Metacomputing Support for the SARA3D Structural Acoustics Application

Shirley Moore[+], Dorian Arnold, and David Cronk
Innovative Computing Laboratory
University of Tennessee-Knoxville

This paper reports on using the NetSolve metacomputing system to provide remote access to HPC platforms for users of the SARA3D structural acoustics code. NetSolve is a client/server system that provides access to HPC hardware and software from familiar desktop environments, using a variety of client interfaces, including Fortran, C, and Java programming interfaces, and MATLAB and Mathematica interactive interfaces. SARA3D is a finite element program for computing the frequency response of general, three-dimensional structural acoustics problems. NetSolve is being used to provide real-time response for post-processing and visualization of large-scale SARA3D output data. SARA3D has been implemented as a NetSolve service which runs on HPC machines. SARA3D can be invoked remotely from the NetSolve MATLAB or Fortran or C client interfaces on the user's workstation. The SARA3D input file is transferred by NetSolve from the client to the server, and the SARA3D output is transferred from the server to the client when the SARA3D computation is finished. Nonblocking NetSolve calls are used to obtain coarse-grained parallelism.

Introduction

SARA3D is typical of many DoD applications in that it has a computational phase followed by a post-processing phase. In the case of SARA3D, the computational phase solves a coupled fluid-structure model to produce a large output file containing displacements and pressure throughout the model. The output data many then be post-processed in various ways to produce smaller data sets suitable for analysis and visualization.

Previously, the SARA3D computation and post-processing were carried out on large HPC machines at ERDC. The computational portion has been parallelized using a combination of MPI and OpenMP. The post-processing portion uses non-parallel Fortran subroutines which output a MATLAB data file. The MATLAB file is transferred to the user’s home machine for analysis and/or visualization. Because the post-processing is serial, it is a bottleneck for the performance of SARA3D.

The use of NetSolve allows the user to interactively invoke and control execution of the computational and post-processing portions of SARA3D. The SARA3D input file is transferred by NetSolve from the NetSolve client running on the user’s workstation to a NetSolve server that provides the SARA3D service. Depending on the input file, the NetSolve request may invoke the main SARA3D computation and/or a post-processing operation. A NetSolve request may invoke post-processing on an output file that was

[+] Project Principal Investigator. Email address: shirley@cs.utk.edu

generated from a previous SARA3D computation. Nonblocking NetSolve calls allow post-processing operations to be performed concurrently, thus obtaining coarse-grained parallelism without requiring any parallel code to be written. Because the separate post-processing tasks require no intercommunication, the parallel speedup is essentially linear.

Current work involves building a repository of SARA3D output data that may be shared among users. In addition, we are constructing a Web interface to the NetSolve SARA3D service that will enable users to invoke and control SARA3D computations and post-processing from a Web browser.

2 SARA3D

SARA3D is a finite element program for computing the frequency response of general, three-dimensional structural acoustics problems[1]. SARA3D can also be used to solve structural vibrations, radiation, scattering, and electroelasticity problems. In structural acoutistics problems, SARA3D solves the time harmonic problem of a structure submerged in an infinite fluid subjected to incident traveling waves or to vibrational loads within the structure. Separate fluid and structure models are created and connected via coupling elements. The coupled fluid structure model results in a complex, symmetric, banded set of equations that can be efficiently solved by Gaussian elimination for displacements and pressures throughout the model. Farfield or nearfield pressures can be computed from the normal velocities and pressures calculated on the surface of the structure.

The main computational portion of SARA3D outputs a very large file that provides surface pressures and surface velocities at the fluid-structure interface as a function of frequency. This file may then be post-processed to calculate the following quantities:

1. Three-dimensional representation of field pressures in the nearfield as a function of frequency,
2. Three-dimensional representation of field pressures in the farfield as a function of frequency,
3. Three-dimensional representation of radiated power as a function of frequency.

A portion of a SARA3D input file is shown below:

```
fsweep,,fstart,fend,finc
wet_vp,5
end,fsweep
$
ainc=10.
sx=1, xy=1, sz=0
contours,pressure,,rad,,,ainc,phi1,,sx,sy,sz
contours,pressure,,rad,,,ainc,phi2,,sx,sy,sz
```

The above example input skips the beginning of the input file which specifies the mesh generation and element and boundary condition commands.

fsweep performs the direct solution of the equations at the specified frequencies. This command calls the subroutines which calculate the element matrices and load vectors and the solver which assembles and solves the equations. **wet_vp** computes the normal velocities and pressure at the fluid structure interface and places these on an output file (called **tape23** by default). This command also computes radiated power which is placed on another output file (called **tape53** by default). The **contours,pressure** command takes **tape23** as input and creates an output file (**tape41** by default) for contour plotting of field pressures. By specifying different azimuthal angles (arguments **phi1** and **phi2** in the above example), contour plotting data for different output planes can be generated. The **contours,pressure** calculations for each plane are completely independent and thus can be performed concurrently. See section 4 for a description of how such concurrency has been achieved using nonblocking NetSolve calls. Within a plane, each calculation of field pressure (i.e., each point in both space and time) is totally independent of the other calculations. Thus, coarse-grained parallelism with no communication between concurrent tasks could be used for these computations as well, which should further improve post-processing response time.

The restart command is used to restart an analysis that has been previously partially completed. For example, the input file below would use an output file previously generated by **wet_vp** (**tape23** by default) to carry out further post-processing.

```
$restart after fsweep (requires tape23)
restart
ainc=10.
sx=1, sy=1, sz=0
contours,pressure,,,,,ainc,,,sx,sy,xz
stop
```

SARA3D runs can generate a number of output files in addition to those discussed above. Table 1 lists all the disk files used in SARA3D.

Default name	**Description**
tape5	Input data file
tape6	Output file
tape7	Log file
tape11	Binary version of tape5
tape12	All data describing the model (i.e., coordinates, element connectivity and boundary conditions), organized by elements
tape13	Element nicknames and destinations
tape15	Structural element stiffness and mass
tape17	Selected results and coordinates
tape19	Solution
tape20	Static displacement
tape21	Reactions

tape23	Normal velocities and pressures at the numerical integration points (3 per element) on the wetted surface
tape24	Substructure scale factor
tape25	Power input to axisymmetric substructure
tape29	Error summary file
tape31	Frequency table (ASCII)
tape33	Tangential and circumferential velocities at the numerical integration points on the wetted surface
tape34	Radial, axial and circumferential velocities or pressure at selected nodes
tape40a	Velocity contour plots (ASCII)
tape41	Pressure contour plots (binary)
tape41a	Pressure contour plots (ASCII)
tape43	Pressure contour plots (binary) by mode
tape43a	Pressure contour plots (ASCII) by mode
tape53	Radiated and electrical power

Table 1. SARA3D File Summary

3 The NetSolve System

The NetSolve system allows users to access computational hardware and software resources that are distributed across a network[2]. The development of NetSolve was motivated by the need for an easy-to-use, efficient mechanism for using computational resources remotely. Ease of use is obtained as a result of different client interfaces, some of which require no programming effort from the user. When given a request, NetSolve looks for a computational resource that can solve the problem, chooses the best one available, sends the request to the server (with retry for fault tolerance), and returns the answer to the user.

NetSolve utilizes standard Internet protocols and is available for all popular variants of the UNIX operating system, and parts of the system are available for the Microsoft Windows 95, 98 and NT platforms. Figure 1 shows the architecture of the NetSolve system and its relation to the applications that use it. The shaded parts of the figure represent the NetSolve system. At the top tier, the NetSolve client library is linked with the user's application. The application then makes calls to NetSolve's application programming interface (API) for specific services. The client can be a programming language, such as Fortran or C, or it can be an interactive interface such as Matlab or Mathematica, both of which are widely used by application engineers in specifying and solving engineering problems.

A NetSolve client sends a request to a NetSolve agent. The agent chooses the "best" NetSolve resource according to the size and nature of the problem to be solved. The agent maintains a database of NetSolve servers along with their capabilities (hardware performance and installed software) and dynamic usage statistics. The agent, in its resource allocation phase, attempts to find the server that will service the request most quickly, balance the load among its servers, and keep track of failed servers. The

NetSolve server is a daemon process that awaits client requests. The server can run on single workstations, clusters of workstations, symmetric multi-processors, or massively parallel computers. A key component of the NetSolve server is a source code generator which parses a NetSolve problem description file (PDF). The PDF contains information that allows the NetSolve system to create new modules and incorporate new functionalities. The software packages thus incorporated may consist of either subroutine libraries or stand-alone programs such as SARA3D. See section 4 for an explanation of how a problem description file was written to implement SARA3D as a NetSolve service.

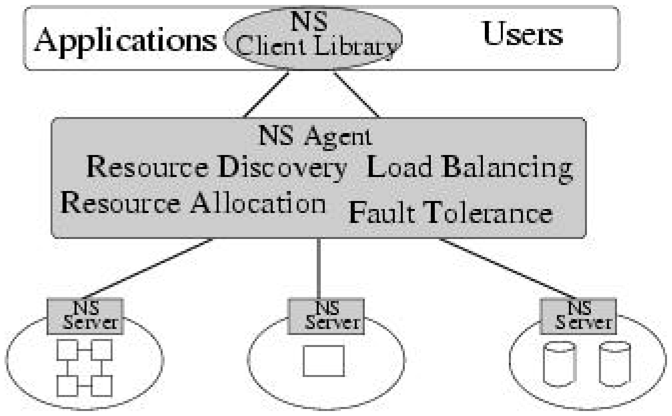

Figure 1. The NetSolve System

4 SARA3D as a NetSolve Service

4.1 Installation on the NetSolve server

Implementing SARA3D as a NetSolve service required writing a problem description file (PDF) for SARA3D. A NetSolve problem is defined as a 3-tuple:
< *name,inputs,outputs* >, where

- *name* is a character string containing the name of the problem
- *inputs* is a list of input objects
- *outputs* is a list of output objects

An object is described by an *object type* and a *data type*. In the case of SARA3D, we defined the problem to have one input which is a file object of character data type and one output which is also a file object of character data type. We then wrote the wrapper code so that the input file would be SARA3D's default **tape5**, and the output file would be SARA3D's **tape7** (see section 2 for an explanation of SARA3D input and output files). **tape7** gives a summary of the run and includes a list of all the output files generated by the run which could subsequently be retrieved from the NetSolve server. We placed the SARA3D executable on the NetSolve server machine and included the SARA3D PDF filename in the NetSolve server's configuration file to make SARA3D available on the server.

4.2 Invocation from the MATLAB client on a Windows system

To be able to invoke SARA3D from the NetSolve MATLAB client, the user must have installed the NetSolve client software on his or her machine. The NetSolve client is available for both Unix and Windows systems. The Windows version of the NetSolve client is distributed in the form of a self-extracting exe file. The user can download the desired NetSolve client from the NetSolve home page at http://www.cs.utk.edu/netsolve/ and follow the installation instructions to install and test the client software. Once the client has been installed, a MATLAB user need only carry out the following steps to be able to access NetSolve services from MATLAB:

a. Start up MATLAB
b. Click on File → Set Path …
c. Add the NETSOLVE_DIR\matlab directory to the path

Then typing

>> **netsolve('?')**

will print the NetSolve agents and servers currently available, and typing

>> **netsolve**

will print the list of problems that can be solved.

If the user would like more detailed information on a specific problem, e.g., sara3d, he can type

>> **netsolve('sara3d')**

and the output will give a short description, an example of how to invoke the service, and a description of the inputs and outputs.

To perform a computation using NetSolve, the user can send either a *blocking* or a *nonblocking* request. With a blocking request, control is returned to the user only after the computation has been successfully completed on the server. For example, if the user had a SARA3D input named **tape5,** a blocking request to the SARA3D service could be invoked as follows:

>> **[outfile] = netsolve('sara3d', 'tape5')**

A nonblocking request allows the user to regain control and check for completion of the request later, while performing other MATLAB operations in the meantime, possibly sending multiple requests to NetSolve. Multiple requests will be solved on different processors or machines if possible, thus achieving parallelism. For example, a nonblocking request to the SARA3D service could be sent as follows:

```
>> [r] = netsolve_nb('send','sara3d','tape5')
```

The left-hand side of a nonblocking request always contains a single argument. Upon completion of this call, that argument contains a NetSolve *request handler*. The request handler can then be used to probe or wait for completion of the request.

```
>> [status] = netsolve_nb('probe',r)
```

returns immediately and prints the status of a pending request. To obtain the result of a computation one must use the 'wait' action:

```
>> [outfile] = netsolve_nb('wait',r)
```

Typing the following command will return a description of all pending requests which includes an estimate of times to completion.

```
>> netsolve_nb('status')
```

5 Conclusions and Future Work

The usefulness of NetSolve for performing SARA3D post-processing computations in an efficient manner and for invoking such computations remotely from MATLAB on the user's desktop workstation has been demonstrated on testbed machines at the University of Tennessee and BBN Technologies. Making the NetSolve SARA3D service available to ERDC users will require integrating NetSolve with the ERDC environment. The two main problems that need to be solved are 1) authentication of NetSolve requests and 2) interfacing with the batch queueing systems.

A version of NetSolve is available that includes Kerberos support[3, 4]. Kerberized NetSolve clients can interoperate with both Kerberized and non-Kerberized NetSolve servers. In either case the client sends a request to the server. A non-Kerberized server will return a status code indicating it will accept the requested operation. A Kerberized server will return an "authentication required" response. The client is then required to send Kerberos credentials to the server before the request will be processed. Provided the user has run **kinit** and the ticket-granting ticket has not expired, the NetSolve client automatically contacts the Kerberos Key Distribution Center for a ticket and sends it to the server. The server implements access control via a simple list of Kerberos 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 userid of the particular UNIX user who submitted the request. The current version of Kerberized NetSolve performs no encryption of the data exchanged among NetSolve clients, servers, or agents. Feedback is needed on whether this currently provided Kerberos authentication capability for NetSolve meets the MSRC security requirements.

In order to satisfy the BBN Technologies goal of using NetSolve to provide access for their users to the SARA3D service on large MSRC machines, NetSolve will need to work within the batch queueing environment of these machines. Because NetSolve is designed to be an interactive system, some mechanism will need to be found to support such interactive use in the batch queueing environment. It may be sufficient to service quick requests on a NetSolve server running on a login node, and have that server submit longer-running and/or parallel computations to the queueing system.

NetSolve has a mechanism called *task farming* which is a way of managing large numbers of requests for a single NetSolve problem[3, 5]. In the present distribution, the netsl_farm() call is only available from C, but it will soon be made available from MATLAB. This call is appropriate when many somewhat similar computations must be performed in parallel. The call manages the requests for the user so that only one call need be made to submit all the requests and another call to retrieve all the results. When this call becomes available from MATLAB, we plan to use it to obtain parallelism for SARA3D post-processing in a more convenient and efficient manner than the current method of using nonblocking NetSolve calls.

Other future work includes using the NetSolve adaptive solver interface to incorporate new solvers into the current SARA3D code[6]. In addition, we plan to provide a Web interface to the SARA3D service similar to the Web interface already provided for the IPARS subsurface modeling code[7]. Finally, in order to allow SARA3D users to archive and share SARA3D data files, we plan to implement a repository of SARA3D data sets using the Repository in a Box (RIB) toolkit which is already in use at the MSRCs for cataloging software[8].

References

1. Allik, H., R. Dees, S. Moore, and D. Pan, *SARA-3D User's Manual*. 1995, BBN Acoustic Technologies.
2. Casanova, H. and J.J. Dongarra, *NetSolve: A Network Server for Solving Computational Science Problems.* International Journal of High Performance Computing Applications, 1997. **11**(3): p. 212-223.
3. Arnold, D.C., S. Blackford, and J.J. Dongarra, *Users' Guide to NetSolve V1.3*. 2000, Department of Computer Science, University of Tennessee, http://www.cs.utk.edu/netsolve/.
4. Arnold, D.C., S. Browne, J. Dongarra, G. Fagg, and K. Moore, *Secure Remote Access to Numerical Software and Computational Hardware*. in *DoD HPC Users Group Conference (HPCUG2000)*. June, 2000. Albuquerque, New Mexico.
5. Casanova, H., M. Kim, J. Plank, and J. Dongarra, *Adaptive Scheduling for Task Farming with Grid Middleware.* International Journal of Supercomputer Applications, 1999, 13(3): p. 231-240..

6. Arnold, D.C., S. Blackford, J. Dongarra, V. Eijkhout, and T. Xu, *Seamless Access to Adaptive Solver Algorithms*. in *16th IMACS World Conference on Scientific Computation, Applied Mathematics and Simulation*. 2000. Laussanne, Switzerland.
7. Arnold, D.C., W. Lee, J. Dongarra, and M. Wheeler, *Providing Infrastructure and Interface to High-Performance Applications in a Distributed Setting*. in *Proc. High Performance Computing*. 2000, pp. 248-253, Society for Computer Simulation International, April 2000.
8. Browne, S., P. McMahan, and S. Wells, *The Repository in a Box Toolkit for Software and Resource Sharing*. 1999, University of Tennessee Computer Science Department Technical Report UT-CS-99-424.