Acta Scientific COMPUTER SCIENCES

Volume 3 Issue 7 July 2021

Review Article

A Generic High Data Rate Archiving Software Solution: In Context of an Astronomy Experiment

D Sarkar[1]*, Shikha Srivastava[2], S Padmini[2], A Jain[2], N Chouhan[1] and Anita Behere[2]

[1]Astrophysical Sciences Division, Bhabha Atomic Research Centre, Trombay, Mumbai, India

[2]Electronics Division, Bhabha Atomic Research Centre, Trombay, Mumbai, India

***Corresponding Author:** D Sarkar, Astrophysical Sciences Division, Bhabha Atomic Research Centre, Trombay, Mumbai, India.

Received: May 10, 2021

Published: June 25, 2021

Abstract

Data explosion is the present-day challenge that all communities are facing. Scientific Experiments contribute to a large extent to this data generation process. Space and ground-based multi- messenger astronomical experiments generate high-speed data which needs to be archived and analyzed. The Major challenges faced by many of these experiments are remote location, low bandwidth, poor network connectivity, remote operation of the telescope, high data rate, varied data from different subsystems and, the delay between acquisition and analysis. One such category of a scientific experiment is ground-based gamma-ray telescopes built to detect Cerenkov radiation generated by the interaction of the earth's atmosphere and gamma rays from different astronomical sources. The Major Atmospheric Cerenkov Experiment (MACE) is a 21m diameter ground-based high-energy gamma-ray telescope set up by BARC at Hanle (32.7 ∘ N, 78.9 ∘ E, 4270 m asl) in the Ladakh region of North India. The telescope consists of various subsystems that generate data at different rates. The archiving system has to handle this varied data rate, providing sufficient read-write speed for real-time analysis. The camera and data acquisition system generates maximum data at an estimated rate of 1kHz with an average throughput of ~20 MB/sec. This rate may increase with the increase of hit pixels leading to the generation of ~500GB of data per observational night. The storage of data and subsequent analysis requires robust and fault-tolerant data archiving software. In this paper, we describe the challenges faced in designing the archival software. We present a detailed software architecture, design, implementation, and testing of the Data Archival System (DArS) Software for the MACE Telescope as a solution to overcome the challenges. In the light of the MACE Telescope, we have proposed a generic software design for archiving software for systems with similar functional requirements.

Keywords: Gamma-ray Astronomy; Data Archival; Software Design; Design Patterns; Multithreading

Introduction

Scientific experiments are of various domains like space-based experiments, ground-based astronomical experiments, particle accelerators, nuclear reactors. All these experiments generate data that needs to be archived and analyzed. Different experiments have different data rates, data volumes, and analysis techniques. For better resolution, the scale of these experiments has increased, moving from a single instrument to multiple instruments. For ground-based telescope experiments, scientists have moved to the concept of an array of telescopes. Some of these present-day telescopes Square Kilometer Array (SKA) [1], Cherenkov Telescope Array (CTA) [2], Event Horizon Telescope (EHT) [3], will flood the globe with the data generated from these experiments. It is required to understand the challenges of this massive data handling and build a data storage and analysis system which can handle the high data generation rates.

We present the archival software design for a ground-based Gamma-Ray Experiment. The design can be generalized for other experiments with similar functional requirements and data storage challenges. Gamma-Ray Astronomy deals with the study of the astrophysical sources of photons over a wide energy range (1MeV to 10^8TeV) [4], thus involving three principal detector techniques- Satellite experiments, ground-based atmospheric Cerenkov telescopes, and air shower arrays. Major Atmospheric Cerenkov Experiment (MACE) Telescope [5] belongs to the family of ground-based Very High Energy Gamma-Ray Astronomy Telescopes. It works in the energy threshold of 20GeV to 10TeV. It is based on the Imaging Atmospheric Cerenkov Technique (IACT) [6]. The telescope generates large volume data that needs to be archived and analyzed. The event data generated by the telescope undergoes data analysis steps like pre-processing, calibration, image cleaning, parameterization, gamma-hadron separation, energy spectrum formation, light curve generation, and sky map production. Apart from this, telemetry data is collected to carry out data quality checks on the event data and evaluate system health. This analysis chain demands the need for building a data storage system capable of handling large data generated at varied rates from different subsystems of the telescope.

Challenges faced for designing an archival system

There are several challenges a designer has to face when designing a data archiving system. The design of any system comprises both hardware and software aspects. The first challenge faced is related to the physical transfer of data. These telescopes are located in remote places. In the case of an array of telescopes, they may be geographically distributed. So the transfer of data may not be possible over the internet, leaving behind either *in situ* analysis or physical transfer of data. In the case of an array of telescopes, the data rate becomes very high, and data from all the telescopes need to be accumulated and analyzed to get the physics meaning out of it. This delay between data acquisition and accumulation of data in the data center delays the analysis process. The analysis technique also varies from one experiment to the other. So optimized analysis algorithms that work for one experiment may not workfor another experiment. It demands customized experiment-specific software.

The first challenge for the MACE Telescope data archival system was the hardware design that had to sustain difficult dry low atmospheric pressure at Hanle, which increases hard disk failure rates. The next challenge was to design the set of software modules once the hardware design was finalized. Our goal was to optimize the number of independent software modules and exploit parallelism in each of the software entities. Initially, we discuss here the existing model and the limitations faced by it in the next section.

Existing design and motivation

The major motivation for the work was to develop a generic design of data archival system software that satisfies the functional and performance requirements to handle large data volume with the varied data rate. Initially, the application developed for data archival was a single-threaded graphical user interface based application (Figure 1). Statistics and event updates were displayed in the same application. But with this approach, we faced delayed display of updates since a single thread was unable to handle data receiving and visualization. At the same time, it was unable to handle simultaneous datareceiving from different subsystems at different data rates. Over time feature requirements for the application also increased. The existing model lacked the flexibility of expansion for multi-telescope system. So we needed to explore the design characteristics for software development and redesign the software with scalability to resolve these issues. In [7] model-based software development approach for real-time systems was discussed. The model-based approach helps to understand the functional requirements better in the initial stage of the design itself. In [8] the authors discuss the different advantages and disadvantages of the agile approach of software development. The agile approach provides flexibility in design, minimizing release time, and quick functional additions. But it is heavily dependent on user interactions. In [9] different software design approaches and design patterns are discussed.

The paper is organized into sections. In the next section, we describe the subsystems of the MACE telescope along with the data characteristics, rates, and volume. The third section describes the design solution of the archival system. This design includes hardware design followed by detailed software design and architecture. The last section deals with the implementation, testing, and performance of the proposed model.

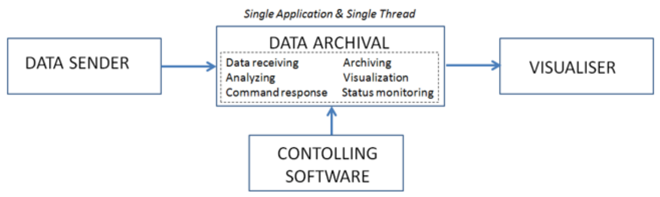

Figure 1: Single thread based existing archival design.

MACE telescope subsystems and data rates

To design any high data handling software it's important to understand the data generation process and the types of data involved in the system first. The telescope aims to recognize the onset of cosmic events of interest based on predefined trigger conditions. An array of Photo Multiplier Tubes (PMTs) in the focal plane of a large optical reflector constitutes a camera to record the Cerenkov image for each triggered shower. The camera is triggered when a preset number of PMTs detect a light level above a set threshold within a short integration time of nanoseconds. Digitally the light level in all the pixels is recorded. The image is analyzed to determine whether it has the characteristics of a gamma-ray shower. The subsystems of the telescope along with the data generated are described as follows.

Camera electronics and data acquisition system

Camera Electronics and Data Acquisition subsystem [10] responsible for event generation. This subsystem constitutes an array of PMTs, Central Camera Controller (CCC), First Level Trigger Generator (FLT), Second Level Trigger Generator (SLTG), and Data Concentrator (DC). The modular structure of MACE Camera consists of 1088 PMTs, grouped into 68 Camera Integrated Module (CIM), each module consisting of 16 PMTs. CCC is the central controller of this subsystem. FLT and SLTG contribute to triggering generation and validation. DC is responsible for the accumulation of the event data generated from 1088 PMT and records the signal generated by the PMTs.

A PMT is considered to be hit if charge content is above some threshold value. An event is said to occur if a pre-configured trigger combination is satisfied by the hit PMTs. For a hit, the event profile is also saved along with the accumulated charge content. Event data comprises event number, packet identifier, packet length, timing information, hit pattern accumulated charge content of channels along with the profile data of the hit channels and hardware parameters of the electronics. The size of one event packet varies with the percentage of hit pixels. The estimated event rate for the MACE Camera is 1 kHz. Considering 10% of hit ratio the approximate size of an event packet is 20KB, thus a maximum of 20MB/s data will be generated. Considering 8 hours of average observational night ~500GB of data will be generated per night. This data volume may increase depending on the increase of hit ratio and the number of events generated.

Apart from the event data, telemetry data are also generated at a rate of one telemetry data packet every 5 seconds from this subsystem for health monitoring. These parameters include Anode Current, Single Channel Rates (SCR), and Prompt Coincidence Rates (PCR), and Chance Coincidence Rate (CCR), a list of disabled PMT, power supply status and overall camera temperature.

Active mirror alignment system (AMCS)

356 mirror panels are mounted on the telescope basket, which is used to focus the incident Cerenkov light produced by the air shower onto the Camera. The alignment of the mirror panels is controlled by the Active Mirror Alignment System (AMCS) [11]. AMCS generates periodic status information consisting of panel encoder values and laser status.

Telescope control unit (TCU)

MACE telescope is alt-azimuth mounted. TCU [12] is responsible for tracking of the sources. Pointing of the telescope is controlled by reading the angular position of encoders connected to both axes, which provides a resolution of a few arc-seconds. Telescope tracking is monitored through the parameters generated from TCU at an interval of 100ms. These parameters include zenith and azimuth angles, pointing errors, motor speed and current, telescope mode and tracking status.

Sky monitoring system (SkMS)

SkMS quantifies the sky transparency level and checks the tracking accuracy of the telescope during observation. This subsystem consists of CCD (Charged Coupled Devices) camera mounted parallel to the optical axis of the telescope. During observation SkMS periodically grabs the sky image and analyzes them to generate the sky condition and tracking accuracy data.

Weather monitoring system (WMS)

WMS is responsible for measuring the weather parameters and indicates the feasibility of system operation. Weather parameters like wind speed, wind direction, rainfall, and temperature collected from the WMS guide the telescope operation interlocks.

Calibration system

Calibration System consists of LED Calibration System [13] and Sky Calibration System generating LED and Sky data respectively. Calibration data are used in the first phase of the data analysis chain. Calibration events are captured by artificially triggering the Camera based on the configuration parameters. These parameters include the frequency of calibration events, calibration mode, LED status, and amplitude. LED calibration data are used for relative gain calibration of PMTs and Sky Calibration data are used for pedestal evaluation during data analysis.

MACE operator console (OC)

Telescope subsystems are controlled and monitored by a set of software that forms the MACE OC [14]. MACE OC controls the subsystems on a command and response basis. The data generated by MACE OC constitutes of configuration data and schedule data for observations. Configuration data include both system-related and observation related configurations. Schedule information consists of sources to be observed along with the time and source coordinates.

Data archival system (DArS)

DArS stores the data generated from all the subsystems. It maintains a directory structure for storing event data and telemetry data. The data are saved on the basis of observation year, month, day, and observation number. Initially, the data are saved in a self-defined structured binary format which is termed as Level-0 data. Level-0 data are analyzed by the analysis subsystem. DArS is also responsible for data availability and backup.

Data analysis system (DAnS)

The data saved undergoes both offline and online analysis by the Data Analysis System (DAnS). Data quality checks are applied to Level-0 data and converted into Level-1 binary data. The hardware information is also removed from the Level-0 data. The Level -1 data in binary are converted into Level-2 ROOT [15] format with minimized required information in compressed form. The Level-2 data are used for offline data analysis. The raw data received from the subsystems simultaneously undergo online analysis.

All the subsystems are time-synchronized with GPS based Master Clock [16]. The data generated from the subsystems mentioned above are tabulated along with the hourly data rates in table 1.

Sl. No	Subsystem	Data Generated	Data Rate Bytes/hr.
1.	Camera Electronics and DataAcquisition System	Event Data	72GB
		Telemetry Data	15 MB
2.	Active Mirror Alignment System	Alignment Status	5 MB
3.	Telescope Control Unit	Tracking Parameters	15 MB
4.	Sky Monitoring System	Sky Data	60 MB
5.	Weather Monitoring System	Weather Parameters	6 MB
6.	LED and Sky Calibration System	Calibration Data	40 MB
7.	Operator Console and Monitoring System	Configuration and Schedule Files	20 KB
8.	Data Analysis System	Level 2 Event Data, Analysis Results	8 GB

Table 1: Data rates from different telescope subsystems.

Design solution of the archival system

Hardware design

MACE data archival system hardware has been designed as a combination of Solid State Drives (SSDs), hard disk drives, and tape drives. SSDs are deployed at a critical stage due to their low failure rate compared to general-purpose hard disks, at places of low atmospheric pressure. To optimize cost bulk storage is realized using hard disks and tape drives exploiting redundancy as well as diversity for fault tolerance. SSD drives are configured as RAID-6 [17] array with 2.8 TB usable capacity. Hard disk-based storage is realized using general-purpose disks and comprises of Just a Bunch of Disks (JBOD), arranged in RAID-6. Tape storage is based on LTO-6 technology each with 2.5 TB capacity and 2.5:1 compression. Block diagram schematic of the Archival System hardware is provided in

(Figure 2). It consists of two identical archival servers and three storage servers connected with the tape library and JBODs. The servers consist of Intel hex-core dual processors, 96 GB RAM, 15 MB cache, RAID array of SSD drives, SAS RAID card with 16 internal and 8 external ports, SAS HBA card with 8 external ports, quad Gigabit Ethernet NIC, IPMI, and redundant power supplies. A gigabit switch connects both the servers to the MACE network. Under normal conditions these servers perform their pre-assigned tasks, but if one fails the other can take over. Among the three storage servers, one is kept spare and rest two are connected to the MACE network. Each storage server hosts a tape library of 60TB capacity and JBOD of 48TB capacity. So the aggregate capacity with both the storage controllers is 120 TB tape capacity and 96 TB usable disk capacity.

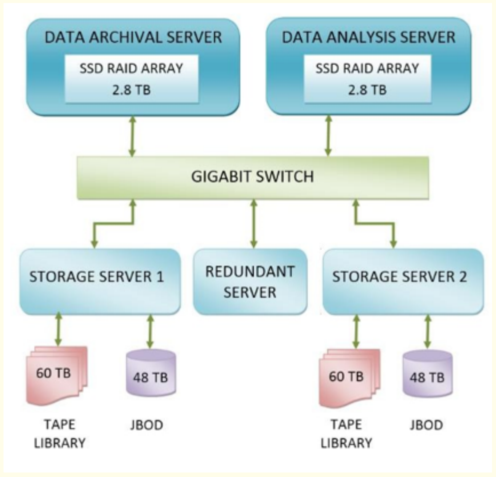

Figure 2: Hardware architecture of data archival system.

Software requirement

The data archival system has six functional requirements. These requirements are data handling, archiving, meta-data mapping, data staging and backup, online and offline data analysis, and remote backup.

Data handling

The incoming data from different subsystems of the telescope are received over TCP/IP protocol at different data rates and handled by customized self-defined applications. The Application layer hosts different software applications responsible for data exchange and handling. This data needs to be handled efficiently to maintain data integrity and prevent data loss. Apart from the subsystem data, commands are received from MACE OC which guides the data handling and archiving policies. Subsystem data has to be sent to OC and the other data monitoring software.

Archiving

Raw data received needs to be archived in separate files maintaining a well-defined directory structure to facilitate data retrieval. The data path for storage should be configurable. This is the Level -0 data that are to be archived in the SSDs.

Meta-data mapping

Data files generated by the telescope subsystems during observation needs to be stored in observationfolders. Users will retrieve these data files based on meta-data parameters like source name, date of observation, file type, and observation number. These parameters will be mapped to the observation data files for data retrieval.

Data staging and backup

Data from the SSD are to be copied to the tape drives as well as hard disk drives. The older data needs to be deleted to create space for the new incoming data. This data copy and deletion are to be done during the day time to avoid data corruption during the data staging process.

Online and offline analysis

The online analysis of the telemetry data packets gives an overall system health performance measure.A quasi online approach is to be followed in case of event data online analysis using simplistic algorithms with less stringent parameter cuts. A detailed offline analysis procedure will be performed later on Level-2 ROOT files.

Remote backup

A similar file structure architecture is to be maintained for remote backup and further data analysis at BARC Mumbai. MACE Telescope site Hanle and BARC Mumbai are connected through 256 kbps shared satellite link. This link will be used for remote control and monitoring of the telescope, as well as the transfer of test run data. Due to limited bandwidth and large data volume, backup of observation run data has to be done offline. The final backup of the data will be done on tapes and physically transferred to BARC Mumbai.

The above requirements are realized by a set of software processes. In this paper, we will focus on the architecture and design of the software responsible for data handling, archiving, and meta-data mapping.

Subsystem Interconnections

MACE Telescope consists of several subsystems described in Section 2 residing at varied physical locations. DArS hardware resides in the control room and is connected to the other subsystems over Ethernet. The four systems directly communicating with DArS are - DC sending event data, MACE OC sending operating commands, Telemetry Server (TS) sending subsystem data and Online Data Analysis Server (DAnS) receiving event file path for quasi online analysis (Figure 3). DArS is connected to DC, which is mounted on top of the telescope, through dual redundant optical fiber 1 Gbps link. DC collects data from the CIM modules over fast LVDS links. MACE OC located at the control room is connected to DArS over 100 Mbps Ethernet link. The data and command path are completely separated to minimize network delay. The other systems like TCU, WMS, SkMS, and AMCS control servers are located outside the control room. DArS is connected with these subsystems through fiber optic communication links. These subsystems multicast system health parameters at a given interval which needs to be archived at DArS. Instead of having separate communication interfaces with each of the subsystems, DArS has a single interface with TS for handling subsystem telemetry data. DAnS analyses the event data stored at DArS. These two systems are connected over the Gigabit switch and have a Network Attached Storage (NAS) layout for the sharing of data files.

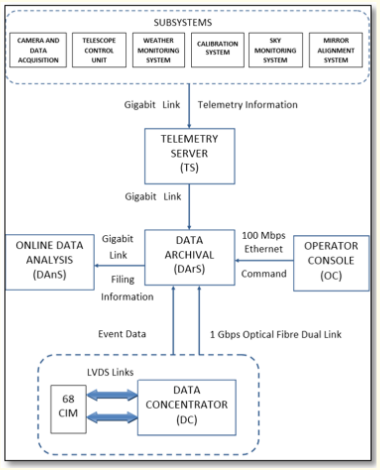

Figure 3: Subsystem interconnections with data archival system.

Software architecture

The DArS software has been designed following a Layered Architecture. It consists of five layers, which are the Data layer, the Data access layer, the Communication layer, the Service layer, and the Presentation layer (Figure 4). Each layer has its well- defined functionality.

Data layer

The data layer consists of the data from different subsystems along with the meta-data stored in files and databases. This layer constitutes raw data in binary format, analyzed data, and meta-data for data retrieval.

Data access layer

Data Access layer is responsible for handling the data from the Communication Layer. File Manager and Data Mapper components are responsible for performing the tasks in this layer. The major role of the File Manager is to receive data and archive the same in the data containers of the data layer. IndividualArchiver is responsible for file handling of a specific subsystem. The file manager's responsibility is to delegate the filing job to the respective Subsystem Archiver. Meta Data Mapper module is responsible for interpreting the data files, generating meta-data from these files, and managing the database. The meta-data generated is required for data retrieval.

Communication layer

This layer is the core of DArS software with the primary responsibility of receiving data and command from subsystems. On receiving data the communication manager notifies the upper layer components for specific actions. Along with this, it notifies the Data Access layer components for storage. This layer consists of managers for each subsystem. OC-DArS Manager, DC- DArS Manager, TS-DArS Manager, andDAnS-DArS Manager handle communication between MACE OC, DC, TS, and DAnS with the DArS respectively. Communication Manager internally follows a client-server architecture where the communication managers act as servers to the respective subsystem clients. This layer has a TCP/IP protocol based communication.

Service layer

The Service layer consists of four components and these are Command Parser, Status Manager, State Manager, and Analysis

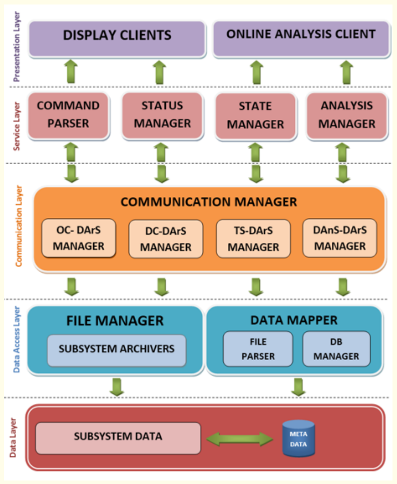

Figure 4: DArS software architecture.

Manager. These components receive data from the communication layer and perform actions according to their roles. Command Parser parses the commands received from OC, interprets them, and notifies the communication layer for sending the response toOC. This module alsonotifies the other modules of the service layer to perform actions based on commands received. Status Manager is responsible for monitoring data archival system health parameters and data checks. These parameters include network connectivity, disk space status, data validity, packet loss rate, and event data rate. The monitoring parameters evaluated by the Status Manager are passed on to the upper layer for visualization. State Manager manages state-based behavior of DArS based upon the commands received from MACE OC. Analysis Manager is responsible for notifying the Online Analysis Client with observation run and data file information.

Presentation layer

The presentation layer is the topmost layer of the five-layered architecture. It consists of Display Client and Online Analysis Client. The Display Client provides instantaneous event data and telemetry data parameter visualization. The Online Analysis Client generates periodic statistics for these parameters.

Software design features

The five-layered architecture has been designed with an object-oriented paradigm. Each component module from the architecture is segregated into one or multiple classes. In figure 5, we have presented the classes and their relationships. An object-oriented approach has resulted in ease of maintenance and re-usability. The SOLID [18] principles have been followed in the implementation of the classes. Each class has been strictly designed with a single responsibility. For example, the communication manager classes from the Communication Layer is responsible for handling TCP and UDP based communication with the DArS interfaces. UDP multicast protocol is used for receiving telemetry data from subsystems.Separate classes handle the responsibility of command interpretation, status monitoring, state management, and analysis notification. These classes are implemented following an open-closed principle, which makes the design loosely coupled.

The classes are designed using various design patterns [19]. Singleton pattern is followed for the creation of objects. The interaction between the classes is guided by an event-driven architecture using the Event Listener and Notification scheme based on the Publish-Subscribe pattern. In Publisher Subscriber pattern one class can subscribe to multiple events and publish multiple notifications. Classes subscribe only to the events of interest. This supports loose coupling among the classes and hence provides scalability. Communication Layer components continuously listen for data packet receiving events and notify to the File Manager and Service Provider Classes for performing required actions. Events are generated on receiving commands, event data, and telemetry data, and the respective modules like Command Parser, Status Manager, and File Manager are notified. Individual Subsystem Archivers are responsible for subsystem data archiving. File Manager Class is the base class for data archiving. Façade pattern is used for designing File Manager and Subsystem Archiver classes. The user interacts with the File Manager for filing and File Manager decides which Subsystem Archiver to pass on the data packet for storage. Communication Manager implementation is based on the Reactor pattern for communication establishment, read and write to sockets. Each communication thread reacts to I/O events by dispatching the work to the appropriate handler. The Handler performs the actual work. In this case, it is handled through a synchronous event de-multiplexer select.

The data archival software needs to handle data rates at high speed with full utilization of the memory cores. This is achieved by using a multithreaded approach. Each thread is assigned a single task. There are six threads including the main thread. The threads are responsible for communication with MACE OC, DC, TS, and DAnS. Separate threads facilitate data handling at an efficient speed. The design avoids resource sharing among these threads by copying the data to reduce overheads associated with locking. The classes are unified, following packaging principles. Some of the packages developed are like Communication Manager, File Manager, Event Notification and Listener, Thread, and Configuration can be reused for software with similar architecture.

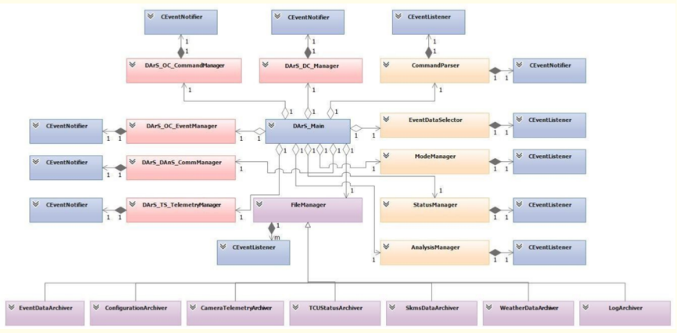

Figure 5: Class diagram.

Implementation and testing

Implementation

DArS software has been implemented following an iterative and incremental life cycle model [20]. The software is deployed on the Linux operating system, using C++ as the programming language and following POSIX standards. The sequence of events and interaction between the classes for implementing the functionality is represented in (Figure 6). The software features implemented are described in this section.

Robust communication with operator console (OC)

The primary responsibility of the data archival software is to establish communication with its interfaces. TCP based communication is established between DArS and OC to receive commands for operations. Robust communication with OC is provided with the features like disconnection check, re- connection, and remote operation from BARC, Mumbai to telescope site at Hanle.

Dual redundant data link with data concentrator (DC)

Event data sent from DC are received by DArS over TCP based dual redundant link. It receives data packets from both the links and maintains a queue of recently received packet identifiers. If the data packet identifier of the received packet is already present in the queue, the packet is discarded as a duplicate packet.

Command Interpretation

During an observation run, DArS receives Set Filing Parameter, Start DAQ, and Stop DAQ commands from MACE OC. As shown in figure 5, command interpretation is handled by the Command Parser. Set filing parameter command initiates the preparation for filing. These commands control the TCP sockets for communication. Depending on the execution status of these commands, success or failure response along with error codes are sent to OC. The error conditions are also logged in run based log files.

File management

The communication classes receive the data and notify to the File Manager for data storage. On receiving the Set Filing Parameter command, the File Manager invokes the Configuration Archiver, Telemetry Archiver, and Event Data Archiver to create their respective directories and files. File Manager starts saving the data in these files, once it receives the Start DAQ command.

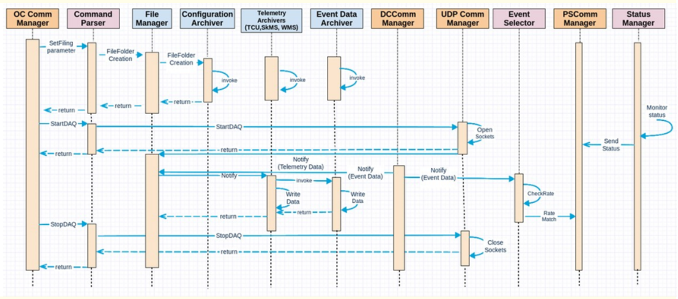

Figure 6: Sequence of operations.

Telemetry data collection

Telemetry data collected from other subsystems are also saved in the observation run directory. This is done by the TS-DArS communication manager which receives telemetry data from the telemetry server over a multicast mechanism and redirects it to the File Manager for archiving.

Meta-data mapping

Each event data file contains run specific meta-data described as the Run Header. Run header includes observation date, source, and telescope subsystem information. The data files archived in the observation run directory is mapped with this run header information by the Metadata Mapper class.

The meta-data information corresponding to the run is saved in the database.

Monitoring and error handling

Status parameters like packet loss, event data validity, data rate, disk space, and file closure are monitored by Status Manager. DArS maintains its count of events to account for packet loss. Event data validity is cross-checked at an interval of every 1000 data packets. If the case of data validation failure, the event is logged in the log files. The event data rate is independently calculated by DArS and data acquisition system to recognize data loss occurrences. Available disk space is continuously monitored. Warning and error messages are generated on 80% and 95% percent of disk space utilization. Warning message initiates data backup from SSD to tape drives and hard disks. On event data file closure it sends file information to an online data analysis server for performing quasi-online event data analysis.

Testing

Three-level functional testing [21] was carried out. Testing consisted of unit testing of individual classes followed by integration testing of the packages developed, system testing, and performance testing. Users have done system validation.

Unit testing and integration testing

Unit testing was carried out for each component in the layered architecture. Communication Manager components were tested for frequent connection establishment, disconnection, and reconnection with the other network interfaces. Service layer components like Command Parser were tested by sending commands from stub modules to DArS. File Manager and Meta-data Mapper were tested by generatingrequired data from the simulators of individual subsystems. Event generation was done by configuringLED Driver at different data rates. The File Manager was tested for correctly delegating the filing responsibility to respective Subsystem Archiver classes. DC-DArS Communication Manager and File Manager packages were integrated and tested for events receive at different rates and proper filing at a specified location.

System testing and user validation

Initially, individual components were tested. Then system-level testing was carried out with the actual subsystems. System-level testing was done with 4 CIM Set-up at BARC, Mumbai. User-level software validation was performed, after system testing.

Performance testing

The final version of the software was tested for its performance related to different aspects like handling of high and varied event data rates, validation of data files, disk space check, and stability. Multiple test runs were carried out, with different trigger configurations, calibration parameters, and run durations. Calibration limits were finalized, by testing the software to cover the full range of event rates for 4 CIMsetup. The software was also tested for warnings and errors related to disk space and data loss. The stability of the software was initially tested by carrying out test runs of continuous 12 hours with 4 CIM set up in the lab and later at the telescope site with 68 CIMs.

Conclusion and Future Work

DArS software could successfully handle the required rate of 1 kHz. The software response was tested successfully by increasing the data rate from 10Hz to 1 kHz. Calibration limits fixed from 10 Hz to 1 kHz.

~840GB of data were collected from 2018 to 2019 during the testing phase with 4 CIM Set up at the lab. We have also tested the software with 68CIMs at telescope site Hanle from October 2018 till February 2019. ~ 1TB (Figure 7) of Level 0 data were generated from the test runs at telescope site Hanle duringthe installation phase of the telescope from 2017 to 2021. Observational test runs were carried out in 2021, with all the telescope subsystems installed completely. With full CIM load and maximum data rates, the DArS software performed as expected without any deviation from the test runs carried out atthe lab, with just 5% CIM modules. A stable version of the software was released at the site for use.

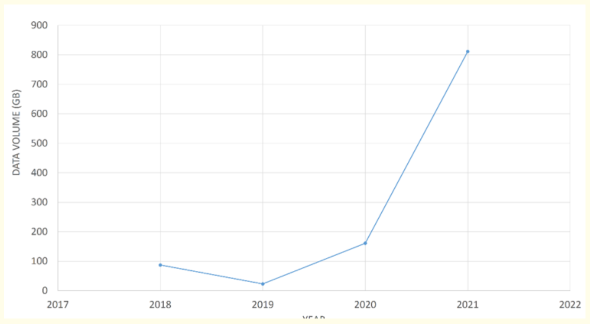

Figure 7: Data collected by DArS s/w during installation phase of the telescope.

In the upgraded model (Figure 8), we have fully utilized the archival hardware consisting of a hex- core processor. In our approach, we separated the core application responsible for data handling and graphical displays for visualization. Since data handling was the major challenge, we separated one thread explicitly for event data handling and the other for handling data from the rest of the subsystems. We followed a multithreaded approach consisting of 7 threads. The threads have the specific task of Event Data Receiving, Telemetry Data Receiving, Command Response handling with MACE OC, Data Archiving, Status Monitoring, Visualization, and Analysis of the data. The model can extend to N threaded application. N depends on the number of most important functionalities in the archiving software, major data-producing entities, and physical cores. Event notifications are easier with a single application multithreaded approach with minimizing notification delays. The layered architecture followed for designing the software helped in segregating the tasks and logical layering of the components. Each component of the layers could be designed independently, providing flexibility and scalability of the software implementation. Usage of design patterns, open-closed principles, and multithreading while designing the software has helped us to develop stable and maintainable software.

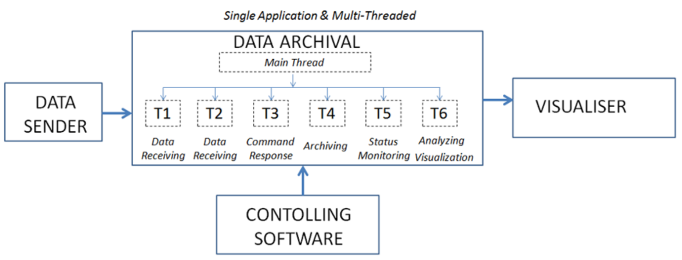

Figure 8: Upgraded data archival software model.

Similar functional requirements guide the archival of data generated from any system. The complexity of the data archival process depends on the complexity of the system generating the data. In this case, MACE Telescope subsystems generate a large volume of data, with different data structures at varied data rates. The hardware of the Data Archival system provides reliability and fault tolerance. The functional requirements like data handling, archiving, and meta-data mapping are realized with the Multi-threaded and Multi-Layered DArS application. These design features guided the implementation of the DArS software. Performance tests successfully carried out at the lab and telescope site proved the robustness and consistency of the software developed. The high-level layered software architecture and the design features described in this paper can be adopted as generic architecture and design for developing any data archival system software for the telescopic system. In this work, we have exploredonly a single telescope system. In the future, we would like to explore the implementation of this software model in the case of multi-telescope systems.

Acknowledgment

We are thankful to our colleagues at the Astrophysical Sciences Division and Electronics Division, BARC,for their inputs and suggestions during this work. Computer Division BARC Mumbai has designed the hardware layout of the Data Archival system. This work would not have been complete without the support of our colleagues working at the telescope site, Hanle. Apart from this, we are grateful to everyone involved in the MACE Project for their support.

Bibliography

1. A Scaife. “Big telescope, big data: towards exascale with the Square Kilometre Array”. 378 (2020).
2. M P Arribas., *et al*. “Trigger and data rates expected for the CTA Observatory”. in Heidelberg Symposium on High Energy Gamma-Ray Astronomy (2012).
3. The Event Horizon Telescope Collaboration. “First M87 Event Horizon Telescope Results. III. Data Processing and Calibration”. *The Astrophysical Journal Letters* 875.1 (2019).
4. R A Ong. “Very high-energy gamma-ray astronomy”. *Physics Reports* 305.3-4 (1998): 93-202.
5. C Borwankar., *et al*. “Simulation studies of MACE-I: Trigger rates and energy thresholds”. *Astroparticle Physics* 84 (2016): 97-106.
6. T Saito. “Study of the High Energy Gamma-ray Emission from the Crab Pulsar with the MAGIC telescope and Fermi -LAT”. arXiv:1105.3400astro-ph.HE (2011).
7. B Selic and L Motus. “Using models in real-time software design”. *Control Systems, IEEE* 23 (2003): 31-42.
8. B Choudhary and S K Rakesh. “An Approach using Agile Method for Software”. in 1st International Conference on Innovation and Challenges in Cyber Security (ICICCS 2016), Greater Noida, India (2016).

9. R N Taylor and A v d Hoek. "Software Design and Architecture The Once and Future Focus of Software Engineering". in Future of Software Engineering, IEEE Computer Society (2007): 226-243.

10. K Jha and A Behere. "Data Transfer Scheme for High Event Rates for MACE Camera". in National Symposium on Nuclear Instrumentation (2010).

11. P Kurup., *et al*. "Active Mirror Alignment Control System for the MACE Telescope". in National Symposium on Nuclear Instrumentation (2010).

12. P Kurup., *et al*. "MACE Telescope Servo Controller Design". in National Symposium on Nuclear Instrumentation (2010).

13. N Chouhan., *et al*. "Gain Calibration System for the MACE Telescope". in Symposium on Advances in Control and Instrumentation, Mumbai (2014).

14. S Bharade., *et al*. "State Based Control Design of MACE Console System". in National Symposium on Nuclear Instrumentation (2010).

15. R Brun and F Rademakers. "ROOT: An object oriented data analysis framework". *Nuclear Instruments and Methods* 389 (1997): 81-86.

16. D Sarkar., *et al*. "GPS based Event Time Stamp and Time Synchronization of various Subsystems of MACE". in XXXV Meeting Of Astronomical Society Of India, Jaipur (2016).

17. A K Sahai. "Performance aspects of RAID architecture". in 1997 IEEE International Performance, Computing and Communications Conference (1997).

18. R C Martin. "Design Principles and Design Patterns" (2000).

19. E Gamma., *et al*. "Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional Computing Series".

20. I H Sarker., *et al*. "A survey of Software Development Process Models in Software Engineering". *International Journal of Software Engineering and its Applications* 9.11 (2015): 55-70.

21. I Hooda and R S Chhillar. "Software Test Process, Testing Types and Techniques". *International Journal of Computer Applications* 111.13 (2015) 10-14.

Volume 3 Issue 7 July 2021