Archive Engine for Large Data Sets

Nikolay Malitsky

EPICS Collaboration Meeting
San Francisco, USA

October 5, 2013

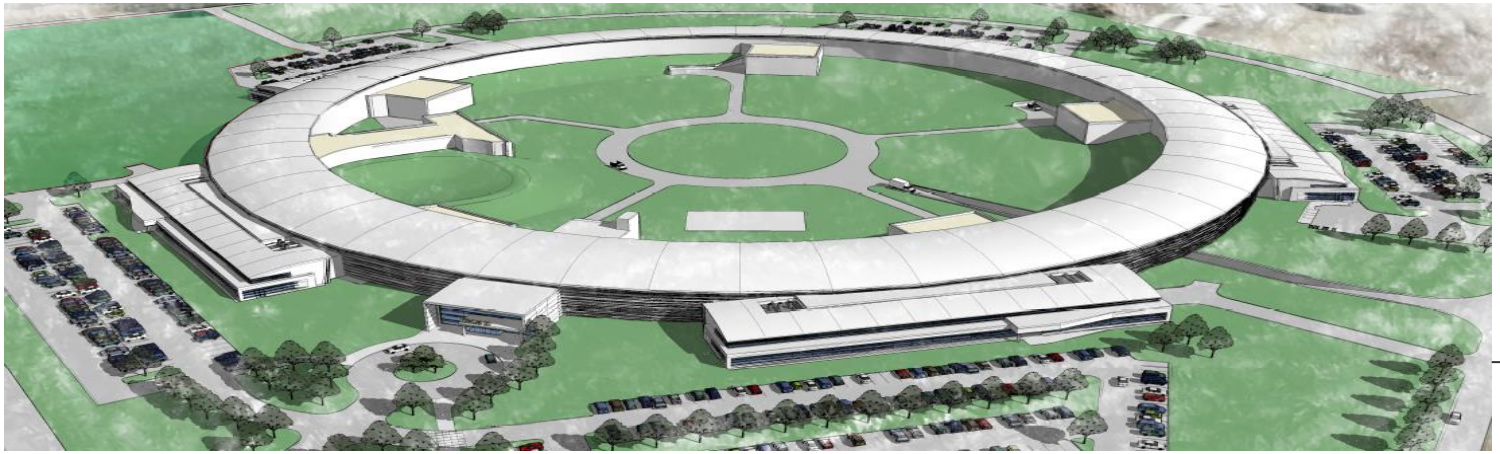
BROOKHAVEN NATIONAL LABORATORY
a passion for discovery

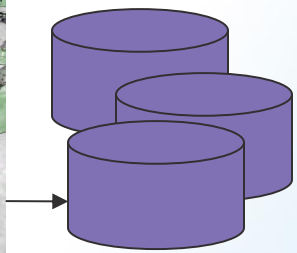
U.S. DEPARTMENT OF ENERGY | Office of Science

Outline

- **Objectives**: new scope, new scale, new applications
- **Incremental Approach**: big picture, classic and HDF5 versions
- **Major techniques:** type system, chunk model, HDF5 file format
- **Next Use Case:** time series of frames

Objectives

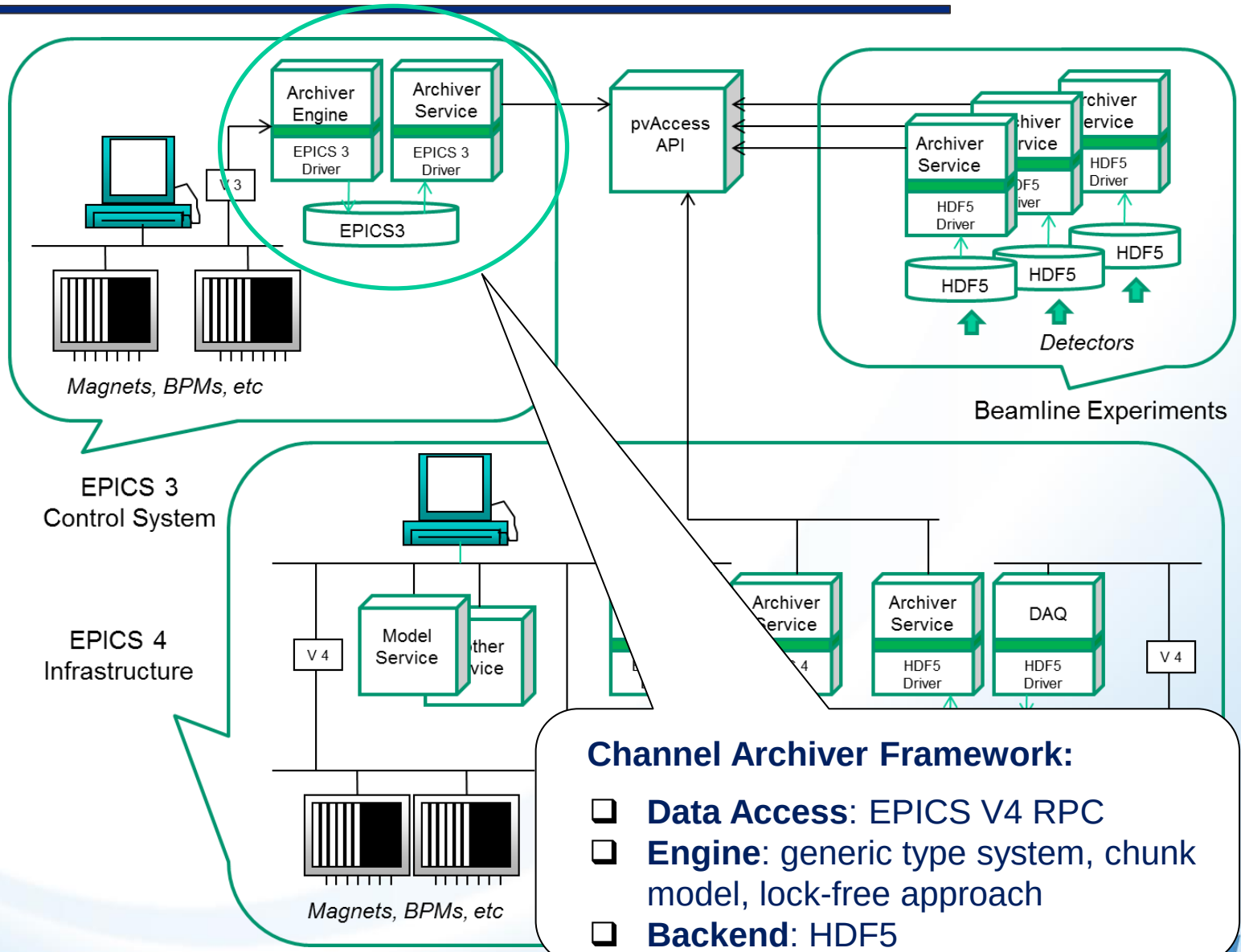




2TB/day

- **New Scale:** 1 M samples/s from 30,000 heterogeneous physical devices of power supplies, diagnostics, RF, vacuum system, *etc*
- **New Scope:** Transition from EPICS 3 to EPICS 4 bringing the middle layer and support of the user-defined data types
- **New Applications**: Beamline DAQ

Integrated and Incremental Approach



Channel Archiver Framework:

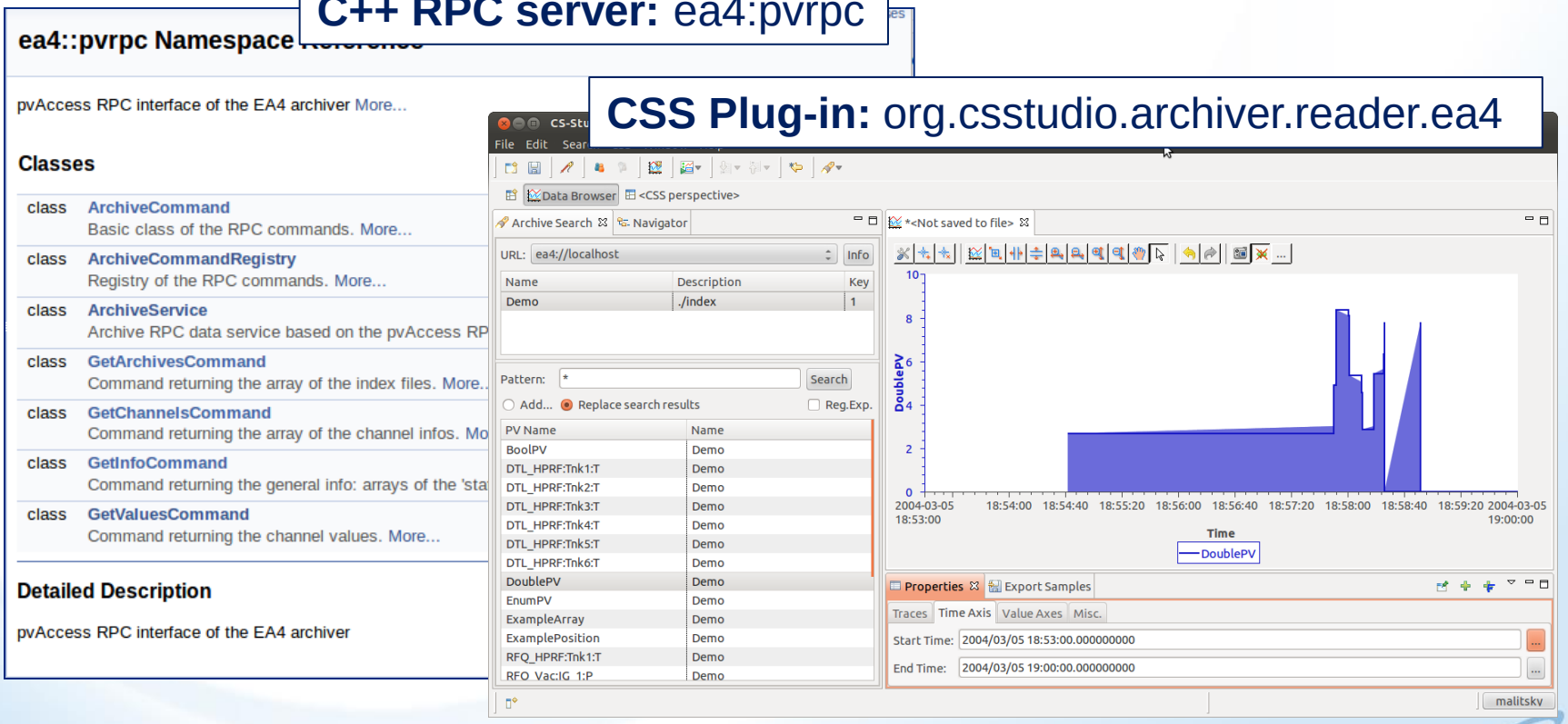
- **Data Access**: EPICS V4 RPC
- **Engine**: generic type system, chunk model, lock-free approach
- **Backend**: HDF5

Classic Version

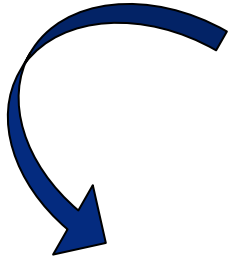
- **Data Access**: XML-RPC and EPICS V4 RPC
- **Engine**: Lock-free Circular Buffer => 100 K scalar samples/s
- **Backend:** Index + original Data files => 300 K scalar samples/s

C++ RPC server: ea4:pvrpc

CSS Plug-in: org.csstudio.archiver.reader.ea4



Data Access: Get Channel Values



```
result = archiver.values(
         int key,
         string name[],
         int32 start_sec,  int32 start_nano,
         int32 end_sec,  int32 end_nano, int32 count,
         int32 how)
```

```
result := { string name,  meta, int32 type,
            int32 count,  values }[]
```

```
meta := { int32 type;
          type==0: string states[],
          type==1: double disp_high,
                   double disp_low,
                   double alarm_high,
                   double alarm_low,
                   double warn_high,
                   double warn_low,
                   int prec,  string units
        }
```

```
values := { int32 stat,  int32 sevr,
            int32 secs,  int32 nano,
            <type> value[] } []
```

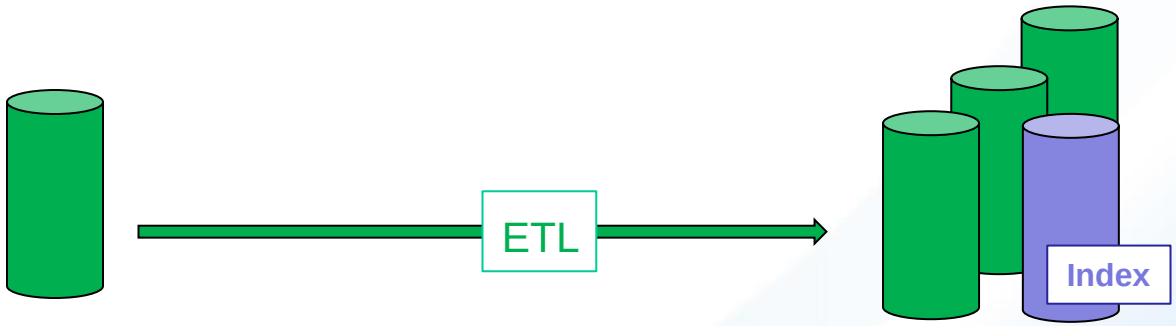
Heterogeneous Array

Solution: EPICS 4 PVData-based dynamic structure of self-described members

HDF5 Version

- **Data Access**: EPICS V4 RPC
- **Engine**:
  - Lock-free everywhere => 0.3 – 0.5 M scalar samples/s
  - Generic Type catalog
  - Generic TS Chunk
- **Backend:** Index + HDF5 files => 3 M scalar samples/s (HDD)

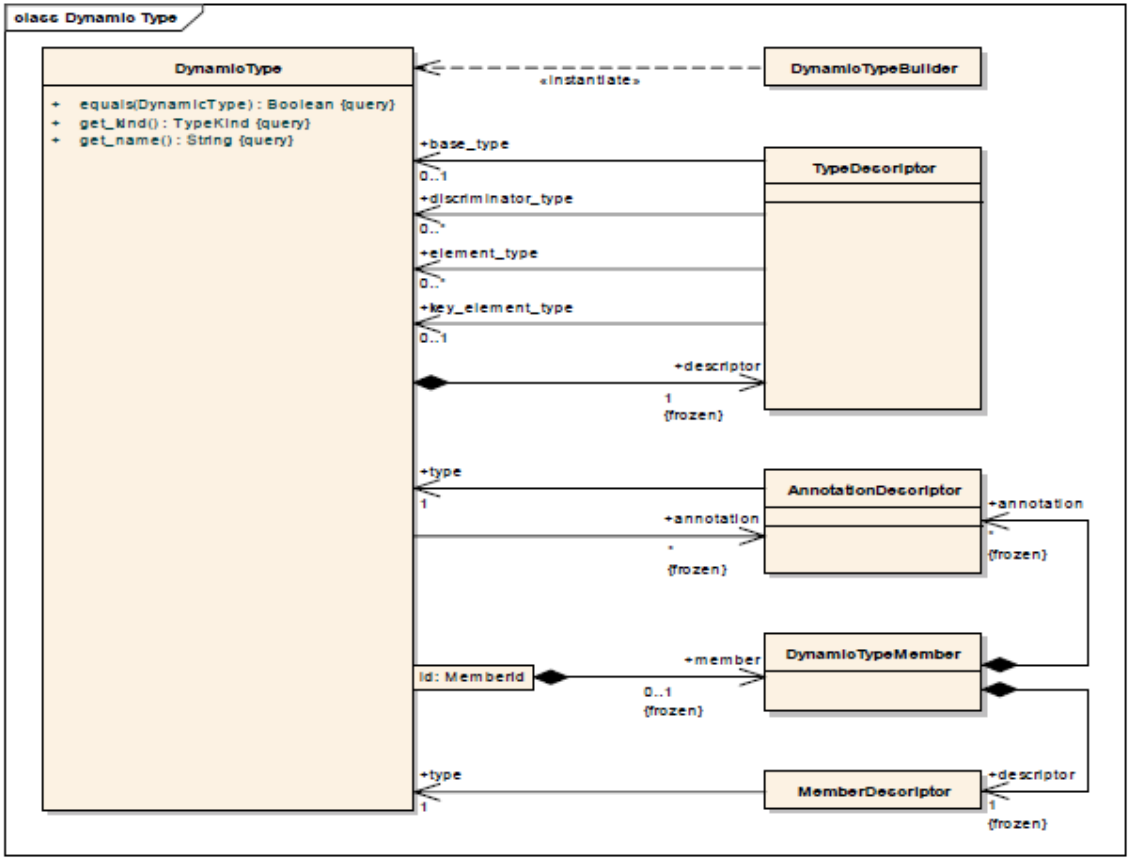
- Extract, Transform, Load (ETL):



Hard drive: SSD
HDF5 chunk: one TS Chunk

Hard drive: HDD
HDF5 chunk: many TS Chunks

Open Type System



Structured Data Types

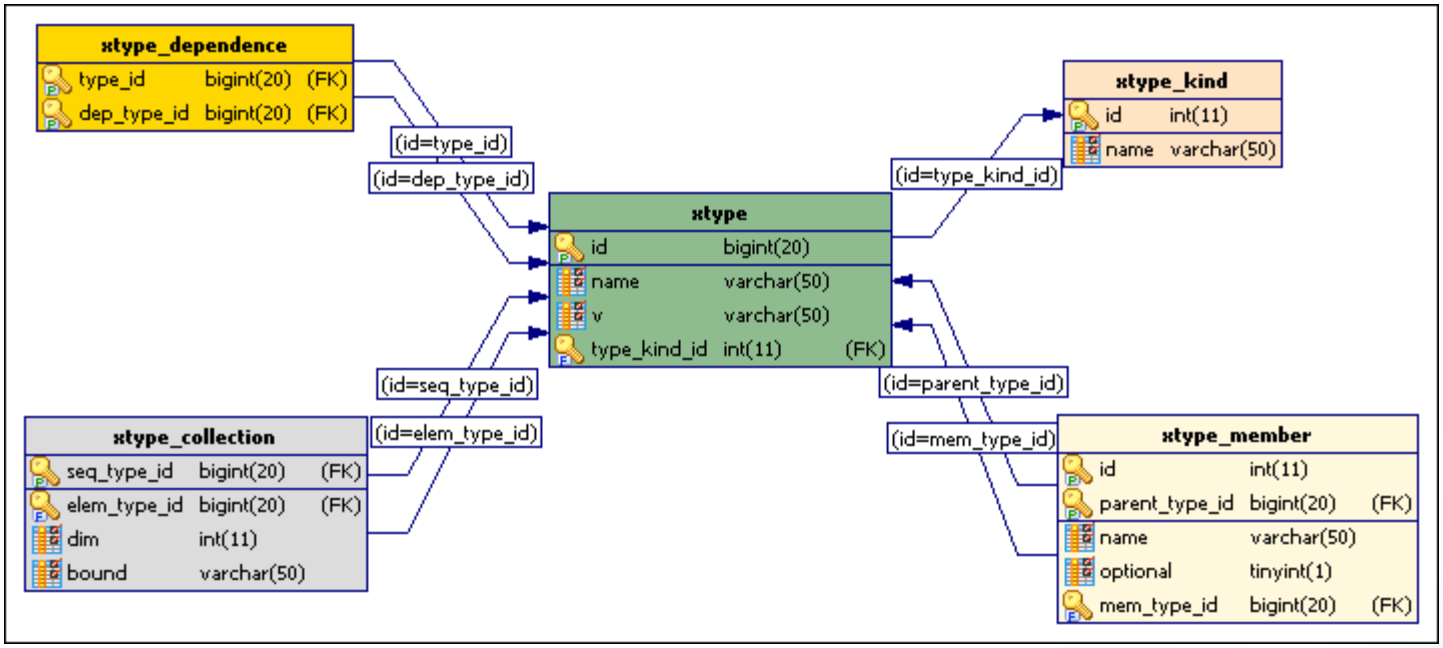
- DBR Types
- PV Data
- HDF5
- SciDB
- …

OMG DDS Extensible and Dynamic Topic Types

Version 1.0: November 2012

RDB-based Type Catalog

with D. Dohan

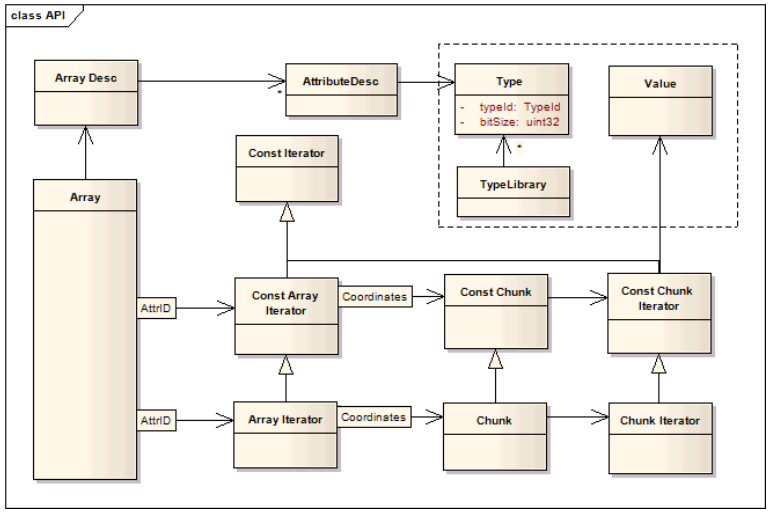


- **xtype_kind:** enumeration values of int16, float, string, sequence, structure, etc.
- **xtype:** collection of all type ids and versions
- **xtype_dependence***:* auxiliary table with the type dependencies
- **xtype_member:** structure-member associations
- **xtype_collection:** sequence-element associations

Chunk-Based Model

SciDB

Array – Chunk – Sample



Channel is a sequence of the TS chunks

TS Sample:

TSSample

Generic API

Type

*char**

Memory Buffer

TS Chunk:

Generic API

Type

HDF5 Backend

The HDF Group

HDF5 Conceptual Data Model:

- **group:** a folder of named objects, such as groups, datasets, and data types
- **dataset:** a chunk-based multidimensional array of data elements with attributes, data space and data types
- **datatype:** a description of a specific class of data element

Channel is a HDF5 group including the following datasets:

Channel Datasets	Datatype	Attributes	DBR Use Case	Original Channel Archiver
Intervals [chunk-based collection]	• timestamps • index of the first sample • number of samples		Intervals associated with the index file	File offsets and buffer sizes of Data Headers
Info [one element]	Channel-specific	Type Name	CtrlInfo	CtrlInfo's of Data Headers
Data [chunk-based collection]	Channel-specific	Type Name	One of the DBR scalar or waveform types	DBR type and count of Data Headers + Buffers

Next Use Case: Time Series of Frames

The proposed generic structure of the sparse multi-dimensional array is defined after the "natural" experiment-oriented representation built from the combination of two datasets: time series of detector-specific frames and time series of the frame positions in the scan-specific multi-dimensional space (angle, energy, pressure, etc)

- mapped into EPICS 4 PVData and HDF5 representations
- consistent with all (22) NeXus Application Definitions