Artemis: Integrating Scientific Data on the Grid

Rattapoom Tuchinda, Snehal Thakkar, Yolanda Gil, and Ewa Deelman

USC Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina Del Rey, CA 90292
{pipet, thakkar, gil, deelman}@isi.edu

Abstract

Grid technologies provide a robust infrastructure for distributed computing, and are widely used in large-scale scientific applications that generate terabytes (soon petabytes) of data. These data are described with metadata attributes about the data properties and provenance, and are organized in a variety of metadata catalogs distributed over the grid. In order to find a collection of data that share certain properties, these metadata catalogs need to be identified and queried on an individual basis. This paper introduces Artemis, a system developed to integrate distributed metadata catalogs on the grid. Artemis exploits several AI techniques including a query mediator, a query planning and execution system, ontologies and semantic web tools to model metadata attributes, and an intelligent user interface that guides users through these ontologies to formulate queries. We describe our experiences using Artemis with large metadata catalogs from two projects in the physics domain.

Introduction

Scientific data analysis is quickly moving to realms that require efficient management of data repositories in the petabyte scale [National Virtual Observatory project (NVO)[1]; Laser Interferometer Gravitational Wave Observatory (LIGO)[2]; Earth System Grid (ESG)[3]; Southern California Earthquake center (SCEC)[4]]. Grid technologies [Foster and Kesselman 99; Foster et al 01] are used to manage distributed resources: compute, data, instruments etc across organizational boundaries. Middleware, such as the Globus Toolkit[5], enable the discovery of remote resources and access to them in a secure fashion. Grids support the distributed management of large data sets through secure and efficient data transfer, storage resource management, and data replica management. Grids also enable distributed, large-scale, computation and data intensive analysis. Grids are very dynamic: the availability of the resources may vary significantly over time.

Although grids make many large data repositories available, integrating them in a highly dynamic distributed environment is a challenge. Metadata catalogs address some of these problems by supporting queries to data based on metadata attributes of the files and managing collections of data as well as versioning and provenance information [Moore et al 01, Singh et al 03]. Mediator techniques can be used to integrate diverse data sources [Levy 2000] but have not been used in grid environments that are extremely dynamic. Data sources can appear, be temporarily unavailable or disappear, change contents or access mechanism, and migrate to different locations. These changes are not only at the implementation and installation level, but also at the conceptual level in terms of the metadata used to describe and index their contents. As scientists model scientific phenomena and analyze large data sets, they generate new data products that need to be described using possibly new metadata that needs to be incorporated in the repository. In addition to operating in this very dynamic environment, scientific applications need to support a high degree of heterogeneity in the data. This is because they are often multidisciplinary efforts that integrate results from complementary observations and analysis of the same underlying phenomenon. That phenomenon may be described in different ways in different disciplines or modeling environments.

This paper describes Artemis, a query formulation and planning system that enables users to easily query metadata catalogs on the grid. It includes a query mediator based on planning techniques that dynamically updates its domain model, and an ontology-based query formulation system that helps a user create and refine queries interactively based on the current contents of the repositories. Artemis is integrated with an existing Metadata Catalog Service (MCS), which is based on the widely used Globus toolkit for grid computing. We have evaluated Artemis with data from two different scientific domains.

Artemis illustrates the potential of Artificial Intelligence techniques for data integration on the grid, and complements prior work on Pegasus, a workflow generation and mapping system for grid computing [Blythe et al 03; Deelman et al 03 Gil et al 04]. As Pegasus continues to be used in a variety of scientific domains to

[1] http://www.us-vo.org
[2] http://ligo.caltech.edu
[3] http://www.earthsystemgrid.org
[4] http://www.scec.org/cme
[5] http://www.globus.org

map, manage and execute workflows with hundreds of thousands of jobs and terabytes of data products [Deelman et al 04], the need to provide grids with a system like Artemis becomes more immediate.

The paper begins by outlining the key problems to be addressed in dynamic heterogeneous query planning. After describing Artemis in detail, we discuss our experiences to date integrating data from real scientific applications and the future extensions that are needed to support data integration on the grid.

Motivation

Science today is conducted in a collaborative fashion. In order to enable scientific discovery, collaborations encompass numerous researchers sharing computation, data and network resources, and instruments. Additionally, scientists share their applications and data not only in the raw form, but processed in some fashion.

Because of the size of the data, terabytes today, and petabytes in the near future, data is often distributed and replicated in the environment. Additionally, metadata—descriptive data about particular data products—is often stored and queried separately from the data itself. The metadata may be distributed as well. For example, in the case of the LIGO scientific collaboration that encompasses scientists from the US LIGO as well as the GEO[6] efforts (a British-German gravitational-wave detector experiment) scientists share the data collected independently in the US and in Europe. Each of the experiments: LIGO and GEO publish their data and their metadata to the Grid independently of the other experiments. Additionally, even after publication, the LIGO and GEO metadata may evolve independently of each other.

In order to conduct cross-correlation studies between the LIGO and GEO data, scientists need to discover the necessary data products in distributed metadata catalogs. However, it may be difficult for a LIGO researcher to fully understand the GEO metadata. After identifying the desired data products through the metadata queries, researchers can locate and access the corresponding data.

The metadata being published by experiments is often not the definitive set of descriptive attributes. As the understanding of the data collected increases, the attributes may change or may be augmented to include the new information. For example, it is often the case that raw data is of no scientific value unless it is carefully calibrated. Calibration is not an easy process and is often iterative. Consequently, the metadata changes as the calibrations are updated.

The LIGO/GEO metadata integration problem is in some sense simple, because the metadata comes from the same discipline. More complex issues arise when the metadata crosses discipline boundaries. For example, a gravitational wave scientist may want to identify potential objects in the sky based on a hypothesis that these particular objects (such as pulsars for example) can be a potential source of gravitational waves. In order to identify these sources, the scientist may want to search astronomy databases. Astronomy data sets are published by different entities (representing a particular telescope or mission for example) and include diverse metadata. Thus, a gravitational-wave scientist needs to be able to locate, query, and interpret these astronomy-specific distributed metadata sources.

In summary, three key problems need to be addressed:

- **Dynamic updates of sources:** A given metadata catalog may change over time not only in content, but also in the definitions and structure of the metadata. Each time a scientist processes data or runs a simulation, new data is generated that may need to be described with attributes that did not exist before in the catalog. New metadata types are continually being incorporated.
- **Integrating multiple sources:** Answering a single user request may require a sequence of queries to various catalogs as well as integrating intermediate results.
- **Bridging user requirements and system metadata across disciplines:** Users that are expert in a certain scientific discipline may need to access data in other sciences but may be unfamiliar with the terms and structure of the metadata. Metadata for different scientific disciplines or groups are naturally defined or structured differently, and are continuously and asynchronously redefined or restructured as scientific advances take place. This accentuates the already challenging problem of mapping a user's terms to the terms that are known to the system.

The next section describes our approach to address these challenges integrating several Artificial Intelligence techniques that include query mediation, planning, ontologies, semantic web languages and tools, and intelligent user interfaces.

Artemis: An Interactive Query Mediator for Distributed Metadata Catalogs

We developed Artemis, an interactive query planning system that integrates metadata catalogs available on the grid. Figure 1 shows the architecture of the Artemis system. The goal of the Artemis system is to enable the user to easily query various objects stored in different data sources. Metadata catalogs, such as the Metadata Catalog Service (MCS) [Singh et al 03] store the metadata about different objects in different data sources on the grid. Artemis utilizes data integration techniques in conjunction with an interactive query builder to enable the user to easily create complex queries to obtain objects that match user criteria.

[6] http://www.geo600.uni-hannover.de/

In this section, first we briefly discuss the MCS. Next, we describe the process of dynamically generating various data models based on the information from the Metadata Catalog Services. Then, we show how the Prometheus mediator is used in the Artemis system. Finally, we explain an interactive query building component of the Artemis system termed MCS Wizard.

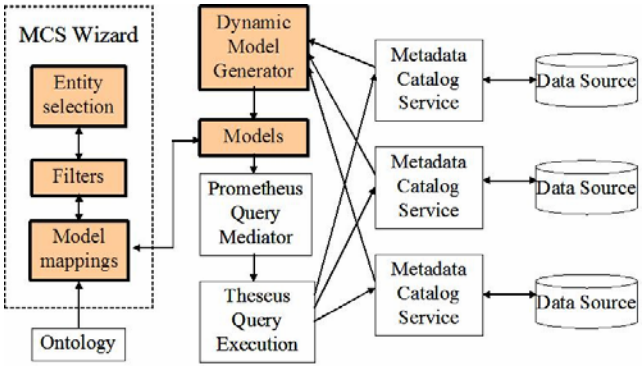

Figure 1. The Architecture of Artemis. The components in the lighter boxes are pre-existing modules, while the components in the darker boxes are developed specifically for Artemis. Note that Artemis is capable of integrating an arbitrary number of data sources

Metadata Catalog Services

Artemis utilizes the Metadata Catalog Service (MCS) [Singh et al 03] that has been developed as part of the Grid Physics Network (GriPhyN)[7] and NVO projects. The aim of these projects is to support large-scale scientific experiments. MCS is a standalone catalog that stores information about logical data items (e.g., files). It also allows users to aggregate the data items into collections and views. MCS provides system-defined as well as user-defined attributes for logical items, collections and views. One distinguishing characteristic of MCS is that users can dynamically define and add metadata attributes. MCS can also provide the names of the user-defined attributes. As a result, different MCS instances can be created with alternative contents. In our implementation, we assume that different MCS instances contain disjoint information. This assumption is not necessary for Artemis as it can handle replicated metadata, rather, it is dictated by the difficulty of maintaining the consistency of updates in a wide area network, which is typically part of a grid's networking infrastructure. For example, if two MCS instances contain the same information, when one MCS is updated the second one should be updated at the same time. Guaranteeing this level of consistency is very expensive in the wide area network and is beyond the scope of this work. Each data item in an MCS is described with a set of metadata expressed as attribute-value pairs. Some attributes are common metadata, such as creation date, name, author, etc (see for example, the Dublin Core[8]). Other attributes represent user-defined metadata and are typically specific to the domain and the contents of the files. Additionally, the set of attributes can be different for each item.

An MCS can be queried for files by specifying values for any attribute. In addition to items, the MCS can be queried for collections and views. Collections are usually created to group related items together. For example, if one experiment results in ten items, a collection can be created for those items. All items in the collection may have their own metadata. In addition, a collection can have its own metadata as well. Collections usually represent an organization of the data imposed by the data publisher and may carry authorization information. Views are similar to collections but rather than being defined by the data publisher, they are defined by users based on their own grouping of items. As such, views do not impose authorization constraints on the items they contain. Both collections and views may have a hierarchical structure.

The MCS architecture is very flexible because it easily supports the addition of new content with novel metadata that was not previously defined in the MCS. This flexibility presents a challenge to the problem of integrating multiple MCS sources for two reasons:

- **Rapidly changing metadata:** Since new items, views and collections can be added to an MCS at any time and each may contain a new type of attribute, it is not possible to define a model of the contents of an MCS a priori. This is one of the novel issues addressed in our work in integrating the mediator. Although mediators have mechanisms to handle dynamic data sources, MCS content varies more rapidly.
- **Lack of semantic specifications of metadata attributes:** the current MCS architecture does not include mechanisms to attach semantic information to the metadata attributes within an MCS (other than their type—string, integer, etc.,) or to indicate definitions shared across multiple MCS instances. Metadata such as the time interval for an observation with an instrument may be expressed in a variety of ways such as a start and end times, or as a start time and duration, or duration, etc. The same attribute may also be stored with different names. When several MCS instances are available, a user has to analyze their metadata attributes by hand and formulate different queries as appropriate for each of them.

[7] http://www.griphyn.org/index.php

[8] http://dublincore.org/documents/dces

We address the first issue by dynamically creating models of the information available in various Metadata Catalog Services. We address the second issue by using ontologies to describe, map, and organize metadata information. The next section describes the process of dynamically generating models of the metadata.

Dynamic Model Generation

The dynamic model generator creates a relational model of the entities available in the metadata catalogs. As we described earlier the Metadata Catalog Service utilizes an object-oriented model to allow the user to associate different metadata with each item in the catalog. While the object-oriented model provides user with a greater flexibility by allowing them to define arbitrary metadata attributes for any item, the data integration techniques target mainly relational models. Thus, in our work, the dynamic model generator first needs to create a relation domain model for the mediator by querying various metadata catalogs specified in the metadata layer.

Domain models are generated dynamically in four steps:

1. For each entity type supported by the MCS instances (e.g. items, collections, and views) Artemis queries all the MCS instances and finds all the attribute types.
2. For each entity type Artemis creates a domain predicate and sets its arguments to the list of attributes found in the previous step.
3. For each MCS and entity type Artemis creates a source predicate whose arguments are the list of attributes for the given entity type in that MCS. It also adds the *Itemname* attribute to the source arguments. The *Itemname* attribute is returned to the user, so the user can obtain the item from the Grid using the *Itemname*.

The original MCS did not support the queries described in step 1 above. As a result of the Artemis project, the MCS API was extended to accommodate this functionality.

Artemis creates domain rules for the mediator to relate the source predicates from step 3 to the domain predicates from step 2. If the MCS does not have some attributes in the attribute list Artemis uses a constant "empty" to denote the missing attribute.

For example, consider two simplified Metadata Catalog Services shown in Table 1.[9]

MCS	Object Type	Attributes
MCS 1	Item	keyword, starttime, endtime
MCS 2	Item	ci, about, sttime, etime

Table 1: Example of two MCS instances.

[9] For simplicity, we assume just one entity type. Both MCS and Artemis can support multiple entity types.

MCS 1 contains metadata for objects of type item. The metadata is described by the following attributes: keyword, starttime, endtime. Similarly MCS 2 also contains metadata for objects of type item. However, the metadata in MCS 2 is described by different attributes. The dynamic model generator creates the following domain model for the mediator.

```
items(keyword, starttime, endtime,ci,about, sttime, etime)
:-
   Mcs1items(keyword, starttime, endtime)^
   (ci = "empty")^
   (about = "empty")^
   (sttime = "empty")^
   (etime = "empty")

items(keyword, starttime,endtime,ci, about, sttime, etime)
:-
   Mcs2items(ci, about, sttime, etime)^
   (keyword = "empty")^
   (starttime = "empty")^
   (endtime = "empty")
```

The domain model states that objects of type item can be queried based on the following attributes: *keyword*, *starttime*, *endtime*, *ci*, *about*, *sttime*, and *etime*. The attribute list is a combination of attributes of both MCS instances. Furthermore, to obtain information about the objects of type item, Artemis needs to query both MCS instances and then it needs to construct the union of the resulting objects.

In general, MCS catalogs can be updated at any time. The frequency of updates can depend on the type of application and/or situation. For example, astronomy metadata may not vary too frequently once it is curated and published by the collaboration to the broad astronomy community. However, it may change often soon after the data is collected by a telescope, when it is made available to the researchers that are part of the initial collaboration.

Currently, Artemis regenerates the domain model for each query. However, as the number of metadata catalogs grows, it may become very expensive to regenerate the domain model for each query. In the future, Artemis may query individual MCS instances to find out whether there have been new attributes added and only then regenerate its model. Alternatively the MCS can be augmented with the capability to send out notifications to subscribed systems when changes to the attribute definitions are made.

Mapping Models to Ontology

The model mapping module shown in Figure 1 is used to attach semantics to the models generated by the dynamic model generator. This step is very important as it enables the user to compose the query using the terms in the ontology. This module generates the mappings between the terms defined in the metadata catalogs and the concepts represented in an ontology by interactively asking the user to map different terms from the model to the concepts in

the ontology. The model mapping module is currently limited in scope. Much research is still needed to fully support semantic interoperability and semantic data integration, but we believe that current technology may already have a great impact by improving the current state of the art of metadata services on the grid.

The model mapping module can include several alternative ontologies, each with its own scope and expressive power. Each ontology contains a set of associated mappings of the attributes in each MCS to the terms in that ontology. For each primitive type in an ontology, there exist predicates that specify the types of constraints that can be used in queries for that type. For example, a query over a numeric type can specify comparison or range constraints. If an attribute is mapped to a numeric type in the ontology then only such queries are semantically valid.

In our current implementation, we create the mappings and the predicates manually. An important area of future work is to create them dynamically. As users add new metadata attributes to an MCS, they should be required to provide additional information that can be used to create the necessary mappings. This will require a tighter integration of the model mapping module with the MCS architecture, and will require modifications to the API specification of the current MCS. We anticipate that this will be a complex process but one that could have significant impact for the scientist utilizing the grid system. We can illustrate the process using the example shown in Table 1. The user may elect to use an ontology that contains the concepts: *keyword*, *city*, and *time-range*. The user then needs to specify that the concept *keyword* maps to the keyword *attribute* in MCS 1 and the attribute *about* in MCS 2. Similarly, the city concept maps to the ci attribute in MCS 2. Finally, the starttime, endtime, sttime, and etime attributes map to the time-range concept.

We have used semantic web languages and tools to develop the model mapping component. The ontologies are expressed in OWL[10]. We use Jena's[11] RDQL to query the ontology about its contents, for example for the hierarchical structure of the ontology. The interactive query formulation system described next generates questions for users based on the contents of the ontologies, and helps them formulate queries for the Prometheus mediator based on the mappings between the models and the ontologies.

Data Integration using Prometheus

Artemis uses Prometheus, a mediator system that supports uniform access to heterogeneous data sources, including web services, web sources, and databases [Thakkar et al 03]. Prometheus uses planning techniques to expand a given query into a set of operations that specify how to access the appropriate data sources.

[10] http://www.w3.org/TR/2002/WD-owl-guide-20021104/
[11] http://jena.sourceforge.net/

The Prometheus mediator receives the user's query in a form of a query on the domain predicates. This query is received from the user interface. The mediator utilizes the given query and the domain model generated by the dynamic model generator to obtain a datalog program to answer the user query. As various metadata catalogs may be widely distributed in the wide area, some catalogs may take a long time to respond to the query. Therefore, it is important to query the metadata catalogs in parallel. The Prometheus mediator utilizes the Theseus execution engine [Barish and Knoblock 03] to efficiently execute the generated datalog program. The Theseus execution engine has a wide variety of operations to query databases, web sources, and web services. Theseus also contains a wide variety of relational operations, such as, selection, union, or projection. Furthermore, Theseus optimizes the execution of an integration plan by querying several data sources in parallel and streaming data between operations.

In the original MCS, the user had to query each individual MCS in turn and combine the results by hand. With Artemis, the user can seamlessly query across the various MCS catalogs and obtain combined results. The desired result set can be further refined through the Artemis interface. For example, if the first query returns too many objects, the user may want to add additional constraints on the attribute values. In the original MCS the user would have to again query all the catalogs with the new query. However, with Artemis, the constraint can be directly applied to the initial results. The burden of issuing the new queries is shifted from the user to the system.

Using this domain model, the mediator system can answer queries to obtain items, views, or collections that satisfy the query. However, the mediator requires that the queries are formulated in terms of its domain model and therefore in terms of the MCS metadata attributes that will be unfamiliar to users. The interactive query formulation system that we describe next addresses this problem.

Interactively Composing Queries using MCS Wizard

Artemis includes the MCS Wizard, an interactive query builder based on the idea in the Agent Wizard [Tuchinda and Knoblock 04]. The Agent Wizard allows users to build a complex agent by guiding users through a list of questions. The MCS Wizard follows a similar approach. To generate a set of simplified questions to pose to the user, the MCS Wizard takes the model generated by the dynamic model generator and exploits the previously created ontology.

Figure 2 shows the sequence of steps in the MCS Wizard. The MCS Wizard begins its interaction with the user by asking what ontology they would like to use and what type of object (items, collections, views) they would like to query. Based on the response, it helps the user formulate the query by navigating through the class hierarchy and the predicates that can be used to define the query expressions. The user is guided through the classes until the query is specified in terms of the classes that have

mappings to attributes that exist in the MCS instances.

The MCS Wizard then queries the mediator and presents the user with the answers. The MCS Wizard allows the user to refine the query if the results retrieved are not satisfactory. This is an important capability since queries are likely to return empty results because users will typically be unfamiliar with the contents of the MCS catalogs. An important area for future extension of the MCS Wizard is to generate guidance to the user in terms of how to reformulate or relax the query in order to find relevant contents by analyzing the intermediate results generated by the mediator.

To illustrate the user interaction with MCS wizard we use an example scenario where the user would like to find all objects present in the two MCS instances described in Table 1 that contain atmospheric data. In the first step, the user picks the ontology containing the concepts: *keyword*, *city*, and *time-range*. Next, the user picks the object type *item* and specifies the mapping between the attributes and the concepts in the ontology. Next, the MCS Wizard asks which concepts the user wants to query on. The user picks for example the concept *keyword* and specifies that the keyword must contain the term *atmospheric data*. The MCS Wizard generates the following queries for the Prometheus mediator:

Q1(keyword, starttime,endtime,ci, about, sttime, etime):-
items(keyword, starttime,endtime,ci, about, sttime, etime)^
(keyword contains 'atmospheric data')

Q1(keyword, starttime,endtime,ci, about, sttime, etime):-
items(keyword, starttime,endtime,ci, about, sttime, etime)
(about contains 'atmospheric data')

The query states that the mediator should find all items that have an attribute *keyword* containing atmospheric data or an attribute *about* atmospheric data. The Prometheus mediator queries both MCS instances and finds the relevant objects. The MCS Wizard shows the relevant objects to the user.

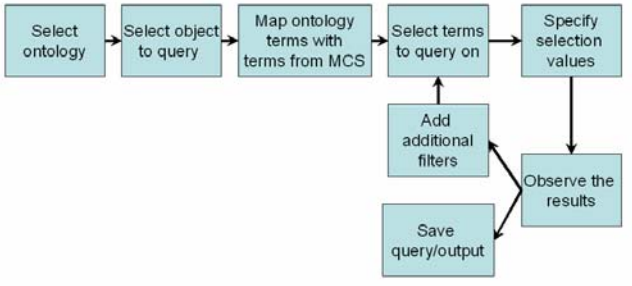

Figure 2. Building a query using MCS Wizard

Experiences to Date with Scientific Data Sources

We evaluated the effectiveness of our system by integrating 12 MCS services. These MCS catalogs contained information from three different systems: ESG, LIGO, and National Imagery and Mapping Association (NIMA)[12] Feature Vector data information covering different areas of the world. The NIMA Feature Vector data consists of 17,000 files that provide information about different feature vectors, such as road networks and railroad networks in different areas. We added this data to increase the complexity of the system. In total, the 12 MCS services contained information about 30,000 different files. On an average, each file had 50 attributes. Our goal was to be able to allow users to easily query for different files with different metadata using a simple interface.

We created our own ontology in OWL to define domain specific attributes as well as spatial attributes. We used an existing time ontology [Pan 2004] to map temporal attributes. This ontology mapped attribute names from different MCS instances with attribute names more familiar to users. Once the ontology and the mappings were created, Artemis users were able to build queries using terms from the ontology.

To test that Artemis would work even when metadata in MCS instances were updated or one of the services went down, we manually added metadata to one of the MCS services as Artemis was running. Then, we used the Artemis system to build a new query. During the dynamic model generation Artemis recognized the addition of the new MCS and built a domain model that included the new metadata.

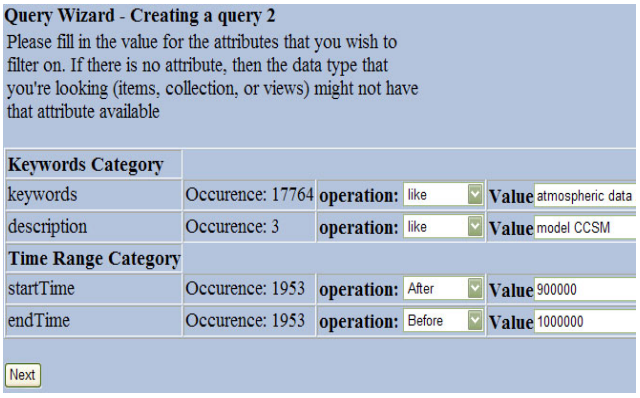

Figure 3. User interaction with MCS wizard

Figure 3 shows an example interaction aimed at finding a set of items that has "atmospheric data" in the *keywords* attribute, contains "model CCSM" in the *description*

[12] http://www.nima.mil

attribute, and has *starttime* and *endtime* of the data between 900000 and 1000000. First, the user picks *item* as the type of entity from the list of entity types (e.g., *items*, *views*, and *collections*). Next, the user selects terms from the ontology on which the user wants to put conditions. Next, the user is presented with a set of options to create filters on selected terms (e.g., *keywords* with filter operation "contains" on "atmospheric data", *description* with filter operation "contains" on "model CCSM") as shown in Figure 3. The time range query component is specified through the *starttime*'s operation "After" and the *endtime*'s operation "Before." The MCS wizard then formulates and sends the corresponding query to the mediator. The resulting list of items returned from the mediator is shown to the user. The user then has the option to either save the list of item names or to further refine the query.

If a user were to query the items based on three terms in the ontology, the user would need, on average, twelve simple mouse clicks in the MCS wizard interface to obtain the results. If the user were to perform these queries without using the Artemis system, the user would have to formulate separate query expressions to each of the MCS services. Furthermore, using Artemis, the user only needs to know the terms in the ontology instead of having to know attribute names from each of the 12 MCS instances. Overall, our experiences have shown that Artemis is able to cope with the three challenges outlined in the paper.

Related Work

The myGrid project [Wroe et al 03] is developing and exploiting semantic web technology to describe and integrate a wide range of services in a grid environment. Data sources are modeled as semantic web services, and are integrated through web service composition languages. The result is a workflow that may include not only steps to access to data sources as in Artemis, but also as simulation or other data processing steps. The workflows are generated by instantiating pre-defined workflow templates, while Artemis generates the entire set of access queries to the different data sources automatically and dynamically.

In [Ludäscher et al 03], the authors describe a mediator-based system that utilizes the semantics of the data exported by the data sources to integrate the data. A key assumption in that work is that the data sources export the semantics of the data. However, as we showed earlier, the MCS contains very weak semantic information. Thus, Artemis is not only responsible for integrating data from various MCS instances, but also assigning semantics to data from various MCS instances.

Recently, view integration researchers have developed various systems to integrate information from various data sources. A good survey of the view integration techniques is available in [Levy 2000]. Traditionally, view integration systems assume that a domain model is given by domain experts to the mediator system. In our case, the Artemis system automatically generates the domain model by reading information from various MCS instances.

Conclusions and Future Work

We described Artemis, a system that integrates distributed metadata catalogs in grid computing environments. Artemis exploits several AI techniques including a query mediator, a query planning and execution system, as well as ontologies and semantic web tools, and intelligent user interfaces. Artemis can automatically generate models of the data sources in order to dynamically incorporate new metadata attributes as they appear in the catalogs when new data items are added to the collection. Artemis isolates the users from the complexity of using distributed heterogeneous catalogs. It does so by providing interactive assistance to formulate valid, integrated queries using a single ontology that is automatically mapped to the particular metadata of each catalog.

We also plan to convert the existing system into a service. The Prometheus mediator and the Agent Wizard are already available as web services. The MCS is also implemented as either a web or grid service. Once Artemis is converted to the grid service, the user will be able to check status of their requests, by checking status of the service. This feature would be very useful when the number of MCS is very large as the user queries may take a long time to execute.

Another important improvement would be to add query reformulation capabilities to the mediator, so that it will be able to exploit the ontologies to reformulate queries.

An interesting test for Artemis will be the National Virtual Observatory (NVO) project, where a diversity of catalog services exist or are being created to store data collected from various telescopes. The catalog services will include not only MCS-based services but also other catalogs that will have a different structure. The architecture of Artemis and its mediator system are well suited to support heterogeneity of data sources. A single access point to such a large heterogeneous collection of data would enable rapid analysis of astronomy observations.

Scientific end-to-end workflows should integrate both data retrieval and data analysis steps. Artemis could provide the former, while the Pegasus system [Deelman et al 04] could provide the latter. The Artemis mediator already has mechanisms for robust execution of queries, while these capabilities are still being incorporated into Pegasus. Combining the capabilities of both systems to generate complete workflows will be an important area of future research.

Ideally, end users would want to query the collection using their own ontologies and definitions. The grid has much need for semantic web technology to support individual and community-based descriptions of data in a distributed setting. With systems like Artemis illustrating the added value and ultimate potential of Artificial

Intelligence techniques for data integration, the grid community is more likely to embrace this technology. This adoption would satisfy the requirements posed by complex and ambitious applications, scientific and otherwise.

Acknowledgements

We would like to thank Gurmeet Singh for the development and setup of MCS and Craig Knoblock for useful comments on the paper.

This material is based upon work supported in part by the National Science Foundation, under Award No. IIS-0324955, in part by the Air Force Office of Scientific Research under grant number F49620-01-1-0053, in part by a gift from the Microsoft Corporation, and in part by the National Science Foundation under grants ITR-0086044 (GriPhyN), AST0122449 (NVO) and EAR-0122464 (SCEC/ITR). The views and conclusions contained herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of any of the above organizations or any person connected with them.

References

Annis, J., Zhao, Y., Voeckler, J., Wilde, M., Kent, S. and Foster, I., Applying Chimera Virtual Data Concepts to Cluster Finding in the Sloan Sky Survey. in *Supercomputing*. 2002. Baltimore, MD.

Barish, G. and C.A. Knoblock. An Expressive and Efficient Language for Information Gathering on the Web. Proceedings of the Sixth International Conference on AI Planning and Scheduling (AIPS-2002) Workshop: Is There Life Beyond Operator Sequencing? - Exploring Real-World Planning. 2002. Toulouse, France.

Berners-Lee, T., James Hendler and Ora Lassila. "The Semantic Web" *Scientific American*, May 2001.

Blythe, J., Deelman, E., Gil, Y., and Kesselman, C. "Transparent Grid Computing: A Knowledge-Based Approach". Proceedings of the Fifteenth Annual Conference on Innovative Applications of Artificial Intelligence (IAAI), August 2003, Acapulco, Mexico.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G., Patil, S., Su, M., Vahi, K. Pegasus: Mapping Scientific Workflows onto the Grid. To appear in the Proceedings of across Grid EU Conference, 2004.

Deelman, E., Blythe, J., Gil, Y., Kesselman, C., Mehta, G.,Vahi, K., Blackburn, K., Lazzarini, A., Arbree, A., Cavanaugh, R., and Koranda, S. Mapping Abstract Complex Workflows onto Grid Environments, Journal of Grid Computing, Vol.1, no. 1, 2003, pp. 25-39.

Foster, I. and C. Kesselman, Eds. *The Grid: Blueprint for a New Computing Infrastructure.* 1999, Morgan Kaufmann.

Foster, I., C. Kesselman, and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications*, 2001. 15(3): p. 200-222.

Gil, Y., Deelman, E., Blythe, J., Kesselman, C., and Tangmunarunkit, H., Artificial Intelligence and Grids: Workflow Planning and Beyond. IEEE Intelligent Systems Special Issue on E-Science, Vol. 19, No. 1, Jan/Feb 2004.

Gil, Y., De Roure, D., and Hendler, J. (Eds) Guest Editor's Introduction to the IEEE Intelligent Systems Special Issue on E-Science, Vol. 19, No. 1, Jan/Feb 2004.

Levy, A., Logic-Based Techniques in Data Integration, in Logic Based Artificial Intelligence, J. Minker, Ed. 2000, Kluwer Publishers.

Ludäscher, B., Gupta, A., and Martone M. E. A Model-Based Mediator System for Scientific Data Management, B. In T. Critchlow and Z. Lacroix, editors, Bioinformatics: Managing Scientific Data. Morgan Kaufmann, 2003.

Moore R. W., *Boisvert, R., and Tang, P*, Data Management Systems for Scientific Applications. *"The Architecture of Scientific Software," pp. 273-284, Kluwer Academic Publishers*, 2001.

Pan, F. and J.R. Hobbs. Time in OWL-S. To Appear In Proceedings of the AAAI Spring Symposium, 2004.

Singh, G., S. Bharathi, A. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman. A Metadata Catalog Service for Data Intensive Applications. Proceedings of the Supercomputing Conference, November 2003.

Thakkar, S. and C.A. Knoblock. Efficient Execution of Recursive Integration Plans. Proceeding of 2003 IJCAI Workshop on Information Integration on the Web. 2003. Acapulco, Mexico.

Thakkar, S., J.-L. Ambite, and C.A. Knoblock. A view integration approach to dynamic composition of web services. In Proceedings of 2003 ICAPS Workshop on Planning for Web Services. 2003. Trento, Italy.

Tuchinda, R. and C.A. Knoblock. AgentWizard: Building Information Agents by Answering Questions. Proceedings of the 2004 International Conference on Intelligent User Interfaces. 2004. Portugal.

Wroe, C., R. Stevens, C. Goble, A. Roberts, and M. Greenwood. (2003). "A Suite of DAML+OIL ontologies to describe bioinformatics web services and data". To appear in *Journal of Cooperative Information Science.*