Acta Polytechnica **53**(4):338–343, 2013

DEVELOPING CONTROL AND MONITORING SOFTWARE FOR THE DATA ACQUISITION SYSTEM OF THE COMPASS EXPERIMENT AT CERN

Martin Bodlák[a], Vladimír Jarý[a,*], Igor Konorov[b], Alexander Mann[b], Josef Nový[a], Severine Paul[b], Miroslav Virius[a]

[a] *Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Břehová 7, 115 19 Prague 1, Czech Republic*

[b] *Physik Department, Technische Universität München, James-Franck-Str. 1, 85748 Garching, Germany*

[*] corresponding author: `Vladimir.Jary@cern.ch`

Abstract. This paper focuses on the analysis, design and development of software for the new data acquisition system of the COMPASS experiment at CERN. In this system, the data flow is controlled by custom hardware; the software will therefore be used only for run control and for monitoring. The requirements on the software have been analyzed, and the functionality of the system has been defined. The system consists of several distributed nodes; communication between the nodes is based on a custom protocol and a DIM library. A minimal version of the system has already been implemented. Preliminary results of performance and stability tests have shown that the system fulfills the defined requirements, and is stable. In the next phase of development, the system will be tested on the real hardware. It is expected that the system will be ready for deployment in 2014.

Keywords: data acquisition, remote control, monitoring, COMPASS.

1. Introduction

Modern high energy physics experiments depend strongly on the computer systems that are used for the simulations, data acquisition, data storage, and data analysis. This paper focuses on the software development for a new data acquisition system for the COMPASS experiment. First, the experiment is briefly introduced. Then the existing DAQ system, based on the ALICE DATE package, is explained, and the problems with it that triggered the development of the new systems are analyzed. Next, a new system featuring the hardware controlled data flow is presented and the results of a requirements analysis are summarized. Then, a proposal for a minimal version of the software that fulfills these requirements is presented. The first results of performance and scalability tests are then summarized. Finally, the plans for future development are presented.

2. The COMPASS experiment

COMPASS[1] COMPASS is a high-energy fixed-target experiment situated at the Super Proton Synchrotron at CERN, built for studying the gluon and quark structure and for spectroscopy of hadrons using high intensity muon and hadron beams [1]. The scientific program was approved by the CERN scientific council in 1997; it includes research on hadron structure and spectroscopy with high-energy hadron and muon beams. After several years of preparation and commissioning, data gathering began in 2002.

The experiment is currently entering its second phase, known as COMPASS-II, which covers studies of Primakoff scattering, the Drell-Yan effect, and generalized parton distributions [2]

3. Existing DAQ architecture

The COMPASS spectrometer is composed of detectors that are used to track and identify particles and to measure energies of particles. The data acquisition (DAQ) system is used to read out data from detectors, to assemble full events from fragments from different detector channels, and to store events into permanent storage. The DAQ system also provides control and monitoring.

The COMPASS DAQ system consists of several layers [3]. The front-end electronics that form the lowest layer continuously preprocess and digitize analog data from the detectors. There are approximately 250000 detector channels. Data from multiple channels is read out and assembled by the concentrator modules called CATCH[2] and GeSiCA[3]. These modules receive the signals from the time and trigger system; when the trigger signal arrives, the readout is performed. The subevent is created by adding the time stamp and the event identification to the data. The subevents are transferred using the optical bus S-Link to the readout buffers that form the following layer of the

[1]COMPASS is the acronym for the Common Muon and Proton Apparatus for the Structure and Spectroscopy

[2]COMPASS Accumulate, Transfer, and Control Hardware

[3]GEM and Silicon Control and Acquisition module

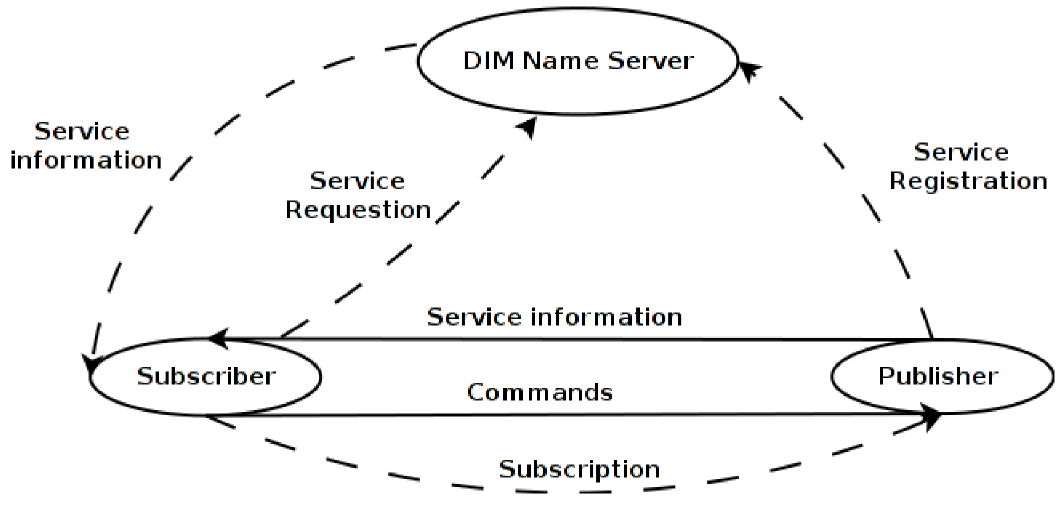

Figure 1. DIM Name Server.

architecture. Readout buffers are standard servers equipped with custom PCI cards called spillbuffers. Spillbuffers are used for buffering subevents, which allows the load to be distributed across the full cycle of the SPS accelerator. Finally, the subevents are sent over the Gigabit Ethernet to the event builders that assemble full events and transfer them to permanent tape storage. The remaining CPU power of the event building servers is used for data quality monitoring and for filtering purposes.

The data acquisition is powered by the DATE software [4], which has been adopted from the ALICE experiment. This package is designed to perform data acquisition tasks in a distributed environment; each node must be powered by Intel-compatible hardware and a GNU/Linux operating system. DATE is a very flexible and scalable system – at the ALICE experiment, it is deployed on hundreds of nodes, but it can also be used in a small laboratory experiment on a single node. From the functionality point of view, the DATE package provides data flow control, event building, data quality monitoring, event sampling, information reporting, interactive configuration, and run control.

4. Motivation for upgrade

During the first year of data taking (i.e. 2002), the system collected 200 TB of data [5], and this amount rose almost 10 times to 2 PB in 2010. The requirements on the DAQ system increase as the trigger rate and the beam intensity increase, and the existing system has already experienced performance problems and increased dead time. Additionally, as the hardware gets older, its failure rate also increases.

One possible solution is to upgrade the hardware of the event builders and the readout buffers. Unfortunately, the spill buffer cards in the readout buffers are based on the deprecated PCI technology. It would therefore be necessary to develop and produce a PCI Express version of these cards. However, no software development (with the exception of the kernel driver for the new spill buffer card) would be necessary in this scenario.

Another scenario proposes replacing the event building network with custom hardware based on FPGA programmable circuits[4] [6]. This hardware would perform the readout and the event building, so the software would be used only for control and monitoring. This system would consist of fewer components, so it should be more reliable and easier to maintain. As an additional benefit, the existing readout buffers and event builders could be used as an online filtering farm.

5. Requirements on the new software

The control and monitoring software is to be deployed in a distributed heterogeneous environment. Some nodes (e.g. the user interface application) will be installed on standard Intel-compatible hardware, but the control and the monitoring nodes will be deployed on custom cards with softcore processors. We have evaluated the possibility of using the DATE package on the new DAQ architecture; unfortunately this idea has been rejected – mainly because DATE requires Intel-compatible hardware and, moreover, the system is too complex.

However, the DATE data format must remain unchanged in order to make the new architecture compatible with the tools for the physical analysis. Additionally, some DATE components, e.g. the online filter, could be reused. In order to retain compatibility with the detector control system, the DIM library must be used for network communication.

Moreover, the system must support the remote control, and multiple users should be able to access the system. However, only one user can take the control over the system at any time. The other users can monitor the behavior of the system.

[4]Field Programmable Gate Array

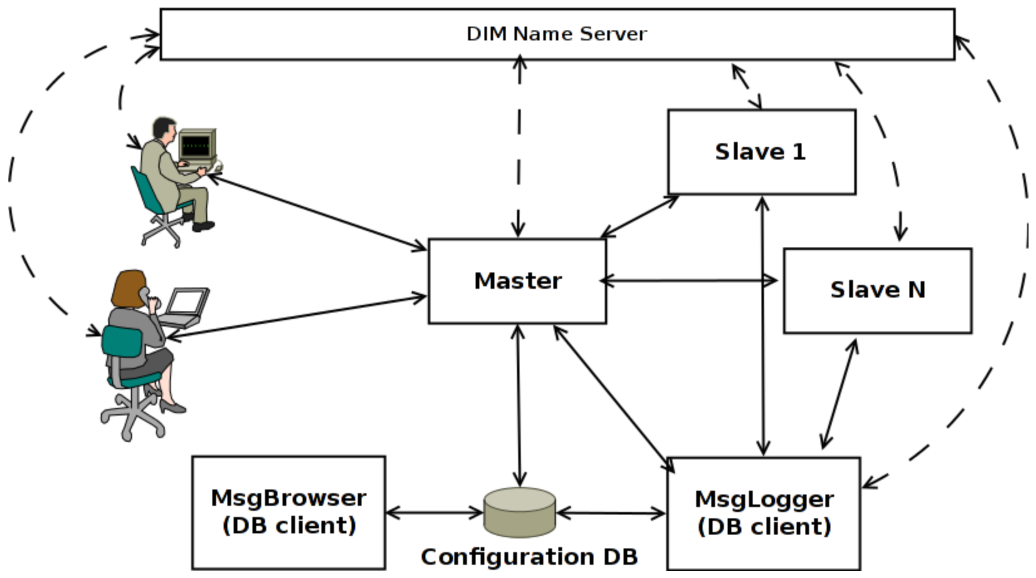

Figure 2. Software layers.

Data acquisition does not require real time operation, so it is not necessary to depend on the real time operating system and special libraries.

6. DIM Library

DIM (Distributed Information Management) is a software package that provides asynchronous, one-to-many communication in a heterogeneous network environment [7].

The library is based on the TCP/IP protocols; the client-server paradigm is extended by the concept of a DIM name server (DNS). In order to publish an information service or a command, a server (publisher) must register it at the name server. When a client (subscriber) wishes to subscribe to some service, it asks the name server which server has published that service. The communication with the name server is transparently provided by the library; the user has only to provide the location of the DNS (usually by exporting an environmental variable).

DIM is a C library, but interfaces to C++, Java, and Python languages also exist. The performance of the library was measured for different message sizes, and the C++ and Java interfaces were compared. We found that, for larger messages, the DIM library can saturate the network [8]. The Java version of the library calls the native C code through the Java Native Interfaces. The performance hit caused by JNI calls is about 20 % for smaller messages; for larger messages, the performance hit can be neglected.

Additionally, it has been found that the Java version of the DIM library is still incomplete. It was therefore decided to focus on C++ and to abandon Java. However, the Python language can be used for some auxiliary tasks, such as waking up the slaves.

7. Overview of the software architecture

Using the results of the requirements analysis, we designed the structure of the control and monitoring system. The system is deployed on several distributed nodes. Communication between nodes is based on the DIM package. The nodes can be divided into several categories, according to their purpose. According to Figure 2, one node acts as the master. The master node receives commands from applications that provide the graphical user interface, and forwards these commands to the slave nodes. In turn, the slave nodes generate monitoring data and confirmation messages that are sent back to the master. At one time, multiple user interface applications can receive data from the master. However, only one instance can issue commands. In addition, communication between the master node and the user interface is based on the DIM library, which means that remote control is possible.

The configuration of the system is stored in the online MySQL database – the MySQL database was chosen for its compatibility with the existing DAQ system. This configuration includes the lists of slave nodes for different scenarios, e.g. calibrating or data taking. When the system starts up, the master loads the appropriate list of slaves from the database, it connects to them (via SSH) and wakes up the slave processes. The system configuration is propagated from the master to all slaves using the DIM services. In this way, only the master process needs access to the database.

One node called Message Logger is used to collect messages generated during important events (messages of an informative character, e.g. change of the current state of a node) or unexpected events (i.e. errors) produced on other nodes. These messages are

Header		
1. Data size	4 bytes	Size of data in 32b words (= header_size+body_size+trailer_size)
2. Version	4 bytes	Version of data protocol
3. Sender ID	4 bytes	Unique ID of message's sender (from DB)
4. Message number	4 bytes	Number of message
5. Receiver ID	4 bytes	Unique ID of message's receiver (0=multicast) (from DB)
6. Message ID	4 bytes	Message ID
7. Time	4 bytes	Time stamp 1/2
8. Time	4 bytes	Time stamp 2/2
Body		
9. Body	0–4N bytes	Body of message (can be empty)
Trailer		
10. Reserved	4 bytes	0x00000000
11. Reserved	4 bytes	0x00000000
12. Message number	4 bytes	Number of message (the same as in header of message)
13. Message number	4 bytes	

Table 1. Transport protocol.

temporarily stored in a memory buffer, and under specified conditions (i.e. the number of messages in the buffer or the time elapsed since the first message stored in a buffer) are stored in the MySQL database. This message logger is de facto a DIM server that receives messages sent by other nodes (see Figure 2). A message is sent directly to the DIM server as continuous text, and is parsed on the server side according to the log protocol.

Messages already stored in the database can be viewed using a Message Browser application. Message Logger and Message Browser replace the functionality of the infoLogger and the infoBrowser applications from the DATE package.

It was decided to use the QT framework for implementing the slave, the master, and the GUI parts. QT extends the object model of C++ language, and it also provides a large class library that covers graphical components (widgets), networking, database access, multithreading, or data manipulation. Additionally, the framework supports all the major platforms - Windows, MacOS, and GNU/Linux with X11. The slave and the master were successfully tested on QT version 4.2.1 and the GUI on version 4.6.1. The slave includes both the DIM server and DIM client parts. For all outgoing communication, the slave uses only DIM services. However, for incoming communication it uses DIM commands and DIM services. The Master is also a combination of a DIM server and a DIM client. It uses DIM services for regular, more frequent or multicast messages, and DIM commands for precise sending of messages to control the state of a single node. GUI has only the client part of the DIM library, so it communicates by sending commands to the master and receiving status messages from master's services.

8. Transport protocol overview

We proposed and implemented a custom protocol that is used for exchanging information between nodes. This protocol defines a common frame for transporting messages and commands. This provides easier information manipulation in them. A precise description of this protocol is summarized in Table 1. The program uses the QByteArray class for assembling the frames

9. Performance tests

The architecture described above was thoroughly tested during the winter shutdown of the experiment. The DIM name server, the master and the slave processes were deployed on the existing event builders located in the COMPASS experimental hall; the message logger, the database server, and the graphical user interface were deployed on computers in the remote control room [9]. Communication between all nodes is provided by the Gigabit Ethernet, which limits the theoretical transfer speed with a value of 128 MB/s.

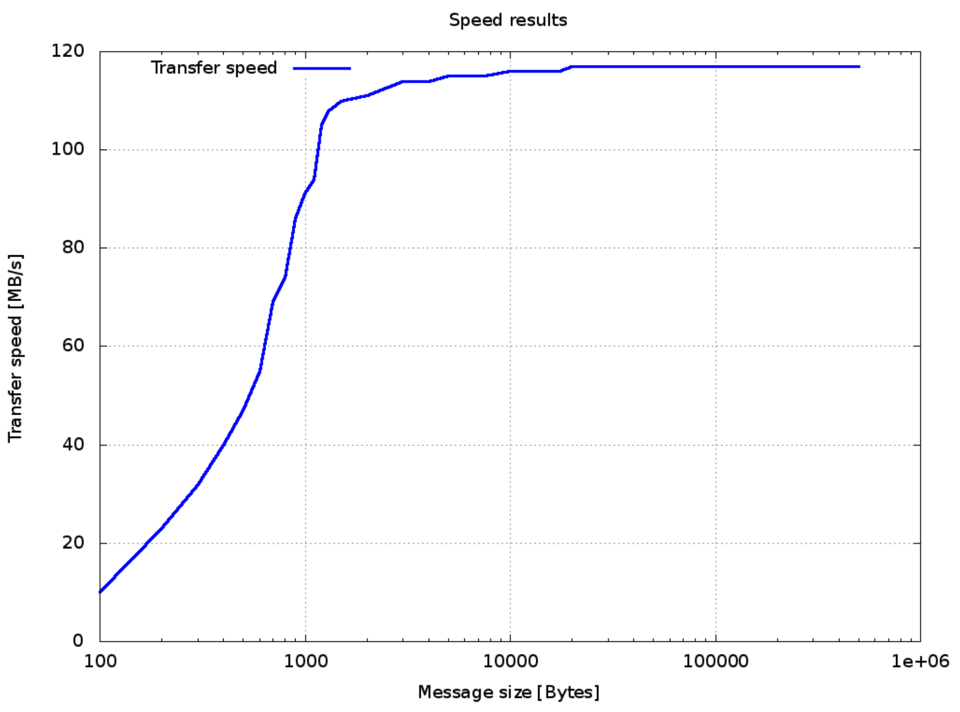

FIGURE 3. Transfer speed.

Several tests were performed for different message sizes and for different number of slaves. As shown in Figure 3, the system can almost saturate the network for messages larger than 1 kB. For smaller messages, the performance bottleneck is caused by the communication with the DIM name server. At these speeds, the system is able to exchange approximately 90000 1 kB messages per second. As real time operation is not required, this result is promising – it is expected that the final system will require a transfer rate of thousands of messages per second at most.

The stability of the system in time was also evaluated; the results are summarized in Figure 4. The system exchanged messages continuously between the master and 10 slave nodes over a period of 20 hours. During this period, no memory leaks were detected, and the transfer speeds remained constant. The observed peaks were probably caused by the time synchronization with the NTP server. However, this issue is not yet fully comprehended, and it requires further investigation and testing.

10. CONCLUSION AND OUTLOOK

The existing data acquisition system of the COMPASS experiment has been analyzed. The readout part of the data acquisition is based on deprecated PCI technology, so two different upgrade scenarios were considered. The first scenario involves developing a PCI Express version of the spillbuffer cards, while the second scenario proposes replacing the readout buffers and event builders with a new architecture based on the FPGA technology. We analyzed the requirements posed on the software for this new hardware architecture, and we proposed and implemented the minimal control and monitoring application framework. The preliminary performance and stability test results have been discussed - the performance of the system should meet the expected requirements.

During the rest of the winter shutdown, further tests are scheduled. In order to test the system under more realistic conditions, the slave processes need to be deployed on the GeSiCA concentrator modules equipped with the prototypes of the new hardware. At longer scale, the system specification needs to be extended and finalized in 2012 so that the system will be ready for final testing during the expected annual shutdown of all CERN accelerators in 2013. It is planned to deploy the new architecture for the 2014 Run.

ACKNOWLEDGEMENTS

This work has been supported by Czech Ministry of Education, Youth and Sports grants LA08015 and SGS 11/167.

REFERENCES

[1] P. Abbon et al. (the COMPASS collaboration). *The COMPASS experiment at CERN.* In: Nucl. Instrum. Methods Phys. Res., A 577, 3 (2007) pp. 455–518.

[2] Ch. Adolph et al. (the COMPASS collaboration). *COMPASS-II proposal.* CERN-SPSC-2010-014; SPSC-P-340 (May 2010).

[3] L. Schmitt et al. *The DAQ of the COMPASS experiment.* In: 13th IEEE-NPSS Real Time Conference 2003, Montreal, Canada, 18–23 May 2003, pp. 439–444.

[4] T. Anticic et al. (ALICE DAQ Project). *ALICE DAQ and ECS User's Guide.* CERN EDMS 616039, January 2006.

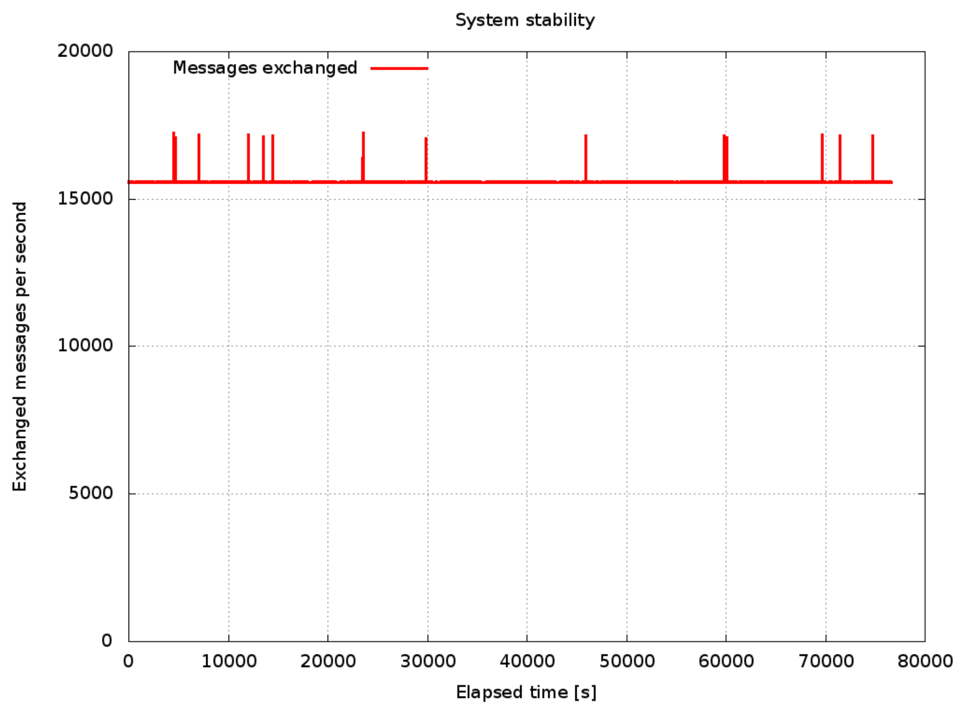

Figure 4. System stability.

[5] T. Nagel. *Cinderella: an Online Filter for the COMPASS Experiment*. München: Technische universität München, January 2009.

[6] A. Mann, F. Goslich, I. Konorov, S. Paul. *An AdvancedTCA Based Data Concentrator and Event Building Architecture*. In 17th IEEE-NPSS Real-Time Conference 2010, Lisboa, Portugal, 24–28 May 2010.

[7] P. Charpentier, M. Dönszelmann, C. Gaspar. *DIM, a Portable, Light Weight Package for Information Publishing, Data Transfer and Inter-process Communication*. Available at: `http://dim.web.cern.ch` [2013–08–20]

[8] V. Jarý, T. Liška, M. Virius. *Developing a New DAQ Software For the COMPASS Experiment*. In: 37th Software Development, Ostrava: VŠB – Technická univerzita Ostrava, 2011, ISBN 978-80-248-2425-3. p 35-41.

[9] M. Bodlák, V. Jarý, T. Liška, F. Marek, J. Nový, M. Plajner. *Remote Control Room For COMPASS Experiment*. In: 37th Software Development, Ostrava: VŠB – Technická univerzita Ostrava, 2011, ISBN 978-80-248-2425-3. pp. 1–9.

[10] *COMPASS page* [online]. 2010. Available at: `http://wwwcompass.cern.ch` [2013–08–20]