HYPERARCHIVER: AN EVOLUTION OF EPICS CHANNEL ARCHIVER

M. del Campo*, I. Arredondo, ESS-Bilbao, Zamudio, Spain
J. Jugo, University of the Basque Country, Leioa, Spain
M. Giacchini, L. Giovannini, INFN/LNL, Legnaro, Italy

Abstract

Data storage is a primary issue in any research facility. In the EPICS middleware based accelerator community, Channel Archiver has been always considered the main reference. It works with Oracle and MySQL, probably the best well known relational databases. However, demanding requirements at minimum costs have fostered the development of a wide range of alternatives, like MDSPlus (Consorzio RFX), SciDB (BNL) or Hypertable (IFNF). This document launches a tool called HyperArchiver, which was firstly developed at IFNF (Italy) and eventually customised by ESS Bilbao (Spain). Based on a NoSQL database named Hypertable, it focuses on large data sets management with maximum scalability, reliability and performance. Besides the update and further customization made at ESS Bilbao, HyperArchiver is presented with a set of GUIs, in order to provide an easy use and integration with any general control system. A LabVIEW VI and two cross-platform PyQt GUIs for both Hypertable data retrieval and HyperArchiver control have been developed and successfully tested at ESS Bilbao (see Figures 1–3).

INTRODUCTION

Particle accelerators are very complex and expensive devices. Therefore a reliable distributed control system allowing easy maintenance and upgrading turns basic in this kind of facilities. EPICS (Experimental Physics and Industrial Control System) is a set of Open Source software tools and applications, which provides the infrastructure for distributed control systems. Many particle accelerator, large experiments and major telescopes facilities use it to build complex control systems made of tens or even hundred of computers, networked together to allow communication between them and provide control and feedback from different locations. In this context, data storage is a very important issue, not only as part of the control system itself, but also for the proper functioning and use of the accelerator and its experimental lines.

EPICS ARCHIVING TOOLS

The most standard EPICS standalone client for data archiving is called Channel Archiver [1], which is based on the relational databases MySQL and Oracle (therefore it is also called RDB Archiver). Unfortunately, they both have some well known drawbacks, which explain the efforts made towards the improvement of its performance and usability [2]. However, this has also fostered the development of a wide range of alternatives focused on high performance and large-scale dataset management, like MDSPlus, SciDB (BNL) [3] or Hypertable (IFNF) [4]. In this work only the last one will be discussed, but it should be kept in mind that they are all possible solutions for the RDB Archiver limitations.

Channel Archive

The Channel Archiver is the standard archiving tool for EPICS. Using the EPICS Channel Access (CA) network protocol, it can collect real-time data from any CA server on the network. It stores the full data set offered by CA (values, timestamps, status information, engineering unit names and display, control and alarm limits), allowing scanning at a fixed period or on change.

Since its original design, the EPICS Channel Archiver has undertaken several significant transformations. The version publicly known as an EPICS extension [1] was developed at the Spallation Neutron Source (SNS), Los Alamos, USA. Its main feature is the usage of a relational database instead of the indexed files used by the original Channel Archiver [5]. This version of Channel Archiver supports both MySQL and Oracle RDB and is entirely written in Java as part of the Control System Studio (CSS). This tool significantly improves data access and retrieval, in comparison to the original indexed file. Nevertheless, both Oracle and MySQL have some disadvantages like the expensive costs of Oracle or the effective maximum table size in MySQL, which can become a great limitation considering that RDB Channel Archiver stores all samples in one single sample table.

HyperArchiver

HyperArchiver [6] is a modification of the Java RDB ArchiveEngine which uses Hypertable instead of MySQL or Oracle. It was developed at INFN/LNL in Legnaro (Italy) and customised at ESS Bilbao (ESS-B) afterwards. Hypertable is a non relational, high performance and distributed database released under GNU license, which focuses on management of large data sets with maximum scalability and reliability. HyperArchiver is actually a hybrid version, as samples (bulk data) are stored in Hypertable database, but the basic static data of EPICS PVs is still recorded in RDB Archiver's MySQL database. First tests at Brookhaven National Laboratory (BNL), USA, have already shown very good performance on data insertion and retrieval [7].

HYPERARCHIVER AT ESS BILBAO

HyperArchiver Customization

HyperArchiver version released at Legnaro was meant to be used with Hypertable 0.9.3.3, and had some proved

* mcampo@essbibao.org

ISBN 978-3-95450-124-3

limitations on arrays management. For these reasons, at ESS-B some further modifications where required before it could be properly used. The features added at ESS-B to the original HyperArchiver are the following:

- Upgrade from 0.9.3.3 to 0.9.4.3 and further Hypertable versions (a new architecture based on namespaces was introduced from version 0.9.4.3). Last setups at ESS-B are currently working with version 0.9.5.6.
- Improve the management of large arrays and the way they are stored inside Hypertable. This was something that had not been tested during HyperArchiver development at INFN/LNL, mainly focused on single PVs.
- Provide the user the possibility to automatically create the EngineConfigImport XML configuration file directly from an EPICS database file. This goes along with a general effort at ESS-B to standarize all configuration files, providing conversion tools.

Integration into the General Control System

The standard use of HyperArchiver remains the same as for the classical RDB Channel Archiver. They both can be easily controlled from any command-line interface terminal. Nevertheless in order to simplify its use, especially for operators who might not be used to the UNIX terminal interface, an additional effort was made to develop some graphical user interfaces (GUI).

PyQt: PyQt4 is a set of python bindings for the Qt 4 cross-platform GUI/XML/SQL C++ framework. A standalone PyQt4 GUI was written in python at ESS-B and and has been established as the internal standard interface for HyperArchiver clients. It allows not only the initialization and termination of the HyperArchiver services but also the customisation of all its configuration parameters with no need to use the command line terminal or any external text editor tool. This avoids the user a lot of tedious work.

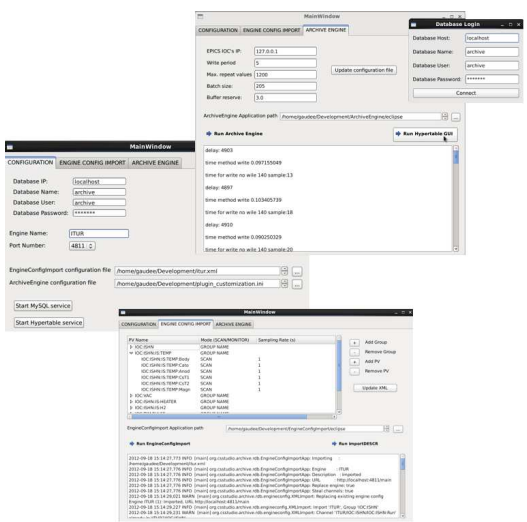

Figure 1: PyQt GUI for HyperArchiver Control.

LabVIEW: HyperArchiver is just one of many tools used in a control system. Therefore its integration into the general control system should be as easy and quick as possible. This was the purpose of a set of LabVIEW virtual instruments (VI) designed to control the Archive Engine with the same functionalities as the PyQt interface. This makes extremely easy the integration of a data storage engine in any of the LabVIEW control programs which are used at ESS-B at the moment, by simply including the mentioned VIs (programs/subroutines) inside the general block diagram.

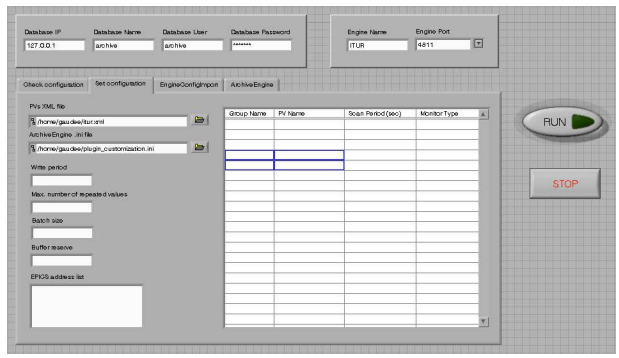

Figure 2: LabVIEW VI for HyperArchiver control.

Data Visualization

Regarding data visualization, the INFN/LNL version of Hypertable included a further modification of CSS code for the retrieval of historical data from Hypertable instead of MySQL or Oracle. It pictures data by means of the CSS data browser. CSS is a very powerful tool which provides a lot of functionalities. However, at this moment at ESS-B, less complex and more specific tools were requested. In this way, another python GUI was developed in order to have at disposal a simple data retrieval tool, lighter than CSS. It uses the Hypertable Thrift Client Interface to access the database and after thorough tests at ESS-B [8] it proved to be even faster than CSS Data Browser. Moreover, the use of python made it possible to integrate the data retrieval tool into the general HyperArchiver PyQt GUI, making available all the functionalities required for the control and use of HyperArchiver from a single multiplatform user friendly interface.

RESULTS

This work presents the EPICS experience at ESS-B regarding data storage and database management. To date, two different versions of EPICS Archive Engine have been used and tested to study their performance and suitability to ESS-B requirements: RDB Archiver and HyperArchiver.

RDB Archiver: RDB Archiver was set up with a MySQL server. Both the Archive Engine and the relational database worked without any issue during laboratory tests. However, first real tests at ITUR (the ESS-Bilbao's front-end test stand for ion sources) revealed a potencial problem regarding size limitations of the tables. ITUR's control system is characterized by the use of several array PVs, sam-

ISBN 978-3-95450-124-3

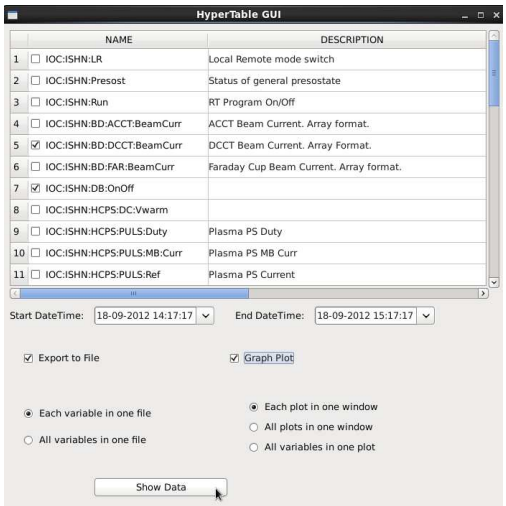

Figure 3: PyQt GUI for Hypertable data retrieval.

pled at a high sampling rate, for waveform storage. Considering that RDB Archiver manages arrays creating an entry into a single MySQL for every item of each array, size limitation turned out to be a problem. The maximum table size for MySQL databases is usually determined by operating system (OS) constraints on file sizes, not MySQL, but can be so restrictive as 4GB on a 32-bit architecture MySQL. Table 1 lists some rough aproximations on different OS [9]. The use of a proprietary distributed cluster of Oracle's databases was dismissed at first at ESS-B, which leans towards the promotion of the Open Source model. The better suitability of a distributed large-scale dataset oriented database thence emerged.

Table 1: File Size Limits on Different OS

Operating System	File size Limit
Win32 w/ FAT/FAT32	2 GB / 4 GB)
Win32 w/ NTFS	2 TB
Linux 2.2-Intel 32-bit	2 GB (LFS: 4 GB)
Linux 2.4+	4 TB (using ext3)
Solaris 9/10	16 TB
MacOS X w/ HFS+	2 TB

HyperArchiver: HyperArchiver is actually a modification of RDB Archiver which stores data into the NoSQL database Hypertable instead of the RDB MySQL or Oracle. Hypertable is designed to manage the storage and processing of data on a large cluster of commodity hardware, providing resilience to machine and component failures. At ESS-B it has proved to be a reliable and scalable alternative to MySQL. Furthermore, other accelerator facilities have shown their interest on HyperArchiver as an alternative to the traditional RDB Archiver, like Diamond (UK).

Briefly, HyperArchiver emerged as an evolution of the standard RDB Archiver, modified to work with Hypertable as the main underlying database. The current trend towards NoSQL databases seems natural in the field of particle accelerators. Nowadays this kind of facilities generate huge amounts of data, which have to be immediately processed and properly recorded. NoSQL databases have been especifically developed to manage large volumes of data that do not necessarily follow a fixed schema. Moreover, they employ distributed architectures, which allows scalability and tolerance to hardware failure, both issues of great importance in any research facility.

NEXT STEPS

This aim of this work is to emphasize the feasibility of an evolution in the data archiving field of EPICS control systems. This is made through the example of HyperArchiver, a customization of the classical EPICS RDB Archiver which uses Hypertable instead of MySQL or Oracle. However, the huge amount of NoSQL databases focused on large datasets like Hypertable, creates a wide range of similar possibilities based on different databases, or even on different storage methods. The most important ones have already been mentioned, but with no further references, because they have not been tested yet at ESS-B. Therefore, even if HyperArchiver has proved to be a good and reliable archiving client, the logical path to follow in the future would be to test and compare other methods available, in order to determine the advantages and disadvantages of each one, trying to figure out which one is more suitable for each scenario. It is worth mentioning that this job has already been started at BNL, where HyperArchiver was actually conceived. They have created a common test bench to evaluate the various archiver developments [10].

REFERENCES

[1] http://ics-web.sns.ornl.gov/kasemir/archiver/

[2] J. Rowland, M.T. Heron, S.J. Singleton, K. Vijayan, M. Leech, "Algorithms and Data Structures for the EPICS", ICALEPCS 2011, Grenoble, France.

[3] N. Malitsky, D. Dohan, "A Prototype of the Next EPICS Archiver Based on the SciDB Approach", ICALEPCS 2011, Grenoble, France.

[4] http://www.lnl.infn.it/~epics/joomla/archiver.html

[5] http://arxiv.org/abs/cs/0110066v1

[6] M. Giacchini, L. Giovannini, M. Montis, G. Bassato, J.A. Vasquez, G. Prete, A. Andrighetto, R. Petkus, R. Lange, K. Kasemir, M. del Campo, J. Jugo, "HyperArchiver: An EPICS Archiver Prototype Based on Hypertable", ICALEPCS 2011, Grenoble, France.

[7] http://www.lnl.infn.it/~epics/HyperArchiverAix2010.pdf

[8] M. del Campo, J. Jugo, M. Giacchini, L. Giovannini, "EPICS HyperArchiver: Initial Tests at ESS Bilbao", IPAC2011, San Sebastiàn, Spain.

[9] http://dev.mysql.com/doc/refman/5.1/en/table-size-limit.html

[10] http://sourceforge.net/projects/epics-archbench

ISBN 978-3-95450-124-3