

Building Performance Topologies for Computational Grids*

Martin Swany and Rich Wolski
Department of Computer Science
University of California
Santa Barbara, CA 93106
{*swany,rich*}@cs.ucsb.edu

Abstract

This paper describes the architecture, implementation and performance of a service for the delivery of dynamic performance information in Grid environments. Based on usage requirements gleaned from real applications being developed as part of the GrADS project, we have implemented a high-performance service for use by Grid schedulers. The organization of the system is discussed and performance results are presented.

1 Introduction

In a Computational Grid setting, the Information Service (IS) is a key component. Experience has shown that schedulers for the Grid [1, 4, 5, 12, 14] require high performance and timely delivery of IS data – particularly performance data. If the IS is slow, the scheduler itself will be slow, thereby negatively impacting the user’s perception of application performance. Further, it is critical that the IS be accessible in as “open” a fashion as possible – with few restrictions on protocol syntax or programming interfaces.

The Network Weather Service (NWS) [19, 20] is a system for collecting and managing dynamic performance data that is designed to meet these goals by automatically adapting the behavior of its internal components to changing performance conditions. The internal protocols and management strategies that the NWS uses to adapt its own execution are not intended to be visible across the client interface (because they may change, they are complex, etc.) To maintain the necessary flexibility and interoperability, these details must be hidden with a minimum of sacrificed client performance. In this paper, *we describe a new approach for serving dynamic performance data that is designed to accommodate flexibility currently required by emerging Grid efforts without sacrificing the performance of the data delivery mechanisms.*

This work is part of a larger effort – the Grid Appli-

*This work was supported, in part, by a grant from the National Science Foundation’s NGS program (EIA-9975020) and NMI program (ANI-0123911) and by the NASA IPG project.

cation Development Software (GrADS) [3, 11] project – that investigates a comprehensive Grid programming approach. GrADS-developed automatic program schedulers require fast and robust delivery of performance data in order to make scheduling decisions at run-time. Without the optimized abstractions and the caching infrastructure we have developed, these schedulers (which run before and during a program’s execution) must wait unacceptably long periods of time for resource performance data to be delivered. This waiting time is incurred as program execution delay and hence negatively impacts delivered application performance. Our work meets the performance needs of GrADS schedulers while remaining flexible enough to support a variety of IS data infrastructures (such as the Globus MDS-2 [6]).

We believe that the abstraction and service that we have developed to support high-performance data delivery are useful in contexts other than that of the GrADS software tools. As standard data models for the Grid emerge, applications and users will still require the ability to absorb and manipulate performance data using a variety of representations and formats. Our goal is to provide a framework that will enable this flexibility while, at the same time, maintaining the performance integrity of the underlying monitoring system – the NWS. Indeed, in the new era of Grid Computing that is envisioned for the Open Grid Service Architecture (OGSA) [8], we believe that it is important for performance objects to be defined so that they can be delivered by a variety representational mechanisms. However, this flexibility cannot come at the expense of performance.

At present, however, much of the current practice in Grid computing uses the Globus Metacomputing Directory Service (MDS) [6] and/or the Lightweight Directory Access Protocol (LDAP) [18] for resource discovery and information retrieval. LDAP imposes a particular structure on the data that it serves. This paper addresses our approach to using a data model from within this presentation mechanism that is designed to support high-performance data delivery for Grid scheduling. Our solution is enabled by the NWS’s caching infrastructure described in previous work [16]. In this paper, we describe the use of this infrastructure to support *VO-grids* – a new high-performance abstraction that enables resource performance discovery. VO-grids follow

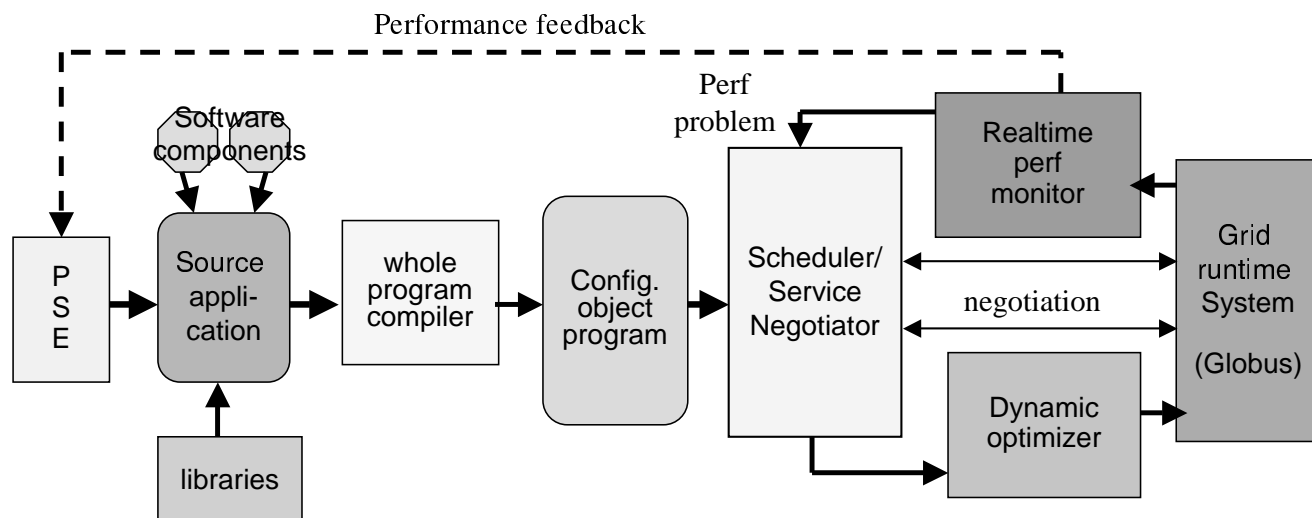


Figure 1. The GrADS Architecture.

the Globus “Virtual Organization” model [6] in design by allowing users to set up virtual collections of resources within a more global Grid resource pool. Network Weather Service VO-grids provide dynamic performance forecasts in multi-dimensional arrays that can be constantly and asynchronously updated. As such, user applications can index these performance arrays with very little programming and execution overhead. At the same time, it is critical that the VO-grid interface be one that can be supported by the Grid performance monitoring and forecasting system as it scales up to very large Grid sizes. Using the hierarchical forecasting infrastructure supported by the NWS, we describe how VO-grids can be constructed scalably using the current Grid Information System infrastructure as a framework.

One may wonder whether our solution is unique to the LDAP interface. We believe that it is not. It is clear that in the context of distributed resources and data stores that the requirements for application-specific caching and indexing are still key to effective operation. We believe, however, that the hierarchical structure of the data model is necessary to support scalability, regardless of the underlying technology in play.

As such, we have implemented the functionality necessary to build VO-grids as a separate, extensible service. The *NWS Topology Service* extracts forecasting information from NWS to build VO-grids based on user-specified requirements. Our early experiences with the *NWS Topology Service* indicate that the performance of the system (described herein) represents a dramatic improvement over the current state-of-the-art in Grid performance data management. We present these results as part of the ongoing work in the Grid Application Development Software (GrADS) [3] project which focuses on development software frameworks for high-performance Grid programs. In addition, this system will be part of the NSF’s Middleware Initiative (NMI) release and the general deployment version

of the NWS.

Briefly, then, this paper outlines three novel contributions.

1. It describes VO-grids as a new, high-performance and scalable API and set of data structures for enabling Grid performance discovery and scheduling.
2. It details a generalized data model used by a prototype VO-grid implementation we have developed, which we believe will extend to encompass a variety of presentation formats.
3. It presents the a brief overview of the *NWS Topology Service* – an extensible facility for building and maintaining VO-grids.

We report on preliminary performance observations we have made of the system using the GrADS ScaLAPACK [12] dynamic scheduler as an initial VO-grid client.

2 GrADS

As mentioned previously, this work was developed as part of the GrADS [3, 11] project. The goal of the GrADS project is to investigate comprehensive software environments for developing Grid applications. Figure 1 depicts the structure of the GrADS software architecture.

Before execution, a Configurable Object Program is prepared by the compilation systems. When the program is to be launched, the Scheduler/Service Negotiator (S/SN) interacts with a variety of run time services provided by the Grid fabric to make decisions about program configuration and scheduling. In particular, the system requires current short-term forecasts for resource performance levels so that it can make proactive scheduling decisions. The NWS generates such forecasts automatically, but to maximize use-

fulness, they have to be delivered to the S/SN (through the Globus [7] infrastructure) quickly and reliably.

3 Design Considerations

The Grid scheduler requires predictions of end-to-end network performance between some set of hosts. We observe that regardless of how this data is served to the scheduler, it is usually treated as a two-dimensional matrix of performance characteristics between machines. Using this data structure, each machine is given an index, and the network performance (typically bandwidth or latency) between any two machines i and j is stored in the matrix element corresponding to the ordered-pair (i, j) .

It is our experience that this information can be delivered through a variety of language-specific or service-specific APIs. However, once delivered, almost all Grid schedulers we have encountered (GrADS or otherwise) use the information to form two-dimensional matrices. Our goal in this work is to use the internal information about how the data is managed to provide a high-performance, general interface for delivering these data structures to the scheduler.

Note that there are many ways to represent this information other than a matrix. Indeed, it has been suggested that linked structures reflecting the “true” topology of the network might prove to be a better interface data structure. If future schedulers require such a data structure, we believe that the mechanisms we have developed can be easily adapted. However, in a situation where there is no library interface into which we can embed the logic needed to construct a host grid from this annotated graph, we are simply forcing a Grid scheduler or Grid program to do the work. To date all schedulers we have encountered attempt to form an indexed matrix from the data presented (either explicitly or implicitly), regardless of how it is delivered. As such, we take our cue for the VO-grid API and matrix data structure from the user community at large.

The scalability of our approach is a second potential concern. In particular, it is not feasible for the underlying monitoring system (in this case, the NWS) to maintain a database of N^2 measurements and forecasts explicitly. Instead, we rely on the hierarchical structure of the NWS clique mechanism [10, 20] to provide a scalable way to estimate end-to-end performance (see Figure 5). A more complete description of how our system populates the full N^2 matrix from a hierarchical measurement topology is given in Section 5 of this paper. As a design requirement, however, we recognize that the data structure that will be presented to the application scheduler must be one that can scale to large numbers of resources.

Finally, we note that the Grid interface and data representation landscape are changing. In particular there have been, and continue to be, a variety of desirable presentation formats for the data such as LDAP, XML [22], and Java objects. These formats, however, do not always offer equivalent performance characteristics. As such, the NWS uses its own optimized wire protocol and adaptive messag-

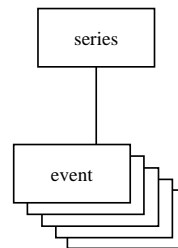


Figure 2. Event elements under their parent Series object

ing system [2] internally. VO-grids are implemented in the caching layer, described in other work [16]. Key to achieving multi-presentation flexibility without sacrificing performance is the object model we have chosen, which we describe briefly in the next section and is also described more completely in [16].

Finally, we make note of the fact that other scheduling methodologies might not need to know the full interconnect matrix, but may wish to query an information system for the connection characteristics between two nodes that are fixed in the configuration for some reason. That is, a full matrix of information is not always required by each NWS client. Since the full matrix describes a fully interconnected, directed graph, all subsequent topologies are, in effect, subgraphs of this general representation. As such, the performance and robustness characteristics of any other topology served by our system will be no worse than for the full-interconnected case since all other topologies are simple extractions from this most general representation.

In summary, our implementation recognizes three key design requirements:

1. Network aware applications require multi-dimensional “performance Grids” (termed VO-grids) to be extracted from a pool of networked compute elements.
2. It must be possible to construct VO-grids scalably and to deliver the data structures that compose the VO-grid API with the minimum possible impact on application performance.
3. The VO-grid API should not be tied to a particular presentation format or set of wire protocols.
4. VO-grids are the general representation of arbitrary performance topologies, each of which can be served with similar performance and reliability characteristics.

4 Data Model and Objects

The NWS takes measurements of various resources and uses statistical techniques to produce forecasts [20] of the

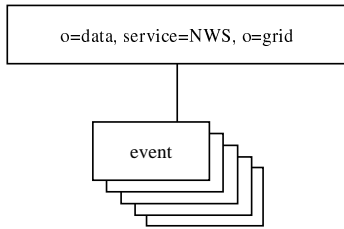


Figure 3. Event elements under the o=Data

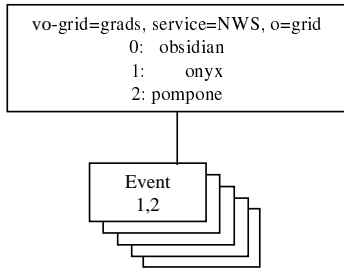


Figure 4. Event elements with indices under a VO-grid object

future performance for those resources. Clients may query the system for the forecasts (or measurements) from a specified resource using the native NWS API or the caching LDAP daemon described in [16]. Often, a client will make repeated requests to the system for a group of logically related data. The basic mechanism of the VO-grids approach allows multiple related data elements to be returned by a single query to the system.

Normally, a query is made for measurements (and forecasts of those measurements) that are being taken from a single host or between a pair of hosts. When a Grid scheduler, such as those being developed by the GrADS [11] project, begins to run, it will ask for an information set that defines the “state” of some resource pool. If a decision must be made about where a job will go in a Grid environment, resource characteristics such as available memory and processing power on individual hosts, and their interconnection qualities to other hosts are typically relevant. This set of information, scoped by a Virtual Organization [6], forms a VO-grid.

The NWS treats measurements as time series, recording the timestamp and the measurement whenever a performance reading is gathered. As such, we term each measurement a “measurement event” to indicate that it has both a measurement value and a time coordinate in its most general form. In [16], these ordered pairs are identified as **GridEvent** objects which are grouped by an association with a common name called a **GridSeries** object. This relation-

ship is depicted in Figure 2. In the NWSlapd [16] implementation measurements can also be addressed directly under the *o=Data* branch of the directory information tree, as seen in Figure 3.

In practical terms, a GridSeries object acts to name an index over a set of **GridEvents** and thus to limit the scope of queries over the global information base. For instance, using LDAP, a base of *series=nws.cs.ucsb.edu:8060 . bandwidthTcp.32.16.64 . nws.cs.utk.edu:8060, service=NWS, o=Grid* and a filter of *time stamp > 1015015530* is equivalent to the SQL statement *select * from event where name like 'nws.cs.ucsb.edu:8060 . bandwidthTcp.32.16.64 . nws.cs.utk.edu'* and *timestamp > 1015015530 order by time stamp*. Similarly, we can also create an LDAP base that provides a query for all results from a given host, activity or clique.

Thus, the VO-grid is an index over the **GridEvent** pool that covers an N^2 matrix of end-to-end performance values as discussed above. It is useful because it generalizes to represent all measurements that are named by pairs of resources (e.g. end-to-end network measurements) and is efficient for use within the implementation of an application scheduler.

Moreover, we have also observed that many Grid Information System users find it cumbersome to form restrictive queries themselves, but instead post overly broad queries that they then filter locally using some additional utility (e.g. *grep*). We believe that it is because the data is not served in an appropriate format (like a VO-grid) that current Grid programmers and users are prompted to make this extra level of effort. Note, however, that global queries (no matter how cumbersome the alternative may be) will become less and less practical as the scope of the Grid continues to grow. Only scoped subsets of information can be (or need to be) addressed by any given index or query. Abstractly, this notion of scoping is the basis of the “Virtual Organizations” described in [6]. VO-grids combine this scoping with a general and efficient data representation. By creating VO-grids of dynamic performance information, users will find the system easy to use and efficient implementation will still be practical.

The **VO-grid** object is depicted in Figure 4. It is an object that contains meta-information about some collection of data elements and acts as their parent in the LDAP hierarchy. In this example, “*VO-grid=GrADS, service=NWS, o=Grid*” is such an object. Note that the **VO-grid** object contains a list of the hosts in this grid of information along with their indices. The children of this node are a collection of the appropriate data elements that are “joined” (in the database sense) with an ordered pair of the appropriate indices in this grid. By “joining” the data elements with their appropriate indices, the **VO-grid** can be trivially mapped into a two dimensional array in a user program.

The following is an example of an object returned by a **VO-grid** query.

```

dn: forecast=opus0.cs.uiuc.edu:8061.
   bandwidthTcp.1024.1024.1024.mystere.
   ucsd.edu:8061,service=NWS,o=Grid
objectClass: top
objectClass: service
objectClass: GridForecast
forecast: opus0.cs.uiuc.edu:8061.
   bandwidthTcp.1024.1024.1024.mystere.
   ucsd.edu:8061
index0: 20
index1: 44
timestamp: 1042849485
value: 3.254350
mse-forecast: 3.625830
mse-error: 1.351129
mae-forecast: 3.803970
mae-error: 0.968662

```

Note that in this particular object, the value and two different forecast values are returned.

Since our object model is normalized, it allows us to compose objects as we see fit. In the case of the network performance grid, latency and bandwidth can be joined to form a composite network characterization object. In addition to networking information, there are other metrics that are valuable in scheduling for the Grid – processor utilization and available memory. These data elements are logically separate – they are gathered and stored independently – but it is useful to join them in a “host status” object as well.

The current implementation allows a VO-grid to be specified via a configuration file or with a reference to the Grid Information Index Server (GIIS) of a Virtual Organization. The configuration file option is simply a list of hosts to be considered. The VO option allows a host grid to be specified with an LDAP query consisting of the tuple of server, base and filter.

The creation of a VO-grid can also be done dynamically with an LDAP modify or insert operation given that appropriate security mechanisms are used to prevent abuse. After various experiments with the system, the importance of dynamic VO-grid creation became apparent. Currently, a user-specific VO-grid is created with an LDAP modify operation using an object such as this one:

```

dn: vo-grid=mygrid,service=nws,o=grid
grid-member: torc0.cs.utk.edu
grid-member: torc1.cs.utk.edu
grid-member: torc2.cs.utk.edu
grid-member: torc3.cs.utk.edu
grid-member: torc4.cs.utk.edu
grid-member: torc5.cs.utk.edu
grid-member: torc6.cs.utk.edu
grid-member: torc7.cs.utk.edu
grid-member: torc8.cs.utk.edu
grid-member: opus0.cs.uiuc.edu

```

```
grid-member: mystere.ucsd.edu
```

This “object” is not created, but rather drives the creation of a VO-grid. Queries against the newly created name look the same as those shown above.

4.1 Relation to XML-based Systems

At first glance this system may seem to exist only to deal with the peculiarities of the LDAP interface. However, we contend that the notion of scoping and the need to use it to control the performance of information-system queries is generally applicable to other information presentation mechanisms as well. As such, we have implemented a prototype VO-grid system that supports delivery of the same objects in both LDAP and XML (via the Simple Object Access Protocol (SOAP) [13]). This system was designed to support the Grid Monitoring Architecture (GMA) [17] as well as the MDS2 [6]. Clearly, these design constraints also anticipate the requirements of the emerging Open Grid Service Architecture (OGSA) [8] and provide a migration path from LDAP to XML, SOAP and WSDL [21] (the core upon which OGSA technologies are being defined).

To ease migration to this new technology, we have implemented the XML/SOAP version with an analogous object structure to the LDAP version. The following is an example of the WSDL describing the VO-grid service.

```

<complexType name="forecast">
  <all>
    <element name="forecast"
      type="xsd:string"
      minOccurs="1"
      maxOccurs="1"/>
    <element name="timestamp"
      type="xsd:int"
      minOccurs="1"
      maxOccurs="1"/>
    <element name="index0"
      type="xsd:int"
      minOccurs="1"
      maxOccurs="1"/>
    <element name="index1"
      type="xsd:int"
      minOccurs="1"
      maxOccurs="1"/>
    <element name="value"
      type="xsd:float"
      minOccurs="1"
      maxOccurs="1"/>
    <element name="mseforecast"
      type="xsd:float"
      minOccurs="1"
      maxOccurs="1"/>
    <element name="mseerror"
      type="xsd:float"

```

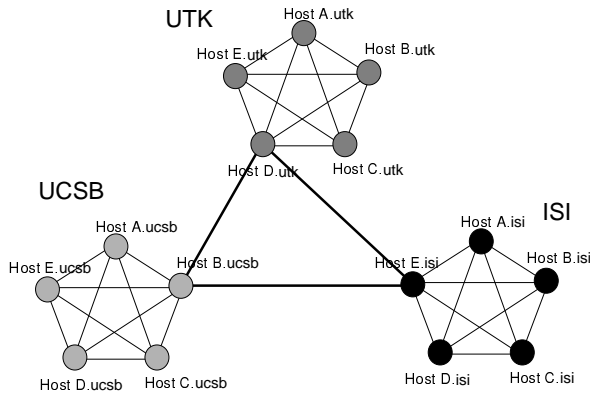


Figure 5. A hierarchy of cliques

```

    minOccurs="1"
    maxOccurs="1" />
<element name="maeforecast"
  type="xsd:float"
  minOccurs="1"
  maxOccurs="1" />
<element name="maeerror"
  type="xsd:float"
  minOccurs="1"
  maxOccurs="1" />
</all>
</complexType>

```

It is easy to see that these objects are a simple translation from the LDAP version. In this way, the protocol is abstracted away while the semantics of the service remain the same.

5 Topology System

An appropriate data model and abstraction (as described in the previous section) do not, by themselves, alleviate the tension that exists between the need to present accurate, up-to-date information to clients, and the ability to scale the system to large numbers of hosts. In this section, we address a few of the service architecture characteristics that are necessary to build and deliver VO-grids efficiently. Some of the system design issues have been explored previously in work such as the IDMaps [9] project, which shares a similar set of goals. However, note that this section does not comprise a complete description of the NWS Topology Service; rather we refer to [15] for complete details. The focus of this paper is on the “performance topology” as presented to schedulers, as opposed to the actual topology that is used to generate it.

The first feature of the NWS approach that allows for scalable measurement gathering is the *clique structure* (also

	onyx	crow	ash	rave	trc1	trc2	trc3	trc4	opus	amaj	bmaj	cmaj	dmaj
onyx	X	X	X	X	X				X				
crow	X	X	X	X									
ash	X	X	X	X									
rave	X	X	X	X									
trc1	X				X	X	X	X	X				
trc2					X	X	X	X					
trc3					X	X	X	X					
trc4					X	X	X	X					
opus	X				X				X	X	X	X	X
amaj									X	X	X	X	X
bmaj									X	X	X	X	X
cmaj									X	X	X	X	X
dmaj									X	X	X	X	X

Figure 6. Partial Grid of measurements

	onyx	crow	ash	rave	trc1	trc2	trc3	trc4	opus	amaj	bmaj	cmaj	dmaj
onyx	X	X	X	X	X	X	X	X	X	X	X	X	X
crow	X	X	X	X	X	X	X	X	X	X	X	X	X
ash	X	X	X	X	X	X	X	X	X	X	X	X	X
rave	X	X	X	X	X	X	X	X	X	X	X	X	X
trc1	X	X	X	X	X	X	X	X	X	X	X	X	X
trc2	X	X	X	X	X	X	X	X	X	X	X	X	X
trc3	X	X	X	X	X	X	X	X	X	X	X	X	X
trc4	X	X	X	X	X	X	X	X	X	X	X	X	X
opus	X	X	X	X	X	X	X	X	X	X	X	X	X
amaj	X	X	X	X	X	X	X	X	X	X	X	X	X
bmaj	X	X	X	X	X	X	X	X	X	X	X	X	X
cmaj	X	X	X	X	X	X	X	X	X	X	X	X	X
dmaj	X	X	X	X	X	X	X	X	X	X	X	X	X

Figure 7. Complete Grid of forecasts

described in [20]). Small groups of hosts that are typically interconnected by a fast, reliable network, are designated as individual *cliques*. Each host within a clique directly measures network performance to all other hosts in that clique. Therefore, for each clique of size C , the NWS gathers $C(C-1)$ network measurements for each measured characteristic. The measurements are conducted within the clique are taken sequentially using a token-based mutual exclusion protocol [10, 20] that also implements various fault tolerance features as well. Cliques are dynamically configurable by the NWS administrator and may contain any number of hosts. A host may also participate in multiple cliques simultaneously.

To insure scalability, local-area cliques can be arranged in a hierarchy by designating a “representative” host from each clique to participate in a higher-level clique. The measurements for this distinguished host-pair can then represent the measurement between any hosts located in separate local-area cliques.

For example, consider the clique hierarchy shown in Fig-

ure 5. In it, hosts *B.ucsb*, *D.utk* and *E.isi* participate in a second-level clique. Measurements from *B.ucsb* to *D.utk* can then be used to represent any UCSB-UTK host pair.

For VO-grids, the matrix representation of end-to-end performance between only the hosts that are being monitored is depicted in Figure 6; we would like to transform this in to a fully populated grid, shown in Figure 7. The NWS is able to do so by providing *forecasts* for the areas in which no measurements have been taken. If the cliques of hosts are arranged so that performance between representative end-points is common to all hosts, the forecast data for the representatives can be propagated to the empty parts of the matrix.

Returning to the example depicted in Figure 5, all hosts at UCSB experience similar connectivity characteristics to any host at UTK. As such, the NWS monitors the connectivity between only a distinguished pair of representative hosts – one at UCSB and one at UTK – and then uses that information to represent all UCSB-UTK communication. Since forecast data is used, transient or unpredictable performance responses are not replicated since they are “filtered” out by the forecasters.

An easy way to approach the problem of grouping hosts into a hierarchy of cliques is to use the domain portion of a fully-qualified DNS name and to assume that those form an equivalence class. This approach, however, is only an initial approximation. In practice Domain Name System (DNS) names denote administrative scope and not network topology. The DNS domain *npaci.edu*, for instance, is used by many sites across a wide geographic distance. The IP address of a host, on the other hand, is the definitive location of the host as far as the network is concerned. If it weren't, then traffic wouldn't get there at all! However, it isn't clear from looking at most pairs of IP addresses whether they actually share a subnetwork or not. That fact can only be determined from the tuple of address and netmask. This is why the Topology Service should provide this information to user programs rather than having them derive it independently: the Topology Service can do so with additional information that is perhaps not available to end-user programs.

Initially, on the GrADS testbed, the clique structure has been specified so that topological relationships are explicit in the clique hierarchy. As such, the NWS published the DNS names and IP addresses of the “clique leaders” so that scheduling systems could use this information to form their own complete host grid internally. The VO-grid deployment for GrADS takes this structure into account when building the complete matrix automatically.

To implement this this prototype for GrADS, we simply used combinations of IP addresses and subnet masks to form the basic set of equivalence classes. More generally however, this approach discards a great deal of information that is apparent in the relations between Autonomous Systems and potentially ignores the effects of Layer 2 tunneling and the virtual private networks. We are developing a far more comprehensive topology service that takes much

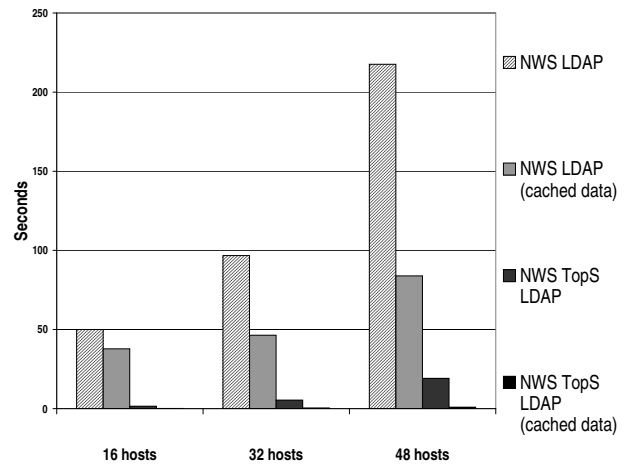


Figure 8. LDAP queries to local infrastructure

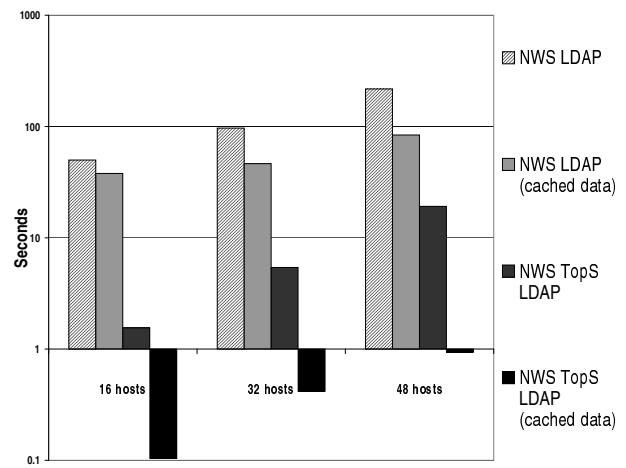


Figure 9. LDAP queries to local infrastructure (log scale)

of this into account as part of our current extensions to this work.

6 Results

Our first implementation of VO-grids for GrADS was based on the NWS's caching LDAP daemon, which is described in [16]. As part of the GrADS project, a Grid-enabled version of ScaLAPACK [12] has been developed that uses the LDAP interface to the NWS. We found that despite the performance enhancements documented in [16], that we could optimize data delivery even further. The following results represent response time averages for 5 identical tests.

Figure 8 shows a comparison of NWS LDAP and NWS Topology Service query times for the full N^2 VO-grid matrix of bandwidth forecasts required by the GrADS ScaLA-

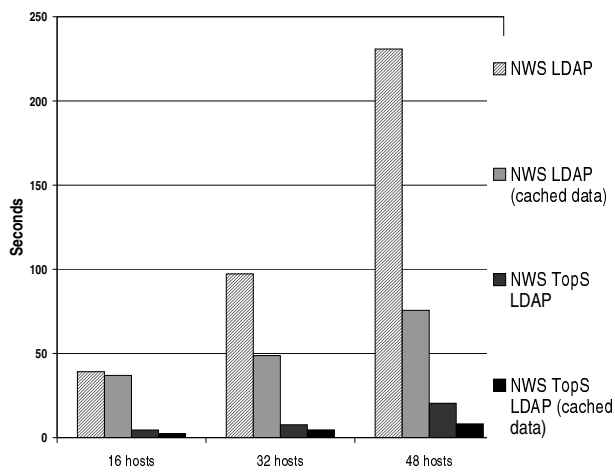


Figure 10. LDAP queries to remote infrastructure

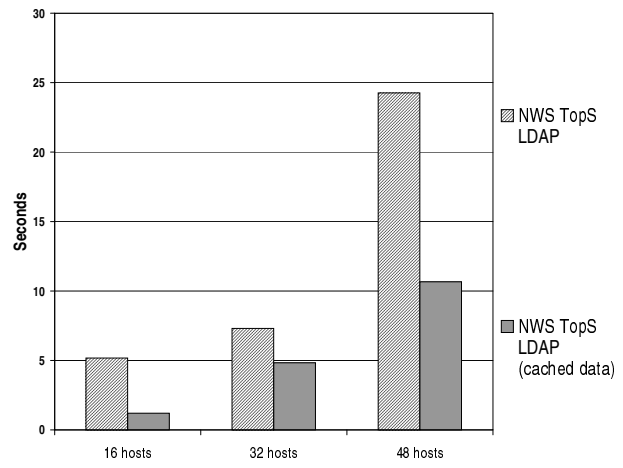


Figure 12. LDAP queries to nearby infrastructure

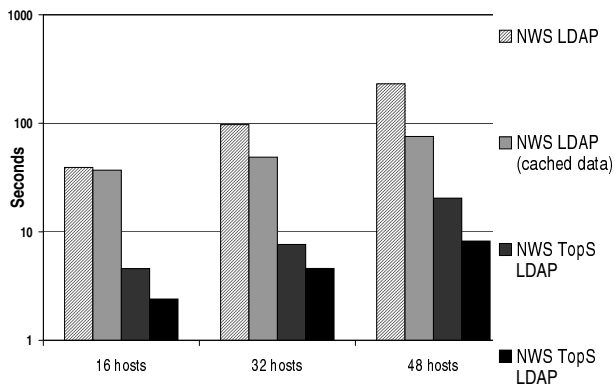


Figure 11. LDAP queries to remote infrastructure (log scale)

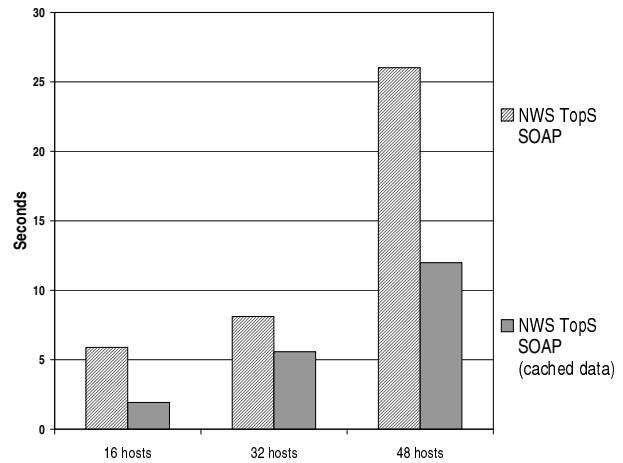


Figure 13. SOAP queries to nearby infrastructure

PACK code when the client and the NWS server are co-located. The first (leftmost) bar for each host count is the total fetch time (in seconds) for fetching the N^2 elements using the caching NWS LDAP server when the data is not in cache. The next bar from the left is the fetch time if the data is in cache. The third bar from the left shows the cold-cache fetch time from the NWS Topology service. The last bar (rightmost) shows the cached fetch times.

Comparing cold-cache performance demonstrates the effectiveness of having the information system (as opposed to the application-level components) aggregate the data. Because the NWS Topology service can incorporate intimate knowledge of how the NWS manages its data internally, it can optimize the data aggregation. The cached times show the performance that the GrADS ScaLAPACK client actually achieved for all but its first query. The range of values

is such that a cache hit against the NWS Topology Service LDAP server isn't even discernible. We have included a log-scale plot of the same data in Figure 9.

Similarly, Figure 10 shows the same comparisons over the same range of host counts when the client was located at U. Tennessee and the NWS daemons were running on a host at UC Santa Barbara. Figure 11 shows these results on a log-scale plot as well. Clearly, in either the local or remote access cases, the NWS Topology Service implementing a VO-grid for GrADS is able to achieve relatively high-performance levels across the scale of the GrADS testbed.

From a practical standpoint, the result of using VO-grids and the NWS Topology Service to deliver them is to make the overhead of dynamic information access a negligible component of the overall Grid overhead. For 48 hosts, the local access time is less than a second. The GrADS ScaLA-

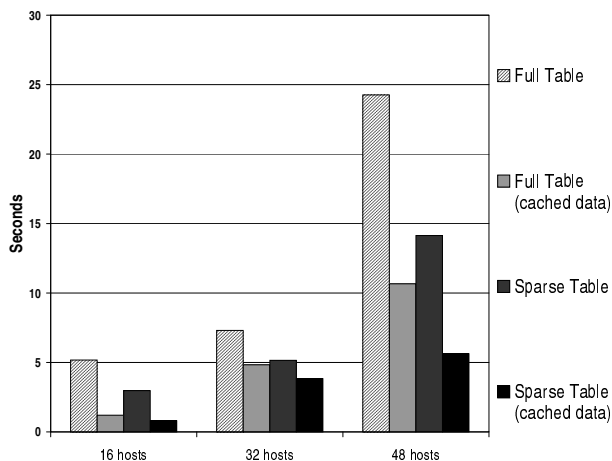


Figure 14. Comparison of delivery times with different VO-grid encodings.

PACK computation executes for approximately 30 minutes in its optimal configuration, making the local fetch times well under the measurable performance variation of the core computation by itself. In the remote access case, where the complete set of N^2 values must be sent from the NWS Topology service daemon to the LDAP client built into the scheduler, the time required is still less than 10 seconds.

Next, we evaluate the performance of the system when dynamic creation of VO-grids is enabled. The GrADSoft system has been updated to include the creation of an ephemeral VO-grid based on client-side configuration. This mechanism requires an additional interaction with the server and the execution of the logic necessary to complete the VO-grid view. Figure 12 shows queries of various sizes from UC San Diego to UC Santa Barbara. Cases with and without cached data are represented. Here we see the additional overhead of the run-time creation of the VO-grid, but the performance is still acceptable. Also note that for this test, two separate forecast values were computed – adding to the overhead even further.

Further evaluation of this implementation involves evaluation of the performance of the parallel SOAP/XML delivery mechanism introduced in Section 4.1. Figure 13 shows an experimental configuration identical to the one depicted in Figure 12 differing only in that the SOAP/XML version is used. The results show that the overhead of SOAP is only slightly higher than that of LDAP. However, we note that the SOAP version is less mature and further improvement is possible.

Finally, we explore one final optimization. Depending on the configuration of a VO-grid, there is some amount of redundant data that is passed from server to client. As the number of hosts grows, this redundant data becomes more of a performance factor. To combat this we have implemented a method where redundant data is suppressed and the encoding of the VO-grid becomes more analogous to

a “sparse matrix” representation. However, rather than regions of zero-valued elements, the VO-grid structure has regions of same-valued elements. We implemented a test of this procedure with the GrADSoft infrastructure (since we can encode the logic to reconstitute a “dense” matrix from a “sparse” one in the GrADS libraries.) Figure 14 shows the average response time using this sparse VO-grid representation for queries. This type of encoding clearly improves response time as the size of the result set grows.

These results demonstrate the validity of our technique across a wide range of conditions. We note, however, that further optimizations of VO-grid delivery are possible. Still, the VO-grid technique remains a valuable initial step in the evolution of performance information systems for Grid computing.

7 Conclusion

Our approach to delivering end-to-end performance readings to Grid applications is two-fold. First, we define VO-grids so that information system queries can be scoped effectively. VO-grids take advantage of an underlying data model we have defined for performance data and NWS forecasting techniques which enable scalability. While VO-grids are a general abstraction, we have implemented them in the form of a cached index within the framework of the Grid Information Service (as defined by the Global Grid Forum). The caching service is, itself, a component of the NWS Topology service – a scalable mechanism for managing NWS performance data and VO-grid structures.

While both of these innovations are general, particularly with respect to the Grid Monitoring Architecture (GMA) and the Open Grid Service Architecture (OGSA), they were developed as part of the Grid Application Development Software (GrADS) project. GrADS requires extremely high performance from its information system. At the same time, all of the GrADS application schedulers needed to manipulate end-to-end performance information as an indexed, two-dimensional matrix. VO-grids are our realization of this representation in a form that is general enough to be useful in other contexts.

References

- [1] B. Allcock, I. Foster, V. Nefedova, A. Chervenak, E. Deelman, C. Kesselman, J. Leigh, A. Sim, and A. Shoshani. High-performance remote access to climate simulation data: A challenge problem for data grid technologies. In *Proceedings of SC01*, 2001. http://www.globus.org/research/papers/sc01ewa_esg_chervenak_final.pdf.
- [2] M. Allen and R. Wolski. Adaptive timeout discovery using the network weather service. In *Proceedings of HPDC-11*, July 2002. <http://www.cs.ucsb.edu/~rich/publications/nws-adapt.pdf>.
- [3] F. Berman, A. Chien, K. Cooper, J. Dongarra, I. Foster, L. J. Dennis Gannon, K. Kennedy, C. Kesselman, D. Reed,

- L. Torczon, , and R. Wolski. The GrADS project: Software support for high-level grid application development. Technical Report Rice COMPTR00-355, Rice University, February 2000.
- [4] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.
- [5] H. Casanova, G. Obertelli, F. Berman, and R. Wolski. The AppLeS Parameter Sweep Template: User-Level Middleware for the +Grid. In *Proceedings of SuperComputing 2000 (SC'00)*, Nov. 2000.
- [6] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *10th International Symposium on High-Performance Distributed Computing*. IEEE, August 2001. <http://www.globus.org/research/papers.html#GlobusToolkit>.
- [7] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications*, 1997.
- [8] I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The physiology of the grid: An open grid services architecture for distributed systems integration. January, 2002.
- [9] P. Francis, S. Jamin, C. Jin, Y. Jin, D. Raz, Y. Shavitt, and L. Zhang. IDMaps: A global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, October 2001.
- [10] B. Gaidioz, R. Wolski, and B. Tourancheau. Probes to avoid measurement intrusiveness in the network weather service. In *Proc. 9th IEEE Symp. on High Performance Distributed Computing*, pages 147–154, August 2000.
- [11] GrADS. <http://hipersoft.cs.rice.edu/grads>.
- [12] A. Petitet, S. Blackford, J. Dongarra, B. Ellis, G. Fagg, K. Roche, and S. Vadhiyar. Numerical libraries and the grid. In *Proc. of SC01*, November 2001.
- [13] Simple object access protocol (SOAP). W3C Recommendation, <http://www.w3.org/TR/SOAP>, May 2000.
- [14] N. Spring and R. Wolski. Application level scheduling: Gene sequence library comparison. In *Proceedings of ACM International Conference on Supercomputing 1998*, July 1998.
- [15] M. Swany and R. Wolski. Topology discovery in the network weather service. <http://www.cs.ucsb.edu/~swany/papers/nws-topo.ps>.
- [16] M. Swany and R. Wolski. Representing dynamic performance information in grid environments with the network weather service. 2nd IEEE International Symposium on Cluster Computing and the Grid (to appear), May 2002.
- [17] B. Tierney, R. Aydt, D. Gunter, W. Smith, V. Taylor, R. Wolski, and M. Swany. A Grid Monitoring Architecture. Grid forum working group document, Grid Forum, February 2001. <http://www.gridforum.org>.
- [18] M. Wahl, A. Coulbeck, T. Howes, and S. Kille. Lightweight directory access protocol (v3): Attribute syntax definitions. Internet Engineering Task Force, RFC 2252, December 1997.
- [19] R. Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1998. also available from <http://www.cs.utk.edu/rich/publications/nws-tr.ps.gz>.
- [20] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 1999.
- [21] Web services description language (WSDL) version 1.2. W3C Recommendation, <http://www.w3.org/TR/wsdl12>, July 2002.
- [22] Extensible markup language (XML). W3C Recommendation, <http://www.w3.org/TR/REC-xml>, February 1998.