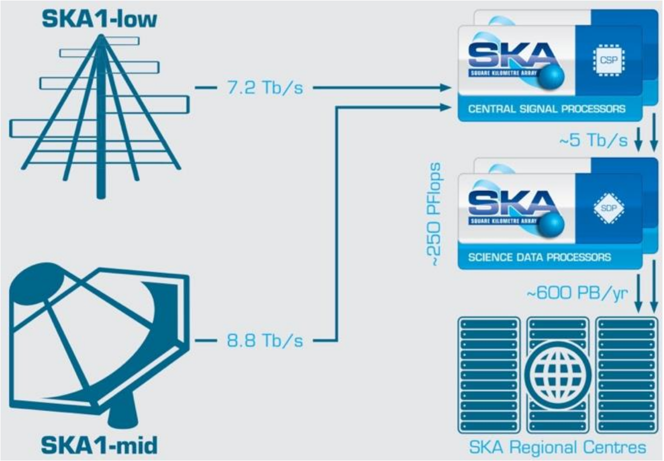
Finanziato dall'Unione europea NextGenerationEU
Ministero dell'Università e della Ricerca
Italiadomani PIANO NAZIONALE DI RIPRESA E RESILIENZA
ICSC Centro Nazionale di Ricerca in HPC, Big Data and Quantum Computing

Evolution of RICK into a robust, user-ready, library for interferometric imaging

***De Rubeis Emanuele**, Claudio Gheller, Giovanni Lacopo, Giuliano Taffoni, Luca Tornatore*

Spoke 3 II Technical Workshop, Bologna Dec 17 -19, 2024

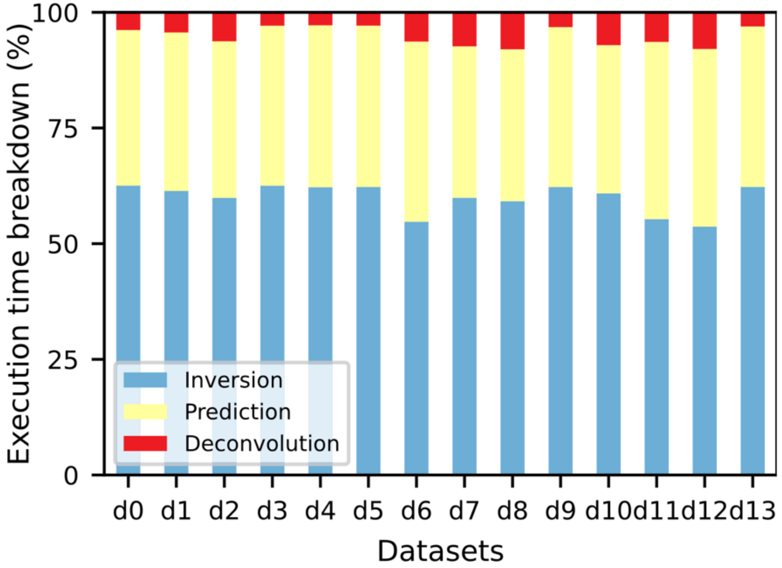
Scientific Rationale



Why HPC for radio astronomy?

Current and upcoming radio-interferometers are expected to produce **volumes of data of increasing size**. This means that current **state-of-the-art software needs to be re-designed** to handle such unprecedented **data challenge**.

Imaging in radio astronomy represents one of the most **computational demanding** steps of the processing pipeline, both in terms of memory request and in terms of computing time.

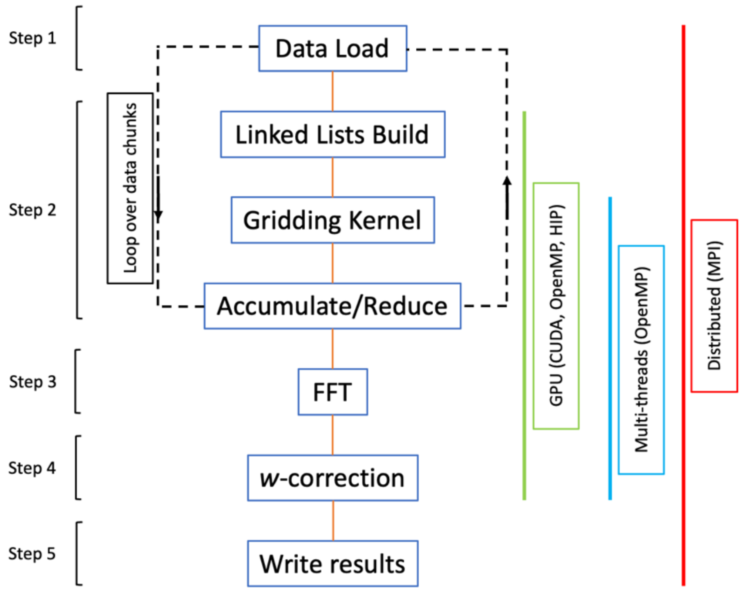


Corda et al. (2022)

Technical Objectives, Methodologies and Solutions

RICK (Radio Imaging Code Kernels) is a code that addresses the ***w*-stacking algorithm** (Offringa+14) for imaging, combining parallel and accelerated solutions.

- The code is written in **C** (with extensions to **C++**)
- **MPI** & **OpenMP** for CPU parallelization
- The code is capable of **running full on NVIDIA GPUs**, using CUDA for offloading
- HIP or OpenMP are also available for other architectures (such as AMD)



Adapted from De Rubeis et al. (2025)

Main Results

The code is **publicly available on the Spoke3 Github** (https://github.com/ICSC-Spoke3/RICK)

About

Development of a code for radio astronomy imaging enabled to exploit heterogeneous HPC resources.

- Readme
- Activity
- Custom properties
- **0** stars
- **1** watching
- **0** forks

Report repository

Releases

No releases published
Create a new release

Packages

No packages published
Publish your first package

Languages

- **C** 46.5%
- **Cuda** 16.4%
- **C++** 15.7%
- **Shell** 13.5%
- **Makefile** 4.5%
- **Python** 2.9%
- **MATLAB** 0.5%

Radio Imaging Code Kernels

Radio Imaging Code Kernels (RICK) is a software that is able to exploit parallelism and accelerators for radio astronomy imaging. **This software is currently under development**.

RICK is written in C/C++ and can perform the radio interferometric inversion through the following routines:

- gridding
- Fast Fourier Transform (FFT)
- w-correction

It exploits the Message Passing Interface (MPI) and OpenMP for parallelism, and is able to run on both NVIDIA and AMD GPUs using CUDA, HIP, and OpenMP for GPU offloading.

If you use RICK for your work please cite De Rubeis et al. (2024).

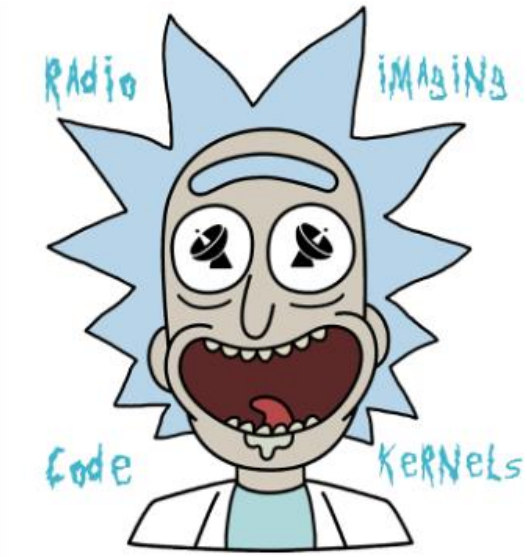
Why and where to use RICK?

RICK can be used to test its performances for your personal dataset. It has been tested for several radio interferometers:

- Low Frequency Array (LOFAR)
- Jansky Very Large Array (JVLA)
- Atacama Large Millimeter Array (ALMA)

If you tested RICK on an interferometer not present on this list, please contact us with a brief comment or report on your experience.

Together with this, we are currently working on transforming RICK into a C library that can be called from within the most-used softwares for imaging (such as WSClean or CASA), so that it can provide final, cleaned images usable for scientific purposes. More info about the library can be found into the *library* branch.



We also wrote a tentative of *Wiki*...

Main Results

A paper has been accepted on Astronomy and Computing (De Rubeis et al. 2025), describing the full, NVIDIA GPU offloading of the code and scaling performances.

Full length article

Accelerating radio astronomy imaging with RICK

E. De Rubeis [a,b,*], G. Lacopo [c,d], C. Gheller [b], L. Tornatore [c], G. Taffoni [c]

[a] *Dipartimento di Fisica e Astronomia, Università di Bologna, via Gobetti 93/2, I-40129 Bologna, Italy*
[b] *Istituto di Radioastronomia, INAF, Via Gobetti 101, 40121 Bologna, Italy*
[c] *Astronomical Observatory of Trieste INAF, via GB Tiepolo 11, 34143 Trieste, Italy*
[d] *Dipartimento di Fisica, Università degli studi di Trieste, via Alfonso Valerio 2, 34127 Trieste, Italy*

ARTICLE INFO

Keywords:
High performance computing
Radio astronomy
Interferometry
Data analysis

ABSTRACT

This paper presents an implementation of radio astronomy imaging algorithms on modern High Performance Computing (HPC) infrastructures, exploiting distributed memory parallelism and acceleration throughout multiple GPUs. Our code, called RICK (Radio Imaging Code Kernels), is capable of performing the major steps of the w-stacking algorithm presented in Offringa et al. (2014) both inter- and intra-node, and in particular has the possibility to run entirely on the GPU memory, minimising the number of data transfers between CPU and GPU. This feature, especially among multiple GPUs, is critical given the huge sizes of radio datasets involved.

After a detailed description of the new implementations of the code with respect to the first version presented in Gheller et al. (2023), we analyse the performances of the code for each step involved in its execution. We also discuss the pros and cons related to an accelerated approach to this problem and its impact on the overall behaviour of the code. Such approach to the problem results in a significant improvement in terms of runtime with respect to the CPU version of the code, as long as the amount of computational resources does not exceed the one requested by the size of the problem: the code, in fact, is now limited by the communication costs, with the computation that gets heavily reduced by the capabilities of the accelerators.

Main Results

RICK has been tested on Leonardo (CINECA, #9 Top500 Nov. 2024), using real LOFAR-VLBI data as representative of SKA pathfinder data volumes.

- **Full-CPU** (MPI+OpenMP) and **full-GPU** (MPI+CUDA) tests.
- Strong and weak scaling.
- Different data and image size configurations.
- Analysis based on single steps of the code:
  - gridding
  - FFT
  - *w*-correction
  - total

Table 2
Configuration and computational mesh used in the *Small*, *Intermediate* and *Large* tests. The mesh size takes into account the total amount of memory required for the real and imaginary part depending on the size of the grid.

	Small	Intermediate	Large
N. visibilities (approx)	0.54×10^9	47.05×10^9	47.05×10^9
Input data size (GB)	4.4	533	533
$N_u = N_v \times N_w$	$4096^2 \times 16$	$16384^2 \times 32$	$65536^2 \times 32$
Mesh size (GB)	10.81	258.81	4098.81

De Rubeis et al. (2025)

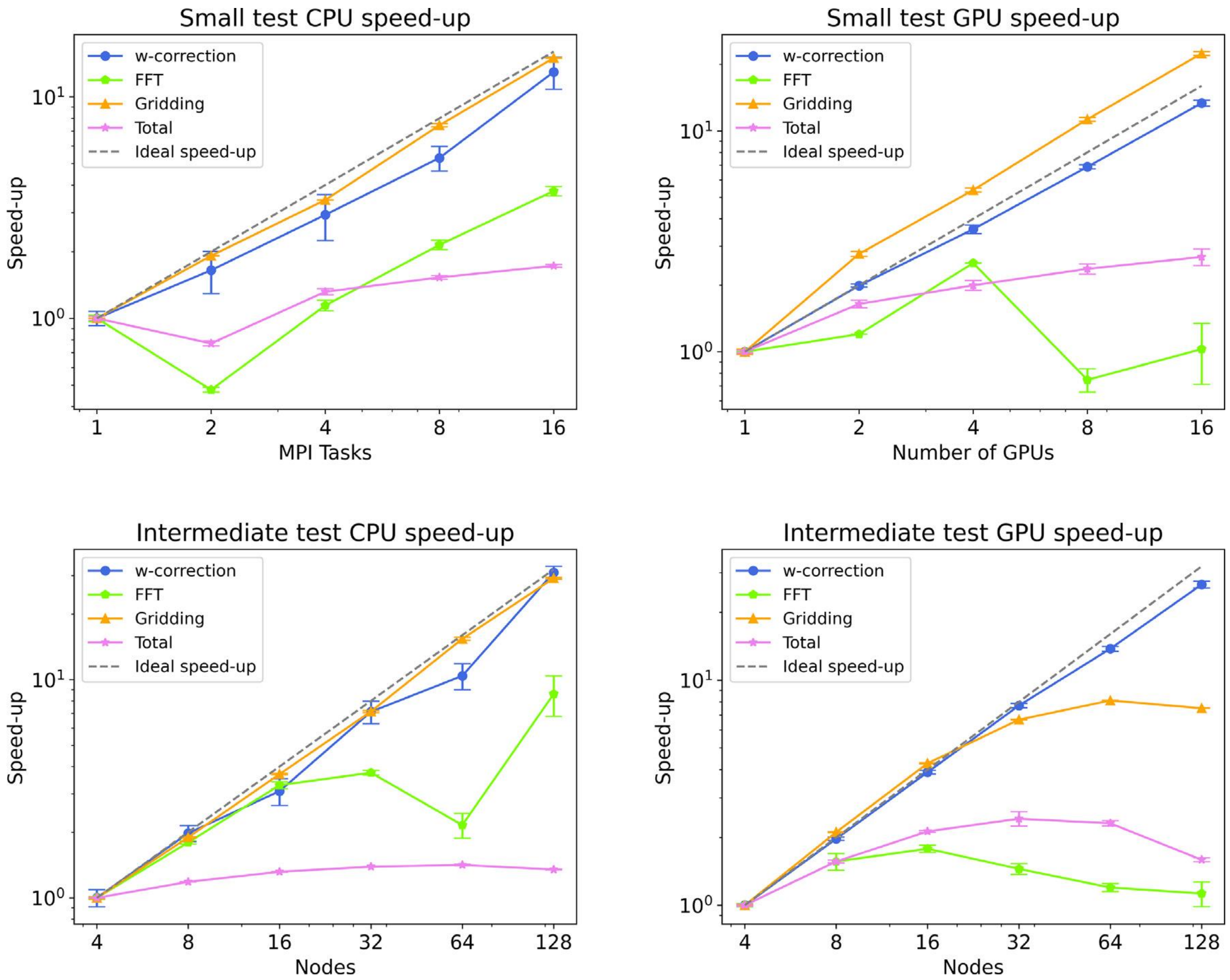
Main Results

Strong scaling tests

- 1 MPI task <-> 1 GPU

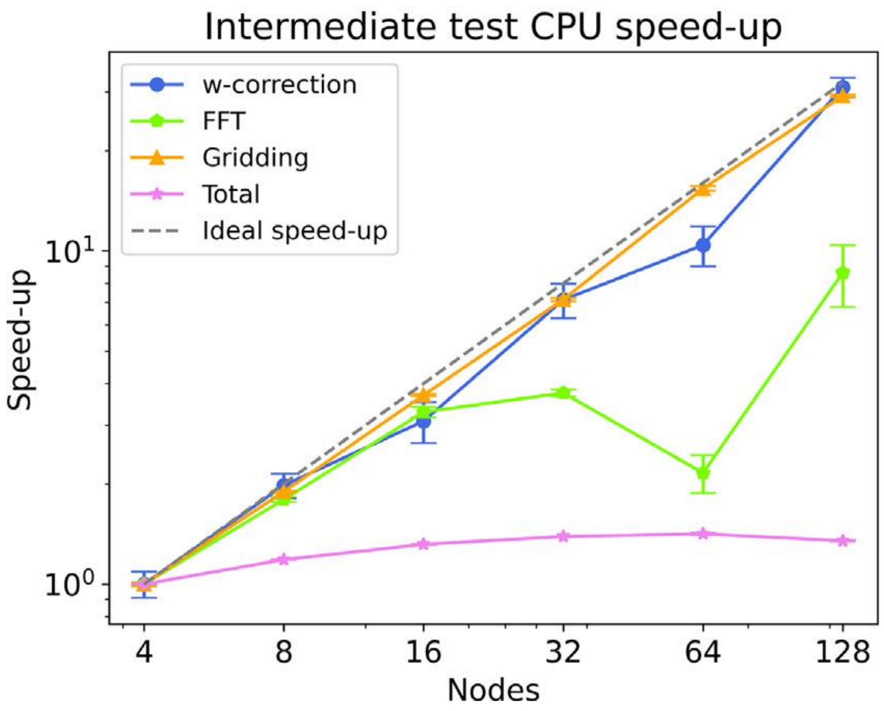
- Intermediate: 4 MPI tasks (or GPUs) + 8 OpenMP threads per node

$$S_p = \frac{T_1}{T_{N_p}}$$

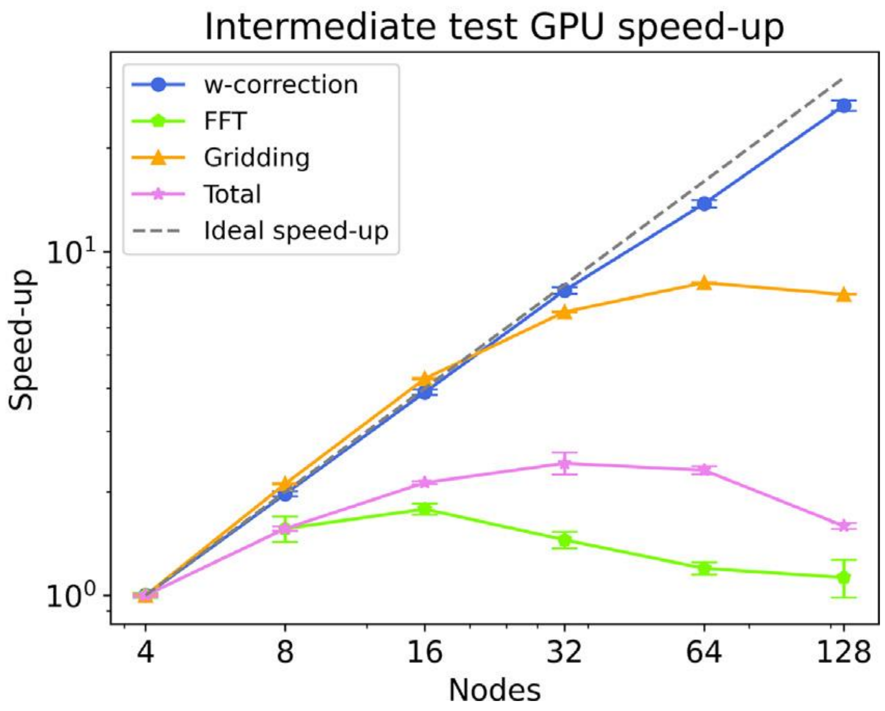


De Rubeis et al. (2025)

Main Results



FULL-CPU



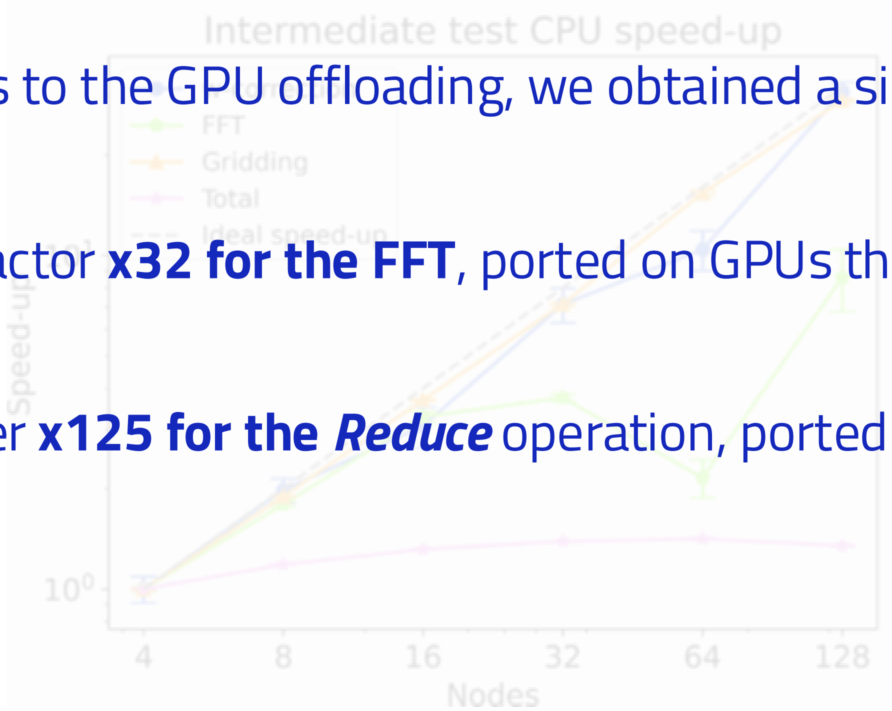
FULL-GPU

De Rubeis et al. (2025)

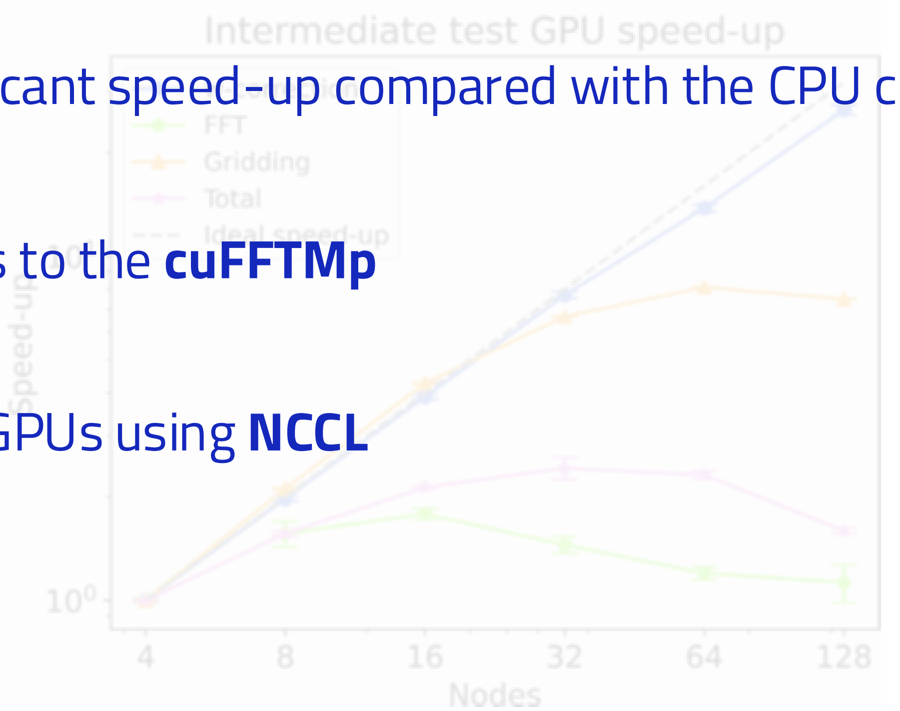
Main Results

Thanks to the GPU offloading, we obtained a significant speed-up compared with the CPU code.

- A factor **x32 for the FFT**, ported on GPUs thanks to the **cuFFTMp**
- Over **x125 for the *Reduce*** operation, ported on GPUs using **NCCL**



FULL-CPU



FULL-GPU

De Rubeis et al. (2025)

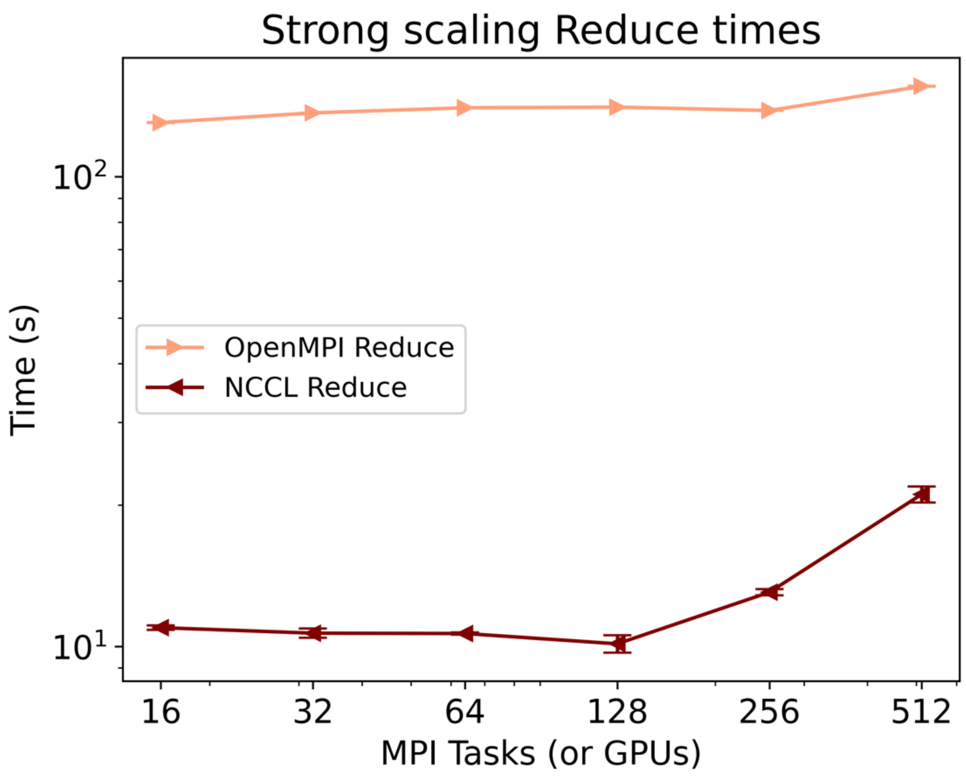
Main Results

But… watch out for the reduce!

RICK is now limited by the communication costs, with the reduce operation that becomes the true bottleneck of the code.

This means that it is important to choose the right number of computational resources based on the problem size.

Increasing power does not necessarily return an increasing performance!



De Rubeis et al. (2025)

Ongoing work

After the code development, our goal is now to **convert RICK into a library** that can be called from within any, commonly-used, code for interferometric imaging (such as WSClean or CASA).

To do this, we had to "extract" the main steps of the code into individual, **self-consistent functions**, to avoid having any dependence on the global variables used by the RICK environment

```
void gridding(
    int rank,
    int size,
    int nmeasures,
    double *uu,
    double *vv,
    double *ww,
    double *grid,
    double *gridss,
    MPI_Comm MYMPI_COMM,
    int num_threads,
    int grid_size_x,
    int grid_size_y,
    int w_support,
    int num_w_planes,
    int polarisations,
    int freq_per_chan,
    float *visreal,
    float *visimg,
    float *weights,
    double uvmin,
    double uvmax)
```

```
void fftw_data(
    int grid_size_x,
    int grid_size_y,
    int num_w_planes,
    int num_threads,
    MPI_Comm MYMPI_COMM,
    int size,
    int rank,
    double *grid,
    double *gridss)
```

```
void phase_correction(
    double *gridss,
    double *image_real,
    double *image_imag,
    int num_w_planes,
    int xaxistot,
    int yaxistot,
    double wmin,
    double wmax,
    double uvmin,
    double uvmax,
    int num_threads,
    int size,
    int rank,
    MPI_Comm MYMPI_COMM)
```

These scripts are into the *library* branch of the RICK Github repo (*Wiki* for the library still to be completed...)

Ongoing work

After the code development, our goal is now to **convert RICK into a library** that can be called from within any, commonly-used, code for interferometric imaging (such as WSClean or CASA).

To do this, we had to "extract" the main steps of the code into individual, **self-consistent functions**, to avoid having any dependence on the global variables used by the RICK environment

Each function can be compiled individually as a **dynamic library**, and called from within **any** (even Python) **script**

Ongoing work

We are also working to

- Properly combine the visbilities correlations to get ***Stokes I*** flux values, useful to get true, scientific fluxes.
- Implement different **weighting** schemes to change the resolution of the radio maps

$$
\begin{aligned}
I &= < |e_x|^2 + |e_y|^2 > \\
Q &= < |e_x|^2 - |e_y|^2 > \\
U &= 2 < |e_x||e_y| \cos\delta > \\
V &= 2 < |e_x||e_y| \sin\delta >
\end{aligned}
$$

Hamaker & Bregman (1996)

Accomplished results

From Elba meeting (May 2024)...

KPI	Target	Deadline	
Release of the v2.0	Release of RICK v2.0	December 31st, 2023	✓
	New Reduce and FFT on GPUs available, code fully on GPUs	December 31st, 2023	✓
Paper to be submitted	Paper on the v2.0 release of the code	M8	✓
Release of the *library* refactoring	Weighting and uv-tapering, parallel and accelerated version	M9	
	RICK dynamic library, self-consistent functions (both CPU and GPU)	M9	✓

Next steps

- Completion of tests for ***Stokes I* flux values** and **weighting strategies**
- Porting of the RICK library on AMD GPUs, exploiting the latest **distributed library for FFT on AMD** GPUs (rocFFT)
- Trying to **insert RICK library into WSClean**, to make it also scientifically productive