Indian Journal of Science and Technology, Vol 8(35), DOI: 10.17485/ijst/2015/v8i35/54980, December 2015
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

Modelling Causal Consistency for Distributed Systems using Hierarchical Coloured Petri Net

Saeed Saeedvand[1*], Mortaza Abbaszadeh[2] and Fahimeh Ansaroudi[1]

[1]Computer Engineering Department, Faculty of Electrical and Computer Engineering, University of Tabriz, Tabriz, Iran; Saeedvand@tabrizu.ac.ir, f.ansaroudi@ucna.ac.ir
[2]Department of Computer Science, Ilkhchi Branch, Islamic Azad University, Ilkhchi, Iran; m_a138yahoo.com

Abstract

Data Consistency of distributed systems is considered as one of the greatest challenges in distributed systems due to delay of transmitting information in the network. Coloured Petri Net (CPN) simulator has high potential in modeling different algorithms and formal analysis. In this paper, a hierarchical model for modeling causal consistency with global time (UTC) has been presented for the first time, and then this model has been performed and proved for CPN by using the presented simulator. Since casual consistency provides an acceptable consistency, in this paper, it has been shown that how we can make distributed systems concurrent to reach causal consistency.

Keywords: Causal Consistency, Coloured Petri Net, CPN Tools, Distributed Systems, Modeling

1. Introduction

Distributed systems are foundation of services in different computer systems. In distributed system, data-consistency is one of the main issues in this area. So, it is essential to keep data of distributed systems consistent despite of many limitations and challenges. The most ideal case of distributed systems is having consistent data in real-time. In this case, by data consistency we mean that reading and writing data of distributed system in each time should be seen equally (Strict consistency)[1]. In practice, performing strict consistency is impossible for different reasons such as delay in sending data between distributed systems. So, to keep data of distributed system to be consistent different approaches are presented. Since functions performed in memory have causality relationship[2], causal consistency algorithm is considered to present a structure following executed operations in distributed systems according to a special order. Causal consistency algorithm is one of the consistent models which provide an acceptable level of consistency. In comparison with other presented models, this model performs weaker than sequential model[3] and stronger than other consistency models such as Weak and FIFO or PRAM[4,5]. It should be mentioned that sequential consistency model has some limitations in terms of implementation, so various and sometimes unnecessary challenges are created. In causal consistency model, when a process in a distributed system begins to perform a task, the value of one variable can be changed. So, all other distributed systems must properly observe (and can read) these changes in the order that they have been performed, if they have causality relationship. Petri net[6–8] is a graphical tool to describe formal concurrent activities currency and to analyze them in terms of proving them mathematically. CPN Tools are advanced simulations for modeling petri nets, and in this part, there is programming in ML artificial intelligence. Therefore, in this paper, a hierarchical model is presented for modeling causal consistency function. Then, it is performed and proved in CPN tools modeling environment. According to previous researches carried out in terms of casual consistency, Mahajan and his colleagues[12] in their research have proved that there is not any stronger model than causal consistency in real time unilateral convergence for constant availability of system. Simultaneously, Lloyd[9,10] showed that causal consistency with read-only and just write transactions can be obtained in a

*Author for correspondence

wide network with the least delay and high availability for data centres. In another research[11] presented a scalable causal consistency model using physical watch and dependency matrix. They used dependency matrix for types of their model and used physical watch to synchronize them. To implement the model, they performed two protocols in the "Orbe" software and presented a model for scalability and efficiency. In the first protocol, they presented an approach to extract dependencies, and in the second protocol, they kept causal consistency. They studied the results in the case of delay in sending and receiving dependencies presented an algorithm using causal consistency to consider security, availability and durability in distributed data[12,13]. Their main aim was to present a model for "bolt-on" shim layer to consider the relative problems and then to solve them. In this paper, firstly, the structure of "bolt-on" is described, and then causal consistency model is stated on it. In the section of implementation and results, throughput and latency created in "bolt-on" is depicted.

2. Coloured Petri Net

Petri net is a graphical tool to formally describe activities currency in concurrent systems and to analyze them[6–8]. A Petri net is consisted of four main foundations; place, transition, arc and token. Places depict possible states of system. Each place considered as a circle or oval can consist of some tokens. Tokens depict that system or a part of it at current time in which state can be placed. Transitions which are drawn as rectangular show the activities of system. A transition is enabled when all of its input places have tokens. An enable transition can fire and in this case one token will be subtracted from its input place and a token will be added to its output place. When two or more transactions are enabled concurrently, each one can be fired. Selecting one transaction from enabled transactions cannot be predicted and is random. Oriented arcs connect places to transactions and the reverse. There is no arc between two places or two transactions. Coloured Petri Net is the expanded form of Classical Petri Net which has the ability of programming with ML programming language. ML is the programming language for artificial intelligence whose composition with coloured Petri net modeling made it useful for creating recursive functions and different commands on the edges of models. By using Coloured Petri Net, adding operators and multi-set marking is possible[14], and the best tool for modeling and verification of models is CPN tools (CPN),[15] used in the cases such as security and data base systems and also in smart algorithms[16,17]. In general, CPN model is a formal model as a math description of syntax and semantics model, and the presented approach has been shown in this paper.

3. Causal Consistency Model

Causal consistency model[2], is one of the strongest consistency models that have also been used in the performance of social networks like face book[18]. In casual consistency model, causality relation must be followed. In order to describe casual consistency model, it should be firstly supposed that each running process is executed in a distributed system, and these processes work with common data. Distributed data must have an acceptable consistency in all systems (in fact, when the word "data" is used, it means common data among distributed system that must be kept consistent. Casualty relationship has two basic and general rules, and is defined as follows.

- If a process reads a variable that has already written that data by another process and if this process is going to write one new data in the same read variable, then causality occurs.
- If two consecutive writings are performed by the process in a variable even without reading the result of writing another process, then causality occurs again.

If a process is going to read a variable in which causal mode has occurred with two above mentioned modes, then the order of writings must be appropriately executed so that causal consistency model can be followed. An example has been presented in Figure 1 to show causal consistency.

There are four processes in Table 1, and each one has been considered in a distributed form. In this table, there are two kinds of writing and reading shown respectively by the letters "W" and "R", and three values of a, b and c are used in reading and writing operations. According to two rules that have been mentioned for causal consistency, and as it is obvious in Table 1, there is a causal relationship between "W(x)a" in process 1 and "W(x)b" in process 2 (the first rule), and also between "W(x)a" and "W(x)c" in process 1 (the second rule). So, in order to keep the consistency of executing the commands in distributed systems, all reading activities done by other processes must adhere the order of these writing activities

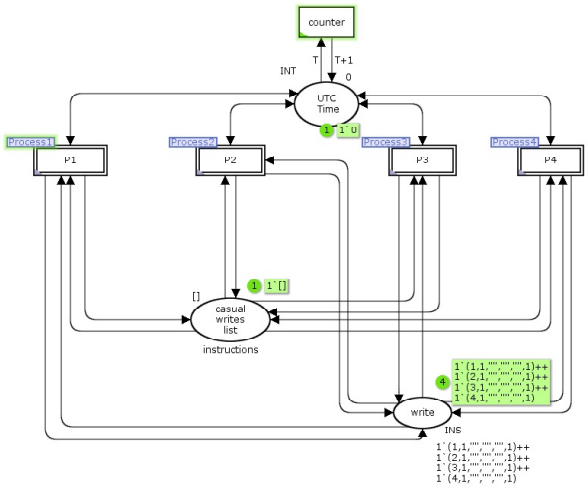

Figure 1. Top level of the presented hierarchical model.

Table 1. An example for causal consistency

P1:	W(x)a			W(x)c		
P2:		R(x)a	W(x)b		R(x)c	R(x)b
P3:		R(x)a			R(x)c	R(x)c
P4:		R(x)a			R(x)b	R(x)c

(In this example, all other processes must read the value of "W(x)a" and then the value of "W(x)b". Also, they should read the value of "W(x)a" and then "W(x)c"). For example, in process 3 and in reading activity, firstly the value "a" is read for variable "x" and then the value of "c" and "b" is read after the value of "a", and this shows that causal consistency has been followed. In addition, in process 4, causal consistency model has been followed by observing the order of readings in casualty form.

4. The Presented Model for Causal Consistency

In this section, we review the presented model, and explain all parts of this model consisting of Colour Sets, Initial Markings, ML Codes, Places and Transactions. Figure 1 depicts top level of the presented model. According to the hierarchical model, the main level of model is depicted in Figure 1. In the main level of this model, 3 places and 5 transactions exist. As it has been shown, 4 processors have been described in this model. Each processor represents one distributed system, and each of them has one sub-level that will be discussed in next sections. In the main level of this model, there is one place named "UTC Time". The task of "counter" transition is to increase the global time for synchronization of systems and their processors. The "counter" transition includes the time in input edge by "T" variable, and then one unit is added to its value, and again fires it to "UTC Time" place. Place of "write" is used for storing the information related to the last write of processes of each processor in the related variable, and the last performed operations are registered. The place of "causal write list" is considered for recording the write with causal relationship, and they can be applied in two rules mentioned for causal relationship.

4.1 The Description of Colour Sets and Variables

In the designed model, some colour sets, variable and initial marking in ML language are described for determining data types and data structures that will be discussed in details. The described colour sets are used to determine the type of places for the model and are as follows.

colset Id = INT;
colset Op = STRING;
colset gid = INT;
colset vr = STRING;
colset pid = INT;
colset vl = STRING;
*colset INS = product pid * id * Op * vr * vl * gid;*
colset instructions = list INS;

"Id" colour set has been defined for specifying identification number of each existing transaction in each processor. "OP" color set is used for specifying the type of actions performed in each transaction. The model designed for determining two "read" and "write" actions represent the actions taken in transactions. "Gid" color set determines global identification number of each transaction which this value specifies the implementation order of transactions. "vr" color set is for specifying the name of a variable in the memory on which the corresponding actions is performed ("X" variable in table 1). "pid" color set is for determining the number of the related processor which in this model due to considering four processor the value of this color set is from one to four. The "vl" color set is taken into account to determine the value of the variable existing in color set "vr" (the values of a, b and c in Table 1). The "INS" color set which is a product of previous colour sets is considered to specify an instruction in a processor or in a distributed system. Data type of "instruction" color set is a list and creates a list of "INS" color set

(a list of instructions). We display the used variables in next sections. It must be mentioned that the defined variables are for using on the edges in models.

var id, id1, id2, id3 : id;
var select : BOOL;
var vl, vl1, vl2 : vl;
var vr, vr1, vr2, vr3 : vr;
var gid, gid1, gid2, gid3 : gid;
var op1, op3, op2, op4 : Op;
var S, S1, S2, S3 : STRING;
var L, A1, B : instructions;
var M : INS;
var T : INT;
var pid, pid1, pid2, pid3 : pid;
var c : STRING;

All variables used in model are of the type of predefined color sets that are created for using on the edges of model. In fact, the task of these variables is to read the values of places that have been defined in terms of color set. It should be mentioned that the only new variable which is defined here refers to "T" variable that has not been specified in the previous step by using the color set. This variable has an integer value and is used for determining current time UTC.

4.2 Description of Initial Markings, Models of the System

In Figure 2, the part of the process number 1 has been shown by "P1" in Figure 1. Due to equality of overall structure of each sub level for "P1", "P2", "P3" and "P4", we describe only one of these sub levels. The only difference between sub levels is the process number of each sub level and the initial marking value of each of them that will be discussed later in this paper. There are a number of Initial Markings in the presented model. Each of these Initial Markings belongs to one place. In fact, Initial Markings are the initial values of places shown by green colour in the Figure 1. The most important Initial Marking in the model belongs to PX instructions in which "X" represents the number of process. The transactions in the systems are in the places of PX instructions, and they determine the implementation order of each process. For example, Initial Markings of the model for four different processes in distributed systems are as follows.

Process1: [(1, 0,"write","x","", 0), (1, 1,"write","x","", 3)]
Process2: [(2, 0,"read","x","", 1), (2, 1,"write","x","", 2)]
Process3: [(3, 0,"read","x","", 1), (3, 1,"read","x","", 4), (3, 2, "read", "x", "", 5)]
Process4: [(4, 0,"read","x","", 1), (4, 1,"read","x","", 4), (4, 2, "read", "x","", 5)]

The above mentioned Initial Markings that are the most important Initial Marking model are considered according to the example of Table 1. The values of ("a", "b" and "c") have been replaced by 1, 2 and 3. For example, there are two operations in the first row and in process 1, and each one has been placed in parentheses. In two parentheses, it has been shown that there is a process in processor 1 in which there are two "write" transactions aiming to write variable "X". The values placed in parentheses are respectively color sets defined in "instructions" color set that has been explained in previous sections. For instance, the global implementation order of process1 transactions in "UTC" is respectively zero and 3. Similarly, the Initial Markings of processors 2, 3 and 4 indicate the transactions for implementing them in each process. In addition to initial marking defined in this section, there are other Initial Markings in this model that will be discussed in next sections concurrently by describing corresponding places.

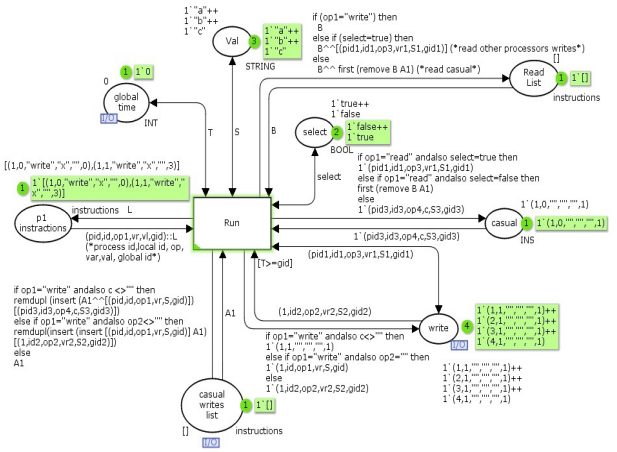

Figure 2. The corresponding sub level of processor number one in causal consistency model.

4.3 Description of Model Functions

In this section, we describe the function used in this model. All the described functions are in ML language and are written recursively. These functions are implemented on the edges of the model. These functions use binded data of places and perform some computational operations and then fire the results in the places by using the output edges.

4.3.1 Function 1

```
fun sort(a0,a1,a2,a3,a4,a5) [ ] = [(a0,a1,a2,a3,a4,a5)]
|sort (a0,a1,a2,a3,a4,a5) ((c0,c1,c2,c3,c4,c5)::queue) = if
   (a5<c5) then
  (a0,a1,a2,a3,a4,a5)::(c0,c1,c2,c3,c4,c5)::queue
else
  (c0,c1,c2,c3,c4,c5)::(sort (a0,a1,a2,a3,a4,a5) queue);
```

This function is responsible for sorting a list of processes on the basis of determined time. This function receives a list of products and a product as input arguments, and then inserts product in the list regularly. Sorting the arguments of the list is based on the last element of product. The last element of product represents global id or defined "gid" color set in the section of the colour sets. The output of this function is a list of sorted products based on global id.

4.3.2 Function 2

```
fun remove [ ] y=y |
remove ((a0,a1,a2,a3,a4,a5)::z) n=
remove z (rm (a0,a1,a2,a3,a4,a5) n);
```

The remove function receives two lists of products as input arguments and removes all the elements of the first list in the second list and returns the remained elements of the second list as output. In the presented model, this function is used when we want to remove one of executed processes from available list of each processor.

4.3.3 Function 3

```
fun insert [ ] Q = Q|
insert ((a0,a1,a2,a3,a4,a5)::q) Q = insert q (sort(a0,a1,a2,
   a3,a4,a5) Q);
```

This function receives two lists of products as input arguments, and then by calling sort function, it inserts the first list into the second list regularly. The output of this function is a sorted list based on global id and, its difference with the first function is that uses the sort function is used by the insert function to sort two in the presented model, this function is used when we want to add regularly a new process to the list of old processes according to global id.

4.3.4 Function 4

```
fun first [ ] = [ ]|
first ((a0,a1,a2,a3,a4,a5)::q) =1 `(a0,a1,a2,a3,a4,a5);
```

This function receives a list of products as input argument and returns its first element as an output which is a product. This function is used to separate the first available process in the list of processes.

4.4 Implementation Procedure of the Model

In this section, the implementation procedure of the model is presented. Places and performance of transactions are described concurrently. In the presented model, according to the presented structure in Figure 2, each transaction located in "PX" instructions begins to perform according to its "gid" and the value in global time, (the transactions with the "gid" of color set equal or greater than the value in the place of global time are allowed to perform). Afterwards, one transaction is allowed to perform, and two cases can occur.

- First if the transaction wants to do "write" action, in this case according to the structure of causal consistency and the functions and codes written on edges of the model, checks to see if the causal relationship is between consecutive writings or not (rules 1 and 2 in the sections of causal consistency model). In other words, it can be said that if the current process has already read the result of writing of other process, and then it wants to write a variable, or perform two consecutive writings in a variable, then causal consistency occurs. Therefore, the writing operations of each process are located in "Read List" place. When the writing transaction occurs, at first, "Val" place writes a new value in "write" place of each transaction separately. Then in the place of "causal write list" the time of these writings are recorded according to the order of UTC time. In fact, the place of "causal write list" is the place that eliminates the ability of reading unallowable data. Also, when causal state occurs for writing, the last written value for the related variable is located in causal place to use it in reading.
- Second if a transaction performs reading operations, at first, the model checks to find out that on the basis of the type of reading transaction, whether reading the value of the related variable is performed from the last value of the current process (internal memory), or whether reading is the result of writing other process. This is done by using the place of the "select" transaction. But if reading data of related variable is performed from the last value of the current process, then reading transaction is done from write place with data of the current process and, the result is located in place of "Read list". But, if reading the data of related variable is the result of writing of other process, in order to

maintain causal consistency by using ML functions written on edges of "causal write list", the first writing is selected and is not read by the current process (following the order of reading processes) and then locates it in place of "Read list". In fact, by performing this transaction (operation), it is guaranteed that each process reads the writings which are in causal state in the order that transactions occurred. Therefore, they consider the entire model and the distributed processes of causal consistency.

5. Analysis of State Space

After implementing this model and performing various experiments in the presented model to prove stat space, the presented model is simulated by using CPN tools, and the results are as follows:

Statistics

State Space
Nodes: 15212
Arcs: 189365
Secs: 100
Status: Partial
Scc Graph
Nodes: 15212
Arcs: 189365
Secs: 2
Boundedness Properties

Liveness Properties

Dead Markings
8475[9999, 9998, 9997, 9996, 9995,…]
Dead Transition Instances
None
Live Transition Instances
None
Fairness Properties

No infinite occurrence sequences.

Regarding to the obtained results, state space of the model contains 15212 nodes and 189365 edges. The state space of CPN tools has a visual tool for state of deadlock. In this model, it has been shown that no deadlock has occurred in nodes, and it can be observed that causal consistency is not violated by checking it. This is shown by 305 nodes in Figure 3. As it is shown in this Figure all the commands are executed correctly and completely and this indicates the proper function of the model. It must be stated that automated analysis

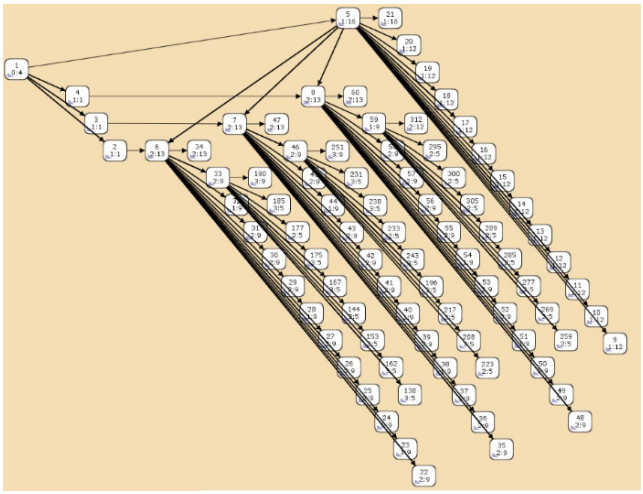

Figure 3. State space of the model in the presented example with 305 nodes.

of states of system in CPN tools simulator is performed based on computational tree logic (CTL)[19].

6. Conclusion

Since data management on the distributed systems is one of the greatest challenges in distributed systems, a new hierarchical model has been presented in modeling CPN for the first time in this paper. In the presented model, causal consistency model was modelled in distributed systems to keep data consistent among several processes. With regard to the fact that CPN tools simulator is a strong simulator to prove distributed undeterministic systems, the presented model has been implemented by CPN tools simulator and by using ML functions. In the presented model, all the rules of Causal Consistency are followed by the processes, and in this way, the violation of these states are prevented. In the final section, the analysis of space states was performed by using CLT and CPN tools, and the states were proved in terms of execution and computation aspects. It should be mentioned that, in this research, it has been shown that the model lacks any deadlock, therefore, the presented model can be used to be implemented in various systems.

6. References

1. Herlihy MP, Wing JM. Axioms for concurrent objects. Proceedings of the 14th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages - POPL; 1987.

2. Ahamad GNM, Burns JE, Kohli P, Hutto PW. Causal memory: Definitions implementation, and programming. Springer, Distributed Computing. 1995; 9:37–49.
3. Lamport L. How to make a multiprocessor computer that correctly executes multiprocess programs. IEEE Trans Comput. 1979; 28(9):690–1.
4. Dubois M, Scheurich C, Briggs F. Memory access buffering in multiprocessors. Proceedings of 13th Annual International Symposium on Computer Architecture; 1986. p. 434–42.
5. Sandberg L. PRAM: A scalable shared memory; 1988.
6. Jensen K. Coloured petri nets basic concepts, analysis methods and practical use. Basic Concepts of Monographs in Theoretical Computer Science. An EATCS Series, Springer-Verlag; 1992.
7. Jensen K. Coloured petri nets. basic concepts, analysis methods and practical use. Analysis Methods of Monographs in Theoretical Computer Science. An EATCS Series, Springer-Verlag; 1994.
8. Jensen K. Coloured petri nets basic concepts, analysis methods and practical use. Analysis Methods of Monographs in Theoretical Computer Science. An EATCS Series, Springer-Verlag; 1997.
9. Lloyd W, Freedman MJ., Kaminsky M, Andersen DG. Don't settle for eventual: Scalable causal consistency for wide-area storage with COPS. Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP'11); 2011. p. 401–16.
10. Lloyd W, Freedman MJ., Kaminsky M, Andersen DG. Stronger semantics for low-latency geo-replicated storage. 10th USENIX Symposium on Networked Systems Design and Implementation; 2013.
11. Du J, Elnikety S, Roy A, Zwaenepoel W. Orbe: Scalable Causal Consistency Using Dependency Matrices and Physical Clocks. Proceedings of the 4th Annual Symposium on Cloud Computing, SOCC '13; 2013.
12. Mahajan P, Alvisi L, Dahlin M. Consistency, availability, convergence [Technical report TR-11-22]. Austin, Texas: Computer Science Department, UT Austin; 2011.
13. Bailis P, Ghodsi A, Hellerstein JM, Stoica I. Bolt-on causal consistency. Proceedings of the Annual Symposium on Cloud Computing, SIGMOD'13; 2013.
14. van der Aalst WMP, Stahl C. Modeling business processes: A Petri net-oriented Approach. The MIT Press; 2011.
15. CPN tools. Available from: http://cpntools.org/download
16. Pashazadeh S. Modeling and verification of access rights in take-grant protection model using colored Petri nets. IJINS. 2013; 2(1):413–25.
17. Pashazadeh S, Saeed S. Modelling of walking humanoid robot with capability of floor detection and dynamic balancing using colored petri net. IJFCST. 2014; 4(2).
18. Bailis P, Fekete A, Ghodsi A, Hellerstein JM, Stoica I. The potential dangers of causal consistency and an explicit solution. Proceedings of the 3rd ACM Symposium on Cloud Computing, ACM SOCC; 2012.
19. Katoen J-P, Baier C. Principles of model checking (Representation and Mind Series). Cambridge, Massachusetts, USA: The MIT Press; 2008. p. 229–433.