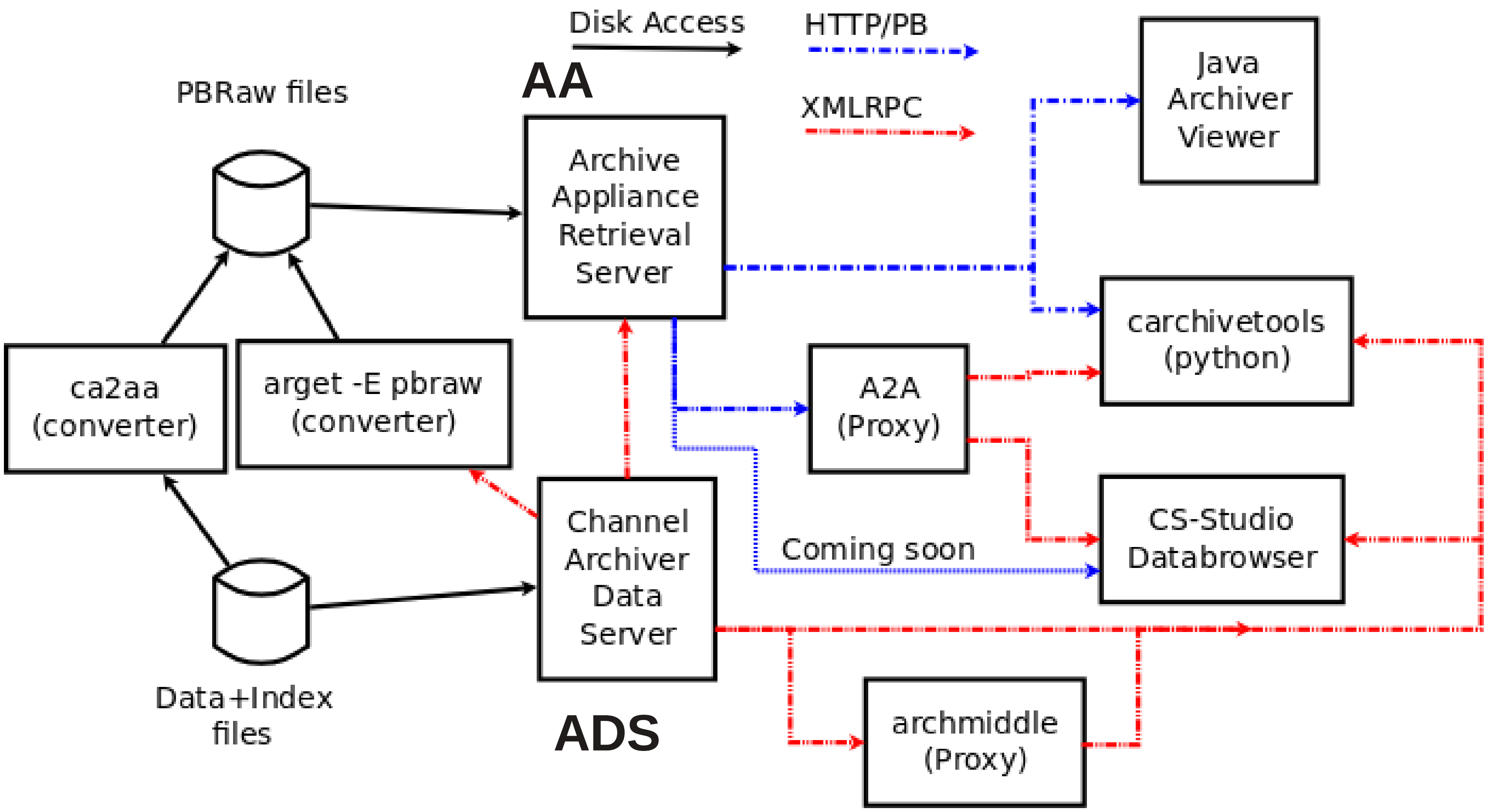
Archiver Appliance at NSLS2

Michael Davidsaver
BNL NSLS2

AA Ecosystem



AA @ NSLS2

- Beginning in Jan. 2014
- Learned how to install/configure AA server
- Performance/Robustness testing
- Development of client tools
  - Python client library
  - XMLRPC proxy for legacy clients
- Data file converter (work in progress)

Problems encountered (now fixed)

- SLAC specific PV name rules
- STS must be in RAM
- DBE_VALUE vs. DBE_ARCHIVE
- Invalid HTTP response headers
- Resource leak handling policies.py
- Re-discovered Java bug #6693490
- CAJ searching bug (fixed in 1.1.5)
- Start archiving ~50k PVs at once
  - Deadlock when internal message queue filled
- DoS
- Misc. Java exception messages

Open ~~issue~~ opportunity

- Binning operator inconsistencies
- AA provides operators for server side data reduction
  - maxSample_60(my:pv:name)
- One HTTP request per PV per operator
- Problem with first/last bins
  - max/min return a sample, first/last may not
  - Result lengths are different
  - Combining is difficult

Workaround

- Define one operator to give plot-able data
  - caplotbinning_60(my:pv)
  - Emulates ArchiveDataServer how=3
- Works for display clients (ie. DataBrowser)
  - The 90% case

Storage Performance

- NSLS2 Accelerator
  - Storing ~58k Pvs
  - ~5.5k events/sec
  - ~10 GB/day (avg. ~130 KB/s)
  - Load is not challenging for Channel Archiver ArchiveEngine or Archiver Appliance

Retrieval Performance

- ArchiveDataServer vs. Archiver Appliance
  - carchivetools (python client)
- Test case
  - 51 PVs
  - ~5 second update period (0.2 Hz)
  - 24 hour interval
  - Raw data to disk (HDF5)
- Result size ~17 MB uncompressed
- ADS – 158 sec
- AA – 2.5 sec (~6.8 MB/s)
  - 3.8 sec when writing compressed (6.7 MB)

Retrieval Performance (2)

- Increase interval to 30 days
  - Client store in memory (not disk)
  - Jan. → Feb. 2015 (4 months in the past)
  - 28,646,587 total events (dbr_time_double)
  - ~540 MB (~20 B/event)
- AA
  - Cold cache 18 sec. (~30 MB/s)
  - Hot cache 12 sec. (~45 MB/s)

“Snapshot” Performance

- Fetch a single time
  - 260 PVs (eBPM X position)
  - At most 2 samples per PV
- AA
  - Cold cache (~10 sec)
  - Hot cache (~1 sec)
- ADS
  - ~6 sec

Performance Results

- Bulk retrieval
  - AA retrieval throughput is far fast than ADS
  - XML vs. binary
  - RPC vs. streaming (pipelining)
  - Limiting factor is mainly client (python), and disk I/O speeds
- “Snapshot” retrieval
  - Performance similar
  - AA a little slower due to lack in indexing?

carchivetools

- Python client for ADS and AA
  - https://github.com/epicsdeb/carchivetools
- CLI tools (arget and argrep)
- API “from carchive.untwisted import *”
- a2aproxy (XMLRPC to HTTP/PB proxy)
- archmiddle (XMLRPC to XMLRPC proxy)
  - rewrites queries to simpify client config.

carchivetools examples

CLI

```
$ argrep --wildcard 'LTB*ICT*Q-I'
LTB-BI{ICT:1}Q-I
…
$ argrep --regexp 'LTB.*ICT.*Q-I'
LTB-BI{ICT:1}Q-I
...
```

Python

```
>>> arsearch(['LTB*ICT*Q-I'], match=WILDCARD)
set(['LTB-BI{ICT:1}Q-I',
 ...)
>>> arsearch(['LTB.*ICT.*Q-I'], match=REGEXP)
set(['LTB-BI{ICT:1}Q-I',
 ...)
```

```
$ arget --start '-1 min' --wildcard 'LTB*ICT*Q-I'
LTB-BI{ICT:1}Q-I
2015-03-18 18:51:44.711198 -0.00343209054212
...
Found 45 points
...
```

```
>>> retval=arget(['LTB*ICT*Q-I'], start='-1 m', \
                     match=WILDCARD)
>>> for name,(meta,val) in retval.items():
      ...
```

```
$ arget –skip-first -s '1:30' -e '1 m' -W 'LN*Sts'
...
```

File Converters

- arget -E pbraw …
  - Query ADS, write AA files
  - Jaka Bobnar and Ambroz Bizjak (Cosylab/FRIB)
- ca2aa
  - Direct conversion from Channel Archiver data files to Archiver Appliance file format.
  - MD and JB
  - Work in progress
  - https://github.com/mdavidsaver/ca2aa

Conclusion

- NSLS2 accelerator switching to Archiver Appliance
- Some teething problems, now a stable system
  - Only retrieval/configuration problems.
  - **No storage corruption found!**
- More work to be done with retrieval
- Client tools built and tested
  - Migration path developed