2006－OS－103 (5)
2006／7／31

DIMMnet-1を用いた分散共有メモリシステムにおけるページ転送方式の改良

中平勝人[i]，齋藤彰一[ii]，上原哲太郎[iii]，國枝義敏[iv]

i 和歌山大学大学院システム工学研究科 ii 和歌山大学システム工学部
iii 京都大学学術情報メディアセンター iv 立命館大学情報理工学部

DIMMnet を用いた分散共有メモリシステム用にページ転送方式の改良を提案し，ソフトウェア分散共有メモリシステム Fagus 上に実装した．DIMM メモリスロットに搭載されるタイプの NIC である DIMMnet は，主に PC クラスタ用として開発され，PCI バス搭載型 NIC の問題点を解決している．我々は，Fagus が採用している Entry Consistency モデルの特性を利用して並列計算の実行処理の流れを通信フェイズと計算フェイズに分離することで，現在動作している DIMMnet-1 版 Fagus のマルチスレッドを廃止し，より DIMMnet 環境に適した分散共有メモリシステムを実現した．

Improvement of a Page Transfer Method for a Distributed Shared Memory System using DIMMnet-1

Katsuhito NAKAHIRA[i], Shoichi SAITO[ii], Tetsutaro UEHARA[iii], Yoshitoshi KUNIEDA[iv]

i Graduate school of Systems Engineering, Wakayama University
ii Faculty of Systems Engineering, Wakayama University
iii Academic Center for Computing and Media Studies, Kyoto University
iv College of Information Science and Engineering, Ritsumeikan University

A new page transfer method is developed for a distributed shared memory system using DIMMnet. This page transfer method is implemented on software distribution shared memory system "Fagus". A DIMMnet board, which is specially designed to compose a PC cluster, is inserted in a DIMM memory slot of ordinary PCs and functioning as a kind of NIC. As using a DIMM memory slot (bus), it can solve several problems on communication overhead of an ordinary NIC on a PCI slot (bus). In this paper, we propose a new implementation method of "Fagus" especially on considering both its characteristic of "Entry Consistency model" and DIMMnet communication characteristics. Concretely, in this new implementation, first, communication phases are separated from calculation phases clearly in any parallel execution. Second, instead of using multi-threading for the implementation of the send/receive functions of original "Fagus", we propose a revised implementation by using single-threading.

1 はじめに

安価な PC を相互に接続して並列処理を行う PC/WS クラスタシステムは，コスト対パフォーマンス比に優れた並列計算環境である．近年，PC に使用されているプロセッサの理論演算能力の向上は著しく，十数年前のベクトル型スーパーコンピュータと同等の処理能力を有するものも出現している．したがって，今後もプロセッサの高性能化が進むことで，より処理能力の高い PC/WS クラスタシステムを実現することができると考えられる．

本研究で基盤として利用したソフトウェア分散共有メモリシステム Fagus[1]は，Entry Consistency モデル[2]（以下，**EC モデル**という）に基づき，PC/WS クラスタシステムをターゲットとして我々が開発したソフトウェアシステムである．通信には UDP/IP を使用しており，Linux 上で動作する．また，ユーザとコンパイラに対して，cc-COMA（compiler controlled Cache Only Memory Architecture）環境[3]を提供する．Fagus を用いて並列処理を実行する場合，ノードに対するデータ分散やアドレス管理をプログラマが意識する必要がない．そのため，MPI 規格による並列プログラムを記述する場合と比べて，並列計算プログラムを容易に作成することができる．

しかし，ネットワークインタフェースの接続に使用される PCI バスの性能では，通信帯域幅の限界がボトルネックとなり，高性能 PC の性能を最大限引き出して処理を行うことができていない．そこで，こうした従来の PCI バス搭載型ネットワークインタフェースの問題点を解消するため，DIMM スロットに搭載するネットワークインタフェースである DIMMnet[4,5]が開発された．DIMMnet は DIMM スロットにインタフェースカードを搭載することで，PCI バスの通信帯域幅を大幅に越えることに成功している．

本稿では，我々のグループで開発した DIMMnet-1 版 Fagus（以下，**D1-Fagus** という）[6]を基にして新たに開発した DIMMnet-1 版 Fagus2（以下，**D1-Fagus2** という）と，EC モデルの特長を利用した DIMMnet 環境に適したページ転送方式について述べる．D1-Fagus では，その前身である UDP 版 Fagus の特長であるマルチスレッド方式を踏襲している（**図 1** 参照）．マルチスレッド方式では，ユーザプログラムを実行するスレッドと通信系統の動作を実行するスレッドが並行して動作することで処理効率を高めている．しかし DIMMnet で通信を行う場合，通信処理スレッドのポーリングが CPU の処理能力を著しく消費する．結果，ユーザスレッドの演算処理能力に悪影響を与えることが確認された．本研究では，Fagus が採用している EC モデルの特性を利用して，処理を演算実行区間と通信実行区間に分離した．これにより，マルチスレッドを廃止し，ポーリングによる演算処理能力への影響をなくした．

以下，2 章では本稿のシステム開発に使用したネットワークインタフェース DIMMnet について述べる．3 章ではソフトウェア分散共有メモリシステム Fagus の概要を述べる．4 章では D1-Fagus について述べる．5 章ではマルチスレッドを廃止した D1-Fagus2 と，新しいページ転送方式について述べる．6 章では実装したシステムの評価方法とその結果，考察を述べ，7 章では本稿のまとめを述べる．

2 DIMMnet

DIMMnet は DIMM スロットに搭載する NIC で，

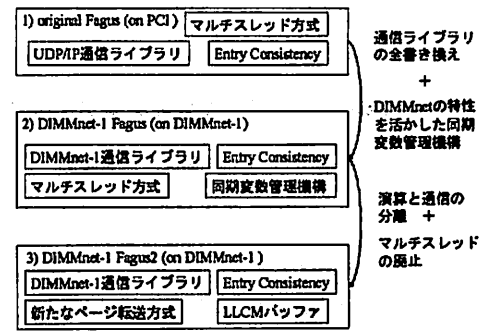

図 1：Fagus の改変/バージョンアップの概要

主に PC クラスタでの使用を考慮して開発された．従来の PCI バスに搭載される NIC では，PCI バスのバンド幅や遅延時間の限界によりネットワークや高性能 PC の本来の性能が発揮されない問題が生じる．一方，メモリバスに直接装着される NIC である DIMMnet は，高バンド幅，低遅延である．したがって，上述の問題を軽減すると期待できる．また，メモリバスは，CPU の性能向上に追随してバンド幅が向上すると考えられることから，NIC 搭載に最適なインタフェースと言える．

本研究で使用したネットワークインタフェースである DIMMnet-1 は PC66，PC100，PC133 仕様の DIMM スロットに搭載することが可能である．DIMMnet-1 は 128KB の低遅延共有メモリ（**LLCM**：Low Latecy Common Memory）と 64MB～1GB の SO-DIMM を備える．これらのメモリは通常のメモリと同様に，ユーザプロセスからのアクセスが可能である．DIMMnet-1 は，受信したデータをホスト CPU を介することなく，これらのメモリに書き込むことができる．つまり，DIMMnet でのデータ送信は，リモートメモリに対して直接書き込みを行うことと同義である．これは，受信側の処理コストを削減できる．DIMMnet-1 は PCI バス搭載型 NIC と比較して，非常に高い通信バンド幅と優れた低遅延性を示した[4,5]．

DIMMnet には，低遅延通信の特長をもつ **AOTF**（Atomic On-The-Fly）[4]と，高バンド幅通信の特長をもつ **BOTF**（Block On-The-Fly）[5]の 2 つの通信プロトコルがある．AOTF は，ユーザモードのまま DIMMnet 上のメモリに対してデータの書き込みを行うことで，1～8 バ

イトのデータ送信を起動することができる．BOTF はユーザプロセスがパケットを作成し，464 バイト以下の連続したデータを送信することができる．

3 ソフトウェア分散共有メモリシステム Fagus

ソフトウェア分散共有メモリシステム Fagus は，cc-COMA 環境を実現するソフトウェアシステムである．我々が開発中である自動並列化コンパイラ MIRAI[3]の実行時環境として開発された．Fagus の主な特長として，自動並列化コンパイラでの利用を考慮した命令群や UNIX 系の OS であれば容易に移植可能な汎用性が挙げられる．cc-COMA とは，ソフトウェアで COMA 環境を創り出し，共有メモリの一貫性制御をコンパイラに委ねる並列環境である．

3.1 一貫性制御方式

単純な COMA 環境では共有メモリの一貫性制御のために頻繁にデータ転送が発生し，実行性能が低下するおそれがある．そのため，Fagus ではより弱い整合性モデルである EC モデルを採用している．EC モデルでは同期変数と共有変数を予め関連付ける．ユーザプログラム内で共有変数を参照するに際には，関連付けた同期変数の獲得と解放を行うことで，共有変数の排他制御と更新を行う．Fagus 利用時には，まずコンパイラが同期変数と共有変数の関連付けを行う．その上で，共有変数の書き込みと読み出しの前後に，コンパイラが関連付けた同期変数の獲得と解放の処理を行うコードを埋め込む．以上により，コンパイラによる解析に基づく一貫性制御を実現する．

3.2 マルチスレッド方式

UDP 版 Fagus と D1-Fagus では，並列プログラムを実行するユーザスレッドに並行して，通信関連のスレッドを動作させる．こうしたマルチスレッド方式をとることで通信効率の向上を図っている．他ノードへのメッセージ送信は，ユーザスレッドが送信処理を送信スレッドに依頼することで行う．その後，依頼したユーザスレッドは本来の計算処理に戻る．また他ノードからの同期変数の要求などの通信に対しては，受信スレッドが相手の要求メッセージを受信し，その要求に応じた処理を実行する．以上の方式により，ユーザスレッドが並列プログラムの実行に専念することを可能としている．

4 DIMMnet-1 版 Fagus (D1-Fagus)

D1-Fagus は，UDP 版 Fagus を基に，DIMMnet 上で動作させるために，DIMMnet-1 用の通信ライブラリを新たに実装したものである．また，DIMMnet の特性を利用して実行処理のコスト削減を工夫している．以下，D1-Fagus の特長について述べる．

4.1 D1-Fagus の特長

UDP 版 Fagus の通信ライブラリは UDP/IP で信頼性のある通信を実現するために，UDP/IP に特化した実装がなされている．しかし，DIMMnet は独自のプロトコルである AOTF と BOTF を通信プロトコルとして使用しているため，UDP 版 Fagus の通信ライブラリをそのまま DIMMnet に対応させるのは困難である．そのため，D1-Fagus では，新たに DIMMnet-1 用の通信ライブラリを UDP 版 Fagus のライブラリインタフェースを継承して開発し，共有変数の一貫性制御方式やマルチスレッド方式といった特長も UDP 版 Fagus から継承した．D1-Fagus の独自機能としては，DIMMnet-1 の LLCM に同期変数管理情報を配置し，他のノードの同期変数を直接書き変えることにより，効率的な同期変数管理を行うことを可能としている．

4.2 マルチスレッド方式とデータ受信

D1-Fagus では，UDP 版 Fagus と同様に，ユーザプログラムを実行するユーザスレッドとそれに並行して通信関連系スレッド（送信スレッド，受信スレッド，受信処理スレッド）が動作している．UDP 版 Fagus との主な違いは，他ノードからのデータを受信する受信スレッドの動作である．DIMMnet-1 では，データ受信割り込みが発生しないため，受信領域にデータが書き込まれたかどうか確認するためにポーリングを行う必要がある．しかし，

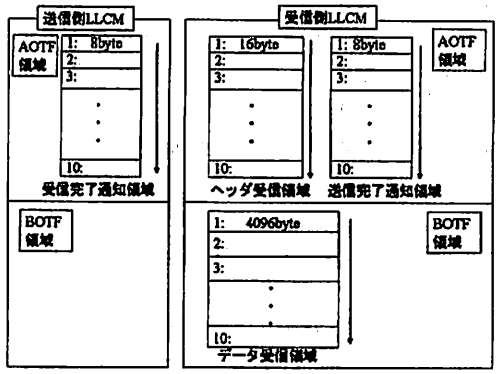

図2：LLCMバッファ構造

このポーリングをマルチスレッド方式で行うと，常に受信スレッドのポーリングがCPUの半分近くを消費する．これにより，ユーザスレッドの演算処理能力が半減する．この問題を避けるため，D1-Fagusでは，データの送信完了を，TCP/IPで通知することで受信スレッドのポーリングコストを削減して，問題を軽減している．

5 DIMMnet-1版Fagus2(D1-Fagus2)

D1-FagusがもちいたTCP/IPによる送信完了通知は，DIMMnetの通信速度の向上を想定した場合，ボトルネックになる可能性が高い．本研究では，TCP/IPを使用しない通信方式を提案し，実装した．この方式を実装したFagusをDIMMnet-1版Fagus2(D1-Fagus2)という．以下，D1-Fagus2の通信方式について述べる．

5.1 通信方式の概要

D1-Fagus2では，Fagusが採用しているECモデルの特性を利用して，プログラムの実行処理の流れを，演算処理のみを行う区間（**計算フェイズ**）と通信処理のみを行う区間（**通信フェイズ**）に分離している．マルチスレッド方式では，ユーザスレッドが演算をしている間も，絶えず受信スレッドがポーリングを実行していた．そのため演算にCPUの全処理能力を使用できなかった．しかし，演算と通信の処理を明確に分離することで，演算処理中に送受信のためのポーリング処理の影響を受けない仕組みを実現した．また，通信効率を向上させるため，AOTFとBOTFの受信領域にバッファ機能を実現した（詳細は次節で述べる）．D1-Fagus2における使用上の制約として，同期変数の獲得と共有変数の更新は，全ノードが一斉に実行しなければならない点がある．そのため，より高度な並列プログラミングの知識を要求することになる．Fagusは，cc-COMA環境を提供しているので，ユーザが並列化コンパイラを使用することでこの問題を無視することができる．

5.2 LLCMバッファリング

D1-Fagus2では，通信フェイズ時の送受信の待ち時間を削減するために，DIMMnet-1のLLCMに，独自のバッファを構成した（**図2参照**）．このバッファはリングバッファであり，各ノードと一対一で対応している．つまり，並列計算に参加するノードの台数分，図2と同様の領域一式を必要とする．図2の各領域の番号は，他の領域の番号と関連付けられている．例えば，受信したヘッダの情報は，同番号のデータ受信領域に受信したデータと対応している．また，それらのデータの送受信を他ノードに知らせるために，同番号の受信完了通知領域と送信完了通知領域を使用する．なおヘッダ受信領域で受信するヘッダとは，Fagusが持つ独自のヘッダ（以下，**Fagusヘッダ**という）で，送信ノードの情報や送信したデータの種類が格納されている．以下，各領域の役割について述べる．

5.2.1 送信ノードのバッファ領域

受信完了通知領域とは，受信ノードがデータの受信を完了したことを送信ノードに伝えるために用いる領域である．DIMMnetの送信は，送信ノードが受信ノードのリモートメモリに対して直接書き込むため，受信ノードが受信前のデータを上書きしないように，受信ノードの状態に注意する必要がある．受信ノードは受信を完了したデータについて，AOTFを用いて送信ノードの受信完了通知領域に受信完了を通知する．送信ノードは，この領域を確認し，次の送信時に，受信完了となっているデータ受信領域にデータを書き込む．

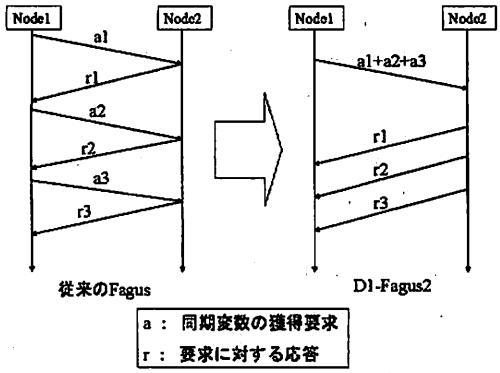

図3：通信方式の違い

5.2.2 受信ノードのバッファ領域

バッファ領域には，ヘッダ受信領域とデータ受信領域と送信完了通知領域の3種類の領域がある．以下それぞれの役割について述べる．

(1)ヘッダ受信領域

Fagusヘッダを書き込む16byte単位の領域である．AOTFを用いて送信される．

(2)データ受信領域

共有変数のデータを書き込む4096byte単位の領域である．BOTFを用いて送信される．

(3)送信完了通知領域

DIMMnetでは受信割り込みが発生しないため，ポーリングによってデータの送信完了を確認する．送信ノードはAOTFを用いて，受信ノードの送信完了通知領域にデータの送信完了を伝える．受信ノードは送信完了通知領域をポーリングすることでデータの送信完了（受信）を確認する．

5.3 演算と通信の分離

一般的に多くのDSMシステムで採用されているRelease Consistencyモデルでは，ページフォルトでページ参照を検出し，その都度ページ要求を行う．このため，ページを供給するノードは，常にページ要求を受信できなければならない．しかし，受信割り込みのないDIMMnetで，演算処理中にページ要求を受信するためにはポーリングが必要である．そのため，CPUが演算処理の実行に集中できず，処理時間が増加していた．

一方，Fagusが採用しているECモデルは臨界領域の実行を開始する時点で一貫性制御を行う．このため，臨界領域内でどの共有変数を参照するかを事前に知ることができる．これによりデータが必要になった時点で，逐一通信を行うのではなく，あらかじめ必要なデータを先読みし，それらをまとめて通信を行うことが可能である．D1-Fagus2では，この特性を利用して，プログラムを演算のみを行う計算フェイズと通信のみを行う通信フェイズに分離している．具体的には，同期変数の獲得や共有変数の更新が通信フェイズに該当する．また，通信フェイズに発生するデータ要求を逐次実行するのではなく，その通信フェイズ中のすべてのデータ要求を一括してあらかじめ送信する．これにより，従来のFagusと比べて通信回数を減少させた(**図3**参照)．以下に実行処理の流れを述べる(**図4**参照)．なお，図4中の（番号）は以下の各文の番号と対応している．

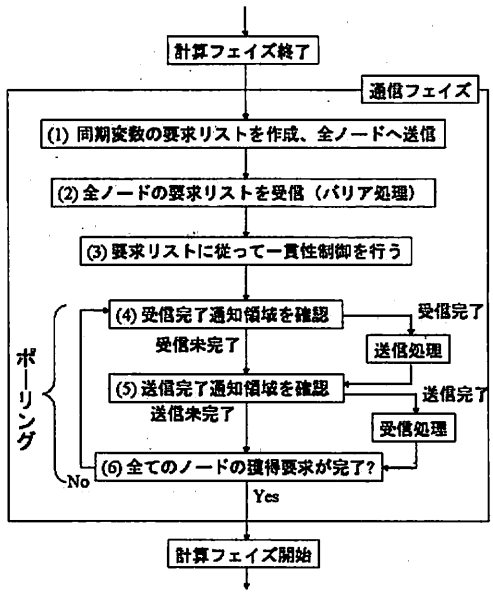

図4：通信フェイズの実行手順

(1) ある計算フェイズが終了したい，次の計算フェイズで自ノードが必要とする同期変数の要求リストを作成し，全ノードに送信する．DIMMnetの通信プロトコルは相手ノードのメモリの受信領域に直接書き込めるため，他ノードが通信フェイズに入ることを待たずに送信を開始できる．

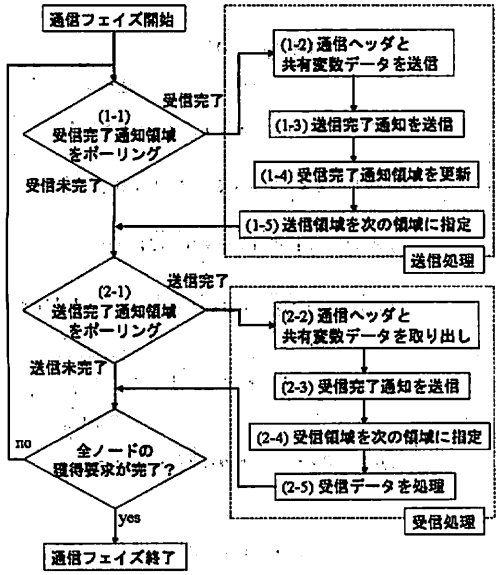

図5：送受信の手順

(2) 全ノードからの要求リストを受信するまで，待ち状態となる．

(3) 要求リストに基づいて，同期変数の獲得や状態の変更を行い，共有変数の送受信リストを作成する．

(4) 受信完了通知領域のポーリングを開始する．受信完了していれば，送信処理を行う．

(5) 送信完了通知領域のポーリングを開始する．送信完了していれば，受信処理を行う．

(6) 自ノードが送受信しなければならない獲得要求をすべて完了後，通信フェイズを終了し，計算フェイズに移行する．この時，他ノードの状況に関係なく計算フェイズに移行できる．

5.3.1 データ送信手順

D1-Fagus2 の通信フェイズでのデータ送信手順について述べる(**図5**参照)．なお，図5中の(番号)は以下の各文の番号に対応する．

(1-1) 送信ノードは，受信完了通知領域をポーリングし，書き込み可能なら送信処理を開始する．

(1-2) Fagus ヘッダを作成し，AOTF を用いて受信ノードのヘッダ受信領域に書き込む．

(1-3) Fagus ヘッダ送信後，BOTF を用いて共有変数のデータを書き込む．

(1-4) データ書き込み後，AOTF を用いて送信完了通知を受信ノードの送信完了通知領域に書き込む．さらに受信ノードのデータ上書きを防止するため，受信完了通知領域の値を未完了に変更する．

(1-5) 書き込みバッファ領域を次のバッファ領域に指定する．

5.3.2 データ受信手順

D1-Fagus2 の通信フェイズでのデータ受信手順について述べる(**図5**参照)．なお，図5中の(番号)は以下の各文の番号に対応する．

(2-1) 送信完了通知領域をポーリングし，送信完了通知を確認すれば，受信処理を開始する．

(2-2) AOTF 受信領域のヘッダ受信領域から通信ヘッダ，BOTF 受信領域のデータ受信領域からデータをそれぞれ取り出す．

(2-3) データの受信が完了し，バッファが空になりデータ書き込みが可能になったことを通知するため，AOTF を用いて送信ノードの受信完了通知領域を更新する．

(2-4) 読み込みバッファ領域を次のバッファ領域に指定する．

(2-5) 受信した通信ヘッダの要求に従ったデータ処理を行う．

6 評価

D1-Fagus と D1-Fagus2 の性能比較について述べる．評価環境は，Intel PentiumIII 950MHz Dual，1.5GB メモリ，100BaseTx Ethernet，DIMMnet-1(250MHz)，Linux-2.4.2 が2台である．

6.1 評価方法

行列 A，B，C を用意し，A と B の積を C に書き込む行列積のプログラムを用いて評価比較を行う．評価プログラムのソースコードの概要を**図6**に，データ分割の様子を**図7**に示す．今回の実験では 512×512，1024×1024，2048×2048 の3つの行列を用いて評価を行った．

評価プログラムの fagus_からはじまる関数は Fagus が提供する API である．1台目のノードが奇数行，2台目のノードが偶数行をそれぞれ計算する．1行目の fagus_set_sync()

```
1: fagus_set_sync(sync_set, sync, mode);    同期変数の詰め込み
2: fagus_acquire(sync_set);                 同期変数の獲得
3: for(i=0; i<配列の要素数; i++){
4:   for(j=0; j<配列の要素数; j++){
5:     for(k=0; k<配列の要素数; k++){
6:       c[i][j] = a[i][k] * b[k][j];       行列積演算
7:     }
8:   }
9: }
10: fagus_release(sync_set);                同期変数の開放
11: fagus_sync_updatecache();               共有変数の更新
```

図6：評価プログラムのソースコード

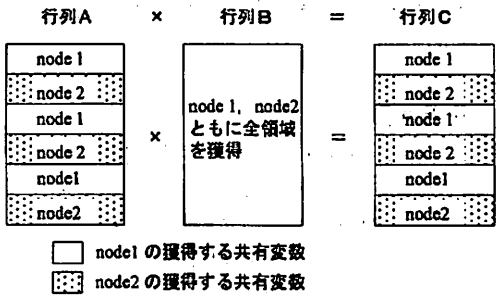

図7：評価プログラムのデータ分割の様子

で，自ノードが計算フェイズに必要とする同期変数の獲得要求リストを作成する．作成した同期変数の獲得要求リストを引数にして，2行目の fagus_acquire() で同期変数の一括獲得を行う．このとき通信フェイズとなり，fagus_acquire()の内部で実行される．通信フェイズ終了後，行列積の演算を実行する計算フェイズ（3行目から9行目）が開始される．演算終了後，獲得した同期変数の一括解放を10行目の fagus_release()で行う．11行目の fagus_sync_update()で共有変数を更新（他ノードへ，更新されたデータの送信）するため，再び通信フェイズに入る．なお，D1-Fagus では，同期変数の一括獲得ができないため，行列積の計算に入るループの前後で，個々に同期変数の獲得と解放を実行している．

6.2 評価結果

処理全体の実行時間を**図8**，通信のみの実行時間を**図9**にそれぞれ示す．図中の記号AはD1-Fagus，BはD1-Fagus2の実行結果を指している．

全体の実行結果として，扱うデータの量が少ない場合，実行時間はほぼ同じ結果である．

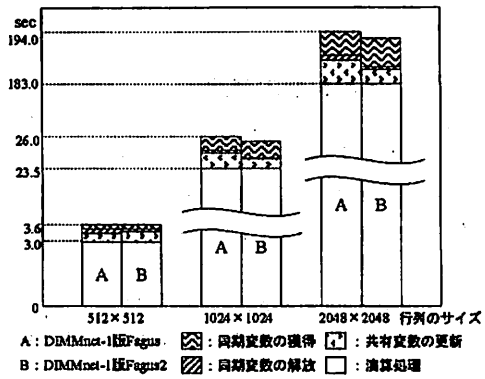

図8：並列計算の評価結果

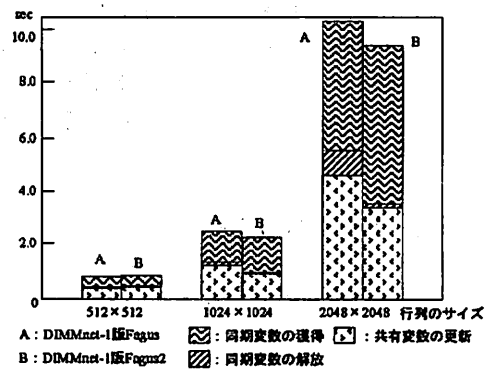

図9：通信の評価結果

しかし，行列の大きさが大きくなるにつれて，D1-Fagus2の実行速度がD1-Fagusより高速となる結果となった．通信時間のみを比較した場合，D1-Fagusと比較して，同期変数の獲得の実行速度がやや遅く，同期変数の解放と共有変数の更新での実行速度が速くなっている．

6.2.1 同期変数の獲得

図6の2行目 fagus_acquire()による同期変数の獲得の評価結果について考察する．今回の評価実験では，行列Bを初期化する通信が，同期変数の獲得時に発生する通信の大半を占めている．この通信では，行列Bの所有者であるノード1が送信のみを行い，行列Bを要求したノード2が受信のみを行う．D1-Fagus2では，無駄なポーリングを抑えるため，送受信のうち即時に実行できる処理を優先して行う．しかし，今回のように一方が送信処理のみ，もう一方が受信処理のみを繰り返す通信の場合，ポーリング時間を減らせ

ず，実行速度が遅くなったと考えられる．

6.2.2 同期変数の開放

図6の10行目fagus_release()による同期変数の開放の評価結果について考察する．同期変数の開放では，D1-Fagusと比較して，速度向上が見られた．D1-Fagus2では，計算フェイズと通信フェイズの分離によって，すべての通信を同期変数の獲得時にまとめている．そのため，同期変数の解放時に新たに通信が発生することがない．しかし，D1-Fagusでは同期変数の開放時に，同期変数の状態を他ノードに伝えるための通信が発生する．以上のことから，結果の差は，同期変数の解放時にも通信を行うD1-Fagus と獲得時のみ通信を行うD1-Fagus2との差であると考えられる．

6.2.3 共有変数の更新

図6の11行目fagus_sync_update()による共有変数の更新の評価結果について考察する．共有変数の更新では，D1-Fagus2 の実行速度が，通信するデータ量が増加するにつれて，D1-Fagusを上回る結果となった．D1-Fagus2では，共有変数の獲得と同様に，更新が必要な共有変数の更新要求を一括して行う．しかし，D1-Fagusでは，更新が必要な共有変数1つに対して，更新要求を1つ送っている．そのためD1-Fagus2と比較して通信回数が増加する．その結果が実行速度の差に現れたものと考えられる．

7 おわりに

本稿では，DIMMnet-1の通信機構を用いたページ転送方式と，EC モデルを利用した通信処理と計算処理の分離の仕組みについて述べた．D1-Fagus2は，獲得する共有変数の大きさが増加するにつれ，D1-Fagusより高速に動作した．また新たに実装したページ転送方式により，TCP/IPを使用せずにポーリングコストを抑えた通信機構を実現した．具体的には，計算フェイズと通信フェイズを完全に分離したことで，一括した通信が可能となった．特に，細かな通信が多発するケースにおいて，通信のとりまとめの効果が顕著となり，通信時間が短縮される結果になった．

今後は，ポーリングコストのさらなる削減と，DIMMnet-1の内蔵メモリであるSO-DIMMを用いた，より規模の大きな一括通信を行う．これにより，さらなる高速化を図る予定である．それに加えて，DIMMnet-1の後継として開発されているDIMMnet-2上で動作するDIMMnet-2版Fagusの開発を予定している．

謝辞 本システムの開発は，総務省戦略的情報通信研究開発制度の一環として行われたものである．

参考文献

[1] 横手聡，齋藤彰一，上原哲太郎，國枝義敏："コンパイラによる制御が可能なDSMシステム「Fagus」の実現"，情報処理学会研究会報告 2000-OS-85, Vol.2000, No.75, pp.47-54(2000).

[2] Bershad, B.N., Zekauskas, M.J. and Sawdon, W.A.: "The Midway Distributed Shared Memory System", Proc. of COMPCON '93, pp.528-537(1993).

[3] Tetsutaro UEHARA, Tsuneo NAKANISHI, Masaaki MINEO, Shoichi SAITO, Kazuki JOE, Akira FUKUDA, Yoshitoshi KUNIEDA: "MIRAI: Automatic Parallelizing and Distributing Compiler based on cc-COMA approach", Proc. of Int. Conf. on PDPTA'2001, Vol.III, pp.1193-1199(2001).

[4] 田邊昇，濱田芳博，山本淳二，今城英樹，中條拓伯，工藤知宏，天野英晴："DIMMスロット搭載型ネットワークインタフェースDIMMnet-1とその低遅延通信機構AOTF"，情報処理学会論文誌，Vol.44, No.SIG 1(HPS 6), pp.10-22(2003).

[5] 田邊昇，山本淳二，濱田芳博，中條拓伯，工藤知宏，天野英晴："DIMMスロット搭載型ネットワークインタフェースDIMMnet-1とその高バンド幅通信機構BOTF"，情報処理学会論文誌，Vol.43, No.SIG4 pp.866-878 (2002)

[6]笠松領介，齋藤彰一，上原哲太郎，國枝義敏："DIMMnetによる分散共有メモリシステムの同期変数管理機構の開発"，情報処理学会研究報告，2005-ARC-164, pp.115-120(2005).