DATABASE APPLICATIONS: BEST PRACTICES

Igor Mandrichenko

FIFE Workshop

6/1/2015

Portfolio

- Collaboration tools
  - Logbook
  - Shift Scheduler
  - Project Organizer
  - Speaker’s Bureau
  - DES collaboration management databases

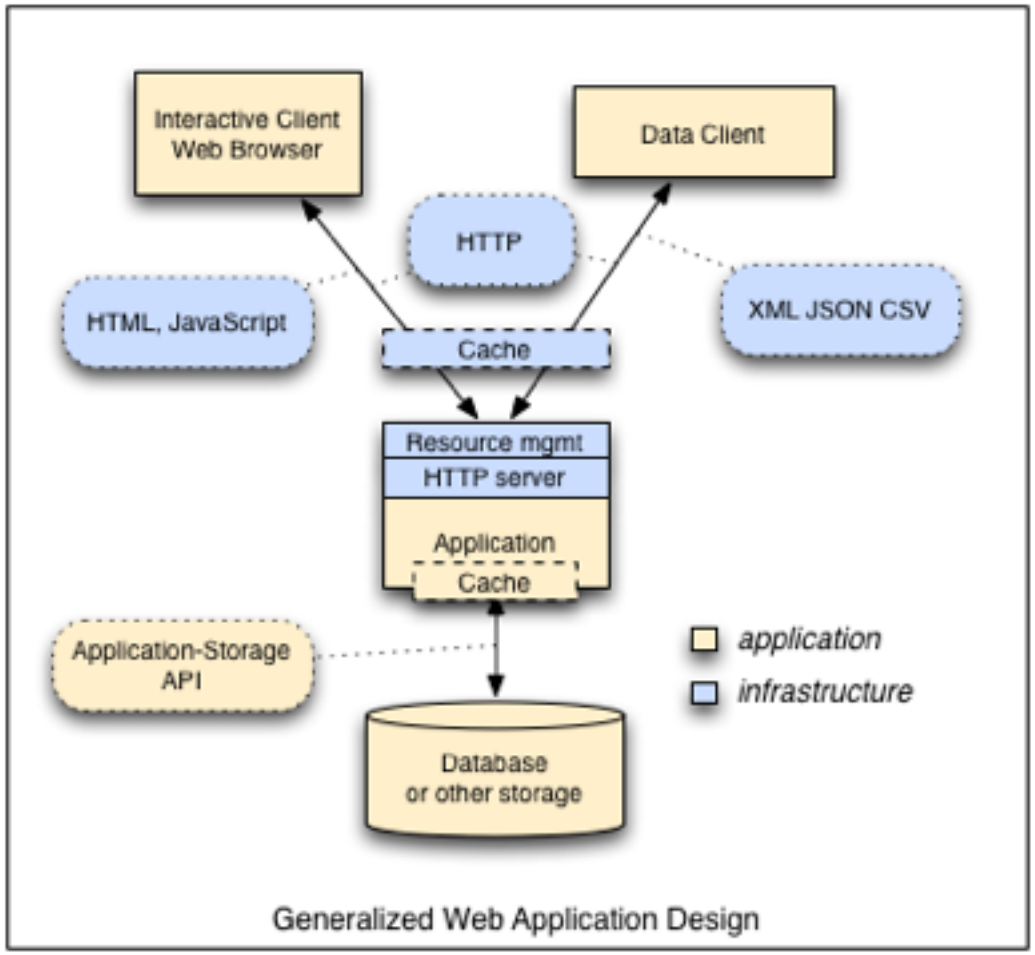
Portfolio

- IFBeam database
  - Real time
  - High availability
  - Highly redundant (main DB server with 2 replicas, mirrored disks on each - 6 copies of data)
  - BluArc backups
- Conditions databases
  - 2 flavors - simple (Minerva, MicroBooNe) and more sophisticated (NOvA, DUNE)

Data access from the grid

- Direct access to databases from the grid does not scale
  - Database servers do not have capability to handle and manage high unorganized load created by the grid clients
- Exposing database schema to the client is not always necessary and usually is a bad idea
- Very old solution: put a web server between the client and the database
  - Instead of:
    - select data from table1, table2 where .. and t=123 ..
  - web method:
    - http://server.fnal.gov/application/get_data?t=123

Common application architecture



Benefits of web sever vs. database

- Abstraction
  - Representation translation is done on the server
  - Server hides details of actual database data representation from the client
  - Schema flexibility
  - Server implementation can be changed without affecting the client - as long as the interface stays the same
    - Change of the underlying database product version or even flavor
- Web protocol is connection- and state-less
  - Higher flexibility in resource management
- Security
  - Read-only web server is much safer and often does not need any security restrictions

Benefits of hiding the database behind a (redundant) web sever

- Load balancing
- Resource management
  - Managing data servers is much easier than managing clients
- Elasticity
  - Add or remove underlying resources to meet the changes in the demand, priorities
- High availability
  - No downtime upgrades, including schema changes
    - Upgrade some servers while others are running, then switch
  - Hardware failures, OS upgrades, etc.

Redundant Web Services Infrastructure (RWSI)

- Server side
  - scalable, expandable (we use VMs), high availability environment to run web-based data access applications
  - powerful resource management and monitoring
- Client side
  - light weight HTTP data access C library
    - based on libcurl
  - sample Python code (trivial)
- Standard Internet protocols, formats, tools
  - HTTP, CSV, REST
  - Cache, Proxy

Common approach to application development

- Separate data representation functionality from the rest of the application into a web service
- Deploy the database (contact CCD)
- Use the Redundant Web Services Infrastructure to run one or more instances of the web service

- (Very different) applications with web data access
  - IFBeam DB - only web access
    - real time data and off-line
  - Conditions databases
  - Logbook - XML API (read, post)

Another good practice: caching

- Caching reduces load on the data server by remembering answers to common questions without necessarily understanding them

- Well developed by the Internet industry
- Standard tools
  - nginx, squid, etc.

- Caution:
  - caching works only when requests are repeated within sufficiently short time (cache lifetime should be longer)
  - cache will remember old data even after the database is updated (cache lifetime should be shorter)

Caching

- How to make caching work ?
- Use case: time dependent data (conditions data)
- Clients ask for data in time intervals
  - get me data for times between t0 and t1
  - the requests will never repeat if (t0,t1) pairs do not repeat

Caching

- Solution: Round t0 down and t1 up to some “round” time points, say to the nearest midnight, hour, run boundary, 15 minutes, etc.
- Example:
  - instead of data?t0=12:05:17&t1=12:25:05
  - do this: data?t0=12:00:00&t1=13:00:00
- Different requests become identical, cacheable
  - data?t0=12:25:06&t1=12:47:02
  - becomes data?t0=12:00:00&t1=13:00:00
- Careful: clients get more data than they asked for, more data is sent over the network

Caching

- Client side caching
- If practical, combine many “small” requests into a bigger one
- Instead of this:
  - getData(run=100,subrun=1),
    getData(run=100,subrun=2),
    getData(run=100,subrun=3)
- Do this:
  - getData(run=100)
  - Then extract subruns

Caching

- Server side caching
- If URL-based web (application independent) caching is impossible or impractical, use caching on the server
  - Since data server translates DB representation to the application representation, there may be some caching opportunities there

What if I need a database application ?

- When designing a database, come talk to us
  - We may have done this before
  - We may already have the application for you
    - Example: we have conditions databases for NOvA, Minerva, giving them as is to MicroBooNe, DUNE (LBNE)
- If you need access to IFBeam DB or need monitoring tool based on it, we can help
- If you have a web application to deploy, we can help you set up your own redundant web services environment or run it for you