TensorCore and Tensorization

Siyuan Feng

Dec 5, 2019

Contents

1 TensorCore Introduction

2 TensorCore Support in TVM

3 Future Work

Contents

1 TensorCore Introduction

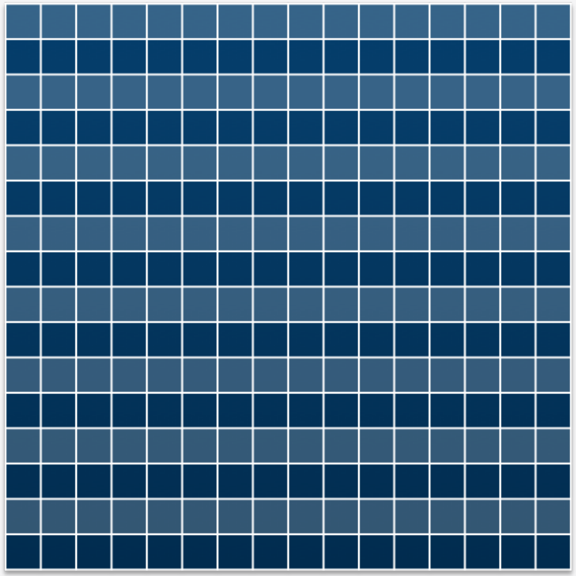
2 TensorCore Support in TVM

3 Future Work

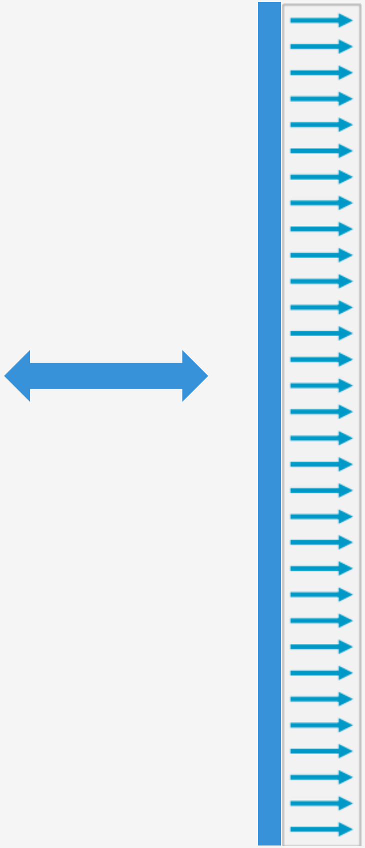
What are TensorCores

$$\underset{\text{FP16 or FP32}}{D} = \underset{\text{FP16}}{\begin{pmatrix} A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\ A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\ A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\ A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3} \end{pmatrix}} \underset{\text{FP16}}{\begin{pmatrix} B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\ B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\ B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\ B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3} \end{pmatrix}} + \underset{\text{FP16 or FP32}}{\begin{pmatrix} C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\ C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\ C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\ C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3} \end{pmatrix}}$$

Warp-Level Operation



wmma::fill_fragment(Cmat, 0.0f);



Programming TensorCore

16x16x16 MatMul

Create Fragments

```
__device__ void tensor_op_16_16_16
   ( float *d, half *a, half *b, float *c) {
   wmma::fragment<matrix_a> Amat;
   wmma::fragment<matrix_b> Bmat;
   wmma::fragment <accumulator> Cmat;
```

Load Fragments

```
   wmma::load_matrix_sync(Amat, a, 16);
   wmma::load_matrix_sync(Bmat, b, 16);
```

Perform MatMul

```
   wmma::fill_fragment(Cmat, 0.0f);
   wmma::mma_sync(Cmat, Amat, Bmat, Cmat);
```

Store Results

```
   wmma::store_matrix_sync(d, Cmat, 16,
                wmma::row_major);
}
```

TensorCore Summary

- TensorCores are hardware accelerators
- Warp-level operation
- New memory scope fragment

Contents

1 TensorCore Introduction

2 TensorCore Support in TVM

3 Future Work

Steps for TensorCore Support in TVM

Memory Scope

↓

Create Schedule

↓

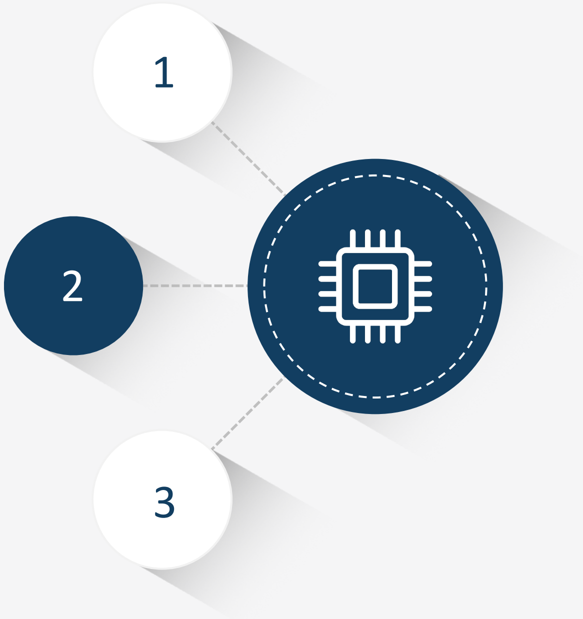
Tensorization

Current Memory Scope

global

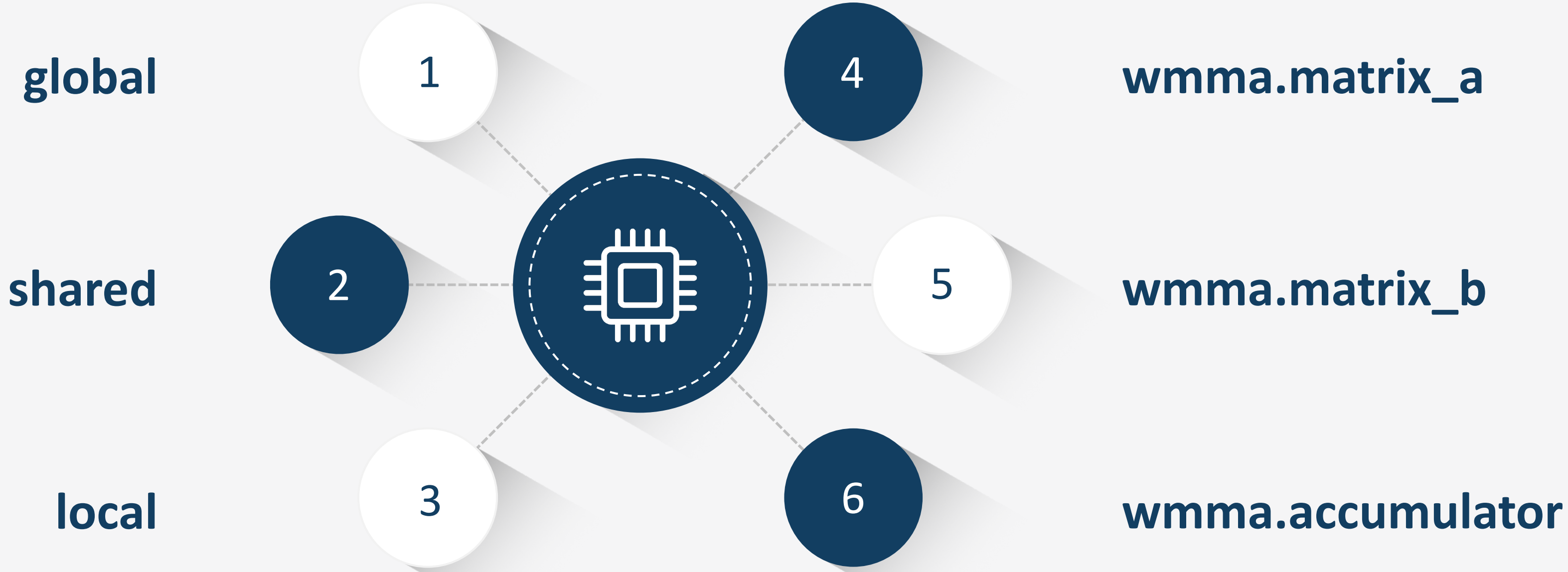
shared

local



Memory Scope → Create Schedule → Tensorization

Special Memory Scope



Memory Scope → Create Schedule → Tensorization

Traditional GPU Memory Scope Order

Memory Scope → Create Schedule → Tensorization

Enhanced TensorCore Memory Scope Order

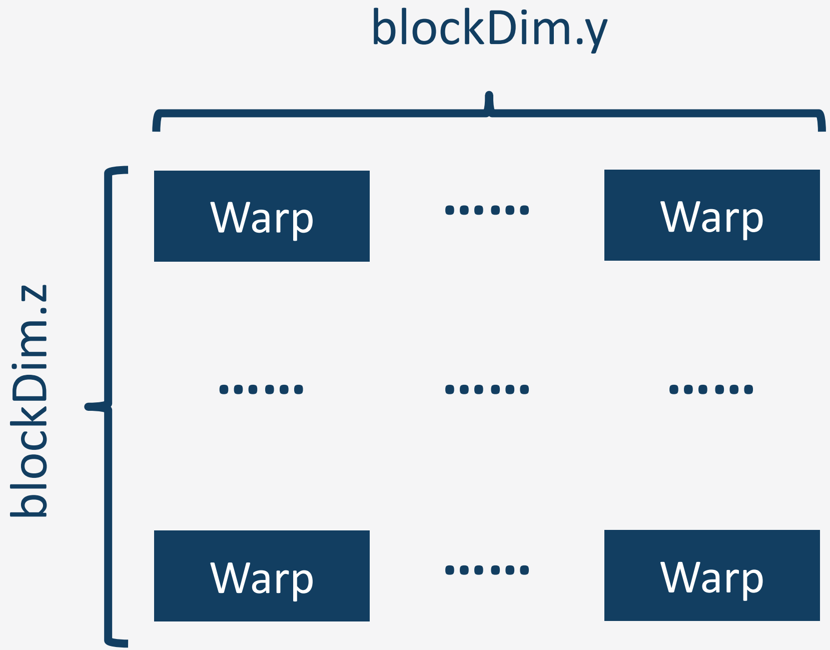
Memory Scope → Create Schedule → Tensorization

Warp Level Schedule

blockDim.x = warp_size= 32

Memory Scope → **Create Schedule** → Tensorization

Warp Level Schedule



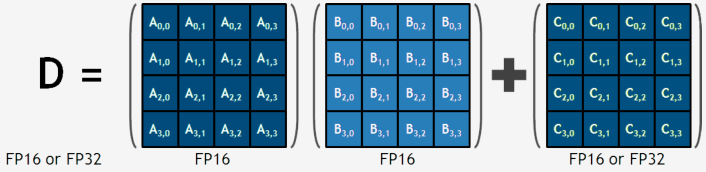
blockDim.x = warp_size= 32

Memory Scope → **Create Schedule** → Tensorization

Tensorization

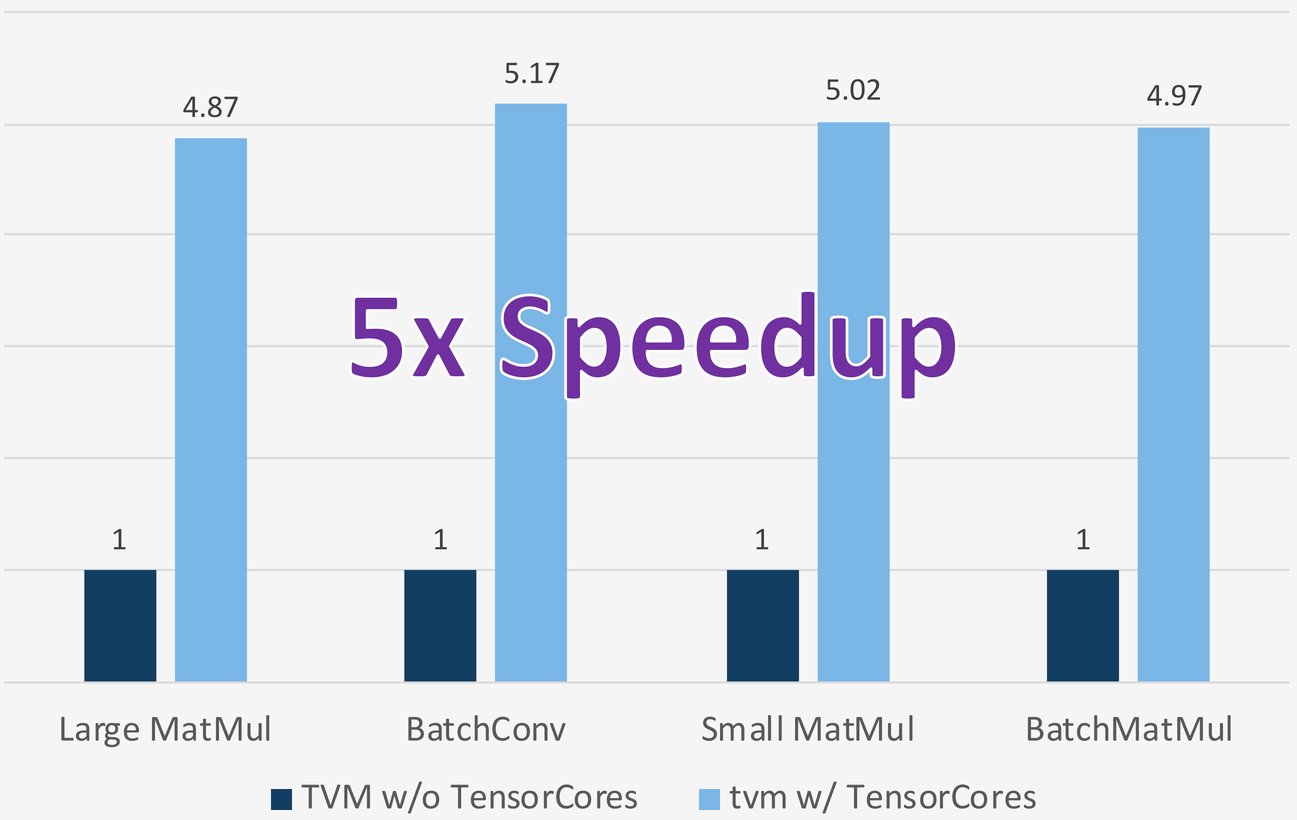
```
for (i, 0, 16) {
    for (j, 0, 16) {
        for (k, 0, 16) {
            C[i*16 + j]= (C[i*16 + j] + (float32(A[i*16 + k])*float32(B[k*16 + j])))
        }
    }
}
```

```
tvm_mma_sync(C, 0, A, 0, B, 0, C, 0);
```

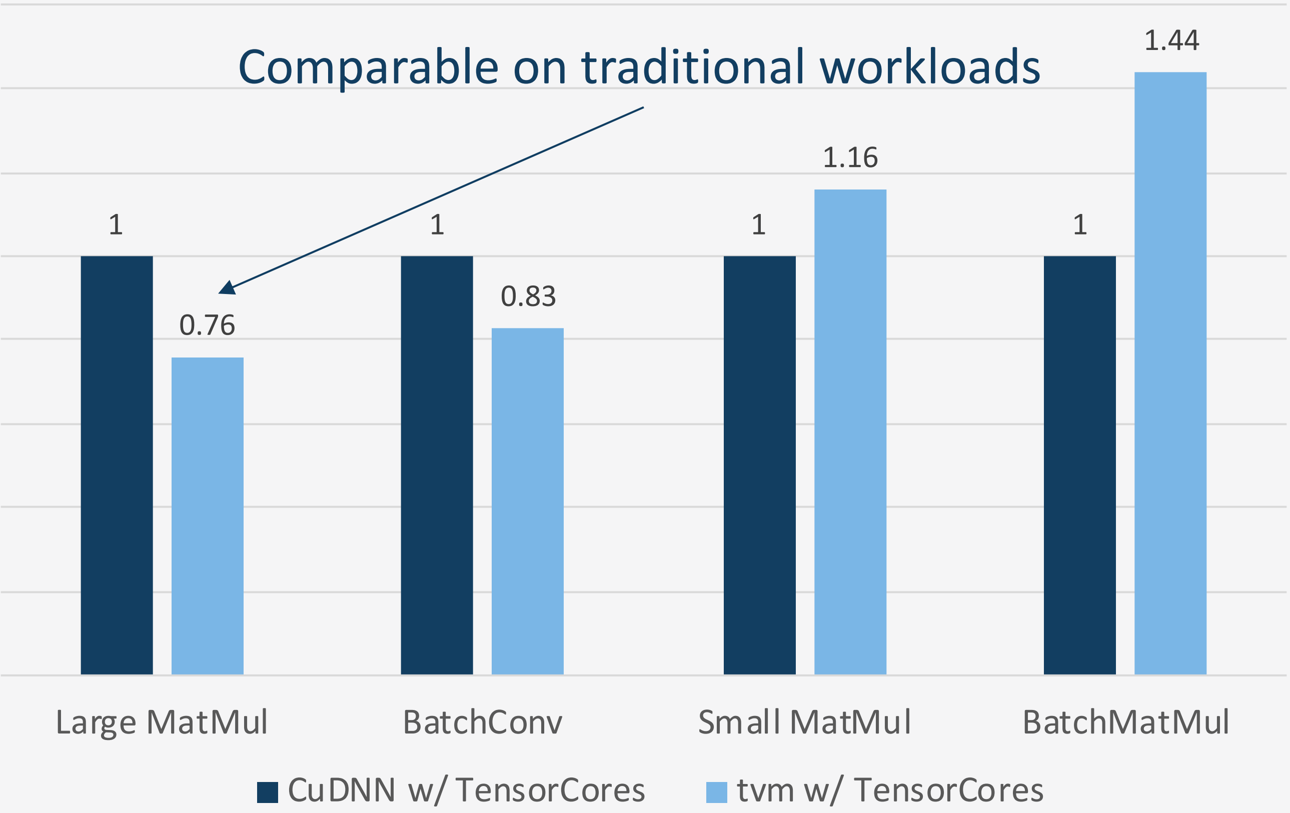


Memory Scope → Create Schedule → **Tensorization**

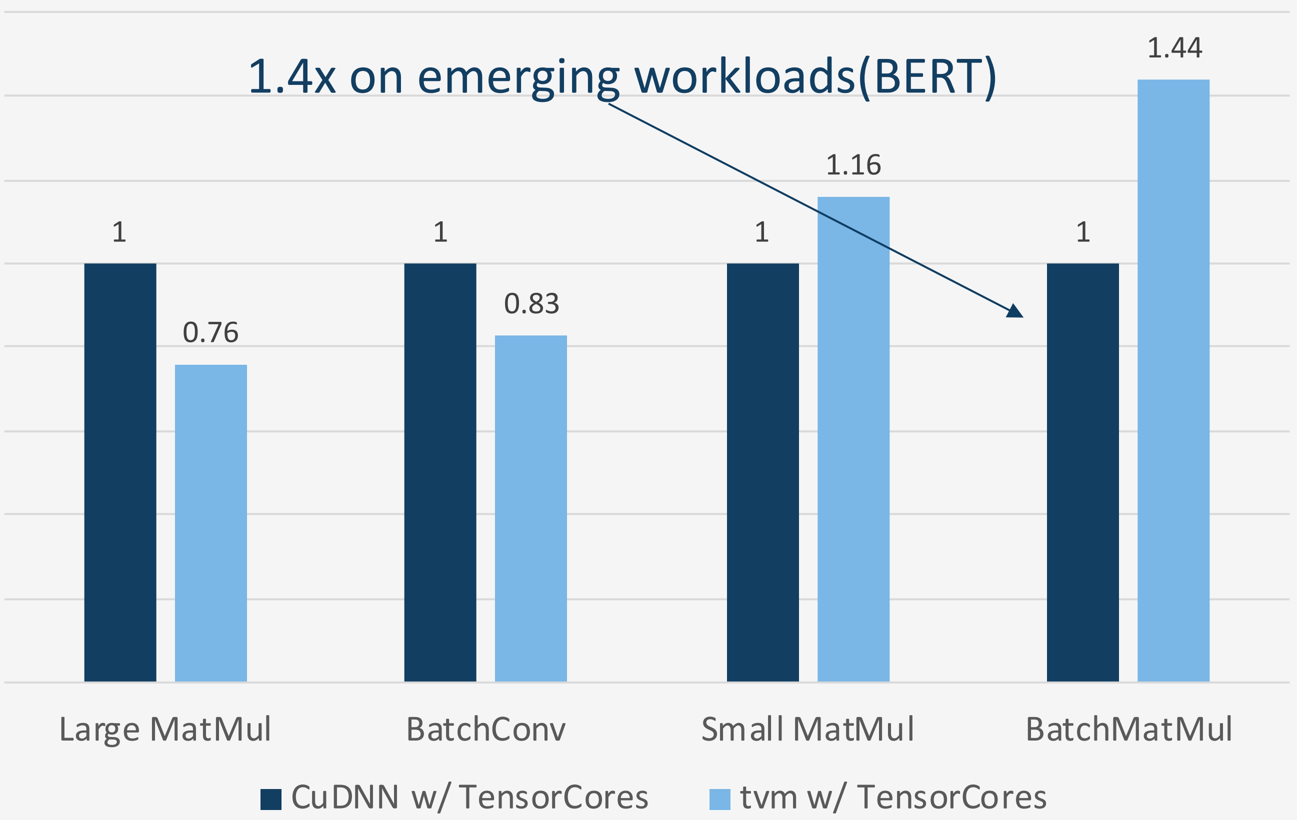
Performance Improvements over non-TensorCore



Performance Comparison vs CuDNN



Performance Comparison vs CuDNN



TVM TensorCore Support Summary

- Massive speed up over non-tensorcore
- Competitive performance with CuDNN
- Based on tensor intrinsic

Contents

TensorCore Introduction 1

2 TensorCore Support in TVM

Future Work 3

Future Work

1. Use TensorCore in TOPI and Relay

2. Apply TensorCore to popular ML model, such as BERT

Thank you

Siyuan Feng

Dec 5, 2019