Scalable Computing: Practice and Experience
Volume 6, Number 3, pp. 85–94. http://www.scpe.org
ISSN 1895-1767

NETWORK SCHEDULING FOR COMPUTATIONAL GRID ENVIRONMENTS

MARTIN SWANY* AND RICH WOLSKI†

Abstract.
The problem of data movement is central to distributed computing paradigms like the Grid. While often overlooked, the time to stage data and binaries can be a significant contributor to the wall-clock program execution time in current Grid environments.
This paper describes a simple scheduler for network data movement in Grid systems that can adaptively determine data distribution schedules at runtime on the basis of Network Weather Service (NWS) performance predictions. These schedules take the form of "spanning trees." The distribution mechanism is an enhancement to the Logistical Session Layer (LSL), a system for optimizing data transfers using "logistics."

Key words.
Grid computing, data logistics, data staging

1. Introduction. As Computational Grid environments proliferate, the community must constantly evolve the way in which computing systems are used. Distributed computing on the Grid has enabled new ways of harnessing computing resources and yet, has exposed its own set of challenges. One such problem is that of data movement. Applications that are drawn to the Grid because of large resource requirements frequently consume or generate large amounts of data. The problems of data locality and data movement are becoming more prominent and critical to the performance and deployability of Grid systems. Further, due to the dynamism inherent in Grid environments, it is clear that mechanisms for data staging must be adaptive like the computations themselves.

AppLeS [8] demonstrated the beginning of a new way of thinking about programming the Grid–scheduling from the perspective of the application. In this spirit, we propose to approach the problem of adaptively scheduling buffers in the network with proactive support from the application. This paper examines simple optimizations that we can facilitate by thinking of Grid resources in terms of cooperating elements in a storage and computing "overlay" network. By enabling this type of functionality, using techniques such as the Logistical Session Layer (LSL) [34] or the Internet Backplane Protocol (IBP) [28], the breadth of the services offered by a Grid is improved.

The goal of this work is to investigate scheduling and routing techniques focused on optimizing data movement in Grid environments. In order to investigate such scheduling we will draw on previous work as follows. The Logistical Session Layer (LSL) [34] provides the basic platform for cooperative data forwarding that responds to requests from the scheduler. The Network Weather Service (NWS) [43] provides us with network performance monitoring and forecasting capabilities. Finally, the NWSlapd [37], the caching and delivery subsystem of the NWS, caches network performance forecasts and aggregates them into a form suitable for consumption by the scheduler.

There has been a tremendous amount of work in this community to optimize collective operations for parallel computing [4, 27, 24, 5, 18, 39, 20, 40]. Certainly, these approaches are all related at some fundamental level (and discussed somewhat in Section 6). However, our approach is focused on pre-runtime data distribution (or staging) rather than collective operations as such. Initial data distribution is an important component of actual Grid deployment. This fact is often obscured by pre-staged binaries or locally-generated random input data, but for Grid systems to realize their potential, these issues must be addressed.

Our approach to this problem is unique in a number of key ways:

- It treats Grid resources as a graph with edge values derived from current network performance forecasts
- It adaptively builds distribution trees for arbitrary topologies by creating a schedule based on the Minimum Spanning Tree (MST) over that graph
- Cooperative forwarding among peers is accomplished with the *Logistical Session Layer* (LSL), which uses cascaded TCP connections.

Grid environments are extremely dynamic. Network performance depends on ambient load. To best adapt our execution at runtime, forecasts based on current performance information are necessary. Distribution trees based on this information will often vary wildly in shape. We need an extremely general tree construction mechanism to accommodate the diversity of Grid systems. Finally, as we use LSL for our distribution platform,

*Department of Computer and Information Sciences, University of Delaware, Newark, DE, 19716 (swany@cis.udel.edu).
†Department of Computer Science, University of California, Santa Barbara, CA, 93106 (rich@cs.ucsb.edu).

we get the benefits of performance-enhancing buffering in the network, and the reliability and deployability of TCP.

In this paper we will first describe the assumptions in our approach to scheduling. Next, we will describe a simple scheduling approach, based on spanning tree, that is general enough to address our needs. Finally, we describe the enhancements to LSL necessary to implement a schedulable distribution mechanism and evaluate the performance improvements that even simple scheduling can afford in this space.

2. Problem. The general problem that this work addresses is that of the "logistics" of data movement in Computational Grid environments. In fact, the logistics of data movement are the main reason why computing "power" is not a fungible resource like electrical power. Users need computations to be performed on specific bits of data, whereas electricity can be consumed regardless of the location or means of its generation. The problems of data locality and movement are universal and are a critical consideration in Grid systems.

There has been much recent work considering cooperative data sharing between networked peers [30, 33, 6, 28, 22]. These cooperative approaches have had impact in both the parallel processing and network computing domains. In this spirit, we consider an environment in which Grid resources are enabled to utilize and provide cooperation of this sort. Our goal is to consider scheduling these resources and examine potential performance optimizations that might emerge. This work builds on the ideas of "Logistical" [34, 6, 28], "overlay" [3, 38, 17] and "peer-to-peer" [30, 33, 22, 44] networking to treat the problems of communication in Grid systems in a novel manner.

The GrADS [7] project is a large, multi-institution project whose goal is to investigate comprehensive software environments for developing Grid applications. As such, the GrADS environment is focused on program development and compilation as well as runtime Grid support. 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 runtime services provided by the Grid fabric and discovers the "state" of the Grid at that time. The S/SN uses this state information to make decisions about program configuration and scheduling. In particular, the system requires current short-term forecasts of resource performance levels so that it can make proactive scheduling decisions. The NWS generates such forecasts automatically, but to be useful, they have to be delivered to the S/SN (through the Globus [13] infrastructure) quickly and reliably.

Considering the problem of initial data distribution, our assumptions can be captured by the following scenario. Let us imagine that a user is launching a program in a Grid environment such as the GrADS [7] project's testbed. In the GrADS architecture, the Configurable Object Program, or COP, is distributed by the Application Manager in the first phases of execution. This is not, of course, unique to GrADS. In many Grid paradigms a user has a set of program executables that need to be distributed to the resources before execution can begin.

In other Grid usage models, end-users utilize resources through previously existing software infrastructure. This software exports services through application interfaces using remote procedure calls, or RPC. NetSolve [11] is an example of such a system. The problem that these systems face is similar to the program distribution problem in that some amount of data must often be sent from the user to Grid resources prior to the beginning of any meaningful execution. This problem is strongly related in that it concerns initial data distribution and thus, it can be modeled similarly.

These problems are equivalent to some degree in that either prior to runtime or during an initial phase of runtime, some data has to be sent to the each computational node before any real application progress can be made. Often, we choose to abstract this problem away with file-sharing techniques. In fact, network file systems (e.g. NFS) can be used within a single site so that we only need to transfer once to nodes that share files this way, but there are many cases where systems do not share files in this fashion. Further, NFS can suffer from poor performance and since data (programs or user data) is to be moved over the network, we prefer to deal with the associated overhead explicitly. Certainly, there are many situations and scenarios that differ in simple ways from this basic model, but this captures our assumptions and, in fact, models real Grid systems quite well.

2.1. Problem Modeling. Consider the simple depiction of these data transfers in Figure 2.1. In these graphs, the value along the edge denotes some cost. In this case it is the time to transfer some amount of data. Figure 2.2 obviously demonstrates a distribution pattern (or tree) with a lower overall cost.

Further, in Grid environments, resources are often located in groups or clusters, so the potential performance improvement from such optimizations becomes more obvious. Figure 2.3 illustrates the fact that in many real cases, a hierarchical distribution scheme can greatly reduce the overall cost of the paths through the network.

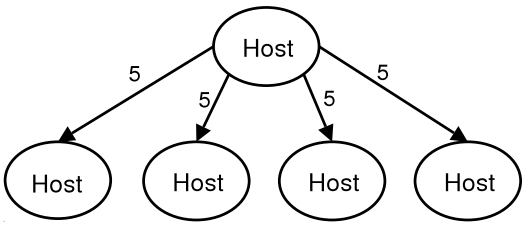

FIG. 2.1. *Cost tree for default distribution strategy*

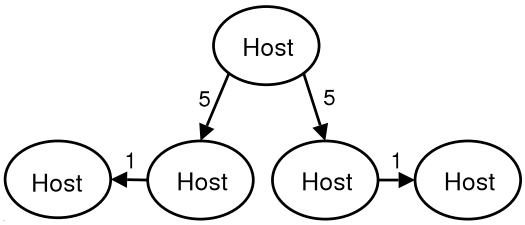

FIG. 2.2. *Less costly distribution tree*

This modeling approach allows us to think about the problem of data distribution as a graph and offers obvious chances for optimization.

3. Scheduling Algorithms. The crux of this work is the observation that by treating the resources of the Grid as a "network", we can schedule the cooperation of these resources in the formation of a single-source, data distribution tree. This schedule can be computed dynamically, based on current performance information. A distribution tree must be able to direct the data to each node, or "span" the tree.

Consider a directed graph G with vertices and edges: $G = (V, E)$. Each edge has a weight or cost c_{ij} for each $(i, j) \in E$. A spanning tree (T) is a graph with $T \subseteq G$ such that $\forall V$ there is a $(u, v) \in T$ that is incident on it (i.e., T spans the set V).

The Minimum Spanning Tree $MST(G) = T$ where $\sum_{(u,v)\in T} c(u, v)$ has the minimum cost of all spanning trees.

A traditional, and provably optimal, approach to the solution of MST is known as Prim's algorithm [29]. This algorithm uses a greedy approach in the construction of the solution tree. Briefly, the algorithm proceeds as follows.

To find the MST (T), we create an empty tree T and move the starting node of the tree (v_{start}) from V to T:

$$v_{start} \in T \mid T \cap G = \emptyset \tag{3.1}$$

Then, we iterate while $|V| > 0$. At each step we examine edges in the "cut" (edges that begin in T and end in V) and select the minimum cost edge:

$$\min(e) \in E' \mid e(u, v)\, u \in T \, and \, v \in V \tag{3.2}$$

Node v is then moved to T and we examine the newly added node and edge to see if its addition has offered a better path to nodes already in T.

While the spanning tree problem is at the heart of this approach to scheduling, there are additional factors that must be considered in our model. In the previous section, we considered extremely simple graphs. Obviously for Internet hosts, the time to transmit data to a number of hosts is not linear with the number of hosts. Multiple outgoing edges interfere with one another – they are not independent. In terms of the network, the more streams there are sharing the resource of outgoing network capacity, the less each stream gets. This could complicate the model significantly. In fact this problem is very similar to what is known as the "weighted graph minimum-energy broadcast problem", which has been shown to be NP-hard [41]. Further work in the same problem

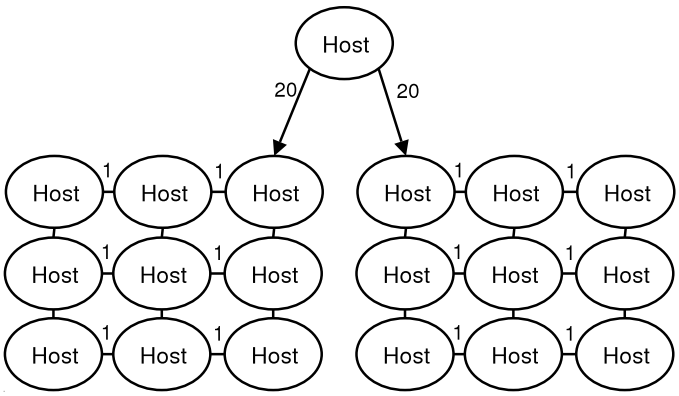

FIG. 2.3. *A distribution tree for clusters*

space [12] shows that the problem remains NP-hard even when realistic bounds are placed on transmission levels (reducing them to a small fixed set), but gives hope for polynomial-time solutions if a solution exists.

Another potential combinatorial problem arises in our situation as well. The "Steiner Network" is different from the MST problem in that only a subset S of G must be spanned. This problem has been shown to be NP-hard [19]. This problem is the heart of the problem of "minimum spanners" [10] again demonstrated to be NP-complete. However, we note that since the set with which we are concerned is not a subset of G that we avoid the difficulties associated with these problems.

These previous results treat their realm of discourse to be in "metric space," meaning that the triangle inequality holds. Internets are not, in general, in metric space. This makes the problem more tractable initially, but ultimately complicates the model. In particular, rather than power levels, our spanning-tree problem has the above described constraint that we can refer to as "lateral inhibition." The more edges (streams) that are incident on a node, the less well any of them perform. In the extreme, the interference between streams is unique for every stream configuration. This combinatorial space implies that the optimal solution for such a problem is NP-hard. However, we note that this approach is not necessarily concerned with an optimal solution, rather we wish to empirically determine the efficacy of this general class of solution.

The MST problem is known to be related to many problems in distributed data movement. While we do not deal with it directly in this work, the minimum cost path and all-pairs minimax problems [2] provide a basis for multi-hop forwarding of the sort proposed by LSL [34] and IBP [28]. Parallel streams with diverse paths allow us to couch routing in terms of maximum flow algorithms. However, utilizing parallel streams between identical locations, with default paths, only serves to increase the value of a single arc. This would certainly increase the observed bandwidth, but our treatment of the single-stream case still holds without loss of generality.

4. System Architecture. To deploy and test this scheduler on a Grid system, we rely on various components of Grid software. Specifically, this software depends on the Network Weather Service, the NWS's caching LDAP delivery system and the Logistical Session Layer.

4.1. Network Weather Service. The Network Weather Service [43, 42] is a system developed to provide performance monitoring and online performance prediction to Grid schedulers such as ours. Grid environments are extremely dynamic and in order to manage this dynamism, a scheduler must have near-term performance predictions upon which to base runtime decisions. The NWS measures, among other things, TCP bandwidth and latency between hosts in a scalable and unintrusive manner. By applying various non-parametric statistical techniques on the timeseries produced by these ongoing measurements, the NWS is able to produce forecasts that greatly improve prediction over naive techniques. Further, these measurements can be combined with past instrumentation data to produce accurate estimates of bandwidth [36] or transfer time.

An additional component of the NWS, called the NWSlapd [37, 35], provides necessary functionality as well. First, this system caches performance predictions near querying entities making it possible to scale the performance information infrastructure and provide ubiquitous forecasts to network-aware schedulers. This part of the system also assembles measurement information into a network "view" that can be easily and quickly queried. Note, however, that the NWS does not actually initiate measurements between every pair of hosts (n^2

tests.) Rather, the NWSlapd interprets the hierarchy of measurements that the NWS does take and fills in a complete matrix of forecasts (as described in [35].)

The complete matrix of forecasts provides us with the node-node adjacency matrix representation of our network. The adjacency matrix is populated by the observed bandwidth (and/or latency) between host i and host j in the (i, j)th element. Note that the graph that this matrix represents is fully-connected as every host on the Internet can reach every other host with some bandwidth.[1] This provides the initial graph G upon which our scheduler operates.

4.2. Scheduler Implementation. Our initial scheduling approach is simply to describe a spanning tree for the nodes in our resource pool. To do this, we simply use Prim's algorithm as described in Section 3. In order to produce a minimum spanning tree, we need a metric where a smaller value is "better". Since we are operating with bandwidth forecasts, we convert the bandwidth estimates "transfer time" estimates by considering $1/bandwidth$ as the "value" of an edge.

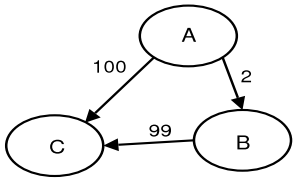

FIG. 4.1. *Simple Illustration of Tree Depth*

One simple technique that we have implemented allows us to minimize the depth of the spanning tree. Our goal is to minimize the number of hops that a stream must pass through as each hop adds some amount of overhead. Consider the graph in Figure 4.1. Strictly speaking, the minimum spanning should include the arc $A \rightarrow B$, and that from $B \rightarrow C$. However, it reduces the depth by a level and increases the overall cost of the tree to span via the arc from $A \rightarrow C$.

This has an effect in practice. Due to small variations in measurements through time, machines with functionally similar connectivity have slightly different forecasts. To keep the trees more simple, we would like to consider measurements within some ϵ of one another as the same. A perfect choice for this value is the historical forecasting error from the NWS.

The scheduler performs as expected. When presented with the results of a performance query from NWS containing information about the GrADS testbed [14], the system was clearly able to discern separate clusters at the University of Tennessee and University of Illinois and suggest a distribution tree taking that into account. Figure 4.2 depicts spanning tree produced by the scheduler, and this graph is generated from that output using GraphViz [15], a graph plotter. The initial set of results (in Section 5) utilize this host pool and similar distribution schedules.

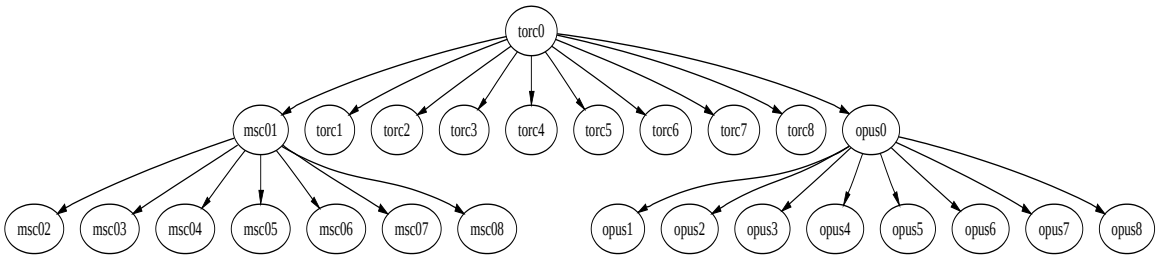

FIG. 4.2. *Spanning Tree*

Note that Figure 4.2 is created automatically. Other than guessing based on the names of the hosts (not on the domain name), there is no way to discern these clusters at the network level. In some cases, only empirical performance measurements show these relationships, as shown previously by Effective Network Views

[1] With the exception of hosts behind firewalls. While our techniques are even more natural in those cases, a discussion of that application beyond the scope of this work.

(ENV) [32]. It is interesting to note that we have recovered the structure of the network with our scheduler technique alone.

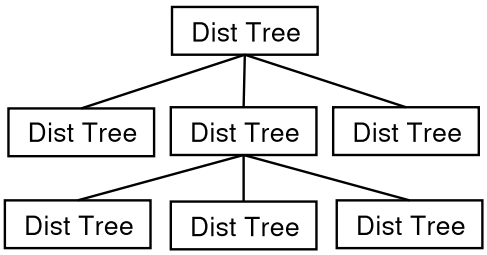

FIG. 4.3. *Distribution records in a tree*

4.3. Logistical Session Layer Data Distribution. The scheduler produces a distribution tree which is given to the Logistical Session Layer [34] (LSL) to control the data distribution. LSL is a system for cooperative forwarding and buffering of network traffic that has been shown to greatly increase end-to-end network performance. LSL utilizes TCP, so questions of "friendliness" are not an issue and data integrity guarantees are those of TCP. [2] However, LSL endeavors to allow TCP to perform better by keeping the round-trip time on any sublink to a minimum. This use of TCP also facilitates incremental deployability, yet takes advantage of improving transport-layer performance.

For this particular experiment, we have implemented a new message option in the LSL stack. Each option defines a distribution tree including information about the children of that node. The hierarchy of distribution headers is recursively encoded and decoded so that only the relevant portions of the subtree are transmitted to downstream neighbors until ultimately, the leaf nodes get a distribution tree with a single entry. Figure 4.3 illustrates this.

The acknowledgment of data receipt at the ultimate destination is implicit with the closing of the TCP socket. At each LSL node, necessary data is sent out all outgoing sockets and the sending side of each of those sockets is closed. Each daemon then waits for each downstream neighbor to close its socket, signaling that all destinations have received the data. At the leaf nodes, the sockets are closed normally once all data is written to the filesystem. We note that direct notification from destination to source may be more desirable in many cases and such a modification is straightforward.

Internally, the implementation is not aggressively optimized, and further performance improvements are certainly possible. There is also no security model at this time. Our technique could easily work over SSH-encrypted and authenticated tunnels and this is one implementation possibility that we are investigating.

5. Results. To test the efficacy of our system, we have deployed it across the GrADS testbed [14]. This set of Grid resources ranges from 50 to 100 nodes across the U.S. located primarily at the University of California, San Diego, the University of Illinois, Champaign-Urbana, and the University of Tennessee, Knoxville. The sites are connected by Internet2's Abilene [1] backbone and enjoy relatively high-speed connectivity.

To evaluate the difference between direct distribution (the direct approach) and our scheduler in as fair a manner as possible, we have modeled the direct distribution within our software infrastructure. That is, the direct distribution version is simply a flat tree. This allows for overlapping communication among the streams and is not terribly inefficient. At any rate, the data movement is not serialized among the nodes as it often is in daily use. [3]

Two sets of tests were run. The first set contains 18 nodes located at two sites. The second set contains 52 nodes in 6 clusters at 3 sites. In all cases the source of the data was located at the University of California, Santa Barbara. Again, this models situations that are demonstrably realistic.

Figure 5.1 shows the distribution time, in seconds, for files of various sizes. This test utilized the 18 node pool described above. We can see that this case illustrates remarkably well how hierarchical, cooperative data distribution can improve performance and reduce distribution time. Figure 5.2 shows file distribution times for the larger (52 node) host pool. Again, the performance improvement from making simple scheduling decisions

[2] Whether this is sufficient or not is another matter, as we have done no harm.

[3] The authors speak from experience. What Grid developer hasn't iterated through a file copy to each node of some set?

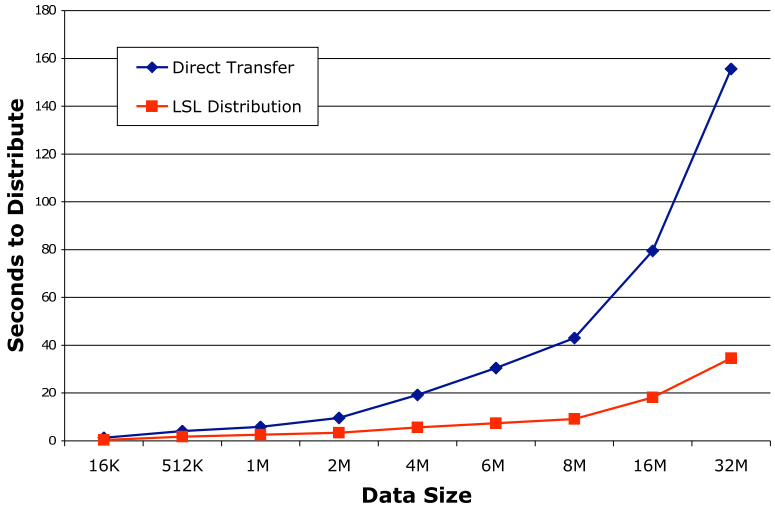

FIG. 5.1. *Distribution Times for 18 Hosts*

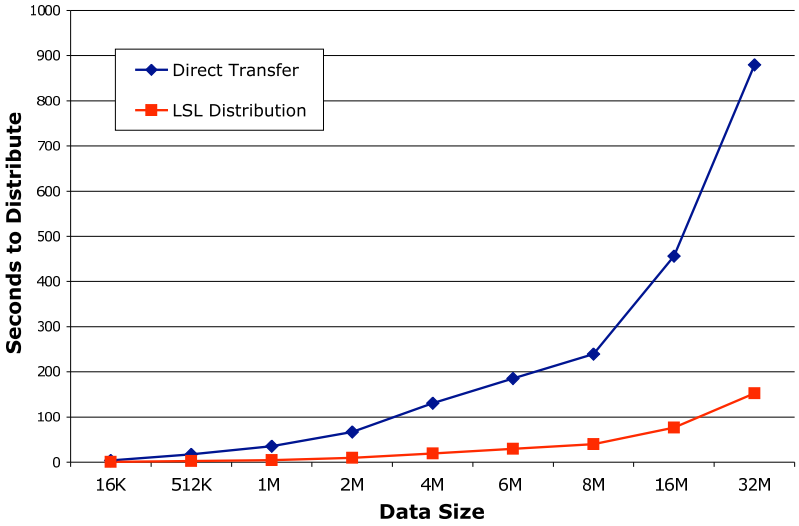

FIG. 5.2. *Distribution Times for 52 Hosts*

is quite significant. We note that clusters represent the best case for distribution techniques such as this and clusters are frequently components in a Grid.

Figure 5.3 depicts the delivered bandwidth that we observe in data transfers to the 18 node host pool. Figure 5.4 shows this same metric for the larger host pool. We have initiated a data transfer that has an average performance more than the physical link to which the machine is attached (12.5MB/sec).

6. Related Work. There are many aspects of research that are similar and related. LSL is part of the more general inquiry of Logistical Networking [28, 6]. This work investigates a more rich view of storage in the network and our scheduling approach is applicable to either infrastructure.

Globus GASS [9] and GridFTP [16] are data movement and staging service for Grid systems that could be scheduled using the techniques that we have described. The MagPIe [20, 40, 21] project has investigated performance optimizations for collective operations. Improving the performance of collective operation has been investigated in many different contexts [4, 27, 24, 5, 18, 39], although primarily the focus has been MPI.

Scheduling application activity based on the state of the network is seen many places including REMOS [23], Topology-d [26] and the Network Weather Service [42].

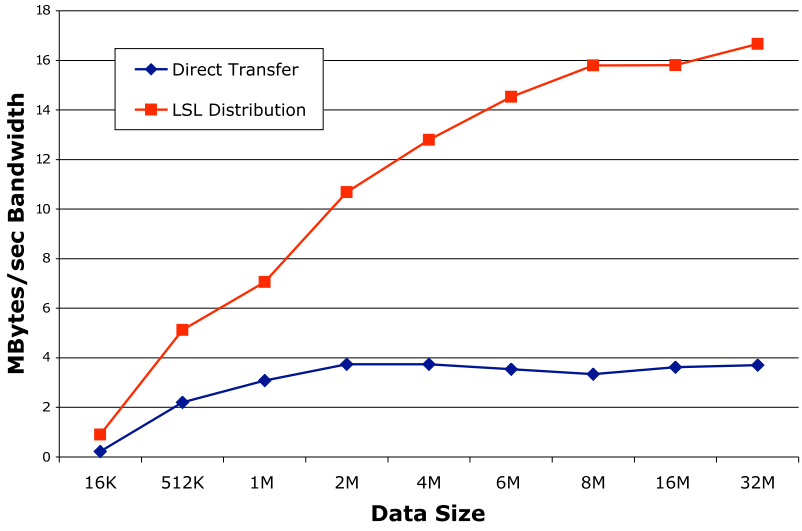

FIG. 5.3. *Delivered Bandwidth of Distribution Tree (18 Hosts)*

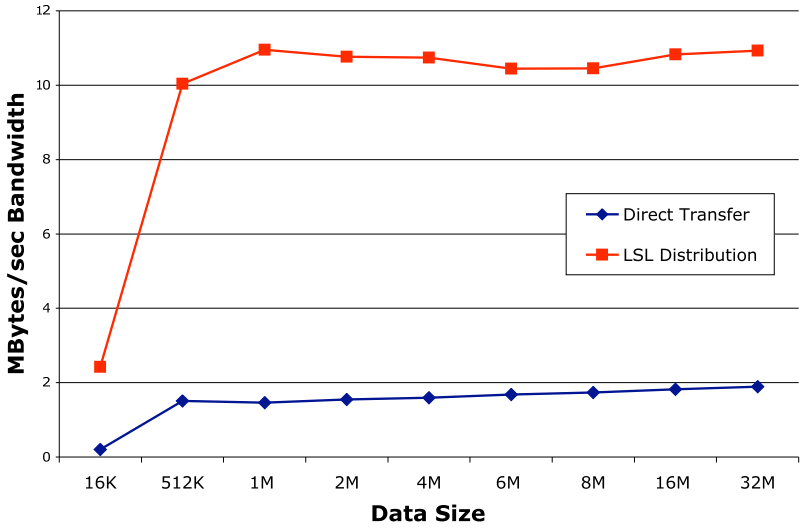

FIG. 5.4. *Delivered Bandwidth of Distribution Tree (52 Hosts)*

Our approach is quite similar to recent work by Malouch, et. al [25], which treats multicast proxies as nodes in a network optimization problem. We note that their arc incidence constraints are different than those that we propose. Further, their simulations were aimed at evaluating various heuristics, while our goal is to understand the performance improvements from simple scheduling in real networks.

Overcast [17] is a network overlay based multicast system. Overcast uses node to node protocols to build and evaluate the distribution trees. Our approach creates distribution trees at runtime and assumes no state in the network. Rather, we assume the availability of network performance forecasts to determine distribution trees. Our concerns about node failure are also quite different given our utilization of TCP as a transport layer.

Recent work in application-level multicast explores the applicability of peer-to-peer networks [31] for this purpose. They note a benefit of their work is the lack of a constantly-running routing protocol, a benefit that we share. In contrast to their approach, however, we don't increase the time to distribute data, rather we decrease it.

7. Conclusion. We have focused on the problem of initial data distribution in Grid environments. By building on previous system components, such as the NWS and LSL, we have developed a novel system for data distribution. We have developed a scheduler that is able to instantiate cooperative data forwarding based on LSL and performance data from NWS. This scheduling technique and infrastructure allow us to form distribution trees that greatly increase performance and reduce time to distribute data. Techniques such as this will only become more important as Grids proliferate.

REFERENCES

[1] *Abilene*. `http://www.ucaid.edu/abilene/`.

[2] R. Ahuja, T. Magnanti, and J. Orlin, *Network Flows: Theory, Algorithms, and Applications*, Prentice Hall, Upper Saddle River, New Jersey, 1993.

[3] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris, *Resilient overlay networks*. Proc. 18th ACM SOSP, Banff, Canada, October 2001.

[4] V. Bala, J. Bruck, R. Cypher, P. Elustondo, A. Ho, C.-T. Ho, S. Kipnis, and M. Snir, *CCL: A portable and tunable collective communication library for scalable parallel computers*, IEEE Transactions on Parallel and Distributed Systems, 6 (1995), pp. 154–164.

[5] M. Banikazemi, V. Moorthy, and D. Panda, *Efficient collective communication on heterogeneous networks of workstations*, in International Conference on Parallel Processing, 1998, pp. 460–467.

[6] M. Beck, T. Moore, J. Plank, and M. Swany, *Logistical networking: Sharing more than the wires*, in Proc. of 2nd Annual Workshop on Active Middleware Services, August 2000.

[7] 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*, Tech. Report Rice COMPTR00-355, Rice University, February 2000.

[8] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao, *Application level scheduling on distributed heterogeneous networks*, in Proceedings of Supercomputing 1996, 1996.

[9] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke, *GASS: A data movement and access service for wide area computing systems*, Sixth Workshop on I/O in Parallel and Distributed Systems, (1999).

[10] L. Cai, *Np-completeness of minimum spanner problem*, Discrete Applied Mathematics, 48 (1994), pp. 187–194.

[11] H. Casanova and J. Dongarra, *NetSolve: A Network Server for Solving Computational Science Problems*, The International Journal of Supercomputer Applications and High Performance Computing, (1997).

[12] O. Egecioglu and T. Gonzalez, *Minimum-energy broadcast in simple graphs with limited node power*, in Proc. IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2001), August 2001, pp. 334–338.

[13] I. Foster and C. Kesselman, *Globus: A metacomputing infrastructure toolkit*, International Journal of Supercomputer Applications, (1997).

[14] GrADS, `http://hipersoft.cs.rice.edu/grads`.

[15] *Graphviz*, `http://www.research.att.com/sw/tools/graphviz`.

[16] *GridFTP*, `http://www.globus.org/datagrid/gridftp.html`.

[17] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek, and J. W. O'Toole, Jr., *Overcast: Reliable multicasting with an overlay network*, in Proceedings of Fourth Symposium on Operating System Design and Implementation (OSDI), October 2000, pp. 197–212.

[18] N. T. Karonis, B. R. de Supinski, I. Foster, W. Gropp, E. Lusk, and J. Bresnahan, *Exploiting hierarchy in parallel computer networks to optimize collective operation performance*, in 14th International Parallel and Distributed Processing Symposium, 2000, pp. 377–386.

[19] R. M. Karp, *Reducibility among combinatorial problems*, in Complexity of Computer Computations, R. Miller and J. Thatcher, eds., Plenum Press, 1972, pp. 85–104.

[20] T. Kielmann, H. E. Bal, S. Gorlatch, K. Verstoep, and R. F. Hofman, *Network performance-aware collective communication for clustered wide area systems*, Parallel Computing, 27 (2001), pp. 1431–1456.

[21] T. Kielmann, R. Hofman, H. Bal, A. Plaat, and R. Bhoedjang, *Mpi's reduction operations in clustered wide area systems*, 1999.

[22] J. Kubiatowicz, D. Bindel, Y. Chen, P. Eaton, D. Geels, R. Gummadi, S. Rhea, H. Weatherspoon, W. Weimer, C. Wells, and B. Zhao, *Oceanstore: An architecture for global-scale persistent storage*, in Proceedings of ACM ASPLOS, ACM, November 2000.

[23] B. Lowecamp, N. Miller, D. Sutherland, T. Gross, P. Steenkiste, and J. Subhlok, *A resource query interface for network-aware applications*, in Proc. 7th IEEE Symp. on High Performance Distributed Computing, August 1998.

[24] B. Lowekamp and A. Beguelin, *Eco: Efficient collective operations for communication on heterogeneous networks*. In International Parallel Processing Symposium, pages 399–405, Honolulu, HI, 1996., 1996.

[25] N. Malouch, Z. Liu, D. Rubenstein, and S. Sahu, *A graph theoretic approach to bounding delay in proxy-assisted*. In 12th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV'02), May 2002. 143, 2002.

[26] K. Obraczka and G. Gheorghiu, *The performance of a service for network-aware applications*, in Proceedings of 2nd SIGMETRICS Conference on Parallel and Distributed Tools, August 1998.

[27] J.-Y. L. Park, H.-A. Choi, N. Nupairoj, and L. M. Ni, *Construction of optimal multicast trees based on the parameterized communication model*, in Proceedings of the International Conference on Parallel Processing (ICPP), 1996, pp. 180–187.

[28] J. S. Plank, A. Bassi, M. Beck, T. Moore, D. M. Swany, and R. Wolski, *Managing data storage in the network*, IEEE Internet Computing, 5 (2001), pp. 50–58.

[29] R. Prim, *Shortest connection networks and some generalizations.* Bell System Technical Journal, 36, 1389–1401, 1957.
[30] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, *A scalable content-addressable network*, in SIGCOMM, 2001, pp. 161–171.
[31] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker, *Application-level multicast using content-addressable networks*, Lecture Notes in Computer Science, 2233 (2001), pp. 14–25.
[32] G. Shao, F. Berman, and R. Wolski, *Using effective network views to promote distributed application performance.* In Proceedings of the 1999 International Conference on Parallel and Distributed Processing Techniques and Applications, 1999.
[33] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakrishnan, *Chord: A scalable content-addressable network*, in SIGCOMM, August 2001.
[34] M. Swany and R. Wolski, *Data logistics in network computing: The Logistical Session Layer*, in IEEE Network Computing and Applications, October 2001.
[35] ———, *Building performance topologies for computational grids.* Proceedings of Los Alamos Computer Science Institute (LACSI) Symposium, October 2002.
[36] ———, *Multivariate resource performance forecasting in the network weather service*, in Proceedings of SC 2002, November 2002.
[37] ———, *Representing dynamic performance information in grid environments with the network weather service.* 2nd IEEE International Symposium on Cluster Computing and the Grid, May 2002.
[38] J. Touch, *The XBone.* Workshop on Research Directions for the Next Generation Internet, May 1997.
[39] S. Vadhiyar, G. Fagg, and J. Dongarra, *Performance modeling for self adapting collective communications for MPI.* Proceedings of Los Alamos Computer Science Institute (LACSI) Symposium, October 2001.
[40] R. V. van Nieuwpoort, T. Kielmann, and H. E. Bal, *Efficient load balancing for wide-area divide-and-conquer applications*, in PPoPP '01: ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, June 2001, pp. 34–43.
[41] P.-J. Wan, G. Calinescu, X. Li, and O. Frieder, *Minimum-energy broadcast routing in static ad hoc wireless networks*, in INFOCOM, 2001, pp. 1162–1171.
[42] 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.
[43] R. Wolski, N. Spring, and J. Hayes, *The network weather service: A distributed resource performance forecasting service for metacomputing*, Future Generation Computer Systems, (1999).
[44] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, *Tapestry: An infrastructure for fault-tolerant wide-area location and routing*, Tech. Report UCB/CSD-01-1141, UC Berkeley, Apr. 2001.

Edited by: Wilson Rivera, Jaime Seguel.
Received: July 5, 2003.
Accepted: September 1, 2003.