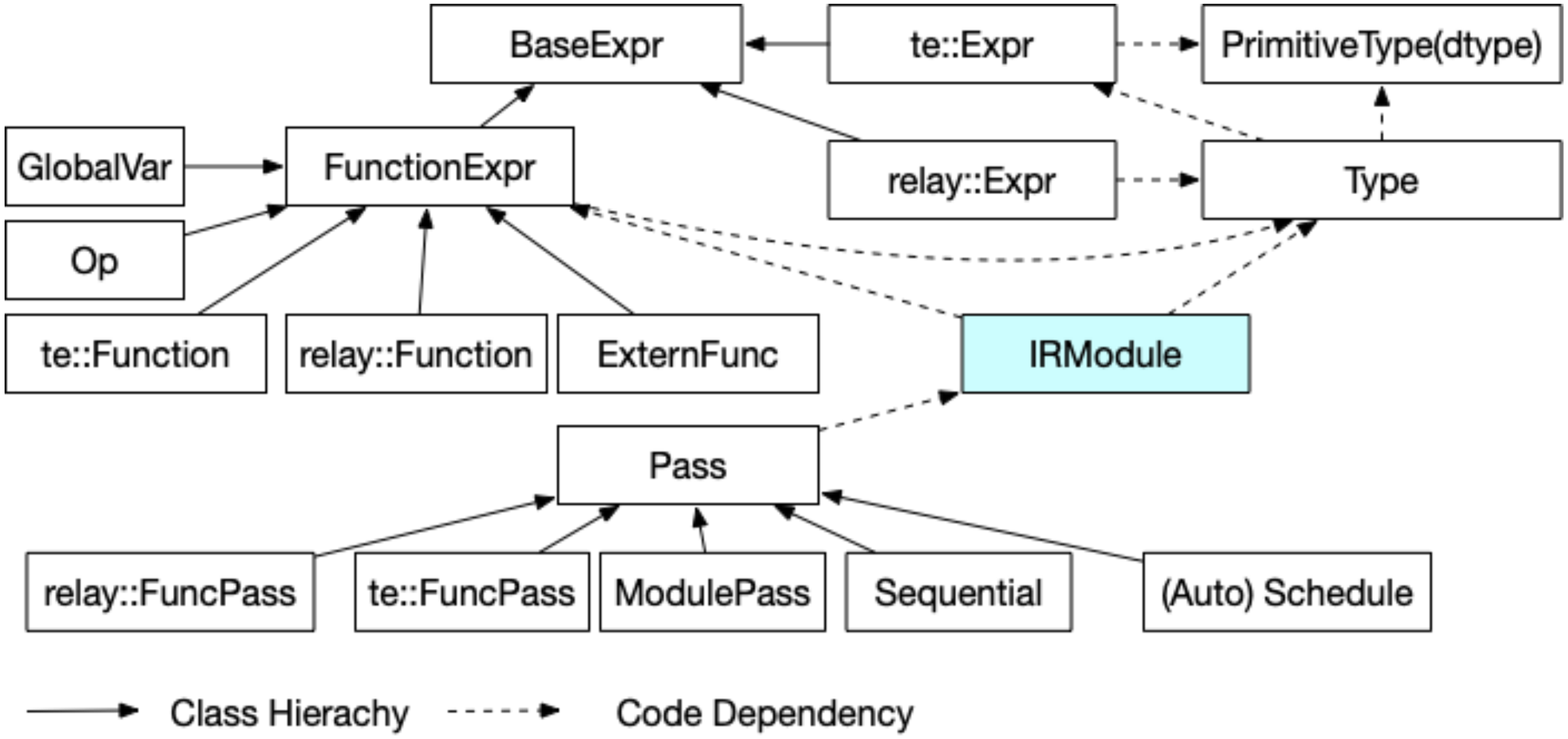
TSIM Binary

New Hardware Design in Verilog

Verilator

Toward Unified IR Infra

Overview of New IR Infra



Single unified module/pass, type system, with function variants support

Compilation Flow under the New Infra

Import

IRModule (relay::Function)

High-level optimizations

Lower

IRModule (te::Function, ExternFunc, …)

(Auto) Schedules
Low-level optimizations

Codegen

runtime::Module

Mixed Function Variants in the Same Module

```
def @relay_add_one(%x : Tensor((10,), f32)) {
    call_destination_passing @te_add_one(%x,  out=%b)
}

def @te_add_one(%a: NDArray, %b: NDArray) {
    var %n
    %A = decl_buffer(shape=[%n], src=%a)
    %B = decl_buffer(shape=[%n], src=%b)
    for %i = 0 to 10 [data_par] {
        %B[%i] = %A[%i] + 1.0
    }
}
```

First-class Python Support

```
@tvm.hybrid
def te_add_one(a, b):
    n = var(“n”)
    A = bind_buffer(shape=[n], a)
    B = bind_buffer(shape=[n], b)
    for i in iter_range(n, iter_type=”data_par”):
        A[i] = B[i] + 1
```

Use hybrid script as an alternative text format

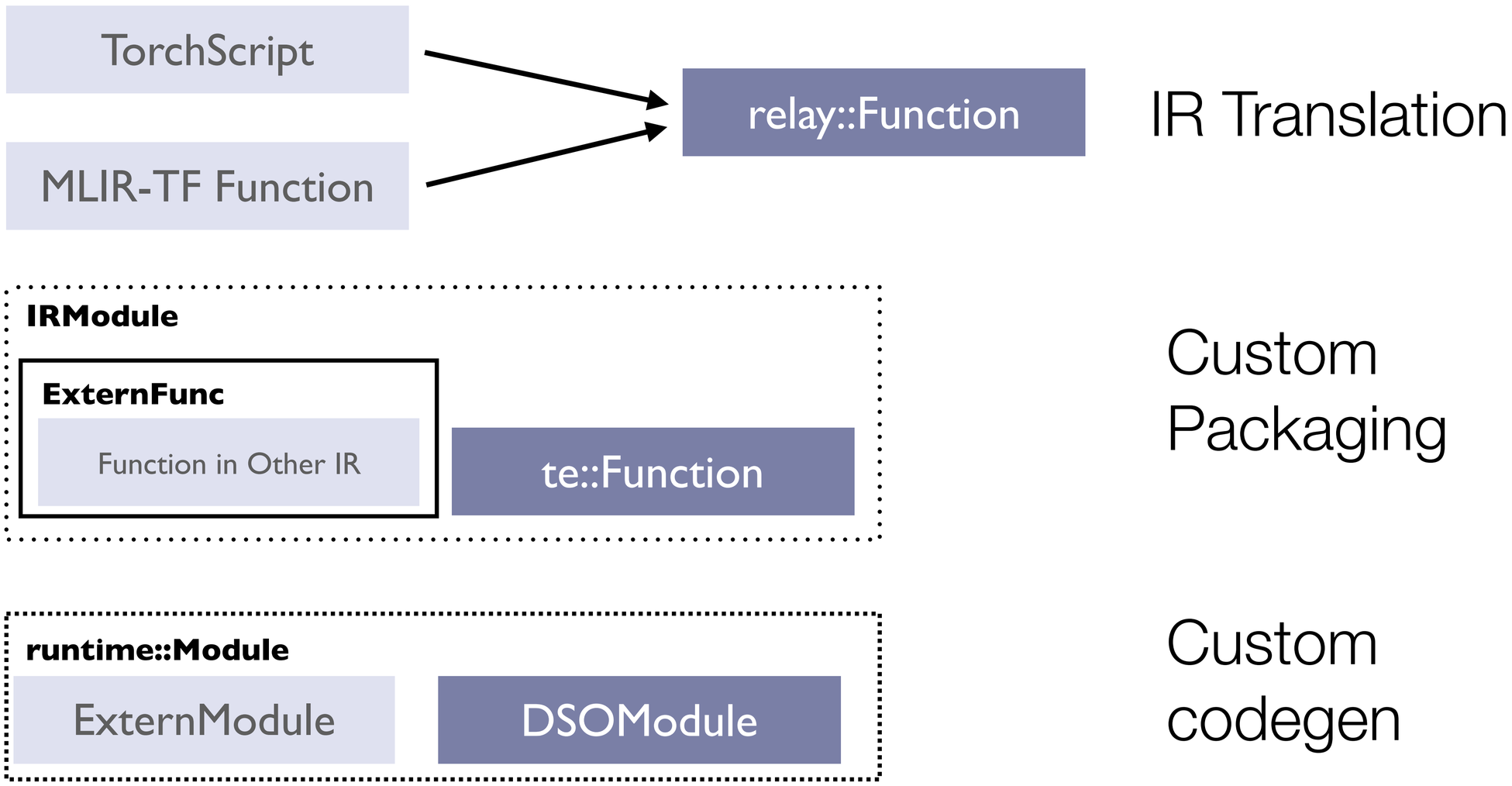
```
mod = tvm.IRModule([te_add_one])
print(mod[”te_add_one”].args)
```

Directly write pass, manipulate IR structures

Accelerate innovation,
e.g. use (GA/RL/BayesOpt/your favorite ML method) for AutoSchedule

Easy shift to C++ when product ready

Interpolate with Other Compilers



Timeline

RFC: now

Main area of focus in next 3-4 month

More updates about tensor-level IR at the TVM conference

Community

Open Source Community

Open source: ~280 contributors from UW, Berkeley, Cornell, UCLA, Amazon, Huawei, NTT, Facebook, Microsoft, Qualcomm, Alibaba, Intel, …

Incubated as Apache TVM recently. Independent governance, allowing competitors to collaborate.

Open Code

Open Development

Open Governance

Acknowledgement

Apache (incubating) TVM community

Our awesome community members!

Alibaba for hosting the meetup

TVM Conference 2019

Thu, December 5th 2019, Seattle.

https://sampl.cs.washington.edu/tvmconf/

2 days before NeurIPS, Vancouver is 3h away from SEA