DDS: The Dynamic Deployment System

A. Lebedev[1] *and A. Manafov*[1]

[1]GSI, Darmstadt, Germany

The Dynamic Deployment System (DDS) [1] is a toolset that automates and significantly simplifies a deployment of user-defined processes and their dependencies on any resource management system (RMS) using a given topology.

During 2017 we focused on shared memory channels, lobby-based deployment and DDS session feature.

Shared memory channels

In the initial implementation DDS agents used to have only a network connection transport for communication with user tasks. This introduced certain implications, for instance, there was no guarantee that all key-value updates or custom command messages will be delivered to the user. As a fallback solution a shared memory was used to cache messages coming from network channels, to make sure that all messages are actually delivered to the user tasks. In order to improve and simplify this algorithm we have implemented a generic shared memory channel. The channel has similar API as DDS network channel, it supports two way communication, asynchronous read and write operations. Its implementation is based on the boost::message_queue library [2], on the DDS protocol which is used for message encoding and decoding and on the boost::asio library [2] for thread pooling and implementation of the proactor design pattern. The shared memory channel is used for communication between DDS agents from the same lobby and between DDS agents and user tasks, which significantly simplifies and secures the implementation. There is no need to cache messages any more as we now can guarantee the delivery. All messages are stored directly in the shared memory and managed by the message queue.

Lobby-based deployment

The main goal of the DDS is to be able to handle hundreds of thousands user processes. In DDS world each user process is controlled by a DDS agent (watchdog). Having all agents connecting back to a central DDS server (commander) is extremely resource consuming. We therefore implemented a so-called lobby-based deployment feature.

DDS agents of a given user on one host represent a lobby. A lobby leader is the only agent, which has a direct network connection to commander. A lobby leader is elected locally on each host. The election process is a local negotiation between agents and no connection to the commander is required. All other agents are lobby members communicating with the commander via the leader. Agents of a given lobby communicate with each other via shared memory channels.

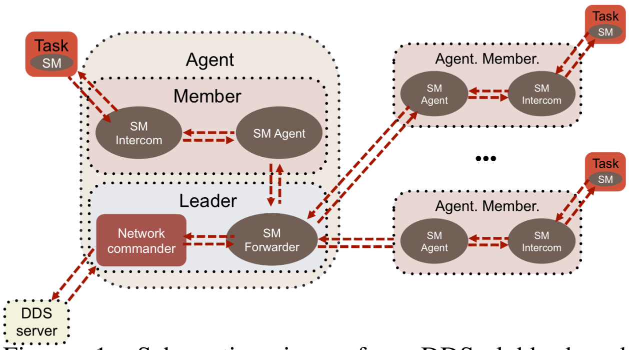

Figure 1: Schematic view of a DDS lobby-based deployment.

DDS session

We used to have only one main use case, which is Alice Online. For such a case it was enough to run one DDS session per user per host. Now other uses cases coming into the game. In offline analysis there are use cases, when a single user needs to run multiple different topologies hosting DDS commander on the same host. For the Grid and different RMS it might be useful to run DDS in a batch mode. For all these new use cases it is necessary to support multiple runtime topologies per user per host.

In order to cover these use cases a DDS session feature has been introduced. This new feature offers users a possibility to run multiple commanders on the same host. Each new commander instance creates a DDS session. Sessions are sandboxed and isolated, therefore can't disturb each other. Sessions can be operated (listed, cleaned, sorted, etc) using the new dds-session command.

References

[1] The Dynamic Deployment System (DDS), http://dds.gsi.de

[2] The BOOST C++ Libraries, http://boost.org

DOI:10.15120/GR-2018-1

New release v6.3 of data acquisition framework MBS

J. Adamczewski-Musch[1], Nikolaus Kurz[1], Sergei Linev[1,]

[1]GSI, Darmstadt, Germany.

Since November 1, 2017, the MBS version 6.3 has become the new production version [1]. MBS version 6.3 replaces the old production version 6.2. The new MBS and its commands can be used as before.

New features

MBS version 6.3 provides several new features and improvements at the GSI installations. These are described in the following.

New GSI servers for MBS with dedicated VLAN

At GSI a pair of new servers for the MBS systems is in operation since May 2017. It will provide the DHCP, TFTP and NFS services for the diskless MBS nodes. In addition to the NFS mount point of the MBS installation ("/mbs") and various user partitions, the server will also provide NFS nodes with software installations of dabc ("/dabc"), Go4 ("/analysis"), and of EPICS ("/epics") frameworks for suitable platforms. Together with the new server, all MBS nodes moved from GSI LAN into the Virtual Local Area Networks (VLAN) MBS-NETZ (mbs nodes for experiments) or MBS-NETZ-ACC (nodes for accelerator tasks).

Support of 64 bit Linux

The complete source code of the MBS framework has been remanufactured to work also on 64 bit Linux systems. This has been tested on several X86 PC nodes running with Debian 7 and Debian 9. As a benefit from the enhanced address space, a larger pipe memory can be set up on such systems.

Linux device driver software

The driver software for the PCIe optical receiver boards PEXOR and KINPEX has been newly implemented for x86 Linux platforms [2]. This consists in a new kernel module "mbspex" with corresponding C library. Additionally, a command line tool" gosipcmd" allows front-end configuration and controls from the system shell.

Front-end control GUIs

Several Graphical User Interfaces (GUI) applications have been developed to monitor and control the properties of different kinds of front-end boards at the GOSIP read-out chain. They are based on the Qt graphical library and are executed locally on the MBS readout nodes hosting the KINPEX board. Communication between such GUIs and the read-out slaves is provided by the mbspex device driver library safely concurrent to the MBS read-out. Currently GUIs are available for the proprietary GSI front-ends POLAND, NYXOR, FEBEX2 (TUM-addon), APFEL, and TAMEX2 (with PADI).

Remote control via DABC

In addition to the existing MBS status server socket, two new socket channels have been introduced to MBS for control with the software framework DABC [3]. They can be started optionally. Firstly they allow a remote steering of the MBS console from external scripts using the DABC "mbscmd" executable. Moreover, a DABC session connected to MBS in this way offers an HTTP server with a web browser GUI designed for generic MBS monitoring and control. Such DABC web server for MBS has been installed at all GSI MBS Linux nodes, and can be started by alias command "webmbs" [1].

White Rabbit support

The future FAIR general machine timing distribution will be based on the White Rabbit system [4]. The White Rabbit timing receiver (WRT) hardware PEXARIA (PCIe), EXPLODER5A (PCIe, USB), and VETAR2A (VME) is supported by MBS. WRTs provide a special "Time Latch Unit" (TLU) that can record the time stamp when an input signal changes, e.g. by the trigger signal. Driver software release "Cherry v4" of GSI-CSCO has been deployed on all MBS Linux platforms. Additionally, dedicated drivers for VMEbus on RIO4 and IPV systems have been developed. Especially for MBS, a "direct TLU access" mode is being tested to speed up the timestamp read-out compared with regular etherbone cycles.

New mass storage interface LTSM

A new API Lightweight Tivoli Storage Manager (LTSM) [5] is going to replace the RFIO interface to directly write DAQ data into the tape robot at GSI. MBS v6.3 still supports the RFIO protocol for local disk servers and for any existing GSI RFIO tape servers. Additionally, a special local RFIO server has been developed as a gateway to the LTSM archive [1]. This gateway application will be further tested and deployed at GSI in 2018.

References

[1] J. Adamczewski-Musch, N. Kurz, S. Linev: “GSI Data Acquisition System MBS Release Notes V6.3”, GSI, November 2017, https://www.gsi.de/mbs

[2] J.Adamczewski-Musch, N.Kurz, S.Linev: "MBSPEX and PEXORNET - Linux device drivers for PCIe Optical Receiver DAQ and control", IEEE TNS Vol 6-2 (February 2018), https://doi.org/10.1109/TNS.2017.2783043

[3] The DABC framework, http://dabc.gsi.de

[4] General Machine Timing System at FAIR wiki, https://www-acc.gsi.de/wiki/Timing/WebHome

[5] T. Stibor et al, LTSM source code, available: https://github.com/tstibor/ltsm/

 DOI:10.15120/GR-2018-1

Progress in FairDB development

D.Bertini[1] and Evgeny Lavrik[2]

[1]GSI, Darmstadt, Germany; [2]Universität Tübingen, Germany.

Introduction

FairDB[1] is a ROOT[2] based virtual database which allows to communicate and store data in different database management systems, such as PostgresSQL, MySQL, SQLite, based on the configuration. One of the primary use for it is FairROOT[3][parameter storage. FairDb is an insert only database, meaning there is no need to update the single entries and the whole history of the entries is available.

Data serialization

The base FairDb parameter classes have been expanded to support the data serialization in the JSON format. This allows the data exchange with non-ROOT environments such as LabVIEW[7] and web services. In addition to the existing data aggregation mechanism the introduction of the relational mechanism allows to establish one-to-one and one-to-many links between stored entries.

Database ROOT class generation

To improve the user experience, reduce the number of errors and further enhance the feature set of the FairDB the database class generation mechanism have been added [5]. It provides a web application based user interface to define the data format, which needs to be stored. Here, the user defines the classes, their properties and relations between classes. The classes are organized in projects, which can be loaded to and from the disk (Fig. 1).

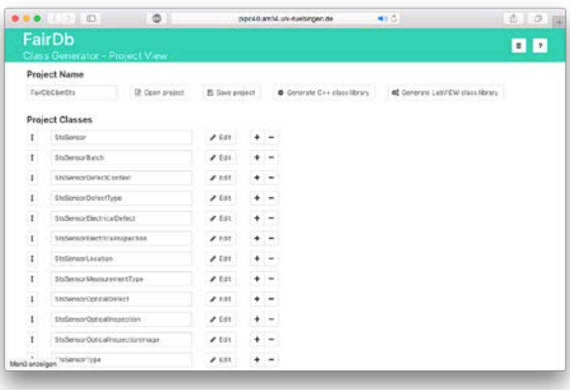

Figure 1: Graphical user interface to define the database classes organized together in projects.

Precise configuration of the properties such as their C++ type, database type, JSON type, default value etc. is available (Fig.2).

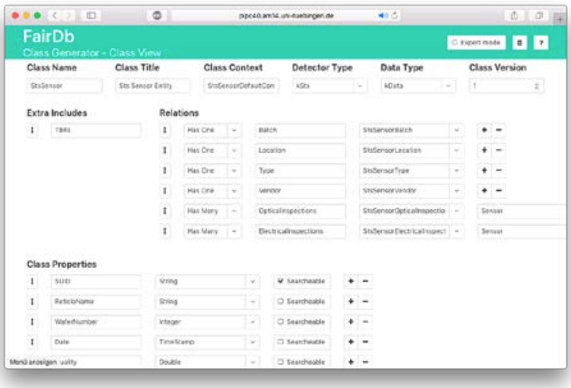

Figure 2: Graphical user interface to define the database class properties and relations.

After defining the class data, the user is offered to generate the class library, which is ready to be included into the FairROOT framework. Additionally the database configuration file and a template database data priming macro are included into the generated library.

For the detector groups using LabVIEW in their work, the generation of the LabVIEW class library is available.

Database content management system

Based on the user input for the class generation the content management system for the user data can be generated. This includes the RESTful web service, which communicates to the FairDB and serves the data in the JSON format for the consuming web application. The service provides the role-based data access control, requiring the user to authenticate before accessing the data. The secure HTTPS protocol is enforced for the data exchange between web service and web application. The web application itself allows the user to view, edit and add data to the database. The administrator's workplace allows managing users, who have access to the database, define their role and permissions to view and edit data. The content management system is generated based on the template and can be further expanded for functionality such as plotting the graphs based on the stored data. Visualization of the ROOT-native data such as TGeoVolume is available with JSROOT [6] framework.

Using FairDB: QA Data Scheme

The usage of the FairDB was covered in [4] and since then has been expanded to support the recent developments.

DOI:10.15120/GR-2018-1

In a framework of the STS sensor optical inspection project the relational database schema have been developed and used for the data export (Fig. 3). The data gathered for 25 sensors inspected have been exported into FairDB which resulted in more than 49000 records inserted and the SQLite database file size of 18,9 MB.

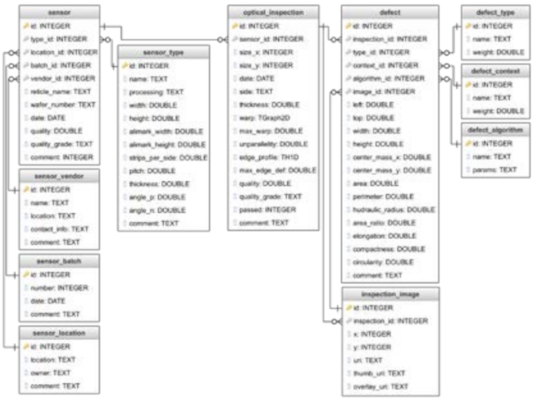

Figure 3: The database schema for the optical QA of the STS silicon sensors

The class generation described above was used to create the ROOT classes for data storage. The primary measurement data is obtained from LabVIEW program and stored in JSON format. A ROOT export macro was used to read the JSON data, deserialize it and store in the FairDB.

Using the generation of the database content management system the exported data was made available for the external users to be visualized and edited if allowed. The figure 4 (Fig. 4) shows the user workspace to view and edit the information about the sensor vendor.

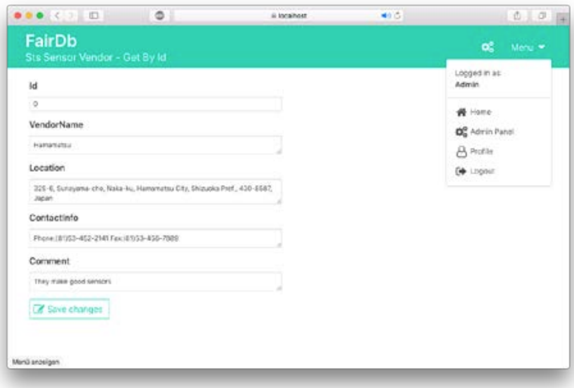

Figure 4: The generated graphical user interface to view the data of a data

Conclusion

FairDB virtual database provides flexible way to store and access detector specific data. The recent developments were targeted to enhance the user experience when defining the data to be stored. The generation of ROOT classes in a standard way not only simplifies the process but further improves the stability of the code as well.

The generation of the content management systems allows the ease of visualization and manipulation.

References

[1] D. Bertini. FairRoot Virtual Database (User Manual)
[2] R. Brun, F. Rademakers, P. Canal, I. Antcheva, D. Buskulic, O. Couet, A. and M. Gheata {\it ROOT User Guide} CERN, Geneva 2005
[3] The FAIR simulation and analysis framework 2008 J. Phys.: Conf. Ser. 119 032011
[4] D. Bertini {\em et al.}, CBM Progress Report 2015
[5] https://cbmgsi.githost.io/evgeny.lavrik/dbClassGen
[6] https://github.com/root-project/jsroot
[7] http://ni.com/labview

 DOI:10.15120/GR-2018-1

Status of the ALICE Tier2 Centre at GSI and first prototype of an ALICE Analysis Facility

K. Schwarz[1], *S.Fleischer*[1], *R.Grosso*[1], *J.Knedlik*[1], *P. Kramp*[1], *T.Kollegger*[1,]

[1]GSI, Darmstadt, Germany

This article describes the improvements implemented in 2017 in order to increase the reliability and performance of the ALICE Tier2 Centre at GSI as well as the setup of a first prototype of an ALICE Analysis Facility.

ALICE Tier2 centre at GSI and ALICE Grid in Germany

The ALICE Tier2 centre and the National Analysis Facility at GSI provide a computing infrastructure for the ALICE Grid and for the local usage of the German ALICE groups. The storage resources pledged to the global ALICE community (2300 TB) are provided via a Grid Storage Element which consists of a set of xrootd daemons in a redundant setup mode running on top of the Lustre file system. The main elements are two xrootd redirectors with a DNS alias as single point of entry in combination with three xrootd data servers as well as two xrootd forward proxy servers. The redirector of the GSI storage element is using the split directive of xrootd and redirects external clients to the external interfaces of the xrootd data server machines and internal clients to the internal interfaces which are directly connected to the local Infiniband Cluster. The xrootd forward proxy servers provide the possibility to Grid jobs running inside the protected HPC environment to read and write data from and to external data sources using the proxy interface. All ALICE Grid jobs running at GSI make use of the container technology Singularity. In this way the jobs can run smoothly in their standard Scientific Linux environment on top of the Debian based host system of the GSI HPC cluster. The complete setup is shown in fig. 1.

Throughout the year GSI participates in centrally managed ALICE Grid productions and data analysis activities, but also analysis jobs of individual users are running on the ALICE Tier2 centre. The overall share of successfully computed jobs in 2017 contributed by the German Grid sites, the GSI Tier2 centre and Forschungszentrum Karlsruhe (ALICE Tier1 centre) has been 10% of all ALICE Grid jobs worldwide. This corresponds well with the promised CPU resources for 2017: 28000 HEP-SPEC06 for GSI Tier2 (7% of the global Tier2 requirements) and 60500 HEP-SPEC06 for FZK (25% of the global Tier1 requirements).

References

[1] K. Schwarz et. al. First Challenges for the ALICE Tier2 Centre at GSI (PS05-4-466) Journal of Physics: Conference Series 331 (2011) 052018

[2] J. Knedlik, P. Kramp, Site specific XRootD related development, this Scientific Report

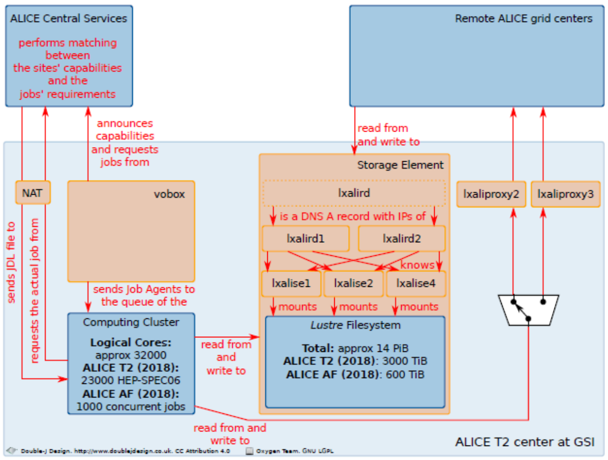

Figure 1: Setup of the ALICE tier centre at GSI

Prototype of an ALICE Analysis Facility

In LHC Run 3 ALICE data analysis will take place on a few dedicated Analysis Facilities of which GSI will become one of the first. On the storage element only analysis data sets (AODs) will be stored. A first prototype of an ALICE Analysis Facility has been set up at GSI based on the design of the current ALICE Tier2 centre, but optimised for I/O performance. Initial resources are 600 TB disk space and 1000 concurrent job slots taken from the Tier2 allocation. A first complete data set has been transferred and a first analysis train ran successfully.

XRootD Plugins

In order to optimise the current storage infrastructure, namely the access to Lustre through XRootD data servers, several XRootD Plugins have been developed. One solution provided is a server based Plugin which is able to redirect XRootD file operations directly to Lustre thus bypassing the need for Grid jobs running at GSI to double the network traffic by first communicating with the XrootD data servers. A second Plugin creates symlinks for data stored on Lustre providing logical file names as they are used in the ALICE Grid File Catalogue. A third solution provided is a Plugin which reports to the central ALICE Grid Monitoring the ALICE Grid quota on Lustre instead of the full Lustre capacity at GSI as it did before. All three Plugins described above are in operational mode at the ALICE Analysis Facility prototype at GSI.

DOI:10.15120/GR-2018-1

Status of the R3BRoot framework

D. Kresan[1], M. Al-Turany[1], V. Wagner[2], M. Heil[1], B. Löher[1,2], the R^{3}B collaboration, and the FairRoot group

[1]GSI, Darmstadt, Germany; [2]TU, Darmstadt, Germany

Reactions with Relativistic Radioactive Beams (R^{3}B) is an international collaboration [1] which will perform nuclear physics experiments at the future FAIR facility. The focus of these experimental studies is put on nuclear structure and dynamics of exotic nuclei far off stability, as well as astrophysical aspects and technical applications. Combination of different detection sub-systems and a sophisticated data acquisition require a dedicated software package to allow physicists both to simulate an experiment and to analyze experimental data. The R3BRoot software framework [2] was created for this purpose. It is based on FairRoot [3] – common software for FAIR experiments. A group of 17 developers is currently working on extending features of R3BRoot and including support for all planned R^{3}B detectors. The framework is written in C++, is ROOT based, compiled of approximately 35000 lines of code and can be deployed on Linux and macOS. The C++11 standard is supported. The code is stored and distributed via GitHub [4] with continuous integration workflow. Close to 40% code coverage with quantitative automatic tests allow to immediately detect possible violations of the program functionality and to results of numeric algorithms.

Data analysis

Several experiments, including the commissioning run of R^{3}B, will be performed fall of 2018 using beam from SIS18 at GSI. The largest effort is put into the development of algorithms for detector calibration up to so-called HIT-level: measurement of coordinates in cm (local frame), energy-loss in MeV and time in ns. Following detectors are currently calibrated in R3BRoot: start detector LOS, proton and gamma calorimeter CALIFA, neutron detector NeuLAND, heavy fragment tracker arm PSPx, TOFd, and proton ToF wall. The later analysis stage, which is called fragment tracker, should combine the single measurements and fit an ion trajectory in a non-homogeneous dipole field in order to determine mass and momentum of a reaction fragment. This measurement in combination with data from proton ToF wall and NeuLAND will give the possibility to calculate properties of incident reaction.

Another important tool, which is also being developed, is an online event-display to support near-line analysis during data taking. The online event display of R3BRoot will be a tool for monitoring of the detectors and for an immediate decision about the quality of the data. Several successful tests were already performed with cosmic rays.

Simulation

Simulation of the upcoming R^{3}B experiments is required for:

- detailed design of a setup,
- feasibility study of a measurement of an observable,
- development and testing of physics analysis algorithms,
- a better understanding of background effects, present in experimental data.

In order to develop an algorithm for fragment tracking and to compare simulated and experimental data on the HIT level, digitizer tasks have been implemented for TOFd and 3 tracking fiber detectors. A digitizer is an algorithm, which simulates the response of a detector and converts Monte Carlo results into detector-like signals. Mainly effects from geometry granularity, read-out scheme, and non-perfect resolution were included.

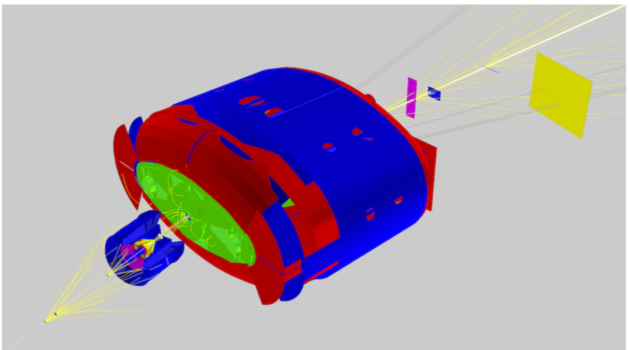

Figure 1: Visualisation of the R^{3}B detector geometry and particle trajectories using the simulation event display of FairRoot. Large volume in the middle – GLAD magnet. Beam direction from the bottom left corner.

Concerning the neutron measurement, a new algorithm based on statistical scoring is being developed and tested with simulated data. This new technique should have better performance with respect to separating classes of multi-neutron events.

The simulation code for CALIFA spectrometer is under construction to match detector response of the real-size prototype, which will be also tested during experiments end of 2018.

The development of the fragment tracker is ongoing. Currently, we have the tool for propagating a charged particle in the magnetic field of GLAD through the detector geometry and a machinery for the fitting of ideal measurements without taking into account energy-loss in a material. The first realistic version of the algorithm is expected mid of 2018.

Summary

Current status of software development in R^{3}B, presented in this report, is the result of coordination of activities within the international community, representing different detector working groups. R3BRoot is on a good track towards supporting experimental runs fall of 2018 / 2019

DOI:10.15120/GR-2018-1

and towards performing high-level physics analysis of accumulated data. Preliminary tests of the fragment tracker show feasibility of momentum reconstruction with resolution in the order of 10^{-3}.

References

[1] R^3B, https./www.gsi.de/r3b
[2] R3BRoot, http./www.r3broot.gsi.de
[3] M. Al-Turany et al., J. of Phys. Conf. Ser. 396 (2012) 022001
[4] R3BRoot code repository, https./github.com/R3B-RootGroup/R3BRoot.git

Ongoing site specific XRootD related development

J. Knedlik[1], *P. Kramp*[1]

[1]GSI, Darmstadt, Germany

1.1 Abstract

Operating an XRootD service, the established software standard for WAN data access in HEP and HENP. Accessing the scientific data, stored on-top of the HPC infrastructure at the ALICE Tier 2 centre and the ALICE Analysis Facility prototype at GSI, revealed multiple challenges and requirements. This article describes the current state of development for XRootD based solutions, especially XRootD client & server plug-ins.

1.2 ALICE Analysis Facility & XRootD

GSI is operating the only German ALICE Tier 2 centre and will operate one of the first ALICE Analysis Facilities(ALICE AF) for which a prototype is currently being set up. In this context, GSI will provide computing and storage resources to the ALICE community. In ALICE's AliEn Grid framework data is accessed through the XRootD protocol. XRootD enables the use of this data through a scalable federated storage system. At GSI, instead of operating XRootD servers with local storage, the shared HPC Lustre filesystem is used as storage backend of the XRootD servers. In AliEn, Grid jobs requiring the same data are preferably scheduled on the same site to lower the need for traffic between sites. This means, that local data is reused many times and therefore it is essential to optimize the I/O performance.

1.2.1 Improving the I/O performance at GSI using XRootD plug-ins

With the current storage infrastructure at GSI, namely the access to Lustre through the XRootD data servers, the following room for improvement has been identified: The three XRootD dataservers can provide limited I/O bandwidth and all data read locally via XRootD from Lustre needs to be sent over the network twice (Lustre to XrootD server & XrootD server to client), effectively doubling the network traffic for an I/O operation.

In addition, the need for additional XRootD dataservers to handle data is eliminated.

As an improvement to last years solution[1], an XRootD client plug-in, which had to be loaded by all clients running on the GSI hpc cluster, an XRootD redirector plug-in was developed.

A redirector server may load this plug-in in order to redirect clients to locally available files, if both client and redirection target are inside a private network, as this guarantees local availability of the required file at the Lustre filesystem. In order to allow redirection to a local file, additional changes needed to be implemented into the XRootD base code.

The cooperation with the XRootD core development team resulted in the integration of necessary client and server side changes in the XRootD base code. Since XRootD Version 4.8.0, a local redirection of a client by a redirector is possible with the use of the cms plug-in we developed. In addition, the plug-in is able to distinguish between new and old clients and will only redirect newer clients that have the capability to handle such a redirection to a local file.

1.3 XRootD Disk Caching Proxy for opportunistic resources

In cooperation with KIT, an infrastructure for the utilization of opportunistic resources such as clouds, has been developed for CMS. The idea is to build a virtual site inside the opportunistic resource. In order to minimize external I/O and to provide high data locality, an XRootD disk-caching-proxy is used.
In this setup, all clients access data through a redirector, which tells the client the location of the desired data. The redirector either directs the client towards the locally available shared filesystem, in case the data exists on it, or to the disk-caching-proxy. The disk-caching-proxy forwards the request to external sites and retrieves the data. In the meantime, the data is being cached on the shared filesystem for later use.
In case jobs working with the same data are bundled together, this infrastructure minimizes I/O and speeds up data accesses inside the virtual site.
This infrastructure relies on XRootD-plug-ins developed at GSI. One plug-in handles the referral to the redirector2 while the second plug-in handles the redirection to the proxy or shared file system3. A test setup has been deployed on the bwhpc4 cluster NEMO at Freiburg

1.4 Conclusion

In conclusion we have implemented XRootD plug-ins redirecting clients to data on Lustre which significantly improves the I/O performance. Last years proposed changes have been integrated into the XRootD main base by the XRootD development team. In addition, an XRootD infrastructure to utilize opportunistic resources has been developed.

References

[1] "GSI Scientific Report 2016", doi = 10.15120/GR-2017-1, http://dx.doi.org/10.15120/GR-2017-1

[2] https://github.com/pkramp/RedirPlugin/tree/kit-proj

[3] https://github.com/jknedlik/XrdProxyPrefix/tree/kit-proj

[4] http://www.bwhpc-c5.de/en/

 DOI:10.15120/GR-2018-1

Annealing studies of avalanche photodiodes irradiated with different γ-doses

R. Ganai[1]*, A. Wilms*[1]*, D. Scharnberg*[1]*, J. Bailey*[1]*, A. El Mosleh*[1]*, P. Wicke*[1]*, H. Al-Turany*[1]*, C. Warneke*[1]*.*

[1]GSI, Darmstadt, Germany.

Introduction

Avalanche **P**hoto**D**iodes (APDs) used in **H**igh **E**nergy **P**hysics (HEP) experiments are supposed to operate at very hard environmental conditions like high magnetic elds and high radiation doses for a long period of time. Hence the long term annealing behavior needs to be investigated. Preliminary results of an annealing period of ~7 days under applied bias voltage are presented in this report.

Annealing studies of APDs

Sixteen APDs were irradiated with di erent γ-doses of 37 Gy, 100 Gy, 200 Gy, 500 Gy, 1000 Gy, 1500 Gy, 2200 Gy and 2680 Gy. Eight of them were operated with bias voltage applied during irradiation and the others had no electrical contacts during irradiation. After irradiation, the APDs were annealed with a reverse bias voltage of 100 V applied for ~7 days at the Photo Sensor Laboratory (PSL), GSI. During annealing, the temperature was raised from room temperature to 80° C and was then kept constant. The current values for biased and unbiased APDs during irradiation were monitored during the whole annealing period and are shown in Figure 1 and Figure 2 respectively. The applied γ-dose values are stated in the included legends.

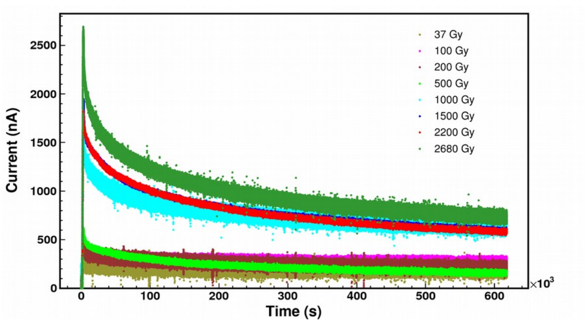

Figure 1: Annealing curves of eight di erent APDs irradiated with di erent γ-dose under the condition of applied bias voltage during irradiation.

In order to investigate the long term annealing behaviour of the APD current, the tail parts of the curves shown in Figure 1 and Figure 2 (in the time interval of $200*10^3$ s – $620*10^3$ s) were tted with a linear function. A comparison of the decrease in the rate of change of the leakage current values for all sixteen APDs is shown in Figure 3.

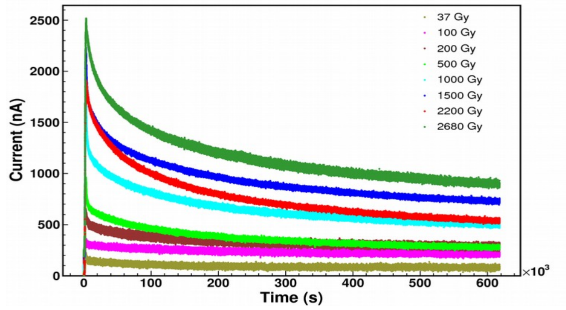

Figure 2: Annealing curves of eight di erent APDs irradiated with di erent γ-doses with no electrical connections during irradiation.

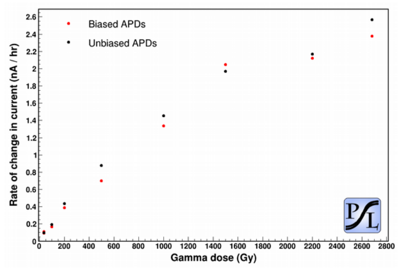

Figure 3: Comparison of the decrease in the rate of change of the leakage current for APDs irradiated under biased and unbiased conditions. The error bars are within the marker size.

Results and conclusion

A set of 16 APDs was annealed with a reverse bias voltage of 100 V after irradiation with di erent γ-doses. The annealing temperature was increased from room temperature to 80° C and was then kept constant. The decrease of the APD leakage current was studied, regarding the in uence of long term annealing e ects. The higher the γ-radiation dose was, the faster the leakage current decreases. This is consistent with the assumption of an exponential time dependence of trap recovery with a large time constant (~10^5 s). The unbiased APDs showed higher rate of decrease in current with respect to biased APDs except for a γ-dose of 1500 Gy which needs to be tested in more detail.

Acknowledgment

We sincerely acknowledge the help rendered by our colleagues at UGC-DAE Consortium for Scienti c Research, Kolkata Centre and Mr. Vinod Singh Negi, for their sincere e orts to irradiate the APDs with di erent γ-doses.

Development of a 128 channel ≤ 500 ps RMS TDC system for MBS DAQ

H. Heggen[1], S. Minami[1] and N. Kurz[1]

[1]Experimentelektronik – GSI, Darmstadt, Germany

Introduction

The precise measurement of time has become one of if not the most important tool in particle identification experiments which usually rely on time-of-flight, drift time and/or pulse-width measurements. Most of the in-house time-to-digital converter (TDC) developments of recent years were aimed at achieving highest precision (down to 7 ps) [1,2]. However, there are many applications that do not require such high precision and which would benefit from a system with moderate precision but higher channel density and lower channel price. Here we report on the development of a TDC system with 128 channels and a precision better than 500 ps for the MBS (Multi Branch System) data acquisition framework [3,4,5].

Prototype hardware

The prototype hardware (see figure 1) features 128 differential input channels for LVDS signals. The channels are distributed over 8 board-to-cable connectors to allow the connection of 16-channel front-end cards via flexible flat cables. A first 16-channel analog front-end based on the PADI (preamplifier-discriminator) ASIC [6] has also been developed (see figure 1). The front-end features a standard 2.54 mm square socket for signal input making it easy to adapt to various detector systems. The PADI front-end is designed for input signals between a few mV and ~200 mV. Larger signals can be accommodated by equipping an attenuator network at every input.

Results & Outlook

Laboratory tests with the described prototype system already yield an average precision well below 400 ps RMS for both, channel-to-channel and pulse-width measurements. The TDC is dead-time free and currently has a double-pulse resolution of ~10 ns. However, development of the FPGA circuit implementation is still ongoing [7] and these figures are not final yet.

The prototype system has also just been deployed in a first beam time at KVI-CART, reading out a total of 1024 channels of multi-anode photomultiplier tubes to test various fibre detector prototypes for the NUSTAR-R3B experiment. The tests were very successful for both, detectors and electronics and preparations have been started to equip the R3B experiment with 2048 readout channels for the 2018 beam time.

For the coming years, more than 6000 readout channels are foreseen at the R3B experiment and the development of dedicated hardware has been started. Various other NUSTAR experiments and Super-FRS particle identification are also considering the use of this new TDC system, potentially adding up to another 25 000 channels in the long-term.

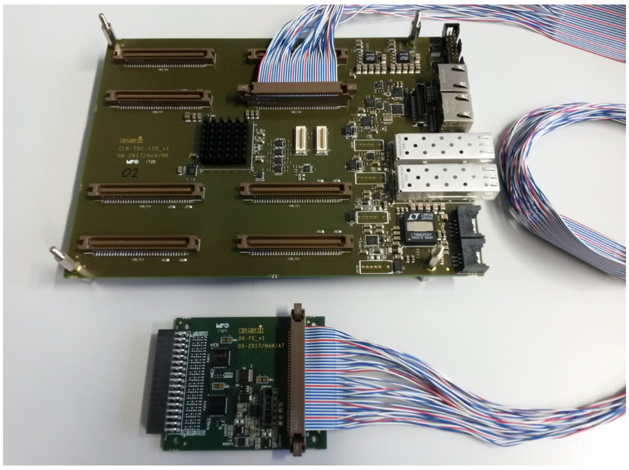

Figure 1: The prototype hardware consists of the CLK-TDC-128 motherboard with 128 channels (LVDS inputs) tailored to read out up to eight 16-channel analog front-ends based on the PADI ASIC [3].

References

[1] C. Ugur et al., "264 Channel TDC Platform applying 65 channel high precision (7.2 psRMS) FPGA based TDCs", IEEE Nordic Mediterranean Workshop on FPGA based TDCs (NoMe TDC), 2013, p. 1-5;

[2] C. Ugur et al., "A novel approach for pulse width measurements with a high precision (8 ps RMS) TDC in an FPGA", Topical Workshop on Electronics for Particle Physics (TWEPP), 2016.

[3] H.G. Essel and N. Kurz, "The General Purpose Data Acquisition System MBS", IEEE Trans. Nucl. Sci., vol. 47, no. 2, 2000, p. 337-339, www.gsi.de/mbs;

[4] J. Adamczewski-Musch et al., "MBSPEX and PEXORNET-Linux Device Drivers for PCIe Optical Receiver DAQ and Control", IEEE Trans. Nucl. Sci., vol.65, no. 2, 2018, p. 788-794;

[5] S. Minami et al., "Design and Implementation of a Data Transfer Protocol Via Optical Fiber", IEEE Trans. Nucl. Sci., vol. 58, no. 4, 2011, p. 1815-1819.

[6] M. Ciobanu, "PADI, an Ultrafast Preamplifier-Discriminator ASIC for Time-of-Flight Measurements", IEEE Transactions on Nuclear Science, vol. 61, no. 2, 2014, p. 1015-1023.

[7] S. Minami et al., "An FPGA implementation of a 128-channel sub-nanosecond time-to-digital converter", in this report.

Experiment beamline: FRS / R3B
Experiment collaboration: NUSTAR-R3B / NUSTAR-SuperFRS-Experiments
Accelerator infrastructure: Super-FRS

 DOI:10.15120/GR-2018-1

An FPGA implementation of a 128-channel sub-nanosecond time-to-digital converter

S. Minami[1], H. Heggen[1] and N. Kurz[1]

[1]GSI, Darmstadt, Germany

There has been demand for the development of a time-to-digital converter (TDC) which has moderate time resolution, better than 500 ps, but has high channel density to be cost effective. The technique to realize field programmable gate array (FPGA) based TDCs have long been studied by many former works [1-3]. The easiest way to implement a TDC in a FPGA is to sample an input signal with a flip-flop operated by a clock with period $T_{clk.}$, resulting in a TDC bin width of T_{clk}. In order to achieve better precision than the method limited by maximum clock frequency of FPGAs typically around 500 MHz, we have adopted a method sampling the input signal with a set of several phase-shifted clocks of the same frequency. In the ideal case that all clocks are distributed without delay and skew, this results in a TDC bin width of T_{clk}/N_{clk} where N_{clk} is the number of shifted clocks. The method consumes low amount of logic resources, allowing the integration of more channels per FPGA [3].

In recent years, various front-end electronics with FPGAs and the small form-factor pluggable (SFP) transceivers have been developed to coop with our data acquisition (DAQ) framework, the multi-branch-system (MBS), supporting readout by a standard PC equipped with a PCIe card via our custom data transfer protocol GOSIP [4]. We started the implementation of a TDC on one of those existing devices with an XC7K160T, the second smallest FPGA in Kintex-7 family, classified as the best price/performance/watt at 28 nm[5].

The maximum frequency of the FPGA is 400 MHz, however we started with a clock frequency of 250 MHz and 8 phase-shifted clocks created by one Mixed-Mode Clock Manager (MMCM) module just to achieve the targeted precision of $\leq$500 ps [5]. As to the number of input channels, 128 differential channels were implemented since that is close to the limitation imposed by the number of IOs of the FPGA. A 12-bit coarse time counter operated by 250 MHz clock enables a maximum trigger window of 16 µs. Ones among the 8 flip-flops for fine time determination are summed and converted into 4-bit-wide data. A fine time and a coarse time for leading edges and trailing edges of a single input are combined and stored in a ring-buffer with a depth of 1024. At the very moment a pre-set trigger delay time has passed after accepting a trigger signal, the data in the ring-buffer are examined on whether they are within a pre-set time window relative to the trigger time. Only valid data are recorded in the second buffer of the same size. After parallel processing of 129 channels including the trigger signal, the data are merged into one of two output buffers with a size of 16 Kbytes connected to the GOSIP readout module. The implemented TDC utilized only 24 % of the FPGA logic resources as listed in Table 1.

Resource	Used	Available	Utilization
Slice LUT	24324	101400	24 %
Slice Flip-Flop	46664	202800	23 %
Block RAM	275	325	85 %
MMCM	2	8	25 %
IO	384	400	96 %

Table 1: Resource utilization of the FPGA TDC[5]

A new electronics dedicated TDC applications, CLK-TDC-128, has been designed and produced [6]. The precision of the FPGA TDC on the new electronics was evaluated by measuring time differences between 2 channels using a LVDS fan-out board. The single channel RMS resolutions were demonstrated to be better than 400 ps for all the channels.

References

[1] J. Kalisz, “Review of methods for time interval measurements with picosecond resolution”, Metrologia 41, 2004, p.17-32

[2] C. Ugur et al. "264 Channel TDC Platform applying 65 channel high precision (7.2 psRMS) FPGA based TDCs," IEEE Nordic Mediterranean Workshop on FPGA based TDCs, (NoMe TDC) 2013 pp.1-5

[3] M. Büchele et al., “The GANDALF 128-Channel Time-to-Digital Converter”, Physics Procedia 37, 2012, p. 1827-1834; A. Balla et al., “The characterization and application of a low resource FPGA-based time to digital converter”, Nucl. Instr. Meth. Phys. Res. A 739, 2014, p. 75-82

[4] H.G. Essel and N. Kurz, “The General Purpose Data Acquisition System MBS,” IEEE Trans. Nucl. Sci., vol. 47, no. 2, 2000, p. 337-339; J. Adamczewski-Musch et al., “MBSPEX and PEXORNET-Linux Device Drivers for PCIe Optical Receiver DAQ and Control” IEEE Trans. Nucl. Sci., vol.65, no. 2, 2018 p. 788-794; S. Minami et al., “Design and Implementation of a Data Transfer Protocol Via Optical Fiber” IEEE Trans. Nucl. Sci., vol. 58, no. 4, 2011, p. 1815-1819

[5] Xlinx Inc. 7-Series FPGAs Data Sheet: Overview DS180 2018; 7-Series FPGAs Cloking Resources User Guide UG472 2017

[6] H. Heggen et al., “'Development of a 128 channel ≤ 500 ps RMS TDC system for MBS DAQ” in these reports

DOI:10.15120/GR-2018-1

Monte Carlo simulations in FairMQ

R. Karabowicz[1], M. Al-Turany[1], D.Klein[1], T.Kolleger[1], D.Kresan[1], A.Lebedev[1], A. Manafov[1], A. Rybalchenko[1], F. Uhlig[1]

[1]GSI, Darmstadt, Germany

The FairMQ framework developed currently by the FairRoot group allows users reliable and transparent data exchange between devices – computer processes created by the user. The following paper summarizes our effort to run Monte Carlo simulations in separate devices parallelly and store or analyse created data.

Simulation parallelization

The Monte Carlo simulations are arguably one of the most time and resource consuming parts of the computing aspects of the modern experiments. Many efforts are committed to facilitate the costly simulations, and these can be grouped by the level of parallelization applied. The most general strategy uses multi-processor computer farms to run individual simulations that store output data in separate output files. Similar approach was adopted by the GEANT4 group, that uses multithreading to delegate simulation of individual events to different threads. Another level of parallelism is to assign transport of single particles to separate threads within one process. In this report the first of the described approaches has been applied.

FairMQ simulation example

As a base for this work a running FairMQ example described in [1] has been used. There, a set of devices to read, analyse and store FairRoot compliant data has been described. Two of these devices could be reused in the simulation example, namely the FairMQEx9TaskProcessor to process data, and the FairMQEx9Sink to store the output data in a ROOT file. The ParameterMQServer had to be greatly extended, its purpose to read parameter files and to provide parameters needed to reconstruct the data has been extended by the ability to receive parameters and to store them in output files. A completely new device has been introduced, a FairMQSimDevice, to perform Monte Carlo simulations using standard FairRunSim class, with user's set of detectors specified in the running executable.

The example topology is presented in the Figure 1. Each box in the figure represents a running device. The data flows from left to right through channels, drawn as lines. The leftmost devices, the FairMQSimDevices, run separate simulations of the same detector set. It should be stressed that it is possible to start any number (limited by the available resources) of these devices, thus increasing the simulation event rate. In the shown example, the data is sent straight to one of the FairMQEx9Sinks, where it is stored directly in the ROOT file. Furthermore, the same data is sent over to FairMQEx9TaskProcessor, which processes the data in the way specified by the user task attached to the processor. Such processed data is subsequently stored in the ROOT file by the second FairMQEx9Sink. To perform more complicated tasks, the user may create any other path for the data to be processed, implementing layers of desired TaskProcessors.

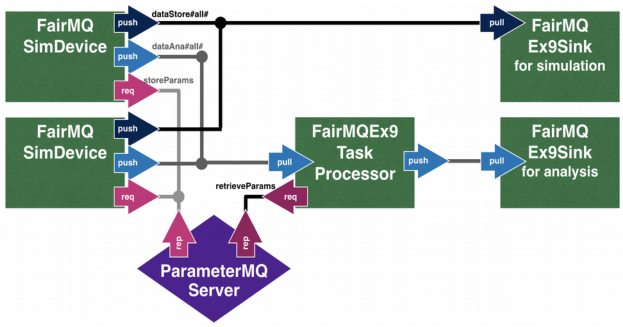

Figure 1: Schematic view of the running topology.

In the bottom of the figure the ParameterMQServer device is presented. It connects to the simulation devices, which use the server to store the simulation parameters. After receiving any of the parameters, the server automatically saves them in a file specified by the user. The other connection is reserved for the processing devices, which need simulation parameters to process data. Besides, the ParameterMQServer may read and distribute parameters stored in ASCII or ROOT files.

FairSink

It is worthwhile to mention, that the FairMQSimDevices do not themselves write any data to disk. To allow this, the FairRootManager has been redesigned in a way similar to that presented in [2]. Namely, the data storing functionality has been moved to a separate abstract class, called FairSink. The default FairRoot macros store data in the output ROOT files using FairRootFileSink implementation. In the case of FairMQ, the produced date is sent via FairMQSimDevice, which as well derives from FairSink.

Summary

This paper presents the example of running the Monte Carlo simulations in the FairMQ framework. The data from simulations running parallelly on different processes (even on different computers) may be sent to a device capable of either storing the data or processing it. These building blocks may be used by users as a base for creating more sophisticated topologies.

References

[1] R. Karabowicz et al, "Message based reconstruction example in FairRoot", GSI Scientific Report 2016.

[2] R. Karabowicz et al, "Redesign of the FairRootManager", GSI Scientific Report 2014.

DOI:10.15120/GR-2018-1

Advancing shared memory transport in ALFA

A. Rybalchenko[1], M. Al-Turany[1], D. Klein[1],

[1]GSI, Darmstadt, Germany;

The ALFA framework [1] is a common ALICE/FAIR software layer that offers a platform for simulation, reconstruction and analysis for particle physics experiments. In addition to standard services provided for simulation and reconstruction, such as event generation, magnetic field and so on, ALFA also provides tools for inter-node and inter-process data transport, configuration and deployment. The FairMQ component of ALFA [2] provides building blocks for creating processes that communicate between each other via message passing. It offers an abstract interface with different implementations of different transport technologies. This report presents the most recent developments and improvements to the FairMQ transport layer with the focus on the shared memory transport.

Shared memory transport in FairMQ

The original implementation of FairMQ used ZeroMQ library [3] for transport of the data via network or inter-process communication. It was quickly realized that in order to stay future-proof, the transport details need to be hidden from the user behind an abstract interface. Since then two new transports have been introduced (and fourth is under development) that the user can switch to via a simple configuration change, without having to modify any code. The new shared memory transport [4] builds upon Boost.Interprocess library [5] in connection with ZeroMQ to offer an efficient and zero-copy transport implementation between processes on the same node.

Unmanaged region

The shared memory transport, as all other transports in FairMQ, hides all memory allocation details from the user, with the aim of ease of use. The user can simply request memory of certain size, which, after filling, will be transferred to further processes. However, in some special cases the user needs to control the memory layout and allocation in much greater detail. Example for this is a detector readout that requires certain memory layout restrictions and optimizations. Ideally, in such a use case the data from this carefully controlled memory should reach all of its receivers with no or minimal copies. This becomes particularly important for shared memory, where no copy is the main goal of the transport.

To support this a feature called Unmanaged Region has been introduced to FairMQ. It allows to define a custom memory region, which is under full control of the user. The region is created using the transport methods provided by FairMQ, therefore ensuring maximum efficiency and no unnecessary copies. The user can then give FairMQ subsets of this region, which will be sent as messages to interested processes. In case of shared memory, no copy of the data is involved. The receiver will see the message as a regular message and does not need any special knowledge of the region layout. Furthermore, the sender is notified whenever a message has been processed by the receivers and the transport, so that it can then cleanup and reuse the parts of the region.

Automated cleanup

All the shared memory creation and destruction in FairMQ is done behind the scenes. Because shared memory can outlive the process, a proper cleanup needs to be done to avoid polluting the system with unused memory. This is done automatically by the transport implementation by monitoring shared memory usage and cleaning up when all users have finished their work. Even for the cases when processes crash, the memory will be cleaned up by an automated and independent monitor process. The cleanup tools also offer monitoring information on the shared memory.

Flexible message creation

The most efficient way to prepare messages for transfer in FairMQ is by requesting messages of certain size and filling them with data. This avoid unnecessary copies of user buffers to transport buffers. However, certain algorithms, such as compression or reconstruction may not know the size of their data in advance. To make sure they also can be handled in an efficient way, FairMQ now provides a mechanism to request an upper bound size buffer from the transport, which can be efficiently shrinked after filling by giving the framework the used size. For shared memory this allows rapid reuse of unused memory, and for other transports this reduces the size of the transfers and reduces the amount of meta information to include with each message.

References

[1] M. Al-Turany et al., “ALFA: A new Framework for ALICE and FAIR experiments”, GSI Scientific Report 2013

[2] A. Rybalchenko, and M. Al-Turany, “Streaming data processing with FairMQ”, GSI Scientific Report 2013

[3] ZeroMQ: http://zeromq.org/. Last visit 15.03.2018

[4] A. Rybalchenko, and M. Al-Turany, “FairMQ status”, GSI Scientific Report 2016

[5] Boost.Interprocess: http://www.boost.org/doc/libs/1_66_0/doc/html/interprocess.html Last visit 15.03.2018

Experiment beamline: none
Experiment collaboration: none
Experiment proposal: none
Accelerator infrastructure: none
PSP codes: none
Grants: none
Strategic university co-operation with: none

DOI:10.15120/GR-2018-1