Session 8: Data Management and Persistency

Jacek Becla
David Malon

Outline

- Organizational notes
- Online/calibrations/conditions
- Reports from running experiments
- Transitions
- New development, emerging ideas, future
- Software at a glance
- Summary

Organizational Notes

- Almost all submitted abstracts accepted
- 28 talks, 20 min per talk
  - 5 online/configuration/conditions
  - 9 operational/experience
  - 14 new development, others

BaBar (6), ATLAS (5), CMS (3), POOL (3), CDF (2), COMPASS (2), D0 (2), ALICE (1), CLEO (1), GLAST (1), LCIO (1), PHENIX (1)

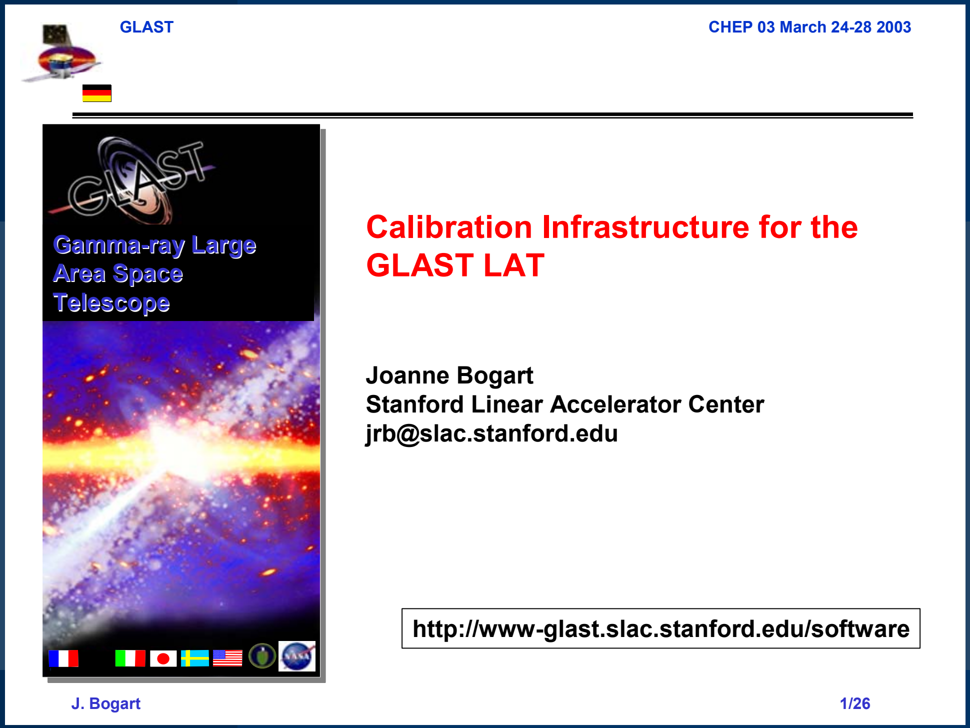
- Many GRID talks moved to other sessions
- Good interest/attendance
  - Given large number of parallel sessions

Outline

- Organizational notes
- **Online/calibrations/conditions**
- Reports from running experiments
- Transitions
- New development, emerging ideas, future
- Software at a glance
- Summary

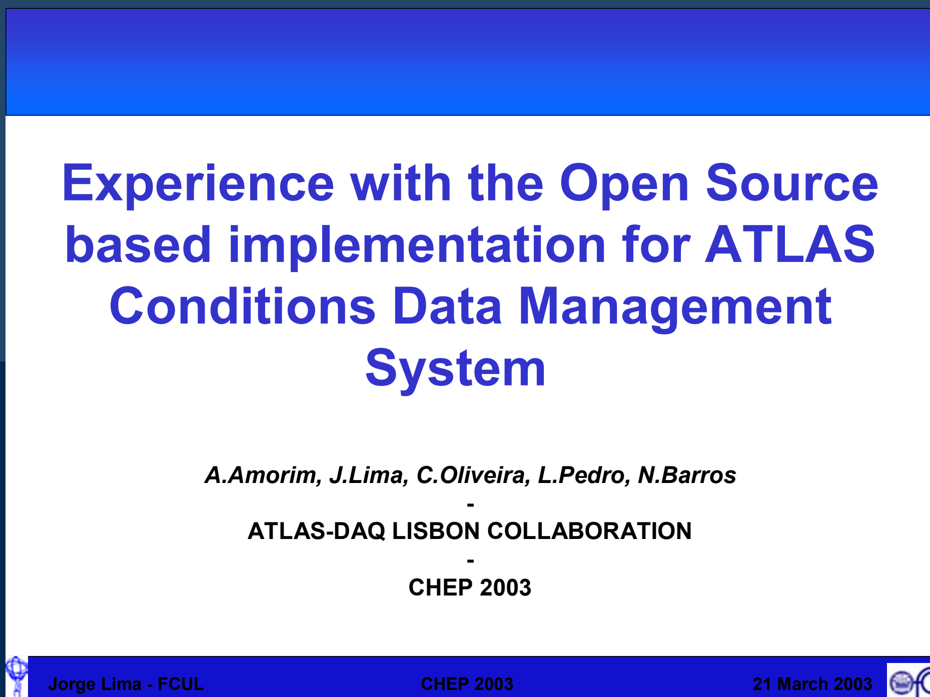
Online/Calib/Cond

- Heard from:
  - GLAST
  - ATLAS
  - BaBar
  - CLEO



"So far, the system is living up to expectations. The design effort was long and difficult; implementation and debugging haven't been bad. However, there is plenty left to do".

- *"Don't need to provide easy access to subset of a particular calibration data set" ("Anyone wanting calibration data gets the whole dataset")*
- Currently supports MySQL and XML, later will also support ROOT


Jorge Lima - FCUL CHEP 2003 21 March 2003

Experience with the Open Source based implementation for ATLAS Conditions Data Management System

A.Amorim, J.Lima, C.Oliveira, L.Pedro, N.Barros

-

ATLAS-DAQ LISBON COLLABORATION

-

CHEP 2003

- Open source
- Use only Standard SQL features
- Portability important
- Starting point: RD45 and BaBar Conditions
- Found deficiencies
- Proposing several extensions

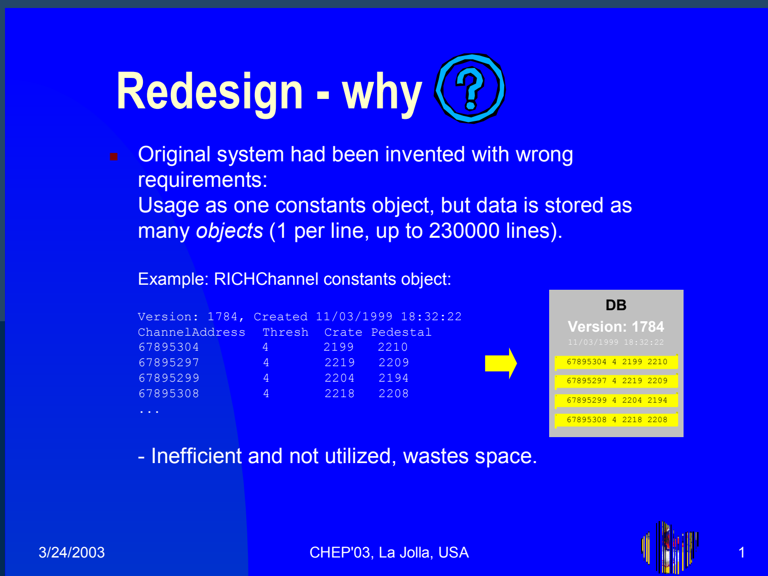
Conditions DB

- Main features
  - New conceptual model for metadata
    - 2-d space of validity and insertion time, revisions, persistent configurations, types of conditions, hierarchical namespace for conditions
  - Flexible user data clustering
  - Support for distributed updates and use
  - State ID
  - Scalability problems solved
  - Significant (100-1000x) speedup for critical use cases
- Status
  - In production since Fall'02
  - Data converted to new format
  - Working on distributed management tools

Configuration Database for BaBar On-line

- Currently there is one implementation, built on top of the Objectivity/DB ODBMS. Features which are important for implementation:
  - Direct addressing of persistent objects, no SQL-like queries.
  - Support for inter-object associations.
  - Mapping to C++ classes.
- Configuration database is a vital part of the BaBar DAQ system and proved to be sufficiently performant and reliable.

- Conditions DB freshly redesigned
- New computing model: abandoning ROOT-based conditions
- Online, Calib & Cond stay in Objy



"Initial implementation: naïve, performance seemed good"

- Online DB in Objy
- Accessed via CORBA
- Redesigned
  - original system based on wrong requirements
  - inefficient, wasted space
  - new system: all 23000 constants in one persistent object
  - 20 GB, data converted < a day

Online/Calib/Cond: General Trends

- Running experiments
  - Did not anticipate problems, bottlenecks
  - Found initial implementation insufficient
  - Non-trivial redesigns
  - Backwards compatibility/switching ok, thanks to small volumes of data
- Non-running experiments
  - Finding existing APIs insufficient
  - Open source RDBMS

Should there be community-wide redesign?

Outline

- Organizational notes
- Online/calibrations/conditions
- **Reports from running experiments**
- Transitions
- New development, emerging ideas, future
- Software at a glance
- Summary

Reports from Running Experiments

- SAM
- DAN

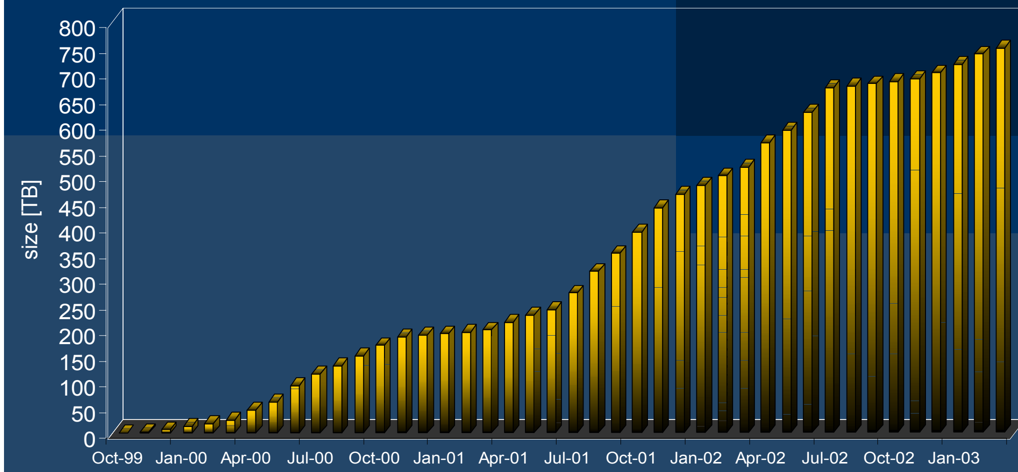
Also
- CMS (Monte Carlo Prod DB)
- Alice (detector construction)

Statistics

- Total size 750 TB
- 576000 database files
- Over 100 Objectivity/DB federations
- 88 TB of disk space - 50 servers
- Over 50 other servers
  - Lock servers, journal servers
- 60+ million collections

2

Database Operations



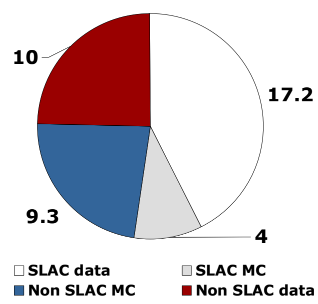
Providing Persistency for BaBar

- Growing complexity and demands
- Changing requirements
- Hitting unforeseen limits in many places
- Non-trivial maintenance
  - Most problems are persistent-technology independent
  - System becoming more and more distributed
- Very lively environment
  - Production not as stable as one would imagine

CHEP'03

3 of 18

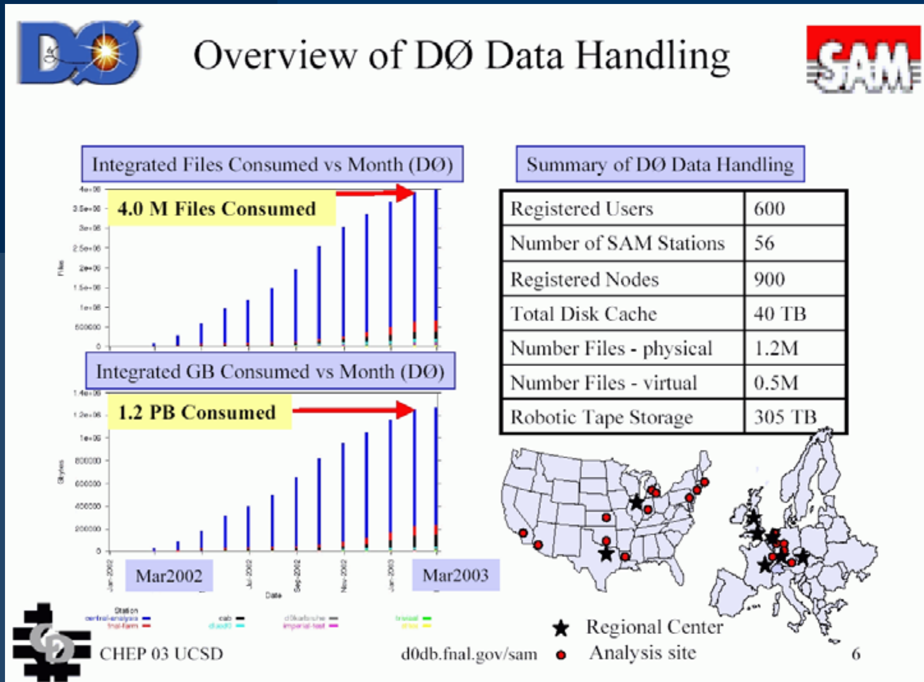
Data Transfer, Imports, Exports



Production since Nov'02 (TB)

- Non SLAC production ratio increased from 0 to 42%
- 25 institutions contribute to simulation production, 1 site (INFN-Padova) runs Event Reconstruction
- Export of full Objy dataset to IN2P3
- High performance copy programs – bbcp, bbftp
- 4 hosts dedicated for import/export operations

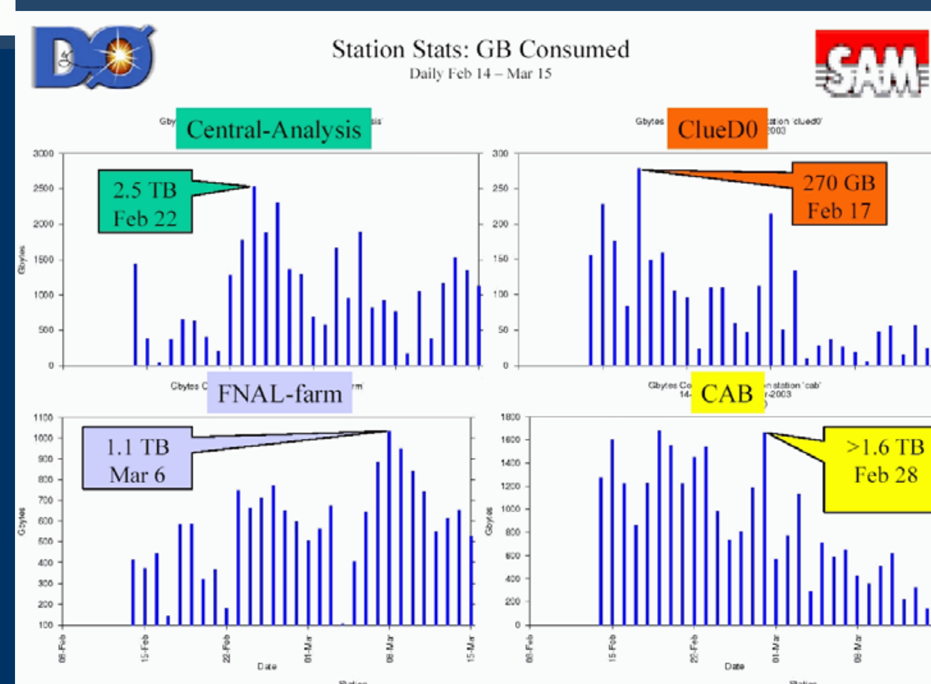
12



Registered Users	600
Number of SAM Stations	56
Registered Nodes	900
Total Disk Cache	40 TB
Number Files - physical	1.2M
Number Files - virtual	0.5M
Robotic Tape Storage	305 TB

SAM - Sequential data Access via Metadata

- Sophisticated and capable data distribution system
- Intelligent caching, data fetching from remote and FNAL SAM stations



"Getting SAM to meet the needs of D0 in the many configurations is and has been an enormous challenge."

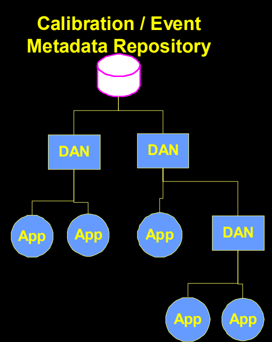
"The system is continually being improved."

DAN
(Database Architecture Networked)

Mediation layer between application and database with multilevel cache

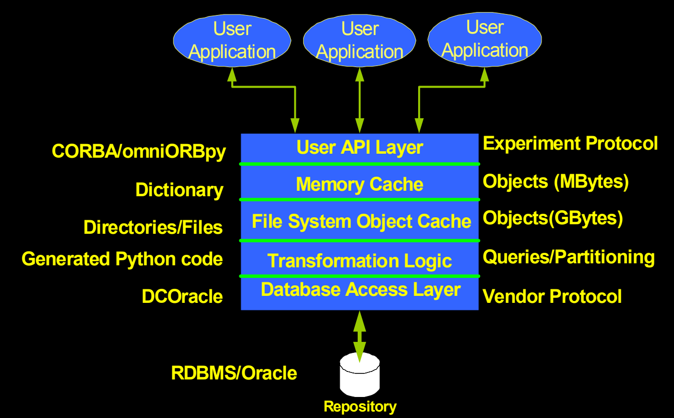
What is DAN?

- A multi-tiered Python server between the database and the user applications
- A server that performs database transactions on behalf of the user
- A service that provides an application-level protocol for accessing calibrations and event meta data

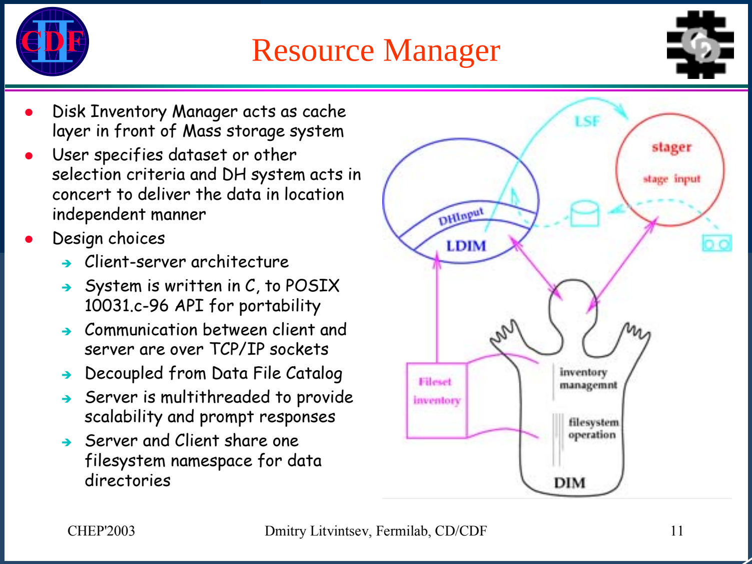


3/27/2003 DAN 5

Architecture and Design



3/27/2003 DAN 6



Report on current CDF data handling

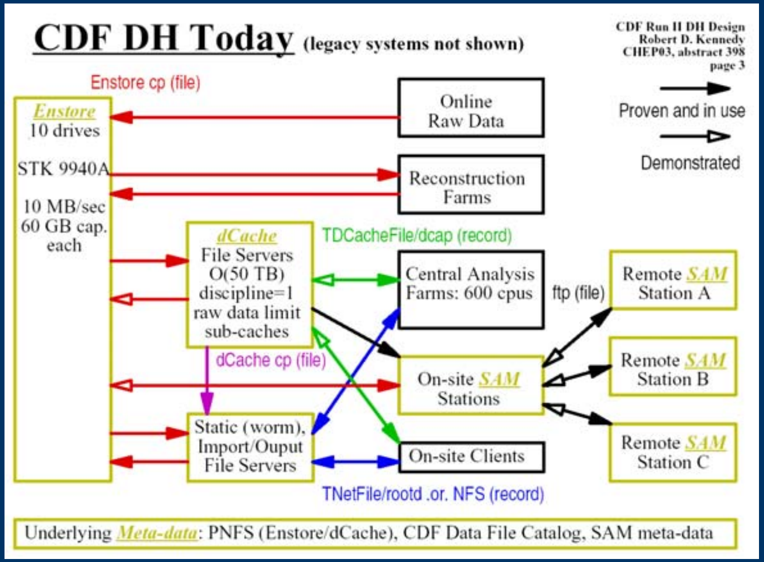
moving to

⇒ dCache = Network-accessible disk cache as front-end to MSS

Originally developed at DESY (Patrick Fuhrmann et al), now co-maintained by DESY and Fermilab CCF dept. Primary goal: rate-adapting front-end to an MSS. Oriented towards on-site client access. Expects reliable network, so no integrity check.

⇒ SAM = Data Handling framework, a "proto-DataGrid"

Originally developed by D0, FNAL CD. In use for some time at D0. "Stations" serve local disk caches to clients, talk to other Stations or a MSS to get files not in cache.



Phenix – file catalog replication

Database technology choice

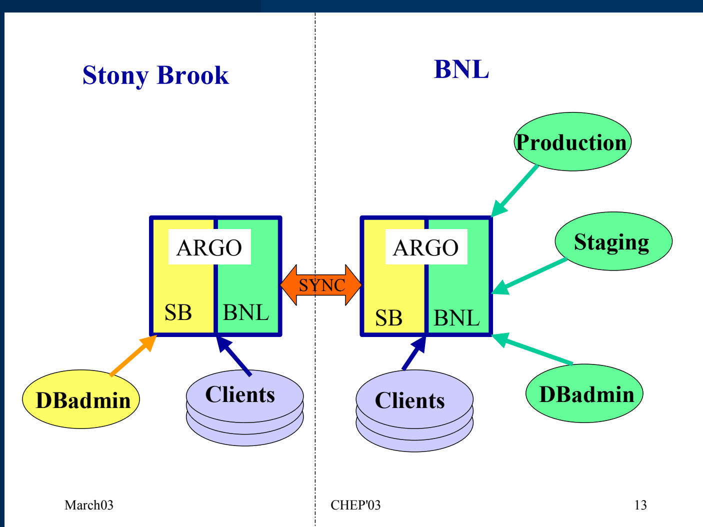
- Objectivity – problems with peer-to-peer replication
- Oracle was an obvious candidate(but expensive)
- MySQL didn't have ACID properties and referential integrity a year ago when we were considering our options. Had only master-slave replication
- postgreSQL seemed a very attractive DBMS with several existing projects on peer-to-peer replication
- SOLUTION: to have central Objy based metadata catalog and distributed file replica catalog

March03 CHEP'03 9

PostgreSQL Replicator

- http://pgreplicator.sourceforge.net
- Partial, peer-to-peer, async replication
- Table level data ownership model
- Table level replicated database set
- Master/Slave, Update Anywhere, Workload Partitioning data ownership models are supported
- Table level conflict resolution

March03 CHEP'03 11



- LISTEN and NOTIFY support message passing and client notification of an event in the database. Important for automating data replication
- 20 K new updates < 1min

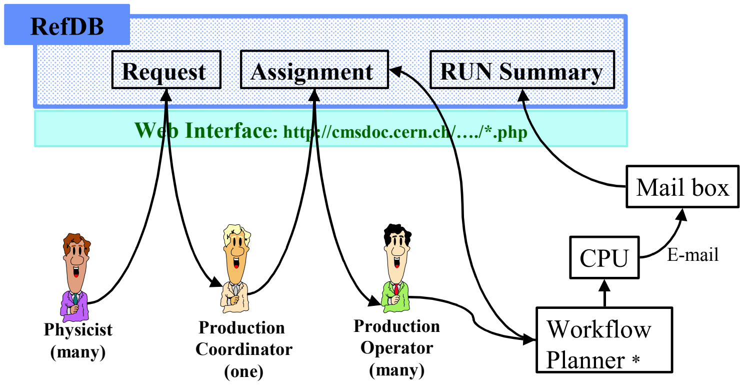
Would this peer-to-peer approach scale with large numbers of catalogs?

RefDB: The Reference Database for CMS Monte Carlo Production

Véronique Lefébure
CERN & HIP

CHEP 2003 - San Diego, California
25^{th} of March 2003

General Data Flow



*IMPALA, McRunjob, CMSProd

Functionalities of RefDB

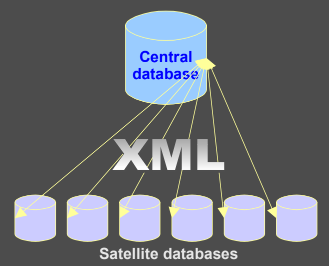
1. Management of **Physics Production Requests**
2. Distribution, Coordination and Progress Tracking of Production around the World: **Production Assignments**
3. Definition of **Production Instructions** for workflow-planner
4. **Catalogue Publication** of Real and Virtual Data

- **MySQL** Database hosted at CERN
- **Web-server, .htaccess** and **Php** scripts

Detector Construction Database System
for ALICE Experiment

Alice Detector Databases – architecture

Satellite databases
- Placed in laboratories-participants
- contain source data
  - produced at laboratories
  - delivered by manufacturers
- working copies of data from central repository
- Partial copies of metadata (read only)



Central database
- placed at CERN (temporarily was placed at WUT)
- Plays role of central repository
- contains
  - central inventory of components
  - copies of data from laboratories
  - metadata, e.g. Dictionaries

Communication
- passing messages in XML
- mainly off-line (batch processing)
- no satellite-satellite communication!
- request-response model (like in HTTP)
  - only satellite database can initiate communication

CHEP'03 March 27th, 2003 San Diego Wiktor S. Peryt, Warsaw University of Technology

Oracle for central DB

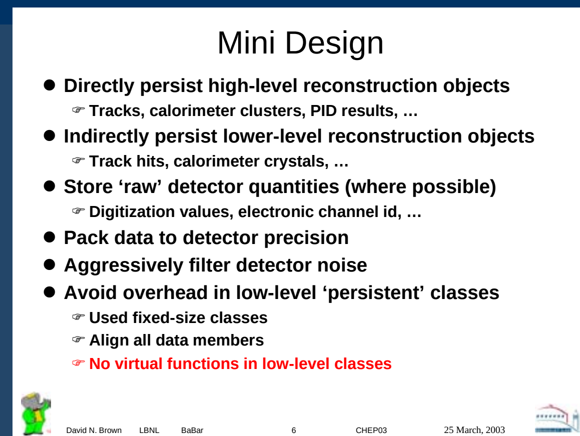
PostgreSQL for satellite DBs

ALICE reached pre-production phase and have tools for populating detector construction database.

Some satellite databases already exist
- Strasbourg/Utrecht, Trieste, Helsinki - SSD
- Torino - SDD
- Moscow - PHOS
- Nantes - Dimuon Station
- Darmstadt - TPC & TRD (in statu nascendi, db scheme and dictionaries, not populated yet; btw - the only satellite database with Oracle DBMS)
- CERN - central repository. DB scheme installed on SUN cluster; application server in preparation by CERN team. Start for normal production expected in April.
- Prototype version of Central Database - installed on server in Warsaw ~ 1 year ago

Improvements - BaBar

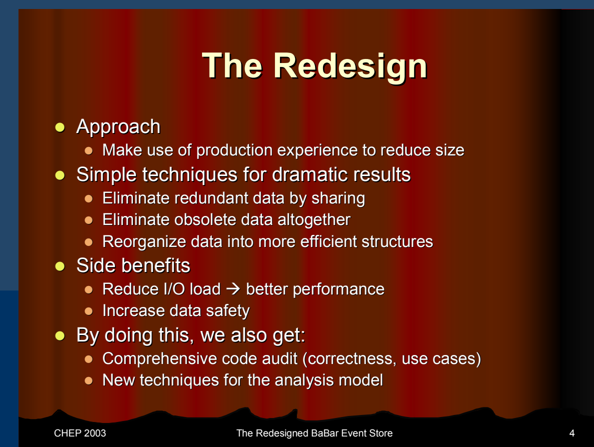
- New Mini
- Load balancing
- Data compression
- Event store redesign
- Turned off raw, rec, sim



Mini Design

- Directly persist high-level reconstruction objects
  - Tracks, calorimeter clusters, PID results, ...
- Indirectly persist lower-level reconstruction objects
  - Track hits, calorimeter crystals, ...
- Store ‘raw’ detector quantities (where possible)
  - Digitization values, electronic channel id, ...
- Pack data to detector precision
- Aggressively filter detector noise
- Avoid overhead in low-level ‘persistent’ classes
  - Used fixed-size classes
  - Align all data members
  - No virtual functions in low-level classes

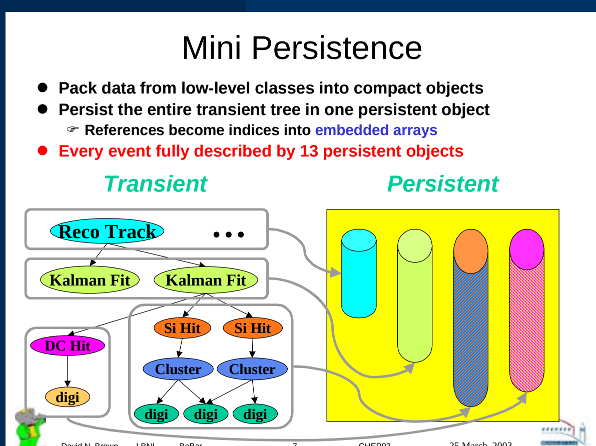
David N. Brown LBNL BaBar 6 CHEP03 25 March, 2003



The Redesign

- Approach
  - Make use of production experience to reduce size
- Simple techniques for dramatic results
  - Eliminate redundant data by sharing
  - Eliminate obsolete data altogether
  - Reorganize data into more efficient structures
- Side benefits
  - Reduce I/O load → better performance
  - Increase data safety
- By doing this, we also get:
  - Comprehensive code audit (correctness, use cases)
  - New techniques for the analysis model

CHEP 2003 The Redesigned BaBar Event Store 4



Mini Persistence

- Pack data from low-level classes into compact objects
- Persist the entire transient tree in one persistent object
  - References become indices into embedded arrays
- Every event fully described by 13 persistent objects

David N. Brown LBNL BaBar 7 CHEP03 25 March, 2003

- Solid, flexible base very important
- Not enough focus on analysis in the first two years
- Understanding importance of designing persistent schema

BaBar learning from experience

Outline

- Organizational notes
- Online/calibrations/conditions
- Reports from running experiments
- Transitions
- New development, emerging ideas, future
- Software at a glance
- Summary

Technology Transitions. Prototyping in LHC

CMS, Bill Tanenbaum

ROOT Based Framework

- Replace **Objectivity** with **ROOT** in framework
- All persistency capable classes **ROOT**ified (including metadata)
- Use **STL** classes (e.g. **vector**).
- No **ROOT** specific classes used, except for Persistent References (**TRef** class)
- No redesign of framework
- Foreign classes used extensively

24/03/2003 Bill Tanenbaum US-CMS/Fermilab 3

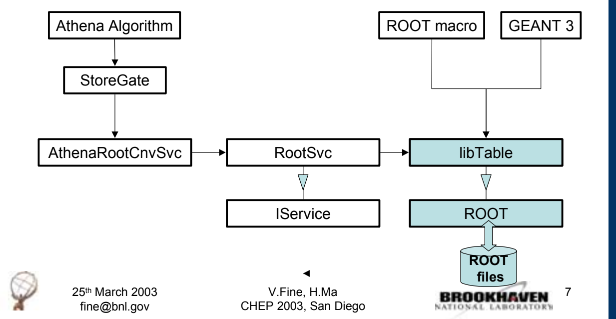
CMS, David Chamont

- 4 kinds of containers for the crossing data model
- 3 kinds of persistency managers.

ATLAS, Valeri Fine

Access to data inside and outside of ATLAS Athena framework

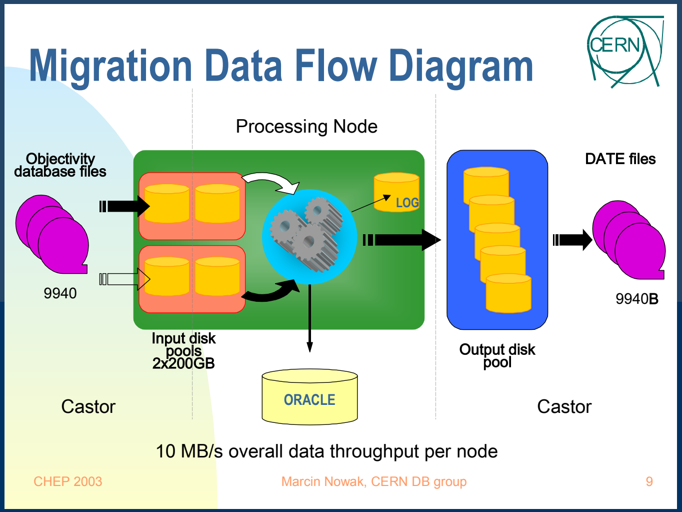
ROOT I/O for Athena algorithm and non-Athena applications

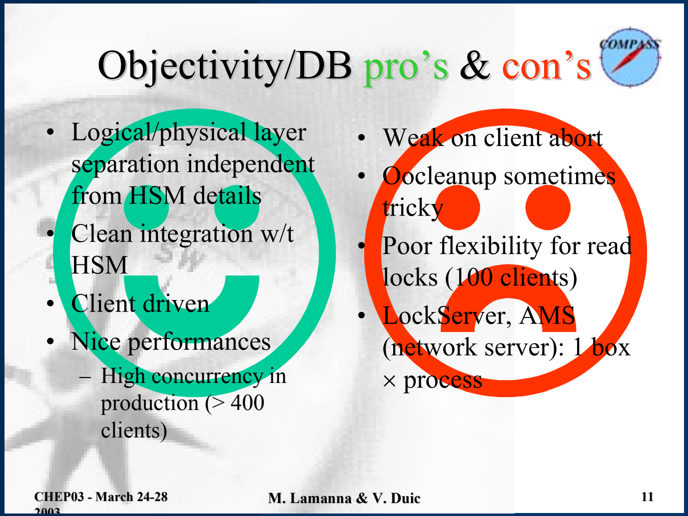


Both ATLAS and CMS committed to POOL as baseline

Technology Transitions: Compass, HARP

- Compass: 300TB
- HARP: 30TB
- Moving from Objy to hybrid: Oracle+flat files
- Bulk data stored as BLOBs





- The proposed new data storage system:
  - A hybrid solution based on a relational database and flat files
  - Preserving essential features of the current system
    - navigational access to events and reconstructed events

"Would have stayed with Objy, should CERN not terminate the contract"

Technology Transitions: BaBar (!)

- New Computing Model
  - Deprecate Objy-based event store
    - To follow general HEP trend
    - To allow interactive analysis in ROOT
  - Deprecate ROOT-based conditions
  - Very aggressive timescale

The Implementation: KANGA (ROO)

Kind ANd Gentle Analysis (without Relying On Objectivity)

A significant factor in the rapid deployment of KANGA was the earlier design decision to completely decouple the event store technology from the analysis framework.

Outline

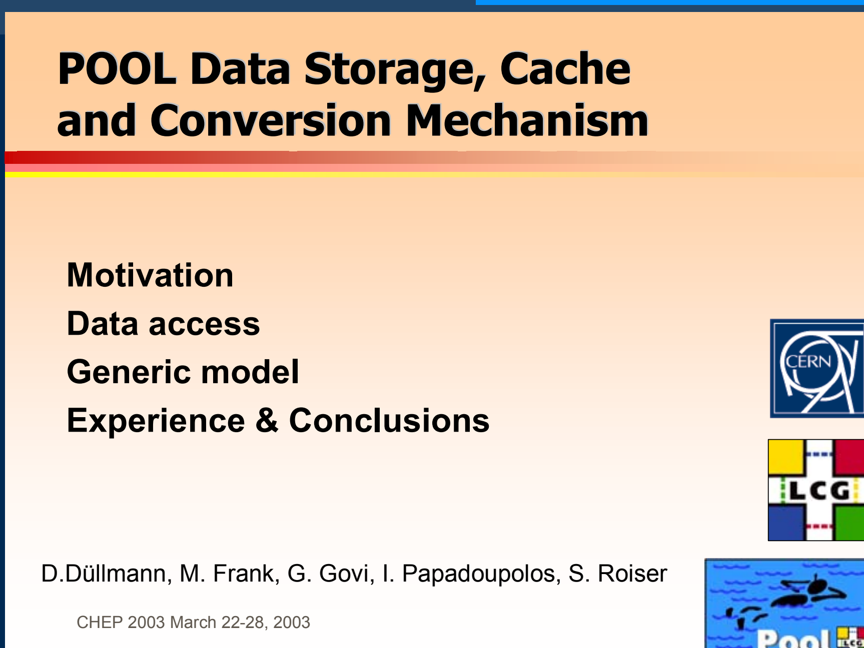
- Organizational notes
- Online/calibrations/conditions
- Reports from running experiments
- Transitions
- **New development, emerging ideas, future**
- Software at a glance
- Summary

New Development - POOL


POOL Project Overview

Dirk Düllmann
(on behalf of the POOL team)

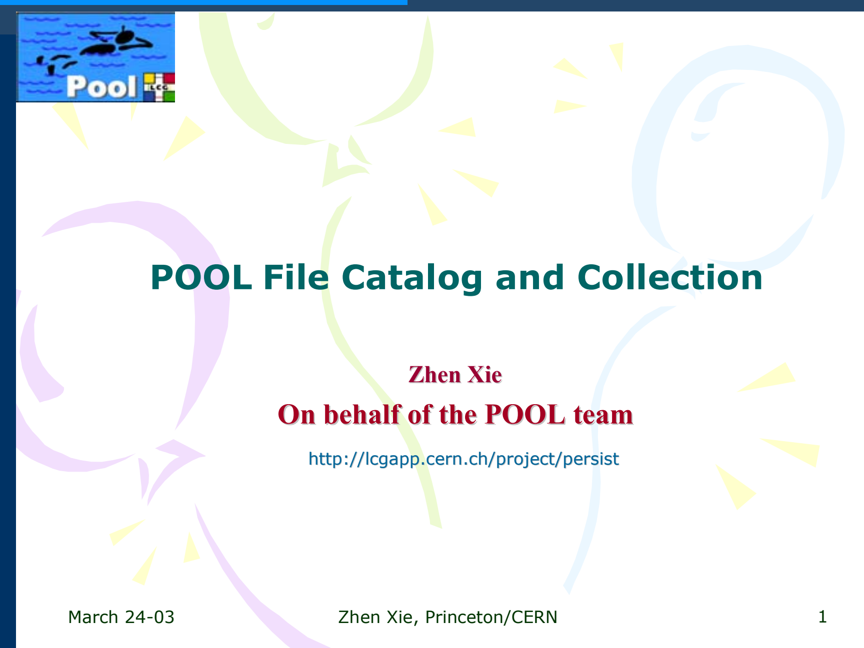
CHEP '03 – San Diego
24th March 2003


POOL Data Storage, Cache and Conversion Mechanism

- Motivation
- Data access
- Generic model
- Experience & Conclusions

D.Düllmann, M. Frank, G. Govi, I. Papadoupolos, S. Roiser

CHEP 2003 March 22-28, 2003


POOL File Catalog and Collection

Zhen Xie
On behalf of the POOL team
http://lcgapp.cern.ch/project/persist

March 24-03 Zhen Xie, Princeton/CERN 1

- POOL is a component based system
  - A technology neutral API
    - Abstract C++ interfaces
  - Implemented reusing existing technology
    - ROOT I/O for object streaming
      - complex data, simple consistency model (write once)
    - RDBMS for consistent meta data handling
      - simple data, transactional consistency

POOL Work Package breakdown

- Based on outcome of SC2 persistency RTAG
- File Catalog
  - keep track of files (and their physical and logical names) and their description
  - resolve a logical file reference (FileID) into a physical file
  - pool::IFileCatalog
- Collections
  - keep track of (large) object collection and their description
  - pool::Collection<T>
- Storage Service
  - stream transient C++ objects into/from storage
  - resolve a logical object reference into a physical object
- ject Cache (DataService)
  - keep track of already read objects to speed up repeated access to the same data
  - pool::IDataSvc and pool::Ref<T>

- **Hide any technology details from the clients**
- **Clients deal with objects or object references**
  - Hide all cache/persistency specific details
- **No compromise on transient data representation due to technology details**
  - Each technology can be handled transparently
  - Transient representation sufficient for persistency
  - Ensure independence of experiment framework

- **Physics software should be independent of the underlying data storage technology**

File Catalog-implementation

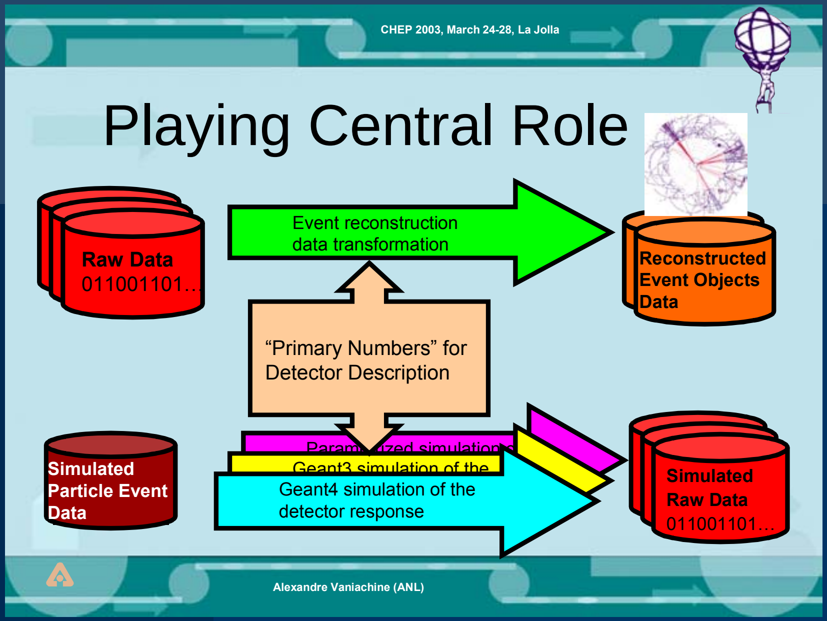
- XML catalog
  - disconnected
  - ~ 20K entries
- MySQL catalog
  - local cluster
  - ~ 1M - 10M entries
- EDG-RLS based catalog
  - on the grid
  - large…

March 24-03 Zhen Xie, Princeton/CERN 8

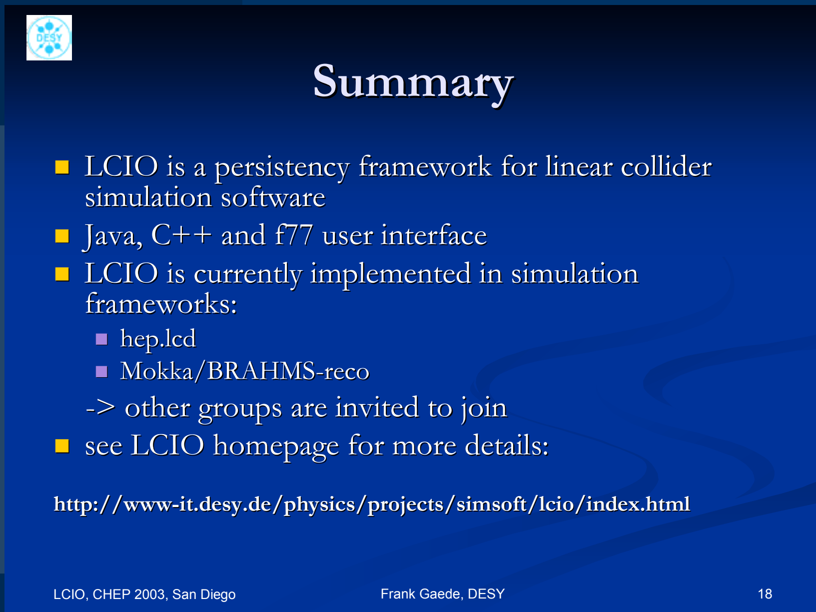
Hiding persistency is de facto standard now

“Primary Numbers” Database for ATLAS Detector Description Parameters

A. Vaniachine



- MySQL based
- Structure for parameters, names, values and attribute metadata (units, comments, …)
- Treat geometry as virtual data, transformation applied to primary numbers



Summary

- LCIO is a persistency framework for linear collider simulation software
- Java, C++ and f77 user interface
- LCIO is currently implemented in simulation frameworks:
  - hep.lcd
  - Mokka/BRAHMS-reco

  -> other groups are invited to join
- see LCIO homepage for more details:

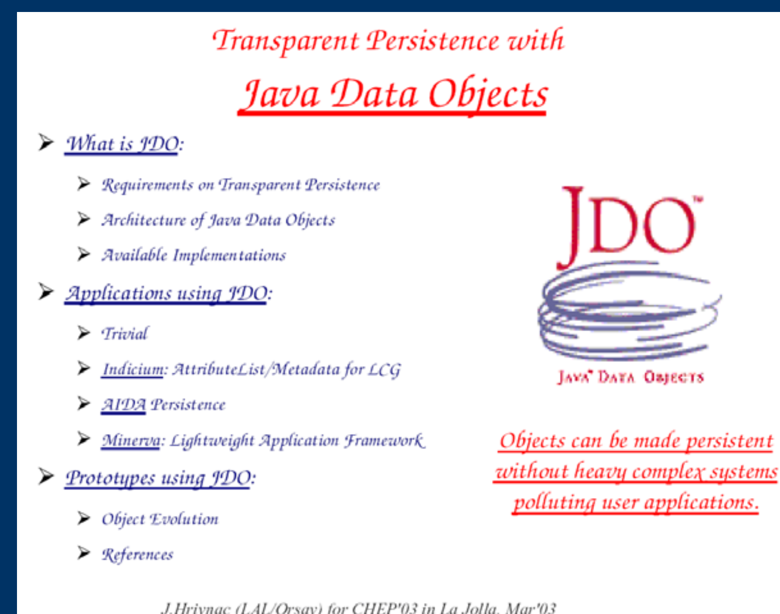
http://www-it.desy.de/physics/projects/simsoft/lcio/index.html

LCIO, CHEP 2003, San Diego   Frank Gaede, DESY   18

Users have to agree on interfaces

Use XML to document data

Prototyping POOL collections and metadata in Java



Transparent Persistence with

Java Data Objects

- *What is JDO:*
  - *Requirements on Transparent Persistence*
  - *Architecture of Java Data Objects*
  - *Available Implementations*
- *Applications using JDO:*
  - *Trivial*
  - *Indicium: AttributeList/Metadata for LCG*
  - *AIDA Persistence*
  - *Minerva: Lightweight Application Framework*
- *Prototypes using JDO:*
  - *Object Evolution*
  - *References*

JDO

JAVA DATA OBJECTS

Objects can be made persistent without heavy complex systems polluting user applications.

J.Hrivnac (LAL/Orsay) for CHEP'03 in La Jolla, Mar'03

- Persistence—saving and restoring object states—is a minimalist view: it is necessary, but is it sufficient?

provenance can be almost fractal in its complexity

Event collections, events, event components, constants to produce them, and finer and finer…

Other emerging ideas

- Current U.S. ITR proposal is promoting knowledge management in support of dynamic workspaces
- One interesting aspect of this proposal is in the area of ontologies
  - An old term in philosophy (cf. Kant), a well-known concept in the (textual) information retrieval literature, and a hot topic for semantic web folks
  - Can be useful when different groups define their own metadata, using similar terms with similar meanings, but not identical terms with identical meanings
  - Could also be useful in defining what is meant, for example, by “Calorimeter data,” without simply enumerating the qualifying classes

David M. Malon, ANL CHEP'03, San Diego 24 March 2003 23

The metadata muddle

Widely varying sources, hard to integrate and query in consistent way

Outline

- Organizational notes
- Online/calibrations/conditions
- Reports from running experiments
- Transitions
- New development, emerging ideas, future
- **Software at a glance**
- **Summary**

Software at a Glance

Event Store

- Objy
  - BaBar, PHENIX, CLEO
  - BaBar's Event Store being migrated to ROOT I/O
  - Technically capable
- ROOT I/O
  - D0, CDF, current mainstream for LHC
  - Missing features augmented by POOL and ROOT team

Metadata

- MySQL
  - Very popular
  - Lightweight, now supports transactions
- PostgreSQL
  - PHENIX, Alice
  - ACID, lightweight, listen/notify
- Oracle
  - COMPASS, Alice, SAM, BaBar
  - For some too expensive

Summary

- Technology transitions
- Heard many more redesign talks than design talks
- Clear preference for open source
- Layered approach to reduce dependency on specific persistency technologies
- LHC experiments collaborating on a common solution (POOL)
  - perhaps BaBar as well

THANK YOU to all session 8 speakers