DOI: 10.1051/epjconf/201713601006

The pLISA project in ASTERICS

Giulia De Bonis[1,a] and Cristiano Bozza[2,3,b]

[1] *Istituto Nazionale di Fisica Nucleare (INFN), Sezione di Roma*
[2] *Dipartimento di Fisica "E. R. Caianiello", Università di Salerno*
[3] *Istituto Nazionale di Fisica Nucleare (INFN), Gruppo Collegato di Salerno*

Abstract. In the framework of Horizon 2020, the European Commission approved the ASTERICS initiative (ASTronomy ESFRI and Research Infrastructure CluSter) to collect knowledge and experiences from astronomy, astrophysics and particle physics and foster synergies among existing research infrastructures and scientific communities, hence paving the way for future ones. ASTERICS aims at producing a common set of tools and strategies to be applied in Astronomy ESFRI facilities. In particular, it will target the so-called multi-messenger approach to combine information from optical and radio telescopes, photon counters and neutrino telescopes. pLISA is a software tool under development in ASTERICS to help and promote machine learning as a unified approach to multivariate analysis of astrophysical data and signals. The library will offer a collection of classification parameters, estimators, classes and methods to be linked and used in reconstruction programs (and possibly also extended), to characterize events in terms of particle identification and energy. The pLISA library aims at offering the software infrastructure for applications developed inside different experiments and has been designed with an effort to extrapolate general, physics-related estimators from the specific features of the data model related to each particular experiment. pLISA is oriented towards parallel computing architectures, with awareness of the opportunity of using GPUs as accelerators demanding specifically optimized algorithms and to reduce the costs of processing hardware requested for the reconstruction tasks. Indeed, a fast (ideally, real-time) reconstruction can open the way for the development or improvement of alert systems, typically required by multi-messenger search programmes among the different experimental facilities involved in ASTERICS.

1 Introduction

1.1 ASTERICS

ASTERICS (ASTronomy ESFRI and Research Infrastructure CluSter) [1] is a Research Infrastructure funded by the European Commission's Horizon 2020 framework. It collects experiences from astronomy, astrophysics and particle physics and aims at producing a common set of tools and strategies to be applied in the Astronomy ESFRI facilities[1], pushing for creating synergies among different

[a] e-mail: giulia.debonis@roma1.infn.it

[b] e-mail: cbozza@unisa.it

[1] See [2] for a description of the mission and the objectives of the European Strategy Forum on Research Infrastructures (ESFRI) and see [3] for a detailed report of the current roadmap.

experiments and scientific communities and for carrying out a multi-wavelength and multi-messenger approach in data analysis[2]. The ESFRI facilities in ASTERICS fully match the request for multi-wavelength and multi-messenger: SKA (radio) [5], E-ELT (infrared/optical) [6], CTA (gamma-rays) [7] and KM3NeT (neutrinos) [8]; complementary facilities (LOFAR, Euclid, VIRGO, LIGO, HESS, MAGIC, ANTARES, IceCube...) complete the framework.

1.2 OBELICS

Activities in ASTERICS are organised in five Working Packages (Figure 1-a). WP3 is named OBELICS (OBservatory E-environments Linked by common ChallengeS) and focuses on interoperability and software re-use for data generation, integration and analysis (Figure 1-b). Specific tasks of action, aimed at promoting multi-wavelength/multi-messenger data analyses, are the establishment of open standards and software libraries, the development of common solutions for data processing and extremely large databases, the study of advanced analysis algorithms and strategies.

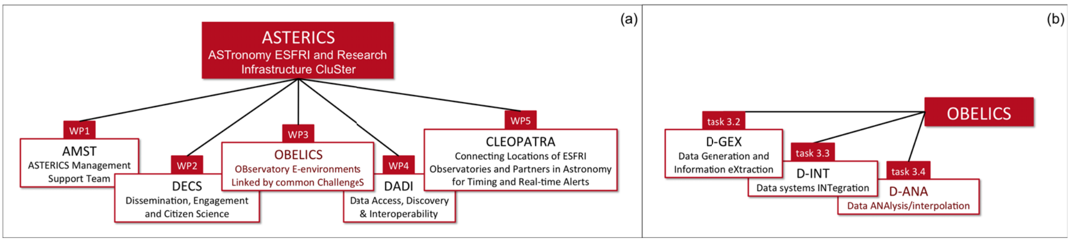
(a)
ASTERICS
ASTronomy ESFRI and Research Infrastructure CluSter
WP1
AMST
ASTERICS Management Support Team
WP2
DECS
Dissemination, Engagement and Citizen Science
WP3
OBELICS
OBservatory E-environments Linked by common ChallengeS
WP4
DADI
Data Access, Discovery & Interoperability
WP5
CLEOPATRA
Connecting Locations of ESFRI Observatories and Partners in Astronomy for Timing and Real-time Alerts
(b)
OBELICS
task 3.2
D-GEX
Data Generation and Information eXtraction
task 3.3
D-INT
Data systems INTegration
task 3.4
D-ANA
Data ANAlysis/interpolation

Figure 1. (a) ASTERICS and its Working Packages (WPs). (b) Sub-tasks in OBELICS.

2 pLISA

2.1 General Features

The pLISA project is inserted in sub-task 3.4 (D-ANA) and fulfils the mission and the objectives of ASTERICS and OBELICS, in particular for what concerns interoperability. The name of the project stands for *parallel Library for the Identification and Study of Astroparticles* and each word in this definition recalls important features of the ASTERICS/OBELICS initiative and distinctive elements of the project itself.

The term *parallel* refers to parallel programming and parallel computing architectures and reflects the pLISA plans of using Graphical Processing Units (GPUs) as computing accelerators, consistent with OBELICS about the development of new computing technologies. The acceleration feature is a key element when addressing the issue of real-time reconstruction, essential for the development of alert systems between experiments in a multi-messenger perspective. We have focused in particular on NVIDIA boards and CUDA [9] [10], and the implementations of the code have been devised as explicitly parallel, for running on GPUs since the very beginning.

The term *Library* qualifies pLISA as a toolbox, aiming at flexibility, interoperability, open standards and common solutions, independently of specific implementations operated in individual infrastructures, in order to adapt the code structure for the data analysis of a generic event-based experiment.

[2] See [4] for an example of the multi-messenger approach in neutrino astronomy.

That said, the study for pLISA has been initiated in the framework of KM3NeT and the peculiarities of KM3NeT (and ANTARES [11]) data taking, trigger systems and reconstruction algorithms have been taken into account and have been the starting point for the definition of the main features of pLISA.

The approach followed in pLISA for what concerns *Study and Identification* is through Multi-Variate Analysis (MVA) [12]: neural networks, boosted decision trees and similar machine-learning techniques have been investigated, in agreement with the claim of advanced algorithms and strategies professed by OBELICS; both regression and classification problems have been considered when working out the features of the library.

Finally, *Astroparticles* refers to the scientific target of the ASTERICS enterprise.

2.2 Implementation

The programming language used for pLISA is C++[3]; the current implementation is the "skeleton" of an header file. Classes and constants are very strongly typed, and extensive use of *namespaces* has been made, for minimising the chances of name clashes when pLISA is used alongside other libraries and for helping developers and users in producing bug-free code by enforcing type compliance. In addition, it is convenient to follow the structuring in namespaces to get an overview of the code (Figure 3). Classes are interfaces (i.e. purely abstract classes containing only pure virtual methods); furthermore, concerning the data storage pLISA puts requirements only on the information to be provided (*Features*, Figure 3) not on the way it is stored. This approach has been adopted to meet the objectives of ASTERICS and OBELICS for what concerns flexibility and interoperability, and also because GPU-based implementations require data structures in a format that differs from those in CPU-based implementations. Indeed, every experiment has its own data model and choosing one might lead to incompatibilities with others; with the solution under study in pLISA, non-implemented properties for data that are non-existing or meaningless in a specific dataset do not take memory/disk space; in addition the transient/persistent data model of user code need not to be changed, provided "reader" classes are produced by users. Optimised memory access (Figure 2) is obtained allowing that data in the GPU memory can be read just on-demand, and proper seamless caching mechanisms can be implemented (e.g. "lazy retrieval", i.e. retrieval of a full memory block only after a certain number of accesses are performed); moreover, memory transfer is optimised by adding flags that describe the internal encoding (namespace *Devices*, Figure 3).

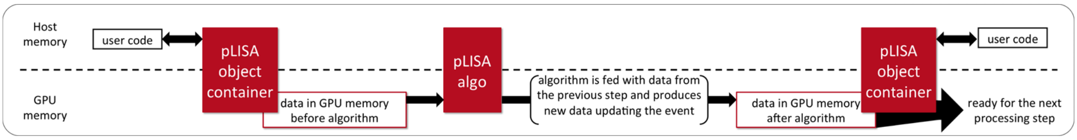
Host memory
user code
pLISA object container
pLISA algo
algorithm is fed with data from the previous step and produces new data updating the event
pLISA object container
user code
GPU memory
data in GPU memory before algorithm
data in GPU memory after algorithm
ready for the next processing step

Figure 2. pLISA operates a connection between the Host memory (CPU) and the Device memory (GPU). The user code resides in the CPU; the processing is completed in the GPU.

Acknowledgement

ASTERICS is a project supported by the European Commission Framework Programme Horizon 2020, Research and Innovation action, under grant agreement n. 653477.

[3] Successfully compiled with *Visual C++* and *g++* (GCC) 4.8.5.

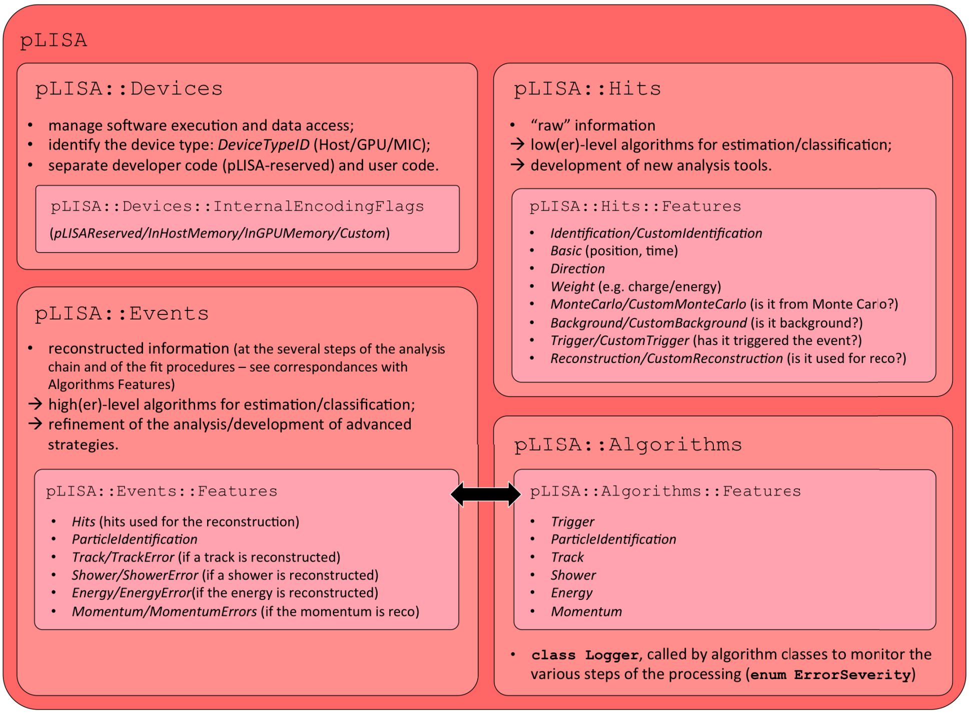
pLISA
pLISA::Devices
• manage software execution and data access;
• identify the device type: *DeviceTypeID* (Host/GPU/MIC);
• separate developer code (pLISA-reserved) and user code.
pLISA::Devices::InternalEncodingFlags
(*pLISAReserved/InHostMemory/InGPUMemory/Custom*)
pLISA::Hits
• "raw" information
→ low(er)-level algorithms for estimation/classification;
→ development of new analysis tools.
pLISA::Hits::Features
• *Identification/CustomIdentification*
• *Basic* (position, time)
• *Direction*
• *Weight* (e.g. charge/energy)
• *MonteCarlo/CustomMonteCarlo* (is it from Monte Carlo?)
• *Background/CustomBackground* (is it background?)
• *Trigger/CustomTrigger* (has it triggered the event?)
• *Reconstruction/CustomReconstruction* (is it used for reco?)
pLISA::Events
• reconstructed information (at the several steps of the analysis chain and of the fit procedures – see correspondances with Algorithms Features)
→ high(er)-level algorithms for estimation/classification;
→ refinement of the analysis/development of advanced strategies.
pLISA::Events::Features
• *Hits* (hits used for the reconstruction)
• *ParticleIdentification*
• *Track/TrackError* (if a track is reconstructed)
• *Shower/ShowerError* (if a shower is reconstructed)
• *Energy/EnergyError*(if the energy is reconstructed)
• *Momentum/MomentumErrors* (if the momentum is reco)
pLISA::Algorithms
pLISA::Algorithms::Features
• *Trigger*
• *ParticleIdentification*
• *Track*
• *Shower*
• *Energy*
• *Momentum*
• **class Logger**, called by algorithm classes to monitor the various steps of the processing (**enum ErrorSeverity**)

Figure 3. The structure of pLISA, in terms of namespaces. The outermost "container" is the namespace pLISA.

References

[1] ASTERICS Web Page, `https://www.asterics2020.eu`

[2] ESFRI Web Page, `http://ec.europa.eu/research/infrastructures/index_en.cfm?pg=esfri`

[3] *The ESFRI Roadmap 2016. Strategy Report on Research Infrastructures*, `http://ec.europa.eu/research/infrastructures/pdf/esfri/esfri_roadmap/esfri_roadmap_2016.pdf`

[4] De Bonis, G. (ANTARES), EPJ Web of Conferences **121**, 05014 (2016)

[5] SKA Web Page, `https://www.skatelescope.org`

[6] E-ELT Web Page, `http://www.eso.org/public/teles-instr/e-elt/`

[7] CTA Web Page, `https://portal.cta-observatory.org/Pages/Home.aspx`

[8] KM3NeT Web Page, `http://www.km3net.org`

[9] NVIDIA CUDA Web Page, `https://developer.nvidia.com/cuda-zone`

[10] J. Sanders, E. Kandrot, *CUDA by Examples.* (Addison-Wesley, 2011)

[11] ANTARES Web Page, `http://antares.in2p3.fr`

[12] T. Hastie, R. Tibshirani, J. Friedman, *The Elements of Statistical Learning.* (Springer, 2009)