基于半环仲裁集的环网分布式互斥算法*)

王 征[1,2] 刘心松[2]

(西南财经大学经济信息工程学院 成都 610074)[1]

(电子科技大学计算机科学与工程学院 8010 研究室 成都 610054)

摘 要 分布式互斥是环网分布式系统的重要问题。根据此类系统的特点,提出了新型的分布式互斥算法。该算法以请求者自身为中心,基于半环生成分布式互斥仲裁集;采用 Lamport 逻辑时戳保证消息的时序性;算法采用"探测"消息进行系统的容错处理。分析与仿真证明,该算法具有较低的消息复杂度、较短的响应延迟以及较好的容错性能。

关键词 分布式互斥,环网,仲裁集

Ring Network Distributed Mutual Exclusion Algorithm Based on Half-ring Quorums

WANG Zheng[1,2] LIU Xin-song[2]

(School of Economic Information Engineering, Southwest University of Finance and Economics, Chengdu 610074, China)[1]

(8010 R&D, University of Electronic Science and Technology, Chengdu 610054, China)[2]

Abstract Distributed Mutual Exclusion (DME) is an important problem of distributed ring systems. According to the properties of ring networks, a novel algorithm is presented for them. Based on these half-rings, the algorithm generates distributed mutual exclusion quorums and regarded requesters as centers. And Lamport's logical timestamps are utilized to guarantee the time sequence. Furthermore, "Probe" messages are employed to implement the fault-tolerance of the algorithm. Analysis and simulation results show that it has lower message complexity; shorter response delay and better fairness than the traditional algorithms do so.

Keywords Distributed mutual exclusion, Ring networks, Quorum

1 引言

随着大规模生物及化学计算、天气预报、石油勘探等行业对超级计算能力的需求增长,环网结构受到了超级计算工业领域的普遍重视[1,2]。但目前对环网系统的研究主要集中于应用算法的实现,对提高环网系统本身的理论研究还不够全面。网络中的节点往往需要并发的访问一个共享资源,而该资源只能被一个节点使用;访问该资源的代码段被称为临界区(CS)。如何保证节点互斥的使用临界区、缩短进入临界区的响应时间、减少互斥操作所需的消息数等问题是分布式互斥算法所研究的重点。

在互斥算法研究领域中,Lamport, Richard& Agrawala 和 Maekawa 等人对全分布互斥算法做出了里程碑式的贡献[3]。其中,Lamport 提出了逻辑时戳(Logical Timestamp)的概念,在没有全局物理时钟的全分布式系统中,消息可以根据逻辑时戳表示的优先级进行全局排序。Richard&Agrawala巧妙地利用延迟应答的方式将系统中各进程的资源请求按逻辑时戳组成一个动态令牌环,该算法中应答与释放消息合而为一,将算法具有在 N 个节点的系统中的进入一次临界区所需消息数降为 $2(N-1)$,该算法尽管简单易行,但是消息复杂度和时间复杂度均很高。Maekawa 提出了仲裁集(Quorum)的概念[4],并提出了基于有限投影平面的仲裁集构建方法,消息数下限为 $3m-5m(M=\log N)$。从而将算法的消息复杂度降为 $O(\log N)$。传统的 Maekawa 算法,单个节点不用请求系统中所有节点的许可。只需要请求一个节点请求集的许可。该算法的请求集需要满足以下条件,这里设网络系统有 N 个节点 $\{p_1, p_2, p_3, \cdots, p_N\}$;仲裁集为 $QS=\{S_1, S_2, S_3, \cdots, S_k\}$,其中 $K=\sqrt{N}$,S_i 中包含 $\sqrt{N}$ 个节点:

(1) $\forall S_i, S_j \subset QS \quad S_j \cap S_i \neq \emptyset$ 相交性;

(2) $\forall i \leqslant N \quad P_i \in S_i$ 等职性;

(3) $\forall S_i, S_j \subset QS \quad \forall i \neq j \quad S_i \not\subset S_j$ 最小性。

其中,仲裁集只要满足(1)(2),就能够保证互斥的正确性,在竞争发生时,相交区域的节点作为仲裁节点决定请求节点进入临界区域的顺序;(3)主要是为了提高系统性能,而非必要条件。目前,研究人员提出了一些仲裁集的构建算法,但是这些通常都是基于系统的逻辑结构,而非物理结构;脱离系统实际拓扑结构的算法设计将导致互斥操作的低响应率和高消息复杂度,进一步导致系统性能下降[5]。

本文提出了基于半环仲裁集的环网分布式互斥算法(RNDME:Ring Network Distributed Mutual Exclusion)。第 2 节介绍算法的主要思想;同时,给出算法采用的数据/消息结构;最终给出算法描述,并证明该算法的正确性。第 3 节从理论上分析算法的性能,主要包括消息复杂度、响应延迟等。最后将给出 RNDME 的仿真实验结果,并与其他算法作对比。最后总结全文。

2 RNDME 算法

本节首先给出算法的主要思想,并说明其可行性;其次给出算法所需的数据结构和消息结构;最后给出算法实现的伪代码。

2.1 算法思想

RNDME 算法充分考虑了环网(Ring)系统的实际拓扑结

*)本文受四川应用基础研究项目(NO. 04JY029-017-2)和科技型中小企业技术创新基金(04C26225110223)资助。王 征 博士,主要研究方向为分布式系统、网络操作系统等;刘心松 教授,博士生导师,主要研究方向为分布式并行技术等。

构对分布式互斥算法带来的影响：环网系统中的节点通常有2条链路；所有节点构成一个 N 个节点的环；节点相互路由信息，采取存储转发通信模式；可以形式化定义为：

定义 1(环形网络) 该网络中共有 N 个节点(进程)为 $\{P[i] \mid 0 \leqslant i \leqslant N-1\}$；$P[i]$节点只能同 $P[i-1 \bmod N]$和 $P[i+1 \bmod N]$节点进行直接通信($0 \leqslant i \leqslant N-1$)，例如，$P[0]$只能同 $P[1]$和 $P[N-1]$进行通信；不相邻的节点只能通过中间节点进行消息转发。

环网的特性给分布式互斥算法的实施带来很多困难。

首先，环网系统采取存贮转发通信模式和节点非直接相连的拓扑结构；而传统的分布式互斥算法通常假设所有得节点间都采用直接相连的通信模式。环网系统中，两节点间一条消息的传递，可能包括多条存储/转发消息的生灭；因此传统算法不考虑节点间距离以及采用点到点通信的方式将导致算法性能在环网系统中严重下降。例如，传统的 Maekawa 算法在构建仲裁集时，不考虑节点的物理位置，则仲裁集内部通信将激增。

其次，如前文所述，环网系统是具有静态的、整齐的网络拓扑结构，因此，节点可以通过计算获取自身位置、通信目的地距离等信息，而不用从网络中搜索。为满足最小性，传统算法通过广播/组播等主动搜索在每个节点中建立仲裁集的方法将导致系统的网络流量增加。

RNDME 算法的主要思想基于环网的两个性质构建仲裁集：

(1)环网中的每个节点至少属于一个半环。

(2)环网中的任意两个节点为中心的半环均相交。

其中第一条性质符合仲裁集的等值性；第二条符合仲裁集的相交性。

下面引入这样的例子说明 RNDME 算法的主要思想：

图 1 显示了一个具有 13 个节点的环形网络，节点编号 0-12。当节点 1 和节点 8 需要访问临界资源时，二者同时向两侧节点发送请求消息；其中节点 1 的消息穿过其仲裁集{0,12,11,2,3,4}，节点 8 的消息传过仲裁集{9,10,11,7,6,5}；两仲裁集相交于仲裁节点 11，由其判断二者的优先级，并决定进入临界区的顺序，最终完成向节点 1 发送应答等操作。注意，节点 4 和 5 尽管相邻；但是，彼此没有转发任何消息；在此处，两个仲裁集未能相交。此外，该示例中的两个请求节点位于距离较远，仅交汇于节点 11；如果距离较近，则两个仲裁集的交集将包含多个节点作为仲裁节点(交汇点取决于两个节点的距离)。

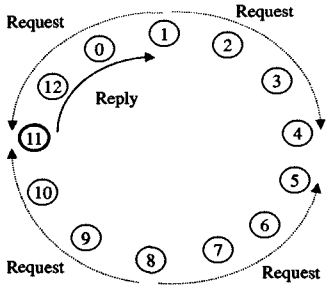

图 1 半环仲裁集算法示例

2.2 定义

本文中给出若干分布式互斥模型的定义，以便于概念化、形式化的描述算法；其中请求 R_i 的优先级定义为 $\Pr(R_i)$。

定义 1(并发，Concurrent) R_i 和 R_j 是并发的，当且仅当 R_i 在 R_j 的生存周期内产生或者 R_j 在 R_i 的生存周期内产生，这个定义是对称的[5]。

定义 2(并发集，ConcurrentSet)

$(Cset)$：$Cset_i = \{R_j \mid R_j \text{ is concurrent with } R_j\} \cup \{R_i\}$

定义 3(并发集首，ConcurrentHeader)

$(Cheader_i)$：R_i：$\Pr(R_i) > \Pr(R_j)\ \forall R_i, R_j \in Cset_k \quad i \neq j$

定义 4(并发集尾，ConcurrentTailer)

$(Ctailer_i)$：R_i：$\Pr(R_i) < \Pr(R_j)\ \forall R_i, R_j \in Cset_k \quad i \neq j$

定义 5(前驱，Pred)

$Pred\ (R, Cset_k) = R_j$ iff $R_j \in Cset_k \wedge \Pr(R_i) < \Pr(R_j) \wedge \neg(\exists R_k \in S \mid \Pr(R_i) < \Pr(R_k) < \Pr(R_j))$

定义 6(后继，Succ)

$Succ\ (R, Cset_k) = R_j$ iff $R_j \in Cset_k \wedge \Pr(R_i) > \Pr(R_j) \wedge \neg(\exists R_k \in S \mid \Pr(R_i) > \Pr(R_k) > \Pr(R_j))$

2.3 数据结构与消息

本算法中的消息的格式(具体消息的参数有所调整)如下：*msg_name*(*sender*, *receiver*, *parameter*)；*sender* 为消息发送者；*receiver* 为消息接收者；*parameter* 为消息参数列表；算法中采用的消息及其发送方式如下：

Request(i, *path*, *timestamp*)：请求消息，用于节点 P_i 向半环 *path* 上所有的节点请求进入临界区。*timestamp* 是消息携带的 Lamport 逻辑时戳；值得注意的是：该消息在多跳网络中缓存/转发，但是并不改变逻辑时戳值和转发节点计数器的值，从而保证事件排序的正确性；此外，该消息通过进程间通信发送给自身所在节点，而不通过网络。*Path* 列表用于记录应答消息转发时通过的路径。

Probe 消息，该消息参数与前者一致，用来“探索”节点所属的半环。

Reply(i, j, *Path*)：应答消息，用于 P_i 节点传递本应答给 P_j 节点。

Release(i, *half_ring*, *timstamp*)：释放消息，用于节点 P_i 向半环上所有的节点表明自己已经退出临界区，允许其他节点进入。

Failed，*Inqurie* 和 *Relinquish* 消息的同 Maekawa 算法中的一致，限于篇幅原因，不作累述；读者可以参见文献[3]。

算法采用的数据结构的命名格式为 Variable_Name$_i$，其中，i 是拥有该数据结构的节点标识。算法中具体的数据结构如下：

用于实现 Lamport 逻辑时戳的整型变量：LC_i (Local Clock)，其代表 Lamport 算法的本地计数器，MS_i 代表接收到的最大时戳，关于 Lamport 的详细算法与证明参见文献[4]。特别注意，算法中的优先级大小与 Lamport 时戳大小是反序的关系；

请求队列 RQ_i，该队列用于保存接收到的请求消息，与传统算法不同，该队列长度不固定；同时该队列中的元素按照时戳确定的优先级进行排序；

仲裁集 D_Quorum_i，该集合用于保存节点 P_i 当前使用的仲裁集包含的节点标志。

2.4 算法描述

算法的实现采用消息/事件驱动方式描述，即当进程接收到特定消息或者进程中发生特定事件时，进程调用规定过程。RNDME 算法中的过程名按照“On_EventName”形式定义，EventName 代表被触发的事件名；P_i 进程中的相应过程如下：

```
Procedure On_ReqCS; //P[i]请求进入临界区
  Send Request(i) to P[i+(N div 4) mod N ]; // Request 消息中的优先级等信息均未列出
  Send Request(i) to P[i-(N div 4) mod N ]; //算法基于双向环，
```

```
因此向两个方向发送消息
End;
……
Procedure On_TranRequest; //如果 P[i]不是最终节点,则为 P[j]转发 Request 消息
  If Not (i is Dstination) then // 是否 Destination 的判断见 On_ReqCS 过程
  {
    Insert Request(j) into RQi;
    Transfer Request(j) to Destination;
  }
  //与传统算法不同,转发之前,Request 消息将被保留下来作为仲裁判断的依据
End; //如果收到 Release 消息,将对应的 Request 消息删除后继续向下转发或终止
Procedure On_RcvRequest; // Pi 接收到 Pj 请求 Rj
  If ( Pi in Rj→path) then
  {
    LCi:=1+MAX(LCi,Rj. timestamp); //修正时戳
    Insert Rj into RQi by the order LCi;
    If Rj is the CHeader of Cseti then
      If ( Pi is the last of Rj→path) then //判断头尾
        Send Reply(i,j,Path) to Pj; //应答
      Else Transmit Rj to Next(Rj→path); //转发
  }
End;
Procedure On_RcvReply; // Pi 接收到应答消息
    If ∑Replyj== D_Quorumi then //应答组成仲裁集
      Pi enters into the CS;
End;
Procedure On_RcvRelease; // Pi 接收到释放消息
    Delete Rj from RQi
    If RQi is not NULL then //转发应答
      Transmit the Rk of the CHeader of Cseti;
End;
Procedure On_ExCS; // Pi 退出临界区
Send Releasei to the D_Quorumi; //发送释放消息
End;
```

Failed,Inqurie 和 Relinquish 的接收与发送事件定义同 Maekawa 算法一致,限于篇幅原因,不作累述;有兴趣的读者可以参见文献[4]。

当规定延迟内,*Reply* 消息未返回,请求节点释放 *Probe* 消息,探测并由残存的网络生成一个新的环网,其具体过程类似于 FDDI 的重建过程;重建之后,本算法可以继续在该网络上运行;该部分限于篇幅原因,不作累述。

2.5 证明

为保证算法的正确性,证明半环结构能够组成仲裁集。

定理 1 环网中,半环算法的请求集集合组成仲裁集。

证明:构成仲裁集的集合必须满足相交性和等职性,由上述两个过程可知,在$\{P[i]\mid 0\leqslant i\leqslant N-1\}$集合中,从属于 $P[i]$节点的请求集(设请求集集合为 $C:\{S_i\mid 0\leqslant i\leqslant N-1\}$)包括如下节点(进程):

环中的 $P[i]$节点的半环请求集为 $S_i:\{P[j]\mid i+(N \text{ div } 4) \bmod N\leqslant j\leqslant i+(N \text{ div } 4) \bmod N\}$;由此可见:$S_i$ 包含 $P[i]$节点,因此 C 中的请求集满足等职性;

由于其自身性质:$\forall i,j\ dist(P[i],P[j])\leqslant N \text{ div } 2$,另外,由半环算法定义可知,$|S_i|+|S_i|\geqslant N\geqslant|S_i\cup S_i|$,因此由鸽巢原理可知 S_i 和 S_i 至少有一个节点重合,因此,该请求集集合满足相交性。

至此,已经证明了上述集合符合构成仲裁集的必要条件;仲裁集算法可以依托这组集合正确运行。证毕。

3 性能分析与仿真

3.1 性能分析

分布式互斥算法的性能通常由消息复杂度(完成一次互斥操作的平均消息数)和时间复杂度(完成一次互斥操作所需的时间/响应延迟)度量;前者反映了算法给系统带来的负荷(信息处理和网络通信开销);后者反映了算法的时间性能。

随机分布的传统仲裁集在由于其消息发送的无序性和无复用,导致了消息发送量很大;尽管 Maekawa 仲裁集在理论上的消息复杂度为 $O(\log N)$,但实际上,由于一条请求消息单向在全双工环形网络中转发/传递/生灭,将会通过许多不属于请求节点所属的仲裁集的点,因此传统 Maekawa 算法性能在实际环形网络环境中骤降。S_i 仲裁集完成一次互斥操作所需的消息总数为 $3\times(\sum Hop(P[i],P[j])$至 $5\times(\sum Hop(P[i],P[j])$,其中 $P[j]\in S_i$;由于,环形网络的自身特性,节点 $P[i]$完成一次互斥操作所需消息数为 $3\times\sum||N-i|-|N-j||$至 $5\times\sum||N-i|-|N-j||$,其中 $P[j]\in S_i$;以 7 个节点的 Maekawa 仲裁集为例:$S_1=\{1,3,4\}$ $S_2=\{2,4,5\}$ $S_3=\{3,5,6\}$ $S_4=\{4,6,7\}$ $S_5=\{5,7,1\}$ $S_6=\{6,1,2\}$ $S_7=\{7,2,3\}$,传统算法中各仲裁集完成一次互斥操作需要的消息数均为:15~25,远远超过理想环境中的计算结果。

采用半环算法后,所需消息总数为 $3\times0.5\times N$(轻载环境下,无需进行互斥操作,因此仅需要请求-应答-释放)至 $5\times0.5\times N$(重载环境下,需要互斥操作,因此还需要在上述基础上加入互斥-取消等操作),以 7 个节点的环形网络为例,各个仲裁集 $S_1,S_2,\cdots,S_7$ 所需的消息数为 12~20;由此可见半环仲裁集所需的总消息数少于随机分布仲裁集的所需。

对于算法的时间复杂度性能分析,参考 A. W. Fu 提出的仲裁集最大延迟和平均延迟[6]。节点 s 在仲裁集群 C 中的延迟为(其中 V 是节点集合):

定义 2 最大延迟为:$\text{Max\_}Delay(C)=\max_{s\in V}\{Delay(s,C)\}$

定义 3 平均延迟为:$\text{Mean\_}Delay(C)=\frac{1}{|V|}\times\sum_{s\in V}Delay(s,C)$

最大内部延迟:上文中随机分布的仲裁集的最大延迟为 3;而半环仲裁集的最大延迟为 2。

平均内部延迟:上文中随机分布的仲裁集的平均延迟为 3;而半环仲裁集的平均延迟为 2,低于前者的平均延迟。

同时,传统 Maekawa 算法没有容错能力,当某节点崩溃时,系统必须重新计算生成仲裁集;而基于有限投影平面的仲裁集构建方法异常复杂[6],将消耗大量的计算/存储资源。而通过 Probe 消息进行半环仲裁集重构,因其算法简单易行,所以对系统性能的影响很小。

3.2 仿真结果

图 2 的仿真结果也证明了上述分析(以 7 个节点的系统为例)。其中 Maekawa 算法中的节点为随机散布。仿真试验主要度量了两种算法的消息复杂度和响应延迟。

图 2 上显示了 RNDME 算法和 Maekawa 算法的消息复杂度对比,从图中可以看出,RNDME 算法具有较低的消息复杂度,这是因为它充分考虑了环网网络中节点位置对仲裁集的影响;同时,该算法改造了请求等消息,这种改造使得原本需要点对点逐次发送仲裁集中所有的消息仅发送给临近节点即可。由于 Maekawa 算法将节点随机散布在网络中,请求节点需要逐个将消息发送给仲裁集,消息传播的多跳导致该算法的消息复杂度激增。图 2 下显示了 RNDME 算法和 Maekawa 算法的响应延迟对比。尽管传统 Maekawa 算法不需要遍历网络,但仲裁集的随机散布在整个网络中,往往导致消息在线路上冲突,并且导致节点间距离过大,使它在环网中的响应延迟反而比 RNDME 长。

(下转第 95 页)

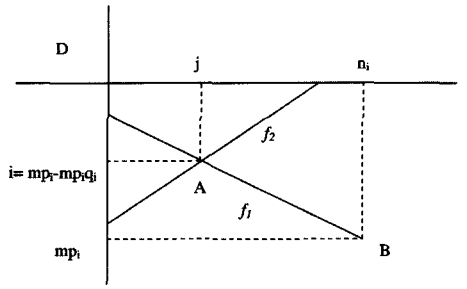

图 4　每一轮匹配过程的计算情况

在基本算法中，分段位置直接影响到算法的效率。对于一个误差上限为 k 的问题实例，假定模式串可分为 s 段，各段长度遵循分布 $D1=(p_1,\cdots,p_s)$，满足 $p_i\geqslant 1/m$，同样文本串也可分为 s(或 $s+1$)段。由算法的分段策略，对应的模式段和文本段满足后部匹配，即第 i 个模式段的后 $l(l\leqslant mp_i)$ 个字符和对应文本段的后 l 个字符应完全匹配，否则将分段。假定匹配字符在模式串中的满足分布 $D2=(q_1,\cdots,q_s)$，满足 $\sum_{i=1}^{s} mp_iq_i\geqslant m-k$ 且 $q_i\geqslant 1/(m-k)$，则 $l=mp_iq_i$。在这种设置下，算法对分段 i 所需要的计算量是 L_i。图 4 显示了算法在每一轮匹配过程中 D 表的计算情况。图 4 中 A 点表示计算 $f_2{}^{\#}$ 反对角线时第一次出现连续相同的元素 $D(i,j),D(i,j)$ 位于 $f_1{}^{\#}$ 对角线上。由此沿着 $f_1{}^{\#}$ 对角线继续计算至 B 点，B 点在 $f_1{}^{\#}$ 对角线上的下一个元素即为编辑距离增加的元素，在 B 点处分段，则 B 点元素即为 $D(mp_i,n_i)$，n_i 表示第 i 轮匹配时文本串的分段位置。由于 A,B 在同一对角线上，则有 $(j-i)=(n_i-mp_i)$，可得 $j=n_i-mp_iq_i$。至 A 点共计算了 $(i+j)$ 条反对角线 $(i+j\leqslant mp_i)$，故其总计算量 $\sum_{i=1}^{s} L_i$ 不大于 $\sum_{i+j=0}^{i+j=mp_i}(i+j)$，则可表示为 $\sum_{i=1}^{s} L_i=mp_iq_i+(1+2+\cdots+mp_i)=mp_iq_i+mp_i(mp_i+1)/2$ 个 $D(i,j)$，因此整个计算过程的总计算量为 $\sum_{i=1}^{s} L_i=m\sum_{i=1}^{s} p_iq_i+\frac{m}{2}\sum_{i=1}^{s} p_i+\frac{m^2}{2}p_i^2$，其上下限范围是 $3m/2-k+m^2/2s\leqslant \sum_{i=1}^{s} L_i\leqslant 3m/2+m^2/2$，即 $O(m^2/s)\leqslant\sum_{i=1}^{s} L_i\leqslant O(m^2)$，其中 L_i 的期望值为 $E(L_i)=(1/s)\sum_{i=1}^{s} L_i$，即每一轮的平均计算量为 $(O(m^2/s^2),O(m^2/s))$。算法所需空间为 $Max_{i=1}^{s}(L_i)$，计算可得 $O(m^2/s^2)\leqslant Max_{i=1}^{s}(L_i)\leqslant O(m^2)$。当分布 $D1$ 和 $D2$ 为均匀分布时，算法的时间复杂度和空间复杂度最低，分别为 $O(m^2/s)$ 和 $O(m^2/s^2)$，此时二维随机变量 $p_i=q_i=1/s$。若令 s 等于 $\log m$，此时算法最优的时间复杂度为 $O(m^2/\log m)$，空间复杂度为 $O(m^2/\log^2 m)$。

对于加入了回溯策略的完整算法，其复杂度相对于基本算法可能会有增加。从算法描述中可以发现，回溯的位置影响到算法效率，该位置是随机的。由于加入了回溯策略后，任一文本子串也只可能和模式子串进行一次比较，即算法只会计算一次它们之间的 D 表，因此最坏情况下的时间和空间复杂度也只等于基本的动态规划算法，为 $O(mn)$。

因此，本文算法在最好的情况下(即无需回溯的情况下)的复杂度为 $O(m^2/\log m)$，在需要多次回溯的情况下其最坏复杂度也不高于基本的动态规划概率匹配算法。形式上，我们的算法是对基本 DP-ASM 算法的改进，由于目前所有基于动态规划的字符串概率匹配算法都需要计算 D 表(或简化的 D 表)，因此和本文算法在底层思想上是一致的，任何加快 D 表计算速度的改进算法都能为我们服务。目前已知最有效的 DP-ASM 算法是计算简化的 D 表，需要 $O(km)$ 时间和 $O(m)$ 空间复杂度，通过这些简化的 D 表也能实现我们的算法策略，因此我们的算法也能达到这个效率，即本文算法能够兼容 DP-ASM 算法效率的进一步发展。

结束语　本文提出了一个改进的字符串概率匹配算法，该算法能求解模式串以不低于 p 的概率按序落入文本串的问题，并能找出这个匹配实例。该算法在最好的情况下(即无需回溯的情况下)的复杂度为 $O(m^2/\log m)$，m 为模式串长度。在需要多次回溯的情况下其最坏复杂度也不高于基本的动态规划概率匹配算法，且能利用最新概率匹配算法的改进结果以改进算法的效率。就问题的一般性来说，本文算法将在计算生物学、信息安全和信号处理等领域有广阔的应用价值。在后续的工作中我们将进一步研究该算法的应用问题，并结合最有效的基于动态规划策略的概率匹配算法提高算法效率。

参 考 文 献

[1] Galil Z, Park K. An improved algorithm for approximate string matching, SIAM J. Comput, 1999, 19(6): 989-999
[2] Navarro G. A guided tour to approximate string matching. ACM Comouting Surveys, 2001, 33(1): 31-88
[3] Chang W, Lawer E. Sublinear approximate string matching and biological applications. Algorithmica, 1994, 12(4): 327-344
[4] Ukkonen E. Algorithms for approximate string matching. Information and Control, 1985, 64: 100-118
[5] Ukkonen E. Finding approximate patterns in strings. Journal of Algorithms, 1985, 6(1): 132-137
[6] Chang W, Lampe J. Theoretical and empirical comparisons of approximate string matching algorithms // Proceedings of the 3d Annual Symposium on Combinatorial Pattern Matching (CPM '92). LNCS, vol. 644, Springer-Verlag, Berlin, 1992: 172-181

(上接第 90 页)

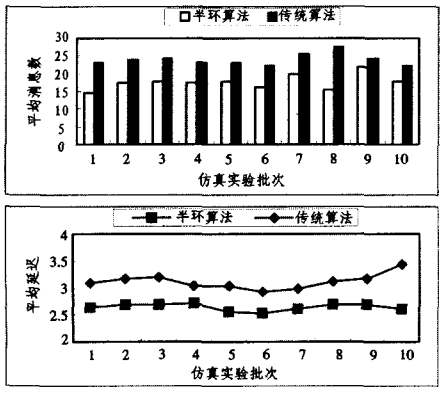

图 2　两种算法性能比较

结束语　本文根据环网的特点，提出了新型的分布式互斥算法 RNDME。该算法基于环网的半环结构生成分布式互斥仲裁集；并且用 Lamport 逻辑时戳保证消息的时序性。分析与仿真证明，和 Maekawa 算法相比，该算法具有较低的消息复杂度、较短的响应延迟和较好的容错性能。

参 考 文 献

[1] 舒继武，等. 大规模问题数据并行性能的分析[J]. 软件学报，2000, 11(5): 628-633
[2] 黄铠，徐志伟，著. 可扩展并行计算技术、结构与编程[M]. 陆鑫达，等译. 北京：机器工业出版社，2000: 145-223
[3] 尹俊文，邹鹏，等. 分布式操作系统[M]. 长沙：国防科技大学出版社，2001: 68-82
[4] Maekawa M. A logN Algorithm for Mutual Exclusion in Decentralized Systems [J]. ACM Trans. Computer Systems, 1985, 3(2): 145-159
[5] 刘丹，刘心松. 基于读写特征的分布式互斥算法 [J]. 电子学报，2004, 32(2): 326-329
[6] Fu A W. Delay-optimal Quorum Consensus for Distributed Systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 1(8): 59-69