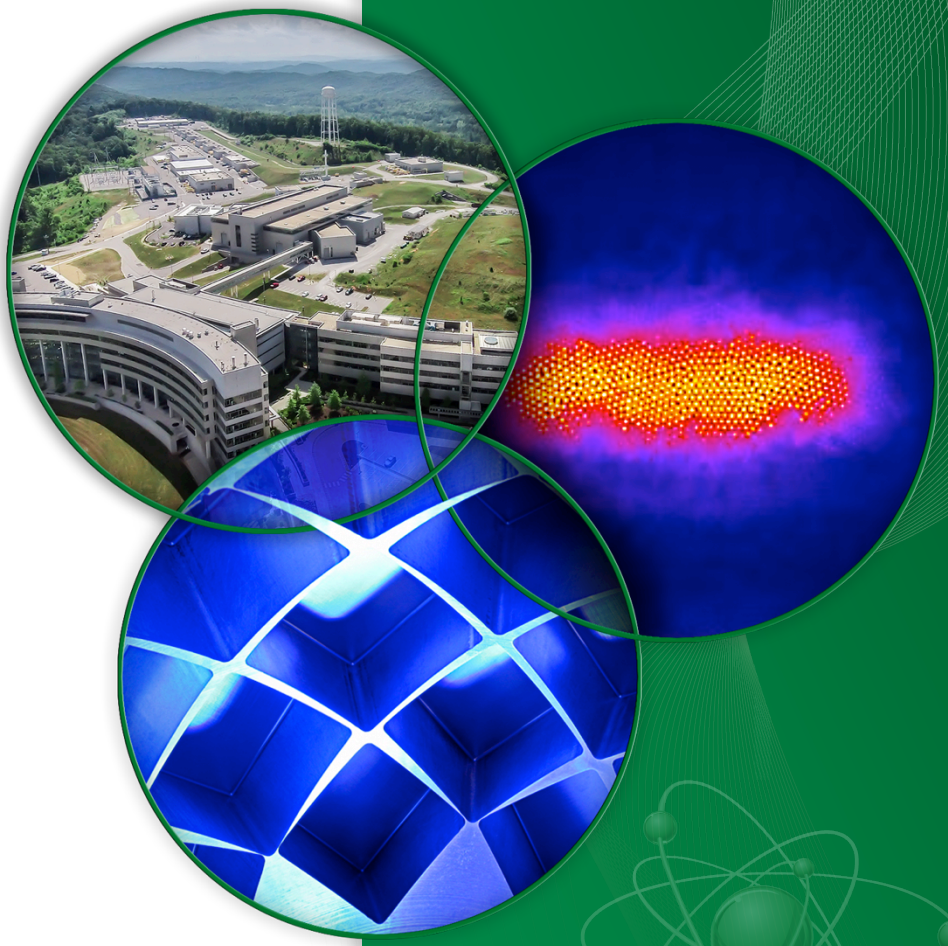
EPICS V4 Evaluation for SNS Neutron Data

Kay Kasemir

Oct. 2015

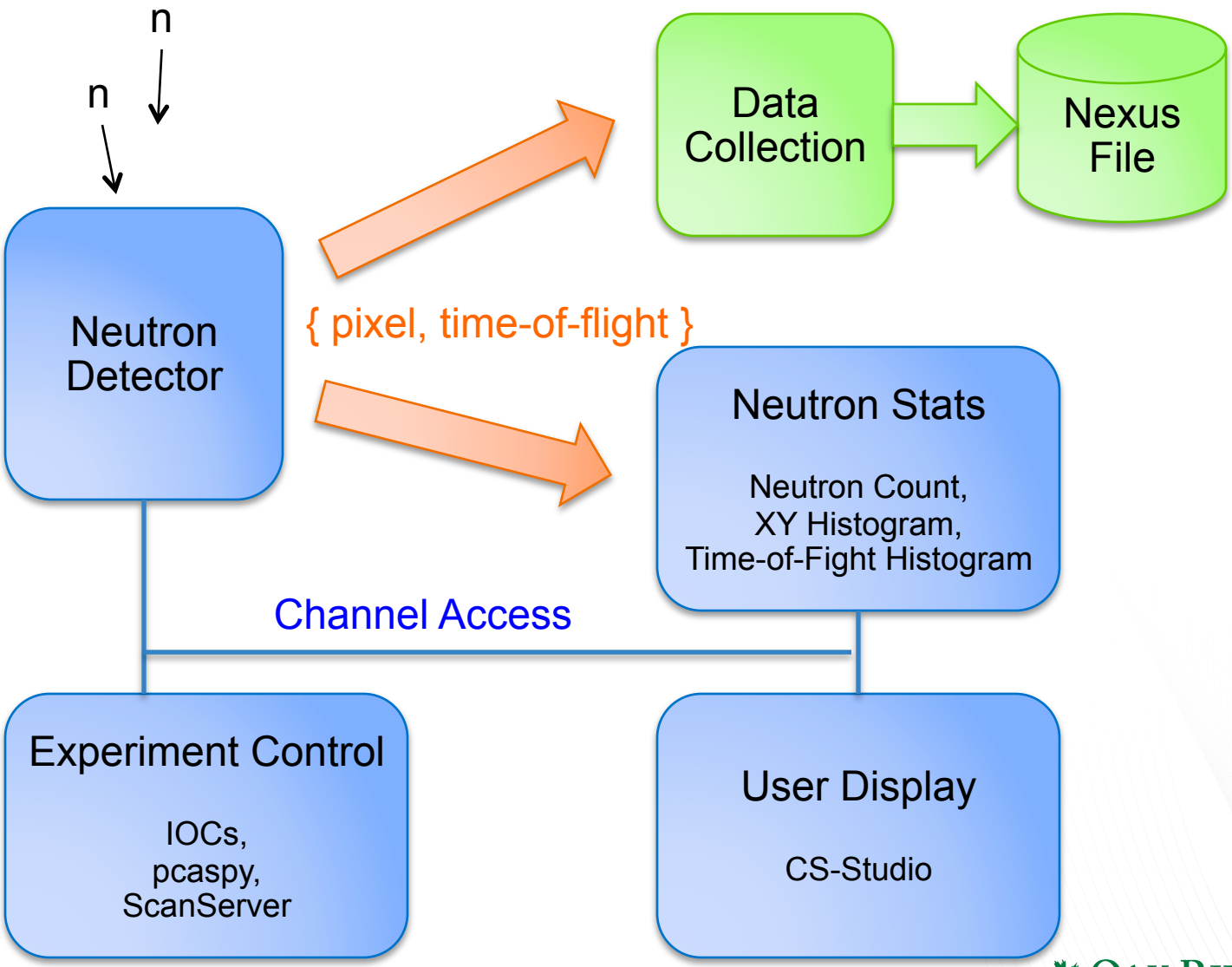


see also ICALEPS
Poster/Paper WEPGF105

ORNL is managed by UT-Battelle
for the US Department of Energy

OAK RIDGE National Laboratory | SPALLATION NEUTRON SOURCE

Basic Idea: V4 for SNS Neutron Data



V4 Rumors

- V4 Developer meeting == Discussion of NTImage
- V4 replaces Channel Access

  .. but only for Java
- Useful when you implement “Services”

  i.e. not when you don’t
- `struct dbr_time_double` has been replaced by

```
FieldBuilderPtr fb = fieldCreate->createFieldBuilder();
StructureConstPtr alarm =       // StructureConstPtr  == std::tr1::shared_ptr<const Structure> !!
   fb->add("severity", pvInt)->
       add("status", pvInt)->
       add("message", pvString)->createStructure();
StructureConstPtr timeStamp =
   fb->add("secondsPastEpoch", pvLong)->
       add("nanoseconds", pvInt)->
       add("userTag", pvInt)->createStructure();
StructureConstPtr structure  =
   fb->addArray("value", pvDouble)->
       add("alarm", alarm)->
       add("timeStamp", timeStamp)->createStructure();
```

What is V4?

pvData – Structured Data
- Java, C++
- Normative Types: Structs w/ time, alarm, ..

pvAccess – Network protocol
- Similar to CA
  - Search via UDP 5076
  - Connect by default on TCP 5075
- Server decides on byte order
- Partial transfers, whatever client requests
- Clever ‘size’: 1 byte if <255, …

Protocol freeze in Oct. 2014

Orig. SNS neutron network protocol	V4 pvData, pvAccess
‘events’ via UDP broadcast to any number of listeners	‘monitors’ via TCP to 2-3 listeners
Protocol documented in code	Protocol documented in specification
Custom MS Visual C++ code took years to get stable	Java, portable C++, python code with wider developer and user base
No ‘debug’ tools	pvinfo, pvget, CSS Probe, …
Clients receive neutron events	Clients can see last (=stale) value, then new events
Can put anything into network package	Need to fit into pvData

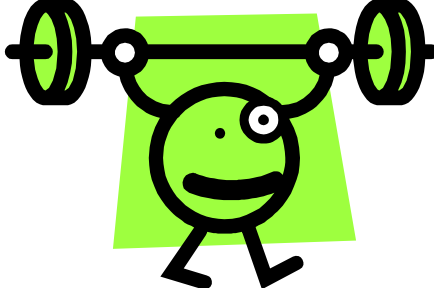
SNS Neutron Data

```
uint64    eventID
uint64    pulseID      // timeStamp
double    protonCharge
struct
{
    uint32 time_of_flight
    uint32 pixel
}         events[]
```

SNS Neutron Data as pvData

```
Structure
   // Time stamp for all;
   // eventID in .userTag
   time_t  timeStamp

   NTScalar protonCharge
      double  value

   NTScalarArray time_of_flight
      uint[]      value

   NTScalarArray pixel
      uint[]      value

   NTScalarArray position_x
      uint[]      value
   .. a few more optional elements
```

What you get for free



```
MINGW32:~
[ky9@bl9-dassrv1 v4test]$ ./pvinfo neutrons
CHANNEL   : neutrons
STATE     : CONNECTED
ADDRESS   : 10.111.18.131:5075
structure
    uri:ev4:nt/2012/pwd:NTScalar pulse
        ulong value
        double protonCharge
        time_t timeStamp
            long secondsPastEpoch
            int nanoSeconds
            int userTag
    uri:ev4:nt/2012/pwd:NTScalarArray time_of_fligh
        uint[] value
    uri:ev4:nt/2012/pwd:NTScalarArray pixel
        uint[] value


[ky9@bl9-dassrv1 v4test]$
[ky9@bl9-dassrv1 v4test]$
[ky9@bl9-dassrv1 v4test]$
```

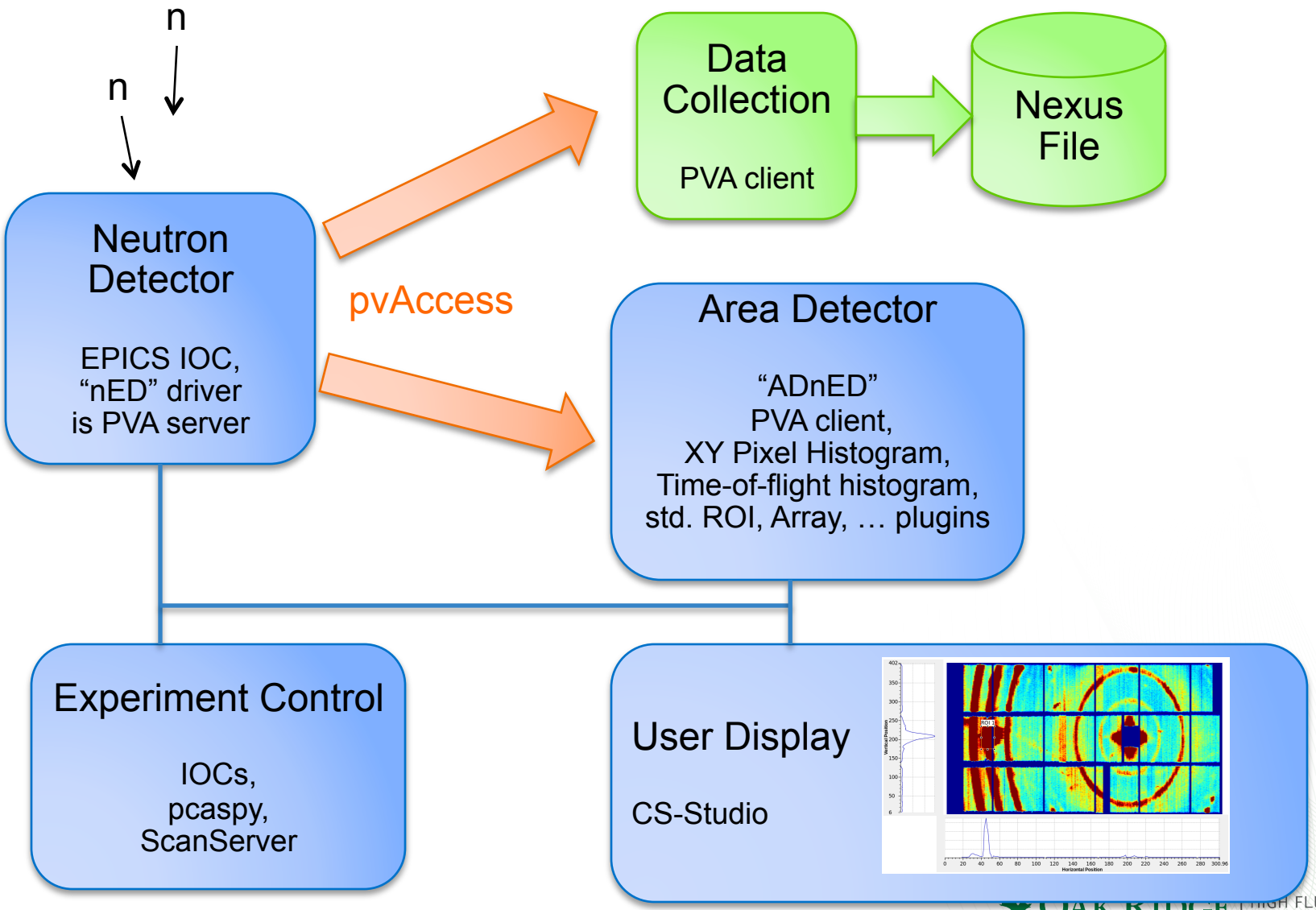
```
MINGW32:~
[ky9@bl9-dassrv1 v4test]$ ./pvget neutrons
neutrons
structure
    uri:ev4:nt/2012/pwd:NTScalar pulse
        ulong value 6471
        double protonCharge 2e+08
        time_t timeStamp
            long secondsPastEpoch 1404995064
            int nanoSeconds 153360791
            int userTag 0
    uri:ev4:nt/2012/pwd:NTScalarArray time_of_flight
        uint[] value [6471,6471,6471,6471,6471,6471,6471,6471,6
    uri:ev4:nt/2012/pwd:NTScalarArray pixel
        uint[] value [64710,64710,64710,64710,64710,64710,64710


[ky9@bl9-dassrv1 v4test]$
```

```
MINGW32:~
[ky9@bl9-dassrv1 v4test]$
[ky9@bl9-dassrv1 v4test]$
[ky9@bl9-dassrv1 v4test]$ ./pvget -m -r "field(pulse.timeStamp.secondsPastEpoch,pixel)" neutrons
neutrons
structure
    structure pulse
        structure timeStamp
            long secondsPastEpoch 1404995234
    uri:ev4:nt/2012/pwd:NTScalarArray pixel
        uint[] value [233080,233080,233080,233080,233080,233080,233080,233080,233080,233080]


neutrons
structure
    structure pulse
        structure timeStamp
            long secondsPastEpoch 1404995234
    uri:ev4:nt/2012/pwd:NTScalarArray pixel
        uint[] value [233090,233090,233090,233090,233090,233090,233090,233090,233090,233090]
```

Tests

- Server that generates fake neutron events
  - Similar to pvDatabaseCPP example
  - Can run standalone or in V3 IOC
- “pvget –m” for initial tests
  - Hit CPU limit because of string formatting
- Custom client
  - Tests for missing ‘event ID’
- Results:
  - Saturating 1GB network around 15M SNS events/sec
    - 100 updates/sec, each with 150000 events
    - 1 ‘event’ ⇔ T.o.F + pixel ⇔ 8 bytes, and indeed about 8 b/evt on network!!
  - On 10GB network 100M SNS events/sec w/o problems
    - Limit was CPU load, not network
  - Required for SNS: 10M events/sec?

Installed on 5 SNS Beam Lines



Looking Good!

+ Can package data in flexible ways
+ Already have network tools to inspect, monitor
+ Performance is good
+ Community of developers and users

  Thanks to Matej Sekoranja, Marty Kraimer, David Hickin for fast bug fixes and help

See FRIB EPICS Meeting for detailed presentations on nED, ADnED