Auto-Tuning Strategies for Parallelizing Sparse Matrix-Vector (SpMV) Multiplication on Multi- and Many-Core Processors

Kaixi Hou, Wu-chun Feng
{kaixihou, wfeng}@vt.edu

Shuai Che
Shuai.Che@amd.com
Sparse Matrix Storage

- Sparse matrix: the majority of entries are zeros
- An efficient storage only records nonzero entries
  - Need to ignore zero entries and put all nonzeros together

A 4x4 Matrix

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>8</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

Dense Representation

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>3</th>
<th>5</th>
<th>8</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>2</td>
<td>5</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

Sparse Representation

Ignore all the zeros

Only store the 6 nonzeros
Sparse Matrix-Vector Multiplication (SpMV)

- Problem definition: multiply a sparse matrix $A$ and a dense vector $x$, and return the result as a dense vector $y$

\[
\begin{align*}
\begin{array}{cccc}
1 & & & \\
3 & 2 & & \\
& 5 & 8 & 4 \\
\end{array}
& \times
\begin{array}{c}
a \\
b \\
c \\
d \\
\end{array}
= \\
\begin{array}{c}
1a \\
3a+2c \\
0 \\
5b+8c+4d \\
\end{array}
\end{align*}
\]

Sparse matrix $A$ \hspace{1cm} Dense vector $x$ \hspace{1cm} Dense vector $y$
Factors Affecting SpMV Performance

- **Storage formats of sparse matrix A**
  - CSR, ELLPACK, DIA, COO, BCCOO, BRC, CSR5, etc.

- **Parallelization strategies**
  - Different formats correspond to different algorithms
  - Even same format can lead to different parallel strategies, e.g., granularities of parallelism, optimizations, etc.

- **Input sparse matrices themselves**

[Sedaghati_ICS15, Li_PLDI13, etc.]

This work’s focus
Motivating Examples

- How input sparse matrices and parallelization strategies affect performance?
Motivating Examples

- How input sparse matrices and parallelization strategies affect performance.

**Different Input Sparse Matrices** prefer different parallelization strategies.

**Different Nonzero Row Sizes** prefer different parallelization strategies.
Motivating Examples

- How input sparse matrices and parallelization strategies affect performance:

  - Different parallel strategies

  Therefore, we need a framework to automatically look for the best parallel strategies for different sparse matrices

Different Input Sparse Matrices prefer different parallelization strategies

Different Nonzero Row Sizes prefer different parallelization strategies
Outline

• Sparse Matrix and SpMV
• Motivation

• Background
  – CSR format

• Reconfigurable SpMV Method
  – Binning schemes
  – Kernel choices
• SpMV Data Mining Framework

• Evaluations & Conclusion
Compressed Sparse Row (CSR) Format

- Widely-used sparse matrix format
  - Store row pointers, column indices, and nonzero values

CSR Representation
Reconfigurable SpMV Method

- Overview of our SpMV method
  - Binning schemes and kernels can be customized
Binning Schemes

- For load balance, we group (permute) rows into different bins, according to their nonzero numbers.
- However, how to choose correct granularities for binning?
  - Small granularities lead to high binning overhead.
  - Large granularities lead to high row variance in the same bin.

![Binning Execution Time Graph](image)
• In our method, we treat multiple neighboring rows as a single “virtual” row
  – We have a set of candidate granularity units (denoted as $U$) to determine the number of neighboring rows
• In our method, we treat multiple neighboring rows as a single “virtual” row
  – We have a set of candidate granularity units (denoted as $U$) to determine the number of neighboring rows

![Diagram showing binning schemes with rows r0 to r5 and candidate granularity units U=1 and U=2.](image-url)
In our method, we treat multiple neighboring rows as a single “virtual” row

- We have a set of candidate granularity units (denoted as $U$) to determine the number of neighboring rows.
Binning Schemes

• In our method, we treat multiple neighboring rows as a single “virtual” row
  – We have a set of candidate granularity units (denoted as $U$) to determine the number of neighboring rows

• Better locality, throughput, etc.
Kernel Choices

- Different SpMV kernels to process different types of rows
  - Assign a row to one thread
  - Assign a row to multiple threads (wavefront-level)
  - Assign a row to multiple wavefronts (thread-block-level)
Kernel Choices

- We use up to a thread block to process one nonzero row
- Current work only focuses on short and medium row sizes
  - Our bin-based method can easily be extended to support long rows (e.g., dynamic parallelism based method)

Majority of rows have a very small number of nonzeros
SpMV Data Mining Framework

• Overview of our data mining framework to look for the optimal binning policies and SpMV kernels

- Example
  - \( \text{gran}=10 \) nonzeros
  - \( \text{gran}=20 \) nonzeros

- Binning Schemes
- SpMV kernels

- Example
  - serial kern.
  - subvec kern.
  - ... vector kern.

- Best Binning & Kernel Selection
- Execute

- Training set (~2k matrices)
- Feature Selection
- Machine Learning Model
- Predict
- New matrix
- Feature Selection
The classification tool is C5.0 for data mining *

We select over 2K sparse matrices from the University of Florida sparse matrix collection as the training set
- 75% are used for training
- The rest are used for testing

The error rate of learning is 5~15%
- 1st stage of learning (for binning schemes) is around 5%
- 2nd stage (for parallelization strategies) is less than 15%

Finally, we have two generated rule-sets
- One is for how to select binning schemes
- Another is for how to select kernels for each bin

* https://www.rulequest.com/see5-info.html
Outline

• Sparse Matrix and SpMV
• Motivation

• Background
  – CSR format

• Reconfigurable SpMV Method
  – Binning schemes
  – Kernel choices

• SpMV Data Mining Framework

• Evaluations & Conclusion
Benchmark Suite

- We select 16 matrices from the University of Florida sparse matrix collection.
Experiment Platform

- AMD A10-7850K APU: A real HSA hardware
- It features **four 3.7 GHz CPU cores** and **eight 720MHz GPU compute units**
- Our system is equipped with 16 GB memory
- We use AMD Heterogeneous System Architecture (HSA) - Linux amdkfd v1.4 release
- We use CL Offline Compiler CLOC V0.9.5 (HSA 1.0F) with SNACK support
Performance Evaluation

• Speedups from our framework

- **Kernel-auto** is the kernel from our SpMV framework by automatically selecting binning and parallelization strategies
Performance Evaluation

- Speedups from our framework

  - **Kernel-auto** is the kernel from our SpMV framework by automatically selecting binning and parallelization strategies
  - Compared to **kernel-serial**, we can achieve 1.7x to 11.9x speedups
Performance Evaluation

• **Speedups from our framework**

  - **Kernel-auto** is the kernel from our SpMV framework by automatically selecting binning and parallelization strategies
  - Compared to **kernel-serial**, we can achieve 1.7x to 11.9x speedups
  - Compared to **kernel-vector**, we can get 1.2x to 52.0x speedups

Assign one row to each thread-block
Performance Evaluation

• Speedups from the prior state-of-the-art GPU SpMV “CSR-Adaptive”*

![Graph showing speedups for various matrices]

– Our SpMV can yield better performance over 10 out of 16 sparse matrices and achieve up to 1.9x speedups

* J. Greathouse, M. Daga, “Efficient Sparse Matrix-vector Multiplication on GPUs Using the CSR Storage Format”, SC 2014
Conclusion

• Proposed a SpMV framework using the machine learning model to automatically find the optimal parallel strategies
  – Focusing on the CSR format
  – Choosing the appropriate grouping policy to organize independent rows (as “virtual” rows) into different bins
  – Looking for the suitable kernels to process the bin rows

• Achieved significant performance improvements over the SpMV kernels using single kernel

• Achieved up to 1.9x speedups over other state-of-the-art SpMV kernels
Discussion & Future Work

• Grouping all rows to a single bin
  – Need more features of matrix to identify when to put all rows into a single bin

Different parallel strategies
Discussion & Future Work

- Grouping all rows to a single bin
  - Need more features of matrix to identify when to put all rows into a single bin

- Extending our work to fully utilize both CPU and GPU
  - High-volume bins on throughput-oriented processors
  - Low-volume bins on latency-oriented processors
Discussion & Future Work

- Grouping all rows to a single bin
  - Need more features of matrix to identify when to put all rows into a single bin

- Extending our work to fully utilize both CPU and GPU
  - High-volume bins on throughput-oriented processors
  - Low-volume bins on latency-oriented processors

THANK YOU!

More info: kaixihou@vt.edu