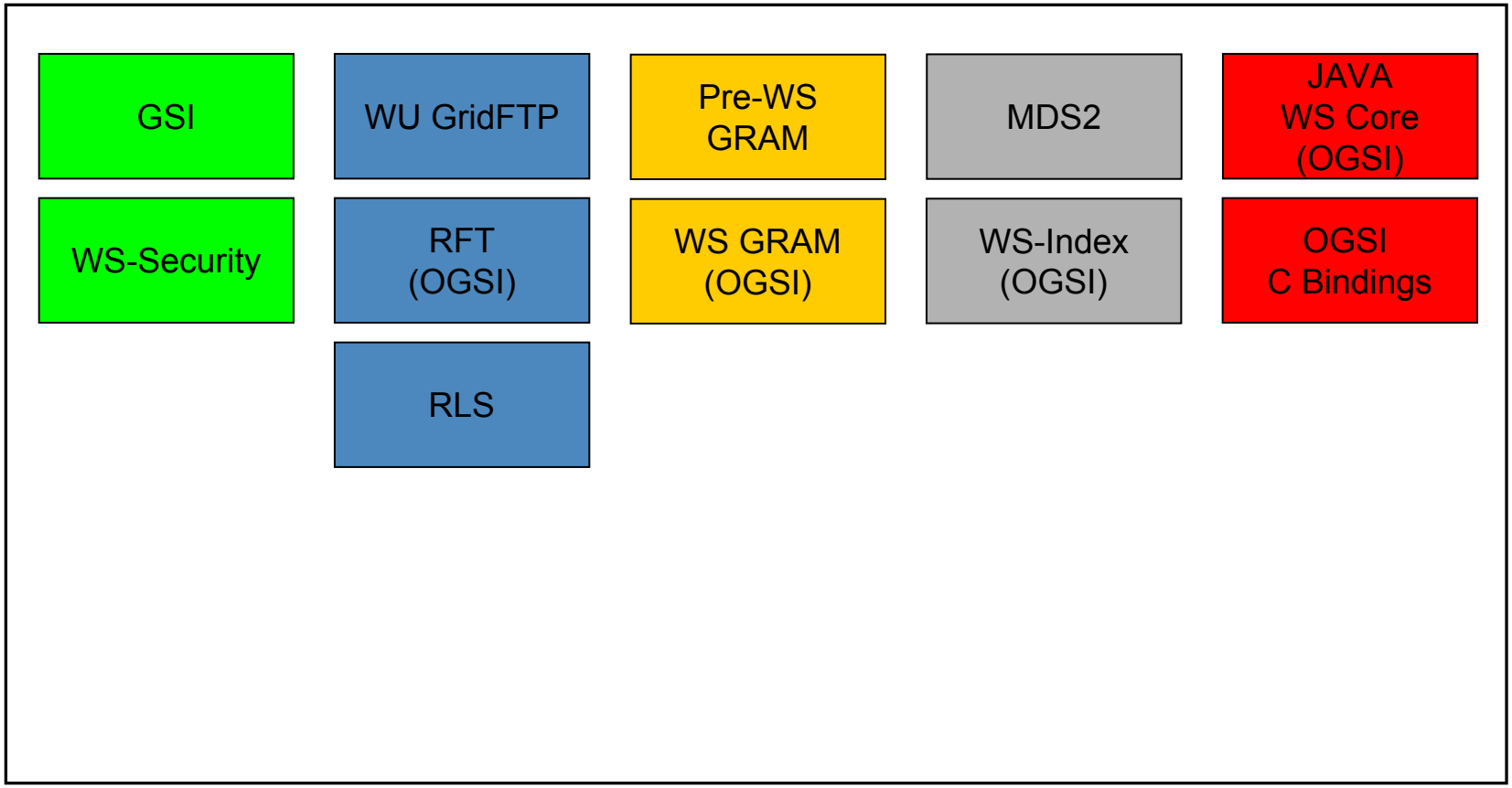
Grid Components

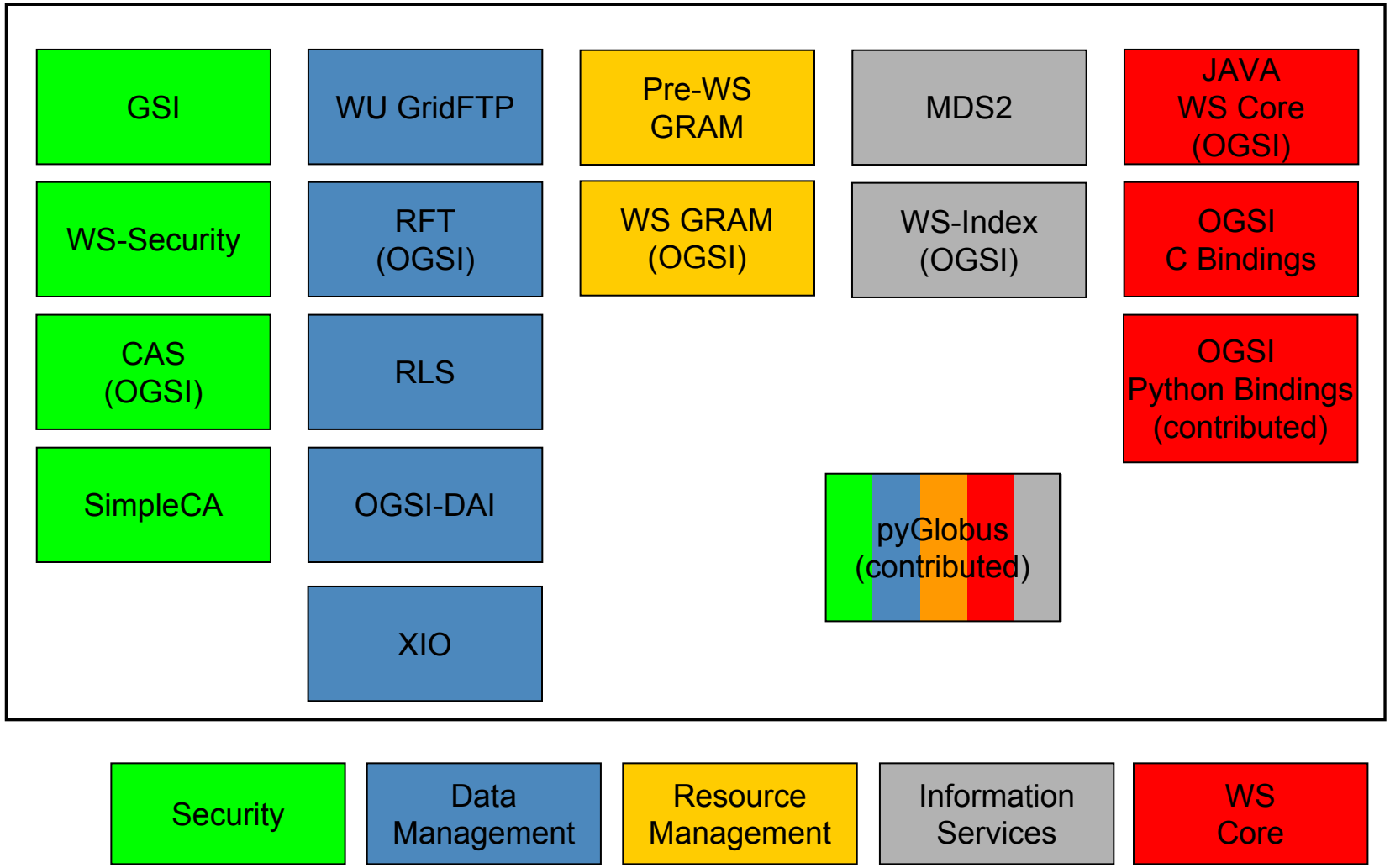
the globus alliance
www.globus.org

Components in Globus Toolkit 3.0



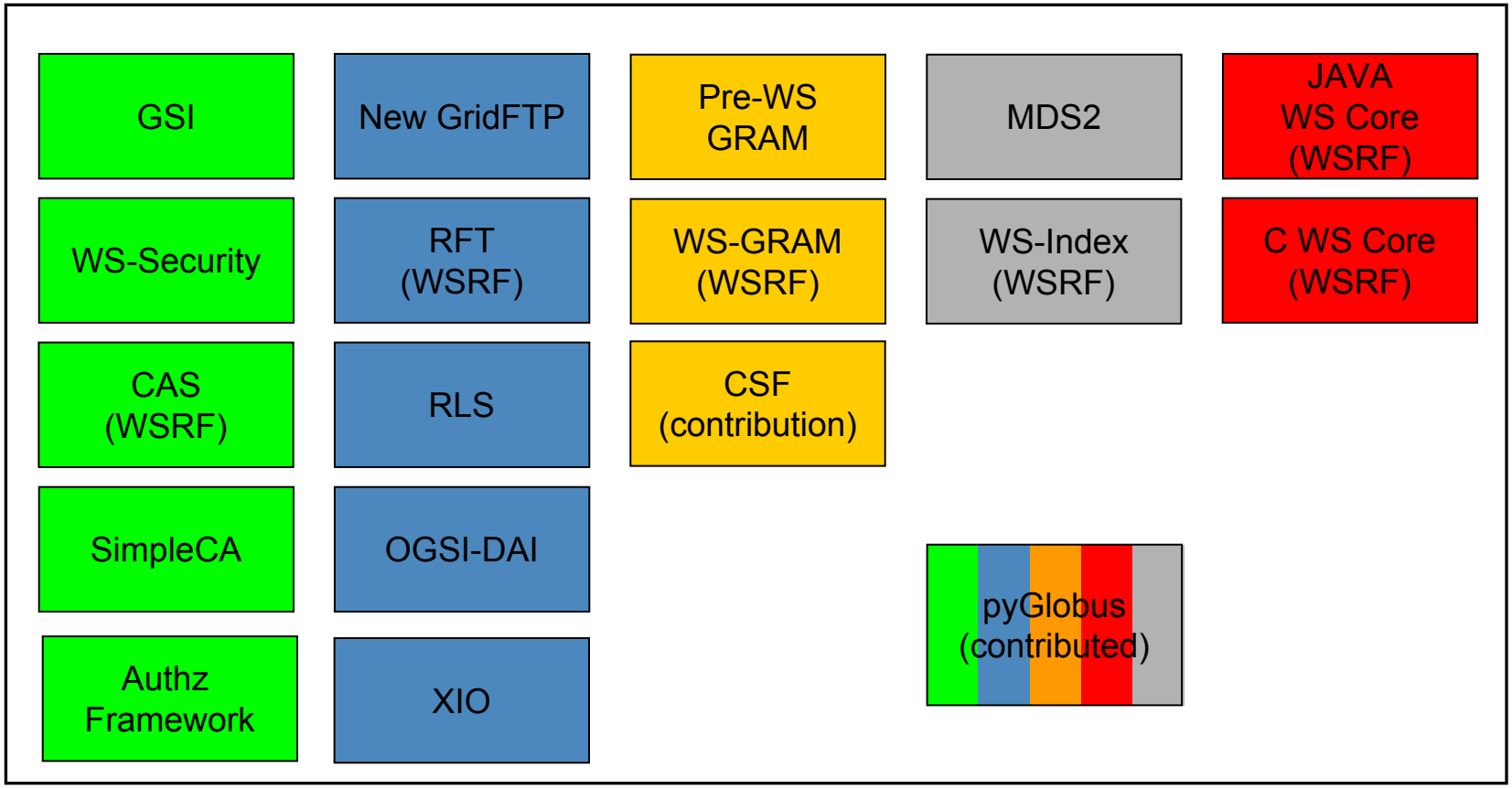
the globus alliance
www.globus.org

Components in Globus Toolkit 3.2



the globus alliance
www.globus.org

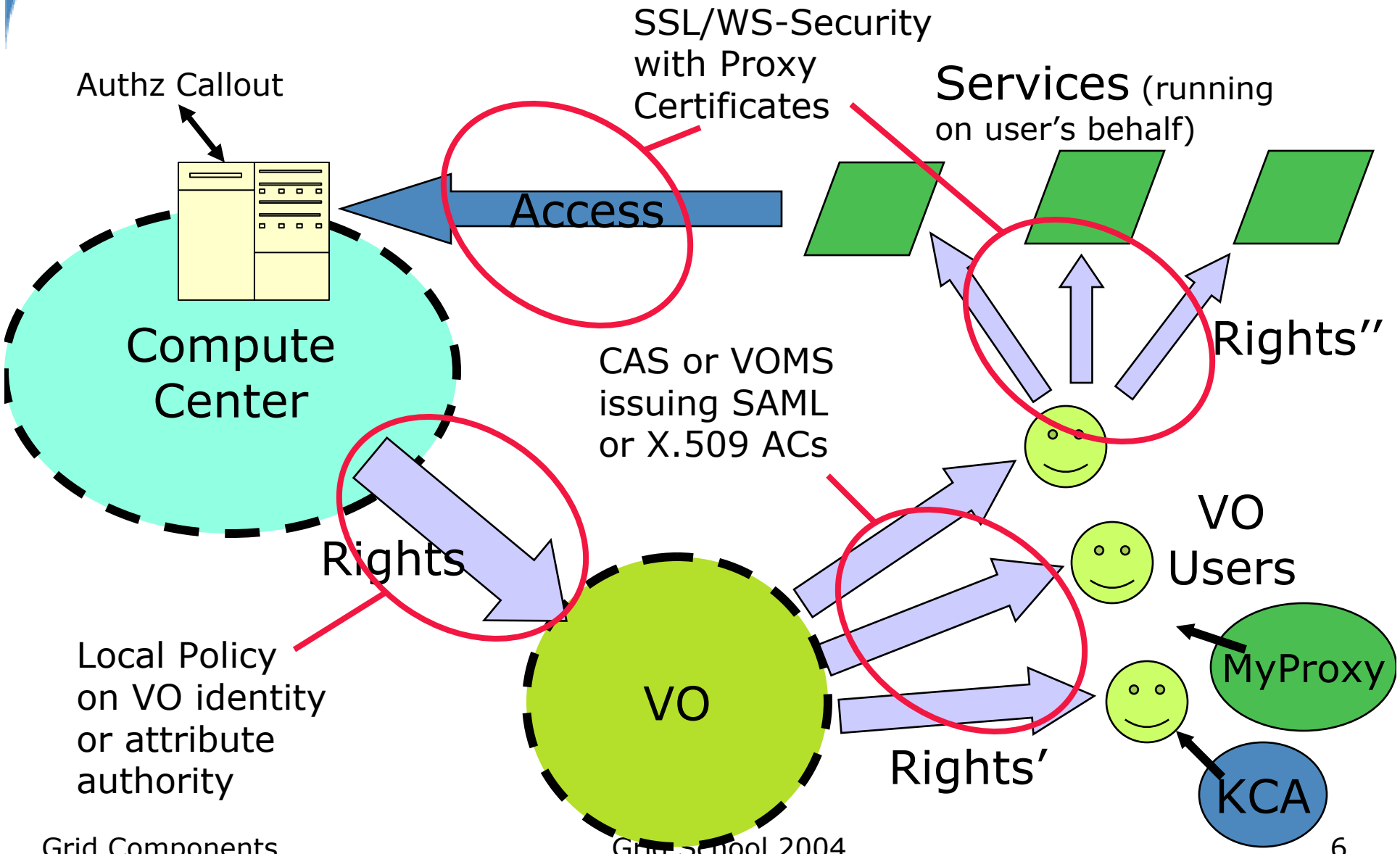
Planned Components in GT 4.0



Component to be covered

- Credential management and authorization
- Job submission and management
- Data discovery and data transfer

GSI Implementation



KX.509 and KCA

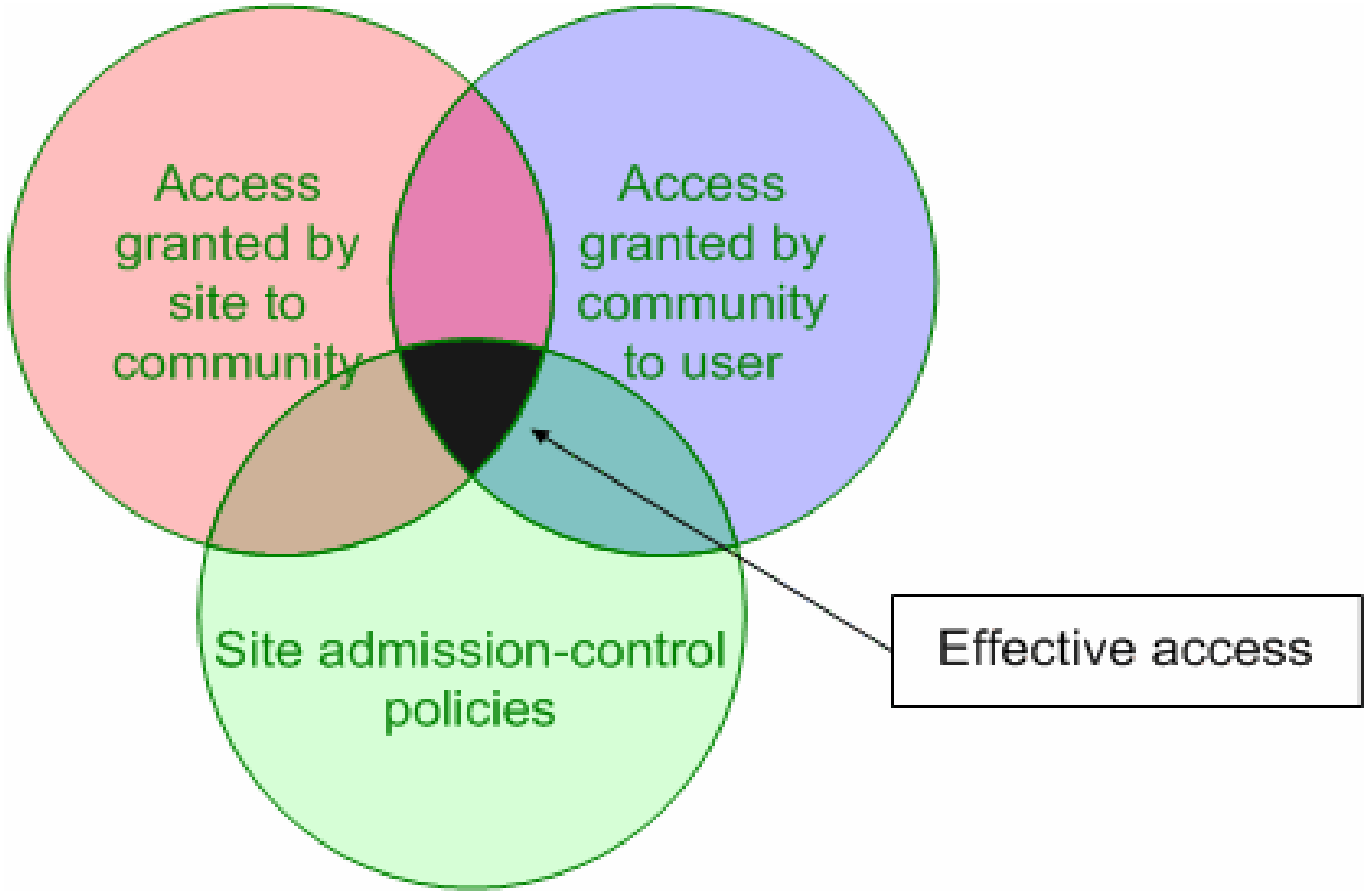
- Institutions that already have a Kerberos realm can use KX.509 and KCA to provide local users with Grid proxy certificates without using a Certificate Authority.
- When users authenticate with Kerberos, they may obtain proxy certificates in addition to their Kerberos tickets.
- KCA is a Kerberized certification service, and KX.509 is a Kerberized client that generates and stores proxy certificates.
- Unlike MyProxy, KX.509 and KCA create credentials for users, so remote sites must be configured to trust the local KCA service's certification authority.

PKINIT

- PKINIT is a service that allows users to use Grid certificates to authenticate to a Kerberos realm.
- For sites that use Kerberized services (like AFS), this allows remote Grid users to obtain the necessary Kerberos tickets to use the site's local facilities properly.
- PKINIT replaces the Kerberos "klog" command and uses the user's Grid certificate to eliminate the need for a Kerberos passphrase.

the globus alliance
www.globus.org

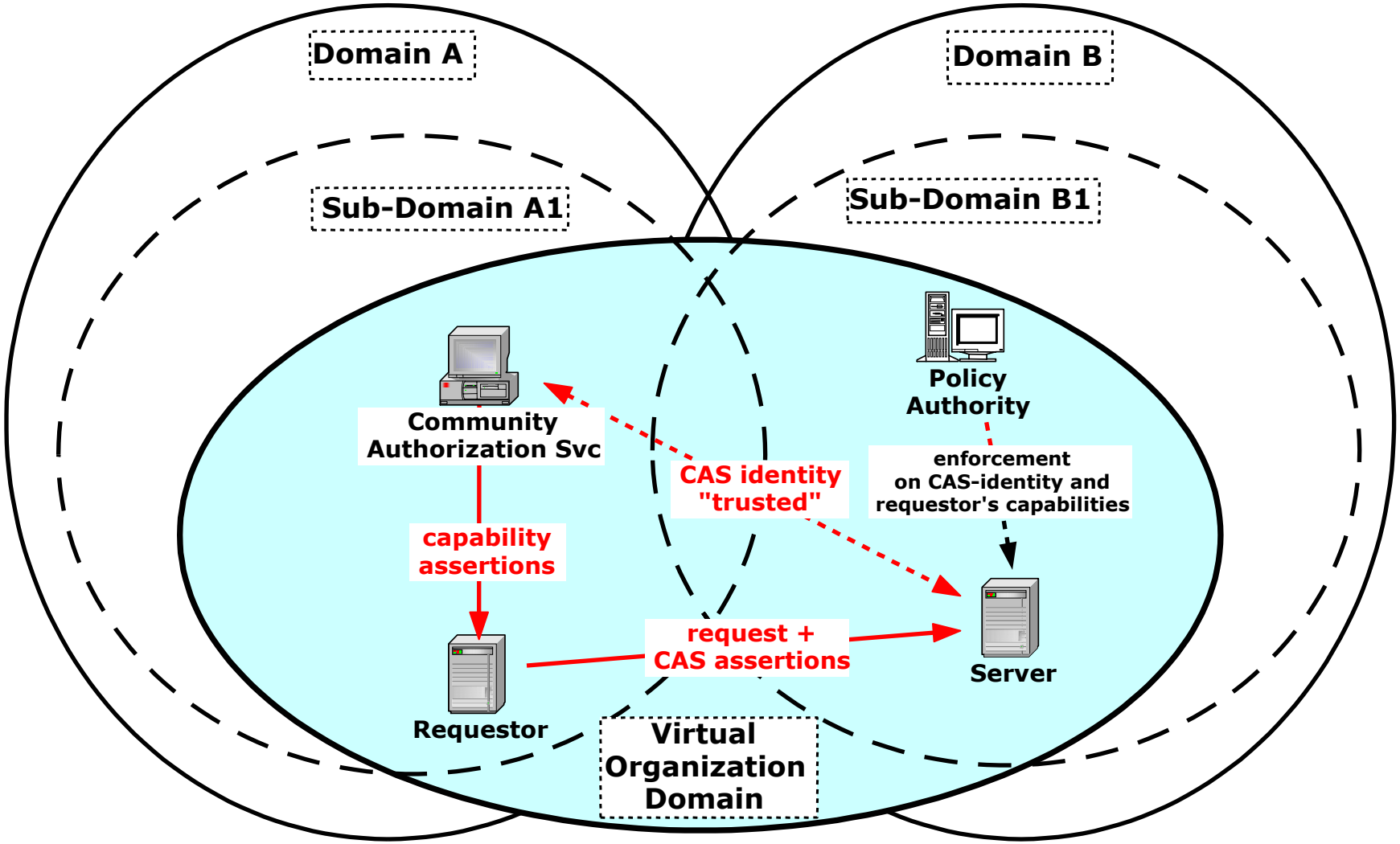
Effective Policy Governing Access Within A Collaboration



Community Authorization Service

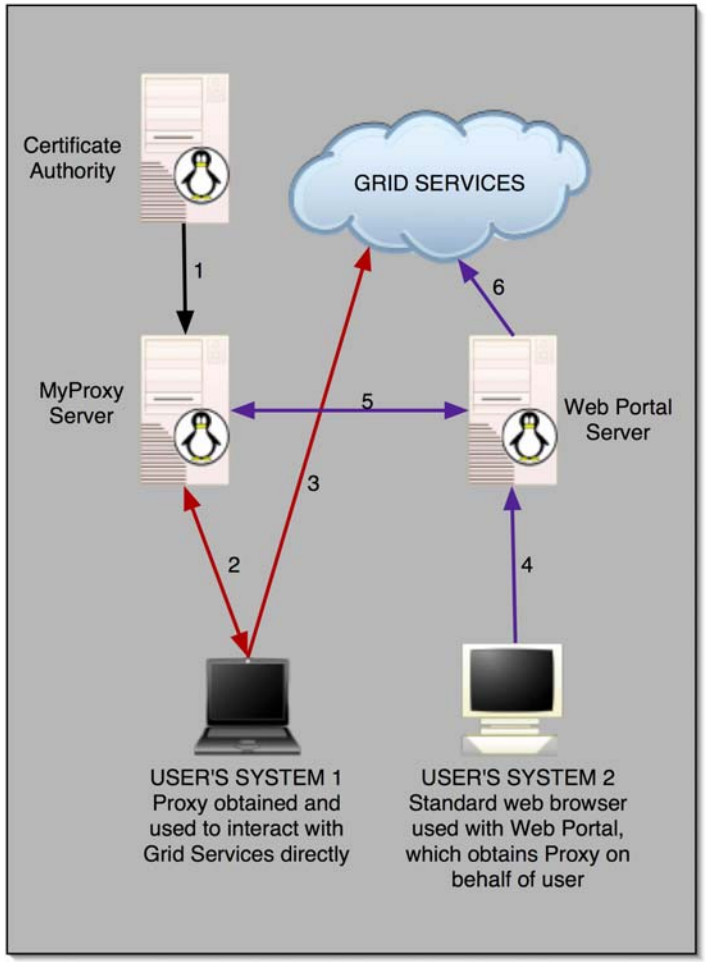
- Question: How does a large community grant its users access to a large set of resources?
- Community Authorization Service (CAS)
  - Outsource policy admin to VO sub-domain
  - Enables fine-grained policy
- Resource owner sets course-grained policy rules for foreign domain on "CAS-identity"
- CAS sets policy rules for its local users
- Requestors obtain capabilities from their local CAS that get enforced at the resource

Community Authorization Service

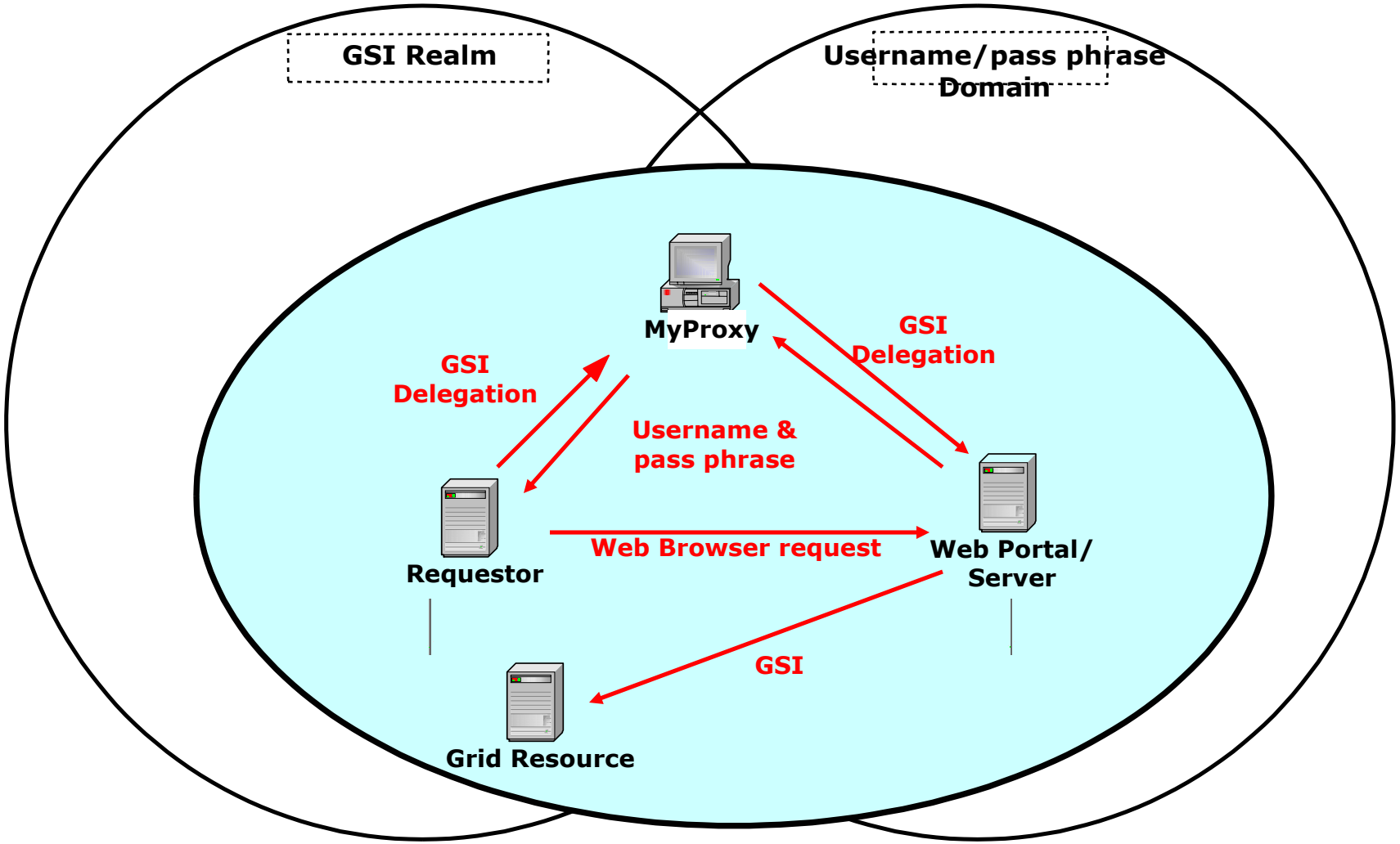


MyProxy

- MyProxy is a remote service that stores user credentials.
  - Users can request proxies for local use on any system on the network.
  - Web Portals can request user proxies for use with back-end Grid services.
- Grid administrators can pre-load credentials in the server for users to retrieve when needed.
- Also handle credential renewal for long-running tasks
- Greatly simplifies certificate management!



MyProxy: Passphrase-X.509 Federation Service



Beyond Local Identity for Authorization

- Mapping to local identity works ok, but has limitations
  - Scalability, granularity, consistency…
- Requirement for greater flexibility
- GT2 has simple API callout to deployment-time libraries/services
- GT3 implement standardized version based on GGF/OASIS work

the globus alliance
www.globus.org

Remove Authz from Applications

- Allow deployment-time selection of supported mechanisms and policies
- OGSA resource virtualization allows for policy on application-independent operation invocation
- Place as much security functionality as possible into sophisticated hosting environments

Resource Management

GRAM Motivation

- Given a job specification, provide a service that can:
  - Create an environment for a job
  - Stage files to/from the environment
  - Submit a job to a local scheduler
  - Monitor a job
  - Send job state change notifications
  - Stream a job's stdout/err during execution

GRAM Overview

- Resource Specification Language (RSL) is used to communicate requirements
- A set of client interfaces enabling programs to be started on remote resources, despite local heterogeneity
- A set of service components for mapping to local scheduling systems
- Two versions:
  - Pre-WS GRAM
  - WS-GRAM

Important Notice!!

- Our goals are:
  - Highly functional interface
    - grid service GWSDLs
    - C API
    - Java API
  - Expressive RSL
  - Only basic command line clients
  - Collaborate with others to create more capable and complete clients
    - E.g. Condor-G grid manager, Platform's CSF

GRAM Features

- Standard protocol for building high-level tools
  - Brokers, metaschedulers, …
- File staging
- At most once submission
- Job status monitoring and control

Resource Specification Language

- Much of the power of GRAM is in the RSL
- Schema defined language for specifying job requests
  - XML based in WS-GRAM
  - LDAP query syntax based in Pre-WS GRAM
  - GRAM translates this common language into scheduler specific language
- GRAM service understands a well defined set of elements
  - executable, arguments, directory, …

RSL-2 Schema

- Use standard XML parsing tools to parse and validate an RSL specification
  - xmlns:http://www.globus.org/namespaces/2003/04/rsl/gram"
  - Functions to process the DOM representation of RSL specification
    - Extracting RSL attributes
    - RSL substitutions
    - Can be used to assist in writing brokers or filters which refine an RSL specification

RSL-2 Example

```
*GNS = "http://www.globus.org/namespaces"
<?xml version="1.0" encoding="UTF-8"?>
<rsl:rsl
    xmlns:rsl="GNS/2003/04/rsl"
    xmlns:gram="GNS/2003/04/rsl/gram"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
            GNS/2003/04/rsl
            ./schema/base/gram/rsl.xsd
            GNS/2003/04/rsl/gram
            ./schema/base/gram/gram_rsl.xsd">
   <gram:job>
    <gram:executable><rsl:path>
       <rsl:stringElement value="/bin/ls"/>
    </rsl:path></gram:executable>
  </gram:job>
</rsl:rsl>
```

RSL Elements For GRAM

- <gram:executable> (type = rsl:pathType)
  - Program to run
  - A file path (absolute or relative) or URL
- <directory> (type = rsl:pathType)
  - Directory in which to run (default is HOME)
- <arguments> (type = rsl:stringArrayType)
  - List of string arguments to program
- <environment> (type = rsl:hashtableType)
  - List of environment variable name/value pairs

RSL Attributes For GRAM

- <stdin> (type = rsl:pathType)
  - Stdin for program
  - A file path (absolute or relative) or URL
  - If remote, entire file is pre-staged before execution
- <stdout> (type = rsl:pathArrayType)
  - stdout for program
  - Multiple file paths (absolute or relative) or URL's
  - If remote, file is incrementally transferred
- <stderr> (type = rsl:pathArrayType)
  - stderr for program
  - Multiple file paths (absolute or relative) or URL's
  - If remote, file is incrementally transferred

the globus alliance
www.globus.org

RSL Attributes For GRAM

- <count> (type = rsl:integerType)
  - Number of processes to run (default is 1)
- <hostCount> (type = rsl:integerType)
  - On SMP multi-computers, number of nodes to distribute the "count" processes across
  - count/hostCount = number of processes per host
- <project> (type = rsl:stringType)
  - Project (account) against which to charge
- <queue> (type = rsl:stringType)
  - Queue into which to submit job
  - Queue properties reflected in the MDS resource description

RSL Attributes For GRAM

- <maxWallTime> (type = rsl:longType)
  - Maximum wall clock runtime in minutes
- <maxCpuTime> (type = rsl:longType)
  - Maximum CPU runtime in minutes
- <maxTime> (type = rsl:longType)
  - Only applies if above are not used
  - Maximum wall clock or cpu runtime (schedulers's choice) in minutes
    - CPU runtime makes sense on a time shared machine
    - Wall clock runtime makes sense on a space shared machine

the globus alliance
www.globus.org

RSL Attributes For GRAM

- <maxMemory> (type = rsl:integerType)
  - Maximum amount of memory for each process in megabytes
- <minMemory> (type = rsl:integerType)
  - Minimum amount of memory for each process in megabytes

RSL Attributes For GRAM

- <jobType> (type = gram:jobRunEnumerationType)
  - Value is one of “mpi”, “single”, “multiple”, or “condor”
    - mpi: Run the program using “mpirun -np <count>”
    - single: Only run a single instance of the program, and let the program start the other count-1 processes/threads
      - Good for scripts, and for multi-threaded programs
    - multiple: default value - Start <count> instances of the program using the appropriate scheduler mechanism
    - condor: Start a <count> Condor processes running in “standard universe” (I.e. linked with Condor libraries for remote I/O, checkpoint/restart, etc.)

RSL Attributes for GRAM

- <scratchDir> (type = rsl:pathType)
  - A unique subdir under <path> is created for job
  - If path is relative, it is relative to:
    - First - A site configured scratch directory
    - Second – Users HOME directory on JM host
  - The job may use SCRATCH_DIRECTORY in RSL substitutions
- <gassCache> (type = rsl:pathType)
  - Overrides the default GASS cache directory
  - Default is site configurable, or ~/.globus/.gasscache if not configured
- <libraryPath> (type = rsl:pathArrayType)
  - Set job environment so apps built to use shared libraries will run properly

the globus alliance
www.globus.org

RSL Attributes for GRAM

- <fileStageIn> (type = rsl:fileInputArrayType)
  - List of remote url to local file pairs to be staged to host where job will run
- <fileStageInShared> (type=rsl:fileInputArrayType)
  - List files to be staged to the GASS cache
  - Links from cache to local file will be made
- <fileStageOut> (type = rsl:fileOutputArrayType)
  - List files to be staged out after job completes
- <fileCleanUp> (type = rsl:pathArrayType)
  - List files to be removed after job completes

the globus alliance
www.globus.org

RSL Substitutions

- RSL supports variable substitutions
  - Definition example
    - <rsl:substitutionDef name="MY HOME">
      <rsl:stringElement value="/home/user1"/>
      </rsl:substitutionDef>
  - Reference example
    - <gram:executable>
      <rsl:substitutionRef name="MY HOME"/>
      <rsl:stringElement path="/a.out"/>
      </gram:executable>
- Allows for late binding of values
  - Can refer to something that is not yet defined

GRAM Defined RSL Substitutions

- GRAM defines a set of RSL substitutions before processing the job request
  - Client submitted RSL can assume these substitutions are defined and refer to them
- Allows for generic RSL expressions to adapt to site and resource configurations
  - Goal: Clients should not have to do manual configuration of resources before they submit jobs to them
  - GRAM defined RSL substitutions define minimal information necessary to bootstrap

the globus alliance
www.globus.org

GRAM Defined RSL Substitutions

- Machine Information
  - GLOBUS_HOST_MANUFACTURER
  - GLOBUS_HOST_CPUTYPE
  - GLOBUS_HOST_OSNAME
  - GLOBUS_HOST_OSVERSION

GRAM Defined RSL Substitutions

- Path to Globus installation
  - GLOBUS_LOCATION
- Miscellaneous
  - HOME
  - LOGNAME
  - GLOBUS_ID
  - SCRATCH_DIRECTORY

GRAM RSL Examples

*GNS = "http://www.globus.org/namespaces"

```
<!--- GRAM RSL Namespace --->
<?xml version="1.0" encoding="UTF-8"?>
<rsl:rsl
    xmlns:rsl="GNS/2003/04/rsl"
    xmlns:gram="GNS/2003/04/rsl/gram"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="
            GNS/2003/04/rsl
            ./schema/base/gram/rsl.xsd
            GNS/2003/04/rsl/gram
            ./schema/base/gram/gram_rsl.xsd">
```

GRAM RSL Examples

```
<rsl: rsl <!--- insert GRAM RSL Namespace --->
  <gram:job>
    <gram:executable><rsl:path>
      <rsl:stringElement value="/bin/ls"/>
    </rsl:path></gram:executable>
    <gram:directory><rsl:stringElement value="/tmp"/>
    </gram:directory>
    <gram:arguments><rsl:stringArray>
      <rsl:string><rsl:stringElement value="-l"></rsl:string>
      <rsl:string><rsl:stringElement value="-a"></rsl:string>
    </rsl:stringArray></gram:arguments>
  </gram:job>
</rsl:rsl>
```

GRAM RSL Examples

```
<rsl: rsl <!--- insert GRAM RSL Namespace --->
    <rsl:substitutionDef name="EXE">
      <rsl:stringElement value="my_exe"/>
    </rsl:substitutionDef>
    <gram:job>
      <gram:executable><rsl:path>
        <rsl:substitutionRef name="HOME"/>
        <rsl:substitutionRef name="EXE"/>
      </rsl:path></gram:executable>
    </gram:job>
</rsl:rsl>
```

GT3 GRAM Client Interfaces

- Java & C client stubs for GT3 Services
- C client library for Pre-OGSI GRAM
- Java & C Pre-OGSI GRAM client API for OGSI GRAM services
  - APIs use the stubs mentioned above
  - GT2 API compatibility for GT3 services
  - Ease transition from GT2 to GT3
  - managed-job-globusrun uses the Java API
- Java & C GT2-3 RSL Translator API
  - Accepts a GT2 RSL and translates to GT3 RSL (XML)
- PyGlobus (Keith Jackson, krjackson@lbl.gov)
  - GT2 and GT3 GRAM Python bindings

the globus alliance
www.globus.org

Gram Clients

- globusrun command line
  - Useful for ***simple*** scripting and testing
- Functions:
  - Submit RSL string to specified host
  - Create GASS server for staging of local files
  - List jobs
  - Manage jobs

the globus alliance
www.globus.org

pyGlobus GRAM Interface

- cancel_job(self, jobContact)
  - Cancels a job.
- check_status(self, jobContact)
  - Checks the status of a current job.
- refresh_credentials(self, jobContact, cred)
  - Refresh credentials associated with a job.
- submit_request(self, resourceManager, description, jobStateMask, callbackContact)
  - Submits a job request to a resource manger.
- Asynchronous versions also available

gramClient Example

```
from pyGlobus.gramClient import *
from threading import *
cond = 0
rm = "host.lbl.gov"
rsl = "&(executable=/bin/date)"

def done(cv,contact,state,err):
    global cond
    if state == JOB_STATE_FAILED:
        print "Job failed"
    elif state == JOB_STATE_DONE:
        print "Job is done"
    else:
        print "ERROR: ", err
    cv.acquire()
    cond = 1
    cv.notify()
    cv.release()
```

```
def main(rm, rsl):
    condV = Condition(Lock())
    try:
        gC = GramClient()
        cbContact =
           gC.set_callback(done, condV)
        jobContact =
           gC.submit_request(rm, rsl,
              JOB_STATE_ALL, cbContact)
        while cond == 0:
            condV.acquire()
            condV.wait()
            condV.release()
       gC.remove_callback(cbContact)
    except GramClientException, ex:
        print ex
```

C Job Submission Example

```
callback_func(void *user_arg, char *job_contact,
              int state, int errorcode)
{
    globus_i_globusrun_gram_monitor_t *monitor;
    monitor = (globus_i_globusrun_gram_monitor_t *) user_arg;
    globus_mutex_lock(&monitor->mutex);
    monitor->job_state = state;
    switch(state)
    {
    case GLOBUS_GRAM_PROTOCOL_JOB_STATE_PENDING:
{
    globus_i_globusrun_gram_monitor_t *monitor;
    monitor = (globus_i_globusrun_gram_monitor_t *) user_arg;
    globus_mutex_lock(&monitor->mutex);
    monitor->job_state = state;
    switch(state)
    {
    case GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED:
        if(monitor->verbose)
        {
globus_libc_printf("GLOBUS_GRAM_PROTOCOL_JOB_STATE_FAILED\n");
        }
        monitor->done = GLOBUS_TRUE;
        break;
    case GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE:
        if(monitor->verbose)
        {
globus_libc_printf("GLOBUS_GRAM_PROTOCOL_JOB_STATE_DONE\n");
        }
        monitor->done = GLOBUS_TRUE;
        break;
 }
  globus_cond_signal(&monitor->cond);
  globus_mutex_unlock(&monitor->mutex);
}
```

```
globus_l_globusrun_gramrun(char * request_string,
                           unsigned long options,
                           char *rm_contact)
{
    char *callback_contact = GLOBUS_NULL;
    char *job_contact = GLOBUS_NULL;
    globus_i_globusrun_gram_monitor_t monitor;
    int err;
    monitor.done = GLOBUS_FALSE;
    monitor.verbose=verbose;
    globus_mutex_init(&monitor.mutex, GLOBUS_NULL);
    globus_cond_init(&monitor.cond, GLOBUS_NULL);

    err = globus_module_activate(GLOBUS_GRAM_CLIENT_MODULE);
    if(err != GLOBUS_SUCCESS)
    { … }
    err = globus_gram_client_callback_allow(
            globus_l_globusrun_gram_callback_func,
            (void *) &monitor,
            &callback_contact);
        if(err != GLOBUS_SUCCESS)
        { … }
  err = globus_gram_client_job_request(rm_contact,
            request_string, GLOBUS_GRAM_PROTOCOL_JOB_STATE_ALL,
             callback_contact, &job_contact);
  if(err != GLOBUS_SUCCESS)
        { … }
 globus_mutex_lock(&monitor.mutex);
   while(!monitor.done)  {
      globus_cond_wait(&monitor.cond, &monitor.mutex);
   }
 globus_mutex_unlock(&monitor.mutex);
   globus_gram_client_callback_disallow(callback_contact);
   globus_free(callback_contact);
```

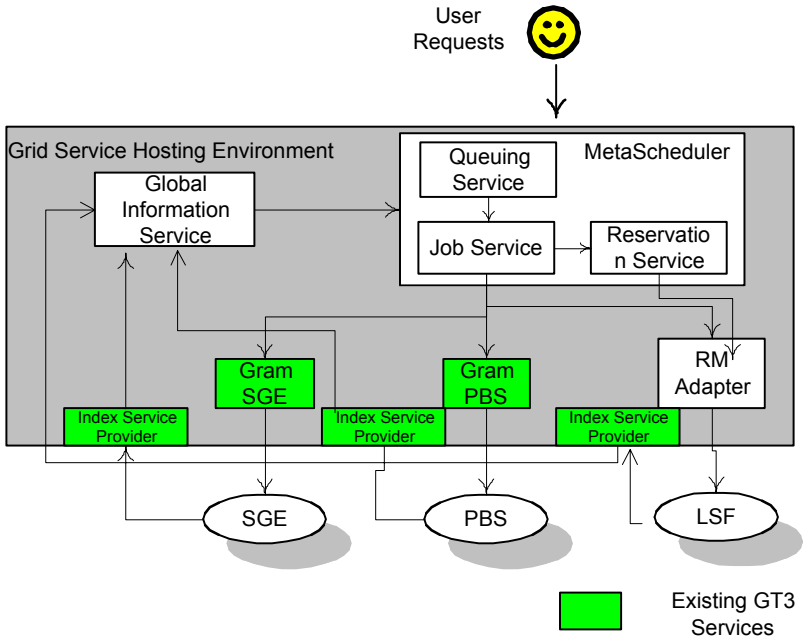
Condor, Condor-G, DAGman

- Condor addresses many workflow challenges for Grid applications.
  - Managing sets of subtasks
  - Getting the tasks done reliably and efficiently
  - Managing computational resources
- Similar to a distributed batch processing system, but with some interesting twists.
  - Scheduling policy
  - ClassAds
  - DAGman
  - Checkpointing and Migration
  - Grid-aware & Grid-enabled
  - Flocking (linking pools of resources) & Glide-ins

Condor
High Throughput Computing

Platform CSF

- An open source implementation of OGSA-based metascheduler for VOs.
  - Supports emerging WS-Agreement spec
  - Supports GT GRAM
  - Uses GT Index Service
- Fills in gaps in existing resource management picture
  - Integrated with Platform LSF and Platform Multicluster
  - Anticipated for inclusion in GT 4.0 release



GRAM in GT3 Releases

- Two versions of resource management services
  - OGSI compliant
    - MMJFS, MJFS
  - Pre-OGSI
    - Gatekeeper, jobmanager

the globus alliance
www.globus.org

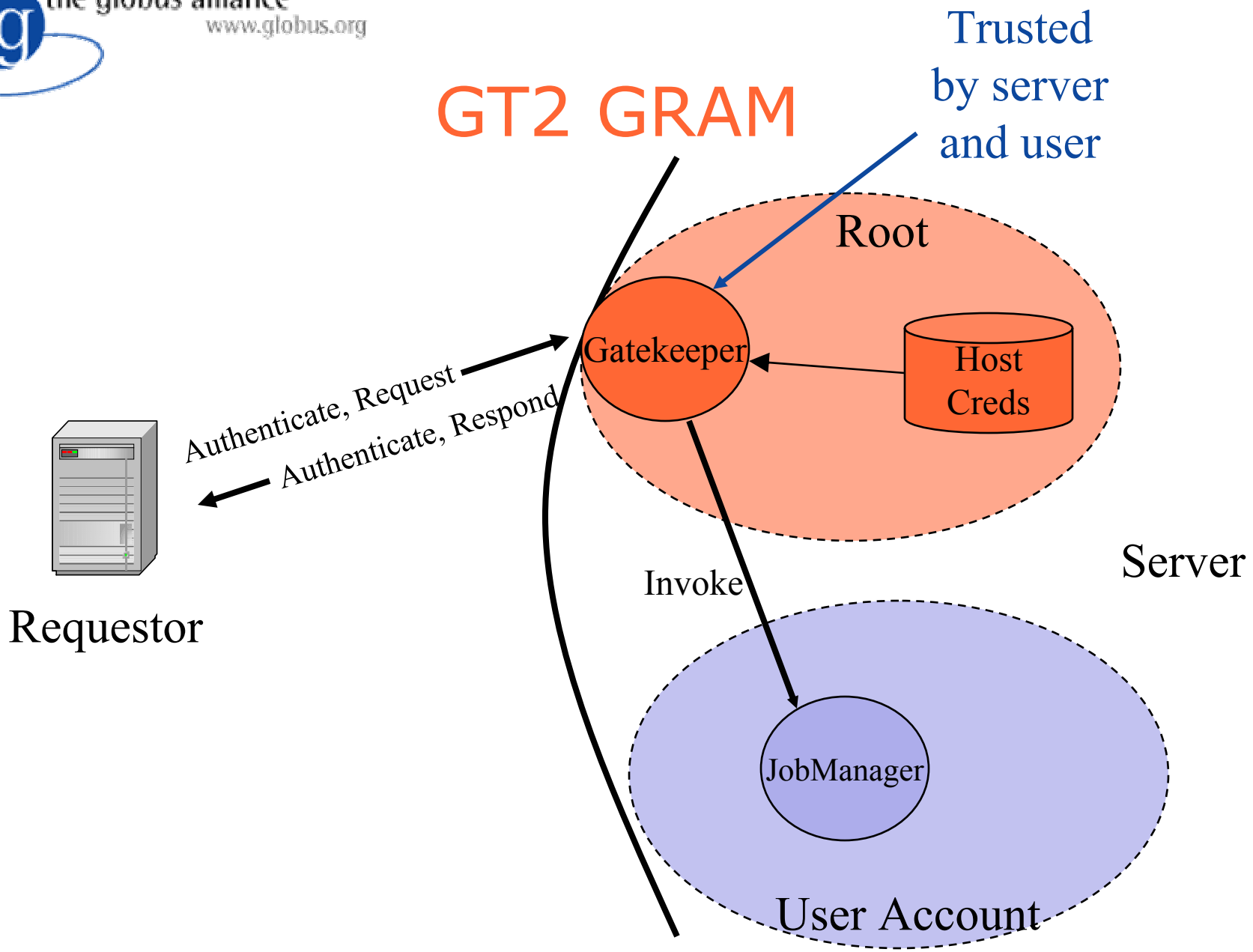
OGSI Compliant GRAM

- A set of OGSI compliant services that provide remote job execution
  - (Master) Managed Job Factory Service (MJFS)
  - Managed Job Service (MJS)
  - File Stream Factory Service (FSFS)
  - File Stream Service (FSS)
- Resource Specification Language (RSL-2) schema is used to communicate job requirements
- Remote jobs run under local users account
- Client to service credential delegation is done user to user, *not* through a third party

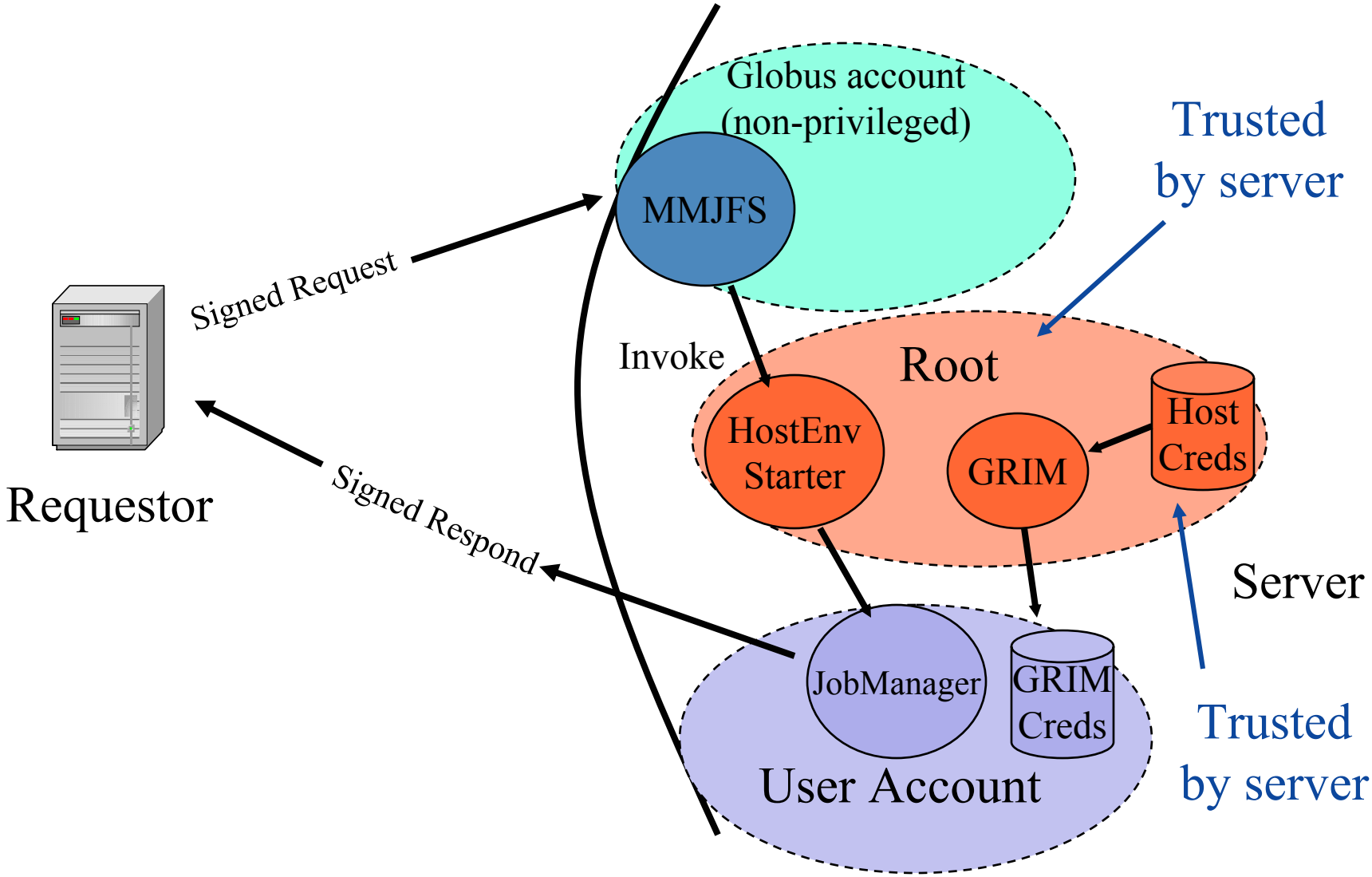
Pre-OGSI GRAM

- A set of non-OGSI compliant services that provide remote job execution
  - Gatekeeper
  - Jobmanager
- Resource Specification Language (RSL) is used to communicate job requirements
- Remote jobs run under local users account
- *Client to service credential delegation is done through a third party (gatekeeper)

GT2 GRAM


Trusted
by server
and user
Root
Gatekeeper
Host
Creds
Authenticate, Request
Authenticate, Respond
Requestor
Server
Invoke
JobManager
User Account

GT3 GRAM

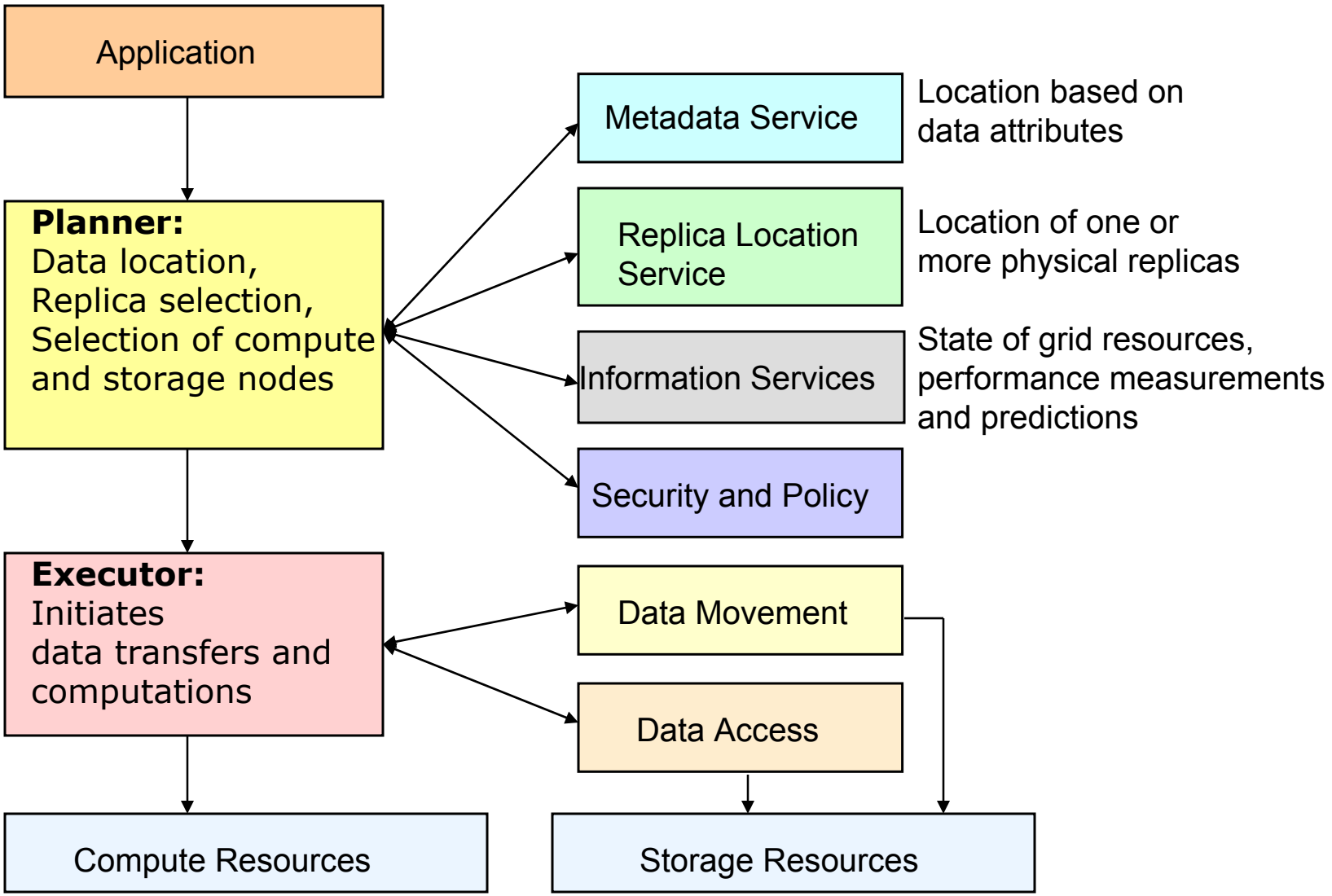

Globus account
(non-privileged)
MMJFS
Trusted
by server
Signed Request
Invoke
Root
HostEnv
Starter
GRIM
Host
Creds
Requestor
Signed Respond
Server
JobManager
GRIM
Creds
User Account
Trusted
by server

GRAM Documentation

- http://www.globus.org/gram

Data Services

Functional View of Grid Data Management



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)

the globus alliance
www.globus.org

GridFTP

- Data-intensive grid applications transfer and replicate large data sets (terabytes, petabytes)
- GridFTP Features:
  - Third party (client mediated) transfer
  - Parallel transfers
  - Striped transfers
  - TCP buffer optimizations
  - Grid security
- Important feature is separation of control and data channel

What is GridFTP?

- A secure, robust, fast, efficient, standards based, widely accepted data transfer protocol
- A Protocol
  - Multiple Independent implementation can interoperate
    - This works. Both the Condor Project at Uwis and Fermi Lab have home grown servers that work with ours.
    - Lots of people have developed clients independent of the Globus Project.
- We also supply a reference implementation:
  - Server
  - Client tools (globus-url-copy)
  - Development Libraries

the globus alliance
www.globus.org

GridFTP: Secure

- Uses GSS security API
- "Normal" Globus uses GSI (X.509 certs / Public Key)
- GSS supports Kerberos bindings
- However, Kerberos and GSS have not kept up with GSI features
- This means certain features don't work (Data Channel Authentication)

the globus alliance
www.globus.org

GridFTP: Robust and Reliable

- Our extensions provide for "Restart Markers" that list byte ranges written to disk
- If any remote resource fails, the restart markers can be used to pick up the transfer, including "holey" transfers
- There is a default restart "plug-in" provided, but this can be modified to provide customized restart policy.

the globus alliance
www.globus.org

GridFTP: Fast and Efficient

- TCP Buffer size control
  - Current implementation is manual only
  - Wu Feng (LANL) has a prototype of DRS working
- Parallelism (multiple sockets between two endpoints)
  - "works around" TCP limitations
  - Can get 90%+ link utilization
- Striping (multiple network endpoints, I.e. clusters)
  - Mutiple levels of parallelism (CPU, disk, NIC, etc)
- Recent tests at LANL
  - Disk transfers limited by disk speed
  - Memory transfers achieved 100 MBs, per link

the globus alliance
www.globus.org

GridFTP: Standards Based

- FTP protocol is defined by several IETF RFCs
- Start with most commonly used subset
  - Standard FTP: get/put etc., 3rd-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

GridFTP: Standards Based (cont)

- Existing standards
  - RFC 959: File Transfer Protocol
  - RFC 2228: FTP Security Extensions
  - RFC 2389: Feature Negotiation for the File Transfer Protocol
  - Draft: FTP Extensions
- New drafts
  - GridFTP: Protocol Extensions to FTP for the Grid
    - Grid Forum GridFTP Working Group
    - Submitted for public comment

the globus alliance
www.globus.org

GridFTP: Widely Accepted

- GridFTP is the de facto standard for transport in Grid Computing
- A significant fraction of Grid Projects, both in the US and abroad are using GridFTP
  - ESG, PPDG, EUDG, PPARC, DOE SG, LCG, NorduGrid, NEES, GriPhyN, SDSS, NVO
- Our requirements are gathered from a wide range of communities and applications.

the globus alliance
www.globus.org

globus-url-copy

- Copy source URL to destination URL
  - http, https, FTP, gsiftp, and file URLs supported
  - 3rd party transfer
  - Options for restart, window size, parallelism, etc.

```
globus-url-copy
    gsiftp://sourceHostName:port/dir1/dir2/file17
    gsiftp://destHostName:port/dirX/dirY/fileA
```

Demonstration: globus-url-copy Command Line Tool

globus-url-copy [options] sourceURL destURL

OPTIONS

-b | -binary

Do not apply any conversion to the files. *default*

-tcp-bs <size> | -tcp-buffer-size <size>

specify the size (in bytes) of the buffer to be used by the underlying ftp data channels

-bs <block size> | -block-size <block size>

specify the size (in bytes) of the buffer to be used by the underlying transfer methods

the globus alliance
www.globus.org

Globus-url-copy (cont.)

-p <parallelism> | -parallel <parallelism>

specify the number of streams to be used in the ftp transfer

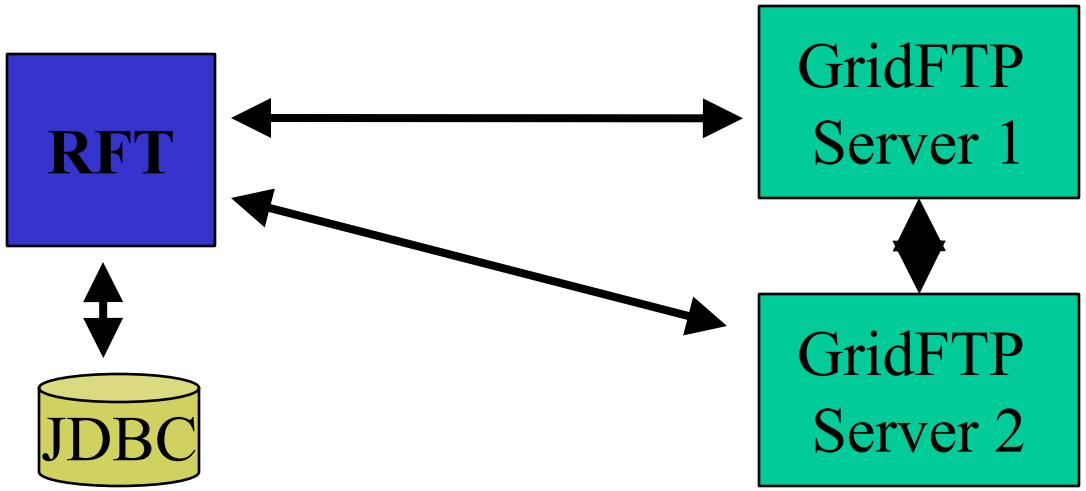
-notpt | -no-third-party-transfers

turn third-party transfers off (on by default)

GridFTP Command Line Tool

Reliable File Transfer Service

- Reliably performs a third party transfer between two GridFTP servers

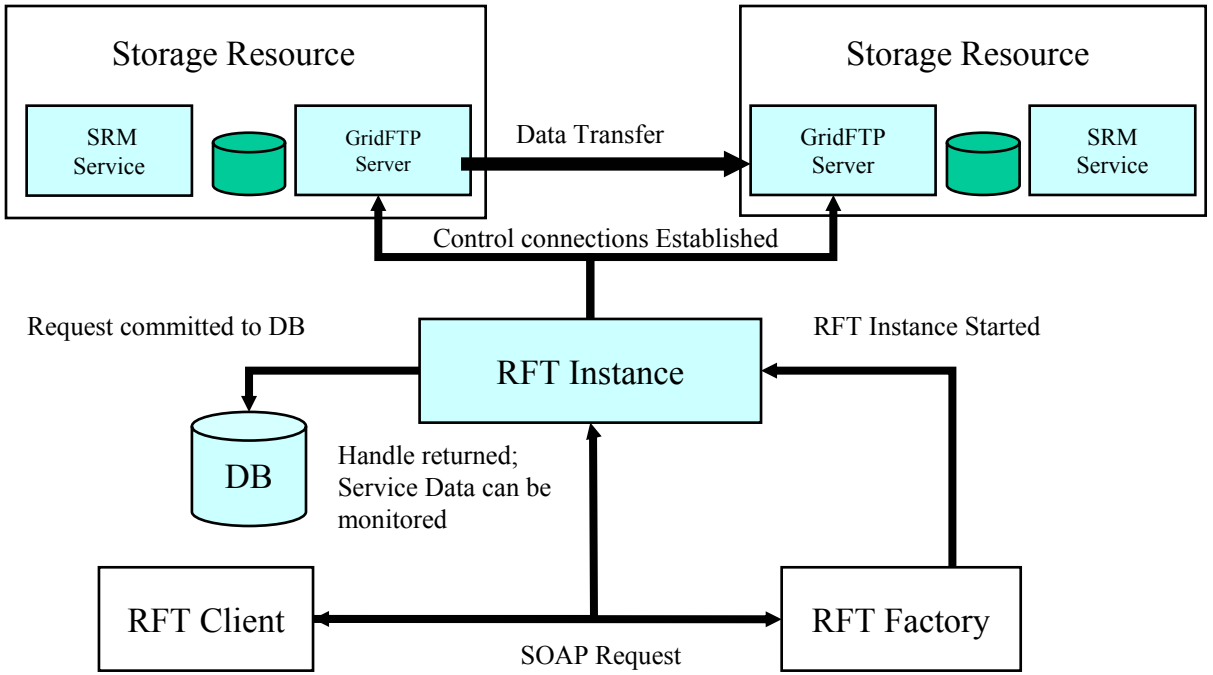


- OGSI-compliant service exposing GridFTP control channel functionality
- Recoverable with progress and restart monitoring
  - Automatically restarts interrupted transfers from the last checkpoint

http://www-unix.globus.org/toolkit/reliable_transfer.html

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

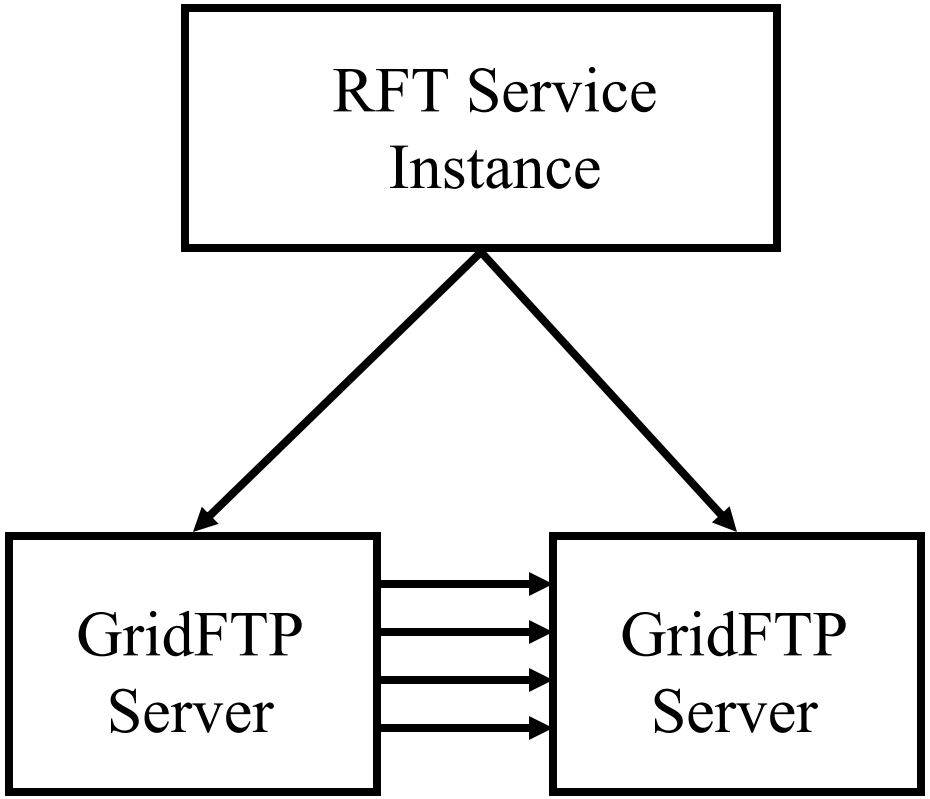


Example TransferRequest

```
true          # type of transfer true=binary false=ascii
16000         # block size in bytes
16000         # TCP buffer size in bytes
false         # No third party transfer (tpt)
1             # Number of parallel streams
true          # Data Channel Authentication (DCAU)
1             # Concurrency - number of concurrent transfers
/DC=org/DC=doegrids/OU=Services/CN=dg0n1.mcs.anl.gov    # Source Host Subject
/DC=org/DC=doegrids/OU=Services/CN=dg0n1.mcs.anl.gov    # Dest Host Subject
gsiftp://dg0n1.mcs.anl.gov/sandbox/madduri/             # Source URL1
gsiftp://dg0n2.mcs.anl.gov/sandbox/madduri/             # Dest URL1
```

RFT in Action

- Service is OGSI compliant
- Uses existing GridFTP (non-OGSI) protocols and tools to execute 3rd Party Transfer for the user
- Provides extensive state transition notification



* The scenarios in this presentation are offered as examples and are not prescriptive

RFT Service Data

- Version
  - version of RFT
- FileTransferProgress
  - Denotes the percentage of file that is transferred
- FileTransferRestartMarker
  - Last restart marker for a particular transfer
- FileTransferJobStatusElement
  - Status of a particular transfer
- FileTransferStatusElement
  - Denotes the status of all the transfers in the request
- GridFTPRestartMarkerElement
  - Raw gridftp restart marker for the transfer
- GridFTPPerfMarkerElement
  - Raw gridftp performance marker for the transfer

RFT Documentation

- http://www.globus.org/toolkit/reliable_transfer.html

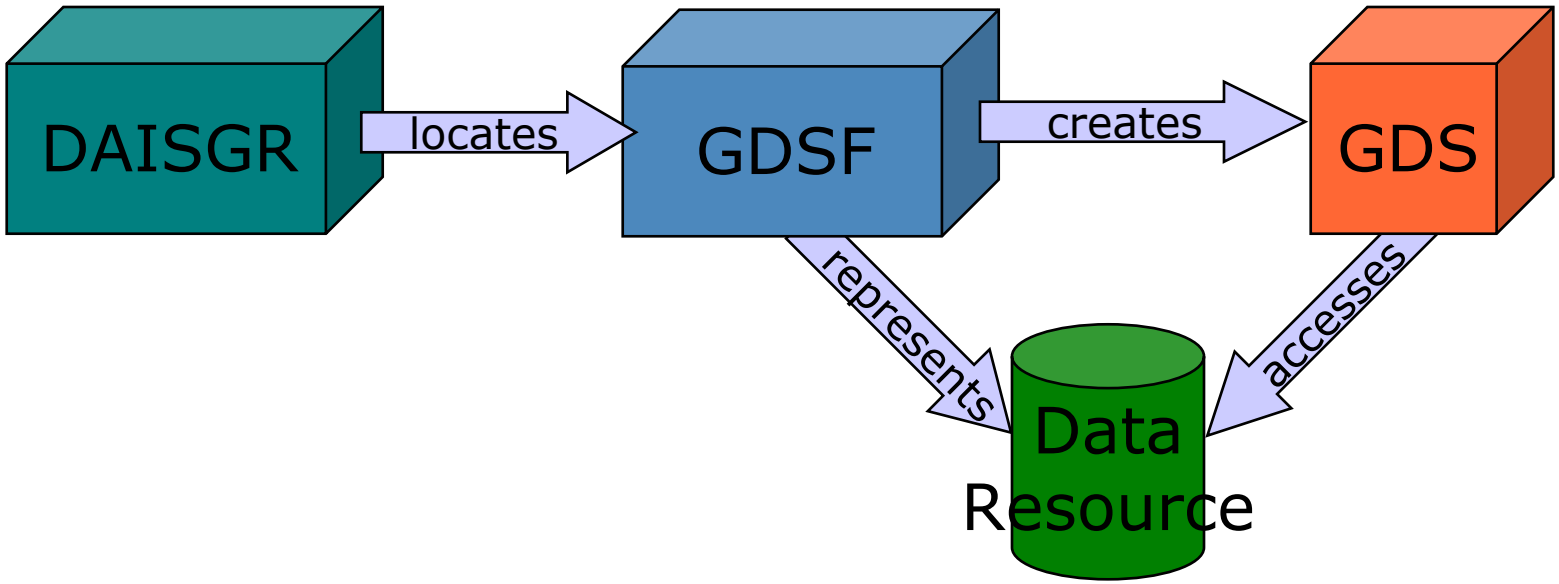
the globus alliance
www.globus.org

OGSA-DAI

- Provides a common access interface to heterogeneous data resources
  - e.g. RDBMS, XMLDBs, structured files
- Platform and language independent
  - BUT does not hide underlying data model
- Provides base for higher-level services
  - Data integration
  - Data federation
- http://www.ogsadai.org.uk

OGSA-DAI Services

- OGSA-DAI uses three main service types
  - DAISGR (registry) for discovery
  - GDSF (factory) to represent a data resource
  - GDS (data service) to access a data resource



the globus alliance
www.globus.org

OGSA-DAI development

- Built around an activity framework
  - Functionality provided in release includes:
    - SQL / XPath statements, Delivery via GridFTP, Compression, XSL Transforms
  - Extensible
    - Developers can add functionality
    - Could import third party trusted activities
  - Allows for optimisation
- Client toolkit for Java provides a quick way to build applications to access OGSA-DAI wrapped data resources

Replica Management in Grids

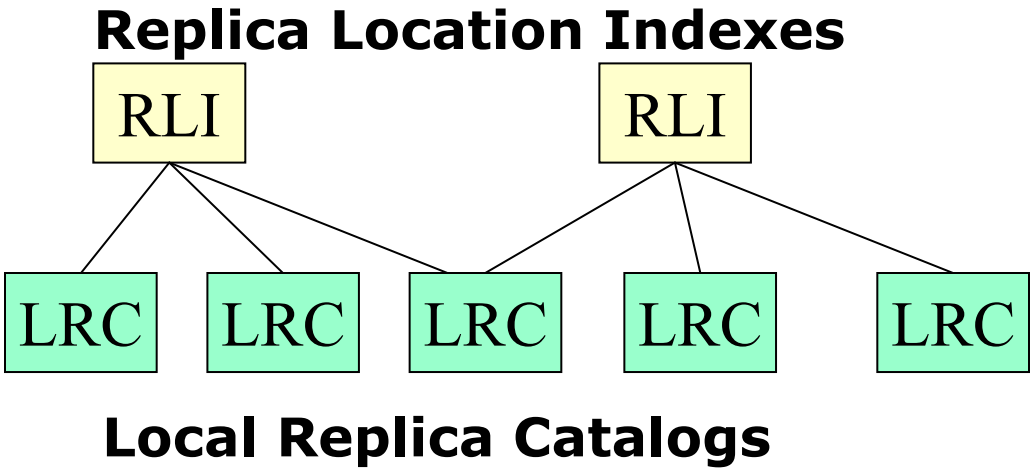
- Data intensive applications
  - Produce Terabytes or Petabytes of data
- Replicate data at multiple locations
  - Fault tolerance
  - Performance: avoid wide area data transfer latencies, achieve load balancing
- Issues:
  - Locating replicas of desired files
  - Creating new replicas and registering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

RLS Framework

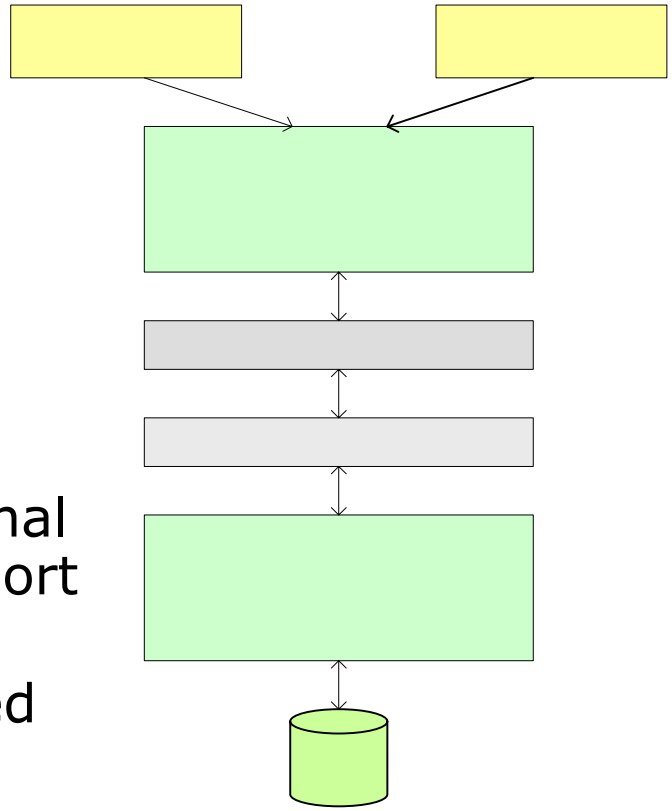
- Local Replica Catalogs (LRCs) contain consistent information about logical-to-target mappings



- Replica Location Index (RLI) nodes aggregate information about one or more LRCs
- LRCs use soft state update mechanisms to inform RLIs about their state: relaxed consistency of index
- Optional compression of state updates reduces communication, CPU and storage overheads
- Membership service registers participating LRCs and RLIs and deals with changes in membership

Components of RLS Implementation

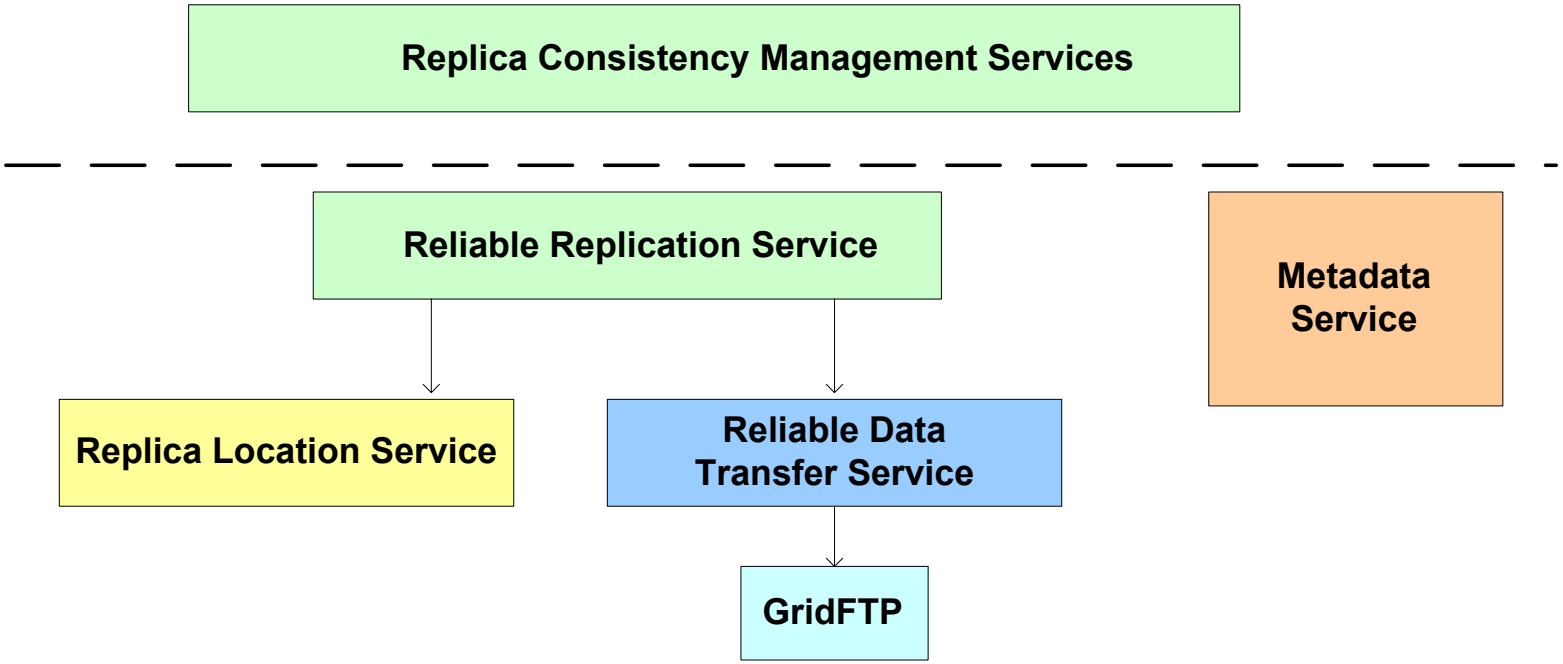
- **Common server implementation for LRC and RLI**
- **Front-End Server**
  - Multi-threaded
  - Written in C
  - Supports GSI Authentication using X.509 certificates
- **Back-end Server**
  - MySQL or PostgreSQL Relational Database (later versions support Oracle)
  - No database back end required for RLIs using Bloom filter compression
- **Client APIs: C and Java**
- **Client Command line tool**



RLS Implementation Features

- Two types of soft state updates from LRCs to RLIs
  - Complete list of logical names registered in LRC
  - Compressed updates: Bloom filter summaries of LRC
- Immediate mode
  - Incremental 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 membership configuration only
  - No membership service

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

globus-rls-cli: Client Command Line 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

Examples of simple create, add and query operations

```
% globus-rls-cli create ln1 pn1 rls://smarty

% globus-rls-cli query lrc lfn ln1 rls://smarty
  ln1: pn1

% globus-rls-cli add ln1 pn2 rls://smarty

% globus-rls-cli query lrc lfn ln1 rls://smarty
  ln1: pn1
  ln1: pn2
```

globus-rli-client Bulk Operations

- **bulk add <lfn> <pfn> [<lfn> <pfn>**
  - Bulk add lfn, pfn mappings
- **bulk delete <lfn> <pfn> [<lfn> <pfn>**
  - Bulk delete lfn, pfn mappings
- **bulk query lrc lfn [<lfn> ...]**
  - Bulk query lrc for lfns
- **bulk query lrc pfn [<pfn> ...]**
  - Bulk query lrc for pfns
- **bulk query rli lfn [<lfn> ...]**
  - Bulk query rli for lfns
- Others bulk attribute adds, deletes, queries, etc.

Examples of Bulk Operations

```
% globus-rls-cli bulk create ln1 pn1 ln2 pn2 ln3 pn3
    rls://smarty

% globus-rls-cli bulk query lrc lfn ln1 ln2 ln3
    rls://smarty

  ln3: pn3
  ln2: pn2
  ln1: pn1
```

LIGO Data Replication Challenge

- Replicate 200 GB/day of data to multiple sites securely, efficiently, robustly
- Support a number of storage models at sites
  - CIT → SAM-QFS (tape) and large IDE farms
  - UWM → 600 partitions on 300 cluster nodes
  - PSU → multiple 1 TB RAID-5 servers
  - AEI → 150 partitions on 150 nodes with redundancy
- Coherent mechanism for data discovery
- Know what data we have, where it is, and replicate it fast and easy

Lightweight Data Replicator (LDR)

- What data we have…
  - Globus Metadata Catalog Service (MCS)
- Where data is…
  - Globus Replica Location Service (RLS)
- Replicate it fast…
  - Globus GridFTP protocol
  - What client to use? Right now we use our own
- Replicate it easy…
  - pyGlobus daemons

the globus alliance
www.globus.org

LDR Roles

- Publisher
  - Provides information about available files, location, metadata
- Provider
  - Makes files available for replication
- Subscriber
  - Replicates data from a provider to itself

LDR Daemons

- LDRMetadata
  - MySQL Backend, replicates metadata catalog using GridFTP
- LDRWant
  - Figures out what files are missing, uses Globus RLS (Replica Locator Service)
- LDRSchedule
  - Queues Transfers
- LDRTransfer
  - Manage transfer of files using GridFTP
- LDRVerify
  - Checks integrity of replicas

the globus alliance
www.globus.org

For more information

- Globus Components
  - www.globus.org
- OGSA-DAI
  - http://www.ogsadai.org.uk/
- LDR
  - http://www.lsc-group.phys.uwm.edu/LDR/