HPC and accelerators exploitation for radio imaging software: a step towards the SKA era

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

Spoke 3 Technical Workshop, Trieste October 9 / 11, 2023
Scientific Rationale

Current and upcoming radio interferometers are expected to produce volumes of data of increasing size that need to be collected, transferred, processed and stored.

The Square Kilometre Array (SKA) is expected to produce ~300 petabytes of data per year, but we are already facing this challenge with SKA pathfinders, such as LOFAR and MeerKAT.

Imaging, in particular, is the most computationally-demanding step of the whole data reduction, especially when large fields-of-view and/or high-resolution images are required.

For example, this 6.6 deg² (83,950x83,500 pixels) image with LOFAR-VLBI required 250,000 CPU hours

Sweijen et al. (2022)
Technical Objectives, Methodologies and Solutions

For this reason, we want to implement parallelism and accelerators for the most demanding steps of imaging, namely gridding, Fast Fourier Transform (FFT) and w-correction.

Our code, named Radio Imaging Code Kernels (RICK), exploits HPC resources for radio astronomy imaging:

- **Written in C/C++**
- **MPI** and **OpenMP** for parallelism
- **CUDA, OpenMP** and **HIP** for GPU offloading

Gheller et al. (2023)
Technical Objectives, Methodologies and Solutions

Our goal is to have the **whole code working on GPUs**, in order to minimize the data movement between host and device: for this, several solutions have been adopted

- For the FFT step, the **cuFFT** library from NVIDIA has been implemented. This allows to distribute the FFT among several GPUs, crucial for large sized problems, using NVSHMEM

Starting data distributed among $N$ MPI tasks on the CPU

Data copied among $N$ GPUs

Z2Z 2D FFT transformed data

Data re-distributed in the natural order on GPUs

Data copied back to CPU
Technical Objectives, Methodologies and Solutions

Our goal is to have the whole code working on GPUs, in order to minimize the data movement between host and device: for this, several solutions have been adopted

- For the FFT step, the cuFFTMp library from NVIDIA has been implemented. This allows to distribute the FFT among several GPUs, crucial for large sized problems, using NVSHMEM

- Relies on descriptors, ad-hoc data structure

- Descriptors need to be defined and allocated-freed within each loop on the w-direction

- Necessity of a copy DtD, in order to re-distribute data in the correct order
Technical Objectives, Methodologies and Solutions

Our goal is to have the whole code working on GPUs, in order to minimize the data movement between host and device: for this, several solutions have been adopted:

- The `reduce` has been ported on GPUs thanks to NCCL
- RDMA in-node and inter-node for GPU-GPU communication
- Portable on AMD GPUs
- High-speed technology for GPU-GPU connection available in most HPC platforms

- The high-speed interconnection can be exploited intra-node only
Accomplished milestones

- **Winter 2023**
  - Introduction of the CFITSIO library to write outputs in parallel in FITS format

- **Spring 2023**
  - Implementation of the cuFFT and cuFFTMp to port the FFT step on GPUs

- **Summer 2023**
  - ISCRA-C for Leonardo (CINECA) for benchmarks

- **Fall 2023**
  - Tests with real LOFAR-VLBI data
  - Paper accepted at ADASS 2023
Accomplished Work, Results

Are those two the same thing?

4096 x 4096 x 100
Accomplished Work, Results

- 1 GPU per MPI task
- With cuFFTmp we are able to handle the problem only starting from 8 GPUs
Accomplished Work, Results

- The NCCL reduce on GPU is faster with respect to the MPI reduce on CPU by a factor of \(x50\) within the node.
- Among multiple nodes we cannot use anymore NVLink.
Accomplished Work, Results

- We found a problem with the OpenMPI implementation in Leonardo (CINECA), possibly related to the wrong allocation of buffers
### Next targets and KPI (by next checkpoint: April 2024)

<table>
<thead>
<tr>
<th>KPI</th>
<th>Target</th>
<th>Deadline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper for ADASS 2023</td>
<td></td>
<td>November 2023</td>
</tr>
<tr>
<td>Release of the v2.0</td>
<td>Complete the reduce on CPU</td>
<td>December 31st, 2023</td>
</tr>
<tr>
<td></td>
<td>New reduce and FFT on GPUs available</td>
<td>December 31st, 2023</td>
</tr>
<tr>
<td>Paper submission</td>
<td>Paper on the v2.0 release of the code</td>
<td>January 31st, 2024</td>
</tr>
<tr>
<td>Release of the v.2.1</td>
<td>Weighting and uv-tapering, parallel and accelerated version</td>
<td>April 30th, 2024</td>
</tr>
<tr>
<td></td>
<td>Computation and communication overlap</td>
<td>April 30th, 2024</td>
</tr>
</tbody>
</table>