PROVIDING INFRASTRUCTURE AND INTERFACE TO HIGH-PERFORMANCE APPLICATIONS IN A DISTRIBUTED SETTING

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

Wonsuck Lee and Mary F. Wheeler
Center for Sub-Surface Modeling
University of Texas
Austin, TX 78712
[wslee, mfw]@ticam.utexas.edu

Keywords: Distributed Computing, Problem Solving Environments, Sub-Surface Modeling, Heterogeneous Network Computing.

ABSTRACT

The NetSolve project was established to aid scientists who prefer not to be concerned with the usual tedium associated with finding and maintaining software libraries which they use to create programs, toolkits and problem solving environments particular to their scientific domain. This article introduces the reader to the NetSolve system and discusses how it can be leveraged to build robust infrastructure for simulation frameworks, toolkits and other programs. The IPARS simulator is used as a concrete example of this approach. We further show how the ubiquity of the web and web browsers can be exploited to make simulators generally available without the need for downloading software.

1 INTRODUCTION

Two things remain consistent in the realm of computer science: a need for more computational power than we have at any given point, and a desire for the simplest, yet most complete interface to our resources. Recently, much attention has been given to the area of Grid Computing (Foster and Kesselman 1998), but while we are making great advances in our ability to harness the cumulative functionalities of disparate resources, we have done little to minimize the effort and know-how that one needs to properly and productively utilize this collection of computational resources. The problem is that the systems that do for us what we need tend to require large investments of time (including installation and programming), knowledge of distributed computing, and a total commitment to the system involved. On the other hand, the systems that present easy-to-use interfaces are often lacking in functionality.

In this article, we briefly describe our approach to Grid Computing, NetSolve. NetSolve allows for the easy access to computational resources distributed in both geography and ownership. We also describe a parallel simulator with support for visualization that runs on workstation clusters and show how we have used NetSolve to provide an interface that allows one to use the simulator without obtaining the simulator software or the tools needed for visualization. This methodology can be easily extended to make arbitrary simulation tools or programming software widely available, easily accessible, and, perhaps most importantly, executed remotely. Sections 2 and 3 provide overviews of the NetSolve and IPARS systems respectively. We describe how IPARS was integrated into the NetSolve system, and the benefits of such an integration in Section 4. Section 5 describes related work, and finally, Section 6 summarizes the work and its implications.

2 NETWORK-ENABLED SOLVERS

The NetSolve project is being developed at the University of Tennessee and the Oak Ridge National Laboratory. Its original motivation was to alleviate the difficulties that domain scientists usually encounter when trying to locate/install/use numerical software, especially on multiple platforms. NetSolve provides remote access to computational resources, both hardware and software. Built upon standard Internet protocols, like TCP/IP sockets,

[*]Mathematical Science Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831

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 and NT platforms.

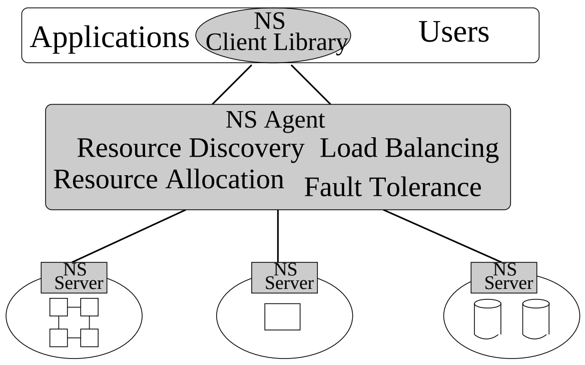

Figure 1: Architectural Overview of the NetSolve System

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.

```
...                     | ...
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

Figure 2 helps to show what the programming code would look like before and after the NetSolve API has been integrated. 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 provides 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 notebook computers. Furthermore, as explained below, NetSolve adds heuristics that attempt to find the most expeditious route to solve any given problem. NetSolve currently supports the C, FORTRAN, Matlab, and Mathematica programming interfaces as languages of implementation for client programs.

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 likely 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.

For more detailed information on the NetSolve sytem and its usage, refer to (Casanova and Dongarra 1998) and (Casanova et al. 1996).

2.1 The Status Of NetSolve

The next official release of NetSolve is planned for March of 2000. Features to be 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. Currently, NetSolve-1.2, 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.

3 A FRAMEWORK FOR RESERVOIR AND ENVIRONMENTAL SIMULATION

It is pointed out by J. Wheeler (Wheeler 1998) that the need for a simulation framework that supports reservoir and fluid-flow dynamics research arises from shear size of realistic simulators; on the order of 20,000 lines of code may be required to support a physical model. It is difficult and inefficient for individual researchers to develop such a framework before even beginning to test their ideas. The Integrated Parallel Accurate Reservoir Simulator (IPARS) is designed to lay the ground work that makes it easier for such researchers to carry out their work. IPARS is a simulation framework to study fluid flow through porous media or, more practically, through underground structure. It houses petroleum engineering applications and environmental geological models. Constant work is being done to increase the number of physical models.

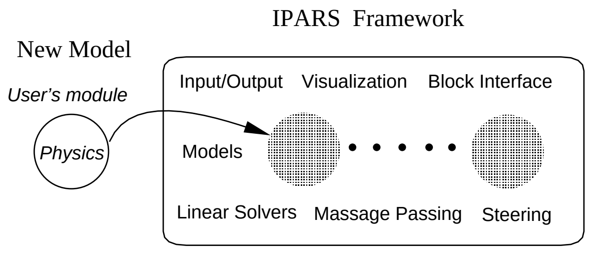

Figure 3: IPARS framework which houses several physical models and framework support modules.

3.1 Framework Characteristics

IPARS is designed to provide many common parts of a simulation framework including an input/output interface, visualization, and message passing interfaces etc., see Figure 3. The ability to model complex physical processes such as geochemistry and coupled geomechanics cannot readily be added to existing simulators. The same is true of many computational enhancements, such as unstructured grids and interactive simulation. IPARS is structured to support multiple physical and mathematical models. Thus, IPARS makes individual research at universities and industries more efficient.

In addition to the advantages of framework support described above, IPARS has several distinguishing aspects in simulation capabilities. Most reservoir simulations today sacrifice grid resolution to reduce cost; fewer than 100,000 grid elements may be used even though one grid block covers a few kilometers in field. It is intended that IPARS be able to economically solve problems involving a million or more grid elements and thereby greatly improve grid resolution.

Efficient, realistic well management is an unsolved problem in reservoir/aquifer simulation; for some large reservoirs, over 50% of both CPU and manpower costs are directly attributable to well management. In addition, well management is primarily a sequential calculation; efficient implementation on parallel computers will be difficult (perhaps impossible.) The IPARS simulator provides a platform for attacking this problem too.

The simulator framework supports three dimensional transient flow of multiple phases containing multiple components through immobile phases (rock/soil). The bulk phase of medium (*ie.* rock plus fluid) can be regarded compressible to include elastic property of bulk rock. The thermodynamic quantities, for example, phase densities, compressibility factor, viscosities may be arbitrary functions of pressure and composition or may be represented by simpler functions (*e.g.* constant compressibility). The initial system is isothermal but an effort is being made to include incorporation of non-isothermal calculations.

The most general mathematical representation of such a system without mutual solubility between hydrocarbon and water phases is

$$\frac{\partial\left(\rho_i S_i \phi_i\right)}{\partial t} - \nabla \cdot \sum_{j}^{N_p} \frac{K k_{rj} \xi_j}{\mu_j} x_{ij}\left(P_j - \gamma \Delta D\right) = q_j \ .$$

for N_c hydrocarbon phases and N_p consisting phases. Here the first term with time derivative represents change of i-th phase mass in time, the term containing the inner product with a gradient operator is change due to transport of phase. The right hand side is a source/sink term. A detailed explanation, which is out of scope of this paper, of this equation can be found in any advanced book in petroleum reservoir engineering (Peaceman 1977).

The reservoir consists of one or more fault blocks. Each fault block has an independent user-defined coordinate system and gravity vector. Flow between fault blocks, however, can occur only through a common flat face. The primary grid imposed on each fault block is a logical cube but may be geometrically irregular. Currently, the framework supports both rectangular grids and corner-point grids. Dynamic grid refinement of the primary grid on each fault block is supported by the framework but also must be supported by the individual physical models. Grid elements may be keyed out to efficiently represent irregular shapes and impermeable strata.

The simulator is formulated for parallel distributed memory machines. For the message passing interface, the MPI standard is used, however, the system is designed so that any reasonable message passing system can be substituted. On multiprocessor machines, the grid system is partitioned among the processors such that each processor is assigned a subset of the total grid system. Dynamic domain decomposition is used to distribute grid elements among the processors. Each CPU separately processes the data input file, but the control processor (a single processor) collects the data to prepare the output. Dynamic load balancing is provided. Here we have described the computationally intensive aspects of IPARS focusing on high performance scientific computation, see the IPARS manual (Wheeler 1998) for full functionality of IPARS.

3.2 IPARS User Interface

Free-form keyword input is used for direct data input to the computation stage of the simulator. The ASCII keyword input file(s) is explicitly defined to serve as an output file from a graphical front end or geostatistical grid and property generator. Multiple levels of output are provided in the simulator. These will range from selective memory dumps for debugging to minimal output for automatic history matching. Visualization is controlled by the input file and produces an ASCII output file which is readable by TECPLOT (AMTEC ENGINEERING n.d.), the commercial software chosen for visualization. TECPLOT has useful functionalities for three dimensional data sets and fluid properties.

4 INTEGRATING IPARS INTO NETSOLVE

The interface into and out of the IPARS sub-system as explained in Sect. 3.2 is straightforward. The task at hand was to make the IPARS system known to NetSolve via this interface yielding the availability of a parallel installation of IPARS, and the hardware to run it on, from a single function call.

4.1 The IPARS-Enabled NetSolve Server

After installing and testing a version of the IPARS code that runs on a cluster of dual-node Linux work stations, we moved to incorporate that system into NetSolve. We wrote a functional wrapper to the IPARS system that takes as parameters an input and several output filenames. This wrapper runs the simulation and also calls the scripts which uses TECPLOT to post-process the output into a series of graphical frames. These frames represent snapshots of different parameters being observed in the field of study. The UNIX utility, convert, is then used to convert each set of frames (corresponding to different parameters) into a single movie file for each set. These movie files, along with the ascii output file are placed in the files specified by the output filename parameters.

As described in Sect. 2, the NetSolve system provides a code generator that parses a NetSolve PDF in order to extend the servers' functional capabilities. Inevitably, this was the tool used to create a NetSolve server with IPARS capability.

```
@PROBLEM ipars
@INCLUDE "ipars.h"
@LIB /home/user/lib/libipars.a
@DESCRIPTION
Parallel Sub-Surface Flow Simulator
@INPUT 2
@OBJECT STRING CHAR model
IPARS physical model to use
@OBJECT FILE CHAR infile
Input data file
....
```

Figure 4: Portion of PDF File Used for Integration

Figure 4 shows a segment of the PDF that was used. The **PROBLEM** parameter of this file defines the name we want client applications to use when referring to this module. The **INCLUDE** and **LIB** directives are used in the compilation of the module. This module will take two inputs, a string of characters and a character file, as described by the last five lines shown. Among other things, this PDF will eventually describe the code that determines how to call the abovementioned wrapper with the inputs given from a client program. After this configuration and a compilation, the NetSolve server is ready to be attached to a NetSolve agent/system and service requests. Note that although we only talk about one server-cluster, it is also possible to have several IPARS-enabled server clusters or parallel machines attached to the system; the NetSolve agent would dynamically marshal requests to the best performing candidate.

```
...
/* code to initialize input file, model, etc.  */
...
status = netsolve("ipars", model, infile, outfile, ... );
...
/* code to view output files */
```

Figure 5: Sample C code used to request IPARS module using NetSolve

4.2 Using The NetSolve Client Interface

At this point, one can now use any of the NetSolve client interfaces to access the IPARS simulator. Figure 5 shows an example of how this might be done from the C language API. Recall Section 3.2 where it was stated that the IPARS input file was designed so that it could easily be created by a graphical front end or geostatistical grid and property generator. This sample code shows how easily NetSolve can be used from such a program. The call to **netsolve()** can be made after the point where the entire input file has been processed. Integrating IPARS and NetSolve in this way has three major impacts:

1. With a single installation of IPARS many users can use the simulator without having to go through the hassles of installation and maintenance.
2. From any NetSolve host machine (even architectures to which IPARS has not been ported) IPARS can be used. Currently, IPARS can be run only on the LINUX platform.
3. Individual users can achieve significant speedup by accessing server clusters that are orders of magnitude faster than their local resource.

The use of the NetSolve client means that the users have the added benefit of access to all the functional components of NetSolve, not just the IPARS system. This includes on the order of one hundred numerical solver routines to solve things like systems of equations, eigenvectors and eigenvalues, matrix multiplies, etc.

4.3 The Finishing Touch

To further facilitate the user, we have taken our interface a step further. We have made the IPARS simulator accessible to the standard web browser. Using HTML forms and the Common Gateway Interface(CGI), we have created a complete Web interface to IPARS that basically

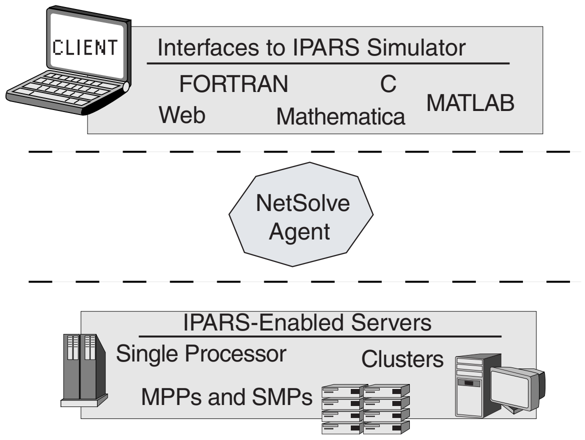

Figure 6: Overview of the NetSolve/IPARS Integration

sits on top of the NetSolve middleware system. The total package has all the components that every application should have: complete portability, an easy and intuitive interface, and run-time load-balancing to ensure maximum performance. All this is achieved without ever downloading or installing any special components.

5 RELATED WORK

The WebFlow (Haupt et al. 1999*b*) system was developed at the Northeast Parallel Architectural Center at Syracuse University. It has a three tier design similar to that of NetSolve, where the front-end is a web applet that utilizes visual icons and a drag-and-drop technique to formulate computational graphs that represent execution modes and data flow. A distributed object based, scalable and reusable Web server and Object broker forms the middle tier. Back-end services that provide access to high performance computational resources form the final tier. The Gateway (Haupt et al. 1999*a*) project is the successor to WebFlow that acts on some of the same premises, but includes enhancements like fault tolerance and security.

WebSubmit (McCormack et al. n.d.) and UNICORE (Almond and Snelling 1998) are very similar projects that use the World Wide Web as interfaces to create a simple, yet seamless, mode of access to high performance computational resources. The main thrust of their effort is to provide uniform access to multiple variations of hardware resources and queuing systems, so as to relieve users of the daunting task of learning the peculiarities of each system. Their differences lie in their implementation and scope. WebSubmit is implemented using CGI and TCL, while UNICORE uses Java. Also, WebSubmit is meant for independent tasks at a single site, whereas UNICORE is extended for interdependent tasks at multiple, geographically distributed sites.

WebOS (Vahdat et al. 1998), primarily developed at the University of California at Berkeley, is an effort to provide a common set of operating system services to wide-area applications. These services include mechanisms for naming, remote execution, persistent storage, resource management, authentication and security. WebOS helps to make distributed applications highly available, incrementally scalable, and dynamically reconfigurable.

Coincidentally, other projects have taken advantage of NetSolve and use it in a similar fashion as IPARS. MCell (Bartol and Stiles n.d.) is a general Monte Carlo simulator of cellular microphysiology. MCell uses Monte Carlo diffusion and chemical reaction algorithms in 3D to simulate the complex biochemical interactions of molecules inside and outside of living cells. Collaborative Environment for Nuclear Technology Software (CENTS) is a project of the Oak Ridge National Laboratory that aims to lay the foundation for a Web-based distance computing facility for executing nuclear engineering codes. CENTS allows users to focus on the problem to be solved instead of the specifics of a particular nuclear code. In both cases, users submit their problems via Web browsers, and both graphical and text based output is brought back to the user, through the use of NetSolve.

6 CONCLUSION

In this article, we described our use of the NetSolve system to facilitate distributing high-performance applications. IPARS is a parallel simulator that we make widely and easily accessible to those that wish to use it. Our design is by no means specific and can be used to provide the same convenience to other users wishing to use other software systems or library toolkits. Our belief is that, using our approach, very little effort is required for users to tap into all the resources they need to achieve the computation they want.

Acknowledgements: This project was supported in part by the National Science Foundation (contract #ACI-9876895,) the National Partnership for Advanced Computational Infrastructure (P.O. #10152763 under Prime Contract #ACI-96-19020) and the Center for Research on Parallel Computation (Subcontract #292-3-54397). We also wish to thank Michelle Miller and Terry Moore, both of the University of Tennessee, for their helpful editing comments.

References

Almond, J. and Snelling, D. (1998), UNICORE: Secure and Uniform Access to Distributed Resources via the World Wide Web. A white paper from http://www.kfa-juelich.de/zam/RD/coop/unicore/.

AMTEC ENGINEERING, I. (n.d.), 'Visualize Implement Achieve Performance'. Project Web Site: http://www.amtec.com.

Bartol, T. and Stiles, J. R. (n.d.), 'A General Monte Carlo Simulator of Cellular Microphysiology'. Project Web Site: http://www.mcell.cnl.salk.edu/.

Casanova, H. and Dongarra, J. (1998), 'NetSolve's Network Enabled Server: Examples and Applications', *IEEE Computational Science & Engineering* **5**(3), 57–67.

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

Foster, I. and Kesselman, C., eds (1998), *The Grid, Blueprint for a New computing Infrastructure*, Morgan Kaufmann Publishers, Inc.

Haupt, T., Akarsu, E. and Fox, G. (1999*a*), The Gateway System: Uniform Web Based Access to Remote Resources, *in* 'ACM 1999 Java Grande Conference'.

Haupt, T., Akarsu, E. and Fox, G. (1999*b*), WebFlow: a Framework for Web Based Metacomputing, *in* 'High Performance Computing and Networking '99'.

McCormack, R., Koontz, J. and Devaney, J. (n.d.), Seamless Computing with WebSubmit. Concurrency: Practice and Experience (in press). Found at http://www.itl.nist.gov/div895/sasg/cpeRev.ps.

Peaceman, D. W. (1977), 'Fundamentals of numerical reservoir simulation', *Elsevier Scientific Publishing Company* .

Vahdat, A., Anderson, T., Dahlin, M., Culler, D., Belani, E., Eastham, P. and Yoshikawa, C. (1998), WebOS: Operating System Services for Wide Area Applications, *in* 'The Seventh IEEE Symposium on High Performance Distributed Computing'.

Wheeler, J. (1998), IPARS Simulator Framework User's Guide, Technical report, University of Texas at Austin. TICAM Technical Reports.