The Resource Manager the ATLAS Trigger and Data Acquisition System

I ALEKSANDROV[a], G AVOLIO[b], G LEHMANN MIOTTO[b], I SOLOVIEV[d]
on behalf of the ATLAS TDAQ Collaboration

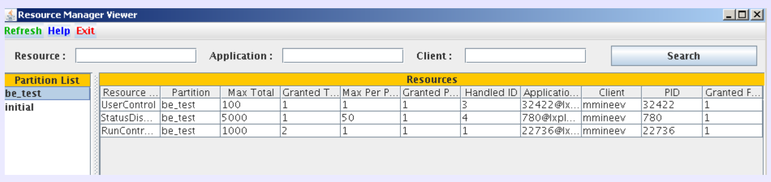
a: Joint Inst. for Nuclear Research – JINR (RU), b: CERN, d: University of California Irvine (US)

1. Introduction

The **Trigger and Data Acquisition[1]**(TDAQ) system of the **ATLAS[2]** detector at the Large Hadron Collider (LHC) at CERN is composed of a large number of distributed hardware and software components (about 3000 machines and more than 15000 concurrent processes at the end of LHC's Run I) which provide the data-taking functionality of the overall system.

The Resource Manager (RM) is one of the core components of the ATLAS online Data Acquisition system. The Resource Manager marshals the right for applications to access resources which may exist in multiple but limited copies in order to avoid conflicts due to program faults or operator errors. The access to resources is managed in a manner similar to what a lock manager would do in other software systems. The Resource Manager is queried about the availability of resources every time any application needs to be started.

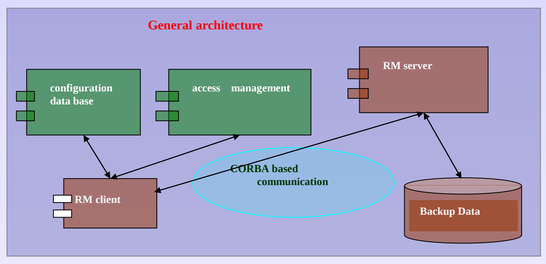
During routine ATLAS operations many applications must be started and stopped within a small time window. The Resource Manager is designed to handle of order 30k requests within a few seconds from O(1k) clients in the data acquisition system via a custom API. A GUI is also available for use by experts to view and update resources as needed.



2. The Resource Manager - Architecture

The overall architecture of the RM is presented in the figure below. The RM component is essentially divided into RM server and client parts. The RM client performs requests to the RM server using **CORBA based Communication**[3]. The RM client uses **access management**[4] for some types of action before issuing requests in order to check if such actions are permitted for user. Clients can ask for allocation or release resources, get information about allocated resources and update configuration data that is stored in the RM server dynamic data base (DDB). RM clients use a **configuration data base**[5] in order to load objects that correspond to resources and their associations.

The RM server runs as a single instance convering the entire TDAQ system. The RM server is a passive component and reacts only to client requests. It consists of an RM DDB and wrapper, which provides multi-threaded processing of DB requests as well as support for backups. The RM DDB keeps all data concerning allocated resources and corresponding allocation parameters like client, application, program identifier etc.

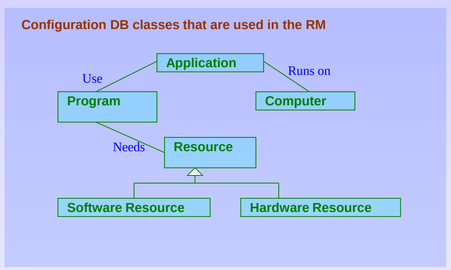


The RM server is essential to be able to control RM resource usage while the TDAQ system is running. The server should therefore be robust and handle all failures in a graceful way including restarts in case of a crash and recovery of data from a backup. Data backups consist of a base file storing all RM DDB information and a logout file storing any changes on top of the base file data. The base file itself is updated periodically. The logout file is immediately updated when a corresponding action is completed successfully. When the RM server is started and restored from backup the base file is used to recover the bulk of the data and then the remaining actions are recovered from the logout file by parsing records.

The RM server is implemented in C++. The client is implemented in C++ and in java. Java is used mostly in different GUI calls. A python implementation of the RM client functionality was recently developed for monitoring resource states.

3. The Resource Manager Context

All the available resources that are used by the RM and their associations to software processes are described in the configuration database. You can see simplified diagram of corresponding classes in the figure below. There are two types of RM resources in the ATLAS TDAQ configuration database. An RM Software Resource is used when only limited copies of some software may run in the ATLAS TDAQ system on any host. An RM Hardware Resource is used when only one application can start on the same host. RM Hardware Resources are associated with program which has no association with computer. This greatly decreases the number of resources that should be stored in the database. The RM creates whatever hardware resources are needed on the fly when an application starts

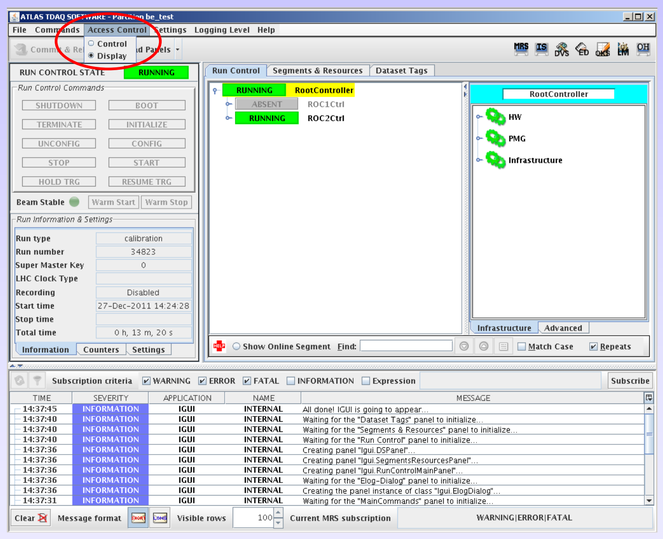


Most ATLAS TDAQ processes are started using the **Process Manager**[6] (PMG). The PMG is the main user of RM client functionality, sending requests to the RM server via the client API to allocate resources for applications to be started. Once a task is complete requests are also sent to the RM server to free any allocated resources. The PMG is designed to avoid situations where applications consuming resources crash and therefore do not free what they have been allocated.

4. The Resource Manager Use Cases

An example of the use of RM software resources is the Integrated Graphical User Interface (IGUI). The IGUI can run in control and display mode as you can see in the picture marked by the red ellipse. Only one copy of the IGUI can run with control rights and a limited number of IGUI instances can run in display mode. Two corresponding resources are defined in the configuration database. The IGUI uses the RM client java interface to request resources.

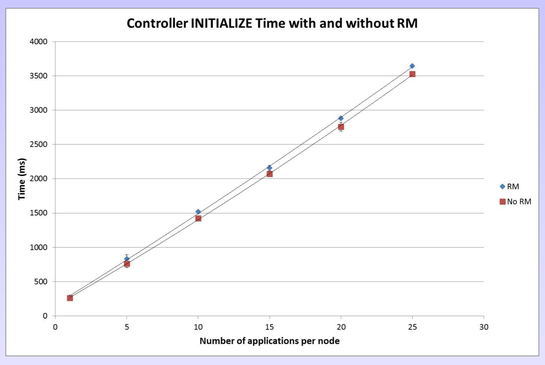
RM Hardware resources are most widely used in the ATLAS Readout System (ROS). The read-out cards (RobinNP) can not be shared. Only one RobinNP controller can start on the host PC. Just one hardware resource is stored in configuration database. The PMG requests controller resources before starting it on a host using the host identifier as one of the request parameters. The RM server checks if the corresponding resource is already allocated and gives permission to proceed if free.



5. The Resource Manager Upgrades

During the LHC's Long Shutdown period, the Resource Manager's requirements have been reviewed in light of the experience gained during the LHC's Run I. As a consequence, the Resource Manager has undergone a full re-design and re-implementation cycle with the result of a reduction of the code base by 40% with respect to the previous implementation thus leading to a **more maintainable component**.

The RM DDB implementation contained some classes using multiple references in order to minimize memory use and provide fast response to any request. Backup support was based on usage of a configuration database backup facility that stored data in XML format. The usage of new features available in the C++11 standard, such as boost multi-containers, made it possible to build a more simple composite key based data store while still providing the needed functionality. The use of boost archiving features also helped to simplify the backup facility.



The RM should have minimal influence on application start time. Controller initialization tests for different numbers of applications per node were performed with and without the RM running. The plot shows that there is no overhead from the inclusion of the resource manager.

The redesign of the RM Server did not lead to changes in the RM Client. The only additions to the client were some python functions to facilitate more easy access to resource information.

6. Conclusions

The Resource Manager Server runs as a single process and robustly controls all RM resources across the TDAQ system whenever applications are running.

New features available in C++11 helped to significantly simplify RM server re-implementation.

Resource Manager server re-design and re-implementation made the component more maintainable, which is important for long-term support.

References

1. The ATLAS Collaboration, 2002, *ATLAS high-level trigger, data-acquisition and controls: Technical Design Report*
2. The ATLAS Collaboration, 2008, *The ATLAS Experiment at the CERN Large Hadron Collider*, J. Instrum.
3. Common Object Request Broker Architecture home www.corba.org
4. Leahu, Marius Constantin; Dobson, Marc; Avolio, Giuseppe, "Access Control Design and Implementations in the ATLAS Experiment", IEEE Transactions on Nuclear Science, vol. 55, issue 1, pp. 386-391 2008
5. Lehmann Miotto G et al, "Configuration and control of the ATLAS trigger and data acquisition", Nuclear Instruments and Methods in Physics Research Section A, Volume 623, Issue 1, p. 549-551., 11/2010
6. Avolio, Giuseppe; Dobson, Mark; Miotto, Giovanna Lehmann, Wiesmann, Matthias, "The Process Manager in the ATLAS DAQ System", IEEE Transactions on Nuclear Science, vol. 55, issue 1, pp. 399-404 2008