Federated Models For Monitoring Data Access

Y. Chen[‡], K. Farkas[‡], R. Liu[§], D. Milojicic[‡], S. Rafaeli[†], K. Saikoski[†], V. Talwar[‡], W. Vambenepe[*]

HP Labs[‡], HP Brazil[†], HP Technology Solutions Group[*], Tsinghua University[§]

[firstname.lastname]@hp.com, rliu@tsinghua.edu.cn

Abstract

Model-based automation is becoming an important approach in managing IT, resulting in a diverse set of model repositories using possibly different model schemas and located within one or more different administrative domains. For effective automation, management tools typically require access to multiple repositories, and thus, mechanisms are required to integrate the data models confined to administrative, geographic, and model schema boundaries. In this paper, we introduce two such mechanisms, one for maintaining the links between repositories, and the second for controlling access to the repositories. To illustrate the use of these mechanisms, we describe how we are applying them to the problem of providing access to the data collected about jobs in a Grid computing environment. This discussion demonstrates the value of our approach.

Keywords

Model, Federation, Monitoring, ChinaGrid, Web Service

1. Introduction

Service Oriented Architectures (SOA) and geographically distributed data centers have resulted in composite services comprised of multiple independent services running within different data centers and under different administrative domains. Monitoring and consequently managing these composite services requires common management interfaces and common understanding of the management data (metrics and topologies). Common interfaces are enabled by management standards, such Web Services Distributed Management (WSDM) and WS-Management, while metrics and topologies are described in data model schemas, such as Common Information Model (CIM), Resource Description Framework (RDF) and GLUE. Because different data centers and services support different architectures managed by different management systems, there is a need for federation of data models.

In this paper, we examine the federation of data models in the context of monitoring a distributed IT environment comprising multiple administrative domains. A challenge

in such contexts is to build management tools that can access the monitoring data collected for the components of the IT environment and published by a heterogeneous collection of data repositories. Data models offer the ability to capture in a machine-readable form the meta-data that describes the context under which data is collected, how the data is represented, and properties about it. For example, a model could be used to capture the metrics that are available for each application running in a cluster, the definitions of these metrics, and the units. Or, a model could be used to describe the structure of an application or service, the metrics that are available for each component in the structure, the corresponding metric definitions, and how to access their values. However, to simplify accessing these models, they must be federated. Federating such models presents several challenges. First, the models are distributed across domains and must be stitched together in a loosely-coupled manner. Second, each model has its own schema and representation format. Third, trust and security issues need to be considered when traversing across domains.

We make the following contributions in this paper

- We present an architecture for federated model-based data access and propose a *link* mechanism to stitch different models together. This link mechanism is implemented as a *federated object* and we identify the attributes needed for this object.
- We present a federated identification mechanism for controlling access to models.
- We illustrate how this architecture may be applied to making monitoring data available from multiple domains in a Grid.

The rest of this paper is organized as follows. In Section 2 we present the design of our model federation solution. Section 3 describes its application to a Grid environment. We review competitive approaches in Section 4 and finally conclude with next steps in Section 5.

2. Design

In this section, we describe our approach for integrating models from multiple data repositories and how we provide for federated identities. We then discuss how these concepts would be integrated into a federated-model architecture.

2.1 Federated Model Access

To integrate diverse models, we introduce the notion of a *federated object*, which provides additional information about how to access a resource from another domain. This object can be used to navigate across models from different domains. We define the federated object to contain the following attributes**:** <*model service endpoint, resource addressing information, model transformation information, security policies, properties*>. This information can be represented as an XML element, as shown in the template below.

```
<FederatedObject>
   <RemoteResourceEPR>
     <!--  a WS-Addressing EPR for the remote resource -->
   </RemoteResourceEPR> ?
   <Model>
      <!-- A URI representing a model used by the remote resource -->
   </Model> +
   <Association>
    <!—Association to other Federated Object -->
   </Association>*
   <!—there could be more than one model -->
   <ModelTransformation source="uri" target="uri">
      <!-- resources to translate from the (or one of the) model of the remote
           repository to another -->
      <Service>
         <!-- EPR to a service that can translate some elements of this model -->
      </Service>*
      <Document>
          <!-- a URL to a document (XSLT, QVT...) to describe a way to translate
               this model -->
      </Document>*
   </ModelTransformation>*
    <xs:any/>*
</FederatedObject>
```

This XML element conveys the Endpoint Reference (EPR) to the remote repository as well as meta-information about potential model transformations required to access the remote repository. This information consists of an identifier of the resource model schema(s) supported by the remote repository as well as any available information about ways to translate these resource models. In general, translating between models is difficult owing to the need to not only translate syntactic differences but to also translate semantic ones. For syntactic translations, approaches such as an eXtensible Stylesheet Language Transformations (XSLT) document may be used. For semantic translations, semantic web technology might be employed or a service that applies a set of pre-defined rules to map on model space to another. Of course, if the remote repository uses the same resource model as the local repository, neither the <*Model*> nor the <*ModelTranslation*> element is used.

The resource addressing information and security policies are specified as part of the metadata section of the remote repository EPR, along with any other meta-information relevant to the use of this EPR. The "resource addressing information" attribute refers more specifically to the ability to make explicit the way in which SOAP headers are used for addressing in the cases where the addressing headers are based on the model of the resource. When a federated link object is referred, the information defined in the federated link object is accessed first, and then the information is used to locate the resource information from certain domain via WS model access interfaces provided by that domain.

Finally, a federated object may encode associations with other federated objects, and hence, a collection of federated objects may be used to describe a model composed of the models referenced by each of the individual federated objects. These associations are encoded in the "association" attribute.

2.2 Security

Model repositories employ access control policies to control access to models. This means that when a customer queries for an instance of a model *m*, he might have only access to the subset of the instance of *m*, that is *m'*, which is a pruned version of *m*. The lack of a single security administrator responsible for authenticating users and managing their access privileges as a single entity raises the need for a federated identification mechanism. Federated identity infrastructures enables cross-administrative *single sign-on* by leveraging a trust relationship among data repositories that allows users authenticated at one repository to access other trusted repositories without re-authenticating. The repositories share information about users performing inter-domain transactions, and such information is used to correctly identify the user in the new domain. The new domain can then enforce its own local access control mechanism. The following illustrates the pruning process.

1. Customer *c* logs at the model federation service
2. He queries for an instance of model *m*
3. The model federation service queries each repository manager *i* for model instance *m*, which is a subset of those available from repository *i*
4. The query is sent with enough information about the customer so that repository manager *i* can identify the customer's local user account.
5. Repository manager *i* finds the requested instance of the model *m*
6. Repository manager *i* verifies authorization access of customer *c* to the instance of *m* and returns the instance

Note that in step 4, the repository manager will map the customer's model federation service ID (e.g., smith@composer) into a completely different ID (e.g., john@reposit1) if the customer is known to each component by different usernames. This mapping is required for each model repository to have its own access control policies and also to have independent identification mechanisms.

Note also that while the above illustrates pruning model instances, the same mechanism could be used to prune the model itself. When models are pruned rather than model instances, a customer will be allowed to obtain all instances of the portion of the model to which he has been granted permission. Hence, instance pruning provides greater selectivity than model pruning.

2.3 Conceptual Architecture

Figure 1 presents a conceptual architecture for the model federation service and shows its relationships with the remote data repositories. The key component of the monitoring service is the model navigator module, which performs the following

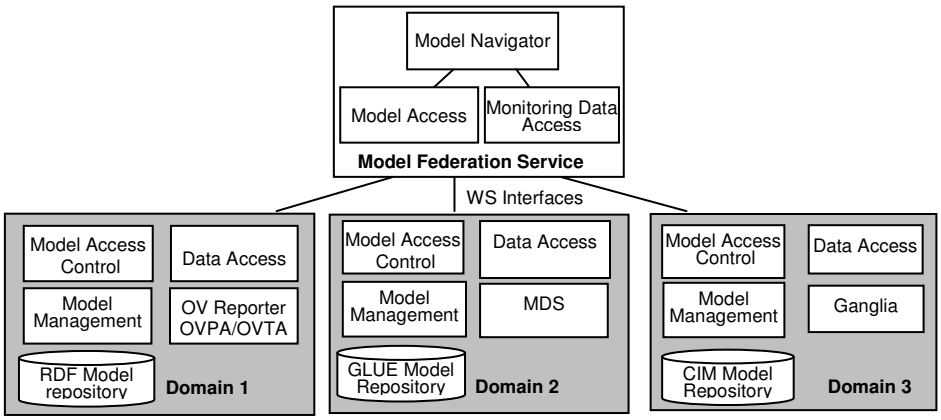

Figure 1: Conceptual Architecture of Model Federation

functions: accessing models using Web services, transforming models, resolving the model federation links, and invoking the data access modules. As a first step, the model navigator parses through the "federated link objects". It then interacts with the Model Access Control (MAC) modules to obtain the models from the individual domains. A MAC module performs user identification and model and model instance pruning. The model navigator sends information about the user making the request, and the MAC maps that information to a local user account. Using the privileges associated to this user account, the MAC prunes the model or model instances before returning it to the model navigator. The model is then transformed and parsed. The parsed topology information can then be navigated through by a user or an automated program. A particular application or system of interest is selected for data access during the navigation. The data access module interacts with the data access on each domain through Web services interfaces to obtain the monitoring data of interest.

3. Implementation

In order to demonstrate our approach, we describe a model federation service prototype we are developing which uses models and *federated objects* to access the monitoring data for jobs running in ChinaGrid [7, 8].

3.1 ChinaGrid Overview

ChinaGrid, created by Chinese Ministry of Education, is one of the largest Grid computing projects in the world. It aims to enable universities across China to collaborate on research, scientific and education projects. ChinaGrid will eventually connect as many as 200,000 students in 100 universities across the country when the project is completed. At present, ChinaGrid connects more than 20 universities and serves on average several thousand job requests per day. ChinaGrid supports an *application service* model wherein applications are statically deployed on multiple

clusters within the Grid, and are invoked as needed to service user-provided jobs. The currently offered application services provide image processing, bioinformatics, massive information processing, fluid dynamics, and remote education tools.

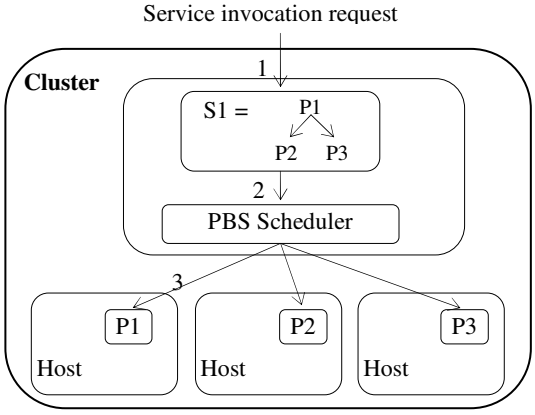

Figure 2: Service invocation workflow in ChinaGrid

In ChinaGrid, to execute a job requires invoking one or more of these services. To invoke a service, the tasks to be done by the service are dispatched to the service by the Grid job scheduler. These tasks are then broken down into one or more processes by the service, and scheduled on the systems belonging to the cluster that is associated with that service. Figure 2 illustrates the service invocation process for a service invocation S1. As shown in Figure 2, service invocation S1 comprises three processes P1-3 running on the hosts inside the cluster to fulfill its tasks.

As noted, a job may comprise multiple service invocations, and, hence, the monitoring data for such a job may be captured and recorded in multiple domains. Without *federated objects* representing the job and providing links to retrieve the model and data, it is difficult for monitoring data consumers to construct the execution scenario of the job. In addition, while different domains may have different access control policy, the model federation service provides a unified interface for monitoring data access.

3.2 ChinaGrid Model Federation Implementation

Figure 3 presents the overview of the federated data access solution we are building to simplify the access to monitoring data in ChinaGrid. In this context, the model federation service provides two roles. First, it provides a local data repository that stores the federated objects associated with each job that has been processed by ChinaGrid and the federated objects associated with each job's sub-jobs. Second, it accepts requests from the consumer for information about these objects, and handles the task of obtaining the requested models and corresponding monitoring data, that is, if the user is allowed to receive them. The user identification information is sent to each domain holding monitoring data. The domain's MAC translates that information to a

local user account and uses the user account's privileges to return only the monitoring data the user is allowed to access. As also shown in the figure, each domain has its own archive service that stores the monitoring data and model that describes its execution environment. However, in ChinaGrid, these models are all derived from a common schema and hence model transformation is not required.

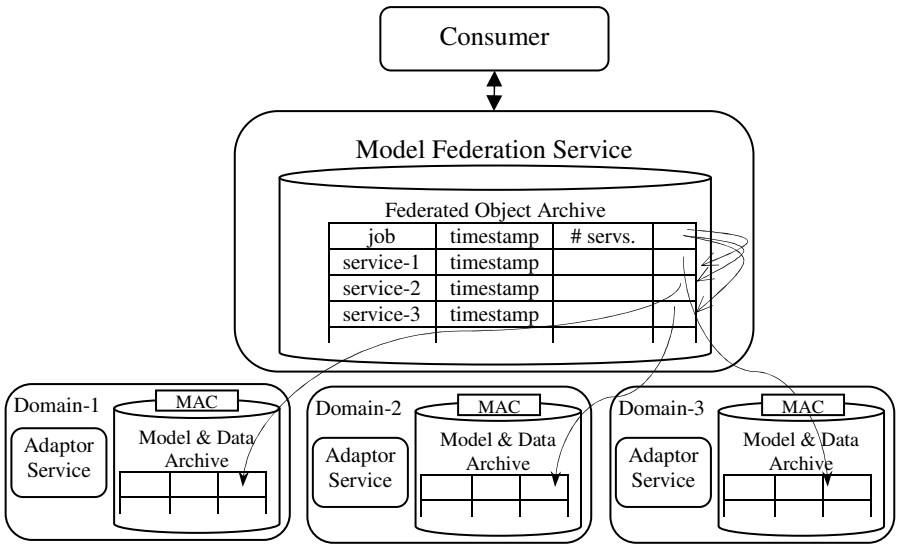

Figure 3: ChinaGrid Monitoring Data Model Federation Prototype

When a new user job is submitted, the model federation service receives the job description from the Grid job scheduler and creates a new *federated object* representing this user job with a timestamp. When a service of this job is launched to run on a cluster server, the model federation service receives the service information from the service provider and creates the corresponding *federated object* for the service. This *federated object* for the service has a link to the service's corresponding manageable resource on the cluster. At the same time, a reference to the *federated object* of this service is added to the job's *federated object*. When all services of the job have been executed, the properties of the job's *federated object* (e.g., number of services) are updated and all federated objects that have been created are stored in the federated object archive.

Through the model federation service interface, a consumer can construct the historical scenario according to *federated object*s and their relationships. Furthermore, the client can navigate the model and data archive of different domains following the links in the *federated object*s and retrieve the monitoring data and model. Figure 4 presents the model exported by the ChinaGrid domains. An instance of this model describes, for a given service, the processes that are associated with it, the hosts on which they are executing, and the cluster. Furthermore, by navigating through this model instance, the consumer can get all the context information for the service execution. For example, the consumer can get the CPU and memory usage of each process which is running for

the service, as well as the CPU load and memory usage of the host and cluster where the process and service are running.

Figure 5 summarizes the relationship between the federated objects and model instances. As shown, a job comprises three service invocations, and the federated object for each invocation provides the link to *service* resource model instance for the service invocation. Consumers may then navigate through related domain specific resources, e.g. *Process, Container etc.,* via the federated service.

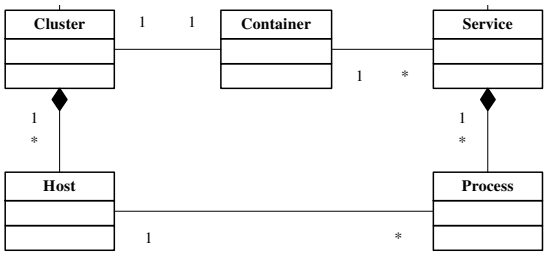

Figure 4: ChinaGrid Domain Model

Figure 6 shows an example of a user job federated object. It includes an *Association* attribute that provides the reference to the related federated objects (e.g., that for Service invocation 1). The federated object may contain property information such as owner and timestamp. The corresponding federated object for Service-1 is shown in Figure 7. It contains links to the remote domain and models as well as an *Association* attribute that points to the job federated object.

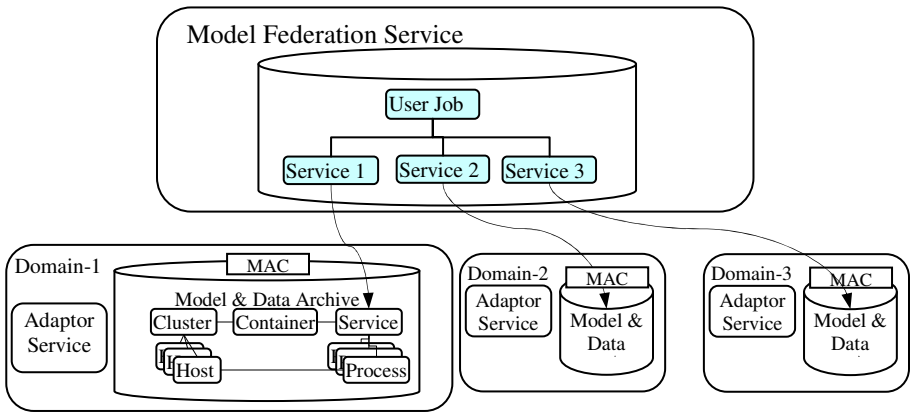

Figure 5: Federated Objects and their Relationship to the Domains for an Example Job Comprising Three Service Invocations

```
<FederatedObject name=”UserJob1”
xmlns:wsa="http://www.w3.org/2005/02/addressing">
   <Association name=”Service1” role=”service”>
            <wsa:EndpointReference>
                     <wsa:Address>
                        <!—URI of Federated Object Service-1 -->
                     </wsa:Address>
                     ……
            <wsa:EndpointReference>
                     …
   </Association>
   <timestamp>2005-3-20 9:20:40</timestamp>
   < owner> Rui Liu </ owner>
</FederatedObject>
```

Figure 6: Federated Object for User Job

```
<FederatedObject name=”Service-1” xmlns:wsa="http://www.w3.org/2005/02/addressing">
<RemoteResourceEPR>
            <wsa:EndpointReference>
                     <wsa:Address>
                        <!—URI of the remote resource -->
                     </wsa:Address>
                     ……
            <wsa:EndpointReference>
  </RemoteResourceEPR>
<Model>
            <wsa:EndpointReference>
                     <wsa:Address>
                        <!—URI of the  model used by the remote resource -->
                     </wsa:Address>

            <wsa:EndpointReference>
</Model>
<Association name=”UserJob1” role=”userjob”>
            <wsa:EndpointReference>
                     <wsa:Address>
                        <!—URI of Federated Object UserJob-1 -->
                     </wsa:Address>
            <wsa:EndpointReference>
</Association>
< owner> Rui Liu </ owner>
<timestamp>2005-10-4 5;20:30 </timestamp>
</FederatedObject>
```

Figure 7: Federated Object for Service-1

3.3 Security Implementation

The Model Access Control module is implemented as a set of sub-modules. The MAC service receives a query. The message carrying the query also carries user identification (credentials or information about the user). This step corresponds to step 1 in Figure 8. Using the identification information, the MAC service obtains the local user account and the privileges associated with it from the authorization module (steps 2 and 3). The MAC service then queries the repository (step 4). The query returns the data requested and the authorization policies associated with the data (step 5). The MAC service sends the data, user privileges and authorization policies to the filter module (step 6) that prunes the data. The pruned data is then returned to the entity performing the query (step 7).

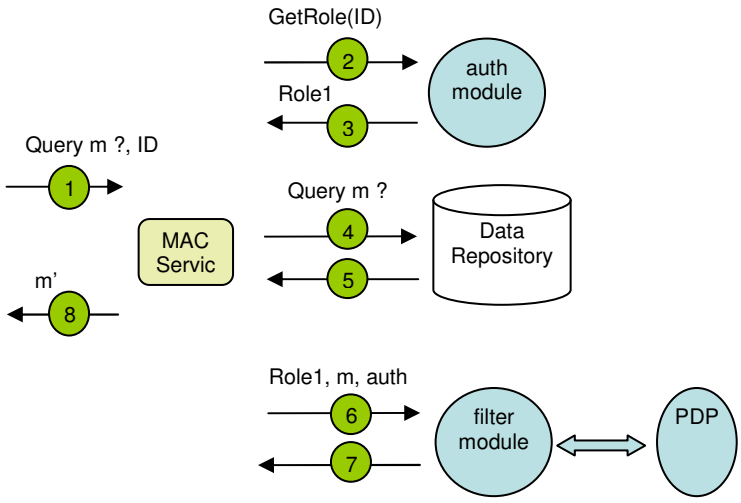

Figure 8: Model Access Control Module

User information is exchanged between repositories using the OASIS Security Assertion Markup Language (SAML) [2]. SAML is an XML standard for exchanging authentication and authorization information. It guides how identity-related information can be exchanged between communicating parties. SAML standardizes the representation of credentials in an XML format called assertions. An assertion is a declaration of facts (statements) about a subject. The assertion conveys the means used to authenticate the user and the set of attributes relevant to the data provider to correctly identify the user.

We use the OASIS standard eXtensible Access Control Markup Language (XACML) [3] for providing access control. XACML is a general purpose policy system. It defines the syntax of a policy language and the semantics for processing those policies. A filter parses the model requested by the user and checks the access permission of each element of the model against the user's permission and XACML policies. Elements not

authorized are removed from the model. On the left, a simplified example of a XACML policy file where customers belonging to group *users* are allowed to *read* resource *system*.

4. Related Work

In an ideal world, there would be only one standard, integrated model and all resources and services in an enterprise would adhere to it. Unfortunately, there are too many competing standards. Not even the next best approach – a unified, common model schema is possible. Therefore, the federation is a requirement.

The need to federate models is a critical piece of efforts to move IT management towards an ITSM (IT Service Management) approach. Most of the IT management software in use today maintains some kind of model, in many cases proprietary. Integrating assets to create reusable IT services requires seamless access to data currently help in these various repositories. Many such efforts are in progress. One example is integration of the OpenView model for configuration management with OpenView server management to create services for automated provisioning. Other examples related to automation require integration with the OV ServiceDesk, again, through model federation. In the IT Infrastructure Library (ITIL) methodology, a configuration management database (CMDB) is used as the federation access point.

Model federation approach was used in distributed simulation for the needs of the Department of Defense [4] and it was also applied to economic simulation [5]. There is a similarity with component-based software, using readily available software components. Finally, model federation was applied to IT governance [6].

5. Summary and Future Work

We are currently developing a model federation prototyping a federated data access mechanism for ChinaGrid using the access control and federated object mechanisms presented in this paper. We are also working with Intel to exchange models for monitoring data from PlanetLab. As next steps, we are planning to further explore ways for representing the metadata attached to management services and manageable resources, including the metadata listed in the paper. We are planning to explore how to translate between different repository models, focusing on the specific case of data models for monitoring.

References

[1] Housley, R., et al., *"Internet X.509 Public Key Infrastructure Certificate and CRL Profile",* RFC2459. January 1999.

[2] Mishra, P., et al. *"Conformance Requirements for the OASIS Security Assertion Markup Language (SAML) V2.0",* OASIS Standard, 15 March 2005.

[3] Moses, T., XACML 2.0 Core: eXtensible Access Control Markup Language (XACML) Version 2.0. Oasis Standard, 1 Feb 2005.

[4] Nance, R.E., *"Distributed Simulation with Federated Models: Expectations,*

Realizations and Limitations", Proc 31st Conf on Winter Simulation: Vol 2, pp 1026 – 1031, 1999.

[5] Calpin, J., et al. *"Extending the High Level Architecture Paradigm to Economic Simulation",* MITRE Report.

[6] Sullivan, D., *"Enterprise Content Management: Address Policies and Procedures Early and Often",* DM Review, September 003.

[7] Hai Jin, ChinaGrid: Making Grid Computing a Reality. ICADL 2004: 13-24

[8] Rui Liu, Yuan Chen, Keith Farkas, Dejan Milojicic and Weimin Zheng, "*Automating the Access to Monitoring Data in ChinaGrid*", Proceedings of the 13th Workshop of the HP OpenView University Association (HP-OVUA 2006) (to appear).