Adaptive Sparse Matrix-Matrix Multiplication on the GPU

Martin Winter
Graz University of Technology,
Austria
martin.winter@icg.tugraz.at

Daniel Mlakar
Graz University of Technology,
Austria
daniel.mlakar@icg.tugraz.at

Rhaleb Zayer
Max Planck Institute for Informatics,
Germany
rzayer@mpi-inf.mpg.de

Hans-Peter Seidel
Max Planck Institute for Informatics,
Germany
hpseidel@mpi-inf.mpg.de

Markus Steinberger
Graz University of Technology,
Austria
steinberger@icg.tugraz.at

A Artifact Description Appendix: Adaptive Sparse Matrix-Matrix Multiplication on the GPU

A.1 Abstract

The following appendix provides the necessary information to acquire the framework and rerun the experiments used to evaluate the framework.

A.2 Description

A.2.1 Check-list (artifact meta information)

- **Data set:** Matrix data set found on SuiteSparse Matrix Collection (Formerly the University of Florida Sparse Matrix Collection)
- **Hardware:** Recent GPU hardware from NVIDIA, tested on NVIDIA GTX 1080Ti, NVIDIA GTX TITAN X Pascal and NVIDIA GTX TITAN Xp
- **Output:** The output is provided in the output stream as well as in a .csv file
- **Experiment workflow:** Run *runall.bat/sh* file to gather results for all matrices in a folder or run individual matrices through the framework
- **Experiment customization:** The number of iterations used for the timing measurements can be altered. A CPU implementation can be enabled as well to confirm the results of the framework output
- **Publicly available?:** ACM DL

A.2.2 How software can be obtained (if available)

The framework is downloadable from ACM DL.

A.2.3 Hardware dependencies

Recent GPU hardware from NVIDIA, tested on NVIDIA GTX 1080Ti, NVIDIA GTX TITAN X Pascal and NVIDIA GTX TITAN Xp. Remaining system specifications should have a comparatively small impact on performance, performance was tested on an Intel Core i7-7700 paired with 32 GB RAM on Ubuntu 16.04 LTS.

A.2.4 Software dependencies

- CMake 3.2 or higher
- CUDA 9.1 / 9.2 / 10.0
- C++14 compliant compiler, tested on:
 - MSVC (Visual Studio 2017)
 - GCC 6 / GCC 7
- CUB v1.8.0

A.2.5 Datasets

The framework itself can parse *Matrix Market Format* (.mtx) files (Matrix data set found on SuiteSparse Matrix Collection (Formerly the University of Florida Sparse Matrix Collection)) and upon first parsing a matrix in this format also performs a conversion into a binary format that will be used for consecutive runs, which greatly reduces loading times. These are stored with the .hico extension.

A.3 Installation

On Linux simply run the provided *setup.sh* script which will clone CUB, setup and build the project. On Windows, download and extract CUB into the folder *include/external*, then create a build folder in the top directory and use CMake to setup the project to build.

A.4 Experiment workflow

The framework can be operated in one of two modes:

- Single Matrix

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
PPoPP'19, February 16–20, 2019, Washington, DC, USA

© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM ISBN 978-1-4503-6225-2/19/02...\$15.00

<https://doi.org/10.1145/3293883.3295701>

- In this setup the framework can be run for a single matrix and optionally confirm the resulting output matrix by comparing it to a host-based solution
- Complete testrun
 - A `runall.bat/.sh` file is provided that consecutively calls the framework with all matrices provided in a folder, this was used to run the large testcases

The output is provided in the output stream as well as in a separate `.csv` file which includes all important matrix stats (rows, columns, nnz, average nnz per row, etc.) as well as timing measurements. This script is setup such that each test run is done as a separate process, such that failed launches do not impede launches after that. The output of the script are timing measurements and when enabling *Debug* within the framework (the template instantiation must have this enabled) also detailed measurements as well as memory measurements.

For all matrices in the testset the framework computes $C = A \cdot A$ (or $C = A \cdot A^T$ if the input matrix is not square) and measures the time required for the multiplication procedure, only the input matrix A is provided directly on the device to the procedure and the result matrix C is also returned as a device matrix. Conversion operators are provided to transfer matrices between host and device as well as convert the *COO* format to *CSR* if required.

A.5 Evaluation and expected result

To replicate the results gathered in this paper it suffices to run the `runall.bat/.sh` file on a folder containing matrices found in the SuiteSparse Matrix Collection (Formerly the University of Florida Sparse Matrix Collection) dataset. The next page holds two plots detailing the exact performance measurements for the complete test set compiled using the hardware described above.

A.6 Experiment customization

The number of iterations used for the timing measurements can be altered. A CPU implementation can be enabled as well to confirm the results of the framework output. Debug can be enabled to get more detailed timing/memory results.