the globus alliance
www.globus.org

GRAM, RFT & Job Submission, Execution Management for GT4 Developers

Stuart Martin

Session Overview

Q: What is this session about?

A: This presentation will cover the features, interface, architecture, performance, and future plans of the Globus Toolkit v4 Web Services Grid Resource Allocation and Management (WS GRAM) component.

- Not repeating Resource Management session material!

- Four-part discussion (~ 20 mins/each)
  - Overview of GRAM Model
  - How to use client software
  - How to administer servers
  - gridwise - How to implement a new scheduler adapter

GRAM: Part 1

Overview of GRAM Model…

What is GRAM?

- GRAM is a Globus Toolkit component
  - For Grid *job management*
  - Part of our Resource Management strategy
- GRAM is for stateful job control
  - Reliable operation
  - Asynchronous monitoring and control
  - Remote credential management
  - File staging by controlling FTP
- GRAM implements a protocol
  - For communicating with schedulers

Grid Job Management Goals

Provide a service to securely:

- Create an environment for a job
- Stage files to/from environment
- Cause execution of job process(es)
  - Via various local schedulers
- Monitor execution
- Signal important state changes to client
- Enable client access to output files
  - Streaming access during execution

Job Submission Model

- Create and manage one job on a resource
- Submit and wait
- Not with an interactive TTY
  - File based stdin/out/err
  - Supported by all batch schedulers
- More complex than RPC
  - Optional steps before and after submission message
  - Job has complex lifecycle
    - Staging, execution, and cleanup states
    - But not as general as Condor DAG, etc.
  - Asynchronous monitoring

Job Submission Options

- Optional file staging
  - Transfer files "in" before job execution
  - Transfer files "out" after job execution
- Optional file streaming
  - Monitor files during job execution
- Optional credential delegation
  - Create, refresh, and terminate delegations
  - For use by job process
  - For use by GRAM to do optional file staging

Job Submission Monitoring

- Monitor job lifecycle
  - GRAM and scheduler states for job
    - StageIn, Pending, Active, Suspended, StageOut, Cleanup, Done, Failed
  - Job execution status
    - Return codes
- Multiple monitoring methods
  - Simple query for current state
  - Asynchronous notifications of client

Secure Submission Model

- Secure submit protocol
  - PKI authentication
  - Authorization and mapping
  - Further authorization by scheduler
- Secure control/cancel
  - Also PKI authenticated
  - Owner has rights to his jobs and not others’

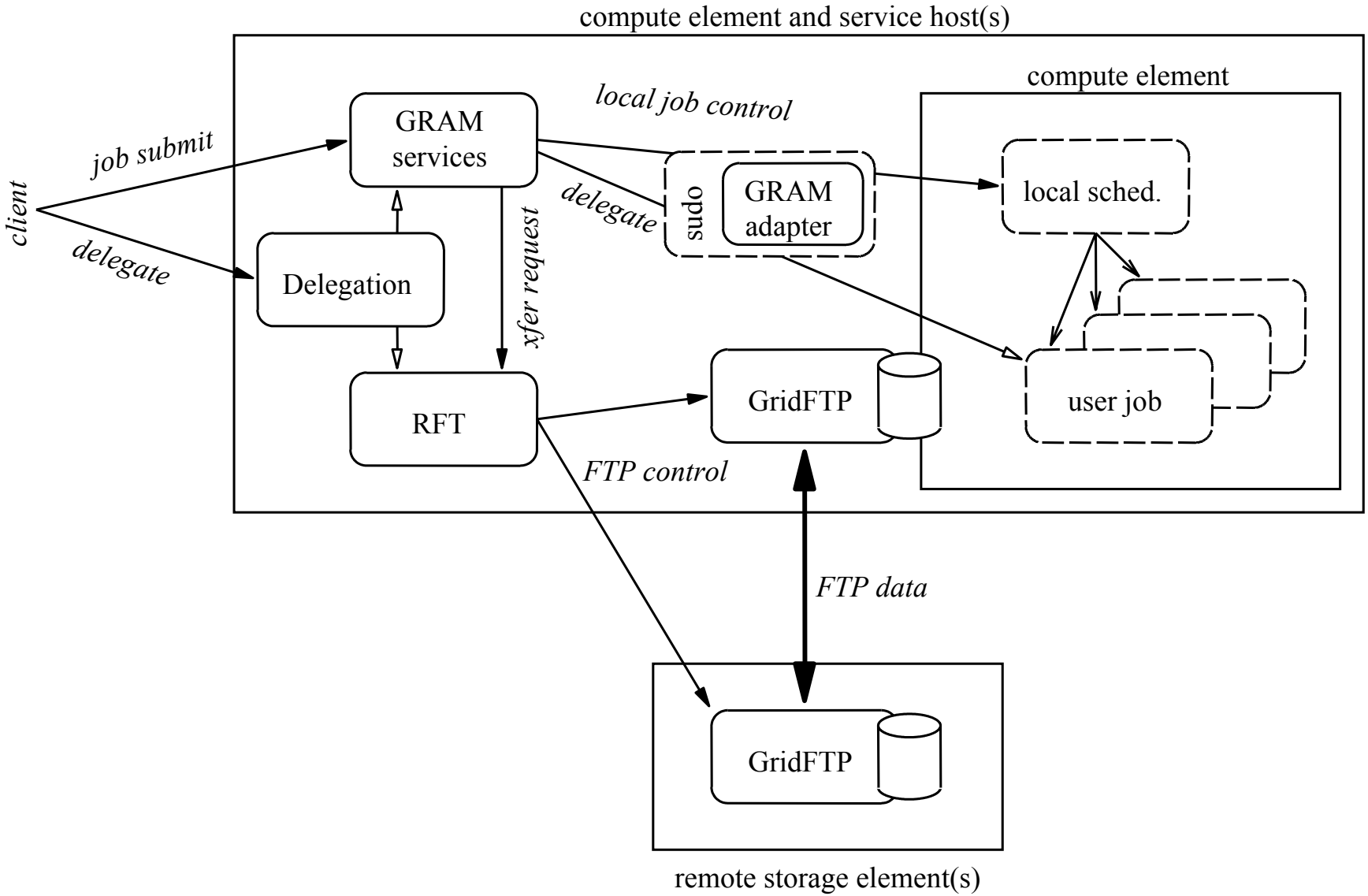
Secure Execution Model

- After authorization…
- Execute job securely
  - User account "sandboxing" of processes
    - According to mapping policy and request details
  - Initialization of sandbox credentials
    - Kerberos
    - AFS
    - Client-delegated credentials
  - Multiple levels of audit possible

Secure Staging Model

- Before and after sandboxed execution…
- Perform secure file transfers
  - Create RFT request
    - To local or remote RFT service
    - PKI authentication and delegation
    - In turn, RFT controls GridFTP
      - Using delegated client credentials
  - GridFTP
    - PKI authentication
    - Authorization and mapping by local policy files
    - further authorization by FTP/unix perms

WS-GRAM Approach



Other Approach Highlights

- Scalability improvements

  (discussed next)
- sudo/auth_and_exec
  - to limit damage risk from software failures
  - to improve audit capabilities
- Extensibility
  - Retain: scheduler adapter structure
    - To extend for new platforms
  - Improved: authorization callouts
    - To better integrate with site practices

Usage Scenarios: the Ideal

"GRAM should add little to no overhead compared to an underlying batch system"

- Submit as many jobs to GRAM as is possible to the underlying scheduler
  - Goal - 10,000 jobs to a batch scheduler
  - Goal – efficiently fill the process table for fork scheduler
- Submit/process jobs as fast to GRAM as is possible to the underlying scheduler
  - Goal - 1 per second

- We are not there yet…
  - A range of limiting factors at play

Usage Scenarios: the Attempt

- Efforts and features towards the goal
  - Allow job brokers the freedom to optimize
    - E.g. Condor-G is smarter than globusrun
    - Protocol steps made optional and shareable
  - Reduced cost for GRAM service on host
    - Single WSRF host environment
    - Better job status monitoring mechanisms
  - More scalable/reliable file handling
    - GridFTP and RFT instead of globus-url-copy
    - Removal of non-scalable GASS caching
- GT4 tests performing better than GT3 did
  - But more work to do

GRAM 3.9.4 performance

- Service performance & stability
  - Throughput
    - GRAM can process ~70 /bin/date jobs per minute
  - Job burst
    - Many simultaneous job submissions
    - Are the error conditions acceptable?
  - Max concurrency
    - Total jobs a GRAM service can manage at one time without failure?
  - Service uptime
    - Under a moderate load, how long can the GRAM service process jobs without failure / reboot?

Reasonable Applications Today

- High throughput job sets: two approaches
  1. Use GRAM for every task
     - Walk the edge of GRAM scalability
  2. Use GRAM for provisioning "slaves"
     - Course-grain jobs handle task/transaction flow
     - As in Condor glide-ins
- Large-scale jobs w/ MPICH-G4
  - Co-allocation but no co-reservation yet
- Special jobs
  - DIY discovery/control extensions

GRAM: Part 2

How to use client software…

How to use Client Software

- Command line programs
- WSDL interface

Command Line Programs

- globusrun-ws
  - Submit and monitor gram jobs
- grid-proxy-init
  - Creates client-side user proxy
- wsrf-query
  - Query a services resource properties
- globus-url-copy
  - Transfer files to remote hosts
- globus-credential-delegate
- globus-credential-refresh
  - Credential management to remote hosts

globusrun-ws

- New in GT 3.9.4
  - Replaces managed-job-globusrun (java)
- Written in C (C WS Core)
  - Faster startup and execution
- Supports GRAM multi-jobs or single jobs
  - Submission, monitoring, cancellation
- Credential management
  - Automatic or user-supplied delegation
- Streaming of job stdout/err during execution
  - Advanced use of GridFTP client library

Simple Job: Step 1

- Create a user proxy
  - Your temporary grid credential
- Command Example:

```
% grid-proxy-init
Your identity:
/DC=org/DC=doegrids/OU=People/CN=Stuart Martin 564728
Enter GRID pass phrase for this identity:
Creating proxy.......................... Done
Your proxy is valid until: Fri Jan  7 21:35:31 2005
```

Simple Job: Step 2

- Submit job to a GRAM service
  - default factory EPR
  - generate job RSL to default localhost
- Command example:

```
% globusrun-ws -submit -c /bin/touch touched_it
Submitting job...Done.
Job ID: uuid:002a6ab8-6036-11d9-bae6-
0002a5ad41e5
Termination time: 01/07/2005 22:55 GMT
Current job state: Active
Current job state: CleanUp
Current job state: Done
Destroying job...Done.
```

Complete factory contact

- Override default EPR
  - Select a different host/service
  - Use "contact" shorthand for convenience
    - Relies on proprietary knowledge of EPR format!
- Command example:

```
% globusrun-ws -submit -F \
https://140.221.65.193:4444/wsrf/services\
/ManagedJobFactoryService \
-c /bin/touch touched_it
```

Read RSL from file

- Command:

```
% globusrun-ws -submit -f touch.xml
```

- Contents of touch.xml file:

```
<job>
  <executable>/bin/touch</executable>
  <argument>touched_it</argument>
</job>
```

Batch Job Submissions

```
% globusrun-ws -submit -batch -o job_epr -c /bin/sleep 50
  Submitting job...Done.
  Job ID: uuid:f9544174-60c5-11d9-97e3-0002a5ad41e5
  Termination time: 01/08/2005 16:05 GMT

% globusrun-ws -monitor -j job_epr
  job state: Active
  Current job state: CleanUp
  Current job state: Done
  Requesting original job description...Done.
  Destroying job...Done.
```

Batch Job Submissions

```
% globusrun-ws -submit -batch -o job_epr -c /bin/sleep 50
  Submitting job...Done.
  Job ID: uuid:f9544174-60c5-11d9-97e3-0002a5ad41e5
  Termination time: 01/08/2005 16:05 GMT

% globusrun-ws -status -j job_epr
  Current job state: Active

% globusrun-ws -status -j job_epr
  Current job state: Done

% globusrun-ws -kill -j job_epr
  Requesting original job description...Done.
  Destroying job...Done.
```

Common/useful options

- globusrun-ws -J
  - Perform delegation as necessary for job
- globusrun-ws -S
  - Perform delegation as necessary for job's file staging
- globusrun-ws -s
  - Stream stdout/err during job execution to the terminal
- globusrun-ws -self
  - Useful for testing, when you have started the service using your credentials instead of host credentials

Staging job

```
<job>
  <executable>/bin/echo</executable>
  <directory>/tmp</directory>
  <argument>Hello</argument>
  <stdout>job.out</stdout>
  <stderr>job.err</stderr>
  <fileStageOut>
    <transfer>
      <sourceUrl>file:///tmp/job.out</sourceUrl>
      <destinationUrl>
         gsiftp://host.domain:2811/tmp/stage.out
      </destinationUrl>
    </transfer>
  </fileStageOut>
</job>
```

RFT Options

```
<fileStageOut>
  <transfer>
    <sourceUrl>file:///tmp/job.out</sourceUrl>
    <destinationUrl>
       gsiftp://host.domain:2811/tmp/stage.out
    </destinationUrl>
    <rftOptions>
      <subjectName>
  /DC=org/DC=doegrids/OU=People/CN=Stuart Martin 564728
      </subjectName>
      <parallelStreams>4</parallelStreams>
    </rftOptions>
  </transfer>
</fileStageOut>
```

RSL Variable

- Enables late binding of values
  - Values resolved by GRAM service
- System-specific variables
  - ${GLOBUS_USER_HOME}
  - ${GLOBUS_LOCATION}
  - ${GLOBUS_SCRATCH_DIR}
    - Alternative directory shared with compute node
    - Typically providing more space than user's HOME dir

RSL Variable Example

```
<job>
  <executable>/bin/echo</executable>
  <argument>HOME is ${GLOBUS_USER_HOME}</argument>
  <argument>SCRATCH = ${GLOBUS_SCRATCH_DIR}</argument>
  <argument>GL is ${GLOBUS_LOCATION}</argument>
  <stdout>${GLOBUS_USER_HOME}/echo.stdout</stdout>
  <stderr>${GLOBUS_USER_HOME}/echo.stderr</stderr>
</job>
```

How to use Client Software

- Command line programs
- WSDL interface

ManagedJobFactory portType

- createManagedJob operation
  - Creates either an MMJR or MEJR
  - Input:
    - Initial Termination Time
    - Job ID
      - UUID of the job resource, for job reliability/recoverability
    - Subscribe Request
      - Client can include a request to subscribe for job state notifications with the job submission to avoid an extra operation call
    - Job Description / RSL
      - Either a single or multi-job description
  - Output:
    - newTerminationTime - new termination time of the job resource
    - managedJobEndpoint - EPR of the newly created job resource
    - subscriptionEndpoint - EPR of the notification subscription

ManagedJob portType

- release operation
  - Release a *holdState* set in the job description
    - Only one hold state can be set/released
  - Input: None
  - Output: None
- State change notifications
  - State - job state (Active, Pending, Done, Cleanup…)
  - Fault - fault causing a Failed state (if applicable)
  - Exit Code - exit code of the job process if Failed
  - Holding - boolean indicating if the job is in a hold state

ManagedJob portType

On destroy, or soft state termination…

The MJS will cleanup everything

1. Stop any outstanding tasks
   - Cancel/terminate the execution
   - Destroy RFT stage in, out requests
2. Process CleanUp state
   - Submit request to RFT to remove files/directories
     - RSL attribute *fileCleanUp*
   - Remove job user proxy file
3. Destroy job resource

ManagedExecutableJobService

- Executes the requested job process(es) specified in the RSL
- Resource Properties (ManagedExecutableJobPortType)
  - serviceLevelAgreement - the RSL / Job Description
  - state - the current job state
  - faults - the fault causing a Failed state
  - localUserId - the username of the resource owner
  - userSubject - the GSI subject of the resource owner
  - holding - boolean indiciating the job is holding
  - stdoutURL - the GridFTP URL to the stdout file
  - stderrURL - the GridFTP URL to the stderr file
  - credentialPath - the local path to the user proxy file
  - exitCode - the exit code of the job proces (if applicable)

ManagedMultiJobService

- Processes a multi-job RSL, and submits the sub-jobs to the specified ManagedJobFactoryService.
  - Sub-jobs cannot be multi-jobs themselves.
- Resource Properties (ManagedMultiJobPortType)
  - serviceLevelAgreement - the multi-job RSL / Job Description
  - state - the current overall state
  - faults - the fault causing a Failed state
  - localUserId - the username of the resource owner
  - userSubject - the GSI subject of the resource owner
  - holding - boolean indiciating all jobs are holding
  - subJobEndpoint - list of endpoints to the sub-jobs

Important Notice!!

- Our goals are:
  - Highly functional interface
    - grid service WSDLs
    - C API
    - Java API
  - Expressive RSL
  - Basic command line clients
    - Should be useable from shell scripts
  - Collaborate with others to create more capable and complete clients
    - E.g. Condor-G grid manager, Platform's CSF

GRAM: Part 3

How to administer servers…

Typical GRAM service setup

- Host credentials
  - For client/service authentication
  - For client authorization of the service
  - Existing GT2/GT3 host certs can be used
- Gridmap file
  - Entries for each user allowed to execute job's
  - Maps the grid ID to a local user account
  - Same syntax as GT2, GT3 gridmap files
- Installed sudo
  - GRAM runs commands in the user's account

sudo configuration

- sudo policies
  - Done by hand by root
    Runas_Alias GRAMUSERS = ! root, ! wheel, …
  - globus ALL=(GRAMUSERS) NOPASSWD:
    /sandbox/globus/install/libexec/globus-gridmap-and-execute
    /sandbox/globus/install/libexec/globus-job-manager-script.pl *

    globus ALL=(GRAMUSERS) NOPASSWD:
    /sandbox/globus/install/libexec/globus-gridmap-and-execute
    /sandbox/globus/install/libexec/globus-gram-local-proxy-tool *
- globus-gridmap-and-execute
  - Redundant if sudo is locked down tightly
  - Enforce that GRAM only targets accounts in gridmap
    - So sudo policy need not enumerate all GRAM users at large/dynamic sites
    - In fact, you can audit this tool and change GRAMUSERS to ALL if you like…
  - Replace this with your own authz tool (callout)

Local scheduler

- Manages the jobs on compute resource
  - pbs, LSF, Condor, SGE, loadleveler, …
- Fork "scheduler" is default
  - included in the GT release and installed automatically
- Installing a scheduler bundle

```
% ./install-wsrf $GLOBUS_LOCATION
% $GL/sbin/gpt-build
        scheduler/gt4-gram-pbs-3.9-src_bundle.tar.gz
% gpt-postinstall
```

File staging functionality

- GridFTP Server
  - Recommendation: run on a separate host from GRAM service container to improve performance
    - For better scalability
- RFT
  - Requires PostgreSQL DB setup

GRAM / GridFTP file system mapping

- Associates compute resources and GridFTP servers
- Maps shared filesystems of the gram and gridftp hosts, e.g.
  - Gram host mounts homes at /pvfs/home
  - gridftp host mounts same at /pvfs/users/home
- GRAM resolves file:/// staging paths to local GridFTP URLs
  - File:///pvfs/home/smartin/file1... resolves to:
  - gsiftp://host.domain:2811/pvfs/users/home/smartin/file1
- $GL/etc/gram-service/globus_gram_fs_map_config.xml
- Map will be a MJFS Resource Property in 4.0

Non-default Setup

- setup-gram-service-common
  - To change GRAM configuration
  - Run in $GLOBUS_LOCATION/setup
- GridFTP Server config
  - Default is for localhost, port 2811
  - --gridftp-server=gsiftp://gridftp.host.org:1234
- RFT Service config
  - Default is localhost, port 8443
  - --staging-host=host.domain.org
  - --staging-port=4321

Setup: Container Credentials

- Default: host credentials
  - /etc/grid-security/containercert.pem
  - /etc/grid-security/containerkey.pem
- To configure for a user proxy
  - Update container global security descriptor
    - Comment out <credential> element $GL/etc/globus_wsrf_core/global_security_descriptor.xml
  - Tell GRAM the subject to expect for authorization of the RFT service
    - ./setup-gram-service-common --staging-subject= "/DC=org/DC=doegrids/OU=People/CN=Stuart Martin 564720”
  - Use “-self” argument with globusrun-ws

GRAM: Part 4 - gridwise

- How to implement a new scheduler adapter…

Help, FAQs, diagnostics

- 4.0 gram documentation (evolving)
  - Guides: admin, user, developer, overview
  - Comments and suggestions welcome!
  - http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/execution/wsgram/
- See WS JAVA Core Dev Guide
  - http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/common/javawscore/developer/index.html#debugging
  - Application logging - log4j
    - For gram use log4j.category.org.globus.exec=DEBUG
  - Tracing SOAP messages

Further Help, FAQs, Diagnostics

- First search the globus email lists
  - http://www-fp.globus.org/about/email-archive-search.html
- No luck, then post questions to a list
  - Discuss@globus.org
    - 1000+ participants
  - Developer-discuss@globus.org
    - 100+ participants
- If you've found a bug
  - http://bugzilla.globus.org/
  - GRAM product, wsrf* components