24

# NetSolve: past, present, and future; a look at a Grid enabled server[1]

**Sudesh Agrawal, Jack Dongarra, Keith Seymour, and Sathish Vadhiyar**

*University of Tennessee, Tennessee, United States*

## 24.1 INTRODUCTION

The emergence of Grid computing as the prototype of a next-generation cyber infrastructure for science has excited high expectations for its potential as an accelerator of discovery, but it has also raised questions about whether and how the broad population of research professionals, who must be the foundation of such productivity, can be motivated to adopt this new and more complex way of working. The rise of the new era of scientific modeling and simulation has, after all, been precipitous, and many science and engineering professionals have only recently become comfortable with the relatively simple world of uniprocessor workstations and desktop scientific computing tools. In this world, software packages such as Matlab and Mathematica, and languages such as C and Fortran represent general-purpose scientific computing environments that enable users – totaling more than a million worldwide – to solve a wide variety of problems

[1] Work supported in part by the NSF/NGS GRANT #NSF EIA-9 975 015, and NSF GRANT ACI-9 876 895

through flexible user interfaces that can model in a natural way the mathematical aspects of many different problem domains. Moreover, the ongoing, exponential increase in the computing resources supplied by the typical workstation makes these scientific computing environments more and more powerful, and thereby tends to reduce the need for the kind of resource sharing that represents a major strength of Grid computing. Certainly, there are various forces now urging collaboration across disciplines and distances, and the burgeoning Grid community, which aims to facilitate such collaboration, has made significant progress in mitigating the well-known complexities of building, operating, and using distributed computing environments. But it is unrealistic to expect the transition of research professionals to the Grid to be anything but halting and slow if it means abandoning the scientific computing environments that they rightfully view as a major source of their productivity.

The NetSolve project [1] addresses this difficult problem directly: the purpose of NetSolve is to create the middleware necessary to provide a seamless bridge between the simple, standard programming interfaces and desktop systems that dominate the work of computational scientists, and the rich supply of services supported by the emerging Grid architecture, so that the users of the former can easily access and reap the benefits (shared processing, storage, software, data resources, etc.) of using the latter. This vision of the broad community of scientists, engineers, research professionals, and students, working with the powerful and flexible tool set provided by their familiar desktop computing environment, and yet being able to easily draw on the vast, shared resources of the Grid for unique or exceptional resource needs, or to collaborate intensively with colleagues in other organizations and locations, is the vision that NetSolve is designed to realize.

## 24.2 HOW NetSolve WORKS TODAY

Currently, we have released Version 1.4.1 of NetSolve. The NetSolve homepage, located at http://icl.cs.utk.edu/NetSolve/, contains detailed information and the source code.

In any network-based system, we can distinguish three main paradigms: *proxy computing*, *code shipping* [2], and *remote computing*. These paradigms differ in the way they handle the user's data and the program that operates on this data.

In *proxy computing*, the data and the program reside on the user's machine and are both sent to a server that runs the code on the data and returns the result. In *code shipping*, the program resides on the server and is downloaded to the user's machine, where it operates on the data and generates the result on that machine. This is the paradigm used by Java applets within Web browsers, for example. In the third paradigm, *remote computing*, the program resides on the server. The user's data is sent to the server, where the programs or numerical libraries operate on it; the result is then sent back to the user's machine. NetSolve uses the third paradigm.

There are three main components in the NetSolve system: the agent, the server, and the client. These components are described below.

*Agent*: The agent represents the gateway to the NetSolve system. It maintains a database of NetSolve servers along with their capabilities and dynamic usage statistics for use in scheduling decisions. The NetSolve agent attempts to find the server that will service the request, balance the load amongst its servers, and keep track of failed servers. Requests are directed away from failed servers. The agent also adds fault-tolerant heuristics that attempt to use every likely server until it finds one that successfully services the request.

*Server*: The NetSolve server is the computational backbone of the system. It is a daemon process that awaits client requests. The server can run on single workstations, clusters of workstations, symmetric multiprocessors (SMPs), or massively parallel processors (MPPs). One key component of the server is the ability to wrap software library routines into NetSolve software services by using an Interface Definition Language (IDL) facility called the *NetSolve Problem Description File* (PDF).

*Client*: A NetSolve client user accesses the system through the use of simple and intuitive application programming interfaces (APIs). The NetSolve client uses these APIs to make a request to the NetSolve system, with the specific details required with the request. This call automatically contacts the NetSolve system through the agent, which in turn returns to the server, which can service the request. The client then contacts the server to start running the job with the input data. Figure 24.1 shows this organization.

●Au: ' Try' has been replaced by 'contact'. Please advise if the edit is appropriate.

As mentioned earlier, NetSolve employs fault tolerance to fulfill client requests. The NetSolve system ensures that a user request will be completed unless every single resource capable of solving the problem has failed. When a client sends a request to a NetSolve agent, it receives a sorted list of computational servers to ●contact. When one of these servers has been successfully contacted, the numerical computation starts. If the contacted server fails during the computation, another server is contacted, and the computation is restarted. Table 24.1 shows how calls are made to NetSolve through Matlab, C, and Fortran client interfaces.

There are a number of topics that we will not be able to cover fully in this chapter. The list includes the following:

- Security
  Uses Kerberos V5 for authentication
- Separate server characteristics
  Prototype implementation of hardware and software servers
- Hierarchy of agents
  Providing a more scalable configuration
- Monitor NetSolve network
  Track and monitor usage
- Network status
  Use of network weather service
- Internet Backplane protocol
  Middleware for managing and using remote storage
- Fault tolerance
- The dynamic nature of servers

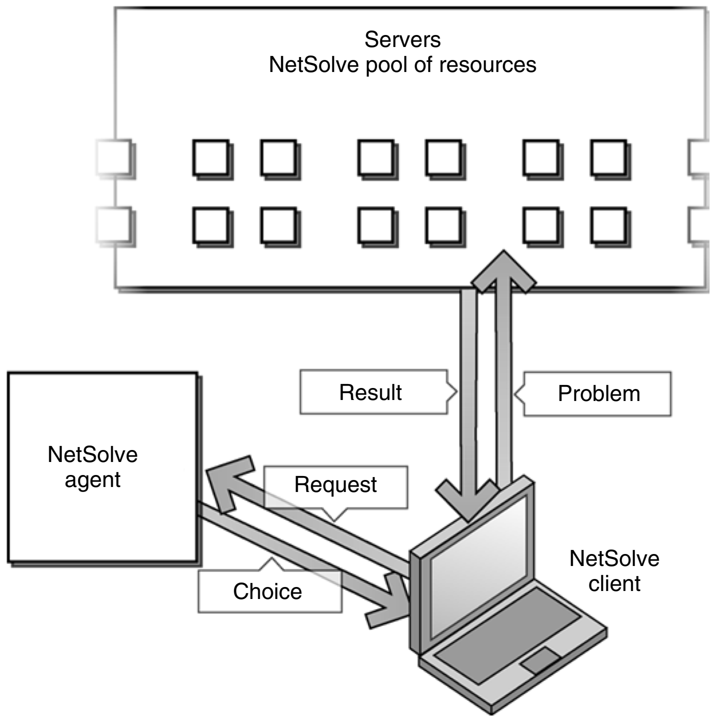

**Figure 24.1** NetSolve's organization.

**Table 24.1** Solving a linear system A $x$ = b with NetSolve

| |
|---|
| 'A' is an $m \times m$ matrix and 'b' is an $m \times n$ matrix<br>In C and Fortran, the right-hand side is overwritten by the solution |
| From MATLAB:<br>$x$ = netsolve('linsol', A,b) |
| From C:<br>netsl("linsol",nsinfo,m,n,A,lda,b,ldb) |
| From FORTRAN:<br>CALL NETSL("linsol", NSINFO,M,N,A,LDA,B,LDB) |

• Automated adaptive algorithm selection
  Dynamically determine the best algorithm based on the system status and the nature of user problem.

Additional details can be found on the NetSolve [3] Web site.

## 24.3 NetSolve IN SCIENTIFIC APPLICATIONS

NetSolve has been used in a number of applications for resource management purposes, to enable parallelism in the applications and to help users avoid installation of cumbersome software. Following Sections detail some of the applications that reap the benefits of using NetSolve.

### 24.3.1 IPARS (integrated parallel accurate reservoir simulators)

IPARS [4], developed under the directorship of Mary Wheeler at the Center for Subsurface Modeling, at the University of Texas' Institute for Computational and Applied Mathematics, TICAM, is a framework for developing parallel models of subsurface flow and transport through porous media. IPARS can simulate single-phase (water only), two-phase (water and oil), or three-phase (water, oil, and gas) flow through a multiblock 3D porous medium. IPARS can be applied to model water table decline due to overproduction near urban areas, or enhanced oil and gas recovery in industrial applications.

A NetSolve interface to the IPARS system that allows users to access the full functionality of IPARS was constructed. Accessing the system via the MATLAB, C, Mathematica, or FORTRAN interfaces automatically executes simulations on a cluster of dual-node workstations that allow for much quicker execution than what would be possible on a single local machine. The NetSolve system also does the postprocessing of the output to use the third-party software, TECPLOT, to render the 3D output images. Among other things, NetSolve provides a gateway to the IPARS system without downloading and installing the IPARS code. This means it can even be used on platforms that it has not yet been ported to. The interface was further enhanced by embedding it in HTML form [5] within a Web browser so that with just access to a Web browser one can enter input parameters and submit a request for execution of the IPARS simulator on a NetSolve system. The output images are then brought back and displayed to the Web browser. This interaction showed how the NetSolve system can be used to create a robust Grid computing environment in which powerful modeling software, like IPARS, becomes both easier to use and to administer.

### 24.3.2 MCell

MCell [6] is a general Monte Carlo simulator of cellular microphysiology. MCell uses Monte Carlo diffusion and chemical reaction algorithms in 3D to simulate the complex biochemical interactions of molecules inside and outside of living cells. NetSolve is used as a resource management framework to manage the execution of a large number of MCell simulations on a large number of resources in the NetSolve Grid. One of the central pieces of the framework is a scheduler that takes advantage of MCell input data requirements to minimize turnaround time. This scheduler is part of the larger AppLeS at the University of California, San Diego. The use of robust, coherent, and fault-tolerant NetSolve pool of resources allowed the MCell researchers to implement parallelism in the simulations with simple sequential calls to NetSolve.

### 24.3.3 SARA3D

SARA3D [7] is a code developed by BBN Acoustic Technologies that is used to solve structural acoustic problems for finite-element structures emerging in a user-defined fluid. The SARA3D application accepts as input a file describing the structure, the materials involved, the fluid properties, and the actual analyses to be performed. As output, the application generates a variable number of files that contain different types of data calculated during the analyses. These output files can be further processed to compute quantities such as radiated power, near and far-filled pressures, and so on. The SARA3D application also consists of a large number of phases. The output files produced by one phase of the application will serve as inputs to the next phase of the application. These intermediate files are of no particular interest to the end user. Also, it is desirable that the final output files produced by the application be shared by a large number of users.

For these purposes, SARA3D was integrated into the NetSolve framework utilizing the data staging capability of NetSolve. Different NetSolve servers implementing the different phases of the application were started. The user caches his input data near the servers using the data staging facility in NetSolve. The user also indicates to the servers to output the intermediate results and the final outputs to data caches. By this framework, only relevant and sharable data are transmitted to the end users.

### 24.3.4 Evolutionary farming

Evolutionary Farming (EvCl) is a joint effort between the Department of Mathematics and the Department of Ecology and Evolutionary Biology at the University of Tennessee. The goals of the EvCl research are to develop a computer-based model simulating evolution and diversification of metapopulations in a special setting and to explore relationships between various parameters affecting speciation dynamics. EvCl consists of two main phases, *evolve*, which simulates the evolution and *cluster*, which is used for species determination. Since the problem involves separate runs of the same parameters due to the stochastic nature of simulation, the NetSolve task-farming interface was used to farm these separate simulations onto different machines. The use of NetSolve helped in significantly reducing the cumbersome management of disparate runs for the problem.

### 24.3.5 LSI-based conference organizer

Latent Semantic Indexing (LSI) [8] is an information retrieval method that organizes information into a semantic structure that takes advantage of some of the implicit higher-order associations of words with text objects. The LSI method is based on singular value decomposition (SVD) of matrices consisting of documents and query terms. Currently, LSI is being used at the University of Tennessee to construct a Web-based conference organizer [9] that organizes conference sessions based on submitted abstracts or full-text documents. The computationally intensive SVD decompositions will be implemented by the NetSolve servers allowing the developers of the Conference Organizer to concentrate their research efforts in other parts of the organizer, namely, the Web interfaces. Integrating the Conference Organizer into the NetSolve system also allows the users of the organizer to avoid installing the SVD software in their system.

## 24.4 FUTURE WORK

Over time, many enhancements have been made to NetSolve to extend its functionality or to address various limitations. Some examples of these enhancements include task farming, request sequencing, and security. However, some desirable enhancements cannot be easily implemented within the current NetSolve framework. Thus, future work on NetSolve will involve redesigning the framework from the ground up to address some of these new requirements.

On the basis of our experience in developing NetSolve, we have identified several requirements that are not adequately addressed in the current NetSolve system. These new requirements – coupled with the requirements for the original NetSolve system – will form the basis for the next generation of NetSolve.

The overall goal is to address three general problems: ease of use, interoperability, and scalability. Improving ease of use primarily refers to improving the process of integrating user codes into a NetSolve server. Interoperability encompasses several facets, including better handling of different network topologies, better support for parallel libraries and parallel architectures, and better interaction with other Grid computing systems such as Globus [10] and Ninf [11]. Scalability in the context used here means that the system performance does not degrade as a result of adding components to the NetSolve system.

The sections below describe some of the specific solutions to the general problems discussed earlier.

### 24.4.1 Network address translators

As the rapid growth of the Internet began depleting the supply of IP addresses, it became evident that some immediate action would be required to avoid complete IP address depletion. The IP Network Address Translator [12] is a short-term solution to this problem. Network Address Translation (NAT) allows reuse of the same IP addresses on different subnets, thus reducing the overall need for unique IP addresses.

As beneficial as NATs may be in alleviating the demand for IP addresses, they pose many significant problems to developers of distributed applications such as NetSolve [13]. Some of the problems pertaining to NetSolve include the following:

- *IP addresses are not unique*: In the presence of a NAT, a given IP address may not be globally unique. Typically the addresses used behind the NAT are from one of the several blocks of IP addresses reserved for use in private networks, though this is not strictly required. Consequently, any system that assumes that an IP address can serve as the unique identifier for a component will encounter problems when used in conjunction with a NAT.
- *IP address-to-host bindings may not be stable*: This has similar consequences to the first issue in that NetSolve can no longer assume that a given IP address corresponds uniquely to a certain component. This is because, among other reasons, the NAT may change the mappings.
- *Hosts behind the NAT may not be contactable from outside*: This currently prevents all NetSolve components from existing behind a NAT because they must all be capable of accepting incoming connections.

- *NATs may increase network failures*: This implies that NetSolve needs more sophisticated fault-tolerance mechanisms to cope with the increased frequency of failures in a NAT environment.

To address these issues we are currently investigating the development of a new communications framework for NetSolve. To avoid problems related to potential duplication of IP addresses, the NetSolve components will be identified by a globally unique identifier, for example, a 128-bit random number. The mapping between the component identifier and a real host will not be maintained by the NetSolve components themselves, rather there will be a discovery protocol to locate the actual machine running the NetSolve component with the given identifier. In a sense, the component identifier is a network address that is layered on top of the real network address such that a component identifier is sufficient to uniquely identify and locate any NetSolve component, even if the real network addresses are not unique. This is somewhat similar to a machine having an IP address layered on top of its MAC address in that the protocol to obtain the MAC address corresponding to a given IP address is abstracted in a lower layer.

An important aspect to making this new communications model work is the *relay*, which is a component that will allow servers to exist behind a NAT. Since a server cannot accept unsolicited connections from outside the private network, it must first register with a relay. The relay acts on behalf of the component behind the NAT by establishing connections with other components or by accepting incoming connections. The component behind the NAT keeps the connection with the relay open as long as possible since it can only be contacted by other components while it has a control connection established with the relay. To maintain good performance, the relay will only examine the header of the messages that it forwards, and it will use a simple table-based lookup to determine where to forward each message. Furthermore, to prevent the relay from being abused, authentication will be required.

Since NATs may introduce more frequent network failures, we must implement a protocol to allow NetSolve components to reconnect to the system and resume the data transfer if possible. We are still investigating the specifics of this protocol, but at the least it should allow the servers to store the results of a computation to be retrieved at some time later when the network problem has been resolved. Additionally, this would allow a client to submit a problem, break the connection, and reconnect later at a more convenient time to retrieve the results.

### 24.4.2 Resource selection criteria

In the current NetSolve system, the only parameter that affects the selection of resources is the problem name. Given the problem name, the NetSolve agent selects the ‘best’ server to solve that problem. However, the notion of which server is best is entirely determined by the agent. In the next generation of NetSolve, we plan to extend this behavior in two ways. First, we should allow the user to provide constraints on the selection process. These selection constraints imply that the user has some knowledge of the characteristics that will lead to a better solution to the problem (most probably in terms of speed). Second, we should allow the service providers (that is, those organizations that provide NetSolve servers) to specify constraints on the clients that can access that service. For example, an organization may want to restrict access to a certain group of collaborators. We are currently examining the use of XML as a resource description language.

## 24.5 CONCLUSIONS

We continue to evaluate the NetSolve model to determine how we can architect the system to meet the needs of our users. Our vision is that NetSolve will be used mostly by computational scientists who are not particularly interested in the mathematical algorithms used in the computational solvers or the details of network-based computing. NetSolve is especially helpful when the software and hardware resources are not available at hand. With Grid computing, there exists many interesting areas of research to explore and much room for improvement. We envision future work in features like dynamically extensible servers whose configuration can be modified on the fly. The new strategy will be to implement a just-in-time binding of the hardware and software service components, potentially allowing servers to dynamically download software components from service repositories. Parallel libraries could be better supported by data distribution/collection schemes that will marshal input data directly from the client to all computational nodes involved and collect results in a similar fashion. Efforts also need to be made so that clients can solve jobs with large data sets on parallel machines; the current implementation requires this data to be passed in the call since the calling sequence expects a reference to the data and not a reference via a file pointer, and this may not be possible.

As researchers continue to investigate feasible ways to harness computational resources, the NetSolve system will continue to emerge as a leading programming paradigm for Grid technology. Its lightweight and ease of use make it an ideal candidate for middleware, and as we discover the needs of computational scientists, the NetSolve system will be extended to become applicable to an even wider range of applications.

## REFERENCES

1. Casanova, H. and Dongarra, J. (1997) NetSolve: A network-enabled server for solving computational science problems. *The International Journal of Supercomputer Applications and High Performance Computing*, **11**(3), 212–223.
2. Agrawal, S. and Dongarra, J. J. (2002) Hardware Software Server in NetSolve, UT-CS-02-480, University of Tennessee, Computer Science Department, Knoxville, 2002.
3. NetSolve Web site http://icl.cs.utk.edu/netsolve/.
4. IPARS Web site http://www.ticam.utexas.edu/CSM/ACTI/ipars.html.
5. TICAM Web site http://www.ticam.utexas.edu/∼ut/webipars/AIR.
6. MCell Web site http://www.mcell.cnl.salk.edu.
7. Allik, H., Dees, R., Moore, S. and Pan, D. (1995) SARA-3D User's Manual, BBN Acoustic Technologies.
8. Berry, M. W., Dumais, S. T. and O'Brien, G. W. (1995) Using linear algebra for intelligent information retrieval. *SIAM Review*, **37**(4), 573–595.
9. COP Web site http://shad.cs.utk.edu/cop.
10. Foster, I. and Kesselman, C. ●(1997) Globus: A metacomputing infrastructure toolkit. *International Journal of Supercomputer Applications.*

●Au: Please provide the volume number and the page range for this reference.

●Au: Reference 14 and 15 have not been cited in text. Please provide the place of citation.
●Au: Please provide the standard number for this reference.

11. Nakada, H., Sato, M. and Sekiguchi, S. (1999) Design and implementations of Ninf: towards a global computing infrastructure. *Future Generation Computing Systems*, Metacomputing Issue, **15**(5–6), 649–658.
12. Egevang, K. and Francis, P. (1994) The IP Network Address Translator (NAT), RFC Tech Report 1631, May 1994.
13. Moore, K. (2002) Recommendations for the Design and Implementation of NAT-Tolerant Applications, Informal Communications, ●February 2002.
14. ●XML-RPC Web site http://www.xml-rpc.com/.
15. Simple Object Access Protocol (SOAP) 1.1 Web site http://www.w3.org/TR/SOAP/.