EMI ES

a common interface to ARC, gLite and UNICORE computing elements

Dr. Bernd Schuller

Jülich Supercomputing Centre

On behalf of the EMI ES team

EMI ES team

- Massimo Sgarravato, Eric Frizziero, Luigi Zangrando (gLite)
- Martin Skou Andersen, Aleksander Konstantinov, Balazs Konya, Oxana Smirnova (ARC)
- Shahbaz Memon, Shiraz Memon, Bernd Schuller (UNICORE)

- https://twiki.cern.ch/twiki/bin/view/EMI/WebHome

Basic assumptions and scope

- Service of the “compute element” type
  - ARC CE, Cream CE, UNICORE atomic services
- Accepts and manages single jobs (“activities”) that run on some backend
  - Cluster managed by an LRMS
  - “Standalone” compute node
- Out of scope (at least for v1.0)
  - Brokering / forwarding of jobs
  - Multi-jobs (e.g. JSDL parameter sweep)

Outline

- Some background
- EMI Execution Service
  - Functionality
  - Processing model and state model
  - Data staging
  - Delegation
  - Activity description schema

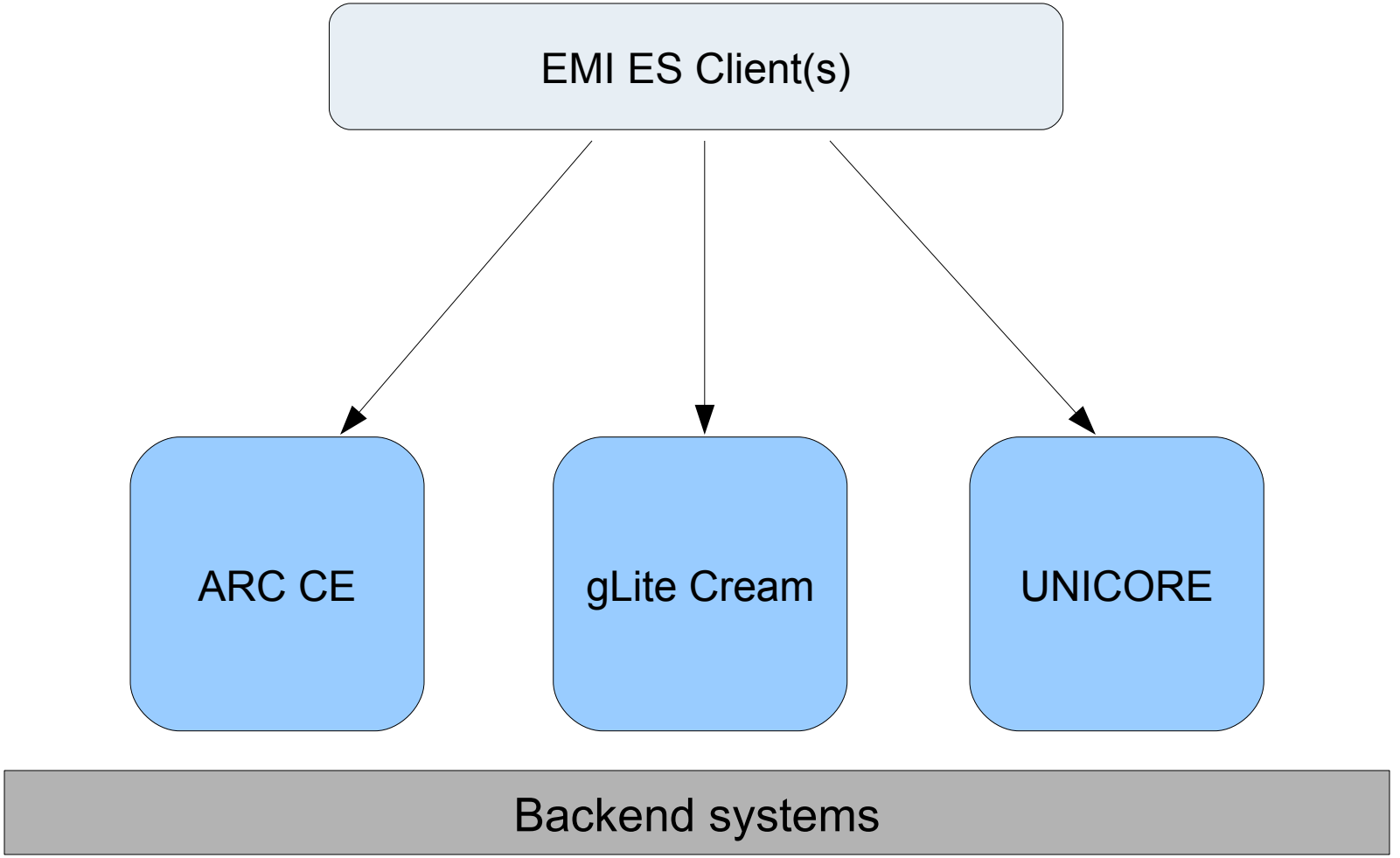
Background: EMI

- EMI is a “joint venture” of European middleware providers
  - ARC, UNICORE, gLite, dCache, …
- Goals
  - Support existing gLite, ARC, UNICORE, dCache
  - Harmonisation, evolution, integration
    - Less maintenance effort, reduced duplication, …
  - New developments: clouds, virtualisation, messaging, ….

Background: EMI ES

- Common specification by key people from ARC, UNICORE, gLite
- Originally based on work done in the Open Grid Forum PGI working group
- Status and plans:
  - First version (EMI Milestone) released December 2010, consists of document and xsd/wsdl
  - Prototype implementations in 2011 by all the providers
  - Second version will be done → feedback loop
- https://twiki.cern.ch/twiki/bin/view/EMI/EmiExecutionService

Benefits



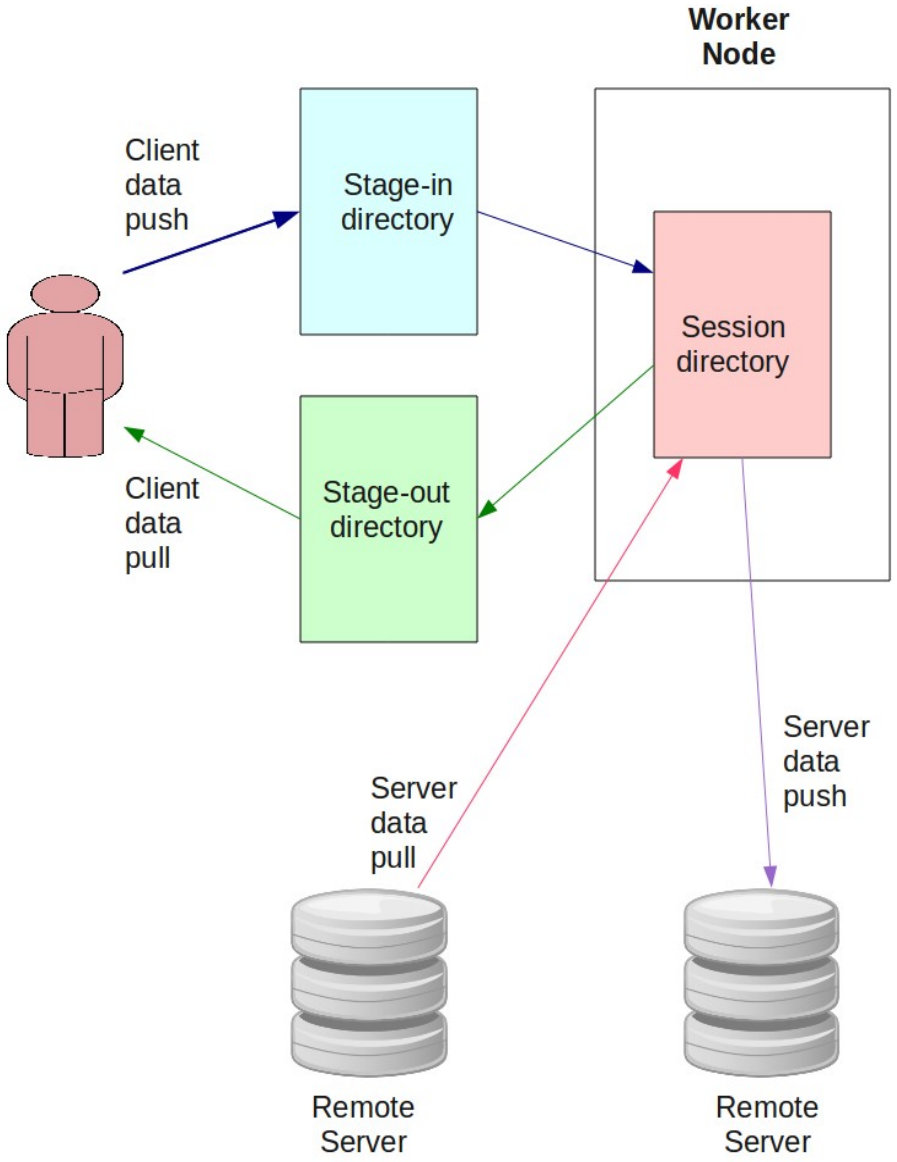
Functionality

- Activity Factory
  - Create new activities
  - Get service info in GLUE2 format
  - Manage delegated credentials
- Activity Management
  - Manage activities (pause, resume, abort, …)
  - Get activity status
  - Get activity info in GLUE2 format

Bulk operations

- Where possible, operations support sets of requests (“vector operations”)
- Operations are asynchronous
  - Avoid slow replies
- Example:
  - Submission of many jobs
    - Service immediately replies with vector of IDs
    - Validation processed async.

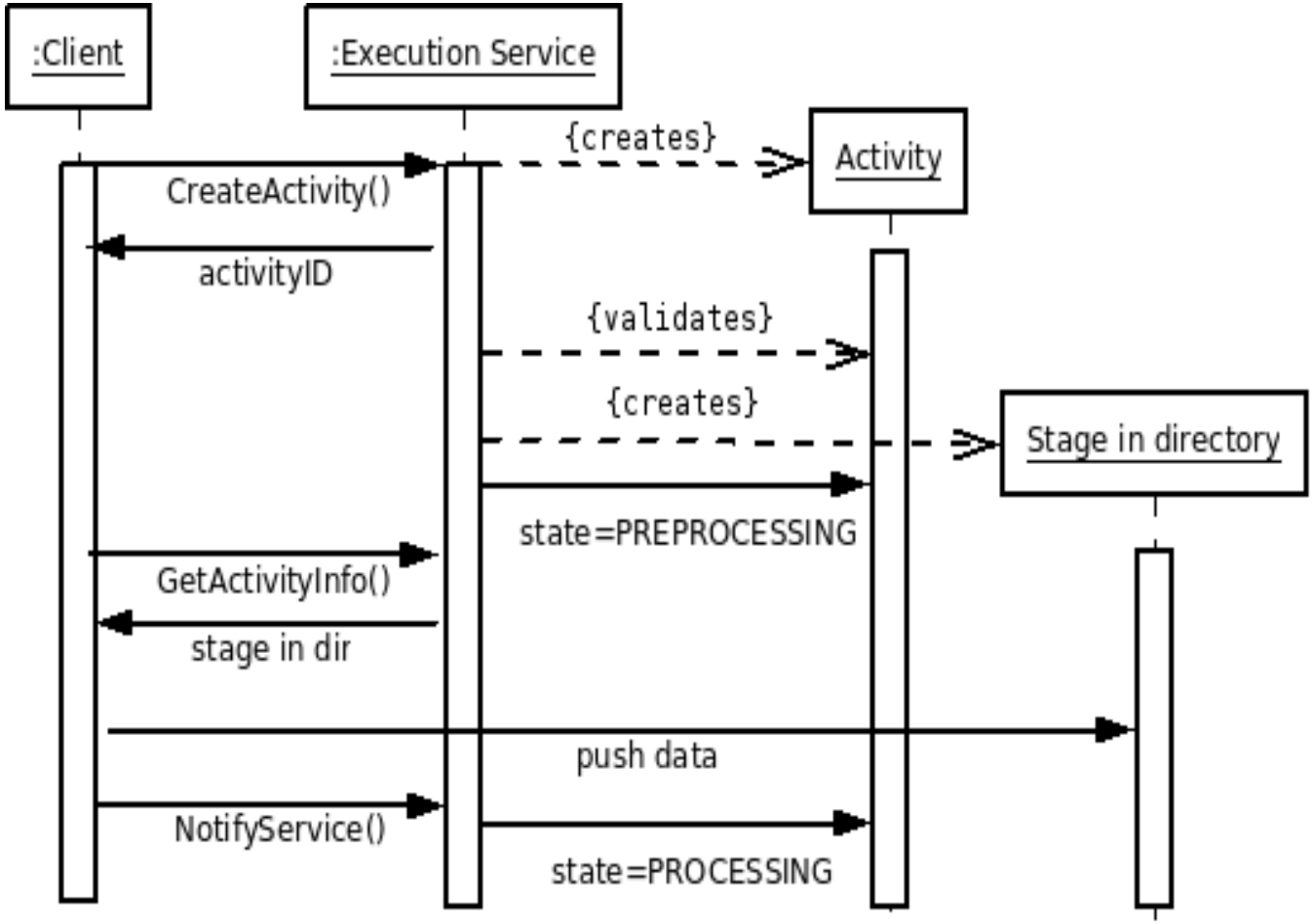
Data Staging



Data staging

- Three dirs, available at various stages in the processing
  - stage in / session dir / stage out
- Modes
  - Server pull/push
  - Client push/pull

Activity creation and data push



State model

- States are grouped into 5 phases
  - **ACCEPTED** : the activity has been created and is being validated
  - **PREPROCESSING**: the activity environment, including data is being prepared
  - **PROCESSING**: the job is being submitted to and processed by the underlying system or it is already handled by the batch system. This phase is split into the following states:
    - **PROCESSING-ACCEPTING** – ES communicates with the underlying batch system
    - **PROCESSING-QUEUED** – job was accepted by the batch system, but the payload is not yet running
    - **PROCESSING-RUNNING** – the payload is running
  - **POSTPROCESSING**: the job has left the batch system. It is possibly doing stage-out, releasing resources (de-provisioning)
  - **TERMINAL**: there is no more activity by the service, the activity is in a final state. Output data is available for client data pull

State attributes

- Indicate some transient condition, or convey additional information about the primary state
- Examples in the PREPROCESSING state
  - Client-stagein-possible
  - Provisioning
  - Client-paused
  - Server-stagein

Delegation

- Allow to delegate trust to the service, avoiding GSI
  - Currently *limited* to proxies
  - SAML assertions as a future option
- InitDelegation
  - Client asks for a X509 CSR from the server
  - Server responds with CSR + delegation ID
- PutDelegation
  - Client puts the signed CSR + delegation ID to the server, allowing the server to create a proxy certificate

Job description

- XML schema
  - https://twiki.cern.ch/twiki/pub/EMI/EmiExecutionService/es-schemas-1.0-src.tar
- Elements
  - Job metadata (job name and such)
  - Executable + its attributes
  - Runtime environments
  - Resources
  - Parallel environment
  - Data staging specification

Resource requirements

- Operating system and platform
- Network type and node connectivity
- Physical and virtual memory
- Slot allocation
  - Number of slots
  - Slots per host
  - Exclusive execution
- CPU time or wall clock time
- Queue name

Executable + its attributes

- Executable
  - May be also by a runtime environment (“abstract” application)
- Input/output/error redirect
- Arguments
- Environment
- Pre/post command to execute

Runtime environments

- Defined using Name, Version and Option tags
- Provide the job's “ecosystem”
  - Used by the ES to generate “concrete” job
  - May provides environment variables, path settings, etc.
    - e.g. “module load deisa”
  - May provide an executable
    - e.g. “Gromacs” → /opt/gromacs/bin/gromacs

Parallel environments

- Simple, user-friendly way of setting up parallel jobs
- Have Type (MPI, OpenMP, ...), Version and additional Option tags
- Resource specification
  - ProcessesPerSlot
  - ThreadsPerProcess

Data staging

- InputFile
  - Name, Source
- OutputFile
  - Name, Target
- Name
  - Relative to session directory
- Source, Target
  - URI
  - Options
  - Delegation ID

Summary

- EMI ES effort has defined a joint web service interface to ARC, gLite and UNICORE compute services
- Incorporating good ideas from all three
- Provides common understanding of
  - Basic functionalities
  - Processing model
  - Data staging
  - Activitiy description and its semantics
- “Placeholder” solution for delegation that will need further work within EMI as a whole
- **Implementations expected in 2011**

Thank you!

EMI is partially funded by the European Commission under Grant Agreement RI-261611