Home Proxy Cacheによる分散共有メモリの高速化

市川明弘[†] 小野 航[††] 中條拓伯[†]
工藤知宏[†††] 天野英晴[††]

ワークステーション（WS)/PCクラスタ上の並列処理環境として分散共有メモリシステムを実装する場合，標準的なEthernetでは通信遅延が大きいため高速に通信を行えるネットワークインタフェースを用いて，キャッシュのミスペナルティを削減する試みがなされている．WS/PCクラスタではネットワークインタフェースをI/Oバスに接続するのが一般的である．その場合，キャッシュミスが生じてホームメモリの転送要求が発生するとI/Oバスを経由してホームメモリのページデータが転送される．しかしながら，I/Oバスを経由した場合，バスバンド幅の限界や転送遅延から，ページ転送に要する時間が大きく，ミスペナルティが性能向上の妨げとなる．そこで，ネットワークインタフェースにホームメモリのキャッシュを設けてI/Oバスを経由したページデータの転送回数を減らし，ミスペナルティの削減を試みる．本論文では，ホームメモリをネットワークに対してキャッシングするHome Proxy Cacheを提案し，ワークステーションをシリアルリンクで接続したクラスタに実装した場合の性能を評価し，今後の方向性について述べる．

High Performance Distributed Shared Memory with Home Proxy Cache

Akihiro Ichikawa,[†] Wataru Ono,[††] Hironori Nakajo,[†]
Tomohiro Kudoh[†††] and Hideharu Amano[††]

There have been a plenty of studies of trying to reduce miss penalty by using a high speed network interface instead of Ethernet in implementing distributed shared-memory system on a workstations (WS)/PC cluster. In a WS/PC cluster system, since a network interface is usually connected to an I/O bus, transferring page data between element cluster and network interface via an I/O bus is performed when cache miss occurs. However, from limit of bus bandwidth or transfer latency, miss penalty prevents higher performance in accessing to home memory via an I/O bus. Therefore, we try to reduce the number of times of transferring page data via an I/O bus by using cache of home memory on a network interface to relax an average of miss penalty. In this paper, we propose Home Proxy Cache which is a cache of home memory. We have implemented Home Proxy Cache in a WS cluster in which each WS is connected via fast serial links. We describe a implementation overview, results of experiments and future works.

1. はじめに

WSクラスタ，およびPCクラスタ上で分散共有メモリシステムを構築する場合，ノード間の通信遅延が大きいため，ノード間の通信量を少なくしなくてはならない．そのため，各ノードはキャッシュを介して共有メモリにアクセスして他のノードとの通信量を減らし，共有メモリへの見かけの平均アクセス時間を短くする必要がある．一般に，共有メモリではキャッシュの状態の変化が生じた場合，ホームメモリ，および各ノードの持っているキャッシュの一貫性を保つ必要が生じる．WS/PCクラスタにおいて，各ノードのキャッシュの状態の変化を監視するのはその構造上非現実的である．そこで，キャッシュの状態変化の起こったノードがホームメモリを管理しているマネージャノードにホームメモリ，およびキャッシュの一貫性を保つように依頼する．マネージャがキャッシュの一貫性を保つ方法としては，該当するキャッシュをすべて無効化する方法や，差分を送って内容を更新する方法がある．また，キャッシュの内容とホームメモリの内容をつね

† 神戸大学工学部情報知能工学科
Department of Computer and Systems Engineering, Kobe University
†† 慶應義塾大学理工学部情報工学科
Department of Computer Science, Keio University
††† 新情報処理開発機構
Real World Comupting Partnership

に一致させておくか，もしくはキャッシュの内容が最新である状態を許すかといった選択肢もある[1]．

LANを構築するのに一般的に用いられているEthernet上のTCP，もしくはUDPによる通信は，送受信時のシステムコールのオーバヘッドによる通信遅延が大きく，ユーザレベルでの送受信が可能なネットワークが必要となる．本研究室ではユーザが直接利用可能なシリアルネットワークとして，STAFF-Link（Serial Transparent Asynchronous First-in First-out Link）[2]を開発し，WS/PCクラスタの通信遅延自体の削減をはかった[3),4]．しかしながら，要素プロセッサ上でのルーティング処理は負荷が大きくシステムの性能を劣化させるため，専用のルータボードを開発し，ルーティング処理の負荷を軽減した[5),6]．また，ホームメモリの管理を要素クラスタで行うとプロセッサに負荷がかかり，並列処理アプリケーションのプロセッサ利用率が低くなってしまう．そこで，STAFF-Linkのインタフェースボード上にプログラマブルプロセッサを搭載し，ホームメモリの管理の一部をインタフェースボードで行って要素クラスタのプロセッサの負荷を軽減した．しかしながら，ホームメモリ自体は要素クラスタのメインメモリ上に確保されているため，他ノードのキャッシュミスによりホームメモリからページデータを読み出す場合や，ライトバックによりページデータの書き戻す場合は，要素クラスタのプロセッサに処理を依頼しなくてはならない．ネットワークインタフェースは要素クラスタのI/Oバスを介して接続されているのでページデータはI/Oバスを経由して転送されるため，I/Oバスの転送遅延によるミスペナルティが性能向上の妨げとなる．また，その間は要素クラスタのプロセッサを占有するために並列処理アプリケーションのプロセッサ利用率が低くなってしまう．

そこで，ネットワークインタフェース上にホームメモリをI/Oバスの外側に対してキャッシングするためのHome Proxy Cache（HPC）[7),8]を設け，要素クラスタに対して発行されるホームメモリへのアクセス要求をI/Oバスの外側のネットワークインタフェース上で処理する．これにより，平均ミスペナルティが削減され，また，要素クラスタのプロセッサの負荷を小さくして並列処理アプリケーションのプロセッサ利用率を高める．本論文では，HPCを実装した分散共有メモリシステムをWSクラスタ上に構築してその性能の評価を行い，HPCの有効性について検証する．

2. システムの構成について

今回実装したシステムの各ノードのハードウェア，

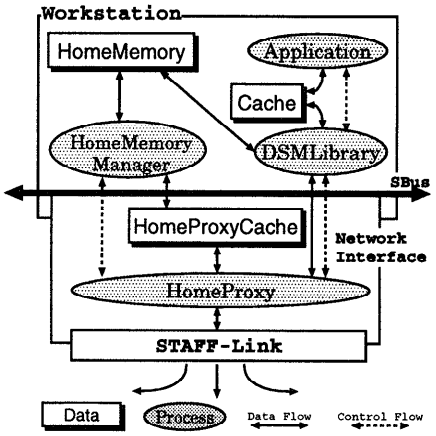


図1 システムのノード構成
Fig. 1 Structure of a node in this system.

およびソフトウェアの構成を図1に示す．

要素クラスタとSTAFF-Link用のネットワークインタフェースはSBusを介して接続されており，ノード間の通信はSTAFF-Linkを用いて行う．また，図1中のHome Memory Manager，Application，DSM-Library，Home Proxyはソフトウェアにより実現される機能を表しており，Home Memory，Cache，Home Proxy Cacheにはページ単位でデータが格納されている．

本システムでは，メモリコンシステンシモデルにシーケンシャルコンシステンシモデル[9)]を，キャッシュコヒーレンシ制御には無効化型ライトバック方式を採用している．ホームメモリは一定の大きさのページに分割され，各ページの管理には分散フルビットマップディレクトリ[10)]を用いている．Cacheは並列処理アプリケーションから見た共有メモリに対するキャッシュで，ダイレクトマッピング方式で管理している．並列処理アプリケーションが用いる共有メモリの各ページに対してキャッシュラインを動的に確保するためにcapacity miss，およびconflict miss[11)]は起こらない．

2.1 ハードウェア構成

要素クラスタに用いたワークステーションの仕様を**表1**に示す．また，ネットワークインタフェースの構成を図**2**に示す．

ネットワークインタフェースはSTAFF-Link用のルーティングハードウェアであり，プログラマブルプロセッサとしてTI社製のDSP（TMS320C40）を搭載し，内部クロック周波数を40 MHzで動作させている．また，プログラムコード用，およびデータ用のRAMをそれぞれ2 MB搭載している．DSPをプロセッサとして採用したのはGlobal BusとLocal Busの2つ

表1 要素クラスタのスペック
Table 1 Specification of element cluster.

Workstation	Sun SPARCstation-20
CPU	SuperSPARC (60 MHz) × 2
OS	Solaris 2.5.1 (SunOS 5.5.1)
Memory	64 MB

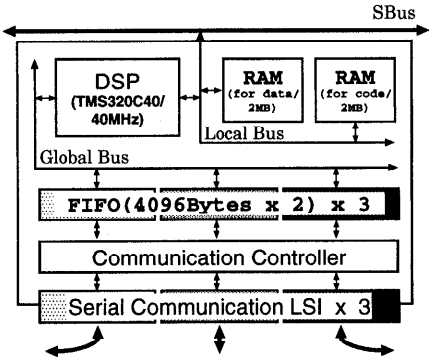


図2 ネットワークインタフェースの構成
Fig. 2 Structure of network interface.

のバスを持っており，2つのバスにまたがるデータ転送を1ステップで実行できるからである．要素クラスタとネットワークインタフェースはSBusを介して接続しており，要素クラスタのプロセッサからプログラムコード用，およびデータ用のRAMへのアクセスが可能である．

また，ネットワークインタフェース上にはノード間の通信に用いるSTAFF-Linkのポートを3ポート搭載している．STAFF-Linkの各ポートは**シリアル通信用LSI**，**送受信用のFIFO（各4KB）**，**通信コントローラ**の3つの要素から成り立っている．STAFF-Linkを通じたデータの送受信は，送信用FIFOから受信用FIFOへのデータ転送によって行われる．通信路上のシリアル通信およびFIFO間とのシリアル-パラレル変換はシリアル通信用LSIにより自動的に行われ，FIFOの状態によって通信を停止/再開を行うハンドシェークは通信コントローラが行うため，DSPから見るとノード間の通信はFIFOメモリへのアクセスと仮想化される．送信用FIFOから受信用FIFOへのデータ転送速度は最大で140 Mbpsである．

2.2 ソフトウェア構成

本システムのソフトウェアは **DSMLibrary**，**Home Memory Manager**，**Home Proxy** の3つの要素から構成される．DSMLibrary，およびHome Memory Managerは並列処理アプリケーションにリンクされるプログラムで，要素クラスタ上で動作する．DSMLibrary，Home Memory Managerではそれぞれ以下の処理を行う．

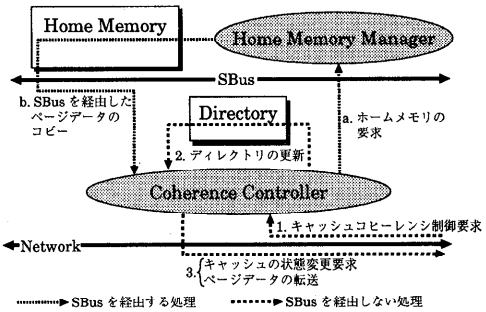


図3 キャッシュコヒーレンシ制御の処理の流れ
Fig. 3 Processing flow of preserving cache coherency.

- DSMLibrary
  - 並列処理アプリケーションのキャッシュミスの検知
  - システムに対するキャッシュコヒーレンシ制御の要求の発行
  - システムからのキャッシュコヒーレンシ制御の要求の処理
- Home Memory Manager
  - Home Proxyからのホームメモリの転送要求の処理

Home Proxy Cache

ネットワークインタフェース上のプロセッサにCoherence Controllerを設けてキャッシュコヒーレンシ制御を行う場合，Coherece Controllerは，Home Memory Managerに代わりホームメモリのディレクトリ管理，およびキャッシュコヒーレンシ制御のためのキャッシュの状態変更の要求の発行を行う．この場合のキャッシュコヒーレンシ制御要求の処理の流れを**図3**に示す．

他ノードのキャッシュの無効化要求のようなホームメモリへのアクセスが必要ない場合の処理の流れは1 → 2 → 3となり，Home Memory Managerに処理要求を出す必要がないので要素プロセッサの負荷を軽減でき，並列処理アプリケーションのプロセッサ利用率が向上する．しかしながら，ホームメモリへのアクセスが必要な場合の処理の流れは1 → 2 → a → b → 3となり，Home Memory Managerにページデータの転送要求を発行する必要がある．bのSBusを経由したページデータの転送は遅延が大きく，ミスペナルティの削減の妨げとなる．また，ページデータの転送は要素プロセッサを占有するため要素プロセッサの負荷が大きくなる．そこで，ホームメモリをI/Oバスの外側に対してキャッシングを行うためにHome Proxy Cacheを導入し，I/Oバスを経由するページデータの

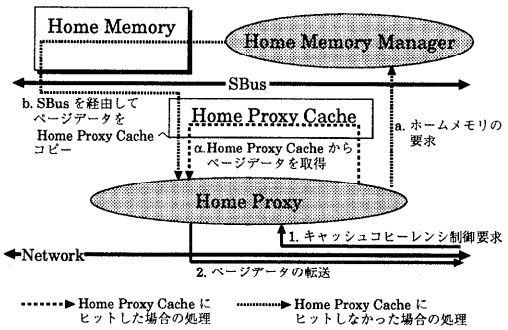


図4 キャッシュコヒーレンシ制御の処理の流れ
Fig. 4 Processing flow of preserving cache coherency.

転送回数の削減を試みる．

Home Proxy Cache（HPC）を導入した場合の，ホームメモリへのアクセスの必要なキャッシュコヒーレンシ制御要求を受けとった場合の処理の流れを図4に示す．ただし，ディレクトリ管理の処理は省略している．

Home Proxy はHPCの管理を行うとともに，前述の Coherencr Controller としての処理も行う．Home Proxy はホームメモリへのアクセスが必要な要求を受けとった場合，HPCに処理対象となるページデータがキャッシングされているかを判断する．この処理はディレクトリを参照するだけなので，高速に処理が行われる．HPCにヒットした場合の処理の流れは $1 \rightarrow \alpha \rightarrow 2$ となり，ネットワークインタフェース上にあるページデータを他のノードに転送できるので処理が高速化され，ミスペナルティが削減できる．また，Home Memory Manager に処理要求を発行する必要がないので，要素プロセッサの負荷を軽減できる．HPCにヒットしなかった場合の処理の流れは $1 \rightarrow a \rightarrow b \rightarrow 2$ となり，Home Memory Manager にページデータを HPC に転送するように要求を発行する．HPCの capacity miss によってページデータをホームメモリに書き戻す場合にも Home Memory Manager に要求を出して SBus を経由したページデータの転送を行う必要がある．

HPCは256ページ分のエントリを持ち，LRUで管理され，ページデータを格納するキャッシュラインの選択にはフルアソシアティブ方式を採用している．このため，HPCでは conflict miss は起こらない．HPCは LRU で管理されているので，ページデータの転送要求が頻繁に来るページは HPC にヒットしやすい．また，リードアクセスが大半を占めるがたまにライトアクセスが行われるようなページに関しても HPC のヒット率は高くなる．反対に，ライトアクセスが頻繁に行われ，各ノードを移動するようなページは HPC にキャッシングされないので，HPCによってミスペナルティが小さくなることは期待できない．

表2 リードミス時のミスペナルティ
Table 2 Miss penalty caused by read-miss.

ヒット/ミスヒット	ミスペナルティ（μs）
ヒット	4856
ミスヒット	5458

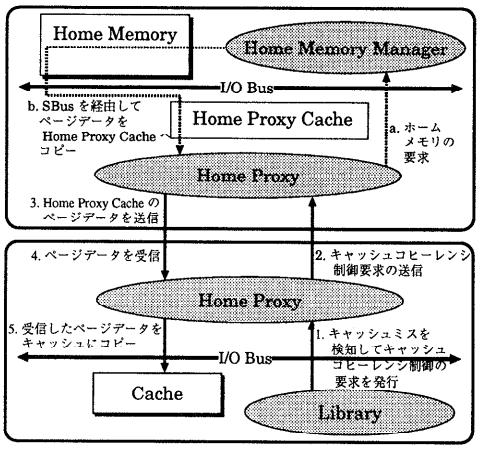


図5 リードミス時の処理の流れ
Fig. 5 Processing flow of read-miss.

3. 性能評価

本システムの基本性能と，並列処理アプリケーションとして行列演算，および SPLASH2 ベンチマークの FFT を実行した場合の性能の評価を行う．基本性能の評価では，ホームメモリへのページデータの要求の発生するキャッシュコヒーレンシ制御として，並列処理アプリケーションがキャッシュへのリードミスを起こした場合のミスペナルティを計測し，評価を行う．

3.1 基本性能の評価

リードミス時のミスペナルティ

表2に，HPCにヒットした場合とヒットしなかった場合のミスペナルティを示す．表2より，HPCにヒットした場合，ミスペナルティは約 600 μs 減少している．よって，HPCによるミスペナルティ削減の効果は約10%となる．

この場合の処理の流れを**図5**に示す．また，ミスペナルティの内訳のうち，主要なものを**表3**に示す．

表3より，ミスペナルティの約55%をページデータの受信処理で費やしていることが分かる．よって，ネットワークインタフェースに機能を追加してページデータの受信処理に必要な時間を少なくしてミスペナルティを削減することで，HPCの効果は現在より大きくなると思われる．

表3 処理時間の内訳
Table 3 Composition of miss-penalty.

処理内容	処理時間 (μs)
処理b	602
処理3	1027
処理4	3012
処理5	610

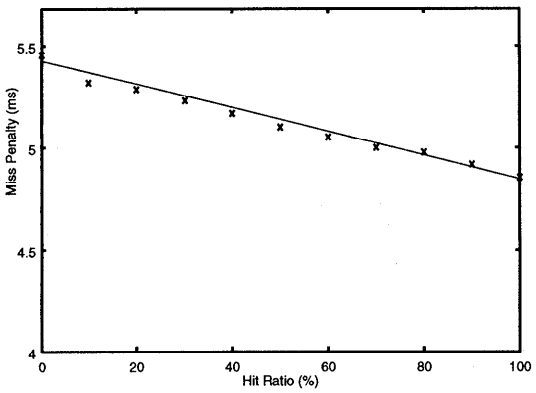


図6 ヒット率と平均ミスペナルティ
Fig. 6 Average of miss penalty with hit ratio.

表4 要素プロセッサの利用率
Table 4 Utilization ratio of element processor.

ヒット/ミスヒット	ヒット	ミスヒット
ジョブの処理時間 (ms)	4312	5768
ジョブのみ (ms)	4283	4283
プロセッサ利用率 (%)	99.3	74.3

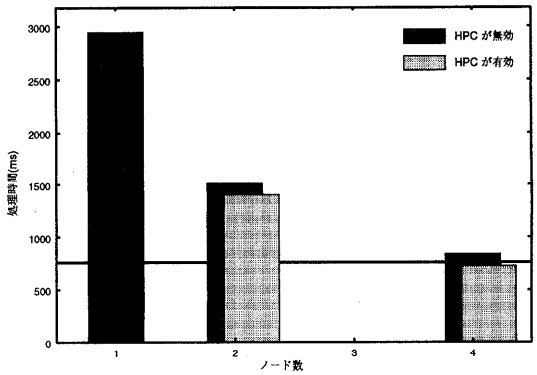


図7 128 × 128 正方行列の積の演算時間
Fig. 7 Processing time to make product of 128 × 128 matrix.

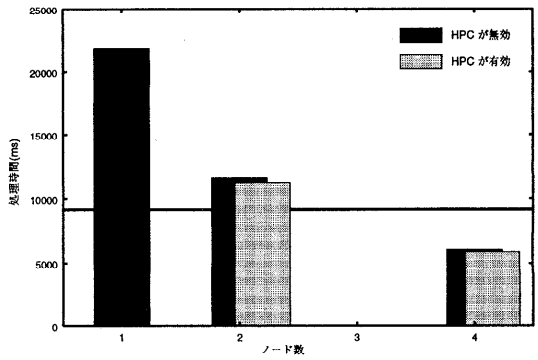


図8 256 × 256 正方行列の積の演算時間
Fig. 8 Processing time to make product of 256 × 256 matrix.

ヒット率を変化させた場合の平均ミスペナルティ

HPC のヒット率と平均ミスペナルティのグラフを**図6**に示す.

図6より, HPC へのヒット率に応じて平均ミスペナルティが小さくなっていることが分かる. よって, 同一ページのページデータの転送要求が連続して到着した場合, 2番目以降の要求は HPC にヒットするので平均ミスペナルティは小さくなる. 連続して到着した要求は STAFF-Link の FIFO メモリ内で待ち行列を構成しているので各要求が処理されるまでの平均待ち時間と未処理の要求の数が少なくなり, システムの処理性能が向上するものと思われる.

並列処理アプリケーションのプロセッサ利用率

次に, HPC にヒットした場合とヒットしなかった場合の, Home Memory Manager の動作するノードの並列処理アプリケーションのプロセッサ利用率を**表4**に示す. ただし, 表4中の "ジョブの処理時間" はあるビジーなジョブの処理に要した時間で, その間に 3000 回のページデータの要求が到着する. HPC にヒットした場合, しなかった場合とは, それぞれ 3000 回の要求のすべてがヒットした場合, もしくはヒットしなかった場合である. また, "ジョブのみ" はそのジョブの処理の間にページデータの要求などのチェックをいっさい行わなかった場合の処理時間である. 表4より, HPC を導入した場合, プロセッサの利用率が大幅に向上することが分かる. よって, HPC は並列処理アプリケーションのプロセッサ利用率の向上に有益であるといえる.

3.2 アプリケーションによる性能の評価

行列演算による評価

アプリケーションとして, 正方行列の積の演算を用いて, 本システムの性能を評価する.

行列の大きさを 128 × 128, 256 × 256, 512 × 512 と変えて計測した結果を**図7**, **図8** および**図9**に示す. ただし, 図中の横線は本システムを用いずに1台のワークステーションを用いて計算を行った場合の処理時間である.

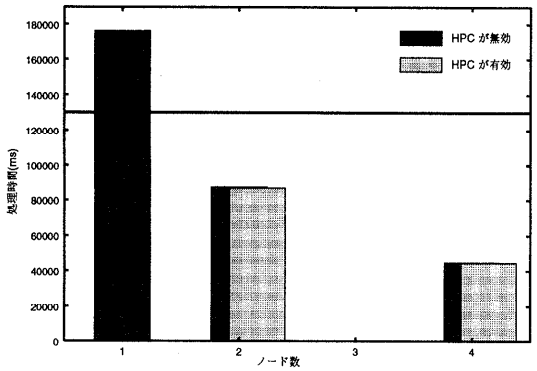


図 9 512 × 512 正方行列の積の演算時間
Fig. 9 Processing time to make product of 512 × 512 matrix.

表 5 4 ノードの場合の処理時間 (ms)
Table 5 Processing time using 4 nodes.

サイズ	HPC 有	HPC 無	ヒット率	性能比
128 × 128	731.32	840.95	57.1	1.15
256 × 256	5879.58	6059.34	57.1	1.03
512 × 512	44594.67	44816.05	57.1	1.00

HPC は他のノードからの要求に対して効果を発揮するものなので，単一のノードでシステムを構成した場合には意味を持たない．よって，1 ノードで HPC を有効にして処理を行った場合の計測は行っていない．

行列の大きさが 128 × 128 の場合，4 ノードで計算を行ってようやく並列処理を行わない場合と同等となる．これは計算量に対して共有メモリへのアクセス時のキャッシュミスが多いために，演算時間の大半をミスペナルティが占めたためである．しかしながら，すべての場合において台数効果が良好に現れていることが分かる．

それぞれの場合において，4 ノードで計算を行った場合の HPC を有効にした場合と無効にした場合の処理時間，および HPC が有効である場合の性能比を**表 5** に示す．すべての場合において HPC のヒット率は 57%ほどとけっして高くはない．しかしながら capacity miss は起きていないので，この数値は行列演算の静的なキャッシュへのアクセスパターンによるもので，HPC のライン数を増やしてもヒット率は高くならないことが分かる．HPC による処理性能の向上率は，128 × 128 の行列の場合では 15%ほどとなっているが，512 × 512 の行列の場合にはほぼ 0 である．これは，計算量に対して共有メモリへのアクセスによるキャッシュミスの頻度が小さいため，HPC にヒットすることによるミスペナルティの減少，および要素プロセッサの利用率の向上が演算時間に対して小さくなっているためと思われる．

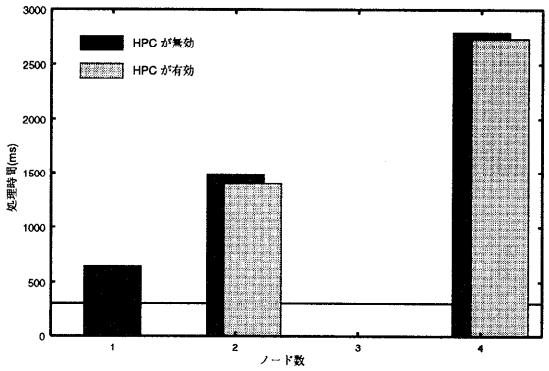


図 10 データポイントが 4096 のときの処理時間
Fig. 10 Processing time to process 4096 data points.

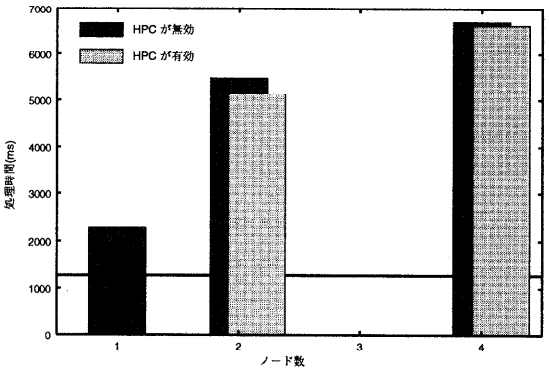


図 11 データポイントが 16384 のときの処理時間
Fig. 11 Processing time to process 16384 data points.

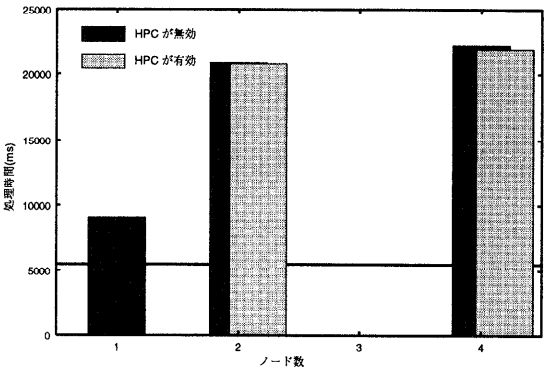


図 12 データポイントが 65536 のときの処理時間
Fig. 12 Processing time to process 65536 data points.

FFT による評価

次に，アプリケーションとして SPLASH2 に含まれる FFT を用いて本システムの性能を評価する．処理するデータポイントの数を 4096，16384，65536 と変えて計測した結果を**図 10**，**図 11**，**図 12** に示す．ただし，図中の横線は 1 台のワークステーションを用い

表6 ノード数による処理時間の変化
Table 6 Change of processing time by number of nodes.

データ数	2ノードで実行	4ノードで実行	性能比
4096	1400.081	2735.080	0.51
16384	5132.685	6637.545	0.77
65536	20759.235	21939.345	0.95

表7 4ノードの場合の処理時間（ms）
Table 7 Processing time using 4 nodes.

データ数	HPC有	HPC無	ヒット率	性能比
4096	2735.080	2797.046	93.0	1.02
16384	6637.545	6735.059	93.1	1.01
65536	21939.345	22152.225	93.3	1.01

て処理を行った場合の処理時間である．

行列積の計算のときと同様に，1ノードでHPCを有効にして処理を行った場合の計測は行っていない．

すべての場合において，1台のワークステーションで処理を行った場合より処理に時間がかかっている．また，システムを構成するノード数が少ないほど処理時間が短い．各ノードに処理を分散すると，ノード間で同期をとったり処理の途中経過をノード間で共有したりするといったコストの高い処理を行う必要がある．FFTは粒度の細かい処理を行うので，各ノードの計算量に対して同期をとったりデータを共有したりすることが多い．よって，システムを構成するノード数が多くなるほど必要な処理時間が大きくなったものと思われる．それぞれの場合において，HPCを有効にしたときの2ノードでの処理時間と4ノードでの処理時間，およびその比率を**表6**に示す．また，4ノードで，HPCを有効にした場合と無効にした場合の処理時間，および性能比を**表7**に示す．表6より，処理するデータポイントの数が小さいほど処理するノードが増えたときの性能の劣化が激しくなる．これは，データポイントの数が少ないほど処理時間に占めるノード間の同期，もしくは途中経過の共有に要する時間が大きくなるため，システムを構成するノード数が増えるとノード間の通信量が増え，処理時間が増大することを示している．また，表7より，処理するデータポイントの数が小さいほど，HPCによる性能の向上比が大きくなる．これは，処理時間に占める通信時間が大きくなるため，HPCによるミスペナルティ減少の影響が大きくなるためと思われる．

すべての場合においてHPCのヒット率は高い値を示しているが，HPCが無効である場合に対する性能比は高くない．FFTの処理では，他のノードのキャッシュの無効化をともなうライトミスや他のノードのキャッシュの属性変更をともなうリードミスなど，複雑な処理をともなうキャッシュ管理が頻繁に起こるためにHPCにヒットした場合の効果が相対的に小さくなるためである．

表8 リードミス時のミスペナルティ
Table 8 Miss penalty caused by read-miss.

ヒット/ミスヒット	ミスペナルティ（μs）
ヒット	1972
ミスヒット	2574

4. 考　　察

表2より，HPCによるミスペナルティの削減の効果は約10%にとどまっている．これは，HPCにヒットした場合には行われないSBusを経由したページデータのコピー（図4：処理b）の処理時間が約600 μsであるのに対し，送信されてきたページデータをネットワークインタフェース内のメモリに読み込む処理（図4：処理4）に約3 msを費やしているため，処理bの処理時間がミスペナルティに対して小さくなっているからである．今回実装に用いたネットワークインタフェースでは2 KBのデータを連続して送信できるが，受信時には1バイトずつデータの到着を確認しなくてはならない．また，STAFF-Linkは8ビット幅のリンクを持つので送受信時にデータのPack/UnPackをソフトウェアで行う必要があるため，リンク間の転送速度が140 Mbpsであるにもかかわらずノード間の実効転送速度が約10 Mbpsにとどまっている．

そこで，STAFF-Linkの代わりにMyrinetを用いた場合のミスペナルティの予測を行う．Myrinet用の通信ライブラリPMのノード間の実効転送速度が約256 Mbps[12]であることから，4096バイトのページデータの転送が約128 μsで行える．よって，ノード間のすべてのページデータの転送処理が2884 μs短縮できると思われる．

これを基に予測したリードミス時のミスペナルティを**表8**に示す．表8より，HPCにヒットした場合のミスペナルティ削減の効果は約30.5%となる．この場合，通信時間に対して処理bの処理時間が大きくなりI/Oバスのトラフィックを緩和するというHPCの有効性が示される．しかしながら，今回Home Proxyの機能を実装したDSPの性能に限界があるため，DSPがボトルネックとなると思われる．

行列演算，およびFFTを4ノードで行った場合のページデータの転送回数を計測し，演算時間の予測を行った結果をそれぞれ**表9**，**表10**に示す．ただし，“転送数”とは各ノードのキャッシュミスに起因して送信されたページデータ数である．表9，および表10よ

表9 4ノードでの行列演算の処理時間 (ms)
Table 9 Processing time to make product of each matrix using 4 nodes.

サイズ	転送数	HPC有	HPC無	性能比
128 × 128	128	362.04	471.67	1.30
256 × 256	256	5141.02	5320.78	1.03
512 × 512	512	43117.55	43338.93	1.01

表10 4ノードでのFFTの処理時間
Table 10 Processing time to perform FFT using 4 nodes.

データ数	転送数	HPC有	HPC無	性能比
4096	156	2285.020	2346.986	1.03
16384	520	5135.903	5233.417	1.02
65536	1978	16230.651	16443.531	1.01

り，行列演算，FFTのいずれの場合も性能が向上することが分かる．特に行列演算の場合，最大30%の性能の向上がみられた．しかしながらFFTの場合は大きな性能の向上はみられなかった．

本システムはメモリコンシステンシモデルにシーケンシャルコンシステンシを採用しているため，行列演算のような粒度の粗いアプリケーションは並列化の効果が上がるが，FFTのような細粒度のアプリケーションでは並列化を行うとノード間の通信が非常に頻繁に起こるために通信遅延による性能の劣化が起こる．よって，粗粒度のアプリケーションにおけるHPCの効果を上げるにはノード間通信にかかるコストを下げるだけでよいが，細粒度のアプリケーションの性能を上げるにはコンシステンシモデルの緩和やキャッシュコヒーレンシプロトコルの見直しが必要であることが分かる．リリースコンシステンシモデルや，無効化を行わずに更新によるキャッシュ管理を採用するなどの方法が考えられる．

5. おわりに

本論文ではWS/PCクラスタにおける分散共有メモリシステムの性能を向上させる手法として，ネットワークインタフェース上にホームメモリのI/Oバスの外側に対するキャッシュとしてHPCを提案し，実際に実装してその性能の評価を行った．その結果，ホームメモリに対する要求が生じる場合のミスペナルティが小さくなり，また，ホームメモリを管理するノードで動作する並列アプリケーションのプロセッサ利用率が上がることが分かった．しかしながら，FFTを用いた評価では並列度を上げると性能が落ちるという結果が出た．

HPCの効果をより大きなものにするにはミスペナルティの削減が必要であり，その方法としてMyrinet上の通信ライブラリPMを用いた場合についての考察を行った．その結果，HPCにヒットした場合のミスペナルティ削減の効果が約30%になることが分かった．また，行列演算についての性能予測から，粗粒度のアプリケーションにおいてHPCがより有効であることが分かった．しかしながら，細粒度のアプリケーションにおいては依然としてHPCの効果は小さく，アプリケーションの性能を上げるにはメモリコンシステンシモデルやキャッシュコヒーレンシプロトコルの改善などによる通信量の削減を行う必要がある．

その他の留意事項として，DSMLibraryやHome Memory ManagerがHome Proxyからの要求をチェックするルーチンは手動で並列処理アプリケーション内に埋め込まれていることがあげられる．本来なら自動化するべきだが，処理に時間のかかるループ内にルーチンを埋め込んだり，繰返し数の多いループは分割してその間にルーチンを埋め込むなどの処理が必要となるため自動化は難しく，今後の課題となる．

以上のような改善を施してソフトウェア的なシステムの高性能化を図り，そのうえでネットワーク，およびネットワークインタフェースにどのような要求が生ずるのかを突き止めていく予定である．

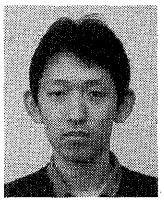
謝辞 ネットワークインタフェース上のプログラム開発環境には，TI社のUniversity Programを利用させていただきました．深く感謝いたします．また，この研究の一部の費用は，並列・分散処理研究推進機構により援助されたものです．

参考文献

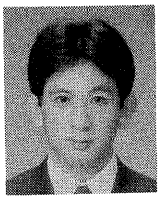
1) 天野英晴：並列コンピュータ，昭晃堂 (1996).
2) 中條拓伯，中野智行，松本 尚，小畑正貴，松田秀雄，平木 敬，金田悠紀夫：分散共有メモリ型超並列計算機JUMP-1におけるスケーラブルI/Oサブシステムの構成，情報処理学会論文誌，Vol.37, No.7, pp.1429–1439 (1996).
3) 薬師神昌夫，中條拓伯，金田悠紀夫：WSクラスタにおける機能分散ネットワークを用いたDSMシステムの実装，情報処理学会研究報告，Vol.96, No.80, pp.137–142 (1996).
4) Nakajo, H., Tanaka, H., Nakanishi, Y., Kohata, M. and Kaneda, Y.: Distributed Shared-Memory for a Workstation Clusters with a High Speed Serial Interface, *Proc. Int. Conf. and Exhibition on High-Performance Computing and Networking* (*HPCN98*), pp. 588–597 (1998).
5) 市川明弘，薬師神昌夫，中條拓伯，金田悠紀夫：高速シリアルリンクをDSMシステムの実装とその評価，情報処理学会研究会報告，ARC125-9,

pp.49–54 (1997).
6) Nakajo, H., Ichikawa, A. and Kaneda, Y.: An Implementation and Evaluation of a Distributed Shared-Memory System on Workstation Clusters using Fast Serial Links, *Proc. Int. Symp. on High Performance Computing* (*ISHPC*), pp.143–158 (1997).
7) 小野 航, 市川明弘, 安生健一朗, 山本淳二, 中條拓伯, 工藤知宏, 天野英晴：Home Proxy：WSクラスタにおける高性能DSMの実現, 並列処理シンポジウムJSPP'98論文集, pp.351–358 (1998).
8) Ono, W., Nakajo, H., Ichikawa, A., Anjo, K., Kudoh, T. and Amano, H.: Home Proxy Cache for High Performance DSM on a Workstation Cluster, *Proc. Int. Conf. Parallel and Distributed Processing Techniques and Applications* (*PDPTA'98*), pp.891–898 (1998).
9) Lamport, L.: How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs, *IEEE Trans. Comput.*, Vol.C-28, No.9 (1979).
10) 富田眞治：並列コンピュータ工学, pp.146–149, 昭晃堂 (1996).
11) Lenoski, D.E. and Weber, W.-D.: *Scalable Shared-Memory Multiprocessing*, Morgan Kaufmann (1995).
12) 手塚宏史, 堀 敦史, 石川 裕：ワークステーションクラスタ用通信ライブラリPMの設計と実装, 並列処理シンポジウムJSPP'96論文集, pp.41–48 (1996).

(平成10年9月4日受付)
(平成11年3月5日採録)



市川 明弘（学生会員）
昭和49年生．平成9年神戸大学工学部情報知能工学科卒業．平成11年同大学大学院自然科学研究科情報知能工学専攻博士前期課程修了．



小野 航
平成9年慶應義塾大学理工学部電気工学科卒業．現在同大学大学院理工学研究科計算機科学専攻修士課程在学中．WSクラスタの研究，開発に従事．

中條 拓伯（正会員）
昭和36年生．昭和60年神戸大学工学部電気工学科卒業．昭和62年同大学大学院工学研究科電子工学修士課程修了．平成元年神戸大学工学部情報知能工学科助手．博士（工学）．計算機アーキテクチャ，並列処理，並列入出力システムの研究に従事．電子情報通信学会会員．

工藤 知宏（正会員）
平成3年慶應義塾大学大学院理工学研究科博士課程単位取得退学．博士（工学）．東京工科大学情報工学科講師，助教授を経て，現在新情報処理開発機構所属．軽量通信・メモリアーキテクチャの開発に従事．

天野 英晴（正会員）
昭和56年慶應義塾大学工学部電気工学科卒業．昭和61年同大学大学院工学研究科電気工学専攻博士課程修了．現在，慶應義塾大学理工学部情報工学科助教授．工学博士．計算機アーキテクチャの研究に従事．