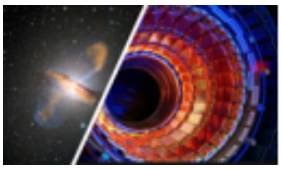


PhD course of National Interest in Technologies for Fundamental Research in Physics and Astrophysics

Annual Report

PhD Student : Saim Ali
Cycle and a.a : 39th, 2023/24
Supervisor: Dr. Fabio Roberto Vitello, Dr. Eva Sciacca

Title: Analysis of astrophysical phenomena using efficient and parallelized models on HPC computing systems.

Aims and objectives:

The aim of the research is efficiently processing and analyzing large astrophysical datasets. The cosmological simulations and Astrophysics generates massive datasets from simulations like the DEMNUni suite, which model the universe's evolution. A single-threaded process struggles with massive datasets due to speed and memory limitations. Parallel computing using **MPI, openMP and openACC** overcomes this with dividing the task into multi processing and multi threading and taking data in chunks efficiently.

Research Project:

The project uses VisIVO Server, an open source platform specifically designed for astrophysical data visualization and analysis. The three main components are: **VisIVO Importer, VisIVO Filter** and **VisIVO Viewer.**

The VisIVO Server provides a variety of filters, with my research focusing on the **Point Distribute Filter**. This filter generates a table representing a volume derived from selected fields of the input table, distributed using the **NGP (Nearest Grid Point), CIC (Cloud-In-Cell, by default), or TSC (Triangular Shaped Cloud)** algorithms. The filter exclusively accepts the VBT (VisIVO Binary Table) data format for processing.

CIC (Clouds-in-cells method): The Point Distribute Filter employs three interpolation algorithms: **Cloud-In-Cell (CIC), Triangular Shaped Cloud (TSC), and Nearest Grid Point (NGP)**. I parallelize the CIC algorithm for efficient data distribution and interpolation by mapping positions, computing weights, and distributing density across grid points.

The data for this research is provided in Gadget format from DEMNUni simulations. These simulations use the Tree Particle Mesh-Smoothed Particle Hydrodynamics (TreePM-SPH) code GADGET-3. The DEMNUni suite consists of large-scale cosmological N-body simulations, referred to as the "Dark Energy and Massive Neutrino Universe" (DEMNUni). They track the evolution of Cold Dark Matter (CDM) and Hot Dark Matter (HDM) neutrino particles, treating them as two separate collisionless species.

Research Progress: To enhance the efficiency of the Point Distribute Filter, a strategy for parallelization using MPI (Message Passing Interface), and openMP was developed in the Ist year and openACC in 2nd year with full implementation in the 2nd year with fully functional code by the second year of my PhD.

Here is the detailed parallelization using **MPI, openMP and openACC**.

1. Data Distribution with MPI (Feb to April 2025)

The parallelization with **MPI** is designed as a highly effective **distributed reading model**, where multiple processes simultaneously read their own non overlapping chunks of data. The code is separating argument parsing (ParametersParser) from the core filter logic (within startFilter). **MPI** is used to enable parallel execution across multiple processes. This implementation strategically centralizes initial checks and data distribution on Rank 0 to ensure consistency and efficiency in a distributed memory environment. Process Rank 0 handles all parameter parsing, and it can stop the entire process early if critical issues like a missing input file are detected.

The **MPI_Bcast** is used for filename length, filename data, and exit_flag. It efficiently propagates critical state information (validated parameters or error conditions) from Rank 0 to all other processes. The specific data reading logic within VSTable or the op.execute() method, is designed to read only their assigned chunk from the input file. This prevents redundant file opening and parsing by individual processes, which would be highly inefficient in a distributed reading model. The multiple MPI_Barrier calls synchronization, ensuring all processes have reached a specific stage (e.g. after initial parameter check, after filename distribution, after final table validation) before proceeding to the next phase of computation.

2. Data Collection and Reduction (may to june 2025):

The code achieves scalability by using a more advanced two-stage reduction strategy. Instead of multiple MPI reduce calls on each grid point, it consolidates the entire grid of

data into a single, optimized highly collective MPI reduce operation. This ensures that MPI reduce calls are made only once after thread level reduction is completed by each process. Following the main computational loop, each process gathers the results from its local threads, summing contributions there into a single, contiguous buffer. The second stage uses MPI for a global reduction across processes using one MPI reduce call. This operation aggregates the contributions from all processes into a single, final result on the root process, which is then used to update the main data structure.

The entire reduction operation is not the bottleneck but the computational loop of the CIC. This has been confirmed by recording runtimes of corresponding sections of the code. The code scalability is not optimal, but it is still reasonable with increasing number of threads due to file I/O (input output) being done on each iteration specifically, the **getcolumn()** and **putcolumn()** functions. It however accomplishes reasonable scalability with an increasing number of MPI processes as the file **I/O operations** do not hinder process level concurrency.

3. Parallel Processing with OpenMP (july to august 2025):

The data is first divided among multiple MPI processes, where each process is assigned a unique chunk of data to read and process (start_chunk to end_chunk). Inside each MPI process, the **#pragma omp parallel for** directive is defined to further parallelize the main computation loop into multithreading. This directive tells the OpenMP runtime to distribute the iterations of the **for** loop among the available **threads**. The OpenMP parallelization is implemented within the if(m_cic) block, leveraging **shared-memory parallelism** to accelerate the computation by utilizing the multiple cores of a modern CPU. **omp_get_thread_num()** returns a unique integer ID for the current thread that is executing the code. These IDs are assigned sequentially, starting from 0 up to the total number of threads.

Now finally I parallelized the code with the Hybrid Model (MPI and openMP) to get better scalability, and efficiency but still I did not achieve the high scalability of the execution time of the code as compared to the serial code.

4. OpenAcc (Open Accelerators)(sep to oct 2025): By replacing OpenMP with OpenACC **pragmas #pragma acc parallel**, computations can be executed on the GPUs. I Installed the **OpenAcc** on my Personal Computer and I am working on the implementation as I have not yet acheived the scalability of the code with openACC, but hopefully at the end of my 2nd year I will be able to achieve the required scalabilty.

Here is the link to my GitHub repository which contains code parallelized with MPI and OpenMP.

https://github.com/itxsaimali/VisIVOServer-PointDistribute-miniapp/tree/Parallel-code

Challenges and action taken: During the implementation I faced a lot of challenges like:

1. I had to develop a hybrid model with MPI and openMP which was so challenging for me then on the suggestions of my supervisors I divided the task in this way that first implements MPI and then openMP.
2. First I was using block-reading logic, which was not efficient and fruitful. Then I switched to a distributed reading model with each process reading data independently.
3. I faced difficulties in the chunk mechanism when I tested the code with large data sets. The number of rows was not sufficient, so
4. I worked on the code to fix the issue that every process only read its own chunk of data.
5. There was duplication in a parallel environment and a lack of basic validation like parse and check parameters, filename and create a VSTable object, which was not allowed then I fixed that issue with validated parameters from Rank 0 to all other processes.
6. The major issue that i am facing is the scalability of the execution time for MPI, openMP, and openACC, which is yet not achieved with high scale, still i am working on it to get it as soon as possible.

List of attended courses and passed exams

1. Machine learning for Physics, 3 CREDITS Exam done
2. High energy physics detectors in space, 2,5 CREDITS Exam done
3. Deep network and structure learning 2 CREDITS Exam done
4. Machine learning programming for Physics, 3 CREDITS Exam done

List of attended conferences, workshops and schools.

1. **Course on computing and HPC for Astronomy and Astrophysics**

   (24 June 2024 to 5 July 2024 - Bologna, https://indico.ict.inaf.it/event/2785/)

2. **TECH-FPA PhD Retreat 2025**
   Gran Sasso National Laboratory (LNGS) of the National Institute for Nuclear Physics (INFN) L'Aquila, Italy (17/02/2025 – 21/02/2025)
3. **INAF Workshop on Astro Statistic 2024**
   National Institute for Astrophysics, Catania Italy (07/10/2024 – 09/10/2024)
4. **Parallel Programming with MPI and OpenMP**

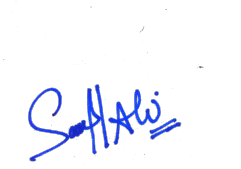
   Johannes Gutenberg-Universität Mainz, Germany. (01/04/2025 – 04/05/2025)

5. **Italian Language and Culture Course - A2, 2024/2025**
   University of Catania, **ItalStra**

List of published papers/proceedings.

Thesis title (even temporary)

Analysis of astrophysical phenomena using efficient and parallelized models on HPC computing systems.

Date, 01/10/2025 Signature 

The supervisors Signature :