コンポーネント指向可視化対話シミュレータの分散実行

Distributed Execution of Components-Oriented Visual/Interactive Numerical Simulation System

澤嶋友輔*, 梅谷征雄†
Yusuke SAWASHIMA and Yukio UMETANI

概要

Demands for high performance numerical simulator is being increased to cope with the vast amount of computations needed for the three dimensional simulation. We enabled the distributed execution of our visual/interactive piano simulator by applying Microsofts COM and DCOM technology to the execution of each piano component(string,bridge,sound board).Experiments showed some feasibility in case for the distribution of sound board component.
Key Words:Problem Solving Environment,Component,COM,DCOM

1 はじめに

現在コンピュータによる物理現象・工学現象のシミュレーションは、実験・理論に次ぐ第３の手法として活用されている。シミュレーションは実験で観測の難しい量の情報を得るためには有効な手法である。今回の研究では、工学現象のシミュレーションとしてピアノの物理モデルの動きを観察するピアノシミュレータにおける分散化を計ることにする。現在の楽器シミュレーションの世界ではモデルの詳細化・忠実化といった流れに基づいてモデルの高次元化といったものが求められるようになってきている。そうなると、一つのマシン上でシミュレーションを行う仕組みではマシンにかかる負荷が大きく、十分な応答性を確保できないと考えられる。

既存のシミュレータとしてコンポーネント指向可視化対話シミュレータというものがある。そこに実装されたピアノシミュレータでは弦の振動が駒に伝わり、駒の振動が響板に伝わるという仕組みをとっている。そこでは、弦・駒・響板のそれぞれの物理計算の部分を独立したコンポーネントとして用意して、実際にシミュレーションを行う際にそれらのコンポーネントを配置、接続してシミュレーションを行えるようになっている。

上記のコンポーネントは Windows の COM(Component Object Model)[1] を用いて実装されているが、現状ではすべてのコンポーネントが DLL(Dynamic Link Library) として、クライアントプロセス上にロードされて動作するようになっているため、先に述べたモデルの高次元化に耐えられるような仕組みになっていない。そこでこの研究では弦・駒・響板といったコンポーネントを別のプロセスまたは、別のリモートマシン上で動かすことによって、クライアントの処理の負荷を軽減させるということを目的とする。つまり、この分散化の研究によって将来の高次元モデルの楽器シミュレーションを行う上で、スマートに実験を行えるような環境作りを狙う。

2 コンポーネント指向可視化対話シミュレータ

先に述べたコンポーネント指向可視化対話シミュレータは物理コンポーネントを増やすたびにシミュレータのコードを大幅に変更しないで済むような工夫をしてある。ここでいう物理コンポーネントとは物理現象を決定するパラメータ値の変更、物理現象を解析するための計算、その計算結果の可視化、データ入出力のそれぞれの機能を持っているコンポーネントのことである。

ピアノシミュレータで物理コンポーネントは、「String」、「Bridge」、「Board」の３つとしてある。「String」

* 静岡大学 情報学研究科情報学専攻　所属（〒432-8011 浜松市城北 3-5-1,cs8040@cs.inf.shizuoka.ac.jp）
† 工博　静岡大学教授　情報学部情報科学科（〒432-8011 浜松市城北 3-5-1,umetani@cs.inf.shizuoka.ac.jp）

コンポーネントクラスはピアノの物理モデルの弦とハンマーの部分、「Bridge」コンポーネントクラスは駒の部分、「Board」コンポーネントクラスは響板の部分をコンポーネント化したものである。これらの物理コンポーネントはそれぞれ、Create メソッドによりクライアントのエージェント部分がそれぞれのクラスのオブジェクトを作成する。このメソッドが呼ばれると新しい String,Bridge,Board オブジェクトのポインタが返される。それによって、その他の virtual キーワードがつけられた各インターフェースクラスの純粋仮想関数を宣言したメソッドを呼び出すことが出来る。

2.1 シミュレータ構成

図２．１にコンポーネント指向可視化対話シミュレータの構成を示す。

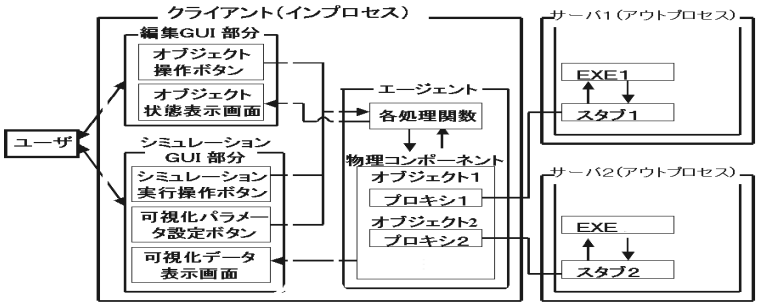

図２．１　コンポーネント指向可視化対話シミュレータ構成
Fig2.1 Structure of Components Oriented for Visual/Interactive Numerical Simulation System

ここでのサーバ１、サーバ２が弦・駒・響板の物理コンポーネントを実行する。

2.2 シミュレータ画面

図２．２、図２．３に、ユーザとシミュレータとのインターフェースにあたるシミュレータの画面を示す。

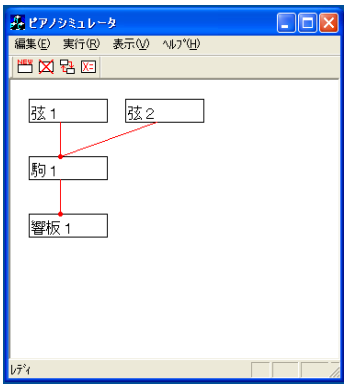

図２．２シミュレータ・オブジェクト編集ウィンドウ
Fig2.2 Simulator Object Edit Window

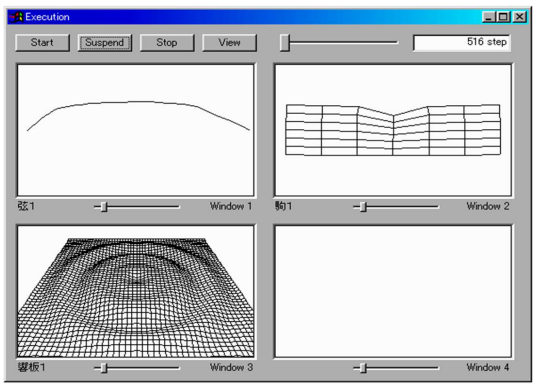

図２．３シミュレーション実行ウィンドウ
Fig2.3 Simulation Execute Window

図２．２は物理コンポーネントのオブジェクトの作成・削除・パラメータの変更・コネクションを行うウィンドウである。図２．２では、オブジェクトとして「弦１」「弦２」「駒１」「響板１」が作成されていて、「弦１」と「駒１」、「弦２」と「駒１」、「駒１」と「響板１」、がデータのやりとりが出来るようにつなげられている、という状態を表している。

図２．３は図２．２のシミュレータ・オブジェクト編集ウィンドウで作成されたシミュレーションの実行を表示・制御するためのウインドウである。中央の４つのウィンドウがそれぞれオブジェクトの可視化データを表示する。画面上方の４つのボタンは左から、シミュレーションを開始するための Start ボタン、シミュレータを一時停止するための Suspend ボタン、シミュレータを停止するための Stop ボタン、オブジェクトをウィン

ドウに表示させるための View ボタン、となっている。またその右のスライドバーはシミュレーション速度を調節するためのバーで、さらにその右のウィンドウは時間ステップのカウンターを表示している。

3 コンポーネント指向可視化対話シミュレータの分散化

3.1 手順

既存のコンポーネント指向可視化対話シミュレータが COM-DLL によって実装されているために、オブジェクトがクライアントのプロセス内でしか扱えない仕組みになっている。

そこで、コンポーネントをアウトプロセスで動作できるようにするため今回は、オブジェクトを同一マシン上の別プロセスで動かすことと、オブジェクトをリモートマシン上に配置して、完全にクライアントのマシンから独立させて動作させることにする。技術的には Microsoft の ATL-COM を使って既存のシミュレータが作成されているので、同じく Microsoft の DCOM という COM のネットワーク分散の技術を使用して、それを実現する。その際、ローカルでのアウトプロセス化のためには

(1)COM-DLL 化されたオブジェクトサーバを EXE のサーバに変更する
(2) プロキシ・スタブファイルを作成し通信手段をとる
(3) データのマーシャリングが行えるようにインタフェースを定義して通信する

リモートマシン上でのアウトプロセス化のためには

(4) アウトプロセスサーバで作成した EXE サーバとプロキシ・スタブファイルをリモートサーバへ移動する
(5) リモート、クライアント上でサーバを登録する。RPC が行えるように、DCOMCNFG ツールで設定する
(6)DCOM の通信方法で通信する

という手順を踏む。

3.2 方法

クライアントのエージェントの部分がオブジェクトを管理するところまでは COM-DLL と変わりは無いが、今まではオブジェクトをそのままクライアント領域で扱っていたのに対し、エージェントが、EXE サーバ下に置かれたオブジェクトとプロキシとスタブを通してデータの受け渡しを行うことになる。

ここで、コンポーネントを EXE サーバ下に置くと、コンポーネントとクライアントが同じアドレス空間を共有しないことになる。クライアントは、インターフェイスを介してコンポーネントと通信する。クライアントに渡されるインターフェイスは、基本的に関数へのポインタの配列なので、クライアントはインターフェイスに対応するメモリにアクセスできなければならない。既存のシミュレータのようにコンポーネントが DLL にある場合は、コンポーネントとクライアントが同じアドレス空間を共有しているから、クライアントはメモリに簡単にアクセスできる。しかし、クライアントとコンポーネントが別のアドレス空間にある今回のモデルでは、クライアントがコンポーネントのプロセスのメモリにアクセスできないことになる。よってインターフェイスの関数が呼び出せなくなってしまう。この問題を解決するために次の条件を満たすようにする。

(1) プロセスが別のプロセス内の関数を呼び出せるようにする
(2) プロセスが別のプロセスにデータを渡せるようにする
(3) クライアントが、アクセス先がインプロセスコンポーネントであるかアウトプロセスコンポーネントであるかを意識しなくて済むようにする

(1) を実現するために、同一マシン上の別プロセス分散ではローカルプロシージャ呼び出し（LPC) を、リモート分散ではリモートプロシージャ呼び出し（RPC) を行う。クライアントからプロキシとスタブを通してこれらのコールを行うことによって、プロセスの境界を越えて関数の呼び出しを実現する。

(2) を実現するためには、クライアントのアドレス空間からコンポーネントのアドレス空間に関数のパラ

メータを渡さなければならない。この作業をマーシャリングという。他方のプロセスが同じマシン上にある場合は一方のプロセスのデータを他方のプロセスのアドレス空間にコピーすればよいだけであったのだが、両方のプロセスが別のマシン上にあるリモートサーバの場合、マシン間の相違をなくすためにデータを標準の形式に変換する必要がある。このために COM では、コンポーネントの作成時に IMarshal というマーシャリングを行うインタフェースが要求される仕組みになっていて、IMarshal のメンバ関数を呼び出して、関数呼び出しの前にパラメータをマーシャリングし、呼び出しのあと元に戻すという作業を行ってくれるので特別に意識することなく実現できる。

最後に (3) を実現するために、クライアントが一旦コンポーネントの DLL にアクセスして、ローカル、リモートの場合は、その DLL から EXE にアクセスするという構造になっていれば、問題ない。ローカル、リモートサーバを立ち上げるためにプロキシ DLL を用意して、それらを立ち上げるようにクライアントから命令がきた場合は、プロキシ DLL を介して EXE のサーバにアクセスし、今までどおりインプロセスの場合はそのまま DLL のサーバにアクセスするようにする。

以上に述べたことは COM/DCOM の標準の機能を使う事により実現できる。

3.3 ローカル分散の実装

・手順の (1) に述べた COM サーバの作成には、Microsoft 社の VisualC++ [3] [4] の ATL [1]Wizard（コード自動生成機能）を用いる。

・プロキシ・スタブファイルの作成（手順の (2)）

Wizard により COM のサーバを作る途上インタフェースのファイルとして作成される IDL ファイル（クライアントからのアクセスとしてのタイプライブラリファイルを生成する）を、MIDL コンパイラという Microsoft の IDL コンパイラに通すことによって標準のプロキシ・スタブファイルが生成される。このファイルには作成するサーバの一意なクラス ID の情報が含まれているので、EXE のサーバを登録する場合、そのサーバへのアクセスができる仕組みである。

・マーシャリング（手順の (3)）

インタフェースのファイルに記述されている仕様によりサーバの関数を呼び出す際に値の受け渡しなどでは IDL ファイルに記述されている関数に (in) と (out) のパラメータ属性をつけて記述することによりプロキシとスタブをさらに最適化することができる。

・クライアントの変更

クライアント上インプロセスで COM を呼び出すように記述されていた部分をローカルのサーバを呼び出すように変更する。これは引数のところの指定を変えるだけで済む。

3.4 リモート分散の実装

・DCOMCNFG ツールによる設定（手順の (5),(6)）

リモート分散を行うには、ローカルで動かす場合にマシン間を超えてアクセスするという指定を DCOMCNFG という Windows2000、または WindowsNT に標準で組み込まれているツールを用いて行えば良い。ここでリモートで動いていることを確認するために、リモートマシン上でタスクマネージャを開いて COM サーバが EXE のプロセスとして立ち上がっていること、またシミュレーション中に動作していることを確認する

4 性能計測

4.1 環境

- クライアントマシン
  - デスクトップ PC
  - IBM APTIVA
  - CPU：K-6　400MHZ

- リモートサーバマシン
  - デスクトップ PC
  - DELL
  - CPU：Pentium4　1700MHZ

以上の環境のもとで負荷分散の効果を計測した。両者は 100BASE-T　LAN で接続した。通信プロトコルとして TCP/IP、通信アプリケーションとして Microsoft ネットワークを用いた。

今回は弦、駒、響板を各々１つ組み合わせて実験を行った。また、弦の計算→駒の計算→響板の計算→それぞれの通信という過程を 1step として、2000step 実行した場合の時間 (Sec) を計測した。

4.2 分散形態

３つの異なる状態で実測を行った。

- インプロセス通信
- ローカルアウトプロセス通信
- リモートアウトプロセス通信

インプロセス通信ではオブジェクトも DLL 化されており、クライアントのプロセス内で通信を行う為にクライアントマシンのみを使って実測を行った。

ローカルアウトプロセス通信ではオブジェクトが EXE サーバの管理下にある。こちらもクライアントマシンのみを使って実測を行った。

リモートアウトプロセス通信ではオブジェクトの EXE サーバをリモートサーバマシン上に配置して通信を行いながら実測を行った。

図４　シミュレーションプロセス (1step)
Fig4 Simulation Process (1step)

4.3 計測結果

結果を表４．１に示す。

表４．１:それぞれの COM 分散形態における実行時間の計測 (2000steps)
Table4.1: Measure Execution Time for Each Destributed COM Form(2000steps)

形態	時間 (sec)
全てのオブジェクトをインプロセス	13.890
弦オブジェクトのみローカル	20.487
駒オブジェクトのみローカル	23.124
響板オブジェクトのみローカル	23.259
弦オブジェクトのみリモート	21.013
駒オブジェクトのみリモート	22.465
響板オブジェクトのみリモート	14.952
全てのオブジェクトをローカル	38.890
全てのオブジェクトをリモート	31.232

この結果により、一番分散化のメリットの感じられるのは響板オブジェクトのみをリモートサーバマシンに持っていった場合で、既存のインプロセスシミュレータに近いパフォーマンスを見せている。これは響板オブジェクトで行われる計算が他のオブジェクトに比べて一番負荷がかかる処理となっているために、分散化のオーバーヘッドを相殺するリモートマシン上での処理時間の短縮があったためと推測される。

4.4 響板の分割数変化

前回の計測で響板をリモート分散させた場合が一番効率が上がることがわかった。それで次に響板の分割数を増やし、計算負荷を上げてみた。結果を表４．２に示す。

表４．２:響板の分割数を変化させた場合における実行時間の計測 (2000steps)
Table4.2: Measure Execution Time in Case of changed Board Division(2000steps)

響板分割数	インプロセスによる時間 (sec)	響板リモート分散による時間 (sec)
５０×５０	13.890	14.952
６０×６０	15.973	14.962
７０×７０	21.961	16.633

分割数を増やすにしたがい、分散化の効果が大きくなることが確認できた。

5 終りに

以上の実装によりピアノシミュレータが分散環境で動くようになり、計測により負荷分散の効果を検証できた。今回の結果としては響板オブジェクトをリモート分散させた場合に一番効果があった。そして響板の分割数を上げていき、負荷を増大すると更なる改善が見られた。このことより、実響板のモデル等でのシミュレーションにおける負荷分散の目処が立ってきた。しかし、現段階では決定的なアドバンテージが得られるような性能向上は認められていない。各々のコンポーネントのスレッド化処理などにおいて更なる性能向上を求めるのが今後の課題である。

謝辞

本研究を行うに当たり、日頃親切に御協力頂いた梅谷研究室の皆様に厚く御礼を申し上げます。

参考文献

[1] Richard Grimes,Alex Stockton,GeorgeReily,Julian Templeman 著　田中正造　訳: ”ATL COM プログラミング”, 翔泳社,1998
[2] Dale Rogerson 著　バウン　グローバル株式会社訳: ”Inside COM”, 株式会社アスキー,1997
[3] 山本信雄　著：”Visual C++ 1,3”, 翔泳社,2000
[4] 林晴比古　著：”新 Visual C++6.0 入門”, ソフトバンク,1998
[5] Chris corry,Vincent Mayfield,John Cadman,Randy Morin 著　株式会社　日本ドキュメンテックス訳: ”COM/DCOM 実践プログラミング”, アスキー出版局,1999
[6] http://www.users.gr.jp/ml/archive/com/:COM Mailing List Log
[7] http://www.asia.microsoft.com/japan/developer/thisWeek/combasics/combasics1.asp :COM プログラミングの基本上
[8] http://www.asia.microsoft.com/japan/developer/thisWeek/combasics/combasics2.asp :COM プログラミングの基本中
[9] http://www.asia.microsoft.com/japan/developer/thisWeek/combasics/combasics3.asp :COM プログラミングの基本下