システムソフトウェアと
オペレーティング・システム 76－14
（1997．8．21）

WSクラスタにおけるスケーラブルネットワークの予備評価

中西剛紀† 中野智行††
中條拓伯† 金田悠紀夫†

ワークステーションクラスタを用いた並列処理においてスケーラビィリティ(台数効果)を確保するためには，プロセス間通信の実現方式が問題となる．しかしながら，既存のLAN環境は通信に要するオーバヘッドが大きく，細粒度の並列処理に有効であるとはいえない．
本稿では，ネットワーク資源をメモリ資源として見なすことで低オーバヘッドのプロセス間通信を実現するLMMC (Link Memory based Communication) モデルの実験システムをシリアルリンクを相互結合網として利用するワークステーションクラスタ上に構築し，その基本的な通信性能を評価することでLMMCモデルがワークステーションクラスタにおける並列処理に有効な通信モデルであることを示す．

Preliminary Evaluation of Scalable Network Mechanism on WS Clusters

YOSHINORI NAKANISHI,† TOMOYUKI NAKANO,†† HIRONORI NAKAJO† and YUKIO KANEDA†

To achieve high scalability in workstation clusters for fine-grained parallel processing, a method of interprocess communication is significantly important. However, there exists significant communication overheads in the traditional LAN environment. In a LMMC model, low overhead interprocess communication is implemented with an abstraction that network resource are considered as memory resources which are managed and protected by memory management hardware and operating systems. In this paper, we propose an LMMC (Link Memory based Communication) model experiment system on workstation clusters using serial links called STAFF-Link, and evaluate the basic performance of an LMMC model. As a result of experiments, we show the effectiveness and the possibility of an LMMC model.

1. はじめに

近年のワークステーションの性能向上および低価格化により，ネットワークで接続された複数のワークステーションを用いた並列処理システムが注目されている．これらは，ワークステーションクラスタ，もしくはNOW (Network of Workstation) と呼ばれ，各方面で盛んに研究が行なわれている[1]．

しかしながら，現状では，ワークステーションクラスタ上での並列処理環境が十分に整備，活用されているとはいえない．その原因としては次のことが挙げられる．

- ネットワーク形態
  既存のLAN環境で広く用いられているEthernetがバス型のネットワークであるため，ノード数増加に応じた通信バンド幅の確保が困難である．
- 通信に要するオーバヘッドが大きい
  現状のワークステーションでは，ネットワークデバイスを扱う際，およびプロトコル (TCP/IP等) 処理に要するオペレーティングシステムのオーバヘッドが大きい．

これらの問題は，現状のLAN環境が細粒度の並列処理を考慮に入れていなかったことに起因するものと考えられる．ネットワーク形態については，並列計算機の相互結合網とLANの中間のカテゴリに位置する，System Area Network (SAN) が提唱されている[2]．ここではSANが満たすべき条件を以下の様に考える．

- システム全体のネットワークのバンド幅が十分であり，ノード数の増加に応じたバンド幅を確保できる．
- 分散環境とは異なり，ノード間を数十～数百mに限定することで，高速で信頼性のある通信路を提供する．

† 神戸大学工学部情報知能工学科
Department of Computer and Systems Engineering, Faculty of Engineering, Kobe University
†† 日立ソフトウェアエンジニアリング株式会社
HITACHI Software Engineering Corporation

- 様々な結合トポロジを柔軟に構成できる．

そこで，我々は高速なシリアルリンクを用いて上記のようなSAN環境を構築し，その上で低レイテンシの通信を実現する通信機構として，LMMC (Link Memory Management based Communication) と呼ばれる通信モデルを提案した[3]．LMMCモデルでは多数のリンクに存在する物理的な通信バッファを1つのメモリ空間と見なし，通信を通信バッファへのメモリアクセスによって行うことにより，低オーバヘッドのプロセス間通信を実現する．

本稿では，LMMCモデルの実現方式について述べ，基本的な通信性能の評価を行う．まず2章でLMMCモデルについて説明し，3章で関連研究について述べ，4章でLMMCモデルの実験システムにおいて基本的な通信性能の評価を行い，5章でまとめと今後の課題について述べる．

2. LMMCモデル

2.1 LMMCアーキテクチャ

LMMCモデルを実現するためのハードウェア構成をLMMCアーキテクチャと呼ぶ．LMMCアーキテクチャでは，ワークステーション間の接続に用いられるポートはLMMCネットワークインタフェースに接続され (図1)，プロセッサはLMMCネットワークインタフェース (プロセッサとのインタフェース用のポート，ネットワークインタフェース間の通信用のポート，通信用バッファ，およびそれらを制御するコントローラから構成される) に付随する通信用のバッファを自ノード内にあるメモリ素子と同様に扱う．これにより，ノード間の通信は (リモート) メモリ操作に置き換えられ，ノード間の通信路を入出力デバイスとしてではなく，メモリ管理用のハードウェア (Memory Management Unit, MMU) とオペレーティングシステムの管理により保護されたメモリ資源として扱うことができ，低オーバヘッドのノード間通信を実現することが可能となる．

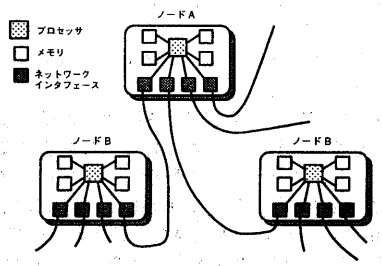

図1 LMMCアーキテクチャ

この場合，通信バッファを実メモリ空間にマッピングする方法が問題となる．LMMCアーキテクチャでは，メッセージパッシングモデルで記述された多くのプログラムに用いられる非同期送信・同期受信を効率良く実現するために，自ノード内の受信バッファと接続先の受信バッファを実メモリ空間へ配置する方式を採っている．これによりデータの送信は非同期のリモートメモリへのアクセスで，データの受信はローカルメモリへのアクセスで行われる (図2)．

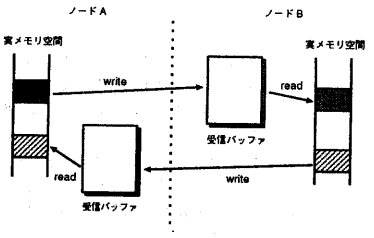

図2 通信バッファのマッピング

2.2 LMMCソフトウェアインタフェース

LMMCアーキテクチャ上でLMMCモデルのプロセス間通信を行うためのオペレーティングシステムモジュールをLMMCソフトウェアインタフェースと呼ぶ．ユーザプロセスはこのインタフェースを直接利用したり，このインタフェース上に構築されたメッセージパシングシステムやDSM (Distributed Shared Memory) システムを用いることでプロセス間通信を行う．

LMMCソフトウェアインタフェースでは，各プロセスのページテーブルエントリの項目を一致させることで複数のプロセスでページを共有することにより，多対多の通信を行なうことができる．このような共有可能な単方向の通信路をポート (port) と呼ぶ．各ポートはポート識別子と呼ばれる正の整数によって，ネットワークインタフェース間で一意に定められる．

本稿では物理アドレス空間に配置された通信バッファをリンクメモリ (link memory)，リンクメモリをページサイズで分割した単位をリンクページフレーム (link page frame)，仮想アドレス空間上の通信用のページを仮想リンクページ (virtual link page) とそれぞれ呼び，通常のメモリ，ページフレーム，仮想ページと区別する．

2.2.1 ポート管理のためのデータ構造

ページテーブル

ページテーブルはユーザプロセス毎に用意される．仮想リンクページとリンクページフレームを対応させることによりリンクメモリへのポートの割り当て状態を管理する (図3)．

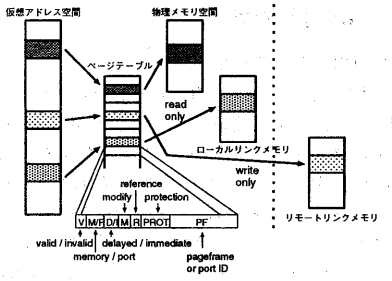

図3　ページテーブル

リンクページフレームテーブル

リンクページフレームテーブルは現在のリンクページフレームとポート識別子の対応を保持するテーブルであり，1つのネットワークインタフェースに対して送信用と受信用のテーブルを用意する (図 4)．主に，ネットワークページングの際にどのリンクページフレームを操作の対象にするかを決定するのに用いられる．

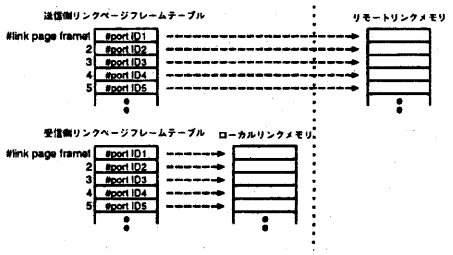

図4　リンクページフレームテーブル

ポートテーブル

ポートテーブルはポート識別子とポートに対応づけられたプロセスの識別子と仮想ページとの対応を保持するテーブルであり，1つのネットワークインタフェースに対して送信用と受信用のテーブルを用意する (図 5)．

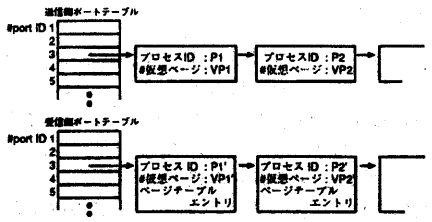

図5　ポートテーブル

2.3　リモートページング

LMMC モデルでは，通信バッファ用のメモリは通常のメモリと同様にページフレームに分割され，MMU とオペレーティングシステムによりページ単位で管理される．物理メモリに対する仮想記憶と同様に，ユーザプロセスが利用できる通信用のメモリ領域は，実際の通信バッファの容量による制限を受けない．そのため，通信バッファを管理する上でページングが必要となるが，この際の動作は，ページングの対象となるページが自ノード内に存在せず，接続先の受信バッファに存在するため，通常の仮想記憶のものとは異った機構を要する．

LMMC アーキテクチャでは，通信バッファに対するページングが必要となった場合，接続先のノードに対してページング処理の要求を発行し，要求を受けとったノードはトラップをかけてオペレーティングシステムにページング処理を依頼する (図 6)，本稿ではこのようなページング動作をリモートページング (remote paging) と呼び，通常のページングと区別する．

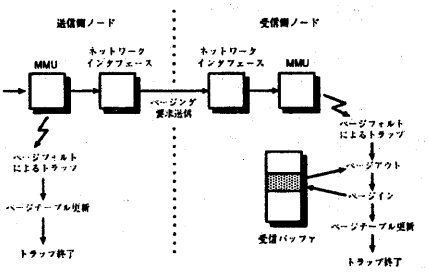

図6　リモートページング

一方，LMMC ソフトウェアインタフェースにおいては，リモートページングの処理は上述のデータ構造 (2.2.1 節) を用いて次の手順で行われる．

ページフォルトが発生した (送信側) ノードでは次の処理が行われる．

(1)　トラップが発生．

(2)　ページテーブルを用いて，書き込もうとした仮想リンクページが対応しているポートを調べる (有効/無効ビットがクリアされているページテーブルエントリにはリンクページフレームの代わりにポート識別子が格納されている)．

(3)　リンクページフレームテーブルを用いて，ページアウトするポートを決定．

(4)　ページインするポート識別子とページングの対象となるリンクページフレーム番号をパラメータとし，リモートページングの要求を発行．

(5)　ポートテーブルを用いてページアウトされたポートを共有しているプロセスおよび仮想リンクページを調べ，対応するページテーブルエントリを変更．

(6) ポートテーブルを用いてページインするポートを共有するプロセスおよび仮想リンクページを調べ，対応するページテーブルエントリを変更．
(7) トラップから復帰．

リモートページアウトを受けとった (受信側) ノードでは次の処理が行われる．

(1) ネットワークインタフェースがプロセッサにトラップをかける．
(2) 受けとったリモートページング要求のパラメータから，ページアウトすべきポートをリンクページフレームテーブルから調べる．
(3) ページアウトの処理を行う．
(4) ページアウトされた仮想ページフレームに対応するページテーブルエントリを変更．
(5) リモートページング要求のパラメータで指定されたポートをページインする．
(6) ページインされた仮想ページフレームに対応するページテーブルエントリを変更．
(7) トラップから復帰．

3. 関連研究との比較

Active Messages[4] はメッセージが到着した時に起動されるユーザレベルハンドラへのアドレスをメッセージに含ませることにより，通信のオーバヘッドを低減している．しかしながら，メッセージの到着毎にトラップが発生し，ハンドラの起動によるコストも大きい．LMMC モデルでは普段の通信ではトラップは発生せず，リモートページアウトが発生した時のみトラップが生じる．

通信路をメモリ空間へマッピングするアプローチは SHRIMP のネットワークインタフェース[5] や ソフトウェアメモリベース通信 (MBCF)[6] などで用いられている．SHRIMP ネットワークインタフェースでは，送信プロセスと受信プロセスの仮想メモリ空間のマッピングにより通信を行うという点で LMMC モデルと同様のインタフェースを提供する．しかしながら，LMMC モデルでは通信用のメモリ管理が通常のメモリと統合されているため，ワークステーションクラスタ環境において効率的なスケジューリングや同期機構を実現し，並列システムの性能をさらに向上させることが可能である．

また，MBCF では，ページ管理機構を用いて保護された高速な通信を行うという点で LMMC モデルと同様のアプローチだが，MBCF が既存の LAN 環境への汎用性を重視するのに対し，LMMC モデルは高速なシリアルリンクを相互結合網として利用するワークステーションクラスタ環境に対して，効率の良い並列処理環境を構築する．

4. 基本性能評価

4.1 STAFF-Link

STAFF-Link (Serial Transparent Asynchronous First-in First-out Link) は，超並列計算機 JUMP-1 のディスク入出力ユニットとクラスタ間の接続に用いるために開発したシリアルリンクであり[7]，FIFO メモリインタフェースにより，ユーザプロセスから直接アクセスすることが可能である．現時点では，SUN のワークステーション用拡張I/O スロット SBus に接続され，1 つの SBus スロットに対して 4 ポートのリンクを持つことができる．

SBus インタフェースカードから 4 枚の STAFF-Link 通信制御ドータカードを実装できるマザーボードを図 7 に示す．

以前の設計においては，STAFF-Link の FIFO のステータスをソフトウェアによりチェックしていた．すなわち，送信時には送信用 FIFO が Full にならないように Full フラグを監視しながら転送し，受信時には Empty フラグを見ながら受信 FIFO からデータを読みとっていた．

しかしながら，このチェックコードは転送レートを低下させることとなるため，送信 FIFO の Full フラグと受信 FIFO の Empty をハードウェア的に組み込むように改良を施した．具体的には，送信時に Full フラグが立っていた場合や受信時に Empty であった場合には SPARC には再実行アクノリッジを返送することとし，送信または受信できないときは，できるようになるまで同じ命令を繰り返すようにし，ソフトウェアによるフラグチェックが不要となった．これより従来の約 3 倍の転送性能が見込めることとなる．

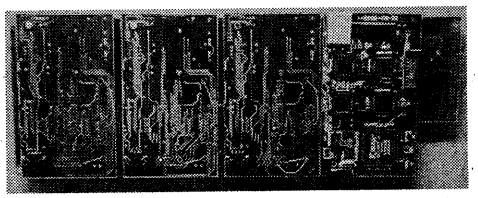

図 7 STAFF-Link マザーボード

STAFF-Link では，ハードウェアは簡単なフロー制御しか行わず，通信制御用のプロセッサは存在しない．この実験システムでは，ソフトウェアによる MMU のエミュレーションという形で実装を行った．

4.2 実験システムの構成

現状のワークステーションクラスタにおける LMMC モデルの通信性能を評価するため，本稿では STAFF-Link を用いた実験システムを構築し，基本的な性能を評価した．図 8 に実験システムの構成を示す．

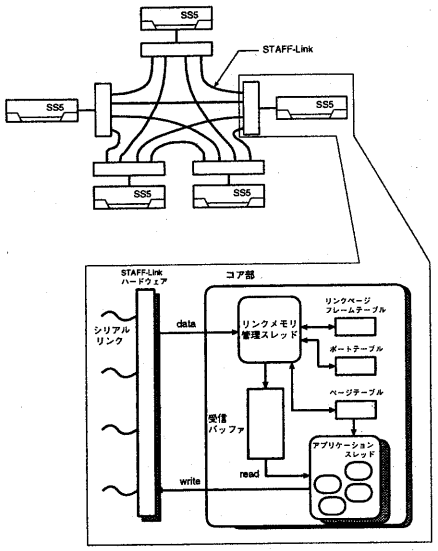

図8　実験システムの構成

リンクメモリ管理スレッドは，STAFF-Link から受信したパケットの解釈を行う．到着パケットがリモートメモリライトメッセージであった場合，受信バッファ用のメモリ領域にデータを書き込む．リモートページアウト要求であった場合，前述のデータ構造 (2.2.1) とアルゴリズムを用いてページング動作を行う．

コア部はリンクメモリ管理スレッドとアプリケーションプログラムが利用するサービスルーチンから構成され，アプリケーションプログラムはコア部と静的にリンクされて 1 つのプロセスとなる．

アプリケーションプログラムを構成する各スレッドに対してコア部が提供するルーチンは以下の 3 つである．

初期化ルーチン　受信バッファ用の領域を主記憶上に確保し，各データ構造を初期化する．また，リンクメモリ管理スレッドを起動する．

リンクメモリライトルーチン　仮想リンクメモリアドレス，データバッファ，サイズをパラメータにとり，ページテーブルを用いてアドレス変換を行い，(リモートページングが必要な時はリモートページング要求を送信後) リモートライトパケットを生成して送信する．

リンクメモリリードルーチン　仮想リンクメモリアドレスをパラメータにとり，ページテーブルを用いてアドレス変換を行い，対応するアドレスのメモリ内容を返す．

4.3　通信時間の計測

本実験システムにおいて 2 ノード間の通信時間を計測した．実験環境を表 1 に示す．

計測は転送サイズが小さい場合と大きい場合で行い，

表1　実験環境

host machine	SUN SPARCserver5
processor	microSPARCII (110MHz)
main memory	32 MBytes
pagesize	4096 bytes
I/O bus	SBus (25MHz) 32bit data bus
OS	Solaris2.4

ノード間の通信時間は，ラウンドトリップ時間 (ノード A からノード B にデータを送信し，ノード B が受信後，同じデータをノード A に送信し，ノード A が受信するまでの時間) から算出した．計測結果を図 9，図 10 に示す．

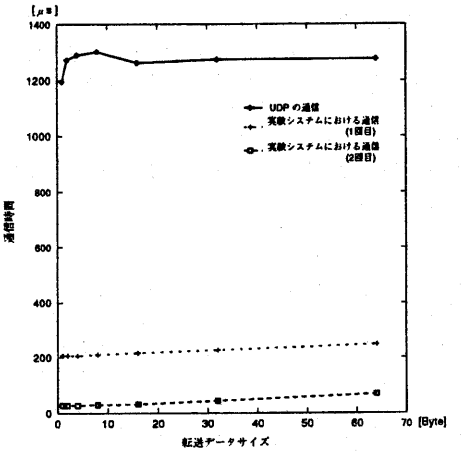

図9　2 ノード間の通信時間 (データサイズ小)

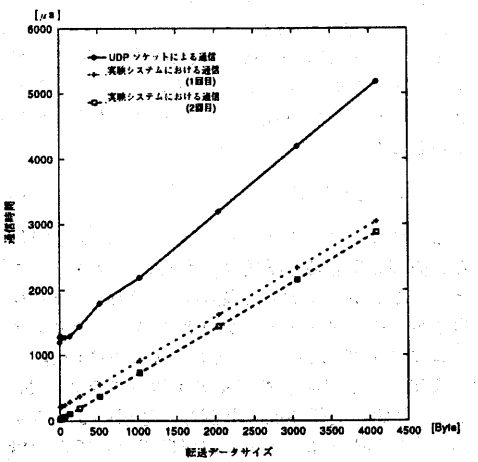

図10　2 ノード間の通信時間 (データサイズ大)

4.4 オーバヘッドの解析

図 9 より，片道の通信時間は 1 回目の通信 (この場合ノード A からノード B) と，2 回目の通信 (ノード B からノード A) で約 180μs の差があることがわかるが，これは最初の通信時に，SS5 の MMU における TLB 操作などの処理を要するためと考えられる．2 回目以降の通信ではこの処理を必要としないため，低オーバヘッドの通信が可能である．

データサイズ 1byte の時のノード B からノード A への通信時間を例にとり，細かい処理時間を調べた結果，送信側のノードにおいてリモートメモリライトルーチンに 17μs，受信側のノードにおいて，STAFF-Link の FIFO からリンクメモリ領域にデータが書き込まれるのに 20μs かかっていることがわかった．今回の実験システムでは，リモートメモリライトルーチンにおいて STAFF-Link に書き込みを行なう際の排他制御はホスト側のプロセッサで行っているが，ネットワークインタフェースに通信用のプロセッサが存在すれば，処理時間の短縮が見込める．また，MMU によるアドレス変換が行われれば，さらに通信時間を短縮できると考えられる．

今回の実験ではワークステーションクラスタのノード数を 2 台に限定しているが，Ethernet でバス型に結合されたワークステーションクラスタではノード数の増加に伴い，通信時に Ethernet へのアクセスの競合が起こり，通信時間が長くなる．LMMC モデルでは，STAFF-Link のようなシリアルリンクを相互結合網として利用するワークステーションクラスタを対象としており，ハイパーキューブなどの高次元ネットワークを構成することにより，システムの規模に応じた通信バンド幅を得ることが可能である．

一方，通信の際にリモートページングが必要となった場合は通常の送信と違い，オペレーティングシステムによる処理に起因するオーバヘッドが予想される．リモートページングの処理時間の短縮およびページフォルト率を低く抑えることがワークステーションクラスタシステムの処理能力低下を回避するための重要な課題となると考えられる．

5. おわりに

ここでは，ワークステーションクラスタ上で効率の良い並列処理環境を構築するためのネットワーク機構として LMMC モデルを提案し，LMMC モデルを実現するためのアーキテクチャとオペレーティングシステムについて考察した．また，STAFF-Link を用いて構築した LMMC モデルの実験システムにおいて，2 ノード間の通信時間を計測し，通信に要するオーバヘッドを解析することで LMMC モデルでの通信の可能性を示した．

リモートページングによるオーバヘッドを抑えるためには，効率の良いページ置換アルゴリズムが必要となるが，通常の仮想記憶に用いられる LRU (Least Recently Used) や FIFO などは，メモリアクセスの空間的あるいは時間的局所性を前提としているため，通信バッファに対するアクセスにおいて，これらが有効であるとはいえない．

今後は，リモートページングに対して効率の良いページ置換アルゴリズムの確立，LMMC ソフトウェアインタフェースを実際にオペレーティングシステムのモジュールとして実装したシステムの構築を予定している．

参考文献

1) T. E. Anderson, D. E. Culler, and D. A. Patterson, "A Case for NOW(Networks of Workstations)," *IEEE Micro*, Vol. 15, No. 1, pp. 54–64, (1995).

2) J. L. Hennessy and D. A. Patterson, "Computer Architecture: A Quantitative Approach (2nd Edition)," Morgan Kaufmann, (1995) pp.563–629.

3) 中條拓伯，中野智行，金田悠紀夫，"WS クラスタにおけるスケーラブルネットワーク機構，" 電気通信学会技術研究報告，Vol. 97, No. 87, pp. 43–48, (1997).

4) T. Eicken, D. E. Culler, S. C. Goldstein, and K. E. Schauser, "Active Messages: a Mechanism for Integrated Communication and Computation," *Proceedings of the 19th International Symposium on Computer Architecture*, pp. 256–267, (1992)

5) M. A. Blumrich, K. Li, R. Alpert, C. Dubnicki, and E. W. Felton, "Virtual Memory Mapped Network Interface for the SHRIMP Multicomputer," *Proceedings of the 21st Annual International Symposium on Computer Architecture*, pp. 142–153, (1994).

6) 松本，駒嵐，渦原，平木，"メモリベース通信による非対称分散共有メモリ，" コンピュータシステムシンポジウム論文集，情報処理学会 pp. 37–44, (1996).

7) 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).