Middle-East Journal of Scientific Research 23 (9): 2135-2144, 2015
ISSN 1990-9233
© IDOSI Publications, 2015
DOI: 10.5829/idosi.mejsr.2015.23.09.22514

Distributed Data Mining and Dynamic Load Balancing Algorithms in Cluster of Novel Mobile Agent Frameworks Using TCP

[1]S. Kavitha and [2]K.V. Arul Anandam

[1]Research Scholar in Computer Science Research and Development Centre, Bharathiar University, Coimbatore - 641 046, and Assistant Professor, SRM University, Chennai, Tamil Nadu, India
[2]Department of MCA, Govt. Thirumagal Mills College, Gudiyatham-632602, Vellore, Tamil Nadu, India

Abstract: The Load balancing is an important one for the development of parallel and distributed computing applications and also improves the efficiency and performance of distributed system by allowing load migration for the purpose of efficient resource utilization. The aim of the distributed system is sharing of resources. Mobile agents provide a novel technology for implementing load balancing mechanism on the cluster of workstation. It is intelligent software program which migrates in heterogeneous network. The mobile-agent based approach can minimize the network traffic and enhance the flexibility of a load balancing mechanism. Load balancing is a computer networking method to distribute workload evenly across two or more computers, network links, CPUs, hard drives or other resources. It aims to optimize resource use, maximum throughput, minimize response time and avoid overload of any of the resources. Load balancing is good resource utilization. A Dynamic load balancing algorithm in green parallel and distributed computing using mobile agent is proposed and compared with the client-server load balancing algorithm and shown efficiency of the algorithm.

Key words: Mobile Agent · Load Balancing · Green Computing · Parallel and Distributed Computing · TCP

INTRODUCTION

Mobile agent technology provides a more efficient solution then the client-server based approach in terms of management of distributed resources. Traditional technologies achieve communication among distributed resources by using location transparency, while mobile agents provide local interaction for communication and mobile logic facilities. It migrates on remote hosts and performs different operations. It can be effectively used in many areas with several reasons including improvements in latency and bandwidth of client-server applications as well as reducing vulnerability to network disconnection [1]. Mobile logic and local interaction, load balancing, low traffic in the network and flexibility are the important reasons that sustain the use of mobile agents for the management of distributed resources. Mobile agent can improve load balancing for three reasons as follows. The first reason is to improve the efficiency and performance that the mobile agent can reduce data transmitting, save network bandwidth and overcome network latency because it can more independently and transfer computations into data fields. The second reason is to improve the reliability that the mobile agent can be executed asynchronously and independently on destination nodes. The third reason is to improve the adaptability that the mobile agent is intelligent, mobile, flexible and active so it can complete assigned tasks substituting origin host. It can also adopt according to the changing of environment and respond to it.

Load balancing is dividing the amount of work that a computer has to do between two or more computers so that more work gets done in the same amount of time and in general, all users get served faster [2]. It can be implemented with hardware, software or a combination of both. It requires multiple servers and also usually combined with failover and use backup services. The advantages of load balancing is to equalize the workload among the nodes by minimizing execution time, minimizing communication delays, maximizing resource utilization,

Corresponding Author: Dr. K.V. Arul Anandam, Department of MCA, Govt. Thirumagal Mills College, Gudiyatham-632602, Vellore, Tamil Nadu, India.

maximizing throughput, improve performance, decreasing consumption of system resources and distributing users across available servers based on requirements for workgroup and load sensitivity [3]. There are two factors which help to achieve green computing in load balancing of cluster of computers. First factor is reducing energy consumption: Load balancing helps in avoiding overheating by balancing the workload across all the nodes of cluster of computers, hence reducing the amount of energy consumed. Second factor is reducing carbon emission [4]. Energy consumptions and carbon emissions are directly proportional to each other. The more the energy consumed, higher is the carbon footprint. The energy consumption is reduced with the help of load balancing so; the carbon emission is also reduced. Then can achieve green computing [5].

The link between energy consumption and carbon emission has increased due to the demand of ever-increasing computing and storage problems arising in the internet age which is improve energy-efficiency in parallel and distributed computing and to achieve green computing [6]. Load balancing assist to achieve a higher user satisfaction and resource utilization ratio and also improving the overall performance and utilization of resource of the systems in distributed environment [7].

The followings are the Load balancing Metrics which is consider at the development of algorithm in green cluster.

Throughput: It is used to calculate the number of tasks whose execution has been completed. It should be high to improve the system performance.

Overhead Associated: It determines the amount of overhead involved while implementing a load-balancing algorithm. It includes overhead due to movement of tasks, inter-processor and inter-process communication. It should be minimized.

Fault Tolerance: It is the ability of an algorithm to perform uniform load balancing in case of link failure. The load balancing should be a good fault-tolerant technique.

Migration Time: It is the time to migrate the jobs or resources from one node to other. It should be minimized in order to enhance the performance of the system. It should be minimized.

Response Time: It is the amount of time taken to respond by a particular load balancing algorithm in a distributed system.

Resource Utilization: It is used to check the utilization of resources. It should be optimized for an efficient load balancing.

Scalability: It is the ability of an algorithm to scale according to the requirement.

Performance: It is used to check the efficiency of the system. This has to be improved at a reasonable cost, e.g., reduce task response time while keeping acceptable delays.

The main aim of using mobile agent in the load balancing is that it could be successfully utilized in increasing the system performance with the following reasons. Mobile Agent reduces the network load. i.e., it visits all nodes in cyclic manner instead of all-to-all communication. It encapsulates protocols. i.e., Protocols defined for task exchange that can be encapsulated in the agent. It executes asynchronously and autonomously. i.e., the agent's creator is free after agent's creation and dispatching. It adapts dynamically. i.e., the agent will react according to current situation. It is heterogeneous in nature. i.e., the load balancing could be applied to heterogeneous systems. It is robust and fault-tolerant. i.e., if a host or link is being shut down, all agents could change their paths and continue their operation on another host in the network.

The rest of this paper is organized as follows. In the section that follows, some concepts of load balancing with mobile agent is briefly reviewed. Section 2 shows the node structure in distributed load balancing. Section 3 present novel mobile agent model of distributed dynamic load balancing. Section 4 represent novel mobile agent model of distributed in cluster. Section 5 discussed the TCP package in distributed load balancing. Section 6 proposed distributed dynamic load balancing algorithm in green cluster using mobile agent model. Section 6 discussed the experimental analysis. Finally section 7 summarizes the main contributions of this paper.

Node Structure in Distributed Dynamic Load Balancing: Each node needs to exchange status information with every other node in the system. There are four policies or strategies in dynamic load balancing. Fig. 1 illustrates the structure of the node in distributed load balancing [8, 9].

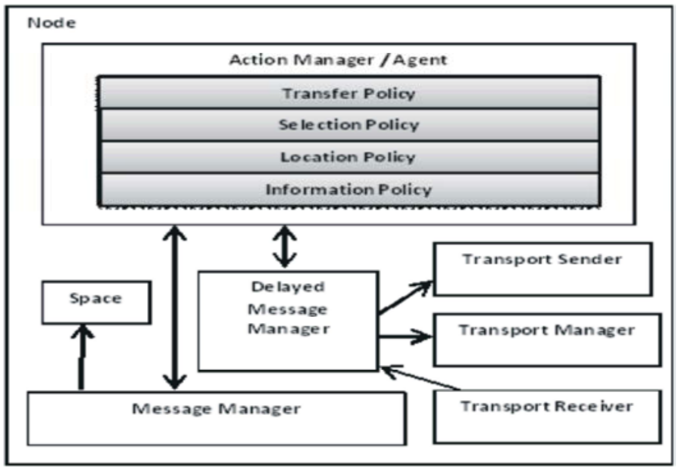

Fig. 1: Node Structure in Distributed Load Balancing

Transfer Policy: The policy which selects a job for transferring from a local node to a remote node is referred to as Transfer policy or Transfer strategy. It is dynamic in nature. Each node will be computing its extra workload with reference to average load in the system. This module executes two predefined policies like local and global. According to the response time of the job, it submitted the job for execution. If the response time of job at local site is less than the global site then local policy is executed otherwise global policy is executed.

Selection Policy: It identified the processors involved in the load exchange. This module calculates the CPU, Memory and I/O of each resource on a particular node.

Location Policy: It refers to the strategy used in finding an appropriate task transfer partner. It is mostly used the polling approach. Overloaded nodes will identify randomly set of under loaded nodes. It selects a destination node for a transferring the task. If the node is failure, connection established to the backup node and assign the task to that node because of avoiding searching time for assigning task to another node otherwise communication cost will increase.

Information Policy: It collects information about the nodes in the system and also provides details about the nodes to another node for further processing. It has system idle, CPU, Memory and Input/output etc. details. It updates the status every changes in the system utilization.

Novel Mobile Agent Model of Distributed Dynamic Load Balancing: Mobile Agent is a light-weight Agent because it carries only the instructions which are given by the Interface Agent on behalf of Actor. Message Manager handles the messages passed between actors. Delayed Message Manager holds the messages for Mobile Agent while they are moving from one site to another site. Actor Manager manages the state of actors that are currently executing and also the locations of mobile actors created on the site. Actor Manager act as an agent that have four policies to assign the loads evenly and also proper utilization of resources and energy that leads to the green computation. Space provides middle agent services such as matchmaking and brokering services. The mail addresses or names of all agents are not globally known in the open agent system and also an agent may not have the other agent's addresses with whom it needs to communicate. For this, middle agent services need to be supported. Transport Manager contains a public port for message passing between different sites. Transport Sender on the same platform receives the message from the Message Manager and delivers it to the Transport Receiver in the different site. Facilitator agent acts as a load balancer. It collects information's from all site's databases and submit to the clients database. (Global mining). Fig. 2 describes the novel mobile agent technology that used in distributed dynamic load balancing concepts [10].

Novel Mobile Agent Model of Distributed Dynamic Load Balancing in Cluster: This model consists of cluster of novel mobile agent models with large distributed network. Each model has local databases for storing the large size of data after processing the data with unlimited nodes. In this model, mobile agent only carries the instruction which is obtained from all nodes and leaves the processed data in the corresponding local databases because to improve the speed of migration from one node to another

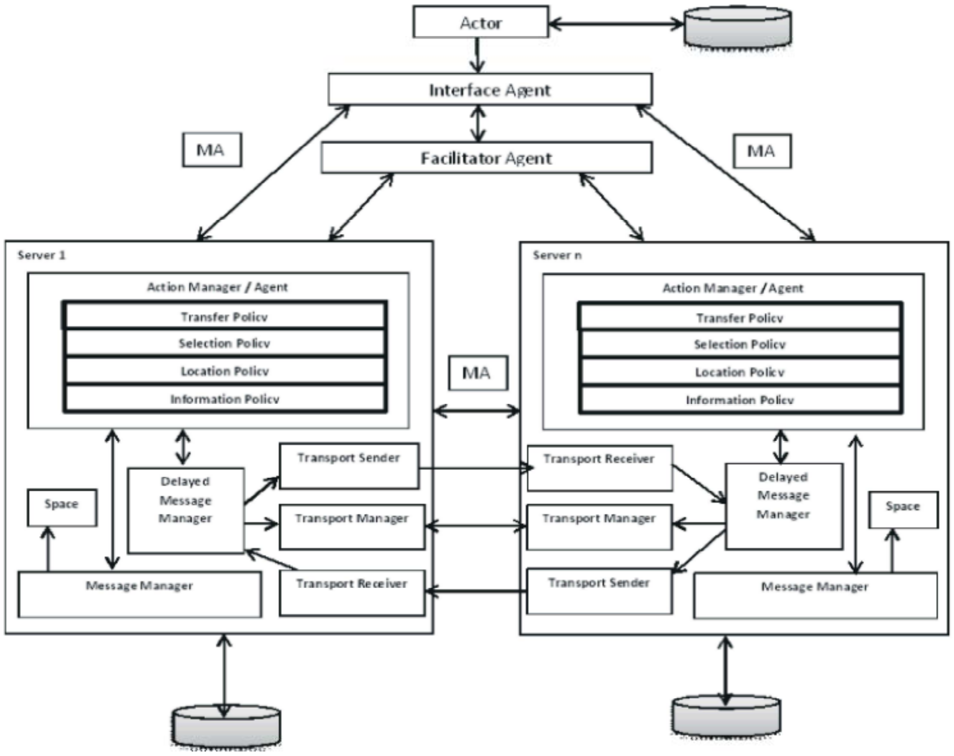

Fig. 2: Novel Mobile Agent Model of Distributed Dynamic Load Balancing

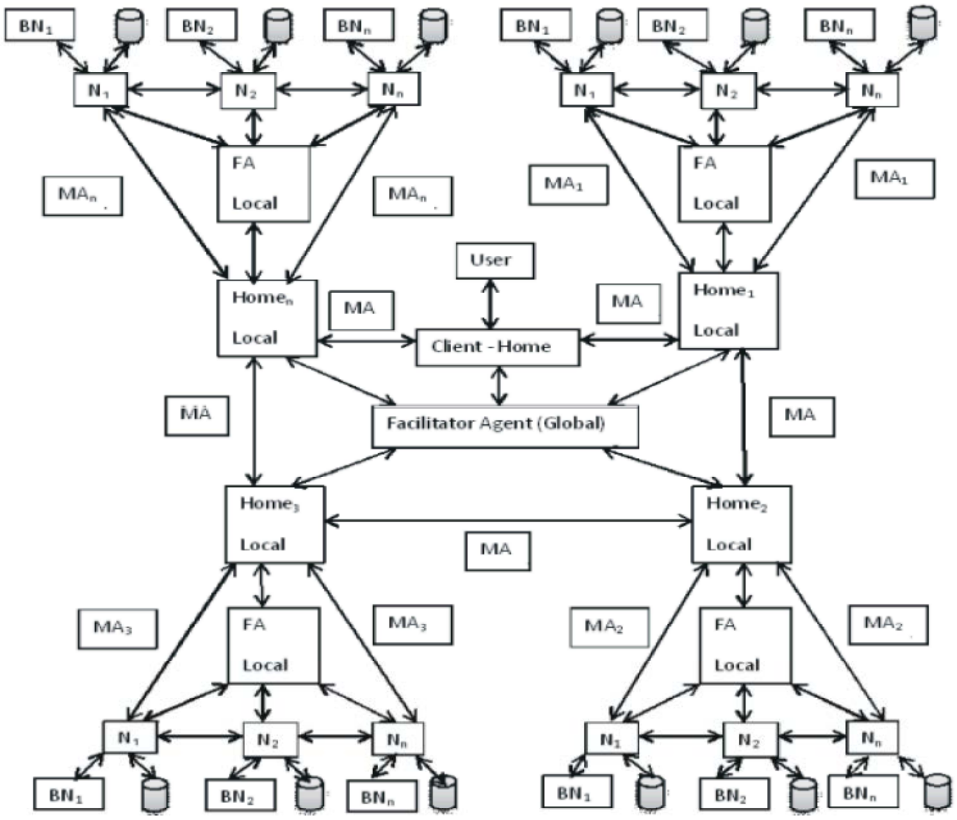

Fig. 3: Novel Mobile Agent Model of Distributed Dynamic Load Balancing in Cluster

node and also reduce the mobile agent container size. Finally, Mobile agent returns to the home. Every home node act as server for the main server and continue the above processes. Each model connected with the backup server whenever server failure which is easily connected to the backup servers for avoiding unnecessary communication in the network and also time consuming. It makes proper load balancing with the use round robin algorithm and also priority based allocation of loads among the servers [11, 12]. Proper load balancing leads to the reduction of energy consumption and carbon emission that leads to the green computing [11]. Fig. 3 shows Novel Mobile Agent Model of Distributed Dynamic Load Balancing in Cluster.

Novel Mobile Agent based Distributed Data Mining Algorithm in Cluster:

Notations:

DB	-	Database.
N	-	Number of Rows.
I	-	Partition of Database.
D	-	Number of Transaction.
n	-	Number of sites (S1, S2, < Sn).
DBi	-	Distributed Data sets at Si, DB =U DBi, i= 1 to n.
X.Sup	-	Support count of a X at DB –Global.
X.Supi	-	Support count of a X at DBi –Local.
Minsup	-	Minimum support threshold.
GFI	-	Global Frequent Item Set.
CGFI	-	Candidate Global Frequent Item Set.
X	-	Global Frequent Item iff, X.Sup>= minsup * D.
X	-	Local Frequent Item iff, X.supi>= minsup * Di.
LFI I	-	Local Frequent Item set at site-i.
PGFI	-	Possible Global Frequent Item Sets- These are item sets at sites-i, which are not part of LFI i, but by adding these count at central place converts.
CGFI	-	Cluster Global Frequent Item Sets. These are item sets at novel mobile agent model-i, which are not part of LFI i, but by adding these count at central place converts.

Input: Distributed-Data-Set DBi i=1 to n, minsup, Distributed sites' address.

Output: Global Frequent Item set – GFI

- Mobile Agent (MA) launches from Agent Submitter Interface based on the client to all sites.
- MA initialize with the load state in the server, perform the local functions, save the current states and store the task in the local data ware-house.

for i=1 to n do

{MA.send (Location=I, S=Support, Address of all distributed sites);}

- Agent Submitter Interface assigns the work to facilitator agent and facilitator agent partition the work into all the sites. It is done in parallel.

for I=1 to N begin

{ Assign the data to each site from partition I; }

- MA has the number of nodes to visit and do the step 2 processes in every node up to reach the last node.
- Likewise MA move all the server, do all tasks and store in the local data warehouse.
- If it is the last node and last server, it returns to the home.
- Each Cooperative MAi Computes LFII in parallel from each local database, PGFI and their count at each site.

for I=1 to n do
{If frequent present then
Compute LFII
else

PGFI j=all sites = All Item Sets at site-j ◎LFIi, i=1 to n,
} i<>j

- Send LFI to facilitator agent and also send count of PGFIi, i=1 to n, to facilitator agent from each infrequent site.
- Compute GFI and CGFI at facilitator agent. GFI=◎ LFIi,i=1 to n; CGFI= ◎ LFIi - ◎ LFIi, i=1 to n.
- Calculate GFI at facilitator agent using PGFI count.

for all X ε CGFI do

{If X.Sup= X.Sup I, i=1 to n>=MinSup*D then { GFI=GFI ◎ {X}}}

Note: Step 9 and step 10 are performed in parallel.
Note: Step 1 to 10 is done parallel in each frame work separately.

- GFI of each novel mobile agent send to the corresponding client (Local Home).
- Global Facilitator Agent calls the above steps from 1 to 10 and performs the task in parallel. Finally, collects the CGFI and send to the user for viewing the result from the client.

TCP Package in Distributed Load Balancing: TCP is a stream-service, connection-oriented protocol with an involved state transition diagram and uses flow and error control. The TCP package can simulate the heart of TCP as represented by state transition diagram [13].

Transmission Control Blocks (TCBs): TCP uses a structure to hold information about each connection for controlling the connection. TCP keeps an array of TCBs in the form of a table because any time there can be several connections. This is referred to as TCB.

Fields Includes in Each TCBs Using TCP Connection:

State – It defines the state of the connection according to the state transmission diagram.
Process – It defines the process using this connection at this machine as a client or a server.
Local IP address – It defines the IP address of the local machine.
Local Port Number – It defines the local port number.
Remote IP address – It defines the IP address of the remote machine.
Remote Port Number – It defines the remote port number.
Interface – It defines the local interface.
Local Window – It comprise several subfields and holds information about the window at the remote TCP.
Remote window - It comprise several subfields and holds information about the window at the remote TCP.
Sending sequence number – It holds the sending sequence number.
Receiving sequence number – It holds the receiving sequence number.
Sending ACK number – It holds the value of the ACK number sent.
Round-trip time – Several fields may be used to hold information about the RTT.
Time-out values – Several fields can be used to hold the different time-out values such as the retransmission time-out, persistence time-out, keep alive time-out.
Buffer size – It defines the size of the buffer at the local TCP.
Buffer Pointer – It is a pointer to the buffer where the receiving data is kept until it is read by the application.

Timers: TCP needs to keep track of its operations.

Main Module: The main module is invoked by an arriving TCP segment, a time-out event or a message from an application program. The action is to be taken depends on the current state of the TCP so it is a very complicated module.

Input Processing Module: Input processing module handles all the details needed to process data or an acknowledgment received when TCP is in the ESTABLISHED state. It sends an ACK if needed takes care of the window size announcement, does error checking.

Output Processing Module: Output processing module handles all the details needed to send out data received from application program when TCP is in the ESTABLISHED state. It handles retransmission time-outs, persistent time-outs and so on.

State Transition Diagram for Both Client and Server: It keeps track of all different events happening during connection establishment, connection termination and data transfer. The TCP software is implemented as a finite state machine. A finite state machine that goes through a limited number states. The machine is in one of the states. It remains in that state until an event happens. The event can take the machine to a new state. At the same time, the event can also make the machine perform some actions. This is shown in Fig. 4 and also Table 1 is list out the states and description for TCP.

TCP Connection: A connection –oriented transport protocol establishes a virtual path between the source and destination. The followings are the phases of transmissions [14].

Connection Establishment: TCP transmits data in full-duplex mode. When two TCPs in two machines are connected, they are able to send segments to each other simultaneously. i.e., each party must initialize communication and get approval from the other the other party before any data is transferred. The connection established in TCP is called three-way handshaking. For example, an application program (Client) wants to make a connection with another application program (Server) using TCP as the transport layer protocol.

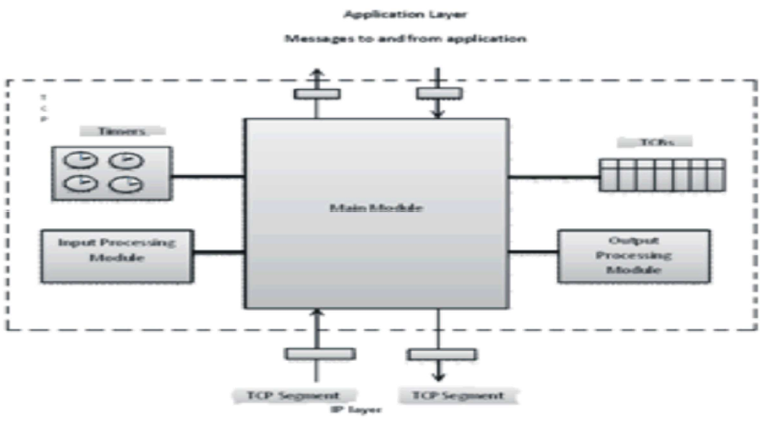

Fig. 4: TCP Package

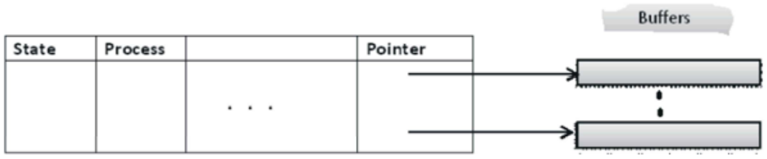

Fig. 5: TCBs

Table 1: States for TCP

State	Description
CLOSED	There is no connection.
LISTEN	Passive open received; waiting for SYN.
SYN-SENT	SYN sent; waiting for ACK.
SYN-RCVD	SYN+ACK sent; waiting for ACK.
ESTABLISHED	Connection established; data transfer in progress.
FIN-WAIT-1	First FIN sent; waiting for ACK.
FIN-WAIT-2	ACK to first FIN received; waiting for second FIN.
CLOSE-WAIT	First FIN received, ACK sent; waiting for application to close.
TIME-WAIT	Second FIN received, ACK sent; waiting for 2MSL time-out.
LAST-ACK	Second FIN sent; waiting for ACK.
CLOSING	Both sides have decided to close simultaneously.

Data Transfer: After connection is established, bidirectional data transfer can take place. The client and server can send data and acknowledgments in both directions.

Connection Termination or Connection Reset: Any of the two parties involved in exchanging data (client and server) can close the connection, although it is usually initiated by the client. There are two options for connection termination that are three-way handshaking and four-way hand=shaking with a half-close option. Connection Reset is that the TCP at one end may deny a connection request, may abort a connection or may terminate an idle connection. All of these are done with the RST (reset) flag. Connection establishment and termination and Connection termination using three-way handshake are shown in Fig. 5. and Fig. 6 respectively.

Distributed Dynamic Load Balancing Algorithm in Green Cluster Using Mobile Agent Model

Calculation of Threshold Value: The threshold value of a node is the limiting value of its workload and is used to decidewhether a node islightly or heavily loaded. The threshold value of a node may be determined as follows:

$$\text{Threshold value of a node } (n_i) = \frac{\sum_{i=1}^{N} W}{N} * C_i$$

W_i – Amount of workload of a node.
N – Number of node in our system.
C_i – Predefine constant value depends on the processing capability of node n_i.

Distributed Dynamic Load Balancing Algorithm Using Mobile Agent Model in TCP: Load balancing algorithms have two goals. 1. To improve the load distribution. 2. To minimize the communication for achieving the proper load balancing. The load balancing algorithm is implemented in the TCP package because to reduce the communication among the nodes for load balancing and also introduce the back-up server whenever the main server was failure [15, 16].

Receive: a TCP segment, a message from an application or a time-out event.

```
1. Search the TCB table.
2. If (corresponding TCB is not found)
    1. Create a TCB with the state CLOSED.
3. Find the state of the entry in the TCB table.
4. Case (state)
```

Closed:

```
1. If ("passive open" message from application received)
    1). Change the state to LISTEN.
2. If ("active open" message from application received)
    1). Send a SYN segment.
    2). Change the state to SYN-SENT.
3. If ( any segment received)
    1). Send an RST segment.
4. If (any other message received)
    1). Issue an error message.
5. Return.
```

SYN-SENT:

```
1. If (time=out)
    1). Change the state to CLOSED.
2. If (SYN segment received)
    1). Send a SYN + ACK segment.
    2). Change the state to SYN-RCVD.
3. If (SYN + ACK segment received)
    1). Send an ACK segment.
    2). Change the state to SYN-RCVD.
4. If ( any other segment or message received)
    1). Issue an error message.
5. Return.
```

SYN-RCVD:

```
1. If (ACK segment received)
    1). Change the state to ESTABLISHED.
2. If (time-out)
    1). Send an RST segment.
    2). Change the state to CLOSED.
3. If("close" message from application received)
    1). Send a FIN segment.
    2). Change the state to FIN-WAIT-1.
4. If(RST segment received)
    1). Change the state to LISTEN.
5. If any other segment or message received)
    1). Issue an error message.
6. Return.
```

ESTABLISHED:

```
1. If (FIN segment received)
    1). Send an ACK segment.
    2). Change the state to CLOSE-WAIT.
2. If ("close" message from application received)
    1). Send a FIN segment.
    2). Change the state to FIN-WAIT-1.
3. If ( RST or SYN segment received)
    1). Issue an error message.
4. If ( data or ACK segment received)

RequestResources (no-of-resources, amount-of-workload)
{ rs = Query resources information from the nodes;
Sort the available resources with amount-of-workload by ascending order;
i = 1;
while ( rs.next( ) && i <= no-of-resources ) {
if ( node.WL + amount-of-workload < thresholdvalue ){
// if query is local then executes in local nodes otherwise in global nodes based on response time;
// Round Robin algorithm for allocation of resources.
if (node.status ="Failure"){
Connection established with the Back-up node;
Node is in sleep state;
URLLIST[i][0] = backupnode.HOST_ID;
URLLIST[i][1] = backupnode.HOST_ADDRESS;}
else
{Back-up node is in sleep state;
URLLIST[i][0] = node.HOST_ID;
URLLIST[i][1] = node.HOST_ADDRESS; }
Current_workload = node.WL + amount-of-workload;
Update the Current_workload to the relevant address;}
i++; }
return URLLIST;}
5. If ( "send" message from application received)
    1). Result obtained after processing the queries.
6. Return.
```

FIN-WAIT-1:

```
1. If (FIN segment received)
    1). Send an ACK segment.
    2). Change the state to CLOSING.
2. If ( FIN + ACK segment received)
    1). Send an ACK segment.
    2). Change the state to TIME-WAIT.
3. If ( ACK segment received)
    1). Change the state to FIN-WAIT-2.
4. If ( any other segment or message received )
    1). Issue an error message.
5. Return.
```

FIN-WAIT-2:

```
1. If (FIN segment received)
    1). Send an ACK segment.
    2). Change the state to TIME-WAIT.
2. Return
```

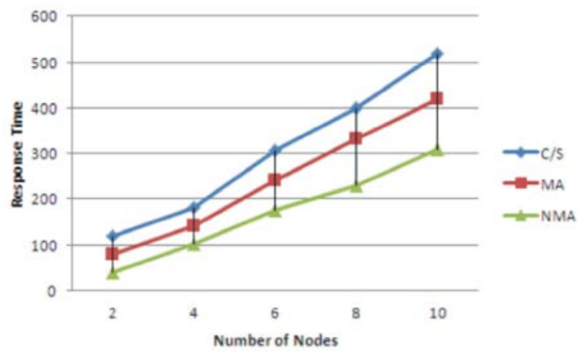

Fig. 6: Performance Scale Up of Different Algorithms with Number of Nodes and Response Time

CLOSING:

```
1.  If (ACK segment received)
        1). Change the state to TIME-WAIT.
2.  If (any other segment or message received)
        1). Issue an error message.
3.  Return
```

TIME-WAIT:

```
1.  If (time-out)
        1). Change the state to CLOSED.
2.  If (any other segment or message received)
        1). Issue an error message.
3.  Return
```

CLOSE-WAIT:

```
1.  If ("close" message from application received)
        1). Send a FIN segment.
        2). Change the state to LAST-ACK.
2.  If (any other segment or message received)
        1). Issue an error message.
3.  Return
```

LAST-ACK:

```
1.  If (ACK segment received)
        1). Change the state to CLOSED.
2.  If (any other segment or message received)
        1). Issue an error message.
3.  Return
```

Performance Analysis: The load balancing algorithms are implemented in java programming with RMI concept. As illustrated Fig. 6 that shows the performance scalability and also proper allocation of loads among the nodes. It also indicates the improvement of the novel mobile agent load balancing algorithm than the client server and mobile agent based load balancing algorithms. The novel mobile agent based load balancing algorithm using TCP is used for allocating the works properly among the nodes. This proper load balancing leads to increase the network life because of the individual nodes energy saving and also energy consumed is much less as compared to the other load balancing algorithms.

DISCUSSION

The cluster novel mobile agent framework with distributed data mining algorithm is introduced with parallel and distributed processing. The back-up node is introduced for proper load balancing which is in sleep mode. When the server is failure, back-up node set into active mode and it accepts the task for the process but server set to sleep mode for power management. In the traditional algorithm, when the server sent failure notice, communication time increases for again search of another node to process. Here, the communication overhead is reduced because of the introduction of the back-up node. The load balancing is proper among the cluster of computers that leads to reducing the usage of power and carbon emission among nodes. Mobile agent can improve the load balancing because of improving efficiency and performance, reliability and adoptability. Mobile Agent reduces the network load because it visits all nodes in cyclic manner instead of all-to-all communication. Energy consumption should be well-managed by optimal routing designs. This is achieved by the mobile agent because routing instructions are clearly defined in the mobile agent The TCP package also introduced because of full-duplex service, connection-oriented service and reliable service. It has three-way hand shaking communication so; unnecessary communication among nodes is avoided. All these concepts are used for the proper load balancing that leads to green clusters.

CONCLUSION

This paper describes the framework of our proposed efficient distributed data mining and load balancing techniques using mobile agent in green cluster computing for high speed computing and optimum resource utilization with reduction of energy and carbon emission. Our proposed algorithm use mobile agent to distribute work load in a cluster. The mobile agent can travel cyclic manner among the hosts in a network and also carry the instructions about the details of the network of disconnection environment. According to the preliminary results, system gives a good results and something to be modified to reach the target. The results will be discussed in detail in future work.

REFERENCES

1. Yun Yali, Yaping Li and Haixiao Han, 2012. A Mobile Agent-based Secure and Efficient task Allocation Algorithm for Cloud and Client Computing, 2012 Fourth international Conference on Multimedia Information Networking and Security, pp: 1-3, 978-0-7695-4852-4, IEEE, DOI 10.1109/MINES.2012.30.
2. Chechina Natalia, Peter King and Phil Trinder, 2010. Using Negotiation to Reduce Redundant Autonomous Mobile Program Movements, 2010 IEEE/WIC/ACM International Conference on Web Intelligence Agent Technology, pp: 343-346, 978-0-7695-4191-4,() IEEE, DOI 10.1109/WI-IAT.2010.22.
3. Htwe Thandar Su Nge, Aye Thida and Pa Pa Nyunt, 2011. Mobile Agents Based Load Balanced Resource Scheduling System, 978-1-61284-840-2, IEEE, 486-489.
4. Barazandeh Iman and Seyed Saeedolah Mortazavi, 2009. Two Hierarchical Dynamic Load Balancing Algorithms in Distributed Systems, 2007 Second International Conference on Computer and Electrical Engineering, 978-0-7695-3925-6/09, IEEE, DOI 10.1109/ICCEE.2009.253, pp: 516-521.
5. Lin Kai, Min Chen, Sherali Zeadally and Joel J.P.C. Rodrigues, 2011. Balancing energy consumption with mobile agents in wireless sensor networks, Future Generation Computer Systems, Elsevier, DOI 10.1016/j.future.2011.03.001, 28: 446-456.
6. Hussin Masnida, Young Choon Lee and Albert Y. Zomaya, 2011. Priority-based scheduling for large-scale distribute systems with energy awareness, IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing, 978-0-7695-4612-4/11, DOI 10.1109/DASC.2011.96, 503-509.
7. Mayuri A. Mehta, 2012. Designing an Effective Dynamic Load Balancing Algorithm Considering Imperative Design Issues in Distributed Systems, International Conference on Communication Systems and Network Technologies, 978-0-7695-4692-6/12 IEEE DOI 10.1109/CSNT.2012.92, pp: 397-401.
8. Nehra, Neeraj and R.B. Patel, 2007. Towards dynamic load balancing in heterogeneous cluster using mobile agent, 2007 International Conference on computational intelligence and multimedia applications, 0-7695-3050-8/07 () IEEE DOI 10.1109/ICCIMA.2007.209.
9. Aakanksha and Punam Bedi, 2007. Load balancing on dynamic networking using mobile process group, 0-7695-3059-1/07 () IEEE DOI 10.1109/ADCOM.2007.27, pp: 553-558.
10. Kavitha S., K. Senthil Kumar and K.V. Arul Anandam, 2014. Mobile Agent Model with Efficient Communication for Distributed Data Mining, S. Sathiakumar *et al*. (eds.), Proceedings of International Conference on Internet Computing and Information Communications, Advances in Intelligent Systems and Computing 216, DOI:10.1007/978-81-322-1299-7_39, ©Springer India ISBN: 978-81-322-1298-0 (Print) 978-81-322-1299-7 (Online), pp: 421-429.
11. Nehra Neeraj, R.B. Patel and V.K. Bhat, 2006. A Multi-Agent system for Distributed Dynamic Load Balancing on Cluster, pp:135-138, 1-4244-0716-8, IEEE.
12. Gondhi Naveen Kumar and Dr. Durgesh Pant, 2009. An Evolutionary Approach for Scalable Load Balancing in Cluster Computing, pp: 1259-1264, 978-1-4244-2928-8, IEEE International Advance Computing Conference (IACC 2009).
13. Forouzan Behrouz A., Data Communications and Networking', Fourth Edition, McGraw-Hill Higher Education (McGraw-Hill Forouzan Networking Series), New York, ISBN 978-0-07-296775-3 - ISBN 0-07-296775-7.
14. Forouzan Behrouz A., TCP/IP Protocol Suite', Third Edition, TATA McGraw-Hill Publishing Company Limited, Third Edition, New Delhi, ISBN 0-07-296772-2.
15. Deng Huafeng, Zhong Linhui and Ye Maosheng, 2010. An efficient load balancing algorithm in distributed systems', 2010 International Forum on Information Technology and Applications, 978-0-7695-4115-0/10, IEEE DOI 10.1109/IFITA.2010.285, pp: 397.
16. Kavitha, S. and K.V. ArulAnandam, 2014. Distributed Dynamic Load Balancing in Green Cluster of Mobile Agent Frameworks Using TCP, International Symposium on Research Innovation for Quality Improvement in Higher Education 2014, Paper id PID:2307, The Research and Development Centre together with Social Science, Science and Arts Departments of Bharathiar University, Coimbatore.