マルチメディア通信と分散処理 85－17
（1997. 11. 6）

PCクラスタを利用した並列処理環境WinDSE

市場 裕臣† 大西 淑雅‡ Bernady O. Apduhan† 末吉 敏則§
† 九州工業大学 情報工学部
‡ 九州工業大学 情報科学センター
§ 熊本大学 工学部

概要

我々は，ワークステーションクラスタ上に分散共有メモリモデルに基づく並列処理機能をユーザに提供する環境としてDSE(Distributed Supercomputing Environment)の開発を行ってきた．一方，計算機シェアの大半を占めるパーソナルコンピュータにおいても，Windows NT/95のようにプリエンプティブなマルチタスク/マルチスレッド機能や強力なネットワーク機能を提供するOSが一般的に利用されるようになっており，分散協調処理による並列処理も可能となっている．そこで，本稿ではDSEに関する過去の研究で得られた知見を活用し，Windows NT/95上で動作する並列処理環境WinDSEの構成および実装を行った。この結果，WinDSEはDSEと遜色のない実行効率で動作することが確認できた．

WinDSE: A Distributed Shared Memory Based Parallel Computing Environment on a PC Cluster

Hiroomi Ichiba† Yoshimasa Ohnishi‡ Bernady O. Apduhan† Toshinori Sueyoshi§

† Department of Artificial Intelligence, Kyushu Institute of Technology
‡ Information Science Center, Kyushu Institute of Technology
§ Department of Computer Science, Kumamoto University

Abstract

We have developed a parallel and distributed computing environment on a workstation cluster, called DSE, which can support the primitives for distributed shared memory based parallel processing. Considering the growing market share of today's PCs and the capabilities of its OS to provide preemptive multitasking/multithreading as well as powerful networking functions has driven our motivation to build a parallel processing environment on a cluster of PCs. We made good use of the knowledge and expriences gained from our past research on DSE and implemented WinDSE on Windows NT/95. In this paper, we report the implementation and the performance evaluation of WinDSE.

1 はじめに

プロセッサ単体の性能やネットワーク技術の著しい発展に伴い，分散環境上に散在する複数の計算機をまとめて1台の並列計算機として見做す，クラスタコンピューティングに関する研究が盛んに行われている [1,2,3,4]．

クラスタコンピューティングでは，各ノードに存在する複数のプロセスが互いに協調して並列処理を実現する．その並列処理モデルは，IPC(InterProcess Communication)によるデータ通信の手法によって，メッセージパッシングモデルと分散共有メモリモデルの2つに大別される．前者がメッセージの送受信によってデータ

の受渡しを行うのに対し，後者は分散配置したメモリ上に仮想的な共有メモリを構築し，プロセス間のデータ授受を実現する．これら2つのモデルは並列プログラミングにおいて以下のような特徴を挙げることができる．前者はメッセージの送受信時に暗に同期がとられるが，最新情報の位置を常に把握する必要がある．これに対し，後者はプログラム中で明示的に同期をとる必要があるが，データの位置透過性が保証される．そのため，従来のアプリケーションプログラムの移植が比較的容易となる．また，効率の良い共有データ初期配置機構ならびにキャッシュ機構を導入することによって [6]，細粒度のデータ並列性を要するアプリケーションにおいても性能向上を期待することができる．

本研究室では，ワークステーションクラスタ上で動作し，後者のモデルに基づく並列処理機能を提供する環境として分散スーパーコンピューティング環境 DSE(Distributed Supercomputing Environment) の研究を行っている ([1,5,6,7,8])．DSEのみならず、PVMやMPI等に代表されるクラスタコンピューティング環境は，主としてワークステーションクラスタ上に実装されてきた．一方，計算機シェアの大半を占めるPC(Personal Computer)においては，各ベンダから高性能なプロセッサや通信資源が低価格で提供されてきている．また，PCのOSとして広く使用されているWindows NT/95は，ワークステーション上のUNIXでしか利用できなかったマルチタスク/マルチスレッド機能やネットワーク機能を提供できるようになった．このことは，PCクラスタを用いたクラスタコンピューティング環境の構築を可能にしている．

そこで，本研究ではWPVM[9]，WINMPICH[10]およびWMPI[11] 等に採用されているメッセージパッシングモデルではなく，並列アプリケーションの記述性に富んだ分散共有メモリモデルに基づく並列処理環境 WinDSE(Windows DSE) の構築を目的とする．WinDSE の構築にあたっては，DSE に関する研究で得られた知見 ([5,7]) を活用し，ユーザレベルで効率の良い並列処理機構を実現する．

以下，本稿では，第2章でDSEの概要について，第3章でWinDSEの実装について説明を行う．つづいて，第4章で離散コサイン変換による性能の評価結果を示したあと，最後に第5章でまとめて，今後の課題について述べる．

2 DSEの概要

DSEは，ワークステーションをノードとする分散システム上で，分散共有メモリモデルに基づく並列処理を行えるように設計されたソフトウェアである．以下に，DSEの概要について記す．

2.1 システムモデル

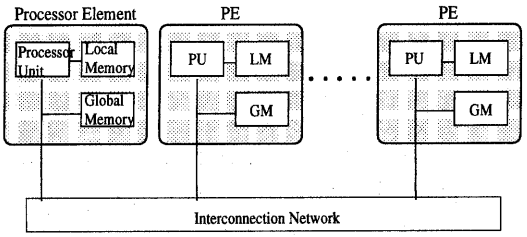


図 1: システムモデル

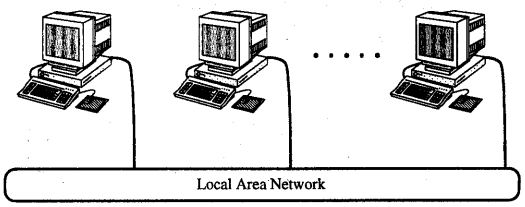


図 2: システム構成

DSEにおけるシステムモデルを図1に示す．各プロセッサ要素 (PE) は，プロセッサユニット (PU)，ローカルメモリ (LM) およびグローバルメモリ (GM) から構成される．PU間の通信は，各PEを接続する相互結合網を介して行われる．LMはローカルのPUによってアクセスされるメモリ領域であり，命令コードおよびローカルデータが格納される．また，GMはシステム全体を通じてユニークなアドレス空間を生成し，各PEで共有されるデータが配置される．

また，DSEの動作するシステム構成を図2に示す．これは，複数のワークステーションがLANを介して接続された形で表される．図1との対応としては，基本的にPEがWSに，相互結合網がLANに相当する．

2.2 ソフトウェア構成

DSEのソフトウェア構成を図3に示す．DSEはDSE ProcessとDSE Kernelから構成される．DSE Process

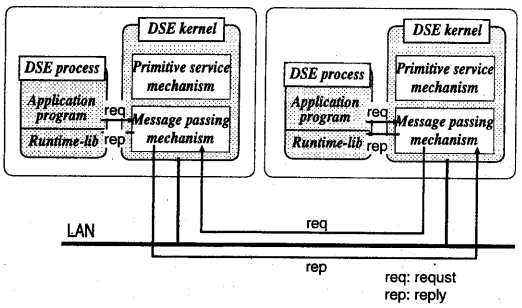


図 3: ソフトウェア構成

は並列アプリケーションの実行主体であり，並列処理を行う際には DSE Kernel に対してプリミティブ処理の要求を行う．要求を受けた DSE Kernel は，その要求が自ノード内で処理できるものであれば，適切な処理を行った後に DSE Process にその応答を返す．要求が他ノードに対するものであれば，該当するノードに存在する DSE Kernel へ処理の依頼を行う．ここで，プリミティブとは，共有メモリアクセス，並列プロセス (DSE Process) 管理，同期機構，および DSE Kernel の保持する各種情報の獲得機構などの並列処理を行うために不可欠な機能の総称である．これらの処理要求ならびに応答は全てメッセージを介して行われるため，DSE Kernel の構成はメッセージドリブンアーキテクチャとなっている．

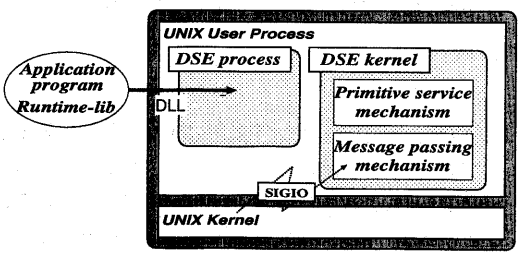


図 4: DSE の内部構成

2.3 実装手法

UNIX を OS とする WS 上に DSE を実装した場合において，DSE Process と DSE Kernel とを異なる UNIX プロセスに実装した場合 [1]，DSE Process-DSE Kernel 間あるいは DSE Kernel-DSE Kernel 間のメッセージを介した通信遅延を避けることはできない．しかし過去の調査から，この通信遅延が DSE における並列処理効率が大きく妨げていることが明らかなっている [5]．そこで，図 4 に示すように，DSE では DSE Process と DSE Kernel を Sun Microsystems 社の提供するダイナミックリンクライブラリを利用して，同一の UNIX プロセスに実装している [7]．この実装手法により，DSE Process-DSE Kernel 間のソケットを介した通信を UNIX プロセス内でのポインタ操作に置き換えることが可能となる．しかしながら，単一プロセス実装によってアクセスの高速化が図れる反面，DSE Process と DSE Kernel の両処理のコンテキストが混在してしまう．DSE では，この問題点に対処するため，SIGIO シグナルによるソフトウェア割り込みを使用している．SIGIO は所定の設定を行うことによって，ソケットディスクプリタに対する入出力が可能になった際に該当するプロセスに対して発せられる．DSE Kernel はメッセージに駆動して処理を行うことから，DSE では，DSE Kernel をこのシグナルハンドラに実装している．

これにより，DSE Process と DSE Kernel の処理は図 5 のようになる．DSE Process の実行中にメッセージ (DSE Kernel への処理依頼) の到着をシグナルによって検知すると，処理の実行権は DSE Process から DSE Kernel に移行する．DSE Kernel は当該ソケットからメッセージを受信し，適切なメッセージの解析，処理，および応答を行った後，処理を DSE Process に戻す．ゆえに，DSE Process と DSE Kernel のスケジューリングは DSE Kernel を優位とする横取りスケジューリングとなる．

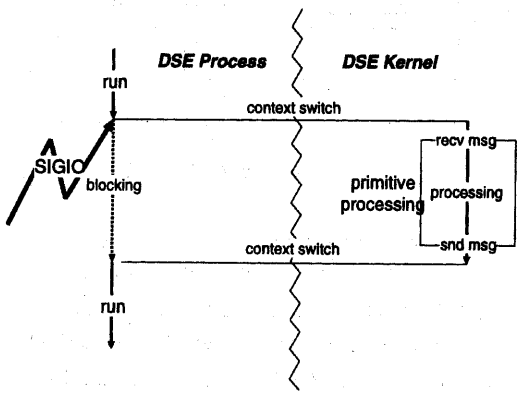


図 5: DSE の内部動作シーケンス

3 WinDSEの実装

ここでは，Windows NT/95 を OS とする PC クラスタにおいて DSE と同様の並列処理機能を提供する WinDSE の実装方法について述べる．WinDSE の実装方針として，DSE の研究で得られた知見を採用する．すなわち，WinDSE Process と WinDSE Kernel を単一の Windows プロセス上に実装し，両者のコンテキストの明確な区別を行う．

以下，WinDSE における DSE Process と DSE Kernel をそれぞれ WinDSE Process，WinDSE Kernel と呼ぶ．WinDSE Kernel の機能をライブラリとして提供し，並列アプリケーションにリンクすることによって WinDSE Process との単一プロセス化を図る．一方，Windows NT/95 のシグナルセットに SIGIO に相当するものが存在しないため，DSE で採用したコンテキストの切り替え手法をとることはできない．そこで，WinDSE では Windows NT/95 の提供するスレッド機構を用いて両処理のコンテキストの保証を行う．WinDSE の内部構成を図 6 に示す．図中の斜線部分が，WinDSE でスレッドを用いて実装される部分である．ここで，WinDSE Kernel は並列所処理機能を提供するための 5 つの機能から構成される．

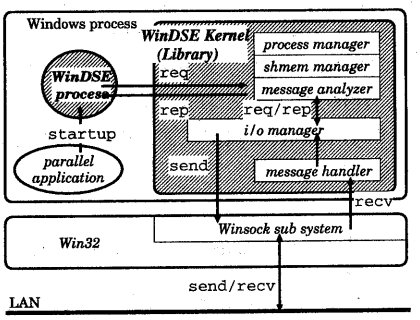


図 6: WinDSE の内部構成

並列アプリケーションが実行されると，各ノードでプライマリ・スレッドによって WinDSE Kernel がスレッドとして起動される．生成された WinDSE Kernel は，指定された仮想的なトポロジに従い，Winsock(Windows socket) を用いて他ノードに存在する WinDSE Kernel と適切なコネクションを確立し，同時に各 WinDSE Kernel に対してユニークな識別子をつける．コネクションの確立完了後，識別子 0 の WinDSE Kernel を生成したプライマリスレッドから並列分岐を実行し，複数の WinDSE Process による並列処理を実現する．なお，WinDSE で実行されるスレッドは全て Windows NT/95 におけるフォアグランドプロセスと同等のプライオリティに設定されており，そのスケジューリングは OS によるラウンドロビンによって行われる．

図 6 中の message handler は，他の WinDSE Kernel から非同期に着信するメッセージを受信する．Windows NT/95 上においても，この機構を実現するために，非同期ソケットを作成して入出力を行うのが一般的である．しかしながら，この機構は UNIX における SIGIO シグナルとは異なり，Windows フォーマットメッセージを介してプロセスに通知される．すなわち，このメッセージを受信するためには，プロセス内の Windows メッセージキューをポーリングしていなければならない．このため，WinDSE では，非同期メッセージを受信するために，上記の非同期ソケットを使用した手法をとらずに，今回の実装では，単純に select() を用いてソケットをポーリングすることした．

4 性能評価

WinDSE の実装の有効性を確認するために，小規模なアプリケーションを用いて性能評価を行った．ここでは，画像の情報量を削減する離散コサイン変換アルゴリズムを用いたアプリケーションを使用して，WinDSE の性能評価を報告し，DSE における評価結果と比較を行う．

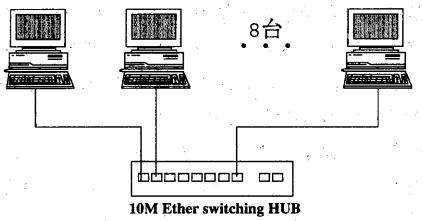
4.1 評価環境

図 7 に，WinDSE および DSE において性能評価を行った環境を示す．WinDSE を評価した PC は，133MHz Intel Pentium, Window95, 48M バイトのメモリを搭載したものを使用し，8 台の PC を 10Base-T, 10M Ether switching HUB を介して接続した．DSE の評価環境に関しては，110MHz MicroSparc2, SunOS 4.1.4, 64M バイトのメモリを搭載した WS が，同じネットワーク環境を介して 8 台接続したもの使用した．

4.2 離散コサイン変換 DCT

2 次元 DCT の符号化は，画像を $M \times N$ 画素のブロックに分割し，各ブロックの各画素に対し，式 (1) の計算を行うことによって実現される．

並列化した 2 次元 DCT 符号化では，Fetch & Add



	WinDSE	DSE
Node	PC	WS
CPU	133MHz Pentium	110MHz Micro Sparc2
Memory	48Mbytes	64Mbytes
Nework	10Base-T	10Base-T
HUB	10M Ether switching HUB	10M Ether switching HUB

図 7: 評価環境 (WinDSE, DSE)

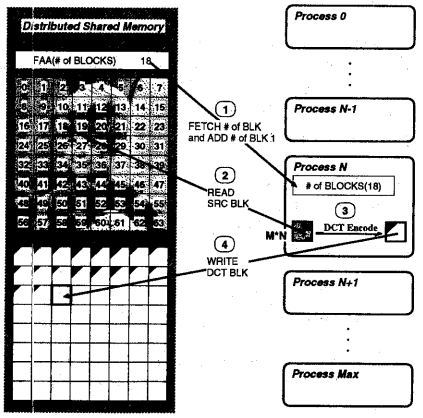


図 8: 離散コサイン変換の並列化

による動的負荷分散を用いて，図 8 に示す手順で処理を行う．まず共有変数 FAA の値を読みとり，インクリメントした値を書きこむ (①)．共有変数 FAA は初期設定値から開始し，アクセスする度に任意の値を加えることができる同期付共有変数である．次に FAA の値を参照し，原画像中の適切なブロック (16 × 16) をロードする (②)．続いて画像を 2 次元 DCT 符号化した後 (③)，結果を共有メモリ内の適切な位置にストアする (④)．以下，全てのブロックが終了するまで，①~④を繰り返す．なお，プログラムの連続性を考慮して，原画像データや符号化データ等の共有データは分散共有メモリ中の連続した領域に配置した．

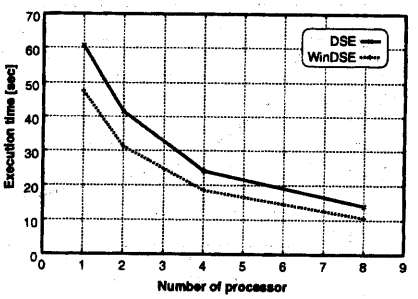
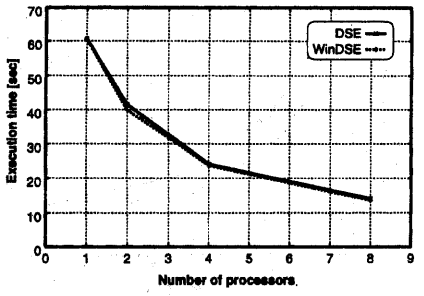


図 9: 測定結果

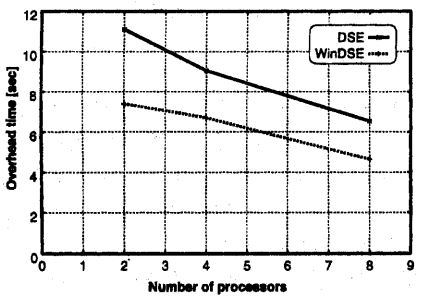
4.3 評価結果

2, 4, 8 台の計算機を利用して，WinDSE および DSE 上で，2 次元 DCT の符号化に要した時間をそれぞれ図 9 に示す．グラフ中の値はそれぞれ 100 回の測定平均を示している．グラフ中の横軸はプロセッサ台数を，縦軸は処理時間 (秒) を表している．計測には，GetSystemTime(), gettimeofday() をそれぞれ使用した．図 9 から台数が増加するにつれ，WinDSE, DSE ともに類似した形跡で処理時間が減少していることを確認することができる．

そこで，WinDSE と DSE の実行効率を比較するために，CPU の動作周波数の差異を隠蔽するよう，図 9 の結果に補正を加える．具体的には，WinDSE の結果に，逐次処理時間 (DSE)/逐次処理時間 (WinDSE) を乗算する．この結果で得られたグラフを図 10(a) に示す．さらに，図 9 の結果と並列化した際の理想の時間との差異を算出したグラフを図 10(b) に示す．このグラフは，並列化したことによって生じたオーバーヘッド (通信，同期等) に近似することができる．図 10(a) のグラフでは，WinDSE と DSE において，結果としてほぼ一致した処理時間が得られた．また，図 10(b) のグラフから，台数の増加に伴い，オーバーヘッド時間が減少しているのが分かる．これは，台数増加によって，1 台当たりにメッセージ送受信量が減少するためである．さらに，この値は，WinDSE, DSE において同等の割合で減少しており，計測点における比は図 10(a) を算出する際に使用した値とほぼ一致している．このことから，並列化に伴うオーバヘッドも CPU の動作周波数の同様の比率で影響を受けていると考えられる．上記の結果から，本実装による WinDSE は高性能実装を行った DSE とほぼ同等の実行効率を示すと考えられる．



(a) 動作周波数における補正後の実行時間



(b) 並列化に伴うオーバーヘッドの見積り

図 10: 補正結果

5 おわりに

本稿では，分散協調処理可能となったPCクラスタ上に，ユーザレベルで効率の良い並列処理機能を提供するのを目的とした並列処理環境WinDSEの構成ならびに実装を示した．

この目的を実現するために，WinDSEでは，WSクラスタ上で研究されてきたDSEにおける技術を念頭においた設計・実装を行った．具体的には，WinDSEの提供する機能および構成はDSEのシステムモデルを継承し，実装においては，DSE ProcessとDSE Kernelの単一プロセス実装に着目し検討を行った．その結果，WinDSEではWindows NT/95の提供するマルチスレッド機構を用いて，単一プロセス化，さらにWinDSE ProcessとWinDSE Kernelのコンテキストの切り替えを実現した．また，並列2次元DCT符号化アプリケーションを用いた性能評価では，WSクラスタを用いたDSEと比較して遜色のない性能を得ることができ，WinDSEの有効性を示した．

クラスタコンピューティング環境では，その性能はネットワークを介した通信処理やプロセッサの動作速度等の様々な要因に影響を受ける．したがって，今後は様々なアクセスパターンをもつ並列アプリケーションによる性能評価および内部解析を行い，WinDSEの有効性と問題点を明らかにする．

さらに，頻繁な共有メモリアクセスのために性能が頭打ちになっていた並列アプリケーションの性能向上を図るため，現在研究中のソフトウェア・キャッシュ機構ならびにfalse sharingを回避するためのデータ配置機構をWinDSEに導入する予定である．

参考文献

[1] T. Tezuka, K Ryokai, B.O. Apduhan, and T. Sueyoshi: "Implementation and Evaluation of Distributed Supercomputing Environment on a Cluster of Workstations," Proc. 1992 International Conference on Parallel and Distributed Systems, pp.58-65, 1992.

[2] P. Keleher, A.L. Cox, S. Dwarkadas, and W. Zwaenepoel: "TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems," Proc. Winter'94 Usenix Conference, pp115-131, 1994.

[3] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Mancheck, and V. Sunderam: "PVM 3 User's Guide and Reference Manual," Oak Ridge National Laboratory, 1994.

[4] Message Passing Interface Forum: "MPI: A Message-Passing Interface Standard," 1995.

[5] 手塚 忠則, 了戒 清, 末吉 敏則: "分散システムを利用した並列処理環境における通信処理の影響," 情報処理「マルチメディア通信と分散処理」ワークショップ論文集, pp.249-256, 1993.

[6] 宮原 敦夫, 大西 淑雅, 末吉 敏則: "クラスタコンピューティングにおけるソフトウェア・キャッシュ機構の評価," 第11回情報処理学会九州支部研究会論文誌, pp250-259, 1996.

[7] 大西 淑雅, 了戒 清, 末吉 敏則: "分散共有メモリモデルに基づくHPC環境の高性能実装と性能評価," 情報処理学会論文誌, Vol.36, No.7, pp1729-1737, 1995.

[8] Y. Ohnishi, Y. Sugimoto, and T. Sueyoshi: "A High-Performance Cluster Computing Environment Based on Hybrid Shared Memory / Message Passing Model," IEICE Trans. Inf.&Ssyt, Vol E80-D, No 4, pp448-454, 1997.

[9] http://alentejo.dei.uc.pt/wpvm/

[10] http://www.erc.msstate.edu/mpi/mpiNT.html

[11] http://dsg.dei.uc.pt/ fafe/w32mpi/