Light weight Disk Pool Manager for LCG2

Ian Bird
Jean-Philippe Baud
James casey
IT-GD, CERN

October 2004

Overview

Introduction

- Recent experience and current thinking gives following strategy for storage access
- SRM is common interface to storage; 3 cases:
  - Integration of large (tape) MSS (at Tier 1 etc) –
    - Responsibility of site to make the integration
  - Large Tier 2's – sites with large disk pools (10's Terabytes, many fileservers), need a flexible system
    - dCache provides a good solution, but needs effort to integrate and manage
  - Sites with smaller disk pools (1 – 10 Terabytes), less available management effort
    - Need a lightweight (install, manage) solution
- Lightweight Disk Pool Manager is complementary to dCache solution in LCG-2

Disk Pool Manager aims

- Provide a solution for the small Tier-2s in LCG-2
  - This implies 1 to 10 Terabytes in 2005
- Focus on manageability
  - Easy to install
  - Easy to configure
  - Low effort for ongoing maintenance
  - Easy to add/remove resources
- Support for multiple physical partitions
  - On one or more disk server nodes
- Support for different space types – volatile and permanent
- Support for multiple replicas of a file within the disk pools

Manageability

- Few daemons to install
  - Disk Pool Manager
  - Name Server
  - SRM
- No central configuration files
  - Disk nodes added dynamically through tools/API
- Easy to remove disks and partitions
  - Allows simple reconfiguration of the Disk Pools
  - Administrator can temporarily remove file systems from the DPM if a disk has crashed and is being repaired
  - DPM automatically configures a file system as “unavailable” when it is not contactable

Features

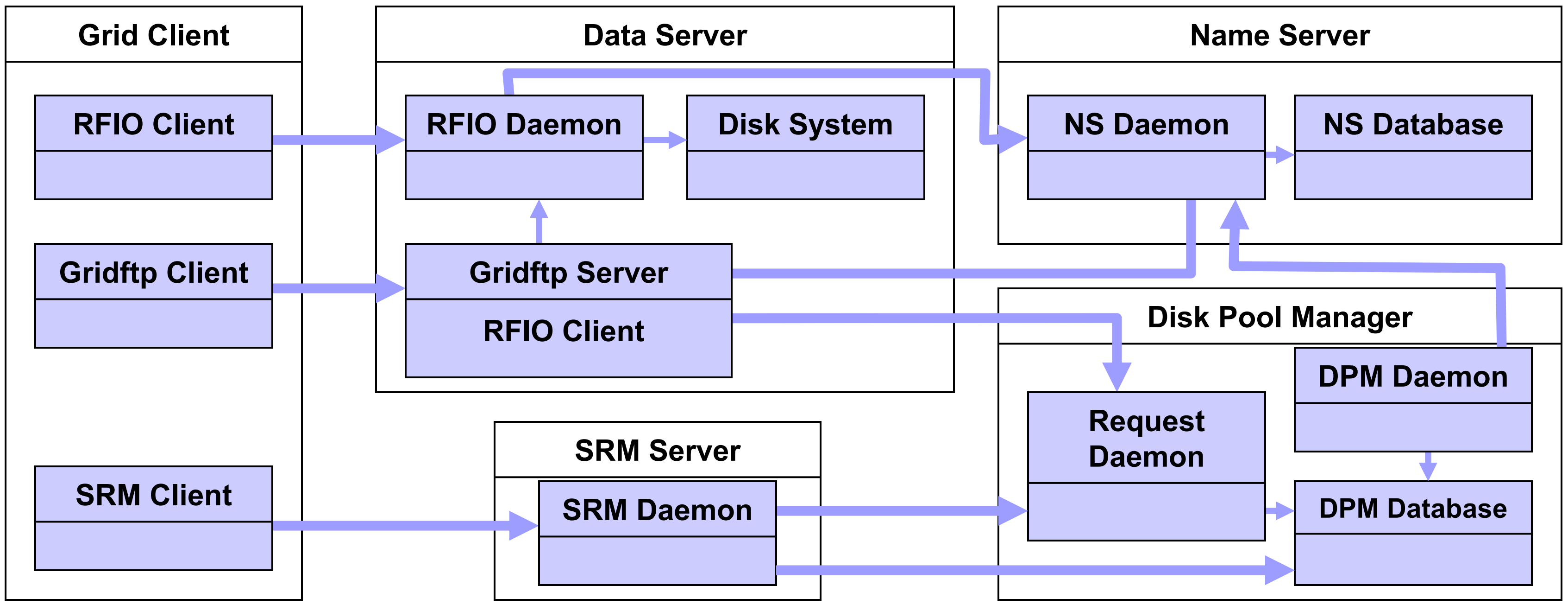
- DPM access via different interfaces
  - Direct Socket interface
  - SRM v1
  - SRM v2 Basic
  - Also offer a large part of SRM v2 Advanced
    - Global Space Reservation (next version)
    - Namespace operations
    - Permissions
    - Copy and Remote Get/Put
- Data Access
  - Gridftp, rfio, ROOT I/O
- DPM Catalog shares same code as LCG File Catalog
  - Possibility to act as a “Local Replica Catalog” in a distributed catalog

DPM Details

Features

- Namespace operations
  - All names are in a hierarchical namespace
  - mkdir(), opendir(), etc…
- Security – GSI Authentication and Authorization
  - Mapping done from Client DN to uid/gid pair
  - Authorization done in terms of uid/gid
  - VOMS will be integrated
    - VOMS roles appear as a list of gids
  - Ownership of files is stored in catalog, while the physical files on disk are owned by the DPM
  - Permissions implemented
    - Unix (user, group, other) permissions
    - POSIX ACLs (group and users)
- Retries and timeouts
  - Make client resilient to temporary outage of server

Architecture



Policies

- The policies are pool attributes
- There are currently 4 types of policies:
  - File system selection for storing a new file
    - The default policy is Round Robin as long as there is enough free space
  - Garbage collector
    - The default policy is to remove the least recently used files which are not pinned
  - Request selection
    - The default policy is FIFO
  - Migration policy: to automatically migrate durable files from Tier2 to Tier1 when space is needed for example
- All policies can be replaced online (shared library) and do not require code recompilation nor daemon restarts

Pool selection

- The pools have 2 attributes for this: space type (Volatile or Permanent) and restriction to certain VOs
- However a given pool might have no restriction: the pool is shared by all users and for any type of space
- We recommend to have different pools for Volatile and Permanent space: reliable hardware for permanent storage.
- Disks on CPU servers (worker nodes) can be used for Volatile space
- Hot files can be replicated to several disks and the DPM selects the best replica (less used or closest to the CPU)

Disk Pool Manager APIs

- There are 2 categories of APIs:
  - Administrative: disk pool configuration
    - dpm_addfs (char *, char *, char *, int);
    - dpm_addpool (struct dpm_pool *);
    - dpm_getpoolfs (char *, int *, struct dpm_fs **);
    - dpm_getpools (int *, struct dpm_pool **);
    - dpm_modifyfs (char *, char *, int);
    - dpm_modifypool (struct dpm_pool *);
    - dpm_replicate (char *);
    - dpm_rmfs (char *, char *);
    - dpm_rmpool (char *);
  - User: these map to the SRM v2.1 calls

Possible usage

- Replacement of ‘Classic SE’
  - Only metadata operations needed (the data does not need to be copied)
- Disk Cache behind a firewall:
  - Worker nodes behind a firewall would use the Volatile space in the disk pool and SRM remote get/put
  - GFAL would provide an automatic interface for it

Status – DPM

- Prototype DPM server is ready this week
  - DPM direct socket interface complete
    - All SRM v1/BASIC functionality
  - SRM available next week
    - Handling of high server load still to be done
- Still ongoing
  - RFIO
    - Working with CASTOR team so that the same secure RFIO client will work with DPM as CASTOR
  - Gridftp interface to RFIO complete this week
  - Security integration
    - Waiting on GCSI_gSoap-2.6 and Csec packages from CASTOR team

DPM – Status

- First full version next week
  - Full SRMv1 and direct socket interface complete
  - Includes Gridftp/RFIO with security

- Slight delay in prototype delivery
  - Less manpower on project than expected
  - One developer moved to ‘Robust Data Transfer’ Service Challenge
  - gLite expected effort did not materialize

The LHC File Catalog (LFC)

James Casey, IT-GD, CERN

October 2004

Overview

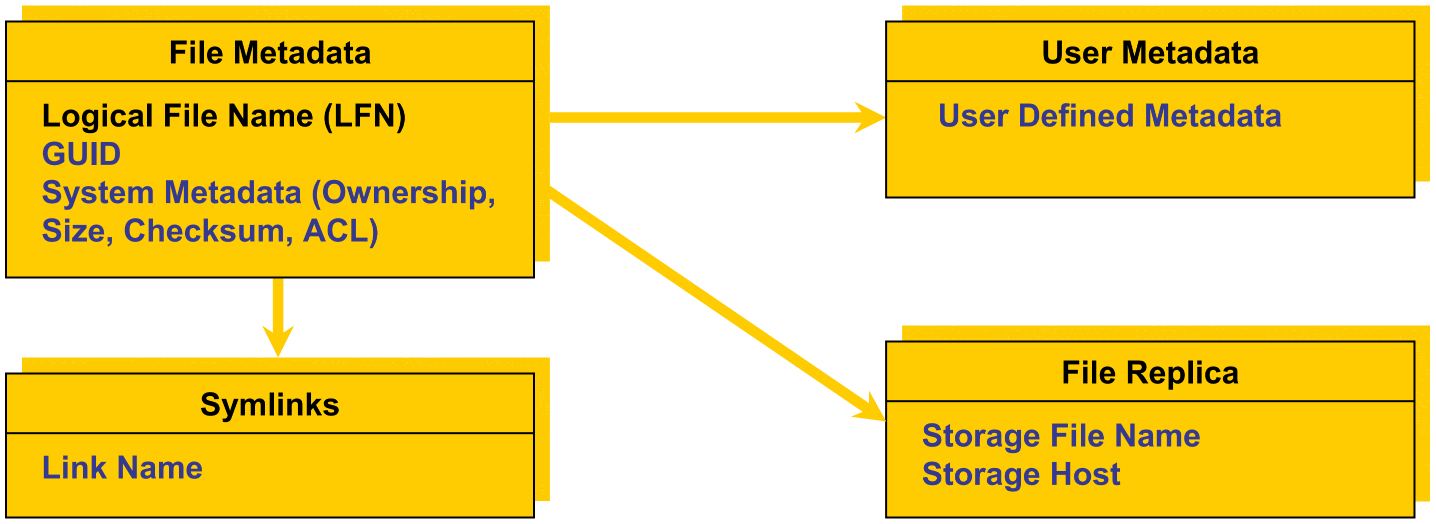
LCG File Catalog

- Based on lessons learned in DC’s in last few months
  - Fixes performance and scalability problems seen in EDG Catalogs
    - Cursors for large queries
    - Timeouts and retries from the client
  - Provides more features than the EDG Catalogs
    - User exposed transaction API
    - Hierarchical namespace and namespace operations
    - Integrated GSI Authentication + Authorization
    - Access Control Lists (Unix Permissions and POSIX ACLs)
    - Checksums

LCG File Catalog

- Based on existing code base
  - Supports Oracle and MySQL database backends
- Aim is to enable rapid development and deployment
  - Prototype exists and has undergone functional testing
  - Integration with GFAL and lcg_util complete
  - Performance and Scalability testing underway
  - ROOT Integration in progress
  - First version deployed for Certification – October 2004
  - POOL Integration will be provided – October 2004

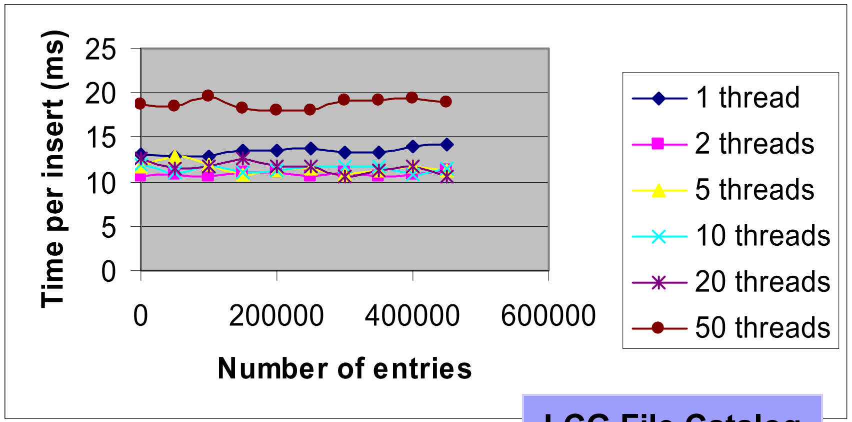
LCG File Catalog Schema



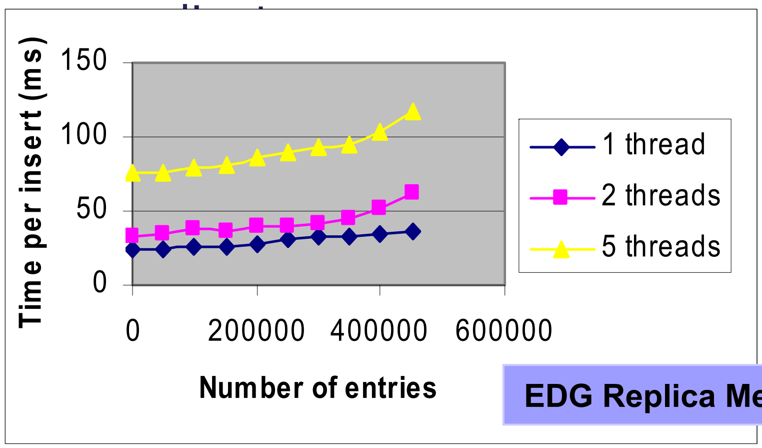
- LFN acts as main key in Database. Has:
  - Unique Identifier (GUID)
  - Information on Physical Replicas
  - Symbolic Links to it
  - A small amount (one field) of user attached metadata

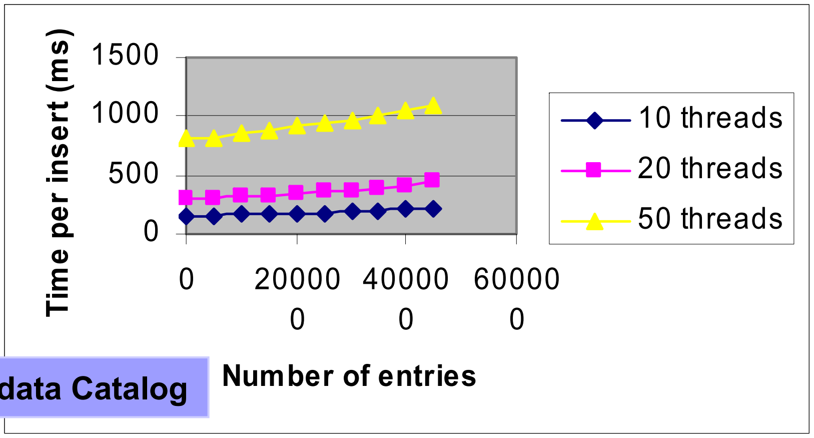
LCG File Catalog - Performance Comparison

- Average LFN,GUID mapping insert time:
  - EDG < 30ms for 1 or 2 threads
    - degrades quickly with number of entries and number of threads
  - New Catalog < 15 ms up to 20 concurrent inserts.
    - No degradation with number of entries
    - Scales much better with increasing



LCG File Catalog

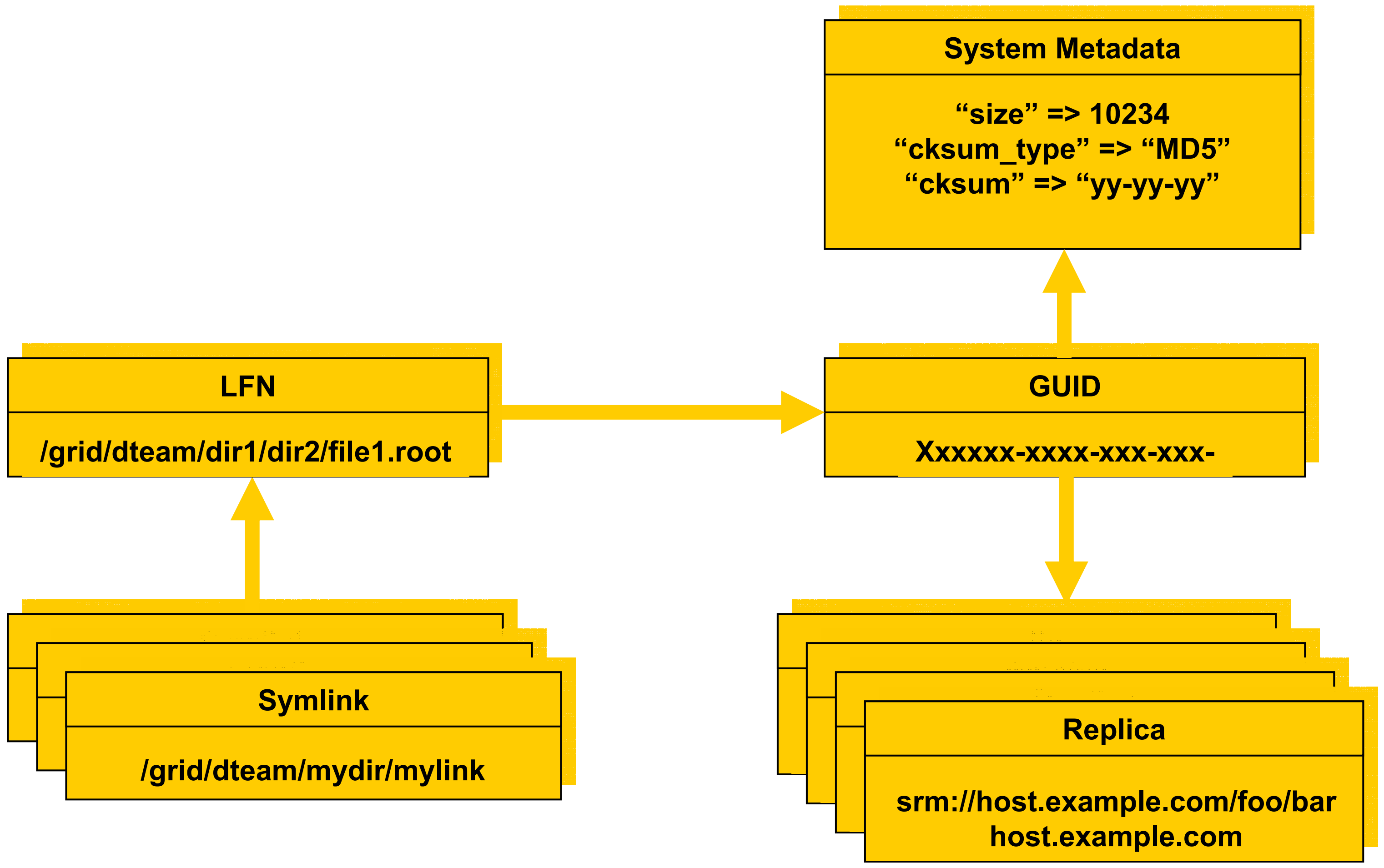




EDG Replica Metadata Catalog

LFC Details

Relationships in the Catalog



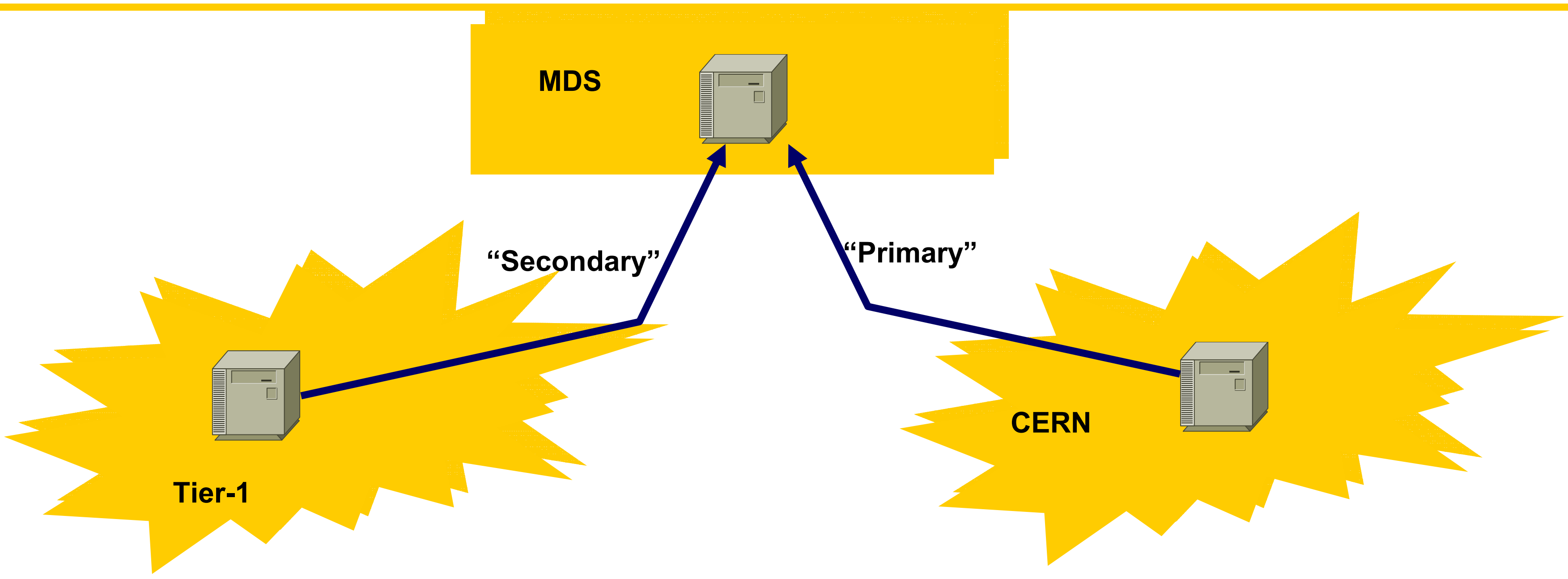
Features (1/2)

- Namespace operations
  - All names are in a hierarchical namespace
  - mkdir(), opendir(), etc…
  - Also chdir()
  - GUID attached to a directory
- Security – GSI Authentication and Authorization
  - Mapping done from Client DN to uid/gid pair
  - Authorization done in terms of uid/gid
  - VOMS will be integrated
    - VOMS roles appear as a list of gids
  - Ownership of files is stored in catalog
  - Permissions implemented
    - Unix (user, group, other) permissions
    - POSIX ACLs (group and users)

Features (2/2)

- Transactions
  - Exposed to user
    - starttrans(), endtrans(), aborttrans() methods
  - Auto-rollback on failure of mutating method call
- Cursors for queries
  - Modelled on opendir()/readdir()/closedir()
- Retries and timeouts
  - Make client resilient to temporary outage of server

Failover



- One site designated as "primary"
  - All connections go there
- On failure of primary (or scheduled switch)
  - Primary is set to "DRAINED" i.e. no new connections accepted
  - Secondary publishes in MDS as "primary"

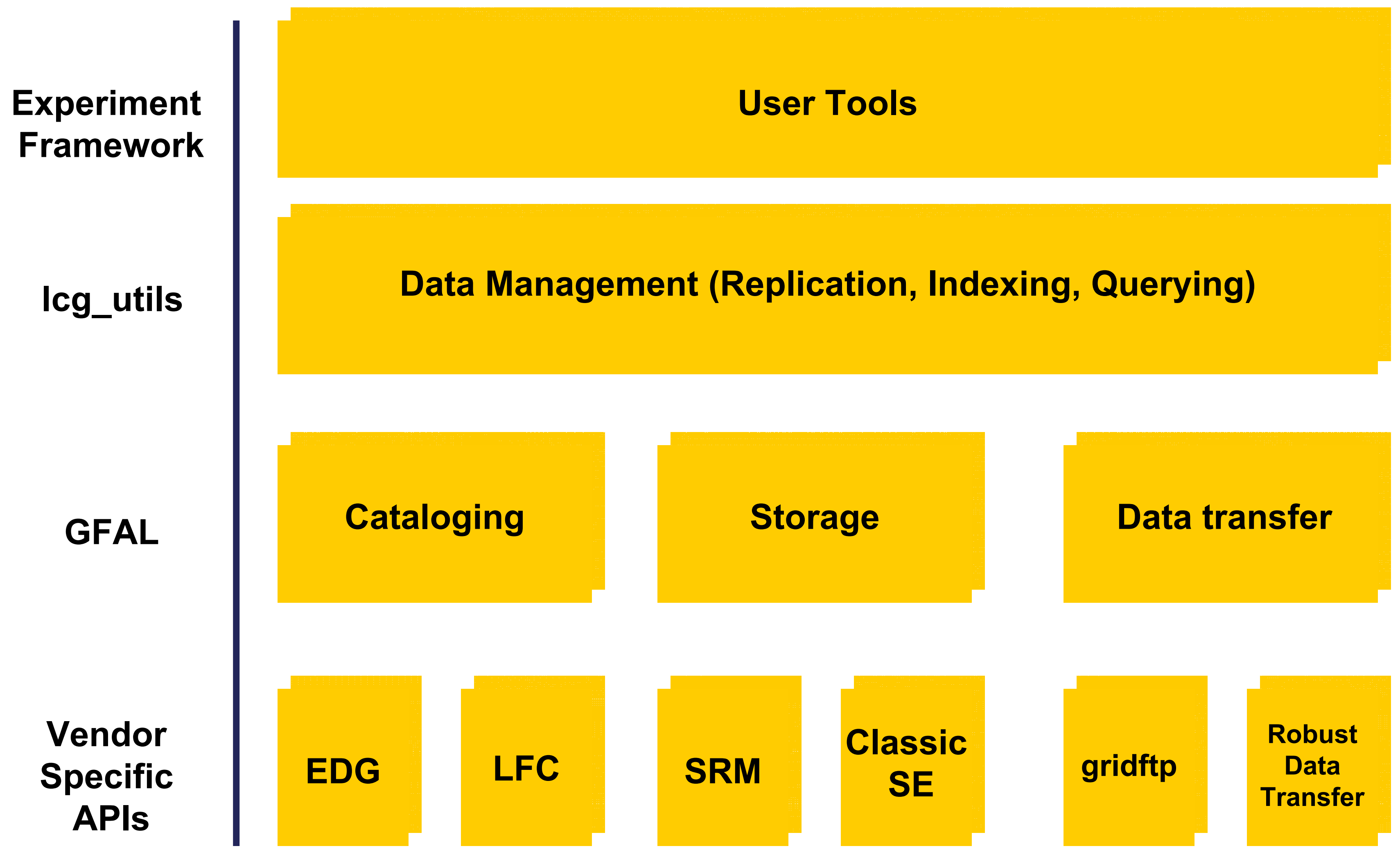
GFAL Overview

GFAL and lcg_util

- GFAL
  - originally a low-level IO interface to Grid Storage
  - Now provides:
    - File Catalog abstraction
    - Information system abstraction
    - Storage Element Abstraction (EDG SE, EDG ‘Classic SE’, SRM v1)
- lcg_util
  - Provides a replacement for the EDG Replica Manager
    - Provides both direct C library calls and CLI tools
    - Is a thin wrapper on top of GFAL
    - Has extra experiment requested features compared to the EDG Replica Manager

GFAL Details

Layered Data Management APIs



GFAL

- GFAL is a library to provide Grid Access
  - POSIX I/O
  - Catalog Interaction
  - Storage Interaction
- Single shared library in threaded and unthreaded versions
  - libgfal.so, libgfal_pthr.so
- Single header file
  - gfal_api.h

Catalog APIs in GFAL

```
int create_alias (const char *guid, const char *lfn, long long
   size)
int getfilesizeg (const char *guid, long long *size)
int guid_exists (const char *guid)
char *guidforpfn (const char *surl)
char *guidfromlfn (const char *lfn)
char **lfnsforguid (const char *guid)
int register_alias (const char *guid, const char *lfn)
int register_pfn (const char *guid, const char *surl)
int setfilesize (const char *surl, long long size)
char *surlfromguid (const char *guid)
char **surlsfromguid (const char *guid)
int unregister_alias (const char *guid, const char *lfn)
int unregister_pfn (const char *guid, const char *surl)
```

Storage APIs in GFAL

```
int deletesurl (const char *surl)
int getfilemd (const char *surl, struct stat64 *statbuf)
int set_xfer_done (const char *surl, int reqid, int fileid, char
    *token, int oflag)
int set_xfer_running (const char *surl, int reqid, int fileid,
    char *token)
char *turlfromsurl (const char *surl, char **protocols, int
    oflag, int *reqid, int *fileid, char **token)

int srm_get (int nbfiles, char **surls, int nbprotocols, char
    **protocols, int *reqid, char **token, struct srm_filestatus
    **filestatuses)
int srm_getstatus (int nbfiles, char **surls, int reqid, char
    *token, struct srm_filestatus **filestatuses)
```

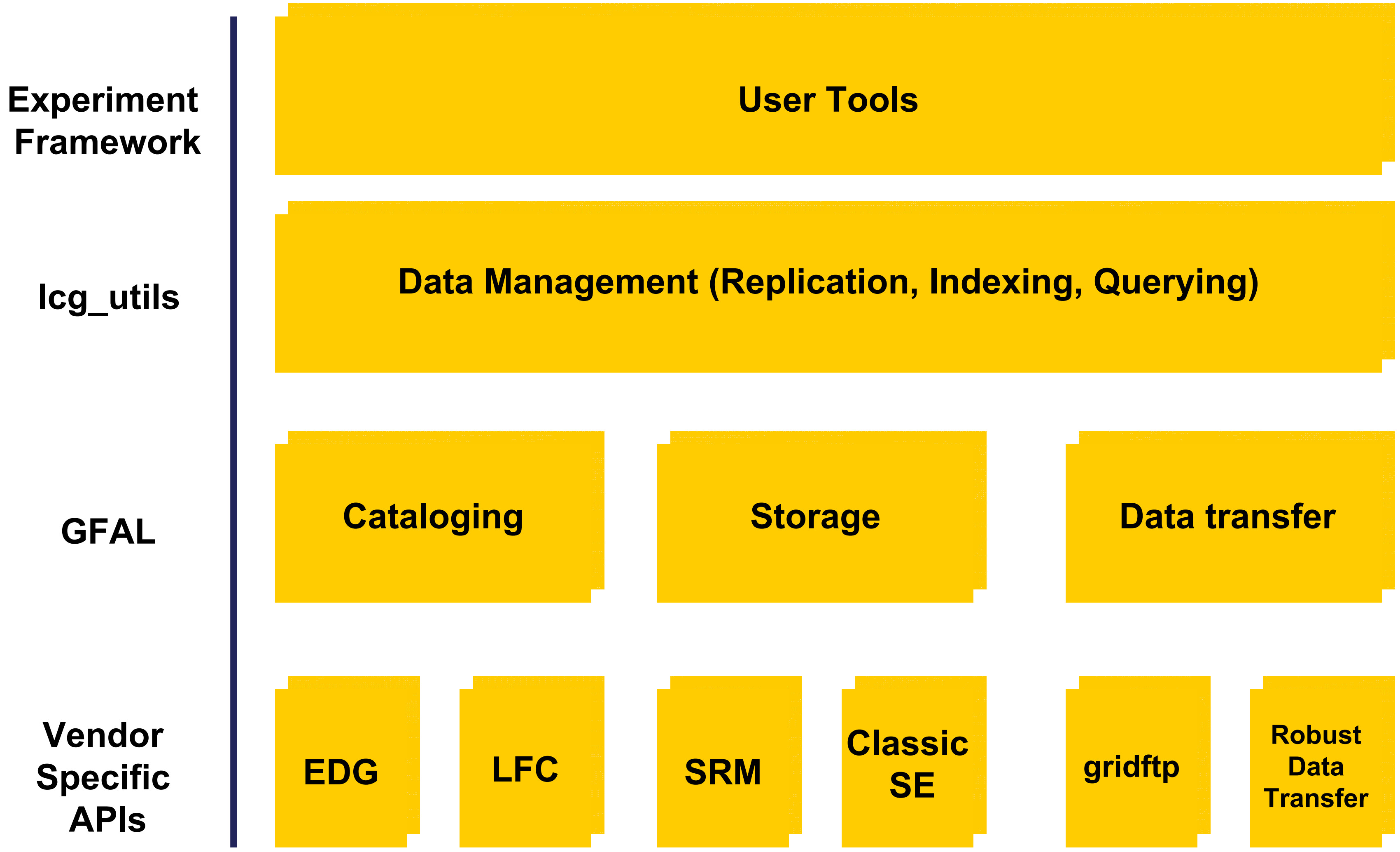
Status – LFC

- File Catalog will be part of C&T ‘October Release’
  - MySQL and Oracle Server implementation on SLC3
  - Client for RH7.3 and SLC3
- RPMs built
  - Undergoing testing this week on C&T testbed
- Details of service discussed with IT/DB
- Migration
  - Working with ATLAS to migrate their catalog
  - Will test on EIS testbed before going into production
- Outstanding Issues
  - Resource Broker integration to be tested via new DataLocationInterface
  - POOL Integration still to be done

GFAL & lcg_utils

- Integrated with LFC
  - Both EDG and LFC catalogs are supported with a runtime switch
  - Allows for gradual upgrades rather than big-bang style upgrade
- Easy to add another catalog to GFAL
  - e.g. gLite Fireman, Globus RLS
- GFAL becomes common API for data management comonents in LCG-2

Layered Data Management APIs



Status – Robust Data Transfer

- 'Radiant' – new name for the service/software
- Currently in tuning phase with sites
  - Fermi achieving 250MB/s sustained transfer rates
- Service Challenges start in December
  - NIKHEF/SARA first, then Karlsruhe and Fermi
- CERN Configuration
  - Currently only CASTOR gridftp servers with local disks
  - Deploying CASTOR SRM disk-only with a single stager for service challenges
  - Will test with full MSS connectivity in 2005
  - Will deploy small dCache configuration (2-node) for interoperability tests and debugging