December 19, 2013

The XSEDE Global Federated File System (GFFS) - Breaking Down Barriers to Secure Resource Sharing

Andrew Grimshaw, University of Virginia

Co-architect XSEDE

XSEDE

Extreme Science and Engineering Discovery Environment

"The complexity of software is an essential property, not an accidental one. Hence, descriptions of a software entity that abstract away its complexity often abstract away its essence."

— Fred Brooks – *No Silver Bullet*

"Give me simple abstractions and make them work reliably"

— Kent Blackburn

"Perfection is achieved not when there is nothing more to add, but when there is nothing left to take away."

— Antoine de Saint-Exupery

Agenda

- XSEDE Architectural Background
- Globus Online
- X-WAVE/GFFS
  - Architectural themes
  - The Global Namespace
  - The Global Federated File System (GFFS)
  - *Execution Management Services**
- Demo
- Conclusion & Research Challenges

XSEDE Architectural Background

Distinguishing characteristics: Architecture

- XSEDE is *designed* for innovation and evolution
  - there ***is*** an architecture defined
    - based on set of design principles
    - rooted in the judicious use of standards and best practices
    - Integrated set of replaceable components designed to work together
- Professional systems engineering approach
  - responds to evolving needs of existing, emerging, and new communities
    - incremental development/deployment model
  - new requirements gathering processes
    - ticket mining, focus groups, usability panels, shoulder surfing
  - ensure robustness and security while incorporating new and improved technologies and services
  - process control, quality assurance, baseline management, stakeholder involvement

Two Approaches

- XUAS - Web/cloud - Globus
- X-WAVE/GFFS – Standards-based, integrated architecture
- See
  - Level 3 decomposition document
  - https://www.ideals.illinois.edu/handle/2142/45117
  - Or google search xsede level 3 architecture
- Use cases
  - https://software.xsede.org/registry-dev/index.php

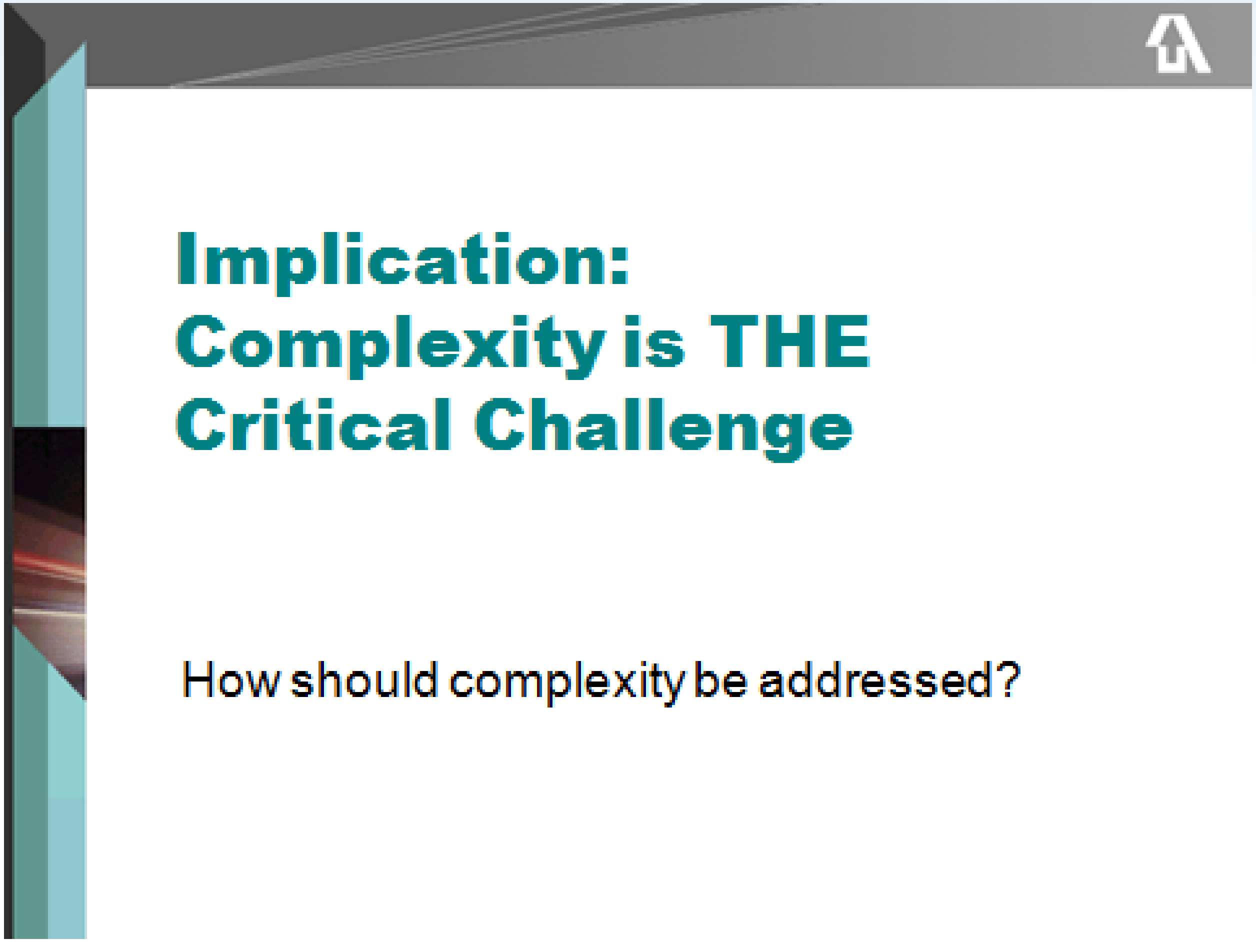
XSEDE

X-WAVE:
XSEDE Wide Area Virtual Environment

- Architectural themes
- The Global Namespace
- The Global Federated File System (GFFS)
- *Execution Management Services**

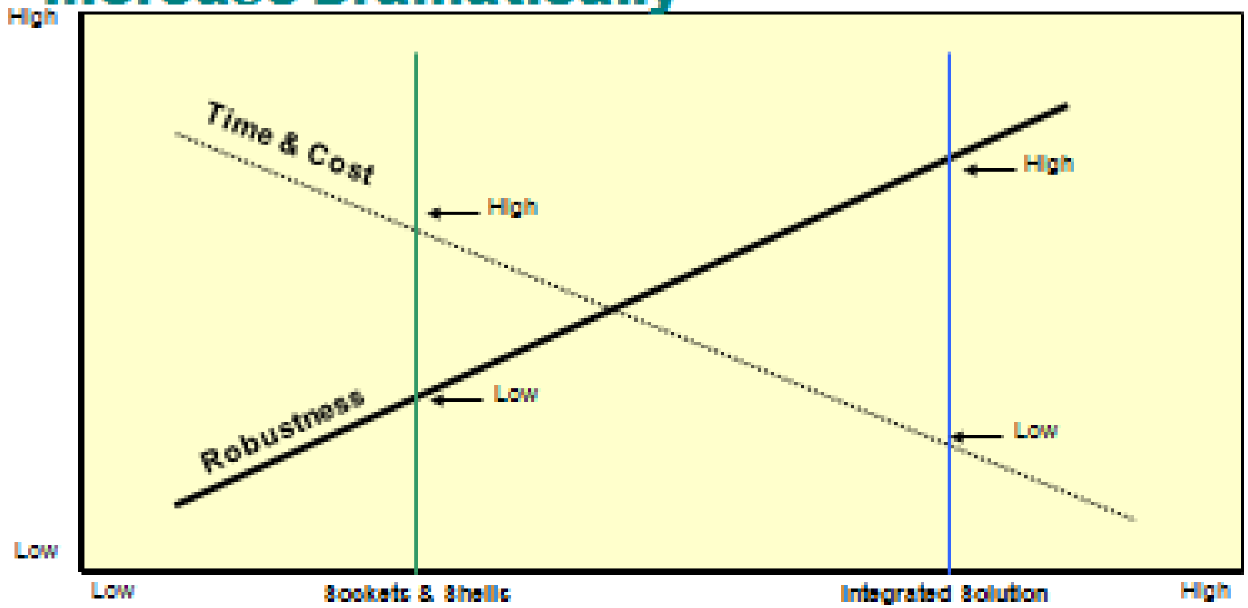
**If sufficient time*

An aside on distributed systems – or – What I've learned in the last 34 years



High-level versus low-level solutions

As Application Complexity Increases, Differences Between the Systems Increase Dramatically



Puzzle Ball

The Importance of Integration in a Grid Architecture

- If separate pieces are used, then the programmer must integrate the solutions.
- If all the pieces are not present, then the programmer must develop enough of the missing pieces to support the application.

Bottom Line: Both raise the bar by putting the cognitive burden on the programmer.

Back to architecture

What we mean by architecture

- Architecture defines the XSEDE system's interfaces and components and how they interact
  - each component is motivated by one or more requirements
  - each component is defined in terms of required capabilities: interfaces and qualities of service
- What is a system architecture?
  - Set of design principles
  - A definition of the basic interfaces/components
  - A definition of how the components refer to one another and interact in order to meet requirements
  - An abstraction on top of the underlying components

Principles

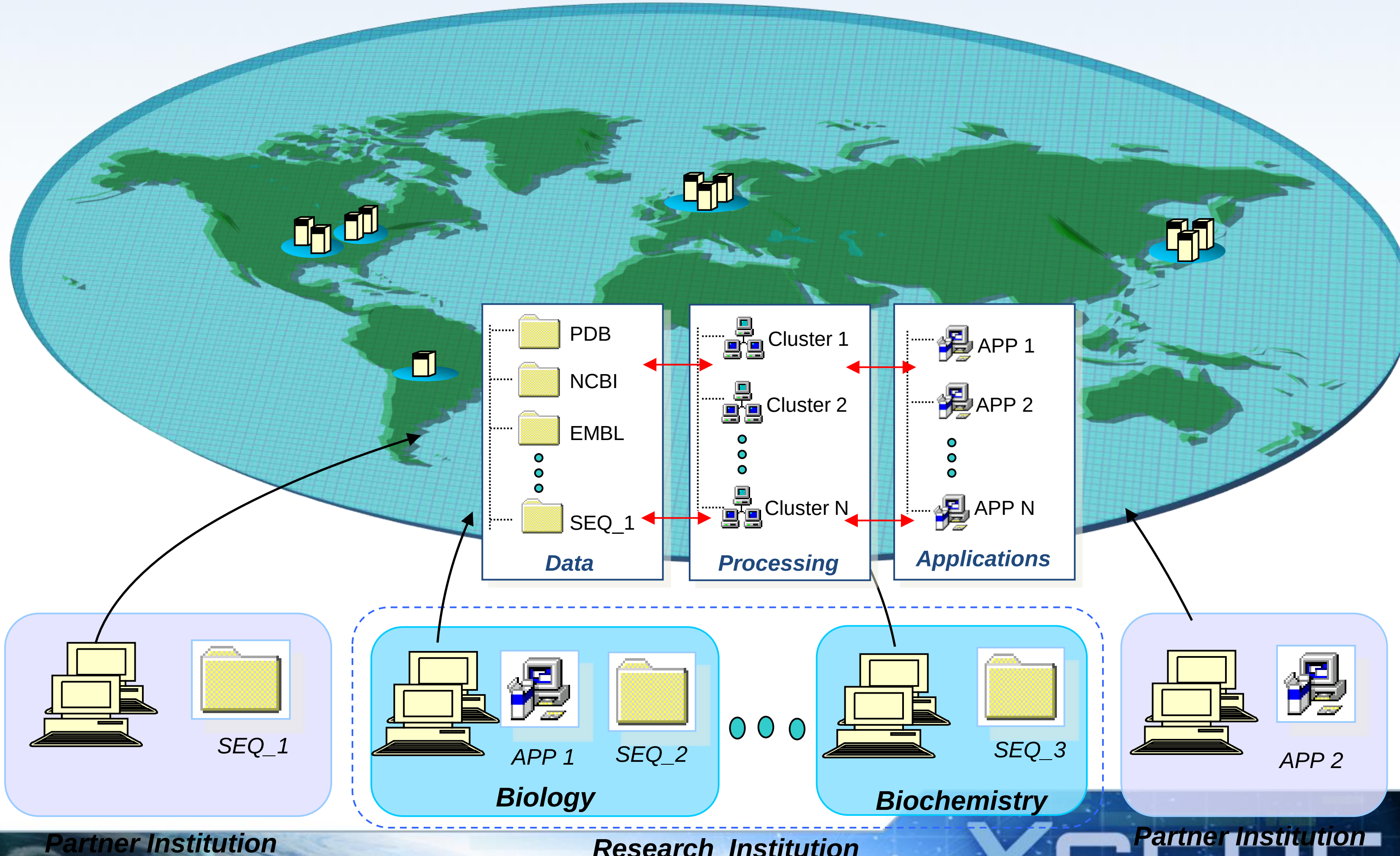
- Leverage familiar paradigms to simplify use
  - Pathnames, files and directories
  - Queues
  - Users/groups
  - Access control lists
- Interoperation between grid middleware islands
- Keep it simple
  - A small number of interfaces (types) that can be used in many ways
- Document everything
- Diversity of Implementation

X-WAVE/GFFS:

The global namespace

Inspired by *Plan 9*

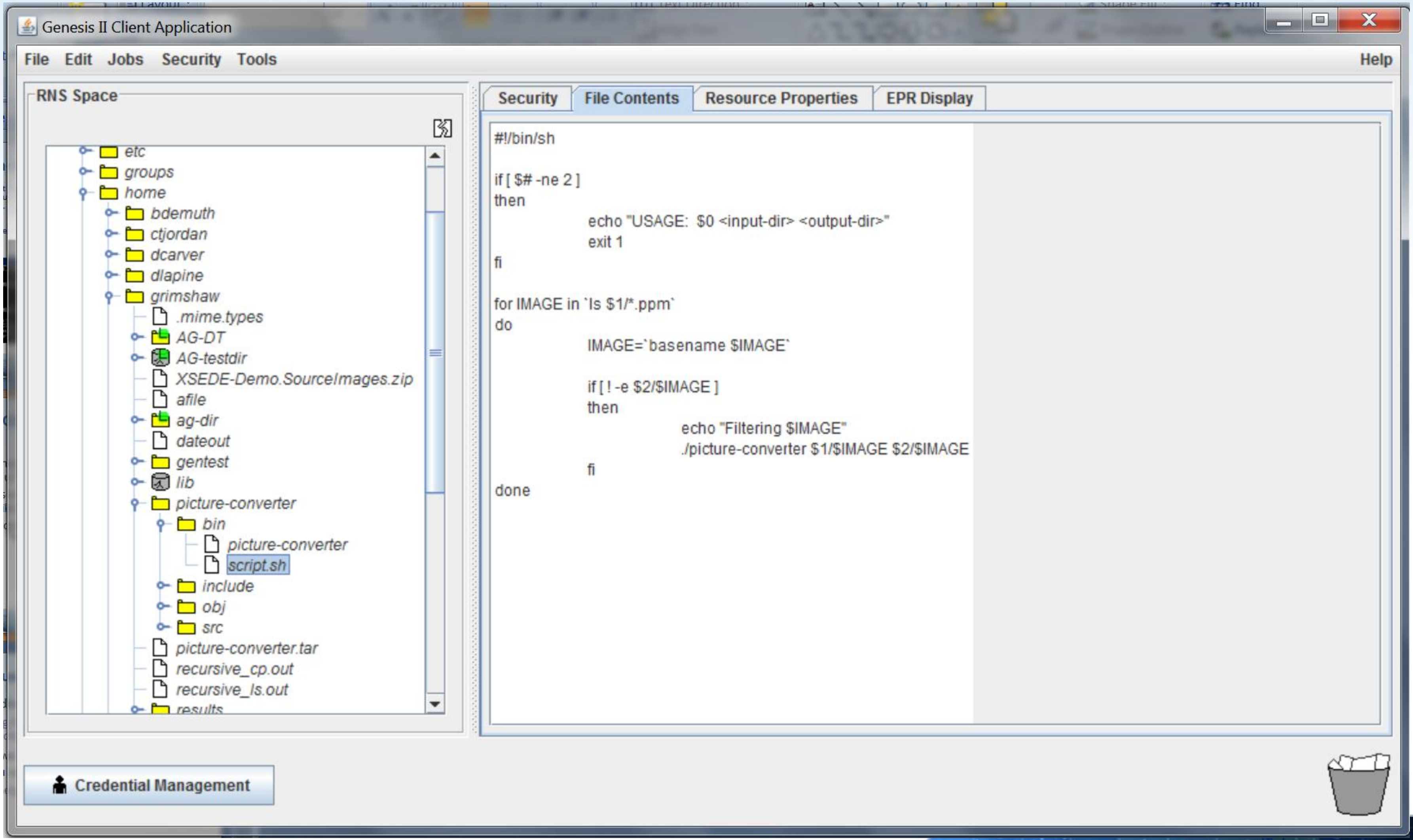
Basic idea: map resources into a global directory structure



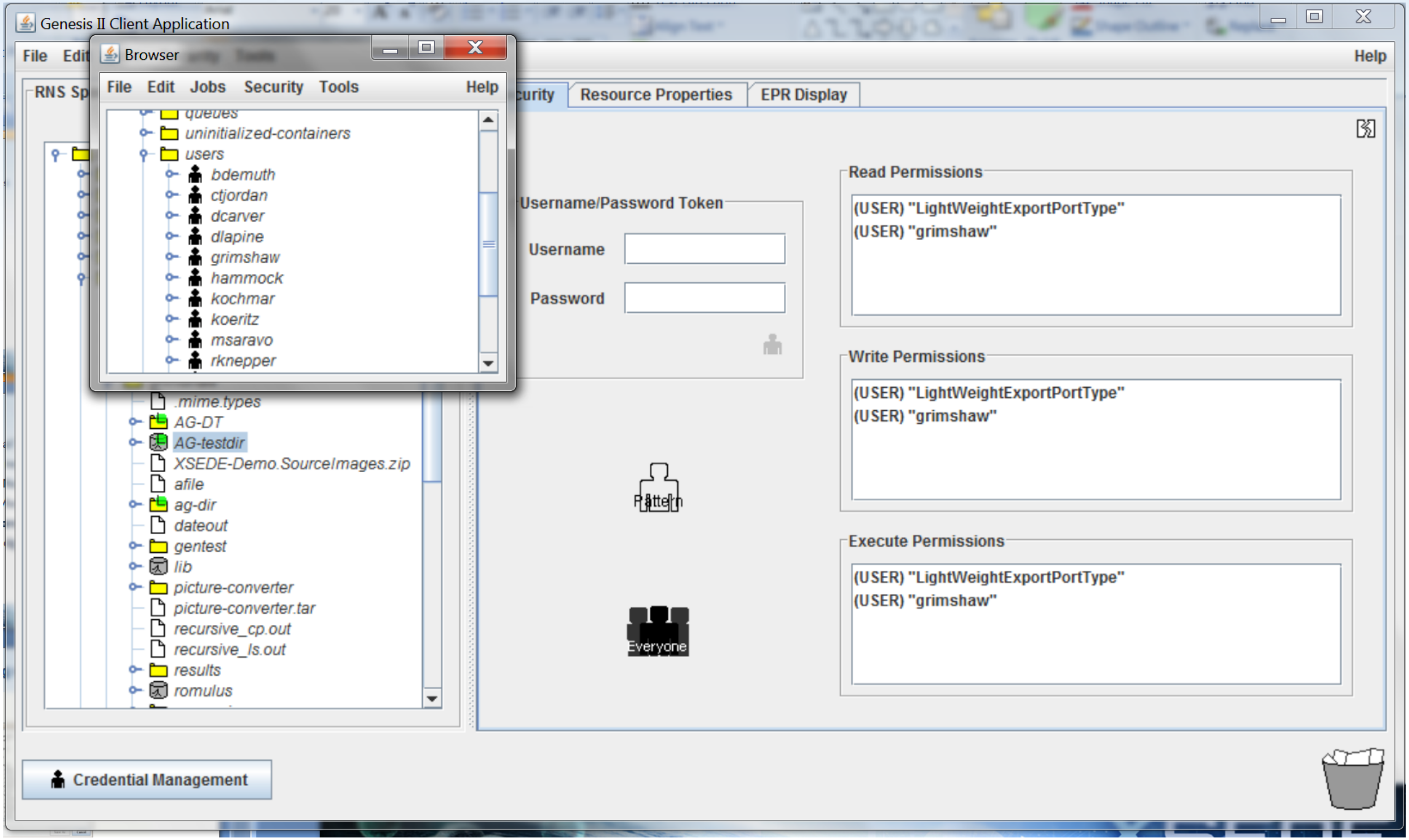
All kinds of resources

- Compute resources
  - PBS queue on Forge, SGE queue on Ranger, a PBS queue on your cluster
- Data Resources
  - Your home directory at NCSA, your home directory in your lab, and instrument in your lab, a relational database, the archive at PSC
- Identity Resources
  - The XSEDE Kerberos infrastructure, your Kerberos system, your LDAP, or create your own identities
- Scheduling resources
  - Meta schedulers, global job queues, build your own job queue that sends jobs to your cluster and your colleagues cluster
- Job resources
  - Jobs are resources, you can “ls” the jobs in a queue, you can “ls” the working directory of the job while it is running, as well as copy files in and out
- Groups/role resources
  - Create and manage your own groups

View of portion of the Global Namespace



Identity resources for authorization: Access Control Lists

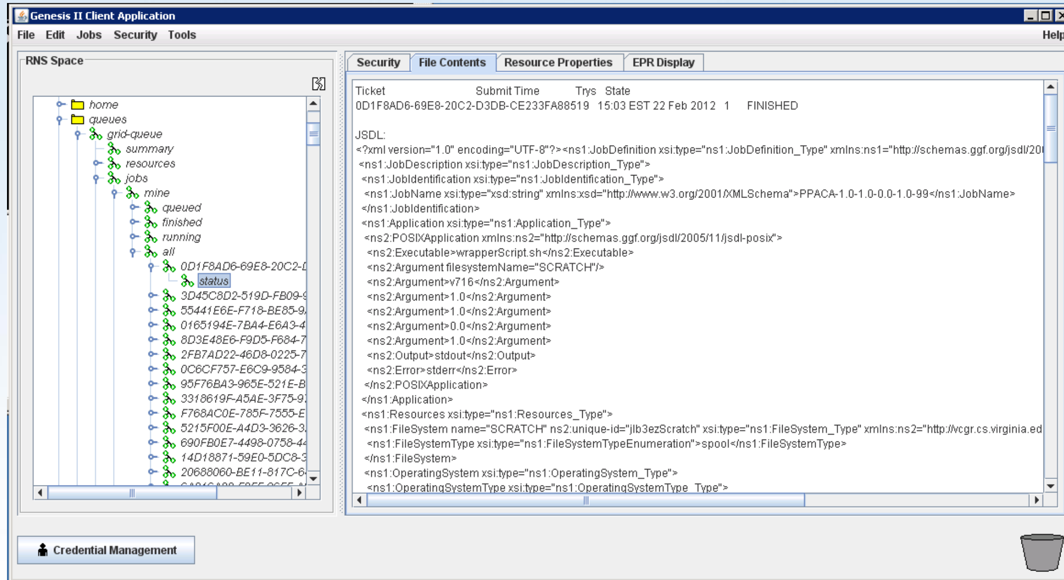


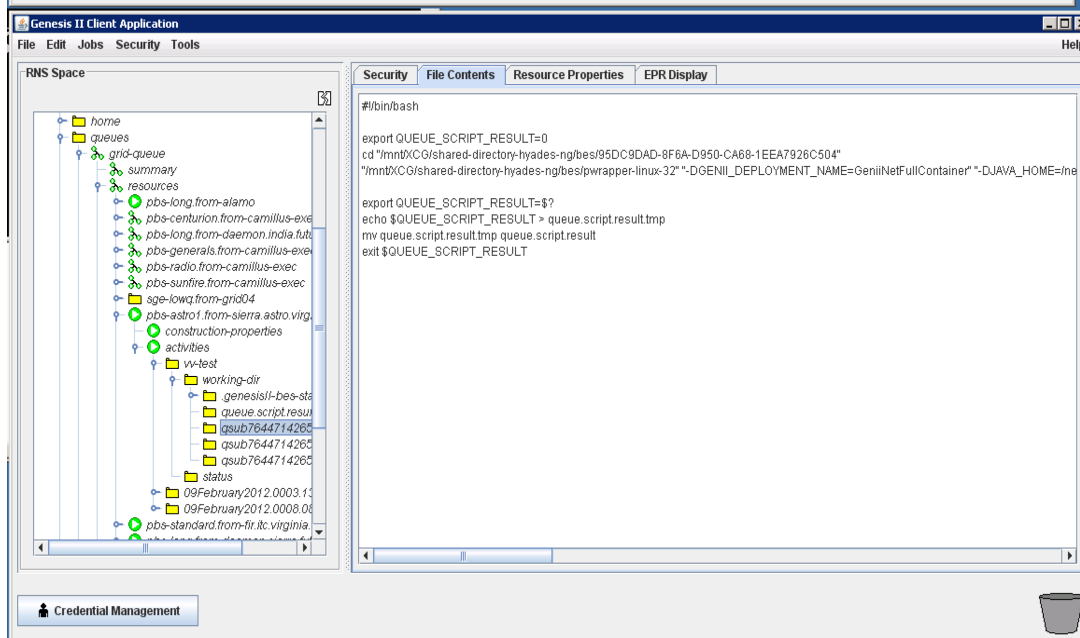
Compute resources too

/queue/grid-queue/jobs/mine/all/0D..status

/queues/grid-queue/resources/pbs-astro …/activities/W-test/working-dir

This is the directory of the running job – where ever it is





Then put a file system façade on top and you have the

Global Federated File System

Three Examples Illustrate GFFS Typical Uses Cases

- Accessing data at an NSF center from a home or campus
- Accessing data on a campus machine from an NSF center
- Directly sharing data with a collaborator at another institution

We'll come back to these later

XSEDE

GFFS – Basic Idea

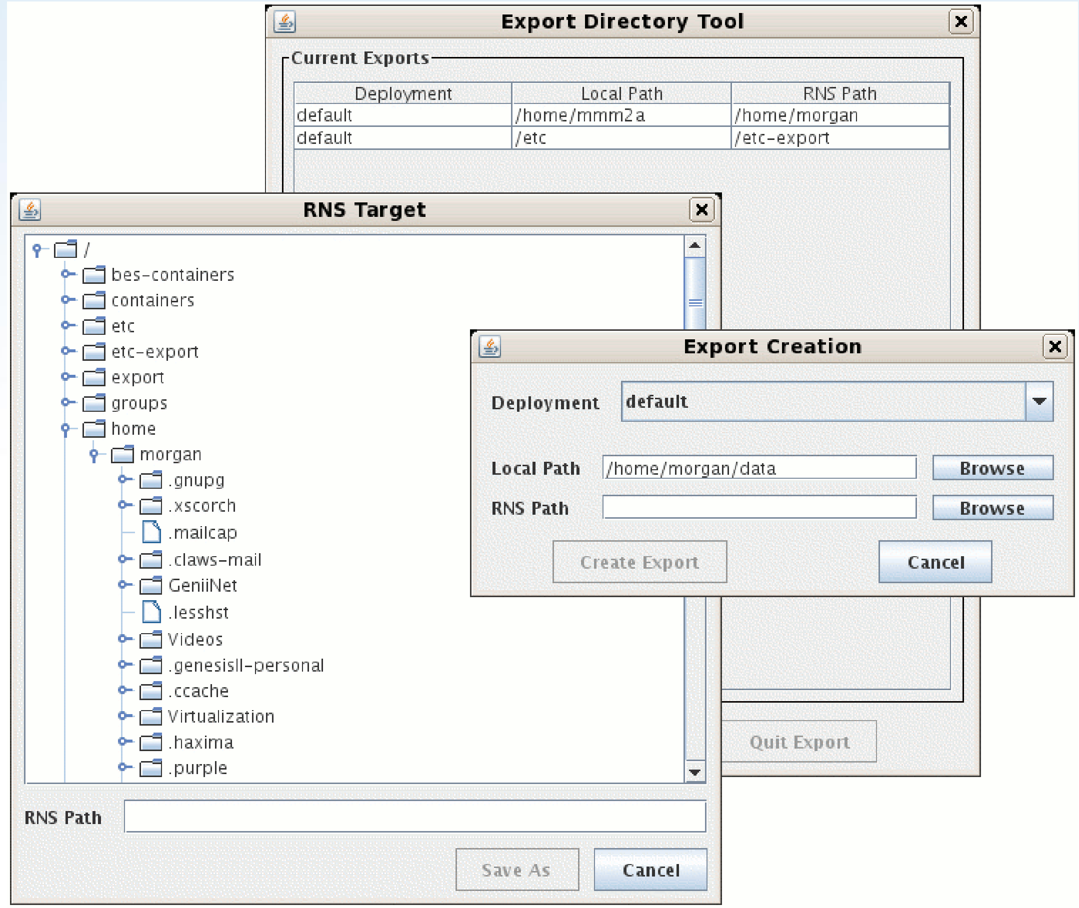
- Access the global namespace
  - Command line
  - Graphical User Interface
  - Map into local file system, “mount” XSEDE
- Put resources into the global namespace
  - Export directories
  - Clusters, supercomputers, cloud resources
  - Identities

Accessing the GFFS

- Via a file system mount
  - Global directory structure mapped directly into the local operating system via FUSE mount
- XSEDE resources regardless of location can be accessed via the file system
  - Files and directories can be accessed by programs and shell scripts as if they were local files
  - Jobs can be started by copying job descriptions into directories
  - One can see the jobs running or queued by doing an “ls”.
  - One can “cd” into a running job and access the working directory where the job is running directly

Putting resources into the GFFS

- Exporting directory trees
- Changes made in native file system visible to GFFS
- Changes made to files via GFFS propagated to native files



Shared storage as well

- The “rule” is – if you create a file or directory the storage used is in the same storage container as the parent directory
  - For an export this is obvious
- To place data on a remote storage service, mkdir (or use the GUI) and specify the target container. All data going into that directory will be stored on that container

Replication

- A directory tree of files and directories can be replicated on another storage container
  - Arbitrary k-replication – though there is a performance and storage cost
- Consistency is eventual consistency
- Interesting research question

  “How and when, and where should the system automatically make replica’s?”

XSEDE

Three Examples Illustrate Revisted

- Accessing data at an NSF center from a home or campus
  - Export directory at NSF center that you want to access
  - FUSE mount the XSEDE GFFS into your local file system
  - Create, Read, Update, and Delete files at the center from home
- Accessing data on a campus machine from an NSF center
  - Export directory on campus file server into the GFFS
  - FUSE mount the GFFS on the login node at the center, or specify state-in/stage out in a job description
  - Create, Read, Update, and Delete files at home from the center
- Directly sharing data with a collaborator at another institution
  - Export directory on campus file server into the GFFS
  - Give your collaborator desired level of access (RWX)
  - Collaborator FUSE mounts the GFFS their desktop
  - Share files.

Switch to brief demo

XSEDE

Extreme Science and Engineering Discovery Environment

Conclusion

- The XSEDE X-WAVE architecture goal is to accelerate science by lowering the barriers to collaboration
  - Usability by leveraging known user interactions
  - Integration of diverse resource into a shared namespace
  - User control of access to their resources – whether they be data, compute, or applications
- The GFFS allows users to securely share and easily access data regardless of location
  - A laboratory instrument
  - An XSEDE file system
  - Storage services
  - The session directory of a running job

Research challenges

- Performance, Performance, Performance
- Location, Location, Location
- The trade-off between performance, availability, cost, easy of use, security
- Leveraging commercial spaces with pay as you go infrastructure
  - Must have a way to "charge" for different qualities of service
  - Grid economies?
- The sociology of centers – how to overcome institutional inertia?


Our reach will forever
exceed our grasp, but,
in stretching our horizon,
we forever improve our world.
XSEDE
Extreme Science and Engineering
Discovery Environment

X-WAVE:
Execution Management Services

What are Jobs in XSEDE?

- A **job** is a unit of work that executes a program
  - Really pretty generic: much like PBS or LSF job
  - Program may be sequential, threaded, hybrid GPGPU program, or traditional parallel using MPI or OpenMP
  - Programs can be command line programs or shell scripts that take zero or more parameters
- Jobs MAY specify *files* to be staged in before execution and out after execution
  - This MAY include executables and libraries
- Jobs MAY specify *file systems* to mount, e.g., SCRATCH or GFFS (Global Federated File System)
- Jobs MAY specify resource *requirements* such as operating system, amount of memory, number of CPU's, or other matching criteria
- Jobs MAY be *parameter sweep* jobs with arbitrary number of dimensions

BESes: Basic Execution Services

- BESes run jobs on particular compute resources
  - Manage *data staging* for jobs
  - Monitor job *progress/completion*
  - Maintains *job state*
- “Compute resources” may be workstations, clusters, or supercomputers
- Each BES has a set of resource properties such as operating system, memory, number of cores, etc. that can be used to match jobs to BESes for execution

Grid Queues

- Work much like any other queuing system
- Grid users submit jobs to grid queue
- Maintain:
  - List of (BES) compute resources available for scheduling
  - Description of capabilities of each compute resource
  - List of jobs and statuses
- Match jobs to available compute resources
  - Ask matching resources to run jobs
- Monitor job progress/completion
- Cmd-line and GUI tools to manage jobs in queue
  - qsub, qstat, qkill, qcomplete, queue manager