EPICS V4 EVALUATION FOR SNS NEUTRON DATA*

K.U. Kasemir, G.S. Guyotte, M.R.Pearson, ORNL, Oak Ridge, TN37831, USA

Abstract

Version 4 of the Experimental Physics and Industrial Control System (EPICS [1]) toolkit allows defining application-specific structured data types (pvData) and offers a network protocol for their efficient exchange (pvAccess). We evaluated V4 for the transport of neutron events from the detectors of the Spallation Neutron Source (SNS) to data acquisition and experiment monitoring systems. This includes the comparison of possible data structures, performance tests, and experience using V4 in production on a beam line.

MOTIVATION

On SNS beam lines, each neutron event consists of a pixel ID that identifies the location of the detector where a neutron was observed, and a time-of-flight measurement that describes when the neutron was detected relative to the most recent beam pulse. Depending on the beam line and its specific configuration, event rates can reach a few million events per seconds.

This neutron event information needs to be transferred from detectors to processing stages that provide users of the experiment with visual feedback, accumulate information that allows for the automation of the experiment, and finally stream the events into a data collection pipeline for long-term storage of the experiment data.

The original SNS beam line data acquisition software used a locally developed UDP/IP-based network protocol to transmit neutron event information [2]. The limited performance and reliability of this protocol necessitated an update of the overall data acquisition software [3].

EPICS V4

Based on a proven track record for control of the SNS accelerator, EPICS had been chosen as a toolkit for the beam line control system upgrade. While Channel Access [4], the original EPICS network protocol, has a well defined and functional set of data objects, this set is fixed to types suitable for describing a single data point, for example a temperature reading or voltage set point.

*This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan(http://energy.gov/downloads/doe-public-access-plan).

EPICS V4 [5] is an addition to the EPICS toolkit that introduces an alternative to the existing EPICS V3 data types and network protocol.

pvData

pvData is the EPICS V4 library for structured data. It can describe such data in an operating-system independent way, hold it in memory, and copy complete or partial data containers. The data can include time stamps, numeric values, enumerated data, text and alarm information. Data can also be assembled into arrays and structures.

Normative Types

pvData allows clients to define nearly arbitrary data structures, which is ideal for packaging site-specific information. At the same time it limits the interoperability of applications. *Normative Types* are a set of agreed-upon pvData types that all implementers of V4 applications are encouraged to support. All original EPICS V3 data types are described as Normative Types, allowing for an eventual transition from V3 to V4. In addition, data types like N-dimensional images or statistical samples that are often used in higher-level control system applications are available as Normative Types.

pvAccess

pvAccess is the V4 network protocol that allows for the exchange of pvData. It is conceptually similar to V3 Channel Access, using UDP/IP for channel name resolution, and then establishing a TCP/IP connection between pvAccess servers and clients to exchange data. It supports basic read and write access. A subscription mode efficiently updates clients on changes in the data by only transferring the modified structure elements. Finally, a combined write/read mode supports remote service calls by atomically sending parameters, awaiting the execution of the remote service, then returning the result.

Both pvData and pvAccess have been implemented in C++ and Java, with additional bindings for Python [6].

SNS NEUTRON DATA

SNS neutron data consists of a list of pixels and time-of-flights as already described, combined with a sequential pulse number and the proton charge of the accelerator pulse that generated these neutrons.

The following pvData structure would be a direct representation:

```
// Sequential pulse number
uint64   pulse
```

ISBN 978-3-95450-148-9

```
// Proton Charge
double   proton_charge
// Event array, each element is
// time-of-flight & pixel
struct
{
   uint32  time_of_flight
   uint32  pixel
} events[]
```

Each neutron event naturally combines the affected detector pixel and the time of flight when the event occurred. The above data structure always provides each such event as a tuple of { time_of_flight, pixel }.

Some consumers, however, require only a subset of this tuple. A time-of-flight histogram only needs to inspect the time_of_flight elements, and a spatial X/Y histogram only the pixel elements. With the above structure they need to subscribe to the "events" array and thus always receive both the time of flight and pixel information.

The following data structure holds the same information, but allows clients to subscribe to just the information of interest:

```
// Time stamp for everything in this
// structure.
// timeStamp.userTag holds
// sequential pulse number
time_t timestamp

// Proton Charge
NTScalar proton_charge
  double value

// Time-of-Flight values for N neutron events
NTScalarArray time_of_flight
   uint[]  value

// Pixel IDs for N neutron events
NTScalarArray pixel
   uint[]  value
```

With this optimized data structure, all producers and consumers agree that the "time_of_flight" and "pixel" arrays always contain the same number of elements, because corresponding array elements constitute one neutron event.

A tool that accumulates the X/Y histogram can now subscribe to just the "pixel" element, receiving only this data and thus reducing the network traffic. Tools that require the complete information can still subscribe to the whole pvData structure.

In addition, this updated structure packages the sequential pulse number into the 'user' element of the normative time stamp type, and bases the proton charge, time of flight and pixel elements on Normative Types, allowing for compatibility with generic V4 client tools.

pvData allowed us to package the events as either an array-of-structures or a structure-of-arrays, and we chose the latter to optimize network traffic for the various use cases. For certain detector types or operating modes, the above structure can be extended with additional data elements, for example to transmit internal detector counts, which are used during calibration.

PERFORMANCE TESTS

We implemented a V4 server that emits data of the above format with sequential pulse numbers as well as a V4 client that subscribes to this data, counting the received elements and specifically detecting missed pulse numbers [7].

Figure 1 shows network traffic results from executing the demonstration server and client on a 1 gigabit Ethernet link. The server was sending 100 updates per second, varying the number of events in each of these updates. When each packet contains 150000 events, 100 times a second, this would equate transmitting 15 million SNS neutron events per second. Up to about 15 million events per second, the measured network traffic scaled linearly. There is very little overhead on the expected network traffic based on the underlying data size, proving that pvAccess efficiently serializes the pvData.

As we increased the event rate beyond 15 million SNS neutron events per second, the network traffic asymptotically approaches 95 MB per second. The client starts to indicate lost pulse updates. CPU loads of the server and client were only about 30%, indicating that we reached the limit of TCP on 1GigE.

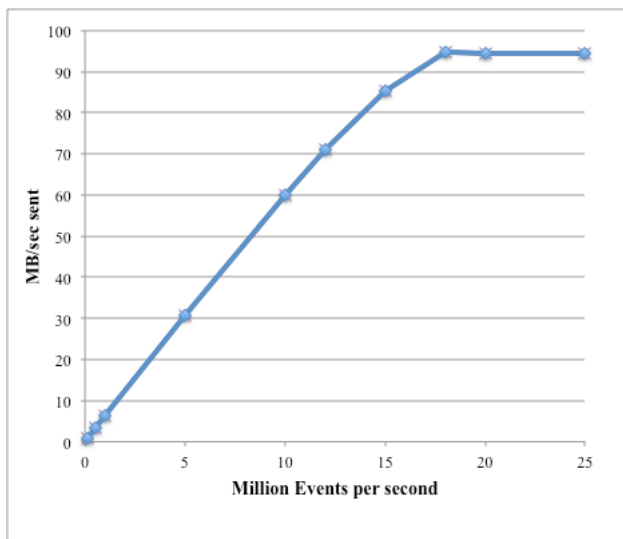

Figure 1: Network traffic on 1GigE network when sending various amounts of neutron events packaged into 100 updates per second.

On a 10GigE test setup, the simulated SNS neutron event rate could be increased to about 100 million events per second before reaching CPU load limits on the server.

GENERIC V4 TOOLS

EPICS V4 includes generic command line tools. The "pvinfo" command displays the IP address of the V4

ISBN 978-3-95450-148-9

server and the pvData structure of a V4 channel. This is useful for testing if a V4 server is online, and to check if the data contains the expected elements.

The "pvget" command can display the complete structure of a current value, or subscribe to selected elements. For example, the following command would show updates to the pixel array of the data sent by the V4 server, on a V4 channel named "neutrons", used in the performance tests:

```
pvget –m –r neutrons.pixel
```

USE OF V4 AT SNS BEAM LINES

As part of the update to the SNS detector control software to EPICS, the nED software [8] was developed to provide a pluggable framework for interfacing to the various detector configurations found at different beam lines. One nED module is a V4 server that publishes the SNS neutron data in the format we described.

There are two primary network clients for this data. One is a streaming data acquisition system that writes all neutron events to files for later analysis. The other is ADnED, an EPICS Area Detector driver which provides user displays and information for automation [9].

Ideas from the example server code [7] were used to create the V4 server in nED. Similarly, the example server was useful to create test data during the development of ADnED, allowing independent development and testing of these tools.

CONCLUSION

SNS neutron data can be packaged in pvData. By comparing different packaging options, we were able to optimize the network traffic based on the expected types of network clients.

The performance of pvAccess easily meets our requirement of about 10M events/sec on 1GigE and exceeds it on 10GigE.

While the original SNS beam line data acquisition software was limited to Microsoft Visual C++ on Windows, the EPICS V4 libraries for pvData and pvAccess are available on Linux, Mac OS and Windows, for C++, Java and Python. This allowed us to implement nED and ADnED in C++ on Linux to obtain the required performance, while test and calibration tools are often implemented in Python, offering more flexibility.

While the original SNS beam line neutron event data server and clients had no additional network test tools, we can now use the generic EPICS V4 command line tools to test if a server is online, or to monitor the data on the network.

At the time of writing, the SNS beam lines USANS, CORELLI, HYSPEC, VISION and SEQUOIA have been updated to use V4 pvData and pvAccess, along with nED and ADnED, for the critical first stages of neutron data transfer. Operation has been very reliable, especially considering that pvData and pvAccess are new developments. The SNS is the first facility to utilize these technologies in production systems on operating beam lines.

ACKNOWLEDGMENT

We thank Matej Sekoranja, Marty Kraimer and David Hickin for their assistance while learning about V4, their help when implementing the performance test code, and their fast response whenever we found problems in pvData and pvAccess.

REFERENCES

[1] http://www.aps.anl.gov/epics/
[2] R.E. Riedel, "Overview of Data Acquisition at the SNS", NOBUGS 2004, http://lns00.psi.ch/nobugs2004/papers/paper00055.pdf
[3] S.M.Hartman, "SNS Instrument Data Acquisition And Controls", ICALEPCS 2013, San Francisco, CA, USA.
[4] http://www.aps.anl.gov/epics/docs/CAproto.html
[5] L.R. Dalesio et al, "EPICS V4 Expands Support to Physics Application, Data Acquisition, and Data Analysis", ICALEPCS 2011, Grenoble, France.
[6] http://epics-pvdata.sourceforge.net
[7] K. Kasemir, EPICS V4 Example Server and Client for SNS Neutron Data, https://github.com/kasemir/EPICSV4Sandbox
[8] G.Guyotte, "nED – EPICS-based Neutron Data Acquisition and Detector Control Software", EPICS Meeting, FRIB, MSU, Lansing, MI, 2015.
[9] M. Pearson, "ADnED – V4 Neutron Event Data in areaDetector", EPICS Meeting, FRIB, MSU, Lansing, MI, 2015. https://github.com/areaDetector/ADnED

ISBN 978-3-95450-148-9