Synchronization Algorithms for Multi-cores and Multiprocessors

Cherish G
M.Tech, Dept. of CSE
RNSIT
Bengaluru, India
e-mail: cherish.gn@gmail.com

Satish Kumar T
Asst. Professor, Dept. of CSE
RNSIT
Bengaluru, India
e-mail: satish.savvy@gmail.com

Abstract— A distributed system is a group of processors that do not allocate memory. As an alternative, each processor has its own local memory, and the processors communicate with one another through communication lines such as local-area or wide-area networks. The processors in a distributed system vary in size and function. Such systems may include small handheld or real-time devices, personal computers, workstations, and large mainframe computer systems. Distributed systems, will have their own set of unique challenges, including synchronizing data and creating sense of conflicts. Effective synchronization algorithms performance depends on runtime factors that are rigid to predict. The designers have protocols to employ the synchronization operation and waiting mechanisms to wait for synchronization delays. In this paper an effort is made to investigate synchronization algorithm that vigorously select waiting mechanisms and protocols in response to runtime factors so as to attain enhanced performance.

Keywords- *Distributed Systems,Election Algorithms, Beowulf Cluster, Performance Evaluation*

*****

I. INTRODUCTION

A distributed system is a collection of autonomous computers that is perceptible to the users of the system as a single computer. This definition has two features. The first feature deals with hardware where the computers are independent. The second feature deals with software where the users think of the system as a distinct computer. Both are crucial.

Synchronization in centralized systems is primarily accomplished through shared memory. Some distributed algorithms need the use of a coordinator. If the coordinator fails, the system can carry on execution by restarting a new replica of the coordinator. It can do so by maintaining a backup coordinator that is ready to assume responsibility if the coordinator fails. Another approach is to choose the new coordinator after the coordinator has failed. The algorithms that govern where a new copy of the coordinator must be resumed are called election algorithms. Two algorithms, the bully algorithm and the ring algorithm, can be used to choose a new leader in case of failures.

In distinct CPU systems, mutual exclusion, critical regions and other synchronization problems are usually interpreted using techniques such as semaphores and monitors. These techniques are not suitable to employ in distributed systems since they consistently rely on the survival of shared memory. For example, two processes that are cooperating by means of a semaphore should be capable of using the semaphore. If they are operating on the same computer, they can distribute the semaphore by storing it in the kernel, and then accomplish system calls to access it. Nevertheless if they are running on various computers, this scheme no longer works and additional techniques are required.

An appeal for highly consistent and synchronous systems is viewed. As an outcome, there has been a reasonable change from centralized systems to distributed systems. There are only some disadvantages for this system. The major one is that the different nodes preserve their individual time by means of local clocks and their values in time may not be same for the dissimilar nodes i.e. there is no global clock inside the system so that variety of actions in the distributed atmosphere can be synchronized. The variety of clocks in the system if set to an ordinary time value at a moment, wander separately owing to inevitable reasons. Therefore some sort of uninterrupted mechanism for synchronization is required so that they can organize and work mutually to accomplish the objectives of the distributed system.

II. EXISTING SYSTEM

Leader election is the method of nominating a single process as the coordinator of some task scattered across several computers. Various distributed algorithms require one process to operate as initiator, sequencer, coordinator or to carry out some special role. We have already seen quite a few examples, such as the coordinator in the centralized mutual exclusion algorithm. Generally it does not subject to which process takes on this control, but any one of the process is allowed to do it.

The paper by Scott D Stoller [1] proposed a substantial effort on self-stabilizing algorithms for leader election which shows that the Bully Algorithm without difficulty can be modified for asynchronous systems with exploiting a failure detector, as an alternative of precise time-outs; this yields a standard way out to leader election in synchronous systems.

Priyanka Gupta and Rajeev G.Vishwakarma [2] focused on comparison of different available algorithms to support their structure, assumptions and the complexity. Here the message complexities of various algorithms from various papers are taken. In this paper the newly proposed Bully algorithm uses several nodes with its unique identification number.

Seema Balhara, Kavita Khanna [3] focuses on the information about the various existing leader election mechanisms which is used for selecting the leader in different problems. The leader election is critical crisis in distributed system as data is distributed amongst different nodes which are geographically separated.

The paper by Vaibhav P. Gajre [4] compares Bully election algorithms in distributed systems by various authors. In the paper, comparison of base and systematic version of bully algorithm to minimize the number of messages when electing the coordinator is analyzed and deals with how a process recovers from a crashed state in distributed systems.

Hetal Katwala, Prof. Sanjay Shah [6] proposed a comparative analysis of the different election algorithms in distributed system and shows different election algorithm with different approach.

The election algorithm proposed by Sandipan Basu [7] is an enhancement of original Bully algorithm proposed by Hector Garcia-Monila proposed algorithm overcomes the overhead of sending too many messages between nodes.

III. PROPOSED SYSTEM

Distributed systems consist of multiple processors that communicate through a network. To control the communication between different nodes and to interchange the data between them, a leader among them is required. In the paper, different election algorithms for selecting the leader in distributed systems is being implemented and also analogizing the performance of each of these election algorithms. These election algorithms are implemented using the message passing interface (MPI 1.4.2).

IV. DESIGN

The system architecture is shown in fig 1. The purpose of the design is to plan the solution to the problem specified by the requirements manuscript. This stage is the first step in moving from problem to the solution domain.

The next layer is the implementation layer. Runtime parameters are chosen and selected benchmark kernels are run. Algorithms are synchronized to find out the optimal values for these parameters. Statistical method is used to find out the parameters which really affect the performance.

The third layer is the communication layer where the messages can be passed through the network. A Beowulf cluster is rigged up consisting of 4 nodes which can be maximized as per the requirement. Open MPI is used for passing messages between the master and the slave nodes.

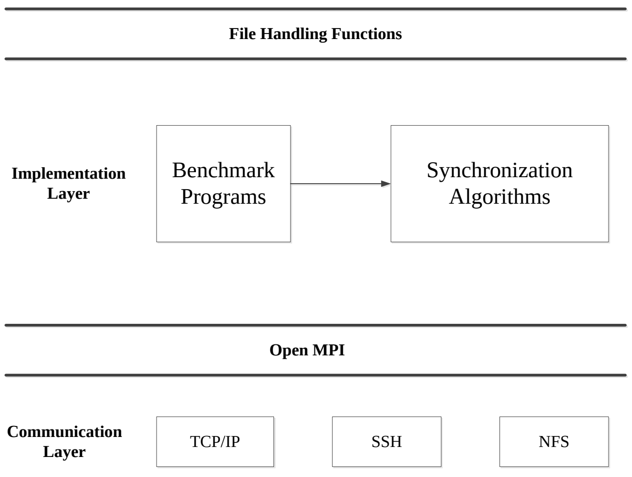

Fig 1. System Architecture

Network File System (NFS) is used to allocate a common folder containing the source code. Open MPI makes use of SSH to communicate inside the nodes. Accordingly open SSH has to be installed and the password authentication has to be detached on all nodes. TCP/IP protocol is used for the communication between the nodes.

A. *ELECTION ALGORITHMS*

Various distributed algorithms require one process to operate as coordinator, initiator, sequencer, or if not to achieve some unique role. In common, it does not subject to which process takes on this control.

If all processes are precisely the same, there is no procedure to choose one of them as unique, without any distinguishing characteristics. Accordingly, we will assume that each process has a unique number, for example its network address. In general, an election algorithm makes an effort to find the process with the highest unique identification number and elect it as coordinator.

Moreover, we also assume that each process knows the process identification number of all other process. The processes do not know which ones are currently active and which ones are currently inactive. The purpose of an election algorithm is to make sure that when an election starts, it concludes with all processes approving on who the new coordinator is to be.

1. Bully algorithm

The bully algorithm is proposed by Garcia-Molina [G82] and works on a completely connected network of processes. It assumes that communication links are fault-free, processes can fail only by stopping, and failures can be detected using timeout. Each process has a unique id, Pi_UID. Once a failure of the current leader is detected, the bully algorithm allows the non-faulty process with largest id eventually to choose itself as the leader. The algorithm uses three different types of messages: *election, reply, and leader*.

Algorithm:

```
Begin
        Process P finds that coordinator is down.
        P initiates leader election by broadcasting
        election message to all processes.
        Each process replies with its unique, Pi_UID
        if Pi_UID > P then
                P waits for the response
        if there is no reply with Pi_UID then
                Promote that host to leader
        if Pi_UID < P then
                P initiates another election
end
```

2. Ring algorithm

An absolutely different approach to attain mutual exclusion in a distributed system (e.g. Ethernet). In ring election algorithm the nodes (the slaves and the master) are ordered in a coherent ring in which they can only communicate with their consistent neighbors. If the master node fails, only the direct neighbors of the master will realize. The slave nodes will then proceed messages around the ring of slaves to notice which one has the highest unique identifier, UID (same as the bully algorithm) and then it becomes the new coordinator.

Algorithm:

```
Begin
        Process P finds that coordinator is down.
        P initiates leader election by broadcasting
        election message to its next process.
        if receiving process is down then
                Sender skips over it and sends election
                message with its UID to all other processes
                along the ring
        else
                sender sends election message with its
                UID to all other processes along the ring
        Repeat the above statements for each process
         along the ring.
        Process P determines the highest UID and
        chooses it as new coordinator
end
```

3. LeLann-Chang-Roberts (LCR) algorithm

This algorithm is for ring networks. Each message in the network goes from one process to another process, i.e. no broadcasting. This means that each process knows exactly about only one other process - its neighbor. This could be viewed as linked list.
Assume clockwise unidirectional ring. One or more P_i's where P_i represents the number of processes along the ring that can obtain the initiative and initiate an election, by forwarding an election message containing their process id to P_i+1. When a P_i unexpectedly or upon receiving a message goes in an election, it chooses itself as a member. If the Pi receiving an election message has a larger process id and is not already a member, then it sends an election message with its own id to P_i+1. If its own process id is smaller, it passes on the message with the id it has received to the next node. If it receives a message with its unique process id then it announces itself as the leader.

Algorithm:

```
Begin
        Send message with identifier = i to other processes
        if identifier j of current process > i then
                Send the message to neighbors with
        identifier i
        else
                Drop message with identifier i
                Send the message with identifier j to
        neighbors
        Continue this process until it receives back a message
        with its identifier
        if a process receives a message with its id then
                Process = leader
        else
                return null
end
```

4. Hirschberg-Sinclair (HS) algorithm

This algorithm (1980) requires $C = O(n \log n)$ message complexity for finding the largest (or smallest) of a set of *n* uniquely numbered processors ordered in a circle. However, we now allow bidirectional communication. Instead of sending ith identity all the way around the ring, a process p sends it in both directions to travel some distance 2^k away, where k is incremented in phases. Traveling identities are dropped and passed as in Chang-Roberts. Only if the identity comes back from both directions, p proceeds to phase k+1. Otherwise, it only relays messages between its neighbors.

Algorithm:

To initiate an election (phase 0):
Send (ELECTION (my_id, 0, 0)) to left and right;

Upon receiving a message ELECTION (j; k; d) from left (right):

```
if ((j >my_id) ^ (d _ 2k )) then
        send (ELECTION(j; k; d + 1)) to right (left);
if ((j >my_id) ^ (d = 2k )) then
        send (REPLY(j; k)) to left (right);
if (my_id = j) then announce itself as leader;
```

Upon receiving a message REPLY (j; k) from left (right):

```
if (my_id!= j) then
        send (REPLY(j; k) to right (left);
else
```

```
if (already received REPLY(j; k))
        send (ELECTION (j; k + 1; 1)) to left and right;
```

B. EXPERIMENTAL SETUP

Table 1 show the system configuration used in setting up the Beowulf cluster. All experiments undertake in this research uses a set of 1 to 50 nodes respectively.

Table 1: Target Architecture

# of nodes	1-50
# of cores per node	4
Memory per node	4GB
Open MPI version	1.4.2
Make	Dell 390 Optiplex
Processor	Intel core i5
OS	Ubuntu 12.04LTS
Clock Frequency:	2.5GHz

The beowulf cluster is configured as shown in figure 2.

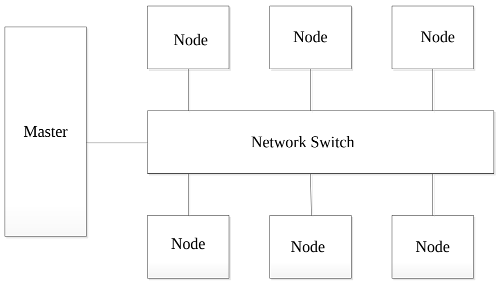

Fig. 2 Beowulf Cluster

V. RESULTS

The performance of Bully algorithm, Ring algorithm, LeLann-Chang-Roberts algorithm, Hirschberg-Sinclair algorithm on the proposed architecture is represented using graphs in figure 3 and figure 4.

Algorithm	No. of Procs	crash process	Initiator process	New Coordi -nator	Time
Bully algorithm	7	7	3	6	34.0039
	10	10	4	9	42.0047
	50	50	23	49	126.011
	100	100	21	99	334.027
	200	200	56	199	594.047
Ring algorithm	7	7	3	6	22.0016
	10	10	4	9	26.0020
	50	50	23	49	68.0053
	100	100	21	99	172.013
	200	200	56	199	302.022

Table 2: Performance of Bully vs. Ring algorithm

Table 2 shows the timing needed for Bully and Ring leader election algorithms to choose a leader. Similarly in table 3 LCR vs. HS is shown. The table shows the number of processors where at any point in time a process might be crashed and an initiator is chosen for electing a leader.

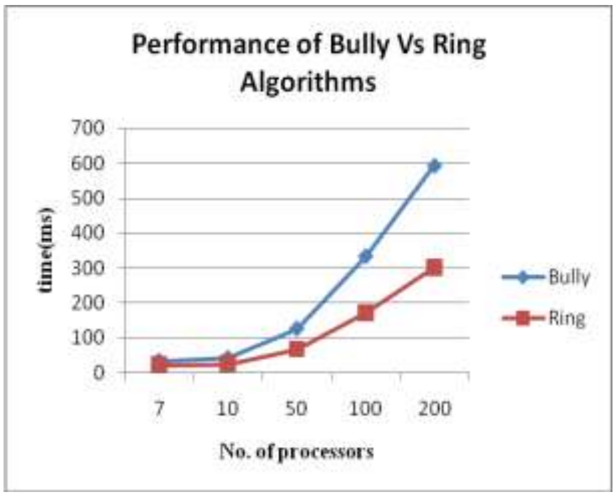

Fig 3: Comparison of Bully vs. Ring algorithm

In figure 3, a comparison between Bully vs. Ring algorithm is shown. Similarly in figure 4, the comparison of LCR vs. HS is shown. The graph shows Ring algorithm out performing Bully and LCR proving better than HS algorithms.

Algorithm	No of procs	Initiator process	New coordinator	Time
LCR algorithm	10	Random number	10	0.00022
	50	Random number	50	0.00046
	100	Random number	100	0.00090
	200	Random number	200	0.00132
Hirschberg-Sinclair algorithm	10	Random number	10	0.00104
	50	Random number	50	0.00146
	100	Random number	100	0.00190
	200	Random number	200	0.00305

Table 3: Performance of LCR vs. HS algorithm

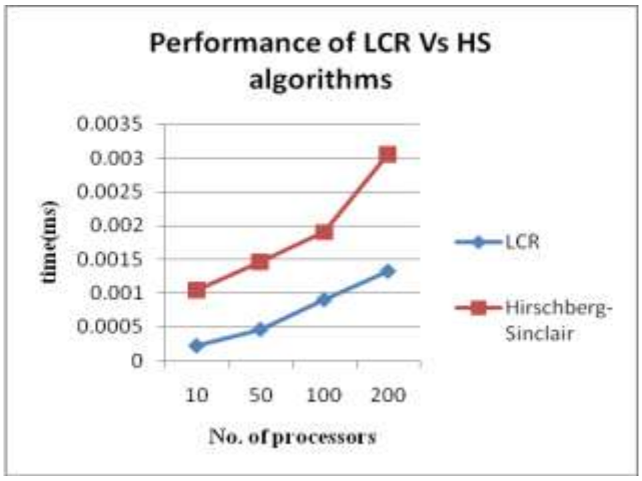

Fig 4: Comparison of LCR vs. HS algorithm

VI. CONCLUSION

Basically the paper tries to enforce the implementation of different kinds of distributed algorithms from which we can achieve synchronization across network. The algorithms that are implemented partly takes help of the underlying technologies of OpenMPI way of working in order to exploit the optimization of the code for implementing the set of algorithms chose. Therefore this paper aims at simulating the algorithms which are non-deterministic for the real time implementation of it based on the kind of application and also the kind of synchronization that needs to achieve. Hence we can conclude that this paper aims in providing the algorithms not only as a tool for implementation but also gives a proper visualization graphically for understanding the working of the implemented algorithms as a whole package and can even test the algorithms with a click of mouse across different systems assuming there's a cluster or else runs on the host system itself.

REFERENCES

[1] Abraham Silberschatz, "Communication and Synchronization in Distributed Systems", IEEE Transactions on software engineering, Vol. se-5, No. 6, November 1979.

[2] Scott D. Stoller, "Leader election in Distributed systems with crash failures", Dept. of Computer Science, Indiana university, Bloomington, IN 47405, USA, 17th July 1997.

[3] Priyanka Gupta, Rajeev G. Vishwakarma, "Comparison of Various Election algorithms in Distributed System", *International Journal of Computer Applications (0975 – 8887) Volume 53– No.12, September 2012.*

[4] Seema Balhara, Kavita Khanna, "Leader Election Algorithms in Distributed System", International Journal of Computer Science and Mobile Computing, IJCSMC, Vol. 3, Issue. 6, June 2014**.**

[5] Vaibhav P. Gajre, "Comparison of Bully Election Algorithms in Distributed System", International Journal of Scientific and Research Publications, Volume 3, Issue 9, September 2013.

[6] Priyanka D. Bhute, "A New Approach for Electing a Coordinator in Anonymous System", International Journal of Computer Science and Information Technologies, Vol. 5 (3), 2014.

[7] Hetal Katwala1, Prof. Sanjay Shah, "Study on Election Algorithm in Distributed System", *IOSR Journal of Computer Engineering (IOSRJCE), Volume 7, Issue 6 (Nov. - Dec. 2012).*

[8] H. Garcia Molina, "Elections in a Distributed Computing System." *IEEE Trans*. Comp, 1982, vol.31, no. 1, pp.48-59.

[9] N. Fredrickson and N. Lynch, "Electing a Leader in a Synchronous Ring."*J.ACM*, 1987, vol.34, no.1, pp.98-115.

[10] S. Park, Y. Kim and J.S. Hwang, "An Efficient Algorithm for Leader-Election in Synchronous Distributed Systems," IEEE TENCON, 1999.

[11] Sandipan Basu, "An Efficient Approach of Election Algorithm in Distributed Systems", Indian Journal of Computer Science and Engineering (IJCSE), vol. 2, No. 1, pp. 16-21.March 2011.

[12] Le Lann, G., "Distributed Systems – Towards a Formal Approach", in Information Processing 77, B. Gilchrist, Ed.Amsterdam, The Netherlands: North-Holland, pp. 155-160, 1977.

[13] M. S. Kordafshari, M. Gholipour, M. Jahanshahi, A.T. Haghighat, "Modified Bully Election Algorithm In Distributed System", *Wseas Conferences, Cancun, Mexico, May 11-14, 2005.*

[14] Sinha P.K, Distributed Operating Systems Concepts and Design, Prentice-Hall of India private Limited, 2008.

[15] A.Arghavani, E.Ahmadi, A.T.Haghighat, "Improved Bully Election Algorithm in Distributed Systems", *5th International Conference on IT & Multimedia at UNITEN (ICIMU 2011) Malaysia, 2011*.

[16] Sung-Hoon Park, "A Stable Election Protocol based on an Unreliable Failure Detector in Distributed Systems", Proceedings of IEEE Eighth International Conference on Information Technology: New Generations, pp. 976-984, 2011.

[17] Muhammad Mahbubur Rahman, Afroza Nahar, "Modified Bully Algorithm using ElectionCommission", *MASAUM Journal of Computing (MJC), Vol.1No.3, pp.439-446, October 2009, ISSN 2076-0833*.

[18] Andrew S and Tanenbaum, "Distributed Systems Principles and Paradigms", Beijing: Tsinghua University Press, pp.190–192, 2008.

[19] M.Gholipur, M.S.Kordafshri, M.Jahanshani, A.M.Rahmani," Modified Bully Election Algorithm in Distributed Systems" 2005.

[20] Operating System Concepts (16 th Chapter Distibuted Synchronization) GergGagne, Abraham Silberschatz, Peter B. Galvin.

[21] Tai Yun Kim,"A Leader Election Algorithm in a Distributed Computing System", Department of Computer Science, Korea University-1, Ga, Anam: Dong, Seoul, pp. 136-701.

[22] Chang-Young Kim, Sung-HoonBauk, "The Election Protocol for Reconfigurable Distributed Systems", ICWN, pp. 295-301, 2006.

[23] Prof. Sanjay Shah & HetaJasmin Jhaveri, "A Dynamic Election Strategy in Distributed System", International Journal of Engineering Research and Technology (IJERT), Vol. 1 Issue 3, May – 2012.

[24] Md. Golam Murshed and Alastair R. Allen, "Enhanced Bully Algorithm for Leader Node Election in Synchronous DistributedSystems", Computers 2012, Vol. 1, pp. 3-23.

[25] Masafumi Yamashita and Tsunehiko Kameda. "Leader Election Problem on Networks in which Processor Identity Numbers Are NotDistinct", *IEEE Transactions on parallel and distributed systems vol 10, NO 9, September 1999.*

[26] Jeremie Chalopin, Emmanuel Godard, Yves Metivier, "Election in partially anonymous networks with arbitrary knowledge in message passing systems", Springer-Verlag 2012.

[27] Chang , E. and Roberts, R. An improved algorithm for decentralized extrema-finding in circular configurations of processes. Comm. ACM, 22, 5 (May 1979), 281-283.

[28] Hirschberg, D.S. and Sinclair, J.B. Decentralized extrema-finding in circular configurations of processors. Comm. ACM, 23, 1 (Nov, 1980), 627-628.

[29] Pawan Kumar Thakur, Ram Kumar, Ruhi Ali and Rajendrakumar Malviya ,"A New Approach of Bully Election Algorithm for Distributed Computing", *Int. J. of Electrical, Electronics and Computer Engineering (IJEECE) Vol 1(1): 72-79,2011.*

[30] Sukumar Ghosh, *Distributed systems: an algorithmic approach*. Vol. 13. Chapman & Hall/CRC, 2006.