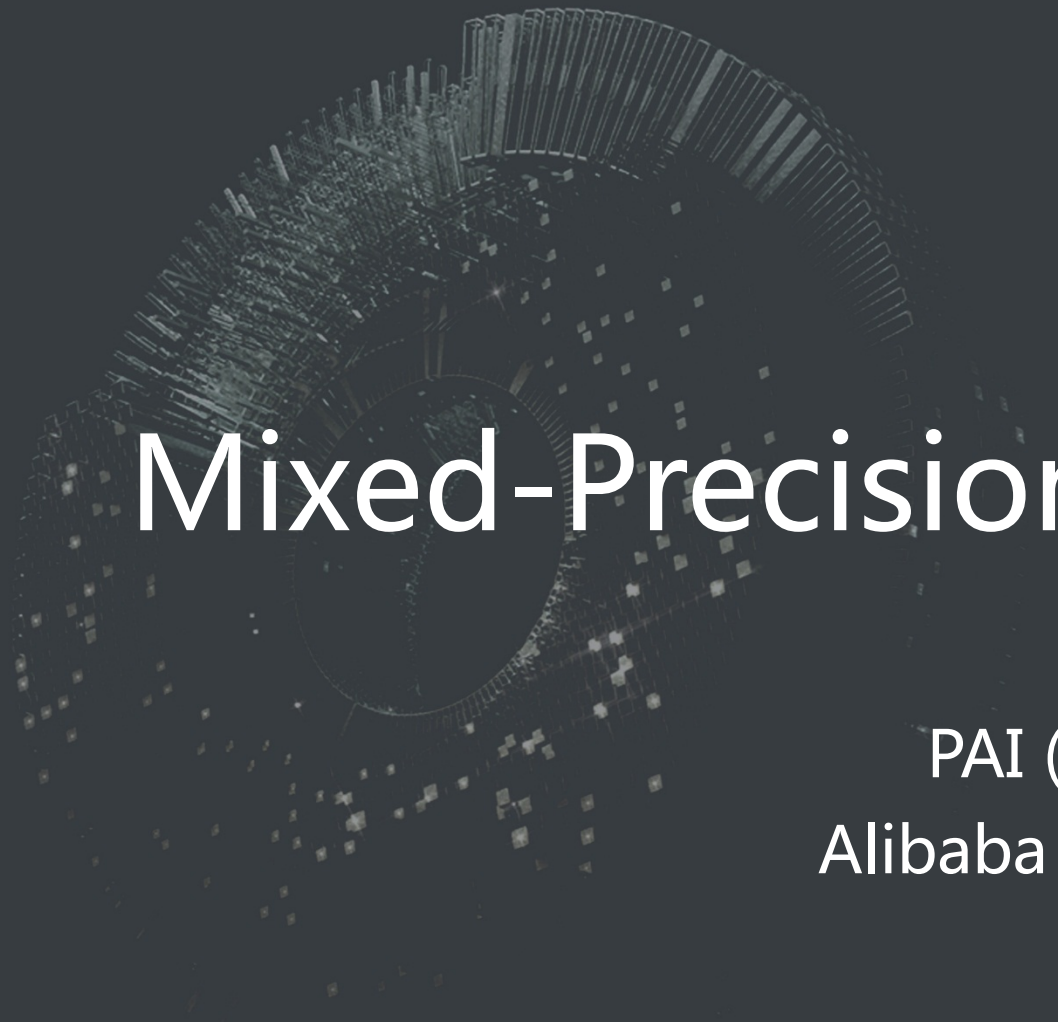
计算平台事业部
COMPUTING PLATFORM

TensorCore AutoCodeGen and Mixed-Precision Training/Inference



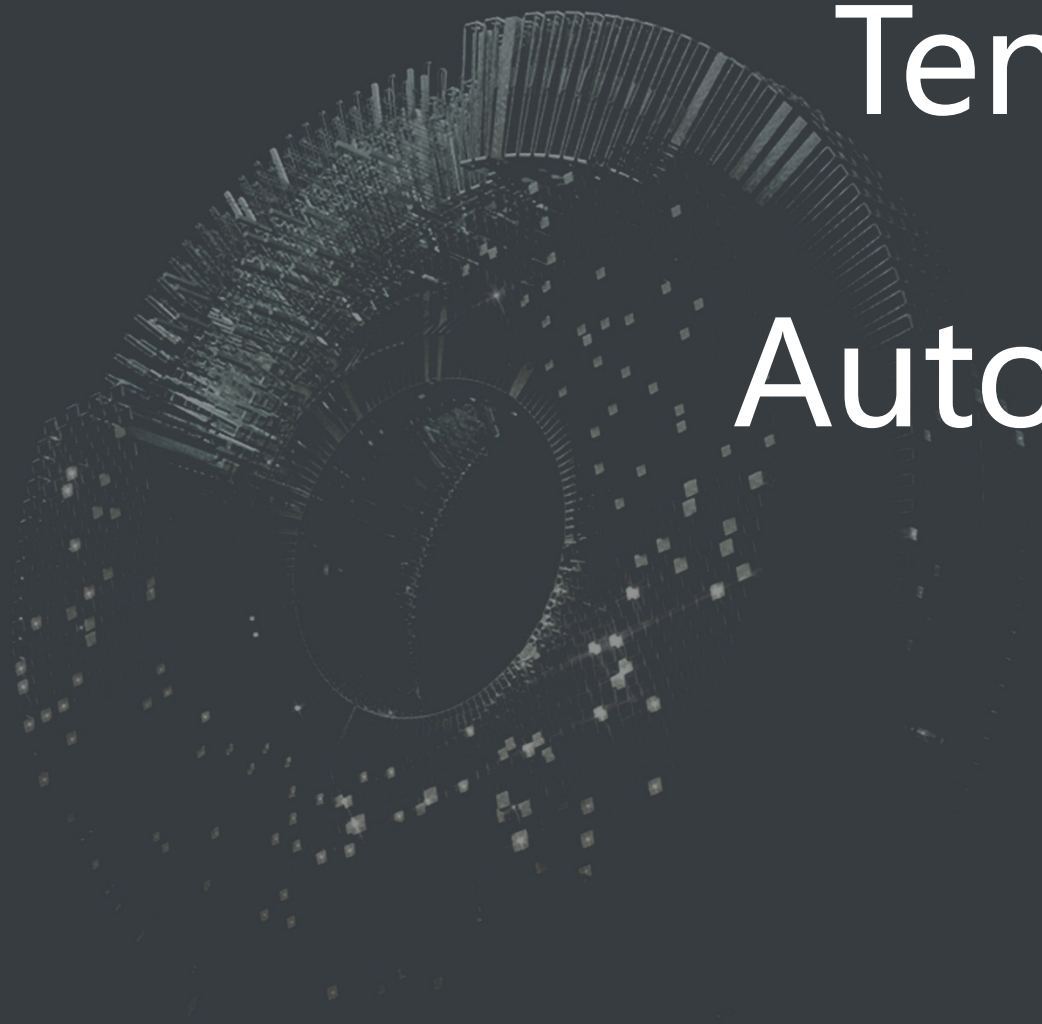
PAI (Platform of AI)
Alibaba Cloud Intelligence

Outline

计算平台事业部
COMPUTING PLATFORM

- TensorCore AutoCodeGen in TVM
- FP16 Mixed-Precision Training on PAI
- INT8 Inference on PAI-Blade

计算平台事业部
COMPUTING PLATFORM

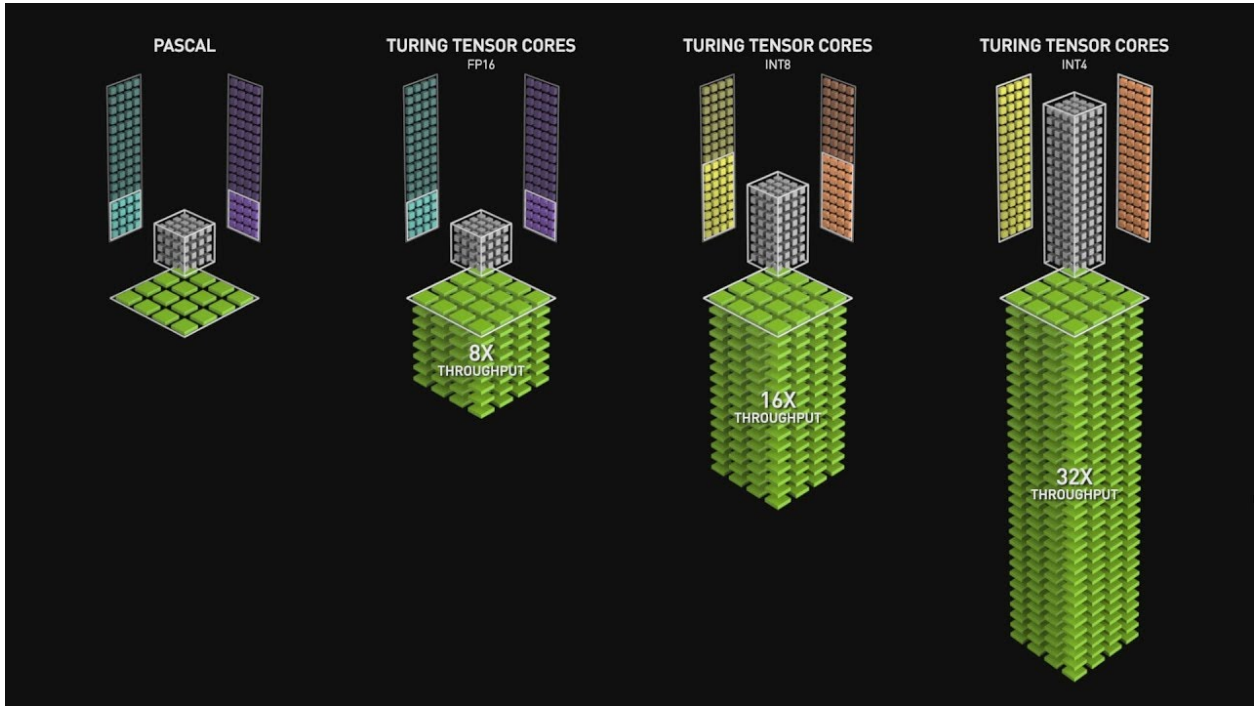


TensorCore AutoCodeGen

Background

计算平台事业部
COMPUTING PLATFORM

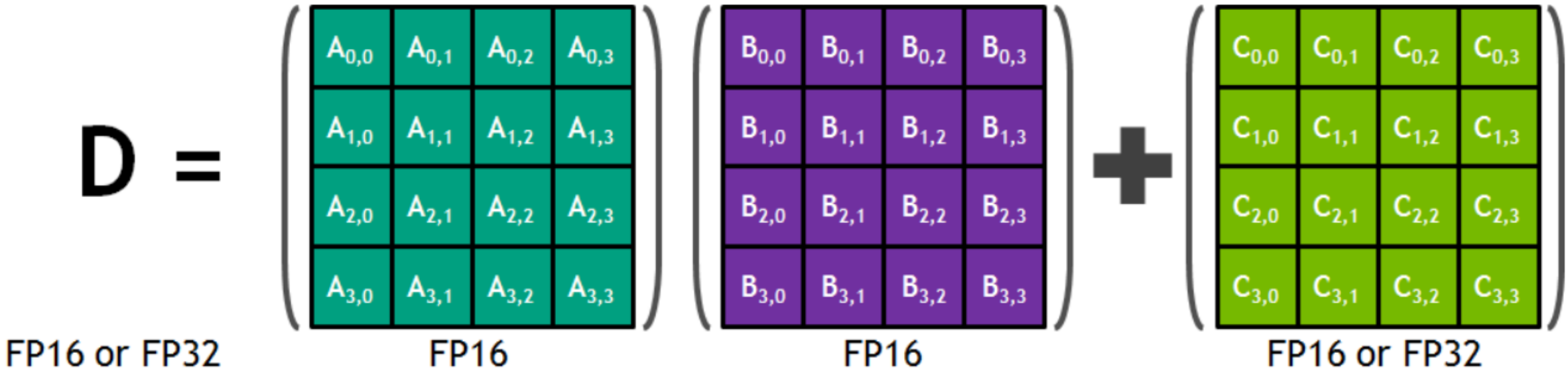
- TensorCore
  - A revolutionary technology that delivers groundbreaking AI performance.
  - Performs *mixed-precision* matrix multiply and accumulate in a single operation.



Background

计算平台事业部
COMPUTING PLATFORM

- TensorCore
  - *Programmable* matrix-multiply-and-accumulate units
  - *Warp-level* matrix operations exposed in the CUDA *WMMA API*



Background

计算平台事业部
COMPUTING PLATFORM

CUDA C	WMMA API
half a[16x8]	wmma::fragment<wmma::matrix_a, m, n, k, half,wmma::col_major> a[1]
a = A[index]	wmma::load_matrix_sync(a, &A[index], stride)
c = float(a)*float(b) + c	wmma::mma_sync(c, a, b, c)
C[index] = c	wmma::store_matrix_sync(&C[index], c, stride, nvcuda::wmma::mem_col_major)

Background

计算平台事业部
COMPUTING PLATFORM

- TVM TensorCore Intrinsics
  - Authored by @Hzfengsy
  - Intrinsics: tvm_load_matrix_sync, tvm_mma_sync ...
  - New Memory Scopes: wmma.matrix_a/b, accumulator
  - Tensorization on warp level schedule

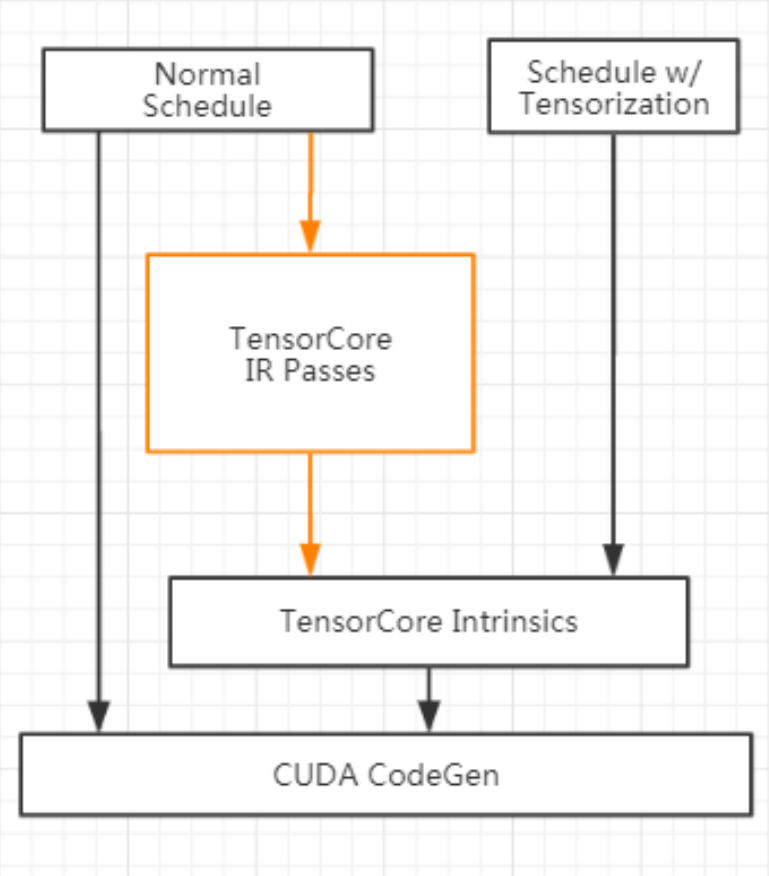
Motivation

- The overhead of writing warp-level schedule for TensorCore
  - Work at the scheduling level: the less the better
  - The requirement of familiarity with WMMA API
- Unified matmul schedule for GPU
  - Maintainability & Common Optimization Sharing
  - Search across the entire space (TensorCore + non-TensorCore)

Our Solution

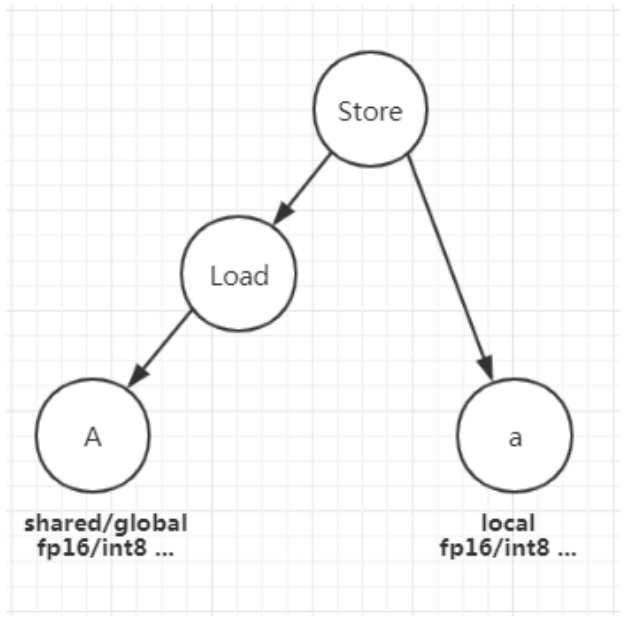
计算平台事业部 COMPUTING PLATFORM

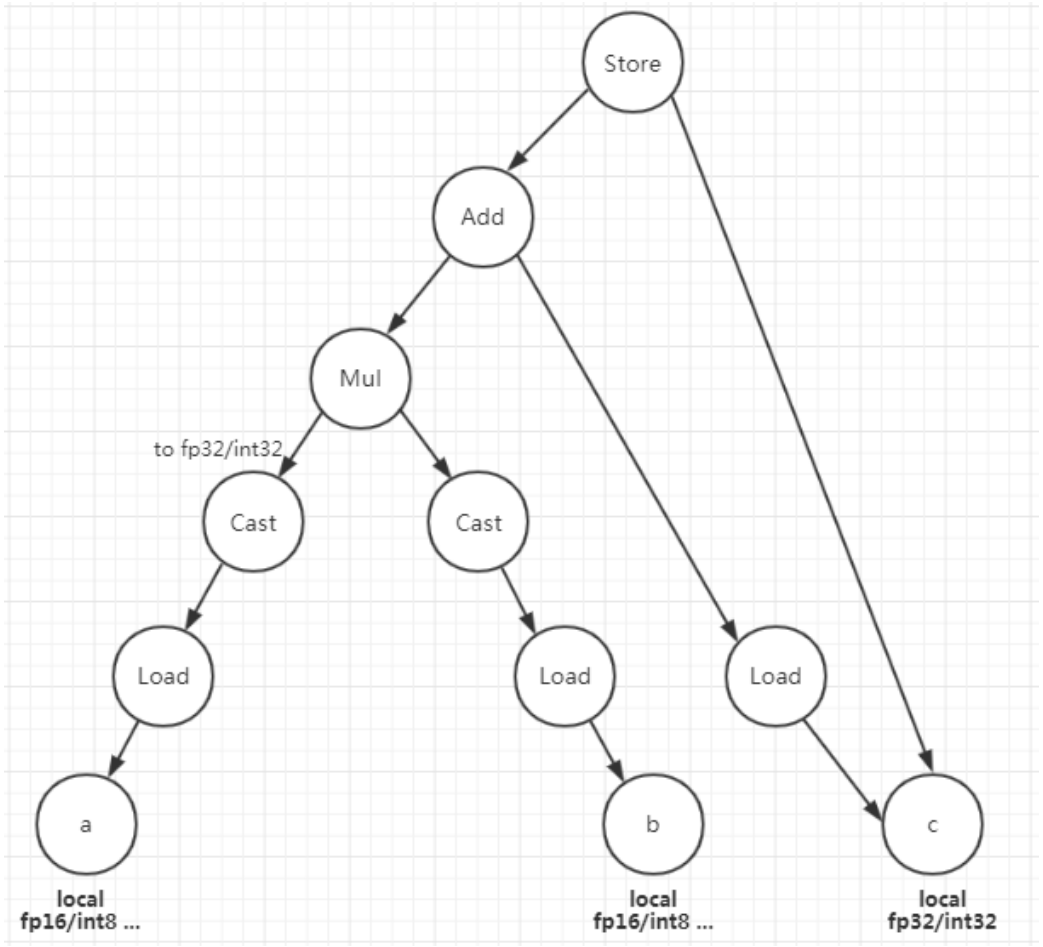
- Generate TensorCore code directly from normal thread-level schedule
  - Normal schedule: the schedule for CUDA codegen
    - Need to satisfy warp tile requirements (16x16x16 …)
  - Kind of Auto Tensorization
    - IR passes to automatically transform sub-tree to TensorCore Intrinsics

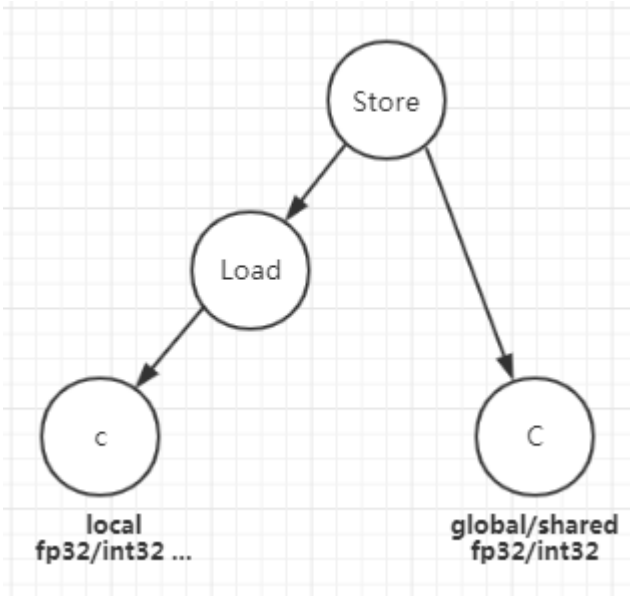


Pattern Matching

计算平台事业部
COMPUTING PLATFORM



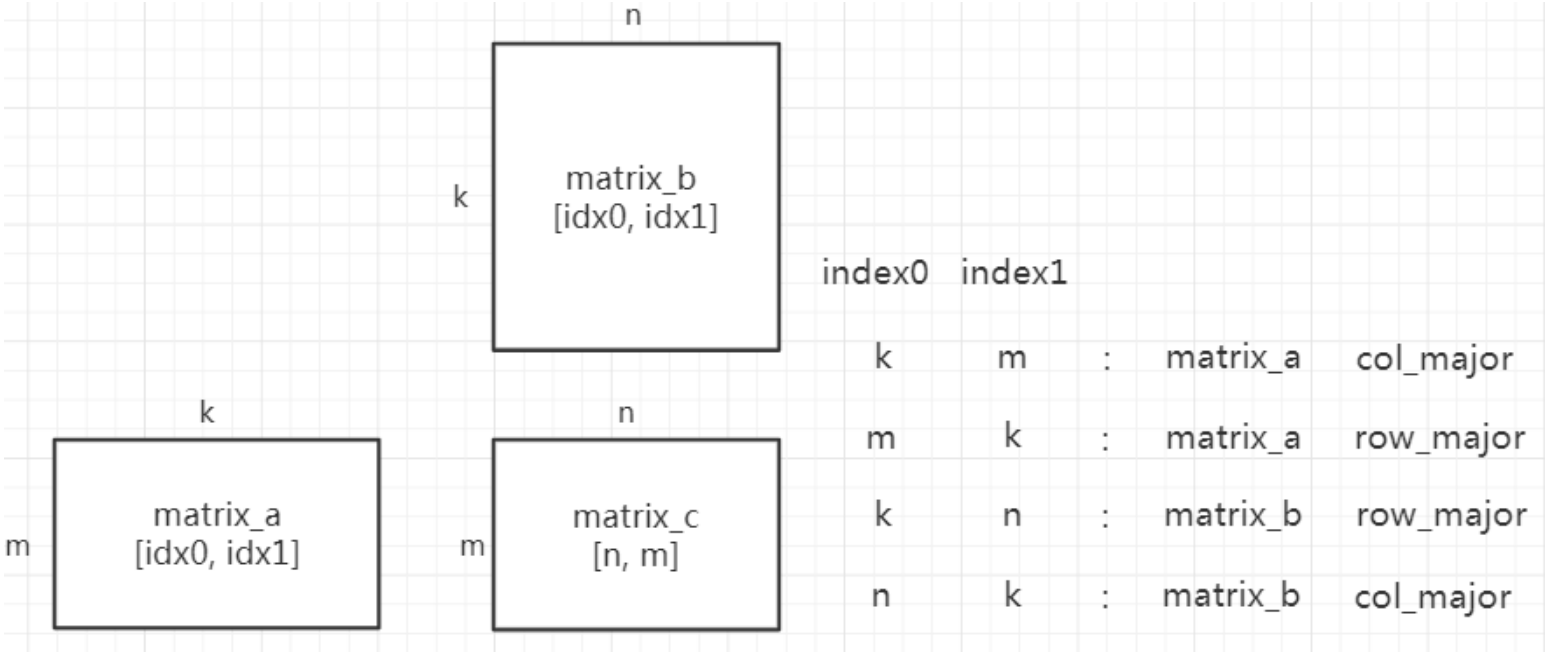




Matrix Identification

计算平台事业部 COMPUTING PLATFORM

- To figure out whether an input matrix is *matrix_a* or *matrix_b*, *row_major* or *col_major*.
  - Visit the body of *ComputeOp* to get the indices of input matrices: *index0*, *index1*
  - Compare the access indices with the *axis*/*reduce_axis* of *ComputeOp*



index0	index1			
k	m	:	matrix_a	col_major
m	k	:	matrix_a	row_major
k	n	:	matrix_b	row_major
n	k	:	matrix_b	col_major

Thread Index Unification

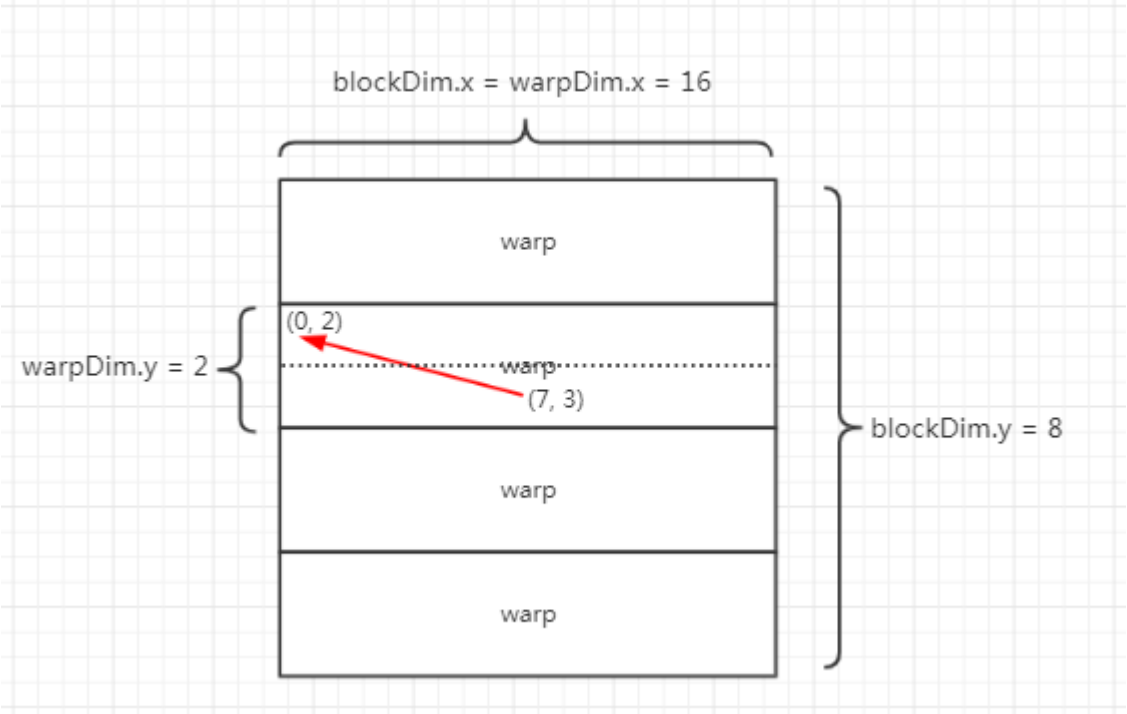
计算平台事业部 COMPUTING PLATFORM

- Thread index inside a warp should be the same for *wmma::load/store*

  - threadIdx.x
    -> 0

  - threadIdx.y
    -> threadIdx.y/warpDim.y*warpDim.y

    warpDim.y = 32/warpDim.x = 32/blockDim.x



Loop Scaling

计算平台事业部 COMPUTING PLATFORM

- *"wmma::mma_sync(c, a, b, c)" = "c = float(a)*float(b) + c" x (16x16x16/32)*

- Find the *IterVar* to scale according to the access indices of fragment registers

```
for (int k_inner_inner = 0; k_inner_inner < 16; ++k_inner_inner) {
    for (int j_c = 0; j_c < 8; ++j_c) {
      compute_local[j_c] = (compute_local[j_c] + ((float)(A_shared_local[k_inner_inner] * B_shared_local[((k_inner_inner * 8) + j_c)])));
    }
  }
```

```
for (int k_inner_inner = 0; k_inner_inner < 1; ++k_inner_inner) {
  for (int j_c = 0; j_c < 1; ++j_c) {
    wmma::mma_sync(compute_local[0], B_shared_local[0], A_shared_local[0], compute_local[0]);
  }
}
```

Performance Optimization

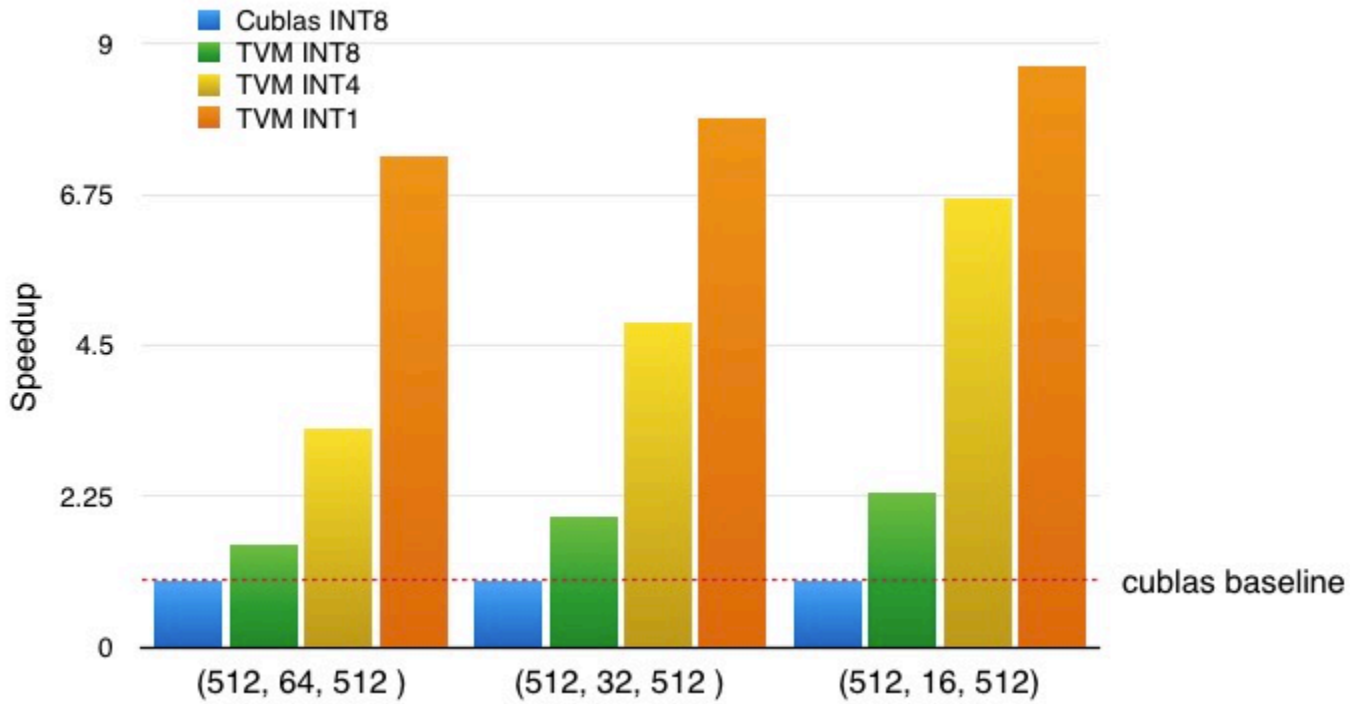
- Same as non-TensorCore CUDA codegen
  - Auto tune tiling sizes
  - Vectorized load/store for higher bandwidth utilization
  - Double buffer to hide memory load latency
  - Storage align to reduce bank conflicts of shared memory
  - Virtual threads for data reuse (on going)

Performance on V100 (FP16)

计算平台事业部
COMPUTING PLATFORM

M, N, K	cuBLAS TensorCore	TVM TensorCore	speedup
512, 16, 512	7.7470us	5.2570us	1.47X
512, 32, 512	8.0140us	6.0220us	1.33X
512, 64, 512	8.7530us	6.2390us	1.40X
512, 128, 512	9.0290us	7.1610us	1.26X
256, 256, 256	6.9380us	4.5930us	1.51X
1024, 32, 512	8.3320us	6.3770us	1.30X
2048, 32, 512	9.0640us	7.5070us	1.21X

Performance on T4



计算平台事业部
COMPUTING PLATFORM

FP16 Mixed-Precision Training on PAI

Auto Mixed-Precision

计算平台事业部
COMPUTING PLATFORM

PAI -name=$framework_name
-Dscript=$training_script
-DautoMixedPrecision=true
[-DlossScaling=$scale_factor]

No need to modify or add any line of code.

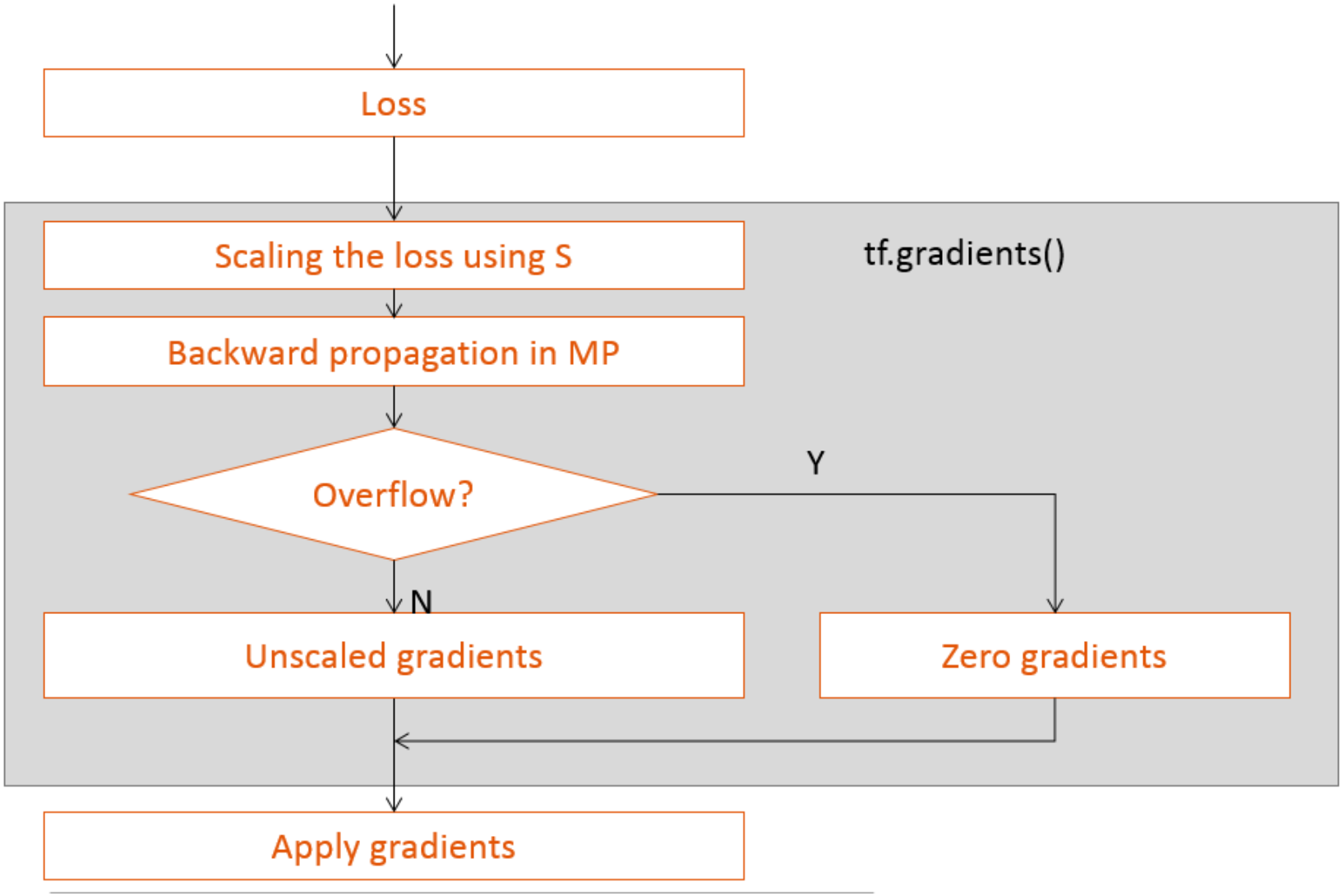
Loss Scaling in TF

计算平台事业部
COMPUTING PLATFORM

```
loss = loss_fn()
opt = tf.AdamOptimizer(learning_rate=...)

# Choose a loss scale manager which decides how to pick the right loss scale
# throughout the training process.
loss_scale_manger = tf.contrib.mixed_precision.FixedLossScaleManager(5000)

# Wraps the original optimizer in a LossScaleOptimizer.
loss_scale_optimizer = LossScaleOptimizer(opt, loss_scale_manager)

# Call minimize() on the loss scale optimizer.
train_op = loss_scale_optimizer.minimize(loss)
```

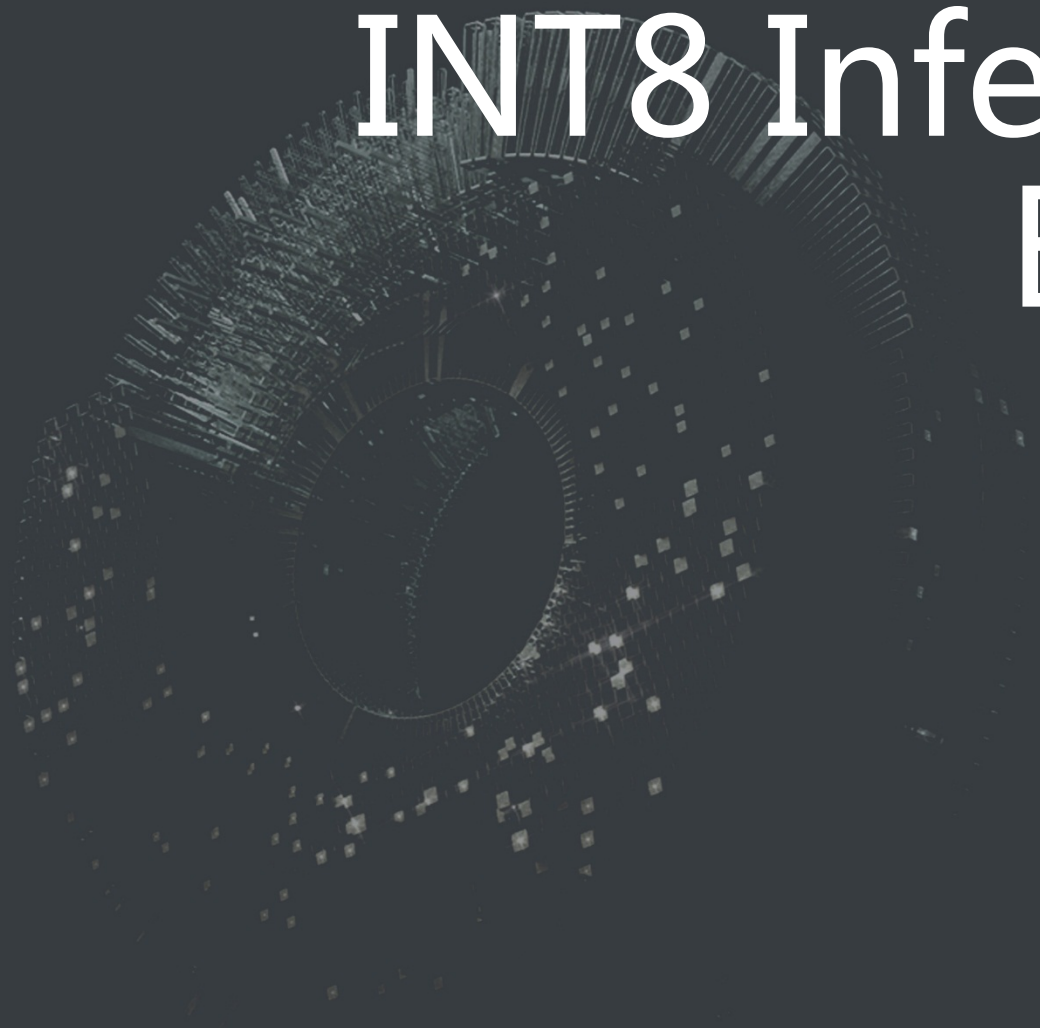
Loss Scaling in PAI-TF

计算平台事业部
COMPUTING PLATFORM



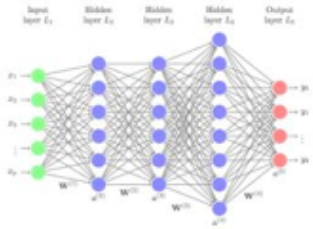
计算平台事业部
COMPUTING PLATFORM

INT8 Inference on PAI-Blade



PAI-Blade

计算平台事业部
COMPUTING PLATFORM



Model Analysis

Model Quantization

Graph optimization

Op Fusion Engine

DL Compiler Proxy

TensorRT Customized Optimizer

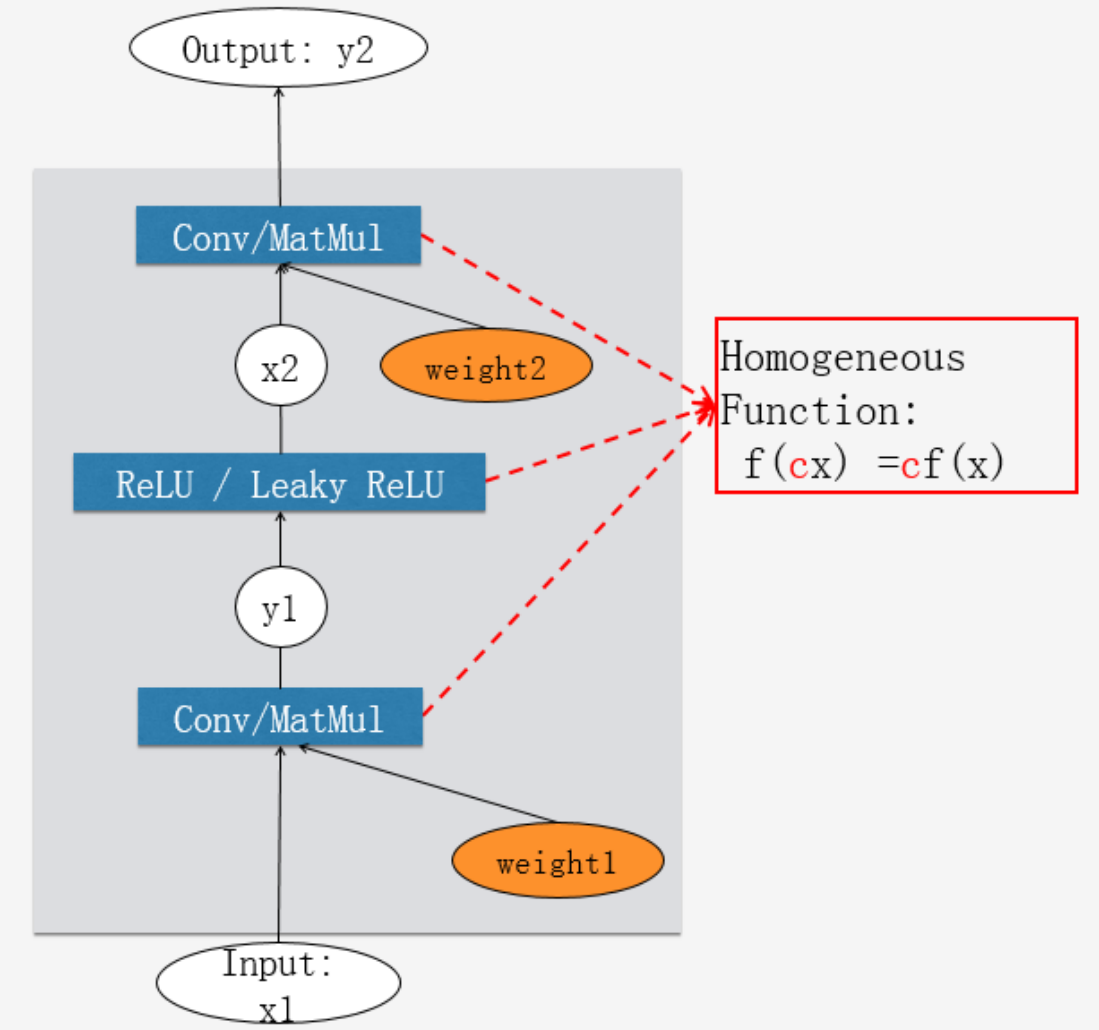
Blade Graph Optimizer

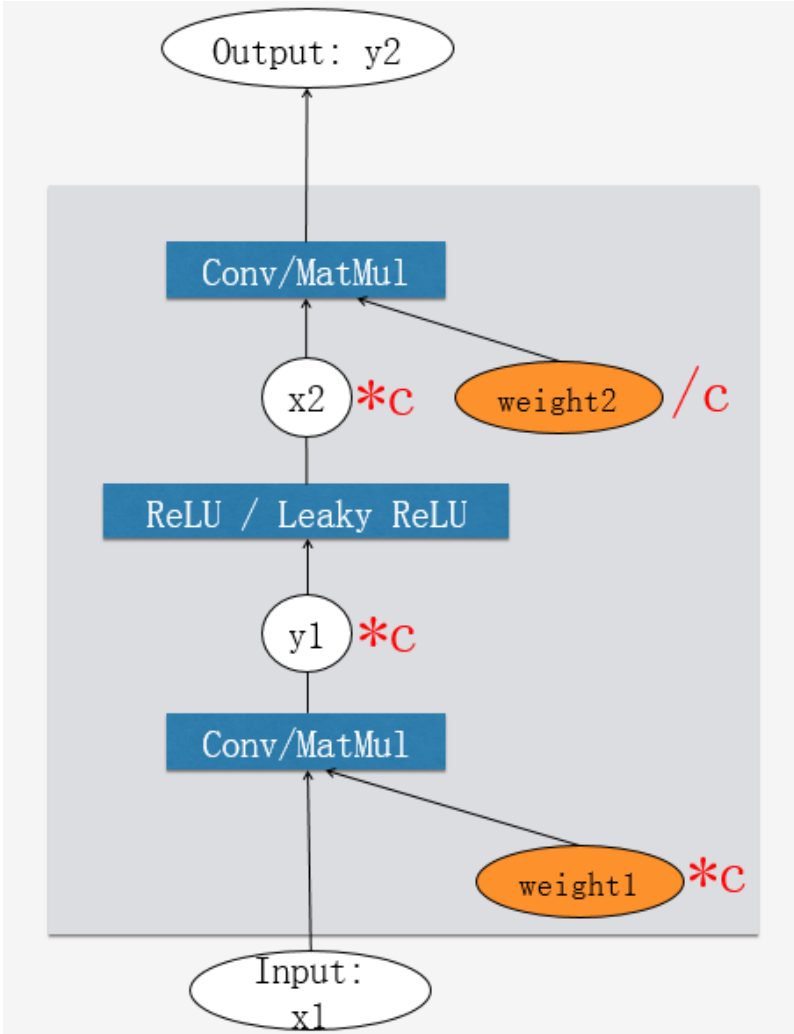
TAO Compiler(XLA)

TVM

cuBLAS/cuDNN/CUTLASS/
Blade Kernel Lib

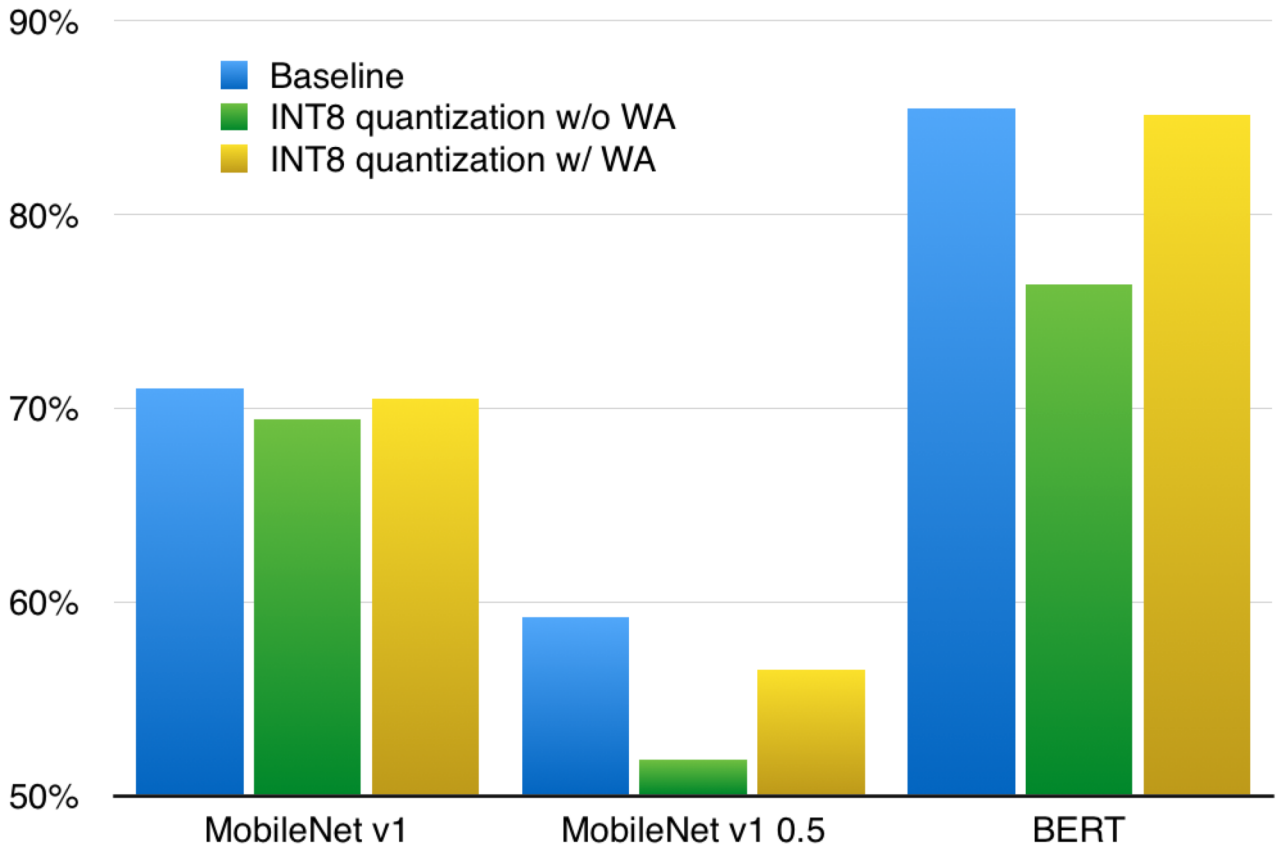
Weight Adjustment





Weight Adjustment

计算平台事业部
COMPUTING PLATFORM



References

- TensorCore AutoCodeGen Tutorial
- TensorCore AutoCodeGen Pull Request
- TensorCore Intrinsics Tutorial
- TensorCore Intrinsics Pull Request
- Programming Tensor Cores in CUDA 9

计算平台事业部
COMPUTING PLATFORM

Thanks!