計算機アーキテクチャ 119－5
（1996. 8. 27）

ネットワーク仮想化機構とその評価

佐藤充, 田中英彦
東京大学 工学系研究科
東京都文京区本郷 7–3–1

概要

並列計算機上で複数のユーザをサポートするための機構として, ハードウェアを用いたネットワークノード番号変換を用いることを提案する. ハードウェアによる変換を行なうことによって, 頻繁に生じるであろう相互結合網経由の通信を高速に行なうことができ, 分散共有メモリシステムにも対応できる. このようなノード番号変換を行なうと, 複数のノードを単一ノードとして扱うことによって, 等価的に相互結合網とノードとのスループットをあげることができる. 本稿ではネットワーク仮想化機構の概念について述べた後, 簡単な評価結果についても報告する.

Virtual Network System and its Preliminary Evaluation

Mitsuru SATO, Hidehiko TANAKA
School of Engineering, the University of Tokyo
7–3–1 Hongo, Bunkyo-Ku, Tokyo

Abstract

We propose the hardware system which supports multi-user system on parallel computers. Translation of network node ID with hardware makes communications through interconnection network fast and supports distributed shared memory. By using this mechanism, we can tolerate multiple node as a single node, and this makes throughput between interconnection network and node fast. In this paper, we describe concept of Virtual Network System and report result of preliminary evaluation of Virtual Network System.

1 はじめに

数値演算を主な目的とした並列計算機は，高速なネットワーク，高速な要素プロセッサなどによって徐々に実用的なレベルに達しようとしている．しかしその一方で，汎用的な並列計算機はまだまだ一般に浸透しているとはいい難い．

近年のワークステーション・クラスタなどの流行は，ポータビリティーを保ちつつ高速性や経済性を重視した並列システムに対する要求を示していると言えるであろう．にもかかわらず，並列計算機がその役割を果たしていないのは，現在の並列計算機システムにいくつか欠けているものがあるためであると考えられる．それらは，たとえば

- 複数ユーザが利用できる環境
- ユーザが扱いやすい並列プログラミング環境

などである．

これらの問題を解決するためのハードウェアサポートとして，本稿では「ネットワーク仮想化機構」を提案する．このネットワーク仮想化機構は簡単なハードウェアを用いて，これまでの並列計算機では実現できなかった上記の問題を解消するための仕組みである．

以下では，まず研究の背景およびネットワーク仮想化機構について述べ，その後にシミュレーションによる簡単な予備評価とその結果について述べる．

2 背景

2.1 汎用並列計算機

数値演算が主な目的である計算機は，その規則性を利用してできる限りの高速化を行なう．しかし汎用計算機では，定型的な問題から非定型的な問題まで扱わなくてはならない．また問題の規模についても，小さな問題から大きな問題まで効率良く扱えなくてはならない．したがって，汎用のシステムでは，単純にシステム全体を使ってひとつのアプリケーションを最適に実行するというわけにはいかず，全体的な効率を考えて実行しなくてはならない．

2.2 並列計算機における粒度コントロール

一般に，並列計算機上でプログラムを効率良く実行するためには，計算の粒度を適切に設定することが必要である [1]．粒度が小さ過ぎると，実際の計算よりもタスク (スレッド) の切替のオーバーヘッドが大きく，効率がよくない．逆に大き過ぎると，充分に並列度を利用することができず，並列計算機としての利点を活かせない．適切な粒度は計算機アーキテクチャ，特に要素プロセッサと通信用ハードウェア，および相互結合網ハードウェアに大きく依存するものである．

汎用の並列計算機を考えた場合，ハードウェアの汎用性，ソフトウェアの移植性，既存の技術を容易に利用できるなどの点から，要素プロセッサとして一般に用いられている高性能 RISC プロセッサを用いるのが適切であると思われる．この高性能 RISC プロセッサは，基本的に逐次的な流れの計算を前提としており，プログラムの持つ時間的・空間的局所性を利用して，レジスタやキャッシュなどにより性能を向上させている．したがって，これらのプロセッサでのタスクスイッチのオーバーヘッドはかなり大きく，計算を効率良く行なうための処理の粒度はある程度以上大きくなければならない．

2.3 マルチユーザ環境

並列計算機の資源を複数のユーザが同時に利用する方法としては，時間的分割と空間的分割の 2 つの方法が考えられる．時間的分割は，単一プロセッサでマルチユーザを実現する場合と同様に，タイムスライスによって複数ユーザを切替える方法である．一方，空間的分割は，PE 空間を各ユーザに分け与え，それぞれの領域を単一のユーザが利用するものである．

時間的分割では，スイッチのためのオーバーヘッドが単一プロセッサの場合に比べて大きくなる．システムの規模が大きくなるほど，このスイッチのためのオーバーヘッドは大きくなる．

また，汎用のシステムにはさまざまな規模の処理が実行される．処理の中には規模の大きなものから小さなもの，またそれらのもつ並列度もさまざまに分布しているであろうことが予想される．前節で述べた最適な計算粒度を考慮すると，小さな規模の計算では，たとえ並列度が大きくても小数の PE で計算を実行した方が全体の利用効率が高くなることが予想される．つまり，それぞれの処理に応じて空間的分割を行なうことが，全体の効率を向上させることにつながる．

以上のことから，本研究で対象とするシステムは，基本的に各ユーザに適当な PE を割り当てる空間的分割による複数ユーザのサポートを行なうことを考える．当然のことながら，レイテンシ隠蔽などのためにさらに時間的分割も併用することも考えられるが [2]，空間的分割が基本にあるということを前提とする．

3 ネットワーク仮想化機構

3.1 ネットワーク番号変換

ひとつの並列計算機上に複数のユーザを空間的に共存させるには，これまでは相互結合網にパーティショニング機能を設け，そのハードウェアに頼って分割する方法がとられてきた．

しかしこの方法では，

- 静的な分割しかできない
- 相互結合網のトポロジに依存した分割しかできない

などの問題点があり，実際に利用しても，台数の少ない，独立した並列計算機が複数あるようなイメージを提供するにとどまっていた．

そこで，空間分割をソフトウェア的に行なう方法が考えられる．すなわち，相互結合網へのアクセスがある度にテーブル引きを行ない，物理的空間ではなく論理的な空間を提供するというものである．したがって，このテーブルを更新するだけで，動的にマッピングを変更することができ，柔軟性が向上するという利点が生じる．また，ハードウェアが提供する物理トポロジを直接見せるのではなく，論理的なトポロジ空間をユーザに提供することができる．

このようなテーブル引きの機構は，たとえばメッセージパッシングライブラリのようなライブラリレベルで提供したり，オペレーティングシステムで提供したり，ハードウェアで提供したりと，さまざまなレベルのものが考えられる．

本稿では，この中でハードウェアを用いた手法を提案する．すなわち，MMU(Memory Management Unit) のように，ハードウェアによりテーブル引きを行ない，ネットワーク上でのアドレス (ノード番号) 変換を行なう方法である．

以下では，各ノードに物理的に割り振られている番号を「物理ノード番号」，各ユーザから見たノードの番号を「論理ノード番号」と呼ぶ．この用語を用いると，このテーブル引きは論理ノード番号から物理ノード番号への変換と言い直すことができる．これを「ネットワークノード番号変換」と呼ぶ．

このネットワークノード番号変換をハードウェアで実現することのメリットは，

- 頻繁に生じるであろうノード間通信を高速に実現することができる
  大きなブロック転送を主とする通信では，メッセージパッシングをベースにし，ネットワークイメージを仮想化するのにライブラリまたはOSレベルで変換するだけでも充分実用的であろう．しかし，たとえばデータベースシステムのように本質的に通信を頻繁に行なうアプリケーションでは，通信のレイテンシが全体の計算時間を左右する場合がある．そこでハードウェアインプリメントによる高速な処理が有効となる．
- 分散共有メモリを利用した場合にも機能する
  分散共有メモリは，細かな通信を行なうのに適したシステムである．しかし，通信というインタフェースがメモリアクセスに一元化されるために，OSやライブラリなどが途中で介在する余地がなくなってしまう．したがってハードウェアでの実現が必須となる．

などが挙げられる．逆に，複雑なハードウェアを用意しなければならないという欠点も生じる．この欠点に対しては，実用上少量の簡単なハードウェアで実現できることを第3.5節で説明する．

このネットワーク仮想化機構を用いて，ユーザが他のノードに対するアクセスを行なった時の様子を図1に示す．

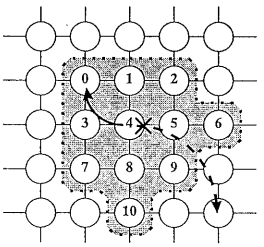

図 1: Accessing to Another Node

3.2 アドレス変換

前節で説明したように，ハードウェアによるネットワークノード番号変換は，ネットワーク空間上におけるアドレス変換と位置付けることができる．システムがグローバルなメモリ空間を提供している場合，このネットワーク空間上におけるアドレスはメモリアドレスそのものになる．この場合，ネットワークノード番号変換とは，アドレス変換の一部となる [3].

ネットワークノード番号変換は，ネットワークノード番号+メモリアドレスをアドレスとした一括型のアドレス変換と比較すると，ネットワーク空間上におけるアドレスとメモリアドレスとを区別しているという点が違っている．これは，

- ローカル/リモートというのは物理的に大きな違いである
- ネットワーク空間とメモリ空間は違う資源である

といった観点によるものである．すなわち，すべてをアドレスという形で一元化してしまうより，ネットワーク空間とメモリ空間を別に分けて管理した方が管理しやすく，パフォーマンス・チューニングもやりやすいであろうという考えに基づいている．

3.3 ノード番号エイリアス

このようなネットワーク仮想化を行なうと，任意の物理ノード番号に適当な論理ノード番号を割り振ることができる．ひとつの例として，複数の物理ノード番号に同一の論理ノード番号を割り当てた場合を考える．この場合，システムの物理構成は図 2 のようになる．

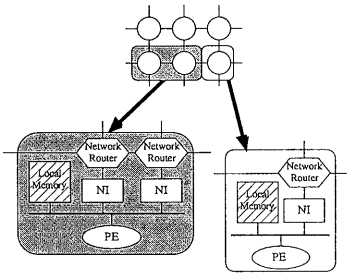

図 2: Node Number Alias

論理ノード番号→物理ノード番号の変換に適当なキー(メモリアドレスの一部のビットなど)を付け加えると，そのノードに到達するための経路を複数作ることができる．その結果としてネットワークとノード間のスループットを向上させることができる．

この方式は，相互結合網のスループットを物理的に向上させる方法とは違い，ソフトウェア的にテーブルの書き換えのみで行なわれるので，高い柔軟性を持つ．たとえば，相互結合網自体には変更を加えず，各ノードに 2 つずつのネットワーク・インタフェースを持たせることによって，容易にネットワーク・ノード間のスループットを等価的に倍にすることができる．また，ネットワークトラフックの高い特定のノードのみ，ネットワーク・ノード間のスループットを向上させ，全体の効率をよくするということも，テーブルの書き換えのみで行なうことができる．

また逆に，ひとつの物理ノード番号に対して複数の論理ノード番号を与えることもできる．この場合，いわゆる仮想プロセッサを実現していることになる．ただし，この場合はすべてハードウェアに任せるというわけにはいかず，メッセージを受信したノードが，そのメッセージはどの論理ノード番号宛かを調べる必要がある．

3.4 トポロジ変換

仮想ネットワーク機構は，あるユーザの利用する範囲内で自由に論理ノード番号をつけることができる仕組みであると言い替えることもできる．すなわち，物理的な接続を気にしなければ[1] ユーザに対して自由なネットワーク空間を提供できるということである．

たとえば，図 3 のように相互結合網の一部を User1 と User2 が分割して利用している場合を考える．User1 のもつ空間は物理的には不規則な形をしているが，これに論理ノード番号として

(0, 0), (0, 1), (0, 2),
(1, 0), (1, 1), (1, 2),
(2, 0), (2, 1), (2, 2)

という (0, 0) から始まる規則的なものを与えることによって，User1 は自分のもつ空間があたかも，閉じたトーラスのように扱うことができる．

User2 も同様に，あたかも自分の持つ空間が木構造をしているかのようにアクセスすることができる．

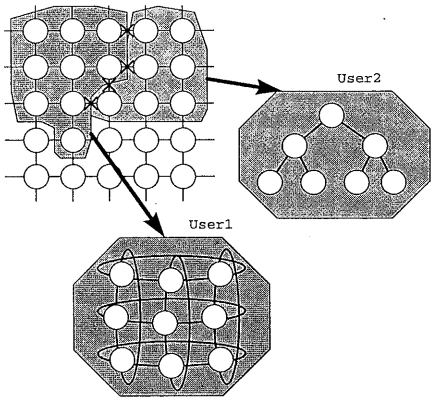

図 3: Topology Translation

このようなトポロジ変換は，OS または言語において，記述しやすい形でネットワーク空間を扱うことができるという点で有用である．しかし一方で，MIMD 的に fork を繰り返して並列実行するようなモデルでは意味がない．この仕組みが有効であるかどうかは，どのような実行モデルを採用するか，またどのようなプログラミングスタイルで並列プログラムを記述するかによって大きく違ってくる．

[1] たとえば，パケット型の相互結合網において，Virtual Cut Through などのルーティング技術を用いれば，相互結合網でのホップ数が増えてもレイテンシは高々数クロックしか増えない

3.5 実現のためのハードウェア構成

これまで述べてきた, ハードウェアによるネットワークノード番号変換を実現するための仕組みは, 他のノードへのアクセスを検出してテーブル引きを行なうという単純な動作だけで実現できる.

大規模な並列計算機になると, ノードの数が大きくなり, 必要なテーブルも大きくなる. しかし, 本研究では前提として空間分割による複数ユーザのサポートを考えているので, ハードウェアの内部に持たせなくてはならないテーブルは, 現在 active なユーザの空間 (それほど頻繁に変化はしない), および OS の空間のみである. それ以外はメモリに追い出し, メモリのアドレス変換と同様に TLB という形で保持すればよい. したがって, ハードウェアで持たなくてはならないテーブルのサイズはかなり削減することができる.

このユニットはグローバルな共有メモリを実現する際にも役立てることができる. ネットワークノード番号変換とともにメモリアドレス変換機能を持たせることによって, グローバルメモリアドレスを扱うことが可能になる.

4 予備評価

仮想ネットワーク機構の実現性を示すために, 予備評価を行なった. 本章では, その結果を示し, 仮想ネットワーク機構の有用性について確認する.

4.1 評価環境

評価はシミュレーションによって行なった. 図4に示す構造のソフトウェアシミュレータを作成し, アプリケーションを実行してその実行時間によって評価した. シミュレータは, Processor Emulator(PE), Network Interface Unit(NIU), Network Router(NR), Local Memory(LM) から構成される. 他のノードへのアクセスのためのインタフェースは, メモリアクセスをベースに実現されている (分散共有メモリシステム). 相互結合網としては, RDT(Recursive Diagonal Torus)[4] を用いた.

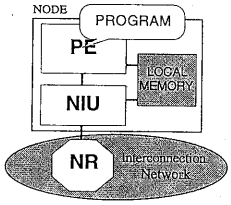

図 4: Structure of Simlulator

4.2 評価項目

評価は, ネットワーク番号エイリアスによる性能向上について行なった. ここでは, ノードと相互結合網とのスループットのみを向上させて, どのくらい全体の性能向上に寄与するかについての確認を行なった.

5 評価結果

アプリケーションとして行列演算 (行列とベクトルの積) を実行した. NR と NIU の間のスループットを向上させて, アプリケーションの実行時間 (計算時間のみ), および NR と NIU の稼働率を測定した. 横軸は NR-NIU のスループットで, 1 が相互結合網の NR 間のスループットと同じ, 2 はその 2 倍, 以下 8 倍まで増加させている. また, NR の稼働率は, NR が自ノードに対して送らなくてはならないパケットを持っている時にだけカウントしている (つまり, 通過するだけのパケットが NR 内部に存在していても稼働しているとは見なさない).

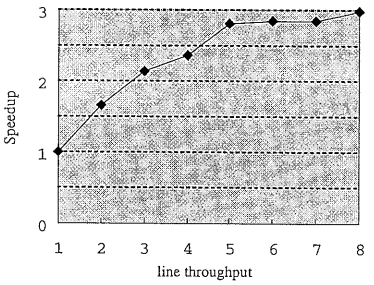

図 5: Execution Time

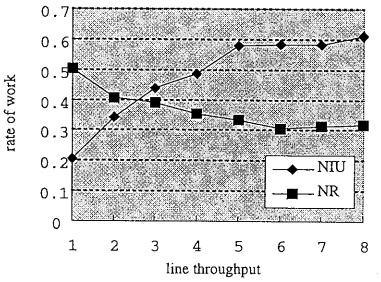

図 6: rate of NR and NIU's work time

図5から，相互結合網とノードの間のスループットを向上させると，あるレベルまでは良く性能が向上していることがわかる．この原因は，このアプリケーションが相互結合網に多大な負荷をかけており，ノードと相互結合網のやりとりが計算時間を決定しているからである．それは図6からわかる．スループット1のところでは，NIUの稼働率に比べてNRの稼働率がかなり高くなっていることがわかる．これはノードに送らなくてはならないパケットを持っているのに，ノードとのスループットが低くてパケットが詰まっている状態を表している．NR–NIU間スループットを向上させると差は縮まり，実行時間は短くなる．スループット5を越えると，もはやパケットが詰まるということはなくなり，NIUの稼働率は一定値に達し，性能向上は止まる．

このように，相互結合網とのやりとりが頻繁なアプリケーションでは，ノードと相互結合網のスループットを向上させることによってあるところまでは性能向上が図れることが確認された．

6 問題点と今後の課題

本手法では解決できていないいくつかの事項がある．

- 物理的な相互結合網の分割
  本手法では，ノード番号変換を行なうことによってプロテクションを実現するが，パケットがルーティングの都合で他のユーザの領域に侵入することまでは防ぐことはできない．ネットワークトラフィックが小さい場合は問題ないが，あるユーザが相互結合網に多大な負荷をかけている場合は，そのパケットの侵入が問題になる可能性もある．
- トポロジ変換の有効性
  論理トポロジを元にプログラムを最適化してしまうと，ネットワークトラフィックや通信のタイミングなどによって，逆に性能が低下してしまう可能性もある．またトポロジを意識したプログラミングが果たして有効なのかという疑問もある．

また今回の測定も，特定の場合のみについて測定したものであり，決して一般的な結果であるとは言えない．そのため，さらに一般的なアプリケーション，ネットワークで測定する必要がある．

7 まとめ

本稿では，マルチユーザ環境の元で，ネットワーク空間を仮想化する手法について提案し，その予備評価を行なった．

この手法はハードウェアでネットワークノード番号変換を行ない，それぞれのユーザごとに別々の空間を提供するものである．この手法では，

- ソフトウェアで変換を行なうのに比べて高速に変換できる
- 他のユーザ空間との間でプロテクションを実現できる
- ネットワークエイリアスを利用することによってノードと相互結合網との間のスループットをソフトウェア的に向上することができる
- ユーザに対して論理的なトポロジを提供できる

という特徴がある．

また，シミュレータを用いた予備評価を行なった．その結果，ネットワークエイリアスによるノード・相互結合網間のスループット向上によって相互結合網およびノードの負荷が均等化され，性能向上に寄与できることがわかった．

謝辞

本研究を行なうにあたって，シミュレータ製作および性能測定に協力していただいた富士通の三吉貴史氏および日立製作所の吉平健治氏に感謝します．

参考文献

[1] 日高康雄，小池汎平，田中英彦．PIE64における複合粒度並列処理を用いた最適粒度制御 –細粒度並列と粗粒度並列; 二つの異質な世界の分離と融合–．並列処理シンポジウム JSPP'95 論文集，pp. 115–122, 1995.

[2] Anoop Gupta, John Hennessy, Kourosh Gharachorloo, Todd Mowry, and Wolf-Dietrich Weber. Comparative evaluation of latency reducing and tolerating techniques. In *The 18th Annual International Symposium on COMPUTER ARCHITECTURE CONFERENCE PROCEEDINGS*, Vol. 19, pp. 254–263, May 1991.

[3] 松岡浩司，岡本一晃，廣野英雄，横田隆史，堀敦司，児玉祐悦，佐藤三久，坂井修一．超並列計算機RWC-1における．情報処理学会研究報告 93-ARC-101, Vol. 93, No. 71, pp. 17–24, 1993.

[4] 楊愚魯，天野英晴．超並列向きのプロセッサ結合網の提案．情報処理学会計算機アーキテクチャ研究会報告，No. 96-20, October 1992.