THE EPICS ARCHIVER APPLIANCE

Murali Shankar, Luofeng Li, SLAC, Menlo Park, CA, USA*
Michael Davidsaver, BNL, Upton, New York, USA
Martin Konrad, National Superconducting Cyclotron Laboratory, Michigan State University, East Lansing, MI 48824, USA**

Abstract

The EPICS Archiver Appliance was developed by a collaboration of SLAC, BNL and MSU to allow for the archival of millions of PVs, mainly focusing on data retrieval performance. It offers the ability to cluster appliances and to scale by adding appliances to the cluster. Multiple stages and an inbuilt process to move data between stages facilitate the usage of faster storage and the ability to decimate data as it is moved. An HTML management interface and scriptable business logic significantly simplify administration. Well-defined customization hooks allow facilities to tailor the product to suit their requirements. Mechanisms to facilitate installation and migration have been developed. The system has been in production at SLAC for about 2 years now, at MSU for about 2 years and is heading towards a production deployment at BNL. At SLAC, the system has significantly reduced maintenance costs while enabling new functionality that was not possible before. This paper presents an overview of the system and shares some of our experience with deploying and managing it at our facilities.

REQUIREMENTS

At SLAC, BNL and MSU, we use EPICS as our control system for our various facilities; many of these have several million Process Variables (PVs) that are used for monitoring and control. A common requirement is to be able to archive some of these PV's to facilitate troubleshooting and analysis.

Archiving a million PVs has some non-technical requirements. In addition to scaling gradually by adding additional appliances/hardware, we need better support for managing all aspects of the system. This includes establishing policies for archiving and the flexible configuration of archiving on a per PV basis. The ability to manage the system using a web based UI and from within scripts is a necessity. These management functions must include the ability to make changes to the PV archiving configuration without having to restart the system. Finally, we needed a simple migration path from the ChannelArchiver [1, 2].

In terms of storage, the data rates are high enough to require faster storage but the data volumes are also large. In addition, the archivers require the storage to be always available. A solution using multiple stages of storage is most economical. The first stage is expensive fast storage local to the appliance and thus always available. The final stages of storage can be much slower and cheaper with relaxed requirements for availability including support for tape storage. The final design should be extensible enough to support storage from external vendors like S3.

Our customers also wanted a solution that focused on data retrieval performance. The most common use of archive data is to help trouble shoot machine operations. Thus, the bulk of the data retrieval requests are for recent data. Data can then be moved off to cheaper, slower storage once it has aged. Our goal was to be able to retrieve a day's worth of 1Hz double data in less than 0.5 seconds; the actual retrieval numbers are much faster. In addition, support for various forms of data reduction during data retrieval is critical.

ARCHITECTURE

The EPICS Archiver Appliance uses an appliance model for deployment. An installation is a cluster of appliances. Each appliance (see Fig. 1) has multiple storage stages and multiple processes.

A wide variety of storage configurations are possible (on a per PV basis). The out-of-the-box configuration has these storage stages

- STS (Short term store) - The most recent couple of hours worth of data is typically stored here. This is typically a RAM disk.
- MTS (Medium term store) - The most recent couple of days worth of data is stored here. At SLAC, we use RAIDed 15k SAS drives for the MTS. At MSU, these are mirrored 1.2TB SAS drives.
- LTS (Long term store) - The rest of the data is stored here. At SLAC, this is bulk storage (with tape backups) that we rent from our computing dept. This is a GPFS filesystem located elsewhere and is mounted over NFS. At MSU, this is a NetApp alliance with 2.8 TB of storage.

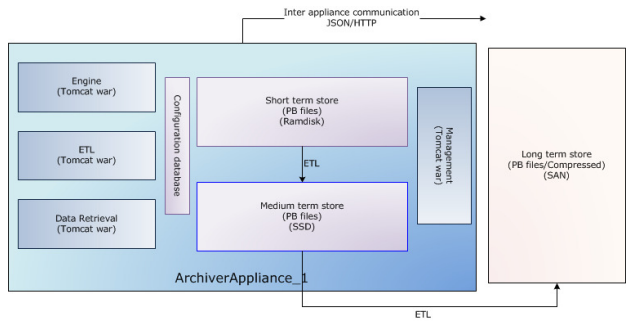

Figure 1: Components of an appliance.

* Work is supported by the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515 for LCLS I and LCLS II
** This work was supported in part by the National Science Foundation under the Cooperative Agreement PHY-11-06007

ISBN 978-3-95450-148-9

Each appliance has 4 processes; these are J2EE WAR files and are deployed on separate Tomcat containers.

- Engine - This component establishes EPICS Channel Access monitors for each PV in the appliance. The data is written into the STS. This component is based on the CS-Studio engine.
- ETL - This component moves data between stores - that is, it moves data from the STS to the MTS and from the MTS to the LTS.
- Retrieval - This component gathers data from all the stores, stitches them together to satisfy data retrieval requests.
- Mgmt - This component executes business logic, manages the other three components and holds runtime configuration state.

Appliances and components talk to each other using various means including JSON/HTTP. The archiving configuration is typically stored in a MySQL database; each appliance has its own configuration database.

SERIALIZATION

The configuration database has an entry for each PV; this is a JSON object that has the details on how the PV is being archived. This includes the sampling mode and rate; it also includes a list of data stores (see listing 1).

```
"dataStores": [
   "pb://localhost?name=STS&rootFolder=...&partition
   Granularity=PARTITION_HOUR...",
   "pb://localhost?name=MTS&rootFolder=...&partition
      Granularity=PARTITION_DAY...",
   "pb://localhost?name=LTS&rootFolder=...&partition
      Granularity=PARTITION_YEAR..."
],
```

Listing 1: Each PV has a list of datastores.

Each data store is implemented using a storage plugin; these are Java objects that implement standard interfaces. The out of the box install uses the PlainPBStoragePlugin; this plugin uses Google's ProtocolBuffers (PB) [3] as the serialization mechanism. ProtocolBuffers provide a future-proof serialization framework with bindings for many languages. Each EPICS V3 DBR type is mapped to a distinct PB message. For EPICS V4 [4], NTScalars and NTScalarArrays are mapped to their V3 counterpart PB messages. All other V4 types are serialized using V4 serialization and stored as generic PB messages; thus giving the archiver the ability to store any V4 type.

The PlainPBStoragePlugin stores its data in chunks; each chunk contain serialized PB messages; one message per archive sample; one sample per line. In addition, the samples are ordered by their record processing timestamps; these are guaranteed to be monotonically increasing. Each chunk has a key that is based on the PV name and the time partition of the chunk; for example, EIOC/LI30/MP01/HEARTBEAT:2012_08_24_16.pb. Partition boundaries are strictly enforced; thus, the chunk key has enough information to identify the boundaries of the contained data. By default, the PlainPBStoragePlugin stores its data in files; one file per chunk with the chunk key as the file name.

These various constraints let us use various search algorithms on PB files without the need for an index. However, in the future, if an index is needed for certain PV's, the system can be enhanced to use indexes for these PV's. PB files try to optimize on storage consumption. On average, a PB ScalarDouble consumes about 21 bytes per sample to store the timestamp, value, status and severity along with several other optional fields.

ARCHIVE PV WORKFLOW

Supporting complex configuration on a per PV basis can be overwhelming if the end user had to make these decisions every time they add a PV to the archiver. In addition, many facilities use automated scripts to manage their archiving requests. This implies that the system must automatically make these decisions on behalf of the user.

When users request PVs to be archived, the mgmt and engine components sample the PV to determine event rate, storage rate and other parameters. In addition, various fields of the PV like the NAME, ADEL, .MDEL, .RTYP etc. are also determined. All of these parameters are passed to an installation specific policy that is implemented as a Python script. The policy examines this information and makes configuration decisions on behalf of the user. For example, you can establish and encode a policy to archive waveforms at a rate no more than a 1Hz.

Optionally, as part of a policy, we can also archive fields in addition to the VAL field. For example, one can establish a policy to archive the HIHI, LOLO in addition to the VAL field for all records of RTYP ai. These fields are stored as part of the data for VAL field.

A clustered solution also implies that a decision must be made as to which appliance is used to archive a PV. To help with this decision, the archivers maintain capacity metrics and use a minimax algorithm to automatically assign PVs to appliances.

DATA RETRIEVAL

The EPICS Archiver Appliance comes with plugins for the ArchiveViewer and the CS-Studio databrowser [5]. We are also currently working on a HTML5 viewer (Fig. 2) for archive data; this is bundled as part of the appliance.

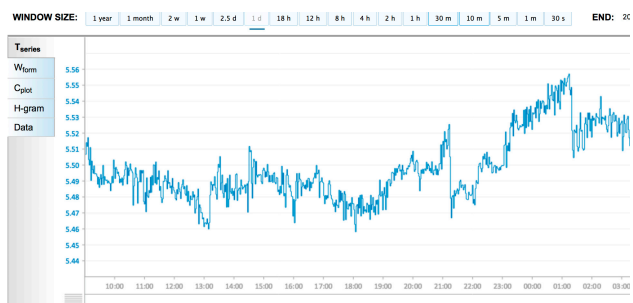

Figure 2: HTML5 viewer (in development).

ISBN 978-3-95450-148-9

A principal focus is data retrieval performance; the following chart (see Fig. 3) shows response times for data retrieval requests from data gathered over several weeks from a production system. Data retrieval requests are categorized in terms of their time span (endtime - starttime). Over 75% of the requests are for time spans less than a day and complete within 100ms (on an average). Requests with time spans of up to a week take an average of 250ms. Requests for time spans of a year with data reduction can complete in less than a couple of seconds; this involves binning over 30 million samples into about 8000 samples at runtime.

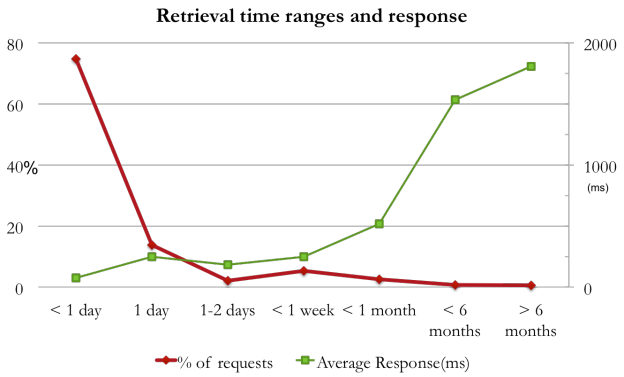

Figure 3: Response times vs time span of requests.

The EPICS Archiver Appliance supports data retrieval over HTTP in multiple formats/MIME types. A binary RAW format is supported that minimizes translation from the PB format. There are Java and Python libraries for clients interested in retrieving data using the RAW format. The carchivetools suite [6] includes command line Python scripts that retrieve data using the RAW format from the appliance. It also includes tools for searching. In addition, we have support for JSON, CSV, MAT, TXT and SVG MIME types. These formats let clients use tools like Matlab, Python, Excel, JMP and others to get data from the archiver.

Getting data into a tool necessitates construction of a data retrieval URL as the first step; this URL contains only the PV name and the start and end times of the data retrieval request. Both data retrieval and business logic requests can be dispatched to any random appliance in the cluster (Fig. 4); the appliance has the functionality to route/proxy the request accordingly.

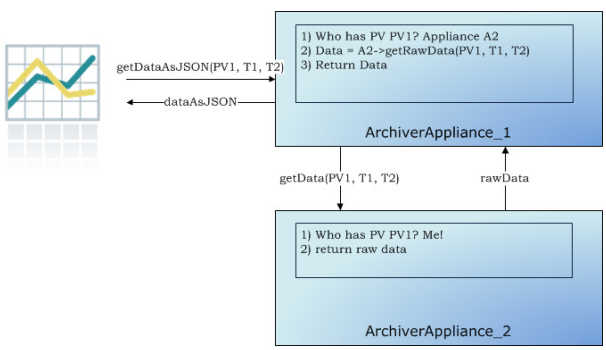

Figure 4: Requests can be dispatched to any appliance.

This enables us to use load balancers like mod_proxy_balancer (Fig. 5) in front of all the appliances in the cluster.

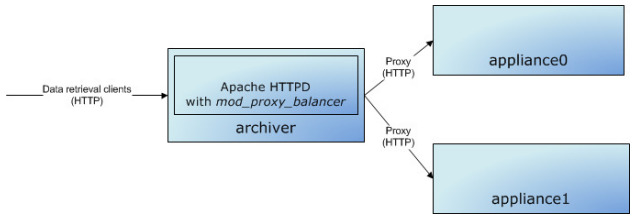

Figure 5: Traffic mgmt between clients and appliances.

Reducing data during data retrieval is a crucial requirement; however, there is no single correct way to reduce data. The EPICS Archiver Appliance has support for processing the data during data retrieval. This includes typical statistics operators like mean, standard deviation etc. These operators bin the data into bins of specified duration and then apply the specified operator on the bin. The appliance has the ability to precompute data reductions using these operators as part of the ETL process. If applicable, these precomputed data reductions are used during data retrieval to speed up the response times. These same operators can also be used to decimate data as ETL moves it from one store to another. Thus, one can store a few days worth of data at full rate and then decimate the data as it is moved to a slower store.

ADMINISTRATION

The EPICS Archiver Appliance offers a web UI (Fig. 6) for typical configuration tasks.

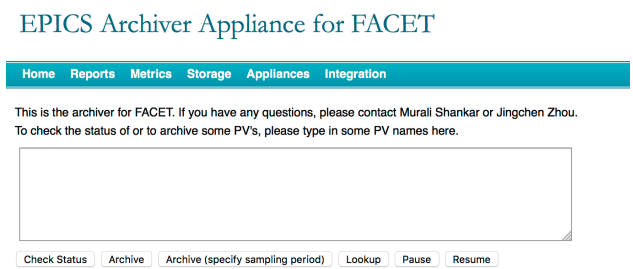

Figure 6: Web based management UI.

The web UI communicates with the appliances using JSON/HTTP and uses business logic exposed as web service calls. All of these web service calls are also available for use from external scripting tools like Python. There is a rich catalog of business logic that lets the user add/modify/delete PVs from the archiver, pause/resume PVs, reshard/consolidate data etc. None of these business logic operations requires an appliance restart. In addition, a wide variety of reports based on static and dynamic information are available. These include reports based on storage rate, disconnected PVs and many others. The reports are also accessible from Python scripts; thus one

ISBN 978-3-95450-148-9

can script the entire monitoring and administration of a cluster of appliances using Python scripts.

INTEGRATION WITH CHANNELARCHIVER

To facilitate a smooth transition from the ChannelArchiver, the EPICS Archiver Appliance includes the ability to proxy the ChannelArchiver XMLRPC data server. This allows facilities to skip a migration step where massive amounts of data have to be converted to a new format. The appliance also includes the ability to import ChannelArchiver XML configuration files. However, for facilities that do wish to convert their data, MSU has developed utilities to convert ChannelArchiver data into appliance PB files. The carchivetools suite also includes two backend servers, a2aproxy and archmiddle that can be used as a switchable proxy between a ChannelArchiver and an EPICS Archiver Appliance.

INSTALLATION

The EPICS Archiver Appliance requires recent versions of Linux, Java and Tomcat (and MySQL if configuration is stored there). While some installation scripts are provided; the easiest way to get going quickly is to use the provided Puppet modules. In addition, a quick start script is provided to facilitate easy evaluations.

DEPLOYMENTS

Table 1: Facilities with Production Deployments

Name	Lab	PVs	GB/day	Years	Cluster
LCLS	SLAC	200K	19	1.5	3
NSLS2	BNL	61K	42	0.5	1
NSCL	MSU	83K	1	2	2*
FACET	SLAC	34K	1	2	1
TestFac	SLAC	37K	1	2.5	1

* NSCL has two appliances to provide for redundancy

At SLAC, we have significantly reduced the maintenance costs of archiving; many tedious tasks have been eliminated. Many new use cases are currently being explored, including the use of archivers for fault analysis. The ease with which one can set up a small personal archiver that can then be integrated with a wide variety of tools is very useful. Many IOC developers regularly use a personal archiver to aid in development. Some operators use a separate archiver to gather data at high rates for a small set of PVs and then run complex processing on these datasets.

CUSTOMIZATION

It is unlikely that the features of the EPICS Archiver Appliance match a facility's requirements exactly over a period of time. The EPICS Archiver Appliance has been built with customization in mind. In addition to generating custom builds for a site specific look and feel, the product can be customized with site specific policies, configuration stores etc. The storage plugin interface allows a facility to store the data using an alternate storage scheme. If the PB serialization scheme is acceptable, but the notion of storing PB chunks as files is not; then Java NIO.2 can be used to store the chunks in any key value store. If not, a custom type system can be used to support an alternate serialization scheme.

SUMMARY

The EPICS Archiver Appliance has been in production use at SLAC/MSU/BNL and other facilities for two years. It has eliminated tedious maintenance tasks, reduced maintenance costs and allowed us to add more PV's to the archivers. Various facilities are in the process of evaluating/switching to the EPICS Archiver Appliance. A collaborative ecosystem has developed around the product with many labs contributing to the effort.

REFERENCES

[1] ChannelArchiver http://icsweb.sns.ornl.gov/kasemir/archiver/manual.pdf
[2] K. U. Kasemir and L.R.Dalesio, "Overview of the EPICS Channel Archiver", ICALEPCS, 2001.
[3] Protocol Buffers: https://developers.google.com/protocol-buffers/?hl=en
[4] EPICS V4 Normative Types http://epicspvdata.sourceforge.net/alpha/normativeTypes/normativeTypes.html
[5] Control System Studio http://controlsystemstudio.org/
[6] carchivetools https://github.com/epicsdeb/carchivetools

ISBN 978-3-95450-148-9