マルチメディア通信と分散処理　94－11
分散システム/インターネット運用技術　14－11
（１９９９．７．１５）

仮想空間の空間分割型分散管理における管理領域の動的変更に関する一考察

清水　孝一　　山本　潮　　小野里　好邦

群馬大学大学院工学研究科

〒 376-8515 群馬県桐生市天神町 1-5-1

あらまし　近年のインターネットの普及により，仮想空間をネットワーク上に展開することによって，人々のコミュニケーションの場として様々なサービスに発展してきている．しかし，参加者の増加に伴いサーバに対する負荷が懸念されており，多くのユーザをサポートするための研究が盛んに行なわれている．そこで，多くのユーザを仮想空間内に投影させる方式の一つとして，サーバを複数用意し，一つのサーバが管理する領域を分割し，多くのユーザをサポートすることが可能な方式が考えられ実装されている．しかし，任意の管理領域にクライアントが集中した場合，この空間分割方式では効果をあげることができない．最悪の場合，一つの空間に全てのクライアントが集中してしまう恐れもある．本稿では，この分割管理されているサーバ管理領域を動的に変化させ，サーバの負荷を分散させるための方式を提案する．また，本方式を用いてプロトタイプを作成したので，従来の方式と比較し考察する．

キーワード　仮想空間，空間分割型管理方式，動的な領域変更，負荷分散

A Study on Floating Method of Regions in Virtual Environment

Kouichi Shimizu, Ushio Yamamoto, and Yoshikuni Onozato

Graduate School of Engineering, Gunma University

Abstract　Recently, according to increasing Internet users, various services have been developed. One of such services is providing virtual environments where the users communicate with each other. However, by the number of users increases, much reearch effort has been devoted to support large number of users. One of such methods is that a virtual space is devided into some blocks and each block is managed by each server. However, each server manages each fixed virtual space and take care of users in its managing region, and such approach is not effective, in case that many users concentrate in some part of virtual space. In the worst case, all of the users concentrates in one block of virtual space. In this paper, we propose the multi server-managed virtual space model in which the load of managing virtual space is well-balanced to servers by dynamically changing managed regions. We demonstrate a prototype system, as a result of comparing our prototype system with multiple server system without dynamic change of managed system.

keywords　Virtual Space, Space Division Management, Dynamically Changing Managed Region, Load Balancing

1 はじめに

インターネットを使用した仮想空間サービスが頻繁に提供されるようになり，ユーザがアバタ(AVATAR)と呼ばれる分身として仮想空間内に投影され，多くの人が同時にコミュニケーションを行なうことができるようになった．しかし，仮想空間を管理しているサーバの処理できる許容人数を超えるユーザがそのサービスを利用する傾向にある．そのため，より多くのユーザを投影させることができる仮想空間を構築するための研究が行なわれている．その一つとして，一つのサーバで全体空間の処理を行なうのではなく空間を分割し，それぞれを用意した複数のサーバで管理する手法が存在する [1]．この手法を用いることによって全てのサーバの管理領域に対して均等にクライアントが分配されるような状況において，全体としてより多くのクライアントをサポートすることができる．しかし，常にそのような効果が期待できるわけではなく，最悪の場合では一つのサーバ領域に全てのクライアントが集中することもあり得る．本稿では，仮想空間の管理領域に着目し，全体としての仮想空間を複数のサーバで管理し，それぞれのサーバに対する負荷を効果的に分散させるようにすることを目的として，クライアントが一部の管理領域に集中した時，そのサーバの領域を動的に変化させ，それぞれのサーバの負荷が分散する方式を提案する．本方式を用いることで，サーバ間通信量は増大するが，全体としてのサーバ負荷が分散され，情報伝送における遅延時間を小さくすることによってより多くのクライアントをサポートできることを期待している．

以下，2章では従来の仮想空間管理方式の簡単な概要や問題点を述べる．3章では動的な管理領域変更方式について述べ，4章ではシステム構成に関して述べる．5章では，提案方式によるプロトタイプシステム，及び評価について述べ，最後にまとめと今後の課題について述べる．

2 仮想空間管理方式

2.1 仮想空間の管理

多人数参加型仮想空間に求められるものとして，あるシーンにクライアントのアクションを反映させることによって，複数のクライアントが同じシーンを視覚的にとらえた時，その情報をリアルタイムに，かつ情報の欠損が無く得ることができなければならないということが第一にあげられる．これを一貫性の保証という．

この一貫性の保証を保つためにクライアントから更新情報がサーバに対して送られ，サーバが管理するそれらの情報が各クライアントに送られる．一つのサーバでこのような情報全てを管理しようとすると，負荷の集中により情報の遅延や欠損などが起こる可能性がある．即ち，リアルタイム性が失われたり滑らかな動作を表現できなくなったりする恐れもある．よって，このような単一のサーバによる方式に対してサーバを複数用意し，全体の空間を管理することによってそれぞれのサーバに対しての負荷を分散させる「複数サーバ」方式が存在する．複数サーバ方式には主に以下の二つが考えられる．

まず，それぞれのクライアントをサーバごとに割り振る「クライアント分配管理方式」が考えられる．ワールドを複数のサーバで管理しておき，クライアントがワールドに接続する時，それぞれのサーバに接続しているクライアント数が均等になる様に管理する方式である．クライアントがサーバに均等に分配されるため，サーバ負荷が分散される．

しかし，クライアントの視界を考えてみると近距離の情報が欲しい場合が多い．これは他のアバタと会話をしている状況や，動的なオブジェクトやアバタを見ている時である．この様な時，他のサーバが近距離情報を管理している場合に，サーバ・クライアント間通信だけでなく，サーバ・サーバ間通信が頻繁に起こり，サーバに負荷が余計にかかる状況が存在する．

そのため，近距離に位置するアバタはできるだけ同一サーバで管理することが望ましいと考えられるため，クライアントをサーバに分配するのではなく空間に対してサーバを分配する方式が考えられる．

その方式として，仮想空間を任意数の領域に分割し，それぞれを複数のサーバで領域ごとに管理する方式が存在する．これを「空間分割管理方式」という．空間分割管理方式では他のサーバが管理しているクライアントの情報を正確に表現しない方式も存在する．「正確に」とは，簡易的な画像としてアバタを表したり，厳密な位置を表したりせずに，存在を示すだけの方式である [2]．

2.2 空間分割管理方式の問題点

上述の空間分割管理方式においては，その効果が現れない状況が存在する．つまり，多くのクライアントが集中した管理領域を管理するサーバは他のサーバよりも負荷が大きくなってしまい，複数のサーバによる負荷の分散ができていない状況が発生してしまうことになる．

図1は管理領域の一部にクライアントが集中してしまっている状況を示している図である．実線は全体の仮想空間の境界であり，それを四つのサーバで管理している．点線はそれぞれのサーバが管理している領域の境界線である．サーバ2とサーバ3の領域にクライアントが集中し，サーバ1とサーバ4にはクライアントが全く存在していない．そのため，サーバ2とサーバ3に負荷が集中してしまい，負荷の分散が行なわれていない状態が発生している．

3 動的な管理領域変更方式

三次元仮想空間における重要な項目の一つとして，多くのクライアントの接続がある場合においても情

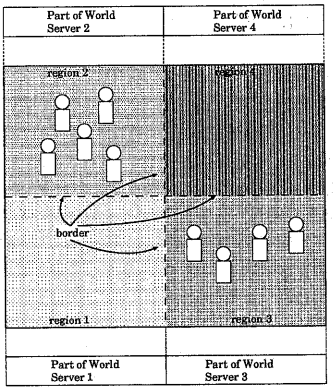

図 1: 空間分割管理方式の問題点

報伝送における遅延時間を小さくすることが挙げられる. サーバ負荷の増加は情報処理能力の低下を招き, 遅延時間の増加につながるためサーバ負荷の分散は非常に重要である. そこで,2.2 節で述べた問題点を改善するために, 以下のような方式をとることによって, 仮想空間を空間分割管理する時, それぞれのサーバへの負荷が分散されるシステム方式を提案する.

3.1 概要

従来の空間分割管理方式では, サーバが管理している領域 (以後, 管理領域) が固定的だったため, クライアントの集中が起こり得た. そこで, クライアント数の閾値をサーバで決めておき, 管理領域に接続してきたクライアント数がその閾値を超えた時, 管理領域を動的に変更させ, 管理領域の一部を他のサーバに管理してもらう方式を提案する.

3.2 サーバ管理領域の動的な変更

サーバ管理領域の動的な変更に関するアルゴリズムを以下に示す.

Step1 任意の管理領域にクライアントが集中し, 管理領域に設定したクライアント数の閾値を超えた時, 他のサーバに対して管理依頼を送信する. 最もクライアントが少ない領域の一部が管理依頼の対象となる.

Step2 「依頼」を受けとったサーバはクライアント数を調べ,「応答」を返す.「応答」の内容は以下である.

- クライアント数が, 設定した下限のクライアント数の閾値より少ない時は「承諾」, このサーバが行なう動作として「依頼」を出してきたサーバの管理領域の一部を管理する
- クライアント数が, 設定した下限のクライアント数の閾値より多い時は「拒否」, このサーバが行なう動作として「依頼」を出してきたサーバの管理領域の一部を管理しない

Step3 「応答」を受けとったサーバは,「承諾」を受けとった時, サーバに接続しているクライアントに対して, 管理領域が変更されたことを示す「変更」を送信する. 複数のサーバからの「承諾」を受けとった場合, それぞれのサーバに対して均等にクライアントを割り振る. 全て「拒否」の時は, 全体のサーバに対して負荷がかかっている状態なので, サーバ間のクライアントの移動や, クライアントがワールドとの接続を切断するまで待つ.

Step4 「変更」を受けとったクライアントは, 管理される領域に応じて, 接続するサーバを変える.

図 2から図 4に以上の方法で行なわれた領域の動的な変更の動作例を示す. それぞれの図は全体の世界における 1 つの空間を 2 つのサーバが管理している状態を上空から見ている図である. 実線はそれぞれのサーバに対する管理領域であり, クライアントからサーバに対して引かれている矢印はクライアントがそのサーバに管理されていることを示している.

図 2では, サーバ 2 の管理領域に対してクライアントが集中している状態である.*Step1* として, サーバ 2 は自身の管理領域におけるクライアント数閾値を設定しておき, 接続クライアント数が閾値を超えた時, 全てのサーバに対して管理領域の一部の管理の「依頼」を送信する. *Step2* として,「依頼」を受信したサーバは, 接続しているクライアント数を調べ,「応答」を送出してきたサーバに対して「応答」を送信する. この時, サーバ 1 にはクライアントはいないので,「応答」は「承諾」である. 図 2では, サーバ 2 のクライアントが少ない方の管理領域の一部を管理するために,「承諾」を送信している.

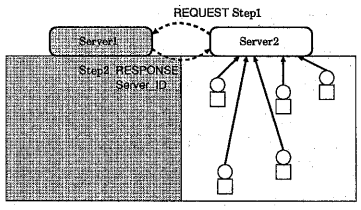

図 2: 領域変更の動作 Step1,2

図 3は,*Step3* の動作を示している図であり,「承諾」を受けたサーバ 2 が, 管理領域が変更されたことをクライアントに対して送信している状態を示している.*Step2* より, サーバ 1 がサーバ 2 の領域の一部を管理することになったためである. よって, クラ

イアントは接続するべきサーバを状況に応じて変える必要がある. また, 変わったことを認識している必要がある. 故に, サーバ 2 に接続している全てのクライアントに対して管理領域が変化したことを送信しなければならない.

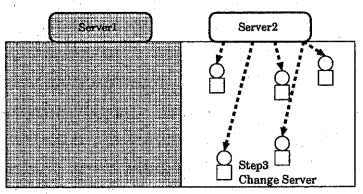

図 3: 領域変更の動作 Step3

図 4は,*Step4* の動作を示している図であり, サーバ 2 の管理領域が変化したことによってクライアントが接続するサーバがサーバ 1 に変わっている状態である. 先のサーバ変更の指示を受けたクライアントは今まで接続していたサーバ 2 との接続を切断し, 新たにサーバ 1 に接続する. この時, サーバ 2 に接続していた 2 人のクライアントは領域の変更を受け, 負荷の少ないサーバ 1 に管理されるため, サーバ 1 に接続する. 図 4では, クライアント数がそれぞれのサーバに対して分散されていることがわかる.

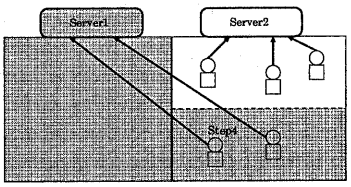

図 4: 領域変更の動作 Step4

3.3 サーバ管理領域の復元

前小節で任意のサーバに対してクライアント数が閾値を超えた時の管理領域の変更を行なったが, 他の領域の一部を管理しているサーバの管理領域のクライアント数が多くなってきた時, 負荷の分散が行なわれなくなる可能性があるので, 領域の復元を行なう必要がある. 以下にそのアルゴリズムと事例を示す.

Step1 管理領域を元に戻すために,「依頼」を送信してきたサーバに対して管理領域の一部を元の状態に戻す指示である「復元」を送信する.

Step2 「復元」を受けとったサーバと,「復元」を送信したサーバは, それぞれ接続しているクライアントに対して, 管理サーバが変わったことを示す指示を送信する.

Step3 指示を受けたクライアントはそれぞれ元のサーバに接続し直す.

図 5から図 7として, 領域の復元例を示す. この図は先の領域変更を行なった図 4の状態に続く図である. 図 5では, 他の管理領域からクライアントが新たに接続してきた状態を示している. この時, サーバ 1 は管理領域を元の状態に戻す必要がある. よって, 管理領域を元に戻すために, 依頼を送信してきたサーバに対して管理領域の一部を元の状態に戻すよう指示を出す.

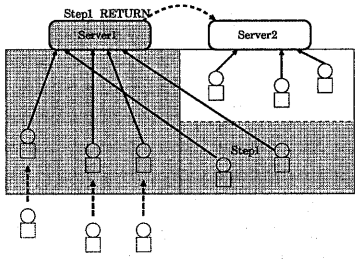

図 5: 領域復元の動作 Step1

図 6は, 領域の復元の指示を受けたサーバ 2 が, 管理領域が元の状態に戻ったという指示をそれぞれのサーバに接続しているクライアントに対して送信している図である.

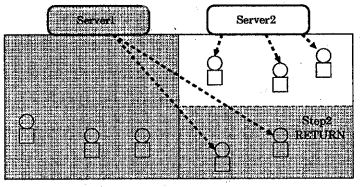

図 6: 領域復元の動作 Step2

図 7は, 管理領域が元の状態に戻ったことを示す図である. 図 6では, サーバ 2 に 3 人のクライアントが接続していたが, 元の管理領域の状態と同様に 5 人のクライアントが接続している.

4 システム構成

サーバ, クライアントの構成として, 図 8に示す.

4.1 サーバモジュール概要

サーバを構成しているモジュールとして, 以下の三つがあげられる.

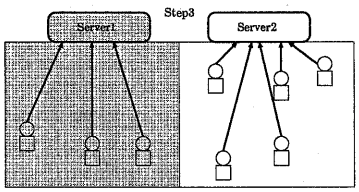

図 7: 領域復元の動作 Step3

- メインサーバモジュール (図では Main Server)
- サーバ間通信制御モジュール (図では Server-Server Connection Control(SSCC))
- サーバ・クライアント間通信制御モジュール (図では Server-Client Connection Control(SCCC))

メインサーバモジュールは, 他のサーバ, 新しいクライアントとの接続を待ち, SSCC,SCCC を生成するためのモジュールである. また, 領域の変更依頼や, 復元などを制御する.

SSCC は, 他のサーバからの情報を受けとるモジュールであり, 他のサーバで起こったクライアント数の変化, 依頼 (クライアントの位置座標の更新, クライアントのサーバ接続切替え, 領域管理依頼など) について受信し, クライアントに情報を送信するために,SCCC を呼び出すモジュールである.

SCCC は, クライアントの数だけ生成され, クライアントとの情報の送受信, また, 他のサーバにクライアントが接続している時, 他のサーバの SSCC に対して情報を送信する. また, 空間情報 (ここでは, オブジェクトや, 壁) を有し, 接続してきたクライアントに対して, 空間情報を送信する. ただし, クライアントが他のサーバに接続する時, 対応している SCCC は消去される.

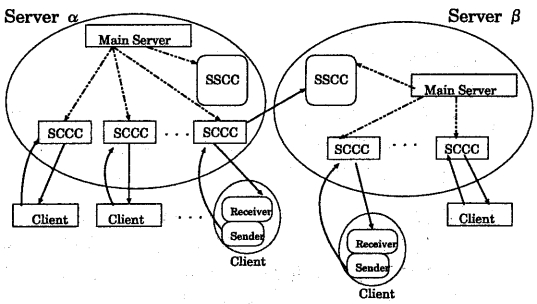

図 8: サーバ, クライアント構成

4.2 クライアントモジュール概要

クライアントはサーバからの情報を受けとる Receiver モジュール, また, クライアントの情報をサーバに送信するための Sender モジュールとして考えることができる. Sender モジュールは, 相対位置座標,ID, 動作コマンドをサーバに対して送信する. また, クライアントはサーバからの以下のような情報により, 自身や他のクライアントの状態を更新しなければならない.

(1) サーバの管理する空間に対する他のクライアントの相対位置座標,ID

(2) 管理サーバの切替え

(3) 管理領域の変更と復元

(1) の情報は,Receiver モジュールが SCCC モジュールから ID, 相対位置座標, 動作コマンドを受けとることによってどのアバタからの情報かがわかるため, 他のクライアントの更新を行なうことができる. 全てのクライアントにはワールド全体の ID が設定されており, どの空間のサーバに管理が移っても ID は一意である.

(2),(3) は, 今まで接続していたサーバとのコネクションを切断し, 他のサーバへの接続を成立させるための情報である.

5 評価実験

前節に示した空間モデルのプロトタイプシステムにおいて, 多くのクライアントが同時に接続し, 充分な時間が経過した時, クライアントの送出した情報が相手に届かなかった割り合いを測定し, 評価した.

本実験では, 疑似的なクライアントとして, ダミープログラムを作成し, そのプログラムによるクライアントがあたかも仮想空間に接続しているかのようにして評価させることにした. このダミープログラムは一定の間隔で更新情報をサーバに送信する. この更新情報は, クライアントの位置座標であり, 乱数によって, 移動方向, 距離を変化させることにする.

5.1 プロトタイプシステム

本稿で提案した動的な管理領域変更方式を二つのサーバから構成されるプロトタイプシステムとして構築した.

サーバとして,UNIX ワークステーションを用い, クライアントとしては Microsoft Windows 上で Sony が提供している,Community Place Browser(Version2.0 Preview Release R1) による接続と C で記述した疑似クライアントによる接続を可能とするように実装を行なった. また, これらの通信は同一の LAN 内で行なう.

プロトタイプシステムを図 9に示す. 図の粗い点線は Server α と Server β の境界を示しており, ク

ライアントがどちらかの管理領域に集中した時, 管理領域の一部が集中していないサーバに管理される. それぞれの管理空間は,x 軸方向に空間がつながっており自由に行き来できるようになっている. アバタの身長を 1 とした時の一つのサーバ管理領域の大きさを x 軸に 50,y 軸に 20,z 軸に 50 と設定する. 回りは壁で覆われており, 壁を通り抜けることも, 壊すこともできない. 従って, この領域以外はクライアントが存在することはできないが, この空間内ならばどのように動いても構わない.

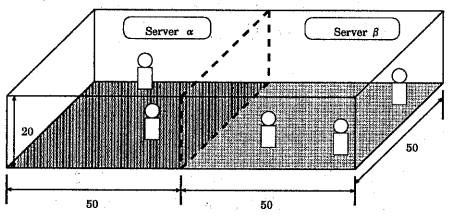

図 9: プロトタイプシステムの管理領域

5.2 実験方式

測定クライアントを A とし, 情報を送出する任意のクライアントを B とする. クライアント B から送出された情報がクライアント A に届かなかった割合 (廃棄率) を調べた.

接続クライアント数を固定して情報送出頻度を変える実験と情報送出頻度を固定してクライアント数を変える実験を行なった.

5.3 実験結果, 及び考察

接続クライアント数を 10 人に固定し, 横軸に情報送出頻度 (回/sec) をとして変化させた時のグラフが図 10であり, 情報送信頻度を 5 回/sec に固定し, 横軸にクライアント数を変化させた時, それぞれ単一サーバ, 空間分割方式, 動的な領域変更方式の廃棄率の変化を縦軸に示した図 11である.

図 10では情報送信頻度が多くなるにつれ, 情報の廃棄が頻繁に起こるようになっている. これは, サーバの能力に対して全ての情報を処理することが困難になるということを示している. 動的に領域を変化させた時, 情報送信頻度が 4 回以上の時, 従来の方式より廃棄率が低くなっているが, 情報伝送頻度が少ない時は single server の方が廃棄率が低くなっている. これは, 空間分割方式や, 動的な領域変更方式ではサーバの切替が起こるため, サーバの処理する情報量が少ない時あまり効果が現れないことを示している.

また, 図 11ではクライアント数が 10 人以上の時, 従来の方式に比べ, 廃棄率は低くなっており, 従来の方式に比べ, サーバの負荷分散がなされていることがわかる.

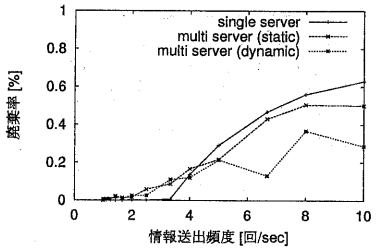

図 10: クライアント数固定廃棄率

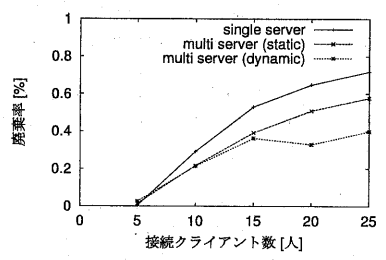

図 11: 情報送出頻度固定廃棄率

6 まとめと今後の課題

本稿ではクライアントが任意のサーバ管理領域に集中した時に動的に管理領域を変更することでサーバの負荷を分散させるための方式を提案した. また, クライアントが受けとる情報量の廃棄率によって従来の方式と比較した結果, サーバの負荷分散の改善が見られた. 本方式における特徴として, クライアントが管理領域の一部に集中した状況でのサーバ負荷分散が可能であるという点があげられる.

今後の課題として, どのように動的に管理領域を変更する方法が最も効率的であるかということについて考慮していくことが重要である.

参考文献

[1] 小野 仁, 西村 浩二, 相原 玲二: "分散サーバによる VR 空間の実現", 電子情報通信学会, 85-12, pp. 67-72, Nov. 1997.

[2] 箕浦 大祐, 山名 岳志, 正木 茂樹, 一之瀬 進, "多人数参加型 3 次元仮想空間における大規模人数表示方法", 電子情報通信学会論文誌, vol.J81-D-II, No. 5, pp. 962-971, Mar. 1998.