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 and performance of a network performance topology system for Grid environments. Based on usage requirements gleaned from real applications, we have implemented a high-performance service for delivery of this information called the NWS Topology Service. This implementation is compatible with extant Grid Information Services, and we describe the model for the objects being delivered.

1 Introduction

Performance information systems in Grid environments are subject to a number of unique design considerations. In particular, as adaptive Grid applications evolve, it is becoming increasingly clear that the performance of Grid information systems is a critical component of *application performance* [1, 5]. If the data is not delivered to the application quickly, the application itself runs slowly, regardless of how carefully it is written.

The Network Weather Service (NWS) [9, 10] is a system for collecting and managing dynamic performance data that is designed to meet the performance needs of its clients (as well as their functionality needs). Its current implementation focuses on scalability, robustness and tolerance of drastic performance fluctuation in both the resources it uses and monitors. However, as Grid Computing continues to evolve, it is ever more apparent that a performance information system must remain flexible in the extreme. Internally, the NWS uses a set of protocols and management strategies that are not intended to be visible across the client interface (because they may change, they are complex, etc.) In this paper, *we describe a new approach for serving NWS data that is designed to accommodate extreme flexibility currently required by emerging Grid efforts without sacrificing the performance of the data delivery mechanisms*. It is our contention that even if (when) a standard data model emerges for the Grid, 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 by the Open Grid Service Architecture (OGSA) [3], we believe that it is important that objects be defined in a manner such that they can be delivered by a variety representational mechanisms. Only by anticipating that the system and its data will be used in so many new a different ways can we ready ourselves for change.

At present, however, much of the current practice in Grid computing uses the Globus Metacomputing Directory Service (MDS) [2] and/or the Lightweight Directory Access Protocol (LDAP) [8] for resource discovery and information retrieval. LDAP imposes a particular structure (however flexible) on the organization data that it serves. This paper addresses our approach to using a data model in a presentation layer that is designed to support high-performance resource discovery and Grid scheduling. Our solution is enabled by the NWS's caching infrastructure described previously in [6]. In this work, we describe the use of this infrastructure to support *VO-grids* — a new high-performance abstraction that enables resource performance discovery. VO-grids follow the Globus "Virtual Organization" model [2] 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 are 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

*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.

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.

However, one may wonder whether our solution is unique to the LDAP interface. We believe that it is not. While the implementation of OGSA is only now beginning, it is clear that in the context of distributed resources and data stores, confederated resources and imperfect data 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 (which this paper details) indicate that the performance of the system (also described herein) represents a dramatic improvement over the current state-of-art in Grid performance data management. We present these results as part of the on-going work in the Grid Application Development Software (GrADS) [1] 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 deployment of the NWS.

Briefly, then, this extended abstract 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, that we believe will extend to encompass a variety of presentation formats.

3. It presents the design 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 [5] dynamic scheduler as an initial VO-grid client.

2 Design Considerations

To schedule Grid applications, often the Grid scheduler (human or automatic) requires predictions of network performance, end-to-end, between a set of potentially useful 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 schedulers use the information to form two-dimensional matrices. Our goal in this work is to use the knowledge that the information system possesses 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 will generalize to be able to construct it as well. However, in a situation in which there is no library interface we can embed the logic 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 our 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] to provide a scalable way to estimate end-to-end performance. A more complete description of how our system populates the full N^2 matrix is given in Section 4 of this extended abstract. As a design requirement, however, we recognize that the data structure which will be presented to the application scheduler must be one that we can build scalably.

Finally, we contend that what is important from performance information systems is not the interface to the information system. Getting the data objects, whether it be via LDAP, XML or some other "on the wire" encoding, should be so mundane as to be invisible. At the same time, as standards emerge for information management, we want our system to be able to support them with the same level of performance described herein. The key to achieving this level of extensibility for our system is the object model, which we describe in the next section.

If we can deliver a full mesh of information to sched-

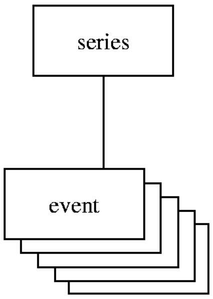

Figure 1. Event elements under their parent Series object

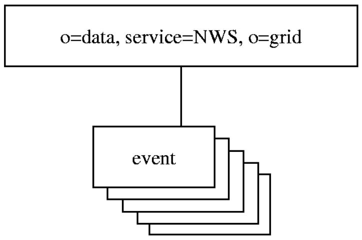

Figure 2. Event elements under the o=Data branch of the tree.

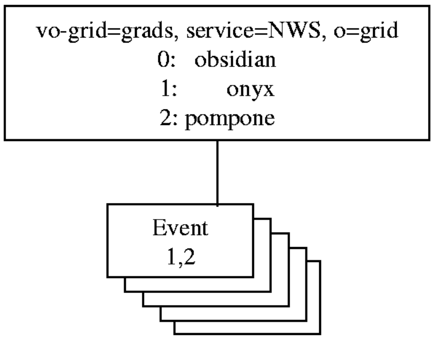

Figure 3. Event elements with indices under a VO-grid object

ulers quickly, then we would make a useful service available as part of the "middleware" of the Grid. Finally, we make note of the fact that other scheduling methodologies might not need to know the full interconnect matrix, but might rather want to query an information system for the connection characteristics between two nodes, that are fixed in the configuration for some reason. In this case, the information system must have the same information as the common case outlined above – the requirements are the same.

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.

3 Data Model and Objects

We have described the object model for the NWS more extensively in [6]. The flexibility that we described is key in our approach to solving this problem. Intuitively, we can think about the answer to the query "what is the bandwidth for a specified resource collection?" as being a set of records (a "cursor" in database terms.)

The data elements in the NWS are objects of the GridEvent (event) type. They are normally grouped under the GridSeries (series) object, which contains meta-information about the series, shown in Figure 1. All events are also available under the *o=Data* branch of the directory information tree, as seen in Figure 2. This is conceptually where application-specific events (i.e., those not part of an ongoing **GridActivity**) are referenced.

Consider that the GridSeries object really just acts to name an index over a set of **GridEvents** and define a parameterized query over that pool. (i.e. 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 "timestamp $>$ 1015015530" is equivalent to "select * from event where name like 'nws.cs.ucsb.edu:8060 . bandwidthTcp.32.16.64 . nws.cs.utk.edu' and timestamp $>$ 1015015530 order by timestamp") Similarly, we can also create a namespace that defines all results from a given host, activity, or clique.

We have implemented an object that creates an index over the GridEvent pool that is useful for schedulers and that we refer to as a "VO-grid." This object defines a set of GridEvents that covers the N^2 matrix of performance values that we discussed in the previous section. It is very useful because this is a natural unit to consider – and this is what schedulers want. Also, we have observed that the many users don't take the time to form restrictive queries, rather posing overly broad questions and filtering them locally. This type of usage 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. This is completely compatible with the notion of the "Virtual Organizations" described in [2]. By creating "VO-grids" of performance information users will still find the system easy to use and efficient implementation will still be practical.

This approach is also useful to perform queries that order or bound by a given value, based on the namespace. This is important given the lack of support for floating point in most LDAP implementations. Also, this mechanism can be used as a "cache preload" mechanism. Indeed, by forming these relatively static queries, they are easier to cache. On the other hand, this does limit the types of queries that can be easily made, but we regard this as an acceptable tradeoff. We contend that, in general, *arbitrary* queries and joins are not necessary. Most of the useful queries are fairly intuitive and can be easily anticipated.

The **VO-grid** object is depicted in Figure 3. Is 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 object and it contains a list of the hosts in this grid with their indices. The children of this node are collection of the appropriate data elements, "joined" with an ordered pair of the appropriate indices in this grid. This can be thought of as a cursor of elements that is returned as a result of query in a relational database system. With index information in each element, the grid can be easily reassembled.

Since our object model is highly normalized, it allows us to compose objects as we see fit. In the case of the network performance grid, latency and bandwidth are joined for 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, gathered and forecast separately, but it is useful to join them in a "host vitals" object.

Finally, how can these grids be specified and created? The current implementation allows for a file-based interface and a GIS-based interface that allows a grid to be specified with and LDAP query (i.e. the tuple of server, base and filter.) There is no reason that this could not be made dynamic with an LDAP modify or insert operation, given that appropriate security mechanisms were used to prevent abuse. We currently have a system operational that creates and caches host grids for the GrADS [1] testbed machines.

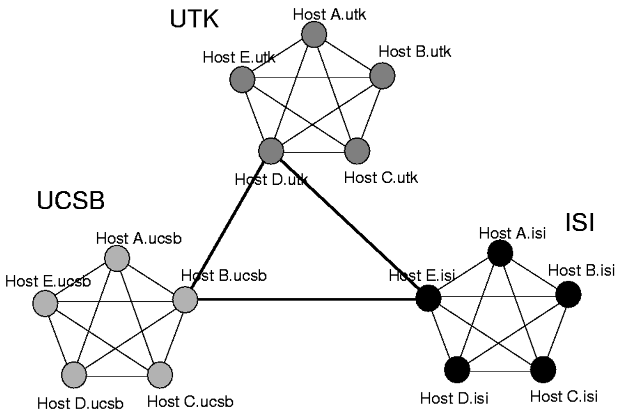

Figure 4. A hierarchy of cliques

3.1 Relation to XML-based Systems

At first glance this system may seem to exist completely to deal with the peculiarities of the LDAP interface. However, it is completely relevant to other information presentation mechanisms as well. We are in the process of implementing a system that will support delivery of the same objects in LDAP and XML to support the Grid Monitoring Architecture (GMA) [7] as well as the MDS. Clearly, these design constraints also neatly anticipate the requirements of the emerging OGSA [3] architecture.

Consider objects containing this same information that are encoded using XML. Even if you could count on the ordering of multi-valued attributes, it would still be the responsibility of the application developer or scheduler to pull the objects apart and reassociate them into a performance grid. "Why not just deliver the actual grid in XML?", one might ask. Discarding the initial impulse to represent each row as a comma-delimited set of values, as this would again require the user to parse the message and would lose the benefit of XML, we assume that each value would need to be tagged. To tag every value and give a structural encoding of the host grid information seems to us to obviate the simplicity of XML encoding. In short, the notion of an index is valuable in this setting as well.

Complete paper will include objects translated into XML.

4 Topology System

The real success in this work lies in presenting effective network information to Grid programs. We have to balance

	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 5. 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 6. Complete Grid of forecasts

the need for accurate network information with a design that allows the system to scale to large numbers of hosts. Some of these issues have been explored previously in work such as the IDMaps [4] system. which shares a similar set of goals.

The NWS allows for scalable measurements to be taken by arranging groups of hosts in which a full mesh of measurements is taken, known as *cliques*, into a hierarchy. This is depicted in Figure 4. This gives us a connectivity grid that is depicted in Figure 5, and we would like to transform this in to a fully populated grid, shown in Figure 6. The NWS is able to do this by providing *forecasts* for the areas in which no measurements have been taken.

The easy way to approach the problem of grouping 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 lacking. In practice Domain Name System (DNS) domains 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. This is determined from the tuple of address and netmask. This information is available on the host, so we require the collusion of the sensor to get this information.

In the GrADS testbed, the clique structure has been specified so that this information is implicit. This allowed the scheduling systems to use the notion of "clique leaders" to form the complete host grid. We note that this is a result of manual configuration and is not a general solution.

The topology system's internals are not the focus of this paper. Future work will detail how measurements can be effectively grouped into the host grids, and how user queries can be used to drive more effective measurement control. For this prototype implementation, we simply used combinations of IP addresses and subnet masks to form the basic set of equivalence classes. This 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 of this into account.

5 Performance Results

This service was implemented in the NWS's caching LDAP daemon, which is described in [6]. As part of the GrADS project, a Grid-enabled version of ScaLAPACK [5] has been developed that uses the LDAP interface to the NWS. We found, despite the performance enhancements documented in [6], that we could optimize data delivery even further. Figure 7 shows a comparison of NWS LDAP and NWS Topology Service query times for the full N^2 matrix of bandwidth forecasts required by the GrADS ScaLAPACK code when the client and the NWS server are colocated. The 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 if 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, and 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

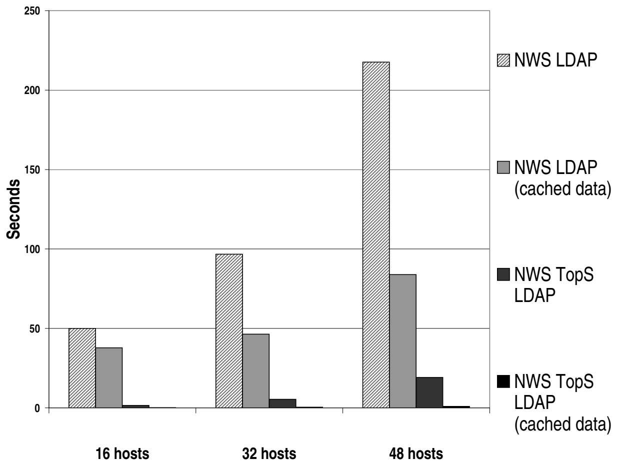

Figure 7. LDAP queries to local infrastructure

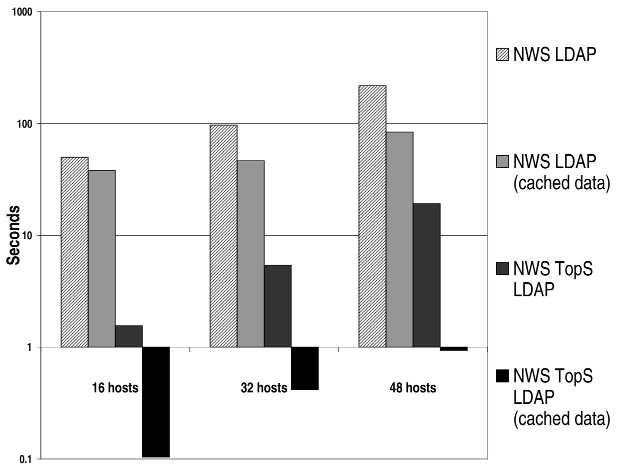

Figure 8. LDAP queries to local infrastructure (log scale)

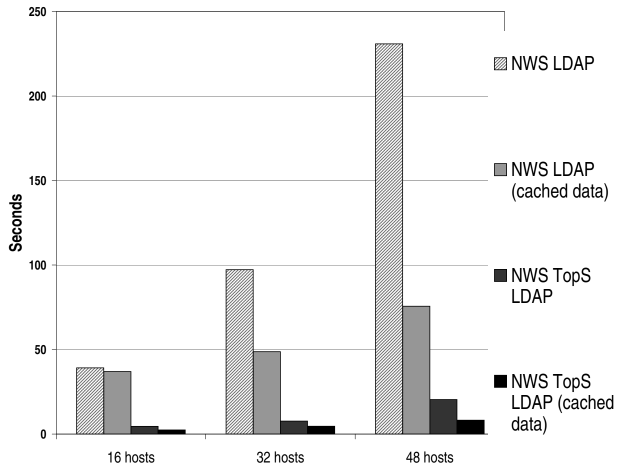

Figure 9. LDAP queries to remote infrastructure

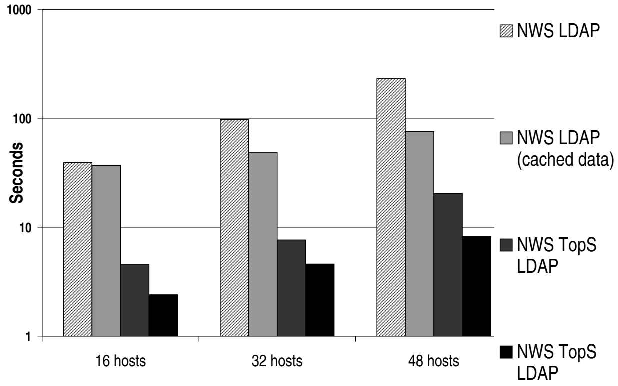

Figure 10. LDAP queries to remote infrastructure (log scale)

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 8.

Similarly, Figure 9 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 10 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.

In the full paper, should this extended abstract be accepted, we will provide a more comprehensive set of empirical results from different Grid settings.

6 Conclusion

We have shown that our object model is flexible enough to afford new functionality, and have demonstrated how realistic and useful queries can be made against a GIS (independent of the protocol used). This methodology should be of broad applicability to the Grid community, as it demonstrates that GIS systems can be made to perform appropriately.

Finally, we have demonstrated the efficacy of a simple topology system that is useful for scheduling in Grid environments.

References

[1] 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. *Journal of High Performance Computing Applications*, 15(4), Winter 2001.

[2] 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`.

[3] 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.

[4] 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.

[5] A. Petitet, S. Blackford, J. Dongarra, B. Ellis, G. Fagg, K. Roche, and S. Vadhiyar. Numerical libraries and the grid. In *(to appear) Proc. of SC01*, November 2001.

[6] 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.

[7] 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, Feburary 2001. http://www.gridforum.org.

[8] 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.

[9] 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.

[10] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 1999.