基于读写特征的分布式互斥算法

刘 丹, 刘心松, 丘志杰, 邱元杰
(电子科技大学 8010 研究室, 四川成都 610054)

摘 要: 在 LK 算法基础上, 提出一种对读写请求作不同互斥处理的分布式互斥算法))) RWME(read/ write mu2 tual exclusion) 算法. 在同步延迟仍为 T 的前提下, 降低了传统非令牌类型互斥算法的消息复杂度. 在 Lamport 全局时戳的基础上, 定义了适合于读写互斥的全局时戳))) 读写时戳, 并由其来保证各读写进程互斥访问临界区的公平性和正确性. 通过对算法的性能分析验证其是高效的, 并给出了正确性证明.

关键词: RWME 算法; 分布式互斥; 读写时戳; 消息复杂度
中图分类号: TP31614 文献标识码: A 文章编号: 03722112 (2004) 022032622 04

A Distributed Mutual Exclusion Algorithm Based on Read/ Write Character

LIU Dan, LIU Xin2song, QIU Zhi2jie, QIU Yuan2jie
(8010 Division, University of Electronic Science & Technology, Chengdu, Sichuan 610054, China)

Abstract: Based on LK algorithm, a distributed mutual exclusion algorithm) read/ write mutual exclusion (RWME) algorithm is presented. It puts different mutual exclusion operations for reading request and writing request. The algorithm belongs to nontoken2 based type. It saves the message complexity and still has T synchronization delay. A read/ write globe clock stamp which based on Lamport clock stamp is defined for the read/ write mutual operations. Using the read/ write globe clock stamp, reading and writing re2 quests can access critical sections with fairness and freedom from deadlock or starvation. Proved by performance analysis, the algorithm has high2 performance. A correctness proof is provided.

Key words: RWME algorithm; distributed mutual exclusion; read/ write clock stamp; message complexity

1 引言

近年来出现了许多高效的分布式互斥算法[2, 3], 包括基于令牌[4~ 6]和非令牌[7~ 14]类型的. 基于令牌类型的消息复杂度(一次访问临界区 CS(critical section) 的消息收发数) 可降低至 O(logN), 同步延迟一般为 O(logN) T(T 为消息传送的平均延迟). 基于非令牌类型的消息复杂度介于[0, 3* (N- 1)] 范围, 同步延迟介于[T, 2T] 范围. 本文提出的 RWME 算法属于非令牌类型, 其在 LK[1] 算法基础上, 对于读写操作作不同互斥处理, 以降低消息复杂度, 并基于 Lamport 全局时戳[7], 定义了适合于读写互斥访问的全局时戳))) 读写时戳, 该时戳对于读请求进行子编号, 避免了读操作饥饿现象.

2 基于读写特征分布式互斥算法))) RWME 算法

211 算法思想及数据结构

系统中 N 个节点通过网络连接且没有共享存储器, 节点间消息通信被认为是 FIFO、无差错且有限延迟的. 一节点可包含多个进程.

采用读(R) 锁和写(W) 锁实现互斥. 当请求对 CS 读操作时, 加 R 锁; 请求写操作时, 加 W 锁. R 锁记录在本地, W 锁扩散到网络. 互斥发生在 R 锁与 W 锁之间及 W 锁之间. 申请 R 锁通常不涉及消息通信, 仅在极少数情况下为避免死锁将 R 锁扩散向网络.

节点 S_i 维护一个锁队列 LQ_i (图 1), 每个正被访问的 CS_j 对应的锁表示为 $L_{(j, t, m)}$, 简记为 L_j. 锁状态 t I {n, r, w}, n 表示该锁无属主; r 表示锁属主为读请求进程, 即 R 锁; w 表示锁属主为写请求进程, 即 W 锁. 若是 R 锁, m 为共享计数.

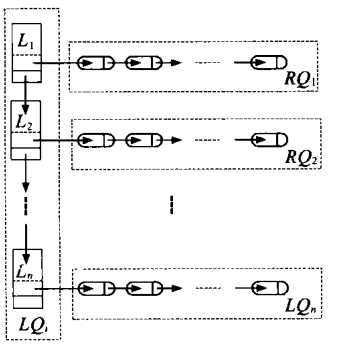

图 1 节点 i 上的锁队列及其请求关系图

L_j 对应一请求队列 RQ_j, 记录按优先权排序的未完成访问 CS_j 的请求. 当 RQ_j = 5 时, 将 L_j 从 LQ_i 中删除.

收稿日期: 2003202228; 修回日期: 2003207202
基金项目: 四川省科技攻关项目(No. 02GG006018)

定义四类消息,请求 REQ、应答 REP、集合应答 CREP 和锁转换 CHG. 若不作特殊说明, P_i 表示节点 S_i 上进程, R_i 表示 P_i 发出的 REQ.

定义 1　同时请求　若在 R_j 生存期中又产生 R_i,或在 R_i 生存期中又产生 R_j,则 R_i 和 R_j 互为同时请求,表示为集合 $Cset_i$ 或 $Cset_j$

REQ 记为 $R_{(i,j,c,t,s)}$,简记为 R_i. 当进程请求加锁时产生. 其中 i 表示消息发送进程 P_i, j 表示请求访问 CS_j, c 为优先权,也记为 c_{R_i}, t I {r, w} 表示读或写请求, s I {l, n} 表示请求所处的状态))) 本地就绪态(l)或网络阻塞态(n). l 态指该请求已具有获得锁的权力(经协商已收到所有应答),在等待其他高优先权请求释放锁; n 态指该请求等待网络协商批准其获得锁的权力.

用 $RV_i[k]$, P k I [1, N] 表示 R_i 的应答集合. $RV_i[k]=1$ 表示 P_k 已对 R_i 应答,当所有节点应答后, R_i 由 n 态变为 l 态. 对于 $R_{(i,j,c,r,l)}$,无需进行网络协商. 初始时 $RV_i[k]=1$, P k I [1, N]. 对于 $R_{(i,j,c,w,n)}$,初始时 $RV_i[k]=0$, P k I [1, N], $RV_i[i]=1$.

R_i 也解释为同时请求的应答消息. 即 P_j 收到 R_i 时,若同时 P_i 也收到 R_j,双方可比较 cR_i 和 cR_j 而有序地访问 CS,无需给对方再发送 REP[1].

定义 2　R_i 的前驱和后继

$Pred(R_i, RQ_j)=R_j$,当 R_j I RQ_j H $(c_{R_i} < c_{R_j})$ H $(R_k=5)$,其中 $\{R_k \mid c_{R_i} < c_{R_k} < c_{R_j}\}$

$Succ(R_i, RQ_j)=R_j$,当 R_j I RQ_j H $(c_{R_i} > c_{R_j})$ H $(R_k=5)$,其中 $\{R_k \mid c_{R_i} > c_{R_k} > c_{R_j}\}$

REP 记为 $REP_{(i,j,c)}$, CREP 记为 $CREP_{(j,c)}$,其中 i 表示消息发送进程 $P_{i,j}$ 表示请求访问 CS_j, c 表示其应答的 R_k 的优先权. 当 P_i 退出 CS 时,在 RQ_j 中选出优先权最高的 R_k,并向 P_k 发送 $CREP_{(j,c)}$. P_k 收到 $CREP_{(j,c)}$,则认为所有优先权高于 R_k 的请求都完成了对 CS 的访问. 将 RQ_j 中所有优先权高于 R_k 的请求删除. 设置 $RV_k[m]=1$, P m I [1, N].

当 P_i 完成写请求退出 CS 时,若该请求的后继是读请求,则 S_i 相应的锁类型由 w 变为 r,同时向后继所在节点发送消息使后继访问 CS. 然而除后继节点外的其他节点的锁类型仍为写锁,而此时是读请求在访问 CS,若其他节点上的进程有读请求,应能同时访问 CS. 由于这些节点上的锁类型不能及时更新,造成读读互斥,降低了并发性.

为此,使用消息 CHG,在上述情况发生时,退出 CS 的进程向所有其他非后继请求所在节点发送 CHG 消息要求将锁类型由 w 变为 r. 另外,若 S_i 只发起过读请求,因外节点 S_j 不包含 S_i 的读请求, S_j 在退出 CS 而放锁时不能查知 S_i 有同时的读请求,不会给 S_i 发送 CREP 消息,导致其他进程的请求在 S_i 的队列中始终存在,使排在这些请求后面的请求永久得不到处理,造成请求进程死等.

为此,当产生读请求时,若其前趋是其他节点的写请求,则向该前趋发送该读请求. 这样当前趋请求完成时,将发送 CREP 消息通知该读请求进入 CS. 此时,读请求扩散向网络.

2.2　读写时戳

由于读请求不向网络扩散,若直接采用 Lamport 时戳来生成各请求的优先权会产生饥饿情况. 按 Lamport 时戳, S_i 上 R_i 的 c 为(SN, PID),其中 SN 是 S_i 独立编号的请求消息序号, PID 是 S_i 的标号. 设 S_i 连续发生读请求而无写请求,其 SN 值很大,而这些请求不会扩散到网络,因此其他节点的 SN 可能很小,以至其他节点的写请求即使后发生也因 SN 很小而使其优先权高于 S_i 上先发生的读请求,造成饥饿.

定义 3　读写时戳 (SN_w, SN_r, PID), SN_w 同 Lamport 时戳中 SN,用于网络优先权比较. SN_r 作为序号的补充,用于本地优先权比较. 对于写请求, $SN_r=0$. 两个写请求之间的所有连续读请求的 SN_w 相同,而它们的优先权以 SN_r 来判定.

R_i 的 c_{R_i} 按读写时戳生成和比较.

定义 4　读写时戳比较

$(SN_{w1}, SN_{r1}, PID_1) > (SN_{w2}, SN_{r2}, PID_2)$: 当且仅当 $(SN_{w1} < SN_{w2})$ 或 $(SN_{w1}=SN_{w2}$ 且 $SN_{r1} < SN_{r2})$ 或 $(SN_{w1}=SN_{w2}$ 且 $SN_{r1}=SN_{r2}$ 且 $PID_1 < PID_2)$

2.3　算法实现

算法包括申请读锁、释放读锁、申请写锁、释放写锁、收到 REQ、收到 REP、收到 CREP 及收到 CHG 几个过程,如下.

CheckExecuteCS($R_{(i,j,c,t,s)}$)

若 $R_{(i,j,c,t,s)}$ 是 RQ_j 中优先权最高者,且 $RV_i[k]=1$, P k I [1, N],则返回真,否则返回假.

P_i 申请读锁 $L_{(j,r,m)}$　(1)若 LQ_i 中无 $L_{(j,t,m)}$ 则产生 $L_{(j,r,m)}$;(2)产生 $R_{(i,j,c,r,l)}$ 并插入 RQ_j 中,若 CheckExecuteCS $(R_{(i,j,c,r,l)})$ 为真,则获得读锁,返回;(3)若 $R_{(i,j,c,r,l)}$ 前趋不是本地请求,将 $R_{(i,j,c,r,l)}$ 发给其前趋所在节点;(4)若 CS_j 的锁是读锁 $L_{(j,r,m)}$,且 RQ_j 中无优先权高于 c_{R_i} 的写请求,则 m = m+ 1, P_i 获得读锁,返回;(5)阻塞 P_i 于 $R_{(i,j,c,r,l)}$ 上;唤醒后, m= m+ 1,获得 $L_{(j,r,m)}$,返回.

P_i 释放读锁 $L_{(j,r,m)}$　(1)将 $R_{(i,j,c,r,l)}$ 从 RQ_j 中删除, m= m - 1,若 m X 0,则返回;(2)若 $RQ_j=5$,将 $L_{(j,r,m)}$ 删除,返回;(3)将 $L_{(j,r,m)}$ 变为 $L_{(j,w,m)}$;(4)若 $R_{(i,j,c,r,l)}$ 的后继是本地就绪写请求 $R_{(i,j,c,w,l)}$,唤醒该后继请求上的阻塞进程,返回;(5)若 $R_{(i,j,c,r,l)}$ 的后继是远地写请求 $R_{(k,j,c,w,n)}$,则向后继节点发送 $REP_{(i,j,c)}$.

P_i 申请写锁 $L_{(j,w,m)}$　(1)若 LQ_i 中无 $L_{(j,t,m)}$ 则产生 $L_{(j,w,m)}$;(2)产生 $R_{(i,j,c,w,n)}$ 并发送到其他所有节点;(3)将 $R_{(i,j,c,w,n)}$ 插入 RQ_j 中, P_i 阻塞于 $R_{(i,j,c,w,n)}$ 上;唤醒后获得 $L_{(j,w,m)}$ 返回.

P_i 释放写锁 $L_{(j,w,m)}$　(1)删除 $R_{(i,j,c,w,l)}$. 若 $RQ_j=5$,删除 $L_{(j,w,m)}$,返回;(2)取 RQ_j 中优先权最高的 $R_{(k,j,c,t,s)}$,若是写请求则转(3);否则转(4);(3)若 k= i,将 $R_{(i,j,c,w,l)}$ 上的阻塞进程唤醒,返回. 若 k X i,发送 $CREP_{(j,c)}$ 给 P_k,返回;(4)将 $L_{(j,w,m)}$ 变为 $L_{(j,r,m)}$;(5)在 RQ_j 中从 $R_{(k,j,c,r,l)}$ 开始的连续 n 个读请求的集合 Z 中,若优先权最高者 $R_{(k,j,c,r,l)}$ 是本地转(6),否则转(7);(6)将 Z 中所有本地读请求上的阻塞进程唤醒,向其他进程发送 $CHG_{(j,c)}$,返回;(7)向 P_k 发送

$CREP_{(j,c)}$, 向其他进程发送消息 $CHG_{(j,c)}$, 返回.

P_k 收到 REQ (1) 当 P_k 收到 $R_{(i,j,c,t,s)}$, 将其插入 RQ_j 中, (若无 $L_{(j,t,m)}$ 则产生); (2) 若 RQ_j 队列中无本地请求 $R_{(k,j,\infty,t,s)}$, 则发送 $REP_{(k,j,c)}$ 给 P_i, 返回; (3) 对所有本地请求置 $RV_k[i]=1$, 找出本地请求中所有优先权高于 c 的请求集合 Z, 若 Z X 5 转 (6); (4) 若本地只有写请求, 则在产生 $R_{(k,j,c,t,s)}$ 时, P_k 曾发送 $R_{(k,j,\infty,t,s)}$ 给 P_i, 该消息将被作为应答消息, 返回; (5) 若本地有读请求, 其前趋因本次消息达到发生了变化且变化后为写请求, 则将相应读请求发给 P_i, 返回; (6) 对 Z 中最高优先权 $R_{(k,j,cd,t,s)}$, CheckExecuteCS ($R_{(k,j,cd,t,s)}$) 为真则唤醒阻塞在 $R_{(k,j,cd,t,s)}$ 上的进程, 返回.

P_i 收到 REP (1) 收到 $REP_{(k,j,c)}$, 设置 $RV_i[k]=1$; (2) Check2 ExecuteCS ($R_{(i,j,c,t,s)}$) 为真, 将 $R_{(i,j,c,t,s)}$ 上的阻塞进程唤醒;

P_i 收到 CREP (1) 收到 $CREP_{(j,c)}$, 置 $RV_i[k]=1$, P k I [1, N]. 将 RQ_j 中所有优先权 $\infty > c$ 的请求删除; (2) 若 $R_{(i,j,c,w,n)}$ 是写请求, 将 $R_{(i,j,c,w,n)}$ 变为 $R_{(i,j,c,w,l)}$, 将 $L_{(j,t,m)}$ 变为 $L_{(j,w,m)}$, 将 $R_{(i,j,c,w,l)}$ 上阻塞进程唤醒, 返回; (3) 若 $R_{(i,j,c,r,l)}$ 是读请求, 将 $L_{(j,t,m)}$ 变为 $L_{(j,r,m)}$, m= 0, 将 $R_{(i,j,c,r,l)}$ 上阻塞进程唤醒, 返回.

P_i 收到 $CHG_{(j,c)}$ (1) 将 $L_{(j,w,m)}$ 变为 $L_{(j,r,m)}$, m= 0; (2) 将 RQ_j 中所有优先权 $\infty > c$ 的写请求删除; (3) 判断 RQ_j 中有无本地读请求, 且其优先权高于 RQ_j 中任何写请求, 有则唤醒这类请求上的阻塞进程.

2.4 算法示例

图 2~5 给出了算法的一些情况研究. 无论在何种情况下, 其消息数都比传统算法低, 尤其是在并发读请求频度高时, 优势更明显.

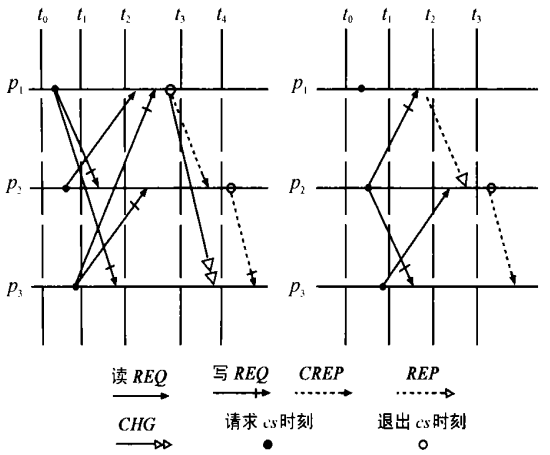

图 2 p_1p_3 为写请求 p_2 为读请求消息数为 8

图 3 p_1p_3 为读请求 p_2 为写请求消息数为 5

图 2, 图 3 的请求顺序为 $p_1 > p_2 > p_3$.

3 消息复杂度

定义 5 若在写请求 R_{jw} 生存期中又产生写请求 R_{iw}, 或在 R_{iw} 生存期中又产生 R_{jw}, 则 R_{iw} 和 R_{jw} 互为同时写写请求, 表示为集合 $Cset_{iw}$ 或 $Cset_{jw}$.

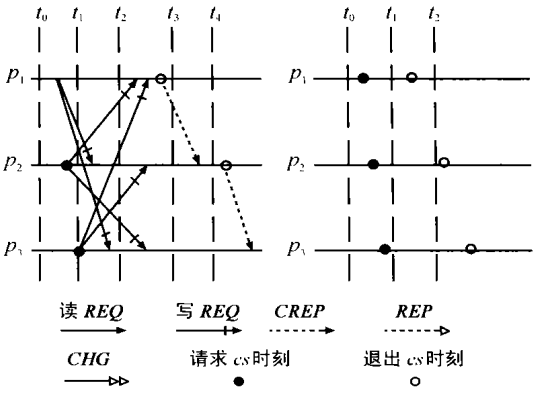

图 4 $p_1p_2p_3$ 为写请求消息数为 8

图 5 $p_1p_2p_3$ 为读请求消息数为 0

图 4, 图 5 的请求顺序为 $p_1 > p_2 > p_3$

定理 1 写请求的消息复杂度介于 [(N21), 3N2| $Cset_{iw}$| - 2] 之间

证明 P_i 请求写访问 CS_j, P_i 送 (N- 1) 个 R_i, 并将收到 (N- | $Cset_{iw}$|) 个应答.

当| $Cset_{iw}$| \ 2, 分三种情况

(1) 若 R_i 的后继为写请求, 则 P_i 完成对 CS_j 访问时将发送 CREP. 这样, 访问 CS_j 的消息数为 2N- | $Cset_{iw}$| . 当所有进程都同时发生写请求时, 消息数为 N.

(2) $Cset_{iw}$ 中无请求的 $c_{Rk} < c_{Ri}$, P_i 将不发送 CREP 消息. 这样, 访问 CS_j 的消息数为 2N- | $Cset_{iw}$| - 1. 当所有进程都同时发生写请求时, 消息数为 N- 1.

(3) R_i 的后继为读请求, 则 P_i 在访问 CS_j 结束时将发送 CREP 消息给后继, 并发送 N- 2 个 CHG 消息. 这样, 访问 CS_j 的消息数为 3N- | $Cset_{iw}$| - 2, 当所有进程都同时发生写请求时, 消息数为 2(N- 1).

当| $Cset_{iw}$| = 1, 即所有请求都是串行发生的. 消息数为 2 (N- 1).

定理 2 读请求的消息复杂度介于 [0, 2] 间

证明 当 P_i 请求读访问 CS_j, 有四种情况: (1) R_i 的前趋和后继都不是其他进程的写请求, 则不发消息, 消息数为 0; (2) R_i 的前趋是其他进程的写请求而后继不是其他进程的写请求, 则加 R 锁时发送一个 REQ 消息, 放 R 锁时不发消息, 消息数为 1; (3) R_i 的前趋不是其他进程的写请求而后继是其他进程的写请求, 则加 R 锁不发消息, 放 R 锁时发一个 CREP 消息, 消息数为 1; (4) R_i 的前趋和后继都是其他进程的写请求, 则加 R 锁时发送一个 REQ 消息, 放 R 锁时发一个 CREP 消息, 消息数为 2.

4 算法正确性证明

4.1 公平性

定义 6 R_j 和 R_i 的距离 dist(R_j, R_i)= 1+ | R_k| , | R_k| 表示 R_k 的数目, {R_k| $c_{Rj} > c_{Rk} > c_{Ri}$}

定理 3　RWME 算法具有公平性, 即证若 $c_{Rj} > c_{Ri}$] P_j 在 P_i 前访问 CS.

证明　归纳法

(1) 设 $Pred(R_i, RQ_i) = R_j$, 即 $dist(R_j, R_i) = 1$, 有以下二种情况: (a) 若 R_i I $Cset_j$, 则 P_i、P_j 访问 CS 前都将收到对方的请求, 由于 P_i 优先权低, 所以 P_j 先访问 CS; (b) 若 R_i | $Cset_j$, 则在 P_i 产生 R_i 前, 已经应答 P_j. 因此, P_j 必在 P_i 前访问 CS.

因此, 当 $c_{Rj} > c_{Ri}$ 且 $dist(R_j, R_i) = 1$, 则 P_j 在 P_i 前访问 CS.

(2) 设 $dist(R_j, R_i) = x, x > 1$, 有 P_j 在 P_i 前访问 CS. 需证明在 $dist(R_j, R_i) = x+1$ 时, P_j 在 P_i 前访问 CS.

设 $Pred(R_i, RQ_i) = R_k$, 有

$dist(R_k, R_i) = 1$, 由(1)] P_k 在 P_i 前访问 CS　　(1)

$dist(R_j, R_k) = x$] P_j 在 P_k 前访问 CS　　(2)

由上式(1) (2) 得, P_j 在 P_i 前访问 CS.

由式(1) (2), 若 $c_{Rj} > c_{Ri}$] P_j 在 P_i 前访问 CS, 证毕.

412　防止饥饿

定理 4　RWME 算法不会产生饥饿

证明　归纳法

设 R_j 在系统中优先权最高

(1) 当 $dist(R_j, R_i) = 1$ 时, R_i 必能访问 CS.

(a) P_j 是系统中请求优先权最高的进程, 其必能访问 CS; (b) P_j 访问 CS 完成后将给 P_i 发应答, 由于 R_i 的优先权仅次于 R_j, 所以在收到 P_j 的应答后, P_i 将是优先权最高的进程, 其必为就绪请求. 因此 P_i 必能访问 CS.

(2) 设 $dist(R_j, R_i) = x, x > 1$ 时, R_i 必能访问 CS. 需证明在 $dist(R_j, R_i) = x+1$ 时, R_i 必能访问 CS.

(a) 若 $dist(R_j, R_k) = x$, 则有 R_k 必能访问 CS.

(b) 又 $dist(R_k, R_i) = 1$, 由式(1), R_i 必能访问 CS, 证毕.

5　结论

本文提出了一种对读写访问的不同互斥特征作不同处理分布式互斥算法, 算法降低了消息复杂度, 尤其在请求并发度高且读请求频繁发生时, 消息复杂度得到明显降低. 该算法在 Lamport 时戳基础上, 针对读操作的特殊性, 定义读写时戳, 保证读写请求公平有序地访问 CS. 在 863 重大专项/ 缩小数字鸿沟- 西部行动0 中的课题/ 低成本公共信息平台建设和应用示范(电子政务)0 的应用中, 表明该算法是高效的.

参考文献:

[1] Lodha S, Kshemkalyani A. A fair distributed mutual exclusion algorithm [J]. IEEE Trans. Parallel and Distributed Systems, 2000, 11(6): 537 - 549

[2] Y I Chang. A simulation study on distributed mutual exclusion [J]. J. Parallel and Distributed Computing, 1996, 33: 107- 121.

[3] M Singhal. A taxonomy of distributed mutual exclusion [J]. J Parallel and Distributed Computing, 1993, 18(1): 94- 101.

[4] J Helary, A Mostefaoui, M Raynal. A general scheme for token and tree2based distributed mutual exclusion algorithms [J]. IEEE Trans. Parallel and Distributed Systems, 1994, 5(11): 1185- 1196.

[5] Y C Kuo, S T Huang. A geometric approach for constructing coteries and K coteries [J]. IEEE Trans. Parallel and Distributed Systems, 1997, 8(4): 402- 411.

[6] M Naimi, M Trehel. An improvement of the log(n) distributed algo2rithm for mutual exclusion [A]. Proc. Seventh Int'l Conf. Distributed Computing System [C]. Berlin, 1987. 371- 375.

[7] L Lamport. Time, clocks and ordering of events in distributed systems comm [J]. ACM, 1978, 21(7): 558- 565.

[8] G Ricart, A K Agrawala. An optimal algorithm for mutual exclusion in computer networks comm [J]. ACM, 1981, 24(1): 9- 17.

[9] M Singhal. A dynamic information structure mutual exclusion algorithm for distributed systems [J]. IEEE Trans. Parallel and Distributed Sys2tems, 1992, 3(1): 121- 125.

[10] O Carvalho, G Roucairol. On mutual exclusion in computer networks, technical correspondence. Comm [J]. ACM, 1983, 26(2): 146- 147.

[11] M Maekawa. A $\sqrt{N}$ algorithm for mutual exclusion in decentralized sys2tems [J]. ACM Trans. Computer Systems, 1985.

[12] W S Luk, T T Wong. Two new quorum based algorithms for distributed mutual exclusion [A]. Proc. 17th Int'l Conf. Distributed Computing Sys2tems [C]. Baltimore, MD, USA, 1997.

[13] D Peleg, A Wool. Crumbling walls: A class of practical and efficient quorum system [J]. Distributed Computing, 1997, 10(2): 120- 129.

[14] Guohong Cao, Singhal M. A delay2optimal quorum2based mutual exclu2sion algorithm for distributed systems [J]. IEEE Trans. Parallel and Distributed Systems, 2001, 12(12): 1256- 1268.

作者简介:

刘　丹　男, 1969 年 4 月生于成都市, 讲师, 博士生, 研究方向为分布式并行处理, 宽带网络与通信, 操作系统与网络软件等.

刘心松　男, 1940 年 12 月生于四川石柱, 教授, 博士生导师, 研究方向为分布式并行处理, 宽带网络与通信, 操作系统与网络软件等, 已在正式学术刊物上发表学术论文 100 余篇.