計算機アーキテクチャ 97－9
（１９９２．１２．４）

セミ・マルコフ過程を用いたＡＳＵＲＡクラスタのモデル化

城 和貴　　内藤 潤

（株）クボタ
コンピュータ事業推進室

Abstract

分散共有メモリ型マルチプロセッサＡＳＵＲＡの構成要素であるクラスタに着目し、その基本性能について解析する。モデル化の目的は、ＡＳＵＲＡクラスタのキャッシュ・ヒット率及びデータのリード／ライト比が、キャッシュ・コヒーレンス制御を含むリクエストや、それに対する待ち状態に、どのような影響を与えるかを解明し、性能評価を与えることである。このモデル化にあたっては、待ち状態の記述が容易であるセミ・マルコフ過程を利用した。さらに、構築されたモデルを用いて、プロセッサ利用率、バンド幅、リクエスト及びその待ち時間について実際の評価を行なった。

An Analytic Model for the Performance of the ASURA Cluster using a Semi-Markov Processing

Kazuki JOE　　Jun NAITO

Office of Computer Business, KUBOTA Corporation
ASTEM RI
17 Chudoji, Minami-machi, Shimogyo-ku, Kyoto 600 Japan

E-mail: {joe, naito}@kocb.astem.or.jp

Abstract

This paper presents a discrete time model of bus interference for the cache coherence mechanism in the ASURA cluster, which is a basic element of the cluster-based shared memory multiprocessor computer ASURA. It differs from earlier models in its ability to model variable waiting time both for normal requests and cache coherent requests. Actually, the aim of our model is to analyze how the ratio of cache hit and read/write requests can affect the performance from the view point of normal/coherent requests and their waiting states. The model produces as output the processor utilization, bandwidth, requests and their waiting probabilities. This modeling capability is attained by using a Semi-Markov process.

1 はじめに

プロセッサ・クラスタ方式の並列計算機は将来のスーパーコンピューティングへの有力な一方式として、商用 [11]、研究用 [4][2][3] 共に研究開発が推進されている。我々は既に大規模な並列処理計算機への第一歩として、クラスタ・ベースの階層型マルチプロセッサ・システムＡＳＵＲＡを提案し [13]、ＡＳＵＲＡ全体を確率モデルで表した性能評価について報告している [12]。

[12] では、主に各プロセッサのキャッシュ・ヒット率とクラスタ間のネットワークのトラフィックについて報告しているが、ここではコヒーレンス制御を始めとするキャッシュ制御のためのリクエストは考慮していない。一般に密結合型並列計算機の解析モデルを考えた場合、キャッシュ制御に伴うリクエストを予測することは難しく、例えば時間的な状態記述を行えるモデル等を利用しなければ、モデル自体が複雑なものとなってしまう。

そこで本稿では、マルコフ連鎖の各状態で任意の時間滞在出来るセミ・マルコフ過程 [7] に着目し、ＡＳＵＲＡ全体の詳細な解析モデル化の第一歩として、その構成要素であるＡＳＵＲＡクラスタのモデル化を試みた。ＡＳＵＲＡクラスタは、それ自身が密結合型の並列計算機であるため、キャッシュ・コヒーレンス制御のためのリクエストがネットワークに対してどのような影響を与えるかを解析することは、一般の密結合型並列計算機の解析としてはもとより、クラスタ・ベース型の大規模並列計算機の構成要素のデータとしても有用である。

本稿ではＡＳＵＲＡについて簡単に触れた後、セミ・マルコフ過程についての概略、本モデル構築の上での仮定を述べ、状態記述の定義を行い、具体的な計算手順を示した後、簡単な性能評価を行う。

2 ＡＳＵＲＡの概要

ＡＳＵＲＡは（株）クボタと京都大学とが共同で開発している実験システムである [13][9]。図 1 にそのアーキテクチャを示す。ＡＳＵＲＡはメモリの階層性に重点を置いたプロセッサ・クラスタ方式の密結合型マルチプロセッサ・システムである。

プロセッシング・エレメント（ＰＥ）はオンチップの１次キャッシュと４ＭＢの２次キャッシュを持つＲ４０００ＭＣである。これらのローカル・キャッシュはスヌープ方式のキャッシュ・コヒーレンス制御によるイリノイ・プロトコル [1] を一部変更したものに従う。

８個のＰＥと２５６ＭＢのローカル・メモリ、ネットワーク・インタフェース（ＮＩＦ）をバスによって結合し、１つのＡＳＵＲＡクラスタを構成する。このローカル・メモリはクラスタ内のＰＥによって共有される。ＮＩＦは１６ＭＢのグローバル・メモリと３２ＭＢのグローバル・キャッシュから構成される。グローバル・メモリはＡＳＵＲＡシステムの分散共有メモリとして全てのＰＥによって共有される。グローバル・キャッシュはＡＳＵＲＡクラスタ内のＰＥで共有され、フルマップ・ディレクトリ方式のキャッシュ・コヒーレンス制御によるシナプス・プロトコル [1] に従う。また、グローバル・キャッシュは上位のキャッシュとの間にＭＬＩ特性を持つため [13] １次及び２次キャッシュはグローバル・キャッシュをキャッシング出来る。

複数のＡＳＵＲＡクラスタを階層型のネットワークによって結合することによりＡＳＵＲＡシステムが形成される。現段階では、４つのＡＳＵＲＡクラスタをレジスタ・インサーション・リング [8] を一部変更した方式でリング結合しクラスタ・グループを形成する。さらに３２クラスタ・グループをクロスバ結合することにより、最大１０２４ＣＰＵからなるシステムを構成することが出来る。

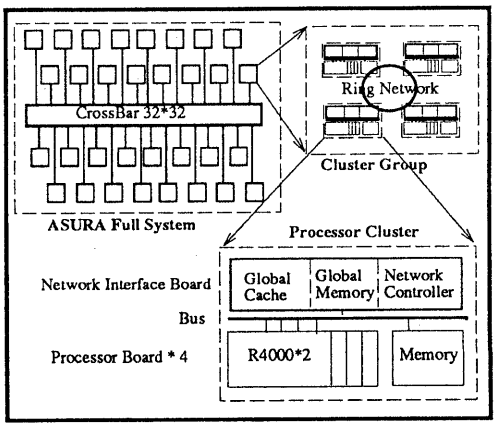

図 1: ＡＳＵＲＡのアーキテクチャ

3 準備

3.1 セミ・マルコフ過程

並列計算機性能評価のモデル化にマルコフ連鎖を利用したものが報告されている [5][6]。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[7]に譲るとして、ここではＳＭＰについての簡単な説明を行う。

ＳＭＰとはＫ個の状態からなる確率過程である。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。

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

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、式1が利用できる。また、ＳＭＰの状態ｉから脱出する確率λ_iは次のようにして求められる。

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

3.2 モデルの仮定

性能評価のモデル化のために、ＡＳＵＲＡクラスタの構造を次のように定義する。Ｎ個のプロセッサと同数のキャッシュ・メモリ、１つのメモリ・モジュールがバス結合され、スヌープ方式のキャッシュ・コヒーレンス制御を行う。コヒーレンス・プロトコルはイリノイ型[1]を一部変更したものに従う。なお、ＡＳＵＲＡクラスタの詳細なアーキテクチャについては文献[10]を参照されたい。

本稿ではＡＳＵＲＡクラスタのキャッシュ・コヒーレンス制御を考慮に入れた性能を評価するために、プロセッサの状態遷移をＳＭＰによって表し、これをエンベデッド・マルコフ連鎖と見なすことにより、セミ・マルコフ過程を定義し、メモリ・リクエストの待ち時間を考慮したモデルを形成する。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. 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。
2. プロセッサの状態は、計算を行なっている状態、バスを握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る。
3. プロセッサ内キャッシュにヒットした状態は計算状態と解釈する。
4. ２次キャッシュにヒットした状態は、バスを使わないリク エストと解 釈し、その間に同時に計算も行なっているとする。
5. 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。ただし、インバリデーションの様に、複数のプロセッサに影響を与えるものに関しては、便宜上、リクエストの確率を引き上げることで近似する。
6. 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。
7. ファイルＩ／Ｏ等によるバスへのリクエストは考慮しない。

[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。

4 モデル化

4.1 状態の定義

ＳＭＰを用いたＡＳＵＲＡクラスタのモデル化を行なうために、プロセッサの状態定義を行なう。プロセッサの状態を大別して

１）計算
２）通常のリード／ライト・リクエスト
３）２）に伴う待ち状態
４）２）に付随するキャッシュ制御リクエスト
５）４）に伴う待ち状態

と考える。ＡＳＵＲＡクラスタのプロセッサは2で示したように２種類のキャッシュを持つが、本モデルにおいてはチップ内キャッシュに対するリクエストは状態１）に含まれるものと考える。従って、各状態においてのリード／ライトは、２次キャッシュもしくはバス（メモリ）に対するリクエストを意味する。以上のことを踏まえて本モデルでは次のような状態定義を行なった。

ＣＯＭ	計算状態、もしくはプロセッサ内キャッシュのアクセス
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
$\overline{\text{ＷｈＩＶ}}$	Ｗｈにより発生するインバリデーション
ＷｈＩＶ	ＷｈＩＶへの待ち状態
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
$\overline{\text{Ｒｃ}}$	Ｒｃへの待ち状態
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
$\overline{\text{Ｒｄ}}$	Ｒｄへの待ち状態

RdWB	Rdにより発生するライトバック
$\overline{\text{RdWB}}$	RdWBへの待ち状態
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
$\overline{\text{Wc}}$	Wcへの待ち状態
WcIV	WcIVにより発生するインバリデーション
$\overline{\text{WcIV}}$	WcIVへの待ち状態
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
$\overline{\text{Wd}}$	Wdへの待ち状態
WdWB	Wdにより発生するライトバック
$\overline{\text{WdWB}}$	WdWBへの待ち状態
Rp	ラインのリプレースにより発生するライトバック
$\overline{\text{Rp}}$	Rpへの待ち状態

このように定められたSMPはASURAクラスタ内の1つのプロセッサの状態記述をしており、全体のモデル化はN個の等価なSMPで表す。このため、本来他のプロセッサに対してライトバックやインバリデーションに伴うライン・リプレースを要求する代わりに、要求したプロセッサ自身がその動作を行なうものとする。これは3.2の仮定に従うものである。

4.2 状態遷移図

図2は定義された状態間の遷移を示す。図2において、hは（2次）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。プロセッサのオペレーションに対するリクエスト率ではないことに注意。）、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（バスに対して）リクエストを出した時に待ち状態に陥る確率（この時、各サイクル毎のリクエスト率Rが必要となる）、xはキャッシュが既に一杯である確率を意味する。図2からも明らかなように、このSMPをEMCと見るとこれはエルゴード的であるので、式1を使って定常分布を求めることが出来る。

状態COMから遷移するのは（プロセッサ内キャッシュ以外への）リクエストが発生した場合のみである。このリクエストが（2次）キャッシュ・ヒットするかしないか、キャッシュ・ミスした場合にそのラインがダーティか否か、バスが空いているかどうか、リクエストはリードかライトか、によってそれぞれRh、Rc、Rd、$\overline{\text{Rc}}$、$\overline{\text{Rd}}$、Wh、Wc、Wd、$\overline{\text{Wc}}$、$\overline{\text{Wd}}$に遷移する。Rh、WhIV、RpからはCOMにのみ遷移する。Whからはライト・ヒットしたラインがクリーンであった場合はインバリデーションを流さなければならないので、バスの状態によってWhIVもしくは$\overline{\text{WhIV}}$に、クリーンでない場合はCOMに遷移する。WhIVからは直接COMに遷移する。なぜならインバリデーションを受けとるキャッシュでの当該ラインはクリーンであるため、バスへのリクエストは起こらないからである。Rcからはキャッシュに空きがあったり、空きがなくてもリプレースされるラインがダーティでなければ、バスへのリクエストは起こらずCOMに遷移するが、キャッシュに空きがなく、かつリプレースされるラインがダーティである場合は、当該ラインのライトバックを行なうためにバスの状態によって、Rpもしくは$\overline{\text{Rp}}$に遷移する。Rdからは当該ラインのライトバックを行なうために、バスの状態によってRdWB、もしくは$\overline{\text{RdWB}}$に遷移する。なお、先に述べたようにここでのライトバックは実際は他のキャッシュでの動作であるが、モデルを簡単にするために、ライトバックは自分のキャッシュで行なうこととする。RdWBからはRcの時同様、COM、もしくはバスの状態によってRpか$\overline{\text{Rp}}$に遷移する。Wcからはインバリデーションを流さなければならないので[2]、バスの状態によってWcIVもしくは$\overline{\text{WcIV}}$に遷移する。WcIVからはRcの時同様、COM、もしくはバスの状態によってRpか$\overline{\text{Rp}}$に遷移する。Wdからは当該ラインのライトバックを行なうために、バスの状態によってWdWBもしくは$\overline{\text{RdWB}}$に遷移する。ここでのライトバックもRdWBの時同様、自分のキャッシュが行なうことを仮定している。WdWBからはRcの時同様、COM、もしくはバスの状態によってRpか$\overline{\text{Rp}}$に遷移する。待ち状態$\overline{\text{WhIV}}$、$\overline{\text{Rc}}$、$\overline{\text{Rd}}$、$\overline{\text{RdWB}}$、$\overline{\text{Wc}}$、$\overline{\text{WcIV}}$、$\overline{\text{Wd}}$、$\overline{\text{WdWB}}$ $\overline{\text{Rp}}$はバスの状態に応じてそれぞれWhIV、Rc、Rd、RdWB、Wc、WcIV、Wd、WdWB、Rpに遷移する。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3のように表される。

4.3 滞在時間

Rh	Wh	WhIV	Rc	Rd	RdWB
10	10	33	121	163	33

Wc	WcIV	Wd	WdWB	Rp
121	33	163	33	33

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

[2]イリノイ型と違ってクリーン・エクスクルーシブの状態がないのでダーティでないなら必ずインバリデーションを流すことになる。

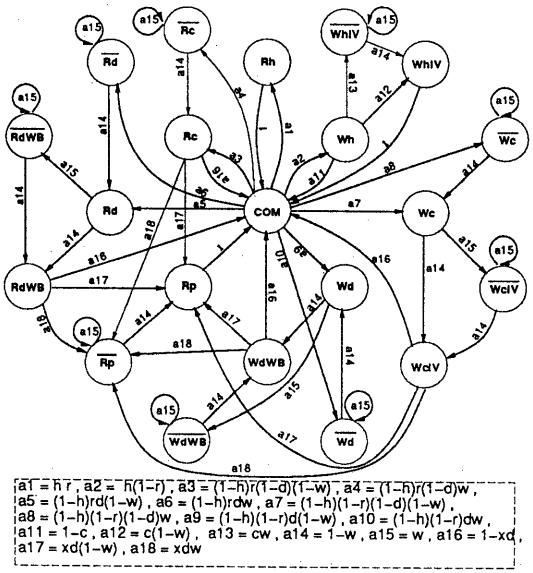

図 2: ＡＳＵＲＡクラスタにおけるプロセッサの状態遷移

$$
\begin{pmatrix}
0 & a1 & a2 & 0 & 0 & a3 & a4 & a5 & a6 & 0 & 0 & a7 & a8 & 0 & 0 & a9 & a10 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a11 & 0 & 0 & a12 & a13 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a17 & a18 \\
0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a14 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a17 & a18 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
a16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a17 & a18 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 \\
a16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a17 & a18 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 \\
a16 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a17 & a18 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & a14 & a15 &
\end{pmatrix}
$$

図 3: ＥＭＣの遷移確率行列

は、各状態の平均滞在時間が必要である。まずＡＳＵＲＡクラスタの仕様から定まるものとしては、表 1がある。ただし、各値はプロセッサのサイクル数を表す。状態ＣＯＭでの滞在時間は、先に述べたように通常の計算時間に加えて、プロセッサ内キャッシュに対するアクセス時間も含めたものとするので、２０サイクル程度が妥当かと思われる。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。従って、各待ち状態の滞在時間（Ｗｔとする）は適当な初期値を与え、以後バスへのアクセスを行なう各状態の確率をもとに繰り返し計算することによって収束させることとする[3]。

4.4 ＳＭＰの定常分布

ＳＭＰの定常分布は、ＥＭＣの定常分布 $\{\pi_i\}$ と前節で定義した滞在時間 $\{\eta_i\}$ を使って式 1から求められる。$\{\pi_i\}$ はＥＭＣの遷移確率行列を Pとすると、

$$\lim_{m\to\infty} P^m \tag{3}$$

の各列（全て等しい）である。

図 3からこれを計算すると、ＥＭＣの定常分布 $\{\pi_i\}$ は

$$
\frac{\begin{pmatrix}
1-w \\
hr(1-w) \\
h(1-r)(1-w) \\
ch(1-r)(1-w) \\
ch(1-r)w \\
(1-d)(1-h)r(1-w) \\
(1-d)(1-h)rw \\
d(1-h)r(1-w) \\
d(1-h)rw \\
d(1-h)r(1-w) \\
d(1-h)rw \\
(1-d)(1-h)(1-r)(1-w) \\
(1-d)(1-h)(1-r)w \\
(1-d)(1-h)(1-r)(1-w) \\
(1-d)(1-h)(1-r)w \\
d(1-h)(1-r)(1-w) \\
d(1-h)(1-r)w \\
d(1-h)(1-r)(1-w) \\
d(1-h)(1-r)w \\
d(1-h)(1-w)x \\
d(1-h)wx
\end{pmatrix}}{3-h+ch(1-r)-w(1+h)+(1-h)(dx-r(1-d))} \tag{4}
$$

で求められる。また、各状態からの脱出確率は式 2で求められる。

4.5 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、（セカンダリ）キャッシュのヒット率ｈ、データ・リクエストに対するリードの割合ｒ、全てのデータに対するリード／ライト・データの割合 RW_ratio[4]を考えることとする。

これらのパラメータとＡＳＵＲＡクラスタのプロセッサ数Ｎで、4.2や 4.3で使用した変数ｄ、ｃ、ｗ、ｘ、Ｒ、Ｗｔを表す。

[3] ＡＳＵＲＡでのリクエストは通常の場合でもコヒーレンス制御の場合であっても同等に見なされるので、各待ち状態の平均滞在時間は同じと仮定できる。

[4] データをリード・オンリー、ライト・オンリー、リード／ライトと分けた場合、キャッシュ・コヒーレンス制御に関係するのは同期変数等のリード／ライト・データである。

あるプロセッサがあるラインをダーティで持っている確率は $h(1-r)$ だから、自分以外のあるプロセッサがダーティで持ちそれ以外のプロセッサがダーティで持っていない確率 d は次式で表される。

$$d = (N-1)h(1-r)(1-h(1-r))^{N-1} \quad (5)$$

同様に、自分以外の少なくとも一つのプロセッサがあるラインをリード・ヒットでキャッシュに持ちそのラインがリードかつライトの属性を持つ確率 c は次式で表される。

$$c = 1-(1-hrRW\_ratio)^{N-1} \quad (6)$$

キャッシュ・インしたときに既にキャッシュに空きがない状態は次のようにして求められる。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、

$$x = (1-inv)^{N-1} \quad (7)$$

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出来る。) あるプロセッサがインバリデーションを流す確率は、自分がクリーンのラインをキャッシュに持ち、そのラインにライト・アクセスを行なうか、もしくは、自分の持っていないダーティではないラインに対してライトアクセスを行なう場合である。よって、

$$inv = hc(1-r)R + (1-h)(1-d)(1-r)R \quad (8)$$

によって inv は求まる。

バス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、そもそもバスが空いてなかった場合と、バス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、バスが空いていない確率は次のようにして求める。バスへのリクエストを出す状態の集合を S とする。

$$S = \{WcIV, Rc, Rd, RdWB, Wc, WcIV, Wd, WdWB, Rp\} \quad (9)$$

あるプロセッサの状態が $i \in S$ 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、バスが空いていない確率を BUSY とすると、

$$BUSY = (N-1)\sum_{i \in S}(P_i - \lambda_i) \quad (10)$$

となる。ただし、P_i は SMP の状態 i での定常分布である。次に、他のプロセッサからのリクエストとの競合に負ける確率は、

$$1 - \frac{1-(1-R)^N}{NR} \quad (11)$$

で求まる。なぜなら、あるサイクルで少なくとも1つのプロセッサがリクエストを出す確率は $1-(1-R)^N$、その時のリクエスト数は NR だからである。よって、待ち状態に陥る確率 w は

$$w = BUSY + (1-BUSY)(1 - \frac{1-(1-R)^N}{NR}) \quad (12)$$

となる。

サイクル毎のプロセッサのリクエスト率 R は、次のようにして求める。まず、状態が $i \in S$ の時、η_i の時間の間に一回だけリクエストが起きる。さらに、状態 S の待ち状態の集合 $\overline{S}$ を

$$\overline{S} = \{\overline{WcIV}, \overline{Rc}, \overline{Rd}, \overline{RdWB}, \overline{Wc}, \overline{WcIV}, \overline{Wd}, \overline{WdWB}, \overline{Rp}\} \quad (13)$$

とすると、状態が $i \in \overline{S}$ の時はサイクル毎にリクエストを発行していると考える。(勿論、実際のインプリメンテーションとは異なる。) よって、サイクル毎のプロセッサのリクエスト率は、

$$R = \sum_{i \in S}\frac{P_i}{\eta_i} + \sum_{j \in \overline{S}} P_j \quad (14)$$

である。

Wt は待ち状態にいる時、バスが次に空くまでの平均時間であるので、

$$Wt = \sum_{i \in S} P_i \eta_i \quad (15)$$

によって求まる。

なお、x、w、R、Wt はモデルに与えるパラメータだけでは決まらず、後に述べるように、繰り返し計算することにより収束させて求める。

4.6 モデルの計算手順

このようにして定義したモデルを実際に計算する手順は以下の通りである。

1. パラメータとして、キャッシュ・ヒット率 h、リード・リクエストの割合 r、リード/ライト・データの割合 RW_ratio[5] を定める。
2. サイクル毎のリクエスト率 R と平均待ち時間 Wt に適当な初期値 (例えば、R に $1-\frac{1}{N}$、Wt には $\eta_i (i \in S)$ の平均) を与える。

[5] 本稿ではこの割合を5%と見積もったが、本来は並列化プログラムに左右される値である。

3. EMCの定常分布 $\{\pi_i\}$ を求める（式 4）
4. 平均滞在時間 $\{\eta_i\}$ を求める（表 1と式 15）
5. 各状態からの脱出確率 $\{\lambda_i\}$ を求める（式 2）
6. R（式 14）、x（式 7）、w（式 12）、Wt（式 15）を求める
7. 前回のRと新しく求めたRの差が適当な値よりも大きければ3に戻る

5 評価

5.1 評価対象

SMPを用いたASURAクラスタのモデルに対する評価対象として、バンド幅、プロセッサ利用率、通常及びコヒーレント制御のリクエスト時間の割合、それらの待ち時間の割合等を考える。

バンド幅（$Bandwidth$）の定義を、プロセッサのサイクルあたり、（セカンダリ）キャッシュ及びメモリに対するアクセプトされたリクエスト数とすると、

$$N \sum_{i \in \{Rh,Wh,Rc,Rd,Wc,Wd\}} \frac{P_i}{\eta_i} \qquad (16)$$

プロセッサ利用率（$Processor\_Utilization$）は、キャッシュにアクセス中にプロセッサは計算を行なうことが出来るという仮定3.2のもとでは、

$$\sum_{i \in \{COM,Rh,Wh\}} P_i \qquad (17)$$

全体に対する通常のメモリ・リクエストに要する時間の割合（$Normal\_IO\_request$）は、

$$\sum_{i \in \{Rc,Rd,Wc,Wd\}} P_i \qquad (18)$$

その待ち時間の割合（$Normal\_IO\_wait$）は、

$$\sum_{i \in \{\overline{Rc},\overline{Rd},\overline{Wc},\overline{Wd}\}} P_i \qquad (19)$$

同じく、コヒーレンス制御のリクエスト時間の割合（$Coherent\_IO\_request$）は、

$$\sum_{i \in \{WhIV,RdWB,WcIV,WdWB,Rp\}} P_i \qquad (20)$$

その待ち時間の割合（$Coherent\_IO\_wait$）は、

$$\sum_{i \in \{\overline{WhIV},\overline{RdWB},\overline{WcIV},\overline{WdWB},\overline{Rp}\}} P_i \qquad (21)$$

で表される。

5.2 評価結果

ASURAクラスタの評価に対するパラメータとして、

1. $r = 0.8, 0.7 < h < 1.0, RW\_ratio = 0.1$
2. $h = 0.8, 0.7 < r < 1.0, RW\_ratio = 0.1$
3. $h = 0.97, 0.7 < r < 1.0, RW\_ratio = 0.1$

としたものの結果をグラフ 4、5、6に示す。

図 4では、キャッシュ・ヒット率と性能の関係を示している。キャッシュ・ヒット率が９５％以下の場合はプロセッサ利用率が極端に悪く、９７％程度まではコヒーレント制御リクエストに対する待ち時間が多いので、実際に高性能を発揮出来るのは、ヒット率９７％以上である。

図 5では、リード・リクエスト率と性能の関係を示している。まず、全体の傾向を知るためキャッシュ・ヒット率を低目（８０％）に設定して評価した結果、リクエストの待ち状態が８割前後にも達しており、リード・リクエスト率の影響を読みとりにくい。

図 6では、望ましいキャッシュ・ヒット率におけるリード・リクエスト率と性能の関係を示している。プロセッサ利用率はリード・リクエスト率が９０％を越える付近から急速に向上しているのが分かる。さらに、コヒーレント制御に対する待ち時間は急減少している。

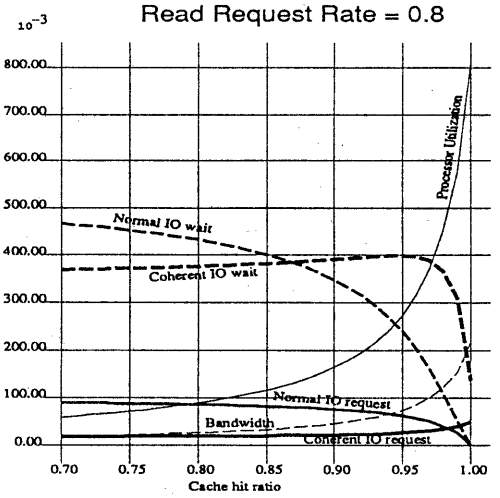

図 4: $r = 0.8, 0.7 < h < 1.0$ での性能

6 結論

分散共有メモリ型並列計算機ASURAの構成要素であるクラスタのキャッシュ・コヒーレント制

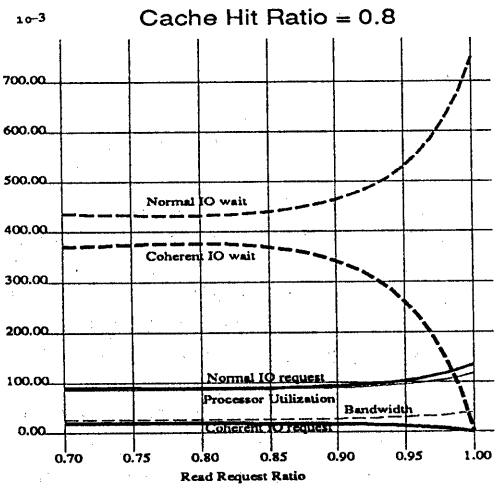

図 5: $h = 0.8, 0.7 < r < 1.0$ での性能

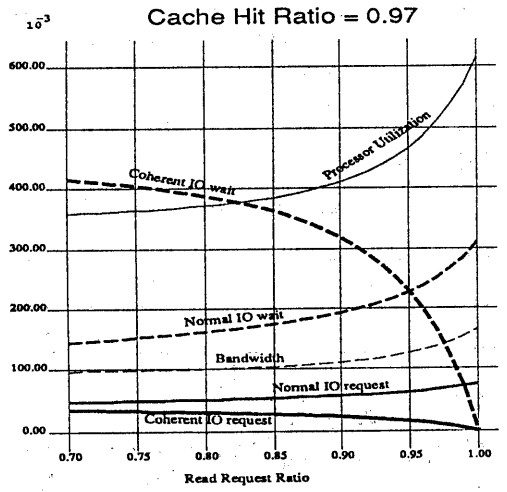

図 6: $h = 0.97, 0.7 < r < 1.0$ での性能

御を考慮した基本性能を、セミ・マルコフ過程を用いることにより解析モデルで表し、キャッシュのヒット率及びリード／ライト命令の比率がクラスタ全体の性能にどのように影響を与えるかを示した。その結果、全体の性能はキャッシュのヒット率は勿論のこと、リード／ライト命令の比率にも左右されることが確認された。

本モデルはＡＳＵＲＡ全体の解析モデル構築のための基礎的な部分であり、キャッシュ・コヒーレント制御を考慮に入れた大規模なクラスター・ベースの並列計算機のモデル化を行なうことが今後の研究課題である。

謝辞

日頃ご討論頂く京都大学工学部富田教授並びに同研究室の諸氏に感謝致します。また、研究の機会を与えて頂いた（株）クボタ山口部長並びに名古屋大学工学部阿草教授に感謝いたします。

参考文献

[1] 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.

[2] Daniel Lenoski et al. The directory-based cache coherence protocol for the dash multiprocessor. In *Proceedings of the International Symposium on Computer Architecture*, pages 148–159, 1990.

[3] David R. Cheriton et al. Paradigm: A highly scalable shared-memory multicomputer architecture. *IEEE Computer*, pages 33–46, 1991.

[4] David J. Kuck, Edward S. Davidson, and Duncan H. Lawrie Ahmed H. Sameh. Parallel supercomputing today and the cedar approach. 電子情報通信学会論文誌, J71-D(8):1361–1374, 1988.

[5] Marco Ajmone Marsan and Mario Gerla. Markov models for multiple bus multiprocessor systems. *IEEE Transactions on Computers*, 31(3):239–248, 1982.

[6] Ibrahim H. Onyuksel and Keki B. Irani. A markovian queueing network model for performance evaluation of bus-deficient multiprocessor systems. In *Proceedings of the 1983 International Conference on Parallel Processing*, pages 437–439, 1983.

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

[8] Andrew S. Tanenbaum. *COMPUTER NETWORKS*. Prentice Hall, 1981.

[9] 齋藤 秀樹, 森 眞一郎, 城 和貴, David Fraser, 田中 高士, and 富田 眞治. イベント対応型キャッシュ・コヒーレンス制御方式とそのバリア同期への応用. Technical Report 92-ARC-95-2, 情報処理学会計算機アーキテクチャ研究会, 1992.

[10] 内藤 潤, 城 和貴, 松野 宏昭, and 新田 博之. ＡＳＵＲＡクラスタの性能評価. Technical Report 92-ARC-97-10, 情報処理学会計算機アーキテクチャ研究会, 1992.

[11] 日本アライアントコンピュータ. *The CAMPUS/800 Supercomputer*. 技術資料, 1991.

[12] 城 和貴, 柳原 守, David Fraser, 田中 高士, 新田 博之, 森 眞一郎, 齋藤 秀樹, and 富田 眞治. 分散共有メモリ型マルチプロセッサ「ＡＳＵＲＡ」の階層性とその評価. Technical Report 92-ARC-95-1, 情報処理学会, 1992.

[13] 森 眞一郎, 齋藤 秀樹, 五島 正裕, 富田 眞治, 田中 高士, David Fraser, 城 和貴, and 新田 博之. 分散共有メモリ型マルチプロセッサ「阿修羅」の概要. Technical Report 92-ARC-94-6, 情報処理学会, 1992.