Data Management in LHCb: consistency, integrity and coherence of data

Marianne Bargiotti
CERN

Outline

- Brief overview of DIRAC, LHCb grid project
- DIRAC Data Management System
  - File Catalogues, Storage Element, Replica Manager
- Data integrity checks and sources of data inconsistency both on LCG File Catalog (LFC), BK Metadata Catalog and SEs.
- Monitoring of problems and running agent
- What has been achieved & future plans

Introduction

- Main elements of the LHCb Data Management are:
  - BK meta data Data Base
  - LCG File Catalog (LFC)
  - Storage Elements (SE)
- Central topic is how to keep these elements in a consistent state
- These checks have as final aim the production of coherent pictures among catalogs and storages in order to prevent any data loss or corruption.

DIRAC Architecture

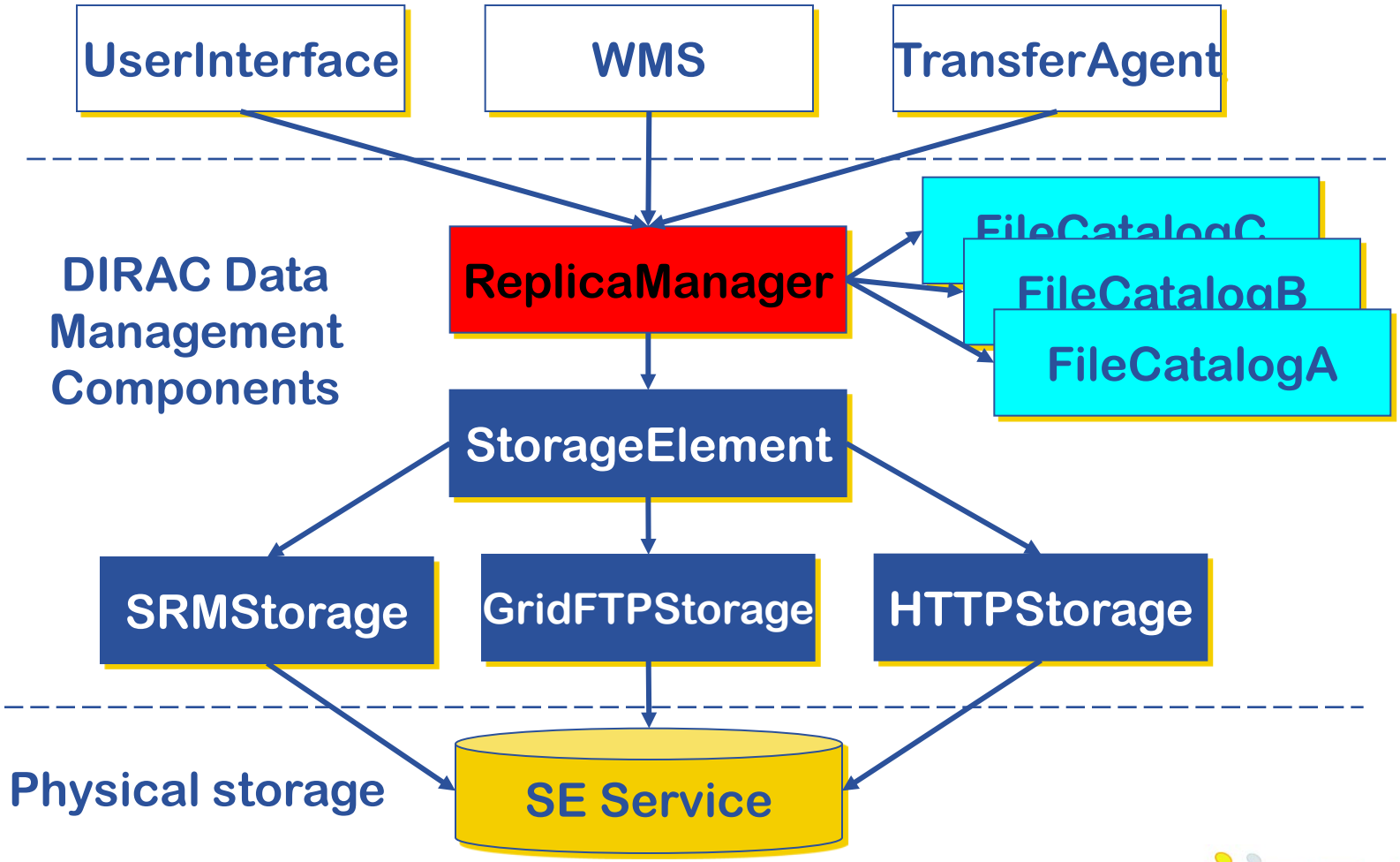
- DIRAC (Distributed Infrastructure with Remote Agent Control) is the LHCb's grid project
- DIRAC architecture split into three main component types:
  - Services - independent functionalities deployed and administered centrally on machines accessible by all other DIRAC components
  - Resources - GRID compute and storage resources at remote sites
  - Agents - lightweight software components that request jobs from the central Services for a specific purpose.
- The DIRAC Data Management System is made up an assortment of these components.

DIRAC DM System

- **Main components of the DIRAC Data Management System:**
  - **Storage Element**
    - abstraction of GRID storage resources: Grid SE (also Storage Element) is the underlying resource used
    - actual access by specific plug-ins
    - srm, gridftp, bbftp, sftp, http supported
    - namespace management, file up/download, deletion etc.
  - **Replica Manager**
    - provides an API for the available data management operations
    - point of contact for users of data management systems
    - removes direct operation with Storage Element and File Catalogs
    - uploading/downloading file to/from GRID SE, replication of files, file registration, file removal
  - **File Catalog**
    - standard API exposed for variety of available catalogs
    - allows redundancy across several catalogs

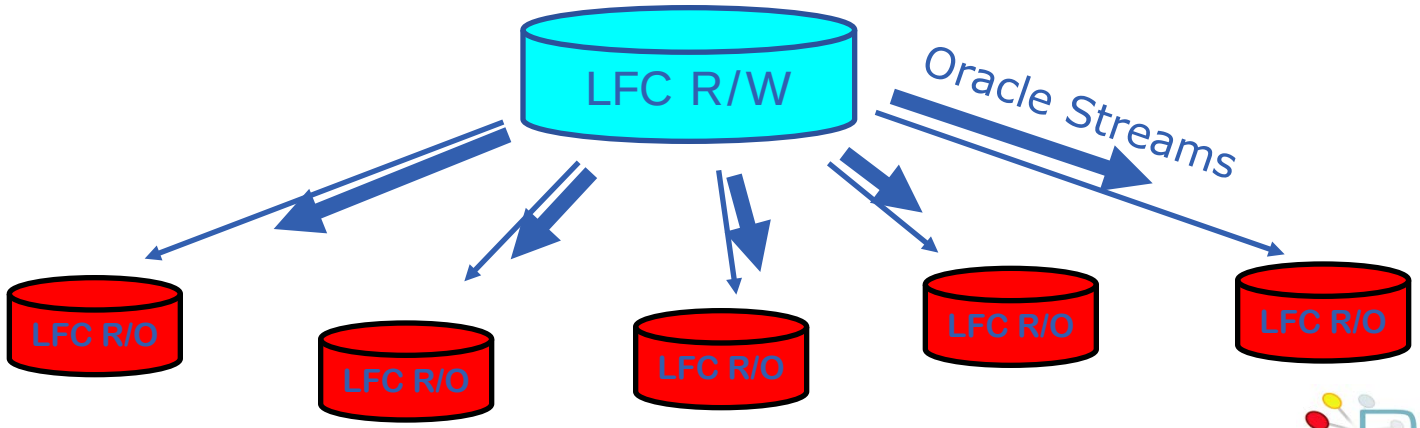
DM Clients

Data Management Clients



DIRAC Catalog

- DIRAC Data Management was designed to work with multiple File Catalogs
- Baseline choice:
  - Central LFC with multiple mirror instances
    - One single master (R/W) and many R/O mirrors
    - coherence ensured by single write endpoint
    - redundancy and load balancing ensured by multiple mirrors
    - real time updates with Oracle Streams



Data integrity checks

- Considering the high number of interaction among DIRAC and Catalogues, integrity checking is an integral part of the Data Management system to ensure the knowledge and accessibility of data consuming LHCb resources.
- Two types of check exist:
  - those running as Agents within the DIRAC framework
  - those launched by the Data Manager to address specific situations.
- The Agent type of checks can be broken into two further distinct types.
  - Those solely based on the information found on SE/LFC/BK
  - those based on a priori knowledge of where files should exist based on the Computing Model (i.e. DST always present at all T1's disks)
    - The former type would include the following three checks:
      - BK->LFC,
      - LFC->SE,
      - SE->LFC and could also provide information summarizing the use of LHCb storage resources.

Data integrity checks & DM Console

- **The Data Manager launched checks have been developed in a rolling fashion (as per request) and are accessible through the Data Management Console interface.**
  - **The role of the Data Management Console is the portal through which the Data Manager performs routine activities. Currently the Console supports the creations of bulk transfer/removal jobs and many catalog operations (fixes)**
  - **The Console will allow launching of integrity checks.**

LHCb and DIRAC naming conventions

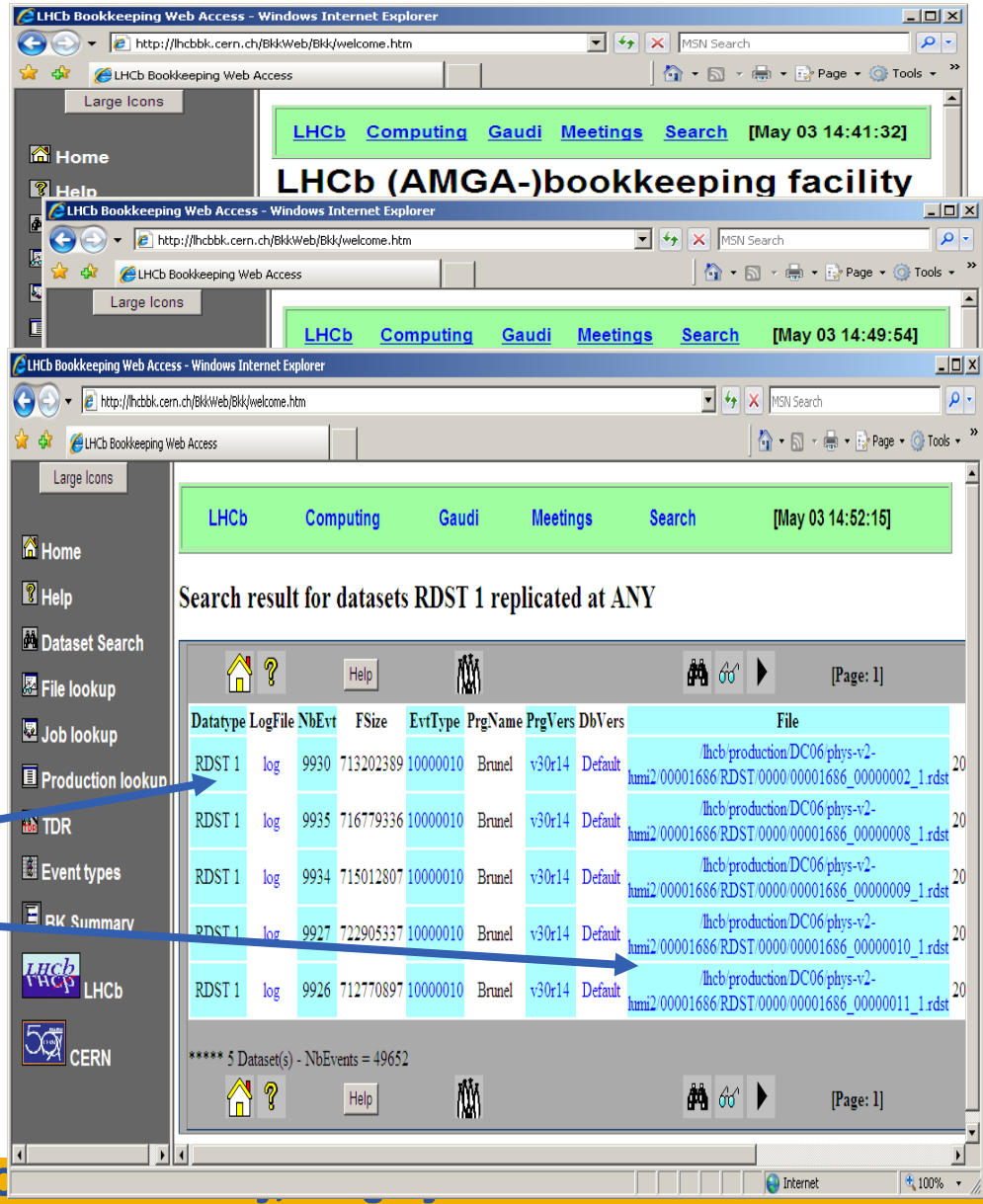
- **LHCb file naming convention:**
  - Currently all the production's Lfns registered on catalog are of this kind:
    - /lhcb/production/…/ProdID/DIGI/… TAPE
    - /lhcb/production/…/ProdID/RDST/… DISK
  - The pfn structure is the following:
    - srm://SEHost/SAPath/lfn
      - where the SEHost and SAPath are defined in the DIRAC Configuration System (CS)
      - the lfn part reflects the VO specific side
- **Having strict convention helps integrity checks as explained after**

LHCb Bookkeeping Meta Data

- The Bookkeeping (BK) is the system that manages the meta-data infos (file-metadata) of data files. It contains information about jobs and files and their relations:
  - Job: Application name, Application version, Application parameters, which files it has generated etc..
  - File: size, event, filename, from which job it was generated etc.
- Three main services are available:
  - Booking service: to write data to the bookkeeping
  - Servlets service: for the BK web browsing and selection of the data files.
  - AMGA server: for remote application use.

LHCb Bookkeeping Web Page

- The web page allows to browse the bookkeeping contents and get information about file and their provenance.
- It is also used to generated Gaudi Card, the list of files to be processed by a job.
- On left frame links many browsing options: File look-up, Job look-up, Production look-up, BK summary
  - Dataset search: retrieve a list of files based on their provenance history.
- The result is:



BK problems

- **BK documented problems:**
  - **many missing lfns registred on BK but failed to be registred on LFC**
    - **missing files in the LFC: users trying to select lfns in the BK can't find their traces in the LFC**
      - **Failing of registration on the LFC due to failure on copy, temporary lack of service…**
  - **Integration of the bookkeeping into DIRAC requires that when a file is completely removed from the grid a message must be sent to flag this file as having no replicas.**
    - **The file catalogue plug-in for the Bookkeeping implements this functionality.**
  - **Integrity checks require access to large numbers of entries**
    - **therefore need for efficient Access Interface**

BK vs LFC

- **BK→LFC: massive check on productions:**
  - data from an extensive BK dump.
  - checking from BK dumps of different productions against same directories on LFC
  - for each production:
    - checking for the existence of each entry from BK against LFC
    - check on file sizes
    - if file not present on LFC, log file issued identified by production id

    → **update of LFC with missing file infos**

LFC registration of replicas

- The DIRAC registration of files on the LGC File Catalog can be divided into 2 different separate atomic operations
  - booking of meta data fields with the insertion in the dedicated table of lfn, guid and size (put to zero at the starting point)
  - replica registration once the transfer is over and update of metaData fields with actual values

→ greater control with single steps

→ but on the other side possible source of errors and inconsistencies: in fact if the second step fails the file is registered without any replica or with zero size.

LFC documented problems

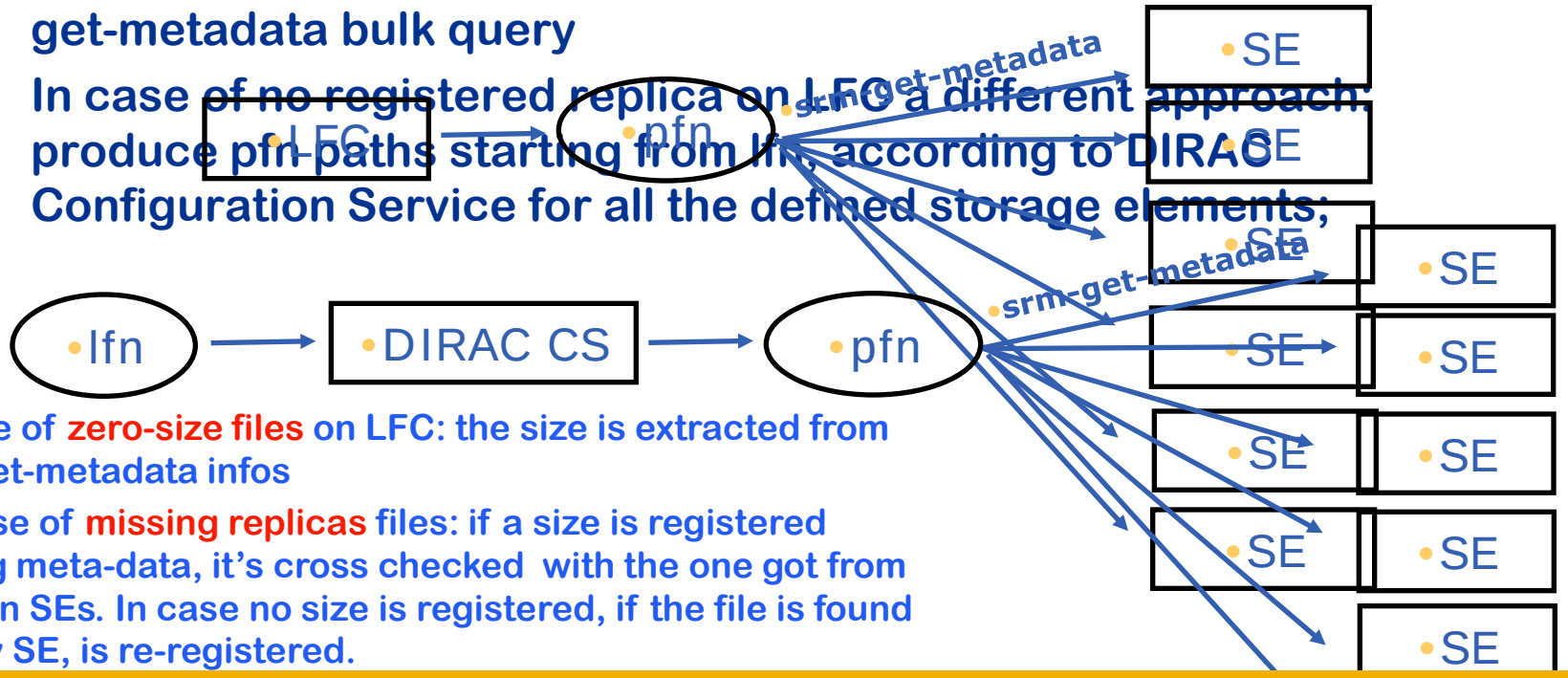
- **LHCb specific problems:**
  - **zero size files:**
    - file metadata registred on LFC but missing information on size (set to 0)
  - **missing replica files:**
    - missing replica field in the Replica Information Table on the DB
  - **wrong SAPath:**
    - srm://gridka-dCache.fzk.de:8443/castor/cern.ch/grid/lhcb/production/DC06/v1-lumi2/00001354/DIGI/0000/00001354_00000027_9.digi GRIDKA-tape
  - **wrong SE host:**
    - CERN_Castor, wrong info in the LHCb CS
  - **wrong protocol**
    - sfn, rfio, bbftp…
  - **mistakes in files registration**
    - blank spaces on the surl path, carriage returns, presence of port number in the surl path..

Catalog requirements

- **File catalogue requirements:**
  - efficient access to large amount of data
  - this needs special Administration Interface (not used by users)
    - bulk replica queries
    - bulk modification of replicas
    - bulk deletion of files and replicas
    - deletion of LFC directories

LFC vs SE

- **What to do: check whether the LFC replicas are really resident on Physical storages too (check the existence and the size of files):**
  - **extract the pfns from the catalog and check vs the SEs through srm-get-metadata bulk query**
  - **In case of no registered replica on LFC a different approach: produce pfn-paths starting from lfn, according to DIRAC Configuration Service for all the defined storage elements;**



- in case of zero-size files on LFC: the size is extracted from srm-get-metadata infos
- in case of missing replicas files: if a size is registered among meta-data, it's cross checked with the one got from SRM on SEs. In case no size is registered, if the file is found on any SE, is re-registered.

Many possible causes for data missing or corruption:

- dCache bug: replicas registered in the dCache file system but not on disk pools (3718 lost files at PIC)
- robot tape overwriting existing tapes (2687 files lost at RAL)

LFC vs SE Agent

- For the LFC->SE checks an agent has been implemented.
  - Checks performed:
    - zero size files
    - zero replica files
    - wrong or obsolete protocol
  - The agent loops continuously over the LFC directory structure, checking file by file and comparing with srm query results from Storages. All the statistics and the repair status are stored in a dedicated MySql DB
  - Some statistics (latest hours):

	Total	Repaired
Zero Size	15344	15074
Zero Replicas	70865	--
No SRM paths	117187	--

SE vs LFC

- **SE contents against catalogues**
  - List the contents of the SE
  - Check against the catalogue for corresponding replicas
    - Possible because of file naming conventions
    - Files paths on SE always 'SEhost/SAPath/LFN'
  - Files missing from the catalogue can be
    - Re-registered in catalogue
    - Deleted from SE
    - Depending on file properties
  - Missing efficient Storage Interface for bulk operations and for getting meta data

    → Waiting for SRM v2!

Prevention of inconsistencies

- **Failover mechanism:**
  - each operation that can fail can be wrapped in a XML record as a request which can be stored in a DB.
  - These DB are sitting in the LHCb VO Boxes, which ensures that these records will never be lost
  - These requests are executed by dedicated agents running on VO Boxes, and will be retried as many time as needed until a definite success
  - Examples: files registration operation, data transfer operation, BK registration…
  - See Andrew C. Smith's talk
- **Many other internal checks are also implemented within the DIRAC architecture to avoid data inconsistencies as much as possible. They include for examples:**
  - checking on file transfers based on file sizes or checksums
  - roll back and retry mechanism for the catalogs registration

Conclusions & Perspectives

- Integrity checks suite is an integrated part of Data Management activity
- Further development will be put in place with SRM v2
- Most of efforts are now in the prevention of inconsistencies (checksums, failover mechanisms…)
- Final target: minimizing the number of occurrences of frustrated users looking for non-existing data.

Back up

Storage Element

- DIRAC StorageElement is an abstraction of a Storage facility
  - Access to storage is provided by plug-in modules for each available access protocol.
  - Pluggable transport modules: srm, gridftp, bbftp, sftp, http,…
- Storage Element is used mostly to get access to the files
- Grid SE (also Storage Element) is the underlying resource used

Replica Manager

- **Replica Manager provides logic for all data management operations**
  - File upload/download to/from Grid
  - File replication across SEs
  - Registration in catalogs
  - etc.
- **Keeps a list of active File Catalogs**
  - All registrations applied to all catalogues