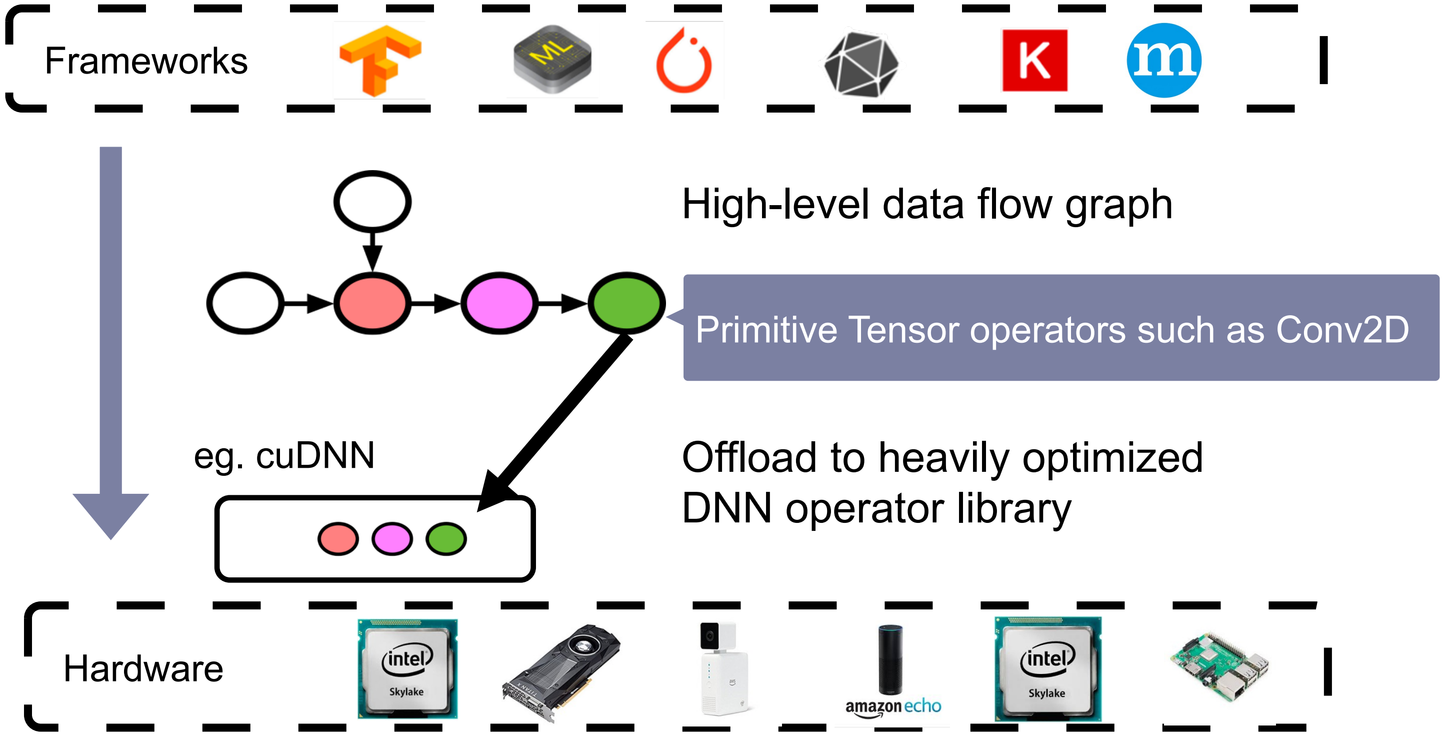
Making Deep Learning More Portable with Deep Learning Compiler

Cody Yu (hyuz@amazon.com),

Senior Applied Scientist, Deep Engine Science, AWS AI
Project Management Committee (PMC) member, Apache TVM

Existing Deep Learning Frameworks



Why Deep Learning Compiler?

Usability: Users have to program/deploy models for a framework

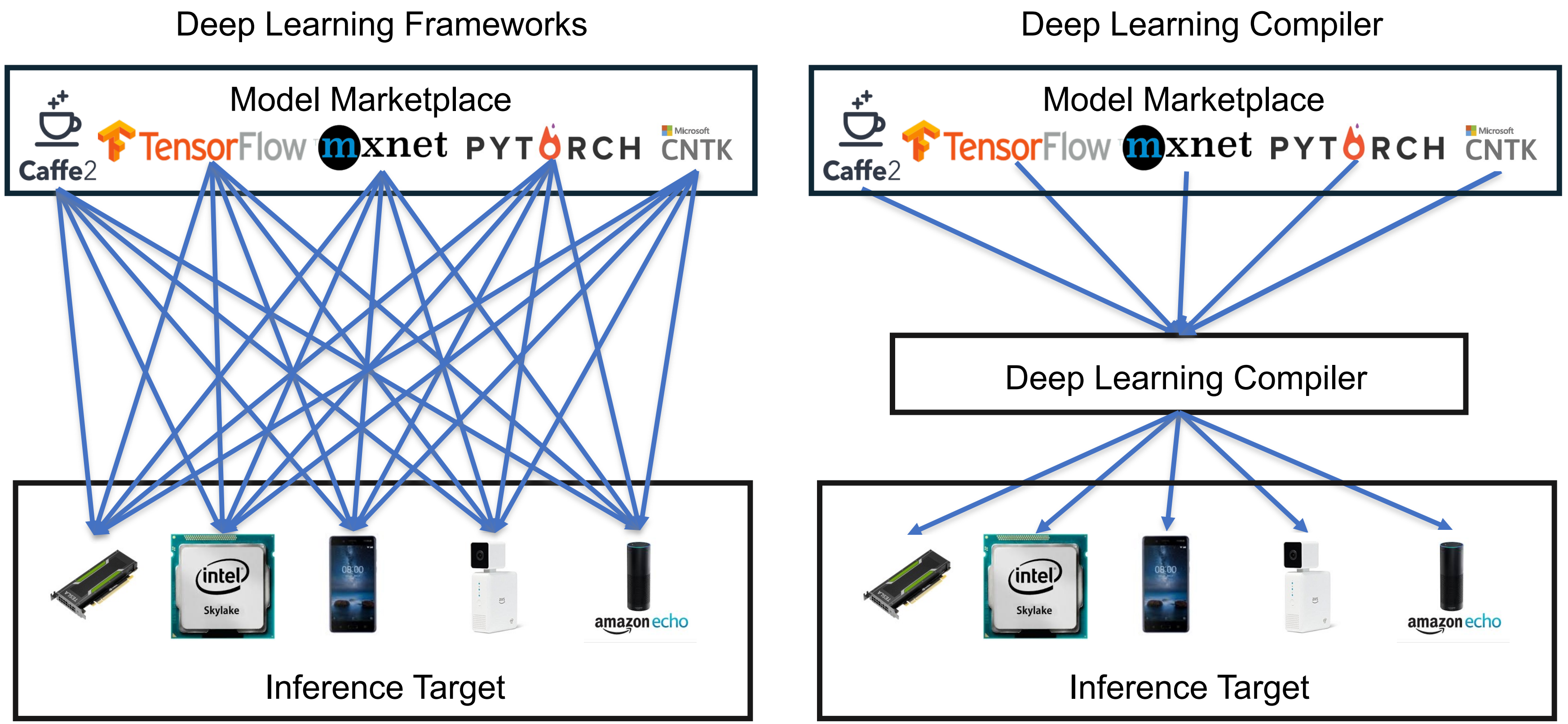
- Multiple frameworks
- Multiple platforms

Performance portability: Frameworks invoke kernel libraries

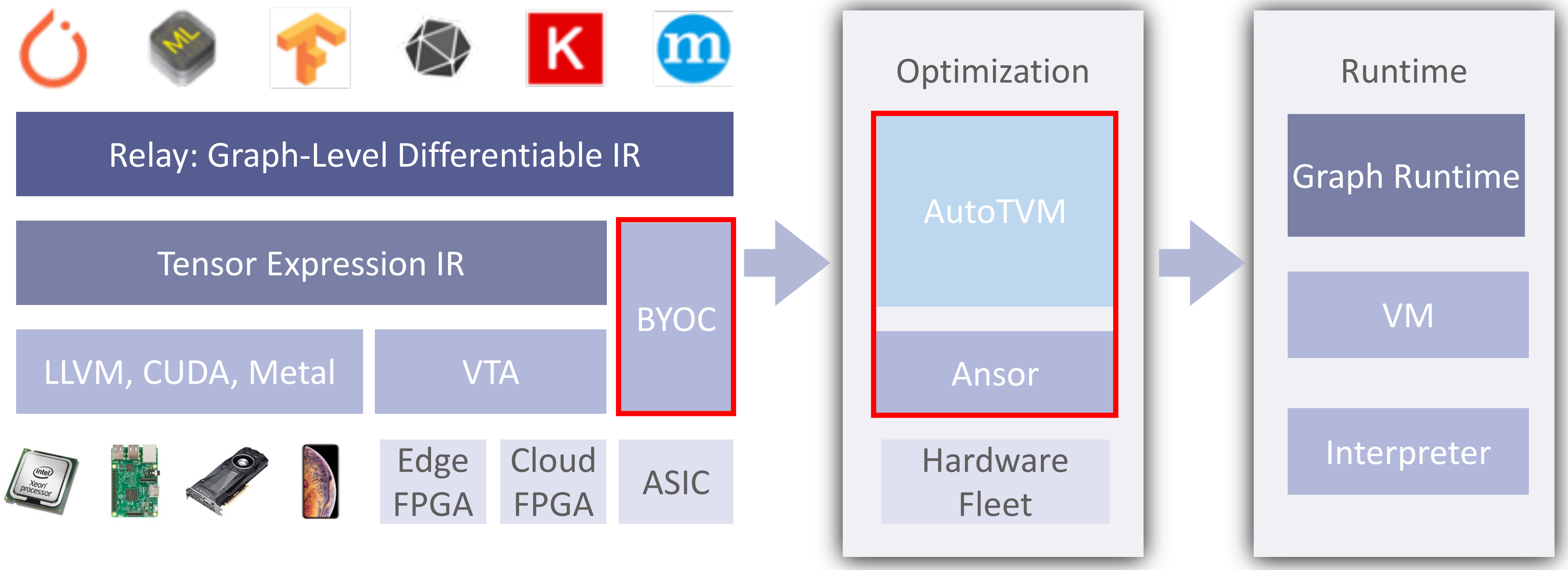
- What if the targeting platform doesn’t have a (high-performance) kernel library?
- What if the operator is not included in the kernel library?
- Does the kernel library do enough optimization?
- How to apply more aggressive graph-level optimizations (e.g., operator fusion)?

Deep Learning Compiler

DL compiler serves as a unified intermediate layer similar to LLVM



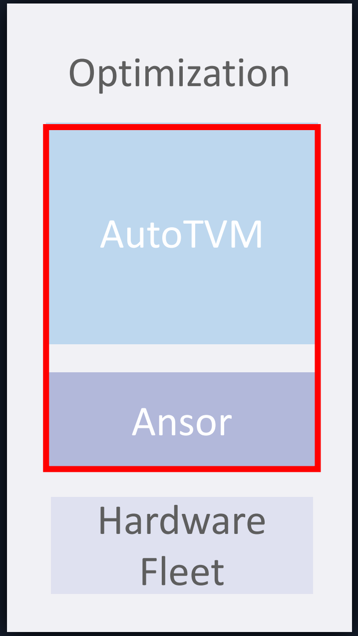
Apache TVM: An End-to-End Deep Learning Compiler



Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan et al. "**TVM: An automated end-to-end optimizing compiler for deep learning**." *In 13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18), pp. 578-594. 2018*.

Porting the Performance to General Processors (CPU, GPU)

AutoTVM: Performance auto-tuning with schedule templates
Ansor: generating high-performance schedules from scratch



Tensor-Level IR in TVM

- Tensor-level IR is similar to code AST
  - The arithmetic expression of the operator. Example: Tensor addition

```
n = te.var("n")
A = te.placeholder((n,), name="A")
B = te.placeholder((n,), name="B")
C = te.compute(A.shape, lambda i: A[i] + B[i], name="C")
```

```
for (int i = 0; i < n; ++i) {
  C[i] = A[i] + B[i];
}
```

  - The hardware dependent schedule
    - Example: loop splitting with the optimized factor for a certain device

```
s = te.create_schedule(C.op)
bx, tx = s[C].split(C.op.axis[0], factor=64)
```

```
for (int bx = 0; bx < ceil(n / 64); ++bx) {
  for (int tx = 0; tx < 64; ++tx) {
    int i = bx * 64 + tx;
    if (i < n) {
      C[i] = A[i] + B[i];
}}}
```

- Relay IR can be lowered to tensor-level IR by given the target hardware device

AutoTVM: Template-based Performance Auto-Tuning

- It is challenging to have a schedule fitting to all devices
  - e.g., NVIDIA T4, V100, and A100 have different GPU architectures and configurations
- Can we let TVM realize the best schedule configuration by given the target device?
- AutoTVM: A learning- based auto-tuning framework

Original Schedule

```
s = te.create_schedule(C.op)
bx, tx = s[C].split(C.op.axis[0], factor=64)
```

Schedule with an AutoTVM tuning space

```
s = te.create_schedule(C.op)
cfg = autotvm.ConfigSpace()
bx, tx = cfg.define_split("c_factor", C.op.axis[0], num_outputs=2)
```

Symbolic axes can be used by the rest schedule primitives

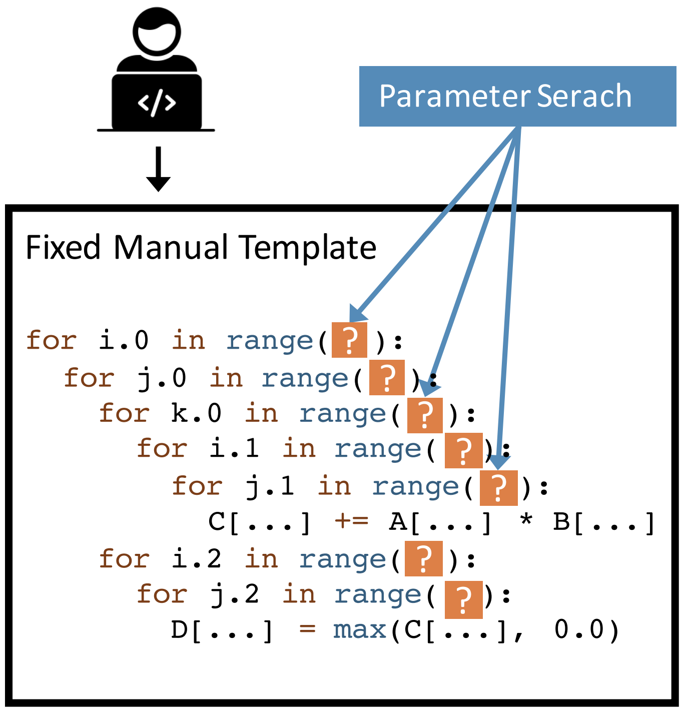
Challenges in AutoTVM

Template-guided search

- Use templates to define the search space
- Write a template for every operator

Drawbacks

- Templates are hard to write
  - Need knowledge of hardware and operator
- The number of required templates is large
  - 15k+ lines of code in TVM repo
  - continues to grow as new op comes
- The templates are not optimal
  - Manual enumeration cannot cover all optimizations



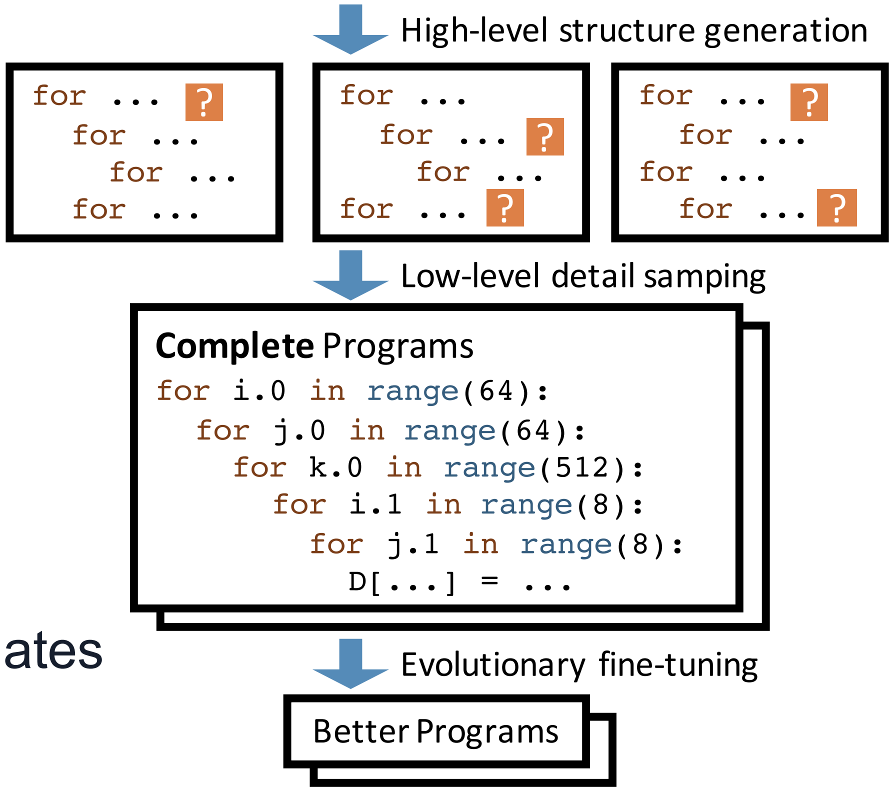
Ansor: Generating Schedules from Scratch

Hierarchical Approach

- Phase 1: High-level program structure generation
  - Generate a few program structures (**sketches**)
- Phase 2: Low-level detail sampling
  - Turn sketches to **complete** programs
- Phase 3: Performance fine-tuning with a cost model
  - Evolutionary tune the program performance

Feature Highlights

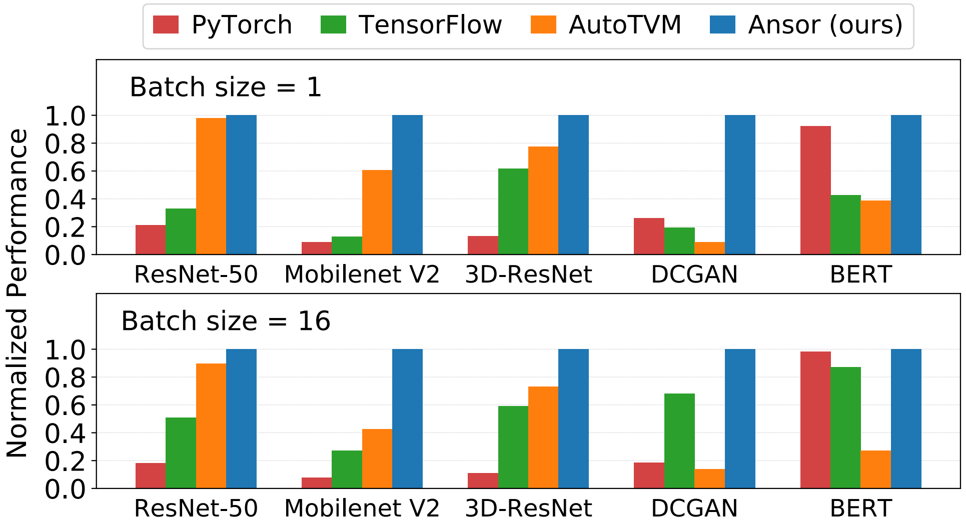
- Tune **any** compute function without predefined schedule templates
- Extract tuning tasks based on operator fusion results
- Prioritize performance bottlenecks



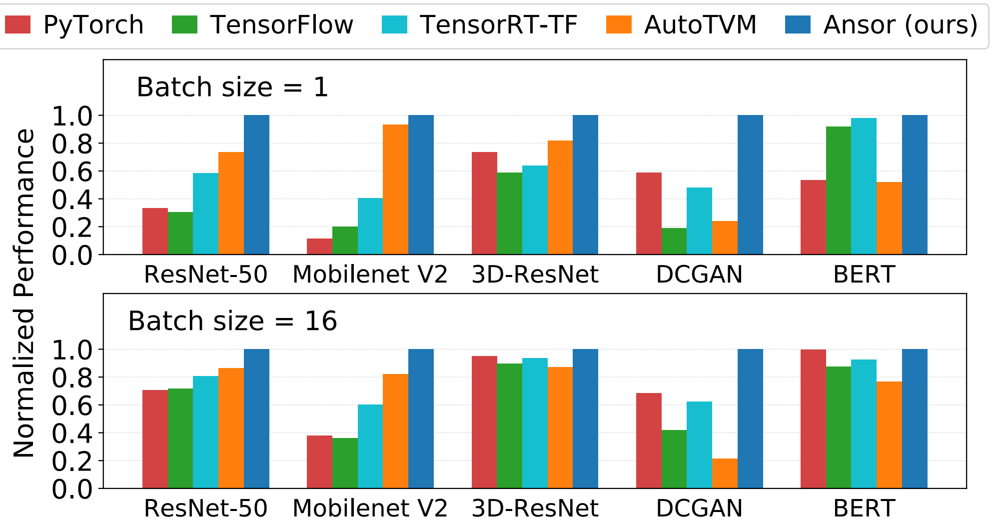
(c) Ansor's Hierarchical Approach

Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Wu, Cody Hao Yu, Ameer Haj-Ali, Yida Wang, Jun Yang, Danyang Zhuo, Koushik Sen, Joseph E. Gonzalez, Ion Stoica. "**Ansor: Generating high-performance tensor programs for deep learning**." *In 14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20), pp. 863-879. 2020.*

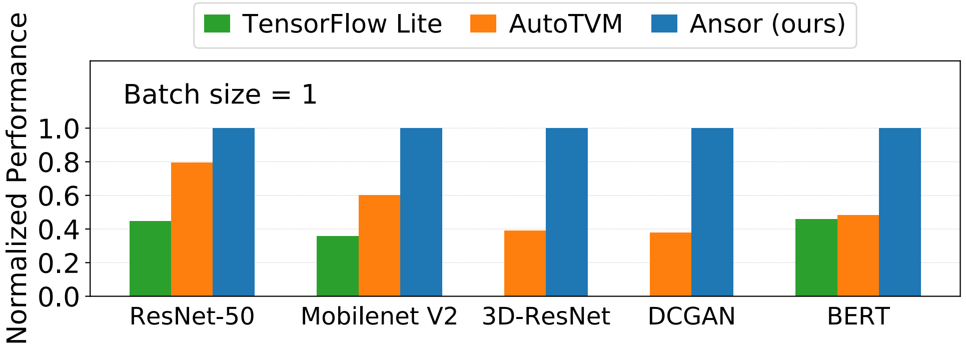
Evaluation: End-to-End Network



Intel-Platinum-8269CY (20 cores)



NVIDIA V100



ARM Cortex-A53 on Rasp3 (4 cores)

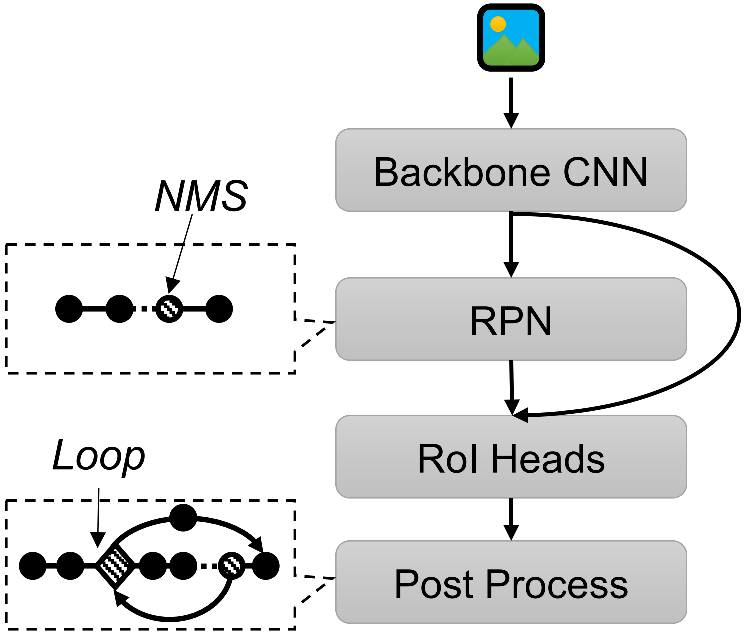
Analysis

- Ansor performs best or equally the best in all test cases with up to 3.8x speedup
- **Ansor delivers portable performance**

Porting the Performance to Custom Accelerators

BYOC: Bring Your Own Codegen to Apache TVM

A Motivating Example: R-CNN



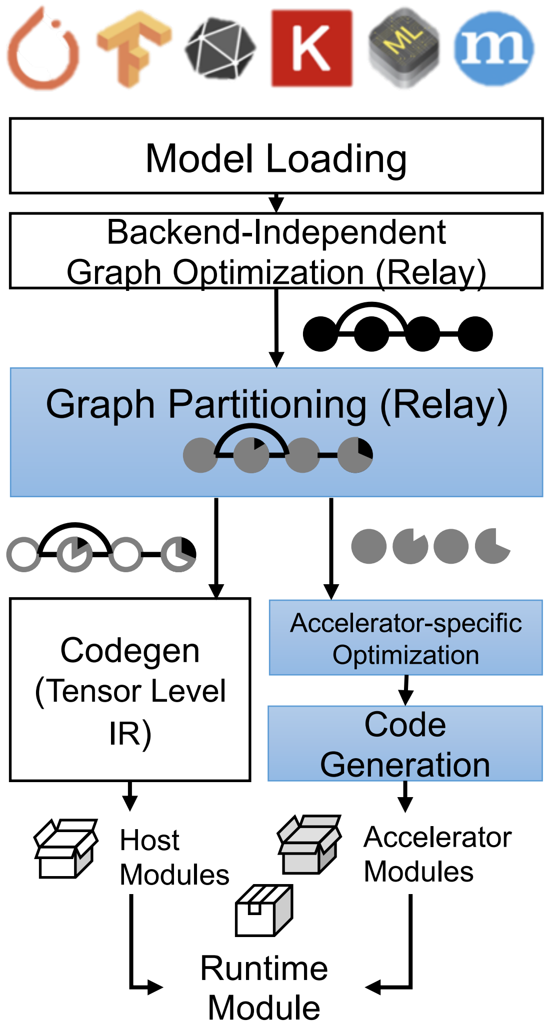
A common CNN (e.g., ResNet) (fully offloadable)

Region Proposal Network
(non-maximum suppression is usually not offloadable)

Region of Interest (fully offloadable)

Post process (loop structure is usually not offloadable)

Bring Your Own Codegen (BYOC) Flow Overview



Convert a model file in any format to a Relay graph

Apply common graph optimizations
(e.g., constant folding, dead code elimination, etc)

Partition the graph to two sets
- Left: Unsupported ops/structures remain on TVM
- Right: Supported ops/structure to your codegen

Custom quantization, data layout transform, etc

Generate/compile code for your runtime

Chen, Zhi, Cody Hao Yu, Trevor Morris, Jorn Tuyls, Yi-Hsiang Lai, Jared Roesch, Elliott Delaye, Vin Sharma, Yida Wang. "**Bring Your Own Codegen to Deep Learning Compiler**." *arXiv preprint arXiv:2105.03215 (2021)*.

BYOC is popular in Apache TVM!

- Many backend integrations are open source available
- Many AI accelerator vendors have embraced Apache TVM with BYOC as their official compiler solutions
  - AWS
  - Marvell
  - Qualcomm
  - SiMa.ai
  - Tencent
  - MediaTek
  - ITRI
  - …and more

Accelerator	Compiler Stack
NVIDIA GPUs	NVIDIA TensorRT
NVIDIA GPUs	NVIDIA CUTLASS
Apple NN processors	Apple CoreML
Apple NN accelerator platforms	Apple BNNS Library
Xilinx DPU (cloud & edge FPGAs)	Xilinx Vitis-AI
Intel x86 CPUs	Intel OneDNN
Arm Cortex-M NPUs	Arm CMSIS-NN
Arm Ethos NPUs	Arm Ethos compiler
Arm CPUs	Arm Compute Library

Status of Apache TVM (as of Dec. 2021)

Community

- Github stars: 7.5k
- Contributors: 600+

Discuss forum

- 122k pageviews
- ~3k user visits per month

Regular events/meetups

- Monthly community meetup
- Annual conference

TVM Community @ Discord
https://discord.gg/QnZRa6eC