ISBN: 978-3-95450-221-9 ISSN: 2226-0358 doi:10.18429/JACoW-ICALEPCS2021-WEPV048

AN ARCHIVER APPLIANCE PERFORMANCE AND RESOURCES CONSUMPTION STUDY

R. Fernandes[†], H. Kocevar, S. Armanet, S. Regnell, European Spallation Source, Lund, Sweden

Abstract

At the European Spallation Source (ESS), 1.6 million signals are expected to be generated by a (distributed) control layer composed of around 1 500 EPICS IOCs. A substantial amount of these signals – i.e. PVs – will be stored by the Archiving Service, a service that is currently under development at the Integrated Control System (ICS) Division. From a technical point of view, the Archiving Service is implemented using a software application called the Archiver Appliance. This application, originally developed at SLAC, records PVs as a function of time and stores these in its persistence layer. A study based on multiple simulation scenarios that model ESS (future) *modus operandi* has been conducted by ICS to understand how the Archiver Appliance performs and consumes resources (e.g. RAM memory) under disparate workloads.

INTRODUCTION

The ICS Division at ESS is mandated to deliver a system to control both its accelerator and end-station instruments. To create the system, the open-source framework EPICS [1] was chosen. With worldwide usage, EPICS allows the creation of Input/Output Controllers (IOCs) which software applications (e.g. Archiver Appliance, CS-Studio) may consume (i.e. connect to) to tackle domain specific businesses (e.g. signals archiving, signals displaying).

Typically, an IOC is an executable (i.e. software process) that utilizes resources from EPICS modules to interface (logical or physical) devices and exposes their input and output signals as Process Variables (PVs). Eventually, an IOC may also implement logic to control these devices.

A PV is a named piece of data, usually associated with devices to represent input and output signals (e.g. status, setpoint). A PV can be read, written or monitored by applications and tools using the Channel Access (CA) library.

Given that a significant number of PVs will be archived by the Archiver Appliance [2] at ESS, the present paper introduces a study to understand how this application performs when storing (i.e. writing) and retrieving (i.e. reading) PV data into and from its persistence layer thanks to a panoply of simulation scenarios designed to stress test it. In addition, the paper explores how the Archiver Appliance consumes resources (e.g. RAM memory) when handling these scenarios.

SIMULATION SCENARIOS

Thanks to discussions with domain experts to understand the type of data and volume important to test the Archiver Appliance with, four dimensions were identified along with relevant ranges of values. This helped specify simulation scenarios close to what ICS will likely face in terms of PV archiving needs and requirements from end-users, thus testing the application in a (more) meaningful way. The dimensions and ranges of values are:

- Number of PV waveforms: 1, 100, 1 000, 10 000
- Data points (per waveform): 1 000, 10 000, 100 000
- Data type: integer (4 bytes), double (8 bytes)
- Update frequency: 1 Hz, 14 Hz

Based on these, 48 simulation scenarios were specified (see Table 1).

Table 1: Simulation Scenarios

Scenario ID	Number Waveforms	Data Points	Data Type	Update Frequency
AAPS-0010	1	1 000	Integer	1
AAPS-0020	1	1 000	Integer	14
AAPS-0030	1	1 000	Double	1
AAPS-0040	1	1 000	Double	14
AAPS-0050	1	10 000	Integer	1
AAPS-0060	1	10 000	Integer	14
AAPS-0070	1	10 000	Double	1
AAPS-0080	1	10 000	Double	14
AAPS-0090	1	100 000	Integer	1
AAPS-0100	1	100 000	Integer	14
AAPS-0110	1	100 000	Double	1
AAPS-0120	1	100 000	Double	14
AAPS-0210	100	1 000	Integer	1
AAPS-0220	100	1 000	Integer	14
AAPS-0230	100	1 000	Double	1
AAPS-0240	100	1 000	Double	14
AAPS-0250	100	10 000	Integer	1
AAPS-0260	100	10 000	Integer	14
AAPS-0270	100	10 000	Double	1
AAPS-0280	100	10 000	Double	14
AAPS-0290	100	100 000	Integer	1
AAPS-0300	100	100 000	Integer	14
AAPS-0310	100	100 000	Double	1
AAPS-0320	100	100 000	Double	14
AAPS-0410	1 000	1 000	Integer	1
AAPS-0420	1 000	1 000	Integer	14
AAPS-0430	1 000	1 000	Double	1
AAPS-0440	1 000	1 000	Double	14
AAPS-0450	1 000	10 000	Integer	1
AAPS-0460	1 000	10 000	Integer	14
AAPS-0470	1 000	10 000	Double	1
AAPS-0480	1 000	10 000	Double	14
AAPS-0490	1 000	100 000	Integer	1
AAPS-0500	1 000	100 000	Integer	14
AAPS-0510	1 000	100 000	Double	1
AAPS-0520	1 000	100 000	Double	14
AAPS-0610	10 000	1 000	Integer	1
AAPS-0620	10 000	1 000	Integer	14
AAPS-0630	10 000	1 000	Double	1
AAPS-0640	10 000	1 000	Double	14

[†] ricardo.fernandes@ess.eu

AAPS-0650	10 000	10 000	Integer	1
AAPS-0660	10 000	10 000	Integer	14
AAPS-0670	10 000	10 000	Double	1
AAPS-0680	10 000	10 000	Double	14
AAPS-0690	10 000	100 000	Integer	1
AAPS-0700	10 000	100 000	Integer	14
AAPS-0710	10 000	100 000	Double	1
AAPS-0720	10 000	100 000	Double	14

TOOLS

A tool named Prometheus was used to collect metrics to help evaluate the performance of the Archiver Appliance when storing and retrieving the aforementioned simulation scenarios into and from its persistence layer, as well as the stress – in terms of resources consumption – these produce on the physical machines involved in the study (i.e. MicroTCA running the IOC, Supermicro running the Archiver Appliance, and another Supermicro running a Python tool to retrieve data).

Metrics collected by Prometheus were then aggregated/displayed by another tool named Grafana. This tool can synthetize vast amounts of collected metrics very quickly in a graphical fashion through personalized dashboards – thus providing highly insightful information.

STORAGE STUDY

Environment

To enable a proper study of the Archiver Appliance in terms of PV data storage performance and resources consumption, three components were configured and used: a producer, a consumer, and a network connecting the two.

Producer The producer of PV data is an EPICS IOC running in a dedicated MicroTCA machine. The IOC was built with EPICS base version 7.0.3.1 as a 64 bit executable and uses the following EPICS modules:

- asyn (version 4.7)
- sncseq (version 2.2.8)
- procserv (version 2.8.0)
- aatest (version 1.0)
- aatestsioc (version 1.0)

In detail, the IOC is based on the EPICS asyn module and it extends the asynPortDriver C++ class. Several asyn parameters were implemented to provide control and monitoring of the PV waveforms. The IOC startup specifies how many waveforms are generated along with the number of data points for each waveform. Two types of waveforms are provided: integer (4 bytes) and double (8 bytes). On each waveform update only the first data point is modified – this is sufficient to trigger the PV update, consequently forcing the Archiver Appliance to archive the new data. It is also possible to control the delay between two consecutive waveform updates allowing the simulation of different update frequencies (e.g. 14 Hz). Multiple CPU cores may be utilized by running several IOCs at the same time, each generating/serving a fraction of the PV waveforms – crucial for running the "heaviest" simulation scenarios, which would not have been possible otherwise.

The MicroTCA machine runs CentOS 7 64 bit and has the following main characteristics:

- Manufacturer: Schroff
- Model: 3U
- CPU: Intel Xeon E3-1505M 2.80 GHz (4 cores)
- RAM: 16 GB
- Chassis: MTCA NAT-MCH

Consumer The consumer of PV data is an instance of the Archiver Appliance running in a dedicated physical storage server. The Archiver Appliance is a Java-based application that automatically records PV data in function of time in its persistence layer using Protocol Buffers [3], an extensible mechanism for serializing structured data. The instance refers to version 0.0.1 (Fall 2018 Release) of the Archiver Appliance and uses Java version 1.8.0_191 with a heap size of 16 GB. The storage server runs CentOS 7 64 bit and has the following main characteristics**:**

- Manufacturer: Supermicro
- Model: SSG-6029P-E1CR12L
- CPU: 2 x Intel 6126 2.60 GHz (12 cores per CPU)
- RAM: 128 GB
- Storage: ZFS 0.7.12 composed of 12 x 6 TB HDD (NL-SAS) and 2 x NVMe Intel P4600 4 TB

Network The network is based on a Gigabit fiber optic cable configured to transmit standard Ethernet frames (with a payload equal to 1 500 bytes) at a maximum throughput of 1 Gb/s between the producer and consumer.

Metrics

The following resources were monitored and metrics about their usages collected every five seconds while running a certain simulation scenario: CPU (load in percentage (%)), RAM (usage in gigabyte (GB)) and Network (traffic in megabit per second (Mb/s)). In addition, at the end of running the scenario, the following metrics were calculated: Disk (usage in gigabyte (GB)) and Dropped (amount of PV frames dropped in percentage (%)).

Methodology

The main method followed during the storage performance and resources consumption study of the Archiver Appliance was to have each of the 48 simulation scenarios configured to generate either 3 600 frames or 50 400 frames in a run, depending on whether the update frequency is 1 Hz or 14 Hz, respectively. This meant that each scenario would theoretically run for exactly 60 minutes.

Even though no special cache was (explicitly) implemented/configured in the MicroTCA and Supermicro machines, or in the Archiver Appliance, each simulation scenario ran with interval spaces of at least 30 minutes. This pause not only increased the confidence that the activities of a previous scenario (e.g. CPU usage) would not impact the next scenario (thus potentially distorting the collected metrics) but also eased the delimitation of time range queries when subsequently displaying and analysing metrics in Grafana. Moreover, to make sure that collected metrics would not be disturbed by an unknown/undesired state of the control layer, the IOC was systematically restarted before running a new scenario.

For simulation scenarios involving 1, 100 and 1 000 waveforms, only one IOC was successful in generating these. For scenarios involving 10 000 waveforms, 8 IOCs were launched simultaneously, each generating 1 250 waveforms. The main reason for this segmentation was to spread the computation cost across multiple CPU cores (of the MicroTCA) as only one IOC trying to manage this very large number of waveforms revealed impracticable (i.e. only one IOC would saturate the CPU core where it ran and, consequently, stop responding).

To help determine the rate of data effectively stored by the Archiver Appliance, a tool was developed at ICS. This tool, implemented in Python, retrieves data stored in the Archiver Appliance and calculates the dropping factor (i.e. frames that have not been archived but should have been) of a certain PV for a given time range. Specifically, it subtracts from the number 1 the result of the division of the number of frames stored in the Archiver Appliance by the number of frames that are theoretically supposed to be stored in this application. Since the retrieval of archived data for all the PVs involved in a particular scenario revealed unfeasible (prohibitively time consuming), the solution found was to retrieve only a subset of these PVs in an evenly distributed fashion (as it was assumed that frames were being archived or not (i.e. dropped) in a normal distribution fashion – thus the dropping factor of PVs would be similar to each other).

In addition, the tool consumes a CSV-based text file containing information about each simulation scenario to know how to process it. One of the main advantages of this approach is that, in case of need, the text file can easily be extended with additional scenarios without the burden to have to modify the tool to cope with these.

Results

Table 2 contains the results of the performance and resources consumption of the Archiver Appliance from a storage point of view, while [4] provides additional details.

Table 2: Results of the Storage Study (Based on Collected Metrics)

Scenario ID	CPU Producer	RAM Producer	CPU Consumer	RAM Consumer	Disk Consumer	Network Traffic	Dropped Frames
AAPS-0010	1%	2.6 GB	< 1%	13.8 GB	< 0.1 GB	< 1 Mb/s	0%
AAPS-0020	6%	2.6 GB	< 1%	13.8 GB	0.2 GB	1 Mb/s	0%
AAPS-0030	1%	2.6 GB	< 1%	13.8 GB	< 0.1 GB	< 1 Mb/s	0%
AAPS-0040	6%	2.6 GB	< 1%	14.9 GB	0.4 GB	1 Mb/s	0%
AAPS-0050	1%	2.6 GB	< 1%	14.1 GB	0.1 GB	< 1 Mb/s	0%
AAPS-0060	10%	2.6 GB	< 1%	15.5 GB	1.9 GB	5 Mb/s	0%
AAPS-0070	1%	2.6 GB	< 1%	14.4 GB	0.3 GB	1 Mb/s	0%
AAPS-0080	10%	2.6 GB	< 1%	22.3 GB	3.8 GB	9 Mb/s	0%
AAPS-0090	1%	2.6 GB	< 1%	24.9 GB	1.3 GB	3 Mb/s	0%
AAPS-0100	10%	2.5 GB	< 1%	56.1 GB	18.7 GB	47 Mb/s	0%
AAPS-0110	1%	2.6 GB	< 1%	26.2 GB	2.7 GB	7 Mb/s	0%
AAPS-0120	10%	2.5 GB	3%	72.1 GB	37.8 GB	94 Mb/s	0%
AAPS-0210	1%	2.4 GB	< 1%	60.4 GB	1.3 GB	3 Mb/s	0%
AAPS-0220	9%	2.5 GB	< 1%	72.6 GB	18.9 GB	46 Mb/s	0%
AAPS-0230	1%	2.4 GB	< 1%	61.1 GB	2.7 GB	7 Mb/s	0%
AAPS-0240	9%	2.5 GB	2%	74.5 GB	37.8 GB	91 Mb/s	0%
AAPS-0250	1%	2.5 GB	< 1%	72.4 GB	13.4 GB	34 Mb/s	0%
AAPS-0260	9%	2.5 GB	5%	72.1 GB	188.0 GB	448 Mb/s	0%
AAPS-0270	2%	2.5 GB	1%	71.1 GB	27.0 GB	67 Mb/s	0%
AAPS-0280	11%	2.5 GB	7%	86.9 GB	350.0 GB	873 Mb/s	< 1%
AAPS-0290	2%	2.6 GB	1%	84.3 GB	138.0 GB	330 Mb/s	0%
AAPS-0300	7%	2.6 GB	6%	87.5 GB	403.0 GB	983 Mb/s	69%
AAPS-0310	3%	2.6 GB	12%	87.4 GB	242.0 GB	660 Mb/s	< 1%
AAPS-0320	9%	2.6 GB	19%	87.6 GB	418.0 GB	983 Mb/s	90%
AAPS-0410	2%	2.4 GB	< 1%	87.7 GB	13.0 GB	33 Mb/s	0%
AAPS-0420	11%	2.4 GB	1%	92.5 GB	185.0 GB	270 Mb/s	0%
AAPS-0430	2%	2.4 GB	< 1%	90.0 GB	26.0 GB	67 Mb/s	0%
AAPS-0440	12%	2.4 GB	6%	86.7 GB	370.0 GB	520 Mb/s	0%
AAPS-0450	2%	2.5 GB	1%	90.3 GB	131.0 GB	320 Mb/s	0%
AAPS-0460	3%	2.5 GB	9%	88.2 GB	263.0 GB	640 Mb/s	67%
AAPS-0470	3%	2.5 GB	9%	88.2 GB	263.0 GB	640 Mb/s	0%
AAPS-0480	10%	2.6 GB	11%	90.4 GB	517.0 GB	983 Mb/s	80%
AAPS-0490	4%	3.1 GB	7%	89.8 GB	382.0 GB	983 Mb/s	68%

AAPS-0500	10%	3.4 GB	11%	90.7 GB	722.0 GB	983 Mb/s	98%
AAPS-0510	4%	3.6 GB	9%	88.8 GB	134.0 GB	983 Mb/s	97%
AAPS-0520	11%	3.6 GB	6%	90.9 GB	142.0 GB	983 Mb/s	100%
AAPS-0610	10%	2.5 GB	4%	79.6 GB	140.0 GB	315 Mb/s	0%
AAPS-0620	47%	2.7 GB	9%	85.3 GB	682.0 GB	983 Mb/s	66%
AAPS-0630	8%	2.5 GB	9%	91.0 GB	250.0 GB	634 Mb/s	5%
AAPS-0640	47%	2.6 GB	4%	84.0 GB	60.0 GB	983 Mb/s	98%
AAPS-0650	7%	3.6 GB	9%	85.1 GB	340.0 GB	976 Mb/s	72%
AAPS-0660	51%	3.6 GB	12%	86.0 GB	530.0 GB	930 Mb/s	95%
AAPS-0670	7%	3.6 GB	11%	91.2 GB	260.0 GB	873 Mb/s	91%
AAPS-0680	56%	3.6 GB	10%	86.8 GB	460.0 GB	899 Mb/s	99%
AAPS-0690	11%	9.0 GB	16%	92.1 GB	210.0 GB	983 Mb/s	99%
AAPS-0700	78%	9.0 GB	11%	86.1 GB	360.0 GB	983 Mb/s	99%
AAPS-0710	N/A	N/A	N/A	N/A	N/A	N/A	N/A
AAPS-0720	N/A	N/A	N/A	N/A	N/A	N/A	N/A

RETRIEVAL STUDY

Environment

To enable a proper study of the Archiver Appliance in terms of PV data retrieval performance and resources consumption, three components were configured and used: a producer, a consumer, and a network connecting the two.

Producer The producer of PV data is an instance of the Archiver Appliance running in a dedicated physical storage server. The instance refers to version 0.0.1 (Fall 2018 Release) of the Archiver Appliance. The storage server is a Supermicro machine and runs CentOS 7 64 bit. See subsection Environment (in section Storage Study) for additional details about the Archiver Appliance and storage server.

Consumer The consumer of PV data is a Python tool that retrieves data from the Archiver Appliance and runs in a dedicated physical machine. Thanks to its multi-threading architecture, the tool may simulate multiple clients (e.g. CS-Studio) retrieving data from the Archiver Appliance simultaneously (i.e. in parallel). The machine used for the Python tool runs CentOS 7 64 bit and has the following main characteristics:

- Manufacturer: Supermicro
- Model: SYS-1018R-WC0R
- CPU: 2 x Intel E5-2637 3.50 GHz (4 cores per CPU)
- RAM: 64 GB

Network The network is based on a Gigabit fiber optic cable configured to transmit standard Ethernet frames (with a payload equal to 1 500 bytes) at a maximum throughput of 1 Gb/s between the producer and consumer.

Metrics

The following resources were monitored and metrics about their usages collected every five seconds while running a certain simulation scenario: CPU (load in percentage (%)), RAM (usage in gigabyte (GB)) and Network (traffic in megabit per second (Mb/s)). In addition, at the end of running the scenario, the following metrics were calculated: Data Retrieved (in gigabyte (GB)) and Retrieval Time (in second (s)).

Methodology

The main method followed during the retrieval performance and resources consumption study was to retrieve PV data stored in the Archiver Appliance using a Python tool created for this purpose. It retrieved data using a RESTful interface provided by the Archiver Appliance. Although the tool was prepared to retrieve data in different formats – namely: TXT, CSV, JSON and RAW – it was decided that the retrieval study should be based on CSV since this format is the one that users will likely use when retrieving data, as well as being very suitable as a baseline/reference when compared with other formats (due to its simplicity).

Through a configuration parameter, the Python tool was able to launch multiple threads at the same time, each retrieving PV data independently from remaining threads. The idea was to simulate multiple clients (i.e. people and/or applications) retrieving data simultaneously. In this retrieval study, metrics were collected while running the tool with 1 thread, 10 threads and 100 threads.

Since the simulation scenarios based on 100, 1 000 and 10 000 PV waveforms are essentially the same as the scenarios based on 1 waveform (in terms of data points, data type and update frequency – the only difference being the number of waveforms), it was decided to only retrieve the latter scenarios from the Archiver Appliance when conducting the retrieval study.

Results

Table 3 contains the results of the performance and resources consumption of the Archiver Appliance from a retrieval point of view, while [4] provides additional details.

Table 3: Results of the Retrieval Study (Based on Collected Metrics)

Scenario ID	Number Threads	CPU Producer	RAM Producer	CPU Consumer	RAM Consumer	Network Traffic	Data Retrieved	Retrieval Time
AAPS-0010	1	N/A	N/A	N/A	N/A	N/A	< 0.1 GB	< 1 s
AAPS-0010	10	N/A	N/A	N/A	N/A	N/A	< 0.1 GB	< 1 s

AAPS-0010	100	13%	71.7 GB	4%	2.0 GB	711 Mb/s	0.7 GB	5 s
AAPS-0020	1	0%	71.7 GB	1%	1.9 GB	35 Mb/s	0.1 GB	6 s
AAPS-0020	10	6%	71.7 GB	2%	2.0 GB	429 Mb/s	0.1 GB	8 s
AAPS-0020	100	14%	71.7 GB	5%	2.0 GB	984 Mb/s	9.5 GB	84 s
AAPS-0030	1	N/A	N/A	N/A	N/A	N/A	< 0.1 GB	< 1 s
AAPS-0030	10	N/A	N/A	N/A	N/A	N/A	0.1 GB	1 s
AAPS-0030	100	12%	71.7 GB	5%	2.0 GB	950 Mb/s	1.4 GB	10 s
AAPS-0040	1	1%	71.7 GB	2%	2.0 GB	69 Mb/s	0.2 GB	10 s
AAPS-0040	10	1%	71.7 GB	1%	1.9 GB	81 Mb/s	1.9 GB	16 s
AAPS-0040	100	12%	71.7 GB	5%	2.0 GB	984 Mb/s	18.9 GB	164 s
AAPS-0050	1	1%	71.7 GB	2%	1.9 GB	31 Mb/s	0.1 GB	5 s
AAPS-0050	10	4%	71.7 GB	2%	2.0 GB	302 Mb/s	0.7 GB	5 s
AAPS-0050	100	14%	71.7 GB	5%	2.0 GB	984 Mb/s	6.7 GB	57 s
AAPS-0060	1	2%	71.7 GB	2%	2.0 GB	144 Mb/s	1.0 GB	59 s
AAPS-0060	10	14%	71.7 GB	4%	2.0 GB	984 Mb/s	9.4 GB	84 s
AAPS-0060	100	14%	71.7 GB	5%	2.0 GB	984 Mb/s	94.0 GB	826 s
AAPS-0070	1	2%	71.7 GB	2%	1.9 GB	49 Mb/s	0.1 GB	7 s
AAPS-0070	10	12%	71.7 GB	4%	2.0 GB	978 Mb/s	1.3 GB	11 s
AAPS-0070	100	13%	71.7 GB	5%	2.0 GB	984 Mb/s	13.4 GB	117 s
AAPS-0080	1	2%	71.7 GB	2%	2.0 GB	158 Mb/s	1.9 GB	107 s
AAPS-0080	10	12%	71.7 GB	5%	2.0 GB	984 Mb/s	18.8 GB	169 s
AAPS-0080	100	13%	71.7 GB	5%	2.0 GB	984 Mb/s	187.9 GB	1 646 s
AAPS-0090	1	2%	71.7 GB	2%	2.0 GB	146 Mb/s	0.7 GB	41 s
AAPS-0090	10	14%	71.7 GB	5%	2.0 GB	984 Mb/s	6.7 GB	60 s
AAPS-0090	100	15%	71.7 GB	5%	2.0 GB	984 Mb/s	67.1 GB	577 s
AAPS-0100	1	2%	71.7 GB	2%	2.0 GB	142 Mb/s	9.4 GB	595 s
AAPS-0100	10	14%	71.7 GB	4%	2.0 GB	984 Mb/s	93.9 GB	840 s
AAPS-0100	100	15%	73.7 GB	5%	2.0 GB	984 Mb/s	938.9 GB	8 283 s
AAPS-0110	1	2%	73.7 GB	2%	2.0 GB	151 Mb/s	1.3 GB	79 s
AAPS-0110	10	13%	73.7 GB	5%	2.0 GB	984 Mb/s	13.4 GB	118 s
AAPS-0110	100	16%	75.8 GB	5%	2.0 GB	984 Mb/s	134.1 GB	1 153 s
AAPS-0120	1	2%	75.8 GB	2%	2.0 GB	151 Mb/s	18.8 GB	1 122 s
AAPS-0120	10	13%	75.8 GB	4%	2.0 GB	984 Mb/s	187.8 GB	1 698 s
AAPS-0120	100	16%	77.1 GB	5%	2.0 GB	984 Mb/s	1 877.7 GB	16 304 s

RESOURCES SATURATION STUDY

The results based on the metrics collected during the storage study (see Table 1) and retrieval study (see Table 2) make it evident that the network is the resource that reduced the success to store data and increased the time it took to retrieve data due to saturation. In this section, an attempt is therefore made to understand 1) the maximum number of PVs that can safely be archived (i.e. no data is dropped) by the Archiver Appliance and 2) the maximum number of threads (i.e. people and/or applications) that can concurrently retrieve data from this application if the network is never saturated (i.e. is not a bottleneck) and in function of the remaining two other resources, namely: CPU and RAM. Consequently, the attempt – which is based on extrapolations using regression lines – focuses on understanding the saturation of these two resources of the storage server. See subsection Environment (in section Storage Study) for additional details about this server. This understanding may help better manage/implement the underlying infrastructure if, for example, a system owner needs to archive a set of PVs in a proper way – by, e.g., procuring a storage server with particular characteristics – or if already deployed resources are sufficient (performance and storage wise) to cope with these PVs.

Storage

Based on the metrics collected during the PV data storage study, regression lines were calculated to model both the CPU load and the RAM usage of the storage server in function of the number of PV waveforms stored (i.e. archived) concurrently. For the calculations of these regression lines, a (single) PV waveform was assumed to have the following characteristics:

- Data points: 37 000 ((1 000 + 10 000 + 100 000) / 3)
- Data point size: 6 bytes ((4 + 8) / 2)
- Update frequency: 7.5 Hz ((1 + 14) / 2)

CPU The regression line that models the CPU load of the storage server in function of the number of PV waveforms that it may store concurrently is the following:

*cpu_load = 0.00001 * number_waveforms + 0.02463*

This means it takes around 100 000 waveforms being stored concurrently by the Archiver Appliance to saturate the CPU of the storage server. Above this number, the CPU becomes a bottleneck forcing the application to start dropping (i.e. not archiving) PV data.

RAM The regression line that models the RAM usage of the storage server in function of the number of PV waveforms that it may store concurrently is the following:

$$ram\_usage = 0.00003 * number\_waveforms + 0.40192$$

This means it takes around 33 333 waveforms being stored concurrently by the Archiver Appliance to saturate the RAM memory of the storage server. Above this number, the RAM memory becomes a bottleneck forcing the application to start dropping (i.e. not archiving) PV data.

Retrieval

Based on the metrics collected during the PV data retrieval study, regression lines were calculated that model both the CPU load and the RAM usage of the storage server in function of the number of threads (i.e. people and/or applications) that it may serve concurrently.

CPU The regression line that models the CPU load of the storage server in function of the number of threads that it may serve concurrently is the following:

$$cpu\_load = 0.00118 * number\_threads + 0.02739$$

This means it takes around 847 threads to retrieve data concurrently from the Archiver Appliance to saturate the CPU of the storage server. Above this number, the CPU becomes a bottleneck forcing the application to increase the time it takes to satisfy requests.

RAM The regression line that models the RAM usage of the storage server in function of the number of threads that it may serve concurrently is the following:

$$ram\_usage = 0.0001 * number\_threads + 0.55964$$

This means it takes around 9 999 threads to retrieve data concurrently from the Archiver Appliance to saturate the RAM memory of the storage server. Above this number, the RAM memory becomes a bottleneck forcing the application to increase the time it takes to satisfy requests.

CONCLUSION

The results of this study imply that the Archiver Appliance appears reliable to cope with the storage (i.e. writing) of a high number of PV waveforms. This will be crucial at ESS since several thousands of devices (interfaced in EPICS) will have their control signals exposed through PV waveforms, including some of long sizes.

Thanks to the collected metrics, it can also be concluded that the Archiver Appliance consumes CPU and RAM memory in function of the number of PVs to be archived (as expected). In no simulation scenario that ran, did the CPU or the RAM of the machine running this application represent bottlenecks. Moreover, scenarios AAPS-0710 and AAPS-0720 were not executed due to the Archiver Appliance being unable to handle all the PVs (with an erratic behavior observed). Despite restarting the IOC and re-installing the Archiver Appliance, this behavior still persisted. The reason for this is unclear but we hypothesize that the sheer number of PVs and long sizes were the likely cause.

Due to the study, it is now possible to better predict/calculate if existing deployed resources (e.g. network, storage space, archiver appliance instances) are able to successfully cope with a request to, e.g., archive a new system or if additional resources are needed and provided in a preemptive fashion (thus without having to wait for PV data to be dropped to only then add more resources).

Finally, a detailed report about the performance of the Archiver Appliance and how it consumes resources to handle the simulation scenarios can be accessed in [4].

ACKNOWLEDGEMENT

The authors would like to thank everyone who helped enable the Archiver Appliance performance and resources consumption study at ESS, in particular Benjamin Bertrand, Johan Christensson, Alessio Curri, Fábio dos Santos Alves and Faye Chicken for their effective support.

REFERENCES

[1] EPICS, `https://en.wikipedia.org/wiki/EPICS`

[2] M. Shankar *et al.*, “The EPICS Archiver Appliance”, in Proc. ICALEPCS 2015, Melbourne, Australia, October 2015, paper WEPGF030.

[3] Protocol Buffers, `https://en.wikipedia.org/wiki/Protocol_Buffers`

[4] Archiver Appliance Performance and Resources Consumption Study, `https://gitlab.esss.lu.se/ics-software/archiver-appliance-performance-and-resources-consumption-study`