GFD-E.179
Mariusz Mamoński,
Poznan Supercomputing and Networking Center, Poland.

HPCP-WG, OGSA-BES-WG

March 20, 2011

Smoa Computing HPC Basic Profile Adoption – Experience Report

Status of This Document

This document provides information to the Grid community about the adoption of the OGF specifications: GFD.114, GFD 108, GFD.135 in the Smoa Computing service.
It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Trademark

Abstract

This document describe experience gained while implementing the Smoa Computing Service [1], a component which development was steered by the following OGF specifications:

- GFD 108 - OGSA® Basic Execution Service Version 1.0 [2]
- GFD.114 - HPC Basic Profile, Version 1.0 [3]
- GFD.135 - HPC File Staging Profile, Version 1.0 [4]

In addition to comments on those specifications, this document presents possible extension to the Basic Execution Service: a separate interface for managing Advance Reservations based on the BES-Factory port.

Contents

1. Introduction

In the mid 2005 the OpenDSP (Open DRMAA Service Provider) project was launched [5] at Poznan Supercomputing and Networking Center (PSNC). The main goal of the project was to develop a service giving consistent, remote, multi-user access to various DRM systems using standardized DRMAA interface in the layer between the DRM system and the service. Leverage of the DRMAA [6] C binding and usage of the C language in the core modules was expected to result in high performance of the service. Although the OpenDSP service could accept job described in the JSDL [7], the remote interface of the service was self-designed, and thus it is not a part of any standard.

After more than two years, and four successive releases of the OpenDSP, a decision was taken to update the remote interface, and exploit another OGF standard: the OGSA Basic Execution Service (profiled by the HPC Basic Profile specification). With the new interface also the other aspects of the service have changed:

- refined architecture,
- privilege separation instead of setuid binaries,
- JSDL used as the sole format for internal job representation,
- support for modules written in Python,
- more extensions points added,
- exposed Advance Reservation capability of underlying DRMS,
- file staging support as the part of job life cycle,
- separate interface for lightweight, direct file staging (via SOAP attachments).

Also a new name was given to the service: Smoa[1] Computing, as the DSP acronym was quite often confused with Digital Signal Processing.

[1] Smoa is not an acronym. It is just a name of the magic land where, as the tale says, every job finds its eternal happiness.

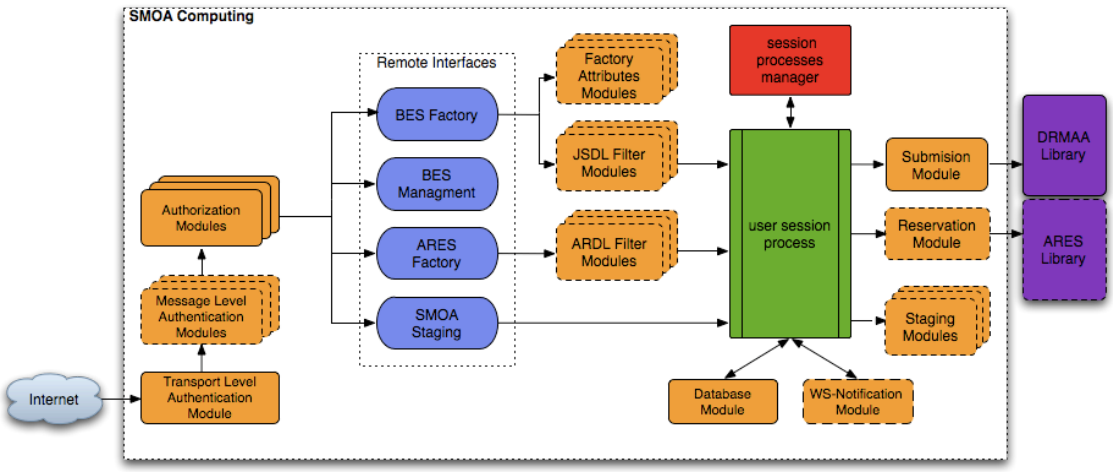

Figure 1 Smoa Computing architecture

The above diagram depicts the overall Smoa Computing Architecture. The service interface is composed of four Web Service ports:

- BES-Factory - interface for job creation, monitoring and management [2],
- BES-Management - interface for managing the service [2],
- ARES Factory - interface for advance reservation creation and management - a Smoa Computing extension (described in section Advance Reservation Interface),
- Smoa Staging - interface for direct (client-service) file transfer via SOAP attachments - a Smoa Computing extension.

The system architecture is based on dynamically loadable modules as depicted on Figure 1. The descriptions of various modules types are listed below:

- Transport Level Authentication module – selects protocol used in the transport layer. Supported protocols are HTTP, HTTPS, HTTPG and XMPP.
- Message Level Authentication module – turns on message level security. Currently supported mechanisms are UsernameToken and SAML assertions.
- Authorization module – decides whether an authenticated user (identified by some Distinguished Name provided by the authentication module) can access the resource. This type of module should also provide local account mapping. Some of the possible authorization mechanisms are plain grid-map file, callout to external authorization service or fixed mapping to local account.
- Factory Attributes module – this kind of module can be used to provide additional information about the cluster accessible via the GetFactoryAttributesDocument call.
- JSDL Filter module – intercepts and alters incoming JSDL documents.
- ARDL Filter module – intercepts and alters incoming ARDL (Advance Reservation Description Language) documents.
- Staging module – a module for handling Stage-In/Stage-Out request described in JSDL documents. Currently Smoa Computing has libcurl-

based and Amazon Simple Storage Service (S3) staging module implemented.
- Database module – handle connectivity with the database backend.
- WS-Notification module – responsible for sending job status notifications to WS-Notication 1.3 compliant notification brokers.
- Submission module – a module for submitting, monitoring and controlling of batch jobs. Currently Smoa Computing has only one submission module which leverages DRMAA interface to communicate with various batch systems.
- Reservation module – a module for advance reservations creation, monitoring and control.

The service implementation follows the privilege separation model. Thus only a relatively small part of the overall system (the Session Process Manager in Figure 1) runs with superuser privileges. This component is responsible for creation of new processes (called User Session Processes) that run with mapped users effective privileges.
The Smoa Computing service was successfully tested with the following DRM systems:
- Sun Grid Engine,
- Platform LSF,
- Torque,
- PBS Pro,
- Condor,
- Apple XGrid,
- SLURM.

2. State model

The Smoa Computing service took advantage of the BES extensible state model, and provided its own states specialization, which introduced 6 sub-states:
- *Stage-In* - The input files are being staged in.
- *Stage-Out* - The output files are being staged out.
- *Suspended* - The job was either suspended by user or system.
- *Held* - The job was held in a queue.
- *Queued* -The job is waiting in a DRMS queue (meaning inherited from the *Pending* state)
- *Executing* - The job is actually running on the execution host (meaning inherited from the BES Running state)

The Smoa Computing state model is presented on Figure 2.

What might seem to be peculiar is that the *Stage-In* state is a specialization of the *Queued* state instead of the *Running* state (as suggested in the HPC File Staging profile). The rationale behind this model (in the Smoa Computing use case) was that the service is always deployed on top of the existing queuing systems, and for this reason it must stage all input files before submitting a job to the local system's queue.

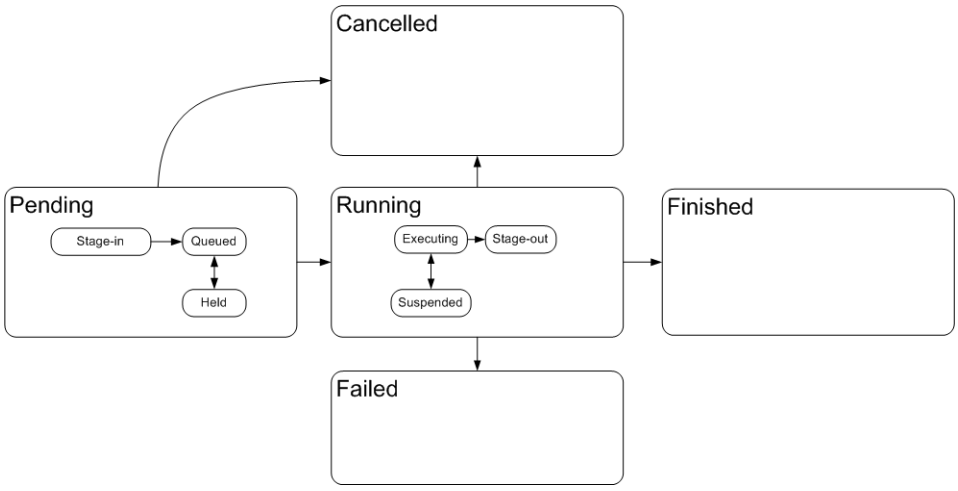

Figure 2 BES state model specialization in Smoa Computing

3. Mapping between JSDL and DRMAA

The Smoa Computing service for the job submission, control and management use only DRMAA interfaces. Thus it was crucial to map all the JSDL elements that are marked as mandatory by the HPC Basic Profile to the corresponding DRMAA 1.0 Job Template attributes. The mapping used by the Smoa Computing service is presented in the below table:

JSDL element name	DRMAA attribute name
JobName	DRMAA_JOB_NAME
Executable	DRMAA_REMOTE_COMMAND
Argument	DRMAA_V_ARGV
Environment	DRMAA_V_ENV
WorkingDirectory	DRMAA_WD
Input	DRMAA_INPUT_PATH
Output	DRMAA_OUTPUT_PATH
Error	DRMAA_ERROR_PATH

Values of the other JSDL elements (e.g. TotalCpuCount) are communicated to the underlying batch system via DRMAA_NATIVE_SPECIFICATION Job Template attribute. The translation to the native options is handled in the batch system specific fashion, by so-called JSDL Filter module (e.g. *lsf_jsdl_filter* for the Platform LSF).

4. OGSA-BES/HPC-Basic Profile specifications comments

- The OGSA-BES specification provides quite comprehensive set of possible faults. Many of the faults are defined as complex elements (e.g. the *InvalidRequestFault* requires that every invalid element is listed by name in the fault message). This leads to a more complex code of the error handling on the service side and displaying error message on the client side. We fully understand that such fine grained approach helps if detailed information about the error state is needed by a consuming systems, but in our case we could not find any use case for it. Moreover it would be convenient to have all the fault messages definitions extracted into separate schema document (with separate namespace), so it could be easily reused in the other WS ports (i.e. in our case we have the *NotAuthorizedFault* duplicated four times).
- Although the BES-state model is very extensible through the state specialization it is very strict (for interoperability reasons) in the transitions between basic states that are legal. From our experience we found two missing transitions in the BES basic state model:
  1. Transition from *Pending* to *Failed* state - as in our specialized model the *Stage-In* state is a sub-sate of the *Pending* state we encountered problem how to react in our system upon failure on staging input files. Eventually we decided to emit in this case *Running* state notification followed immediately by the *Failed* notification.
  2. Transition from *Running* to *Pending* state - this transition would address more advanced scenarios, like rescheduling jobs upon resource failure. This scenario could also be addressed in the current model by introducing another "re-queued" sub-state of the *Running* state, however we found this redundant.
- It would be helpful if the next version of the OGSA-BES specification would address explicitly, as an optional extension, how to handle parametric sweep jobs. In particular it could define how aggregated status about such activity should be provided.
- The XML document returned by the *GetFactoryAttributesDocument* operation may be quite heavy in case where BES service manages a cluster composed of thousands of nodes. Some more flexible way, over the 'BasicFilter' extension, could be provided in order to limit size of the response message in such cases.
- The HPC Basic Profile states that the support of the *ExclusiveExecution* sub-element of the JSDL document is mandatory by the compliant implementation. However such capability is not widespread among

existing batch systems (e.g. it is not available in Torque, Grid Engine supports it only since version 6.2 Update 3)
- On the other hand the *HPCProfileApplication* is missing other commonly implemented and widely used job attributes: the wall clock time limit and the batch queue name.

5. Advance Reservation interface

The interface of the Advance REServation Factory (ARES Factory) port was influenced by the design of the BES-Factory port. The ARES Factory port is composed of the following 4 operations:
- CreateReservation - Requests to create a new advance reservation. A requested reservation is described in the Advance Reservation Description Language (described in the following section) document. On success an EPR of the newly created advance reservation is returned.
- GetReservationDocument - Returns the ARDL document of the advance reservation.
- GetReservationStatus - Returns list of resources booked by the advance reservation and list of associated computational activities.
- GetActiveReservations - Returns list of EPRs of all advance reservations that are in the system.

6. Advance Reservation Description Language

Similar to the ARES Factory the Advance Reservation Description Language (ARDL) was modeled upon another OGF standard: the Job Submission Description Language (JSDL) specification. An example ARDL document is presented in the below listing:

```
<ardl:ReservationDefinition>
   <ardl:ReservationDescription>
      <ardl:ReservationIdentification>
        <ardl:ReservationName>SampleReservation</ardl:ReservationName>
     </ardl:ReservationIdentification>
     <ardl:TimeWindow>
        <ardl:StartTime>2010-03-21T11:00:00+01:00</ardl:StartTime>
        <ardl:EndTime>2010-03-21T15:00:00+01:00</ardl:EndTime>
     </ardl:TimeWindow>
     <ardl:Resources>
        <ardl:ReservedSlotsCount>1</ardl:ReservedSlotsCount>
        <ardl:UserName>jsmith</ardl:UserName>
     </ardl:Resources>
  </ardl:ReservationDescription>
 </ardl:ReservationDefinition>
```

This document describes request for creating an advance reservation:

- bearing human readable name SampleReservation,
- starting on 11.00 (CET) 21st March 2010,
- ending on 15.00 (CET) 21st March 2010,
- for one slot (which usually corresponds to one cpu core),
- with Access Control List set to local user jsmith.

7. Security Considerations

Security issues are not discussed in this document. For Security Consideration of the BES services consult the respective section of the GFD.114 document.

8. Conclusions

We found the Basic Execution Service specification as step forward in making grids more interoperable. The OGSA-BES acting on the Web Service interface level is complementary to the API approaches found in the DRMAA and SAGA specifications. In addition the HPC Basic Profile effort in profiling the OGSA-BES and JSDL specifications was very essential, as it clarified on a basic subset of functionality to be offered by Basic Execution Service, thus facilitating development of a Basic Execution Service interoperable with other vendors implementations.

9. Contributors

Piotr Domagalski,
Mariusz Mamoński,

Poznan Supercomputing and Networking Center
Noskowskiego 10 Street,
61-704 Poznań,
Poland

10. Acknowledgments

We would like to thank especially Piotr Domagalski, who was the former leader developer and architect of the Smoa Computing service.

11. Intellectual Property Statement

permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director.

12. Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

13. Full Copyright Notice

14. References

[1] Smoa Computing, http://gforge.man.poznan.pl/gf/project/Smoacomp/
[2] GFD 108 - OGSA® Basic Execution Service Version 1.0, http://www.ogf.org/documents/GFD.108.pdf
[3] GFD 114-HPC Basic Profile, Version 1.0, http://www.ogf.org/documents/GFD.114.pdf
[4] GFD 135 HPC File Staging Profile, Version 1.0, http://www.ogf.org/documents/GFD.135.pdf
[5] Open DRMAA Service Provider, http://sourceforge.net/projects/opendsp/
[6] OGF DRMAA Working Group, http://www.drmaa.org/
[7] GFD 56 - Job Submission Description Language (JSDL) Specification, Version 1.0 - http://www.gridforum.org/documents/GFD.56.pdf