INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

Dynamic Load Calculation in A Distributed System using centralized approach

Biswajit Sarma
Assistant Professor,
Department of
Computer science and Engineering
Jorhat Engineering College
email:eduneristbiswa@gmail.com

Srishti Dasgupta
Final Year student,
Department of
Computer science and Engineering
Jorhat Engineering College
srishti.rhea@gmail.com

Abstract: The building of networks and the establishment of communication protocols have led to distributed systems, in which computers that are linked in a network cooperate on a task. **The task is divided by the master node into small parts (sub problems) and is given to the nodes of the distributed system to solve, which gives better performance in time complexity to solve the problem compared to the time required to solve the problem in a single machine. Load balancing is the process of redistributing the work load among nodes of the distributed system to improve both resource utilization and job response time while also avoiding a situation where some nodes are heavily loaded while others are idle or doing little work. So before sending these parts of problem by the master to the nodes, master node should know the actual work load of all the nodes. We try a dynamic approach to find out the work load of each participating nodes in the distributed system by the master before sending the parts of the problem to the nodes.**
This paper describes an algorithm which runs in the master machine and collects information from the nodes of the distributed system (client server application) and calculates the current work load of the nodes of the distributed system. The algorithm is developed in such a way that it can calculate the loads of the nodes dynamically. This means the loads can be evaluated if new nodes are added or deleted or during current performance of the nodes. The whole system is implemented on linux machine and local area network.

Keywords— Distributed system, Load balancing, Resource utilization.

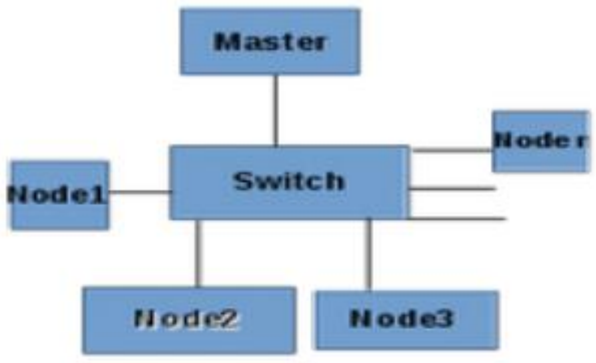
I. INTRODUCTION

A distributed system is a collection of independent computers linked by a network that appears to the users of the system as a single system which is using software to produce an integrated computing facility. A single problem is divided into some parts of small problems and is sent to the nodes of the distributed system. By doing so this system acquires better performance and reliability. A distributed system may have more total computing power than a mainframe if the nodes of the distributed system are loaded perfectly according to each node's capacity. To give the perfect load to each node the master node must know the resources like available memory and CPU utilization of each node. Before sending the sub problems to the nodes, the master node needs to know about the current load and computing ability of the nodes of the distributed system, so that it can give perfect load to each node and get a better performance. Moreover for better performance in distributed system two other points are considered. First of all if one machine crashes, the system as a whole can still survive and secondly computing power can be added in small increments. These two points are very important for dynamic load calculation. There are two possibilities in a network. Due to some network failure one of the nodes may get disconnected from the distributed system or a node may gracefully terminate from the distributed system. In these two situations the master must know that a particular node got disconnected from the distributed system and it has to distribute that unfinished work to other nodes according to their current load. Dynamic load balancing algorithm uses current load information about the active nodes for distribution of the unfinished work to maintain the balanced state of the system.

II. LOAD CALCULATION AND ITS USE IN LOAD BALANCING IN DISTRIBUTED SYSTEM

The traditional load balancing problem deals with load unit migration from one processing element to another when load is light on some processing elements and heavy on some other processing elements. It involves migration *decision,* i.e. which load unit(s) should be migrated and then *migration* of load unit to other nodes.

The works are distributed among nodes by the master node and the master node has to make decisions on how much work it is going to give to each node in the system. These decisions are made according to the current resource utilization in each node. To do this load calculation in each node the master node has two possibilities 1. Static approach and 2.dynamic approach. In static approach we assume that before the work is distributed among the nodes of the system all the nodes are connected and no network failure occurs in such case and nodes cannot terminate during the execution of the work. Since the nodes are connected before the actual distributed computing starts and it remains there until computation finishes so static approach needs only one load calculation just before the work is distributed to the nodes. In the case of dynamic approach no such assumption is made like no network failure occurs or nodes cannot join or terminate from the distributed system. In dynamic approach continuous load calculation is required for each node and it is done by the master node. The assumption made in the static model is of no real value because network failure can occur at any point of time and to increase the computing power of the distributed system new nodes must be allowed to join the system. If a node may terminate due to some reason then the master node has to wait until it get the result of that failure node in a static model which time will be infinite. So in comparison, dynamic model is better compare to static model. In our case since we are using a master node for load calculation our approach is master node approach.



III. DYNAMIC LOAD CALCULATION APPROACH AND ISSUES

The dynamic load calculation approach has two most important policies:

1. To count the number of connected nodes at a particular time. Some node may terminate itself or new nodes may want to connect to the system at any point of time. The master node must keep track of this.

2. The current nodes have to appear in the master node's list with their current load.

These two points must work simultaneously so it gives a result so that load balancing can be performed accurately.

During the design of a dynamic load calculation algorithm the following issues are considered:

1. The whole experiment is based on a local area network (LAN segment).

2. Machines with different resources (memory size and CPU speed) can join the distributed system.

3. Thread is light weighted process ,so when master node calculates load it uses less resources (CPU and memory). Moreover we use the mechanism of lock and unlock and control the execution of thread whenever it is required.

4. A signal is an asynchronous notification sent to a process or to a specific thread within the same process in order to notify it of an event that occurred. We use signals to send notification to a particular thread whenever it is required.

5. The top program provides a dynamic real-time view of a running system. It can display system summary information as well as a list of processes or threads currently being managed by the Linux kernel. We use TOP command to find out the CPU usage and memory usage for each node and calculate the load.

IV. PROPOSED ALGORITHM

4.1. Master side algorithm:

4.1.1. Main thread:

Step1. Create a socket and bind to the well-known address (IP address and port) for the service being offered.

Step2. Place the socket in passive mode.

Step3. Accept the next connection request from the socket, and obtain a new socket for the connection and store the IP address of the new connection.

Step4. In the first connection two threads are created:

a) Menu thread which gives the connected nodes of the distributed system at a particular time and it will be alive forever and will repeat the execution of codes.

b) Information threads which gives the memory and CPU information for a particular IP chosen by the master user.

4.1.2. In menu thread:

Step5. Initially when it is executed this thread sends a particular message (hello) to each IP address connected to the master machine. This will check how many nodes are connected at that moment with the master machine (by checking a reply from nodes alive).

Step6. When the reply messages from the nodes are received the master node forms a menu which shows the connected nodes to the master machine for that particular time.

Step7.From the IP address user can choose a particular node for current information about the node. Such information is on CPU usage and memory information.

Step8. After choosing a particular node from the list it sends a signal to the information thread to transfer the control to that information thread.

Step9. After returning from the information thread it collects the information for that node.

Step10. Display the information (in percentage) for each IP address.

4.1.3. In the Information thread:

Step 11.Initially it waits for SIGNAL. When it receives a SIGNAL it use a mutex lock (that mutex variable is also used in the menu thread for execution sequence).

Step12. Sends request for CPU and memory Usage of the node, gets the current load of node and stores in a data structure according to the IP address.

4.2. In general NODE side algorithm:

Step1. Try to connect to the master with SOCKET.

Step2. Wait for messages and reply according to the message it receives:

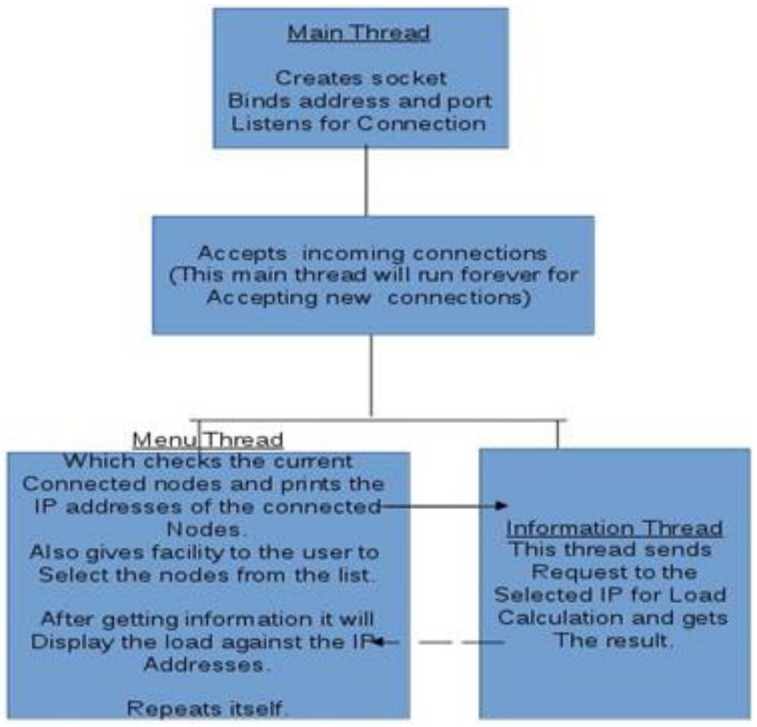
a) If it receives “hello” message (it is basically used for checking whether the node is alive or not), reply back “alive” to the master.

b) Else if it receives MEMORY and CPU information request it sends back the total percentage of work load in the node that is calculated by the following formula:

((CPU LOAD %)+(MEMORY LOAD %))/2 (average of the CPU LOAD AND MEMORY LOAD) and return it.

To calculate the CPU LOAD % and MEMORY LOAD % we use linux TOP command.

4.3Diagram in master side:


Main Thread
Creates socket
Binds address and port
Listens for Connection
Accepts incoming connections
(This main thread will run forever for Accepting new connections)
Menu Thread
Which checks the current Connected nodes and prints the IP addresses of the connected Nodes.
Also gives facility to the user to Select the nodes from the list.
After getting information it will Display the load against the IP Addresses.
Repeats itself.
Information Thread
This thread sends Request to the Selected IP for Load Calculation and gets The result.

V. CONCLUSIONS

We have studied the distributed system load balancing which provides better performance in a distributed system. The main intention of this paper is to exhibit how to provide the latest information about the nodes in the distributed system. This information can be used in a dynamic load balancing algorithm. This can provide a better result for distributed system.

REFERENCES

[1] Parveen Jain and Daya Gupta "An Algorithm for Dynamic Load Balancing in Distributed Systems with Multiple Supporting Nodes by Exploiting the Interrupt Service" in International Journal of Recent Trends in Engineering, Vol 1, No. 1, May 2009.

[2] Vatsal Shah and Kanu Patel "Load Balancing algorithm by Process Migration in Distributed Operating System" in international Journal of Computer Science and Information Technology & Security (IJCSITS), ISSN: 2249-9555 Vol. 2, No.6, December 2012.

[3] Rafiqul Zaman Khan and aved Ali " Classification of Task Partitioning and Load Balancing Strategies in Distributed Parallel Computing Systems" in International Journal of Computer Applications (0975–8887) Volume 60–No.17, December 2012.

[4] Jayna Donga , Vatsal Shah, Kanu Patel and Rahul Goradia "Process Division and Migration based Load Balancing for Distributed Operating System" in International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 12,December 2013.