Overview of the Application Hosting Environment

Stefan Zasada
University College London

The Application Hosting Environment

- Based on the idea of applications as web services
- Lightweight hosting environment for running unmodified applications on grid resources (NGS, TeraGrid, DEISA) and on local resources (departmental clusters)
- Community model: expert user installs and configures an application and uses the AHE to share it with others
- Simple clients with very limited dependencies
- No intrusion onto target grid resources

open middleware infrastructure institute uk
www.omii.ac.uk

Concepts

- Applications not jobs
  - Application could consist of a coupled model, parameter sweep, steerable application, or a single executable
- Out of the box AHE supports single job applications
- We use “application” to denote a higher level concept than a job
  - In AHE terminology, an application may require running multiple jobs
- Architecturally, the AHE is a portal, where the interface is a rich client, not a web browser
  - Of course, AHE services can be used behind a Web portal, if you like

Virtualizing Applications

- Application Instance/Simulation is central entity; represented by a stateful WS-Resource. State properties include:
  - simulation owner
  - target grid resource
  - job ID
  - simulation input files and urls
  - simulation output files and urls
  - job status
- Application exposed as web service

AHE Functionality

- Launch simulations on multiple grid resources
- Single interface to monitor and manipulate all simulations launched on the various grid resource
- Run simulations without manually having to stage files and GSISSH in
- Retrieve files to local machine when simulation is done
- Can use a combination of different clients – PDA, desktop GUI, command line

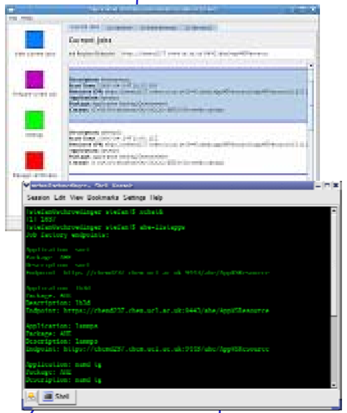
Accessing Resources

Local UCL resources

GridSAM / SGE

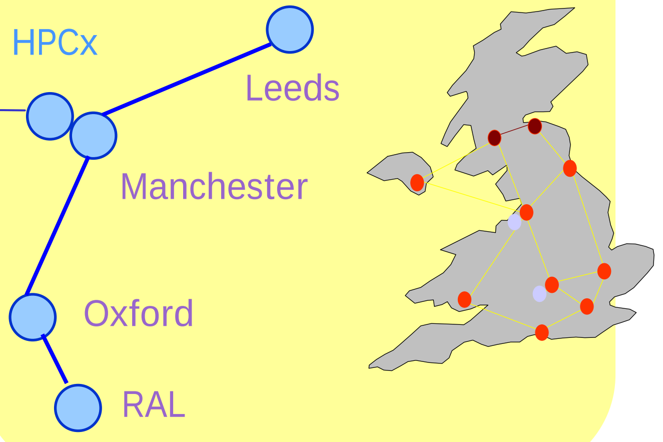
GridSAM / Globus

GridSAM / UNICORE

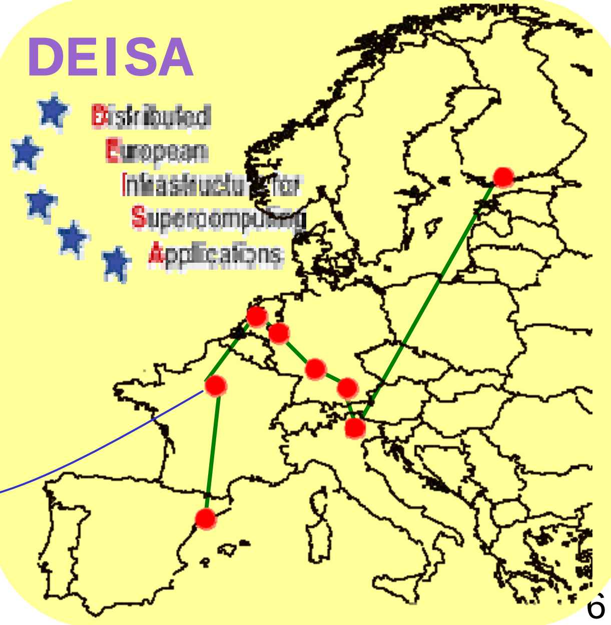


```
[stefan@schroedinger stefan]$ xchat&
[1] 1837
[stefan@schroedinger stefan]$ ahe-listapps
Job factory endpoints:

Application: [illegible]
Package: AHE
Description: [illegible]
Endpoint: https://chemd237.chem.ucl.ac.uk:[illegible]/ahe/AppWSResource

Application: lb3d
Package: AHE
Description: lb3d
Endpoint: https://chemd237.chem.ucl.ac.uk:9443/ahe/AppWSResource

Application: lammps
Package: AHE
Description: lammps
Endpoint: https://chemd237.chem.ucl.ac.uk:9115/ahe/AppWSResource

Application: namd tg
Package: AHE
Description: namd tg
```

NGS



DEISA



open middleware infrastructure institute uk
www.omii.ac.uk

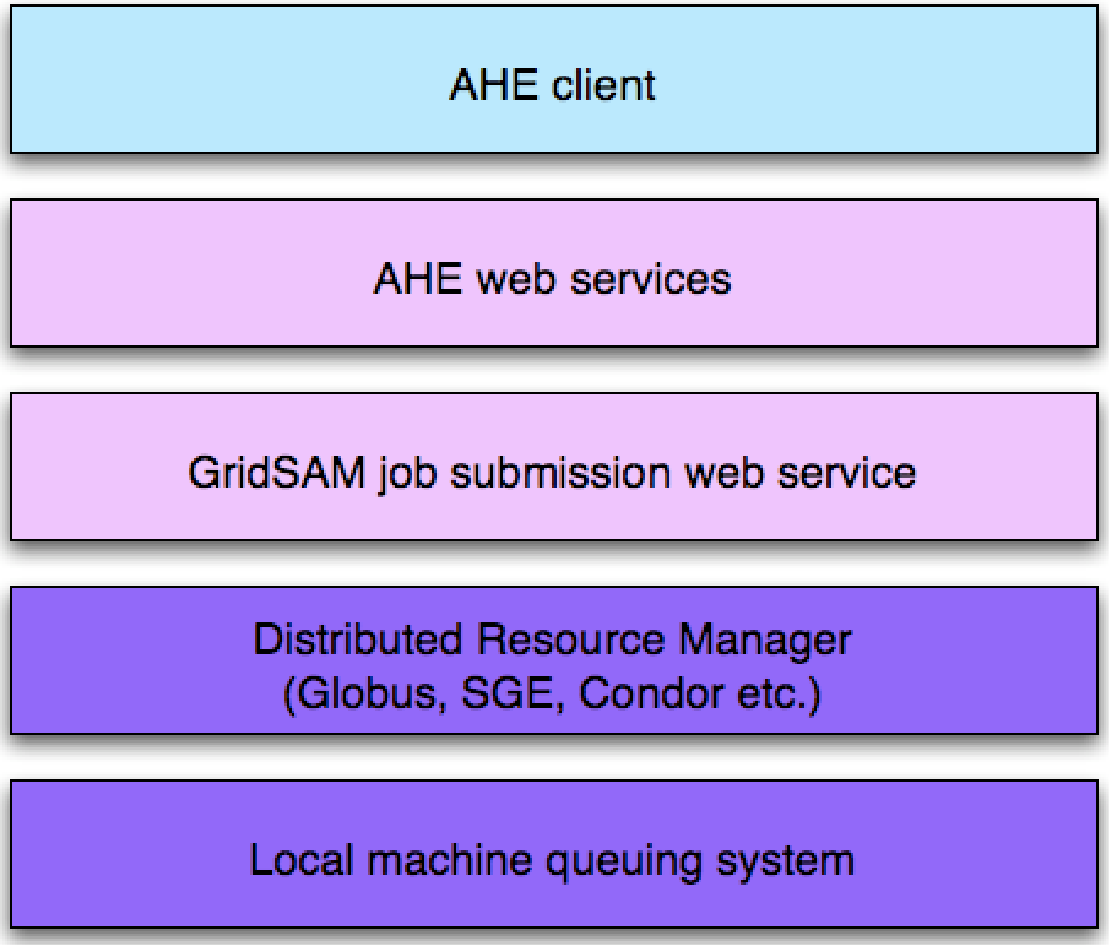
AHE Design Constraints

- Client does not have Globus installed locally
- Client is NAT'd and firewalled
- Client does not have to be a single machine
- Client needs to be able to upload and download files but doesn’t have local installation of GridFTP
- Client doesn’t maintain information on how to run the application
- Client doesn’t care about changes to the backend resources

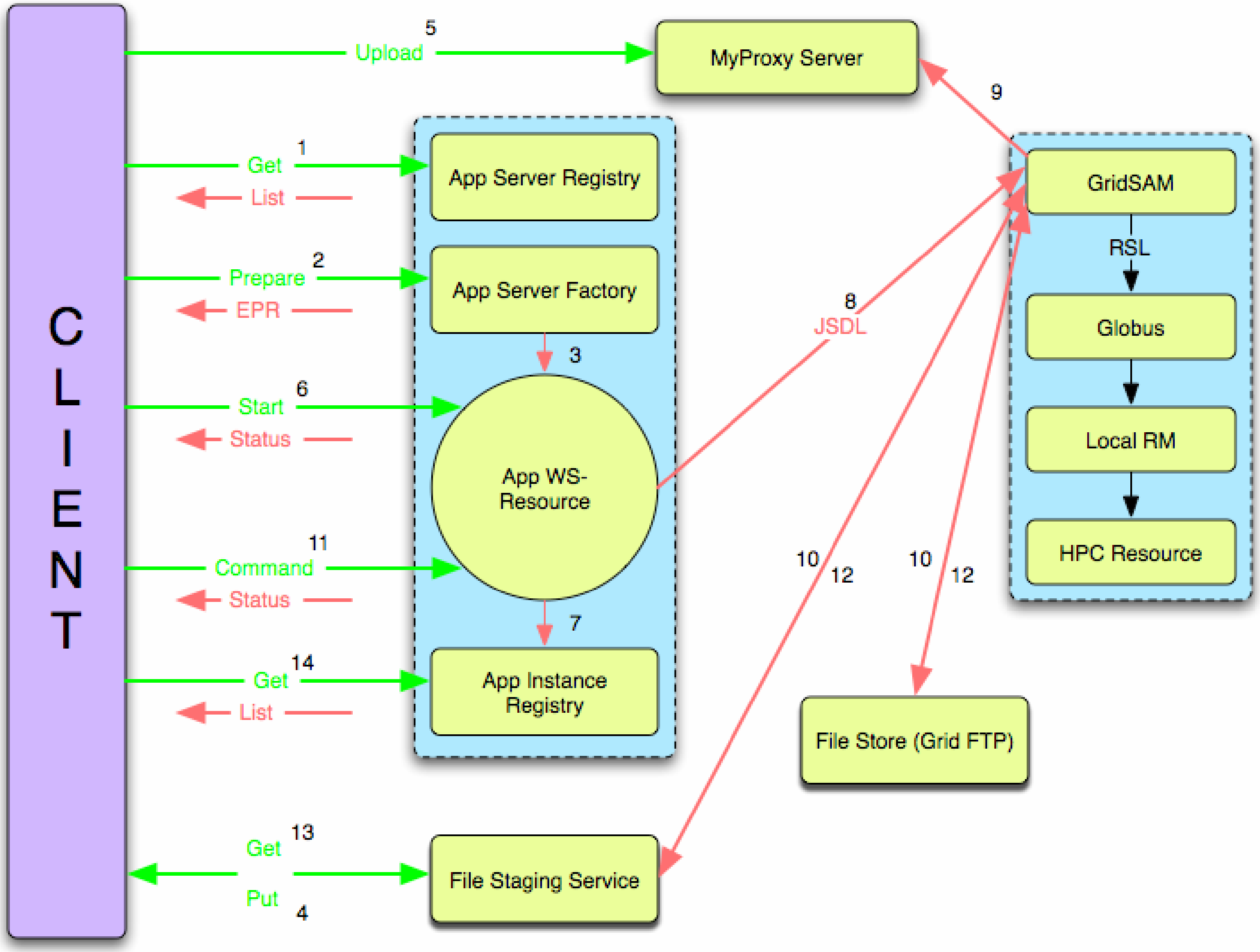
Meeting the Constraints

- AHE Client behind firewall => polls server to update job state etc.
- Uses intermediate filestaging area => GridFTP not installed
- All application specific information for running simulations on the grid resource is maintained on a central service => user can switch clients etc.
- Location of binary on grid resource configured on server => user doesn’t need to know
- GridSAM provides interface to job queue

Layered Architecture of the AHE


AHE client
AHE web services
GridSAM job submission web service
Distributed Resource Manager (Globus, SGE, Condor etc.)
Local machine queuing system

Service Architecture of the AHE



AHE Workflow

1. Optionally user generates proxy credential and uploads it to MyProxy server
2. Client sends “prepare” message to job factory => creates WS-Resource to manage job state
3. Client uploads input files to WebDav server
4. Client sends “start” message to WS-Resource
5. AHE sends job’s JSDL file to GridSAM
6. GridSAM moves input files from WebDav server to grid resource

AHE Workflow

7. GridSAM submits job to job queue
8. Client polls state of job by sending “monitor” message to WS-Resource
9. AHE polls GridSAM for job state and returns result to client
10. When job complete, GridSAM stages output files back to WebDav server
11. Client can stage output files from WebDav server back to local machine

AHE GUI CLIENT Functionality

- Discover Appropriate Resources
- Launch Application
- Monitor Running Jobs
- Query Registry of Running Jobs
- Stage Files to and from Resource
- Terminate Jobs

Client Assumptions

- Client is NAT'd and firewalled
- Client doesn't use GridFTP
- Client doesn't require Globus to be installed
- Client doesn't know anything about how to run the particular job
- Client doesn't maintain any information about job state

GUI Client Implementation

- Implemented in Java using Apache Axis WS API
- Demonstrates interoperability between WSRF::Lite and Java
- Implement job launching as a wizard, with each frame of the wizard covering a separate step in the launching process

Job Building Wizard

Wizard guides user through stages of submitting a job:

- Specifying constraints for job, e.g. number of processors to use
- Sends prepare message to cause the WS-Resource to be created
- Allows user to choose appropriate resource to run job

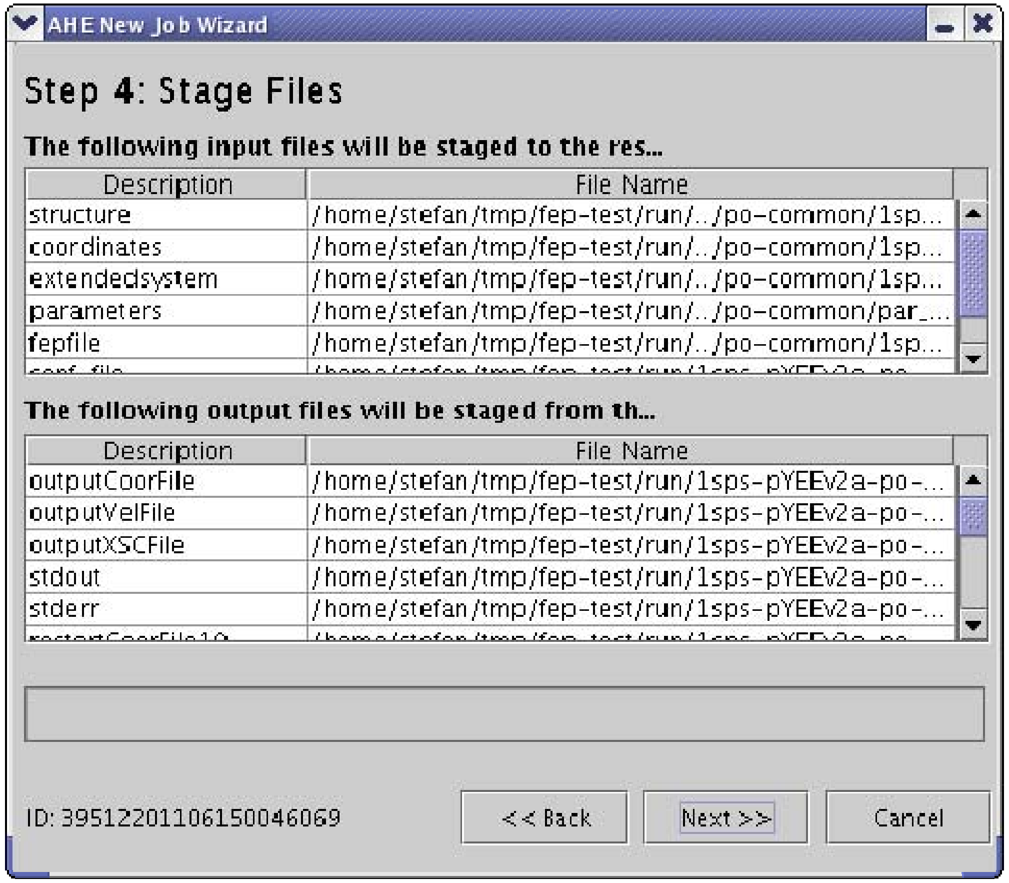
Job Building Wizard

- If plugin is available for particular type of job (e.g. NAMD) then client will attempt to process job configuration file to discover input and output files to be staged, otherwise the user will have to specify manually

- Client stages file to AHE file staging area (WWFS, Web-Dav etc.)

- User checks job details then submits it to AHE

AHE Job Building Wizard



Step 4: Stage Files

The following input files will be staged to the res...

Description	File Name
structure	/home/stefan/tmp/fep-test/run/../po-common/1sp...
coordinates	/home/stefan/tmp/fep-test/run/../po-common/1sp...
extendedsystem	/home/stefan/tmp/fep-test/run/../po-common/1sp...
parameters	/home/stefan/tmp/fep-test/run/../po-common/par_...
fepfile	/home/stefan/tmp/fep-test/run/../po-common/1sp...

The following output files will be staged from th...

Description	File Name
outputCoorFile	/home/stefan/tmp/fep-test/run/1sps-pYEEv2a-po-...
outputVelFile	/home/stefan/tmp/fep-test/run/1sps-pYEEv2a-po-...
outputXSCFile	/home/stefan/tmp/fep-test/run/1sps-pYEEv2a-po-...
stdout	/home/stefan/tmp/fep-test/run/1sps-pYEEv2a-po-...
stderr	/home/stefan/tmp/fep-test/run/1sps-pYEEv2a-po-...

ID: 39512201106150046069

Client Extensibility

- Plugins can be added to process application input files to automatically discover the input and output files that need to be staged
- If no plugin is available then a default case will allow users to specify input and output files manually
- Plugins implement AHEConfParser interface and follow specific naming convention
- Plugin .class files dropped into plug-in directory and picked up by GUI/command line clients

AHE Job Interaction

Once job has been prepared and submitted:

- Client will display job information, and poll resource at regular intervals to update status
- Client can terminate and destroy current jobs
- Once job is complete, client will stage input files back from WWFS if required
- Client allows user to query resource for list of current jobs – no state need be maintained in client

Java Client Development

- As AHE develops functionality, client will be extended (e.g. to support RealityGrid steering and resource co-allocation).
- Java codebase can be used to easily create other clients to interact with AHE (e.g. Java command line clients).
- GUI Client development will be greatly influenced by user feedback from early adopters

AHE Client Deployment

- Deploying client is trivial for the end user:
  - User’s machine must have Java installed
  - User downloads and untars client package
  - Imports X.509 certificate into Java keystore using provided script
  - Configures client with endpoints of AHE services supplied by expert user

- Ready to go!

Summary

- The AHE provides a lightweight, easily deployable environment for running unmodified scientific applications on the grid and local resources
- The AHE server is designed to be deployed by an expert user who uses it to share applications installed on grid resources
- The client is easily installed by any end user, requiring no intervention by system/network administrators
- By calling the command line clients from scripts, complex scientific workflows can be implemented

Practical 1

Running applications with the AHE:

- Install the AHE client on your system
- Set up a keystore containing your grid certificate
- Configure the client with settings for UCL’s AHE server
- Launch the sort application with the AHE GUI client
- Launch the sort application with the AHE command line client
- Manually specify input and output files for an application
- Retrieve application output from NGS machine

http://www.realitygrid.org/AHE/training/nesc/ex1/