NetSolve: a Network-Enabled Solver; Examples and Users.

Henri Casanova

Dept. of Computer Science
University of Tennessee
Knoxville, TN 37996-1301

Jack J. Dongarra

Dept. of Computer Science
University of Tennessee
Knoxville, TN 37996-1301
and
Mathematical Science Section
Oak Ridge National Laboratory
Oak Ridge, TN 37821-6367

Abstract

The NetSolve project, underway at the University of Tennessee and at the Oak Ridge National Laboratory, allows users to access computational resources distributed across the network. These resources are embodied in computational servers and allow the user to easily perform scientific computing tasks without having any computing facility installed on his/her computer. The user access to the servers is facilitated by a variety of interfaces: Application Programming Interfaces (APIs), Textual Interactive Interfaces and Graphical User Interfaces (GUIs). There are many research issues involved in the NetSolve system, including fault-tolerance, load balancing, user-interface design, computational servers, and network-based computing. As the project matures, several promising extensions and applications of NetSolve will emerge. In this article, we provide an overview of the project and examine some of the extensions being developed: An interface to the Condor system, an interface to the ScaLAPACK parallel library, a bridge with the Ninf system, and an integration of NetSolve and ImageVision.

1   The NetSolve project

1.1   Basics

Thanks to advances in hardware, networking infrastructure and algorithms, computing intensive problems in many areas can now be successfully attacked using networked, scientific computing. In the networked computing paradigm, vital pieces of software and information used by a computing process are spread across the network, and are identified and linked together only at run time. This is in contrast to the current software usage model where one acquires a copy (or copies) of task-specific software package for use on local hosts. One can distinguish three main paradigms for such systems. 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. NetSolve uses the *remote computing* paradigm: the program resides on the server; the user's data is sent to the server, where the appropriate programs or numerical libraries operate on it; the result is then sent back to the user's machine.

Figure 1 depicts the typical layout of the system. NetSolve provides the user with a pool of computational resources. These resources are computational servers that have access to ready-to-use numerical software. As shown in the figure, the computational servers can be running on single workstations, net-
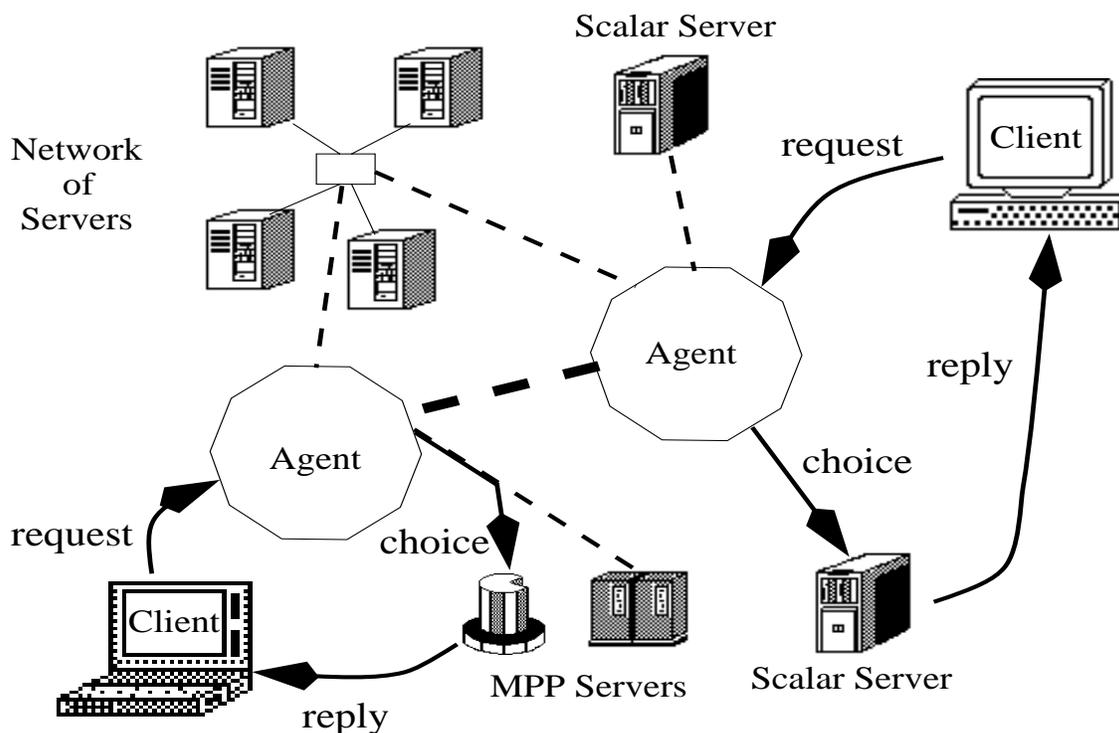
Figure 1: **NetSolve's organization**

works of workstations that can collaborate for solving a problem, or MPP (Massively Parallel Processor) systems. The user is using one of the NetSolve client interfaces. Through these interfaces, he can send requests to the NetSolve system asking for his numerical computation to be carried out by one of the servers. The main role of the NetSolve agent is to process this request and to choose the most suitable server for this particular computation in terms of execution time. Once a server has been chosen, it is assigned the computation, uses its available numerical software, and eventually returns the results to the user. One of the major advantages of this approach is that the agent performs load-balancing among the different resources.

As shown on Figure 1, there can be multiple instances of the NetSolve agent on the network, and different clients can contact different agents depending on their locations. The agents can exchange information about their different servers and allow access from any client to any server if desirable. NetSolve can be used either via the Internet or on an intranet, such as inside a research department or a university, without participating in any Internet based compu-

tation. Another important aspect of NetSolve is that the configuration of the system is entirely flexible: any server/agent can be stopped and (re-)started at any time without jeopardizing the integrity of the system.

1.2 The computational resources

When building the NetSolve system, one of the challenges was to design a suitable model for the computational servers. The NetSolve servers are configurable so that they can be easily upgraded to encompass ever-increasing sets of numerical functionalities. The NetSolve servers are also pre-installed, meaning that the end-user does not have to install any numerical software. Finally, the NetSolve servers provide uniform access the the numerical software, in the sense that the end-user has the illusion that she is accessing numerical subroutines from a single, coherent numerical library.

To make the implementation of such a computational server model possible, we have designed a general, machine-independent way of describing a numerical computation, as well as a set of tools to generate new computational modules as easily as possible. The main component of this framework is a *descriptive language* which is used to describe each separate nu-

merical functionality of a computational server. The description files written in this language can be compiled by NetSolve into actual computational modules executable on any UNIX or NT platform. These files can then be exchanged by any institution wanting to set up servers: each time a new description file is created, the capabilities of the entire NetSolve system are increased.

A number of description files have been generated for a variety of numerical libraries: ARPACK [1], FitPack [2], ItPack [3], MinPack [4], FFTPACK [5], LAPACK [6], BLAS [7, 8, 9], QMR [10], and ScaLAPACK [11]. These numerical libraries cover several fields of computational science; Linear Algebra, Optimization, Fast Fourier Transforms, etc.

NetSolve computational servers providing access to these libraries are currently running at the University of Tennessee and at other locations world-wide. Real-time information on the running servers can be found on the NetSolve web-page located at

http://www.cs.utk.edu/netsolve

1.3 The client interfaces

A major concern in designing NetSolve was to provide several interfaces for a wide range of users. Net-Solve can be invoked through C, Fortran, Java, as well as on Matlab. In addition, there is a Web-enabled Java GUI. Another concern was keeping the interfaces as simple as possible. For example, there are only two calls in the MATLAB interface, and they are sufficient to allow users to submit problems to the NetSolve system. Each interface provides asynchronous calls to NetSolve in addition to traditional synchronous or blocking calls. When several asynchronous requests are sent to a NetSolve agent, they are dispatched among the available computational resources according to the load-balancing schemes implemented by the agent. Hence, the user—with virtually no effort—can achieve coarse-grained parallelism from either a C or Fortran program, or from interaction with a high-level interface. All the interfaces are described in detail in the "NetSolve's Client User's Guide" [12].

1.4 The NetSolve agent

1.4.1 The agent as a database

Keeping track of what software resources are available and on which servers they are located is perhaps the most fundamental task of the NetSolve agent. Since the computational servers use the same framework to contribute software to the system (see Section 1.2), it is possible for the agent to maintain a database of different numerical functionalities available to the users.

Each time a new server is started, it sends an application request to an instance of the NetSolve agent. This request contains general information about the server and the list of numerical functions it intends to contribute to the system. The agent examines this list and detects possible discrepancies with the other existing servers in the system. Based on the agent's verdict, the server can be integrated into the system and available for clients.

1.4.2 The agent as a resource broker

The goal of the NetSolve agent is to choose the best-suited computational server for each incoming request to the system. For each user request, the agent determines the set of servers that can handle the computation and makes a choice between all the possible resources. To do so, the agent uses computation-specific and resource-specific information. Computation-specific information is mostly included in the user request whereas resource-specific information is partly static (server's host processor speed, memory available, etc.) and partly dynamic (processor workload). The agent thus provides the user with location transparency for the processor performing her computation. Rationale and further detail on these issues can be found in [13], as well as a description of how NetSolve ensures fault-tolerance among the servers.

1.5 Conclusion

Agent-based computing seems to be a promising strategy. NetSolve will evolve into a more elaborate system in the future and a major part of this evolution is to take place within the agent. Such issues as user accounting, security, data encryption for instance are only partially addressed in the current implementation of NetSolve and will be the object of much work in the future. As the types of hardware resources and the types of numerical software available on the computational servers become more and more diverse, the resource broker embedded in the agent will need to become increasingly sophisticated. There are many difficulties in providing a uniform performance metric that encompasses any type of algorithmic and hardware considerations in a metacomputing setting, especially when different numerical resources, or even entire frameworks are integrated into NetSolve. Such integrations are described in the following sections.

2 An interface to the Condor system

2.1 Overview of Condor

Condor [14, 15, 16], developed at the University of Wisconsin, Madison, is an environment that can manage very large collections of distributively owned workstations. Its development has been motivated by

the ever increasing need for scientists and engineers to exploit the capacity of such collections, mainly by taking advantage of otherwise unused CPU cycles.

A brief description of Condor's software architecture follows. A Condor pool consists of any number of machines, that are connected by a network. Condor daemons constantly monitor the status of the individual computers in the cluster. Two daemons run on each machine, the *startd* and the *schedd*. The *startd* monitors information about the machine itself (load, mouse/keyboard activity, etc.) and decides if it is available to run a Condor job. The *schedd* keeps track of all the Condor jobs that have been submitted to the machine. One of the machine, the *Central Manager*, keeps track of all the resources and jobs in the pool. When a job is submitted to Condor, the scheduler on the central manager matches a machine in the Condor pool to that job. Once the job has been started, it is periodically checkpointed, can be interrupted and migrated to a machine whose architecture is the same as the one of the machine on which the execution was initiated. This organization is partly depicted in Figure 2. More details on the Condor system and the software layers can be found in [14].

2.2 A Condor pool as a NetSolve resource
2.2.1 Motivation

Interfacing NetSolve and Condor is a very natural idea. NetSolve provides remote easy access to computational resources through multiple, attractive user interfaces. Condor allows users to harness the power of a pool of machines while using otherwise unused CPU cycles. The users at the consoles of those machines are therefore not penalized by the scheduling of Condor jobs. If the pool of machines is reasonably large, it is usually the case that Condor jobs can be scheduled almost immediately. This could prove to be very interesting for a project like NetSolve. Indeed, NetSolve servers may be started so that they grant local resource access to outside users. Interfacing NetSolve and Condor could then give priority to local users and provide underutilized only CPU cycles to outside users.

2.2.2 Implementation

Figure 2 shows how an entire Condor pool can be seen as a single NetSolve computational resource. The Central Manager runs two daemons in addition to the usual *startd* and *schedd*: the *negotiator* and the *collector*. One machine also runs a customized version of the NetSolve server. When this server receives a request from a client, instead of creating a local child

process running a computational module, it uses the Condor tools to submit that module to the Condor pool. The negotiator on the Central Manager then chooses a target machine for the computational module. Due to fluctuations in the state of the pool, the computational module can then be migrated among the machines in the pool. When the results of the numerical computation are obtained, the NetSolve server transmits that result back to the client.

The actual implementation of the NetSolve/Condor interface was made easy by the Condor tools provided to the Condor user. However, the restrictions that apply to a Condor jobs concerning system calls were difficult and required quite a few changes to obtain a Condor-enabled NetSolve server. A major issue however still needs to be addressed; how does the NetSolve agent perceive a Condor pool as a resource? Indeed, it is rather difficult to predict when the job will be scheduled or how often it will be suspended and migrated. Finding the appropriate performance prediction technique will be at the focus of the next step in the NetSolve/Condor collaboration.

3 Integrating parallel numerical libraries

3.1 Motivation

Integrating parallel packages into NetSolve will allow a user on a workstation to access MPP systems to perform large computation. This access can be extremely simple and the user may not even be aware that he is using a parallel library.

3.2 Integrating parallel software packages into NetSolve

ScaLAPACK (Scalable Linear Algebra Package) is a library of high-performance linear algebra routines for distributed-memory message-passing MIMD computers as well as networks of workstations supporting PVM [17] or MPI [18]. ScaLAPACK was developed at the University of Tennessee, Knoxville, the Oak Ridge National Laboratory and the University of California, Berkeley. It is a continuation of the LAPACK [6] project, and contains routines for solving systems of linear equations, least squares problems, and eigenvalue problems. ScaLAPACK views the underlying multi-processor system as a rectangular process *grid*. Global data is mapped to the local memories of the processes in that grid assuming specific data-distributions. For performance reasons, ScaLAPACK uses the two-dimensional block cyclic distribution scheme for dense matrix computations. Interprocess communication within ScaLAPACK is done via the BLACS (Basic Linear Algebra Communication
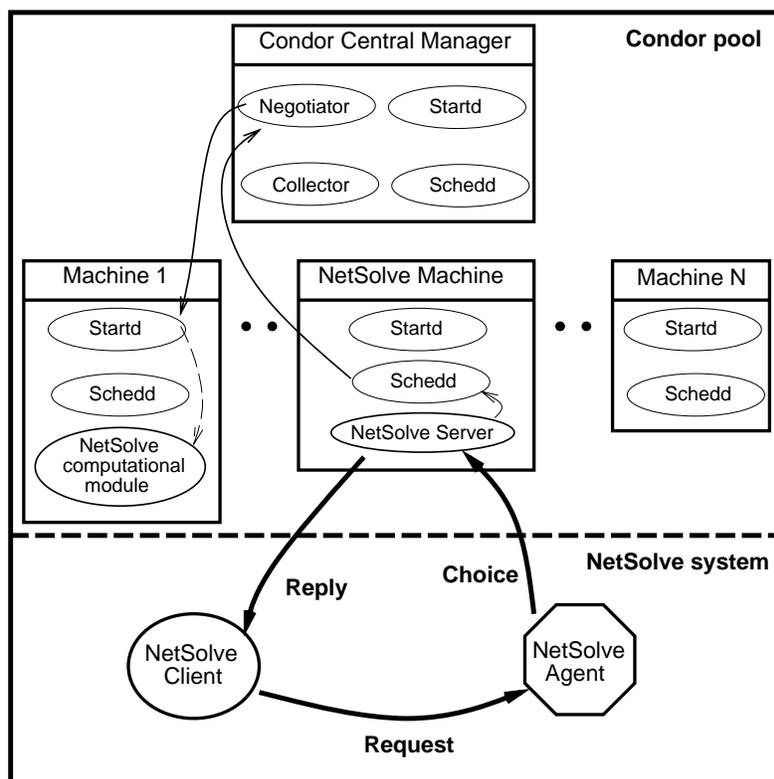
Figure 2: **NetSolve and Condor**

subprograms) [19, 20]. All the details on ScaLAPACK and its software hierarchy can be found in the latest edition of the User's Guide [11].

Figure 3 is a very simple description of how the Net-Solve server has been customized to use the ScaLA-PACK library. The customized server receives data input from the client in the traditional way. The Net-Solve server uses BLACS calls to set up the ScaLA-PACK processor grid. ScaLAPACK requires that the data already be distributed among the processors prior to any library call. This is the reason why each user input is first 2-D block cyclic distributed in that grid when necessary. The server can then initiate the call to ScaLAPACK and wait until completion of the computation. When the ScaLAPACK call returns, the result of the computation is usually available on the processors and is 2-D block cyclic distributed as well. The server then gathers that result and sends it back to the client in the expected format. This process is completely transparent to the user who does not even realize that a parallel execution is taking place.

This approach is very promising. A client can use MATLAB on a PC and issue a simple call like `[x] = netsolve('eig',a)` and have an MPP system use a high-performance library to perform a large eigenvalue computation. We have designed a prototype of the customized server running on top of PVM [17] or MPI [18]. There are many research issues arising with integrating parallel libraries in NetSolve, including performance prediction, choices of processor-grid/matrix-block size, choice of numerical algorithm, processor availability, accounting, etc.

4 NetSolve and Ninf

4.1 A brief overview of Ninf

Ninf, developed at the Electrotechnical Laboratory, Tsukuba, is a global network-wide 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. Computational resources are shared as Ninf remote libraries and are executable at remote Ninf servers. Users can build an application by calling the libraries with the Ninf Remote Procedure Call, which is designed to provide a programming interface similar to conventional function calls in existing languages, and is tailored for scientific computation. In order to facilitate location
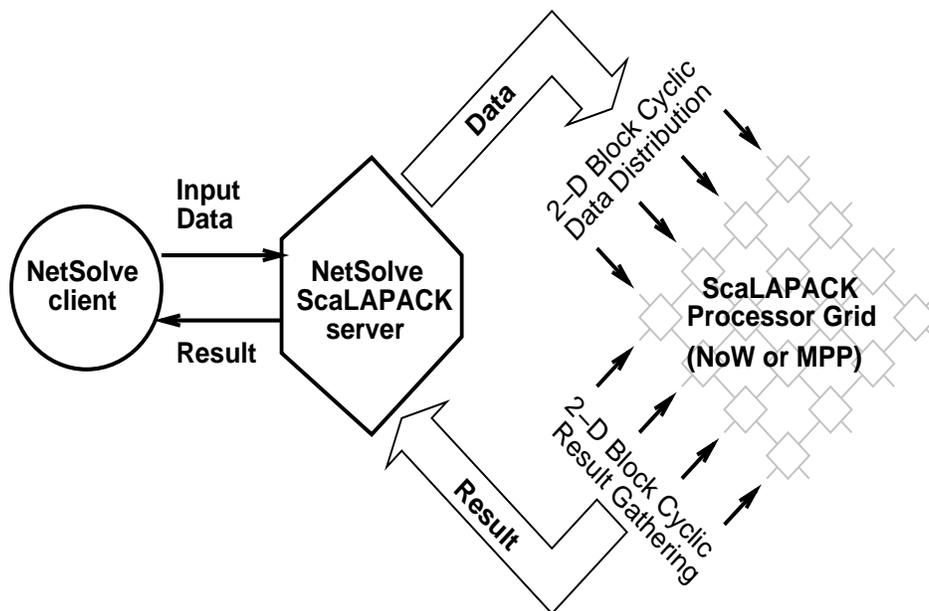
Figure 3: **The ScaLAPACK NetSolve Server Paradigm**

transparency and network-wide parallelism, the Ninf MetaServer maintains global resource information regarding computational server and databases. It can therefore allocate and schedule coarse-grained computations to achieve good global load balancing. Ninf also interfaces with existing network service such as the WWW for easy accessibility. More details on Ninf can be found in [21]. Clearly, NetSolve and Ninf bear strong similarities both in motivation and general design. Allowing the two systems to coexist and collaborate should lead to promising developments.

4.2 A gateway between Ninf and Net-Solve

Some design issues prevent an immediate seamless integration of the two softwares (conceptual differences between the NetSolve agent and the Ninf Metaserver, problem specifications, user interfaces, data transfer protocols, etc.). In order to overcome these issues, the Ninf team started developing two *adapters*: a NetSolve-Ninf adapter and a Ninf NetSolve-adapter. Thanks to those adapters, Ninf clients can use computational resources administrated by a NetSolve system and vice-versa.

Figure 4(i) shows the Ninf-NetSolve adapter allowing access to Ninf resource from a NetSolve client. The adapter is just seen by the NetSolve agent as any other NetSolve server. When a NetSolve client sends a request to the agent, it can then be told to use the NetSolve adapter. The adapter performs protocol trans-

lation, interface translation, and data transfer, asks a Ninf server to perform the required computation and returns the result to the user.

In Figure 4(ii), the NetSolve-Ninf adapter can be seen by the Ninf MetaServer as a Ninf server, but in fact plays the role of a NetSolve client. This is a little different from the Ninf-NetSolve adapter because the NetSolve agent is a resource broker whereas the Ninf MetaServer is a proxy server. Once the adapter receives the result of the computation from some NetSolve server, it transfers that result back to the Ninf client.

There are several advantages of using such adapters. Updating the adapters to reflects the evolutions of NetSolve or Ninf seems to be an easy task. Some early implementation evaluations tend to show that using either system via an adapter causes acceptable overheads, mainly due to additional data transfers. Those first experiments appear encouraging and will definitely be extended to effectively enable an integration of NetSolve and Ninf.

5 Extending ImageVision by the use of NetSolve

In this section, we describe how NetSolve can be used as a building block for a general purpose framework for basic image processing.
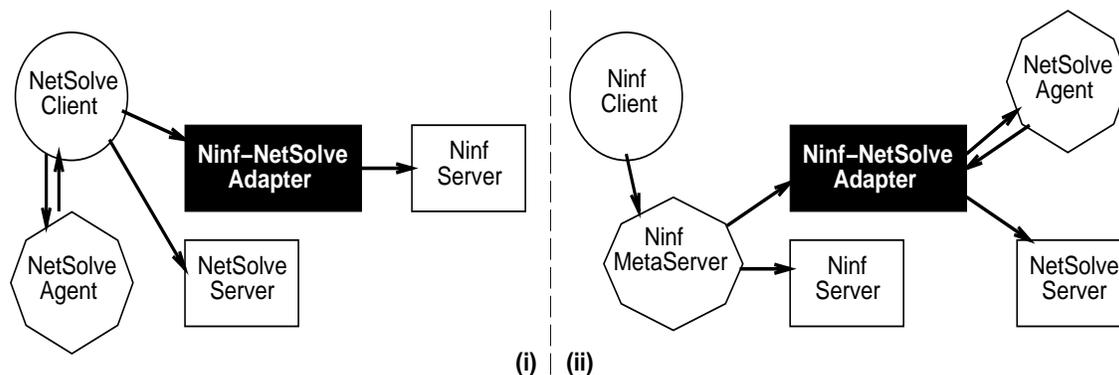
Figure 4: **Going (i) from NetSolve to Ninf and (ii) from Ninf to NetSolve**

5.1 Integrating the ImageVision library into NetSolve

This project is under development at the ICG institute at Graz University of Technology, Austria. The scope of the project is to make basic image processing functions available for remote execution over a network. The goals of the project include two objectives that can be leveraged by NetSolve. First, the resulting software should prevent the user from having to install complicated image processing libraries. Second, the functionalities should be available via Java-based applications. The ImageVision Library (IL) [22] is an object-oriented library written in C++ by Silicon Graphics, Inc. (SGI) and shipped with newer workstations. It contains typical image processing routines to efficiently access, manipulate, display, and store image data. ImageVision has been judged quite complete and mature by the research team at ICG and seems therefore a good choice as an "engine" for building a remote access image processing framework. Such a framework will make IL accessible from any platform (and not only from SGI workstations) and is described in [23].

5.2 NetSolve as an operating environment for ImageVision

The reasons why NetSolve has been a first choice for such a project are diverse. First, NetSolve is easy to understand, use, and extend. Second, NetSolve is freely available. Third, NetSolve provides language binding to Fortran, C, and Java. And finally, NetSolve's agent-based design allows load monitoring and balancing among the available servers. New NetSolve computational modules corresponding to the desired image processing functionalities will be created and integrated into the NetSolve servers. A big part of the project at ICG is to build a Java GUI to IL.

Figure 5 shows a simple example of how ImageVision can be accessed via NetSolve. A Java GUI can be built on top of the NetSolve Java API. As shown on the figure, this GUI could offer visualization capabilities. For computations, it uses an embedded NetSolve client and contacts SGI servers that have access to IL. The user of the Java GUI does not realize that NetSolve is the back end of the system, or that he uses a SGI library without running the GUI on a SGI machine! The protocol depicted on Figure 5 is of course simplistic. In order to obtain acceptable levels of performance, the network traffic needs to be minimized. There are several ways of attacking this problem. For instance, the servers could "keep a state", meaning that some data can be cached on the server for future use. Several issues are involved in the design of such a mechanism as the cache needs proper invalidation mechanisms, replacement policies, etc. Another possibility would be to combine requests to avoid retransmitting redundant data. Such a change can be already emulated my designing appropriate problem description file for the NetSolve servers. However, it may become preferable to include request combination as a standard feature of the NetSolve protocol. The current version of the Java API in GUI in NetSolve allows to reference objects (e.g. images) via URLs. This may prove useful for some applications, and in particular to the ImageVision/NetSolve integration.

6 Conclusion

The scientific community has long used the Internet for communication of email, software, and documentation. Until recently there has been little use of the network for actual computations. This situation is changing rapidly and will have an enormous impact on the future.

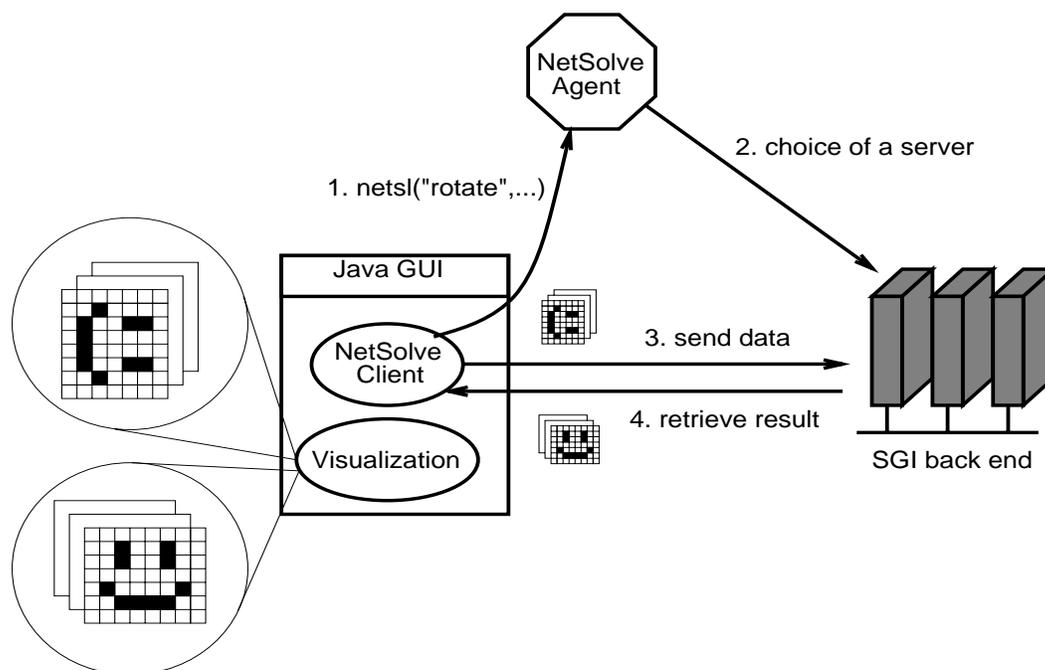We have discussed throughout this article how Net-

Figure 5: **ImageVision and NetSolve**

Solve can be customized, extended, and used for a variety of purposes. We first described in Sections 2 and 3 how NetSolve can encompass new types of computing resources, resulting in a more powerful and flexible environment and raising new research issues. We next discussed in Section 4 how NetSolve and Ninf can be merged into a single metacomputing environment. Finally, in Section 5, we gave an example of an entire application that uses NetSolve as an operating environment to build general image processing framework. All these developments take place at different levels in the NetSolve project and have had and will continue to have an impact on the project itself, causing it to improve and expand.

References

[1] R. Leboucq, D. Sorensen, and C. Yang. *ARPACK Users Guide*. 1997.

[2] A. Cline. Scalar- and Planar-Valued Curve Fitting Using Splines Under Tension. *Communications of the ACM*, 17:218–220, 1974.

[3] D. Young, D. Kincaid, J. Respess, and R. Grimes. Itpack2c: a FORTRAN package for solving large sparse linear systems by adaptive accelerated iterative methods. Technical report, University of Texas at Austin, Boeing Computer Services Company, 1996.

[4] J. Moré, B. Garbow, and K. Hillstrom. Minpack : Documentation file accessible at: "http://www.netlib.org/minpack/readme".

[5] P. Swarztrauber. FFTPACK : Documentation file accessible at: "ftp://ftp.ucar.edu/ftp/dsl/lib/fftpack/readme".

[6] E. Anderson, Z. Bai, C. Bischof, J. Demmel, J. Dongarra, J. Du Croz, A. Greenbaum, S. Hammarling, A. McKenney, S. Ostrouchov, and D. Sorensen. *LAPACK Users' Guide, Second Edition*. SIAM, Philadelphia, PA, 1995.

[7] C. Lawson, R. Hanson, D. Kincaid, and F. Krogh. Basic Linear Algebra Subprograms for Fortran Usage. *ACM Transactions on Mathematical Software*, 5:308–325, 1979.

[8] J. Dongarra, J. Du Croz, S Hammarling, and R. Hanson. An Extended Set of Fortran Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 14(1):1–32, 1988.

[9] J. Dongarra, J. Du Croz, I. Duff, and S. Hammarling. A Set of Level 3 Basic Linear Algebra Subprograms. *ACM Transactions on Mathematical Software*, 16(1):1–17, 1990.

[10] R.W. Freund and N.M. Nachtigal. QMR: A quasi-minimal residual method for non-Hermitian linear systems. *Numer. Math.*, 60:315–339, 1991.

[11] 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.

[12] H. Casanova, J. Dongarra, and K. Seymour. Client User's Guide to Netsolve. Technical Report CS-96-343, Department of Computer Science, University of Tennessee, 1996.

[13] 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.

[14] M. Litzkow, M. Livny, and M.W. Mutka. Condor - A Hunter of Idle Workstations. In *Proc. of the 8th International Conference of Distributed Computing Systems*, pages 104–111. Department of Computer Science, University of Winsconsin, Madison, June 1988.

[15] M. Litzkow and M. Livny. Experience with the Condor Distributed Batch System. In *Proc. of IEEE Workshop on Experimental Distributed Systems*. Department of Computer Science, University of Winsconsin, Madison, 1990.

[16] J. Pruyne and M. Livny. A Worldwide Flock of Condors : Load Sharing among Workstation Clusters. *Journal on Future Generations of Computer Systems*, 12, 1996.

[17] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, and V. Sunderam. *PVM : Parallel Virtual Machine. A Users' Guide and Tutorial for Networked Parallel Computing.* The MIT Press Cambridge, Massachusetts, 1994.

[18] M. Snir, S. Otto, S. Huss-Lederman, D. Walker, and J. Dongarra. *MPI : The Complete Reference.* The MIT Press Cambridge, Massachusetts, 1996.

[19] J. Dongarra and R. van de Geijn. Two dimensional basic linear algebra communication subprograms. Technical Report CS-91-138, Computer Science Dept, University of Tennessee, Knoxville, TN, 1991. Also LAPACK Working Note #37.

[20] R.C. Whaley and J. Dongarra. A user's guide to the BLACS v1.1. Technical Report CS-95-281, Computer Science Dept, University of Tennessee, Knoxville, TN, 1995. Also LAPACK Working Note #118.

[21] 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*, 1996.

[22] G. Eckel, J. Neider, and E. Bassler. *ImageVision Library Programming Guide.* Silicon Graphics, Inc., Mountain View, CA, 1996.

[23] M. Oberhuber. Integrating ImageVision into NetSolve. Available at http://www.icg.tugraz.ac.at/mober/pub, October 1997.

Biographies

Jack J. Dongarra holds a joint appointment as Distinguished Professor of Computer Science in the Computer Science Department at the University of Tennessee (UT) and as Distinguished Scientist in the Mathematical Sciences Section at Oak Ridge National Laboratory (ORNL) under the UT/ORNL Science Alliance Program. He specializes in numerical algorithms in linear algebra, parallel computing, use of advanced-computer architectures, programming methodology, and tools for parallel computers. Other current research involves the development, testing and documentation of high quality mathematical software. He was involved in the design and implementation of the software packages EISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, the National High-Performance Software Exchange and NetSolve; and is currently involved in the design of algorithms and techniques for high performance computer architectures.

Professional activities include membership in the Society for Industrial and Applied Mathematics (SIAM), the Institute of Electrical and Electronics Engineers (IEEE), the Association for Computing Machinery (ACM), and a Fellow of the American Association for the Advancement of Science (AAAS).

He has published numerous articles, papers, reports and technical memoranda, and has given many presentations on his research interests.

http://www.netlib.org/utk/people/JackDongarra/

Henri Casanova earned the Applied Mathematics and Computer Science Engineer degree from

the Ecole Nationale Supérieure d'Electrotechnique, d'Electronique, d'Informatique et d'Hydraulique de Toulouse (ENSEEIHT), Toulouse, France in 1993, as well as the Diplôme d'Etudes Approfondies in Parallel Architectures and Software Engineering from the Université Paul Sabatier, Toulouse, France. In 1992-1993 he was a trainee at the Institut de Recherche en Informatique de Toulouse (IRIT), in Toulouse, France. From November 1993 until November 1994 he did his military service working from the French Ministry of Defense (DGA) as Advisor for the Computer Science Division of the DPAG. In January 1995, he entered the PhD program in Computer Science at the University of Tennessee, Knoxville and has been working as a Graduate Research Assistant since then.

His research interests are diverse and include meta-computing, Internet-based computing, parallel and scientific computing, stochastic modeling (Markov chains, Large Deviation Theory) and performance prediction for distributed computing applications. Casanova is the main developer of the NetSolve project and still maintains the major part of the software.