Offline software for the DAMPE experiment

Chi Wang(王驰)\(^{1,2,1}\) \quad Dong Liu(刘栋)\(^{1,2}\) \quad Yifeng Wei(魏逸丰)\(^{1,2,2}\) \quad Zhiyong Zhang(张志永)\(^{1,2}\)
Yunlong Zhang(张云龙)\(^{1,2}\) \quad Xiaoliao Wang(汪晓莲)\(^{1,2}\) \quad Zizong Xu(许宗宗)\(^{1,2}\) \quad Guangshun Huang(黄广顺)\(^{1,2,3}\)
Andrii Tykhonov\(^3\) \quad Xin Wu(吴欣)\(^3\) \quad Jingjing Zang(藏京京)\(^4\) \quad Yang Liu(刘杨)\(^4\) \quad Wei Jiang(蒋维)\(^4\)
Sicheng Wen(文思成)\(^4\) \quad Jian Wu(伍健)\(^4\) \quad Jin Chang(常进)\(^4\)

\(^{1}\)State Key Laboratory of Particle Detection and Electronics, University of Science and Technology of China, Hefei 230026, China
\(^{2}\)Department of Modern Physics, University of Science and Technology of China, Hefei 230026, China
\(^{3}\)DPNC, Universite de Geneve, CH-1211 Genève 4, Switzerland
\(^{4}\)Purple Mountain Observatory, Chinese Academy of Sciences, Nanjing 210008, China

Abstract: A software system has been developed for the DArk Matter Particle Explorer (DAMPE) mission, a satellite-based experiment. The DAMPE software is mainly written in C++ and steered using a Python script. This article presents an overview of the DAMPE offline software, including the major architecture design and specific implementation for simulation, calibration and reconstruction. The whole system has been successfully applied to DAMPE data analysis. Some results obtained using the system, from simulation and beam test experiments, are presented.

Keywords: software architecture, DAMPE, abstract interface, C++

PACS: 29.50.+v, 29.85.-c, 29.85.Fj
DOI: 10.1088/1674-1137/41/10/106201

1 Introduction

The DArk Matter Particle Explorer (DAMPE) is a satellite-borne detector for detection of high energy cosmic rays [1]. It was launched on 17 December 2015 [2] into an elliptical and sun-synchronous orbit at an altitude of 500 km. Its mission is foreseen to last for at least three years. The primary scientific objective of DAMPE is to indirectly search for dark matter (DM) by measuring the spectra of photons, electrons, and positrons, ranging from 5 GeV to 10 TeV, with high energy resolution (<1.5% at 800 GeV).

The DAMPE detector is composed of four subdetectors, as shown in Fig. 1. The top is a double-layer plastic scintillator strip detector (PSD), which serves as anticoincidence for gamma detection and charge measurement for ions. The PSD is followed by a silicon-tungsten tracker (STK), which is made of six silicon micro-strip layers with three 1.0-mm-thick tungsten plates inserted in front of tracking layers 2, 3, and 4. The BGO electromagnetic calorimeter (BGO ECAL) in the middle is composed of 308 BGO crystal bars with dimensions of 2.5 \times 2.5 \times 60.0 \text{ cm}, which is \sim 31 radiation lengths, making it the deepest calorimeter used in space experiments so far. At the bottom, a neutron detector (NUD) layer is added to improve the electron/proton separation capacity.

Fig. 1. (color online) Structure of the DAMPE detector.
Optimum operation of DAMPE requires an immense amount of hardware development effort, and also, to achieve the final physics goals, it is essential to develop offline software that is suitable for the special experimental data pipe.

This article first gives a review of the general requirements for the DAMPE software. Then, the basic software architecture is described and its implementation is introduced. Finally, some applications for DAMPE orbit calibration and beam test analysis are also presented.

2 Requirements

The main features of high-energy physics (HEP) experiments, such as those at the Large Hadron Collider (LHC), are the complexity of the detector system and the wide geographical distribution of participants. Given the efficiency of worldwide communication among physicists doing analysis work, non-uniform programming logic from individual developers is inadequate in group collaboration. Besides, experiments are expected to run many years and therefore changes in analysis technologies must be envisaged. Thus it is necessary to develop flexible software that can withstand transitions, be capable of integrating efforts from individual people smoothly, and be easily maintained over the long timescale of the project.

The idea of using uniform software is generally accepted in HEP experiments. GAUDI [3] is a well-designed framework driven by LHCb [4]. It has been adopted in many other projects, e.g., ATLAS Athena [5], HARP Gaudino [6], and BESIII BOSS [7]. SNiPER [8] is another analysis software framework designed for noncollider experiments; it is being used by the Jiangmen Underground Neutron Observatory (JUNO) [9] and the Large High Altitude Air Shower Observatory (LHAASO) [10].

The Alpha Magnetic Spectrometer (AMS-02) is a high-energy particle detector designed to study the origin and nature of cosmic rays up to a few TeV in space [11]. It does not use an analysis framework like GAUDI, while the basic component for analysis is a series of specified event classes that are inherited from ROOT’s TObject. The Fermi Large Area Telescope (Fermi-LAT), the primary instrument on the Fermi Gamma-ray Space Telescope (Fermi) mission, is an imaging, wide-field-of-view, high-energy γ-ray telescope, covering the energy range from 20 MeV to >300 GeV. The data analysis software of Fermi-LAT has been designed within the framework of the HEADAS FTOOLS methodology, to ensure cross-mission compatibilities wherever possible and to minimize the learning curve for users of other high-energy astrophysics mission data sets.

Since DAMPE consists of several subdetectors, it is not easy to combine nonuniform reconstruction and calibration codes from them. In the collaboration, many participants are astronomers, who are not so familiar with the computing mode used by particle physicists. The most important thing is that we have to provide DAMPE with suitable analysis tools for worldwide researchers. With these considerations, we started to design the DAMPE offline data analysis software system with an architecture that could be used to organize development activities.

With the experience of data analysis in HEP experiments, the first choice should be GAUDI. However, after 15 years of development, GAUDI has evolved into a bulky system and most of its complex functionalities are not necessary for DAMPE. Adapting GAUDI for DAMPE is not a practical solution, given the time of adaptation and the cost of maintenance in the future as well. Using SNiPER is also not practical as it was in the design and development stage at the beginning of the DAMPE project. Thus, under the condition of limited manpower and time, we started to construct new lightweight software for DAMPE.

The key challenge is how to provide a uniform development environment for programmers and a simple, flexible manipulation manner for the end users. Such a requirement is not only helpful for software development and maintenance but also significant for improving physics analysis quality and efficiency. The overall design of the DAMPE offline software takes into account the following principles:

1) Separate environments for developer and user: One person may have different roles, as a software developer or a high-level user, at different times. This requires multiple computing environments. During development, developers should be able to substitute their own versions of certain program modules while still using the standard version for the rest. After the development task is done, the new module should be easily integrated into the software system and become a new standard version.

2) Data separation: Analysis work is a data-driven application, so the overall design should distinguish data objects from algorithm objects. In general, methods used to process data objects of some types and produce new data objects of other types have a much higher possibility of being modified than the data objects. Data and algorithms should be decoupled into two modules in the software. Besides, all different types of data should be accessible in an uniform manner and selected dynamically.

3) Ease of use: The software system must be easily used by collaboration physicists who are not computing experts and cannot devote a lot of time to learning computing techniques. Two aspects must be addressed. For developers, physicists should provide a series of
algorithms with different analysis techniques while the
software should help them focus on implementation of
the core functionality without being disturbed by gen-
eral minor issues, such as data communication among
modules. For end users, the software should be flexi-
bly enough to arrange the analysis chain, and adding
or replacing certain modules should be possible by just
changing a few lines in a steering file at run time.

Based on the above, C++ has been identified and
adopted to build the main part of the software. By taking
advantage of Boost Python [12], the system can manipu-
late a specific task in a Python script [13] in an extremely
easy and flexible manner. CINT [14] is an interpreter for
C++ code and widely used in HEP analysis, and our
software also has the capability of running in ROOT [15]
as a plug-in module.

3 Basic architecture

The basis of the software is a pre-defined architec-
ture that defines the fundamental components and the
interface. The fundamental components of the software
comprise five parts:

1) **Kernel**, which is the steering part of the whole
offline software.

2) **Event module**, which contains all reconstruc-
tion information based on the ROOT input/output (I/O)
stream, as well as detector simulation and digitization
information.

3) **Algorithms**, which are the central components of
the event data processing.

4) **Event filters**, which are used to reduce the input
events into concrete algorithms.

5) **Services**, which aim at providing common func-
tionality needed by the algorithms.

The architecture of the software is shown as the ob-
ject diagram in Fig. 2.

3.1 Kernel

The kernel is the specification of a piece of software,
and the DAMPE software kernel is composed of three
parts: DmpCore, DmpServerManager, and DmpAlgo-
RithmManager. The main function of DmpCore is to
drive a job, including initialization, event loop and final-
ization.

As shown in Fig. 2, a job is a singleton of the top-level
manager (DmpCore), which manipulates the algorithm
manager and service manager. The software provides a
data communication mechanism and execution logic for
applications. Meanwhile, it transfers the arrangement
and deploy right of concrete algorithms to users at run
time via a Python script. With a predefined interface, it
is possible to replace one concrete component with an-
other, while others remain unchanged.

3.2 Event module

The event module holds the description of the event
data items that are exchanged between algorithms in
an application. It is the central part of an experiment.
The production flow of event data represents the analysis
logic. Mostly, the developer does not have the full picture
of the event type for the next stage, so expansibility of
the event data module is required. Data taken on board
by DAMPE consist of raw scientific events, which are
directly read out from subdetectors, and satellite status
data, such as temperatures and position and velocity of
the satellite, which are periodically sampled with different
time scales. Different data are described with corre-
sponding event classes. The first step of data processing
is to convert raw binary orbit data into raw root format
data. Then, the reconstruction algorithm is applied to
create more physically meaningful data, for an example
of BGO ECAL, from DmpEvtBgoRaw (analog to digi-
tal convertor (ADC) counts of the photomultiplier tube
(PMT)) to DmpEvtBgoHits (energy of each BGO bar).
The DmpEvtBgoHits class is also used to record the out-
put Monte Carlo information of the BGO subdetector.
Figure 3 shows the event classes for raw BGO calorime-
ter data containing dynode IDs [16], see the minimum
detection unit shown in Fig. 9, and ADC counts

![Fig. 2. (color online) DAMPE software object diagram.](image-url)

![Fig. 3. Event class content for raw data and hits](image-url)

Raw data

- **GlobalDynodeID**
- **ADC**
- **Calibration**
- **Hits**
- **GlobalBarID**
- **Energy**
- **EnergyS0**
- **EnergyS1**

Data contains ADC information and Hits contains energy
deposited in each BGO bar, with “EnergyS0” and
“EnergyS1” representing energy reconstructed on
sides 0 and 1, respectively, of the bar.
of PMTs, which are converted to hits data containing BGO bar IDs and the energy deposited in each bar.

In the event module, all event classes inherit from TObject of ROOT. Therefore, they naturally adopt the ROOT I/O stream and enable us to implement the data management mechanism by using the mediator pattern [17].

3.3 Algorithms

The essence of the event data processing applications lies in the physics algorithms. Concrete algorithm classes must inherit from the base algorithm class (DmpVAig), which is provided by the software, to manipulate algorithms via the generic interfaces without knowing what they really do.

The core functionality of an algorithm is in producing event data one by one. This entails implementing a certain function DmpVAlg::ProcessThisEvent() under the architecture of the DAMPE offline software. The remaining small tasks, for instance preparing a new event or communicating with each other, are already implemented in the software. The user should focus on extracting useful information from the data that is used in a new algorithm.

The algorithms are core parts of applications and all components are reusable; for example, the tracking algorithm can be used for alignment and acceptance analysis. If a series of algorithms forms an application, the correct arrangement order is required to archive the newest data from each preceding algorithm in a specified sequence.

3.4 Event filters

Most of the time, a concrete algorithm only accepts events that could pass a set of conditions, and some algorithms share some common requirements. To make the code clearer and simpler, we designed another module, composed of event filters that are decoupled from both algorithms and event classes. The provided application programming interface (API) is DmpVFilter, and the essential part of an event filter is a Boolean function DmpVFilter::Pass(), which is only responsible for checking properties of a certain type of input event. If the event does not suit those conditions, it is discarded before invoking the core analysis part of the algorithm. The execution logic is shown in Fig. 4.

This design is extremely useful for DAMPE data analysis. For instance, most subdetector calibration algorithms have to discard events that come from the South Atlantic Anomaly (SAA), and therefore the orbit filter (DmpFilterOrbit) is developed. The input event class of the orbit filter is the event header that contains the time stamp of each event. By checking the gradient of the trigger rate at different times, we can distinguish whether or not the event is in the SAA. Besides, sometimes we need to compare analysis results in different sections of latitude and longitude, which is also provided by the orbit filter. All event filters can be constructed and inserted into any algorithm at run time dynamically, which makes DAMPE orbit data analysis more flexible and effective.

3.5 Services

The software involves many frequently used functions, which are implemented as services. The difference between a service and an algorithm is whether it reads or analyzes event data. Services are widely used by algorithms; e.g., an I/O service provides I/O support and an Extensible Markup Language (XML) service records all configuration values once a job starts.

It is foreseen that more and more physicists will join the data analysis work of DAMPE in the future, so we designed the system with plug-in mechanisms, and the advantages of a modularized design will become more and more evident.

4 Software implementation

4.1 Simulation

The Monte Carlo (MC) simulation includes the detector description, detector response to cosmic rays, and signal digitization. The materials and geometry needed for the detector are read from Geometry Description Makeup Language (GDML) files, which are translated from AutoCAD files during the engineering design process. The constructed detector is shown in Fig. 5.

The response of the detector to cosmic rays is handled by Geant4 [18], a toolkit widely used in high-energy physics experiments to manage particle generation, propagation, and interactions. The whole simulation procedure is implemented using this software, which produces collections of energy hits for each sensitive detector element. Compared to the common Geant4 application, the difference is that we do not build the simulation package as an executable file. We split the Geant4 Run Manager into three parts to satisfy the algorithm interface to let the software drive it. All Geant4 user actions and detector constructions are implemented in DmpVAig::Initialize(). Since the event loop is already

![Fig. 4. (color online) Execution logic of the algorithm with an event filter.](image-url)
provided by the software, DmpVAlg::ProcessThisEvent() just invokes the functionality of one event execution of the Geant4 Run Manager. Then the resource is released in DmpVAlg::Finalize().

Level 0: binary data transmitted from the satellite, \(~12\) GB per day.

Level 1: 13 kinds of completed telemetry source packages, one for science data and 12 for housekeeping data, produced from level 0 data with operations such as data merging, overlap skipping and cyclic redundancy checking, etc.

Level 2: science data product reconstructed with the DAMPE offline software.

The data processing discussed here is for the period of level 1 translating to level 2. The processing pipeline of the DAMPE flight data includes raw data conversion (RDC) and calibration and reconstruction algorithms implemented in the DAMPE software. The RDC algorithm splits the level 1 science data into calibration files and observation files, and it then converts them into ROOT data files. Calibration files are used to extract calibration constants, which are used in the simulation and reconstruction algorithms. The reconstruction algorithm is used to reconstruct the ADC counts of each sub-detector to physical variables, such as energy and charge.

4.2.1 Calibration

The DAMPE software calibration package consists of calibration algorithms for the four subdetectors, which are independent of each other. Because of the modularized design of the software architecture, many common functions are extracted and packaged into algorithm libraries. All algorithms are created and appended to the algorithm manager one by one, which indicates the sequence of execution. A Python steering file starts the whole job. Here we present a calibration procedure to illustrate a concrete job running under the DAMPE software architecture. The job is to calibrate the absolute energy scale for the BGO ECAL with minimum ionizing particles (MIPs); the execution logic is shown in Fig. 7. There are two filters inserted into the first algorithm: the trigger logical filter and the orbit filter, which are used to check the trigger logical tag and discard events in the SAA, respectively. After subtracting pedestal noise in the first algorithm, the event data (DmpEvtBgoRaw) is updated and then transferred to the energy reconstruction algorithm, where the primary energy reconstruction is performed by using MIPs calibration constants and the ADC value of dynode 8 of the PMT, to generate new event data (DmpEvtBgoHits). Then, the central filter, the MIPs selection filter, of the job starts to work based on the energy of each BGO bar. In the end, the MIPs spectrum is obtained and MIPs peak constants are calculated in the third algorithm.

All calibration constants are stored as ASCII files in the specified format. To manage these parameters easily, we developed a service, called DmpParHolder, to write and read calibration constants. Through the data timestamp, the DmpParHolder can invoke the closest calibra-
tion parameters automatically in the period of reconstruction.

4.2.2 Reconstruction

Data reconstruction is one of the central tasks of the offline data processing. For the DAMPE experiment, a complete chain of reconstruction algorithms has been developed and fitted into the software architecture. The reconstruction package contains the tracking reconstruction algorithm (BGO+STK), charge reconstruction algorithms (PSD+STK), energy reconstruction algorithm (BGO), etc.

The key to event reconstruction for the DAMPE experiment is energy and tracking reconstruction. The most powerful instrument for tracking is the STK detector. Generally, the STK track reconstruction algorithm gives a series of track candidates with different qualities. To obtain a high-precision track, when the directly reconstructed one obtained by the BGO ECAL is available, another global track finder algorithm combines the STK and BGO results and gives a global track. Meanwhile, since the scintillation light attenuates in the long BGO crystal (of 60 cm(157,92),(446,139), the attenuation correction is applied in energy reconstruction, and its algorithm depends on the result of the tracking reconstruction. Therefore, we reconstruct a raw energy and global track first (version 0) and then an iterative reconstruction is implemented. The whole workflow is shown in Fig. 8.

4.2.3 Housekeeping data

For a satellite experiment, many engineering data, such as orbit latitude and longitude, are also essential in physics analysis. This specific information is called housekeeping data (HKD) in the DAMPE experiment. The HKD packages are parsed from the satellite and then inserted into the housekeeping database after being processed by the ground support system (GSS). For the convenience of connecting HKD with physical analysis, an HKD service was developed in the DAMPE software. During the period of raw data conversion of science data, key HKD parameters needed in the analysis are stored into the ROOT files of science data through the interface provided by the HKD service. Users are also able to get these parameters from data with the HKD service.

5 Application

The DAMPE software has been verified and improved during ground tests, including long-time cosmic ray tests, thermal vacuum experiments, and beam tests at CERN. We present some application results here.

The BGO ECAL comprises 14 layers of BGO crystals, each layer composed of 22 BGO bars. One BGO bar is coupled with two PMTs on its both ends as shown in Fig. 9. To extend the dynamic range, the signals are read out from three dynodes (Dy2, Dy5 and Dy8) of the PMT.

The primary purpose of the BGO ECAL is to precisely measure the energy of incident particles. Since the energy scale obtained by the MIPs calibration is only available for Dy8, we perform the calibration of the gains of Dy8/Dy5 and Dy5/Dy2 before the energy reconstruction. For one event, we calculate a valid ADC value of...
Dy8 and transform it to energy by multiplying by the corresponding energy scale. When the readout ADC of Dy8 overflows, the valid ADC of Dy8 is calculated from the lower dynode and the calibration constants of the gains.

In the simulation, the BGO crystals are defined as the sensitive detector for the BGO ECAL. We record the total energy of each bar and simulate the light attenuation of BGO crystals via a parametric method using light attenuation calibration results [19]. The output data format of the BGO ECAL is defined by the DmEvtBgoHits class.

The reconstruction and simulation algorithms have been used for beam test data analysis. Figures 10 and 11 show the energy linearity and energy resolution, respectively, of the DAMPE BGO calorimeter, from the electron beam test at CERN in 2014. Both simulation and beam test data indicate that the energy resolution is <0.8% at 243 GeV, which is better than that of the design objective. More details about the beam test results can be found in Ref. [19].

The good agreement between the simulation and real data confirms not only the validity of the DAMPE analysis software system but also the reliability of the DAMPE detector.

6 Summary

The software for the DAMPE experiment is user friendly, flexible and stable. It uses state-of-the-art computer science techniques and satisfies most needs of
DAMPE offline analysis. With the help of the architecture, collaboration-wide development work can be established in a reliable and extensive manner. The system has been widely exercised during long-term cosmic tests and three beam test experiments, and has now been successfully applied in orbit data analysis. So far, the physics results have proved the effectiveness and robustness of the software.

DAMPE is a pioneer of scientific satellite experiments in China, providing valuable experience for other experiments. The design principles of the software developed for DAMPE presented in this paper can also be a useful guideline. The kernel and some classes with virtual functions can be reused to improve the development progress of software for new and similar experiments.

The authors wish to thank Prof. Weidong Li and Dr. Tao Lin from the Institute of High Energy Physics, Chinese Academy of Sciences, for providing many beneficial suggestions for the software architecture.

References

1. J. Chang et al, Astroparticle Physics, 95: 6–24 (2017)
2. E. Gibney et al, Nature, 528: 443–444 (2015)
3. G. Barrand et al, Computer Physics Communications, 140: 45–55 (2001)
4. LHCB Collaboration. JINST, 3: S08005 (2008)
5. ATLAS collaboration. ATLAS Computing Technical Design Report: 27 (2005)
6. Catanesi M. G et al, Nucl. Instrum. Meth. A, 571 (3): 527–561 (2007)
7. W. D. Li et al, Proceeding of CHEP. Vol. 27, 2006
8. J.H. Zou et al, In Journal of Physics: Conference Series, 664(7): 072053 IOP Publishing, 2015
9. Y. F. Wang, PoS Neutel2013: 030 (2013)
10. Zhen Cao. Chin. Phys. C, 34(2): 249 (2010)
11. Battiston, Roberto. Nucl. Instrum. Meth. A, 588(1): 227–234 (2008)
12. See http://www.boost.org/doc/libs/1.60.0/libs/python/doc/html/index.html
13. See https://www.python.org
14. See https://root.cern.ch/cint
15. Brun Rene, and Fons Rademakers, Nucl. Instrum. Meth. A, 389(1): 81–86 (1997)
16. Y. L. Zhang et al, Chin. Phys. C, 36(1): 71–73 (2012)
17. see https://en.wikipedia.org/wiki/Mediator_pattern
18. Agostinelli, Sea et al, Nucl. Instrum. Meth. A, 506(3): 250–303 (2003)
19. Z. Y. Zhang et al, Nucl. Instrum. Meth. A, 836: 98–104 (2016)