Milestone 1: Tensor core accelerated database Operations

Akshaya Bindu Gowri
Abhishek Singh
Akshata Bobade
Ayushi Dani
Maneendra Hetti Perera
Vegen Shanti Dsilva

Otto-von-Guericke-University Magdeburg, Germany

May 12th, 2020

Overview

- Introduction
- Motivation
- Literature survey
- Project Goals and Initial research questions
- Strategies for join and set operations
- Timeline of the project

Introduction

1. Tensorcore is a Tensor Processing Units (TPU) implemented by Nvidia.
2. Tensorcores can do 4*4 matrix multiplication operation per GPU cycle and are used for Machine Learning / Deep Learning tasks
3. They have massive computation parallelism and high bandwidth memory.
4. We utilize CUDA TCU API for this project

Motivation

To process huge amount of data in the era of Big Data we need some parallel processing techniques to speed up our processing. For this reason we use new GPU's equipped with Tensorcore, exploiting them for database operations.

Literature Survey 1

Analyzing GPU Tensor Core Potential for Fast Reductions

This paper discusses the idea of performing faster matrix multiply accumulate(MMA) using Tensor Cores.

- The idea is to reduce a set of n numbers as a set of m x m (where m is the linear size of involved matrices) MMA tensor core operations
- In comparison to traditional GPU based computing, this approach significantly reduces the number of steps and increases the speed
- Tensor core parallel reduction allows many MMA operations to occur simultaneously in parallel

References - Carrasco, Roberto Vega, Raimundo Navarro, Cristobal. (2019). Analyzing GPU Tensor Core Potential for Fast Reductions. 10.29007/zlmg. .

Literature Survey 2

NVIDIA Tensor Core Programmability,Performance and Precision

- Matrix Multiplication
  - Tiled Matrix Multiply with CUDA 9 WMMA
  - CUTLASS (CUDA Templates for Linear Algebra Subroutines) – hide GPU memory latencies – max performance when N = 16, 384
  - NVIDIA cuBLAS - provides GEMM routines for Tensor Cores - max performance when N = 8, 192
- Batched Matrix Multiplications
  - TiledBatched sgemm API of NVIDIA cuBLAS (cublasSgemmBatched()).
  - CUTLASSBatched GEMM is not supported by NVIDIA Tensor Cores

Reference - S. Markidis, S. W. D. Chien, E. Laure, I. B. Peng and J. S. Vetter, "NVIDIA Tensor Core Programmability, Performance Precision," 2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), Vancouver, BC, 2018, pp. 522-531, doi: 10.1109/IPDPSW.2018.00091.

Project goals:

- To first implement database operation like joins in tensor core , and then try to implement set intersection and filtering operations,thereby improving the performance of the database operations.

Initial research questions:

- How should join operations be implemented using Tensor core multiplication?
- How can we incorporate set operations in tensor cores?

Strategy 1: Computing joins faster with matrix multiplications in tensor cores

In the first step, we transfer the data from CPU to GPU. So we transfer the below R and S tables to GPU at once.

$$R = \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \end{array} \quad S = \begin{array}{c} \begin{array}{cc} b & c \end{array} \\ \begin{pmatrix} 4 & 1 \\ 6 & 2 \end{pmatrix} \end{array}$$

Inside Tensorcore

$$R = \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{pmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{pmatrix} \end{array} \quad S = \begin{array}{c} \begin{array}{cc} b & c \end{array} \\ \begin{pmatrix} 4 & 1 \\ 6 & 2 \end{pmatrix} \end{array} \quad RXS = \begin{array}{c} \begin{array}{cc} a & b \end{array} \\ \begin{pmatrix} 0 & 4 \\ 0 & 5 \\ 0 & 6 \end{pmatrix} \end{array} \begin{array}{c} c \\ b \end{array} \begin{pmatrix} 0 & 0 & 0 \\ 1/4 & 1/6 & 0 \end{pmatrix}$$

$$RXS = \begin{array}{c|c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 & 4/6 & 0 \\ 5/4 & 5/6 & 0 \\ 6/4 & 1 & 0 \end{pmatrix} \end{array}$$

Inside Tensorcore

$$RXS = \begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 & 4/6 & 0 \\ 5/4 & 5/6 & 0 \\ 6/4 & 1 & 0 \end{pmatrix} \end{array}$$

Search for the presence of 1 in every row in parallel

$$\begin{array}{c c} & 1 \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 1 \\ 4/6 \\ 0 \end{pmatrix} \end{array} \quad \begin{array}{c c} & 2 \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 5/4 \\ 5/6 \\ 0 \end{pmatrix} \end{array} \quad \begin{array}{c c} & 3 \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{pmatrix} 6/4 \\ 1 \\ 0 \end{pmatrix} \end{array}$$

Strategy 1: Computing joins faster with matrix multiplications in tensor cores

In the final step, we transfer the join result from GPU to CPU at once.

$$RXS = \begin{array}{c c} & \begin{array}{ccc} a & b & c \end{array} \\ \begin{array}{c} 1 \\ 2 \end{array} & \begin{pmatrix} 1 & 4 & 1 \\ 3 & 6 & 2 \end{pmatrix} \end{array}$$

Strategy 2: Computation of set operations in tensor cores

	firstname
row 0	Michael
row 1	Thomas

Table A

	firstname
row 0	Mike
row 1	Tom
row 2	Michael

Table B

firstname	fisrtname	flag(join)	flag (intersect)
Michael	Mike	0	1
Michael	Tom	0	
Michael	Michael	1	
Thomas	Mike	0	0
Thomas	Tom	0	
Thomas	Michael	0	

Table A × Table B

Flag of A join B with the join condition: A.firstname=B.firstname

Flag of A intersect B

Huang, Y. F., Chen, W. C. (2015). Parallel Query on the In-Memory Database in a CUDA Platform. Proceedings - 2015 10th International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, 3PGCIC 2015. https://doi.org/10.1109/3PGCIC.2015.34

Timeline

Sr. No	Task Item	Expected Start Date	Expected End Date
1	Literature Survey and Initial Planning	30-04-2020	05-07-2020
2	First Milestone presentation ppt review and feedback	05-07-2020	05-11-2020
3	First Milestone presentation	05-12-2020	05-12-2020
4	Designing the approach and feedback	13-05-2020	25-05-2020
5	Second Milestone presentation document and feedback,changes	13-05-2020	28-05-2020
6	Implementing Select -where operation	25-05-2020	28-05-2020
7	Second Milestone presentation	06-01-2020	06-01-2020
8	Unit Testing Item(5)	29-05-2020	30-05-2020
9	Review and feedback regarding Item (5),changes	28-05-2020	30-05-2020
10	Documentation regarding Item(5)	29-05-2020	31-05-2020
11	Implementing Join operation	30-05-2020	15-06-2020
12	Unit Testing Item (11)	15-06-2020	18-06-2020
13	Review and feedback regarding Item (11),changes	18-06-2020	20-06-2020
14	Documentation regarding Item(11)	19-06-2020	20-06-2020
15	Third Milestone documentation ,review, changes	06-01-2020	21-06-2020
16	Third Milestone presentation	22-06-2020	22-06-2020
17	Implementing Set Operations	21-06-2020	29-06-2020
18	Unit Testing Item(18)	30-06-2020	30-06-2020
19	Review and feedback regarding Item (18),changes	07-01-2020	07-03-2020
20	Documentation regarding Item(18)	07-03-2020	07-04-2020
21	Integrated testing	07-04-2020	07-06-2020
22	Any final changes to be implemented and review	07-06-2020	07-08-2020
23	Documentation	07-06-2020	07-12-2020
24	Final Review	13-07-2020	13-07-2020
25	Paper Submission	17-07-2020	17-07-2020

Thanks for your attention!

Are there any questions?