Secure Remote Access to Numerical Software and Computational Hardware

Dorian C. Arnold, Shirley Browne, Jack Dongarra, Graham Fagg and Keith Moore
Computer Science Department
University of Tennessee
Knoxville, TN 37996
[darnold, browne, dongarra, fagg, moore]@cs.utk.edu

Keywords: Distributed Computing, Problem Solving Environments, Heterogeneous Network Computing, Security, Numerical Solvers.

Abstract

In today's research environment, the chief platforms of development tend to be high-end workstations or supercomputers running some variant of the UNIX operating system. While these platforms are powerful and robust, non-computer scientists generally view them as cumbersome and difficult to learn. This serves as part of the motivation that led to the development of NetSolve, a system that allows for seamless access to computational software and hardware resources. Our approach is to implement a client/agent/server system that provides a uniform, intuitive interface to high-performance components.

This article provides an introduction to the NetSolve software system that includes load balancing techniques, dynamic scheduling heuristics and fault tolerance. We also briefly discuss the use of NetSolve at the ARL MSRC and how our extensions to Netsolve using Kerberos V5 allow current MSRC users transparent off site authentication, while allowing system adminstrators fine grained access control to the computational resources.

Introduction

As researchers discover new and innovative ways to harness computational resources and make them available to high-performance applications, the emergence of Grid [5] systems and Grid computing concepts are constant. The Grid analogy is to that of the electrical power grid into which we unwittingly plug components ranging from toasters to laptop computers without regard for the

source of the electric current which may be hundreds of miles away. The vision for Computational Grids is one which has users similarily "plugging in" and seamlessly accessing the resources they need regardless of the geographic location, ownership or component architecture of that resource.

Grid Concepts

The ideal Grid will be:

- **Ubiquitous**
  A user must be able to easily and conveniently locate, access and use computational resources regardless of his own location our computational environment. The Grid should therefore be expansive so as not to prevent non-previleged users from reaping the benefits of the level of resource connectivity that we experience today.

- **Heterogeneous**
  Numerous organizations and institutions have conceived, researched and developed Computational Grid systems. While each of these systems have the same overarching goals, their perspectives and approaches differ. Thusly, they all maintain custom usage models, architectures and components, protocols and security infrastructures. They likewise pose unique advantages and disadvantages in terms of ease-of-use, ease-of-administration, application integration, flexibility, etc. It is unlikely that any one of these systems will emerge as the sole Grid system of choice as their varying features and functionalities appeal to different user communities. Therefore, the Grid must be able to support an environment of heterogeneous Grid Systems. Standards must be developed, and both policy and infrastructure are needed to support inter-system transactions if the concept of the Grid is to be realized.

- **Secure** The Grid must provide assurances to both the user and the service provider. The user must be assured of the quality and accuracy of the service he is being provided, while the service provider must be convinced that he can determine who he wishes to access his resources and control the level and amount of access authorized users have. The service provider must also have a high degree of confidence that his system will not be compromised either by hackers or innocent, yet improper programming or software.

- **Reliable** Although the number and availability of the components of a Grid are expected to (and will) vary dynamically, this fact should be practically hidden from the user, and alternate, if less reliable, resources should be made available.

- **Efficient** The quality of the service provided by the Grid must be very high, and, therefore, the impact of system overhead must be carefully considered. Although, performance might be less important when there is no other viable solution, one of the primary motivating factors to the conception of the Grid was to make supercomputer power generally available, and in some cases, harness supercomputers to provide performance higher than that available from any single machine.

- **Scalable** As the concepts of the Grid become more popular, it must be able to deal with both a large-scale number of components and users at the same time without compromising the service that any single user receives.

The NetSolve Approach

NetSolve is a resource management system that uses remote procedure call (RPC) mechanisms to provide access to otherwise disparate resources. NetSolve was conceived as a way to relieve the computational scientist of the burden of installing and maintaining complex scientific codes, yet still providing them with total access to the software.

This article serves to introduce the user to the NetSolve system and show how it can be used to easily and seamlessly enable the use of widely distributed computational resources. Of primary focus is the Matlab interface to the NetSolve system, as this is the interface of choice used by researchers at the Army Research Laboratory. We give a general overview of the NetSolve project as well as provide details about the Matlab client interface. We then discuss our interactions with the Kerberos V5 environment to provide authentication and access control. To conclude, we point to similar researh efforts in the area of Grid computing.

The NetSolve Computational Environment

The NetSolve project is being developed at the University of Tennessee's Innovative Computing Laboratory of the Computer Science Department. Its original motivation was to alleviate the difficulties that domain scientists encounter when trying to locate/install/use numerical software, especially on multiple platforms. Today, the name NetSolve has become a misnomer, as the system has evolved into much more than a way to access numerical solver routines. NetSolve provides an environment that monitors and manages computational resources, both hardware and software, and allocates the services they provide to NetSolve-enabled client programs. It incorporates load balancing and scheduling strategies to distribute tasks evenly amongst servers. Built upon standard Internet protocols, like TCP/IP sockets, it is available for all popular variants of the UNIX operating system, and parts of the system are available for the Microsoft Windows '95, '98, '00 and NT platforms.

Figure 1 shows the infrastructure of the NetSolve system and its relation to the applications that use it. NetSolve and systems like it are often referred to as Grid Middleware; this figure helps to make the reason for this terminology clearer. The shaded parts of the figure represent the NetSolve system. It can be seen that NetSolve acts as glue layer that brings the application or user together with the hardware and/or software it needs to complete useful tasks.

At the top tier, the NetSolve client library is linked in with the user's application. The application then makes calls to NetSolve's application programming interface (API) for specific services.

Through the API, NetSolve client-users gain access to aggregate resources without the users needing to know anything about computer networking or distributed computing. In fact, the user does not even have to know remote resources are involved. NetSolve currently supports the C, FORTRAN, Matlab, and Mathematica programming interfaces as languages of implementation for client programs. Section discusses the client interface from the perspective of Matlab users in detail.

The NetSolve agent represents the gateway to the NetSolve system. It maintains a database of NetSolve servers along with their capabilities (hardware performance and allocated software) and dynamic usage statistics. It uses this information to allocate server resources for client requests. The agent, in its resource allocation mechanism, attempts to find the server that will service the request the quickest, 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 features that attempt to use every appropriate server until it finds one that successfully services the request.

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 multi-processors or machines with massively parallel processors. A key component of the NetSolve server is a source code generator which parses a NetSolve problem description file (PDF). This PDF contains information that allows the NetSolve system to create new modules and incorporate new functionalities. In essence, the PDF defines a wrapper that NetSolve uses to call the function being incorporated.

The Status Of NetSolve

Version 1.3 was released in May of 2000. Features implemented in this release include a Java GUI to aid in the creation of PDFs, a Microsoft Excel interface, more object datatypes, more server modules included with the distribution, and enhanced load balancing among other things. NetSolve-1.3, including APIs for the Win32 platform, can be downloaded from the project web site at **www.cs.utk.edu/netsolve**. NetSolve has been recognized as a significant effort in research and development, and was named in R&D Magazine's top 100 list for 1999. The reader is directed to [2] for further details of the system not discussed in this article.

The Matlab Client

The key feature of NetSolve is its ability to extend the capabilities of programming languages, like C or Fortran, or that of problem solving environments (PSE), like Matlab [11] or Mathematica [12]. This section focuses on the Matlab interface and shows how NetSolve can i) increase the functions available from this PSE and ii) distribute the workload of this PSE across multiple processors of an multiprocessor machine or across hosts of a workstation cluster. Computational scientists view computers and programs, not as an end in itself, but as a means to an end – in other words, they are not overly concerned with the algorithmic implementations of numerical solvers and mathematical equations; of prime interest are the scientific results attained with the help of computers, albeit in

biology, chemistry, or nuclear engineering. Many times they are not expert programmers, and thus a programming environment like Matlab appeals to them.

Matlab

Matlab is a PSE that integrates a high-level programming language, numerical computations and visualization capabilities. At its core, it provides hundreds of functions well-suited for a variety of domains, including statistical analysis, mathematical modelling, and scientific engineering. The fact that the language is interpreted means that users can interactively define data sets and initiate computations, but this fact doesn't preclude writing complex scripts or programs that contain large series of instructions. MATLAB is used in a variety of application areas including signal and image processing, control system design, financial engineering, and medical research.

Extending Matlab with NetSolve

Matlab has the capability of dynamically linking with libraries, called mex files, and then calling the functions available within these mex files to leverage their functionality. The NetSolve client layer can be compiled into a set of mex files, whose functionality is to interact with the NetSolve agent and server components. The two main functions of this interface are `netsolve()` and `netsolve_nb()`; they make blocking and non-blocking calls to NetSolve, respectively. (The other functions are concerned with error handling.) The `netsolve()` call blocks until such time as the results are available or an error has occured; the `netsolve_nb()` call returns immediately, passing the user a "handle" to the request. This handle can later be used to probe the system to see if the results are available and gather them when they are.

Figure 2 helps to show what the programming code would look like before and after the NetSolve API has been integrated. The code on the left handside is making a call to a Matlab native function to multiply two matrices `A` and `B` and store the result in `C`. The call to NetSolve, on the right handside of the figure, achieves the same end via the NetSolve framework.

The (hidden) semantics of a NetSolve request are:

1. Client contacts the agent for a list of capable servers.
2. Client contacts server and sends input parameters.
3. Server runs appropriate service.

There are many advantages to using a system like NetSolve from within an environment like Matlab. Although within its framework, Matlab has many functions available, NetSolve provides an easy and seamless way to integrate new functions into the Matlab interface. Furthermore, these functions are not available solely from Matlab, but from the other supported interface languages, and any other PSE into which NetSolve has been integrated. Also, none of the many functions supported

by Matlab are supported in their parallel form. The nature of NetSolve is such that virtually any software can be integrated into the system, which includes modules from parallel libraries, like those from ScaLAPACK [4] or PETSc [3]. In the example of Fig. 2, the possibility exists for the `matmul` module of the NetSolve system to be a parallel code running on multiple nodes of a multiprocessor or multiple hosts of a workstation cluster, whereas in the ordinary Matlab call, there is no other possibility but for the code to run solely on the local host, in serial, even when the machine has multiple processors.

Using the non-blocking interface to NetSolve, one can achieve simple, yet highly-effective, levels of parallelism by making simultaneous invocations of the NetSolve system which would then be serviced on multiple server hosts. Again, there is currently no other way to leverage multiple processors within the Matlab environment using ordinary Matlab techniques. In this way, NetSolve can make the power of supercomputers accessible from low-end machines like laptop computers. Furthermore, as explained above, NetSolve adds scheduling heuristics that attempt to find the most expeditious route to solve any given problem.

The power of NetSolve can easily be shown by the mode with which it is used at the Army Research Laboratory (ARL) major shared resource center (MSRC). NetSolve servers have been installed on SGI servers and workstations and configured to provide the services of the LAPACK [1] numerical library. The Matlab client is then used from Microsoft Windows NT workstations to provide users with access to these high-performance numerical routines running on high-performance hardware. Eventually, this infrastructure will be routinely used by ARL's Integrated Modeling and Testing group. numerical routines that then run on supercomputers, like their SGI Origin 2000. Through a similar collaboration, NetSolve will be employed at the Engineering Research and Development Center MSRC in some of their structural acoustics applications.

Although the Microsoft platform has become the language of communication, it is seldom that high-performance code development takes place in this user-friendly environment. Using NetSolve, however, users are not restricted from invoking complex codes in a highly expedient manner from this setting.

Kerberos Authentication

Interaction in a Grid or any distributed environment ultimately demands that there are reassuring mechanisms in place to restrict and control access to computational resources and sensitive information. In the latest version of NetSolve, we have introduced the ability to generate access control lists which are used to grant and deny access to the NetSolve servers. We use Kerberos V5 [9] services to add this functionality to NetSolve, as it is one of the most trusted and popular infrastructures for authentication services.

The server implements access control via a simple list of Kerberos principal names. In Kerberos nomenclature, a principal is a string that uniquely identifies a user or service within Kerberos; it

is usually comprised of a name (often a UNIX username) and a realm, which defines the Kerberos "domain." This list of principals is maintained by the NetSolve server administrator and is kept in a text file which is consulted by the server. This text file is readable and writeable only by the user under whose previleges the server is being run, and is therefore protected by the UNIX security protocols. (NetSolve servers run in user space and don't require root previleges to be installed.) A request to a NetSolve server must be made on behalf of one of those principal names. If the principal name associated with the Kerberos credentials in the request appears in the list, and the credentials are otherwise valid, the request will be honored. Otherwise, the request will be denied.

Since the NetSolve server was not designed to run as a set-uid program, it is not currently feasible to have the NetSolve server run processes using the user-id of the particular user who submitted the request. NetSolve thus uses its own service principal name of "netsolve" rather than using the "host" principal. What this means (among other things) is that service principals need to be generated for each NetSolve server, even if host principals are in place.

Figure 3 illustrates the NetSolve/Kerberos interactions that take place when a NetSolve service is being requested. A user first authenticates himself to Kerberos using a Kerberos utility like `kinit`. Through this utility, he talks to the Authentication Service on the Key Distribution Center (KDC) to get a Ticket Granting Ticket(TGT). This ticket is encrypted with the user's password.

When the user wants to talk to a Kerberized NetSolve server, he uses the TGT to talk to the Ticket Granting Service (TGS) (which also runs on the KDC). The TGS verifies his identity using the TGT and issues a ticket for the NetSolve service. The user can then pass this service ticket to the Server who will verify the client's credentials and check to see if he is an authorized user. If so, he will grant the requested services to client.

The reason the TGT exists is to prevent a user from having to enter their password every time they wish to connect to a Kerberized service or keep a copy of their password around. If the TGT is compromised, an attacker can only masquerade as a user until the ticket expires.

The NetSolve system supports the interoperation of Kerberized and non-Kerberized components. In either case the client sends a request to the server, and the established protocol dictates that, if required, the server must send an explicit request for authentication. At this point, the client can either abort the transaction (knowing it doesn't have the proper credentials) or attempt to authenticate itself to receive proper servicing. Currently there is no mechanism to allow the client to insist on authentication of the server - a Kerberized client will happily talk with either Kerberized or non-Kerberized servers.

This version of Kerberized NetSolve performs no encryption of the data exchanged among NetSolve clients, servers, or agents, nor is there any integrity protection for the data stream.

Related Systems

Ninf [10] is an ongoing global computing infrastructure project which allows users to access computational resources including hardware, software and scientific data distributed across a wide area network with an easy-to-use interface. The architecture of this system bares many similarities with that of NetSolve; in fact, in the past collaborative efforts have been made to create "bridges" that enable the two systems to leverage each other's components.

The Globus [6] project has a somewhat different scope than that of NetSolve. Although they both aim to aggregate computational resources, Globus' thrust seems to be towards supercomputer components, whereas NetSolve targets any scientist of engineer and provide them with high-levels of service regardless of the infrastructure available to them.

Legion [7] is an object-based meta-computing software project. It is designed to scale to up to a system of "millions of hosts and trillions of objects, tied together with high-speed links." The Legion system has support for data management, fault tolerance, site autonomy and a wide range of security options.

Condor [8] is a meta-computing system of a slightly different flavor. The idea behind Condor is to use potentially idle cpu cycles to service needy processes. It monitors workstations and whenever it detects idleness (the meaning of which is set by the owner of the workstation) it schedules that workstation to receive tasks. Whenever the workstation should become non-idle, it preempts currently running tasks and schedules them to run on a different host.

Conclusion

The NetSolve system is an enabling toolkit that has the ability to harness computational resources into a single virtual machine. These computational resources include both hardware and software components. NetSolve also provides the tools to extend its capabilities. The Kerberos extensions of NetSolve provide it with trusted mechanisms by which to control access to computational resources. As the system continues to mature, NetSolve will continue to emerge as one of the more popular Grid systems of choice.

Acknowledgements

This work has been partially supported by the DoD High Performance Computing Modernization Program, ARL Major Shared Resource Centers, through Programming Environment and Training (PET).

Views, opinions, and/or findings contained in this report are those of the authors and should not be construed as an official Department of Defense position, policy or decision unless so designated by other official documentation.

*

References

[1] E. Anderson, Z. Bai, C. Bischof, S. Blackford, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, and D. Sorensen. *LAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, third edition, 1999.

[2] D. Arnold, S. Blackford, and J. Dongarra. Users' Guide to NetSolve V1.3. Technical report, Computer Science Dept., University of Tennessee, May 2000.

[3] S. Balay, W. D. Gropp, and B. F. Smith. *Modern Software Tools in Scientific Computing*, pages 163–202. Birkhauser Press, 1997.

[4] L. S. Blackford, J. Choi, A. Cleary, E. D'Azevedo, J. Demmel, I. Dhillon, J. Dongarra, S. Hammarling, G. Henry, A. Petitet, K. Stanley, D. Walker, and R. C. Whaley. *ScaLAPACK Users' Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, 1997.

[5] I. Foster and C. Kesselman, editors. *The Grid, Blueprint for a New computing Infrastructure*. Morgan Kaufmann Publishers, Inc., 1998.

[6] I. Foster and K Kesselman. Globus: A Metacomputing Infrastructure Toolkit. In *Proc. Workshop on Environments and Tools*. SIAM, to appear.

[7] A. Grimshaw, W. Wulf, J. French, A. Weaver, and P. Jr. Reynolds. A Synopsis of the Legion Project. Technical Report CS-94-20, Department of Computer Science, University of Virginia, 1994.

[8] M. Litzkow, M. Livny, and M. Mutka. Condor - A Hunter of Idle Workstations. In *Proc. of the 8th International Conference of Distributed Computing Systems, San Jose, CA*, pages 104–111, June 1988.

[9] B. Clifford Neuman and Theodore Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33–38, September 1994.

[10] S. Sekiguchi, M. Sato, H. Nakada, S. Matsuoka, and U. Nagashima. Ninf : Network based Information Library for Globally High Performance Computing. In *Proc. of Parallel Object-Oriented Methods and Applications (POOMA), Santa Fe, CA*, 1996.

[11] Inc. The Math Works. *Using MATLAB Version 5*. The Math Works, Inc., 1992.

[12] S. Wolfram. *The Mathematica Book, Third Edition*. Wolfram Median, Inc. and Cambridge University Press, 1996.

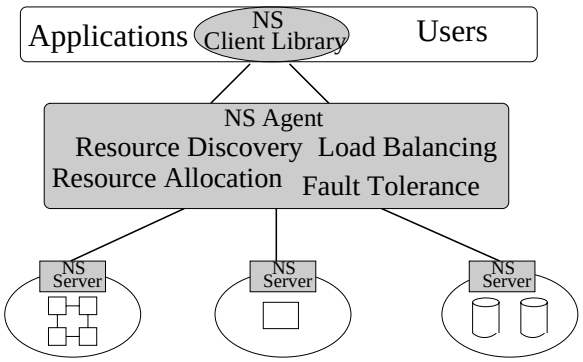

Figure 1: Architectural Overview of the NetSolve System

```
...                                  ...
A = load(input_matrix1);             A = load(input_matrix1);
B = load(input_matrix2);             B = load(input_matrix2);
C = matmul(A, B);                    C = netsolve('matmul', A, B);
...                                  ...
```

Figure 2: Sample Matlab code: Left side before NetSolve, right side after NetSolve integration

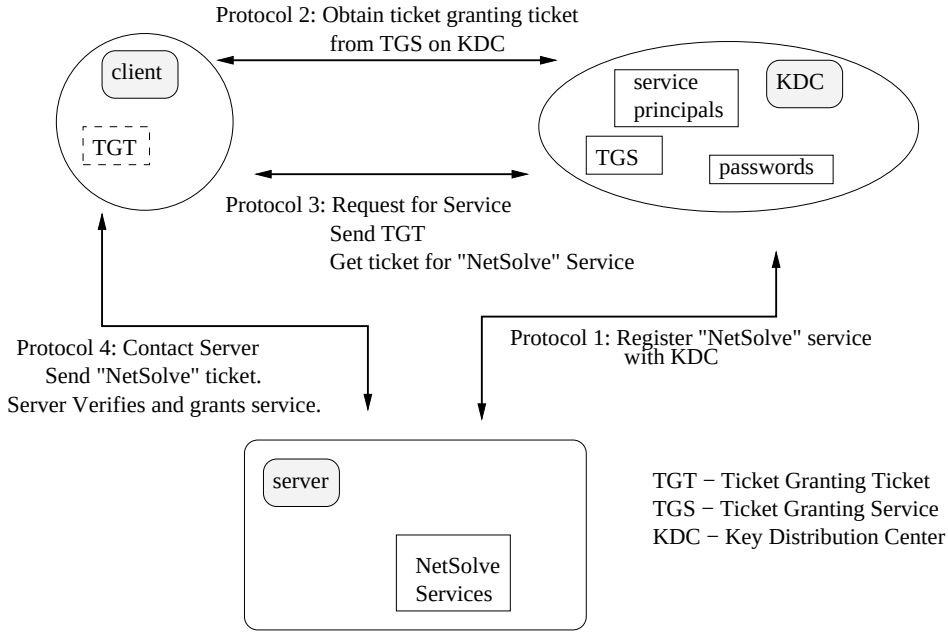

Figure 3: NetSolve/Kerberos Interaction: A NetSolve server "registers" its services with the KDC, which authenticates clients wanting to use that service within NetSolve. The NetSolve server becomes a Kerberos service and the NetSolve client becomes a Kerberos client as well.