空間的影響範囲を考慮した分散論理時刻管理方式 Breathing Space-Time Buckets アルゴリズム

阿部一裕[†] 古市昌一[††] 尾崎敦夫[††]
田中秀俊[††] 中島克人[††]

並列離散事象シミュレーションにおける，新しい分散論理時刻管理手法である *Breathing Space-Time Buckets* アルゴリズム（*BSTB*）を提案する．移動体を対象としたシミュレーションでは，距離的に離れた移動体どうしが相互作用を及ぼさないなど，シミュレーション要素間の相互作用に空間的な局所性を持つ場合がある．*BSTB* では Breathing Time Buckets アルゴリズム（BTB）にイベントの空間的影響範囲を考慮する拡張を行った．また *BSTB* を含め，いくつかの分散論理時刻管理機構を備えた並列オブジェクト指向シミュレーション環境 *OSim* について述べる．最後に *BSTB* アルゴリズムの実行性能の評価について論じる．評価結果より *BSTB* は，各種条件の下，特に演算ノード数が多い場合に BTB と比較し高い実行性能を得られることが分かった．

Breathing Space-Time Buckets Algorithm —A Distributed Logical Time Management Method Utilizing Spatial Influence Ranges of Events

KAZUHIRO ABE,[†] MASAKAZU FURUICHI,[††] ATSUO OZAKI,[††]
HIDETOSHI TANAKA[††] and KATSUTO NAKAJIMA[††]

We propose *Breathing Space-Time Buckets algorithm* (*BSTB*) which is a new distributed logical time management method for parallel discrete event simulation. *BSTB* extends the Breathing Time Buckets algorithm (BTB) by exploiting the spatial locality of events, which is typical in moving objects simulations for example distant mobile objects don't influence each other. Then we describe parallel object oriented simulation environment *OSim*, which supports multiple time management schemes including *BSTB*. Finally, we discuss the performance evaluation of *BSTB*. Evaluation results show that BSTB achieves better execution performance than BTB especially under a larger number of computation nodes.

1. はじめに

離散事象シミュレーションでは，シミュレーション時刻の管理単位である論理プロセス（LP）間で，タイムスタンプが付加されたイベントが受渡しされる．LP はイベントをタイムスタンプの順に処理していくが，並列計算機上で離散事象シミュレーションを実行する場合，複数の演算ノード上に分散配置された LP のシミュレーション時刻の進みかたは各 LP により異なるため，高い並列度を保ちつつタイムスタンプ順にイベントを実行していくことは困難な問題である．そのため各 LP を極力非同期に動かしつつ，因果関係を満たす（実行結果がすべての LP で同期をとりシミュレーション時刻を進めていった場合と同じ結果が得られる）ようにイベントを実行させるための分散論理時刻管理手法は，並列離散事象シミュレーション分野における中心的研究課題となっている[1)]．

分散論理時刻管理手法は，保守的なものと，楽観的なものに大別される．

保守的な手法では，LP はイベントを実行する際に，そのイベントのタイムスタンプよりも古い時刻のタイムスタンプを持つイベントを受け取ることがないと判断できてはじめて，そのイベントを実行する．

楽観的な手法ではイベントを投機的に並列実行する．すでに実行したイベントのタイムスタンプよりも過去の時刻のタイムスタンプ値を持ったイベントを LP が

† 三菱電機株式会社先端技術総合研究所
Advanced Technology R&D Center, Mitsubshi Electoric Corporation
†† 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubshi Electoric Corporation

受け取ったなど，因果律を破る可能性がある順序でイベントが実行された場合，因果律を破る順序でイベントを実行したLPと，誤った順序で実行されたイベントの影響を受ける可能性があるLPは，状態を過去の値に戻し，その間に生成したイベントを無効化するロールバック操作を行う．

我々は大規模な移動体シミュレーションプログラムを対象とした効率的な並列実行環境の提供およびプログラミング技法の確立に関する研究を行ってきた[2)~4)]．移動体シミュレーションでは，距離的に離れた移動体どうしが相互作用を及ぼさないなど，イベントは，しばしば局所化される．しかしながら従来提案されてきた分散論理時刻管理手法では，イベントの実行順序を検査する際に，イベントの持つ局所性を利用しておらず，保守的すぎる判定により本来因果関係のないイベントにより不必要なロールバックが生じ，並列度の低下をまねく場合があるなどの問題点があった．

この問題点を解決するために，我々は *Breathing Space-Time Buckets* アルゴリズム（*BSTB*）と呼ぶ新たな分散論理時刻管理手法を開発した．*BSTB* は楽観的な分散論理時刻管理手法の1つであるBreathing Time Buckets アルゴリズム（BTB）[7)] にイベントの空間的局所性を利用する拡張を行った手法であり，イベントの実行順序が矛盾を生じることなく行われているかどうかを判定する際に，各イベントの空間的影響範囲を考慮し，空間的に局所化されたイベント限界を使用する．空間的に局所化されたイベント限界を使用することにより，*BSTB* はロールバック総数を減少させることができ，高い並列度を達成することが可能となる．

我々が開発した並列オブジェクト指向シミュレーション環境 *OSim*[2)] に *BSTB* を組み込み，*OSim* を用いて記述したシミュレーションプログラムをIntel Paragon XP/S 上で問題パラメータを変化させて実行した．実行結果より *BSTB* は BTB と比較し，特に演算ノード数が多い場合に高い実行性能を示すことが分かった．

本稿では，まず従来提案されてきた各種時刻管理アルゴリズムを簡単に説明するとともに，その問題点についてふれた後 *BSTB* のアルゴリズムを説明する．次に *BSTB* を含む複数の分散論理時刻管理手法を備えた並列シミュレーション環境 *OSim* の特徴と実装について述べる．最後に *BSTB* の実行性能を示し，その有用性を評価する．

2. アルゴリズム

イベントを独立に実行できるシミュレーション時刻の区間のことを Time Window と呼ぶ．並列性を向上させるための手段として，多くの分散論理時刻管理アルゴリズムで Time Window 手法が利用されている．本章では *BSTB* を説明するための準備として Time Window 手法を利用した，いくつかの従来手法と，その問題点について論じた後，*BSTB* のアルゴリズムを説明する．

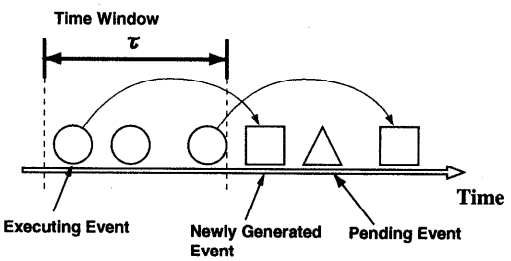


図1 Time Buckets Synchronization におけるイベントダイアグラム例
Fig. 1 Event diagram example of Time Buckets Synchronization.

Time Buckets Synchronization

Time Buckets Synchronization 法[5)] は，イベントと，そのイベントが新たに生成するイベントのタイムスタンプの最小時間間隔である最小遅延時間 τ が既知であり，この時間間隔 τ の間にあるイベントは独立に実行できることを利用した保守的な分散論理時刻手法である．

図1は，Time Buckets Synchronization 法におけるイベントの生成関係を示したダイアグラムである．図において横軸はシミュレーション時刻を表す．

Time Buckets Synchronization 法では，各LPは実行サイクルごとに同期をとり，タイムスタンプの値が最小遅延時間 τ の時間幅を持った Time Window 内にあり実行待ち状態にあるイベントを並列実行する．すべてのLPが Time Window 内にあるイベントの実行を終了した後，Time Window を シミュレーション時刻 τ だけ移動させる．

Time Window の時間幅は最小遅延時間と等しいため，Time Window 内のイベントが新たに生成するイベントのタイムスタンプの値は Time Window の上限時刻より大きく，LPは実行したイベントのタイムスタンプ値よりも小さい時刻のタイムスタンプ値を持ったイベントを，次回以降の実行サイクルで受け取ることはない．

この手法は，同期のためのオーバヘッドが小さいため，最小遅延時間 τ が十分大きく，独立して実行できるイベントの数が多いならばイベントを並列実行する

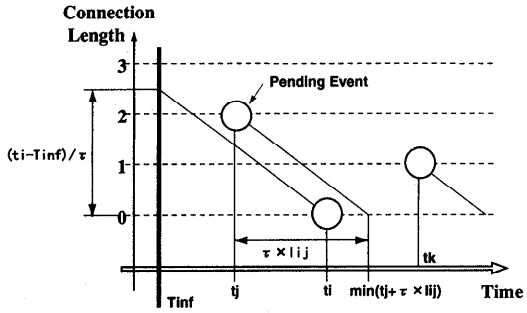


図2 Moving Time Window におけるイベントダイアグラム例
Fig. 2 Event diagram example of Moving Time Window.

ことにより高い実行効率をあげることができる．しかしながら平均的な遅延時間に比べ最小遅延時間 τ の値が過度に小さい場合，同時に実行可能と判定されるイベントの数が少なくなるため並列度が低くなる問題がある．

Moving Time Window

LP 間でのイベントの送受信先が固定されている場合，LP をノード，イベントを送信する LP を始点，受信する LP を終点とするアークから作られる有向グラフ上を，イベントの影響が直接または，このイベントが生成するイベントにより間接的に伝達される．有効グラフ上で LP_i を表すノードから LP_j を表すノードへ到達する経路において通過するアーク数が最も少ない経路の通過アーク数を LP_i から LP_j への接続距離と呼ぶ．

Moving Time Window 法[6] は，最小遅延時間 τ と LP 間の接続距離を利用し，各 LP においてイベントキュー内にある実行待ち状態のイベントが実行可能かどうかを判定する際に，各 LP で 時間幅が変動する Time Window を用いる保守的な分散論理時刻手法である．

Moving Time Window 法では，各 LP 間で同期をとり，各 LP でイベントを実行可能かどうか判定した後，実行可能と判定された LP に関してはイベントを実行するサイクルを繰り返す．

図2 は Moving Time Window 法におけるイベントダイアグラムの一例である．図において横軸はシミュレーション時刻，縦軸はある LP（LP_i）から見た各 LP（LP_j, LP_k など）の接続距離である．図2の例では，LP_i から接続距離が 2 離れた LP_j がタイムスタンプ値 t_j のイベントを持ち，LP_i から接続距離が 1 離れた LP_k がタイムスタンプ値 t_k のイベントを持つ場合を示している．

実行サイクルのはじめに全イベントのタイムスタンプの最小値 T_{inf} を求める．LP_i で このサイクルで実行しようとするイベントのタイムスタンプの値を t_i とする．

接続距離が $(t_i - T_{inf})/\tau$ 内にある LP のイベントは直接または間接的に LP_i にタイムスタンプ値が t_i より小さいイベントを送る可能性がある．LP_i は，接続距離が $(t_i - T_{inf})/\tau$ 内にある各 LP_l がこのサイクルで実行しようとするイベントのタイムスタンプの値 t_l を調べる．

LP_i が実行しようとするイベントのタイムスタンプの値 t_i が $T_{inf} < t_i < \min(t_l + l_{i,l} \times \tau)$ （$l_{i,l}$ は LP_i と LP_l の接続距離）の範囲にある，すなわち $[T_{inf}, \min(t_l + l_{i,l} \times \tau)]$ の時間区間の Time Window 内にあるならば，LP_i は，次回以降のサイクルにおいて t_i より小さいタイムスタンプ値を持つイベントを受け取ることはないので LP_i が持つタイムスタンプ値 t_i のイベントは，このサイクルにおいて実行可能と判定される．

この手法では実行しようとするイベントのタイムスタンプ値より，小さいタイムスタンプ値のイベントを受け取る可能性がある場合はイベントを実行しない保守的な手法であるため，Time Buckets Synchronization と同様，平均的な遅延時間に比べ最小遅延時間 τ の値が過度に小さい場合，同時に実行可能と判定されるイベントの数が少なくなるため並列実行するイベントの数が低下する問題点がある．

Breathing Time Buckets

Breathing Time Buckets アルゴリズム（BTB）[7] は，各 LP でイベントを投機的に実行し，実行したイベントが新たに生成するイベントのタイムスタンプ値から Time Window の時間幅を動的に決定する楽観的な分散論理時刻管理手法である．

図3 は，一次元的に配置された LP が隣接 LP にイベントを送信する問題における BTB のイベント生成関係を示したダイアグラムの一例である．図において横軸はシミュレーション時刻，縦軸はイベントを送受信する LP の配置を表す．

BTB では，各 LP 間で同期して，1 つのイベントを投機的に実行し，実行終了後イベント実行順序が因果関係を満たすものであったかどうかを判定するサイクルを繰り返す．

イベント実行時に新たに生成されたイベントは送信先の LP に送ることなく生成した LP で保存しておく．すべての LP でイベント実行を終了した後，このサイ

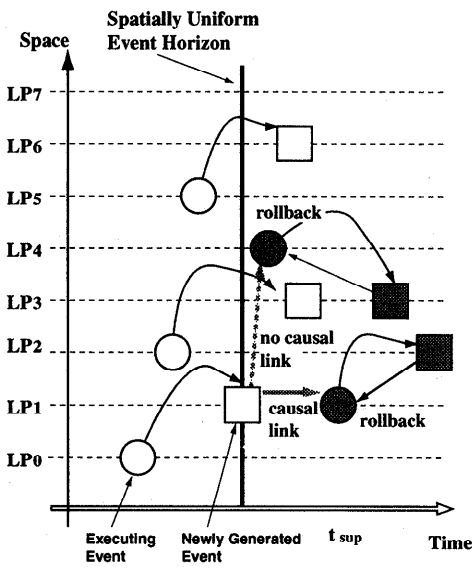


図 3 BTB におけるイベントダイアグラム例
Fig. 3 Event diagram example of Breathing Time Buckets.

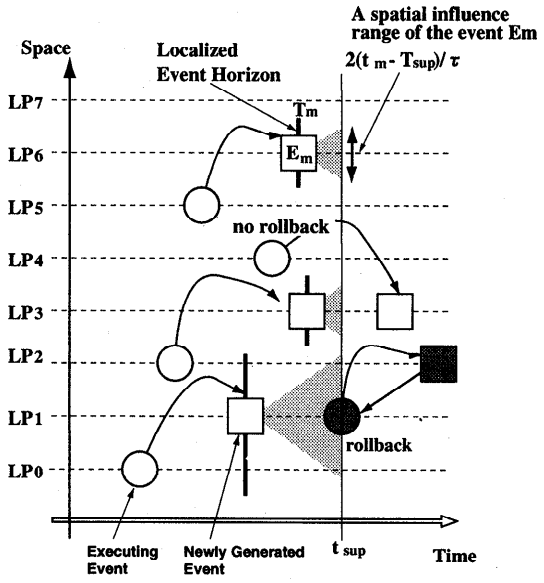


図 4 *BSTB* におけるイベントダイアグラム例
Fig. 4 Event diagram example of Breathing Space-Time Buckets.

クルで新たに生成されたイベントのタイムスタンプの最小値を求め，この値をイベント限界時刻とする．

このイベント限界時刻よりも将来のタイムスタンプ値を持つイベントを実行した LP は，新たに生成されたイベント，またはこのイベントが次回以降のサイクルで生成するイベントから，このサイクルで実行したイベントよりも過去のタイムスタンプ値を持ったイベントを受け取る可能性があるので，状態値をイベント実行前の値に戻し，新たに生成したイベントを棄却するロールバック処理を行う．イベント限界時刻よりも過去のタイムスタンプのイベントを実行した LP は，新たに生成したイベントを送信先の LP に送る．

BTB ではイベント限界時刻を新たに生成されたイベントの時間的分布に従い動的に決めることにより，Time Window の幅を適応的に設定でき，Time Buckets Synchronization 法と比較し高い並列度を上げることができる．また投機的にイベントを実行するためロールバック処理が必要となるが，ロールバックは，1 つ前に実行したイベントまでにしか遡らないので，ロールバックの準備のための実行コストは Time Warp 法[8] と比較し低い．

しかしイベント限界時刻を，そのサイクルで新たに生成したすべてのイベントのタイムスタンプの最小値として空間的に一様に定義するため，イベントの空間的到達範囲が局所化できる場合でも，たとえば，図 3 の LP_4 が実行したイベントのように，本来因果関係

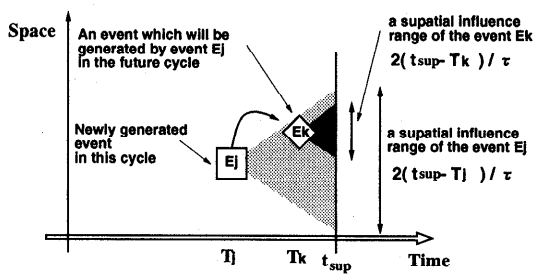


図 5 イベントの空間的影響範囲
Fig. 5 Spatial influence hranges of events.

がないイベントによりロールバックが引き起こされる可能性がある．

Breathing Space-Time Buckets

Breathing Space-Time Buckets アルゴリズム（*BSTB*）は，Breathing Time Buckets アルゴリズムに対し，Moving Time Window 法で用いられた最小遅延時間と LP 間の接続距離からイベントの空間的影響範囲を考慮する手法を適用し，実行したイベントが新たに生成するイベントのタイムスタンプ値から，各 LP ごとに異なる Time Window の時間幅を動的に決定する楽観的な分散論理時刻管理手法である．イベントの空間的影響範囲を考慮しイベント限界を各 LP ごとに求めることにより BTB と比較し，ロールバックする LP の個数を減少させることが可能となる．

図4は，一次元的に配置されたLPが隣接LPにイベントを送信する問題における*BSTB*のイベント生成関係を示したダイアグラムの一例である．図において横軸はシミュレーション時刻，縦軸はイベントを送受信するLPの配置を表す．

*BSTB*では下記に示す手順に従い，各LPで同期して，1つのイベントを投機的に実行し，実行終了後イベント実行順序が因果関係を満たすものであったかどうかを判定するサイクルを繰り返す．

(1) 各LPで，イベントキュー中から，最もタイムスタンプの値が小さいイベントを取り出す．このイベントのタイムスタンプの値を t_i とする．

(2) 全LPで t_i の最大値 t_{sup} を求める．

(3) 各LPで，(1)でイベントキューから取り出したイベントを実行する．このイベントを実行中に，新たに生成したイベントの送信先を LP_j，タイムスタンプの値を T_j とする．イベント送信先の LP_j には，タイムスタンプの値 T_j のみが送られ，イベント自体は生成元のLPで保存しておく．

(4) 新たに生成された時刻 T_j，送信先 LP_j のイベントが時刻 t_{sup} までに影響を及ぼす空間的影響範囲は，**図5**に示すように，次回以降のサイクルで，このイベントにより生成されるイベントの空間的影響範囲も含めて，たかだか LP_j を始点とする，接続距離 $(t_{sup}-T_j)/\tau$ の範囲である．すべてのイベントの実行が終了した後，各LPで各々に送信されてきたイベントのタイムスタンプ値の最小値 $\min(T_j)$ を求める．

$t_{sup} < \min(T_j)$ であるならば，新たに生成され LP_j に送られるイベントにより，このサイクルで実行したイベントに影響を及ぼすことがないと判断される．

$\min(T_j) < t_{sup}$ であるならば，LP_j から接続距離が $(t_{sup}-\min(T_j))/\tau$ 内にあるLPに $\min(T_j)$ の値を伝搬させる．各LPでは伝搬された $\min(T_j)$ の最小値を各LPにおける局所的イベント限界とする．

(5) 局所的イベント限界よりも将来の時刻のタイムスタンプを持つイベントを実行したLPは，次以降のサイクルで，実行したイベントのタイムスタンプ値よりも，過去の時刻のタイムスタンプを持ったイベントを受け取る可能性があるため，生成したイベントを棄却し，自己の状態値をイベント実行前の値に戻すロールバック処理を行う．

局所的イベント限界よりも過去の時刻のタイムスタンプを持つイベントを実行したLPは，このサイクルで新たに生成したイベントを送信先LPに送る．

図3，図4中の LP_4 が実行したイベントの例からも分かるように，Breathing Space-Time Bucketsアルゴリズムは，Breathing Time Bucketsアルゴリズムと比較し局所的なイベント限界を用いることにより，イベントの影響を空間的に局所化しロールバックされるLPの数を減少できる効果がある．

3. 並列オブジェクト指向シミュレーション環境 OSim

OSim[2]は，並列環境下において各種分散時刻管理方式を切り替えて使用することが可能な並列オブジェクト指向シミュレーション実行環境であり，ユーザは，プログラムを書き換えることなく，各種分散論理時刻管理手法を選択して使用することができる．*OSim*では，Time Warp[8]，Breathing Time Buckets[7]，Time Driven[9]など既提案のアルゴリズムおよび本稿で新たに提案したBreathing Space-Time Bucketsアルゴリズムを利用できる．

また*OSim*は，OCore[10]を用いて記述している．OCoreはRWCP: Real World Computing Partnership（新情報処理開発機構）プロジェクトにおいて開発された並列オブジェクト指向言語であり，従来の並列オブジェクト指向言語の上に，オブジェクトの集合の構造化と，オブジェクト間の通信の効率化を目的としたCommunityの機能を提供している．また，ローエンドのワークステーションクラスタからハイエンドのParagon，CM-5などの超並列計算機までプログラムを書き換えることなしにシームレスに対応することが可能である．

*OSim*では，**図6**に示すように，1つのLPを，時刻管理オブジェクト1個と，アプリケーションオブジェクト複数個からなる複合オブジェクトとして実現している．時刻管理オブジェクトは，LPの状態値の管理，メッセージキューの管理などLP間の論理時刻に関する因果関係を管理するオブジェクトであり，*OSim*のライブラリとして提供される．アプリケーションオブジェクトは，シミュレーション対象の動作を実行するオブジェクトでありユーザが記述する．

1つのアプリケーションオブジェクトはLPの1時刻分の状態を保存する．Time Driven，BTB，*BSTB*アルゴリズムの場合，各LPあたり状態値の参照用と，状態値の更新用の2つのアプリケーションオブジェク

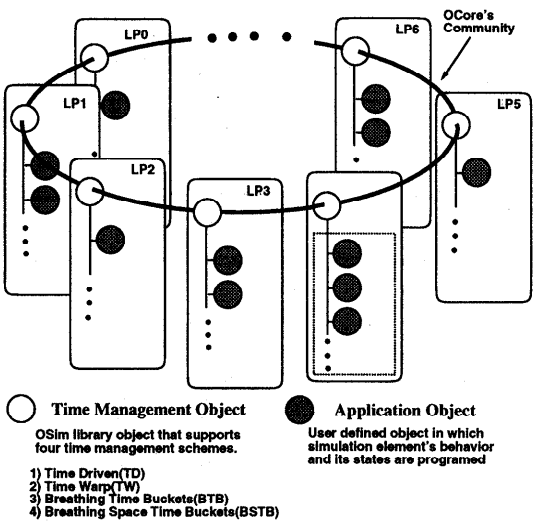


図 6 *OSim* における論理プロセスの内部構造
Fig. 6 Structure of logical processes in *OSim*.

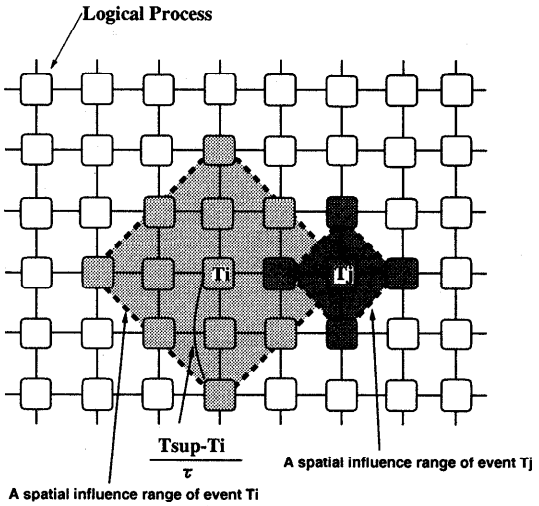


図 7 Torus network におけるイベントの空間的影響範囲
Fig. 7 Spatial influence ranges of the events in a torus network.

トを持つ．Time Warp アルゴリズムの場合，複数回の状態更新に遡りロールバックが起こる可能性があるため，LP が状態更新されるたびにアプリケーションオブジェクトのコピーを 1 つ作成し状態値の履歴を保存する．LP が管理するアプリケーションオブジェクトの数がシステム規定値を超えた場合，すべての LP でイベント実行を中止し各 LP におけるシミュレーション時刻の最小値 GVT を求める．GVT 以前のタイムスタンプを持ったイベントが生成されることはないので，GVT 以前の時刻の状態値を保存するアプリケーションオブジェクトを破棄する Fossil Collection を行う[8]．

各分散論理時刻管理技法用の時刻管理オブジェクトを用意し，LP へのメッセージ送信，アプリケーションオブジェクトでのメッセージ受信に同一インタフェースを提供することにより，ユーザプログラムを書き換えることなく，ライブラリを変更することで分散論理時刻管理手法を切り替えることを可能としている．

1 つの LP を構成するオブジェクトは，すべて同一 PE 上に配置される．また時刻管理オブジェクトの集合を Community として定義し，LP 間の同期の効率化をはかっている．

4. 実行結果

評価モデルに複雑なモデルを用いた場合，実行時の性能を決定する要因の解析が複雑になり，各手法の特徴を見いだすことが困難になる．しかし，単純なモデルを用いた場合，実問題との差が大きくなり有用な結果を得ることができない．本稿では，移動体シミュレーションのモデルとして，2 次元トーラス結合の論理プロセスのネットワーク☆（図 7）をシミュレーションモデルとして用い文献 6)，11)，12) でとられたように，モデルのパラメータを変化させることにより実際問題で生じる状況を作り出し，本稿で新たに提案した Breathing Space-Time Buckets アルゴリズムの実行性能を評価した．

比較のため同一問題を Breathing Time Buckets（BTB），Time Driven（TD）[9] でも実行した．TD は各 LP 間で同期してシミュレーション時刻を進めるアルゴリズムである．LP が，その時刻のイベントを持っている場合そのイベントを実行し，持っていない場合その時刻では何もしないといった時刻管理を行う．

トーラス結合ネットワークの規模は 16 ×16 で，並列実行する LP の数は 256 個である．各 LP は，イベントをタイムスタンプの順に実行する．論理プロセスにおけるイベント実行は，4 近傍の論理プロセスから，論理プロセスを 1 つランダムに等確率で選択し，タイムスタンプ $T+\Delta t$（T は実行中のイベントのタイムスタンプ，Δt は，イベントの最小遅延時間 τ，平均遅延時間 T_{ave} としたとき $[\tau, 2T_{ave}-\tau]$ の一様分布に従う値）のイベントを送信する．また初期状態イベントにおけるイベントの総数は 768（1LP あたり平

☆ たとえば，交通流シミュレータのような移動体シミュレーションの場合だと，空間を適当な領域に分割し，各領域内の道路網や信号機を LP に，自動車の動作をイベントに対応させるようなモデル化を行う．

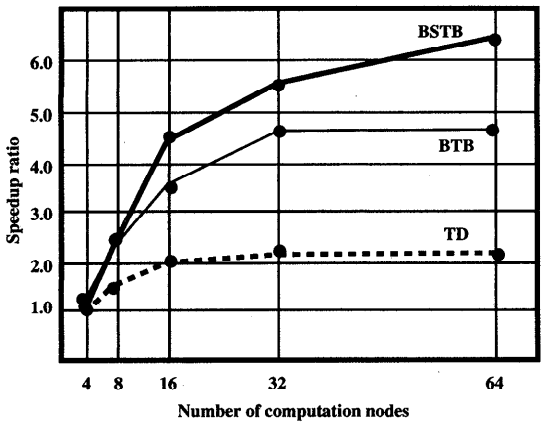


図 8 プロセッサを変化させたときの実行速度の変化
Fig. 8 Relative speedup ratio $(\frac{Execution\ time(4nodes,using\ TD)}{Execution\ time(N\ nodes)})$.

均 3 個）とした☆.

タイムスタンプの値が T_i であるイベントが時刻 T_{sup} までに及ぼす空間的影響範囲は，最小遅延時間が τ である場合，図 7 に示すように，そのイベントの送信先 LP を中心とする接続距離 $(T_{sup}-T_i)/\tau$ 内にある LP となる.

実行する演算ノード数，イベントの最小遅延時間 τ，平均遅延時間 T_{ave} をパラメータとして変化させ，Interl Paragon XP/S（クロック 50 MHz，各演算ノードの主記憶 16 MB，OSF/1 R1.3）上で実行した.

図 8 は，実行する演算プロセッサ数を 4，8，16，32，64 と変化させた（平均遅延時間 $T_{ave}=10$，最小遅延時間 $\tau=6$）場合の実行速度の変化を示している．実行速度は，Time Driven アルゴリズム，4 演算ノードで実行した場合の演算速度を 1 とした値である.

Time Driven（TD）では 16 演算ノード，Breathing Time Buckets（BTB）では 32 演算ノードで演算速度が飽和し台数効果がみられなくなるが，Breathing Space-Time Buckets（*BSTB*）では 64 演算ノードでもまだ台数効果が出ている．これは *BSTB* では，イベント限界時刻を局所的に求めているので，全体で同期をとるコストが減少した効果と思われる.

図 9 は，平均遅延時間 T_{ave} を 10 に固定し，最小遅延時間 τ を 2，4，6，8 と変化させた（演算ノード数は 64）場合の実行速度の変化を示している．τ の変化にともないイベントの生成パターンが変化し実行時間にも影響を与えるため，実行速度を各 τ の TD での演算速度を 1 としたときの値で示している.

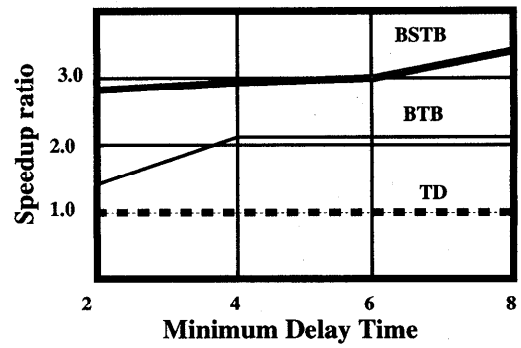


図 9 最小遅延時間を変化させたときの実行速度の変化
Fig. 9 Relative speedup ratio $\frac{Execution\ time(using\ TD)}{Execution\ time(using\ BSTB,BTB)}$.

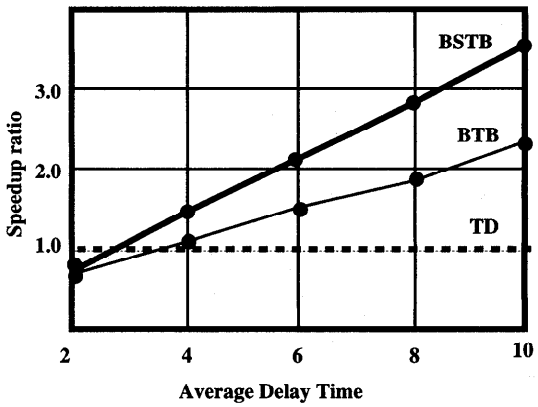


図 10 平均遅延時間を変化させたときの実行速度の変化
Fig. 10 Relative speedup ratio $\frac{Execution\ time(using\ TD)}{Execution\ time(using\ BSTB,BTB)}$.

BSTB では，最小遅延時間が大きいほど，その実行サイクルで新たに生成されたイベントの空間的範囲が狭くなりイベント限界が局所化される効果が高まるため，実行速度も向上している．BTB で最小遅延時間が小さいときに実行速度が低下するのは，最小遅延時間が小さいほど，イベントの時間的分布が広がり，イベント限界がより小さい時刻のところで設定されロールバックする LP の数が増加するためと思われる.

図 10 は，平均遅延時間 T_{ave} を 2，4，6，8，10 と変化させた（演算ノード数は 64，最小遅延時間は $0.9T_{ave}$）場合の実行速度の変化を示している．T_{ave} の変化にともないイベントの生成パターンが変化し実行時間にも影響を与えるため，実行速度を各 T_{ave} の TD での演算速度を 1 としたときの値で表示している.

平均遅延時間が小さい場合，各論理時刻で実行されるイベントの数が多くなり TD アルゴリズムでの並

☆ BTB，*BSTB* では，各実行サイクルにおいてすべての LP がおおむねイベントを 1 つずつ実行するので，並列度は全 LP 数である 256 となる．TD では同一シミュレーション時刻のタイムスタンプを持つ LP の数だけの並列度となり，最小遅延時間，平均遅延時間の値により並列度は異なる.

列度が高くなるため，1つのイベントを実行する実行コストが低い TD が BTB，*BSTB* より高速となっている．

以上の実行結果より，*BSTB* は BTB と比較し総じて演算速度が高く，特に演算ノード数増加にともなう処理速度向上の飽和に対する耐性が BTB より高く，また最小遅延時間減少にともなう演算速度の低下の割合が BTB より小さいことが分かった．また平均遅延時間が小さい場合，同一シミュレーション時刻のイベント数が増加し TD でも並列度が十分大きいため BTB，*BSTB* など投機的実行にともなうオーバヘッドが必要となる手法よりも TD の方が高い演算速度を達成することが分かった．

5. おわりに

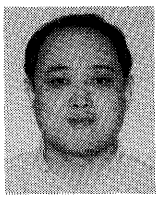
本稿ではイベントの空間的影響範囲を考慮した分散論理時刻管理方式である Breathing Space-Time Buckets アルゴリズムを提案した．また並列計算機上で大規模離散事象シミュレーションを行うための並列オブジェクト指向シミュレーション環境 *OSim* について述べた．移動体シミュレーションのモデルとして 16 × 16 トーラスネットワークモデルを用い，本稿で提案した Breathing Space-Time Buckets アルゴリズムの有用性を示した．

本稿は RWCP: Real World Computing Partnership （新情報処理開発機構）プロジェクトにおいて 1992 年度から 1996 年度までに超並列三菱研究室として実施した研究内容に基づいたものである．

参考文献

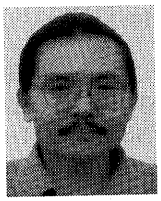
1) Fujimoto, R.M.: Parallel Discrete Event Simulation, *Comm. ACM*, Vol.33, No.10, pp.30–53 (1990).
2) 阿部一裕ほか：超並列オブジェクト指向シミュレーション環境 OSim, *Proc. JSSST Workshop on Object Oriented Computing*, S3-2, 研究会資料シリーズ No.2, ISSN 1341-870X (1996).
3) Ozaki, A., et al.: Parallel Car Traffic Simulation Based on Space-Time Object, *8th European Simulation Symposium*, Vol.2, pp.33–37 (1996).
4) Furuichi, M., et al.: A Space-Time Object: An Object Oriented Model for Parallel Simulation, *Object Oriented Siumulation Conference*, pp.86–91 (1996).
5) Steinman, J.: Multi-Node Test Bed: A Distributed Emulation of Space Communications for the Strategic Defense System, *Proc.21st Annual Pittsburgh Conference on Modeling and Simulation*, Vol.21, Part 3, pp.1111–1115 (1990).
6) Lubachevsky, B.D.: Efficient Distributed Event-Driven Simulations of the Multiple-Loop Networks, *Comm. ACM*, Vol.32, No.1, pp.111–123 (1989).
7) Steinman, J.: SPEEDES: Synchronous Parallel Environment for Emulation and Discrete Event Simulation, *Proc. SCS Western Multiconference on Advances in Parallel and Discrete Simulation*, Vol.23, Part 3, pp.1111–1115 (1991).
8) Jefferson, D.: Virtual Time, *ACM Trans. Programming Languages and Systems*, Vol.7, No.3, pp.404–425 (1985).
9) Chandy, M. and Misra, J.: Distributed Simulation: A case study in design and verification of distributed programs, *IEEE Trans. Softw. Eng.*, SE-5, 5, pp.440–452 (1979).
10) Konaka, H., et al.: Adaptive Data Parallel Computation in the Parallel Object-Oriented Language *OCore*, *Proc. 2nd Intl. Euro-Par Conf.*, Vol.I, 1123 of Lecture Notes in Computer Science, pp.587–596 Springer-Verlag, (1996).
11) 高井峰生，山城登久二，成田誠之助：離散事象シミュレーションにおける保守的同期手法の評価，情報処理学会論文誌，Vol.37, No.11, pp.2010–2019 (1996).
12) Fujimoto, R.M.: Performance Measurements of Distributed Simulation Strategies, *Trans. SCS*, Vol.6, No.2, pp.89–132 (1989).

(平成 10 年 6 月 1 日受付)
(平成 10 年 12 月 7 日採録)



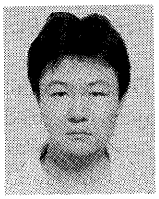
阿部 一裕（正会員）

1965 年生．1988 年大阪大学工学部応用物理学科卒業．1990 年同大学院応用物理学専攻博士課程前期修了．同年 4 月より三菱電機（株）中央研究所に勤務．1992 年度から 1996 年度までリアルワールド・コンピューティング（RWC）プロジェクトに参画し，並列分散シミュレーション方式の研究開発に従事．現在同社先端技術総合研究所において，ネットワークエージェントシステムの研究開発に従事．モバイルエージェント，組織間交渉方式，並列分散シミュレーションに興味を持つ．



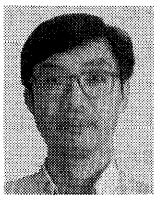
古市 昌一（正会員）

1958年生．1982年広島大学総合科学部総合科学科卒業．同年4月より三菱電機（株）情報電子研究所に勤務．1992～1994年イリノイ大学アーバナシャンペン校コンピュータサイエンス学科大学院に留学，修士課程修了．現在三菱電気（株）情報技術総合研究所において，並列分散シミュレーションシステムの研究開発に従事．負荷バランス，オブジェクト指向シミュレーション，分散協調型シミュレーションに興味を持つ．ACM，IEEE Computer Society，米国シミュレーション学会（SCS）各会員．



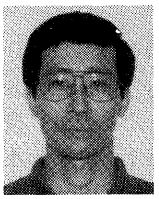
尾崎 敦夫（正会員）

1964年生．1988年九州工業大学工学部情報工学科卒業．1990年同大学院電気工学部計算機コース博士課程前期修了．同年4月より三菱電機（株）情報電子研究所に勤務．1992年度から1996年度までリアルワールド・コンピューティング（RWC）プロジェクトに参画し，並列シミュレーションの研究開発に従事．現在同社情報技術総合研究所において，並列分散シミュレーションシステムの研究開発に従事．負荷バランス，オブジェクト指向シミュレーション，並列分散協調型シミュレーションに興味を持つ．1996年 European Simulation Symposium Best Paper Award 受賞．



田中 秀俊（正会員）

1963年生．1986年東京大学工学部計数工学科卒業．同年4月三菱電機（株）入社．1989～1995年（財）新世代コンピュータ技術開発機構に出向，現在三菱電機（株）情報技術総合研究所において，主にデータ解析および最適化に関する研究開発に従事．IEEE Computer Society 会員．



中島 克人（正会員）

1953年生．1977年京都大学工学部電気第二工学科卒業．1979年同大学院修士課程修了．同年三菱電機（株）に入社し，汎用・専用計算機の開発に従事．1982年より第五世代コンピュータ・プロジェクトに参画し，逐次型および並列型推論マシンのアーキテクチャ/言語処理系等の研究開発に従事．1992年よりリアルワールド・コンピューティング（RWC）プロジェクトに参画．現在，同社情報技術総合研究所において，並列・分散処理向けアーキテクチャおよびミドルウェアの研究開発等に従事．最適設計・スケジューリング技術等にも興味を持つ．工学博士．IEEE Computer Society 会員．