階層型マルチリングによるGVT決定アルゴリズム

小林真也[†] 福田宗弘[††]

分散並列型の離散事象シミュレーション技法であるタイムワープメカニズム（Time Warp Mechanism）では，各プロセッサで処理を行っているイベントの論理時刻の最小時刻印であるGVT（Global Virtual Time）を正しく決定することが求められる．本論文では，各プロセッサを階層化されたリング状に論理的に接続し，そのリングに沿って制御トークンを巡回することでGVTを決定するアルゴリズムの提案を行う．

Hierarchical Multi Rings GVT Argorithm for Time Warp Mechanism

Shin-ya Kobayashi[†] and Munehiro Fukuda[††]

It is necessary for Time Warp Mechanism to compute the smallest timestamp of unprocessed events as GVT (Global Virtual Time). We propose a new argorithm that calculates GVT by circulating tokens among processors logically organized in a hierarchical multi-ring network.

1. はじめに

離散事象シミュレーションでは，イベント間の因果関係を正しく保ちながら処理を進める必要がある．しかし，分散した複数のプロセッサで，それぞれ個別にイベントを同時実行する分散並列型のシミュレータでは，シミュレートされる世界におけるイベントのすべてを把握することは容易ではないために，結果的に，イベント間の因果関係を正しく保ちながら処理を進める制御法の実現は容易でない．この分散並列型シミュレーションにおける制御方法には，因果関係の違反をまったく許さない保守的アルゴリズム（Conservative Algorithms）と，一時的な違反を許す代わりに，違反の検出を行い，その後ロールバックにより違反を解消する楽観的アルゴリズム（Optimistic Algorithms）がある．後者の違反の検出，解消を行う方法としては，タイムワープメカニズム（Time Warp Mechanism）が広く知られており，主に米国で活発な研究がなされている[1),5)]．また，国内では，タイムワープメカニズムと同様の制御法として，先行制御方式が提案されている[2)~4)]．

† 愛媛大学工学部
Faculty of Engineering, Ehime University
†† 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, The University of Tsukuba

タイムワープメカニズムでは，ロールバックを可能とするために，処理履歴の保存を行う必要があるが，記憶容量の点から，処理履歴を無制限に保存することはできない．したがって，ロールバックにより取り消されることがないと確定したイベントの処理履歴を消去する必要がある．また，シミュレーション結果の確定という点からも，確定できる処理の検出は必要となる．シミュレーション結果の確定は，イベントの因果関係による順序関係が，時刻印の順によって決まる順序関係に含まれるという事実を利用することよって行われる．つまり，2つの互いに因果関係のあるイベントが存在するときに，必ず，時刻印の小さいイベントが，時刻印の大きいイベントに先行している．したがって，システムにおけるすべてのプロセッサで未完了のイベントの論理時刻印の最小値（GVT: Global Virtual Time）を求めれば，GVTの値よりも小さい時刻印を持つイベントの処理は，現在未完了であるイベントの影響を受けることなく，確定したイベント（処理）であると判断できる．したがって，タイムワープメカニズムを用いたシミュレータでは，分散環境において，GVTを正しく決定できることが必須である．

本論文では，プロセッサを論理的に階層的なリング状に接続することで，GVTの決定を行うアルゴリズムの提案と，その正当性の証明を行う．また，GVT決定に要するメッセージ数や時間の点から従来方式との比較を行う．

2. タイムワープメカニズム

タイムワープメカニズム（Time Warp Mechanism）は，分散並列型シミュレーションに対する楽観的な制御である．タイムワープメカニズムでは，各プロセッサは，他のプロセッサと非同期的にイベントの処理を行うことができる．しかし，非同期的なイベント処理は，論理時刻上で過去からのメッセージが送られてくる可能性があり，これは矛盾状態となる．そこで，タイムワープメカニズムでは，イベントの処理履歴を保存しておき，矛盾状態が発生した場合には，履歴に基づきロールバックを行うことで矛盾の解消を行う．しかしながら，タイムワープメカニズムは下記に示す2つの問題を持つことになる．

(1) 履歴を無制限に保存することは，メモリ資源を無限に要求することになる．
(2) 入出力操作のように，いったん実行してしまうと取り消せない操作が存在する．

これら2つの問題を同時に解決する方法として,（実時間上の）将来に起こりうるロールバックで戻される論理時刻の下限 T_L を求める方法がある．この下限 T_L 以前の履歴は保存する必要がなく，また，T_L 以下の時刻印である入出力操作はロールバックすることがないので，処理を完了することができる．この下限が GVT（Global Virtual Time）である．GVT の定義は，

定義 1 実時刻 T_R における GVT の値 GVT_T は，実時刻 T_R における未完了のイベントの最小時刻印である．

となる．なぜならば，イベント間の因果関係において，（論理）時刻印の大きいイベントが，時刻印の小さいイベントに先行することはないため，実時間上で将来起こりうるロールバックで，現在未完了のイベントの時刻印の最小値以前に戻ることはないためである．

3. GVT 決定アルゴリズム

3.1 トランジェントメッセージ問題

システム内に論理時刻を管理するプロセッサ〔以下，管理プロセッサ（Manager）と記す〕を設置し，すべてのプロセッサの処理を停止させて，管理プロセッサに各プロセッサにおける，未完了イベントの時刻印の最小値（LVT: Local Virtual Time）を報告させても，管理プロセッサは GVT を正しく決定することができない．これは，プロセッサが処理を停止していても，まだ受信側プロセッサに到着していない移動中のメッセージがネットワークに存在する可能性があるためである．たとえば**図 1** では，管理プロセッサからの LVT 報告指示に対して，プロセッサ P_A と P_B は，それぞれ 20 と 15 を返し，GVT は 15 として求められるが，実際には，時刻印 10 の送信中メッセージ（トランジェントメッセージ）があるために，正しい GVT は 10 である．このように，送信中メッセージの存在によって引き起こされる問題をトランジェントメッセージ問題と呼ぶ．

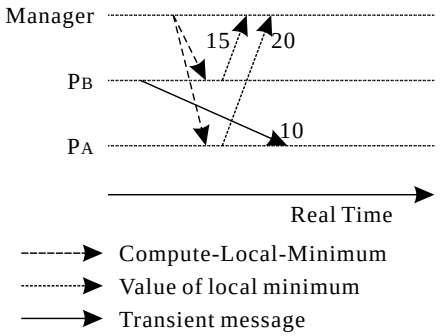

図 1 トランジェントメッセージ問題
Fig. 1 Transient message problem.

3.2 同期的 GVT 決定アルゴリズム

トランジェントメッセージ問題に対する解決法として，同期的 GVT 決定アルゴリズムがある．これは，すべてのメッセージに対して受信側のプロセッサが確認メッセージを送ることで，送信側と受信側のプロセッサの少なくともどちらか一方にトランジェントメッセージに対する責任を持たせようという方法である．具体的には，以下のアルゴリズムによって GVT を決定する．

(1) 管理プロセッサが，各プロセッサに GVT 計算の開始を宣言する “GVT 計算開始” メッセージを放送する．
(2) “GVT 計算開始” メッセージを受け取ると，各プロセッサはイベントの処理を中断し，“GVT 計算開始確認” メッセージを管理プロセッサに送り返す．各プロセッサは，管理プロセッサから別のメッセージが届くまで停止する．
(3) 管理プロセッサは，各プロセッサからの “GVT 計算開始確認” メッセージを受信すると，“局所最小値決定” メッセージを放送する．
(4) “局所最小値決定” メッセージを受信すると，各プロセッサは，プロセッサ内の未処理イベントと，自分が送信を行ったが，まだ確認メッセージを受け取っていないメッセージの時刻印の最小値（LVT）を求め，この値を管理プロセッサに送る．

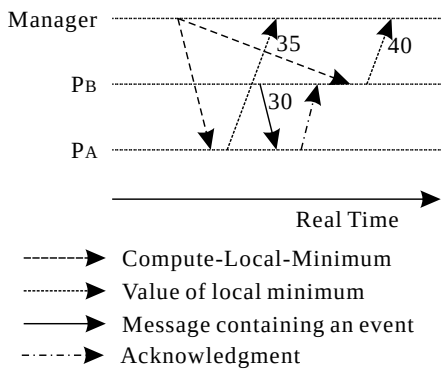

図 2 同時報告問題
Fig. 2 Simultaneous reporting problem.

(5) 管理プロセッサは，各プロセッサから局所最小値を受信すると，それらの最小値を GVT とし，その値を各プロセッサへ放送する．

3.3 同時報告問題

上記の同期的 GVT 決定アルゴリズムでは，GVT 決定時に，すべてのプロセッサがイベントの処理を停止する状態を得る必要があり，処理に遅延がもたらされるという問題がある．そこで，管理プロセッサは，最初に “GVT 計算開始” メッセージを送ることなく，“局所最小値決定” メッセージを放送し，各プロセッサは，“局所最小値決定” メッセージを受け取った場合，ただちに，LVT を管理プロセッサに伝達するものの，処理を中断することなく継続できるとすると，管理プロセッサは正しい GVT を決定できない[6]．その様子を**図 2** に示す．

図 2 において，プロセッサ P_A が，管理プロセッサからの局所最小値決定メッセージを受け取ると，局所最小値が 35 であると報告する．一方，プロセッサ P_B に送られた局所最小値決定メッセージが通信ネットワーク上で遅延が発生し，しばらく後まで届かなかったとする．その間に，プロセッサ P_B は時刻印 30 のメッセージをプロセッサ P_A に送り，時刻印が 40 の次のイベントの処理を始めだす．また，プロセッサ P_A から確認メッセージが戻ってきた後に，プロセッサ P_B が局所最小値決定メッセージを受け取ったとする．これに対し，プロセッサ P_B は，次のイベントの時刻印である 40 を局所最小値として報告する．この状況では，P_A はすでに，局所最小値として 35 を報告しているので，たとえ確認メッセージを使ったとしても，時刻印 30 メッセージは，GVT 決定に反映されず，管理プロセッサは，誤った値である 35（35 と 40 の最小値）を GVT として決定する．

3.4 Samadi の GVT 決定アルゴリズム

Samadi の GVT 決定アルゴリズム[7] は，基本的には，同期的 GVT 決定アルゴリズムと同様に，送信プロセッサに確認メッセージを送り返すものであるが，プロセッサが管理プロセッサからの局所最小値決定メッセージを受信以降，GVT の値を受け取るまでの間に，メッセージを受け取った場合には，それに対する確認メッセージにタグの付加を義務づける．たとえば，図 2 において，P_A は，プロセッサ P_B から送られてきた時刻印 30 のメッセージに対しては，タグつきの確認メッセージを送り返す．P_B は，局所最小値の決定を行う際には，(1) プロセッサ内の未完了イベントの時刻印，(2) すでに送信済みであるが，まだ確認メッセージが送り返されていないメッセージの時刻印，(3) 最新の GVT 計算以降に受け取ったタグつき確認メッセージの時刻印の最小値を局所最小値とする．したがって，図 2 で，P_B が局所最小値を管理プロセッサに報告する前に，時刻印 30 のメッセージに対するタグつき確認メッセージが届いたとすると，上記の (3) によって，正しく局所最小値の報告が行われる．一方，P_B の局所最小値の報告までに，確認メッセージが送り返されてこなければ，上記の (2) によって，正しく局所最小値の報告が行われる．

3.5 Bellenot の GVT アルゴリズム

Bellenot の GVT アルゴリズム[8] では，N 台のプロセッサ（1 番から N 番）のうち，前半の 1 番から $\lceil N/2 \rceil$ 番のプロセッサと，後半の $\lfloor N/2 \rfloor$ 番から N 番のプロセッサで，それぞれ，1 番目のプロセッサと N 番目のプロセッサをルートとする同型対称の 2 進木構造を 2 つ構成し，さらに 2 つの木構造の葉を相互に接続することで，メッセージ経路グラフを構成する☆．

GVT の決定は以下のように行われる．まず，1 番のプロセッサから，N 番のプロセッサの方向に向かって，開始メッセージが送られる．各プロセッサは開始メッセージを受け取ると，受け取り確認メッセージを送信元へ送り返す．また，開始メッセージを受け取ると（2 つの開始メッセージを受け取るプロセッサでは，2 つのメッセージを受け取ると），各プロセッサは，次のプロセッサへ開始メッセージを送る．

N 番のプロセッサは，開始メッセージを受け取ると，開始メッセージとは逆の方向に，LVT の値を含んだ，LVT メッセージを送る．送られてきた LVT メッ

☆ プロセッサ数が奇数の場合は，中間のプロセッサを，2 つの木構造の双方に入れて構成し，相互の葉を接続する際に，1 つのプロセッサとして融合する．

セージの値と，自らの LVT の値の最小値を書き込んだ LVT メッセージを 0 番のプロセッサの方向へ送る．

0 番のプロセッサが LVT メッセージを受け取ると，そのメッセージに書き込まれた値と，自らの LVT の値から GVT を決定する．その後，N 番の方向へ向かって，GVT を伝達する．

3.6 Mattern の GVT 決定アルゴリズム

Mattern の GVT 決定アルゴリズム[9] では，メッセージに対する確認メッセージを使用しない代わりに，各プロセッサを論理的にリング状に接続し，そのリング上に，管理プロセッサから発信される制御メッセージを 2 周巡回させることで，トランジェントメッセージのに対処する方法である．

Mattern の方法では，一巡目の制御メッセージ巡回時に，制御メッセージに対し，一巡目の制御メッセージが送られるまでに，各プロセッサ宛てに送られたメッセージの数を書き込む．したがって，各プロセッサが二巡目の制御メッセージを受け取る際には，一巡目の制御メッセージが他のノードを巡回した時点以前に，自ノード宛てに送られたメッセージの総数が判明する．そこで，各ノードは，それらのメッセージをすべて受け取るまでは，次の制御メッセージを次のノードに転送しない．このようにすることで，二巡目には，一巡目にトランジェントメッセージとなっていたメッセージが消滅する．また，制御メッセージには各ノードでの未完了イベントと，一巡目の制御メッセージの送信以降に行ったメッセージの時刻印の最小時刻印がかかれており，管理プロセッサが，この値の最小値が GVT として求める．

3.7 メッセージ数と処理時間

上記の Samadi のアルゴリズム，Mattern のアルゴリズム，Bellenot のアルゴリズムの，GVT 決定およびすべてのプロセッサへの GVT の伝播に必要となるメッセージ数，ならびに，プロセッサ間での通信 1 回あたりの時間を単位とした処理時間は下記のようになる．なお，システム内の（管理プロセッサも含めて）プロセッサ数を N と表す．

まず，Samadi のアルゴリズムでは，管理プロセッサがシステム内のすべてのプロセッサとメッセージを交換することで GVT を決定し，また，すべてのプロセッサとメッセージを交換することで GVT を伝達する．そのため，必要となるメッセージの総数は $3(N-1)$ となる．また，必要となる時間は，管理プロセッサからの放送型通信が一度に行え，かつ，各プロセッサから管理プロセッサへの通信が一度に行えるならば，3 となる．また，管理プロセッサでの受信が逐次的である場合には，$1+(N-1)+1=N+1$ となる．Samadi のアルゴリズムは，放送型の通信を利用しているために，メッセージ数や通信回数を単位とした処理時間の点では優れているが，管理プロセッサで行うべき処理の量が $O(N)$ で増加するため，プロセッサ数の増加にともなって，より高速な動作が可能な管理プロセッサを用意しなければならず，拡張性の点での大きな障害となる．

次に，Bellenot のアルゴリズムでは，GVT の決定のために必要なメッセージの総数は $4(N-1)$ となる．また，メッセージ経路グラフの一方の端からもう一方の端までのステップが $2\lceil\log_2(\frac{N}{2}+1)\rceil-1$ となるため，処理時間は $6\lceil\log_2(\frac{N}{2}+1)\rceil-3$ となる．

また，Mattern のアルゴリズムでは，GVT の決定のために，制御メッセージを二巡させ，GVT の伝達のためにさらに一巡させるため，メッセージ数は $3N$ となる．また，処理時間も同様に $3N$ となる．

4. 階層マルチリング型 GVT 決定法

4.1 アルゴリズム

我々が提案する階層マルチリング型 GVT 決定法では，各プロセッサを，**図 3** に示すような接続関係で論理的に接続する．リングの結合関係は階層的であり，プロセッサと階層的なリングの接続は以下のように規定される．なお，このプロセッサ間の論理的な結合関係は，シミュレーションの開始に先立ち，利用者が明示的に決定することもできるし，あるいは，各リング中のプロセッサ数 n と各レベルのリング数 l を決め

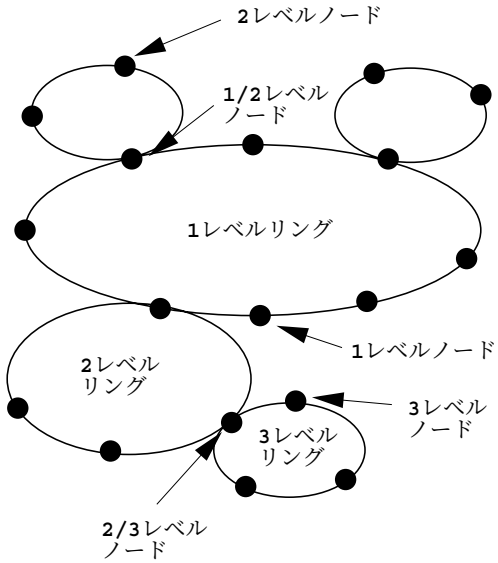

図 3 階層マルチング型結合
Fig. 3 Hierarchical Multi Rings Connection.

れば機械的に決定することも可能である．

- プロセッサは必ずいずれかのリングに属する．
- 第 1 レベルのリングはただ 1 つ存在する．
- 第 N レベルのリングと第 $N-1$ レベルのリングは 1 つの共有プロセッサを介して接続されている．

なお，以下の説明において，第 N レベルのリングを単に第 N レベルリングと呼び，第 N レベルリング上のプロセッサを第 N レベルプロセッサと呼ぶ．また，第 $N-1$ レベルプロセッサでありかつ第 N レベルプロセッサであるプロセッサ（つまり，第 $N-1$ レベルリングと第 N レベルリングの接点となっているプロセッサ）を第 $N-1/N$ レベルプロセッサと呼ぶ．また，第 $N+1$ レベルと接続していない第 N レベルリング（つまり，より下位のリングと接続していないリング）をボトムレベルリングと呼び，ボトムレベルリング中のプロセッサをボトムレベルプロセッサと呼ぶ．

階層マルチリング型 GVT 決定法では，各リングごとにトークンを巡回させることで，GVT 決定に必要な情報や，確定した GVT を伝播させる．トークンは各リングに沿って，つねに巡回を続けている．GVT を決定できるのは，第 1 レベルプロセッサのみであり，ボトムレベルリングから，第 1 レベルリングへ向かって，GVT 決定に必要な情報が伝わっていき，第 1 レベルからボトムレベルリングへ向かって，確定した GVT が伝わっていく．

アンチメッセージ☆ではない，通常のメッセージに対しては，確認メッセージが送り返される．また，メッセージを受信したことにより，プロセッサでロールバックが発生した場合には，そのメッセージの確認メッセージに，ロールバックされる時刻の情報を添付する．したがって，ロールバックを誘引したメッセージを送信したプロセッサは，そのメッセージの確認信号を受け取った時点で，ロールバックを誘引した事実と，メッセージを受信したプロセッサの論理時刻がロールバックにより，どこまで遡ったかを知る．階層マルチリング型 GVT 決定法では，ボトムレベルプロセッサ，中間レベルのプロセッサ，および第 1 レベルプロセッサが以下のような動作をすることで，GVT を決定する．なお，以下の説明で用いる用語を次のように定義する．

第 N レベルトークン：第 N レベルリング中を巡回するトークン．

ボトムレベルトークン：ボトムレベルリングを巡回するトークン．

ITA（Individual minimum Timestamp of Active Transactions）：あるプロセッサで実行中のイベントの時刻印の最小値．なお，階層型 GVT 決定法では，メッセージの送信を行うイベントは，そのメッセージに対する確認信号が送り返されるまでは，実行中であるものとして取り扱う．

LVT（Local minimum Timestamp of Active Transactions）：リングの接点以外のプロセッサでは自プロセッサの ITA．リングの接点である第 $k/k+1$ レベルプロセッサでは，自プロセッサの ITA と第 $k+1$ レベルトークンにより伝えられる第 $k+1$ レベルプロセッサの LVT の最小値．

ボトムレベルプロセッサおよび中間レベルプロセッサ

ボトムレベルや中間レベルのトークンにはそれぞれのレベルに属するすべてのプロセッサの LVT，ならびにロールバック情報と確定 GVT が書かれている．確定 GVT は第 1 レベルプロセッサが決定した GVT の値で，上位のレベルから伝播してきたものである．ロールバック情報については後述する．

ボトムレベルや中間レベルのプロセッサはトークンが巡回するごとに，以下の動作を行う．なお，ここでは注目しているレベルを第 k レベルであるとする．

- トークンに書かれている確定 GVT の値を読み出し，この値を基に確定状態にすることができるイベントを判断し，記憶領域の開放や，入出力の確定を行う．
- 巡回してきたトークンに自プロセッサの LVT の値を書き込む．
- 前回トークンが巡回して以降，今回のトークンの巡回までの間に，自プロセッサのメッセージが他のプロセッサのイベントのロールバックを誘引した場合には，ロールバックされたイベントの時刻印と ‘+’ の記号をセットにして，トークンに書き込む．この ‘+’ の記号とロールバックされたイベントの時刻印のセットを ‘+’ ロールバック情報と呼ぶ．なお，‘+’ ロールバック情報は，ロールバックされるイベントの処理プロセッサに送られるアンチメッセージではない．
- 前回トークンが巡回して以降，今回のトークンの巡回までの間に，自プロセッサのイベントが他のプロセッサのイベントによりロールバックされた場合には，ロールバックされたイベントの時刻印と ‘−’ の記号をセットにして，トークンに書き込む．この ‘−’ の記号とロールバックされたイベ

☆ アンチメッセージとは，タイムワープメカニズムにおいて，イベント情報の伝達などの目的でプロセッサ間でやりとりされる本来のメッセージの取り消しを表すメッセージである．

ントの時刻印のセットを ‘−’ ロールバック情報と呼ぶ．

また，2つのレベル（第 $k-1$ レベルと第 k レベル：$k \geq 2$）との接点となる第 $k-1/k$ レベルプロセッサであるプロセッサは，上記のほかに，下位レベルのトークンである第 k レベルトークンと上位レベルのトークンである第 $k-1$ レベルトークンの受信時に，それぞれ下記のような動作を行う．

- 第 k レベル（下位レベル）トークン受信時
  - 第 $k-1$ レベルトークンによって伝えられた確定 GVT の値を，トークンに書き込む．
  - トークンに書かれているすべてのプロセッサの LVT の値の最小値を求め，その値をプロセッサ内に記憶しておく．なお，第 $k-1$ レベルトークンには自プロセッサの LVT は書き込まない．
  - トークンに書かれている ‘+’ ロールバック情報と ‘−’ ロールバック情報を記憶し，トークンからそれらの情報を削除する．なお，第 $k-1/k$ レベルプロセッサは自らが関係する ‘+’ ロールバック情報や ‘−’ ロールバック情報は第 k レベルトークンに書き込まない．
- 第 $k-1$ レベル（上位レベル）トークン受信時
  - 下位レベル（第 k レベル）のトークンにより伝えられた ‘+’ ロールバック情報と ‘−’ ロールバック情報を第 $k-1$ レベルトークンに書き込む．
  - 下位レベル（第 k レベル）のトークン受信時に得られた LVT と自プロセッサの ITA の値の最小値を自プロセッサの LVT として第 $k-1$ レベルトークンに書き込む．
  - 第 $k-1$ レベルトークンに書かれている GVT の値を記憶しておく．

第 1 レベルプロセッサ

第 1 レベルトークンには第 1 レベルプロセッサの ITA や各レベルから伝播してきたロールバック情報，それに確定 GVT が書かれている．

第 1 レベルプロセッサの動作も基本的には，中間レベルのプロセッサの動作と同様のものであるが，GVT の決定を行うなどの若干の違いがある．具体的には，第 1 レベルのプロセッサはトークンが巡回するごとに，以下の動作を行う．

- トークンに書かれている各プロセッサの LVT の値と自プロセッサの ITA，ならびにトークン中と自プロセッサが持っている ‘+’ ロールバック情報に書かれている時刻の最小値を確定 GVT として，トークンに書き込む．また，その値を基にイベント処理の確定を行う．
- 巡回してきたトークンに自プロセッサの LVT の値を書き込む．
- 前回トークンが巡回して以降，今回のトークンの巡回までの間に，自プロセッサのメッセージが他のプロセッサのイベントのロールバックを誘引した場合には，ロールバックされたイベントの時刻印と ‘+’ の記号をセットにした ‘+’ ロールバック情報をトークンに書き込む．
- 前回トークンが巡回して以降，今回のトークンの巡回までの間に，自プロセッサのイベントが他のイベントのトークンによりロールバックされた場合には，ロールバックされたイベントの時刻印と ‘−’ の記号をセットにした ‘−’ ロールバック情報をトークンに書き込む．
- トークンに相互に対応する ‘+’ ロールバック情報と ‘−’ ロールバック情報があれば，その両方を消去する．

上記に加え，第 2 レベルとの接点にあたる第 1/2 レベルプロセッサは，以下の動作を行う．

- 第 2 レベルトークンにより伝えられた ‘+’ ロールバック情報と ‘−’ ロールバック情報を第 1 レベルトークンに書き込む．
- 第 2 レベルのトークン受信時に得られた LVT と自プロセッサの ITA の値の最小値を自プロセッサの LVT として第 1 レベルトークンに書き込む．
- 第 1 レベルトークンに書かれている GVT の値を記憶しておく．

4.2 GVT 決定および伝達に要するメッセージと時間

階層 GVT 決定アルゴリズムでは，各リングを巡回するトークン（メッセージ）は，各ノードの LVT の更新情報と，新たな GVT の伝達という役割を同時に担っている．そのため，1 回ごとのトークンの送受信に対して，LVT の情報伝達用であるのか，GVT 伝達用であるのかを明確に区別できないために，厳密な意味で，1 回あたりの GVT 決定動作のためのメッセージ数を求めることはできない．しかし，各プロセッサが行う 1 回ごとのトークンの送信が，第 1 レベルリングでの GVT の決定に反映され，また，その 1 回のトークンの送信が，間接的であるにしろ，必ず第 1 レベルでの決定に影響があること☆から，GVT の決定

☆ たとえ LVT の更新がなかったとしても，これは，LVT の更新がなかったということの明示的な報告であり，報告が行われなかったということとは，本質的に異なる．

に必要となる，メッセージの送信回数は，各ノードあたり 1 回であると見なせる．また，トークンは GVT 決定のための LVT 情報の伝達の役割と同時に，GVT の伝達の役割を担っていることから，結果的に，GVT 決定動作のためのメッセージ数を N とすることは，妥当であると考えられる．

次に，GVT 決定のために必要となる時間を通信時間の点から検討する．まず，各リングに接続している下位レベルのリング数が等しいとする．さらに，各リングごとのプロセッサ数を n，第 k レベルの各リングに接続されている $k+1$ レベルのリング数を l $(\leq n-1)$，リングの階層数を L とおくと，システム内のプロセッサの総数 N は

$$N = 1 + \sum_{x=1}^{L}((n-1)l^{x-1}) = 1 + (n-1)\frac{l^L - 1}{l-1}$$

と表せる．したがって，

$$L = \log_l\left(\frac{(N-1)(l-1)}{n-1} + 1\right) = \log_l\left(\frac{Nl - N - l + n}{n-1}\right)$$

となる．

一方，ボトムレベルのプロセッサでの LVT の更新が第 1 レベルに伝わり，その結果 GVT が更新され，さらにその GVT の値がボトムレベルに到達するまでの通信回数 C を求めると以下のようになる．まず，各リングにおいて，隣接するリングへの情報の伝播に要するプロセッサ間通信の最大回数は，$n-1$ 回となる．また，ボトムリングでの LVT 更新が第 1 レベルに伝播するまでに経由するリング数はボトムレベルと第 1 レベルを含めて L となる．したがって，

$C = 2(n-1)L$

となる．ここで $L = \log_l(\frac{Nl-N-l+n}{n-1})$ を代入すると．

$$C = 2(n-1)\log_l\left(\frac{Nl - N - l + n}{n-1}\right)$$

となり，さらに $l = n-1$ と仮定すると．

$$C = 2l\log_l\left(\frac{Nl - N + 1}{l}\right) = 2l(\log_l(Nl - N + 1) - 1)$$

となる．ただし，これも，メッセージ数と同様に，各トークンは GVT の決定のための LVT の伝達と，GVT の伝達を同時に行っているため，長期的には，$2l(\log_l(Nl-N+1)-1)$ で 2 回 GVT の決定が行われることになる．したがって，GVT 決定に要する時間は，$l(\log_l(Nl-N+1)-1)$ であると見なすことができる．また，各リングのプロセッサ数が同数であるという階層マルチリングの対象性がない，より一般的な場合では，$l\lceil\log_l(Nl-N+1)-1\rceil$ となる．

表 1 メッセージ数と処理時間
Table 1 Number of messages & Processing Time.

	Messages	Time
Samadi's	$3(N-1)$	3
Bellenot's	$4(N-1)$	$6\lceil\log_2(\frac{N}{2}+1)\rceil - 3$
Mattern's	$3N$	$3N$
Proposed	N	$l\lceil\log_l(Nl-N+1)-1\rceil$

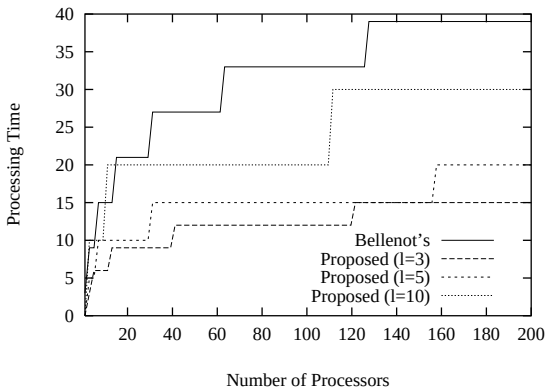

図 4 処理時間
Fig. 4 Processing Time.

これらの結果と，3.7 項で求めた従来方式におけるメッセージ数，時間をまとめると**表 1** のようになる．また，Bellenot のアルゴリズムと提案方式における，プロセッサ数に対する処理時間の増加を**図 4** に示す．図 4 より，提案方式において，プロセッサ数が同一であれば，各レベルでのリング数 l が小さい方が，処理時間の点から望ましいことが分かる．

これらの結果より，提案方式は以下の点で，従来方式よりも優れている．

(1) Samadi のアルゴリズムでは，管理プロセッサの負荷が規模の拡張に比例するのに対して，提案方式ではそのような負荷の増加が引き起こされるプロセッサは存在しない．
(2) Bellenot のアルゴリズムとの比較では，どちらも処理時間の点ではいずれも $O(\log N)$ であり，同等であるが，GVT 決定に必要となるメッセージ数がほぼ 4 分の 1 ですむ．
(3) Mattern のアルゴリズムが，処理時間の増加が $O(N)$ であるのに対して，提案方式は，$O(\log N)$ となっている．また，必要とするメッセージ数も 3 分の 1 である．

5. 正当性の証明

階層マルチリング型GVT決定アルゴリズムがGVTを正しく決定できることを示す.

以下の証明において,“第 $k-1/k$ レベルプロセッサ($k \geq 2$)が正しいLVTを決定する”とは,第 $k-1/k$ レベルプロセッサが,そのプロセッサが属している第 k レベルリング,ならびに,その第 k レベルリングに属しているプロセッサからたどれる,より下位のリングに属するプロセッサすべてにおける未完了イベントの時刻印の最小値よりも小さいか等しい値をLVTとして決定することを意味する.また,“第1レベルプロセッサが正しいGVTを決定する”とは,第1レベルプロセッサが,システム内の未完了イベントの時刻印の最小値よりも小さいか等しい値をGVTとして決定することを意味する.さらに,“第1レベルプロセッサが正しいGVTを保持する”とは,第1レベルプロセッサが保持しているGVTが,システム内の未完了イベントの時刻印の最小値よりも小さいか等しいことを意味する.

5.1 GVTの決定

まず,任意の時点で第1レベルプロセッサが正しいGVTを決定していれば,その後,最初のロールバックが発生する直後までは,第1レベルプロセッサが正しいGVTを決定することを示す.

定理1 プロセッサにロールバックが発生しなければ,そのプロセッサのITAが減少することはない.

証明1 各プロセッサで処理されるイベントは到着順に昇順の時刻印が付与される.したがって,ロールバックが発生しなければ,プロセッサで現在未完了のイベントの時刻印よりも小さな時刻印を持つイベントが新たに未完了となることはなく,プロセッサのITAが減少することはない. □

定理2 以前にボトムレベルプロセッサがトークンに正しいLVTを書き込んだ時点以降,現在までにボトムレベルプロセッサにロールバックが発生しなければ,ボトムレベルトークンに書き込まれているLVTの値は,ボトムレベルプロセッサの現在のLVTの値よりも小さい.

証明2 以前にボトムレベルプロセッサが正しいLVTをボトムレベルトークンに書き込んで以降,ボトムレベルプロセッサでロールバックが発生していないので,定理1より,ボトムプロセッサのITAが減少することはない. □

定理3 上位レベル(第 $k-1$ レベル)とボトムレベル(第 k レベル)の双方に属するボトムレベルプロセッサAが,以前に第 $k-1$ レベルトークンに正しいLVTを書き込んで以降,次回の第 $k-1$ レベルトークンへのLVTの書き込みまでに,いずれのプロセッサにおいてもロールバックが発生していなければ,次回にプロセッサAが第 $k-1$ レベルトークンに書き込むLVTは正しい.

証明3 定理1より,以前にプロセッサAが正しいLVTを書き込んで以降,ロールバックが発生してないので,プロセッサAのITAの値は減少することはなく,その値はプロセッサAにおける未完了イベントの時刻印の最小値である.

また,定理2より,プロセッサAがボトムレベルトークンによって受け取った,ボトムレベルプロセッサのLVTの値は,現在のボトムレベルプロセッサのITAの値よりも小さいか等しい.

したがって,プロセッサAが次回の第 $k-1$ レベルトークンへのLVT書き込み時に判断しているLVTの値は,ボトムレベルに属している各プロセッサにおいて未完了であるすべてのイベントの時刻印の最小値よりも小さいか等しい. □

定理4 ボトムレベルプロセッサではない第 $k-1/k$ レベルプロセッサAが,以前に正しいLVTを第 $k-1$ レベルトークンに書き込んで以降,次回の第 $k-1$ レベルトークンへのLVTの書き込みまでに,いずれのプロセッサにおいてもロールバックが発生していないという条件の下で,プロセッサAが属している第 k レベルリング中にある第 $k/k+1$ レベルプロセッサBが正しいLVTを第 k レベルトークンに書き込んでいるならば,次回にプロセッサAが第 $k-1$ レベルトークンに書き込むLVTは正しい.

証明4 プロセッサBが正しいLVTを第 k レベルトークンに書き込んでいるので,プロセッサBを介して接続されている,より下位のレベルのリングにあるプロセッサにおいて未完了であるイベントの時刻印の最小値は,プロセッサBが第 k レベルトークンに書き込んでいるLVTの値よりも大きいか等しい.

また,定理1より,以前にプロセッサAが正しいLVTを書き込んで以降,ロールバックが発生していないので,プロセッサAのITAの値は減少することはなく,その値はプロセッサAにおける未完了イベントの時刻印の最小値である.

さらに,定理2より,プロセッサAが第 k レベルトークンによって受け取った,第 k レベルプロセッサのLVTの値は,現在の第 k レベルプロセッサのITAの値よりも小さいか等しい.

したがって,次回にプロセッサAが第 $k-1$ レベ

ルトークンに書き込む LVT は正しい． □

定理 5 第 1/2 レベルプロセッサが，以前に第 1 レベルトークンに LVT を書き込んで以降，すべてのプロセッサでロールバックが発生していなければ，第 1/2 レベルプロセッサが，次回に第 1 レベルトークンに書き込む LVT は正しい．

証明 5 定理 3 と定理 4 から，帰納的推論を行うことで明らかである． □

定理 6 第 1 レベルプロセッサが，以前に正しい GVT を決定して以降，現在までにすべてのプロセッサでロールバックが発生していなければ，第 1 レベルプロセッサが認識している GVT は正しい．

証明 6 定理 5 より，第 1/2 レベルプロセッサが第 1 レベルトークンに書き込んだ LVT は，書き込みを行った時点で，正しい LVT である．また，ロールバックがそれ以降発生していないので，ITA が減少したプロセッサは存在しない．したがって，トークンに書き込まれている LVT は，ロールバックが発生するまでの任意の時刻においても正しい．

さらに，第 1 レベルプロセッサは，第 1 レベルトークンを受け取ったときに，トークンに書かれている LVT の値と自プロセッサの LVT の値を基に GVT を決定するが，自プロセッサの LVT は正しく，また，トークンに書かれている LVT の値も正しいので，トークンを受け取ったときに判断する GVT の値は正しい．

また，トークンを受け取った後，次回トークンが巡回してくるまでの間，システム内でロールバックが発生しないので，ITA が減少することはなく，第 1 レベルプロセッサが認識している GVT の値よりも，小さな時刻印であるイベントが未完了となることはない．以上より，定理は証明された． □

定理 7 ロールバックが発生する直前に第 1 レベルプロセッサが正しい GVT を認識していれば，ロールバック発生直後においても，その GVT の値は正しい．

証明 7 ロールバック発生直前に第 1 レベルプロセッサが認識していた GVT が正しいので，その値は，すべての未完了イベントの時刻印の最小値と等しいか小さい．また，ロールバックされるイベントの時刻印は，ロールバックを引き起こしたイベントの時刻印よりも必ず大きい．

したがって，ロールバックが発生しても，ロールバック発生直前に第 1 レベルプロセッサが認識していた GVT よりも小さな時刻印を持つイベントが未完了となることはない． □

定理 6 と定理 7 より，システム稼動開始時のように，第 1 レベルプロセッサが正しい GVT を決定している状態以降，最初のロールバックが発生する直後までは，第 1 レベルプロセッサは正しい GVT を決定してることが示された．

次に，ロールバックが発生しても，第 1 レベルプロセッサが正しく GVT を決定できることを示す．まず，ロールバックされるイベントの処理プロセッサ（以下 “ロールバック発生プロセッサ” と呼ぶ）の LVT が減少しなければ，GVT の決定にロールバックが影響を及ぼさないことを示す．

定理 8 ロールバックされたイベントの時刻印が，ロールバック発生プロセッサのロールバック発生直前の LVT よりも大きければ，ロールバックの発生によって，第 1 レベルプロセッサが GVT を誤って判断することはない．

証明 8 ロールバックされたイベントの時刻印が，ロールバック発生プロセッサのロールバック発生直前の LVT よりも大きいので，たとえ，ロールバックによって，ロールバック発生プロセッサの ITA が減少しても，これによって，ロールバック発生プロセッサの LVT が減少することはなく，ロールバック発生直前の LVT はロールバック発生後も正しい．

したがって，ロールバックの発生によって，誤った LVT を受け取っているプロセッサが存在することはないため，第 1 レベルプロセッサが GVT を誤って判断することはない． □

次に，ロールバックによってロールバック発生プロセッサの LVT が減少しても，GVT が正しく決定されることを以下に示す．

定理 9 ロールバック発生プロセッサが，‘−’ ロールバック情報をトークンに書き込むときに同時に書き込まれる LVT の値は，ロールバック発生後の値である．

証明 9 アルゴリズムより明らか． □

定理 10 任意のプロセッサが ‘−’ ロールバック情報をトークンに書き込むときに同時に書き込まれる LVT の値は，その ‘−’ ロールバック情報を発生させたプロセッサのロールバック発生後の LVT の値を反映した値である．

証明 10 定理 9 とアルゴリズムより明らか． □

定理 11 第 1 レベルプロセッサが ‘−’ ロールバック情報を受け取ったときに決定する GVT の値は，その ‘−’ ロールバック情報のロールバック発生プロセッサのロールバック発生後の LVT の値を反映した値である．

証明 11 定理 10 とアルゴリズムより明らか． □

定理 11 より，第 1 レベルプロセッサは ‘−’ ロールバック情報を受け取った後は，その ‘−’ ロールバック

情報のロールバック発生については，ロールバック発生による ITA の減少などの影響を考慮する必要がなくなり，LVT の値だけを考慮して GVT を決定してよいことになる．

次に，ロールバック発生から，第 1 レベルプロセッサが ‘−’ ロールバック情報を受け取るまでの間，第 1 レベルプロセッサが正しい GVT を決定できることを示す．

ロールバックが発生したプロセッサでは，ロールバック直前の ITA がロールバックされたイベントの時刻印よりも大きければ，その ITA はロールバックされたイベントの時刻印まで減少する．しかし，この ITA の減少によるロールバック発生プロセッサの LVT への影響が第 1 レベルプロセッサの GVT 決定に反映されるまでの間，第 1 レベルトークンに書き込まれている LVT の値が，ロールバックされたイベントの時刻印（すなわち，実際の LVT）よりも大きな値となることがある．したがって，このような場合においても，第 1 レベルプロセッサがロールバック発生プロセッサのロールバック発生後の LVT の値を反映した値を受け取るまで，つまり，‘−’ ロールバック情報を受け取るまでは，ロールバックされたイベントの時刻印以下の値を GVT として決定していれば，第 1 レベルプロセッサは正しい GVT を決定しているといえる．以下に，この証明を示す．

なお，以下の証明では，あるイベントのロールバック・再実行が他のイベントのロールバック・再実行を連鎖的に発生する場合，連鎖のきっかけとなった最初のロールバックを 1 次ロールバックと呼ぶ．また，1 次ロールバックでイベントがロールバックされることにより引き起こされるロールバックを 2 次ロールバックと呼び，以下，連鎖の n 番目のロールバックを n 次ロールバックと呼ぶ．

補助定理 1 1 次ロールバックの ‘+’ ロールバック情報が第 1 レベルプロセッサに届くまでは，この ‘+’ ロールバック情報を受け取る各プロセッサの LVT の値は 1 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることはない．

補助定理証明 1 1 次ロールバックの ‘+’ ロールバック情報はロールバック誘引イベントの処理後☆，最も早いトークンの巡回時に LVT とともにトークンに書き込まれる．したがって，‘+’ ロールバック情報をトークンに書き込む以前に届いたトークンには，1 次ロールバックを誘引したイベントの処理以前の LVT が書き込まれている．したがって，このときの LVT の値は，1 次ロールバックを誘引したイベントの時刻印よりも小さいか等しい．また，1 次ロールバックでロールバックされたイベントの時刻印は，そのロールバックを誘引したイベントの時刻印よりも大きい．よって，補助定理 1 は成立つ． □

補助定理 2 1 次ロールバックの ‘+’ ロールバック情報が第 1 レベルプロセッサに届くまで，第 1 レベルプロセッサで決定される GVT は，1 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることはない．

補助定理証明 2 補助定理 1 より明らか． □

補助定理 3 1 次ロールバックの ‘−’ ロールバック情報が届くまで，第 1 レベルプロセッサで決定される GVT は，1 次ロールバックでロールバックされたイベントの時刻印より大きくなることはない．

補助定理証明 3 1 次ロールバックの ‘+’ ロールバック情報が届く前に，そのロールバックの ‘−’ ロールバック情報が第 1 レベルプロセッサに届いたとすると，‘+’ ロールバック情報が届く前であったので，補助定理 2 より，第 1 レベルで決定される GVT は，1 次ロールバックでロールバックされたイベントの時刻印よりも小さい．

一方，1 次ロールバックの ‘−’ ロールバック情報が，そのロールバックの ‘+’ ロールバック情報の後で第 1 レベルプロセッサに届いたとすると，‘+’ ロールバック情報が第 1 レベルプロセッサに届くまでは，補助定理 2 より，GVT はロールバックされたイベントの時刻印よりも小さく，‘+’ ロールバック情報が届いた後は，1 次ロールバックでロールバックされたイベントの時刻印が ‘+’ ロールバック情報によって伝えられたため，第 1 レベルプロセッサで決定する GVT が 1 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることはない． □

補助定理 4 $n-1$ 次ロールバック（$n \geq 2$）の ‘−’ ロールバック情報が第 1 レベルプロセッサに届くまで，第 1 レベルプロセッサが決定する GVT が $n-1$ 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることがなければ，この $n-1$ 次ロールバックが引き起こす n 次ロールバックの ‘+’ ロールバック情報が第 1 レベルプロセッサに届くまで，第 1 レベルプロセッサが決定する GVT は n 次ロールバックでロールバックされたイベントの時刻印よりも小さい．

補助定理証明 4 n 次ロールバックの ‘+’ ロールバック情報は，$n-1$ 次ロールバックの ‘−’ ロールバック情

☆ 注：確定はされていなくてもよい．

報と同時に生成されるため，n 次ロールバックの ‘+’ ロールバック情報は $n-1$ 次ロールバックの ‘−’ ロールバック情報と同時に第 1 レベルプロセッサに届けられる．

また，仮定より $n-1$ 次ロールバックの ‘−’ ロールバック情報が第 1 レベルプロセッサに届けられるまで，第 1 レベルプロセッサが決定する GVT は $n-1$ 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることはないので，n 次ロールバックの ‘+’ ロールバック情報が第 1 レベルプロセッサに届けられるまで，GVT は $n-1$ 次ロールバックでロールバックされたイベント，つまり n 次ロールバックを誘引したイベントの時刻印よりも大きくなることはない．

さらに，n 次ロールバックでロールバックされるイベントの時刻印は，n 次ロールバックを誘引したイベントの時刻印よりも大きい．したがって，補助定理 4 は証明された． □

補助定理 5 $n-1$ 次ロールバック（$n \geq 2$）の ‘−’ ロールバック情報が第 1 レベルプロセッサに届くまで，第 1 レベルプロセッサが決定する GVT が $n-1$ 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることがなければ，この $n-1$ 次ロールバックが引き起こす n 次ロールバックの ‘−’ ロールバック情報が第 1 レベルプロセッサに届くまで，第 1 レベルプロセッサが決定する GVT は n 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることはない．

補助定理証明 5 n 次ロールバックの ‘+’ ロールバック情報が届く前に，そのロールバックの ‘−’ ロールバック情報が第 1 レベルプロセッサの届いたとすると，‘+’ ロールバック情報が届く前であったので，補助定理 4 より，第 1 レベルプロセッサが決定する GVT は n 次ロールバックでロールバックされたイベントの時刻印よりも小さい．

一方，n 次ロールバックの ‘−’ ロールバック情報が，そのロールバックの ‘+’ ロールバック情報の後で第 1 レベルプロセッサに届いたとすると，‘+’ ロールバック情報が第 1 レベルプロセッサに届くまでは，補助定理 4 より，第 1 レベルプロセッサが決定する GVT は n 次ロールバックでロールバックされたイベントの時刻印よりも小さい．そして，‘+’ ロールバック情報が届いた後は，n 次ロールバックでロールバックされたイベントの時刻印が ‘+’ ロールバック情報によって伝えられたため，第 1 レベルプロセッサが決定する GVT が n 次ロールバックでロールバックされたイベントの時刻印よりも大きくなることはない． □

定理 12 あるロールバックの ‘−’ ロールバック情報が第 1 レベルプロセッサに届くまで，第 1 レベルプロセッサが決定する GVT は，そのロールバックでロールバックされたイベントの時刻印よりも大きくなることはない．

証明 12 補助定理 3 と補助定理 5 から帰納的に導かれる． □

以上，定理 11 と定理 12 より，ロールバックが発生する場合においても，第 1 レベルプロセッサはつねに正しい GVT を決定することが示された．

5.2 デッドロック

第 1 レベルプロセッサの決定する GVT が必ず増加し，どのイベントの処理も確定できないデッドロック状態に陥らないことを示す．

定理 13 第 1 レベルプロセッサの決定する GVT は，いつかは必ず増加する．

証明 13 ある時点で，最小の時刻印 TS_{min} を持つ未完了イベント T は，それよりも小さな時刻印を持つ未完了イベントが存在しないので，必ずいつかは処理を完了できる．したがって，そのイベントのプロセッサ N_T の ITA は，必ずいつかは増加する．また，TS_{min} よりも小さな時刻印を持つ未完了イベントが存在しないので，他のプロセッサの ITA は，N_T の ITA よりも大きい．したがって，N_T が受け取るトークンに書かれている他のプロセッサの LVT も，N_T の ITA よりも大きい．よって，イベント T の処理完了時に，N_T の ITA は必ず増加し，変化後の値がトークンに書き込まれる．

また，N_T と同じレベルのリングと上位リングをつなぐプロセッサは，N_T においてイベント T の実行中は，N_T が報告する LVT ($= TS_{min}$) によって，LVT を TS_{min} として上位のリングのトークンに書き込んでいる．また，その上位リングのさらに上位とたどっていくと，第 1 レベルトークンにいたるまで，TS_{min} が LVT として書き込まれている．したがって，イベント T の処理が完了して，N_T が LVT の増加をトークンに書き込むと，その上位レベルのリングのトークンやさらに上位レベルのリングのトークン中の LVT も増加し，第 1 レベルプロセッサにいたるまで，LVT の増加の報告が伝えられる．

よって，第 1 レベルプロセッサが GVT を決定する際に用いた最小の LVT 値は，いつかは必ず増加するため，第 1 レベルプロセッサの決定する GVT はいつかは必ず増加する． □

6. おわりに

本論文では，GVT 決定法として階層マルチリング型アルゴリズムの提案と，その正当性の証明を行った．従来から提案されている Samadi のアルゴリズム，Bellenot のアルゴリズム，Mattern のアルゴリズムと比較すると，Samadi のアルゴリズムは集中型の管理プロセッサの設置を行うために，大規模システムへの適応性が低いのに対し，提案方式は，大規模化により処理すべき量が増加するような管理プロセッサを必要としない．一方，Bellenot のアルゴリズムは，木構造に沿って情報を交換するため，プロセッサ数を N で表すと，処理時間は提案方式と同様に $O(\log N)$ となっているが，GVT 決定に必要となるメッセージ数が提案方式の約 4 倍となっている．また，Mattern のアルゴリズムでは，制御メッセージの巡回経路が，単一のリング状であったために，GVT 決定に要する時間が $O(N)$ であるのに対して，提案方式では，階層化したリング状であるため，$O(\log N)$ にとどまっている．

今後の課題には，実際の分散並列環境上に，今回提案を行った GVT 決定アルゴリズムを用いたタイムワープメカニズムによる分散型離散事象シミュレータを構築し，単位時間あたりのイベント処理能力や，履歴保存用のメモリ使用量などの評価が残されている．

謝辞 本研究の一部は，科学研究費補助金（奨励研究 (A)・課題番号 12780314）の補助を受けている．

参考文献

1) Fujimoto, R.M.: Parallel and Distributed Simulation Systems, John Wiley & Sons (2000).
2) 佐藤 圭，中西 暉，真田英彦，手塚慶一：並列処理型待ち行列網シミュレータ D-SSQ，電子情報通信学会論文誌，Vol.J69-D, No.3, pp.302–311 (1986).
3) 渡辺 尚，中西 暉，真田英彦，手塚慶一：規制先行制御方式を用いた非同期ジョブ並行処理システムの処理能力解析，電子情報通信学会論文誌，Vol.J69-D, No.10, pp.1424–1433 (1986).
4) 鏡味秀行，渡辺 尚，佐藤文明，水野忠則：先行制御方式による並列事象型シミュレータについて，情報処理学会マルチメディア通信と分散処理研究会研究報告，77-12, pp.67–72 (1996).
5) Jefferson, D.R.: Virtual Time, *ACM Trans. Prog. Lang. Syst.*, Vol.7, No.3, pp.404–425 (1985).
6) Fujimoto, R.M.: Optimistic Approach to Parallel Discrete Event Simulation, *Transaction of The Society for Computer Simulation*, Vol.7, No.2, pp.153–190 (1990).
7) Samadi, B.: Distributed simulation, algorithms and perfomance analysis, PhD Thesis, Computer Science Department, University of California, Los Angeles (1985).
8) Bellenot, S.: Global viertyal time algorithms, *Proc. SCS Multiconference on Distributed Simulation*, Vol.22, No.1, pp.122–127 (1990).
9) Mattern, F.: Efficient algorithms for distributed snapshots and global virtual time approzimation, *Journal of Parallel and Distributed Computing*, Vol.18, No.4, pp.423–434 (1993).

(平成 12 年 5 月 30 日受付)
(平成 12 年 9 月 7 日採録)

小林 真也（正会員）

1985 年大阪大学工学部情報工学科卒業．1991 年大阪大学大学院博士課程修了．工学博士．同年金沢大学工学部電気・情報工学科助手．その後，同講師，助教授，同大学大学院自然科学研究科助教授を経て，1999 年愛媛大学工学部情報工学科助教授．その間 1996 年 1〜7 月カリフォルニア大学アーバイン校客員教官．並列処理システム，分散処理システムの研究に従事．電子情報通信学会，日本ソフトウェア科学会，電気学会，IEEE，ACM 各会員．訳書「計算機設計技法」（トッパン），著書「基礎から学ぶ UNIX ワークステーション」（トッパン）．

福田 宗弘（正会員）

1986 年筑波大学情報学類卒業．1988 年筑波大学大学院修士課程理工学研究科了．工学修士．同年日本アイ・ビー・エム（株）東京基礎研究所勤務．1995 年カリフォルニア大学アーバイン校（University of California, Irvine），修士授与．1997 年同大博士課程修了．Ph.D. 同大助手．1998 年筑波大学電子・情報工学系講師．モバイルエージェント，分散シミュレーション，共有メモリマルチプロセッサシステムの研究に従事．