Local distributed mutual exclusion algorithm

ZHANG Zhong-hao[1]

[1]Department of computer, Civil Aviation Flight University of China, Guanghan Sichuan 618307, China

Abstract

A distributed mutual exclusion algorithm has been proposed for large-scale distributed systems in the Internet environment. The algorithm combines the advantages of centralized algorithm and distributed algorithm, and taking into account the system fault tolerance, while message complexity is decreased by orders of magnitude. The algorithm is completely mutually exclusive, and no deadlock. It has a high use value in the rapid development of the cloud computing today.

Keywords: distributed system; mutual exclusion algorithm; message complexity

1. Introduction

Since the advent of cloud computing, its attention has been high, now it has become the hottest topic in the IT sector. What is cloud computing?

Internally it was defined as: Cloud computing is a commercial computational model. It distributes tasks across a large number of computers that form a resource pool; the application enables on-demand access to computing power, storage space, and information services. [1]

Thus, cloud computing is just a service model; its core is still distributed technologies, for the development of cloud computing, we must continue to develop distributed technologies. Distributed mutual exclusion is one of the key problems of distributed systems design. Due to the lack of shared storage and common physical clocks, and uncertain message delay, the mutual exclusion problem in a distributed system becomes very complex.

In recent years, as Maekawa algorithm [2] generalization, the distributed mutual exclusion algorithm based on the quorum has received extensive concern. A number of unique algorithms [3-4] have been proposed to build a quorum to reduce messaging complexity or improving algorithm resilience in node breakdown or communications failure.

But these algorithms are still applicable only to small distributed system, usually confined to a LAN. Distributed computing in the Internet age needs to take into account the characteristics of the Internet. In this paper, the following algorithm - local distributed mutual exclusion algorithm is put forward in fully consider the characteristics of the Internet node communication.

2. System model

Suppose that nodes is N in a distributed system S. All nodes can be numbered as S0, S1 ... SN-1.

In this system there is no shared storage and common physical clock, nodes for asynchronous communication is by message passing. Messaging using the FIFO principle, nodes can only take up critical resources in priority order, to enter the critical region.

No matter the nodes of the distributed system is distributed in several concen-

trated area or random distribution, all the nodes can always be divided into M (M> = 1) according to certain rules clusters, each cluster with K (K<=N/M) computer is designated as the Coordinator. Coordinator R has all the critical resource information of the system, but each coordinator only controls his own resources of the cluster.

As shown in the following figure:

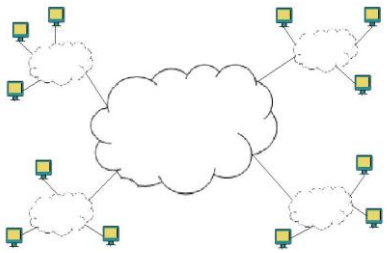

Fig. 1: Model diagram

3. Mutual exclusion algorithm

3.1. The main problem

The basic idea of the algorithm is as follows: In a cluster (Set A) , the node S for accessing critical resources sends a request message to all coordinator R in A, When all coordinator in the cluster agree the application nodes request, the node obtains critical resource. Divided into two cases, when the critical resource in this cluster, handled directly by the coordinator from the cluster; When the critical resource is not in the cluster, the coordinator of A sends a request to the coordinator in the cluster (as B) which has the critical resource, the cluster A was treated as a virtual controlled node.

Therefore, in the former case, the node S sends a request message to Ka coordinator for each critical section access, receiving Ka reply message. When the end of the access to the critical section. Node S needs to send a release message to Ka coordinators. In this case 3*Ka News needs to pass. In the latter case , one coordinator in the A as the controlled node do the same operation to Kb coordination in the cluster B which has the resources , this time 2*Kb message is sent; In after obtaining consent, need to forward this message to other coordinator in A, this time Ka-1 message is sent; Then the Ka coordinators give consent message to the application node; When critical access end, application nodes need to send Ka message to the cluster coordinator, then one coordinator in cluster A sends a to cluster B. Ka+2Kb+Ka-1+Ka+Ka+Kb=4Ka+3Kb-1 messages are sent in this case. So the message complexity of the proposed algorithm can achieve O (3K)-O (7K) in this paper.

In the former case, because there is more than one coordinator, after the expiration of one or more than one coordinator, in order to ensure the redundancy of the system some new coordinators will generate. In the latter case, the cluster that sent the request is treated as a virtual node, so the event will not fail even if coordinator is failure which sent the request message. Thus the system has better fault tolerance.

3.2. Control messages and data structures

Each request priority is determined by Lamport logical timestamp [5]. The smaller the sequence number of the time stamp of the request is, the higher the priority of the request is. IF requests have the same sequence number, the node ID number is smaller whose requests have a higher priority level. And a larger sequence number is assigned to the request than any that have been received, sent or observed.

This algorithm requires three types of messages; each message has the following format.

Request: The request(i,j,reqi) message is from the node Si to the coordinator Sj,

it indicates that Si asks Sj agree it enters the critical section, the request has a time stamp reqi.

Reply: the Reply(j,i) message is From the coordinator Sj to the node Si, it indicates that the node Sj allows node Si to enter the critical section.

Release: The Release(i,j,reqi) message is from the node Si to the coordinator Sj, it indicates that the node Si has released the critical section.

Each control node needs to maintain the following data structure.

req_q: it is a queue which is used to receive requests queued by priority from highest to lowest. Each of the structure's entrance shaped like (Sn,i)，Where Sn is the order number, i is the number of the node requesting critical section. The request has the highest priority in the team first.

Replied: replied is a vector having K elements. K is the size of control nodes. When a new request is sent each time the vector is initialized to 0，and replied[j] is set to 1 when the node Sj received reply(j,i) message.

Mutual Exclusion Algorithm

In order to enter the critical section, the node Si needs to get all the control node approval. If node Si has the approval of all coordinator, it can enter the critical section. Otherwise, it needs to wait for those coordinator who rejects its request allows its request. When the coordinator receives a request of the node Si, the coordinator inserts the request to the tail of the queue req_q. The release message is sent to all coordinators when the node Si completes critical areas access, and then the algorithm continues to run.

According to the idea of the algorithm, the algorithm is described as follows.

A: Request the Critical Section

A1. *／*for a site S_i wishes to enter CS * ／*

S_i sends Request(i,j,req_i) to every site $S_j \in R$;

$replied_i[]=0$;

A2. *Action when S_j receives a Request(i,j,req_i);*

if(req_q==null) {

S_j send reply(j, i)message to S_i;

} else {

insert Request(i,j,req_i) in req_q_j;}

B: executing the Critical Section

A site S_i can access the CS only when receive reply of all control sections.

C: releasing the Critical Section

C1. *Action when S_i exits the CS;*

$replied_i[] =0$;

send release(i，j) to $S_j \in R$;

C2. *Action when S_j receives a release(i，j);*

if(req_q!=null) {

S_j send reply(j, get(req_q_j)) message to get(req_q_j);}

D: Site fault-tolerant operation

if(all quorums of the site or all control sites in system are non-operational){

send reconstruct to every site $S_i \in S$;

reconstruct the quorum of S_i;

} else {

If(part control sites are non—operational){

Invites other sites to become control sites;

Send new controls sites message to other sites;}

If(sites which have CS are non—operational){

Control sites send reply to the next site;}}

4. Correctness proof

Distributed mutual exclusion algorithm is valid, need to look at whether it has full mutually exclusive, to avoid deadlocks and deadlock situation does not occur.

Because of all requests in the queue are according to the priority order, each time the critical region is always gotten by the first node in the queue, so no two nodes simultaneously access the critical section. The algorithm can be full mutually exclusive.

Assume that algorithms may be a deadlock, that is, in a series of nodes, because they are waiting for one or more reply, resulting in a loop waiting so that no one node can enter the critical section. In the wait loop, there must be a node Si, its request has the highest priority, and the other waiting nodes priority is lower than it. In the queue, the lower priority nodes are always ranked higher priority nodes behind, they cannot get a reply before the Si, this algorithm does not deadlock.

5. Performance Analysis

Currently distributed mutual exclusion algorithms generally use Maekawa algorithm, the message complexity of the algorithm is O($\sqrt{N}$).In this algorithm message complexity depends on the number of control nodes in each cluster, the message complexity is O(3K)-O(7K)。And in a large distributed system, the k value has nothing to do with the entire system, it depends on the stability of each node in the cluster, the nodes are more stable, k is smaller, else larger. The probability of error in a single node is small, so k is usually a single digit number, does not increase with the increase of the system. Therefore this algorithm message complexity is smaller than fully distributed algorithm one or more orders of magnitude.

The synchronization delay is still 2T, this algorithm does not improve.

In this algorithm, right is determined by control nodes of the node to access a critical section, and the coordinator has all of the information of the system, and the coordinator itself is controlled node, so even if only one right node in the system does not affect the algorithm running. Therefore, the maximum node fault tolerance of the proposed algorithm is N-1.

6. Conclusion

Comprehensive centralized algorithm message complexity is low, and high fault tolerance of distributed algorithms, the paper proposes a distributed mutual exclusion algorithm has higher fault tolerance and smaller communication cost. The algorithm in synchronization delay is not improved, but the message complexity of the algorithm can be reduced by several orders of magnitude, which has an extremely important value to establish large-scale distributed systems based on the Internet. At the same time, the algorithm increase node fault tolerance to N-1. These results show that the algorithm significantly improves the efficiency of distributed mutual exclusion algorithm, but also improve the efficiency of distributed systems, and has a high value in use.

7. References:

[1] Liu Peng. Cloud computing (second edition)[M] Beijing: Publishing House of electronics industry,2011.5

[2] MAEKAWA M. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized systems[J]. ACM Trans Computer Systems，1985，3(2):145-159.

[3] HARADA T, YAMASHRIA M. Transversal merge operation: a non-dominated coterie construction method for distributed mutual exclusion[J]. Parallel and Distributed Systems, IEEE transactions on, 2005, 2(2): 183-192.

[4] LI Mei-an, LIU Xin-song, WANG Zheng. High performance distributed mutual exclusion algorithm based 0n quorum and dynamic token[J], Journal on Communications, 2006(4): 124-130

[5] LAMPORT L. Time clocks and ordering of events in distributed systems[J]. Comm ACM, 1978, 2l(7): 558-565.