GRAM

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

Job Submission Methodology

- Grid Service Factory Pattern
  - Create Service
    - > Service instance is created
    - > Request is validated
    - > User's job request is *ready* to be started
  - Start operation
    - > User's job request is started
    - > Service instance monitors job request
    - > Updates request SDE
  - Job control
    - > Ensures client received a handle to the job before resources have been consumed

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

GRAM in GT3 Releases

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

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

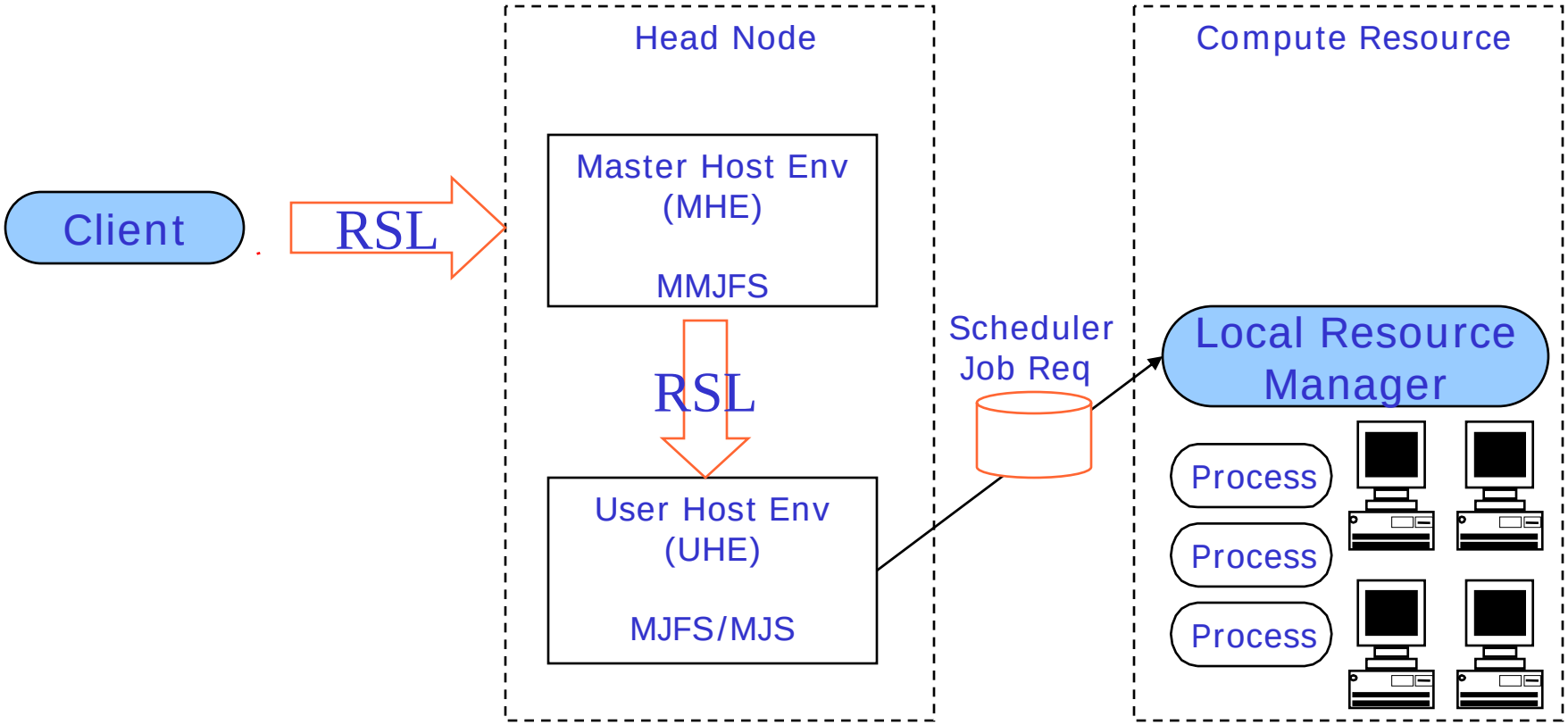
the globus alliance
www.globus.org

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)

the globus alliance
www.globus.org

ManagedJob Job Submission



Resource Specification Language

- Much of the power of GRAM is in the RSL
- XML schema defined language for specifying job requests
  - Managed Job Service translates this common language into scheduler specific language
- GRAM service understands a well defined set of elements
  - executable, arguments, directory, …

the globus alliance
www.globus.org

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

the globus alliance
www.globus.org

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

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

the globus alliance
www.globus.org

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

Extending GRAM RSL

- Use the ANY element in gram:jobAndAnyType
  - No element validation
- Extending the GRAM RSL schema
  - Extend gram:jobType
  - Add new definitions, but ***must*** be one of the pre-existing types
  - Simple examples in next release
- Elements and values will get propagated to the managed job scheduler Perl modules

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

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

the globus alliance
www.globus.org

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>
```

Deprecated RSL-1 Attributes

- gramMyjob
  - Value is one of “collective”, “independent”
  - Defines how the globus_gram_myjob library will operate on the <count> processes
    - collective: Treat all <count> processes as part of a single job
    - independent: Treat each of the <count> processes as an independent uniprocessor job
- dryRun=true
  - Do not actually run job

Deprecated RSL-1 Attributes

- saveState = yes/no
  - Always saves state
  - Causes the jobmanager to save job state/information to a persistent file on disk
  - Allow recovery from a jobmanager crash
- twoPhase
  - Implemented in Managed Job port type
    - > Allows reliable job submission
    - > Allow client to reliably determine completion vs failure of a job

the globus alliance
www.globus.org

Deprecated RSL-1 Attributes

- restart = old jm contact
  - Automatically recovers/restarts
- (stdoutPosition=<int> <int>)
- (stderrPosition=...)
  - Implemented in File Stream port type

GRAM grid services

- We know how to specify a job using RSL
- Now how do we submit and manage that job?
  - Managed Job (Factory) Service
    - > Defines an OGSI/GWSDL interface for submitting, monitoring and controlling a job
    - > MJS uses the File Stream Factory Service to manage the job's stdout and stderr file streaming
    - > MJS exposes the stdout and stderr File Stream Factory Grid Service Handles (GSH) in Service Data Element

Gram Clients

- managed-job-globusrun command line
  - Similar to pre-OGSI globusrun program
  - Useful for ***simple*** scripting and testing
- We anticipate the community will contribute robust and scalable clients
  - E.g. Condor: Condor-G, Grid Manager
  - Platform Computing: Community Scheduler Framework (CSF)

ManagedJobFactory portType

- CreateService operation
  - Prepare a job for submission on a remote resource
  - Input:
    - RSL specifying the job to be run
  - Output:
    - Locator (Grid Service Handle (GSH) and/or Grid Service Reference (GSR)) to ManagedJob service (MJS)
      - WSDL definition of the MJS instance
- Service Data Elements - None

ManagedJob (MJS) portType

- Start operation
  - Start/submit job to the compute resource
    - > Client credential is delegated to MJS instance
    - > Stdout/err FSFSs are started (if not /dev/null)
    - > File staging is done (if necessary)
    - > Submit job to local scheduler
  - Input:
    - > None
  - Output:
    - > Initial job state – typically, UNSUBMITTED
    - > If job state is FAILED, then an MJS fault is included

ManagedJob portType

- Start operation (continued)
  - Faults:
    - > Numerous: RSL, Credentials, Gass cache, file staging, file streaming, …
    - > Extended from OGSI faultType
      - StateWhenFaultOccurred
      - Script – method that may have caused the fault (e.g submit, stage_in, proxy_relocate)
      - gt2ErrorCode
      - cause

ManagedJob portType

- On destroy, or soft state termination
  - The MJS will cleanup everything
    - > Cancel the job
    - > Destroy File Stream Factories/Services
    - > Cleanup directories/files
      - Scratch dir
      - Gass cache

the globus alliance
www.globus.org

ManagedJob portType

- Service Data Elements
  - ManagedJobState
    - > UNSUBMITTED, PENDING, ACTIVE, FAILED, DONE, SUSPENDED, STAGEIN, STAGEOUT
    - > If FAILED, then MJS fault is included
  - ManagedJobSLA
    - > The job's RSL
  - ManagedJobUserIdLocal
    - > The account that the job is running under
  - UserIdGridCredentials
    - > The DN of the credentials used to authenticate with the MJFS
  - stdoutHandle, stderrHandle
    - > GSH to FSFS for job's stdout, stderr

FileStreamFactory portType

- Purpose
  - Enable data streaming from a local file to multiple URL destinations
  - One factory per stdout/err file

FileStreamFactory portType

- CreateService operation
  - Prepare to stream job's stdout or stderr to a destination URL
  - Input:
    - Destination URL
  - Output:
    - Locator to FileStream service

FileStreamFactory portType

- Service Data Elements
  - sourcePath
    - the local file path from which the stream begins
  - fileSize
    - The current size of the file which this factory will stream

FileStream portType

- Start operation
  - Start the streaming to the destination URL
  - Input:
    - None
  - Output:
    - None
  - Faults
    - InvalidUrlFault, InvalidPathFault, FileTransferFault, CredentialsFault

FileStream portType

- Service Data Elements
  - destinationUrl
    - > URL where the data is being streamed to
  - Done
    - > Flag indicating whether if streaming of the data file to the destination URL has completed

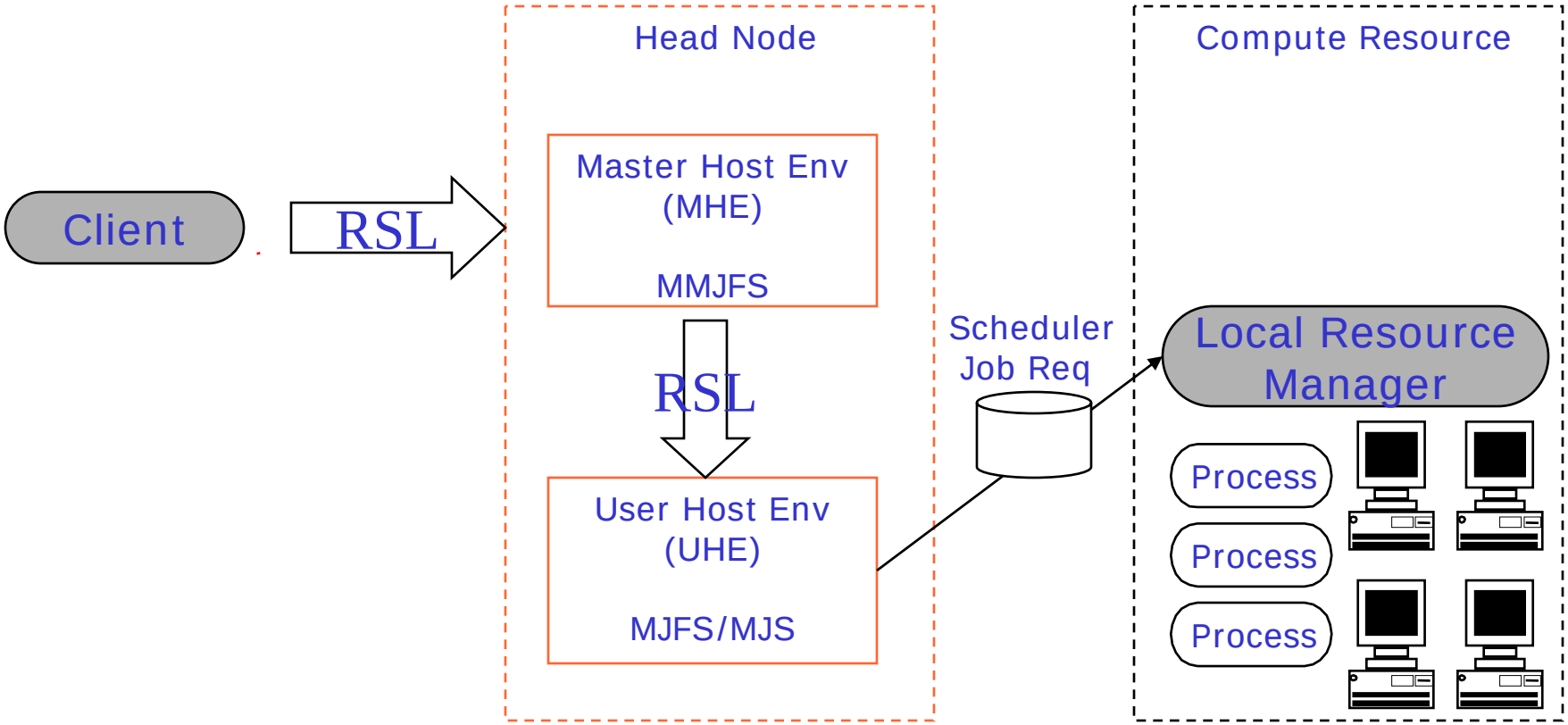
GT3 GRAM Client Interfaces

- Java & C client stubs for MMJFS, MJFS, MJS, FSFS, FSS
- 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

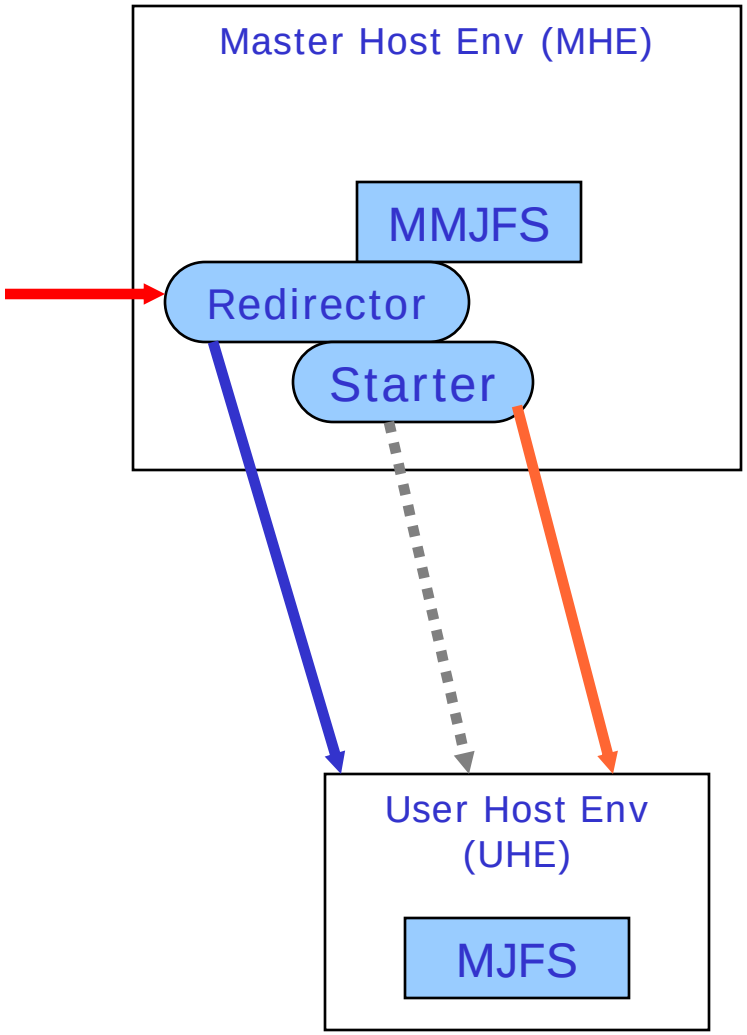
Important Notice!!

- 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

Service Components



UHE Creation

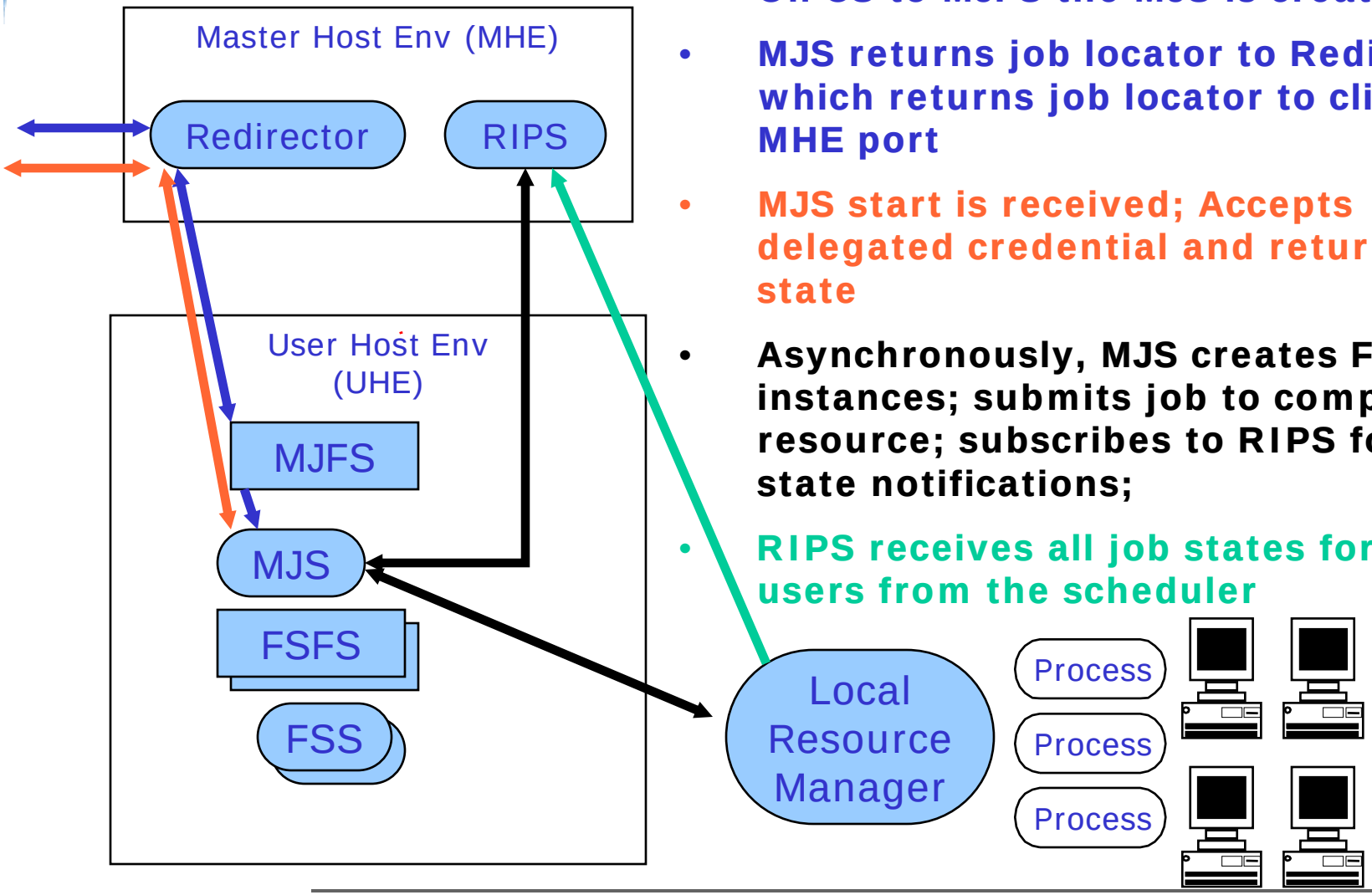


- **MMJFS uses the redirector so CreateService calls are forwarded or result in the starting of a new UHE**
- **A CreateService call is received**
- **Starter prepares and starts the UHE**
- **The Starter waits for the UHE to be started up (ping loop) and returns the target URL to the Redirector**
- **The Redirector forwards the createService call to the MJFS unmodified and mutual authentication/authorization can take place**

Why MMJFS & MJS?

- Avoid running anything substantial as root
  - Don't run SOAP stack as root
  - Buffer overflow attack or similar would compromise whole machine
  - Use a setuid program that is only capable of starting a pre-configured UHE in a user account
- Route all communication through a single port to avoid firewalls

MJFS Job Creation



- **On CS to MJFS the MJS is created**
- **MJS returns job locator to Redirector which returns job locator to client with MHE port**
- **MJS start is received; Accepts delegated credential and returns job state**
- **Asynchronously, MJS creates FSFS and instances; submits job to compute resource; subscribes to RIPS for job state notifications;**
- **RIPS receives all job states for all users from the scheduler**

MJS to Resource Interface

- Job submission
- Job monitoring
- File staging
- Compute cluster file system

Job Submission

- The RSL is converted to the syntax of the local scheduler for submission
  - The MJS serializes the RSL to a file
  - The MJS executes a perl script which:
    - evaluates and translates each RSL element to a scheduler command
    - submits the job
    - returns the scheduler job id to the MJS for monitoring
- Same Perl scripts used in GT2

Job Monitoring

- The MJS instances can monitor jobs in two ways:
  - Resource Information Provider Service (RIPS)
    - A specialized notification service
    - Maintains job information from the scheduler
    - Scheduler info provider outputs queue and job data in XML
  - Poll the scheduler directly
    - Only option for FORK

File Staging

- MJS calls perl script which uses globus-url-copy to transfer files
- Same Perl scripts used in GT2

Compute Cluster File System

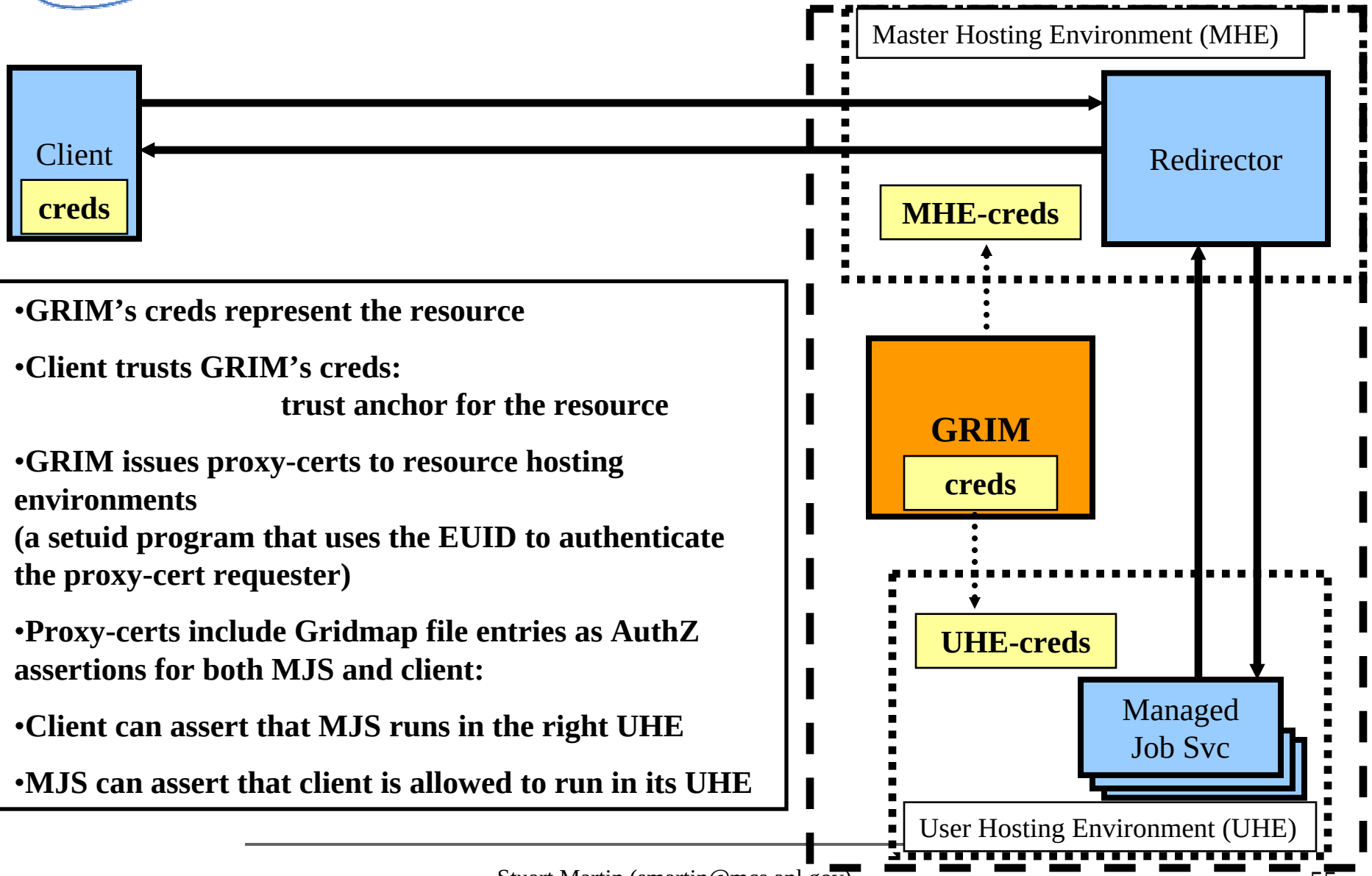
- MJS attempts to make files on the head node available to the job on the compute node
  - gass_cache in user's HOME dir
    - > Typically NFS mounted
      - Proven to be unreliable on large linux clusters
    - > grid credential, staged files, job's stdout/err
- MJS needs to provide job file verification
- Same Perl scripts used in GT2

the globus alliance
www.globus.org

MJS to Resource Interface

- Your scheduler is not supported?
  - No problem. See www.globus.org/gram "scheduler interface tutorial" step by step for writing an interface for an unsupported scheduler
  - JM scheduler setup package
    - submit, *poll, cancel and RIPS info provider

GT3 GRAM Security: "GRIM"



- **GRIM's creds represent the resource**
- **Client trusts GRIM's creds:**
  trust anchor for the resource
- **GRIM issues proxy-certs to resource hosting environments**
  (a setuid program that uses the EUID to authenticate the proxy-cert requester)
- **Proxy-certs include Gridmap file entries as AuthZ assertions for both MJS and client:**
- **Client can assert that MJS runs in the right UHE**
- **MJS can assert that client is allowed to run in its UHE**

the globus alliance
www.globus.org

UHE files

- ~/.globus/uhe-<hostname>
  - server-config.wsdd
    - > Services available in this UHE container
  - log
    - > Log for all UHE and MJS activity/debugging
  - log4j.properties
    - > Controls the logging output to the above log file
    - > log4j.category.org.globus.ogsa.impl.base.gram.jobmanager=DEBUG
  - gridMap
    - > DN(s) used for UHE authentication; only job requestor's
  - var
    - > Currently only MJS recovery files are stored here
  - client-config.wsdd
    - > Standard default

MJS files

- ~/.globus/job/<host>/<unique job ID>
  - Delegated User Proxy
  - Job's Stdout / Stderr files (if requested)
- ~/.globus/.gass_cache
  - Staged in executable and stdin (if requested)
- Scratch_dir (if requested)
- Staged files (if requested)
- RSL file used by perl scheduler scripts
- MJS instance recovery files
  - Stored in <UHE DIR>/var/MJS_recovery/<jobs GSH id>

GRAM Fault Tolerance

- MHE
  - Start/Recover UHEs
    - > Store UHE port in a mapping file for recovery
  - Monitor UHEs
    - > Subscribe to UHE containerState SDE for “SHUTDOWN”
    - > Ping UHE to detect a crash (periodically)
  - Cannot stop/kill UHEs!
- UHE
  - Shuts itself down when “inactive”
    - > No instances
    - > No SOAP messages sent to UHE in x minutes
    - > Set management service SDE containerState to “SHUTDOWN”
      - Can only be done locally by other tasks in the UHE

GRAM Fault Tolerance

- MJS
  - Each MJS instance stores recoverability data to a local file based on the GSH
  - Post-creation code of MJS checks if the recoverability file is found. If so, it resumes from the previous job state from the file.
    - > Different than pre-OGSI GRAM: Client no longer needs to initiate restart of a job manager

Higher level Resource Management Services

- To date, no GT3 co-allocators (DUROC)
  - simultaneous allocation of a resource set
  - mpich-g2 is DUROC's only user
- Community Scheduler Framework may fill this gap in the future

Changes: GT3.0 → 3.2

- Added a grid service fault to the MJS job status SDE
- MJS instance automatic de/reactivation
- Changes to GT3 Core to improve scalability
- Improved fault tolerance
- A job's files are maintained in a unique directory, instead of the gass cache

the globus alliance
www.globus.org

Future Work

- Add Service Agreements to GRAM
  - Based on WS-Agreement
- Advance reservations
- Highly scalable ManagedJobService
  - Target 200k jobs
- Cluster File Verification Solution