計算機アーキテクチャ　97−10
（１９９２．１２．４）

ＡＳＵＲＡクラスタの性能評価

内藤 潤，城 和貴，松野宏昭，新田博之
（株）クボタ
コンピュータ事業推進室

プロセッサ・クラスタ方式の並列計算機ＡＳＵＲＡの構成要素であるＡＳＵＲＡクラスタについて、シミュレーション及び実測を基に性能を評価する。評価目的はキャッシュの有用性とバスバンド幅であるが、簡略化のためキャッシュについてはシングル・プロセッサで評価を行なう。さらにマルチプロセッサの評価においては、単一プロセッサのキャッシュ・シミュレータとセミ・マルコフ過程によるＡＳＵＲＡクラスタの解析モデルを併用した新たな手法を提案し評価を試みる。その結果、キャッシュを効果的に利用出来るようなユーザ・プログラム・レベルでの最適化のみでも、クラスタ内でほぼスケーラブルな性能向上が得られることが確認されたことを示す。

Performance Evaluation of the ASURA Cluster

Jun NAITO, Kazuki JOE, Hiroaki MATSUNO, Hiroyuki NITTA

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

E-mail: {naito, joe, matsuno, nitta}@kubota.co.jp

ASURA is a large scale multiprocessor system, consisting of clusters of multiple bus-based shared memory multiprocessors. In this paper, we evaluate its cluster, presenting simulation results and the results obtained from a prototype machine. The aim of our evaluation is to examine the usability of the cache and the limitations of a system bus, though for simplicity the cache is examined for only the uni-processor case. A new method is introduced, which combines a uni-processor cache simulator and a Semi-Marcov model and is used to evaluate the multiprocessor performance of the ASURA cluster. Finally, we show that how efficient use of the cache allows the ASURA cluster get scalable performance.

1 はじめに

ASURA[5] は、株式会社クボタと京都大学が共同開発しているプロッセッサ・クラスタ方式の共有メモリ型マルチプロセッサシステムである。大規模なマルチプロセッサシステムを構築するに当り、小規模なマルチプロセッサを1つのクラスタとしそのクラスタを複数台結合して階層型のシステムとする方法は、有力な一方式として商用、研究用に研究開発が進められている [3][1][2][4]。このような方式のシステムにおいて、システム全体として高い性能を得るためには、クラスタ間結合のネットワーク性能及びデータ共有方式が重要であるが、クラスタ内で高性能を得られるということが前提となる。そこで本稿では、試作研究用に開発した ASURA クラスタについて評価を行なう。

ASURA は、大規模科学技術計算のような、シングルユーザ／シングルタスクから、多くのユーザが、同時にログインし個々のタスクを同時実行できるような、マルチユーザ／マルチタスク処理を実現できる汎用のマルチプロセッサを目指している。そこで、我々は要素プロセッサとして、整数演算、浮動小数点演算ともバランスのとれた性能を持つ R4000 を採用し、Stardent 社 (現 KubotaPacificComputer 社) の マルチプロセッサシステム Titan 3000 をベースに、ASURA クラスタを開発した。

性能評価は、シングルプロセッサ、マルチプロセッサともに、シミュレーション結果、実測データを併用して行なう。特に、マルチプロセッサにおいては、単一プロセッサのキャッシュシミュレータとセミ・マルコフ過程による ASURA クラスタの解析モデルを併用した新たな手法を提案し、評価を試みる。

まず、2 章で ASURA クラスタのアーキテクチャについて述べた後、3 章で評価方法、評価環境を示し、4 章でシミュレーション結果、実測結果を基に性能評価を行なう。

2 ASURAクラスタのアーキテクチャ

2.1 システム概要

ASURA クラスタは、Titan 3000 (Stardent 社) [7] [8] をベースに開発したバス結合型共有メモリマルチプロセッサシステムである。ASURA クラスタ 用にバスプロトコルの拡張及び新 CPU ボードの開発を行なった。バスプロトコルの拡張にはバスのスペアラインを使用したため、 CPU ボード以外のハードウェアは Titan 3000 と共通である。

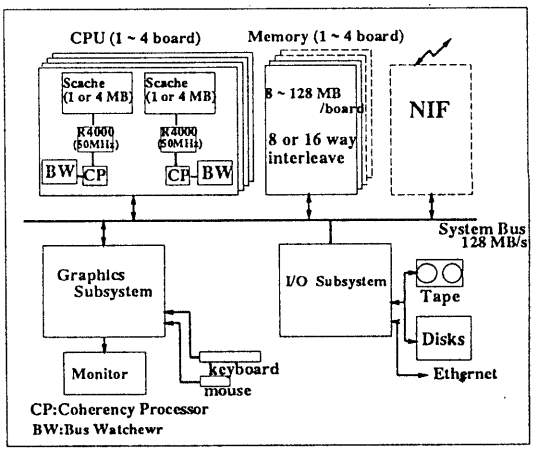

図 1: ASURA Cluster Block Diagram

システムは、CPU(1～4 枚)、メモリ (1～4 枚、但し、CPU ボードとの合計で、最大 6 枚まで)、I/O サブシステム、グラフィックスサブシステムの各ボードより構成される。ASURA 構築時には、空きスロットに NIF ネットワークボードを挿入してクラスタ間を接続する。

各 CPU ボードには 2 個の プロセッサ (50MHz R4000MC, 内部動作 100MHz) を搭載し、システムとしては最大 8 プロセッサまで拡張可能である。また、プロセッサは全て対等である。メモリボードは、1 枚の時 8 way, 2 枚組では 16 way のインターリーブを提供する。メモリ容量はボード 1 枚当たり 8～128MB であり、システムとしては最大 512MB (但し、CPU 4 枚使用時は 256MB) である。

バスは、32 bit address、64 bit data の split transaction 型で、バンド幅は、128MB/s である。アドレス送出のフェイズとアービトレーションのフェイズをオーバーラップしてるため、常時 128MB/s のバンド幅を提供できる。

2.2 プロセッサ アーキテクチャ

ASURA クラスタの 各プロセッサは、50MHz(内部動作 100 MIIz) の R4000、1MB または 4MB の大容量 2 次キャッシュ (R4000 は、命令、データ 、各 8KB の on-chip 1 次キャッシュを持つ)、2 次キャッシュ・タグのコピーを持ち、バス・スヌーピングを行なうバスウォッチャ、各プロセッサのキャッシュ間のコヒーレンス制御をサポートするコヒーレンスプロセッサより構成される。R4000 は、スーパーパイプライン方式の RISC プロセッサで、FPU を内蔵し、整数演算、浮動小数点演算ともに、高いバランスのとれた性能を発揮する。キャッシュ間

表 1: プロセッサの構成

CPU	50MHz R4000MC(内部 100MHz)
Primary Cache (on-chip)	8KB Inst. + 8KB Data, direct-mapped, write back, line size 4 word(16 byte)
Secondary cache	1MB I/D combined, direct-mapped, write back, line size 8 word(32 byte)

のコヒーレンスは、バスウォッチャ、コヒーレンスプロセッサによってハードウェアで保証される。

メモリ更新アルゴリズムは、1 次、2 次キャッシュともに、ライトバック方式であり、ラインサイズは、それぞれ 4 word(16 byte)、8 word(32 byte) である。ライトバック方式の採用及び、大容量の 2 次キャッシュでヒット率あげることによってバスのトラフィックを軽減させている。これは、バス結合型のマルチプロセッサにとって不可欠である。ASURA クラスタでは、write invalidate 型のキャッシュコヒーレンス・プロトコルを採用しているが、write invalidate 型では、ラインサイズが大きくなると無効化する時に、必要以上にデータを無効化してしまい、それによって次のキャッシュアクセスでキャッシュミス (インバリデーションミス) が起こる可能性が高くなる。一方、大容量キャッシュを持つ場合、ラインサイズがある程度大きい程、キャッシュリフィル時の効率が良い。これらを考慮に入れ、2 次キャッシュのラインサイズは、32 byte としている。

2.3 コヒーレンス プロトコル

ASURA クラスタでの、キャッシュ・ステートは、Invalid, Shared、DirtyExclusive の 3 状態である。それぞれ、そのラインが無効 (Invalid)、主記憶と内容が一致し 1 つのプロセッサに所有、あるいは複数のプロセッサに共有されている (Shared)、メモリと内容が不一致でただ一つのプロセッサに所有されている (DirtyExclusive) ことを示す。

コヒーレンス・プロコトルは、イリノイ型 [9] をベースにしている。すなわち、1) 共有ラインへの書き込み時、他のプロセッサのキャッシュ中に存在する当該ラインを無効にする (write invalidate)、また、2)Dirty ラインのキャッシュ間転送時、主記憶を常に更新する。1) で、write invalidate 型を、採用しているのは、write update 型 (共有データへの書き込時、他のプロセッサのキャッシュも更新する) は、パフォーマンス、ハードウェアコストとのトレードオフにより、ASURA クラスタでのインプリメントは有利ではないからである。2) ライトバック時に、常に主記憶を更新する方針をとっているのは、ベースとなる Titan の Bus が、プロセッサ間専用のパスをもたないこと、また、メモリシステムが、インターリーブ機構によって、毎サイクル、ライトトランザクションを受け付けることが可能であるからである。

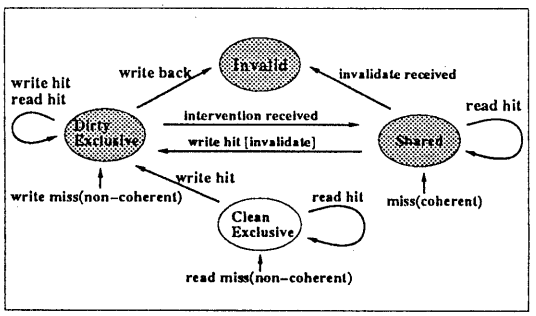

図 2: Secandary Cache State Diagram

ASURA クラスタと、イリノイ・プロトコルとの違いは、shared のラインをキャッシュ間転送しない点である。これは、ASURA クラスタでは、shared のラインはキャッシュ間転送するより、メモリから転送した方が速いからである。また、ASURA クラスタでは、主記憶と一致し、ただ 1 つのプロセッサに所有されているという状態 (CleanExclusive) を持たない。これは、CPU として採用している R4000 による制限からである。R4000 の場合、CleanExclusive から DirtyExclusive への状態変化が、System Interface 上に見えない、つまり、ASURA クラスタの様にタグのコピーを持たせる場合、タグに状態変化を反映させることができない。従って、ASURA クラスタでは、CleanExclusive 状態をもってはいない[1]。CE 状態を持たない場合、ライトミス時及び 1 回目のライトヒット時にインバリデートリクエストの発行を伴うという点で不利であるが、バスウォッチャにタグのコピーを持ち、コヒーレンス制御をパイプライン処理できるメリットの方が大きい。

ASURA クラスタのメモリアクセス、コヒーレンス処理のコストを表 2に示す。2 次キャッシュのミスコストは、ラインリプレイス (ライトバック) を伴う場合、伴わない場合の 1.3 倍である。また、他のプロセッサのキャッシュからリフィルする場合のコストは、メインメモリからリフィルする場合の 1.6 倍である。また、ライトミスのコストは、CE 状態を持つ場合の 1.3 倍となる。

2.4 バスプロトコルの拡張

Titan 3000 のキャッシュコヒーレンス・プロトコルは、ライトスルー方式をとっており、バスプロ

[1] non-coherent 領域に対しては、例外的に許している。

トコルとしてキャッシュコヒーレンス制御のサポートは必要とせず、したがってシステムバスには、特別な信号を備えていない。そこでASURAクラスタでの、コヒーレンス・プロトコルをインプリメントするために、システムバスに、11本の制御信号を追加した（全信号の6%)。また、8プロセッサ間でのラウンドロビン・アービトレーションを行なうために、アービタを再設計した。

2.5 プロセッサ間通信

ASURAクラスタでは、クラスタ内CPUの同期用に3種類のプリミティブを用意している。セマフォ、ドアベル、及び、R4000のLoad Linked & Store Conditional命令である。前2者は、Titan3000で使用されているものであり、ASURAクラスタでも実装した。セマフォは、メモリシステム上にインプリメントしてあり、test-and-clear, test-and-increment命令をサポートする。ドアベルは、任意のプロセッサ間でのインタラプトを実現するメカニズムで、システムバス上のブロードキャストプロトコルとして実装している。このメカニズムは、オペレーティングシステムによって使用される。LL & SCは、クラスタ内のみであるが、他はクラスタ間にも拡張可能である。

3 評価方法

シングルプロセッサ、マルチプロセッサともに、シミュレーション結果及び実測データを基にして評価する。シングルプロッセッサについては、キャッシュの効果に重点を置き、また、マルチプロセッサについては、バスバンド幅に重点を置いて評価する。

3.1 シングルプロセッサ

行列積計算

倍精度の行列積計算を、行列サイズ（100、200、300、400、500、1000）を変えて行ない、行列サイズと実行速度の関係を調べる。またその時の、キャッシュ・ヒット率、使用するバスバンド幅を求める。キャッシュ・ヒット率は、3.3節のシミュレータによるシミュレーション結果により、使用バンド幅については、シミュレーション結果による総メモリトラフィック[2]と実際の計算実行時間によりそれぞれ得られる。なお、キャッシュ中データの再利用を考慮したプログラムと、そうでないもの2種類について実行する。

3.2 マルチプロセッサ

使用バスバンド幅とスケーラビリティ

プロセッサ当りの使用バスバンド幅とマルチプロセッサで実行した時のスケーラビリティの関係を調べる。実行プログラムは、単にメモリコピーをするもので、シングルプロセッサでの実行時に使用バスバンド幅がそれぞれ、20MB/s、26MB/sのものを使用し、これを1、2、4、6、8プロセッサにおいて実行した。各プロセッサのメモリアクセスパターンは同じである。

セミ・マルコフモデルを使ったシミュレーション

ASURAクラスタによる並列プログラム実行時の実測は、後で述べるように実行環境が整っていないこともあって、現在のところ困難である。そこで、本稿では、単一プロセッサのキャッシュ・シミュレータとセミ・マルコフ過程によるASURAクラスタの解析モデルを併用することにより、ASURAクラスタ全体の性能を評価する手法を提案する。

pl（3.3節）はASURAクラスタの中の一つのプロセッサに着目して、そのキャッシュのシミュレーションを行なうツールであるが、クラスタ全体の実際の動きは、他プロセッサからのキャッシュ制御命令等が複雑に作用し合うため、plだけを用いて全体の性能を予測することは困難である。

一方、我々はASURAクラスタの解析的性能評価の手段として、セミ・マルコフ過程を用いたモデル化を行なっている[6]。このモデルはキャッシュのヒット率等の単一プロセッサに対する情報をパラメータとして与えてやると、全体のキャッシュ・コヒーレンス制御を考慮に入れた性能評価を出力するもので、システム全体の定常状態を評価するのに用いられる。

一般に、確率過程等のモデル化では、システムの定常状態の性能を評価することは可能であるが、実際の並列化プログラムを走らせた場合のように、ある程度詳細な解析は困難である。よって、我々はplによる並列化プログラムの実行結果を、ある一定の時間間隔でサンプリングし、それをセミ・マルコフ過程を用いたASURAクラスタのモデルにパラメータとして入力してやることにより、並列化プログラムの挙動をシステム全体から見た解析結果として得ることを試みた。

本手法の利点は、実際のASURAクラスタ全体のシミュレータを開発して解析を行なうのに比べ、計算時間が圧倒的に少ないにも関わらず、システ

[2]シミュレーションできるのは、データアクセスのみであり、命令のリフィルについては、この場合のトラフィックには入れていない。コードサイズが小さいため無視する。

表 2: キャッシュミス & コヒーレンス処理コスト

	転送元	ラインリプレイス	コスト (プロセッササイクル)
Primary miss	Secondary Cache	なし	10
		あり	16
Secondary miss	メインメモリ	なし	111
		あり	144
	他プロセッサのキャッシュ	なし	183
		あり	216
invalidate	N/A	N/A	33

ムの時間的な挙動を解析出来ることである。つまり、実際のシミュレータと解析モデルの中間に位置する評価手法と言えよう。

3.3 キャッシュ シミュレータ

シングルプロセッサのキャッシュ・シミュレーションについては、ASURA クラスタ用に開発したアーキテクチュアル・シミュレータ pl(postloader) を用いた。pl は、ポストロード処理によって、オブジェクトファイルにトレース用ライブラリを埋め込む。簡潔にいえば、実行形式ファイル (a.out format) を読み込み、全ての load/store 命令の後に Cache Simulation library への call routine の挿入、ライブラリ挿入によってずれるジャンプアドレスの修正を行ない、新しいオブジェクトファイルを生成する。新しく生成されたオブジェクトファイルを実行することによって、キャッシュヒット率、ラインリプレイス数、実行された命令数、メモリトラフィック 等の情報が得られる。ただし、pl は、Titan3000 上にインプリメントしているため、オブジェクトファイルが R3000 上で実行可能なコードである必要がある。

3.4 プログラムの開発、実行環境

現段階で、ASURA クラスタ用のオペレーティングシステムは開発途上であり、また、コンパイラも整備されていない (アセンブラは、暫定版が使用可能)。よって、今回評価として実行したプログラムは、全てスタンドアロンであり、プログラムは、Titan3000 の C コンパイラ (R3000 用) によって作成した。ただし、浮動小数点演算で影響の大きい、FPU の ロード／ストア命令については、アセンブラ段階で mips II 命令 (ldc1/sdc1) に変換を行なった。実行時間計測は、CPU ボード上のタイマ (33MHz) による。

4 評価

4.1 シングルプロセッサ

行列積計算

図 3(a) のアルゴリズムによる行列積計算の結果を図 4、5、6に示す。行列サイズが 300 を越えると性能が落ち始める。これは、(a) の場合、最外側ループ毎に Y の全要素へアクセスするため、行列サイズがキャッシュサイズを越えた場合 (キャッシュ 1MB に納まるのは、倍精度で 353x353) には、Y へのアクセスに対して毎回ミスを起こすからである。行列サイズ 500 での性能は、データがキャッシュ中に納まってる時の 1/2 である[3]。また、トータルでのキャッシュミス率は 8%、使用するバンド幅は、29MB/s である。

キャッシュを考慮した行列積計算

前節のような問題に対して、メモリ階層を効果的に利用する手法としてブロッキングが良く知られている。全行、全列を一度に計算するのではなく、より CPU へ近い方の階層にロードしたデータを再使用するよう部分行列 (サブブロック) 毎に計算する。500x500 の行列積計算に対して、ブロッキングによって計算した結果を図 7、8、9に示す。ブロッキングのアルゴリズムは図 3(b)[10] による。適度なブロックサイズを選ぶことにより、行列サイズが大きくなっても性能を落さずに実行できることがわかる。バス結合マルチプロセッサでは、特にキャッシュの有効利用により、速度向上はもとより、バストラフィックを軽減させることが重要である。本評価では、ブロッキングしない場合の 29MB/s 対して、1/10 程度に抑えられている。バストラトラフィックとマルチプロセッサの効率については、4.2節で述べる。

[3] ここでは FPU パイプライン・スケジューリングの最適化は行なっていない。最適化を行なった場合、キャッシュよりデータサイズが小さい場合は、倍近く性能向上し、また、大きい場合は性能向上がほとんど見られないため、更に悪く 1/3 ～1/4 程度となる

```
(a)
   for i:= 1 to N do
      for k:= 1 to N do
         for j:= 1 to N do
            Z[i,j] = Z[i,j] + X[i,k] * Y[k,j];

(b)
   for kk:=1 to N by B do
      for jj:= 1 to N by B do
         for i:= 1 to N do
            for k:= kk to min(kk+B-1,N) do
               for j:= jj to min(jj+B-1,N) do
                  Z[i,j] = Z[i,j] + Z[i,k] * Y[k,j];
```

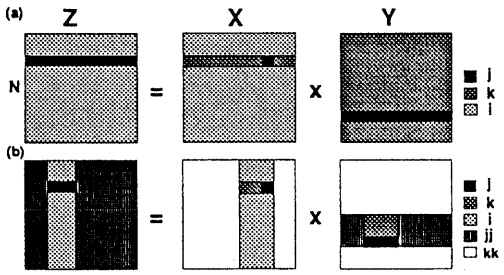

図 3: 行列積計算のデータアクセスパターン (a)UnBlocked (b)Blocked

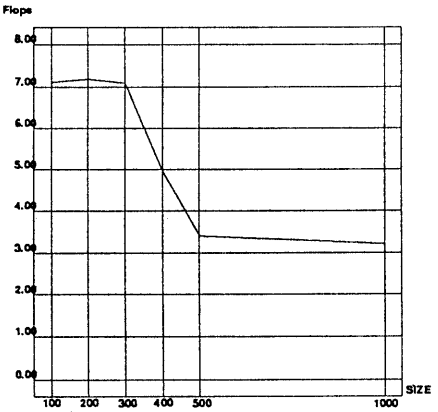

図 4: 倍精度行列積計算の結果 (MFlops)

4.2 マルチプロセッサ

使用バスバンド幅とスケーラビリティ

各プロセッサの使用バスバンド幅が、プロッセサ当り 20MB/s、26MB/s であるプログラムを並列実行した時の結果を図 10に示す。システムバスのバンド幅が 128MB/s であるため、バス、メモリへのアクセス競合がなければ、速度向上の上限は、それぞれ 6.4 倍、4.9 倍であり、それぞれ 6 プロッセッサ、5 プロセッサまでスケーラブルな結果が得られることになる。しかし、実際には各プロセッサとも、同様のメモリアクセスを行なっているため、バスアービトレーション、メモリインタ

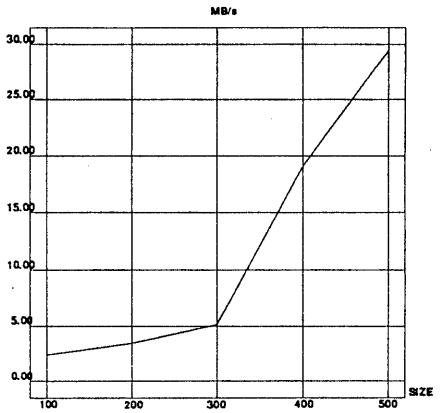

図 5: 倍精度行列積計算の結果 (バスバンド幅 :MB/s)

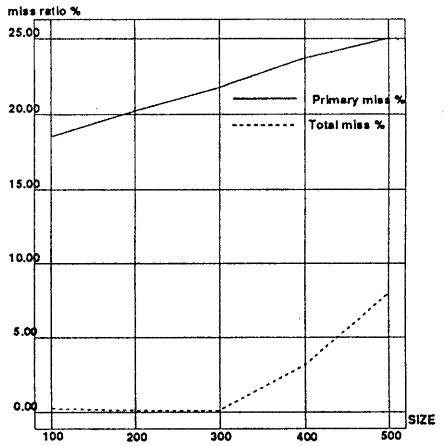

図 6: 倍精度行列積計算の結果 (キャッシュミス率:%)

リーブビジーにより待ち状態が生じる。

セミ・マルコフを使ったシミュレーション

8 プロセッサでの 1000x1000 の行列積演算において、ブロッキングした場合 (ブロックサイズ =100) としない場合についてのシミュレーション結果 (各処理の比率、各リソースの利用率) を図 11、12、13、14 に示す[4]。セミ・マルコフモデルへの入力データは、2000 サイクルごとの pl の出力結果である。図では、代表的な計算部分を一部抜

[4]COMP:計算、CACHE:キャッシュ、MEM:メインメモリ、COHE:コヒーレント動作 (インバリデイト、ライトバック)、NWAIT:キャッシュミス時における待ち、CWAIT:コヒーレント動作時における待ち、BUS:システムバス、PROC:プロセッサ

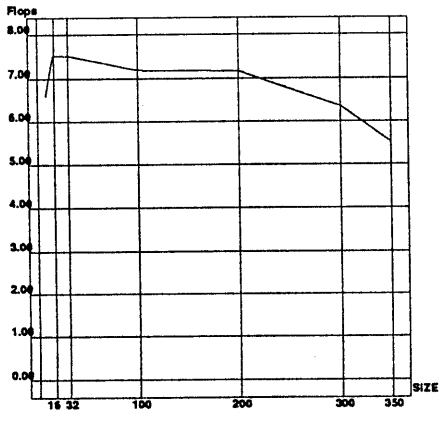

図 7: ブロック倍精度行列積計算 (500x500) の結果 (MFlops)

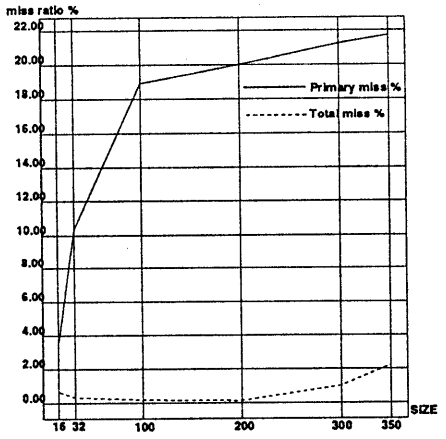

図 9: ブロック倍精度行列積計算 (500x500) の結果 (キャッシュミス率:%)

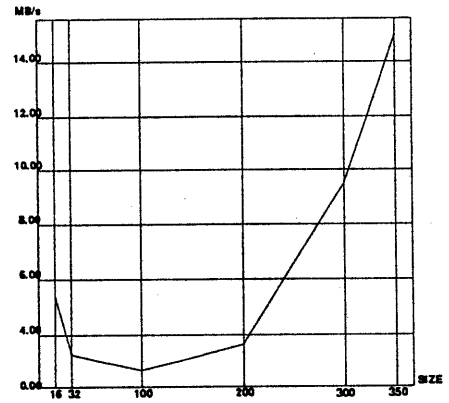

図 8: ブロック倍精度行列積計算 (500x500) の結果 (バスバンド幅:MB/s)

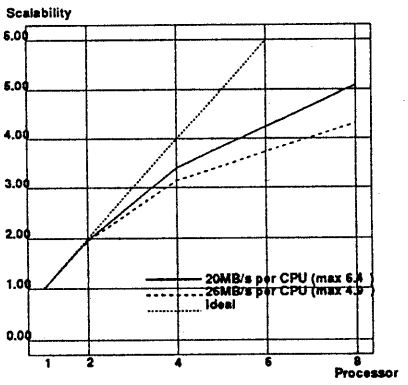

図 10: シングルプロセッサの使用バス幅とスケーラビリティ

きだしている。ブロッキングしない場合、計算時間の割合が 5%で、80% 以上がバスの待ち状態となっている。また、プロセッサの稼働率は 8 % である。これはバスが飽和していることを示している。これに対しブロッキングした場合には、待ち時間が数 %、プロセッサの稼働率は 80〜100 %であり、8 台のプロセッサがほぼ無駄なく稼働していることがわかる。

5 まとめ

プロセッサ・クラスタ方式のマルチプロセッサシステムである ASURA のクラスタについて性能評価を行なった。その結果、キャッシュを効果的に利用出来るようなユーザ・プログラム・レベルでの最適化のみでも、クラスタ内でほぼスケーラブルな性能向上が得られることを確認した。

現在、ASURA クラスタは、2 台のシステムが稼働しており、ASURA OS の開発に使用されている。

謝辞

共に、ASURA クラスタのハードウェア設計に携わった (株) クボタ コンピュータ技術部の方々、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、ASURA プロジェクト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致します。

参考文献

[1] Daniel Lenoski et al. The directory-based cache coherence protocol for the DASH multiprocessor.

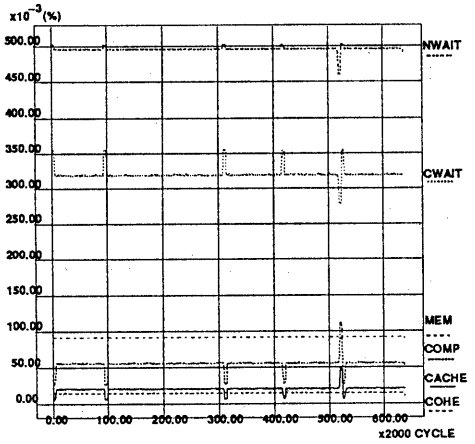

図 11: セミマルコフシミュレーション結果 (比率)

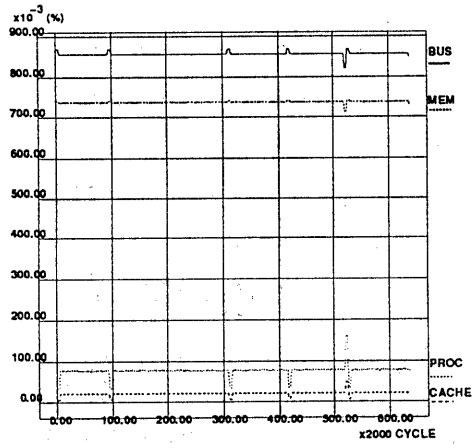

図 12: セミマルコフシミュレーション結果 (利用率)

In *Proceedings of the International Symposium on Computer Architecture*, pages 148–159, 1990.

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

[3] 日本アライアントコンピュータ. *The CAMPUS/800 Supercomputer.* 技術資料, 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] 森 眞一郎, 斉藤 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.

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

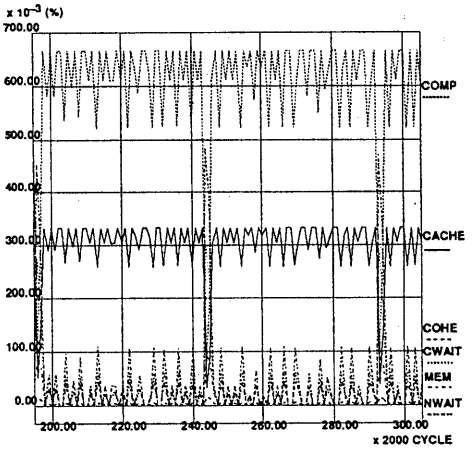

図 13: セミマルコフシミュレーション結果 (Blocking)(比率)

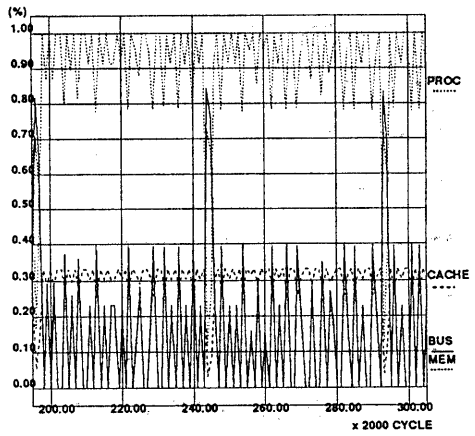

図 14: セミマルコフシミュレーション結果 (Blocking)(利用率)

[7] T. Diede, C F. Hagenmaier, G S. Miranker, J J. Rubinstein and W S. Worley,Jr. "The Titan Graphics Supercomputer Architecture" *IEEE COMPUTER*,13–30, Sep.1988

[8] "Titan 3000 ", Kubota Computer Inc. 1990.

[9] J. Archibald and J.-L. Baer. "Cache coherence protocols: Evaluation using a multiprocessor simulation model", *ACM Trans. on Computer Systems*, 4(4):273–298, Nov. 1986.

[10] M S. Lam, E E. Routhberg and M E. Wolf "The Cache Performance and Optimizations of Blocked Algorithms", In *Proceedings of the Sixth International Conferrence on Architecutual Support for Programming Langages and Operating Systems*, April 1991.