

# The Data Grid

Nicu D. Cornea

Parallel and Distributed Computing  
Spring 2003

March 12, 2003

# Outline

---

- Motivation and Related Work
- The Data Grid: An Architecture for Distributed Management and Analysis of Large Scientific Datasets
- Data Management for Grid Environments
- Data Management in an International Grid Project
- Giggle: A framework for constructing scalable replica location services

# Outline

---

- **Motivation and Related Work**
- The Data Grid: An Architecture for Distributed Management and Analysis of Large Scientific Datasets
- Data Management for Grid Environments
- Data Management in an International Grid Project
- Giggle: A framework for constructing scalable replica location services

# Motivation

---

- Large data collections (tera/peta bytes) becoming common
  - ◆ Global Climate Change
  - ◆ High Energy Physics
  - ◆ Computations Genomics
- Geographic distribution of users and resources
- Computationally intensive analysis of the datasets
- Numerous point solutions to problems like:
  - Data intensive computing
  - Storage management

... but no integrating architecture

## Motivation (2)

---

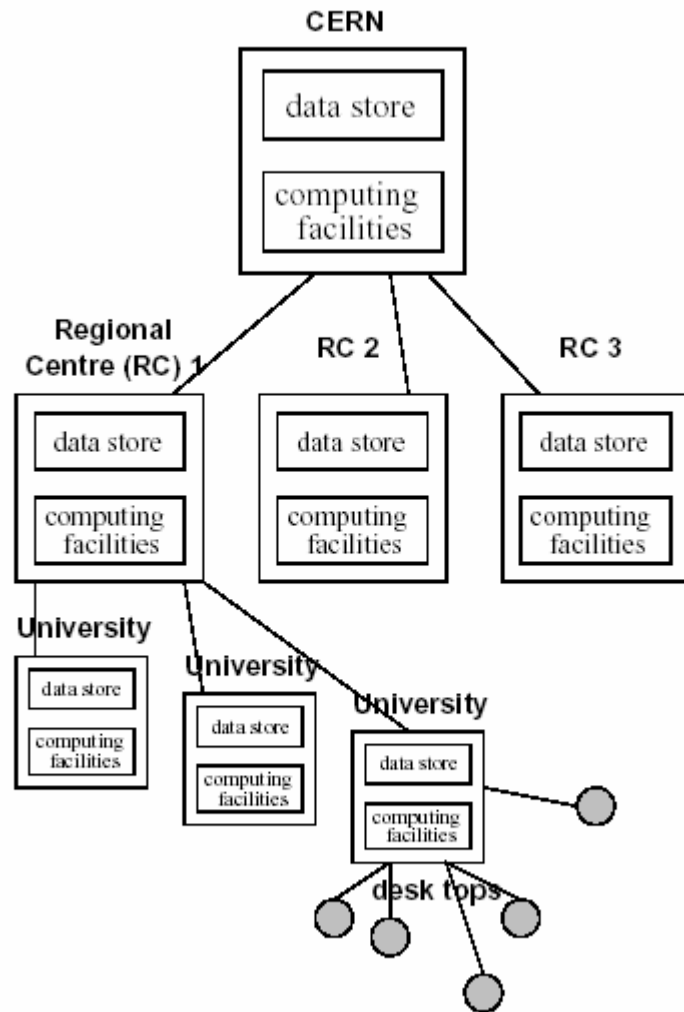
- Communities that require access to distributed data sources
  - Digital libraries
    - ◆ Repositories of digital objects
  - Grid environments for processing distributed data
    - ◆ Distributed visualization, knowledge discovery and management
  - Persistent archives
- Need for a uniform API for managing and accessing distributed data

# Use Cases

---

- High Energy Physics
  - Large collaboration – over 2.000 scientists
  - Research centers distributed all over the world analyze the same data produced in one place (CERN)
  - Analysis jobs
    - ◆ Computational intensive
    - ◆ May require selection of “interesting” data
    - ◆ Application of multiple functions to the same data set - caching
- Earth Observation
  - Data is collected and stored at distributed locations
- Molecular Biology
  - Data stored in a variety of independent databases

# CERN's LHC Project Layout



Large Hadron Collider – a new particle accelerator at CERN (European Organization for Nuclear Research)

- Will produce petabytes of data / year
- Mostly read-only
- Very high data rates (MB/sec, GB/sec)
- World wide distribution of data and processing facilities
  - Regional Centers (RCs, Universities)

# The Data Grid

---

- Specialization and extension of the “Grid” for data-intensive distributed computation
- Define
  - Requirements
  - Components
  - APIs

# General Requirements for Data Grid

---

Provide Grid middleware services supporting the I/O intensive world-wide distributed future experiments

(High Energy Physics, Earth Observation, Bioinformatics, ...)

- Manage and share peta-bytes of data
- Distributed
- Scalable
- Uniform interface
- High throughput and high-speed transfers
- Security
- Replication
- Performance

# Related Work

---

- Distributed file systems
  - NFS, AFS
    - ◆ Convenient interfaces for remote I/O
    - ◆ No support for multi-site replication
    - ◆ Lack of collective I/O functionalities: batch I/O, scheduled I/O
- Parallel file systems
  - Vesta, Gallery
    - ◆ Provide collective I/O
    - ◆ Do not address complex configurations, security
- High Performance Storage System (HPSS)
- High-speed disk caches and cache management
- Remote execution systems
  - Location independent execution
  - No support for parallel I/O

# Related Work

---

- Distributed Databases
  - Replication research focuses on update synchronization, less on transferring large amounts of data
- Globus and Legion
  - Computational grid projects that start adding support for distributed data management
  - GASS (Global Access to Secondary Storage) in Globus
- Particle Physics Data Grid
  - High speed data transfers, transparent access, replica management
- Storage Resource Broker (SRB)
  - Integrates diverse storage systems under uniform metadata-driven access mechanisms
  - Uniform interface
  - Metadata Catalog (MCAT)
- Metadata standards and retrieval mechanisms

# Outline

---

- Motivation and Related Work
- **The Data Grid: An Architecture for Distributed Management and Analysis of Large Scientific Datasets**
- Data Management for Grid Environments
- Data Management in an International Grid Project
- Giggle: A framework for constructing scalable replica location services

# The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets

---

- Ann Chervenak, Carl Kesselman
  - ◆ Information Sciences Institute, University of Southern California
- Charles Salisbury, Steven Tuecke
  - ◆ Mathematics and Computer Science Division, Argonne National Laboratory
- Ian Foster
  - ◆ Mathematics and Computer Science Division, Argonne National Laboratory
  - ◆ Department of Computer Science, The University of Chicago

*Journal of Network and Computer Applications*, 23:187-200, 2001

- Data grid design
- Core data grid services
- Higher-level data grid components
- Implementation

# Data Grid Design

---

- Data grid applications environment
  - Wide area
  - Multi-institutional
  - Heterogeneous
  - Cannot assume uniformity of policy or behavior
  
- Design Principles
  - Mechanism neutrality
  - Policy Neutrality
  - Compatibility with Grid infrastructure
  - Uniformity of information infrastructure

# Data Grid Design (2)

---

- Mechanism Neutrality
  - Independence of low-level mechanisms
    - ◆ Data storage
    - ◆ Metadata storage
    - ◆ Data transfer, ...
  - Achieved by defining interfaces that hide the peculiarities of the above mechanisms
    - ◆ Data access
    - ◆ Third-party data mover
    - ◆ Catalog access

# Data Grid Design (3)

---

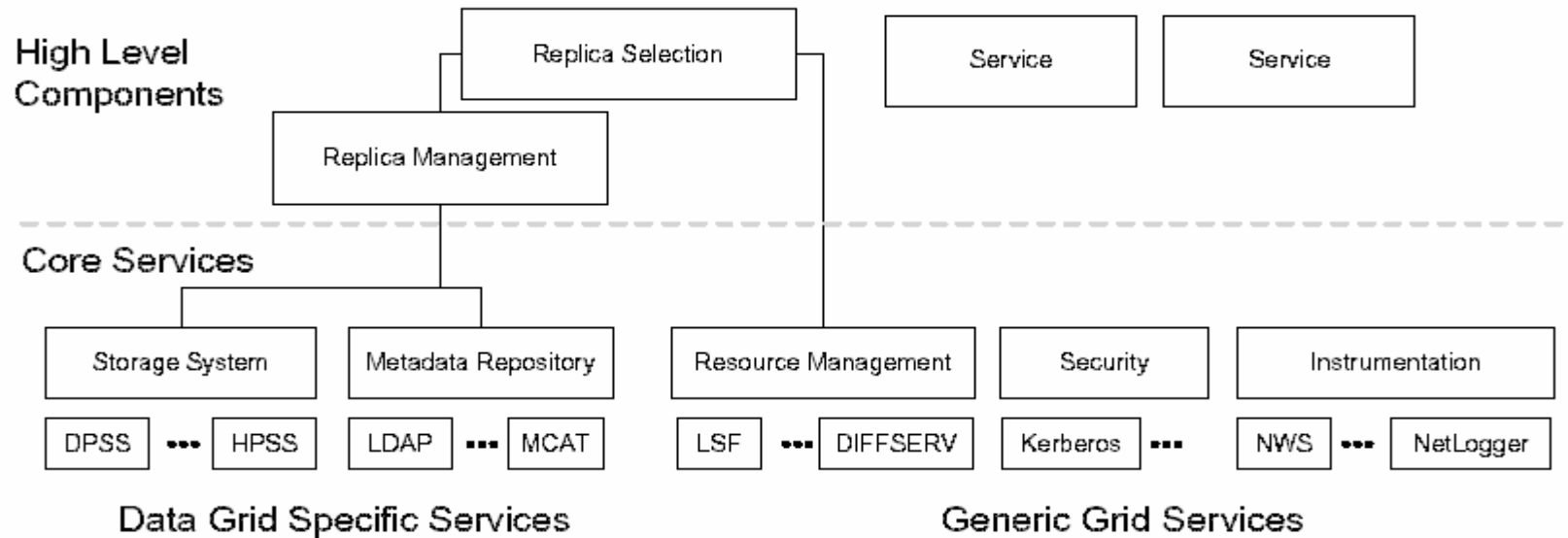
- Policy Neutrality
  - Design decisions with significant performance implications are exposed to the user rather than encapsulated in “black box” implementations
  - Example:
    - ◆ Provides basic operations for
      - Data movement
      - Replica cataloging
    - ◆ Higher-level procedures that can be substituted with application-specific code for
      - Replication policies

## Data Grid Design (4)

---

- Compatibility with Grid infrastructure
  - Exploits underlying Grid infrastructure (e.g. Globus)
    - ◆ Authentication
    - ◆ Resource management
  - More specialized data grid tools are compatible with lower-level Grid mechanisms
- Uniformity of information infrastructure
  - Emphasize uniform and convenient access to information about resources (structure and state) as a means of enabling run-time adaptation
  - Uses the same data model and interface as the underlying Grid information infrastructure

# Data Grid Design (5)



## ■ Layered architecture

### ◆ Lowest layers

- High-performance access to a set of basic mechanisms
- Do not enforce specific usage policies

### ◆ Higher layers

- Implement usage policies
- Build on the mechanisms provided by lower layers

# Core Services

---

Focus on two fundamental services:

- **Data access**
  - ◆ Access
  - ◆ Management
  - ◆ Initializing third-party transfers  
of data stored in storage systems
- **Metadata access**
  - ◆ Access
  - ◆ Management
  - of information about data stored in storage systems

Separation of data and metadata enhances flexibility in storage system implementation while having minimum impact on implementation of behaviors that combine metadata access with storage access

# Storage Systems and Data Access

---

Mechanism neutrality – applications should not need to be aware of the specific low-level mechanisms required to access data

- Storage system = data abstraction
  - Basic component that provides functions for **creating, destroying, reading, writing and manipulating file instances**
  - File instance
    - ◆ basic unit of information in a storage system
    - ◆ Named, un-interpreted sequence of bytes
    - ◆ Does not need to reside on a file system
  - Associates a set of properties with each file instance
    - Name, attributes, access restrictions, size, ...
    - The name is arbitrary and has meaning only to that particular storage system (path, application metadata)
  - Logical definition
    - ◆ Can be implemented using any storage technology that supports the required access functions (UNIX file systems, HTTP servers, HPSS,...)

# Storage Systems and Data Access (2)

---

The behavior of a storage system as seen by the data grid user is defined by an API that describes the possible operations on storage systems and file instances

- **Data Access API**

- Requests for reading/writing a named file instance
- Determine file instance attributes (size, ...)
- Third Party Transfer operation
  - ◆ Transfer the entire content of a file instance from a storage system to another
  - ◆ Used in replica management services
- Additional requirements introduced by data grid considerations
  - ◆ Access functions integrated with each site's security environment
  - ◆ Reservation capabilities
  - ◆ Characterizing and monitoring their own performance
  - ◆ Data movement functions should be able to detect and report errors

# The Metadata Service

---

- Metadata
  - information about the data grid itself, file instances, storage systems, contents of file instances
  - Types:
    - ◆ Application metadata
      - Defines the logical structure or semantics that should be applied to a file instance
      - Example: information content, circumstances under which the data was obtained
    - ◆ Replica metadata
      - Used in the management of replicated objects
      - Example: information for mapping a file instance to a particular storage system location
    - ◆ System configuration metadata
      - Describes the fabric of the data grid itself
      - Example: network connectivity, details about storage systems (capacity, usage policy)

## The Metadata Service (2)

---

- The Metadata Service
  - Uniform means for naming, publishing and accessing different types of metadata
  - Applications identify files of interest by posing queries to the metadata service that includes a metadata repository (catalog)
    - ◆ The query specifies the characteristics of the desired data
    - ◆ The metadata repository associates data characteristics with logical files (entities with globally unique names and one or more physical instances)
  - The replica manager identifies the physical file corresponding to the logical file, using replica metadata
  - Implementation
    - ◆ Distributed directory service
    - ◆ LDAP

# Other Basic Services

---

The Data Grid architecture assumes the existence of a number of other basic services:

- **Authorization and authentication**
  - multi-institutional operation
  - Public key-based GSI
- **Resource reservation and co-allocation**
  - End-to-end performance guarantees
- **Performance measurement and estimation**
  - E.g. Network Weather Service
- **Instrumentation services**
  - End-to-end instrumentation of storage transfers

# Higher-Level Data Grid Components

---

- **Replica Management**
  - Create or delete copies of file instances (replicas) from specified storage systems
  - Repository (catalog) that associates a logical file with one or more replicas
  - Hierarchy of replica catalogs
- **Replica Selection and Data Filtering**
  - Chooses the replica that will provide the application with the desired data access characteristics: speed, cost, security, ...
    - ◆ Can use GIS (network bandwidth) , metadata repository (file size) to determine “the best” replica
  - Provides access to subsets of a file instance

# Implementation

---

Initial design of catalogs for metadata and replica management for two applications: Climate Modeling and Data Visualization

- **LDAP (Lightweight Directory Access Protocol)**

LDAP catalog organized in a tree structure – Directory Information Tree (DIT)

- **Climate Modeling**

- One LDAP catalog for both metadata and replica information
- DIT
  - One root node
  - A node for each of four collections
  - A node for each logical file in the collection
- Node
  - Application metadata in XML format
  - Storage system, path name and a template for constructing the instance file name - > URL
- URL passed to a file transfer interface which moved the data into a local cache
- Demonstrated the ability to map a query based on application metadata into a URL using an LDAP catalog and to move the data using a general purpose transfer API



# Status of the Data Grid Implementation

---

- Preliminary design of the data access API
  - Create, delete, open, close, read, write on file instances
  - Storage to storage transfers
  - Implemented interfaces to: local files, HTTP, FTP, DSPP network disk caches
- Implemented replica management and metadata services
  - LDAP

# Outline

---

- Motivation and Related Work
- The Data Grid: An Architecture for Distributed Management and Analysis of Large Scientific Datasets
- **Data Management for Grid Environments**
- Data Management in an International Grid Project
- Giggle: A framework for constructing scalable replica location services

# Data Management for Grid Environments

---

- Heinz Stockinger
  - ◆ CERN, Switzerland
- Omer F. Rana
  - ◆ Cardiff University, UK
- Reagan Moore
  - ◆ San Diego SuperComputer Centre, USA
- Andre Merzky
  - ◆ Konrad Zuse Zentrum, Berlin, Germany

*Lecture Notes in Computer Science, Volume 2110 2001*

- Data Management – A Wider Perspective
- Support for Data Storage and Access
- Support and Application Services
- Supporting Primitive and Global Services
- Data Formats

# Data Management

---

## Common operations involved in data management

- Data pre-processing and formatting
  - ◆ Transforming raw data into a useful form
  - ◆ Filling in missing data
- Data fusion
  - ◆ Combining different types of data to produce a single unified dataset
- Data splitting
  - ◆ Dividing a dataset into smaller pieces that can be analyzed in parallel
- Data storage
  - ◆ Recording data on various storage mediums
  - ◆ Can involve migration, replication
- Data analysis
  - ◆ Different computational approaches
- Query estimation and optimization
  - ◆ Essential when dealing with huge amounts of data
- Visualization, Navigation and Steering
  - ◆ Visualize and interact with the results of a computation

# Support for Data Storage and Access

- Categorizing storage devices
  - Policy
    - ◆ modifiable by the user whenever possible
  - Operations
    - ◆ Minimum of services
    - ◆ Read, write, lookup, ...
  - State management
    - ◆ Needed by transaction mechanisms
  - Mechanisms
    - ◆ Actual implementation of operations
  - Errors/Exceptions
    - ◆ resource level and global level
  - Structure
    - ◆ Physical organization of a storage resource

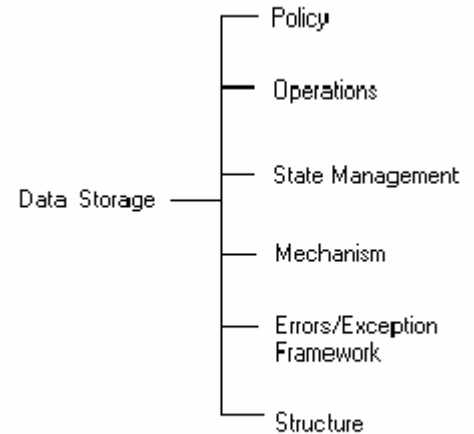


Figure 1. *Categorising Storage Devices*

# Support for Data Storage and Access (2)

---

- Local and Remote Data Access
  - Unified interface desired
  - Local access preferred
  - Allow users to manage data manually
- Minimum access unit
  - Types of data units
    - ◆ Primitive types
      - Float, integer,...
      - Arrays, binary objects
    - ◆ Files
      - Un-interpreted sequence of bytes
- Access patterns
  - Business and commercial computing - random access to single units of data
  - Scientific computing – more regular, sequential processing of large data units

# Metadata management and Standards

---

- Metadata
  - Properties of resources
  - Permissions, access rights
  - Information about stored content
  - Relationship representation
    - ◆ Semantic/functional, spatial/structural, temporal/procedural
- Standards
  - IEEE Open Storage Systems Interconnection (OSSI)
    - ◆ Technical details of mass storage systems
    - ◆ Specifications for storage media, drive technology, data management software
    - ◆ Recently: organizational functions, connections and interactions with other storage systems
    - ◆ Independent of current technologies

## Standards (continued)

---

- ISO Open Archival Information System (OAIS)
  - Long term archiving
  - Data submission, storage and dissemination
- ISO 13250 Topic Maps
  - Standardized notation for representing information about the structure of resources used to define topics and relationships between topics
  - Topic map = a set of related documents that employs that notation
    - ◆ Groupings of addressable objects around topics (occurrences)
    - ◆ Relationships between topics (associations)
  - Topic space
    - ◆ Locations are topics
    - ◆ Distance between topics = number of intervening topics that had to be visited to get from one to the other + the kind of relationships that define this path

# Support and Application Services

---

Data management for Grid-enabled applications must be based on applications in different domains

- High energy physics
  - See 7
- 2MASS 2-micron all sky survey
  - 5 million images, 10 terabytes of data
  - Need to be sorted from a temporal order as seen by the telescope to a spatial order

# Supporting Primitive and Global Services

---

Identifying core services which should be supported by all storage resources to be employed for Grid-enabled applications

- Data access operations
  - Read, write
  - Access to collections of primitive data types
- Local transparency and global namespace
  - Discover the location of a data source and keep that location independent of its access method
  - Interoperability
  - Global namespace
    - ◆ Can be provided through a catalogue (e.g. CERN's Objectivity catalogue)
    - ◆ Or aggregation in containers (digital objects)

# Supporting Primitive and Global Services (2)

---

- Privilege and security operations
  - Enable individuals to access particular data sources without having an account on the system hosting the data source
    - ◆ The collection owning the data has an account on the storage system
    - ◆ Access control at object level within the collection
    - ◆ Easier to manage
    - ◆ Independent of access control of the underlying storage system
- Persistence and Replication
  - URL's cannot be used with distributed data since it includes storage location and access protocol
  - Persistence may be managed by moving all data objects within a data handling system
    - ◆ Implies a hierarchy of data management systems
      - Storage systems
      - Data handling systems
      - Information repository systems
      - Digital libraries

## Supporting Primitive and Global Services (3)

---

- Error and exception handling
  - Central handling
  - Resource level handling
  - Both
- Check pointing and state management
  - Central service that records the state of transactions
  - Users can subscribe to it to ensure state recovery and restart of operation in the event of a failure

# Data formats

---

- Common data formats required to support inter-application exchange of data
- Support for easy conversion between formats

# Outline

---

- Motivation and Related Work
- The Data Grid: An Architecture for Distributed Management and Analysis of Large Scientific Datasets
- Data Management for Grid Environments
- **Data Management in an International Grid Project**
- Giggle: A framework for constructing scalable replica location services

# Data Management in an International Data Grid Project

---

- Wolfgang Hoschek
  - ◆ CERN, European Organization for Nuclear Research, Geneva, Switzerland
  - ◆ Inst. of Applied Computer Science, University of Linz, Austria
- Javier Jaen-Martinez
  - ◆ CERN, European Organization for Nuclear Research, Geneva, Switzerland
- Asad Samar
  - ◆ CERN, European Organization for Nuclear Research, Geneva, Switzerland
  - ◆ California Institute of Technology, Pasadena, CA, USA
- Heinz Stockinger, Kurt Stockinger
  - ◆ CERN, European Organization for Nuclear Research, Geneva, Switzerland
  - ◆ Inst. for Computer Science and Business Informatics, University of Vienna, Austria

*Proceedings of the first IEEE/ACM International Workshop on Grid Computing, (Springer Verlag Press, Germany), India, 2000*

- Architecture
- Data management components

# Architecture

---

- Requirements
  - Easy to understand and maintain
  - Flexibility
    - ◆ Different organizations can use their own packages
  - Allow for rapid prototyping
    - ◆ Use as much of the previous work as possible
  - Scalable to massive high throughput use cases
    - ◆ Careful design and layering
  - Distributed development
    - ◆ Well defined and loosely coupled components

# Architecture (2)

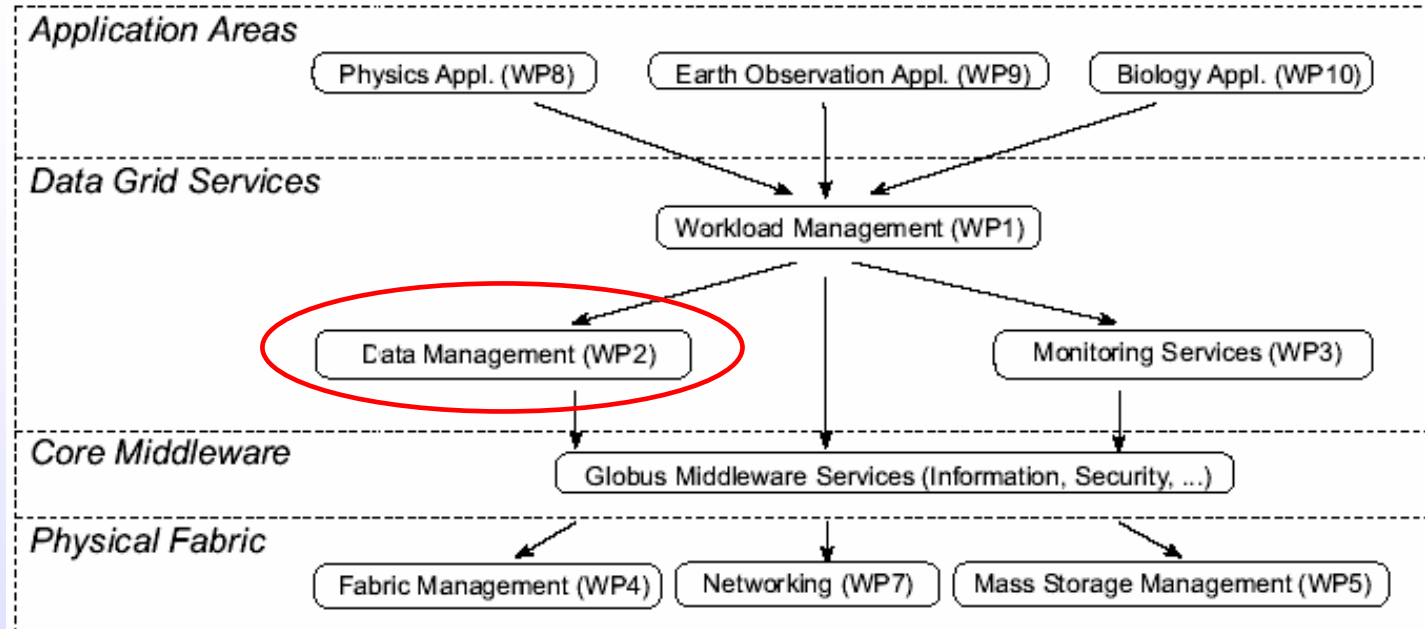


Fig. 2. Overall interaction of project work packages

- Applications
  - Directly use data grid services
  - Require transparent access to distributed data and computing
- Workload management
  - Distributed scheduling and resource management
- Data management
  - Share huge data sets in high-throughput environment
- Monitoring
  - Access to status and error information
- Globus services
  - Core middleware
- Fabric management
  - Manage computing centers providing Grid services
- Networking
  - Provides a virtual private network between the involved resources
- Mass Storage Management
  - Interfaces with the existing Mass Storage Management Systems (MSMS)

# Data Management Package

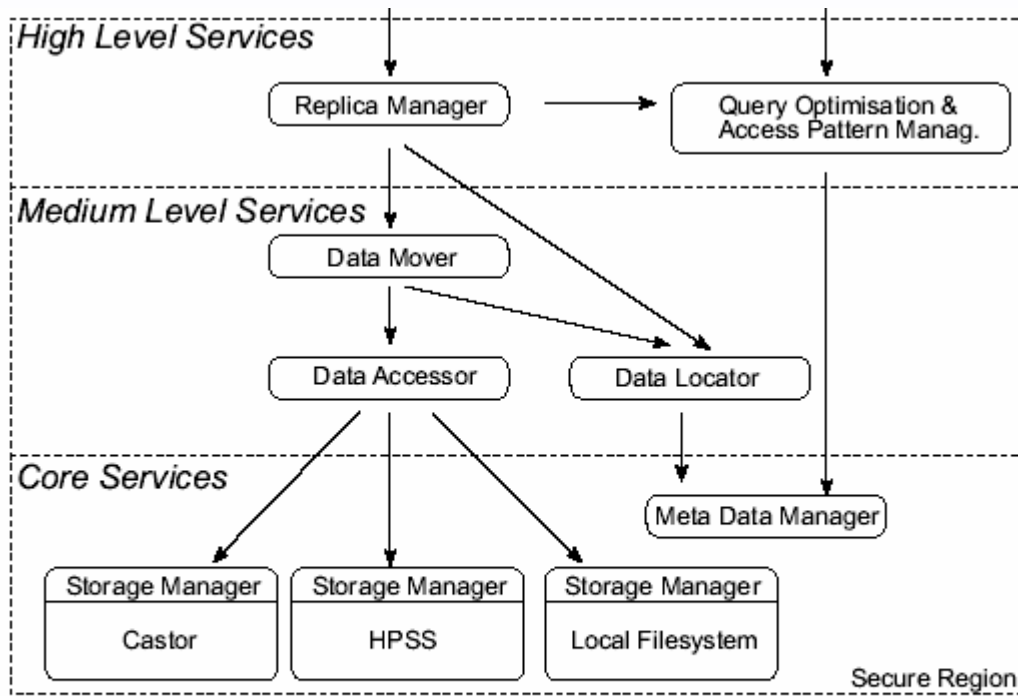


Fig. 3. Overall interaction of data management components

## Layered set of services

- **Replica management**
  - Manages file and metadata copies
  - Customizable replication policies
- **Data mover**
  - Transfers files from one storage system to another
- **Data Accessor**
  - Interface to local storage systems
- **Data Locator**
  - Maps location independent identifiers with location dependent identifiers
- **Meta Data manager**
  - Publish and manage a hierarchical set of objects
- **Query Optimization and Access Pattern Manager**
  - Ensure optimal migration and replication plan for each query

# Data Management Components

---

- Data Accessor
  - Heterogeneity of data repositories
    - ◆ HPSS, Castor, UniTree, Enstore, , DPSS, NFS, AFS, databases
    - ◆ Data identified by file name or attributes
  - Initial work concentrates on
    - ◆ Hierarchical Storage Management systems
      - Data is distributed across a storage hierarchy comprised of high-speed disks, compressed disks and tapes
      - Perfect for HEP application
    - ◆ Local File systems
  - Single interface for accessing different underlying repositories
  - Specific implementation for each storage system that will provide maximum of performance

# Data Management Components (2)

---

- Replication
  - Management of data copies
  - Caching strategy
    - ◆ Better response time by accessing locally cached copies of data
  - Fault tolerance
    - ◆ Availability of data
  - Replication policies
    - ◆ Distributed: administrators, schedulers, storage systems
  - Different requirements for data and meta data replication
    - ◆ Data replication
      - Fast point-to-point file replication mechanisms
    - ◆ Meta data replication
      - Client-server communication mechanism at each Grid site
        - ✓ Globus – Sockets, Nexus library
  - Synchronous or asynchronous updates for data consistency
  - Uses the Data locator service to access physical files
  - Replicas are created and purged at remote sites based on user access patterns – Grid cache

# Data Management Components (3)

---

- Meta Data
  - Types
    - ◆ Catalogues
      - Name and locations of unique and replicated files and indexes
    - ◆ Monitoring information
      - Error status, current and historical throughput, query access patterns
    - ◆ Configuration information
      - Networks, switches, nodes, clusters, software
    - ◆ Policies
      - Dynamic steering
- Meta Data Management Service
  - Large number of objects referenced by identifiers
  - Built on LDAP

# Data Management Components (4)

---

- Security
  - Grid cache
    - ◆ Remote sites hosting replicas of some data should provide the same level of security as the site that owns the data
    - ◆ Security infrastructures will most likely be different across sites
  - Synchronous vs. asynchronous replication
    - ◆ Synchronous replication
      - Updates to one replica propagate to all copies
      - Require permanent write access to other sites
    - ◆ Asynchronous replication
      - Updates are only performed on demand
      - Low consistency
  - Sensitivity levels of data and meta data might be different
  - Involved in replica selection

# Data Management Components (5)

---

- Query Optimization
  - Considers multiple copies of a file
  - Minimizes a cost model: response time, throughput
  - Influencing factors:
    - ◆ Size of file to be accessed
    - ◆ Load on the data server to serve the file
    - ◆ Method/Protocol by which the file is accessed and transferred
    - ◆ Network bandwidth, distance and traffic
    - ◆ Remote access policies
  - Makes use of the Meta Data Management Service
  - Further complications when queries request single objects contained in files
    - ◆ Mapping between object identifiers and files

# Outline

---

- Motivation and Related Work
- The Data Grid: An Architecture for Distributed Management and Analysis of Large Scientific Datasets
- Data Management for Grid Environments
- Data Management in an International Grid Project
- **Giggle: A framework for constructing scalable replica location services**

# Giggle: A Framework for Constructing Scalable Replica Location Services

---

- Ann Chervenak, Ewa Deelman, Carl Kesselman, Bob Schwartzkopf
  - ◆ Information Sciences Institute, University of Southern California, Marina del Rey, CA 90292
- Ian Foster
  - ◆ Department of Computer Science, University of Chicago, Chicago, IL 60637
  - ◆ Mathematics and Computer Science Division, Argonne National Laboratory, Argonne, IL 60439
- Leanne Guy, Wolfgang Hoschek, Peter Kunszt, Heinz Stockinger, Kurt Stockinger
  - ◆ CERN, European Organization for Nuclear Research, CH-1211 Geneva 23, Switzerland
- Adriana Iamnitchi, Matei Ripeanu
  - ◆ Department of Computer Science, University of Chicago, Chicago, IL 60637
- Brian Tierney
  - ◆ Lawrence Berkeley National Laboratory

*Proceeding of the IEEE Supercomputing 2002, November 2002*

- Requirements
- The Giggle Replica Location Service Framework
- Implementation Approaches
- Relaxed Consistency

# Giggle

---

- Giggle = GIGa-scale Global Location Engine
  - A framework for creating replica location service
  - Defines a parameterized set of basic mechanisms
  - Reliability, storage, communication overhead as well as update and access costs can be controlled through parameters
  - Replica location problem
    - ◆ Given a unique logical identifier for desired data content, determine the physical location of one or more copies of this content
  - Replica location service
    - ◆ Maintains and provides access to information about the physical locations of copies

# Definitions

---

- Replica
  - remote read-only copy of an object (file)
- Logical file name (LFN)
  - unique logical identifier for desired data content
- Physical file name (PFN)
  - identifier of a physical copy

# Requirements

---

- Read-only vs. Mutable data
  - Usually large data sets are read-only once they are published
  - Assumes read-only except in the context of replica creation/deletion
  - Mutable data can be supported by versioning
- Scale, Size and Performance
  - Approximations for the HEP experiments:
    - ◆ Up to 200 replica sites
    - ◆ 50 million logical files catalogued
    - ◆ 500 million physical files (replicas)
    - ◆ 20 million physical files at a site
    - ◆ Average query response time of 10 milliseconds
    - ◆ Maximum query response time of 5 seconds
    - ◆ Maximum query rates of 100 to 1000 per second
    - ◆ Maximum update/insertion rate of 50 to 200 per second

# Requirements (2)

---

- Security
  - Privacy
    - ◆ Control knowledge about the existence, location and content of data
  - Integrity
    - ◆ Prevent tampering with the results returned by RLS queries and the content of the replicas
  - RLS concerned with protecting the knowledge about data, while storage systems deal with protecting the data contents
- Consistency
  - Completely consistent view of replicas is not essential
    - ◆ replication is concerned with performance not logical functions
  - Enable appropriate tradeoffs between the cost of inconsistency and RLS implementation and operation cost

# Requirements (3)

---

- Reliability
  - No single point of failure
    - ◆ Service can still operate if any one of the nodes fails
  - Decoupling of local and global state
    - ◆ Access to local replicas should not be affected by failure of remote components

# The Framework

---

- Five essential elements
  - Independent local state
    - ◆ Local Replica Catalog (LRC) maintained at each site
      - Records LFN and PFN for all replicas
  - Unreliable global state
    - ◆ Replica Location Indices (RLI)
      - Maintains additional index state
      - Process queries concerning location of replicas
      - Are not required to maintain consistent state
  - Soft state maintenance of RLI state
    - ◆ Propagation of information from LRCs to RLIs through soft state techniques (i.e. information that times out)
  - Compression of state updates
    - ◆ Compress LRC data before transmitting it to RLI to minimize traffic
  - Membership service
    - ◆ Locates participating replica sites and RLIs

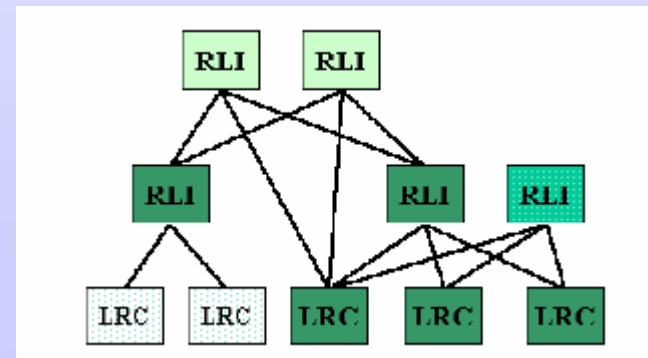
# Local Replica Catalogs - LRC

---

- Information about replicas at a single replica site
- Mapping between arbitrary LFNs and PFNs on the associated storage systems
- Requirements
  - Contents
    - ◆ Must maintain the LFN to PFN mapping
  - Queries
    - ◆ PFN from LFN, LFN from PFN
  - Local integrity
    - ◆ Maintain consistency with the underlying storage system
  - Access control
    - ◆ Authentication and authorization through GSI and CAS (Community Authorization Server)
  - State propagation
    - ◆ Periodically send state information to RLIs
- Virtual Organization Model
  - Multiple RLCs associated with the same storage system, each for a different VO
  - Interact with VO specific RLIs

# The Global Replica Index - GRI

- A set of Global Replica Index Nodes (RLI)
- Each node contains a number (LFN, replica-site) entries
- Six parameters can be manipulated to define RLS structures
  - $G$  – number of RLIs
  - $P_L$  – The function used to partition the LFN name space
  - $P_R$  – The function used to partition the replica name space
  - $R$  – The degree of redundancy in the index space
  - $C$  – The function used to compress LRC information prior to sending
  - $S$  – The function used to determine what LRC information to send when
- Can have hierarchies of RLIs



# Parameters

Table 1: The six parameters used to characterize Giggie RLS structures, and some examples of possible values and their implications. See text for more details.

<b><math>G</math></b>	<b>The number of RLIs</b>	
	$G=1$	A centralized, non-redundant or partitioned index
	$G>1$	An index that includes partitioning and/or redundancy
	$G \geq N$	A highly decentralized index with much partitioning and/or redundancy
<b><math>P_L</math></b>	<b>The function used to partition the LFN name space</b>	
	$P_L = \phi$	No partitioning by LFN. The RLI(s) must have sufficient storage to record information about all LFNs, a potentially large number
	$P_L = \text{hash}$	“Random” partitioning. Good load balance, perhaps poor locality
	$P_L = \text{coll}$	Partitioning on collection name. Perhaps poor load balance, good locality
<b><math>P_R</math></b>	<b>The function used to partition the replica site name space</b>	
	$P_R = \phi$	No partitioning by site name. Indices have entries for every replica of every LFN they are responsible for. Potentially high storage requirements
	$P_R = \text{IP}$	Partition by domain name or similar. Potential geographic locality
<b><math>R</math></b>	<b>The degree of redundancy in the index space</b>	
	$R=1$	No redundancy: each replica is indexed by only one RLI
	$R=G>1$	Full index of all replicas at each RLI. Implies no partitioning, much redundancy/space overhead
	$1 < R < G$	Each replica indexed at multiple RLIs. Less redundancy/space overhead
<b><math>C</math></b>	<b>The function used to compress LRC information prior to sending</b>	
	$C = \phi$	No compression: RLIs receives full LFN/site information
	$C = \text{bloom}$	RLIs receive summaries with accuracy determined by Bloom parameters
	$C = \text{coll}$	RLIs receive summaries based on collection distribution
<b><math>S</math></b>	<b>The function used to determine what LRC information to send when</b>	
	$S = \text{full}$	Periodically send entire state (perhaps compressed) to relevant RLIs
	$S = \text{partial}$	In addition, send periodic summaries of updates, at a higher frequency.

# Global Index Structure Options

---

Different global index structures have different performance characteristics

No partition of index information

- $G = 1$  - Centralized global index
  - Simple to access and query
  - Not robust to failure
  - Not scalable
  - High storage cost at the central site
- $G > 1$ ,  $R = G$  and  $P_L = P_R = \emptyset$ 
  - Full index replicated at  $G$  sites
  - Storage and communication costs increase with  $G$
  - Load balancing and resilience to failure
- $G = N$  (# of replica sites),  $R = G$  and  $P_L = P_R = \emptyset$ 
  - Full index replicated at every replica site
  - High storage cost
  - Highly robust to failure

## Global Index Structure Options (2)

---

Partition the global index and do not include redundancy of index information

- $P_L = \text{hash}(\text{LFN}) \bmod G$ , and  $P_R = \emptyset$ 
  - Partition the LFN space based on LFN name
    - ◆ Each RLI contains information about all replicas of a subset of all LFNs
- $P_L = \emptyset$ ,  $P_R = \text{some\_fn}(\text{ip\_domain}(\text{replica\_site}))$ 
  - Partition the replica site name space based on some notion of geographical locality
  - Each RLI contains information about some replicas for all LFNs
- $P_L = \text{some\_fn}(\text{collection}(\text{LFN}))$  and  $P_R = \emptyset$ 
  - Partition LFN space by collection name
  - Each RLI contains information about all replicas of a subset of all LFNs grouped by some “collection name”
- $P_L = \emptyset$  and  $P_R = \emptyset$ 
  - Partition in two dimensions

# General Purpose Hybrid Strategy

---

- Simple RLS design that would perform well in many situations
  - Moderate number of RCIs
  - Moderate degree of redundancy
  - Partition of LFN space using a hash function
  - Properties
    - ◆ Clients can directly query for a LFN to any RLI that contains the relevant index information
    - ◆ A replica site updates the global index by sending a new index entry to each RLI responsible to the updated LFN – moderate communication for updates
    - ◆ Storage demands for RLIs are moderate

# Indexing with Redistribution

---

- If the set of RLIs changes over time
  - Membership protocol required
  - Redistribution of LFNs or replica sites to RLIs
    - ◆ Partitioning algorithm can use a hashing function
    - ◆ RLI identifiers passed through a hash function that returns a value from 0 to 360 (a circle)
    - ◆ LFNs (if  $P_L \neq \emptyset$ ) or replica sites (if  $P_R \neq \emptyset$ ) go through the same hash function and are associated with the closest RLI clockwise
    - ◆ If an RLI leaves, the LRC sends its soft-state updates to the next RLI clockwise

# Soft State Mechanisms and Relaxed Consistency

---

- How LRC updates propagate to the relevant RLIs
  - Strict consistency is not required
  - Use soft state protocol
  - Soft state information
    - ◆ information that times out
    - ◆ Must be periodically refreshed
    - ◆ Stale information is removed implicitly
    - ◆ Global index information need not be persistent
  - Can use data compression before transmitting
    - ◆ Reduce network traffic
  - Relaxed consistency
    - ◆ Metadata must be consistent
    - ◆ May not be able to locate a copy of a file that, in fact, exists
    - ◆ Need to be able to locate all copies of mutable files
    - ◆ Need to be able to restore the state of replica sites in the event of failure

# Requirements for Global Replica Index Nodes

---

- Secure remote access
  - An RLI must implement the Grid's GSI and CAS
- State propagation
  - RLIs must accept periodic state inputs from LRCs
- Queries
  - Must respond to queries asking for replicas of a given LFN
- Soft state
  - RLIs must implement time outs for the information stored in the index
  - Expired information is deleted
- Failure recovery
  - RLIs must not contain persistent replica state information
  - Recreate the contents after a failure solely based on updates from LRCs

# Membership Service

---

- Keeps track of currently active RLIs
- Responds to
  - Requests for a list of active RLIs in a given VO
  - Requests for asynchronous notification of changes in the list of active RLIs
- Possible implementations
  - Centralized server and a soft state registration protocol
  - Hierarchical organization
  - Decentralized soft state protocol based on gossip messages exchanged by RLIs

# Implementation Approaches

---

- RLS0: a centralized, non-redundant global index
  - Single RLI
  - LRCs send their full, uncompressed state to the RLI
- RLS1: LFN partitioning, redundancy, Bloom filters
  - Includes redundancy
  - Index information partitioned based on logical file name space
  - LRC state information is communicated in compressed form using Bloom filters
  - LRC state information is transmitted both as periodic full-state (compressed) and as interleaved partial-state updates
  - Locating replicas for a particular LFN
    - ◆ Determine the RLI that has the information
    - ◆ Contact the RLI which returns the set of LRCs containing replicas
    - ◆ Contact the LRCs to obtain physical file locations

# Implementation Approaches (2)

---

- RLS3: Referral service
  - Use of “extreme” compression techniques
  - The only information that an LRC sends is that it has files in the partition maintained by the RLI
  - RLI refers clients to all storage systems
  - Little network traffic, RLI logic is very simple
  - Potentially large numbers of LRCs returned to the client
- RLS4: Compression, partitioning based on collections
  - Partitioning of LFN name space based on domain-specific logical collections
  - The global index is also partitioned based on collections
  - RLIs contain information about which collection is replicated in which LRC
  - Reduction of storage space for RLIs
  - To determine whether a particular LFN is resident on a storage system the user has to query the LRC directly.

## Implementation Approaches (3)

---

- RLS5: Storage system partitioning, redundancy, Bloom filters
  - Partition the replica site name space based on some notion of geographical locality
- RLS6: A Hierarchical index
  - Extends RLS2 to include a second level of RLIs