The Central Data Acquisition System of the H.E.S.S. Telescope System

C. Borgmeier[1], N. Komin[1], M. de Naurois[2], S. Schlenker[1], U. Schwanke[1] and C. Stegmann[1], for the H.E.S.S. collaboration
(1) Humboldt University Berlin, Department of Physics, Newtonstr. 15, D-12489 Berlin, Germany
(2) Laboratoire de Physique Nucleaire et des Hautes Energies, 4 place Jussieu, T33 RdC, 75252 Paris Cedex 05, France

Abstract

This paper gives an overview of the central data acquisition (DAQ) system of the H.E.S.S. experiment. The emphasis is put on the chosen software technologies and the implementation as a distributed system of communicating objects. The DAQ software is general enough for application to similar experiments.

1. Introduction

The High Energy Stereoscopic System (H.E.S.S.) is an array of imaging Čerenkov telescopes dedicated to the study of non-thermal phenomena in the Universe. The experiment is located in the Khomas Highlands of Namibia. At the end of Phase I, the array will consist of four telescopes, two of which are already operational and being used for data-taking at the time of writing.

Each telescope in the array is a heterogeneous system with several subsystems that must be controlled and read out. The telescope subsystems comprise a camera with 960 individual photo-multiplier tubes, light pulser systems for calibration purposes, a source tracking system, an IR radiometer for atmosphere monitoring, and a CCD system for pointing corrections. Common to the whole array is a set of devices for the monitoring of atmospheric conditions, including a weather station, a ceilometer and an all-sky radiometer.

2. DAQ System Requirements

The DAQ system provides the connectivity and readout of all the systems mentioned above. It takes over run control, the recording of event and slow control data, error handling, and monitoring of all subsystems. The remoteness and small bandwidth connection of the H.E.S.S. site imply that the DAQ system must be stable and easily operated.

pp. 1–4

The main data stream is produced by the cameras which generate events with a size of 1.5 kB. At the design trigger rate of 1 kHz, this yields a maximum data rate of 6 MB/s for four telescopes, resulting in roughly 100 GB of data per observation night. The data rates from the other subsystems are significantly smaller.

On the hardware side the requirements are met by a Linux PC farm with a fast Ethernet network. The details are described in [1].

3. DAQ Software

The H.E.S.S. DAQ system is designed as a network of distributed C++ and *Python* objects, living in approximately 100 multi-threaded processes for the H.E.S.S. Phase I configuration.

For inter-process communication the *omniORB* [2] implementation of the CORBA protocol standard is used which provides language bindings for C++ and Python. CORBA allows to call methods of objects in remote processes and to pass data objects. The transport and storage of objects needs a serialization mechanism which is provided by the *ROOT Data Analysis Framework* [3]. Both the ROOT-based H.E.S.S. data format and the ROOT graphics and histogram classes are used online and offline allowing a seamless integration of data analysis code and hence fast feedback.

The base classes for all DAQ applications are provided by a central library. Derived classes, implementing the base class interfaces, control for example a hardware component or handle different types of data streams. Each DAQ process contains a `StateController` object which implements inter-process communication and run control state transitions.

The DAQ system distinguishes four different process categories as shown in Fig. 1. The **Controllers** directly interact with the hardware and read out the data. Each hardware component is controlled by one Controller process. The Controllers push the data to intermediate **Receivers**, which perform further processing and store the data. The Receivers also provide an interface that allows other processes to sample processed data. **Readers** actively request data from the Receivers at a rate different from the actual data-taking rate. The data or derived quantities are then available for display and monitoring purposes. **Manager** processes are not involved in the data transport but control the data-taking.

4. DAQ Configuration

Different data-taking configurations of the array correspond to different *run types*. A run type is defined by a set of required Controllers, Receivers, Readers, and Managers. Examples for run types are the observation run type with all available subsystems, various calibration run types for the cameras, and

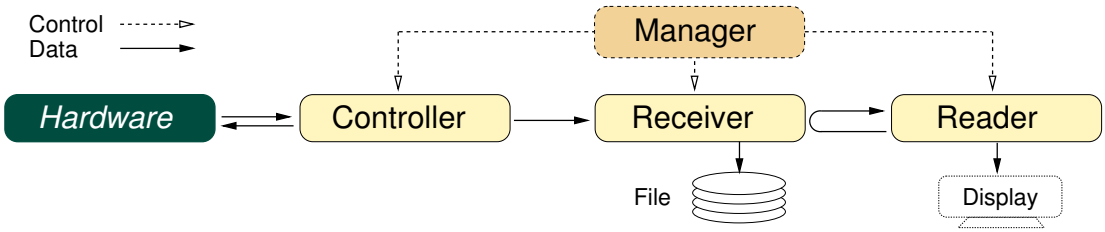

Fig. 1. Types of processes in the DAQ system

dedicated run types for testing and data-taking with specific subsystems, e.g. the tracking system. An actual run is given by its type and a set of parameters.

All run type definitions, run parameters, the configuration of the DAQ system, and observation schedules are stored in a MySQL database acting as central information source and logging facility for the DAQ system.

Setting up the required processes for a specific run type is simplified by combining related processes in groups that are called *contexts*. An example for a context is `CT1` which comprises all Controllers accessing the hardware of telescope number 1. Every context contains one Manager that controls the other processes in the context. At startup, the Manager reads the processes in its context from database tables and launches them. The Manager serves as an intervention point to all processes in the context, passes on state transitions, and takes over error handling in predefined ways.

For data-taking a central DAQ Manager reads the observation schedule from the database and determines the actual sequence of runs according to the availability of contexts. Runs that require different contexts can be processed in parallel. The central DAQ Manager launches the required context Managers and initiates the run preparation. After the preparation of a run the starting and stopping of the data-taking is taken over by a dedicated Manager which controls the participating contexts.

The shift crew interacts with the DAQ system via a central control GUI providing an access point to the system and direct monitoring of the states of the different processes (cf. Fig. 2 (left)). Data monitoring information is shown by a variety of different displays. The displays are generated by instances of a generic Reader which are configurable via database tables allowing a simple and flexible setup of the quantities to be displayed. Fig. 2 (right) shows some examples of monitoring displays.

5. Summary and Outlook

The described system is in operation since the start of the observation program with the first telescope. The setup based on database tables proved its

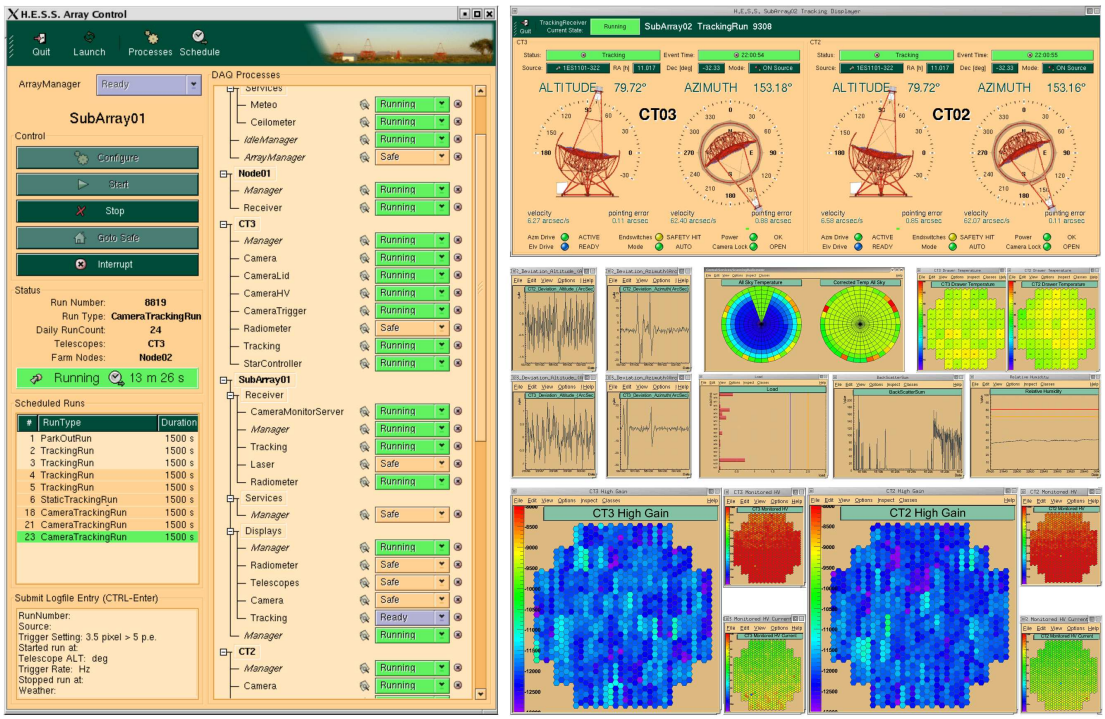

Fig. 2. Central control GUI (left) and examples of monitoring displays (right)

flexibility when integrating new subsystems into the data-taking. Processing parallel runs was exercised in the commissioning phase when observing with the first telescope while aligning the mirrors of the second telescope. The system is now taking data with two telescopes and is expected to scale well to the full Phase I array.

Acknowledgments. This work was supported by the Bundesministerium für Bildung und Forschung under the contract number 05 CH2KHA/1.

6. References

1. Borgmeier C. et al. 2001, Proceedings of ICRC 2001: 2896
2. S.L. Lo and S. Pope, The Implementation of a High Performance ORB over Multiple Network Transports, Distributed Systems Engineering Journal, 1998. See also `http://omniorb.sourceforge.net`
3. R. Brun and F. Rademakers, ROOT – An Object Oriented Data Analysis Framework, Proceedings AIHENP'96 Workshop, Lausanne, Sep. 1996, Nucl. Inst. & Meth. in Phys. Res. A 389 (1997) 81–86.
See also `http://root.cern.ch`