計算機アーキテクチャ 100－3
（1993. 6.11）

並列計算機の解析モデル
－ シミュレーションとの比較 －

城 和貴† 福田 晃†◇

† 奈良先端科学技術大学院大学情報科学科
◇ 九州大学工学部情報工学科

Abstract

キャッシュを有した共有メモリ型並列計算機に対する、セミ・マルコフ過程を利用した解析モデルを提案する。提案されるモデルはコヒーレンス制御やネットワーク競合による待ち状態を、実際の並列プログラムに特有のパラメータを与えることにより、容易に記述できる。また、並列計算機自体のシステム構成も簡単に変更できるため、さまざまなシステム構成を持つ並列計算機の、アプリケーションごとの性能評価を容易かつ詳細に得ることができる。さらに、構築されたモデルを用いて、プロセッサ利用率、通常データ／コヒーレンス制御リクエスト及びそれらの待ち時間について実際の評価を行なう。その結果、実際のシミュレーション結果と較べて4.36%の違いしかないにもかかわらず、計算時間は普通のワークステーションでわずか0.01秒程度であった。

An Analytic Model for a Multiprocessor Computer
－ Comparison with a Simulation －

Kazuki JOE Akira FUKUDA

Graduate School of Information Science
Advanced Institute of Science and Technology, Nara
8916-5, Takayama, Ikoma, Nara 630-01 Japan

E-mail: {kazuki-j, fukuda}@is.aist-nara.ac.jp

Abstract

In this paper, we propose an analytic model using a semi-markov process for a parallel computer with cache memory. The proposed model can be used for the performance evaluation or analysis of a parallel computer with cache memory because it can be easily applied to describing the cache coherence control and the waiting states for network arbitrations or memory contention. Furthermore, the model is so flexible that it allows various kind of system parameters and it offers efficient input parameters. Using the proposed analytic model, we investigate some comparative experiments with widely known simulation. The result shows that there is only 4.36% difference between the actual simulation and our analytic model while the analytic model can evaluate a 1,024 processor system within $\frac{1}{100}$ second.

1 はじめに

並列計算機はキャッシュ・コヒーレンス制御 [4]、[1]、クラスタ方式による大規模化 [13]、スケーラブルな共有メモリ方式 [9] 等、さまざまな研究がなされ、より大規模に、より複雑に進化しつつある。

このような大規模で複雑な並列計算機アーキテクチャの妥当性を確かめるのに、シミュレーションは有効な手法である反面、数万数十万規模のプロセッサの動きを検証するのに要するコストは、新たな問題になるであろう。

一方、並列計算機の性能評価を行なうのに、確率過程等を用いた解析モデルは低コストで有効な手法である。しかしながら、これまでに提案されてきた並列計算機システムの解析モデルでは、キャッシュを持ったアーキテクチャは対象にされていない研究 [5][7][19][2][3][15]、リクエストに対するネットワーク競合等の待ち状態が表現されていない研究 [22] 等が主で、実際のシミュレーションに迫る詳細な解析を可能とするような解析モデルは極めて少ない。

筆者らは既に、特定の並列計算機に対するこのような解析モデルを提案している [25][24]。しかしながら、[25][24] で得られた解析結果は、実際の定量的性能評価とどの程度の差異があるか分からない、という問題があった。そこで本稿では、実際のシミュレーション・モデル [4] を見据えた、並列計算機に対する一般的なモデル化を与え、キャッシュ・コヒーレンス・プロトコルとして Synapse プロトコルを例に取った、具体的な解析モデルを構築し、文献 [4] と全く同条件で性能評価を行い、得られた結果を比較する。

本稿では、まず関連する研究について簡単に述べた後、全体のモデル化のための仮定を述べ、プロセッサ・リクエスト率の分類の定義を与える。次に、共有ブロック [8] に対するキャッシュ・ヒット率と全体の共有ブロック数から特定の共有ブロックのキャッシュ中の存在確率を求める手法を示す。これらの準備の後、モデル化のための状態記述の定義を行い、内部変数の計算方法を示し、具体的な計算手順を示した後、実際の性能評価を求める。さらに、既存のシミュレーション結果 [4] との比較を行い、得られた解析結果を考察する。

2 関連する研究

キャッシュを持たない共有メモリ型並列計算機のネットワークの性能評価として、文献 [14] のクロスバーおよびマルチプル・バスに対するものがある。その結果、マルチプル・バスの本数がプロセッサ数の半分である時に、その性能がクロスバーと同等になることが、シミュレーションを元に報告されている。同様の結果は、確率モデルを用いた解析モデルによって文献 [5] で報告されている。マルチプル・バスとクロスバーの性能評価は、文献 [7] で確率モデルを使ったものと、マルコフ連鎖モデルを用いたものが報告されている。さらに、マルチプル・バスについては、Mudge らが多方面からの解析を研究している [17][18][19]。

文献 [16] では、並列計算機のような離散時間イベントを扱うモデルにおいて、連続時間モデルに対する離散時間モデルの優位性が示されている。ここで提示されたマルコフ連鎖モデルは、文献 [17] でセミ・マルコフ過程を用いることにより、簡略化されている。文献 [17] は、プロセッサの待ち時間を正確にモデル化した最初のものである。モデル化にあたって使用されるパラメータとしては、プロセッサ数、メモリ数、マルチプル・バスの本数、リクエスト時間等があり、当時の最新のアーキテクチャの解析モデルとして、評価が高かったものと思われる。このアプローチは最近、文献 [12] に引き継がれ、リクエストにプライオリティを持った並列計算機のモデル化を試みている。しかしながら [12] は、キャッシュを持たない単一のクロスバー・ネットワークにより結合されたアーキテクチャのモデル化であり、むしろＯＳや並列コンパイラ等、スケジューリングを考慮しなければならないシステムが対象と言えよう。

キャッシュを持つシステムの性能評価としては、文献 [8] において本格的なモデルが提案され、以後それを利用したシミュレーションや解析モデルが多数提案されている。

Dubois らは、コヒーレンス制御命令を引き起こすのは、複数のプロセッサによって共有されるデータにアクセスする時に限ることに着目し、キャッシュを共有データを扱う共有ブロックと、ローカルのアクセスだけでよいプライベート・ブロックに分けるモデルを提案している。さらに、そのモデルを用いて共有ブロックに相当するブロックの挙動をマルコフ連鎖を用いて表し、キャッシュのヒット率とキャッシュの大きさの関係等について報告している [8]。この研究は後に、Dubois、Wang らによって、アクセス・バースト・モデルという、実際の並列プログラムに対応した並列計算機の解析モデルとして報告されている [22]。

シミュレーションとしては、文献 [6][4] がある。前者はアドレス・トレースによるものであるが、後者は Dubois のモデル [8] を利用したパラメトリックなシミュレータであり、キャッシュのプライベート・ブロックは単純な確率モデルを、共有ブロックには正確なシミュレーションを採用し、さまざまなコヒーレンス・プロトコルの性能評価を与えている。この両方の手法を融合したシミュレータも、文献 [23] で提案されている。

解析モデルとしては、Archibold のシミュレーション [4] 同様、さまざまなコヒーレンス・プロトコルの性能評価を与えた文献 [22] がある。ここでは、キャッシュは無限にあるという仮定のもとで、マルコフ連鎖を用いて、各プロトコルの定常状態におけるキャッシュのミス率を求めている。ただし、このモデルでは、リクエストの待ち状態およびキャッシュのリプレースが考慮されておらず、そのため、プロセッサ利用率も求めることもできない。

文献 [21] では、ライト・ワンス・プロトコルを元にしたベーシック・プロトコルとその改良プロトコルに対するモデル化を、Generalized Timed Petri Nets を使って表している。このモデルはキャッシュを持つアーキテクチャを対象にしており、リクエストに対する待ち時間も評価されるが、全体のプロセッサ台数が増えるに従い、状態数が爆発的（プロセッサ数 1 の時には状態数 33 が、プロセッサ数 10 に対しては状態数 41,159 になる。）に増える。従って、我々の目標である、大規模な並列計算機システムに対する安価な解析モデルとは異なるアプローチである。

3 準備

3.1 モデル化の基本方針

通常、キャッシュのない並列計算機のモデル化を行なう場合、あるプロセッサに着目して、状態遷移を考え、それと等価なものが複数あると仮定して、全体のモデルを構築する [17][12]。ところが、図 1 のように、キャッシュを持ちコヒーレンス制御を行なう並列計算機に対しては、このようなモデル化の試みは難しい。例えば、注目しているあるプロセッサが、あるサイクルにおいて、ライト・リクエストを発行したとすると、それに対応するシステム全体の挙動は、そのサイクルにおける、ライト要求されたデータ・ブロックの状況に応じて異なる。つまり、要求されたデータ・ブロックが、1）他のどのキャッシュにも存在しなければ、通常のライト・ミスに、2）複数のキャッシュにクリーンな状態で存在すればインバリデーション命令を伴う、ライト・ヒットもしくはライト・ミスに、3）既に自分のキャッシュ中でモディファイされている場合は単なるライト・ヒットに、それぞれ推移するわけである。このような推移を記述するには、各サイクルにおいて共有ブロックがどのような状態（キャッシングされている／いない、クリーン／クリーンではない）にあるかを予測しなければ、本来のプロセッサ推移状態を記述することは困難である。

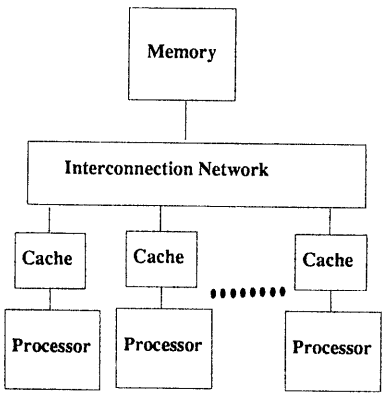

図 1: Multiprocessor with Private Cache

Wang らは、この問題に対し、キャッシュは無限にあるという仮定のもとで、共有ブロックに対する解析モデルを提案し、さまざまなコヒーレンス・プロトコルにおけるキャッシュのミス率を出している [22]。

そこで本稿では、Wang らのモデルにより導き出される共有ブロックに対するキャッシュ・ミス率を利用することにより、上記予測に対する近似を行い、次に、この予測を用いて、セミ・マルコフ過程を利用したプロセッサの状態推移を定義し、キャッシュ制御およびリクエストの待ち時間を考慮に入れた解析モデルを構築する。得られたモデルの正当性は、Archibald のシミュレーション [4] 結果との比較によってなされる。

なお、本稿ではキャッシュ・コヒーレンス・プロトコルとして Synapse [4] を採用しているが、その理由はプロトコル自体の状態数の少なさだけである。従って、他プロトコルでも容易に同様の解析モデルを構築可能である。

3.2 モデルの仮定

本稿では、モデルを簡略化するために、次のような仮定を導入する。

1. 各プロセッサの挙動は等価な確率過程でモデル化される。
2. 全てのプロセッサからのリクエストが全て独立であるような並列プログラムが稼働中であるとする。言い換えれば特殊な同期操作は本モデルでは記述出来ないことになる。
3. プロセッサの状態は、計算を行なっている状態、ネットワークを使ってリクエストを実行している状態、その状態への待ち状態と考え、それぞれ独立した確率変数で与えられる。ただし、本稿においては、それらの平均値のみを使用する。
4. 他のプロセッサの状態に影響を与えるようなリクエストは、その作用が自分自身に影響を与えるものと解釈する。
5. プログラムは安定稼働している状態、すなわち、起動時のページングやキャッシュ・ヒット率の悪さは考慮しない状態を仮定する。またキャッシュは既に使い切った状態で、それに伴うリプレースについては考慮する。

3.3 セミ・マルコフ過程

マルコフ連鎖による並列計算機システムのモデル化は、モデルの簡潔さという利点がある反面、各状態の時間が一定であるという制約があるため、プロセッサからのリクエストに対する待ち状態を記述しにくいという欠点がある。これに対し、セミ・マルコフ・モデルは各状態の時間が任意に設定出来るため、キャッシュ・コヒーレンス制御を含む複雑なリクエストが発生しうる共有メモリ型並列計算機システムのモデル化に適するものと思われる。

セミ・マルコフ過程（以後ＳＭＰと呼ぶ）に関する詳細は文献 [20] に譲るとして、ここでは本稿で用いるＳＭＰについての簡単な説明を行う。

本稿においてのＳＭＰは K 個の状態からなる離散的確率過程である。状態 i においては平均 η_i の間滞在し、状態 j に p_{ij} の確率で遷移する[1]。もしＳＭＰがエルゴード的な既約なエンベデッド・マルコフ連鎖（以後ＥＭＣと呼ぶ）を持つなら、状態 i の定常分布 P_i は次式で表される。

$$P_i = \frac{\pi_i \eta_i}{\sum_{j=1}^{K} \pi_j \eta_j} \quad (1)$$

ただし、$\{\pi_i\}$ はＥＭＣの定常分布である。本稿におけるモデルでは、後に述べるようにエルゴード的な成分が 1 つだけからなるＥＭＣで表されているので、常に（式 1）が利用できる。

次にＳＭＰの状態 i から脱出する確率 λ_i を定義する。これは Mudge の提案したもの [17] を利用する。

$$\lambda_i = \frac{P_i}{\eta_i} = \frac{\pi_i}{\sum_{j=1}^{K} \pi_j \eta_j} \quad (2)$$

状態 i を脱出するレート λ_i は、その平均滞在時間の平均値の逆数で表される。本稿においては、平均滞在時間は少なくとも 1 システム・サイクル・タイ

[1] 各状態の滞在時間がすべて 1 である時、そのＳＭＰは通常のマルコフ連鎖と等価なことに注意されたい。

ムであるため、λ_iは$[0,1]$に含まれる。従って、λ_iはレートとしても確率としても使うことができる。

3.4 リクエスト率の定義

並列計算機においては、プロセッサより出されるリクエスト率は、システムの稼働時に決まるものである。例えば、同じプログラムを走らせたとしても、ネットワークが飽和している状態でのプロセッサ・リクエスト率は、そうでない状態でのリクエスト率とは異なってくる。文献 [11] や [17] では、並列計算機の解析モデル中でのリクエスト率の修正方法について述べられている。これらの文献では、プロセッサ・リクエスト率は、実際のプログラムから決まるリクエスト率に加えて、プロセッサが待ち状態に陥った時に発行する再リクエストも、その割合に含めるとしている。

本稿では、対象とする並列計算機モデルが複雑なため、これまでのような簡単なプロセッサ・リクエスト率の概念では、その解析モデル化が困難である。そこで、本節では本モデル構築のために必要な、4種類のプロセッサ・リクエスト率の定義を行う。

- ノーマル・リクエスト率とは、プロセッサが計算状態の時に引き続いて発行されるリクエストの割合である。
- メモリ・リクエスト率とは、メモリ中のデータに対して発行されるリクエストの割合である。(ノーマル・リクエスト率) × (キャッシュ・ミス率) + (その待ち状態からのリクエスト率) で定義される。
- コヒーレンス・リクエスト率とは、コヒーレンス制御のためのリクエスト率およびその待ち状態からのリクエスト率の和で定義される。
- ネットワーク・リクエスト率とは、ネットワークに対して発行されるリクエストの割合。(メモリ・リクエスト率) + (コヒーレンス・リクエスト率) で定義される。

以上のプロセッサ・リクエスト率は、4章で構築するモデルの内部変数で使用される。本稿においては、それぞれのリクエスト率は、ネットワークのサイクル・タイムに対する、リクエストの割合とする。プロセッサのサイクル・タイムと、ネットワークのサイクル・タイムが極端に違う場合、リクエスト率は1を越える可能性がある。本モデルにおいては、この割合が1を越えないシステムのみを対象とする。従って、本節で定義したリクエスト率は、本稿では、確率としても使えることになる。

3.5 特定の共有ブロックのキャッシング確率

プロセッサからのリクエストが、コヒーレンス制御命令を引き起こすかどうかを知るには、リクエストされたブロックが他のキャッシュにキャッシングされているかどうかを予測しなければならない。これは、共有ブロックの総数Sと、その時点でキャッシングされている共有ブロック数Lを使って、$\frac{L}{S}$で表すことができる。

本稿では共有ブロックに対するキャッシュ・ヒット率Hに関しては、Wang [22] で与えられる Synapse プロトコルでのミス率を使って、

$$H = 1 - \frac{1}{ls}\frac{P(P-1)(1-r)}{(P-r)(1+(P-1)(1-r))} \quad (3)$$

とする。(Pはプロセッサ台数、rはリード・リクエスト率、lsはアクセス・バースト長。本モデルでは、$ls=2$とする。)

次に、キャッシングされている共有ブロック数は、単に共有ブロック総数と共有ブロックのキャッシュ・ヒット率を掛け合わしただけでは問題がある。なぜならば、共有ブロックに対するアクセス・パターンは一様とは限らないからである。そこで、本稿では、この共有ブロックに対するアクセス・パターンとして、Archibald [4] と同じものを使うことにする。

ある共有ブロックが、ＬＲＵスタック上において、i番目にある確率は、

$$g(\frac{1}{5+i} - \frac{1}{6+i}) \quad (4)$$

である [4]。ただし、gは正規化係数である。よって、Lは次の方程式で求まる。

$$\frac{\int_0^L g(\frac{1}{5+x} - \frac{1}{6+x})dx}{\int_0^S g(\frac{1}{5+x} - \frac{1}{6+x})dx} = H \quad (5)$$

式を簡単にして、Lは次で求められる。

$$L = \frac{6 \times \frac{5}{6}\left(\frac{6(5+S)}{5(6+S)}\right)^H - 5}{1 - \frac{5}{6}\left(\frac{6(5+S)}{5(6+S)}\right)^H} \quad (6)$$

4 モデル化

4.1 状態の定義

ＳＭＰを用いた並列計算機のモデル化を行うために、プロセッサの状態定義を行う。この状態定義はキャッシュ・コヒーレンス・プロトコルに従ってなされる。本稿では、Synapse プロトコルに沿ったものである。

ＣＯＭ	計算状態
Ｒｈ	キャッシュからの読み込み
Ｗｈ	キャッシュへの書き込み
ＷｈＩＶ	Ｗｈにより発生するインバリデーション
Ｒｃ	そのライン属性がダーティではないデータのキャッシュ・ミスによる読み込み
Ｒｄ	そのライン属性がダーティであるデータのキャッシュ・ミスによる読み込み
Ｗｃ	そのライン属性がダーティではないデータのキャッシュ・ミスによる書き込み
ＷｃＩＶ	Ｗｃにより発生するインバリデーション
Ｗｄ	そのライン属性がダーティであるデータのキャッシュ・ミスによる書き込み
ＷＢ	Ｗｄ、Ｒｄにより発生するライトバック
Ｒｐ	ラインのリプレースにより発生するライトバック

さらに、ネットワークに対してリクエストを発行する状態iに対して、$\bar{i}$でその状態への待ち状態を表すものとする。

4.2 状態遷移と定常分布

図 2は定義された状態間の遷移を示す。それぞれの状態間の遷移は、ネットワークもしくはキャッシュに対して、通常もしくはコヒーレンス制御のリクエストがなされた場合に起きる。図 2において、hはプライベート・ブロックに対するキャッシュのヒット率、Hは共有ブロックに対するキャッシュのヒット率、rはプロセッサからの通常のリクエスト

に対するリード・リクエストの割合（従って、$1-r$はライト・リクエストを意味することになる。）、dはリクエスト要求のあるデータを含むラインがダーティである確率、cはリクエスト要求のあるデータを含むラインがクリーンである確率、wは（ネットワークに対して）リクエストを出した時に待ち状態に陥る確率（この時、3.4節で定義したネットワーク・リクエスト率$\varphi_{network}$が必要となる）、xはキャッシュが既に一杯である確率、uは共有ブロックへのアクセス率（従って、$1-u$がプライベート・ブロックへのアクセス率となる。）、mはリプレース対象のラインが既にモディファイされている確率を意味する。図 2からも明らかなように、この状態推移をＥＭＣと見るとこれはエルゴード的であるので、その定常分布を求めることができる。

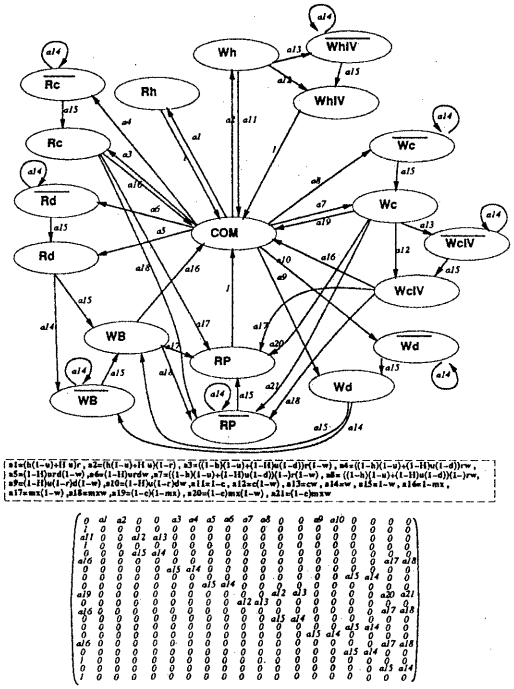

図 2: Synapseプロトコルに従ったプロセッサの状態遷移とその推移確率行列

ＥＭＣの遷移確率行列は上記以外の遷移の確率を０として、図 2のように表される。なお、各状態間の遷移についての説明は、文献 [25],[24] を参照されたい。

4.3 平均滞在時間

Rh	Wh	WhIV	Rc	Rd
1	1	1	8	8
Wc	WclV	Wd	WB	Rp
8	1	8	5	5

表 1: 各状態の平均滞在時間

先に述べたＥＭＣを用いてＳＭＰを形成するには、各状態の平均滞在時間が必要である。これについては、待ち時間以外は Archibald のシミュレーション [4] におけるシステム・モデルを採用する（キャッシュのブロック・サイズは４とする。）。勿論、この値を変更することにより、異なるスペックのアーキテクチャを、容易に解析することができる。一方、待ち状態に関しては、このような明確な値は出てこない。なぜなら、通常のデータ・リクエストに加えて、キャッシュ・コヒーレンス制御のためのリクエストも、各待ち状態に影響を与えるからである。この計算方法については4.4で述べることとする。ＣＯＭの平均滞在時間は、[4] では１から５までの一様乱数であるため、単純に平均を取って３とする。

ＳＭＰの定常分布は、ＥＭＣの定常分布 $\{\pi_i\}$ と、前節で定義した平均滞在時間 $\{\eta_i\}$ を使って、（式 1）から求められる [25][24]。また、定常状態における各状態からの脱出確率 [17] は、（式 2）で求められる。

4.4 モデル中の変数定義

このようにして定められたモデルを評価するには、モデルに対する適当なパラメータが必要である。本稿ではパラメータとして、h, H, r, u, P, S, mを考えることとする。なお、これらのパラメータは、Archibald のシミュレーション [4] に対するパラメータと同一である。ただし、Hは（式 3）で求まる。これらのパラメータで、変数$d, c, w, x, \varphi_i, Wt$を表す。

特定のラインが自分以外の少なくとも一つのキャッシュにおいて、クリーン状態であることを表す変数c、同じく特定のラインがダーティであることを示す変数dは、（式 3、 6）を使って次のように表される。

$$c = 1 - (1 - \frac{Lr}{S})^{P-1} \tag{7}$$

$$d = \frac{P-1}{P}\frac{L(1-r)}{S} \tag{8}$$

変数xは、あるプロセッサからのリクエストがミス・ヒットを起こし、キャッシュ・インした時に、キャッシュに空きがあるかどうかを決定する。仮定により、キャッシュを使い切った時には、他からのインバリデーションが来ないとキャッシュに空きは出来ない。よって、特定のキャッシュが他からインバリデーションを受ける確率をinvとすると、キャッシュに空きがない確率は、

$$x = (1 - inv)^{P-1} \tag{9}$$

となる。（本モデルではキャッシュを使い切った状態を考えているが、キャッシュを使い切っていない状態は$x = 0$とすることにより容易に記述できる。）あるプロセッサがインバリデーションを流す確率は、あるサイクルで共有ブロックへのライトのリクエストを発行し、そのラインが他のキャッシュにクリーン状態でコピーを持つか、もしくは、あるサイクルで共有ブロックへのリクエストがミス・ヒットに終り、そのラインがダーティ状態である場合である。よって、

$$inv = \varphi_{normal}u(1-r)c + \varphi_{normal}u(1-H)d \tag{10}$$

によってinvは求まる。

ネットワークに対してリクエストを出した時、待ち状態に陥る確率とは、そもそもネットワークが空いてなかった場合と、ネットワークは空いているが他のプロセッサからのリクエストとの競合に負けた場合とに分けて考えられる。まず、ネット

ワークが空いていない確率は次のようにして求める。ネットワークへのリクエストを出す状態の集合を Q とする。

$$Q = \{WcIV, Rc, Rd, Wc, WcIV, Wd, WB, Rp\} \quad (11)$$

あるプロセッサの状態が $i \in Q$ の時、次のサイクルでも引続き状態を変えない確率は $P_i - \lambda_i$ であるから、ネットワーク・ビジーである確率 $Busy$は、

$$Busy = (P-1)\sum_{i \in Q}(P_i - \lambda_i) \quad (12)$$

となる。ただし、P_iはＳＭＰの状態 i での定常分布、λ_iはその時の平均脱出確率（式 2）である。

次に、他のプロセッサからのリクエストとの競合に勝つ確率 Win は

$$Win = 1 - \frac{1-(1-\varphi_{network})^P}{P\varphi_{network}} \quad (13)$$

で求まる。なぜなら、あるサイクルでネットワークに対しリクエストが出されている確率は $1-(1-\varphi_{network})^P$、その時の同時リクエスト数は $P\varphi_{network}$、各リクエストは同じ割合でアクセプトされるからである。よって、待ち状態に陥る確率 wは

$$w = Busy + (1-Busy)(1-Win) \quad (14)$$

となる。

サイクル毎のリクエスト率φ_iは、次のようにして求める。まず、ノーマル・リクエスト率は定義より、

$$\varphi_{norm} = \lambda_{COM} \quad (15)$$

メモリ・リクエスト率は、ＣＯＭ状態からミス・ヒットのリクエストを出して脱出する割合と、メモリ・アクセスの待ち状態にあるときの脱出割合の和に等しい。待ち状態は、前述したようにネットワーク競合のための待ち状態と、ネットワーク・ビジーのための待ち状態とに分けられる。ネットワーク競合のための待ち状態の場合、脱出確率は λ_iを使い、ネットワーク・ビジーの待ちの場合、サイクル毎にリクエストを出すと考えると、結局メモリ・リクエスト率は次式で求まる。

$$\begin{aligned}\varphi_{memory} = {} & ((1-u)*(1-h)+u*(1-H))\lambda_{COM} + \\ & (1-Busy)(\lambda_{\overline{Rc}}+\lambda_{\overline{Rd}}+\lambda_{\overline{Wc}}+\lambda_{\overline{Wd}}) + \\ & Busy(P_{\overline{Rc}}+P_{\overline{Rd}}+P_{\overline{Wc}}+P_{\overline{Wd}}) \end{aligned} \quad (16)$$

コヒーレンス・リクエスト率は、コヒーレンス制御のためのリクエストを出す状態への脱出確率と、その待ち状態からの脱出確率との和に等しい。

$$\begin{aligned}\varphi_{coherence} = {} & c(\lambda_{Wh}+\lambda_{Wc}) + \\ & mx(\lambda_{Rc}+(1-c)\lambda_{Wc}+\lambda_{WcIV}+\lambda_{WB}) + \\ & (1-Busy)(\lambda_{\overline{WhIV}}+\lambda_{\overline{WcIV}}+\lambda_{\overline{WB}}+\lambda_{\overline{RP}}) + \\ & Busy(P_{\overline{WhIV}}+P_{\overline{WcIV}}+P_{\overline{WB}}+P_{\overline{RP}}) \end{aligned} \quad (17)$$

Wt は待ち状態にいる時、ネットワークが次に空くまでの平均時間であるから、

$$Wt = \sum_{i \in Q} \eta_i * P_i \quad (18)$$

で求められる。

4.5 モデルの計算手順

このようにして定義したモデルで、与えられたパラメータに対する定常分布を求めるには、非線形方程式を解かなければならない。なぜならば、解であるＳＭＰの定常分布 $\{P_i\}$ を求めるには、リクエスト率φ_jや待ち状態に対する平均滞在時間 Wt を求めなければならず、これらを求めるには、$\{P_i\}$ が求まっていなければならないからである。つまり、状態間の推移確率はＳＭＰの定常分布の関数として表されており、一方、ＳＭＰの定常分布は状態間の推移確率の関数として表されている。

この問題に対する、数学的な解の存在証明は困難 [10] であるため、本稿では、Mudge [17] と同様、イタレーションによる逐次近似法を用いて解を求めることにする。実際、我々の実験においては、以下に示す手順でイタレーティブに解を求めた場合、何ら収束加速のための特別なアルゴリズムを使わなくとも、数回から数十回のイタレーションで収束した。

1. パラメータとして、共有ブロック数 S、共有ブロック参照率 u、プライベート・ブロックのキャッシュ・ヒット率 h、リード・リクエストの割合 r、リプレース時のラインがモディファイされている率 m を与える。
2. ノーマル・リクエスト率φ_{normal}と平均待ち時間 Wt に適当な初期値（例えば、$\varphi_{normal} = \frac{1}{\eta_{COM}}$、$Wt$ には$\eta_i (i \in Q)$ の平均）を与える。
3. ＥＭＣの定常分布 $\{\pi_i\}$、平均滞在時間 $\{\eta_i\}$、各状態からの脱出確率 $\{\lambda_i\}$、ＳＭＰの定常分布 $\{P_i\}$ を計算する。
4. ネットワーク・リクエスト率$\varphi_{network}$と Wt を求める。
5. 前回の$\varphi_{network}$と新しく求めた$\varphi_{network}$の差が適当な値よりも大きければ 3 に戻る。

5 シミュレーションとの比較

本モデルの正当性を証明するため、Archibald のシミュレーション [4] 結果との比較実験を行なった。メモリ・アクセス・タイム等のシステム・モデルは、彼のシミュレーションと同じものを採用した。また、入力パラメータも同じものを用いた。

図 3, 4, 5 は、それぞれ共有ブロック数 Sが 16, 128, 1024 である時の、プロセッサ（キャッシュ）台数に対するシステム・パワー [4] を表している。与えたパラメータは、共有ブロック・アクセス率 $u = \{0.001, 0.05\}$、プライベート・ブロックに対するキャッシュ・ヒット率 $h = \{0.95, 0.98\}$、リード・リクエスト率 $r = \{0.85, 0.7\}$、ブロックのリプレース時のモディファイ率 $m = \{0.3, 0.4\}$ である。システム・パワーは、各プロセッサ利用率の総和に 100 を掛けたものである。

この実験において、我々の解析モデルと文献 [4] のシミュレーションとの誤差は 4.36%であった[2]。文献 [17] では、本論文と同様、セミ・マルコフ過程を用いた解析モデルとシミュレーション結果との比較を行なっているが、その誤差は『8%以内』である（プロセッサ利用率）。我々の解析モデルが文献 [17] と異なりキャッシュを持つ並列計算機といった複雑なシステムのモデル化であることを考えると、我々の解析モデルが極めて精密であることが分かる。

図 6は、$u = 0.05, h = 0.95, r = 0.85, m = 0.3$ において、共有ブロック数を変えた時の、プロセッサ台数 1 から 1,024 のシステム・パワーを表している。プロセッサ台数 8 程度でシステム・パワー最大値に達した後、共有ブロック数が 1,024 の場合は

[2] シミュレーション結果の値は、文献 [4] から注意深く読み取った。

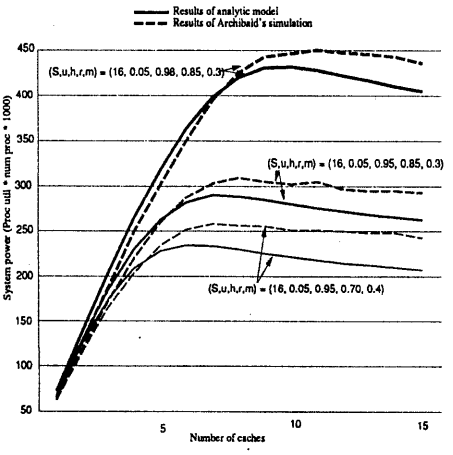

図 3: 共有ブロック数１６での比較

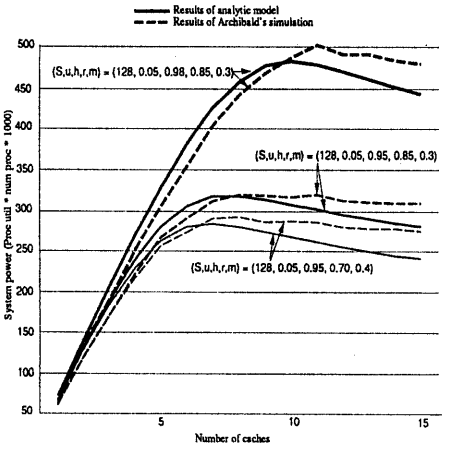

図 4: 共有ブロック数１２８での比較

プロセッサ台数 300 程度まで、それ以外の場合はプロセッサ台数 100 程度まで、プロセッサの増加と共にシステム・パワーは減少していき、その後、プロセサ台数と共有ブロック数に関係なく、システム・パワー約 225 という一定値に陥っている。

勿論、一本の共有バスに 1,024 台のプロセッサを付けた並列計算機が開発されるようなことはないだろう。この一連の実験は、我々の解析モデルの能力を示す一例である。

文献 [4] のシミュレーションは、各実験とも 25,000 システム・サイクル、実験全部で $10 * 15 * 25,000 = 3,750,000$ システム・サイクル走らせなければならないのに対し、我々の解析モデルでは、実験全部を行なうのに、卓上のワークステーションで約 1.0 秒しか必要としなかった。並列計算機のシミュレーションは、プロセッサ台数が増えるに従って実行時間が長くかかるのに対し、我々の解析モデルのようなアプローチでは、プロセッサ台数と実験時間はほぼ無関係である。実際、我々の解析モデルで 1,024 台のプロセッサによるシステム・パワーを評価するにも、$\frac{1}{100}$ 秒しか必要としない。それに比べて、1,024 台のプロセッサからなる並列計算機のシミュレーションがどれだけ高価か言うまでもないであろう。

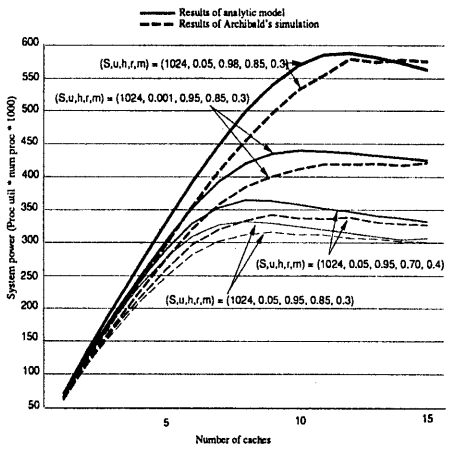

図 5: 共有ブロック数１０２４での比較

これら実験結果は、我々の解析モデルの妥当性と、計算コストの安さを証明するに十分であろう。

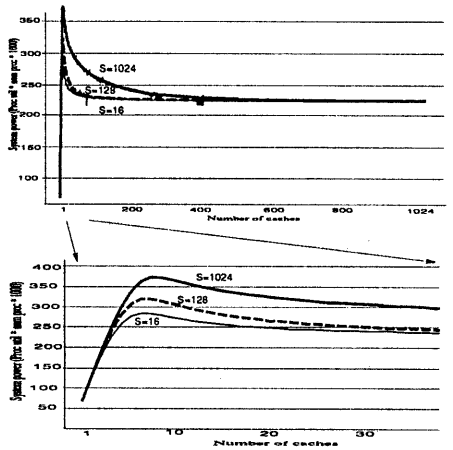

図 6: プロセッサ台数とシステム・パワー

6 結論

キャッシュを持つ並列計算機に対する、セミ・マルコフ過程を利用した解析モデルを提案した。モデル化にあたっては、まず、プロセッサ・リクエスト率の分類の定義を与え、次に、共有ブロックに対するキャッシュ・ヒット率と共有ブロック数から、特定の共有ブロックのキャッシュ中の存在確率を求め、さらに、Archibald のシミュレーション・モデルと同じ入力パラメータを与えられるようにした。構築されたモデルは、通常のメモリ・アクセスはもとより、キャッシュ・コヒーレンス制御のためのリクエストや、キャッシュ・ミスによって引き起こされるリプレースも考慮に入れたもので、さ

らにそれらの待ち状態もモデル化された。

この解析モデルで Synapse プロトコルに従う並列計算機の性能評価を、Archibald のシミュレーションと同じパラメータで行なったところ、4.36% の誤差しか現れず、各実験の計算時間は卓上ワークステーションでわずか$\frac{1}{100}$秒であった。また、同じパラメータで、プロセッサ数を極端に増やす実験を行なった結果、あるプロセッサ数に至れば共有ブロック数やプロセッサ数に無関係に、システム・パワーは変動しないことが判明した。

本モデルで採用した、共有ブロックに対するキャッシュ・ヒット率は、Wang の『無限にキャッシュ・エントリがある』場合のものであり、キャッシュのリプレースメントを前提とした本モデルとの差異は少なからずあるものと思われる。実際、共有ブロック数が少ない場合には、シミュレーション結果との誤差が比較的大きい。よって、今後の研究課題は、本モデルに適した共有ブロックに対するキャッシュ・ヒット率のモデル化である。また、定常分布を求める際の、収束性の保証、本解析モデルをツールとした様々なスペックの並列計算機の内部解析も、今後の重要な研究課題である。

謝辞

非線形システムについて御教授頂ったＡＴＲ人間情報研究所の佐藤雅昭主幹研究員に感謝致します。

参考文献

[1] A.Agarwal, Richard Simoni, John Hennessy, and Mark Horowitz. An evaluation of directory schemes for cache coherence. In *Proceedings of the International Symposium on Computer Architecture*, pages 280–289, 1988.

[2] Santosh G. Abraham and Edward S. Davidson. A communication model for optimizing hierarchical multiprocessor systems. In *Proceedings of the 1986 International Conference on Parallel Processing*, pages 467–474, 1986.

[3] Dharma P. Agrawal and Imadeldin O. Mahgoub. Performance analysis of cluster-based supersystems. In *Proceedings of the International Conference on Supercomputer System*, pages 593–602, 1985.

[4] James Archibald and Jean-Loup Baer. Cache coherence protocols: Evaluation using a multiprocessor simulation model. *ACM Transactions on Computer Systems*, 4(4):273–298, 1986.

[5] Laxmi N. Bhuyan. A combinatorial analysis of multibus multiprocessors. In *Proceedings of the 1984 International Conference on Parallel Processing*, pages 225–227, 1984.

[6] Craig B.Stunkel and W.Eent Fuchs. Analysis of hypercube cache performance using address traces generated by trapeds. In *Proceedings of the 1989 International Conference on Parallel Processing*, pages I–33–I–40, 1989.

[7] Chita R. Das and Laxmi N. Bhuyan. Bandwidth availability of multiple-bus multiprocessors. *IEEE Transactions on Computers*, 34(10):918–926, 1985.

[8] Michel Dubois and Faye A. Briggs. Effects of cache coherency in multiprocessor. *IEEE Transactions on Computers*, 31(11):1083–1099, 1982.

[9] Daniel E.Lenoski. The design and analysis of dash: A scalable directory-based multiprocessor. Technical Report CSL-TR-92-507, Stanford University, CSL, 1992.

[10] J. Guckenheimer and P. Holmes. *Non Linear Oscillations Dynamical Systems and Bifurcations of Vector Fields*. Springer-verlag, 1983.

[11] Kai Hwang and Faye A. Briggs. *COMPUTER ARCHITECTURE AND PARALLEL PROCESSING*. McGraw-Hill, 1985.

[12] Ashwani K.Ramani, Pradip K.Chande, and Pramod C.Sharmma. A general model for performance investigations of priority based multiprocessor system. *IEEE Transactions on Computers*, 41(6):747–754, 1992.

[13] David J. Kuck, Edward S. Davidson, Duncan H. Lawrie, and Ahmed H. Sameh. Parallel supercomputing today and the cedar approach. *Science*, 231(2):967–974, 1986.

[14] Tomas Lang, Mateo Valero, and Ignacio Alegre. Bandwidth of crossbar and multiple-bus connections for multiprocessors. *IEEE Transactions on Computers*, 31(12):1227–1234, 1982.

[15] Imadeldin O. Mahgoub and A. K. Elmagarmid. Performance analysis of a generalized class of m-level hierarchical multiprocessor systems. *IEEE Transactions on Parallel and Distributed Systems*, 3(2):129–138, 1992.

[16] T. N. Mudge and H. B. Al-Sadoun. Memory interference models with variable connection time. *IEEE Transactions on Computers*, 33(11):1033–1038, 1984.

[17] T. N. Mudge and H. B. Al-Sadoun. A semimarkov model for the performance of multiple-bus systems. *IEEE Transactions on Computers*, 34(10):934–942, 1985.

[18] T. N. Mudge, J. P. Hayes, G. D. Buzzard, and D. C. Winsor. Analysis of multiple-bus interconnection networks. *Journal of Parallel and Distributed Computing*, 3(3):328–343, 1986.

[19] T. N. Mudge, J. P. Hayes, and D. C. Winsor. Multiple bus architectures. *IEEE Computer Magagine*, 20(6):42–48, 1987.

[20] S.M. Ross. *Applied Probability Models with OptimizationApplications*. Holden-Day, 1970.

[21] Mary K. Vernon and Mak A. Holliday. Performance analysis of multiprocessor cache consistency protocols using generalized timed petri nets. *ACM Performance Evaluation Review*, pages –, 1986.

[22] Jin-Chin Wang and Michel Dubois. Performance comparison of cache coherence protocols based on the access burst model. *Computer Systems Science and Engineering*, 5(3):147–158, 1990.

[23] 大森 洋一, 城 和貴, 福田 晃, and 荒木 啓二郎. アドレストレースを利用した並列計算機のパラメトリック・シミュレータ. Technical report, 情報処理学会ＨＰＣ, 1993. 発表予定.

[24] 城 和貴. ＡＳＵＲＡの解析モデル. Technical Report 92-ARC-99-17, 情報処理学会ＡＲＣ, 1993.

[25] 城 和貴 and 内藤 潤. セミ・マルコフ過程を用いたＡＳＵＲＡクラスタのモデル化. Technical Report 92-ARC-97-9, 情報処理学会ＡＲＣ, 1992.