# Semantic Enabled Metadata Management in PetaShare

**Xinqi Wang*, Dayong Huang, Ismail Akturk, Mehmet Balman, Gabrielle Allen and Tevfik Kosar**
Center for Computation & Technology, and Department of Computer Science
Louisiana State University, Baton Rouge, LA 70803, USA
E-mail: {xinqiwang, dayong, akturk, balman, gallen, kosar}@cct.lsu.edu
*Corresponding author

**Abstract:** We designed a semantic enabled metadata framework using ontology for multi-disciplinary and multi-institutional large scale scientific data sets in a Data Grid setting. Two main issues are addressed: data integration for semantically and physically heterogeneous distributed knowledge stores, and semantic reasoning for data verification and inference in such a setting. This framework enables data interoperability between otherwise semantically incompatible data sources, cross-domain query capabilities and multi-source knowledge extraction. In this paper, we present the basic system architecture for this framework, as well as an initial implementation. We also analyze a real-life scenario and show integration of our framework into the PetaShare Data Grid where multi-disciplinary data archives are geographically distributed across six research institutions in Louisiana.

**Keywords:** Ontology, metadata management, Data Grid, cross-domain query

**Biographical notes:** Xinqi Wang is currently a PhD student at Center for Computation and Technology and Department of Computer Science at Louisiana State University, Baton Rouge, Louisiana, United States. Since January, 2007, he has been involved in Petashare project and his research mainly focuses on ontology metadata in grid and distributed computing environment.

## 1 INTRODUCTION

One of the key problems in Data Grids is interoperability between different data sources and management of metadata for cross-domain projects. Metadata enables physical data to be effectively discovered, interpreted, evaluated, and processed. Today, the scientific research community faces new challenges in metadata management as computing environments become increasingly large and complex. For example, in the Atlas (2008) and CMS (2008) projects alone, more than 200 institutions from 50 countries use a data collection which increases by around 5 petabytes annually. These large collaborations involve not only domain scientists, but also computer scientists, engineers, and visualization experts who need to access the data to advance research in their own fields. Traditional catalogue based metadata services have limitations in such application scenarios. It is difficult to handle data integration across different domains; management of domain schema evolution often leads to confusion; and data quality degrades since the data verification/quality control cannot be easily built in such a distributed environment from a catalogue metadata service.

Ontology is a fairly new data model tool based on formal logic. It has a rich expressiveness while maintaining decidability. Ontologies enable knowledge engineers to build logic constraints to enable data quality checking, facilitate knowledge collaboration in highly distributed environments, and provide cross-domain data integration and interoperability. Efforts have been made to use ontologies to address metadata management challenges in Data Grids. However, such attempts are still in their early stages.

In this paper we present an ontology enabled metadata management service capable of handling semantic rich queries and tailored for the needs of scientific data collections in Data Grids. This system takes advantage of existing research efforts in ontology research and Data Grid research. We apply this framework to PetaShare, an NSF funded Data Grid project. PetaShare provides an excellent testbed for such a framework since it spans six geographically distant institutions across the state of Louisiana, where each site contains multiple data archives from different science domains.

The organization of this paper is as follows: we pro-

vide background information on ontology-based metadata in section 2; we present the motivating applications for this framework in section 3; we explain our system design and architecture in section 4; we give the details of our implementation on PetaShare in section 5; and finally we conclude in section 6.

## 2 ONTOLOGY-BASED METADATA

Metadata refers to information about data itself, commonly defined as "data about data", and it is essential for Data Grid systems. Without proper metadata annotation, the underlying data is meaningless to scientists. Different approaches based on conceptual modeling techniques are available for building a metadata management service. Controlled vocabulary, schema, and ontologies provide an increasing level of description based on agreements concerning the meaning of terms, allowable data hierarchies, and the overall data model. In this section, we discuss the issues of using ontology to build a metadata management framework for scientific data sets in Data Grids.

Ontology is a data model that explicitly represents a set of concepts and relationships in a particular domain, defining which entries exist and how they relate to each other (Gruber, 1993). One common use case is the Gene Ontology (GOP, 2008) in which a structured representation of gene functions is used in a uniform way to be queried across different gene databases. Gene Ontology is an important collaborative effort and it is arranged in a hierarchical manner using directed acyclic graphs. A controlled vocabulary is provided by analyzing the semantic structure of the data and then implementing a uniform representation of metadata information. The metadata can be queried at different levels over many databases that span the world (GOP, 2008).

One of the challenges in preparing an infrastructure for metadata management is to automate the process of metadata formation. The data is formed by professionals in the area who have knowledge about the structure. The ambiguous vocabulary of the data leads to difficulties in querying metadata systems to search for information. Incorrect outcomes could be obtained either by returning too general terms or too deep terms according to the search criteria (Shatkay and Feldman, 2003; Pérez et al., 2004; Couto et al., 2007). However, a well designed semantic structure can enhance the metadata system by providing better precision in the search process. The ontology based semantic metadata explained in this study helps to cover and represent the majority of the concepts inside the data in order to automate the integration of query process with generation, classification and establishment of connections of actual metadata

Based on description logic, ontology describes the concepts and relations that can exist for an agent or a community of agents in a given domain. Generally it consists of taxonomic hierarchies of classes and the relations between these classes. Ontology has two key advantages compared with traditional data modeling techniques: first, it has more expressive power than other traditional data model techniques; secondly, efficient reasoners are available to perform constraint verification and checking.

The expressive power of ontology can improve knowledge sharing and data integration. Knowledge sharing refers to sharing knowledge or information between members of an organization. Data integration means the provision of a uniform interface to a multitude of data sources (Levy, 1999). Data integration is especially important in a multiple-domain collaborating environment, where physical data sets can be geographically and administratively heterogeneous; the data models and format can be heterogeneous; and the semantics of vocabularies describing the contents of the data sets can be highly heterogeneous among different domains. Data Grid software such as the Storage Resource Broker (SRB) (Baru et al., 1998) and iRODS (SDSC, 2008) have successfully addressed problems related to storage system heterogeneity and administrative heterogeneity. XML based approaches have helped in solving data format issues.

Ontology can be used to map concepts across different knowledge stores and thus integrate the knowledge stores effectively for scientific applications. Local as View (LaV) and Global as View (GaV) are two main approaches to perform data integration (Charathe et al., 1994) (Levy et al., 1996). In LaV, the content of the source schema is described as a query over the global schema; in GaV, the relation in the global schema is defined as a view over the source schemas. LaV has the advantage of better scalability, on the other hand, the maintainability of GaV is better. Research, e.g. (Levy, 1999) (Calvanese et al., 2001), has been conducted on formal logic enabled data integration and some key challenges have been identified. However, practical systems capable of integrating semantically different knowledge stores using semantic tools are uncommon.

A key advantage of the ontology model is the reasoning capability it provides. Constraints can be constructed on top of the concepts and relations in an ontology. Such constraints may or may not be consistent with the ontology model. The relationships among instances inside an ontology model could also be contradictory. An ontology reasoner can be used to check the consistency between concepts in an ontology model or whether a given instance is allowable in the ontology model. High performance reasoners, such as FaCT++ (Tsarkov and Horrocks, 2006), Racer (Haarslev and Moller, 2001), have been implemented by the ontology research community.

Ontology data modeling has the potential to bring great benefit for scientific data management. Stuckenschmidt et al. proposed integrating different domain ontologies for data integration (Stuckenschmidt et al., 2000) and identified different mapping approaches for concepts in different ontologies. The Pegasus group developed a virtual metadata catalog which provides semantic rich information for metadata catalogues (Gil et al., 2006). They integrated data sets from three disciplines by constructing one vir-

tual metadata catalog that hides all the underlying distributed domain ontologies from the query mediator. (Jeffrey and Hunter, 2006) developed an ontology enabled semantic search engine for the SRB/MCAT system to handle heterogeneous data sources. Their system allows users to load different ontology instance data sets into an mySRB interface enabling user searches on heterogeneous ontology repositories.

We argue that data integration using ontology can be futher refined by applying different policies for different ontology concepts. In a distributed environment where domain schemas are constantly evolving, such a framework is more flexible than enforcing all the domain concepts maps to the virtual data catalog. The ontology schema and the ontology instance data should also have different update policies. The ontology reasoner can be used for verification to improve data quality and increase scalability. In the following sections we describe a framework that addresses these issues.

## 3 GUIDING APPLICATION SCENARIOS

Numerical simulations play an increasingly important role in modern scientific and engineering research. They are used as an important tool for scientific investigation particularly when it is impossible or too expensive to perform experiments. With growing computing power, scientists can routinely perform simulations with unprecedented scale and accuracy. The scale and number of simulations can generate huge amounts of data. These data sets are then used for analysis and/or visualization by large distributed collaborations. Effectively storing these data sets and then enabling users to retrieve them are of great importance to scientific research.

At the Center for Computation & Technology, LSU, we are building data archives for various research disciplines, including coastal modeling, astrophysics, and petroleum engineering, along with a separate archive for visualizations and other digital media. Most of the data sets maintained in these archives are related to numerical simulations.

**Coastal Modeling - SCOOP Archive.** The SURA Coastal Ocean Observing and Prediction (SCOOP) program is building a modeling and observation cyberinfrastructure to provide new enabling tools for a virtual community of coastal researchers. Two goals of the project are to enable effective and rapid fusion of observed oceanographic data with numerical models and to facilitate the rapid dissemination of information to operational, scientific, and public or private users (MacLaren et al., 2005). As part of the SCOOP program, the team at LSU has built an archive to store simulation and observational data sets. Currently the archive contains around 300,000 data files with a total size of around 7 Terabytes. Three main types of data files are held in the archive: wind files; surge (water height) files; and data model files. The basic metadata information for these files are: the file type, the model used to generate the file, the institution where the file was generated, the starting and ending date for the data, and other model related information.

**Astrophysics - NumRel Archive** The Numerical Relativity group at LSU is building an archive of simulation data generated by black hole models. One of the motivations is to analyze experimental data from gravitational wave detectors such as LIGO. These simulations are typical of many other science and engineering applications using finite element or finite difference methods to solve systems of partial difference equations. The simulations often take many CPU hours on large supercomputers and generate huge volumes of data. Software packages such as Cactus (Goodale et al., 2003) enables scientists to develop their code in a modular fashion. Each numerical library in the package defines a set of attribute names which can be used as controlled vocabulary. The attribute names could describe input parameters or computation flags. Such information is crucial for user's later retrieval.

**Petroleum Engineering - UCoMS Archive.** Reservoir simulations in petroleum engineering are used to predict oil reservoir performance. This often requires parameter sweeping, where large numbers (thousands) or runs are performed.

In this scenario, users need to provide the initial range of parameter settings. In such a setting, the important metadata can be expressed as follows: parameter name; the range of the parameter in the simulation; the particular parameter value which is set for the run.

**Visualization - DMA Archive.** Scientific data, after being generated by simulations, needs to be further analyzed. One important tool to help scientists is visualization. The Digital Media Archive (DMA) at CCT is being built to store the resulting images from scientific visualization, along with other media such as movies, sound tracks, and associated information. Visualization metadata can be fairly simple: Image Name, Image Size, Image Width, Image Height, and File Format.

## 4 SYSTEM DESIGN

We propose a semantic enabled metadata framework for Data Grid systems. The framework addresses two key issues: (i) the integration of semantically heterogeneous data stores using ontology techniques; (ii) the construction of meaningful constraints for data verification and quality auditing using logic inference tools. We use current data sets in the different CCT data archives to provide test cases.

The metadata system will be coupled with a storage system to provide the necessary data grid functionality. We envision our system to work with iRODS (a Rule Oriented Data System also termed the next generation SRB) as it provides a coherent view over the heterogeneous underlying physical storage hardware. The metadata system should provide a friendly user interface and a set of API functions. The metadata service will take a user query and answer with a set of logical file names when applica-

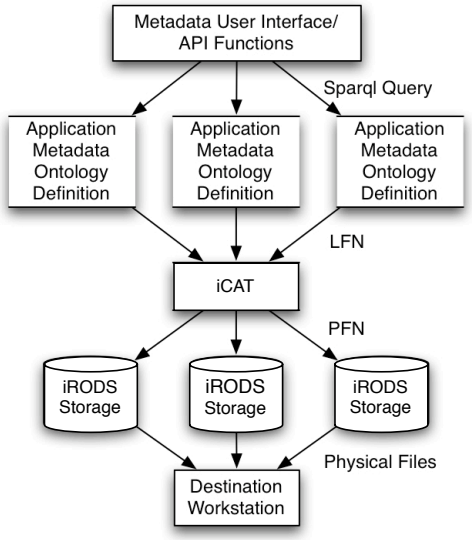


Figure 1: Basic architecture for semantic enabled Data Grid

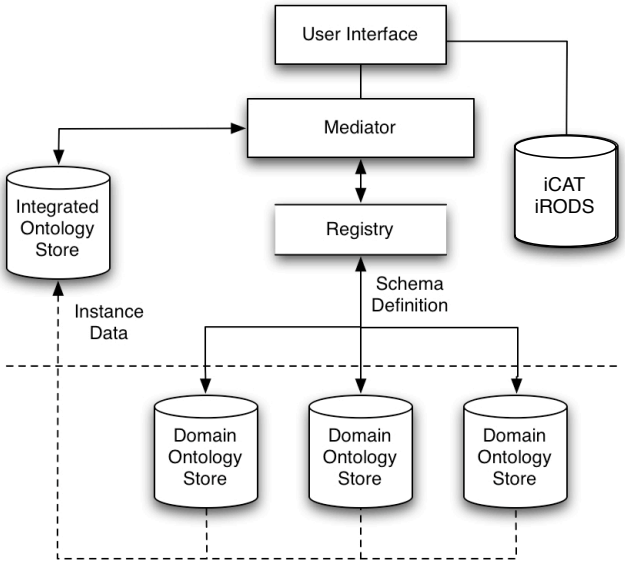


Figure 2: Data integration architecture

ble. Such logical file names are then resolved by iCAT, which offers a replica metadata service and is tightly coupled with iRODS. The actual physical file is then fetched from iRODS to the user. Figure 1 shows the architecture of our system.

### 4.1 Data integration

Data integration means providing a uniform interface for users. Such an interface should be independent of the underlying data schema.

The information stores we plan to integrate are highly heterogeneous: (i) The physical storage geographical locations, administrative domains, and the storage systems are heterogeneous; iRODS can address this problem; (ii) The data type, format, vocabularies are heterogeneous; for each given domain, scientists from different organizations may use completely different controlled vocabulary to describe or annotate the data; (iii) The semantic meaning are heterogeneous; in a collaborating environment, scientists from different domains need to work together and understand each other.

The integrated information store should be able to answer queries similar to the Local as View (LAV) approach we described above. The user needs not be aware of the underlying schema differences. The knowledge store will be responsible for query reformulation, making the local queries and submit them accordingly.

Due to the complexity of the problem, it will be unrealistic to hope one system can solve all the heterogeneity we mentioned above uniformly. Various approaches have been proposed to integrate data stores using ontology. We argue that in a real world setting, there are certain information which can be easily integrated, and there are information which is very difficult to be integrated, these situations need different handling mechanism.

We designed a system which treat the differences with different policies. As shown in Figure 2, the central component in the system is called the mediator. The mediator has access to an integrated ontology store(IOS) which will store the common objects, their relationships, and the instance data. The domain ontology stores (DOS) are distributed and will be maintained by domain experts. The DOSs publish the common objects, their relationships, and their instance data to the IOS. The mediator also maintains a registry, which contains the mappings between all the DOS and the concepts each DOS knows how to handle, Figure 3 shows the relationships between ontology stores in our current implementation.

The common concepts to be stored in IOS are in the different underlying domains and have an explicit relationship between each other. Only the concepts that are completely free of inter-domain relationships should be preserved in the DOS.

When a user enters a query, the mediator analyzes the query first. It is possible that the query can be answered by the integrated ontology directly, one or more underlying ontologies, or a combination of these two. Query reformulation is needed if the query can not be completed answered by the integrated ontology.

The query reformulation is based on the registry information published by the underlying ontologies. Naively it publishes a set of concept names, which are the vocabularies it can understand. When the mediator sees the terms or concepts which it can not handle in integrated ontology, it looks up the registry and assign the query to the ontologies which claimed can answer it.

The ontology store has two major components: the

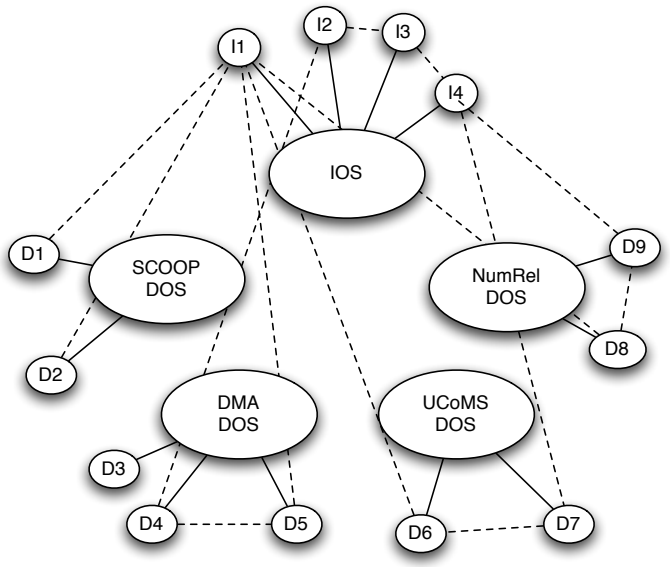


Figure 3: Relationship between ontology stores

schema which defines the domain concepts and relationships, and the instance sets which are instantiated from the domain concepts. The IOS needs the instance data to perform query answering. The registry for mediator does not need to maintain instance data, since the query will be answered by the domain ontology store.

The system should be constructed with a set of DOSs in mind. A set of common concepts which can be integrated into IOS are chosen first. The distributed DOSs publish the instances data of this common sets to the IOS. The distributed DOS also needs to publish other keywords which it can handle query to the mediator's registry.

When a new ontology store is added, the knowledge store administrator needs to identify the relevant common concepts in the new store, and map these concepts to the system's IOS. By doing this the new instance data which enters the new store will get published into the IOS as well. Other than this, the new store also needs to register what terms it can answer by publishing to the IOS's registry.

When the concepts and relationships in certain knowledge domain changes, the DOS needs to notify the mediator about its changes. The mediator decides whether the change needs to be populated to IOS or not. If there is such needs, the mediator notifies the DOS, and DOS will publish those instance data to the IOS.

### 4.2 Logical inference and constraint verification

One of the key advantages ontology holds over traditional knowledge representation technologies is its stronger reasoning capabilities. Being able to perform logical reasoning on ontology not only provides domain experts and system developers with a necessary tool to check and verify logical consistency, but also potentially increases scalability.

As discussed in previous sections, the key concept in our data integration model is the Mediator, which will handle a set of high level concepts commonly agreed upon by the underlying distributed ontology stores. On the other hand, certain implicit connection between ontology from different domains may not be able to be explicitly expressed in the ontology because of the heterogeneous and dynamic nature of the domain involved. Logical reasoning based on a core set of concepts and relations can be performed to discover these hidden connections, thus ensure that scientists get all the knowledge he needs from all the domains available when he performs the search.

### 4.3 Constraint verification

As the size of the ontology grows, it becomes increasingly unrealistic for a human to check the correctness of the knowledge stored in the ontology. As a result, certain logical constraints need to be introduced to ensure the constructed ontology is logically sound. To do this, two questions need to be answered: (i) How to represent the logical constraint? (ii) How to perform the reasoning?

For the first question, there are several ways to represent the constraints as logical rules that the ontology must follow, such as SWRL (2008) submitted by the National Research Council of Canada, Network Inference and Stanford University, SWRL provides a high-level abstract syntax for Horn-like rules in OWL DL and OWL Lite; Racer (Haarslev and Moller, 2001) also offers partial SWRL support through its SWRL-based rule language. Besides SWRL, DLP (Description Logic Programming) (Motik et al., 2007) and various semantic web service rule languages WSMO (2008) and RuleML (2008) also can be used to represent logical rules. After consideration, we decided to use SWRL to describe our constraints because SWRL combines OWL and RuleML and integrates constraints into ontology. Further, Protege, currently the most popular ontology development platform, provides an extension for SWRl that can greatly reduce the workload of developing and testing constraints on existing OWL-based ontology.

Beside constraint representation, different reasoners provide different levels of reasoning capabilities. Chief among them are Racer (Haarslev and Moller, 2001) developed at University of Hamburg, Pellet (Sirin et al., 2005), developed by University of Maryland's Mindswap Lab, Jess (Friedman-Hill, 2008), developed at Sandia National Laboratories, etc. These reasoners have different degrees of shortcomings. Racer, as the most prominent reasoner is also the most mature with the strongest reasoning capabilities, but its current status as a commercial product limits its interoperability with some of the other important products needed in the development ontology, such as Jena (2008). Pellet is open source, which makes it the recommended reasoner of Jena, but currently Pellet only supports an subset of SWRL, and it doesn't support SWRL built-in functions. On the other hand, as far as we know, Jess is the reasoner with the fullest SWRL support available right now, also Jess can be incorporated into Protege

$$Rule1: SCOOP\_File(?x) \wedge Name(?x,?y) \wedge swrlb:startsWith(?y,"S") \longrightarrow SurgeExisted(?x,true)$$

$$Rule2: SCOOP\_File(?x) \wedge Name(?x,?y) \wedge swrlb:startsWith(?y,"W") \longrightarrow WindExisted(?x,false)$$

$$Rule3: SurgeExisted(?x,true) \wedge WindExisted(?y,false) \wedge Name(?x,?z) \wedge Name(?y,?r) \wedge swrlb:contains(?z,?r) \longrightarrow isValid(?x,true)$$

Figure 4: SWRL rules

to check and verify the SWRL rule added into ontology. So at this stage, we plan to use Jess as the underlying inference engine to test ontology verification capabilities we intend to introduce into our data integration application in the future.

In our current implementation, there are several scenarios under which logical inference will be needed to ensure the consistency of the ontology. For instance, in the SCOOP ontology, each instance of a surge file requires a corresponding wind file to ensure its validity. According to the naming convention agreed upon by participants of the SCOOP project, the name of the surge file will begin with "S"; and name of the wind file will begin with "W"; At the same time, the file name of the surge file will contain the file name of its corresponding wind file. For example, if there is a surge file named SWW3LLFNBIO_WANAFe01-UFL_20050825T0000_20050826T1300_20050826T1300_12hs-T272_Z.txt, the name of its corresponding wind file would be WANAFe01-UFL_20050825T0000_20050826T1300_20050826T1300_12hs-T272_Z.txt. That feature of the SCOOP ontology will be used to check the validity of surge files, thus ensuring the logical consistency of the SCOOP ontology.

The three SWRL rules shown in figure 4 were written in Protege's SWRL tab extension, and they took advantage of the SCOOP naming convention to check the existence of a wind file corresponding to a particular surge file. Our initial experimental run on Jess showed that they can ensure that all the valid Surge instances of SCOOP ontology can have their property isValid set true, thus making sure the scientists using our Data Grid middleware will be informed about the validity of the files they are dealing with.

The SCOOP surge file validity is just one example for which logical inference will be needed for constraint verification. As the size and complexity of our ontology store grows, other scenarios requiring constraint verification will likely emerge. As our initial experiment shows, SWRL and Jess are capable of handling the current requirement of constraint verification. We will continue our effort of incorporating inference in our future implementation while at the same time, investigate other possible alternative reasoner and rule representing languages that may offer better capabilities to ensure that our ontology stores are properly verified.

#### 4.3.1 Inference and increased scalability

Beside ontology verification, inference can also be used to reduce the hurdle to add new ontology. Since the domain ontology will be maintained separately by domain experts distributed across different institutions and geographic areas, and they may know little about other domain ontology, it may be difficult to establish semantic connections among concepts existed in multiple domain ontology and described in different terms. Inference can be employed to connect multiple domain ontology via the mediator ontology. What we envision is to create a mediator ontology that will be agreed upon by all the involved domains to represent the common concept and relationships existed in more than more domains, current underlying domain ontology only needs to negotiate with the authority responsible for the creation and maintaining of the mediator ontology. New ontology expected to join the ontology store also need to negotiate with mediator to establish connection between them and the mediator ontology. As a result, the effort needed to enlarge the ontology store will be reduced, and scalability of our Data Grid middle-ware will then be increased. When scientists perform search, inference can be performed on multiple domain ontology via the mediator ontology so that all knowledge related to the search term, even if the knowledge came from a totally different domains, can be retrieved from ontology store.

The above discusses some very preliminary ideas we have in term of increasing scalability and reducing the difficulty for new domain ontology to join. As of now, feasibility of our idea has not yet been tested, our future research will involve experiments to test the feasibility and do-ability of our ideas, if successful, it will be included in our future implementations.

## 5 CURRENT IMPLEMENTATION

In our initial implementation, there are three basic components: Web Interface, Query Translator and Ontology Base, corresponding to User Interface, Query Translator and Ontology RDF Store respectively.

Among their functions, the Web Interface, mainly serves to provide the user with a easy-to-use interface, communicate with Query Translator, display the query result in a easy-to-understand manner and provides various filtering/sorting capabilities. The objective is to create a query engine interface with maximum dynamism to increase usability. To meet this objective, in the current implementation, various AJAX technologies are employed and certain level of dynamism has been injected into the interface design.

The ontology base for our system is constructed with several domain ontology stores. The concepts in each do-

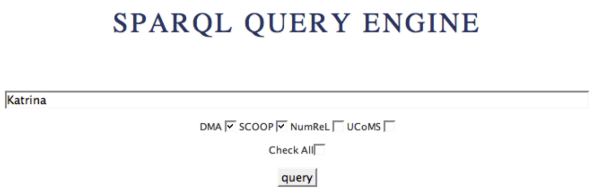


Figure 5: Ontology metadata query interface

main ontology can only be related to the concepts in the same ontology, or mapped to the concepts in the integrated ontology. There is no direct inter-domain concept mapping. The integrated ontology keeps track of all the inter-domain concepts and their mappings.

Currently, only metadata of SCOOP and DMA domains and their instances are put into the ontology. To integrate data from multiple domains, concepts that are available to and can be extended to multiple domains must be defined in the ontology and shared across multiple domain ontology. In the current implementation, only the shared concept "keyword" is defined. Keyword can carry different meaning for different ontologies. It usually annotates the distinct meaning which can be searched and classified. The "keyword" here is used to describe the content of files. Users can use various keyword to ask for files from SCOOP and DMA archives. Another common concepts is the time when the data file was generated, such concepts are common for the numerical simulation data sets and thus should be maintained in the integrated ontology store.

To query the ontology base and iRODS storage system, the query needs to be understood by these systems. On the other hand, a easy-to-use web interface is also necessary for users who aren't necessarily proficient in ontology query languages and iRODS commands. To bridge the divide between human understandable and machine understandable, Jena JENA libraries and iRODS scripts are used to translate user query into machine understandable ontology and iCAT query.

Finally, the Ontology Base stores the various domain ontologies, currently, only metadata of SCOOP and DMA domains and their instances are put into the ontology. To integrate data from multiple domains, concepts that are available to and can be extended to multiple domains must be defined in the ontology and shared across multiple domain ontology. In the current implementation, only the shared concept "keyword" is defined, "keyword" here is used to describe the content of files. Users can use various keyword to query file information from SCOOP and DMA ontology. Users can also require the file to be fetched directly from iRODS server back to his local machines to display in user's web browser.

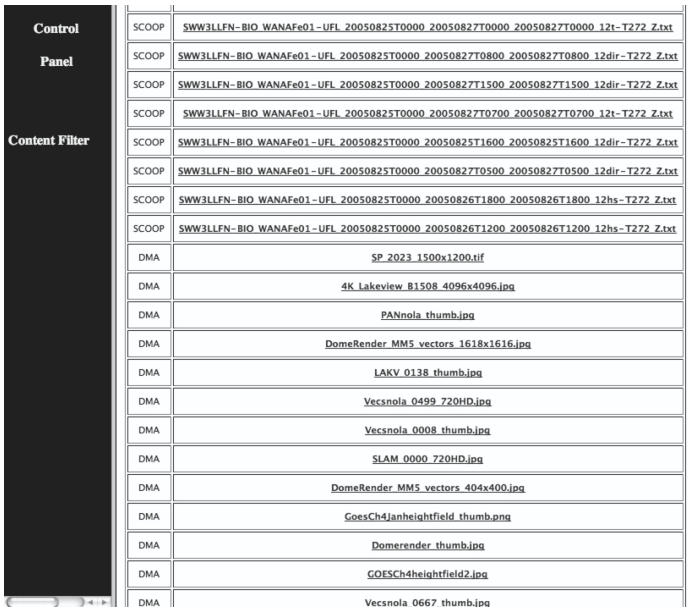


Figure 6: Ontology query result

### 5.1 Use Scenario

The primary objective of our research is to enable knowledge sharing among multiple disciplines and projects. Often data needed by the scientists belongs to different projects or disciplines. For example, in meteorology, raw observational/experimental data are often not enough, visualizations of model data are also needed to achieve better understanding. Suppose there is a meteorologist performing research on Hurricane Katrina using data collected in the SCOOP project, he may need to visualize SCOOP data to help him achieve his research objective. But there may not be the required visualization data in the SCOOP archive, and performing the visualization may require considerable time and effort or more expertise than the scientist has. Our current implementation provides a solution to the scenario mentioned above, the meteorologist can simply query "Katrina", as shown in Figure 5,

the query result will return data related to Hurricane Katrina in the SCOOP project, also data of visualized SCOOP data (from DMA archive) related to Hurricane Katrina will also be returned as part of the query result, as shown in Figure 6, the underlying Integrated Ontology Store will link together two conceptually distinct archives stored on different physical locations and enable cross-querying on both archives. After the query result is returned, the meteorologist can fetch the file, SCOOP or DMA, back remotely from iRODS servers via a web interface. Figure 7 shows a visualized Hurricane image fetched from iRODS servers.

### 5.2 PetaShare Data Grid

PetaShare is a state-level data sharing cyberinfrastructure effort in Louisiana. It aims to enable collaborative-

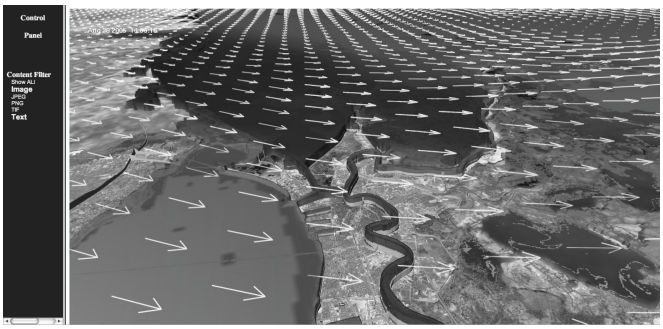


Figure 7: Data fetched from remote site

data-intensive research in different application areas such as coastal and environmental modeling, geospatial analysis, bioinformatics, medical imaging, fluid dynamics, petroleum engineering, numerical relativity, and high energy physics. PetaShare manages the low-level distributed-data handling issues, such as data migration, replication, data-coherence, and metadata management, so that the domain scientists can focus on their own research and the content of the data rather than how to manage it.

Currently, there are six PetaShare sites online across Louisiana: Louisiana State University, University of New Orleans, University of Louisiana at Lafayette, Tulane University, Louisiana State University-Health Sciencies Center at New Orleans, and Louisiana State University-Shreveport. They are connected to each other via 40Gb/s optical network, called LONI (Louisiana Optical Network Initiative). In total, we have 250TB of disk storage and 400TB of tape storage on these sites.

PetaShare brings the idea of a data-aware system model which includes a data-aware scheduler, Stork (Kosar and Livny, 2004), resource allocation and resource selections services, higher lever planners, and workflow managers. Due to the huge data requirements of current scientific applications, there has been extra effort to provide efficient data access methods favoring application performance while effectively utilizing the system and network recourses. PetaShare introduces the data management subsystem to be the I/O module in distributed computing systems (Balman et al., 2008).

### 5.3 Ontology Integration into PetaShare

At each PetaShare site, we have an iRODS server deployed, which manages the data on that specific site. Each iRODS server communicates with a central iCAT server that provides a unified name space across all PetaShare sites. The clients can access PetaShare servers via three different interfaces: petashell, petafs, and pcommands. These interfaces allow the injection of semantic metadata information (i.e. any keywords regarding the content of the file) to the ontology whenever a new file is uploaded to any of the PetaShare sites. The physical metadata information (i.e. file size and location information) is inserted to iCAT using the iRODS API. This is illustrated in Figure 8.

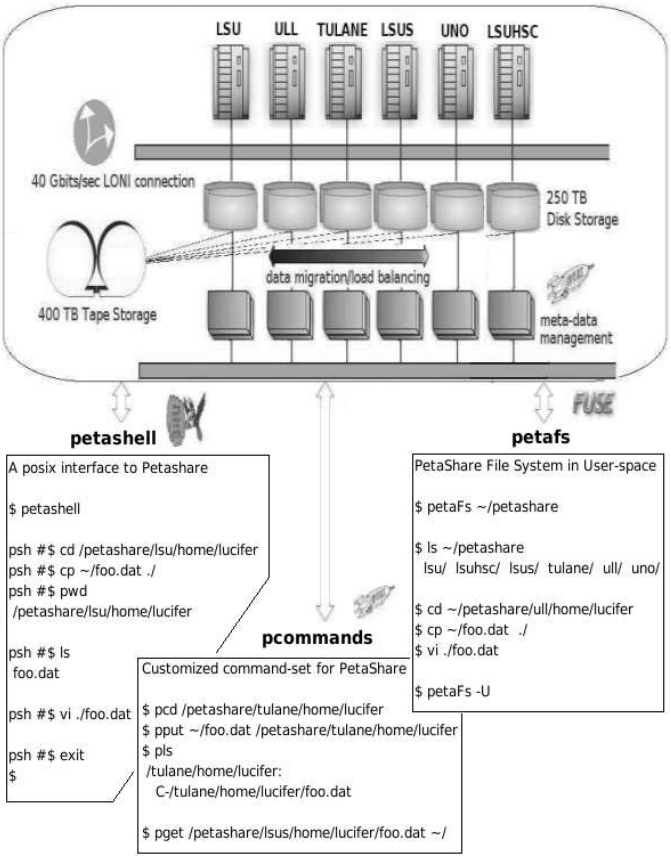


Figure 8: PetaShare architecture

When a user requests an operation on a file, the iCAT server provides information on where the file resides physically, and provides metadata information regarding that file. An uploaded file can be accessed via different logical paths on each site. As an example, a file called 'share-this-file' can be physically located at LSU PetaShare resource (/petashare/lsu/home/user1/share-this-file), and it can be accessed over a different PetaShare resource (e.g. /petashare/tulane/user1/share-this-file). This is illustrated in figure 9. In this illustration, a user wants to upload a file to the PetaShare resource in LSU (step 1). The iRODS server at LSU sends metadata information (physical location of file, size of file, permissions, user-defined metadata) of the file to the iCAT server (step 2). Eventually, file is stored on PetaShare resource at LSU (step 3). Another user at LSUS site wants to download this file and sends a request to iRODS server (step 4). iRODS server translates this request, and queries the iCAT server regarding the file (step 5). iCAT server returns a physical location(and metadata) of the file(step 6). Then, iRODS server asks for the file(step 7) and corresponding file is sent to iRODS server back(step 8). Finally, iRODS server returns the file to the user(step 9).

One major objective of the PetaShare architecture is to enhance the overall performance while maintaining a cyberinfrastructure for easy and efficient storage access. In

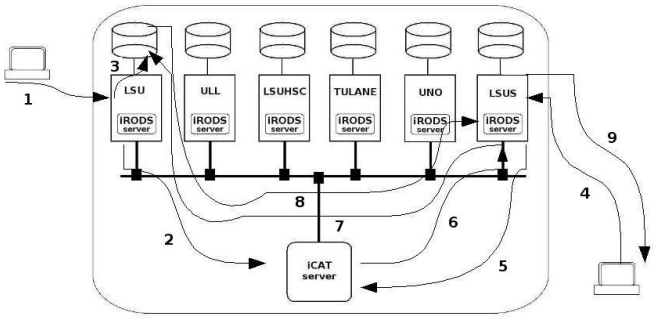


Figure 9: Interaction among components in PetaShare

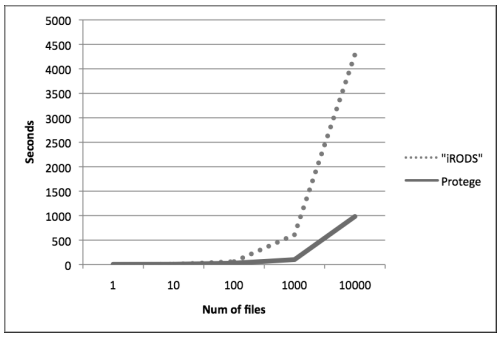


Figure 10: Performance Comparison of Metadata Insertion between iRODS and Protege Database

our framework, the upper level metadata management also returns information about number of replicas and location of each element in the data archive by querying the iRODS module. The ontology based metadata search gives logical filenames that match the semantic search criteria, and then, each logical file name has corresponding set of physical file names, for the searched data entity. One other benefit of our framework is that we do take advantage of locating multiple replicas while transfering data using multiple archives for both reliability and performance.

The data-aware scheduler, Stork, is responsible for organizing the data placement activities in PetaShare project, especially for data movement activities between geographically distributed data archives. Replica information along with the data transfer job enables data placement scheduler to make better decisions. One key factor is that data transfer operation is performed over multiple connections by accessing to multiple data storage servers; such that, we gain performance by using more than a single connection to data archives (Balman and Kosar, 2007).

Replicated files provide a redundant environment in PetaShare architecture. Although PetaShare also offers a location independent data access mechanism, the overall framework will not be affected by site failures through the use of replica management. The integration of replica information into the metadata framework increases the reliability of the overall structure and also enhances the performance of data transfers by distributing the load into multiple data storage servers.

Semantic analysis of data is not limited to locating files corresponding to a search criteria. One recent study (Balaji et al., 2008) uses metadata information along with the semantic structure of the data to reduce the size of the actual data files and re-generate the data in the remote site instead of transfering the file from the data archive. Semantic compression has a broad perspective and it is successfully applied to compact database files (Jagadish et al., 2004). Since data generated by scientific applications are usually structured, semantic analysis and semantic compression are important research topics in Data Grids in terms of performance issues.

### 5.4 Performance Evaluation and Comparison

In our current implementation, there are two ways to enable semantic metadata in Petashare:

(i) take advantage of iRODS's built-in metadata support and our far more semanticly richer metadata into iRODS's metadata system. Advantages of this approach includes potentially better query performance because of fewer layers a query has to go through; This approach also comes with disadvantages, namely, the added requirement to figure out a way to encode semantic metadata into iRODS's metadata system which, so far, only supports simple query over triples.

(ii) build a middleware between currently mostly Java-based ontology infrastructure and traditionally C-based iRODS. Advantages of this approach includes more established support for ontology insertion, modification, merging and query which our system can readily tap into. Disadvantages may involve develop a whole set of tools required to bridge the inherent differences, also our preliminary testing of the browser based ontology system indicated less than satisfactory performance.

To make a better determination on which approach offers better performance, we did some preliminary benchmarking on two different experimental implementation we have done so far. We picked a test case involving the insertion of from 1 to 10000 set of metadata corresponding to 1 to 10000 experimental files produced by the SCOOP project. As shown in figure 10, as the size of insertion metadata set grows, the system based on Protege database displays far superior performance than the system built on iRODS metadata, the performance discrepancy turned out to be a surprise for us as Protege database is required to handle semanticly far more complicated data. Our preliminary conclusion is that the iRODS based system handles metadata insertion by repeatedly inserting triples into databases, while the system based on existing ontology tools, namely Protege database, handles large set of metadata insertion by bundling them together in the memory, then bulk-insert them into the database. Further investigation is needed to determine the exact cause of the performance differences we witnessed.

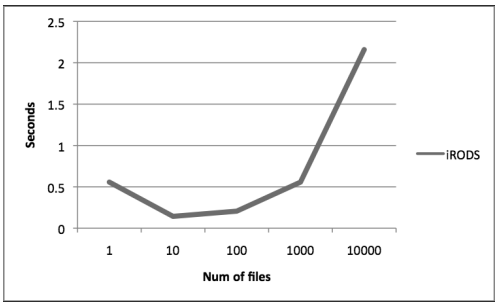


Figure 11: iRODS Query Performance

Although our preliminary performance evaluation indicates the second approach delivers better performances, we decided to take the first approach in our current implemented Petashare system because of the complicity involved in building a functioning middle-ware between iRODS and Protege based semantic metadata system.

Figure 11 illustrates the query performances of our current iRODS-based metadata system, our test cases ranges from query result set of size from 1 to 10000 files, our current system delivers performances ranging from less than 0.2 second to more a little bit more than 2.2 seconds, an acceptable performance in our current implementation.

## 6 CONCLUSION and FUTURE WORK

Metadata management for scientific data faces new challenges as cross-domain collaboration improves in modern science. We designed a metadata system which provides a mechanism to integrate data sets from heterogeneous domains. We showed the initial implementation of this system by applying inference on constraint verification. We have tested our system on a small set of files with limited number of domains and domain concepts. We discussed our attempt to integrate the ontology metadata system into the Petashare Data Grid and showed some preliminary performances benchmarking we did on different approaches we may take. We also did benchmarking on our metadata system already implemented on Petashare data grid. There are still issues that need to be addressed. We need to test the scalability of our framework. We predict there will be performance bottlenecks in terms of ontology persistence and inference. We need develop appropriate middleware to bridge the differences between Petashare and current generation of ontology developing tools. There is also the question of how much inference can be done to increase scalability by alleviating the difficulty of adding new domain ontology and how much negative effect the resulting additional overhead will have on system performance. Another open problem is how to integrate such a system with other metadata-intensive grid software, such as workflow management and provenance systems. These issues need to be further investigated.

## 7 ACKNOWLEDGEMENTS

This project is in part sponsored by the National Science Foundation under award numbers CNS-0619843 (PetaShare) and EPS-0701491 (CyberTools), PHY-0701566 (XiRel), by DOE under award number DE-FG02-04ER46136 (UCoMS), by the Board of Regents, State of Louisiana, under Contract Numbers DOE/LEQSF (2004-07), NSF/LEQSF (2007-10)-CyberRII-01, and LEQSF(2007-12)-ENH-PKSFI-PRS-03, and by the SURA SCOOP program. The authors would also like to thank Dylan Stark for his generous help in building the SCOOP and NumRel ontologies.

## REFERENCES

Atlas (2008). European Organization for Nuclear Research(CERN) A Toroidal LHC ApparatuS(ATLAS) Project. http://atlasexperiment.org/.

Balaji, P., chun Feng, W., Archuleta, J., Lin, H., Kettimuthu, R., Thakur, R., and Ma, X. (2008). Semantics-based distributed i/o for mpiblast. In *PPoPP '08: Proceedings of the 13th ACM SIGPLAN Symposium on Principles and practice of parallel programming*, pages 293–294, New York, NY, USA. ACM.

Balman, M. and Kosar, T. (June, 2007). Data scheduling for large scale distributed applications. In *the 5th ICEIS Doctoral Consortium, In conjunction with the International Conference on Enterprise Information Systems (ICEIS'07)*. Funchal, Madeira-Portugal.

Balman, M., Suslu, I., and Kosar, T. (2008). Distributed data management with petashare. In *MG '08: Proceedings of the 15th ACM Mardi Gras conference*, pages 1–1, New York, NY, USA. ACM.

Baru, C., A., R. M., Rajasekar, and Wan, M. (1998). The SDSC Storage Resource Broker. In *Proceedings of CASCON*, Toronto, Canada.

Calvanese, D., Giacomo, G., and Lenzerini, M. (2001). Description logics for information integration. *Computational Logic: From Logic Programming into the Future, Springer–Verlag*.

Charathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J., and Widom, J. (1994). The TSIMMIS project: Integration of heterogeneous information sources. In *Proceedings of 10th Meeting of the Information Processing Society of Japan*, Tokyo, Japan.

CMS (2008). PPDG Deliverables to CMS. http://www.ppdg.net/archives/ppdg/2001/doc00017.doc.

Couto, F. M., Silva, M. J., and Coutinho, P. M. (2007). Measuring semantic similarity between gene ontology terms. *Data Knowl. Eng.*, 61(1):137–152.

Friedman-Hill, E. (2008). Jess - The Rule Engine for Java Platform. http://herzberg.ca.sandia.gov/jess/.

Gil, Y., V.Ratnakar, and E.Deelman (2006). Metadata Catalogs with Semantic Representations. In *Proceedings of IPAWS*.

Goodale, T., Allen, G., Lanfermann, G., Massó, J., Radke, T., Seidel, E., and Shalf, J. (2003). The Cactus Framework and Toolkit: Design and Applications. In *Vector and Parallel Processing - VECPAR '2002, 5th International Conference*. Springer.

GOP (2008). The Gene Ontology Project. http://www.geneontology.org/.

Gruber, T. R. (1993). A Translation Approach to Portable Ontologies. *Knowledge Acquisition*, pages 199–220.

Haarslev, V. and Moller, R. (2001). Description of the RACER System and its Applications. In *Proceedings of International Workshop on Description Logics (DL-2001)*, pages 1–2, Stanford, CA USA.

Jagadish, H. V., T., R., Chin, B., Anthony, O., and Tung, K. (2004). Itcompress: An iterative semantic compression algorithm. In *ICDE '04: Proceedings of the 20th International Conference on Data Engineering*, page 646, Washington, DC, USA. IEEE Computer Society.

Jeffrey, S. J. and Hunter, J. (2006). A Semantic Search Engine for the SRB.

Jena (2008). A Semantic Web Framework for Java. http://jena.sourceforge.net/.

Kosar, T. and Livny, M. (2004). Stork: Making data placement a first class citizen in the grid. In *ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing Systems (ICDCS'04)*, pages 342–349, Washington, DC, USA. IEEE Computer Society.

Levy, A., Rajaraman, A., and Ordille, J. (1996). Querying heterogeneous information sources using source descriptions. In *Proceedings of the Twenty-second International Conference on Very Large Data Bases (VLDB'96)*, Mumbai, Indian.

Levy, A. Y. (1999). Logic-based techniques in data integration. In *Workshop on Logic-Based Artificial Intelligence*, Washington, USA.

MacLaren, J., Allen, G., Dekate, C., Huang, D., Hutanu, A., and Zhang, C. (2005). Shelter from the storm: Building a safe archive in a hostile world. In *On the Move to Meaningful Internet Systems*.

Motik, B., Volz, R., and Bechhofer, S. (2007). DLP: Description Logic Programs. http://kaon.semanticweb.org/alphaworld/dlp.

Pérez, A. J., Perez-Iratxeta, C., Bork, P., Thode, G., and Andrade, M. A. (2004). Gene annotation from scientific literature using mappings between keyword systems. *Bioinformatics*, 20(13):2084–2091.

RuleML (2008). Rule Markup Language. http://www.ruleml.org/.

SDSC (2008). San Diego Supercomputer Center iRODS project. https://www.irods.org/.

Shatkay, H. and Feldman, R. (2003). Mining the biomedical literature in the Genomic era: an Overview. J. Comput. Biol. v10 i6. 821-855.

Sirin, E., Parsia, B., Bernardo, Grau, C., Kalyanpur, A., and Katz, Y. (2005). Pellet: A practical OWL-DL Reasoner. UMIACS Technical Report.

Stuckenschmidt, H., Wache, H., Ogele, T., and Visser, U. (2000). Enabling technologies for interoperability. In *Workshop on the 14th International Sysposium of Computer Science for Environmental Protection*, Bonn, Germany.

SWRL (2008). World Wide Web Consortium. SWRL: A Semantic Web Rule Language Combing OWL and RuleML. http://www.w3.org/Submission/SWRL/.

Tsarkov, D. and Horrocks, I. (2006). FaCT++ Description Logic Reasoner: System Description. In *Proc. of the Int. Joint Conf. on Automated Reasoning, Springer*.

WSMO (2008). The ESSI WSMO working group. Web Service Modeling Ontology. http://www.wsmo.org/index.html.