並列計算機 Cenju-4 の分散共有メモリ機構

細見岳生[†] 加納健[†] 中村真章[†]
広瀬哲也[†] 中田登志之[†]

Cenju-4は，最大1024ノードまでスケーラブルであることを目的として設計された，分散共有メモリをサポートした並列計算機である．スケーラビリティのために，Cenju-4はビットパターン構成のディレクトリを採用している．この方式は，必要なハードウェア量がスケーラブルであることに加え，他の不精密なディレクトリ方式より精密な管理が行えるという特徴を有する．また，無効化要求のマルチキャストと応答の回収をネットワークで行い，全ノードで共有しているデータの書き込みに要するレイテンシをスケーラブルなものとしている．スケーラビリティに加え，Cenju-4は，メモリアクセスが有限時間内に完了することを保証するために，あるタイプのメッセージをメモリに待避してスタベイションとデッドロックを防止している．これにより，集中ディレクトリ方式でスタベイションの防止，1つのネットワークでデッドロックの防止が可能となっている．本稿では，並列計算機 Cenju-4 における分散共有メモリの実装方式を示す．また，実機で性能測定を行った結果も示す．

A DSM Architecture for a Parallel Computer Cenju-4

Takeo Hosomi,[†] Yasushi Kanoh,[†] Masaaki Nakamura,[†]
Tetsuya Hirose[†] and Toshiyuki Nakata[†]

A parallel computer Cenju-4 is a cache-coherent non-uniform memory access (ccNUMA) multiprocessor and designed to be scalable up to 1024 nodes. For scalability, Cenju-4 adopts a bit-pattern directory. This scheme enables more precise representation than other imprecise schemes, such as a coarse vector scheme. Cenju-4 utilizes multicast and gathering functions of the network for delivering invalidation request messages and for collecting replies. This enables store access latency to be scalable, even when the block is shared among all nodes. Cenju-4 also prevents starvation and deadlock by queuing certain types of messages in the main memory. This enables a full solution to the starvation problem with centralized directory scheme, and to the deadlock problem with one physical or virtual network. In this paper, we present the design of the DSM architecture and some performance results.

1. はじめに

分散共有メモリ（以降，DSM）システムの設計において，性能およびハードウェアコストがスケーラブルであることは重要な要求である．また，メモリアクセスの前進性，すなわちメモリアクセスが有限時間内に完了することを保証することも同時に重要な要求である．

Cenju-4は，ハードウェアがディレクトリ方式の無効化型プロトコルでキャッシュのコヒーレンス制御を行うことによりDSMを実現している．ディレクトリ方式では，システムを構成するノード数にかかわらずディレクトリに必要なメモリのサイズがノードあたり一定であることが，ハードウェアコストのスケーラビリティの点で重要となる．従来提案されてきたLimitLESS[1)]やDynamic Pointer[2)]方式は，一定サイズのメモリ容量で実現できるものである．しかし，共有しているノードを特定するのにその数に比例した時間を要するため，この処理がボトルネックとなりメモリアクセス性能を悪化させる恐れがあった．Cenju-4では，ハードウェアコストがスケーラブルであり，かつディレクトリアクセスのコストが一定な方式を採用してこの問題を解決している．

また，メモリアクセスのレイテンシがノード数に比例して悪化することがないことは，性能のスケーラビリティの面で重要である．従来多くのシステムは，書き込みの処理時にデータを共有しているすべてのノードに1つずつ無効化要求を送信し，それらすべてのノードからの応答を1つずつ回収していた．そのため，

† NEC C&C メディア研究所
C&C Media Research Laboratories, NEC Corporation

無効化要求の送信および応答の回収に共有しているノード数に比例した時間を要していた．Cenju-4 ではメモリオーダリングモデルにストロングモデルを採用しているため，とりわけストアアクセスのレイテンシがノード数の増加に比例して悪化しないことは重要な課題となる．Cenju-4 では，無効化要求をネットワークでマルチキャストし，またそれに対する応答をネットワークでギャザーして 1 つのメッセージとすることでこの問題を解決している．

メモリアクセスの前進性を保証するためには，コヒーレンス制御によりスタベイションおよびデッドロックが発生しないことを保証する必要がある．従来 DASH[3]や Alewife[4]で採用されているコヒーレンスプロトコルは，あるメモリアクセス要求に関してその時点で一貫性を維持することができない状態に陥った場合，要求発行ノードに否定応答を返し再試行させていた．そのため，メモリアクセス要求が否定応答され続ける，すなわちスタベイションが発生する可能性があった．Cenju-4 は，特定のメッセージを一時メモリに待避し後で順に処理を行うキューイング方式を採用して否定応答を排除し，スタベイションを防止している．

また，従来 DASH では，要求と応答用に独立した 2 つのネットワークを持つことでデッドロックの問題に対応していた．しかし，これだけではデッドロックを防止することができないため，ある要求によってデッドロックが発生する可能性がある場合，その要求の処理を進めず要求発行ノードに対して否定応答を返してデッドロックを回避していた．しかし，これはデッドロックの問題をスタベイションの問題に転嫁しているだけであり，メモリアクセスの前進性を保証するという点では問題の解決にならない．Cenju-4 は，ハードウェア制御で特定の種類のメッセージを選択してメモリに待避することにより，デッドロックの問題を解決している．この方式は，多重化されていないネットワークでも実現可能なものである．

以降，2 章では，Cenju-4 についてその構成を簡単に説明する．3 章では，Cenju-4 の分散共有メモリ機構について説明する．4 章では，基本的なアクセス性能と，アプリケーション（NAS Parallel Benchmarks V2.3）を用いて評価した結果を示す．

2. Cenju-4 の概要

図 1 に Cenju-4 の構成概要を示す．Cenju-4 の各ノードはプロセッサ R10000，1 M バイトの 2 次キャッシュ，512 M バイトまで拡張可能なメモリ，PCI バス，

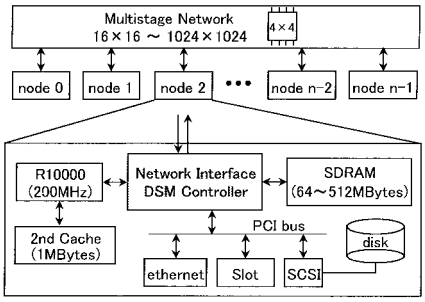


図 1 並列計算機 Cenju-4 の構成概要
Fig. 1 Cenju-4 architecture overview.

それらを制御するチップで構成される．この制御チップはネットワークとも接続し，メッセージ通信や DSM を実現している．ネットワークは最大 1024 ノードまで接続可能な多段網であり，4 入力 4 出力のクロスバスイッチで構成される．このネットワークの特徴を以下に記す．

(a) 2 ノード間のメッセージ到着順序の保証
(b) デッドロック・フリー
(c) マルチキャスト/ギャザー機能のサポート

各ノードのメモリは，プライベートな領域としてあるいは共有領域としてアクセスすることが可能である．どちらの領域として使用するかは OS が管理し，ページ単位で切り替える．制御チップはプロセッサが出力する 40 ビットの物理アドレスの上位 1 ビットでこれを区別する．共有領域としてアクセスされた場合，続く上位 10 ビットがどのノードのメモリかを示すフィールドとなり，残りの 29 ビットがノード内のオフセットとなる．

Cenju-4 は，プライベート領域間でデータ転送を行うメッセージ通信[5]と，共有領域へのメモリアクセスによる通信の両方のノード間通信をサポートしている．両方の通信方式を混在させてプログラムを記述することが可能である．

3. 分散共有メモリ機構

Cenju-4 は，ハードウェアがディレクトリ方式の無効化型プロトコルでキャッシュのコヒーレンス制御を行うことにより DSM を実現している．またメモリオーダリングモデルにストロングモデルを採用している．この DSM 機構は以下の特徴を有している．この章ではこれらの特徴を順に説明していく．

(a) ポインタ形式とビットパターン形式を組み合わせたディレクトリ
(b) ネットワークのマルチキャスト/ギャザー機能

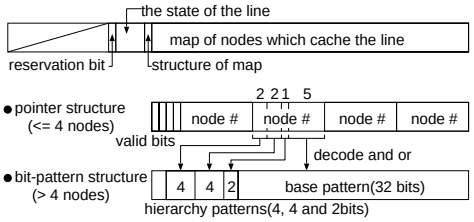


図 2 ディレクトリ構成
Fig. 2 Directory entry.

を利用した無効化要求の送信と応答の回収

(c) スタベイションを防止したキャッシュコヒーレンスプロトコル

(d) 多重化されたネットワークを必要としないデッドロック防止機構

3.1 ディレクトリ

Cenju-4 は，Origin[6]で採用されている Coarse Vector 方式[7]や JUMP-1[8]で採用されている階層ビットマップ方式と同様に，ハードウェアコストおよびディレクトリアクセスのコストがスケーラブルな方式を採用している．ただし，これらの方式はノードの管理が不精密になり，不要なメッセージが多く発生するという問題が存在する．Cenju-4では，ポインタ形式とビットパターン形式を組み合わせたディレクトリを採用することにより，前記 2 方式より精密なノード管理を実現している．以降，Cenju-4 のディレクトリについて説明する．

Cenju-4は，ハードウェアコスト削減のために，ディレクトリ専用のメモリを設けずに主メモリの一部をディレクトリ領域として利用する．そのため，各メモリブロック（128バイト）に割り当てるディレクトリのサイズは，主メモリのビット幅と密接に関係する．Cenju-4 は，ディレクトリの 1 エントリを 64 ビットとし，主メモリのビット幅と一致させている．**図 2** にディレクトリの構成を示す．ディレクトリには，予約ビット，メモリブロックの状態，メモリブロックを共有しているノードを特定する情報（以降，保持ノードマップ）が格納される．予約ビットと状態に関しては次節で述べる．この節では保持ノードマップについて説明する．

保持ノードマップは，共有しているノードの数によりその形式が切り替わる．ノードの数が 4 以下の場合はポインタ形式となる．ポインタ形式は，ノード番号を保持する 4 つのフィールドと，各フィールドが有効かどうかを示す4ビットからなる．一方，ノードの数が 4 を超えた場合はビットパターン形式となる．ビットパターンは，ノード番号（10 ビット）を 2-2-1-5 ビットの 4 つのフィールドに分け，各フィールドをデコードして

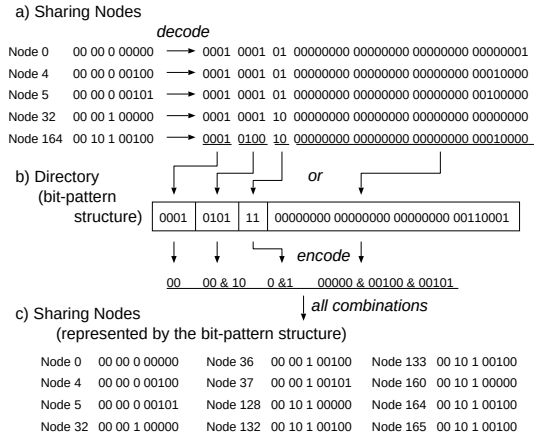


図 3 ビットパターン形式
Fig. 3 Bit-pattern structure.

得られる 4-4-2-32 ビット☆のデータである．ディレクトリには保持しているすべてのノードのビットパターンを論理和した値が格納される．**図 3** に，ビットパターン形式の例を示す．図のケースは，ノード 0，4，5，32，164 の 5 ノードが共有している場合である．この場合，ディレクトリには図 3 (b) に示す値で格納される．

両方式を組み合わせることにより，保持ノード数が 4 以下の場合と 32 ノード以下のシステムで精密なノード管理が行える．しかし，それ以外の場合では不精密になる可能性がある．図 3 に示す例では，実際の保持ノードが 5 ノードであるのに対し，12 ノードがディレクトリで表される保持ノードとなる．

図 4 に Origin で採用されている Coarse Vector 形式，JUMP-1 で採用されている階層ビットマップ形式，およびビットパターン形式の精密さを比較した結果を示す．Cenju-4 においてノードマップとして利用可能なビット数は 44 ビットに限定されており，3 方式で利用しているビット数はこの範囲内に収まるものを選択している．そのため，比較において各方式で利用しているビット数は異なるが，実装上の制約からは公平な比較であると考える．

Coarse Vector 方式は，ノードをいくつかのグループに分割して管理する方式である．たとえば 1024 ノードを 32 ビットで管理する場合，32 ノードからなる 32 グループにノードを分割する．グループ内にコピーを保持するノードが 1 つ以上存在する場合は対応するビットを 1 にし，1 ノードも存在しない場合は 0 とする方式である．階層ビットマップ方式はネットワー

☆ 次節で示すネットワークがサポートするマルチキャストが，4-4-64 ビットによる宛先指定形式を採用しているため，ディレクトリには 4-4-2-32 ビットを採用した．

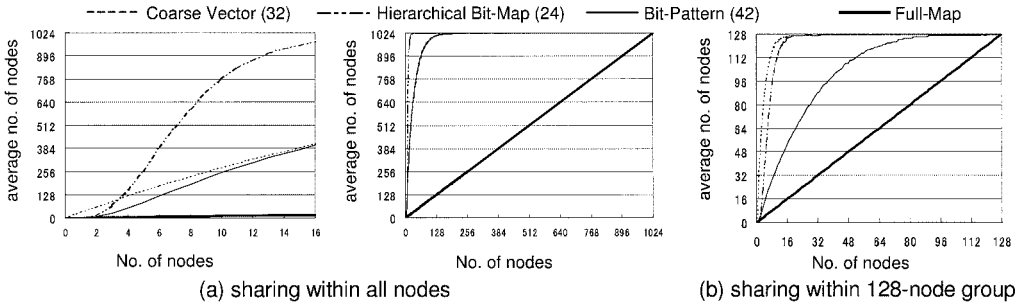


図 4 不精密なディレクトリ方式の振舞い
Fig. 4 Behavior of imprecise node maps (in a 1024-node system).

クの階層構造に強く依存した方式である．そのため，Cenju-4 でこの方式を採用する場合，ネットワークが 6 レベルの 4 進木構造をとることから，6 つの 4 ビットフィールドでノードマップは表現される．各 6 フィールドは木の各レベルに対応し，各 4 ビットは木の枝に対応する．同じレベルのすべてのスイッチは同じ 4 ビットフィールドを利用する．

図 4 (a) は 1024 ノードのシステムで共有しているノード数が変化した場合，図 4 (b) は 1024 ノードのシステムの，ある 128 ノードグループ内で共有しているノード数が変化した場合のグラフである．図から明らかなように，システム全体内で共有が行われる場合，多ノードで共有した場合どの方式にも差はない．しかし，共有しているノード数が少ない場合にビットパターン方式は優れている．また，システムのある 128 ノードグループ内で共有が行われる場合，ビットパターン方式が他の方式よりも優れていることが確認できる．これは，システムが複数のプログラムに分割利用されるマルチユーザ環境にビットパターン形式が適していることを示している．

3.2 マルチキャストとギャザー

Cenju-4 では，無効化要求の転送と応答の回収を効率化するために，ネットワークのマルチキャスト機能とギャザー機能を利用している．**図 5** に Cenju-4 でのマルチキャストとギャザーの動作の概要を示す．

従来，JUMP-1 も無効化要求をネットワークでマルチキャストし，また応答をネットワークでギャザーしていた．しかし，マルチキャストはネットワークの階層構造に強く依存し，そのマルチキャスト方式にあわせたディレクトリ方式には，前節で示したように，ノード管理の精密さに問題があった．また，ギャザー方式は，ギャザーメッセージがマルチキャストメッセージと同じスイッチを逆にたどることを前提とした方式であったため，図 5 に示すように，両メッセージで通る

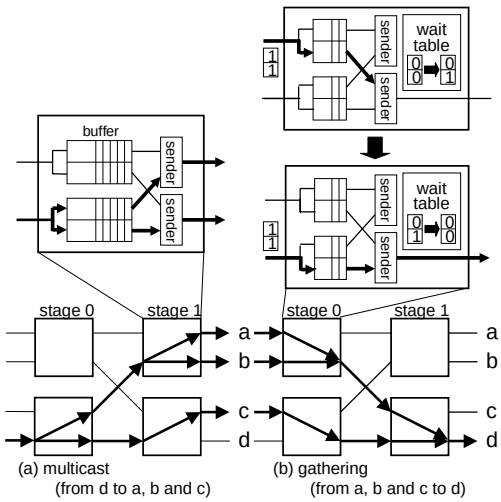


図 5 マルチキャストとギャザー
Fig. 5 Multicast and gathering functions of the network.

スイッチが異なる多段網には適用できなかった．

Cenju-4 で採用したマルチキャスト方式はネットワーク内の各スイッチが出力するポートを計算する方式である．マルチキャストメッセージの送信ノードは複数の宛先を指定した無効化要求をネットワークに出力する．宛先の指定は，ディレクトリと同じポインタ形式とビットパターン形式で行う．ネットワークの各スイッチは，宛先指定，スイッチの位置，ネットワークの構成の 3 つの情報から次段のどのスイッチに出力するかを決定する．この計算を各スイッチが行うことにより，ネットワークの構造に依存しないマルチキャストが実現されている．この方式は，階層構造を持たないメッシュ等のネットワークにも適用可能である．

マルチキャスト機能を実現するためには，2 つのスイッチが同じスイッチにマルチキャストメッセージを書き込む場合のアービトレーションで発生するデッドロックを回避する必要がある．Cenju-4 では，書き込

み時にアービトレーションが不要なクロスポイントバッファ方式でスイッチを構成し，バーチャルカットスルー方式でフロー制御を行い，この問題を解決している．クロスポイントバッファ方式では入力と出力の全組合せ分のバッファがスイッチ内に用意される．図5 (a) に示すように，2入力2出力のスイッチでは4個のバッファが必要となる．Cenju-4では，4×4のスイッチであることから，16個のバッファがスイッチ内に用意される．

マルチキャストされた無効化要求に対する応答の回収に，ネットワークのギャザー機能を利用する．Cenju-4で採用したギャザー機能は，ギャザーメッセージを送出するノードがいくつかの情報からメッセージが通る各スイッチでどのノードから送られてくるメッセージとギャザーするかを求めておく方式である．無効化要求を受けたノードは，受信した無効化要求に付加されていたマルチキャストの宛先，送出する応答の宛先ノード，ネットワークの構成の3つの情報をもとに，ネットワークの各ステージでどのポートからくる応答と待ち合わせを行うかを計算し，ギャザー情報として応答に付加しておく．ギャザー情報は，ネットワークが4入力4出力のクロスバスイッチが6段接続される構成であることから，6つの4ビットフィールドで構成される．各スイッチではそのギャザー情報をもとに応答の待ち合わせを行う．その結果，ネットワークですべての応答がギャザーされ，宛先ノードに1つの応答が届く．この方式により，マルチキャストメッセージとギャザーメッセージが異なるスイッチを通る多段網でもギャザーを実現することが可能となっている．

ネットワークは独立したギャザーを同時に複数行えるように設計されている．無効化要求には10ビットの識別子が付加され，この識別子を応答に付加してどの無効化要求に対する応答であるかを識別する．また，スイッチ内には，そのスイッチでのギャザーパターンを保持する1024エントリのテーブルが用意される．スイッチチップにこのテーブルが占める割合はゲート数で3.6%である．

3.3 コヒーレンスプロトコル

以降の説明では，メモリアクセスを発行したプロセッサを有するノードをマスタ，アドレスが示すメモリを含むノードをホーム，キャッシュにコピーを保持しているノードをスレーブと呼ぶ．

コヒーレンスプロトコルは，複数のノード間で要求や応答メッセージをやりとりして，メモリやキャッシュの状態やデータを更新する処理を行うシーケンスからなる．このコヒーレンスプロトコルは，メモリアクセスが有限時間内に完了することを保証するために，スタベイションおよびデッドロックが防止されたものである必要がある．この節ではスタベイションに関して説明し，デッドロックに関しては次節で説明する．

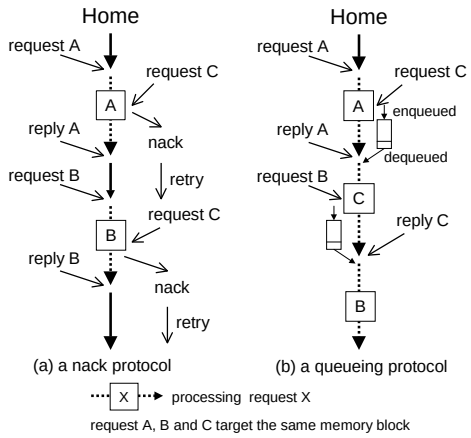


図6 否定応答プロトコルとキューイングプロトコル
Fig. 6 A nack protocol and a queuing protocol.

従来DASHやAlewife等のコヒーレンスプロトコルは，ホームやスレーブにおいて一貫性を維持することができない状態に陥った場合に否定応答をマスタに返し，マスタに再試行させていた．このような状況は，複数のノードが同じメモリブロックに対してメモリアクセスを行った場合に発生する．**図6** (a) は，要求Cが要求Aや要求Bに邪魔されて再試行を繰り返している状況を示している．このような状況が無限に繰り返されるとスタベイションが発生する．

従来，SCI[9]プロトコルはスタベイションの問題を解決していた．SCIでは，キャッシュにディレクトリを配置する分散ディレクトリ方式を採用し，共有しているノードをリスト結合する．複数のノードからの要求があった場合も，それらのノードをリスト結合してその順に要求を満たしていくことで，スタベイションを防止していた．しかし，この方式はキャッシュに分散したディレクトリ構造に強く依存したものであり，メモリで共有しているノードを一元管理するディレクトリ方式には適応できない．

Cenju-4は，スタベイションを防止するために，ベースとなるコヒーレンスプロトコルとしてAlewifeプロトコル（**図7** 参照）を採用している．これにより，スレーブがマスタに対して否定応答する状況が排除される．また，Cenju-4は，ホームがただちに処理できない要求を一時メモリに待避し後で処理するキューイング方式を採用している．これにより，ホームがマスタに否定応答する状況を排除している．

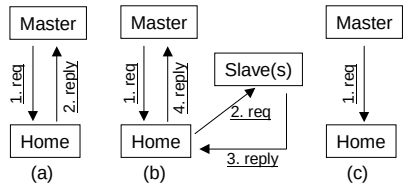


図 7 コヒーレンス制御のシーケンス
Fig. 7 Coherency control sequences on Cenju-4.

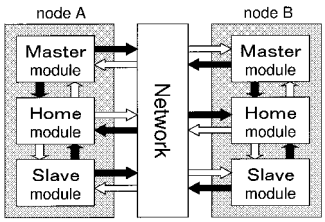


図 8 資源の依存関係
Fig. 8 Resource dependency graph.

図 6 (b) にキューイング方式の動作概要を示す．ホームはどの要求に対しても否定応答を返さない．ホームは，要求を処理して応答を待っている間に受けた要求 B と C をメモリに待避し，応答が戻ってきた時点でそれらの要求をメモリから取り出し処理する．要求を処理して応答を待っている状態であることは，メモリブロックの状態としてディレクトリに保持する．また，異なるメモリブロックに対する要求も同じバッファに待避し，バッファは FIFO キューとして管理する．待避した要求がキューの先頭であればディレクトリの予約ビットをセットする．予約ビットがセットされたメモリブロックの応答を処理すると，予約ビットをアンセットし，キューの先頭から要求を 1 つ読み出し処理する．以降順にキューから要求を読み出し処理を進める．ただし，途中処理がまだできないものがあれば，処理できなかった要求に対応するディレクトリの予約ビットをセットし，そのメモリブロックに対する応答が届くのを再度待つ．この処理は，キューが空になるまで繰り返される．

このようにして待避される要求にはキャッシュブロックの置き換えにより発生する書き戻し要求は含まれない．それは，Cenju-4 が，書き戻し要求に関してホームは受け付けた時点で他の要求を処理して応答を待っている状態であっても処理を行えるプロトコルを採用しているからである．これにより，1 ノードが受ける要求の最大数はプロセッサの最大発行要求数（Cenju-4 では 4）にノード数をかけて得られる数に抑えられる．1024 ノードのシステムでは，32 K バイトの領域がメモリに確保され，バッファとして利用される．

従来，Alewife[10] においても，キューイング方式を採用することによりスタベイションが防止できることが示されている．しかし，メモリブロックごとにバッファが必要であるとし，ハードウェア量から実現は困難としていた．Cenju-4 で採用した方式は，各ノードに 1 つのバッファで済む方式である．

3.4 デッドロックの防止

Cenju-4 では図 7 に示す 3 タイプのコヒーレンス制御フローが存在する．このように，コヒーレンスプロトコルは，マスタ，ホーム，スレーブ間で要求や応答をやりとりして一貫性を維持する．今までの説明では 1 つのメモリアクセス要求に着目して各ノードをマスタ，ホーム，スレーブと分類していた．ただし，各ノードはメモリアクセス要求によってマスタ，ホーム，スレーブと役割を変えて機能する．そのため，各ノードはマスタ，ホーム，スレーブとして機能する 3 つのモジュールを実際は備えている．

要求や応答を受けて各モジュールが行う処理は中断されることがなく，1 つの処理が終了するまで次の処理を開始できない．異なるノードに属するモジュール間の要求や応答の転送にはネットワークが用いられる．ネットワークはそれ自身デッドロック・フリーであるが，2 ノード間のパスは 1 つしかなく追い越しも許していない．そのため，**図 8** に示すような依存関係が生じる．図において各ノードの 3 モジュール，およびネットワークは資源であり，矢印はその資源間で要求や応答の転送が行われ依存関係があることを示している．図から明らかなように，グラフにはいくつものループが存在し，デッドロックが発生する可能性があることを示している．

従来，Alewife は多重化されていないネットワークでのデッドロック解決法を示している．デッドロックが発生しそうになると，ネットワークから入力される全メッセージをメモリに待避する方式である．しかし，メッセージ待避のために確保しなければならない領域のサイズが確定しないため，ソフトウェアで制御する必要があった．Cenju-4 では，特定のメッセージのみをメモリに待避する方式を採用することで，必要なバッファサイズが 1024 ノードで 64 K バイト × 2 と確定し，ハードウェア制御でメッセージの待避を行うことが可能となっている．

Cenju-4 は，図中白矢印で示した依存関係をなくすことでループをなくしデッドロックを防止している．メモリあるいはモジュール内にすべての要求や応答を受けきることが可能な大きさのバッファを設け，その

バッファに必要に応じて待避することで，依存関係をなくしている．必要とされるバッファは，スレーブが受ける要求用のバッファ，ホームがネットワークに出力する要求および応答用のバッファ，マスタが受ける応答用のバッファの3つのバッファである．他の矢印を選択してもループをなくすことは可能であるが，必要なバッファのサイズが最小化される選択を行っている．

マスタモジュールが受ける応答の数は最大でプロセッサの最大発行要求数に抑えられる．マスタモジュールはそれをすべて受け取れるバッファをモジュール内に有している．

スレーブモジュールは要求のみを受け取り，それにはデータは含まれない．1スレーブモジュールが受け取る要求の最大数は，プロセッサの最大発行要求数にノード数を掛けて得られる数に抑えられる．1024ノードのシステムではそれらの要求すべてを待避するのに必要な64Kバイトの領域がメモリに確保され，バッファとして利用される．

ホームモジュールは要求とデータが付加される可能性のある応答を出力する．しかし，このデータはつねにメモリブロックにあり，バッファに待避する必要はない．また無効化要求のマルチキャストをネットワークがサポートしているので，1つの要求あるいは応答を処理して出力する要求や応答の数は1以下である．これにより，バッファに待避される要求や応答の最大数は，スレーブと同様にプロセッサの最大発行要求数にノード数を掛けて得られる数に抑えられる．1024ノードのシステムではそれらの要求すべてを待避するのに必要な64Kバイトの領域がメモリに確保され，バッファとして利用される．

スレーブおよびホームモジュールともに4個のメッセージを受け取るバッファをモジュール内に有している．メモリに確保したバッファはこのモジュール内のバッファが溢れた場合にのみ用いられ，メモリアクセスレイテンシに悪影響を与えないように設計されている．

4. 性能評価

分散共有メモリ機構の性能を明らかにするために，ロードおよびストアアクセスを行ったときのレイテンシを測定した．また，NAS Parallel Benchmarks V2.3 Class Aを用いて，並列アプリケーションでの性能測定も行った．

4.1 基本アクセス性能

この節では，ロードおよびストアアクセスレイテンシの測定結果を示す．それぞれ，読み出しを行って2次キャッシュにミスした場合，複数のノードで共有しているデータに対して書き込みを行った場合のレイテンシである．Cenju-4は多段のネットワークで構成されており，その段数によりレイテンシが異なる．今回の評価では2段（〜16ノード）の場合と4段（〜128ノード）の場合のレイテンシを測定した．また，その結果より6段（〜1024ノード）の場合のレイテンシを推定した．

表1 ロードアクセスのレイテンシ（ns）
Table 1 Load access latencies (ns).

network stages (No.of nodes)	2 (〜16)	4 (〜128)	6 (〜1024)
a) private	470	470	470
b) local(clean)	610	610	610
c) remote(clean)	1690	2210	2730
d) local(dirty)	1900	2480	3060
e) remote(dirty)	3120	4170	5220

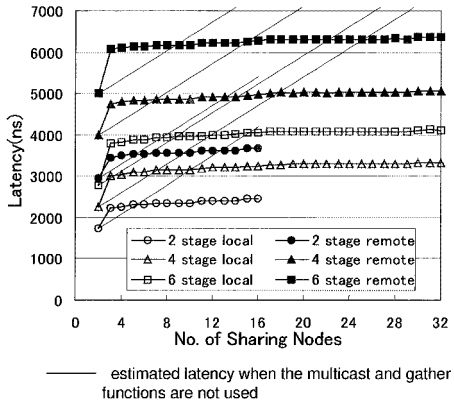


図9 ストアアクセスのレイテンシ
Fig. 9 Store access latencies.

表1にロードアクセスレイテンシを示す．a)，b)ともにノード内のメモリへのアクセスであるが，ディレクトリをアクセスするコストがb)には加わっている．また，b)，c)，d)，e)の差は，ネットワークを通過する回数とコヒーレンス制御シーケンスの違いによるものである．表より明らかなように，ロードアクセスレイテンシはネットワークのステージ数に比例して悪化することがあるものの，ノード数には比例しないことが確認できる．

図9にストアアクセスレイテンシの測定結果を示す．図には，マルチキャストおよびギャザー機能を利用しなかった場合のレイテンシを推定した結果も示している．まず，共有しているノード数が一定の場合，システムを構成するノード数ではなくネットワークのステージ数に比例してレイテンシが増加していることが確認できる．次に，共有しているノード数が与える影響を見る．まずマルチキャストとギャザー機能を

利用しなかった場合，ノード数の増加に比例してレイテンシが悪化している．一方，マルチキャストおよびギャザーを利用することにより，ノード数が与える影響を非常に小さなものとすることができている．1024 ノードで共有した場合，その値は 6.3 μ 秒と推定される．この結果より，マルチキャストおよびギャザー機能を利用することが，ストアアクセスのレイテンシをスケーラブルなものとするための有効な手段であることが確認できる．

一方，共有ノードが 2 か 3 以上かによりレイテンシが大きく変化する．共有ノードが 2 の場合は無効化要求を伝える相手が 1 ノードなのでマルチキャストおよびギャザーが必要ないのに対し，共有ノードが 3 以上の場合は相手が複数ノードとなりマルチキャストおよびギャザーが用いられるためである．この結果から，メモリアクセスレイテンシを最適化するためには，共有しているノード数によってマルチキャストおよびギャザー機能を利用するかしないかを切り替える必要があることが分かる．

4.2 アプリケーションを用いた評価

NPB の CLASS A を用いて評価を行った．用いたワークロードは BT，CG，FT，SP の 4 つである．実行にはネットワーク段数が 4 段の 128 ノードシステムを利用した．これらのプログラムは，提供されている逐次版のプログラムを最外ループのみ並列化したものである．ただし，ループ変換，共有配列の分割非共有化，共有データのノードへの配置指定を行い，メモリアクセスの最適化を施している．

4.2.1 スタベイション/デッドロック機構の評価

スタベイションおよびデッドロック機構がメモリアクセスレイテンシに与える影響を調べるために，メモリに設けた 3 つのバッファへのメッセージの待避率を測定した．スタベイション防止用にホームが管理するバッファ，デッドロック防止用にスレーブが管理するバッファとホームが管理するバッファの 3 つである．待避率は，メモリに待避されたメッセージ数をモジュールが受けたバッファを利用する可能性のあるメッセージの総数で割って求めている．

表 2 にその結果を示す．BT と SP は 64 ノードで実行した場合，CG と FT は 128 ノードで実行した場合のデータである．スタベイション防止用のバッファの待避率は，最も大きい SP でも 0.29%と非常に小さな値となっている．また，デッドロック防止用の両バッファの待避率も，最も大きい BT で 2.9%にとどまっている．

この結果と，メモリに設けたバッファへのメッセージの待避および読み出しのコストはあわせて 200 ns 程度とメモリアクセスレイテンシと比較して小さいことから，スタベイションおよびデッドロック防止機構が平均メモリアクセスレイテンシに与える影響は非常に小さいと考えられる．

表 2 メッセージのバッファへの待避率
Table 2 Ratios of messages queued in three buffers.

	starvation	deadlock	
	home	slave	home
BT	0.27%	2.9%	0%
CG	0.11%	0.92%	2.0%
FT	0%	0.07%	0%
SP	0.29%	0.49%	0.01%

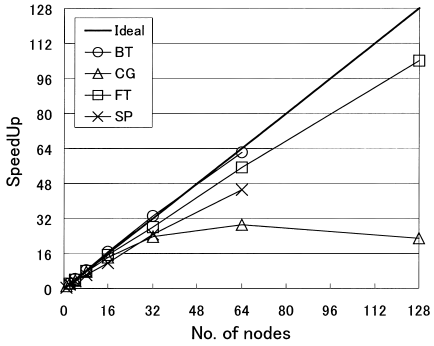


図 10 速度向上率
Fig. 10 Speedups of applications.

4.2.2 DSM 機構のスケーラビリティの評価

DSM 機構のスケーラビリティを評価するために，プログラムの実行性能を測定した．**図 10** に逐次プログラムに対する台数効果を示す．この図から明らかなように，BT，FT，SP は高い台数効果を示すものの，CG は飽和している．この原因を解析するために，16 ノードと，64 あるいは 128 ノードでプログラムを実行した場合の実行時間，実行命令数，2 次キャッシュミス率，およびそれらの内訳を測定した結果を**表 3** に示す．ただし，OS および同期フェーズは，実行命令数および 2 次キャッシュミス率の測定対象外としている．また，これらの値は全ノードの値を平均したものである．

実行時間はノード数増加分は減少しておらず，同期処理が実行時間に占める割合は増加している．このことより，各ノードの負荷の不均衡および同期処理コストが性能スケーラビリティを損なう 1 つの原因となっていることが分かる．また，実行時間から同期処理時間分を除いた値も同様にノード数増加分減少しておらず，平均メモリアクセスレイテンシがノード数の増加に従い悪化していることも分かる．

16 ノードと 64 あるいは 128 ノードの間で，メモ

表 3 プログラムの特徴
Table 3 Characteristics of applications.

	No. of nodes	execution time			executed instructions‡					secondary cache misses‡			
		total (sec)	system	sync.	total†	mem. access†	shared private	shared local	shared remote	ratio	shared private	shared local	shared remote
BT	16	203.722	3.26%	3.84%	25542	12027	82.7%	13.8%	3.60%	0.86%	71.9%	22.5%	5.59%
	64	56.324	2.94%	7.72%	6386	3007	82.7%	13.0%	4.35%	0.82%	75.0%	13.1%	11.9%
CG	16	5.348	1.93%	7.04%	369.0	141.0	66.4%	2.52%	31.1%	2.73%	90.0%	0.66%	9.31%
	128	4.182	0.88%	25.1%	46.61	17.85	66.8%	0.28%	32.9%	2.39%	18.4%	0.73%	80.9%
FT	16	9.222	5.04%	1.67%	1205	419.3	92.5%	4.69%	2.81%	0.77%	47.0%	37.6%	15.4%
	128	1.346	4.37%	8.92%	150.8	52.43	92.5%	4.52%	2.98%	0.79%	45.0%	35.7%	19.3%
SP	16	214.763	7.34%	5.42%	17420	6184	49.9%	19.9%	30.2%	1.24%	21.4%	59.3%	19.4%
	64	68.064	5.89%	12.8%	4356	1547	50.0%	17.3%	32.8%	1.03%	13.8%	39.8%	46.4%

†: $\times 10^6$, ‡: system and synchronization phases are not included in measurements

リアクセス命令の内訳には大きな変化はない．ローカルメモリのアクセス比率がわずかに下がり，リモートメモリへのアクセス比率がその分増加しているのみである．一方，2 次キャッシュミス率の内訳は大きく変化している．ミスにリモートメモリアクセスが占める割合は，BT で 6.34%，CG で 71.5%，FT で 3.90%，SP で 27.1%増加している．これらの増加は，平均メモリアクセスレイテンシを悪化させ性能のスケーラビリティを損なう原因となっている．とりわけ，CG が高いリモートミス率の増加を示しており，それが性能を飽和させる原因の 1 つとなっていることが分かる．

CG では，共有データは全ノードに均等に分割され，各ノードは割り当てられた部分の計算を行う．プログラムのあるフェーズでは，各ノードは共有データの割り当てられた部分を更新し，次のフェーズでは各ノードが共有データをすべてアクセスする．非共有データに関しては，ノード数の増加によりデータサイズが小さくなるため，その分キャッシュミス率も下がる．しかし，共有データに関してはノード数が変化しても各ノードがアクセスするデータサイズが変わらないため，キャッシュミス率に共有データへのアクセスが占める割合が増加している．さらに，共有データへのメモリアクセスに占めるリモートメモリへのアクセス命令数の割合も大幅に増加し，2 次キャッシュミス率に占める割合もあわせて増加している．

以上より，全ノードが共有データすべてにアクセスするプログラムに関しては，リモートメモリへのアクセス比率がノード数の増加に従い高くなり，性能が飽和してしまうことも明らかとなった．

5. おわりに

Cenju-4 の分散共有メモリ機構は高いスケーラビリティを実現することを目的として設計され，以下の特徴を持つ．

(a) ポインタ形式とビットパターン形式を組み合わせたディレクトリ
(b) ネットワークのマルチキャスト/ギャザー機能を利用した無効化要求の送信と応答の回収
(c) スタベイションを防止したキャッシュコヒーレンスプロトコル
(d) 多重化されたネットワークを必要としないデッドロック防止機構

Cenju-4 で採用したディレクトリ方式は，ノードあたりに必要なメモリのサイズが一定であり，かつデータを共有しているノードを特定するのに要する時間が一定で済むという特徴を有する．また，他の同じような特徴を有する方式と精密さを比較すると，大きなシステムの一部を利用するようなマルチユーザ環境で特に優れた振舞いを示すことが確認できた．

Cenju-4 で採用したネットワークのマルチキャストおよびギャザー機能は，ネットワークの構造に依存しないものである．マルチキャストの宛先指定はネットワークの構造ではなくディレクトリ方式にあわせて決定できる．また，メモリアクセスレイテンシを計測した結果，ネットワークのマルチキャストおよびギャザー機能を利用することにより，ストアアクセスのレイテンシをスケーラブルなものとすることができることが確認できた．

スタベイションおよびデッドロックの防止は，1024 ノードのシステムで，それぞれメモリに 32 K バイト，および 128 K バイトのバッファを確保して，そこに特定のメッセージを待避することにより実現できることを示した．また，アプリケーションを用いて評価を行った結果，メッセージがバッファに待避されるような状況は希であることも確認できた．

また，アプリケーションを用いた DSM 機構のスケーラビリティの評価を行った結果，いくつかのアプリケーションでは高い性能を発揮することが確認できた．しかし，全ノードが共有データすべてにアクセスするような振舞いをするプログラムに関しては，リモートメ

モリへのアクセス比率がノード数の増加に従い高くなり，性能が飽和してしまうことも明らかとなった．

今後はより多面にわたる評価を進めていくとともに，Cenju-4で性能が飽和するようなアクセスパターンを持つプログラムの性能を改善する方式の検討を行っていく．

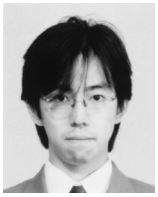
参 考 文 献

1) Chaiken, D., Kubiatowicz, J. and Agarwal, A.: LimitLESS Directories: A Scalable Cache Coherence Scheme, *4th International Conference on Architectural Support for Programming Languages and Operating Systems*, pp.224–234 (1991).
2) Simoni, R. and Horowitz, M.: Dyanmic Pointer Allocation for Scalable Cache Coherence Directories, *Proc. International Symposium on Shared Memory Multiprocessing*, pp.72–81 (1991).
3) Lenoski, D., Laudon, J., Gharachorloo, K., Gupta, A. and Hennesy, J.: The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, *Proc. 17th Annual International Symposium on Computer Architecture* (1990).
4) Agarwal, A., Bianchini, R., Chaiken, D., Johnson, K.L., Kranz, D., Kubiatowicz, J., Lim, B.-H., Mackenzie, K. and Yeung, D.: The MIT Alewife Machine: Architecture and Performance, *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.1112–1118 (1978).
5) Kanoh, Y., Nakamura, M., Hirose, T., Hosomi, T., Takayama, H. and Nakata, T.: Message Passing Communication in a Parallel Computer Cenju-4, *Proc. 2nd International Symposium on High Performance Computing*, LNCS, Vol.1615, pp.55–70, Springer-Verlag (1999).
6) Laudon, J. and Lenoski, D.: The SGI Origin: A ccNUMA Highly Scalable Server, *Proc. 24th Annual International Symposium on Computer Architecture* (1997).
7) Gupta, A., Weber, W.-D. and Mowry, T.: Reducing Memory and Traffic Requirements for Scalable Directory-Based Cache Coherence Schemes, *Proc. 1990 International Conference on Parallel Processing*, pp.312–321 (1990).
8) Matsumoto, T., Nishimura, K., Kudoh, T., Hiraki, K., Amano, H. and Tanaka, H.: Distributed Shared Memory Architecture for JUMP-1 a general-purpose MPP prototype, *Proc. 1996 International Symposium on Parallel Architectures, Algorithms and Networks* (1996).
9) Institute of Electrical and Electronics Engineers, Inc.: *IEEE Standard for Scalable Coherent Interface* (1993).
10) Kubiatowicz, J.D.: Integrated Shared-Memory and Message-Passing Communication in the Alewife Multiprocessor, PhD Thesis, Massachusetts Institute of Technology (1998).

(平成 11 年 8 月 30 日受付)
(平成 12 年 2 月 4 日採録)

細見 岳生（正会員）
1969 年生．1994 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年 NEC 入社．並列計算機アーキテクチャに関する研究に従事．

加納 健（正会員）
1962 年生．1989 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年 NEC 入社．以来 Cenju-3，Cenju-4 等の並列計算機の研究開発に従事．



中村 真章（正会員）
1969 年生．1993 年東京大学工学部電気工学科卒業．同年 NEC 入社．並列計算機アーキテクチャに関する研究に従事．

広瀬 哲也（正会員）
1967 年生．1992 年筑波大学大学院修士課程理工学研究科修了．同年 NEC 入社．並列処理アーキテクチャの研究に従事．

中田登志之（正会員）
1957 年生．1985 年京都大学大学院工学研究科情報工学専攻博士後期課程単位取得退学．同年 NEC 入社．工学博士．並列処理アーキテクチャ/ライブラリ/応用の研究に従事．