マルチメディア通信と分散処理 99－7
（2000．9．22）

多人数参加型仮想空間管理サーバにおける動的領域変更方式に関する実験的考察

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

{ shimizu, kansuke, onozato } @ nztl.cs.gunma-u.ac.jp

あらまし　多くのユーザが仮想空間に入り込み、そこで様々なインタラクションが発生する多人数参加型仮想空間サービスが近年盛んに開発されてきている。しかし、多数のユーザがそのサービスにアクセスすることにより仮想空間を管理するサーバに多大な負荷がかかる問題点が以前から指摘されている。これに対し、我々は仮想空間を分割してそれらを別々のサーバが管理することにより負荷を分散させ、さらに負荷の一局性による分散管理の効率の低下を防ぐための管理領域の動的変更方式を提案し、その効果を示した。本稿では、管理サーバの配置がローカルに集中する場合だけでなく、インターネット上に分散された場合における本方式の効果を調べるために、3つの大学にサーバを置いて実験を行い、その結果について考察する。

キーワード　仮想空間, 動的領域変更方式, 分散管理サーバシステム

Experiments on Floating Method of Regions by Distributed Servers on the Internet in Virtual Environment

Kouichi Shimizu, Ushio Yamamoto, and Yoshikuni Onozato

Graduate School of Engineering, Gunma University
1-5-1, Tenjin, Kiryu, Gunma
{ shimizu, kansuke, onozato } @ nztl.cs.gunma-u.ac.jp

Abstract　Recently, a number of cyberspaces has been developed, in which users control their avators and have interactions with each other. However, one of the problems in managing such virtual environment is the deterioration of service quality by heavy load on the server when many users connect to it. We have proposed a new management method of virtual environment by distributed servers based on dividing virtual space. In this paper, we show the results of experiments of our method on the environments in which not only all servers are on Local Area Network, but also the servers are distributed on the Internet.

keywords　Virtual Space, A Method of Dynamically Changing Managed Region, Distributed Management Server System

1 はじめに

多人数参加型仮想空間においては，基本的にユーザ自身を仮想空間上に投影するアバタ (AVATAR) の位置座標と向きの情報はサーバ (Server) からクライアント (Client) へ基本的に常に送られており，クライアントでは，この位置情報を元にして画面上の他アバタの3次元表示もしくは非表示を決定するような構造である．よって，サーバ・クライアント間で最低限交わすデータとして位置座標と向きの情報があるため，ユーザ数が増加するに当たり CPU 負荷や通信回線における負荷が高くなるため，大きな負荷がサーバに対してかかってしまう．[1] 負荷によるユーザ数の制限をすることは仮想空間サービスを提供する側，参加する側の双方にとって好ましくないことであり，この問題に関して様々な視点からのアプローチがなされている．[2] そこで我々は以前，この問題に対して空間を分割し，それぞれの領域をサーバ負荷に応じて変化させて負荷を分散させるためのプロトタイプシステムを構築し，その評価を行なった．[3]

しかし，実際のサービスでは管理サーバを遠距離の位置に複数配置して運用することも考えられ，クライアントはむしろ遠距離からの接続が主であるため，サービスに近付けた上での仮想空間サーバ負荷分散システムの運用について考える必要がある．

本稿ではより効率的なサーバシステムの構造に関して詳しく述べ，サーバ数を増やした時の閉じた LAN 内での実験，サービスに近付けた状態での実験，即ちインターネット上での実験を行ない，その効果を確かめる．また，サーバ数が増加した時の動的領域変更方式に関する再帰的領域変更方式についても提案する．

以下,2章では仮想空間管理サーバシステム，及びモジュールについて述べ，3章では我々が提案している動的領域変更方式に関しての考察を述べる．4章では，実験環境や結果を述べ，最後にまとめ，及び今後の課題について述べる．

2 サーバシステム，モジュール構成

本章では，仮想空間を構築するためのサーバシステムとモジュール構成について述べる．

2.1 サーバシステム

仮想空間を管理するためのサーバシステムとしていくつかあげられ，全てのサーバ同士にコネクションを確立させることによって通信を行なう構造などが考えられるが，本方式ではメタサーバ (Metaserver) を用いた図1のようなサーバシステムを用いる．

メタサーバはそれぞれの空間を管理するサーバの状態を知り，負荷状況を監視し，領域の変更を指示する．それぞれのサーバは，それ自身に接続しているクライアントからの情報を他のクライアントに送信するために，メタサーバとそれぞれの空間を管理するサーバに送信することで一貫性を保つことができる．

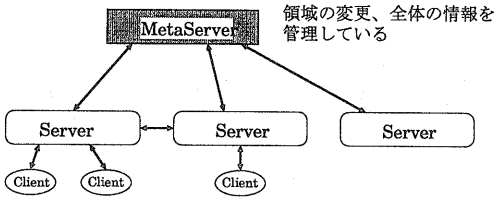

図 1: サーバシステム

これによって，全ての分割領域にクライアントが存在する場合，サーバ・サーバ間に $O(n^2)$ のコネクションが必要となるため，これを解決するために基本的に隣接する空間からのみ情報を得るようにすることによってサーバ・サーバ間コネクション数を $O(n)$ に減少させるようにした．

2.2 モジュール構成

図1のサーバシステムにおけるモジュールについて詳細に説明する．図2に概要図を示す．

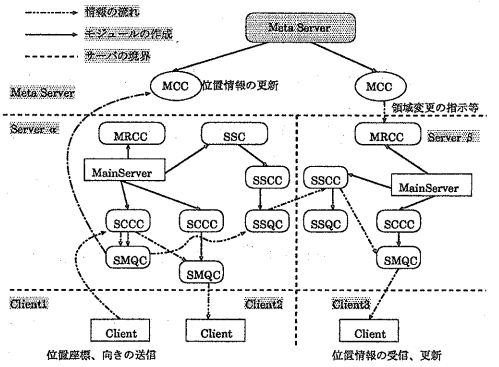

図 2: システムモジュール

システムモジュールの構成としてはメタサーバ，サーバ，クライアントの3つに大きく分けられる．

2.2.1 メタサーバモジュール

メタサーバはワールド (全体の空間) を管理するサーバであり，それぞれの領域を管理するサーバとの接続を行ない，それぞれのサーバに接続しているクライアント数，位置情報を常に知っているものとする．

サーバからクライアントの位置座標を任意の時間間隔で受信することによって全てのクライアントの位置を把握し，領域の変更をサーバに指示する働きを担う．クライアントの動きが大きい時 (走る等) は時間間隔を短く，小さい時 (歩いている，止まっている等) は時間間隔を長くとり，メタサーバの負荷を状況に対応させて減少させている．

メタサーバモジュールは MCC(Metaserver Connection Control) を生成し，ワールドに関する情報を保持している．MCC によって，サーバとの通信を行ない，全てのクライアントの位置情報等を得る．また，領域の変更を行なう必要が生じた時に，それぞれのサーバに領域変更の指示を送信する．

2.2.2 サーバモジュール

サーバモジュールではメインサーバモジュールがメタサーバ，他のサーバ，クライアントと通信するためのモジュールを作成する．図 3 に各サーバモジュールの構成図を示す．

略称	名称	主要動作
MRCC	Metaserver Receiver Connection Control	メタサーバとの通信
SSC	Server-to-Server Constructor	SSCCを隣接しているサーバの数だけ生成する
SCCC	Server-to-Client Connection Control	クライアントとの通信
SMQC	Server-to-Metaserver Queue Control	クライアント、メタサーバに対して送信する情報をＱｕｅｕｅに蓄える
SSCC	Server-to-Server Connection Control	他のサーバからの情報を受信
SSQC	Server-to-Server Queue Control	他のサーバに対して送信する情報をＱｕｅｕｅに蓄える

図 3: サーバモジュールの主要動作

MRCC(Metaserver Receiver Connection Control) モジュールはメタサーバからの領域変更に関する情報を受けとり，その指示をクライアントに送信する役割をする．

SSCC(Server-to-Server Connction Control) モジュールは隣接するサーバの数だけ，SSC(Server-to-Server Contstuctor) モジュールによって作成され，

SSCC は SSQC(Server-to-Server Queue Control) モジュールを作成する．SCCC(Server-to-Client Connection Control) モジュールは自身に接続しているクライアントの数だけ生成され，クライアントからの情報を受けとり SMQC モジュールに渡す働きをする．

SMQC,SSQC モジュールでは，情報を任意の大きさの Queue に蓄え，その Queue に入る情報量を超えた時，一部の情報を欠損させることによってサーバ対する負荷をかけ過ぎないような構造になっている．ここで Queue を用いている理由として常に情報を送り続けると，サーバに対する負荷が大きくなってしまうため，サーバの能力に応じて一定の処理を行なう時，情報を欠損させることでサーバの限界を保つ構造になっている．

2.2.3 クライアントモジュール

クライアントモジュールは基本的に位置情報を送信し，他のクライアントの情報を接続しているサーバから受けとる働きを行なう．また，領域間の移動，領域の変更などの指示をサーバから受けとり，接続するサーバを切替える．詳細は文献 [3] を参照されたい．

3 動的領域変更方式に関する考察

文献 [3] では，全体を管理するサーバ数が少ない状況のみを想定していたため領域変更も単純であった．しかし，全体の空間が大きくなるにつれ，その空間を管理するサーバ数も多くなることで領域変更をしても効果が現れない状況が発生する．よって，本章ではサーバ数が増加した時でも負荷が分散される領域変更方式について簡単に述べる．また，領域間での移動に関するサーバ接続の切替えに関して考察する．

3.1 領域間移動に関する考察

クライアントは透過的にサーバ間移動を行なうことができ，自由に動き回ることが可能であるため，クライアントのサーバ間移動に伴い領域の境目でクライアントが頻繁に行き来する状態が発生する．接続するサーバを頻繁に変えることはサーバに負荷をかけるため，分領域を重ねることでそのような事態を避ける方式が実装されている．[4]

動的領域変更方式を用いた場合，文献 [4] の方式の場合，領域の境目が曖昧になるため領域の管理が困難になる．そこでクライアントの周辺に保護領域を用いてクライアントが保護領域の外側に退出するまでサーバ切替えを行なわない方式を提案する．

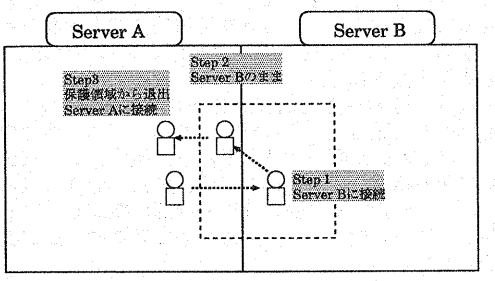

図 4: 分割領域の領域間移動に関する方式

図 4 は領域間を移動するクライアントの動きを示

す図である．実線はそれぞれの領域の境界を示し，点線での矢印はクライアントの動きを示し，点線の四角形はクライアントの保護領域を示している．クライアントは ServerA に初め接続しており，クライアントが領域の移動を Server A → ServerB → ServerA とした時の図である．

Step1 クライアントがサーバの境目を跨いだ時，領域を管理するサーバに接続を切り変える．

Step2 クライアントの周辺に任意の大きさの保護領域を設定する．

Step3 保護領域の内側に存在する間はサーバの切替えを実行しない．しかし，保護領域の外側にクライアントが退出した時，保護領域を消去し，領域を管理するサーバに接続を切替えていなければ領域を管理しているサーバに接続を切替える．

この方式によってクライアントが境目上を行き来する状況において不必要なサーバ切替えがなくなることが期待される．

3.2 動的領域変更方式

それぞれのサーバに接続しているクライアント数を知っているメタサーバが，任意のサーバが管理できるクライアント数である閾値を超えた時，隣接するサーバに領域の一部の管理を任せることによってサーバ負荷の分散を行う方式を文献 [3] で提案した．最も単純に全てのサーバに対して均等にクライアントが接続するような動的領域変更方式を用いていたが，空間が大きくなるにつれ，サーバ数が多くなり文献 [3] の方式では対応し切れない状況が存在する．これは，隣接するサーバの負荷が大きい場合 (クライアント数が多い場合)，期待される効果が得られないという状況が発生するためである．

3.3 再帰的方式に関するアルゴリズム

この問題を解決するために，隣接しているサーバの負荷を分散した後，問題のサーバの負荷を分散させる方式を提案する．これを再帰的方式と呼ぶことにする．再帰的方式のアルゴリズムを以下に示し，動作例を図 5 に示す．

図 5 は全体空間を鉛直上方向から見た図であり，それぞれの正方形が部分空間である．等しい能力のサーバが部分空間を管理しそれらが接することで全体空間を構成している図である．また，無限に並んでいると仮定する．数字は部分空間に接続しているクライアント数であり，矢印の順に再帰的方式が実行されている．

最も濃い色の部分空間が再帰的方式での中心とみなされ，次に濃い色の部分空間は中心の部分空間の 4 周辺の部分空間である．本方式によって，以前の方式に比べ，サーバ負荷が効率的に分散されることが期待される．

Step1 クライアント数が α を越えた時，そのサーバを中心として四方のサーバのクライアント数をメタサーバに問い合わせて，*Step2* を実行する．

Step2 四方のサーバの中で一つ以上が β よりもそのサーバに接続しているクライアント数が少ない時，領域の変更を行ない終了．四方のサーバ全てが β よりも多かった時，*Step3* を実行する．

Step3 最もクライアント数が多かったサーバを中心にして *Step1* を実行する．*Step1* 以降が終了したら一つ前に中心にしたサーバを中心にして *Step1* を実行する．

ここで α はサーバの能力に応じて決められ，管理に支障が出る直前の値とし，サーバに負荷が集中していると考えられる値である．また，β とは，他の管理領域の一部の管理を引き受けても管理に支障が出ないクライアント数の値とする．

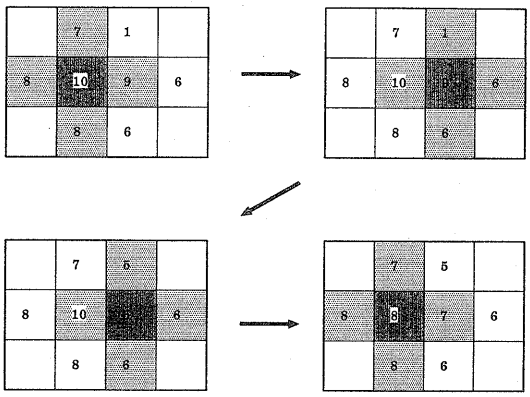

図 5: 再帰的方式による動作

4 評価実験

4.1 概要

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する．このダミープログラムは一定の時間間隔で位置情報を送信する．ダミークライアントの動きとして，任意の場所に関して非常に集まりやすい場所を想定する．これは，面白みのあるアトラクションと考えてもよいし，仮想空間の最もポピュラーであるチャットをする場所とも考えられる．今回の実験では，この集まりやすい場所 (ホットスポット) をそれぞれのサーバ管理領域に一

点用意し，ホットスポットに一定間隔時間後にアバタが移動するという状況を仮定する．

4.2 実験環境

実際のサービスにおいて，サーバとクライアント間には物理的な距離，ネットワーク的な距離といった大きな問題が存在する．よって今回の実験では，独立したLAN内で行なった実験と，インターネット上での実験で行なった．独立したLAN内では通信による遅延時間の影響がほとんど無いと考えられるため純粋にそれぞれの方式におけるサーバの負荷による情報欠損率を示すことができる．インターネット上での実験では，東北大学，静岡大学，群馬大学にそれぞれメタサーバ，サーバ，ダミークライアントを配置して行なった．また，クライアントのパラメータとして，全体の仮想空間に参加するクライアント数を変化させ，サーバに対する負荷の基準を情報の欠損率として測定し，評価した．また，サーバとして,UNIX ワークステーション (Sun Ultra30, Ultra-2, Ultra-1, OS Solaris7) を用い，クライアントとしては MicrosoftWindows 上で Sony が提供している,Community Place Browser (Version2.0 Preview Release R1) による接続と Java 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した．また，疑似クライアントマシンはワークステーション (Sun Ultra30, OS Solaris7) を用いた．

4.3 動的な領域変更方式における実験評価

4.3.1 プロトタイプシステム

本節の実験でのサーバシステムは，図1のサーバシステムを用い，メタサーバ数1，サーバ数3で行なった．全て同じ大きさの立方体 (部分空間) をそれぞれのサーバが管理することでワールドを形成する．一つのサーバは部分空間を管理する．

4.3.2 実験方式

測定クライアントをA，情報を送信するクライアントをBとする．BからA宛に情報を5000回連続で一定間隔で送信した時の情報の欠損率を測定する．一定時間当たりのサーバが処理する情報量にはそれぞれのサーバごとに限界があるため，そのサーバの処理する情報量をこえた時，情報が欠損するシステムになっている．(2.2.2節で示したQueueによる) 図6, 7, 8として，横軸にクライアント数，縦軸に情報の欠損率 (%) の変化を示している．また，3本のグラフで一つの図を形成しており，それぞれ，クライアント数が均等にそれぞれのサーバのホットスポットに分散集中するように，即ち，サーバ負荷分散としては最も理想的な動きに調節した時に得られたグラフ (ideal)，全てのクライアントが任意のホットスポットへ移動するようにダミークライアントを調節した時に得られたグラフ (worst), worstと同様の動きで動的領域変更方式を用いた時のグラフ (dynamic) を描画している．

4.3.3 実験結果

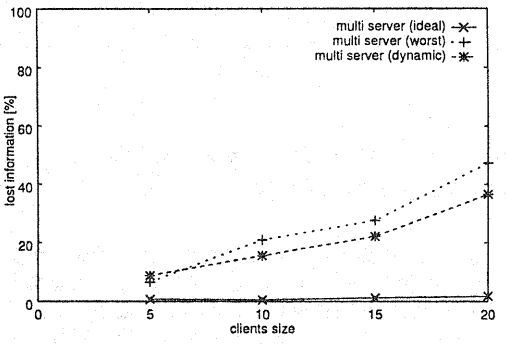

図6: 閉じたLAN内における情報欠損率

図6は，群馬大学の閉じたLAN内で行なった実験であり，通信回線による遅延時間の影響がない状態で行なわれた．結果として，後述するインターネット上で測定したグラフよりも全体としての情報欠損率が低くなっている．理由として，インターネット上で通信が行なわれた場合，遠距離間の通信においては一般的に遅延時間が大きくなるため，情報の到着間隔が変化すると考えられる．そのため，一度に多くの情報が到着したり，暫く情報が到着しないといった状況も発生する．本サーバ方式ではQueueを用いて情報を蓄え，Queueの容量を超えた時情報は欠損される．即ち，ネットワークの回線における通信速度の変化が，一度に大量の情報がQueueに流れ込む状況を引き起こし，情報の欠損が起こっていると考えられる．しかし，LAN内では輻輳などによる遅延時間の増加といった問題は発生しないため，全体としての情報欠損率が低くなっていると考えられる．

3つのサーバに均等にクライアントが接続するidealでは，3つのサーバに均等に負荷が分散されており，全体で20人程度，即ちそれぞれのサーバで管理するクライアント数は7人程度ならサーバの処理能力を超えずに管理できることを示している．

また，領域変更を用いた場合 (dynamic) と，用いない場合 (worst) での相対的な比較では，dynamicはidealには及ばないものの，worstの値よりも人数が増えるにつれて欠損率が低くなっており，その効果が示せた．人数が少ない時にはクライアントの位置情報量がサーバの処理能力を超えないため，動的な領域変更の効果はさほど見られないが，人数が多くなるに従って，領域変更のオーバーヘッドよりも負荷分散の効果の方が大きくなっているためである．

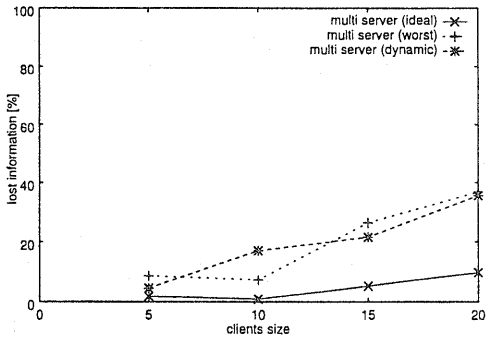

図 7: サーバを東北大学，静岡大学，群馬大学上に置いた状況での情報欠損率

図 7 は，管理サーバを東北大学，静岡大学，群馬大学上に置き，メタサーバとクライアントは群馬大学から接続した時に測定したグラフである．群馬大学から接続するクライアントが東北大学，静岡大学に配置されたサーバに接続し，通信を行なう場合，インターネット上の通信により欠損率にどのような影響が現れるかを調べた．

比較対象として，通信における遅延時間がほとんど無い LAN 内での実験結果である図 6 と比較した時，ideal のグラフにおいて，情報欠損率が高くなるという結果が得られた．原因としてはインターネット上の通信による情報欠損が起こっていると考えられる．

また，worst よりも dynamic の方が，10 人の時に若干欠損率が高くなっているが，クライアントが多くなるにつれて欠損率が低くなり，遠距離にサーバを置いた時においても効果が示せた．

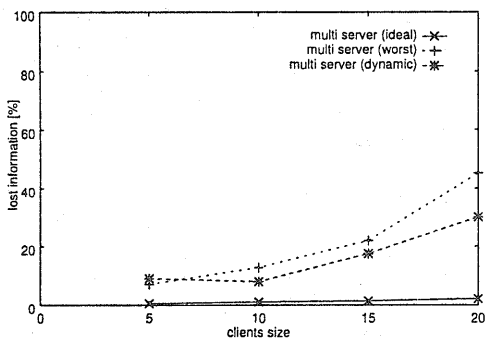

図 8: クライアントを東北大学上に置いた状況での情報欠損率

図 8 は，ダミークライアントを東北大学に配置し，メタサーバ，サーバを群馬大学に配置した時に測定したグラフである．サーバ・サーバ間通信ではインターネット上の通信は発生しないが，クライアント・サーバ間が遠距離であり，インターネット上の通信が発生する状態の時，サーバ負荷にどの程度影響を与えるかについて実験を行なった．

図 6 と比較した時，さほど変わった点はなく，遠距離からのサーバへの接続は本システムにおいて，遅延時間は増加すると考えられるが，情報欠損に関しては影響を与えていないことがわかる．

クライアントが遠距離にある場合でも dynamic の方が worst のよりも欠損率が低くなっており，その効果が示せた．

5 まとめ

本稿では以前から我々が提案している動的な領域変更方式を適用した時の効率的なサーバシステムについて考察をし，インターネット環境での適用を行なった．実験結果に基づいて様々なサーバの配置状況などを考慮した時でも，情報の欠損率が小さくなることが示されたことでインターネット環境でもサーバ負荷が減少することが確認できた．

今後の課題として，実際にサービスとして運用する場合，常に稼働し続けることが要求されるため，長時間の運用に対しての考えも必要である．あらゆる状況での効率的なシステムを実現していくことがこれからの課題であるといえる．

謝辞

本研究における実験を行なうにあたって御協力を頂いた東北大学白鳥研究室，静岡大学富樫研究室，群馬大学小野里研究室の皆様に深く感謝致します．

参考文献

[1] 清末 梯之, 湯田 佳文, 山名 岳志, 加藤 洋一, 正木 茂樹, 一之瀬 進, “クライアントの性能とサービスの多様性に対応した 3 次元サイバースペースシステムの機能分散型サーバアーキテクチャの提案”, 日本バーチャルリアリティ学会論文誌, pp.351-356, Vol.4 No.2 1999.

[2] 豊国 造平, 西村 浩二, 相原 玲二, “複数サーバを用いた大規模仮想空間の分散管理” , 情報処理学会研究会資料, DSM, Nov. 1999.

[3] 清水 孝一, 山本 潮, 小野里 好邦, “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”, 情報処理学会研究報告 分散システム/インターネット運用技術,vol.99, No.56, 14-11, pp. 59-64, Jul. 1999.

[4] 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.