並列・分散システム用ハードウェアモニタの構成について†

白川洋充†† 油谷聡††† 丹波覚††††

近年，VLSI 技術の急速な進歩により共有メモリ型，分散メモリ型あるいはデータフロー型の並列・分散システムが研究レベルから実用レベルに達し，種々のシステムが市場にも出現してきている．並列・分散システムの性能評価は，通常ソフトウェアモニタあるいはハードウェアモニタによってなされる．ソフトウェアモニタは，システムのグローバルな性能評価には適しているが，ソフトウェアモニタ自身のオーバヘッドが無視できないという欠点を持っている．一方，ハードウェアモニタは，イベントのトレースをオーバヘッドなしに測定できるという利点がある．並列・分散システムでは負荷分散のようなダイナミックな性能改善をハードウェアモニタの助けを借りて行わなければならず，その必要性がますます高まっている．本論文では，ノードと同程度のハードウェア量で構成され，しかもノードの CPU とローカルメモリ間のバスにプローブを挿入するだけで動作可能な分散メモリ型の並列・分散システムに適合するハードウェアモニタの設計法を提案し，その実現について述べるものである．また，実験的な分散オペレーティングシステムにおいて実際にハードウェアモニタを用いてイベントを測定した結果を示す．これにより，本ハードウェアモニタが並列・分散システムの性能評価に適用可能であることを示すものである．

1. はじめに

近年，VLSI 技術の急速な進歩により共有メモリ型，分散メモリ型あるいはデータフロー型の並列・分散システムが研究レベルから実用レベルに達し，種々のシステムが市場にも出現してきている．これに伴い，並列・分散システムのためのオペレーティングシステムの研究が盛んになってきた．Vシステム[1),2)]や Mach[3)] のような分散オペレーティングシステムの研究が代表的なものである．これらの研究と同時に，並列・分散システムの性能評価の方法論に関する研究が盛んになってきている[4)]．分散オペレーティングシステムの性能評価がこれらの研究のメインテーマであることは言うまでもないことである．

さて，分散オペレーティングシステムの構築は設計から始まって，試作，性能評価，最後にシステムのチューニングを行い，再び性能評価のサイクルに入る過程を経る．デバッグはこのサイクルの随所に入りうる．

分散オペレーティングシステムの性能評価は，通常ソフトウェアモニタあるいはハードウェアモニタによってなされる．ソフトウェアモニタは，システムのグローバルな性能評価には適しているが，ソフトウェアモニタ自身のオーバヘッドが無視できないという欠点も持っている．一方，ハードウェアモニタは，オーバヘッドなしにシステムの挙動が測定できるという利点があるが，柔軟なシステムでないと限られたイベントしか測定できないことと価格の問題がある．

並列・分散システムでは分散オペレーティングシステムの構築が済んだ後にも並列・分散プログラムを効率よく実行するために負荷分散のようなダイナミックな性能改善を行わなければならない．この場合にはハードウェアモニタの助けを借りなければ効率のよい負荷分散が行えず，ハードウェアモニタの必要性がますます高まっている．

並列・分散システムを構成する各 CPU やローカルメモリをノードと呼ぶ．共有メモリ型の並列・分散システムの各ノードが高機能化し複雑であるのに比べて，分散メモリ型の並列・分散システムの各ノードは，高機能の CPU とローカルメモリ，その他にはノード間の通信を行う専用プロセッサだけがあるという比較的簡単な構成をとるものが多い．しかし，逆にノードの数は数百から数万台に達するものがある[5),6)]．

共有メモリ型あるいはデータフロー型の並列・分散システムの性能評価を行う目的で開発された大規模のハードウェアモニタは多数発表されている[4),7),8)]．

しかしながら，システムの稼働時にも使用する目的で設計された分散メモリ型の並列・分散システムのための小型のハードウェアモニタは存在しない．ノード

† An Organization of a Hardware Monitor for Parallel and Distributed Systems by HIROMITSU SHIRAKAWA (Department of Computer Science and Systems Engineering, Faculty of Science and Engineering, Ritsumeikan University), SATOSHI YUTANI (BPD, Video Products Development Division, Sony Corp.) and SATORU TAMBA (R & D Department, Electronics Division, DAIKIN Industries, Ltd.).

†† 立命館大学理工学部情報工学科
††† ソニー(株) B & P 開発本部映像開発部門
†††† ダイキン工業(株)電子機器事業部技術部

の数がきわめて多い分散メモリ型の並列・分散システムの各ノードに従来の大がかりなハードウェアモニタを用いることは現実的でない．本論文では，ノードと同程度のハードウェア量で構成され，しかもノードの CPU とローカルメモリ間のバスにプローブを挿入するだけで動作可能な分散メモリ型の並列・分散システムに適合するハードウェアモニタの設計法を提案し，その実現について述べるものである．

本論文で提案するハードウェアモニタは小型で，かつノードの CPU と相補関係を保ちながら並列に動作し，ノード上のプログラムの挙動をモニタし解析結果を CPU にフィードバックするものである．このことは，分散メモリ型の並列・分散システムのためのハードウェアモニタを LSI 化し，コプロセッサのようにノード上に搭載することが可能であることを示している．

本論文では，2章で分散メモリ型の並列・分散システムのためのハードウェアモニタの設計思想について述べる．3章は設計の詳細を説明する．4章では実験的な分散オペレーティングシステムを対象にして実際にハードウェアモニタを用いてイベントを測定した結果を示す．これにより，本ハードウェアモニタが並列・分散システムの性能評価に適用可能であることを示すものである．

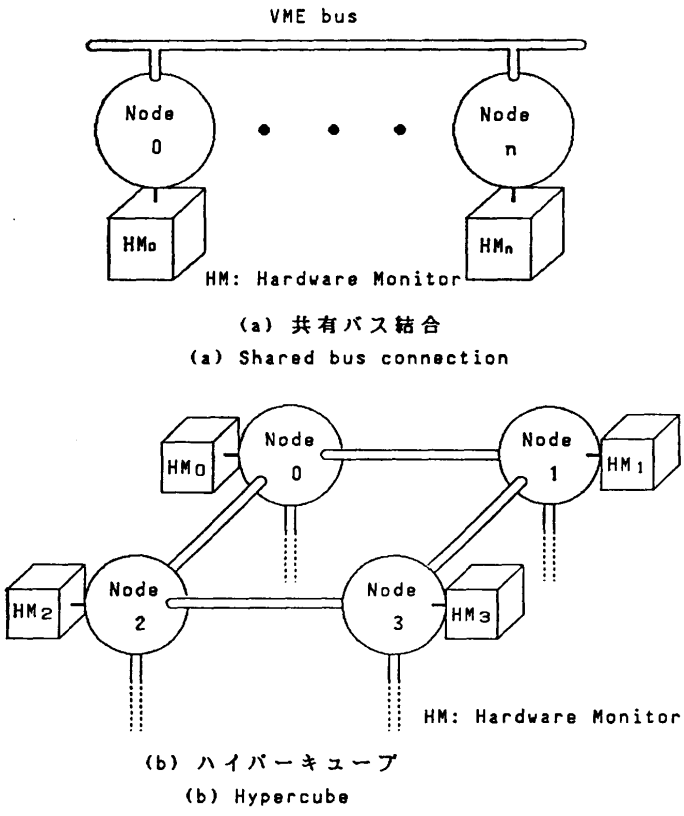


図 1 分散メモリ型の並列・分散システム
Fig. 1 Distributed memory parallel and distributed systems.

2. ハードウェアモニタの設計思想

本ハードウェアモニタは分散メモリ型の並列・分散システムの各ノードに付加させ，システムの性能評価に適用することを目的として設計された．図1に本システムが対象としている分散メモリ型の並列・分散システムを示す．

一般にノードの数がきわめて多い分散メモリ型の並列・分散システムに対応するためには，小型でかつ容易にノードと接続可能なハードウェアモニタが求められる．したがって，ハードウェアモニタとしては，ノードと同程度のハードウェア量で構成され，各ノードの CPU とローカルメモリ間のバスにプローブを挿入するだけで動作可能であることが必要になる．

計算機の性能評価を行うための測定は次のような概念に基づいて行う．Svobodova[9] と Ferrari[10] によっても提案されているように，システムの挙動はシステムの状態の遷移を観測することによって知ることが可能である．したがって，イベントは状態の遷移に相当する．一方，動作中の計算機の状態あるいは実行中のプログラムの状態はメモリの中に記憶されている．メモリの内容はメモリ参照によってのみ変化する．したがって，イベントはメモリ参照を行うことに相当し，以下，単にメモリ参照を行うことをイベントと称することにする．メモリのあるフィールドにビットをセットすることやプログラムカウンタの内容が変化することもイベントである．

次に，ハードウェアモニタを用いて計算機の性能評価を行う際，一般的に次のような測定を行うことが重要である．

1. イベントのカウント
2. イベントのトレース

イベントのカウントとは，あるメモリバンクのアクセス頻度を計るといったような，イベントの生起回数をカウントすることである．一方，イベントのトレースとは，ある選ばれたイベントのシーケンスをイベン

トが生起した順序で記録することをいう.

分散メモリ型の並列・分散システムのためのハードウェアモニタでは,イベントのトレースにタイムスタンプを付けて記録することが重要である.「将来の並列計算機の計装技術」というワークショップが,ACM の主催で1988年5月に開かれた.このワークショップで分散メモリのワーキンググループは,ハードウェアモニタの持つべき機能として,次のような機能を持つことが望ましいという合意に達した[11].「システムのダイナミクスを知るためにはイベントのトレースが必要である.このイベントはグローバルなクロックによってタイムスタンプが押され,順序付けられていることが必要である」.

以上のことを考慮にいれて本ハードウェアモニタは次の特徴を持つように設計されている.

1. 広範囲なモニタドメイン
2. 高入力レート
3. 広い入力データ幅
4. 大記録容量
5. 高解像度
6. イベントの連続測定が可能
7. プログラム可能
8. ノード間との通信パスの存在
9. モニタのハードウェア構成はノードのハードウェア構成に依存しない
10. グローバルなクロックが各ノードに供給されること

1. から 5. は Svobodova が定義したハードウェアモニタが持つべき性能の五つである[9].

1. の広範囲なモニタドメインを有するということは,仮に測定対象のシステムが 32 ビットのアドレスを有するとすると 2^{32} 通りの検出を可能にすることである.従来の並列・分散システムのハードウェアモニタ,例えば C.mmp[7] や,EGPA pyramid[8] ではプローブをそれぞれ 16 本と 8 本用意しており,各プローブに対して一つのコンパレータを使用してイベントの検出を行っている.最近のオペレーティングシステムではマルチスレッドをサポートするなどコンカレント性が高まり,性能評価のために測定すべきイベントの数はきわめて多くなってきている.これに対し,プローブの数を増やしたり,各プローブに対して限られた数のコンパレータで対応するのは適切でない.コンパレータを使用するかぎり,2^{32} 通りの検出を行うには,2^{32} 個のコンパレータを用意することになり現実的ではない.モニタドメインを限定せずに,どのようにしてすべてのイベントの検出を可能にするかが大きな問題になる.また,イベントはアドレス空間上で広範囲に分布している.そこで,本ハードウェアモニタでは,これをハッシュ法を用いて圧縮し,小規模のテーブルに前もって記憶されたイベントと比較・検出するという方法を採用している.この方法は,NBS の方式にみられるような,入力のデータを三つのフィールドに分けてそれぞれのフィールドについて,テーブルに前もって記憶されたイベントがあるかどうかを検出していく方式[12]とは異なっている.ハッシュ法を用いる方が,広範囲に分布するイベントを一様に圧縮し,小規模なテーブルを用いて比較・検出が容易にできるという特徴を持っている.

2. から 5. はハードウェアモニタとして望ましい性能であり,価格と性能のトレードオフである.特に,2. に関しては,ハードウェアモニタの処理をイベントの符号化,テーブル・ルックアップと比較,データの記録という三つのフェーズに分けてパイプライン処理を行い,高入力レートを達成しているのが特徴である.

6. については十分大きな記録容量を2面用意した.この2面のうち1面が記録に使用される.これと並行してすでに記録された他の1面のデータを解析し,終了と同時にその結果を,8. の通信パスを介してノードにフィードバックする.このように,2面の記録用のメモリを設けることにより,イベントの連続測定が可能となる.並列・分散システムでは負荷分散のようなダイナミックな性能評価をしなければならないためにイベントの連続測定が必要となる.

7. についてはハードウェアモニタをより柔軟なシステムにするためのものである.

8. についてはノードからハードウェアモニタにコマンドを送信したり,ハードウェアモニタの解析結果をノードにフィードバックするためのものである.

9. については今後ますます高速化する CPU を搭載するノードに対応するためのハードウェアモニタの適応性と柔軟性である.本ハードウェアモニタは,メモリのサイクルタイムに注目し,メモリのサイクルタイムにパイプライン処理のクロック周波数を対応させ,CPU のクロック周波数とは独立させた.このことにより,ノードのハードウェア構成には依存しないようにすることができた.技術の進歩にともない,CPU のクロック周波数は急速に増加しているのに比

べ，メモリのサイクルタイムの減少は緩慢である．本ハードウェアモニタの最大入力レートはメモリのサイクルタイムで制限されるだけなので，どのような高機能の CPU に対しても適応性がある．

10. については本システムでは，グローバルなクロックが各ノードに供給され，その信号が同時にハードウェアモニタに供給されグローバルな時刻を得る構成となっている．

以上が，本ハードウェアモニタの特徴である．

イベントのカウントとトレースはロジック・アナライザを用いても測定できる．しかしながら，上記の特徴 6. から 8. のように，イベントを連続測定し，そのデータをプログラムにしたがって解析し，その結果をノードにフィードバックするということはできないのが現状である．

3. ハードウェアモニタの構成

ハードウェアモニタが検出するイベントとは，任意のメモリ参照の検出と同じ意味である．本ハードウェアモニタシステムでは，ノードのアドレスバスを監視してイベントの検出を行っている．これにより，CPU が命令フェッチを行う際のプログラムカウンタの値を知ることができる．同様に，任意のデータへのアクセスも検出することができる．ノードとハードウェアモニタの CPU はともに MC 68000 (8 MHz) を使用している．したがって，イベントとしては，アドレスバスの 23 ビットを考えることになる．

本ハードウェアモニタは，イベントを登録するイベントテーブルを持っている．あらかじめ，検出したいアドレスをこのイベントテーブルに登録しておき，監視中の対象システムのアドレスバスと，イベントテーブルに登録されたアドレスとを比較し，一致すればイベントを検出したことになる．なお，イベントテーブルへのイベントの登録は，プログラマブルに行うことができる．

図 2 のブロック図に示されるように本ハードウェアモニタは，大きく分けて，イベントの符号化，テーブル・ルックアップと比較，データの記録，解析部により構成される．

イベントの符号化は，符号化回路によりノードのアドレス信号を符号化する部分である．イベントテーブルは，アクセスタイム 45 ns の SRAM で構成されている．符号の総数は，$2^8=256$ 個であるが，同一符号には 4 個のイベントを登録することができ，最大で 1,024 個のイベントが登録可能である．

テーブル・ルックアップと比較の部分では，イベントの符号化の部分から送られた符号に対するイベントが，イベントテーブルから呼び出される．そして，呼び出されたイベントとアドレスラッチに保持されているアドレスの比較をコンパレータで行う．比較した結

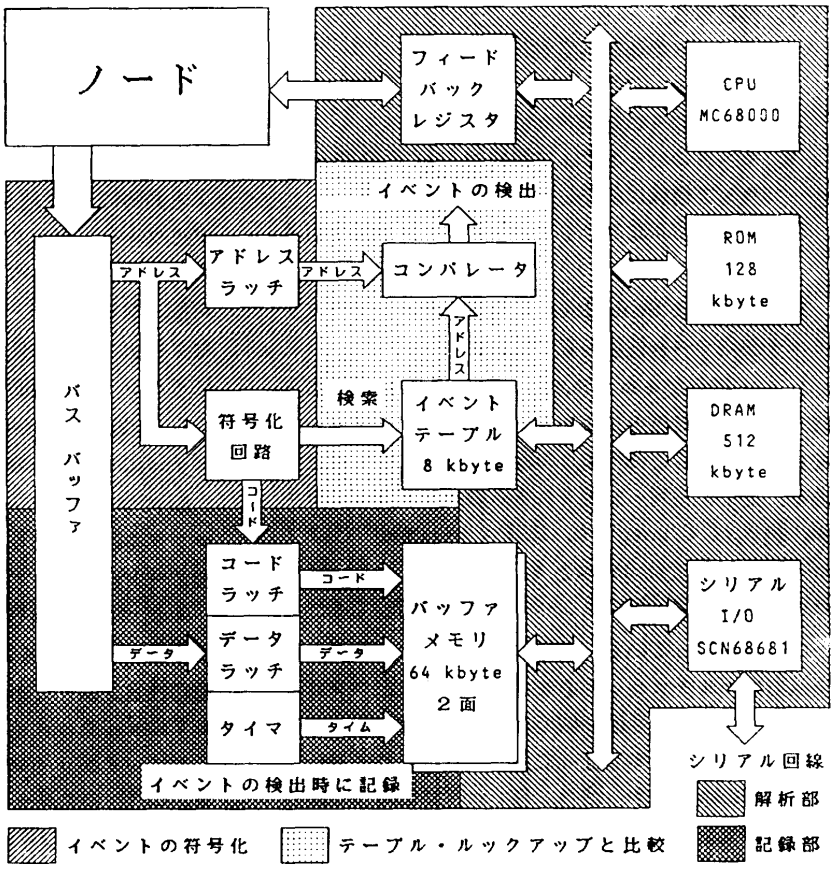


図 2 ハードウェアモニタのブロック図
Fig. 2 Block diagram of the hardware monitor.

果がイベントであれば，そのイベントを記録部でイベントの発生時刻とその他の情報を付加して記録する．

記録部は，バッファメモリを2面持っている．この2面のうち1面が記録に使用される．これと並行してすでに記録された他の1面のデータは解析部で解析される．このように，2面の記録用のメモリを設けることにより，イベントの連続測定が可能となる．

記録部のバッファメモリには，次の三つを一単位としてイベントに関する情報が書き込まれる．

1. イベントに対応するコード
2. イベントを記録した時刻を示すタイムスタンプ
3. イベント発生時のノードのデータバス上のデータ

コードは，符号化回路からの出力が上位8ビット，符号の識別のために下位2ビットの計 10 ビットで構成される．これは，同一符号には4個までイベントの登録が可能なので，それを識別するためである．

バッファメモリ一面の容量は，64 Kbytes である．一方，イベントに関する一単位の情報は，6 bytes なので最大 10,922 個のイベントを一面のバッファメモリに記録することができる．

解析部では，記録部に記録されたイベントを解析したり，イベントテーブルにイベントを登録したり，ソフトウェア開発を行うホスト計算機との通信をシリアル回線を介して行ったりする．

3.1 符号化回路について

本ハードウェアモニタは，広範囲なモニタドメインを持つようにするため符号解析の手法[13]を用いてハードウェアでハッシュを実現し，23 ビットアドレスデータを8ビットに符号化する．そして，イベントテーブルである SRAM に記憶されたイベント（23 ビット）と符号化される前の 23 ビットのデータを比較することによりイベントを検出する．当然ハッシュには衝突が起こる可能性があるが，SRAM のエントリをそれぞれ四つ用意して，3回まで衝突を許すことにより解決している．本方法でも，ソフトウェアのセットアップ時にイベントが3回以上衝突しないようなチェックが必要である．しかし，経験的に 100 個程度のイベントをトレースする場合，ターゲットのイベントのアドレスを変更しなければならないことはなかった．

ここで，符号化回路について説明する．周期 2^4-1 を持つ最大周期列，あるいは M 系列は4段のシフトレジスタで実現することができる．各シフトレジスタ

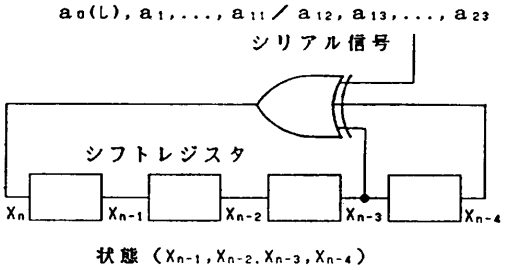


図 3 フィードバックシフトレジスタによるM系列の生成
Fig. 3 A feedback shift register that generates a maximum-length sequence.

の状態を $(X_{n-1}, X_{n-2}, X_{n-3}, X_{n-4})$ で表すとすると，

$$X_n = X_{n-3} \oplus X_{n-4} \quad (1)$$

または，

$$X_n = X_{n-1} \oplus X_{n-4} \quad (2)$$

で与えられる X_n を初段のシフトレジスタに入力すればM系列が得られることが知られている[14]．ここで，⊕は排他的論理和を表す．さて，23 ビットのアドレスの上位 12 ビットと下位 11 ビットのアドレス（A_0 を常にLとしている）のそれぞれをシリアル信号に変換し，図3に示すように，M系列と重ね合わせる．ここでは，上の(1)式を使用している．12単位時間経った後のそれぞれのシフトレジスタの4ビットの状態 $(X_{n-1}, X_{n-2}, X_{n-3}, X_{n-4})$ を組み合わせ8ビットにしたものを最終的なハッシュとする．このように4ビットずつ分けたのは 23 ビットをシリアルに入力して最終的なハッシュを得るには時間がかかりすぎることを考慮した結果による．このように，M系列信号に，逐次信号を重ね合わせシフトレジスタに現れるビットパターンをその信号の符号とすることを符号解析の手法という．

3.2 パイプライン処理

さて，イベントの符号化，テーブル・ルックアップと比較，データの記録は，三つのステージに分けられ，パイプライン処理される．この処理は，図4に示される．本ハードウェアモニタは，メモリ参照をイベントとすると述べたが，メモリのサイクルタイムは，CPU のクロック周波数に比べてあまり速くはない．メモリのサイクルタイムが 500 ns 以上であるノードに対応するように本システムの設計を行い，500 ns ごとにイベントの検出を行うように入力レートを設定した．したがって，各ステージは，500 ns 以内で処理が行われる必要がある．また，技術の進歩によりメモリのサイクルタイムが 100 ns に減少しても，方式自

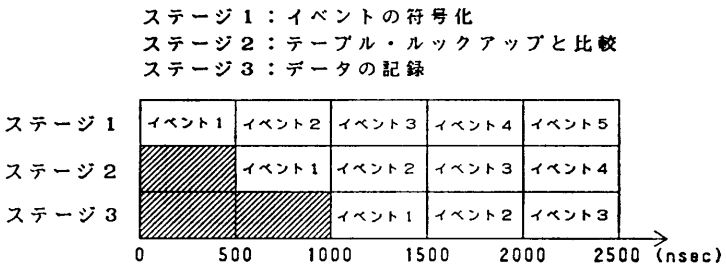


図4 パイプライン処理
Fig. 4 A Gantt chart for a pipeline processing.

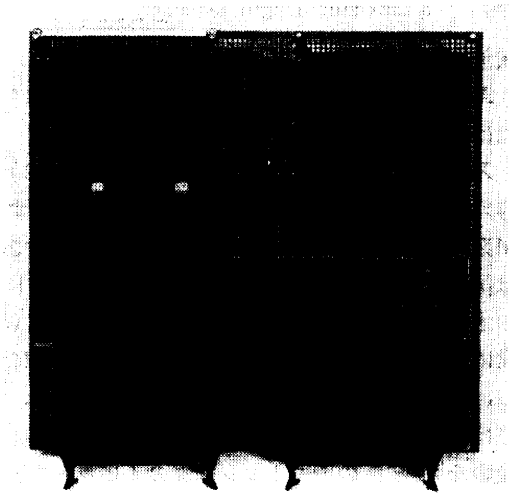


図5 ハードウェアモニタの基板
Fig. 5 A hardware monitor.

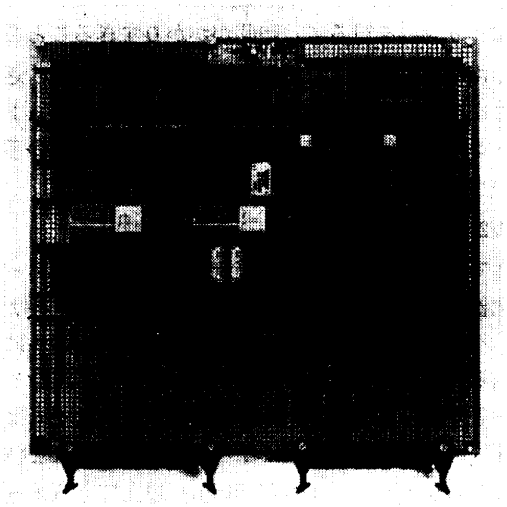


図6 ノードの基板
Fig. 6 A node.

身を変更する必要はなく，高速な素子を採用すれば現在の技術で速いメモリに対応するハードウェアモニタを実現することができる．

実際の実装としては，高速で非同期な VME バス (ANSI/IEEE SDT 1014-1987) 上で分散メモリ型のマルチプロセッサシステムのためのハードウェアモニタを設計し評価を行った．本システムはハードウェアモニタからフィードバックされた情報を用いて，各ノードの状態を把握し，負荷のバランスをとることを目的としている．現在は4枚のノードと，4枚のハードウェアモニタを持つシステムを製作し，分散オペレーティングシステムの評価を行っている．ここで，**図5**に試作した本ハードウェアモニタの基板を，**図6**にノードの基板を示す．なお，ノードとハードウェアモニタはフラットケーブルで接続するようになっている．

4. 実験と考察

ハードウェアモニタとソフトウェアモニタには，それぞれ短所と長所があり，それぞれ目的に合わせて使用すべきであるが，ソフトウェアモニタでは，測定しきれないものもある．そのようなものの中でノード上のオペレーティングシステムのオーバヘッドに深く関係する項目について，ハードウェアモニタを用いて測定を行った．その項目とは，以下のものである．

1. コンテキスト切り替えに関するもの
2. プロセス間通信に関するもの
3. 割り込み駆動方式によるプロセスの応答時間に関するもの

本ハードウェアモニタの時間の精度は，ハードウェアの制約から最小単位を 16 μs に選んでいる．この値は，MC 68000 とよく組み合わされて使用される MC 146818 というリアルタイムクロックが，30 μs から 500,000 μs の周期的なインタラプトをかけることができることを考慮した結果，これと同程度の精度を持つようにした．

これより，上記3項目の測定結果と，それに関するノードのオペレーティングシステムの説明を行う．

4.1 コンテキスト切り替えに関する実験

プロセスは，オペレーティングシステムにおける資源管理の単位であり，スケジューリングの単位である．

プロセスは，生成されると，その処理が終了するまでいくつかの状態を遷移する．プロセスの状態には，

実行状態，実行可能状態，封鎖状態がある．実行可能状態のプロセスは，プロセッサの割り当てなどのスケジューリングの対象であり，レディキューにつながれている．実行状態のプロセスは，レディキューより取り出されて現在実行中のプロセスのことである．封鎖状態のプロセスとは，メッセージの到着待ちなどで，レディキューから外された状態にあり，スケジューリングの対象とならないプロセスのことである．

プロセス間のコンテキストの切り替えは，メッセージによる同期，reschedule() による明示的な実行権の放棄，または，タイマ割り込みによる実行権の放棄により起こる．これら3種類のコンテキスト切り替えに要する時間をハードウェアモニタを用いて測定した結果は，

タイマ割り込みによる場合　281 μs
reschedule() による場合　271 μs
メッセージによる同期の場合　190 μs

である．なお，これらの結果はそれぞれの項目について測定を 1,000 回行い，平均をとるプログラムで実行され得られたものである．

このようなコンテキスト切り替えに関する実験をソフトウェアモニタで行うことは非常に困難である．その理由はソフトウェアモニタは通常ソフトウェア割り込みを用いて実現される場合が多く，ソフトウェア割り込みによるコンテキスト切り替えのためのオーバヘッドが本実験の測定結果と同程度になり実験の意味をなさないからである．

4.2 プロセス間通信に関する実験

ノード内でのプロセス間通信は，メッセージパッシング[15]に基づいている．また，メッセージ通信は，図7のようにポートを介して行われる．つまり，メッセージの送信は，相手のプロセスにではなく，ポートに対して行われる．ポートはメッセージのキューや，メッセージ待ちのプロセスのキューを含むデータ構造である．

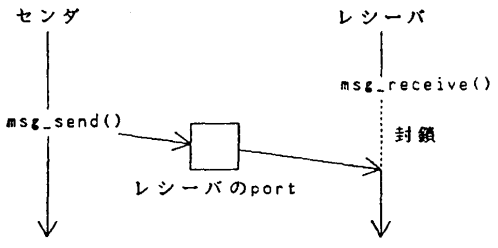


図 7 非同期型の通信
Fig. 7 Asynchronous communication (send and receive).

対象システムのメッセージ送信は，非同期に行われる．非同期型の送信とは，図7のようにメッセージ送信後もセンダのプロセスは封鎖されることなく実行するものをいう．メッセージの送信は，msg_send() により，受信は，msg_receive() により行われる．

ハードウェアモニタを用いて，非同期型の通信についての測定結果を以下に示す．送られるメッセージは1ロングワードである．センダ側は，非同期の通信であるため，タイマ割り込みがあるまで，コンテキストは切り替わらない．msg_send() は，メッセージを受け取るレシーバの状態が実行可能状態か，封鎖状態であるかにより異なり，相手のレシーバが封鎖状態ならば，それを実行可能状態にしなければならないので，余分に時期がかかる．

msg_send() において
レシーバ側が封鎖状態の場合　259 μs
レシーバ側が実行可能状態の場合　132 μs

かかる．

一方，msg_receive() は，ポートが空の場合には，レシーバが封鎖され，余分に時間がかかる．

msg_receive() において
ポートが空の場合　236 μs
ポートが空でない場合　122 μs

という測定結果が得られた．

なお，これらの結果はそれぞれの項目について測定を 1,000 回行い，平均をとるプログラムで実行され得られたものである．

このようなプロセス間通信に関する実験をソフトウェアモニタで行うことは前の実験 4.1 と同様困難である．この理由は高速プロセス間通信を実現している場合にはソフトウェアモニタのオーバヘッド大きすぎるからである．

4.3 割り込み駆動方式によるプロセスの応答時間に関する実験

実験は，周期的な割り込み要求が CPU に対して行われてから，待ち行列または条件変数で封鎖状態にあるプロセスが実行を開始するまでの経過時間を測定するものである．割り込みは，100 ms ごとに周期的にかける．このような割り込み駆動方式によるプロセスは，I/O 関係などのプロセスに用いられている．実験では，次の条件を設けた．まず，スケジューリングの対象となるプロセスを二つに限定する．その中で，待ち行列または条件変数で封鎖されているプロセスは一

つだけとし，最も優先度を高くする．もう一つのプロセスは，入出力要求のようなシステムコールなどを発しないで無限に実行を続ける，いわゆるアイドルプロセスとする．また，前者のプロセスは，実行前には，待ち行列または条件変数で封鎖状態にあり，実行後には，待ち行列または条件変数で自分自身を封鎖する．

割り込み要求が CPU に対して行われてから，割り込み待ちのプロセスが実行状態になるまでに次の二つの期間を経過しなければならない．

(1) 例外処理シーケンスが行われる期間

(2) 封鎖状態が解かれ，スケジューラとディスパッチャが実行される期間

(1)について説明する．まず，割り込み要求が CPU に対して行われる．このとき割り込み要求が CPU に届いても即座に例外処理シーケンスは行われない．コンテキストの切り替えの最中であるならコンテキスト切り替えが終了するまでその割り込みの例外処理シーケンスは延期される．なぜなら，コンテキストの切り替え時には，割り込みが禁止されているからである．

割り込みの例外処理シーケンスは割り込みが禁止されていなければ，最短で 7 μs である．これは，MC 68000 の仕様[16]をもとに，割り込み時の例外処理シーケンスのクロック周期数を 56 として計算した値である．

(1)に関する時間は，割り込みを禁止するプロセスに影響されるが，先に述べた条件のため，割り込みを禁止する期間は，コンテキスト切り替え時のみである．コンテキスト切り替えに要する時間は，4.1 節で示すように 281 μs であるので，(1)の期間の最悪で，7+281=288 μs となる．

(2)について説明する．(1)の後に割り込みハンドラが待ち行列または条件変数上のプロセスの封鎖を解きレディキューにつなぐ．そして，スケジューラがコンテキスト切り替えを実行する．その直後にディスパッチャがそのプロセスを実行させる．先に述べた条件のため，優先度の最も高いレディキューには自分自身しかないので別に測定を行った結果により，(2)の時間は一律に 484 μs であることが知られている．

したがって，割り込み駆動方式によるプロセスが起動されるまでの全体の時間は，(1)の期間と(2)の期間の合計であるので 491 μs から 772 μs の間に分布していることが予想される．

以上のことを考慮して，割り込み駆動方式によるプロセスの応答時間の測定を行った．**図 8** にその結果を示す．図 8 の横軸は，割り込み要求が CPU に対して行われたときに，割り込み待ちのプロセスが封鎖状態から実行状態になるまでの時間を表し，縦軸は，その時間の分布をパーセントで表している．図 8 のグラフは，500 μs の辺りに大きなピークがあり，それ以外は極端に疎らである．割り込み駆動方式によるプロセスの応答時間の最短時間は 496 μs であり，予測値とよく一致する．また，割り込み駆動方式によるプロセスの応答時間の最長時間は 784 μs であり，これも予測値とほぼ一致する．なお，これらの結果は，200,000 回の割り込みに対するデータの収集を行い，平均をとるプログラムで実行され得られたものである．

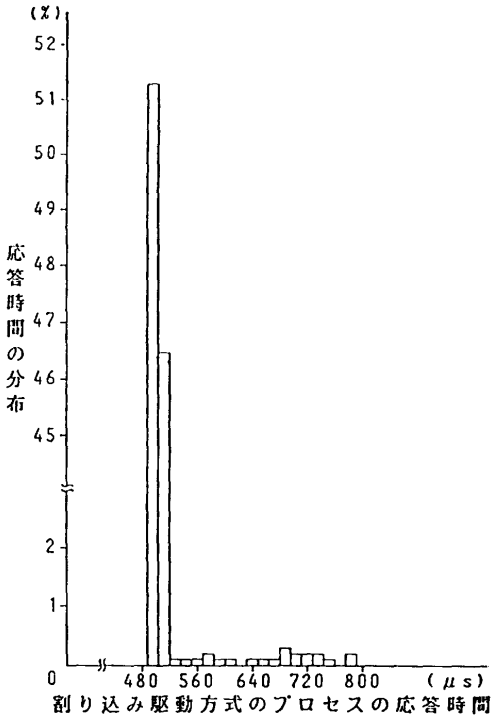


図 8 割り込み駆動方式のプロセスの応答時間に関する実験
Fig. 8 Distribution of response time of an interrupt-driven process.

このような割り込み駆動方式によるプロセスの応答時間の分布を求めることができることが本ハードウェアモニタの特徴の一つである．

5. おわりに

ノードの数がきわめて多い分散メモリ型の並列・分散システムのハードウェアモニタの設計法を提案し，

その実現について述べた．本ハードウェアモニタの特徴はノードと同程度のハードウェア量で構成され，しかもノードの CPU とローカルメモリ間のバスにプローブを挿入するだけで動作可能であることである．

今回，ハードウェアモニタを用いて，VME バス結合による分散メモリ型の並列・分散システムの一つのノードに限って分散オペレーティングシステムのオーバヘッドに深く関わる項目について測定を行い．本ハードウェアモニタの有効性を示した．

本ハードウェアモニタは，多くの分散メモリ型の並列・分散システムで見られるようなノード，つまり，高機能の CPU とローカルメモリがあり，その他にはノード間の通信を行う専用プロセッサだけがあるという比較的簡単な構成をとるものを対象として設計されている．したがって，共有メモリ型の並列・分散システムで通常問題となるキャッシュメモリの一致性，メモリ管理におけるアドレスの変更，CPU の命令の先取りに対する検討はしていないので，これらのシステムには適用できない．共有メモリ型あるいはデータフロー型の並列・分散システムのハードウェアモニタをどのように設計しなければならないのかということは未だ完全には解決されていない今後に残された重要な課題である．

分散メモリ型の並列・分散システムで複数のノードが存在する場合に本ハードウェアモニタを用い，動的負荷分散に利用した結果は別の機会に発表する予定である．

参考文献

1) Cheriton, D. R.: The V Kernel: a Software Base for Distributed Systems, *IEEE Softw.*, Vol. 1, No. 2, pp. 19-42 (1984).
2) Cheriton, D. R. and Zwaenepoel, W.: Distributed Process Groups in the V Kernel, *ACM Trans. Comput. Syst.*, Vol. 3, No. 2, pp. 77-107 (1985).
3) Accetta, M. et al.: Mach: A New Kernel Foundation for UNIX Development, *Proc. of the Summer 1986 USENIX Tech. Conf. and Exhib.*, pp. 93-112 (1986).
4) Simmons, M. et al.: *Instrumentation for Future Parallel Computing Systems*, ACM Press (1989).
5) Seitz, C. L.: The Cosmic Cube, *Comm. ACM*, Vol. 28, No. 1, pp. 22-33 (1985).
6) Dally, W. J.: The J-Machine System, in Wiston, P. H. with Shellard, S. A. (ed.), *Artificial Intelligence at MIT*, Vol. 1, Chapter 21, MIT Press (1990).
7) Wulf, W. A. et al.: *HYDRA/C. mmp.: An Experimental Computer System*, McGraw-Hill (1981).
8) Fromm, H. et al.: Experiences with Performance Measurement and Modeling of a Processor Array, *IEEE Trans. Comput.*, Vol. C-32, No. 1, pp. 15-31 (1983).
9) Svobodova, L.: *Computer Performance Measurement and Evaluation Methods: Analysis and Applications*, American Elsevier Publishing Company (1976).
10) Ferrari, D.: *Computer Systems Performance Evaluation*, Prentice-Hall (1978).
11) Reed, D. A.: Distributed Memory Working Group Summary, in Ref. 4), pp. 239-250.
12) Carpenter, R.: Performance Measurement Instrumentation at NBS, in Ref. 4), pp. 159-184.
13) Frohwerk, R. A.: Signature Analysis: A New Digital Field Service Method, *Hewlett-Packard Journal*, pp. 2-8 (May 1977).
14) Stone, H. S.: *Discrete Mathematical Structures and Their Applications*, p. 335, Science Research Associates Inc. (1973).
15) Gentleman, W. M.: Message Passing between Sequential Processes: the Reply Primitive and the Administrator Concept, *Softw. Pract. Exper.*, Vol. 11, pp. 453-466 (1981).
16) Motorola: M 68000 8-/16-/32-Bit Microprocessors User's Manual, M 68000UM/AD Rev. 5, 6th ed., Prentice-Hall (1989).

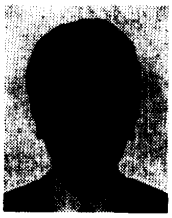
(平成2年9月7日受付)
(平成3年5月7日採録)

白川　洋充（正会員）

昭和15年生．昭和38年大阪大学工学部通信工学科卒業．昭和40年大阪大学大学院工学研究科修士課程修了．昭和43年大阪大学大学院工学研究科博士課程修了．同年電子技術総合研究所に入所．在職中スタンフォード大学人工知能研究所客員研究員．昭和53年宇都宮大学工学部情報工学科助教授．昭和57年より立命館大学理工学部情報工学科教授．オペレーティングシステム，関数型言語の研究に従事．「Lisp プログラミング入門」（オーム社）の著書がある．電子情報通信学会，ソフトウェア科学会，ACM，IEEE 各会員．工学博士．

油谷　　聡（正会員）

昭和40年生．平成元年立命館大学理工学部電気工学科卒業．平成3年同大学院理工学研究科電気工学専攻博士課程前期課程修了．同年ソニー株式会社に入社．在学中並列計算機アーキテクチャとその評価に関する研究に従事．



丹波　　覚（正会員）

昭和37年生．昭和61年立命館大学理工学部電気工学科卒業．昭和63年同大学院理工学研究科電気工学専攻博士課程前期課程修了．同年ダイキン工業株式会社に入社．在学中分散オペレーティングシステムの性能評価に関する研究に従事．電子情報通信学会会員．