New developments on the LHCb Bookkeeping

By

Elisa Lanciotti (CERN),

Zoltan Mathe (UCD), Marianne Bargiotti (CERN), Birger Koblitz (CERN),

Andrew Maier (CERN), Roberto Santinelli (CERN)

3rd EGEE User Forum

11-14 February, Clermont – Ferrand

Contents

- What is the Bookkeeping (BK) and how it works
- Some issues of the current implementation
- Objective of this activity: a restructuring and reorganization of the BK
  - New client for data search functionality
  - New database schema
  - Implementation of the BK inside Dirac
- Conclusions

The LHCb Bookkeeping

It is a job and file meta-data management system

Goal:

- Storing all kind of information about files and jobs in an organized way

- Provide to final users an efficient way to get a collection of datasets for the analysis, making queries on the basis of the files and jobs meta-data

Structure of the Bookkeeping

LHCb BK package: a set of services which implement the BK functionality

BKReceiver: Receives data in XML format from DIRAC

BKManager: Process data and insert them in the DB

tomcat: servlets for BKManager and web page UI

gencatalog: search for replicas

DIRAC: LHCb WMS

Production of jobs

XML: job and output files meta-data

Web page User Interface

AMGA: interface to Oracle back-end

Oracle back-end: meta-data storage

DB: Two schema strategy

A set of tables in a relational DB schema: **warehouse tables (WDB)**

- To store meta-data of files and jobs

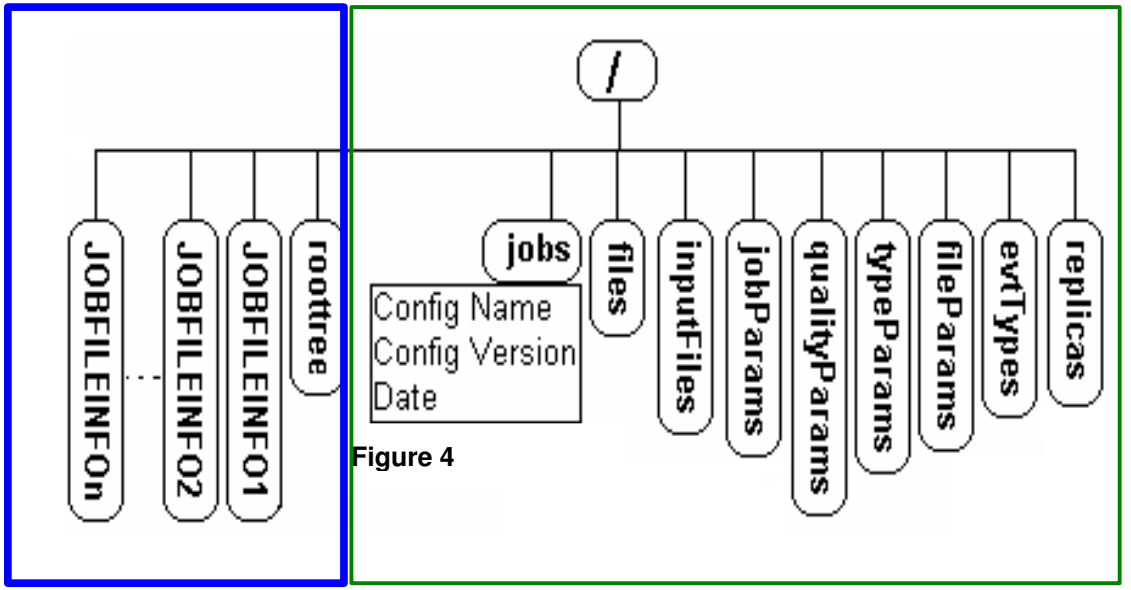


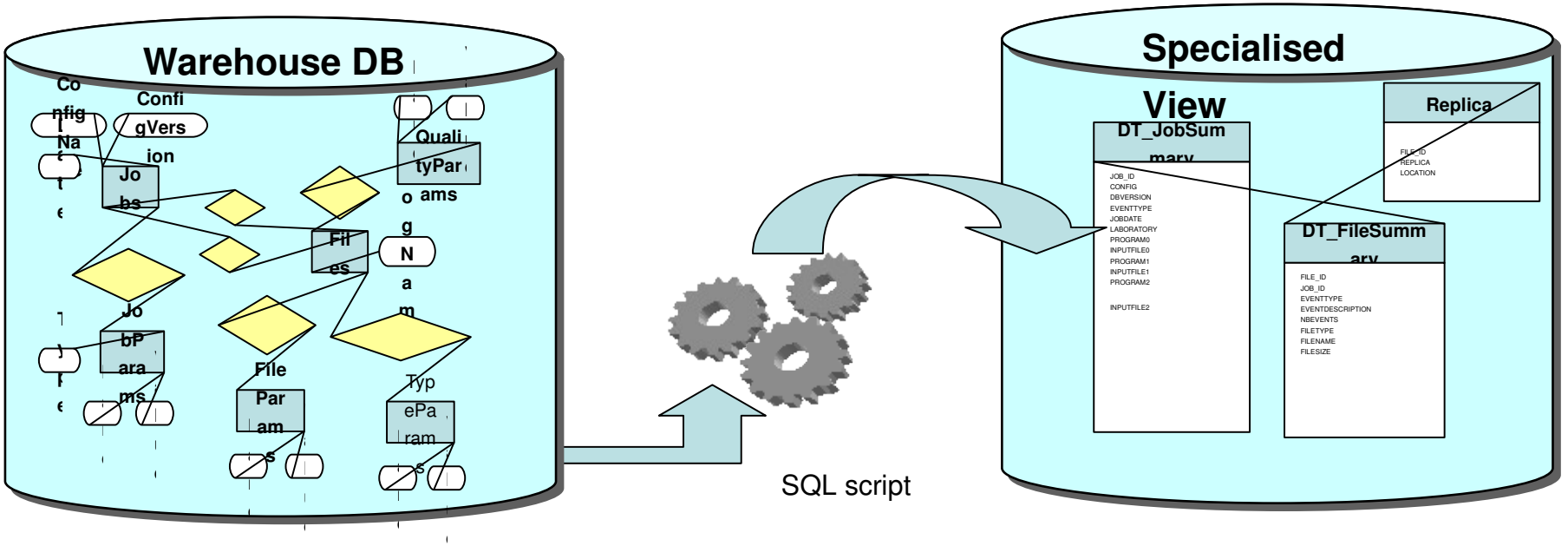
Figure 4

- **Materialized views**
  - Users queries are addressed to this tables

Main entities:

- **Files**: name, date, size, quality parameters, etc..
- **Jobs**: date, production, etc..

Generation of the specialized views



- Update each night. Low resources required from Oracle back-end, despite that the WDB contains about 50 GB
- Provide the best performance to fulfill complex queries

Main issues from the user point of view

Accessibility:

- Access failures are common
- Retrieving a list of LFN is ok. But if one asks for replicas at a site, the request has to be submitted several times (only 200 file processed at a time)
- The user interface web page has some broken functionality

Data information:

- Too few selection criteria: only few predefined attributes can be used (BK has been used so far only for simulated data)

Objective of this activity

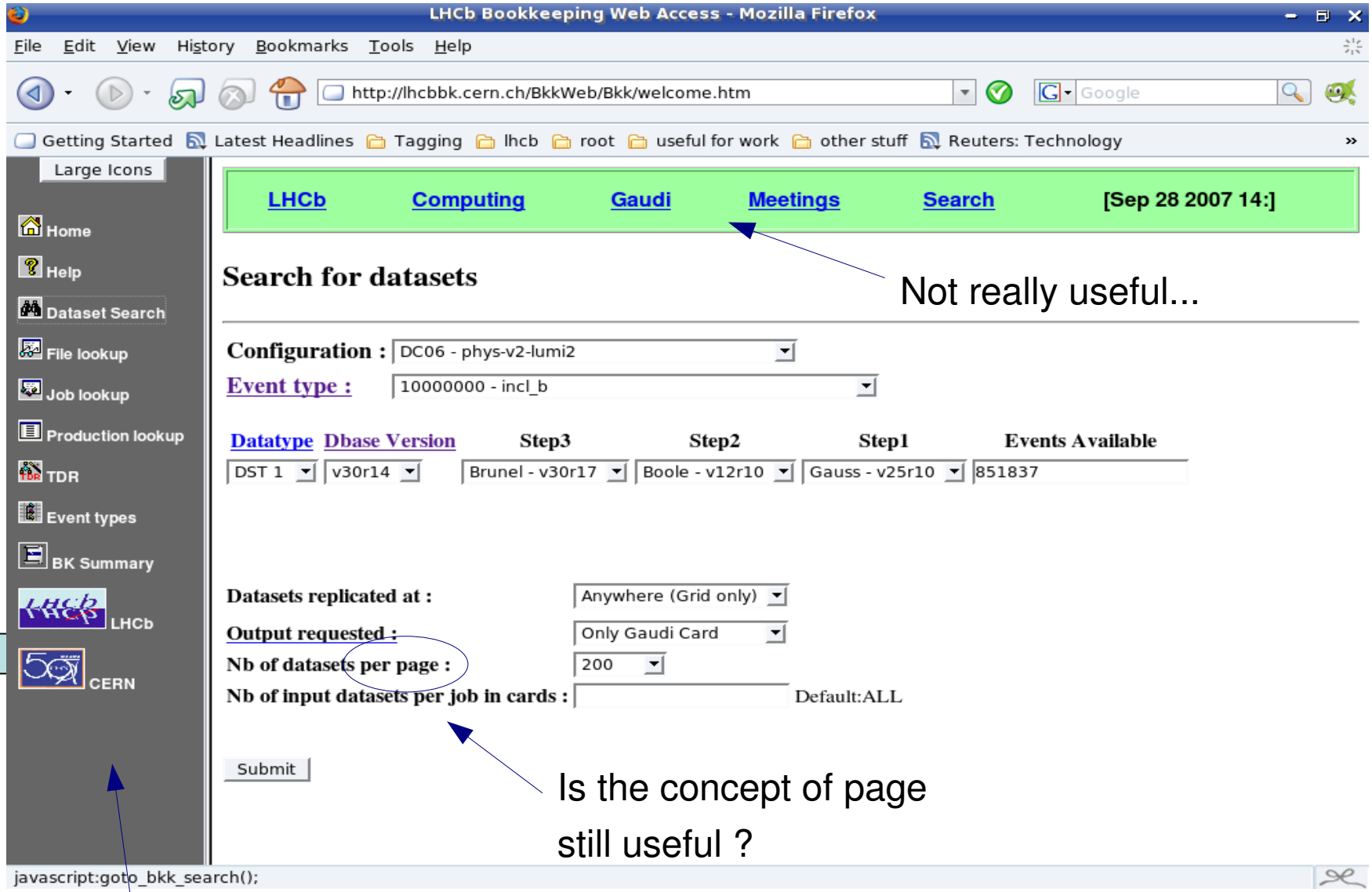
To fulfill the user requirements:

- More stability and reliability of the service
- Data available from any search field, and in any order
- Access to all PFN in one site submitting only one request. Output provided in a text file

And in addition to that, further restructuring is needed:

- Adapt the DB schema to store real data files meta-data
- Migrate the code from Java to python and integrate it into DIRAC framework

Bookkeeping data search functionality



New python client for data search

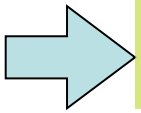
Prototype of a new client for data search functionality:

- Same functionality of the web page UI, but rewritten from scratch
- Based on the python AMGA API: it's faster and easier to support
- More flexible: query built starting from whatever attribute the user wishes (in the web page the user must first choose the CONFIG, then the EVTTYPE, FILETYPE and so on)
- The file list is provided in a text file which can be included in the options file of the analysis job. The output, even when the user selects a site, is provided in one go.
- Need to develop a GUI for it!

BK Schema: New DB schema needed

Objective:

- Eliminate from tables the attributes that are obsolete
- Adapt the DB schema to the forthcoming data taking: the concept of 'job' means 'run' for real data.
- Add new attributes relevant for real data runs: input from the experiment
- Optimize the design of the DB schema



Implemented a prototype on a development instance of the BK

The new schema

- New attributes relevant for runs:*start time, end time, beam luminosity, beam energy, etc*...

New tables for runs:

Processing pass: a set of application versions and condDB tags

To be inserted in the views to select runs

Data taking period: set of runs that can be mixed at processing stage. They must have similar conditions: beam, detector stability, trigger conditions

Processing pass table

- Application Version
- CondDB tag

Data taking period table

- Beam energy
- Luminosity
- Trigger conditions
- etc..

- Views redefined on the basis of the typical queries for real data

Implementation of the BK service inside DIRAC

- DIRAC is the LHCb Workload and Data Management System. It also includes other services for job monitoring, job accounting integrated in the same framework
- The BK is implemented as a new service inside DIRAC3: BookkeepingManager service

Advantages:

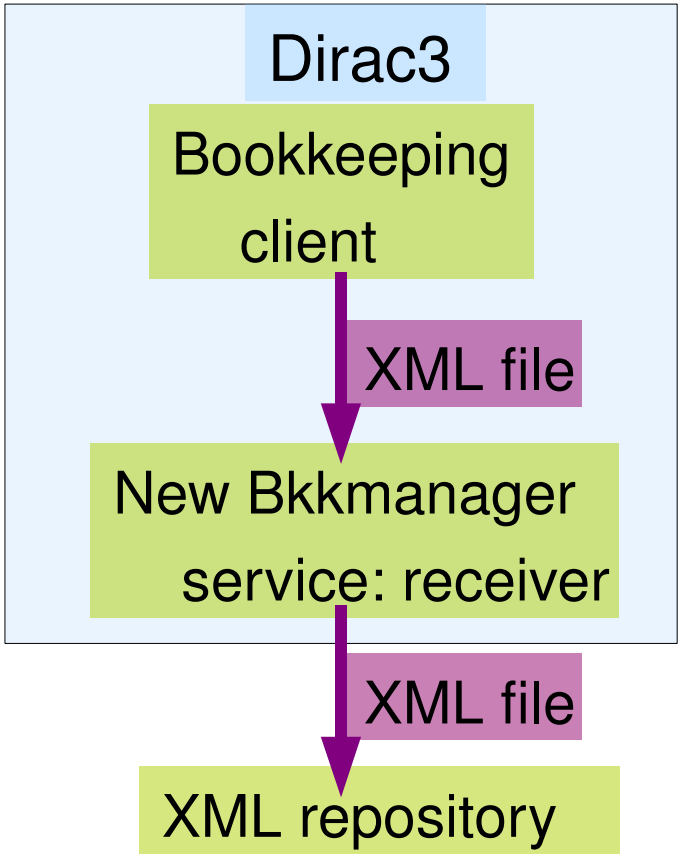
- All the services are synchronized (sender and receiver)
- Implemented in python: easier to support
- All the code organized in same CVS repository: more portable
- Optimization of the functionality

A new DIRAC service:BookkeepingManager

So far: only the functionality of the receiver has been implemented into the BookkeepingManager service

Next to do:

implement all the BK functionality into DIRAC3 framework, starting from the old BKManager: reads the content of XML files and insert them in the database



Conclusions

- Presentation of the LHCb Bookkeeping
- Need of some restructuring of the LHCb BK

So far:

- Prototype of a new client for data search
- New DB schema proposal
- The service of the receiver implemented in the DIRAC framework

Next to do:

- Implement the BK Manager into DIRAC3 framework
- Test the performance with some data to tune the new DB schema and migrate the data from the old schema
- Develop a GUI