A Decomposition-Based Synthesis Algorithm for Sparse Matrix-Vector Multiplication in Parallel Communication Structure

Mingfei Yu, Ruitao Gao, Masahiro Fujita
University of Tokyo
Outline:

1) Motivations

2) Related Works

3) Minimum Scheduling Problems

4) The Proposed Decomposition-Based Algorithm

5) Experimental Results

6) Conclusions
Motivation:

In the research field of deep learning:

(1) Technologies are emerging, such as convolutional neural network (CNN), self-attention, which are widely adopted in various domains.

(2) Larger datasets and deeper models are bringing about significant improvements in accuracy, and at the same time cause computational power to become a bottleneck.

(3) Current implementations heavily rely on hardware with many processing elements (PEs).
Motivation:

Convolution computation can be interpreted as Sparse Matrix-Vector Multiplication (SpMV) → Two difficulties to parallel schedule SpMV:

(1) Arbitrary distribution of zero elements in the involved sparse matrix, resulted by unstructured pruning.

(2) Search space for corresponding optimum scheduling problem increases sharply when matrix size getting larger.
Outline:

1) Motivations

2) Related Works

3) Minimum Scheduling Problems

4) The Proposed Decomposition-Based Algorithm

5) Experimental Results

6) Conclusions
Related Work:

Cons:

(1) Accuracy loss
(2) Manual effort (extra instruction set)
(3) Difficult to apply on PEs with interconnection

Cons: Weak scalability

Scalable  Fast  High quality

Target Architecture:

The adopted ring-connected architecture:

Merits:

(1) Simple enough.
A save of hardware resource for data communication.

(2) Qualified for popular implementations, such as DMV[1].

---

Outline:

1) Motivations

2) Related Works

3) **Minimum Scheduling Problems**

4) The Proposed Decomposition-Based Algorithm

5) Experimental Results

6) Conclusions
Minimum Scheduling Problem:

\[ a(t_0, m_0) \] – in cycle \( t_0 \), data \( a \) is in the local memory of PE- \( m_0 \).

\[ a_{\text{trans}}(t_0, m_0) \] – in cycle \( t_0 \), PE- \( m_0 \) sends data \( a \) to the adjacent PE.

(1) \[ a_{\text{trans}}(t_0, m_0) \rightarrow a(t_0, m_0) \land a(t_0 + 1, m_0 + 1). \]

(2) \[ a_{\text{trans}}(t_0, m_0) \rightarrow a(t_0, m_0) \land a(t_0 + 1, m_0 + 1) \).

(3) \[ a(t_0, m_0) \rightarrow a_{\text{trans}}(t_0, m_0). \]

(4) \[ a(t_0 + 1, m_0 + 1) \rightarrow a_{\text{trans}}(t_0, m_0). \]
Minimum Scheduling Problem:

\[
\begin{pmatrix}
y[1] \\
y[2] \\
y[3] \\
y[4]
\end{pmatrix} =
\begin{pmatrix}
0 & w[1][2] & w[1][3] & w[1][4] \\
w[3][1] & 0 & w[3][3] & w[3][4] \\
0 & w[4][2] & 0 & w[4][4]
\end{pmatrix}
\begin{pmatrix}
x[1] \\
x[2] \\
x[3] \\
x[4]
\end{pmatrix}
\]

\[R = 3\quad C = 4\]

\(Q:\) the number of non-zeros in the matrix.
\(M:\) the number of available PEs.
\(C:\) the largest numbers of non-zeros in a column.
\(R:\) the largest numbers of non-zeros in a row.

\(T:\) the minimum number of achievable computing cycles.

\(T_0:\) the minimum number of computing cycles, in the aspect of the amount of calculation.

\[
T_0 = \lceil Q/M \rceil = \lceil 11/4 \rceil = 3.
\]

\[
T = \max\{R, C, T_0\} = \max\{3, 4, 3\} = 3 = C.
\]
Minimum Scheduling Problem:

<table>
<thead>
<tr>
<th>Cycle</th>
<th>Core 1</th>
<th>Core 2</th>
<th>Core 3</th>
<th>Core 4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cycle 1</td>
<td>0</td>
<td>w[2][2]</td>
<td>w[3][3]</td>
<td>w[4][4]</td>
</tr>
<tr>
<td>Cycle 2</td>
<td>0</td>
<td>w[1][2]</td>
<td>w[2][3]</td>
<td>w[3][4]</td>
</tr>
<tr>
<td>Cycle 4</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>w[1][4]</td>
</tr>
</tbody>
</table>

\[
\begin{bmatrix}
  y[1] \\
  y[2] \\
  y[3] \\
  y[4]
\end{bmatrix} =
\begin{bmatrix}
  0 & w[1][2] & w[1][3] & w[1][4] \\
  w[3][1] & 0 & w[3][3] & w[3][4] \\
  0 & w[4][2] & 0 & w[4][4]
\end{bmatrix}
\begin{bmatrix}
  x[1] \\
  x[2] \\
  x[3] \\
  x[4]
\end{bmatrix}
\]

- Multi-Input Vector
- Core 1
- Core 2
- Core 3
- Core 4

yu@cad.t.u-tokyo.ac.jp  A Decomposition-Based Synthesis Algorithm for SpMV in Parallel Communication Structure
Minimum Scheduling Problem:

<table>
<thead>
<tr>
<th></th>
<th>core1</th>
<th>core2</th>
<th>core3</th>
<th>core4</th>
</tr>
</thead>
<tbody>
<tr>
<td>cycle1</td>
<td>w[1][1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>y[1]</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

| cycle2 | 0     | w[1][2] | 0     | 0     |
|        | 0     | y[1]    | 0     | 0     |

| cycle3 | 0     | 0     | w[1][3] | 0     |
|        | 0     | 0     | y[1]    | 0     |

| cycle4 | 0     | 0     | 0     | w[1][4] |
|        | 0     | 0     | 0     | y[1]    |

multi – Input Vector & multi – Output Vector

\[
\begin{pmatrix}
  y[1] \\
y[2] \\
y[3] \\
y[4]
\end{pmatrix} = 
\begin{pmatrix}
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0 \\
  0 & 0 & 0 & 0
\end{pmatrix} \cdot 
\begin{pmatrix}
  x[1] \\
x[2] \\
x[3] \\
x[4]
\end{pmatrix}
\]

yu@cad.t.u-tokyo.ac.jp

A Decomposition-Based Synthesis Algorithm for SpMV in Parallel Communication Structure
Outline:

1) Motivations

2) Related Works

3) Minimum Scheduling Problems

4) The Proposed Decomposition-Based Algorithm

5) Experimental Results

6) Conclusions
The Proposed Decomposition-Based Algorithm:

For two SpMV calculations with the same number of zeros in the matrix, if the zero distribution in the former’s matrix can be converted into the one in the latter’s matrix through matrix transformation, their parallel scheduling solutions can be converted into each other correspondingly as well.

A decomposition-based synthesis algorithm
The Proposed Decomposition-Based Algorithm:

Process flow of the proposed algorithm:

Step 1: Conduct global matrix transformation to concentrate the zeros.

Step 2: Operate assignment and division to obtain sub-matrices.

Step 3: Obtain scheduling solution through modification on the matched template, while templates are stored in a library.

Step 4: For those sub-matrices that cannot match any existing template, solve local optimum scheduling problem to enrich the library incrementally.

Step 5: Establish the global parallel scheduling solution by combining all the local ones.
Outline:

1) Motivations

2) Related Works

3) Minimum Scheduling Problems

4) The Proposed Decomposition-Based Algorithm

5) Experimental Results

6) Conclusions
Experimental Results:

<table>
<thead>
<tr>
<th>Size</th>
<th>Sparsity</th>
<th>Cores</th>
<th>Cycles for DMV</th>
<th>Run in time sec</th>
<th>Cycles(Overhead)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Method1&lt;sup&gt;a&lt;/sup&gt;</td>
<td>Method4&lt;sup&gt;b&lt;/sup&gt;</td>
</tr>
<tr>
<td>8</td>
<td>0.7</td>
<td>4</td>
<td>16</td>
<td>0.01</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>4</td>
<td>16</td>
<td>0.07</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>0.3</td>
<td>4</td>
<td>16</td>
<td>0.09</td>
<td>0.01</td>
</tr>
<tr>
<td>12</td>
<td>0.7</td>
<td>4</td>
<td>36</td>
<td>1.29</td>
<td>0.01</td>
</tr>
<tr>
<td>12</td>
<td>0.5</td>
<td>4</td>
<td>36</td>
<td>7.33</td>
<td>0.03</td>
</tr>
<tr>
<td>12</td>
<td>0.3</td>
<td>4</td>
<td>36</td>
<td>15.20</td>
<td>0.06</td>
</tr>
<tr>
<td>16</td>
<td>0.7</td>
<td>4</td>
<td>64</td>
<td>/</td>
<td>0.03</td>
</tr>
<tr>
<td>8</td>
<td>0.7</td>
<td>8</td>
<td>8</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>0.5</td>
<td>8</td>
<td>8</td>
<td>0.30</td>
<td>0.00</td>
</tr>
<tr>
<td>8</td>
<td>0.3</td>
<td>8</td>
<td>8</td>
<td>2.41</td>
<td>0.01</td>
</tr>
<tr>
<td>16</td>
<td>0.7</td>
<td>8</td>
<td>32</td>
<td>/</td>
<td>0.04</td>
</tr>
</tbody>
</table>

<sup>a</sup> A state-of-art SAT solver.
<sup>b</sup> The same SAT solver with the proposed algorithm adopted.
Experimental Results:

Compared with exhaustive search-based method, the proposed one has achieved:

- remarkabe scalability of being able to schedule SpMV calculations with sizes of involved matrices up to 36 (limited by the size of CNF files).
- a 250 times speed up in average to generate scheduling solution for all our trials so far.
- utilizing 65.27% of the sparsity to reduce the required computation cycles, which has achieved 91.39% of the performance of exhaustive search-based method and is therefore identified as near optimum.
Outline:

1) Motivations

2) Related Works

3) Minimum Scheduling Problems

4) The Proposed Decomposition-Based Algorithm

5) Experimental Results

6) Conclusions
Conclusions:

- Minimum schedule SpMV computation onto ring-connected architecture
  - Technologies to improve the quality of scheduling solution
  - Classification of sparse matrices for scheduling solution reuse

- A decomposition-based synthesis algorithm for scheduling SpMV
  - Integrate the proposed technologies to improve scheduler performance
  - Effectively improve the scalability of the scheduler

- Future work
  - Implementation targeting real-word deep learning models
Thanks for watching.