



# **RRS: Replica Registration Service for Data Grids**

Arie Shoshani, Alex Sim, Kurt Stockinger

Computational Research Division  
Lawrence Berkeley National Laboratory  
University of California, Berkeley, California, USA

*VLDB Workshop on Data Management in Grids  
Trondheim, Norway, 2-3 September 2005*

# Why Replica Registration Service?



- ◆ Various large-scale **scientific experiments** have developed their own **catalogs** for keeping track of files:
  - (Specialized) File catalogs
  - (Distributed) Replica catalogs
- ◆ Different **Grid projects** have developed catalogs with similar features but are often not compatible (selected list):
  - Globus-RLS vs. EDG-RLS vs. LCG-Catalogs vs. ...
  - gLite Fireman (File + replica catalog et. al)
  - Storage Resource Broker (SRB)
  - ...

# Towards Standardization of File Access on Data Grids



- ◆ Goal: **Interoperable file access** across experiments/projects
- ◆ Take experience from **Storage Resource Manager**:
  - Standardization of access to hierarchical storage systems
  - **Collaboration** between several laboratories:
    - ANL, CERN, Fermilab, JLab, LBNL=Berkeley Lab
- ◆ The idea is not to standardize particular catalogs but to develop **uniform interface** to access **existing catalogs**
- ◆ Advantage of joint **standardization on interface**:
  - Allows **multiple systems to coexist**
  - **Analogy** from (VL)DB-perspective:
    - SQL for commercial Database Management Systems (DBMS)

# Design Principles of RRS

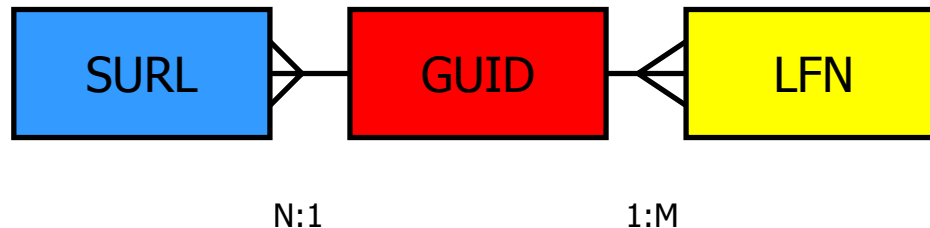


- ◆ **Abstraction** of concepts for registering files and replicas:
  - Uniform interface to file/replica/meta-data catalogs
- ◆ **Collective** file registration **requests** (as opposed to single registration operations)
- ◆ **Persistent** registration **queues**:
  - Important for copying and registering thousands of files
- ◆ Trigger **error directives** in case of failures
- ◆ **Goal**:
  - Grid clients should be able to **connect to any registration service** irrespective of underlying back-end implementation
  - Allow **applications** to **choose catalogs** according to their needs by implementing a RRS interface (rather than an entire catalog)

# Terminology: LFN – GUID – SURL



- ◆ SURL (Site URL, generalization of PFN = physical file name)
  - `srm://sdm.lbl.gov:4004/tmp/run12/file17.raw`
- ◆ GUID (Globally Unique Identifier)
  - `guid://123456`
- ◆ LFN (Logical File Name, alias name for a GUID)
  - `lfn://star/run12/file17.raw`



Note: in case that only LFN is used then LFN = GUID  
(e.g. STAR High Energy Physics experiment at Brookhaven National Laboratory)

# Name Space and File Attributes

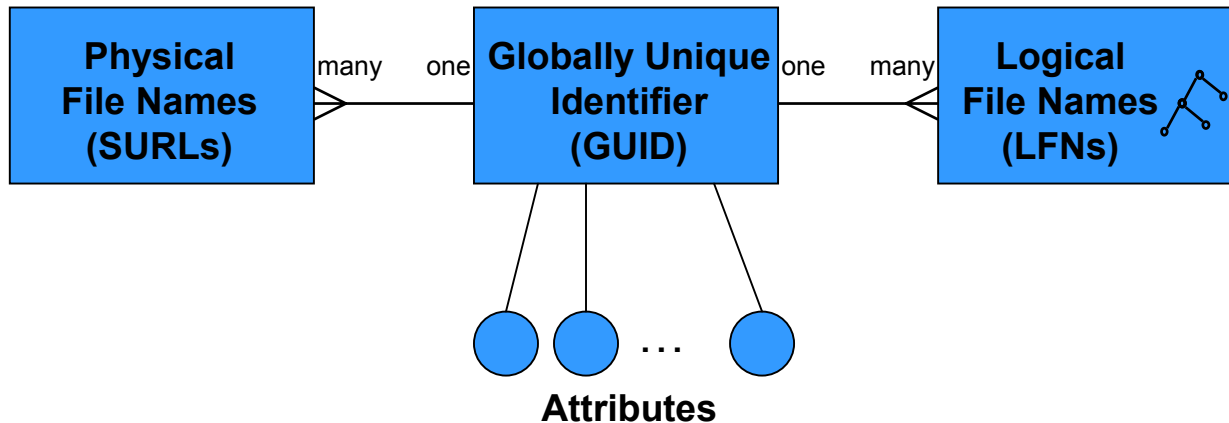


## ◆ Name space:

- LFNs often organized into **directory structures**, e.g. UNIX
- Allows referring to directory in a similar way like to a file

## ◆ File attributes:

- Associated **with GUIDs** and represent **global properties**
- Independent of physical file location
- To verify correctness of file transfer (**core attributes**):
  - fileFize, checkSumType, checkSumValue





# Replica Registration Service - API



## ◆ Basic API:

- Register Functions
- Discovery Functions

## ◆ Advanced API:

- Attribute Functions
- Name Space Management

# Basic API – Register Functions



- ◆ **openCollectiveRegistration**
  - Prepare for multiple registration request calls
  - Specific registration and error mode
  - Return request token
- ◆ **register**
  - Register one or more files using request token
- ◆ **closeCollectiveRegistration**
  - Close specific registration using request token
- ◆ **getRegistrationStatus**
  - File status (done, in progress, pending)
  - Error codes (file not found, exists)
- ◆ **abortRegistration**
  - Stop registration request and unregister files
- ◆ **unregister**
  - Unregister files
- ◆ **getUnregisterStatus**
  - Retrieve status of unregister request

# Basic API – Discovery Functions



## ◆ getFileReferences

- Retrieve GUID, LFN and SURLs of a given file reference

## ◆ getFileReferencesStatus

- Retrieve status of file reference

# Advanced API – Attribute Functions



- ◆ getFileAttributes
  - Check sum, file size
- ◆ getFileAttributesStatus

# Advanced API – Name Space Management



- ◆ makeDirectory
  - Create directory and register in catalog
- ◆ removeDirectory
- ◆ listDirectory
  - List content of directory
- ◆ listDirectory
- ◆ getListDirectoryStatus

# Example: openCollectiveRegistration



IN:     String userID,  
          String registrationMode, // default: "atEnd"  
          String registrationErrorDirective, // default: "stop"  
          Int errorDirectiveTrigger // default: 1

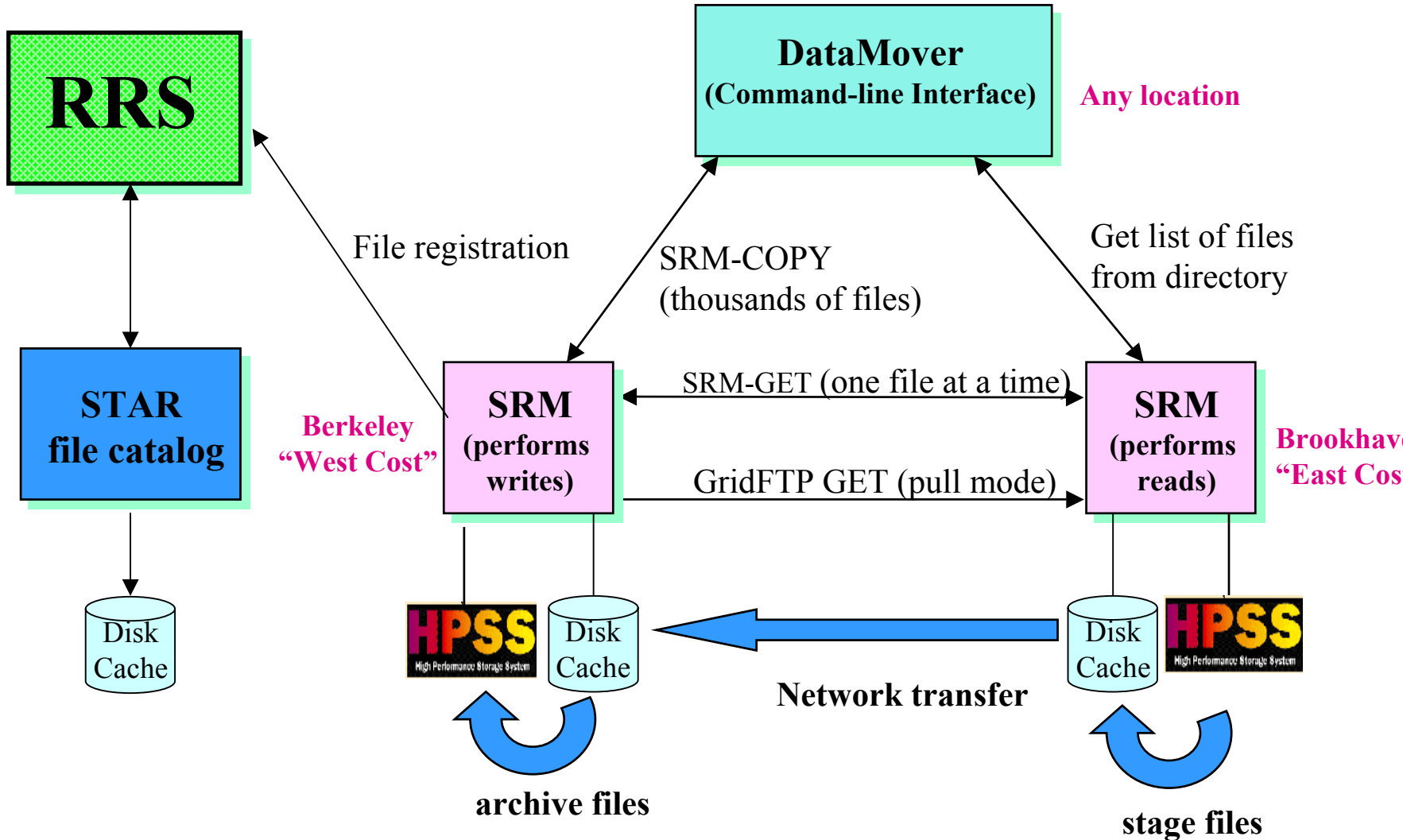
OUT:    String requestID,  
          String statusCode,  
          String statusExplanation

# RRS in Production Use #1



- ◆ We **implemented the RRS** for the STAR **High Energy Physics** Experiment at Brookhaven National Laboratory
- ◆ Has been in production use for **> 1.x years**
- ◆ Data files are **replicated daily** from Brookhaven to Berkeley for post-processing and analysis (files are stored in HPSS)
- ◆ **Before RRS:**
  - File registration at Berkeley Lab was done manually (error prone) or by scripts
  - Several thousands of files registered per month
  - Total volume: > 5 Terabytes
- ◆ **With RRS:**
  - Fully automatic file registration process
  - RRS implemented as daemon module to interact with **DataMover** (by Berkeley Lab)
  - DataMover, in turn, interacts with two **SRMs** (@ Berkeley and Brookhaven)
- ◆ **Error rates** for file registration **dropped**
  - From 1% to 0.02% according to person in charge of “production”

# RRS in Production Use #2



# RRS in Production #3



## ◆ When target **SRM** receives a file:

- Archive file
- Notify RRS

## ◆ **RRS**:

- Register file to **STAR file catalog** (based on MySQL)
- Three registration modes:
  - Continuous, every-n-files, at-end (bulk-registration)
- Registration request is **fault tolerant**
  - Retry mechanism in case RRS is down or temp. unavailable
- Continuous registration mode turned out to be the most effective for large-scale registrations

# Conclusions



- ◆ In the past...
  - Many **interoperable file/replica catalogs** were built
  - Motivation for **designing** Replica Registration Service (RRS):
  - Good experience with standardization of **Storage Resource Manager (SRM)**
- ◆ We **implemented** RRS:
  - Register files and directories
  - Focus on fault tolerance
- ◆ We **used** RRS in production environment:
  - Successfully **registered** up to **5 TB files** over several month
  - Data stored in **HPSS** and transferred from Brookhaven to Berkeley
  - **Error rate** of registration **dropped** from 1% to 0.02%
- ◆ More information:
  - Cf. paper
  - **Scientific Data Management Research Group**
    - <http://sdm.lbl.gov/>
  - Live demo at **Super Computing 2005**, Seattle, Washington
    - in **Berkeley Lab Booth**