Available online at www.sciencedirect.com
SCIENCE DIRECT®
Advances in Space Research 37 (2006) 1884–1888
ADVANCES IN SPACE RESEARCH (a COSPAR publication)
www.elsevier.com/locate/asr

YODA++: A proposal for a semi-automatic space mission control

M. Casolino, M.P. De Pascale, M. Nagni *, P. Picozza

INFN and University of Rome Tor Vergata, Department of Physics, Via della Ricerca Scientifica 1, 00133 Rome, Italy

Received 1 November 2004; received in revised form 10 June 2005; accepted 20 June 2005

Abstract

YODA++ is a proposal for a semi-automated data handling and analysis system for the PAMELA space experiment. The core of the routines have been developed to process a stream of raw data downlinked from the Resurs DK1 satellite (housing PAMELA) to the ground station in Moscow. Raw data consist of scientific data and are complemented by housekeeping information. Housekeeping information will be analyzed within a short time from download (1 h) in order to monitor the status of the experiment and to foreseen the mission acquisition planning. A prototype for the data visualization will run on an APACHE TOMCAT web application server, providing an off-line analysis tool using a browser and part of code for the system maintenance. Data retrieving development is in production phase, while a GUI interface for human friendly monitoring is on preliminary phase as well as a JavaServerPages/JavaServerFaces (JSP/JSF) web application facility. On a longer timescale (1–3 h from download) scientific data are analyzed. The data storage core will be a mix of CERNs ROOT files structure and MySQL as a relational database. YODA++ is currently being used in the integration and testing on ground of PAMELA data.

Keywords: Cosmic rays; Data analysis; Object oriented framework; Software architecture

1. Introduction

PAMELAs main objective is the accurate measurement of the antiproton and positron fluxes, with a sensitivity and statistics out of the reach of balloon-borne experiments. The energy range goes from below 100 MeV to above 100 GeV and the search of antihelium will have a sensitivity better than 10^{-7} in antihelium to helium ratio (Simon, 2003; Sparvoli et al., 2005). Also solar and heliospheric physics issues will be addressed (Casolino, 2005). Several detectors will identify the crossing particles: a Scintillator/time-of-flight system (Barbarino, 2005) for trigger and determination of absolute charge of the incident particle; a silicon tungsten calorimeter (Boezio, 2002) for energy measurement and adronic/electromagnetic shower discrimination; a silicon tracker (Adriani, 2003) inside a permanent magnet for momentum measurement; a bottom scintillator together with a neutron detector (Galper et al., 2001) for backscattering and neutrons measurements; an anticoincidence system (Pearce, 2003; Orsi et al., 2005) for background rejection. The PAMELA telescope will be installed on board of the Russian Resurs DK-1 satellite and will be launched in the year 2005. The expected data flow of the experiment will be 20 Gbyte/day, received in four different downlink sessions over two stations. Data quality and housekeeping information will be analyzed in a short time. The software running on the on-board CPU, based on a ERC-32 architecture, has been developed using a realtime operating system software (Straumann et al., 2001; Casolino et al., 2005). The housekeeping data acquisition was designed in order to have an efficient experiment control on ground too, to recognize and correct system's anomalies. Connected to the quick-look phase, another constraint is to share data through the collaboration as soon as possible in several formats. For these reasons we used CERNs ROOT framework (ROOT, 1996) to

* Corresponding author. Tel.: +39 6 725 94909.
E-mail address: nagni@roma2.infn.it (M. Nagni).

0273-1177/$30 © 2005 Published by Elsevier Ltd on behalf of COSPAR.
doi:10.1016/j.asr.2005.06.033

develop a custom analysis tool and to store data in a worldwide used file format; anyway, in order to meet another worldwide used standard data format, also the XML/XSL language is under review.

2. YODA environment: general scheme

YODA is the acronym for your own data analysis: it has grown over time from a collection of raw data unpack software to an environment fulfilling all the receiving/storage/processing operations of the experiment. Currently YODA++ comprises a hardware/software system designed to receive and store data allowing users to perform analysis at different levels with different tools. The implementation of each requirement on a different tool allows to retain an high flexibility reducing code refactoring (that is a software re-architecture due to unforeseen situations or because of the end of the prototyping phase) to the single software tool. All the tools are "glued" together using a high level scripting language (in our case Python), to define how each single tool cooperates with the other or how to manage the exceptions possibly raised by the tool. The operations performed by the tools of YODA++ include:

- statistics/analysis on raw data to determine quality of transmission;
- statistics/analysis on inner instrument packet data to determine instrument status;
- data storage in a relational database and in ROOT files;
- remote data access either through a web-application or a web-service.

Data have to pass a three-step process (Fig. 1) before the physics data can be used in the analysis; each of these steps is managed by a specific tool. In the first step the RawReader (RR) collects the data from the receiving station and processes them in order to estimate the transmission quality and to remove the headers introduced by the host satellite's transmission protocol. The first step produces intermediate files which are the input to the second step, that is the YodaReader (YR). The YR has to check the data stream to satisfy PAMELAs transmission protocol in order to extract the several packets types. The third and last step is the physics data analysis.

The first two steps of data processing are used to recognize peculiar situations not managed by PAMELA on-board software or to check response to previously transmitted commands from Earth, while the third step is mainly connected with long term analysis.

3. Satellite data collection: from satellite to ground

The expected amount of particle data from PAMELA is about 2 Gbyte/day while the expected background data from false triggers coming from secondary particles produced in the main body of the satellite can be up to 18 Gbyte. The information is stored in a memory device of the satellite Resurs-DK1 and can be transmitted to ground in portions in several downlink sessions.

The receiving antenna system TNA-7D has a parabolic reflector of 7-m diameter with azimuth-elevation fulcrum-rotating mechanism and has two frequency diverged radio channels. PAMELA data reception is

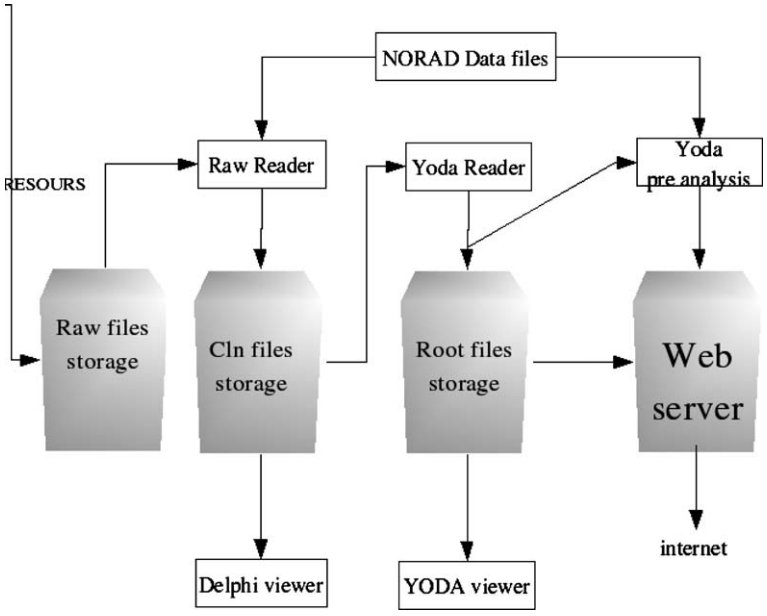

Fig. 1. (Left) The data flow from downlink to distribution. It is possible to see the three main processing steps and their dependencies.

performed at digital processing data system (DPDS), which is the component of Moscow Ground station at NTSOMZ. The information from DPDS enters the operational data set archive server that provides a secure connection between the receiving station and the ground segment of PAMELA where the first analysis takes place.

4. The YODA reader

For our specific needs the objects composing the YR tool have been designed on the ROOT libraries to generate a corresponding file structure. C++ was used as the main programming language.

In general, we have four types of objects:

- *Packet types objects*. These objects wrap both the specific data produced by the single devices (TrackerPacket, CalorimeterPacket, etc.) and those produced by the CPU controlled procedures (several telemetry packets).
- *Algorithm type objects*. While the packets could be considered a static structure, i.e., just like wrappers, these classes (tracker reader, calorimeter reader, several telemetries reader) are the dynamical counterpart which knows the rules to read and check each specific packet data from the raw ones (that is the files generated from the RawReader).
- *A class containing utility methods*. This class (PamelaRun) and the derived ones (SpecificModelPamelaRun) represent an interface between YR and the ROOT objects and structures (TFiles and TTrees);
- *An additional facade class*. Inside it is defined a command line user interface and basically is where the program starts.

The packet types are structured around a protocol wrapper called *Header* which is meant to define general characteristic of the specific data it belongs to.

Each packet uses its own specific reader and each reader inherits from a common parent: the *Algorithm* object.

We want to stress how the flexibility of this approach allowed to insert in the reader classes some FORTRAN routines; at the same time specific interfaces are in a developing phase to allow the same group to read ROOT files inside their FORTRAN programs.

A facade class (Gamma, 1995) is the higher level class of the YR and acts as an interface with the user command line or the graphical user interface (GUI) whenever it should be available; at the same time, if a batch process is needed, the operations performed by this class could be easily inserted into a Python script external to the YR.

Summarizing, the code can be represented as an action of the EventReader searching for the Header packet signature inside a stream of data. The stream is then passed to the HeaderReader class which checks if the various parameters defined inside the actual stream are coherent with the header parameters (PacketLength, Counter, Timing, CRC, etc.). If all the constraints on the data format are verified, the EventReader class will call, for each different kind of data generated by PAMELA, a specific Reader according to an identification code in the header as previously mapped. Finally, through the PamelaRun class, the EventReader will store the read packet and the relative header inside a long term data architecture composed both by ROOT files and a relational database structure.

5. The YODA viewer and the YODA data manager

Once the data are extracted, it can be managed using the ROOT framework either by direct interaction with the ROOT files or using specific scripts taking advantage of the CINTerpreter (CINT). For example, collecting all the header files generated once the raw file has been processed, it is easy to make a first check on data quality (Fig. 2).

The collection of all these scripts represents the so called *YODA Viewer*, which can be a stand-alone GUI[1] and/or a web application, extracting graphs from the same application serving the internet users; the main reasons for this are both to reduce the number of code lines written and to reuse the already available code. The YODA viewer shows the most meaningful information extracted from downlinked data. This can be done using ROOT scripts generating either graph/histogram or XML files.

Referring to the XML files, showing HouseKeeping and information, it is worth to note how they can be easily put on a browser in a human readable form, processing them through a specific XSL file. Applying an XSL file on a XML does not reduce the information contained in it but just formats it in a different way, combining it eventually with other XML files. The process can be performed either by the browser itself (having access to both the XML/XSL repository) or inside a web application. Even if in a preliminary phase, an on-demand web application is being developed running on a TOMCAT server (written using JSP/JSF specifications), in order to allow the user to have a data quick-look using just a computer connected to the internet and a browser without any need for additional software.

[1] A prototype has already been implemented using the Signal-Slot features of the ROOT GUI libraries on a Model-View-Controller. This architecture decouples how the data are handled inside the program from how the data are shown to the final user. This approach is extremely important when the data analysis logic is shared between several laboratory but the graphical interface is not.

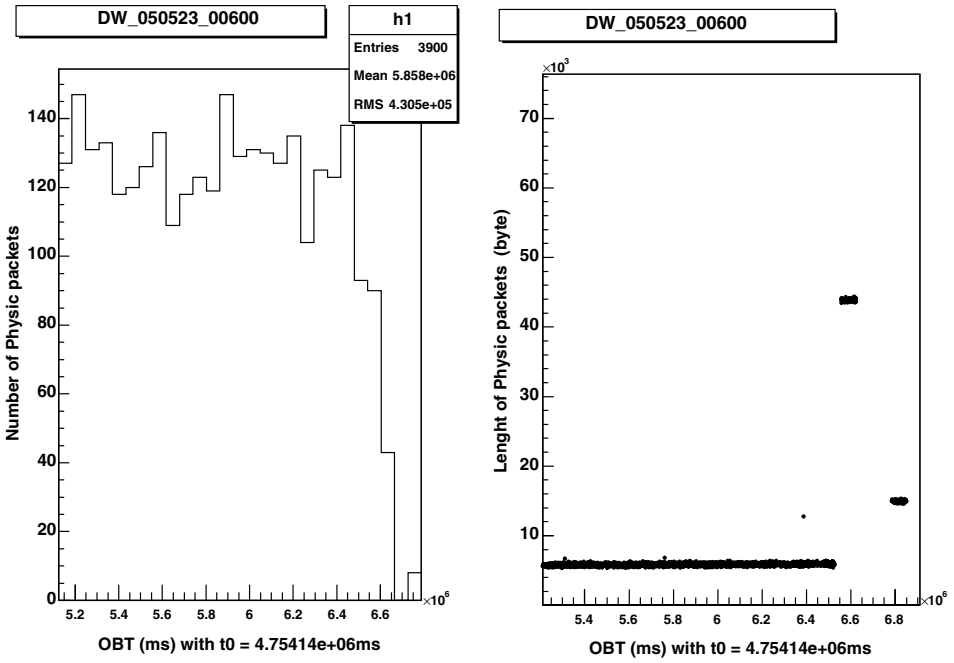

Fig. 2. (Left) The number of PhysicsEvent vs. OnBoardTime. (Right) The PhysicsEvent packet size vs. OnBoardTime. It is clear that the higher the rate, the smaller the physics data packet size. The physics packet nominal size should be below some tens of KB because of the data compression embedded in the detectors hardware. A bigger size could be a warning for a detector hardware malfunctioning; however, it could be due to physics phenomena (particles shower), or to some specific operations (calibrations) as well.

We already mentioned that the highest level of the YODA++ is composed by a collection of Python scripts; this collection constitutes the so called *YODA data manager*. This collection of scripts is dedicated to synchronize actions of all the described tools. For example one script is dedicated to monitoring a specific folder, detecting new files and starting the YR; another script is dedicated to automatically processing a list of defined ROOT scripts and executing all the YODA viewer's scripts collection. The Python language is an interpreted language and does not have performances comparable with the compiled C++. A high level manager allows to rapidly modify the YODA++ architecture because the single tools are connected in a light way, so that each tool remains untouched but the data workflow can be modified independently.

6. Conclusion

The YODA architecture, born as a monolithic software project, has developed in a complex and flexible architecture using a mix of several independent softwares. A wise development allowed to use the best language for each goal (C++, Java, FORTRAN, XML, XSL, Python, SQL, etc.). This structure has proved to be flexible enough to be integrated into a completely new external computing factory in a fast time. Future features in the YODA++ will include high computing performances for physics analysis, using long term storage, but thanks to the acquired flexibility it will be possible to focus more on the algorithm analysis rather than on how integrate it the existing framework.

References

Adriani, O. et al. The magnetic spectrometer of the PAMELA satellite experiment. Nucl. Instr. Meth. Phys. Res. A 511, 72, 2003.

Barbarino, G.C. et al. The PAMELA time-of-flight system: status report. Nucl. Phys. (Proc. Suppl.) B 125, 298, 2005.

Boezio, M. et al. A high granularity imaging calorimeter for cosmic-ray physics. Nucl. Instr. Meth. Phys. Res. A 487, 407, 2002.

Casolino, M. et al. The PAMELA CPU, These proceedings, 2005.

Casolino, M. Cosmic ray observation of the heliosphere with the PAMELA experiment, These proceedings, 2005.

Galper, A.M. et al. Measurements of primary protons and electrons in energy range 10^{11}–10^{13} eV in the PAMELA experiment, in: Proceedings of the 27th ICRC, Hamburg, OG 2219, 2001.

Gamma, E. et al. Design Pattern, first ed Addison-Wesley Professional, Reading, MA, 1995.

Orsi, S. et al. These proceedings, 2005.

Pearce, M. The anticounter system of the PAMELA space experiment, in: Proceedings of the 28th ICRC July 31–August 7, 2003 Tsukuba, Japan, OG 1.5, 2125, 2003.

ROOT – An Object Oriented Data Analysis Framework, in: Proceedings of the AIHENP'96 Workshop, Lausanne, September, 1996. Nucl. Inst. Meth. Phys. Res. A 389 (1997) 81–86. Available from: <http://root.cern.ch/>.

Simon, M. XXVIII ICRC, OG 1.5, 2117 Tsukuba, Japan, 2003.
Sparvoli, R. et al. Space qualification tests of the PAMELA instrument, 2005.
Straumann, T. Open source real-time operating systems overview, in: Eighth International Conference on Accelerator and Large Experimental Physics Control Systems, San Jose, USA, 2001.