UPPSALA
UNIVERSITET

Project: Sparse Matrix Formats

Accelerator-based Programming

Oskar Tegby

October 2022

1 Introduction

This project studies the computational bandwidths of an iterative solver for a finite element problem using a data structure for sparse matrices called compressed row storage (CRS). To be specific, we study the computational throughput of a conjugate gradient method with a sparse matrix that comes from the finite element discretization of the Laplacian on a three-dimensional domain $(0, 1)^3$ meshed with bricks. Notably, this problem is studied both on the processor using C++ and on the graphics card using CUDA.

All tests were run on the Snowy node of the UPPMAX cluster using a NVIDIA Tesla T4 graphics card, and an Intel Xeon E5-2660 processor, and repeated 50 times to reduce the noise in the measurements.

2 Tasks

This section addresses the grading criteria to pass the assignment one-by-one in order to clearly show that the required functionality and knowledge has been obtained.

2.1 Criteria 1

2.1.1 Instruction

Provide a correct program for the basic tasks involving the CRS matrix both on the CPU and the GPU.

2.1.2 Solution

Please find the attached code. The correctness of these implementations is assured by observing that the error of the ℓ^2 norm decreases by a factor four when doubling the problem size, N, when executing the code assignment

`/app.v -repeat 100 -number Number`,

where `Number` refers to the datatype used, which is either `float` or `double`, and `v` refers to either the `host` or `cuda` code.

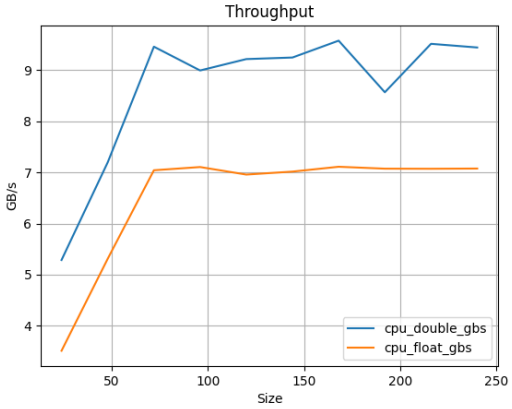
2.2 Criteria 2

2.2.1 Instruction

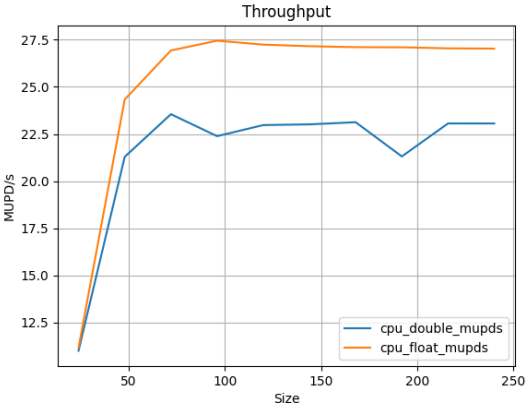
Provide computational measurements of acceptable performance on both platforms and display them in an understandable manner.

2.2.2 Solution

Figure 1 shows the performance of the CPU version of the code, and Figure 2 shows the performance on the GPU. Clearly, the GPU beats the CPU by over a factor ten.

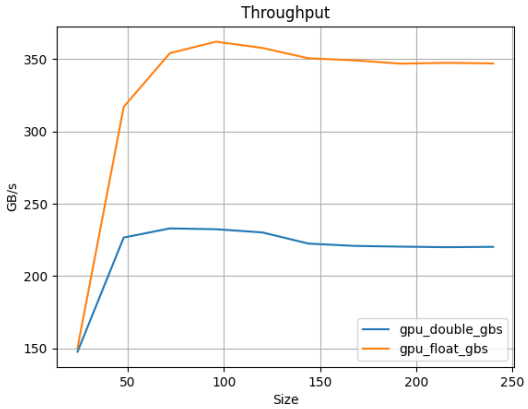


(a) The CPU throughput in GB/s.

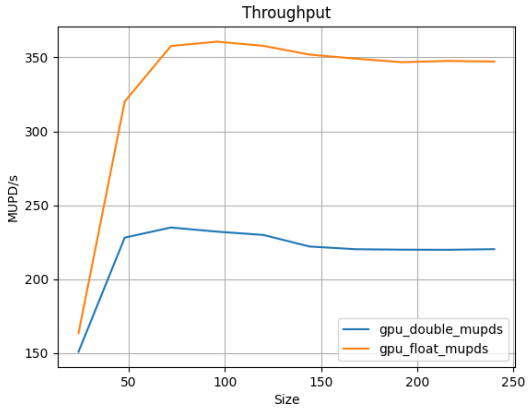


(b) The CPU throughput in MUPD/s.

Figure 1: The throughput achieved on the CPU.



(a) The GPU throughput in GB/s.



(b) The GPU throughput in MUPD/s.

Figure 2: The throughput achieved on the GPU.

2.3 Criteria 3

2.3.1 Instruction

Provide explanations for the expected performance of the sparse matrix-vector product and conjugate gradient solver with basic CRS matrix storage as well as the improved ELLPACK/SELL-C-Sigma approach by Kreutzer et al.

2.3.2 Solution

The performance with basic CRS storage is poor since the potential performance gained by vectorization and parallelization of the matrix-vector multiplication is ruined by the irregular data storage. That is solved by the ELLPACK storage, which fills all of the rows with zeros so that the rows are the same length.

However, that introduces the issue of storing a lot of unnecessary zeros, which is what CELL-C-σ solves by locally sorting σ rows of the data. That is, the data is sorted in groups of σ elements. The C in SELL-C-σ indicates how large the groups which are given extra zeros are. This decreases the number of zeros substantially, which improves performance without losing the ability to easily vectorize and parallelize the implementation on the GPU.

The expected performance increase should be about a factor two given the greatly decreased number of zeros stored in ELLPACK/SELL-C-σ.

2.4 Criteria 4

2.4.1 Instruction

Discuss the topics of data both in the report and during oral discussion of the project.

2.4.2 Solution

Please see Criteria 3.