命令レベルシミュレーションによるadaptive routingの評価

上樂 明也　舟橋 啓　鯉渕 道紘　若林 正樹　天野英晴

慶應義塾大学理工学部

相互結合網におけるルーティングアルゴリズムであるadaptive routingは, 相互結合網の性能向上に大変有効な手段であると考えられており, 近年, 様々なadaptive routingの提案および評価が行なわれている. 既存の評価によりadaptive routingの有効性は確認されたが, その評価方式は実際のアプリケーションの通信パターンを反映しない確率モデルシミュレーションによるものがほとんどであった. そこで, 本研究では, 2次元トーラスにより各要素が結合された分散共有メモリ型並列計算機の命令レベルシミュレータを実装し, シミュレータ上でSPLASH2アプリケーション集を実行してadaptive routingの性能評価を行なった結果を示す. 本研究により現実的な通信パターンにおいてもadaptive routingが有効であることが確認された.

An Evaluation of Adaptive Routing Using Instruction-level Simulation

Akiya Jouraku　Akira Funahashi　Michihiro Koibuchi　Masaki Wakabayashi
Hideharu Amano

Keio University

Adaptive routing has been known as the best solution for high performance interconnection networks, and many researches on evaluation of adaptive routing algorithms have been exerted in recent years. However, most of these evaluations are only based on a probabilistic model simulation which doesn't reflect the characteristic of actual parallel applications. In this paper, we implemented an instruction-level simulator of distributed shared-memory multiprocessor system with interconnection network using instruction level simulator libraries. Through the evaluation using the developed simulator, it is shown that adaptive routing offers greater performance than deterministic routing in actual parallel applications.

1　はじめに

近年, 数百・数千台以上のプロセッサ要素から構成される大規模な並列計算機に関した研究, 実装が盛んに行われている. このような大規模並列計算機では, 各プロセッサ要素間を結合する手段として, 直接網や間接網などの相互結合網を用いて構成することが多い. 従来, 並列計算機用の相互結合網では, 目的ノードまで一定した経路やチャネルを用いる固定ルーティング (deterministic routing) が多く用いられていたが, 混雑や故障時に性能が低下したり, 経路を形成できなくなる問題点があった. そこで, 状況に応じて, 利用する経路やチャネルを切り替えることのできる適応型ルーティング (adaptive routing) の研究が進み [?], 最近の並列計算機ではこれらの機能を持ったルータが実際に用いられるようになっている.

これらの適応型ルーティングの評価は, 主として確率モデルシミュレーションにより行われてきた. これらのシミュレーションでは, 一つのノードに対する交信の確率を大きくすることにより, 混雑状態を確率的に実現する方法や, 行列の転置等特殊な交信パターンを確率的に実現する方法が用いられてきた.

このため, 短いシミュレーション時間で大規模なシステムの評価が取れる利点がある一方, 実際の並列プログラムにおける交信を本当に再現しているのかどうか疑問があった.

実際の並列プログラムにおける交信パターンを忠実に再現する方法として, 命令レベル方式のシミュレーションがある. この方式は, 個々のノードで実際に並列プログラムを実行する様子を命令レベルでシミュレーションすることにより, 実際に結合網中に流れるパケットの時刻と宛先を忠実に再現することができる. しかし, シミュレーション時間が大きくなることから, サイズの大きなシステムに対して利用することは困難であった.

しかし, 最近の PC/WS の性能向上によりシミュレーション速度が急速に向上すると共に, 命令レベルシミュレーションの方法も, より洗練され, 高速化が図られてきた. 我々の研究室でも, 高速な命令レベルシミュレーションの環境を, 簡単に構築することができるシミュレーション環境 ISIS[?] の整備を進めており, その一部として, 相互結合網シミュレータライブラリ SPIDER[?] が開発されている. これらのライブラリの組み合わせにより, 相互結合網で構成されたある程度のサイズの並列計算機の命令レベルシミュレーションが可能になっている [?].

本報告では, シミュレーション環境 ISIS のライブラリの一つとして, adaptive routing が可能なルータを実装し, これを用いることにより, いくつかのアプリケーションプログラムを実行して, 命令レベルシミュレーションによる adaptive routting の評価を行う. 第 2 節では, ここで対象とする adaptive routing である Turn model と Duato's Protocol を簡単に紹介する. 第 3 節はシミュレータの構成について述べ, 第 4 節にシミュレーション結果を提示して考察を行い, 第 5 節で結論を述べる.

2 adaptive routing

結合網におけるソース, デスティネーション間のパケット転送において, 決められた 1 つの経路しか選択できない deterministic routing に対し, adaptive routing は, 動的に複数の経路を選択することができるため, 混雑箇所および故障箇所の迂回が可能となる. これにより, ホットスポット発生の抑制およびチャネル利用率の向上が実現されるためスループットが向上し, また, 耐故障性を実現することも可能となるため, adaptive routing は, 並列計算機の性能に多大な影響を与える相互結合網の性能向上に大変有効な手段とされている.

近年, 様々なアプローチによりデッドロックフリーを保証した adaptive routing が数多く提案されてきたが, 本研究においては, その中でも代表的なものである Glass と Ni らによる Turn model[?] と Duato による Duato's protocol[?] を用いて評価を行なった.

2.1 Turn model

デッドロックが発生するのは, 結合網内におけるパケットが互いに他をブロックし合うことにより抜け道がない循環構造を形成してしまうためである. Glass と Ni らはこの点に注目し, パケットがルーティング中に方向を変えるパターンを制限することで循環の発生を防ぎデッドロックフリーを実現した Turn model を提案した.

ここでは最も単純な 2 次元メッシュでの Turn model を考える. まず, 2 次元メッシュにおいて可能な循環構造は図 ??(a) に示すように 2 種類になる. Turn model では各循環構造について, 循環が発生しないために最低限必要な数の進路変更を禁止する. ここでは図 ??(b) のように進路変更を禁止することにより循環構造の発生を防ぐ. この例では, 北方向への進路変更は必ず最後に行なわなければならないので特に north-last 法と呼ばれる. この他にも west-first などの方法がある.

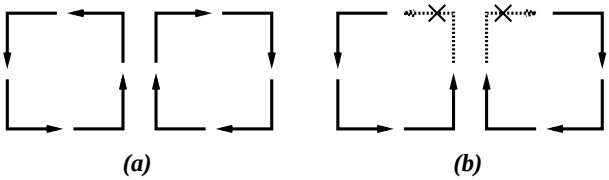

図 1: Turn model

Turn model では禁止されていない複数の進路変更を行なう限りは, 適応的に経路を選択することができる.

2.2 Duato's protocol

Turn model に代表されるように, 既存の adaptive routing のほとんどは, 何らかの手段により循環構造の発生を防ぐことによってデッドロックフリーを保証している. しかし Duato は, adaptive routing においては循環構造が発生してもデッドロックフリーを実現することが可能であることを証明して, デッドロックフリーな adaptive routing の必要十分条件を提案した. Duato の条件の概要は次のようになる.

a. 結合網全体に渡って循環の無い論理的な経路 (escape path) を用意する

b. escape path と,escape path により循環が切断されてデッドロックが起きなくなる他の経路 (adaptive path) によって結合網中のどのノード間でもパケットが送れるようにする

この 2 点を満足できれば a.,b. の経路を適応的に選択することにより, デッドロックフリーな adaptive routing が可能になる. Duato は, 以下の手順により,k-ary n-cube での fully adaptive routing を実現した.

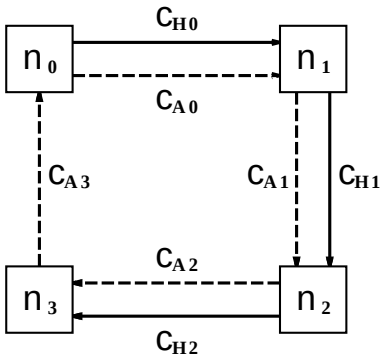

図 2: 単方向リング

1. はじめに単方向リング (図??) のネットワークについて考える. パケットが存在しているノードより目的地のノード番号が大きい場合は, チャネル C_H を使い, 低い場合は C_A を使用するという制限を設けることにより循環のない escape path が保証される.escape path が全てのノード間に存在するため, このネットワークはデッドロックフリーである.

2. 次に, ネットワークを双方向リングに拡張する. しかし,Duato の方法では, ルーティングは常に最短経路を通ること (minimal routing) から, ルーティングの途中でそれまで使用していたリンクと逆方向のリンクを使用することはありえず, 逆方向のリンクは全く別のネットワークとして分離して考えることができる. よって, 双方向リングのネットワークでもデッドロックフリーである.

3. 次に, ネットワークを多次元のものに拡張する. 次元の使用順を固定すると, 結局は双方向 (単方向) リングを決まった順に使用するだけになる. よって, ルーティングのときに使用されるチャネル間の依存関係は各次元で独立であるため, デッドロックフリーである. 以上により, k-ary n-cube における escape path が用意された.

4. 最後に, バーチャルチャネル C_F を新たに用意し, そのチャネルでは次元の使用順を無視して移動可能とする. これにより, デッドロックフリーな fully adaptive routing が可能となる.

ただしこのルーティングでは,2. を満足させるために, 常に最短経路を通るという minimal routing を守らなければならない.

3 命令レベルシミュレータの実装

本研究の目的は, 実際のアプリケーションを利用した命令レベルシミュレーションを行ない, 相互結合網における adaptive routing の評価を行なうことである. 本研究ではこれを実現するために並列計算機シミュレーションライブラリ ISIS と結合網シミュレーションライブラリ SPIDER で提供されている機能ブロックを用いる. まず, この二つのライブラリについて述べる.

3.1 並列計算機シミュレータライブラリ ISIS

並列計算機シミュレータライブラリ ISIS は, 並列計算機の性能評価を目的として慶應義塾大学で開発されたシミュレータライブラリである. ISIS は, 並列計算機を構成する個々の機能ブロックをシミュレートする「部品」を多数集めたライブラリであり, 各機能ブロックはユニットとして個別に実装されている. 必要なユニットを繋ぎ合わせることにより所望の並列計算機シミュレータを構築することができるため, ISIS はアーキテクチャに依存することがなく, アーキテクチャの変更によるシミュレータ再実装等のコストを大幅に削減可能としている. また, シミュレーション方式として, 確率モデルシミュレーション, トレース駆動型シミュレーション, 命令レベルシミュレーションのどの方式にも対応可能となっている.

ISIS は実装言語として, オブジェクト指向をサポートし高速動作が可能な C++言語を用い, 各ユニットをクラスライブラリの形で提供している. ユニット間の共通性を基底クラスとして抽出することにより, 各ユニットはそれぞれ個別のクラス階層を形成しており, 用意されていないユニットについても継承を用いることにより比較的容易に実装を行なうことができるように設計されている. ユーザは, これらのユニットを評価する対象に合わせて選択, 使用することによりシミュレータの実装を容易に行うことができる.

ユニット間の結合と通信のインタフェースには, 情報を仲介する存在としてパケット, 結合を仲介する

存在としてポートの概念が導入されている．パケットは，相互接続されたユニット間でやり取りされる全ての情報を表現する．例えばルータ間で送受信されるフリット，プロセッサがバスに出力するメモリアクセス要求などがこれに相当する．ポートは，ユニット同士を結合している通信路への入出力端子を表現する．接続したいユニット同士のポートを接続することで，ユニット間の通信路が構築される．

現在，MIPS 社の R3000 プロセッサ，R3010 浮動小数点演算コプロセッサ，バス結合型並列計算機用のスヌープキャッシュ等のユニットが実装されている．また，SPLASH2 プログラム集 [?] をシミュレータ上で実行するための環境も整備されている．ISIS は，シミュレーションを実行している際の各プロセッサのメモリアクセスを監視し，シミュレーションの実行結果としてアドレストレースファイルを生成する機能を備えているため，他のトレース駆動型シミュレータへのデータ入力システムとしても使用することが可能となっている．ISIS は小規模並列計算機の性能評価を目的として開発されたため，ISIS が提供するライブラリのみでは，各ユニットの結合形態はバス結合のみに限定されている．

3.2 相互結合網シミュレータライブラリ SPIDER

SPIDER は，相互結合網の性能評価を目的に慶應義塾大学で開発されたシミュレータライブラリである．SPIDER は，相互結合網を構成する各機能ブロックを C++のクラスライブラリの形で実装し，ISIS と同様なシミュレータ構築支援環境を提供している．また，命令レベル方式での評価を行なうことを考慮して ISIS が提供する幾つかのユニットを使用している．

SPIDER は相互結合網を構成する主な機能ブロックであるパケット，ルータ，ネットワークインタフェースそれぞれについて，共通機能を抽出した基底クラスおよび実装済の派生クラスを提供している．パケット転送方式としては一般的な wormhole routing および virtual cut-through がサポートされており，結合網のトポロジおよびルーティングアルゴリズムは，SPIDER が提供する基底ルータクラスの派生クラスにおいて実装者が手続き指向で容易に記述することができるようになっており，様々なトポロジおよびルーティングアルゴリズムを容易に実装することが可能となっている．ネットワークインタフェースとしては，確率モデルシミュレーションを行なうための一様乱数に基づいてアクセスを発行するユニットと命令レベルシミュレーションが可能な ISIS で提供しているユニットで構成したノードをルータに接続するためのユニット [?] がそれぞれ実装されている．後者のユニットは，ノード内バスとルータ間のパケットの相互変換，フロー制御を行い，また，分散共有メモリをサポートする．このユニットを用いることにより相互結合網により各ノードが結合された分散共有メモリ型並列計算機の命令レベルシミュレーションが可能となっている．

3.3 ターゲットマシンの構成

本報告における adaptive routing シミュレータでは，各ノードが相互結合網により接続された分散共有メモリ型並列計算機を命令レベルでシミュレーションすることにより，相互結合網上での adaptive routing の効果を評価する．

この分散共有メモリ型並列計算機の命令レベルシミュレータの機能ブロックを構成を図 ??に示す．プロセッシングユニット部分には，R3000 プロセッサ，R3010 浮動小数点演算コプロセッサ，キャッシュ，共有メモリなどの命令レベルシミュレーションが可能な ISIS で提供されている機能ブロックを利用し，相互結合網部分にはルータおよびネットワークインタフェースなどの SPIDER で提供されている機能ブロックを利用している．相互結合網のトポロジは 2 次元トーラスを用いている．

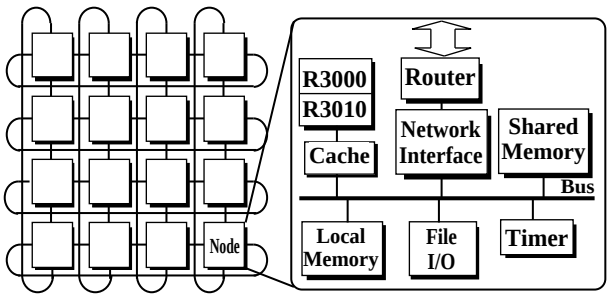

図 3: 命令レベルシミュレータの機能ブロック図

命令レベルシミュレータの実行ファイルは，必要なライブラリを用いてトップモジュールを記述し，コンパイル，リンクをすることにより生成される．

3.4 adaptive routing の実装

SPIDER が提供するルータクラスにおいては，ルーティングアルゴリズムに関するインタフェースのみが定義されているため，ユーザが派生クラスを作成してルーティングアルゴリズムの実装を行なうようになっている．本研究では，2 次元トーラスにおける 1 つの deterministic routing と 2 つの adaptive routing をそれぞれ派生クラスを作成して実装した．

各ルーティングアルゴリズムの概要は次のようになっている.

- e-cube routing
  - X,Y 方向の順に最短経路によるルーティングを行なう deterministic routing
  - バーチャルチャネルを各方向に 2 本用意
  - ルーティングの自由度 (選択可能なバーチャルチャネルの数) は 1
- Turn model
  - north last 法を用いる
  - バーチャルチャネルを各方向に 3 本用意
  - 最短経路のみを使用
  - partially adaptive
  - ルーティングの自由度は 1〜2
- Duato's protocol
  - バーチャルチャネルを各方向に 3 本用意
  - ルーティングの制限は第 2 節で説明した通り
  - fully adaptive
  - ルーティングの自由度は 1〜4

4 評価

実装した命令レベルシミュレータ上で 2 つの SPLASH2 アプリケーション集を各ルーティングアルゴリズムにより実行した. 以下にその結果を示し, 検討を行なう.

シミュレーション条件

シミュレーションにおける条件は次の通りである.

- 2 次元トーラスのサイズは 16(4×4) および 64(8×8)(ただし RADIX は 16 のみ)
- 物理チャネルのバンド幅は 1 [flit/clock]
- パケット長は 8 flit に固定
- ルーティングに 1 clock
- アービトレーションに 1 clock
- パケット転送方式は wormhole routing および virtual cut-through
- 各ノードから結合網内に投入されているリクエストパケットの数は常に最大で 2
- ネットワークインタフェースにおける送受信用バッファのサイズは, 32 packet(16 ノード構成時) および 128 packet(64 ノード構成時)
- チャネルバッファのサイズは 1, 2, 4, 8 と変化させてそれぞれ評価を行なう

使用したアプリケーションは SPLASH2 アプリケーション集の FFT と RADIX であり, FFT の問題サイズは 4096, RADIX の問題サイズは 32768 とした.

4.1 FFT による評価

16 ノード構成時および 64 ノード構成時における結果を図 ??と図 ??にそれぞれ示す.

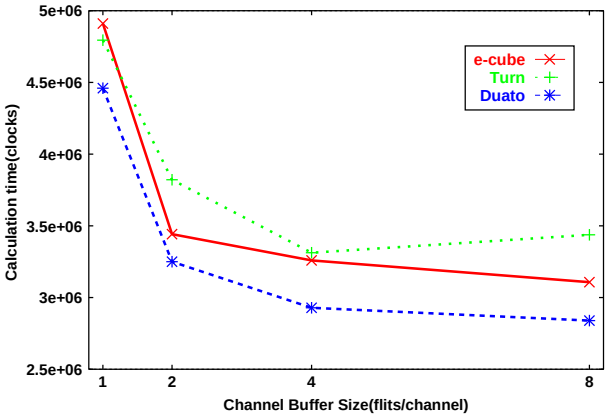

図 4: FFT における計算時間 (16 ノード構成時)

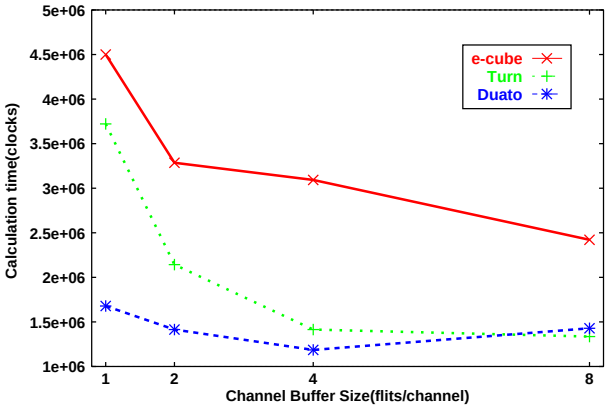

図 5: FFT における計算時間 (64 ノード構成時)

それぞれの図において, 横軸はチャネルバッファ容量, 縦軸は初期化を除くアプリケーションの総実行時間 (並列処理部分の計算に要した時間) を表している. これらの図より, Duato's protocol はどちらのサイズにおいても e-cube routing に対して計算時間の短縮を実現していることが分かる. 計算時間の短縮の度合は, 16 ノード構成においては約 5〜10%ほどであるが, 64 ノード構成においては約 42% 〜63%と非常に大きなものとなっている. これは, ノードのサイズが増加するとパケットの平均 hop 数が大きくなるため, adaptive routing において選択可能な経路の数が増加したことが大きく影響していると思われる. Turn model も 64 ノード構成においては, 最

大で約 55%の大きな計算時間の短縮を実現しているが, チャネルバッファサイズが 1, 2 の時には計算時間短縮の度合は小さくなっている. また, 16 ノード構成においては e-cube routing とほぼ同等もしくはそれより長い計算時間を要している場合もある.

4.2 RADIX による評価

16 ノード構成時における結果を図 ??に示す. 各座標軸が示す値は FFT の場合と同様である. 図 ??より, e-cube routing と Turn model にはほとんど計算時間に差がみられないことが分かる. RADIX における通信パターンでは, ある 1 つのノードに他のノードからのパケットが同時に集中的に送られるというような局所的なアクセスが多くなっているため, 目的ノードに各方向から同時に複数のパケットが到着するというような状態が発生しホットスポットが発生しやすくなっていると考えられる. Duato's protocol もチャネルバッファサイズが 1, 2 の時にはこの影響を受けていると考えられるが, チャネルバッファサイズが 4, 8 の時には他に比べて約 10%〜23%の計算時間の短縮を実現している.

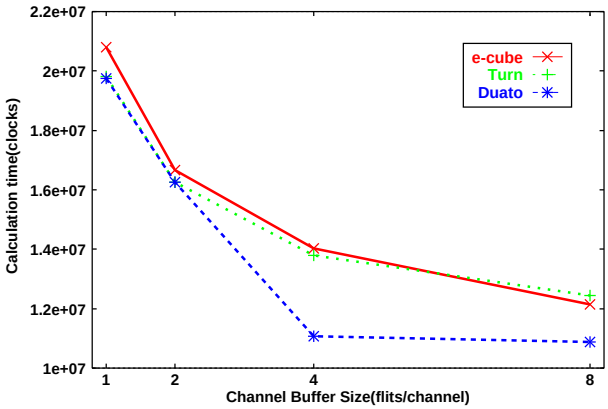

図 6: RADIX における計算時間 (16 ノード構成時)

5 結論

本研究では 2 次元トーラスにより各要素が結合された分散共有メモリ型並列計算機の命令レベルシミュレータを実装し, シミュレータ上で SPLASH2 アプリケーション集の FFT と RADIX を実行することにより, 現実的な通信パターンにおける adaptive routing の性能評価を行なった. シミュレーションの結果, Duato's protocol を用いることにより確実に性能向上を実現することが可能となるということが分かり, これにより現実的な通信パターンにおいても adaptive routing が有効であるということが確認された.

参考文献

[1] J.Duato, S.Yalamanchili, L.Ni, "INTERCONNECTION NETWORKS an Engineering Approach", IEEE Computer Society Press.

[2] C. J. Glass and L. M. Ni, "Maximally Fully Adaptive Routing in 2D Meshes", Proceedings of International Conference on Parallel Processing, pp278-287, Aug. 1992

[3] J.Duato, "A Necessary and sufficient condition for deadlock-free adaptive routing in wormhole networks", ICPP94. pp.1142-1149

[4] 若林 正樹, 米田 卓司, 天野 英晴,"並列計算機シミュレーションライブリの提案", 電子情報通信学会, CPSY 97, Dec. 1997

[5] 米田 卓司, 若林 正樹, 緑川 隆, 西村 克信, 天野 英晴,"並列計算機のための相互結合網シミュレータ SPIDER", 電子情報通信学会, CPSY 98, Jan. 1998

[6] 小守 継夫, 若林 正樹, 天野 英晴, "相互結合網評価用命令レベルシミュレーション", 情報処理学会研究報告, HOKKE-99, Mar. 1999

[7] Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, Anoop Gupta, "The SPLASH-2 Programs: Characterization and Methodological Considerations", Proceedings of the 22nd International Symposium on Computer Architecture, pp 24–36, June 1995.