HYPERARCHIVER: AN EPICS ARCHIVER PROTOTYPE BASED ON HYPERTABLE

M. Giacchini, L. Giovannini, M. Montis, G. Bassato, J.A. Vasquez, G. Prete, A. Andrighetto,
INFN/LNL, Legnaro (PD), Italy
R. Petkus, BNL, Upton Long Island, New York, USA. R. Lange, HZB, Berlin, Germany
K. Kasemir, ORNL, OakRidge, Tennesse, USA.
M. Del Campo, ESS-Bilbao, Zamundio, Spain
J. Jugo, University of the Basque Country, Leiola, Spain.

Abstract

This work started in the context of NSLS2 project at Brookhaven National Laboratory. The NSLS2 control system foresees a very high number of PV variables and has strict requirements in terms of archiving/retrieving rate: our goal was to store 10K PV/sec and retrieve 4K PV/sec for a group of 4 signals. The HyperArchiver [1] is an EPICS [2] Archiver implementation engined by Hypertable, an open source database whose internal architecture is derived from Google's Big Table. We discuss the performance of HyperArchiver and present the results of some comparative tests.

EPICS ARCHIVER BRIEF HISTORY

The Channel Archiver [3] is an archiving tool-set for EPICS based control systems. It can archive any kind of record available through Channel Access [4], the EPICS network protocol. Bob Dalesio designed the original index file, data file layout, and implemented the first prototype of Channel Archiver. From then on, many people in the collaboration have been involved on this EPICS extension. The largely used Archiver version is still based on that design and last release is dated on August 29, 2006. That version is based on its own binary file format to archive the PVs. In July 2009 SNS stopped using such version and designed a new archiver. The new version can be engined with two kinds of RDB: MySQL or Oracle. All code has been written in Java and embedded into the Control System Studio (CSS)[5]. CSS can be used to browse the data and to retrieve them. The SNS archiver is based on Oracle RDB in production and MySQL for test purposes; the data base is accessible via Control System Studio (CSS) data browser. This archiver version realized at SNS embeds a Java Archiver into CSS.

TARGET

NSLS2 project at BNL expects a large amount of data that have to be acquired really fast. The goal was storing 10K PV/sec and retrieving 4K PV/sec for a group of 4 signals with a more reliable and safe archiving architecture. The SNS archiver, based on Oracle, seems not fast enough and expensive because of Oracle DB licence fee. The MySQL version has potential scalability concerns, and is designed for a single machine, then heavy processing loads may require expensive hardware and ad hoc solutions. At that time, summer 2009, two approaches were evaluated:

- Rewriting the Rtree and double-linked list embedded DB structure
- Look for a complete replacement of the embedded DB

After an initial period spent in analysing the first solution, we switched to the second one and looked for a completely new technology. The idea came from the observation that the fastest and largest DB available nowadays is Google, whose search engine is based on the proprietary BigTable filesystem. Google never published the details of Bigtable implementation, nor released a licensed version of its filesystem. However, we found an open source product based on similar concepts and technology (Hypertable [6]) and started working on it. Hypertable is a massive, parallel, high performance database, that isn't a RDB nor a SQL DB. During our first tests, Hypertable was only available as part of an open source software project (under GNU2 Licence). Nowadays Hypertable is delivered either in a free version (used for our project) and in a commercial version (by Hypertable Inc.) with payable support. We are using now the 0.9.6.0 version (the last beta release); the roadmap has fixed the first major release 1.0 in December 31st 2011, which will mark the end of the beta period and will introduce a fully functional BigTable implementation.

SYSTEM DESCRIPTION

The system used is a virtual Linux box with 16 CPUs, 10 GB Ram and 3.6 TB of ISCSI disk space on a HP server model DL380G7. A similar system running on a native OS, instead of a virtualized box, would exhibit better performances: we decided, however, to use a virtual machine to make easy cloning the “box” and sharing it with other developers, saving them the time to set-up a complete system from scratch. The virtual layer technology we used is KVM and the underlying operating system is Linux CentOS 5 (a RedHat Linux Enterprise recompiled from source).

The ISCSI disk has been tested separately to be sure it couldn't be a bottleneck for our tests and it showed a R/W

speed of 45 MB/sec; we can assume, therefore, it didn't affect our evaluation.

The bench-test core is an IOC with the following features:

- 10K Record type Analog Input
- Every record has a scan rate 1 sec

The HyperArchiver is configured to insert 10K samples/sec and to extract, from CSS, 1K samples of 4 PVs.

The HyperArchiver configuration is an hybrid system based on MySQL and Hypertable 0.9.5.0.pre6. MYSQL DB is used to store the configuration setup while Hypertable is used to store the Channel data: channelName Status, TimeStamp, Severity, Simple Mode, SimplePeriod, type, and value.

The data read from Channel Access by Channel Archive Engine v.1.2.5 through JCA are concatenated in a string and stored into Hypertable.

The Hypertable schema has only one column named "pv", the rowkey is based on a PV name with the EPICS Time Stamp as follow:

- Rowkey: 100:aiExample.1315639247116
- ColumName: pv
- Data:100:aiExample#c#LOW_ALARM#c#1315639247116#c#MINOR#c##c#jdbc:mysql://localhost:3306/archive#c#Monitor#c#10.0#c#double#c#3.0

Implementation Details

Starting from the original Archiver developed at SNS and based on Java, we made minor changes on various classes; the most important work has been done on package named org.csstudio.archive.rdb which has been designed as shown in Figure 1. The new package named org.csstudio.archive.htrdb2 (Figure 2) has deep

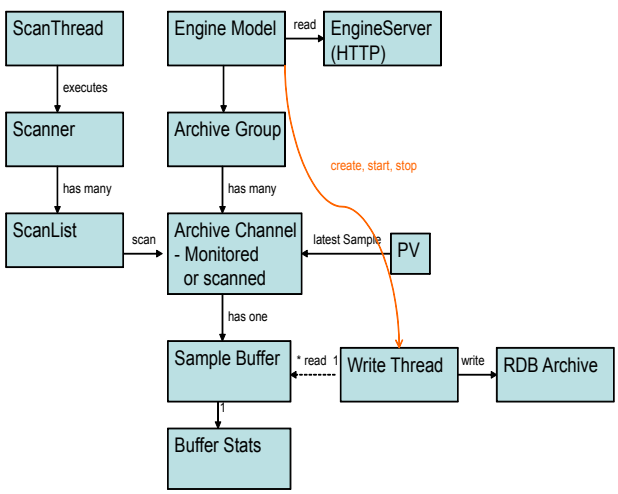

Figure 1: Generic RDB Archive Engine model.

modifications in the class named RDBArchive. Using the Trift client API we added two new classes (Figure 2) :

- HTArchive which contain connections methods;
- HtTableArchiveSerialized which contains methods to manages datas.

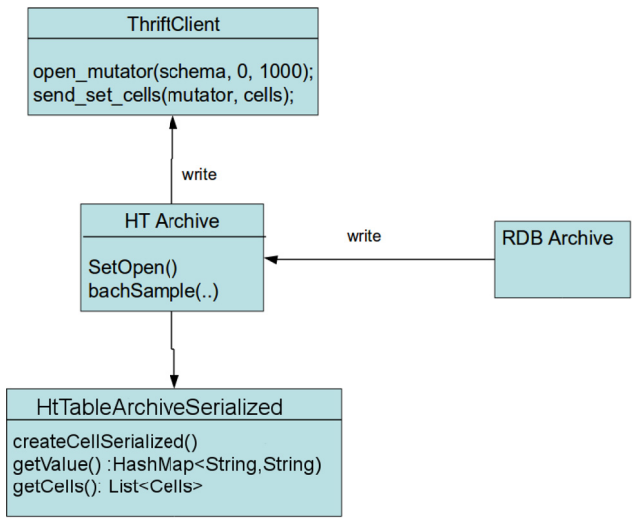

Figure 2: Classe's schema of org.csstudio.archive.htrdb2 package.

The data stored can be readout by dataBrowser2, replacing the package org.csstudio.archivereaderrdb by org.csstudio.archivereader.htrdb2. The most importat modification is on RawSampleIterator class where the Hypertable connection and retrieve task has been realized.

PERFORMANCES

Several test has been carried out at LNL. Using the BNL iocServer we started with a scan period of 0.1 sec for each PV of 10K analog input records. The archive engine seems to be unable to sustain that acquisition rate, so we decide to postpone a more detailed analysis on this and moved a step back using a scan period of 1 sec. for all PVs. With this set-up the average insertion rate is 13 MB/s and the retrieve rate of 13MB/s from CSS. Data were retrieved through the CSS GUI.

CONCLUSIONS

BNL has proposed a common test bench to evaluate the various archiver developments[7]. We used that guidelines for our tests to share and compare our benchmarks with the EPICS community. As mentioned before, the HyperArchiver is still an hybrid system in his internal architecture; it shows anyway good performances and seems a promising research line. Keeping in mind the fundamental spirit of collaboration in Epics community, future steps on this project will be done after a discussion with the laboratories interested on his development. This will help us to better focus on common targets and methods, and obtain the maximum results with the largest benefits for the community.

ACKNOWLEDGEMENTS

Sincere acknowledgements to Bob Dalesio who has made possible the beginning of this research during the stage of M. Giacchini at BNL in 2009.
Many thanks to the great BNL controls team and particularly to R. Petkus and R. Lange for their valuable help. The highly professional collaboration of K.Kasemir from SNS, his great experience and skills were essential. Last but not least, acknowledgements to ESS Bilbao, in particular to M.Campo and Prof. J.Jugo who first trusted on this project and brought it to the production stage at ESS-Bilbao's facilities at Zamundio (Spain) [8].

REFERENCES

[1] HyperArchiver: http://www.lnl.infn.it/~epics/joomla/archiver.html
[2] Epics: http://www.aps.anl.gov/epics/
[3] ChannelArchiver: http://sourceforge.net/apps/trac/epicschanarch/wiki
[4] J.O. Hill: Channel Access: A Software Bus for the LAACS, ICALEPCS 1989, Vancuuver.
[5] Control System Studio (CSS): http://cs-studio.sourceforge.net/
[7] ICALEPCS 11, N. Malitsky, D.Dohan “A Prototype of the Next EPICS Archiver Based on the SciDB Approach”
[8] IPAC 11, M. del Campo, J. Jugo, M. Giacchini, L. Giovannini “EPICS HYPERARCHIVER: INITIAL TESTS AT ESSBILBAO”