Homework Assignment 2: Sparse Matrix Multiplication in Fortran

A. Ammar, Y. Damour, P. Reinhardt, A. Scemama

November 21, 2024

You can work in groups of up to three people. We expect that you will use Git to work collaboratively on your project. You may submit your project either as a `tar.gz` file or provide an address to a publicly available Git repository.

We expect the following files and structure in your submission:

- A `LICENSE` file specifying the license for your code.
- An `AUTHORS` file listing the names of all contributors.
- A `README.md` file providing a brief description of the directory structure and the project.
- An `INSTALL.md` file with clear instructions on how to compile and run the program.
- A `tests` directory containing tests to ensure that the program behaves as expected.
- (Optional) A `doc` directory for additional documentation, if the project requires more detail than the `README.md` file can provide.
- A `src` directory containing all the source files of your program.

In this assignment, you will implement and compare different methods of multiplying sparse matrices using Fortran. The provided files include sparse matrices with varying degrees of filling, stored in a compact format. Your task is to analyze and report on the computational efficiency of each multiplication method.

1 Objectives

Write a Fortran program that:

- Reads two sparse matrices from files line by line.
- Stores and manipulates the matrices in sparse and dense formats.
- Multiplies the matrices using different methods and measures computational performance.
- Analyzes the scaling behavior of multiplication with respect to matrix size and filling degree.

You will provide a report containing results and analysis of your experiments, including:

- Timing and profiling data for each multiplication method.
- Observations on scaling behavior with matrix size and filling degree.

2 Instructions

2.1 Sparse Matrix Representation

- Matrices are symmetric, and only the upper (or lower) triangle is stored in the file.
- In memory, the matrix should be stored using three 1D arrays in the compact format provided. You may store only the upper triangle or symmetrize the matrix in memory.

2.2 Matrix Multiplication with Sparse Matrices

- Implement a subroutine or function to calculate the product of two sparse matrices in their sparse format.
  - Count the actual number of multiplications performed relative to the theoretical maximum of N^3, where N is the matrix dimension.
  - Consider the symmetry of the matrices to minimize redundant calculations.

2.3 Matrix Multiplication with Dense Matrices

- Store the same matrices in dense format as 2D arrays:
  - Use arrays `A(N,N)` and `B(N,N)` for the matrices and `C(N,N)` for the result.
- Multiply the dense matrices using the following methods:
  1. The BLAS routine `DGEMM`.
  2. A hand-written Fortran routine that computes `C(i,j) = \sum_k A(i,k)*B(k,j)`.

2.4 Timing and Performance Analysis

- Measure the time taken for each multiplication method. If the time spent in a single multiplication is too small to measure, repeat the multiplication 100 or 1000 times in a loop to obtain a measurable value.
- Report the following:
  - Execution time for sparse matrix multiplication.
  - Execution time for dense matrix multiplication using `DGEMM`.
  - Execution time for dense matrix multiplication using your hand-written routine.

2.5 Profiling and Scaling Analysis

- Profile the performance of the three multiplication methods and compare their efficiency.
- Analyze the scaling behavior with respect to:
  - Matrix size.
  - Matrix filling rate (percentage of non-zero elements).
- For a 125×125 matrix, experiment with scaling the indices of the non-zero elements by a factor s, such that `A(i,j) -> A(s*i,s*j)`. Ensure that you consider how scaling affects the filling degree of the matrix.