Storage Resource Managers: Middleware Components for Grid Storage

Arie Shoshani
Alex Sim
Junmin Gu
Computing Sciences Directorate
Lawrence Berkeley National Laboratory

Outline

- What are Storage Resource Managers - Motivation
- Typical Analysis Scenario and the use of SRMs
- SRM functionality
- Real examples of working SRMs
- Implementation Challenges
- File Pinning Deadlocks
- Advantages of using SRMs
- Conclusions and Future Work

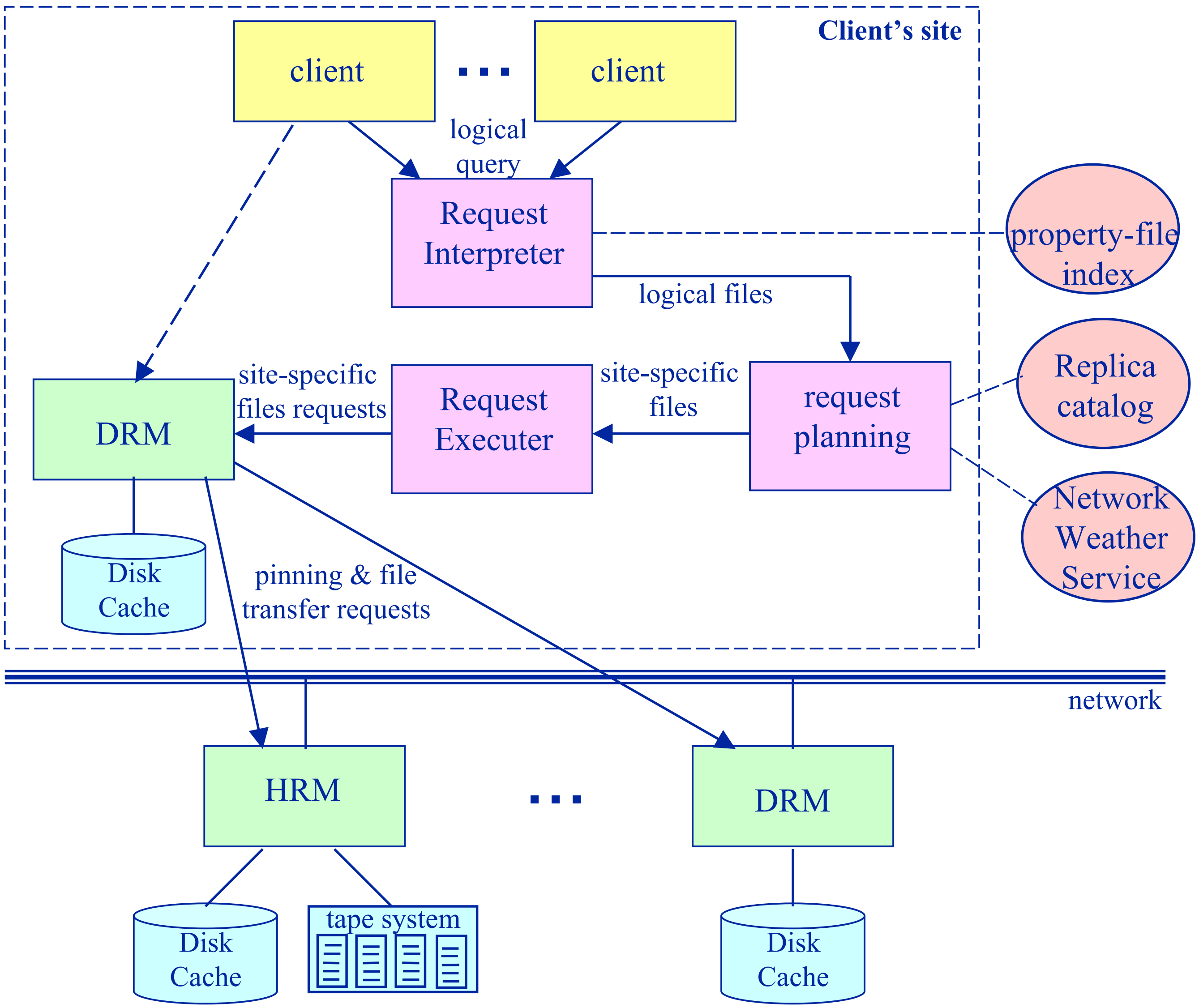
Motivation

- Grid architecture emphasized in the past
  - Security
  - Compute resource coordination & scheduling
  - Network resource coordination & scheduling (QOS)
- SRMs role in the data grid architecture
  - Shared storage resource allocation & scheduling
  - Especially important for data intensive applications
  - Often files are archived on a mass storage system (MSS)
  - Wide area networks – minimize transfers
  - large scientific collaborations (100's of nodes, 1000's of clients) – opportunities for file sharing
  - Nodes may be organized by tier levels
  - File replication and caching may be used
  - Have to support non-blocking (asynchronous) requests

SRM is a Service

- SRMs and File transfers
  - SRMs DO NOT perform file transfer
  - SRMs DO invoke file transfer service if needed (GridFTP, FTP, HTTP, …)
- SRM functionality
  - Manage what should reside on a storage resource at any one time
  - Get files from remote locations when necessary
  - Pin files in storage till they are released
  - Timeout on pins
  - Provide grid access to/from mass storage systems (HPSS, Enstore, JasMINE, Castor, …)
- Types of storage resource managers
  - Disk Resource Manager (DRM)
  - Tape Resource Manager (TRM)
  - Hierarchical Resource Manager (HRM=TRM + DRM)

Typical Analysis Scenario and the use of SRMs



Three scenarios that SRMs should be able to support

- A client communicates directly with DRM/HRM
  - No way to call client back
  - May ask to get a local / remote file
  - May ask to put a file
- An agent calls DRM on behalf of a client
  - E.g. Request executer
  - It is possible to call agent back
  - May ask for local / remote file
- A DRM calls another DRM (or HRM)
  - As a result of a request for a remote file
  - To request a file to be pinned

DRM Functionality

- **Manages disk cache**
  - Keeping track of files in its disk
  - Allocating space for files to be brought to its disk
  - Pinning files for clients and keeping track of pins
- **Manages multi-file requests**
  - Queuing and keeping track per client of all files requested in a single request
  - enforces pin lifetime policies
  - enforces user priority policies
  - enforces user quota limit policies per request and per client

DRM Functionality

- Optimizes disk cache use
  - Replacement policy - makes decisions on which files to remove when space is needed
  - Admission policy - optimize use of files in disk to be shared by clients based on anticipated use
  - Service policy – to optimize disk use, but being fair to clients
- Key point
  - When "get file" is requested
    - If file in disk – return that file
    - If not in disk – get it from it source location
  - Consistent view with HRM (next)

HRM Functionality

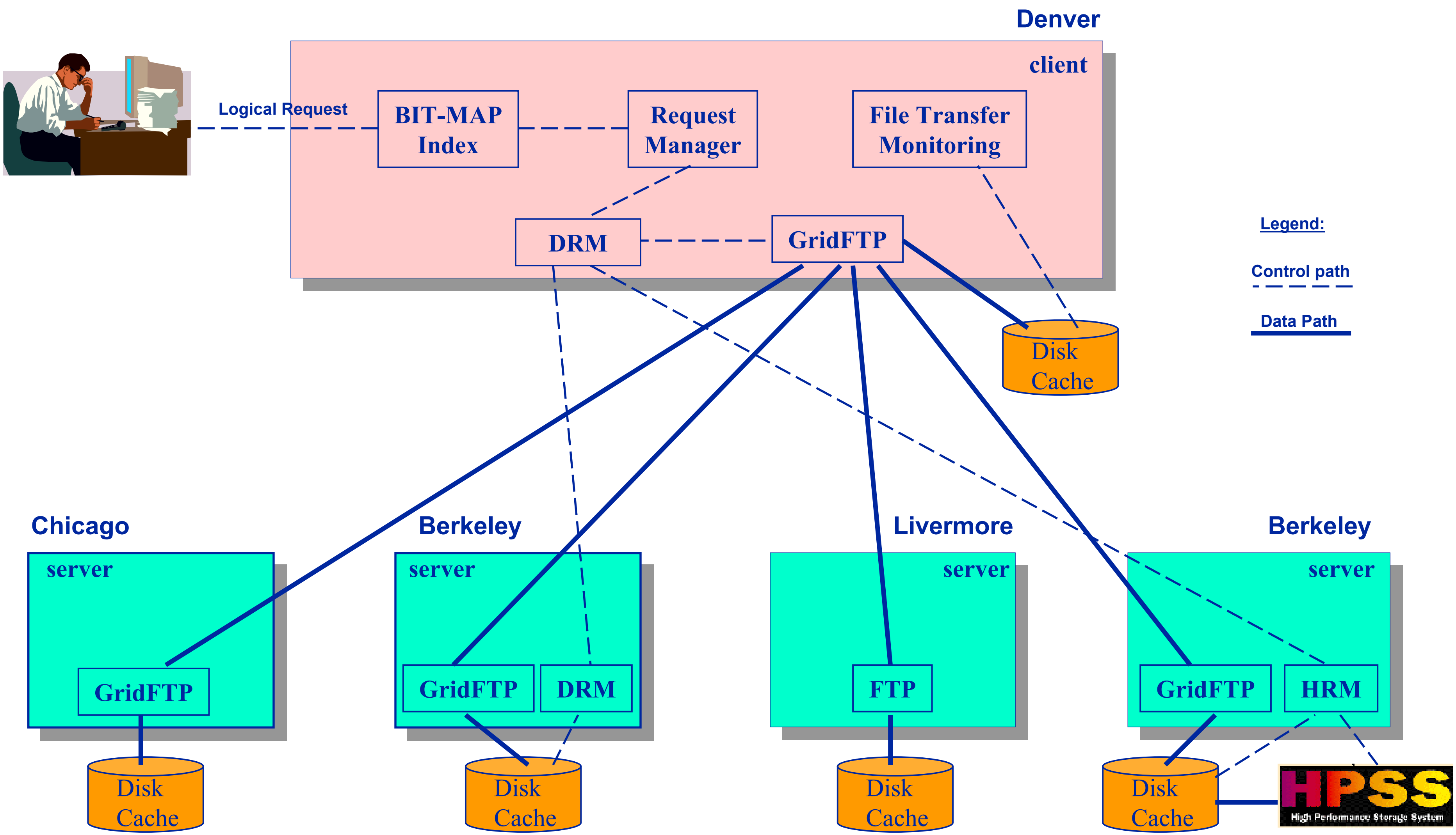
- Same as DRM, but also:
  - Queuing of file staging and archiving from/to tape
  - Reordering of request to optimize tape mounting and reading (ordered by files on the same tape)
  - Monitoring staging/archiving progress and error messages from MSS (e.g. HPSS)
  - Reschedules transfers that failed
- Enforce MSS policy
  - Number of simultaneous file transfer requests
  - Fair treatment of users when reordering tape requests
- Same interface (methods, API) as DRM

Interface Functionality

- Want to get a file
  - Request_to_get (push/pull)
  - Release
  - Abort
  - Status
  - Call_back (when file is available)
- Want to put a file
  - Request_to_put (push/pull)
  - Release
  - Abort
  - Status
  - Call_back_1 (when file is transferred to disk)
  - Call_back_2 (when file is transferred to tape – for HRM)

Supercomputing 2001 Demo

Computing Sciences Directorate, LBNL



Middleware Software Shown in Demo

1) Request Interpreter - BitMap index

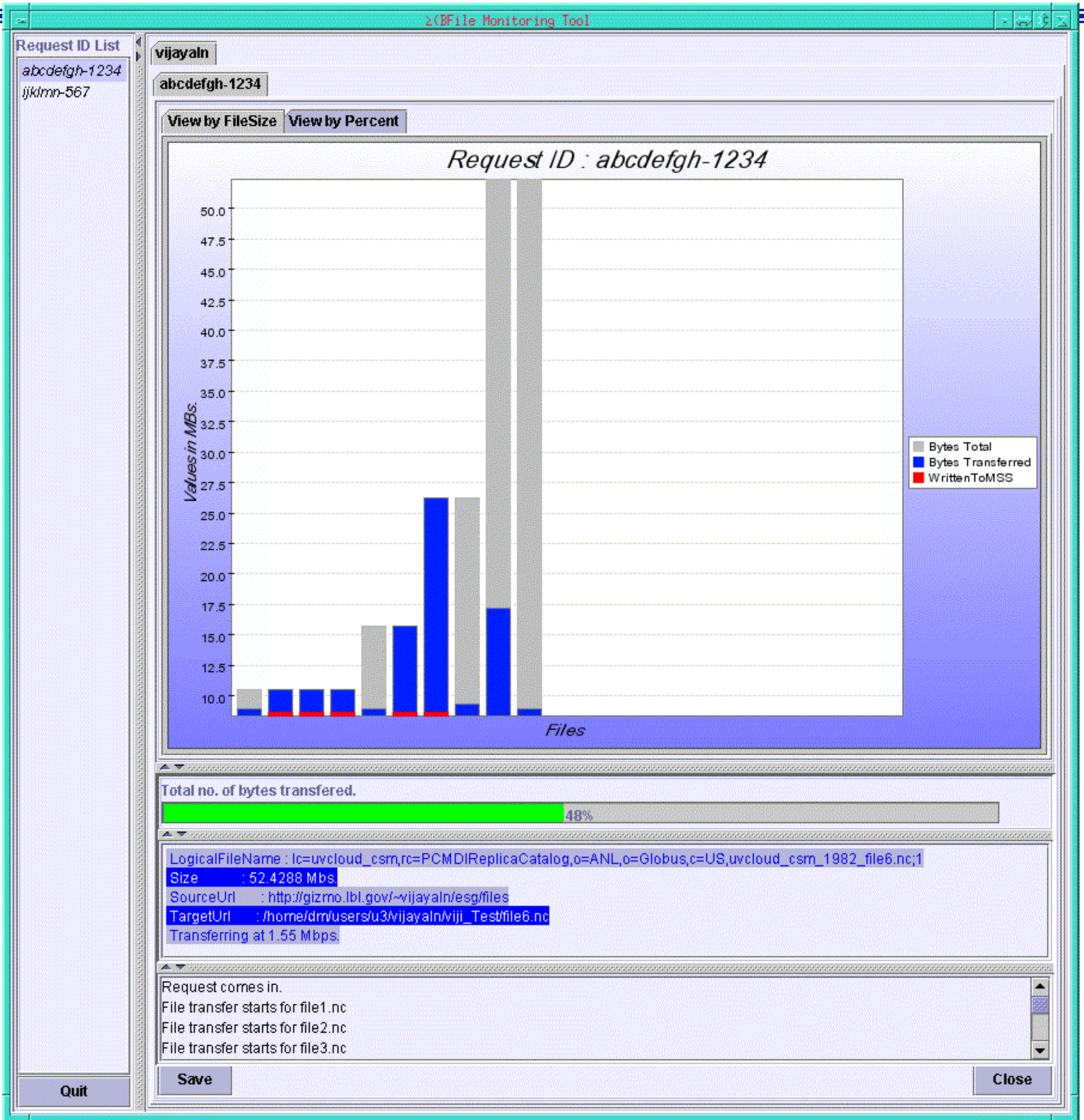
- in: logical request
  - ((0.1 < AVpT < 0.2) AND (10 < Np < 20)) or (N > 6000)
- out: a set of logical files
  - {star.simul.00.11.16.tracks.156,…, star.simul.00.11.16.tracks.978}
- Size of data to be indexed:
  10^8 objects x 500 attributes x 4 bytes = 200 GB

2) Request Executer

- in: a set of files
  - {star.simul.00.11.16.tracks.156,…, star.simul.00.11.16.tracks.978}
- out: selected URLs
  - gsiftp://dg0n1.mcs.anl.gov/homes/sim/gsiftp/star.simul.00.11.16.tracks.156
  - hrm://dm.lbl.gov:4000/home/dm/srm/data1/star.simul.00.11.16.tracks.978
- Uses Replica Catalog
- Monitors transfer progress

Monitoring File Transfer

Computing Sciences Directorate, L B N L



Implementation Challenges (1)

- Managing storage resources in an unreliable distributed large heterogeneous system
- Long lasting data intensive transactions
  - Can't afford to restart jobs
  - Can't afford to loose data, especially from experiments
- Type of failures
  - Storage system failures
    - Mass Storage System (MSS)
    - Disk system
  - Server failures
  - Network failures

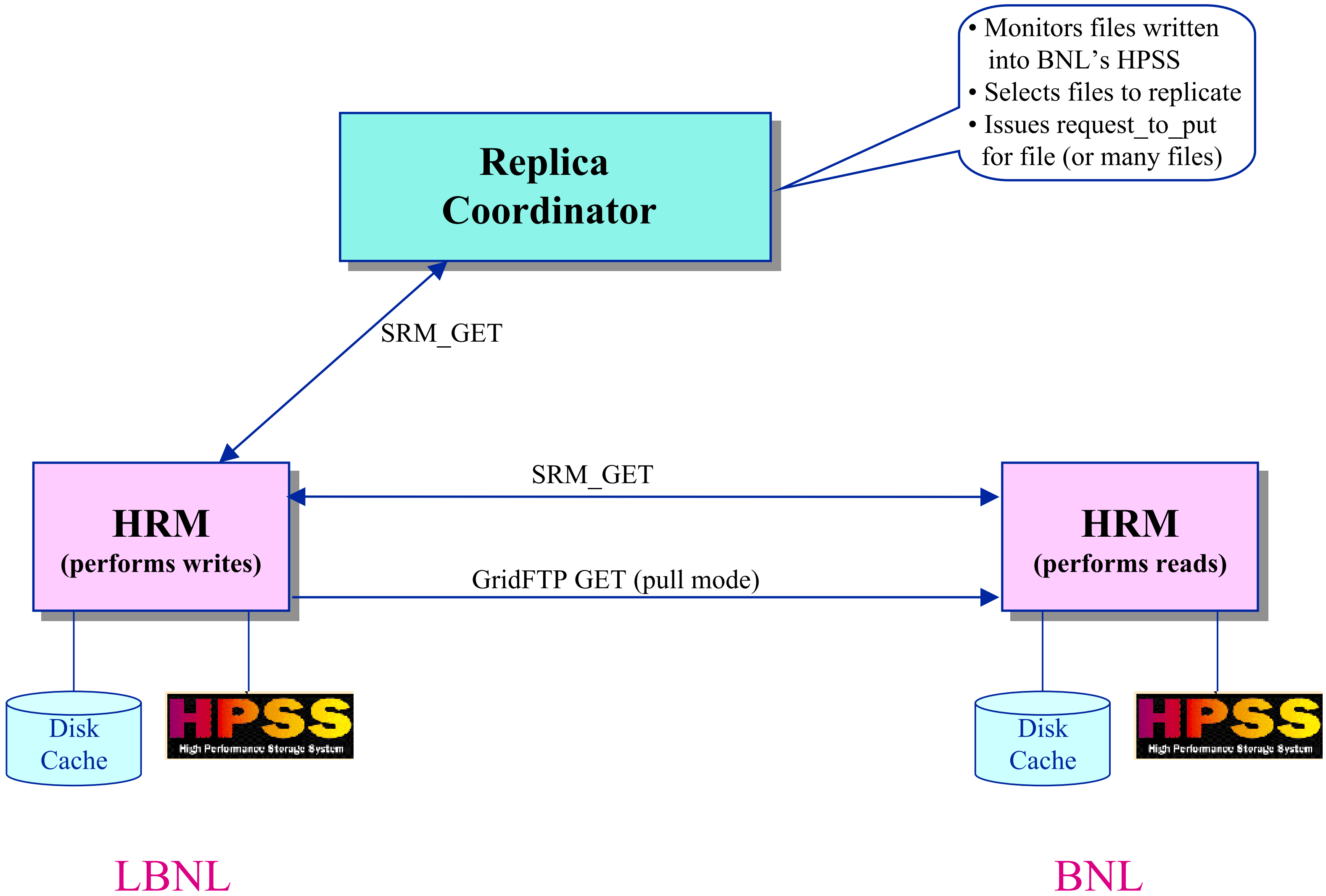
Implementation Challenges (2)

- **Heterogeneity**
  - Operating systems (well understood)
  - MSS - HPSS, Castor, Enstore, …
  - Disk systems – system attached, network attached, parallel
- **Optimization issues**
  - avoid extra file transfers - What to keep in each disk caches over time
  - How to maximize sharing for multiple users
  - Global optimization
  - Multi-Tier storage system optimization

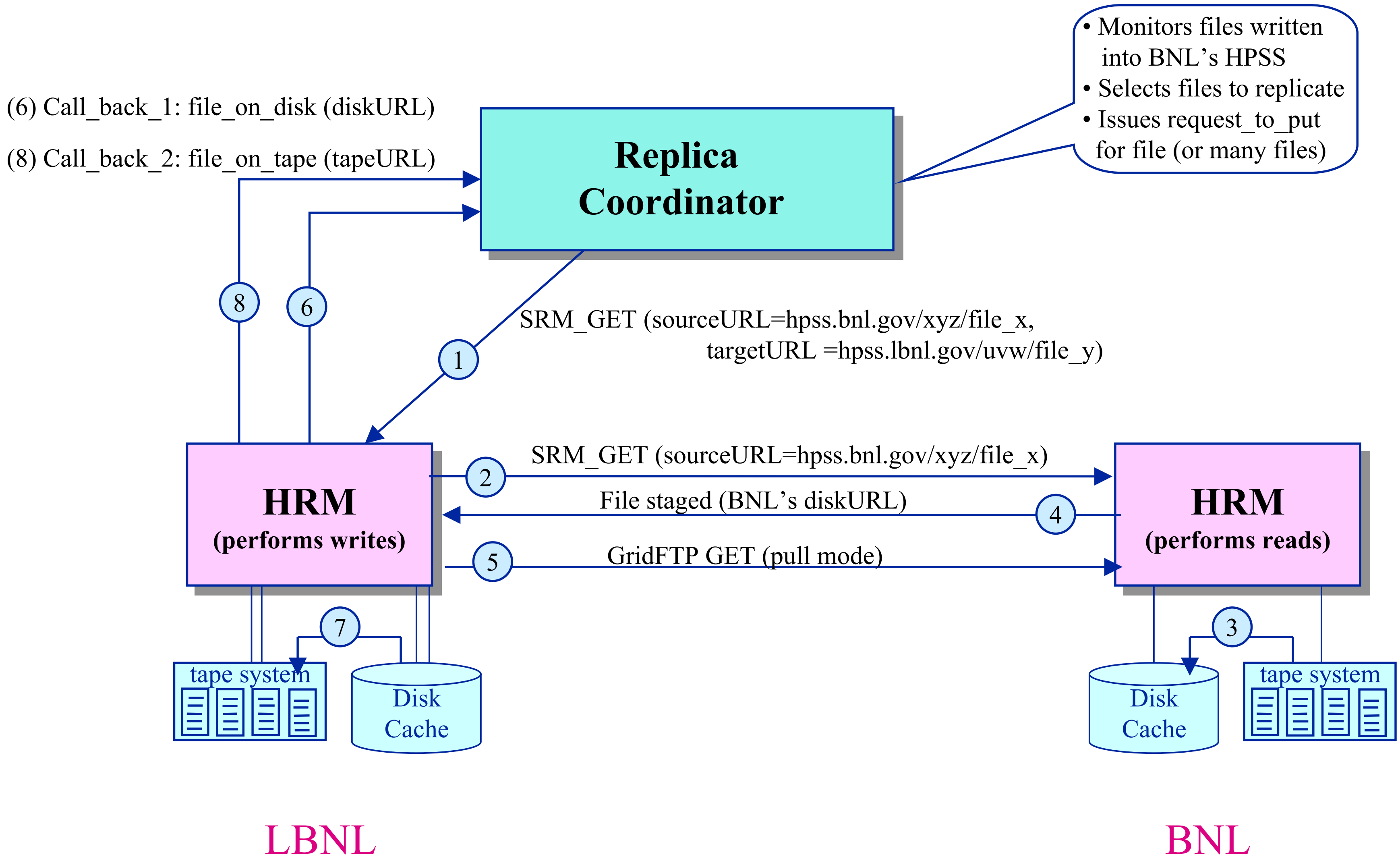
File Pinning Deadlocks

- Pin is the concept of “space locking”
- Assume a site X has space for 2 files
  - Process A needs 2 files on site X, and has one file pinned
  - Process B needs 2 files on site X, and has one file pinned
  - => A & B will be deadlocked until some other process finished
- Can be avoided by “two-phase pinning”
  - Allocate space first, then move files
  - Impractical for very large file requests (e.g. 500 files)
  - Need to enforce protocol for smaller file request
  - Or support pre-allocation (more difficult)
- Streaming model
  - Provide default “quota”
  - Do not provide service till files in quota are released
  - Support for “file bundles” – to allow small group of concurrent file requests

Use of HRMs for managing large file replication tasks

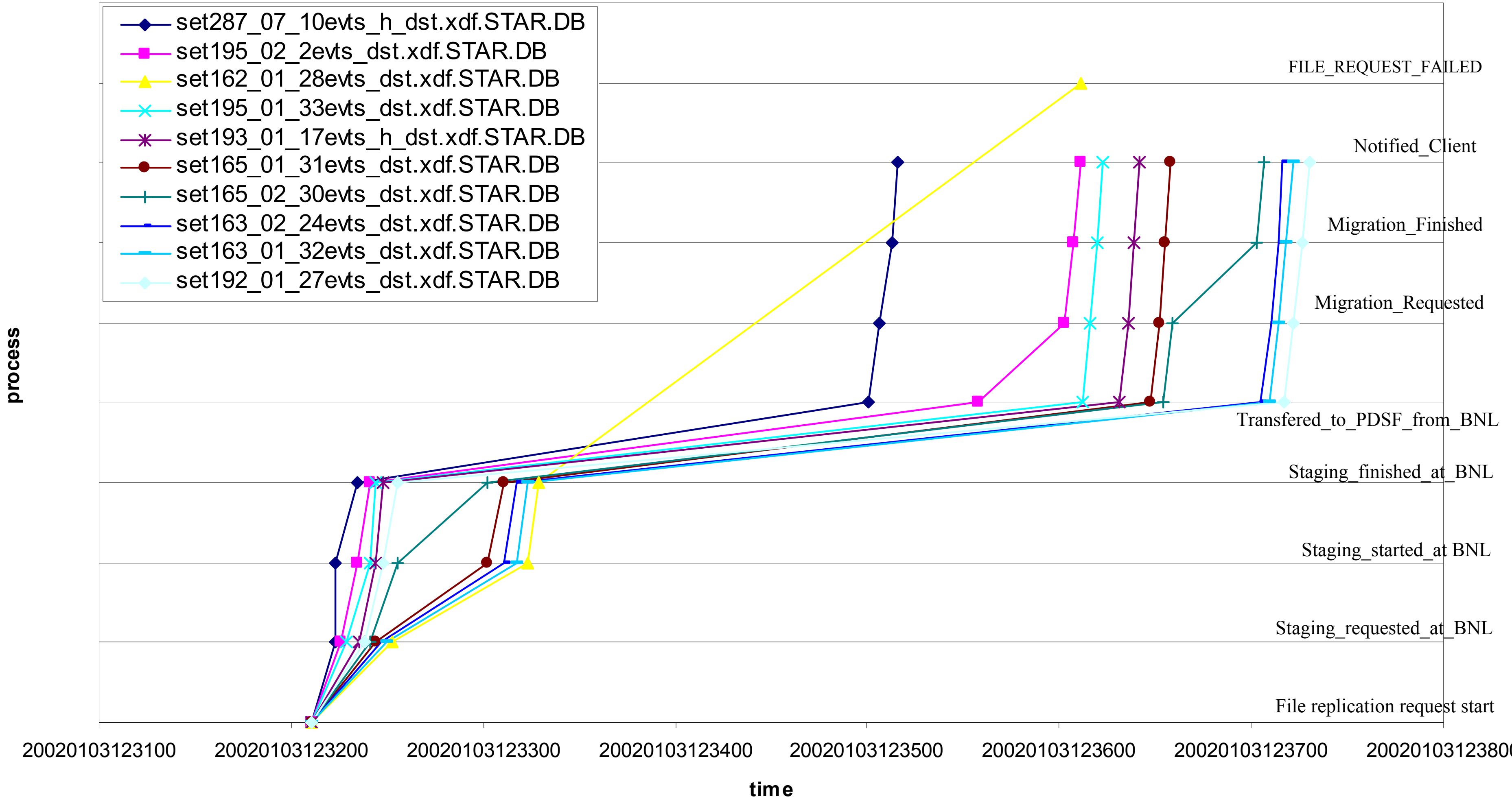


Sequence of actions
(detailed view)



Computing Sciences Directorate, LBNL

Tracking File Replication (From HPSS to network to HPSS)



Advantages of using SRMs

- Smooth synchronization between storage resources
  - Pinning file, releasing files
  - Allocating space dynamically on as "needed basis"
- Insulate clients from storage and network system failures
  - Transient MSS failure
  - Interruption of large file transitions
- Facilitate file sharing
  - Eliminate unnecessary file transfers
- Support "streaming model"
  - No need for space pre-allocation by SRMs
  - No need for reservation and release by client
  - No need for accounting and charging
- Control number of concurrent file transfers
  - From MSS – avoid flooding and thrashing
  - From network – avoid flooding and packet loss

Conclusions and Future Work

- **Conclusions**
  - SRMs essential for shared resources
  - SRMs essential for dealing with large files
  - SRMs are needed to support local policies of grid sharing
  - SRMs treat network delays similar to MSS delays
  - SRMs support "streaming model" – a practical model
  - SRMs – key elements to storage sharing on grids
- **Future work**
  - Developing Standard SRM interfaces
    - http://sdm.lbl.gov/srm
  - Having HRM implementation adaptable to multiple MSSs
  - Security and access control (e.g. login to MSSs)
  - Access authorization – community access service (CAS)
  - "On-demand" space allocation, accounting, and charging