Combining Workstations and Supercomputers to Support Grid Applications: The Parallel Tomography Experience

Shava Smallen* Walfredo Cirne* Jaime Frey¶ Francine Berman* Rich Wolski†
Mei-Hui Su§ Carl Kesselman§ Steve Young‡ Mark Ellisman‡

* Computer Science and Engineering Department
University of California, San Diego
$[ssmallen, walfredo, berman]$ $@cs.ucsd.edu$

¶ Department of Computer Science
University of Wisconsin
$jfrey@cs.wisc.edu$

† Department of Computer Science
University of Tennessee
$rich@cs.utk.edu$

§ Information Sciences Institute
University of Southern California
$[mei, carl]$ $@isi.edu$

‡ National Center for Microscopy and Imaging Research
University of California, San Diego
$mark@ncmir.ucsd.edu$

Abstract

Computational Grids are becoming an increasingly important and powerful platform for the execution of large-scale, resource-intensive applications. However, it remains a challenge for applications to tap into the potential of Grid resources in order to achieve performance. In this paper, we illustrate how work queue applications can leverage Grids to achieve performance through ***coallocation****. We describe our experiences developing a scheduling strategy for a production tomography application targeted to Grids that contain both workstations and parallel supercomputers.*

Our strategy uses dynamic information exported by a supercomputer's batch scheduler to simultaneously schedule tasks on workstations ***and*** *immediately available supercomputer nodes. This strategy is of great practical interest because it combines resources available to the typical research lab: time-shared workstations and CPU time in remote space-shared supercomputers. We show that this strategy improves the performance of the tomography application compared to traditional scheduling strategies, which target the application to either type of resource alone.*

1. Introduction

The aggregation of heterogenous resources into a Computational Grid [9] provides a powerful platform for the execution of large-scale resource-intensive applications. The simultaneous use of heterogeneous resources can greatly improve the performance of many applications, and permits researchers to run applications at the very large problem sizes critical to the discovery of new results. Although we are gaining considerable experience in the development of infrastructures which integrate distributed, heterogeneous resources, we have less experience developing applications which can leverage the distributed resources of the Grid to improve performance.

One application which has profited from leveraging the processing power of the Computational Grid is the Parallel Tomography (GTOMO) application being used in production at the National Center for Microscopy and Imaging Research (NCMIR). GTOMO is an embarrassingly-parallel application implemented with a work queue scheduling strategy. It uses Globus [10] services to perform a 3-D reconstruction from a series of images produced by NCMIR's electron microscope. As is the case with many laboratories, NCMIR owns a limited number of workstations (which are used as desktop machines and as a platform for parallel processing) and has access to supercomputer time. *In this*

This research was supported in part by NSF grants ASC-9701333 and ASC-9318180, DoD Modernization contract 9720733-00, NPACI/NSF award ASC-9619020 and Cooperative Agreement ANI-9807479, NIH/NCR grants RR04050 and RR08605, CAPES grant DBE2428/95-4, and DARPA/ITO under contract #N66001-97-C-8531.

paper, we describe a coallocation strategy for using both supercomputers and interactive workstation clusters to improve the execution performance of GTOMO within the context of a typical lab environment.

The scheduling strategy for GTOMO works at the application-level to target the application to both interactive workstation clusters and supercomputers. In an interactive workstation cluster, typically a *time-shared* computational platform, jobs begin execution immediately but share the CPU and network with other competing processes. In contrast, job submissions to a supercomputer, typically a *space-shared* computational platform, must wait in a batch queue until the desired number of the machine's processors become available for dedicated use. The time an application spends waiting in the queue impacts its *turnaround time*, the time elapsed from the submission of the application by the user until all of the results are available. Because the queue wait time can be quite lengthy [20], an application's turnaround time can be relatively large compared to its execution time.[1] Furthermore, the queue wait times make it difficult to use supercomputers and workstations concurrently, a strategy that could increase the processing power available to an application. Our strategy avoids unpredictable queue time delays by adaptively submitting requests to the supercomputer that can start running immediately.

The adaptive scheduler developed for GTOMO is framed as an AppLeS [2]. An AppLeS application scheduler integrates with the target application to develop a schedule for deploying the application in a shared, dynamic Grid environment [3, 23, 22]. The scheduler makes predictions of the performance the application may experience on prospective resources at execution time. Using these predictions, a potentially performance-efficient schedule for the application is identified and deployed. We developed a simple and effective coallocation strategy for the GTOMO AppLeS which targets both supercomputers and interactive workstations. Our experiments show that the GTOMO AppLeS coallocation strategy improves the turnaround time of the application over strategies which target either interactive workstations alone or a parallel supercomputer alone. We believe that the GTOMO AppLeS coallocation strategy will be effective for other work queue applications as well.

The next section provides a brief description of GTOMO. Section 3 describes our coallocation strategy for scheduling GTOMO over workstations and supercomputers. Section 4 presents the results of comparing our strategy against other scheduling alternatives. Section 5 discusses related work. Section 6 concludes the paper and discusses future work.

[1] In practice, queue times may range from seconds to days.

2. GTOMO Structure

Tomography allows for the reconstruction of the 3-D structure of an object based on 2-D projections through it taken at different angles. Electron microscopy is a classical use for tomography. Biological specimens on the cellular and sub-cellular level are viewed with an electron microscope and their images are recorded at a number of different angles. These images are then aligned and reconstructed into 3-D volumes using analytic and iterative tomographic techniques [18].

Reconstructing a typically sized volume using a simple algorithm (filtered back-projection) currently takes several hours on a workstation. NCMIR researchers have been interested in increasing the computation speed of the reconstruction for two reasons. First, they want to make use of more elaborate tomographic algorithms, which produce more refined 3-D volumes. These algorithms are more computationally intensive than the algorithms currently used. Second, NCMIR is interested in *on-line* tomography where the volume is rendered while the biologist is still collecting data on the microscope. This provides immediate feedback about the specimen being viewed and thus may prompt the researcher to change the experiment as a whole, or just some parameters of it (e.g., orientation and/or number of projections). For this to be useful, a rough reconstruction would have to finish in 5 to 10 minutes. No single processor can achieve this presently, which led the NCMIR researchers to explore parallelism.

The tomography application is highly amenable to parallelism. Because specimens are only rotated about a single axis as images are acquired, for any slice orthogonal to the axis of rotation, all information for that slice falls onto a single line on each of the projections (see Figure 1). More importantly, any such slice can be reconstructed independently of projection information for the rest of the volume. This makes the reconstruction embarrassingly parallel. Therefore, the the tomographic reconstruction can naturally be implemented as a *work queue*. In our implementation, we use Globus services to support efficient application execution within a heterogeneous, distributed environment.

The structure of GTOMO is depicted in Figure 2. There are four types of application processes: *driver, reader, writer*, and *ptomo*. The driver controls the work queue: it assigns one work unit or *slice* to a free ptomo until no more slices remain. The driver is invoked by the user and starts up the other processes. The reader and writer are I/O processes and hence have direct access to the user file system. The reader reads input files off the disk and sends them to the ptomos for processing. The writer receives output files from ptomos and writes them to disk. Note that the reader and writer enable GTOMO to run across different file system domains. The ptomo receives input files from a

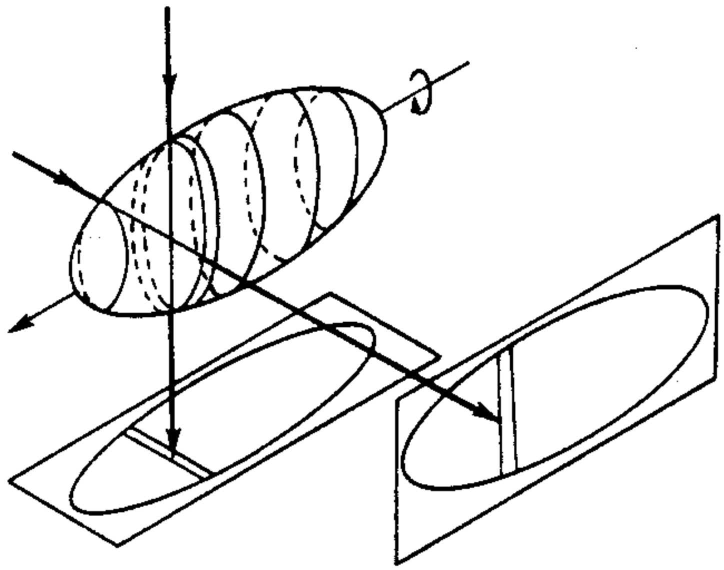


Figure 1. Projection geometry relating to a single-axis tilting experiment (from [12])

reader, does all the computational work, and sends output to a writer. In this study, we use one reader, one writer, and any number of ptomos. Due to the multi-threaded nature of Globus' Nexus communications library, one reader can service I/O requests for many ptomos simultaneously, and the same applies to the writer.

3. Scheduling GTOMO

Generally speaking, the set of potential resources available to GTOMO consists of workstations $w_1, ..., w_\omega$ and supercomputers $s_1, ..., s_\sigma$. A request to run a process p on workstation w causes p to start immediately, but p time-shares w with other processes. To use a supercomputer s, one has to specify how many processors n will execute copies of p and for how long t. The n copies of p do not necessarily start immediately; they might wait in the queue for an indeterminate amount of time until n nodes become available for t seconds. However, supercomputer processes run over dedicated resources once they are acquired.

Scheduling a GTOMO job consists of (i) choosing the requests to send to both supercomputers and workstations, and (ii) assigning work for the ptomos. For (ii), we use the work queue strategy shown in Figure 2 that assigns work on demand. For the first, we have to determine performance-efficient values of n and t for each available supercomputer s. Our goal is to select n and t in a way that minimizes GTOMO's turnaround time. Note that difficulty in predicting supercomputer queue wait times make it difficult to find an optimal n and t [8, 20, 14]. We avoid the queue time prediction problem by using supercomputer nodes that are immediately available. Therefore, we minimize the turnaround time of GTOMO by scheduling its execution at once on workstations and any immediately available supercomputer nodes.

We assume that the supercomputer scheduler can provide us with the maximum values of n and t for which execution can begin immediately. In our implementation, this information is supplied by the showbf command provided with the Maui Scheduler [15], a scheduler available for the IBM SP2. The showbf command returns a set of *backfill windows*, $b_1, ..., b_\theta$. Each $b_i = (n, t)$ where n nodes are available for immediate execution for the next t seconds.

The GTOMO AppLeS scheduler uses the following algorithm to schedule the ptomos:

```
for i = 1 to σ:
  b = showbf(s_i);
  for j = 1 to θ:
    start b_j.n ptomos on s_i for time b_j.t
for i = 1 to ω:
  start ptomo on w_i
```

Therefore, if backfill windows are available on any of the supercomputers, the job will be coallocated on those idle supercomputer nodes and workstations. If no backfill windows are returned by any of the supercomputers, the job

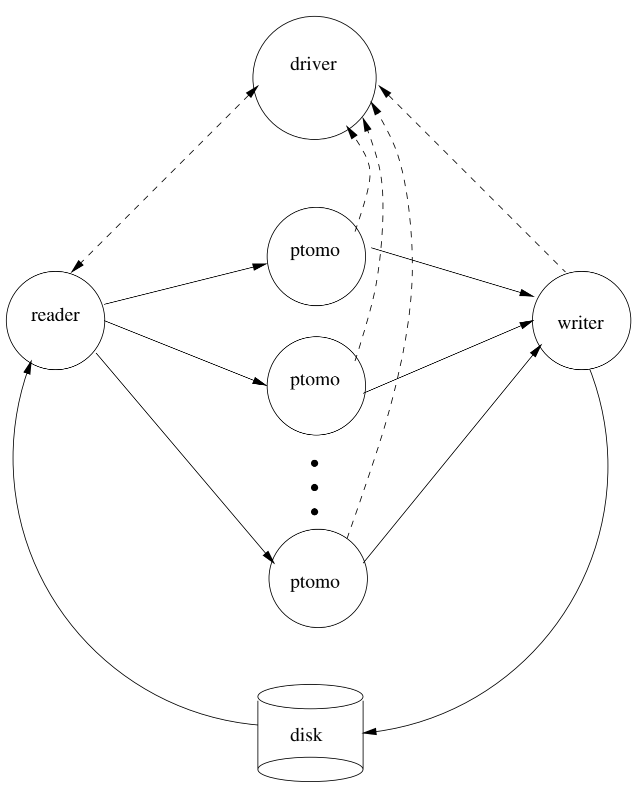


Figure 2. Application components of GTOMO. Solid lines represent transfer of input and output. Dotted lines denote control connections

will run only on workstations. The reader is scheduled on the machine where the input data is located and the writer is scheduled on the machine where the output data will be placed.

Note that the nodes immediately available in the SP2 may not be available for the full duration of the application. Therefore, the GTOMO AppLeS scheduler has to cope with ptomo processes that detach themselves from the application before execution has completed. We have added a fault recovery mechanism to GTOMO, which enables us to treat this problem as a ptomo failure. Whenever a ptomo fails, the slice it was processing is returned to the work queue. We can use such a simple scheme because processing a slice has no side effects. The advantage of reducing this problem to fault recovery is, of course, that it also covers real faults.

4. Experimental Results

We denote the GTOMO AppLeS scheduling strategy as *SP2Immed/WS* since it adaptively combines both the immediately available SP2 nodes and workstations. In order to ascertain how this strategy performs, we compared it against other possible scheduling strategies: using workstations only (*WS*), using only the nodes that are immediately available in the SP2 (*SP2Immed*), and requesting a predetermined number of nodes in the SP2 and probably waiting for them in the queue (*SP2Queue*). WS and SP2Queue respectively are the standard ways to use a cluster of workstations and a parallel supercomputer.

We ran experiments on a cluster of 7 workstations available in the Parallel Computation Laboratory (PCL) at U.C. San Diego and on the San Diego Supercomputer Center's SP2, one of the supercomputers available to NCMIR scientists. The PCL workstation cluster includes one 200 MHz UltraSPARC 2, a 110 MHz Sparc 5, a 85 MHz Sparc 5, and four 400 MHz Pentium IIs. The workstations are connected by a mixture of 10 and 100 Mbit/s ethernet subnets. The SP has 128 thin node POWER2 processors running at 160 MHz where processor pairs are interconnected by a 110 MB/s bidirectional network [21]. Other users were present on all resources during the experiments. Our dataset consisted of 300 slices; each input slice was 238 KB and the output slice was 1.2 MB.

We note that it is problematic to design experiments which compare multiple scheduling strategies under the same load and queue conditions for multi-user production environments. In such environments, the load and availability of resources change over time, so reproducibility of the same ambient load conditions is generally not an option. In contrast, it is possible to achieve reproducibility using simulation, but it may be difficult to represent dynamic load variation in complex heterogeneous systems authentically.

For our experiments, we performed sets of runs of SP2Immed, SP2Immed/WS, WS, and SP2Queue back-to-back [2] hoping that experiments within the same set would enjoy roughly similar load conditions. Moreover, we monitored the number of free nodes in the SP2 and used this information to discard execution sets in which the nodes available to SP2Immed/WS and SP2Immed differed by more than two. In this case, we considered the load conditions for strategies in the same set to be different. This was the case in 37 (out of 100) experiment sets. We therefore ended up with 63 valid experiment sets.

There are two other details to note in the design of our experiments. First, when we use only the resources immediately available in the SP2, it might be that there are no nodes available to execute the application. In this case, we did not run the set of experiments until the necessary resources became available. This happened 9 times out of 63 attempts. Notice that by excluding this retry time, we present optimistic turnaround times for the SP2Immed method. A user using this method would have experienced longer delays.

Second, we needed to decide on n and t when we used the SP2 in the traditional way (i.e., SP2Queue). Note that the determination of the best n requires an accurate queue time prediction. Since such predictions are not available, we rotated among values of n likely to be used by GTOMO users: 8, 16, and 32 nodes. We then executed benchmarks on the SP2 to determine the average processing time of one slice, t_b. This enabled us to conservatively determine t given n (a conservative estimate is needed because a job is killed when its execution exceeds t) using the following:

$$t = 2 \times \frac{t_b \times number\ of\ slices}{n}$$

The results of the 63 experimental sets are partitioned into three groups using the number of nodes requested for SP2Queue: SP2Queue(8), SP2Queue(16), and SP2Queue(32). Figure 3 shows the results of the experiment sets in which SP2Queue used 8 nodes, Figure 4 shows the results for 16 nodes, and Figure 5 shows the results for 32 nodes. Figures 3-5(a) depict the turnaround times of the different strategies (WS, SP2Immed/WS, SP2Immed, and SP2Queue). Each set of bars in the figure depicts a set of four executions, one under each of the four strategies. Since several of the SP2Queue turnaround times did not fit on the graphs, Table 1 displays the turnaround times for the SP2Queue runs. The number of nodes SP2Immed and SP2Immed/WS acquired in each set of experiments is shown in Figures 3-5(b).

[2]We used a 5 minute interval between experiments to ensure that the Maui scheduler had time to update its availability information.

We see that the SP2Immed/WS strategy yielded the best performance in all cases except one (Figure 4, run 8). Further study indicated contention on the reader and writer due to the selection of too many ptomos (in this experiment set, we received the highest number of immediately available SP2 nodes). A future scheduler improvement would be to model the contention and incorporate it into the GTOMO AppLeS.

We also assess the variability of each strategy using the coefficient of variance, c_v, which measures the amount of variance relative to the mean [7]. It is defined as follows:

$$c_v = \frac{standard\ deviation}{mean}$$

The SP2Immed/WS strategy exhibited the lowest c_v in all groups of experiments. Table 2 shows the mean, coefficient of variance, minimum, and maximum values for each strategy in each group of experiments. Table 2(a) shows the results of the experiment sets where SP2Queue used 8 nodes, Table 2(b) shows the results for 16 nodes, and Table 2(c) shows the results for 32 nodes. As expected, SP2Queue's c_v is quite large due to the unpredictable wait times in the queue. While its turnaround time was sometimes close to SP2Immed/WS (544s for SP2Queue vs. 601s for SP2Immed/WS for the one time it beat SP2Immed/WS), its worst time was more than two orders of magnitude greater than SP2Immed/WS (88,323s for SP2Queue vs. 601s for SP2Immed/WS). Also, we note that the SP2Immed strategy had a high c_v in the SP2Queue(8) results due to the variability of number of nodes acquired. This variability was amortized in the SP2Immed/WS strategy because of the relatively low c_v of WS.

5. Related Work

The GTOMO code is also used in the Computed Microtomography (CMT) Project at Argonne National Laboratory (ANL) [26, 27]. In contrast to NCMIR, projections are collected from a x-ray source at the Advanced Photon Source (APS) located at ANL. Their work has focused on on-line tomography where data is collected at APS, transferred to a 128 node SGI Origin 2000 for processing, and then transferred back to the user for visualization. Currently, they are able to deliver a reconstructed image to the user within minutes after data acquisition has completed. The CMT and NCMIR versions of GTOMO are currently being integrated as part of the NPACI Telescience Alpha Project [25].

Application scheduling for Grids is a recent and very active area. Existing work has focused primarily on resource discovery and scheduling [4, 17, 10, 28] and coallocation

run	8 nodes	16 nodes	32 nodes
1	685.0235	591.2153	1293.9811
2	759.2754	581.2684	532.6011
3	698.2693	20483.5053	536.9106
4	3077.8569	723.0844	541.6972
5	701.3900	17268.0273	658.2000
6	708.8977	2097.6415	565.0041
7	691.5886	579.8207	27480.0918
8	1805.0212	543.7824	9868.9585
9	6163.5884	581.0620	2267.3595
10	687.5382	615.1581	614.1135
11	689.0494	793.9383	17735.0314
12	682.3540	9193.4053	39286.8783
13	700.8448	742.4905	34642.5641
14	4625.1468	2164.4710	1120.0531
15	1260.1495	621.3329	88322.6571
17	3249.5812	574.9321	1809.4320
18	710.8228	593.0303	664.4993
19	718.0481	1203.5411	6815.6301
20	721.3216	575.2712	607.1331
21	719.9383	33715.8070	29675.0315
22	707.4284	580.8794	
23	717.6290		

Table 1. Turnaround times for SP2Queue

strategy	mean	c_v	min	max
SP2Immed	1946.68	2.10	504.81	19694.46
SP2Immed/WS	437.99	0.17	346.71	554.50
WS	775.98	0.19	596.26	1133.44
SP2Queue	1430.94	1.05	682.35	6163.59

(a) SP2Queue(8) results

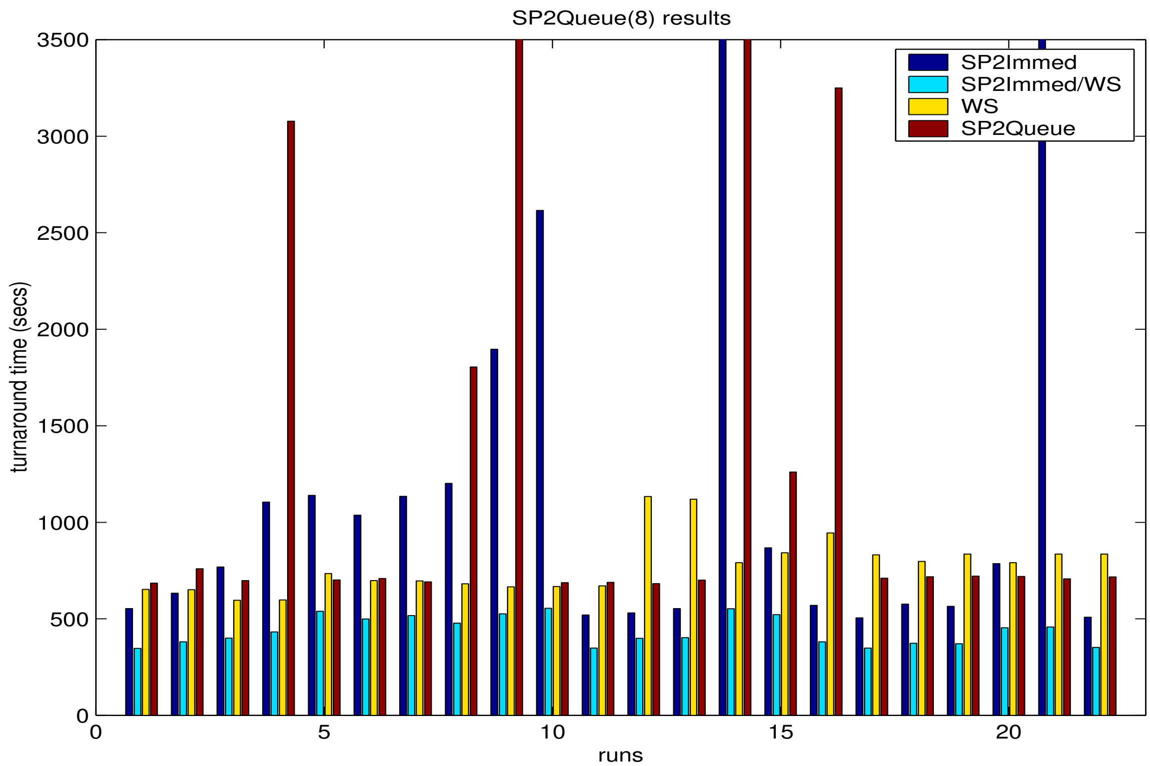
strategy	mean	c_v	min	max
SP2Immed	660.58	0.28	502.75	1105.61
SP2Immed/WS	402.31	0.17	342.03	600.98
WS	777.53	0.19	588.76	1127.03
SP2Queue	4515.41	1.94	543.78	33715.81

(b) SP2Queue(16) results

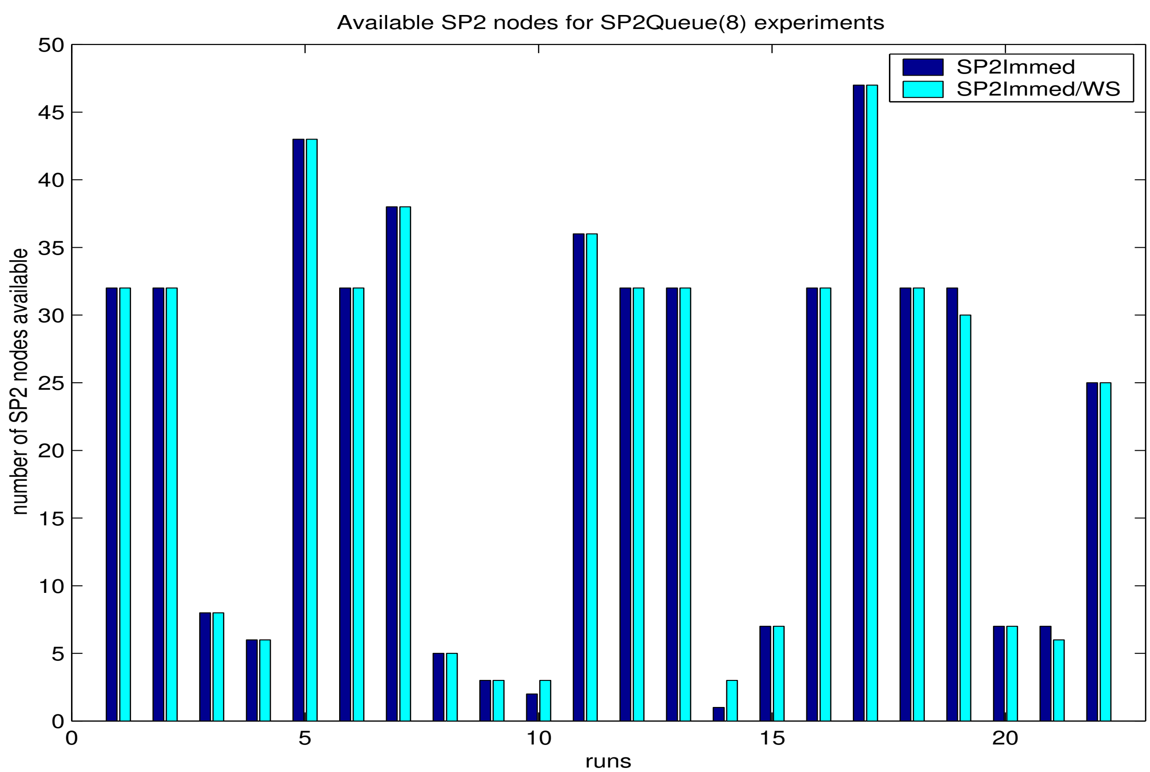
strategy	mean	c_v	min	max
SP2Immed	659.30	0.33	500.24	1262.72
SP2Immed/WS	397.37	0.12	342.70	519.13
WS	789.69	0.20	587.68	1128.41
SP2Queue	13251.89	1.66	532.60	88322.66

(c) SP2Queue(32) results

Table 2. Summary results of experiments

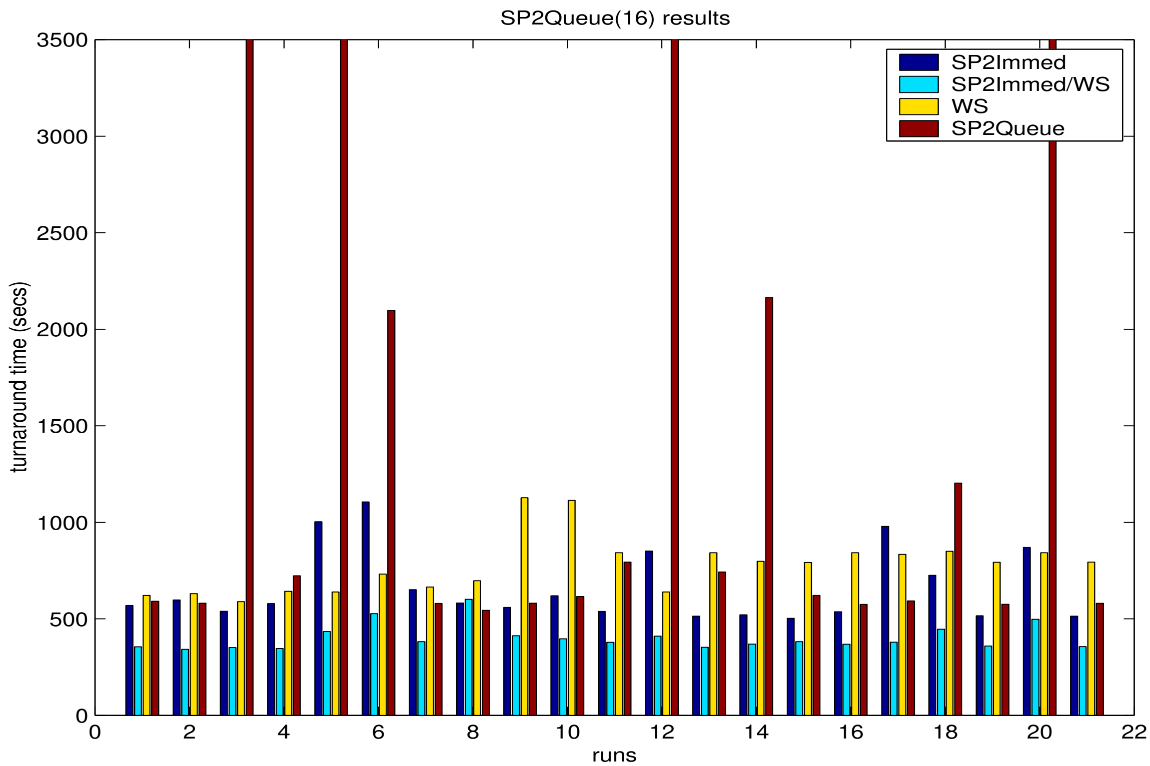


(a)

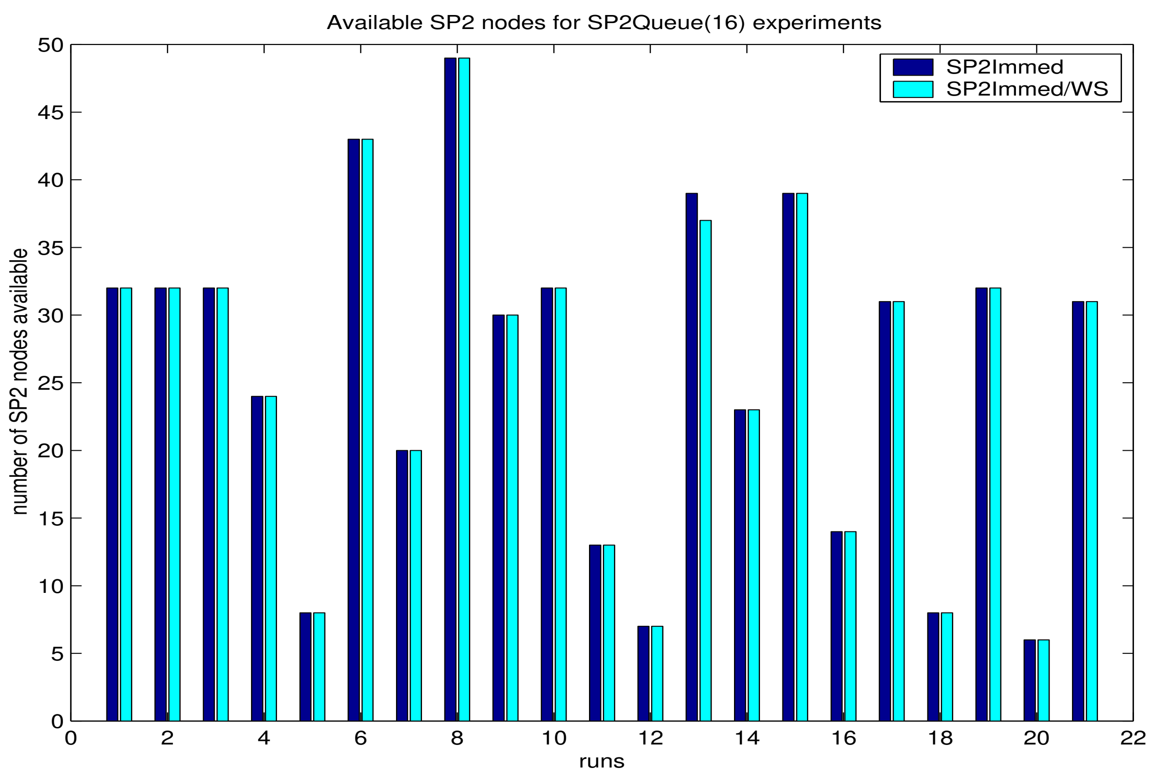


(b)

Figure 3. Experiment results when 8 nodes were requested for SP2Queue. (a) Turnaround time. (SP2Immed values too large to fit on graph: run 14 - 5067s, run 21 - 19694s) (b) Number of SP2 nodes used by SP2Immed/WS and SP2Immed.

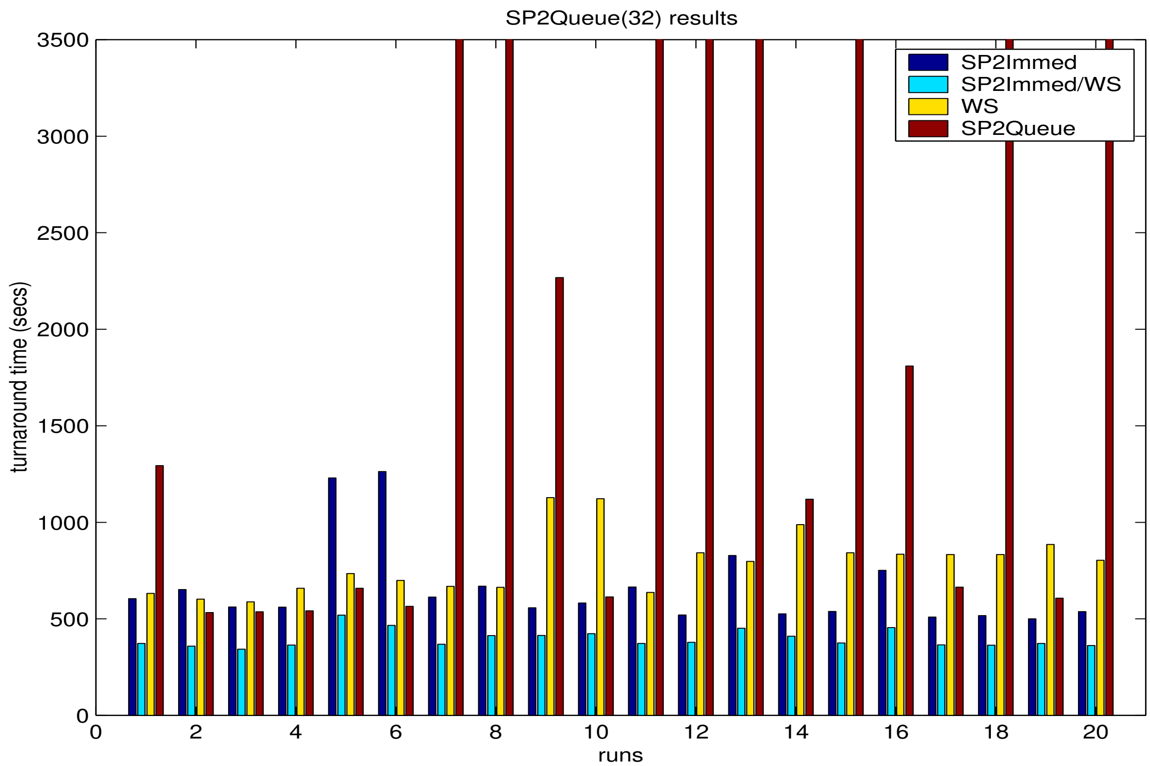


(a)

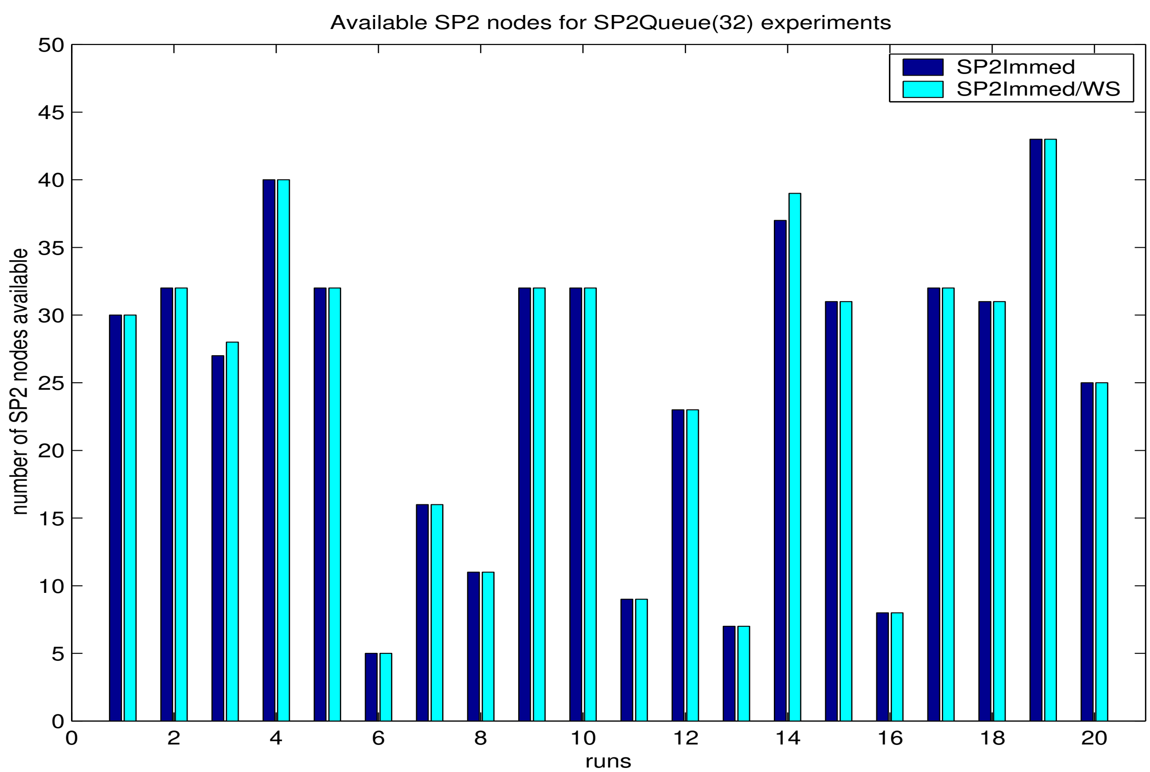


(b)

Figure 4. Experiment results when 16 nodes were requested for SP2Queue. (a) Turnaround time. (b) Number of SP2 nodes used by SP2Immed/WS and SP2Immed.



(a)



(b)

Figure 5. Experiment results when 32 nodes were requested for SP2Queue. (a) Turnaround time. (b) Number of SP2 nodes used by SP2Immed/WS and SP2Immed.

among workstations [3, 1, 19, 22, 23, 9]. The work reported herein extends the target domain for GTOMO by targeting both parallel supercomputers and interactive resources simultaneously.

6. Discussion and Conclusions

In this work, we show how to combine workstations and supercomputers to run GTOMO, a work queue application used in production at NCMIR. Our solution automatically selects all resources immediately available across the system. We leverage the Maui Scheduler to obtain information on immediately available SP2 nodes. This strategy has the advantage of not requiring predictions of how long requests wait in the supercomputer queue. Our experimental results show that the GTOMO AppLeS scheduling strategy consistently outperforms three other strategies that can be used for scheduling in an typical laboratory setting where researchers have access to a local cluster of workstations and supercomputer time.

We have learned three interesting lessons about Computational Grids in general as a result of this effort. First, the interface exported by the resource scheduler has great impact on application schedulers. In fact, we can implement our strategy in a very straightforward manner thanks to the Maui Scheduler's showbf command. On the other hand, the Maui Scheduler (as with other supercomputer schedulers, for that matter) precluded us from trying something more sophisticated due to the difficulty in predicting queue times for supercomputer requests. Emerging efforts such as S^3 [6], GARA [11], and more generally, the Grid Forum Scheduling Working Group [13] are working to change this.

Second, evaluating solutions for real applications running over production environments has proven to be difficult due to the impossibility of reproducing the system load and queue conditions for comparison runs. Others have encountered the same problem. Indeed, a simulation environment specifically targeted toward Grids such as the Bricks project [24], the MicroGrid [16], or the work described in [5] would be very useful.

Third, fault tolerance is likely to be even more important in Grid computing than it is in parallel computing. For our solution in particular, fault recovery was a natural way to deal with the time expiration of SP2 requests. In general, using autonomous and distributed resources increases the chance that some component of the application will fail.

The GTOMO AppLeS scheduler has been incorporated with the production version of GTOMO at NCMIR and is used daily be researchers. Current work involves extending the applicability of the scheduler to additional resources and different scenarios of the application.

Dedication and Acknowledgements

This paper is dedicated to Steve Young who was a cornerstone of this work. Steve was a respected scientist and treasured friend and we will miss him greatly.

We are grateful to the NPACI partnership and SDSC researchers for their assistance with this work. We benefited greatly from discussions with the AppLeS team, the Globus team, and our colleagues at NCMIR.

References

[1] D. Andersen, T. Yang, O. Ibarra, and O. Egecioglu. Adaptive partitioning and scheduling for enhancing WWW application performance. *Journal of Parallel and Distributed Computing*, 49:57–85, Feb 1998.

[2] AppLeS webpage at `http://apples.ucsd.edu`.

[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] H. Casanova and J. Dongarra. Netsolve: A network server for solving computational science problems. In *Proceedings of Supercomputing 1996*, 1996.

[5] H. Casanova, A. Legrand, D. Zagorodnov, and F. Berman. Using Simulation to Evaluate Scheduling Heuristics for a Class of Applications in Grid Environments. Technical Report RR1999-46, École Normale Supérieure de Lyon, LIP, 1999.

[6] W. Cirne and F. Berman. Application scheduling over supercomputers: A proposal. Technical Report UCSD-CS99-631, University of California, San Diego, October 1999.

[7] J. L. Devore. *Probability and Statistics for Engineering and the Sciences*. Duxbury Press, 1995.

[8] A. Downey. Using queue time predictions for processor allocation. In *3rd Workshop on Job Scheduling Strategies for Parallel Processing, in conjunction with IPPS'97*, 1997.

[9] I. Foster and C. Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, July 1998.

[10] I. Foster and C. Kesselman. The Globus Project: A Status Report. In *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, 1998.

[11] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy. Distributed resource management architecture that supports advance reservations and co-allocation. In *Intl Workshop on Quality of Service*, 1999.

[12] J. Frank and M. Radermacher. Three-dimensional reconstruction of nonperiodic macromolecular assemblies from electron micrographs. In J. K. Koehler, editor, *Advanced Techniques in Biological Electon Microscopy III*. Springer-Verlag, 1986.

[13] Grid Forum webpage at `http://www.gridforum.org/`.

[14] R. Gibbons. A historical application profiler for use by parallel schedulers. In *3rd Workshop on Job Scheduling Strategies for Parallel Processing, in conjunction with IPPS'97*, 1997.

[15] Maui Scheduler webpage at `http://www.mhpcc.edu/maui`.
[16] MicroGrid webpage at `http://www-csag.ucsd.edu/projects/grid/microgrid.html`.
[17] H. Nakada, M. Sato, and S. Sekiguchi. Design and implementations of ninf: towards a global computing infrastructure. *Future Generation Computing Systems*, 1999. Metacomputing Issue.
[18] G. Perkins, C. Renken, S. Young, S. Lamont, M. Martone, S. Lindsey, T. Frey, and M. Ellisman. Electron tomography of large multicomponent biological structures. *J. Struct.Biol.*, 120:219–227, 1997.
[19] G. Shao, R. Wolski, and F. Berman. Predicting the cost of redistribution in scheduling. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*, 1997.
[20] W. Smith, V. Taylor, and I. Foster. Using run-time predictors to estimate queue wait times and improve scheduler performance. In *5th Workshop on Job Scheduling Strategies for Parallel Processing, in conjunction with IPPS'99*, 1999.
[21] NPACI's SP webpage at `http://www.npaci.edu/SP`.
[22] N. Spring and R. Wolski. Application level scheduling of gene sequence comparison on metacomputers. *12th ACM International Conference on Supercomputing*, July 1998.
[23] A. Su, F. Berman, R. Wolski, and M. M. Strout. Using AppLeS to schedule a distributed visualization tool on the computational grid. *International Journal of Supercomputer and High-Performance Applications*, 1999.
[24] A. Takefusa, S. Matsuoka, H. Nakada, K. Aida, and U. Nagashima. Overview of a performance evaluation system for global computing scheduling algorithm. To appear in the 8th HPDC conference, 1999.
[25] Telescience webpage at `http://www.npaci.edu/Alpha/Telescience/`.
[26] G. von Laszewski, M.-H. Su, J. Insley, I. Foster, J. Bresnahan, C. Kesselman, M. Thiebaux, M. Rivers, S. Wang, B. Tieman, and I. McNulty. Real-time analysis, visualization, and steering of tomography experiments at photon sources. *Ninth SIAM Conference on Parallel Processing for Scientific Computing*, Apr 1999.
[27] Y. Wang, F. D. Carlo, I. Foster, J. Insley, C. Kesselman, P. Lane, G. von Laszewski, D. Mancini, I. McNulty, M.-H. Su, and B. Tieman. A quasi-realtime x-ray microtomography system at the Advanced Photon Source. *Proceedings of SPIE99*, 3772, 1999.
[28] J. Weissman. Gallop: The benefits of wide-area computing for parallel processing. *Journal of Parallel and Distributed Computing*, 54, Nov 1998.

Shava Smallen received a B.S. and is currently pursuing a M.S. at the Department of Computer Science and Engineering at the University of California, San Diego. Her current research interest is in scheduling on Computational Grids.

Walfredo Cirne is a Ph.D. student at the University of California, San Diego and an assistant professor at Brazil's Universidade Federal da Paraiba (currently on leave). He has previously worked on Machine Learning and Network Security. Now, his research efforts concentrate on scheduling on Computational Grids. In particular, he is investigating how application schedulers can use resources controlled by space-shared supercomputers. He received his B.S. and M.S from the Universidade Federal da Paraiba.

Jaime Frey is currently pursuing a Ph.D. at the Department of Computer Science at the University of Wisconsin. His current research interests include scheduling in distributed systems. He received his B.S. from the University of California, San Diego.

Francine Berman is a Professor of Computer Science and Engineering at the University of California, San Diego. She is also a Senior Fellow at the San Diego Supercomputer Center, Fellow of the ACM, and founder of the Parallel Computation Laboratory at UCSD. Her research interests over the last two decades have focused on parallel and distributed computation, and in particular the areas of programming environments, tools, and models that support high-performance computing. She received her B.A. from the University of California, Los Angeles, her M.S. and Ph.D. from the University of Washington.

Rich Wolski is an Assistant Professor in the Department of Computer Science at the University of Tennessee and a partner in the National Partnership for Advanced Computational Infrastructure. His research interests include parallel and distributed computing, on-line performance analysis techniques and software, compiler runtime system, and dynamic scheduling. He received his B.S. from the California Polytechnic University, San Luis Obispo and his M.S. and Ph.D. from the University of California at Davis/Livermore Campus.

Mei-Hui Su is a programmer at the Information Sciences Institute located at the University of Southern California. Her current research interest is in distributed parallel computing. She received her B.S. from the University of California, Berkeley.

Carl Kesselman leads a research group at the Information Sciences Institute located at the University of Southern California. His research focus has been in the development of new methods, tools, and programming environments for large-scale, high-performance computer systems. He received his B.S. from the University of Buffalo, his M.S. from the University of Southern California, and his Ph.D. from the University of California, Los Angeles.

Steve J. Young was a Specialist in Psychiatry and an Associate Director of the National Center for Microscopy and Imaging Research (NCMIR). Until his untimely passing on January 9th, 1999, he led the technology development team at NCMIR, contributing to the work presented here and nearly all projects at the Center. His research interests ranged from the neurophysiology of perception to the design of software to improve our ability to accurately represent complex relationships between biological systems. He was a unique man with remarkable talents. He was also a friend and colleague who shared his extensive knowledge freely enhancing the lives of all around him, including the authors of this paper. He earned his Undergraduate Degree at Berkeley and Ph.D. at the University of California, Los Angeles. He became a Professor of Physiological Psychology at the University of Colorado at Boulder before moving to UCSD where together with Ellisman he established the NCMIR.

Mark H. Ellisman is a Professor of Neurosciences and Bioengineering at the University of California, San Diego and Director of the National Center for Microscopy and Imaging Resource and the Center for Research on Biological Structure at UCSD. He leads the Neuroscience Thrust for the National Partnership for Advanced Computational Infrastructure and is a Senior Fellow at the San Diego Supercomputer Center. His research interests over the last three decades have focused on structure and function of the nervous system including the development of advanced imaging instruments and computational approaches for the refinement of data about and visualization of multiscale biological complexes. He received his B.A. from the University of California, Berkeley and M.A. and Ph.D from the University of Colorado at Boulder.