Report

On

Hardware Software Server in NetSolve

by

Sudesh Agrawal

Professor: Dr. Jack Dongarra

Computer Science Department

University of Tennessee, Knoxville

Date: March 11, 2002

Table of Contents

Abstract

NetSolve is a project that makes use of distributed computational resources connected by computer networks to efficiently solve complex scientific problems. It is a remote procedure call (RPC) based client/agent/server system that allows users to discover, access, and utilize remote software modules and the hardware needed to run these modules. NetSolve facilitates heterogeneous computing, or the ability to combine different machine architectures and/or operating systems to solve a problem.

Over the period of time NetSolve has gained great popularity among various universities, research groups and individuals. It has always been the policy of NetSolve project team to make NetSolve as easy as possible to be used even by a naïve user. This pilot project is a step forward towards accomplishing this goal. NetSolve has been very successful in providing the user with an easy to use interface to access the software, however it is still an arduous task for a user to integrate his/her software into the NetSolve.

This project involved categorizing servers in NetSolve into two domains viz. Software Servers and Hardware Servers, this categorization serves many purposes as discussed in this report, however the main advantage is easy integration of software into the NetSolve system, which is discussed in detail in the report.

Acknowledgments

I would like to take this opportunity to thank Dr. Jack Dongarra, Director - ICL for giving me an opportunity to work on NetSolve as my PILOT project.

My sincere thanks go to Dr. Jim Plank and Dr. Mike Berry for being on my PILOT project committee. I sincerely appreciate their interest in this project and their valuable time in going through this report and also for time during the defense of my pilot project.

I would also like to thank Michelle Miller for her valuable suggestions on and off during the project. Her suggestions during the deadlocks helped giving the project a renewed momentum.

I am indebted to Dorian Arnold for the kind help provided by him in setting the initial stage for the project. His opinions and suggestions were very useful when the project was initially conceived.

My sincere appreciation goes to Terry Moore, for his interest in my pilot project and the interesting brain storming sessions with him in the ICL lounge.

I would also like to thank to NetSolve team, particularly Sathish Vadhiyar, Kiran Sagi, Zhiao Shi, Keith Seymour, DongWoo Lee, Chris Hurt, Keith Moore, Victor Eijkhout and others for their patience during the NetSolve meetings listening to me.

Last but not the least, I would like to thank Shirley Moore, Brett Ellis, icl-help members, and all other ICL staff for their constant help during the course of the project.

NetSolve

Definition

NetSolve is a client-server system that enables users to solve complex scientific problems remotely. The system allows users to access both hardware and software computational resources distributed across the network. NetSolve searches for computational resources on the network, chooses the best one available, and solves a problem using retry for fault-tolerance, and returns the answers to the user.

Components

?? **Agent**: The agent represents the gateway to the NetSolve system. It maintains a database of NetSolve servers along with their capabilities and dynamic usage statistics for use in scheduling decisions. The NetSolve agent attempts to find the server that will service the request, 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 heuristics that attempt to use every likely server until it finds one that successfully services the request.

?? **Server**: 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 multiprocessors (SMPs), or massively parallel processors (MPPs). One key component of the server is the ability to wrap software library routines into NetSolve software services by using an Interface Definition Language (IDL) facility called the NetSolve Problem Description File (PDF).

?? **Client**: A NetSolve client user accesses the system through the use of simple and intuitive application programming interfaces (APIs). The NetSolve client uses this APIs to make a request to the NetSolve system, with the specific details required with the request. This call automatically contacts the NetSolve system through agent, which in turn returns the server, which can service the request. The client then contacts the server to start running the job with the input data.

NetSolve Architecture

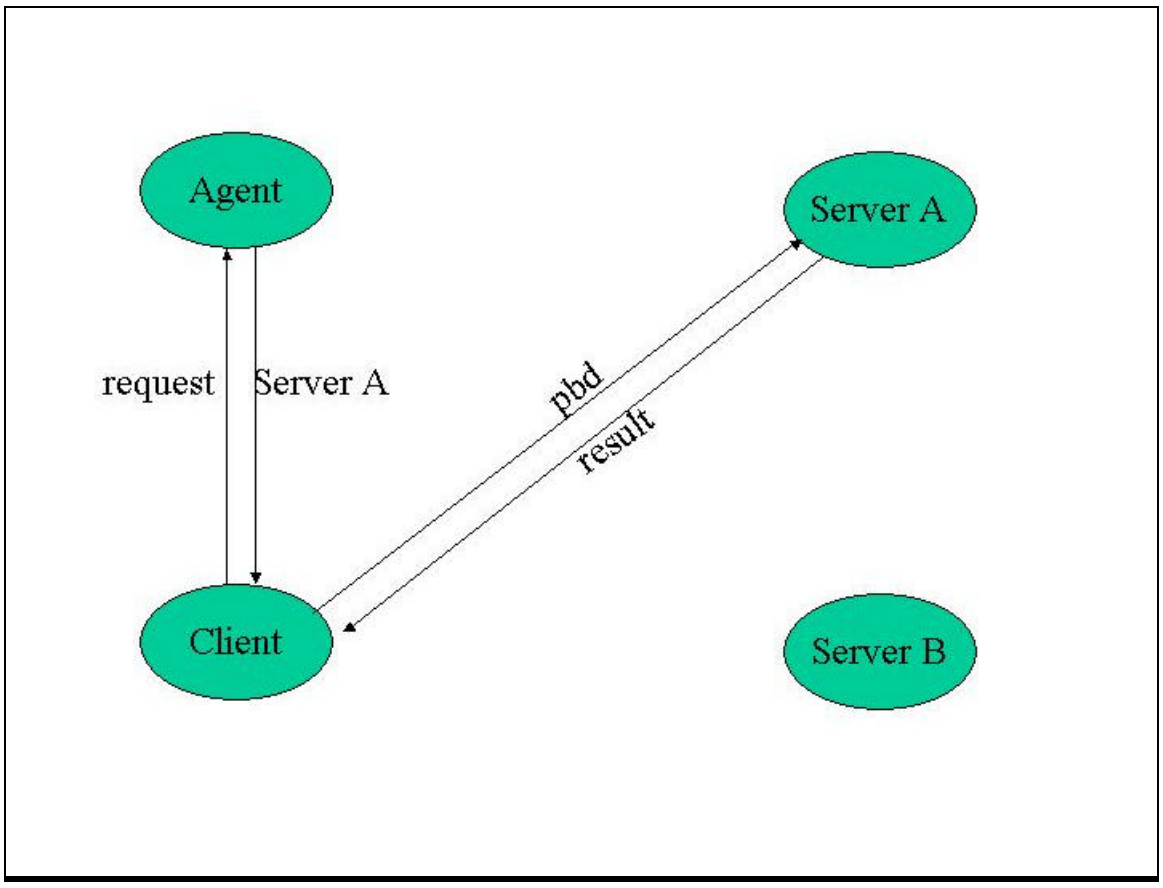

Fig. 1

Explanation

Client initiates a request to the agent with specifics of the problem it wants to solve. Agent, which has the database of servers, upon request, sends a server, which is capable of running the job. The client after receiving the server contacts the server and submits the job. The server runs the job and returns the result back to the client.

Hardware-Software Server

What is it all about?

In the current development phase of NetSolve, NetSolve server provides two functionalities, firstly as database or repository of software and secondly as hardware to run and execute the software. Any server in the NetSolve has both the functionalities.

This project involved separating these functionalities and categorizing servers into software and hardware server. Depending on the intent of the machine owner, the owner can either setup a server to store the software or to provide the CPU cycles to execute the software on his machine.

Motivation

In a broad sense, NetSolve users can be mainly categorized in two categories, Software users and Software providers. NetSolve team's main focus has been to provide an easy to use interface to both categories of users.

Though NetSolve in itself a wonderful concept, its usage has been hampered by the arduous effort involved in setting up and using NetSolve. As mentioned in the abstract, NetSolve has been very successful in providing easy to use interface for the users who just want to access the libraries already available within the system. However, NetSolve has not been able to attract a sizeable community to integrate their applications in the NetSolve.

One of the reasons that make it hard for a user to integrate their software in NetSolve is that the user has to manually install software on all the machines and compile and start their own servers. This sometimes seems an intimidating task for a naïve user.

If NetSolve could attract more software providers, which in turn would enrich NetSolve system with more capabilities, NetSolve would be able to attract more software users. This would then serve dual purpose of attracting both kinds of user.

Hardware - Software categorization is one step towards attracting users to provide their software to NetSolve. There are various benefits that can be accomplished through this categorization as mentioned later.

Functionalities

The current NetSolve has been modified in a way that servers can be registered as Hardware/Software server, Hardware server, and Software server. The user would be able to specify a command line argument to start up server as desired. However this functionality has not been implemented yet.

When a user registers a server as hardware server, the machine becomes available to any NetSolve user for execution of his problem. The agent then adds this server to the database in the hardware server list. It also knows the architecture of the hardware server. Thus if the client submits a request, the agent can make a better decision whether to use this hardware server or not. If other servers are overloaded, it can select this hardware server as potential server.

When a user registers a server as software server, this server acts as a software repository. The agent knows what all software or problems this server contains. The agent also knows about the architecture of this server. When client makes a request, agent can create a mapping of software server containing the particular software with all the hardware servers with the same architecture.

Thus, the return value from the agent contains a mapping of hardware and software server. The client can then contact the hardware server with the set of input data. The transfer of the software is done automatically. The hardware server contacts the software server and requests the software server to send the software.

Software caching

The software is dynamically transferred from the software server to the hardware server. This however is done only once, when the software arrives at the hardware site, hardware server can cache the software. The hardware server administrator decides the caching policy. Currently, the hardware server can cache maximum of 10 problems, and after that it removes the problem on LRU basis.

Implementation

As far as the implementation goes, there has been a major change in NetSolve code. Basically, the agent scheduling algorithm, server communication with agent and client, client communication with agent and server has changed. This means all the essential parts in the code have been touched.

The protocol has been changed and new protocol variables have been added to accomplish the new communication protocol, see Appendix A for the added protocols. The affected files have been listed in the Appendix B. The changes can be noted in the figure 1 and 2. Fig. 1 represents the original architecture of NetSolve. Fig. 2 represents the architecture of Hardware-Software server implementation in NetSolve

LRU policy has been implemented to cache and decache the problems/software with the hardware server.

Benefits

?? Increased ease in integration of software into the NetSolve system. The software provider can start up just one server and the software could be distributed across the servers in NetSolve system. As a result the software could be available to different servers. The software provider can as well shut down the original server, leaving the software to float around in the NetSolve system. This however posses a security problem, which is currently not addressed in the project

?? Increased clarity of the NetSolve architecture

?? Depending on the load at a particular server, agent can make a decision if the software can be copied to other servers. Hence decreasing the over all execution time of the problem.

?? More and more servers would possibly be able to solve one particular problem, which otherwise was not possible.

?? Would encourage user to provide software or hardware to NetSolve system. For e.g. the user who doesn't want NetSolve to use his machine to execute any problem can just start up a software server. On the other hand, any computer can be turned into hardware server.

HW-SW Architecture

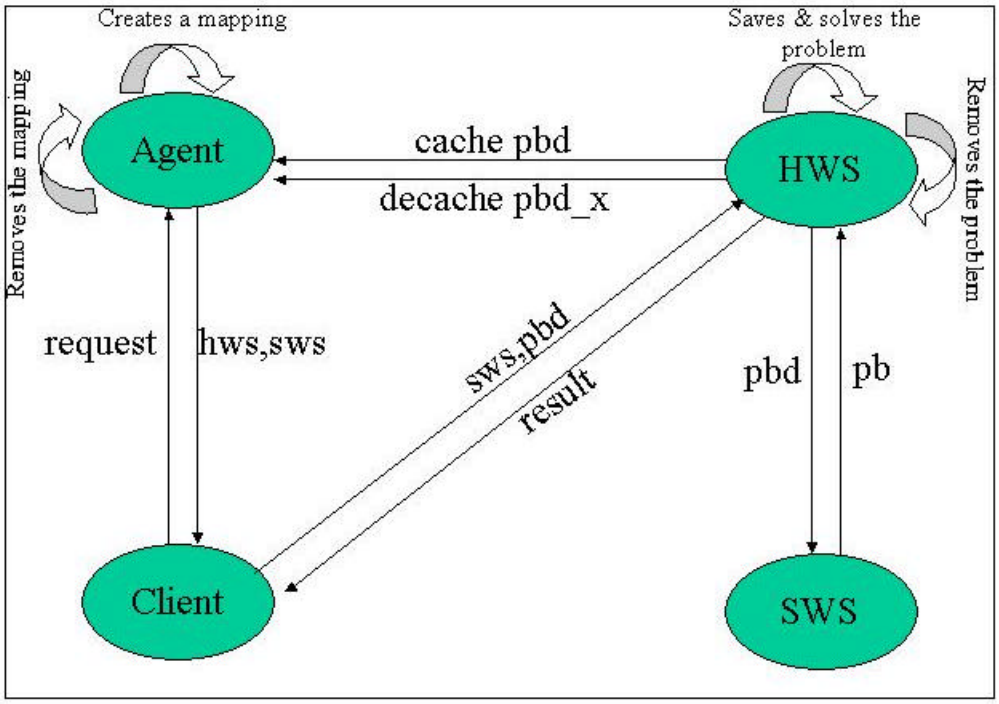

Fig. 2

Explanation

Client initiates a request to the agent with specifics of the problem it wants to solve. Agent maintains the database of hardware and software servers, upon request creates a mapping of hardware server (hws), which is capable of running the job, with the software server (sws) that has the software in its repository. The client after receiving the mapping contacts the hardware server (hws) and requests it to pull the software (pbd) from the software server (sws). Hardware server contacts the software server and requests for the software; software server sends the software (pb). Hardware server then contacts the agent and acknowledges the receipt of the problem and caches the problem with itself, runs the job and return the result back to the client. After a specified time it decaches the software.

Future work

During the course of project many things have been observed particularly regarding agents. The agent is a very critical part of NetSolve system. However the current implementation of agent is very rudimentary. The agent maintains the information about the servers and their capabilities along with their usage parameters like workload. It also carries the information like latency and bandwidth.

Currently agent creates a mapping of problem and server and upon request from the client it sends the first server available in the list. It is a possibility that the server might be located very far from the client, which is definitely not the best choice. This might cause a delay in the communication between the client and server, when there is a server very near to the client already available.

This notion can be extended to the hardware software server implementation as well. The current implementation of agent just verifies if the hardware server has the same architecture as the software server, if yes, it creates a list of mapping and sends the first mapping to the client. There is a possibility that the servers involved in the particular request are very far from each other or for that matter from client.

The NetSolve agent can be made lot powerful and intelligent. As a NetSolve policy, NetSolve really doesn't advocate high performance as long as user gets the results in reasonable time. Hence agent could make the decisions based on comprehensive analysis of information available to it.

The agent design should be reconsidered in detail. The new design should incorporate artificial intelligence to a much larger extent in agent. This would not just gather much public attention but also would be great research work.

Appendix A

New Protocols added in the hardware-software server implementation

NS_PROT_SEND_PROBLEM	Hardware server sends this value to software server, requesting to send the software.
NS_PROT_SENT_PROBLEM	This value is sent by software server to hardware server, acknowledging that the problem has been sent
NS_PROT_TRANSFER_PROBLEM	This value is sent by client to hardware server, requesting it to pull the software from software server
NS_PROT_TRANSFERRED_PROBLEM	The hardware server sends this value to the client, acknowledging that the problem has been received from the software server.

APPENDIX B

Files changed in implementation of hardware-software server

1. src/Agent/agent_process_message.c
2. src/Agent/scheduler.c
3. src/Server/server_process_message.c
4. src/Server/server_init.c
5. src/Client2Proxy/CP_sendjobrequest.c
6. src/CoreFunctions/serverdesc.c
7. src/Proxy/NetSolve/netsolveproxybasics.c
8. src/Proxy/NetSolve/netsolveproxyutil.c
9. src/Server/Standard/standardservice.c
10. include/serverdesc.h
11. include/agentprocessmessage.h
12. include/client2server.h
13. include/netsolveproxy.h
14. include/protocol.h
15. include/serverprocessmessage.h

References

1. Innovative computing Laboratory 2001 report
2. Ian Foster, Carl Kesselman et al.; The Grid: Blueprint for a New Computing Infrastructure, © 2001 Morgan Kaufmann Publishers.
3. User's Guide to NetSolve V1.4, full reference at http://icl.cs.utk.edu/netsolve