

トレース駆動型シミュレーションによる 共有バス結合型 MP の拡張方式の評価

高橋正人、前田慎司、宮田裕行、菅隆志

三菱電機(株) 情報システム研究所

共有バス / 共有メモリ構成マルチプロセッサシステムは、プログラマビリティに優れ、ユニプロセッサシステムの自然な拡張として実装も比較的容易であるため、この構成に基づく多くの商用マシンが登場している。一方この構成はバスが性能ボトルネックになり、プロセッサの拡張性に乏しいという欠点がある。そこでプログラマビリティの良さを維持したまま、より高い拡張性を実現するためいくつかの方式が提案されている。本稿ではその中から、多重インタリーブバス構成と、階層バス構成に注目し、それらの構成と通常の単一バス構成についての台数効果の比較を、トレース駆動型シミュレーションにより示す。作業負荷には、一台の UNIX ワークステーションにおいて複数のユーザがプログラム開発、文書作成を行なう状況をシミュレートする SPEC SDM sdet を用いた。その結果、今回用いたトレースにおける、16 プロセッサまでのマルチプロセッサ構成では、階層バス構成は単一バス構成とほぼ同等の性能であったが、多重インタリーブバス構成はバスコンテンションの減少効果が大きく単一バス構成よりも拡張性に優れることが判明した。

A Performance Evaluation of Scalable Bus-based Architectures using Trace-driven Simulation

Masato Takahashi, Shinji Maeda, Hiroyuki Miyata, Takashi Kan

Computer & Information Systems Laboratory, Mitsubishi Electric Corporation

5-1-1, Ofuna, Kamakura, Kanagawa, 247, Japan

As shared-bus, shared-memory type multiprocessor architecture has good programmability and relatively straightforward implementation, there are many commercial machines based on this architecture. However, this architecture has limited scalability. Recently, several bus-based architectures have been proposed which achieve higher scalability while preserving good programmability. This paper represents a trace-driven study on the performances of two of those architectures: the multiple interleaved bus architecture, and the hierarchical bus architecture. We use a part of SPEC SDM sdet as our workload. Our result indicates the multiple interleaved bus approach achieves higher scalability than the single bus or the hierarchical bus approach.

1 はじめに

共有バス / 共有メモリ構成マルチプロセッサシステムは、ユーザにデータ分割の問題に直面させることなく、逐次型計算機と同様のプログラミング環境を提供し、逐次型計算機での既存のソフトウェア資産もそのまま活用できる。また実装も、ユニプロセッサシステムの自然な拡張として、比較的容易であるため、この構成に基づいた多くの商用マシンが登場している。

しかしこの構成の短所として、バスが性能のボトルネックになり、プロセッサ数の拡張性に乏しいという問題があげられる。そこでバスベースの構成の長所を活かしたまま、より高い拡張性をもつ構成として、多重インタリーブバス構成 [Tung91] と、階層バス構成 [Wils87] が提案されている。

多重インタリーブバス構成の一例を図1に示す。参照アドレスに応じて二次キャッシュとバスおよびメモリがインタリーブされており、バス負荷が分散される。これにより、バスにおける競合を低下させ性能ボトルネックとなることを防ぎ、より高い拡張性を実現しようとするものである。

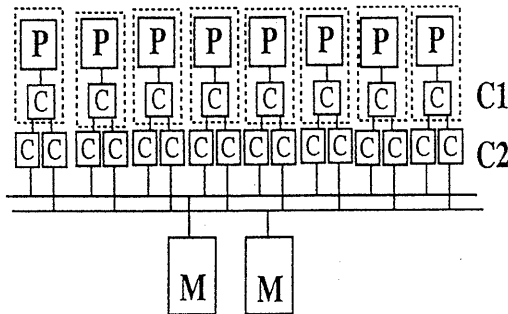


図1: 多重インタリーブバス構成 [Tung91] の一例

階層バス構成の一例を図2に示す。これはバスを階層化することによりバス負荷を分散し、より高い拡張性を実現することを目指すものである。なお、最近のマイクロプロセッサの構成がそうであるようにオンチップの一次キャッシュを持つことを想定し、図1、図2においては、チップ部分を点線で囲んで示している。

本稿では、単一バスから成る共有バスマルチプロセッサシステムの、拡張性に富む後継として、どちらがよりふさわしいかの関心から、トレース駆動型シミュレーションによって台数効果の評価を行った結果を報告する。

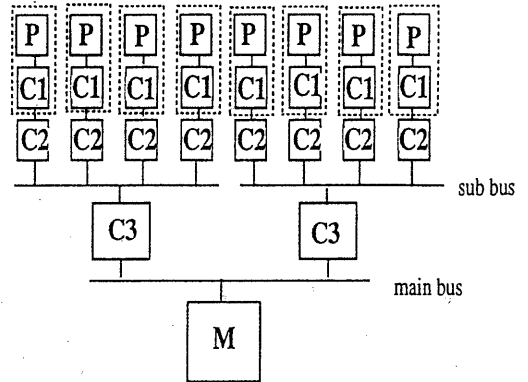


図2: 階層バス構成 [Wils87] の一例

2 評価方法

2.1 作業負荷とトレース

作業負荷として、UNIX が稼働するマルチプロセッサシステム に対する標準的ベンチマークのひとつである SPEC SDM sdet [Case92] を用いた。SPEC SDM sdet は、UNIX 環境において複数のユーザが、プログラム開発ないしドキュメント作成作業を並行作業している負荷を実現するベンチマークである。ここでは16ユーザからなる負荷を設定した。

トレース採取ツールとして g88 [Bedi90] を用いた。g88 は、Motorola 社の RISC プロセッサ 88100 の挙動をエミュレートし、この上で評価対象とする OS およびアプリケーションを動作させることができるソフトウェアである。g88 はまた、16 プロセッサ¹までのマルチプロセッサ構成をシミュレートでき、この際プロセッサごとの命令レベルメモリ参照トレースが採取できる。この機能を用いて、複数台の 88100 を共有バス結合した構成で、対象型マルチプロセッサ機能を持つ UNIX OS である OSF/1 を稼働させ、上記の作業負荷を与えたときの各プロセッサのメモリ参照列をトレースとして採取した。以上の状況で、約 8500 万レコードから成るトレースを採取した。

¹[Bedi90] に示された g88 では、マルチプロセッサ構成として4プロセッサまでの構成に限定されていたが、今回16プロセッサまでのマルチプロセッサ構成に対応するように拡張した。

2.2 評価対象のマルチプロセッサ構成

以下に今回、評価の対象とした、単一バス構成、多重インタリーブバス構成、階層バス構成のパラメータの設定について述べる。原則として、それぞれの構成間において、プロセッサ数およびキャッシュ容量（とその他のキャッシュパラメータ）およびバスに関する仕様が同一であることを、性能比較の条件としている。

2.2.1 単一バス構成

単一バス構成については、プロセッサ数は、1, 4, 8, 16 を評価の対象とした。一次キャッシュはライトスルーのオンチップキャッシュを想定し、容量 8KB のデータキャッシュと、容量 8KB の命令キャッシュを持つ分離型とした。一次キャッシュのラインサイズはデータキャッシュ、命令キャッシュ共に 32B であり、マッピング方式は 4 ウエイアソシエティブを想定した。

2 次キャッシュは容量 256KB のデータ・命令統合型キャッシュとした。ラインサイズは 64B であり、マッピング方式はダイレクトマップを想定した。スヌーピングキャッシュコヒーレンシプロトコルとして、イリノイプロトコル [Papa86] によって管理されるものとした。イリノイプロトコルは、今回と同一のキャッシュ・バス性能および同一の作業負荷状況下において、バークレイプロトコル [Katz85] とならんで、性能的に最適と評価された [高橋 93] ものである。

バスに関しては、C-busII [Coro92] あるいは Futurebus+ [Fbus91] とした近年のバスの仕様を反映させ、高負荷時にはバス使用中にもバックグラウンドでアービトレーションが可能な機構を備えるものとした。バス幅は 8B、プロセッサと同じ周波数で作動することを想定した。

タイミング情報に関しては、一次キャッシュのヒットは 0 ウェイト、二次キャッシュのヒットは 1 ウェイトとした。バストランザクションに関しては、キャッシュ間データ転送が 13 サイクル、メモリキャッシュ間データ転送が 18 サイクル、無効化信号が 3 サイクルとした。アービトレーションのコストは 6 サイクルとした。

2.2.2 多重インタリーブバス構成

多重インタリーブバス構成では、プロセッサ数は、4、8、16 を評価の対象とした。プロセッサチップは、単一バス構成の構成要素と同一のものをを用いることを前提としているため、オンチップ一次キャッ

シュのパラメータは単一バス構成のものと同一とした。

この構成では、二次キャッシュはインタリーブされているが、比較の公平さのため、ひとつのプロセッサが抱える二次キャッシュの容量の合計が、単一バス構成の二次キャッシュ容量と一致するように、256KB となるように設定した。

インタリーブの方法については、単一バス構成と、多重インタリーブバス構成で二次キャッシュにおけるデータアドレスのマッピングが変わらないように配慮し、 n -インタリーブ構成においては、単一バス構成の容量 256KB の二次キャッシュを先頭から n 等分し、別個のキャッシュに設定した。このように設定することにより、単一バス構成と多重インタリーブバス構成の二次キャッシュにおけるコンフリクトミスの発生率が両者の間で大きな差を引き起こすことを避け、性能比較の対象からマッピング方式の影響を排除した。

二次キャッシュのその他のパラメータは単一バス構成と同一とし、タイプは統合型キャッシュ、ラインサイズは 64B、マッピング方式はダイレクトマップを想定した。スヌーピングキャッシュコヒーレンシプロトコルには、やはり単一バス構成と同じく、イリノイプロトコルを、インタリーブされたバスに結合する各々のキャッシュに対して採用した²。

個々のバスの仕様およびタイミング情報は単一バス構成のものと同一とした。

2.2.3 階層バス構成

階層バス構成では、プロセッサ数 16 のみを評価の対象とした。1 つのサブバスが 4 台のプロセッサを持ち、メインバスが 4 つのサブバスを持つ構成である。プロセッサチップは、単一バス構成の構成要素と同一のものをを用いることを前提としているため、オンチップ一次キャッシュのパラメータは単一バス構成でのものと同一とした。二次キャッシュも、容量、ラインサイズ等すべてのパラメータを単一バス構成の二次キャッシュと同じに設定した。

複数の二次キャッシュの上位にバスを介して存在する三次キャッシュとして、ダイレクトマップの統合型キャッシュを想定した。容量は 2MB、ラインサイズは 64B である。

二次キャッシュと三次キャッシュの各々には、イリノイプロトコルを拡張し、階層バス構成に対応

²多重インタリーブバスでは、バスは複数になっているものの、スヌーピングキャッシュコヒーレンシプロトコルを各々のバスに対して適用すればよく、既存のスヌーピングキャッシュコヒーレンシプロトコルを使用できる。

した書き込み無効化型スヌーピングプロトコルを設計し、用いた。二次キャッシュは、Clean Exclusive, Clean Sharable, Dirty Exclusive, Invalid の4状態を持つ。三次キャッシュは、Clean Sharable, Dirty Exclusive, Invalid に加えて最新データが自分の下位キャッシュにあることを示す状態の4状態を持つ。

このプロトコルは、読み込みに関してはデータの共有を許すが、書き込みに関してはデータの共有を許さない。書き込みの際には、書き込みを行なったプロセッサが所有する一次キャッシュおよびその上位にある二次キャッシュと三次キャッシュ以外に存在しているデータのコピーは無効化される。この際、性能向上を目的として、二次キャッシュへのデータ書き込みの際には、直接の上位キャッシュである三次キャッシュへはデータ書き込みそのものは同時には行わず、二次キャッシュにおいてデータ更新がなされたことのみが直接の上位キャッシュである三次キャッシュへ通知され、そこから無効化信号がメインバス上に発行される。同時に、この三次キャッシュでは、該当エントリの状態フラグが「最新データが自分のいずれかの下位キャッシュに存在する」ことを示す状態へと変更される。その後、三次キャッシュへの該当エントリへのバスリードがあった場合は、三次キャッシュの状態フラグにしたがって、三次キャッシュはリード要求を下位のバスに送り、二次キャッシュから最新データが供給される。

なお、タイミング情報に関しては、三次キャッシュのヒットを1ウエイトとし、一次キャッシュ、二次キャッシュのヒットについては単一バス構成の設定と同じとした。また、バストラザクションに関しても、単一バス構成の設定と同じく、キャッシュ間データ転送が13サイクル、メモリキャッシュ間データ転送が18サイクル、無効化信号が3サイクル、アービトレーションが6サイクルとした。

この階層バス構成と性能比較される単一バス構成は、三次キャッシュに相当するものを持っていない。このため性能比較の面で、このことが、階層バス構成に有利に働くことが予想された。しかし結果的に、この有利な条件にもかかわらず、性能は単一バス構成と同等かそれ以下という結果となっているため、性能の比較において実質的に問題はなかった。

3 評価結果

3.1 各構成における台数効果

評価の結果得られた、各構成における台数効果のグラフを図3に示す。

その結果、今回使用したトレースの下では、以下のことが明らかとなった。

- 単一バス構成において、8プロセッサ付近で性能の飽和が見られる。
- 多重インタリーブバス構成は、単一バス構成にみられる性能の飽和をみることなく、16プロセッサに至るまで性能を伸ばしている。
- 16プロセッサの階層バス構成は、16プロセッサ単一バス構成と同等程度の性能しか達し得なかった。

以下では、シミュレーション過程でのプロセッサの稼働状況を見ることによってこれらの結果をもたらした要因を検討する。

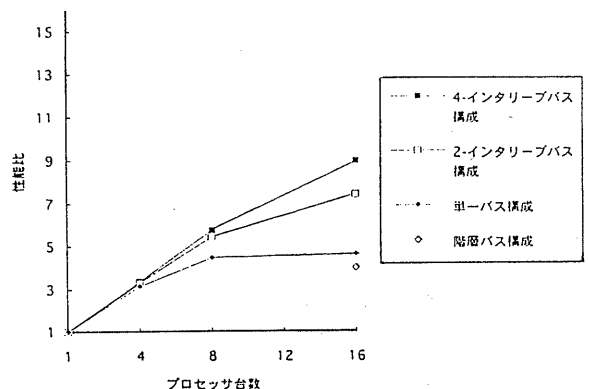


図3: 各構成におけるプロセッサ台数 - 性能比

3.2 多重インタリーブバス構成の効果

まず、多重インタリーブバス構成の効果を見るために、プロセッサ台数8と、16における単一バス構成と、多重インタリーブバス構成の、プロセッサ稼働状況の比較を図4に示す。

バスを単一のものから多重インタリーブにすることによって、バス解放待ちの時間割合が減少し、プロセッサの効率的な実行に寄与していることがわかる。

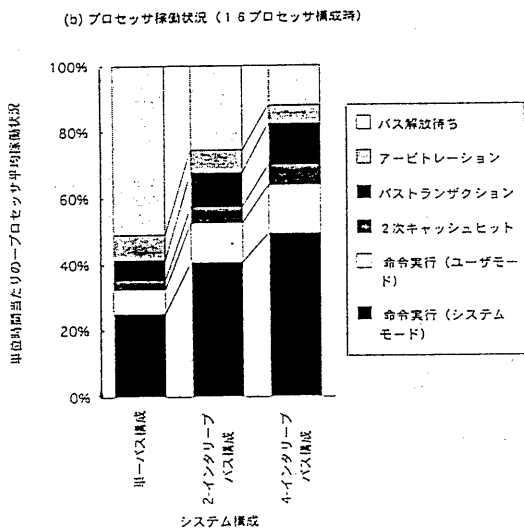
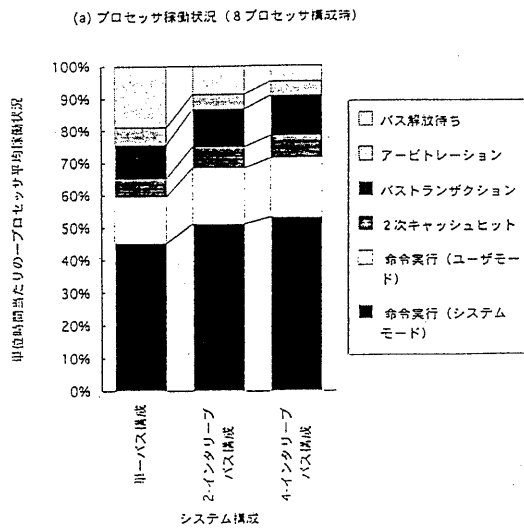


図4: 単一バス構成と多重インタリーブ構成でのプロセッサ稼働状況

グラフの視点を変えて、単一バス構成で、プロセッサ数を1~16台まで拡張したときのプロセッサ稼働状況の内訳を図5に示す。プロセッサ数の増加につれて、バス解放待ちをしている時間が実行時間のほぼ半分を占めるようになっており、プロセッサの実質的効率が落ちている。これがプロセッサ8台以上で性能が飽和する原因であると考えられる。

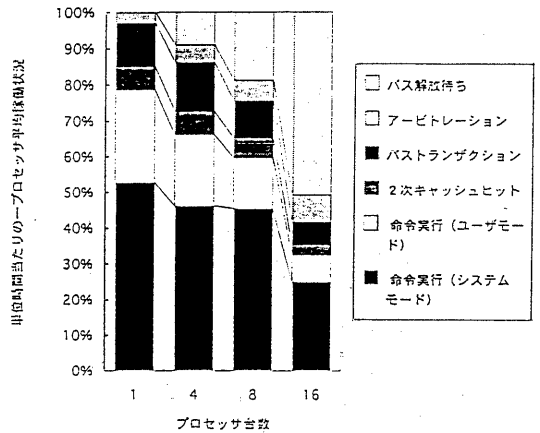


図5: 単一バス構成での拡張時におけるプロセッサ稼働状況

一方、4-インタリーブバス構成で、プロセッサ台数の拡張を図った場合のプロセッサ稼働状況の内訳を図6に示す。4-インタリーブバス構成では、16台プロセッサ構成でもバス解放待ちの増加は単一バス構成に比べてわずかであり、プロセッサの命令実行を圧迫していない。

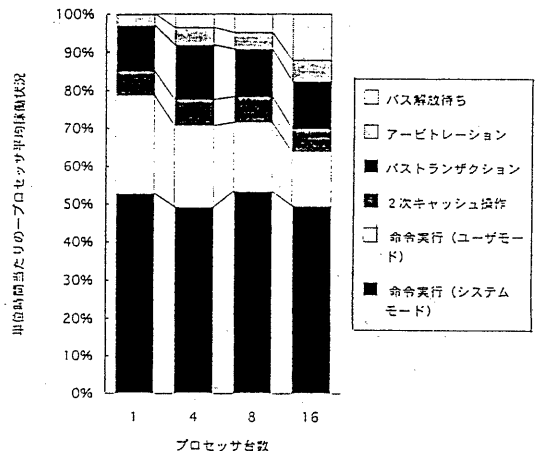


図6: 4-インタリーブバス構成での拡張時におけるプロセッサ稼働状況

3.3 階層バス構成の効果

16 プロセッサからなる階層バス構成におけるプロセッサ稼働状況を図7に示す。図7中の、“バス解放待ち(ローカルバス)”は自クラスタ内のローカルバスの解放を待つ時間、“バス解放待ち(メインバス)”自クラスタ内のローカルバスをすでに獲得しておりその上でメインバスの解放を待つ時間、“バス解放待ち(リモートバス)”はローカルバス、メインバスともに獲得した上で、さらに他クラスタのリモートバスの解放を待つ時間、の割合である。

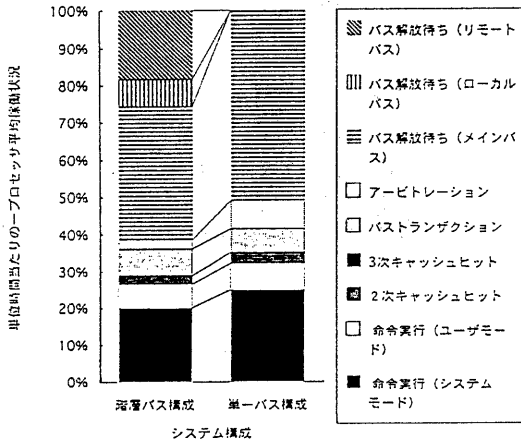


図7: 階層バス構成でのプロセッサ稼働状況

階層バスで性能がでなかった原因は、やはりバス待ち時間の多さであることが示されている。メインバスのバス使用率は97.6%、それ以外のバスの平均使用率は52.9%であった。この構成では、他のクラスタへのリモートバスへのアクセスがある場合、自クラスタのローカルバスだけでなくメインバスも占有してしまうためバスの競合が大きくなった結果であると考えられる。

一方、階層バス型の評価の結果はトレースの種類にも大きく依存する。今回のトレースは、ローカルバス内で処理が閉じるような種類の科学技術計算プログラムでなく、一台のUNIXワークステーション上での複数のユーザによるプログラム開発、および文書作成を想定したSPEC SDM sdetであったため、階層バス構成にとって最も不向きな作業負荷における評価例と考えることもできる。

4 まとめ

ユーザに対してデータ分割の問題を課すことのない共有メモリ/共有バス型マルチプロセッサは、プログラマビリティの良さという長所を持つと同時に、拡張性の乏しさという短所を合わせ持っている。そこで、プログラマビリティの良さを維持したまま、拡張性を向上させるシステム構成の候補として、いくつかの候補が挙げられている。本稿では、それらの構成の中から、多重インタリーブバス構成と階層バス構成に関して、トレース駆動型シミュレーションによりプロセッサ数16までの性能評価を行なった。作業負荷としてSPEC SDM sdetを用いUNIX環境上での16ユーザによるプログラム開発およびドキュメント作成作業の負荷を与えた。この結果、16プロセッサの階層バス構成に関しては、今回の使用トレースの条件では、単一バスを越える性能は見られなかった。また単一バス構成が8プロセッサ程度でバス飽和が生じ、それ以上のプロセッサ数の増加が実質的な性能向上をもたらさない一方、これを多重度4のインタリーブバス構成にすることにより16プロセッサにおいてもバス飽和を生じることなく性能向上がみられ、拡張性が高められることが示された。なお今回用いたSPEC SDM sdetとは異なる特性を持つ作業負荷についても現在評価を行なっている。

謝辞

本研究の遂行にあたり、日頃から貴重な御助言をいただき次世代方式技術開発部の川田部長に感謝いたします。

参考文献

- [Tung91]Cheng-Hsien Tung, On Large Scale Shared-Memory Multiprocessor Systems, Proc. of the 4th ISMM/IASTED Int. Conf. Parallel and Distributed computing and systems, Washington D.C., Oct. 1991, pp.139-143
- [Wils87]Andrew W. Wilson Jr., Hierarchical Cache/Bus Architecture for Shared Memory Multiprocessors, Proc. 14th Int. Symp. Computer Architecture, 1987, pp.244-252
- [Bedi90]Robert Bedicheck, Some Efficient Architecture Simulation Techniques, USENIX - Winter 1990 pp.53-63
- [Case92]Brian Case, Microprocessor Report - Updated SPEC Benchmarks Released, Sept. 16, 1992, pp. 14-19
- [Papa86] M.Papamarcos et.al., A Low-Overhead Coherence Solution for Multiprocessors with Private Cache

Memories, Proc. of 11th Int'l Symposium on Computer Architecture, IEEE, 1985, pp.414-530

[Katz85] R. Katz et.al., Implementing a Cache Consistency Protocol, Proc of 12th Int'l Symposium on Computer Architecture, IEEE, 1985, pp.276-283

[Coro92] C-busII Specification Revision 1.1, Corollary, Inc., California. 1992

[Fbus91] IEEE Standard for Futurebus+ Physical Layer and Profile Specification, IEEE Std 896.2-1991

[高橋 93] 高橋, 前田, 宮田, 菅、トレース駆動型アーキテクチャシミュレータによる多段キャッシュを持つ共有バス結合型MPの性能評価, 情報処理学会研究会報告 93-ARC-102, 1993