

EPICS V4 Expands Support to Physics Application, Data Acquisition, and Data Analysis



L. Dalesio, Gabriele Carcassi, Martin Richard Kraimer, Nikolay Malitsky, Guobao Shen, Michael Davidsaver, BNL, Upton, Long Island, New York, U.S.A, Ralph Lange, Bessy, Berlin, Germany, Matej Sekoranja, Cosylab, Ljubljana, Slovenia, James Rowland, Diamond Light Source, Oxfordshire, England, Greg White, SLAC, Menlo Park, California, U.S.A. Timo Korhonen, PSI, Villigen, Switzerland.

EPICS
October, 2012

Outline

- Version 3 recap
- Version 4
- Version 4 Architecture for Machine Control
- Version 4 Architecture for Beam Line Control and DAQ
- Conclusions

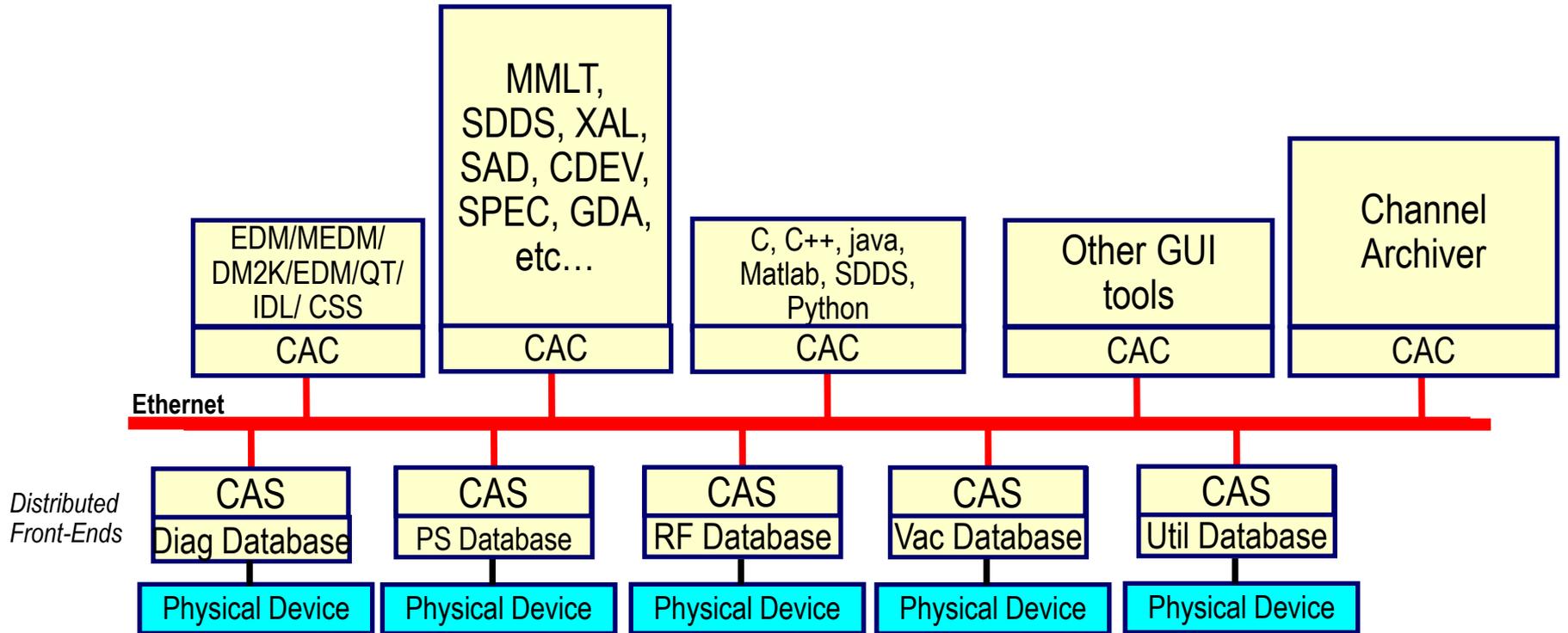
Version 3 Supports Instrumentation

- Records represented either an input signal, an output signal or an operation to perform on a set of signals
 - Analog input, analog output, (multi-bit)binary input, (multi-bit) binary output, motor, event, PID, calc, etc.....
 - Agreeing on what a device is – is difficult. Is it a power supply or a magnet? Does a motor have an LVDT, an encoder, back lash?
- Records implement continuous control in an autonomous controller to perform DCS functionality.
- Many different types of research and industrial facilities successfully applied this to their plant for equipment control.
- Process Variables (PVs) are available across the network
 - Any field of any record can be a process variable.
 - Functions on PVs are: get, put, monitor
 - This service was designed and implemented to be robust and fast (15K PVs per second to a client on a 100 MB network)
 - Channels always have a time stamp, alarm severity, and alarm status – the simple data type was not useful in most cases
 - Channels have metadata to describe display, control, and alarm information.
- MANY clients were developed on this interface in many languages on many operating systems implementing the full range of SCADA capabilities.
 - With two site developing EPICS, there were two display managers.

Version 3 Has Limited Support for Devices

- Records did not operate on things more complex than scalar signals.
 - No time domain, no frequency domain, no images.
 - No way to represent things more complicated than scalar signals and 1 dimensional arrays
- Process Variables available across the network could not support everything needed
 - No atomic command / response mechanism
 - No way to ask for a PV subject to parameters.
 - PVs metadata did not always fit properly for every field of a record – such as the display precision – what is the time stamp of this?
 - Typically a get is done on connection for display, alarm limits, and control metadata changes are not reflected.
 - Meta data was sent all of the time, so only time stamp and current alarm information is monitored.
- MANY clients added layers on top of V3 Process Variables to implement more complex data models

EPICS Version 3 Architecture



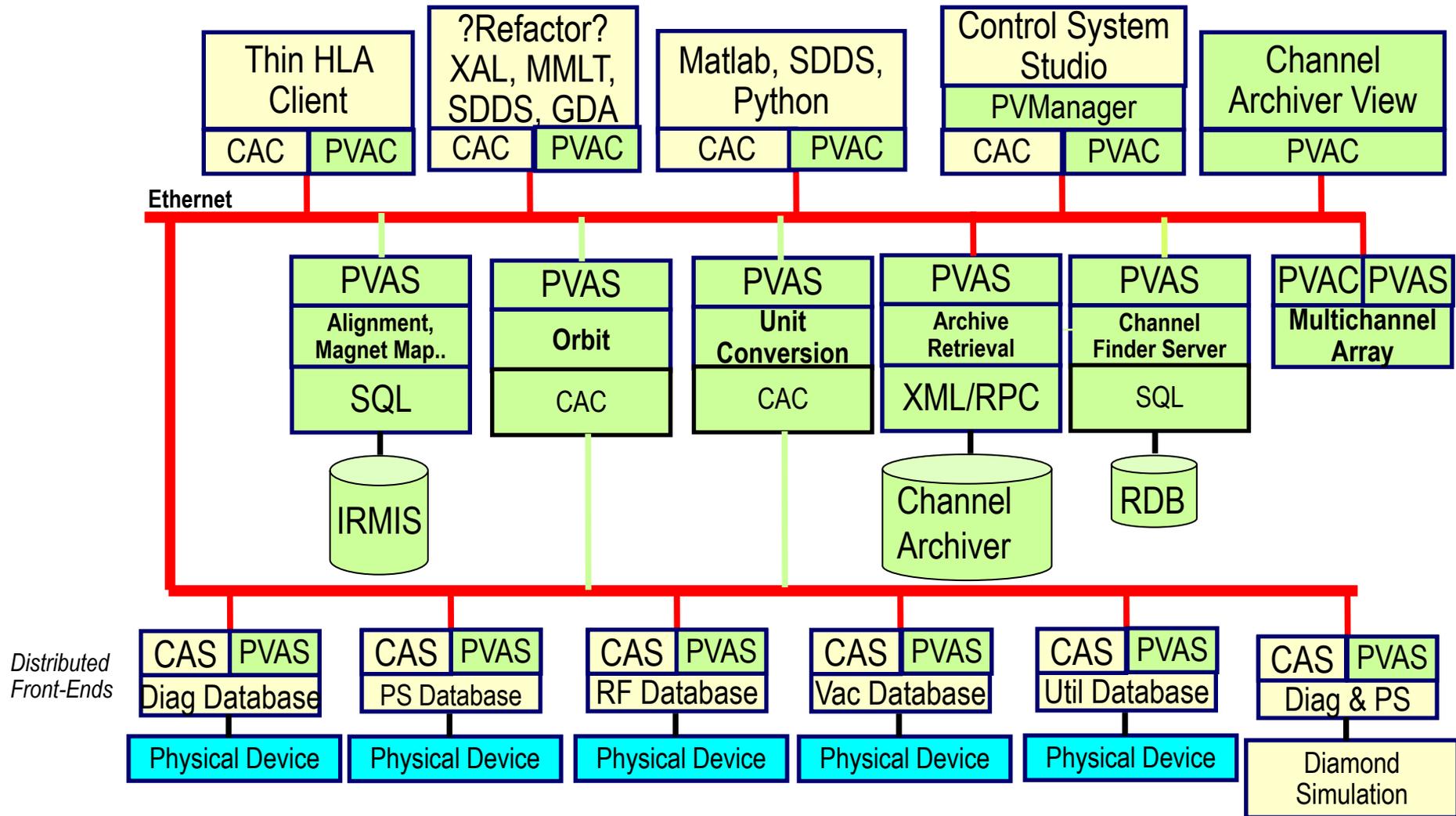
Version 4 Supports Complex Data Structures

- Java IOC can represent devices
 - We will likely not implement devices, as it is still difficult to agree on what these are
 - We will use V3 records at NSLS II.
- PVData (PVs) are available across the network
 - Functions on PVs are: get, put, monitor, put/process/get (command/response)
 - It is also hard to create object models on more complex devices such as a telescope or an accelerator.
- Normative types are defined to provide metadata for more complex constructs: multi-channel array, table, N dimensional Array, Image.
 - PVData always has a time stamp, alarm severity, and alarm status
 - Vectors have useful metadata and distinctions: time domain vector, frequency domain vector, histogram
 - Operations can be performed on two PVs with the same normative types.
 - We have NOT
- PVService supports creating middle layers services
- MANY servers are being developed on this interface to implement middle layer through a collaboration for physics applications. A second collaboration is being established for beam line control, data acquisition, and data analysis.

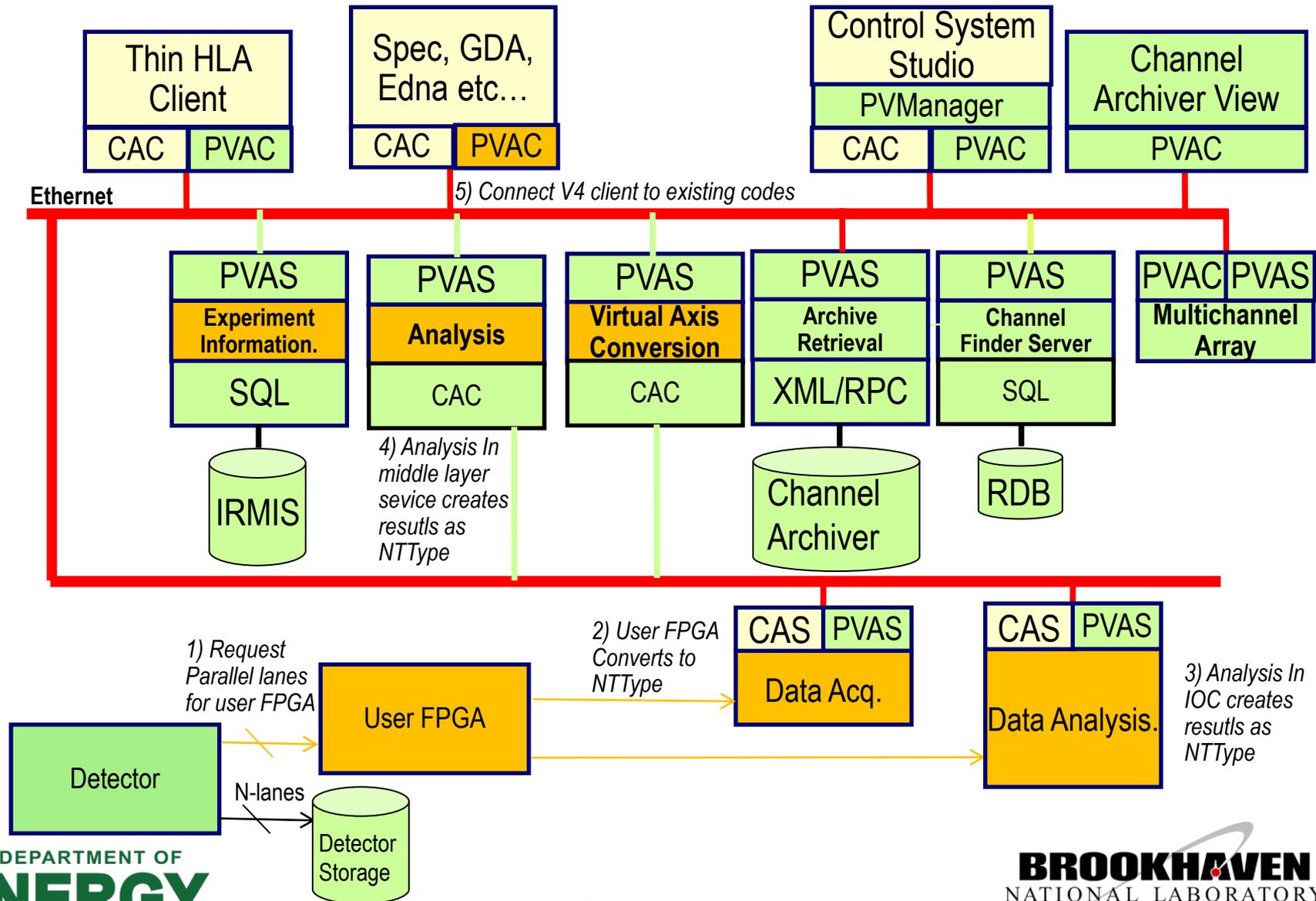
Specific Normative Data Types

- NTMultichannelArray – represent a collection of single values as an ordered array
e.g. all of the temperatures along a beam line or the Australian Synch's Concatenate Record used on AI's
- NTTimeDomainArray
e.g. a scope trace from a digitizer or the Circular Buffer in the Compress Record
- NTHistogram e.g. information on a 60 Hz power supply RB posted every hour or the Histogram in the Compress Record
- NTNDArray e.g. multiple frames of a detector taken at 1 KHz for 1 second
- NTFrequencyDomainArray
e.g. FFT of 10 KHz data taken for 1 second to study noise frequencies or FFT record output
- NTStatistic e.g. any data being compressed from its original rate – fast sampling in hardware to EPICS DB,
new function on any database waveform, response from a request to an archive server
- NTImage e.g. image data being collected at a detector into the areaDetector application
- NTTable e.g. a way to return any list of values or collection of name, value pairs of different data type such as twiss parameters
or the metadata for a camera set up: filter, exposure time, camera used, etc...
This is the catch all data type that can define a structure of single values or arrays (of the same length)
- NTChannelFinderDirectory
e.g. returned as an ordered list of PVs from a query to a directory service to populate a multi-channel array or table

Applying Version 4 to Machine Control



Version 4 to Beam Line Control and DAQ



Status

- PVData: Done (C++, Java)
- PVAccess: Done, performance like CA3, (C++, Java)
- V3 IOC Integration: Done
 - Add-on for V3 IOCs to serve all records as V4 PVAccess [DLS]
- PVService: Several (Java, C++, Python)
 - Gather [PSI], Directory Service (ChannelFinder) [BNL], Save-Set (MASAR) [BNL], History (ChannelArchiver) [DLS], Model (Tracy, Elegant) [BNL&PSI], ..
- V4 Clients: Several (Python)
 - NSLS2 physics: Orbit display & correction, beam-based alignment
- Normative types: Being defined
 - Scalar, Matrix, Table: Done
 - Image,: Ongoing
- V4 IOC:
 - Java implementation served as demo for PVData, PVAccess, .. but so far not deployed
 - C++ implementation about to start

Conclusions

- The interface is intended to allow us to create a standard client/server architecture for high level applications such as: areaDetector, Matlab Middle Layer Toolkit, SAD, SDDS, XAL, GDA, MDS+
- NSLSII is committed to apply this technology to physics applications, and is starting to apply it for applications in experiment control, data acquisition, and data analysis.
- Low level applications will be developed for large data sets in either the V4 Java IOC or by extending the V3 IOC to connect to PVAccess.
- New structures are easy to create – but we plan to carefully limit these to general and useful normative types.
- We are in the stage of development most similar to the early days of the EPICS collaboration: our first applications are being deployed, the normative types are not settled, there are still many challenges and lot of excitement.