Towards Enabling Automatically Differentiable High Performance Computing For Cosmological N-body Simulations

Wassim Kabalan[1], François Lanusse[2], Alexandre Boucaud[1], Eric Aubourg[3], Josquin Errard[1]

[1]Université Paris Cité, CNRS, Astroparticule et Cosmologie, F-75013 Paris, France
[2]Université Paris-Saclay, Université Paris Cité, CEA, CNRS, AIM, 91191, Gif-sur-Yvette, France
[3]Université Paris Cité, CNRS, CEA, Astroparticule et Cosmologie, F-75013 Paris, France

Abstract

A series of recent works highlighted the potential of so-called full-field cosmological inference for the analysis of upcoming weak lensing and galaxy clustering surveys, with the promise of being able to access the non-Gaussian information contained in the data. Even though, implicit inference does not necessarily need a differentiable forward model to perform full-field inference, explicit inference instead could be extremely challenging to do without the gradient information of the forward model. However, building a differentiable forward model of the large-scale structure currently constitutes one of the main bottlenecks, as none of the publicly available differentiable N-body codes support large-scale distribution necessary for large cosmological volumes.

To address this, we introduce `jaxDecomp`, a JAX (Bradbury et al., 2018) library that efficiently decomposes the simulation data into 2D slices (pencils) to facilitate multi-node parallel Fast Fourier Transforms (FFTs) and halo exchanges, leveraging the power of compiled code directly within `JAX` code. This library will enable the large-scale distribution of simulations on High Performance Computing clusters (HPC), resources and will seamlessly integrate with existing open source simulation codes like `JaxPM` or `pmwd`. We will present in particular our early scaling tests of this approach on the Jean-Zay GPU supercomputer.

N-Body simulations in cosmology

- Full-field inference is the optimal inference method as it accounts for both Gaussian and non-Gaussian information in the data. Such inference scheme relies strictly on simulations, also referred to as forward model. The quality of the inference is tied to the realism the forward model, motivating the development of **accurate and scalable simulations**.
- N-body simulations mimic the evolution of the dark matter distribution in a volume of universe from initial conditions. They are a crucial step of the forward modeling of the large-scale structure (LSS) as seen on Figure 1.

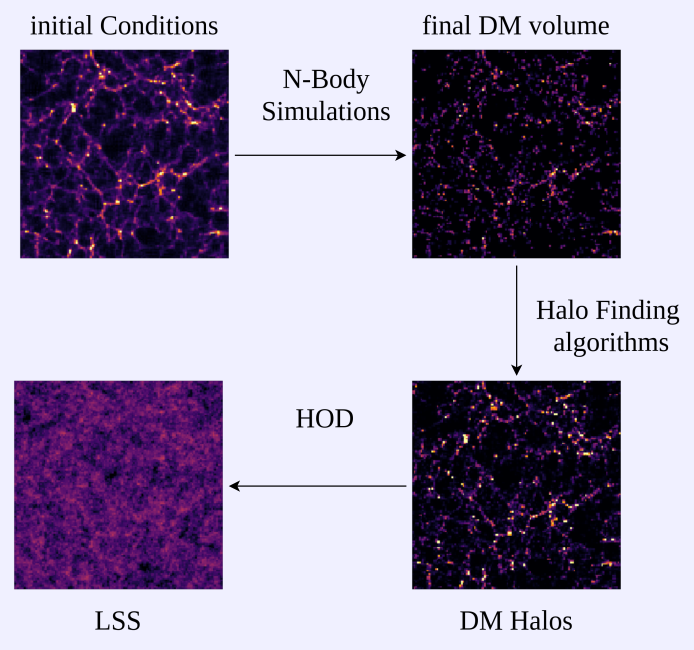


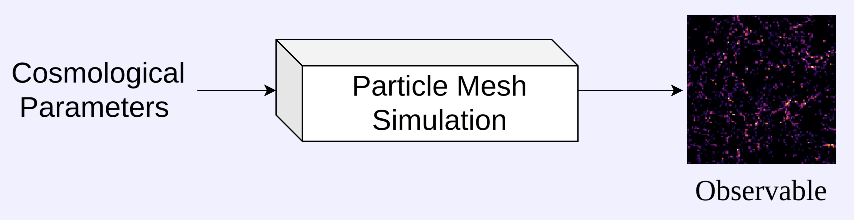
Figure 1: Simulation based forward modeling for the large scale structure.

- Among the various types of N-body simulations, some are accurate and computationally intensive like the hydro-dynamical ones, and some like particle-mesh solutions trade-off **computing efficiency** at the cost of accuracy in short-range interactions.
- A particle-mesh simulation starts from the Poisson equation $\nabla^2 \Phi(\mathbf{r}) = 4\pi G \rho(\mathbf{r})$, and at each step, the gravitational force is computed from the over-density field δ **in Fourier space**

$$\mathbf{f}(\mathbf{k}) = -\nabla \nabla^{-2} \delta(\mathbf{k})$$

- Forces are interpolated at particle positions, and then positions are velocities are updated accordingly.

Differentiable Particle-Mesh solvers with JAX



- Differentiability in a forward model **unlocks** the use of **advanced inference strategies**, including Maximum A Posteriori estimation, Hamiltonian Monte Carlo, or Variational Inference, which rely on access to gradients of the likelihood function.
- Fast, differentiable simulations allow us to analyze how changes in initial conditions influence simulation outcomes, providing insights into the behavior of complex systems.
- The particle-mesh solver FastPM (Feng et al., 2016) uses an iterative process that **can be differentiated**, unlike the more precise hydro-dynamical simulations. The result is `JaxPM`[a], a `JAX`-accelerated and gradient-enabled particle-mesh N-body simulator.

Caveat: these differentiable N-Body simulations suites are currently limited by the memory constraints of the hardware they are running on and the computational demand of the 3-dimensional Fourier transforms they are relying on.

[a]`https://github.com/DifferentiableUniverseInitiative/JaxPM`

Distributed 3D Fourier transforms

In this work, we propose a 2D decomposition based on 2DECOMP&FFT (Rolfo et al., 2023) for the 3D FFT using `cuDecomp` (Romero, Costa, and Fatica, 2022), allowing for **efficient distribution across multiple GPUs and nodes** on super computers such as Jean-Zay or Perlmutter.

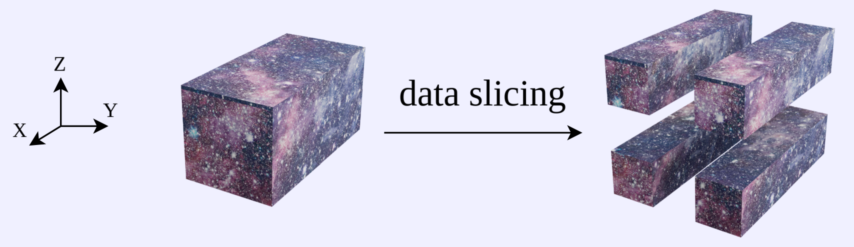


Figure 2: Illustration of cosmological data volume decomposition for distributing FFT operations.

Distributed 3D Fourier transforms (cont.)

The algorithm that we call later `PFFT3D` is the following

1. simulation data (positions, velocities) are split by slicing the Y and Z axes, keeping the X-axis undistributed as shown on Figure 2.
2. 1D FFTs are applied along the X-axis, followed by FFTs along the Y and Z axes after transposing the matrix to ensure a full axis on each GPU, as shown on Figure 3.

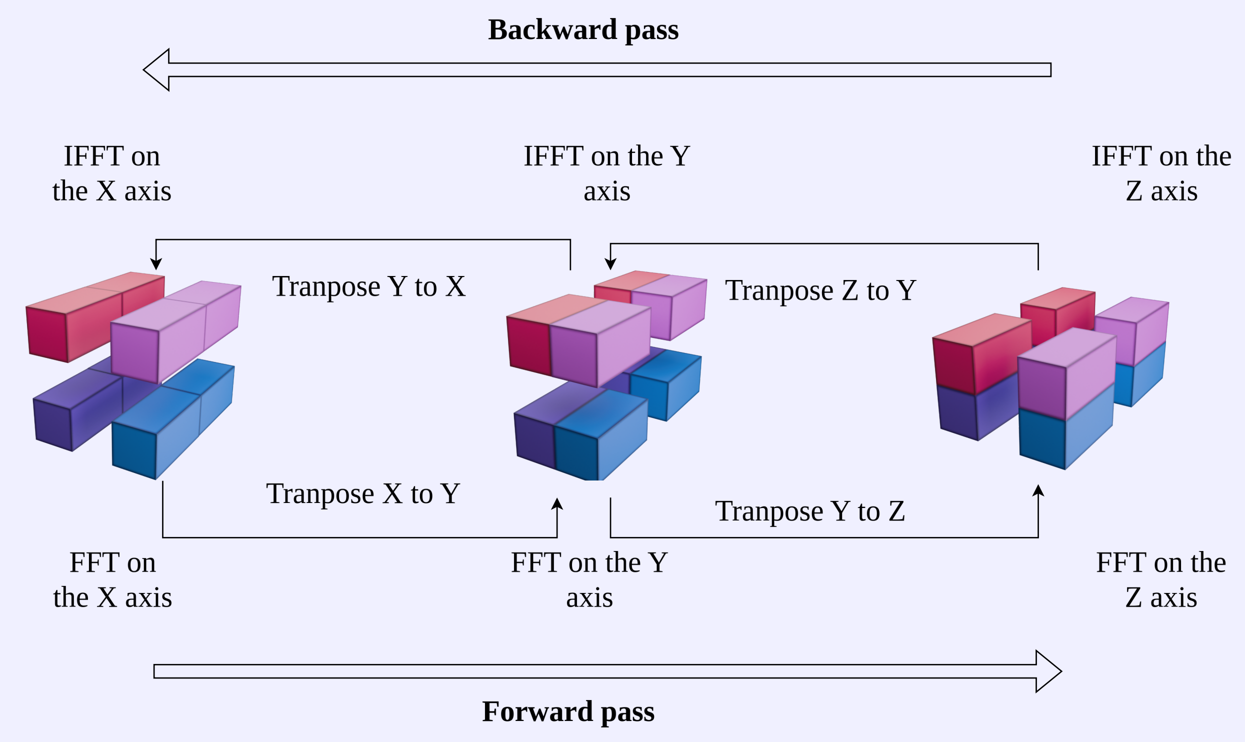


Figure 3: Forward and backward pass of the distributed 3D FFTs.

3. the inverse Fourier transforms (IFFT) are performed in reverse order to the FFT operations, completing the backward pass.

Distribution and memory benchmarks

We performed benchmarks of our `PFFT3D` function on both the number of GPUs, and on various data size, up to 2048 cubes (~70GB), and the results can be seen on Figure 4.

- `PFFT3D` is able to scale in term of memory to any given size, whereas `JAX` base own FFT cannot go beyond 1024^3 on a GPU with 32GB memory,
- `PFFT3D` with NCCL backend outperforms in speed the `JAX` FFT implementation on all simulation sizes.

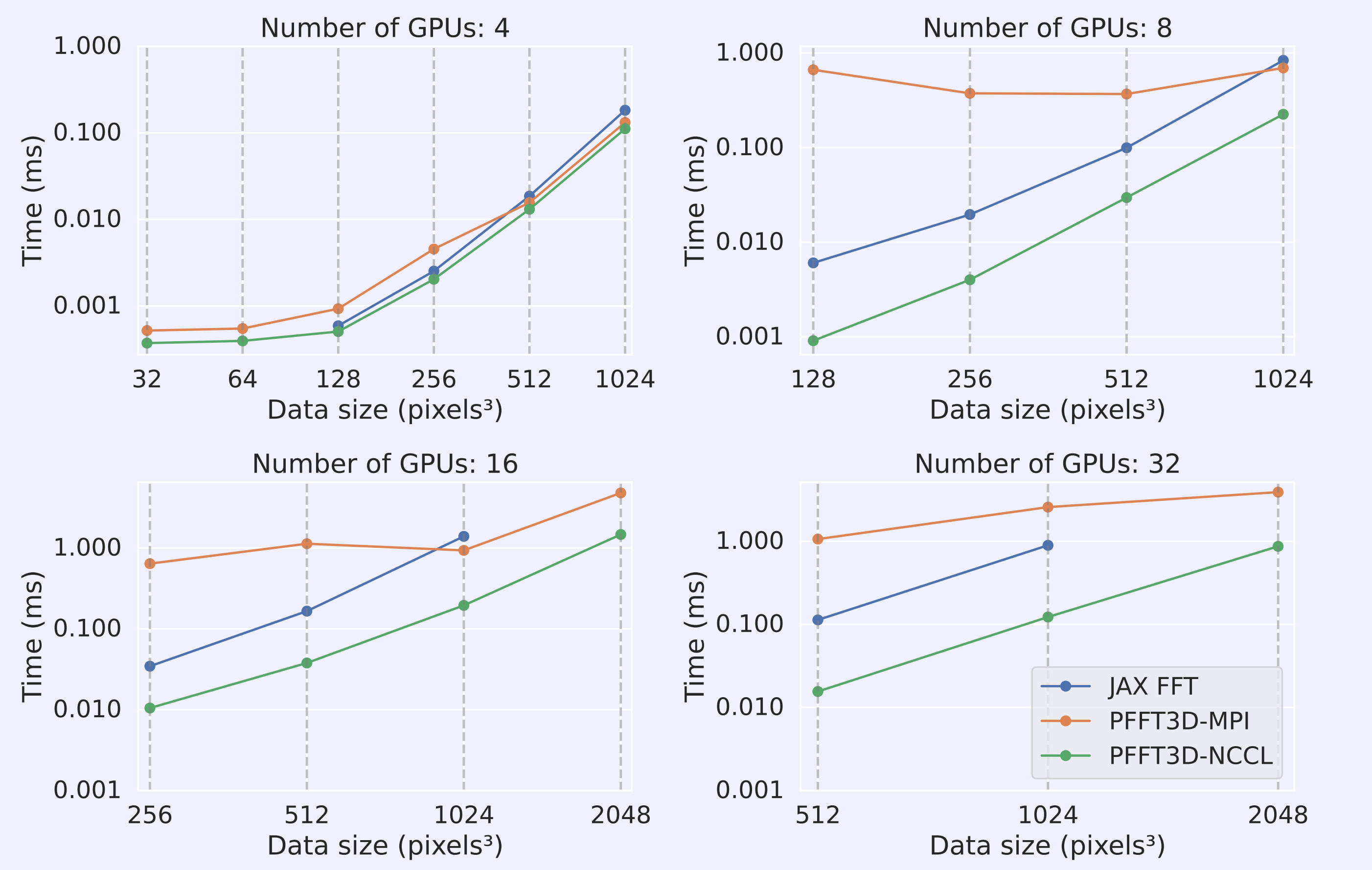


Figure 4: Comparison of a single forward pass of a 3D distributed FFT between the pure `JAX` implementation and our `PFFT3D` function using two backends, MPI and NVIDIA NCCL, as a function of the size of the simulation. The four plots correspond to an increasing number of GPUs.

Conclusions

- With this work on distributed 3D FFT operations across multiple GPUs, we have unlocked a first step towards fully differentiable and highly scalable N-body simulations.
- This approach will soon be integrated into the `JaxPM` and `pmwd` libraries, and we plan to release the code as an open-source library in the near future.
- We plan to pave the way for other HPC demanding cosmological applications like differentiable spherical harmonics transform.

References

Bradbury, James et al. (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.4.25. URL: `http://github.com/google/jax`.

Feng, Yu et al. (Aug. 2016). "FastPM: a new scheme for fast simulations of dark matter and haloes". In: *Monthly Notices of the Royal Astronomical Society* 463.3, 2273–2286. ISSN: 1365-2966. DOI: `10.1093/mnras/stw2123`. URL: `http://dx.doi.org/10.1093/mnras/stw2123`.

Rolfo, Stefano et al. (2023). "The 2DECOMP&FFT library: an update with new CPU/GPU capabilities". In: *Journal of Open Source Software* 8.91, p. 5813. DOI: `10.21105/joss.05813`. URL: `https://doi.org/10.21105/joss.05813`.

Romero, Joshua, Pedro Costa, and Massimiliano Fatica (2022). "Distributed-memory simulations of turbulent flows on modern GPU systems using an adaptive pencil decomposition library". In: *Proceedings of the Platform for Advanced Scientific Computing Conference*. PASC '22. Basel, Switzerland: Association for Computing Machinery. ISBN: 9781450394109. DOI: `10.1145/3539781.3539797`. URL: `https://doi.org/10.1145/3539781.3539797`.

Acknowledgements: This work was granted access to the HPC resources of IDRIS under the allocation 2024-AD011014949 made by GENCI