The NetSolve Environment: Progressing Towards the Seamless Grid

Dorian C. Arnold and Jack Dongarra
Computer Science Department
University of Tennessee
Knoxville, TN 37996
[darnold, dongarra]@cs.utk.edu

Abstract

The NetSolve software project has matured into a robust system that reliably manages and interconnects disparate computational resources. Resource management and allocation policies, heterogeneity, fault-tolerance and security are some of the issues that need to be resolved in such environments.

This article is meant to introduce the reader to the NetSolve system and offers a discussion of some of the key developments that have taken place in the project during recent months. To make the article coherent and somewhat self-contained, brief insight is given into some of the more fundamental aspects of NetSolve; the newer features and enhancements are given a more detailed discussion. For completion, there is a discussion of successful uses and integrations of the NetSolve system.

1. Introduction

Despite the fact that performance of computer processors continues to increase in accordance with, and even surpass, the rules of Moore's Law, computational scientists are consuming these cycles quicker than the computer engineers can create them. This phenomenom has led to a tremendous thrust of research interests in the concepts of Grid Computing [9]. Grids attempt to harness these cycles and other computational resources into a single, resource more powerful than any of the individual components, whether they be supercomputers, clusters of machines or lower-end workstations. With the "grid" analogy being to that of the electrical power grid, from which we garner electricity without regard for the source of this power, the vision of computational grids is one which has users seamlessly using widely distributed resources without bearing much consideration for this fact.

This article is meant to discuss NetSolve's approach to Grid Computing. NetSolve is a software system based on the concepts of remote procedure call (RPC) that allows for the easy access to computational resources distributed in both geography and ownership. Section 2 of this article gives a general overview of the NetSolve system. Sections 3 through 5 discusses features of NetSolve's three chief components, clients, agents and servers, respectively. We present some examples of NetSolve usage in Sect. 6 and future research plans in Sect. 7.

2. Network-Enabled Solvers

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 a glue layer that brings the application or user together with the hardware and/or software it needs to complete useful tasks.

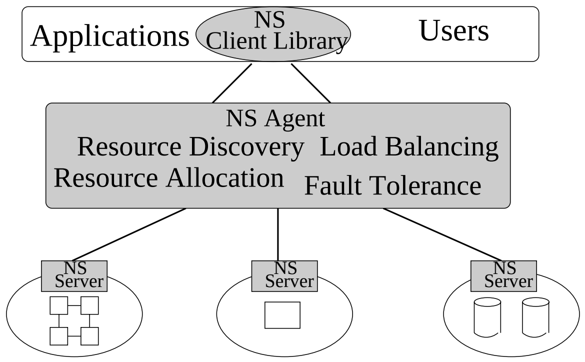

Figure 1. Architectural Overview of the NetSolve System

At the top tier, the NetSolve client library is linked into 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 necessity of user knowledge of computer networking or distributed computing. In fact, the user can often take for granted the fact that remote resources are involved.

Figure 2 helps to show what the programming code would look like before and after the NetSolve API has been integrated. In both cases, it shows a routine `matmul` being called to multiply to matrices `A` and `B` and store the result in matrix `C`. 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.
4. Server returns output parameters or error status to client.

There are many advantages to using a system like NetSolve. NetSolve can provide access to otherwise unavailable software. In cases where the software is in hand, it can make the power of supercomputers accessible from low-end machines like laptop computers. Furthermore, as explained below, NetSolve adds heuristics that attempt to find the most expeditious route to a problem's solution set. NetSolve currently supports the C, FORTRAN, Matlab, and Mathematica programming interfaces as languages of implementation for client programs.

```
...                          ...
A = read_matrix();           A = read_matrix();
B = read_matrix();           B = read_matrix();
C = matmul(A, B);            status = netsolve("matmul", A, B, C);
....                         ...
```

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

The NetSolve agent represents the gateway to the NetSolve system. It serves as an information service maintaining data regarding NetSolve servers and 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, which are marked as unusable. The agent also implements 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 or program being incorporated.

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 [1] for details of the system not discussed in this article.

3. Client Concepts

There are several features that have been implemented for the NetSolve client API to provide users with added flexibility and performance. For brevity, this section only sheds light on some of them.

3.1. The Basics

NetSolve is a functional environment in which the API is used to pass NetSolve objects to and from services as inputs and outputs. The object types are `MATRIX`, a two-dimensional array, `SPARSEMATRIX`, a two-dimensional array stored in a compressed row storage format, `VECTOR`, a one dimensional array, `SCALAR`, `STRING`, an array of characters, `STRINGLIST`, an array of strings, `FILE`, and `UPF`, a user-provided function. The datatypes of these objects may be single or double precision integers and floating point numbers, complex numbers or characters. The NetSolve client interface supports both synchronous and asynchronous calls. The synchronous call transfers control to the NetSolve system which invokes a request on behalf of the client and blocks until the results are available. At this point, it returns control to the user application layer with the results. The asynchronous version of the call passes the request to the NetSolve system and returns immediately. The client program is passed a handle to the request with which it can probe to see if the results are available and gather them when they are.

3.2. Data Persistence

Figure 3 illustrates the typical transactions that take place during a series of NetSolve requests by a single client. What is relevant in this example is that parameter `A` is shared as an input for the first and second requests. Also, output parameters `C` and `D` serve as inputs for subsequent requests. This is exactly the type of scenario that the work described in this section tries to exploit.

We have explored a technique we call request sequencing in an attempt to maximize request throughput by minimizing data transmission between NetSolve components. This interface allows the user to group two or more regular NetSolve requests into a sequence. The system then analyzes the input and output parameters of the sequence to determine when the same parameter occurs more than once and keeps those parameters persistent at or near the server(s) that are servicing the requests. More specifically, we construct a directed acyclic graph (whose nodes represent computational modules and arcs represent the data dependencies of those modules) and schedule this DAG for execution. Our hypothesis is that this reduction in data traffic will yield enough performance improvements to outweigh the overhead of the DAG construction and make sequencing worthwhile. Figure 4 shows the reduction in data flow that occurs when the sequencing mode is employed.

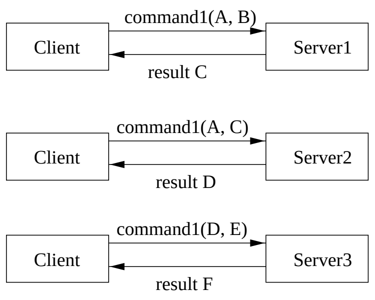

Figure 3. Client-server interactions during a typical reqest scenario.

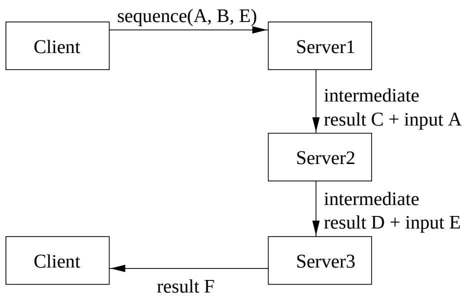

Figure 4. Client-server interactions during a "request sequence".

[2] discusses this interface and gives some experimental results that were achieved using this data persistence technique. The results support our theories that this is a very good way to optimize the use of Grid resources, especially when bandwidth is of the essence, data sets are very large, or both.

3.3. Client Proxies

There are several motivating factors that have led to the implementation of client proxies to act on behalf of the NetSolve client. The proxy, a separate process that resides on the client host, handles (almost) all interactions with the underlying metacomputing resources. The primary reasons for the addition of a proxy to the NetSolve framework are:

- **Lightening the client libraries**
  By separating the interactions with meta-computing resources from the NetSolve client interface, it becomes much more lightweight. One advantage of this scenario is that it increases the uniformity with which the supported client interfaces can be developed. In other words, this modularity means that whenever a feature is added or an enhancement made at the proxy level or below, it only need be implemented once and immediately becomes available to all interfaces.

- **More flexibility on the client side**
  Since the proxy is a separate process with its own thread of control, it is able to interact with meta-computing resources, independently of client interaction. This makes it possible to do things like query the information service, even before the client makes a request, and cache results locally so they can be made immediately available.

- **Integration with other systems**
  One of the philosophies of the NetSolve project is to leverage existing services whenever feasible. Having a proxy negotiate for metacomputing resources on behalf of the client means that different proxies can negotiate for different services. So far, we have implemented proxies to negotiate for Globus services and, of course, the standard NetSolve services. Other systems like Legion are soon to be integrated in this fashion.

- **Supporting more client languages**
  With a standard interface between the client and all proxies, it is possible, especially for third party developers, to easily add new language support to the NetSolve system. They would simply write libraries that interface the NetSolve proxies from their language of choice, allowing programs of that language to become NetSolve-enabled.

Figure 5 depicts the main idea behind the proxy. The client libraries interact with the proxy thanks to a standard API and the proxy interacts with the meta-computing system using system-specific mechanisms. The NetSolve proxy, for instance, uses the agent to discover services, contacts the appropriate server and establishes a session with that server who then receives input data from the client, executes his service and return output data.

3.4. Task Farming

This interface addresses applications that have simple task-parallel structures but require large number of computational resources. Monte-Carlo simulations and parameter-space searches are some popular examples of these applications which are called `task farming` applications. Within the NetSolve system, we designed and built an easily accessible computational framework for task farming applications. As the middleware, NetSolve becomes responsible for the details of managing the Grid services – resource selection and allocation, data movement, I/O, and fault-tolerance.

The task farming interface allows the user to make a single call to netsolve, requesting multiple instances of the same problem. Rather than single parameters, the user passes arrays of parameters as input/output and designates how NetSolve should iterate across the arrays for the task farm. The main challenge in this effort is scheduling. Indeed, for long running farming applications it is to be expected that the availability and workload of resources within the server pool will change dynamically. [8] discusses the design and validation of the NetSolve task farming interface. It also presents an adaptive scheduling algorithm used by the task farming interface to assign tasks to the server resources.

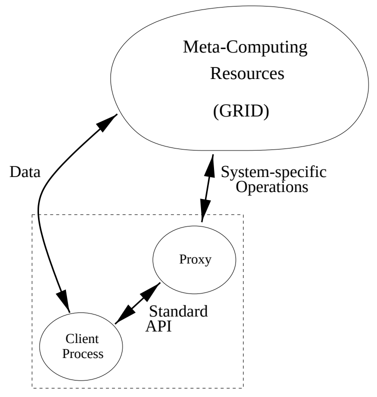

Figure 5. Proxy Architecture

3.5. Transparent Algorithm Selection

Through NetSolve, users are given access to complex algorithms that solve a variety of types of problems, one instance being linear systems solvers. All solvers, however, are not built alike; depending on the characteristics of the matrix being solved some perform poorly and others not at all. NetSolve has incorporated a large number of solver algorithms from a variety of packages like PETSc [5] and Aztec [10]. We have further created an interface that allows the user to generically call a "LinearSolve" routine which transparently analyzes the input matrix and determines which algorithm to use based on input characteristics. [3] further discusses this interface and the heuristics and decisions that are involved in the algorithm selection process. This interface allows the non-expert user to properly and efficiently use solver algorithms without climbing the steep learning curve that would otherwise be involved. This feature exemplifies the ease-of-computing that is only one of the benefits of using a system like NetSolve.

4. The Agent

Though the NetSolve system is merely based upon the concept of remote procedure call (RPC), one may easily overlook the other features it entails and consider it *just* an RPC system. NetSolve is a robust environment that is able to discover, maintain and control a heterogeneous environment of vastly distributed computational resources. The purpose of the agent is to be that point in the system from which to manage and access the resources. The agent always has a complete view of the status of all NetSolve server components and the interactions they may be having with clients. It keeps track of the software capabilities of the servers and is able to direct client requests to appropriate servers. Most all information is “pushed” to the agent; however, previously registered servers must periodically contact the agent, if only to say “I am alive.” The agent uses this fixed periodicity to determine when a server seems to have gone away and updates the database accordingly.

4.1. Network Weather Service Forecasters

When allocating resources, the agent has two goals: i) to choose the best-suited computational server for each incoming request and ii) to keep the load as balanced amongst the servers as possible. The scheduling policies used to make the allocation decisions are fully discussed in [7]; however, we have employed the use of the Network Weather Service (NWS) [13] to help us gather the information necessary to make these decisions. The NetSolve agent calls upon the services of NWS forecasters to predict future availability of server hosts based upon previously collected availability data (see Section 5.2). The forecasters are actually an interface to a variety of prediction modules which take as input a time series of measurements and returns a prediction of the next value. High levels of accuracy are achieved since a prediction error of each module is calculated, and the prediction with the lowest error is used. Previously, NetSolve would look at the last recorded value of a servers status and use that to represent the resources that a server would have available to service a request. NWS allows NetSolve to do a much better job of “guessing” what a server’s availability will be, especially when servers experience high variations in workload.

5. Server Stuff

5.1. Access Control Mechanism

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 [11] services to add this functionality to NetSolve, as it is one of the most trusted and popular infrastructures for authentication services. Using Kerberos, the “administrator” of the server identifies authorized clients using their Kerberos principal, or id. He places his list of clients in a file that is readable only by the user-id that will be used to run the NetSolve server. The only other interaction needed to “kerberize” the server is to create a principal that is used to identify the NetSolve service within the Kerberos realm. Every other involvement needed is just as it would be in non-Kerberos mode. This feature has been implemented such that kerberized and non-kerberized client and server components can gracefully interact with each other. Kerberized servers simply deny service to non-authenticated clients, while clients configured to send authentication credentials will only do so upon demand by a kerberized server.

5.2. NWS CPU Sensors

NetSolve servers must be able to attain and report information regarding their workload so that the agent can determine which servers represent the best choice to service a request. NetSolve has its own version of a `workload manager` that uses UNIX system services like `uptime` to query for workload information. Section 4.1 discusses our integration of NWS forecasters and motivates our use of the NWS CPU sensors as well. There are several features that make this NWS component attractive to the NetSolve project. The NetSolve workload manager was a simple way for the system to implement a feature that at the time was not available any other way; NWS is a project whose purpose is to make this type of service available, and thus much time and expertise has been invested to make sure the reports are as accurate as possible. Secondly, NWS sensors interact with a separate process, called a `memory` to store the data being collected. This process need not be run on the host being monitored, therefore, we can conveniently place the NWS memory on the host running the NetSolve agent (and NWS forecaster) making the information readily available when a forecast is needed. Finally, it offers the uniformity of using the NWS components together. Otherwise, we would have to modify the NetSolve system so it is somehow able to feed the relevant information to the forecaster that helps the agent allocate resources.

6. NetSolve Examples

6.1. Sub-surface Modelling

The implicit parallel accurate reservoir simulator, IPARS, developed at the University of Texas' Institute for Computational and Applied Mathematics, is a framework for developing parallel models of subsurface flow and fluid transport through porous media. It simulates single phase (water only), two phase (water and oil) or three phase (water, oil and gas) flow through a multi-block 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.

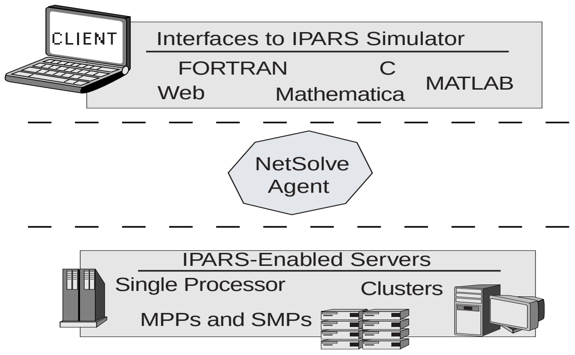

Figure 6. Integrating a fluid flow simulator with NetSolve.

A NetSolve interface to the IPARS system (Fig. 6) allows users to access IPARS (including post-processing of the output to create animated images that exhibit the variations of concentration, pressure, etc. of relevant fluids) [4]. The interface is primarily used from handy machines like laptop computers to run real-time simulations on clusters of workstations that allow for much quicker execution. IPARS runs primarily on LINUX; NetSolve makes it readily accessible from any platform. In addition, we have created a problem solving environment (PSE) interfaced by a web browser which one can use to enter input parameters and submit a request for execution of the IPARS simulator to a NetSolve system. The output images are then brought back and displayed by the web browser. This interaction shows 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 administrate.

6.2. Cellular Microphysiology

MCell is a general Monte Carlo simulator of cellular microphysiology which uses Monte Carlo diffusion and chemical reaction algorithms in 3D to simulate the complex biochemical interactions of molecules inside and outside of living cells. MCell is a collaborative effort between the Terry Sejnowski lab at the Salk Institute, and the Miriam Salpeter lab at Cornell University.

NetSolve's farming interface is very well suited to MCell's needs. One of the central pieces of that framework is a scheduler that takes advantage of MCell input data requirements to minimize execution turn-around time. This scheduler is part of the larger AppLeS [6] at the University of California, San Diego. The use of NetSolve isolates the scheduler from the resource-management details and allows researchers to focus on the scheduler design.

6.3. Nuclear Engineering

The goal of this project is to develop a prototype environment for the Collaborative Environment for Nuclear Technology Software (CENTS). CENTS aims to lay the foundation for a Web-based distance computing facility for executing nuclear engineering codes. Through its Web-based interfaces, CENTS will allow users to focus on the problem to be solved instead of the specifics of a particular nuclear code. Via the Web, users will submit input data with computing options for execution, monitor the status of their submissions, retrieve calculational results, and use CENTS tools for viewing and analyzing result data.

For computational services, CENTS employs a collection of heterogeneous computer systems logically clustered and managed for optimal resource utilization. The prototype environment was accomplished by integrating the NetSolve system and using Monte Carlo Neutral Particle (MCNP) codes via NetSolve's framework. The user is required only to supply the input problem for the MCNP code. After the user supplies the input, NetSolve sends the problem to the most suitable workstation in the environment; the problem is solved, and the output (4 files) is sent back to the user via the web interface.

7. Current Happenings

The NetSolve model is currently being evaluated 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, but use them only as means to do research and simulations in their respective domains, whether it is nuclear engineering or computational chemistry. NetSolve will be especially helpful when lots of computational power is needed to do "embarassingly parallel" tasks, though as discussed below, we will continue to support the efficient execution of parallel jobs in which message passing is eminent. We see NetSolve as an infrastructure that enables scientific communities at different organizations to leverage each others resources and corporate in such a way that aggregates their computational resources into a seamless network that they all benefit from. This section discusses features that are being investigated by the NetSolve project to achieve these ends.

7.1. Dynamically Extensible Servers

One issue that needs to be resolved when creating a giant corporation of NetSolve systems is that of software availability. Especially when the collaboration is an interdisciplinary one, the desired software modules may not always be available at servers to which the client has access. The solution is twofold: i) design and implement servers such that new software modules can be added to their configuration on-the-fly and ii) design and implement a strategy which can be used to publish the availability of new software modules to the servers. In today's version of NetSolve the binding between the server's hardware and software components is configured at server compilation and is statically fixed once the server is running. Our new approach will be to allow the server to do a just-in-time binding of these components when a request is to be serviced. What we must do is generalize a plug-in interface that allows users to dynamically extend and customize the environment's features; one way of doing this would be to implement the modules as dynamically linked libraries (dll) and attaching them to a driver when their services are called upon. We are also developing a software repository component that will be used to house the dlls. Servers will be able to obtain software dynamically from trusted repositories, while repository administrators will be able to control the policies regarding the addition of software.

7.2. Enhanced Client Flexibility

The introduction of the NetSolve client proxy (Sect. 3.3) already progresses us toward a very flexible client layer, that is able to interact with other metacomputing environments. Standardizing the interface between the client and proxy allows third party developers to easily extend NetSolve's client language support. By writing the very thin layer that interfaces our proxies, one can easily interface NetSolve from other languages as well as GUI toolkits and other PSEs. This is depicted in Fig. 7.

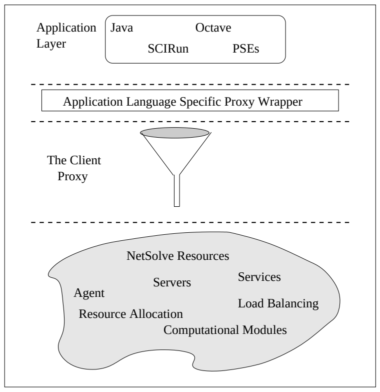

Figure 7. Adding new languages to the NetSolve client interface is as simple as implementing a wrapper in that language to interact with the proxy.

Efforts to improve the flexibility of the client interactions also include the addition of multiple scheduling policies that allow the user to more precisely determine which resources they should use during an execution. Currently, the NetSolve agent handles all scheduling policies based mainly on network and CPU characteristics, however, the client user may be aware of more specific scheduling information like the fact that a particular code may perform better on a certain architecture, or that he is farming a multitude of tasks and the system need not be concerned with starvation amongst particular requests, but rather the turnaround time of the entire task farm. The AppLeS [6] project has done much research in application specific schedulers, and are currently developing templates that are able to efficiently schedule generic classes of applications, (for instance, independently parallel.) These scheduling templates will be the foundation of our schedulers; users will be able to use a particular template based on the nature of their NetSolve interactions.

7.3. Parallel Programming Support

One of the key features of NetSolve is its ability to seamlessly connect users to a host of parallel algorithms and codes. We are increasing our support of the "service" developers by implementing a variety of data distribution/collection schemes particularly for the NetSolve environment. These schemes will be designed to counter current strategies that collect all the data at a single node and then distribute, which defeats the purpose of distributing large jobs whose data is too large to be contained at a single node. Whenever possible, we will marshall the data directly from the client to all computational nodes involved, and collect results in a similar fashion. The distribution suite will include popular algorithms like block distribution, multi-dimensional block distributions and block cyclic distributions.

For parallelizing programs from the client-end, using either the asynchronous, sequencing or task farming interfaces, we are investigating the use of distributed storage facilities like IBP [12] to make it possible for the user to pre-store commonly used data strategically near server(s) before execution begins; furthermore, he will be able to conveniently use handles to these locations as parameters in NetSolve requests. We will also continue the development of these interfaces which in some cases attempt to do some data staging automatically.

8. Conclusion

As researchers continue to investigate feasible ways to harness computational resources, the NetSolve system is emerging into one of the more popular solutions. Its light weight and ease of use make it an ideal candidate for middleware. As we discover the needs of computational scientists, the NetSolve system will be molded to facilitate these users. It is built from the user's perspective – convenient interface design and ease of administration are most important in the NetSolve philosophy, and every effort is made not to sacrifice these elements as the system evolves.

References

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

[2] D. C. Arnold, D. Bachmann, and J. Dongarra. Request Sequencing: Optimizing Communication for the Grid. In *Euro-Par 2000 – Parallel Processing*, August 2000.

[3] D. C. Arnold, S. Blackford, J. Dongarra, V. Eijkhout, and T. Xu. Seamless Access to Adaptive Solver Algorithms. August 2000.

[4] D. C. Arnold, W. Lee, J. Dongarra, and M. Wheeler. Providing Infrastructure and Interface to High-Performance Applications in a Distributed Setting. In A. Tentner, editor, *High Performance Computing 2000*, pages 248–253. Society for Computer Simulation International, April 2000.

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

[6] F. Berman, R. Wolski, S. Figueira, J. Schopf, and G. Shao. Application-Level Scheduling on Distributed Heterogeneous Networks. In *Proc. of Supercomputing'96, Pittsburgh, PA*, November 1996.

[7] H. Casanova and J. Dongarra. NetSolve: A Network Server for Solving Computational Science Problems. *The International Journal of Supercomputer Applications and High Performance Computing*, 1997.

[8] H. Casanova, M. Kim, J. S. Plank, and J. Dongarra. Adaptive Scheduling for Task Farming with Grid Middleware. *The International Journal of Supercomputer Applications and High Performance Computing*, 1999. to appear.

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

[10] S. A. Hutchinson, S. J. N., and T. R. S. Aztec user's guide: Version 1.1. Technical Report SAND95-1559, Sandia National Laboratories, 1995.

[11] B. C. Neuman and T. Ts'o. Kerberos: An Authentication Service for Computer Networks. *IEEE Communications*, 32(9):33–38, September 1994.

[12] J. Plank, M. Beck, W. Elwasif, , T. Moore, M. Swany, and R. Wolski. IBP – The Internet Backplane Protocol: Storage in the Network. In *NetStore '99: Network Storage Symposium, Seatle, WA*, October 1999.

[13] R. Wolski. Dynamically Forecasting Network Performance Using the Network Weather Service. Technical Report TR-CS96-494, U.C. San Diego, October 1996.