Midterm Status Report

Research Paper:

Gamma: Leveraging Gustavson's Algorithm to Accelerate Sparse Matrix Multiplication

by Attaluri et al. (2021)

Pete Miglio, Osvaldo Garcia and Michael Skarlatov

Main Goal

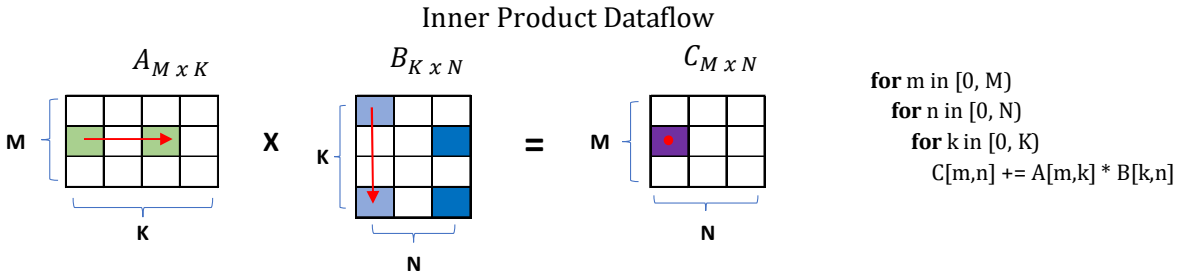
The goal of this project is to emulate and test the sparse matrix multiplication dataflows as described in the research paper. Specifically, these dataflows will be tested: inner-product, outer-product, and Gamma (utilizing Gustavson's Law).

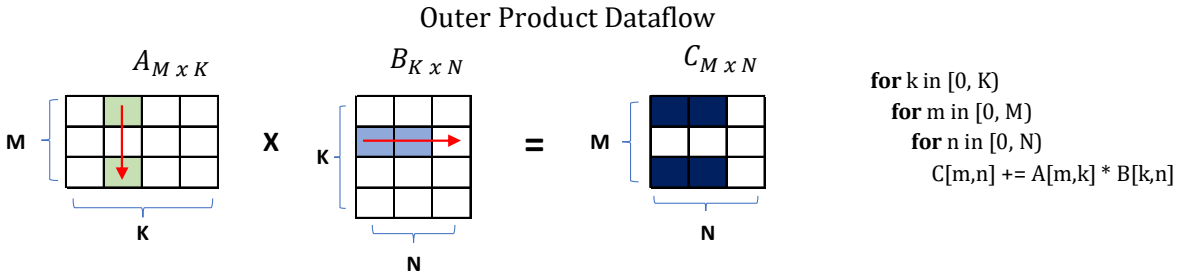
Technology summary

The programming language picked for our project is C++. The source code is hosted in a private repository hosted in GitHub. Windows is the OS and development environment, and Visual Studio 2019 is the main IDE being used.

Inner and Outer Product Dataflows

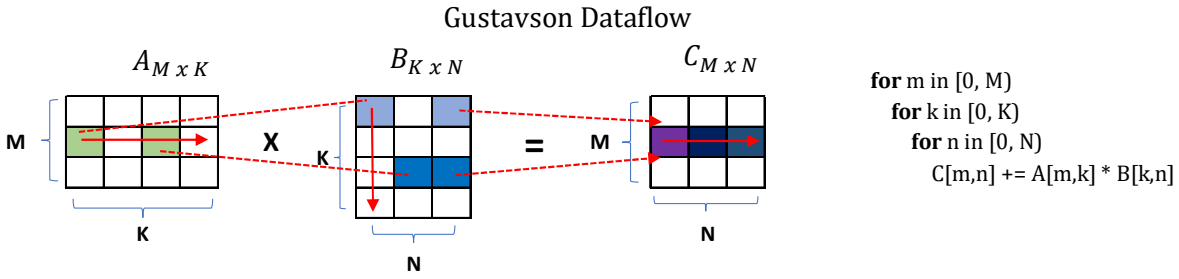
The dataflow is determined by the level of co-iteration. In the inner product dataflow, the co-iteration happens at the innermost loop, in the outer product in the outermost loop and in the middle loop for Gustavson. Originally, our team implemented inner and outer product dataflows without row compression to get a basic understanding of how the dataflows worked. Afterward, mechanisms were designed to compress rows, removing zeroes, and accelerating sparse matrix multiplication. New inner and outer product methods were made and tested with small sparse matrices, and they work as intended. Timers were set in place as well to record the times taken to execute the operations.





Gamma:

The Gustavson algorithm was implemented using the pseudo-code provided in the research paper. The algorithm itself is very simple. The Gustavson dataflow is determined by the middle loop co-iteration and is a row stationary dataflow. It computes the output matrix one row at the time, linearly combining the rows of B for which the rows of A have no zero coordinates. According to the research paper, the Gustavson algorithm is more efficient than inner and outer product. Our team is working on validating these conclusions. At this stage Gustavson has not been tested with compressed sparse matrices.



Compliant with Gamma architecture, our implementation allocates memory dynamically, to store matrices A and B.

```
// allocate matrix
float** matrix = new float* [m];
```

Gamma is more than just Gustavson product. It combines this algorithm with a scheduler, processing elements and FiberCache. FiberCache is an interface built upon the cache that ensures the data fetched is not evicted to ensure the reads are hits in most of the cases and reduce latency.

Generating large sparce matrices:

A class was made in C++ to generate matrices and sparce matrices of varying size. Large sparce matrices can be made using this class but testing on the large sparce matrices has not been done yet. This is one of the most recent additions to the code and will be important for testing the product functions in the future. The class is held in a header file and can initialize a matrix of any

size, return values at certain positions in the matrix, set values at positions in the matrix, get the rows and columns length of the matrix, and print out the matrix.

Future Plans

So far, all our inner and outer product matrix testing has been done on small sparse matrices, so a good next step would be figuring out how to load large sparse matrices as testing material. Once all sparse matrix multiplication methods are implemented, comparison testing will begin, and actual statistics will be recorded and compared to the results of the other dataflows. Our team will not implement any hardware level design like FiberCache and the processing elements scheduler. Instead, we are focusing on the matrix multiplication methods. Our plans also include adding parallelism (must likely using OpenMP) to compare sequential and parallel results with those of the research paper.

Updated team schedule (weekly milestones)

Week	Monday of that Week	Work to be done	Progress	Notes
Week 1	09/20/2021	Research Paper Presentation on 9/23	Completed	
Week 2	09/27/2021	Start working on Gamma and other sparse matrix multiplication methods/dataflows	Completed	
Week 3	10/04/2021	Develop sparse matrix multiplication methods and research existing libraries	In progress	Researching parallel API and methods to use with matrix multiplication
Week 4	10/11/2021	Finish out sparse matrix multiplication methods	In progress	Develop accurate timing method, to measure performance
Week 5	10/18/2021	Create test datasets and run them against the multiplication methods. Benchmark performance.	In progress	Find external datasets used in research paper
Week 6	10/25/2021	Midterm Status Report on 10/28	Completed	
Week 7	11/01/2021	Work on Gamma method	Started	Combine sparse matrix multiplication methods
Week 8	11/08/2021	Finish out Gamma method	Upcoming	
Week 9	11/15/2021	Evaluate datasets against Gamma method	Started	
Week 10	11/22/2021	Work on Final Report and Final Presentation	Not started yet	
Week 11	11/29/2021	Final Presentation on 11/30, Final Report on 12/2	Upcoming	