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[‡]

October 12, 1999

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 the potential of Grid resources in order to achieve performance. In this paper, we illustrate how applications can leverage Grids to achieve performance through coallocation. We describe our experiences developing a scheduling strategy for a real-life parallel tomography application targeted to Grids which contain both workstations and parallel supercomputers. Our strategy uses dynamic information exported by a supercomputer's batch scheduler to simultaneously schedule 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 parallel tomography application compared to traditional scheduling strategies, which target the application to either type of resource alone.*

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

[†]Department of Computer Science and Engineering, University of California San Diego (http://apples.ucsd.edu)

[‡]National Center for Microscopy and Imaging Research (http://www-ncmir.ucsd.edu)

[§]Department of Computer Science, University of Tennessee (http://www.cs.utk.edu/~rich)

[¶]Information Sciences Institute, University of Southern California (http://www.isi.edu)

1 Introduction

The aggregation of heterogenous resources into a Computational Grid [FK98a] provides a powerful platform for the execution of large-scale resource-intensive applications. The simulateneous 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 at the National Center for Microscopy and Imaging Research (NCMIR). GTOMO is a real-life, embarrassingly-parallel production application which uses Globus [FK98b] 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 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 address the different resource scheduling policies for both interactive workstation clusters and supercomputers. In interactive workstation clusters (time-shared machines), jobs begin executing immediately but share the CPU and network with other competing processes. In contrast, job submissions to a supercomputer (space-shared machine) must wait in a batch queue until the desired number of the machine's processors become available for dedicated use. The *turnaround time* [1] experienced by the application can be quite lengthy due to queue wait times [STF99], even if the execution time of the application is relatively short once it gains access to the machine.[2] The queue wait times make it difficult to use supercomputers with workstations simultaneously, a strategy that could increase the processing power available to an application. To avoid unpredictable queue time delays, the GTOMO scheduling strategy adaptively submits job requests to the supercomputer's batch scheduler such that wait time in the batch queue is avoided (i.e. finds idle nodes).

The adaptive scheduler developed for GTOMO is framed as an AppLeS [App]. An AppLeS application scheduler integrates with a target program to develop and deploy an application schedule customized for a shared, dynamic Grid environment [BWF+96, SBWS99, SW98]. The scheduler makes predictions of the performance of the application on prospective resources at execution time (typically using the Network Weather Service monitoring and prediction facility [WSH98]). Using these performance 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 application tasks to both supercomputers and/or interactive workstations based on the identification and use

[1] The time elapsed from the submission of the application by the user until all of the results are available.

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

of immediately available resources. Our experiments show that this coallocation strategy improves the execution performance of the application over strategies which target the application to interactive workstations alone and to a parallel supercomputer alone. We believe that our coallocation strategy will be effective for other coarse-grained parallel 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.

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 [PRY+97].

GTOMO is an embarrassingly parallel implementation of tomography which uses Globus services to run over a heterogeneous, distributed environment. The structure of GTOMO is depicted in Figure 1. There are four types of application processes: *driver, reader, writer*, and *ptomo*. The driver controls a work queue: it assigns one 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. This enables GTOMO to run across different file systems domains. The ptomo receives input files from a 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, although other process configurations are supported by our implementation. 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 for 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 (through space-sharing).

Scheduling GTOMO consists of (i) choosing the requests to send to supercomputers and workstations, and (ii) assigning work for the ptomos. For the latter, we use a simple work queue strategy that assigns work on demand. For the first, we have to determine performance-efficient values of n and t for each available supercomputer s.[3] Our goal is to select n and t in a way that minimizes GTOMO's turn-around time. Note that difficulty in predicting supercomputer queue time makes it difficult to find an optimal n and t [Dow97, STF99, Gib97].

We avoid the queue time prediction problem by using supercomputer nodes that are immediately available. We assume that the supercomputer scheduler can provide us with the maximum values of n and t for which execution can begin immediately. Our implementation uses the showbf command supplied by the Maui Scheduler [Mau] that controls the SDSC SP2 to provide this information. These n and t values are then used by the AppLeS scheduler.

Note that the nodes immediately available in the SP2 may not be available for the full duration of the application. Therefore, the AppLeS application 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 effect. 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 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 8 workstations available at the Parallel Computation Laboratory at UCSD and on the SDSC's SP2. The workstation cluster includes 2 200 MHz UltraSPARC 2s, a 110 MHz Sparc 5, a 85 MHz Sparc 5, and 4 400 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 each processor pair is connected together by a 110 MB/s bi-directional network [SP]. Other users were present on all resources during the experiments.

Note that it is problematic to design experiments which compare all scheduling strategies under the same load and queue conditions in real 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 not an option. For our experiments, we performed runs

[3] For the workstations, all we can do is to increase our nice value, something that is not going to increase the performance of the application.

of SP2Immed, SP2Immed/WS, WS, and SP2Queue back-to-back [4] 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 an execution set in which the nodes available to SP2Immed/WS and SP2Immed differed by more than two. (This was the case in 17 (out of 51) experiment sets.) We therefore ended up with 34 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 do not run the set of experiments until the necessary resources become available. This happened 6 times out of 34 attempts. Notice that by excluding this retry time, we present optimistic turn-around 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). We executed benchmarks on the SP2 to determine the processing time of one slice. This enabled us to conservatively determine t given n (a conservative estimate is needed because a job is killed when its execution exceeds t). Note that the determination of the best n requires accurate queue 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.

Our experiment results are grouped by the number of nodes SP2Queue used. Figure 2 shows the experiments in which SP2Queue used 8 nodes, Figure 3 shows the results for 16 nodes, and Figure 4 shows the results for 32 nodes. Figures 2-4(a) depict how the different strategies (WS, SP2Immed/WS, SP2Immed, and SP2Queue) performed. Each set of bars in the figure depicts four executions, one under each of the four strategies. The number of nodes SP2Immed and SP2Immed/WS acquired in each set of experiments is shown in Figures 2-4(b). The SP2Immed/WS strategy yielded the best performance in all cases except one (Figure 3, run 8). Note that the SP2Immed/WS strategy exhibited much less variance than SP2Queue and SP2Immed. As expected, SP2Queue had the greatest variance. While its turn-around 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 almost two orders of magnitude greater than SP2Immed/WS (27480s for SP2Queue vs. 369s for SP2Immed/WS).

5 Related Work

Both Nimrod and Condor have focused on the execution of collections of independent tasks in clustered workstation environments [ASGC95, LLM98]. In the Nimrod work, tasks are "experiments" in a parameter sweep application; in the Condor work, tasks are independent jobs. The tasks in GTOMO are processes in a scatter/gather structured application which receive input data from a reader process and send output data to a writer process. Note that Nimrod and Condor focus on increasing the throughput of the job collection whereas our goal is to increase the performance of a single application. This framework is important

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

for GTOMO because ultimately researchers would like to use it in quasi-real-time to provide rough 3-D images of the specimen while data is being acquired from the microscope.

Application scheduling for Grids is a recent and very active area. Existing work has focused primarily on resource discovery and scheduling [CD96, NSS99, FK98b, Wei98] and coallocation among workstations [BWF+96, AYIE98, SWB97, SW98, SBWS99, FK98a] 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 real-life coarse-grained parallel application. 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 results show that the GTOMO AppLeS scheduling strategy consistently outperforms three strategies researchers might use 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. Future developing efforts such as S^3 [CB99], GARA [FKL+99], and more generally, the Grid Forum Scheduling Working Group [GF] 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 [TMN+99] or the work desribed in [CLZB99] 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.

Note that our scheduler could be improved in several ways. First, we plan to extend our strategy to add processors that become available (both interactive and in the SP2) during execution. In addition, we plan to consider scaled versions of the program in which a single reader or writer is likely to become a bottleneck. We are currently developing a strategy that determines how many readers and writers are needed to avoid contention. Finally, we intend to integrate the AppLeS GTOMO scheduler with larger end-to-end telemicroscopy applications for which the tomography reconstruction presented here is a fundamental part.

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 benefitted greatly from discussions with the AppLeS team, the Globus team, and our colleagues at NCMIR.

References

[App] AppLeS webpage at `http://www-cse.ucsd.edu/groups/hpcl/apples`.

[ASGC95] D. Abramson, R. Sosic, J. Giddy, and M. Cope. Nimrod: A tool for performing parametised simulations using distributed workstations. In *Proc. 4th IEEE Symp. on High Performance Distributed Computing*, 1995.

[AYIE98] D. Andersen, Tao 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.

[BWF+96] Francine Berman, Richard Wolski, Silvia Figueira, Jennifer Schopf, and Gary Shao. Application level scheduling on distributed heterogeneous networks. In *Proceedings of Supercomputing 1996*, 1996.

[CB99] Walfredo Cirne and Fran Berman. Application scheduling over supercomputers: A proposal. Technical Report UCSD-CS99-631, University of California, San Diego, October 1999.

[CD96] Henri Casanova and Jack Dongarra. Netsolve: A network server for solving computational science problems. In *Proceedings of Supercomputing 1996*, 1996.

[CLZB99] 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.

[Dow97] Allen Downey. Using queue time predictions for processor allocation. In *3rd Workshop on Job Scheduling Strategies for Parallel Processing, in conjunction with IPPS'97*, 1997.

[FK98a] Ian Foster and Carl Kesselman, editors. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, July 1998.

[FK98b] Ian Foster and Carl Kesselman. The Globus Project: A Status Report. In *Proc. IPPS/SPDP '98 Heterogeneous Computing Workshop*, 1998.

[FKL+99] Ian Foster, Carl Kesselman, Craig Lee, Bob Lindell, Klara Nahrstedt, and Alain Roy. Distributed resource management architecture that supports advance reservations and co-allocation. In *Intl Workshop on Quality of Service*, 1999.

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

[Gib97] Richard 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.

[LLM98] M. Litzkow, M. Livny, and M. W. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference of Distributed Computing Systems*, 1998.

[Mau] Maui webpage at `http://wailea.mhpcc.edu/maui/`.

[NSS99] Hidemoto Nakada, Mitsuhisa Sato, and Satoshi Sekiguchi. Design and implementations of ninf: towards a global computing infrastructure. *Future Generation Computing Systems*, 1999. Metacomputing Issue.

[PRY+97] G.A. Perkins, C.W. Renken, S.J. Young, S.P. Lamont, M.E. Martone, S. Lindsey, T.G Frey, and M.H. Ellisman. Electron tomography of large multicomponent biological structures. *J. Struct.Biol.*, 120:219–227, 1997.

[SBWS99] Alan Su, Francine Berman, Richard Wolski, and Michelle Mills Strout. Using AppLeS to schedule a distributed visualization tool on the computational grid. *International Journal of Supercomputer and High-Performance Applications*, 1999.

[SP] NPACI's SP webpage at `http://www.npaci.edu/SP`.

[STF99] Warren Smith, Valerie Taylor, and Ian 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.

[SW98] Neil Spring and Rich Wolski. Application level scheduling of gene sequence comparison on metacomputers. *12th ACM International Conference on Supercomputing*, July 1998.

[SWB97] Gary Shao, Rich Wolski, and Fran Berman. Predicting the cost of redistribution in scheduling. In *Proceedings of the 8th SIAM Conference on Parallel Processing for Scientific Computing*, 1997.

[TMN+99] Atsuko Takefusa, Satoshi Matsuoka, Hidemoto Nakada, Kento Aida, and Umpei Nagashima. Overview of a performance evaluation system for global computing scheduling algorithm. To appear in the 8th HPDC conference, 1999.

[Wei98] Jon Weissman. Gallop: The benefits of wide-area computing for parallel processing. *Journal of Parallel and Distributed Computing*, 54, Nov 1998.

[WSH98] Rich Wolski, Neil T. Spring, and Jim Hayes. The network weather service: A distributed resource performance forecasting service for metacomputing. Technical report, University of California, San Diego, 1998.

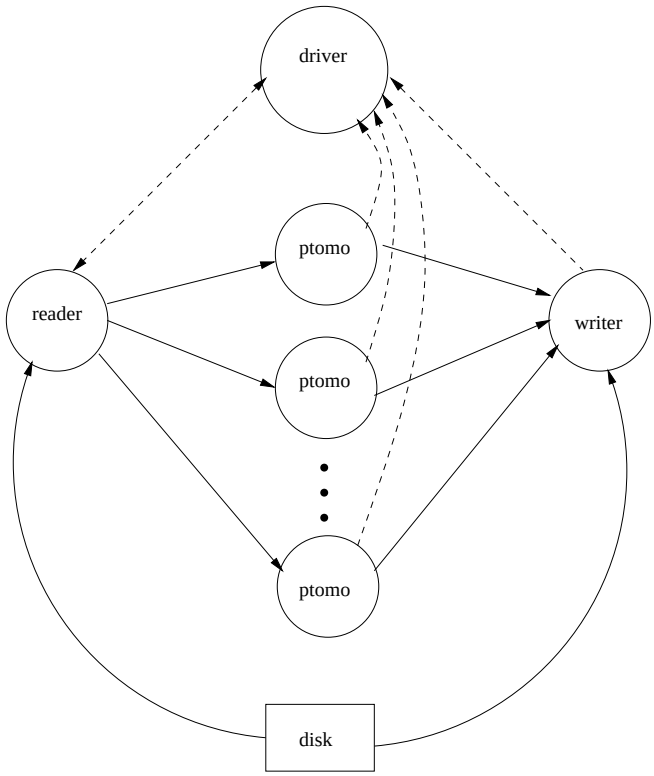
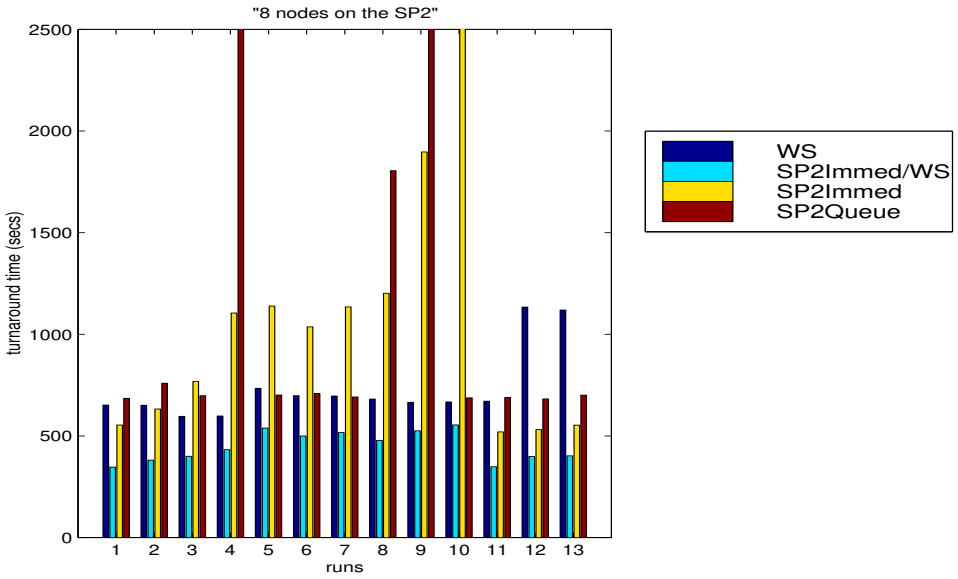
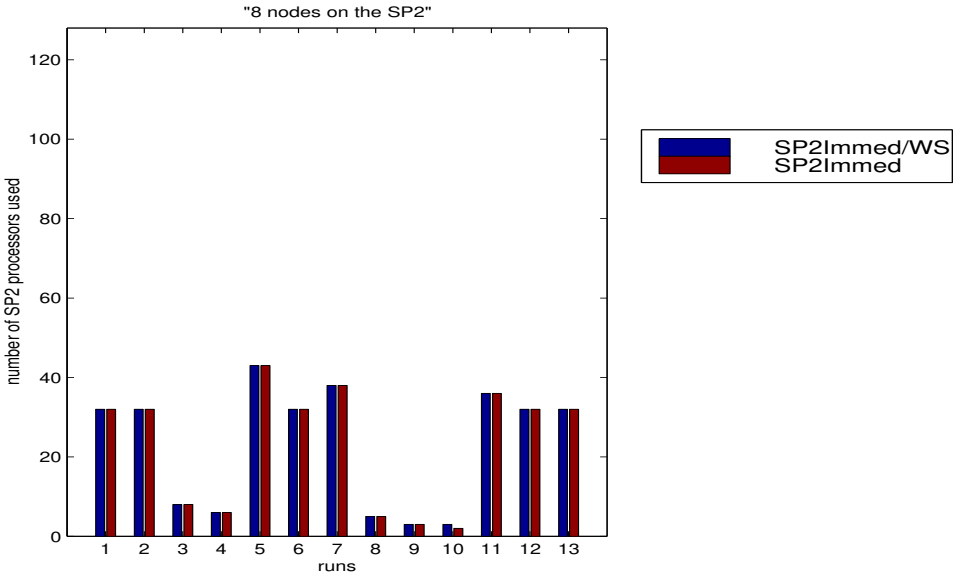


Figure 1: Application Components of GTOMO. Solid lines represent transfer of input and output. Dotted lines denote control connections

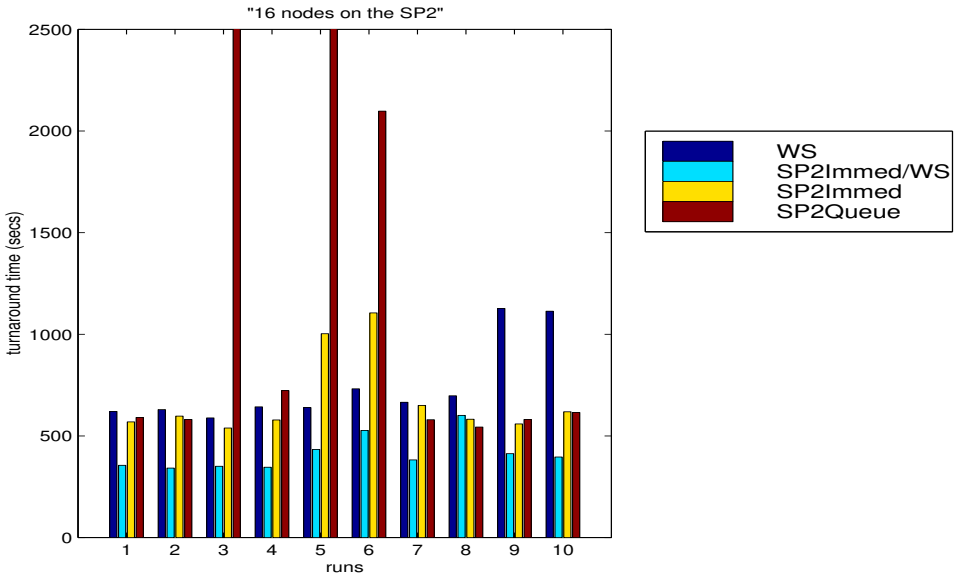


(a)

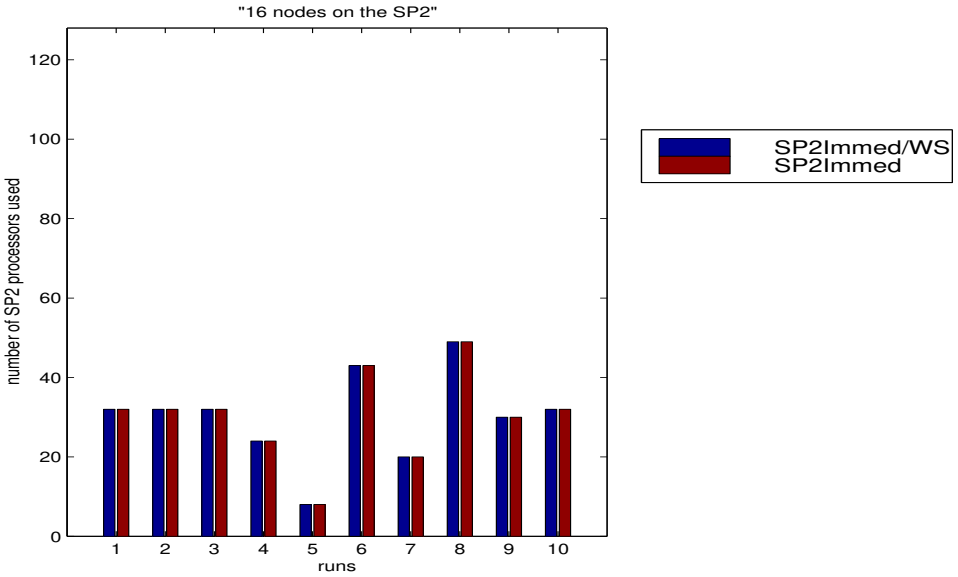


(b)

Figure 2: Experiment results when 8 nodes were requested for SP2Queue. (a) Turnaround time. (Values too large to fit on graph: run 4/SP2Queue - 3078s, run 9/SP2Queue - 6164s, run 10/SP2Immed - 2615s) (b) Number of SP2 nodes used by SP2Immed/WS and SP2Immed.

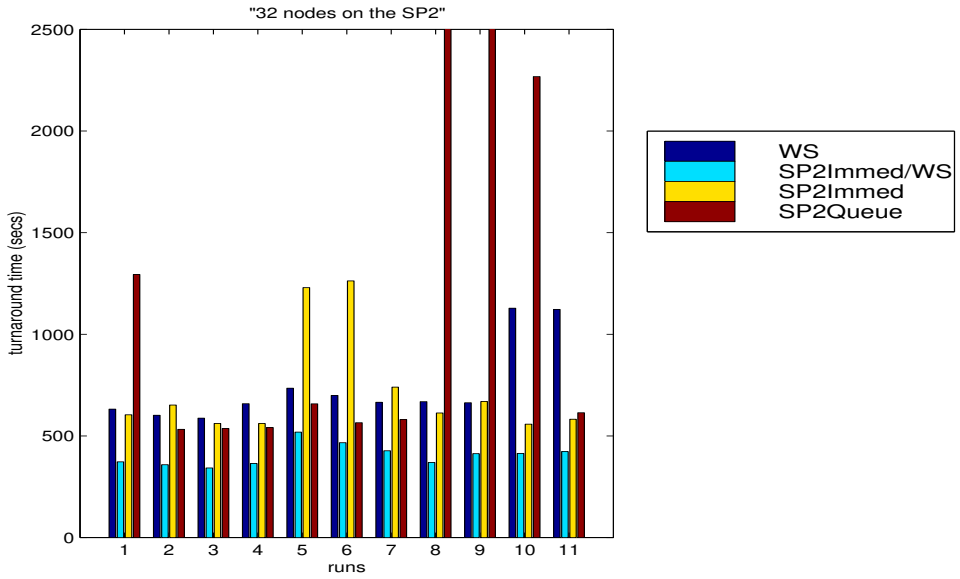


(a)

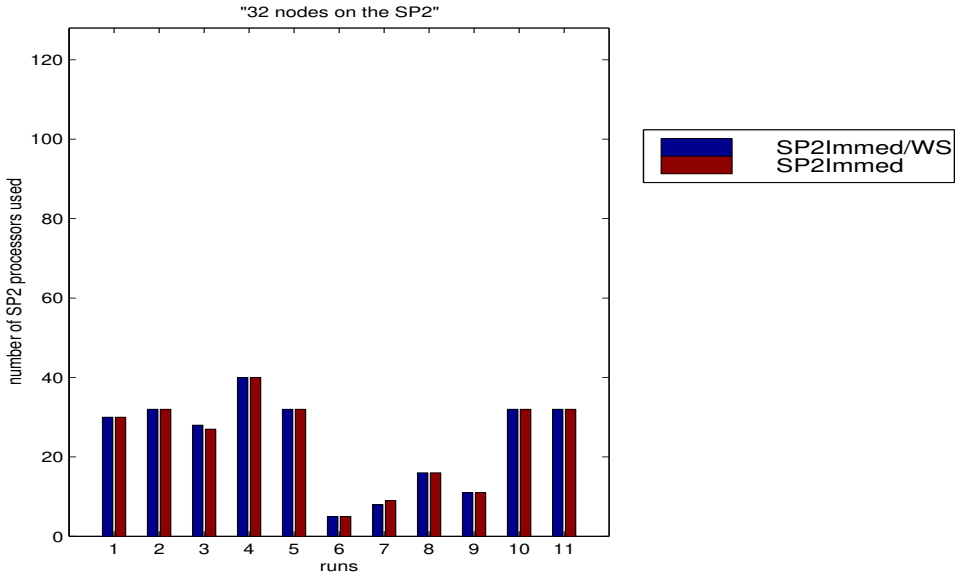


(b)

Figure 3: Experiment results when 16 nodes were requested for SP2Queue. (a) Turnaround time. (Values too large to fit on graph: run 3/SP2Queue - 20484s, run 5/SP2Queue - 17268s) (b) Number of SP2 nodes used by SP2Immed/WS and SP2Immed.



(a)



(b)

Figure 4: Experiment results when 32 nodes were requested for SP2Queue. (a) Turnaround time. (Values too large to fit on graph: run 8/SP2Queue - 27480s, run 9/SP2Queue - 9869s) (b) Number of SP2 nodes used by SP2Immed/WS and SP2Immed.