

We thank the Referee for his time and his constructive comments. We have complied with most of the proposed changes. In the following, the comments made by the Referee appear in black, while our replies are in blue.

This work represents a major GPU porting effort of the Meso-NH model. This model is originally written in Fortran with MPI for distributed-memory parallelization. The work ports significant parts of the model for NVIDIA and AMD architectures using OpenACC. In addition to the directives needed to port GPU kernels, a pre-processor was developed, along with a multi-grid pressure solver as an alternative to the FFT-based one. An extensive performance analysis on different systems is provided.

I found the work insightful and the paper well-organized and written. However, some parts lack the detail needed to fully understand the numerical and computational approach. Without clarifying these details, it becomes quite hard to understand some of the choices that were taken in the effort. I will elaborate below point-by-point.

1. L 78-79: "The current pressure solver consists on [of] a conjugate-residual algorithm accelerated by a flat fast Fourier transform (FFT) precondition." This is insufficient to fully understand the numerical approach to solving the pressure equation. Could you provide more (mathematical) background and mention in which directions FFT(s) are being used and the consequences for the grid spacing and boundary conditions along this and the other directions? Moreover, could you illustrate how this equation is solved in parallel (I could not find a clear answer in the cited references either)?

We now provide the reader with specific references while refraining from adding lengthy mathematical explanations. In section 2.1, we specifically implemented three changes:

(1) We now refer to Skamarock et al. (1997), Bernardet (1995) and to the 20 pages of Chapter 9, Part I of the Meso-NH scientific documentation devoted to the pressure problem. It reads "The current pressure solver consists of a conjugate-residual algorithm (Skamarock et al. 1997) accelerated by a flat Fast Fourier transform (FFT) preconditioner following Bernardet (1995). The horizontal part of the operator to invert in the elliptic pressure problem is processed with FFT while its vertical part leads to the classical tridiagonal matrix. For a detailed description, the reader is referred to Chapter 9, Part I of the scientific documentation available on the Meso-NH web site (<http://mesonh.aero.obs-mip.fr>, last access: 16 December 2024)".

(2) We now mention the initial implementation of the FFT solver for parallel computers done by Giraud et al. (1999). We added "The initial parallel implementation of the FFT pressure solver takes into account two other types of partitioning on each horizontal direction, called x -slice and y -slice. Communication routines have been implemented to move a field between these different decompositions. It is then possible to perform the FFT for each horizontal direction (Giraud et al., 1999)".

(3) We now mention the adaptation of the FFT pressure solver for massively parallel computers. We added "In the case of FFT, moving data from a vertical beam decomposition to x - and y -slices limits the number of processes to the smallest horizontal dimension. For example, a model on a $512 \times 512 \times 128$ grid can only be run with 512 processes. Instead, a 3-dimensional decomposition of the beam was implemented and optimized. For a run using $p_x \times p_y$ processes, the global domain of size $N_x \times N_y \times N_z$ is divided into z -pencils of size $(N_x/p_x) \times (N_y/p_y) \times N_z$. The FFT is first performed on each x -pencil of size $N_x \times (N_y/p_y) \times (N_z/p_x)$ in the x direction, then on each y -pencil of size $(N_x/p_x) \times N_y \times (N_z/p_x)$ in the y direction. Next, the tridiagonal system is solved in the Fourier space for each z -pencil. Finally, inverse FFTs are calculated on each y -pencil, then on each x -pencil. As a result, the above example can now be run with up to $512 \times 128 = 65536$ processes."

2. L. 80. Can you not simply state that it is written in Modern Fortran? If you want to be pedantic, you'd need to state that it has features from older standards (77, 90), too.

Our intention is not to be pedantic. We simply want to point out that Meso-NH uses more recent features than Fortran 95, some of which are useful for porting to GPUs (i.e., `do concurrent`)

3. L. 122. Just to comment that I found that using 'default(present)' in all OpenACC kernel loops really helps with debugging, as one would get a runtime error whenever something is accessed in a kernel that is not on the device.

The 'default(present)' directive is not really applicable or useful here. As we are porting the code piece by piece to the GPU, not all the data resides on the GPU memory. The code will therefore crash if the data

has been calculated on the CPU and is not yet on the GPU memory. And even if the data is on the GPU memory, an update of the CPU or GPU memory copy is required (`!$acc update host/device ...`). Bit reproducibility ensures that no such errors occur.

4. I found Figure 1 quite hard to understand. Could you improve the captions so that it is clear what we are looking at? Is the left a serial computation, and the right one an MPI decomposed one with 2D pencils?

Following your suggestion, the caption of Fig. 1 is now "Schematic of bit-reproducible verification between primary and replica simulations using the `MPPDB_CHECK` library. On the left, the primary simulation is a computation performed on the entire domain, i.e. without any domain decomposition on CPU. On the right, the replica simulation is a parallel computation performed on CPU or GPU on the domain broken down into 4×4 pencils."

5. L213. Same spirit as comment 1. "the FFT algorithm requires all-to-all communications between MPI processes (...)" Is the FFT algorithm requiring all-to-all communications, or is it the Poisson solver? It is unclear how the pressure equation is being solved numerically (1D or 2D FFTs? + CR along which direction?), and how that is implemented in a distributed-memory paradigm.

To avoid confusion, we changed "The FFT algorithm [...]" into "The FFT pre-conditioner [...]". See our response to comment 1 regarding the details on the FFT pressure solver added in Sect. 2.1.

6. L 218. "The most promising alternative for solving this type of elliptic equation is the use of a geometric multigrid solver for regular structured grid". This claim needs to be substantiated or reconsidered, as it is not obvious, especially for GPUs: As you coarsen in an MG method, the GPU occupancy is being massively reduced, making it perform extremely poorly on GPU-based systems. So, I would say that geometric multigrid solvers do not pair that well with GPUs.

Our claim is now substantiated by adding "In particular, Müller et al. (2015) ported a C/CUDA version of a geometric multigrid algorithm that scaled up to 16384 GPUs."

We also added a paragraph regarding the cost of FFT-based solvers on supercomputers: "Such a negative impact of all-to-all communications in the FFT pre-conditioner has been seen with Meso-NH running on MIRA, a Blue Gene/Q system at Argonne National Laboratory by showing sub-optimal scalability when using 2 billion threads (Lac et al. 2018; see their Fig. 1). Verma et al. (2023) performed a scaling analysis of their GPU-FFT library for grid sizes of 1024^3 , 2048^3 , and 4096^3 , utilizing up to 512 A100 GPUs. They reported a ratio of communication time to total time of 50% when using 8 GPUs and over 90% when using more than 128 GPUs. Ibeid et al. (2020) showed in exascale projections for grid size of 65536^3 that the FFT total time is due solely to the FFT communication time, which is dominated by the network access cost."

7. L 235. I see that along one direction the (direct) Thomas algorithm is used, while in other two an iterative (MG) method is used. The linear algebra behind this approach is quite unclear to me, so please provide more mathematical details so a reader can easily follow the method without navigating into the code or other references.

For reasons of readability of the manuscript as a whole (a criticism made by the Reviewer #1), we prefer to refrain from adding lengthy mathematical explanations and we refer the reader to the 20-page documentation of the MG method (Müller 2014).

8. L. 250. A comparison between FFT-based and multigrid is performed, but I am missing a lot of details needed for reproducibility and better understanding. What kind of tolerance is being used in the FFT-based flavor (CR method), and in the MG one? What kind of smoother is being used in the geometric multi-grid method? These details need to be clear for better interpreting the results.

To clarify, we added "It should be noted that the comparison between FFT and MG pressure solvers is presented only in terms of computational performance. No reproducibility of pressure between solvers is expected. Similar accuracy, i.e. the same threshold in the residual divergence of the pressure value, is however demanded by both solvers."

9. L 288. "The test case uses advection, turbulence, cloud microphysics, pressure solver and other components". Consider being more exhaustive here.

To clarify, the sentence is now "The test case uses advection, turbulence, cloud microphysics, pressure solver

(see section 2.1 for more details) and other components. These other components include elements not covered by the above-mentioned processes, such as gravity and Coriolis terms (executed on GPUs), radiation (called every 900 s only, executed on CPUs), time advancement of all variables and I/O operations (which are largely disabled in our simulations).”

10. L 322. I read that there can be several MPI tasks per GPU. It is unclear how this is implemented in practice. A sketch with the domain decomposition colored by MPI tasks, along with the GPUs that handle each group of tasks, would be very insightful.

The binding configuration was explained Line 298 (“Binding on CPU cores and GPUs is carefully chosen [...]. If several binding configurations have been tested, only the one giving the fastest results is kept.”). Then we added “For example, the best run on an Adastral node with 16 MPI processes uses a 4×4 subdomain grid. To optimize MPI communications, processes with neighboring subdomains are mapped to nearby GPUs, prioritizing proximity and direct network links. To optimize memory bandwidth, each process is pinned to a separate CPU core, evenly distributed across the 8 L3 caches (2 processes per cache) and 4 NUMA nodes. Finally, MPI processes are paired with their closest GPU for optimal host-device memory transfers.”

11. Please re-consider the performance analysis in light of the fact that with MG the GPU occupancy decreases at coarse levels, and if this can explain some of the observations.

It is possible that the low occupancy rate at coarse levels explains the behavior of the MG solver. However, we can note that the occupancy per GPU is fixed when the number of GPUs remains unchanged, but that the solver performance decreases if the number of CPUs is increased. Other phenomena (MPI performance, compiler optimizations, software stack...) could probably explain what is going on and are not easy to differentiate. As a result, we believe it is difficult to determine with any certainty the reasons for what is observed.

12. Finally, in other fluid dynamics domains, direct FFT-based solvers (i.e., FFT factorization along two directions, and Gauss elimination along the last one) show 3x to 100x speed-up compared to multi-grid approaches. While their communication patterns are more complex, their fast performance and good GPU utilization make them quite attractive for GPU-based systems. This goes a bit in contrast with the present observations, though the baseline FFT-based solver is not direct here. I would recommend putting this work in perspective w.r.t. other efforts in the literature that have made similar comparisons.

See our response to comment 6 where several papers are cited on the scalability of FFT-based and MG solvers on exascale systems.

Feel free to contact me directly if something is unclear at P.SimoesCosta@tudelft.nl.

Your comments are very clear. Thanks again for your time.

References

Bernardet, P.: The pressure term in the anelastic model: a symmetric solver for an Arakawa C grid in generalized coordinates, *Mon. Weather Rev.*, 123, 2474–2490, [https://doi.org/10.1175/1520-0493\(1995\)123<2474:TPTITA>2.0.CO;2](https://doi.org/10.1175/1520-0493(1995)123<2474:TPTITA>2.0.CO;2), 1995

Giraud, L., Guivarch, R., and Stein, J.: A Parallel distributed Fast 3D Poisson Solver for MesoNH, in: *Euro-Par’99 Parallel Processing. Lecture Notes in Computer Science*, vol. 1685, pp. 1431–1434, Springer Berlin Heidelberg, Berlin, Heidelberg, https://doi.org/10.1007/3-540-48311-X_201, 1999

Ibeid, H., Olson, L., and Gropp, W.: FFT, FMM, and multigrid on the road to exascale: Performance challenges and opportunities, *J. Parallel. Distrib. Comput.*, 136, 63–74, <https://doi.org/10.1016/j.jpdc.2019.09.014>, 2020

Lac, C., Chaboureau, J.-P., Masson, V., Pinty, J.-P., Tulet, P., Escobar, J., Leriche, M., Barthe, C., Aouizerats, B., Augros, C., Aumond, P., Auguste, F., Bechtold, P., Berthet, S., Bieilli, S., Bosseur, F., Caumont, O., Cohard, J.-M., Colin, J., Couvreur, F., Cuxart, J., Delautier, G., Dauhut, T., Ducrocq, V., Filippi, J.-B., Gazen, D., Geoffroy, O., Gheusi, F., Honnert, R., Lafore, J.-P., Lebeaupin Brossier, C., Libois, Q., Lunet, T., Mari, C., Maric, T., Mascart, P., Mogé, M., Molinié, G., Nuissier, O., Pantillon, F., Peyrillé,

P., Pergaud, J., Perraud, E., Pianezze, J., Redelsperger, J.-L., Ricard, D., Richard, E., Riette, S., Rodier, Q., Schoetter, R., Seyfried, L., Stein, J., Suhre, K., Taufour, M., Thouron, O., Turner, S., Verrelle, A., Vié, B., Visentin, F., Vionnet, V., and Wautelet, P.: Overview of the Meso-NH model version 5.4 and its applications, *Geosci. Model Dev.*, 11, 1929–1969, <https://doi.org/10.5194/gmd-11-1929-2018>, 2018.

Müller, E. H.: *TensorProductMultigrid*, p. (last access: 16 December 2024), <https://bitbucket.org/em459/tensorproductmultigrid/src/master/>, 2014

Müller, E. H., Scheichl, R. and Vainikko, E.: Petascale solvers for anisotropic PDEs in atmospheric modelling on GPU clusters, *Parallel Computing*, 50, 53–69, <https://doi.org/10.1016/j.parco.2015.10.007>, 2015.

Skamarock, W. C., Smolarkiewicz, P. K., and Klemp, J. B.: Preconditioned conjugate-residual solvers for Helmholtz equations in nonhydrostatic models, *Mon. Weather Rev.*, 125, 587–599, [https://doi.org/10.1175/1520-0493\(1997\)125<0587:PCRSFH>2.0.CO;2](https://doi.org/10.1175/1520-0493(1997)125<0587:PCRSFH>2.0.CO;2), 1997

Verma, M., Chatterjee, S., Garg, G., Sharma, B., Arya, N., Kumar, S., Saxena, A., K., M., and Verma, M. K.: Scalable multi-node fast Fourier transform on GPUs, *SN comput. sci.*, 4, 625, <https://doi.org/10.1007/s42979-023-02109-0>, 2023.