Automatic TensorCore Scheduling

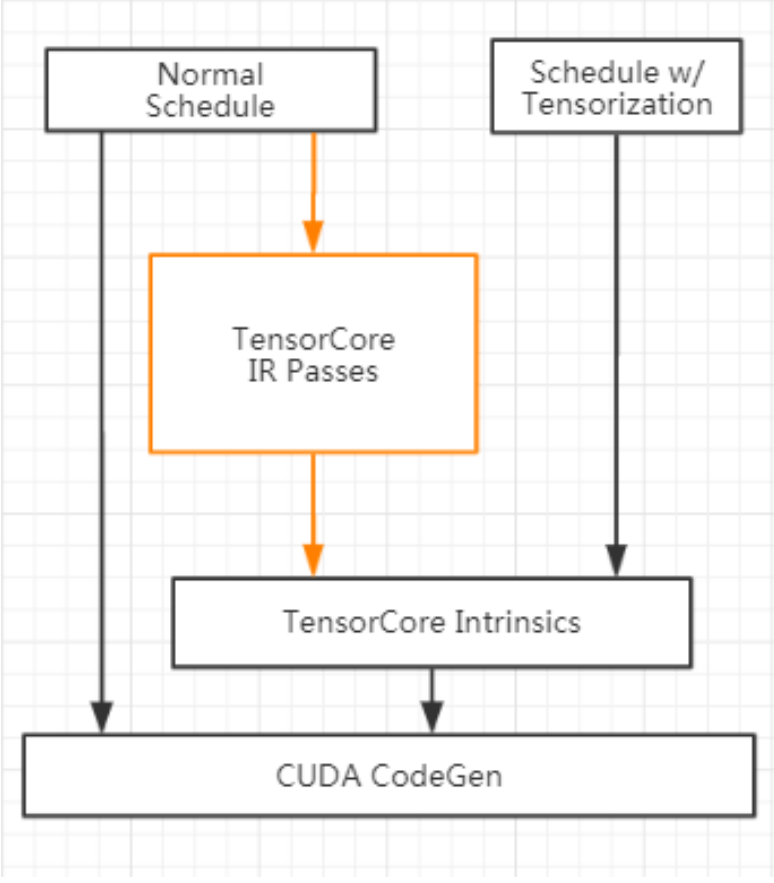
Xiaoyong Liu

PAI (Platform of AI)
Alibaba Cloud Intelligence

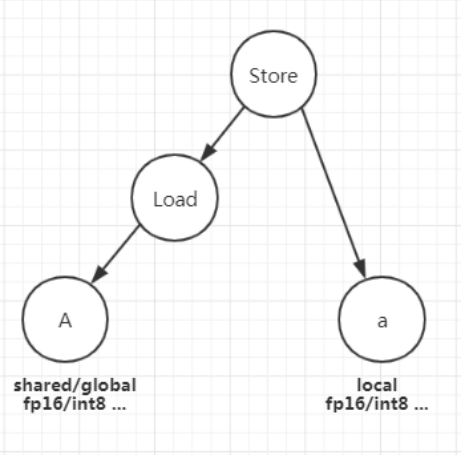
Presenting the work of PAI team !

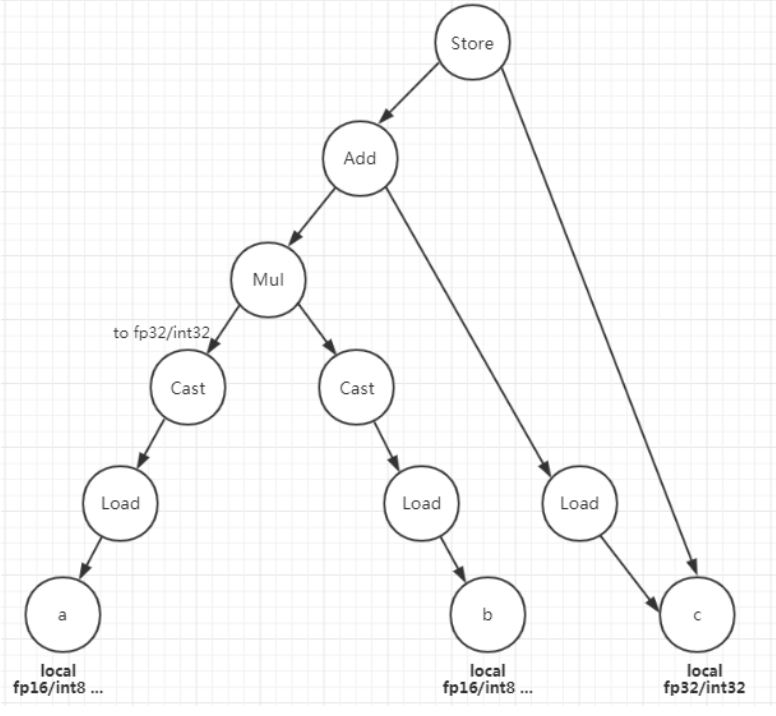
The Solution

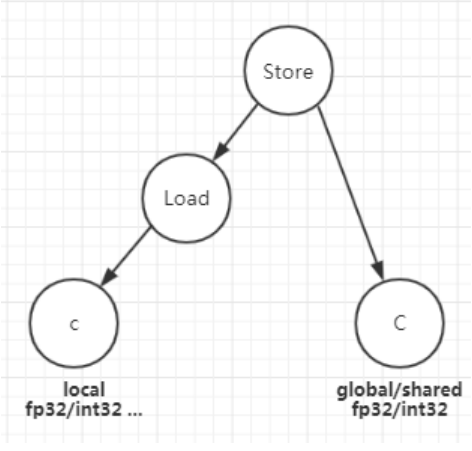
- **Generate TensorCore code automatically**
  - **Thread-Level Schedule for CUDA codegen**
    - warp tile shape
      - (16x16x16) : CUDA9
      - (32x8x16, 8x32x16) : CUDA10+
  - **Kind of Auto Tensorization**
    - IR passes to transform sub-tree to TensorCore Intrinsics



Pattern Matching

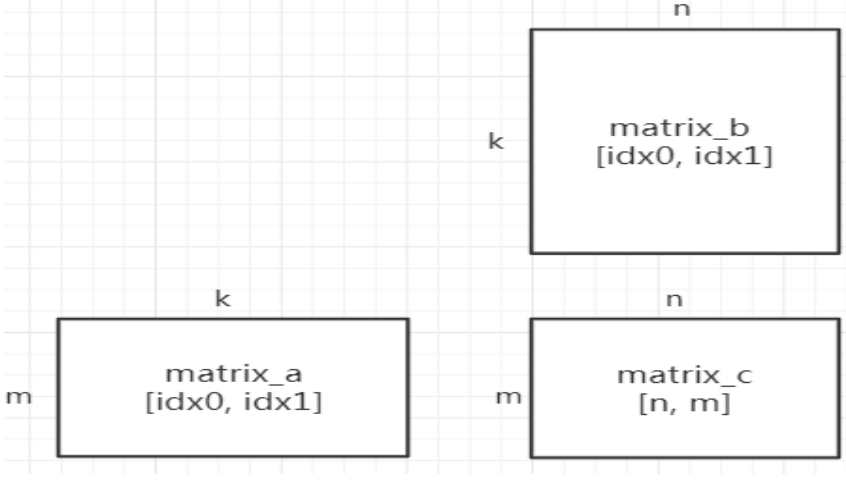






Matrix Identification

- input matrix attribute: *matrix_a* / *matrix_b*, *row_major* / *col_major*.
  - Retrieve indices of input from *ComputeOp* : *index0*, *index1*
  - Compare the indices to 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

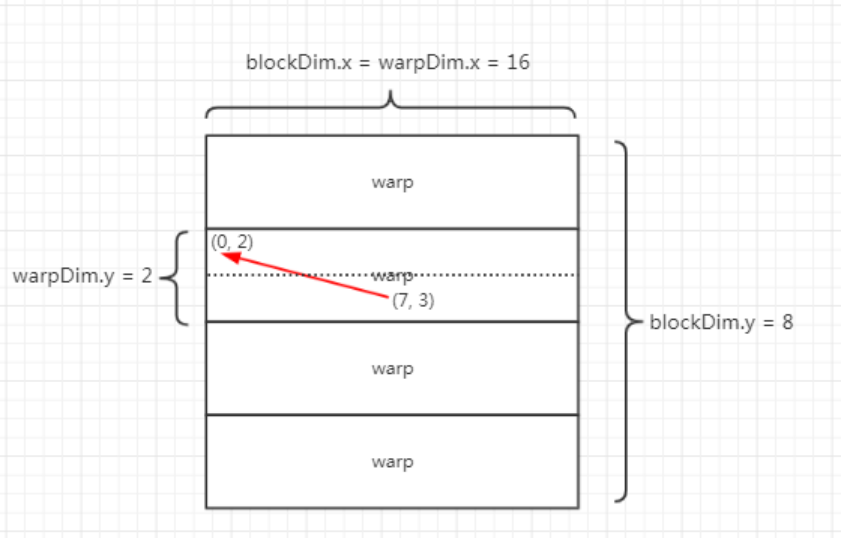
Thread Index Unification

- **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

- *"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]);
  }
}
```

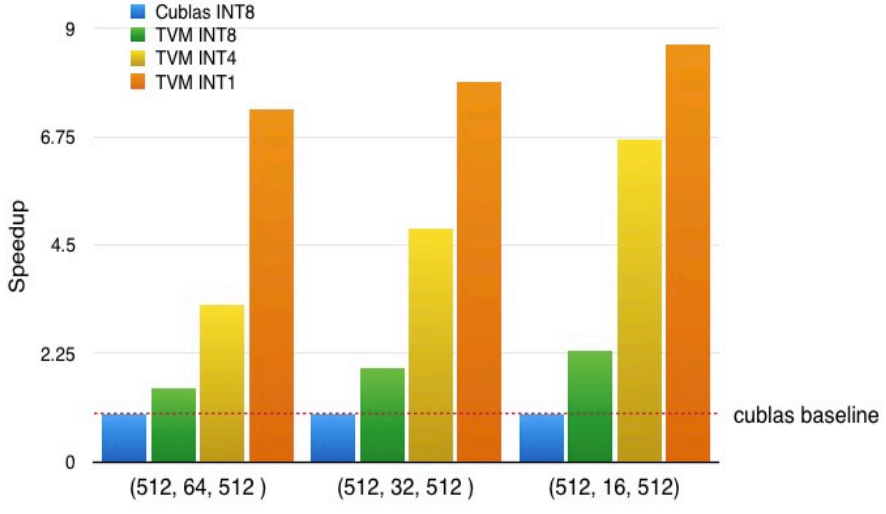
Performance Optimization

- 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)

Comparing with cublas TensorCore

FP16 on V100

INT8/4/1 on T4



https://docs.tvm.ai/tutorials/optimize/opt_matmul_auto_tensorcore.html