TVM@AliOS

PRESENTATION
AGENDA

- TVM @ AliOS Overview
- TVM @ AliOS ARM CPU
- TVM @ AliOS Hexagon DSP
- TVM @ AliOS Intel GPU
- Misc

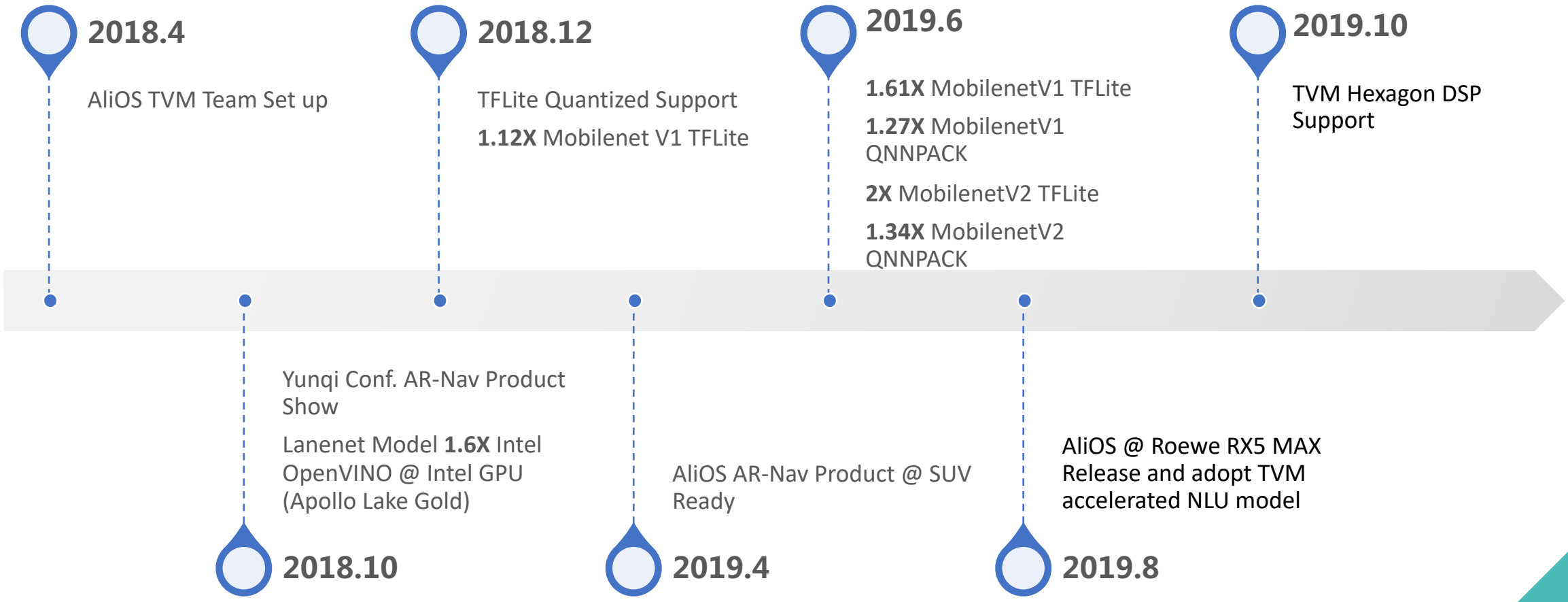
PART ONE
TVM @ AliOS Overview

AliOS Overview

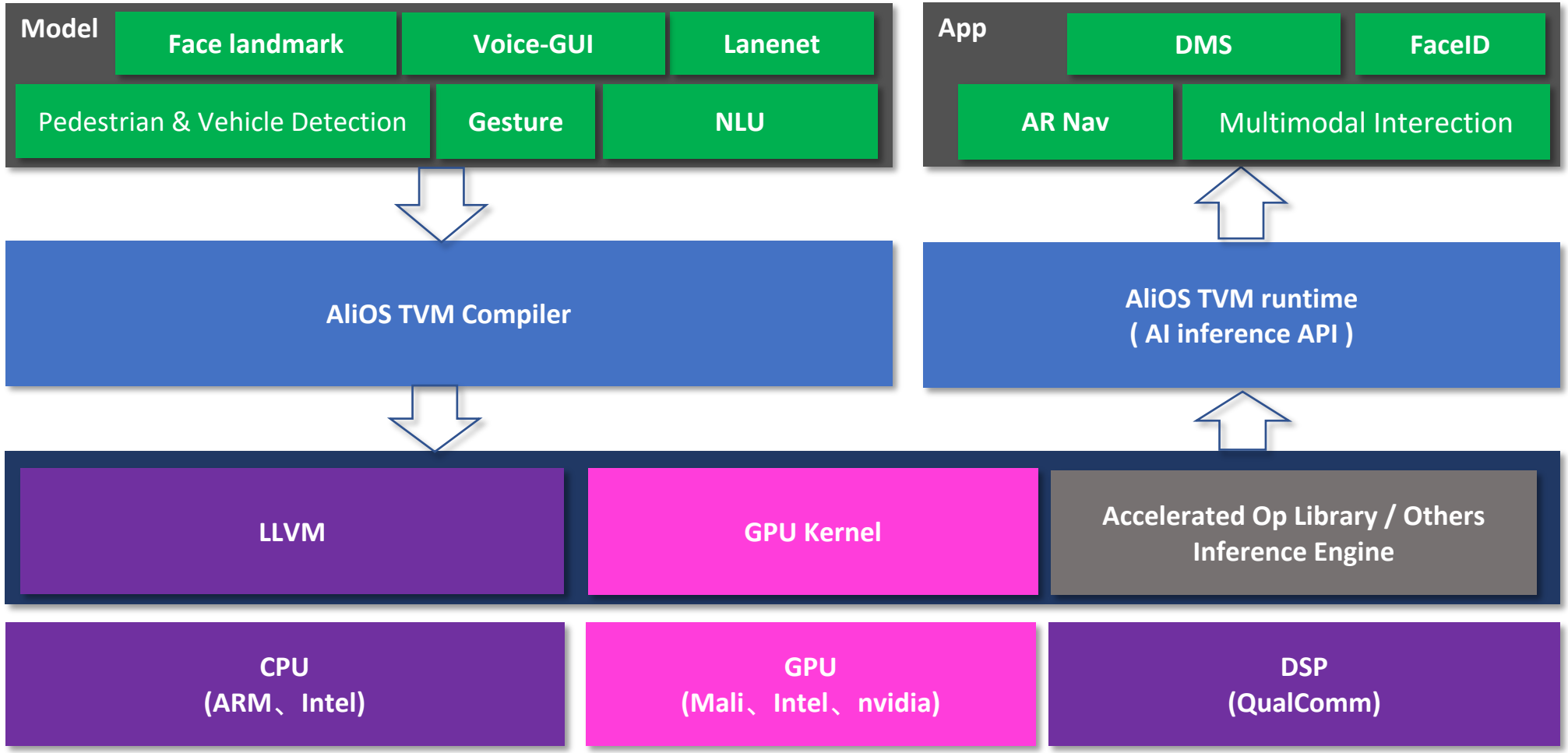
- AliOS (www.alios.cn) is a newly designed to drive everything toward intelligence. The AliOS is running in vehicles, Phone, Pad and IoT terminals. Provide cloud and devices co-related solutions for different terminals.

- To help traditional car firms embrace new ‘connected’ era by acting as the IT ‘chassis’ of auto industry

TVM Timeline @ AliOS



AliOS TVM Arch



PART TWO
AliOS TVM @ ARM CPU

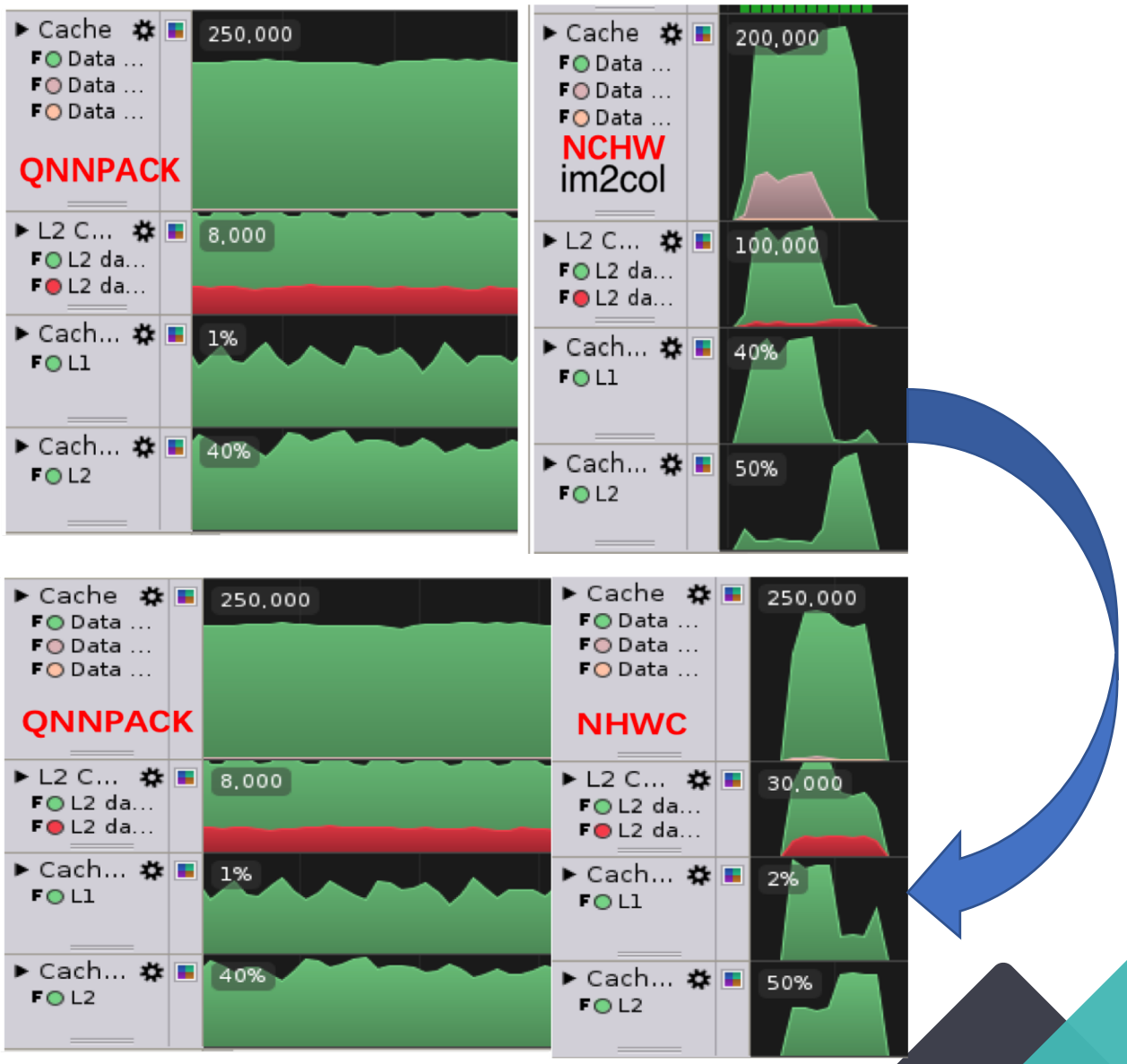
AliOS TVM@ARM CPU

- Support TFLite（Open Source and Upstream Master）
- Optimize on INT8 & FP32

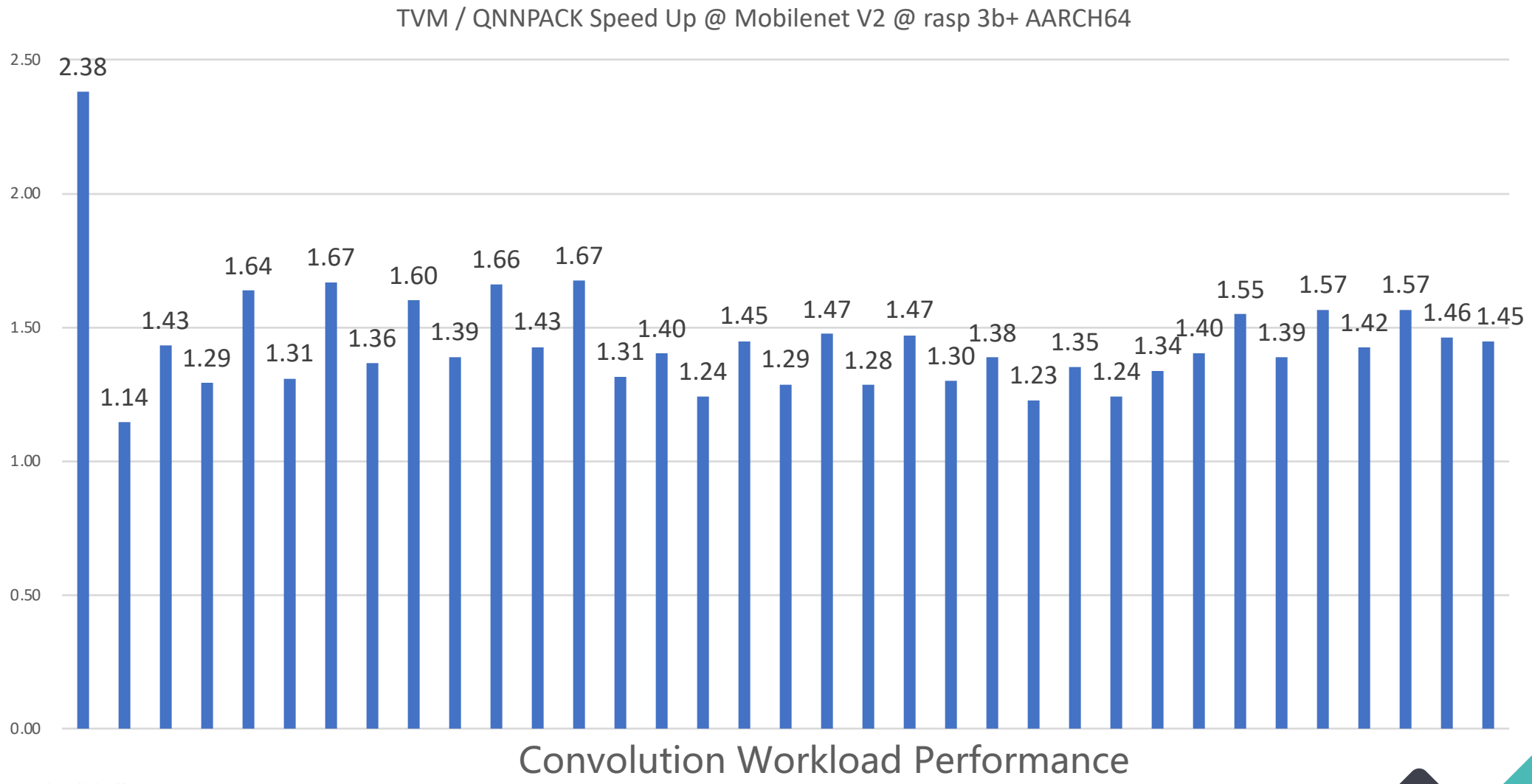
AliOS TVM @ ARM CPU INT8

Convolution

- NHWC layout
- im2col + pack
- Tensorize GEMM



AliOS TVM @ ARM CPU INT8



AliOS TVM @ ARM CPU INT8

Depthwise Convolution

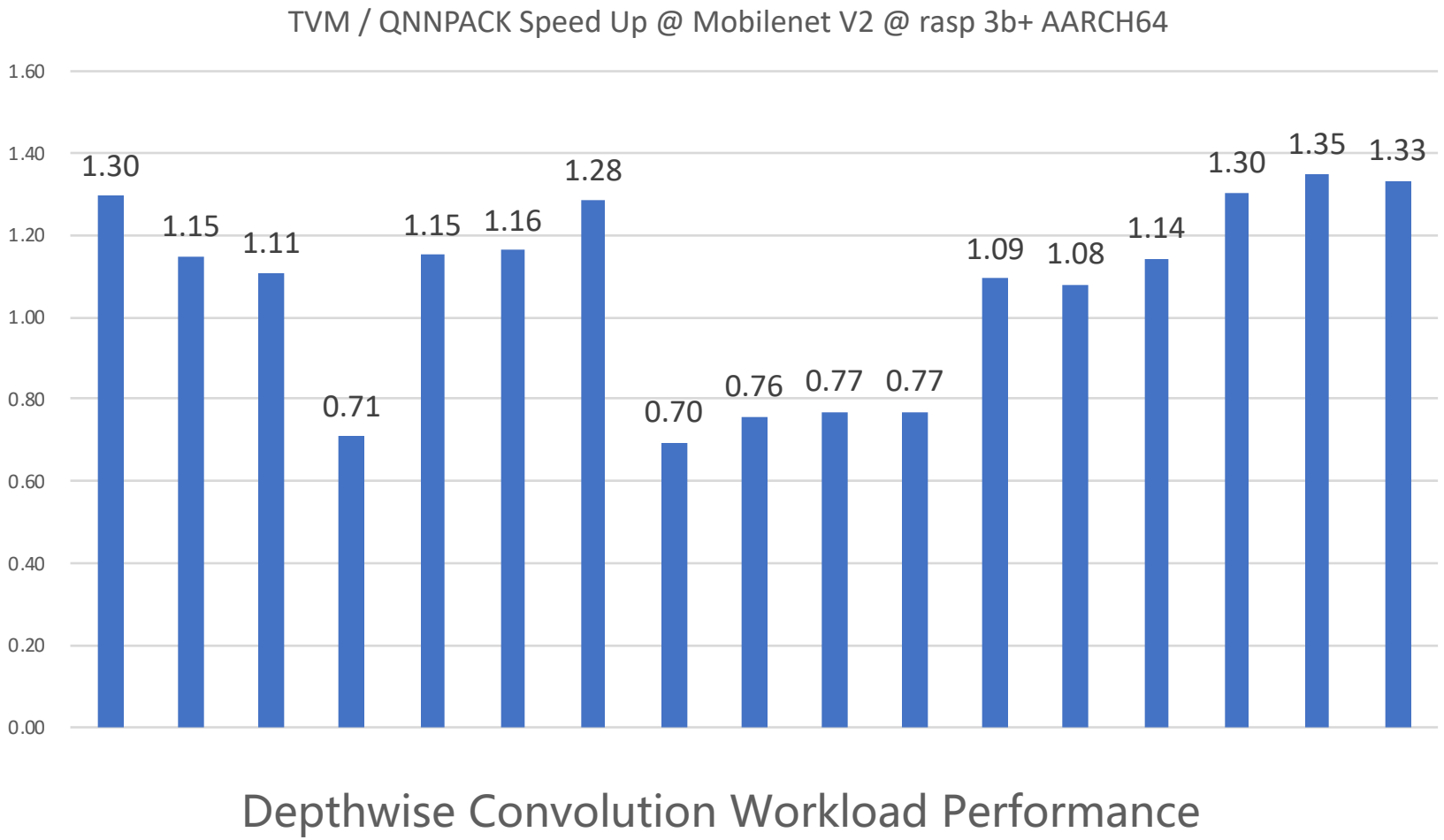
- NHWC layout

- Using TVM schedule primitive completely, no tensorize

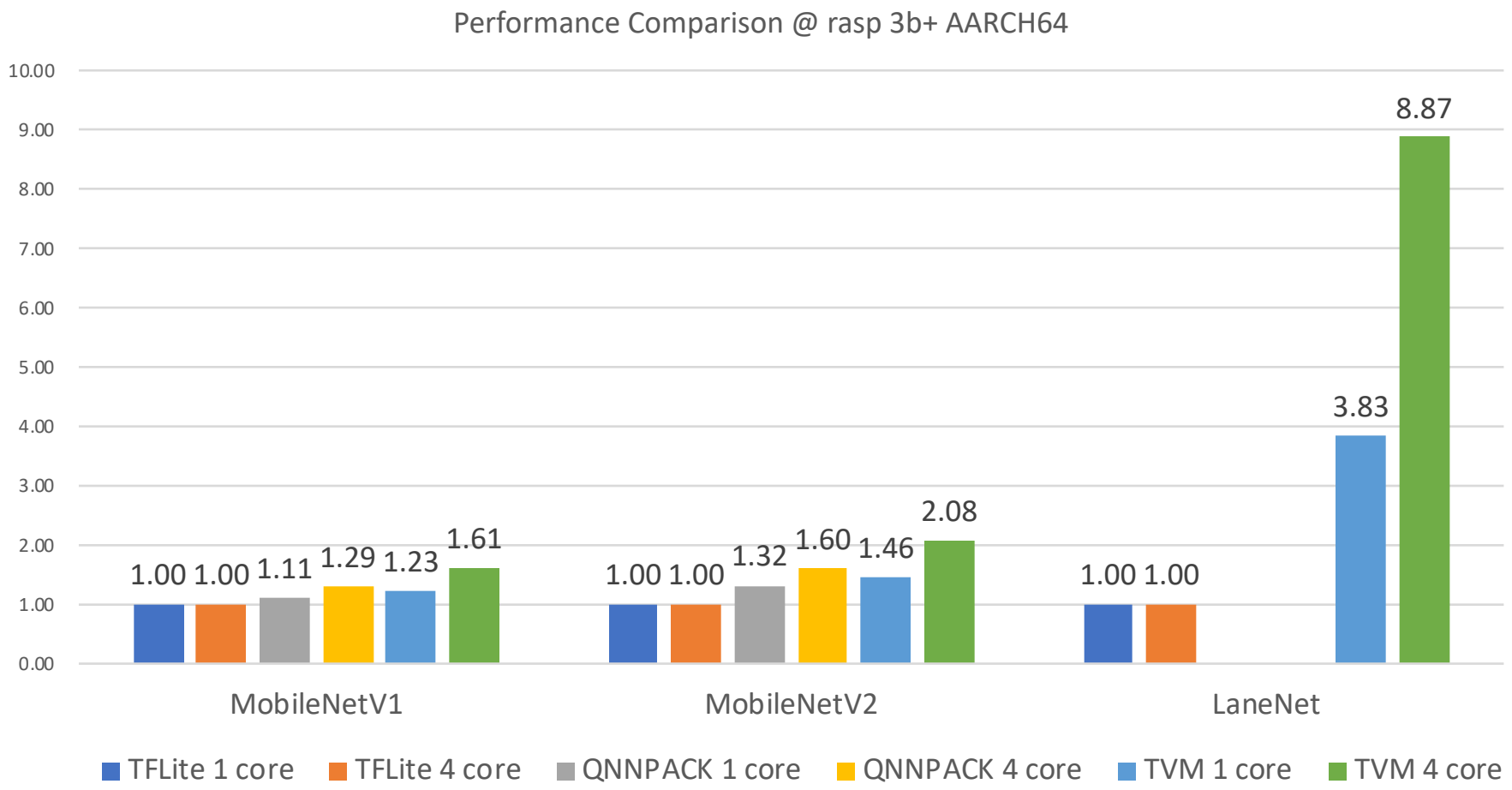
- Some Experience:

1. Avoid DataPack
2. Generate SMLAL instruction if your ARM does not have dot
3. compute_at is very important

AliOS TVM @ ARM CPU INT8



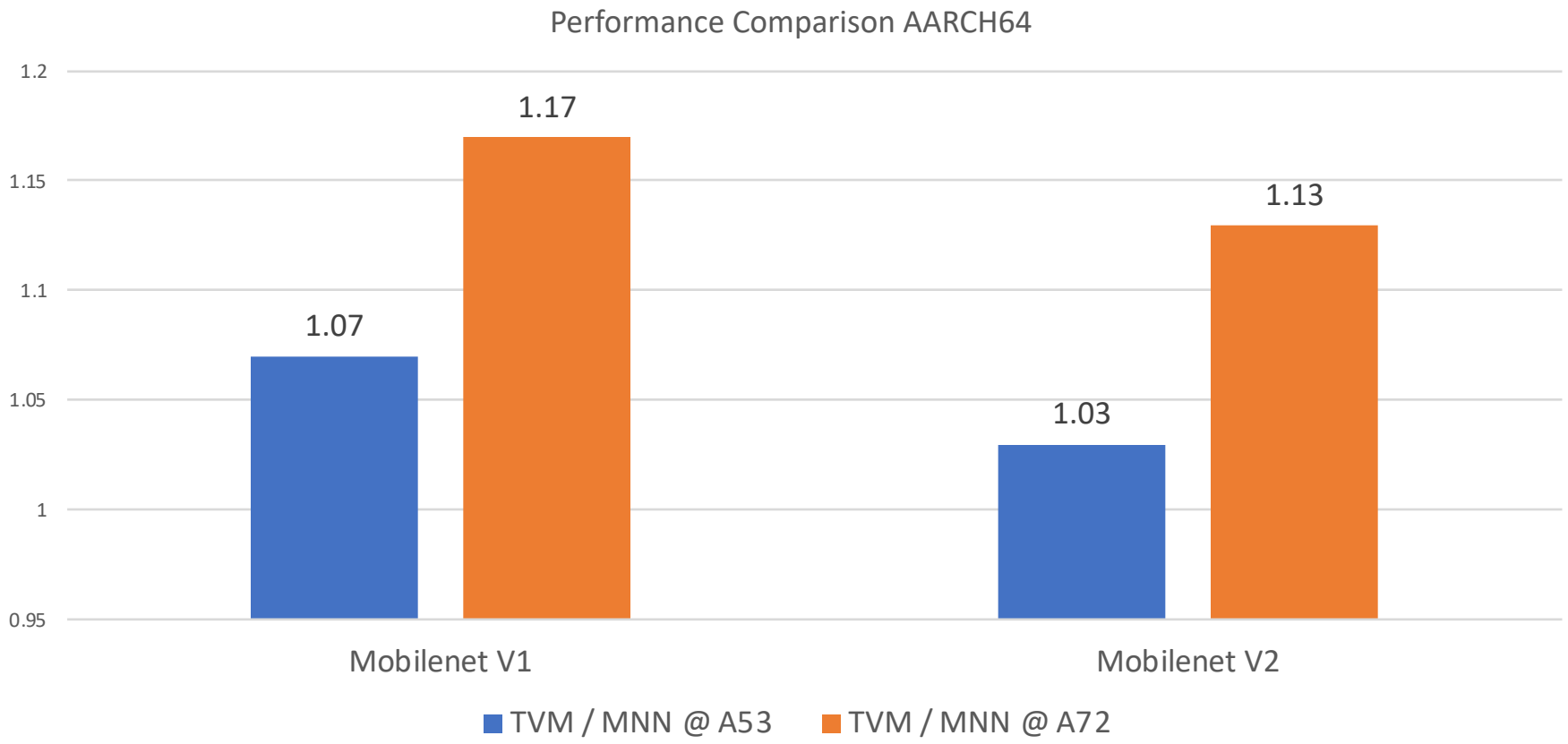
AliOS TVM @ ARM CPU INT8



AliOS TVM @ ARM CPU FP32

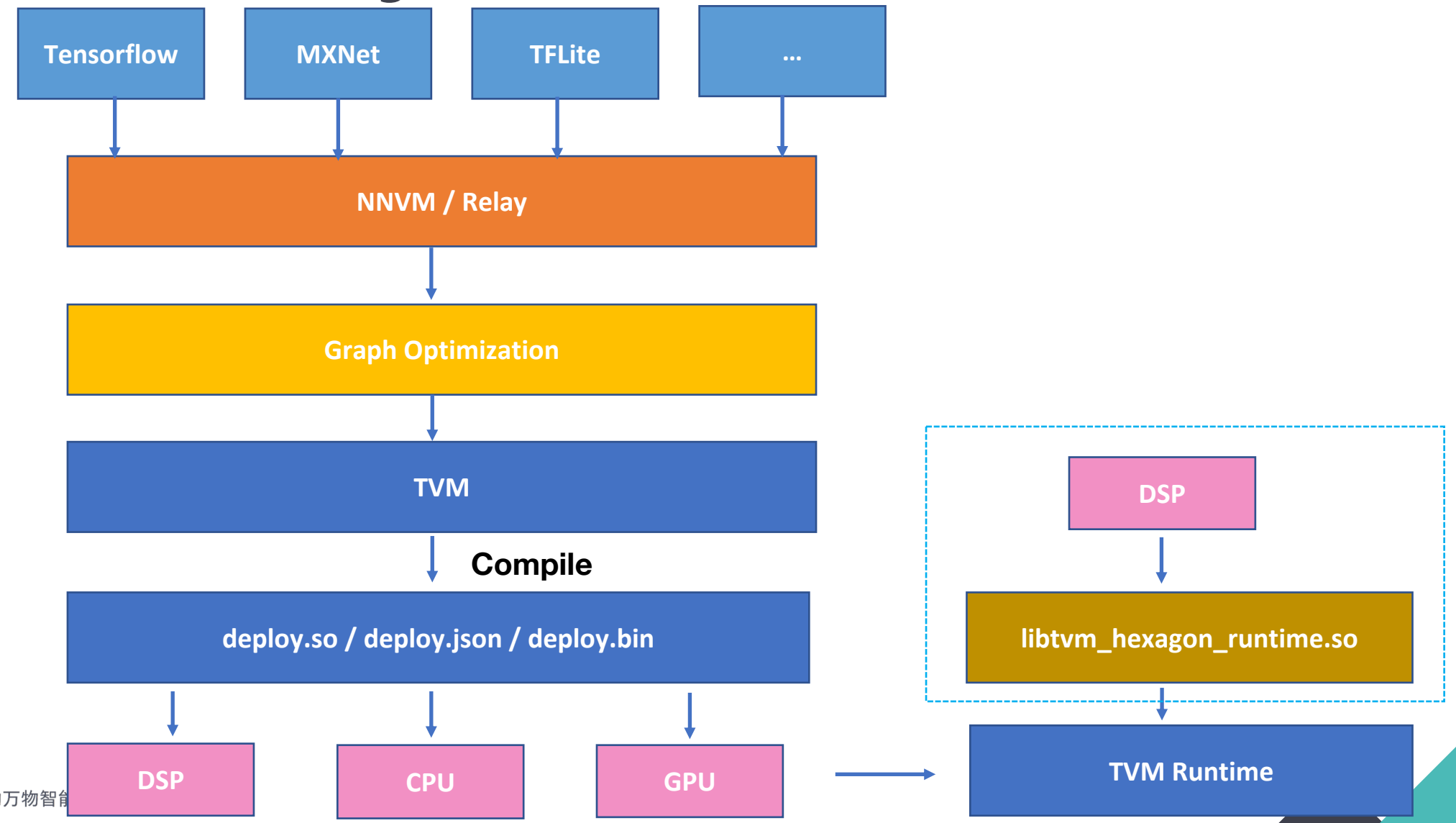
- NHWC layout
- For pointwise convolution, we implement im2col schedule
- No tensorize, but in schedule to cooperate with LLVM to simulate GEMM microkernel

AliOS TVM @ ARM CPU FP32



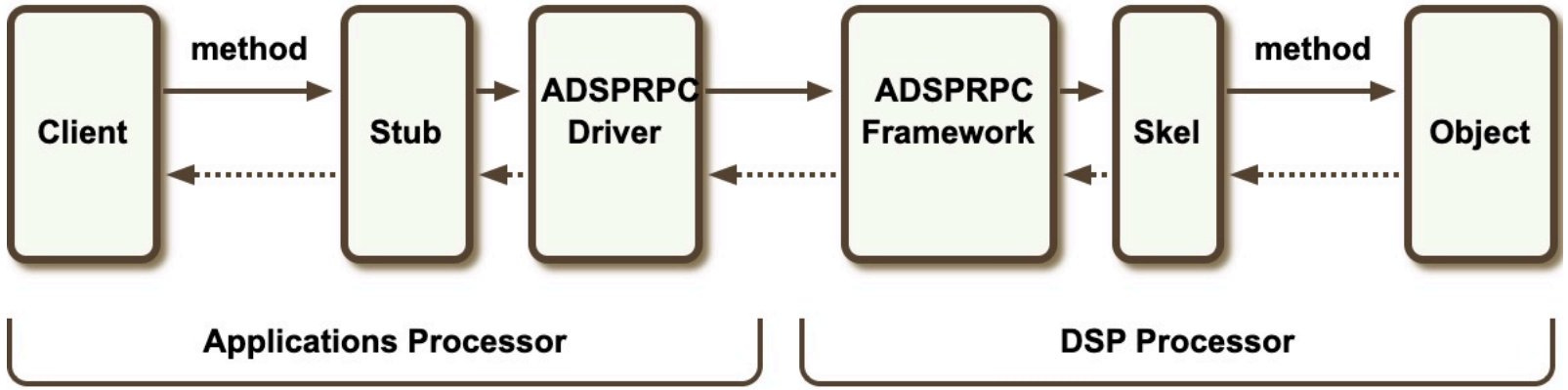
PART THREE
AliOS TVM @ Hexagon DSP

AliOS TVM @ Hexagon DSP



AliOS TVM @ Hexagon DSP

- Compute Kernel Offload to DSP， loop nests marked as pipeline
- Implement complete Hexagon runtime based on community PR.



AliOS TVM @ Hexagon DSP

- Add Hexagon Code Generator inherits LLVM and could generate HVX instruction
- Add one Hexagon runtimes named as libtvm_hexagon_runtime.so to support parallel.
- Could run end-to-end TFLite Mobilenet V2 quantized model on Simulator / Device.

AliOS TVM @ Hexagon DSP

- Performance is our focus next. We begin to do some work now. Such as writing Tensorize to generate vrmpy instruction when we meet GEMM.

```
vec_a = tvm.call_pure_intrin('uint8x128', 'reinterpret', vec_ai32)
vec_b = ins[1].vload([0, 0], "uint8x128")
vec_c = tvm.const(0, 'int32x32')

vdot = tvm.call_llvm_intrin('int32x32',
                            'llvm.hexagon.V6.vrmpyubv.128B',
                            tvm.const(0, 'int32'),
                            vec_a, vec_b)
if index == 0:
    ib.emit(outs[0].vstore(0, vdot))
else:
    ib.emit(outs[0].vstore(0, vdot + outs[0].vload([0], 'int32x32')))
return ib.get()
```

AliOS TVM @ Hexagon DSP

For 128 x 32 x 4 GEMM:

Before:

```
{  v9.w = vmpyio(v5.w,v0.h) }
{  v10.w = vmpyio(v7.w,v1.h)
   v8.w = vasl(v14.w,r1) }
{  v6.w += vmpyie(v12.w,v3.uh)
   nop }  :endloop0
{  v8.w += vmpyie(v4.w,v2.uh)
   v3.w = vasl(v10.w,r1) }
{  v3.w += vmpyie(v7.w,v1.uh)
   v30.w = vasl(v9.w,r1)
   v4.w = vadd(v8.w,v6.w) }
{  v30.w += vmpyie(v5.w,v0.uh)
   v31.w = vadd(v3.w,v4.w) }
{  v0.w = vadd(v30.w,v31.w)
   vmem(r0++#1) = v0.new }
{  r0 = #0;          jumpr r31 }
```

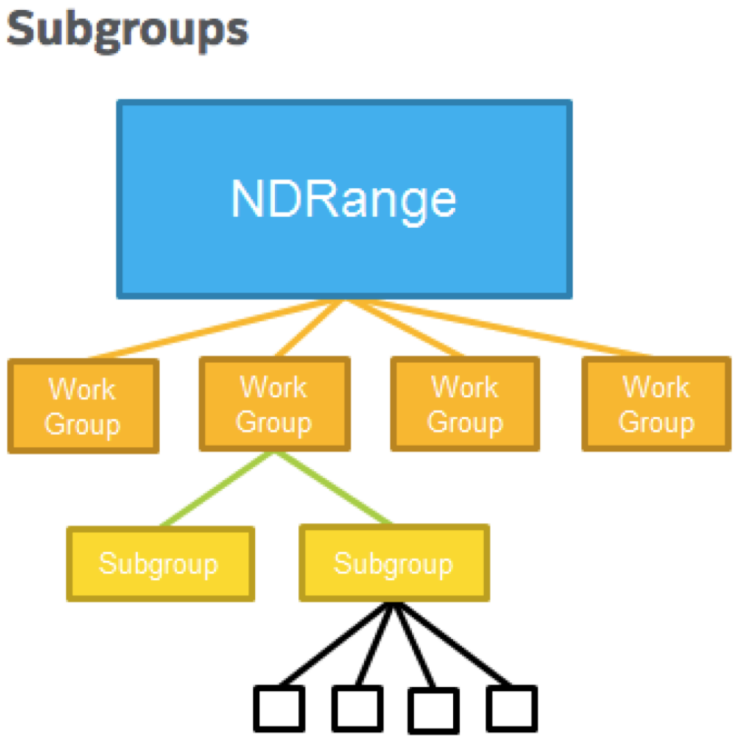
After:

```
{  v3.uw = vrmpy(v2.ub,v0.ub)
   v1 = vsplat(r2)
   v2 = v1
   vmem(r0++#1) = v3.new }
{  nop
   r2 = memw(r1++#4) }  :endloop0
{  v2.uw = vrmpy(v2.ub,v0.ub)
   v30 = vsplat(r2)
   vmem(r0++#1) = v2.new }
{  v1.uw = vrmpy(v1.ub,v0.ub)
   vmem(r0++#1) = v1.new }
{  v31.uw = vrmpy(v30.ub,v0.ub)
   vmem(r0++#1) = v31.new }
{  r0 = #0;          jumpr r31 }
```

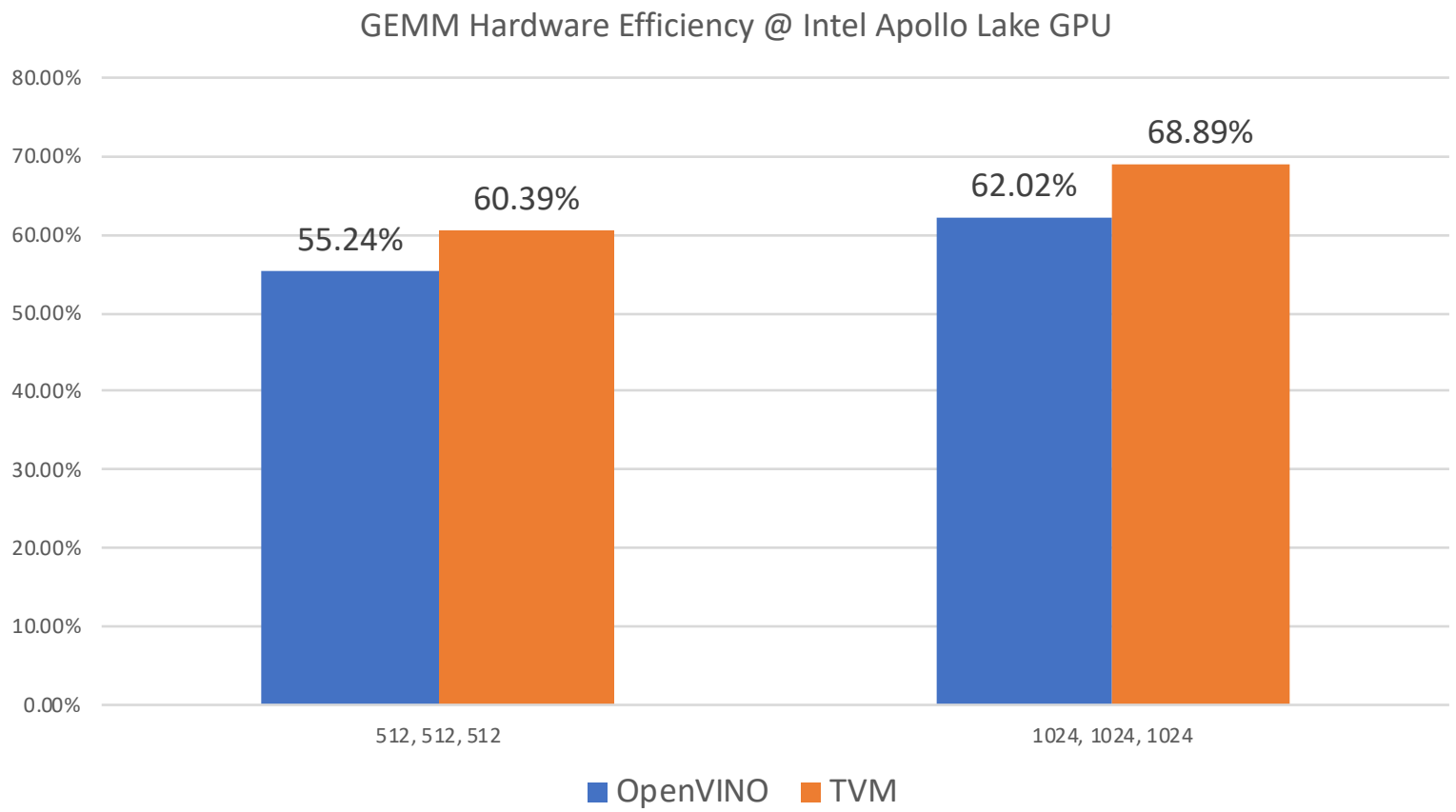
PART FOUR
AliOS TVM @ Intel GPU

AliOS TVM @ Intel GPU

- Implement the schedule from scratch
- Leverage Intel Subgroup Extension



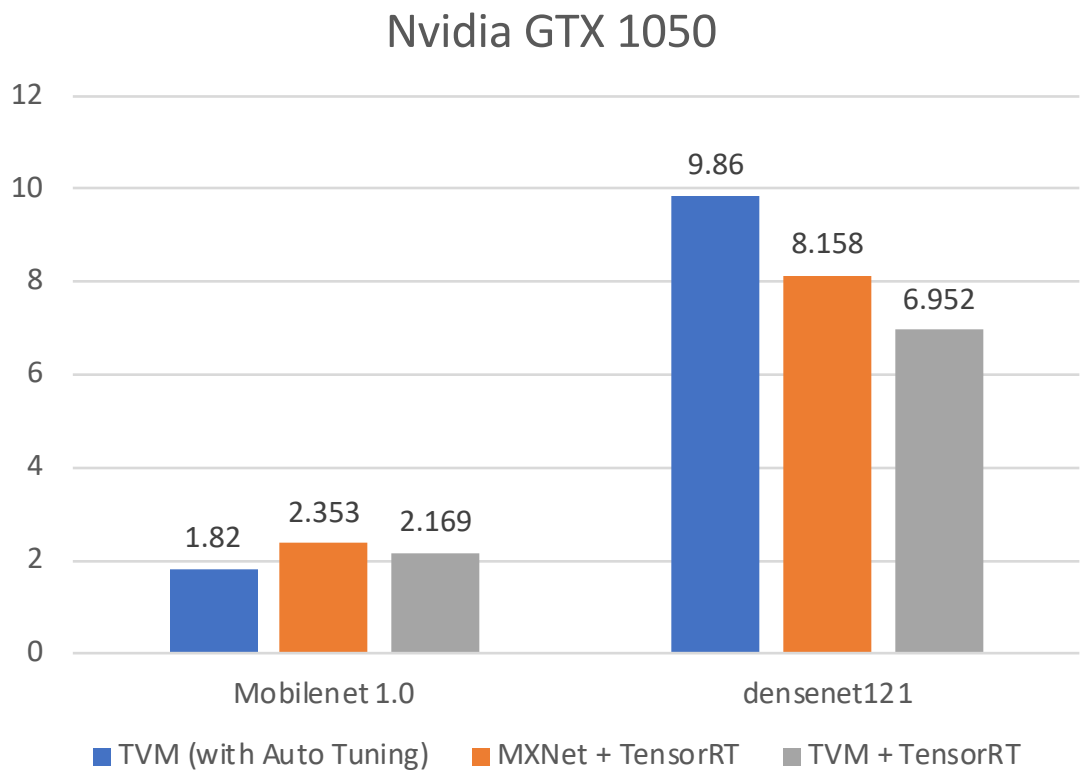
AliOS TVM @ Intel GPU



PART Five
Misc

Misc

- Integrate other inference engine (like TRT)
- Online Service
- C++ RPC (Merged into Master)



THANKS

AliOS | 驱动万物智能