Data management and processing system for the Baikal-GVD telescope

A.D. Avrorin[a], A.V. Avrorin[a], V.M. Aynutdinov[a], R. Bannash[g], I.A. Belolaptikov[b], V.B. Brudanin[b], N.M. Budnev[c], I.A. Danilchenko[a], G.V. Domogatsky[a], A.A. Doroshenko[a], R. Dvornický[b,h], A.N. Dyachok[c], Zh.-A.M. Dzhilkibaev[a], L. Fajt[b,h,i], S.V. Fialkovsky[e], A.R. Gafarov[c], K.V. Golubkov[a], T.I. Gress[c], Z. Hons[b], K.G. Kebkal[g], O.G. Kebkal[g], M.M. Kolbin[b], K.V. Konischev[b], A.V. Korobchenko[b], A.P. Koshechkin[a], F.K. Koshel[a], A.V. Kozhin[d], V.F. Kulepov[e], D.A. Kuleshov[a], M.B. Milenin[e], R.A. Mirgazov[c], E.R. Osipova[d], A.I. Panfilov[a], L.V. Pan'kov[c], D.P. Petukhov[a], E.N. Pliskovsky[b], M.I. Rozanov[f], E.V. Rjabov[c], G.B. Safronov[b], B.A. Shaybonov*[b], M.D. Shelepov[a], F. Šimkovic[b,h,i], A.V. Skurikhin[d], I. Štekl[i], O.V. Suvorova[a], V.A. Tabolenko[c], B.A. Tarashansky[c], S.A. Yakovlev[g], A.V. Zagorodnikov[c] and V.L. Zurbanov[c]

[a]*Institute for Nuclear Research, Moscow, 117312 Russia*
[b]*Joint Institute for Nuclear Research, Dubna, 141980 Russia*
[c]*Irkutsk State University, Irkutsk, 664003 Russia*
[d]*Institute of Nuclear Physics, Moscow State University, Moscow, 119991 Russia*
[e]*Nizhni Novgorod State Technical University, Nizhni Novgorod, 603950 Russia*
[f]*St. Petersburg State Marine Technical University, St. Petersburg, 190008 Russia*
[g]*EvoLogics, Germany*
[h]*Comenius University, Bratislava, Slovakia*
[i]*Czech Technical University in Prague, Prague, Czech Republic*
E-mail: bairsh@yandex.ru

The Baikal-GVD neutrino telescope is being constructed in Lake Baikal. The robust Baikal Analysis and Reconstruction Software (BARS) has been developed to convert raw data into physics results. To provide a stable and continuous analysis of all the data, an automatic data processing using a database is developed. The flexibility of the concept makes it easy to add new steps at any point of the analysis chain. The software is a ROOT-based collection of C++ classes driven by a central event loop. Now the basic algorithms have been implemented in the system such as event building, signal extraction, background rejection and simple reconstruction.

35th International Cosmic Ray Conference — ICRC2017
10–20 July, 2017
Bexco, Busan, Korea

*Speaker.

1. Introduction

Two out of eight clusters of strings of the Baikal-GVD underwater neutrino telescope have been installed in Lake Baikal and are currently taking data [1]. GVD will be a multifunctional device with the purpose to measure high-energy cosmic neutrinos with high angular precision and with high efficiency for astrophysical sources located in the Southern Hemisphere. For that purpose, a new dedicated software called BARS (Baikal Analysis and Reconstruction Software) has been developed in the Baikal collaboration. It is written in C++ and based on the widely-used ROOT package [2] and uses core of the MARS framework [3]. It provides a robust and flexible platform for dealing with the data from the telescope. A possibility to add a new analysis code in standard manner is integrated in the system. It already has a variety of tools to perform the primary data processing and analysis. Below we introduce the general approach of the system, briefly review steps of primary data processing, and describe the automation of the system.

2. General Approach

Data processing and analysis are realized as a set of C++ programs that solve some concrete task. Generally basic elements of the programs are blocks of data and algorithms that perform some manipulations with the data. Algorithms are encapsulated into modules that have the same structure which allows to assemble and sequence them to provide specific task. The blocks of data are encapsulated into data containers through which modules can relay data to each another. Thus data and algorithms are separated from each other that benefits of very flexible basic structure of data processing. Each module may have three abstract methods implemented: *PreProcess*, which is called at the start of execution; *Process*, which is called once per entry; *PostProcess*, which is called at the end of the execution. It is not mandatory for a module to have all these methods implemented. Modules are arranged in a specific module list, data containers are collected in a specific data container list (Fig. 1). At the start of a program the module list is filled by modules in their execution order. The data container list is initialized with no entries. After that *PreProcess* methods are called for each module in the list. At this stage modules usually create an output and find the necessary input data containers in the data container list, create some histograms etc. In the *Process* stage the modules perform the main calculations with events in a cycle. Usually it stops when all input data have been read. Then the *PostProcess* methods are called for each module to finalize the module's job, e.g. write histograms to disk, close files. Typically the first module in the list reads entries that need to be processed from a file, while the last module writes the derived data to disk. All these methods can return 3 types of values: *true* means that the method was successfully executed and the program should continue; *false* means that something wrong happened and the program should stop and *continue* means that the program should skip the current event and continue with the next one. Thus it is possible to create modules that work as an event filter that throw *continue* for rejected events.

The data processing and analysis implemented in BARS is divided into several steps, each of them is performed by an independent program which takes the output of one or more of the previous stages as input. Throughout the analysis chain the data are organized in ROOT trees containing a set of data containers that are stored on disk using ROOT serialization.

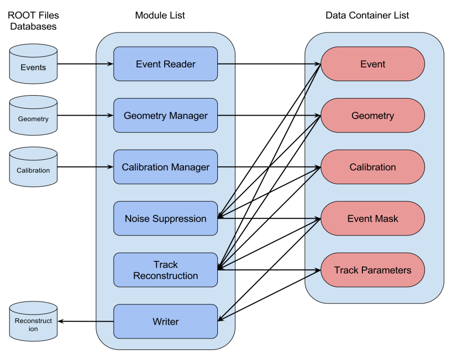

Figure 1: General structure of BARS on the example of reconstruction program.

3. Raw Data

The initial input to BARS is the raw data recorded by several clusters. A cluster is an independently working subarray of the Baikal-GVD telescope that consists of 8 strings of optical modules (OM). An OM consists of a large area PMT housed in a pressure resistant glass sphere. OMs are grouped to sections. There are 3 sections per string. A section consists of 12 OMs and a Central Module (CM) that digitizes signals from the OMs with a 200 MHz FADC and realizes a local trigger coincidence in FPGA logic. The trigger system of a cluster is illustrated in Fig. 2. A typical trigger condition is a signal amplitude in 4 FADC counts exceeding low and high thresholds in two adjacent OMs in a 100 ns time window. When the trigger condition is fulfilled, a request signal is generated to the Cluster Center module (CC) through a 1km long line. When the request signal is received, CC generates an acknowledgement signal to all CMs in a cluster. When CM is caught the signal, a timestamp is defined and the CM starts to form data. And it extracts 5 mks FADC slice from 30 mks buffer according to *offset* parameter. Only useful information around the signal waveforms in a 5 mks slice is transferred to the shore and stored as raw data in binary files. Additionally, raw data contain ascii files from the acoustic positioning system and from OM monitoring. Overall, such a trigger system approach allows to have all signal waveforms on each channel in an event from only one triggered pair of adjacent channels.

All raw data are stored in custom binary format, archived and transferred to the central storage and processing facility in the Joint Institute for Nuclear Research (JINR) in Dubna, Russia every 4 hours. For a typical event rate of 100 Hz, the unzipped raw data occupy approximately 50 Gb per day and are compressed by bzip2 archiver by a factor of 4. The data are converted to a couple of data containers that stores in ROOT trees.

4. Joint Events Production

As it is seen from the previous section, trigger event information is scattered in separate blocks of data that are obtained by 24 CMs. The first task of data processing is to unite them into one joint cluster event. The task is complicated by the fact that CM timers are not synchronized with an external source. So it is impossible to use absolute timestamps to do that, and we compare time

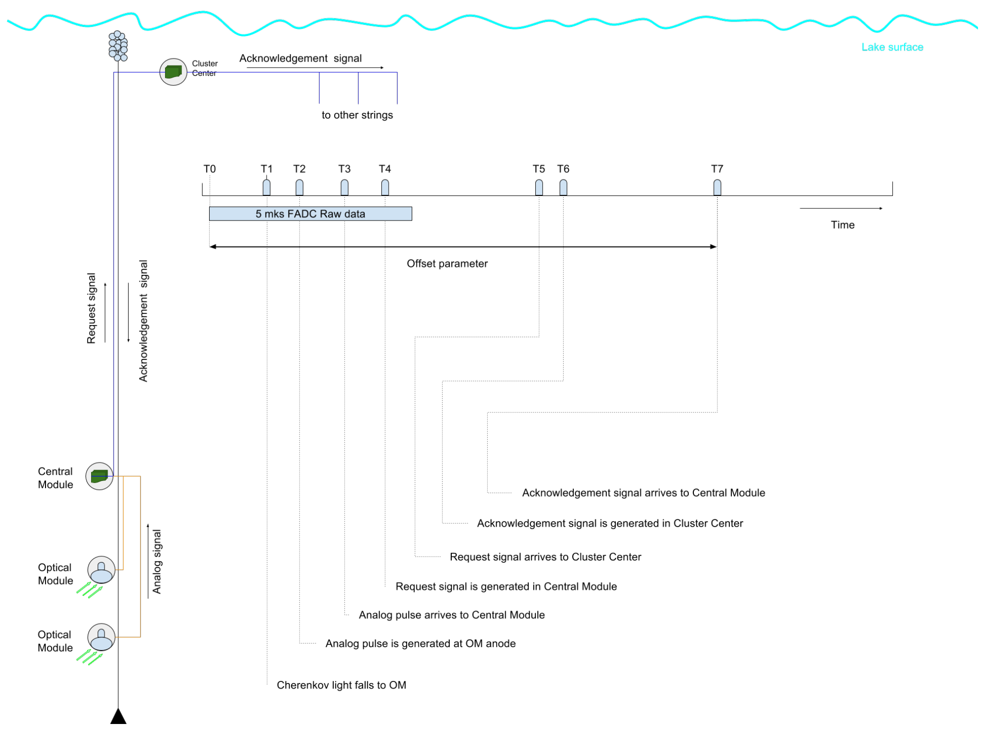
Lake surface
Cluster Center
Acknowledgement signal
to other strings
T0
T1
T2
T3
T4
T5
T6
T7
5 mks FADC Raw data
Time
Offset parameter
Request signal
Acknowledgement signal
Central Module
Optical Module
Optical Module
Analog signal
Acknowledgement signal arrives to Central Module
Acknowledgement signal is generated in Cluster Center
Request signal arrives to Cluster Center
Request signal is generated in Central Module
Analog pulse arrives to Central Module
Analog pulse is generated at OM anode
Cherenkov light falls to OM

Figure 2: This sketch shows how the trigger system works (see text). For the physics analysis we need to find out time T1 for all hits from observables which are waveforms in FADC with time T3 minus T0. The time difference between T2 and T1 depends on the transit time of the PMT. The time difference between T3 and T2 depends on the channel cable delay (approx. 90 m). The time difference between T4 and T3 depends on the CM logic latency and *request signal delay* which is one of the parameters of the detector. The time difference between T5 and T4 depends on the section cable delay (approx. 1 km). Since cables have different lengths, request signal delays are used in order to align time interval T5 minus T4 for all sections. The time between T6 and T5 depends on the latency of the Cluster Center. It should be noted that the absolute time T6 is a common time for all sections and we can have joint cluster events due to this time. The time between T7 and T6 depends on the section cable delay. The time difference between T7 and T0 is fixed by the *offset* parameter of the detector.

differences between sequential data blocks from one CM and another. All CM data are united sequentially to CC data. It is important to note that the run timers are calculated using already joint events to unite the next joint events. The quality of the joining procedure is confirmed if it is not interrupted during the whole run and the distribution of differences of time differences is gaussian with a sigma of 6 ns.

As mentioned above, two clusters are operating now. They work independently from each other. To unite events from several clusters, a new system that attach synchronized timestamps to events of each cluster was developed and is being tested now.

5. Signal Extraction

The next step of data processing is to extract the main characteristics from digitized PMT pulse waveforms. The waveforms are obtained in such a way that there are 10 pedestal FADC counts before the pulses that are used to subtract it. First, several tests are applied to the waveform to be sure that it is a real signal rather than electronics cross-talk. Using piecewise-linear approximation of the waveform the signal arrival time (time mark) is calculated at the front at half maximum level; later it is shifted using a time walk correction function that was obtained from time calibration. Also the maximum amplitude, the time over half maximum and the charge as the sum of digitized counts are calculated.

6. Time and Amplitude Calibration

After signal extraction, time and amplitude calibrations can be performed. For the analysis of physical events we should obtain the relative moments of the time T1 of Cherenkov light hitting OMs from observables of time marks (see section 5) T3 minus T0 (Fig. 2). To do that, so called intrasection and intersection calibrations were developed. Intrasection calibration deals with time delays T3 minus T2 that is caused by signal passing through 90 m cable and T2 minus T1, that is the transit time of the PMT. Both delays are measured in the laboratory before the module deployment. Delay T2 minus T1 is also controlled in-situ using a so called test pulse that is generated straight to the readout by OM controller and OM built-in LED pulse signal that is delayed to constant time from the first one. Also the total channel delay is cross-checked by the time difference between OM LED pulses at two adjacent OMs.

Delays between sections are defined in intersection calibration. In other words, the task is to align T0 times for all sections. They are shifted due to different T7 minus T6 delays that are caused by different cable lengths between CC and CMs. Intersection calibration is carried out with a similar approach as the intrasection calibration. It is derived from location of trigger pulses on 5 mks FADC slices and known offset parameters. The results are cross-checked by using artificial light sources such as a LED matrix and a laser source that illuminates all strings in a cluster.

Thus time calibration is carried out in separate calibration runs with a special trigger condition, switched on artificial light sources and generating test pulses directly to readout electronics. Time calibration is described in more detail in [6].

An amplitude calibration procedure is applied to convert signal amplitudes and charges into physically meaningful photoelectron units. Ordinary muon runs are used for this purpose, using the off-trigger region at the beginning of the 5 mks FADC slice (the trigger region is at its center) that contain mostly noise hits. It is known that noise background of deep Baikal water has 1 p.e. nature. The derived amplitude distribution is fitted with a gaussian that describes 1 p.e. signals and the low-amplitude exponential part. The gaussian mean defines the dependence between photoelectrons and observables in a linear range up to 100 p.e.

7. Data quality control

Data quality depends on the performance of the subsystems of the telescope. Information related to data quality is extracted from the data accumulated in the previous steps and analyzed, and if it reveals any problems in the hardware or its settings, an alarm message is sent to the responsible persons for maintenance. As some problems cannot be solved instantaneously, they have to be faced in the further analysis, and solutions have to be found to handle them in the software. To identify and quantify the problems, several quality criteria have been developed. First, the low-level trigger system performance is analyzed using the information of local trigger coincidence on FADC, request and acknowledgement signal counters, timers on CMs and CC. If all the information is self-consistent, a corresponding flag is set for the trigger event. Otherwise an *unsuitable* flag is set in order to use it potentially in the next steps of data analysis. About 98.5% of events have a clear trigger origin. Most of the rest of events occur at the vicinity of dead time windows and require more study. The channel and CMs rates of participation in the trigger are monitored by fitting of three types of histograms: rate vs. time, time intervals distribution between events, Poisson distribution of event rate. Also positions of trigger pulses that must be at the middle of the 5 mks FADC slices are controlled.

8. Dynamic geometry

The OMs are attached to flexible strings and can move tens of meters away from their original position over the course of the season. To account for the dynamic geometry of the detector, we utilize the acoustic positioning system (APS) [5]. APS is a network of acoustic beacons attached to cluster strings at depths of 736, 928, 1093, 1274 and 1366 meters. APS software on the shore station regularly polls beacons for the acoustic signal delay data, uses it to reconstruct beacon coordinates and stores them.

The APS output is a set of plain text records containing beacon id, coordinates and a timestamp. Before being used for reconstruction, this data undergoes offline preprocessing. During this stage, beacon coordinates are linearly interpolated at regular intervals, data quality metrics are evaluated and the output is stored in a ROOT file.

During data processing a BARS module reads acoustic data and interpolates OM coordinates from beacon positions for each event. The architecture of the dynamic geometry facility in BARS allows for multiple interpolation models that vary in accuracy and computational complexity. This way the detector geometry evaluation performance can be tuned for analysis scenarios with different speed constraints, e.g. on-line reconstruction vs. off-line calibration procedures.

9. MC data and the unified event format

Monte-Carlo simulations of the GVD clusters are performed with updated Fortran-based software that was developed for the first generation Baikal NT-200 telescope and is independent from BARS. The simulation is performed in two stages. In the first stage primary and secondary particle interactions are modeled within the observed medium. The trigger conditions and noise signals are applied afterwards and the output is stored in a binary file. BARS can import this data to a ROOT file containing objects for detector geometry, events and MC-specific information. The imported geometry is functionally equivalent to the dynamic geometry described in the previous section, and the MC events are stored in the unified event format — an array of time and amplitude values for all pulses in the event.

Unified events are the default input for every noise suppression and reconstruction module in BARS. Joint experimental data is converted to unified events after calibration. All data exclusive to experimental events (e.g. astronomical time or impulse waveform) or MC events (e.g. number of muons in the bundle) is stored in separate context containers. This approach allows us to use MC and experimental data interchangeably when running analyses that should work with both.

10. Noise suppression and event reconstruction

Optical modules produce noise pulses due to PMT dark current and lake noise with amplitudes at the one-photoelectron level and at a typical rate of 20-30 kHz. Therefore each string holds about 4-6 noise pulses per event window of 5 mks. These pulses must be distinguished from signal pulses. Noise suppression in BARS is performed with filter modules. Filter modules take a unified event and produce the event mask - a data structure that holds signal, noise or another utility flag for each impulse in the event.

The following method for noise suppression has been developed. Pulses produced in adjacent OMs are required to pass a certain amplitude threshold and are checked for causal connection. Pairs of pulses satisfying such selection criteria are combined in groups across the detector. An initial estimation of track parameters is obtained for each group as follows. Time and coordinates of the track candidate are obtained from time and coordinates of the first in time hit. For the direction estimation, vectors defined by pairs of impulses at different strings are constructed with the vector direction being from the earlier pulse to the later one. The sum of such vectors normalized to unity is used for the initial track direction estimation. A dedicated quality function is calculated for the considered group of pulses using preliminary track parameters. The group of pulses corresponding to the smallest ratio of quality function divided by the number of pulses is selected for track fitting. The expected purity of hit selection depends on the polar angle of the incoming particle and is not worse than 98% for an amplitude threshold of 1.5 photoelectrons. Along with the described technique other filters may be used in the processing chain to improve the reconstruction quality. Passing several event masks to the mask summation module will produce a combined output that masks only pulses flagged as noise in all inputs.

An event mask after noise filtration is supplied to the module performing the reconstruction of the track. The fit of the track is performed using MINUIT minimization routines from the ROOT data analysis framework. The aforementioned quality function which incorporates signal delay with respect to direct Cherenkov light and distance to track with a weight of signal amplitude is minimized. A median of 2 degrees for the mismatch angle distribution for upgoing atmospheric neutrinos was achieved with such a procedure, simulating just one cluster. The parameters of a muon track are stored in a dedicated container which is saved in the output file. Several output containers produced with different filtration or reconstruction parameters may be stored in the event.

11. Automatization of data processing

To apply the developed data processing steps for each run in a year, a processing automatization is required. Each step should be launched for new non-processed raw data fully available on disk and be skipped for already processed data. A data processing step should be started only for a run if all necessary input data from the previous one or several steps are available. It should be possible to vary the processing chain depending on the input data. For example, the time calibration procedure must be applied for calibration runs only. The intrasection calibration procedure is carried out in a so called autonomous or standalone regime when generated CM request signals are shortened and returned as acknowledgement signal to the same CM. For such runs the joint event production step should be skipped.

To meet such requirements, an automatization system was developed based on Shell scripts and MySQL database. We followed the simple and efficient ideas described in [4]. Status of data processing is contained in an appropriate run-based database table that has a pair of run number and season as a primary key. Each step in the data chain has a field in the table. The field contains NULL if the step has not been carried out successfully yet; datetime information if it has been carried out successfully or datetime of unix epoch start if it should not be carried out at all for the run. Thus to reprocess a step for some run, for example, with a new version of software, it is required only to put NULL in the appropriate field.

The first Shell script looks at raw data availability and puts an entry in the table for new runs. Following scripts corresponding to each step search for runs that should be processed. They check appropriate field and process runs that have NULL in that field. If some previous steps are required to be accomplished a script checks that their appropriate fields are not NULL. If the program finishes successfully the script inserts current datetime into the corresponding field. If not, an error code is inserted to a separate field. Finally the sequence of scripts is put into job scheduler *cron* and the processing chain runs in automatic regime. Only when a non-standard situation occurs, manual intervention is required.

The system has sufficient flexibility to add new features and add new processing steps with ease.

12. Summary

This paper presents an overview of the data management and processing system for the Baikal-GVD project that processes data in an automatic regime, as well as a sequence of processing steps that have already been implemented in the framework. Further steps are under development in order to achieve a full analysis chain needed to produce scientific output. They will be easily integrated in the system.

References

[1] A.D. Avrorin et. al., *Status of the Baikal-GVD experiment* this proceedings.

[2] ROOT Web pages, http://root.cern.ch/

[3] T. Bretz, R. Wagner *The MAGIC analysis and Reconstruction Software* in proceedings of *28th ICRC*, (2003) 2947-2950.

[4] D. Dorner, K. Berger, T. Bretz, M. Gaug *Data Management and Processing for the MAGIC Telescope* in proceedings of *29th ICRC*, (2005) **5** 175-178.

[5] A.D. Avrorin et. al., *Hydroacoustic Positioning System for the Baikal-GVD* this proceedings.

[6] A.D. Avrorin et. al., *Baikal-GVD: Time Calibrations in 2016* this proceedings.