Providing Infrastructure and Interface to High-Performance Applications in a Distributed Setting. (Extended Abstract)

Dorian C. Arnold [*] Wonsuck Lee [†] Jack Dongarra [*‡] Mary Wheeler [†]

December 28, 1999

[*]Department of Computer Science, University of Tennessee, Knoxville, TN 37996
[†]Texas Institute for Computational and Applied Mathematics, University of Texas, Austin, TX 78712
[‡]Mathematical Science Section, Oak Ridge National Laboratory, Oak Ridge, TN 37831

1 Introduction

Two things remain consistent in the realm of computer science: i)there is always a need for more computational power than we have at any given point, and ii)we always want the simplest, yet most complete and easy to use interface to our resources. In recent years, much attention has been given to the area of Grid Computing. The analogy is to that of the electrical power grid. The ultimate goal is that one day we are able to plug any and all of our resources into this Computational Grid to access other resources without need for worry, as we do our appliances into electrical sockets today.

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.

2 Network-enabled Solvers

The NetSolve project, under development at the University of Tennessee and the Oak Ridge National Laboratory, has been a successful venture to actualize the concept of Computational Grids. 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 is of a client/agent/server design in which the client issues requests to agents who allocate servers to service those requests; the server(s) then receives inputs for the problem, does the computation and returns the output parameters to the client. The NetSolve client-user gains access to limitless software resources without the tedium of installation and maintenance. Furthermore, NetSolve facilitates remote access to computer hardware, possibly high-performance supercomputers with complete opacity. That is to say that the user does not have to possess knowledge of computer networking and the like to use NetSolve. In fact, he/she does not even have to know remote resources are involved. The NetSolve system is further enhanced by features like fault-tolerance and load balancing. At this point, we offer a brief discussion of the three aforementioned components.

The NetSolve agent represents the gateway to the NetSolve system. It maintains a database of servers along with their capabilities (hardware performance and allocated software) and usage statistics. It uses this information to allocate server resources for client requests. The agent, in its resource allocation mechanism, balances load amongst its servers; it is also the primary component that is concerned with fault-tolerance.

The NetSolve server is the computational backbone of the system. It is a daemon process that awaits client requests. The server can be run on all popular strains of the UNIX operating system and has been ported to run on almost any architecture. The server can run on single workstations, clusters of workstations, or shared-memory multiprocessors. It provides the client with access to software resources and also provides mechanisms that allow one to integrate any software with NetSolve servers.

The NetSolve client user submits requests (possibly simultaneously) and retrieves results to/from the system via the API provided for the language of implementation. NetSolve currently supports the C, FORTRAN, Matlab, and Mathematica programming interfaces. The functional interface completely hides all networking activity from the user.

2.1 The Status of NetSolve

NetSolve-1.2 was released in November of 1998. A new release is pending for 1999. Features to be implemented in this release include a Java API, a Microsoft Excel interface, enhanced load balancing, dynamic downloading of server software, and, of course, request sequencing. 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 one of the most significant efforts in research and development today, and was recently named to R&D Magazine's top 100 list for 1999.

3 A Framework for Reservoir Simulation

The Implicit Parallel Accurate Reservoir Simulator (IPARS) is a reservoir simulator framework suitable for research and eventual conversion to a commercial simulator. The need for a simulator framework that will support 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. Thus, IPARS makes individual research at universities and commercial labs more efficient. Most reservoir simulations today sacrifice grid resolution to reduce cost; fewer than 100,000 grid elements may be used even though this requires that some grid elements contain more than one wellbore. It is intended that IPARS be able to economically solve problems involving a million or more grid elements and thereby greatly improve grid resolution.

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.

Efficient, realistic well management is an unsolved problem in reservoir 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.

3.1 Framework Characteristics

The simulator framework supports three dimensional transient flow of multiple phases containing multiple components plus immobile phases (rock) and adsorbed components. Phase densities and 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 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.

On multiprocessor machines, the grid system is distributed among the processors such that each processor is assigned a subset of the total grid system. The simulator is formulated for a distributed memory, message passing machine. The MPI standard is used, but the system was designed to facilitate the incorporation of any reasonable message passing system. Dynamic domain decomposition is used to distribute grid elements among the processors. Well calculations are also be dynamically distributed. Each CPU separately processes the data input file. A single processor (or host machine) collects and organizes output data. Dynamic load balancing is provided for in the initial code but may not be optimal. Execution on a single processor machine is, of course, possible but may not be as efficient as a simulator written solely for single processor machines.

The IPARS framework supports an arbitrary number of wells each with one or more completion intervals. A well may penetrate more than one fault block but a completion interval must occur in a single fault block. Well grid elements may be assigned to more than one processor. The framework assigns primary responsibility for a well calculation to one of these processors and communicates well data amongst processors. For each well element, the framework also provides estimates of the permeability normal to the wellbore, the geometric constant in the productivity index, and the length of the open interval. Other well calculations are left to the individual physical models.

Either a multi grid or domain decomposition linear solver can be used in the simulator. Recent work indicates domain decomposition is the better choice.

3.2 Input\Output

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 for 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, the commercial software chosen for visualization. It has useful functionalities for 3 dimensional data sets and fluid properties.

4 Interfacing IPARS with NetSolve

Our motivation is to show how advanced systems, like IPARS, could easily be interfaced and made accessible to communities at large. In this section, we layout the how we did this and show how other applications can benefit from this same technology.

4.1 The IPARS-enabled NetSolve Server

The interface to the IPARS sub-system as explained in Sect. 3.2 is straightforward: a single file input containing all the parameters and field data describing simulation and several output files. The first output file describes the results of the simulation in an ASCII format. The other output files contain data that is used to support visualization. Several scripts that are external to the simulator are used to post-process these visualization files and create graphical results.

As discussed in Sect. 2, the NetSolve system provides the tools that allow the servers functional capabilities to be extended. We used these tools to create a server with IPARS capability. This server will use MPI to run the simulator in parallel on a cluster of workstations, in our case Linux boxes. We also wrote a functional wrapper to the IPARS system that takes an input filename and an output filename. The input is the same as IPARS would take regularly. The wrapper runs the simulation and also calls the scripts which post-process the output into a series of graphical frames. The convert utility is then used to convert these frames into a single movie file, which becomes the output file. The NetSolve server is configured to take a file as input and return a file as output. These are the same files passed to/from the IPARS functional wrapper. After this configuration, the NetSolve server is ready to be attached to a NetSolve system and service requests. Note that it is also possible to have several IPARS-enabled server clusters attached to the system, and the NetSolve agent would dynamically marshall requests to the best candidate yielding better performance.

4.2 Using the NetSolve Client Interface

At this point, one can now use any of the NetSolve client interfaces to access IPARS. This has two major impacts: i)with a single installation of IPARS many users can benefit from the simulator without having to go through the hassles of installation and maintenance, and ii)from any host machine (even architectures to which IPARS has not been ported) IPARS can be used. A further result is that you can get significant speedup by accessing server clusters that are orders of magnitude faster than your local computer. The fact that we are using the NetSolve client means that the user has access to all the functionalities of any and every NetSolve server in the system – an added benefit.

4.3 The Glossy Finish

In order to further facilitate the user, we take our interface a step further. We make the IPARS simulator accessible to the ever-present web browser. Using HTML forms and the Common Gateway Interface(CGI), we create a complete interface to IPARS that basically sits on top of the NetSolve middleware system.

The total package has all the components that every application should have: complete portability, easy and intuitive interface, and run-time load-balancing to ensure maximum performance, and all without ever downloading or installing any component (other than the web browser which can be assumed to be standard.)

5 Conclusion

In this article, we described our use of the NetSolve system to facilitate distributed computing over networks than can be widely distributed. 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 with very little effort users should be able to tap into all the resources they need to achieve the computation they want.