Energy Efficiency in Sparse Matrix Multiplication for different Hardware and Algorithms

Kevin Rohan
Dept. of Electrical and Computer Engineering
Carnegie Mellon University

Wilson Sa
Dept. of Electrical and Computer Engineering
Carnegie Mellon University

***Abstract*— Sparse Matrix-Vector Multiplication (SpMxV) is widely used in modern day applications, like feature extraction, data analytics, numerical methods etc. The throughput obtained in sparse matrix multiplication is lower compared to dense matrix multiplication due to the compression format used to represent these matrices. This project proposes a comparative analysis between the various hardware and algorithms used to compute the product of such matrices. Computation is performed on FPGA, CPU and GPU. The hardware is designed to exploit the parallelism in the algorithms used. The algorithm used is an Outer Product based Sparse Matrix Multiplication. This algorithm is of high memory bandwidth, runs in parallel and reduces the number of memory accesses. This reduces the overhead.**

Keywords—SpMxV, FPGA, GPU, CPU, Energy Efficiency, Parallel Computing

I. INTRODUCTION

Sparse Matrix arises in many situations in modern day applications. Many of the applications exploit the property of sparse matrices storing data very sparsely. This reduces the data stored very significantly which is very useful in many of the trends like big data, distributed computing etc. The computation of information from these matrices involves large sparse matrix multiplications. There are dedicated kernels in the computers to determine the product. The computational throughput is not as good as dense matrix multiplication due to the irregularities in the memory access patterns and compression formats used to store them.

In this project, the possibilities of implementation of sparse matrix multiplication is explored in CPU, GPU and FPGA. The hardware is used as a co-processor to the existing system and the operation is carried out. The efficiency is computed for different forms of implementation.

Algorithmic optimizations is explored by using various forms of compressed matrix representations. Matrix multiplication is explored for both the case of inner product and outer product based operations. These algorithms are implemented on the proposed architecture which was mentioned in the previous paragraph.

The rest of the report is organized as follows, section 2 provides a description of the related work which has already been implemented, section 3 provides details of the technical description of the project, section 4 provides the schedule in which we would like to proceed, milestones and division of work between each other.

II. RELATED WORK

[1] A scalable sparse matrix-vector multiplication kernel for energy-efficient sparse-blas on FPGAs.

This paper provides the details related to the various forms of sparse matrix representations. The basic hardware we are implementing in our project is derived from this paper where a FPGA is used as a co-processor to the existing system and a comparative analysis is provided between different hardware used.

[2] OuterSPACE: An Outer Product based Sparse Matrix Multiplication Accelerator.

This paper proposes an algorithm which exploits parallelism to improve the performance of sparse matrix multiplication, where an outer product is used instead of an inner product based multiplication.

We are planning to implement the algorithm in [2] by using the hardware in [1].

III. TECHNICAL DESCRIPTION

A. *Sparse Matrix Representations:*

Sparse Matrices are represented in three forms as shown in figure 1:

Coordinate list (COO)

COO stores a list of (row, column, value) tuples.

Compressed sparse row (CSR, CRS or Yale format)

The *compressed sparse row* (CSR) or *compressed row storage* (CRS) format represents a matrix by three (one-dimensional) arrays, that respectively contain nonzero values, the extents of rows, and column indices.

Compressed sparse column (CSC or CCS)

CSC is similar to CSR except that values are read first by column, a row index is stored for each value, and column pointers are stored

$$\begin{bmatrix} 1 & 0 & 2 \\ 4 & 0 & 0 \\ 0 & 3 & 0 \end{bmatrix}$$

COO:

Row	2	0	1	0	0
Column	1	0	0	2	0
Value	3	3	4	2	-2

CSC:

Row	0	1	2	0
Column	0	2	3	4
Value	1	4	3	2

Figure 1. Sparse Matrix Representations

B. *Hardware Optimizations:*

Conventional Hardware architecture uses a dual port RAM to compute the product of the matrices. The dual port RAM is connected with a multiplier followed by an adder to compute the product of matrices. The disadvantage in this method are Low Utilization rate, Uncertain Memory Access Patterns as CSR is used and Cache Overhead is very high.

In the proposed architecture as seen in figure 2, a co-processor is used to the existing processor. This co-processor provides the computation space where in CSC pattern is used for matrix multiplication the speed up is obtained as each element in the column is represented in a parallel computation block. This reduces the Cache overhead.

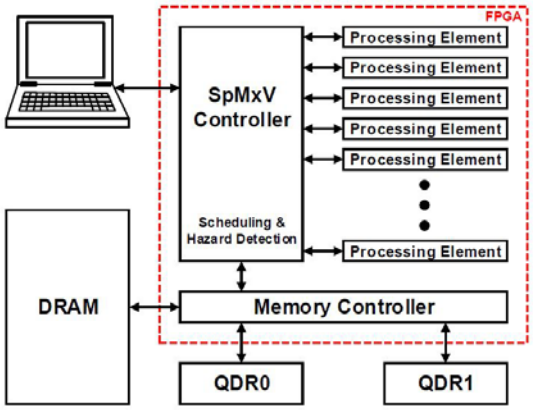

Figure 2. SpMxV KernelHardware Optimizations:

C. *Software Optimizations:*

To optimize Sparse Matrix Multiplication (SMM) operation, we will carry out an analysis on the common pitfalls of various algorithms on different pieces or hardware architectures. One major bottleneck we will take a critical look at is redundant memory accesses which occur in traditional Sparse Matrix Multiplication algorithms. If we are able to identity redundant memory access as the major bottleneck, we can test different commonly used algorithms to identify the

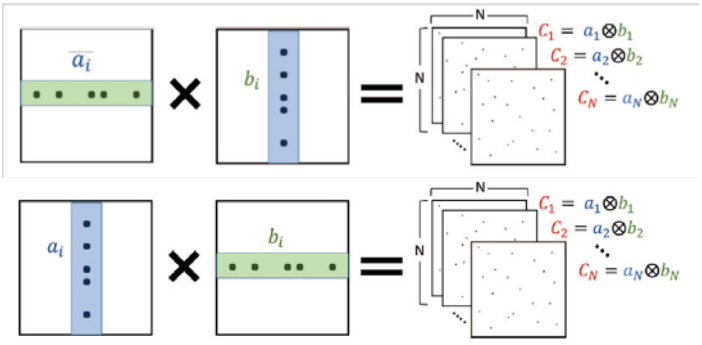

Figure 3. Top row shows an outer, bottom row illustrates an inner matrix multiplication

operations which will most greatly reduce this memory access. We may attempt different permutations of algorithms which may exploit different combinations between multiply and merge subset of operations.

To get started, we will use the gem5 simulation tool to test various hardware architectures through a diverse set of metric multiplication operations using SuiteSparse collection. We also plan on using the Stanford Network Analysis Project as a tool measure throughput and performance of the operations.

Using the SuiteSparse collection, we plan to implement the following four major algorithms for test: Intel Math Kernel Library (MKL), OuterSPACE [2],cuSPARSE [3], and CUSP

[4]. The algorithms will mainly differ by methodologies of carrying out matrix multiplication. Traditionally, matrix-to-matrix multiplication is performed using an inner dot product approach. In an inner matrix multiplication, the first matrix 'A' is decomposed to rows, multiplied by the second matrix 'B' decomposed to columns. The algorithm most closely resembling this is MKL approach. Conversely, in an outer dot product approach, the first matrix 'A' is decomposed to columns, multiplied by the second matrix 'B' decomposed to rows. The OuterSPACE algorithm most closely resembles this approach. Further, once the matrices are decomposed by column and row, the matrices are *multiply* and *merge* operations to combine into a final result. These differences is where the cuSPARSE and CUSP algorithms differ from the others.

The OuterSPACE paper concludes that the outer dot product approach has an improvement of throughput between 7.9x and 14.0x on comparable architectures, while keeping power budgeted and area constrained. We plan on carrying a similar comparison, as well as analysis of these software optimizations in connection with the parallelized hardware architecture optimization efforts above.

IV.SCHEDULE

October 10: Receive FPGA hardware (Rohan)
October 16: Get the conventional Sparse Matrix Matrix Multiplication Implementation running on the CPU.
October 16: Complete gem5 analysis of different algorithms (Sa)
October 23: Presentation – II (Sa/Rohan)
November 2: Complete hardware comparison tests on FPGA (Rohan)
November 15: Complete integration between algorithm and hardware to begin tests (Sa/Rohan)
November 30: Complete comparisons with algorithms across different architectures (Sa/Rohan)
December 4 : Paper Presentations (Sa/Rohan)
December 6 : Project Posters and Demo (Sa/Rohan)
December 12: Complete report (Sa/Rohan)

V. REFERENCES

[1] Dorrance, R., Ren, F. and Marković, D., 2014, February. A Scalable Sparse Matrix-vector Multiplication Kernel for Energy-efficient Sparse-blas on FPGAs. In Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays (pp. 161-170).

[2] Pal, S., Beaumont, J., Park, D.H., Amarnath, A., Feng, S., Chakrabarti, C., Kim, H.S., Blaauw, D., Mudge, T. and Dreslinski, R., 2018, February. OuterSPACE: An Outer Product based Sparse Matrix Multiplication Accelerator. In 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA) (pp. 724-736).

[3] (2014) cuSPARSE Library. https://developer.nvidia.com/cuSPARSE.

[4] N. Bell, S. Dalton, and L. N. Olson, "Exposing fine-grained parallelism in algebraic multigrid methods," SIAM J. Scientific Comput., 2011.