2012 International Conference on Information and Computer Applications (ICICA 2012)
IPCSIT vol. 24 (2012) © (2012) IACSIT Press, Singapore

Modeling a Concurrency Control Mechanism in Distributed Databases using Hierarchical Colored Petri Net

Saeid Pashazadeh

Department of Information Technology

Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran

E-mail: pashazadeh@tabrizu.ac.ir

Abstract. Formal methods are widely used for studying and verifying characteristics of concurrency control mechanisms (CCMs) and protocols in distributed databases. Colored Petri net (CPN) has high modeling capabilities and is one of the best methods for formal analysis and verification of CCMs. In this paper, a novel model of CCM based on two phase locking (2PL) using CPN has presented. State space analysis of model permits us to prove that all schedules of concurrent execution of transactions using 2PL in a case study is deadlock free nor not. Then a simple case study along with its state space analysis has presented. Results show that CPN is proper method for modeling CCM using 2PL and proving deadlock freeness property of all schedules of transactions.

Keywords: Concurrency control mechanism, Verification, Colored Petri net, State space analysis, Two phase locking

1. Introduction

Concurrent execution of transactions in distributed databases faces with a lot of problems. CCM is used for isolation and noninterference execution among conflicting transactions to preserve database consistency through consistency preserving execution of transactions. Access of concurrent trasactions to shared data requires locking. But simple locking and unlocking of shared data do not guarantee the serializability of transactions [1]. 2PL protocol is one of the well known locking protocols that enforce conflict serializabilty. 2PL has two phases, locking and unlocking phases. In 2PL scheduling all locking operations of transactions must precede their first unlock operation. But 2PL may cause the deadlock. Conservative 2PL is one of the extensions of 2PL which prevents deadlock in expense of decreasing the concurrency level. Formal modeling of CCMs is useful in studying different characteristics of them.

2. Related Works

Petri nets are formal methods that benefits from easy graphical user interface. Using extended place/transition nets is done for performance evaluation study on distributed database CCMs [2]. Modeling two phase commit protocol for transaction management in distributed environment is studied using time Petri net [3]. Formal specification for concurrency control of database transactions using conventional Petri net based on 2PL protocol which can ensure the serialization of concurrency scheduling is also studied [4]. Quantitative performance study of 2PL in parallel database systems is performed using a novel simulation-based methodology [5]. Standard 2PL and the primary-copy methods are modeled using high level Petri net (colored Petri net) too [6]. In this paper a new model of concurrency control based on the 2PL is introduced using colored Petri net.

3. Colour Sets, Initial Markings and Models of The System

Fig. 1 shows the top level model of the system and Fig. 3 shows the model of each transaction. Colour sets that are used in the models are as follows:

```
colset   RESOURCE = string;
colset   SEQUENCE = int;
colset   ACCESS = string;
colset   LOCK = string;
val      TransactionsNO = 3;
colset   TRANSACTION = index
     Trans with 1.. TransactionsNO;
colset   BOOLEAN = bool

colset   SEQxACCESSxRES =
         product SEQUENCE *
         ACCESS *RESOURCE;
colset   TRANSLIST = list
         TRANSACTION;
colset   TRANSLISTxLOCKxRES
         = product TRANSLIST*
         LOCK*RESOURCE;
```

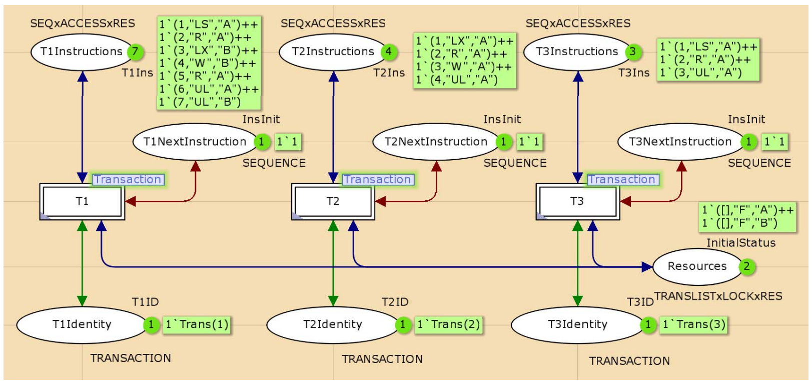

Fig.1 Top-level module of the hierarchical resource management model.

4. Description of Model's functions

Most of the functionality of the CPN's models is based on the user written functions in functional ML language. Model's functions and their operation is as described follows. Fig. 2 shows the structure chart of model's functions.

Function *getTransIndex* takes a transaction identity of colset TRANSACTIAON as first parameter and a list of transcations with colset TRANSLIST as second parameter and returns the index of transaction's position in the list (counting from 0). If list do not contains this transaction, then function returns -1 as result.

Function *eliminate*Trans takes a transaction identity t of colset TRANSACTION as first parameter and a list of transactions' identities h::L of colset TRANSLIST as second parameter. If the transaction in first parameter exists in the list of transactions in the second parameter, then function returns a list that occurs from eliminating transaction t from the list. In otherwise returns the unchanged list of transactions.

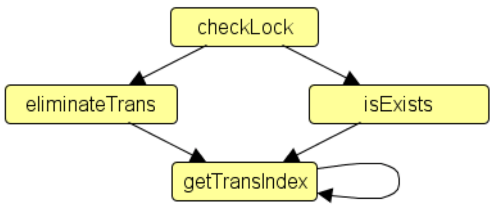

Fig.2 Structure chart of model's functions.

Function *isExists* takes a transaction identity of colset TRANSACTION as first parameter and a list of transactions' identities of colset TRANSLIST as second parameter. This function returns true if the first parameter is exists in the list of second parameter. In otherwise this function returns false.

Function *checkLock* is the most important function of the model that is shown in Fig. 3. First formal parameter of it, RAcc is of colset ACCESS and represents the required access on a resource by current instruction. Second parameter ELock is of colset LOCK and represents the current existing lock on the resource that current instruction wants to have RAcc access on it. Third parameter CurT is of colset TRANSACTION and represents the identity of transaction that current instruction of under study belongs to it. Fourth parameter TList is of colset TRANSLIST and represents the list of transactions that have lock of type ELock on the resource that will be used by current instruction.

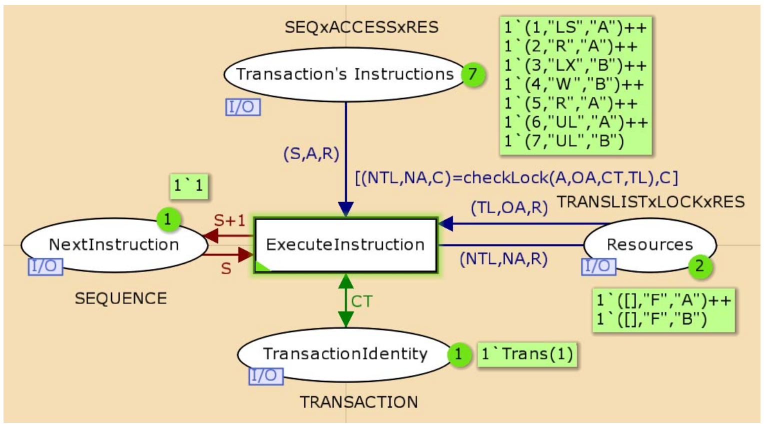

Fig.3 CPN model of transaction module instance corresponding to transaction T1.

This function returns a record (with three fields) as the result. Third field of the result is of colset BOOLEAN. Function returns true for the third parameter when the current instruction can execute based on the locks compatibility rules. Function return false in two different cases. First case is when the current instruction is not permitted to execute based on the existing locks on the resource and lock compatibility rules. After execution of some other instructions may be this instruction can be executed. In this case, second filed of output result which is of colset LOCK, will represent the final lock type on the resource after executing the current instruction. Second case occurs when the instruction has conflict based on the 2PL algorithm. E.x. if a transaction wants to have write operation on a resource before applying an exclusive lock on it. In latter case, the second field of output record contains the error message for better identifying the errors of transactions' instructions for easier trace of the model in execution time. First field of the result is of colset TRANSLIST and represents the list of transactions that have lock on the resource after successful execution of current instruction.

Tab 1. Operation of system when current requesting transaction exists in the list of transactions that have lock on the resource.

		Existing Lock	
		Exclusive Lock	***Shared Lock***
Required Access	***Exclusive Lock***	**(TList, ELock, false)**	**(TList, ELock, false)**
	Shared Lock	**(TList, ELock, false)**	(CurT::TList, ELock, true)
	Unlock	*(TList,"Unlocks No Lock Error",false)*	
	Read	*(TList, "No Lock Error", false)*	
	Write	*(TList, "No Lock Error", false)*	

Tab 2. Operation of system when current requesting transaction does not exists in the list of transactions that have any lock on the resource

		Existing Lock	
		No Lock	***else***
Required Access	***Exclusive Lock***	(CurT::TList, RAcc, true)	*(TList,"Unknown Old Access Error",false)*
	Shared Lock	(CurT::TList, RAcc, true)	*(TList,"Unknown Old Access Error",false)*
	Unlock	*(TList,"Unlocks No Lock Error",false)*	-
	Read	(TList, ELock, true)	-
	Write	*(TList, "No X Lock Error", false)*	-

Operation of model by means of function *checkLock* is summarized in Tab. 1, Tab. 2, and Tab. 3. Tab. 1 represents the operation of the model when transaction CurT requires access RAcc on a resource which transactions of list TList have lock ELock on it and transaction CurT exists in the list of TList. Tab. 2 represents the cases which transaction CurT do not exists in the list TList.

Tab. 3 represents the cases which no lock exists on the resource or any unknown lock is presented by mistake on the resource. If RAcc is beyond the range of predefined accesses the function returns record (TList,"unknown Access",false) as result.

My proposed model presents a simple case study that contains three transaction and two types of resources. State space of model has 102 nodes and 174 arcs. Deadlock of petri net will appear as a node with no outgoing arc. State space analysis shows that the perti net have a single deadlock state. This state is the deadlock state of the petri net and is not deadlock state of the system. This state shows that all schedules of tarnsaction can run and finish deadlock free.

Tab 3. Operation of system when no lock exists on the resource or in appearance of unknown lock.

		Existing Lock	
		Exclusive Lock	***Shared Lock***
Required Access	***Exclusive Lock***	*(TList, "Duplicate LX Error", false)*	If length(TList) > 1 then **(TList, ELock, false)** else (TList, RAcc, true)
	Shared Lock	*(TList, "LX to LS Error", false)*	*(TList, "Duplicate LS Error", false)*
	Unlock	Let val NewTList = eliminateTrans(CurT,TList) in if List.null NewTList then (NewTList, "F", true) else (NewTList, ELock, true) end	
	Read	(TList, ELock, true)	
	Write	(TList, ELock, true)	*(TList, "No X Lock Error", false)*

5. Conclusion

CPN is flexible and powerful method for modeling and formal analysis of the distributed nondeterministic systems. In this paper a new novel, scalable and extendable model of 2PL CCM using hierarchical CPN presented. State space analysis of the model is done using CTL formulas. State space analysis permits us to prove that all schedules of transactions are deadlock free or not. State space calculation and analysis is done fast and completed in few seconds. Model can easily change for modeling and study of other CCMs like strict 2PL. Model of CPN that used for formal verification, can easily extend for performance analysis too.

6. References

[1] Christopher J. Date. An Introduction to Database Systems, 8th ed., Pearson Education, 2004, pp. 465-494.

[2] M. Tamer Ozsu. Modeling and Analysis of Distributed Database Concurrency Control Algorithms Using an Extended Petri Net Formalism, IEEE Transactions on Software Engineering, vol. SE-11, no.10, pp. 1225-1240, Oct. 1985.

[3] Bidyut Biman Sarkar; Nabendu Chaki. Modeling and Analysis of Transaction Management for Distributed Database Environment Using Petri Nets, Proc. World Congress on Nature & Biologically Inspired Computing (NaBIC 2009), pp. 918-923, Dec. 9-11, 2009.

[4] Hou Jie; Li Fengying; Wang Huijiao. Petri Net Based Model for Concurrent Control of Database System, Proc. International Conference on Intelligent Computing and Integrated Systems (ICISS), pp. 813-815, Oct. 22-24, 2010.

[5] Bao Chyuan Chyuan Jenq; Brian C Twichell; Thomas Walter Keller. Locking Performance in a Shared Nothing Parallel Database Machine, IEEE Transactions on Knowledge and Data Engineering, vol. 1, no. 4, pp. 530-543, Dec 1989.

[6] Klaus Voss. Prototyping and Verifying Distributed Database Systems Using Executable High-Level Petri Net Models, Proc. IEEE International Conference on Systems, Man, and Cybernetics, "Computational Cybernetics and Simulation", vol. 4, pp. 3395-3400, Oct. 12-15, 1997.