ISSN:2321-1156
International Journal of Innovative Research in Technology & Science(IJIRTS)

Synchronization Algorithms in Distributed File System

J.RamyaRani, SR.Drishya, RejiniSusanVarghese, RAK SaravanaGuru.
SCSE Dept, VIT University, Vellore, Tamil Nadu, India.
jrani51@gmail.com,sr.drishya@gmail.com,rejini21@gmail.com,meetsaro@gmail.com.

Abstract

An emblematic network file system is centralized and is dependency on a dedicated file server. This may lead to several routine issues and consistency disorder. This design suggests a system which is scalable and distributed which can be called as "server-less distributed storage system" .Self-synchronization in distributed file-systems is the domain of study. An efficient algorithm called γ synchronization algorithm superior in terms of time and communication complexity is proposed in this survey

Keywords Server, File-volume, Synchronizer

Introduction

Network File system is the present, typical remote file-access-protocol. Network File system does not maintain caching. It also needs recurrent access to the server. Apart from all these, the main disadvantage is that it does not offer self-synchronizing techniques. One of the most widespread distributed file systems in the case of Andrew file system. The advantages includes which support of each file-volume here it is a single copy is provides for read and write whilst all others are "read only replication" [2].

Asynchronous algorithms are significantly lower in terms of complexity to their respective synchronous algorithm, hence it is significantly difficult to design and analyze them, and a common simulation technique that helps user to carve an algorithm that gives an intuition that it is running in a synchronous network. A file volume is represented using a spanning tree. These are used to construction and maintenance of this spanning tress is carried out using synchronizer algorithms. The term self-stabilization refers to the recovery from any many synchronizer algorithms. Furthermore such type of systems bears the occurrence of frequent and unexpected faults and delivers the system from the unpredictable data corruption and loss.

Problem Definition

This paper deals with distributed algorithms in two networks models namely synchronous network model and synchronous network model. The asynchronous network is also called as "point to point communication network". It is well explained with the help of the network and E represents the bi-directional non-interfering communication channels existing between them. The processors have no shared memory and each node has a discrete characteristic. The task of each node is to process messages obtained from the neigh-bors, to perform local computations and to send messages to the neigh bors. These actions are carried in very little less time. The messages which carry only a specified quantity of information are supposed to have constant length, the messages reaches within finite but unpredictable time.

In the case of a synchronous network, messages are sent only at integer times of a global clock. Each processor has access to this global clock. The maximum number of messages that can be sent at a certain pulse and over a communication link is limited to one. The performance of the algorithms is measured using the time and communication complexities. The communication complexity is the total number of messages sent during the execution of pulses transferred from its initial time until its extinction. A characteristic fact in communication networks is the trade-off that occurs between the communication and time [3].

The proposed system design consists of a synchronizer that is efficient enough to run any synchronous algorithm in any asynchronous network. A fresh pulse is produced at a node after it has received all the messages of the synchronous algorithm that has been sent to those nodes by its neigh bors at the preceding pulses. The characteristic makes sure that the network behaves like a synchronous network when the synchronous network when a particular execution of the particular synchronous algorithm is taken into consideration. The main challenge in the design of a synchronizer is that a node does not know which the messages were sent to it by its own neigh bors and also link delays are uncertain in these cases. Hence it is not possible to wait for messages for synchronizations purposes. The total complexity of the resulting algorithm and time requirements added by a synchronizer v per each pulse of the synchronous algorithm by C (v) and T (v) respectively, synchronizers need an initialization phase which is taken into consideration only when the algorithm is executed only once. Let C_{init} (v), T_{init} (v) be the complexities of the initialization phase of the synchronizer v. The complexities of combination of 'S' and asynchronous algorithm A which is a combination of 'S' and synchronizer 'v' are $C_A=C_s+T_s.C$ (v) + T_A= Ts- T (v) +T_{init} (v) where C_A, T_A, Cs,

Ts are the communication and time complexities of algorithms A and S respectively. A synchronizer v is said to be efficient if only if all the parameters C (v), T (v), C (v), T_{init} (v) are small enough. The former two parameters are really essential as they denote the overhead per pulse

Existing Approach

A node is regarded is safe if each message of synchronous algorithm sent by the node for a particular pulse has been reached the destination without any hindrance. The messages need to be sent only to its neighbor. After the generation of a particular pulse, each node becomes safe with respect to a certain pulse. If it is required that an acknowledgment is to be returned whenever a message of the algorithm is obtained from the neighbor, then a node is regarded as safe after all the messages are acknowledge. The asymptotic complexity is not maximized by acknowledgements alone. So a particular node learns that it is secure in constant time after it is generated a new pulse. If this situation all the neigh bors of that particular node are also considered as safe with respect to a certain pulse. Hence only thing that should be taken into consideration is to determine how to transmit the messages to each node with lower communication and time overhead. The synchronizers α and β are based on the above mentioned concepts and are used to achieve synchronization and asynchronous network [4].

Synchronizer α

Synchronizer α also called as α synchronization algorithm makes use of the acknowledgement mechanism mentioned in the previous section. Here each node eventually knows that it is safe and transmits this message directly to all its neighboring nodes. A node pulse it is produced whenever a node comes to know that each of its neigh bors are safe.

The communication complexity of synchronizer α is

C (α) = O (|E|) where E= V^2. The time complexity is found to be T (α) = O (|1|). This is because only one message is sent in each direction across each link and only the neighboring nodes are involved in the communication process.

Synchronizer β

The synchronizer β also called as synchronizer β algorithm, requires an initialization phase for consistency maintenance. A leader is selected for each network and a spanning tree is built which is rooted at the leader. Its operation can be explained as follows. After the execution of a certain pulse, the leader understands that the nodes are said to be safe state. This forces the leader to send a message to all other nodes in the network. On the receipt of this safe message the nodes will generate a new pulse. A procedure called converge cast mechanism which starts at the leaves and finishes at the root is to calculate the time. Here a node conveys a safe message to its father when it is understands that all the descendants and the node itself are in safe state.

The communication channel of the synchronizer β is C (β) = O (|V|). The time complexity of synchronizer β is T (β) = O (|V|). This is because of the operations of the span tree. The time of the proportional height of the tree, this can be V-1 in the worst case.

Proposed Statement

This paper suggests an algorithm called synchronization γ algorithm which can be used to maintain the reliability in a distributed asynchronous file system. The synchronizer γ is a combination of α synchronizer and β synchronizer. Both the synchronizers α and β are discussed in the previous section. The synchronizer α is efficient in terms of time but ineffective in terms of communication. On the contrary the synchronizer β is superior in terms of communication but useless in terms of time. The synchronizer γ on the other hand is proficient with respect to both communication and time.

Synchronizer γ

The synchronizer γ which is combination of both the synchronizer α and synchronizer β consists of two phases. The initialization phase the network is divided in to clusters. Any spanning forest of the communication graph defined as (V, E) of the network determines the clusters. An intra- cluster tree is a cluster of nodes defined by each tree in the forest. A communication link is selected between any neighboring clusters. A leader is also elected in each cluster. This leader will manage the process of the clusters with respect to the intra-cluster tree. A cluster is proved to be safe if all the nodes in the cluster are found to be safe [5].

The synchronizer γ is performed in to two phases. The synchronizer β is applied to each cluster intra- cluster trees independently in the first phase of the algorithm. The leader of the cluster sends a message all the other nodes in the cluster as well as to all leaders of neigh boring clusters whenever it learns that its own cluster is in safe state. In the second stage of the synchronizer γ all the individual's clusters stay idle for all the neigh boring clusters to be safe and then they will generate the next pulse as soon as they are notified that those clusters are in the safe state. The second stage actually applies synchronizer α among the clusters.

A though description of the synchronizer γ synchronization algorithm is given in the following section. The leader of a cluster will broadcast a PULSE a message along with the spanning tree and this message forces the nodes to enter the fresh pulse. As soon as a node enters the first stage of the synchronizer γ, SAFE message are converge cast long each intra- clusters tree as in the case of synchronizer β. This procedure is initiated at the leaves and proceeds by sending SAFE messages to father nodes once they recognize that they are safe. For non-leaf non cluster nodes once they find

out that their own cluster is safe they are broadcast a CLUSTER-SAFE message to all of their adjoining clusters. The adjoining nodes in receipt of this message advance it to their descendents in the cluster along all selected communication links.

The second phase of this algorithm is even more complex. Here the time required for the adjoining nodes to be safe is determined. This is acquired by a typical converge cast process in which a READY message is sent by a node to its father once all the adjoining clusters or any of its descendants enter the safe state. The node in receipt of the READY message from its descendants and CLUSTER-SAFE messages from its father along all preferred completer procedure is then repeated and the leader broadcasts the PULSE message to all the nodes in its own cluster so that they can enter into next pulse.

Let 'E_p','H_p' be the set consisting of tree links and chosen communication links in a particular partition- P and the maximum height of a tree in a forest of partition 'P' respectively. It can be calculated that a maximum of four messages are transmitted across each communication link of E_p. So communication complexity, C (γ) = O ($|E_p|$). Also it can be confirmed that O (H_p) time is required for each cluster to prove that it is safe and an extra time of O (H_p) time is required for each cluster to prove are in safe state. So the time complexity, T (γ) = O (H_p). Thus it can be proved that communication and time complexities of synchronizer β. It is illustrated in table 1. So the synchronizer γ can effectively be implemented in an asynchronous distributed file system to achieve consistency.

Table 1

Sl NO	Synchronizer	Time Complexity	Communication Complexity
1	α	O(|1|)	O(|E|) where $E=V^2$
2	β	O(|V|)	O(|V|)
3	γ	O(H_p)	O($\|E_P\|$)

Future Work

From the study of the communication and times complexities of the synchronizer γ it is evident that these complexity parameters will be smaller if only if we are able to find a partition 'P' for which parameter 'E_p' and 'H_p' are lesser. This characteristic can be established by designing an algorithm where each cluster is selected as a 'maximal subset of nodes whose diameter does not go beyond the logarithm of its cardinality '.

The proposed approach enables us to simulate a method to implemented virtual technologies for very large scale distributed systems such as grid systems [1]. We have designed the system using a bounded degree spanning tree and a stabilization algorithm. This method can significantly be used to analyze other topologies for large scale systems. This method also aids in the increment of stabilization time and also helps in the implementation of conflicts resolution algorithms [6].

Conclusions

Self-synchronization of the distributed file systems can to be obtained using synchronization algorithms. This paper proposes an efficient and new simulation methodology called synchronizer γ. The synchronizer γ has been proved to be a resourceful technique for designing distributed algorithms in any a synchronization network. The algorithm also enlightens the way for the implementation of many low complexity algorithms.

References

[1] "Jian Yin, Lorenzo Alvisi, Mike Dahlin, and alvin Lin, Using Leases to Support Server-Driven Consistency in Large-Scale Systems Proceedings of the 18th International Conference on Distributed Computing System, May 1998".

[2]" S. Dolev, Self-stabilization, The MIT press, March 2000".

[3] "Jian Yin, Lorenzo Alvisi, Michael D Dahlin, and alvin Lin, Hierarchical ache consistency in a WAN, Proceedings of the 1999 USENIX Symposium on Internet Technologies and Systems (USITS99), October 1999".

[4] "ARJOMANDI, E., FISHER, M .J. AND LYNCH, N. A. A difference in efficiency between synchronous and asynchronous system". J. ACM 30, 3 (July 1983), 449-456".

[5] "AWERBUCH, B. Applications of the network synchronization for distributed BFS and Max-Flow algorithms."

[6] "Ko, Bong-Jun; Rubenstein, Daniel Distributed self-stabilizing placement of replicated resources in emerging networks"