ハイパフォーマンス
コンピューティング 48−17
（1993. 8. 20）

離散事象型並列シミュレーションにおけるマッピング手法

高井峰生 成田誠之助

早稲田大学理工学部

離散事象型シミュレーションには大きな並列性が存在するが、論理シミュレーション以外では有効な手法が提案されていない。大規模システムのシミュレーションにはトランザクション中心のモデル構成概念が一般的な事から、本稿ではトランザクション中心のモデルに対する静的なマッピング手法を提案した。この手法はトランザクションフローグラフを用い、実行時のモデルの負荷を予想して各PEに負荷を均等に分配する。実験として仮想駅を想定し、AP1000を使って実際にシミュレーションを行なった。各PEでは、入出力だけを考慮し、逐次処理と同様のアルゴリズムで処理したが、従来に比べて大きく処理効率が向上した。

A Mapping Method of Parallel Discrete Event Simulation

Mineo Takai　Seinosuke Narita

School of Science and Engineering, Waseda University

Discrete event simulation involves much parallelism, but many schemes suffer from pour efficiencies except for logic simulation. In large scale simulation, a simulated system is transformed into a transaction oriented model. This report proposes a static mapping method for transaction oriented models. It employs transaction flows for estimating an execution load and levels average load on each PE. A virtual station model was simulated on the parallel machine AP1000 to verify the mapping method. Each PE processes like a sequential simulator, with only consideration given to I/O messages. The proposed method was proved to be more efficient than conventional approaches.

1 はじめに

離散事象型シミュレーションはさまざまなシステムの有効な分析手段として利用されている。近年、生産や計算機、通信システムのようなシミュレーション分析を必要とする分野の複雑・大規模化に伴い、ますますその重要性が増してきた。しかし、このような複雑・大規模なシステムをシミュレーションする場合、従来から用いられている離散系汎用シミュレーション言語では、莫大なシミュレーション時間が必要となる。しかし、多大な時間を要する大規模なシステムになるほど、システム内には並列性が存在することが多く、この問題を並列処理により解決する方法は、以前から多くのものが提案されてきている [10]。ところが、論理回路シミュレーション等、特定分野のシミュレーションでは成果が上がっているものの、他の多くの分野ではその有効性が認められていないのが現状である。

離散系汎用シミュレーション言語のほとんどは、モデル構成概念としてトランザクション中心のモデルが扱える[1]。これは大規模なシステムをシミュレーションする際、モデル構成としてトランザクション中心のモデル化を行なうのが一般的なことによる。そこで、本稿ではトランザクション中心のモデルをシミュレーションする際の並列処理手法について、負荷分散を考えた効果的なマッピング手法を提案し、並列処理計算機 AP1000 を用いてその有効性を確認する。

2 従来の並列処理手法

従来の離散系並列シミュレーションの研究は、時刻管理の形態で集中時刻管理と分散時刻管理の 2 つに大別できる。

集中時刻管理は、各 PE の時刻を一致させてシミュレーションを進めることにより、シミュレーション時刻の非単調減少を保証するものである。この管理法は論理回路シミュレーション、特に信号値が High,Low の 2 値でユニット遅延のモデルなど、同時刻に多くの事象が発生するような特別なシミュレーションでは有効である。しかし、時刻を集中して管理することにより、PE 数の増大に伴って同期のオーバーヘッドが著しく増加するため、専用機でないと大規模なシステムには適用できないと思われる。

分散時刻管理は各 PE で時刻を独立して管理する手法であり、システムの並列性が失われないのが特徴である。しかし、シミュレーション時刻の非単調減少を保証するため、各 PE がそれぞれ他の PE の入力メッセージを待ち、デッドロック状態に陥ることがある。また、発生確率が低いメッセージを待つことにより、処理効率が著しく低下する場合がある。これらを回避するために保守的手法 [7][8] と楽観的手法 [9] が提案されている。

保守的手法は入力される事象の時刻を用い、今後入力されないと保証される時刻までの事象のみ処理を行う。デッドロックの発生する可能性のある場合、出力する事象のない場合でもヌルメッセージと呼ばれる空のメッセージを出力したり、一定時間ごとに同期をとったりする必要がある。

これに対し、楽観的手法は事象カレンダーに登録されている事象は全て処理し、時刻の矛盾が起こった場合はシステムの状態を矛盾の発生した時刻まで戻し、処理を進めていく。このため、楽観的手法では処理の正当性が保証されていない時刻の事象処理については、履歴を保存しておかなければならない。

分散時刻管理では、これらのオーバーヘッドを小さくする事が効率を上げるポイントになる。

3 並列処理手法

トランザクション中心のモデルを並列シミュレーションする場合、集中時刻管理を用いると効率的な処理ができないことが予想される。そこで、分散時刻管理を用い、並列処理に関するオーバーヘッドを低減しなければならない。

分散時刻管理で従来から提案されてきた、保守的手法・楽観的手法は、主にデッドロックを回避し、メッセージ待ちによる PE のアイドル時間を

[1] トランザクション中心のモデル記述をする言語で代表的なものには GPSS,SLAM などがある。また、イベント中心の記述をする SIMSCRIPT もトランザクション中心のモデル記述が可能である。

減少させることにより処理の効率化を計るものである。しかし、シミュレーションモデルを各PEに静的にマッピングする場合、これらの手法はマッピングの仕方によって効率が大きく左右することが知られている[4][5]。シミュレーションモデルを負荷を均一にマッピングしない場合、シミュレーション全体の処理時間はもっとも過負荷に割り付けられたPEの処理時間に依存してしまう。これに対し、均一に負荷が分散された場合、それぞれのPEのシミュレーション時刻の進行は均一になる。各PEでの時刻差が少なければ、楽観的手法ではオーバーヘッドの原因となるロールバックの回数が減少する。また、保守的手法においても、例えば問い合わせ型ヌルメッセージ法では問い合わせの数が減少するなど、やはりオーバーヘッドを低減できる。

このことからも、負荷分散を考えたマッピングは非常に大切である。そこで、トランザクション中心のモデルの特徴を利用することにより、効果的な負荷分散が可能なマッピング手法を考える。

3.1 トランザクション中心モデルの特徴

トランザクション中心のシミュレーションモデルを構成する要素には、恒久要素と一時要素がある。シミュレーションは、恒久要素の中を一時要素、すなわちトランザクションが動き回ることにより進められる。対象システムをモデル化する時、システムの恒久要素をノード、恒久要素間のトランザクションの流れをアークとすると、トランザクションのフローグラフが得られる。

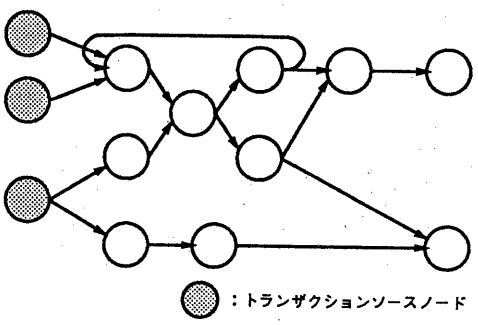

図 1: トランザクションフローグラフの例

トランザクションフローグラフの例を図1に示す。

広く研究されている論理回路モデルと、一般的なトランザクション中心のモデルを比較した場合、次のような違いがある。

1. 一つの事象による状態の変化が大きい。また、統計処理なども行なうため、事象一つあたりの処理コストが大きい。

2. 一時要素（論理回路モデルでは信号の変化）の消滅・生成が頻繁に行なわれない。消滅・生成がある場合、その位置はモデルの恒久要素として最初から規定されている。

3. 遅延時間、分岐などに乱数が用いられる。多くの場合、これら遅延時間や分岐確率、トランザクションの発生率などを変化させてシステムの状態変化を分析することがある。

2から、トランザクションフローグラフを用いると、各恒久要素のトランザクション流入量の予測が可能なことがわかる。ただし3より、一つ一つのトランザクションの動きは予測できないので、一定時間内の平均値や近似値で考える必要がある。また、遅延時間の指定形式によっては、アクティビティ内でのトランザクションのFIFOは保証されないことがある。

3.2 マッピング手法

トランザクション中心のモデルの場合、モデル要素の処理コストはトランザクションの流入量によって変化する。そこで、負荷を各PEに均等に分散させるには、トランザクションの流量を予測し、各要素の処理コストを決定する必要がある。

また、実際にマッピングをする場合、各PEのメッセージ送信・受信に対するオーバーヘッドも考慮に入れなければならない。しかし、2つのPEに注目した場合、お互いのPEに対する通信量は等しい。この事から、

(1message 送信コスト)=(1message 受信コスト)

と仮定すると、サブモデルを2つに分割し2PEに割り当てる場合、互いの通信に関するオーバーヘッ

ドは等しいので、通信のコストを考慮する必要はない。そこで、以下の要領でマッピングを行う。

1. モデルに合わせたトランザクションフローグラフを作成する。

2. トランザクションソースノードの単位時間あたりのトランザクション発生量を計算する。ソースノードで乱数が使用されている場合は平均値を用いる。

3. トランザクションフローグラフに従い、各ノードのトランザクション流入量を計算する。

4. 各ノードの総処理コストを1トランザクションあたりのコストとトランザクション流入量の積で決定する。リスト処理など、事前に予測が不可能な動的コストは考慮に入れない。

5. シミュレーションモデル全体の処理コストを計算する。

6. 処理コストがモデル全体のコストの半分になるように2つのサブモデルに分割する。

7. メッセージ受信を必要とする場所に受信コストに合わせたダミーソースノードを付加する。同様にメッセージ送信場所にもダミーノードを付加する。

8. 分割後のサブモデルに対して5から7を繰り返す。

この要領でマッピングした場合、各PEの負荷は必ずしも均一ではない。これは、モデルの切口によってトランザクションの流量が異なるためである。しかし、割り当てられたサブモデルのコストに比べて通信のコストが小さいことが前提にあるので、特に問題はないと思われる。

また、4で動的コストを考慮しないことに対する影響であるが、これは事象カレンダーのリスト長が非常に長くなった場合に問題となる。しかし、静的コストが各PEに均一に分散された場合、特定のPEのみリスト長が長くなることはないのでやはり問題にはならない。

以上より、トランザクションの流量が過渡的でない定常モデルでは、ほぼ均等に負荷を分散したマッピングが可能になる。

このマッピング手法の例を図2に示す。ノード中の数字は、各ノードの総処理コストを示している。網目のノードは通信コストを考慮するためのダミーノードである。

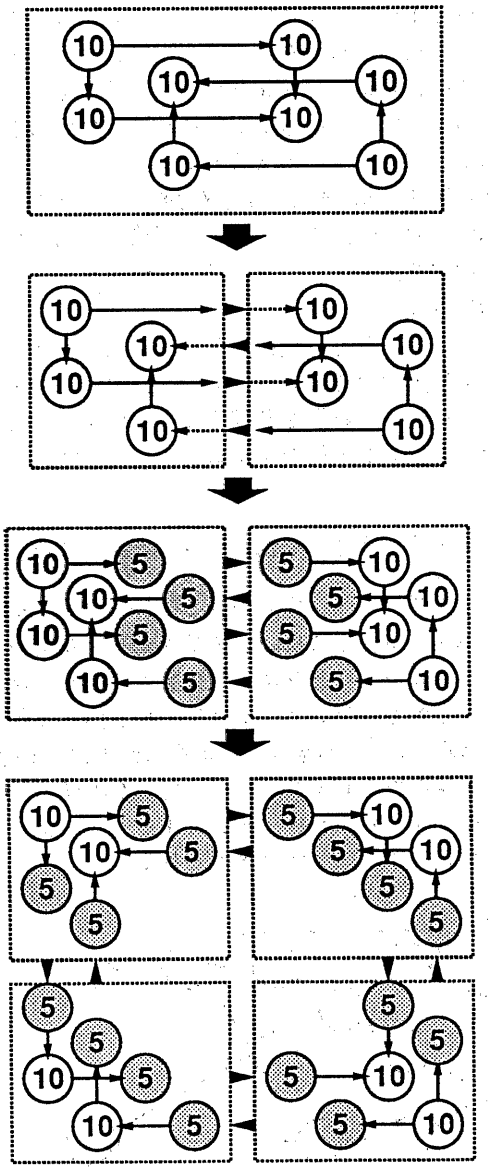

図 2: マッピングの例

4 実装と評価

4.1 AP1000 アーキテクチャについて

AP1000 は富士通研究所で開発され、メッセージ通信を基本とした分散メモリ型の並列計算機である。PE(セル) は、16 台から 1024 台まで接続可能である。AP1000 はセル間の 1 対 1 通信を行なうトーラス状の T-Net、ブロードキャスト用の B-Net、バリア同期やステータスを得るための S-Net の 3 種類のネットワークを持っている。また、メッセージの送信、あるいはメッセージの受信におけるメッセージハンドリングのオーバーヘッドを削減するために、各セルにはメッセージコントローラが搭載されている。

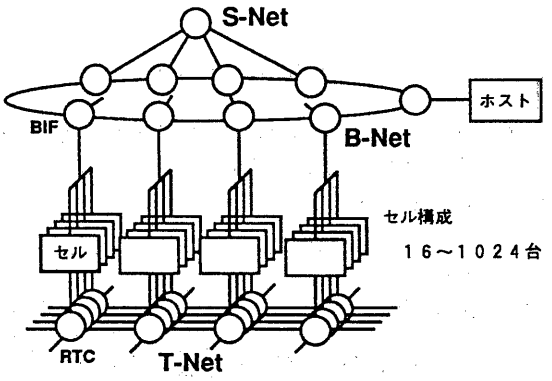

図 3: AP1000 のアーキテクチャ

離散系並列シミュレーションを分散時刻管理を用いて行なう場合、各プロセッシングエレメントが独立に非同期通信を行える事が必要とされる。また、事象カレンダーや状態・統計変数はサブモデルごとに分割し、分散メモリに配置できるため、共有メモリは特に必要としない。このため、AP1000 のようなメッセージパシングを主体としたアーキテクチャが最適であると思われる。

今回、AP1000 で利用したネットワークは、モデル情報の分散・収集のために B-net を利用した他は、すべて T-net を用いて行なった。これは、一般的なメッセージパッシング型のアーキテクチャで実現するためには、T-net のみを用いてシミュレーションする必要があることによる。しかし、今回の実装では主にシミュレーション実行中の処理に主眼を置いているため、データの分散収集にはそれに適した B-net を用いた。

また、3.2で提案したマッピング手法はサブモデルを 2 分割していくので、トーラス結合の各セルに対して特別な変更をしなくてもマッピング可能である。

4.2 各セルの処理

離散系シミュレーションは、事象カレンダーと呼ばれるシステム内に発生する事象を生起時刻順に登録するリストと、システムを構成する要素の状態を保持する状態変数、統計収集のための統計変数を用いて行われる。逐次処理の場合、シミュレーションの進行は次のように進められる。

1. 事象カレンダーから先頭（生起時刻の一番小さな）事象を取りだし、システム内の当該要素の状態、統計変数を更新する。
2. 事象を処理したことによって新たに事象が発生した場合、それらを事象カレンダーに登録する。

   1に戻る。

2で登録する事象は、もとの事象の生起時刻より小さいことはないので、全ての事象を生起時刻が非単調減少に処理することができる。これによって、シミュレーション時刻が減少することなく進められる。

これを並列処理用に拡張すると次のようになる。

1. 他のセルからメッセージ入力がある場合、メッセージを事象カレンダーに事象として登録し、処理保証時刻を更新する。
2. 事象カレンダーの先頭の事象の生起時刻と、処理保証時刻を比較し、処理保証時刻の方が小さい場合、1に戻る。
3. 事象の生起時刻の方が小さい場合、それが自セルで処理されるイベントかどうか調べ、他のセルが処理すべきイベントであった場合、メッセージとして他のセルに送信する。

   2に戻る。

4. 自セルで処理すべき事象であった場合、システム内の当該要素の状態、統計変数を更新する。

5. 事象を処理したことによって新たに事象が発生した場合、それらを事象カレンダーに登録する。

1に戻る。

一般的な方法では、5で新たに発生した事象が自セル内で処理すべき事象か調べ、他のセルで処理される事象であった場合、その時点でメッセージとして出力する。しかし、3.1のトランザクション中心モデルの特徴でも述べたように、アクティビティ内でのトランザクションのFIFOは保証されないので、5の時点でメッセージを出力してしまうと、出力メッセージの持つタイムスタンプの非単調減少が保証できなくなる。そこで、他のセルが処理する事象も一旦自セルの事象カレンダーに登録し、自セルのシミュレーション時刻が出力事象の時刻になった時点で、メッセージとして出力するようにした。

このように分散時刻管理を用いる場合でも、1セルにマッピングされたモデルは逐次処理と同様のスケジューリングで行なうことにする。これにより余分な処理を省き、オーバーヘッドを低く抑えることができると考えられる。

4.3 実行例

離散系シミュレーションはさまざまな分野で利用されているが、本稿では対象モデルとして仮想駅構内を想定し、人の流れをシミュレーションした。仮想駅のモデルを図4に示す。

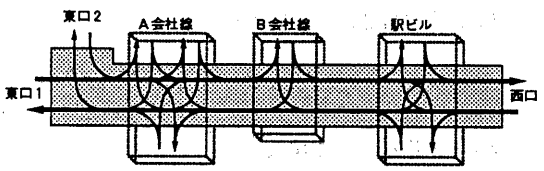

図 4: 仮想駅モデル

仮想駅には2つの会社線と駅ビルが存在し、東西に出入口がある。モデルのそれぞれの入口(トランザクションソースノード)から、人(トランザクション)が6.0 ± 1.0秒間隔で発生し、構内を動き回る。分岐する場所においてトランザクションの流れは確率的に決定されるものとした。

また、このモデルをトランザクションフローグラフに変換すると図5のようになる。

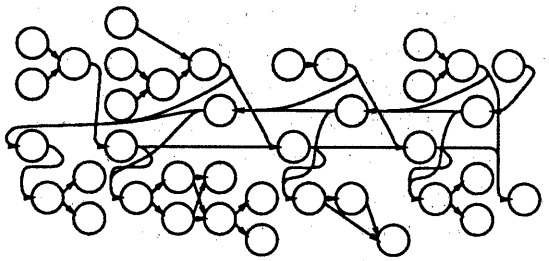

図 5: 仮想駅のトランザクションフローグラフ

4.4 デッドロック回避について

図5の様なモデルの場合、単純にモデルを分割していくとセル間にループが発生し、デッドロックが発生する可能性がある。

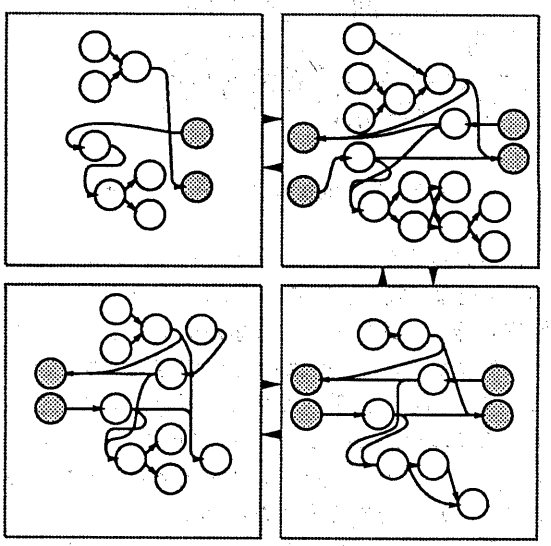

図 6: デッドロックの可能性のある場合

しかし、東口→西口、西口→東口の流れを別のセルにマッピングすることにより、セル間のループを除去することができる。

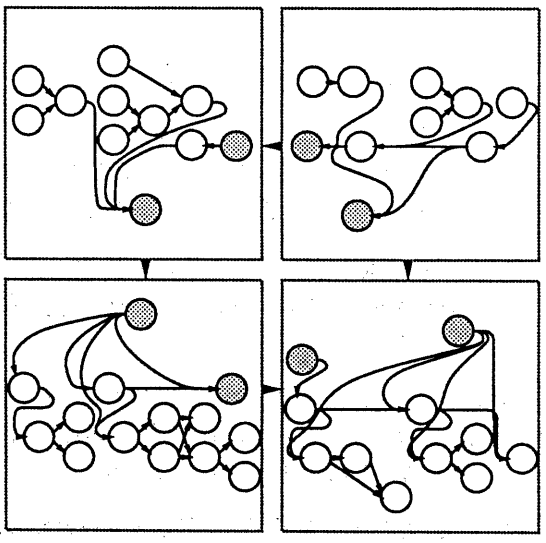

図 7: デッドロックの可能性のない場合

従来から研究されてきている保守的手法、楽観的手法はデッドロックを回避する目的以外にも、メッセージ待ちによる処理効率の低下を抑える目的がある。しかし、論理回路シミュレーションなどと異なり、今回のような粒度の荒い問題に対しては、メッセージ待ちによる処理効率の低下は非常に小さいと考えられる。

そこで、モデルのセルへのマッピングは全てデッドロックの発生しないような形にし、基本的には保守的手法を用いてデッドロック回避に対する特別な考慮はしなかった。

4.5 実行結果と考察

3.2のマッピング手法にしたがい、実際のモデル分割は手作業で行なった。提案したマッピング手法の特徴であるトランザクション流量を考慮した負荷分散と、単純に各ノードの1トランザクションあたりの処理コストで負荷分散した場合について、それぞれシミュレーションを実行した。また、今回のモデルは処理ノード数が38個と比較的小数でであるため、あまり多くのセルを使うと負荷分散が考慮できない。そこで、8台までのセルを用いた場合の実行時間とセル利用率を測定した。

シミュレーション実行時間の結果を図8に示す。

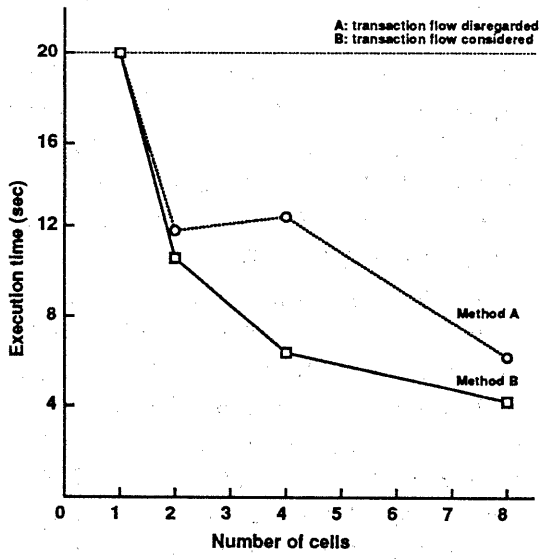

図 8: シミュレーション実行時間

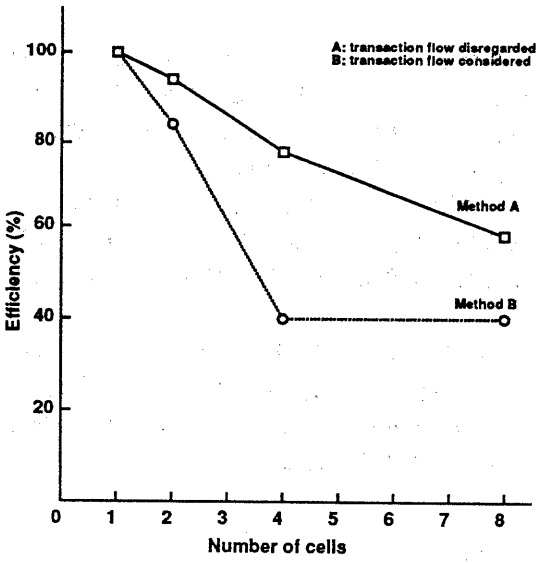

図 9: セル利用率

2,4,8台のセルを使用した場合、トランザクションの流量を考慮する (Method A) としない場合 (Method B) に比べて11 %～48 %の処理時間短縮が得られた。

トランザクションの流量を考慮しない場合、セル数を2台から4台に増やした時に実行時間が増加してしまっている。これは、トランザクションの流量に偏りがあったため、2台の場合と比較して通信コストのオーバーヘッド分だけ処理時間が増加したためと考えられる。

また、セルの利用率は図9のようになった。トランザクションの流量を考慮した場合、しないものに比べてセル利用率の低下が少ないことが分かる。対象システムを分析する手段として離散系シミュレーションを用いる場合、システム内に流入するトランザクションの量を変化させ、システムの状態変化を評価することがしばしば行なわれる。このような場合でも、本マッピング手法を用いればトランザクション流入量に対応したマッピングを行える。

5 おわりに

離散系シミュレーションで一般的なトランザクション中心のモデル構成概念における静的マッピング手法を提案した。また、これを仮想駅モデルで実行し、手法の効果が確認できた。今回は1モデルでの評価にとどまったが、この手法は分岐が確率的であり、トランザクションの発生量が過渡的でない定常モデルに対しては有効であると思われる。

現在は、提案しているマッピング手法に従い、なるべく通信コストが小さくなるように手作業でモデルの分割を行なっているが、今後はこれを自動化していく予定である。

謝辞

並列計算機 AP1000 を使わせていただいた富士通研究所の方々、ならびに早稲田大学村岡研究室の皆様方に深く感謝致します。

参考文献

[1] 中西俊男:コンピュータシミュレーション, 近代科学社 (1977)

[2] 森戸晋, 相澤りえ子:SLAMII によるシステムシミュレーション入門, 構造計画研究所 (1986)

[3] 森戸晋:離散系シミュレーションの現状と今後の研究動向, 計測と制御, Vol.30 No.2, pp.101-109(1991)

[4] 松本 幸則, 瀧 和男:バーチャルタイムによる並列論理シミュレーション, 情報処理学会論文誌, Vol.33 No.3, pp.387-396(1992)

[5] 工藤知宏, 木村哲朗, 天野英晴他:問い合わせに基づく並列論理シミュレーションアルゴリズム, 電子情報通信学会論文誌 D-1, Vol75 No.4, pp.221-231(1992)

[6] 工藤知宏, 木村哲朗他:共有メモリを想定した並列論理シミュレータ, 電子情報通信学会研究会報告, CPSY91-23, pp.151-158(1991)

[7] Chandy K. M., Misra J.:"Distributed Simulation : A case study in design and verification of distributed programs", IEEE Trans. Software Eng., Vol.5 No.5, pp.440-452(1979)

[8] Misra J.:"Distributed discrete-event simulation", Comput. Surv., Vol.18 No.1, pp.39-65(1986)

[9] Jefferson,D.R.:"Virtual Time", ACM Trans. Program. Lang. Syst., Vol.7, No.3, pp.404-425(1985)

[10] Fujimoto R M:"Parallel Discrete Event Simulation", Winter Simulation Conf. Proc., Vol.1989, pp.10-28(1989)