PROCEEDINGS OF SCIENCE

EMI Execution Service implementation in ARC

Aleksandr Konstantinov*
University of Oslo, P.b.1048 Blindern, N-0316 Oslo, Norway
E-mail: aleksandr.konstantinov@fys.uio.no

Martin Skou Andersen
Niels Bohr Institute, Blegdamsvej 17, 2100 København Ø, Denmark
E-mail: skou@nbi.ku.dk

The Advanced Resource Connector (ARC) Grid middleware including its Computing Service A-REX was designed almost 10 years ago, and has proved to be an attractive distributed computing solution. Historically ARC was using proprietary interfaces for managing computational jobs on remote clusters. With time as coverage of various Grid middlewares expanded interoperability became important. Few attempts have been made to address this issue. One of them was adoption of OGSA Basic Execution Service (BES) standard. Unfortunately it proved to be not covering enough functionality for production use. Various implementations provided by different Grid middlewares - including ARC - solve that problem by implementing different proprietary extensions hence breaking idea of compatibility. Recent development of EMI Execution Service interface (EMI-ES) specifications produced much richer and almost production-ready definition which takes into account requirements of 3 participating middlewares - gLite, Unicore and ARC. To prove its usability and to test its completeness and interoperability capabilities it is important to provide few implementations. This paper represents the implementation of the EMI-ES interface made in the A-REX service.

EGI Community Forum 2012 / EMI Second Technical Conference
26-30 March, 2012
Munich, Germany

*Speaker.

PoS(EGICF12-EMITC2)081

1. Introduction

Historically not so many efforts for standardizing computational job control interfaces can be identified in Grid community. Both Local Resource Management Systems and their Grid layers are mostly adopting proprietary solution. And although the tendency in Grid world is to use Web Service technology that still only provides communication layer leaving semantics incompatible.

The most notable effort to provide true standard among Grid computing services is OGSA Basic Execution Service (BES)[2] and corresponding job description language JSDL[3]. This standard was adopted by multiple Grid middlewares. Among them UNICORE[4], ARC[5], gLite[6], etc. But interoperability tests show only very basic functionality achieved between different implementations. The OGSA BES provides only very basic functionality and its extensions being developed by various groups simultaneously are tuned for different aims and overlap. Also standards development procedure is usually slow and not consistent with much quicker life-cycle of production middleware.

The plethora of non-compatible interfaces is driving development of client applications and frameworks capable to communicate using multiple interfaces. Among those CondorG [7] can be mentioned as one of oldest. ARC also provides set of client side plug-ins for communication with 2 own proprietary interfaces, OGSA BES-enabled[8] CREAM[9] service of gLite and UNICORE atomic services[14].

This complex situation raised effort backed by 3 middlewares within European Middleware Initiative project[10] to develop common interface with rich enough capabilities, flexible and with fast development cycle. The EMI is a project developing a software platform for high performance distributed computing. The EMI Grid middleware is used by scientific research communities and distributed computing infrastructures all over the world including the Worldwide LHC Computing Grid[11]. The EMI-ES is one of its core activities.

2. Description of interface

This new interface was called EMI Execution Service (EMI-ES) [12] and defines non-strict super-set of functionalities from production versions of all participating middlewares. The distinctive features of new interface - especially if compared to the BES - are:

- Quick development cycle tied to development of EMI components allows for almost immediate response to needs of participating middlewares.
- Integrated support for data pre-staging and post-staging with flexible control for staging options. Staging of specific data is conditional depending on job request and outcome of computational job.
- Support for delegation of client credentials to service closely linked to data staging functionality. It allows for different delegated credentials to be assigned to different input and output data.
- To accommodate for various implementations of job processing life-cycle EMI-ES defines only basic job state cycle. And states related to supported functionalities - data staging, fail-

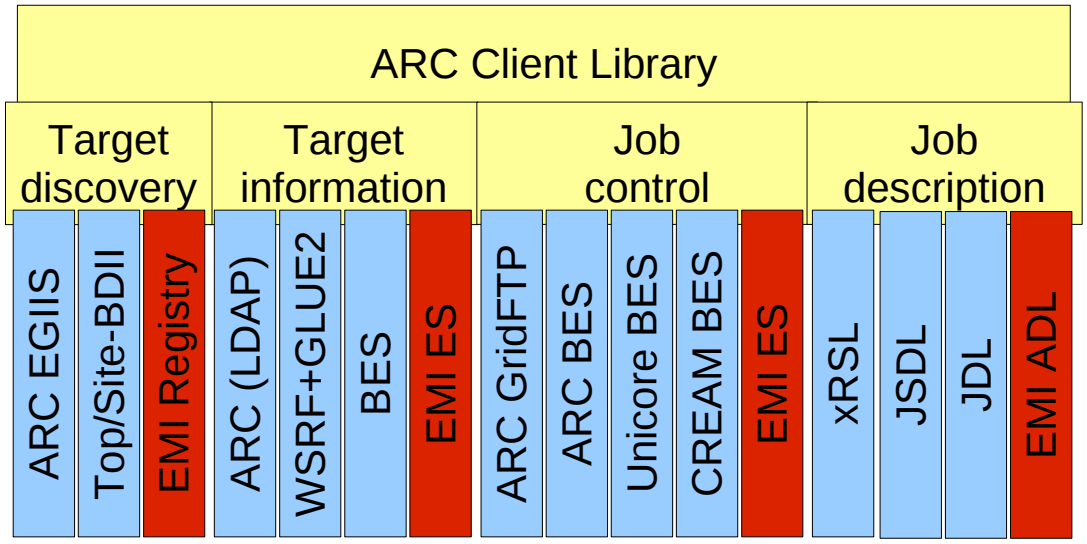

Figure 1: Architecture of ARC client

ure reason, etc. - are represented by state modifiers which are only loosely coupled to specific states. This allows for greater flexibility in how and at which state specific functionalities are implemented.

3. Implementation

The EMI provides three implementations of computing services offering different set of features and suited for different purposes - A-REX, CREAM[13] and UNICORE/X[14]. All three are implementing EMI-ES interfaces as part of their EMI participation.

The ARC middleware implements EMI-ES interface both at client and service side. The client part of ARC is a library - called *ARC client library* - implemented in C++ with a set of dynamically loadable plug-ins each representing specific module for communicating with computing service, indexing service or processing job request language. This modular design allows for seamless addition of new interfaces and functionalities. For describing computational job internally ARC uses structures initially based on JSDL and later significantly extended. The OGF[1] standard GLUE2[15] model is a base for representing computational services internally. Significantly extended from their prototypes, those structures are able to express wide range of applications. This coincides well with EMI-ES interface because it also chooses GLUE2 for representing service capabilities.

The figure 1 shows currently available set of modules implemented for the ARC client library. 3 additional modules were designed as part of EMI-ES implementation - EMI job description language (EMI ADL) handling module, EMI-ES computational job control module and EMI-ES service information query module. Also for discovering EMI-ES service as part of Grid infrastructure service discovery module which can talk to new EMI Registry service EMIR[16] was also imple-

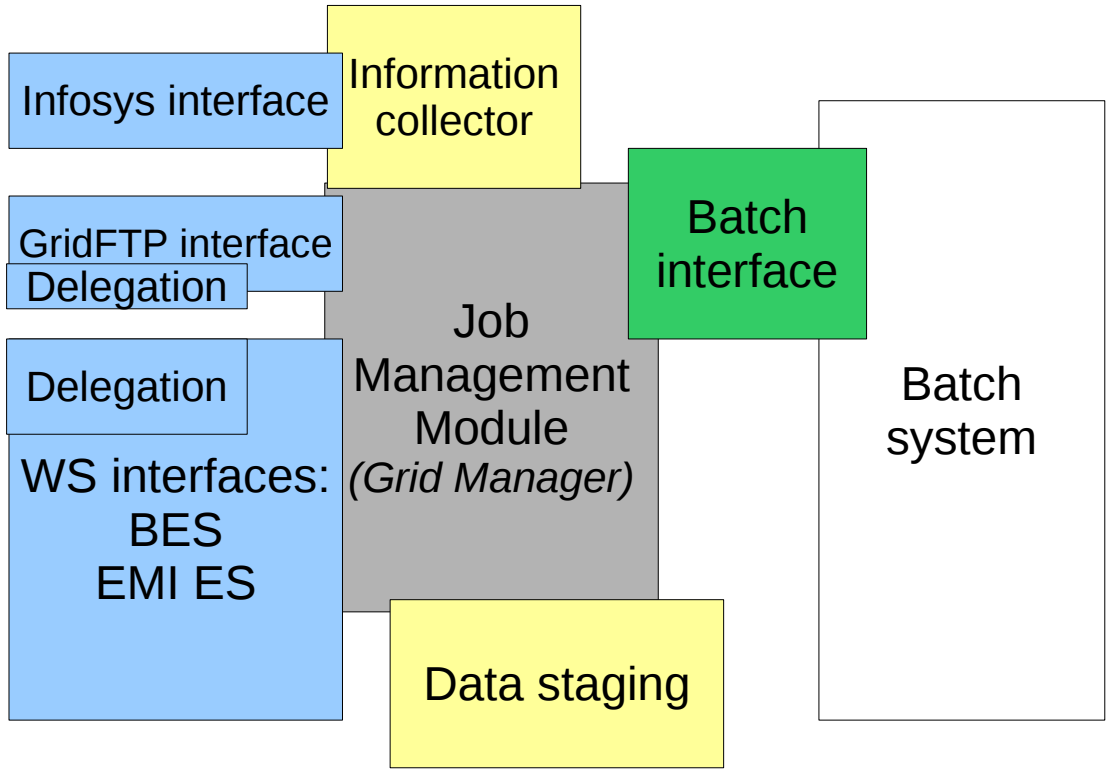

Figure 2: Architecture of ARC computing service

mented as part of the library. In this way ARC client library provides full solution for handling computational activities within Grid infrastructure based only on common agreed EMI interfaces.

On service side ARC implements EMI-ES interface as part of A-REX service. The ARC Computing Service A-REX[17] also has modular design. Different parts of functionality are split over multiple modules as seen on figure 2. Communication part is handled by WS framework of ARC with job processing implemented as separate module known as Grid Manager. Data staging is done in another reusable module which provides balanced data download/upload functionality[18]. And credentials delegation is implemented as a dedicated library of C++ classes which provides bridge between communication and job control modules. Such separation makes implementation of new interface relatively simple task. It also allows to have multiple same and different kinds of interfaces active simultaneously. That makes it possible even to deploy experimental interfaces on production systems hence providing possibility to test new interface in real production environment.

Currently neither the client nor the service parts provide full EMI-ES implementation. Missing functionality is mostly related to a way A-REX processes jobs internally. There is not so much of it - most notable are multiple prologues and epilogues for executable application and compatible multi-node request support. However some of the EMI-ES specs triggered implementation of new features in A-REX hence enhancing its usefulness. During the implementation process the EMI-ES specification was identified as covering most of the use cases of ARC middleware. Whether

the non-covered functionality is really used in production environment and how it could be implemented in EMI-ES functionality is still to be determined and may become subject for future development of EMI-ES specifications. For purpose of better understanding capabilities and shortcomings of EMI-ES in production environment, ARC community is planning to promote EMI-ES interface to be enabled on sites using ARC middleware.

4. Interoperability

The first interoperability tests showed promising results although not full interoperability was immediately achieved. Besides few technical mistakes in interpreting EMI-ES specifications also more serious shortcomings were revealed. The most important of them is weak definition of service representation in GLUE2 model. That makes impossible for automated tools to properly discover and use EMI-ES services and made interoperability tests a manual work.

5. Status and Conclusion

The implementation of new EMI-ES interface in ARC shows promising results as in some fields it covers even broader functionality than was available before.

Preliminary interoperability tests run between different implementations of EMI-ES client and service show that basic functionality is properly covered by specifications. But interconnection between interfaces constituting service and way service describes itself through GLUE2 model still are undefined. In case of ARC that means that although each plugin implementing part of EMI-ES is capable to communicate to other implementations the whole set failed to perform.

That should drive further development of EMI-ES specifications to cover issues discovered during interoperability tests.

6. Acknowledgment

This work was done as part of the EMI project and partially funded by the European Commission under Grant Agreement INFSO-RI-261611.

References

[1] *Open Grid Forum*, `http://www.ogf.org`

[2] I. Foster at al., *OGSA Basic Execution Service Version 1.0*, **GFD-R.108**, (2008) [`http://www.ogf.org/documents/GFD.108.pdf`]

[3] A. Anjomshoaa at al., *Job Submission Description Language (JSDL) Specification, Version 1.0*, **GFD-R.136** (2008), [`http://www.ogf.org/documents/GFD.136.pdf`]

[4] A. Streit et al., *UNICORE 6 - Recent and Future Advancements*, *Berichte des Forschungszentrums Jülich* **JUEL-4319** (2010), ISSN 0944-2952 [`http://hdl.handle.net/2128/3695`]

[5] M. Ellert et al., *Advanced Resource Connector middleware for lightweight computational Grids*, *Future Generation Computer Systems* **23** (2007) [`doi:10.1016/j.future.2006.05.008`]

[6] *gLite - Lightweight Middleware for Grid Computing*, [`http://glite.cern.ch/`]

[7] J. Frey et al., *Condor-G: A Computation Management Agent for Multi-Institutional Grids*, *Proceedings of the Tenth IEEE Symposium on High Performance Distributed Computing (HPDC10) San Francisco, California*, (2001) [http://research.cs.wisc.edu/condor/doc/condorg-hpdc10.pdf]

[8] *BES-Enabled CREAM*, http://grid.pd.infn.it/omii/cream-bes

[9] P. Andreetto at al., *CREAM: A simple, Grid-accessible, Job Management System for local Computational Resources*, *Proc. XV International Conference on Computing in High Energy and Nuclear Physics (CHEP'06)*, (2006), ISBN 10:0230-63017-0, ISBN 13:978-0230-63017-8.

[10] *European Middleware Initiative*, http://www.eu-emi.eu

[11] *WLCG - Worldwide LHC Computing Grid*, http://wlcg.web.cern.ch/

[12] B. Schuller at al., *EMI Execution Service Specification*, https://twiki.cern.ch/twiki/pub/EMI/EmiExecutionService/EMI-ES-Specification_v1.07

[13] *CREAM (Computing Resource Execution And Management) Service*, http://grid.pd.infn.it/cream/

[14] *UNICORE/X*, http://www.unicore.eu/unicore/architecture/service-layer.php#anchor_xnjs

[15] S. Andreozzi at al., *GLUE Specification v. 2.0*, **GFD-R-P.147**, (2009) [http://www.ogf.org/documents/GFD.147.pdf]

[16] *EMI Registry (EMIR)*, https://twiki.cern.ch/twiki/bin/view/EMI/EMIRegistry

[17] F. Paganelli at al., *ARC Computing Element. System Administrator Guide*, **NORDUGRID-MANUAL-20**, (2012) http://www.nordugrid.org/documents/arc-ce-sysadm-guide.pdf

[18] D. Cameron et al., *Adaptive data management in the ARC Grid middleware*, *Journal of Physics: Conference Series 331* **062006** (2011) [http://dx.doi.org/10.1088/1742-6596/331/6/062006]