SLAC-PUB-16456

REAL-TIME PERFORMANCE IMPROVEMENTS AND CONSIDERATION OF PARALLEL PROCESSING FOR BEAM SYNCHRONOUS ACQUISITION (BSA)*

K. H. Kim#, S. Allison, T. Straumann, E. Williams
SLAC National Accelerator Laboratory, Menlo Park, CA 94025, USA

Abstract

Beam Synchronous Acquisition (BSA) provides a common infrastructure for aligning data to each individual beam pulse, as required by the Linac Coherent Light Source (LCLS)[1]. BSA allows 20 independent acquisitions simultaneously for the entire LCLS facility and is used extensively for beam physics, machine diagnostics and operation. BSA is designed as a part of LCLS timing system [2,3] and is currently an EPICS record based implementation, allowing timing receiver EPICS applications to easily add BSA functionality to their own record processing. However the lack of real-time performance of EPICS [4] record processing and the increasing number of BSA devices has brought real-time performance issues. The major reason for the performance problem is due to the lack of separation between time-critical BSA upstream processing and non-critical downstream processing. We are improving BSA with thread level programming, breaking the global lock in each BSA device, adding a queue between upstream and downstream processing, and moving out the non-critical downstream to a lower priority worker thread. We are also investigating the use of multiple worker threads for parallel processing in Symmetric Multi-Processor (SMP) system.

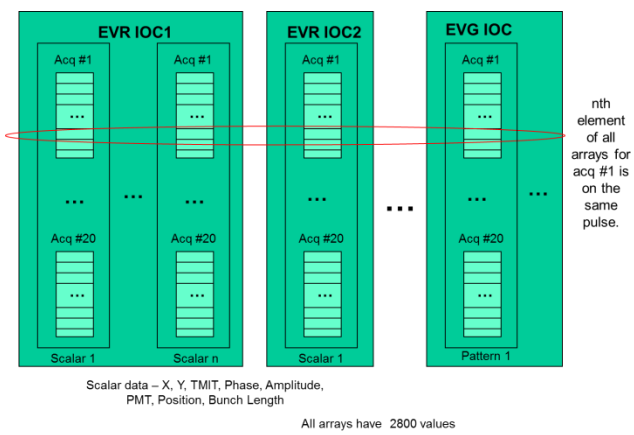

Figure 1: Data Acquisition across IOCs

BEAM SYNCHRONOUS ACQUISITION

BSA has been designed as a part of event system in LCLS-I to acquire all of beam dependent scalars across multiple IOCs in the entire accelerator facility on the same beam pulse, allowing correlation analysis using the pulse by pulsed aligned acquisition data (Figure 1). BSA acquires up to 2,800 values per scalar in one acquisition request; each value of the 2,800 can be an average of up to 1,000 values. It also provides RMS values and other statistics. The BSA can process 20 different acquisitions simultaneously [5].

BSA is implemented in three parts. A user request for an acquisition is done by EPICS CA client. Data gathering is processed on the Event Generator (EVG) and Event Receiver (EVR) IOCs. When gathering is finished, access of prepared data waiting on IOCs is done by CA clients, with checks for a good acquisition.

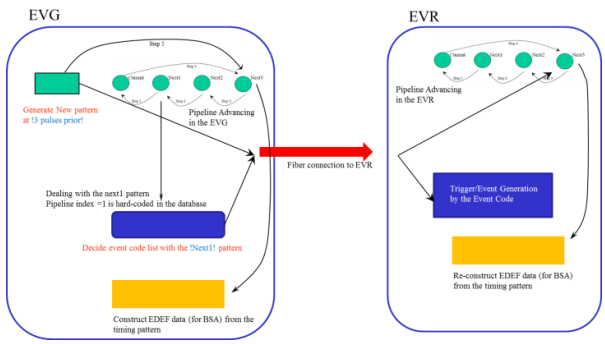

Figure 2: 360Hz tasks in EVG and EVR

360Hz Task in EVG

Acquisition setup and start requests done on the EVG IOC. The EVG IOC performs 360Hz checking and user notification when a BSA is finished. A 360Hz event task wakes up on an interrupt from a clock synched AC powerline zero-crossing which also provides timeslot for the event system. The event task is the heart of EVG IOC, generating a timing pattern for 3 pulses ahead and schedules timing events for the next pulse. The event task also checks for a match between the new pulse's timing pattern, beam code, and each active acquisition for BSA. It keeps a count of the number of measurements and the number of values in the current average per acquisition. One part of the timing pattern represents which acquisitions are matched. The timing pattern is broadcasted to EVRs and contains pulse id, timestamp, and additional BSA information. The pulse information is pipelined, thus the timing pattern is for 3 pulses ahead (Figure 2).

*Work supported by the the U.S. Department of Energy, Office of Science under Contract DE-AC02-76SF00515 for LCLS I and LCLS II.
#khkim@slac.stanford.edu

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Basic Energy Sciences, under Contract No. DE-AC02-76SF00515.

360Hz Task in EVR

The Event Definition (EDEF) bits included in 360Hz data are sent by EVG to EVR. EVR IOC caches the data on the EVR data interrupt. The EVR IOC 360Hz event task is activated on the next fiducial interrupt (actually, event code 1), copying the data to the end of the timing pipeline, and then shifting the pipeline. Finally, the 360Hz task updates EDEF table which is used by BSA processing done later in the same pulse (Figure 3).

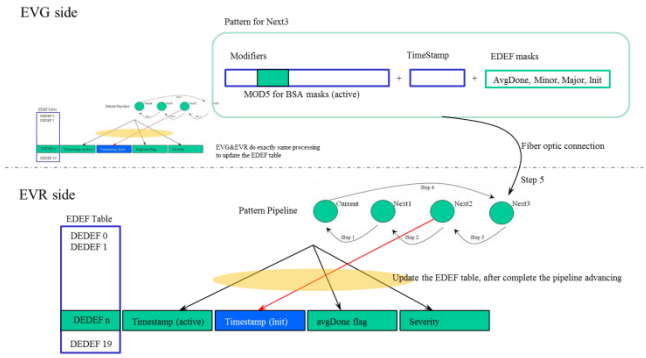

Figure 3: 360Hz tasks and EDEF table update

BSA Processing

BSA processing is driven by data source processing variable (PV). The data source PV provides new data with timestamp and triggers BSA processing. The data receptor PV receives new data and timestamp, and then checks the timestamp against the EDEF tables. If there is a timestamp match, the data is included in the BSA buffer. The data receptor PV calculates RMS and average, if required, and then requests BSA record processing to update the BSA buffer PV. Each BSA record then delivers data to a compress record.

IMPROVEMENT REQUIRED

LCLS-I has around 1,150 BSA PVs and more than 980 BSA PVs work successfully. However, around 160 PVs sometimes fail. The reason of BSA failure varies, and is sometimes due to misconfigured data source PVs, and rates. Some data sources PVs deliver the data too late without enough time to process BSA. The misconfiguration and lazy data source PVs are fixed, as they are found, to resolve the BSA problems.

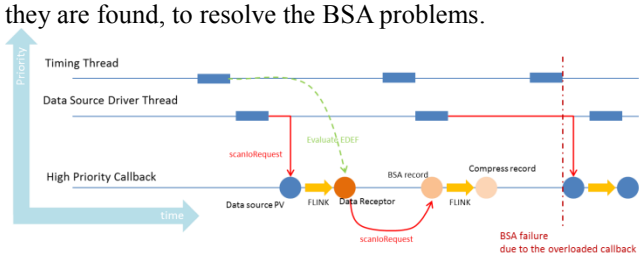

Figure 4: BSA fail scenario

We also found, under test conditions, that some BSA PVs fail even when there is an enough time budget. Sometimes the data source PVs are not completed within the expected time, and get a wrong timestamp. Thus, we discovered an unexpected delay on the data source PV and BSA processing, due to EPICS record processing. The data source and BSA processing are processed by high priority callback thread in the EPICS IOC. The callback thread also processes other records and results in unexpected delay (Figure 4).

BSA IMPROVEMENT

EVR side BSA processing has been implemented in two parts – data reception and BSA record processing – in the current implementation. The data receptor PV is triggered by the data source PV through a forward link when new data is ready. The data receptor matches timestamp to decide if the data goes into the BSA buffer, and performs RMS and average calculations, and then requests BSA record processing to update BSA data buffer implemented in a compress record. The timestamp matching in the data receptor PV is the only time-critical part of BSA processing. It cannot be delayed though the rest of the processing can be delayed if we do not lose information.

If we separate out the time-critical part, and process it immediately after data is ready, the rest is queued up for a lower priority thread. The lower priority thread takes care of the non-time-critical parts and finally updates BSA data buffer whenever it can get CPU time (Figure 5).

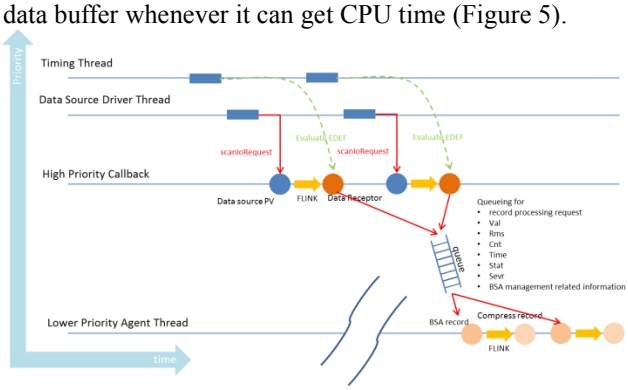

Figure 5: Processing timeline for BSA improvement

We provide a new API for this improvement. The new API is called by the device driver of the data source. In this case, the time-critical part is run in the driver context. We keep the EPICS record interface – data receptor PV – for backward compatibility. In this case, the time-critical part is run by the high priority callback thread. But, the non-time-critical part is run by the lower priority thread.

The new EPICS base R3.15 provides parallel callbacks. We can parallelize the high priority callback with the new feature. The improvement also allows multiple lower priority agent threads for the non-time-critical parts. The multiple threads share a single queue to receive BSA information from the upstream time-critical part (Figure 6). It improves performance in Symmetric Multi-Processor (SMP) system. Recently, we moved to the linuxRT [6] operating system from RTEMS [7] and the new platform provides the multi-core system. The parallel

callbacks and multiple lower priority agent threads bring a performance improvement for the multi-core system.

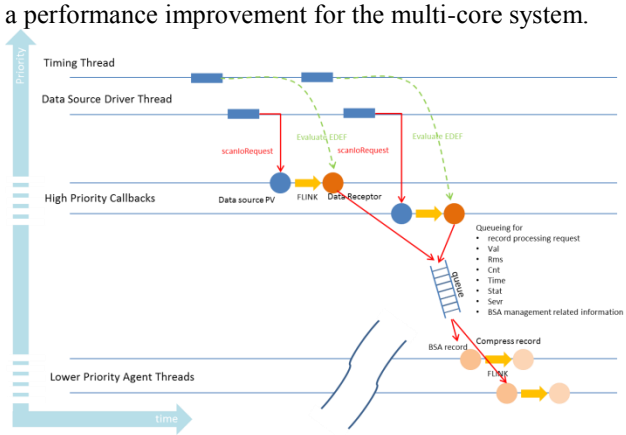

Figure 6: Consideration for parallel callbacks for multi-core system

PROTOTYPING AND TEST RESTULTS

We made a quick prototype for the BSA improvement and tested its performance.

Test Results for a Single Core System

We tested the prototype with a single core system (MVME6100/RTEMS/epics-3.15.1). For a realistic test, we forced 100% CPU load with an infinite loop in the lowest priority thread which has no effect on BSA processing and any other mission critical tasks. The system could handle up to 64 BSA PVs at 120Hz rates for 8 active EDEFs. The BSA capacity almost doubled. The system only allows 32 BSA PVs with the same condition. We also found that the time-critical part spends around 5 micro-seconds, and non-time-critical part spends slightly longer than 10 micro-seconds. Thus, the queueing mechanism and lower priority agent thread improves the BSA capacity for a single core system.

Test Results for a Multi-Core System

We also tested the prototype with a multi-core system (x86, 32 cores/linuxRT/epics-3.15.1). We also forced 100% CPU load in the lowest priority infinite loop thread. We tried parallel callbacks for upstream processing and multiple lower priority agent threads for downstream processing.

We assumed a dramatic performance improvements but it only doubled compares to the original code. The number of callback threads and the number of agent threads did not affect the result. We varied these numbers from16 to 32.

Finally, we discovered a global locking and lockset issues in EPICS record processing, and it affects the performance of record processing for parallel callbacks. We asked the EPICS core development team to fix this issue and they provided a snapshot version for testing. We contributed to the debugging of the snapshot version and continue our testing with it.

CONCLUSION

We have developed a quick prototype for BSA improvement. We separated out time-critical and non-time critical parts and queued the non-time-critical processing into a lower priority thread. We verified the prototype shows improvement for a single core system. We also tested the prototype for a multi-core system with parallel callbacks and multiple agent threads. We expected a dramatic performance improvement for the multi-core system but it only shows a similar improvement as the single core system due to a global locking issue in the EPICS record processing. We need more multi-core testing with an improved version of EPICS.

We will implement an API for production which can be called by the data source driver to avoid EPICS record processing overhead. It will improve performance.

REFERENCES

[1] J. Arthur, *et.al.*, "Linac Coherent Light Source (LCLS) conceptual design report," SLAC-R593, SLAC (2002)

[2] P. Krejcik, *et.al.*, "Timing and Synchronization at the LCLS," DIPAC 2007, Venice, Italy, May 2007, SLAC-PUB-12593

[3] J. Dusatko, et.al., "The LCLS Timing Event System," BIW 2010, Santa Fe, New Mexico, USA, May 2010

[4] "Experimental Physics and Industrial Control System," http://www.aps.anl.gov/epics

[5] M. Zelazny, *et.al.,* "Orbit Display's use of the Physics Application Framework for LCLS," SLAC-PUB-13788 SLAC (2009)

[6] "Real-Time Preemption Patch-Set," http://elinux.org/images/4/4e/Real-Time-Preemption-Patchset.pdf

[7] "RTEMS Real Time Operating System (RTOS)," https://www.rtems.org