ISSN : 0974 - 7435 Volume 10 Issue 23

2014

BioTechnology

An Indian Journal

FULL PAPER

BTAIJ, 10(23), 2014 [14363-14369]

Quorum generation algorithm based on distance repeat times limitation

Li Meian*, Wang Fang, Wang Wei
College of Computer and Information Engineering, Inner Mongolia Agriculture University, Huhehaote, (CHINA)
E-mail: limeian1973@126.com

ABSTRACT

A new symmetric quorum generation algorithm has been proposed in this paper to solve the problem that how to generate a symmetric quorum with the shortest length but has a smaller time complexity so that it can generate a symmetric quorum when the distributed system has a larger scale. The concept of permission total repeat expression times proposed firstly in this paper to limit the attempt times when a node inserted in the quorum. Selecting the nodes in the backup vector to insert the quorum can reduce the attempt times again. Comparing the attempt times with other quorum generation algorithms, the algorithm proposed in this paper will reduce the time complexity 10% at least and the quorum length is still the shortest as Maekawa.

KEYWORDS

Quorum generation; Algorithm; Repeat times; Limitation.

INTRODUCTION AND RELATIVE WORKS

Distributed computing is a useful computing model. It's the basis of many modern computing models such as grid computing and cloud computing. Distributed mutual exclusion is the most important method to resource access control in distributed computing system. So researches on distributed mutual exclusion became more and more important but more and more difficult in last forty years. Lamport[1] proposed the concept of timestamp, and proved that receiver can contain the messages series consistence by timestamp. On the basis of timestamp, Lamport proposed the first symmetric distributed mutual exclusion algorithm based on message pass. It sends request message to all nodes in the distributed system and needs request, reply and release message each time when a distributed mutual exclusion has been implemented. So the message complexity was $O(3N)$. R&A[2] proposed another algorithm to reduce the message complexity. It combined rely and release message as one reply message, and reduced the message complexity to $O(2N)$. From these algorithms presented above, the message complexities are proportional to N and N is the scale of the distributed system. Makawa[3]proposed the concept quorum and four conditions of symmetric quorum. On these bases, Malawa proposed a distributed mutual exclusion algorithm with the message complexity of $O(3n)$ and n is the quorum length. Makawa proved that the lowest limitation of quorum length is $\sqrt{N}$. So there is $n \geq \sqrt{N}$. Makawa's algorithm implemented the symmetric mutual exclusion through sending messages to part nodes in the distributed system. So its message complexity became $O(3n)$ from $O(2N)$ and translated the distributed mutual exclusion algorithm design to the quorum generation algorithm design.

How to design a quorum generation algorithm to ensure the quorum is symmetric, the quorum length is the shortest, the time complexity of quorum generation is the smallest is the most important things to researches on distributed mutual exclusion algorithm design now. Because the quorum length is contradict to the time complexity of quorum generation algorithm, shorter quorum length maybe need higher time complexity.

WK[4] proposed the concept of cyclic quorum and proved that cyclic quorum is equivalent to difference set and gave out the symmetric quorum from 7 to 111 based on exhaustive computing. LI[5]proposed a quorum generation algorithm based on cyclic coding. It's time complexity is $2\sqrt{N}$ and the quorum length is $2\sqrt{N}$. LI[6]proposed another quorum generation algorithm based on local recursion(Call it as LRA). Its quorum length is $\sqrt{N}$ and the time complexity is $\prod_{n=1}^{\sqrt{N}}(N-n*(n-1)-1)$. LI[7] proposed another quorum generation algorithm based on backup nodes(Call it as ARA). Its quorum length is closed to $\sqrt{N}$ but the time complexity is $O(\sqrt{N}!)$. LRA has shorter quorum length but higher time complexity, and ARA has longer quorum length but lower time complexity. If there is a quorum generation algorithm has LRA's quorum length and ARA's time complexity? In this paper, a quorum generation algorithm will be proposed to resolve this problem.

ALGORITHM IDEA

WK's[4]algorithm proved that cyclic quorum is equal to cyclic difference and proposed to use exhaustive search method to generate quorum. Its quorum length is $\sqrt{N}$ and the time complexity is $O(\sqrt{N}!)$. When the scale of a distributed system is larger than 150, the quorum generation time will be three or four days to a PC server. So reduce the time complexity of the quorum generation and ensure the quorum length is the shortest would be the most important things now.

Figure 1 is the sample of quorum generation process based on exhaustive search algorithm when N=13. Because 13=3*4+1, limit the quorum length to 4. Based on the exhaustive search algorithm, the first node included in the quorum is node 0, the second node is node 1, and the third node is node 2, and so on. When node 3 inserted in the quorum, there are four nodes in the quorum, but there four nodes can not construct a symmetric quorum. Then became the quorum vector from{0,1,2,3} to {0,1,2,4} and so on. The attempt times should be {1+1+1+10}+{1+6}=20.

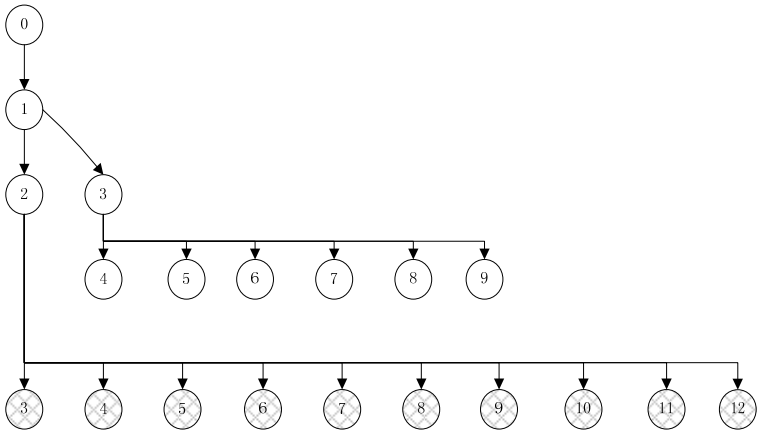

Figure 1: Process of quorum generation based on exhaustive search

In fact, because {2-1}={1-0}, the differences of {2,1} and {1,0}are same. The distance 1 can be repeat generated by the differences of {2,1} and {1,0}. So when the repeat generation times of all distances is larger than the limitation of the quorum, these nodes which generate repeat distance must not be included in the quorum. Because 4*3+1=13, when N=13, quorum length is 4. Four nodes can generate 12 distances. So any distance can not be permit to generate two times. In FIGURE 1, the shadow nodes will generate repeat distances. So they can not be included in the quorum. When we select node 2 to insert in the quorum, calculate the difference set of the quorum, find out there are repeat distances, so node 2 can not included in the quorum, then select node 3 as the third node to insert in the quorum. Use this method to generate the quorum, the attempt times are {1+1+2+6}=10. It's half of the attempt times of the exhaustive search algorithm when N=13.

Definition 1: Repeat expression of distance. In a quorum, if there are more than two pairs of nodes that the differences of their index are same, call the distance represented by the differences is repeat expressed. This phenomenon is called repeat expression of distance.

Definition 2: Repeat expression times of distance. If a distance is repeat expressed, the repeat expressed times are called the repeat expression times of the distance.

Definition 3: Total repeat expression times. In a quorum, the summaries of the repeat expression times of all distance needed in the quorum are called the total repeat expression times in the quorum.

Definition 4: Permission total repeat expression times. In a limited length quorum, the difference of the distances needed to express and the differences generated by any two nodes in the quorum is called the permission total repeat expression of distance.

To a distributed system with N nodes, the quorum length limitation is n, the number of distances needed to express should be N-1, the number of differences generated by the quorum should be n*(n-1), the permission total repeat expression times should be n*(n-1)-(N-1).

Theorem: Repeat expression limitation. If the real total repeat expression times are larger than the permission total repeat expression times, the quorum with n nodes can not exist.

Proof. Assume the quorum length limitation is n, the system scale is N. The number of the differences generated by any two nodes should be n*(n-1). The number of distance needed in the quorum should be N-1, the permission total repeat expression of distance should be n(n-1)-N+1. Assume the real nodes number is m(m<n) now, and the real total repeat expression times are larger than the permission total repeat expression times. Insert other n-m nodes in the quorum, assume there is not any new repeat permission times will be generated, the total repeat expression times will be larger than the permission total repeat expression times. Assume the real total repeat expression times is RT, RT>((n(n-1)-N+1)). That's to say, the real distances generated by n nodes in the quorum should be n(n-1)-RT<n(n-1)- ((n(n-1)-N+1))=N-1. The real distances generated by n nodes in the quorum are smaller than N-1. So the quorum with n nodes can not exist. **End**.

Repeat expression limitation theorem can be used to judge if the quorum with n nodes exist. During the process of quorum generation, if the real total repeat expression times are larger than the permission total repeat expression times but the real number of nodes is smaller than n, the quorum with n nodes can not exist, the n-k nodes last will not try. This method can reduce the time complexity of the quorum generation algorithm because this process can not generated quorum will not be attempted.

From Figure 1, when N=13, n=4, the permission total repeat expression times is 0, the attempt series of nodes should be {0,1,2,3,4,5,6,7,8,9}, the attempt times should be 10. When there are two nodes {0,1}in the quorum, the distance 1 has been expressed, the next node included in the quorum must not generated the distance 1. So the next node can not be {2} because the difference of {1,2} is same as the difference {0,1}. The next node should be {3}. When the quorum state is {0,1,3}, the next node should not be {4},{5}. That's to say, thinking of the needed distances, the attempt nodes can be reduced again. Use the last node's index in the quorum add the distances needed to express as the backup nodes, and use the repeat expression limitation theorem to judge if the quorum exist, we can reduce the time complexity of the quorum generation algorithm remarkably. In Figure 1, if use the backup nodes, the attempt series should be{0,1,3,6,7,8,9} , the attempt times are 7, it's 70% of the attempt times of only use the repeat expression limitation theorem and it's 35% of the exhaustive search when N=13, n=4.

ALGORITHM DESCRIPTION

For describing and controlling the quorum generation algorithm running, some data structures and processes needed as follow in Java style:

Data structure and process

quorum[N]. It's a vector representing quorum 0. Use it to record those nodes included in quorum 0. qumrum[N] is a 0-1 vector with N elements.

back[]. It's a vector used to store these nodes witch could be inserted in the quorum. Because the number of backup nodes can not certain, the vector length can not certain, too.

void init(Quorum(N)). It's a process of initiating of the quorum with N elements. Because any scale system includes the node 0 and node 1, this process will initiate the quorum with node 0 and node 1.

int[] searchStateZero(int[] state). This is a process to search and return those locations can not been generated by the quorum in the distance denoted vector state.

```
TestInsert Process Description
INPUT:Quorum,k
OUTPUT:Quorum

PROCESS DESCRIPTION
ptimes=k*(k-1)-N+1;
if (k == 1) {
        for (int i = 0; i < back.length; i++) {
                times++;
                quorum[back[i]] = 1;//this quorum is the test quorum;
                if (searchStateZero(generateState(q)).length != 0) {
                        quorum[back[i]] = 0;
                }else{
                        this.quorum=quorum;
                          //quorum generation finished;
                        break;

                }
        }
} else {
        for (int i = 0; i < back.length; i++) {
                times++;
                qumrum[back[i]] = 1;
                int rdistances=searchStateNotZero(generateState(quorum)) .length;
                  //real different distances;
                int rtimes=(quorum.length-1)*quorum.length-rdistances;
                  //real repeat times;
                if(rtimes>ptime){//Judge the real repeat times and the permission repeat times;
                        break;
                }else if ((quorum=testInsert(quorum,k - 1))!=null) {
                        break;
                }else{
                        quorum[back[i]] = 0;

                }
        }
}
```

Figure 2 Description of testInsert process

int[] backup(int[] quorum, int[]zero). This is a process to search and return those backup nodes. Trough adding the index of nodes in quorum and any distance in the zero and get those backup nodes.

int[] generateState(int[] quorum). It's a process to generate the distance denoted vector trough subtracting the indexes of nodes in quorum two and two.

int[] testInsert(int[] quorum, int k). It's a core process to test if a node in the backup nodes can be inserted in the quorum successfully. The quorum represents the quorum state now and k represents the nodes number permitted to insert in the quorum. This process is a recursion process. Describe it in Java style as Figure 2.

Algorithm description

According to the data structures and the processes described above, the running process of the quorum generation algorithm proposed in this paper will be described as Figure 3.

```
Algorithm Description
INPUT:N
OUTPUT:quorum

ALGORITHM DESCRIPTION

Input(N);
Init(quorum);
while (quorum.testInsert(quorum, k-2)==null) {
          quorum = new qUorum(N);
          init(quorum);
          k++;
}
```

Figure 3 Formalized specification of algorithm

Example of algorithm running

Assume N=13, show the running process of the algorithm described in Figure 3 as Figure 4. The vector in Figure 4 shows the distance needed to be generated by the quorum.

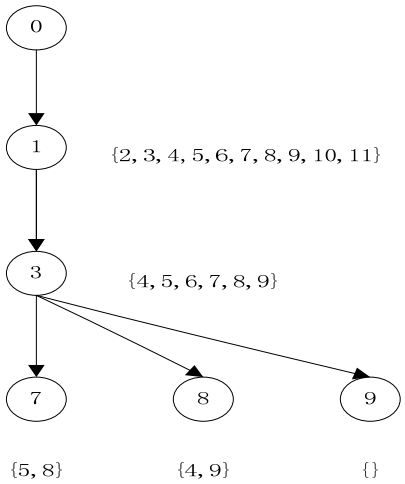

Figure 4 Running process of this algorithm when N=13

The backup node will be generated by the index of the node with the largest index in quorum add the distance needed to be generated from lower to higher. From Figure 4, we know that the attempt times are four when initiate the quorum with $\{0,1\}$.

PERFORMANCR ANALYSIS

Quorum length

Because of the limitation of length, the quorum length of this algorithm proposed in this paper should be close to the lowest limitation of quorum length when N is a given scale. So the quorum length is $\sqrt{N}$. Because every backup node will be inserted after the last node in the quorum, some repeat distance expressions may be occur, the quorum length may be increased a little. So the quorum length of this algorithm should be close to $\sqrt{N}$, but a little higher than $\sqrt{N}$. Show the comparison of the real quorum length with several algorithms as TABLE 1 when $N \in [37,51]$. In TABLE 1, CA represents the cyclic quorum generation algorithm, LRA represents the local recursion algorithm, LLA represents the lowest limit quorum length algorithm(such as Maekawa's algorithm or WK's algorithm), and ARA represents the advanced recursion algorithm. ABL represents the algorithm based on the repeat limitation proposed in this paper. And N is the distributed system scale, every location of TABLE 1 denotes the quorum length corresponding N and different quorum generation algorithms.

TABLE 1: Comparison with several quorum generation algorithms

N	**37**	**38**	**39**	**40**	**41**	**42**	**43**	**44**	**45**	**46**	**47**	**48**	**49**	**50**	**51**
LLA	7	8	7	8	8	8	8	8	8	8	8	8	8	8	8
CA	9	10	9	10	9	10	10	11	11	12	10	12	11	11	11
LRA	7	8	7	8	8	8	8	8	8	8	8	8	8	8	8
ARA	8	8	8	8	8	8	8	8	8	8	8	9	9	9	9
ABL	7	8	7	8	8	8	8	8	8	8	8	8	8	8	8

Form TABLE 1, when $N \in [37,51]$, the quorum length of ABL proposed in this paper is same as the quorum length of LLA, is close to ARA, but a little smaller than ARA. So ABL achieved the first goal that has the same quorum length as LRA when $N \in [37,51]$. Figure 5 shows the comparison of quorum length with LLA and ABL. It shows that ABL has the same quorum length as LLA, or has a litter more length than LLA when $N \in [50,134]$.

Time complexity

The another goal of ABL proposed to reduce the time complexity of the quorum generation algorithm on condition that the quorum length should be close to the lowest limit of quorum.

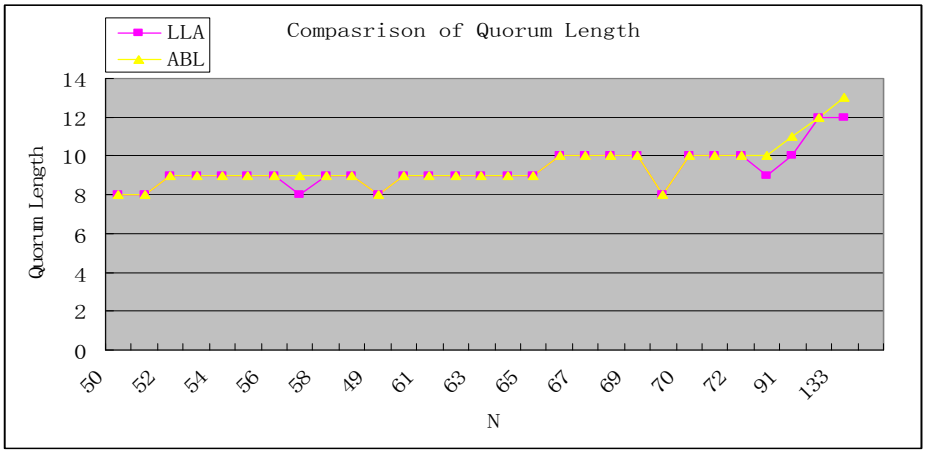

Figure 5:Comparison of quorum length with LLA and ABL

The global recursion quorum generation algorithm has $N-n$ optional nodes when there had been n nodes in the quorum. If the quorum length is $\sqrt{N}$, the attempt times should be $(N-1)!/(N-\sqrt{N}-1)!$. LRA has N-(n-1)*n-1 optional nodes when there are n nodes in the quorum. If the quorum length is $\sqrt{N}$, the attempt times should be $\prod_{n=1}^{\sqrt{N}}(N-n*(n-1)-1)$ .ARA only has n optional nodes when there are n nodes in the quorum. When the quorum length is $\sqrt{N}$, the attempt times should be $\prod_{n=1}^{\sqrt{N}} n=\sqrt{N}!$. So the time complexity of ARA should be $O(\sqrt{N}!)$. Because there are (N-1)-n(n-1) optional nodes to insert in the quorum of ABL when there are n nodes in the quorum, the attempt times should be $\prod_{n=1}^{\sqrt{N}}(N-n*(n-1)-1\approx N^{\sqrt{N}}/2^{\sqrt{N}})$. It's same as LRA. Show the real attempt times comparison of LLA, LRA ,ARA and ABL when $N\in[37,51]$ as TABLE 2 when the initiation nodes all are two. Among it Ratio is the real time ratio of ABL and LRA. Because the time complexity is directly proportional to the attempt times, the time complexity of quorum generation algorithm will represent by the attempt times. In TABLE 2, Ratio 1 denotes the percentage of the time complexity of ARA and LRA, Ratio 2 denotes the percentage of the time complexity of ABL and LRA

TABLE 2: Comparison the real time of several quorum generation algorithms

N	WKA	LRA	ARA	ABL	Ratio1	Ratio2
37	1.03E+07	21348	921	654	4.31%	3.06%
38	6.15E+07	7725089	930	38372	0.01%	0.50%
39	1.54E+07	17516	930	1009	5.31%	5.76%
40	9.55E+07	11446634	939	7475	0.01%	0.07%
41	1.18E+08	13608195	939	7258	0.01%	0.05%
42	2.70E+07	17516	940	1911	5.37%	10.91%
43	1.77E+08	19401328	1121	1433	0.01%	0.01%
44	1.77E+08	41898	4389	135	10.48%	0.32%
45	2.16E+08	21671	2218	6	10.23%	0.03%
46	2.61E+08	42481	5997	3663	14.12%	8.62%
47	3.14E+08	28647	5904	73	20.61%	0.25%
48	3.77E+08	1194468	5998	5545	0.50%	0.46%
49	4.51E+08	1774481	3229	427	0.18%	0.02%
50	5.37E+08	28142109	6310	1903	0.02%	0.01%
51	6.37E+08	5677414	6310	35146	0.11%	0.62%

From TABLE 2, it's sure that the time complexity of ABL is far less than the time complexity of LRA. The maximum Ratio 1 is 20.61%, the maximum Ratio 2 is 10.91% when $N \in [37,51]$. That means ARA can reduce the time complexity at least 79% to LRA and ABL can reduce the time complexity at least 89%. That's to say, the time complexity of ABL is lower than ARA. So ABL has the same quorum length as LRA and has lower time complexity than ARA.

Figure 6 shows the comparison of the time complexity of ARA and ABL. It shows that the maximum ratio of ARA is larger ABL. That's to say, ABL can reduce the time complexity to lower than ARA.

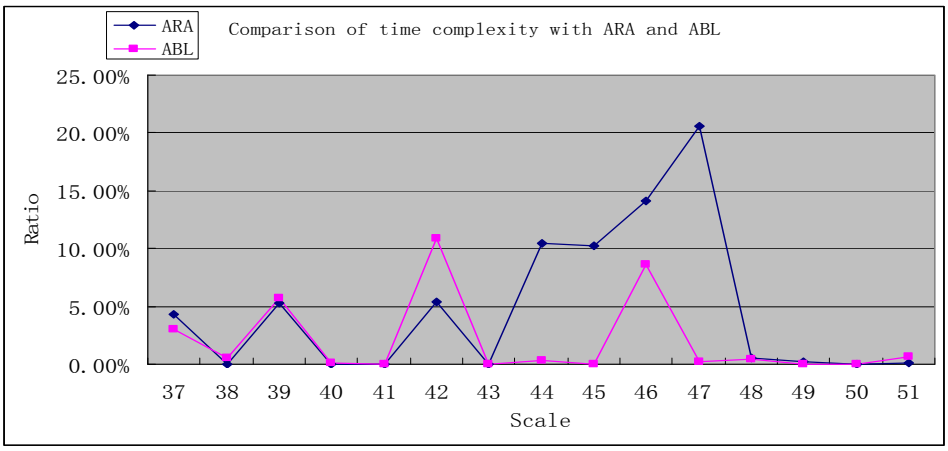

Figure 6: Comparison of time complexity with ARA and ABL

Space complexity

ABL needs one quorum, one distance denoted vector, and one backup nodes vector, so the space complexity is $O(N)$. It's same as the space complexity of LLA or LRA and ARA.

CONCLUSION

ABL reduces the time complexity from $O(N!/(N-\sqrt{N})!)$ of LLA to $O(N^{\sqrt{N}}/2^{\sqrt{N}})$ on the condition that the quorum length has been ensured to be $\sqrt{N}$, the space complexity has been ensured to be $O(N)$. They are all same as the quorum length and space complexity of LLA and LRA. But it's shorter than ARA. When $N \in [37,51]$. ABL can reduce the time complexity 89% at least to LRA, and reduce 10% at least to ARA and has the same quorum length as LRA and LLA.

REFERANCE

[1] L Lamport.Time;clocks and ordering of events in distributed systems.Comm.ACM. 558, 21(7), 1978.

[2] G Ricart,A K Agrawala;An optimal algorithm for mutual exclusion in computer networks. Communications of the ACM. 9,24 (1),1981.

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

[4] Ng W K, Ravishanker C V;Coterie Twmplates-A New Quorum Construction Method..Proc. Of the 15th International Conference on Distributed Computing Syatem. Piscataway, New Jersey,USA,1995.

[5] LI Mei-An,Liu Xin-Song,Wang Zheng;High performance distributed mutual exclusion algorithm based on cyclic coding. Tien Tzu Hsueh Pao/Acta Electronica Sinica. 1397, 33(8), 2005.

[6] LI Mei-an,LIN Lan,CHENG Zhi-dang;Distributed Cyclic Quorum Generation Algorithm Based on Binary Plus One. Computer Engineering.59-61, 38(14),2012.

[7] LI Meian,Wang Buyu,Chao Lumeng,Liu Jiangping;Symmetric Distributed Quorum Generation Algorithm Based on Backup Nodes. Information Technology Journal. 3781,15,2013.