Biological Sequence Alignment On The Computational Grid Using The GrADS Framework [⋆]

Asim YarKhan [a] Jack J. Dongarra [a,b]

[a] *Computer Science Department, University of Tennessee, Knoxville, TN 37996*

[b] *Computer Science and Mathematics Division, Oak Ridge National Laboratory, Oak Ridge, TN 37831*

Abstract

In spite of the existence of several Grid middleware projects, developing and executing programs on the computational Grid remains a user intensive process. The goal of the Grid Application Development Software (GrADS) project is to make the Grid simpler to use despite the dynamically changing status of Grid resources. Protein and genome sequence alignment is a basic operation in bioinformatics, and it requires large datasets and tends to be highly compute intensive. In this paper, we present work done to grid-enable a biological sequence alignment package (FASTA) and to run it under the GrADS framework. We discuss the advantages of using GrADS framework for FASTA.

Key words: GrADS project; Grid scheduling; Biological sequence alignment;

[⋆] This work is supported in part by the National Science Foundation contract grant #E81-9975020, SC R36505-29200099, R011030-09, "Next Generation Software: Grid Application Development Software (GrADS)", and National Science Foundation contract grant #R38143-79200003, "GrADS Extension" and also in part by the Applied Mathematical Sciences Research Program of the Office of Mathematical, Information, and Computational Sciences, U.S. Department of Energy under contract DE-AC05-00OR22725 with UT-Battelle, LLC.

Email addresses: `yarkhan@cs.utk.edu` (Asim YarKhan), `dongarra@cs.utk.edu` (Jack J. Dongarra).

1 Introduction

The Grid Application Development Software (GrADS) project [5] has defined a framework to make it simpler to prepare and execute programs on a computational Grid. In order to guide the development of this framework, an implementation known as GrADSoft [8,9] was developed together with a set of software packages that use the framework. This set of packages includes the numerical linear algebra library ScaLAPACK [25], the astrophysics problem solving environment Cactus [2,19] and satisfiability solvers for circuit design [7].

This paper describes the work done to enable a parallel, master-worker implementation of the biological sequence alignment application FASTA [24] to run on the GrADSoft framework. There have been many implementations of grid based sequence alignment applications. Our implementation is designed to demonstrate several things. Firstly, it demonstrates the ease with which the GrADS infrastructure can be used to grid-enable a legacy code. Secondly, it acts be an example of a grid application bound by data locality, where the computation must be scheduled at the site of the data.

Several projects provide generic facilities to run master-worker applications on a grid, such as the Condor MW (Master-Worker) implementation [15], the AppLeS Master Worker Application Template (AMWAT) [28] and the NetSolve system [1]. The Condor MW and the AMWAT approaches require that the application provide a specific programming interfaces that can be called by a scheduler. The NetSolve interface is simpler, but it is designed only for problems that can be decomposed into a bag-of-tasks that are executed using a simple remote procedure call interface. All of these approaches would require substantial changes to a pre-existing application, and none of them is designed to schedule an application using the data-locality constraints (i.e., distributed, partial data sets) that are addressed in this work.

Spring and Wolski [30] discuss scheduling a Master-worker implementation of FASTA on a metacomputer using application specific performance models. A static schedule based on run time resource information and application specific performance models resulted in a faster execution than simple self-scheduling. But the best execution time was obtained with an adaptive approach that uses resource information to schedule most of the work, and then self-scheduling to complete. However, this work did not consider data-locality constraints when making scheduling decisions.

There are many other projects aimed at enabling Grid resources to be used for bioinformatics, including the Japanese BioGrid [6], North Carolina BioGrid [22], MyGrid [21], and APBioGrid [4]. These projects generally provide a por-

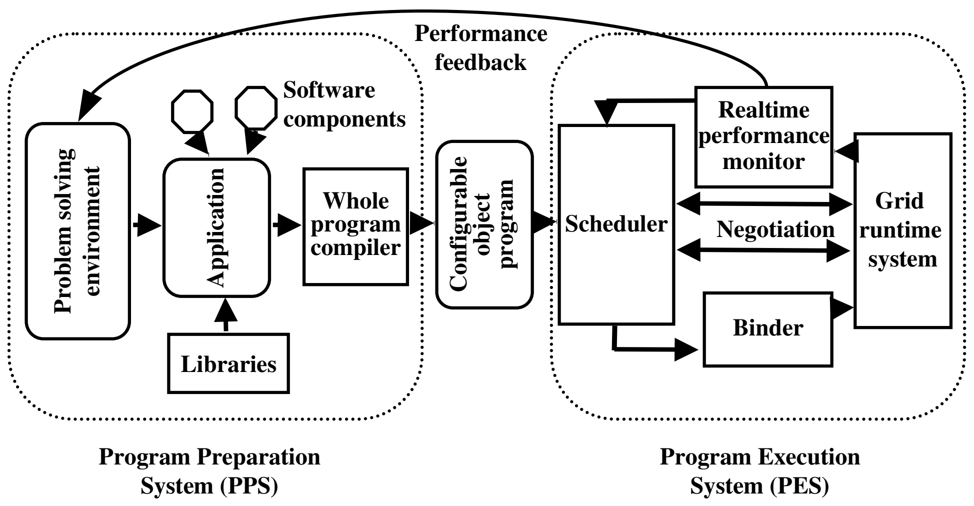
Performance feedback
Software components
Problem solving environment
Application
Whole program compiler
Libraries
Configurable object program
Scheduler
Realtime performance monitor
Negotiation
Grid runtime system
Binder
Program Preparation System (PPS)
Program Execution System (PES)

Fig. 1. The Grid Application Development Software (GrADS) Architecture

tal interface from which a user can run a specific set of programs on Grid resources. In contrast, the GrADS project is designed to enable the end user to run any arbitrary executable on the Grid resources in an transparent and efficient manner.

2 The Computational Grid and the GrADS Project

Several Grid computing infrastructure projects exist to enable the use of geographically and administratively distributed resources, such as Globus [13] and Legion [16]. However, developing and deploying applications across Grid resources remains a user intensive process. Among other tasks, the user is responsible for ensuring that all the required framework (e.g., libraries and databases) exists on the Grid resources, that the resources are available and not busy, and that the connectivity between the resources is sufficient. Additionally, the user needs to track the execution of the application to ensure that the Grid resources have not changed so as to cause the application to fail or be unacceptably delayed.

The Grid Application Development Software (GrADS) [5] project is a multi-university research project aimed at simplifying distributed heterogeneous computing. The GrADS project provides tools and technologies for the development and execution of applications in a Grid environment. In the GrADS vision, the end user simply presents their parallel application to the framework for execution. The framework is responsible for scheduling the application on an appropriate set of resources, launching and monitoring the execution, and if necessary, rescheduling the application on a different set of resources. A high-level view of the GrADS architecture [17] is shown in Figure 1.

In GrADS, the *Program Preparation System* (PPS) handles application development, composition, and compilation. The application code is transparently manipulated to integrate software libraries and to prepare it for further processing. An intermediate view of the application is developed (the *Configurable Object Program*), which encapsulates all the results of this stage for later usage, including application specific performance models and data mappers.

The *Program Execution System* (PES) handles resource discovery, scheduling, execution, performance monitoring and rescheduling. In order to execute an application, the user submits the application parameters to the GrADS system and the PES is invoked. The scheduler uses a grid run-time system which is built on top of Globus Monitoring and Discovery Service (MDS) [13] and Network Weather Service (NWS) [33] to determine the availability and status of the appropriate grid resources. The performance model and mapper are used by the scheduler to determine a good subset of the resources for the execution. The binder compiles the code to the resource-specific format, and enables the performance monitoring to take place. The application is then launched on the scheduled resources. A real-time performance monitor tracks the application performance on the grid resources, and if the performance contract (i.e., expected performance behavior) [31] is violated, the rescheduler may migrate the application to alternate resources.

3 Biological Sequence Matching and the Grid

Sequence matching is one of the most important primitive operations in computational biology, often forming the basis of more complicated and sophisticated operations. For example, projects that assemble DNA from shotgun sequencing use similarity searches to find overlapping fragments. The two tasks involved in matching sequences are similarity computation and alignment. In similarity computation, a metric is calculated that measures the syntactic difference between two sequences. In the alignment task, the costs of additions, deletions and substitutions required to match one sequence with the other are calculated. The matrix of costs associated with the additions, deletions and substitutions are determined by biologists. Sequence matching tasks often need to be performed repeatedly over huge protein and genome databases.

The full pairwise sequence matching task is usually solved as using dynamic programming technique, similar to calculating the edit distance between two strings. Well known dynamic algorithms exist to compute the alignment (e.g., Smith-Waterman [29], Needleman-Wunsch [23]), however the computational costs of these algorithms is high. With the size of the protein and genome databases growing rapidly, the methods tend to be too slow on traditional computing resources. Many heuristic approaches to speeding up the alignment problem exist, two of the better known are the fast BLAST algorithm [3] and the slightly slower, more accurate FASTA algorithm [24]. FASTA is approximately 10-50 times faster than the Smith-Waterman algorithm and is often used as a good compromise between speed and accuracy.

As of early 2005, the BioMirror archive [14] contains about 66 Gigabytes of bioinformatics databases in compressed format from various sources, and this collection is growing rapidly. Given the size and rate of growth of the protein and genome sequence databases, it is undesirable to transport and replicate all the databases at all the sites involved in a Grid. Our approach can use databases that exist at a subset of the sites, bringing the computation to the location of the data. Since searches over different parts of the reference database can be carried out without any communication between nodes, the sequence alignment application is an excellent fit for a Grid implementation.

There are many scenarios in bioinformatics that could benefit from data locality constraints so that large datasets do not need to be transferred over the network. In one example scenario, all-to-all genome alignments over multiple databases can be used to study phylogeny. In another example that could benefit from data locality, the Encyclopedia of Life (EOL) project [18] seeks to characterize all proteins encoded by publicly available genomes through putative assignment of structures models and functions.

In this paper, we present one approach to adapting the sequence alignment package FASTA [12] to run on a computational Grid.

4 FASTA, GrADSoft and Data Locality

The FASTA sequence alignment code developed by William Pearson [24,12] was used as the base for this work. Pearson's original MPI-based master-worker implementation of FASTA assumed that the reference databases were only available at the master node. This reference data was distributed using messages from the master to the workers to provide an approximately equal portion to each worker. Each worker was then sent a query sequence, which was processed against its reference data, and the results were returned. The master would collate all the results, and send out the next query sequence.

In the GrADS version of FASTA the reference protein and genome databases are replicated on some or all of the grid nodes, either in whole or in part. This is intended to reflect a real-world situation where multiple large databases are created and located at distributed sites. These large databases should not be shipped over the network, so computation has to be scheduled at the site of the data so as to cover all the desired reference data and to complete as soon as possible. Aside from this change to FASTA's startup routines, the original code was left unaltered. One of the aims of the GrADS project is to grid-enable pre-existing code with minimal (or preferably no) changes to the code.

In GrADSoft, static execution schedules are used to select the resources that the application will use. These are developed using application specific performance models. The specifics of scheduling are discussed in the next section.

During execution, the master sends a message informing each worker what portion of which database it should load into memory. Since the worker nodes on the Grid need not be homogeneous, in order to balance the workloads, different amounts of work are scheduled at each worker. If necessary, workers may also be assigned partial databases as their workload. The master node then distributes query sequences to each worker and collects and collates the results. This is repeated for each remaining query sequence.

The GrADS adaptation of FASTA provides an interesting scheduling challenge due to the data locality requirement and large computational requirement.

5 Performance Model, Mapper and Scheduling

In GrADSoft, the *performance model* and *mapper* are compiled routines whose input includes a set of machine and network parameters, and whose output is the estimated execution time for the application on those resources. The application specific performance model and mapper are loaded by GrADSoft at runtime, and are used by the scheduler to evaluate possible schedules and to guide a search process which selects an appropriate schedule for the application.

The FASTA performance model estimates the execution time on a specific set of Grid nodes given measures of machine and network characteristics (e.g., the free memory, CPU power/availability and network latency and bandwidth). These resource characteristics are obtained from Globus MDS and the Network Weather System and may be statistically estimated into the near future. To ensure good performance, the performance model requires that the reference data fit completely in the available free memory. Performance on new CPUs is estimated by a scaling factor based on the relative performance of a known benchmark (i.e., matrix-matrix multiply) on the new CPU. This is obviously a simplification, but the estimates were found to be acceptable.

The performance model was determined by running a variety of experiments over query sequences and databases on unloaded systems. The observed execution times were fitted to a nonlinear model using on the length of the query

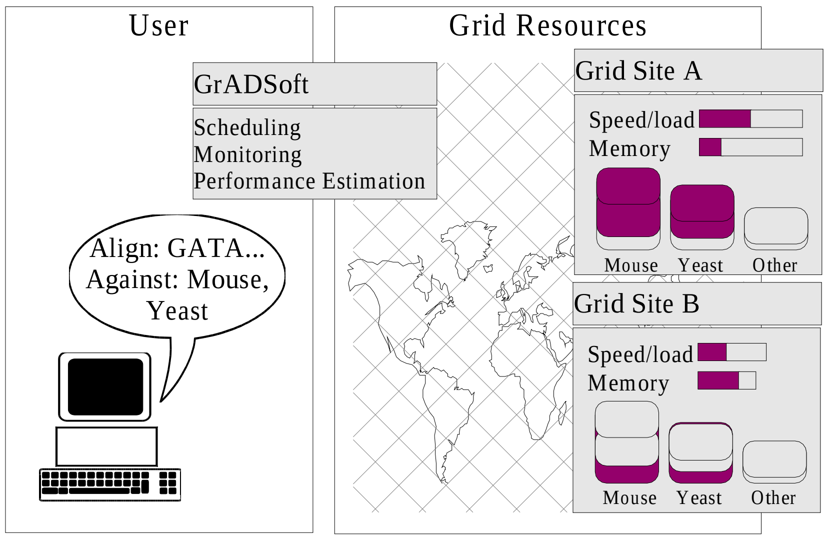

Fig. 2. An overview of FASTA running on the Grid; system status and database locality are used in making scheduling decisions

strings and the size of reference databases. Since there is no worker-to-worker communication in this application, the only communication time to be estimated was the time to distribute the queries to the worker nodes and collect results. This model has been designed to extrapolate beyond the parameters of the original experiments, but it can only be expected to be accurate within those parameters.

A *mapper* is an necessary complement for the performance model. For a given set of resources, the mapper allocates different amounts of work to the grid nodes in order to minimize the overall execution time and cover all the desired reference databases. The amount of work allocated depends on which databases are available on each node and the status of the node and the network. A linear approximation to the performance model is used to estimate the execution time for different data distributions. A freely available linear solver [32,20] was used to optimize the data allocation so as to minimize the execution time. Figure 2 shows the GrADSoft framework using data locality and resource information in order to make scheduling decisions.

Scheduling in GrADSoft works by presenting trial sets of eligible grid nodes to the application specific performance model in order to obtain an estimated execution time. The performance model calls the mapper to define a data distribution over the nodes, and then returns the estimated time using this data distribution and the current resource status. The set of nodes which have then lowest estimated execution time for the problem are used for the final execution. The scheduler generates the trial sets of nodes using deterministic greedy orderings of all available nodes along with some prior network knowledge so that nodes within a single cluster tend to be presented together. Other schedulers can also be used in the GrADSoft framework, for example, an exhaustive search scheduler exists and experiments have been performed using a simulated annealing scheduler [34]. Several papers have discussed details of the scheduling in GrADSoft [11,10,8].

The GrADSoft scheduler generates a static schedule for its applications by allocating the workers different portions of the reference databases. For master-worker applications, self-scheduling and its variants can result in faster execution times for many applications (see [28] for a study of master-worker scheduling). However, Pearson's original MPI master-worker implementation was not designed for static scheduling, and since we are trying to demonstrate the ease with which pre-existing codes can be grid-enabled by GrADSoft, we have not changed that. Additionally, the data-locality constraint in our implementation makes self-scheduling more complicated. Since the reference databases may not exist at all workers, when a self-scheduled worker requests additional work, it may be better for it to handle certain databases rather than others.

6 Performance Contracts and Execution

A key feature of the GrADS architecture is the performance contract which specifies an expected execution performance to be obtained on a set of grid resources. A performance contract can be developed using the application specific performance model which takes into account the capabilities and current state of the grid resources. Since we are dealing with a changing grid environment, there are many circumstances that may cause an application to violate its performance contract. Some possible reasons for failing performance contracts could be that other processes have been launched on the nodes, or the communication links have become crowded.

In the program preparation phase, the GrADSoft binder component edits the FASTA binary to enable performance monitoring. Calls to a performance measurement and monitoring tool AutoPilot [27,26] are inserted into the binary. These calls spawn a parallel thread which reports the status of the execution to an external AutoPilot manager program. AutoPilot monitors CPU performance information in addition to other message passing measures. The performance model and mapper specific to the FASTA application are loaded for use by the GrADSoft scheduler.

During the program execution phase, GrADSoft uses the GrADS runtime information system to obtain current information about available grid resources. The Scheduler uses this information with the performance model and mapper to generate a near-optimal schedule for the application. The application is then launched on the selected grid resources, with a master process being run on the first host and worker processes being run on all the other hosts. The master process uses information from the mapper to inform each worker which portions of what reference databases it should load into memory. For each query sequence, the master sends the sequence to each worker, and collects the replies from the worker. The master collates and prepares the results for the user.

While the application runs, the AutoPilot monitor thread reports the status to the AutoPilot manager. The external AutoPilot manager can be used to present the user with various views of the performance measures in order to determine if the execution is progressing as expected.

7 Summary and Conclusions

The goal of this work was to demonstrate that the GrADS architecture can be adapted to handle applications with data-locality constraints, and that these applications can be grid-enabled with ease.

GrADSoft greatly reduces the burden on the end user of finding, selecting and using the appropriate grid resources for their application. Porting and running FASTA, an master-worker sequence-matching application, was greatly simplified. A performance model and mapper had to be constructed for the application and minor changes were made to the FASTA code, in order to have the workers use local databases rather than have the master distribute all the data. After that, GrADSoft was able to handle all the details of scheduling, executing and monitoring the application.

The GrADS architecture provides a plausible method for providing large amounts of computing power to applications on demand. In this demonstration, the logistic tasks of scheduling, moving the application to the site of the data and gathering the results are handled by the framework. With the rapid growth of large, distributed data collections in fields such as biology, astronomy, and physics, the techniques described here could find wide applicability.

References

[1] Sudesh Agrawal, Jack Dongarra, Keith Seymour, and Sathish Vadhiyar. NetSolve: Past, present, and future — A look at a Grid enabled server. In F. Berman, G. Fox, and A. Hey, editors, *Grid Computing: Making the Global Infrastructure a Reality.* Wiley, New York, NY, USA, 2003.

[2] Gabrielle Allen, David Angulo, Ian Foster, Gerd Lanfermann, Chuang Liu, Thomas Radke, Ed Seidel, and John Shalf. The Cactus Worm: Experiments with dynamic resource discovery and allocation in a Grid environment. *International Journal of High Performance Computing Applications*, 15(4):345–358, 2001.

[3] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215:403–410, 1990.

[4] APBioGrid (Asia Pacific BioGrid) web site `http://www.mygrid.org.uk/`.

[5] Fran Berman, Andrew Chien, Keith Cooper, Jack Dongarra, Ian Foster, Dennis Gannon, Lennart Johnsson, Ken Kennedy, Carl Kesselman, John Mellor-Crummey, Dan Reed, Linda Torczon, and Rich Wolski. The GrADS Project: Software support for high-level Grid application development. *International Journal of Supercomputer Applications*, 15(4):327–344, 2001.

[6] Japanese BioGRID web site `http://www.biogrid.jp/`.

[7] Wahid Chrabakh and Rich Wolski. GridSAT: A Chaff-based Distributed SAT Solver for the Grid. In *SC2003: Igniting Innovation. Phoenix, AZ, November 15–21, 2003*. ACM Press and IEEE Computer Society Press, 2003.

[8] Holly Dail. A modular framework for adaptive scheduling in Grid application development environments. Master's thesis, University of California at San Diego, March 2002. Available as UCSD Tech. Report CS2002-0698.

[9] Holly Dail, Fran Berman, and Henri Casanova. A decoupled scheduling approach for Grid application development environments. *Journal of Parallel and Distributed Computing*, 2003. To appear.

[10] Holly Dail, Henri Casanova, and Francine Berman. A decoupled scheduling approach for the GrADS program development environment. In *Proceedings of Supercomputing Conference*, November 2002.

[11] Holly Dail, Otto Sievert, Fran Berman, Henri Casanova, Asim YarKhan, Sathish Vadhiyar, Jack Dongarra, Chuang Liu, Lingyun Yang, Dave Angulo, and Ian Foster. Scheduling in the Grid Application Development Software Project. In Jarek Nabrzyski, Jennifer M. Schopf, and Jan Weglarz, editors, *Grid Resource Management: State of the Art and Future Trends*, International Series in Operations Research & Management Science. Kluwer Academic Publishers Group, 2003.

[12] The FASTA package of sequence alignment programs at `ftp://ftp.virginia.edu/pub/fasta`.

[13] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2):115–128, Summer 1997.

[14] D. Gilbert, Y. Ugawa, M. Buchhorn, T.T. Wee, A. Mizushima, H. Kim, K. Chon, S. Weon, J. Ma, Y. Ichiyanagi, D.M. Liou, S. Keretho, and S. Napis. Bio-Mirror

project for public bio-data distribution. In *Bioinformatics*, volume 20, pages 2338–40. Nov 2004.

[15] Jean-Pierre Goux, Sanjeev Kulkarni, Michael Yoder, and Jeff Linderoth. Master-Worker: An enabling framework for applications on the computational grid. *Cluster Computing*, 4(1):63–70, 2001.

[16] Andrew S. Grimshaw, William A. Wulf, and the Legion team. The Legion Vision of a Worldwide Virtual Computer. *Communications of the ACM*, 40(1):39–45, January 1997.

[17] K. Kennedy, M. Mazina, J. Mellor-Crummey, K. Cooper, L. Torczon, F. Berman, A. Chien, H. Dail, O. Sievert, D. Angulo, I. Foster, R. Aydt, D. Reed, D. Gannon, L. Johnson, C. Kesselman, J. Dongarra, S. Vadhiyar, and R. Wolski. Toward a framework for preparing and executing adaptive grid programs. In *16th International Parallel and Distributed Processing Symposium (IPDPS '02 (IPPS and SPDP))*, pages 171–171. IEEE, April 2002.

[18] W. Li, R. Byrnes, J. Hayes, V. Reyes, A. Birnbaum, A. Shabab, C. Mosley, D. Pekurowsky, G. Quinn, I. Shindyalov, H. Casanova, L. Ang, F. Berman, M. Miller, and P. Bourne. The Encyclopedia of Life Project: Grid Software and Deployment. *Journal of New Generation Computing on Grid Systems for Life Sciences*, 2004.

[19] Chuang Liu, Lingyun Yang, Ian Foster, and Dave Angulo. Design and evaluation of a resource selection framework for Grid applications. In *Proceedings of the 11th IEEE Symposium on High-Performance Distributed Computing*, July 2002.

[20] The linear programming package lp_solve at `ftp://ftp.es.ele.tue.nl/pub/lp_solve`.

[21] MyGrid web site `http://www.mygrid.org.uk/`.

[22] North Carolina BioGRID web site `http://www.ncbiogrid.org/`.

[23] S. B. Needleman and C. D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *J. Mol. Biol.*, 48:443–453, 1970.

[24] W.R. Pearson and D.J. Lipman. Improved tools for biological sequence comparison. *Proceedings of the National Academy of Sciences of the United States of America*, 85:2444–2448, 1988.

[25] Antoine Petitet, Susan Blackford, Jack Dongarra, Brett Ellis, Graham Fagg, Kenneth Roche, and Sathish Vadhiyar. Numerical libraries and the Grid. *The International Journal of High Performance Computing Applications*, 15(4):359–374, November 2001.

[26] Randy L. Ribler, Huseyin Simitci, and Daniel A. Reed. The Autopilot performance-directed adaptive control system. *Future Generation Computer Systems*, 18(1):175–187, September 2001.

[27] R.L. Ribler, J.S. Vetter, H. Simitci, and D.A. Reed. Autopilot: Adaptive Control of Distributed Applications. *Proceedings of the 7th IEEE Symposium on High-Performance Distributed Computing*, July 1998.

[28] Gary Shao. *Adaptive Scheduling of Master/Worker Applications on Distributed Computational Resources*. PhD thesis, University of California at San Diego, May 2001.

[29] T. F. Smith and M. S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147:195–197, 1981.

[30] Neil Spring and Rich Wolski. Application level scheduling of gene sequence comparison on metacomputers. In *Proceedings of the 12th International Conference on Supercomputing*, pages 141–148. ACM Press, 1998.

[31] Frederik Vraalsen, Ruth A. Aydt, Celso L. Mendes, and Daniel A. Reed. Performance Contracts: Predicting and monitoring Grid application behavior. In *Proceedings of the 2nd International Workshop on Grid Computing*, Nov 2001.

[32] H.P. Williams. *Model Building in Mathematical Programming*. Wiley, Chichester, New York, second edition, 1995.

[33] Rich Wolski, Neil T. Spring, and Jim Hayes. The Network Weather Service: A distributed resource performance forecasting service for metacomputing. *Future Generation Computer Systems*, 15(5–6):757–768, October 1999.

[34] Asim YarKhan and Jack J. Dongarra. Experiments with scheduling using simulated annealing in a Grid environment. *Lecture Notes in Computer Science*, 2536:232–242, 2002.