



An Evaluation of the Application Hosting Environment Uk e-Science Engineering Task Force

Matteo Turilli, Oxford e-Research Centre

3 October 2007

Abstract:

The Application Hosting Environment (AHE) is a lightweight hosting environment that allows scientists to run applications on grid resources in a quick, transparent manner. The AHE provides resource selection, application launching, workflow execution, provenance and data-recovery, exposing a WSRF compatible interface to job management on remote grid resource using WSRF::Lite as its middleware. The review is based on version 1.0.1 and 1.0.2 of the AHE Server. Version 1.0.1 has a stand-alone installation process while version 1.0.2 has been included into the OMII stack.

UK e-Science Technical Report Series

Report UKeS-2007-08

Available from http://www.nesc.ac.uk/technical_papers/UKeS-2007-08.pdf

Copyright © 2007 The University of Edinburgh. All rights reserved.

ETF Grid Middleware Evaluation – AHE Server/Client Installation

General Information

“The Application Hosting Environment (AHE) is a lightweight hosting environment that allows scientists to run applications on grid resources in a quick, transparent manner. The AHE provides resource selection, application launching, workflow execution, provenance and data-recovery, exposing a WSRF compatible interface to job management on remote grid resource using WSRF::Lite as its middleware”.¹

AHE is a client-server architecture. The main idea behind AHE is to make it very easy for the end-user to submit jobs to a grid resource. The knowledge required for obtaining a proxy certificate, submitting a job, monitoring its state and retrieving its output is minimal. All the complexity is contained in the server set-up and configuration.

This evaluation focuses on the installation process of the server component of the AHE architecture and is not meant to be an in depth analysis of the whole client-server architecture. Consideration about memory and space requirements, stress testing and code audit are left for further investigations.

The review is based on version 1.0.1 and 1.0.2 of the AHE Server. Version 1.0.1 has a stand-alone installation process while version 1.0.2 has been included into the OMII stack.

Provider

The Application Hosting Environment (AHE) is funded by the EPSRC “Rapid Prototyping of Usable Grid Middleware” Project, GR/T27488/01, and by OMII under the Managed Programme RAHWL (Robust Application Hosting in WSRF::Lite) project.²

The GR/T27488/01 will expire on 2 April 2007.³ At the time of writing, I have not found public information about a further funding for the AHE development. Recently, the AHE server has been included into the installation of the OMII stack 3.2.0. OMII-UK

¹ <http://www.omii.ac.uk/downloads/release.jsp?releaseid=208>

² <http://www.realitygrid.org/AHE/index.shtml>

³ <http://gow.epsrc.ac.uk/ViewGrant.aspx?GrantRef=GR/T27488/01>

is a collaboration between centres at Southampton, Edinburgh and Manchester. It has been operating since 2004 and they declare on their web site their intention to be a long term provider of software and support for the e-Science community.

I did not find quantitative information about the actual user base of the AHE server. In (Coveney, Saksena et al. 2006)⁴ it is reported that “there is a significant AHE user base with others planning to use it. Favourable experiences have been reported for NAMD and LAMMPS applications hosted within the AHE as compared to alternative strategies for running jobs on grids”.

The recent introduction of the AHE server into the OMII stack facilitates the installation process and the distribution of the AHE server could be promoted by its introduction into the OMII stack as this facilitates the installation process of the server.

Licensing

GNU General Public License (GPL). “AHE: Application Hosting Environment (C) Copyright 2006, University College London, United Kingdom (C) Copyright 2006, University of Manchester, United Kingdom The Application Hosting Environment(AHE) comes with no warranty of any kind. It is a copyrighted code distributed free of charge under the terms of the GNU Public License (<http://www.gnu.org/copyleft/gpl.html>), which is commonly known as "open source" distribution. This means that anyone is free to use, modify, or extend AHE in any way they choose, but if you distribute a modified version of AHE, it must remain open-source, meaning you distribute it under the terms of the GPL. You should clearly annotate such a code as a derivative version of AHE. If you release any code that includes AHE source code, then it must also be open-sourced, meaning you distribute it under the terms of the GPL”.⁵

Supported Platforms

The AHE server runs on 32-bit Linux platforms. In the OMII 3.2.0 release notes is stated that “A re-factored server installation that downloads the individual components on

⁴ Coveney, P. V., R. S. Saksena, et al. (2006). The Application Hosting Environment:Lightweight Middleware for Grid-Based Computational Science. UK e-Science All Hands Meeting 2006.

⁵ <http://www.omii.ac.uk/downloads/release.jsp?releaseid=208>

demand will enable us to expand the number of supported platforms for these components in later releases”.⁶

The AHE client is a stand-alone application written in Java. As such, it can be downloaded as a tar.gz archive and installed independently from the server component of the architecture. The client has been tested on Fedora Core Linux versions 2-5 and MacOSX 10.3-4. It is expected to work on any Unix-like system with a Java Virtual Machine implementation.

The AHE server 1.0.1 was evaluated on a desktop PC, HP dc7600, Pentium D 3.20GHz, 1GB ram and the Linux distribution Ubuntu 6.10. The AHE server version 1.0.2 was evaluated on a Xen virtual machine with the Linux distribution Scientific Linux 4. The AHE client has been evaluated on an iMac with OSX 10.4.

Support

No commercial support is available. Support is offered by the two developers of the AHE server/client suite at <http://forge.nesc.ac.uk/projects/ahe/>. The site has no requests for support and the two forums are empty (opened on 11 Apr 2006). The site seems limited to the version 1.0.1. It seems possible that OMII will support the version 1.0.2.⁷ OMII support is offered via FAQ, extensive documentation, helpdesk and wiki web site.⁸

System Management

Documentation for System Managers (or expert users)

The installation documentation is only offered in pdf format.⁹ Documentation is fairly accurate, although some errors have been found.

The guide consists of a comprehensive step-by-step description of the procedures to follow to set up the AHE server. The organization of the documentation is a bit confusing, starting with the description of how to set up the AHE server and following with detailed analyses of how to set up the packages required for running the AHE server.

⁶ <http://www.omii.ac.uk/news/newsdetail.jsp?id=47>

⁷ http://www.omii.ac.uk/docs/3.2.0/user_guide/omii_release/sc_services/ahe/run_ahe_server.htm

⁸ <http://www.omii.ac.uk/support/>

⁹ <http://www.realitygrid.org/AHE/doc/AHEServerInstallationGuide.pdf>

A more canonical organization of the document would probably have begun with the requirements to end up with the AHE server installation.

The document is not a quickstart guide. It is made clear that the set up of the server is a matter for experienced users. This is consistent with the design choice of relegating all the complexity of the architecture into the server component, leaving the client as simple as possible.

Server Deployment

AHE server is a web-based container of services that require a certain amount of quite cumbersome configuration. The list of required modules and software is long:

- PostgreSQL
- Perl
- WSRF::Lite
- Apache
- XML::XPath
- XML::DOM::XPath
- Cwd
- HTTP::DAV
- SOAP::Lite
- DateTime::Format::W3CDTF
- DateTime::Format::Epoch
- Crypt::SSLeay
- Net::SSLeay
- Crypt::RSA
- OMII (for running the AHE server)
- GridSAM service (for running the AHE server)

Configuration requirements are quite flexible regarding the version of all the software mentioned above. Modules and software used during my installation were often more recent than those indicated in the installation guide. I had a problem with the version of

CPAN. I had to upgrade it to version 1.8802 to be able to install some of the required perl modules.

System requirements depend on the number of requests that postgresql, perl and apache will have to serve. I did not find any quantitative analysis of the performance of a deployed AHE server. This kind of analysis could be useful if the AHE server will be considered for adoption in the NGS. This would also be the starting point to test the stability of the overall AHE/gridsam client/server architecture.

The installation process, as described by the installation guide, is not compatible with any previous installation of postgresql and apache. All the required software is assumed to be compiled from the source and installed into the AHE server's home directory. The installation guide suggests a similar approach also for the perl distribution. Fortunately, a script for setting automatically the perl path is given for those who decide to use the system-wide installation of perl.

The AHE server installation does not use any previously installed software or library (apart, obviously, a linux OS). This 'from-scratch' approach to the installation of the AHE server and related software makes irrelevant the Linux distribution under which the tar.gz will be installed but opens the possibility for conflicts with previously installed components. It seems that the implicit assumption of the designers is that the AHE server will run on a more or less dedicated machine.

Installation requires the definition of 17 environment variables. 3 must be defined at the beginning of the installation. The other 14 are stored into a config file. This file is upgraded during the step-by-step configuration process of the AHE server and its various required software.

The apache server is configured 'by default' to listen to port 8000 and 8443 (SSL). These two ports must be open in the firewall so to allow the users to connect with the AHE server. Ports are defined at configuration time and can be arbitrarily chosen.

The process of installation of the AHE server version 1.0.1 is laborious but the situation got better with the 10.2 version included into the OMII stack. The installation process has been simplified with a greater degree of automation. The whole installation can be done in 29 steps, a lot of which consist in pressing 'return' so to accept the default configurations.

The configuration process of the AHE server 1.0.2 starts with setting up the Java keystore for both the OMII and the AHE containers. The situation of the perl modules is still somewhat cumbersome. All the relevant modules have been included into the OMII installer. To avoid problematic overlapping with the perl installation already present, the modules installed with OMII (12) must be manually removed.

Once again this ‘dirty trick’ shows clearly the limitation of a monolithic, independent installation system. An installation based on common binary packaging (rpm or deb) would make the whole process much more flexible. There would be a clear cut separation between application installation and their configuration. The management of the configuration could be better automated, especially at upgrade time, using tools like debconf.

With the inclusion into the OMII stack the process of installation of the AHE server has become acceptable but far from ideal. The AHE server and client have been configured with detailed step-by-step instructions in less than one working day. The fine-tuning of the service and the addressing of errors in the communication between client and server, or server and gridsam instance, requires a deep understanding of the whole architecture. This is definitively a weakness of the overall architecture that would greatly benefit from a set of automated tests to validate and monitor an AHE server installation.

The question remains open of what the target of this installation is. The idea of a fully independent installation process seems to go towards a model in which power users install in their own home directory the whole AHE architecture. This approach could rise both methodological and security concerns. A security audit of the code and of the security model of the AHE architecture should address precisely those concerns.

Client Deployment

The client of the AHE distribution is aimed to be a lightweight application. The main idea behind the AHE design is to give to the user a simple and efficient interface for performing resource discovery and job submissions, monitoring running jobs, querying registry of running jobs, staging files to and from resources, terminating and destroying jobs.

The client is coded in Java and it requires Java 1.4.2 or above. When installed on a Linux OS, it is better to use the Java suite offered by Sun and not the Java suite offered via rpm packages as some problems have been reported. The client has been tested on Linux and OSX and can be installed in user space.

The configuration required by the user to deploy the client is minimal. The user has to set a variable and to create the Java keystore with the provided script. Once started, the client can be configured with the set of parameters given by the AHE server admin.

Parameters are:

- The web address of the file staging area
- User/Password for the file staging area
- The CA certificate to add to the Java keystore of the client
- The user certificate

The configuration can be done with a minimal understanding of the meaning of the parameters involved. The client does not require the user to install any component of the Globus toolkit. A class of about 15 users without knowledge of the AHE client configuration and with little knowledge of the grid infrastructures took no more than one hour to have the AHE client up and running starting from the downloading of the tarball.

The AHE client communicates with the AHE server via the one or two ports on which tomcat is configured to run. These are the only ports that must be open to allow the client/server communication.

The tested client (version 1.0.1) is fairly stable. There is a recurrent bug with the tables of the monitored jobs. Tables tend to close when clicked. When the tables are opened a second time they do not crash anymore. We observed one incident in which the client froze for no evident reasons.

Account Management

Security

The architecture of the AHE environment is depicted in figure 1

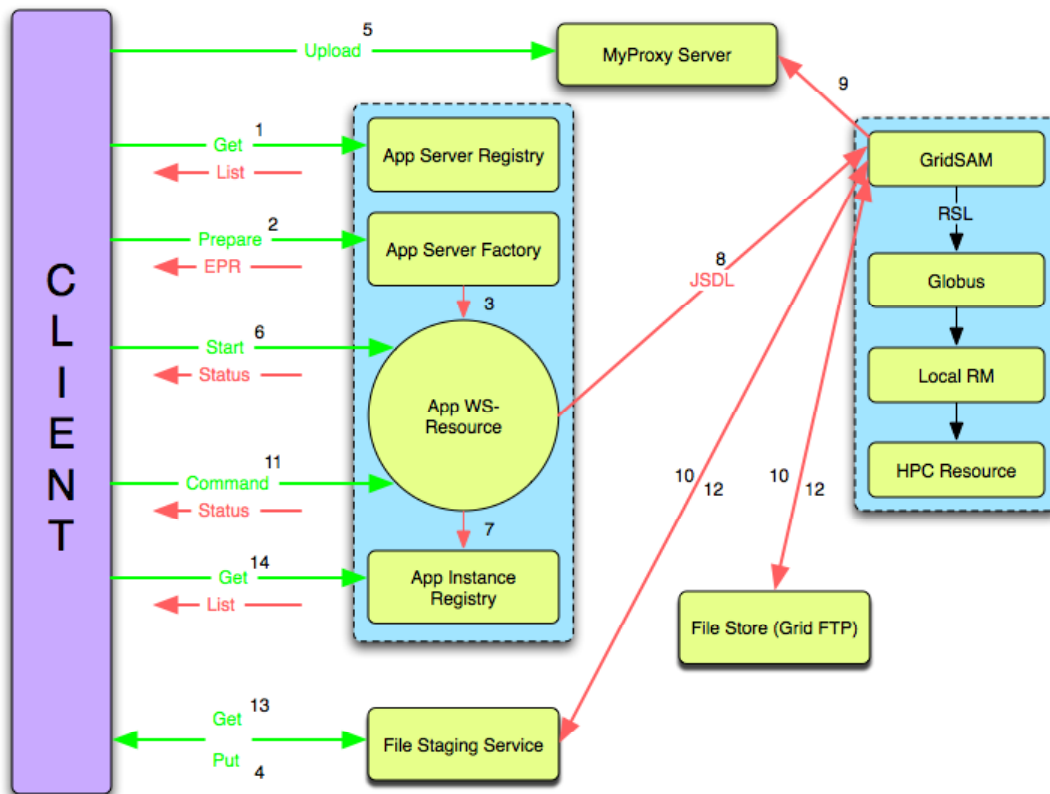


Figure 1 – AHE Architecture diagram

Authentication happens between:

1. AHE client -> AHE server: mutual authentication via certificates
2. AHE client -> NGS node: proxyserver
3. AHE client -> AHE server's file staging area: webdav

Case 1 - AHE server <-> AHE server

The host certificates and the CA root certicate(s) are loaded into the java keystore of the AHE server/OMII installation. The keystore is protected by a password that is stored, in clear, in a configuration file. The password is stored into this file because the web server (tomcat) needs to operate, non-interactively, the mutual authentication with the AHE-

client. It would be worth to investigate the possibility of encrypting this password. As minor consideration, passwords are not hidden from the screen when defined interactively at installation time.

The host unencrypted key and the host certificate are both stored into the \$OMII_HOME directory three in .pem format. The usual permission restrictions are applied to these files. On the client side, the CA certificate is installed into the Java keystore of the AHE client. In this way, the AHE client can trust the AHE server.

Case 2 - AHE client <-> NGS

The NGS node authenticates the AHE client with a certificate generated by a certificate proxy server. The AHE client can be configured to automatically generate this certificate via proxy server at start-up time. The AHE server does not 'know' if the client that connects is a legal grid user. This authentication is left entirely to the NGS node (gridsam instance). The authentication via a proxy certificate is the standard procedure adopted by the NGS so, in this respect, the AHE architecture seems fully compatible with the current NGS authentication policy.

Case 3 - AHE client <-> AHE server's file staging area

The AHE server may be configured to restrict the access to its file staging area that is where the outputs of the submitted jobs are collected. The authentication is done via a pair username/password stored, in clear, in a configuration file. This method of authentication is probably insufficient if the jobs submitted via the AHE architecture produce sensitive data.