Argonne NATIONAL LABORATORY

EPICS Version 4 Development

Andrew Johnson
AES/SSG

U.S. DEPARTMENT OF ENERGY

Outline

- A brief history
- Who is developing EPICS V4 and why
- What does “V4” actually mean?
- Why it will be better than V3
- Capabilities of pvManager
- The current roadmap
- Where to find out more
- Questions

History of EPICS V4

- 2006: Marty Kraimer left Argonne
  - Developing a new IOC architecture, first implementation in Java
  - Informed by abortive V4 design conversations over previous years
- 2007: Bob Dalesio joined BNL to manage NSLS-II Controls
  - Looking for more capabilities than EPICS V3 could provide
    - Higher level abstractions, structured data, services layer
  - SBIR funding paid for Marty to continue development
- Cosylab (Matej Sekoranja) developed the JCAS for Gemini
  - Plugged this and CAJ/JCA into Marty's now-working Java IOC
  - Marty & Matej started C++ development, funded by NSLS-II and SBIR grants
- 2010: First public appearance of names 'pvData' & 'pvAccess'
  - Based on design ideas and code from the Java IOC
  - SLAC, Diamond and BNL agreed to collaborate on development in July
  - EPICS V4 introduced to community at BNL EPICS meeting in October

EPICS V4 Working Group

- Formal charter adopted in September 2011, revised annually
- Website, mailing list & code repositories at SourceForge
- Weekly video conferences (G+ Hangout)
  - Agenda and minutes available online
- Workshop meetings every 3-6 months
- Current members:
  - BNL: Bob Dalesio, Guobao Shen, Nikolay Malitsky, Michael Davidsaver, Gabriele Carcassi
  - SLAC: Greg White (co-chair)
  - Marty Kraimer
  - Cosylab: Matej Sekoranja
  - ANL: Andrew Johnson (co-chair), Siniša Veseli
  - Diamond: David Hickin
  - PSI: Timo Korhonen, Dirk Zimoch
  - HZB: Ralph Lange, Benjamin Franksen

EPICS V4 Project Scope

- "EPICS V4 is a set of computer communications protocols, and a software framework, for high performance distributed control, message passing, and high level software services, as may be used in large scientific instruments and industrial plants." — V4 Working Group home page
  - The V4 protocol can be used alongside the V3 Channel Access protocol, so most V4 clients will be able to communicate seamlessly with V3 IOCs.
- NSLS-II and PSI are developing middle-layer V4 services that can communicate with both V3 and V4 IOCs
  - E.g. accelerator model and orbit servers, save/restore, etc.
- Diamond are working on integrating AreaDetector, using the V4 network protocol to distribute image-processing chains across IOCs.
- Two parallel working groups are collaborating on higher-level V4 applications
  - Control System Studio group: BOY, BEAST, etc.
  - Relational Database group: Channel finder, ELOG, cables, maintenance, spares, etc.

V4 Modules

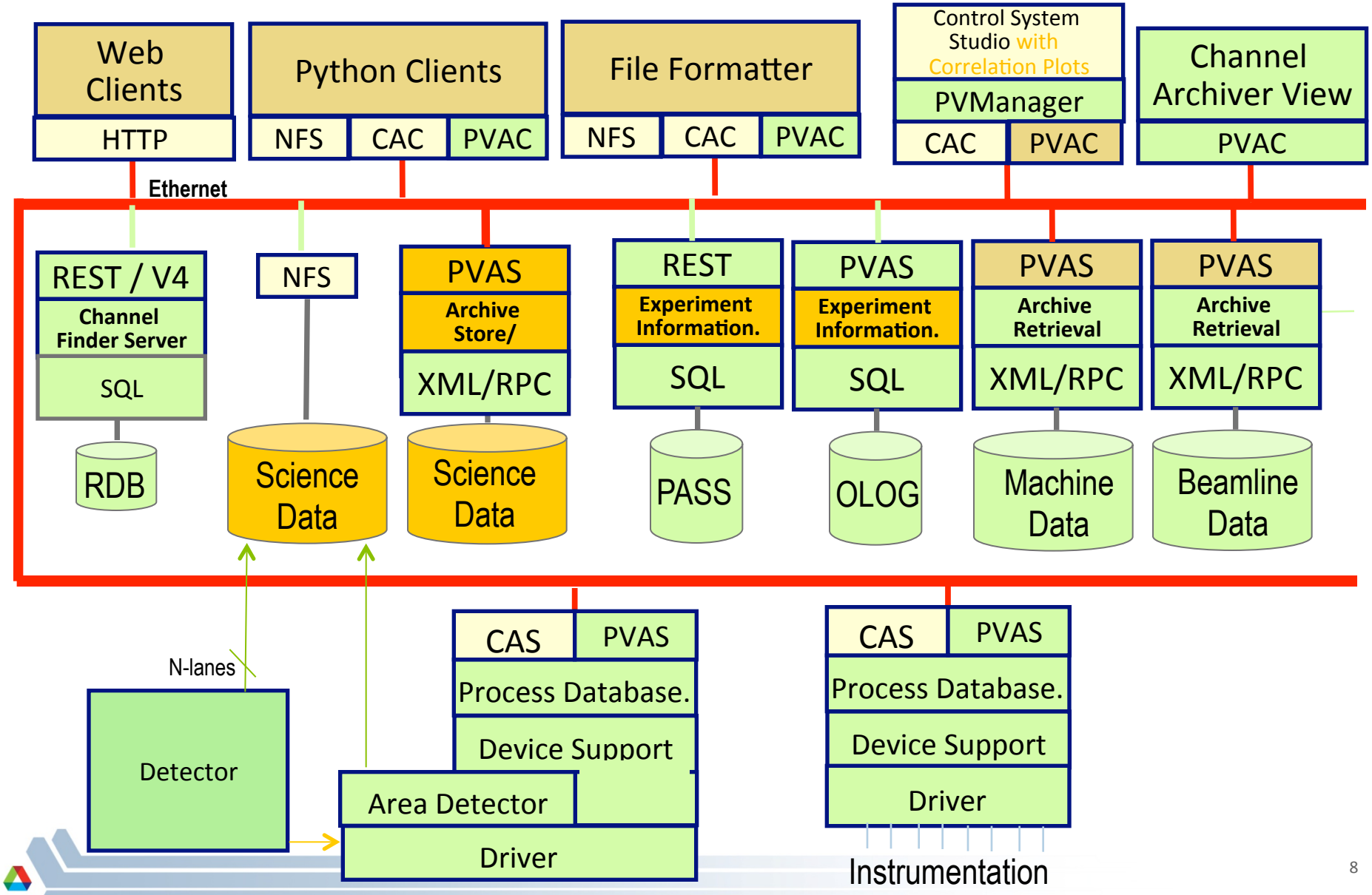
- pvCommon (Java and C++)
  - various utility libraries, including some of C++ Boost
- pvData (Java and C++)
  - Data types, structured data, introspection, serialization and deserialization
- pvAccess (Java and C++)
  - network protocol, both client and server side
- pvaSrv (C++ only)
  - IOC-specific pvAccess server code

What about IOCs?

- A V4 IOC *is* a V3 IOC, with a pvAccess server (pvaSrv) added
  - Still supports Channel Access
  - No changes needed to the existing database or record types
  - Currently V4 is built against Base 3.14.12.3
  - Next V4 major release will be built against Base 3.15.x
    - Various new IOC features come with 3.15
- A version of synapps for Base 3.15 is in development

- Marty's Java IOC is being renamed to reduce confusion

NSLS-II Experiment Control / DAQ Architecture



Advantages of V4

- Flexibility
  - pvData provides the usual data types (scalar and array), structs, and unions
    - Data structures have an introspection interface, generic code can access them
  - pvAccess can transport any pvData structure, CA only has DBR_xxx types
  - pvAccess supports RPC (command/response) operations
- Optional standardization
  - Normative Type specification defines common standard structures that generic tools will know how to handle
    - All have optional description, time-stamp & alarm condition fields
    - NTScalar: Single 8-, 16-, 32-, or 64-bit integer (signed or unsigned), float, double, or string
    - NTScalarArray: Array of the above types
    - NTMatrix: 2-dimensional (*n* by *m*) array of double values
    - NTTable: Rows of columnar values with column labels
    - NTHistogram: 1-dimensional histogram of counts with bin ranges
    - NTImage: AreaDetector's NDArray as a pvData structure
    - etc.

Higher-level Interfaces

- Converting a V3 IOC to V4 is just like adding a module (or upgrading Base)
  - Running the pvAccess server adds one line to st.cmd and Makefile
  - Next major release will support multi-channel synchronous get/put

- High level API: pvManager (currently Java only, comes with recent CSS)
  - Simplifies data collection & aggregation from pvAccess & Channel Access
  - Provides 3 APIs: Data interfaces, Building blocks, and Expressions
    - Notify specific threads, queue or cache incoming data, adjust data rates
    - pvManager can perform calculations, construct arrays and tables etc.
  - Understands the common V4 Normative Types directly
  - Use formulae instead of just pv names
    - Can reference pluggable data sources, formula functions, user-written services
      - For example query a JDBC database, run a script

What's in it for Beamlines?

- Split areaDetector processing pipelines across computers
  - Camera on one IOC, processing and saving on other IOCs
- Multi-value get/put operations
  - Get or put data to multiple fields of a record in a single operation
  - Synchronized operations (record stays locked)
  - E.g.: A scan client could completely configure a scan record in one put
- In future releases, multi-value operations will be able to cross records
  - A single get/put to several different records at once
  - Still synchronized (all records locked before first get/put)
  - E.g.: Sample the complete state of an experiment at once

Current work, future directions

- Support for Microsoft Windows
- Python bindings for pvAccess, both client and server
- Revising the Normative Type definitions
  - pvData now supports unions, not used in current NTs
  - Split NTImage into two types (AreaDetector, basic 2D-images)
- Efficient handling of large arrays and data structures
- Multicast network support
- Multi-channel atomic I/O for pvaSrv (IOC server)
- Simple APIs for basic client programs

- Eventually merge C++ modules into Base (4.x)

More Information

- EPICS V4 website
  - http://epics-pvdata.sourceforge.net/
  - Also linked from the EPICS website
- Links to
  - Downloads
    - Java & C++ code for version 4.3.0
  - Documentation
    - Protocol spec, Java & C++ API reference documents, Normative Type spec
  - Working Group
    - Members, minutes of hangouts and workshops