様　式　Ｃ－１９、Ｆ－１９、Ｚ－１９　（共通）

科学研究費助成事業　　研究成果報告書

平成　２８　年　　６　月　２４　日現在

機関番号：　８２６７０
研究種目：　若手研究(B)
研究期間：　2014～2015
課題番号：　２６７３００４４

研究課題名（和文）分散GPGPU環境における離散イベントシミュレーション実行基盤

研究課題名（英文）A Discrete Event Simulation Framework for Distributed GPGPU Environments

研究代表者
大原　衛（Ohara, Mamoru）

地方独立行政法人東京都立産業技術研究センター・開発本部開発第一部・主任研究員

研究者番号：８０４６３００８

交付決定額（研究期間全体）：（直接経費）　　1,900,000円

研究成果の概要（和文）：画像処理用プロセッサを用いて汎用の計算を高速化するGPGPU技術が普及期を迎えている．本課題では，離散イベントシミュレーション（DES）をGPGPU技術を用いて効率的に実行する手法の研究を行った．GPGPUでは処理の分岐が性能の低下を引き起こす．本研究では，時間的な分岐を空間的な冗長に置き換えることで，イベント処理時間を削減する手法を開発した．また，長時間実行されるシミュレーションの信頼性向上のため，ソフトウェア若化法をDESに適用する手法と，これを適用した際の信頼性解析モデルを構築した．

研究成果の概要（英文）：GPGPU techniques, which utilize graphic processors to accelerate general purpose computing, have become popular in the past several years. In this study, we developed a technique for efficiently executing discrete event simulations (DES) using GPGPU. Processing branches lead performance degradation in GPGPU. Thus we reduced the branches by introducing temporal redundancy. Moreover, we developed a number of software rejuvenation schemes for parallel DES. We also constructed an analytical model for evaluating reliability of a simple software rejuvenation technique.

研究分野：　ディペンダブルコンピューティング

キーワード：　GPGPU　離散イベントシミュレーション　タイムワープ法　空間的冗長化　ソフトウェア若化　信頼性解析モデル

１．研究開始当初の背景

コンピュータシミュレーション（以下，シミュレーション）は，未知の物理現象の解明や工業製品の設計など，学術，産業界の様々な分野で広く用いられている．シミュレーションは大別して連続系シミュレーション（Continuous System Simulation; CSS）と離散イベントシミュレーション（Discrete Event Simulation; DES）に分類できる．CSS には天候，災害予測シミュレーションなど，近年その重要性を増した大規模な応用がある．また，DES はネットワークや大規模論理回路（LSI）などの設計に用いられ，産業界における重要性が高い．CSS と DES の双方において，シミュレーションの対象となる問題の規模は拡大し続けており，高速化が強く求められている．

近年，画像処理用プロセッサに内蔵された多数のコアを並列に用いて，汎用の計算を高速化する General Purpose computation on Graphic Processing Units（GPGPU）技術が普及期を迎え，これを用いたシミュレーションの高速化に関する研究が盛んに行われている．とくに CSS において，大きな性能向上が実現された事例が数多く報告されている．一方で，DES に GPGPU を用いた事例は世界的に非常に少ない（学会発表⑦）．DES の GPGPU による高速化が CSS のそれに比較して進展していない理由は，計算モデルの違いにあると考えられる．DES では離散的な時刻に発生する一連のイベントを扱う．イベントには因果関係があるため，時間を分割した並列化は容易でない．さらに，DES で扱われる空間は CSS に比較して稠密でないことが多いため，空間分割による並列化も難しい．

２．研究の目的

本課題の目的は，GPGPU を用いた DES の効果的な並列化手法を構築することにより，CSS が得た成功を DES を多用する産業界にも及ばせることにある．

また，大規模問題のシミュレーションには長大な時間を要する．計算中に一部の GPU が故障したり，ソフトウェアのバグが発現したりして計算結果にエラーが生じる可能性を考慮しなければならない．本課題では，エラーが生じうる環境で複数の GPU が正しく計算を行うための分散アルゴリズムを構築する．

３．研究の方法

以下の二つの項目についてそれぞれ研究を行った．

(1) GPGPU での DES の実行手法

楽観的な並列 DES（Parallel DES; PDES）手法であるタイムワープ法を GPGPU で効率的に実行する手法を検討する．タイムワープ法は，DES の対象とする問題を時間的に分割することによって並列化を行う．タイムワープ法では，イベント間の因果関係をいったん無視して並列実行し，後に因果関係が保たれているかを検証する．結果的に因果関係が保たれていれば，並列実行による性能の向上が期待できる．因果関係に誤りがある場合には，以前の状態を復元して正しい順番でイベントを再実行する．

GPU は SIMD（Single Instruction stream, Multiple Data stream: 単一命令流・複数データ流）型の並列計算環境である．すなわち，複数のプロセッサが異なるデータを対象に同一の命令を実行する．プロセッサごとに異なる命令を実行することはできず，処理の分岐は，すべての分岐を実行した後に無効な結果を破棄することによって実現する．

GPGPU で効率的に PDES を実行するためには，この点を考慮する必要がある．PDES では一般に，プロセッサはそれぞれ異なる種類のイベントを実行する．また，一部のプロセッサで因果関係の誤りが生じた場合，これらは状態回復を行う必要がある一方で，他のプロセッサは通常の処理を継続する．このような処理の多様性が PDES の効率を低下させる可能性がある．

そこで本研究では，各種イベントと状態回復を空間的に冗長化して実行し分岐の影響を軽減する．状態回復には一種の逆計算を用いる．逆計算とは，イベント処理の効果を打ち消すような計算を言う．このようにすることで，通常のイベント処理と同様のメモリアクセスパターンを用いて状態回復を行うことができる．

(2) PDES のソフトウェア高信頼化手法

並列分散システムでは一般に，システムの一部の構成要素の故障によってシステム全体に障害が発生するようなことは避けなければならない．このためには，構成要素の故障を検出し，これを交換したり他の構成要素で代替したりする必要がある．

構成要素の故障要因としては，大きくハードウェアの故障とソフトウェアの故障に大別できる．ソフトウェアの故障とは，いわゆるバグによってソフトウェアが正しい動作を行えないことを言う．　研究開始当初は，主にハードウェアの故障を想定し，この対策法を確立する計画であったが，実際に運用されている並列計算機を調査した結果，ハードウェア故障への対策が既に実装されている環境も少なくないことが分かった．

そこで本研究では，ソフトウェアの故障対策について検討を行った．通常，ソフトウェアのバグの大部分は，開発時に行われる試験によって検出，除去される．しかしながら現実的には，開発時にすべてのバグを除去することは難しく，実行時にバグが発現して異常終了したり，誤った結果が得られたりすることがある．

このような除去の難しいバグへの対策としてソフトウェア若化（software

rejuvenation）法が提案されている．バグの発現を確率的事象と捉えると，稼働時間が長くなるほど発現の可能性が高くなる．ソフトウェア若化法では，定期的にソフトウェアや計算機を再起動させることでバグの発現を防ぐ．従来，ソフトウェア若化法は，長時間動作するインターネットサーバなどの応用で用いられてきたが，近年では高性能計算の応用でもその効果が期待されている．本研究では，ソフトウェア若化をタイムワープ法に適用する方法をいくつか考案し，そのうちのもっとも簡単な手法について信頼性解析モデルの構築を行った．

４．研究成果

(1) GPGPU での PDES の実行

従来のタイムワープ法におけるイベント処理は，典型的には図１のような流れとなる．各プロセスが次のイベントを取り出し，そのイベントの種類によって処理が分岐する．イベント処理が終了すると，その結果に因果律的な矛盾がないかを確認し，矛盾を検出した場合は以前の状態を回復する．シミュレーションモデル中にN個のプロセスがあるとすると，従来法では，一つのプロセスに対して１スレッドを割り当て，合計 N 個のスレッドを用いる．ここで，スレッドとは，GPU における並列実行の最小単位である．

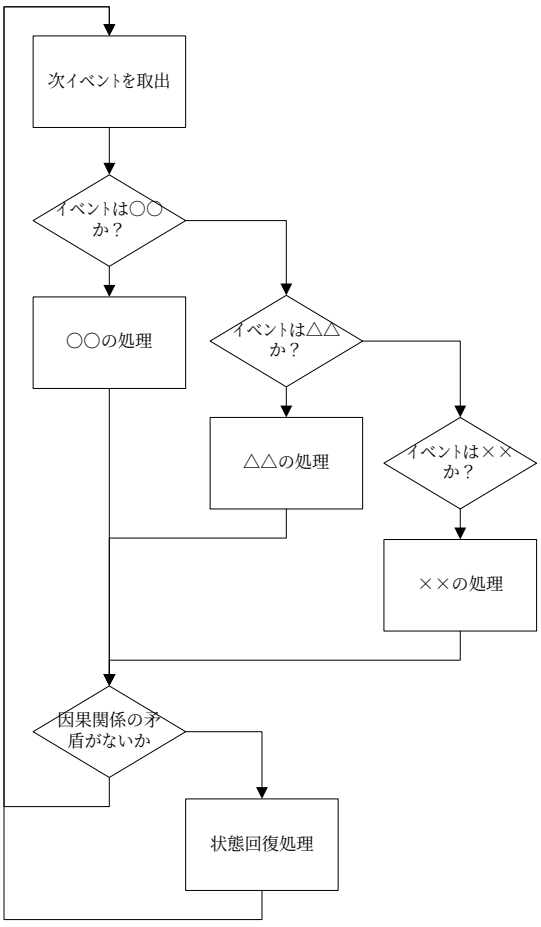

図１ 従来法のフローチャート

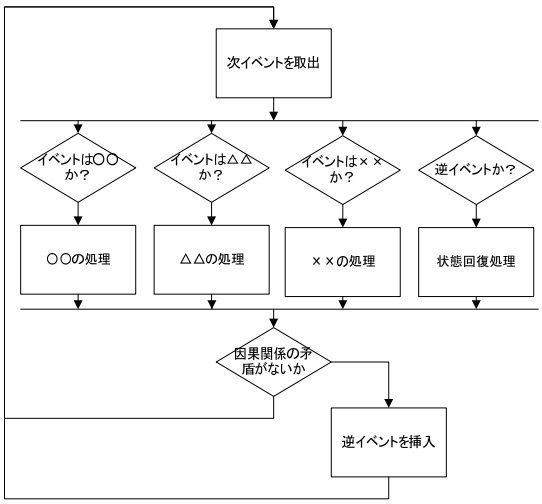

図２ 提案法のフローチャート

これに対して，本研究の提案法では，一つのプロセスに複数のスレッドを割り当てる．図２は提案法における各プロセスのイベント処理フローである．シミュレーションモデルにおけるイベントの種類が M 種であるとすると，逆計算用のスレッドを含めて，合計で $N \times (M+1)$個のスレッドを用いる．一つのプロセスに割り当てられる $M+1$ 個のスレッドのうち，実際に有効なイベントを実行するのは１スレッドだけである．残りのスレッドの処理結果は破棄される．GPU が十分な数のスレッドを同時実行することができれば，この空間的冗長実行によってイベント処理の高性能化が図れる．

NVIDIA 社の Kepler K40 GPU を用いて，4 種のイベントを用いるシミュレーションモデルで 10,000 個のイベント処理に要する時間を計測した．表１に数値例を示す．提案法によって，従来法に比べて最大で約 25%程度の性能向上が図れた．また，プロセス数 N に最適値があることが分かった．これは，プロセス数が少ないときには GPU のスレッド数に対して並列数が十分でないが，プロセス数が多くなると GPU のスレッドを使い切り，性能が低下するためと思われる．一般に，GPU における同時実行スレッド数の最適値は，GPU のハードウェアだけでなく，メモリアクセスパターンにも依存するため自明ではない．本研究ではすべてのイベント種類を並列化する最大の冗長化を行ったが，使用する GPU，シミュレーションモデルによっては，冗長化を制限する方が高い性能が得られる．このような最適化手法の確立は今後の課題である．

表１ 従来法と提案法の処理時間の比較
（単位：ms）

N	100	400	1600	6400	10000
従来法	506	590	707	1,283	1,493
提案法	401	450	524	1,142	1,686
比	0.79	0.76	0.75	0.89	1.09

(2) タイムワープ法へのソフトウェア若化の適用

タイムワープ法では，効率的なシミュレーション実行のためには，各プロセスの処理の進行が大きく異ならないようにすることが重要である．ソフトウェア若化を行う際にも，一部のプロセスの進行が妨げられることは避けなければならない．この目的で，本研究の提案手法では，システム中のすべてのプロセスが同期的に若化を行う（学会発表③—⑥）．

タイムワープ手法では，大域的な時刻であるグローバルバーチャルタイム（Global Virtual Time: GVT）を定期的に求める．このための分散アルゴリズムとして，同期的なものと非同期的なものが提案されている．提案手法では，同期的 GVT 計算アルゴリズムを仮定し，ソフトウェア若化は GVT 計算の直後に行う．

システムは時間 T ごとに GVT 計算を行う．また，k 回の GVT 計算を行うごとに連携チェックポインティングを行う．チェックポインティングが完了すると，すべてのプロセスが再起動し，チェックポイントデータを読み込んで状態を再構築する．その後，すべてのプロセスの状態再構築を待ち合わせて，シミュレーションを再開する．

GVT 計算間隔 T やソフトウェア若化周期 k が大きく，ソフトウェア若化が稀にしか行われない場合には，バグが発現しプロセスに障害が発生する可能性がある．連続する 2 回のソフトウェア若化の間にいずれかのプロセスに障害が発生したとき，障害プロセスはまず再起動を行い，次いで直前のチェックポイントデータを読み込む．タイムワープ手法では，以前の状態に復帰した障害プロセスからの古いタイムスタンプを伴ったメッセージによって，他の正常なプロセスでロールバックが発生することが想定される．これを防ぐために，障害プロセスはチェックポイントデータを読み込んだ後に，他のすべてのプロセスに特別なメッセージを送信する．このメッセージを受信したプロセスは，シミュレーションの実行を中断する．

他のプロセスから受信したすべてのメッセージを処理し状態の回復を完了すると，障害プロセスはこれを他の全プロセスに通知する．PDES の多くの応用では，すべてのプロセスが同じソフトウェアを実行する．このため，一つのプロセスに障害が発生した際には，他のプロセスにおいても障害が発生しやすい状態であることが予想される．本手法では，障害プロセスからの状態回復の通知を受信すると，ソフトウェア若化を行い，すべてのプロセスを再起動する．

提案法を用いたシステムの非可用性は，以下のように表される．ただし，解析モデルで用いる記法を表 2 にまとめた．

$$\bar{A} = \frac{H}{S},$$

$$H = \int_0^{kT} \{t_c + t_r + R(x) + t_c + t_r + t_c\} dF_N + U,$$

$$S = \int_0^{kT} \{x + t_r + t_c + R(x) + t_c + t_r + t_c\} dF_N + U,$$

$$U = \int_{kT}^{\infty} \{t_c + t_r + t_c\} dF_N,$$

$$R(x) = \frac{x}{\alpha + (1-\alpha)\beta}.$$

表 2　記法

A	可用性
S	連続する 2 回の若化の時間間隔
H	S のうち若化と障害によるオーバヘッド
U	S のうち障害が発生しない場合のオーバヘッド
X	直前のソフトウェア若化から障害発生時点までの間隔
$R(X)$	障害で失われたイベントの再実行時間
F	X の確率分布関数
N	プロセス数
F_N	N 個のプロセスに最初に発生する故障の確率分布関数
T	GVT 計算間隔
k	ソフトウェア若化周期
t_c	チェックポイント生成・読込時間
t_r	プロセス再起動時間
α	1 イベント当たりのロールバック発生率
β	1 回のロールバックによって失われる平均イベント数

本研究ではさらに，並列計算機上にシミュレータを試験的に実装し，解析モデルの諸パラメータの計測を行った．チェックポイント生成・読込時間 t_c とソフトウェア若化周期 k の間に明確な相関が見られない一方で，プロセス再起動時間 t_r は k の増加とともに長くなる傾向が観察された．これらの成果について学会発表を行った（学会発表①②）．今後は，実測から得られたパラメータを用いて最適なソフトウェア若化周期を決定する手法の開発に取り組む．

５．主な発表論文等
（研究代表者、研究分担者及び連携研究者には下線）

〔学会発表〕（計　７件）

① S. Fukumoto and M. Ohara, “A Fundamental Study on Software Rejuvenation in Time Warp Simulation,” the 46th Annual IEEE/IFIP International Conference on Dependable Computing and Networks (DSN2016), 30 June 2016, Toulouse (France).
② 大原衛，福本聡，“タイムワープシミュレーションのソフトウェア若化に関する基礎的検討”，電子情報通信学会 DC 研究会，2016 年 5 月 10 日，宇奈月杉乃井ホテル（富山県黒部市）．
③ S. Fukumoto and M. Ohara, “Software Rejuvenation Schemes for Time Warp-based PDES,” the 21st IEEE Pacific Rim International Symposium on Dependable Computing (PRDC2015), 19 Nov 2015, Zhangjajie (China).
④ 大原衛，福本聡，“タイムワープシミュレーションにおけるソフトウェア若化法に関する一考察”，第 73 回 FTC 研究会，2015 年 7 月 17 日，浅虫温泉椿館（青森県青森市）．
⑤ M. Ohara and S. Fukumoto, “A Note on Rejuvenation in Time Warp-based Distributed Systems,” 9th International Conference on Mathematical Methods in Reliability (MMR2015), 3 June 2015, Tokyo (Japan).
⑥ S. Fukumoto and M. Ohara, “Rejuvenation Strategies in Time Warp-Based Distributed Systems,” 電子情報通信学会 R 研究会，2015 年 5 月 22 日，隠岐島文化会館（島根県隠岐の島町）．
⑦ 大原衛，“GPGPU を用いた並列離散イベントシミュレーション研究の動向，” OR 学会信頼性研究会，2015 年 5 月 22 日，隠岐島文化会館（島根県隠岐の島町）．

６．研究組織
(1)研究代表者

大原　衛（OHARA, Mamoru）
東京都立産業技術研究センター・開発本部
開発第一部・主任研究員
研究者番号：80463008

(2)研究分担者
（　　）

研究者番号：

(3)連携研究者

福本　聡（FUKUMOTO, Satoshi）
首都大学東京・システムデザイン研究科・教授
研究者番号：50247590