# HealpixMPI.jl: an MPI-parallel implementation of the Healpix tessellation scheme in Julia

Leo A. Bianchi ⓘ [1,2]

1 Dipartimento di Fisica Aldo Pontremoli, Università degli Studi di Milano, Milan, Italy **2** Institute of Theoretical Astrophysics, University of Oslo, Blindern, Oslo, Norway

Summary

Spherical Harmonic Transforms (SHTs) can be seen as Fourier Transforms' spherical, two-dimensional counterparts, casting real-space data to the spectral domain and vice versa. As in Fourier analysis where a function is decomposed into a set of amplitude coefficients, an SHT allows any spherically-symmetric field, defined in real space, to be decomposed into a set of complex harmonic coefficients $a_{\ell,m}$, commonly referred to as alms, where each quantifies the contribution of the corresponding spherical harmonic function.

SHTs are important for a wide variety of theoretical and practical scientific applications, including particle physics, astrophysics, and cosmology. However, SHTs are generally computationally expensive operations and thus often constitute the *bottleneck* of the scientific software they are part of. For this reason, much effort has been spent over the last couple of decades to obtain fast and efficient SHT implementations. In such a setting, parallel computing naturally comes into play, especially for time-consuming software to be run on large High-Performance Computing (HPC) clusters.

The Julia package `HealpixMPI.jl` constitutes an extension package of `Healpix.jl` (Tomasi & Li, 2021), efficiently parallelizing its SHT-related functionalities. `Healpix.jl` is a Julia-only implementation of the HEALPix (Gorski et al., 2005) library, which provides one of the most used two-sphere tessellation schemes and a series of SHTs-related functions.

The main goal of the Julia package presented in this paper, `HealpixMPI.jl`, is to efficiently employ a large number of computing cores to perform fast spherical harmonic transforms. This paper presents the key features implemented to achieve this, together with a statement of need and the results of a parallel scaling test.
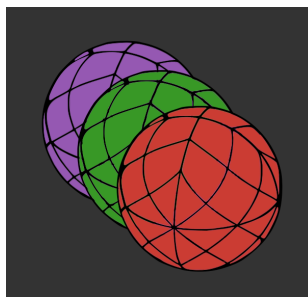
Figure 1: `HealpixMPI.jl`'s logo

Statement of need

Together with a variety of applications, spherical harmonic transforms are extremely relevant in different cosmological research topics, e.g., Loureiro et al. (2023) and Euclid Collaboration (2023). Among those, SHTs are essential for the analysis of cosmic microwave background (CMB) radiation, which is one of the most active cosmology research areas. CMB radiation is, in fact, very conveniently described as a temperature (and polarization) field on the celestial sphere, making spherical harmonics the most natural mathematical tool for analyzing its measured signal. On the other hand, from a computational point of view, CMB field measurements need, of course, to be discretized, requiring a mathematically consistent pixelization of the sphere and the functions defined on it. This is exactly the goal HEALPix was targeting when it was released more than two decades ago; it quickly became the standard library for CMB numerical analysis. HEALPix code can be, of course, used for a wider variety of applications, but its bond with CMB analysis has always been particularly strong, especially given the research focus of its main authors. Not surprisingly, the cosmic microwave background is also the research context wherein `HealpixMPI.jl` was born.

SHTs are often the computational bottleneck of CMB data analysis pipelines, as the one implemented by Cosmoglobe (Watts et al., 2023) based on the software Commander (Eriksen et al., 2004). Given the significantly increasing amount of data produced by the most recent observational experiments, efficient algorithms alone are no longer enough to perform SHTs within acceptable run times, and a parallel architecture must be implemented. In the specific case of Cosmoglobe and Commander, the goal for the next years is to be able to run a full pipeline, and thus the SHTs performed in it, on large HPC clusters *efficiently* employing at least 10^4 cores.

To achieve this, an implementation of massively parallel spherical harmonic transforms beyond machine-size limitations is unavoidably needed. The concept of `HealpixMPI.jl` was born as a contribution to Cosmoglobe's pipeline targeting this exact goal.

The latest SHT engine: DUCC

As of the time this paper was submitted, `Healpix.jl` relied on the SHTs provided by the C library `libsharp` (Reinecke & Seljebotn, 2013). However, `libsharp`'s development ceased a few years ago, and its functionalities have been included as an SHT sub-module in DUCC (Reinecke, 2019), an acronym of "Distinctively Useful Code Collection."

The timing between the development of `HealpixMPI.jl` and a Julia interface for DUCC has been quite fortunate. This allowed `HealpixMPI.jl` to be up-to-date with the state of the art of spherical harmonics upon its first release. In fact, DUCC's code is derived directly from `libsharp`, but has been significantly enhanced with the latest algorithmical improvements as well as the employment of standard C++ multithreading for *shared-memory* parallelization of the core operations.

Hybrid parallelization of the SHT

To run SHTs on a large number of cores, i.e., on an HPC cluster, `HealpixMPI.jl` provides a hybrid parallel design, based on simultaneous usage of multithreading and MPI, for shared- and distributed-memory parallelization respectively, as shown in Figure 2.
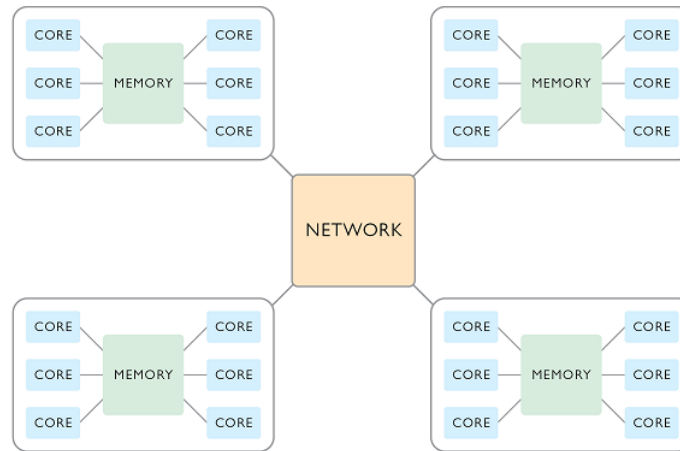
Hybrid parallel computer

Figure 2: Multi-node computing cluster representation. The optimal way to parallelize operations such as the SHTs on a cluster of computers is to employ MPI to share the computation *between* the available nodes, assigning one MPI task per node, and multithreading to parallelize *within* each node, involving as many CPUs as locally available. Figure taken from www.comsol.com.

In the case of 'HealpixMPI.jl', native C++ multithreading is provided by `DUCC` for its spherical harmonic transforms by default; while the MPI interface is entirely coded in Julia and based on the package `MPI.jl` (Byrne et al., 2021).

Moreover, the MPI parallelization requires data to be distributed across the MPI tasks. As shown in the usage examples, this is implemented by mirroring `Healpix.jl`'s classes with two new *distributed* data types: `DAlm` and `DMap`, encoding the harmonic coefficients and a pixelized representation of the spherical field respectively.

Usage example

An usage example with all the necessary steps to set up and perform an MPI-parallel `alm2map` SHT can be found on the front page of `HealpixMPI.jl`'s repository.

In addition, refer to Jommander, a parallel and Julia-only CMB Gibbs Sampler, for an example of code based on `HealpixMPI.jl`.

Scaling results

This section shows the results of parallel benchmark tests conducted on `HealpixMPI.jl`. In particular, a strong-scaling scenario is analyzed: given a problem of fixed size, the wall time improvement is measured as the number of cores exploited in the computation is increased.

To obtain a reliable measurement of massively parallel spherical harmonics wall time is certainly nontrivial, especially for tests employing a high number of cores; intermittent operating system activity (aka, jitter) can significantly distort the measurement of short time scales. For this reason, the benchmark tests were carried out by timing a batch of 20 `alm2map` + `adjoint_alm2map` SHT pairs. For reference, the scaling shown here is relative to unpolarized spherical harmonics with $N_{side} = 4096$ and $\ell_{max} = 12287$ and were carried out on the Hyades cluster of the University of Oslo. The benchmark results are quantified as the wall time multiplied by the total number of cores, shown in a 3D plot (Figure 3) as a function of the number of local threads and MPI tasks (always one per node).
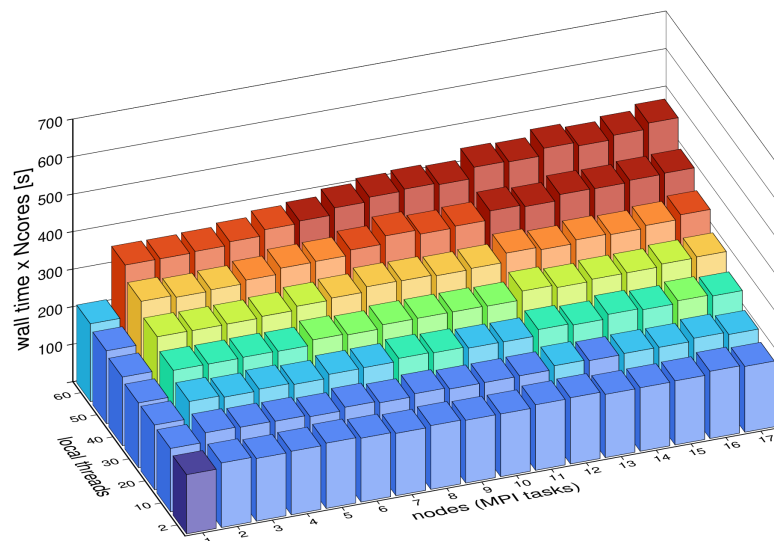
Figure 3: The measured wall time is multiplied by the total number of cores used and plotted as a function of the number of local threads and MPI tasks used. The total number of cores corresponding to each column is given by the product of these two quantities.

Increasing the number of threads on a single core, for which no MPI communication is needed, the scaling results nearly ideal up to ~ 50 cores. For 60 and higher local threads we start observing a slight slowdown, probably given by the many threads simultaneously trying to access the same memory, hitting its bandwidth limit.

While switching to a multi-node setup, we introduce, as expected, an overhead given by the necessary MPI communication whose size, unfortunately, remains constant as we increase the number of local threads. This leads to the ramp-like shape along the "local threads" axis shown by the plot. However, the overhead size scales down, even if not perfectly, when we increase the number of nodes, as the size of the locally stored data will linearly decrease. This is shown by the relatively flat shape of the plot along the "nodes"-axis.

Acknowledgements

References

Byrne, S., Wilcox, L. C., & Churavy, V. (2021). MPI.jl: Julia bindings for the message passing interface. *Proceedings of the JuliaCon Conferences*, *1*(1), 68. https://doi.org/10.21105/jcon.00068

Eriksen, H. K., O'Dwyer, I. J., Jewell, J. B., Wandelt, B. D., Larson, D. L., Gorski, K. M., Levin, S., Banday, A. J., & Lilje, P. B. (2004). Power spectrum estimation from high-resolution maps by Gibbs sampling. *The Astrophysical Journal Supplement Series*, *155*(2), 227–241. https://doi.org/10.1086/425219

Euclid Collaboration. (2023). *Euclid preparation: XXVIII. Modelling of the weak lensing angular power spectrum*. https://arxiv.org/abs/2302.04507

Gorski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., & Bartelmann, M. (2005). HEALPix: A framework for high-resolution discretization and fast analysis of data distributed on the sphere. *The Astrophysical Journal*, *622*(2), 759–771. https://doi.org/10.1086/427976

Loureiro, A., Whiteaway, L., Sellentin, E., Lafaurie, J. S., Jaffe, A. H., & Heavens, A. F. (2023). Almanac: Weak lensing power spectra and map inference on the masked sphere. *The Open Journal of Astrophysics*, *6*. https://doi.org/10.21105/astro.2210.13260

Reinecke, M. (2019). DUCC. In *GitLab repository*. GitLab. https://gitlab.mpcdf.mpg.de/mtr/ducc

Reinecke, M., & Seljebotn, D. S. (2013). Libsharp – spherical harmonic transforms revisited. *Astronomy & Astrophysics*, *554*, A112. https://doi.org/10.1051/0004-6361/201321494

Tomasi, M., & Li, Z. (2021). *Healpix.jl: Julia-only port of the HEALPix library* (Version 3.0, p. ascl:2109.028).

Watts, D. J., Basyrov, A., Eskilt, J. R., Galloway, M., Gjerløw, E., Hergt, L. T., Herman, D., Ihle, H. T., Paradiso, S., Rahman, F., Thommesen, H., Aurlien, R., Bersanelli, M., Bianchi, L. A., Brilenkov, M., Colombo, L. P. L., Eriksen, H. K., Franceschet, C., Fuskeland, U., … Zhou, Y. (2023). COSMOGLOBE DR1 results: I. Improved Wilkinson microwave anisotropy probe maps through Bayesian end-to-end analysis. *Astronomy &Amp; Astrophysics*, *679*, A143. https://doi.org/10.1051/0004-6361/202346414