Information Federation in Grids

Mehmet S. Aktas[#+*1], *Geoffrey Fox*[#+2], *Marlon Pierce*[#3]

[#]*Computer Science Department, Indiana University*
Bloomington, IN, USA

[+]*Community Grids Laboratory, Indiana University*
Bloomington, IN, USA

[*]*Informatics Technologies Institute*
Marmara Research Center, Gebze, TR

[1]mehmet.aktas@bte.mam.gov.tr
[2]gcf@indiana.edu
[3]mpierce@indiana.edu

Abstract— **Independent Grid projects have developed their own solutions to Information Services. These solutions are not interoperable with each other, target vastly different systems and address diverse sets of requirements. To address these challenges, we designed a novel architecture for a Grid Information Service that provides unification, federation and interoperability of major grid information services. The proposed approach forms an add-on information system that interacts with the local information services and assembles their metadata instances. We introduce a prototype implementation and present its evaluation. The results indicate that the proposed approach achieves unification and federation of custom implementations of grid information services with negligible processing overheads.**

Index Terms— ***Grid Information Services, Information Federation, Metadata Services, Hybrid 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. Service Oriented Architecture (SOA) principles have recently gained great importance [1] to manage data in such data-intensive application environments.

Independent projects have developed their own customized implementations of information service specifications to manage information in SOA-based applications. 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. In turn, this creates a limitation on the client-end, as the users have to interact with more than one service.

Zhuge identifies the two mainstream research in the next-generation of Web in [3]: a) Web's challenges in supporting intelligent services, b) computational science problems to utilize the idle resources of large numbers of distributed computers. To overcome the two aforementioned mainstream research themes, much work is needed to improve the existing Web including the Web Services, Semantic Web and Web Intelligence [4].

This research addresses the challenges encountered in Web Services area and proposes a solution to a fundamental problem, which is to locate services of interest in different SOA-based, independent and distributed Grid application domains. It introduces novel ideas on how to integrate information coming from different Grids at application-level. It describes a system, which is highly applicable in semantic computing, knowledge networking, and grid computing application use domains.

To present the applicability of this investigation, we identify metadata management requirements of two application use domains: Global Multimedia Collaboration System (GlobalMMCS [5]) and Pattern Informatics Geographical Information System Grid (PI GIS-Grid [6]). GlobalMMCS is a peer to peer collaboration environment, where videoconferencing sessions, with any number of widely distributed services, can take place. GlobalMMCS requires persistent archival of session metadata to provide replay/playback and session failure recovery capabilities. The PI GIS-Grid is a workflow-style Grid application, which requires storage of transitory metadata needed to correlate activities of participant entities. Both application domains require a unified information service, which can support large-scale, read-mostly, quasi-static and small-scale, highly-updated, dynamic information. Although much work has been done on information management in Grid Information Services, to our best knowledge, none of the previous work addresses a hybrid approach, which integrates different information services.

We propose a Grid Information Service Architecture called Hybrid Grid Information Service (Hybrid Service) to cope with these challenges. 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.

Section 2 reviews the literature. Section 3 presents the architectural design. Section 4 explains how to use the proposed system. Section 5 discusses the prototype implementation. Section 6 analyses the prototype evaluation. Section 7 concludes the paper with future research directions.

II. Literature review

Information integration is the process of unifying information residing at multiple sources with a unified access interface [7]. Unifying heterogeneous data sources under a single architecture has been target of many investigations [8]. 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. Although the global schema approach captures expressiveness capabilities of customized local schemas, it does not scale up to high number of data sources. In the local-as-view approach, each local-system's schema needs to be mapped against each other to transform the queries. In turn, this leads to large number of mappings that need to be created and managed.

The proposed system differs from local-as-view approaches, as its query transformation happens between a unified schema and local schemas. It utilizes and leverages previous work on global-as-view approach for integrating heterogeneous local data services. The previous work mainly focuses on solutions that automate the information federation process at semantics level. Different from the previous work, the proposed approach presents a system architecture that enables information integration at application level. To our best knowledge, the proposed system is a pioneer work in Grids, as it describes a Grid Information Service architecture that enables unification and federation of information coming from different metadata systems.

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) to manage interoperation among major grid projects such as EGEE [9] and UK National Grid Service [10]. 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. The GIN working group utilizes Grid Laboratory Uniform Environment (Glue) Information Service Schema, which is an effort to support interoperability between US and Europe Grid systems. In this research, we introduce a system architecture that meets the interoperability guidelines suggested by OGF GIN work group. To facilitate testing of the proposed system, we experimented the integration of the Glue Schema with the Hybrid Services and showed that it worked in [27].

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: Universal Description, Discovery, and Integration (UDDI) [11] and Web Services Context (WS-Context) [12]. The UDDI Specification is a widely used standard that enables services advertise themselves and discover other services. The 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 [13], [25] implements an extended version of existing out-of-box UDDI Specification. The WS-Context XML Metadata Service [14], [26] implements existing out-of-box Web-Service Context Specification.

Information security is a fundamental issue in Grid Information Services, as the Grid/Web Service metadata may not be open to anyone. Although, we are aware of its necessity, 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.

Managing interactions of different processes through an associated memory is a well studied area in distributed systems [15]. A pioneer work in this research field, TupleSpaces, was first introduced by Gelernter and Carriero at Yale University [16] as a part of Linda programming language. It forms an associated shared memory through which two or more processes can exchange/share data. It provides mutual exclusive access, associative lookup and persistence for a repository of tuples that can be accessed concurrently. Sun Microsystems introduced a java based specification (the JavaSpaces Specification [17]) for Linda and provided its implementation. MicroSpaces [18], an open-source java implementation of the JavaSpaces Specificion, introduces an alternative collection of java libraries and provide API semantics identical with Sun's implementation. IBM has a tuplespaces implementation called TSpaces [19]. Linda has been extended to support different types of communication and coordination between systems and has increased some interest in diverse communities such as the ubiquitous computing (sTuples [20]) and Semantic Web (Triple Spaces [21], Semantic Web Spaces [22]). Despite of their important features, previous java implementations of TupleSpaces, such as Sun's implementations, have high complexity and require a number of daemon services to run including naming services and restart services. MicroSpaces implements a multi-threaded application, free of daemon services, however it is still dependent on RMI to regulate interactions of entities.

The proposed system employs an associated memory paradigm, TupleSpaces, to support mutually exclusive access, which in turn enables data sharing between processes. It presents similarities and differences with previous java implementations of TupleSpaces paradigm. Similar to Java-based implementations, it implements the JavaSpaces Specification that was also introduced by Sun Microsystems. Different from Sun's implementation, it introduces a light-weight version of JavaSpaces that does not rely on existence of daemon services such as naming services. Different from MicroSpaces, it does not depend on RMI technology to regulate interactions of communicating parties.

III. HYBRID GRID INFORMATION SERVICE

The proposed information service is an add-on system that interacts with local information services and unifies them in a higher-level architecture. It is named as Hybrid Grid Information Service (Hybrid Service). It provides unification, federation and interoperability of Grid Information Services. It assembles metadata instances coming from different local information systems. It is able to interact with clients, which may be supporting different communication protocols. It also provides a uniform access interface and manages one-to-many local information services.

Figure 1 illustrates a detailed view of the abstraction layers. (1) *The Uniform Access Interface layer* provides uniform access interface. It consists of multiple XML APIs for varying schemas such as Extended UDDI and WS-Context. It allows clients with different communication protocols interact with the system. (2) *The Request-processor layer* extracts incoming requests. It supports access control and notification capabilities. The access control capability is responsible for enforcing controlled access to the Hybrid Service. The notification capability enables the interested clients to be notified of the state changes in metadata. (3) *TupleSpaces Access API and Tuple Pool layers* are for in-memory storage. The TupleSpaces Access API supports all query/publish operations that can take place in memory. The Tuple Pool is a generalized in-memory storage mechanism, which provides mutually exclusive access and associative lookup capabilities.

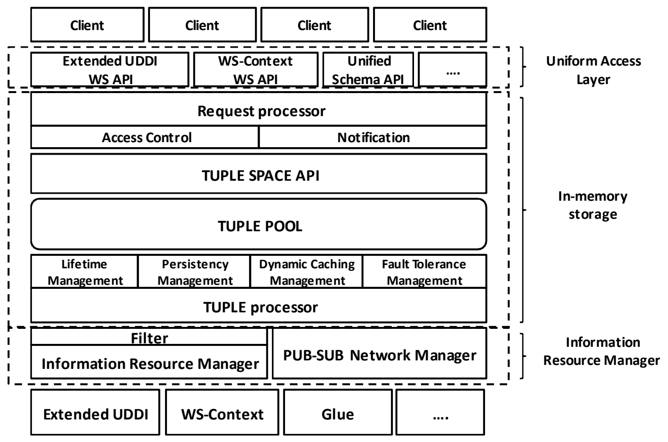

Figure 1 The abstraction layers of the Hybrid Service from top-to-bottom.

(4) *The Tuple-processor layer* processes metadata stored in the Tuple Pool and provides following capabilities: LifeTime Management, Persistency Management, Dynamic Caching Management and Fault Tolerance Management. The LifeTime Management is about evicting tuples with expired leases. The Persistency Management deals with backing-up newly-stored / updated metadata into the information service back-ends with pre-defined backup-interval times. The Fault Tolerance Management has to do with creating replicas of the newly added metadata. The Dynamic Caching Management is about replicating/migrating metadata, under high demand, onto replica servers, where the demand originated. (5) *The Filter layer* is for information integration. It provides decoupling between the Hybrid Service and sub-systems. It does filtering and transformations based on the user-defined rules and provides mapping between a unified schema and local schemas. (6) *The Information Resource Manager layer* manages low-level information service implementations. It handles the management of local information service implementations and provides a uniform, single interface to sub-information systems. (7) *The Pub-Sub Network Manager layer* manages communication between Hybrid Service instances. It handles communication between the distributed Hybrid Service nodes to support access-request distribution, replica-content placement and consistency enforcement.

IV. USING THE HYBRID GRID INFORMATION SERVICE

On receiving the client request, the Request-processor extracts the incoming request. The Request-processor processes the incoming request with an appropriate XML API. It knows which XML API to use, simply by checking with the specification-mapping metadata file. (Note that, for each supported schema, there is a user-provided specification metadata file, which defines all necessary information for the functions that can be executed on the instances of the schema under consideration.) Then, the Request-processor extracts the inquiry/publish request from the incoming message and executes these requests on the Tuple Pool.

Once the request is extracted and processed, the system executes its 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 clients can keep track of information regarding a particular metadata instance.

On receipt of a request, the system first checks, if the metadata is available in the memory by checking with the metadata-key table. (Note that, the system keeps all locally available metadata keys in a table in the memory.) If the requested metadata is not available in the local system, then the request is forwarded to the Pub-Sub Network Manager 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 metadata instances are stored in the Tuple Pool as tuple objects, the Tuple-processor 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 reasons), the Information Resource Manager is used to provide connection with the back-end resource. If the metadata is to be replicated/stored into other Hybrid Service instances (for fault-tolerance reasons), the Pub-Sub Network Manager is used for managing interactions with the rest of the Hybrid Services.

V. Prototype Implementation

The Hybrid Service prototype implementation [23] consists of various modules such as Query-Publish, Expeditor, Filter, Resource Manager, Sequencer, Access-Storage.

The Query-Publish module implements the Uniform Access Interface and the Request-processor abstraction layers. It is responsible for processing the incoming requests issued by end-users. It provides a uniform access interface with two capabilities: Access control and Notification. *Access control:* We implemented a simple access control mechanism, which requires an authentication token to restrict who can perform inquiry/publish operation. The authorization token is obtained from the system at the beginning of client-server interaction and enables authorized access to the system. *Notification:* We also implemented a notification scheme, which is achieved by utilizing publish-subscribe based messaging scheme. This enables users of Hybrid Service to utilize a push-based information retrieval capability, where the interested parties are notified of the state changes. We use NaradaBrokering software [24] as the messaging infrastructure and its libraries to implement subscriber and publisher components.

The Expeditor module implements the TupleSpaces Access API, Tuple Pool and Tuple-processor abstraction layers. The Tuple Spaces Access API provides an access interface on the Tuple Pool, which is a generalized in-memory storage mechanism. We implemented a lightweight version of the JavaSpaces Specification [17], an implementation specification of TupleSpaces paradigm [16], for in-memory storage. The Tuple-processor handles with metadata stored in Tuple Pool 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 Filter module implements the Filter abstraction layer and provides decoupling between the Hybrid Information Service and the sub-systems. 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 at semantic-level. 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 Filter module provides a mapping capability based on the user-defined mapping rules. The Filter module obtains the mapping rule information from the user-provided mapping rule files. As the mapping rule file, we use the XSL (style-sheet language for XML) Transformation (XSLT) file. The XSLT provides a general purpose XML transformation. If the metadata is an instance of the unified schema, then the system maps this metadata to the default local information service backend. Here, the mapping happens between the Unified Schema and the local information service schemas (such as WS-Context or extended UDDI schemas). 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 module, illustrated in Figure 2, implements the Information Resource Manager abstraction-layer and manages local information services. The Resource Manager module separates the Hybrid System from its sub-systems. It knows which sub-system classes are responsible for a request and what method needs to be executed by processing the specification-mapping metadata file that belong the local information service under consideration. On receipt of a request, the Information Resource Manager checks with the corresponding mapping file and obtain information about the specification-implementation. Such information could be about a class (which needs to be executed), it's function (which needs to be invoked), and function's input and output types, so that the Information Resource Manager can delegate the handling of incoming request to appropriate sub-system. By using this approach, the Hybrid Service can support one-to-many information services as long as the sub-system implementation classes and the specification-mapping metadata (SpecMetadata) files are provided. The Resource Handler is an external component to the Hybrid Service. It is used to interact with sub-information systems. Each specification has a Resource Handler, which allows interaction with the database. The Hybrid System classes communicate with the sub-information systems by sending requests to the Information Resource Manager, which forwards the requests to the appropriate sub-system implementation. Although the sub-system object (from the corresponding Resource Handler) performs the actual work, the Information Resource Manager seems as if it is doing the work from the perspective of the Hybrid Service inner-classes. This approach separates the Hybrid Service implementation from the local schema-specific implementations. The Resource Manager module is also used for recovery purposes. We have provided a recovery process to support persistent in-memory storage capability. This type of failure may occur if the physical memory is wiped out, when power fails or machine crashes. This recovery process converts the database data to in-memory storage data (from the last backup). It runs at the bootstrap of the Hybrid Service. This process utilizes user-provided "find_schemaEntity" XML documents to retrieve instances of schema entities from the information service backend. Each "find_schemaEntity" XML document is a wrapper for schema specific "find" operations. At the bootstrap of the system, firstly, the recovery process applies the schema-specific find functions on the information service backend and retrieves metadata instances of schema entities. Then, the recovery process stores these metadata instances into the in-memory storage to achieve persistency.

The Sequencer module implements the synchronization to impose an order on the actions that take place in the system. The responsibility of the Sequencer module is to assign a timestamp to each metadata, which will be stored into the Hybrid Service. To do this, the Sequencer module interacts with Network Time Protocol (NTP)-based time service implemented by NaradaBrokering software. This service achieves synchronized timestamps by synchronizing the machine clocks with atomic timeservers available across the globe.

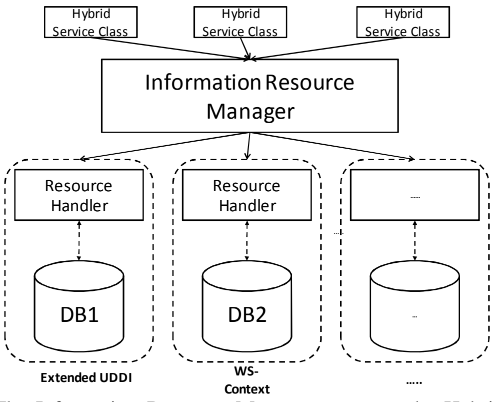

Figure 2 The Information Resource Manager separates the Hybrid Service implementation from the implementations of local metadata-systems.

The Access-Storage module implements the Pub-Sub Network Manager abstraction layer. It is responsible for communication between the distributed Hybrid Service nodes to support the functionalities of a replica hosting system. These functionalities include access-request distribution, replica-content placement and consistency enforcement. It utilizes publisher and subscriber sub-components to provide communication among the instances of Hybrid Services. On receiving the access/storage/update requests from the Expeditor module, it publishes the request to the corresponding topics. On receiving the access/storage/update requests from the Network, it carries out the operation on the Tuple Pool by utilizing TupleSpace Access API.

VI. PROTOTYPE EVALUATION

A prototype evaluation is conducted to explore performance and scalability research issues for Unified Schema XML API standard operations. In this investigation, the following research questions are addressed:

a) What is the performance of the Hybrid Service for the Unified Schema XML API standard operations?

b) How do Unified Schema XML API functions compare against other supported Schema XML APIs such as WS-Context XML API?

c) What is the scalability of Hybrid Service prototype for Unified Schema XML API standard operations?

For this evaluation, we used 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. The performance of the system is tested with a multithreaded client program that takes the following arguments: a) the number of threads (N) and b) number of messages (T) to be fired by each thread. We illustrate our timing methodology in the pseudo code below.

```
SET the number of threads to N
SET the number of transaction to be executed to T
CREATE N number of threats
STOP the threads until N threads are created and ready
ThreadSleep(random(1000))
FOR X = 1 to T
        SET start to 0, stop to 0
        START time
        Hybrid_Service_API(..)
        STOP time
        PRINT  (elapse time)
END FOR
```

We conducted two experiments to understand the behavior of the system with respect to information federation. These are performance and scalability experiments.

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: (1) 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; (2) 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. The simulation parameters are given in Table 2. Note 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.

Simulation parameters	
Metadata size	*1.7 KB*
Registry size	*5000 metadata*
Backup interval time	*Every 10 milliseconds*
Observation number	*200*

Table 2 Simulation parameters. Metadata examples for the experiments are given in the Appendices A and B.

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.

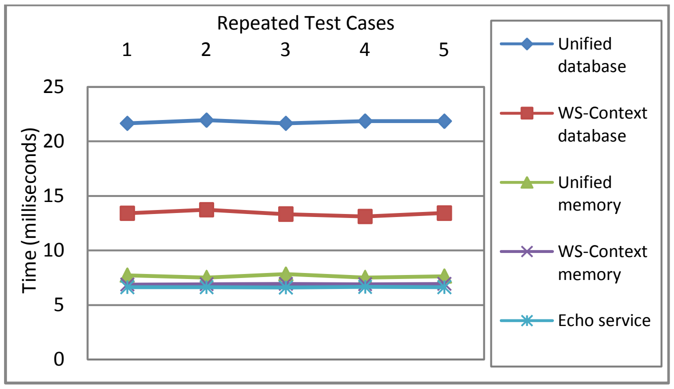

Figure 3 Average round-trip time chart for metadata inquiry requests for five-time repeated test cases.

The scalability experiment is conducted to understand how well the Hybrid Service - Unified Schema XML API performs under increasing message rates. To answer this question, we ramped-up the work load (number of messages sent per second) until the system performance degrades. The results are depicted in Figure 4. The simulation parameters are the same as in Table 2.

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.

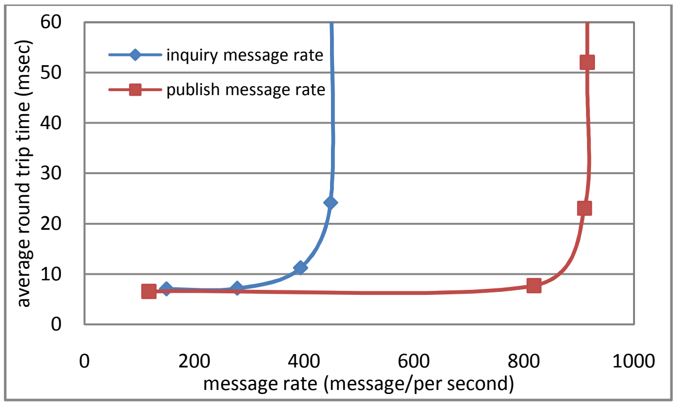

Figure 4 Average round trip time chart for metadata inquiry requests under increasing message rates.

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 (particularly for increasing number of simultaneous publish requests). We use water-marking technique to add stability to the system and overcome this problem. The high-mark reflects the threshold of a single host. Exceeding the high water-mark is not desirable, as it degrades the system performance. Thus, to provide stability, if the workload becomes above the high-watermark, incoming messages are forwarded to other nodes that can handle the requests. The discussion on distribution and load balancing is left out, since it is out of scope. This paper mainly focuses on application-level information integration in Grid Information Services.

VII. Conclusion and Future Research Directions

We introduced a novel architecture for a Hybrid Grid Information Service supporting handling and discovery of not only quasi-static, stateless metadata, but also session related metadata. The Hybrid Service is an add-on architecture that runs one layer above existing information service implementations. It provides unification, federation and interoperability of Grid Information Services. To achieve unification, the Hybrid Service is designed as a generic system with front and back-end abstraction layers supporting one-to-many local information systems and their communication protocols. To achieve federation, the Hybrid Service is designed to support information integration technique in which metadata from several heterogeneous sources are transferred into a global schema and queried with a uniform query interface. To manage both quasi-static and dynamic metadata and provide interoperability with wide-range of Web Service applications, the Hybrid Service is integrated with two widely-used, WS-I compatible Web Service Specifications: UDDI and WS-Context and their implementations: Extended UDDI XML Metadata Service [25] and WS-Context XML Metadata Service [26].

We performed a set of experiments to evaluate the performance and scalability of the Hybrid Service to understand whether it can achieve information federation in Grids with acceptable costs. The evaluation study pointed out that the Hybrid Service achieves information federation amongst different Grid 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 the proposed work by investigating an information security mechanism for the distributed Hybrid Service network.

Acknowledgment

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

Appendix: XML Metadata Examples

A. Context XML Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<wscontext:context
        xmlns:wscontext="http://datatype.fthpis.cgl/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <contextKey>ABCCE800-AB35-11DA-A4FC-
C80C5880CB18</contextKey>
        <serviceKey>ABCCE800-AB35-11DA-A4FC-
C80C5880CB19</serviceKey>
        <sessionKey>ABCCE800-AB35-11DA-A4FC-
C80C5880CB20</sessionKey>
        <name>context://GIS/PI/ABCCE544-CX35-11EA-BVFC-
C34C7789CB33</name>
        <value>context:///GIS/VC/3ea29661-2d5e-11db-8c56-
cf37cd202027/3ebd7162-2d5e-11db-8c56-cf37cd202027/cost</value>
        <valueType>String</valueType>
        <lease>
                <timeout>1000</timeout>
                <isInfinite>false</isInfinite>
        </lease>
        <version>1</version>
</wscontext:context>
```

B. Unified Schema XML Metadata

```
<?xml version="1.0" encoding="UTF-8"?>
<unified_schema:service
        xmlns:hybrid_schema="http://datatype.generic.fthpis.cgl/"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <serviceKey>856679F0-B4B6-11DA-A1DD-
E719F6E12358</serviceKey>
```

```
        <serviceType>Web Feature Service</serviceType>
        <name>Service Name</name>
        <description>
                <value>Service Description</value>
        </description>
        <serviceEndpointAddress>http://gf7.ucs.indiana.edu:8092/wfs-
streaming-service/services/wfs</serviceEndpointAddress>
        <metadata>
                <metadataKey>7115B940-A95E-11DA-B940-
CB4E3E38D98F</metadataKey>
                <serviceKey>856679F0-B4B6-11DA-A1DD-
E719F6E12358</serviceKey>
                <name>session-id</name>
                <value>0001</value>
                <lease><isInfinite>true</isInfinite></lease>
                <version>1</version>
        </metadata>
        <lease><isInfinite>true</isInfinite></lease>
</unified_schema:service>
```

REFERENCES

[1] Booth, D., Haas, H., McCabe, F., Newcomer, E., Champion, M., Ferris, C., and Orchard, D. *W3C Web Service Architecture Document, available at http://www.w3.org/TR/ws-arch/, 2004.*

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

[3] Zhuge, H., *The Knowledge Grid, World Scientific Publishing Co.* 2004.

[4] Zhuge, H., *China's E-Science Knowledge Grid Environment.* IEEE Intelligent Systems, 19 (1) (2004) 13-17, 2004.

[5] Wu, W., et al., *Grid Service Architecture for Videoconferencing*, in *"Grid Computational Methods" Edited by M.P. Bekakos, G.A. Gravvanis and H.R. Arabnia.*

[6] Aydin, G., Aktas, M. S., et al., *SERVOGrid Complexity Computational Environments (CCE) Integrated Performance Analysis.* GRID' 05: Proceedings of the 6th IEEE/ACM International Workshop on Grid Computing, November 2005.

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

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

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

[10] NGS, The National Grid Service (NGS) - Web site available is at http://www.grid-support.ac.uk/, Access date: 9/08.

[11] Bellwood, T., Clement, L., and von Riegen, C., *UDDI Version 3.0.1: UDDI Spec Technical Committee Specification, available at http://uddi.org/pubs/uddi-v3.0.1-20031014.htm*. Access date: 9/08.

[12] 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, Access Date: Sept 7, 08, available at http://www.arjuna.com/-library/specs/ws_caf_1-0/WS-CTX.pdf.*

[13] Aktas, M. S., *Extended UDDI XML Metadata Service web site, available at http://www.opengrids.org/extendeduddi*, Access Date: Sept 7, 2008

[14] Aktas, M. S., *Fault Tolerant High Performance Information Service - FTHPIS - Hybrid WS-Context Service web site, available at http://www.opengrids.org/wscontext*, Access date: 9/2008.

[15] Tanenbaum, A., Van Steen, M., *Distributed Systems Principles and Paradigms*. 2002. Cited in page 326.

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

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

[18] Coleman, r., Bhardwaj, A., Dellucca, A., Finke, G., Sofia, A., Jutt, M., Batra, S., *MicroSpaces software with version 1.5.2 available at http://microspaces.sourceforge.net/*. 2004.

[19] Wyckoff, P., Lehman, T. J., McLaughry, S., *T Spaces.* IBM Systems Journal, 1998. **37(3)**: p. 454-474.

[20] Khushraj, D., Lassila, O., Finin, T. *sTuples:Semantic Tuple Spaces*. in *IEEE Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems:Networking and Services (MobiQuitous'04)*. 2004.

[21] Krummenacher, R., Strang, T., Fensel, D. *Triple Spaces for and Ubiquitous Web of Services*. in *W3C Workshop on the Ubiquitous Web*. March 2005. Tokyo, Japan.

[22] Tolksdorf, R., Nixon, L., Liebsch, F., Nguyen, M.D., Bontas, P.E., *Semantic Web Spaces*, in *Technical Report B-04-11*. July 2004, Freie Univesitat Berlin, Institut fur Informatik: Berlin, Germany.

[23] Aktas, M. S., *Hybrid Grid Informatin Service web site, available at http://www.opengrids.org/hybrid*. Access date: 9/2008.

[24] Pallickara, S. and G. Fox. *NaradaBrokering: A Middleware Framework and Architecture for Enabling Durable Peer-to-Peer Grids*. in *Lecture Notes in Computer Science*. 2003: Springer-Verlag.

[25] Aktas, M. S., *XML Metadata Services*, Concurrency and Computation: Practice & Experience, Volume 20, Issue 7, May 2008.

[26] Aktas, M. S., *Information Services for Dynamically Assembled Semantic Grids*, SKG' 05: Proceedings of the First International Conference on Semantics, Knowledge and Grid, November 2005.

[27] Aktas, M. S., *Information Federation in Grid Information Services*, Doctoral Thesis, Indiana University, 2007.