areaDetector Update:

EPICS V4 and More

Mark Rivers

GeoSoilEnviroCARS, Advanced Photon Source

University of Chicago

Outline

- Last TWG areaDetector talk was April 2016
  - Releases since then: R2-5, R2-6 (very soon)
- EPICS V4 driver and plugin
- Other new and improved drivers
- Other new and improved plugins
- ImageJ plugins
- Future plans

EPICS Version 4 in a Nutshell

- New Protocol, “pvAccess”
- Structured data
- Introspection interface, “pvData”
- Dynamic typing
- Streaming
- Standard Scientific Types
- RPC and put/get
- New smart database
- Codec based transport
- All APIs in C++ and Java
- Python and Matlab
- High Performance
- High Reliability

```
$ eget -s XCOR:LI24:900:TWISS
non-normative type
structure
    double energy 5.00512
    double psix 37.7625
    double alphax 13.6562
    double betax -2.78671
    double etax -0.00698294
    double etaxp 0.00107115
    double psiy 31.9488
    double alphay 116.762
    double betay 5.2592
    double etay 0
    double etayp 0
```

Figure: Example EPICS v4 pvAccess method “eget”, getting PV whose value is a structure giving the Courant-Snyder parameters at a corrector magnet. This example is from the LCLS control system at SLAC.

The EPICS V4 “Normative Types”

The Normative Types Spec defines a standard for commonly used data types,
http://epics-pvdata.sourceforge.net/alpha/normativeTypes/normativeTypes.html

5. General Normative Types
   1. NTScalar
   2. NTScalarArray
   3. NTEnum
   4. NTMatrix
   5. NTURI
   6. NTNameValue
   7. NTTable
   8. NTAttribute
6. Specific Normative Types
   1. NTMultiChannel
   2. NTNDArray
   3. NTContinuum
   4. NTHistogram
   5. NTAggregate

```
$ eget -s XCOR:LI24:900:RMAT
     0.0727485     0.0289316             0             0     0.0652488     0.00125391
     0.0578214     0.0391775             0             0      -0.027185   -0.000192344
             0             0    0.00943029       1.14291             0              0
             0             0    -0.0013367    -0.0348832             0              0
  -0.000370971  -0.000283933             0             0    -0.0182387   -0.000198345
       0.10031      0.018722             0             0      -10.5721      -0.179568
```

```
$ eget pva://mccas0.slac.stanford.edu:39633/QUAD:LTU1:880:RMAT?type=design
```

```
$ eget -s LCLS:ELEMENTS
ELEMENT      ELEMENT_TYPE      EPICS_DEVICE_NAME     S_DISPLAY           OBSTRUCTION
CATHODE               MAD          CATH:IN20:111        2014.7                    N
 SOL1BK               MAD          SOLN:IN20:111        2014.7                    N
   CQ01               MAD          QUAD:IN20:121        2014.9                    N
   SOL1               MAD          SOLN:IN20:121        2014.9                    N
   XC00               MAD          XCOR:IN20:121        2014.9                    N
…
```

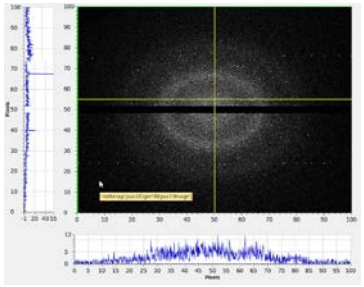


Figure: An extract of the Table of Contents of the Normative Types Specification document, together with examples of 4 selected types

What does an EPICS V4 PV for structured data look like?

The pvAccess PV name

EPICS V4 data type identifier “NTNDArray”

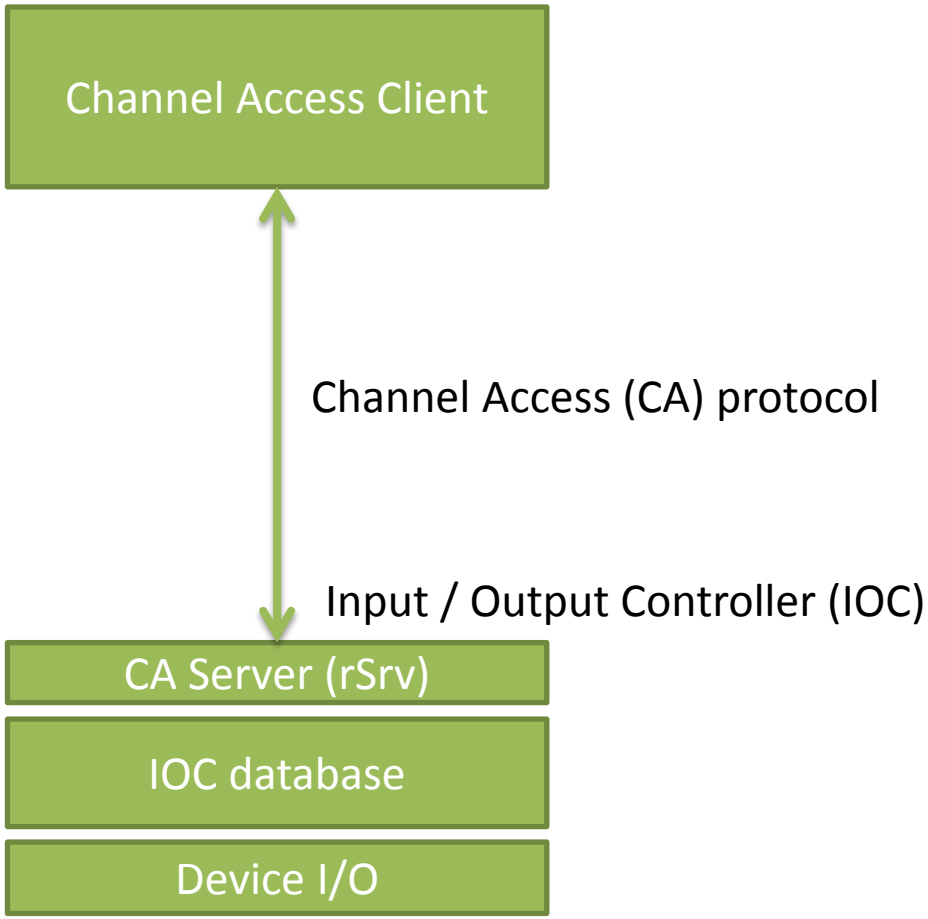
The image raw data

The image meta data; giving how to interpret the data in the *value* field, and other information.

*Figure: A screenshot of the output of the EPICS V4 “*pvget*” command, showing data of a PV which encapsulates all the data of an areaDetector NDArray in in Normative Type* NTNDArray. *Example from B. Martin’s AD work at BNL.*

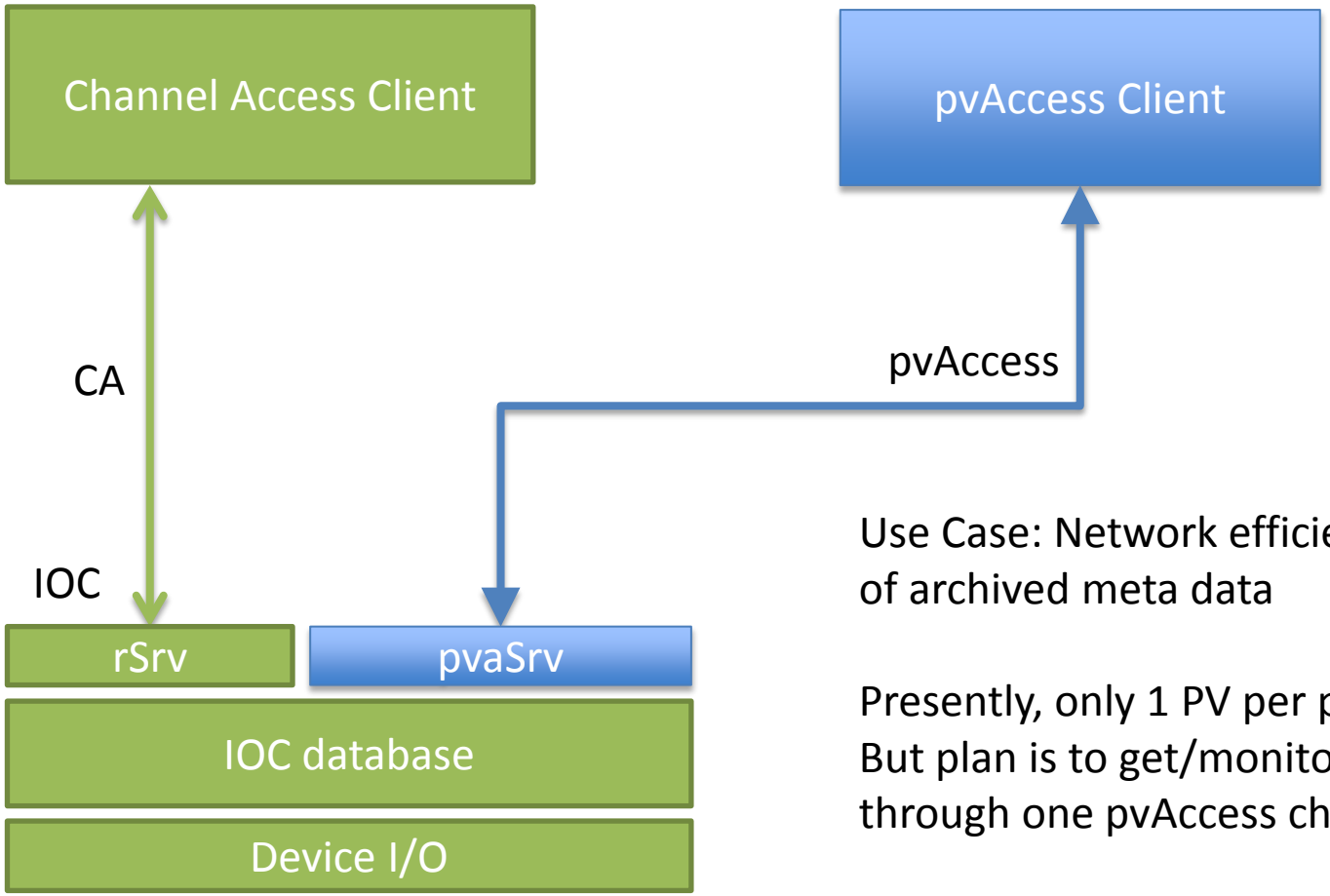
EPICS Version 3 basic block diagram

EPICS in the nominal usage: An EPICS client communicates over Channel Access (CA) protocol to an Input/Output Controller (IOC) Channel Access server (module rSrv in an IOC)



EPICS Version 4 is an extension of V3

V4 IOC == V3 IOC + pvAccess Server

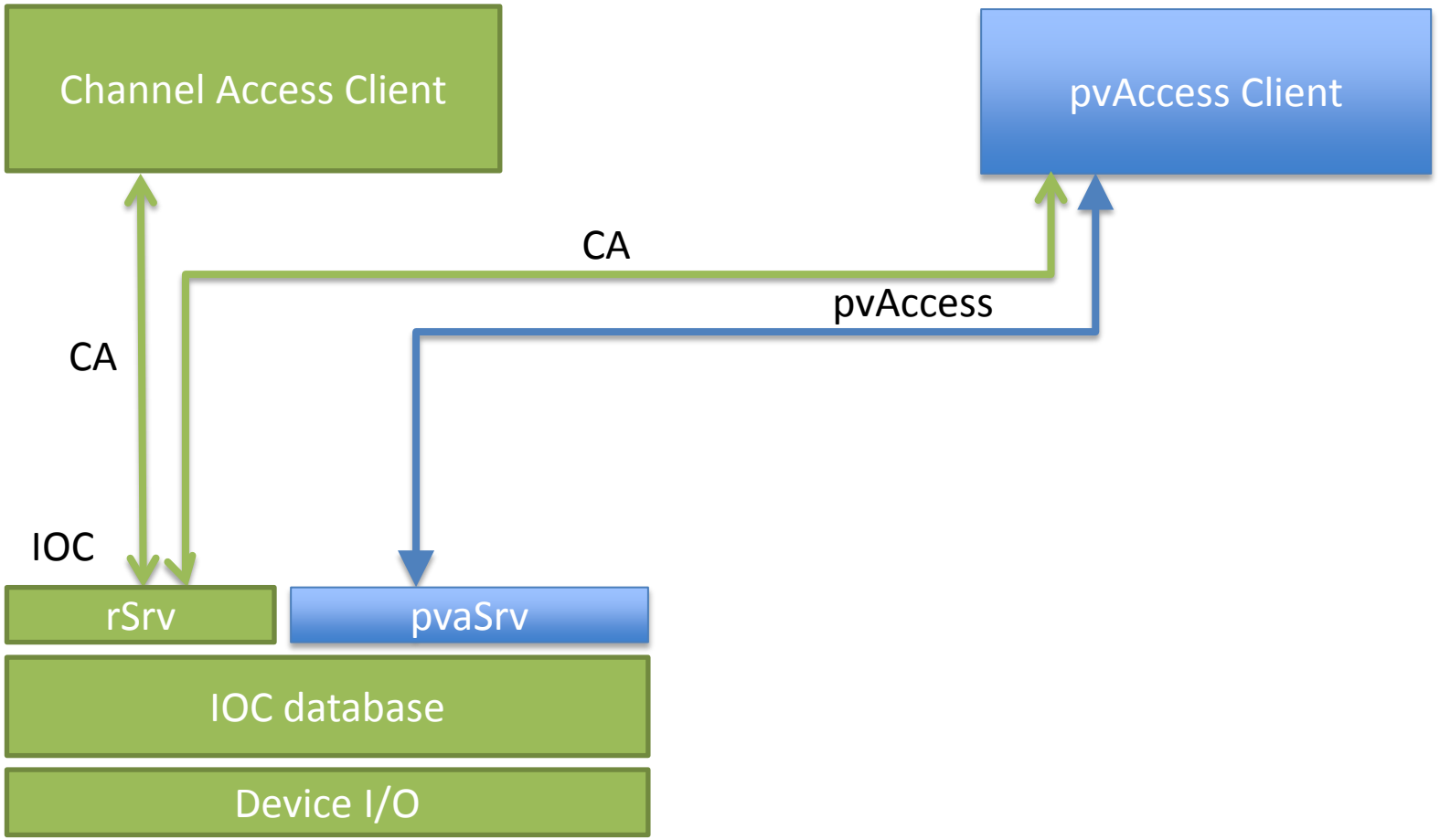


Use Case: Network efficient acquisition of archived meta data

Presently, only 1 PV per pvAccess channel. But plan is to get/monitor a group of PVs through one pvAccess channel.

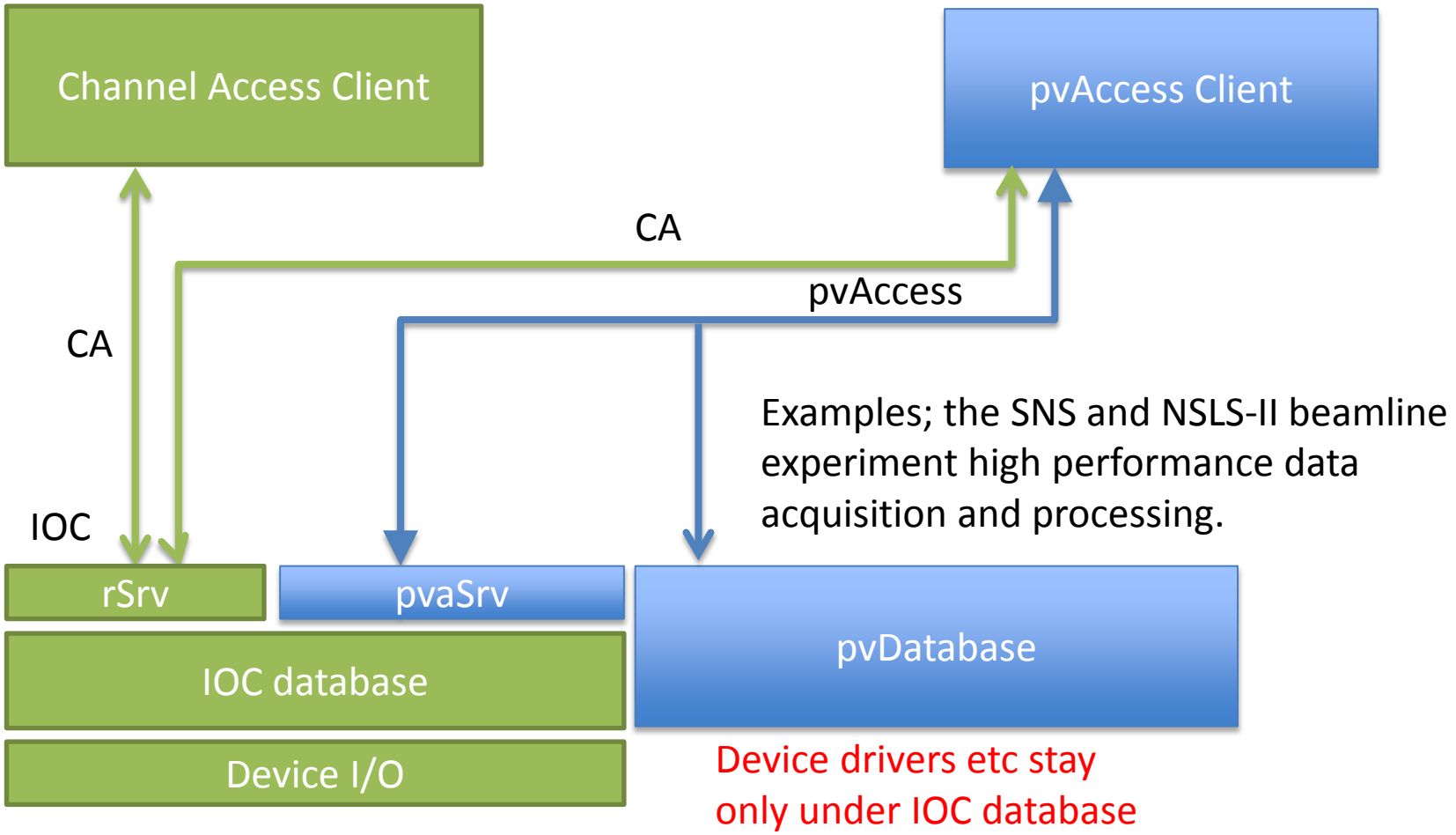
EPICS Version 4 includes CA

The pvAccess API includes Channel Access support, so one client lib does both



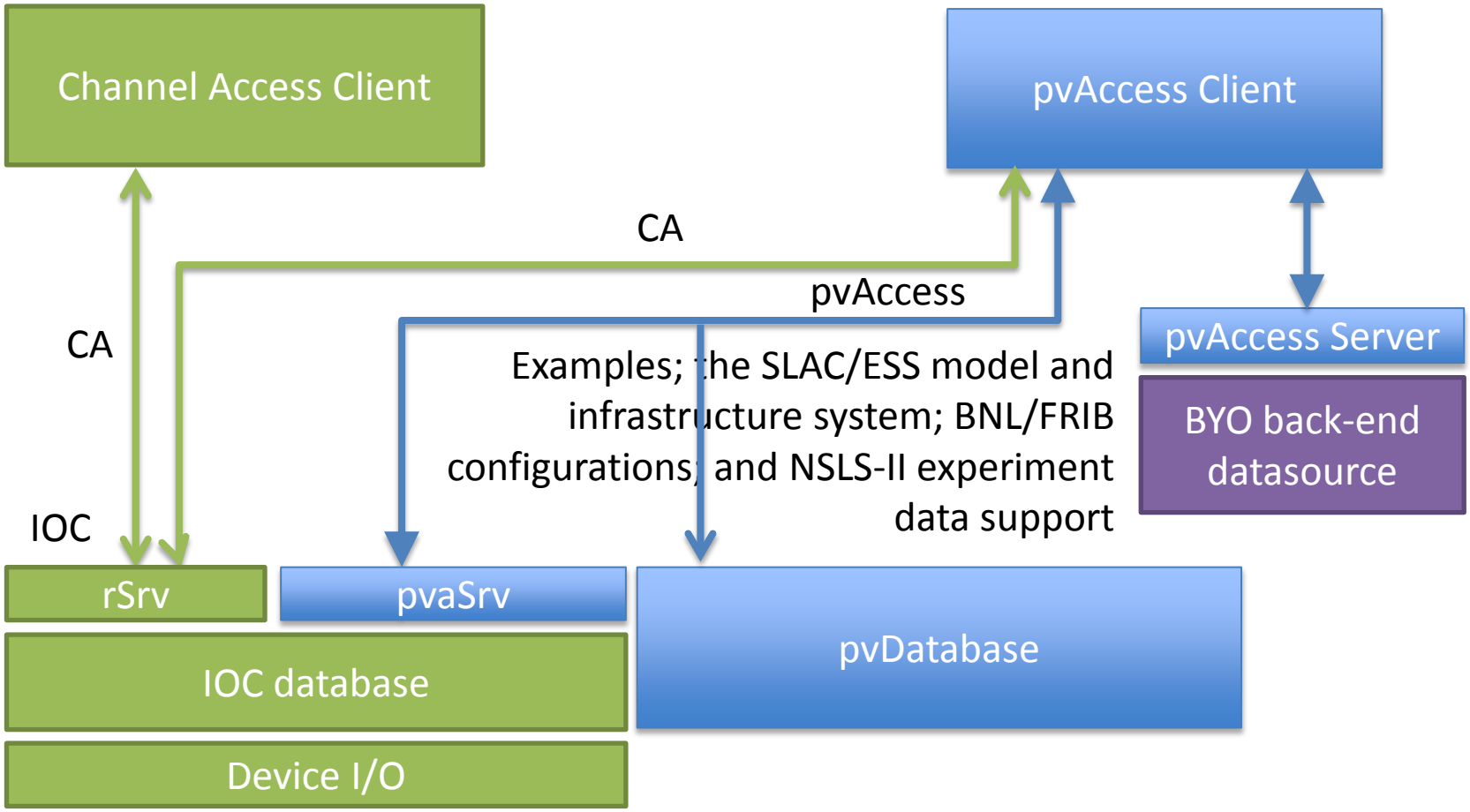
EPICS Version 4 new database

A new smart database, “pvDatabase” can be used for data assembly and processing



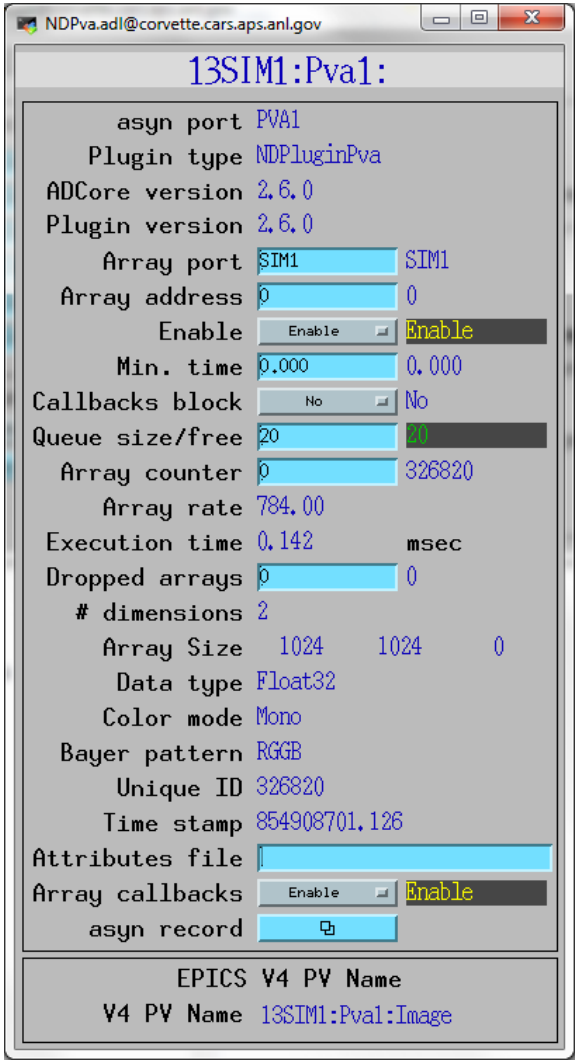
EPICS Version 4 middleware support

RPC and Service Oriented Architecture (SOA)



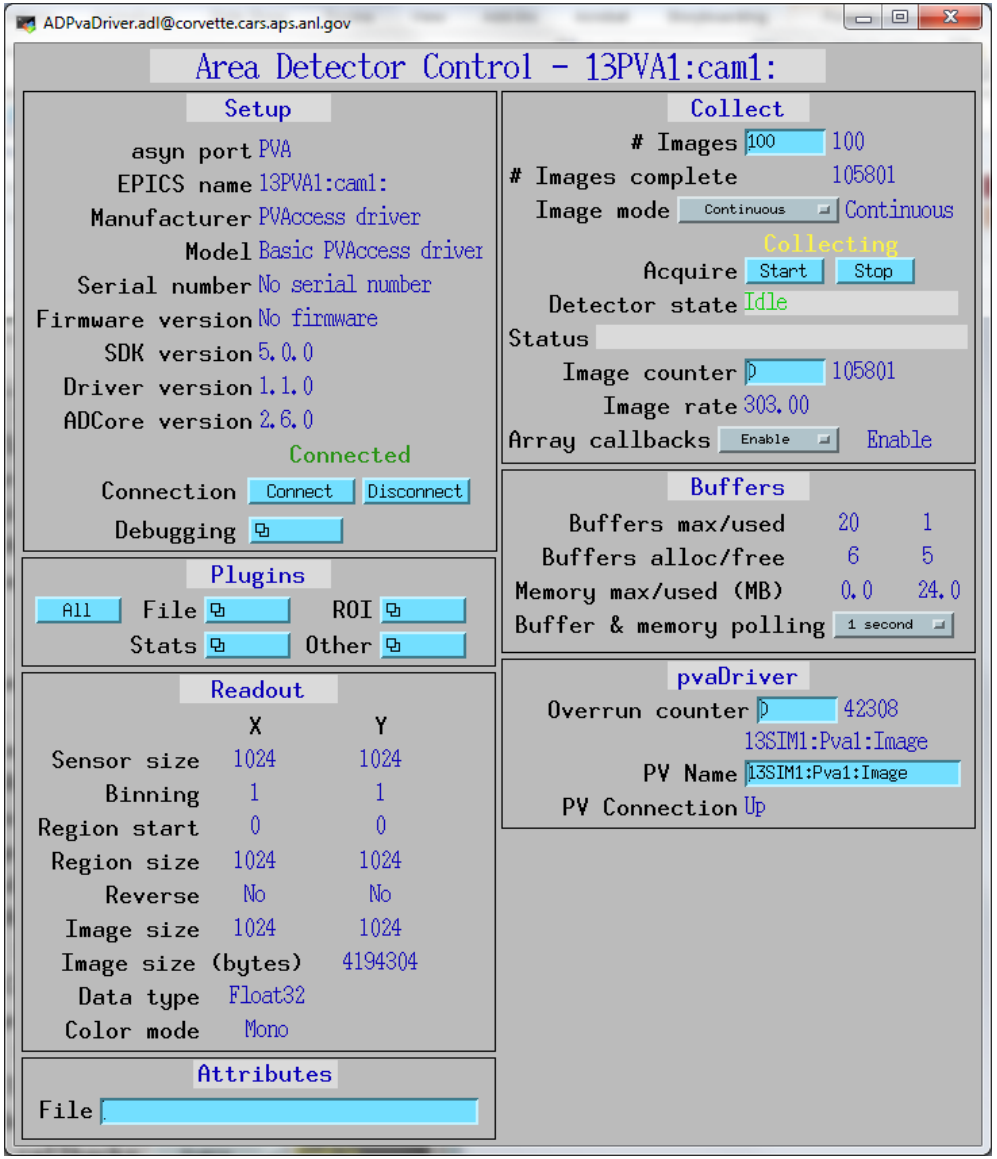
NDPluginPva

- New plugin that converts NDArrays into the EPICSv4 normative type NTNDArray
- An embedded EPICSv4 server serves the new NTNDArray structure as an EPICSv4 PV
- High performance, ~3.2GB/s shown here
- Can be received by any EPICSv4 client
  - Java, Python, C++ versions of pvAccess
  - CSS has a widget that can display NTNDArrays
  - caQtDM has a new camera widget to display NTNDArrays
  - Tim Madden is writing a version of the Java areaDetector ImageJ plugin based on pvAccess; will talk to this plugin
  - Can include an NTNDArray receiver in another IOC



pvAccess Driver

- New pvAccess driver receives NTNDArrays over the network, converts to NDArrays and calls plugins
- Can be used to run areaDetector IOC and plugins on another machine or in another process
- High performance:
  - ~1.2 GB/s shown here with interprocess communication
  - Saturating 10 Gb Ethernet links has been demonstrated



pvAccess: Command Line Tools

```
>pvinfo 13SIM1:Pva1:Image
CHANNEL  : 13SIM1:Pva1:Image
STATE    : CONNECTED
ADDRESS  : 164.54.160.82:5075
epics:nt/NTNDArray:1.0
    union value
        boolean[] booleanValue
        byte[] byteValue
        short[] shortValue
        int[] intValue
…
        uint[] uintValue
        ulong[] ulongValue
        float[] floatValue
        double[] doubleValue
…
    int uniqueId
    time_t dataTimeStamp
        long secondsPastEpoch
        int nanoseconds
        int userTag
…
```

```
>pvget 13SIM1:Pva1:Image -r
'field(uniqueId,dimension)'
13SIM1:Pva1:Image
structure
    int uniqueId 886580
    dimension_t[] dimension
        dimension_t
            int size 1024
            int offset 0
            int fullSize 1024
            int binning 1
            boolean reverse false
        dimension_t
            int size 1024
            int offset 0
            int fullSize 1024
            int binning 1
            boolean reverse false
```

NDPluginPVA and pvAccess Driver Demo

NDDriverStdArrays

- Driver that allows any EPICS Channel Access client to create NDArrays in an areaDetector IOC.
- Logical inverse of NDPluginStdArrays
  - That plugin converts NDArrays in an IOC into standard EPICS waveform records for use by Channel Access clients.
  - Also writes to additional records to describe the array structure. (SizeX, SizeY, ColorMode, etc.)
- NDDriverStdArrays
  - This driver receives EPICS waveform records from CA clients
  - Converts them to an NDArray in the IOC.
  - Clients must also write to additional EPICS records to define the structure of the NDArray.
- Similar in concept to pvaDriver
  - pvaDriver only works with EPICS V4 PVAccess clients
  - NDDriverStdArrays works with EPICS V3 Channel Access clients.

NDDriverStdArrays Use Case
David Vine, LBNL

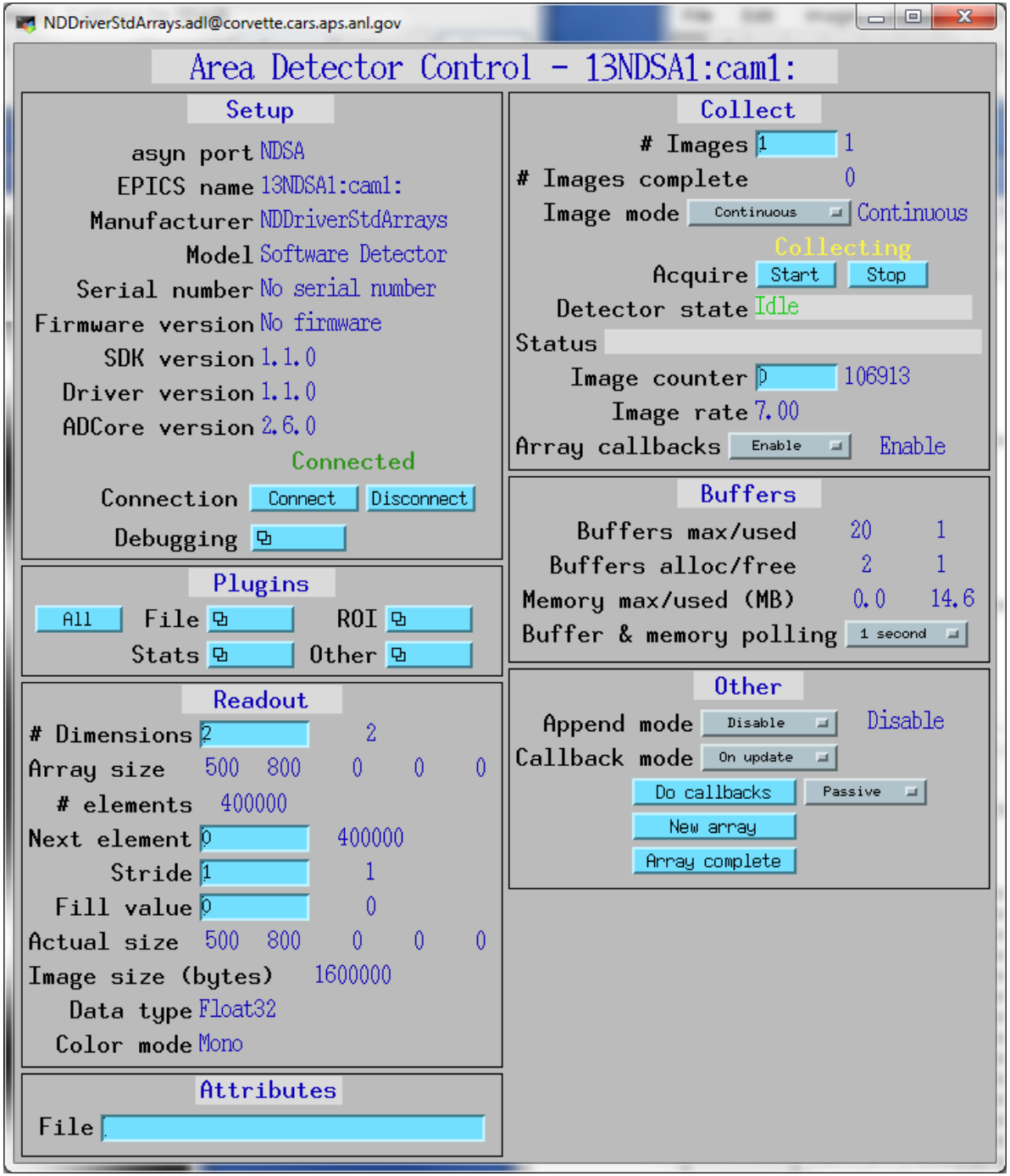
- Display data from sscan record
- Simple Python client reads data from sscan record, writes to NDDriverStdArrays
- Allows real-time display of 2-D scans
- Can write data in standard HDF5 file format
- Any other AD plugins can be used on the scan data

NDDriverStdArrays

- Driver receives 1-D arrays, normally from an EPICS waveform record.
- NDimensions record controls actual number of dimensions to use when creating the NDArray.
- Dimensions array record controls actual size of each of the NDimensions dimensions.
- ColorMode record can be used to define the color mode (Mono, Bayer, RGB1, RGB2, or RGB3) of the NDArray.
- Data type of the NDArray set by the DataType record.
  - Int8, UInt8, Int16, UInt16, Int32, UInt32, Float32, or Float64.
- Datatype (FTVL) of the EPICS waveform record can be
  - CHAR, UCHAR, SHORT, USHORT, LONG, ULONG, FLOAT, or DOUBLE.
- Driver does any required type conversion between the waveform record data and the NDArray data.

NDDriverStdArrays

- 2 modes of operation, AppendMode=Disable/Enable
  - Disable: each write to the waveform record generates a complete NDArray.
  - Enable: waveform record contains just part of the NDArray data; data appended by multiple writes to the waveform record to construct the complete NDArray. sscan record client would work in this mode
- Records in Enable mode:
  - NewArray, NextElement, Stride, ArrayComplete
  - CallbackMode record controls when the driver does callbacks
    - OnUpdate, OnComplete, OnCommand


NDDriverStdArrays.adl@corvette.cars.aps.anl.gov
Area Detector Control - 13NDSA1:cam1:
Setup
asyn port NDSA
EPICS name 13NDSA1:cam1:
Manufacturer NDDriverStdArrays
Model Software Detector
Serial number No serial number
Firmware version No firmware
SDK version 1.1.0
Driver version 1.1.0
ADCore version 2.6.0
Connected
Connection Connect Disconnect
Debugging
Plugins
All File ROI
Stats Other
Readout
Dimensions 2 2
Array size 500 800 0 0 0
elements 400000
Next element 0 400000
Stride 1 1
Fill value 0 0
Actual size 500 800 0 0 0
Image size (bytes) 1600000
Data type Float32
Color mode Mono
Attributes
File
Collect
Images 1 1
Images complete 0
Image mode Continuous Continuous
Collecting
Acquire Start Stop
Detector state Idle
Status
Image counter 0 106913
Image rate 7.00
Array callbacks Enable Enable
Buffers
Buffers max/used 20 1
Buffers alloc/free 2 1
Memory max/used (MB) 0.0 14.6
Buffer & memory polling 1 second
Other
Append mode Disable Disable
Callback mode On update
Do callbacks Passive
New array
Array complete

NDDriverStdArrays Demo

Runs Python Unit Tests

- Complete NDArrays
- Line scans (appends line at a time)
- Point scans (appends pixel at a time)
  - 4 detectors signals, sent to 4 ROIs

Pilatus Driver Update

- Now supports Energy and ThresholdEnergy independently
  - ThresholdEnergy is used to set per-pixel discriminators
  - Energy is used to compute the correct flat-field normalization
- Previously EPICS only exposed ThresholdEnergy
  - Camserver was supposed to set Energy=2*ThresholdEnergy
  - Bug in some versions of camserver, e.g. our new 1M CdTe detector. It was freezing Energy at 60 keV, and we collected an entire run of improperly normalized data
- There are occasions when ThresholdEnergy = Energy/2 is not the optimum setting, i.e. to avoid strong fluorscence.

Pilatus Driver Update

- If camserver is saving TIFF files then driver now reads the TIFFImageDescription tag from the TIFF file.
  - This is a long string that camserver writes to the file containing all of the detector settings, including threshold, energy, etc.
  - Driver adds this information to the NDArray using an NDAttribute called TIFFImageDescription.
- The NDFileTIFF plugin in ADCore R2-6 was changed to write this complete attribute to the TIFFImageDescription tag in the new TIFF file.
- It will also be written by the NDFileNetCDF, NDFileHDF5, and NDFileNexus plugins.
  - However these plugins are currently limited to 256 character string attributes, so some of the information will be lost because the string is longer than 256 characters.

ADAndor

- Added support for Electron Multiplying (EM) Gain.
- Added ability to set the BaselineClamp in the Andor SDK.
- Enforce minimum values of ADShutterOpenDelay and ADShutterCloseDelay based on query of SDK.
- Implemented ReverseX and ReverseY.
- Fixed bug with AndorPreAmpGain; previously it was not actually calling SetPreAmpGain().
- Added support for Full Vertical Binning (FVB) readout mode.
- Added support for EPICS shutter control.

Pixirad driver update

- CdTe pixel array detector available with multiple chips.
  - Detector pool has Pixirad-2, 402x1024. Now working at 1-ID.
- Updated driver to work with new PIII chip.
  - Square 62 um pixels pixels, not hexagonal.
  - Single chip is 402x512 pixels.
  - Pixel mode (PM). Each pixel is independent.
    - Highest rate capability (> 1MHz), no sharp color separation, multiple counts if threshold is low.
  - Neighbor Pixel Inhibit mode (NPI).
    - Only one counter per event allowed to count. The hit is allocated to the pixel receiving the highest fraction of the total charge. Highest position resolution and the lowest possible noise but sharpness of the color separation not completely solved yet
  - Pixel Summing mode (PSM).
    - The signals of 4 neighbor pixels are summed together in each pixel to correctly evaluate the total energy of any event involving up to 4 pixels.

Other detectors

- New ADCameraLink, ADPCO drivers
  - Drivers for PCO cameras from Tim Madden
- New ADLambda driver
  - Driver for Lamda pixel array detectors from John Hammonds
- Many detectors (Pilatus, Prosilica, Point Grey, marCCD, mar345, Andor, etc.)
  - Added support for SerialNumber, FirmwareVersion, SDKVersion, DriverVersion, and ADCoreVersion which were added in ADCore R2-6.
- ADSimDetector, ADCSimDetector, pvaDriver
  - Moved out of ADExample and into their own repositories
- ADExample no longer used

ADSupport

- New repository ADSupport
- Source code for all 3rd party libraries used by ADCore.
  - hdf5, jpeg, netCDF, nexus, szip, tiff, xml2, zlib
- ADBinaries no longer used.
  - Pre-built libraries for Windows too difficult to maintain, now all built from source using EPICS build system
- ADSupport can be built for
  - Windows (Visual Studio or MinGW, 32/64 bit, static/dynamic),
  - Linux (currently Intel architectures only, 32/64 bit),
  - Darwin
  - vxWorks (currently big-endian 32-bit architectures only).
- vxWorks previously supported only netCDF file plugin
  - All file saving plugins now supported on vxWorks 6.x (TIFF, JPEG, netCDF, HDF5, Nexus).
  - HDF5 and Nexus are not supported on vxWorks 5.x because the compiler is too old.
  - Could imagine migrating saveData to HDF5 since it now works on all platforms.

Optional Support

- All 3rd party libraries now optional.
  - For each library XXX 4 Makefile variables control support for that library. XXX can be JPEG, TIFF, NEXUS, NETCDF, HDF5, XML2, SZIP, and ZLIB.
  - WITH_XXX YES then drivers or plugins that use this library will be built. NO then drivers and plugins that use this library will not be built.
  - XXX_EXTERNAL YES then the library is not built in ADSupport, but assumed to be found external to the EPICS build system. NO then the XXX library will be built in ADSupport.
  - XXX_DIR If defined and XXX_EXTERNAL=YES then the build system will search this directory for the XXX library.
  - XXX_INCLUDE If defined then the build system will search this directory for the include files for the XXX library.
- All EPICS modules except base and asyn are now optional.

NDPluginOverlay

- New Ellipse shape to draw elliptical or circular overlays.
- Improved efficiency by only computing the coordinates of the overlay pixels when the overlay definition changes or the image format changes.
  - Important for ellipse because it uses trig functions
- Added CenterX and CenterY parameter for each overlay.
  - Can now specify the overlay location either by PositionX and PositionY, (upper left corner), or by CenterX and CenterY,
  - If CenterX/Y is changed then PositionX/Y will automatically update, and vice-versa.
- Changed the meaning of SizeX and SizeY for the Cross overlay shape.
  - Previously the total size of a Cross overlay was SizeX*2 and SizeY*2.
  - It is now SizeX and SizeY.
  - Now consistent with the Rectangle and Overlay shapes, i.e. drawing each of these shapes with the same PositionX and SizeX/Y will result in shapes that overlap in the expected manner.

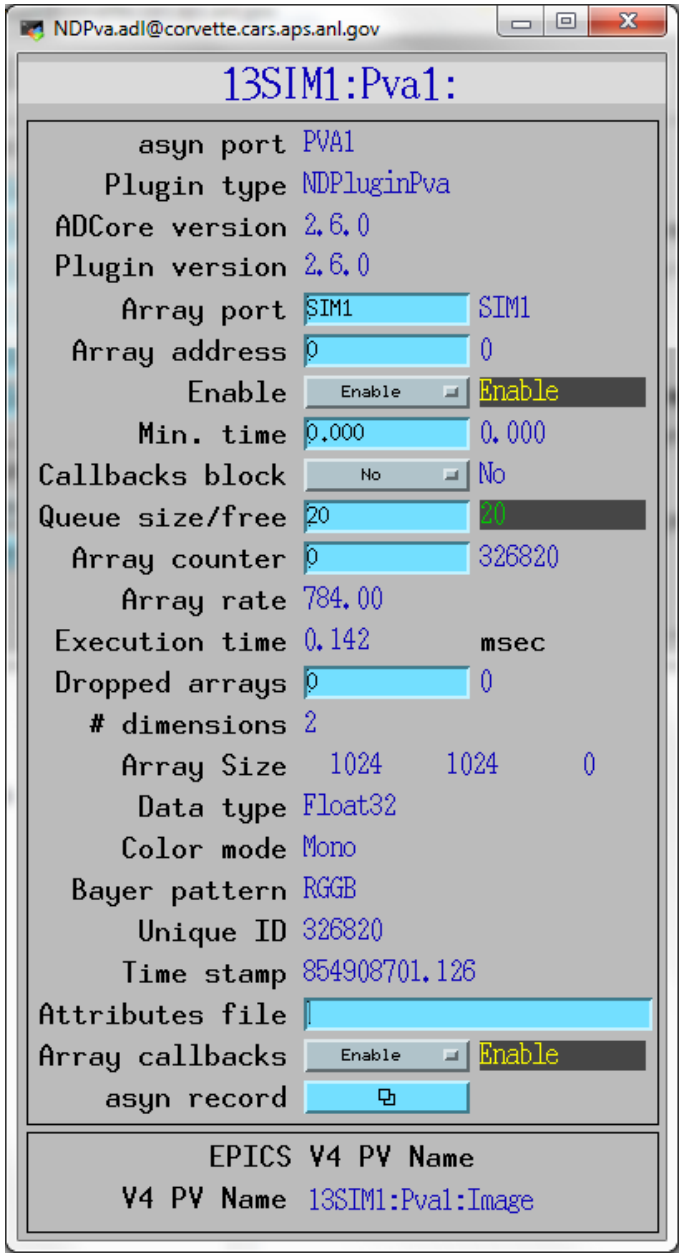
Other Plugin Enhancements

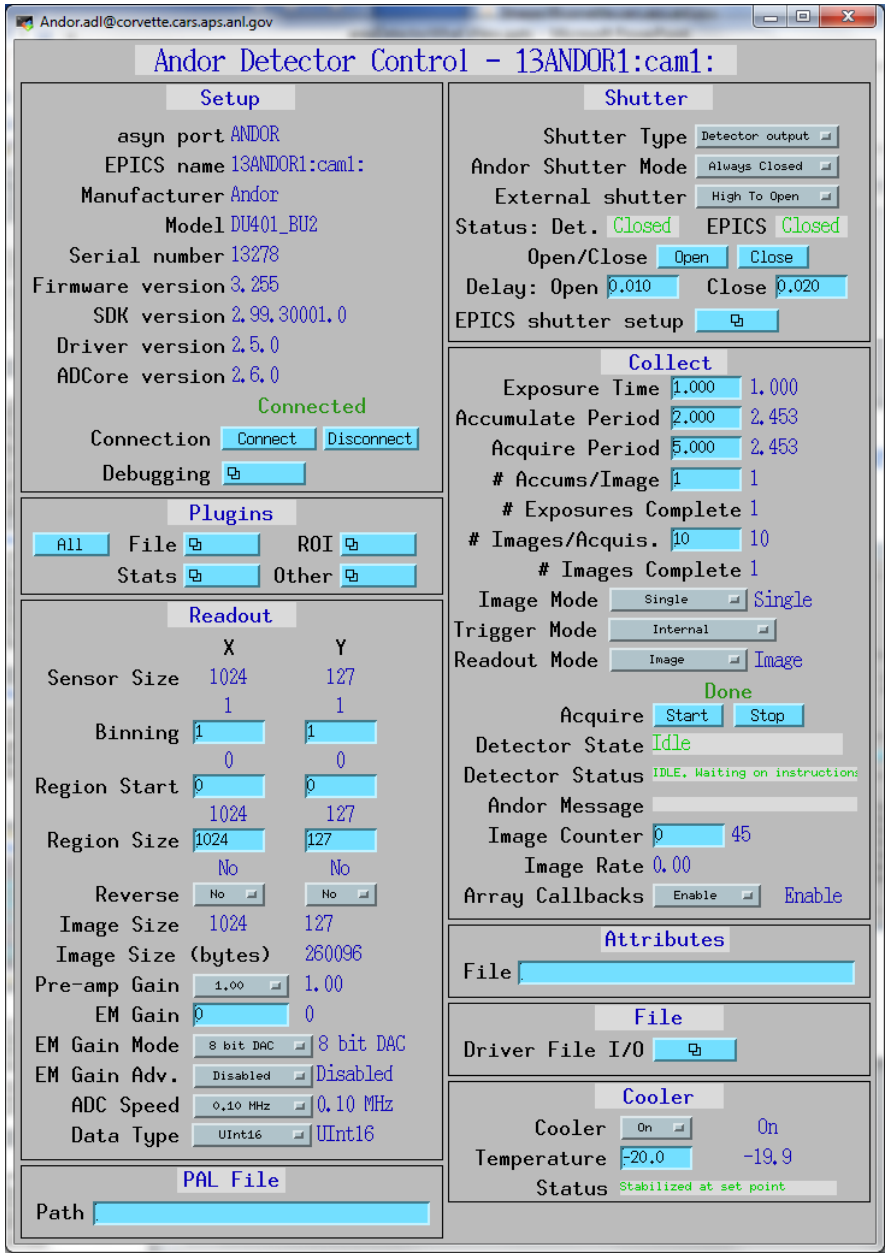
- All plugins: ElapsedTime PV for last execution.
  - This gives the execution time in ms the last time plugin ran.
  - Works both with BlockingCallbacks=Yes and No.
  - Convenient for measuring plugin performance without having to run detector at high frame rates.
- NDPluginROI
  - Added CollapseDims to optionally collapse (remove) output array dimensions whose value is 1. For example an output array that would normally have dimensions [1, 256, 256] would be [256, 256] if CollapseDims=Enable
- NDPluginStats
  - Added calcuations of 3rd and 4th order image moments, this provides skewness and kurtosis.
  - Added eccentricity and orientation calculations.
- NDPluginFile
  - If the NDArray contains an attribute named FilePluginClose and the attribute value is non-zero then the current file will be closed.

Base Class Enhancements

- asynNDArrayDriver, NDArrayBase.template, NDPluginBase.adl, ADSetup.adl, all plugin adl files
  - ADCoreVersion_RBV, DriverVersion_RBV.
  - Show the version of ADCore and of the driver or plugin that the IOC was built with.
- ADDriver, ADBase.template, ADSetup.adl
  - SerialNumber_RBV, FirmwareVersion_RBV, and SDKVersion_RBV.
  - Show the serial number and firmware version of the detector, and the version of the vendor SDK library that the IOC was built with.
- Because NDPluginBase.adl and ADSetup.adl grew larger all plugin and driver adl files have changed their layouts.

Base Class Enhancements





Viewers

- EPICS_AD_Viewer enhancements
  - Automatically set the contrast when a new window is created.
    - Eliminates the need to manually set the contrast when changing image size, data type, and color mode in many cases.
  - When image window is automatically closed and reopened because the size, data type, or color mode changes the new window is now positioned in the same location as the window that was closed.
- Gaussian Profiler
  - Like the normal ImageJ line profile in “live” mode, but with fitting of Gaussian and live display of fit parameters

Viewers

- EPICS_AD_Controller. Allows using the ImageJ ROI tools (rectangle and oval) to graphically define the following:
  - The readout region of the detector/camera
  - The position and size of an ROI (NDPluginROI)
  - The position and size of an overlay (NDPluginOverlay)
  - The plugin chain can include an NDPluginTransform plugin which changes the image orientation and an NDPluginROI plugin that changes the binning, size, and X/Y axes directions. The plugin corrects for these transformations when defining the target object.
  - Chris Roehrig wrote an earlier version of this plugin.

Viewer Demo

Future Ideas: ADCore R3-0

- Put more functionality into ADDriver base class
  - Currently it does not do much, all code is in each driver for:
    - Doing callbacks to plugins
    - Processing new exposure time with writeFloat64 function
  - writeFloat64 in ADDriver base class would call setExposure() in derived class
  - Derived class would call ADDriver::doPluginCallbacks(), which would handle setting attributes, getting timestamp, calling plugins, etc.
- Model 3 motor driver and quadEM, which also use asynPortDriver, are written this way
- Demultiplexor/multiplexor plugin
  - Allow multiple plugins to work on the same data stream when it saturates a single core
- Simplify file saving modes (no more Single, Capture, Stream)

Future Ideas: ADCore R4-0

- Change NDArray to NTNDArray for passing data to plugins
- Utilize EPICS V4 infrastructure
- Smart pointers automatically eliminate all unnecessary copying
- V4 clients can immediately receive data with no need to convert to waveform records
- Bruno Martins has demonstrated this working
- Seems like a good path for the future