Personal Workspace for Large-Scale Data-Driven Computational Experiment

Yiming Sun, Scott Jensen, Sangmi Lee Pallickara, Beth Plale

Computer Science Department, Indiana University
Lindley Hall 215, 150 S. Woodlawn Ave., Bloomington, IN 47405, USA
yimsun@cs.indiana.edu
scjensen@cs.indiana.edu
leesangm@cs.indiana.edu
plale@cs.indiana.edu

Abstract [1] **— As the scale and complexity of data-driven computational science grows, so grows the burden on the scientists and students in managing the data products used and generated during experiments. Products must be moved and directories created. Search support in traditional file systems is arcane. While storage management tools can store rich metadata, these tools do not satisfy the nuances of the individual computational science researcher working alone or cooperatively. We have developed a personal workspace tool, myLEAD, that actively manages metadata and data products for users. Inspired by the Globus MCS metadata catalog and layered on top of the UK e-Science OGSA-DAI tool, myLEAD provides capture, storage and search tools to the computational scientist. In this paper we experimentally evaluate the performance of the myLEAD metadata catalog.**

I. INTRODUCTION

Computational science is changing as several factors converge to open opportunities for advancements. Environmental sensors are becoming more robust and taking on a commodity status. Computational testbeds such as Teragrid [5] provide vast quantities of computational resources in a distributed testbed to a broad community of users. Finally, workflow systems such as Taverna [18], Kepler [2], and GBPEL [21] are maturing to where complex data-flow processing can be carried out securely on behalf of a user.

As computational science becomes more automated the need diminishes for the user to continuously intervene and steer the computation. A natural outcome of this is the automation of data product management during workflow execution and after the fact. This development has several important implications on data management. First, programs must be able to search for data used in an experiment. Many systems allow user browsing through a web browser, however, programmatic access requires an underlying framework such as a computational grid [11]. Second, data used in experiments and generated by the workflows must be searchable on domain-specific metadata, and the metadata representation must be more sophisticated than very long, free form file names. Finally, metadata must be generated automatically – at least to the fullest extent possible.

The traditional file system is ill equipped to handle the needs of workflow-driven computational science. It shifts the burden of organizing and searching for data onto the user while providing limited tool support. Additionally, with data loosely scattered across local disks on workstations in the lab, on clusters, and on large-scale supercomputer resources, failsafe measures on data are generally haphazardly enforced, ensuring a certain ease with which data can be lost in an accident.

Metadata catalogs are a solution to metadata storage and sophisticated search. The notion of a separate DBMS hosted catalog for metadata is gaining a foothold in computational science through tools such as myGrid [14], MCS [25], and SRB [23], in distributed computing through Lustre [15] and GPFS [4], and even in enterprise networks through the Acopia ARX [1]. While the effectiveness of a metadata catalog is determined by the quality of the metadata, the efficiency is determined by the performance of the metadata catalog, which ultimately translates to how fast can users store, access and retrieve the information.

In [20] we introduced the notion of the myLEAD personal workspace. A user's personal workspace is a virtual repository of a user's data products. Its conceptual space is organized as *projects* that each contains *experiments*. An experiment can have *collections* and *logical files*. Each of these data objects can have rich *attributes* associated with it. The kinds of data objects stored to the personal workspace are input and output data files, a user's favorite executables, model input parameters, workflow templates, images, analysis results, notification (status) messages, and a user's imported data sets to name a few. Full deployment of the personal workspace is still in its early stages. Built as part of the NSF funded Linked Environments for Atmospheric Discovery (LEAD) project, the workspace is intended to serve upwards of several hundred users within the next 12 months.

While details are given in Section 3, in general terms the personal workspace is a metadata catalog service and separate back end storage manager. The catalog accepts data product descriptions on a wide range of data products including text, binary, images, workflow scripts, and input parameters. All data products are representable in the LEAD Metadata

[1] This work supported under NSF cooperative agreement ATM-0331480 and EIA-0202048.

Schema (LMS) [22]. A thin service layer provides atomic inserts into the catalog and back end, and performs other duties in cooperation with a workflow system [12]. The metadata catalog is inspired by the Globus MCS metadata catalog [25] and layered on top of the UK e-Science OGSA-DAI tool [3]. It is a distributed service with an instance located at each site in the LEAD grid [9]. A user's personal workspace is fully housed at one LEAD grid site, and backed up to a master site. While metadata objects all reside in the metadata catalog, some of the smaller data products reside there as well. Larger products are stored to disk and are replicated. Replica management is under the control of the Globus Distributed Replica Service (DRS) [7].

The contribution of this paper is an experimental evaluation of the performance of the myLEAD metadata catalog service. We microbenchmark the catalog through baseline insert and query operations. The scalability of the service is measured for up to 500 simultaneous users. Finally, myLEAD version 0.5 is compared to the Globus Toolkit MCS v3.1. Our study shows that complex attributes can be added on-the-fly without harming the performance of the kinds of queries we know will occur frequently in a personal workspace. We also quantify the benefit of bulk adds to the catalog.

The remainder of the paper is organized as follows. In Section 2 we describe the data model of the personal catalog and follow that in Section 3 with a discussion of the architecture of the service. Section 4 gives the results of the experimental evaluation. Section 5 contains related work. Conclusions and future work are given in Section 6.

II. Motivation

Our motivating application domain is mesoscale meteorology and the large-scale, workflow driven model executions that need to be carried out in real time in order to enable next generation weather forecasting and prediction [9]. Meteorology researchers execute forecast workflows, which can be as simple as a single model run, or can run on-demand, ingesting data products from observational data sources multiple times as it refines the model output. For instance, a researcher or student may perform mesoscale weather forecast on a selected geographic region using the Weather Research and Forecasting (WRF) Model. The forecast may be performed with progressively refined granularity of resolution according to the outcome of each run. For a region that is calm, the experiment may stop at a 256-km resolution, where as for region showing considerable activity (e.g. heavy storms), the experiment may continue with finer granularity up to 2-km resolution.

To reduce the chance of erroneous predictions and to obtain accurate results, several similar experiments are run with slightly varying initial conditions (e.g. ensemble runs), and the outcomes compared to make a final decision. An early estimate of usage of myLEAD is 500 users, where at any one moment 25 users are executing an ensemble run. Each workflow can use up to 1,200 functional nodes and can read or generate up to 10,000 data products per node [19].

III. Data Model and Service Architecture

The logical data model for a personal workspace is of "projects", "experiments", "collections", "logical files", and "attributes". A project can have one or more experiments, which themselves can contain any number of collections. Logical files can belong to one or more collections, experiments, or projects. Any of these data objects can have any number of attributes associated with them. Attributes can be a keyword, or simple or complex name-value pairs. Attributes can be added after a data object is created.

The relational schema we have developed for the data model is highly generalized. As given by the UML diagram in Fig. 1 for a slightly simplified version of the schema, the database maintains experiments, projects, and collections for all user spaces in a single table; logical files are in a separate table. The term "Attribute" is used in the general sense to mean a descriptive feature of a data object. We capitalize it hereafter to distinguish it from the database tables by the same name. An Attribute is implemented in the database schema as attribute and element tables. The former defines the name and structure *(i.e., data type)* of a possibly complex Attribute. The latter can be thought of as a <name, value> pair belonging to one or more entries in the attribute table.

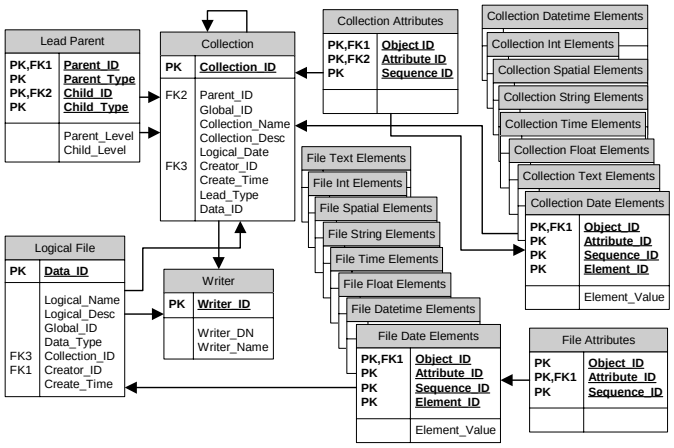

Fig. 1 Simplified relational schema of myLEAD database. The organizational details (experiments, projects, etc.) and application attributes are coded in the data and not the schema. This gives the catalog science domain independence.

The attribute table defines attributes as containing one or more elements. An attribute can be added on-the-fly by adding a new row to the attribute table. (It is slightly more complicated because the attribute must be declared before an instance created.) This design decision reflects the balance we maintain between support for annotations after-the-fact and efficient querying. Additional details on the schema and database support can be found in [17].

A. Architecture

The myLEAD personal workspace is a distributed tool as depicted in Fig. 2. At the lowest layer are the data resources. Metadata is managed by the myLEAD service and stored to a relational database. Much of the logic is implemented in database stored procedures. The data products themselves are

either co-located with the metadata in the database (such as workflow scripts), or passed to a replica manager, such as the Globus Toolkit Distributed Replica Service (DRS). We envision in the future providing support for Unidata THREDDS Data Server (TDS), which provides application specific features such as the ability to subset and aggregate files, and crack open a binary netCDF file.

The myLEAD agent provides client-side services to the user who is working interactively through the portal, and to the remaining LEAD system, the workflow execution engine in particular. The myLEAD agent responds to activities being carried out on the LEAD grid by listening on an event notification channel on which status events about workflow progress are published. The agent uses the notification messages to determine the current state of a particular workflow, and actively manages the user space by, for instance, creating a new collection when a major mode transition has taken place [21]. User interaction with the tool is through the LEAD Science Gateway [13]. We are building support for user interaction beyond the portal, however. In particular, users will be able to download archived experiments of collections to their laptop.

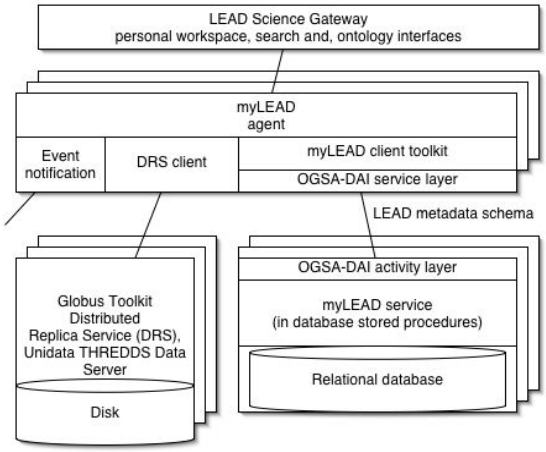

Fig. 2 Architecture of myLEAD Personal Workspace. The agent is a single service layer on top of the storage repository and metadata catalog. It intercepts workflow activity by listening on event notification channels.

IV. Experimental Evaluation

To test the performance of myLEAD, we designed a suite of test cases that span the different dimensions of the system. One subset of the test cases profiles common execution paths. The second subset focuses on the total response time as seen by an individual user. While these two subsets provide valuable information on myLEAD performance in a "vacuum" and allow us to identify critical performance bottlenecks, the third subset of cases simulates a real-world use scenario of hundreds of concurrent users. The fourth subset compares myLEAD and MCS. The next few subsections discuss the database catalog that we built up to carry out the tests, the test harness and environment, and finally, the experiments themselves.

A. Test Database

The test metadata catalog database is populated with sufficient data to simulate the realistic personal workspaces of 50 users. Of the 50 personal user spaces, 49 serve as background data for the microbenchmarks, then are exercised fully during the user scalability test. The target account is populated with a minimal set of data initially. Once all create tests cases have been run, the data set contained under this space is nearly identical to the background spaces. Each background user space is pre-loaded with 1 project, 1 experiment, 14 collections, 21,635 files, 3814 attributes and 3814 elements.

The target user space initially has 1 project, 1 experiment, 9 collections, 35 files, 20 attributes and 20 elements. As inserts are carried out, the target user account grows to the size of the others. The test database is estimated to be 170MB when fully populated. The logical file table has over 1 million entries.

B. Test Harness and Test Environment

The myLEAD server used in our performance evaluation is installed on a dual AMD 2.0GHz Opteron processor node with 16GB memory running Gentoo Linux, kernel version 2.6.9. The server uses the following software packages: Globus Toolkit version 3.2.1 WS-Core package, OGSA-DAI version 6 for OGSI, Jakarta Tomcat Server version 4.1.31, and MySQL version 5.0.3 beta.

The test client is installed and run on a different but identical node, and both nodes are a part of a 16-node cluster linked by Gigabit Ethernet. SUN JDK 1.5.0-06 for AMD64 is used for running both the clients and the server. The Globus MCS installation is v3.1, using OGSA-DAI 4, and Globus Toolkit v3.2.1. It runs on an AMD 2.0GHz Opteron processor with 16GB of memory running Gentoo Linux kernel 2.6.9.

This paper describes an evaluation of the myLEAD metadata catalog. As such, as captured in Fig. 3, the test client interacts directly with the myLEAD server through the client toolkit. The client toolkit, written in Java, provides a programming interface to the service. Inserts and queries are built through successive calls to the toolkit. The object when completed is converted into an XML document.

On the server side, the document is parsed and handed off to the responsible activity for processing. "Invoke database" as named in Fig. 3 is a slight misnomer since much of the logic of the server is implemented in database stored procedures. When the operation finishes successfully or fails, a status code is returned to the client in a response document. The client parses the response document and passes the result onto the user.

C. Microbenchmarks

To determine the base overheads, we measure insert and query operations for several cases:

- ***Insert attribute objects*** – iteratively and in bulk, with number of attributes ranging from 10 to 1,100.
- ***Insert file objects*** – iteratively and in bulk with number of files ranging from 1 to 6,000.

- ***Insert nested attributes*** – from an attribute having 2 nested levels to one having 5.
- ***Query attributes*** – nested, multiple, and geospatial attributes.

Insert attribute objects. In the first run, we add attribute descriptions to the metadata catalog, either one at a time, or in bulk for 1 to 1,100 attributes and measure response time as seen by the test client (framework depicted in Fig. 3). As can be seen in Fig. 4, in the case where attributes are added iteratively, response time is linear to the number of attributes added. This is expected as adding 100 attributes in 'one-at-a-time' mode is nothing more than a sequence of independent adds. Bulk adding, on the other hand, shows better than logarithmic scalability. The results clearly highlight the need to educate users to use bulk inserts.

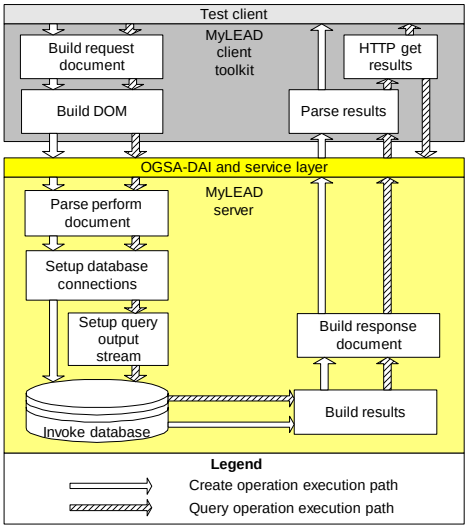

Fig. 3 Test framework showing functional decomposition through client toolkit and myLEAD server.

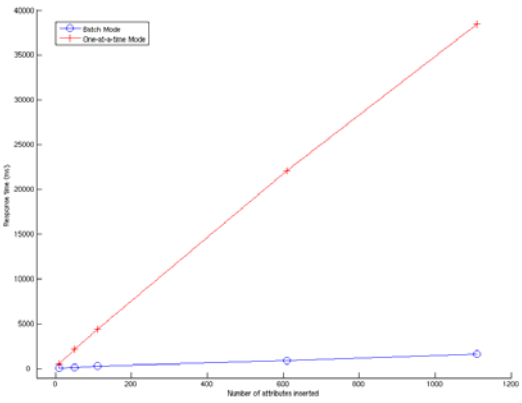

Fig. 4 Attribute object insertion. Response time for iterative and bulk inserts

Insert file objects. Inserting file objects displays a similar relationship as does inserting attributes as is shown in Fig. 5. We test file object inserts at 1, 500, 1000, 3000, 5000, and 6000. Just as with adding attributes, adding file objects in bulk is significantly more efficient than adding them one-at-a-time. Though we provide a more detailed comparison against the Globus Toolkit MCS catalog in Section 6.6, we show here for comparison sake, how myLEAD compares to MCS in file loads. Note that MCS does not support bulk loads.

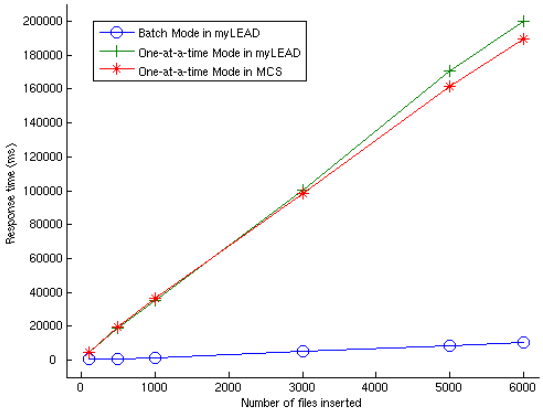

Fig. 5 File objects insertion. Response time for iterative and bulk inserts using myLEAD, and in comparison to MCS.

Insert nested attributes. Support for nested attributes is a unique design characteristic of myLEAD. We insert 500 nested attributes in bulk mode, testing with the nesting from 2 to 5 levels. The graph shown in Fig. 6 captures the average response time for a single request. The linear relationship indicates that the cost of adding another level of nesting is no more than the cost of adding another attribute – which is a couple milliseconds.

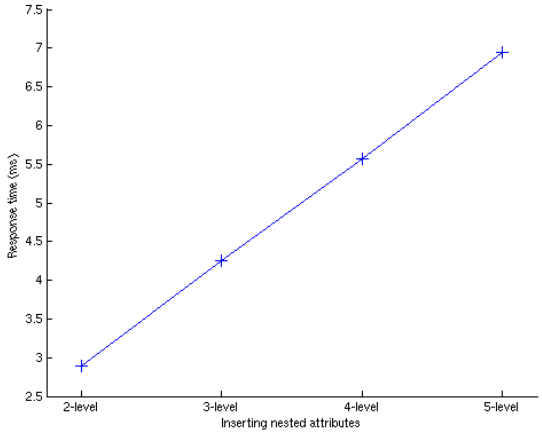

Fig. 6 Inserting attributes of increasing levels of nesting. Measures average response time for inserting single nested attribute.

Query of attributes. Getting a measure of base query time can be difficult because so much depends on the complexity of the query and the presence of indexes, etc. We select as our final microbenchmark, queries to retrieve various kinds of attributes, including nested attributes, several independent single-level attributes, and geospatial attributes.

The results for seven query cases are given in Fig. 7. The queries over nested attributes increase proportionally with each level of nesting as expected. Querying 5 independent

attributes takes slightly less time than does querying an attribute with 5 levels; this is also expected. The geospatial query uses MySQL's built in Geometry type, a subset of the "SQL with Geometry Types" environment proposed by Open GIS Consortium [6]. In particular, we store a geospatial bounding box as a Polygon, and support 'contains', 'within', and 'intersects' queries. In this test, a 'contains' query is used. The nearly 2 seconds performance we are seeing is problematic, as spatial queries will very likely be frequent because mesoscale meteorology research is research into regional weather phenomenon.

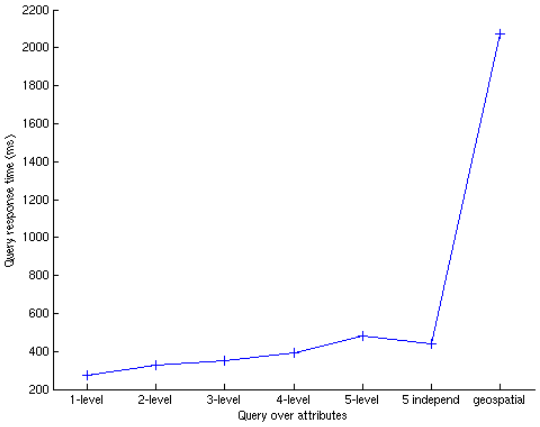

Fig. 7 Query over attributes

D. Scalability

To evaluate user scalability, we measure the response time for one user under increasing load imposed by additional users. The variable number of users is started up before the target user queries for a named file object. The background activity is added in increments of 50 users up to 500 users. The results, shown in Fig. 8, show good scalability, ranging from a user perceived performance that ranges between 300 and 380 ms for anywhere from no additional users to 500 additional users.

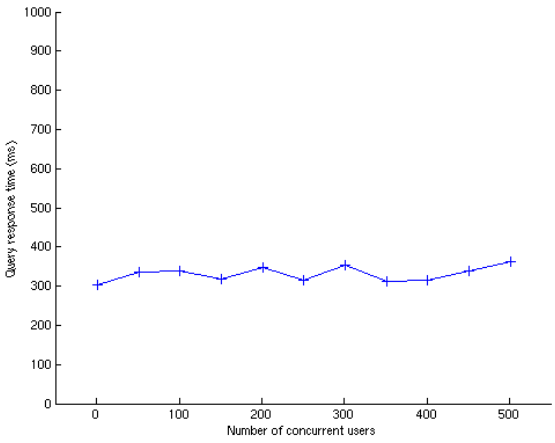

Fig. 8 Scalability in number of concurrent users

E. Function Profiles

Function profiling identifies any major functional bottlenecks and captures absolute performance at the functional unit. The functional units as described in Section 4.2 (Fig. 3) are listed along the X-axis in Fig. 9 and 10. We evaluated two insert cases and three query cases and discuss one of each in this paper. The insert case inserts 500 file objects in batch mode. The result, shown in Fig. 9, is a "box and whiskers" plot depicting the median, the upper and lower quartiles, and the smallest and greatest values in the distribution. As can be seen, the majority of time is spent in the stored procedures. This is not unexpected since the bulk of the logic of the service is implemented as stored procedures. The other noteworthy point is the overhead of the OGSA-DAI service stack, which includes SOAP processing and instantiation of a service instance to service the client. The presence of multiple outlier points in the OGSA-DAI overhead highlights variability in processing time partly explained by the slow start-up costs of creating a client-side "grid data service" to serve the request.

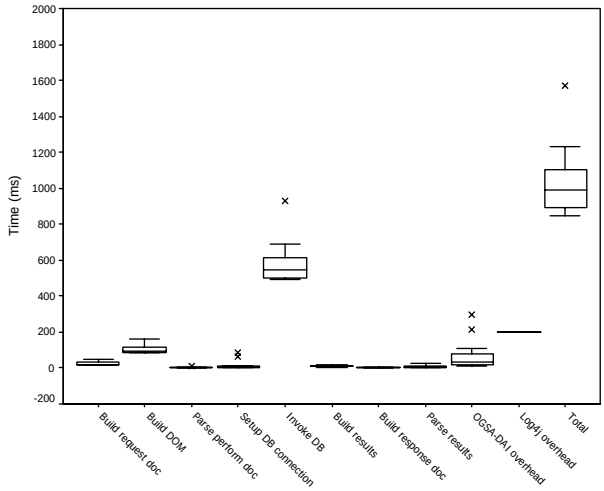

Fig. 9 Profile of batch insert of 500 files. Files loaded in bulk are stored to different collections.

We profile a query operation with a query over a collection given a collection name, and returning the subtree of 1000 files. Whereas the query is simple, the result set is rather large, at about 0.5MB. To optimize the returning of results, myLEAD employs an OGSA-DAI optimization that uses HTTP to pull the result set. OGSA-DAI creates a servlet that acts as a handle to the results, and returns to the client a URL that can then be used to retrieve the dataset. As can be seen in Fig. 10, the time to pull the results into the test client is considerably smaller than it would be if the SOAP service stack were used. (The latter approach was tested, but not shown here.)

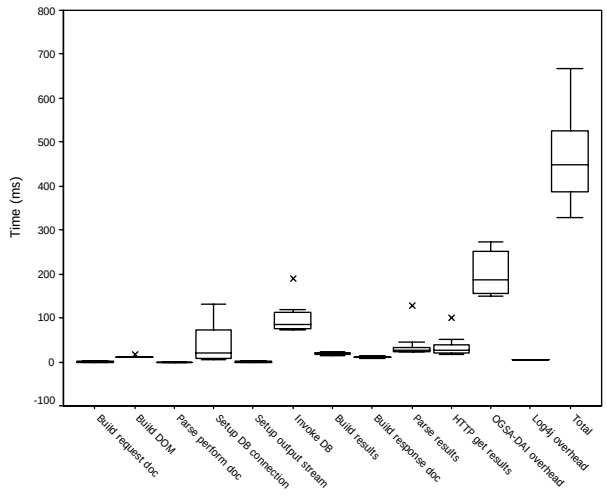

Fig. 10 Query collection given a name and return a collection-level subtree consisting of the metadata for the collection metadata plus metadata for 1000 file objects.

F. Comparison of MyLEAD and MCS

In the final experiment we compare aspects of myLEAD to Globus MCS. MCS served as early inspiration for myLEAD. Its designer's decision to decouple the metadata catalog from a back end storage manager is appropriate for the LEAD project where heterogeneity exists in local storage resource managers (i.e., OPeNDAP [8] is widely used in the meteorology community.) Additionally, its support for annotation of data product after-the-fact addresses an extremely important requirement in computational science experimentation in general – and mesoscale meteorology is no exception.

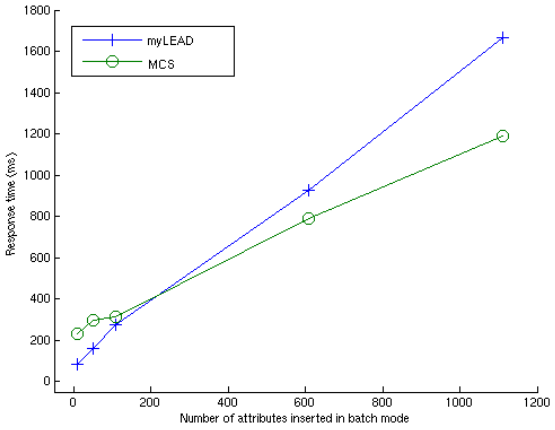

Fig. 11 Comparison of bulk adding of attributes in myLEAD and MCS

It was important that the optimizations and restructuring that we did to MCS to adapt it as a personal workspace catalog did not result in poorer performance than MCS for the most common actions. Fig. 5 and 11 indicate that we met the objective reasonably well. Fig. 5 shows that inserting file objects iteratively (one at a time) into the myLEAD catalog can be done in times comparable to MCS. Bulk load is one optimization built into myLEAD that shows significant performance gains over the alternate approach.

MyLEAD uses a more complex internal representation for attributes than does MCS. In myLEAD an "attribute" is hierarchical in nature and consists of at least two entries - an attribute and any number of elements. In MCS an "attribute" is a name-value pair. In addition, attributes in myLEAD must be defined before being used. This prevents users from mistyping an attribute name then not being able to find it later, and prevents a proliferation of names for the same attribute, again complicating the search process. As can be seen from the results shown in Fig. 11, when the number of attributes to be added in bulk is relatively small, fewer than 250, myLEAD out-performs MCS. However, when the number of attributes is more than 250, the overhead of the more complex attribute structure becomes a more dominant factor in overall response time. The reason for this is that when the number of attributes is small, myLEAD's approach of looking up the definitions and inserting two entries using numerical keys is faster than is inserting one entry using character string. Eventually the time spent looking up and inserting two entries becomes more than the time spent to inserting the same number of entries using string, and thus myLEAD response time degrades more quickly. We believe this additional overhead at attribute load time will not degrade overall performance because an attribute add is a one-time operation where as queries are more frequent. MyLEAD is optimized for queries, in particular, the containment query.

Containment Queries. The final performance comparison is on containment queries. A containment query searches for an entity in a tree given attributes about the entity's children. It is of the type:

> *"Find and return the experiments that **contain** a Collection having attributes α and β and a Logical File having attributes χ and δ."*

Since the personal workspace is presented to the user as a hierarchical, tree-oriented space, the containment query is expected to be a very frequently issued query. The test database contains 100 experiments. Each experiment contains 50 collections with each collection having 10 attributes and containing 100 files. Each file has 10 attributes. The selectivity on files is 10%, and on collections is 20%. The query returns 5 experiments (since MCS has no formal notion of an experiment, a parent collection is returned).

We measure two containment queries. The first query returns just the IDs of the experiments while the second returns the entire subtree for each experiment. The results, shown in Fig. 12, show myLEAD has better than an order of magnitude improvement over MCS when returning the global IDs of the experiments meeting the query criteria, and slightly less than an order of magnitude improvement when the full subtree of each experiment is returned. Note that the scale is a log scale.

The reason for the sizeable performance difference is best explained through an example. Suppose a user's workspace in myLEAD consists of 1 project, 1 experiment, 1 collection and

1 file, and at each level, there are associated attributes. To obtain the full project subtree, myLEAD just needs one query call to the myLEAD server. The equivalent workspace in MCS consists of 3 nested collections, and 1 file attached to the inner-most sub-collection. Attributes are associated to each level. To query the entire subtree, the user must issue 3 list calls to return the objects at each nested level, and 2 calls to obtain the attributes at each level for a total of 11 calls. Should there be multiple objects at each level, the number of calls would increase significantly, because it must issue 1 list call for each logical collection, and 2 attribute calls for each object.

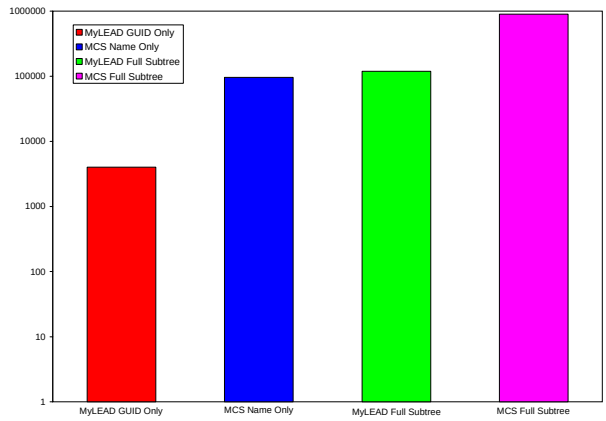

Fig. 12 Comparison for two containment queries. The two bars on the left show response time for myLEAD and MCS when experiment IDs are returned. The two bars on the right show response time when the entire subtree of each qualifying experiment is returned.

This result points to what we consider to be a key optimization in myLEAD – that is, support for efficient "contained" and "containment" queries. We believe these queries will form a large part of the queries that myLEAD will service, so their careful optimization is critically important to myLEAD overall performance.

V. Related Work

The structure of the myLEAD metadata catalog was motivated in part by the Globus Metadata Catalog Service (MCS) [25]. As in MCS, collections can be nested within collections to any depth, but myLEAD also allows for a hierarchy of collection types. The use of a hierarchy of collections or objects to describe scientific experiments is not unique to MCS and myLEAD; the metadata catalog used in the SciPort experiment management system [28] uses a six-level hierarchy consisting of: consortium, program, project, experiment, study, and transformation. However, unlike SciPort where the levels are system defined, early experiences in myLEAD indicated the need to allow a community to define its own hierarchy. SciPort also differs from myLEAD in that searching for data in SciPort is done mainly through browsing, and although the LEAD science gateway has support for browsing one's workspace, context queries can also be used to find files, collections, or experiments based on a rich set of complex metadata attributes.

Two additional characteristics that myLEAD shares with MCS are its use of the OGSA-DAI grid database service [3] and the ability to add metadata attributes without changing the database schema. OGSA-DAI provides general database access and delivery mechanisms as a grid service, and both MCS and myLEAD extend these general services with metadata catalog-specific activities. OGSA-DAI does not provide the tools for metadata generation or cataloging, but provides a context in which they could be deployed [3].

NASA's RHESSI Experimental Data Center (HEDC) [27] is a scientific data repository for high-energy solar observation data and related metadata. Similar to the LEAD project, HEDC deals largely with binary data that is stored in files and the related metadata is cataloged in a relational database. In addition, as in myLEAD, a scientist's data and derived products are private until explicitly shared. However, HEDC differs from myLEAD in that instead of a flexible hierarchy it has 3 system-defined levels – catalogs, "high level events" (HLEs), and analyses (ANAs). In addition, although metadata is separated into generic and domain specific, the attribute structure is targeted towards the HEDC domain.

In the biomedical domain, the Mobius [16] grid-based data management project took a different approach from myLEAD in that instead of mapping a community schema to metadata attributes and elements in a relational database, it supports a service that catalogs XML schemas for different data sources. In the biomedical field much of the data and derived results are stored in established, community databases so collecting heterogeneous schemas is a reasonable design choice. This differs from meteorological research where most of the data is stored as binary files.

MyGrid manages data and metadata of scientific studies in the biological sciences. The myGrid repository component catalogs metadata and annotations regarding experiments [26]. The myGrid information model [24] is a hierarchy starting with "study" and including levels named programme, investigation, and experiment instance. As with SciPort and HEDC, the hierarchy is system defined. As with Mobius, a key feature of myGrid is the ability to handle input based on different data schemas – a capability that is more relevant to the biological sciences. Although attributes in myGrid resemble the simple name/value pairs of MCS, the myGrid system provides the ability to include annotations based on a common vocabulary related through a domain-specific ontology. The LEAD project is also constructing a common vocabulary, related through an ontology to broader concepts that could be used in user queries.

VI. Conclusion and Future Work

In this paper, we present the myLEAD personal workspace manager, focusing on its performance and scalability. The results from the performance tests are satisfactory, while pointing the way towards additional optimizations. Current efforts underway include support for the domain standard XML schema for describing data products in LEAD. The

LEAD Metadata Schema (LMS) is a profile of the Federal Geographic Data Committee (FGDC) standard [10]. The next release of myLEAD will fully support products described in the schema. We are also working on publishing select data objects from the user's personal space into a LEAD-public registry. Once deployed, which we anticipate for late summer 2006, we will begin gathering usage characteristics to develop a realistic workload from which further optimizations will be motivated.

Although the user's personal workspace is developed as a service in a larger infrastructure developed for mesoscale meteorology, its unique features are solutions to some of the pervasive problems in computational science today, and its application will reach far beyond the single domain and find common ground in many other science domains as well.

REFERENCES

[1] Acopia Networks, "File virtualization with the Acopia ARX," Acopia White Paper, 2005.

[2] Altintas, I., C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock, "Kepler: an extensible system for design and execution of scientific workflows," *In Proceedings of the 16th International Conference on Scientific and Statistical Database Management (SSDBM)*, 2004.

[3] Antonioletti, M., M. Atkinson, R. Baxter, A. Borley, N. P. Chue Hong, B. Collins, N. Hardman, A. C. Hume, A. Knox, M. Jackson, A. Krause, S. Laws, J. Magowan, N. W. Paton, D. Pearson, T. Sugden, P. Watson, and M. Westhead, "The design and implementation of grid database services in OGSA-DAI," *Concurrency and Computation: Practice and Experience*, Vol. 17, No. 2-4, 2005, pp. 357 – 376.

[4] Barkes, J., M. R. Barrios, F. Cougard, P. G. Crumley, D. Martin, H. Reddy, and T. Thitayanum, "GPFS: a parallel file system," *IBM International Technical Support Organization*, SG24-5165-00, Apr 1998.

[5] Beckman, P. H., "Building the TeraGrid," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, Vol. 363, No. 1833, Aug 2005, pp. 1715 – 1728.

[6] Buehler, K., and L. McKee, "The OpenGIS guide – introduction to interoperable geoprocessing," Open Geodata Interoperability Specification (OGIS), Open GIS Consortium Inc., Technical Report. 1996.

[7] Chervenak, A., R. Schuler, C. Kesselman, S. Kornada, and B. Moe, "Wide area data replication for scientific collaborations," *In Proceedings of Grid Computing, 2005, The 6th IEEE/ACM International* Workshop, Nov 2005, pp. 1 – 8, 13- 14.

[8] Cornillon, P., J. Gallagher, and T. Sgouros, "OPeNDAP: accessing data in a distributed, heterogeneous environment," *Data Science Journal*, Vol. 2, Nov 2003.

[9] Droegemeier, K. Brewster, M. Xue, D. Weber, D. Gannon, B. Plale, D. Reed, L. Ramakrishnan, J. Alameda, R. Wilhelmson, T. Baltzer, B. Domenico, D. Murray, A. Wilson, R. Clark, S. Yalda, S. Graves, R. Ramachandran, J. Rushing, and E. Joseph, "Service-oriented environments for dynamically interacting with mesoscale weather," *Computing in Science and Engineering*, IEEE Computer Society Press and American Institute of Physics, Vol. 7, No. 6, 2005, pp. 12 – 29.

[10] *Federal Geographic Data Committee, Content Standard for Digital Geospatial Metadata Workbook Version 2.0*, Federal Geographic Data Committee. (May 1st, 2000).

[11] Foster, I., and C. Kesselman, *The Grid: blueprint for a new computing infrastructure*, Morgan Kaufman Publishers, Inc. San Francisco, California, 1999.

[12] Gannon, D., B. Plale, S. Marru, G. Kandaswamy, Y. Simmhan, and S. Shirasuna, "Chapter 10: Dynamic, adaptive workflows for mesoscale meteorology," *Workflows for eScience: Scientific Workflows for Grids*, Ed. I. Taylor, E. Deelman, D. Gannon, and M. Shields, Springer Verlag, 2006, pp. 97 – 114.

[13] D. Gannon, et, al. "Building grid portal applications from a web-service component architecture," *In Proceedings of the IEEE*, Vol. 93, No. 3, March 2005, pp. 551-563.

[14] Goble, C., C. Wroe, R. Stevens, and myGrid consortium (2003), "The myGrid project: services, architecture and demonstrator," *In Proceedings of the UK e-Science programme All Hands Meeting*, University of Manchester, Manchester, UK.

[15] Halbwachs, N., P. Caspi, P.Raymond, and D. Pilaud, "The synchronous data flow programming language Lustre," *In Proceedings of the IEEE*, Vol. 79, No. 9, 1991, pp. 1305 – 1320.

[16] Hastings, S., S. Langella, S. Oster, T. Kurc, T. Pan, U. Catalyurek, D. Janies, and J. Saltz, "Grid-based management of biomedical data using an XML-based distributed data management system," *In Proceedings of the 2005 ACM Symposium on Applied Computing (SAC)*, March 2005.

[17] Jensen, S., B. Plale, S. L. Pallickara and Y. Sun, "A hybrid XML-relational grid metadata catalog," To appear *Workshop on Web Services-based Grid Applications (WGSA'06) in association with International Conference on Parallel Processing (ICPP-06)*, Aug 2006.

[18] Oinn, T., M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. Pocock, A. Wipat, and P. Li, "Taverna: a tool for the composition and enactment of bioinformatics workflows," *Bioinformatics*, Vol. 20, No. 172004, Nov 2004, pp. 3045-3054.

[19] Plale, B., "Usage study for data storage repository in LEAD," *LEAD Technical Report, LEAD TR001*, October 2005.

[20] Plale B., D. Gannon, J. Alameda, B. Wilhelmson, S. Hampton, A. Rossi, and K. Droegemeier, "Active management of scientific data," *IEEE Internet Computing special issue on Internet Access to Scientific Data*, Vol. 9, No. 1, Jan/Feb 2005, pp. 27 – 34.

[21] Plale, B., D. Gannon, Y. Huang, G. Kandaswamy, S. L. Pallickara, and A. Slominski, "Cooperating services for managing data driven computational experimentation," *IEEE Computing in Science and Engineering (CiSE)*, Vol. 7, No. 5, Sep/Oct 2005.

[22] Plale, B., R. Ramachandran, and S. Tanner, "Data management support for adaptive analysis and prediction of the atmosphere in LEAD," *22nd Conference on Interactive Information Processing Systems for Meteorology, Oceanography, and Hydrology (IIPS)*, Jan 2006.

[23] Rajasekar, A., M. Wan, and R. Moore, "MySRB & SRB - components of a data grid," *The 11th International Symposium on High Performance Distributed Computing (HPDC-11)* Edinburgh, Scotland, Jul 24 – 26, 2002.

[24] Sharman, N., N.Alpdemir, J. Ferris, M. Greenwood, P. Li, and C. Wroe, "The myGrid information model," *In Proceedings of the Third UK e-Science AHM*, Nottingham, UK, Aug 2004.

[25] 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," *ACM Supercomputing Conference*, Phoenix, AZ, Nov 2003.

[26] Stevens, R.D., A.J. Robinson, and C.A. Goble, "MyGrid: personalised bioinformatics on the information grid," *Proc. 11th Int'l Conf. Intelligent Systems for Molecular Biology*, International Society for Computational Biology, 2003.

[27] Stolte, E., C. von Praun, G. Alonso, and T. Gross, "Scientific data repositories – designing for a moving target," *In Proceedings of SIGMOD*, San Diego, Jun 2003.

[28] Wang, F., P. Liu, J. Pearson, F. Azar, and G. Madlmayr, "Experiment management with metadata-based integration for collaborative scientific research," *In Proceeding of the 22nd ICDE*, Atlanta, Georgia, Apr 2006.