Optimization Techniques for Sparse matrix–vector Multiplication on GPUs

Hardik Rana
16CO138
hardikrana276@gmail.com

Harshal Shinde
16CO223
hsharshal9@gmail.com

Date: 25/10/2018

1.Problem Statement

Implementation of the optimisation techniques for sparse matrix vector multiplication as proposed in the “Arash Ashari, Naser Sedaghati, John Eisenlohr, Srinivasan Parthasarathy, P. Sadayappan, Fast Sparse Matrix-Vector Multiplication on GPUs for Graph Applications”.

Sparse matrix-vector multiplication (SpMV) is a widely used computational kernel. The most commonly used format for a sparse matrix is CSR (Compressed Sparse Row), but a number of other representations have recently been developed that achieve higher SpMV performance. However, the alternative representations typically impose a significant preprocessing over-head. While a high preprocessing overhead can be amortized for applications requiring many iterative invocations of SpMV that use the same matrix, it is not always feasible – for instance when analyzing large dynamically evolving graphs.

Ashari et al proposed an algorithm named ACSR that uses the standard CSR format for Sparse-Matrix multiplication. In *ACSR*, thread divergence is reduced by grouping rows with similar number of zeros into a bin. It also uses dynamic parallelism for rows that span a wide range of non zero counts.

SpMV is also implemented using cuSparse library for comparison with ACSR.So our goal is to implement ACSR algorithm for sparse matrix-vector multiplication and compare it with SpMV implementation using cuSparse library using by running those implementations on different datasets and to compare results of them.

2.Project Execution Plan

Following sections describe the implementation framework which we are going to use, input datasets for running implementation of SpMV and expected outputs.

2.1 Implementation Framework

The code for the implementation is written in Cuda C. The code is run on NVIDIA GPUs. The compiler used is NVIDIA Cuda compiler.

2.2 Inputs

We will take standard datasets for running parallel implementation from the following website https://sparse.tamu.edu/. They will be given as input to the compiled cuda code. Following are some of the datasets which we are going to use to evaluate our parallel algorithm:

- Amazon-2008
- in-2004
- flickr
- hollywood-2009
- in-2004
- cnr-2000
- eu-2005

2.3 Expected Outputs

After running parallel implementation on above mentioned datasets we will plot one graph for datasets vs runtime of algorithm for that datasets. The ACSR implementation of SPMV must take less time than BCCOO,BRC,TCOO and HYB. CUDA timing API will be used to measure time taken to execute algorithm and we can use speed up calculation to show our algorithm efficiency.

3. Project Timeline

- **Week1 : (October 8-14,2018):**
  - ➢ Went through the list of paper and resources of each paper sir has given.
  - ➢ Chosen this optimization techniques of SPMV as our project.
- **Week2 : (October 15-21,2018):**
  - ➢ Went through the paper of optimization techniques of SPMV in detail
  - ➢ Learned about different sparse matrix representations [CSR,COO, ELL,HYB].
  - ➢ Learned about different optimization techniques presented in the Paper.
- **Week3 : (October 22-28,2018):**
  - ➢ Went through the paper of Adaptive CSR [ACSR] and learned how it will reduce the time of doing sparse matrix-vector multiplication
  - ➢ Learned about the concepts of dynamic parallelism and binning
  - ➢ Go through the algorithm ACSR using [which is presented in the Paper]
  - ➢ Prepared project proposal report
- **Week4 : (October 29- November 4,2018):**
  - ➢ To implement the ACSR algorithm presented in paper in CUDA and to implement the SPMV implementation using NVIDIA's cuSparse library.
- **Week5 : (November 5-11, 2018):**
  - ➢ To complete implementation of previous week if anything is remaining
  - ➢ To run above implementation on the datasets mentioned in the proposal report and to compare the performance of two algorithms
- **Week6**: **(November 12-15, 2018)**
  - ➢ To prepare the final report based on the results obtained

4.Work Distribution

The work divided is based on the basis of our understanding upon reading the paper.Please do not consider this as a final, as it will changed when we actually sit for implementation.

SNo	Work	Assignee
1	Reading the paper and noting down the understanding of each part and algorithm	Hardik,Harshal (independently)
2.	Explain understanding of paper to each other and point the core important points,and prepare the report of understanding of algorithms	Hardik,Harshal
3.	Implement the ACSR (adaptive CSR) algorithm for Sparse matrix-vector multiplication	Hardik
4.	SPMV implementation using NVIDIA's cuSparse library	Harshal
5.	Run the code of both implementation of different standard datasets and not down the performance of both ACSR and cuSparse implementation of SPMV.	Hardik,Harshal (independently)
6.	Compare the analysis made by each one of us and plot the final performance analysis	Hardik,Harshal
7.	Final Report	Hardik,Harshal

5.References:

1.Maggioni and Berger-Wolf, Optimization techniques for sparse matrix – vector multiplication on GPUs, J. Parallel Distrib. Comput. 93–94(2016)66–86.
Check Paper

2.Fast Sparse Matrix-Vector Multiplication on GPUs for Graph Applications
Check Paper