International Journal on Future Revolution in Computer Science & Communication Engineering ISSN: 2454-4248
Volume: 4 Issue: 1 09 – 16

A Hybrid of Improved Bulls and Weighted Round Robin to optimize the Leader and Load Balancing in Cloud and Distributed Computing Environment

Suvarna Lakshmi C,
Assistant Professor,
Dept of Computer Science,
Aurora's Degree and P.G. College,
Chikkadapally, Hyderabad,
Telangana, India,
vcsl189@gmail.com.

M. V. Ramana Murthy,
Professor & Head,
Dept. Of Mathematics &
Computer Science
MGIT, Gandipet, Hyderabad,
Telangana, India,
mv.rm50@gmail.com.

Rajani Bellamkonda
Asst. Professor, Dept. of Computer
Science,
Aurora's Degree and P.G. College
Chikkadapalli, Hyderabad,
Telangana, India
Email-id: bkrajani@gmail.com

S. China Ramu
Assoc. Professor, Dept. of CSE,
CBIT, Gandipet, Hyderabad, Telangana,
India,
Email-id:chinaramu@cbit.ac.in

Abstract: Day by Day there is increase of internet users which leads to increase the traffic in the network which causing the generation of huge data. It requires the balancing of network load on the network servers with different Load balancing techniques. It is also required to have efficient algorithm to analysis the huge data in distributed manner to identify the leader to act as centralized point of contact for services. If we audit on the heap adjusting systems, there are a few potential outcomes to upgrade the methods. In the present scenario, we have the methods, round robin algorithm (static load adjusting), Weighted Round Robin algorithm and Least Load algorithm (Dynamic Load Balancing). A researcher D. Chitra Devi .et .al has given the idea of enhanced weighted round robin algorithm (EWRR) which gives much better reaction when contrasted with basic round robin calculation. Another scholar Rashmi Saini et. al recommended the half breed of round robin calculation and minimum Load Algorithm.

From the above scholars' articles, I hereby propose a resolution by improved Bulls algorithm along with Weighted Round Robin (WRR) algorithm to achieve high performance in Distributed and Cloud Computing domain in terms of leader election from a group of distributed and non-failed processes, load balancing dynamically and coordinate other nodes.

Bulls algorithm uses the following message types:

- **Election Message**: Sent to announce election.
- **Answer (Alive) Message**: Responds to the Election message.
- **Coordinator (Victory) Message**: Sent by winner of the election to announce victory.

When coordinator fails to recover a process P, from failure or detecting before failure, the process P performs the following actions:

1. If P has the highest process id, it sends a Victory message to all other processes and becomes the new Coordinator. Otherwise, P broadcasts an Election message to all other processes with higher process IDs than itself.
2. If P does not receive any Election message, then it broadcasts a Victory message to all other processes and becomes the Coordinator.
3. If P receives an Answer from a process with a higher ID, it sends no further messages for this election and waits for a Victory message. When there is no Victory message after a stipulated period, it restarts the process from the beginning.
4. If P receives an Election message from another process with a lower ID it sends an Answer message back and starts the election process at the beginning, by sending an Election message to higher-numbered processes.
5. If P receives a Coordinator message, it treats the sender as the coordinator.

Keywords: *Bulls Algorithm, Cloud computing, Distributed systems, Load Balancing, weighted round robin, least load balancing algorithm.*

*****

I. INTRODUCTION

The Fig:1 shows the environment of Load Balancing and schedule design in distributed and cloud environment.

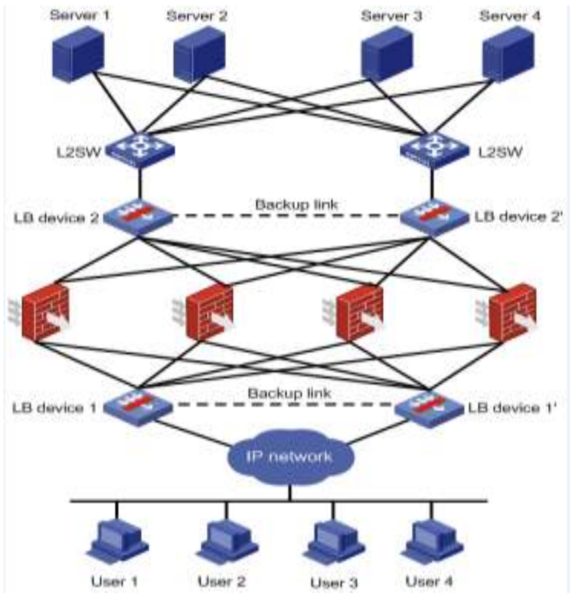

Fig1: Schedule & load balance design

Let us take 4 servers (Server1, through Server4) are interconnected with two routers (L2SW1 & L2SW2), and 2 Load Balancing devices (LB Device2 and LB Device 2') are connected to two routers (L2SW1 & L2SW2). The 2 Load Balking devices connected to another 2 Load Balancing devices (LB Device 1 and LB Device1') through interconnected Fire walls for security purpose and connected through cloud to different users located geographically.

Any request of the cloud should reach to either Load Balancing device 1 or Load Balancing device 2. Then request upwards to the Server and response will be received in any direction.

Following are the scheduling and load balancing algorithms.

1) Dynamic Load Balancing

It distributes the total work load among all the processors at runtime before initiating the process. The master assigns new processes to the slaves based on the new information collected.

2) Least Load Algorithm [3]:

```
BEGIN PROCEDURE LEAST_LOAD_ALGO
array SERVERS = {$s_1, s_2, s_3 ... s_n$};
array SERVER_LOAD = {$l_1, l_2, l_3 ... l_n$};
WHILE(request) DO
Integer POS=FIND_MIN(SERVER_LOAD);
   GOTOSERVER[POS];
END WHILE

END PROCEDURE

BEGIN PROCEDURE FIND_MIN (SERVER_LOAD [1 to n])
Integer POS=0;
For I=1 to n-1 do
IF SERVER_LOAD[POS]>SERVER_LOAD[I]THEN
  POS=I;
END IF
RETURN POS;
END PROCEDURE
```

The above algorithm uses the policy of Shortest Remaining Processing Time which is the optimal algorithm for minimizing mean response time. The job that has the least remaining process time will be served. Some of the demerits in this policy are

- The dispatcher or load balancer or job scanner should know the time required to execute the job before its actual execution.
- Larger size jobs may need to wait for a while.

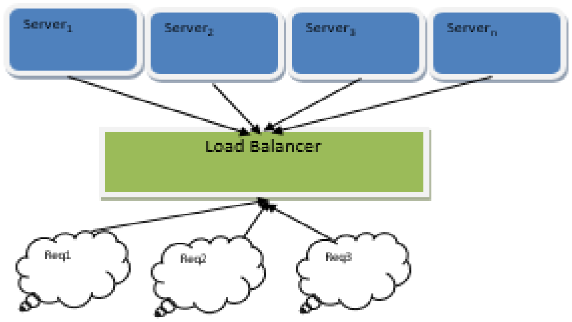

Fig2: Least Load for client request

3) Static Load Balancing

In Static Load Balancing, the processor's execution is resolved towards the start of its execution. At that point just, the work stack is conveyed to the processors towards the begin level as per the execution decided. At the underlying stage, the processors are thought to be free. It sends message to all the remote processors with respect to new the heap state, if the heap condition of the processor surpasses a heap level farthest point. On the off chance that it isn't over-burden than the procedure is distributed locally.

4) Round Robin [3] Algorithm

Round robin is a simple continuous looping technique, in which the user content access request is responded by the load balance in the circular fashion, handles all the processes. The first access grants to the first available server by giving its IP Address, and second to the second server IP Address and so on in cyclic manner. Whenever a server IP Address is given, instantly its IP Address is moved back to the list of available IP Addresses and gradually it moves back to the top of the list and becomes available again.

```
BEGIN PROCEDURE ROUND_ROBIN
STATIC INTEGER COUNT=0
INTEGER QUANTUM=q
array SERVERS = {$s_1, s_2, s_3 ... s_n$};
INTEGER SQ = q*n
```

```
WHILE (request) DO
INTEGER Range = count % SQ

IF Range > 0 and Range <= Q
THEN set SERVER[S1]
ENDIF

IF Range > 1*Q and Range <= 2*Q
THEN set SERVER[S2]
ENDIF

IF Range > (N-1) *Q and Range <= N*Q
THEN set SERVER[Sn]
ENDIF

END WHILE
END PROCEDURE
```

5) Weighted Round Robin [9] Algorithm:

In this algorithm, the resource capabilities of the machine are considered and the machines having the higher capacity assigns the higher number of tasks, based on which weightage is given to each machine. But the issue is that it doesn't think about the length of the assignments to choose the suitable machine.

6) Enhanced Weighted Round Robin (EWRR) [9] Algorithm:

D. Chitra Devi. et.al states that the EWRR calculation is the most optimal and it assigns the job to most suitable machine by considering the machine's data like its handling limit and length of the arrived tasks with its priority. The static scheduling of this algorithm uses the processing capacity of the machine. The allocation of task to a suitable machine is decided based on the length of each task..

The dynamic scheduling option (at run time) of this calculation also utilizes the load on each of the Virtual Machines (VM) along with the information mentioned above to decide the allocation of the task to the appropriate machine. During run time, there is a probability that the task may take longer execution time than the initial calculation, due to the execution of more number of cycles (like a loop) on the same instructions based on the complicated run time data.

In such situations, the load balancer rescues the scheduling controller and rearranges the jobs according to the ideal slot available in the other unutilized/underutilized machines by moving a waiting job from the heavily loaded machines.

Nature of Load Balancing [3]

- **Co-agent**: In cooperative situation, all processors have the responsibility to complete their own bite of task undertaking, yet all processors cooperate to accomplish an objective for better efficiency. In non-cooperative situation individual processor go about as autonomous elements and touch base at choices about the utilization of their assets with no impact of their choice on whatever remains of the framework.

- **Process Migration:** When a system decides to export a process, it provides process migration parameter too. It decides whether to create locally or remotely. This algorithm is capable to decide that it should make changes of load distribution during execution of process.

- **Resource Utilization**: It include automatic load balancing. A distributed system might have unexpected number of processes that demand more processing power. In such cases the algorithm is capable to utilize resources efficiently or can be moved to underutilized processors.

Both Round Robin and Least Load Algorithms have their own drawbacks. Both are working with great specific criteria, yet additionally have certain restrictions. The main disadvantage of Round Robin is that load balancing of various sizes and complexity of load or request due to lack of precision. Where as in Least Load Algorithm, the load at server is considered before distributing it, which results in low efficiency of the system. Only a set of servers gets load in case of sparse load condition thereby leaving some of the systems ideal.

Assets	Least Load	Round Robin
Nature	Dynamic	Static
Stability	Medium	High
Cooperative	Yes	No
Resource Utilization	Medium	Low

Table1: Comparison of Load Balance Algorithms [3]

Property	Round Robin	Weighted Round Robin	Enhanced Weighted Round Robin	Least Load
Nature	Static	Static	Dynamic	Dynamic
Stability	High	High	Higher	Medium
Co operative	No	No	No	Yes
Resource Utilization	Low	Medium	High	Medium

Table2: Round Robin, Weighted Round Robin, EWRR, Least Load comparisons

ANALYSIS EVIDENCE

Server	Total Data (MB)	Request per Sec	Bandwidth (Bytes per Sec)	Busy Threads
Main	168	5.78	480.23	8
Server1	66	2.89	152.678	2
Server2	78	3.032	167.35	4
Server3	99	2.732	154.897	3

Table3: Round Robin [3]

Server	Total Data	Request	Bandwidth (Bytes per	Busy Threads

	(MB)	per Sec	Sec)	
Main	189	6.078	512.899	7
Server1	57	3.189	168	3
Server2	65	3.032	176.789	4
Server3	57	3.94	164.234	3

Table4: Weighted Round Robin [3]

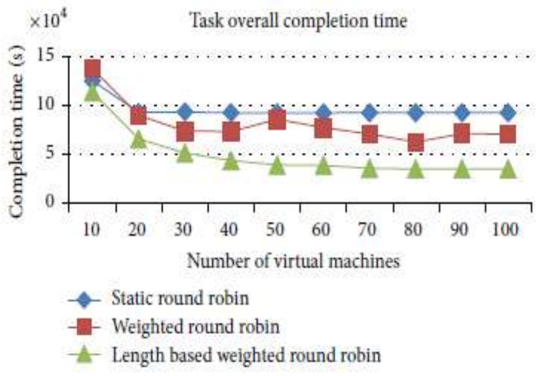

Fig:3[9] Execution Completion time

Above tables shows the comparisons of all algorithms whether single or hybrid. The Table3 data represents that the main server has 8 busy threads whereas server 1 has only 2 threads that means the threads are not equally distributed.

Table4 represents the least load algorithm, which shows that as compared to round robin, least load algorithm is much better.

Fig:3 is the hybrid of the two which gives the better result as proved by Rashmi Saini.et al. Table4 and Fig:3 shows the efficiency of WRR algorithm.

To improvise faster and efficient processing in Distributed and Cloud computing environment, we have election algorithm so called Bully algorithm. The Bully algorithm used in distributed computing system for dynamically leader election based on process ID. The highest process ID number is elected as the leader or coordinator.

The aim of an election Algorithm execution is selecting the leader that all processes agree with it. In other words, electing a process with the highest priority or highest ID number as a leader or coordinator without other processes contradicting this decision.

Assumptions

- Each process knows the process ID and address of every other process
- Communication is reliable
- A process initiates an election if it just recovered from failure or it notices that the coordinator has failed
- Three types of messages: Election, OK, Coordinator
- Several processes can initiate an election simultaneously
- Need consistent result

Bully Algorithm Model

- Any process P can initiate an election.
- P sends ***Election*** messages to all process with higher process IDs and awaits the response.
- The response is called ***OK*** message
- When there is no "OK" message, then P becomes coordinator and sends Coordinator messages to all processes with lower process IDs.
- If it receives an *OK message*, it drops out and waits for the ***Coordinator's*** message
- If a process receives an *Election* message, immediately sends *Coordinator* message if the process has highest process IDs
- Otherwise, returns an *OK message*, and starts the election
- If a process receives a *Coordinator* message, then it treats the sender as the coordinator.

Electing a leader is a classical problem in distributed computing system. Synchronization between processes often requires one process acting as a coordinator. If an elected leader node fails, then the other nodes of the system need to elect another leader **without wasting** the time. The bully algorithm is a classical approach for electing a leader in a synchronous distributed computing system, which is used to determine the process with highest priority number as the coordinator. In this scenario, we have discussed the limitations of Bully algorithm and proposed a simple and efficient method for the Bully algorithm which reduces the number of messages during the election. Our analytical mockup shows that, the proposed algorithm is more efficient than the Bully algorithm.

II MOTIVATION

The fundamental disadvantage of Bully algorithm is that, it has more number of message passing and do not have fault tolerant. As it is specified, before message passing it has the order O(n^2). It increases network traffic due to five stages to decide next leader. Hence, it would waste lot of time for the process to continue their normal execution process.

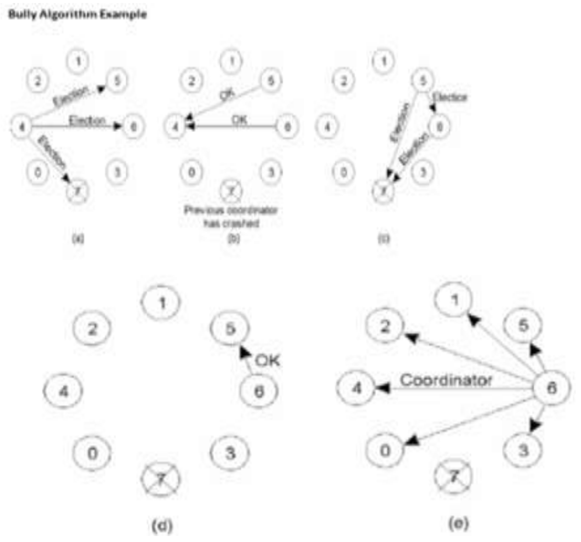

Fig4: Stages of Bully Election Algorithm

Fig:4 explains the various stages of Bully leader election algorithm. Let us consider, there are 7 groups of processes and 4th process identifies as the leader process and failed, due to not having the response from the leader.

- Process 4 sends ELECTION message to its higher processes. That is processes 5, 6 and 7.
- Process 5 and 6 respond by sending OK messages, telling process 4 that they would take over the execution of electing the leader.
- Process 5 and 6 holds an election individually leading to two simultaneous elections.
- The reply of OK message from process 6 to process 5 tells that it would continue the process of electing the leader.
- Process 6 waits for a clock slice time for a reply from process 7. Since process 7 does not send the reply due to failure, process 6 will be declared as leader and inform to the rest of all processes as coordinator.
- After the timeout, process 7 wins the election and informs to all the processes by sending the COORDINATOR message and terminates the election algorithm.

Advantages and limitations

The advantages of Bully algorithm are that this algorithm is a distributed method with simple implementation. [5] [2] [11]

This technique requires at most five phases and the likelihood of recognizing a crashed process. During the execution of algorithm, it is lowered in contrast to other algorithms. Therefore, other algorithms impose heavy traffic in the network in contrast to Bully algorithm [10]. The major advantage of this algorithm is that only the processes with higher priority number respect to the priority number of process, that detects the crash coordinator will be involved in election, but not all the processes are involved. However, the two major limitations of Bully algorithm are the number of stages to decide the new leader and the huge number of messages exchanged due to the broad-casting of election and OK messages [1]

Improved Bully Algorithm:

Generally, in fault-tolerant distributed systems the leader node must perform some specific controlling tasks and this node is well known to the other nodes. This node does not necessarily possess any extra processing feature to become elected, but having the highest process id. Election algorithms need a special mechanism to elect the leader. After crash failure of the leader node, it is urgently needed to reorganize the existing active nodes to call for an election and to elect a leader to continue the operation of the entire system.

Besides having all the assumptions of the existing algorithm, we assume that

- All processes hold an election flag
- If the flag is true election cannot be initiated by any process.
- All processes have a variable to store coordinator information.

Step1

Initially all election flags are set to false. When a process, P, notices that the coordinator crashed, it initiates an election algorithm

- P sends an ELECTION message to all nodes.
- All processes set their election flag to true, so that no other process can start parallel election until current election reaches the end.
- Coordinator variable reset to zero.
- If nobody responds, P wins the election and becomes a coordinator.

Step2

Once the process receives ELECTION message from one of the processes with lower numbered:

- The receiver sends an OK message back to the sender to show that it is alive and will assume control.
- The sender P separate process ID of beneficiary and store it in coordinator variable. Only process IDs greater than the stored ID can override the coordinator ID value.
- Finally, all processes responded higher process ID among them is stored in coordinator variable value.
- The sender P gather coordinator ID from variable and educated it (coordinator process ID) as coordinator.
- The elected coordinator process cross check with its higher processes, if any higher process is alive then it will take over the control, else as of now currently elected process will be the coordinator.
- The new coordinator declares its victory by sending a message to all processes, letting them know, it is the new coordinator.
- All processes set the coordinator ID in coordinator variable and reset election flag to false.

Step3

Immediately after the process with higher number compare to coordinator is up, bully algorithm is run.

Fig:5 shows the steps involved in modified Bully election algorithm.

- Process 4 holds an election
- Process 5 and 6 responds, informing 4 about their presence in the system by OK message.
- Processes 4 informs 6 to become coordinator.
- Process 6 checks with process 7 if it comes back.
- Since no reply from process 7, process 6 wins and broadcasts the Coordinator Message to all the processes.

Advantages and limitations

Improved Bully algorithm is having all advantages like fail-safe mechanism, no parallel election and reduced number of messages over the network.

How long the election initiator should wait to get response from all higher processes. If we keep a timer then the limitation could be the timeout value. Higher timeout will raise performance issue and lower timeout may miss responses from higher processes due to busy network traffic. However fail-safe mechanism will be very helpful in this case.

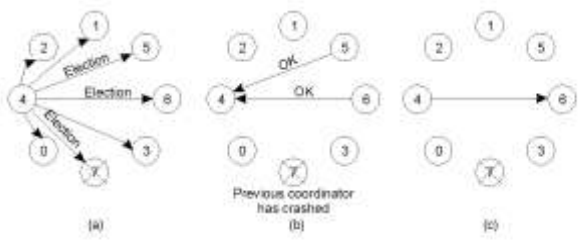

Fig. 5 : Improved Bully Algorithm

Bully Algorithm Procedure

Selecting start node (green) and recognizing coordinator node by changing the color (red). Election message send from 4 to 5 and waits till to receive the Message. Once received the response sends election message to next node viz., 6. Again waits for response. In this way each should respond then only first round of election is complete. But this procedure should continue to be remaining all node.

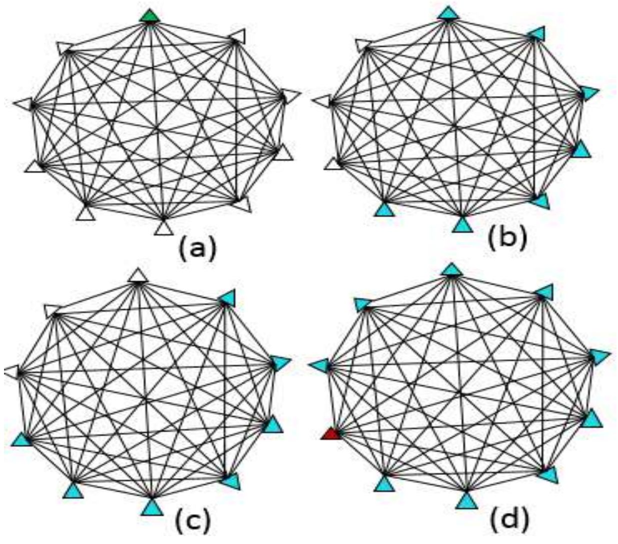

Fig:6 Bully Algorithm Simulation

Then only can decide which node is to be acted as Coordinator. Once the Coordinator identified, then the Coordinator sends message to all nodes as the leader it self. If 10 nodes process started then the total message count will be 51.

Fig:6(a) and Fig:6(b) shows that, process ID 4 identifies the absence of the leader and initiates the election by sending the election message to its higher IDs namely to processes 5, 6, …, 10. All these processes in turn start their own election and concludes the election by the coordinator message where process ID 10 is the new leader. These activities are depicted in Fig:6(c) and Fig:6(d).

In the modified Bully algorithm, a message sent from a node to all remaining nodes and do not wait for their response. All the remaining nodes either respond or not. Non-responding nodes are changed to non-performing mode. Those who are responded are active nodes and decides the Coordinator node among these nodes. Once the Coordinator identified, the Coordinator itself sends message to all nodes as the leader. If the same 10 node cluster procedure starts from node 4, then the total message count will be 25

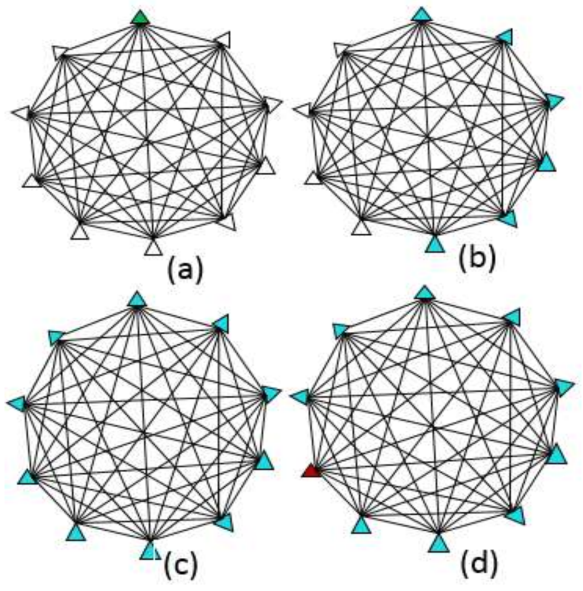

Fig:7 Improved Bully Algorithm

The mockup of Bully algorithm is presented in Fig:7(a) and Fig:7(b) shows that, process ID 4 identifies the absence of the leader and initiates the election by sending the election message to its higher ups namely to processes 5, 6, …, 10. Unlike the Bully algorithm, all these processes reply to the initiator process 4 instead of starting their own election. in Fig:7 (c) and Fig:7(d) shows that process 4 decides the new coordinator (which is 10 in our simulation) and informs process 10 to take over and the election gets concluded by the broadcast of coordinator message where process ID 10 is the new leader.

Message Comparison

Table5 shows the comparison for both algorithms. In this table we represent the message growth following by corresponding number of processes in the distributed network. Table shows that number of messages are increasing drastically in the Bully algorithm compare to the modified Bully algorithm.

Processes	Messages Count	
	Bully Algorithm	Modified Bully Algorithm
5	24	13
10	99	28
15	224	43
20	399	58
25	624	73

TABLE5: MESSAGE COMPARISON OF BULLY AND MODIFIED BULLY ALGORITHMS

Fig:8 shows a comparison graph where both Bully and modified Bully are highlighted in different colors. Graph presents the comparison where number of nodes represented by X- axis and number of messages represented by Y- axis. Graph shows that Bully is having curve shape that describe $O(n^2)$ and modified Bully algorithm is having linear growth described by a straight line with the complexity of O(n).

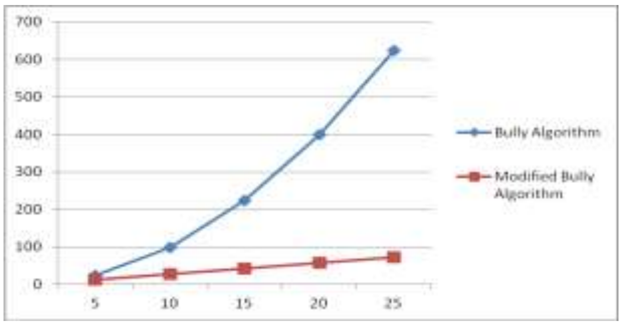

Fig:8 Number of messages used during the election

Simulation Result

Our simulation result shows that modified election algorithm is more efficient as it reduces the number of messages, also avoided any parallel election process. The comparative results are well explained by simulation logs, comparison graph and table.

Analytical Comparison

If only one process detects crashed coordinator

N: The number of processes

P: The priority number of processes that find out the crashed coordinator

Tm: The number of messages passing between processes when the Pth member detects the crashed Coordinator.

In bully modified algorithm the number of massages passing between processes for performing election is obtained from the following formula:

$Tm = 2 * (N - P) + N \rightarrow (1)$

Which has Order O(n). In the worst case that is P = 1 (process with lowest priority number finds out crashed coordinator):

$T1 = 2 * (N - 1) + 1 = 3N - 1 \rightarrow (2)$

Whereas the number of massages passing between processes in the Bully algorithm for performing election is obtained from the following formula:

$Tm = (N - P + 1) (N - P) + N - 1 \rightarrow (3)$

In the worst case that is P = 1 (process with lowest priority number detects crashed coordinator):

$T1 = N2 - 1 \rightarrow (4)$

Which has Order $O(n^2)$. Number of messages in proposed bully algorithm will be equal to $3n - 1$, hence the modified Bully algorithm is better than bully algorithm.

Now assume that the set of processes in S = {P1, P2, P3, ... Pn} from processes find out the crashed coordinator concurrently (P1 is the lowest process).

In Bully algorithm, considering worst case and assuming lowest process start election, then:

- Total number of election message sent to set (S) of n processes ({P1, P2, P3… Pn}) are (n - 1).
- Total response message received by P1 is (n - 1)
- Now P2 will send election message to n – 2 processes.
- Total response message received by P2 is (n - 2).
- Similarly, for P3, P4… and P_n.
- Finally, P_n informing to every process by sending coordinator message is again (n - 1) message.

The number of message passing between processes for performing election is obtained from the following formula:

$Tm = (n - 1) + (n - 2) + (n - 3) + \ldots + (n - n - 3) + (n - n - 2) + (n + n - 1) + (n - 1)$

Simplifying the above formula, we get

$Tm = n (n + 1) / 2 \rightarrow (5)$

which is of O(n2)

In our modified algorithm, considering worst case and assuming lowest process start election, then:

- Total number of election message sent to set (S) of n processes ({P1, P2, P3…Pn}) are (n - 1).
- Total response message received is (n - 1).
- Informing to coordinator and coordinator to check with past coordinator involve two messages
- Finally all the processes i.e. (n-1) are received a message from the coordinator message.

The number of messages passing between processes for performing election is obtained from the following formula:

$Tm = (n - 1) + (n - 1) + 1 + 1 + (n - 1)$, or

$Tm = 3n - 1$ or $3n \rightarrow (6)$

which is of O(n).

III CONCLUSION

In this paper, we discussed the drawbacks of Bully algorithm and then we presented an optimized method for the Bully algorithm called modified bully algorithm. Modified Bully algorithm shows improved performance than the Bully algorithm. The additional advantages of modified Bully algorithm are that it is very simple, having fail-safe mechanism, no parallel election, and reduced number of messages.

Our analytical simulation shows that our algorithm is more efficient rather than the Bully algorithm, in both number of message passing and the number of stages, and when only one process runs the algorithm message passing complexity decreased from O(n^2) to O(n). In this analysis we consider the worst case in modified algorithm. Result of this analysis clearly shows that modified algorithm is better than bully algorithm with fewer message passing in less number of stages.

REFERENCES:

[1] Sung-Hoon-Park,"A Probablistically Correct Election Protocol in Asynchronous Distributed System ", APPT, LNCS 2834, pp.177-183, 2003. 13

[2] Chang Ben Ari, "Principles of Concurrent and Distributed Prog ramming," Pearson Education, 2nd edition, 2006. 10

[3] Rashmi Saini and Ashish Bisht," A Hybrid Algorithm for Load Balancing" *International Journal of Advanced Research in Computer Science and Software Engineering 5(7), July- 2015, pp. 1-6*

[4] Cardellini, Colajanni, and Philip S. Yu, ―*Dynamic Load balancing on web-server systems,* published in IEEE internet Computing, vol. 3, no. 3, pp 28-39, 1999.

[5] H.Garcia-Molina, "Elections in Distributed Computing System," IEEE Transaction Computer, vol.c.310, pp.48-59, 1982. 9

[6] C. Lin, S. Lu, X. Fei et al., "A reference architecture for scientific workflow management systems and the VIEW SOA solution," *IEEE Transactions on Services Computing*, vol. 2, no. 1, pp. 79– 92, 2009.

[7] Daniel A. Menascé, George Mason University *"Trade-offs in Designing Web Clusters"* IEEE Internet Computing1089- 7801/ 02 ©2002 IEEE.

[8] O.K. Tonguz and E. Yanmaz, ―*On the Theory of Dynamic Load Balancing*, in Proc. IEEE Global Telecom. Conf. (GLOBECOM'03), vol. 7, pp. 3626-3630, Dec. 2003.

[9] D. Chitra Devi and V. Rhymend Uthariaraj," Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks", Hindawi Publishing Corporation, The Scientific World Journal, Volume 2016, Article ID 3896065, 14 pages.

[10] G. Le Lan, "Distributed System – Towards a Formal Approach," Information Processing, B. Gilchrist, Ed.Amsterdam, The netherlands: North-Holland, pp.155-160, 1977. 12

[11] Andrew S and Tanenbaum, "Distributed Systems Principles and Paradigms," Beijing: Tsinghua University Press, 2008. 11

[12] Harikesh Singh, Dr. Shishir Kumar *"Dispatcher Based Dynamic Load Balancing on Web Server System,* International Journal of Grid and Distributed Computing, Vol. 4, No. 3, September 2011.