

Smaller and Faster: Data Compression in areaDetector

Slides: Mark Rivers

GeoSoilEnviroCARS, Advanced Photon Source

University of Chicago

EPICS meeting, ITER, June 2019

Presenter: Ulrik Kofoed Pedersen, Head of Beamline Controls

Demo: Gary Yendell

Diamond Light Source

Outline

- ADCore releases since 2018: R3-4, R3-5, R3-6
- This talk:
 - New data compression features
 - Demo of data compression
 - Other major features from these releases

Data Compression Motivation

- We are already in the era of “big data” with existing detectors. Eiger, Pilatus, Lambda, PCO, FLIR/Point Grey, Xspress 3, etc.
 - Can all produce data faster than most disk systems can handle
 - All exceed 1 Gbit network capacity, and some exceed 10 Gbit.
 - Rapidly fill up disks
- Will become a more serious issue with synchrotron upgrades
 - Increased count rates will allow existing detectors to run at their maximum speed
 - New generations of even faster detectors will be coming
- Data compression can help with these issues
 - Must be fast and easy to use

Support for Compressed NArrays and NTNDArrays

- NArray has 2 new fields to support compression
- .codec field (struct Codec_t) to describe the compressor

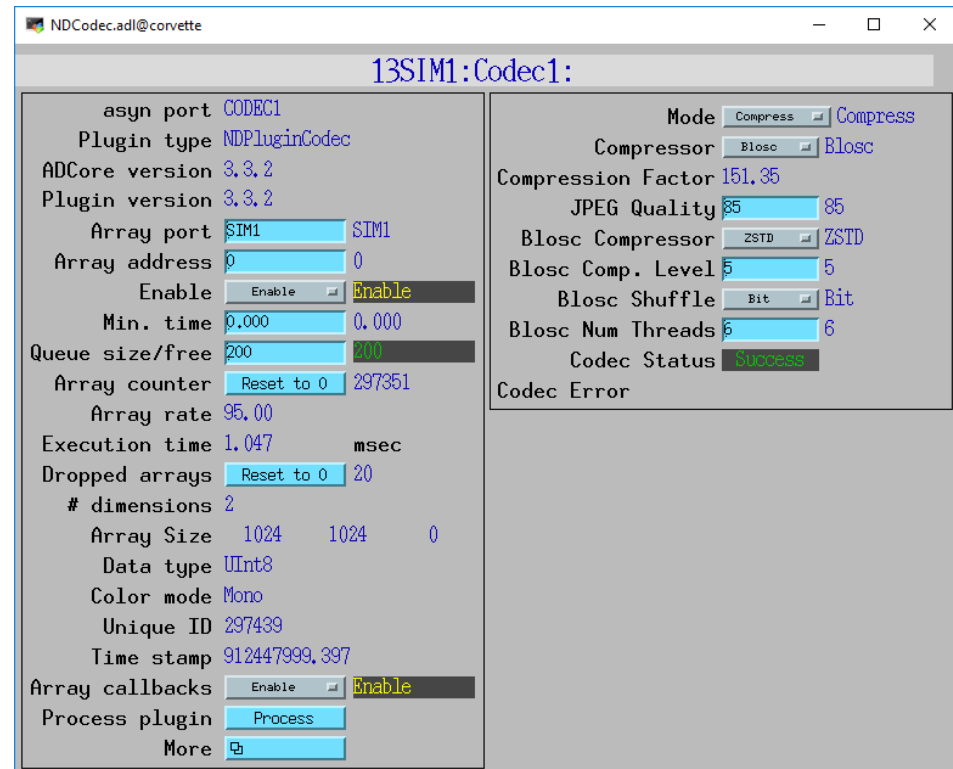
```
typedef enum {
    NDCODEC_NONE,
    NDCODEC_JPEG,
    NDCODEC_BLOSC,
    NDCODEC_LZ4,
    NDCODEC_BSLZ4
} NDCodecCompressor_t;
```

```
typedef struct Codec_t {
    std::string name;          /**< Name of the codec */
    int         level;         /**< Compression level. */
    int         shuffle;       /**< Shuffle type. */
    int         compressor;    /**< Compressor type */
}
```

- .compressedSize (size_t) field with compressed size if codec.name is not empty.
- pvAccess NTNDArray has always had .compressedSize and .codec fields, but never previously implemented in servers or clients

NDPluginCodec (R3-4)

- New plugin for data compression and decompression
- Written by Bruno Martins from FRIB
- Mode:
 - Compress or Decompress
- Compressor:
 - None
 - JPEG (JPEGQuality selection)
 - Blosc (many options, next slide)
 - LZ4
 - BSLZ4 (Bitshuffle/lz4)
- CompFactor_RBV:
 - Actual compression ratio
- CodecStatus, CodecError
- JPEG is lossy, all others lossless



Blosc Codec Options

- BloscCompressor options. Each has different compression performance and speed
 - BloscLZ
 - LZ4
 - LZ4HC
 - Snappy
 - Zlib
 - Zstd
- BloscCLevel
 - Compression level: 0=no compression, 9=maximum compression.
 - Increasing execution time with increasing level.
- BloscShuffle
 - Choices = None, Byte, Bit.
 - Differences in speed and compression performance.
- BloscNumThreads
 - Number of threads used to compress each NDArray

LZ4 and BSLZ4 Codecs

- These are the codecs used by the Eiger server from Dectris
 - They don't use the Blosc codecs, but rather the native LZ4 and Bitshuffle/LZ4 codecs.
- Dectris server can optionally use these compressions for HDF5 files saved locally on their server
- Dectris server always uses one of these compressions for data streamed over the ZeroMQ socket interface to the ADEiger driver
- These can now be decoded directly in ADEiger, or passed as compressed NDArrays to NDPluginCodec and other plugins
- Compressed arrays can be passed directly to NDFileHDF5 to be written with newly supported *direct chunk write* feature.
More on this later.

Codec Parameter Records (R3-5)

- Codec_RBV and CompressedSize_RBV records to asynNDArrayDriver and hence to all plugins.

The screenshot shows a window titled "NDPluginBaseFull.adl@corvette" with a sub-window "13SIM1:HDF1:". The window is divided into two main panels. The left panel contains various configuration parameters for the plugin, and the right panel shows execution statistics and codec details.

Left Panel Parameters:

- asyn port: FileHDF1
- Plugin type: NDFileHDF5 ver1.10.1
- ADCore version: 3.5.0
- Plugin version: 3.5.0
- Array port: CODEC1
- Array address: 0
- Enable: Enable
- Min. time: 0.000
- Max. array rate: 0.000
- Max. byte rate: 0.000e+00
- Callbacks block: No
- # threads: 1
- Max # threads: 1
- Queue size/free: 2000 / 2000
- Sort mode: Unsorted
- Sort time: 0.100
- Sort size/free: 20 / 0
- # disordered: Reset to 0
- Array callbacks: Disable
- Process plugin:
- asyn record:

Right Panel Execution Statistics:

- Array counter: 0
- Array rate: 0.00
- Execution time: 0.018 msec
- Dropped arrays: 0
- Dropped outputs: 0
- # dimensions: 2
- Array size: 1024 1024 0
- Codec: blosc
- Compressed size: 10148
- Data type: UInt8
- Color mode: Mono
- Bayer pattern: RGG
- Unique ID: 220421
- Time stamp: 924292037.847

Attributes Panel:

- File:
- Macros:
- Status: File not found

HDF5 Changes (R3-5 & R3-6)

- NDFileHDF5 file writing plugin has always supported the “built-in” compression filters from HDF5:
 - N-bit
 - SZIP
 - ZLIB
- HDF5 Dynamically Loadable Filters ([ref](#))
 - R3-3 added support for Blosc filters
 - Thanks Xiaoqiang Wang, PSI
 - New support for LZ4 and Bitshuffle/LZ4 filters
 - All of these compressors are called from the HDF5 library.
 - Limits performance because of the overhead of the library.
- New support for HDF5 “Direct Chunk Write” ([ref](#))(Use **R3-6**)
 - The NDArrays can be pre-compressed, either in NDPluginCodec, or directly by the driver (e.g. ADEiger)
 - Much faster, much of the code in the HDF5 library is skipped.
 - Thanks Gary Yendell, Diamond Light Source
- Fixed a number of memory leaks, some were significant
- Added FlushNow record to force flushing datasets to disk in SWMR mode

HDF5 Direct Chunk Write Performance

- 1024x1024 32-bit images
- simDetector generating ~ 1350 frames/s = 5.4 GB/s.
- Blosc LZ4 ByteShuffle compression
 - Compression level 6.
 - NDPluginCodec
 - 6 Blosc threads
 - 3 plugin threads.
 - Compression factor is ~ 64
- Time to save a single HDF5 file with 10,000 frames.

	No compression	NDFileHDF5 compression	NDPluginCodec compression, direct chunk write
File size (MB)	40,000	650	650
Total time (s)	106	32	7.4
Frame/s	94	312	1,351
MB/s uncompressed	389	1,250	5,405
MB/s compressed	N.A.	20	88

- HDF5 library can only compress 312 frames/s
- NDPluginCodec & direct chunk write keeps up with simDetector 1,350 frames/s

HDF5 Decompression Plugin Filters (ADSupport R1-7)

- HDF5 supports dynamic loading of compression and decompression filter libraries at run time.
- The Blosc, LZ4 and BSLZ4 have been built into the HDF5 library in ADSupport so that dynamic loading is *not* required when using NDFileHDF5.
- However, to decompress HDF5 files compressed with Blosc, LZ4 or BSLZ4 with other applications dynamic loading of the filters will be required
- ADSupport now builds these dynamic filter libraries for Linux, Windows, and Mac.
- Must set the following environment variable to use them:

```
HDF5_PLUGIN_PATH=[areaDetector]/ADSupport/lib/linux-x86_64
```

HDF5 Decompression Plugin Filters

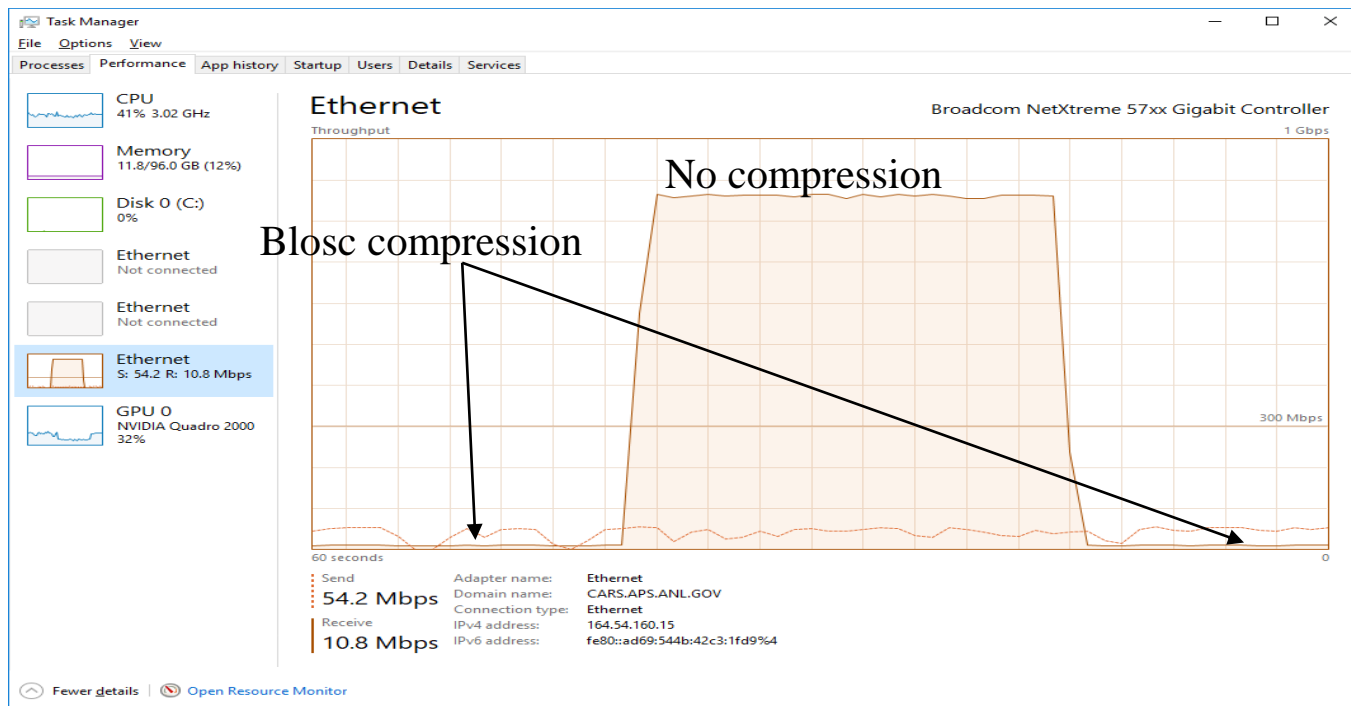
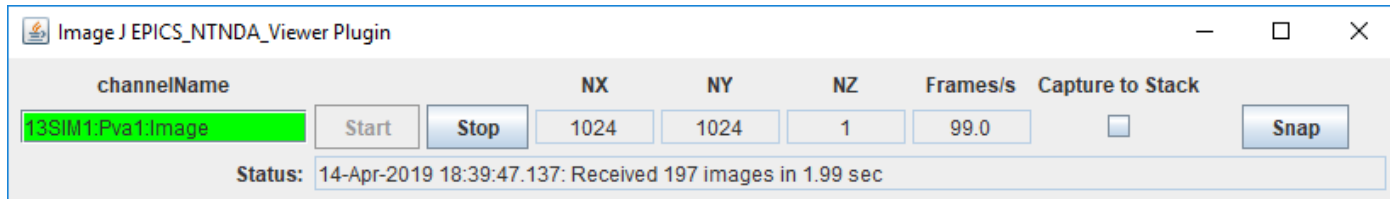
```
>h5dump --properties test_hdf5_direct_chunk_3.h5
HDF5 "test_hdf5_direct_chunk_3.h5" {
GROUP "/" {
    GROUP "entry" {
...
        DATASET "data" {
            DATATYPE  H5T_STD_U32LE
            DATASPACE  SIMPLE { ( 100, 1024, 1024 ) / ( 100, 1024, 1024 ) }
            STORAGE_LAYOUT {
                CHUNKED ( 1, 1024, 1024 )
                SIZE 4082368 (102.742:1 COMPRESSION)
            }
            FILTERS {
                USER_DEFINED_FILTER {
                    FILTER_ID 32001
                    COMMENT blosc
                    PARAMS { 2 2 4 4194304 8 1 1 }
                }
            }
        }
...
        DATA {
            (0,0,0): 173140, 173141, 173142, 173143, 173144, 173145, 173146,
            (0,0,7): 173147, 173148, 173149, 173150, 173151, 173152, 173153,
            (0,0,14): 173154, 173155, 173156, 173157, 173158, 173159, 173160,
            (0,0,21): 173161, 173162, 173163, 173164, 173165, 173166, 173167,
```

Demo

Compression

ImageJ pvAccess Viewer

- Now supports displaying compressed NTNDArrays
- Supports all compressions (JPEG, Blosc, LZ4, BSLZ4)
- Can greatly reduce network bandwidth when the IOC and viewer are running on different machines



ADEiger Changes

- Now supports Bitshuffle/LZ4 on Stream interface over ZeroMQ
 - Previously only LZ4 was supported
- New StreamDecompress bo record to enable/disable decompression. If disabled:
 - NDFileHDF5 can use Direct Chunk Write without ever decompressing
 - NDPluginPva can send to ImageJ without ever decompressing
 - NDPluginCodec can decompress for other plugins like NDPluginStats, etc.
- Eiger Simplon API version 1.6 -> 1.8 changes WIP
 - <https://github.com/areaDetector/ADEiger/pull/27>

ADEiger Changes

eigerDetector.adl@corvette

Eiger Detector Control 13EIG1:cam1:

Setup

asyn port **EIG**
 EPICS name **13EIG1:cam1:**
 Manufacturer **Dectris**
 Model **Eiger 500K**
 Serial number **E-01-0147**
 Firmware version **1.6.6**
 SDK version **Unknown**
 Driver version **2-5**
 ADCore version **3.4.0**

Connected

Connection
 Debugging

Acquisition

Threshold (eV)	<input type="text" value="4000.000"/>	4000.000
Photon energy (eV)	<input type="text" value="6000.000"/>	6000.000
Exposure time (s)	<input type="text" value="1.000e-02"/>	9.992e-03
Acquire period (s)	<input type="text" value="1.000e-02"/>	1.000e-02
# Images	<input type="text" value="1000"/>	1000
# Triggers	<input type="text" value="1"/>	1
# Images Complete		613

Trigger mode Internal Se:
 Trigger Exp. (INTE) 0.000000
 Manual Trigger No

Acquire Status **Collecting**
 # Queued arrays **0**
 Wait for plugins No
 Acquire busy **Acquiring**
 Acquire Message **Triggering**
 Detector State **Acquire**
 Detector Armed **Armed**
 Readout Time **0.000008**
 Rate Cutoff **18469**
 Image counter 1156
 Image rate **100.00**

ROI mode Disabled
 Flatfield Corr. Enabled
 FW Compres. Disabled
 Compress. Alg. LZ4
 Array Callbacks Enable
 Data Source Stream

File Writer

Enable No
 State **ready**
 Images/File 500
 File Name Pattern series_\$id
 Current seq. id **113**
 Save Files Local Yes
 Local Path /home/epics/scratch/eiger/
 Local Path Exists **Yes**
 Create dir. depth -1
 DCU RAM Disk Free **197.631** GB
 Auto Remove No
 Delete All **Delete All!!!**

Detector Metadata

Beam Center X	<input type="text" value="0.000"/>	0.000	pix.
Beam Center Y	<input type="text" value="0.000"/>	0.000	pix.
Wavelength	<input type="text" value="2.0664"/>	2.0664	Ang.
Det. distance	<input type="text" value="0.000"/>	0.000	mm
Angle Start			
Chi	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	deg
Kappa	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	deg
Omega	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	deg
Phi	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	deg
Two Theta	<input type="text" value="0.000"/>	<input type="text" value="0.000"/>	deg

Plugins

Stats

Detector Status

Detector State **idle**
 Error Parameters
 Temperature C **26.3**
 Humidity % **1.8**
 Links
 DCU Buff. Free % **100.0**
 Read Status Rate

Stream

Enable/Disable Yes
 State **ready**
 Dropped Frames **0**
 Decompress No

Detector Info

Description **Dectris Eiger 500K**
 Detector Size **1030 514**
 Pixel Size **0.000075 0.000075**
 Sensor Material **Si**
 Sensor Thickness **4.500000e-04**

Monitor

Enable/Disable No
 State **normal**
 Timeout (mS) 500.0

Buffers

Buffers used **11**
 Buffers alloc/free **29 18**
 Memory max/used (MB) **0.0 26.0**
 Buffer & memory polling
 Empty free list

Attributes

File
 Macros
 Status **File not found**

Shutter

Shutter mode None
 Status: Det. **Closed** EPICS **Closed**
 Open/Close
 Delay: Open Close
 EPICS shutter setup

Improvements in ADCore (R3-4)

- New MaxByteRate record for plugins to limit “output rate”
 - For most plugins this limits the byte rate of the NDArrays passed to downstream plugins
 - For NDPluginStdArrays it limits the byte rate of the data callbacks to waveform records, and hence to Channel Access clients
 - For NDPluginPva it limits the byte rate of the data callbacks to pvAccess clients
- Optimization improvement when output arrays are sorted.
 - Previously it always put the array in the sort queue, even if the order of this array was OK.
 - Introduced an unneeded latency because the sort task only runs periodically.
 - Caused ImageJ update rates to be slow, because it made PVA output comes in bursts, and some arrays were dropped either in the pvAccess server or client.
 - Now if the array is in the correct order it is output immediately.

Documentation Improvements (R3-5)

- Documentation was changed from manually edited HTML pages to reStructuredText (.rst) files processed with Spinx.
 - Most tables are left in native HTML because .rst conversion is poor quality
- Server changed from <https://cars.uchicago.edu/software/epics/> to areaDetector.github.io/
- Advantages:
 - Easier to edit
 - Nicer looking pages
- New Travis CI job at top-level areaDetector
 - Runs doxygen and sphinx to update the areaDetector.github.io files every time there is a push to the top-level areaDetector repository.
- Thanks to Stuart Wilkins from BNL who set up the process and converted all of the files in ADCore, ADProsilica, and ADFastCCD.
- Other detector repositories still need to be converted.
 - Use pandoc to convert .html to .rst. Manual editing still required.

areaDetector Plugin NDPluginProcess

April 16, 2018

Mark Rivers

University of Chicago

Contents

- [Overview](#)
- [Configuration](#)
- [Screen shots](#)

Overview

NDPluginProcess performs arithmetic processing on NDArray data. It performs the following operations in the order listed. Each of these operations can be individually enabled and disabled.

1. Subtracts a background array which has been previously acquired.
2. Divides by a flat field array which has been previously acquired, and then multiplies by a flat field scale factor.
3. Multiplies by a scale factor and adds an offset.
4. Clips to a maximum specified value.
5. Clips to a minimum specified value.
6. Applies a recursive digital filter.
7. Converts to the specified output data type.
8. Exports the processed data as a new NDArray object.

If any of the above operations is enabled, then the array is first converted to NDFloat64 data type, i.e. double-precision float. The operations are all performed in double-precision, and then the array is converted to the specified output data type.

NDPluginProcess is both a **recipient** of callbacks and a **source** of NDArray callbacks. This means that other plugins, such as the NDPluginStdArrays, NDPluginStats, and NDPluginFile plugins can be connected to an NDPluginProcess plugin, in which case they will use the processed data.

NDPluginProcess is fully N-dimensional. It can be used for 2-D images, 3-D (color) images, or any type of N-dimensional data.

NDPluginProcess inherits from NDPluginDriver. The [NDPluginProcess class documentation](#) describes this class in detail.

NDPluginProcess.h defines the following parameters. It also implements all of the standard plugin parameters from [NDPluginDriver](#). The EPICS database NDProcess.template provide access to these parameters, listed in the following table. Note that to reduce the width of this table the parameter index variable names have been split into 2 lines, but these are just a single name, for example NDPluginProcessSaveBackground.

Parameter Definitions in NDPluginProcess.h and EPICS Record Definitions in NDProcess.template						
Parameter index variable	asyn interface	Access	Description	drvInfo string	EPICS record name	EPICS record type
Background subtraction						
NDPluginProcess SaveBackground	asynInt32	r/w	Command to use the most recently acquired array as a background. Note that this recently acquired array should have been acquired with EnableBackground=0, or else that array will already have had the background subtracted, which is probably not what was intended!	SAVE_BACKGROUND	\$(P)\$(R)SaveBackground \$(P)\$(R)SaveBackground_RBV	bo bi

Show all X

areaDetector 3-5 Site Page « NDPluginOverlay NDPluginPva » Source Search

NDPluginProcess

author: Mark Rivers, University of Chicago

Contents

- NDPluginProcess
 - Overview
 - Recursive filter implementation
 - Predefined filters
 - Recursive Average
 - Average
 - Sum
 - Difference
 - Recursive Average Difference
 - Copy to Filter
 - Configuration
 - Screen shots

Overview

NDPluginProcess performs arithmetic processing on NDArray data. It performs the following operations in the order listed. Each of these operations can be individually enabled and disabled.

- Subtracts a background array which has been previously acquired.
- Divides by a flat field array which has been previously acquired, and then multiplies by a flat field scale factor.
- Multiplies by a scale factor and adds an offset.
- Clips to a maximum specified value.
- Clips to a minimum specified value.
- Applies a recursive digital filter.
- Converts to the specified output data type.
- Exports the processed data as a new NDArray object.

If any of the above operations is enabled, then the array is first converted to NDFloat64 data type, i.e. double-precision float. The operations are all performed in double-precision, and then the array is converted to the specified output data type.

NDPluginProcess is both a recipient of callbacks and a source of NDArray callbacks. This means that other plugins, such as the NDPluginStdArrays, NDPluginStats, and NDPluginFile plugins can be connected to an NDPluginProcess plugin, in which case they will use the processed data.

NDPluginProcess is fully N-dimensional. It can be used for 2-D images, 3-D (color) images, or any type of N-dimensional data.

NDPluginProcess inherits from NDPluginDriver. The [NDPluginProcess class documentation](#) describes this class in detail.

NDPluginProcess.h defines the following parameters. It also implements all of the standard plugin parameters from NDPluginDriver. The EPICS database NDProcess.template provide access to these parameters, listed in the following table. Note that to reduce the width of this table the parameter index variable names have been split into 2 lines, but these are just a single name, for example `NDPluginProcess.SaveBackground`.

Parameter Definitions in NDPluginProcess.h and EPICS Record Definitions in NDProcess.template						
Parameter index variable	asyn interface	Access	Description	drvInfo string	EPICS record name	EPICS record type
Background subtraction						
NDPluginProcess SaveBackground	asynInt32	r/w	Command to use the most recently acquired array as a background. Note that this recently acquired array should have been acquired with EnableBackground=0, or else that array will already have had the background subtracted, which is	SAVE_BACKGROUND	\$(P)\$(R)SaveBackground \$(P)\$(R)SaveBackground_RBV	bo bi

FourthDraft (1).docx All Instructions f...docx ALL FET to USBan...pdf Show all

NDPluginAttribute Time Series (R3-5)

- Previously NDPluginAttribute time series code was internal
- Changed so that it now uses NDPluginTimeSeries, same change that was made to NDPluginStats in R3-5
- Fewer lines of code, and adds Circular Buffer mode
- The time-series waveform PVs are the same
- The PVs to control the time-series (start/stop, # of points) have changed, so clients may need modifications

NDPluginAttribute Time Series (R3-5)

NDPluginAttribute.adl@corvette

13SIM1:Attr1:

asyn port ATTR1

Plugin type NDPluginAttribute

ADCore version 3.5.0

Plugin version 3.5.0

Array port SIM1

Array address 0

Enable Enable

Min. time 0.000 0.000

Queue size/free 2000

Array counter 20767

Array rate 99.00

Execution time 0.034 msec

Dropped arrays 0

dimensions 2

Array Size 1024 1024 0

Data type UInt8

Color mode Mono

Unique ID 5261

Time stamp 924289857.423

Array callbacks Enable

Process plugin Process

More

Reset

Attributes Combined attributes

Time Series

Acquire

Number of points 1024

Current point 1024

Read rate 1 second

AcquireMode Fixed length

NDPluginAttribute8.adl@corvette

13SIM1:Attr1:

Attribute	Value	Value sum	Plot	
1 NDArrayUniqueId	NDArrayUniqueId	12096.0000	30092254859.000	<input type="button" value="Plot"/>
2 NDArrayTimeStamp	NDArrayTimeStamp	924289926.688	24653581528521.320	<input type="button" value="Plot"/>
3 NDArrayEpicsTSSec	NDArrayEpicsTSSec	924289926.000	21616365669907.000	<input type="button" value="Plot"/>
4 NDArrayEpicsTSnSec	NDArrayEpicsTSnSec	697873797.000	10495530955428.000	<input type="button" value="Plot"/>
5 ImageCounter	ImageCounter	12096.0000	15677658936.000	<input type="button" value="Plot"/>
6 ID_Energy	ID_Energy	23.7792	447049.9650	<input type="button" value="Plot"/>
7 RingCurrent	RingCurrent	102.0542	1851778.9635	<input type="button" value="Plot"/>
8 AcquireTime	AcquireTime	0.0100	176.4400	<input type="button" value="Plot"/>

Time Series

Acquire

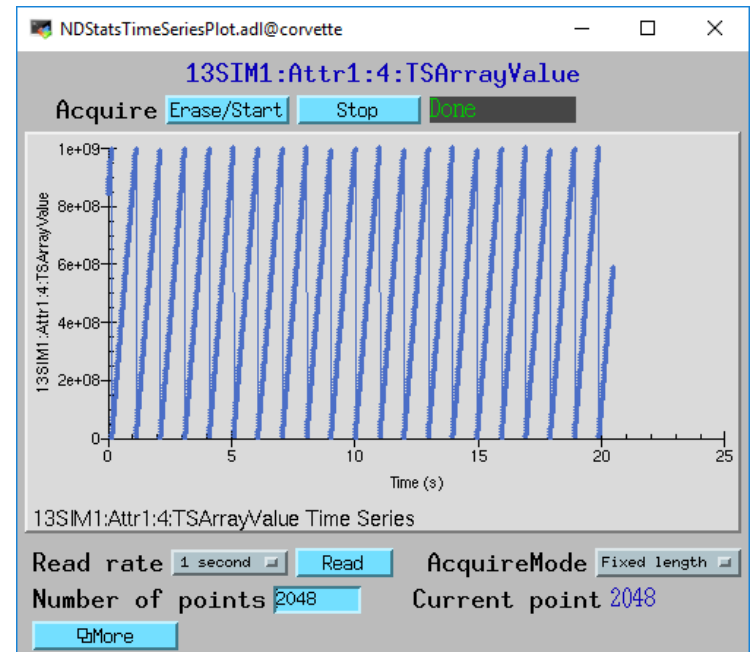
Number of points 1024

Current point 1024

Read rate 1 second

AcquireMode Fixed length

Reset



Roadmap: ADCore R5-0?

- Use NTNDArrays inside drivers and plugins
- Use pvDatabase
 - “local” provider within IOC
 - “pva” provider between IOCs
- Smart pointers automatically eliminate all unnecessary copying
- Eliminates need for NDPluginPva
- V4 clients can immediately receive data from any point in plugin chain
- Distribute load to multiple IOCs without pvaDriver
- Bruno Martins has demonstrated this working for ADSimDetector and NDPluginStats