TVM @ Xilinx

Elliott Delaye
Distinguished Engineer
Dec 5th, 2019
TVM Target devices and models

HW Platforms

- ZCU102
- PYNQ
- Ultra96

Models

- Face detection
- Pose estimation
- Video analytics
- Lane detection
- Object detection
- Segmentation
Interested in designing ASIC & FPGA for AI? Design engineer positions are available at Facebook in Menlo Park.

I used to be a chip designer many moons ago: my engineering diploma was in Electrical...
Large-Scale FPGA-based Convolutional Networks

Clément Farabet¹, Yann LeCun¹, Koray Kavukcuoglu¹, Eugenio Culurciello², Berin Martini², Polina Akselrod², Selcuk Talay²

Machine Learning on Very Large Data Sets, Cambridge University Press, 2011.
Synchronous Dataflow (SDF) vs Matrix of Processing Elements (MPE)

End points are pure layer-by-layer compute and feed-forward dataflow architecture

\[ WC_{memory} = SUM(Wi) \]

\[ AT_{memory} = \text{Sum}(ATi) \]
\[ ATi = \text{batch} \times FMi_{DIM} \times CHi \times (Ki + Si) \]

\[ WC_{memory} = MAX(Wi) \]
\[ AT_{memory} = 2 \times \text{batch} \times \text{Max}(FMi_{DIM} \times FMi_{DIM} \times CHi) \]

Lin, X., Yin, S., Tu, F., Liu, L., Li, X. and Wei, S. LCP: a layer clusters paralleling mapping method for accelerating inception and residual networks on FPGA. DAC’2016
Synchronous Dataflow (SDF) vs Matrix of Processing Elements (MPE)

Xilinx Research Labs

Alibaba iDST Resnet-18 @ HotChips 30 (2018)
Inference Flow

Deep Learning Frameworks

- mxnet
- TensorFlow
- Caffe

Frontend

- Framework Tensor Graph to Xilinx Tensor Graph
- Tensor Graph Optimization

Compiler

Quantizer

Image

Runtime

- CPU Layers
- FPGA Layers

Model Weights

Calibration Set

https://github.com/xilinx/vitis-ai
Inference Flow

Deep Learning Frameworks
- MxNet
- TensorFlow
- Caffe

Frontend
- Framework Tensor Graph to Xilinx Tensor Graph
- Tensor Graph Optimization

Compiler

Quantizer
- Model Weights
- Calibration Set

Image

Runtime
- CPU Layers
- FPGA Layers

https://github.com/xilinx/vitis-ai
TVM as Unified ML Front End

```
@relay.transform.module_pass(opt_level=4)
class AccelModule:
...
```
TVM as Unified ML Front End

Why not use Relay directly?

ONNX Runtime (MSFT)
ARM-NN (ARM)
MNN (Alibaba)
MLIR (Google)

@relay.transform.module_pass(opt_level=4)
class AccelModule:
...

Compiler
Quantizer
Partitioner
TVM Partitioning

- More than supported/not supported, pattern matching graph colorization
- Choices how to partition especially for multi-branch networks (i.e. YOLOv3, SSD)
TVM Graph Partitioning/Fusion

Pre-Processing → Subgraph 1 → Parallel Subgraphs

CPU → FPGA → CPU

Post-Processing
TVM Code Generation

Parallel Subgraphs

CPU Pre-Processing

FPGA

Subgraph 1

CPU

FPGA

Parallel Subgraphs

CPU Post-Processing

```
tvm_fpga_cpu/
|-- README.md
|-- fpga
|   |-- tvm_compiler.json
|   |-- tvm_quantizer.json
|   `-- weights_data.h5
|-- tvm_fpga_cpu.json
|-- tvm_fpga_cpu.params
|  `-- tvm_fpga_cpu.so
```
Example of FPGA node in TVM graph

```
{
  "nodes": [
    {
      "op": "null",
      "name": "data",
      "inputs": []
    },
    {
      "op": "tvm_op",
      "name": "accel_0",
      "attrs": {
        "flatten_data": "0",
        "func_name": "accel_fused",
        "num_inputs": "1",
        "num_outputs": "1"
      },
      "inputs": [[0, 0, 0]]
    },
    {
      "op": "tvm_op",
      "name": "flatten0",
      "attrs": {
        "flatten_data": "0",
        "func_name": "fuse_flatten",
        "num_inputs": "1",
        "num_outputs": "1"
      },
      "inputs": [[1, 0, 0]]
    }
  ]
}
```

Calls Xilinx’s tvm.extern registered function to access the FPGA runtime APIs
Registering TVM op in Python at runtime

File contrib_xlnx.py:

```python
...  
@tvm.register_func("tvm.accel.accel_fused")
def accel_fused(graph_path, output_layout, out, *ins):
    path = c_char_p(graph_path.value).value
    layout = c_char_p(output_layout.value).value
    # Calls Xilinx Python APIs to run subgraph on input data
    # C++ APIs possible but requires compiling with TVM
    ...
    n2cube.dpuSetInputTensorInHWCFP32(task, input_name, X, len(X))
n2cube.dpuRunTask(task)
n2cube.dpuGetTensorData(address, value, size)
...```

>> 16
Compute Pipelines for Heterogenous Systems

- Max throughput when all compute elements are running in parallel
- Performance results based on Xilinx own runtime pipeline available in github
  - (https://github.com/Xilinx/ml-suite/blob/master/examples/deployment_modes/mp_classify.py)
  - Streamlined multi-process pipeline using shared memory
  - Usually need >4 Pre-Process cores running to keep up with FPGA

- TVM pipeline needed. CPU/FPGA (even GPU!) partitions ideally run in parallel
  - Xilinx ZU7EV = FPGA + (ARM Cortex-A53)+(ARM Cortex-R5)+(ARM Mali-400 MP2)
  - Potentially useful by all accelerator platforms, not just FPGA
  - Xilinx looking forward to working with others who are also interested in this
Special thanks to:
Jorn Tuyls
Ehsan Ghasemi

e-mail: elliott@xilinx.com
Adaptable.
Intelligent.