IA-SpGEMM: an Input-aware Auto-tuning Framework for Parallel Sparse Matrix-Matrix Multiplication

Zhen Xie†, Guangming Tan†, Weifeng Liu § ‡, Ninghui Sun†

†State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences
§ University of Chinese Academy of Sciences
‡Department of Computer Science and Technology, China University of Petroleum, Beijing

ICS ‘19, June 26-28, Phoenix AZ, USA
Overview

- Sparse GEMM (SpGEMM) overview
- Two overheads and motivation
- Our SpGEMM algorithms
  - Algorithms: SpGEMM in DIA, COO and ELL formats
  - Performance improvements
- IA-SpGEMM: Input-aware Auto-tuning Framework
  - Input features (sparse features and density representation)
  - MatNet design
- Experimental results
- Conclusion
Overview

- Sparse GEMM (SpGEMM) overview
- Two overheads and motivation
- Our SpGEMM algorithms
  - Algorithms: SpGEMM in DIA, COO and ELL formats
  - Performance improvements
- IA-SpGEMM: Input-aware Auto-tuning Framework
  - Input features (sparse features and density representation)
  - MatNet design
- Experimental results
- Conclusion
Sparse Matrix - Data Storage Format

• If most elements in a matrix are zeros, we can use sparse representations to store the matrix

• Different formats represent different space occupations and memory access sequences

\[
A = \begin{bmatrix}
1 & 2 & 0 & 0 \\
0 & 3 & 4 & 0 \\
0 & 0 & 5 & 6 \\
0 & 0 & 0 & 7
\end{bmatrix}
\]

Compressed Sparse Row (CSR)
\[
\text{ptr} = [0 \ 2 \ 4 \ 6 \ 7] \\
\text{col\_ind} = [0 \ 1 \ 1 \ 2 \ 2 \ 3 \ 3] \\
\text{data} = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7]
\]

Coordinate (COO)
\[
\text{ptr} = [0 \ 2 \ 4 \ 6 \ 7] \\
\text{rows} = [0 \ 0 \ 1 \ 1 \ 2 \ 2 \ 3] \\
\text{cols} = [0 \ 1 \ 1 \ 2 \ 2 \ 3 \ 3] \\
\text{data} = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7]
\]

Diagonal (DIA)
\[
\text{pos} = [-1 \ -1 \ -1 \ 0 \ 1 \ -1 \ -1] \\
\text{offsets} = [0 \ 1] \\
\text{data} = [1 \ 3 \ 5 \ 7 \ 2 \ 4 \ 6 \ *
\]

ELLPACK (ELL)
\[
\text{nnz} = [2 \ 2 \ 2 \ 1] \\
\text{col\_ind} = [0 \ 1 \ 1 \ 2 \ 2 \ 3 \ 3 \ *] \\
\text{data} = [1 \ 2 \ 3 \ 4 \ 5 \ 6 \ 7 \ *]
\]
Sparse GEMM (SpGEMM) - Basics

- Multiply a sparse matrix $A$ by a sparse matrix $B$, obtain a result sparse matrix $C$.

$$
\begin{array}{c|c|c}
& 1 & \\
2 & 3 & \\
4 & 5 & 6 \\
\end{array} \quad \times \quad \begin{array}{c|c|c}
& a & \\
b & c & \\
d & e & \\
f & & \\
\end{array} \quad = \quad \begin{array}{c|c|c}
1d & 1e & \\
3b & 3c & 2a \\
5d & 6f & 4a+5e \\
\end{array}
$$

$A$ (4x4) $\text{nnz}A = 6$

$B$ (4x4) $\text{nnz}B = 6$

$C$ (4x4) $\text{nnz}C = 8$
Overview

- Sparse GEMM (SpGEMM) overview
- Two overheads and motivation
- Our SpGEMM algorithms
  - Algorithms: SpGEMM in DIA, COO and ELL formats
  - Performance improvements
- IA-SpGEMM: Input-aware Auto-tuning Framework
  - Input features (sparse features and density representation)
  - MatNet design
- Experimental results
- Conclusion
SpGEMM Overhead 1 – Memory access

- Due to the *sparsity of A*, a large number of *irregular memory access* occur.

- The memory access sequences of A & B using different formats are hugely different.
SpGEMM Overhead 2 – Sparse accumulation

• Because of the compressed sparse format, the result nonzeros are “accumulated” into C, but not “added” to predictable locations of C.
Three motivations

- These libraries are sensitive to the input sparse matrices and show huge performance differences.
- To some extent, SpGEMM is similar to sparse matrix-vector multiplication (SpMV) for irregular and indirect memory access pattern, many auto-tunning on SpMV have been dedicated. (OSKI, SPARITY)
- Memory access still occupies a certain amount of execution time.
Two solutions

- Reducing memory requirements or accelerating memory access on vector architecture by using classic storage formats, such as DIA, COO and ELL.
- Developing an autotuning framework to determine the optimal SpGEMM algorithm.
Overview

- Sparse GEMM (SpGEMM) overview
- Two overheads and motivation
- Our SpGEMM algorithms
  - Algorithms: SpGEMM in DIA, COO and ELL formats
  - Performance improvements
- IA-SpGEMM: Input-aware Auto-tuning Framework
  - Input features (sparse features and density representation)
  - MatNet design
- Experimental results
- Conclusion
Three format-special SpGEMM algorithms

DIA method:

- Step 1: coordinate transformation before multiplication.
- Step 2: output mapping after multiplication.

DIA Method

**Step 1.** Coordinate transformation

- Convert the diagonal coordinates to real coordinates and find the multiplied row number.

- Take the 2nd diagonal as an example

**Step 2.** Output diagonal mapping

- Symbolic Phase

**Step 3.** Memory allocation

- Numeric Phase

- Lock-free scheduling
Analysis of DIA method

Pros:

- Greatly reduces the overhead of memory access for diagonal matrix.
- Directly accumulates intermediate results to the target address without extra memory consumption.

Cons:

- Extra format conversion overhead.
- Not suitable for non-diagonal matrices.
Three format-special SpGEMM algorithms

COO method:

- Step 1: divides matrix A into k parts by row and matrix B into k parts by column before multiplication.
- Step 2: all the partial results are merged after multiplication.
Analysis of COO method

Pros:

• It greatly reduces the length of dense vector than that of the SPA method.

Cons:

• It brings additional overhead for partitioning and merging matrices.
Three format-special SpGEMM algorithms

ELL method:

- Step 1: Since each line of ELL format contains same non-zero number, so this format makes it possible for reducing the overhead of symbol phase.

- Step 2: The allocated memory space of C is used as hash table to accumulate intermediate results.
Analysis of ELL method

Pros:

• The symbolic phase can make full use of the SIMD instructions to speed up the efficiency of loading and assigning data.

• In numeric phase, the allocated memory space of C benefits the advantage of hash table without memory consumption.

Cons:

• Need to strictly meet the ELL format requirements.
Performance improvements

• Observation 1: Huge performance differences by various inputs, formats, algorithms and platforms.

• Observation 2: No single algorithm can constantly contribute the best performance

• Observation 3: It is difficult to obtain high prediction accuracy by using traditional machine learning algorithm.

| Method                  | Dominance |   | Dominance |   | Percentage | Average Speedup | Speedup by "Ideal Tool"
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>MKL (Baseline)</td>
<td>2874</td>
<td>-</td>
<td>35.07%</td>
<td>-</td>
<td></td>
<td>72.04x</td>
<td>8.94x</td>
</tr>
<tr>
<td>DIA method</td>
<td>491</td>
<td>1107</td>
<td>5.99%</td>
<td>13.51%</td>
<td></td>
<td>7.63x</td>
<td></td>
</tr>
<tr>
<td>COO method</td>
<td>283</td>
<td>506</td>
<td>3.45%</td>
<td>6.17%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ELL method</td>
<td>1496</td>
<td>2879</td>
<td>18.26%</td>
<td>35.13%</td>
<td></td>
<td>9.92x</td>
<td></td>
</tr>
<tr>
<td>SPA vector-based</td>
<td>259</td>
<td>748</td>
<td>3.16%</td>
<td>9.13%</td>
<td></td>
<td>1.31x</td>
<td></td>
</tr>
<tr>
<td>Hash-based</td>
<td>2150</td>
<td>4307</td>
<td>26.24%</td>
<td>52.56%</td>
<td></td>
<td>6.37x</td>
<td></td>
</tr>
<tr>
<td>Hcap-based</td>
<td>642</td>
<td>1951</td>
<td>7.83%</td>
<td>23.81%</td>
<td></td>
<td>6.21x</td>
<td></td>
</tr>
<tr>
<td>MKL (Baseline)</td>
<td>1708</td>
<td>-</td>
<td>20.96%</td>
<td>-</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>DIA method</td>
<td>745</td>
<td>1544</td>
<td>9.14%</td>
<td>18.94%</td>
<td></td>
<td>346.0x</td>
<td>46.16x</td>
</tr>
<tr>
<td>COO method</td>
<td>342</td>
<td>586</td>
<td>4.20%</td>
<td>7.19%</td>
<td></td>
<td>8.20x</td>
<td></td>
</tr>
<tr>
<td>ELL method</td>
<td>1989</td>
<td>3044</td>
<td>24.40%</td>
<td>37.35%</td>
<td></td>
<td>32.96x</td>
<td></td>
</tr>
<tr>
<td>SPA vector-based</td>
<td>830</td>
<td>2529</td>
<td>10.18%</td>
<td>31.03%</td>
<td></td>
<td>1.58x</td>
<td></td>
</tr>
<tr>
<td>Hash-based</td>
<td>1757</td>
<td>2363</td>
<td>21.56%</td>
<td>28.99%</td>
<td></td>
<td>21.40x</td>
<td></td>
</tr>
<tr>
<td>Hcap-based</td>
<td>779</td>
<td>1015</td>
<td>9.56%</td>
<td>12.45%</td>
<td></td>
<td>12.18x</td>
<td></td>
</tr>
<tr>
<td>cuSPARSE (Baseline)</td>
<td>3827</td>
<td>-</td>
<td>51.07%</td>
<td>-</td>
<td></td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>CUSP</td>
<td>208</td>
<td>769</td>
<td>2.78%</td>
<td>10.26%</td>
<td></td>
<td>6.27x</td>
<td>2.40x</td>
</tr>
<tr>
<td>NSPARSE</td>
<td>3459</td>
<td>3525</td>
<td>46.16%</td>
<td>47.04%</td>
<td></td>
<td>3.71X</td>
<td></td>
</tr>
</tbody>
</table>
Overview

• Sparse GEMM (SpGEMM) overview
• Two overheads and motivation
• Our SpGEMM algorithms
  • Algorithms: SpGEMM in DIA, COO and ELL formats
  • Performance improvements
• IA-SpGEMM: Input-aware Auto-tuning Framework
  • Input features (sparse features and density representation)
  • MatNet design
• Experimental results
• Conclusion
The collection phase includes extracting two patterns of input and executing all the algorithms. The training phase generates the MatNet model by two-way strategy.

Two respective inputs:
1. fine-grained: sparse features
2. coarse-grained: density representation
Input features

Sparse features relate to the distribution and characteristic of four formats.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>row, col, nnz</td>
<td>the number of rows, columns and non-zero elements</td>
</tr>
<tr>
<td>nnz_ratio</td>
<td>the ratio of non-zero elements in CSR format</td>
</tr>
<tr>
<td>max, min, average</td>
<td>the maximum, minimum and average of non-zero elements</td>
</tr>
<tr>
<td>VAR</td>
<td>the variance of non-zero elements</td>
</tr>
<tr>
<td>dia_num</td>
<td>the number of diagonals in DIA format</td>
</tr>
<tr>
<td>dia_ratio</td>
<td>the number of diagonals divided by all the diagonals</td>
</tr>
<tr>
<td>dia_pad,  ell_pad</td>
<td>the ratio of padding data in DIA and ELL format</td>
</tr>
<tr>
<td>CV</td>
<td>the coefficient of variation of non-zero elements</td>
</tr>
</tbody>
</table>

Density representation as primary image input of CNN represents snapshot matrix which abstracts most of matrix features.
MatNet design

Two respective inputs:
The red box represents the four input data of the model.

Output: Predicted probability of various formats and algorithms

Connection layer:
Connecting coarse-grained and fine-grained patterns
Overview

• Sparse GEMM (SpGEMM) overview
• Two overheads and motivation
• Our SpGEMM algorithms
  • Algorithms: SpGEMM in DIA, COO and ELL formats
  • Performance improvements
• IA-SpGEMM: Input-aware Auto-tuning Framework
  • Input features (sparse features and density representation)
  • MatNet design
• Experimental results
• Conclusion
Experiment setup

Platform:
- **Intel Xeon E5-2690 v4**, 2 processors, 28 cores @2.60 GHz, 35 MB LLC, 2*4 channel memory, 136.6 GB/s Bandwidth. MKL v19.0.
- **AMD EPYC 7501**, 2 processors, 64 cores@2.00 GHz, 64 MB LLC, 2*8 channel memory, 341 GB/s Bandwidth.
- **nVidia Tesla P100**, 56 SMs@1328 MHz, 4096 KB L2, 732 GB/s bandwidth. cuSPARSE v8.0.61, CUSP v0.5.1.

Dataset:
- 2726 matrices from the SuiteSparse matrix collection are used to randomly construct 8195 matrix pairs for evaluation by over 220 GB total size

Baseline:
- CPU: Intel MKL v19.0.0.117 and hash-based method
- GPU: NVIDIA cuSPARSE v8.0.61 and NSPARSE.
Loss and accuracy

- With the increase of training times, prediction accuracy increases and loss decreases.
- The training process of various formats and algorithms is related to amount of the training data.
- The prediction accuracy of three platforms can reach more than 93%.

![Graphs showing training and accuracy for Intel, AMD, and NVIDIA platforms.]
Speedup results

- DIA format brings the greatest performance improvement, up to 346.0x
- Overall performance improvement: 3.27x, 13.17x and 2.23x (with overhead)
Overview

• Sparse GEMM (SpGEMM) overview
• Two overheads and motivation
• Our SpGEMM algorithms
  • Algorithms: SpGEMM in DIA, COO and ELL formats
  • Performance improvements
• IA-SpGEMM: Input-aware Auto-tuning Framework
  • Input features (sparse features and density representation)
  • MatNet design
• Experimental results
• Conclusion
Conclusion

1. We propose a variety of SpGEMM algorithms for DIA, COO and ELL format, and compare performance of various algorithms.

2. We present an Input-aware Auto-tuning Framework for SpGEMM (IA-SpGEMM), which could automatically determine the best format and algorithm for any sparse matrix pairs.

3. The results show that IA-SpGEMM yields better performance than four other state-of-the-art libraries. And we also expect more sparse and input-sensitive algorithms can be inspired by our method.
Thanks!

Any Questions?