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

Accelerated radio astronomy with RICK

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

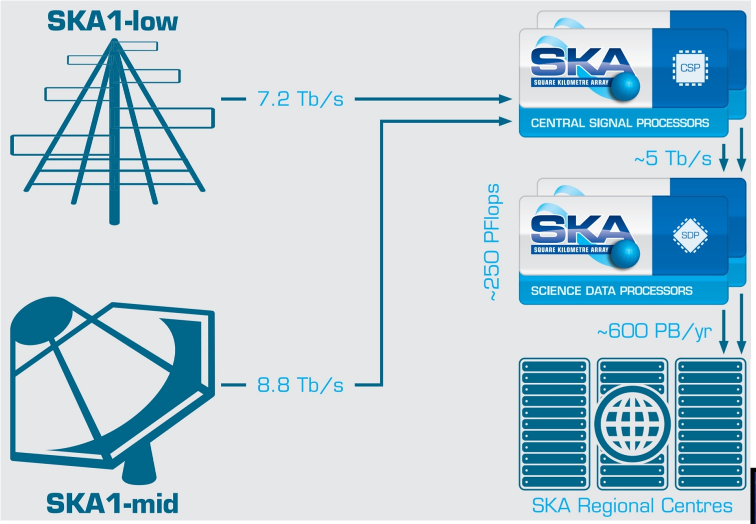
Spoke 3 General Meeting, Elba 5-9 / 05, 2024

Scientific Rationale

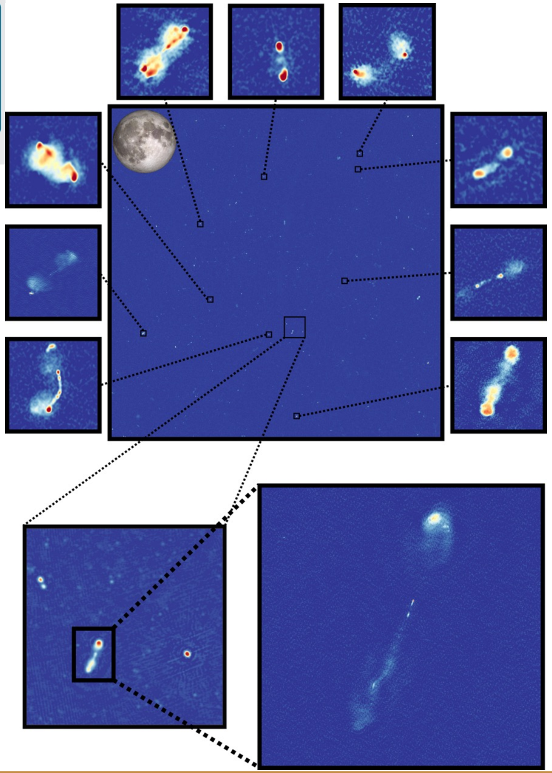
Why HPC for radio astronomy?

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.



Sweijen et al. (2022)

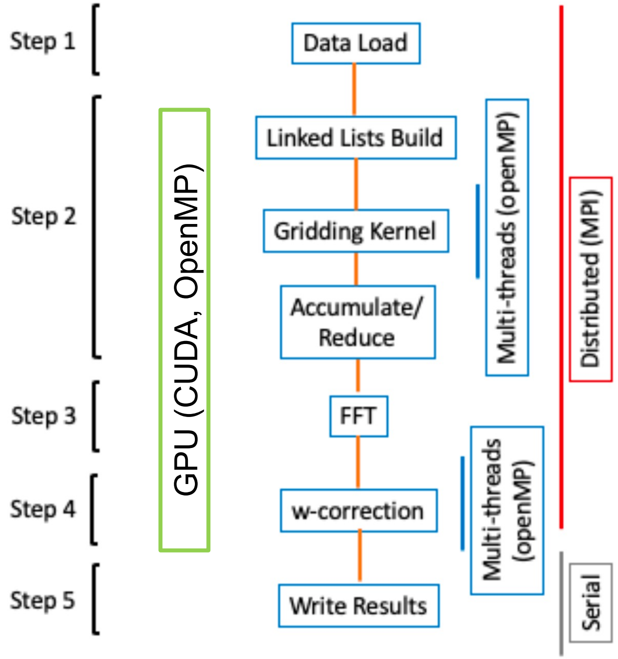


For example, this over 7 billions pixels image can take ~250,000 core hours!

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.

- **C**, **C++**, **CUDA**, **HIP**
- **MPI** & **OpenMP** for CPU parallelization
- The code is now capable of **running full on GPUs**, using CUDA, HIP or OpenMP for offloading
- An optimized version of the **reduce** has been developed on both CPU (combining MPI+OpenMP) and GPU (using **NCCL** or RCCL, for Nvidia and AMD respectively); the **FFT** is done through the **cuFFTMp** library for Nvidia
- Currently under benchmarking on **Leonardo** (CINECA, No.4 Top500 June 23)



Adapted from Gheller et al. (2023)

Technical Objectives, Methodologies and Solutions

Why do we need multiple GPU?

Modern and future radio interferometers will produce a huge amount of data, that hardly fit into the memory of a single GPU (not even a single node)

↓

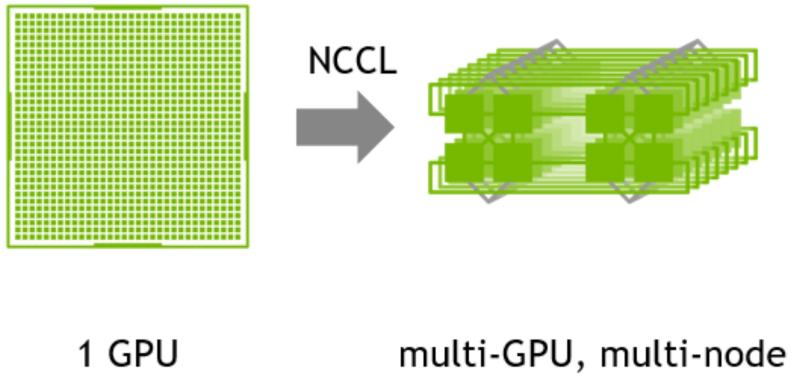
The solution is to distribute the problem among multiple GPUs and multiple nodes

Technical Objectives, Methodologies and Solutions

NVIDIA Collective Communication Library (NCCL)

NCCL is a library of multi-GPU collective communication used to support the *Reduce* operation.

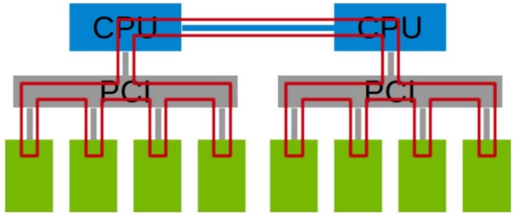
- Provides **fast collectives** over **multiple GPUs both intra- and inter-node**.
- Supports a variety of **interconnection technologies** (e.g. NVLink, PCIe).
- NCCL closely follows the popular collectives API defined by **MPI**, so can be very "natural" to use.



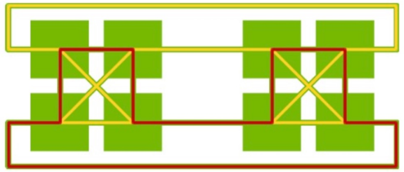
Technical Objectives, Methodologies and Solutions

NVIDIA Collective Communication Library (NCCL)

NCCL implements the *Reduce* operation as an intra-node **ring**, and an inter-node **ring**, when GPUs assigned to the main tasks communicate with RDMA with GPUs in different nodes **without passing through the CPUs**.



PCIe / QPI : 1 unidirectional ring



DGX-1 : 4 unidirectional rings

```
ncclUniqueId id;
ncclComm_t comm;

ncclCommInitRank(&comm, size, id, rank);

ncclReduce(send_g, recv_g, size_g, ncclDouble, ncclSum, taget_rank, comm, stream)
```

Technical Objectives, Methodologies and Solutions

NVIDIA Collective Communication Library (NCCL)

NCCL implements the *Reduce* operation as an intra-node **ring**, and an inter-node **ring**, when GPUs assigned to the main tasks communicate with RDMA with GPUs in different nodes **without passing through the CPUs**.

```
ncclUniqueId id;
ncclComm_t comm;

ncclCommInitRank(&comm, size, id, rank);

ncclReduce(send_g, recv_g, size_g, ncclDouble, ncclSum, taget_rank, comm, stream)
```

The requirement of a dedicated stream for the *Reduce* comes from the presence of asynchronous memory copies that collided with the ones within a previous function call

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

Technical Objectives, Methodologies and Solutions

cuFFTMp

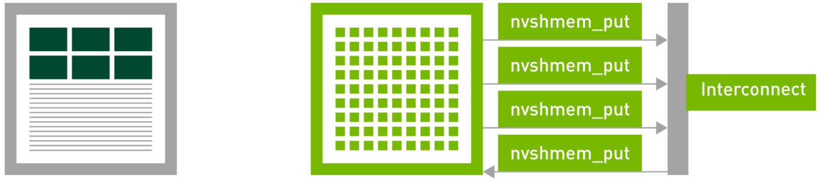
Fast Fourier Transform (FFT) is a **critical operation** in radio astonomy, because it determines the relationship between the "observed" and the "desired" data (the final image).

For the FFT step, RICK now implements the cuFFTMp library, that enables the **distribution of the FFT problem using NVSHMEM**.

NVSHMEM uses asynchronous, GPU-initiated, data transfers, **avoiding synchronization overheads** between the CPU and the GPU.

MPI

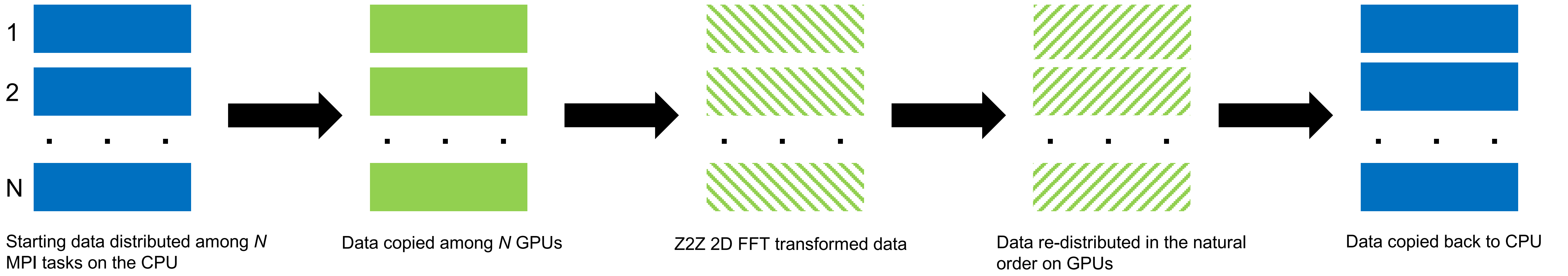
NVSHMEM



Technical Objectives, Methodologies and Solutions

cuFFTMp

Data are distributed among multiple GPUs and inverse-transformed.



Technical Objectives, Methodologies and Solutions

cuFFTMp

- We may need to to this FFT process even 100s-1000s times, and each time we need to create and destroy a *descriptor*, which is the ad-hoc data structure used by the cuFFTMp library for the FFT.

✓ This was critical for the performance, but we overcame this problem using CUDA kernels to write the *to-be-transformed* data for each loop, and then writing them directly inside the descriptor.

- The joint usage of NVSHMEM and NCCL can cause severe runtime errors during the FFT.

➢ There is the possibility to switch off NCCL support for NVSHMEM at runtime by setting NVSHMEM_DISABLE_NCCL=1.

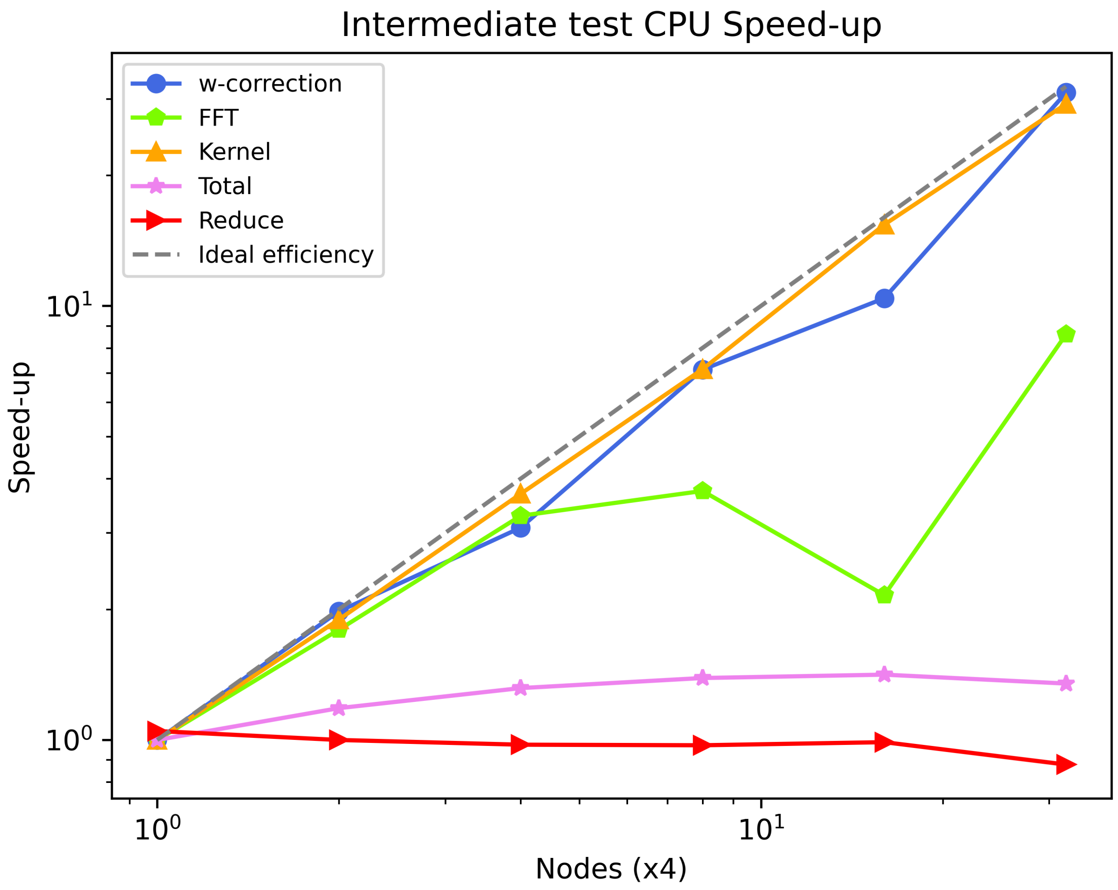
Accomplished Work, Results

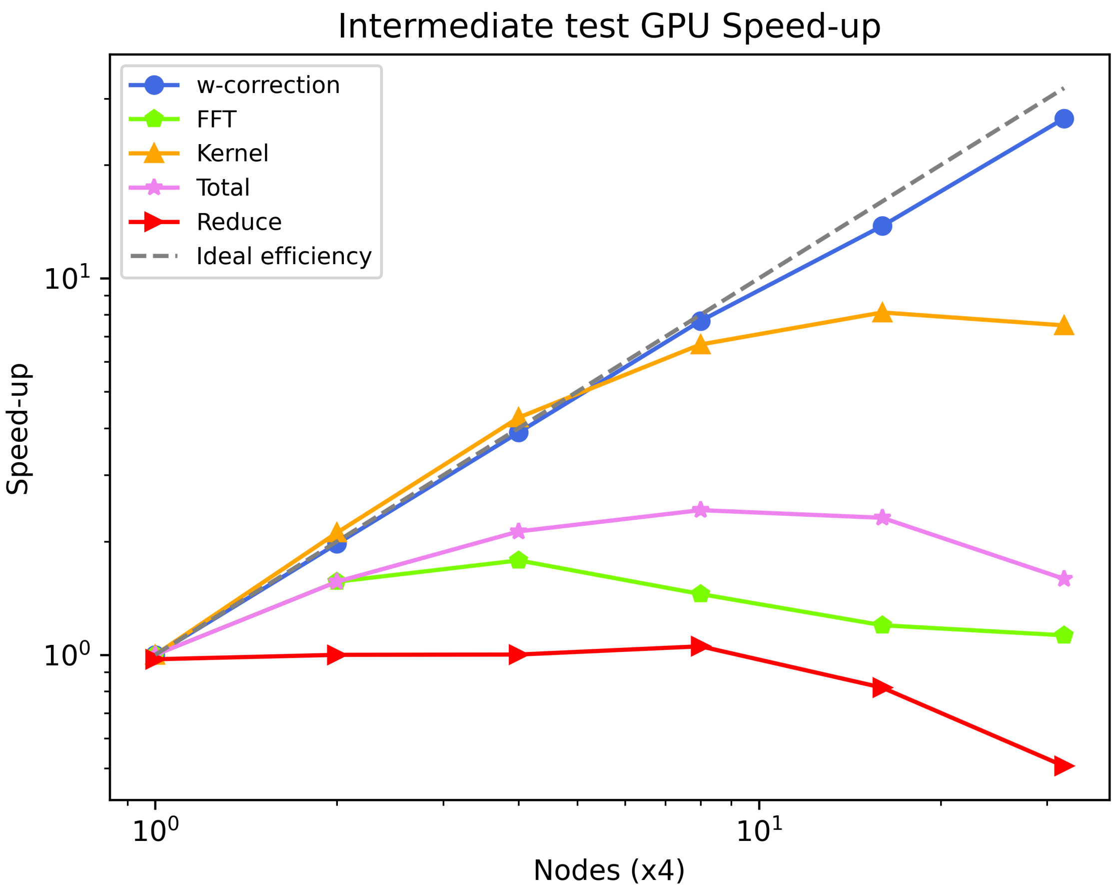
Currently, we are testing RICK on Leonardo (CINECA) using NVIDIA HPC-SDK (v 23.11). The testing dataset are **real, LOFAR-VLBI data**, the closest facility to SKA in terms of overall data size.

Comparing the code on GPUs, with respect to the one on CPUs, we obtained:

- for a small input dataset (4GB) we reached a speed-up factor up to **~x27** for both Reduce and FFT
- for a large input dataset (530GB) we reached a speed-up factor of **~x175** for the Reduce and **~x32** for the FFT

Accomplished Work, Results





Timescale, Milestones and KPIs

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 v.2.1	Weighting and uv-tapering, parallel and accelerated version	M9
	Optimization of memory occupation on GPUs	M9

Next Steps and Expected Results

- With RICK we test the benefits that the exploitation of HPC resources can bring to future generation of radio interferometers, in our case for imaging data

- Preliminary results suggest that the **improvement could possibly be huge**, meaning that this can represent a **promising approach** to the SKA (and precursors) data volumes

- The code is now capable of **working fully on NVIDIA GPUs with CUDA**.
  Full AMD GPUs support is not yet available because of the lack of a proprietary distributed library for FFT (such as cuFFTMp for NVIDIA)

- The large data size requires distributing the workload among multiple nodes: in return, we obtain that the **comunication** becomes our main bottleneck. **The key is the right choice of computational resources**.