ParCFD2024
35th International Conference on Parallel Computational Fluid Dynamics
Sep 02-04 2024, Bonn, Germany

ENABLING LIGHTER AND FASTER SIMULATIONS WITH REPEATED MATRIX BLOCKS

Josep PLANA-RIU*, F.Xavier TRIAS*, Guillem COLOMER*, Àdel ALSALTI-BALDELLOU†, Xavier ÁLVAREZ-FARRÉ‡ AND Assensi OLIVA*

* Heat and Mass Transfer Technological Centre
Technical University of Catalonia, ESEIAAT
Carrer de Colom 11, 08222 Terrassa (Barcelona), Spain
e-mail: {josep.plana.riu, francesc.xavier.trias, guillem.colomer, asensio.oliva}@upc.edu

† Department of Civil, Environmental and Architectural Engineering (ICEA)
University of Padova
Via Francesco Marzolo 9, 35131 Padova, Italy
e-mail: adel.alsaltibaldellou@unipd.it

‡ High-Performance Computing and Visualization Team
SURF
Science Park 140, 1098 XG Amsterdam, The Netherlands
e-mail: xavier.alvarezfarre@surf.nl

Key words: Sparse matrix-vector product, Sparse matrix-matrix product, Arithmetic intensity, High performance computing

Abstract. When pushing towards lighter and faster Computational Fluid Dynamics simulations, the contribution and construction of every component should be considered. The sparse matrix-vector product (`SpMV`) is the most expensive kernel among all operations. In some situations, e.g, with spatial reflection symmetries, the sparse matrices have some repeated blocks that could be exploited for better performance. By transferring the repeated blocks only once, the amount of data to transfer is reduced, and thus, the memory footprint of the simulation will be reduced. Moreover, with this framework, the `SpMV` is transformed into a sparse matrix-matrix product (`SpMM`), reducing the memory footprint and speeding-up the simulation. The method is tested in a differentially heated cavity in order to test the performance gains with the use of the `SpMM` compared to the use of a `SpMV`.

1 INTRODUCTION

Pushing toward bigger and bigger simulations of the incompressible Navier-Stokes equations requires an increase in the computational power available in high-performance computing (HPC) systems. Nonetheless, as these cases require a lot of memory and data transfer, it is not only the time spent computing that is relevant. Yet, the time spent in data transferring becomes the most relevant part of the time budget of the simulation, leading to what is known as a memory-bound process [1].

If the Navier-Stokes equations are solved numerically, it is straightforward to divide the operations into the well-known sparse matrix-vector product (`SpMV`), the dot product (`dot`), the linear combination of vectors (`axpy`), and the elementwise product of vectors (`axty`). These operations appear naturally in an algebraic approach, yet in stencil-based approaches these operations are implemented based on nested mesh-loops.

`SpMV`, being the most computationally expensive operation, requires transferring the data of the sparse matrix and the data of the whole full vector. This implies that the amount of data to be transferred for bigger simulations takes a relevant time compared to the time spent executing the operation. In this sense, some conditions might be exploited to transfer a smaller amount of data, either in the matrix, in the vector, or both. This implies that the kernel's arithmetic intensity (AI), defined as the ratio of the computing load and the data transferred, is low. Other techniques as the presented by Greathouse and Daga [2] for GPU compute units aim to improve the performance of `SpMV` by mapping properly the loads of the sparse matrix, leading to remarkable speed-ups compared to the original CSR-based algorithms.

For instance, Krasnopolsky [3] developed the concept of solving n flow states simultaneously to later on ensemble averaging the results of these flow states. In this paper, the `SpMV` operations were translated to sparse matrix-matrix products (`SpMM`) to all the n flow states simultaneously as a single operation. By doing so, the amount of data to be transferred compared to running n times the simulation was reduced, as the sparse matrix was only transferred once. By doing so, the AI would increase notably, as the amount of computations would be preserved while the amount of data transferred was reduced.

Later on, Alsalti-Baldellou et al. [4] exploited the domain's symmetries or repetitions to reduce the matrix's size to transfer by splitting the domain in the inner cells within the symmetric part and the bounds between every symmetric contribution. By doing so, the number of right-hand sides (RHS), i.e., the number of columns in the full matrix, would be 2^d, being d the number of symmetries in the domain.

While in the former, the methodology was only applied in the solution of the linear system of equations of the projection method, and no results in the speed-up of `SpMM` were presented, the latter applied this methodology to all the `SpMV` operations within the simulation, i.e., in the use of the gradient, divergence, and Laplacian operators.

Thus, the presented methodology can be applied in different situations apart from the ones previously mentioned: ensemble averaging of the time averaged results of turbulent flow simulations, application in domains with mirror symmetries or repeated geometrical structures (e.g. wall mounted cubes, wind farm), or parametric studies by changing relevant values in the simulations. Nonetheless, the framework of this study is based on ensemble averaging the solutions, following the works of Krasnopolsky [3].

2 APPLICATION TO CFD SIMULATIONS

Following the notation of Verstappen and Veldman [5], the semi-discrete incompressible Navier-Stokes equations read as follows for a staggered case,

$$M\mathbf{u}_s = 0, \qquad (1) \qquad\qquad \Omega\frac{d\mathbf{u}_s}{dt} + C(\mathbf{u}_s)\mathbf{u}_s = -\Omega G\mathbf{p}_c + D\mathbf{u}_s + \mathbf{f}_s, \qquad (2)$$

where M is the face-to-cell divergence operator, $\Omega = I_3 \otimes \Omega_s$, where Ω_s is a diagonal matrix containing the staggered volumes; C is the convective operator represented by a skew-symmetric matrix given a symmetry-preserving discretization; D is the diffusive operator and G is the cell-to-face gradient operator, and $\mathbf{f}_s$ represents the body forces.

More precisely, the tests were run in one high memory node of MareNostrum5 supercomputer (2x Intel Xeon Platinum 8480, 2x56 CPU cores) loading the node with around 400k cells per CPU core (46.65M cells), so that the largest cases would fit in the nodes.

The time-integration scheme used was a Heun third-order Runge-Kutta scheme (RK3) with a self-adaptive time step size computed based on the eigenbounds of the convective and diffusive operators. The results were tested to be independent of the scheme used.

The discretization scheme for the operators considered only the first neighbor cell, leading to 7 non-zeros per row in a three-dimensional case. Nonetheless, the Laplacian operator for the Poisson equation was tested considering the first neighbor (`7p`), a cross pattern (`13p`, 13 non-zero entries per row), and a cube pattern (`27p`, 27 non-zero entries per row). A two-dimensional representation of these stencils is shown in Fig. 1.

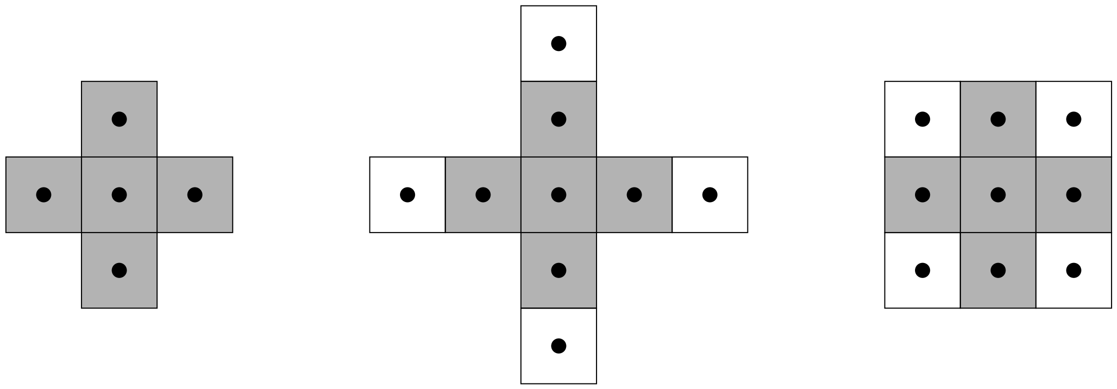

Figure 1: Two-dimensional representations of the stencil for `7p` (left), `13p` (center) and `27p` (right). The shaded cells represent the first neighbors in all representations.

The tests were run by setting a fixed number of iterations in the Poisson solver solution. The speed-up was computed as follows: considering the $n = 1$ simulation as the baseline, with a `SpMV` time $T_{\texttt{SpMV}}$, the speed-up for a given n with $T_{\texttt{SpMM(n)}}$ is determined by

$$P_{n,\texttt{SpMM}}(n) = \frac{nT_{\texttt{SpMV}}}{T_{\texttt{SpMM(n)}}}. \qquad (3)$$

In this case, the tests will be applied to a differentially heated cavity (DHC) setup with Rayleigh number $\mathrm{Ra} = 10^{10}$ and Prandtl number $\mathrm{Pr} = 0.71$ with aspect ratio 4 in a setup similar to Krasnopolsky [3] so that multiple flow states, which correspond to 1, 2, 4, and 8 RHS, will be launched simultaneously for a few iterations to compute the speed-ups of the SpMM operations. A higher number of non-zeros in the sparse matrix positively affects the speed-ups obtained, as for 8 RHS, the speed-ups go from maximum values of ≈2.5 to ≈3.5 for 7p and 27p, respectively.

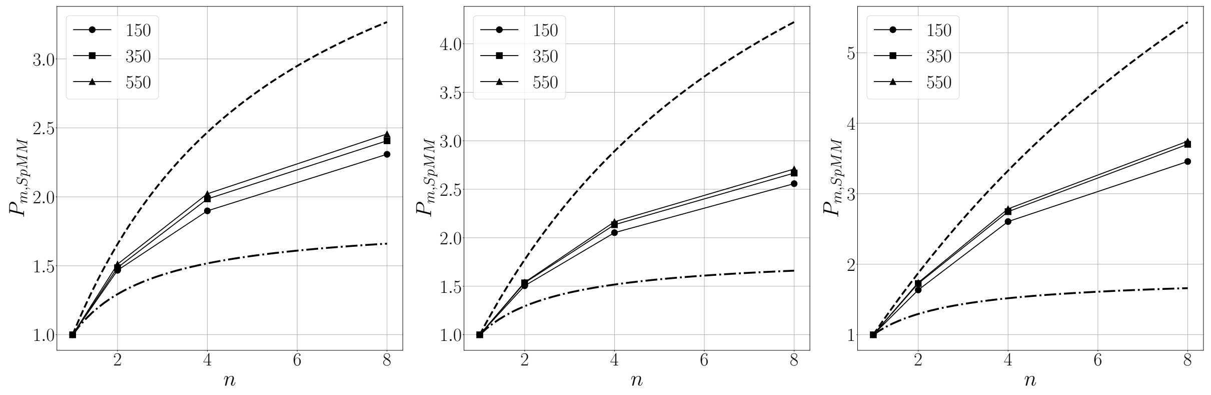

Figure 2: Speed-up for the 7p (left), 13p (center), and 27p (right) discretizations for a given number of Poisson solver iterations in the DHC case. The dashed line provides the upper bound of the speed-up, whereas the dot-dash line defines the lower bound, according to Alsalti-Baldellou et al. [4].

Fig. 2 presents the results for 150, 350, and 550 iterations per solution of the Poisson equation for all 7p, 13p, and 27p. The results in all three cases lay between the theoretical upper and lower bounds, according to Alsalti-Baldellou et al. [4]. It would be expected to obtain a slightly better performance for a greater number of iterations as the weight of the Poisson equation would rise in the overall wall clock time. However, the relevance of a greater number of iterations in the speed-up should decrease the bigger the value.

The method has been tested as well for other cases such as a turbulent planar channel flow of $\mathrm{Re}_\tau = 180$ with a similar mesh and load, leading to equivalent results to the presented in Fig. 2.

3 CONCLUSIONS

In this work, the speed-up analysis obtained in the use of SpMM in simulations in which there are repeated matrix blocks compared to the use of a single RHS (i.e., a SpMV) is presented and compared against the theoretical upper and lower bounds for three different configurations, presented in Fig. 1: 7p, 13p and 27p.

It can be seen in Figs. 2,?? that the numerical speed-ups are obtained between the theoretical upper and lower bounds, being approximately equidistant to both bounds in all the cases run for both DHC and CF. Moreover, it can be observed that the denser the sparse matrix, the higher the speed-up, leading to an increased interest in applying the method to simulations in which the discretization is of a higher order, i.e., with more non-zeros per row, compared to using only the first neighbor, with 7 non-zeros per row.

ACKNOWLEDGEMENTS

The work presented in this abstract is supported by the *Ministerio de Economía y Competitividad*, Spain, SIMEX project (PID2022-142174OB-I00). J.P-R. is also supported by the Catalan Agency for Management of University and Research Grants (AGAUR). Calculations were performed on the MareNostrum 5 GCC supercomputer at the BSC. The authors thankfully acknowledge these institutions.

REFERENCES

[1] S. Williams, A. Waterman and D. Patterson, "Roofline: an insightful visual performance for multicore architectures," *Communications of ACM*, vol. 52, pp. 65-76, 2009.

[2] J. L. Greathouse and M. Daga, "Efficient Sparse Matrix-Vector Multiplication on GPUs using the CSR Storage Format," *SC '14: Proceedings of the Internatioanl Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 769-780, 2014.

[3] B. I. Krasnopolsky, "An approach for accelerating incompressible turbulent flow simulations based on simultaneous modelling of multiple ensembles," *Computer Physics Communications*, vol. 229, pp. 8-19, 2018.

[4] À. Alsalti-Baldellou, X. Álvarez-Farré, G. Colomer, A. Gorobets, C. D. Pérez-Segarra, A. Oliva and F. X. Trias, "Lighter and faster simulations on domains with symmetries," *Computers & Fluids*, vol. 275, 106247, 2024.

[5] R. W. C. P. Verstappen and A. E. P. Veldman, "Symmetry-preserving discretization of turbulent flow," *Journal of Computational Physics*, vol. 187, pp. 343-368, 2003.