# Object Databases as Data Stores for HEP

Dirk Düllmann

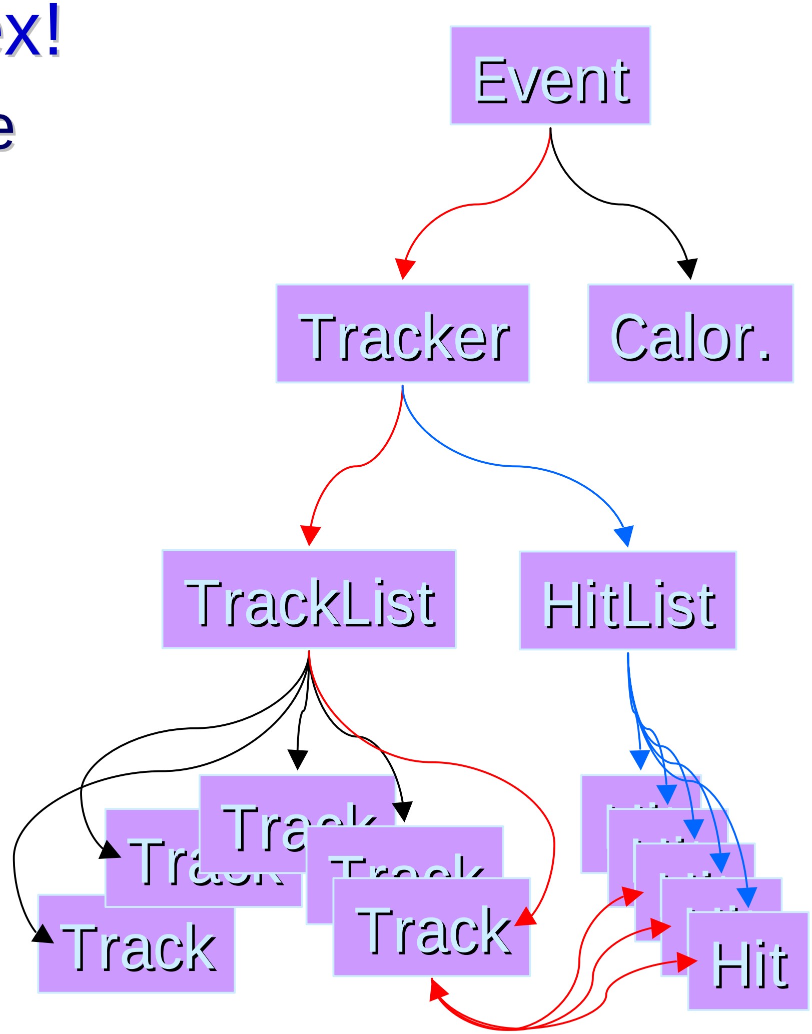
IT/ASD & RD45

# Outline of this Talk

- Intro
  - HEP Data Models
  - Data Management for LHC
- Object Database Features
  - What makes ODBMS suitable for HEP Data Models ?
- Objectivity/DB Features
  - Scalability
  - Data Management and Distribution
  - Alternative products and approaches
- HEP Projects using Objectivity/DB
  - Prototypes and production systems

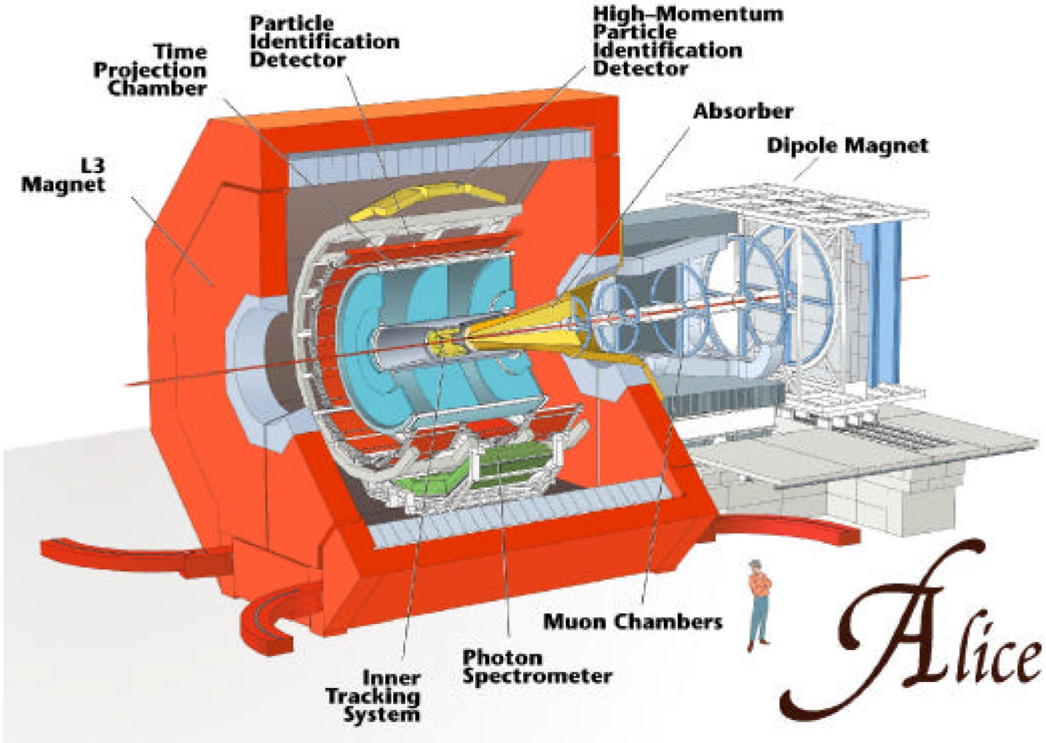
# HEP Data Models

- 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



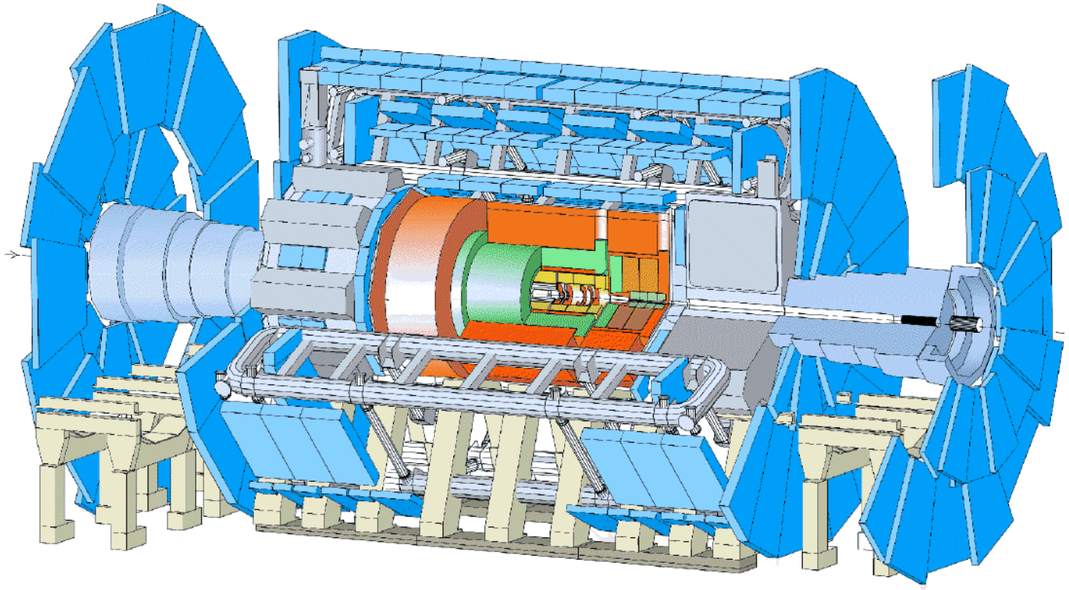
# ALICE

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



# ATLAS

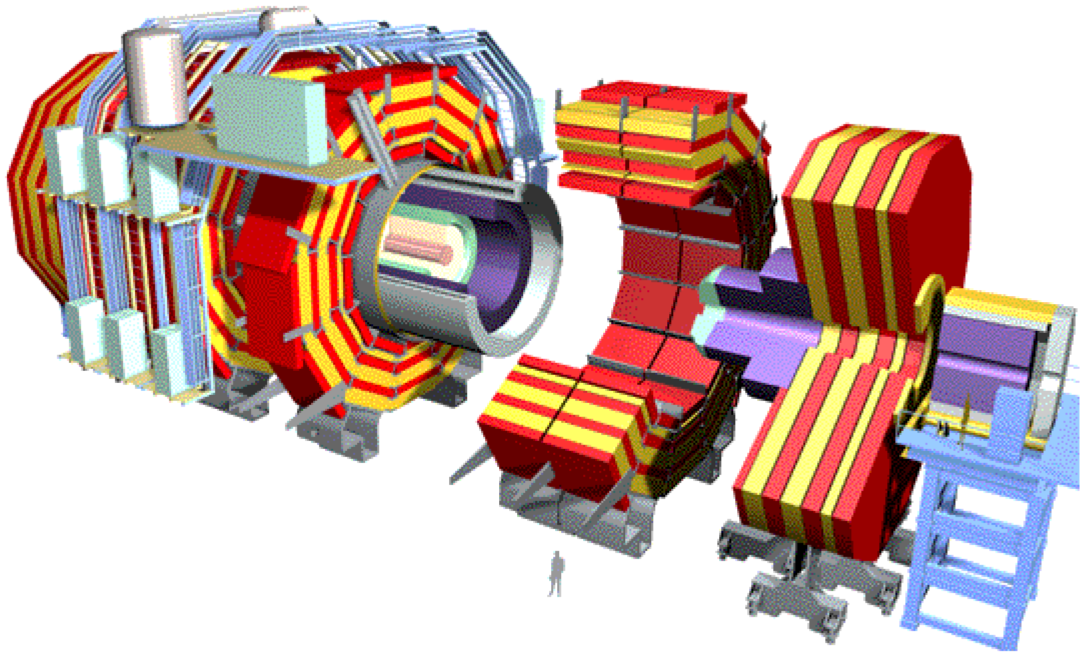
- General-purpose LHC experiment
- High Data rates:
  - 100MB/second
- High Data volume
  - 1PB/year
- Test beam projects using Objectivity/DB in preparation:
  - Calibration database
  - Expect 600GB raw and analysis data



# CMS

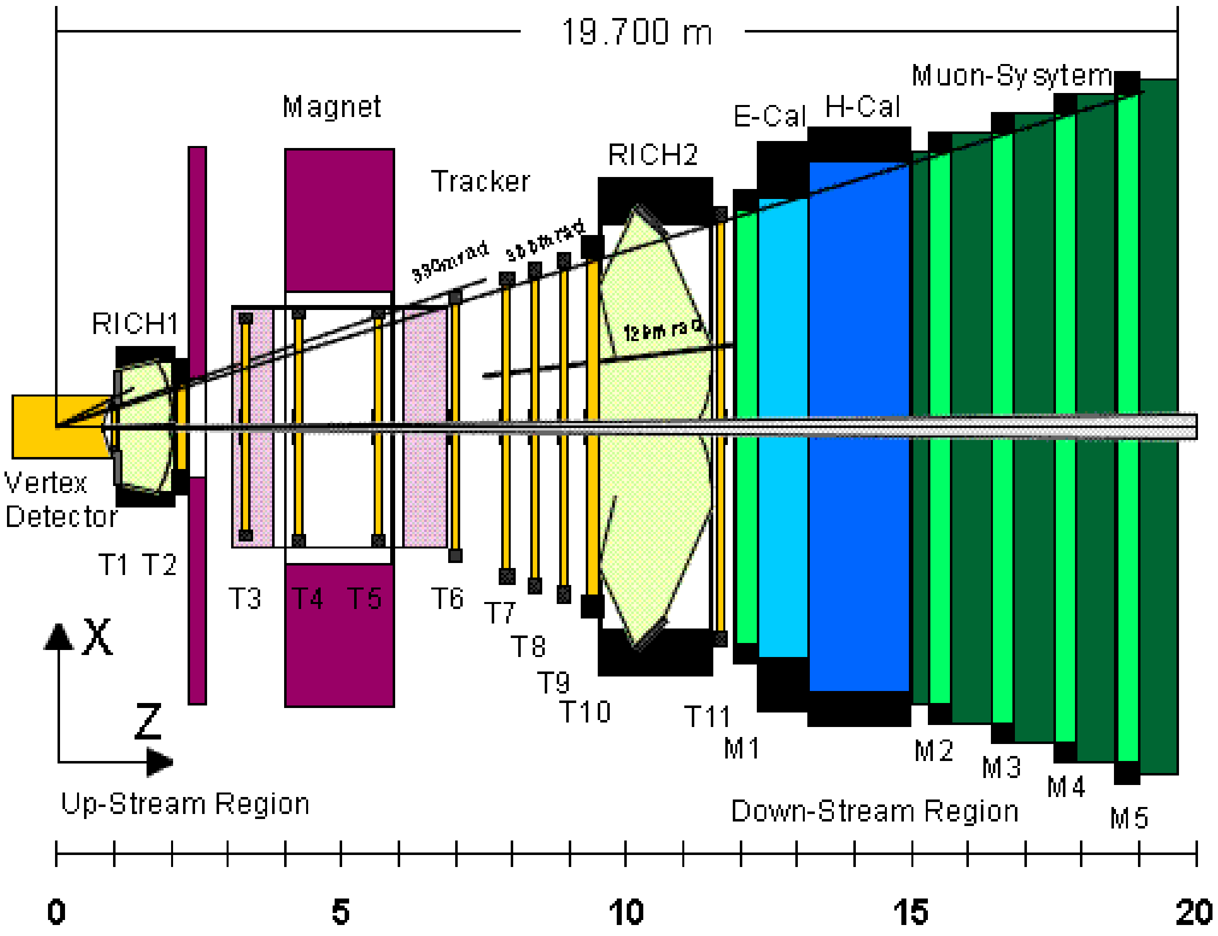
- General-purpose LHC experiment
- Data rates of 100MB/second
- Data volume of 1PB/year

- 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.



# 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
    - See Marcin Nowak's talk later this month

# Object Database Features

# 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

# Navigational Access

| Database# | Cont.# | Page# | Slot# |
|---|---|---|---|

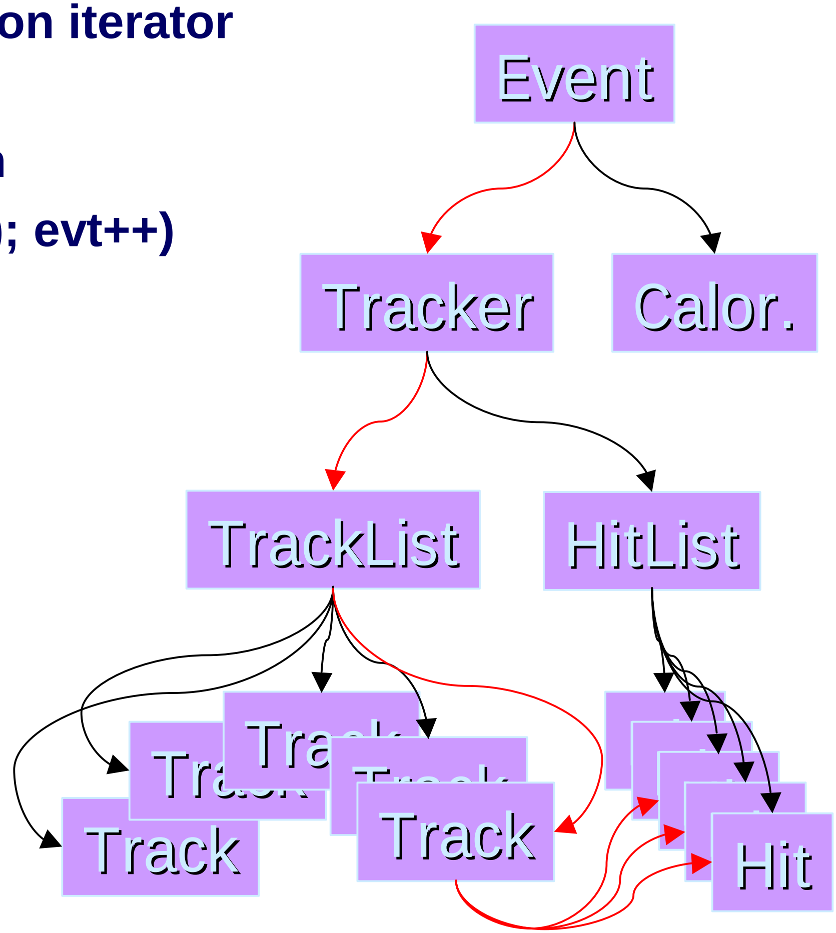
- Unique Object Identifier (OID) per object
  - Direct access to any object in the distributed store
  - Natural extension of the pointer concept
- OIDs allow to implement networks of persistent objects ("associations")
  - Cardinality: 1:1 , 1:n, n:m
  - uni- or bi-directional (referential integrity!)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

# How do "smart pointers" work?

- d_Ref<Track> is a smart pointer to a Track
- The database automatically locates objects as they are accessed and reads them.
- User does not need to know about physical locations.
  - No host or file names in the code
  - Allows de-coupling of logical and physical model

# A Code Example

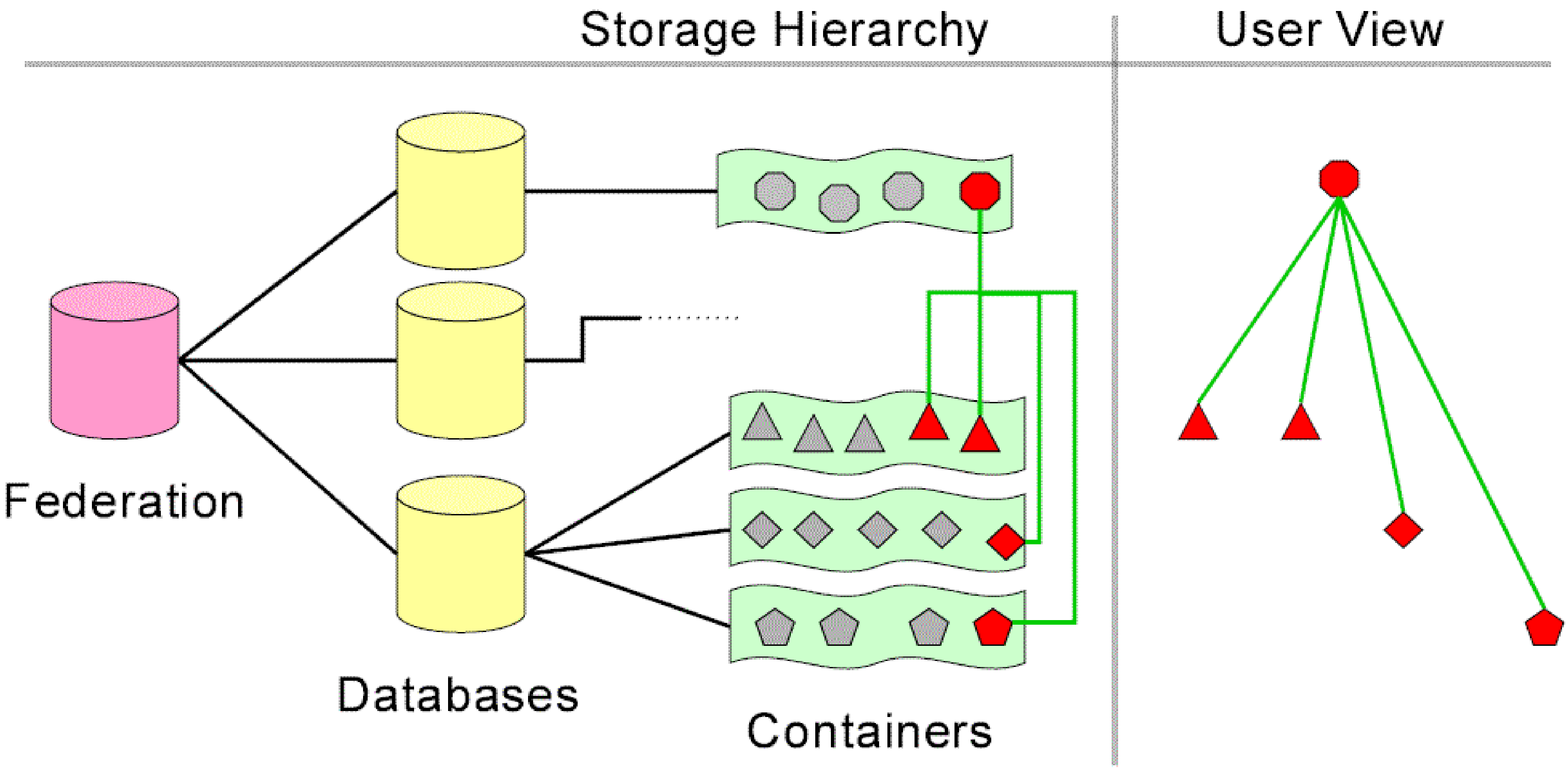
```
Collection<Event> events;           // an event collection
Collection<Event>::iterator evt;  // a collection iterator

// 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;
    }
```



# Object Clustering

- Goal: Transfer only "useful" data
  - from disk server to client
  - 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

# Physical Model and Logical Model




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

# Concurrent Access

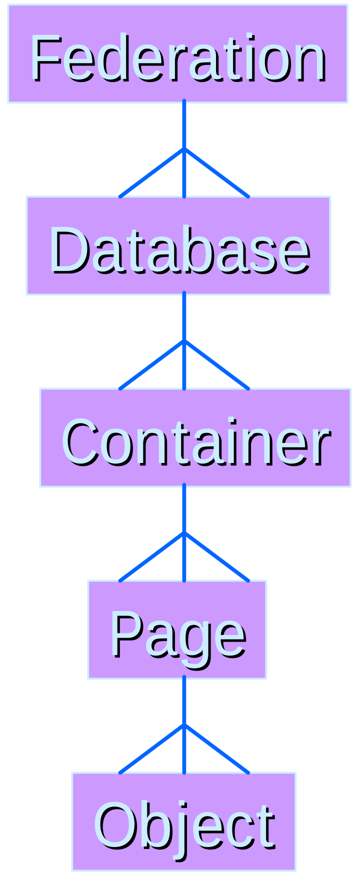
- Support for multiple concurrent writers
  - e.g. Multiple parallel data streams
  - e.g. Filter or reconstruction farms
  - e.g. Distributed simulation
- Data changes are part of a Transaction
  - ACID: Atomic, Consistent, Isolated, Durable
- Access is co-ordinated by a lock server
  - MROW: Multiple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

# Objectivity Specific Features

# 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



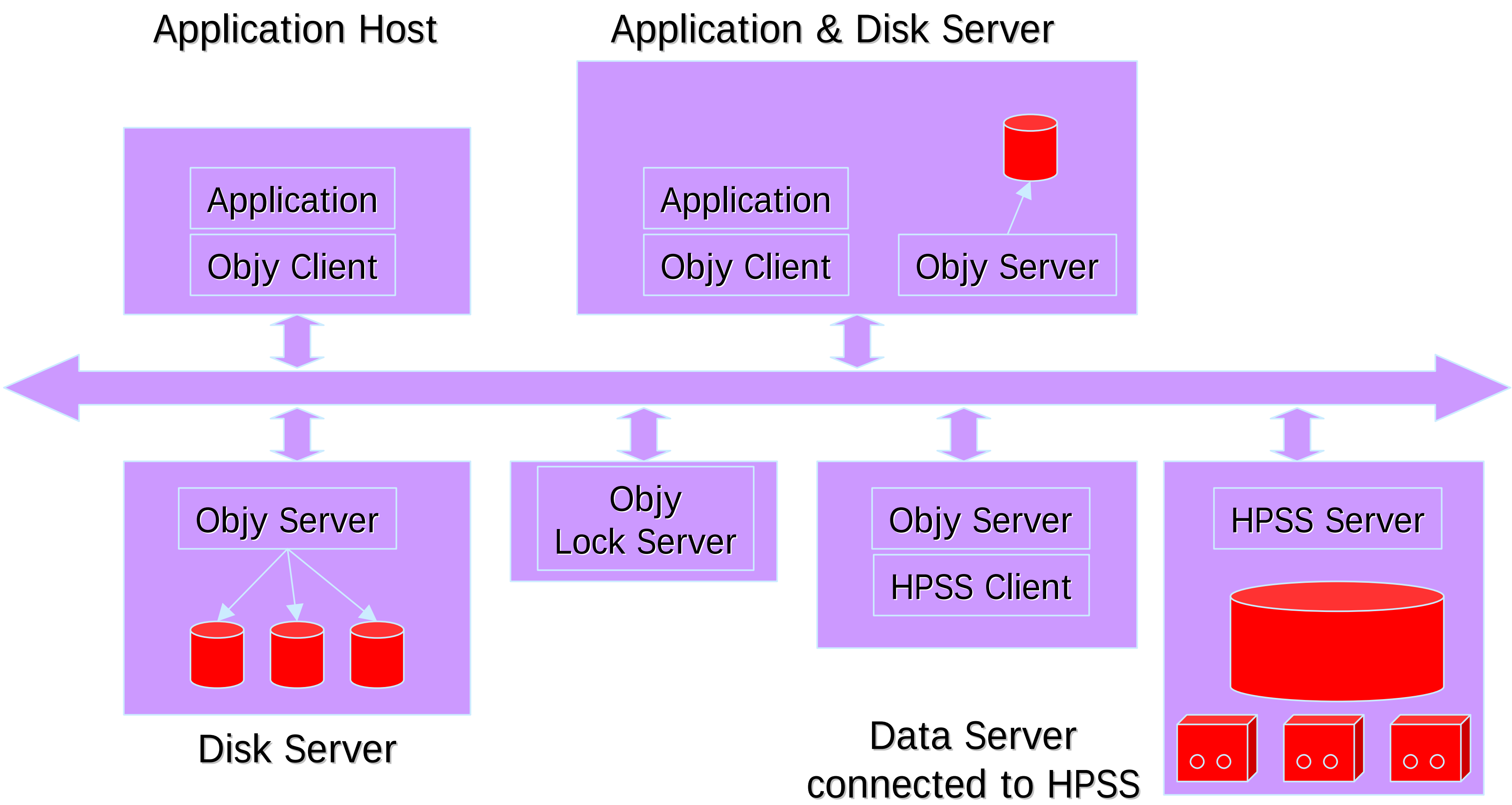
# Scalability Tests

- Federated Databases of **500GB** have been demonstrated
  - Multiple federations of 20-80GB are **used in production**
  - **32 filter nodes writing in parallel** into one federated database
  - **200 parallel readers** (Caltech Exemplar)

- Objectivity/DB shows expected scalability
  - Overflow conditions on architectural limits are handled gracefully
  - Only minor problems found and reported back
    - 2GB file limit; fixed and tested up to 25GB

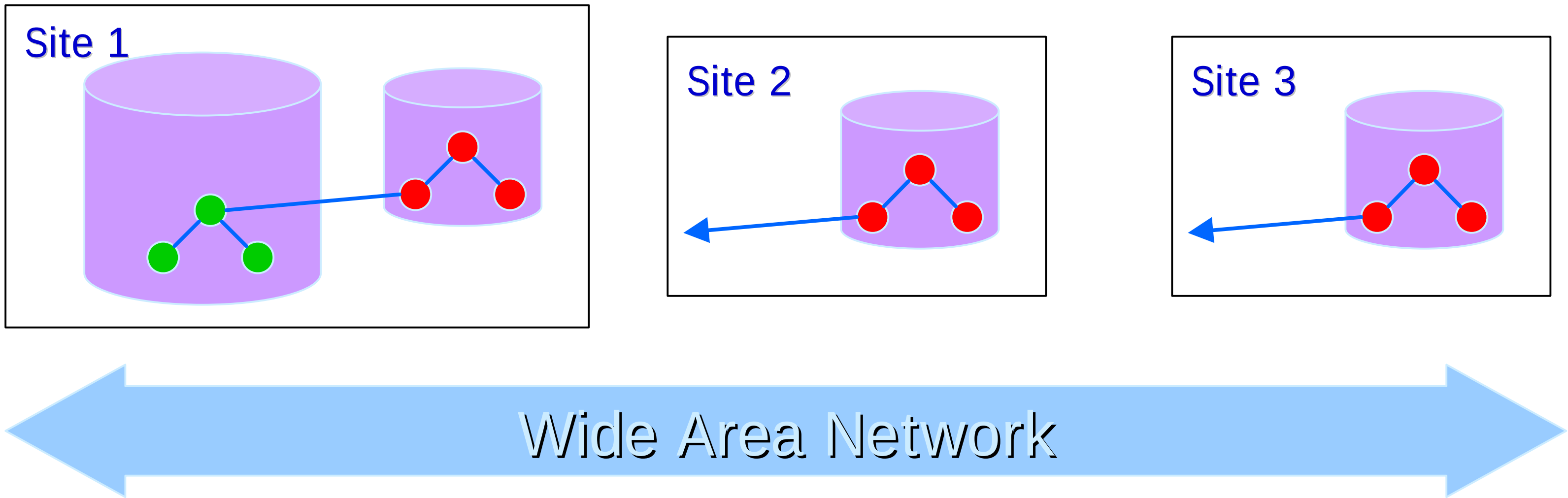
- Federations of hundreds of TB are possible with the current version

# A Distributed Federation



# 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

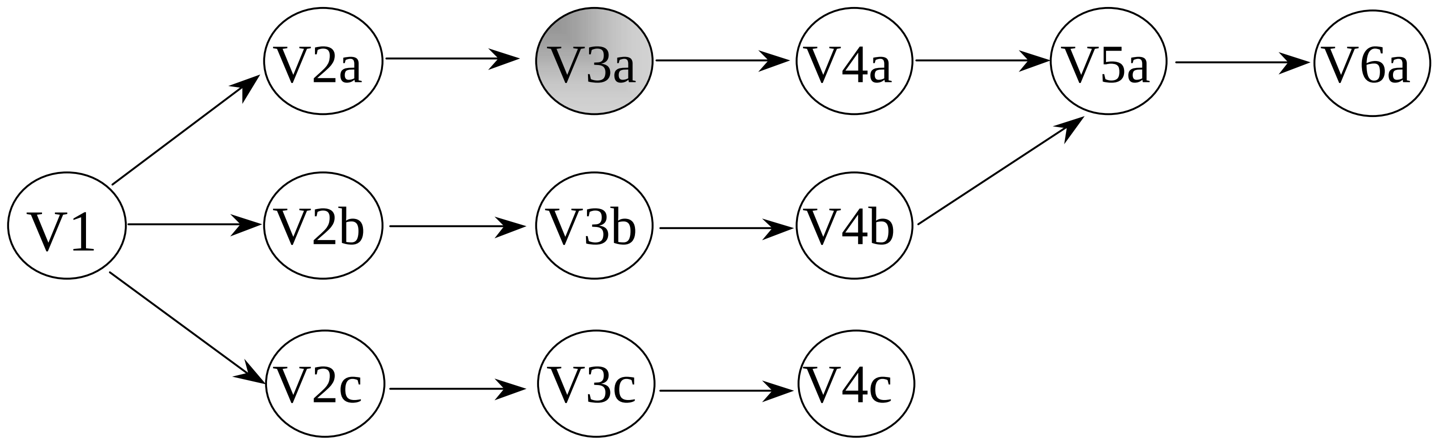


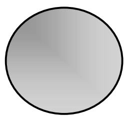
# Schema Evolution

- 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

# Object Versioning




 - default version

- Maintain multiple versions of an object
- Used to implement versions of calibration data in the BaBar calibration DB package

# Other O(R)DBMS Products

- Versant
  - Unix and Windows platforms
  - Scalable, distributed architecture
    - Independent databases
  - Currently most suitable fall back product
- O2
  - Unix and Windows platforms
    - Incomplete heterogeneity support
  - Recently bought by Unidata (RDBMS vendor) and merged with VMARK (data warehousing)

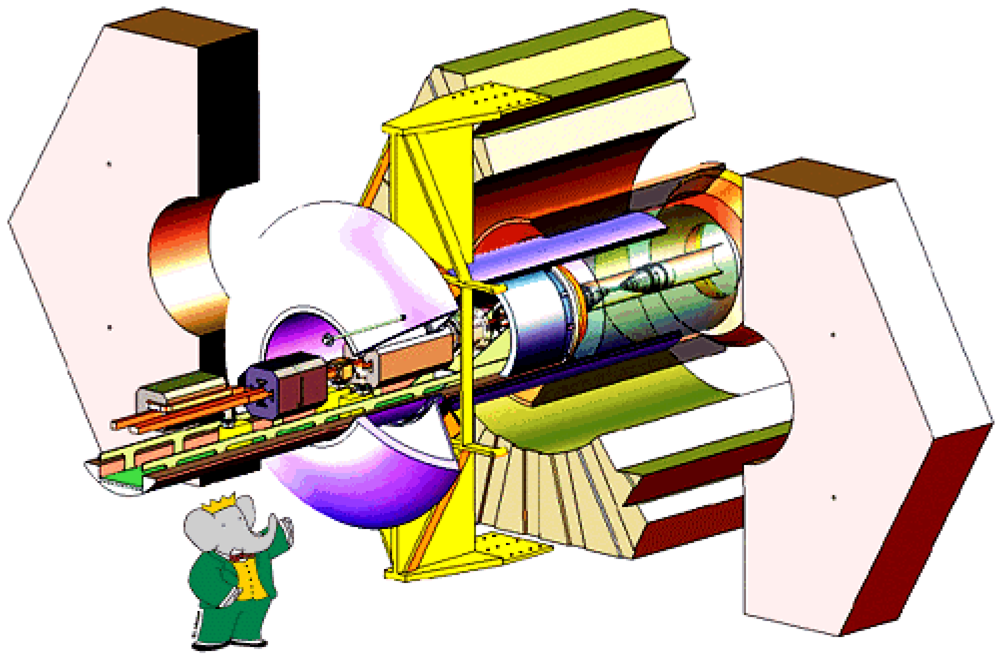
# Other O(R)DBMS Products II

- Objectstore - Object Design Inc.
  - Unix and Windows platforms
  - Scalability problems
  - Proprietary compiler, kernel driver
  - ODI re-focussed on web applications
- POET
  - Windows platform
  - Low end, scalability problems
- What will the big Object Relational Vendors provide?

# HEP Projects
# based on Objectivity/DB

# Production - BaBar

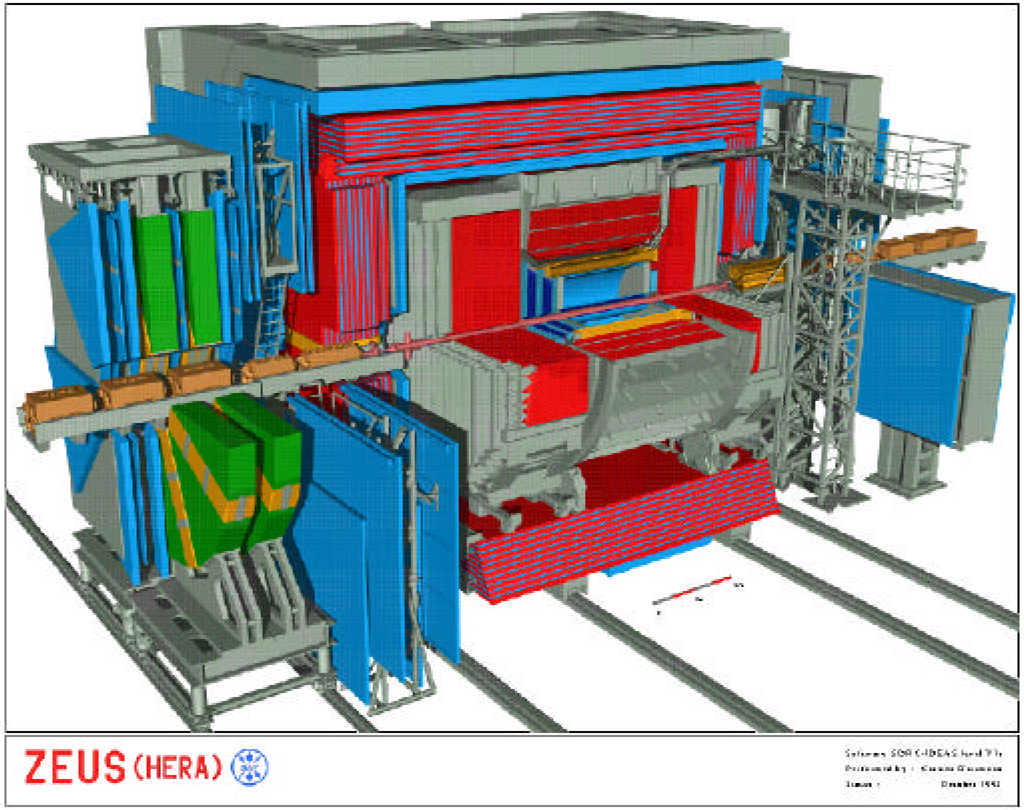
- BaBar at SLAC, due to start taking data in 1999
- Objectivity/DB is used to store event, simulation, calibration and analysis data
- Expected amount 200TB/year, majority of storage managed by HPSS
- Mock Data Challenge 2:
  - Production of 3-4 Million events in August/September
  - Partly distributed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



# Production - ZEUS

- ZEUS is a large detector at the DESY electron-proton collider HERA
- Since 1992 study of interactions between electrons and protons
- Analysis environment: mainly FORTRAN code based on ADAMO

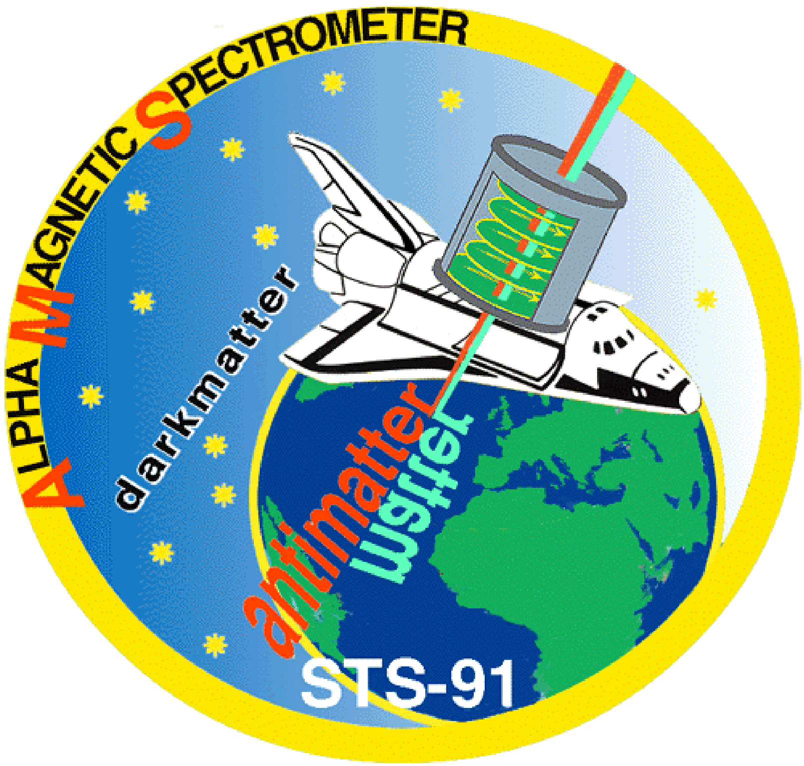
- Objectivity/DB is used for event selection in the analysis phase
- Store ~20GB of "tag data" - plan to extend to 200GB
- Reported a significant gain in performance and flexibility compared to the old system



# Production - AMS

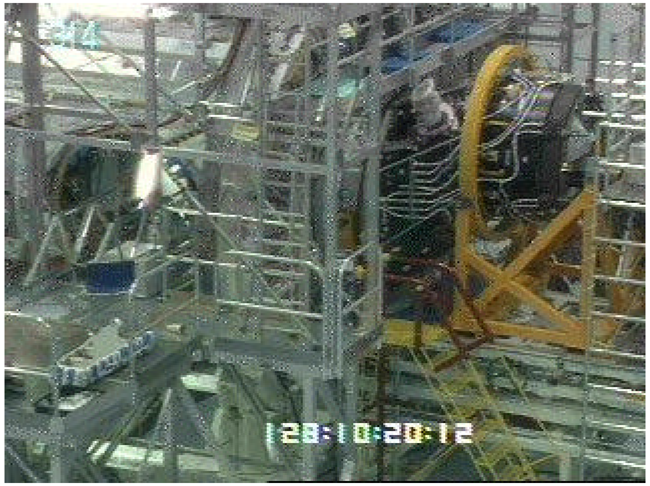
- The Alpha Magnetic Spectrometer will take data first on the NASA space shuttle and later on the International Space Station.
- Search for antimatter and dark matter

- Data amount 100GB
- Objectivity/DB is used to store production data, slow control parameters and NASA auxiliary data



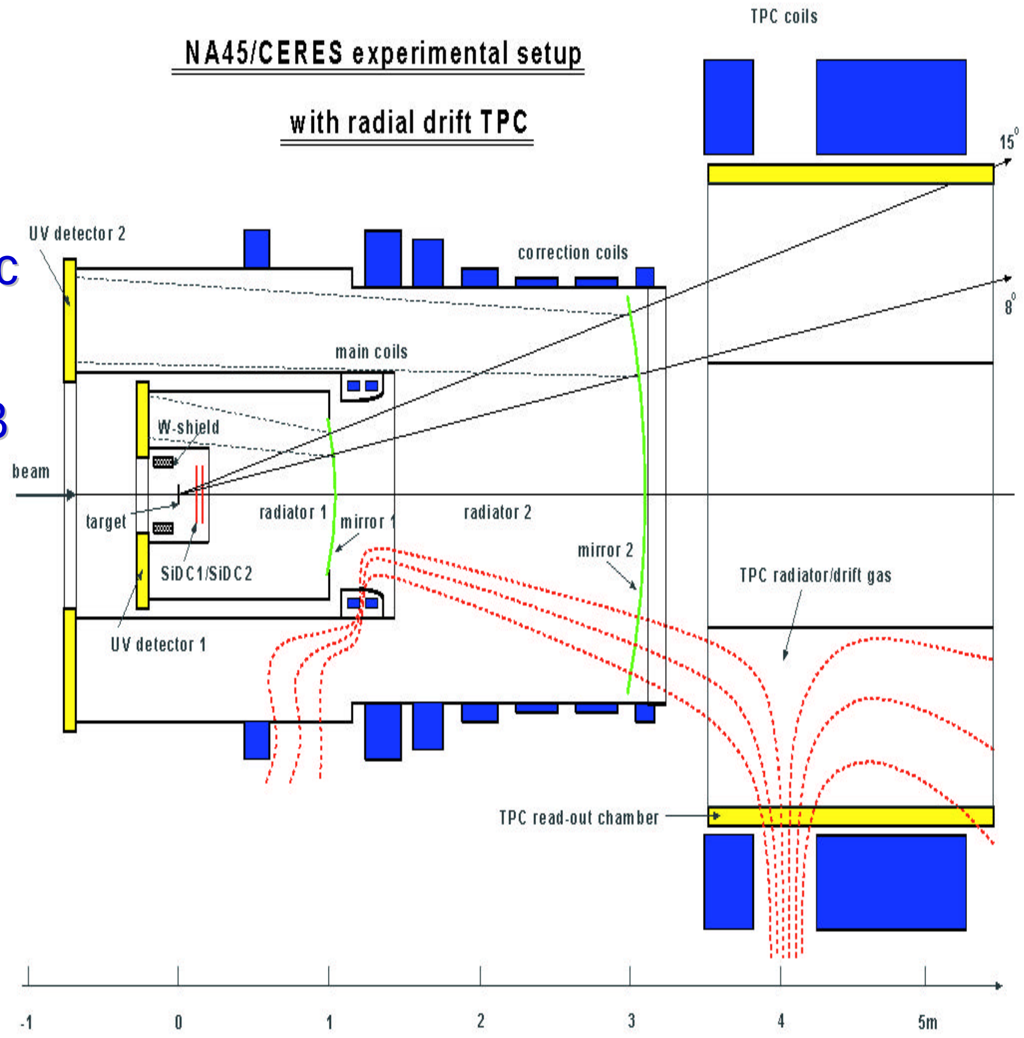
# Ready to go…





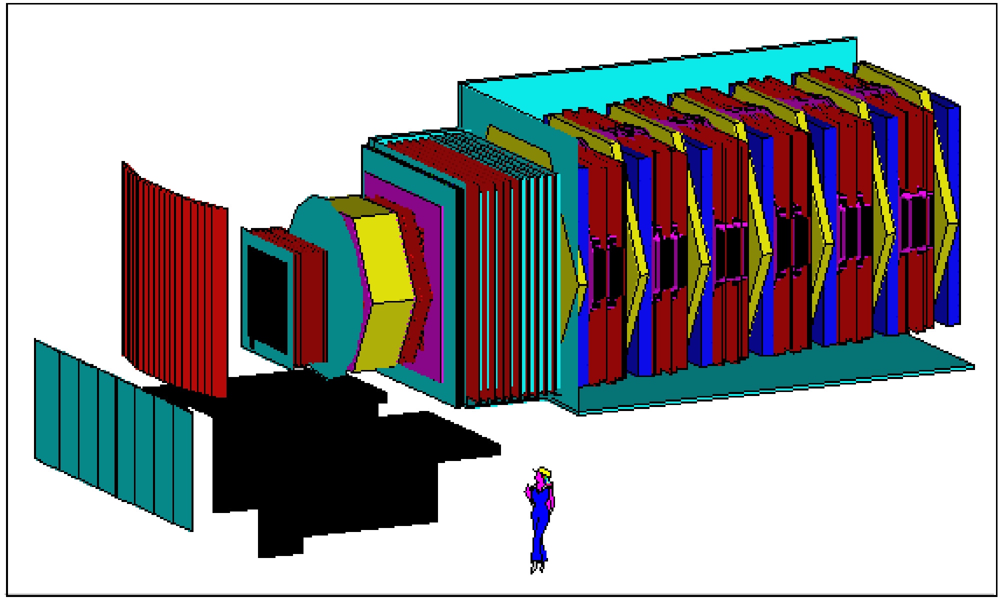

# Production - CERES/NA45

- 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.



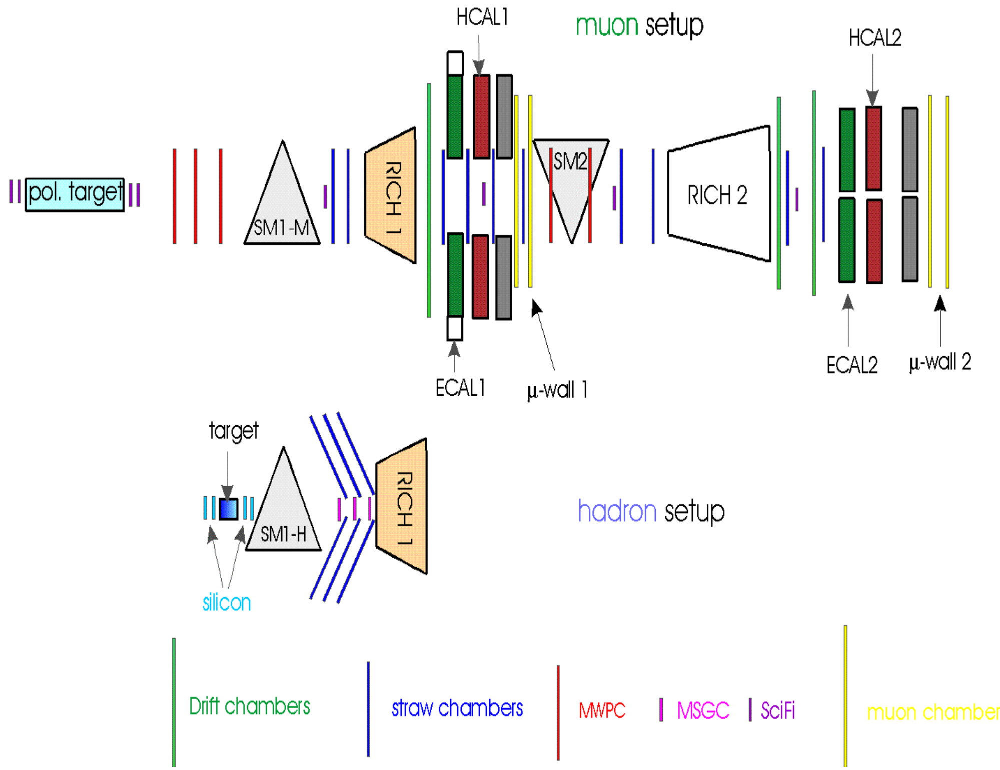
# Production - CHORUS

- Searching for neutrino oscillations
- Using Objectivity/DB for an online emulsion scanning database.
- Plans to deploy this application at outside sites.
- Also studying Objectivity/DB for TOSCA - a proposed follow-on experiment.



# 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

- Objectivity/DB 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
- Expect to enter production phase at CERN soon
  - Objectivity service will be set-up during this year