The Journal of Open Source Software

The 2DECOMP&FFT library: an update with new CPU/GPU capabilities

Stefano Rolfo [1¶], **Cédric Flageul**[2*], **Paul Bartholomew**[3*], **Filippo Spiga**[4*], **and Sylvain Laizet**[5*]

1 STFC Daresbury Laboratory, Scientific Computing Department, UKRI, UK **2** PPRIME institute, Curiosity Group, Université de Poitiers, CNRS, ISAE-ENSMA, Poitiers, France **3** EPCC, The University of Edinburgh, Edinburgh, UK **4** NVIDIA Corporation, Cambridge, UK **5** Department of Aeronautics, Imperial College London, London, UK ¶ Corresponding author * These authors contributed equally.

DOI: 10.21105/joss.05813

Software

- Review
- Repository
- Archive

Editor: Patrick Diehl

Reviewers:

- @CeciliaCoelho
- @p-costa

Submitted: 30 August 2023
Published: 21 November 2023

Summary

The 2DECOMP&FFT library is a software framework written in modern Fortran to build large-scale parallel applications. It is designed for applications using three-dimensional structured meshes with a particular focus on spatially implicit numerical algorithms. However, the library can be easily used with other discretisation schemes based on a structured layout and where pencil decomposition can apply. It is based on a general-purpose 2D pencil decomposition for data distribution and data Input Output (I/O). A 1D slab decomposition is also available as a special case of the 2D pencil decomposition. The library includes a highly scalable and efficient interface to perform three-dimensional Fast Fourier Transforms (FFTs). The library has been designed to be user-friendly, with a clean application programming interface hiding most communication details from application developers, and portable with support for modern CPUs and NVIDIA GPUs (support for AMD and Intel GPUs to follow).

Statement of need

The 2DECOMP&FFT library (Li & Laizet, 2010) was originally designed for CPU hardware and is now used by many research groups worldwide. The library is based on a 2D-pencil decomposition for data distribution on distributed memory systems and is used as the core of many CFD solvers such as Xcompact3d (Bartholomew et al., 2020) and CaNS (Costa, 2018), with excellent strong scaling performance up to hundreds of thousands of CPU cores. 2DECOMP&FFT mainly relies on MPI, and it offers a user-friendly interface that hides the complexity of the communication. Version 2.0.1 of the library also offers a 1D slab decomposition, which is implemented as a special case of the 2D decomposition. Two alternatives are possible:

- Initial slabs orientation in the XY plane;
- Initial slabs orientation in the XZ plane.

In many configurations the slabs decomposition gives some gain in performance with respect to the 2D-pencil decomposition. This is a consequence of having data already in memory when transposing between the two dimensions of the slab. Therefore, it is possible to perform a simple memory copy between input and output arrays instead of the full MPI communication.

The library also offers a very efficient and flexible interface to perform 3D Fast Fourier Transform (FFT) on distributed memory systems. However, 2DECOMP&FFT is mainly designed to perform data management and communication and the actual computation of the 1D FFT is delegated to 3rd-party libraries. The supported FFT backends are: FFTW (Frigo & Johnson,

2005), the Intel Math Kernel Library (MKL), and the CUDA FFT (cuFFT), which is used for FFT on NVIDIA GPUs. A Generic FFT backend, based on Glassman's general N Fast Fourier Transform (Ferguson, 1982), is also available to make the library more portable.

While the 2DECOMP&FFT library has been designed with high order compact schemes in mind, it is possible that some derivatives can be evaluated using an explicit formulation based on local stencils. For this reason a halo support API is also provided to support explicit message passing between neighbouring pencils.

Finally, the library provides infrastructure to perform parallel data I/O using MPI I/O or ADIOS2 (Godoy et al., 2020). The API provide several features such as: writing single or multiple 3D arrays into a file, writing 2D slices of the data, and data compression either via ADIOS2 or by writing reduced precision or resolution with the MPI I/O backend.

The first version of the library was released in 2010 as a tar.gz package, with a Makefile approach, and could only make use of CPUs. It has not been modified since its release. The new version of the library can now leverage NVIDIA GPUs, modern CPUs, and various compilers (GNU, Intel, NVHPC, CRAY). It has CMAKE capabilities as well as a proper continuous integration framework with automated tests. The new library was designed to be more appealing to the scientific community, and it can now be easily implemented as an independent library for use by other software.

GPU porting

An initial port of 2DECOMP&FFT to GPUs was performed within the solver AFiD-GPU (Zhu et al., 2018), which was mainly based on CUDA-Fortran for some kernels and CUDA-aware-MPI for communications. A second library, named cuDECOMP, which was directly inspired by 2DECOMP&FFT, takes full advantages of CUDA and uses NVIDIA's most recent libraries for communications, such as NVIDIA Collective Communication Library (NCCL), is presented in Romero et al. (2022). Indeed, cuDECOMP only targets NVIDIA GPUs. The updated 2DECOMP&FFT mainly uses a mix of CUDA-Fortran and openACC for the GPU porting together with CUDA-aware-MPI and NCCL for the communications. In addition to previous work, the FFT module is ported to GPUs using cuFFT. The next step is also to implement OpenMP for GPU porting to support both AMD and Intel GPU hardware.

How to use 2DECOMP&FFT

The 2D Pencil Decomposition API is defined with three Fortran modules which should be used by applications as:

```
use decomp_2d_constants
use decomp_2d_mpi
use decomp_2d
```

where `use decomp_2d_constants` defines all the parameters, `use decomp_2d_mpi` introduces all the MPI related interfaces, and `use decomp_2d` contains the main decomposition and transposition APIs. The library is initialised using:

```
call decomp_2d_init(nx, ny, nz, p_row, p_col)
```

where `nx`, `ny`, and `nz` are the spatial dimensions of the problem, to be distributed over a 2D processor grid $p_{row} \times p_{col}$. Note that none of the dimensions need to be divisible by `p_row` or `p_col`. In the case of `p_row=p_col=0`, an automatic decomposition is selected among all possible combinations available. A key element of this library is a set of communication routines that perform the data transpositions. As mentioned, one needs to perform 4 global transpositions to go through all 3 pencil orientations (i.e., one has to go from x-pencils

to y-pencils to z-pencils to y-pencils to x-pencils). Correspondingly, the library provides 4 communication subroutines:

```
call transpose_x_to_y(var_in,var_out)
call transpose_y_to_z(var_in,var_out)
call transpose_z_to_y(var_in,var_out)
call transpose_y_to_x(var_in,var_out)
```

The input array `var_in` and output array `var_out` are defined by the code using the library and contain distributed data for the correct pencil orientations.

Note that the library is written using Fortran's generic interfaces so different data types are supported without user input. That means in and out above can be either real or complex arrays, the latter being useful for applications involving 3D Fast Fourier Transforms. Finally, before exit, applications should clean up the memory by:

```
call decomp_2d_finalize
```

Detailed information about the decomposition API are available here. Several examples detailing the usage of the different library functionalities can be found here.

Acknowledgements

The first version of the library was initially designed thanks to several projects funded under the HECToR Distributed Computational Science and Engineering (CSE) Service operated by NAG Ltd. The new library has been designed thanks to the support of EPSRC via the CCP Turbulence (EP/T026170/1) and work funded under the embedded CSE programme of the ARCHER2 UK National Supercomputing Service (http://www.archer2.ac.uk) (ARCHER2 eCSE03-2).

References

Bartholomew, P., Deskos, G., Frantz, R. A. S., Schuch, F. N., Lamballais, E., & Laizet, S. (2020). Xcompact3D: An open-source framework for solving turbulence problems on a Cartesian mesh. *SoftwareX*, *12*, 100550. https://doi.org/10.1016/j.softx.2020.100550

Costa, P. (2018). A FFT-based finite-difference solver for massively-parallel direct numerical simulations of turbulent flows. *Computers & Mathematics with Applications*, *76*(8), 1853–1862. https://doi.org/10.1016/j.camwa.2018.07.034

Ferguson, W. E. (1982). A simple derivation of Glassman's general N fast Fourier transform. *Computers & Mathematics with Applications*, *8*(6), 401–411. https://doi.org/10.1016/0898-1221(82)90016-5

Frigo, M., & Johnson, S. G. (2005). The design and implementation of FFTW3. *Proceedings of the IEEE*, *93*(2), 216–231. https://doi.org/10.1109/JPROC.2004.840301

Godoy, W. F., Podhorszki, N., Wang, R., Atkins, C., Eisenhauer, G., Gu, J., Davis, P., Choi, J., Germaschewski, K., Huck, K., Huebl, A., Kim, M., Kress, J., Kurc, T., Liu, Q., Logan, J., Mehta, K., Ostrouchov, G., Parashar, M., … Klasky, S. (2020). ADIOS 2: The adaptable input output system. A framework for high-performance data management. *SoftwareX*, *12*, 100561. https://doi.org/10.1016/j.softx.2020.100561

Li, N., & Laizet, S. (2010). 2DECOMP&FFT - a highly scalable 2D decomposition library and FFT interface. *Cray User Group 2010 Conference*, 1–13.

Romero, J., Costa, P., & Fatica, M. (2022). Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library. *Proceedings*

of the Platform for Advanced Scientific Computing Conference. https://doi.org/10.1145/3539781.3539797

Zhu, X., Phillips, E., Spandan, V., Donners, J., Ruetsch, G., Romero, J., Ostilla-Mónico, R., Yang, Y., Lohse, D., Verzicco, R., Fatica, M., & Stevens, R. J. A. M. (2018). AFiD-GPU: A versatile Navier–Stokes solver for wall-bounded turbulent flows on GPU clusters. *Computer Physics Communications*, *229*, 199–210. https://doi.org/10.1016/j.cpc.2018.03.026