バス結合型並列計算機の交信用メモリの性能評価†

鳥居 淳†† 竹本 卓††
天野英晴†† 小椋 里††

共有メモリとローカルメモリの双方を持つバス結合型マルチプロセッサの交信に関する性能を評価する．ブロードキャストメモリ（BCM），マルチキャストメモリまたはライトブロードキャスト型スヌープキャッシュ（MCM），ライトスルースヌープキャッシュ（WTC）の3種類について確率過程モデルを構築して評価し，これと並列計算機テストベッドATTEMPT-0を用いた実測値を組み合わせて検討した．その結果，解析モデルでは，プロセッサ数が多い場合アイドル時間の増加によりアクセス頻度が低下するという現象をモデル化できないため，実測値より混雑が激しくなるという結果が得られることがわかった．WTCは小規模なシステムでは有効に働くが，バス占有率が大きくプロセッサ数が多くなると対処できない．BCMとMCMは，ごく限られた条件の下ではMCMが有利であるが，両者共にプロセッサ数が多い場合でもWTCに比べて高い性能を持つ．コストの点を考慮すると，アドレス空間が小さい場合変換テーブルが不要で制御が簡単なBCMが，大きい場合はコピーの量が少ないMCMが有利であるといえる．

1. はじめに

複数のプロセッサがバスを介してメモリを共有する構成方法は小規模な並列計算機の実現に対して最も簡単かつ有効な実現法であり，既に多くの商用機，実験機が登場している．この構成は以下の2つに分類することができる．

主記憶共有型：各プロセッサはローカルメモリを持たず，スヌープキャッシュを介して主記憶を共有する．

交信メモリ型：プロセッサはプログラムコードやローカル変数を格納するローカルメモリを持ち，プロセッサ間のデータ交換は交信用共有メモリを用いる．

主記憶共有型では全プロセッサが単一アドレス空間を共有するため，プロセスの動的割り付けが容易で，全体としてメモリの無駄が少なく，シングルプロセッサ用のOSを拡張して用いることができる．このため，最近の汎用目的の商用機のほとんどがこの構成をとっている．これに対して交信メモリ型では，ローカルデータをローカルメモリに置くことにより，バス上のデータ交信量が抑えられるため，多数のプロセッサを接続できるという利点がある．

従来，主記憶共有型に対しては，キャッシュの一貫性を保持する制御プロトコルを中心に，解析モデル，シミュレーション，実際の測定に基づく様々な評価が行われている[1)-3),7)]．これに対し交信メモリ型に対する評価の試みは数少ない[9),13)]．われわれは，交信メモリとして各種のマルチリードメモリを用いたシステムについて検討すると共に[14)]，評価モデルについても提案を行ってきた．本論文では各種の交信用メモリを対象として，文献9）で提案した評価モデルの拡張を行う．次に，並列計算機テストベッドATTEMPT-0[4)]上に実際的なアプリケーションを実装することにより，評価モデルと実測値の差に関して検討する．最後にこれらの評価モデルと実測値に基づき，各種の交信用メモリの性能を検討する．

2. 交信用メモリとその解析モデル

2.1 交信用メモリの構成

交信メモリ型では，共有メモリは純粋にプロセッサ間のデータ交換のみに用いられる．しかし，共有メモリモジュールを単純にバスに接続しただけではメモリアクセスの競合による性能低下が予想されるため，次に示す交信データのアクセスについての性質を利用して，競合を低減することが考えられる．

- あるプロセッサが書き込んだデータを，他のプロセッサが複数回読み出すことが多い（プロセッサの読み出し局所性）．
- あるプロセッサが書き込んだデータを，複数のプロセッサが読み出すことが多い（データのマルチキャスト性）．

これらの性質を利用して競合低減を図った交信メモリを以下に示す．

ブロードキャストメモリ[10)-12)]：同一アドレス空間に

† A Performance Evaluation of Communication Memory Systems for Parallel Machines with a Shared Bus by SUNAO TORII, TAKASHI TAKEMOTO, HIDEHARU AMANO and SATOSHI OGURA (Faculty of Science and Technology, Keio University).

†† 慶應義塾大学理工学部

配置される交信用メモリを全プロセッサがそれぞれ持ち，あるプロセッサが書き込みを行うと，そのデータはバスを介してすべてのプロセッサの交信メモリに同時に書き込まれる（図1）．各プロセッサは自分の交信用メモリの内容を独立に読み出せるため，読み出し競合がなく，読み出し局所性，マルチキャスト性を共に利用できる．

マルチキャストメモリ[14]：ブロードキャストメモリでは，あるプロセッサが書き込みを行っている間はどのプロセッサも交信用メモリから読み出しを行うことはできない．すなわち受け手のプロセッサは，自分が必要としないデータの書き込み時にも読み出しを行うことができない．マルチキャストメモリではデータのブロックごとにタグを設け，受け手のプロセッサにとって必要なデータだけを選択して交信メモリに書き込む．このため，不要なデータの書き込みにより読み出しがブロックされることはない（図2）．

スヌープキャッシュ：スヌープキャッシュの一貫性の維持は，ある一面ではプロセッサの交信と見ることができる．スヌープキャッシュは書き込み放送型と書き込み無効化型に分類することができる．このうち，書き込み放送型では必要なプロセッサのキャッシュにのみ書き込みデータが放送されるため，共有状態を保存したまま一貫性を維持できる．このため，データのマルチキャスト性，読み出しの局所性を共に利用できる．これに対し，書き込み無効化型のキャッシュはデータの共有状態を保存することを指向していないため，読み出しの局所性は利用できるがデータのマルチキャスト性は利用できない．

書き込み放送型のスヌープキャッシュを交信メモリとして用いた場合の性能は，キャッシュが十分大き

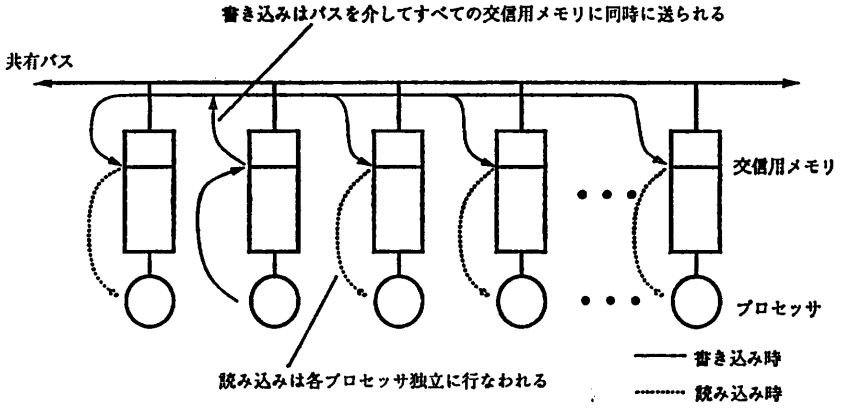


図1 ブロードキャストメモリ
Fig. 1 Broadcast memory.

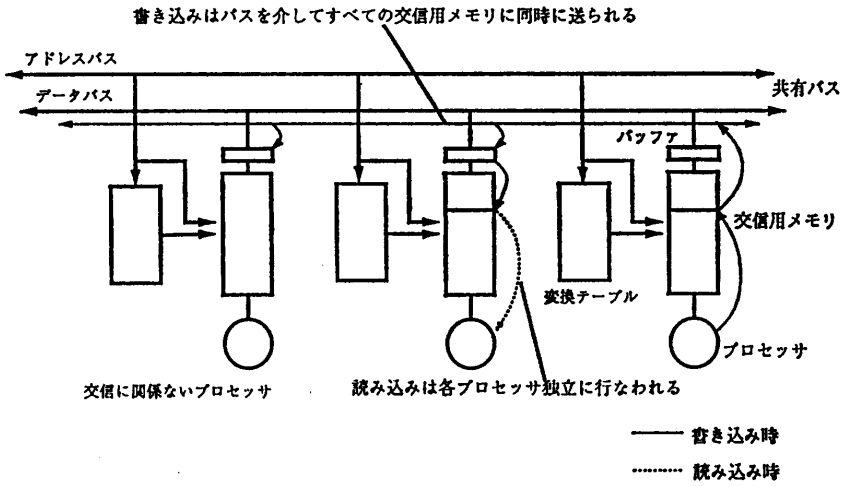


図2 マルチキャストメモリ
Fig. 2 Multicast memory.

く，主記憶へのブロックの追い出しが少なければ，マルチキャストメモリとほぼ等しくなる．そこで，本論文では両者を代表させて，マルチキャストメモリのみを評価する．書き込み無効化型キャッシュはライトバック型とライトスルー型に分類できる．両者の相違はページのプライベート状態の有無にあるが，交信データのみを置く場合，プライベート状態は性能向上にはとんど寄与しない．このため，本論文では構造が簡単な利点を持つライトスルー型を書き込み無効化型の代表とする．すなわち，本論文ではブロードキャストメモリ (BCM)，マルチキャストメモリ (MCM)，ライトスルー型スヌープキャッシュ (WTC) の3種類を評価の対象として考える．

2.2 解析モデル

主記憶共有型におけるスヌープキャッシュの解析モデルは，シングルプロセッサのキャッシュ解析に用いる確率モデルを発展させたものが多い[7),8)]．しかし，主記憶共有型では，プログラムコード，ローカルデータ，共有データのすべてがキャッシュを介してアクセスされるので，モデルが複雑になりパラメタ数が増加し，設定が困難である．

これに対して，交信メモリ型では共有メモリ上にはプライベートデータは置かないので，共有データに対するアクセスのみをモデル化すれば良い．このため，モデルは簡単になり，アプリケーションが定まればパラメタの設定は容易である．離散的確率過程モデルを構築し，アプリケーションごとにパラメタを決定すれば，かなりの精度で性能を予測できる．われわれは過去に BCM について確率過程モデルを提案したが[9)]，このモデルは書き込みと読み出しの時間が等しくなければならないなど制約が多かった．そこで本論文では各種交信用メモリに対応できるように改良，拡張を行い，これを用いて評価を行う．

まず BCM のモデルを示す．各プロセッサは図3に示す状態遷移を行う．ここで BCM の読み出しアクセスタイムを単位時間として，離散化を行っている．すなわち，各プロセッサは計算状態に T_c 時間滞在してから，バスの状態により，確率 P_w で交信メモリに対する書き込み状態へ，確率 P_c で計算状態のまま，$1-P_w-P_c$ で読み出し状態へ遷移する．バス上で他のプロセッサが書き込みを行っていた場合（すなわち他のプロセッサがバスの使用権を取っていた場合：BUS=0)，待ち状態に遷移する．次の単位時間でバスが空く場合，読み出し状態への遷移が可能となる．また，書き込み状態へ遷移するためには次の単位時間でバスを獲得 (BUS=1) する必要がある．

システム中には複数のプロセッサが存在するので，この状態遷移モデルをプロセッサ数分用意し，バス獲得の競合解消機構を設けてシステム全体をモデル化する．

MCM のモデルは BCM のモデルを基にしており，バスが占有される場合でも，ある確率 ($P_{nonblock}$) で読み出しが可能になる点のみが異なる．これに対し WTC のモデルは図4に示すように，ヒット率 (P_{hit}) によってページのスワップインが生じる．ここではキャッシュは十分大きいと仮定しており，ミスヒットはすべて他のプロセッサによる無効化により生じると考える．

これらのモデルのパラメタは，アプリケーションプログラムの交信メモリへのアクセス間隔の規則性により決定することができる．アクセス間隔の規則性は

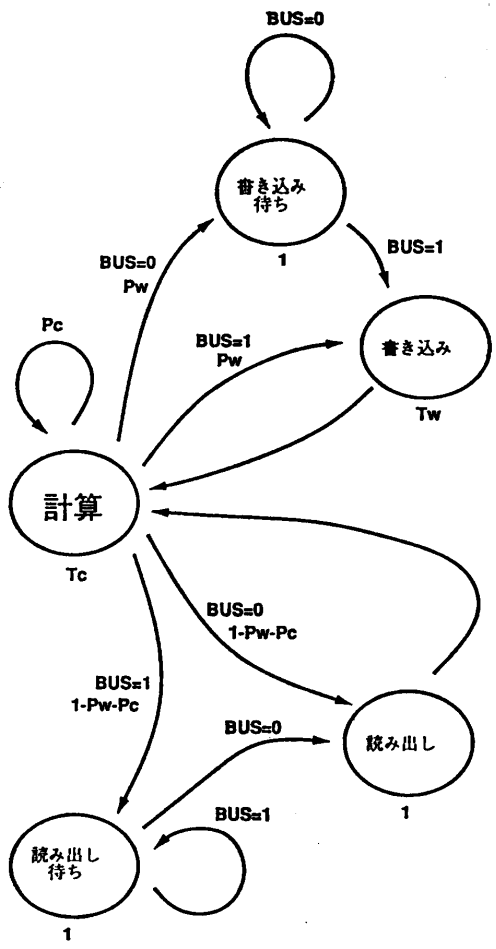


図3 ブロードキャストメモリ (BCM) の解析モデル
Fig. 3 A probability process model for BCM.

P_c と t_c の値によって定まる．行列計算のように，各プロセッサが共有データ（行列やベクトルの要素）に対して一定時間間隔でアクセスする場合は $P_c=0$ とし，t_c をその要素を用いて行う演算時間に設定する．逆に各プロセッサが不規則な時間間隔でアクセスする場合は，まずシングルプロセッサのトレースからアクセス間隔の平均時間を測定する．次にアクセスの発生がポアソン分布と仮定し，t_c を1とした時のアクセス間隔が測定した平均時間に等しくなるように P_c を設定する．

2.3 理論解析法

前節で提案したモデルは，各プロセッサごとに状態を持ち，プロセッサは各状態に一定時間留まるとしたため，数式的に解くことは困難であり，簡単な解析プログラムを作る必要がある．しかし BCM と MCM に関しては $t_c=1$ とした場合に，理論解析可能である．BCM は MCM の特殊な場合（$P_{nonblock}=0$）であるとモデル化されているため，ここでは MCM についての解析方法を述べる．

理論解析を行う場合，なるべく状態数を減らしたほうが解析が容易になる．そこで，書き込み待ち状態と書き込み状態は合わせて1つの状態Wとし，同様に読み出し待ち状態と読み出し状態も合わせて状態Rとする．また，計算状態を状態Cとする．

以下マルコフチェーンモデルによる解析法および期待値モデルによる解析法について述べる．

2.3.1 マルコフチェーンモデルによる解析

マルコフ解析では，個々のプロセッサではなくシステム全体の状態を考える．システムの状態はモデル中のプロセッサの分布状況で表現する．すなわち，C，W，R の各状態にあるプロセッサの数の組 (c, w, r) に着目して，全体の状態を表す．ここで，$c+w+r=n$（n はプロセッサ数）の関係が成り立つことから，w, c の2変数で状態を表現できる．書き込みにかかる時間 t_w は1単位時間以上であるので書き込みを行っているプロセッサが，現在書き込みに何単位時間消費したかを示す変数 s が必要である．すなわち，システムの状態は3つの組 (c, w, s) で表せる．

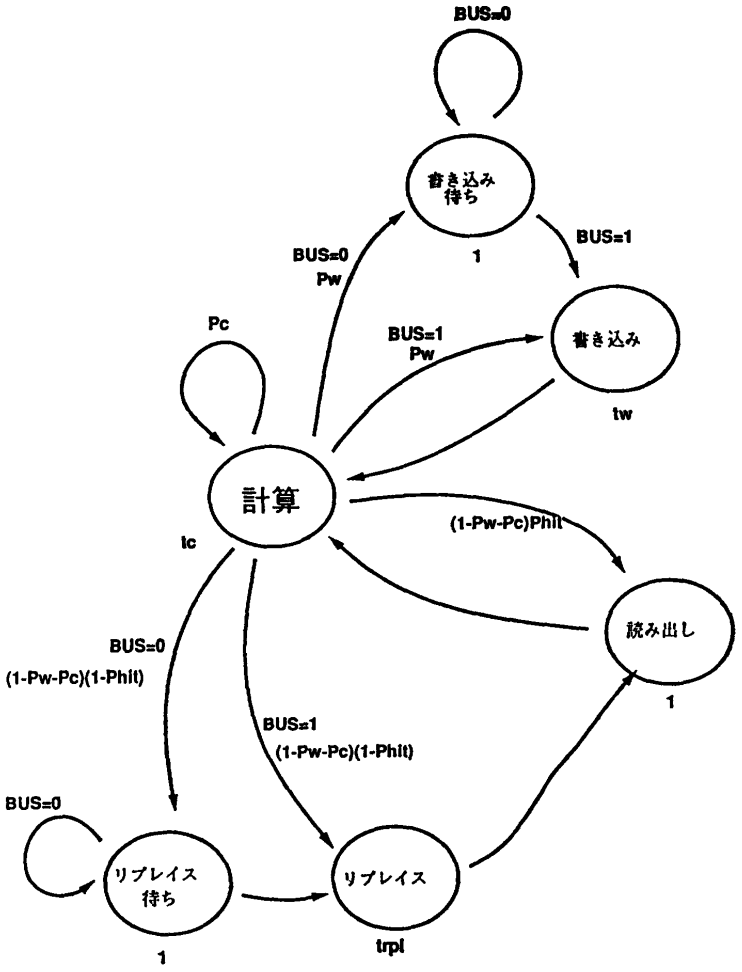


図4 ライトスルーキャッシュ（WTC）の解析モデル
Fig. 4 A probability process model for WTC.

次に状態遷移行列 S を作るために2つの状態間の推移確率を考える．状態 $I(c_i, w_i, s_i)$ が状態 $J(c_j, w_j, s_j)$ に推移する確率 P_{IJ} は次のように求めることができる．状態Iから状態Jに移る時，各状態へ推移するプロセッサ数を次の変数で表す．

状態 C から状態 W に推移…x 個
状態 C から状態 C に推移…y 個
状態 C から状態 R に推移…z 個
状態 R から状態 C に推移…r_e 個
状態 W から状態 C に推移…w_e 個

（a） $w_i=0$ の場合

書き込みを行っているプロセッサが存在しないため，今までブロックされていたプロセッサはいっせいに読み出しを行う．この場合 r_e, w_e は一意に決まるため，x, y, z も一意に決まる（付録(3)式）．したが

って推移確率 P_{IJ} は簡単に求まる（付録(4)式）．

(b)　$w_i \neq 0$ の場合

読み出し状態のプロセッサは書き込みにより $1-P_{nonblock}$ の確率でブロックされる．書き込み中のプロセッサは s_i が t_w に等しくなったら計算状態に戻ることができる．したがってこの時だけ $w_e=1$ となり，それ以外の場合は $w_e=0$ である（付録(5)式）．

$P_{nonblock}=0$ の場合，すなわち BCM の場合は一意に $r_e=0$ に決まる．ところが MCM では r_e は 0 から $(n-c_i-w_i)$ までのすべての値をとる可能性がある．そこで，r_e のとりうるすべての値についての期待値を計算することにより P_{IJ} を求める(付録(6)式)．

システムの各状態の存在確率はベクトルの形で表される．これは状態確率ベクトルと呼ばれ，ある時点においてどの状態がどのくらいの確率で存在するかを表したものである．この状態確率ベクトル s_k に推移確率行列 S を掛けると，1単位時間後の状態確率ベクトル s_{k+1} が求まる．推移確率行列 S は P_{IJ} を要素とす

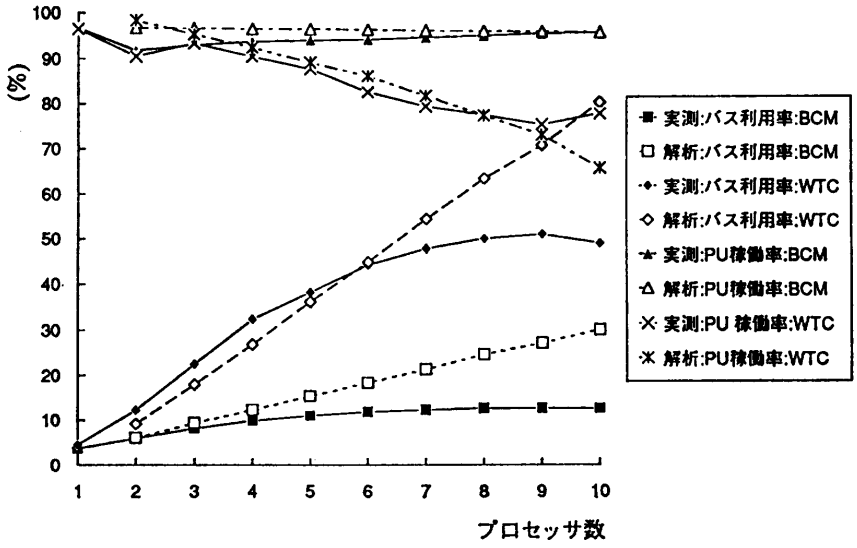


図 5　解析モデルと実測値との比較 (GAUSS)

Fig. 5　The number of processors VS. processor/bus utilization ratio (GAUSS): results from probability process model and measurement are compared.

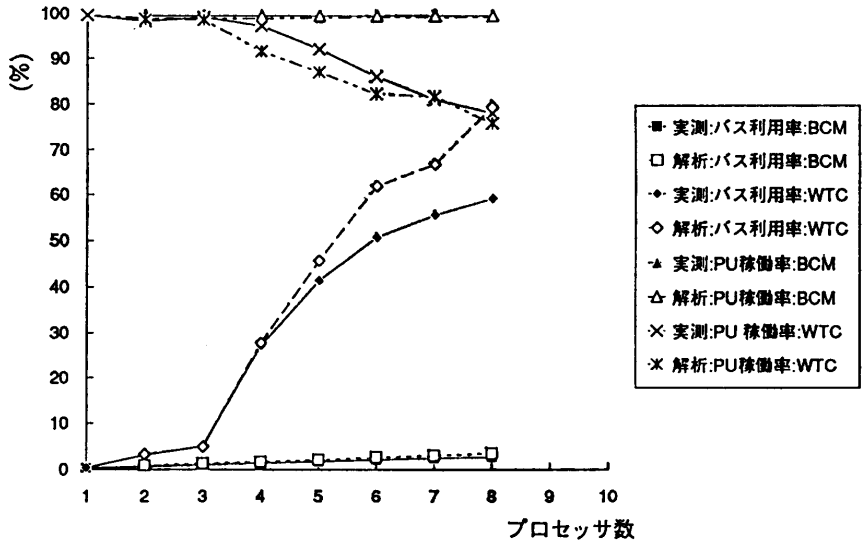


図 6　解析モデルと実測値との比較 (CG)

Fig. 6　The number of processors VS. processor/bus utilization ratio (CG): results from probability process model and measurement are compared.

る正方行列である.

$s_1 = S \cdot s_0$ …初期状態

$s_{k+1} = S \cdot s_k$ …反復式 (1)

(1)式により，定常状態における各状態の存在確率を表す状態確率ベクトル s_e を求めることができる.

2.3.2 期待値モデルによる解析

マルコフチェーンモデルではシステムのとりうるすべての状態をモデル化するのに対し，期待値モデルではシステムの平衡状態のみをモデル化している.

いま，ステップ k において，C, W, R, それぞれの状態にあるプロセッサの数の期待値を $c^{(k)}$, $w^{(k)}$, $r^{(k)}$ とおくと，次のステップで，推移確率から得られる期待値に相当する数のプロセッサがそれぞれの状態に分岐すると考えられる．ここで，解析を容易にするため，以下のような仮定をする.

- 状態Wにあるプロセッサ数の期待値は1より大きくならない.
- 書き込みにかかる時間を t_w 単位時間とすると，t_w 回に1回の割合で書き込みが実行できる．したがって，状態Wから状態Cに推移するプロセッサ数の期待値を $\frac{1}{t_w} \cdot w^{(k)}$ とする.
- 書き込みによる読み出しのブロックを考慮するため，状態Wにあるプロセッサ数の期待値を，読み出しを行うプロセッサがブロックされる確率とする．すなわちブロックされるプロセッサ数の期待値を $w^{(k)} \cdot r^{(k)}$ とする.

以上から，次のステップにおいて各々の状態にあるプロセッサ数の期待値は次のように計算される.

表 1 シンクロナイザのアクセス時間
Table 1 Access time of synchronizer.

機能	時間	バス操作の有無
Read(X)	200 ns	無
Write(X. data)	650 ns	有
Fetch&Dec(X)	750 ns	有

表 2 キャッシュのアクセス時間
Table 2 Access time of cache.

アクセスの種類	時間	転送バイト数
Read hit	150 ns	4 byte
Read miss	18, 200 ns	64 byte
Write	750 ns	4 byte

表 3(a) 解析モデルのパラメタ (GAUSS)
Table 3(a) Parameter of probability process model (GAUSS).

	MCM/BCM	WCT
T_c	42	56
T_w	3	5
P_w	0.34	0.34

表 3(b) 解析モデルのパラメタ (CG)
Table 3(b) Parameter of probability process model (CG).

	MCM/BCM	WCT
T_c	126	181
T_w	3	5
P_w	0.14	0.14

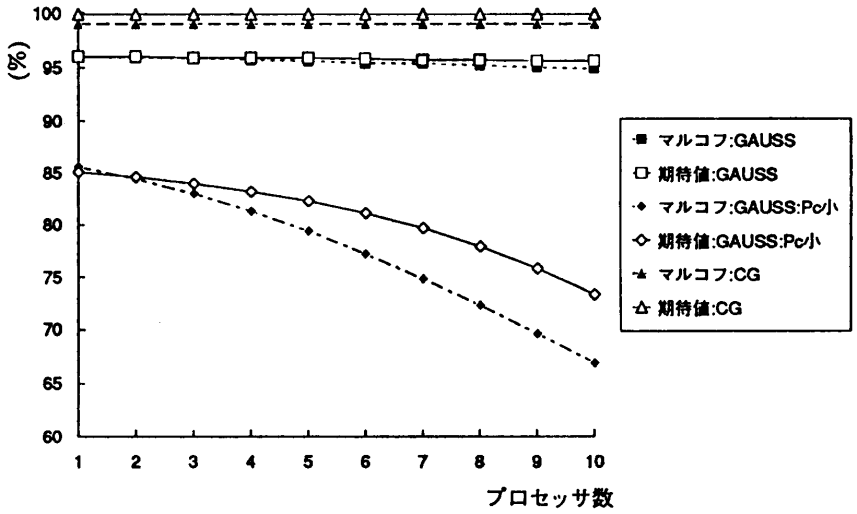


図 7 マルコフ解析と期待値モデルによる結果 (プロセッサ利用率)
Fig. 7 The number of processors VS. bus utilization ratio: result from the Morcov-analysis and the expected value model are compared.

$$c^{(k+1)}=P_c c^{(k)}+\frac{1}{t_w}w^{(k)}+(r^{(k)}-w^{(k)}r^{(k)}+P_{\text{nonblock}}w^{(k)}r^{(k)})$$

$$w^{(k+1)}=P_w c^{(k)}+\left(1-\frac{1}{t_w}\right)w^{(k)} \quad (2)$$

平衡状態では $c^{(k+1)}=c^{(k)}=c_e$, $w^{(k+1)}=w^{(k)}=w_e$ であると考えられる．この条件と(2)式から，平衡状態における各状態のプロセッサ数の期待値が求まる（付録(7)式）．

3. 並列計算機 ATTEMPT-0

本論文では，実測評価にわれわれが開発した並列計算機テストベッド ATTEMPT-0[6] を用いた．ATTEMPT-0 は Futurebus[5] を用いたバス結合型マルチプロセッサで，各プロセッサ（68030+68882）はローカルメモリと，ライトスルーキャッシュを持つ．プロセッサ間の同期はシンクロナイザ[4]と呼ばれる一種のブロードキャストメモリによって行われる．シンクロナイザは1ワードのデータごとにタグを持ち，その制御により MCM としても BCM としても使用することができる．ATTEMPT-0 では，シンクロナイザの大きさは 32Kbyte (8K エントリ) となっている．アクセスに要する時間を**表1**に示す．キャッシュは 64Kbyte，ページサイズ 64byte の2ウェイセットアソシアティブ，20MHz 動作となっている．このキャッシュではバススヌープとプロセッサアクセスを同時に行うことが可能である．キャッシュアクセスに必要な時間を**表2**に示す．さらに，ハードウェアイベントモニタによって共有バスおよびキャッシュに関する種々のデータを測定することが可能である．

4. 評　　価

4.1 評価用プログラム

ATTEMPT-0 上で以下のプログラムを実行し，測定データを採取した．

GAUSS：ガウスの消去法により連立一次方程式の解を求めるプログラム
行列は交信メモリ上に展開され，各プロ

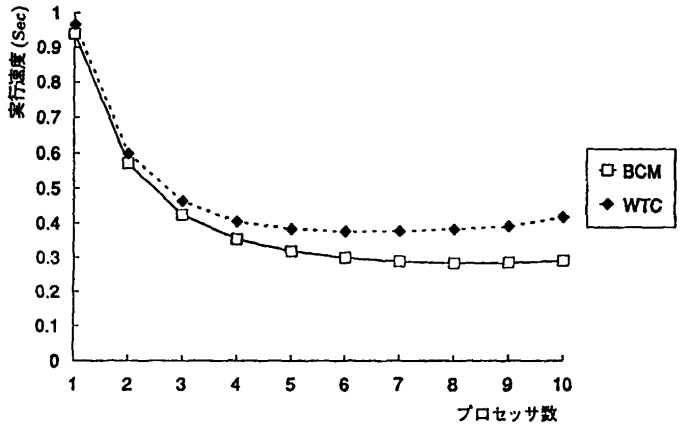


図 8 プロセッサ数対実行速度 (GAUSS)
Fig. 8 The number of processors VS. execution time (GAUSS).

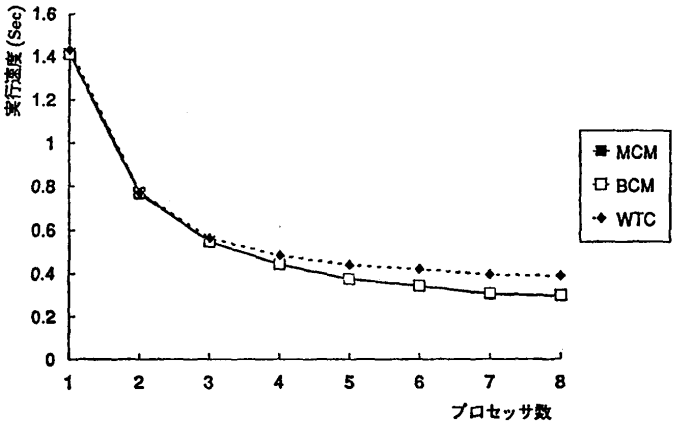


図 9 プロセッサ数対実行速度 (CG)
Fig. 9 The number of processors VS. execution time (CG).

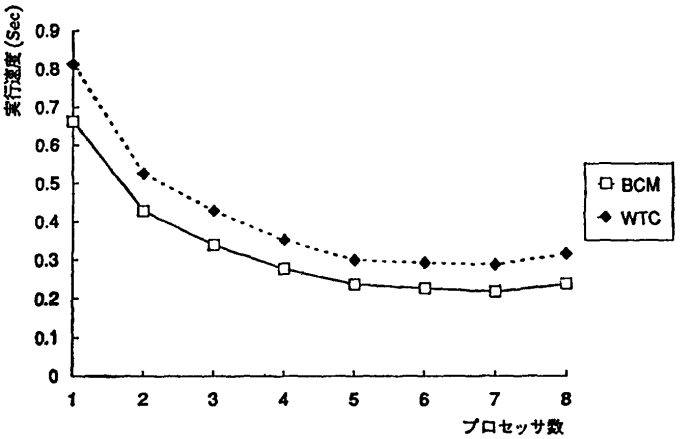


図 10 プロセッサ数対実行速度 (LOGIQUE)
Fig. 10 The number of processors VS. execution time (LOGIQUE).

セッサはそのうち数行を受け持ち，ピボットをブロードキャストして消去を並列に行う[12]．

CG：共役勾配法により連立一次方程式の解を求めるプログラム

反復法であり，(行列)×(ベクトル)，ベクトルの内積，ベクトルの更新を反復する．行列は分割されて各プロセッサのローカルメモリ上に置かれ，交信メモリ上には，ベクトルのみが置かれる．(行列)×(ベクトル) ではベクトルの各要素をプロセッサが交信メモリから読み込み積和をとる．内積では各プロセッサで担当の要素の積和を計算してから，マスタプロセッサがプロセッサ分の総和を取る．行列のサイズは 50×50，3重対角行列である．

LOGIQUE：並列論理シミュレータ

Chandy-Misra の問い合わせを用いた分散時刻管理に基づく並列論理シミュレータ[15]．スケジューリングリストはローカルメモリ上に置き，回路情報，信号の伝達を表すメッセージキュー等を交信メモリ上に置く．

ATTEMPT-0 のシンクロナイザの容量制限から，ここではシミュレーションの対象に比較的小規模な回路 (8 bit ALU) を用いた．

GAUSS と CG は共に行列計算プログラムで，交信メモリに対するアクセス間隔が規則的である点で共通であるが，交信の性質は異なる．GAUSS では全プロセッサに対するブロードキャストを多用するのに対し，CG では3重対角行列を扱うので，データは最大3プロセッサで共有される．

LOGIQUE は行列計算に比べ，各プロセッサが集中的かつ時間的に不規則に交信用メモリにアクセスする性質を持っている．

4.2 解析モデルと実測値の比較

解析モデルで必要なパラメタは，ATTEMPT-0 のハードウェア仕様と，1プロセッサで実際のプログラムを実行した結果により設定する．このデータを**表3**に示す．ただし，ライトスルーキャッシュのヒット率については並列実行時の実測値を用いた．

(1) 解析モデルと実測値の比較

GAUSS と CG について解析モデルの結果と実測値（バスの占有率，プロセッサ稼働率）を比較した．この結果を（**図5, 6**）に示す．この2つは行列計算なので一定時間計算後に交信メモリをアクセスするようにパラメタを設定した．BCM と MCM の差はわずかであったため図上では1本の線 (BCM) として扱っている．

図によると，BCM のプロセッサ稼働率は解析モデルと実測値がほぼ一致し，バス占有時間は解析モデルのほうが大きい．WTC については，交信メモリのアクセス競合による待ち時間の増大から，プロセッサの稼働率が大きく低下する．この程度は解析モデルのほうが大きい．バス占有率も同様に解析モデルのほうが大きい．一般的には，解析モデルは系の平均的な状態

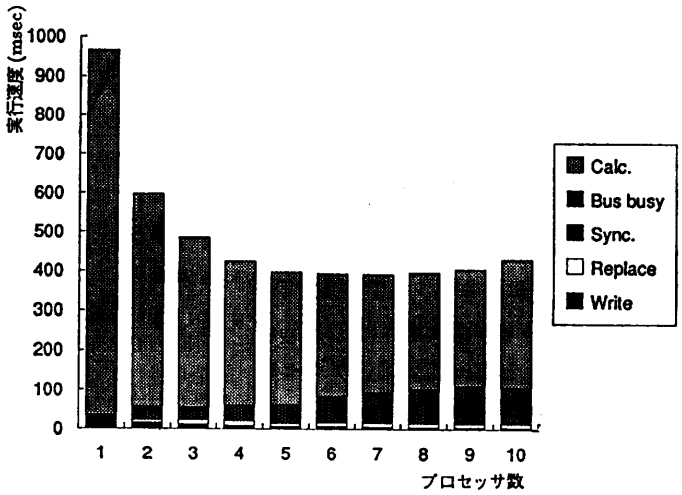


図 11(a) WTC における交信時間の割合 (GAUSS)
Fig. 11(a) Communication time of WTC (GAUSS).

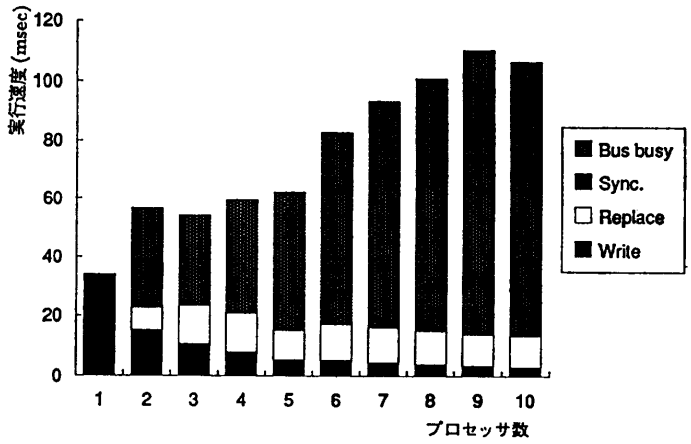


図 11(b) WTC における交信時間の内訳 (GAUSS)
Fig. 11(b) Detail of communication time of WTC (GAUSS).

を表現するため，実際よりも楽観的な結果を得るが，ここでは逆の結果が得られている．これは，本論文で用いた解析モデルがバスと交信メモリのみをモデル化しており，プロセッサ数が増加するのに伴って，プロセッサのアイドル時間が増加する点を考慮していないためである．

GAUSS では後退代入，CG では内積計算時に並列性が低下するため，プロセッサ数の増加に従ってアイドル時間が増加する．また，同期により全体の処理は最も遅れたプロセッサに足なみを揃えるので，プロセッサのアイドル時間はさらに増大する．以上の原因によるプロセッサのアイドル時間の増大により，プロセッサが交信メモリをアクセスする頻度は少なくなる．この結果バスの占有率は低下し，プロセッサの見かけの稼働率（アイドルを含む）は増大する．

実測によると，実際に交信メモリのアクセス間隔は，GAUSS において 10 プロセッサで 1 プロセッサの約 3 倍程度長くなっている．この値を考慮して解析モデルによりバス占有率を求めると，実測値とほぼ一致することが確かめられた．

以上，プロセッサ数が多い場合に解析モデルの結果が実測値と異なる原因について示したが，これは必ずしも解析モデルの有用性を損なうものではない．解析モデルは交信メモリにとって最も条件の厳しい場合の結果を出すため，これを用いれば一種のワーストケースデザインを行うことができる．

（2） マルコフ解析と期待値モデルの比較

図 7 に各アプリケーションにおけるマルコフ解析と期待値モデルによるプロセッサ稼働率の解析値を示す．この解析を行った範囲では，混雑が少ないためいずれの結果もほぼ等しくなっている．

また，GAUSS に関して P_c を小さくした場合の結果も同図に示している．この場合，期待値モデル解析ではマルコフ解析に比べ，プロセッサ数が多い時に混雑が少なくなっている．これは期待値モデルではシステムが常に平均的な状態で動作し，バスアクセスが平均的に分散すると仮定しているためである．

4.3 各種交信用メモリの性能評価

（1） 実測に基づく評価

各アプリケーションのプロセッサ数対実行時間の測定結果を図 8〜10 に示す．WTC については，さらに実行時間の内訳を図 11〜13 に示す．GAUSS，CG，LOGIQUE いずれもプロセッサ台数が増えると速度向上が鈍り，WTC を用いた場合には，むしろ遅くなる傾向を示す．これは，先に述べたようにアルゴリズム上の並列性の低下の影響が大きい．

BCM ではデータがワード単位で転送され，ブロック転送による不必要なバス使用がないので，バスの利用率が大幅に低くなる（図 5）．そのため GAUSS，CG では，特にプロセッサ数が多い場合に，WTC と比較して効率が高い．また，CG を用いて BCM と MCM を比較したところ，MCM を用いても BCM と比べて実行速度はほとんど向上しなかった．これは

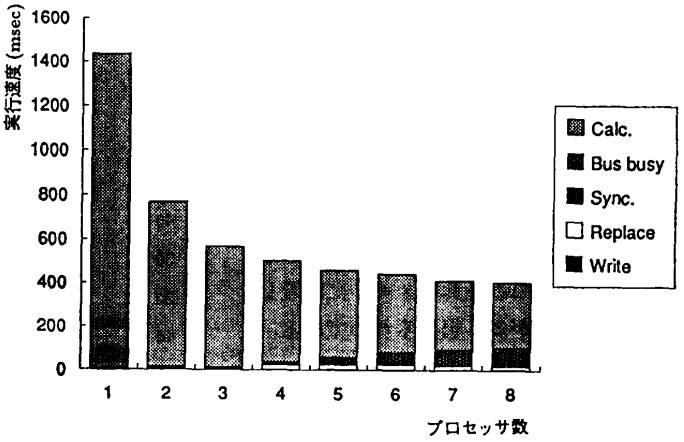


図 12(a)　WTC における交信時間の割合 (CG)
Fig. 12(a)　Communication time of WTC (CG).

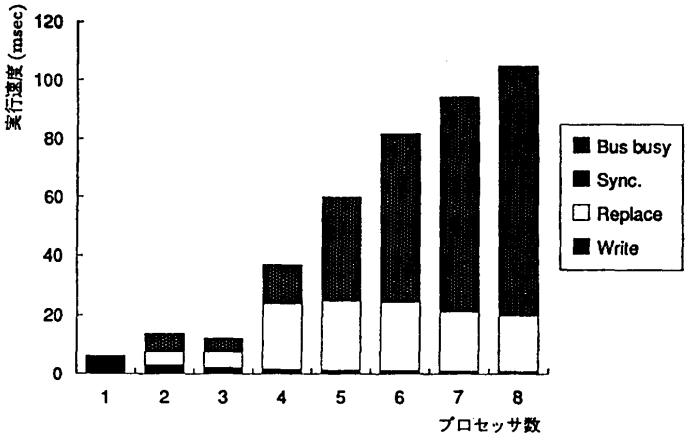


図 12(b)　WTC における交信時間の内訳 (CG)
Fig. 12(b)　Detail of communication time of WTC (CG).

CG では計算時間が実行時間の 99% を占めているため，バス上の交信とメモリ読み出しがほとんど衝突しないためと考えられる．

LOGIQUE では共有データが非常に多く，すべての共有データを BCM に配置することが困難であるため，アクセスが頻繁に行われるメッセージキューと呼ばれるデータを BCM に置いた場合と WTC に置いたもので比較を行った．図 10 に示されるように，プロセッサ数にかかわらずメッセージキューを BCM に配置したほうが 10% から 20% 程度実行速度が向上している．また，プロセッサ数が増加すると処理速度の向上が鈍化するが，これは対象とする問題の並列性の減少が原因である．

いずれのアプリケーションでも WTC ではバス混雑の影響が大きく，実行時間の 20%～30% がバスが空くのを待っている時間である．これは ATTEMPT-0 のリプレイス速度が遅いことも一因である．

WTC のヒット率を図 14 に示す．プロセッサ数が多い場合，CG のヒット率が GAUSS や LOGIQUE に比べて低い．これは CG では，頻繁にデータが更新されるベクトルの部分のみを共有メモリに配置しており，無効化が多く起こるためである．

（2） 解析モデルによる性能予測

解析モデルを用いることにより，実測できない状況の性能予測が可能である．まず，バスのアクセスが最も頻繁な GAUSS に関してプロセッサ数がさらに多くなったと想定した場合の解析結果を図 15 に示す．WTC は 10 プロセッサで既に性能低下が大きく，これ以上プロセッサ数が多い場合についての解析は意味がないので，ここでは BCM，MCM の解析結果のみを示す．

現状のプロセッサ能力とバスの転送容量を仮定した場合，アクセス競合による待ち時間が全計算時間の 25% 以内という条件では，接続できるプロセッサ数は，BCM が 25，MCM が 32 となり，かなり多くのプロセッサを接続できることがわかる．また，プロセッサ数が多く交信メモリに対して頻繁に書き込みが起こる場合には MCM が BCM よりも高い性能を示す．すなわち，プロセッサ数が 20 を越えると BCM アクセス競合待ち時間が

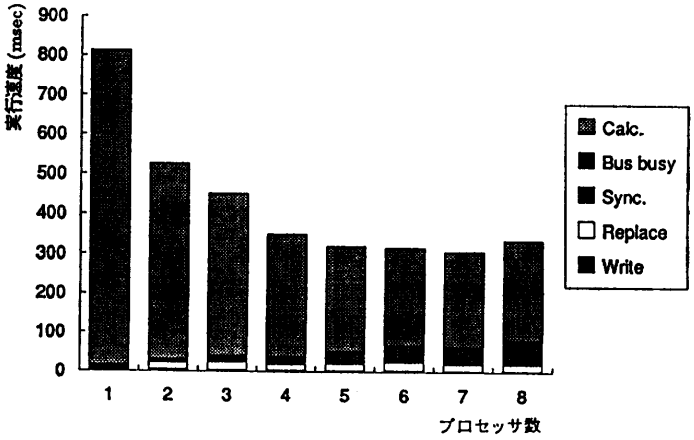


図 13(a) WTC における交信時間の割合 (LOGIQUE)
Fig. 13(a) Communication time of WTC (LOGIQUE).

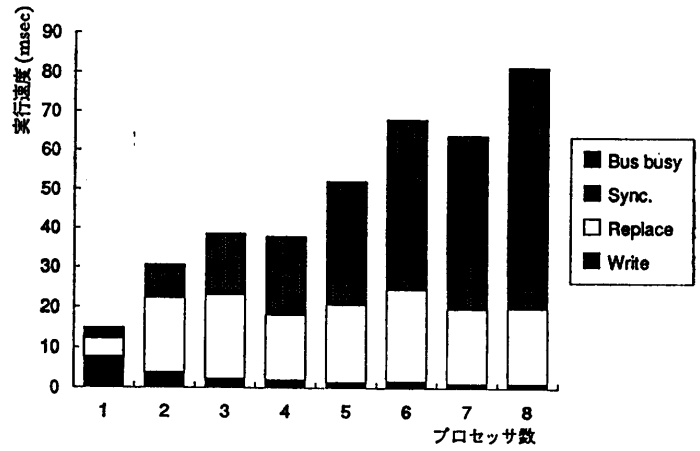


図 13(b) WTC における交信時間の内訳 (LOGIQUE)
Fig. (b) Detail of communication time of WTC (LOGIQUE).

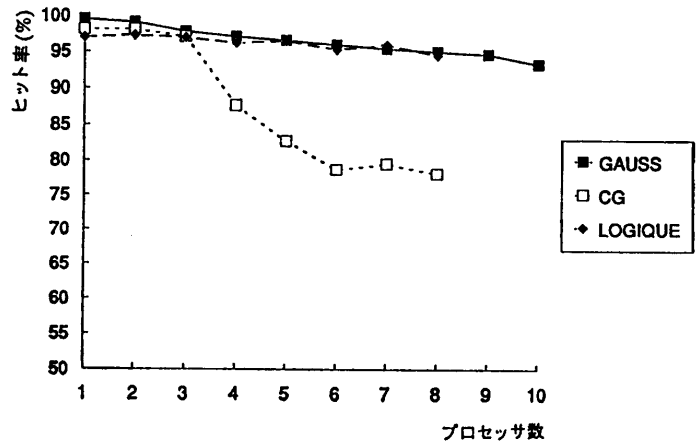


図 14 キャッシュヒット率
Fig. 14 Cache hit ratio.

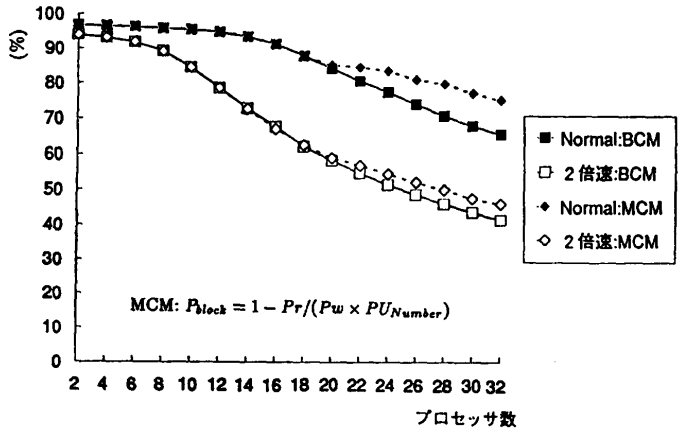


図 15　プロセッサ処理能力増加時のプロセッサ利用率
Fig. 15　Processor utilization ratio using more powerful processors.

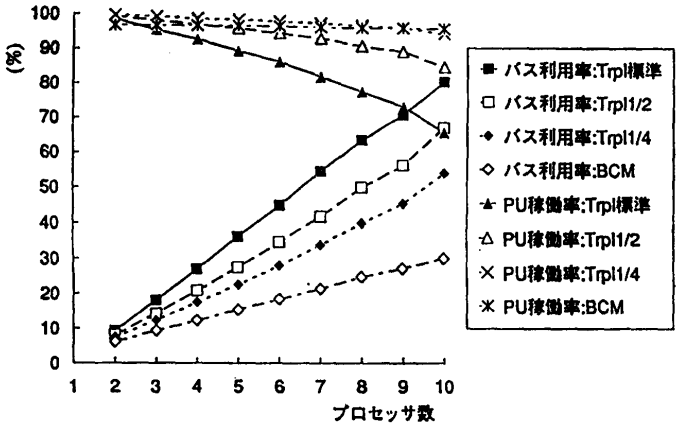


図 16(a)　WTC のリプレイス時間短縮時の結果 (GAUSS)
Fig. 16(a)　The number of processors VS. processor/bus utilization ratio with quick replacement of WTC (GAUSS).

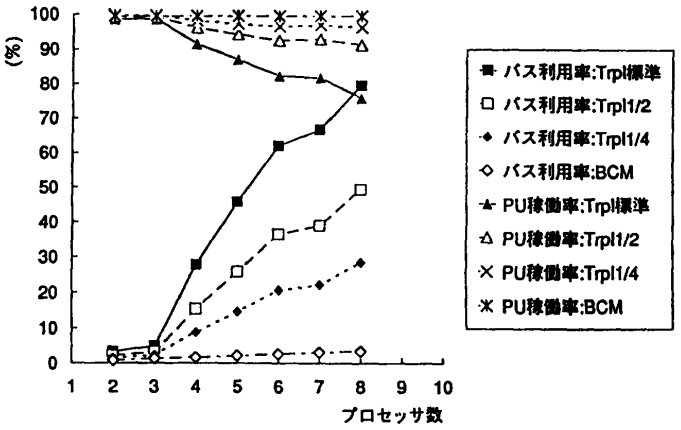


図 16(b)　WTC のリプレイス時間短縮時の結果 (CG)
Fig. 16(b)　The number of processors VS. processor/bus utilization ratio with quick replacement of WTC (CG).

急激に増大するのに対し，MCM のほうは増加の勾配が緩やかである．

ATTEMPT-0 ではプロセッサ能力とバスの転送容量はバランスしている．しかし，ハードウェア技術の発展を考えると，バスの転送容量を大きくするのは困難であるが，プロセッサの処理能力は急速に向上している．そこで，プロセッサの性能が現在の2倍になった場合を想定した結果を同じ図15に示す．この場合 MCM の優位性は小さくなっている．これは，GAUSS においては1回のマルチキャストにおいてデータを受けとるプロセッサ数が多いため，MCM が有利になるのはかなりプロセッサ数が多い場合であるが，プロセッサの性能が高いとその数に達する前にアクセス競合による待ち時間が大きくなるためである．このように MCM が効果を発揮するのはかなり限られた場合であることがわかる．

ところで，ATTEMPT-0 では，ブロックサイズが大きいため WTC のリプレイスに要する時間が他のバスアクセスに比して長い．これが，WTC の性能を落していることが考えられるため，リプレイスの時間のみを 1/2, 1/4 と想定して解析を行った (図 16)．リプレイス時間を 1/2, 1/4 にすると，GAUSS, CG 共にプロセッサ稼働率はプロセッサ数が多い時ほど高く，1/4 では 10 プロセッサ程度の範囲では BCM とほぼ同等の稼働率が得られる．また，バス利用率もリプレイス時間短縮に伴って低下しているが，1/4 の場合でも BCM に比べて利用率は高くなっている．この結果から 10 プロセッサ程度では，WTC でも十分な性能が得られるが，プロセッサ数がさらに増えた場合には，バスが混雑し不利となることがわかる．

5. おわりに

各種の交信メモリについて，解析モデルと実測による評価を行った．

解析モデルでは，プロセッサ数が多い場合，アイドル時間の増加によりアクセス頻

度が低下するという現象をモデル化できないため，実測値より混雑が激しくなるという結果が得られた．

また，BCM と MCM は，ごく限られた条件の下では MCM が有利であるが，両者共にほぼ同等の高い性能を持つことが明らかになった．一方 WTC は小規模なシステムでは，有効に働くが，バス占有率が大きいためプロセッサ数が多くなると性能が著しく低下する．コストの点も考慮すると，アドレス空間が小さい場合は，アドレス変換テーブルが不要で制御が簡単な BCM が有利であり，大きい場合はコピーの量が少ない MCM が有利であるといえる．

本論文の解析ではアプリケーションがやや行列計算に偏った．今後，より多くのアプリケーションについてアクセスの性質を調べる必要がある．また，交信データは交信メモリ上にすべて載ると仮定し，マルチキャストメモリとライトブロードキャスト型キャッシュを同じモデルで扱ったが，交信メモリを仮想化した場合の解析も必要である．

謝辞 アプリケーションプログラムを提供していただき，貴重なご助言をいただいた東京工科大学講師工藤知宏氏に感謝します．また，貴重なご助言をいただいた慶應義塾大学木村哲郎氏，寺沢卓也氏に感謝します．

参考文献

1) Sites, R. L. and Agarwal, A.: Multiprocessor Cache Analysis Using ATUM, *Proc. of 15th ISCA*, pp. 186-195 (1988).
2) Stunkel, C. B. and Fuchs, W. K.: Analysis of Hypercube Cache Performance Using Address Trace Generated by TRAPEDS, *Proc. of 1989 ICPP*, pp. 133-140 (1989).
3) O'Krafka, B. W. and Newton, A. R.: An Empirical Evaluation of Two Memory-Efficient Directory Methods, *Proc. of 17th ISCA*, pp. 138-147 (1990).
4) Amano, H., Terasawa, T. and Kudoh, T.: Cache with Synchronization Mechanism, *Proc. of 11th IFIP*, pp. 1001-1006 (1989).
5) IEEE Standard Backplane Bus Specification for Multiprocessor Architectures: Futurebus, IEEE ANSI/IEEE Std869. 1-1987 (Jun. 1988).
6) 鳥居 淳，天野英晴：並列計算機テストベッド ATTEMPT の交信機構の評価，並列処理シンポジウム JSPP '91 論文集，pp. 205-212 (1991).
7) Archibald, J. and Baer, J. L.: Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model, *ACM Trans. Comput. Syst.*, Vol. 4, No. 4, pp. 273-298 (1986).
8) Katz, R. H. et al.: Implementing a Cache Consistency Protocol, *Proc. of the 12th ISCA*, pp. 276-283 (1985).
9) 天野，地川，吉田，相磯：マルチリードメモリを用いた並列計算機の性能解析，電子通信学会論文誌，Vol. J67-D, No. 9 (1984).
10) Nakagawa, T. et al.: A Multi-microprocessor Approach to Discrete System Simulation, *Proc. on Compcon80 Spring*, pp. 350-355 (1980).
11) Amano, H., Yoshida, T. and Aiso, H.: $(SM)^2$: Sparse Matrix Solving Machine, *Proc. of the 10th ISCA* (1983).
12) 金田，小畑，前川：*BC* プロセッサアレイと高並列マトリックス計算，情報処理学会論文誌，Vol. 24, No. 2, pp. 175-181 (1983).
13) 内堀，古谷，西田：マルチリードメモリの動作解析，第 25 回情報処理学会全国大会論文集，5F-4 (1982).
14) Amano, H., Boku, T. and Kudoh, T.: $(SM)^2$-II: A Large-Scale Multiprocessor for Sparse Matrix Calculations, *IEEE Trans. Comput.*, Vol. 39, No. 7 (1990).
15) 工藤，木村，寺沢，天野：共有メモリを想定した並列論理シミュレータ，信学技報，CPSY 91-23 (1991).

付 録

$$x=w_j$$
$$y=c_j-(n-c_i)$$
$$z=(n-c_i)+c_i-w_j-c_j \qquad (3)$$
$$P_{IJ}=\begin{cases}\dfrac{c_i!}{x!\cdot y!\cdot z!}P_w^x P_c^y P_r^z & \cdots(x\geq 0, y\geq 0, z\geq 0)\\ 0 & \cdots(\text{その他の場合})\end{cases}$$

ただし，$P_r=1-P_w-P_c$ (4)

$$x=\begin{cases}w_j-w_i+1 & \cdots(s_i=t_w)\\ w_j-w_i & \cdots(s_j\neq t_w)\end{cases}$$
$$y(r_e)=\begin{cases}c_j-r_e-1 & \cdots(s_i=t_w)\\ c_j-r_e & \cdots(s_j\neq t_w)\end{cases}$$
$$z(r_e)=r_e+c_i+w_i-w_j-c_j \qquad (5)$$
$$P_{IJ}=\begin{cases}\displaystyle\sum_{r_e=0}^{n-c_i-w_i} P_i(r_e)\frac{c_i!}{x!\,y(r_e)!\,z(r_e)!} & (s_j=s_i+1, x\geq 0, y\geq 0, z\geq 0)\\ 0 & (\text{その他の場合})\end{cases}$$

ただし，

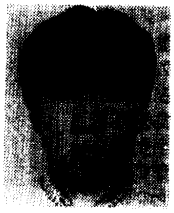
$$P_i(r_e)=\frac{(n-c_i-w_i)!}{r_e!\cdot(n-c_i-w_i-r_e)!}\cdot P_{\text{block}}^{r_e}\cdot(1-P_{\text{block}})^{n-c_i-w_i-r_e} \qquad (6)$$

$$c_c=\frac{-(P_c-nt_wP_w)-\sqrt{(P_c-nt_wP_w)^2-4nt_wP_w(t_wP_w+1)}}{2t_wP_w(t_wP_w+1)}$$

$$w_c=t_w\cdot P_w\cdot c_c$$

$$r_c=n-c_c-w_c \qquad (7)$$

(平成3年 8 月 1 日受付)
(平成3年12月 9 日採録)



鳥居 淳（学生会員）

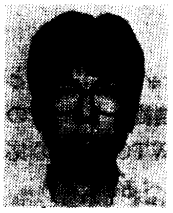
1967 年生．1990 年慶應義塾大学理工学部電気工学科卒業．並列計算機アーキテクチャ，性能評価の研究に従事．現在慶應義塾大学大学院修士課程に在学中．

天野 英晴（正会員）

1958 年生．1986 年慶應義塾大学理工学部大学院博士課程修了．工学博士．並列計算機の研究に従事．現在慶應義塾大学理工学部専任講師．著書「誰にもわかるディジタル回路」（オーム社），「並列処理機構第5章」（丸善）．

竹本 卓（学生会員）

1968 年生．1991 年慶應義塾大学理工学部電気工学科卒業．計算機アーキテクチャ，システム性能解析の研究に従事．現在慶應義塾大学大学院修士課程に在学中．



小椋 里

1969 年生．並列計算機アーキテクチャの研究に従事．現在慶應義塾大学電気工学科に在学中．