Remote Procedure Call:Implementation

Vinay, Ranjeet Kumar
Student of Information Technology
Dronacharya College of Engineering, Gurgaon

Abstract- Remote Procedure Call (RPC) is a powerful primitive used for communication and synchronization between distributed processes. Remote procedure calls (RPC) seemto be a helpful paradigm for providing communication across a network between programs written in a high-level language. RPC have a problem that it reduces the amount of parallelism, because it is synchronous in nature. This paper shows how simple processes can be used to find a way of avoiding a difficulty in this problem. We will describe how two important classes of algorithms, branch and sure are often run in exceedingly parallel manner exploitation the RPC. The results of some experiments comparing these algorithms on a single processor will be discussed. The combination of blocking RPC calls and light-weight processes provides both simple semantics and efficient exploitation of parallelism.

I. INTRODUCTION

The idea of remote procedure calls is kind of easy. It is based on the observation that procedure calls are a well-known and well- understood mechanism for transfer of control and data within a program running on a single computer [1]. In distributed systems, organizing multiple processors in several ways in which are planned. At one finish of the spectrum area unit tightly-coupled systems with multiple processors on identical bus and sharing a standard memory. At the different finish area unit the loosely-coupled systems consisting of a variety of freelance computers, every with its own software package and users, exchanging files and mail over a public knowledge network. In between, area unit systems consisting of mini or microcomputers act over a quick native network and every one running one, system-wide software package. we've used a system within the latter category as a work for the implementation of some distributed algorithms[2].

II. PARALLELISM

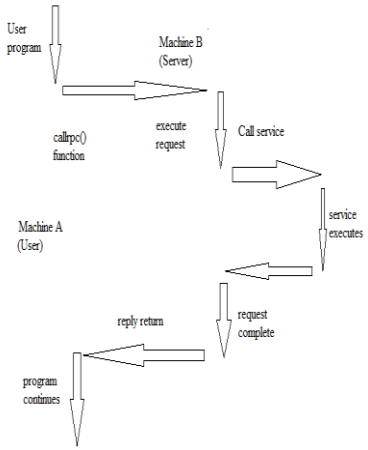
Parallel computing is a form of computation in which many calculations are carried out simultaneously, operating on the principle that large problems can often be divided into smaller ones, which are then solved concurrently. There are several different forms of parallel computing: bit-level, instruction level, data, and task parallelism.Parallelism has been employedfor many years, mainly in high-performance computing, but interest in it has grown lately due to the physical constraints preventing frequency scaling.

With RPC, when the server is active, the client is always idle, waiting for the response. Thus there is never any parallelism possible. The client and the server are effectively co-routines. With other communication models it may be possible to have the client continue computing while the server is working, in order to gain performance. Furthermore, with a single threaded server and multiple clients, the situation is even worse. While the server is waiting for, say, a disk operation, all the clients have to wait.

III. HOW RPC WORKS

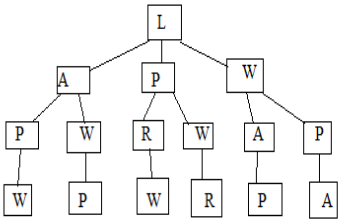
An RPC is dry to a function call. Caller waits for a response to returned from the remote procedure and the calling arguments are passed to remote procedure when an RPC is made in functional call. During the flow of activity Figure 1 shows an RPC call between two networked system .To send the request to server and wait the client make a procedure call. Until either a reply is received, or it times out the thread is blocked from processing. The server calls to send something fixed that performs the requested service, and sends the reply to the client when the request arrives .The client program continues, after the RPC is completed. Network applications are specially supported by RPC. Remote Procedure calls is uniquely identified by the triple: 1.program number , 2.version number ,3.procedure number .The Each of which has a unique procedure number is related to remote procedure to identify the program number. To enable multiple versions of an RPC protocol to be available simultaneously by version numbers. Each

version contains a number of procedures that can be called remotely.



IV. PARALLEL BRANCH AND BOUND USING RPC

For solving large class of combinational optimization problems the technique which is used is branch-and-bound method. For Travelling salesman problem, Integer programming, Machine Scheduling problems and many others are applied by branch and bound method. In this to find the shortest route for a salesman to visit each of the n cities in his occupied exactly once it chosen to implement Travelling Salesman Problem.To structure the space of possible solutions the branch and bound method uses a tree. To build the tree it is shown below by the Branching rule. A node of the tree represents a partial tour for the use of Travelling Salesman Problem. Each node has a branch for every city that is not on this not complete about it.Example is the left branch of the figure shows the Delhi-Patna-Chennai-Mumbai.To avoid the searching of whole tree the bounding rule is used. The bounding rule is simple for Travelling Salesman Problem.



V. CONCLUSIONS &RESULTS

We have performed some measurements on the Travelling Salesman Problem programs. We both performed a sequential using single processor version and a parallel using multiple processor version.for simple vision, we use only two level processor according to their importance. We used in this one processor for the client side and many six processors for the server side. For Travelling Salesman Problem the depth of the sub trees are important parameters. If we use different processors in client side then the effectiveness will be less or low.Examplefot this is, if it traverses just at one level, then the best solution in the left most branch of the tree cannot be used as a bound in its neighbor branch, as these branches are searched simultaneously. Increasing the depth of the root sub tree will decrease this effect, at the cost of more communication between the root processor and its subcontractors. To achieve highperformance, a good compromise has to be found. For an 10-city problem we found the optimal search depth of the client to be three levels. The results for an 10-city problem using this search depth are shown are below

Executed successfully

version	time (sec)
sequential	28412.54
1 server	18442.55
2 server	15231.32
3 server	13211.12
4 server	11432.31
5 server	9452.10
6 server	7132.42

The above table shows the speedup over the 1-server side. With 1 client and 6 servers a distributed program gets 4 times better result than sequential program is achieved. Note that with only one server, there is still some parallelism, as the client can find the next sub-tree to hand out, while the server is working on the previous sub-tree. Our

implementations of Travelling Salesman problem, has been Calmlykept simple at starting, as to gain experience with programming using RPC and lightweight process we are implementing them.

REFERENCES

[1]. Implementing Remote Procedure Calls by ANDREW D. BIRRELL and BRUCE JAY NELSON, Xerox Palo Alto Research Center

[2]. REMOTE PROCEDURE CALLS IMPLEMENTING USING DISTRIBUTED ALGORITHM by G. MURALI[i], K.ANUSHA[ii], A. SHIRISHA, S.SRAVYA

[3]. A Critique of the Remote Procedure Call by ParadigmAndrew S. Tanenbaum, Robbert van Renesse