社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－ARC－164 (20)
2005／8／4

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

笠松領介[†], 齋藤彰一[††], 上原哲太郎[†††], 國枝義敏[††††]

[†]和歌山大学大学院システム工学研究科
[††]和歌山大学システム工学部
[†††]京都大学大学院工学研究科附属情報センター
[††††]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新たな同期変数管理機構を開発し，ソフトウェア分散共有メモリシステムFagus上に実装した．DIMMnetは，DIMMメモリスロットに搭載されるNICである．DIMMnetは主にPCクラスタ用として開発され，従来のPCIバス搭載型NICの問題点を解決している．我々は，Fagusの通信方式を従来のUDP/IPからDIMMnetのAOTFとBOTFに変更した．さらに，DIMMnetの低遅延共有メモリ上にFagusの管理データを配置することで，同期変数の管理コストの低減を実現した．

Development of a Synchronization Variable Control System on a Distributed Shared Memory System using DIMMnet

Ryosuke KASAMATSU[†], Shoichi SAITO[††], Tetsutaro UEHARA[†††], Yoshitoshi KUNIEDA[††††]

[†]Graduate school of Systems Engineering, Wakayama University
[††]Faculty of Systems Engineering, Wakayama University
[†††]Center for Information Technology, Graduate School of Engineering, Kyoto University
[††††]College of Information Science and Engineering, Ritsumeikan University

A new synchronization variable control system is developed for a distributed shared memory system using DIMMnet. This synchronization variable control system is implemented on software distribution shared memory system Fagus. DIMMnet is a NIC installed in a DIMM memory slot. DIMMnet is developed for the PC cluster, and can solve problems of a NIC loaded in a PCI bus slot. We changed the communication method of Fagus from UDP/IP to AOTF and BOTF of DIMMnet. Furthermore, a management cost of synchronization variables has been decreased by arranging management data of Fagus on a LLCM of DIMMnet.

1 はじめに

安価で高い処理能力を有する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おいて優れた並列計算環境である．近年のPCの処理能力は，プロセッサの理論演算能力のみに注目した場合，十数年前のベクトル型スーパーコンピュータに比肩する能力を持つものも珍しくない．したがって，今後もプロセッサの高性能化が進むことにより，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．

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]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．FagusはUDP/IPを使用してLinux上で動作し，cc-COMA(compiler controlled-Cache Only Memory Architecture)環境[3]をユーザとコンパイラに提供する．Fagusを用いた場合，業界標準であるMPI規格に基づいた並列プログラムを記述する場合と比較して，ノードに対するデータ分散やアドレス管理などをプログラマが意識する必要が無いため，容易に並列計算プログラムを作成することができる．

しかし，従来のPCIバスに接続されるネットワークインタフェースでは，PCIバスの性能の限界がボトルネックとなり，昨今の高性能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，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はインタフェースカードをDIMMスロットに搭載することにより，PCIバスを大幅に越える通信帯域幅を利用することができる．

本稿では，現在稼動しているDIMMnet-1を用いたFagusの開発と，特にDIMMnet-1の特長を利用した同期変数管理機構と，それによるFagusの性能向上について述べる．ソフトウェアDSMシステムにおいて，共有変数に関連付けられた同期変数の管理は重要である．どのノードがどの同期変数を所有しているか，という情報はシステム内で一意に管理されなければならない．しかし，一般的に並列処理を行うプログラムは，多大な量のデータを用いた大量の計算処理を行うため，同期変数の管理のための

ネットワーク資源やCPU処理能力資源の利用は最低限に抑える必要がある．本研究では，DIMMnet-1が提供する仮想的な共有メモリ空間を用いることにより，同期変数管理のための処理コストの低減を図る．

以下，2章では本稿のシステムを開発するにあたって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章では本システムの前身にあたるUDP版Fagusについて述べる．5章ではDIMMnetの機能を用いて新たに実装したDIMMnet版Fagusと，新しい同期変数管理機構について述べる．6章では実装したシステムの評価方法とその結果，考察を述べる．7章では本稿のまとめを述べる．

2 DIMMnet

DIMMnetはPCクラスタ用に開発された，DIMMスロットに搭載されるNICである．従来のPCIバスに搭載されるNICは，PCIバスのバンド幅や遅延時間の限界により性能に制約を受ける．しかし，DIMMnetが搭載されるメモリバスは，PCIバスと比較して高バンド幅，低遅延であることから，従来のPCIバス搭載型NICの限界を大幅に超越した性能をもつNICを開発することができる．また，将来的にもメモリバスは，ムーアの法則による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の搭載において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じように，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．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はAOTF(Atomic On-The-Fly)[4]と呼ばれる低遅延通信機構と，BOTF(Block On-The-Fly)[5]と呼ばれる高バンド幅通信機構を備えている．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．リモート書き込み時間270ns，実測ラウンドトリップタイム1.84μsを示す，低オーバヘッドな送信アーキテクチャである．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．単方向通信継続バンド幅1017MB/s，双方向通信バンド幅2034MB/sを示す，高バンド幅の送信アーキテクチャである．

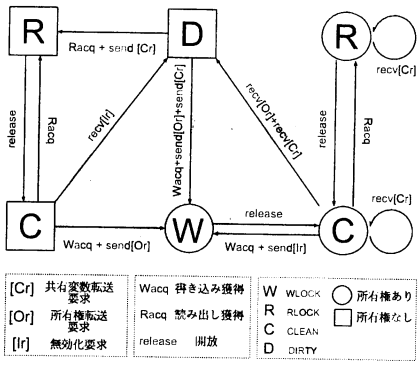

図1: 同期変数の状態遷移図

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，我々が開発中の自動並列化コンパイラMIRAI[3]の実行時環境として開発した，cc-COMA環境を実現するソフトウェアシステムである．Fagusは自動並列化コンパイラで利用しやすい命令群，UNIX系のOSであれば容易に移植可能な汎用性などを主な特長とする．cc-COMAとは，COMA環境をソフトウェアでエミュレートし，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のデータ転送が頻発し，実行性能が低下するおそれがあるため，Fagusではより弱い整合性モデルであるEntry Consistency(EC)モデル[2]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して，同期変数の獲得と解放の処理を実行することで，共有変数の排他制御と更新を行う．Fagusでは，並列計算を実行する前準備として，同期変数と共有変数の関連付けをコンパイラもしくはユーザに要求する．さらに，共有変数の書き込みと読み出しを行う場合には，随時同期変数の獲得と解放の処理を実行させる．これにより，コンパイラはECモデルに沿ったメモリ一貫性の下で並列計算を行う並列プログラムを作成することができる．

3.2 同期変数管理機構

Fagusは，それぞれのノードが管理する同期変数の状態によって共有変数の制御を行う．同期変数には，DIRTY，CLEAN，RLOCK，WLOCKの4状態がある．各状態の遷移図を図1に示す．WLOCKに遷移することができるのは同期変数の所有権を有するノードのみである．所有権は同期変数毎にFagus全体で同時には1台のノードだけが有する．所有権を有するノードは所有者と呼ばれる．所有権を有しないノードがWLOCKに遷移する場合には，所有者に対して所有権を要求する必要がある．これによ

り，同期変数の排他制御を実現している．以下，同期変数の 4 状態について述べる．

(1) WLOCK
同期変数を書き込み獲得している状態である．関連付けられた共有変数の書き込みと読み出しを行うことができる．書き込みの排他制御が行われており，他のノードは WLOCK と RLOCK に遷移することができない．
(2) RLOCK
同期変数を読み出し獲得している状態である．関連付けられた共有変数を読み出すことができる．読み出しの排他制御が行われており，所有者は WLOCK に遷移することはできるが，解放することはできない．
(3) CLEAN
同期変数を獲得していないが，関連付けられた共有変数は有効な状態である．この状態から同期変数を獲得する場合は，共有変数を更新する必要が無い．
(4) DIRTY
同期変数を獲得しておらず，関連付けられた共有変数が無効な状態である．この状態から同期変数を獲得する場合は，共有変数を更新する必要がある．

4 UDP 版 Fagus

UDP 版 Fagus は通信プロトコルとして UDP/IP を使用することにより，TCP/IP を使用する一般的な分散並列処理環境と比較して高速な通信と低コストなマルチキャスト通信を実現している [1]．UDP/IP はパケットの到達が保証されないことから，UDP 版 Fagus は ACK メッセージを用いてパケットの到達保証を実現している．

4.1 同期変数管理機構

UDP 版 Fagus は同期変数を個々のノードで管理し，必要な場合にはマルチキャスト送信を行って同期変数の書き換えを要求する．また，所有者を変更する際にも，UDP のマルチキャスト送信により，すべてのノードに対して同期変数の更新を要求する．同期変数の更新を要求されたノードは，CPU を使用して明示的に要求の処理を行う必要がある．

4.2 WLOCK に遷移する手順

UDP 版 Fagus の WLOCK に遷移する手順を述べる (図 2 参照)．なお，図 2 中の (番号) は以下の各文の番号に対応する．

(1) WLOCK に遷移する同期変数の所有権の有無を確認する．無ければ所有者に所有権の要求を UDP 送信して待つ．
(2) 所有権の要求を受信した所有者は，その同期変数を獲得しているときには，要求をバッファリングする．バッファリングされた要求は，その同期変数が解放されたときに実行される．
(3) 所有者はコピーセットと呼ばれるビットベクトルを調べ，所有権を要求してきたノードの共有変数が無効かどうか確認し，無効である場合は共有変数を UDP 送信する．

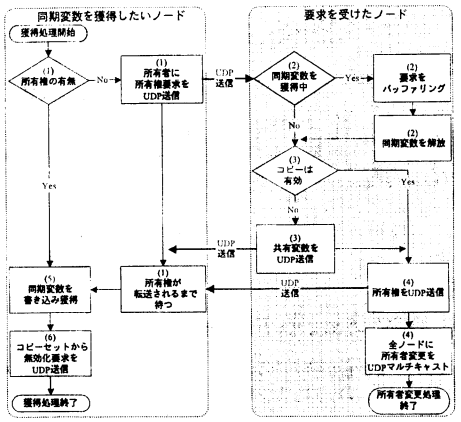

図 2: UDP 版 Fagus の書き込み獲得の手順

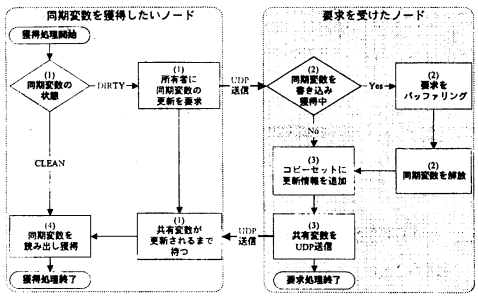

図 3: UDP 版 Fagus の読み出しの手順

(4) 所有権を UDP 送信する．このとき，すべてのノードに対して同期変数の所有者が変わったことを UDP マルチキャストで通知する．
(5) 所有権を受け取った新たな所有者は，WLOCK に遷移する．ここで書き込み獲得が成立する．
(6) コピーセットを調べて，有効な共有変数を持っているすべてのノードに対して無効化要求を UDP 送信する．

4.3 RLOCK に遷移する手順

UDP 版 Fagus の RLOCK に遷移する手順を述べる (図 3 参照)．なお，図 3 中の (番号) は以下の各文の番号に対応する．

(1) RLOCK に遷移する同期変数の状態を確認する．DIRTY であれば所有者に共有変数の更新の要求を UDP 送信して待つ．
(2) 更新の要求を受信した所有者は，その同期変数を書き込み獲得しているときには，要求をバッファリングする．
(3) 所有者は共有変数を UDP 送信し，コピーセットに記録する．
(4) 共有変数が更新されたら，RLOCK に遷移する．ここで読み出し獲得が成立する．

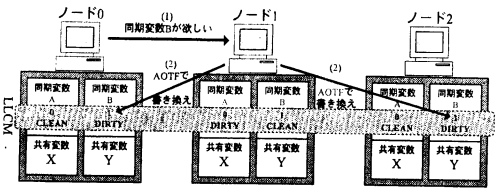

図 4: LLCM 上の同期変数の概念図

5 DIMMnet 版 Fagus

UDP 版 Fagus は通信プロトコルとして UDP/IP を使用するため，独自のプロトコルである AOTF と BOTF をもつ DIMMnet を使用することはできない．また，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 に特化したコーディングがなされているため，UDP 版 Fagus の通信ライブラリを DIMMnet に対応させるのは困難である．そのため，新たな DIMMnet 版 Fagus 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た．

DIMMnet 版 Fagus は，DIMMnet 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特長とする．

5.1 同期変数管理機構

DIMMnet 版 Fagus の同期変数の獲得と解放の手順は UDP 版 Fagus とは異なる．DIMMnet 版 Fagus は同期変数のメンバのうち，その所有者を記録する変数と状態を記録する変数を DIMMnet の LLCM 上に配置する．LLCM 上に配置した同期変数の概念図を図 4 に示す．

LLCM の容量は 128KB と小さく，同期変数のすべてを LLCM 上に配置することができないことから，同期変数の管理上重要となる，所有者のノード ID と同期変数の状態変数のみを LLCM 上に配置する．その他のメンバは通常メモリ上に配置し，UDP 版 Fagus と同様に個々のノードで管理する．所有者の変更と共有変数の無効化要求を行う際 (図 4 の (1)) には，LLCM 上に配置した同期変数の所有者 ID と状態変数に対して AOTF 送信 (図 4 の (2)) を行う．DIMMnet による送信はリモートメモリへの書き込みと同様であることから，所有者の変更と共有変数の無効化要求に際しては，他のノードのホスト CPU を介することなく同期変数を書き変えることができる．これにより，所有者以外のノードは，同期変数の最新の状況を所有者とネットワークに負担を掛けることなく知ることができる．また，UDP 版 Fagus と比較して同期変数の管理に必要な CPU 資源とネットワーク資源を低減することができる．低減した資源は並列プログラムの本来の目的である計算処理や，共有変数の送信処理に振り分けることが可能になり，UDP 版 Fagus と比較してより高速に並列プログラムを実行させることができる．

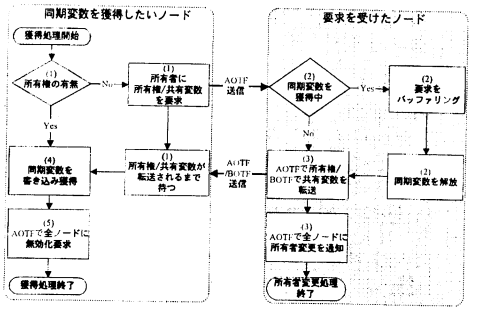

図 5: DIMMnet 版 Fagus の書き込み獲得の手順

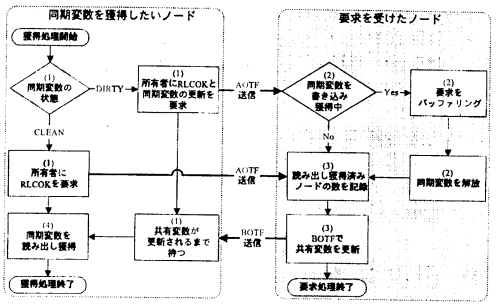

図 6: DIMMnet 版 Fagus の読み出しの手順

5.2 WLOCK に遷移する手順

DIMMnet 版の WLOCK に遷移する手順を述べる (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) WLOCK に遷移する同期変数の所有権の有無を確認する．無ければ同期変数の状態を確認して，CLEAN であれば所有権の要求を，DIRTY であれば所有権と共有変数の更新の要求を所有者に AOTF 送信して待つ．
(2) 所有権の要求を受信した所有者は，その同期変数を獲得しているときには，要求をバッファリングする．
(3) 所有者は所有権と，必要なときには共有変数を AOTF/BOTF 送信する．このとき，すべてのノードに対して同期変数の所有者が変わったことをを AOTF 送信で通知する．
(4) 所有権を受け取った新たな所有者は，WLOCK に遷移する．ここで書き込み獲得が成立する．
(5) Fagus 内のすべてのノードに対して無効化要求を AOTF 送信する．

5.3 RLOCK に遷移する手順

DIMMnet 版の RLOCK に遷移する手順を述べる (図 6 参照)．なお，図 6 中の (番号) は以下の各文の番号に対応する．

(1) RLOCK に遷移する同期変数の状態を確認し，CLEAN であれば RLOCK 要求の要求を AOTF

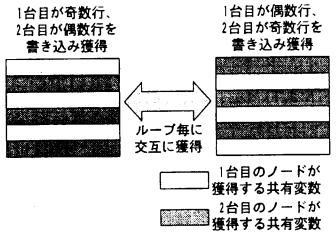

図 7: 評価プログラムの概要

```
for(k=0; k<ループ処理回数; k++){
  for(j=0; j<同期変数の総数; j++){
    if(j == kループ毎に交互に奇数か偶数){
      同期変数の獲得(sync_var[j], WLOCK);
      for(i=0; i<計算回数; i++){
        計算処理;
      }
      同期変数の解放(sync_var[j]);
    }
  }
  実行の同期();
}
```

図 8: 評価プログラムのソースコード

送信する．DIRTY であれば RLOCK と共有変数の更新の要求を所有者に AOTF 送信して待つ．

(2) 要求を受信した所有者は，その同期変数を書き込み獲得しているときには，要求をバッファリングする．

(3) 所有者は同期変数を読み出し獲得しているノードの数を記録して，必要であれば共有変数を BOTF 送信する．

(4) 共有変数が更新されたら，RLOCK に遷移する．ここで読み出し獲得が成立する．

6 評価

UDP 版 Fagus と DIMMnet 版 Fagus の同期変数管理機構の性能評価について述べる．評価環境は，Intel PentiumIII 950MHz Dual，1.5GB メモリ，100BaseTx Ethernet，DIMMnet-1(電気，250MHz)，Linux-2.4.2 を搭載した PC2 台である．

6.1 評価方法

評価プログラムの概要を図 7 に，ソースコードの概要を図 8 に示す．配列の各行を同期変数に割り当て，2 台のノードが k ループ毎に偶数行と奇数行の同期変数を交互に書き込み獲得する．1 台目のノードが奇数行を書き込み獲得しているときには，2 台目のノードが偶数行を書き込み獲得する．二巡目の k ループに入ると，逆に 1 台目のノードが偶数行を，2 台目のノードが奇数行を獲得する．以上により，常に所有権と共有変数の転送が必要となるため，他方のノードに対して常に同期変数の処理を要求する．したがって，2 台のノード間でデータのやり取りが頻繁に行われることから，同期変数の管理性能と共有変数の転送性能を測定するに適したプログラムである．

このプログラムを，同期変数の個数と，同期変数に割り当てた共有変数のデータ量を変化させて評価を行う．共有変数のデータ量を変化させて DIMMnet 版 Fagus と UDP 版 Fagus の実行時間の差を調べることで，双方のシステムの同期変数の処理性能が分かる．また，同期変数の個数を変化させて DIMMnet 版 Fagus と UDP 版 Fagus の実行時間の差を調べることで，双方のシステムの同期変数の処理性能と共有変数の転送性能が分かる．

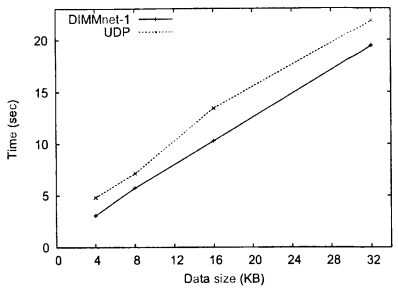

図 9: 同期変数 1024 個の場合
配列サイズ: double 型 1024×512〜4096

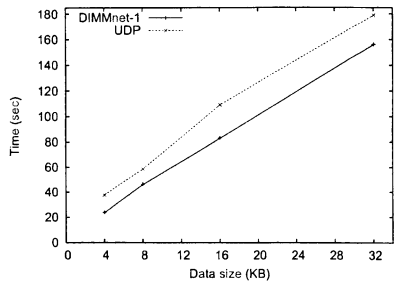

図 10: 同期変数 8192 個の場合
配列サイズ: double 型 8192×512〜4096

6.2 共有変数の処理性能

同期変数 1 つあたりの共有変数のデータ量を 4KB から 32KB まで変化させ，同期変数の個数を 1024 個と 8192 個として評価を行った結果を，それぞれ図 9 と図 10 に示す．横軸が共有変数のデータ量を示し，縦軸が時間を示す．これらのグラフは，同期変数の数が一定であることから，共有変数のサイズが増加した場合の，共有変数を転送しユーザの任意の場所にコピーするという共有変数処理性能の変化を表す．

図 9 と図 10 では，すべての場合において DIMMnet 版 Fagus の実行速度が，UDP 版 Fagus の実行速度を上回っている．しかし，共有変数のサイズに関わらず，実行時間差は変化せずにほぼ一定のまま

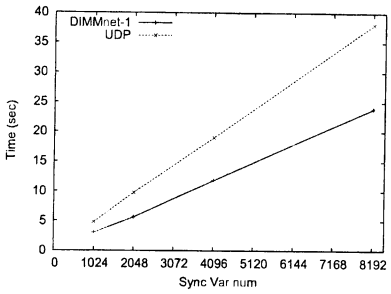

図 11: 共有変数 4KB の場合
配列サイズ: double 型 1024〜8192×2048

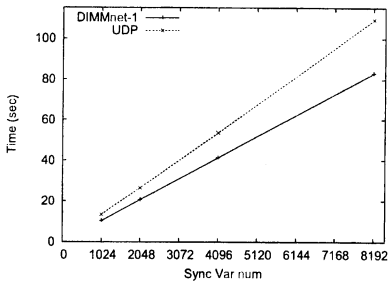

図 12: 共有変数 16KB の場合
配列サイズ: double 型 1024〜8192×2048

推移している．つまり，共有変数のサイズが増加しても同期変数の数が一定であるなら双方のシステムの実行時間の差はほぼ一定であることが分かる．これは，DIMMnet の通信バンド幅が 100BaseTx の Ethernet よりも高いためにデータ転送速度は向上するものの，DIMMnet 版 Fagus と UDP 版 Fagus 双方のシステム内における共有変数のコピー時間はほとんど同じであるために，データ転送とコピー時間を合わせた，全体としての共有変数の転送時間は双方のシステムでほとんど差が無い．このため，共有変数のサイズが増加しても，実行時間差は変化せずにほぼ一定であると考えられる．以上から，この実行時間の差は双方のシステムの同期変数処理時間の差であると考えられる．

6.3 同期変数の処理性能

同期変数 1 つあたりの共有変数のデータ量を一定にして，同期変数の個数が 1024 個と 8192 個の場合の評価を行った．同期変数 1 つあたりの共有変数のデータ量が一定であることから，同期変数の個数が増加すれば共有変数の総量も増加する．共有変数のデータ量が 4KB の結果を図 11 に，16KB の結果を図 12 にそれぞれ示す．横軸が同期変数の個数を示し，縦軸が時間を示す．

図 11 と図 12 から，同期変数の数が増加すると，DIMMnet 版 Fagus と UDP 版 Fagus の実行速度差が大きくなることがわかる．同期変数の個数が増加すれば共有変数の総量も増加することから，本評価の増加分には同期変数処理時間と共有変数処理時間の増加分が含まれる．しかし，6.2 節より，DIMMnet 版 Fagus と UDP 版 Fagus の共有変数処理時間の差はほとんど無いことが分かるため，図 11 と図 12 の実行速度差の増加は同期変数処理時間によるものと考えられる．これは，DIMMnet の高バンド幅通信による速度向上とともに，DIMMnet 版 Fagus の同期変数管理機構の処理オーバヘッドが，UDP 版 Fagus の同期変数管理機構の処理オーバヘッドよりも低いため，同期変数の個数が多くなるにしたがって実行速度差が大きくなるものと考えられる．

7 おわりに

本稿において，DIMMnet 版 Fagus と DIMMnet を用いた同期変数管理機構の仕組みを示した．DIMMnet 版 Fagus は従来の UDP 版 Fagus よりも高速に動作し，その同期変数管理機構は従来の同期変数管理機構と比較して処理オーバヘッドが低いことを示した．これは，個々のノードに対して同期変数の処理を要求するため，個々のノードのホスト CPU が明示的に同期変数を書き換えなくてはならない UDP 版 Fagus と，AOTF 送信を利用して DIMMnet 上の LLCM に配置した同期変数を書き換えるため，個々のノードのホスト CPU は特に何もしなくてよい DIMMnet 版 Fagus との差であると考えられる．

今後は，RLOCK 遷移の仕組みと，DIMMnet を用いたデータ送信機構を見直し，さらなる高速化を図る予定である．

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(2000.8).

[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 and Yoshitoshi KUNIEDA: “MIRAI: Automatic Parallelizing and Distributing Compiler based on cc-COMA approach”, Proc. of Int. Conf. on Parallel and Distributed Processing Techniques and Applications(PDPTA'2001), Vol.III, pp.1193–1199(2001.6).

[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.SIG 4，pp.866–878(2002).