Object Databases as Data Stores for HEP

Dirk Düllmann
CERN, IT/ASD & RD45

`Dirk.Duellmann@cern.ch`

The RD45 Project

- Started in 1995 to find solutions to problems of **LHC data management**
  - Enormous data volumes & rates, project lifetime
- Proposed solutions being adopted by **current** experiments
  - **Objectivity/DB** & HPSS
- In 1999, several experiments will start storing ~200-400TB/year at upto 35MB/second
- Similar problem to LHC, but **today!**

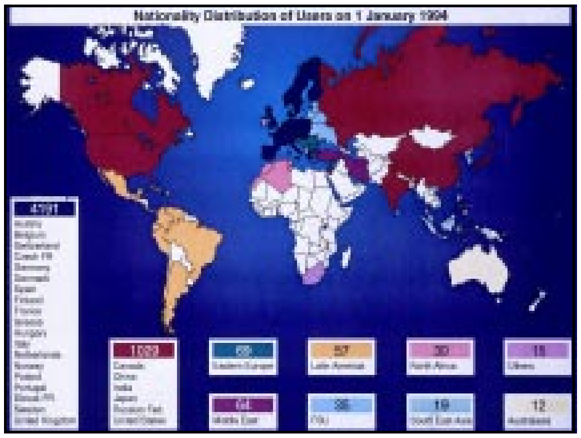
High Energy Physics

- Study of **fundamental constituents** of matter and **forces** between them
  - quarks, electrons, neutrinos, photon, Z^0, $W^\pm$, etc.
  - Higher and higher energies are required to delve deeper and deeper into matter
- i.e. larger, more powerful, more expensive
  - No longer affordable by individual countries

CERN

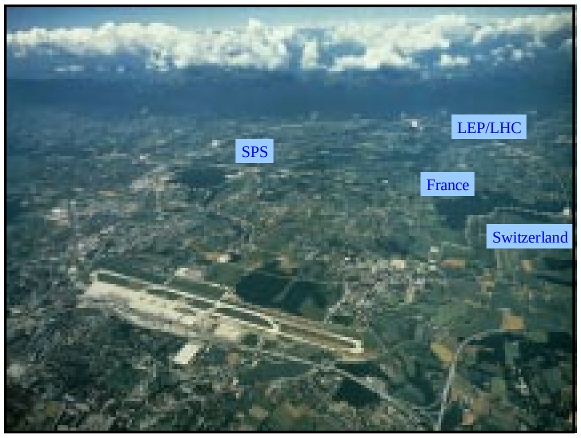
- Conseil Europeen pour la Recherche Nucleaire
- Established in 1954
  - **"kickstart"** research in Europe
- Currently funded by 19 European Countries
- Next main research program:

  Large **H**adron **C**ollider
- Startup: **2005**, data taking **~20** years



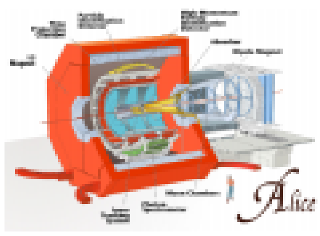
LEP Characteristics

- Circumference of the accelerator: 26.659 km
- Depth of the tunnel housing LEP: -50 to -175 m
- Alignment accuracy : 0.1 mm
- Speed of e^+e^- : 0.035 km/h below light speed
  - over 11200 cycles / second
- LEP can detect:
  - the orbit of the moon,
  - heavy rainfall,
  - changing water levels in Lake Geneva,
  - the Geneva-Paris TGV.
- LHC will collide protons in the same tunnel



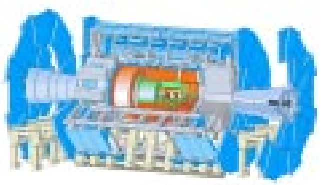
ALICE

- Heavy ion experiment at LHC
- Studying ultra-relativistic nuclear collisions
- Relatively short running period
  - online 1 month/year
  - 1 PB/month
- Extremely high data rates
  - 1.5GB/s



ATLAS

- General-purpose LHC experiment
- High Data rates:
  - 100MB/second
- High Data volume
  - 1PB/year
- Test beam projects using Objectivity/DB in preparation:
  - Calibration database
  - Expect 600GB raw and analysis data

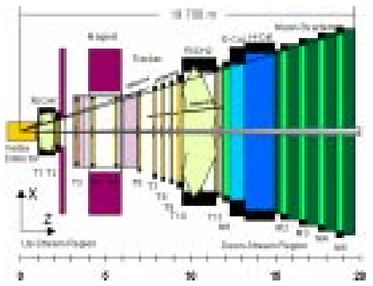


CMS

- General-purpose LHC experiment
- Data rates of 100MB/second
- Data volume of 1PB/year
- Two test beams projects based on Objectivity successfully completed.
- Database used in the complete chain: Test beam DAQ, Re-construction and Analysis

LHCb

- Dedicated experiment looking for CP-violation in the B-meson system.
- Lower data rates than other LHC experiments.
- Total data volume around 400TB/year.



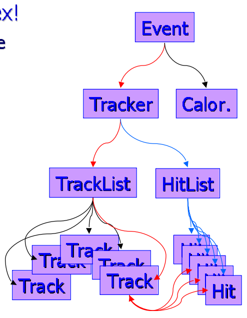
The Large Hadron Collider

- Currently under construction: data ~2005
- 4 large "experiments", run for ~20 years
  - ~2000 people, ~200 institutes, ~20 countries

Time interval	ATLAS/CMS	ALICE
1 second	100 MB	1.6 GB
1 minute	6 GB	100 GB
1 hour	360 GB	6 TB
1 day	8.6 TB	140 TB
1 week	60 TB	1 PB
1 month	260 TB	
1 year (100 days)	**1 PB**	
Total LHC (20 yrs)	**100 PB**	

HEP Data Models

- HEP data models are complex!
  - Typically hundreds of structure types (classes)
  - Many relations between them
  - Different access patterns
- LHC experiments rely on OO technology
  - OO applications deal with networks of objects
  - Pointers (or references) are used to describe relations



Data Management at LHC

- LHC experiments will store huge data amounts
  - 1 PB of data per experiment and year
  - 100 PB over the whole lifetime
- Distributed, heterogeneous environment
  - Some 100 institutes distributed world-wide
  - (Nearly) any available hardware platform
  - Data at regional-centers?
- Existing solutions do not scale
  - Solution suggested by RD45: ODBMS coupled to a Mass Storage System
    - e.g. Objectivity/DB coupled to HPSS

Features of HEP Data

☺Data is essentially read-only
☺Small number of concurrent users (1-100)
☺Very low transaction rate
☺Can accept data loss(!)

💣Data volumes and rates (up to 100PB, 1.5GB/s)
💣Fully distributed
💣Project Lifetime (~25 years)

Physics Data Processing Tasks

- **Reconstruction:**
  - support multiple writers (**100-500** filter farm cpu's)
  - support aggregate data rates sufficient to keep up with acquisition
  - **100MB/s** for ATLAS/CMS; **1.5GB/s** for ALICE
- **Analysis:**
  - support multiple readers (~**150** "effective" users)
  - data volumes and rates hard to predict at this time
  - assume aggregate data rates ≤ reconstruction
- **Simulation:**
  - Very CPU intensive process
  - No "special" requirements
  - Sufficient to satisfy requirements of reconstruction and analysis

Reconstruction

- During past 3 years, NA45 have performed several **production** reconstruction runs using Objectivity/DB
  - initially on the Meiko CS-2 (HyperSparc SMP)
  - ported to NT, writing with 16MB/sec
- New **production** schedule now
  - reprocessing of existing data
  - DAQ tests with new TPC and 16MB/sec Rate
- **Production** demonstration of parallel writing into ODBMS
- **Production** demonstrations also made in CMS Test Beams
- Data Volumes in **10-100GB** range

The RD45 Collaboration

- ANL: D. Malon
- BNL: M. Purschke
- Caltech: J. Bunn, H. Newman, R. Wilkinson
- DESY: M. Gasthuber
- U. Heidelberg: A. Pfeiffer
- LBNL: D. Quarrie
- KEK: Y. Morita
- Krakow: S. Jagielski
- MIT: A. Klimentov
- Orsay: C. Arnault, A. Schaffer
- SLAC: J. Becla, G. Cosmo
- TATA: S. Banerjee
- Tufts: S. Rolli, K. Sliwa
- ETHZ: A. Hasan
- U. Venice: E. Cargnel
- CERN:
  E. Arderiu Ribera
  P. Binko
  J. Conde
  D. Düllmann
  B. Ferrero Merlino
  G. Folger
  K. Holtman
  V. Innocente
  R. Knap
  M. Nowak
  J. Shiers
  L. Silvestris
  L. Tuura
  I. Willers

Project Stages

❶ Develop complete list of requirements, evaluate existing packages, limited prototyping (~6 months)
  - March - November 1995
  - Interim Review November 1995

❷ Detailed prototyping and evaluation, prototype applications, comparisons with established metrics etc.
  - November 1995 - April 1998
  - Reviews in Spring 1996, 1997 & 1998

❸ Implementation phase
  - May 1998+
  - Production Services, Outstanding R&D Issues

Why an ODBMS?

- Key requirements could not be met by language extensions, light-weight object managers
- Strong requirement for consistent interface
  - "The ODMG language bindings are based on one fundamental principle: the programmer should perceive the binding as a single language for expressing both database and programming operations, not two separate languages with arbitrary boundaries between them."

☺ Use of an Objectivity/DB federation met our main requirements
  - Enhancements required but (apparently) sufficient flexibility in architecture to accommodate these

Object Persistency

- Persistency
  - Objects retain their state between two program contexts
- Storage entity is a complete object
  - State of all data members
  - Object class
- OO Language Support
  - Abstraction
  - Inheritance
  - Polymorphism
  - Parameterised Types (Templates)

OO Language Binding

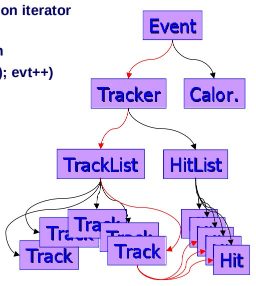
- User has to deal with copying between program and I/O representations of the same data
  - User has to traverse the in-memory structure
  - User has to write and maintain specialised code for I/O of each new class/structure type
- Tight Language Binding
  - ODBMS allow to use persistent objects directly as variables of the OO language
  - C++, Java and Smalltalk (heterogeneity)
- I/O on demand
  - No explicit store & retrieve calls

A Code Example

```
Collection<Event> events;          // an event collection
Collection<Event>::iterator evt;  // a collection iterator

// loop over all events in the input collection
for(evt = events.begin();  evt != events.end(); evt++)
    {
    // access the first track in the tracklist
    d_Ref<Track> aTrack;
    aTrack  = evt->tracker->trackList[0];

    // print the charge of all its hits
    for (int i = 0; i < aTrack->hits.size(); i++)
          cout  << aTrack->hits[i]->charge
                  << endl;
    }
```

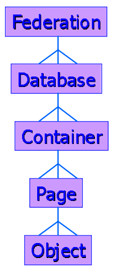


Concurrent Access

- Support for multiple concurrent writers
  - e.g. Multiple parallel data streams
  - e.g. Filter or reconstruction farms
  - e.g. Distributed simulation
- Data changes are part of a Transaction
  - ACID: Atomic, Consistent, Isolated, Durable
- Access is co-ordinated by a lock server
  - MROW: Multiple Reader, One Writer per container (Objectivity/DB)

Objectivity/DB Architecture

- Architectural Limitations: OID size 8 bytes
- 64K databases
- 32K containers per database
- 64K logical pages per container
  - 4GB containers for 64kB page size
  - 0.5GB containers for 8kB page size
- 64K object slots per page
- Theoretical limit: 10 000PB
  - assuming database files of 128TB
- RD45 model assumes 6.5PB
  - assuming database files of 100GB
  - extension or re-mapping of OID have been requested



Scalability Tests

- Created federations of **1TB**
  - limited by available disk space ($$$)
  - A 1TB federation requires only **40** DBs of **25GB** each!
- Created federations up to limit of **2^{16}** databases
- Tested individual databases up to **25GB**
  - database size “limited” by filesystem
  - practical limits (network, staging) suggest 1-10GB maximum size
- Numerous production federations of **10-100GB**
  - > 1k databases
- Plans for 1999 include several federations of **100TB+**
- Enhancement requests for scalability up to 100PB+
  - multi-file databases

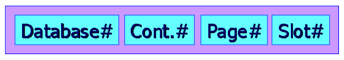
Scalability Tests II

- LHC Experiments predict some **150** “effective simultaneous interactive users”
  - e.g. **20** analysis groups, each with **10-30** members
- Can an ODBMS support ~**150** concurrent analyses?
- A simulation was performed on the Caltech Exemplar
  - 1/3 clients read objects **10KB** in size; compute **0.001s** per object
  - 1/3 clients read objects **100KB** in size; compute **0.1s** per object
  - 1/3 clients read objects **500KB** in size; compute **10s** per object
- Simulation showed no degradation in performance with up to **150** concurrent analyses

☺ **150** effective simultaneous users achievable today

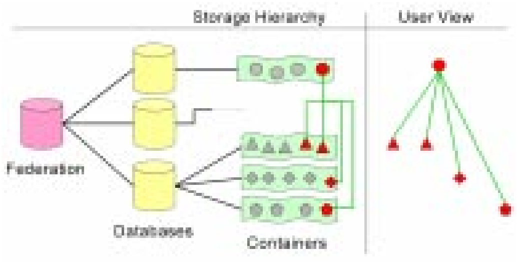
- *Without* using DB partitions, replication etc.

Navigational Access



- Unique Object Identifier (OID) per object
  - Direct access to any object in the distributed store
  - Natural extension of the pointer concept
- OIDs allow to implement networks of persistent objects (“associations”)
  - Cardinality: 1:1 , 1:n, n:m
  - uni- or bi-directional (referential integrity!)
- Transparent use of OIDs through “smart-pointers”

Physical Model and Logical Model



- Physical model may be changed to optimise performance
- Existing applications continue to work

Page Server & Container Locking

- Objectivity/DB
  - Page exchange between client and server
    - Page does contain not only requested data
    - In case of good clustering, it contains other objects that will be requested soon
  - Server only "knows" about I/O pages
    - Thin server, fat client
    - Improved scalability
  - Locking on container level
    - All objects in one container are locked at once
    - Improved scalability and performance

Object Clustering

- Goal: Transfer only "useful" data
  - from disk (page) server to client cache
  - from tape to disk
- Physically cluster objects according to main access patterns
  - Clustering by type
    - e.g. Track objects are "always" accessed with their Hits
- Main access patterns may change over time
  - Performance may profit from re-clustering
  - Clustering of individual objects
    - e.g. All Higgs events should reside in one file
  - first studies on automatic reclustering

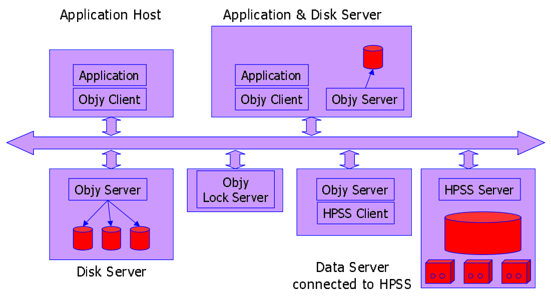
Clustering on a Larger Scale

- Objectivity limits containers to 64k logical pages
  - about 0.5 GB for 8kB page size

  Simple strategy:
  - check container size when a new object is created
  - create a new container if the current one approaches the limit
  - manage a ***persistent*** list of containers
- Objectivity locks on container level
  - Avoid lock contention in multi-processor environments

  Simple strategy:
  - assign one container per process
  - manage the list of containers as one logical container

A Distributed Federation

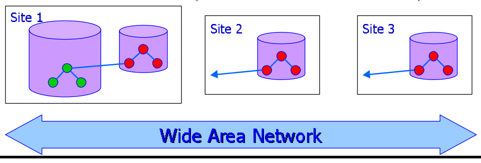


Mass Storage Interface

- Clear from early days of RD45 that an interface to an MSS was required
  - cannot assume that multi-PB disk farms will be affordable and/or manageable
- **HPSS** has emerged as the "MSS of choice" in the HEP community
- An evaluation of Objectivity/DB and HPSS architectures suggested that most appropriate interface would be between the Objectivity/DB server and HPSS client API
- A prototype of such an interface has been produced and is in test at Caltech, CERN and SLAC

Data Replication

- Objects in a replicated DB exists in all replicas
  - Multiple physical copies of the same object
  - Copies are kept in sync by the database
- Enhance performance
  - Clients access a local copy of the data
- Enhance availability
  - Disconnected sites may continue to work on a local replica



Why Replicate?

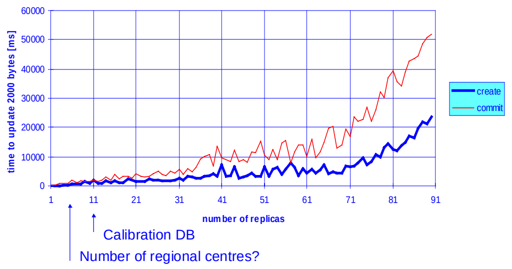
- Increased fault tolerance
  - critical resources are duplicated in more than one location
  - users can continue to work even if WAN is down or one "image" unavailable
- Increased performance
  - access to frequently used objects can be satisfied from a local replica, avoiding WAN network traffic

- Data Collection
  - e.g. "centralisation" of simulated events generated at outside institutes
- Data Distribution
  - tag objects, analysis objects, calibration & production data
- ❶ Calibration Data
  - typically "replicated" to ~10 outside institutes (cf HEPDB)
- ❷ Analysis Data
  - Use ODBMS to maintain coherent "copies" at regional centres

Replication Tests

- Previous tests of replication, based on a pre-release, were limited to NT and the LAN
- Have now extended these tests to NT(Alpha, Intel) + Unix (DEC, HP, IBM, SGI, Sun), large numbers of images (100), and the WAN (CERN-Caltech)
- Replication functions correctly in these environments
- Need appropriate network bandwidth
- Need to understand when replication is appropriate
  - calibration databases, tag databases, collections etc.
- Enhancement requests include "tape replication"
  - need also increased reliability & bug fixes (V5?)

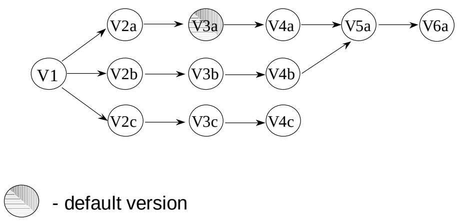
Replication With Many Images



Replication: Problems

- Main problem concerns Fault Tolerant Option (FTO)
  - ***All*** partitions must be online for catalog and/or schema changes
  - Rules out production use
  - Plans to use DRO quorum mechanism in V6(?)
- Outstanding enhancement request to permit "offline" (e.g. tape) replication
  - for bulk data transfer
- Outstanding choice of "best" remote image
  - e.g. images at srv1.cern.ch, srv1.fnal.gov
  - Where does an application on usr1.cern.ch read from?
  - What about usr1.anl.gov?

Object Versioning


$$\begin{tikzcd}
 & V2a \arrow[r] & V3a \arrow[r] & V4a \arrow[r] & V5a \arrow[r] & V6a \\
V1 \arrow[ur] \arrow[r] \arrow[dr] & V2b \arrow[r] & V3b \arrow[r] & V4b \arrow[ur] & & \\
 & V2c \arrow[r] & V3c \arrow[r] & V4c & &
\end{tikzcd}$$

▨ - default version

- Maintain multiple versions of a single "logical" object
- Example: Versions of calibration data in the BaBar calibration DB package

The ODMG Standard

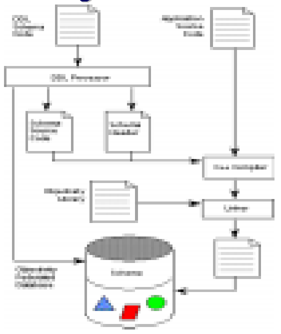
- Standardise on
  - Data definition language ODL
  - Data interchange format OIF
  - Language binding for C++, Java and SmallTalk
- Goal: Simplify code migration from one database vendor to another
  - Most frequently used API elements are covered by the ODMG 2.0 standard.
  - Performance and the feature set of different ODBMS products differs significantly !
- Any large scale migration will still require a significant effort!

DDL Restrictions

- Persistent classes may not:
  - contain other persistent classes as data members
    - They may contain references to other persistent class though
    - Late (multiple-) inheritance from d_Object helps to keep transient and persistent classes in sync
  - contain C++ pointers or references
    - Neither directly nor through embedded classes
    - replace C++ pointers by database smart pointers
- ❷ is the more intrusive change
  - **Type declarations** of pointers referencing persistent objects **have to be changed** for all clients of a persistent capable class.
  - Code that **uses** these variables stays untouched.

Schema Handling

- Definitions of persistent capable classes made in **.DDL** files
- ***ooddlx*** processor generates appropriate headers & source code
  - Schema is added to federated database
- Applications are built using generated files and the Objectivity library



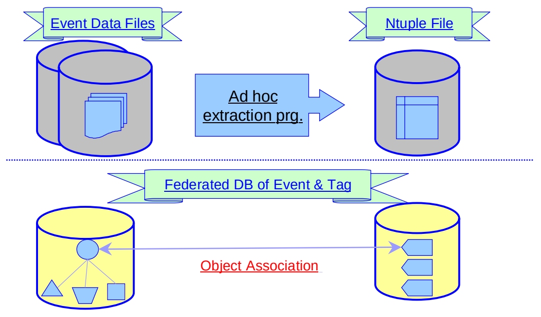
Schema Evolution

- Evolve the object model over the experiment lifetime
  - migrate existing data after schema changes
  - minimise impact on existing applications
- Supported operations
  - add, move or remove attributes within classes
  - change inheritance hierarchy
- Migration of existing Objects
  - immediate: all objects are converted using an upgrade application
  - lazy: objects are upgraded as they are accessed

Data Analysis Scenario

- Hundreds of Physicist world-wide will
  - select events based on multiple interval queries on event properties
    - multi dimensional query : O(10-100) dimensions
    - some properties are user derived
    - list of properties differs between different analysis types
    - interval definition changes frequently
  - navigate to specific detector information
    - sample and store distributions of properties
    - define and store new persistent types for their private analysis objects

Ntuple versus TagDB Model

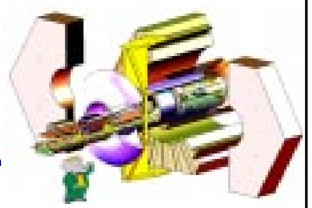


Purpose of Using Tags

- Tags are mainly used to speedup selections
  - Tag data is much better clustered than the original data
- A collection of Tags defines an Event Collection
  - in fact, tag collections are only a special case of an event collection
- Tag attributes may be visualised interactively
  - without the need to write any code
- Association to the Event may be used to navigate to any other part of the Event
  - even from an interactive visualisation program

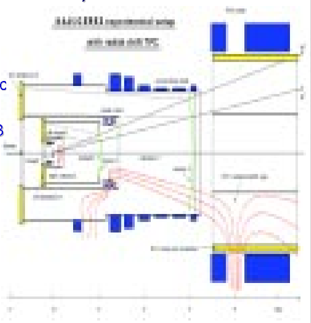
Production - BaBar

- BaBar at SLAC, due to start taking data in 1999
- Objectivity/DB is used to store event, simulation, calibration and analysis data
- Expected amount 200TB/year, majority of storage managed by HPSS
- Mock Data Challenge 2:
  - Production of 3-4 Million events in August/September
  - Partly distributed to remote institutes
- Cosmic runs starting in October



Production - CERES/NA45

- Heavy ion experiment at the SPS
- Study of e+e- pairs in relativistic nuclear collisions
- Successful use of Objectivity/DB from a reconstruction farm (32 Meiko CS2 nodes)
- Expect to write 30 TB of raw data during 30 days of data taking
- Reconstructed and filtered data will be stored using the Objectivity production service.



COMPASS

- COMPASS expects to begin full data taking in 2000 with a preliminary run in 1999.
- Some 300TB of raw data will be acquired per year at rates up to 35MB/second.
- Analysis data is expected to be stored on disk, requiring some 3-20TB of disk space.
- Some 50 concurrent users and many passes through the data are expected.
- Rely on the Objectivity production service at CERN

Summary

- ODBMS based data stores provide
  - **a single logical view** of complex object models
  - integration with **multiple OO languages**
  - support for **physical clustering** of data
  - scaling up to **PB distributed data stores**
  - seamless **integration with MSS** like HPSS
- Adopted by a large number of HEP experiments
  - even FORTRAN based experiments evaluate Objectivity/DB for analysis and data conservation
- Entering production phase at CERN now
  - Objectivity service has been set-up for ATLAS, CMS, COMPASS and NA45