計算機アーキテクチャ 97－4
（1992. 12. 3）

線形システム解析を利用したStochastic Petri Netsの分割解析手法

國松 敦　　山本 欧　　天野 英晴
慶應義塾大学 理工学部

Stochastic Petri Netsの過渡解析手法を示し、その分割解析手法を提案する。この方法では、Stochastic Petri Netsを展開して得たContinous-time Markov chainsの生成行列を対角化することにより、線形微分方程式の解式を導出する。さらにその計算コストを削減するために、線形システム解析に基づきStochastic Petri Netsを分割して解析する。Stochastic Petri Netsで記述されたマルチプロセッサシステムを例にとり、計算時間の比較により、分割解析手法の有効性を明らかにした。

A transient analysis method of Stochastic Petri Nets based on linear system analysis

Atsushi Kunimatsu　　Ou Yamamoto　　Hideharu Amano
Keio University, Faculty of Sceience and Technology

A novel transient analysis method of Stochastic Petri Nets (SPN) is proposed. In this method, an infinitesimal generator of the Continuous-time Markov chains derived from SPN is diagonalized to solve linear differential equations representing the transient state. In order to reduce the computation time, the target SPN is divided into small subnets by using a method from linear system analysis. A simple multiprocessor system represented with SPN is analyzed with the proposed method, and the computation time is evaluated.

1 はじめに

Stochastic Petri Nets(以下 SPN)[Marsan 1989] は Petri Nets(以下 PN) に時間と確率の概念を導入したシステム記述モデルである。SPN はプロトコルの解析などに用いられると共に、計算機システムの性能解析にも有用であり、既に様々な評価が行なわれている。[Marsan 1984] SPN の利点は、マルコフ チェーンに展開した時の状態数が比較的少ないことと、システムを記述したときのネットをそのまま PN として解析出来ることである。

SPN を用いた性能解析は、そのシステムの定常状態に関して行なわれることが多い。[Sanders 1986][Chiola 1988][Ciardo 1989] しかし、計算機システムの挙動を解析する場合、あるシステムに変化が生じた場合に時間と共にそのシステムがどのように変化するか、すなわち、過渡解析が重要になる場合もある。

しかしながら SPN の状態数は、プレース上のトークンの分布に関する組み合せに依存するので、そのまま過渡解析すると計算コストが膨大になってしまう。本研究では、過渡解析を主目的とした、線形性に基づく SPN の分割解析手法を提案する。

2 Stochastic Petri Nets

SPN は PN の性質と Continuous-time Markov chains(以下 CTMC)[Marsan 1989] の性質を合わせ持つシステム記述モデルと言える。ここではまず SPN を紹介し、その後 SPN の解析に必要な CTMC を紹介する。

2.1 Stochastic Petri Nets

SPN は PN のトランジションに発火時間の確率分布を導入したシステム記述モデルである。[Marsan 1989]

同じ PN の拡張である Timed Petri Nets(以下 TPN)[Holliday 1985] に比べ、TPN ではトランジションごとに発火時間と発火確率を割り当てているのに対して、SPN ではトランジションごとに発火時間の確率分布関数 (probability distribution function : PDF) を割り当て発火時間と発火確率の両方を一度に導入している。具体的には、SPN のトランジションの発火時間はランダムであり、同じトランジションでも毎回発火時間が異なるが、十分多くの発火回数において発火時間の分布を調べると、それが割り当てられた確率分布関数になっている訳である。確率分布関数は、X を確率変数 (random variable),P をある事象の起きる確率として次の (1) 式で示される。

$$F_X(x) = P\{X \leq x\} \tag{1}$$

この式はある時間 t までに状態が遷移する確率が

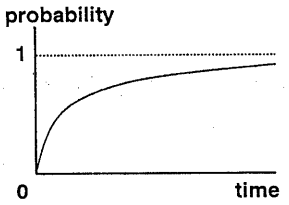

図 1: F_X

$F_X(t)$ であるということを表現している。この場合 $\lim_{x\to-\infty} F_X(x) = 0, \lim_{x\to\infty} F_X(x) = 1$ である。

複数のトランジションが発火可能な場合にはランダムに決められた発火時間のうち最も短いものが発火する。このアルゴリズムによって確率分布関数に対応した発火確率が各トランジションに割り当てられる。

また $F_X(x)$ を微分した

$$f_X(x) = \frac{d}{dx} F_X(x) \tag{2}$$

を確率密度関数 (probability density function :pdf) という。

SPN において非記憶性 (次の状態への遷移確率は現在の状態のみに依存し過去どんなルートを辿ったかという履歴には関係しない性質:memoryless property) を SPN で実現するために減少指数密度関数 (negative exponential pdf)

$$f_X(x) = \mu e^{-\mu x} u(x) \tag{3}$$

が良く使われる。($u(x)$ はステップ関数)

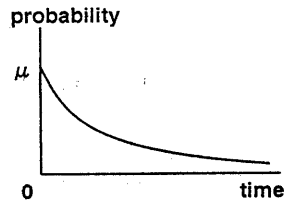

図 2: f_X

この減少指数確率密度関数に対応する確率分布関数の P は

$$P\{X \geq x+\alpha | X \geq \alpha\} = P\{X \geq x\} \tag{4}$$

の条件を満たしており非記憶性を実現している。

(3) 式の μ はこの指数密度関数の係数であり $\frac{1}{\mu}$ は状態遷移の平均時間を表している。また指数密度関数の係数が μ と λ のである二つのトランジションが発火可能な時、μ が発火する確率は $\frac{\mu}{\mu+\lambda}$、λ が発火する確率は $\frac{\lambda}{\mu+\lambda}$ で表すことができる。

例として三つのプロセッサが二つのリソースを取り合うシステムを SPN で記述すると図3の様になる。

SPN の特徴は

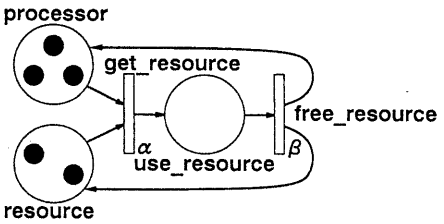

図 3: Stochastic Petri Nets の例

1. 一度に発火するトランジションは一つ
2. ネットのマーキング集合を状態集合と考えて CTMC に展開できる

である。

一つ目の特徴により SPN のマーキングの挙動は PN の挙動と同じであることが保証される。この点が TPN と比較した場合の大きな利点である。TPN では複数のトランジションが発火したり、発火中の時間も状態として区別しなければならず、PN とは異なった挙動をする。このため PN で研究された多くの成果を利用することができない。

また SPN は CTMC に展開できる。図3の SPN を展開してみると図4の様になる。

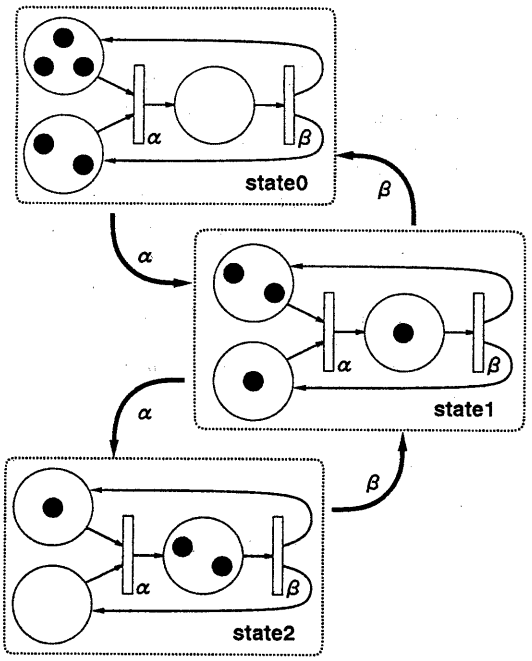

図 4: SPN から CTMC への展開

発火係数が α のトランジションが発火してトークンの分布が状態 1 から状態 2 に変化する時、CTMC の状態 1 から状態 2 へのアークの係数が α となる。性能解析を行う場合にはこの展開した CTMC を利用することになる。

2.2 Continuous-time Markov chains

CTMC は離散時間モデルである Markov chains を連続時間に拡張したものである。CTMC の記述は

- 複数の状態が存在し、その内のどれか一つが現在の状態となる
- 状態から状態への遷移はアーク (矢印) で示す
- 状態遷移のアークには確率分布関数を割り当てる

で与えられる。

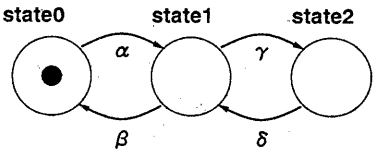

図 5: Continuous-time Markov chains の例

CTMC を解析する場合にはまず生成行列 (infinitesimal generator) を作成する。生成行列は Markov chains の遷移確率行列に相当する。

図 5の例で生成行列を作ると

$$Q = \begin{pmatrix} -\alpha & \alpha & 0 \\ \beta & -\beta-\gamma & \gamma \\ 0 & \delta & -\delta \end{pmatrix} \tag{5}$$

となる。行の数および列の数が状態の数に対応し、第 m 行の状態から第 n 行の状態に遷移する時の指数密度関数の係数が μ の時、行列の第 m,n 要素は μ となる。対角要素は各行の要素の和が 0 になるように設定する。

ここで $\boldsymbol{\pi}(t)$ を時間 t での各状態の確率を表す行ベクトルとすると、この生成行列を使って

$$\dot{\boldsymbol{\pi}}(t) = \boldsymbol{\pi}(t)\boldsymbol{Q} \tag{6}$$

という線形微分方程式を作ることができる。各状態の定常状態における確率を求めるには

$$\boldsymbol{\pi}\boldsymbol{Q} = \boldsymbol{O} \tag{7}$$

$$\sum \boldsymbol{\pi} = 1 \tag{8}$$

この線形連立方程式を解けば良い。しかし過渡解析に際しては (6) 式の微分方程式を直接解くことが必要になる。

3 SPN の解析手法

ここでは図 4の SPN について解析を行う。この CTMC から 生成行列を作ると

$$\boldsymbol{Q} = \begin{pmatrix} -\alpha & \alpha & 0 \\ \beta & -\alpha-\beta & \alpha \\ 0 & \beta & -\beta \end{pmatrix} \tag{9}$$

となる。定常状態での各状態の確率分布を解析するには(7) 式、(8) 式から成る連立方程式を解けばよい。これに対して過渡解析を行なうためには(6) 式の線形微分方程式を解く必要がある。解法として二つ考えられる。

シミュレーション:
発火確率分布関数に基づいてトークン挙動のシミュレーションを複数回行ない、その平均を取る。

微分方程式を解く:
行列計算を利用して線形微分方程式を解く。

前者は、SPN では発火時間がランダムなため、莫大な回数の試行が要求され現実的ではない。

それに対して後者は、直接微分方程式を解いて解となる式を導出する。従って解式から、どの様な指数関数の影響が支配的なのか、永久に振動するのか等の結果の性質も調べることができる。しかし行列計算を多数必要とするため時間がかかる。

本研究では後者の方法を採用しその計算コストを減らす工夫をする。

実際に (6) 式の微分方程式を解くと

$$\boldsymbol{\pi}(t) = \boldsymbol{\pi}(0)e^{\boldsymbol{Q}t} \tag{10}$$

となる。

これを計算するためには行列の固有値を求めて対角化 (もしくはジョルダン標準形化) する必要がある。ここで $\boldsymbol{D}$を固有値を対角要素とする対角行列、$\boldsymbol{P}$ を固有ベクトルを列の要素とする正方行列とすると

$$\boldsymbol{Q} = \boldsymbol{P}^{-1}\boldsymbol{D}\boldsymbol{P} \tag{11}$$

$$= \boldsymbol{P}^{-1}\begin{pmatrix} \lambda_1 & 0 & 0 & \cdots \\ 0 & \lambda_2 & 0 & \cdots \\ 0 & 0 & \lambda_3 & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}\boldsymbol{P} \tag{12}$$

となる。これを利用すると、

$$e^{\boldsymbol{Q}t} = \boldsymbol{P}^{-1}e^{\boldsymbol{D}t}\boldsymbol{P} \tag{13}$$

$$= \boldsymbol{P}^{-1}\begin{pmatrix} e^{\lambda_1 t} & 0 & 0 & \cdots \\ 0 & e^{\lambda_2 t} & 0 & \cdots \\ 0 & 0 & e^{\lambda_3 t} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix}\boldsymbol{P} \tag{14}$$

となって実際に計算することができる。

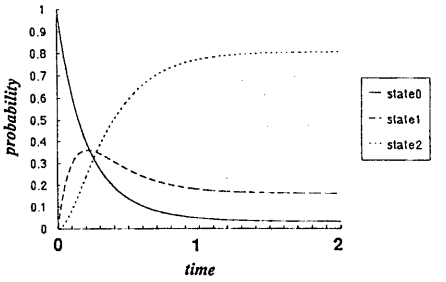

図 6: グラフ

この手法を用いて図4の CTMC を解析すると図6のグラフの様になる。ここでは $\alpha = 1, \beta = 5$ としている。

この結果を利用すると SPN を解析することができる。例えばリソースが一つだけ使用されている状態 (state1) の確率が最も高くなるのは時間軸で 0.2 の時であり、その後少し下がって定常状態になることがわかる。

4 線形性に基づいた分割

SPN の解析は前述の方法を行なえば良いが

- 固有値の計算に QR 分解が必要である
- 固有ベクトルの計算のため固有値の数だけ連立方程式を解く必要がある

このため計算時間が非常に大きくなる。一般に行列の計算には $O(n^3)$ の計算コストが必要になるため、分割して解析できた場合効果は大きい。ここでは線形性に基づいた分割法を考えてみる。

(6) 式のように線形微分方程式で表すことができることから CTMC は線形である。もし CTMC が線形システムとみなせれば複数の CTMC を線形システムの規則に従って結合するすることができる。しかし、線形システムでは入力と出力を定義する必要がある。

ここではまず準備として線形システムの記述を説明し、その後、入力と出力の定義も含めて線形性に基づく分割法を説明する。

4.1 線形システムの記述

線形システムでは$\boldsymbol{\pi}(t)$ を内部状態のベクトル、$\boldsymbol{u}(t)$ を入力のベクトル、$\boldsymbol{y}(t)$ を出力のベクトル, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$を定数の行列として

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{\pi}(t)\boldsymbol{A} + \boldsymbol{u}(t)\boldsymbol{B} \tag{15}$$

$$\boldsymbol{y}(t) = \boldsymbol{\pi}(t)\boldsymbol{C} + \boldsymbol{u}(t)\boldsymbol{D} \tag{16}$$

と表せる。これを線形システム方程式と言う [Huruta 1991]。この時ラプラス変換 ($\mathcal{L}\{\}$) によって

$$\boldsymbol{H}(s) = \mathcal{L}\{\boldsymbol{H}(t)\} \tag{17}$$

$$= \boldsymbol{B}(s\boldsymbol{I} - \boldsymbol{A})^{-1}\boldsymbol{C} + \boldsymbol{D} \tag{18}$$

伝達関数 $\boldsymbol{H}(s)$ が計算できる。これを利用することにより、例えばシステム1の伝達関数を $\boldsymbol{F}(s)$、システム2の伝達関数を $\boldsymbol{G}(s)$ とすると、システム1とシステム2の直列結合から成るシステムの伝達関数は $\boldsymbol{H}(s)$ は

$$\boldsymbol{H}(s) = \boldsymbol{F}(s)\boldsymbol{G}(s) \tag{19}$$

で計算でき、

$$e^{\boldsymbol{Q}t} = \mathcal{L}^{-1}\{\boldsymbol{H}(s)\} \tag{20}$$

によって、システム1とシステム2の直列システムの解が得られる。この場合 $\mathcal{L}\{\boldsymbol{H}(t)\}$ と $\mathcal{L}^{-1}\{\boldsymbol{H}(s)\}$ の計算が問題となる。

4.2 CTMC の線形システム表現

CTMC 自体は線形であるからどのように分割しても線形システムとして解析できる。本研究ではなるべく SPN のネットの分割に対応できるような CTMC の分割法を考えてみる。

まず、SPN のプレースを A, B 二つのグループに分ける。グループ A とグループ B の間を移動するいくつかのトークンに注目すると、対応する CTMC の状態は、

システム 1: 注目する全トークンがグループ A にある状態 (状態数 k)

システム 2: 注目するトークンがグループ A にもグループ B にもある状態 (状態数 l)

システム 3: 注目する全トークンがグループ B にある状態 (状態数 m)

の三つに分けられる。ちなみに全状態数は $n = k + l + m$ である。これから生成行列 $\boldsymbol{Q}$ は

$$\boldsymbol{Q} = \begin{pmatrix} \boldsymbol{Q}_{11} & \boldsymbol{Q}_{12} & \boldsymbol{O} \\ \boldsymbol{Q}_{21} & \boldsymbol{Q}_{22} & \boldsymbol{Q}_{23} \\ \boldsymbol{O} & \boldsymbol{Q}_{32} & \boldsymbol{Q}_{33} \end{pmatrix} \tag{21}$$

と部分行列に分割できる。ここでシステム2において各部分行列を説明すると

- $\boldsymbol{Q}_{21}$ はシステム2の出力行列およびシステム1の入力行列 $(l \times k)$
- $\boldsymbol{Q}_{22}$ はシステム2の内部状態の遷移行列 $(l \times l)$
- $\boldsymbol{Q}_{23}$ はシステム2の出力行列およびシステム3の入力行列 $(l \times m)$

である。

これらから全システムを記述すると図7の様になる。

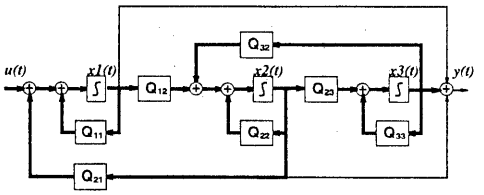

図 7: Q の線形システム

各部を説明すると $\int$ の所でシステム1の内部状態 $x_1(t)$、システム2の内部状態 $x_2(t)$、システム3の内部状態 $x_3(t)$、を計算している。入力はシステム1に与えている。これは初期状態がシステム1にあると仮定していることによる。システム2もしくはシステム3に初期状態がある場合には、状態番号を付け換えてシステム1に初期状態が存在するようにする。全てのシステムの内部状態は最初 0 としておき、初期状態を δ 関数でインパルスとして与える。例えばシステム1の状態数が3であってそのうちの2番目状態が初期状態の場合には、入力として $(0, \delta, 0)$ 与える。実際に計算する場合にはラプラス変換して与えるので $\mathcal{L}\{(0, \delta, 0)\} = (0, 1, 0)$ で計算することになる。出力は各システムの内部状態をそのまま出力としている。このようにして、全体としては初期状態を入力とし、全内部状態を出力とする一つのシステムになっている。

このシステムの内部状態について微分方程式を作ると

$$\dot{\boldsymbol{x}}_1(t) = \boldsymbol{x}_1(t)\boldsymbol{Q}_{11} + \boldsymbol{x}_2(t)\boldsymbol{Q}_{21} + \boldsymbol{u}(t) \tag{22}$$

$$\dot{\boldsymbol{x}}_2(t) = \boldsymbol{x}_1(t)\boldsymbol{Q}_{12} + \boldsymbol{x}_2(t)\boldsymbol{Q}_{22} + \boldsymbol{x}_3(t)\boldsymbol{Q}_{32} \tag{23}$$

$$\dot{\boldsymbol{x}}_3(t) = \boldsymbol{x}_2(t)\boldsymbol{Q}_{23} + \boldsymbol{x}_3(t)\boldsymbol{Q}_{33} \tag{24}$$

$$\boldsymbol{y}(t) = (\boldsymbol{x}_1(t), \boldsymbol{x}_2(t), \boldsymbol{x}_3(t)) \tag{25}$$

となる。これをラプラス変換すると

$$s\boldsymbol{X}_1(s) = \boldsymbol{X}_1(s)\boldsymbol{Q}_{11} + \boldsymbol{X}_2(s)\boldsymbol{Q}_{21} + \boldsymbol{U}(s) \tag{26}$$

$$s\boldsymbol{X}_2(s) = \boldsymbol{X}_1(s)\boldsymbol{Q}_{12} + \boldsymbol{X}_2(s)\boldsymbol{Q}_{22} + \boldsymbol{X}_3(s)\boldsymbol{Q}_{32} \tag{27}$$

$$s\boldsymbol{X}_3(s) = \boldsymbol{X}_2(s)\boldsymbol{Q}_{23} + \boldsymbol{X}_3(s)\boldsymbol{Q}_{33} \tag{28}$$

解析する場合にはこれを各状態 $\boldsymbol{X}_1(s), \boldsymbol{X}_2(s), \boldsymbol{X}_3(s)$ について解いて逆ラプラス変換するわけだが、例えば $\boldsymbol{X}_2(s)$ について解くと

$$\boldsymbol{X}_2(s) = \boldsymbol{U}(s)(s\boldsymbol{I} - \boldsymbol{Q}_{11})^{-1}\boldsymbol{Q}_{12}(s\boldsymbol{I} - \boldsymbol{Q}_{22} - \boldsymbol{Q}_{21}(s\boldsymbol{I} - \boldsymbol{Q}_{11})^{-1}\boldsymbol{Q}_{12} - \boldsymbol{Q}_{23}(s\boldsymbol{I} - \boldsymbol{Q}_{33})^{-1}\boldsymbol{Q}_{32})^{-1} \tag{29}$$

となってしまい、二重の逆行列を求めなければならずこの式を逆ラプラス変換するのは現実的ではない。

4.3 計算可能な分割法

線形システムにおいて逆行列の演算子が出てくるのはフィードバックである。図8のフィードバックに対応する伝達関数は

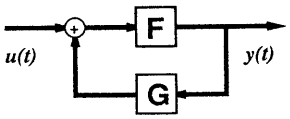

図 8: フィードバック

$$H = F(I - GF)^{-1} \tag{30}$$

である。このフィードバックを取り除けば逆行列の計算回数を減らせる。図7でフィードバックは Q_{21} と Q_{21} である。この二つが O ならば計算可能な式となる。フィードバックを取り除いた新しいシステムと Q は次のようになる。

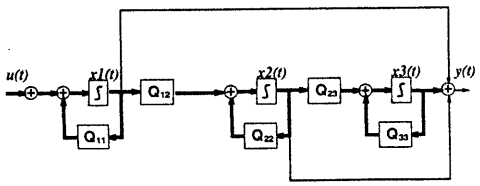

図 9: 新しい Q の線形システム

$$Q = \begin{pmatrix} Q_{11} & Q_{12} & O \\ O & Q_{22} & Q_{23} \\ O & O & Q_{33} \end{pmatrix} \tag{31}$$

この変更と共にグループの分け方にも変更が必要である。グループ A からグループ B へはトークンが移動可能だがその逆は不可というルールを設ける。この分け方は定常解析を行なう場合にも

$$e^{Qt} \underset{t\to\infty}{=} \begin{pmatrix} O & O & O \\ O & O & O \\ O & O & e^{Q_{33}(\infty)} \end{pmatrix} \tag{32}$$

となり、定常解は Q_{33} の行列のみから求められることがわかり有用である。この新しい線形システムから微分方程式を作ると

$$\dot{x}_1(t) = x_1(t)Q_{11} + u(t) \tag{33}$$
$$\dot{x}_2(t) = x_1(t)Q_{12} + x_2(t)Q_{22} \tag{34}$$
$$\dot{x}_3(t) = x_2(t)Q_{23} + x_3(t)Q_{33} \tag{35}$$
$$y(t) = (x_1(t), x_2(t), x_3(t)) \tag{36}$$

となる。これをラプラス変換して $X_1(s)$, $X_2(s)$, $X_3(s)$ について解くと

$$X_1(s) = U(s)(sI - Q_{11})^{-1} \tag{37}$$
$$X_2(s) = U(s)(sI - Q_{11})^{-1}Q_{12}(sI - Q_{22})^{-1} \tag{38}$$
$$X_3(s) = U(s)(sI - Q_{11})^{-1}Q_{12}(sI - Q_{22})^{-1} Q_{23}(sI - Q_{33})^{-1} \tag{39}$$

となる。これらの式は逆ラプラス変換することができる。例えば $X_1(s)$ の場合だと

$$\begin{aligned} X_1(s) &= U(s)(sI - Q_{11})^{-1} \\ &= U(s)(sI - P^{-1}D_{11}P)^{-1} \\ &= U(s)(P^{-1}(sI - D_{11})P)^{-1} \\ &= U(s)P^{-1}(sI - D_{11})^{-1}P \\ &= U(s)P^{-1}\begin{pmatrix} \frac{1}{s-\lambda_1} & 0 & 0 & \cdots \\ 0 & \frac{1}{s-\lambda_2} & 0 & \cdots \\ 0 & 0 & \frac{1}{s-\lambda_3} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{pmatrix} P \end{aligned} \tag{40}$$

となる。

ここで $(sI - D_{11})^{-1}$ は s の1次の分数式を対角要素に持つ対角行列である。この中でもっとも複雑な式である $X_1(s)$ でも s の3次の分数式の和となり部分分数に展開して逆ラプラス変換することができる。

入力の$U(s)$ はどれか一つが1である行ベクトルであった。これを乗ずることにより、複数(システム1内の状態数) 出てきた s の式の一つをインパルス応答によって選択している。つまり初期状態になる可能性のある状態の数だけ s の式が作られているわけである。

5 評価

ここでは線形システムに基づいて分割した時の状態数から解析に必要な計算コストを予測して、分割の効果を評価してみたい。

例として図 10 のシステムを解析した場合を想定して見る。

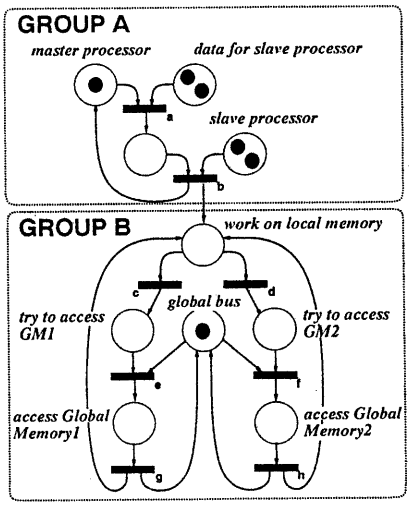

図 10: マルチプロセッサの SPN 表現

このシステムは一つのマスタープロセッサとローカルメモリを持つ二つのスレーブプロセッサ、およびグローバルバスでつながった二つのグローバルメモリからなるマルチプロセッサシステムである。

まず、マスタープロセッサはスレーブプロセッサに与えるデータをディスク上から取ってくる。これが発

火係数 a のトランジションである。スレーブプロセッサはデータを受けとるとローカルメモリ上で処理を開始する。これが発火係数 b のトランジションである。ここまでの処理を GROUP A とする。

次にローカルメモリ上で処理をしているスレーブプロセッサは適当な確率分布に従ってグローバルメモリ1またはグローバルメモリ2をアクセスしに行く。これは発火係数 c と d のトランジションである。グローバルメモリのアクセスはグローバルバスが空いている場合のみ成功し、空いていなければ待たされる。これが発火係数 e と f のトランジションである。最後にグローバルメモリのアクセスが終了するとグローバルバスを解放し、ローカルメモリ上の処理に戻る。これが発火係数 g と h のトランジションである。この部分を GROUP B とする。

この SPN で記述されたシステムを使って、分割しない場合と分割した場合の計算コストを比較してみる。まず前述の方法に従って三つのシステムに分割する。ここではスレーブプロセッサのトークンに注目して、二つとも GROUP A にある場合をシステム1、GROUP A と GROUP B に一つずつある場合をシステム2、二つとも GROUP B にある場合をシステム3とする。注目しているトークンは GROUP B から GROUP A へ戻ってくることはないのでフィードバックは起こらない。

それぞれのシステムの状態数とその状態数で過渡解析を行なった時の実行時間を表1にしめす。

表 1: 状態数と解析時間

	状態数 (n)	ループ	時間	時間/n^3
システム1	5 個	88 回	0.8 秒	0.0064
システム2	7 個	127 回	1.2 秒	0.0035
システム3	12 個	223 回	5.3 秒	0.0031
全システム	24 個	466 回	73.8 秒	0.0053

計測に使用したツールは SPN の記述を与えると CTMC に分解し、その CTMC から生成行列 Q を作って、Q の固有値を求め対角化を行い解式を求めることができる。解析時間の測定に使用したのは最も時間のかかる QR 分解で固有値を求め対角化を行なう部分である。他の部分の処理時間はどれも1秒程である。

ここでループ数は QR 分解のループが何回で収束したかを示している。今回の解析では固有値の計算に QR 分解を使用しており、収束した時に対角要素に固有値が並ぶという性質利用している。この収束速度は与える行列に強く依存し、また固有値が重根であると極端に遅くなるので正確な比較はできないが、なるべく重根を含まない条件で測定を行なった。

行列の計算コストはだいたい $O(n^3)$ であり、この結果もそれに近いものがある。この結果から単純に分割しない場合とした場合の解析時間の比較をすると表(2)のようになる。

表 2: 解析時間の比較

通常解析	分割解析
73.8 秒	7.3 秒

この差は状態数 (n) の数が増えると急激に増大する。逆ラプラス変換に多少時間がかかったとしても分割した場合のメリットは大きいことがわかる。

6 今後の課題

理論的な大きな課題はフィードバックに対応することである。そのためには(29)式を計算機上で解く必要がある。また本研究では SPN のネットを二つに分けて CTMC を3分割したが、理論的には CTMC を何分割しても線形システムとして解くことができ、もっと効率的な分割方法が存在する可能性がある。

現実的な課題は SPN を分割して解析する ツール を実装することである。現在のところ、線形システムに基づいて SPN を自動的に分割解析するシステムはできていない。実装したツールで実際に計算コストを比較しないければ正確な評価は下せない。

7 参考文献

[Marsan 1984]: M.Ajmone Marsan:*A Class of Generalized Stochastic Petri Nets for the Performance Analysis of Multiprocessor Systems*: ACM Transactions on Computer Systems, Vol 2, May 1984

[Marsan 1989]: M.Ajmone Marsan:*STOCHASTIC PETRI NETS:AN ELEMENTARY INTRODUCTION*:Advances in Petri Nets 1989 Lecture Notes in Computer Science

[Holliday 1985]: Mark A. Holliday and Mary K. Vernon:*A Generalized Timed Petri Net for Performance-Analysis*:International Workshop On Time Petri Nets 1985

[Sanders 1986]: W.H.Sanders and J.F.Mayer: *METASAN:A performability evaluation tool based on Stochastic Activity Networks*:Proceedings of the ACM-IEEE Comp. Soc. Fall joint Comp. Conf., Nov. 1986

[Chiola 1988]: Giovanni Chiola:*A Software Package for*

the Analysis of Genaralized Stochastic Petri Net Models:IEEE International Workshop on Timed Petri Nets 1985

[Ciardo 1989]: Gianfranco Ciardo, Jogesh Muppala, Kishor Trivedi:*SPNP:Stochastic Petri Net Package*:IEEE The 3rd International Workshop On Petri Nets and Performance Models 1989

[Huruta 1991]: 古田 勝久, 佐野 昭:基礎システム理論:コロナ社