Database Applications

Igor Mandrichenko

FIFE Workshop

20th-21st June 2016

Scientific Databases

- We support and develop a variety of database products
  - Conditions databases (NOvA, Minerva, MicroBooNe, DUNE, LArIAT, …)
  - IFBeam Database (all IF experiemnts)
  - Hardware Databases (NOvE, Mu2e)
  - Constants, Telemetry, Alarms, Exposures (DES, DESI)
  - Query Engine (NOvA, DUNE, MicroBooNe)

- Support for most of these is 8x5. IFBeam data taking components are supported 24x7 with 12 hours response time

Service Catalog

Service Desk

Monday - Friday, 8-5

https://fermi.service-now.com
or 630.840.2345 or walk-ups (WHGF)

Service Offerings:

Conditions Database Conditions Database provides users access to the experiment constants information through web services to retrieve and upload data.

DES Alarms Viewer DES Alarm Viewer provides DES users access to the experiment information for alarms levels from various devices through a web-based application.

DES Constants Database DES Constants Database provides DES users access to the experiment information for constants through a web-based application.

DES Exposures Viewer DES Exposures Viewer provides DES users access to the experiment information stored in the exposure table in the PostgreSQL database through a web page, which uses QEngine-base reporting technique to query the database.

DES Telemetry Viewer DES Telemetry Viewer provides DES users access to the experiment camera information through a web-based application.

Hardware Database Hardware Database provides users access to the experiment hardware information through a web-based application. Data is stored in a PostgreSQL database and can be retrieved and uploaded by selected users.

IFBeam Conditions Database (IFBeamDB) IFBeamDB is a collection of applications designed to collect, preserve, monitor and make available beam conditions data for the Intensity Frontier experiments at FNAL. The database exists to provide off-line data access to the data with a long latency, guaranteed. It also provides real time monitoring information with a short latency, but this is not guaranteed. This database does not provide on-line data acquisition functionality.

IFBeam Conditions Database (IFBeamDB) (Enhanced) IFBeamDB is a collection of applications designed to collect, preserve, monitor and make available beam conditions data for the Intensity Frontier experiments at FNAL. During the time sensitive period of data taking, enhanced response times are offered.

Query Engine Query Engine provides read only web service for a simple single table query from a PostgreSQL database.

Conditions Databases

- Generally, a representation of a tuple as a function of ”event” or “validity” time

  (x, y, z) = f(Tv)[channel]

  - what were calibration constants on April 1st 2014 at 9:01:02 ?
- User defined data schema (x,y,z)
- Version control: on June 1st 2016 we reprocessed our data and we need to update our calibrations from March 15 2015 to now
  - The user can go back to ”old” version of the calibration data
    - By specifying time to roll back to
    - By tagging the database state with a text tag
- Python API, C/C++, web interface, data browser, GUI

Conditions Databases: users

- "Minerva" style
  - Historically first conditions DB developed to replace COOL
  - Simple data model
    - All channels are measured at the same time
  - Used by Minerva and MicroBooNe

- ConDB
  - Developed for NOvA
  - Potentially every channel has its own time line
  - (Lossy) data compression
    - Do not record new values as long as they are close enough to the old ones
  - NOvA, DUNE, LArIAT

Unstructured Conditions Database (uConDB)

- New development

- Structured databases mentioned earlier:
  - database stores arrays of tuples, indexed by Tv and channel number:

    ```
    (x,y,z) = f(Tv) [channel]
    ```

- Unstructured conditions database:
  - uConDB records the history of the named opaque object changes
  - database is unaware of the structure of the object:

    ```
    {object} = f(name, Tv)
    ```

  - Object can be anything: XML, JSON document, config. file, image, event, …

uConDB features

- Same version control features
- Object version is associated with the validity time
- Folder provides the namespace for the object names

- Architecture - 2 databases:
  - Metadata database – relational, Postgres
  - Object storage backend with simple key-value interface
    - Postgres and CouchBase backends are available

Some uConDB use cases

- Data with complex, or fuzzy, or dynamic schema
- Storage for large number of small files
- Can be used as a storage for named objects with version control
  - Just set Tv=0

- Distributed redundant object storage backend (such as CouchBase) provides
  - Scalability
  - Elasticity
  - High data availability

Query Engine

Simple generic web service for simple relational data queries

Represents a simple database as a web service:

Instead

```
select a, b, c from table where x > 3 order by n
```

Do this:

```
http://…/query?t=table&c=a,b,c&w=x:gt:3&o:n
```

Replies with CSV file

Works with wide range of single-table (or a view) query

- Sorry, no subqueries, group-by's, etc. (yet)

IFBeam Database

- Supported by Database Applications group
- Database holds beam conditions data collected in real time from AD
- Data latency is seconds for real-time database and minutes for the long-term database
- Data is being collected from multiple beam lines
  - NuMI - 230GB/quarter
  - BNB - 430GB/quarter
  - Test Beam
- No significant increase in the data rate is expected from BNB in near future
  - But potentially can go much higher, about 5 times, in longer term
- Long term database is believed to be able to ingest data at about 8x higher rate

IFBeam Database

- Total size of the database now is at 4TB, collected since August 2011
- Current disk capacity is 24TB after mirroring
- Database is replicated on 3 servers. All 3 are mirrored
- Data collection by 3 redundant computers with individual local disk for data buffering in case of the database outage

- Existing resources believed to be adequate to sustain data access demand and data inflow for the near future

- 8x5 support, except for data collection – 24x7 with 12 hours response time

- Data access only via web service, Art module

Database Access via Web Service

- All the applications mentioned are web services with very little, if any, direct access to the database

- Very old, general idea:
  - Instead of talking to the database directly:

    select x,y,z from runs where run_number = 123;
  - Build web service:

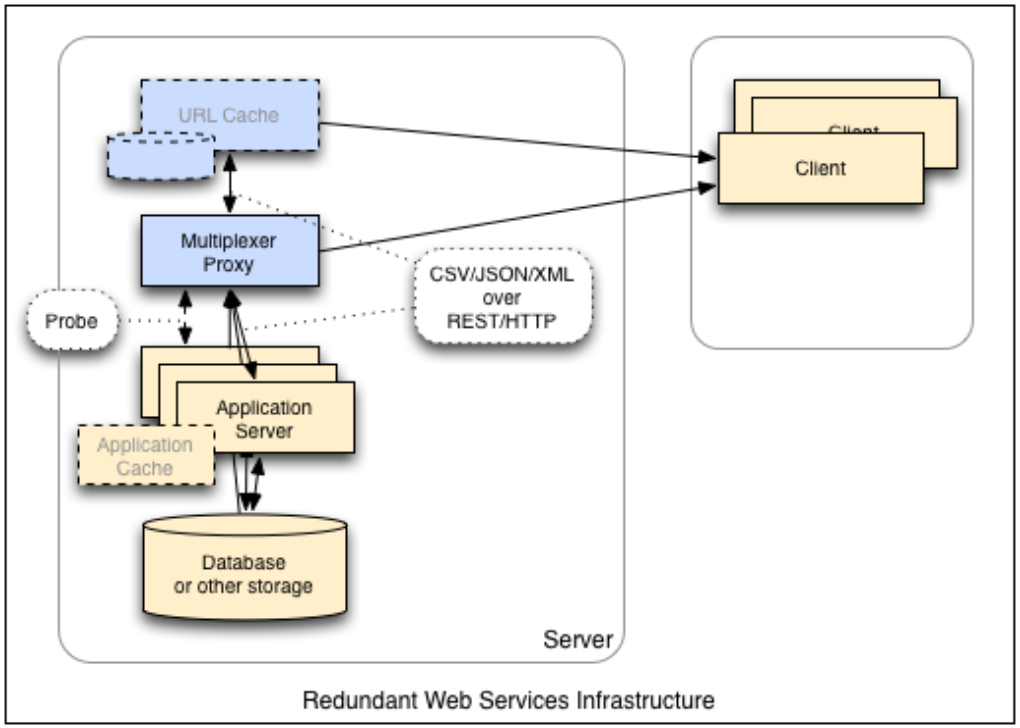
    HTTP GET http://server.fnal.gov/MyRunsDatabase/get_xyz?run=123

- Benefits:
  - The database implementation knowledge moved from the client to the server. The client can operate in the application specific terms
  - Data access throttling, staggering – avoid the database overloading
  - Resource management in the multi-user competing environment
  - Using common, well developed and supported Internet industry standards, tools, protocols, frameworks for data delivery, caching

Web interface support

- Standard W3C/IETF HTTP
  - Common industry tools, frameworks, libraries can be used
- Client side
  - Standard Python urllib2 works fine
  - Thin libWDA C library, based on libcurl for C/C++ access
    - CSV parsing
    - Intelligent (exponential random) retries
- HTTP data uploads
  - Authenticated using signature/shared secret method, similar to Kerberos
  - The secret is never sent over the network
  - RFC 2617 support added to uConDB

Redundant Web Services Infrastructure



http://dbdata0vm.fnal.gov:8080/index

- Currently runs on 13 redundant application server computers
  - 4 real computers
  - 9 virtual computers, one in HA CCD VM cluster
- Handles a dozen applications for multiple experiments
- On average, ~100 requests/second, up to 100 MB/sec
- High availability
- Data caching
- Scalable data access, throttling, staggering mechanism
- Resource management
- Monitoring
- Can run any WSGI application, other HTTP-based standards

Future Development

- Conditional Databases
  - Some GUI improvements
  - Data browser
    - Exists for ConDB
    - Need one for uConDB, “Minerva” style
  - File system backend for uConDB
    - Should be trivial
- Non-SQL databases and mixed architectures:
  - RDB for metadata and NoSQL for actual data
    - Fast indexing, lookup using RDB
    - Expandable storage for actual data