

基于 SPN 模型的可生存性 DBMS 中 恶意事务修复算法的研究

郑吉平^{1),2)} 秦小麟^{1),2)} 钟 勇¹⁾ 孙 瑾¹⁾

¹⁾(南京航空航天大学计算机科学与技术系 南京 210016)

²⁾(南京航空航天大学信息安全研究所 南京 210016)

摘要 在传统的数据库恶意事务修复方案的基础上,采用 Petri 网模型分析事务撤销冲突和操作执行序列异常检测;进而结合可生存性 DBMS 特征提出恶意事务静态和 on-the-fly 修复算法,并在此基础上给出随机 Petri 网恶意事务修复模型;在分析恶意事务修复随机 Petri 网模型和连续时间 Markov 链的一致性后,给出了连续时间 Markov 链的恶意事务修复模型求解.

关键词 可生存性 DBMS; 随机 Petri 网; 连续时间 Markov 链; 恶意事务修复算法

中图法分类号 TP392

SPN Model based Malicious Transaction Repair Algorithms in Survivable DBMS

ZHENG Ji-Ping^{1), 2)} QIN Xiao-Lin^{1), 2)} ZHONG Yong¹⁾ SUN Jin¹⁾

¹⁾(Department of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

²⁾(Institute of Information Security, Nanjing University of Aeronautics and Astronautics, Nanjing 210016)

Abstract Malicious transaction immediate repairing is an important aspect in building survivable DBMS. Based on traditional malicious transaction repairing solutions, this paper solves transaction undo collision and incorrect executing sequence of transaction operations using Petri net models. Considering characteristics of survivable DBMS, algorithms of static and on-the-fly malicious transaction repairing are provided. Further, related repairing system models are put up based on stochastic Petri net models. After consistency analysis of malicious transaction repairing stochastic Petri net model with continuous time Markov chain, this paper provides malicious transaction repairing solutions using continuous time Markov chain.

Keywords survivable DBMS; stochastic Petri net; continuous time Markov chain; malicious transaction repairing algorithms

1 引 言

数据库安全主要关注数据的机密性、完整性和可用性^[1]. 传统数据库安全机制(如认证、推断控制、

多级安全访问控制、多级事务处理等)主要关注数据的机密性. 信息战^[2]前景下,以预防为中心的传统防御机制无法阻止所有的非法攻击,关键信息系统的可生存性^[3]已成为当前研究的热点. 数据库管理系统(DataBase Management System, DBMS)可生存

收稿日期:2006-04-04;修改稿收到日期:2006-06-01. 本课题得到江苏省高技术研究计划项目基金(BG2004-005)、航空科学基金(02F52033)资助. 郑吉平,男,1979 年生,博士研究生,主要研究方向为数据库安全、网络安全. E-mail:zhengjiping@nuaa.edu.cn. 秦小麟,男,1953 年生,教授,博士生导师,主要研究领域为安全数据库、时空数据库、GIS 等. 钟 勇,男,1970 年生,博士研究生,副教授,主要研究方向为数据库安全、网络安全. 孙 瑾,女,1978 年生,博士研究生,主要研究方向为计算机视觉、图像处理与分析.

性主要关注数据的完整性和可用性。在 DBMS 中, 事务层次的非法用户攻击以及合法用户的滥用导致在事务执行序列中存在恶意行为。恶意事务的及时隔离和修复是数据完整性和可用性的重要保证。

已有的恶意事务修复方案主要集中于事后的静态分析, 为了避免恶意事务影响的扩散, 往往终止新事务的提交和执行, 大大影响了 DBMS 的可用性。此外, 由于可生存性 DBMS 的复杂性以及状态的多样性使得对修复问题分析的难度大为增加^[4]。本文首先分析了传统的数据库恶意事务修复方案, 并在此基础上采用 Petri 网模型分析事务撤销冲突和操作执行序列检测。根据当前可生存性 DBMS 的特征, 提出恶意事务静态和 on-the-fly 修复算法, 进而给出随机 Petri 网(Stochastic Petri Net, SPN)的事务隔离和修复模型。在分析恶意事务修复 SPN 模型和连续时间 Markov 链(Continuous Time Markov Chain, CTMC)的一致性后, 给出了 CTMC 的恶意事务修复模型求解。

2 基于 Petri 网模型的传统数据库恶意事务修复解决方案

先给出数据库和事务的理论模型^[5]。

定义 1. 数据库是一组数据对象的集合, 表示为 $DB = \{x_1, x_2, \dots, x_n\}$ 。

定义 2. 一个事务 T_i 是操作与数据库对象之间的偏序关系, 满足:

(1) $T_i \subseteq \{(r_i(x), w_i(x)) | x \in DB\} \cup (a_i, c_i)$;

(2) 如果 $r_i(x), w_i(x) \in T_i$, 则 $r_i(x) <_i w_i(x)$ 或者 $w_i(x) <_i r_i(x)$;

(3) 如果 $c_i \notin T_i$ 则 $a_i \in T_i$,

其中 r, w, a, c 分别代表读、写、中断退出和提交操作。

定义 3. $T = \{T_1, T_2, \dots, T_n\}$ 表示一组事务集合, 一个事务执行历史 H 是 T 上的偏序关系 $<_H$, 满足:

(1) $H = \bigcup_{i=1}^n T_i$;

(2) $\bigcup_{i=1}^n <_i \subseteq <_H$.

2.1 基于 Petri 网模型的事务撤销冲突解决方案

传统恶意事务修复方案是对恶意事务及其感染事务进行撤销或者补偿操作。图 1 所示为两个恶意事务的修复序列^[6]: $H_1 = w_{T_i}(x)w_{T_j}(x)w_{U_i}(x)w_{U_j}(x)$ 。 t_1 和 t_2 时刻事务 T_i, T_j 分别对数据 x 进行写操作。 t_2 时刻后, 系统发现 T_i 和 T_j 是恶意事务或者受感染事

务, 因而在 t_3, t_4 时刻分别对事务 T_i, T_j 进行撤销操作。然而 t_4 时刻后, 数据 x 修复到 t_1 和 t_2 之间数据的状态, 此时数据 x 是恶意事务 T_i 篡改后的状态, 而不是 t_1 时刻之前数据未受感染之前的状态, 即出现多个事务撤销引发的数据不一致现象。因此修复序列 H_1 存在事务撤销冲突。

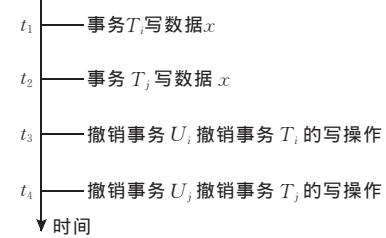


图 1 违背数据一致性的事务撤销/补偿操作序列

定义 4. 恶意事务撤销冲突避免 Petri 网(Malicious Transaction Undo Avoiding Petri Net, MTUA-PN)^[7,8]是一个四元组, $MTUA-PN = (P, T; F, M_0)$ 。

(1) $P = \{p_1, p_2, \dots, p_m\}, T = \{t_1, t_2, \dots, t_n\}$ 分别表示有穷个位置和变迁, 并且 $P \cup T \neq \emptyset$ (非空性), $P \cap T = \emptyset$ (二元性);

(2) $F \subseteq (P \times T) \cup (T \times P)$ 表示 MTUA-PN 中的弧集, 即网中流动关系仅存在于位置与变迁之间;

(3) $M_0: p \rightarrow \{0, 1\}$ 是 MTUA-PN 的初始标识, If $p \rightarrow 1$ 则位置 p 可实施;

(4) $\Gamma = M_0 t_1 M_1 t_2 \dots t_k M_k$ 是 MTUA-PN 的有穷状态实施序列, 其中 $t_1 t_2 \dots t_k$ 为变迁序列。

根据定义 4, 图 2 所示 MTUA-PN 网可以解决执行序列 H_1 的撤销冲突问题。图 2 中, 当 T_i, x 或 T_j, x 位置同时具有标识时 U_i 或 U_j 才可实施, 因此允许的执行序列为(忽略位置 x): $H'_1 = T_i U_i T_j U_j$ 或 $H''_1 = T_j U_j T_i U_i$ 。因而避免了恶意事务撤销时的冲突, 并简化了问题的求解。

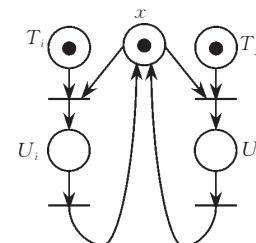


图 2 执行序列 H_1 的 MTUA-PN 冲突避免方案

2.2 基于数据依赖的恶意事务修复 Petri 网解决方案

传统的数据库修复方案中, 针对鉴别出的恶意事务, 首先撤销恶意事务以及依赖的受感染事务, 进

而重新执行受感染的事务。但实际情况下,受感染的恶意事务中只有部分数据依赖的操作需要重新执行^[9,10],受感染事务中无辜的操作无需重新执行,这样可以大大提高修复算法的效率和节省恶意事务修复时间。通过定义写操作链表及相应 Petri 网模型可以解决基于数据依赖的恶意事务恢复问题。

定义 5. 事务上数据写操作链表 $WC(T)$ 是一组偏序关系:

- (1) $\{x_1, x_2, \dots, x_k\}$ 是事务集上的数据集合;
- (2) $>$ 是 $\{x_1, x_2, \dots, x_k\}$ 上的偏序关系,如果 $x_i > x_j$ 则对数据 x_i 的更新操作导致数据 x_j 的更新,同时数据 x_j 的更新是由 x_i 的更新操作所引起;
- (3) $WC: x_1 > x_2 > \dots > x_k$ 为数据集 $\{x_1, x_2, \dots, x_k\}$ 的写操作链表。

定义 5 中的写操作链表可由事务(集)的读集、预写集和后写集^[11]构造。假设给定 3 个事务写操作链表:

$$WC_1: x_1 > x_2 > x_3 > x_4;$$

$$WC_2: x_2 > x_6 > x_7 > x_8;$$

$$WC_3: x_3 > x_5,$$

构造基于数据依赖的恶意事务修复 Petri 网模型^[7,11]如图 3 所示。

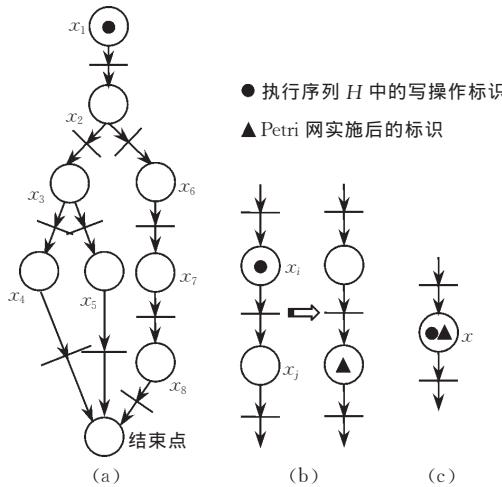


图 3 基于数据依赖的恶意事务修复 Petri 网模型

图 3(a) 表示根据写操作链表 WC_1, WC_2, WC_3 构造的 Petri 网模型,根据执行序列 H ,如果出现相应的写操作,则将相应的位置中增加写操作标识“●”;当网实施时,位置 x_i 减少标识“●”,同时位置 x_j 增加实施后标识“▲”,其中 $x_i > x_j$,如图 3(b) 所示;当位置 x 已经存在写操作标识“●”,由于 Petri 网不正当实施后增加标识“▲”,则出现写操作冲突,如图 3(c) 所示。根据基于数据依赖的恶意事

务修复 Petri 网模型,如果出现图 3(c) 所示的情形,说明系统检测到隐藏的恶意事务活动,应对恶意进行修复,同时事务执行序列中应避免这种情况的发生。

3 基于 Petri 网模型的可生存性 DBMS 中恶意事务修复算法

在 DBMS 中,事务通常按照图 4 所示的顺序执行^[12]。首先通过相关检测机制(例如入侵检测机制)判断事务是否是正常事务。根据事务性质,正常事务,执行;恶意事务,采用隔离机制,隔离恶意事务以及受感染的事务,根据事务或数据之间的依赖关系对相关数据进行修复并反馈。

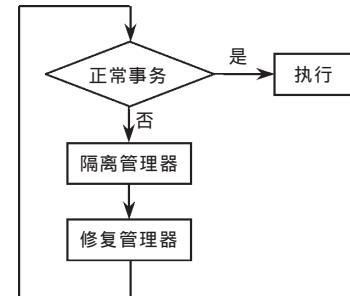


图 4 事务执行流程图

3.1 可生存性 DBMS 中静态恶意事务修复算法

3.1.1 静态恶意事务修复算法

当 DBMS 面临攻击或者合法用户的滥用时, DBMS 入侵检测系统检测违反安全策略的恶意用户行为(即恶意事务)时,必须首先对其隔离并采用相关修复算法对恶意事务及其感染的事务进行修复。根据已知恶意事务的位置以及执行序列 H ,可以构造可生存性 DBMS 中静态恶意事务修复算法^[5,13]。

算法 1. 静态恶意事务修复算法。

输入: 执行序列 H 、恶意事务集合 $B = \{B_1, B_2, \dots, B_m\}$, 其中 B_1 是第一个恶意事务, 依次类推

输出: DBMS 中数据一致性状态

初始化: 写数据操作集合 $w_set = \emptyset$; 临时写数据操作集合 $tw_set = \emptyset$;

撤销事务集合 $ut_set = B$; 临时撤销事务集合 $tut_set = \emptyset$.

算法步骤:

1. 定位 B_i 在执行序列 H 中的位置;
2. 扫描执行序列 H 中每一项读写操作 $op(x)$ 直到序列结束:

2.1. If $op_{T_i}(x) \& T_i \in B$ Then

If $w_{T_i}(x)$ Then $w_set = w_set \cup \{w_{T_i}(x)\}$;

2.2. Else

```

If  $w_{T_i}(x)$  Then  $tw\_set = tw\_set \cup \{w_{T_i}(x)\}$ ;
If  $r_{T_i}(x)$  Then
  If  $T_i \in tut\_set$  skip;
  Else If  $op(x) \in w\_set$  Then  $tut\_set = tut\_set \cup \{T_i\}$ ;
If  $a_{T_i}$  Then
   $tw\_set = tw\_set - tw\_set | T_i$ ;
  If  $T_i \in tut\_set$  Then  $tut\_set = tut\_set - \{T_i\}$ ;
If  $c_{T_i}$  Then
  If  $T_i \in tut\_set$  then
     $ut\_set = ut\_set \cup \{T_i\}$ ;
     $tut\_set = tut\_set - \{T_i\}$ ;
     $w\_set = w\_set \cup tw\_set | T_i$ ;
     $tw\_set = tw\_set - tw\_set | T_i$ ;
  Else  $tw\_set = tw\_set - tw\_set | T_i$ ;
3. 撤销  $ut\_set$  每一个事务操作;
4. 重复上述操作直至恶意事务集合  $B = \emptyset$ .

```

算法 1 中, $tw_set | T_i$ 表示 tw_set 中有关 T_i 操作集合. 考虑执行序列:

$$\begin{aligned}
H_2 : & r_B(x) w_B(x) r_B(u) w_B(u) c_{B,r_{T_1}}(x) w_{T_1}(x) r_{T_3}(z) \\
& w_{T_3}(z) c_{T_3} r_{T_1}(y) w_{T_1}(y) c_{T_1} r_{T_2}(y) w_{T_2}(y) r_{T_2}(v) \\
& w_{T_2}(v) a_{T_2} r_{T_4}(u) w_{T_4}(u) r_{T_4}(y) w_{T_4}(y) r_{T_4}(z) \\
& w_{T_4}(z) c_{T_4},
\end{aligned}$$

其中 B 为恶意事务, 按照算法 1 中步骤对 H_2 依次扫描后相应的集合变化如下所示.

写数据操作集合:

$$\begin{aligned}
w_set = \emptyset \rightarrow & \{w_B(x)\} \rightarrow \{w_B(x), w_B(u)\} \rightarrow \{w_B(x), \\
& w_B(u), w_{T_1}(y), w_{T_1}(x)\} \rightarrow \{w_B(x), \\
& w_B(u), w_{T_1}(y), w_{T_1}(x), w_{T_4}(u), w_{T_4}(y), \\
& w_{T_4}(z)\};
\end{aligned}$$

临时写数据操作集合:

$$\begin{aligned}
tw_set = \emptyset \rightarrow & \{w_{T_1}(x)\} \rightarrow \{w_{T_1}(x), w_{T_3}(z)\} \rightarrow \\
& \{w_{T_1}(x)\} \rightarrow \{w_{T_1}(x), w_{T_1}(y)\} \rightarrow \emptyset \rightarrow \\
& \{w_{T_2}(y)\} \rightarrow \{w_{T_2}(y), w_{T_2}(v)\} \rightarrow \emptyset;
\end{aligned}$$

撤销事务集合: $ut_set = B \rightarrow \{B, T_1\} \rightarrow \{B, T_1, T_4\}$;

临时撤销事务集合: $tut_set = \emptyset \rightarrow \{T_1\} \rightarrow \emptyset \rightarrow \{T_2\} \rightarrow \emptyset \rightarrow \{T_4\} \rightarrow \emptyset$.

算法最后针对撤销事务集合按照相应的写数据操作集合进行撤销或者补偿操作. 算法 1 中 tw_set 存放临时写数据操作集合, 当一个正常事务 T 在 H 的后续执行序列中发现是恶意事务或被恶意事务感染则此事务加入临时撤销事务集合 tut_set 中, 相关操作在内存中进行. 而传统做法是当发现 T 为恶意事务后重新扫描执行序列 H 以发现相关受恶意或

受感染操作, 需要进行相关磁盘操作. 较传统方法, 算法 1 中定义的数据结构大大提高恶意算法的修复效率. 同时, 算法 1 引入临时撤销事务集合 tut_set 可以避免因事务中断操作而无须撤销的情形.

3.1.2 M/M/1 队列的静态恶意事务修复

SPN 模型

定义 6. 静态恶意事务修复随机 Petri 网 (Static Malicious Transaction Repairing Stochastic Petri Net, SMTR-SPN) 是一个五元组, $SMTR-SPN = (P, T, F, M_0, \lambda)$, 其中 P, T, F, M_0 的意义同定义 4, $\lambda = \{\lambda_1, \lambda_2, \dots, \lambda_k\}$ 是在调度管理、修复管理以及修复算法执行等情况下变迁平均实施速率集合.

图 5 是 $M/M/1$ 队列的静态恶意事务修复 SPN 模型^[14]. 事务调度产生执行序列所用时间、修复管理器的空闲资源等待时间以及静态恶意事务修复算法花费时间构成 SPN 模型变迁的平均实施速率集合 $\lambda = \{\lambda_1, \lambda_2, \lambda_3\}$. 模型直观反映了恶意事务的修复流程.

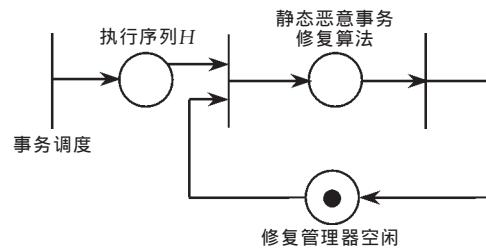


图 5 $M/M/1$ 队列的静态恶意事务修复 SPN 模型

3.2 基于 SPN 模型的 on-the-fly 恶意事务修复算法

3.2.1 On-the-fly 恶意事务修复算法

静态恶意事务修复算法对发现的恶意事务及其受感染的事务进行修复, 在遭受攻击或入侵时可以保障 DBMS 最低限度运行并恢复到正常状态. 然而, 静态恶意事务修复算法的执行过程中, 系统阻止新事务的提交和执行, 降低了 DBMS 系统的可用性. 同时, 静态恶意事务修复算法在可生存性要求较高的场合(如银行、空中交通管制等场景)下显得无能为力. On-the-fly 恶意事务修复算法在恶意事务修复过程中, 允许新事务的继续执行, 因而大大提高了 DBMS 的可用性. 根据已知恶意事务的位置、执行序列 H 以及算法阶段性终止条件, 构造可生存性 DBMS 中 on-the-fly 恶意事务修复算法^[5,12,15,16].

算法 2. On-the-fly 恶意事务修复算法.

输入: 执行序列 H 、恶意事务集合 $B = \{B_1, B_2, \dots, B_m\}$, 其中 B_1 是第一个恶意事务, 依次类推

输出: 修复算法阶段性终止的 DBMS 中数据一致性状态
 初始化: 受感染数据集合 $d_set = \emptyset$; 清除感染后数据
 集合 $c_set = \emptyset$;
 临时数据集合 $t_set = \emptyset$; 临时撤销事务集合
 $tut_set = \emptyset$.

算法步骤:

1. 定位 B_i 在执行序列 H 中的位置;
2. 重复以下操作直到修复算法阶段性终止:
 扫描执行序列 H 中操作 $op(x)$:
 If $op_{T_i}(x) \& (T_i \rightarrow undo_transaction)$ Then 忽略;
 Else If $op_{T_i} \& T_i \in B$ Then
 If $w_{T_i}(x) \& x \notin c_set$ Then $d_set = d_set \cup \{x\}$;
 If a_{T_i} Then 按照定义 4 进行 T_i 的撤销操作;
 Else If $w_{T_i}(x)$ Then
 If $x \notin c_set$ Then $t_set = t_set \cup \{x\}$;
 Else If $w.HSN > x.HSN$ Then
 $\{c_set = c_set - \{x\};$
 $t_set = t_set \cup \{x\}\}$
 If $r_{T_i}(x)$ Then
 If $(x \in d_set | x \in c_set) \& r.HSN \leq x.HSN$ Then
 $tut_set = tut_set \cup \{T_i\}$;
 If a_{T_i} Then $t_set = t_set - t_set | T_i$;
 If $T_i \in tut_set$ Then $tut_set = tut_set - \{T_i\}$;
 If c_{T_i} Then
 If $T_i \in tut_set$ Then
 $\{d_set = d_set \cup t_set | T_i;$
 $t_set = t_set - t_set | T_i;$
 按照定义 4 策略进行 T_i 的撤销操作;
 Else $t_set = t_set - t_set | T_i$;
3. 修复算法阶段性终止;
4. 重复上述操作直至恶意事务集合 $B = \emptyset$.

算法 2 中, $t_set | T_i$ 表示 t_set 中有关 T_i 操作集合, $w.HSN$ 表示操作的执行序列编号 (History Serial Number, HSN), $w.HSN > x.HSN$ 表示数据 x 受写操作 w 影响。修复算法的阶段性终止是指恶意事务及其影响被修复, 考虑执行序列

$$H_3 = r_B(x)w_B(x_1)c_B \dots r_{T_1}(x_1)w_{T_1}(x_2) \dots \\ r_{T_k}(x_k)w_{T_k}(x_k) \dots,$$

即使数据 x_k 被清除, 数据 x_{k+1} 也被感染, 因而修复算法在事务提交结束前无法停止。除上述情况外, 如果数据库服务器处于“忙”状态或者待修复的事务复杂的情况下, 修复算法也可能无法终止。通常, 修复算法由以下几个方面的因素决定:(1) 数据库的运行效率, 包括新事务到达的数量以及软硬件资源数量等;(2) 修复算法的效率;(3) 事务的性质, 如恶意长生事务的修复比平坦事务需要更多时间。

算法 2 的修复流程类似于算法 1, 由于允许新

事务的继续运行, 清除感染后的数据集有再次被感染的可能, 因而引入清除感染后的数据集合 c_set : 当一个已被修复的数据 x 再次受到感染, 修复算法依然保证恶意事务的影响能够被修复。

3.2.2 On-the-fly 恶意修复算法的 SPN 模型

定义 7. On-the-fly 恶意事务修复随机 Petri 网 (Dynamic Malicious Transaction Repairing Stochastic Petri Net, DMTR-SPN) 是一个六元组, $DMTR-SPN = (P, T, F, W, M_0, \lambda)$, 其中 P, T, F, M_0, λ 的意义同定义 6, W 为隔离服务器和修复管理器中空闲的资源。

图 6 为 $M/M/m_n/\dots/m_1/r$ 队列的 on-the-fly 恶意事务修复 SPN 模型^[14,17]。针对受影响的事务进行多阶段隔离, 第 n 阶段隔离后将对恶意事务进行修复。整个修复系统的效率主要由修复管理器和隔离管理器的资源以及 on-the-fly 修复算法的效率决定:当系统资源受限时, 则修复过程降低标准运行; 同时, 当修复算法遇到大量恶意事务或者由于执行序列数据依赖造成破坏迅速扩散, 系统修复过程也将变得缓慢。图 6 所示模型直观地反映了系统对恶意事务 on-the-fly 的修复流程。

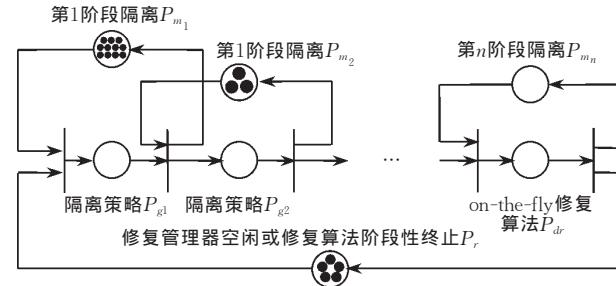


图 6 $M/M/m_n/\dots/m_1/r$ 队列的 on-the-fly 恶意事务修复 SPN 模型

4 恶意事务修复 SPN 模型的 CTMC 求解

4.1 恶意事务修复 SPN 模型与 CTMC 的一致性

在可生存性 DBMS 中, on-the-fly 修复算法采用多阶段隔离方案, 修复以及事务的隔离需要时间, 即隔离管理器以及修复管理器有时间延迟, 因而相应的变迁实施有一定的时间延迟。假定变迁的延迟服从指数分布, 则有^[7]:

$S(x) = 1 - e^{-\lambda_i x}$ (其中 λ_i 是变迁 T_i 的实施速率)
 则隔离或修复管理器的平均延迟为

$$\bar{d}_i = \int_0^{\infty} [1 - S(x)] dx = \int_0^{\infty} e^{-\lambda_i x} dx = \frac{1}{\lambda_i}.$$

定义 8. 两个随机转换系统是同构的当且仅当下列条件成立^[14]：

(1) 在两个系统的状态空间之间存在一个一对一满射函数 F ；

(2) 在一个系统中存在一个状态转换 $S_i \rightarrow S_j$ iff 在另一个系统中存在一个状态转换 $F(S_i) \rightarrow F(S_j)$ ；

(3) 对任意状态，概率 $P[S_i \rightarrow S_j, \tau] = P[F(S_i) \rightarrow F(S_j), \tau]$.

定理 1. On-the-fly 恶意事务修复 SPN 同构于一个 CTMC.

说明：根据所构建的恶意事务修复 SPN，很容易依据网的初始标识构建 SPN 的可达图，同时 on-the-fly 恶意事务修复算法阶段性终止或终止决定网具有有穷个位置和状态。根据所构建的 SPN 可达图，将其每条弧上标注的实施变迁 t_i 换成其平均实施速率 λ_i （或与标识相关 λ_i 的函数），根据延迟的指

	P_{m1}	P_{m2}	P_r	P_{g1}	P_{dr}
M_0	11	3	5	0	0
M_1	6	3	0	5	0
M_2	9	0	0	2	3
M_3	9	3	3	2	0
M_4	6	3	0	5	0

图 7 $M/M/11/3/5$ 队列 DMTR-SPN 的 M-P 矩阵和 CTMC

在已知恶意事务修复模型等价的 CTMC 后，可以计算出模型相关评价指标，如可达标识稳定状态概率等。通过分析可知，SPN 模型的变迁实施速率的指数分布所导致的无记忆性和标识的可数性是构造 SPN 可达图和 MC 之间同构的关键因素。

5 结 论

恶意事务的及时恢复是构建可生存性 DBMS 的重要保证。本文将 Petri 模型应用到恶意事务恢复算法中，并且根据可生存性 DBMS 特征提出恶意事务静态和 on-the-fly 修复算法。相比于传统算法解决方案，新算法充分考虑到恶意事务扩散的影响以及采用相关数据结构避免磁盘操作，因而新算法具有较高的执行效率并节省了修复时间。由于应用需求越来越大，当前 DBMS 系统变得越来越复杂并状态多样性，采用 SPN 模型可以模拟 DBMS 状态的改变，为可生存性 DBMS 的分析提供了坚实的理论基础。

数分布特征以及定义 8 所给定的条件，容易求得相应的 CTMC。定理 1 的证明原理类似于文献[18]。

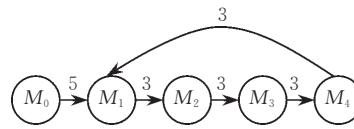
4.2 恶意事务修复 CTMC 求解

对于恶意事务修复模型，可以通过以下步骤构造相应的 CTMC：

(1) 根据定义 6，构建恶意事务修复 SPN 模型；

(2) 由初始标识 M 与位置 P 之间的对应关系，即 SPN 的变迁实施速率同构于一个 MC 状态空间，构成 M-P 矩阵和 Markov 链(Markov Chain, MC)^[19]。

在恶意事务修复模型中，算法的执行在一定条件下结束或阶段性终止，因此相应的 SPN 模型具有有穷个位置和变迁，相应的 MC 即为 CTMC。以两阶段恶意事务修复 $M/M/11/3/5$ 模型为例，其中 $m_1=11, m_2=3, r=5$ 。相应恶意事务修复模型的 M-P 矩阵和 CTMC 如图 7 所示。



参 考 文 献

- Bertino E., Sandhu R.. Database security-concepts, approaches, and challenges. IEEE Transactions on Dependable and Secure Computing, 2005, 2(1): 2~19
- Ammann P., Jajodia S., McCollum C. D., Blaustein B. T.. Surviving information warfare attacks on databases. In: Proceedings of the IEEE Symposium on Research in Security and Privacy, Oakland, California, 1997, 164~174
- Knight J., Sullivan K., Elder M., Wang C.. Survivability architectures: Issues and approaches. In: Proceedings of the 2000 DARPA Information Survivability Conference & Exposition, Los Alamitos, California, 2000, 157~171
- Madan B. B., Trivedi K. S.. Security modeling and quantification of intrusion tolerant systems using attack-response graph. High Speed Networks, 2004, 13(4): 297~308
- Bai K., Wang H., Liu P.. Towards database firewalls. In: Proceedings of the IFTP International Federation for Information, 2005, 178~192
- Ammann P., Jajodia S., Liu P.. Recovery from malicious transactions. IEEE Transactions on Knowledge and Data Engineering, 2002, 14(5): 1167~1185

- 7 Murata T.. Petri nets: Properties, analysis, and applications. In: Proceedings of the IEEE, 1989, 77(4): 541~580
- 8 Zurawski R., Zhou M. C.. Petri nets and industrial applications: A tutorial. IEEE Transactions on Industrial Electronics, 1994, 41(6): 567~583
- 9 Panda B., Haque K. A.. Extended data dependency approach: A robust way of rebuilding database. In: Proceedings of the 2002 ACM Symposium on Applied Computing, New York, 2000, 446~452
- 10 Panda B., Tripathy S.. Data dependency based logging for defensive information warfare. In: Proceedings of the 2000 ACM Symposium on Applied Computing, New York, 2000, 361~365
- 11 Hu Y., Panda B.. Identification of malicious transactions in Database Systems. In: Proceedings of the 7th International Database Engineering and Applications Symposium, 2003, 329~335
- 12 Liu P., Jing J., Luenam P., Wang Y., Li L., Ingsriswang S.. The design and implementation of a self-healing database system. Journal of Intelligent Information Systems, 2004, 23(3): 247~269
- 13 Bernstein P. A., Hadzilacos V., Goodman N.. Concurrency Control and Recovery in Database Systems. Addison-Wesley, 1987
- 14 Lin Chuang. Stochastic Petri Net and System Performance Evaluation. 2nd Edition. Beijing: Tsinghua University Press, 2005(in Chinese)
- (林 闯. 随机 Petri 网和系统性能评价. 第 2 版. 北京: 清华大学出版社, 2005)
- 15 Liu P.. Architectures for intrusion tolerant database systems. In: Proceedings of the 18th Annual Computer Security Applications Conference, 2002, 311~320
- 16 Luenam P., Liu P.. ODAM: An on-the-fly damage assessment and repair system for commercial database applications. In: Proceedings of the 15th IFIP WG 11.3 Working Conference on Database and Application Security, Ontario, Canada, 2001, 239~252
- 17 Liu P., Jajodia S.. Multi-phase damage confinement in database systems for intrusion tolerance. In: Proceedings of the 14th IEEE Computer Security Foundations Workshop, 2001, 191~205
- 18 Dugan J. B., Trivedi K. S., Geist R. M., Nicola V. F.. Extended stochastic Petri nets: Applications and analysis[Ph. D. dissertation]. Department of Electrical Engineering, Duke University, 1984
- 19 Miner A. S.. Computing response time distributions using stochastic Petri nets and matrix diagrams. In: Proceedings of the 10th International Workshop on Petri Nets and Performance Models, Urbana-Champaign, IL, USA, 2003, 10~19



ZHENG Ji-Ping, born in 1979, Ph. D. candidate. His current research interests include database security and network security.

QIN Xiao-Lin, born in 1953, professor and Ph. D. supervisor. His current research interests include secure data-

Background

This research is supported by High-Technology Research of Jiangsu Province of China under grant No. BG2004005 and the Aerospace Science Foundation of China under grant No. 02F52033.

Database security is a very important aspect of information security, especially for some critical information systems. Traditional database systems rely on preventive controls and are very limited in surviving malicious attacks. On the other hand, for the complexity and multi-state of current database systems, it is difficult to analyze such systems via experimentation or simulation. Instead, analytical modeling

base, spatial database, spatio-temporal database and geographical information system.

ZHONG Yong, born in 1970, assistant professor and Ph. D. candidate. His current research interests include database security and network security.

SUN Jin, born in 1978, Ph. D. candidate. Her current research interests include computer vision and image process and analysis.

techniques provide an effective means to study their security behavior. In this paper, the authors introduce Petri net models to analyze and solve malicious transaction repairing problems in traditional and survivable database systems. Also, algorithms of static and on-the-fly malicious transaction repairing and related repairing system models are proposed. Finally, after consistency of malicious transaction repairing stochastic Petri net model and continuous time Markov chain is analyzed, malicious transaction repairing solution using continuous time Markov chain is provided.