深度学习模型优化方法

2019年11月11日 13:38

- TensorRT
- Tensorflow xla
- Tvm
- 模型压缩
- 模型量化
    - 半精度/int8/二值神经网络
    - 混合精度训练

TVM文献

2019年10月25日 14:17

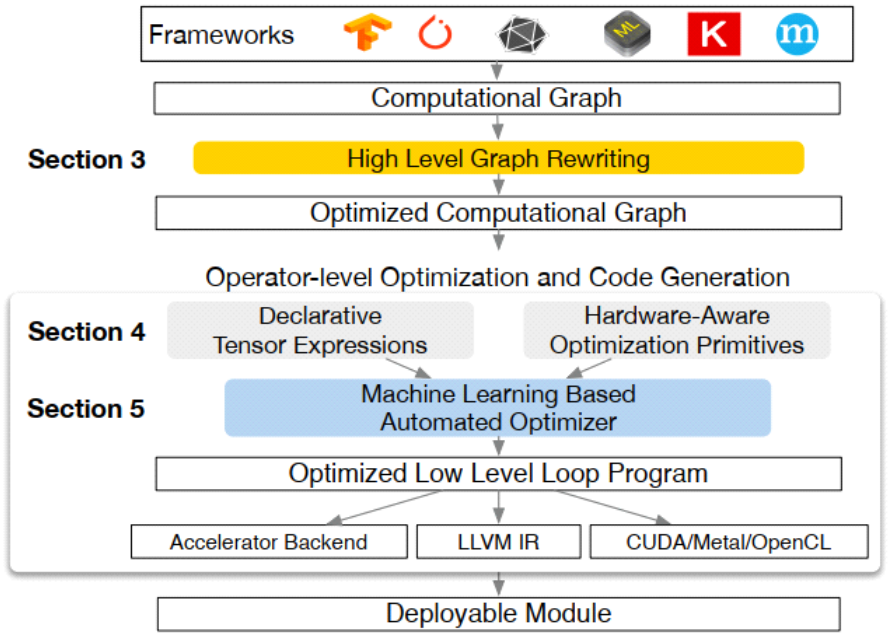
- 

Figure 2: System overview of TVM. The current stack supports descriptions from many deep learning frameworks and exchange formats, such as CoreML and ONNX, to target major CPU, GPU and specialized accelerators.

- 顶层架构
  - 解决问题：
    - 多设备（终端）部署
    - 多层级优化(graph-level, operator-level)
- Optimizing computatinal graphs 优化计算图
  - Operator fusion 算子合并
    - Without saving intermediate results 不保存中间结果
    - 1.2* to 2* speedup

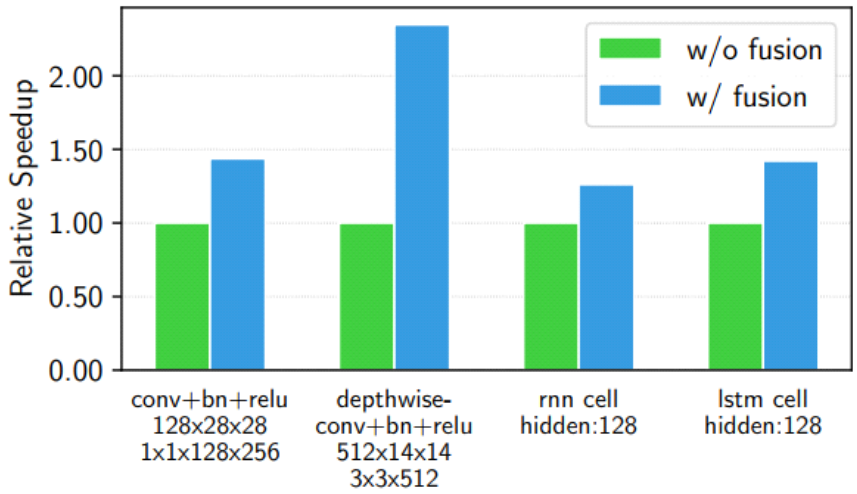


Figure 4: Performance comparison between fused and non-fused operations. TVM generates both operations. Tested on NVIDIA Titan X.

    - Data layout transformation 数据布局优化
        - Data layout optimization converts a computational graph into one that can use better internal data layouts for execution on the target hardware.
- Generating tensor operations
    - Halide's idea: decoupling description from computation rules 计算与调度分离
    - Tensor expression and schedule space
        - Adopting the decoupled compute/schedule principle from Halide [32], we use a schedule to denote a specific mapping from a tensor expression to low-level code
        - 用调度(schedule)来表示tensor表达式到底层代码的映射
    - Nested parallelism with cooperation
        - 协作多线程方式：多个线程共同使用一块共享内存区。可充分利用GPU特性提升性能。

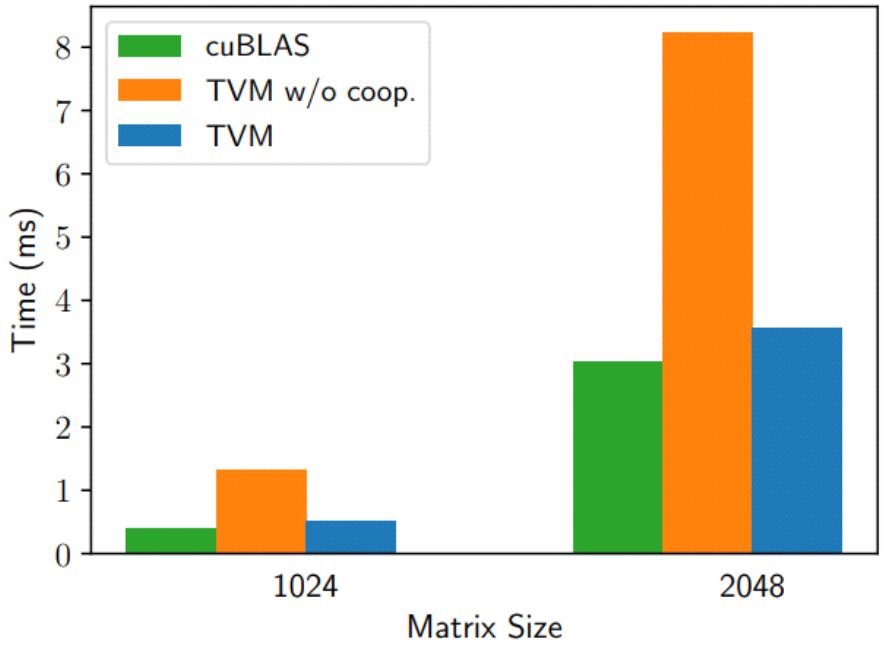


Figure 7: Performance comparison between TVM with and without cooperative shared memory fetching on matrix multiplication workloads. Tested on an NVIDIA Titan X.

- Tensorization 向量化
  - 深度学习中的操作可以分解为向量操作，比如矩阵相乘以及1维卷积
  - Cannot support a fixed set of primitives， need to be extensible
  - Solution for extensibility: separating the target hardware intrinsic from the schedule
  - Tensorization decouples the schedule from specific hardware primitives, make it easy to extend TVM to support new hardware architectures
- Explicit memory latency hiding 内存延时隐藏
  - Latency hiding: the process of overlapping memory operations with computation to maximize utilization of memory and compute resources. 将内存操作与计算操作同时交错执行
  - Depend on the hardware 与具体硬件相关
    - GPU: rapid context switching of many warps of threads
    - TPU: leaner control with a decoupled access-execute architecture
    - Virtual threading scheduling primitives

- Automating optimization
  - Find optimal operator implementations for **each layer** of a DL model 为深度学习模型的每一层找到最佳的算子实现
  - Automated schedule optimizer 自动调度优化
  - A schedule explorer, and a machine learning cost model 使用机器学习模型来找最好的schedule
  - Schedule space specification
    - Schedule template specification API
  - Machine learning based cost model
    - Configuration space很大，可以使用blackbox optimization. But require many

experiments

- An alternate approach: build a predefined cost model to guide the search 定义一个代价模型，从而避免搜索整个配置(configuration)空间
- Ideally, a perfect cost model considers all factors affecting performance, which is too burdensome. Instead, a statistical approach is applied.
- For each schedule configuration, a ML model is introduced to predict the running time on a given hardware.
- 与传统的超参优化任务不同

- Evaluation
  - Implemented in C++, language bindings to python and java
  - Server-class GPU results
  -

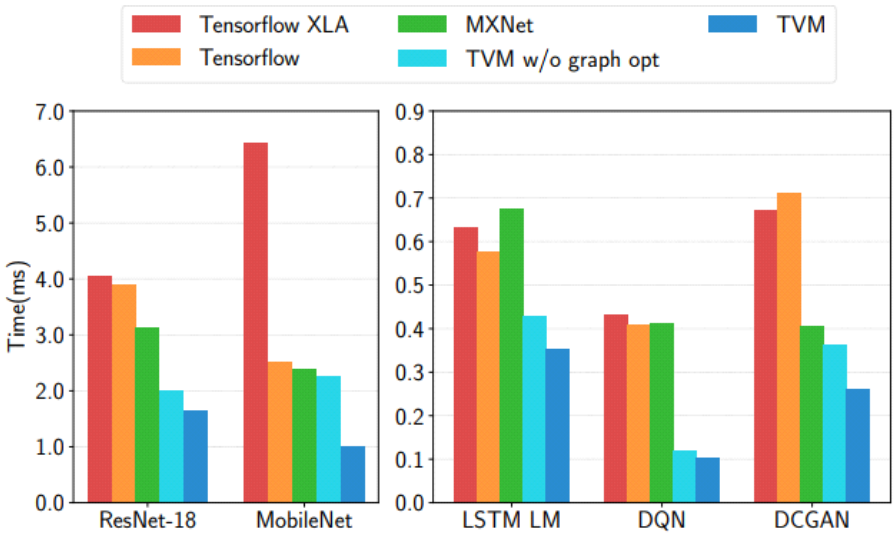


Figure 14: GPU end-to-end evaluation for TVM, MXNet, Tensorflow, and Tensorflow XLA. Tested on the NVIDIA Titan X.

  - To evaluate the operator level optimizations

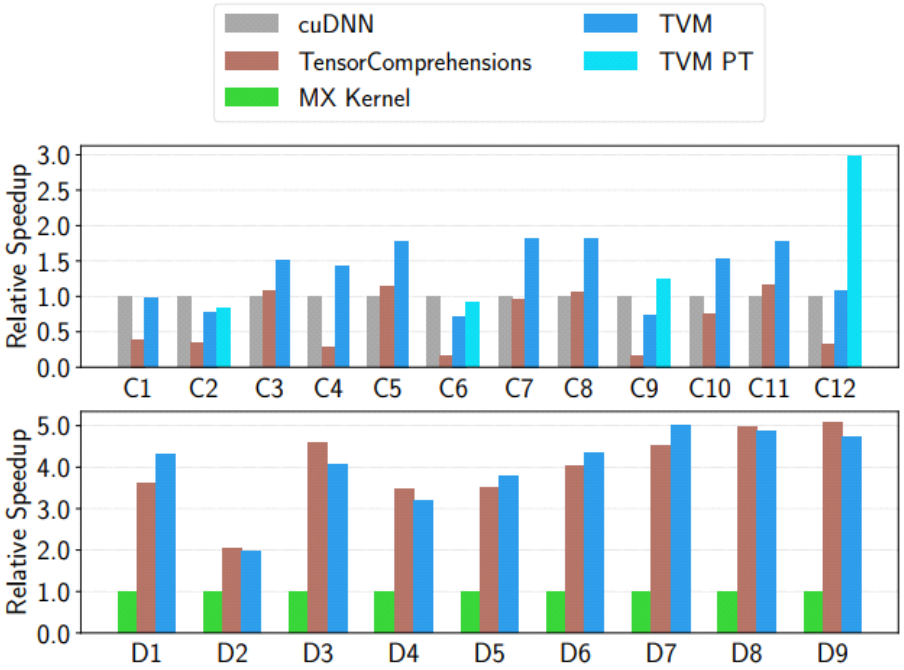


Figure 15: Relative speedup of all conv2d operators in ResNet-18 and all depthwise conv2d operators in MobileNet. Tested on a TITAN X. See Table 2 for operator configurations. We also include a weight pre-transformed Winograd [25] for 3x3 conv2d (TVM PT).

tvm安装配置

2019年10月23日 14:11

- 下载源码 git clone --recursive https://github.com/dmlc/tvm
- Centos cmake版本太旧
- 在容器内编译，容器外使用

- 安装编译环境依赖
  - sudo apt-get install -y python3 python3-dev python3-setuptools gcc libtinfo-dev zlib1g-dev build-essential cmake libedit-dev libxml2-dev
- 创建编译目录，复制并修改配置文件
  - mkdir build
  - cp cmake/config.cmake build
- 依赖LLVM。CPU代码生成必需。
- 需要LLVM 4.0以上版本。官网下载最新二进制版本安装
- 编译
  - Cd build
  - Cmake ..
  - Make -j4
- 安装python包
  - 直接使用环境变量让python在编译目录中查找编译好的包

- 安装依赖
  - apt-get install zlib1g-dev
  - apt install libtinfo-dev
  - apt install libxml2-dev
-

Tvm Quick start

2019年10月23日 14:24

- https://docs.tvm.ai/tutorials/relay_quick_start.html#sphx-glr-tutorials-relay-quick-start-py
- ```
  TVMError: Check failed: bf != nullptr: Target llvm is not enabled
  ```
- Notice that you need to build TVM with cuda and llvm enabled. 运行这个例子需要在编译时启用llvm。
- 编译tvm时llvm并不是必须项。但编译在CPU上运行的代码需要llvm。建议在编译时就安装并启用llvm
- docker容器中运行找不到GPU，原因待查
  ```

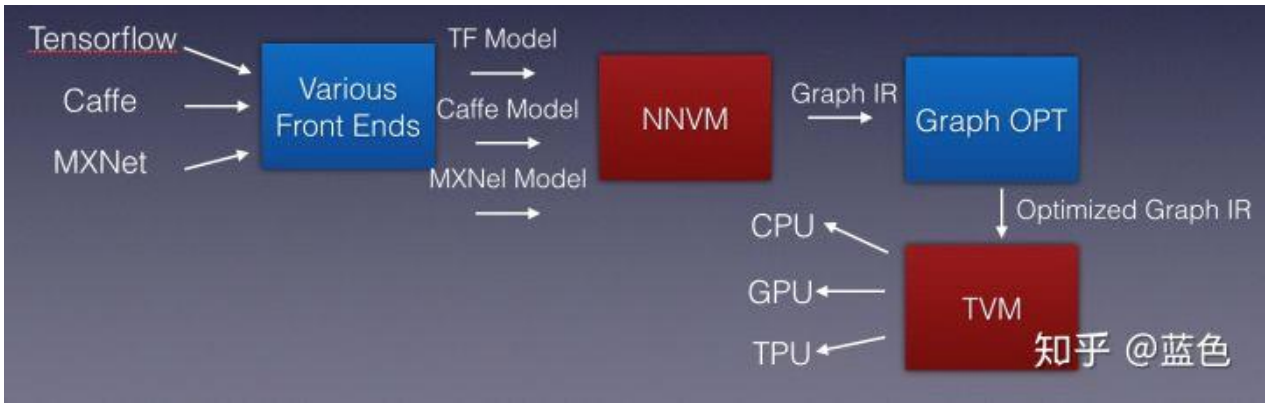
实验1

2019年11月5日 9:27

- https://zhuanlan.zhihu.com/p/60981270
- Onnx 版本太新运行时报错 （段错误） 需要配合1.5.0版本使用
  - pip install -U onnx==1.5.0
- Input_name需要改为"input.1"

Q&A

2019年11月6日 10:46

- 使用CPU target时，默认只使用一半的逻辑核心?
  - 源码中限制不使用超线程。
  - https://github.com/apache/incubator-tvm/blob/master/src/runtime/threading_backend.cc
  - 通过配置TVM_NUM_THREADS 指定线程数
  - 使用超线程有性能提升，但幅度很小
inception v1, 30 inferences	
n_cores	time(s)
1	15.87
2	8.019
4	4.218
8	3.923
- 运行时有很多warning:WARNING:autotvm:Cannot find config for target=llvm…
  - This warning is because you don't tune this workload of model
  - For x86 CPU: https://docs.tvm.ai/tutorials/autotvm/tune_nnvm_x86.html
- 使用tvm编译过的模型运行效率不如编译前的?
  - Inception v1, 30 inferences, W-2123, 8 CPU cores
  - Tvm: 3.9235s, native tensorflow: 3.9050s
  - 可能是由于没有autotune，即模型只加载并编译，但没优化过
  - 需要使用autotvm模块对模型进行优化(tuning)
  - 但是在build_config时又指定了opt_level?
- nnvm与tvm的关系?
  - 

  - NNVM是图编译器，输入是各种DL框架的模型，输出是优化过后的图的中间表示
  - TVM以优化后的图为输入，输出能在不同硬件上运行的模型
- 官方Tunig with CPU例程(https://docs.tvm.ai/tutorials/autotvm/tune_nnvm_x86.html)中，输出结果中task个数为什么是12
  - Resnet18有12个卷积层, 对每一层分别进行优化
- Tensorflow xla的使用? 是否默认使用了xla优化?

    - xla是tensorflow的图编译器，只需要改动极少的源代码，就可加速tensorflow模型。
    - 需要在代码中显式调用
- Tensorflow xla与tvm对比?
    - ？tvm主要用于加速推理，xla可以加速训练