




# 1 HealpixMPI.jl: an MPI-parallel implementation of the 2 Healpix tessellation scheme in Julia

3 **Leo A. Bianchi**  <sup>1,2</sup>

4 <sup>1</sup> Dipartimento di Fisica Aldo Pontremoli, Università degli Studi di Milano, Milan, Italy <sup>2</sup> Institute of  
5 Theoretical Astrophysics, University of Oslo, Blindern, Oslo, Norway

DOI: [10.xxxxxx/draft](https://doi.org/10.xxxxxx/draft)

## Software

- [Review](#) 
- [Repository](#) 
- [Archive](#) 

Editor: [Prashant K Jha](#) 

## Reviewers:

- [@marcobonici](#)
- [@baxmittens](#)

Submitted: 05 February 2024

Published: unpublished

## License

Authors of papers retain copyright  
and release the work under a  
Creative Commons Attribution 4.0  
International License ([CC BY 4.0](#)).

## 6 Summary

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

13 SHTs are important for a wide variety of theoretical and practical scientific application, including  
14 particle physics, astrophysics and cosmology. However, the SHTs are in general computationally  
15 expensive operations and thus often constitute the *bottleneck* of the scientific software they  
16 are part of. For this reason, many efforts have been spent over the last couple of decades to  
17 obtain fast and efficient SHTs implementations. In such a setting, parallel computing naturally  
18 comes into play, especially for heavy software to be run on large High Performance Computing  
19 (HPC) clusters.

20 The Julia package `HealpixMPI.jl` constitutes an extension package of `Healpix.jl` ([Tomasi  
21 & Li, 2021](#)), providing an efficient parallelization of its SHTs functionalities. `Healpix.jl` is a  
22 Julia-only implementation of the HEALPix ([Gorski et al., 2005](#)) library, which provides one of  
23 the most used tassellation schemes of the two-sphere together with a series of SHTs-related  
24 functions.

25 The main goal of the Julia package `HealpixMPI.jl`, presented in this paper, is to efficiently  
26 employ a high number of computing cores to perform fast spherical harmonic transforms. The  
27 key features implemented to achieve this, together with a statement of need and the results of  
28 a parallel scaling test are presented in this paper.

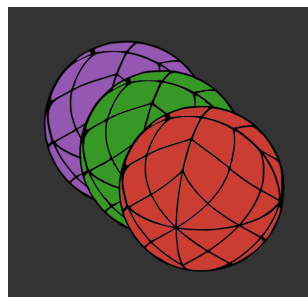


Figure 1: HealpixMPI.jl's logo

## Statement of need

Among a variety of applications, spherical harmonic transforms are particularly relevant for the analysis of cosmic microwave background (CMB) radiation, which is among the most active research fields of recent cosmology. CMB radiation is in fact very conveniently described as a temperature (and polarization) field on the sky sphere, making spherical harmonics the most natural mathematical tool to analyze 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 more than two decades ago was released, quickly becoming 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.

In order 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

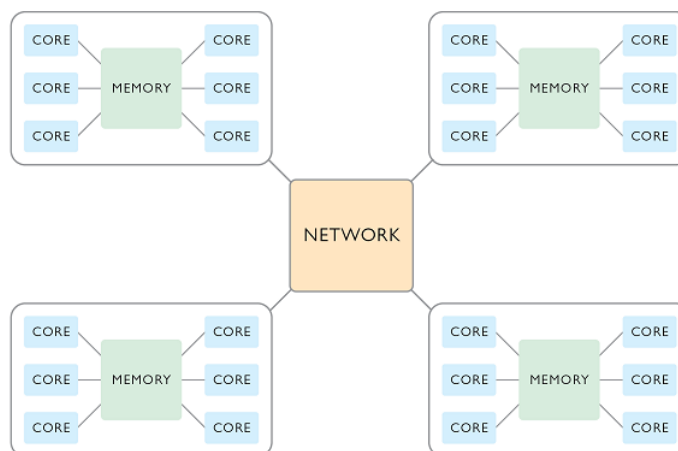
As of the time of this paper being submitted, Healpix.jl relies on the SHTs provided by the C library libsharp (Reinecke & Seljebotn, 2013). However, since a few years ago, libsharp's development has ceased and its functionalities have been included, as an SHT sub-module, in DUCC (Reinecke, 2019), acronym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

The timing between the development of HealpixMPI.jl and a Julia interface for DUCC has been quite lucky. This allowed HealpixMPI.jl to be already up-to-date with the state of the art of spherical harmonics upon its first release. In fact, for what concerns the SHTs, 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 a HPC cluster, HealpixMPI.jl provides a hybrid parallel design, based on a 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](http://www.comsol.com).

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

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

## Usage example

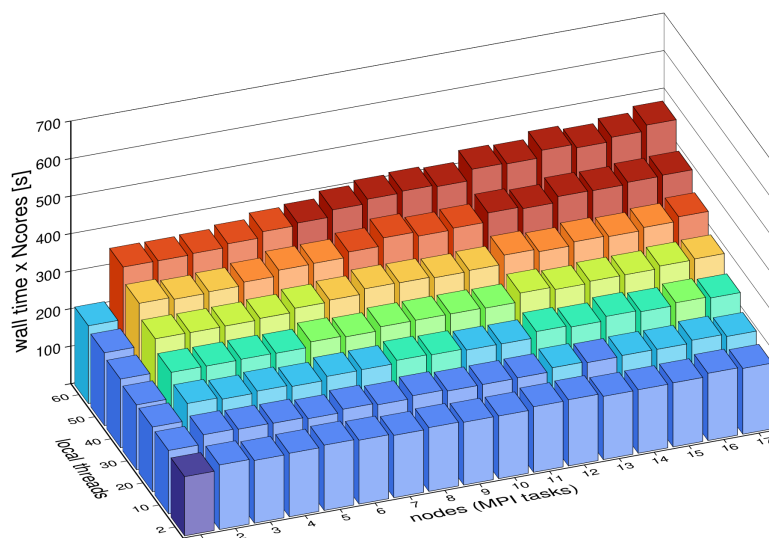
76 An usage example with all the necessary steps to set up and perform an MPI-parallel alm2map  
77 SHT can be found in the front page of HealpixMPI.jl's [repository](#).

78 In addition, refer to [Jommander](#), a parallel and Julia-only CMB Gibbs Sampler, for an example  
79 of code based on HealpixMPI.jl.

## Scaling results

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

84 To obtain a reliable measurement of massively parallel spherical harmonics wall time is certainly  
85 nontrivial: especially for tests implying a high number of cores, intermittent operating system  
86 activity can significantly distort the measurement of short time scales. For this reason, the  
87 benchmark tests were carried out by timing a batch of 20 alm2map + adjoint\_alm2map SHT  
88 pairs. For reference, the scaling shown here is relative to unpolarized spherical harmonics with  
89  $N_{\text{side}} = 4096$  and  $\ell_{\text{max}} = 12287$  and were carried out on the [Hyades cluster](#) of the University  
90 of Oslo. The benchmark results are quantified as the wall time multiplied by the total number  
91 of cores, shown in a 3d-plot ([Figure 3](#)) as a function of the number of local threads and MPI  
92 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 of course given by the product of these two quantities.

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

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

## Acknowledgements

103  
104 The development of `HealpixMPI.jl`, as a part of my master thesis, has been funded by the  
105 University of Milan, through a “Thesis Abroad Grant”. Moreover, I acknowledge significant  
106 contributions to this project from Maurizio Tomasi, Martin Reinecke, Hans Kristian Eriksen  
107 and Sigurd Næss; as well as the support I received from all the members of Cosmoglob  
108 collaboration during my stay at the Institute of Theoretical Astrophysics of the University of  
109 Oslo.

## References

- 110  
111 Byrne, S., Wilcox, L. C., & Churavy, V. (2021). MPI.jl: Julia bindings for the message passing  
112 interface. *Proceedings of the JuliaCon Conferences*, 1(1), 68. <https://doi.org/10.21105/jcon.00068>  
113  
114 Eriksen, H. K., O’Dwyer, I. J., Jewell, J. B., Wandelt, B. D., Larson, D. L., Gorski, K. M., Levin,  
115 S., Banday, A. J., & Lilje, P. B. (2004). Power spectrum estimation from high-resolution  
116 maps by gibbs sampling. *The Astrophysical Journal Supplement Series*, 155(2), 227–241.  
117 <https://doi.org/10.1086/425219>

- 118 Gorski, K. M., Hivon, E., Banday, A. J., Wandelt, B. D., Hansen, F. K., Reinecke, M., &  
119 Bartelmann, M. (2005). HEALPix: A framework for high-resolution discretization and fast  
120 analysis of data distributed on the sphere. *The Astrophysical Journal*, 622(2), 759–771.  
121 <https://doi.org/10.1086/427976>
- 122 Reinecke, M. (2019). DUCC. In *GitLab repository*. GitLab. [https://gitlab.mpcdf.mpg.de/mtr/](https://gitlab.mpcdf.mpg.de/mtr/ducc)  
123 [ducc](https://gitlab.mpcdf.mpg.de/mtr/ducc)
- 124 Reinecke, M., & Seljebotn, D. S. (2013). Libsharp – spherical harmonic transforms revisited.  
125 *Astronomy & Astrophysics*, 554, A112. <https://doi.org/10.1051/0004-6361/201321494>
- 126 Tomasi, M., & Li, Z. (2021). *Healpix.jl: Julia-only port of the HEALPix library* (Version 3.0,  
127 p. ascl:2109.028).
- 128 Watts, D. J., Basyrov, A., Eskilt, J. R., Galloway, M., Gjerløw, E., Hergt, L. T., Herman, D.,  
129 Ihle, H. T., Paradiso, S., Rahman, F., Thommesen, H., Aurlen, R., Bersanelli, M., Bianchi,  
130 L. A., Brilenkov, M., Colombo, L. P. L., Eriksen, H. K., Franceschet, C., Fuskeland, U., ...  
131 Zhou, Y. (2023). COSMOGLOBE DR1 results: I. Improved wilkinson microwave anisotropy  
132 probe maps through bayesian end-to-end analysis. *Astronomy & Astrophysics*, 679,  
133 A143. <https://doi.org/10.1051/0004-6361/202346414>

DRAFT