ヒューマンインタフェースシンポジウム2002論文集, pp. 343-346, 2002

PC クラスタを用いた剛体挙動シミュレーションの並列処理手法の開発とその評価

石井　裕剛[*1]　社領　一将[*1]　遠藤　啓介[*1]　吉川　榮和[*1]

Development and Evaluation of Parallel Processing Method for Rigid Body Simulation using PC Cluster

Hirotake Ishii,[*1] Kazumasa Sharyo,[*1] Keisuke Endou[*1] and Hidekazu Yoshikawa[*1]

Abstract – **To make it easy to construct a virtual environment, it is desirable to simulate a virtual environment based on the physical laws. But the computational cost of the physical simulation is very large, so an expensive super computer is necessary to simulate the virtual environment in real time. In this study, a parallel processing method is developed to simulate a physically based virtual environment on a PC cluster and the performance of the parallel processing method is evaluated by conducting some experiments.**

Keywords : PC Cluster, Physical Simulation, Parallel Processing, Virtual Environment

1. はじめに

実世界と同様な物体操作が可能な仮想空間は、機器の保修訓練や設計・試作、商品のプレゼンテーション等の様々な分野への応用が可能である。仮想空間のシミュレーション手法には、ユーザの動作に応じた仮想物体の動きを予め簡単なルールで定めておく Rule-based Simulation の手法と、各種の物理計算を行うことにより仮想物体の挙動を求める Physically-based Simulation の手法の 2 種類の手法がある。Physically-based Simulation の手法は、Rule-based Simulation の手法と比べ、リアルなシミュレーションが可能である、仮想空間を構築する際の労力が少なくて済む等の利点があるが、シミュレーションする際の計算量が膨大であり、大規模な仮想空間を対象とする場合、単体の計算機でリアルタイム性を確保するためには、高価な計算機が必要である。

一方、パーソナルコンピュータ (PC) の高性能化と低価格化は急速に進行し、大学や企業では 1 人で複数台の PC を保有することも珍しくなくなった。しかし、近年の PC を使った作業は、文章の作成やメールの送受信、ホームページの作成・閲覧が主であり、常に 100%の性能は使用していない。

そこで本研究では、仮想空間を Physically-based Simulation の手法に従ってシミュレーションする際の計算処理を、既設の PC を基に構成した PC クラスタ上で分散して処理できる手法を開発し、そのリアルタイムシミュレーションの実現可能性を検討することを目的とする。

以下、本論文では、剛体挙動のモデル化手法について述べた後、仮想空間のシミュレーションを単体の PC 上で実行する際の計算負荷を計測した結果について述べる。その後、仮想空間のシミュレーションを PC クラスタ上で分散して処理する手法について述べ、実際に各種の PC クラスタ上で実行した結果について述べる。

*1: 京都大学大学院エネルギー科学研究科
*1: Graduate School of Energy Science, Kyoto University

2. 剛体挙動のモデル化手法

本研究で対象とする仮想空間では、主に機器の保修訓練に利用されることを想定し、ユーザの動作に伴って発生する仮想物体同士の衝突時の撃力、接触力（垂直抗力）および重力を実現し、また、仮想物体はその形状が変化しない剛体に限定する。

2.1 剛体挙動シミュレーションの概要

本節では、PC を用いて剛体挙動シミュレーションを実行する際の処理の流れを述べる。まず、シミュレーションを開始する前に、(1) 剛体の初期位置、初期速度等を設定し、(2) ユーザからの動作入力を処理する。その後、(3) 時間幅 t だけ積分を行い、時間 t 後の剛体の位置・姿勢を計算し、(4) その時の剛体同士の接触判定を行う。この際、剛体同士が干渉していると判定されれば、(5) 剛体同士が丁度接触するまでの時間幅Δt を求め、(6) シミュレーションの時間をΔt だけ進める。そして (7) 剛体同士の衝突が起こった時点での接点毎に接触力や撃力等の剛体同士に働く力を計算する。以上の (2)〜(7) までの処理を繰り返し、シミュレーションの時間が時間幅 t だけ経過する度に (8) ユーザに提示する画面を更新する。

2.2 剛体の状態遷移

本研究では、PC を用いて剛体挙動のシミュレーションを実行するにあたり、剛体の状態を自由落下状態、衝突状態、接触状態の 3 つの状態に分けて処理する。

まず、自由落下状態の時には、剛体は重力からの外力を受け、自由落下運動を行う。ユーザが剛体を操作する際も同様に外力として扱う。剛体が衝突状態の時には、他の剛体との接点において撃力が衝突面と垂直方向に働くものとする。剛体が接触状態の時には、他の剛体との接点において互いに接触力が働き、剛体同士が干渉するのを防ぐ。

自由落下状態にある剛体は他の剛体との衝突を条件に衝突状態に遷移する。衝突の際、剛体間の相対速度がある閾値以上である場合、剛体はお互いに跳ね返り、次の瞬間に自由落下状態に戻る。一方、相対速度がある閾値以下である場合には、剛体は接触状態まで遷移し、他の剛体に新たに衝突するまで接触状態を維持する。

2.3 剛体同士の接触判定

安定な剛体挙動シミュレーションを実行するためには、剛体同士の正確な接触判定が必要である。本研究では、仮想空間内に存在する全ての剛体の組み合わせについて、剛体を内包する疑似境界体を用いて粗い接触判定（近似判定）を行った後、ポリゴンレベルでの正確な衝突判定を行う。接触判定アルゴリズムとしては Voronoi Clip [1] を用いる。Voronoi Clip を用いて計算された剛体同士の接点情報は、次節で述べる剛体間の接点に働く力の計算で利用する。

2.4 剛体間の接点に働く力の計算

本研究では、剛体間の接点に働く力の計算に Baraff らの手法 [2] を用いる。すなわち、接点 $i(i = 1, \cdots, n)$ において仮想物体に働く撃力 p_i と衝突前の相対法線速度 v_i、衝突後の相対法線速度 v_i' の関係を定式化すると (1) 式の様になる。

$$\begin{aligned} &\mathbf{V}' = \mathbf{AP} + \mathbf{V} \\ &v_i' \geqq -\epsilon_i v_i \\ &p_i \geqq 0 \qquad\qquad (1) \\ &p_i(v_i' + \epsilon_i v_i) = 0 \end{aligned}$$

ただし、$\mathbf{V}' = (v_1', \ldots, v_n')^T$、$\mathbf{V} = (v_1, \ldots, v_n)^T$、$\mathbf{P} = (p_1, \ldots, p_n)^T$、$\epsilon_i$は接点 i での跳ね返り定数、$\mathbf{A}$ は剛体の質量と姿勢および接点の位置で定まる定数である。接点に働く力を求める問題は (1) 式の条件下で実行可能な解 $\mathbf{V}'$および $\mathbf{P}$ を求める問題となる。そしてこの問題は、線形計画法における線形相補性問題となり、Dantzig のアルゴリズム [3] を用いて解くことができる。

3. 単体の PC を用いた剛体挙動シミュレーション

3.1 シミュレーションの並列化へ向けての検討

本研究では、剛体挙動シミュレーションの処理を PC クラスタを構成する計算用ノードマシンに分散することにより、全体としてのシミュレーション速度の向上を図ることを目的としている。しかし、並列化が可能と考えられる全ての処理を分散・並列処理した場合、PC 間の通信量が膨大となることが予想される。そこで、少ない通信量で高い並列効果を得るために、単体の PC で剛体挙動のシミュレーションを実行し、その時の各処理の負荷を調べ、負荷の高い処理を並列処理化することにした。

3.2 単体の PC を用いたシミュレーションシステムの開発

単体の PC を用いたシミュレーションシステムは、PC/AT 互換機上（OS は Linux）に OpenGL ライブラリと前述の Voronoi Clip ライブラリを用いて C++ 言語で開発した。ユーザへの映像出力デバイスは CRT ディスプレイを用い、動作入力デバイスとしては Logitech 社の３次元マウス Magellan を用いた。ユーザは３次元マウスを操作することにより、仮想空間内に配置した剛体を操作できる。

3.3 シミュレーションの負荷の計測実験

シミュレーションの負荷の計測は、仮想空間内で球形状の剛体がお互いに衝突を繰り返すシミュレーションを実行する際の負荷を計測することにより行った。具体的には、跳ね返り係数 1.0、大きさが 400cm 四方の直方体の内部に、半径 10cm、跳ね返り係数 1.0、224 ポリゴンで構成される均一な質量分布を持つ球形状の剛体を、x,y,z 軸方向に３個ずつ 100cm の間隔で配置し、時間積分幅 0.01 秒で 5000 フレーム間計測し、各処理が全体の処理に対して占める負荷の割合の平均を求めた。シミュレーション中には剛体を操作せず、剛体は外力として重力のみの影響を受ける。この実験では PentiumIII 1.26MHz(Dual)、メモリ 768MB の PC を使用した。

実験の結果を表 1に示す。実験の結果から剛体挙動シミュレーションの各処理の内、ポリゴンレベルの衝突判定と、Dantzig のアルゴリズムを用いた力の計算が、他の処理に比べて負荷が高いことが分かる。従って、本研究では、これら２つの処理を並列化する方向で PC クラスタによるシミュレーションの並列化を行うことにした。

表 1　各処理の計算負荷の比較
Table 1　The Comparison of the Computational Cost of Each Task

処理の種類	全体に対する割合 (%)
ポリゴンレベルの衝突判定	41.48
Dantzig のアルゴリズムの処理	37.35
剛体間の連結関係の抽出	16.27
時間積分	4.88
球を用いた近似判定	0.02

4. PC クラスタを用いた剛体挙動シミュレーション

4.1 シミュレーションの並列化手法

4.1.1 並列化の基本方針

本研究では、PC クラスタを構成する環境として、大学の研究室のような複数台の PC が既にネットワーク接続されている環境を想定しているため、各計算用ノー

ドマシンの計算能力は均一ではない。また、剛体挙動シミュレーション以外のプロセスが計算用ノードマシンを利用するため、シミュレーションを実行中に見かけの性能が変化することも考慮しなければならない。しかし、分散・並列化の効果を十分に発揮するためには、各計算用ノードマシンにおける処理の終了時間をできる限り均一にする必要がある。従って、各計算用ノードマシンに均等に計算処理を割り当てるのではなく、シミュレーションを実行しながら、計算用ノードマシンの見かけの性能を計測し、計算処理の割り当てを動的に変化させる仕組みが必要となる。

4.1.2 並列化手法の詳細

図1に、剛体挙動シミュレーションを分散・並列化する場合の処理の流れを示す。2.1 節で述べた単体の PC 上でシミュレーションを実行する場合との違いは、(4) の厳密な接触判定と (7) の接触力や撃力の計算処理の直前に、処理を分散させるためのスケジューリング処理を加え、(4) と (7) の処理を PC クラスタを構成する計算用ノードマシンに分散して処理する点である。

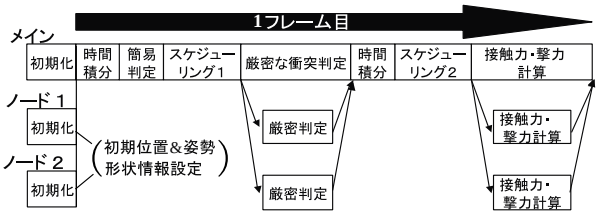

図1 並列処理する場合の処理の流れ
Fig. 1 Flow of the Parallel Processing.

スケジューリング1では、ポリゴンレベルの衝突判定の処理を各計算用ノードマシンへ割り当てる処理を行う。本研究では、ポリゴンレベルの衝突判定は、衝突を判定する2つの剛体を1つの組として、組単位で分散・並列化する。そして、ポリゴンレベルの衝突判定の計算量は、2つの剛体のポリゴン数の積に比例すると仮定する。その後、以下の手順に従って、衝突判定の処理を各計算用ノードマシンに割り当てる。

(1) 処理を負荷の高い順番に並び替える。
(2) 最も負荷の高い処理を、各計算用ノードマシンに割り当てたと仮定したとき、その処理が終了するまでに要する時間を各計算用ノードマシン毎に計算する。
(3) 既に割り当て済みの処理を終了するまでに要する時間と (2) で求めた時間の和が最も小さくなる計算用ノードマシンに、対象の処理を割り当てる。
(4) (2) と (3) を、全ての処理を計算用ノードマシンに割り当てるまで繰り返す。

上記アルゴリズムは、アルゴリズム自体の計算負荷が小さいという利点があるが、必ずしも、全ての計算用ノードマシンの終了時間が均一になるように、最適に処理を割り当てるとは限らない。今後改良する予定である。

一方、スケジューリング2では、接点に働く力の計算処理を各計算用ノードマシンへ割り当てる処理を行う。本研究では、接点に働く力の計算処理は、同時刻に互いに接触している全ての剛体を1つの組として、組単位で分散・並列化する。Dantzig のアルゴリズムを用いて線形相補正問題を解く場合、条件によって解を得るまでの計算量が異なるため、予め正確に計算量を推定することは困難であるが、本研究では、Dantzig のアルゴリズムを用いて線形相補正問題を解く場合の計算量を、接点の数の3乗に比例すると仮定する。接点に働く力の計算処理に必要な計算量の推定結果を基に、各計算用ノードマシンに処理を割り当てる方法は、ポリゴンレベルの衝突判定の処理を割り当てる方法と同じである。

なお、各計算用ノードマシンの性能は、直前に割り当てた計算処理の応答時間を基に推定する。

4.2 シミュレーションシステムの開発

図2に、本研究で開発した PC クラスタを用いた剛体挙動シミュレーションシステムのシステム構成を示す。

まず、メインマシンにのみ実装される制御部は、計算処理を各計算ノードに割り当てるスケジューリング部を持ち、システム全体を制御する役割を担う。計算処理の割り当てを決定すると、メインマシンの通信部は計算用ノードマシンに計算の内容を送信する。各ノードでの計算結果は再びメインマシンの通信部を介して集約される。また、シミュレーションがある一定時間経過する毎にメインマシンの通信部を介して仮想空間の描画情報が描画用ノードマシンに送られ、ユーザに提示される。メインマシンと計算用ノードマシンは PC/AT 互換機 (OS は Linux) を用い、描画用ノードマシンは SGI 社の OCTANE(OS は IRIX) を用いた。また、メインマシンと計算用ノードマシンの通信部は高速通信ライブラリ PMv2 [4] を用いて C++言語で開発し、描画用ノードマ

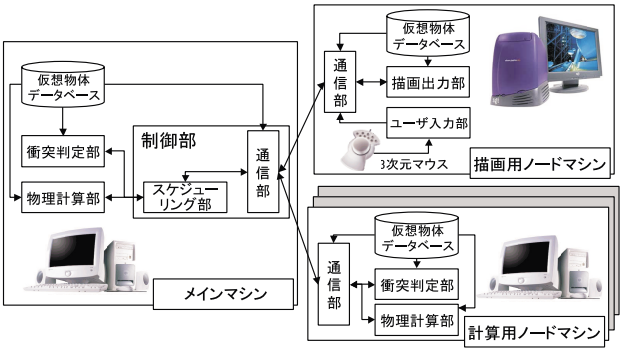

図2 PC クラスタを用いたシミュレーションシステムの構成
Fig. 2 Configuration of the Simulation System using PC Cluster.

表2　実験で用いた計算機環境（全て Dual CPU）
Table 2 Computer Environment used at the Experiment（All Machines have Dual CPU）

条件１	４台の PC(1.26GHz,1.0GHz ２台,700MHz) を 100BASE-T で接続
条件２	４台の PC(1.26GHz,1.0GHz ２台,700MHz) を 10BASE-T で接続
条件３	２台の PC(1.26GHz,1.0GHz) を 10BASE-T で接続
条件４	単体の PC(1.26GHz)

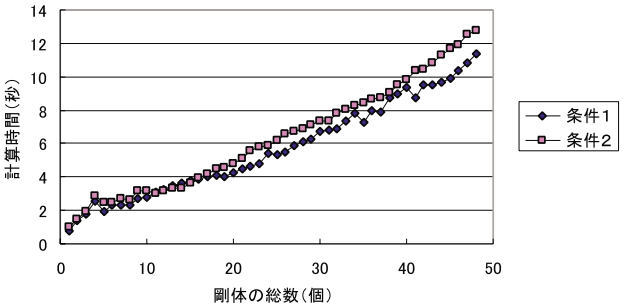

図３　計算時間の比較（ネットワークの速度）
Fig. 3 Comparison of the Computational Time(Network Speed).

シンでユーザに映像を提示する描画出力部は OpenGL ライブラリを用いて C 言語で開発した。

4.3　シミュレーションシステムの評価実験

開発したシミュレーションシステムの性能を評価するために、表 2に示す 4 種類の計算機環境を用いて、複数の剛体が自由落下して床や他の剛体との衝突を繰り返し、全ての剛体が静止するまでのシミュレーションを実行するのに要した時間を計測した。実験に用いた剛体は、床 (12 ポリゴン、跳ね返り係数 0.8)、ボルト (178 ポリゴン、跳ね返り係数 0.8)、ナット (396 ポリゴン、跳ね返り係数 0.8) の 3 種類であり、仮想空間内に配置する剛体の数を 1 個から 50 個まで変化させた全ての場合について計測した。なお、剛体の初期位置と姿勢はランダムに設定し、１フレームの時間幅は 0.1 秒とした。

条件１と条件２での実行結果を図 3に、条件２から条件４での実行結果を図 4に示す。また、全てのフレームにおいてリアルタイムに剛体挙動シミュレーションを実行できたのは、条件 1 の環境で剛体の総数が 21 個まで、条件 4 の環境で剛体の総数が 9 個までであった。

図 3より、自由落下する剛体の総数が 20 個程度までの場合では、ネットワークが 10BASE-T の場合と 100BASE-T の場合でシミュレーション速度に大きな差が無いことが分かる。また、図 4より、PC クラスタを構成する PC の台数が増えるに従って、計算速度が向上していることが分かる。特に、50 個程度の剛体が自由落下する仮想空間をシミュレーションする場合には、

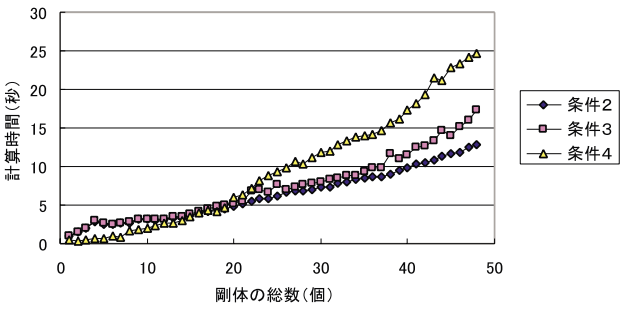

図４　計算時間の比較（マシンの台数）
Fig. 4 Comparison of the Computational Time(Number of Machines).

PC を 4 台接続した環境で、単体の PC を用いる場合に比べ、約 130%の高速化が実現できていることが分かる。また、図 4のグラフにおいて、剛体の総数が 1 個から 15 個までの間で、PC クラスタを用いる場合よりも単体の PC を用いる場合の方が計算時間が短いが、これは、スケジューリングに必要な計算時間が、並列化による計算時間の短縮より大きい為であると考えられる。

以上より、本研究で考案・実装した並列処理手法を用いることにより、剛体の総数が多い場合に、剛体挙動シミュレーションの実行速度を向上させることができ、より複雑な剛体挙動シミュレーションをリアルタイムに実行できるようになることを確認できた。

5.　おわりに

本研究では、剛体挙動シミュレーションを PC クラスタ上で並列処理する手法を開発し、各種の PC クラスタ環境でシミュレーションを実行することにより、性能評価を行った。その結果、特別なハードウエアを使わない既存のネットワーク環境でも、剛体挙動シミュレーションを高速化できることを示した。今回の実験は、PC クラスタの計算用ノードマシンが他の作業に使用されていない状態で行ったが、今後は、他の作業のバックグラウンドで並列処理を行う仕組みを開発し、リアルタイムシミュレーションの可能性を検討したい。

参考文献

[1] Mirtich, B.: V-Clip: Fast and Robust Polyhedral Collision Detection, ACM Transactions on Graphics, Vol. 17, No. 3, pp. 177-208, (1998).
[2] David Baraff: Fast Contact Force Computation for Nonpenetrating Rigid Bodies, SIGGRAPH '94 Computer Graphics Proceedings Annual Conference Series 1994, pp. 23-34, (1994).
[3] Cottle, R. and Dantzig, G.: Complementary pivot theory of mathematical programming, Linear Algebra and its Applications, No. 1, pp. 103-125, (1968).
[4] 住元 真司他: 高速通信機構 PM2 の設計と評価, 情報処理学会論文誌, Vol. 41, No. SIG05, pp. 80-90, (2000).