Using AppLeS to Schedule Simple SARA on the Computational Grid [*]

Alan Su [†] Francine Berman [†] Richard Wolski [‡]
Michelle Mills Strout [†]

February 28, 1999

Abstract

Computational Grids, composed of distributed and often heterogeneous computing resources, have become the platform-of-choice for many performance-challenged applications. Proof-of-concept implementations have demonstrated that both Grids and clustered environments have the potential to provide great performance benefits to distributed resource-intensive applications. However, at the present time, careful staging, scheduling, and/or reservation of resources is essential in order for applications to achieve performance in Grid environments. If Computational Grids and shared computational clusters are to achieve their full potential, it must be possible for users to achieve application performance at any given time, and when other users are present in the system.

In this paper, we describe the initial development of an **AppLeS** (Application-Level Scheduler) for the resource selection portion of the Synthetic Aperture Radar Atlas (**SARA**) application, developed at JPL. We demonstrate the effectiveness of application scheduling for distributed data applications such as SARA by providing a performance-efficient strategy for retrieving SARA data files in everyday, multiple-user Grid environments.

[*]This work was supported in part by NSF ASC-9701333, DoD Modernization Contract 9720733-00, NSF NPACI Award ASC9619020, DARPA Contract N66001-97-C-8531, NSF Graduate Fellowship (UCSD Grant #DGE-9616037), an AT&T Labs Student Grant, and NASA GSRP Fellowship Grant #NGT-2-52251.

[†]Department of Computer Science and Engineering, University of California San Diego, La Jolla, CA 92093-0114

[‡]Department of Computer Science, University of Tennessee, Knoxville, TN 37996-1301

1 Introduction

Distributed resources are becoming an increasingly important platform for computation. Both clusters of workstations and Computational Grids provide a way to aggregate the computational power supplied by collections of resources to provide execution performance for large-scale and resource-intensive computations.

Achieving application performance on such platforms can be challenging. Contention for shared resources has considerable impact on the performance such resources can deliver to an application. In addition, the heterogeneity of the underlying resources makes performance hard to predict. Currently, large-scale applications may achieve performance by careful staging of data and/or computation, as well as scheduling of application tasks and communication. However, for the ordinary user seeking application performance, such measures are often impractical and/or infeasible. The challenge for application developers and users is to develop applications which can be scheduled at execution-time to achieve performance under ordinary conditions, and in shared and dynamic production computational environments.

Experience shows that adaptive techniques are a promising way to schedule applications in Grid and clustered environments [9]. In this paper, we use such techniques to develop an application scheduler for the data transfer phase of the SARA application, a distributed data image acquisition application developed at JPL and SDSC. In Section 2, we briefly outline the basic concepts behind AppLeS application schedulers. In Section 3, we describe an AppLeS application scheduler for a simplified version of SARA, and in Section 4, we provide performance results. Some relevant work is discussed in Section 5, and in Section 6, we touch on future work by outlining an "end-to-end" SARA AppLeS and our plans to use it as a template for the larger class of distributed data applications of which SARA is a member.

2 AppLeS

The focus of the AppLeS (**App**lication-**Le**vel **S**cheduling) project [3] is to design and develop custom application-level scheduling agents for distributed applications. Each application is integrated with its own AppLeS scheduler which develops and implements a custom application schedule, which can adapt to forecasts of deliverable resource performance at execution time. AppLeS agents use performance prediction models, dynamic information, and application-specific information to determine an adaptive custom sched-

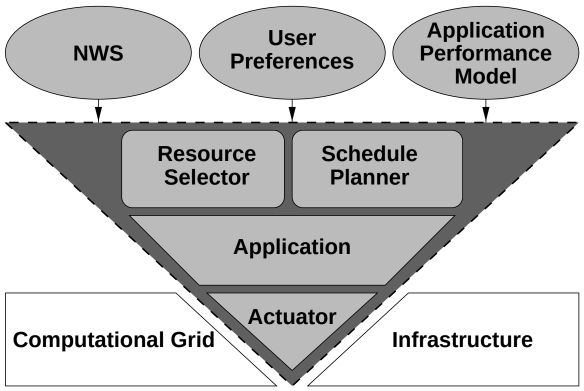

Figure 1: **AppLeS Architecture.** Subsystems include the Resource Selector, Schedule Planner and Application Actuator. Information is provided by the user, the system, and by facilities such as the Network Weather Service.

ule for their application.

AppLeS schedulers are based on the principle that **every potential scheduling decision has a performance impact on the application** [2]. An AppLeS agent uses performance models and resource performance forecasts to derive an adaptive custom schedule for its application, and to choose the schedule from among a set of possible candidates that best optimizes the user's performance criteria. To do so, the agent must

- *select* potentially useful sets of resources,
- *derive* an application schedule that uses the resources in each set,
- *predict* the performance an application would attain if it used these resources and schedule, and
- *compare* different combinations of resources and schedules (based on the performance predicted for each) to choose the one that is likely to yield the best performance.

These functions are implemented by the subsystems depicted in Figure 1.

3 SARA

The Synthetic Aperture Radar Atlas (SARA) is a web-based distributed data application which allows users with access to the World-Wide Web and the Internet to view images of the Earth's surface taken by a synthetic aperture radar [13]. The data upon which these images are generated are collected by satellites as they pass over the Earth, resulting in long, rectangular regions, or **tracks** of data. Because these data are stored in raw uncompressed SAR format, it is partitioned and replicated across several high-capacity storage sites. Via a web page and a Java applet, users of the SARA system can request an image of an arbitrary sub-region of the track with certain features of the data highlighted.

The SARA application is essentially comprised of three logical phases, the first of which is the **data retrieval phase**. During data retrieval, raw SAR data corresponding to the requested region of a track is located and retrieved, usually from a high-capacity *storage subsystem* like the High-Performance Storage System (HPSS) filesystem. This is done with the help of a *metadata server* which maps data tracks to sets of servers which have the data. Data stored in these filesystems achieves access times ranging from seconds (if the files are in a disk cache) to perhaps even hours (if the files are not in disk cache and require operator intervention to mount a set of tapes). The final step of this phase requires moving the raw data to a *processing node* in preparation for processing the image.

In the **data processing phase**, raw data is converted into an image file. The data is filtered, reduced, and encoded, based on two user inputs: the features the user wishes to highlight, and the image format the user requests. The final logical phase of the application begins when the image file is prepared.

The **image transfer phase** involves moving the image file from the processing node to the user's machine. The current SARA application has only a Web-based interface, meaning that the image transfer is done via an HTTP connection, and the image is displayed in a web browser.

The job of scheduling an application such as SARA can be logically performed in a hierarchical and distributed fashion: scheduling decisions made at one phase can be used to make subsequent choices. No scheduling is required of the data server node, since a single data server needs only fetch data from the storage subsystem which it serves and transfer it back to the requester. However, during the data retrieval phase, processing nodes act as resource selectors and schedule planners in deciding which data servers should be employed to retrieve which data sets. These results can then be used by the Java client to make a similar decision about which processing node should be responsible for generating which parts of the requested image. In the next sections, we describe our initial implementation of and results from the resource selection and planning components of the AppLeS agent which operate at the processing node level. A description of further plans for the development of an "end-to-end" SARA AppLeS are described in Section 6.

3.1 Simple SARA

As the initial stage of the development of an "end-to-end" SARA AppLeS, we implemented an AppLeS scheduling agent for the data retrieval phase of the SARA application only (which we term "Simple SARA"). Figure 2 illustrates the basic structure of the SARA application, with the component to which this work applies shown in bold. In particular, we focused on the transfer of raw data for some region of a single track from the storage site to a processing node. Since some of the data may be stored in multiple archives, we concentrated on the more difficult case where data can be retrieved from multiple servers. (If data can be accessed only from a single server, the SARA tool would of course access it from there). In the multiple data server case, wide performance variations on the networks between the data servers and the processing node have considerable impact on data transfer rates. The selection of which data server to use for the fastest transfer of remote data is the focus of the Simple SARA AppLeS.

For the initial prototype, we assumed that the files to be retrieved were on disk rather than tape, so access time to the data is uniform for each of the potential data servers. This is consistent with the SARA application as it is currently used. The performance model used for resource selection by the Simple SARA AppLeS is straightforward:

$$Time = \frac{DataSize}{Bandwidth}$$

Given a user request, the AppLeS agent can calculate *DataSize* and get a forecast of *Bandwidth* from the **Network Weather Service** (NWS) [14, 15]. Since *DataSize* is fixed based on the data SARA needs to process a request, *Time* is inversely proportional to *Bandwidth.* The Simple SARA AppLeS selects the server with the highest forecasted bandwidth and retrieves its data file.

4 Simple SARA Results

In the current implementation of SARA, the user selects the desired data server based on his/her intuition about which site is "closest", i.e. which site will transfer remote data the most quickly. Traffic on shared networks can

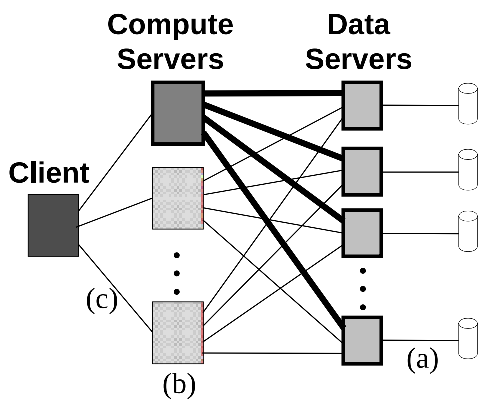

Figure 2: General architecture used in the design of an "end-to-end" SARA AppLeS agent. The portion targeted by Simple SARA is shown in bold. The "end-to-end" SARA AppLeS agent's performance prediction model will account for performance-driving factors: (a) storage access times, (b) data processing and image generation times, and (c) HTTP-based data transfer times.

substantially impact performance and can render a non-intuitive choice of data server to be the "closest". Our goal for the Simple SARA AppLeS was to develop a strategy for determining the "closest" data server based on dynamic information and prediction provided by the Network Weather Service, and to test our approach in ordinary shared, distributed environments.

To do so, we distributed files representative in size and structure of SARA data files to a variety of geographically distributed sites connected by a diverse set of networks. The machine used as the processing node was `alicatado.ucsd.edu`, a Pentium class x86 machine running Linux 2.0.34. The data servers we used included

- `lolland.cc.gatech.edu`: an x86 PC running PC Solaris 2.5, routed to UCSD from Georgia Tech via vBNS,
- `mead2.u.washington.edu`: an IBM AIX 4.2 machine, routed to UCSD from University of Washington, Seattle via general Internet media,[1]
- `perigee.chpc.utah.edu`: a Sun Ultra-2 workstation running Sun So-

[1]Since the time when our initial experiments were run, the vBNS connection to the University of Washington has been activated.

laris 2.6, routed to UCSD from University of Utah via vBNS,

- `sitar.cs.uiuc.edu`: a Sun Ultra-30 running Sun Solaris 2.6, routed to UCSD from University of Illinois, Urbana-Champaign via vBNS, and
- `spin.cacr.caltech.edu`: an IBM SP2 running IBM AIX 4.2, routed to UCSD from CalTech via general Internet media.[2]

We implemented two prototype modules: a data server and a processing server. Our experimental data server is virtually unchanged from the data server used in the actual SARA application.[3] Our processing server is based on the actual SARA processing server module, but is modified to skip the data filtering and image encoding phases. The AppLeS agent is coded into the processing server.

The experiments consisted of *trials*, during which the AppLeS agent contacted the NWS to obtain bandwidth forecasts between each of the potential data servers and the processing node. The server with the highest bandwidth forecast is designated the *selected server*. We assessed the effectiveness of the AppLeS selection of a data server by performing the data transfer from *all* available data servers and comparing the resulting transfer times.[4] We considered the AppLeS scheduler successful for a given trial if the *selected server* yielded the lowest transfer time for that trial.

Figure 3 shows a representative Simple SARA experiment, performed during a normal workday. Each trace represents a series of trials using a particular server, and the server selected by the AppLeS agent is indicated (with a $\diamond$ symbol) for each trial. We see that during this experiment, the AppLeS scheduler selected the fastest server in 80% of the trials. Furthermore, the AppLeS agent never does worse than an "intuitive" static scheduler which selects the server based on geographical distance between the client PC and the remote data server.

In the next subsection, we discuss the idea of "network closeness" and show some results which suggest that it can be used to construct efficient application schedules.

[2]At the time of these experiments, CalTech's vBNS connection had been approved.

[3]The only difference is that the experimental data server outputs to a socket, whereas the production server runs as a CGI-bin script (and therefore outputs to standard output).

[4]Comparison experiments were run back-to-back and multiple times on transfer sizes ranging from 1.5 to 3 megabytes (corresponding to a typical SARA user request). Each comparison run executed for between 15 seconds and 2 minutes with the majority of runs being on the lower end of the range. On average, comparison runs for a single trial enjoyed roughly similar load conditions.

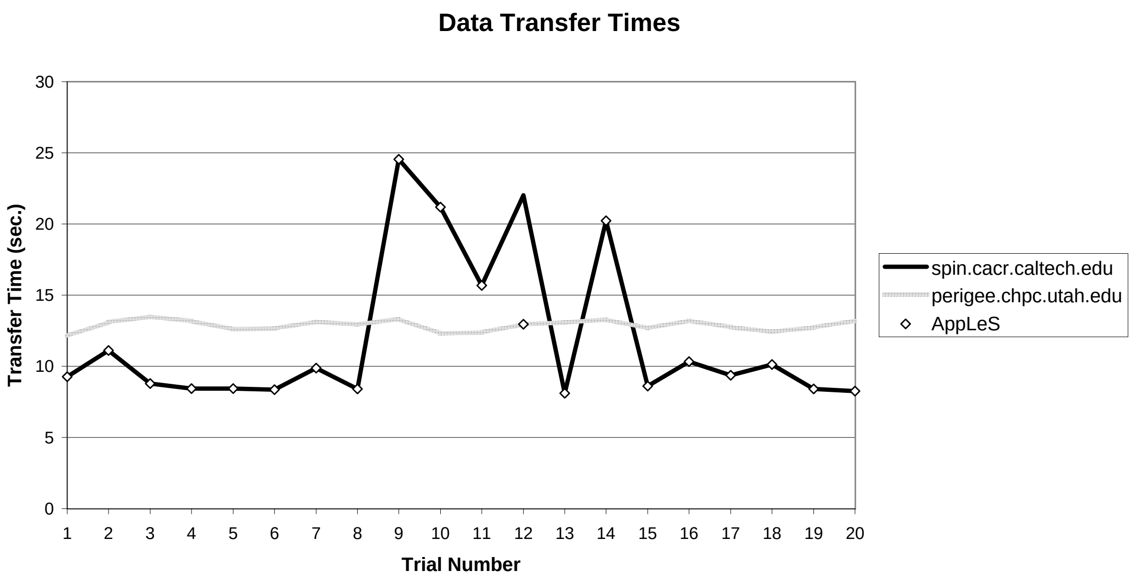

Figure 3: SARA AppLeS performance under typical network behavior

4.1 Observations on What is "Close"

As described previously, the current version of SARA relies on the user to choose the "closest" data server. Typically, many users assume that transfer times increase with geographical distance. That is, the farther away a server is on a map, the longer data transfers will take. However, it is becoming increasingly true that network characteristics between two sites depend less on actual distance and more on the media which connect them. Our Simple SARA experiments led to several interesting observations about what is "close" on the network:

- **"Close" network sites may have little correlation with "close" geographical sites.**
  Fast networks, like the vBNS, make it difficult to determine "closeness" based solely on geographical distance. Sites that may be separated by thousands of miles may be "close" in the network sense, whereas sites separated by only a few hundred miles may be "far".

  Consider experiments run between UCSD, the University of Washington (which was on the general Internet), Georgia Tech (which was on the vBNS), and the University of Illinois (also on the vBNS). During

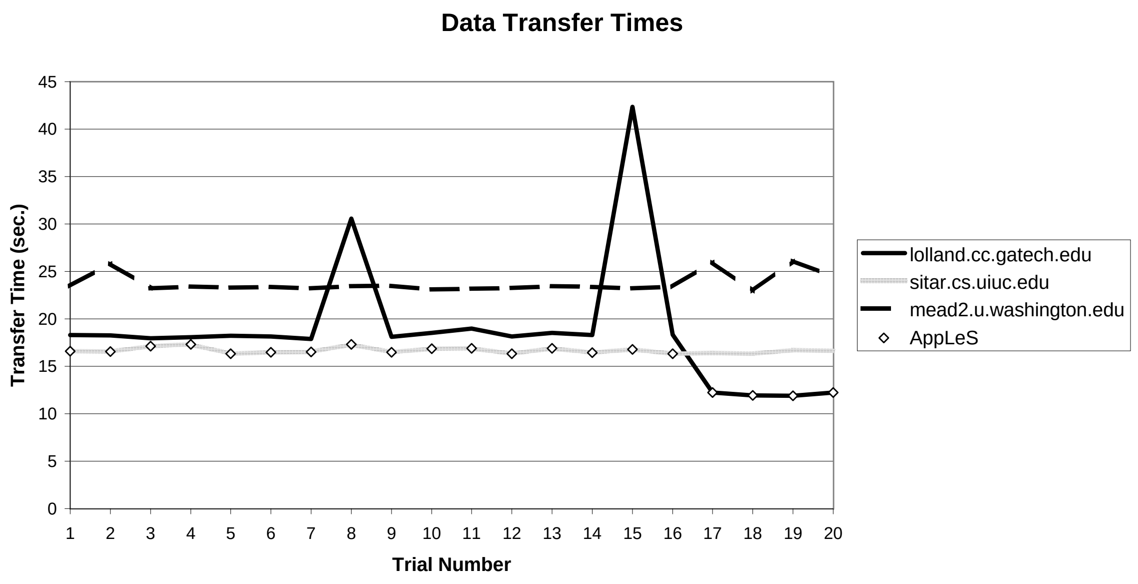

Figure 4: Experiments between UCSD, UW, GeorgiaTech, and UIUC on 8/23/98.

the time frame shown in Figure 4, data files from both Georgia Tech and UIUC enjoy more bandwidth than data files from UW (on average 36.9% and 42.4% respectively) and thus, arrive at UCSD sooner. In this experiment, the AppLeS agent, with the help of NWS forecasts, is able to detect that the UW server, although geographically the closest to UCSD, is not the site that yields the shortest network transfer time.

- **Dynamic factors influence what is "close".**
  In the Internet, network links may be added, removed, and changed on a daily, and often more frequent basis. In addition, the great majority of network links are used by multiple users at any given time. All of these factors contribute to wide fluctuations in the performance of the network. As this happens, the relative network "distance" between sites can change dramatically.

  Although both `sitar.cs.uiuc.edu` and `lolland.cc.gatech.edu` are supposedly connected to the vBNS, we found that Georgia Tech's vBNS connection was subject to intermittent failures during the time frame of our experiments. Figure 4 shows that bandwidth from both

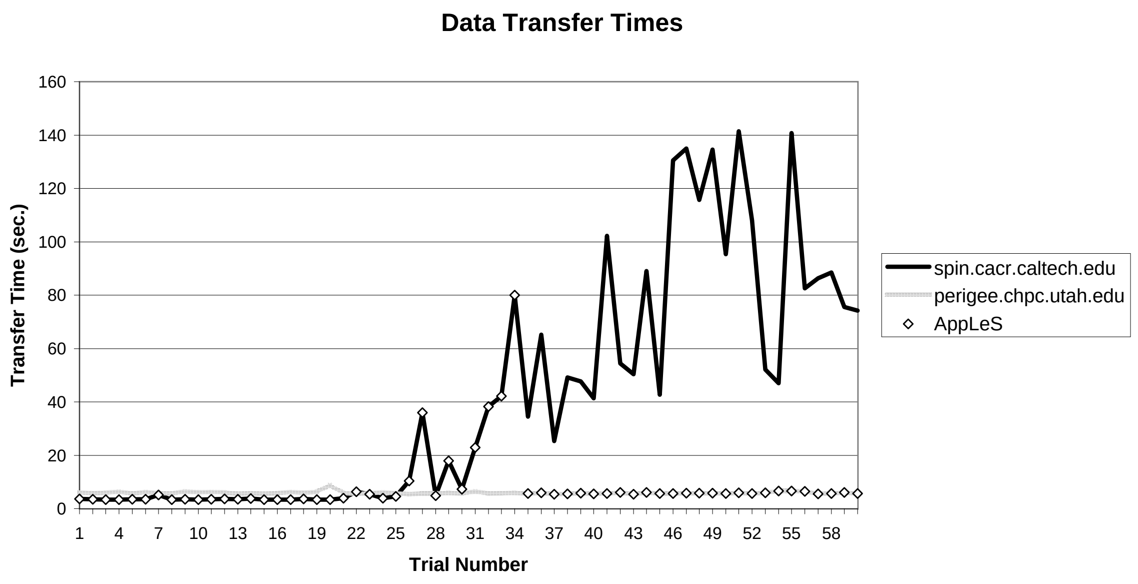

Figure 5: Experiments on 9/21/98 between UCSD, CalTech and University of Utah.

of these two machines is roughly the same. However, at trial 17, we see that bandwidth to the Georgia Tech server is significantly increased, and we conjecture that this is due to the vBNS link being re-established.[5] The NWS measurements and forecasts correctly detect the change allowing the AppLeS agent to profitably switch servers.

Also, consider the experiments performed in the morning on 9/21/98 between CalTech, the University of Utah, and UCSD (Figure 5). During this time frame, the general Internet (to CalTech) provided better performance until trial 26 (approximately 9 AM), at which time the webcast of the Grand Jury testimony of President Clinton commenced. At this point, bandwidth on the general Internet began to fluctuate greatly, and the vBNS connection to Utah provided superior performance.

[5]When this phenomenon was first noticed, we began running `traceroute`s to each machine to determine if network traffic at any given time was using the vBNS. In one instance, when the NWS reported a sudden increase in bandwidth, `traceroute` indicated that the vBNS link was being used. Closely monitoring the link, we found that approximately two hours later, the routing changed from vBNS to the general Internet, accompanied by a sudden decrease in bandwidth.

- **What is "close" may not be obvious to the user.**
  Whereas a user can fairly easily determine the geographical distance between two points, network "closeness" is often an opaque concept to the user. Because of considerable performance variation, and the difficulty in determining the difference between "noise" and "trends" in network performance, it is not straightforward for users to determine which network link will provide the best performance for their application [14].

5 Related Work

"Network distance" in various forms has been studied by a number of groups. For example, NLANR's Measurement and Operations Analysis Team measured web server and cache performance, recognizing the effects of network and server load on effective performance [6]. The multimedia community is also examining this notion extensively, especially in the area of video-on-demand service. Such projects include [5, 11]. We believe that the network behavior exhibited during the Simple SARA experiments support these notions of "network distance" and can be viewed within a larger framework.

Note that SARA is also related to digital library applications which share the goal of accessing files over shared networks. The Alexandria Digital Library Project at UCSB has studied the performance of Web-based applications in the context of digital library and information access systems [1, 16]. This work addresses issues of load balancing among clustered digital library server nodes to improve throughput in response to client requests.

Other web proxy caching work in [4, 8] examines the use of document caches to improve performance for web clients. This work focuses on scheduling in a slightly different context: the process of deciding what and where to perform caching and the subsequent decision of which of several potential caches to use for future queries. These papers are representative of a growing body of work focusing on the development of performance-efficient strategies for accessing and processing remote files in multi-user environments.

6 Future Work

The Simple SARA AppLeS demonstrates that application-level scheduling is a useful strategy for data retrieval. We are currently extending the Simple SARA AppLeS to provide performance for an "end-to-end" SARA tool. Figure 2 illustrates the application architecture being used to develop the

SARA AppLeS agent. In particular, our "end-to-end" SARA AppLeS focuses on applications which require multiple files per image, and may process the image at a variety of different sites (at the data server, at an intermediate compute server, at the web site, etc.). The Simple SARA resource selection strategy will provide a basic building block for an "end-to-end" SARA AppLeS.

Note that the performance model will be more complex for the "end-to-end" SARA. In particular, the interactions of the client and the compute server(s), the compute servers and data servers, as well as data servers of various sorts of media will all have to be modeled. Scheduling the "end-to-end" application will involve decisions about which compute server(s) to use, whether to move the computation to the data, the data to the compute server, or to compute at an intermediate site, and which of the different data servers to target. Optimizing each of these activities individually may not necessarily optimize the execution performance of the "end-to-end" application, so the application must be scheduled as whole.

We also note that SARA is representative of a larger class of **distributed data applications**. High energy physics applications from the Cleo/Nile project [10], Digital Sky [7], San Diego Supercomputer Center's Storage Resource Broker (SRB) [12], and other applications have a similar computational structure to SARA and similar computational goals, although the size of the data files, location of client, servers and visualization sites, computation to be performed on the data, and other details may differ. We plan to use the "end-to-end" SARA AppLeS as a model for the development of a distributed data AppLeS *template* which can be used for scheduling other distributed data applications on the Grid.

Acknowledgements

We are grateful to Jack Dongarra and Bernard Tourancheau for inviting us to the Cluster Workshop at Blackberry Farm. Comments from the anonymous reviewer of the initial version of this paper helped us immensely in improving the paper. Finally, this work greatly benefitted from useful and substantive discussions with our colleagues on the AppLeS team as well as Frederica Darema, George Kremenek, and Roy Williams.

References

[1] Daniel Andresen, Tao Yang, Oscar H. Ibarra, and Ömer Eğecioğlu. Adaptive partitioning and scheduling for enhancing WWW application performance. *Journal of Parallel and Distributed Computing*, 49:57–85, 1998.

[2] F. Berman and R. Wolski. Scheduling from the perspective of the application. In *Proceedings of High-Performance Distributed Computing Conference*, 1996.

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

[4] Azer Bestavros, Robert L. Carter, Mark E. Crovella, Carlos R. Cunha, Abdelsalam Heddaya, and Sulaiman A. Mirdad. Application-level document caching in the internet. In *Proceedings of the Second International Workshop on Services in Distributed and Networked Environments*, 1995.

[5] Guiseppe Bianchi and Riccardo Melen. Performance and dimensioning of a hierarchical video storage network for interactive video services. *European Transactions on Telecommunications*, 7(4):349–358, Jul-Aug 1996.

[6] Hans-Werner Braun and Kimberly Claffy. Web traffic characterization: An assessment of the impact of caching documents from ncsa's web server. In *Proceedings of the Second International World Wide Web Conference*, 1994.

[7] Caltech's **Digital Sky** project (in progress) webpage at `http://www.cacr.caltech.edu/digisky`.

[8] Anja Feldmann, Ramón Cáceres, Fred Douglis, Gideon Glass, and Michael Rabinovich. Performance of web proxy cachin in heterogeneous bandwidth environments. In *Proceedings of Infocom '99*, 1999.

[9] Ian Foster and Carl Kesselman. *The Grid: Blueprint for a New Computing Infrastructure*, chapter 12. Morgan Kaufmann Publishers, Inc., 1998.

[10] K. Marzullo, M. Ogg, A. Ricciardi, A. Amoroso, F. Calkins, and E. Rothfus. Nile: Wide-area computing for high energy physics. *Proceedings of the 1996 SIGOPS Conference*, 1996.

[11] Kyeongho Park, Yanghee Choi, and Chong Sang Kim. Scheduling of storage and cache servers for replicated multimedia data. In *Proceedings of High Performance Computing on the Information Superhighway HPC Asia '97*, 1997.

[12] SDSC's **Storage Resource Broker** project (in progress) webpage at `http://www.npaci.edu/DICE/SRB/index.html`.

[13] Roy Williams. Caltech's **Synthetic Aperture Radar Atlas** project webpage at `http://www.cacr.caltech.edu/r̃oy/sara/index.html`.

[14] R. Wolski. Dynamically forecasting network performance using the network weather service. *Cluster Computing*, 1998. available from `http://www.cs.ucsd.edu/users/rich/publications.html`.

[15] R. Wolski, N. Spring, and J. Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems (to appear)*, 1998. available from `http://www.cs.ucsd.edu/users/rich/papers/nws-arch.ps`.

[16] Huican Zhu, Tao Yang, Qi Zheng, David Watson, Oscar H. Ibarra, and Terence Smith. Adaptive load sharing for clustered digital library servers. In *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, 1998.