細粒度並列処理におけるレイテンシ隠蔽効果の評価

平木 敬[†,††] 島田俊夫[††] 関口智嗣[††]

プロセッサ間通信およびメモリアクセスに伴うレイテンシ問題の克服は，現在並列処理が抱える基本的問題点の一つである．データフロー処理に代表される細粒度並列処理方式を支える特徴として，多重環境によるレイテンシの隠蔽能力は代表的なものである．また，フォンノイマンプロセッサにおいても少数個の環境を並行に処理することによるレイテンシの隠蔽方式が各種提案されてきている．本論文はこのようなレイテンシ隠蔽方式の有効性を命令レベルデータ駆動計算機SIGMA-1を用いて評価した．評価の結果，多重環境はシステムに静的に存在するレイテンシを隠蔽する目的では有効であるが，システム中の通信トラフィックに関連した動的レイテンシの隠蔽能力はほとんど持たないことが明らかとなった．

Evaluation of Latency Hiding on a Fine-grain Parallel Processor

Kei Hiraki[†,††] Toshio Shimada[††] and Satoshi Sekiguchi[††]

Latency associated with memory accesses and process communications are one of the most difficult obstacles in constructing a practical massively parallel system. An instruction-level data-driven computer is an ideal test-bed for evaluating these latency hiding methods because prefetching and multi-threading are naturally implemented in an instruction-level data-driven computer as *unfolding and concurrent execution of multiple contexts*. This paper evalutes latency hiding methods on SIGMA-1, a dataflow supercomputer developed in Electrotechnical Laboratory. As a result of evaluation, these methods are effective to hide static latencies but not effective to hide dynamic latencies.

1. はじめに

プロセッサ製造技術の急速な進歩によりプロセッサの基本クロック周波数が高速化し，プロセッサ性能の著しい向上が実現している．しかしながら，チップ間に跨るプロセッサ間通信およびメモリアクセス速度はプロセッサ内部と比較して相対的に低速に留まり，通信時間の増大が計算機システム，特に並列計算機システムにおける主要な性能低下の要因となっている．

レイテンシは，メモリアクセスまたは通信における要求を発行した命令の実行から，その結果を受理する命令実行までの時間間隔と定義される．メモリアクセスおよび通信に伴うレイテンシが引き起こす性能低下は通信路の高速化という自明な方法に加え，レイテンシの対象の複製をプロセッサにより近接した位置に置くか，レイテンシに相当する期間をレイテンシの対象と独立な計算で補うことにより防止することが可能である．前者はキャッシュメモリ，特に共有キャッシュメモリ方式に代表されるレイテンシ低減方式[1),2)]であり，後者はプリフェッチ，多重スレッド実行に代表されるレイテンシ隠蔽方式[3)~5)]である．並列計算機においては，異なる要素処理装置で実行される計算間の通信や共有メモリアクセス等レイテンシ低減方式の適用が困難である状況が発生するため，メモリアクセスや通信に伴うレイテンシ隠蔽方式の重要性が高い．

一般的に，レイテンシには命令実行からネットワーク出力までの時間遅延，ネットワーク中での時間遅延，メモリアクセスやメモリ上の同期処理に伴う時間遅延，結果のネットワーク遅延，処理装置がネットワークからデータを受理してから実際に使用する命令が実行されるまでの時間遅延が含まれる．これらの操作またはその一部がソフトウェアにより実現される場合には，必要な命令実行時間もレイテンシに加わる．

これらのレイテンシはさらに，システムに静的に存在する静的レイテンシと，動的な要素である動的レイテンシに分けられる[6)]．静的レイテンシは，ネットワークのホップ数に比例する遅延時間，処理装置のパイプライン構成に起因する遅延時間などを含み，シス

† 東京大学理学部情報科学科
Department of Information Science, Faculty of Science, The University of Tokyo
†† 電子技術総合研究所
Electrotechnical Laboratory

テムが非常に空いている状況で現出するレイテンシである．動的レイテンシはネットワーク内での衝突，処理装置内でのキューによる遅延等システムの負荷が重くなった場合に現出するレイテンシである．一般的に動的レイテンシはネットワークの飽和時に見られるように，トラフィックの増大に伴い急速に増大する性質を持つ．

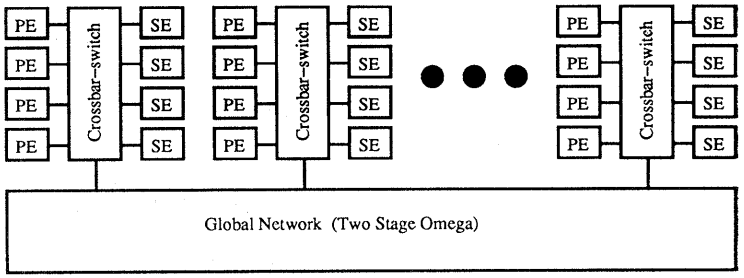

図 1 SIGMA-1 全体構成
Fig. 1 Global architecture of SIGMA-1.

現在逐次計算機を含むフォンノイマン計算機で用いられているレイテンシ隠蔽の方式は，(1)ロード/ストアアーキテクチャによるメモリに対するスプリットロードの採用とプリフェッチ，および，(2)スレッド多重実行が代表的なものである[3]．細粒度データ駆動計算機ではさらに，(3)アンフォールディングによるアクセスの先行実行，(4)命令レベルでの多数スレッドの細粒度多重実行，(5)パネット受理，同期操作と実行のパイプライン的分離により，レイテンシ隠蔽がより効果的に実現可能である．このことから，細粒度データ駆動計算機においては共有データに関して繁雑なコンシステンシ管理問題を避け，直接遠隔メモリをアクセスし，付随するレイテンシを隠蔽する方式が提案されてきた[7]．

レイテンシ隠蔽方式は大規模並列計算機アーキテクチャ上の重要な問題点であるが，実機，特にフォンノイマン型並列計算機による評価はほとんど行われていない．その理由は，レイテンシ隠蔽効果の評価には，深いプリフェッチを可能とするメモリインタフェース，軽いスレッド切替えによる多重スレッド実行が必要であるが，現行のプロセッサアーキテクチャでは実現が非常に困難だからである．本論文では，測定対象として各種レイテンシ隠蔽方式がオーバヘッドなしで実現可能である細粒度データ駆動計算機を用い，レイテンシ隠蔽方式に対する基本的評価を行う．

2. SIGMA-1 におけるレイテンシ

本論文の測定対象である SIGMA-1 は，電子技術総合研究所において研究・開発が行われた命令レベルデータ駆動計算機である．SIGMA-1 は全命令が入力データのマッチング操作で発火する純命令レベルデータ駆動方式であることを特徴とする．アーキテクチャの詳細は文献8),9)に報告されている．SIGMA-1 の全体アーキテクチャは**図1**に示すように，128 台の演算処理装置 (PE) と 128 台の構造体処理装置 (SE) で構成される．4台の PE と4台の SE は1クラスタを構成し，全体で 32 のクラスタからなる．

ここではレイテンシ隠蔽方式と関連の深い特徴をのべる．SIGMA-1 の基本設計は，静的なレイテンシ要素を小さくすることを重要な目的の一つとしている．具体的には，短い2段のパイプライン構成を持つ処理装置，パケット通信と実行の重複処理，クロスバースイッチとオメガ網で構成されるクラスタ化された全体構成がレイテンシの短縮に貢献している．この構成により，クラスタ内メモリへの読み出しアクセスに要する静的レイテンシが8クロック，クラスタ間メモリ静的レイテンシが14クロックである (**表1**)．なお，2入力命令は3クロック以上の時間間隔で処理されるため，ローカルアクセスに約3命令時間，グローバルアクセスに約5命令時間が費やされる．命令の実行順序はすべてデータの到着順序で決定される．したがって2入力命令の入力パケット順序によるパイプラインバブル発生は動的要素により決定され，同一プロ

表 1 SIGMA-1 におけるメモリアクセスレイテンシ
Table 1 Memory access latency in SIGMA-1.

Item	Latency (clocks)
Execution stage to network input	1
Network input to global network input	1
Global network input to global network output	(2)
Global network output to the SE input	(1)
SE input to SE execution	1
SE service latency	1
SE output to network input	1
Global network input to global network output	(2)
Global network output to the PE input	(1)
PE input to PE execution stage	3
Total	(14)8

グラムを再実行した場合，ネットワークのアービトレーション時の優先度など影響を受け，実行時間は常に変化する．SIGMA-1 処理装置は1台あたりピーク性能で5 MIPS であるが，入力パケット順序によるパイプラインバブルおよび多出力命令における出力待ちの結果，平均性能で約 2.5 MIPS の性能を持つ．

次に，SIGMA-1 におけるレイテンシ隠蔽の実現方式を説明する．すでに述べたようにレイテンシ隠蔽はデータを利用・消費するプログラム部分ができるだけ早い時期にデータを要求し，データを生産・送致するプログラム部分からできるだけ早い時期に要求されたデータを返送することにより実現される．

このことを実現するため，データを利用するプログラム部分（以下，データフロー計算では適切ではないが直接的なデータ依存関係で結ばれた一連の命令群をスレッドと呼ぶ）においてはデータに関する逆依存関係を満たす限り早い時期にデータ読み出し要求を出し，データを利用するまでの間に同一スレッドに属し直接データ依存関係のないデータを操作する命令を挿入する（プリフェッチ）．また，異なるスレッド間では，データ要求の発生時とデータを利用する間に異なるスレッドに属する命令実行を挿入することにより実現される（多重スレッド実行）．

プリフェッチおよび多重スレッド実行を効率的に実現するためには，要素処理装置が(1)データ依存および逆依存関係を解決する同期機能を持った局所記憶，(2)オーバヘッドが命令実行時間と比較して小さい高速コンテクスト切替え能力，(3)メモリアクセス要求とメモリアクセス結果利用が分離したスプリットアクセス機能，および(4)データが利用可能となったスレッドから順次実行が可能な柔軟な動的スケジューリング能力を持つことが必要である．SIGMA-1 はダイナミックアーキテクチャに基づく命令レベルデータ駆動計算機であるため，レイテンシ隠蔽に必要なこれらの要求は自然に実現される．

SIGMA-1 におけるプリフェッチは，ループのアンフォールディングにより，同一関数内部におけるメモリアクセス要求と利用のループを分離し，タグの一部であるループインデクスフィールドの異なるメモリ要求を多重に発行することにより実現される．プリフェッチの深さはタグ幅（10 ビット）の許す限り深くすることが可能である．一方，多重スレッド実行は各データに附属するタグのループインデクスフィールド以外の部分である関数識別子の異なる関数を同時に実行することによりオーバヘッドなしで実現される．ただし，実行命令のスケジューリングは常に First In First Out 順序で動的に決定されるため，通常の状態はプリフェッチと多重スレッド実行が混合した状態である．したがって，両者を分離した測定の実現には，命令列内に本来は必要でないスケジューリングのための同期命令を挿入する必要があり，測定上のオーバヘッド要素となっている．

測定を行ったプログラムはすべて DFC 言語[11]により記述され，DFC 最適化コンパイラにより翻訳して実行した（プログラムは関数単位でクラスタに割り付けられ，クラスタ内は乱数的な分散を行った．時間はロジックアナライザで計数し，実行時間の分散が大きいプログラムについては多数回の実行結果の算術平均を使用した．

3. レイテンシ隠蔽効果の評価

第一の性能評価は，ウェーブフロントアルゴリズムを用いた SOR プログラムを対象とした．本プログラムでは，格子状に配置された節点の値を，上方および左方からのすでに更新されている値，当該節点，右方および下方からの更新前の値を使用して求める．したがって並列度は行列の対角線方向に存在し，隣接する節点から送られる更新された値により評価が開始される．被測定プログラムでは，行列から1列ごとに切り出し，処理装置を割り当てる．格子間の通信は構造体の要素毎に同期タグを持つB構造体を使用し，2個の構造体を交互に使用して演算を行った．したがって，行列の1辺の大きさと同数の関数が並列実行を行っている．また，構造体はシステム中に均等となるような静的分割を行い，プログラムの処理装置へのマッピングは SIGMA-1 の持つ動的負荷分散機構を使用した．

測定は(1)自然にアンフォールディングを行うことにより，未計算の節点値についても無制限に先行して読み出しを行い，構造体上で同期待ちをする場合，(2)人為的にアンフォールディングを阻止し，ある節点の実行が終了した時点で隣接する節点の実行を開始する場合，(3)節点1個分だけ先読みを行う場合について行った．

フォンノイマン計算機は先行読み出しおよび多重スレッド実行を行わない場合が基本であり，その場合に最も命令オーバヘッドの小さいオブジェクトプログラムが得られ，先行読み出しおよび多重スレッド実行は命令オーバヘッドを伴って実現されることに対し，

SIGMA-1では無制限先行読み出しを多重スレッド実行することに対応する場合が基本であり，最も命令オーバヘッドの小さいオブジェクトプログラムが得られ，先行読み出しの制限および多重スレッド数の制限には命令オーバヘッドを伴う．すなわち，先読みを行わないか，制限する場合には，アンフォールディングに伴う命令実行は不要であるが，無限先行読み出しの場合と比較して同期命令による命令オーバヘッドを伴う効率の悪いコードを使用する．命令オーバヘッド量は，先行読み出しを行わない場合には，ループ内のメモリ変数数（n）に1を加えたものであり，N個の先行読み出しを行う場合には，$n+(N+1)$である．

図2は，プログラムを1台およびクラスタ内4台のプロセッサで実行し，使用する構造体をクラス内に取る場合の速度をプロットしたものである．図中，横軸は行列の1辺のサイズであり，計算量はその自乗に比例する．構造体をクラスタ外に置く場合でもほとんど重なる同一結果が得られるが，紙面の都合で割愛する．図2では，並列性がプロセッサ台数と比較して十分に大きい問題において，先読みの有無にかかわらずほぼ完全なレイテンシ隠蔽が行われていることが示されている．図2で注目されることは，無限先行読み出しを行う場合において，行列の1辺の大きさの増大に従って実行速度が低下することである．この現象は特に1プロセッサで実行する場合に顕著である．**図3**は1台と4台の速度向上比を行列の1辺の大きさに対してプロットしたものである．無限先行読み出しの場合に台数効果が4を越える原因は，無制限な先行読み出しの結果構造体における同期待ちのデータに比例した個数のデータがマッチングユニット内部に滞留し，マッチングメモリを構成するハッシュ表の性能を衝突により低下させるためである．例えば問題サイズが120の場合でマッチングユニットの利用率を測定すると0.8程度となる．

1ないしは4台の処理装置で実行する場合には，処理の並列度が十分にあるためレイテンシ隠蔽は多重処理により実現されている．したがって，先行読み出しを導入することによりさらに得られるレイテンシ隠蔽の効果はほとんどなく，むしろ処理装置におけるワーキングセットサイズの拡大を引き起こし性能の低下を起こしていると考えられる．図3に示されるように，先行読み出しを0ないし1個に制限することにより，この問題を避けることが可能である．**図4**は実行速度を先行読み出しの個数に対してプロセットしたもので

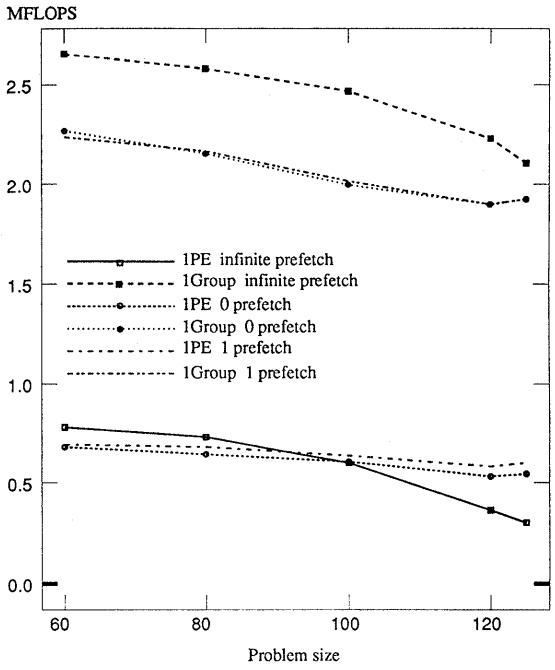

図2 ウェーブフロント問題：行列の1辺の大きさに対する実行速度
Fig. 2 Performance of the wave-front problem on various problem size.

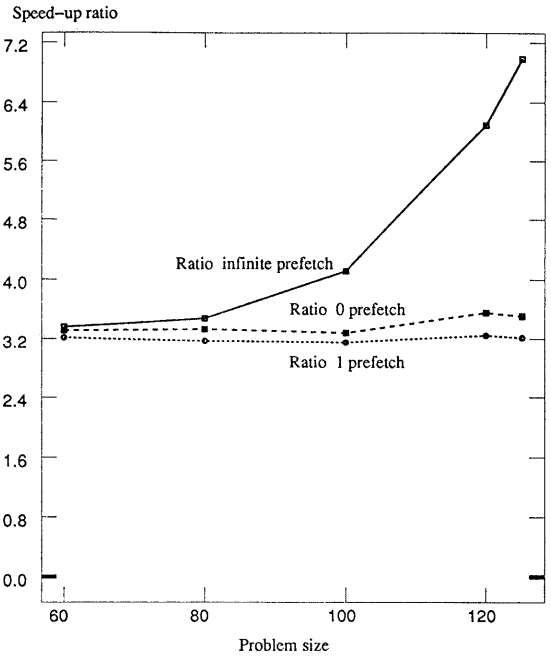

図3 ウェーブフロント問題：先読み方式による実行効率の変化
Fig. 3 Performance of the wave-front problem on various prefetching depth.

あり，わずかではあるが，1個の場合の速度向上が認められ，無限個の場合には速度が低下する．1個の場合の速度向上が認められる原因は，ウェーブフロント問題においては行列の両端で並列性がほとんどなくなる部分があり，先読みを行わないと処理装置にパイプラインバブルが発生するためと考えられる．この図は，プリフェッチの深さを増大させると要素処理装置内部で保持することが必要であるデータセットの大きさが増大し効率が低下するため，必要かつ十分である節度を持った先行読み出しが必要であることを示している．

図5は同一プログラムの実行速度をプロセッサ数を変化させて測定した結果である．処理装置の割り付けは自動負荷分散機構により構造体配置と独立に行われるため，ほとんどのメモリアクセスがグローバルアクセスとなる．その結果，ネットワークにおける衝突が頻繁に発生し動的レイテンシが増加する．図5の結果によると，飽和が顕著でない領域では無限先行読み出しによる速度向上が見られる．しかしながらバンド幅が飽和する領域に達すると急速な性能の頭打ちとそれに引き続く性能低下が発生する．先行読み出しを行わない場合には，アンフォールディングを止めるためのオーバヘッドにより非飽和領域における性能低下が見られるが，飽和領域においても穏やかな振舞いを示し，最終的にはより高い性能が得られる．この現象の原因は，先行読み出しによるレイテンシ隠蔽方式ではネットワークを集中的に使用する結果，飽和領域における衝突がより深刻なものになるためと考えられる．この現象は，文献10)における出力バッファの効果と類似である．

ウェーブフロント問題の評価を通じて，プリフェッチと多重フレッドによるレイテンシの隠蔽は互いに関係していること，レイテンシ隠蔽方式がかえって飽和による性能の悪化をもたらすことを示唆した．しかしながら，ウェーブフロントプログラムの性質上プリフェッチ要素と多重スレッド要素を独立に制御することが困難であり，より詳細な評価を別途行う必要がある．この状況を踏まえレイテンシ隠蔽に伴う問題点をより詳細に評価する目的で，内積プログラムの評価を行った．被評価プログラムとして内積を選択した理由は，メモリアクセス状況，先行読み出し，処理の多重度およびプロセッサ配置を自由にコントロールできるからである．

図6，7，8は先行読み出しが0，1，および無制

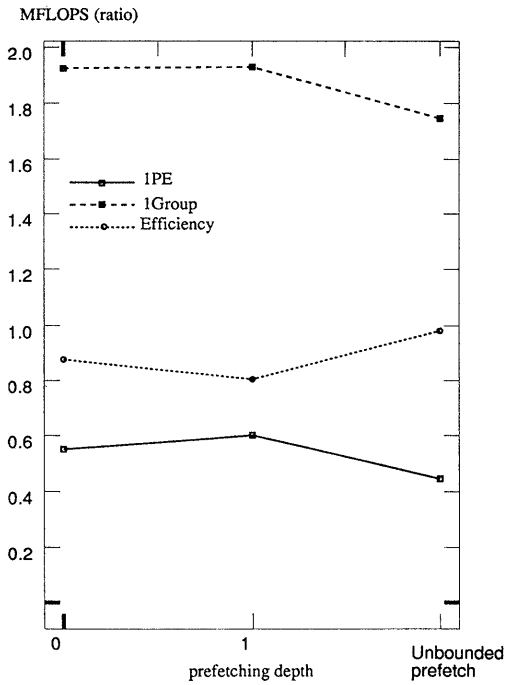

図4 ウェーブフロント問題：先読み方式に対する実行速度
Fig. 4 Performance of the wave-front problem on various prefetching depth.

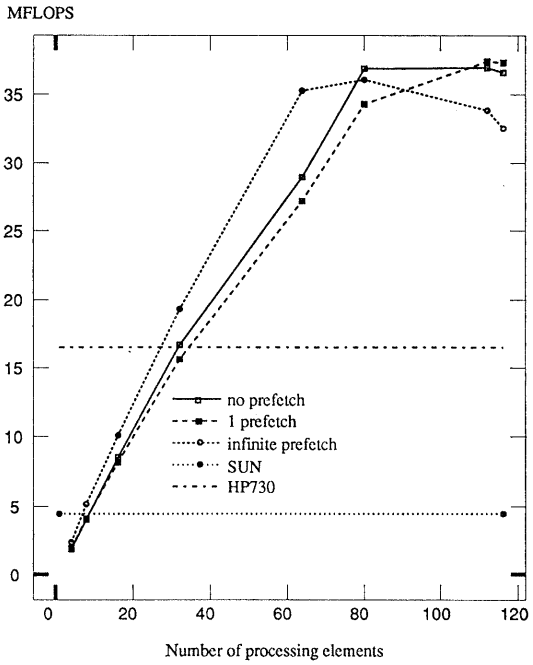

図5 ウェーブフロント問題：台数効果（実行速度）
Fig. 5 Performance of the wave-front problem on various number of processors.

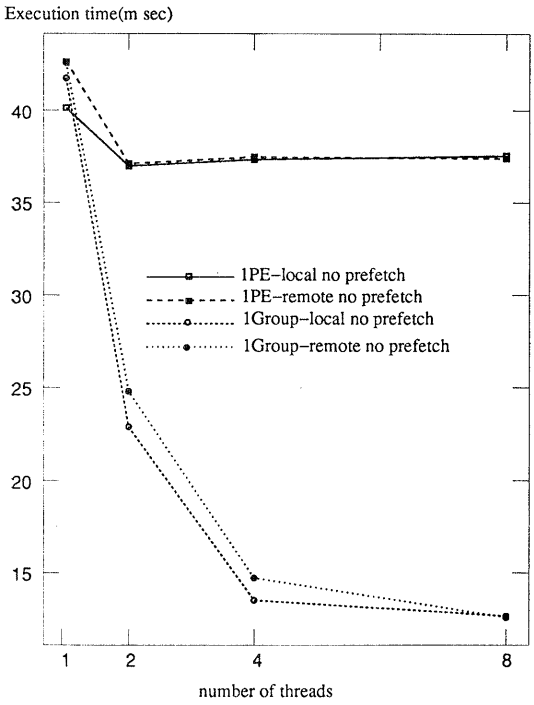

図 6 内積問題：先行読み出しなし (1, 4 PE)
Fig. 6 Inner products: without prefetching (1, 4 PEs).

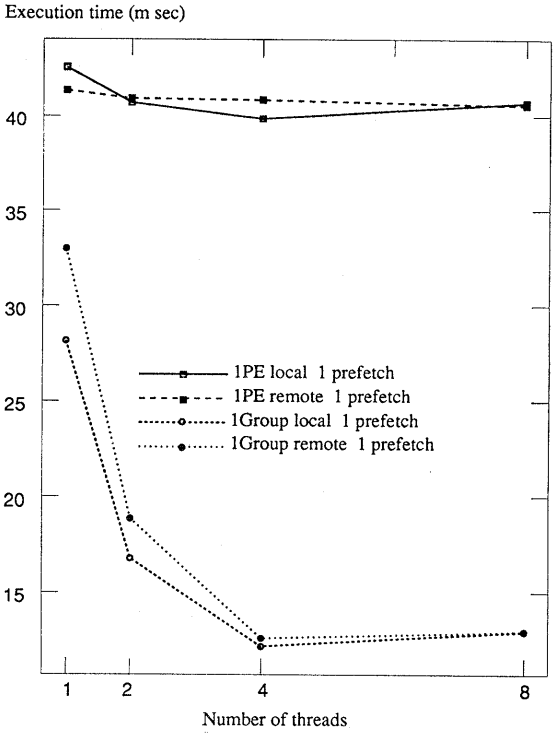

図 8 内積問題：無制限の先行読み出し (1, 4 PE)
Fig. 8 Inner products: Infinite prefetching (1, 4 PEs).

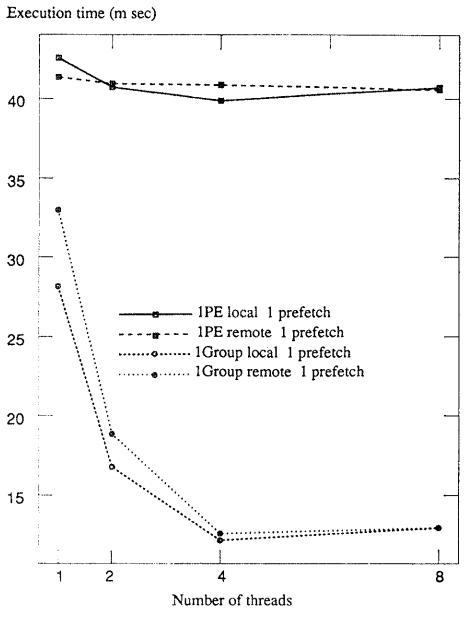

図 7 内積問題：1個の先行読み出し (1, 4 PE)
Fig. 7 Inner products: single prefetching (1, 4 PEs).

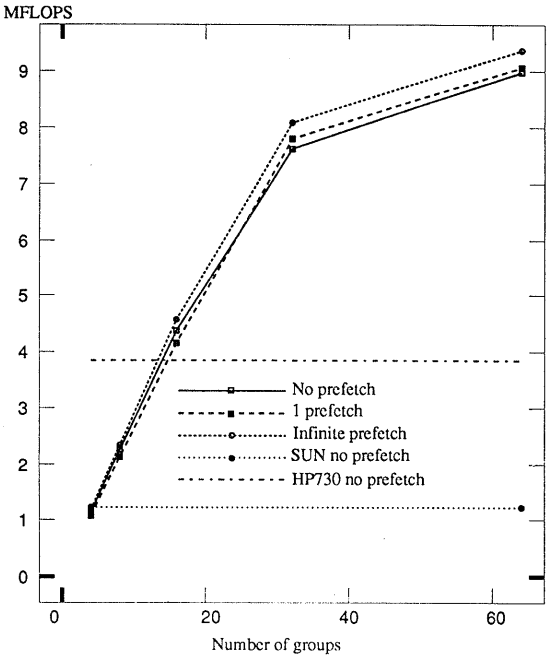

図 9 内積問題：台数効果（種々の先行読み出し方式）
Fig. 9 Performance of Inner products on various number of processors.

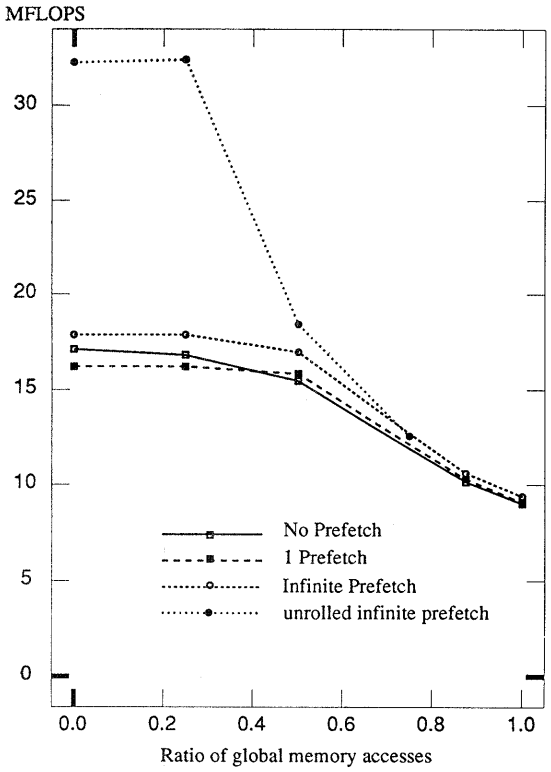

図 10 内積問題：大域メモリアクセス率による実行速度の変化 (64 PE)

Fig. 10 Performance of Inner products on various global accessing ratios.

限の場合における1台および4台のプロセッサの内積実行時間を，メモリがローカルな場合とグローバルな場合について並列実行するスレッド数を1から8まで変化させてプロットしたものである．なお，内積プログラムは本質的に逐次処理であるためスレッド数は並行に起動された関数数に対応する．また，1本のスレッドには並列性がほとんどないため，1台での実行に対し，4台での実行ではメモリ参照以外の命令が1命令ごとにプロセッサ境界を跨ぐため，メモリ参照以外のレイテンシ要素が大きくなるものである．

図6は先行読み出しなしの場合である．多重度が1の場合には処理装置1台においてメモリがグローバルである場合の性能低下が示されている．また，処理装置1台の場合には多重度2以上の場合に完全なレイテンシの隠蔽が行われている．これはグローバルメモリにおいてもレイテンシが約5命令相当と小さいためと考えられる．一方，4台の処理装置の場合には，メモリアクセスに関するレイテンシが総レイテンシに占める比率が低下するため多重度1におけるグローバルメモリでの性能低下が，プロセッサ1台の場合と比較して小さい．しかしながら，大きい総レイテンシによる性能低下は発生し，多重度が4までは顕著な性能向上がみられる．図7は先行読み出し1個，図8は無制限な先行読み出して行う場合の測定結果である．1台の処理装置を用いた場合には高々1個の先行読み出しによりレイテンシ隠蔽が実現されること，先行読み出しによる並列性の導入には限界があり，多重度をあげないと4台の処理装置を満たす並列性が得られないことが示されている．

図9は，内積の演算速度をプロセッサ数に対してプロットしたものである．ここではすべてのメモリアクセスがグローバルアクセスである．図より，飽和領域以前では先行読み出しの個数にかかわらずリニアな性能向上が見られるが，処理装置台数が 64 台に近づくと急速に飽和がみられることが示されている．この現象は先の例と同様ネットワークの飽和によるレイテンシの増大が原因と考えられ，この動的レイテンシ要素は多重処理または先行読み出し方式では解決しえないことが示されている．

図10は処理装置 64 台の場合について，全メモリアクセスに対するグローバルアクセスの比率を変化させた場合の性能を示す．図中最上部の破線は，ループを4回アンロールしたプログラムの性能であり，ループに関するオーバヘッドが減少することにより，ネットワークに対するメモリアクセス頻度が高くなっている．グローバルメモリアクセス比が小さく，メモリが飽和しない状態では，プログラムにより固有の速度を示すが，一旦飽和領域に達すると，性能自身がネットワークで規定されるため急速に性能差が小さくなることが示されている．また，ループをアンロールしたプログラムの場合にはグローバルアクセス率が 0.75 以上の領域では要素処理装置の入力バッファがオーバフローするために正常な実行が不可能であり，性能測定を行っていない．この結果も，レイテンシ隠蔽方式が過度に及ぶとかえって有害であることを示している．

4. おわりに

被測定計算機としてデータ駆動計算機を使用した理由は，すでに述べたように現在利用可能なフォンノイマン型要素処理装置がレイテンシ隠蔽方式の実装に必要な特性を備えておらず，またシミュレーションによっても，プリフェッチや多重スレッド実行に関してさまざまな条件を現出させることが困難であるためで

ある．しかしながら，下記の理由により評価結果はデータ駆動計算機だけでなく，フォンノイマン型要素処理装置を用いた並列システムにも適用可能であると考えられる．

1. 評価において，プリフェッチによる効果と多重スレッド実行による効果を分離して評価するため，関数内部処理ループ単位で逐次性をもった例題を使用した．したがって，4台の要素処理装置を用いたクラスタ単位の実行でも，深いパイプラインを持った計算機と同様の振舞を示すだけであり，実行時における動的スケジューリングの影響を排除してあること．
2. データ駆動アーキテクチャの特徴であるマッチングメモリ機能は，プリフェッチしたデータの到着同期を実現するために使用しているため，フォンノイマン型要素処理装置における同期ビットを持ったキャッシュと等価な機能をもつこと．
3. 要素処理装置と相互結合網が持つ静的レイテンシ量は実現テクノロジで決定されるものであり，アーキテクチャの種別とは独立であること．

ただし，データ駆動計算機における場合と同様のレイテンシ隠蔽効果を得るためにはフォンノイマン要素計算機も(1)多重スレッド実行を小さい命令オーバヘッドで実現可能な同期機構とコンテクストスイッチ機構，(2)キャッシュメモリにおける同期機構，(3)キャッシュメモリに対する外部からのデータ注入機構，(4)深いプリフェッチを命令オーバヘッドなしで実現する命令構成とアクセス機構が備わっていることが望ましい．

本論文ではレイテンシ隠蔽効果について，2種類の例題により評価を行った．評価の結果，レイテンシが静的要因に基づく場合には，先行読み出し，多重処理の双方の方式とも十分な効果を上げることが判明した．しかしながら，ネットワークの衝突など動的な要因によるレイテンシ隠蔽には限界があり，多重環境の並行処理によるレイテンシ隠蔽という手法の限界が示されていると考えられる．また，レイテンシ隠蔽方式の過度の使用は要素処理装置の各部分に蓄積するデータ量の増大を招き，性能の低下や資源の枯渇を引き起こすことが示された．この両者の事実より，レイテンシ隠蔽は静的要因を隠蔽する程度に節度を持って行うことが望ましいと考えられる．

今回の評価は並列計算機の要素処理装置におけるレイテンシ隠蔽方式に限定したものであった．しかしながら，レイテンシ隠蔽方式は要素処理装置間を結合する相互結合網の構成にも大きな影響を持っている．すなわち，レイテンシ隠蔽方式に安定な動作をさせるためにはあらかじめ見積もれる静的レイテンシを持ち，静的レイテンシがレイテンシの主要部分を占める領域の広い相互結合網を使用することが必要である．適切なレイテンシ隠蔽やレイテンシ低減方式の存在を前提とした相互結合網構成に関する評価は今後の課題である．

参考文献

1) Archbold, J. and Baer, J.-L.: Cache Coherence Protocols: Evaluation Using a Multiprocessor Simulation Model, *ACM Trans. Computer Systems*, Vol. 4, No. 4, pp. 273-298 (1986).
2) Sweazey, P. and Smith, A. J.: A Class of Compatible Cache Consistency Protocols and Their Support by IEEE Futurebus, *Proc. 13 th Int. Symp. on Computer Architecture*, pp. 414-423 (1986).
3) Weber, W.-D. and Gupta, A.: Exploring the Benefits of Multiple Hardware Contexts in a Multiprocessor Architecture: Preliminary Results, *Proc. 16th Int. Symp. on Computer Architecture*, pp. 273-280 (1989).
4) Gupta, A., Hennessy, J., Gharachorloo, K., Mowry, T. and Weber, W.-D.: Comparative Evaluation of Latency Reducing and Tolerating Techniques, *Proc. 18th Int. Symp. on Computer Architecture*, pp. 254-263 (1991).
5) Boothe, B. and Ranade, A.: Improved Multithreading Techniques for Hiding Communication Latency in Multiprocessors, *Proc. 19 th Int. Symp. Computer Architecture*, pp. 214-223 (1992).
6) Hiraki, K., Shimada, T. and Sekiguchi, S.: Empirical Study of Latency Hiding on a Fine-grain Parallel Processor, *Proc. 1993 Int. Conf. Supercomputing*, pp. 220-228 (1993).
7) Iannucci, R. A.: Toward a Dataflow/von Neumann Hybrid Architecture, *Proc. Int. Symp. Computer Architecture*, pp. 131-140 (1988).
8) Hiraki, K., Shimada, T. and Nishida, K.: A Hardware Design of the SIGMA-1—A Data Flow Computer for Scientific Computations, *Proc. Int. Conf. Parallel Processing*, IEEE, pp. 851-855 (1984).
9) 島田俊夫，平木　敬，関口智嗣：データフロー計算機 SIGMA-1 の性能評価，JSPP '92 論文集，pp. 345-352 (1992).
10) 児玉祐悦，甲村康人，佐藤三久，坂井修一，山口善教：高並列向け要素プロセッサ EMC-Y の設計，JSPP '92 論文集，pp. 329-336 (1992).
11) 島田俊夫，関口智嗣，平木　敬：データフロー言語 DFC の設計と実現，電子情報通信学会論文誌，Vol. J 71-D, pp. 501-508 (1987).

(平成5年9月27日受付)
(平成6年1月13日採録)

平木　敬（正会員）

昭和51年東京大学理学部物理学科卒業．昭和57年同大学院理学系研究科博士課程修了．同年電子技術総合研究所入所．理学博士．計算機アーキテクチャ全般，特にリスト処理計算機，データフローマシン，スケジューリングなどの研究に従事．元岡賞，市村賞各受賞．情報アーキテクチャ部計算機方式研究室主任研究官を経て，平成3年から東京大学理学部情報科学科助教授．昭和63年から平成2年までIBMワトソン研究センター招聘研究員．

島田　俊夫（正会員）

昭和43年東京大学工学部計数工学科卒業．昭和45年東京大学大学院修士課程修了．同年電子技術総合研究所入所．情報アーキテクチャ部計算機方式研究室室長を経て平成5年より名古屋大学工学部電子情報学科教授．人工知能向き言語，LISPマシン，データフロー計算機の研究に従事．計算機アーキテクチャ，特に並列処理の研究に興味がある．昭和63年度市村賞受賞．工学博士．

関口　智嗣（正会員）

1959年生．1982年東京大学理学部情報科学科卒業．1984年筑波大学大学院修士課程理工学研究科修了．同年電子技術総合研究所入所．現在，情報アーキテクチャ部計算機方式研究室主任研究官．計算機アーキテクチャと数値解析の研究に従事．特に科学技術計算用並列アルゴリズムに興味を持つ．市村賞受賞．現在，データ駆動計算機とスーパコンピュータ評価技術の研究を行っている．日本応用数理学会，日本ソフトウェア科学会各会員．