A Unified Relational Approach to Grid Information Services

Extended Abstract

Peter A. Dinda
Department of Computer Science
Northwestern University

Beth Plale
Department of Computer Science
Indiana University

Abstract

We propose an approach to Grid information services (GIS) that is based on the relational data model instead of the hierarchical data model that is currently prevalent, especially in U.S.-based Grid efforts. Furthermore, the approach integrates static and dynamic information into one information framework. Our core observation is that many highly desirable queries involve constraints on compositions of information. Such queries map naturally into joins within the relational model. In addition, we expect that updates will be a common feature in GIS systems, and relational databases are more likely to achieve high update rates. Herein we lay out an argument for the relational model, describe our plans and the research questions we see, and show early performance results.

1 Introduction

As the scale and diversity of the resources, applications, and users involved in Grid computing [14, 17, 12] continues to explode, the amount of information needed to keep track of them grows commensurately and becomes increasingly dynamic. Simultaneously, applications need to pose and answer increasingly powerful queries over this information in order to exploit Grid resources well and satisfy users. We are exploring a new unified relational approach to managing, updating, and querying Grid information.

Our approach is unified in that we treat the whole spectrum of Grid information, from static to dynamic, in the same way from the perspective of users and resource managers. Our approach is relational in that we apply the relational data model to Grid information, and that we leverage existing relational database to achieve our goals. We believe that this approach holds the key to meeting the following fundamental requirements for a Grid information service:

- Efficient and scalable support for a large and a rapidly growing number of data objects,
- Elegant representation of complex evolving relationships between data objects,
- High performance support for frequent and complex updates to data objects, their attributes, and their relationships, including for objects that may belong to different administrative domains,
- High performance support for complex and evolving queries on data objects and their relationships, in particular for compositional queries, and
- High performance, integrated support for data streams.

The purpose of this paper is to lay out an argument in favor of the relational model, describe our plans and the research questions we see, and show early performance results. Although our focus is on Grid information, we believe the work in this area will generalize beyond the Grid as other directory services face similar trends.

2 Grid information

Distributed computing environments such as computational grids are necessarily awash in information. Applications must be able to locate data sources and suitable computational nodes on which to run. Schedulers require information about utilization and load in order to make their decisions. The owners of some resources need supply and demand information in order to set prices. Grid information occupies a spectrum: some information is relatively stable (host configurations, for example), while other information is quite dynamic (host load, for example). In the following discussion, we refer to an item of Grid information as a data object, or object. We intentionally use this general term to avoid too rapidly binding to a specific model of information representation.

A Grid Information Service (GIS) is a database (in the generic sense of the word) that consists of a set of objects, relationships between objects, and systems needed
to query and update the objects and relationships. An object describes an entity in the Grid. It has a unique identifier, possibly a timestamp, and a set of attributes. The relationships between objects vary depending on the data model used to structure the objects. A hierarchical (tree) or object-oriented data model makes extensive use of the parent-child (“is-a”) relationship. For example, an organization and a person working for the organization are related via a parent-child relationship. The flat-table organization of the relational database model lends itself to stronger reliance on the peer-to-peer (“part-of”) relationship between tables. For example, two computers and the network path that connects them are related via a peer-to-peer relationship.

An object exists in the GIS on behalf of an entity if the entity meets the following four criteria:

1. Usefulness: the entity can be described succinctly in a manner that captures key attributes.
2. Uniqueness: the entity is distinguishable from other entities of the same type.
3. Persistence: the entity is long lasting. The attributes of an entity may not transient, however.
4. Generality: the entity has value to multiple applications or users.

Many Grid entities meet these criteria and a number of specific entities already either exist or have been proposed. These are listed in Figure 1.

These criteria define inclusion as well as exclusion. Any describable, unique entity that has lasting value to the broader computational grid community should be represented in the GIS irrespective of characteristics like frequent update rate that might hinder performance of some implementations of a GIS.

3 The needs of Grid applications

Parallel and distributed applications and the software systems that map them to the Grid make the most demands of the GIS. The following presents several use cases that are intended to highlight common areas in which a relational approach to GIS can be highly effective.

3.1 Traditional parallel applications

Consider even a simple, but commonplace data decomposition problem in a data-parallel SPMD application—how to partition the rows of a matrix into blocks, and then assign those blocks to processors so that computation done on the matrix according to the owner-computes rule will achieve high levels of speedup. This partitioning and mapping problem is simplified on dedicated parallel machines and dedicated clusters because the programmer has considerable implicit information about the static and dynamic properties of the environment.

On a computational grid, however, the static properties of the compute and communications resources are not implicit, and their dynamic properties usually necessitate run-time adaptation. Both to initially map itself onto the grid, and then to adapt its behavior as it runs, the application needs detailed information about these resources. Furthermore, it is generally not interested in individual resources per se, but rather in compositions of them. For example, if it has been coded to run on four processors, then, at startup, it will want to ask questions like “find me a set of four unique hosts which in total have between 0.5 and 1 GB of memory and which are connected by network paths that can provide at least 2 MB/s of bandwidth with no more than 100 milliseconds of latency.” As it runs, it will want to ask things such as “tell me when the load on any one of these four hosts is at least 25% different from the average across all the machines” so that it can trigger a load balancing step when this occurs.

There are three things to notice from this example. First, the application’s queries compose information about multiple resources (data objects in the GIS) in unique, application-specific ways. Second, the data objects on which the queries depend are dynamic to different degrees. Host memory changes more slowly than upgrades to network paths, while host load fluctuates far more quickly than either. The implication is that some data objects in the GIS will be updated quite frequently. Finally, the application needs to query information streams (e.g., for host load values) in addition to information in a database. As we shall see later, our approach addresses these three points.

3.2 Traditional distributed applications

In the same manner as a task-parallel parallel application, a workflow-style distributed application needs to create a virtual datapath for requests on top of the grid’s resources. This induces a mapping process very similar to what we discussed above. The application will ask such things as “find me four processors such that if I map my datapath onto them in this manner then I can achieve a throughput of n requests per second while still keeping the latency below 10 seconds.” Similar questions arise in overlay networks.
Grid entity | Description
---|---
organizations | accountable bodies and owners of resources
people | resource administrators, resource providers, GIS administrators
physical resources | compute resources, network interfaces
services | job manager, load leveler, other GIS’s
communication resources | link capacity, switch capacity, error rate, drop rate
software benchmarks | BLAS, LAPACK, etc.
event producers | generators of event streams
event channels | propagators of event streams
event dictionaries | databases of commonly used event types
instruments | radar systems, telescopes, etc.
network paths | available bandwidth and expected latency
network topologies | hosts, switches, routers
wireless devices | wireless hosts, wavepoints, cells, etc.
virtual organizations | groups of collaborators

Figure 1. Sampling of Grid entities requiring representation in a grid information service.

3.3 Interactive applications

Non-traditional applications, especially those that have interactivity demands that can be expressed as real-time constraints, are emerging to take advantage of the explosion of resources provided by grid computing. Examples of these include interactive scientific visualization, distributed laboratories [31], and computer-aided design. These applications are composed of interacting programs, actuators, sensors, remote data sources, and distributed users and typically have high I/O or computation demands necessitating the inclusion of at least one high-end computational resource such as an SMP-node cluster.

One such application is CMU’s Dv [2, 26], a framework for constructing interactive scientific visualizations based on vtk [35] for wide-area environments. The logical view of a Dv program is as a flowgraph. The flowgraph can be statically mapped onto the available compute nodes, or the mapping can be determined for each individual visualization frame, or it can even change as a frame is processed. The mapping/scheduling process demands information about the static and dynamic properties of the distributed environment and compositions of that information in a way similar to the workflow applications described above.

3.4 Other Grid software systems

Applications also rely on GIS systems indirectly, making use of other Grid software that uses GIS. We give two important examples here: systems that provide a view on highly dynamic Grid information that is not within the purview of GIS, and schedulers that map applications to Grid resources, both within individual sites and across them.

Some important grid information does not meet the inclusion criteria established in Section 2, either because it is too ephemeral or is of interest only to a small set of applications at any one time. Furthermore, some information is exceedingly fine grain and thus very difficult to incorporate into any GIS system. A number of systems have been proposed or developed to provide this information, including NWS [41], RPS [7], and Remos [27]. Currently, the Grid Performance Working Group [19] is attempting to create standards so that such systems can interoperate.

While the data generated by these systems may not be in the domain of the GIS, the often complex structure and location(s) of their instances often is. For example, RPS is a system that provides time-series predictions of fine-grain measurement streams. While RPS is conceptually quite simple, an RPS prediction system is a graph of components, each of which may run on a different host and may communicate with different mechanisms. This flexibility is important because the costs of measurement and prediction vary widely with the resource being monitored. With this flexibility comes the need to manage these constellations of components and their relationships.

Grid schedulers are perhaps the most common interface that applications have and will have to the Grid. To do their work of mapping applications to Grid resources, Grid schedulers generate frequent and complex queries over diverse resources. At least one Grid scheduling researcher has taken it upon herself to develop a system, GridSearcher, that can answer a small subset of such queries by basically providing extra database functionality over an existing GIS system that provides no such functionality [34].
4 Limitations of current directory services

Current GIS systems [11] and proposals [18] are layered on top of general purpose directory services systems. Though directory services [23, 22, 3, 25, 1, 39] have broad use and are unquestionably successful, there are unique characteristics about Grid data that warrant a reevaluation of the solution space.

Traditional directory services manage descriptive information about relatively static objects, ranging from hostname/IP-address mappings to the office locations and telephone numbers of company employees. The data in directory services is frequently accessed but infrequently updated. The storage model is typically hierarchical, that is, entities are described by value-attribute pairs and related to their “parent” objects such that a path exists from the root object to each leaf object. A data object in a hierarchical model is a relative path, beginning with possibly a non-root node, and a terminating node. In a relational model, it is a row of a table.

We have examined several existing grid information service databases in an effort to understand their use in practice. One database we studied in depth used a schema (which was for Globus 1.1) defining sixteen different object types (e.g., globus daemon, globus service, globus service job manager). The database contained 105 object instances. Interestingly, the instances are spread amongst only five object types. In fact, 42% of the objects were job manager queues. We will expand our results in the full paper, but we postulate that the underutilization of the service is in part due to the limited expressibility of the hierarchical model.

As Grid resources and applications evolve, they are placing increasing demands on directory services. These demands take the following five forms.

1. Rapidly growing numbers of data objects.
2. Increasingly complex relationships among data objects.
3. Increasingly frequent updates to objects and their relationships (i.e., more “dynamic objects”).
4. Rapidly escalating demand for sophisticated queries, especially over related objects.
5. Growing interest in querying data streams.

Given the rapid growth of interest in Grid computing, it is clear that the number of data objects is exploding. Furthermore, the success of systems like Globus [13] and other systems [6, 36, 20] suggest that the Grid is poised for even more rapid expansion, resulting in even greater growth in number of objects. Economics are also beginning to come into play. The efficiency of a computational economy, regardless of whether follows a traditional model [40, 4] or the currently-in-vogue “gift economy” model [33] is strongly dependent on the quality and quantity of information that the GIS system can offer. Traditional directory services are not designed to support such large datasets. Perhaps the largest of these systems, DNS, contains only about a hundred million host objects, each with one attribute (their IP address). By our criteria, a GIS should support queries over many more attributes of each host, as well as over the network paths that connect them.

As the number of objects stored in the GIS grows, the relationships between them also grow in complexity, and new relationships are found to have value and therefore must be represented. For example, it has been shown that link-level information about networks, which can be collected scalably and efficiently, can be composed into good estimates of path-level characteristics that themselves can not be collected scalably or efficiently [28]. This introduces a new, extremely important relationship that links objects in the GIS. Other such relationships may appear as research continues. Unfortunately, most directory services support only the hierarchical model, which means that the GIS designer must decide which relationships are important early and encode them as an explicit link in the data model. Modifying a schema after-the-fact can be broadly disruptive.

Updates to Grid data objects are becoming increasingly frequent both because there are simply more data objects (and relationships) and because more and more dynamic kinds of data are being introduced. The update frequency of data objects must be determined by the needs of applications—within the limits of what the directory service can support. We anticipate that frequently updated, or dynamic objects, will eventually form the majority of objects a GIS system handles. Current directory services, including LDAP v3 [22], are well known to perform poorly in the presence of frequent updates.

A related problem with using existing directory services as the basis of GIS systems is their limited and restrictive query language: the language is procedural, lacks sophisticated processing semantics, and requires that the user have explicit knowledge of the tree structure. A declarative query language such as SQL, on the other hand, states what is to be retrieved but leaves the how up to the query engine. Thus significant gains in query efficiency via query optimization can be realized by the latter that are denied to the former. Second, users of the GIS will increasingly require more sophisticated queries that extract data from over the entire data domain. Hierarchical query languages lack the ability to do such sophisticated processing on the data.
A key need is to be able to integrate data on a user specified condition, and perform an aggregate computation over the results. Placing the onus of processing complex query results (i.e., a join) on a user seems to be an unnecessary throwback to an earlier time.

Finally, existing directory services also do not provide support for data streams, a type of data that we expect will become increasingly common as the Grid evolves. For example, as networks improve, Grid schedulers will probably want to schedule smaller and smaller units of work. To decide where to place that work, they will make use of increasingly dynamic information, including such information as prediction streams for RPS-like systems. Why should the scheduler developer have to deal with an entirely different data model and query model for such stream data? We believe that data streams should be unified with other Grid information to the greatest extent possible when the meet the inclusion criteria we outlined previously.

5 Our approach

At the highest level of abstraction, we are investigating and building relational Grid information services that meet the application needs and avoid the limitations that we laid out earlier.

5.1 A hierarchy of types

Independent of the approach, hierarchical or relational, an important need in the GIS community is to develop a common taxonomy of data types for the objects that represent entities in the grid. We believe that this should be an explicit and extensible type hierarchy. This is clearly a part of the Grid Forum standards process and we are participating in the appropriate groups.

5.2 GIS schema

An early focus of our work has been on modeling the core objects and relationships of a GIS. Our current model is shown in Figure 2 as an entity-relationship (ER) diagram. The rectangles in the figure are entities and the diamonds are relationships. Lesser relationships whose primary purpose is to link related entities, are darkened. A partial list of attributes for key entities are shown grouped by a bar near the entity that they describe. “Host” describes a single machine, an x86 SMP for instance. Host attributes can be stable (e.g., physical memory) or dynamic (e.g., processor load). A many-to-many relationship exists between Host and Cluster, meaning a cluster has many hosts and a host could belong to one or more clusters. Host and Endpoint (a generalization of a socket) have a many-to-one relationship, that is, a host can support many endpoints but an endpoint belongs to a single host. A host supports multiple executables (“Module exec”) and an executable is derived from a single source ‘Module source’. “Connection”, in the lower left of the figure, is a relation between two endpoints. Specifically, it is the logical end-to-end connection between endpoints. Its attributes include traceroute response, packet loss rate, ping time, and estimate of bandwidth availability. The attribute types for the connection attributes is currently under discussion by the Global Grid Forum Damed Working Group [16] and will be incorporated here as it develops. Network path, on the other hand, is a physical layout, or network topology, connecting endpoints, switches, routers, and network links.

The ER diagram has its roots in an a database scheme developed for managing RPS components.\footnote{There is a straightforward relationship between an ER diagram and a relational database schema. Intuitively, entities and “greater” relationships become tables, and the lesser relationships are foreign keys between tables.}

Figure 3(a) shows an example query that finds all sources of load information for the host kanga. This is an example of a deterministic query in that it will return complete results regardless of the time required to process the query. Figure 3(b) shows a time-bounded non-deterministic query, which we shall discuss in a subsequent section.

Another current focus is to populate the database using real data from existing Globus MDS and other sources. Unfortunately, after obtaining and examining several data dumps from well recognized and relevant MDS data servers, we were disappointed at the size and simplicity of the data. Our largest MDS dataset thus far has fewer than 1000 machines in it. Because of this lack of data, which we view as a serious detriment to the whole GIS community, we are currently exploring synthetic models to create representative data. There already exist network topology graph generators [42, 24, 30, 8], the applicability and usefulness to our work of which we omit in the extended abstract. We hope to add tools that create appropriate host information at the edge nodes of such graphs.

5.3 Dynamic objects

Updating the GIS is the process of changing the attributes of one or more objects. We categorize the entities represented in the GIS according to the how
frequently their attributes change and according to the
timeliness properties needed of their attribute data. At
this point, we believe there are essentially two cate-
gories: stable grid entities, and dynamic grid entities.

Stable grid entities are major resources, such as com-
pute engines, organizations, etc. These entities are long-
lived, changes to their attributes are infrequent, and can
be propagated relatively slowly. For instance, additional
memory is added to a supercomputer quite infrequently
and news of it can acceptably take days to reach cus-
tomers. Updates to the objects of stable entities and
propagation of those updates in a distributed database
system can be thus be done at a lower priority than up-
dates to objects that change more frequently or require
more rapid dissemination of their changes.

Dynamic grid objects, on the other hand, have either
more frequent updates or require more rapid dissemina-
tion of their changes. As an extreme example, consider

Figure 2. Entity Relationship (ER) diagram for core GIS entities.

Figure 3. Sample queries: (a) a simple deterministic query that finds all sources of load infor-
mation on a particular host, and (b) a time-bounded non-deterministic query using a heuristic.
a network path: the end-to-end path between two machines clearly meets the criteria for inclusion in the GIS as it is of interest to a broad set of users and the choice of one path over another can have significant performance implications. Relevant path attributes might include current bandwidth and predicted bandwidth, both values could see updates with a frequency matching the entities’ ability to generate new values: on the order of milliseconds. Furthermore, a program is not interested in stale updates of network information, even if those updates come at a high frequency. The GIS must assure that updates are rapidly propagated to consumers.

We posit that the total number of highly dynamic objects will exceed the number of stable objects and the rate of update to these objects must be flexible to meet the needs of the timeliness of the data contained in the attributes. Hence, we are studying the update mechanisms of relational GIS systems, testing the following suppositions:

- The updating mechanisms supported by GIS must go beyond hand-coded programs or scripts.
- Because of the projected number of dynamic objects, updates to objects must be pushed to the GIS, not pulled by GIS.
- The GIS must be prepared to push updates to consumers, or at least provide trigger conditions [21].
- The GIS must impose limits on the rate at which data is pushed to it. Too frequent updates will result in stale data being provided to applications.
- Processing to filter update messages to reduce update frequency must be done upstream of the GIS.
- The GIS could benefit from the CORBA-like event stream model. The publish/subscribe model inherent in event channels could be used to allow GIS to selectively subscribe to the event channels of producers publishing update messages [19, 9].

RDBMSes provide high performance for updates (an important aspect of the TPC-C benchmark [38]) which we can exploit within a GIS. Figure 4 gives early evidence of the performance of GIS updates, for inserting and deleting hosts using a schema similar to that of Figure 3. The RDBMS is MySQL 3.23.36 running on Red Hat Linux 7.1 on a dual 1 GHz Xeon machine with 2 GB of RAM and 240 GB of RAID storage.

5.4 Time-bounded non-deterministic queries with optional heuristics

The nature of GIS queries is similar to decision support queries (TPC-H [38]) in relational database systems. GIS queries will require joins over numerous tables at distributed locations. This has the potential for introducing serious performance problems, both for particular queries and also for the GIS as a whole. However, the nature of the answers that GIS users require is also different from that of traditional decision support queries. In particular a typical GIS user is less interested in getting all the answers from a query than in getting some answers within a limited time.

We plan to exploit this different nature by extending the SQL query model and select statement. First, we will allow the user to state that a query is non-deterministic. This means that the database is permitted to return any subset of the full query results. The second extension is to introduce a time bound, an upper limit on how much time the database should work in constructing its non-deterministic answer. The final extension is to allow the user to specify an optional search heuristic. The search heuristic and non-determinism are not intended to be at loggerheads—the heuristic may only guide search, the system may still return different results each time a query is issued with the same heuristic. Figure 3(b) shows an example of a query that uses all of these extensions.

We believe that time-bounded non-deterministic queries can be implemented as heuristically-guided stochastic search on top of an existing database system. While work has been done in the database community on returning partial results, particularly in extending SQL to enable query writers to explicitly limit the cardinality of a result [5], to “get some answers quickly, and perhaps more later” [37], and to do statistical random sampling of databases [29], we are sure that by focusing on the specific needs of GIS systems and users, we
can produce specific solutions that provide better performance and more flexibility with less overhead.

Figure 5 provides an early example of the performance benefits a non-deterministic queries similar to the one in Figure 3(b). The key observation is that we can easily trade off between the number of results returned and the time of the query using standard SQL. The selection probability is the fraction of “raw” table rows that are used in the query and is related to the time limit. In (b), we limit the number of results to 1000 to keep query times reasonable as we do larger joins.

5.5 Unified query model

There are times when information is needed from entities that either fail to meet the criteria for inclusion in a GIS system, or else whose inclusion appears impractical. Data streams, a prominent example, are a commonly occurring grid entity that does not meet the strict criteria, but has high value to a small set of users. A data stream is an event flow from one or more suppliers and consumers, and possibly intermediate computation to transform, aggregate, or filter data.

<table>
<thead>
<tr>
<th></th>
<th>Join buffer size (µs)</th>
<th>Overhead (µs)</th>
<th>Query (µs)</th>
<th>Total (µs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>dQUOB Query</td>
<td>1</td>
<td>34.07</td>
<td>87.00</td>
<td>121.07</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>32.35</td>
<td>87.31</td>
<td>119.66</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>32.43</td>
<td>84.56</td>
<td>116.99</td>
</tr>
<tr>
<td>'Static' Query</td>
<td>1</td>
<td>16.65</td>
<td>2.78</td>
<td>19.43</td>
</tr>
</tbody>
</table>

Table 1. dQUOB query processing cost compared to 'static' query cost (on a per event basis).

The grid community, particularly the Grid Performance Working Group, is working on schemes for extracting information from event streams. These schemes are conceptually quite different from those used to extracting information from a GIS system. We propose a simpler, unified model that views data streams as data objects that happen not to be resident in the GIS. These objects can then be queried using SQL just as other objects in the GIS. This can be accomplished by viewing a data stream “object” as a relation, where an event is a tuple belonging to one of the relations. This model is similar to that of the RGMA project [10] in the European grid effort.

This work leverages off the dQUOB system [32], wherein an SQL query is evaluated purely over data streams instead of tables in a database. Queries can be managed remotely, inserted into a stream at runtime, and optimized at runtime in response to information gathered about the data and from the environment. Figure 1 shows performance of dQUOB query processing for moderately complex queries. Query processing time is compared against a static version of the query, that is, against a hand-coded and hard-compiled version of the query. The measurements were taken on a single processor Sun Ultra 30 247MHz workstation running Solaris 7. The results demonstrate that moderately complex queries execute in the hundred microsecond range.

It will be possible to associate computation with each query to refine or transform the data. In this way, a query can create a “view”, that is, a new type of relation that can serve as input to another query, thus allowing multiple and successively refined views on the data. One of the clear advantages of a unified model comes in the enhanced ties between views on data objects in the GIS and data streams outside the GIS.

5.6 Decentralized administration

One clear benefit of a global hierarchical name space is that it naturally decomposes across administrative boundaries. Localized control within an administrative domain is essential for a wide area solution if broad acceptance is to be achieved. The authors of DNS some fifteen years ago had the uncanny foresight to recognize this. The upper layers of the DNS IP name space (e.g., .edu, .gov, .com) are under tight centralized control with strong replica consistency, while the lowest layer (e.g., cs.indiana) is managed locally. The Grid faces a similar administrative domain problem. The solution we propose acknowledges the success of DNS with a hierarchical decomposition of the global name space into top and middle layers under centralized control. But at the lowest level, where the resources themselves reside and the need for complex relationships is greater, administrators can be free to choose the implementation that best suits their needs. They may choose to decompose the hierarchy all the way down to the leaves, or may instead choose a relational representation.

To enable interoperability between levels, the interface language must be broad. Specifically, it must go well beyond the LDAP protocol with its limited syntax for query and update specification. The Open Grid Services Architecture (OGSA) effort [15] is a positive step towards standardization on a richer protocol.
6 Conclusion

We have outlined an approach to grid information services that is based on the relational data model and unifies static and dynamic information within a single framework. This approach provides a powerful value proposition to adaptive applications but raises a number of research questions. We laid out these questions and how we plan to address them and provided early performance results for parts of the system that we are building.

References


