Using Effective Network Views to Promote Distributed Application Performance

Gary Shao* and Fran Berman†
Department of Computer Science
and Engineering, 0114
University of California, San Diego
San Diego, CA 92093-0114

Rich Wolski†
Department of Computer Science
University of Tennessee
107 Ayres Hall
Knoxville, TN 37996-1301

Abstract *In this paper, we show that distributed parallel application performance in clusters of workstations benefits from knowledge about network topology in cases where LAN resources can become potential performance bottlenecks. These performance bottlenecks often appear in common networking technologies that employ highly shared resources, such as ethernets and token-rings. A method to discover network configuration as it relates to application performance, called Effective Network Views (ENV), is presented, and experimental results with an example application are given to show the value of including ENV information in the scheduling process when good performance is required.*

Keywords: effective network views, topology discovery, application-level scheduling

1 Introduction

All programs achieve performance by exploiting the architectural characteristics of the platforms on which they will run. This is especially true of distributed parallel programs, which are typically developed to leverage the special performance characteristics of distributed processors and their associated interconnection networks. Parallel applications targeted to MPPs with specialized communication structures have often depended on having accurate static information about the underlying communication architecture of these platforms to achieve good performance. In the clusters of workstations environment, information about the configuration and deliverable performance of the interconnection network is seldom accurate or constant. The availability, capability, and arrangement of network resources that can be effectively observed by an application varies with the changing needs and demands of many different users and administrators.

Network capacities are most often characterized by simple measurable values, such as bandwidth and latency. *Bandwidth* is a measure of the quantity of data that can be carried by a network in a fixed period of time, while *latency* is a measure of the time needed for a single data unit to traverse a path in the network. These capacities can often easily be obtained by measurement, even at an ordinary user level [11]. High performance application schedulers often use these measures to promote application performance in distributed and clustered environments [3]. Most useful to application schedulers are *effective* measures of bandwidth and latency, i.e. the network performance in terms of bandwidth and latency that is actually delivered to an application [4].

Networks can also be described by their *topology*, a representation of interconnection relationships between resources. Application schedulers that include knowledge of the net-

*Supported by NASA GSRP grant #NGTS-50131.
†Supported in part by NSF grant #ASC-9701333, DARPA/ITO contract #N66001-97-C-8531, NPACI award #ASC9619020, DoD Modernisation contract #9720733-00.

work topology in their decisions can avoid making some poor scheduling choices, such as allocating frequently communicating tasks to processors linked by slow or heavily loaded networks. Note that it is not the physical topology that the application scheduler requires, but rather the topology corresponding to the performance that the application *experiences* (i.e. observes). We term the network organization that is consistent with observable network behavior the *effective topology* of the network for the application.

In this paper, we show that beneficial effective topology information about networks can be derived from user-level observations and measurements; and that this new information can be used effectively by schedulers to improve application performance. Section 2 describes how user-level observations of network performance can be used to create an effective profile of network configuration, Section 3 illustrates results of this network profiling procedure on several test LAN systems, and Section 4 shows how scheduling techniques can use the network profile data to detect potential resource contention problems and apply appropriate scheduling solutions to solve them. Section 5 gives a short list of related work, and we summarize this paper in Section 6.

2 Creating Effective Network Views

Network configuration can play an important role in the performance of parallel applications running on clusters of workstations. In some environments, relevant information about the interconnection configuration in real networks may be hard to obtain. Sometimes, only the person who installs or services the equipment really knows or cares about exactly how things are hooked up. Common network tools, like *traceroute* [8] and *pathchar* [7], are able to report network routing down to the switch level and produce estimates of router congestion, but once past the switch they are unable to distinguish whether two hosts are on a common subnet sharing a single hub, or on separate subnets. Moreover, even if complete maps of physical connections were available for all components in a network, *this topological information is only useful for scheduling if its effects on application performance can be quantified.*

If the application's performance is to be optimized, what a scheduler needs is a representation of the network *as the application will see it*, i.e. in terms of the *quantifiable* performance that it is likely to deliver to the user's application. The physical network is not as important to the performance-oriented application scheduler as the effective network; configurations of network resources should be represented in terms of their effective properties. In this paper, we describe a prototype mechanism for determining what we call *effective network views* (ENV) which provide a representation of the logical network as perceived by the application. ENV information is created by grouping together network components that have significant interaction and similarities in response to events on the network. The grouping is effective rather than physical because only the measurable performance responses that are observed determine the grouping results. Using an ENV, an application scheduler can place in the same group those hosts that will experience similar performance limitations in response to application-driven loading of shared network resources.

We have developed a procedure for discovering ENV information using a collection of straightforward network performance tests. Because our ultimate goal is to develop scheduling methods and approaches that can be used by any ordinary user, we have chosen to develop techniques which do not require the presence of special system tools or access privileges. We use this approach to make our approach widely portable across a range of clustered and meta-computing environments.

The procedure for creating ENV groupings involves a series of steps designed to identify clusters of machines that will likely have significant sharing of a network resource, such as being on the same physical ethernet subnet.

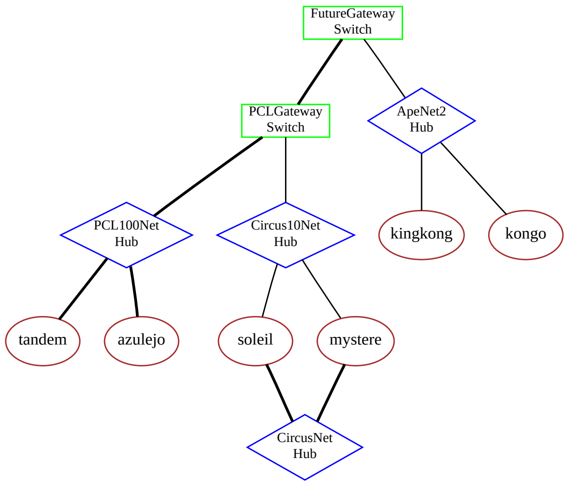


Figure 1: Physical Topology. Shows interconnections of resources on the network, but does not include performance data.

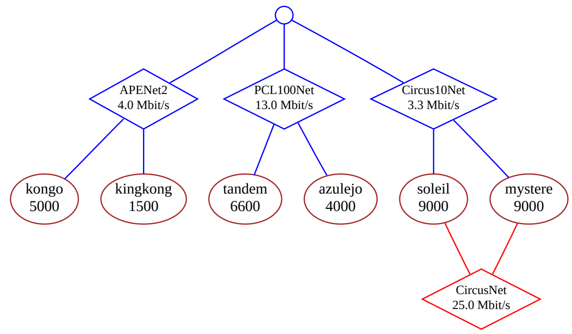


Figure 2: Effective Topology. The figure shows effective configuration of resources, which includes performance information. Values for network bandwidth are taken from ENV measurements, while values for machine speeds are based on benchmark data.

We detect the sharing by looking for similar responses to network events from machines during an initial test phase.

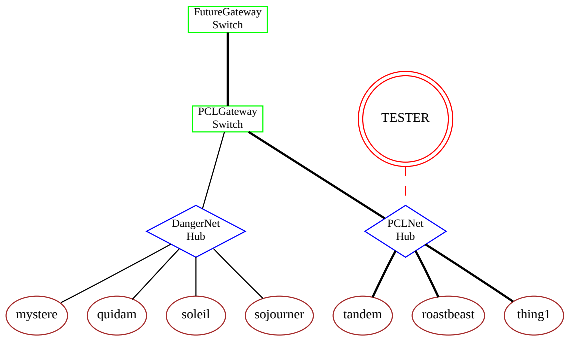
In our procedure, almost all testing activity originates from a single test machine. The resulting ENV data is relative to this test machine. After roughly filtering a set of machines by straightforward criteria like domain name and subnet addressing boundaries, we apply our first active test. The test machine conducts a bandwidth test between itself and each of the other machines being tested, keeping track of the results. Using the results of the first bandwidth test, machines with similar bandwidth values are clustered together.

The second active test chooses pairs of machines within the same candidate cluster, and runs simultaneous bandwidth tests between them and the test machine. If there is significant interference[1] in the 2-host bandwidth test compared to the 1-host bandwidth test, we leave the machines in the same cluster. If we do not detect significant interference, we separate the machines into different clusters. At the end of the clustering procedure, we will have determined which machines can be considered effectively sharing the same subnet, and we will have annotated the topology information with estimates of the subnet performance relative to the test machine view.

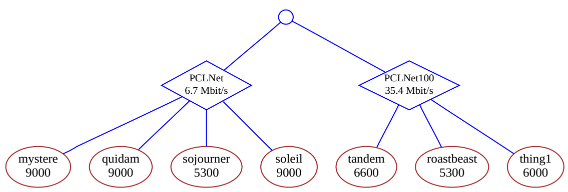
Optionally, a third active test can be run between machines within each cluster to find private network resources that cause different communication behavior with machines inside and outside the cluster. Testing identifies differences between internal and external communication bandwidth, and labels detected private networks with their observed performance values. Figure 1 shows the physical topology of a set of workstations located in the Computer Science and Engineering (CSE) Department at UCSD. Figure 2 shows a sample ENV graph of that same set of workstations that was generated by the ENV tool. In the figure, the test machine was located at the San Diego Supercomputer Center (SDSC) which is separated from the CSE Department by a campus-wide ATM network.

Our ENV representations differ from physical mappings of network resources in one important way: *each view of a network reflects the vantage point of a single process*, i.e. the one that ran the performance tests and collected the results. As an example, an initiating machine whose connection to outside networks is through a 10 Mbit/sec ethernet port could not accurately test the bandwidth of a

[1] currently defined as a 30% decline in measured bandwidth

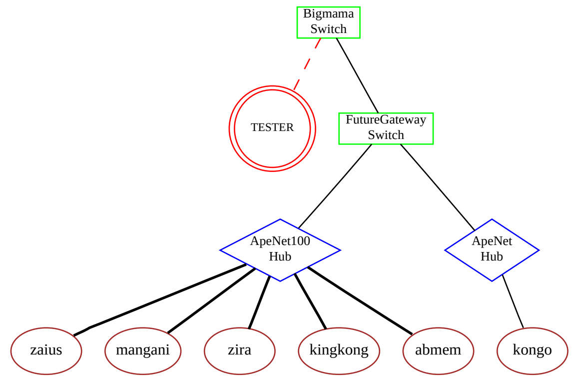


(a) Physical topology

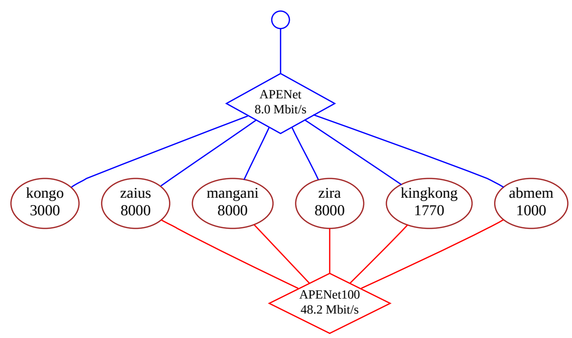


(b) Effective network view

Figure 3: Effective network view of PCL resources from a machine within the PCL network.



(a) Physical topology



(b) Effective network view

Figure 4: Effective network view of APE lab resources from a machine outside the APE lab network.

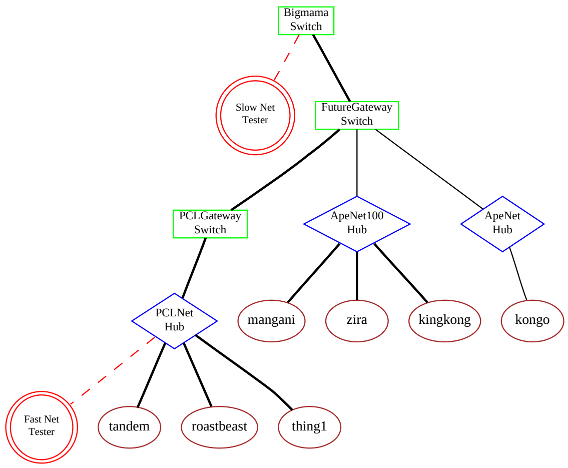
100 Mbit/sec ethernet segment, since it would only see its own 10 Mbit/sec limit on network speed during tests. This notion of perspective is consistent with the application's view: if the application is initiated from a host with the 10 Mbit/sec network speed for communication, it would experience an effective rate of at most 10 Mbit/sec to all other computational resources, no matter how fast the other network links might be. We also concern ourselves here only with bandwidth measurements, believing that latency issues are much less important in the LAN scenarios discussed in this paper.

3 ENV Examples

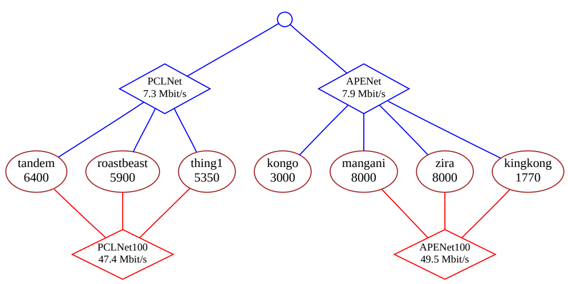
In this section we give some examples of LAN networks, and the resulting ENV profiles that were generated for these networks by a prototype tool implementing the ENV discovery procedure.

The first example (Figure 3) shows a set of machines in the Parallel Computing Lab (PCL) at UCSD. The physical network in the lab is a mixture of 10 and 100 Mbit/sec ethernet subnets, all connected to a central switch. Figure 3 shows an ENV graph generated for these machines, using as a test machine a workstation located within the network on a 100 Mbit/sec ethernet link.

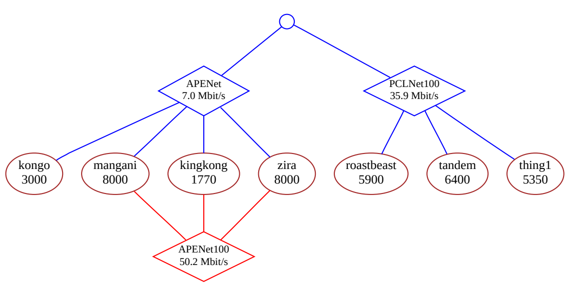
The second example (Figure 4) shows a set of machines in the instructional Advanced Programming Environment (APE) lab at UCSD. This lab also has a mixture of 10 and 100 Mbit/sec ethernet subnets, each going to a different switch from the PCL machines shown above. Figure 4 shows an ENV graph generated for these machines using as a test machine



(a) Physical topology



(b) Effective network view from machine on slow network



(c) Effective network view from machine on fast network

Figure 5: Resulting effective network view can depend on network speed and connectivity of test machine.

a workstation located outside the network being tested. The test machine in this example was located at SDSC, and was connected to UCSD through a series of 10 Mbit/sec ethernet and ATM links.

Number of blocks	1600
Work block transfer size (bytes)	8
Output block transfer size (bytes)	5028
Slave exec. time[a] (usecs/block)	52400
Master exec. time[a] (usecs/block)	3000

[a]relative to a processor with speed = 1000

Table 1: Runtime parameters for Mandelbrot master/slave application.

Figure 5 shows two ENV graphs generated for a set of machines from both the PCL and APE labs; one using the previously described external test machine at SDSC, and another using a machine located on a 100 Mbit/sec link within the PCL lab to perform the tests. These graphs help illustrate the difference between an effective view and a physical mapping. From the point of view of the testing machine in Figure 5(b), communication with all hosts would be through slow ethernet hubs, leading to potential communication bottlenecks. But from the point of view of the testing machine in Figure 5(c), network resources are uncovered that can be reached through fast ethernet hubs, avoiding the same bottleneck problems.

4 Using ENV Results

Once network configuration data has been collected, we wish to use the information to generate schedules which promote application performance. The network discovery process outlined in the previous section produces a graphical representation of a network. This graph is useful for performing several valuable types of analysis. One approach is to use the graph for the problem of *resource selection*; deciding which hosts to use in a computation, and calculating the best placement of processes on selected hosts.

As an example, Figure 6 shows an ENV graph which has been annotated with values

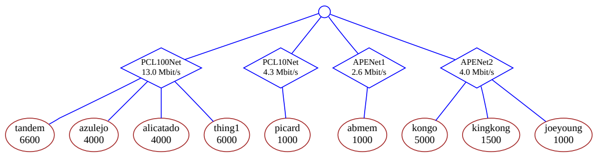


Figure 6: Effective network view for a set of workstations we want to schedule a master/slave application for.

for subnet bandwidth (in Kbits/sec) and relative processor speed (derived from application benchmarking). Onto this configuration we wish to schedule a master/slave Mandelbrot application that has known characteristics as shown in Table 1. This simple application starts off with the master process sending fixed-size work blocks to each slave in the system. Each slave receives work blocks, processes them in the order they are received, and sends fixed-size result blocks back to the master as soon as they are ready. When a result block reaches the master, a new work block is sent to the slave that generated the last result block. The process continues until all available work has been sent. Selection of which workstation to use as the master process has a significant impact on application performance. Figure 7 shows the effect on execution time of choosing each of the workstations in our example graph as the master machine, while using all other machines for slave computation.

A selection algorithm was run on the ENV graph to choose the processor on which to place the master process. The selection was made by identifying which processor would support the highest work throughput (based on processor speed and network bandwidth) from available slave processors, without exceeding either the local subnet capacities of the individual ethernet links, or the processing capacity of the candidate master processor.

Network throughput capacities $NTP(s)$ for each ethernet subnet s were calculated using ENV values for subnet bandwidth $BW(s)$, and a specified value for output blocksize B_{size} with the relation:

$$NTP(s) = BW(s)/B_{size}. \qquad (1)$$

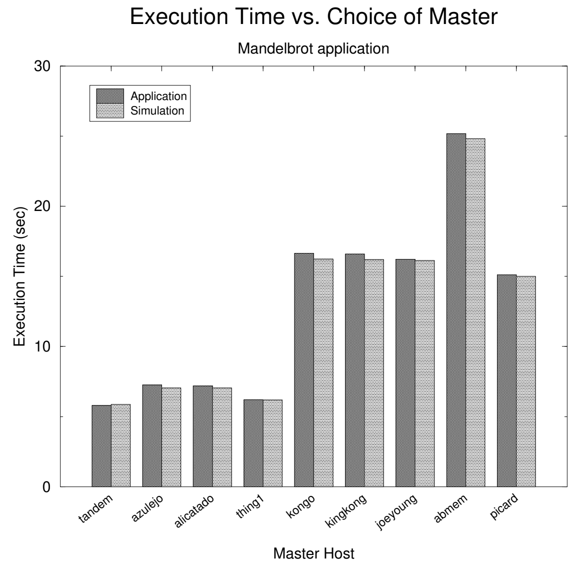


Figure 7: Effect on Mandelbrot execution time caused by selecting different machine as master.

Slave throughput capacities $STP(p)$ for each potential slave processor p were calculated using ENV values for benchmarked processor speed $S(p)$, predicted values for slave availability $A(p)$, and measured values for the time T_{block} a processor with a specified speed would require to process a single block with the relation:

$$STP(p) = S(p) * A(p)/T_{block}. \qquad (2)$$

Our selection algorithm uses these throughput values to form constraints on application performance that result from choosing a particular host for the master process. Details for this algorithm will be presented in a later paper.

Our selection algorithm picked *tandem* as the best choice for master in this particular network configuration and application, a result confirmed by the results shown in Figure 7. What is striking is that it makes considerable difference which machine is assigned the master process. In particular, a poor choice of placement for the master process results in execution times that are greater by a factor of four than a good placement in the experiments shown here.

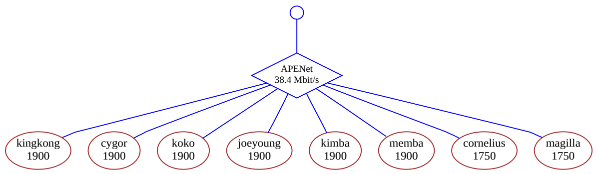


Figure 8: Effective network view for a set of workstations from which we want to select the best number of slaves.

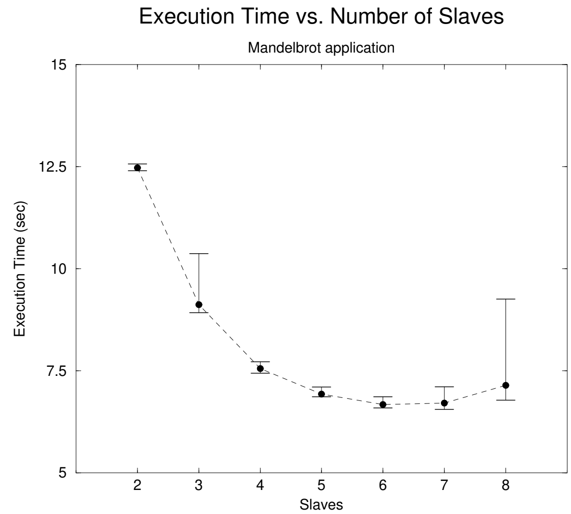


Figure 9: Effect on Mandelbrot execution time caused by selecting different numbers of slaves.

The ENV graph data is also useful in selecting the number and placement of slave processes to give the best application performance. Because of contention for network resources, adding more slaves to a computation than can be supported may actually result in reduced application performance. We can use the ENV graph data in an algorithm that identifies how many processors from each subnet can be added to a computation before the capacity of that subnet, or the capacity of the subnet the master is on, is exceeded.

As a simple example of this slave selection algorithm, the network configuration represented in the ENV graph of Figure 8 contains eight processors, on the same subnet, that we wish to consider as slaves for the Mandelbrot application described earlier. Calculating NTP for this network configuration gives a value of 954 blocks/second. Calculating STP for each of the nearly identical processors gives a value of 172 blocks/second. Assuming each processor contributes its maximum work throughput, we can expect the capacity of the network to be reached when using approximately $954/172 = 5.5$ slave processors. Figure 9 shows the result of varying the number of slaves while running the Mandelbrot application on the processors of Figure 8. The results, showing no average gain in performance over 25 trials for more than six slaves, agrees quite well with our prediction based on ENV data.

5 Related Work

It is widely recognized that achieving good performance of parallel and distributed applications in a clustered environment benefits from knowledge of the underlying network state. Crandall and Quinn [6], Basney and Livny [2], the AppLeS Project [4] and others use network information to improve the execution of applications in multi-user environments. Bolliger and Gross [5] present a framework to support network-aware applications.

Perhaps most closely related to the work presented here is that of Weissman and Zhao [10] and Subhlok, Lieu and Lowekamp [9] who each discuss systems to provide resource information (including network characteristics like bandwidth, latency and topology) for local-area environments. Our work differs from these last two examples by demonstrating a method for determining effective network configurations *using only user-level observations and measurements*, and by not relying on external resource management services.

Finally, the PMH model [1] provides a parameterizable generic programming model, based on a hierarchical view of memory, that is intended to assist in the development of performance-efficient programs for a variety of architectural platforms. The ENV approach shares similar goals, but applied to the problem of scheduling, in representing observable network properties in a structural format that can be related to application performance.

6 Conclusion

We have described in this paper a method for generating effective network views using only user-level testing mechanisms, presented examples for generating effective views for LAN-based systems, and have given an example of how this can be used in distributed application scheduling. We are currently working on extending the concepts to cover the wide-array network (WAN) case as well as the local-area setting described here. In addition, we are developing the ENV discovery tool as a fundamental component to improve the determination of efficient schedules in AppLeS distributed application schedulers.

Acknowledgements

We would like to thank the members of the AppLeS Project and our colleagues at SDSC, NCSA and the University of Tennessee for substantive discussions and assistance with the experiments.

References

[1] Alpern, B., Carter, L., and Ferrante, J. Modeling parallel computers as memory hierarchies. In *Proceedings of the 1993 Conference on Programming Models for Massively Parallel Computers* (Sept. 1993).

[2] Basney, J., and Livny, M. Improving goodput by co-scheduling cpu and network capacity. submitted to International Journal of Supercomputer Applications and High-Performance Computing. *http://www.cs.wisc.edu/condor/publications.html.*

[3] Berman, F. High-performance schedulers. In *The Grid: Blueprint for a New Computing Infrastructure*, I. Foster and C. Kesselman, Eds. Morgan Kaufmann Publishers, July 1998, ch. 12.

[4] Berman, F., and Wolski, R. Scheduling from the perspective of the application. In *Proceedings of the Fifth IEEE Symposium on High Performance Distributed Computing* (Aug. 1996), pp. 100–111.

[5] Bolliger, J., and Gross, T. A framework-based approach to the development of network-aware applications. *IEEE Transactions on Software Engineering 24*, 5 (1998), 376–390.

[6] Crandall, P. E., and Quinn, M. J. A partitioning advisory system for networked data-parallel processing. *Concurrency: Practice and Experience 7*, 5 (1995), 479–495.

[7] Jacobson, V. Pathchar — a tool to infer characteristics of internet paths. Presented at the Mathematical Sciences Research Institute (MSRI), April 1997. Slides available from ftp://ftp.ee.lbl.gov/pathchar.

[8] Kay, J. *Internet Measurement Tool Survey. http://www.caida.org/Tools/taxonomy.html.*

[9] Subhlok, J., Lieu, P., and Lowekamp, B. Automatic node selection for high performance applications on networks. To appear in *Proceedings of the Seventh ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming* (May 1999).

[10] Weissman, J. B., and Zhao, X. Scheduling parallel applications in distributed networks. *Journal of Cluster Computing 1*, 1 (1998).

[11] Wolski, R. Dynamically forecasting network performance using the network weather service. In *Proceedings of the 6th High-Performance Distributed Computing Conference* (Aug. 1997), pp. 316–325.