Design and Implementation of Metadata System in PetaShare

Xinqi Wang and Tevfik Kosar

Department of Computer Science and
Center for Computation & Technology
Louisiana State University
Baton Rouge, LA, 70803, USA
{xinqiwang,kosar}@cct.lsu.edu
http://www.cct.lsu.edu

Abstract. As the size of scientific and commercial datasets grows, it becomes imperative that an expressive metadata framework to be developed to facilitate access to the semantics of the data. However, the drive to enhance expressiveness and maximize performance often turns out to be contradictory. Hence, the design of metadata framework needs to take into consideration many factors in order to achieve balance between expressive power and performance. In this paper, we discuss the metadata framework developed for PetaShare and explain the design and implementation decisions we made in the process.

Key words: Metadata, Data Intensive Computing, Ontology, Protege, iRODS, PetaShare.

1 Introduction

As the size and complexity of the scientific and commercial datasets continue to grow, new data-centric scientific computing infrastructure needs to be built to satisfy the requirement. Among all the challenges involved in building such computing infrastructure, the need to reconcile the often conflicting requirement for expressive power and sufficient performance is quite challenging.

On the one hand, traditional metadata services have limitations in such application scenarios. Data from different disciplines is difficult to integrate due to lack of expressive metadata framework.

On the other hand, semantically rich representation scheme such as ontology-based semantic metadata provides rich expressiveness while maintaining decidability, but it is not designed for large scale scientific computing. As a result, much of the semantic web [7] technologies developed so far do not provide sufficient performances.

In this paper, we present the metadata system we developed for the PetaShare project. Our system seeks to take advantage of both semantically rich ontology representation of metadata and natively supported metadata represented in triples to provide access for users with different requirements.

We also discuss the choices we made and lessons we learned in the design and implementation process.

2 PetaShare

PetaShare [5] is a state-wide distributed data storage and sharing cyberinfrastructure effort in Louisiana. It aims to enable collaborative 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, PetaShare manages more than 250TB of disk storage and 400TB of tape storage across these sites.

At each PetaShare site, there is an iRODS [2] server deployed, which manages the data on that specific site. A Unix-like interface is available to access iRODS storage with commands like "ils", "icd", "ichmod" providing functionalities similar to their respective Unix counterparts, "itrim" for trimming number of replicas in iRODS, "irule" for submitting user defined rules and "imeta" for adding, removing, querying user defined metadata. Currently, each iRODS server communicates with a central iCAT [3] server that provides a unified name space across all the PetaShare sites. This centralized system will soon be replaced by a fully distributed and replicated server system. Clients, either human users or programs, can access the 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.

As part of the PetaShare project, we intend to develop a semantically-enabled metadata management and query system called petasearch. With petasearch, we intend to design an extendable metadata framework that gives a unified view over multidisciplinary datasets. We also plan to provide fast and efficient metadata query services for physically and conceptually distributed datasets of peta-scale. Protege [1] and iRODS-based metadata management and query system we are going to discuss in this paper are intended to serve as testbed for petasearch. Specifically, we seek to focus on performance issues we encountered during the design and testing of these two systems.

3 PetaShare Metadata System

We have implemented two metadata management and query systems in PetaShare, one is based on Protege [1] and and the other one based on iRODS.

As the current de facto standard of ontology design, Protege provides a complete set of tools in support of the design and implementation of ontology-based systems. But the complex nature of the ontologies and the implementation of Protege turn out to be inadequate to provide sufficient performance.

iRODS, on the other hand, is a integral part of PetaShare and it provides its own native metadata system, but the triple-based metadata representation is not powerful enough to satisfy the need for cross-domain metadata management.

Both systems provide some unique advantages the other system currently does not support. At the same time, both systems turned out to be inadequate on other fronts. There are two different approaches we considered to bridge these differences:

1. Take advantage of iRODS's built-in metadata support and try to integrate our semantically rich metadata into iRODS's metadata system. Advantages of this approach include 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 queries over triples.
2. Build middleware between current mostly Java-based Semantic Web infrastructure such as Protege where metadata will be modeled and stored and traditionally C-based iRODS where actual data archives are managed. 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 developing 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 better understand the advantages and deficiencies of the two approaches, we decided to implement two different metadata systems based on the two different approaches respectively. Our hope is at the current stage, the two systems can complement each other. Eventually, our development goal is to overcome the shortcomings of both systems and hopefully merge the two systems into a unified petasearch metadata system.

3.1 Protege-based Metadata System Framework

The reason we chose to implement the upper layer of our metadata system based on Protege-API and the Protege-based database back-end is to take advantage of the semantic expressive power of the ontologies. As the de facto standard for ontology design, Protege supports almost all the W3C standards and provides support for the whole range of ontology related functionalities, from graphic ontology design interface to built-in reasoner all the way to ontology serialization into relational database, which makes Protege and the Protege-related technologies good candidates for implementing a semantic enabled metadata system.

As shown in Figure 1, two different interfaces are available in our system. They are browser-based and commandline-based respectively. The purpose of browser-based metadata interface to PetaShare is to provide an easy-to-use, easy-to-understand method of access so that scientists can query and obtain small numbers of experimental files, while commandline-based interface can be combined with scripts and other programming tools so that more flexible, more powerful access to bulk files is also available in our system.

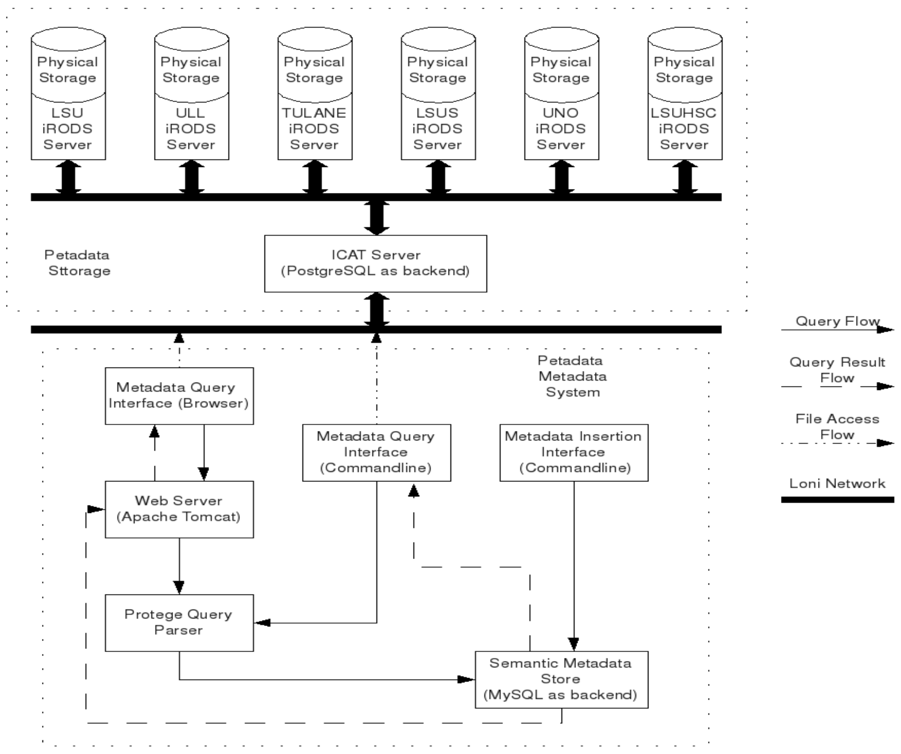

Fig. 1. PetaShare Metadata Framework for Protege

The core of the system consists of Protege Query Parser and Semantic Metadata Store. Protege Query Parser is implemented to parse queries entered by users into Sparql [6] queries understandable to the Protege query engine. In Semantic Metadata Store, metadata definitions in the forms of ontological classes and ontological instances are stored. Protege itself provides two ways of storing ontologies: file-based and relational database-based. The first approach essentially stores ontological classes and instance definitions to text files, although it is easier to implement and access file-based ontology, our experiment showed that file-based ontology can not scale to satisfy the data intensive requirements of PetaShare: attempts to insert metadata instances in excess of ten thousand resulted in insufficient memory error. Even though increase of physical memory size can partially alleviate this problem, but the fact that the Java Virtual Machine places limit on the amount of physical memory it can handle means text-based ontology can not scale as much as we want. Another problem is it often takes more than a dozen hours to load text-based ontology with more than ten thousand instances into memory. The causes of the failure to scale include:

1. The amount of memory required exceeds the maximum memory size the Java Virtual Machine is capable of handling.
2. System is saddled with too high a performance overhead as a result of large numbers of file accesses.

To overcome the above mentioned problems, we decided to take advantage of the second approach and store our ontology in regular relational databases. In our system, we chose MySQL as the back end database in which all metadata are stored in ontological form. We tested insertion of 1 million instances on a workstation with 4 GB memory. Each insertion consisted of creating 15 properties for a particular file in the archive. Our experiment showed that 1 million instances could be inserted in 6898 minutes 59 seconds, approximately 5 days. The relatively slow speed of the insertion owes partially to the overheads associated with representing complex modeling scheme such as ontology through tables in relational database. More experiments are needed to assess the upward limit of relational database-based ontology store.

Another part of our system is called Metadata-insertion interface. It is a Java-based commandline program that can be utilized, with the help of script languages such as perl, to automatically insert metadata about newly created experiment files.

For example, in large science experiments, when an experiment file is created, metadata-insertion interface can be triggered to automatically add appropriate metadata information, such as name, keyword, time of creation, file type, etc, into Semantic Metadata Store. The system administrator can also choose to do bulk-insertion. As of now, we have successfully inserted metadata about more than 1 million files.

Our Protege-based metadata system implementation offers support for ontology-based metadata query, ontology-based automatic metadata insertion, as well as ontology-based file access through both browser and command-line interfaces.

One typical use scenario is as follows: a meteorologist needs some monitoring data on Hurricane Katrina's path of movement. He also would like to see a visualization of the monitoring data.. In real life, raw monitoring and visualized data could belong to different projects, and different projects may have different vocabulary for describing data. The use of the ontologies in the Protege-based system can bridge the semantic differences that may exist among different science projects. We assume here that raw and visualized data belong to different projects. In this use scenario, on PetaShare, the meteorologist could simply open his web browser or the specific PetaShare commandline interface. Here we assume he types "Katrina" into the search box of his web browser and presses the search button. The straight arrow in Figure 1 shows the data flow of his query. PetaShare will then search its metadata store and return a list of files from both projects it thinks are related to Hurricane Katrina, as indicated in Figure 1 by dotted arrows. Then the meteorologist can simply click whatever file he wants to obtain, the metadata system will send out request to other parts of PetaShare to fetch the file back into the machine of the meteorologist.

The biggest advantage for the Protege-based system is the establishment of a unified view of scientific data across different science projects or even different science disciplines. A unified data view can enable scientists to access data from multiple projects from multiple disciplines, regardless of the differences in vocabulary. Such data view is critical in modern, increasingly cross-disciplinary science.

The shortcomings of the Protege-based metadata system are clearly illustrated in Figure 1. Basically, in exchange for the expressive power of the ontologies, we have to build another metadata system independent of the iCAT [3] metadata system used by iRODS. Doubtlessly, the extra set of metadata and everything related to its management add overhead to overall performance of PetaShare. Also almost the entire set of technologies we employed to implement the system is Java-based, which introduces more overhead to performance and more complications to achieve maximum scalability.

3.2 iRODS-based Metadata System Framework

Unlike ontology-based system, the iRODS-based Metadata System does not support a richly representative scheme, namely ontology, like Protege does. On the other hand, iRODS and its corresponding iCAT metadata system serve as the backbone of PetaShare. As a result, metadata system based on iRODS and iCAT is naturally integrated into PetaShare seamlessly. Also, unlike ontology technology which is Java-based and was originally designed for Semantic Web with little prior consideration for performance, iRODS and its corresponding metadata system iCAT were designed with the requirements of data-intensive computing in mind. Better performance can be achieved as a result.

As Figure 2 shows, the framework of the iRODS-based metadata system is far simpler. Only one extra layer of system is added to the existing iRODS-based PetaShare storage. The PetaShare clients have been developed to parse and remote-execute various iRODS commands. One such command is "imeta", which is used for inserting and accessing metadata stored in iCAT.

Command "imeta" can be used to insert metadata about iRODS files, collections, resources and users in the form of Attribute-Value-Unit triples (AVUs) Because iCAT also employs relational database as back end storage and the fact that iCAT deals with metadata far less expressive than ontology does, we expect it to be able to be at least as scalable as the Protege-based system. Our experiment indicates that iCAT can easily handle file metadata in the order of millions of files. In current implementation, command "imeta" can only insert metadata

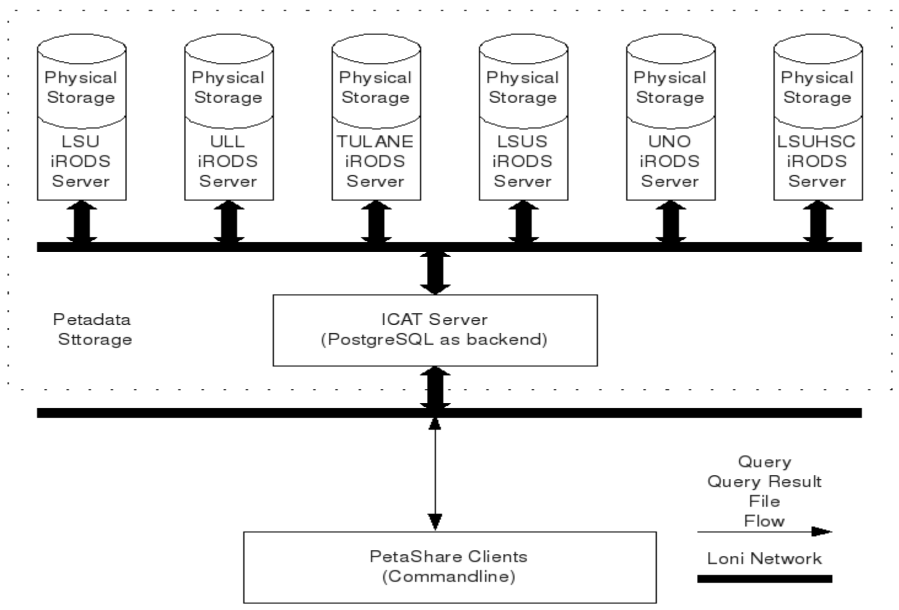

Fig. 2. PetaShare Metadata Framework for iRODS

one AVU at a time. To expand its functionalities, we implemented another version of command "imeta" that supports bulk-insertion function similar to the one provided by Metadata-insertion interface in the Protege-based system.

In the iRODS-based metadata system, a typical query operation would be users typing in what they want to query as parameters of command "imeta". "imeta" will do the query and return a list of files. Users then can use other iRODS commands supported by the PetaShare clients to access the files needed to be accessed.

There are several functionalities missing in command "imeta". For example, although Attribute-Value-Unit triples are easy to handle and understand, their expressive power is far from being able to adequately meet the requirement of an extendable semantic metadata system we envision for PetaShare. On the other hand, at the lowest level of representation, an ontology also consists of triples. We think it is possible to implement certain middleware so that upper level ontology-based metadata can be stored in lower level iCAT store. We are also interested in adding over the long term, the kind of querying and reasoning capabilities supported by the Protege-based system.

4 Issues and Observations in Performance Evaluation Experiments

In this section, we will present our investigation into some of the performance and scalability issues we encountered during the design and implementation process.

Both systems have been deployed on PetaShare, with controlled access granted to four PetaShare science projects. The iRODS-based metadata system is used internally by the PetaShare system itself, and the Protege-based system is deployed as a web-based search engine that allows scientists to search the metadata store and access files stored in PetaShare.The scalability and performance of the Protege-based system is not entirely satisfactory, as a result, we conducted an series of experiments to

We first tested the bulk insertion program we developed for the Protege-based system and modified "imeta" command for the iRODS-based system as users often need to first "build up" their presence on PetaShare, which makes performance and scalability of the bulk insertion program a important issue.

In the case of the Protege-based system, the Java implementation of Protege and the complexity of the ontologies exact a heavy toll on performance of the Protege-based insertion. During our experiment, we attempted to test the limit of the scalability of the Protege-based system. The conflict of a Java-based system and the memory requirement for data intensive applications was laid bare: The Java Virtual Machine can only use at most 2 GB of memory in Unix like system, which is hardly enough for a ontology containing metadata for tens of thousands of files. We experimented with inserting metadata for 1 million files in the Protege-based system. The experiment ran 19 hours 12 minutes and 46 seconds. It succeeded in inserting metadata for 684632 files, then the process crashed when another Java-based program was launched. Another attempt ended with metadata for 734845 files inserted in 24 hours 43 minutes and 53 seconds. The process crashed again presumably because of memory hog. It has also been observed that as the size of metadata grew, the execution of the insertion programs became extremely slow and unresponsive. Further investigation is needed to determine the exact cause of the performance differences we witnessed and how big the ontology can be scaled to.

It is clear that as the size of the ontologies grows, the Protege-based system would encounter scalability problem. In the future implementation, we plan to physically distribute the ontology to expand its scalability.

Scalability test on the iRODS-based system, however, did not go as smoothly as we hoped either. Similarly to what we did on the Protege-based system, we attempted to insert metadata for 1 million files to the iRODS-based system. We observed that there was not discernible slow-down of the insertion speed as the size of metadata inserted grew, but error occurred after 2 hours 2 minutes and 35 seconds and metadata for 109428 files inserted. The error forced the insertion operation to stop and restart. Restarted insertion operation continued metadata inserting without a glitch until 1 hour 57 minutes and 50 seconds later and another approximately 100000 files inserted when another similar error occurred. Our observation was the iRODS-based system had no problem handling metadata for tens of thousands of files, but the insertion needs to be done in batches with approximately 100000 files as one batch. Precisely what caused the limitation on the size of the metadata insertion operation that can finish in one single run is not clear yet. We will tryto answer this question with more experiments in our follow-up work.

Beside scalability, we also tested performance on querying metadata store on the two systems. Our experiments showed that query time on the iRODS-based system is positively correlated to the size of result set. As the size of the result set grew, significantly more time was needed for the query to finish.

In the case of the Protege-based system, however, performance varied little as the query result size changed. Another observations of ours was that in the Protege-based system, queries that would return tens of thousands of files crashed the system. When no crash occurred, the performance was extremely bad, which indicated that the size of available memory that can be utilized by the Java Virtual Machine is also a contributing factor to query performance of the Protege-based system.

5 Conclusion

In this paper, we have introduced two preliminary experimental metadata management systems developed for the PetaShare project. For each system, we have presented the design and underlying technologies, and discussed potential benefits and pitfalls they might bring to the capabilities and performance of the overall metadata system. We have also discussed performance and scalability issues we encountered in the development process.

Acknowledgement This project is in part sponsored by the National Science Foundation under award numbers CNS-0846052 (CAREER), CNS-0619843 (PetaShare) and EPS-0701491 (CyberTools); by DOE under award number DE-FG02-04ER46136 (UCoMS); and 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.

References

1. Protege, `http://protege.stanford.edu/`
2. iRODS, `https://www.irods.org/`
3. iCAT, `https://www.irods.org/index.php/iCAT`
4. Kosar, T., Livny, M.: Stork: Making Data Placement a First Class Citizen in the Grid. In: ICDCS '04: Proceedings of the 24th International Conference on Distributed Computing, pp. 342-349. IEEE Computer Society, Washington, DC, USA(2004)
5. Balman M., Suslu, I.: Distributed data management with PetaShare. In: MG '08: Proceedings of the 15th ACM Mardi Gras conference, pp. 1-1. ACM, New York, NY, USA(2008)
6. SPARQL Query Language for RDF, `http://www.w3.org/TR/rdf-sparql-query/`
7. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. In: Scientific America (1993)