the globus toolkit™
www.globustoolkit.org

Data Management Services in GT2 and GT3

Ann Chervenak
USC Information Sciences Institute

Requirements for Grid Data Management

- Terabytes or petabytes of data
  - Often read-only data, "published" by experiments
  - Other systems need to maintain data consistency
- Large data storage and computational resources shared by researchers around the world
  - Distinct administrative domains
  - Respect local and global policies governing how resources may be used
- Access raw experimental data
- Run simulations and analysis to create "derived" data products

Requirements for Grid Data Management (Cont.)

- Locate data
  - Record and query for existence of data
- Data access based on metadata
  - High-level attributes of data
- Support high-speed, reliable data movement
  - E.g., for efficient movement of large experimental data sets
- Support flexible data access
  - E.g., databases, hierarchical data formats (HDF), aggregation of small objects
- Data Filtering
  - Process data at storage system before transferring

Requirements for Grid Data Management (Cont.)

- Planning, scheduling and monitoring execution of data requests and computations
- Management of data replication
  - Register and query for replicas
  - Select the best replica for a data transfer
- Security
  - Protect data on storage systems
  - Support secure data transfers
  - Protect knowledge about existence of data
- Virtual data
  - Desired data may be stored on a storage system ("materialized") or created on demand

Outline

- Data architecture: layered, composable services
- Data transfer and access
  - **GridFTP:** Provides high-performance, reliable data transfer for modern WANs
  - **RFT:** Reliable File Transfer Service
- Data replication
  - **RLS:** Replica Location Service
  - Higher-level replication services
- OGSA Database Access and Integration Service
- Metadata Catalog Service

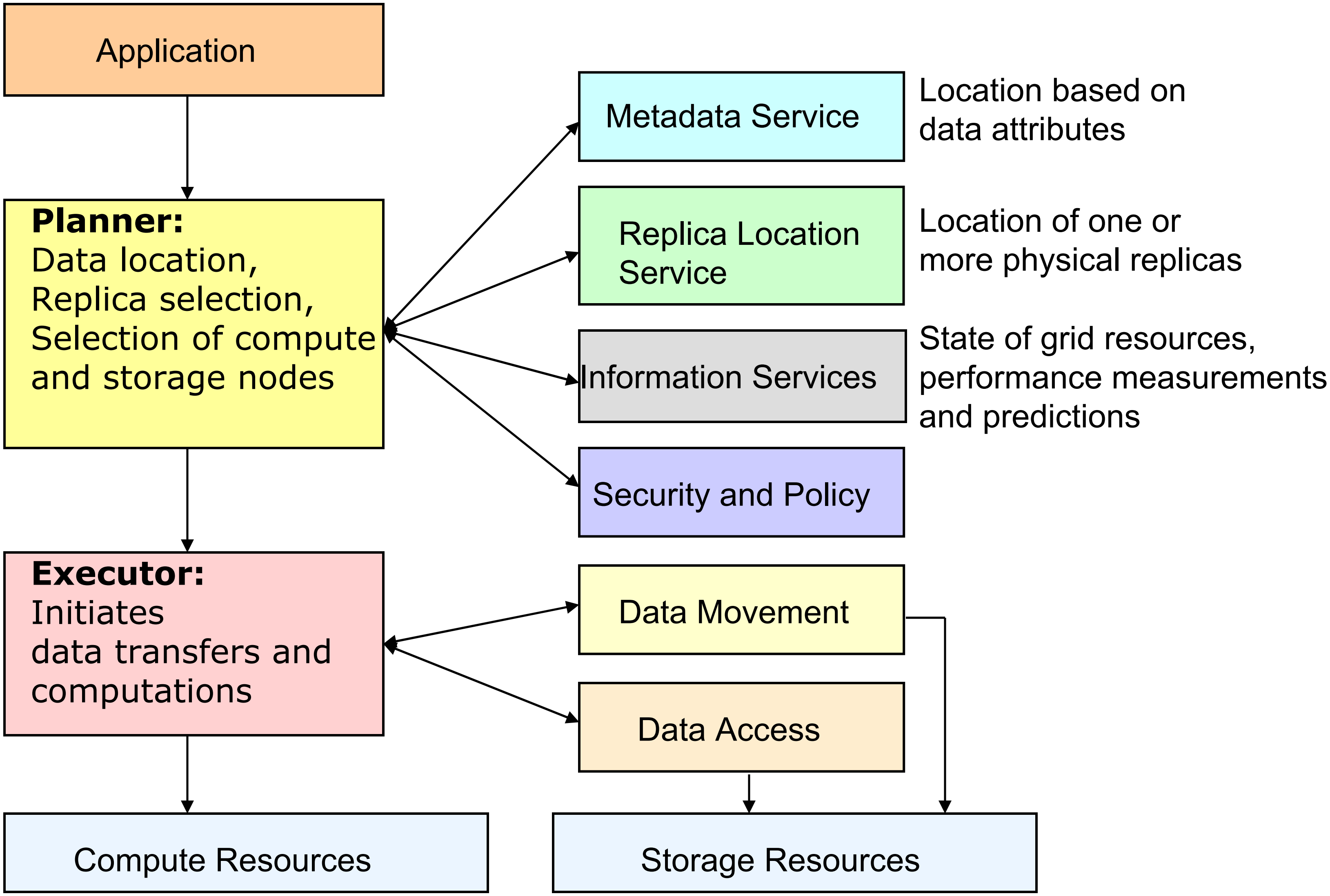
Overall Globus Architecture Philosophy

- The Globus toolkit provides a range of basic Grid services
  - Security, information services, resource management, data management...
- These services are simple and orthogonal
  - E.g., differentiate between Metadata Catalog Service and Replica Location Service
  - Can be used independently, mix and match
  - “Bag of Services” model
- Not a monolithic architecture
- Globus toolkit 2.0: well-defined APIs, extensive use of standards (X.509, LDAP, GSS-API)
- Globus toolkit 3.0: Open Grid Services Architecture

Key Concept: Composable Services

- Build core grid services
- Compose them to provide higher-level functionality
- Common set of underlying services deployed at sites
  - Used for a wide variety of purposes
- E.g., building a grid file system
  - Compose from basic, orthogonal services rather than implementing a "stovepipe" or complete vertical solution

Functional View of Grid Data Management



the globus toolkit™
www.globustoolkit.org

Architecture Layers

Collective 2: Services for coordinating multiple resources that are specific to an application domain or virtual organization (e.g., Authorization, Consistency, Workflow)

Collective 1: General services for coordinating multiple resources (e.g., RLS, MCS, RFT, Federation, Brokering)

Resource: sharing single resources (e.g., GridFTP, SRM, DBMS)

Connectivity (e.g., TCP/IP, GSI)

Fabric (e.g., storage, compute nodes, networks)

GridFTP

- Data-intensive grid applications need to transfer and replciate large data sets (terabytes, petabytes)
- GridFTP Features:
  - Third party (client mediated) transfer
  - Parallel transfers
  - Striped transfers
  - TCP buffer optimizations
  - Grid security

GridFTP: Basic Approach

- FTP protocol is defined by several IETF RFCs
- Start with most commonly used subset
  - Standard FTP: get/put etc., 3^{rd}-party transfer
- Implement standard but often unused features
  - GSS binding, extended directory listing, simple restart
- Extend in various ways, while preserving interoperability with existing servers
  - Striped/parallel data channels, partial file, automatic & manual TCP buffer setting, progress monitoring, extended restart

The GridFTP Protocol

- Based on 4 RFC’s and our extensions
- RFC 959: The base FTP protocol document
- RFC 2228: Security Extensions
- RFC 2389: Feature Negotiation and support for command options
- IETF Draft: Stream Mode restarts, standard file listings

GridFTP Implementation

- The GT2 GridFTP is based on the wuftpd server and client
- Ours is the only implementation right now
  - Likely to be others in the future
- Important feature is separation of control and data channels
- GridFTP is a Command Response Protocol
  - Issue a command
  - Get only responses to that command until it is completed
  - Then can issue another command

Command line tool: globus-url-copy

- This is the GridFTP client tool provided with the Globus Toolkit
- It takes a source URL and destination URL and will do protocol conversion for http, https, FTP, gsiftp, and file (file must be local).
- globus-url-copy sourceURL destURL
- globus-url-copy gsiftp://sourceHostName:port/dir1/dir2/file17 gsiftp://destHostName:port/dirX/dirY/fileA

GridFTP APIs

- globus_ftp_control
  - Provides access to low-level GridFTP control and data channel operations.
- globus_ftp_client
  - Provides typical GridFTP client operations.

globus_ftp_control

- Low level GridFTP driver
  - Control channel management
    - > Both client and server sides
    - > Handles message framing, security, etc
  - Data channel management
    - > Symmetric for client and server sides
    - > Designed for performance: caller controls buffer management, no data copies needed
- Must understand details of GridFTP protocol to use this API
  - Intended for custom GridFTP client and server developers

globus_ftp_client

- Functionality
  - get, put, third_party_transfer
    - Variants: normal, partial file, extended
  - delete, mkdir, rmdir, move
    - Note no "cd". All operations use URLs with full paths
  - list, verbose_list
  - modification_time, size, exists
  - Hides the state machine
  - PlugIn Architecture provides access to interesting events.
- All data transfer is to/from memory buffers
  - Facilitates wide range of clients

Example globus_ftp_client call

- globus_ftp_client_put/get/3rd Party
- Function signature:

globus_result_t **globus_ftp_client_get**
(**globus_ftp_client_handle_t** *handle,
const char *url,
globus_ftp_client_operationattr_t *attr,
globus_ftp_client_restart_marker_t *restart,
globus_ftp_client_complete_callback_t complete_callback,
void *callback_arg)

Writing a GridFTP Client

- Module Activation / Initialization
- Set Attributes (determine much of advanced functionality)
- Select Mode (stream or extended)
- Enable any needed plug-ins
- Execute the operation
- Module Deactivation / Clean up

the globus toolkit™
www.globustoolkit.org

Attributes

- Control much of advanced GridFTP functionality
- Functions
  - globus_ftp_client_operationattr_set_<attribute>(&attr, &<attribute_struct>)
  - globus_ftp_client_operationattr_get_<attribute>(&attr, &<attribute_struct>)
- Two types of attributes:
  - Handle Attributes: Apply for an entire session and independent of any specific operation
  - Operation Attributes: Apply for a single operation

Attributes (Cont)

- Handle Attributes:
  - Initialize/Destroy/Copy Attribute Handle
  - Connection Caching
  - Plugin Management: Add/Remove Plugins
- Operation Attributes
  - Parallelism
  - Striped Data Movement
  - Striped File Layout
  - TCP Buffer Control
  - File Type
  - Transfer Mode
  - Authorization/Privacy/Protection

Example Code: Setting Parallelism Attributes

```
globus_ftp_client_handle_t              handle;
globus_ftp_client_operationattr_t       attr;
globus_ftp_client_handleattr_t          handle_attr;
globus_size_t                           parallelism_level = 4;
globus_ftp_control_parallelism_t        parallelism;

globus_module_activate(GLOBUS_FTP_CLIENT_MODULE);
globus_ftp_client_handleattr_init(&handle_attr);
globus_ftp_client_operationattr_init(&attr);
parallelism.mode = GLOBUS_FTP_CONTROL_PARALLELISM_FIXED;
parallelism.fixed.size = parallelism_level;
globus_ftp_client_operationattr_set_mode(&attr,
      GLOBUS_FTP_CONTROL_MODE_EXTENDED_BLOCK);
globus_ftp_client_operationattr_set_parallelism(&attr, &parallelism);
globus_ftp_client_handle_init(&handle,  &handle_attr);
```

Mode S versus Mode E

- Mode S is stream mode as defined by RFC 959.
  - No advanced features except simple restart
- Mode E (extended mode) enables advanced functionality
  - Adds 64 bit offset and length fields to the header
  - This allows discontiguous, out-of-order transmission and enables parallelism and striping
- Command:

```
globus_ftp_client_operationattr_set_mode(&attr,
  GLOBUS_FTP_CONTROL_MODE_EXTENDED_BLOCK);
```

Plug-Ins

- Interface to one or more plug-ins:
  - Callouts for all interesting protocol events
    - Allows performance and failure monitoring
  - Callins to restart a transfer
    - Can build custom restart logic
- Included plug-ins:
  - Debug: Writes event log
  - Restart: Parameterized automatic restart
    - Retry N times, with a certain delay between each try
    - Give up after some amount of time
  - Performance: Real time performance data

End-to-end transfer performance may be limited by several factors

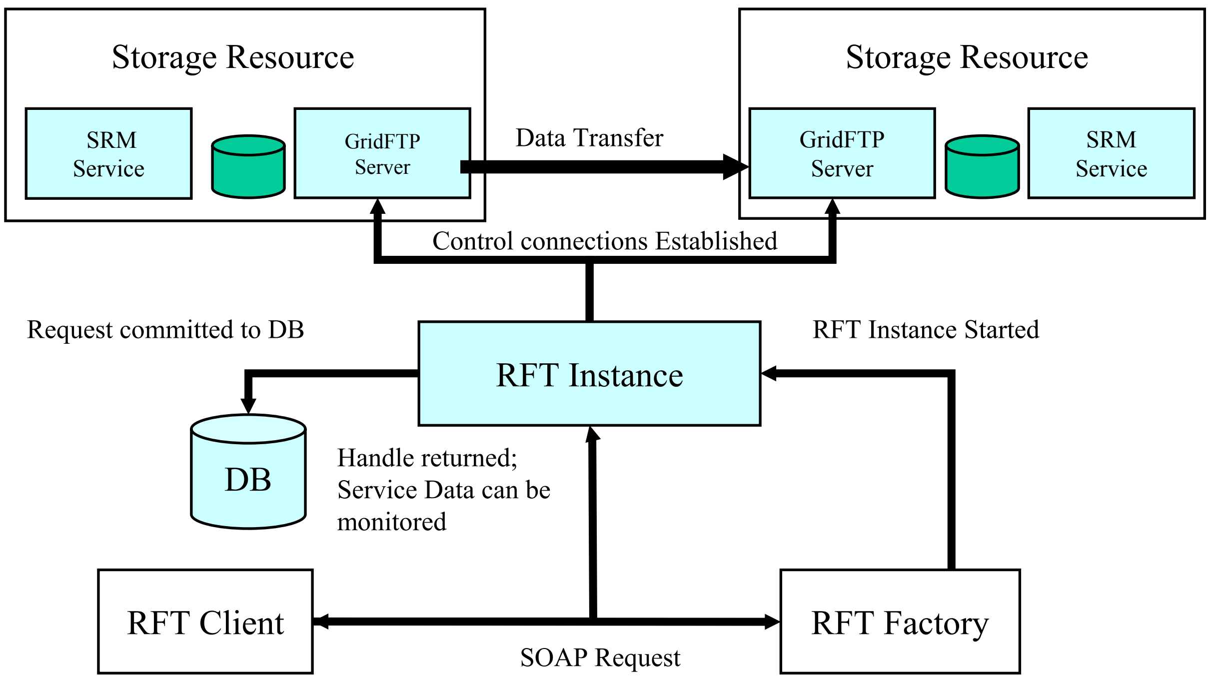
- OS Limitations on streams and buffers
  - Buffer size limits (defaults, Max)
  - We use 64K default, 8MB Max per socket
  - # of sockets per process and total
- Striping and parallelism may require lots of memory and streams
- NICs vary widely in performance
- Buses: Moving a lot of data: On/Off Disk, In/Out the NIC.
- CPUs: Fast network connections and software RAID require a lot of CPU
- Disk: can be the biggest bottleneck
  - RAID helps

GridFTP Development For GT3

- Major redesign planned
- Part 1: Replace existing globus_io libraries with XIO libraries (under development)
  - Pluggable protocol stack
  - TCP, reliable UDP, HTTP, GSI
- Part 2: GridFTP OGSA Service (?)
  - Based on redesign of GRAM job submission, service level agreements
  - Data transfer is just another type of job to be executed

RFT: Reliable File Transfer

- GT3 service
- Multiple-file version available in current release
- Allows monitoring and control of third-party data transfer operations between two GridFTP servers



the globus toolkit™
www.globustoolkit.org

RFT

- A client issues a request to an RFT factory
- Factory instantiates an RFT service instance
- The RFT instance does the following:
  - Communicates with two storage resources running GridFTP servers
  - Initiates a third-party transfer from source to destination GridFTP server
  - Monitors status of the transfer, updating the state describing the transfer in a database
- If the transfer fails because the client or one of the storage resources fails
  - Transfer state in RFT database is sufficient to resume or restart when resources become available

Outline

- Data architecture: layered, composable services
- Data transfer and access
  - **GridFTP:** Provides high-performance, reliable data transfer for modern WANs
  - **RFT:** Reliable File Transfer Service
- Data replication
  - **RLS:** Replica Location Service
  - Higher-level replication services
- OGSA Database Access and Integration Service
- Metadata Catalog Service

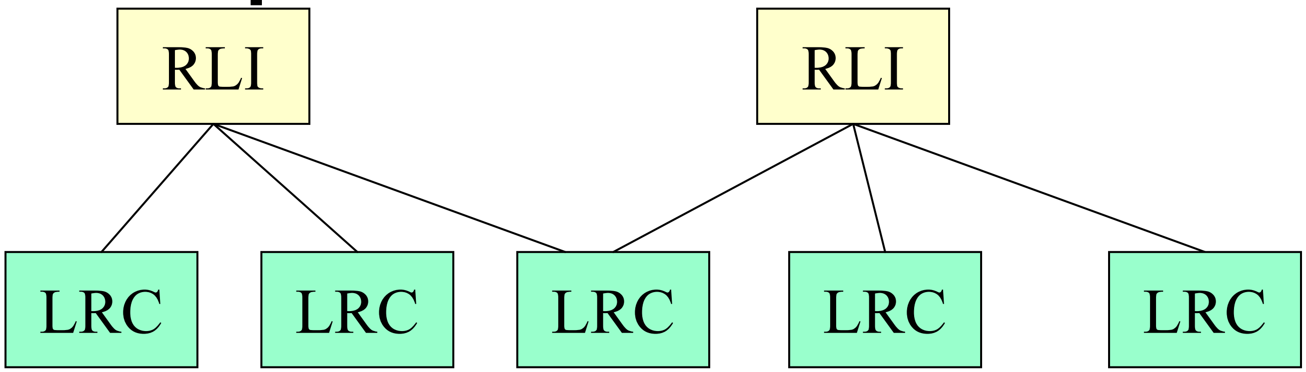
Replica Management in Grids

- Data intensive applications
  - Produce Terabytes or Petabytes of data
- Replicate data at multiple locations
  - Fault tolerance
  - Performance: avoid wide area data transfer latencies, achieve load balancing
- Issues:
  - Locating replicas of desired files
  - Creating new replicas
  - Scalability
  - Reliability

A Replica Location Service

- **A Replica Location Service (RLS)** is a distributed registry service that records the locations of data copies and allows discovery of replicas
- Maintains mappings between *logical* identifiers and *target names*
  - Physical targets: Map to exact locations of replicated data
  - Logical targets: Map to another layer of logical names, allowing storage systems to move data without informing the RLS
- RLS was designed and implemented in a collaboration between the Globus project and the DataGrid project

Replica Location Indexes



Local Replica Catalogs

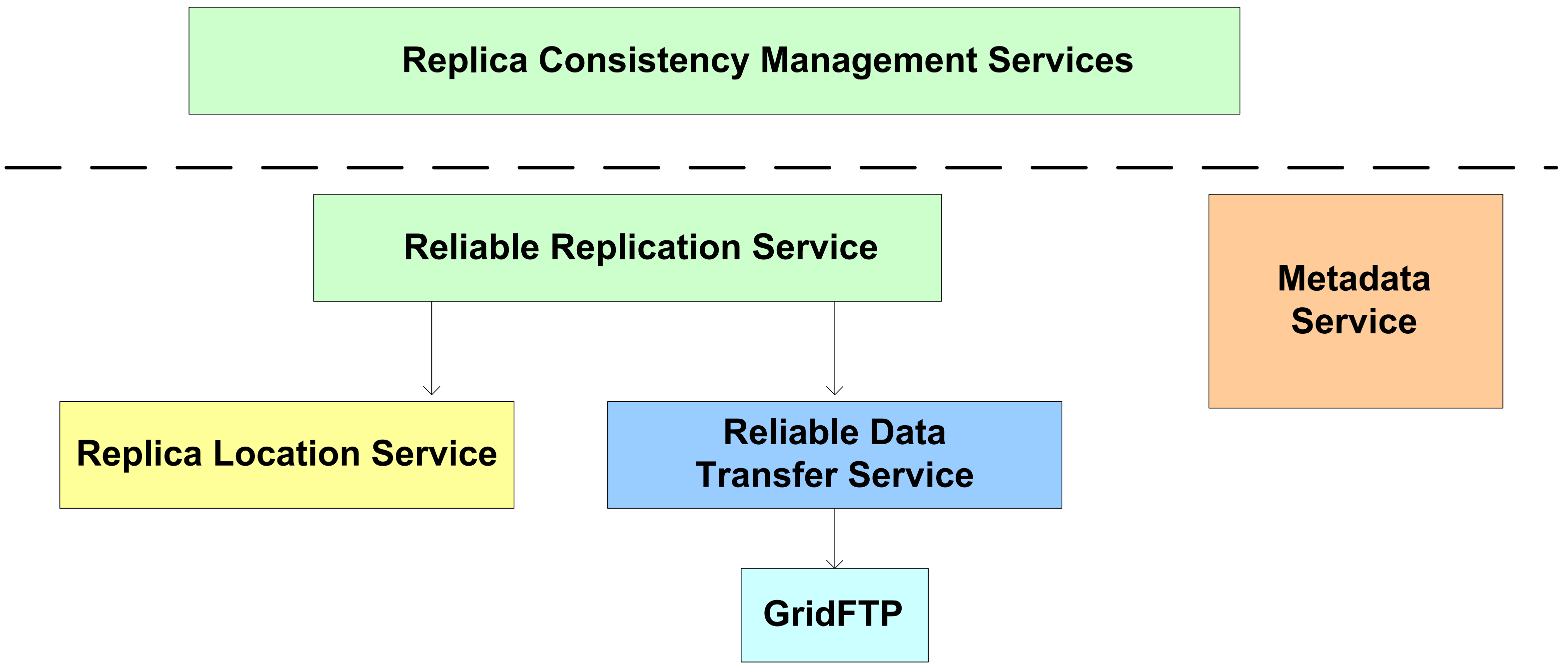
- LRCs contain consistent information about logical-to-target mappings on a site
- RLIs nodes aggregate information about LRCs
- Soft state updates from LRCs to RLIs: relaxed consistency of index information, used to rebuild index after failures
- Arbitrary levels of RLI hierarchy

A Flexible RLS Framework

Five elements:

1. Consistent Local State: Records mappings between logical names and target names and answers queries
2. Global State with relaxed consistency: Global index supports discovery of replicas at multiple sites; relaxed consistency
3. Soft state mechanisms for maintaining global state: LRCs send information about their mappings (state) to RLIs using soft state protocols
4. Compression of state updates (optional): reduce communication, CPU and storage overheads
5. Membership service: for location of participating LRCs and RLIs and dealing with changes in membership

Replica Location Service In Context

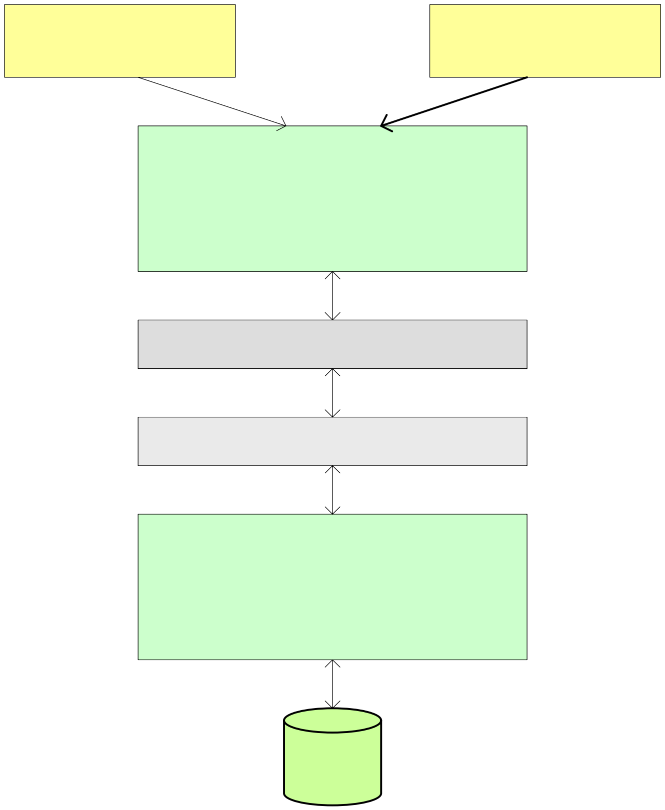


- The Replica Location Service is one component in a layered data management architecture
- Provides a simple, distributed registry of mappings
- Consistency management provided by higher-level services

the globus toolkit™
www.globustoolkit.org

Components of RLS Implementation

- **Front-End Server**
  - Multi-threaded
  - Supports GSI Authentication
  - Common implementation for LRC and RLI
- **Back-end Server**
  - mySQL or PostgreSQL Relational Database
  - Holds logical name to target name mappings
- **Client APIs: C and Java**
- **Client Command line tool**



Implementation Features

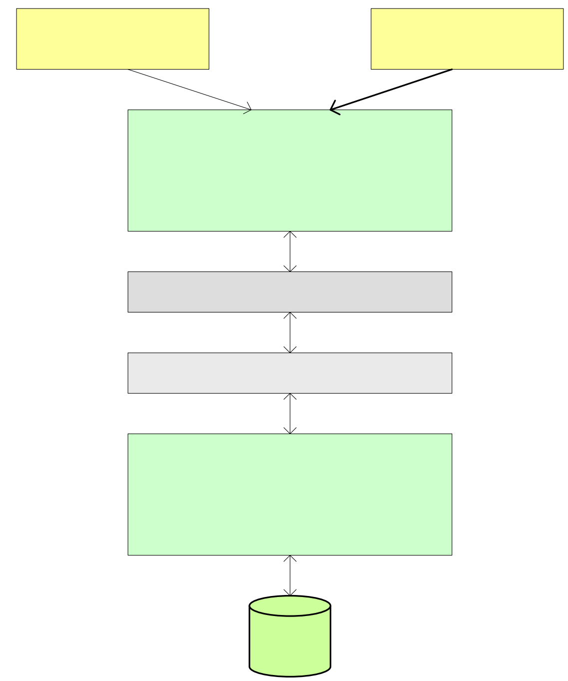
- Two types of soft state updates from LRCs to RLIs
  - Complete list of logical names registered in LRC
  - Bloom filter summaries of LRC
- Immediate mode
  - When active, sends updates of new entries after 30 seconds (default) or after 100 updates
- User-defined attributes
  - May be associated with logical or target names
- Partitioning (without bloom filters)
  - Divide LRC soft state updates among RLI index nodes using pattern matching of logical names
- Currently, static configuration only

Examples: Setting up an LRC and RLI and Sending a Soft State Update

1. Installing the LRC and RLI
2. Configuring soft state updates
3. Registering mappings in LRC
4. Querying the RLI and LRC

1. Installing the LRC and RLI

- First requires installing the underlying database
  - PostgreSQL, MySQL
- For each of these, must install both database and ODBC driver
- See RLS installation guide for instructions on RLS server installation
  - Requires latest Globus Packaging Toolkit (GPT)
  - Source and binary bundles
- Clients
  - C
  - Java (JNI wrapper, native Java client in progress)
  - Command line client tool



2. RLS Server and Soft State Update Configuration

- RLS server configuration
  - Whether an LRC or RLI or both
  - If LRC, configure
    - Method of soft state update to send (stored in database and set via command line tool)
    - May send updates of different types to different RLIs
    - Frequency of soft state updates (in config file)
  - If RLI, configure
    - Method of soft state update to accept (in config file)
- Can configure RLS server to act as a service provider to the MDS (Monitoring and Discovery Service)

2. Configuring Soft State Updates (Cont.)

- LFN List
  - Send list of Logical Names stored on LRC
  - Can do exact and wildcard searches on RLI
  - RLI must maintain a database and update database whenever new soft state update arrives
  - Soft state updates get increasingly expensive (space, network transfer time, CPU time on RLI to update RLI DB) as number of LRC entries increases
  - E.g., with 1 million entries, takes 20 minutes to update mySQL on dual-processor 2 GHz machine (CPU-limited in this case)

2. Configuring Soft State Updates (Cont.)

- Bloom filters
  - Construct a summary of LRC state by hashing logical names, creating a bitmap
  - Compression
  - Updates much smaller, faster
  - Can be stored in memory on RLI, no database
  - E.g., with 1 million entries, update takes less than 1 second
  - Supports higher query rate
  - Small probability of false positives (lossy compressions)
  - Lose ability to do wildcard queries

the globus toolkit™
www.globustoolkit.org

2. Configuring soft state updates (cont.)

- Whether or not to use Immediate Mode
  - Send updates after 30 seconds (configurable) or after fixed number (100 default) of updates
  - Full updates are sent at a reduced rate

- Immediate mode usually sends less data
  - Because of less frequent full updates

- Tradeoffs depend on volatility of data
  - Frequency of updates
  - Need to have fast updates of RLI vs. allowing some inconsistency between LRC and RLI content

- Usually advantageous
  - An exception would be initially loading of large database

3. Registering mappings in an LRC Using Client Command Line Tool

Command line client tool:

globus-rls-cli [-c] [-h] [-l reslimit] [-s] [-t timeout] [-u] [command] rls-server

– If command is not specified, enters interactive mode

- Create an initial mapping from a logical name to a target name:

globus-rls-cli create logicalName targetName1 rls://myrls.isi.edu

- Add a mapping from same logical name to a second replica/target name:

globus-rls-cli add logicalName targetName2 rls://myrls.isi.edu

Registering a mapping using C API

globus_module_activate(GLOBUS_RLS_CLIENT_MODULE)

globus_rls_client_connect (serverURL, serverHandle)

globus_rls_client_lrc_create (serverHandle, logicalName, targetName1)

globus_rls_client_lrc_add (serverHandle, logicalName, targetName2)

globus_rls_client_close (serverHandle)

the globus toolkit™
www.globustoolkit.org

Registering a mapping using Java API

```
RLSClient rls = new RLSClient(URLofServer);

RLSClient.LRC lrc = rls.getLRC();

lrc.create(logicalName, targetName1);

lrc.add(logicalName, targetName2);

rls.Close();
```

4. Querying mappings in an LRC or RLI using the Client Command Line Tool

- Query an LRC server for mappings of logical name

  globus-rls-cli query lrc lfn logicalName rls://mylrc.isi.edu

- Query an LRC server for mappings of target name

  globus-rls-cli query lrc pfn targetName2 rls://mylrc.isi.edu

- Query an RLI server for mappings of logical name

  globus-rls-cli query rli lfn logicalName rls://myrli.isi.edu

Querying mappings using C API

globus_module_activate(GLOBUS_RLS_CLIENT_MODULE)

globus_rls_client_connect (serverURL, serverHandle)

globus_rls_client_lrc_get_pfn (serverHandle, logicalName, offset, resultLimit, resultList)

globus_rls_client_lrc_get_lfn (serverHandle, targetName1, offset, resultLimit, resultList)

globus_rls_client_rli_get_lrc (serverHandle, logicalName, offset, resultLimit, resultList)

globus_rls_client_close (serverHandle)

the globus toolkit™
www.globustoolkit.org

Querying mappings using Java API

```
RLSClient rls = new RLSClient(URLofServer);
RLSClient.LRC lrc = rls.getLRC();
RLSClient.RLI rli = rls.getRLI();
ArrayList list = lrc.getPFN(logicalName);
list = lrc.getLFN(targetName2);
list = rli.getLRC(logicalName);
rls.Close();
```

- By default, offset and limit are 0 but can be set and passed to query functions

Status of RLS and Future Work

- Continued development of RLS
  - Code available as source and binary bundles at:
    www.globus.org/rls
- RLS is part of the GT3.0 (as a GT2 service)
- RLS will become an OGSI-compliant grid service
  - Replica location grid service specification will be standardized through Global Grid Forum
  - First step may be wrapping the current GT2 services in a GT3 wrapper
  - Significant changes related to treatment of data entities as first-class OGSI-compliant services

the globus toolkit™
www.globustoolkit.org

Higher-Level OGSA Replication Services

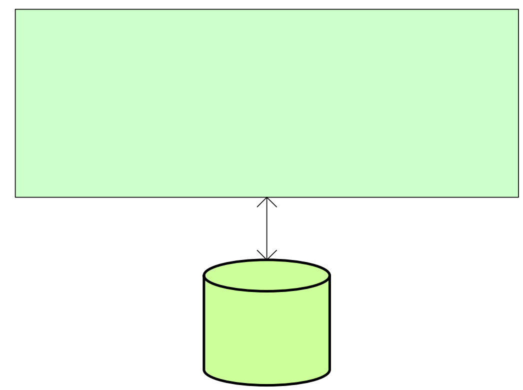
- Registration and Copy Service
  - Calls RFT to perform reliable file transfer
  - Calls RLS to register newly created replicas
  - Atomic operations; roll back to previous consistent state if part of operation fails
- General replication services with various consistency levels/guarantees
  - Subscription-based model
  - Updates of data items must be propagated to all replicas according to update policies
- Plan is also to standardize these through GGF OGSA Data Replication Services Working Group

Outline

- Data architecture: layered, composable services
- Data transfer and access
  - **GridFTP:** Provides high-performance, reliable data transfer for modern WANs
  - **RFT:** Reliable File Transfer Service
- Data replication
  - **RLS:** Replica Location Service
  - Higher-level replication services
- OGSA Database Access and Integration Service
- Metadata Catalog Service

OGSA Data Access and Integration Service (OGSA DAI)

- OGSI-Compliant grid service for access to existing databases
  - GSI security, lifetime management, service data elements, etc.
- Provides both relational and native XML database back ends (mySQL, Xindice, DB2 in progress)
- Provides a general pass-through SQL query interface
- Being standardized through Global Grid Forum
- Reference implementation by UK researchers, IBM



Metadata Services

- Metadata is information that describes data sets
- Metadata Services
  - Store metadata attributes according to a specified schema
  - Answer queries for discovery of data with desired attributes
- Distinguish between *logical* metadata and *physical* metadata
- Metadata Catalog Service
  - Stores logical metadata that describes contents of files and collections
  - Logical metadata is independent of a particular physical instance, applies to all replicas
  - Variables, annotations, some provenance information
- Replica Location Service
  - Stores mappings from logical to physical names

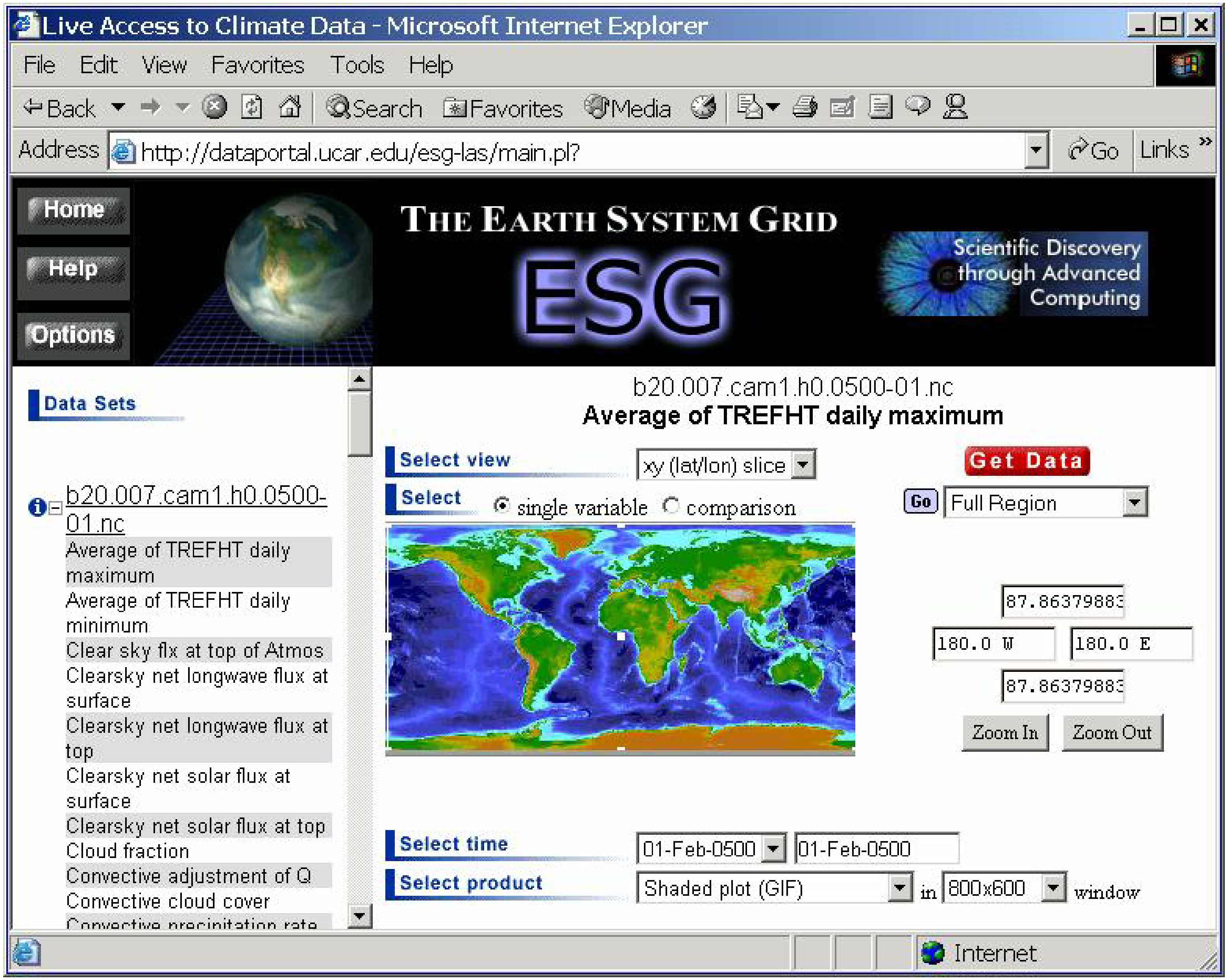
Redesign of MCS

- New implementation will be based on OGSA DAI
- Tools and interfaces customized for metadata management
  - Bulk loading of metadata, standard schemas, standard interfaces
- Extensibility of the metadata service
  - Rich, efficient mechanisms for user-defined attributes
- Distribution and federation of heterogeneous metadata services
  - Exploring relaxed consistency model
  - Heterogeneous metadata services export discovery information to aggregating index nodes

Example of Globus Data Services in Action: The Earth System Grid Project

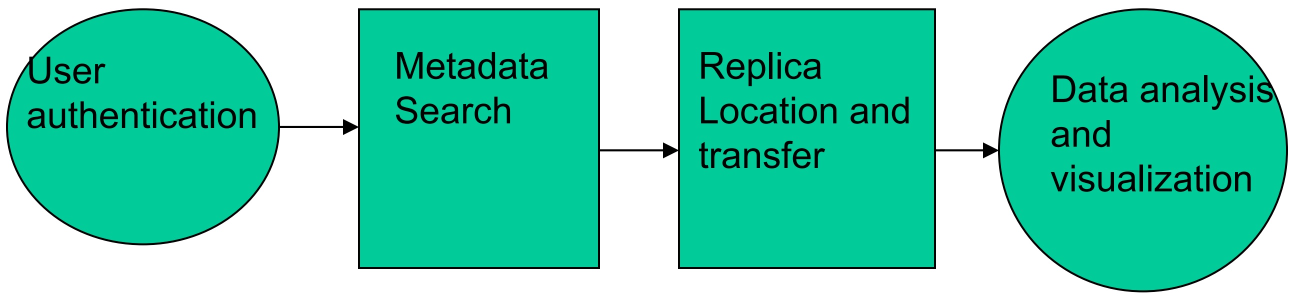
- Addresses challenges associated with enabling the sharing and analysis of, and knowledge development from, global Earth System models
- Through a combination of Grid and emerging community technologies, ESG links distributed federations of supercomputers and large-scale data & analysis servers to provide a seamless and powerful environment that enables the next generation of climate research
- ESG is sponsored by the U.S. DOE Scientific Discovery through Advanced Computing program (SciDAC)

ESG Data Portal



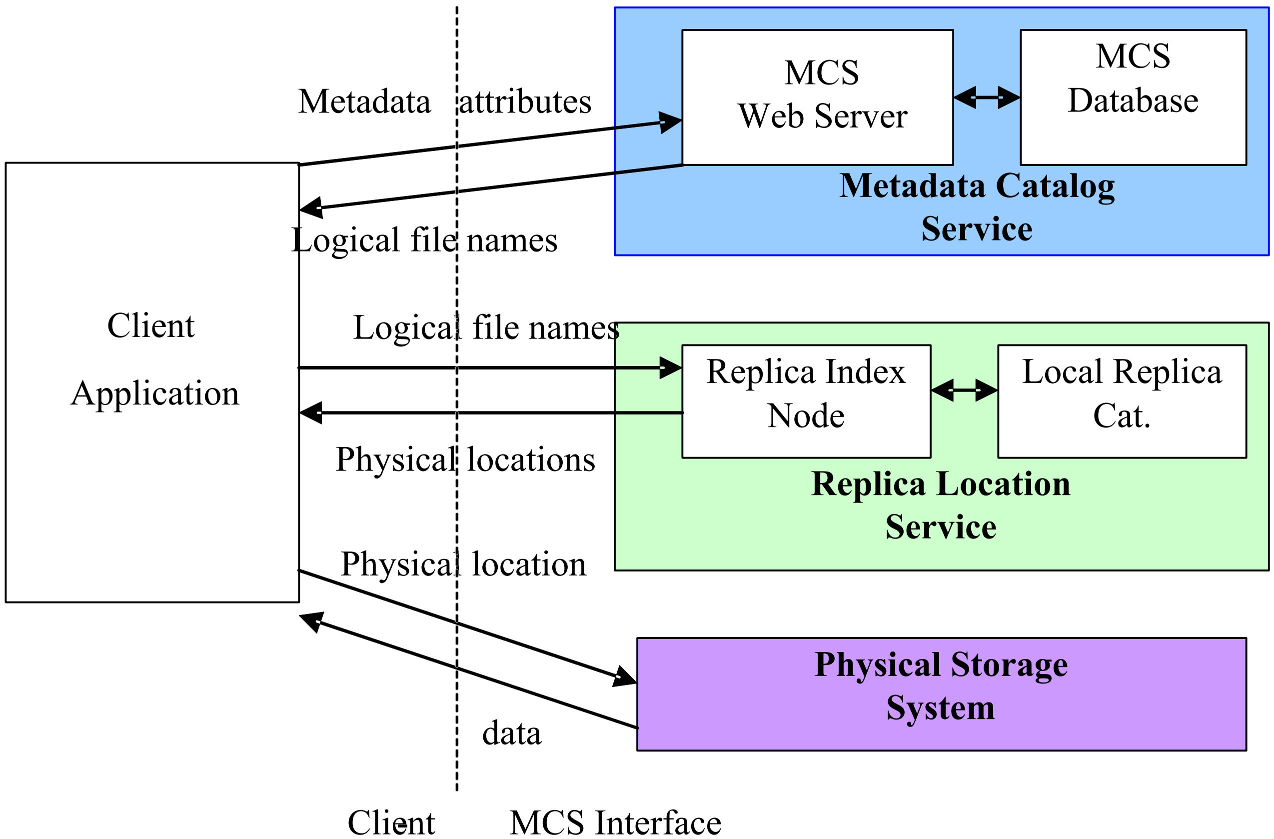
ESG Components

Demonstration Workflow:



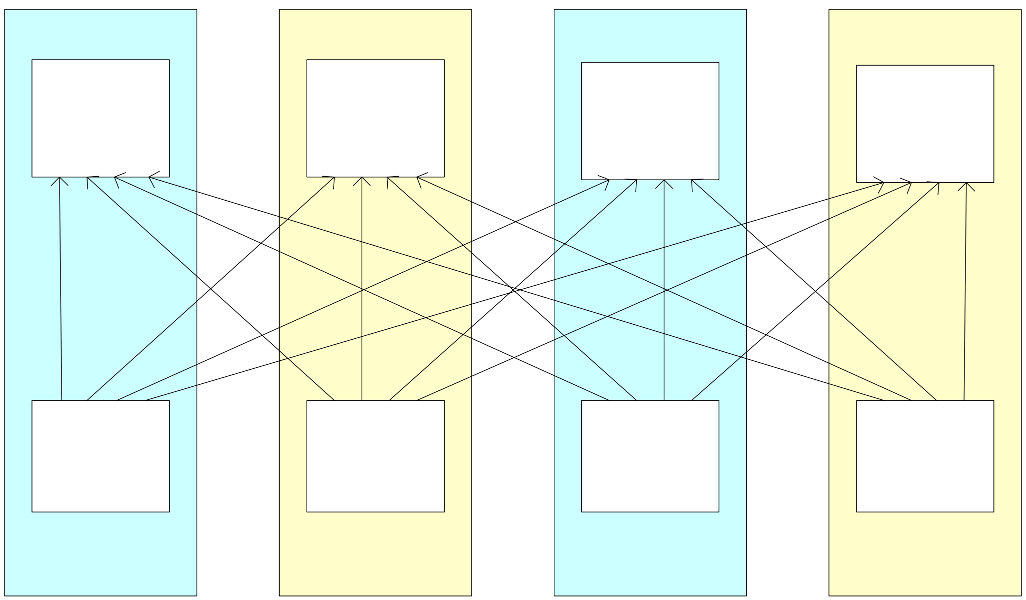
- Globus Toolkit (ANL, ISI)
  - GridFTP data transfer
  - GRAM resource access
  - Community Authorization Service (CAS)
  - Replica Location Service (RLS)
  - Metadata Catalog Service (MCS)
- Web interface (NCAR) and workflow manager
- Hierarchical Resource Mgr. (HRM) (LBNL)
- Metadata (NCAR, LLNL, ISI)
- OpenDAP-G (NCAR, ANL)
- Live Access Server (NCAR)

Use of Metadata Catalogs in Earth System Grid



Replica Location Service Deployment for ESG

- Catalogs at LBNL, NCAR, LLNL, ORNL
- At each location, have deployed a Local Replica Catalog and a Replica Location Index Node
  - Index is replicated everywhere, no single point of failure



Summary: Data Services in GT3

- Presented a layered architecture of data services in GT3
- Composable, orthogonal components
- Some are currently GT2 services: GridFTP, RLS
- Others are OGSI-compliant GT3 services
  - Reliable File Transfer
  - Higher-level replication services
  - OGSA Database Access and Integration Service
  - New version of Metadata Catalog Service
- Combine these services as needed to support higher-level, application-specific data management services