2K-2

GPGPUによる大規模相互結合網シミュレーション

澤野 遼太郎 † 横田 隆史 † 大津 金光 † 大川 猛 † 馬場 敬信 †
† 宇都宮大学工学部情報工学科

1 はじめに

近年の超高性能計算機は大規模な並列構成が主流となっており，ノード間における通信性能は計算機性能全体に大きく影響する．このため，ネットワークトポロジをはじめ，フロー制御，ルーティング等の手法の研究が行われている [1]．通信性能においては，輻輳の制御が非常に大きな影響を及ぼし，輻輳が発生すると著しく性能低下を起こしてしまう．この問題に対処するには輻輳の発生から性能低下として現れるまでのメカニズムを解明する必要がある．そこで，我々は，セルオートマトンを用いて，相互結合網における輻輳の発生・成長のメカニズムを明らかにした [2]．

しかし，大規模な相互結合網におけるシミュレーションには膨大な時間がかかる問題がある．そこで本稿では，シミュレーション時間の短縮を図るために，セルオートマトンを用いたシミュレーションにおけるデータ並列性に着目し，GPGPU によって高速化を試みた結果を示す．また，直感的かつ容易にネットワークの状況を把握するために可視化を行い，即座に確認できるようにした．

2 CPU 版ネットワークシミュレータ

本稿でのネットワークシミュレータのモデルはセルオートマトンを用いる．セルオートマトンとは，自己増殖システムや複雑なシステムをモデル化するのに適したオートマトンであり，交通流シミュレータなどにも用いられている．従来研究で，我々は相互結合網のモデル化を行った [3]．今回は GPU を使うことを前提としたモデル化を行った．まず，各セルで次の状態を求め，全てのセルで求まった後に，次の状態へ更新，パケット消失，パケット生成という 4 ステップで処理を行う．また，次状態を求める前の更新を防ぐことで，依存違反させないようにしている．

表 1: ネットワークシミュレータの概要

項目	パラメータ
システムモデル	セルオートマトン
ルーティングアルゴリズム	Dimension-order
ネットワークトポロジ	二次元メッシュ/トーラス
デッドロック防止法	仮想チャネル
ノード数	最大 100 万ノード
パケット生成方法	一定間隔で生成

表 1 にセルオートマトンによりモデル化したシミュレータの概要，また図 1 にシミュレータの全体像，図 2 に 1 ノード分を示す．図 2 に示したノードは，channel と switch から構成される．switch 部で横と縦どちらの channel からの入力を許可するか判断する．

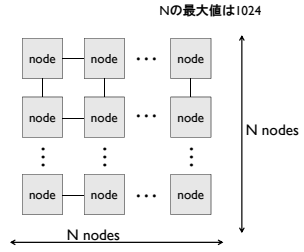

図 1: シミュレータの全体像

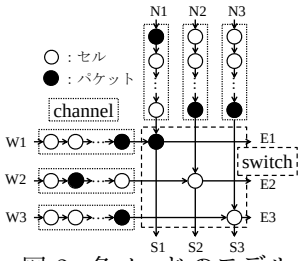

図 2: 各ノードのモデル

本稿のネットワークシミュレータでは，ルーティングアルゴリズムやネットワークサイズなどのパラメータ設定を可能とし，ネットワークのシミュレーションはサイクル単位で行うこととした．パケットの初期値はランダムで，セルオートマトンによるセルの更新ルールは [3] を用いた．

表 2 に CPU 版ネットワークシミュレーションの実行時間を示す．実行サイクル数は 10,000 サイクルで，評価環境は，Intel Core i7 940 2.93GHz である．表 2 より，ノード数が多くなるとシミュレーション時間が長くなり，ノード数 1024 × 1024 のとき約 2 時間近くかかってしまい，即座にネットワークの状況を確認するのは困難である．従って，GPU で並列化を行い高速化する必要がある．

表 2: CPU 版シミュレータ実行時間

ノード数	実行時間 (秒)
1024 × 1024	7095.28
512 × 512	1459.26
256 × 256	376.86
128 × 128	90.62

3 GPU での実装

本稿では，GPGPU プログラムの開発環境として NVIDIA 社の CUDA を用いる．GPU は CPU に比べ数多くの演算器を持ち，プログラムのデータ並列性を利用して，SIMD 計算により大幅に実行時間を短縮することができる．この演算器を CUDA コアと呼び，複数の CUDA コアが SIMD 形式で実行される．この複数の CUDA コアと命令機構を合わせて SM (StreamingMultiprocessor) と呼ぶ．この 1 つの SM を複数動

A GPGPU Approach for Large-Scale Interconnection Network Simulation
†Ryotaro Sawano, Takashi Yokota, Kanemitsu Ootsu, Takeshi Ohkawa and Takanobu Baba
Department of Information Science, Faculty of Engineering, Utsunomiya University (†)

作させることで大量の演算を一度に処理することが可能となっている．本稿ではこの並列性に着目し，ネットワークシミュレータの高速化を図る．

2節で記述したように，本稿のネットワークシミュレータはセルオートマトンにより，4ステップで処理を行っている．また，本稿のシミュレータの設計では，依存違反を排除できる．そのためGPUでは，この4ステップ部分の並列化を行う．

まず，GPUではネットワークサイズ分のスレッドを用意する．ネットワークサイズを1024 × 1024にした場合は，約100万個のスレッドを用意することとなる．各スレッドはスレッド番号を持ち，この番号と対応するノード番号のノードを割り振る．スレッドは割り振られたノードの処理を行う．次状態の判定では，パケットの移動ルールを元に次状態を求め，次状態を記憶しておくバッファに格納する．全てのスレッドで次状態を求め終わり次第，状態更新を行う．状態更新では，バッファに記憶していた次状態を元に，パケットの移動を行う．ノード間をまたぐ状態更新では，パケットの移動先ノードを処理するスレッドが処理を行う．次状態を求めるフェーズと，状態更新フェーズの間でスレッドの同期を取るのは，次状態を求める前に更新してしまうのを防ぐ為である．状態更新が全てのスレッドで終わり次第，次のステップのパケット消失を行う．パケットの消失は各ノードのswitch部で行う．そして最後にパケットの生成を行う．パケットの生成もswitch部で行う．パケットの消失とパケットの生成の間では同期を行わず，各スレッドで非同期にパケットの消失，パケットの生成と行っていく．この4ステップを指定したサイクル数分繰り返す．また，結果出力はCPUが任意のサイクルでGPUとは非同期に取り出す．

4 評価

表3にGPUで実装したシミュレータの実行時間と，CPU版と比較した速度向上比を示す．実行サイクル数は10,000サイクルで，評価にはNVIDIA社Tesla C2075 1.15GHzを使用した．表より，CPU版と比べ最大で約6.4倍の速度向上を達成し，シミュレーション時間を大幅に短縮した．しかし，デバッガでスレッドの動きを確認した際，ワープダイバージェントを起こしていることが確認できた．今後はワープダイバージェントを起こさないようシミュレータを改良していくことが課題である．

表 3: GPU版シミュレータ実行時間

ノード数	実行時間 (秒)	速度向上比
1024 × 1024	1100.41	6.44
512 × 512	466.82	3.12
256 × 256	110.92	3.39
128 × 128	25.58	3.54

5 シミュレーション結果の可視化

本稿では各ノードのパケット分布状況を可視化することとした．パケットの分布状況を可視化することで輻輳の発生・成長メカニズムを直感的に把握することができる．どの位置でどの程度の混雑が発生しているかをより正確にとらえられるよう，各ノードごとのパケット数を計測し，ファイルに出力する．出力されたデータを元に，パケット数の度合を色の濃淡に変換し，画像ファイルを生成して，複数の画像ファイルをつなげた動画ファイルとして出力する．実行中は生成された画像を画面にも出力し，動的にネットワークの状態を確認できるようにした．

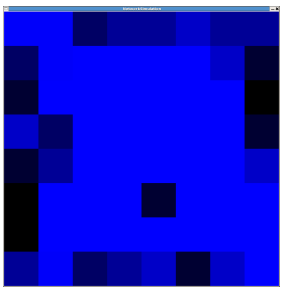

図 3: ノード数 8 × 8

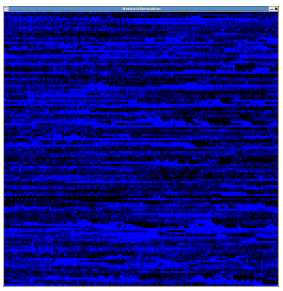

図 4: ノード数 128 × 128

図3はノード数8 × 8の相互結合網でのシミュレーション出力結果，図4はノード数128 × 128の結果である．それぞれ1,000サイクル目の様子であり，ルーティングアルゴリズムはDimension-orderを用いている．出力画面サイズは1024 × 1024で，ネットワークサイズに合わせてノード部分のサイズを変更した．色が淡い方がパケット数が多い状態である．図4では横方向に色の淡い部分が多く見られ，輻輳状態が複数箇所に見られる．このように可視化を行うことで，相互結合網における輻輳状態を確認することができた．

6 おわりに

本稿ではセルオートマトンによる相互結合網シミュレータを，GPGPUで高速化する手法を提案し実装を行った．また，視覚により即時にネットワークの状況を確認できるよう，可視化システムの開発を行った．今後はシミュレータの改善を行い，より大規模のネットワークシミュレーションが行えるようにする．

謝辞

本研究は，一部日本学術振興会科学研究費補助金 (基盤研究 (C)24500055，同 (C)24500054) の援助による．

参考文献

[1] William James Dally and Brain Towels: “Principles and Practices of Interconnection Networks,” Morgan Kaufmann Publishers, 2004.

[2] 横田隆史ほか: “セルオートマトンによる相互結合網の輻輳の解析,” 情報処理学会論文誌:コンピューティングシステム, Vol.47, No.SIG 7(ACS 14), pp.21-42, 2006.

[3] 横田隆史ほか: “セルオートマトンによる大規模相互結合網シミュレーションの試み,” 信学技報, Vol.105, No. 453, pp.31-36 (CPSY2005-32), 2005.