ハイパフォーマンス
コンピューティング 82−11
（２０００．８．３）

連続事象シミュレーションにおける並列化アルゴリズムの実験評価-（２）

滑川 光裕†, 後藤 和則††, 上原 稔†††, 森 秀樹†††, 佐藤 章†††
†嘉悦女子短期大学, ††東京職業訓練短期大学校, †††東洋大学

連続事象シミュレーションシステムにおける効率化を図るための手法として、トランザクションフロー型システムにおける並列化アルゴリズムの提案を行った。この方法では、原則的に連続的な同期が必要となる連続事象並列シミュレーションにおいて、トランザクションの転移（並列化のために分割された空間の間の移動）の予測を用いることにより、同期処理を離散化することによって効率の良い並列シミュレーションを可能にしている。しかしながら、トランザクションの密度が高くなると、同期処理回数は増え、パフォーマンスが低下する。本報告では、トランザクションの密度や並列数によるアルゴリズムの適用範囲について述べる。

A verification of synchronization algorithm for continuous event- (2)

Mitsuhiro NAMEKAWA†, Kazunori GOTO††, Minoru UEHARA†††,
Hideki MORI††† and Akira SATOH†††
†Kaetsu Women's college, †† Tokyo Institute Politechnique Univ., ††† Toyo University

We proposed parallelization algorithm in the transaction flow type system as a technique for already attempting the efficiency improvement in continuous event simulation system. By using the prediction of the transition (transfer between the space divided for the parallelization) of the transaction in continuous event parallel simulation in which the fundamentally continuous synchronization is required, the parallel simulation of which it is efficient by making synchronous processing discrete, is enabled. However, it is not practical, since in this method, the synchronous processing frequency increases, when the density of the transaction rises, and since the performance lowers. In this paper, we describe the effectiveness corresponding to density, and parallelization and application range of the algorithm in the transaction type system.

１．はじめに

近年の情報ネットワーク化により、ワークステーションやパーソナルコンピュータが普及し、それによって、高性能なプロセッサが廉価に入手できるようになった。また、高性能なネットワークについても同様のことから低廉化の傾向にある。これらの高性能なプロセッサとネットワークを利用した計算機クラスタが注目されている。

特に、シミュレーションの分野では、モデルを数値化した膨大なデータを処理することから、大規模かつ複雑なシステムを対象とするシミュレーションは、ベクトル計算機などのスーパーコンピュータでなければ実用的でないとされていた。しかしながら、この計算機クラスタを用いることによって、廉価でありながら高速

なシミュレーション処理が可能となっている。

また、計算機クラスタを利用するメリットとして、従来のＯＳ環境でのシステム開発が可能であるという利点もある。このために、様々な分野での研究開発が行われている。

しかしながら、並列・分散システムの開発は、逐次プログラムと比べると複雑であり、処理の高速化と引き換えに開発コストが高いという傾向がある。

我々は、シミュレーションのもつ並列性を抽出し、複数プロセッサに負荷を分散させることにより、シミュレーション処理の効率を上げる並列化によって、ハイパフォーマンスなシミュレーションシステムの構築を目指している。しかし、これまで行われてきた並列シミュレーションのほとんどが離散事象によるものであり、連続事象を対象としたものは、ほとんど見あたらなかった。それは、連続事象の並列シミュレーションが原則的にシミュレーションクロック毎の同期処理を必要とするため、並列化による処理効率が極めて悪いことにその原因があった。

我々は、シミュレーションモデルを３次元空間の中を流体が流れるような状況を再現するモデルで表現される空間フロー型システムと、ある種のトランザクションがその空間を移動するトランザクションフロー型システムに分類し、社会一般の現象のほとんどを当てはめることの出来るトランザクションフロー型システムにおいて、そのトランザクションの移動予測（これを「転移」と呼ぶ）から同期処理を省略し、同期処理の離散化を行うことによって、効率の良いシミュレーションが可能となる同期方法について提案した。

しかし、実際には、密度が高くなればなるほど、同期処理の離散化が困難となり、本手法による並列処理のパフォーマンスが低下することが考えられる。本論文では、トランザクションの密度や並列数によるアルゴリズムの適用範囲シミュレーションの有効性について述べる。

２．並列シミュレーションについて

２．１　連続事象シミュレーションの並列化手法について

並列シミュレーションでは、シミュレーションのもつ潜在的な並列性を抽出し、効率化を図ることが重要となる。そのための方法として、共通する特定の機能を分離独立させ、それを並列化して処理する方式と、シミュレーション対象となるシステムを空間的に分割し、分割された部分空間ごとに処理を分散させる方式とが、一般に考えられる。前者を機能分散方式、後者を空間分散方式と定義する。

特に、並列シミュレーションでは、負荷が分散されたプロセッサ間において、論理的な矛盾がないように、処理をすすめる必要がある。そのために、プロセッサ間での通信・同期処理というオーバーヘッド時間が必要となるが、この時間をいかに減らすことができるかが処理効率を向上するために重要となる。機能分散でのオーバーヘッド時間は、共通する機能へのアクセス回数とその処理に依存し、空間分散のそれは隣接する部分空間とトランザクションの転移によって決まる。同一機能・性能をもった計算機による並列処理では、どのような分散形態を採ったとしても、シミュレーションのみに関わる処理時間は不変である。したがって、空間分散方式による並列シミュレーションでは、シミュレーション処理時間に無矛盾処理や同期処理のために必要とする準備処理および通信時間、待ち時間などのオーバーヘッド時間の総和を加えたものとなる。

これらのことから、計算機クラスタを用いた並列シミュレーションでは、空間分散方式の方が有利であると考えられる。

２．２　並列シミュレーションのモデルについて

並列シミュレーションの処理効率を向上させる手法を考えるために、そのモデルの分類を行った。

シミュレーションでのモデルにおいては、そのモデルの特徴や構成される要素どうしがお互いにどのような影響を与えられるかという観点から、次の２つに分類することができる。

（ａ）流体フロー型モデル

空間の中を流体のように連続的な物質が動くシステム。気象や海流などの物理現象におけるモデル記述などに用いられる。

（ｂ）トランザクションフロー型モデル

空間の中をある形を持ったシミュレーション要素が動くシステム。このシステムでは、シミュレーション要素どうしの明確な記述が可能である。

これらの分類のうち、（ｂ）のトランザクションフロー型モデルは、シミュレーション要素同士の明確な記述ができることから、社会一般のシステムを記述することが可能である。我々は、このモデル記述方法に基づいた並列化を行なった。

３．同期アルゴリズム

３．１　並列シミュレーションの同期方式

本研究では、連続事象シミュレーションの並列化を行うことを目的としているので、シミュレーションクロックごとに同期処理を行う「毎回同期方式」が基本となる。しかし、効率の良い並列シミュレーションを行うためには、全てのシミュレーションクロックごとに同期処理を行うのではなく、的確なタイミングでプロセッサ間での情報交換が行えるように同期処理を行うことが望ましい。

そこで我々は、トランザクションの動きに注目し、トランザクションがプロセッサ間を移動（これを転移と呼ぶ）するタイミングをとらえて同期処理を行えるような方法を考案した。しかしながら、シミュレーションでは、トランザクションの動きを的確に予測することは不可能であるので、トランザクションの転移を予測する方法を考えた（図１）。実際には、トランザクションの転移予測は、様々なものが考えられるので、その最早の転移予測時刻に同期処理を行い、転移が行われなかった場合には、転移予測を繰り返す方法を用いることにした。これによって、矛盾のない同期予測を行うことができる。

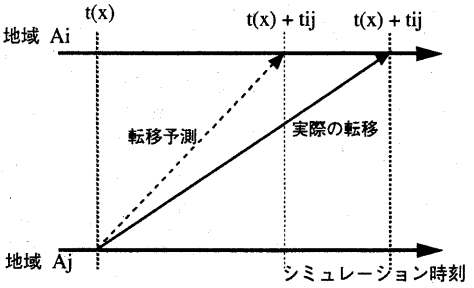


図１　転移予測と実際の転移

３．２　複数プロセッサでの同期処理について

２つのプロセッサ間における同期処理については、前節で述べたとおりである。この２プロセッサ間での同期処理をそれぞれのプロセッサが隣接する空間どうしで行うことで、全体のシミュレーションを進めることができる。

しかし、実際には複数のプロセッサから要求があった場合、同期処理方法によっては、デッドロックを起こす可能性も考えられる。

一般的には１台の管理用プロセッサが全ての並列プロセッサ状態を把握して、同期処理を行えば、デッドロックは起きない。しかし、この方法では、通信時間その他に、オーバーヘッドが多くなるので、自律的に分散して行う方式をとり入れた。

この同期処理は、「マルチエージェント通信」による方式である。

対象とする平面を分割してプロセッサに割り当てる場合、図２に示す正規化された配置の変形で全てを表すことができる。したがって、その複雑さに対応した、正規形について

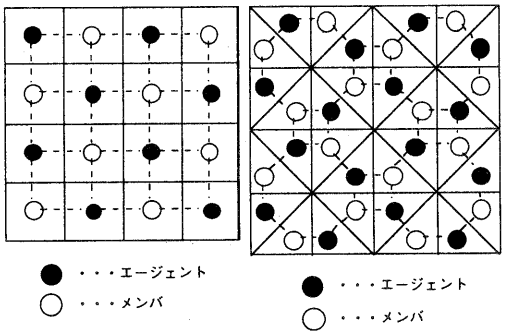
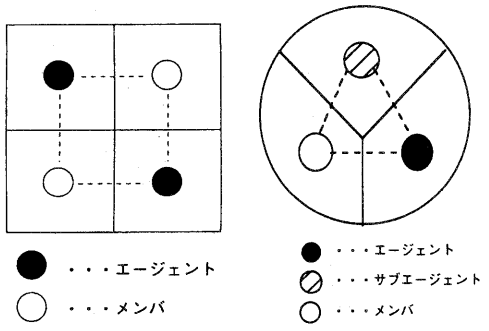


図2　マルチエージェント通信方式

のエージェント・メンバ配置を考えれば、初期の通信方式が実現できる。図3の（a），（b）は、その基本形（要素）である。

このマルチエージェント通信方式では、図2のように各プロセッサと通信をする必要のあるプロセッサ間をそれぞれノードとアークで表し全てのノードに対して、エージェント（●印）とメンバ（○）に分類する。そして、エージェントが主体的にメンバとの通信を行い、メンバはエージェントからの通信を待つことになる。そのために、全てのエージェントにアークとしてつながっているのはメンバでなければならない（図a）が、空間の分割方法によっては、そうならない場合もある。その場合には、サブエージェント



(a) 一般的な構成　　(b) 3つが隣り合わせの例

図3　マルチエージェントの基本形

（◎）を指定することによって、どの通信の組み合わせにおいても、エージェントとメンバの通信関係を成立させることができ、効率の同期通信処理が可能となる。

マルチエージェント通信は、並列シミュレーションを行うプロセッサを統合管理することなく、各プロセッサがデッドロック状態に陥らず、自律的で効率の良い通信同期処理を行う方法である。

各プロセッサのエージェントとメンバが隣接するように配置し、エージェントが通信を主導的に行うことによって、効率の良い通信が可能となる。

4．実験システム

4．1　仮想並列シミュレーション環境

前章で述べた同期および通信のアルゴリズムの検証を行うために、我々は、1台のワークステーション上に仮想的な並列シミュレーション実験環境を作り出し、実験を行った。この実験のために、複数プロセッサでの並列シミュレーションにおける実際の処理を把握するために、あらかじめ2台のワークステーションでの実験を行い、そこでのシミュレーション処理、通信処理、転移予測計算処理などの処理にかかる時間を計測し、その値をパラメータとした。仮想並列シミュレーション環境では、複数プロセッサでの並列シミュレーションを行いながら、各処理が行われるタイミングを管理し、それにあわせて計測したパラメータ値を積算していくことで、並列シミュレーション全体としての処理時間を算出することができる。

4．2　実験モデル

本研究における対象として、微視的モデル道路交通シミュレーションを行った。微視的モデル道路交通シミュレーションは、多数の自動車要素（トランザクション）がシミュレーション空間内を自由に移動するため、社

会システムのモデルとしては、複雑な連続系のシミュレーションであるといえる。

5．実験結果

図4に毎回同期方式と新アルゴリズムにおけるプロセッサ数を増加させたときの実行時間の比較である。プロセッサ数が増加すると

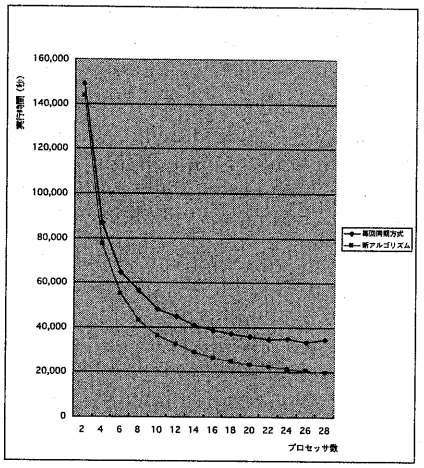


図4　並列シミュレーションの実行時間比較

ともに、オーバーヘッド時間の違いから、その差が大きくなっていくことが分かる。

このオーバーヘッドのうち、大きな部分を占める通信処理時間を、同様にプロセッサ数

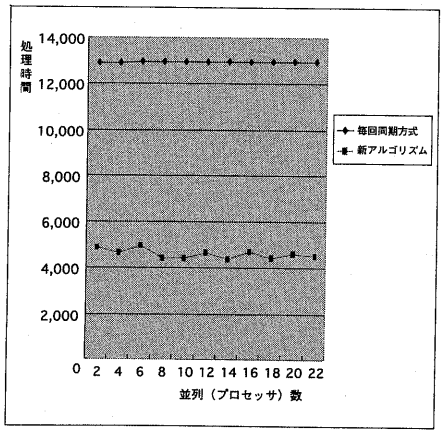


図5　通信処理時間の比較

の増加とともに、どのように変化するかを示したのが、図5である。新アルゴリズムの方が、常に3分の1程度通信処理時間が短く、また、それぞれが一定の処理時間であることから、プロセッサが増加することにより、シミュレーション処理が減少することから、プロセッサ数が増えれば増えるほど、大きな負荷となることがわかる。

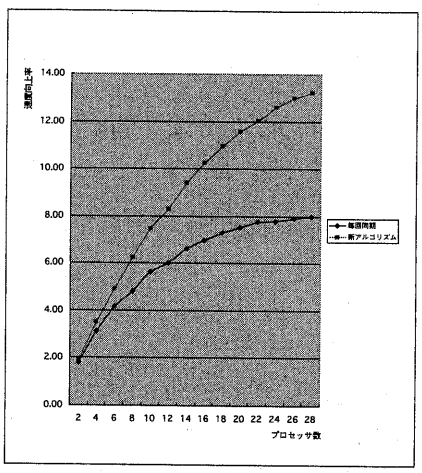


図6　速度向上率の比較（1,500台）

図6に、速度向上率のグラフを示す。このグラフから、プロセッサ数の増加とともに、速度向上率が鈍化していく傾向を知ることができる。特に、毎回同期方式の速度向上率の鈍化傾向は早く表れている。

図7は、交通密度と実行時間の比較を示したものである。交通密度の増加とともに、毎回同期方式と新アルゴリズムの差が少なくなっていくことがこのグラフからわかる。交通密度が高くなるとともに、転移予測が行うことのできる時間間隔が短くなり、事実上、毎回同期に近くなるからである。

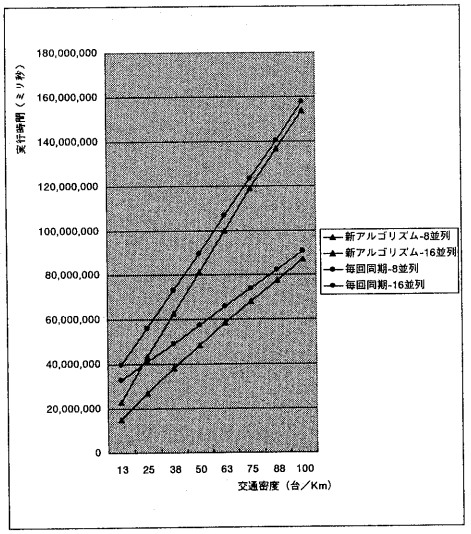


図７　交通密度と実行時間の比較

６．まとめ

本研究では、トランザクションフロー型の連続事象シミュレーションにおいて、効果的な並列化を行うために、各プロセッサが自律的な処理を行うことができ、デッドロックの防止と効率的な通信を行うマルチエージェント方式を導入した。これによって、大幅に処理時間の低減を図ることができるが、プロセッサ数の増加とともに、速度向上率が鈍化する傾向を見ることができた。また、交通密度の増加によっても、処理効率が低下し、毎回同期アルゴリズムからの優位性も失われることも確認できた。

その結果、トランザクションの密度やプロセッサ数によるアルゴリズムの適用範囲に対する見通しを得ることができた。

今後は、他システムへの適用を考えるなど、アルゴリズムの汎用化、精緻化に努めたい。

参考文献

[1] John M. Creagh ： MTTS:A Modelling Methodology for raffic Simulation, Proceedings of the 30th Summer Computer Simulation Conference, pp.393-403, 1998

[2] Hassan, R., : Parallel Simulation Using Conservative Time Windows. Winter Simulation Conference, 709-717, 1992

[3] 猪飼國夫、板倉直明他：メタ論理を構成することを考慮したハードウェアファジィ推論ユニットの製作、第５回インテリジェントシステムシンポジウム論文集、1995

[4] Misra , 1986, Distributed Discrete-Event Simulation., ACM Comp., surveys. Vol18, pp.39-65.

[5] M.Namekawa, A.Satoh, et al. : Clock Synchronization Algorithm for Parallel Road Traffic Simulation in a wide area, Mathematics and Computers in Simulation, Vol.48, No.4-6, pp.351-359, 1999