Principles and Experiences: Building a Hybrid Metadata Service for Service Oriented Architecture based Grid Applications

Mehmet S. Aktas, Geoffrey C. Fox, and Marlon Pierce

Abstract—**To link multiple varying Grids together, there is a need for an Information System which will act as a translator for accessing information from other Grid's Information Services. This study discusses principles and experiences in building a novel architecture of a Hybrid Service, which provides unification, federation and interoperability of different XML metadata services, and presents a performance evaluation of the prototype implementation. The results indicate that the Hybrid Service achieves information integration with negligible processing overheads while preserving persistency of information.**

Index Terms—**Information services, information integration, information federation, hybrid services, metadata services**

I. Introduction

The data requirements of e-Science applications have been increased over the years. These applications present an environment for acquiring, processing and sharing data among interested parties. In order to manage data in such data-intensive application environments, Service Oriented Architecture (SOA) principles have recently gained great importance [1]. A Service Oriented Architecture is simply a collection of services put together to achieve a common goal. These services communicate with each other for either data passing or coordinating some activity.

In order to manage information in SOA-based applications, independent projects have developed their own customized implementations of Information Service Specifications. These solutions are not interoperable with each other, target vastly different systems and address diverse sets of requirements [2]. They require greater interoperability to enable communication between different grid projects so that they can share and utilize each other's resources. Furthermore, they do not provide uniform interfaces for publishing and discovery of information and this in turn creates a limitation on the client-end, as the users have to interact with more than one services.

Mehmet S. Aktas is with the Information Technologies Institute of TUBITAK-Marmara Research Center, Gebze, Kocaeli, 41470 Turkey (corresponding author - phone: +902626772554; e-mail: mehmet.aktas@bte.mam.gov.tr).
Geoffrey C. Fox is with the Community Grids Laboratory at the Indiana University, Bloomington, IN 47403 USA (e-mail: gcf@cs.indiana.edu).
Marlon Pierce is with the Community Grids Laboratory at the Indiana University, Bloomington, IN 47403 USA (e-mail: mpierce@cs.indiana.edu).

To address these challenges, we introduce a Grid Information Service Architecture called Hybrid Service. The Hybrid Service unifies one-to-many information services and their communication protocols. It federates information coming from different information services under a unified architecture. It enables search/access/store metadata with negligible processing overheads. It enables inter-operability with wide-range of Web Service clients, as it supports widely used Web Service Specifications. It is a persistent system, as it backs-up metadata without degradation of the system performance. It is a robust system, as it enables distribution and redundancy of information.

In this paper, we discuss the Hybrid Service Architecture and present an evaluation of its prototype implementation. The organization of the rest of the paper is as follows. Section 2 reviews the background work. Section 3 presents the architectural design details and the prototype implementation of the system. Section 4 analyzes its performance evolution. Section 5 concludes the paper with a summary and discuses the future research directions.

II. Background

An effort towards interoperability in Grid Community has been promoted by the Open Grid Forum (OGF). The OGF has started a research activity called GIN (Grid Interoperation Now) [3] to manage interoperation among major grid projects such as EGEE [4] and UK National Grid Service [5]. The OGF suggests guidelines for interoperability in such a way that each grid's internal information system will enable accessing information from other information services. As the information service schema, the Open Grid Forum GIN workgroup utilizes a subset of the Grid Laboratory Uniform Environment (Glue) Schema [6], which is an effort to support interoperability between US and Europe Grid Projects. In this research, we introduce a system architecture that meets the interoperability guidelines suggested by OGF GIN work group. We integrate the Glue Schema into our design to be able to interoperate with GIN activity participating information services.

The interoperability aspect of the system requires addressing wide range of Web Service applications and providing an interoperation-bridge across the existing implementations of information services. In this research, to achieve interoperability, along with the Hybrid Service, we also provided implementations (Extended UDDI XML Metadata Service and WS-Context XML Metadata Service) for two widely used and WS-I compatible information service specifications: UDDI [7] and WS-Context [8]. The Universal Description, Discovery, and Integration (UDDI) Specification is a widely used standard that enables services advertise themselves and discover other services. The Web Services Context (WS-Context) Specification defines a simple mechanism to share and keep track of common information shared between multiple participants in Web Service interactions. The extended UDDI XML Metadata Service implements an extended version of existing out-of-box UDDI Specification. The WS-Context XML Metadata Service implements existing out-of-box Web-Service Context Specification.

Information integration is the process of unifying information residing at multiple sources and providing a unified access interface [9]. Unifying heterogeneous data sources under a single architecture has been target of many investigations [10]. Previous work on such merger between the heterogeneous information systems can be broadly categorized as global-as-view and local-as-view integration. In former category, data from several sources are transformed into a global schema and can be queried with a uniform query interface. In the latter category, queries are transformed into specialized queries over the local databases. The global schema approach captures expressiveness capabilities of customized local schemas; however, it cannot scale up to high number of data sources. The local-as-view approach requires each local-system's schema to be mapped against each other and lead to large number of mappings that need to be created and managed. To achieve high performance in data integration, there is a need for a higher-level add-on architecture that can assemble the information coming from different metadata systems and carry out queries on the heterogeneous information space. In this research, we achieve such higher-level architecture, by integrating the global-as-view approach with the heterogeneous local information services. This approach encapsulates the expressiveness power of the customized schemas that are being integrated.

Information security is a fundamental issue in Grid Information Services, as the Grid/Web Service metadata may not be open to anyone. Thus, there is a need for an information security mechanism. However, as it is not the main focus of this particular study, we leave out the investigating and leveraging of research in the information security area as future work. We concentrate on the unification, federation and interoperability aspects of the system.

III. Hybrid Service

The Hybrid Service is an add-on system that interacts with local information services and unifies them in a higher-level architecture which provides unification, federation and interoperability of Grid Information Services.

Figure 1 illustrates the Hybrid Service Architecture that assembles metadata instances coming from different local information systems. The Hybrid Service interacts with the clients, which may be supporting different communication protocols, through a uniform access interface, while it manages one-to-many local information services in the background.

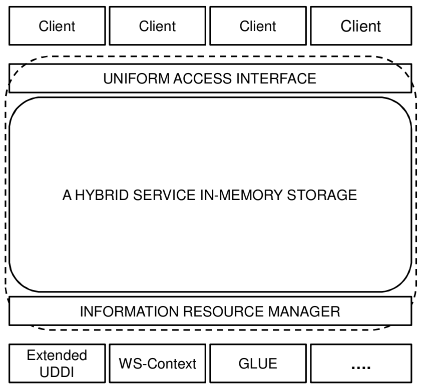

Figure 1. This figure illustrates the general view of the Hybrid Service. The dashed box indicates the Hybrid Service. The rectangle shaped boxes above the Hybrid Service indicates the Web Service clients interacting with the system. The boxes below the Hybrid Service indicate the implementations of varying Information Service Specifications.

Figure 2 illustrates a detailed view of the abstraction layers. As it can be observed in the figure, the Hybrid Service supports a Uniform Access Interface layer that consists of multiple XML APIs for varying different schemas such as Extended UDDI and WS-Context. The Uniform Access Interface layer allows different information service clients to interact with the system.

The Request-processing layer extracts incoming requests. It supports notification and access control capabilities. The notification capability enables the interested clients to be notified of the state changes in metadata. It is implemented by utilizing publish-subscribe based paradigm. The access control capability is responsible for enforcing controlled access to the Hybrid Service. The investigation and implementation of access control mechanism is left out for future study.

TupleSpaces Access API allows access to in-memory storage. (The TupleSpace paradigm [11] was first introduced by Gelernter and Carriero at Yale University as a part of Linda programming language.) In this research, we implemented a lightweight version of the JavaSpaces Specification [12] which is the implementation specification of TupleSpaces paradigm. The TupleSpaces Access API supports all query/publish operations that can take place on the Tuple Pool. The Tuple Pool implements a generalized in-memory storage mechanism based on JavaSpaces Specification. It enables mutually exclusive access and associative lookup to shared data.

The Tuple Processor layer is designed to process metadata stored in the Tuple Pool. Once the metadata instances are stored in the Tuple Pool as tuple objects, the system starts processing the tuples and provides various capabilities. The first capability is the LifeTime Management. Each metadata instance may have a lifetime defined by the user. If the metadata lifetime is exceeded, then it is evicted from the TupleSpace. The second capability is the Persistency Management. The system checks with the tuple space every so often for newly added /updated tuples and stores them into the database for persistency of information. The third capability is the Fault Tolerance Management. The system checks with the tuple space every so often for newly-added/updated tuples and replicates them in other Hybrid Service instances using the publish-subscribe (pub-sub) messaging system. This capability also provides consistency among the replicated datasets. The fourth capability is the Dynamic Caching Management. With this capability, the system keeps track of the requests coming from the pub-sub system and replicates/migrates tuples to other information services where the high demand is originated.

The Filtering layer is designed for information integration. The Hybrid Service supports a federation capability to address the problem of providing integrated access to heterogeneous metadata. To facilitate the testing of this capability, a Unified Schema is introduced by integrating different information service schemas. If the metadata is an instance of the Unified Schema, such metadata needs to be mapped into the appropriate local information service back-end. To achieve this, the Hybrid Service utilizes the Filtering layer. This layer does filtering based on the user-defined mapping rules to provide transformations between the Unified Schema instances and local schema instances. If the metadata is an instance of a local schema, then the system does not apply any filtering, and backs-up this metadata to the corresponding local information service back-end.

The Information Resource Manager layer is responsible for managing low-level information service implementations. It provides decoupling between the Hybrid Service and sub-systems. The Information Resource Manager handles with the management of local information service implementations. It provides decoupling between the Hybrid Service and sub-systems. With the implementation of Information Resource Manager, we have provided a uniform, single interface to sub-information systems.

The Pub-Sub Network layer is responsible for communication between Hybrid Service instances. On receiving the requests from the Tuple Processor, the Pub-Sub Manager publishes the request to the corresponding topics. The Pub-Sub Manager may also receive key-based access/storage requests from the pub-sub network. In this case, these requests will be carried out on the Tuple Pool by utilizing TupleSpace Access API. The Pub-Sub Manager utilizes publisher and subscriber sub-components in order to provide communication among the instances of the Hybrid Services.

The execution logic to process a key-based retrieval request happen as follows. On receiving the client request, the request processor extracts the incoming request. The request processor processes the incoming request by checking it with the specification-mapping metadata (SpecMetadata) files. For each supported schema, there is a user-defined SpecMetadata file, which defines all necessary information for the functions that can be executed on the instances of the schema under consideration. Based on this information, the request processor extracts the inquiry/publish request from the incoming message and executes these requests on the Tuple Pool.

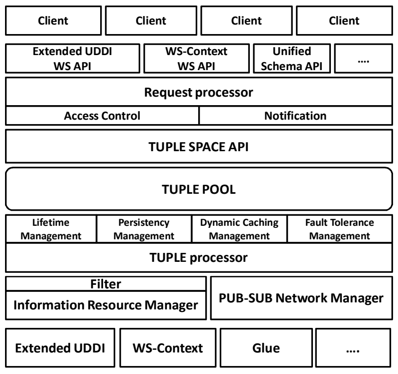

Figure 2 This figure illustrates the abstraction layers of the Hybrid Service from top-to-bottom.

To execute a key-based retrieval request the following strategy is applied. The system keeps all locally available metadata keys in a table in the memory. On receipt of a request, the system first checks if the metadata is available in the memory by checking with the metadata-key table. If the requested metadata is not available in the local system, the request is forwarded to the Pub-Sub Manager layer to probe other Hybrid Services for the requested metadata. If the metadata is in the in-memory storage, then the request processor utilizes the Tuple Space Access API and executes the query in the Tuple Pool.

Once the request is extracted and processed, the system presents access control and notification capabilities. With these capabilities, the system ensures controlled access and informs interested parties of the state changes happening in the metadata. This way the requested entities can keep track of information regarding a particular metadata instance.

If the request is to be handled in the memory, the Tuple Space Access API is used to enable the access to the in-memory storage. This API allows us to perform operations on the Tuple Pool. Once the metadata instances are stored in the

Tuple Pool as tuple objects, the tuple processor layer is being used to check with the Tuple Pool every so often for newly-added / updated tuples.

If the metadata is to be stored to the information service backend (for persistency of information), the Information Resource Management layer is used to provide connection with the back-end resource. If the metadata is to be replicated/stored into other Hybrid Service instances, the Pub-Sub Management Layer is used for managing interactions with the rest of the Hybrid Service network.

IV. EVALUATION

We investigated the performance and scalability aspects of the Hybrid Service. This investigation was conducted for Unified Schema XML API standard operations. In this evaluation, the following research questions are addressed:

- What is the performance of the Hybrid Service for the Unified Schema XML API standard operations?
- How do Unified Schema XML API functions compare against other supported Schema XML APIs such as WS-Context XML API?
- What is the scalability of Hybrid Service prototype for Unified Schema XML API standard operations?

The investigations are conducted using eight nodes of a Linux cluster located at the Community Grids Laboratory of Indiana University. The configuration of the cluster nodes and the software environment for the experiments is listed in Table 1. In the experiments, the performance is evaluated with respect to response time at client applications. The response time is the average time from the point a client sends off a query until the point the client receives a complete response. We conducted two experiments to understand the behavior of the system: performance and scalability.

Program Testing Setup	
Processor	*Intel® Xeon™ CPU (2.40GHz)*
RAM	*2GB total*
Network Bandwidth	*100 Ambits/sec. (among the cluster nodes)*
OS	*GNU/Linux (kernel release 2.4.22)*
Compiler	*Java 2 Standard Edition v.1.5 with maximum heap size of 1024 MB using the –Xmx1024m option*
Servlet container	*Tomcat Apache Server v.5.5.8 with max. multiple thread number of 1000*
Web Service container	*Apache Axis v.2.0*
Database	*MYSQL with v.4.1*
Timing function	*Java 2 with v.1.5 – timing function "nanoTime()"*

Table 1 Program testing environment configuration

The performance experiment is conducted to understand the baseline performance of the Hybrid Service. This evaluation investigates the performance of system for standard Unified Schema operations and compares it against the performance of WS-Context Schema standard operations when there is no additional traffic. To do this following testing cases are completed: a single client sends publish requests to an echo service which receives a message and then sends it back to the client with no processing applied; a single client sends publish requests to a Hybrid Service which grants the request with memory access; a single client sends publish requests to a Hybrid Service which grants the request with database access. In the experiment, both the Hybrid Service and testing client application were located in two different servers located in the Linux cluster. This experiment was repeated five times and we record the average response time. Recall that the Hybrid Service backs-up information every so often for persistency reasons. During the investigation, we observed that if the backup frequency is every 10 seconds or lower, average execution time for publish operation stabilized to ~7.5 milliseconds. Therefore, we chose the value for backup frequency as every 10 sec in the experiments.

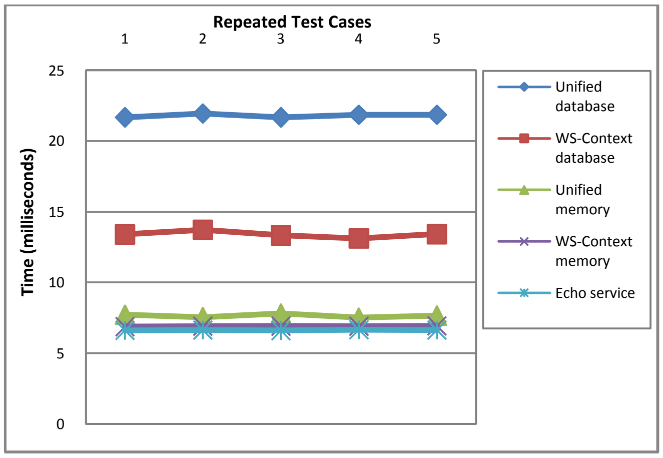

Figure 3 Round Trip Time Chart for Metadata Inquiry Requests

Analyzing the results depicted in Figure 3, we observe that the Hybrid Service achieves noticeable performance improvements in metadata management for standard operations by simply employing an in-memory storage mechanism, while preserving a certain persistency level. We also observe that the Unified Schema operations require more time (compared to WS-Context Schema operations) for database accesses, since the system requires additional time to perform transformation between the Unified Schema and WS-Context Schema instances.

In the scalability experiment, we investigated how well the Hybrid Service - Unified Schema XML API performs for increasing message rates. To answer this question, we ramped-up the work load (number of messages sent per second) until the system performance degrades.

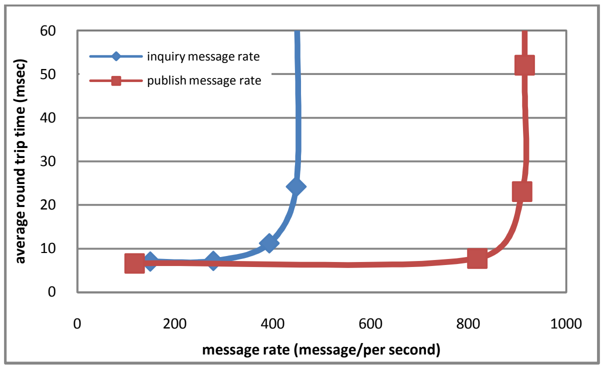

Figure 4 Round Trip Time Chart for Metadata Inquiry Requests

The results are depicted in Figure 4. Analyzing the results, we conclude that Hybrid Service Unified Schema XML API standard operations performed well under increasing message rates. For inquiry request messages, we observe a threshold value after which the system performance starts decreasing due to high message rate. This threshold is mainly due to the limitations of Web Service container, as we observe the similar threshold when we test the system with an echo service that returns the input parameter passed to it with no message processing is applied. For publish request messages, we observe another threshold value where the system performance starts dropping down. The reason for this is the following. As the publish message-rate is increased, the number of updated/newly written metadata in the Tuple Pool is also increased. In turn, the action that writes and transforms the larger number of updates into the default local information service back-end affects the system performance and causes higher fluctuations in the response times for increasing number of simultaneous publish requests.

V. Conclusion and Future Work

The performance evaluation and related tests have served to prove several of the basic concepts of Hybrid Metadata Service architecture while revealing bottlenecks and areas of needed performance improvement. The evaluation study pointed out that the Hybrid Service achieves interoperability amongst different information systems with negligible processing overheads. It also pointed out that the Hybrid Service scales to high number of message rates while supporting integration and preserving persistency of information.

We intend to further improve this service by investigating an information security mechanism for the distributed information service network.

Acknowledgment

The Advanced Information Systems Technology Program of NASA's Earth-Sun System Technology Office supported this research.

References

[1] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. "Web Service Architecture." WC3 Working Group Note, 11 February 2004. Available from http://www.w3c.org/TR/ws-arch

[2] Zanikolas, S., Sakellariou, R., A Taxonomy of Grid Monitoring Systems. . Future Generation Computer Systems, 21(1), 2005: p. pp. 163--188

[3] OGF Grid Interoperation Now Community Group (GIN - CG) - Web site is available at https://forge.gridforum.org/-projects/gin

[4] The Enabling Grids for E-science (EGEE) project - Web site is available at http://www.eu-egee.org/ Access date: October, 2007

[5] The National Grid Service (NGS) - Web site available is at http://www.grid-support.ac.uk/, Access date: October, 2007

[6] GLUE Schema Collaboration. The GLUE Schema homepage. http://infnforge.cnaf.infn.it/glueinfomodel/

[7] Bellwood, T., Clement, L., and von Riegen, C., UDDI Version 3.0.1: UDDI Spec Technical Committee Specification http://uddi.org/pubs/uddi-v3.0.1-20031014.htm. 2003.

[8] Bunting, B., Chapman, M., Hurley, O., Little M,, Mischinkinky, J., Newcomer, E., Webber, J., and Swenson, K. , Web Services Context (WS-Context) ver 1.0 http://www.arjuna.com/library/specs/ws_caf_1-0/WS-CTX.pdf. 2003.

[9] Lenzerini, M., Data Integration: A Theoretical Perspective, in PODS: 243-246. 2002

[10] Ziegler, P., Dittrich, K., Three Decades of Data Integration - All Problems Solved?, in WCC: 3-12. 2004

[11] Carriero, N., Gelernter, D., Linda in Context. Commun. ACM, 32(4): 444-458, 1989.

[12] Sun_Microsystems, JavaSpaces Specification Revision 1.0, 1999 available at http://www.sun.com/jini/specs/js.ps.