Netsolve: An Environment for Deploying Fault-Tolerant Computing

James S. Plank* Henri Casanova Jack J. Dongarra Terry Moore

Department of Computer Science, University of Tennessee, Knoxville, TN 37996

The Issue of Deployment

Consider the plight of the scientific user. In the world of uniprocessor workstations, his/her life is relatively simple. Software packages such as Matlab enable users to solve a wide variety of numerical problems with a convenient and flexible user interface. For less standard problems, a programmer may obtain software solutions from a repository like Netlib. These typically take the form of simple C/FORTRAN subroutines, and are very simple to incorporate into a programming platform. In short, life is good.

Unfortunately, for large problems, uniprocessors are not powerful enough, and often do not contain enough memory to provide solutions in a reasonable time frame. For this reason, programmers must turn to parallel and distributed computing platforms.

In parallel computing environments, multiple machines are combined to provide a computing entity that has enough processing power and memory to solve large problems in a reasonable time frame. However, these environments bring about many more problems, such as heterogeneity, data distribution, distributed ownership of resources, storage availability and component failures.

All of these issues make it difficult for a user to use a parallel computing platform like a single workstation. Any usable software must take all of these issues into account, which means that managing the computing platform becomes as big of an issue as performing the actual computation. When we look at the success stories of scientific computation on parallel platforms, they inevitably take the limited form of dedicated parallel machines like the Cray T3E or IBM SP2, or collections of workstations that are treated as dedicated parallel machines using software such as PVM or MPI. And even these present a very large challenge to any end-user who is not proficient at porting software and writing code.

So where does fault-tolerance fit into the equation? It doesn't. The same variety of obstacles that makes these platforms difficult to use also hinders the deployment of support for fault tolerance. Although many paradigms and algorithms for fault-tolerant distributed computing have been developed and tested, and in some cases made available as public-domain libraries, the average scientific user sees no benefits. With momentum gaining for the *computational power grid*, where disparate computational resources are combined and made available to a wide user base, the issue of *deploying* support for fault-tolerance becomes of paramount importance.

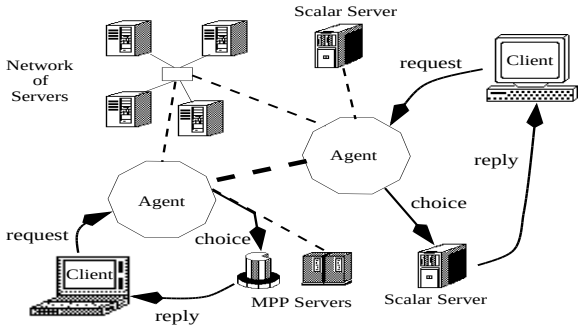

Figure 1: Overview of NetSolve

Introducing NetSolve

NetSolve [1] is a software environment for networked computing. The design of NetSolve allows users to make use of the variety of computational resources at their disposal using familiar interfaces from the world of uniprocessor computing (e.g. Matlab, simple procedure calls). NetSolve uses a *client-agent-server* paradigm as depicted in Figure 1. NetSolve users are the *clients*, and the computational resources are the *servers.* A server may be a uniprocessor, a MPP (Massively Parallel Processor), or a networked cluster of machines. When a user wants a certain computational task to be performed, he/she contacts an *agent* with the request. Each agent maintains information such as availability, load, and supported software, on a collection of servers. When a request from a user comes in, the agent selects a server to perform the task, and the server responds to the client's request.

The user's computation is performed remotely at the server. This means that the user sends his/her data to the server, and the server uses its own software to perform the computation. When the server finishes the computation, it

*plank@cs.utk.edu. This material is based upon work supported by NSF grant CCR-9703390, NSF and Technology Center Cooperative Agreement CCR-8809615 and DOE contract DE-AC05-840R21400

sends the result back to the client, and contacts the agent to notify it of the completion of the task.

As shown in Figure 1 there may be multiple NetSolve agents on the network, and different clients may contact different agents depending on their locations. The agents exchange information about their server pools, and allow access from any client to any server if advantageous.

NetSolve and Fault-Tolerance

NetSolve's design is not radically novel. Depending on the perspective, it may be viewed as an enhanced client-server architecture, or an enhanced RPC (remote procedure call) environment. However, the three-part design achieves a logical separation that allows each part to do what it does best without worrying about the details of the other parts. Specifically, the clients simply specify the computations, without worrying about how they get done. The servers simply perform the computations. Each server may optimize its computations for its own architecture, and most importantly, for each class of computation, the software is ported to a server *once*, by the server's administrator — the client need not be involved. Finally, the agents act as resource brokers, performing tasks that are not logical for the clients or servers to perform.

Due to this modular design, NetSolve's ability to *deploy* fault-tolerance is immense, and exists on two levels. The first is called *intra-server* fault-tolerance. Here, the individual servers employ fault-tolerance in their individual computations. This can take any form: checkpointing and rollback recovery, ABFT, backward error assertions, TMR, global replicated stores, etc. The important thing to note is that the fault-tolerance is part of the computational software, and thus is ported once by the server's administrator. The client is totally uninvolved, except that he/she reaps the benefits of fault-tolerant computation, perhaps unknowingly!

The second level is *inter-server* fault-tolerance, where a failure of one server is noted by an agent, and the server's computation is moved to another server. This can take the simple form of restarting the computation from the beginning on a new server, or having the state of the old server made available so that the new server may restart the computation from the middle. This requires that failed servers can make their state available to the agents, or that some form of *storage servers* be employed so that a server may save its state outside of its own scope.

Current Status

At present, NetSolve exists as a prototype that is being used in several research institutions. This prototype implements NetSolve's client-agent-server model and includes a suite of software for a variety of clients and servers. Currently, the clients may be Matlab scripts, or C/Fortran/Java programs. The following software has been made available as a standard suite for NetSolve servers: ARPACK, FitPack, ItPack, MinPack, FFTPACK, LAPACK, and QMR. Parallel servers have been developed that implement ScaLAPACK [2]. NetSolve has been ported to every major Unix platform. Windows 95/NT versions of the client are available and a server is being developed.

The prototype includes primitive forms of intra-server and inter-server fault-tolerance. The intra-server fault-tolerance takes the form of servers composed of pools of workstations managed by Condor [5]. Condor uses checkpointing and rollback recovery to move computations off machines that are unavailable due to failure, load, or ownership, and onto machines that are free and relatively idle. The inter-server fault-tolerance takes the form of agents detecting server failures by a time-out mechanism, and restarting the clients' computations on other servers.

Thus, although the mechanisms for fault-tolerance are currently very simple, they do represent a major step in *deploying* fault-tolerance to scientific users, since the NetSolve prototype is in use at several institutions. Moreover, through integration with projects such as Globus [3] and Legion [4], NetSolve is gaining momentum as a major enabling application of the computational grid, and will thereby deliver fault-tolerance to a large audience.

Research

The fault-tolerant research of the NetSolve team is divided into two categories: research *on NetSolve* and research *with NetSolve*. Research on Netsolve focuses on the development of storage servers, and the interaction of agents, servers and storage servers to enable inter-server fault-tolerance and migration. Research with NetSolve focuses on implementing a wide variety of fault-tolerant servers. We are currently working on fault-tolerant parallel ScaLAPACK servers that employ disk-based checkpointing, diskless checkpointing, and checksum/reverse computation. We plan to explore other forms of fault-tolerant servers including persistent stores, farming paradigms and recoverable DSM. Once implemented, the performance of these techniques may be compared directly, then deployed.

References

[1] H. Casanova and J. Dongarra. NetSolve: A network server for solving computational science problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(3):212–223, 1997.

[2] J. Choi, J. Dongarra, R. Pozo, and D. Walker. ScaLAPACK: A scalable linear algebra library for distributed memory concurrent computers. In *Proceedings of the Fourth Symposium on the Frontiers of Massively Parallel Computation*, pages 120–127. IEEE Publishers, 1992.

[3] I. Foster and C. Kesselman. Globus: A metacomputing infrastructure toolkit. International Journal of Supercomputer Applications (to appear), 1998.

[4] M. J. Lewis and A. Grimshaw. The core legion object model. In *Fifth IEEE International Symposium on High Performance Distributed Computing*, 1996.

[5] T. Tannenbaum and M. Litzkow. The Condor distributed processing system. *Dr. Dobb's Journal*, #227:40–48, February 1995.

NetSolve's home page is `http://www.cs.utk.edu/netsolve/`.