



# Introduction to The Globus Toolkit™

The Globus Project™

Argonne National Laboratory  
USC Information Sciences Institute

<http://www.globus.org/>

Copyright (c) 2002 University of Chicago and The University of Southern California. All Rights Reserved.  
This presentation is licensed for use under the terms of the Globus Toolkit Public License.  
See <http://www.globus.org/toolkit/download/license.html> for the full text of this license.

# Globus Toolkit™

- A software toolkit addressing key technical problems in the development of Grid enabled tools, services, and applications
  - Offer a modular “bag of technologies”
  - Enable *incremental* development of grid-enabled tools and applications
  - Implement standard Grid protocols and APIs
  - Make available under liberal open source license

## General Approach

- Define Grid protocols & APIs
  - Protocol-mediated access to remote resources
  - Integrate and extend existing standards
  - “On the Grid” = speak “Intergrid” protocols
- Develop a reference implementation
  - Open source Globus Toolkit
  - Client and server SDKs, services, tools, etc.
- Grid-enable wide variety of tools
  - Globus Toolkit, FTP, SSH, Condor, SRB, MPI, ...
- Learn through deployment and applications

# Key Protocols

- The Globus Toolkit™ centers around four key protocols
  - Connectivity layer:
    - > *Security*: Grid Security Infrastructure (GSI)
  - Resource layer:
    - > *Resource Management*: Grid Resource Allocation Management (GRAM)
    - > *Information Services*: Grid Resource Information Protocol (GRIP)
    - > *Data Transfer*: Grid File Transfer Protocol (GridFTP)
- Also key collective layer protocols
  - Info Services, Replica Management, etc.



# The Globus Toolkit™: APIs

## Role of APIs

- While we focus heavily on protocols, the Globus Toolkit is an implementation, and as such requires APIs
  - Globus Toolkit implemented in C
  - Great effort has gone into implementing robust, consistent, and flexible APIs
- APIs in other languages also available
  - E.g. Java & Python CoG Kits

## Three Types of API/SDK

- Portability and convenience API/SDKs
- API/SDKs implementing the four key Connectivity and Resource layer protocols
- Collective layer API/SDKs
  
- This tutorial focuses primarily on the functionality available in #2 and #3
- Developer tutorial includes in-depth API discussions of all three

# Portability and Convenience API

- `globus_common`
  - Module activation/deactivation
  - Threads, mutual exclusion, conditions
  - Callback/event driver
  - Libc wrappers
  - Convenience modules (list, hash, etc).

## Connectivity APIs

- globus\_io
  - TCP, UDP, IP multicast, and file I/O
  - Integrates GSI security
  - Asynchronous and synchronous interfaces
  - Attribute based control of behavior
- Nexus (Deprecated)
  - Higher level, active message style comms
  - Built on globus\_io, but without security
- MPICH-G2
  - High level, MPI (send/receive) interface
  - Built on globus\_io and native MPI



# The Globus Toolkit™: Security

# Security Terminology

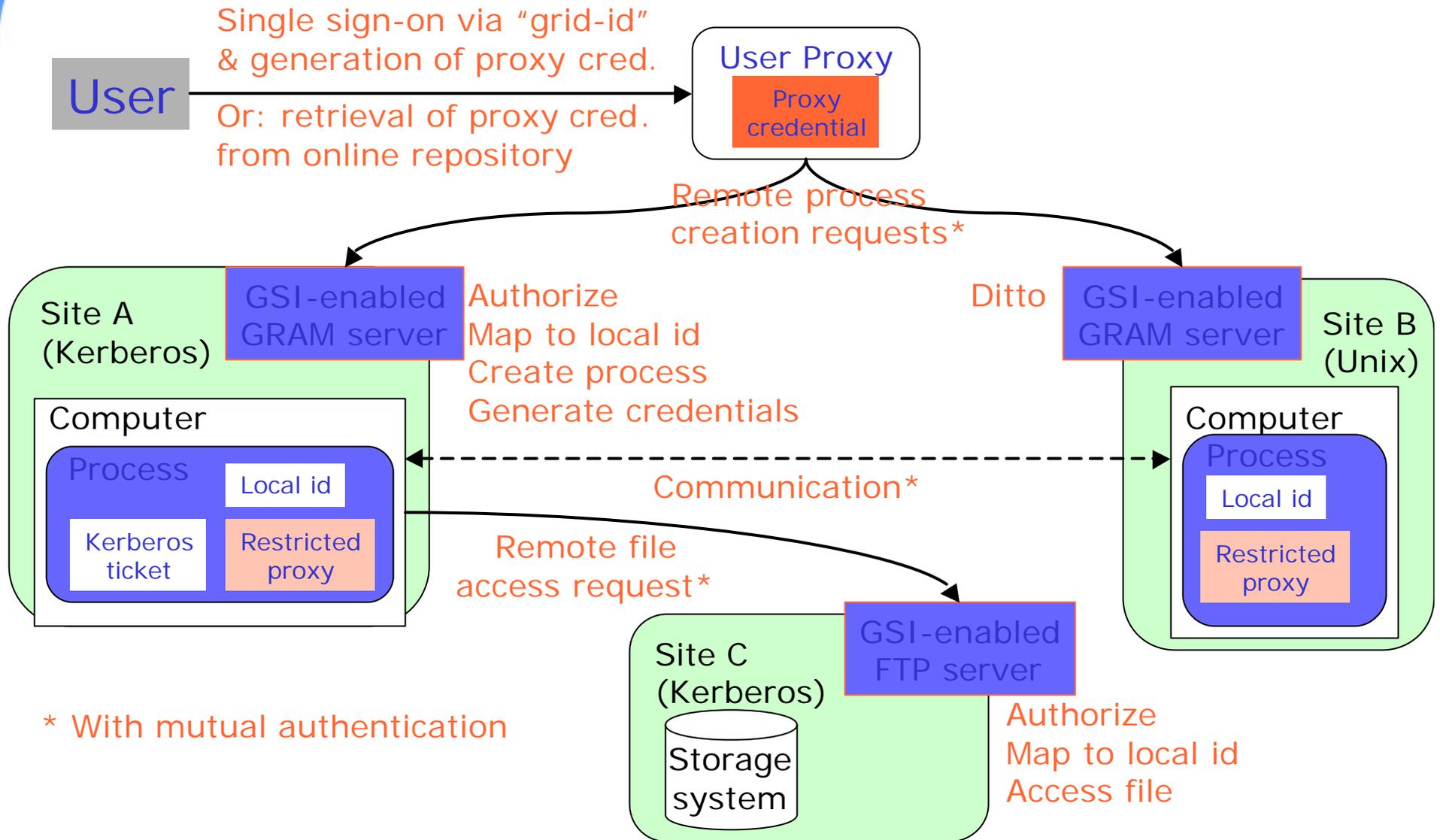
- Authentication: Establishing identity
- Authorization: Establishing rights
- Message protection
  - Message integrity
  - Message confidentiality
- Non-repudiation
- Digital signature
- Accounting
- Certificate Authority (CA)

## Why Grid Security is Hard

- Resources being used may be valuable & the problems being solved sensitive
- Resources are often located in distinct administrative domains
  - Each resource has own policies & procedures
- Set of resources used by a single computation may be large, dynamic, and unpredictable
  - Not just client/server, requires delegation
- It must be broadly available & applicable
  - Standard, well-tested, well-understood protocols; integrated with wide variety of tools

# GSI in Action

"Create Processes at A and B that Communicate & Access Files at C"



\* With mutual authentication

# Grid Security Requirements

## User View

- 1) Easy to use
- 2) Single sign-on
- 3) Run applications  
ftp,ssh,MPI,Condor,Web,...
- 4) User based trust model
- 5) Proxies/agents (delegation)

## Resource Owner View

- 1) Specify local access control
- 2) Auditing, accounting, etc.
- 3) Integration w/ local system  
Kerberos, AFS, license mgr.
- 4) Protection from compromised resources

## Developer View

API/SDK with authentication, flexible message protection,  
flexible communication, delegation, ...

Direct calls to various security functions (e.g. GSS-API)

Or security integrated into higher-level SDKs:

E.g. GlobusIO, Condor-G, MPICH-G2, HDF5, etc.

# Candidate Standards

- Kerberos 5
  - Fails to meet requirements:
    - > Integration with various local security solutions
    - > User based trust model
- Transport Layer Security (TLS/SSL)
  - Fails to meet requirements:
    - > Single sign-on
    - > Delegation

# Grid Security Infrastructure (GSI)

- Extensions to standard protocols & APIs
  - Standards: SSL/TLS, X.509 & CA, GSS-API
  - Extensions for single sign-on and delegation
- Globus Toolkit reference implementation of GSI
  - SSLeay/OpenSSL + GSS-API + SSO/delegation
  - Tools and services to interface to local security
    - > Simple ACLs; SSLK5/PKINIT for access to K5, AFS; ...
  - Tools for credential management
    - > Login, logout, etc.
    - > Smartcards
    - > MyProxy: Web portal login and delegation
    - > K5cert: Automatic X.509 certificate creation

# Review of Public Key Cryptography

- Asymmetric keys
  - A **private** key is used to encrypt data.
  - A **public** key can decrypt data encrypted with the private key.
- An X.509 certificate includes...
  - Someone's subject name (user ID)
  - Their public key
  - A "signature" from a Certificate Authority (CA) that:
    - > Proves that the certificate came from the CA.
    - > Vouches for the subject name
    - > Vouches for the binding of the public key to the subject

## Public Key Based Authentication

- User sends certificate over the wire.
- Other end sends user a challenge string.
- User encodes the challenge string with private key
  - Possession of private key means you can authenticate as subject in certificate
- Public key is used to decode the challenge.
  - If you can decode it, you know the subject
- Treat your private key carefully!!
  - Private key is stored only in well-guarded places, and only in encrypted form

## X.509 Proxy Certificate

- Defines how a short term, restricted credential can be created from a normal, long-term X.509 credential
  - A “proxy certificate” is a special type of X.509 certificate that is signed by the normal end entity cert, or by another proxy
  - Supports single sign-on & delegation through “impersonation”
  - Currently an IETF draft

## User Proxies

- Minimize exposure of user's private key
- A temporary, X.509 proxy credential for use by our computations
  - We call this a user proxy certificate
  - Allows process to act on behalf of user
  - User-signed user proxy cert stored in local file
  - Created via "grid-proxy-init" command
- Proxy's private key is not encrypted
  - Rely on file system security, proxy certificate file must be readable only by the owner

# Delegation

- Remote creation of a user proxy
- Results in a new private key and X.509 proxy certificate, signed by the original key
- Allows remote process to act on behalf of the user
- Avoids sending passwords or private keys across the network

## Globus Security APIs

- Generic Security Service (GSS) API
  - IETF standard
  - Provides functions for authentication, delegation, message protection
  - Decoupled from any particular communication method
- GSS-API Extensions (GGF draft)
  - Small extensions to GSS
- But GSS-API is complicated, so we also provide the easier `globus_gss_assist` API.
- GSI-enabled SASL is also provided

## Results

- GSI adopted by 100s of sites, 1000s of users
  - Globus CA has issued >4000 certs (user & host), >1500 currently active; other CAs active
- Rollouts are currently underway all over:
  - NSF Teragrid, NASA Information Power Grid, DOE Science Grid, European Data Grid, etc.
- Integrated in research & commercial apps
  - GrADS testbed, Earth Systems Grid, European Data Grid, GriPhyN, NEESgrid, etc.
- Standardization begun in Global Grid Forum, IETF

## GSI Applications

- Globus Toolkit™ uses GSI for authentication
- Many Grid tools, directly or indirectly, e.g.
  - Condor-G, SRB, MPICH-G2, Cactus, GDMP, ...
- Commercial and open source tools, e.g.
  - ssh, ftp, cvs, OpenLDAP, OpenAFS
  - SecureCRT (Win32 ssh client)
- And since we use standard X.509 certificates, they can also be used for
  - Web access, LDAP server access, etc.

## Ongoing and Future GSI Work

- Protection against compromised resources
  - Restricted delegation, smartcards
- Standardization
- Scalability in numbers of users & resources
  - Credential management
  - Online credential repositories (“MyProxy”)
  - Account management
- Authorization
  - Policy languages
  - Community authorization

## Restricted Proxies

- Q: How to restrict rights of delegated proxy to a subset of those associated with the issuer?
- A: Embed restriction policy in proxy cert
  - Policy is evaluated by resource upon proxy use
  - Reduces rights available to the proxy to a subset of those held by the user
- But how to avoid policy language wars?
  - Proxy cert just contains a container for a policy specification, without defining the language
    - > Container = OID + blob
  - Can evolve policy languages over time

## Delegation Tracing

- Often want to know through what entities a proxy certificate has been delegated
  - Audit (retrace footsteps)
  - Authorization (deny from bad entities)
- Solved by adding information to the signed proxy certificate about each entity to which a proxy is delegated.
  - Does NOT guarantee proper use of proxy
  - Just tells you which entities were purposely involved in a delegation

## Proxy Certificate Standards Work

- “Internet Public Key Infrastructure X.509 Proxy Certificate Profile”
  - draft-ietf-pkix-proxy-01.txt
    - > Draft being considered by IETF PKIX working group, and by GGF GSI working group
  - Defines proxy certificate format, including restricted rights and delegation tracing
- Demonstrated a prototype of restricted proxies at HPDC (August 2001) as part of CAS demo

# Delegation Protocol Work

- “TLS Delegation Protocol”
  - draft-ietf-tls-delegation-01.txt
    - > Draft being considered by IETF TLS working group, and by GGF GSI working group
  - Defines how to remotely delegate an X.509 Proxy Certificate using extensions to the TLS (SSL) protocol
- But, may change approach here
  - Instead of embedding into TLS, carry on top of TLS
  - This is the current approach in Globus Toolkit

## GSS-API Extensions Work

- 4 years of GSS-API experience, while on the whole quite positive, has shed light on various deficiencies of GSS-API
- “GSS-API Extensions”
  - draft-ggf-gss-extensions-04.txt
    - > Draft being considered by GGF GSI working group. Not yet submitted to IETF.
  - Defines extensions to the GSS-API to better support Grid security

## GSS-API Extensions

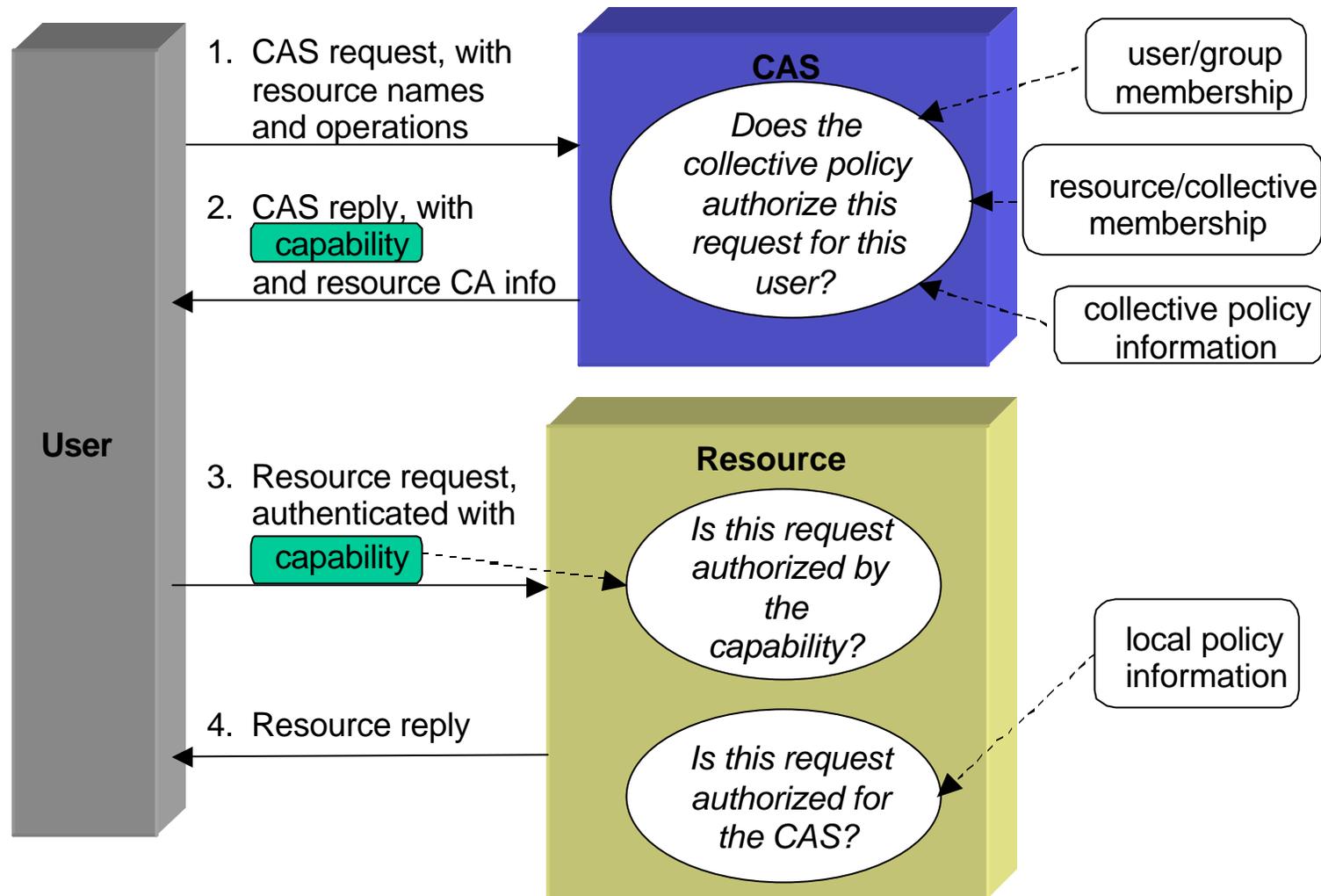
- Credential export/import
  - Allows delegated credentials to be externalized
  - Used for checkpointing a service
- Delegation at any time, in either direction
  - More rich options on use of delegation
- Restricted delegation handling
  - Add proxy restrictions to delegated cred
  - Inspect auth cert for restrictions
- Allow better mapping of GSS to TLS
  - Support TLS framing of messages

# Community Authorization Service

- Question: How does a large community grant its users access to a large set of resources?
  - Should minimize burden on both the users and resource providers
- Community Authorization Service (CAS)
  - Community negotiates access to resources
  - Resource outsources fine-grain authorization to CAS
  - Resource only knows about “CAS user” credential
    - > CAS handles user registration, group membership...
  - User who wants access to resource asks CAS for a capability credential
    - > Restricted proxy of the “CAS user” cred., checked by resource



# Community Authorization (Prototype shown August 2001)



# Community Authorization Service

- CAS provides user community with information needed to authenticate resources
  - Sent with capability credential, used on connection with resource
  - Resource identity (DN), CA
- This allows new resources/users (and their CAs) to be made available to a community through the CAS without action on the other user's/resource's part

## Authorization API

- Service providers need to perform authorization policy evaluation on:
  - Local policies
  - Policies contained in restricted proxies
- We are working on 2 API layers:
  - Low level GAA-API implementation for evaluation of policies
  - High level, very simple authorization API that can easily be embedded into services
- Still in early prototyping stage

## Passport Online CA & MyProxy

- Requiring users to manage their own certs and keys is annoying and error prone
- A solution: Leverage Passport global authentication to obtain a proxy credential
  - Passport provides
    - > Globally unique user name (email address)
    - > Method of verifying ownership of the name (authentication)
    - > Re-issuance (e.g. forgotten password)
  - Passport credentials can be presented to an online CA or credential repository
    - > Creates and issues new (restricted) proxy certificate to the user on demand

## Other Future Security Work

- Ease-of-use
  - Improved error message, online CA, etc.
- Improved online credential repositories
  - See MyProxy paper at HPDC
- Support for multiple user credentials
- Multi-factor authentication
- Subordinate certificate authorities for domains
  - Ease issuance of host certs for domains
- Independent Data Unit Support

## Security Summary

- GSI successfully addresses wide variety of Grid security issues
- Broad acceptance, deployment, integration with tools
- Standardization on-going in IETF & GGF
- Ongoing R&D to address next set of issues
- For more information:
  - [www.globus.org/research/papers.html](http://www.globus.org/research/papers.html)
    - > “A Security Architecture for Computational Grids”
    - > “Design and Deployment of a National-Scale Authentication Infrastructure”
  - [www.gridforum.org/security](http://www.gridforum.org/security)



# The Globus Toolkit™: Resource Management

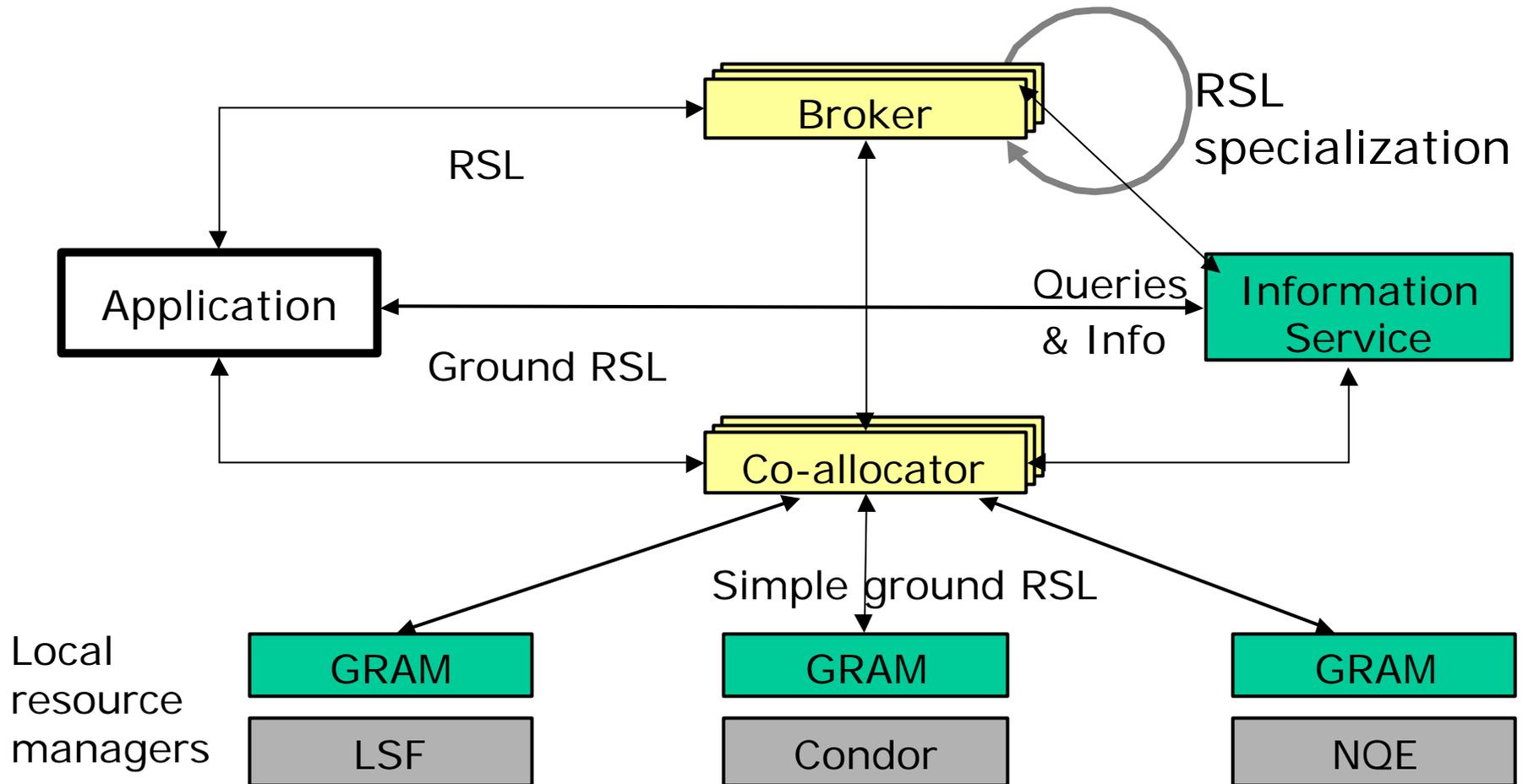
## The Challenge

- Enabling secure, controlled remote access to heterogeneous computational resources and management of remote computation
  - Authentication and authorization
  - Resource discovery & characterization
  - Reservation and allocation
  - Computation monitoring and control
- Addressed by new protocols & services
  - GRAM protocol as a basic building block
  - Resource brokering & co-allocation services
  - GSI for security, MDS for discovery

# Resource Management

- The Grid Resource Allocation Management (GRAM) protocol and client API allows programs to be started on remote resources, despite local heterogeneity
- Resource Specification Language (RSL) is used to communicate requirements
- A layered architecture allows application-specific resource brokers and co-allocators to be defined in terms of GRAM services
  - Integrated with Condor, PBS, MPICH-G2, ...

# Resource Management Architecture



# Resource Specification Language

- Common notation for exchange of information between components
  - Syntax similar to MDS/LDAP filters
- RSL provides two types of information:
  - Resource requirements: Machine type, number of nodes, memory, etc.
  - Job configuration: Directory, executable, args, environment
- Globus Toolkit provides an API/SDK for manipulating RSL

# RSL Syntax

- Elementary form: parenthesis clauses
  - (attribute op value [ value ... ] )
- Operators Supported:
  - <, <=, =, >=, >, !=
- Some supported attributes:
  - executable, arguments, environment, stdin, stdout, stderr, resourceManagerContact, resourceManagerName
- Unknown attributes are passed through
  - May be handled by subsequent tools

## Constraints: "&"

- For example:
  - & (count >= 5) (count <= 10)
  - (max\_time=240) (memory >= 64)
  - (executable=myprog)
- "Create 5-10 instances of **myprog**, each on a machine with at least 64 MB memory that is available to me for 4 hours"

## Disjunction: “|”

- For example:  
    & (executable=myprog)  
      ( | (&(count=5)(memory >=64))  
        (&(count=10)(memory >=32)))
- Create 5 instances of myprog on a machine that has at least 64MB of memory, or 10 instances on a machine with at least 32MB of memory

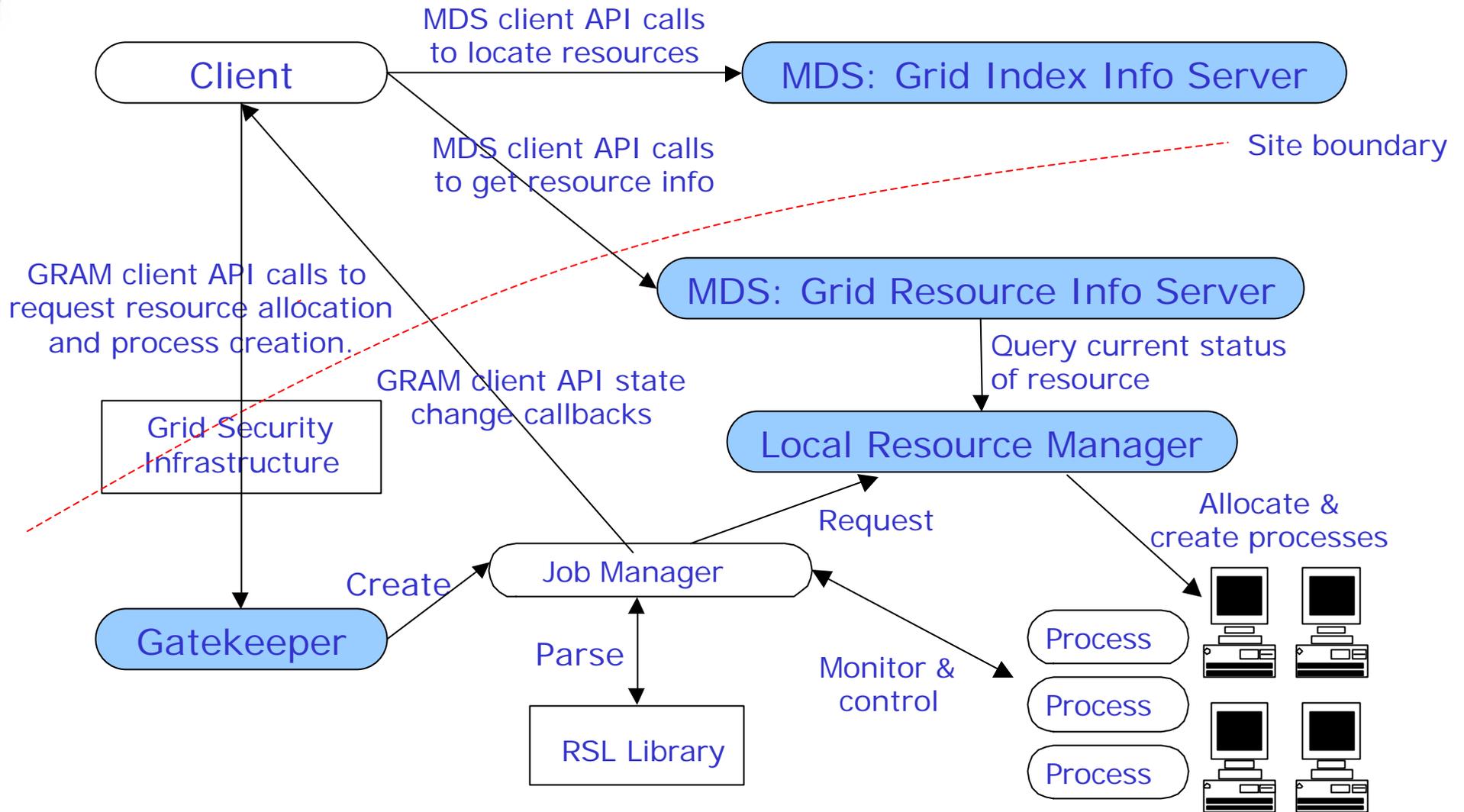
# GRAM Protocol

- GRAM-1: Simple HTTP-based RPC
  - Job request
    - > Returns a “job contact”: Opaque string that can be passed between clients, for access to job
  - Job cancel, status, signal
  - Event notification (callbacks) for state changes
    - > Pending, active, done, failed, suspended
- GRAM-1.5 (U Wisconsin contribution)
  - Add reliability improvements
    - > Once-and-only-once submission
    - > Recoverable job manager service
    - > Reliable termination detection
- GRAM-1.6: Incremental additions
- GRAM-2: Future overhaul, based on Web services

# Globus Toolkit Implementation

- Gatekeeper
  - Single point of entry
  - Authenticates user, maps to local security environment, runs service
  - In essence, a “secure inetd”
- Job manager
  - A gatekeeper service
  - Layers on top of local resource management system (e.g., PBS, LSF, etc.)
  - Handles remote interaction with the job

# GRAM Components



## Co-allocation

- Simultaneous allocation of a resource set
  - Handled via optimistic co-allocation based on free nodes or queue prediction
  - In the future, advance reservations will also be supported (already in prototype)
- Globus APIs/SDKs support the co-allocation of specific multi-requests
  - Uses a Globus component called the Dynamically Updated Request Online Co-allocator (DUROC)

## Multirequest: “ + ”

- A multirequest allows us to specify multiple resource needs, for example
  - + (& (count=5)(memory >=64)  
(executable=p1))  
(&(network=atm) (executable=p2))
    - Execute 5 instances of p1 on a machine with at least 64M of memory
    - Execute p2 on a machine with an ATM connection
- Multirequests are central to co-allocation



## A Co-allocation Multirequest

```
+ ( & (resourceManagerContact=  
  "flash.isi.edu: 754: /C=US/.../CN=flash.isi.edu-fork")  
  (count=1)  
  (label="subjob A")  
  (executable=my_app1)  
)  
Different counts  
( & (resourceManagerContact=  
  "sp139.sdsc.edu: 8711: /C=US/.../CN=sp097.sdsc.edu-1st")  
  (count=2)  
  (label="subjob B")  
  (executable=my_app2)  
)  
Different executables
```

Different resource managers

## Job Submission Interfaces

- Globus Toolkit includes several command line programs for job submission
  - globus-job-run: Interactive jobs
  - globus-job-submit: Batch/offline jobs
  - globusrun: Flexible scripting infrastructure
- Others are building better interfaces
  - General purpose
    - > Condor-G, PBS, GRD, Hotpage, etc
  - Application specific
    - > ECCE', Cactus, Web portals

## globus-job-run

- For running of interactive jobs
- Additional functionality beyond rsh
  - Ex: Run 2 process job w/ executable staging  
`globus-job-run -: host -np 2 -s myprog arg1 arg2`
  - Ex: Run 5 processes across 2 hosts  
`globus-job-run \  
-: host1 -np 2 -s myprog.linux arg1 \  
-: host2 -np 3 -s myprog.aix arg2`
  - For list of arguments run:  
`globus-job-run -help`

## globus-job-submit

- For running of batch/offline jobs
  - **globus-job-submit**                      Submit job
    - > Same interface as globus-job-run
    - > Returns immediately
  - **globus-job-status**                      Check job status
  - **globus-job-cancel**                      Cancel job
  - **globus-job-get-output**                  Get job stdout/err
  - **globus-job-clean**                      Cleanup after job

# globusrun

- Flexible job submission for scripting
  - Uses an RSL string to specify job request
  - Contains an embedded globus-gass-server
    - > Defines GASS URL prefix in RSL substitution variable:  
(stdout=\$(GLOBUSRUN\_GASS\_URL)/stdout)
  - Supports both interactive and offline jobs
- Complex to use
  - Must write RSL by hand
  - Must understand its esoteric features
  - Generally you should use globus-job-\* commands instead

# Resource Management APIs

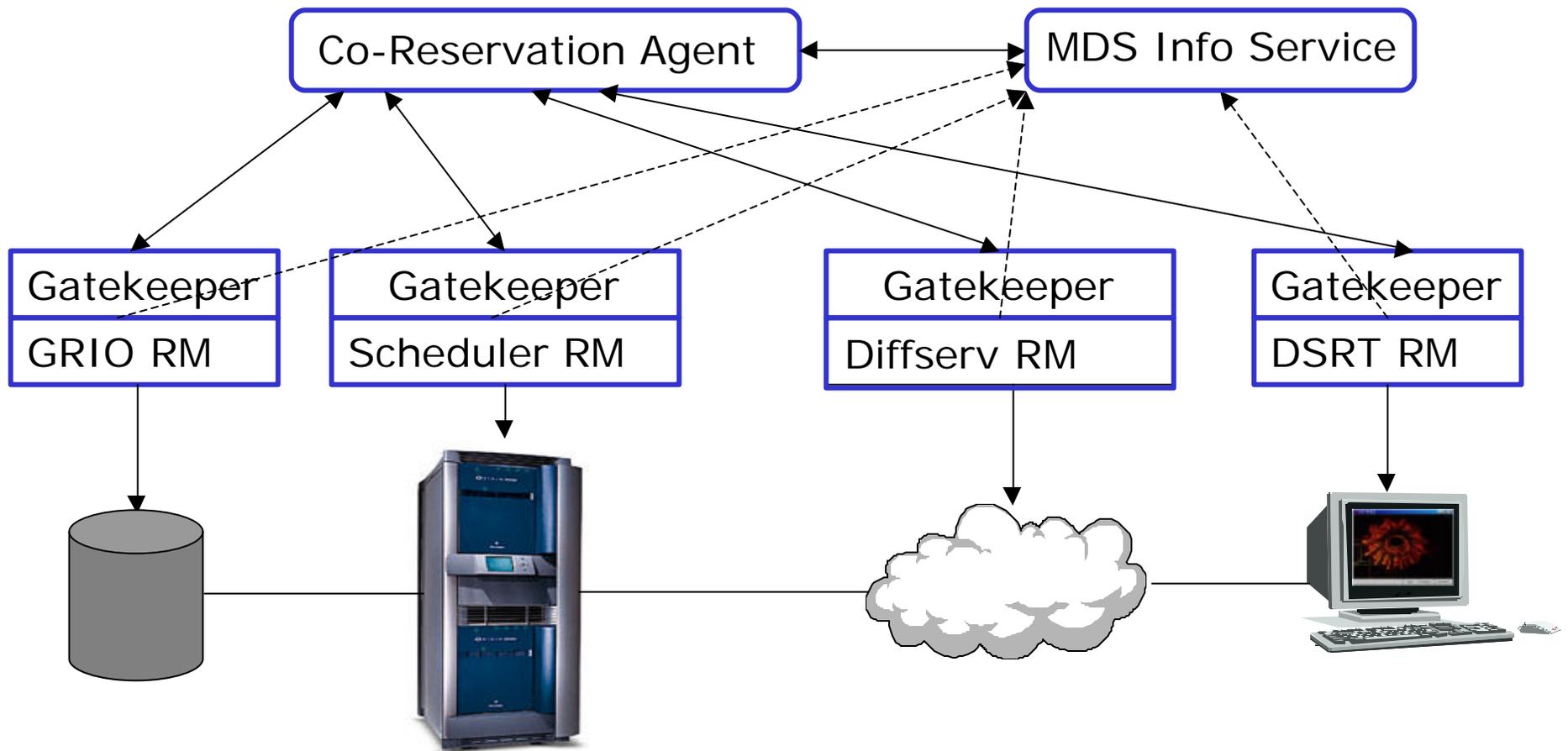
- The `globus_gram_client` API provides access to all of the core job submission and management capabilities, including callback capabilities for monitoring job status.
- The `globus_rsl` API provides convenience functions for manipulating and constructing RSL strings.
- The `globus_gram_myjob` allows multi-process jobs to self-organize and to communicate with each other.
- The `globus_duroc_control` and `globus_duroc_runtime` APIs provide access to multirequest (co-allocation) capabilities.



# Advance Reservation and Other Generalizations

- General-purpose Architecture for Reservation and Allocation (GARA)
  - 2nd generation resource management services
- Broadens GRAM on two axes
  - Generalize to support various resource types
    - > CPU, storage, network, devices, etc.
  - Advance reservation of resources, in addition to allocation
- Currently a research prototype

# GARA: The Big Picture



# Resource Management Futures: GRAM-1.6 (planned for 2Q02)

- Asynchronous client API
- New RSL attribute to pass through scheduler specific commands
  - No more piggy-backing on the environment attributes
- File staging
  - scratch dir, input, output
- Advanced output management
  - Stream/store stdout and stderr to multiple destinations



# Resource Management Futures: GRAM-2 (planned for late 2002)

- Advance reservations
  - As prototyped in GARA in previous 2 years
- Multiple resource types
  - Manage anything: storage, networks, etc., etc.
- Recoverable requests, timeout, etc.
- Better lifetime management
- Policy evaluation points for restricted proxies
- Use of Web Services (WSDL, SOAP)

Karl Czajkowski, Steve Tuecke, others



# The Globus Toolkit™: Data Management

## Data Grid Problem

- “Enable a geographically distributed community [of thousands] to pool their resources in order to perform sophisticated, computationally intensive analyses on Petabytes of data”
- Note that this problem:
  - Is common to many areas of science
  - Overlaps strongly with other Grid problems

## Major Data Grid Projects

- Earth System Grid (DOE Office of Science)
  - DG technologies, climate applications
- European Data Grid (EU)
  - DG technologies & deployment in EU
- GriPhyN (NSF ITR)
  - Investigation of “Virtual Data” concept
- Particle Physics Data Grid (DOE Science)
  - DG applications for HENP experiments

# Data Grids for High Energy Physics

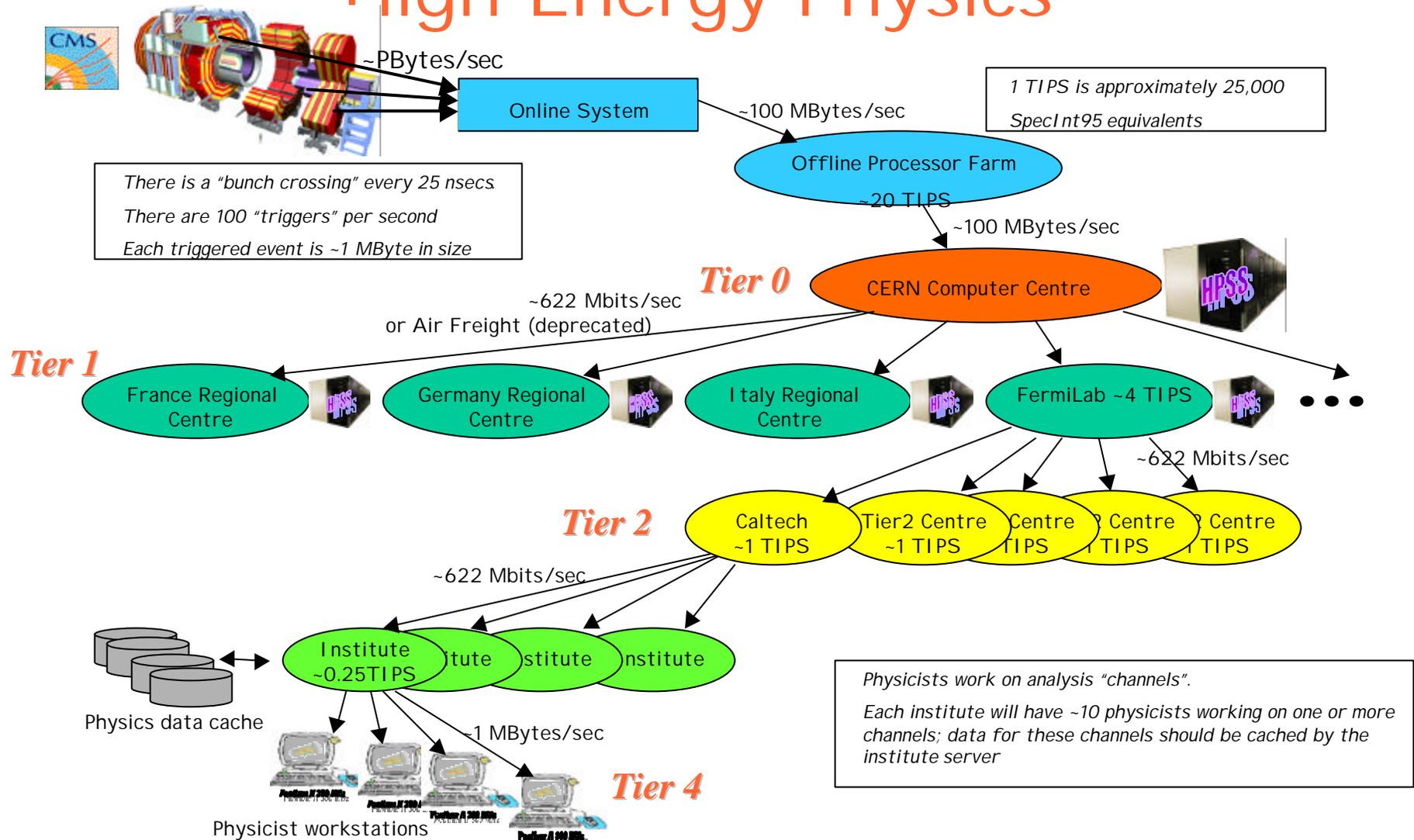


Image courtesy Harvey Newman, Caltech

## Data Intensive Issues Include ...

- Harness [potentially large numbers of] data, storage, network resources located in distinct administrative domains
- Respect local and global policies governing what can be used for what
- Schedule resources efficiently, again subject to local and global constraints
- Achieve high performance, with respect to both speed and reliability
- Catalog software and virtual data

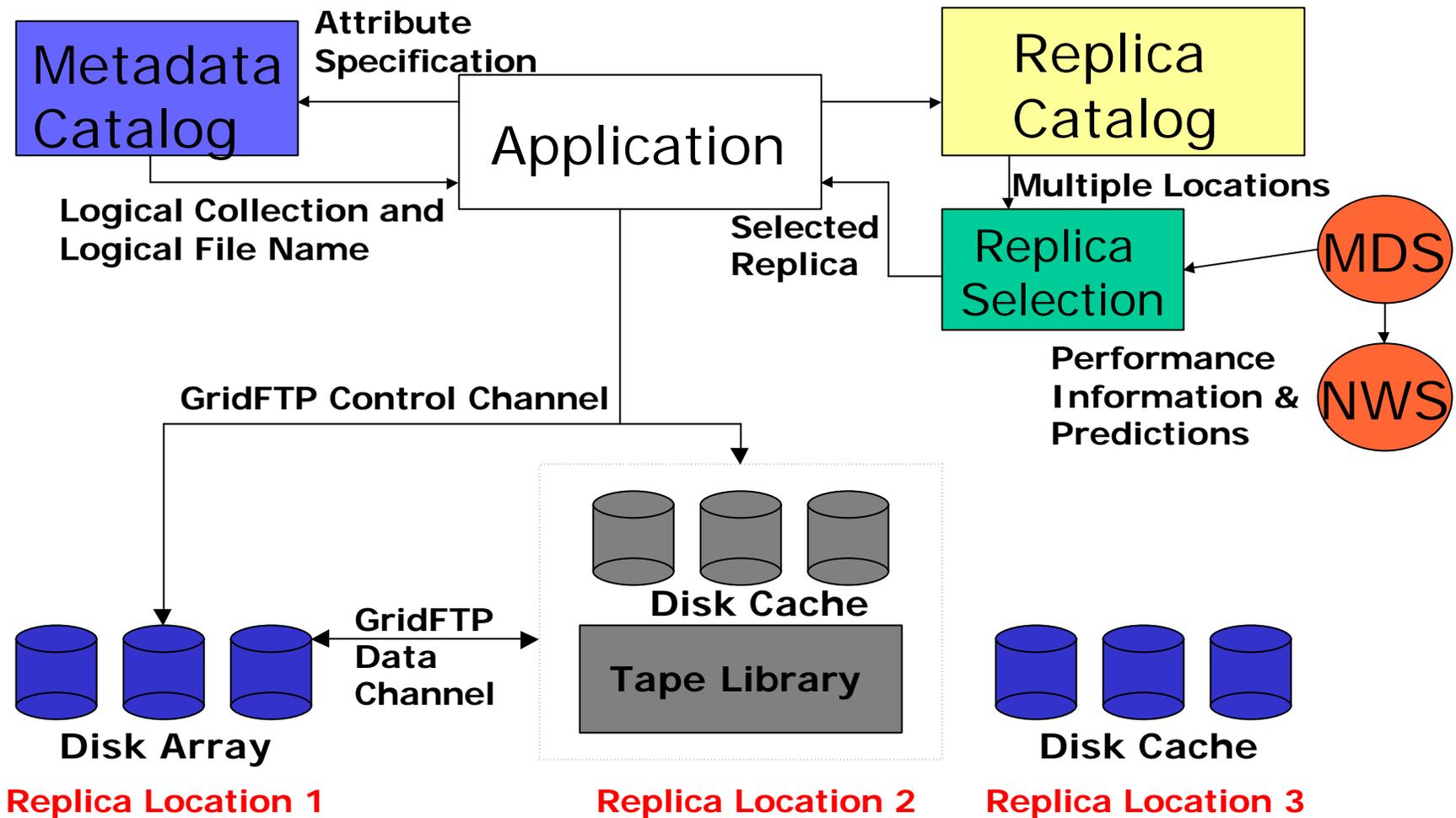
# Data Intensive Computing and Grids

- The term “Data Grid” is often used
  - Unfortunate as it implies a distinct infrastructure, which it isn't; but easy to say
- Data-intensive computing shares numerous requirements with collaboration, instrumentation, computation, ...
  - Security, resource mgt, info services, etc.
- Important to exploit commonalities as very unlikely that multiple infrastructures can be maintained
- Fortunately this seems easy to do!

# Examples of Desired Data Grid Functionality

- High-speed, reliable access to remote data
- Automated discovery of “best” copy of data
- Manage replication to improve performance
- Co-schedule compute, storage, network
- “Transparency” wrt delivered performance
- Enforce access control on data
- Allow representation of “global” resource allocation policies

# A Model Architecture for Data Grids



# Globus Toolkit Components

## Two major Data Grid components:

### 1. Data Transport and Access

- Common protocol
  - Secure, efficient, flexible, extensible data movement
- Family of tools supporting this protocol

### 2. Replica Management Architecture

- Simple scheme for managing:
  - multiple copies of files
  - collections of files

# Motivation for a Common Data Access Protocol

- Existing distributed data storage systems
  - DPSS, HPSS: focus on high-performance access, utilize parallel data transfer, striping
  - DFS: focus on high-volume usage, dataset replication, local caching
  - SRB: connects heterogeneous data collections, uniform client interface, metadata queries
- Problems
  - Incompatible (and proprietary) protocols
    - > Each require custom client
    - > Partitions available data sets and storage devices
  - Each protocol has subset of desired functionality

# A Common, Secure, Efficient Data Access Protocol

- Common, *extensible* transfer protocol
  - Common protocol means all can interoperate
- Decouple low-level data transfer mechanisms from the storage service
- Advantages:
  - New, specialized storage systems are automatically compatible with existing systems
  - Existing systems have richer data transfer functionality
- Interface to many storage systems
  - HPSS, DPSS, file systems
  - Plan for SRB integration

# Access/Transport Protocol Requirements

- Suite of communication libraries and related tools that support
  - GSI, Kerberos security
  - Third-party transfers
  - Parameter set/negotiate
  - Partial file access
  - Reliability/restart
  - Large file support
  - Data channel reuse
  - Integrated instrumentation
  - Loggin/audit trail
  - Parallel transfers
  - Striping (cf DPSS)
  - Policy-based access control
  - Server-side computation
  - Proxies (firewall, load bal)
- All based on a standard, widely deployed protocol

## And The Protocol Is ... GridFTP

- Why FTP?
  - Ubiquity enables interoperation with many commodity tools
  - Already supports many desired features, easily extended to support others
  - Well understood and supported
- We use the term GridFTP to refer to
  - Transfer protocol which meets requirements
  - Family of tools which implement the protocol
- Note GridFTP > FTP
- Note that despite name, GridFTP is not restricted to file transfer!

## GridFTP: Basic Approach

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

- Existing standards
  - RFC 949: 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 Data Working Group

## GridFTP vs. WebDAV

- WebDAV extends http for remote data access
  - Combines control and data over single channel
- FTP splits control and data
  - Supports multiple, user selectable data channel protocols
- Advantage to split channels
  - Third party transfers handled cleanly
  - Can (cleanly) define new data channel protocols
    - > E.g. parallel/striped transfer, automatic TCP buffer/window negotiation, non-TCP based protocols, etc.
  - Amenable to high-performance proxies
    - > E.g. For firewalls, load balancing, etc.

## The GridFTP Family of Tools

- Patches to existing FTP code
  - GSI-enabled versions of existing FTP client and server, for high-quality production code
- Custom-developed libraries
  - Implement full GridFTP protocol, targeting custom use, high-performance
- Custom-developed tools
  - Servers and clients with specialized functionality and performance

## Family of Tools: Patches to Existing Code

- Patches to standard FTP clients and servers
  - gsi-ncftp: Widely used client
  - gsi-wuftpserver: Widely used server
  - GSI modified HPSS pftpd
  - GSI modified Unitree ftpd
- Provides high-quality, production ready, FTP clients and servers
- Integration with common mass storage systems
- Some do not support the full GridFTP protocol

# Family of Tools: Custom Developed Libraries

- Custom developed libraries
  - globus\_ftp\_control: Low level FTP driver
    - > Client & server protocol and connection management
  - globus\_ftp\_client: Simple, reliable FTP client
    - > Plugins for restart, logging, etc.
  - globus\_gass\_copy: Simple URL-to-URL copy library, supporting (gsi-)ftp, http(s), file URLs
- Implement full GridFTP protocol
- Various levels of libraries, allowing implementation of custom clients and servers
- Tuned for high performance on WAN

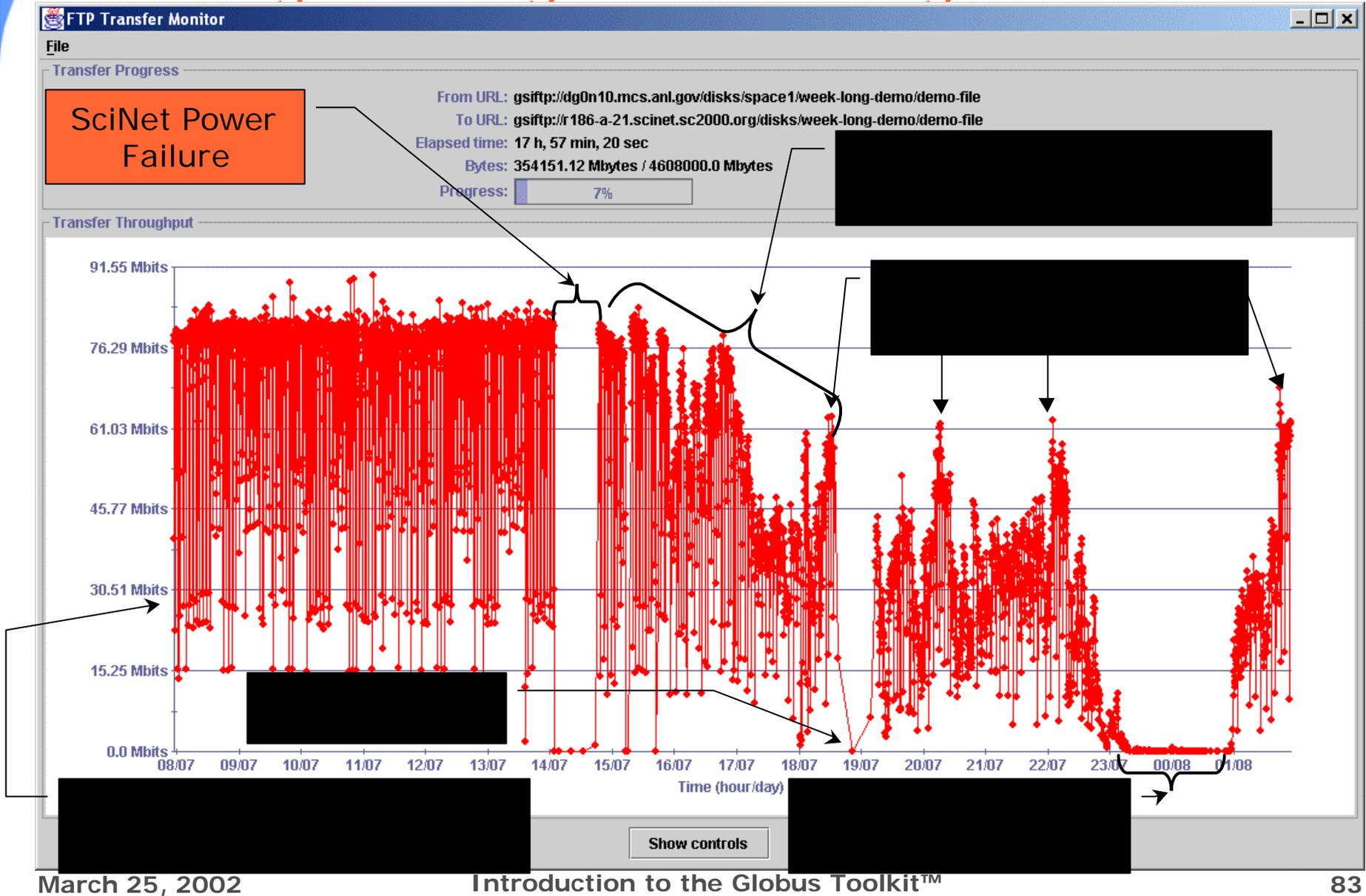
# Family of Tools: Custom Developed Programs

- Simple production client
  - globus-url-copy: Simple URL-to-URL copy
- Experimental FTP servers
  - Striped FTP server (ala.DPSS): MPI-IO backend
  - Multi-threaded FTP server with parallel channels
  - Firewall FTP proxy: Securely and efficiently allow transfers through firewalls
  - Load balancing FTP proxy: Large data centers
- Experimental FTP clients
  - POSIX file interface

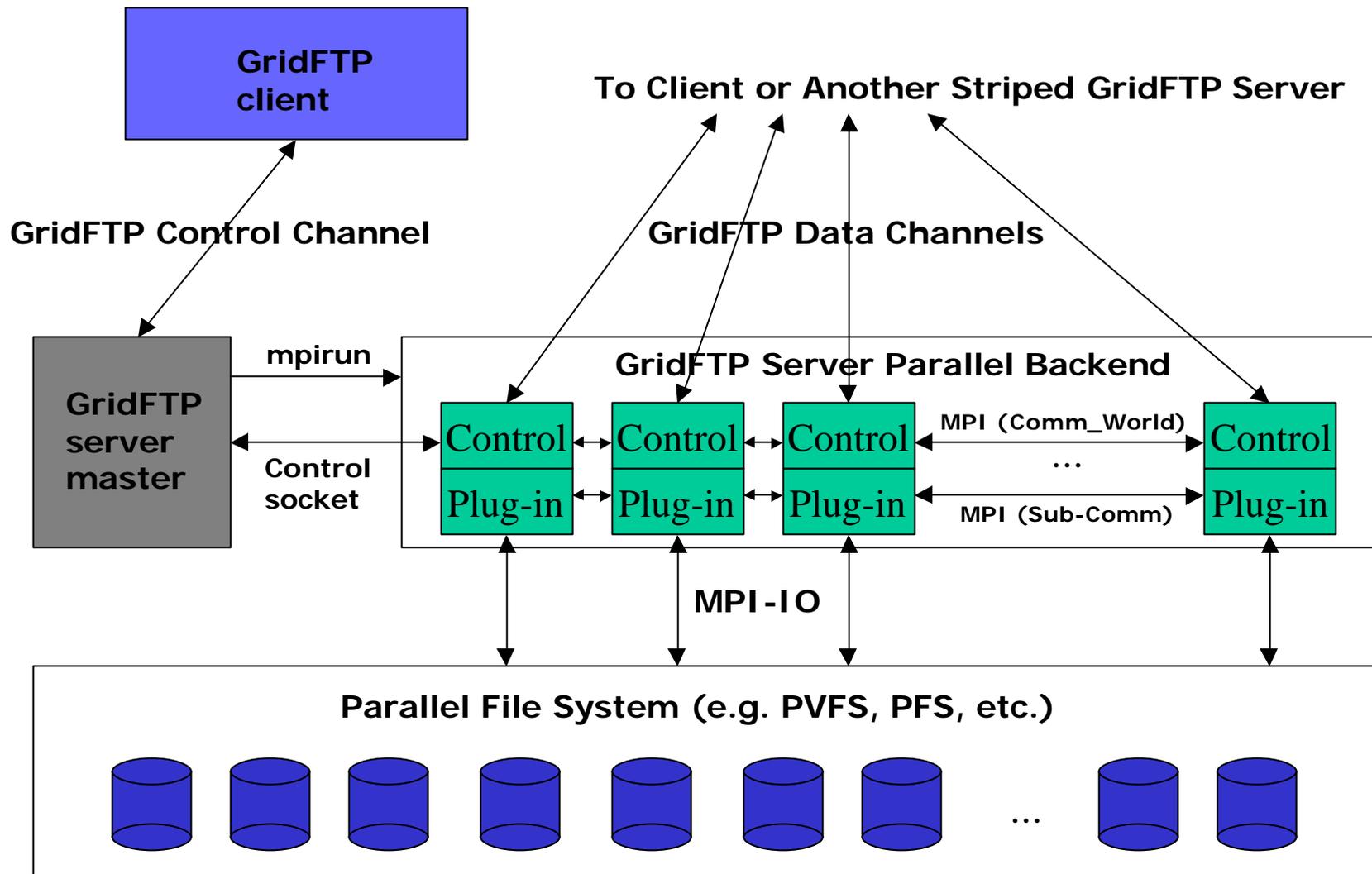
## globus\_ftp\_client Plug-ins

- globus\_ftp\_client is simple API/SDK:
  - get, put, 3<sup>rd</sup> party transfer, cd, mkdir, etc.
  - All data is to/from memory buffers
    - > Optimized to avoid any data copies
  - Plug-in interface
    - > Interface to one or more plug-ins:
      - Callouts for all interesting protocol events
      - Callins to restart a transfer
    - > Can support:
      - Monitor performance
      - Monitor for failure
      - Automatic retry: Customized for various approaches

# GridFTP at SC'2000: Long-Running Dallas-Chicago Transfer



# (Prototype) Striped GridFTP Server



## Striped GridFTP Plug-in Interface

- Given a RETR or STOR request:
  - Control calls plug-in to determine which nodes should participate in the request
  - Control creates an MPI sub-comm for nodes
  - Control calls plug-in to perform the transfer
    - > Includes request info, communicator, `globus_ftp_control_handle_t`
  - Plug-in does I/O to backend
    - > MPI-IO, PVFS, Unix I/O, Raw I/O, etc.
  - Plug-in uses `globus_ftp_control_data_*()` functions to send/receive data on GridFTP data channels

## Striped GridFTP Performance

- At SC'00, used first prototype:
  - Transfer between Dallas and LBNL
  - 8 node Linux clusters on each end
  - OC-48, 2.5Gb/s link (NTON)
  - Peaks over 1.5Gb/s
    - > Limited by disk bandwidth on end-points
  - 5 second peaks over 1Gb/s
  - Sustained 530Mb/s for 1 hr (238GB transfer)
    - > Had not yet implemented large files or data channel reuse.
    - > 2GB file took <20 seconds. New data channel sockets connected for each transfer.
    - > Explains difference between sustained and peak.

## A Word on GASS

- The Globus Toolkit provides services for file and executable staging and I/O redirection that work well with GRAM. This is known as Globus Access to Secondary Storage (GASS).
- GASS uses GSI-enabled HTTP as the protocol for data transfer, and a caching algorithm for copying data when necessary.
- The `globus_gass`, `globus_gass_transfer`, and `globus_gass_cache` APIs provide programmer access to these capabilities, which are already integrated with the GRAM job submission tools.

# Replica Management

- Maintain a mapping between logical names for files and collections and one or more physical locations
- Important for many applications
  - Example: CERN HLT data
    - > Multiple petabytes of data per year
    - > Copy of everything at CERN (Tier 0)
    - > Subsets at national centers (Tier 1)
    - > Smaller regional centers (Tier 2)
    - > Individual researchers will have copies

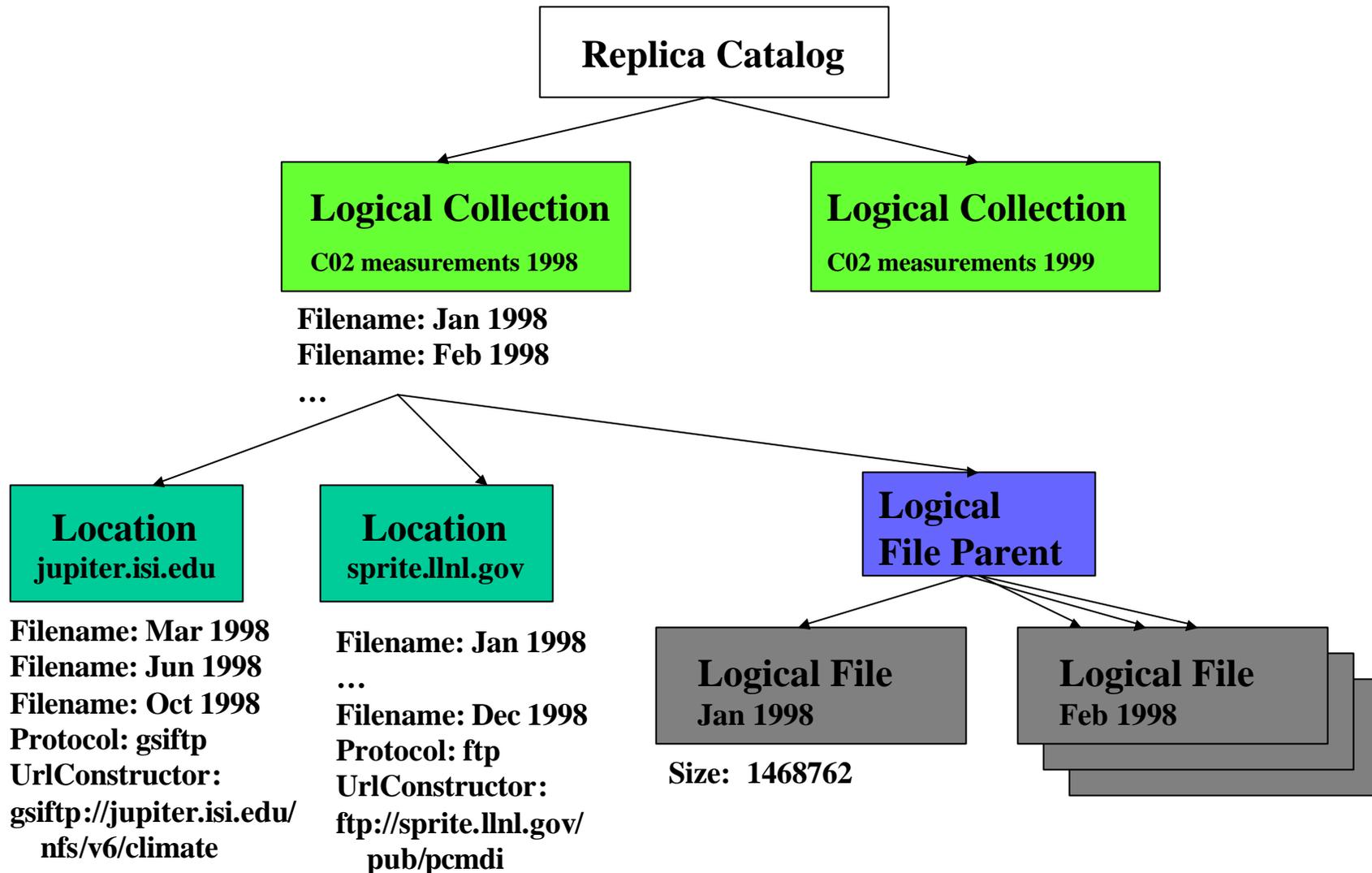
# Our Approach to Replica Management

- Identify replica cataloging and reliable replication as two fundamental services
  - Layer on other Grid services: GSI, transport, information service
  - Use LDAP as catalog format and protocol, for consistency
  - Use as a building block for other tools
- Advantage
  - These services can be used in a wide variety of situations

# Replica Manager Components

- Replica catalog definition
  - LDAP object classes for representing logical-to-physical mappings in an LDAP catalog
- Low-level replica catalog API
  - globus\_replica\_catalog library
  - Manipulates replica catalog: add, delete, etc.
- High-level reliable replication API
  - globus\_replica\_manager library
  - Combines calls to file transfer operations and calls to low-level API functions: create, destroy, etc.

# Replica Catalog Structure: A Climate Modeling Example



# Replica Catalog Services as Building Blocks: Examples

- Combine with information service to build replica selection services
  - E.g. “find best replica” using performance info from NWS and MDS
  - Use of LDAP as common protocol for info and replica services makes this easier
- Combine with application managers to build data distribution services
  - E.g., build new replicas in response to frequent accesses

## Relationship to Metadata Catalogs

- Metadata services describe data contents
  - Have defined a simple set of object classes
- Must support a variety of metadata catalogs
  - MCAT being one important example
  - Others include LDAP catalogs, HDF
- Community metadata catalogs
  - Agree on set of attributes
  - Produce names needed by replica catalog:
    - > **Logical collection name**
    - > **Logical file name**

## Replica Catalog Directions

- Many data grid applications do *not* require tight consistency semantics
  - At any given time, you may not be able to discover all copies
  - When a new copy is made, it may not be immediately recognized as available
- Allows for much more scalable design
  - Distributed catalogs: local catalogs which maintain their own LFN -> PFN mapping
  - Soft-state updates as basis for building various configurations of global catalogs

## Data Transfer APIs

- The `globus_ftp_control` API provides access to low-level GridFTP control and data channel operations.
- The `globus_ftp_client` API provides typical GridFTP client operations.
- The `globus_gass_copy` API provides the ability to start and manage multiple data transfers using GridFTP, HTTP, local file, and memory operations.
  - The `globus-url-copy` program is a thin wrapper around this API

## Replica Management APIs

- The `globus_replica_catalog` API provides basic Replica Catalog operations.
- The `globus_replica_management` API (under development) combines GridFTP and the Replica Catalog to manage replicated datasets.

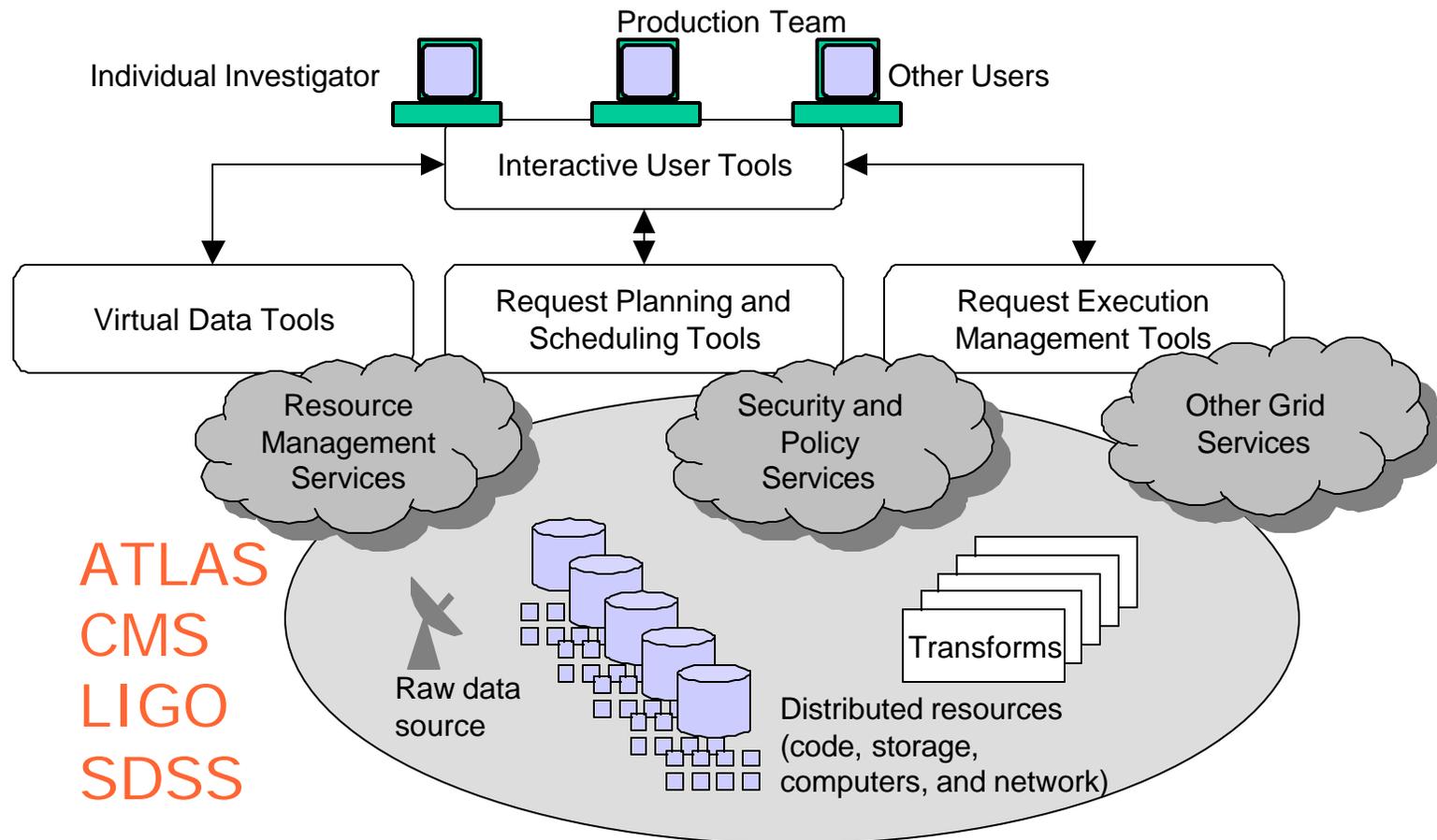
## Future Directions

- Continued enhancement & standardization of protocol
  - Globus Toolkit libraries provide reference implementation
- Continue building on libraries
  - Striped server w/ server side processing
  - Reliable replica/copy management service
  - Proxies for firewalls & load balancing
- Work with more application communities



# Grid Physics Network (GriPhyN)

*Enabling R&D for advanced data grid systems,  
focusing in particular on Virtual Data concept*



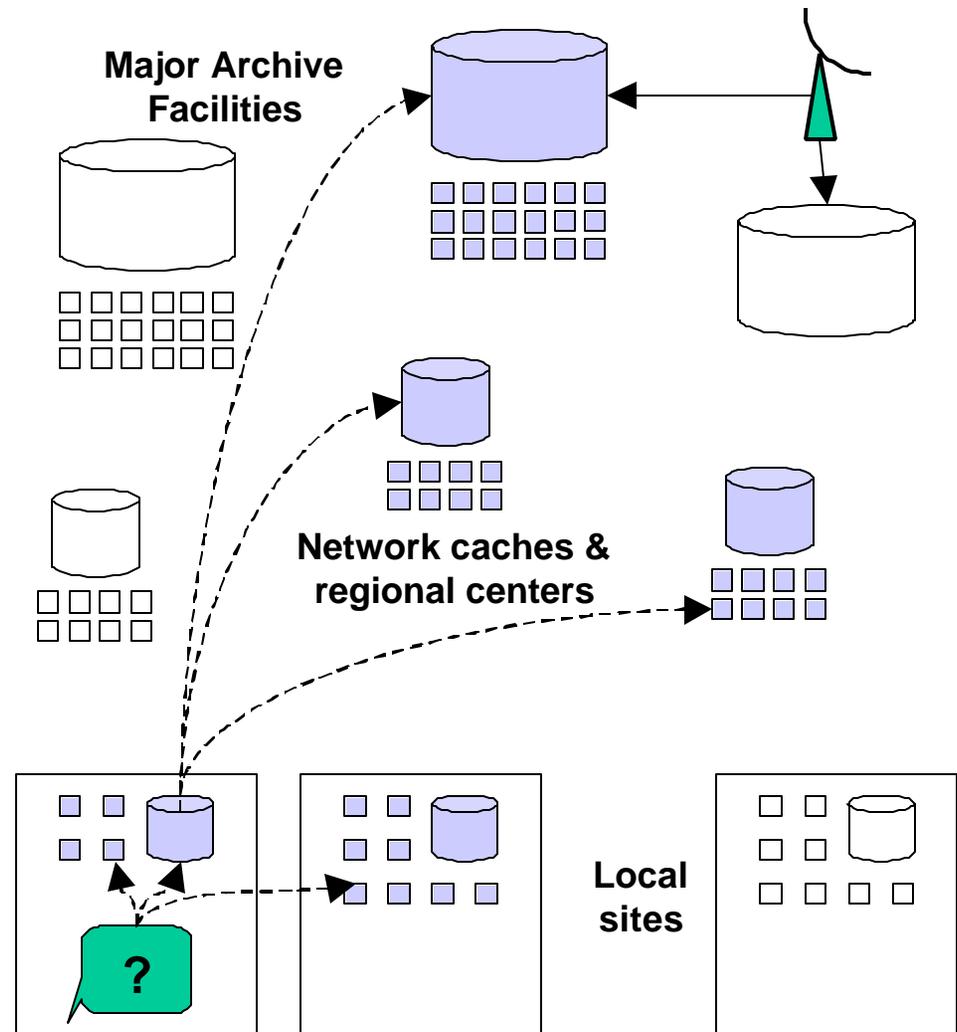
## The Virtual Data Concept

“ [a virtual data grid enables] the definition and delivery of a potentially unlimited virtual space of data products derived from other data. In this virtual space, requests can be satisfied via direct retrieval of materialized products and/or computation, with local and global resource management, policy, and security constraints determining the strategy used.”



# Virtual Data in Action

- Data request may
  - Access local data
  - Compute locally
  - Compute remotely
  - Access remote data
- Scheduling subject to local & global policies
- Local autonomy





# The Globus Toolkit™: Information Services

## Grid Information Services

- System information is critical to operation of the grid and construction of applications
  - What resources are available?
    - > Resource discovery
  - What is the “state” of the grid?
    - > Resource selection
  - How to optimize resource use
    - > Application configuration and adaptation?
- We need a general information infrastructure to answer these questions

## Examples of Useful Information

- Characteristics of a compute resource
  - IP address, software available, system administrator, networks connected to, OS version, load
- Characteristics of a network
  - Bandwidth and latency, protocols, logical topology
- Characteristics of the Globus infrastructure
  - Hosts, resource managers

## Grid Information: Facts of Life

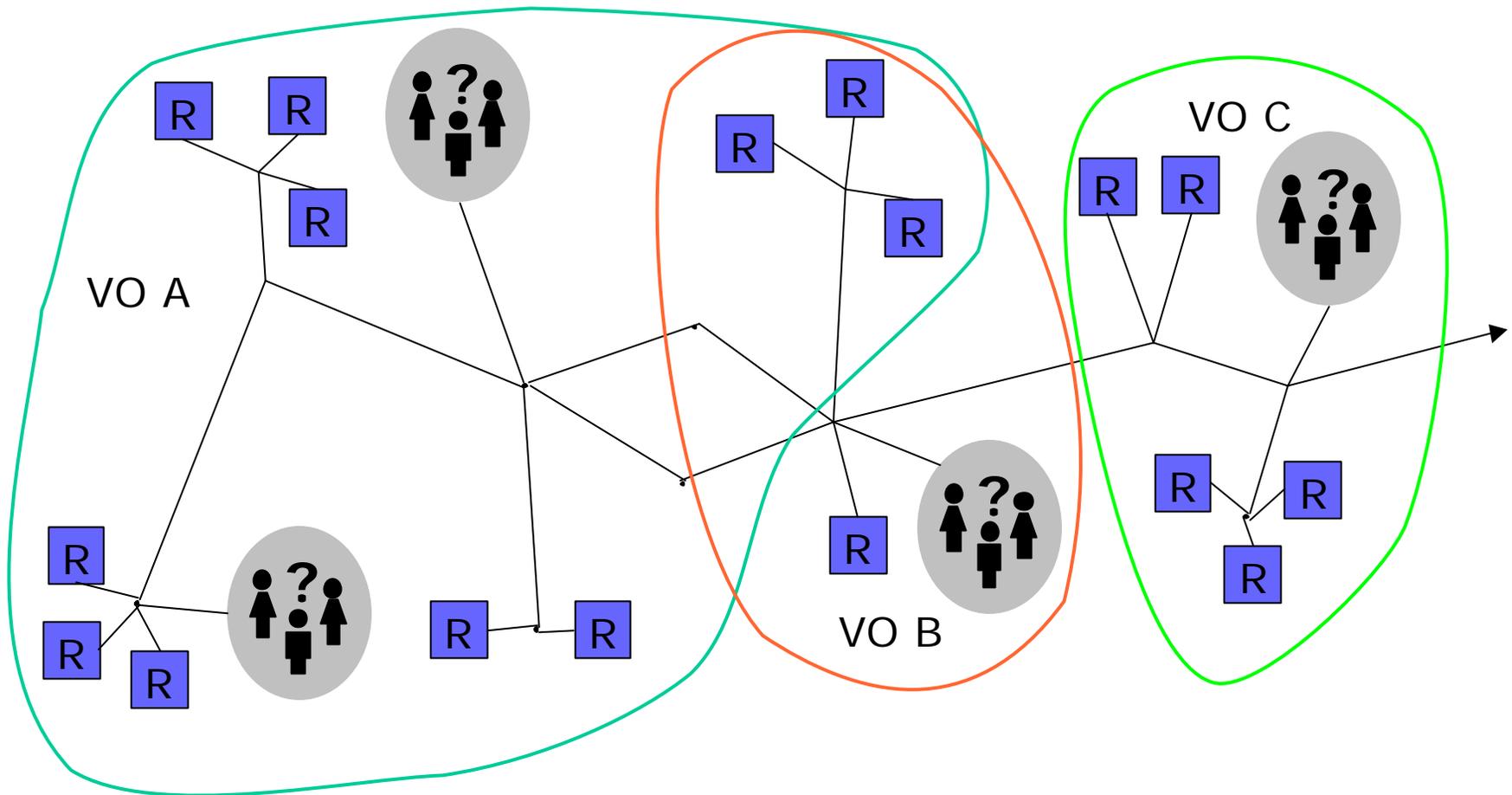
- Information is always old
  - Time of flight, changing system state
  - Need to provide quality metrics
- Distributed state hard to obtain
  - Complexity of global snapshot
- Component will fail
- Scalability and overhead
- Many different usage scenarios
  - Heterogeneous policy, different information organizations, different queries, etc.

## Grid Information Service

- Provide access to static and dynamic information regarding system components
- A basis for configuration and adaptation in heterogeneous, dynamic environments
- Requirements and characteristics
  - Uniform, flexible access to information
  - Scalable, efficient access to dynamic data
  - Access to multiple information sources
  - Decentralized maintenance



# The GIS Problem: Many Information Sources, Many Views



## What is a Virtual Organization?

- Facilitates the workflow of a group of users across multiple domains who share (some of) their resources to solve particular classes of problems
- Collates and presents information about these resources in a uniform view

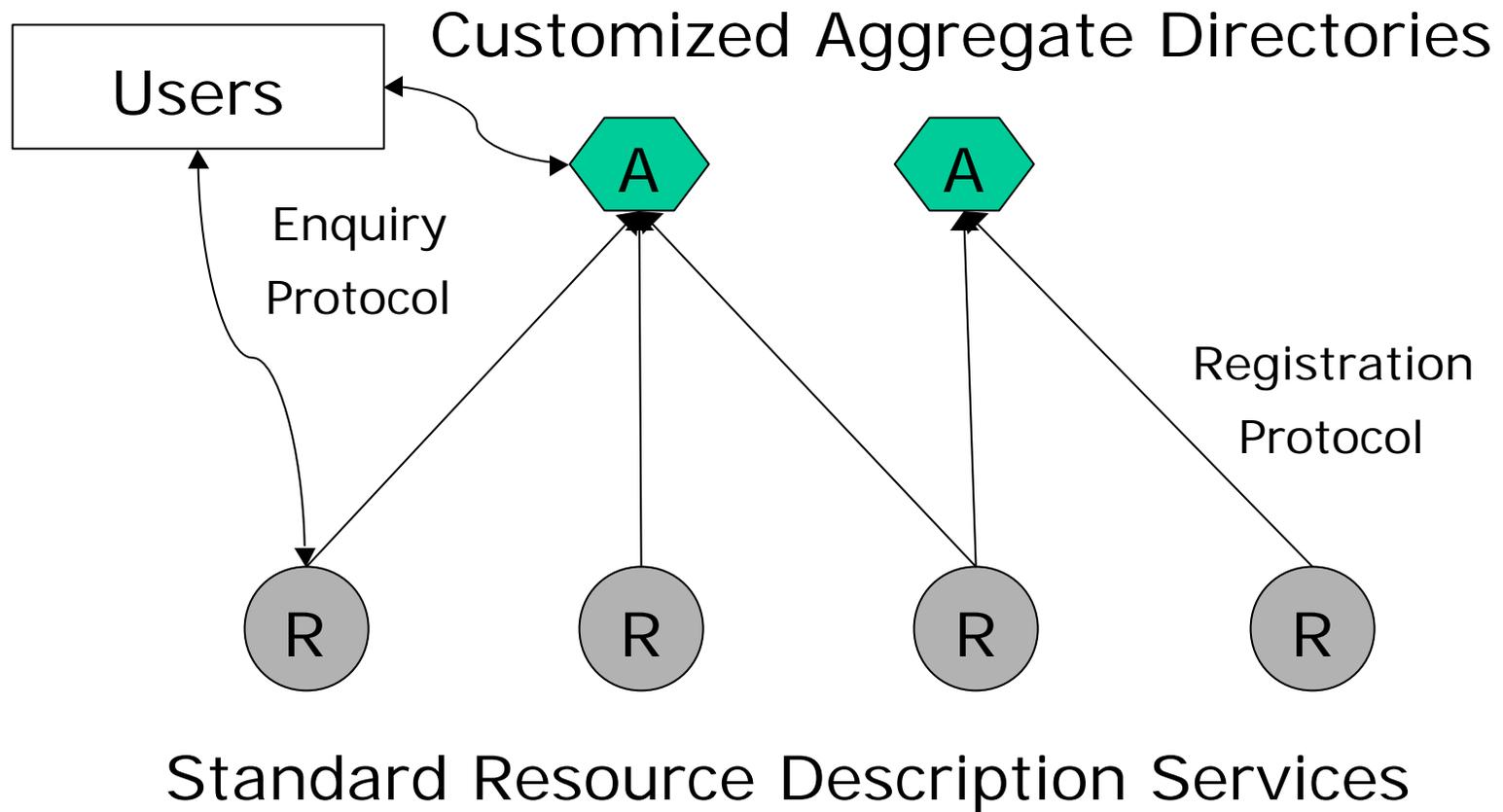
## Two Classes Of Information Servers

- **Resource Description Services**
  - Supplies information about a specific resource (e.g. Globus 1.1.3 GRIS).
- **Aggregate Directory Services**
  - Supplies collection of information which was gathered from multiple GRIS servers (e.g. Globus 1.1.3 GIIS).
  - Customized naming and indexing

# Information Protocols

- Grid Resource Registration Protocol
  - Support information/resource discovery
  - Designed to support machine/network failure
- Grid Resource Inquiry Protocol
  - Query resource description server for information
  - Query aggregate server for information
  - LDAP V3.0 in Globus 1.1.3

# GIS Architecture



# Metacomputing Directory Service

- Use LDAP as Inquiry
- Access information in a distributed directory
  - Directory represented by collection of LDAP servers
  - Each server optimized for particular function
- Directory can be updated by:
  - Information providers and tools
  - Applications (i.e., users)
  - Backend tools which generate info on demand
- Information dynamically available to tools and applications

## Two Classes Of MDS Servers

- Grid Resource Information Service (GRIS)
  - Supplies information about a specific resource
  - Configurable to support multiple information providers
  - LDAP as inquiry protocol
- Grid Index Information Service (GIIS)
  - Supplies collection of information which was gathered from multiple GRIS servers
  - Supports efficient queries against information which is spread across multiple GRIS server
  - LDAP as inquiry protocol

## LDAP Details

- Lightweight Directory Access Protocol
  - IETF Standard
  - Stripped down version of X.500 DAP protocol
  - Supports distributed storage/access (referrals)
  - Supports authentication and access control
- Defines:
  - Network protocol for accessing directory contents
  - Information model defining form of information
  - Namespace defining how information is referenced and organized

# MDS Components

- LDAP 3.0 Protocol Engine
  - Based on OpenLDAP with custom backend
  - Integrated caching
- Information providers
  - Delivers resource information to backend
- APIs for accessing & updating MDS contents
  - C, Java, PERL (LDAP API, JNDI)
- Various tools for manipulating MDS contents
  - Command line tools, Shell scripts & GUIs

# Grid Resource Information Service

- Server which runs on each resource
  - Given the resource DNS name, you can find the GRIS server (well known port = 2135)
- Provides resource specific information
  - Much of this information may be dynamic
    - > Load, process information, storage information, etc.
    - > GRIS gathers this information on demand
- “White pages” lookup of resource information
  - Ex: How much memory does machine have?
- “Yellow pages” lookup of resource options
  - Ex: Which queues on machine allows large jobs?

# Grid Index Information Service

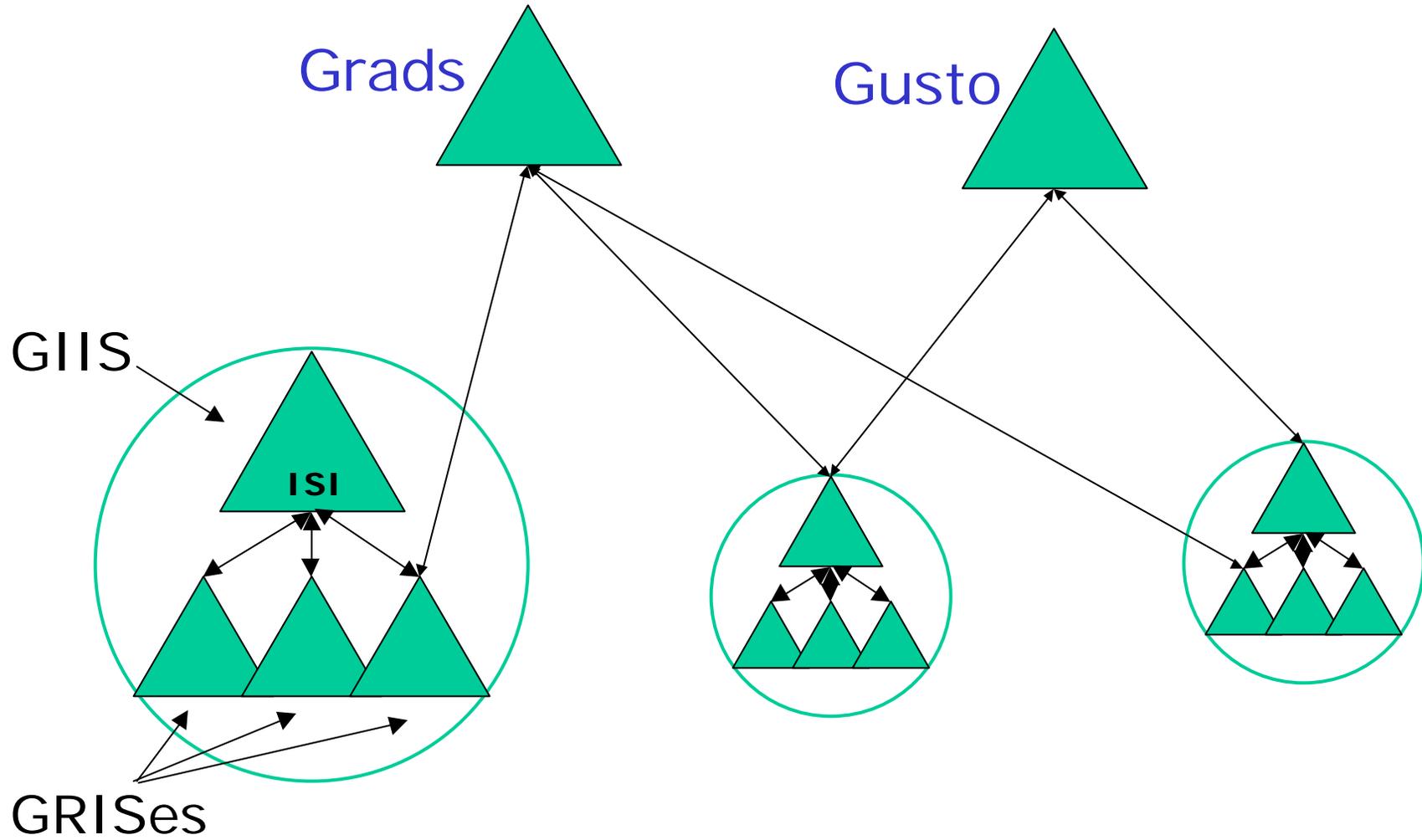
- **GIIS describes a class of servers**
  - Gathers information from multiple GRIS servers
  - Each GIIS is optimized for particular queries
    - > Ex1: Which Alliance machines are >16 process SGIs?
    - > Ex2: Which Alliance storage servers have >100Mbps bandwidth to host X?
  - Akin to web search engines
- **Organization GIIS**
  - The Globus Toolkit ships with one GIIS
  - Caches GRIS info with long update frequency
    - > Useful for queries across an organization that rely on relatively static information (Ex1 above)
- **Can be merged into GRIS**

# Finding a GRIS and Server Registration

- A GRIS or GIIS server can be configured to (de-) register itself during startup/shutdown
  - Targets specified in configuration file
- Softstate registration protocol
  - Good behavior in case of failure
- Allows for federations of information servers
  - E.g. Argonne GRIS can register with both Alliance and DOE GIIS servers



# Logical MDS Deployment



## MDS Commands

- LDAP defines a set of standard commands  
Idapsearch, etc.
- We also define MDS-specific commands
  - grid-info-search, grid-info-host-search
- APIs are defined for C, Java, etc.
  - C: OpenLDAP client API
    - > ldap\_search\_s(), ...
  - Java: JNDI

## Information Services API

- RFC 1823 defines an IETF draft standard client API for accessing LDAP databases
  - Connect to server
  - Pose query which returns data structures contains sets of object classes and attributes
  - Functions to walk these data structures
- Globus does not provide an LDAP API. We recommend the use of OpenLDAP, an open source implementation of RFC 1823.
- LDAP APIs available in other languages
  - E.g. Java JDNI, Perl, Python, etc.

## Searching an LDAP Directory

grid-info-search [options] filter [attributes]

- Default **grid-info-search** options

- |                          |                           |
|--------------------------|---------------------------|
| <b>-h</b> mds.globus.org | <i>MDS server</i>         |
| <b>-p</b> 389            | <i>MDS port</i>           |
| <b>-b</b> "o=Grid"       | <i>search start point</i> |
| <b>-T</b> 30             | <i>LDAP query timeout</i> |
| <b>-s</b> <b>sub</b>     | <i>scope = subtree</i>    |

*alternatives:*

**base** : *lookup this entry*

**one** : *lookup immediate children*

## Searching a GRIS Server

`grid-info-host-search` [options] filter [attributes]

- Exactly like `grid-info-search`, except defaults:

`-h localhost`

*GRIS server*

`-p 2135`

*GRIS port*

- Example:

`grid-info-host-search -h pitcairn "dn=*" dn`

# Filtering

- Filters allow selection of object based on relational operators (=, ~, <=, >=)
  - grid-info-search "cputype=\*"
- Compound filters can be construct with Boolean operations: (&, |, !)
  - grid-info-search "&(cputype=\*)(cpuload1 <= 1.0)"
  - grid-info-search "&(hn ~ =sdsc.edu)(latency <= 10)"
- Hints:
  - white space is significant <sup>required</sup>
  - use -L for LDIF format

## Example: Filtering

```
% grid-info-host-search -L "(objectclass=GlobusSoftware)"
```

```
dn: sw=Globus, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov,  
o=Grid  
objectclass: GlobusSoftware  
releasedate: 2000/04/11 19:48:29  
releasemajor: 1  
releaseminor: 1  
releasepatch: 3  
releasebeta: 11  
lastupdate: Sun Apr 30 19:28:19 GMT 2000  
objectname: sw=Globus, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl,  
dc=gov, o=Grid
```

## Example: Attribute Selection

```
% grid-info-host- search -L "(objectclass=*)" dn hn
```

- Returns the distinguished name (**dn**) and hostname (**hn**) of all objects

```
dn: sw=Globus, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid
```

```
dn: hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid  
hn: pitcairn.mcs.anl.gov
```

```
dn: service=jobmanager, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl, dc=gov, o=Grid  
hn: pitcairn.mcs.anl.gov
```

```
dn: queue=default, service=jobmanager, hn=pitcairn.mcs.anl.gov, dc=mcs, dc=anl,  
dc=gov, o=Grid
```

- Objects without hn fields are still listed
- DNs are always listed

## Example: Discovering CPU Load

- Retrieve CPU load fields of compute resources

```
% grid-info-search -L "(objectclass=GlobusComputeResource)" \  
dn cpuload1 cpuload5 cpuload15
```

```
dn: hn=lemon.mcs.anl.gov, ou=MCS, o=Argonne National Laboratory,  
o=Globus, c=US  
cpuload1: 0.48  
cpuload5: 0.20  
cpuload15: 0.03
```

```
dn: hn=tuva.mcs.anl.gov, ou=MCS, o=Argonne National Laboratory,  
o=Globus, c=US  
cpuload1: 3.11  
cpuload5: 2.64  
cpuload15: 2.57
```



# The Globus Toolkit™: Futures & Conclusions

## Problem Evolution

- Past-present:  $O(10^2)$  high-end systems; Mb/s networks; centralized (or entirely local) control
  - I-WAY (1995): 17 sites, week-long; 155 Mb/s
  - GUSTO (1998): 80 sites, long-term experiment
  - NASA IPG, NSF NTG:  $O(10)$  sites, production
- Present:  $O(10^4-10^6)$  data systems, computers; Gb/s networks; scaling, decentralized control
  - Scalable resource discovery; restricted delegation; community policy; Data Grid: 100s of sites,  $O(10^4)$  computers; complex policies
- Future:  $O(10^6-10^9)$  data, sensors, computers; Tb/s networks; highly flexible policy, control

# The Future: All Software is Network-Centric

- We don't build or buy "computers" anymore, we borrow or lease required resources
  - When I walk into a room, need to solve a problem, need to communicate
- A "computer" is a dynamically, often collaboratively constructed collection of processors, data sources, sensors, networks
  - Similar observations apply for software

## And Thus ...

- Reduced barriers to access mean that we do much more computing, and more interesting computing, than today => Many more components (& services); massive parallelism
- All resources are owned by others => Sharing (for fun or profit) is fundamental; trust, policy, negotiation, payment
- All computing is performed on unfamiliar systems => Dynamic behaviors, discovery, adaptivity, failure

## Summary

- The Grid problem: Resource sharing & coordinated problem solving in dynamic, multi-institutional virtual organizations
- Grid architecture emphasizes *systems problem*
  - Protocols & services, to facilitate interoperability and shared infrastructure services
- Globus Toolkit™: APIs, SDKs, and tools which implement Grid protocols & services
  - Provides basic software infrastructure for suite of tools addressing the *programming problem*