TensorCore and Tensorization

Siyuan Feng  Dec 5, 2019
What are TensorCores

\[
D = \begin{pmatrix}
A_{0,0} & A_{0,1} & A_{0,2} & A_{0,3} \\
A_{1,0} & A_{1,1} & A_{1,2} & A_{1,3} \\
A_{2,0} & A_{2,1} & A_{2,2} & A_{2,3} \\
A_{3,0} & A_{3,1} & A_{3,2} & A_{3,3}
\end{pmatrix}
\begin{pmatrix}
B_{0,0} & B_{0,1} & B_{0,2} & B_{0,3} \\
B_{1,0} & B_{1,1} & B_{1,2} & B_{1,3} \\
B_{2,0} & B_{2,1} & B_{2,2} & B_{2,3} \\
B_{3,0} & B_{3,1} & B_{3,2} & B_{3,3}
\end{pmatrix}
\begin{pmatrix}
C_{0,0} & C_{0,1} & C_{0,2} & C_{0,3} \\
C_{1,0} & C_{1,1} & C_{1,2} & C_{1,3} \\
C_{2,0} & C_{2,1} & C_{2,2} & C_{2,3} \\
C_{3,0} & C_{3,1} & C_{3,2} & C_{3,3}
\end{pmatrix}
\]
Warp-Level Operation

wmma::fill_fragment(Cmat, 0.0f);
Programming TensorCore

16x16x16 MatMul

<table>
<thead>
<tr>
<th>Create Fragments</th>
<th><strong>device</strong> void tensor_op_16_16_16 ( float *d, half *a, half *b, float *c) {</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>wmma::fragment&lt;matrix_a&gt; Amat;</td>
</tr>
<tr>
<td></td>
<td>wmma::fragment&lt;matrix_b&gt; Bmat;</td>
</tr>
<tr>
<td></td>
<td>wmma::fragment &lt;accumulator&gt; Cmat;</td>
</tr>
<tr>
<td>Load Fragments</td>
<td></td>
</tr>
<tr>
<td></td>
<td>wmma::load_matrix_sync(Amat, a, 16);</td>
</tr>
<tr>
<td></td>
<td>wmma::load_matrix_sync(Bmat, b, 16);</td>
</tr>
<tr>
<td>Perform MatMul</td>
<td></td>
</tr>
<tr>
<td></td>
<td>wmma::fill_fragment(Cmat, 0.0f);</td>
</tr>
<tr>
<td></td>
<td>wmma::mma_sync(Cmat, Amat, Bmat, Cmat);</td>
</tr>
<tr>
<td>Store Results</td>
<td></td>
</tr>
<tr>
<td></td>
<td>wmma::store_matrix_sync(d, Cmat, 16, wmma::row_major);</td>
</tr>
</tbody>
</table>

Create Fragments:  
Load Fragments:  
Perform MatMul:  
Store Results:
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

1. Memory Scope
2. Create Schedule
3. Tensorization

Elements:
- global
- shared
- local

Memory Locations:
- wmma.matrix_a
- wmma.matrix_b
- wmma.accumulator
Traditional GPU Memory Scope Order

- Global
- Shared
- Local
- Global

Memory Scope → Create Schedule → Tensorization
Enhanced TensorCore Memory Scope Order

- Global
- Shared
- Fragment Local
- Global

Memory Scope ➔ Create Schedule ➔ Tensorization
Warp Level Schedule

\[ \text{blockDim}.x = \text{warp}_\text{size} = 32 \]
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);
Performance Improvements over non-TensorCore

- Large MatMul: 4.87 TVM w/o TensorCores, 5.17 tvm w/ TensorCores
- BatchConv: 1 TVM w/o TensorCores, 1 tvm w/ TensorCores
- Small MatMul: 1 TVM w/o TensorCores, 1 tvm w/ TensorCores
- BatchMatMul: 1 TVM w/o TensorCores, 1 tvm w/ TensorCores

5x Speedup
Performance Comparison vs CuDNN

Comparable on traditional workloads

<table>
<thead>
<tr>
<th>Workload</th>
<th>CuDNN w/ TensorCores</th>
<th>tvm w/ TensorCores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large MatMul</td>
<td>1</td>
<td>0.76</td>
</tr>
<tr>
<td>BatchConv</td>
<td>1</td>
<td>0.83</td>
</tr>
<tr>
<td>Small MatMul</td>
<td>1</td>
<td>1.16</td>
</tr>
<tr>
<td>BatchMatMul</td>
<td>1</td>
<td>1.44</td>
</tr>
</tbody>
</table>
### Performance Comparison vs CuDNN

<table>
<thead>
<tr>
<th>Workload</th>
<th>CuDNN w/ TensorCores</th>
<th>tvm w/ TensorCores</th>
</tr>
</thead>
<tbody>
<tr>
<td>Large MatMul</td>
<td>1</td>
<td>0.76</td>
</tr>
<tr>
<td>BatchConv</td>
<td>1</td>
<td>0.83</td>
</tr>
<tr>
<td>Small MatMul</td>
<td>1</td>
<td>1.16</td>
</tr>
<tr>
<td>BatchMatMul</td>
<td>1</td>
<td>1.44</td>
</tr>
</tbody>
</table>

1.4x on emerging workloads (BERT)
TVM TensorCore Support Summary

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