

カラーペトリネットによる データベースモデルの性能解析

Database Performance Modeling Based on Colored Petri Net

松本 篤[▼] 三浦 孝夫[◆]
塩谷 勇[▲]

Atsushi MATSUMOTO Takao MIURA
Isamu SHIOYA

本稿ではカラーペトリネット(CPN) に施錠機能を加えた時間付きカラーペトリネット CPN+ を導入し、データベース処理の性能評価に用いることができることを示す。CPN+ のためのコマンド言語を導入し、CPN+ の汎用離散型事象シミュレータによる近似手法を論じる。また評価実験により本アプローチの有効性を示す。

In this investigation, we discuss how well database performance modeling can be established by means of Colored Petri Net (CPN). To do that, we examine prototype features from CPN and define Timed CPN with locking (CPN+) which provides us with suitable modeling for database performance. We introduce a text-based commands that are equivalent to CPN+ by means of general discrete event simulators. By experiments we show how we can go well with the simulators.

1. まえがき

ペトリネット(以降 PN)あるいはその部分クラスによる解析の目的は、利用者の意図するシステムをモデル化して有用な情報を得ることである。ここには構造解析と動作解析の側面があり、前者にはペトリネットの連結性検査等が、後者には実行条件や到達可能性の検査などが含まれる。様々な分野に対する応用が提案され、有用な情報を提供している。特に、ペトリネットモデルを用いることで、データベース処理のモデル化やデータベース性能解析に関する直感的な理解が容易になる。このため、複雑な分散型データベースなどの性能解析などへの適用が可能となる[1]。

本稿では、データベースシステムの性能評価のためのモデルを構築する。このため、カラーペトリネット(以後CPNとする)を拡張し、そのシミュレーションによる実行効率の測定方式を提案する。ここでは状態遷移に注目した解析を行う。状態遷移は非同期的かつ原子的に発生し、遷移中の中間状態を考えない。このような状態は提案する方式で簡単に記述できる。一般に、データベースシステムの性能モデルは、同一の

データに対して複数利用者を同時制御することで最大限の性能を引き出せるように調整することを目的としている。本稿で CPN を用いる結果、オブジェクト概念をトークンに対応させることで、個々のトークンの追跡が可能となり、属性値を与えたコンパクトな記述を得ることができる。さらに、施錠処理や再スタートなどトランザクション処理のモデル化や環境条件を調整することが容易になる。

しかし、一般にペトリネットの実行と結果の解析は大変困難である[1]。この理由は PN が本来並列かつ分散環境で動作するというところに起因している。このため、シミュレーションモデルを構築し、これを実際に動作させることが多い。

本稿では、同時制御機構を含む CPN+ を導入し、離散事象シミュレータにより近似する。データベース性能モデルの構築とシミュレーション結果を基に、Agrawalらの研究[7]と比較し有効性を検証する。

次節でPN/CPNの特徴・問題点を要約し、CPN+を提案する。第3節でCPN+による性能評価モデルの構築を論じ有用性を示す。4節では、離散型事象シミュレータ上でのCPN+の実現を論じ、第5節で実験結果を述べ過去の研究結果と比較する。

2. ペトリネットとその拡張

ペトリネット(PN)モデルでは、事象を表すトランジションを|で、トークンの保存場所であるプレースを○で、また条件と遷移を関連付けるために有向辺(アーク)で、システム内を動き回るトークンを●で表す。入力辺の無いプレースをソース、出力辺の無いプレースをシンクという。原理的には、全ての入力プレースに十分な数のトークンが存在するとき遷移が発火可能となる。その後の遷移は、資源の利用状況など周囲の環境に応じた定義がなされる。発火すると入力プレースからトークンが除かれ、全ての出力プレースに新しいトークンが生成される[1][2][3]。

CPNでは個々のトークンをカラートークンと呼ぶ。これは個別にカラー属性値を有し、状態遷移はカラートークンの状態に依存する。同様に、プレースにもカラー属性型を保持し、トークンが取る属性値を決定する。プレースは、それ自身が保持可能なカラー属性値を決定する。遷移にはガードと呼ぶ条件が対応する。辺にはカラートークンの数を表す式を与えることがある。

PNやCPNには遅延時間の概念を加えることができ、これを時間ペトリネット(TPN)と呼ぶ。TPNでは特殊なプレースや遷移に対して、非決定的な遅延時間を指定でき、現実問題に適合しやすく解析結果が得られやすい。プレース型TPNには遅延のためトークンを保持するプレースがあり、遷移型TPNでは遅延時間を設けて発火する遷移がある(図1参照)。TPNは、FMS、意思決定などで有用であることが知られている[4]。

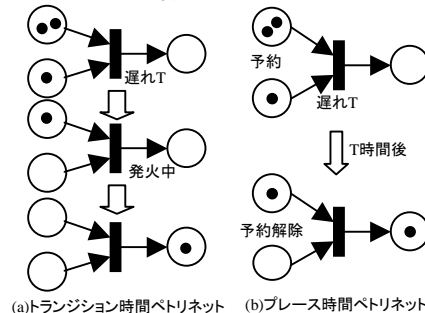


図1 2種類の時間ペトリネット

[▼] 学生会員 法政大学工学研究科電気工学専攻
miurat@k.hosei.ac.jp

[◆] 正会員 法政大学工学研究科電気工学専攻
miurat@k.hosei.ac.jp

[▲] 正会員 産能大学経営情報学部情報学科
shioya@mi.sanno.ac.jp

データベース処理の性能評価モデルを構築するため、本稿では CPN を特化しトークンの分岐生成・合併および施錠に関して拡張した CPN+ を導入する。トークンは整数で識別され、しかもこの値は明示的に操作できる。特に、トークン番号は幹と枝の部分からなる。枝は整数であるが、幹は整数または(再帰的に)幹と枝部分からなる(例えば1, 4-1, 4-2-3)。本稿では正則なCPNだけを論じる。つまり属性値に関する条件や式はプレースや遷移に対して指定できるが、特殊な辺(抑制辺など)は無いとする。また、ソースプレースはひとつしか無く、初期マーキングはソースに対してだけ与えることができる。これらの制限により簡単なCPN+だけを扱えばよく、また状態の変化は開始時から考えることになる。

本研究では、トークンの合併は識別番号が同一の発火に起因するものに制限する。この制限はデータベース処理においては重要である。実際ほとんどの場合、各並列実行単位はモニタの下で監視されており、トークンが1 in M outで分岐生成されれば、入力トークンの識別番号を幹とするM個の枝が生成され、それぞれが一意的番号を有する。トークンがN in 1 outでひとつに合併される時、入力トークンは分岐生成された枝のいずれかに対応する。入力トークンは幹を共有する必要があり、合併後のトークンの識別値は幹の値となる。例えば4-1,4-2,4-3などは合併して4となる。このような事態をCPN+ でモデル化するのは難しくない。

データベース処理において必須機能の一つに施錠機構がある。本稿ではCPN+により対象資源を施錠するための記述手法を示す。与えられた資源名に対し、読み込み施錠と書き出し施錠を考慮することができる。このためにCPN+では施錠解錠機能を導入する。構成要素として2種類の三角形(施錠)、(解錠)があり、それぞれに要求資源と要求数を指定する。

【例1】ある工場の生産工程では、2種類の部品が2秒毎に現われ、それぞれ2台の製造機械、でこれらに対応する。2台は同時には動作できず、の動作時間は4秒、は3秒である。このような環境下における100秒間での生産総量を知りたい。ただし2種類の部品が生じる確率はそれぞれ0.4,0.6である。

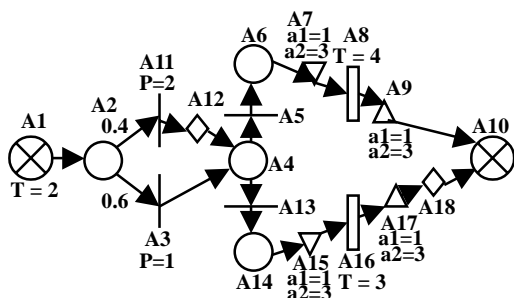


図2 CPN+による生産工程のモデル化

図2はCPN+によるモデルである。部品がプレースA1から発生し、プレースA2で を表すA8または を表すA16に分岐する。A7, A9 及び A15, A17 は施錠、解錠を表し、a1は要求する資源数(a1)と対象(a2)を示す。これらはそれぞれ処理A8, A16を囲んでおり、異なる2つの製造機械が一度に処理することができる部品数を制限している。

3. 同時制御モデルのための CPN+

3.1 データベース性能モデル

データベースシステムでは各利用者の処理は論理的に整合した作業単位と仮定しており、これをトランザクションと

呼ぶ。この結果、無矛盾状態のデータは新たな無矛盾状態へ変化するが、これらの干渉でデータベース処理の性能低下を引き起こす。この問題に対しては、直列可能性を用いた解消手法がよく知られている[5][6]。

直列性判定には、代表的に3つの方式(ブロッキング方式、即時再スタート方式及び楽観的再スタート方式)が知られる。ブロッキング方式では、データ資源に対する施錠要求に対して矛盾状態を調べ、問題が無ければ処理を継続させる。さもなければ矛盾状態が解消するまでブロックされる。2種類の施錠(共有施錠と排他施錠)があり、読み込み要求は同時に実行できるため共有施錠を用いる。しかし更新要求については、単一の施錠が排他的に満たされねばならない。ともえ施錠が生じるため再スタート機構が必要である。

即時再スタート方式では、施錠のタイミングまではブロッキングアルゴリズムと同じであるが、施錠要求が拒否されるとトランザクションを直ちに終了させ、再スタートする点が異なる。再スタートの実行時に同じ競合が繰り返されることを避けるために遅延時間を設ける。

楽観的再スタート方式では、施錠機構がコミット要求時に矛盾状態を調べる。ここでは、すでに完了したトランザクションが更新したデータを完了前に読み込んだならば再スタートを指示する。このアルゴリズムはオーバーヘッドが少ないため、矛盾がほとんど生じないときは効果的である。競合したトランザクションはすでに処理を完了しており遅延時間を設ける必要はない。

トランザクションの発生と消滅はソース・シンクでなされる。実際にはトランザクション総数に制限がありこれをマルチプログラミングレベル(mpl)と呼ぶ。トランザクションが生成されると、他のトランザクションが完了か再スタートするまで readyqueue で待機させられる。この意味で、待ち行列モデルで表現するのが妥当である。原理的には同時制御(cc)待ち行列を用意し、全ての要求制御は cc を経由して監視され、要求が認められればオブジェクトにアクセスすることができる。各トランザクション実行のための遅延時間 think を設ける。要求が拒否されると、トランザクションは blockqueue で待機し、再スタートになった場合には readyqueue に戻される。即時再スタートの場合はここでの遅延時間を与えない。トランザクションが全ての作業を完了したとき、コミット要求を行う。これが許可された場合全ての更新データはデータベースに書き込まれ同期が取られて、処理はソース・シンクに戻される。さもなければ再スタートさせられる。このような論理的待ち行列モデル([7])の例を図3の左側に示す。

他方、物理的待ち行列モデルはソース・シンク、CPUおよびディスク装置(HDD)の動作を記述する。このモデルにも restart と think パスにおける遅延時間が反映されている。1つのCPUと2台のHDDが処理単位としてモデル化され、トランザクションはFCFS方式で制御される。トランザクションに対して、空いているCPUが割り当てられ、ディスクは同確率で割り当てられる。この例を図3の右に示す。

3.2 CPN+ を用いたモデル化

本稿で構築するモデルは読み込み部と書き出し部からなる。読み込み処理に関し、ソースで生成されたトークンは mpl が飽和するまで readyqueue を通過し、遷移 assign read で ReadCount をトークン属性として入手した後に cc へと向かう。これはアクセスするオブジェクト数を規定する。cc ではある確率で再スタートするかオブジェクトをアクセスするかを判断する。再スタートされたトークンは mpl を解錠し、

適当な遅延時間の後,readyqueue に戻される.オブジェクトにアクセスするトークンは CPU/HDD を要求し,その後 FinishReading でオブジェクトアクセス数が0かどうかの検査がなされ,0 でなければこれを減じ,cc に戻って同じ処理をする(図4参照).さもなければ次のステップに進む.

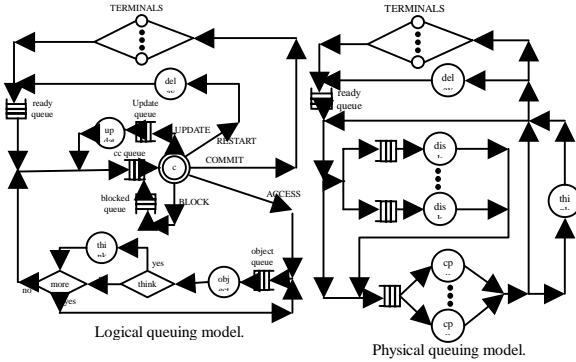


図3 論理キューモデルと物理キューモデル

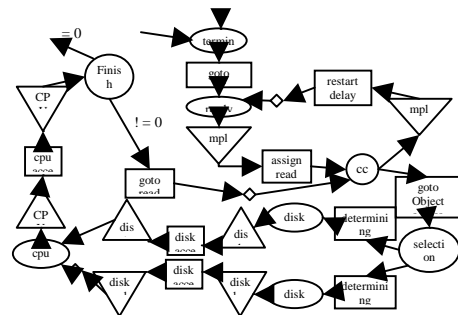


図4 CPN+による読み込み性能評価モデル

書き出し処理では,cc においてともえ施錠の検出がなされ,確率分だけ再スタートされる.トークンは WriteCount 属性値を入手して処理オブジェクト数を規定し,先ほどと同様に CPU/HDD 処理単位の獲得に進む.その後,プレース finish write request で操作回数属性値 WriteCount を検査し,次の処理に進むかどうかを判断する.これらを完了したトークンは遅延更新を行い,問題が無ければコミットに進む(図5参照).

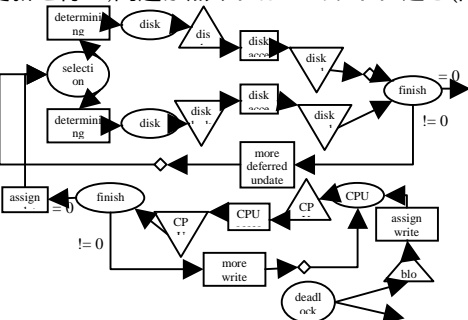


図5 CPN+による書き出し性能評価モデル

4. CPN+の実現

CPN+を処理するためには,文字列による構文の方が解析しやすく,図形インターフェイス(GUI)とプロトタイプシステムを対応させる環境を与える.本研究では,CPN+を離散事象シミュレーションモデルに変換・実行する方式を採用する.このことで比較的容易に CPN+が実現できる.まず CPN+コ

マンドを定義する.

記法/意味

- label: tstart (t1,t2,T) B1,...,Bn
時間ソースプレース
- label: tnend
シンクプレース
- label: goto label2
トークンを移動
- label: trans (N,t1,t2) C1,...,Cn
時間型(t1>0)/非時間型(t1=0)遷移
- label: place (t1,t2,M) C1(p1),...,Cn(pn)
時間型(t1>0)/非時間型(t1=0)プレース
- label: atrans (a1,a2)
属性値の割り当て
- label: lock (a1,a2)
- label: unlock (a1,a2)
施錠(解錠)要求

本研究では離散事象シミュレータとして,簡単に利用できる GPSS (General Purpose Simulation Systems)を用いる.ここでは2種類のエンティティ(静的と動的)があり,静的エンティティは固定的なオブジェクト(例えば storage(配列),facility(待ち行列),logic switch(待ち行列の真偽値))を,動的エンティティは一時的オブジェクトを意味する.動的エンティティの処理は静的エンティティを用いたグラフ表現によりフロー図として記述される[8][9].

CPN+と GPSS の変換は直接的であり,図6に例を示す.対応の詳細は[10]を参照されたい.

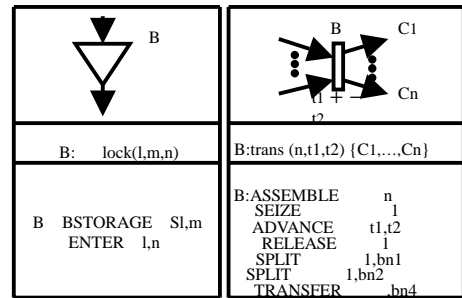


図6 CPN+モデル構成例

```

A1:tstart(2...100)
A2:place A3(0.6),A11(0.4)
A3:atrans(1,1)
A4:lplace(1) A13
A5:trans
A6:place
A7:lock(3)
A8:trans(4)
A9:unlock(3)
A10:tend
A11:atrans(1,2)
A12:goto A4
A13:trans
A14:place
A15:lock(3)
A16:trans(3)
A17:unlock(3)
A18:goto A10

SIMULATE
START 1
A1 GENERATE ,,100,1 A9 LEAVE 3
TERMINATE 1 A10 TERMINATE
GENERATE 2 A11 ASSIGN 1,2,PH
A2 TRANSFER .600,A11,A3 A12 TRANSFER ,A4
A3 ASSIGN 1,1,PH A13 PRIORITY 0
A4 LOOP 1PH,A13 A14 PRIORITY 0
A5 PRIORITY 0 A15 BSTORAGE S3,1
A6 PRIORITY 0 A16 BSTORAGE S2,1
A7 BSTORAGE S3,1 ENTER 3
QUEUE 1 DEPART 2
ENTER 3 A16 BSTORAGE S2,1
DEPART 1 ENTER 2
A8BSTORAGE 1,1 ADVANCE 3
ENTE 1 LEAVE 2
ADVANCE 4 A17 1FAVE 3
    
```

図7 GPSS による生産工程の記述

[例2] 例1を CPN+言語表現に変換した結果が図7左側に示す.右側に GPSS ブロック図を示す.ここで storage1 は A8 を,storage2 は A16 を表す.また queue1 は A6 に,queue2 は A14 に対応する.実際にこれを実行することで解析できる(図8).この例では次のような結果を得る.

- (1) A8 では 16 トークンが入力され,そのうち 15 が完了した . A16 では 12 入力トークン全てを完了した .
- (2) プレース A6 は のための待ち行列であり 26 トークンを受け取ったが 1 つしか即時に完了させなかった . 機械 のための待ち行列 A14 は 23 個のトークンが入力され全てを完了した .
- (3) A6 での平均待機時間は 19.731 秒であり,A14 では 23.826 秒であった .
- (4) これより製造機械 は より優れた生産性を示す .

--AVG-UTIL-DURING--		AVERAGE		CAPACITY	AVERAGE		CURRENT	MAXIMUM
STORAGE	TOTAL	ENTRIES	TIME/UNIT		CONTENTS	CONTENTS		CONTENTS
1	0.620	16	3.875	1	0.620	1	1	
2	0.360	12	3.000	1	0.360	0	1	
3	0.980	28	3.500	1	0.980	1	1	
QUEUE	MAXIMUM	AVERAGE	TOTAL	ZERO	PERCENT	AVERAGE	SAVERAGE	
	CONTENTS	CONTENTS	ENTRIES	ENTRIES	ZEROS	TIME/UNIT	TIME/UNIT	
1	10	5.130	26	1	3.8	19.731	20.520	
2	11	5.480	23	0		23.826	23.826	

図 8 GPSS 出力結果

5. 実験

本アプローチの有効性を示すため, Agrawal らの結果[7]と比較しながら検討する . [7]では独自のシミュレータにより実験したが,このとき次の共通パラメータを用いている .

Mean size of Transaction	8-page readset
Size of Largest Transaction	12-page readset
Size of Smallest Transaction	4-page readset
WRITE probability	0.25
Mean Delay	zero or adaptive
Number of Terminals	200 terminals
mpl	5,10,25,50,75,100,200
External Thinking Time	1 second
Time for Objects	35 milliseconds
Time for calculation	15 milliseconds

本研究の実験の主要な違いは 2 点ある . 読み込みに関するパラメータは max_size, min_size の代わりに平均読み込み数である tran_size とそのバラツキ値を使用する . また再スタート確率とブロッキング確率については([7]ではオブジェクトの総数に対する読み込み数などで間接的に指定しているのに対し)直接この値を利用する .

本稿では db_size を 10000 ページ,1000 ページの実験を行い,競合の確率を変えた実験結果を示す(図 9 および 10) .

実験結果を比べてみると,ブロッキング,即時再スタート,楽観的再スタートのいずれもが極めて類似した特徴を有することが分かる . 例えば,ブロッキングアルゴリズムでは全ての状況で値がスラッシングにより下降しており,即時再スタートの場合,mpl の高い値では一定値をとる . これらは[7]と類似した傾向を示す .

実験結果が異なるのは初期値の捕らえ方の差によるものと思われる . 例えば再スタートやブロッキング確率は db_size, tran_size の組み合わせによって定義されているが,本実験ではこれらの値は動的に決定される .

6. 結論

本稿ではデータベース処理の性能評価モデルを CPN+を用いて構築する手法を提案し,実験を用いてその有効性を示した . また CPN+を従来の離散事象シミュレータに変換し,実行する手法を述べ,CPN+を簡単に動作させる手法を示した . この結果,CPN+がデータベース処理の性能評価モデルを構築するのに適しており,様々な状況を検査できる . 現在,本

アプローチを分散環境へ拡張することを検討している .

なお CPN+ コマンド変換ソフトウェアについては <http://www.dbl.k.hosei.ac.jp/> で公開している .

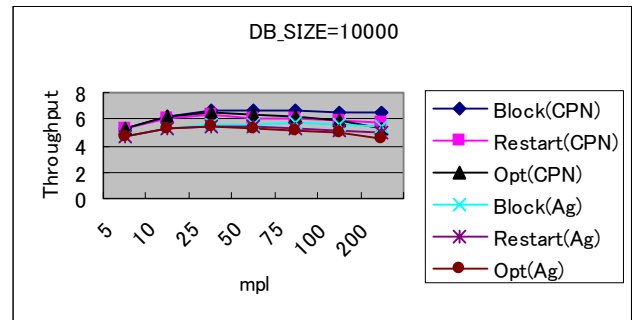


図 9 データベースサイズ 10000

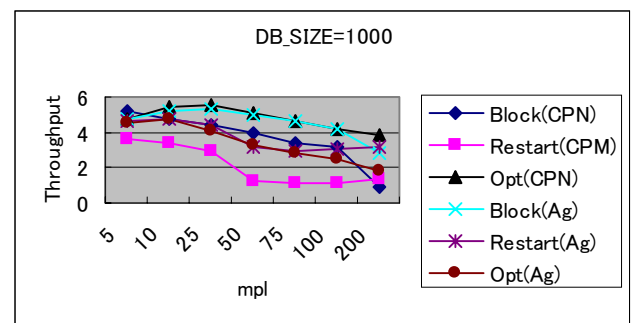


図 10 データベースサイズ 1000

[文献]

- [1] Jensen, K., "Coloured Petri Nets (Vol.1)", Springer-Verlag (1997)
- [2] Peterson, J.L., "Petri Net Theory and the Modelling of Systems", Prentice-Hall (1981)
- [3] Reisig, W., "Petri Net", Springer-Verlag (1985)
- [4] Silva, M., "Petri Net and Flexible Manufacturing", LNCS 424, pp.374-417, Springer-Verlag (1990)
- [5] Bernstein, P., Newcomer, E., "Principles of Transaction Processing", Morgankaufmann (1997)
- [6] Papadimitriou, C., "The Theory of Database Concurrency Control", Computer Science Press (1986)
- [7] Agrawal, R., Carey, M.J. et al., "Concurrency Control Performance Modeling - Alternatives and Implications", ACM TODS, 12-4, pp.609-654 (1987)
- [8] Henriksen J.O. et al., "GPSS/H - Reference Manual", Wolverine Software (1999)
- [9] 中西俊男, "コンピュータシミュレーション, 近代科学社 (1977)
- [10] Matsumoto, A, Miura, T.: "Prototyping Timed Petri Net by Discrete Event Simulator", IEEE PACRIM, pp.337-340(2001)

松本 篤 Atsushi MATSUMOTO

法政大学工学研究科電気工学専攻修士課程在学中 .

三浦 孝夫 Takao MIURA

法政大学工学情報電気電子工学科教授 .

塩谷 勇 Isamu Shiyoa

産能大学経営情報学部情報学科助教授 .