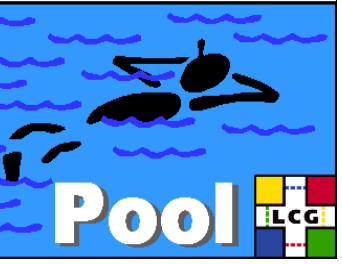


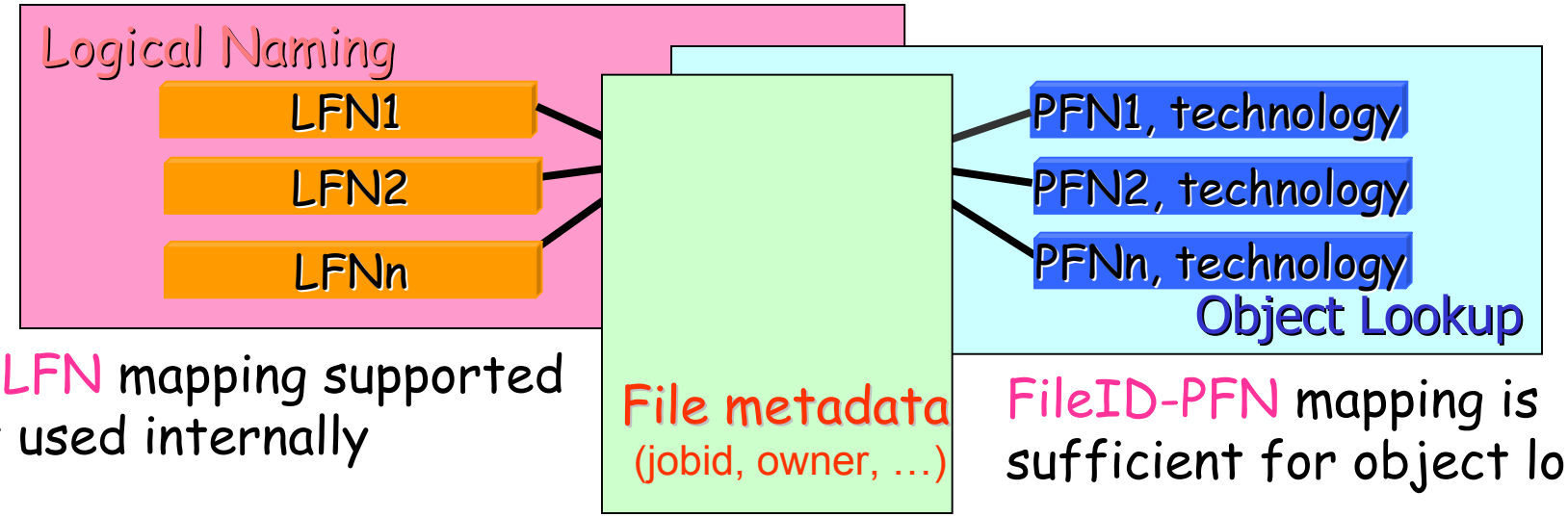
File Catalog and Grid Integration

Maria Girone
IT/DB & LCG/POOL

- Component purpose and design
- Implementations and Functionality
- Use Cases and Performance

Component purpose

- Maintaining a consistent list of accessible files (physical and logical names) together with their unique identifiers
- Files are referred to inside POOL via a unique and immutable file identifier, (FileID) generated at creation time
- POOL added the system generated FileID to the standard Grid m-n mapping
  - Stable inter-file reference
- Optionally, File metadata attributes are associated with the FileID, used for fas query based catalog manipulation. Not used internally by POOL



FileID-LFN mapping supported
out not used internally

FileID-PFN mapping is sufficient for object lookup

GUIDs as File Identifiers

- Global Unique Identifier (GUID) implementation for FileID
  - allows the production of a consistent sets of files with internal references without requiring a central ID allocation service
  - Jobs can execute in complete isolation

- Catalog Fragments can be used asynchronously as a means of propagating catalog or meta data changes through the system
  - Without loosing the internal object references
  - Without risking to clash with any already existing objects
  - Without the need to rewrite any existing data files

Concrete implementations

- XML Catalog
  - typically used as local file by a single user/process at a time
  - no need for network
  - supports R/O operations via http
  - tested up to 50K entries
- Native MySQL Catalog
  - Shared catalog eg in a production LAN
  - handles multiple users and jobs (multi-threaded)
  - tested up to 1M entries
- EDG-RLS Catalog
  - Grid aware applications
  - production service based on Oracle (from CERN IT/DB), already in use for POOL V1.0 (july '03)
  - Oracle iAS or Tomcat + Oracle / MySQL backend

File Catalog Functionality

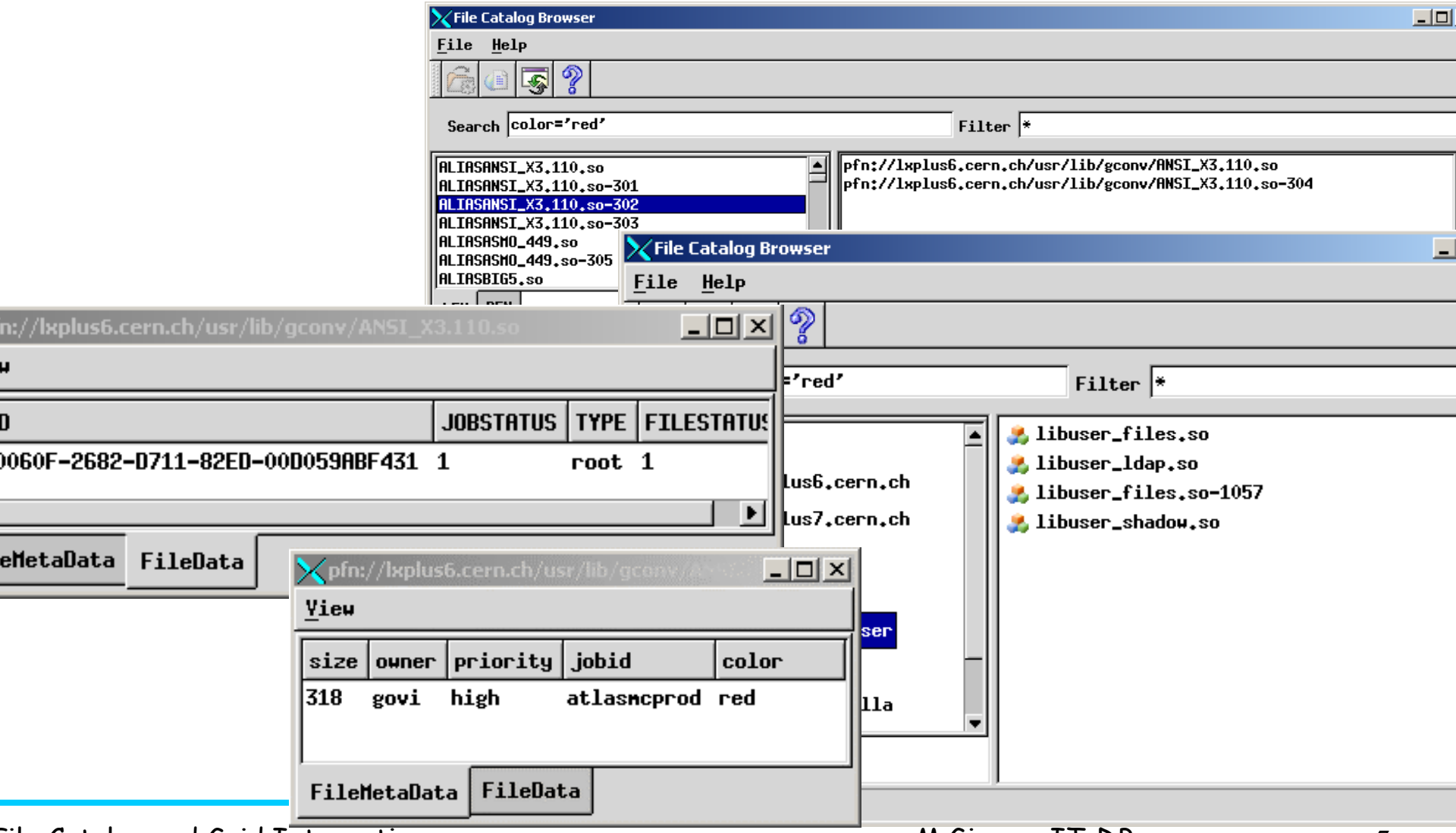
- Catalog API provides
  - insert and update functions on logical and physical filenames
  - lookup functions by filename, FileID or query
  - File Meta data operations (e.g. define or insert meta data describing a file)
- Catolog Administration (eg via command line tools or Python binding)
  - Eg extract a XML fragment and import it into a MySQL catalog
- Changes under transactional control functions
- Selecting on Implementation at connection time
  - XML
    - xmlcatalog_file:/tmp/FileCatalog.xml or file:/tmp/FileCatalog.xml
    - xmlcatalog_http://pc01.cern.ch/file001 (R/O)
  - MySQL
    - mysqlcatalog_mysql://@lxshare070d.cern.ch:3306/testFCdb
  - EDG
    - edgcatalog_http://rlstest.cern.ch:7777/test/edg-local-replica-catalog/services/edg-local-replica-catalog

Catalog Queries

- Application level meta-data can be associated to cataloged files
  - meta-data are normally used to query large file catalogs, e.g. extract catalog subsets based on production parameters
- Catalog fragments can be shipped to a different site or to a decoupled production node. Cross-catalog operations between different implementations are also supported
- Queries are supported at the client side for XML and EDG and at the server side for MySQL (at EDG server side will come in POOL V1.5)
  - "jobid='cmssim' ", "owner like '%wild%' "
  - "pfname like 'path' and owner = 'atlasprod' "
- Evaluation of numerical query expressions not yet fully supported (backend enhancements in progress)

File Catalog Browser Prototype

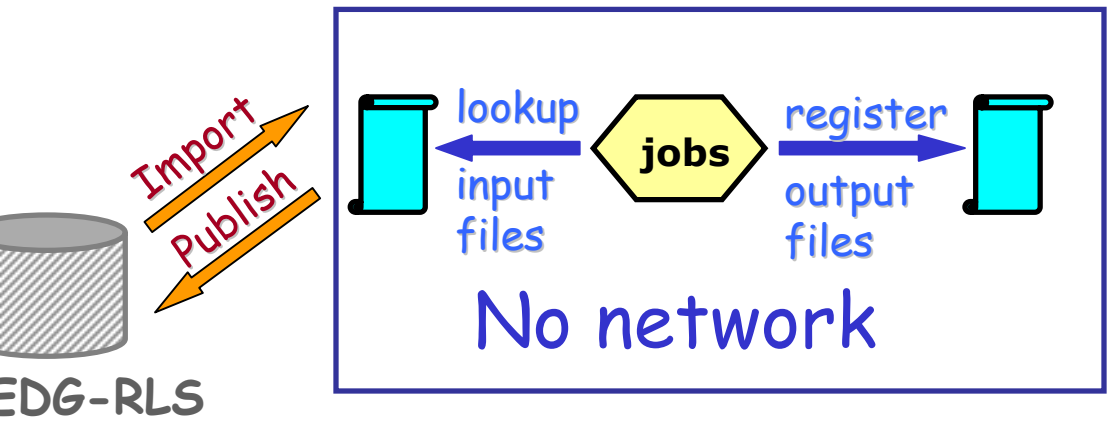
- Python based graphic user interface for the catalog browsing



The File Catalog and the EDG-RLS

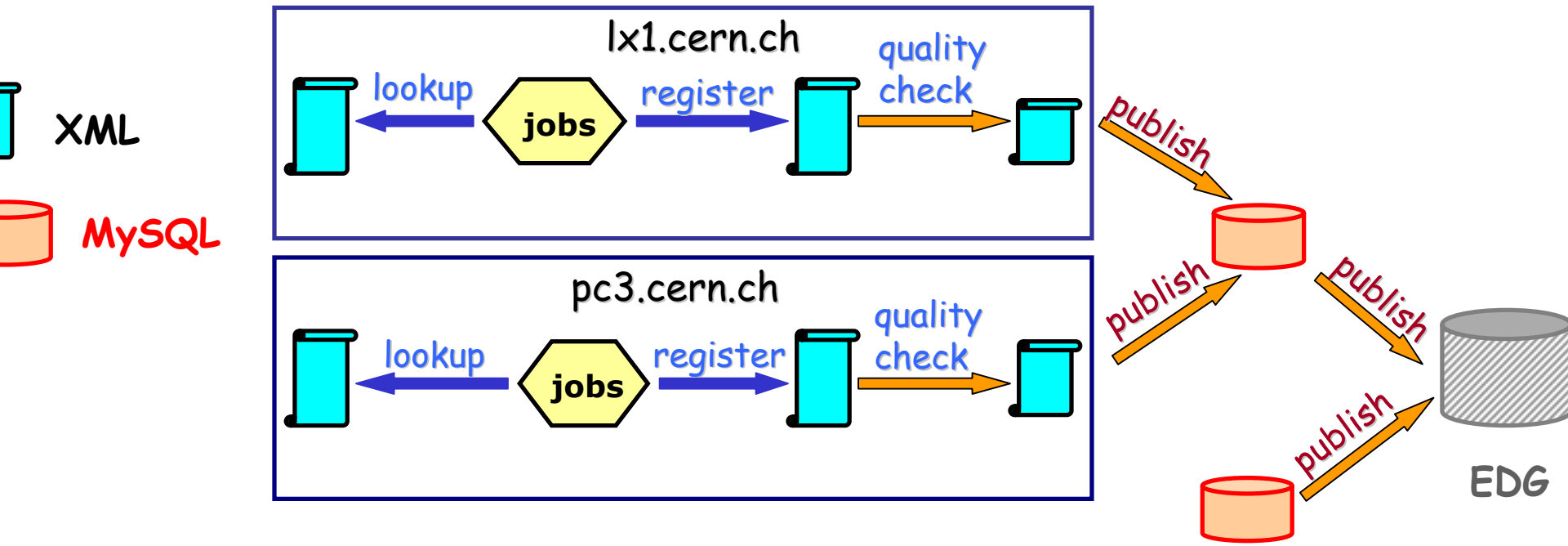
- The Replica Location Service is a critical component of the Grid
  - Job scheduling, data access from running jobs need it
- Grid services relevant to the File Catalog
  - A "Local Replica Catalog" (LRC)
    - Contains GUID <-> PFN mapping for all local files
  - A "Replica Metadata Catalog" (RMC)
    - For file-level meta-data
  - Eventually, also a "Replica Location Index" (RLI)
    - Allows files at other sites to be found
    - All LRCs are configured to publish to all remote RLIs

Use case: isolated system



- The user extracts a set of interesting files and a catalog fragment describing them from a (central) Grid based catalog into a local XML catalog
  - Selection is performed based on file or collection descriptions
- After disconnecting from the Grid the user executes some standard jobs navigating through the extracted data
  - New output files are registered into the local XML catalog
- Once the new data is ready for publishing and the user is connected the new catalog fragment is submitted to the Grid based catalog

Use case: farm production



- A production job runs and creates files and their catalog entries in a local XML file
- During the production the catalog can be used to cleanup files
- Once the data quality checks have been passed the production manager decides to publishes the production XML catalog fragment to the site database one and eventually to the Grid based catalog

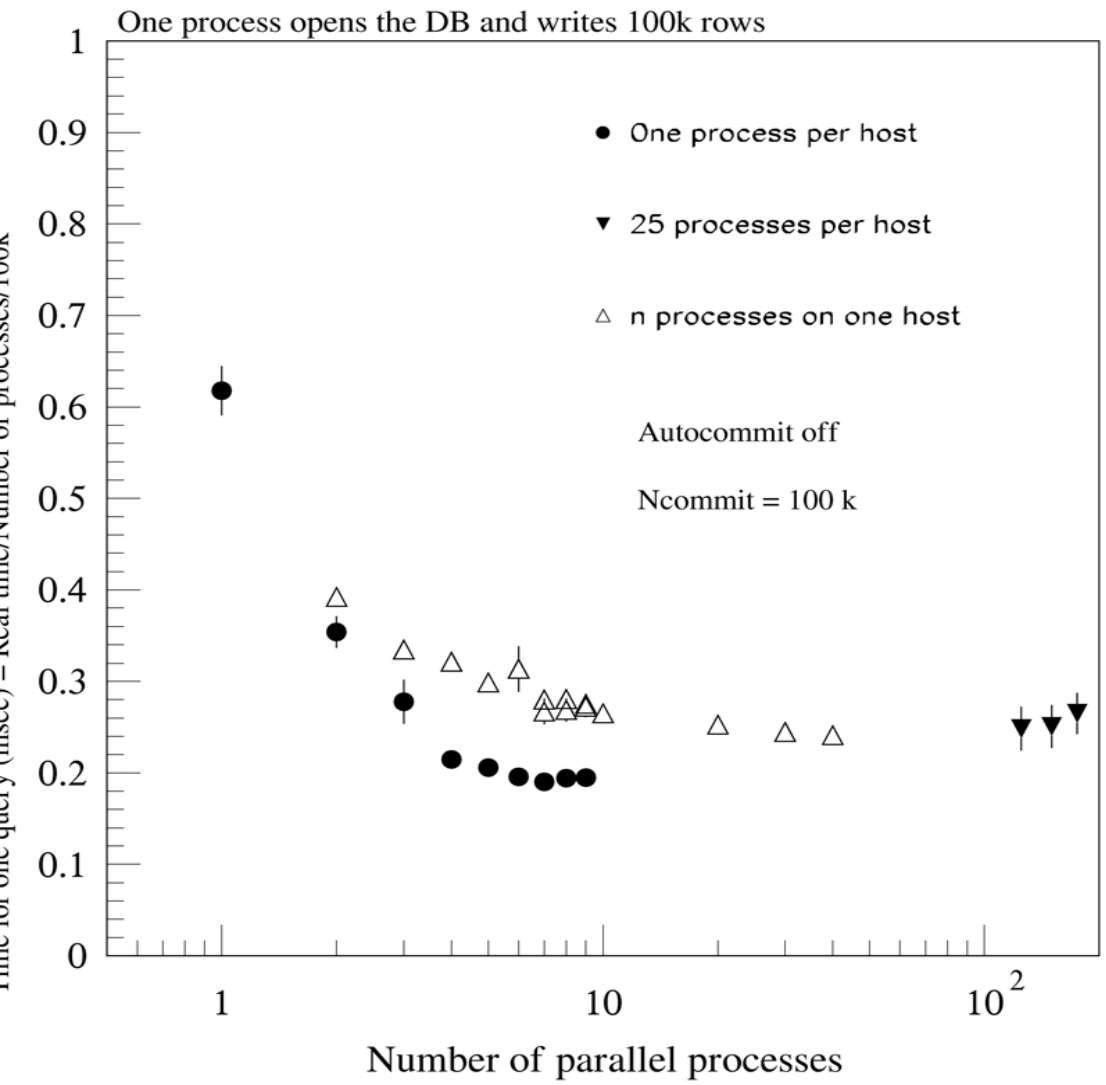
File Catalog performance tests

- Preliminary tests done on POOL V0.5

Pentium III-1.2GHz
free memory-220MB
PFN-200 char; FileID-36 char

- XML: tested up to 50K entries
  - start time:
    - new catalog ~10ms
    - catalog with 20K entries ~6s
  - registerPFN: <0.3ms/entry
- MySQL: tested up to 1M entries
  - up to 300 concurrent clients, commit every 100 entries or less frequent
  - registerPFN: <1.5ms/entry, lookup better than that
- EDG-RLS based catalog: registerPFN: ~35ms/entry, lookupPFN: ~20ms

File Catalog Scaling Tests



- MySQL Catalog
  - update and lookup performances look fine
  - scalability shown up to a few hundred clients
- EDG Catalog
  - Scalability tested up to 100 clients
- LCG-1 expected lookup frequency of ~0.3 Hz
  - Performance well within requirements

Summary

- The File Catalog is a central component of POOL Object Stor
  - Keeps track of a consistent set of POOL database files and associated meta data
- The set of File Catalog implementations provided with POOL satisfies a wide range of experiment requirements
  - POOL client programs can either
    - Transparently make use of grid-wide File Catalogs (EDG-RLS)
    - Or operate without requiring network or central services eg for production activities
  - Performance and scalability within requirements for LCG-1
- EDG RLS Service is available for LCG-1
  - Experiment-specific production services for the EDG Catalog are provided by IT/DB

File Catalog Performance Requirements

- A very preliminary model of the frequency of access to the File Catalog, for
  - Initial LCG-1 service
  - Analysis activities at the LHC start-up
- Strongly based on experiments inputs (mainly CMS), subject to modifications
- It is based on the following assumptions
  - LCG-1 CPU power: 1 GHz, Pentium III (400 SPECInt2000)
  - LHC start-up (2008): total CPU capacity of 20M SPECInt2000 (2M SPECInt2000 at each of 5 T1 centres, with an equal amount shared over all T2s

LCG-1 access figures

- Take CMS PCP as example:
- Total number of events to produce (kine, Simul, Digi and half Reco): 50M [1]

	July	November
– Fraction of expected LCG-1 production [2] (and thus requiring central cataloguing of the files created)	10%	75%

- Based on the length of different job types, it is expected that

– File lookup frequency:	0.05Hz	0.2Hz
– File registration frequency:	0.05Hz	0.1Hz
– Total interaction rate:	0.1Hz	0.3Hz
1 interaction every	10 sec	3 sec

Current tests show performance well in excess of these requirements!

RLS service goals

- To offer **production quality services** for LCG 1
  - Based on Oracle Application Server (1 per VO) and DB
  - 24 x 7 File catalog and file-level metadata services
- Meet the requirements of forthcoming (and current!) data challenges
  - e.g. CMS PCP/DC04, ALICE PDC-3, ATLAS DC2, LHCb CDC'04
- To provide distribution kits, scripts and documentation to **assist** other sites in offering production services
- To leverage the many years' experience in running such services at CERN and other institutes
  - Monitoring, backup & recovery, tuning, capacity planning, …
- To understand experiments' requirements in how these services should be established, extended and clarify **current limitations**