2004－EVA－8　(3)
2004／3／18

メモリスロット装着型ネットワークインタフェース DIMMnetを用いた分散共有メモリシステムFagusの実装

奥野 一矢†　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情報通信システム学科
†††京都大学大学院工学研究科

概要　分散共有メモリシステムの性能のボトルネックは通信にある．この問題を解決するために，DIMMスロット搭載型のNICであるDIMMnetを用いる．我々は，DIMMnet上で動作するアプリケーションとして，通信のボトルネックが軽減されたソフトウェア分散共有メモリシステムFagus/DIMMnetを実装した．Fagusの通信プロトコルをUDP/IPからDIMMnetの持つ通信機構であるAOTF送信機構とBOTF送信機構に変更した．性能評価ではAOTF送信機構，BOTF送信機構の性能評価を行い，今回実装した同期関数と送受信関数の性能評価について述べる．

Implementation of Software Distributed Shared Memory system Fagus using DIMMnet Network Interface Plugged into a DIMM Slot

Kazuya Okuno†　Shoichi Saito††
Tetsutaro Uehara†††　Yoshitoshi Kunieda††
†Graduate School of Systems Engineering, Wakayama University
††Department of Computer and Communication Sciences,
Faculty of Systems Engineering, Wakayama University
†††Graduate School of Engineering, Kyoto University

Abstract Communication overhead is bottleneck for PC /WS cluster using Distributed Shared Memory (DSM). Software DSM system Fagus is implemented using DIMMnet network interface plugged into a DIMM slot to reduce communication overhead. Communication protocol of Fagus is changed from UDP/IP to AOTF and BOTF sending mechanisms.
This paper shows implementation method and performance evaluation of Fagus using DIMMnet. In evaluation, performance of AOTF and BOTF sending mechanisms are measured. Furthermore, evaluation of sending, receiving and synchronizing function of Fagus using DIMMnet is shown.

1 はじめに

PC/WSクラスタを用いる並列処理において，ノード間の通信性能はPC/WSクラスタの性能を決める大きな要因である．特に通信が多発する並列アプリケーションにおいてはクラスタの性能低下は顕著である．なぜなら，ノード内におけるデータ転送速度と比較して，ノード間でのデータ転送速度が低いからである．現在利用されている主なNICはPCIに搭載されるものであり，PCIバスのバンド幅は32bit33MHzPCIで133MB/sである．さらに，送受信の多発する場面では半分の66MB/sまで低下し，高性能なNICであってもPCIバスの限界から通信性能の低下が起きる．

このような背景からPCIバスではなくDIMMスロットに搭載されるNICとして開発されたDIMMnet[1][2]に注目した．DIMMnetはメモリバンド幅（6.4GB/s）を利用するため，高速な通信が可能である．我々は，PCクラスタ上で動作するソフトウェア分散共有メモリシステムFagus[3]を，DIMMnetを搭載したPCクラスタ上に実装した．これにより，通信によるボトルネックの少ないDSMを実現する．FagusはUDP/IPプロトコルを用いるが，DIMMnetは独自のプロトコルによる通信を行う．このため，Fagus内部の通信部のプログラムをDIMMnet用に置き換えた．

本稿では，まず，DIMMnetとFagusについて述べる．次に，Fagusの通信プロトコルの移植手法について述べる．最後に，性能評価について述べて，DIMMnetを用いたFagusの評価および考察を行う．

2 DIMMnet

DIMMnetは新情報処理開発機構，東京農工大学，慶應義塾大学，東芝で共同開発中であり，PC66，PC100またはPC133仕様のDIMMスロットに装着するネットワークインタフェースである．

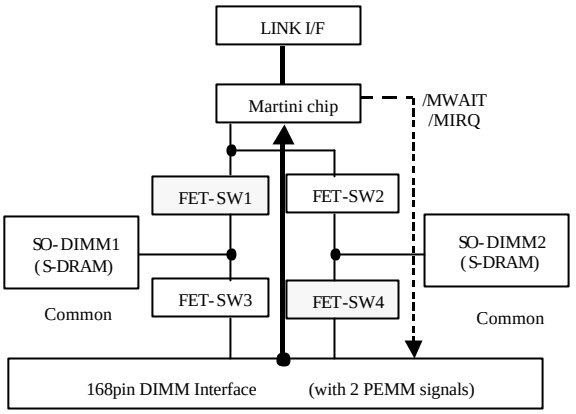

図1 DIMMnetの基本構造

DIMMnetの持つ通信機構には低遅延性を最優先させたAtomic On-the-fly（AOTF）通信機構と低遅延性と高バンド幅と柔軟性を両立するBlock On-the-fly(BOTF)通信機構がある．DIMMnetは，DIMMスロット搭載型NICという特性から，ホスト側からは主記憶と同様なメモリとして，ネットワーク側からはリモートアクセス可能なリモートメモリとして認識される．ユーザが利用できるNICに搭載されたメモリ（以下，NICメモリという）はSO-DIMMと低遅延メモリ（以下，LLCMという）がある．SO-DIMMは64MB～1GBのメモリ空間を利用できる．LLCMは128KBのメモリ空間を利用できる．ユーザがLLCMを効果的に利用するには，AOTF送信用の領域として利用するか，低遅延が要求される小容量のBOTF送信用の領域に限定される．一方，SO-DIMMは，広いメモリ空間を利用できる．しかし，SO-DIMMをローカルとリモートの両側からアクセス可能にするために，SO-DIMMを2枚用いてこれらをFETバススイッチによって切り替える必要がある(図1参照)．以降，このFETバススイッチによるスイッチングをバンク切り替えと呼ぶ．図1ではFET-SW1とFET-SW4がOFF，FET-SW2とFET-SW3がONになっている．このときSO-DIMM1はローカルメモリとして，SO-DIMM2はリモートメモリとして扱われる．

SO-DIMMは，データ受信時にバンク切り替えのためのオーバヘッドが生じる．よってSO-DIMMを効果的に利用するケースは大容量のBOTF送信の領域である．

AOTF送信は，所定のアドレス対してデータの書き込み命令を実行することでパケットを送信する．この命令は，送信すべきデータがレジスタ上に存在すれば，CPUがレジスタ上にあるデータをユーザモードのまま所定の仮想アドレスに書き込むという１命令だけである．AOTF送信のデータ送信サイズは1〜8バイトである．

BOTF送信は，ほとんどパケットそのものに近い状態でホストCPUからNICのハードウェアに渡されるので，少ないクロック数でネットワークに出力することができる．BOTF送信では１パケットで送信できるデータの最大サイズは464バイトである．

3 分散共有メモリシステムFagus

Fagusは，我々が開発中である自動並列化コンパイラMIRAIの実行時環境として実現したソフトウェア分散共有メモリシステムである．Fagusでは，共有メモリの一貫性制御をEC（Entry Consistency）とRC（Release Consistency）によって行う．ユーザは，並列プログラムとFagusの機能を提供するライブラリlibFagusをリンクして使用する．通信には，POSIXスレッドとUDP/IPを利用して通信を行っている．現在，FagusはLINUX上で動作しているが，LINUXそのものに変更を加えていないため，POSIXスレッドとUDP/IPが使用可能なUNIX系OSであれば，容易に移植可能である．

プログラマまたは，コンパイラによって生成された並列プログラムは，ユーザによって各ノードへ配置される．実行時には，プロセス内で一つ以上のユーザスレッドが生成され，ユーザスレッドを用いて並列プログラムの各タスクを実行する．共有変数は，Fagusのメモリ割り当て機能を利用することで，プロセスの仮想メモリ空間に割り当てる．ユーザスレッドは，共有変数のアクセス時に必要に応じて，libFagusにキャッシュ制御の指示を与える．共有変数は，同期変数と関連付けられている．同期変数とは，複数のノード間で共有変数の一貫性を保つために，共有メモリの同期オペレーションを行うための変数である．共有変数にアクセスする時は，共有変数に関連付けられた同期変数のオーナーから同期変数の所有権の獲得を行う必要がある．

4 DIMMnetを用いたFagusの実装

本実装での主な変更点は通信プロトコルの変更である． UDP/IPの送受信関数をAOTFとBOTFを用いて実装した．特に，関数全体を変更した関数は同期関数である．同期関数は，ノード間で交換するメッセージのサイズは小さいことからAOTFのみ用いて実装することで，同期関数の高速化が期待できる．図2に本実装での送受信の流れを示し，以下，データの送受信の流れについて述べる．

4．1 送受信の流れ

dimmnet_sendto() と dimmnet_sendmsg() と dimmnet_recvfrom()は， UDP に用いられる sendto() と sendmsg() と recvfrom() を AOTF と BOTFを用いてDIMMnet用に移植したものである．detect_thread()は，データの受信検知を行うスレッドである．receiver() はFagusの受信スレッドであり，dimmnet_recvfrom()の呼び出しと受信パケットの処理スレッドへの受け渡しを行う．以下，送信側が受信側へデータを送信する流れを述べる．

dimmnet_sendto()またはdimmnet_sendmsg()の実行により，送信側は引数の送信先NodeIDをDIMMnetの送信先RANKに変換する．NodeIDは，Fagusが各ノードを識別するために

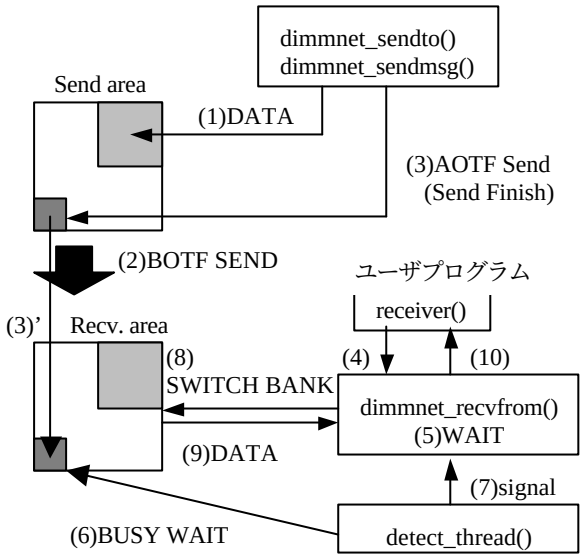

図2 DIMMnetでの送受信の流れ

用いるIDである．次に，DIMMnet用のパケットを構築する．DIMMnet用に定義したパケットヘッダに送信データのサイズとフラグを設定する．ここでは，フラグをSEND_MSGに設定する．サイズとフラグが設定されたパケットヘッダと送信データを，SO-DIMMの送信領域にコピーする(図中(1))．コピーの後，BOTFで送信先RANKに送信する(図中(2))．BOTF送信の後，送信側は受信側の受信検知のため，指定したLLCMの送信領域にAOTFでBOTF_SENDのメッセージを送信する(図中(3)および(3)')．

受信側は，receiver()によってデータの受信を待つ．receiver()はdimmnet_recvfrom()を呼び出して(図中(4))，detect_thread()からのシグナルを待つ(図中(5))．detect_thread()では，指定した受信領域にBOTFの送信完了を示すAOTFデータが到着するまでビジーループで検査する(図中(6))．BOTFの送信完了を示すAOTFデータを確認後，detect_thread()は，dimmnet_recvfrom()に対してシグナルを送信する(図中(7))．シグナルを受信したdimmnet_recvfrom()は，AOTFの受信領域をクリアし，SO-DIMMのバンク切り替えを行う(図中(8))．バンク切り替えを行うことで，送信側から送信されたパケットを取得できる．バンク切り替えを行った後，パケットヘッダを受信領域から取得する．パケットヘッダに含まれるサイズ分のメモリを受信領域から取得し，receiver()に受信データを渡す(図中(9)および(10))．

次に，受信検知のためにAOTFでメッセージを送信した理由について述べる．まず1つ目の理由は，SO-DIMMの受信領域に対してビジーウェイトでメッセージの受信を検知するためには，受信検知毎にバンク切り替えを行う必要がある．バンク切り替えによるオーバヘッドが増大と，受信データの書き込み中に発生するバンク切り替えによるデータの紛失の可能性があるためである．2つ目の理由は，BOTF送信がLLCMを用いて行われた場合でも，受信領域に対してビジーループで受信検知を行った場合，受信データが正常に受信されない現象が確認されたことがあるからである．以上の理由から，BOTF送信でデータ送信を行った後でAOTF送信を用いて受信検知のためのメッセージを送信した．

4．2 同期関数

Fagusの同期関数であるcomm_sync()をAOTF送信を用いてDIMMnet用に改良し，dimmnet_comm_sync()として実装した．comm_sync()は4.1節で述べたDIMMnet用送受信関数の実装によりDIMMnet上でも動作可能である．しかし，comm_sync()の現在の実装方式では，AOTF送信とBOTF送信の両方が必要となる．comm_sync()が実現する同期処理は，数バイトのデータ送受信できれば充分である．そのため，comm_sync()をAOTF送信のみを用いて新たに実装することにより，高速化が期待できる．以下，AOTF送信のみによる処理の流れを述べる．

Nodeidが0番のノード(以下，親ノードと呼ぶ)は，他のノード（以下，子ノードと呼ぶ）すべて

からAOTFによりSYNC_ENTRYメッセージを受信するまで待つ．子ノードは，親ノードにSYNC_ENTRYメッセージをAOTFで送信した後で，親ノードからAOTF送信によるSYNC_PERMITメッセージを受信するまで待つ．すべての子ノードからSYNC_ENTRYメッセージを受信した親ノードは，すべての子ノードに対してAOTFでSYNC_PERMITメッセージを送信し，受信領域を初期化してから同期処理を終了する．親ノードからSYNC_PERMITメッセージを受信した子ノードは，受信領域を初期化してから同期処理を終了する．

5 評価

5．1 評価環境

実験に使用したシステムは，プロセッサがPentiumIII 1266MHz，メモリが1GB，イーサネットとDIMMnetで接続された2台のPC/AT互換機である．評価2ではこの環境の他にギガビットイーサネットで接続されたクラスタを使用する．クラスタシステムは，プロセッサがIntel Xeon 1.70GHz，メモリが1 GBのPC/AT互換機である．

5．2 評価プログラム

本論文の実験では，以下の3つの評価を行った．評価の内容について述べる．

評価1 dimmnet_sendto()とdimmnet_recvfrom()を用いた通信プログラムを用いて通信性能を測定する．送信側は10000バイトのデータを10000回送信する．送信データの受信検知手法はdetect_thread()スレッドを用いるものと，スレッドを用いずにdimmnet_recvfrom()側が直接検知する方法の2種類を行う．

評価2 DIMMnetの拡張ライブラリのAPIである_expBarrier()(評価項目①)と今回実装した同期関数(評価項目②)の性能を測定し，比較する．それぞれ1000000回実行し，1回あたりの実行時間を測定する．また，UDPを用いたFagusの同期関数の性能とクラスタ上でFagusの同期関数も測定し，DIMMnetとファストイーサネット(評価項目③)及びギガビットイーサネット(評価項目④)の性能の比較を行う．

5．3 実行結果および考察

表1に評価1を，表2に評価2の結果をそれぞれ示す．表1より，1Mbps以下という結果が得られた．性能低下の原因を調べるために，表3に評価2における実行時間の内訳を示す．表3より，実行時間のほとんどがバンク切り替えとなっていることが分かる．本実装では，1パケットを受信するごとにバンク切り替えを行うため，バンク切り替えのコストが，そのまま通信速度に影響したと考えられる．この対策として，複数のパケットを受信してからバンク切り替えを行う方法がある．しかし，バンク切り替えのオーバヘッドが改善できるがパケットの処理に遅延が生じる．

次に，detect_thread()を用いた場合と用いない場合では，用いない場合の方が実行時間は短い．UDPの場合では，データの受信を検知したときカーネルに割り込みがかかる．その間，他のスレッドはオーバラップして実行できるためオーバヘッドが小さい．これに対して，DIMMnetでデータの受信を検知するためには，受信領域をビジーループで検査する必要がある．detect_thread()スレッドは生成されてから常にビジーループで待ちつづけるのに対して，スレッドを用いない場合は受信関数内で受信検知を行うため，受信処理以外では無駄なビジーウェイトが発生しなかったため考えられる．

表2より，評価2では_expBarrier()(評価項目①)

表1　評価1の結果

評価項目	1回あたりの実行時間(msec)	スループット（Mbps）
受信検知スレッド有り	23.590	0.339
受信検知スレッド無し	20.000	0.400

表2　評価2の結果

評価項目	1回あたりの実行時間 (μsec)	①の実行時間を1としたときの実行時間
①	4.0881	1.000
②	29.8033	7.290
③	179.8826	44.001
④	136.6572	33.428

表3　評価1の実行時間の内訳

実行の内訳	実行時間の割合(%)
バンク切り替え	99.46
内部処理	0.36
BOTF送信	0.11
同期処理・AOTF送信	0.071

がもっとも性能が高かった．今回実装した同期関数(評価項目②)は，_expBarrier()と比較して1回の同期あたり約7.3倍の時間を要したが，UDPを用いたファストイーサネット(評価項目③)のFagusの同期関数の約6倍の性能を得た．また，ギガビットイーサネット(評価項目④)で接続されたeuropa3クラスタ上での同期関数より約4.8倍の性能を得た．評価2の結果より，_expBarrier()と今回実装した新しい同期関数は，AOTFのみで通信を行うため，UDPと比較しても低遅延で同期を行うことができた．

6　まとめ

DIMMnetを用いてFagusを実装した．FagusをDIMMnet環境に移植するために，通信プロトコルをUDP/IPからAOTF通信とBOTF通信に変更した．性能評価では，受信検知スレッドの有無によりメッセージ受信方式の比較を行い，受信検知スレッドを用いないメッセージ受信方式の方が低オーバヘッドであり，高性能であることがわかった．しかし，バンク切り替えによるオーバヘッドが大きく，SO-DIMMを用いたデータ送信の使用は難しい．そこで，現在，AOTF通信とLLCMを利用したデータ通信について検討を行っている．

同期処理の性能比較においては，低遅延のAOTF送信機構を用いて実装した新しい同期処理の方が，UDPを用いて実装された従来の同期処理よりも高速であった．しかし，DIMMnetライブラリの拡張APIとして実装された同期関数が最も高性能であったため，今後の実装では，拡張APIの同期処理関数を用いる予定である．

参考文献

[1] 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).

[2] 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)

[3] 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).