ISSN:2229-6093

REMOTE PROCEDURE CALLS IMPLEMENTING USING DISTRIBUTED ALGORITHM

G. MURALI [i] K.ANUSHA[ii] A. SHIRISHA[iii] S.SRAVYA[iv]

[i] Assistant Professor, Dept of Computer Science Engineering, JNTU-Pulivendula, AP, India.

[ii],[iii],[iv] B.Tech(4-I) Dept of Computer Science Engineering, JNTU-Pulivendula, AP, India.

ABSTRACT

Remote Procedure Call (RPC) is a powerful primitive used for communication and synchronization between distributed processes. RPC poses a problem that it reduces the amount of parallelism, because of its synchronous nature. This paper shows how simple processes can be used to find a way of avoiding a difficulty in this problem. The combination of blocking RPC calls and light-weight processes provides both simple semantics and efficient exploitation of parallelism.

We will describe how two important classes of algorithms, branch and bound can be run in a parallel way using this RPC. The results of some experiments comparing this algorithms on a single processor discussed.

1.INTRODUCTION:

As computing technology advances, by just increasing the speed of the chips it becomes increasingly difficult and expensive to make the computers faster. In copper wire, electrical signals travel at 2/3 the speed of light or about 20 cm/nanosecond, so very fast computers must be very small, which leads to severe heat dissipation problems among other things. The clear solution is to achieve the same computing power as one very fast computer, but at a fraction of the cost by combining large number of moderately fast computers.

In distributed systems, organizing multiple processors in different ways have been proposed. At one end of the spectrum are tightly-coupled systems with multiple processors on the same bus and sharing a common memory. At the other end are the loosely-coupled systems consisting of a number of independent computers, each with its own operating system and users, exchanging files and mail over a public data network. In between, are systems consisting of mini or microcomputers communicating over a fast local network and all running a single, system-wide operating system. We have used a system in the latter category as a testbed for the implementation of some distributed algorithms.

1.1 Distributed System:

A distributed system consists of different independent computers that communicate with a computer network. To achieve common goals computer interacts each other. Distributed program defined as a computer program that runs in a distributed system and process of writing such programs is called distributed programming. To solve computational problems distributed computing refers to use of distributed system.

Within some geographical area networks where individual computers were physically distributed is referred as the word distributed in such as distributed system, distributed programming. The term distributed are used in wider sense, also referring to independent processes that execute on the same computer and to pass the message they intersect each other.

1.1.1 Application:

For using distributed systems and distributed computing there are two reasons..To connect several computers first application that may require to the use of communication network to communicate. Example for these is data may be produced in one location and may be used in another location. Second reason to use distributed system is single computer may be possible in principle, for practical reasons the use of distributed system is beneficial. Example for this second reason is, by using a cluster of several low-end computers it may be more cost-efficient to obtain the desired level of performance. By using a single high-end computer. Compared to non-distributed system distributed system is more efficient and reliable to use as there are no single point of failure. However, it will be easier to expand and manage than a monolithic uniprocessor system.

Examples of distributed systems and applications of distributed computing include the following:

Telecommunication networks:

1. Telephone networks and cellular networks.
2. Computer networks such as the Internet.
3. Wireless sensor network.

Network applications:

1. World wide web and peer-to-peer networks.
2. Massively multiplayer online games and virtual reality communities.
3. Distributed databases and distributed database management systems.

1.2 Remote Procedure Call (RPC):

1.2.1 What is RPC:

For constructing distributed, client-server based applications RPC is a powerful technique. As calling procedure, the called procedure need not exist in the same address space, so as it is based on the extending the notion of conventional, or local procedure calling. With a network connecting them, the two systems may be on same system or they may be on the different systems a network connection connecting them. Programmers may avoid the details of the interface with the network of distributed applications by using RPC. It allows the application to use variety of transports and physical and logical elements of the data communication and application by the transport independence of RPC isolated the application.

1.2.2 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 a 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.

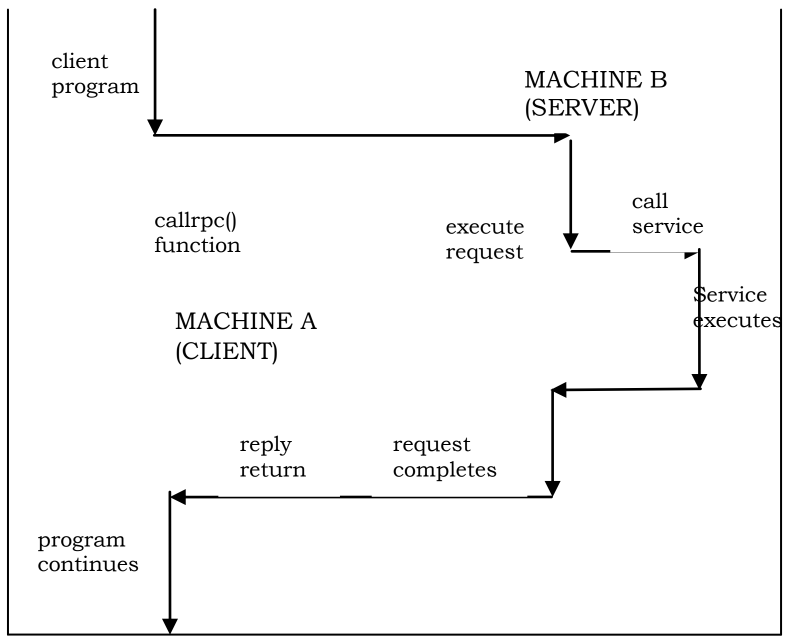


Figure 1.1: Remote Procedure Calling Mechanism

1.3 Travelling Salesman Problem(TSP):

The travelling salesman problem is an NP-hard problem in combinatorial optimization studied in operations research and theoretical computer science. In travelling salesman problem the task is to find a shortest possible tour that visits each city exactly once by using given list of cities and their pair wise distances.

In theory of computational complexity, the decision version of the travelling salesman problem (the task is to decide whether any tour is shorter than S, where given a length S) belongs to the class of NP-complete problems. Thus, it is likely that the worst case running time for any algorithm for the TSP increases exponentially with the number of cities.

The traditional lines of attack for the NP-hard problems are the following:

1. For finding exact solutions devising algorithms (they will work fast only for small problem sizes).

2. Devising suboptimal algorithms, i.e., algorithms that gives either seemingly or probably good solutions, but which could not be proved to be optimal.

3. Finding special cases for the problem for which either better or exact heuristics are possible.

exact algorithms:

To check which one is cheapest (by using brute force search) and the most direct solution would be to try all permutations, followed by combinations. The polynomial factor for running this approach lies within a time of O(n!), so this solution becomes more impractical even for only 20 cities. One of the earliest applications of dynamic programming is the HeldKarp algorithm that solves the problem in time O(n^2 * 2^n).

The exponential space is required by the dynamic programming solution. Using inclusion and exclusion, the problem can be solved in time within a polynomial factor of (2^n) and polynomial space.

Algorithm: Tree traversal algorithm:

```
procedure traverse(node,depth, length);
begin
    {node is a node of the search tree. It contains
a list of the cities on the current partial tour.
length is the length of the partial path so far.
depth is the number of levels to be searched
before the rest of the tree should be handed
out to a subcontractor }
if(length<minimum) then
begin{if(length ≥ minimum) skip this node }
if(node is a leaf) then
minimum := length;
else if(depth = 0 )then
hand out subtree rooted at node
to a subcontractor;
else
for each child c of node do
traverse(c,depth1,length+dist(node,c));
```

end

end

1.4 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 built the tree it is shown 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. Figure:1.2 shows a tree for a 4-cities. Here a leaf represents a solution of problem. Example is the left branch of the figure shows the London-Amsterdam-paris-washington. To avoid the searching of whole tree the bounding rule is used. The bounding rule is simple for Travelling Salesman Problem.

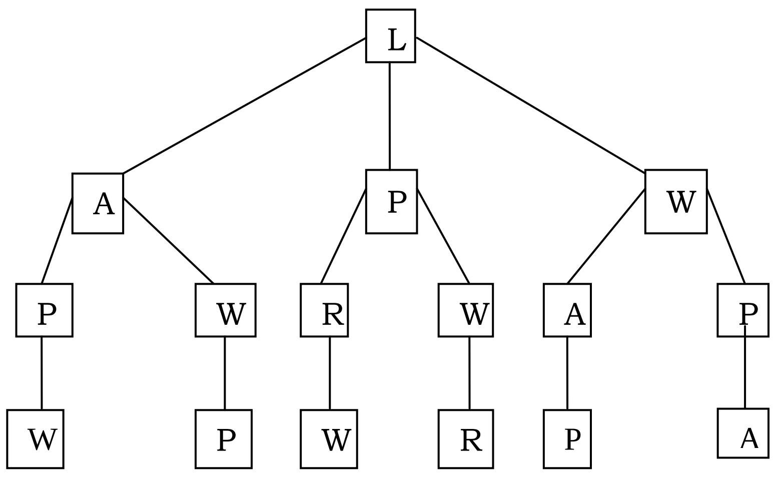


Figure 1.2: Tree of 4-city Traveling Salesman Problem for London, Amsterdam, Paris, and Washington.

The partial tour will never lead to solution better than what is already know when if the length of a partial tour exceeds the length of any already known solution. By searching the parts of the tree in parallel then parallelism will obtained in branch-and-bound algorithm. A new processor could be allocated to every node of the tree when there is enough processors were available. Every processor would select the best partial path from its children and report the result back to its parent. Branch-and-bound method would require O(N!) processors for N cities. Most knowingly among available processors the work has to be divided. In this, each processor starts at the node given to it and generates the complete partial tree reachable from that node down to depth levels. This node to a subcontractor for further evaluation each time the processor generates a node at level depth. The generation and evaluation of partial tree occur in parallel. How the Figure 1 is searched is shown by Figure 2, by using the 2-level processor according to their importance.

In Figure 1.3, the processor that traverse the root processor searches for single or one level. By subconductors, it splits off three subtrees, each of depth two, which is traversed in parallel. The algorithm is shown in Figure 3.The global variable minimum of the length of the shortest path is set to the present algorithm which is given. With a very high value the variable is initialized.

While there are no free subcontractors, a processor only blocks if it tries to hand out a subtree. With a different initial node and probably with a different initial depth, each subcontractor executes the same traversal process. Generally, a subcontractor may split up the work over even more processors, so a subcontractor may also play the role of a root processor.

By using the algorithm described above, the Traveling Salesman Problem has been implemented under Amoeba. The role of a subcontractor played by a processor can be viewed as an Amoeba server. Evaluation of a TSP sub tree is one of the most offering service. Each server repeatedly, waits for some work, performs the work, and then returns the result. A processor which plays the role of a root processor is a client.

By using Remote Procedure Calls the handing out of work is implemented. As stated before, a problem with RPC is the fact that the caller is blocked during the call.

Therefore,

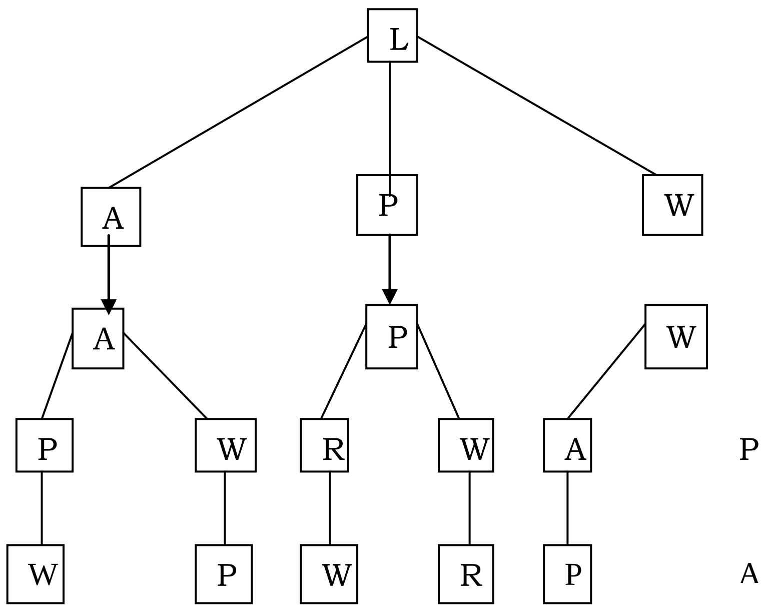


Figure 1.3: Example of a distributed tree search

the client cluster is divided into several tasks (see Figure: 1.3). To performs the tree traversal can be performed by manager task M which is in processor p running on cluster C . To contain N agent tasks like A,1 .. A,N the cluster must have N subcontractors. An agent A,j controls the communication with subcontractor j. Thus it starts the tree traversal of Figure:1.3, after the manager task Mp receives a subtree T to evaluate. It tries to find a free agent, say A,j when it finds a subtree that has to be subcontracted out,

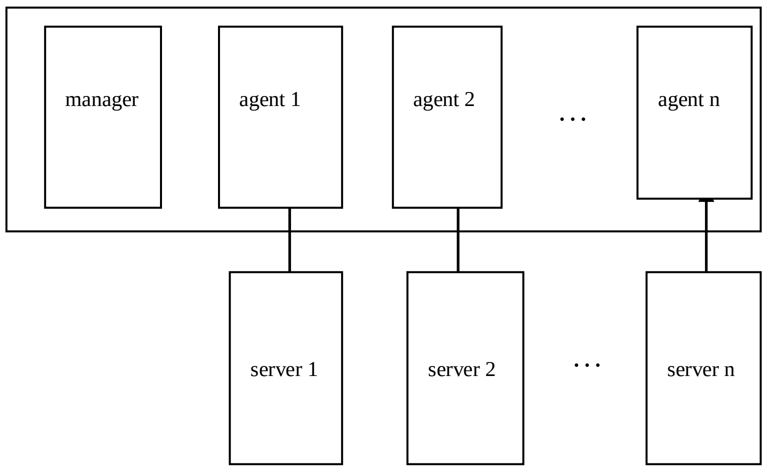


Figure 1.4: Process structure of the TSP program

Here, we describe on processor j , the manager Mj starts executing the process. It returns the result to A,j, when Mj finishes the evaluation of the subtree. This agent checks whether the current best solution has to be updated or not, and then this becomes available again for the next request from M. In the mean time, the manager M continues its tree traversal and it eagerly tries to find new work to distribute. If the manager tries to deal out work while all agents are engaged then the entire client cluster only blocks

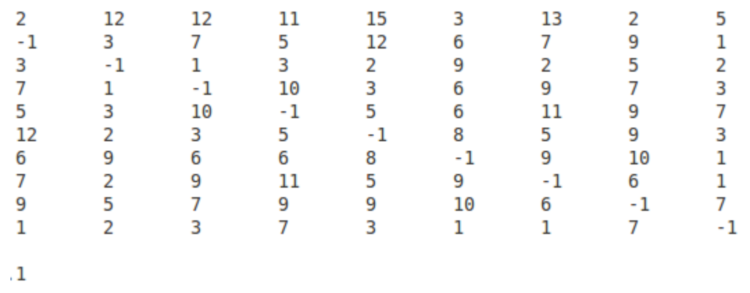
This implementation completely utilizes the parallelism present in the algorithm and it is highly flexible. The implementation uses depth-first search, but it can be easily adapted to other strategies, such as best-first or breadth-first.

Conclusions &Results:

We have performed some measurements on the Travelling Salesman Problem programs.We both performed a sequeltial using single processsor 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 subtrees are important parameters.If we use different processors in client side then the effectiveness will be less or low.Example fot 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 subtree will decrease this effect, at the cost of more communication between the root processor and its subcontractors. To achieve high performance, 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

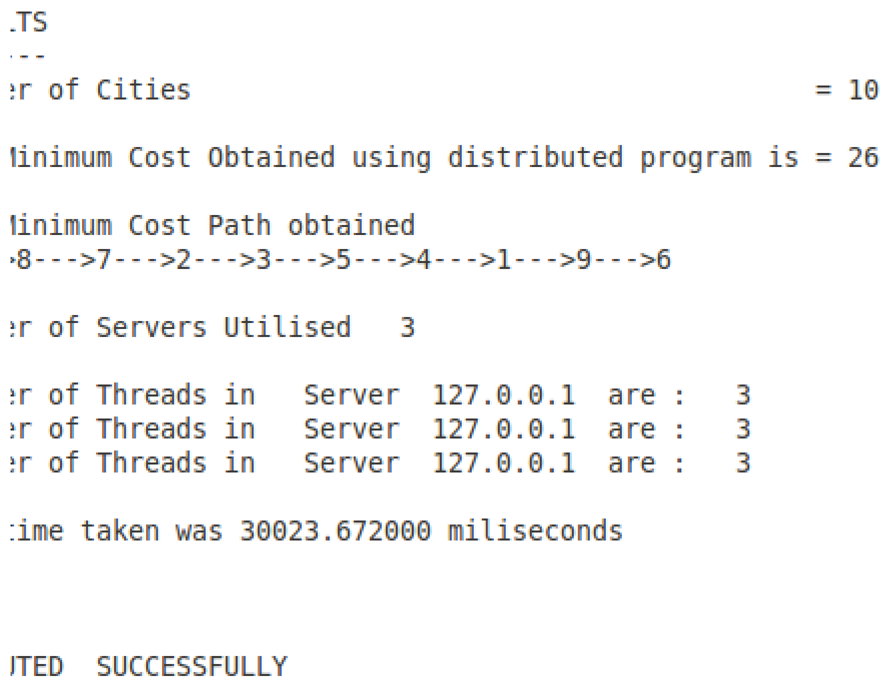
RESULTS:



```
2       12      12      11      15      3       13      2       5
-1      3       7       5       12      6       7       9       1
3       -1      1       3       2       9       2       5       2
7       1       -1      10      3       6       9       7       3
5       3       10      -1      5       6       11      9       7
12      2       3       5       -1      8       5       9       3
6       9       6       6       8       -1      9       10      1
7       2       9       11      5       9       -1      6       1
9       5       7       9       9       10      6       -1      7
1       2       3       7       3       1       1       7       -1

.1
```

Figure 1.5: Input Data



```
.TS
---
er of Cities                                        = 10

linimum Cost Obtained using distributed program is = 26

linimum Cost Path obtained
-8--->7--->2--->3--->5--->4--->1--->9--->6

er of Servers Utilised   3

er of Threads in   Server  127.0.0.1  are :   3
er of Threads in   Server  127.0.0.1  are :   3
er of Threads in   Server  127.0.0.1  are :   3

time taken was 30023.672000 miliseconds



JTED  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

Table 2: Results for 10-city traveling salesman problem

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 subtree to hand out, while the server is working on the previous subtree. Our implementations of Travelling Salesman problem, has been Calmly

ISSN:2229-6093

kept simple at starting, as to gain experience with programming using RPC and lightweight process we are implementing them.

REFERENCES:

[1]Birrell, A. D. and Nelson, B. J.,(Feb. 1984)“Implementing Remote Procedure Calls”, ACM Transactions on Computer Systems, Vol. 2, No. 1, pp. 39-59.

[2] Tanenbaum, A. S. and Van Renesse, R.,(Dec. 1985)“Distributed Operating Systems”, Computing Surveys, Vol. 17, No. 4, pp. 419-470.

[3] Mullender, S. J. and Tanenbaum, A. S.,(Aug. 1986)“Design of a Capability-Based Distributed Operating System”, Computer Journal, Vol. 29, No. 4, pp. 289-299.

[4] Tanenbaum, A. S. and Mullender, S. J.(July 1981)“An Overview of the Amoeba Distributed Operating System”, Operating Syst. Rev., Vol. 15, No. 3, pp. 51-64.

[5] Nelson, B. J.(May 1981)“Remote Procedure Call”, CMU-CS-81-119, Carnegie-Mellon University.

[6] The Amoeba Distributed Operating System, Andrew S. Tanenbaum& Gregory J. Sharp ,VrijeUniversiteit, De Boelelaan 1081a, Amsterdam, The Netherlands.

[7] Bruce Jay Nelson (May 1981). Remote Procedure Call.Xerox Palo Alto Research Center.PhD thesis.

[8] Remote Procedure Calls (RPC) A tutorial on ONC RPC by Dr Dave Marshall of Cardiff University.

[9] Tanenbaum, Andrew S. and Gregory J. Sharp. The Amoeba Distributed Operating system.VrijeUniversiteit, Amsterdam.