B-028

メッセージフロー方式に基づく通信ネットワークシミュレータとその並列化
A Communication Network Simulator Based on Message Flows and I'ts Parallelization

鈴木 悠太郎† 高上 治之† 矢崎 俊志† 石畑 宏明†

Suzuki Yutaro† Takaue Haruyuki† Yazaki Syunji† Ishihata Hiroaki†

1. はじめに

並列計算機では多数の計算機を接続し通信を繰り返しながら処理を進める。現在では数千から数万ノード級のシステムが稼動しているが、ノード数が多いほどネットワークがボトルネックになるため問題となっている。そのためアプリケーションで頻繁に使用される特定の通信パターンの通信時間を見積もることは重要である。

通信時間はシミュレーションにより見積もっているが、従来のシミュレーションでは通信メッセージをパケットに分割し、ネットワーク上のパケットを追跡することで行っているため、数万ノードのシステムのシミュレーションには長大な時間が掛かる[1][2]。

これに対して我々は、メッセージのフローに着目してその流量を元に通信時間を算出することでシミュレーション時間の短縮を図ったシミュレータを提案している[3]。このシミュレータをさらに高速化するために SMP 上で並列処理を行う。

本報告ではこのシミュレータの並列化による速度向上効果について述べる。

2. シミュレータについて

2.1 シミュレータの動作

本方式ではメッセージを流体として扱い、その流量がコンテンション等の通信経路の状態により制限されるモデルに基づいてシミュレーションを行う。ノードペア間の通信をフローと呼び、同一のリンクを通過するフローは、その通過するリンクのバンド幅を等分して使用するものとする。各ノードは送信するメッセージをキューとして持つ。このシミュレータの動作概要を以下に示す。

- 与えられた通信パターン中の各メッセージの経路を、指定されたネットワークトポロジ上で探索する。
- ネットワーク中の全リンクについて、そこを通過するフローを重ね合わせる。リンクにおいて各フローの使用可能なバンド幅は、リンクの元々のバンド幅を重なったフロー数で割った分となる。フローの流量は、通過する全リンク中のフローあたりのバンド幅の最小値とする。
- 求めたバンド幅から残っているサイズ分のメッセージの転送に必要な時間を算出し、その分だけシミュレーションステップを進める。

以上の処理を全メッセージの転送が終了するまで繰り返す。

2.2 シミュレータの実装

以下にシミュレータの実装を示す。ネットワーク上を流れる通信メッセージをフローと呼ぶ。

2.2.1 初期化処理

ネットワークトポロジの構成、通信パターンを読みこれらを生成する。各フローはメッセージ長をパラメータとして持つ。

2.2.2 メッセージ処理ループ

step1. 全ノードについて送信メッセージリストを作成する。メッセージがない場合にはループを抜け終了する。

step2. 全リンク中のフローの重なりを数えるカウンタ(*mcnt*)を 0 にする。

step3. 全フローについて経路を探索し、通過するリンク中の *mcnt* を増加させる。

step4. 全リンクについて、与えられたリンクのバンド幅を *mcnt* で割った値をそのリンクにおけるフローのバンド幅として設定する。

step5. 全フローについて、残りメッセージ長の完了に必要な時間を step4 で求めたバンド幅から算出する。

step6. 全フローから、step5 で求めた最小値を探す。

step7. step6 で求めた最小値の分だけシミュレーションステップを進め、各送信メッセージの残りサイズを減少する。

2.3 シミュレータの並列化手法

step1 から step7 までの各ステップは、全て直前のステップの結果が必要なためステップを超えた並列化はできない。そのため各ステップごとに並列化をする必要がある。スレッドの生成・起動コストを最小限に抑えるため各ステップでスレッド化するのではなく、メッセージ処理ループ全体を関数として実装し、これを複数スレッド上で並列に実行する。

step1、step3、step5、step6 ではノードが持つ要素についての処理であるため、各スレッドはノード方向に分割した担当範囲について処理をする。

step2、step4 はリンク中の要素についての処理である。ここでは全体のリンクを分割し、各スレッドは担当するリンクについて処理する。各ステップは直前の結果が必要であるため、ステップ間にはバリア同期が必要となる。関数呼び出し、アトミック操作、バリア同期には OpenMP を使用した。

† 東京工科大学 Tokyo University of Technology

3. 並列化の結果

3.1 シミュレーションの実行結果

実験は Opteron 1352(物理 4 コア 2.1Ghz) を 2 ソケット搭載した主記憶 2GB のサーバ(OS CentOS 64bit、コンパイラ g++ 4.1.2)で行った。通信パターンは、ノードを円状に並べ、その中からランダムにノードを選び、一筆書きで全ノードが通行するように通信するランダムリングパターン、各ノードはメッセージを 10 個送るものとした。ネットワークトポロジはフルバイセクションバンド幅の Fat tree、表 3.1 に示す各構成でシミュレーションを行った。トポロジは 2n 個のポートを持つスイッチ 3 段により構成され、$n \times n \times 2n$ で表記した。図 3.1 にシミュレーション結果を示す。

表 3.1 トポロジ構成

	ノード数	リンク数
8x8x16	1024	3072
12x12x24	3456	10368
20x20x40	16000	48000
24x24x48	27648	82944
32x32x64	65536	196608

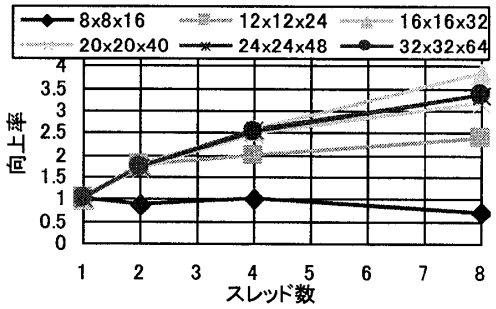

図 3.1 シミュレーション結果

12x12x24 構成ではどれも近い向上率を得ているが、8x8x16 構成ではスレッド数の増加に伴い速度が落ち、スレッド数 8 では約 0.5 倍まで低下している。原因としてバリア同期のコストが考えられたためこれにかかる時間を測定した。

3.2 バリア同期のコスト

OpenMP 標準のバリアと、ビジーウェイトによるバリアを自作し比較した。バリアを 100 万回呼び出すループの中でバリアの直前に加算を繰り返すループを置き時間を測定し、バリア無しの実行時間を引いた上で 100 万分の 1 にすることでバリア 1 回あたりのコストを求めた。比較結果が図 3.2 である。

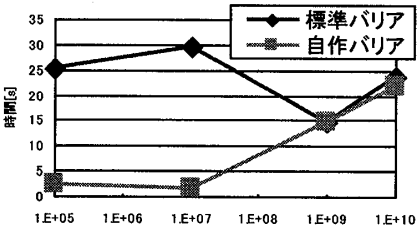

図3.2 バリア同期のコスト

1000 万回の加算ループ程度のコストまでは自作バリアが約 20 倍の速度を出している。ループ 10 億回から標準との差がなくなっていき、更にループ回数が多くなると標準バリアが高速になっている。バリアを自作のものに置き換え測定をした。

3.3 改善後のシミュレーション結果

バリアを変更し測定し直した結果を示す。

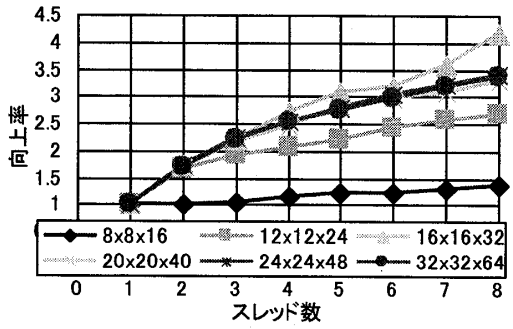

図3.2 改善後のシミュレーション結果

スレッド数 2 の結果は約 1.5 倍、スレッド数 4 で約 2.5 倍、8 スレッドでは高い場合でも約 4 倍にとどまった。12x12x24 構成以上ではどれも近い向上率となっているが、8x8x16 構成ではスレッド数 8 でも 1.5 倍に止まっている。このことから本並列化手法では十分なノード数がなければ性能向上を得にくいとわかる。

バリアを標準から自作のものに変更した顕著な影響は 8x8x16 構成のシミュレーションにしか見られない。図 3.2 のバリアコストから 8x8x16 構成の処理量はループ 10 億回分未満、12x12x24 構成以上だとループ 10 億回分以上の処理量となり向上率に影響しなかったと考えられる。

4. 結論

並列化により一定の速度向上は得られたものの、並列化した各ステップの粒度が小さいため大きな向上とはならなかった。より高いスケーラビリティを得るためにはバリアの不要な処理方式を考案し実装する必要がある。

今後はより詳細な各ステップにおける実行時間及び各スレッドの負荷バランスを調査し、シングルスレッド性能を犠牲にしてスケーラビリティを取るなどの実装変更も視野に入れて高速化を図る。

参考文献

[1] N.Choudhury, Y.Mehta, T.L.Wilmarth, E.J.Bohm, L.V.Kale, "Scaling an Optimistic Parallel Simulation Conference of Large-scale Interconnection Networks", Proceedings op the Winter Sinmulation Conference, pp.591-600, (Dec 4-7, 2005)

[2] T.L.Wilmarth, G.Zheng, E.J.Bohm, Y.Mehta, N.Choudhury, P.Jagadishprasas, L.V.Kale, Performance Prediction using Simulation of Large-scale Interconnection Networks in POSE, In Proceedings of the Workshop on Principles of Advanced and Distributed Simulation, 2005, pp.109-118

[3] 石畑 宏明, "大規模並列コンピュータ用通信ネットワークのシミュレーション方式", 信学技報, CPSY2008-20, pp.55-60(2008-08)