Vol.2012-MPS-88 No.7
2012/5/17

大規模仮想空間における動的領域分割法

榎原博之[†1] 吉岡啓[†1] 松崎頼人[†1]

本論文では，P2P を用いた MMORPG(大規模仮想空間ロールプレイングゲーム) の動的な領域分割手法について提案する．P2P 型 MMORPG は，ゲームが行われる仮想空間を部分領域に分割し，ピアを管理ノードとして各部分領域に設置し管理させることでゲームを進行する．しかし，ピアの処理能力には限界があり処理能力を超える負荷がかかった場合，遅延の発生やゲームの中断につながるため，部分領域を再分割し負荷を分散する必要がある．そこで，各部分領域に部分領域内に存在できるプレイヤ数の上限と下限の 2 種類の閾値を設けることでプレイヤの移動を検知し，動的に領域分割をする負荷分散手法について提案する．動的に領域分割することで，P2P 型通信の問題点であった負荷の集中に対応した負荷分散を行うことができる．シミュレーション実験による検証を行い，既知の手法よりも本提案法が負荷の軽減を図れることを示す．

Dynamic region decomposition method in large-scale virtual space

Hiroyuki Ebara,[†1] Hiraku Yoshioka[†1]
and Raito Matsuzaki[†1]

In this paper, we propose a dynamic region decomposition method of a P2P MMORPG(MMORPG:Massively Multiplayer Online Role Playing Game). The P2P MMORPG divides a virtual space into regions and sets a peer to manage the region as a management node. If a peer takes over load, delay or interruption of the game occurs. So, it is necessary to divide regions in order to distribute the load. We dynamically decompose regions by detecting the movement of players and providing thresholds that is upper and lower limits of the number of players in each region. By our method, it is possible to perform load balancing that corresponds to issues of P2P communication type. We show the proposed methods reduce the load than the conventional method by simulation experiments.

1. はじめに

近年，一般家庭への常時接続回線の急速な普及によって，大人数がネットワークを介して同時に大規模仮想空間内で遊ぶ多人数参加型オンラインゲーム (MMOG: Massively Multiplayer Online Games) が注目されている．1970 年代後半に CAI(Computer Aided Instruction) システムなどの応用によって，MMOG の基礎となる複数人が同じ仮想空間内でプレイできるゲームがアメリカで開発された．1990 年代中盤以降の爆発的なインターネットの普及に伴い MMOG は発展し，数人から数百人で遊ぶシューティングゲームやパズルゲーム，ロールプレイングゲームなどの様々なソフトウェアが登場した．その中でも，MMORPG は圧倒的な人気を誇り，最も加入者数の多い MMORPG である World of Warcraft[1)] は，一時は全世界加入者数が 1200 万人にも及んだ．

MMOG は通信接続方式によって「C/S(クライアント・サーバ) 型」と「P2P(Peer to Peer) 型」の 2 種類に分類することができる．現在の MMOG では一般的に C/S 型の通信接続方式が用いられている．この方式は，中央管理サーバにてゲームサービスを提供し，クライアントとなるユーザがサーバに接続することでゲームを進行する．中央管理サーバがセキュリティの確保やデータの管理を一括して行うので，コンテンツの管理が容易にできるという利点がある．しかし，C/S 型は全ての負荷がサーバに集中するため，サーバの処理能力を超える負荷が発生したとき，最悪の場合サービスが一時停止してしまうスケーラビリティの問題がある．この問題は処理能力の高いサーバや広帯域ネットワークを確保することで補うことができるが，初期費用やネットワーク回線の維持費などが必要となりコストがかさむ．

そこで，C/S 型の通信方式にかわって P2P 型の通信接続方式が検討されている．P2P 型 MMOG では，大規模なゲーム領域を複数の部分領域に分割し，部分領域毎にサーバの役割となる担当ノード (以下，管理ノード) をプレイヤの中から決定し，局所的な C/S 型を構築してゲームを進行する．サーバを介さずにピア同士で負荷を分散するため，上記の C/S 型特有の問題を解決できる．しかし，ゲームを進める上でピア接続切れ時のデータ保護，ピア間で通信する際のデータの整合性の維持，プレイヤの局所的な集中による管理ノードの処理

†1 関西大学
Kansai University

能力を超える負荷の発生といった問題がある．このような問題を解消するため，今日までにいくつかの領域分割手法[2)3)4)]や管理手法[5)6)7)8)]が検討，提案されてきている．

本研究では，MMOG の中でも多人数が同時にプレイする P2P 型 MMORPG における動的な領域分割手法を提案する．P2P 型 MMORPG は,「固定分割手法」や「四分木手法[2)]」といった領域分割手法が既に提案されているが，動的に領域を統合できる手法は知らない．本提案法は，各部分領域に部分領域内に存在できるプレイヤ数の上限と下限の 2 種類の閾値を設けることでプレイヤの移動を考慮した動的な領域分割・結合を行う．本提案を用いることで既知の手法よりも一層の負荷低減を図る．尚，MMORPG は他のジャンルと比較して大規模仮想空間に多くのプレイヤが同時に存在するため，スケーラビリティや探索クエリの増加によって帯域が圧迫されるなどの問題が発生しやすい．そこで，本研究ではスケーラビリティの問題や帯域の圧迫が発生しにくい通信方式である P2P を採用した MMORPG を中心に検討を行う．

本論文は以下の 4 章から構成される．2 章では，管理ノードの役割などについて述べた後，既知の領域分割手法とその問題点，提案法について述べる．3 章では，シミュレーションによる性能評価および考察を行う．最後に 4 章では，本論文の結論と今後の課題について述べる．

2. 領域分割手法

2.1 管理ノード

P2P 型 MMORPG では，大規模なゲーム領域を複数の部分領域に分割しゲームを進行する．この領域分割してできた部分領域を管理するピアのことを管理ノードと呼ぶ．管理ノードは部分領域ごとに設けられ，設けられた部分領域内でサーバの役割を果たす．管理ノードは一般ピアから選ばれ，一般のプレイヤと同じように移動などを行う．従って，管理ノードは自身が管理している部分領域に滞在している必要はなく，別の部分領域に属する場合もある．管理ノードの役割は以下の通りである．相互干渉領域については後で説明する．

- 自身が管理する部分領域内のプレイヤ数が大きくなると，領域を分割し新たな管理ノードを決定する．
- 領域分割後，各部分領域の管理ノードは自分の領域のデータをその領域に属する全プレイヤに送信する．
- 部分領域内に存在するプレイヤと隣接する領域の管理ノードの座標データを取得する．
- 移動によってプレイヤが相互干渉領域に属した場合，プレイヤに相互干渉領域に隣接する管理ノードの座標データを送信する．

管理ノードは計算能力が高く，通信帯域に余裕のあるノードが選定されることが望ましい．また，分割後も同じ管理ノードを引き続き採用すると，管理ノードが領域分割後に部分領域のデータを送信する負荷が減少する．

2.2 相互干渉領域

P2P 型では，大規模仮想空間を部分領域に分割し，各部分領域に管理ノードを割り当て管理することでプレイヤが仮想空間内を自由に移動できるようにしている．しかし，ただ領域を分割するだけでは，プレイヤが管理されている部分領域から違う部分領域へ移動する際にリアルタイムな情報更新ができず，データの誤差が生じてしまう場合がある．そこで，管理ノード間のプレイヤデータを受け渡す場所として相互干渉領域を部分領域と部分領域の間に設置し，データの整合性を維持する．この相互干渉領域を設けることで，管理ユーザ間で整合性のとれた情報更新が可能となる．本論文ではこの相互干渉領域を導入する．

2.3 従来の領域分割手法

2.3.1 領域分割固定法

大規模仮想空間を任意の大きさの部分領域に分割し，各部分領域にサーバの役割をもつ管理ノードを決定してゲームを進行させる．事前に領域を分割し固定しておくことで，領域を分割する処理が必要ないという利点がある．しかし，プレイヤの移動を考慮していないため，プレイヤの密集，過疎には対応できないという欠点がある．

2.3.2 四分木手法[2)]

四分木 (Quad Tree) 手法とは，管理ノードに処理能力以上の負荷がかかると部分領域 (正方形) を 4 つの部分領域 (正方形) に分割する領域分割手法である．この作業を複数回繰り返すことによって，プレイヤの局所的な密集による負荷にも対応することができる．しかし，四分木は領域の分割しか行わないため，部分領域数を減らすことができない．そのため，密集状態が解消された後も分割後の小さくなった部分領域がそのまま残り，プレイヤ数に対して必要以上の部分領域と管理ノードが残るという問題が生じる．管理ノードは自分が管理する領域内から他の領域にプレイヤが移動する際に，移動先の管理ノードにプレイヤのデータ送信を行うので，管理ノードが多いとデータ送信の頻度が増え，負荷の増加に起因する．
図 1 にプレイヤ密集発生時の分割とその後プレイヤが移動した例を示す．図 1 の左上の部分領域に注目すると，プレイヤの局所的な密集発生後，プレイヤが移動し負荷の集中が解消されても分割された部分領域が残ってしまっていることがわかる．

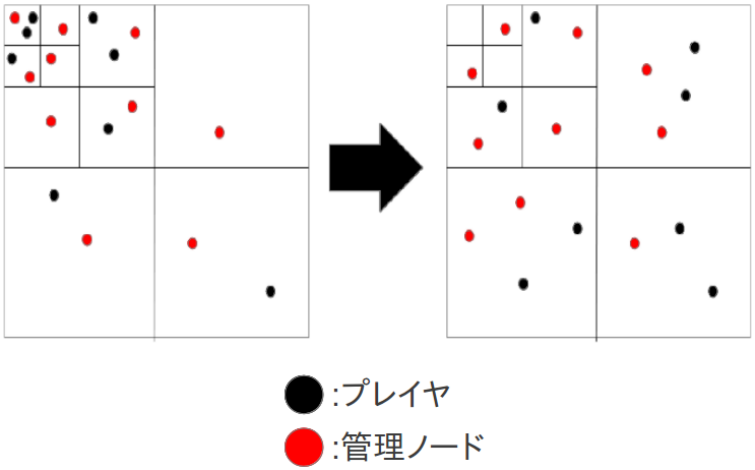


図 1 四分木手法
Fig. 1 Quadtree method.

2.4 基本型動的領域分割手法 (提案法)

前節で挙げた既存法の問題点より，領域分割はプレイヤの移動，部分領域内のプレイヤ数を考慮して動的に行う必要があることがわかる．そこで，分割閾値，結合閾値，領域プレイヤ数の 3 つの閾値を設定する．閾値を用いることで，プレイヤ数に対応して領域数を増減する動的な分割手法を提案する．以下に，分割閾値，結合閾値，領域プレイヤ数について説明する．

- 分割閾値
  任意に決めた各部分領域内に存在できるプレイヤ数の上限値を示す．
- 結合閾値
  任意に決めた各部分領域内に存在できるプレイヤ数の下限値を示す．
- 領域プレイヤ数
  領域を分割する際に各部分領域が内包するプレイヤ数を示す．分割閾値と結合閾値の平均の切り上げ値とする．

基本型動的領域分割手法の概要を以下に示す．

(1) 分割閾値，結合閾値を決める．また，分割閾値と結合閾値の平均を領域プレイヤ数とする．
(2) 部分領域内のプレイヤ数が分割閾値を超える場合，または結合閾値を下回る場合，その領域と接する領域の全てを統合し 1 つの領域 (以下，分割対象領域) とする．
(3) 分割対象領域に対して領域分割 (後述の基本型分割) を行う．
(4) 新しく作成された部分領域に管理ノードを割り当てる．
(5) 分割閾値を超える，または結合閾値を下回る度，(2)〜(4) を繰り返し行う．

本節以降，領域分割手法やアルゴリズムを説明する際に用いる図では，分割閾値 5，結合閾値 1，領域プレイヤ数 3 と仮定する．また，仮想空間は 2 次元平面であり，横軸 (x 軸) と縦軸 (y 軸) からなるものとし，マス目単位で領域を扱うものとする．

2.4.1 基本型分割

分割対象領域の上を優先した左上から領域分割を行う．部分領域の数が多すぎると四分木で述べたような問題が発生するため，プレイヤ数に対して必要最低限の数の部分領域を作成することが望ましい．部分領域の作成数を減らすには，各部分領域の範囲を広くとる必要がある．基本型分割は領域プレイヤ数を超えない限り x 長，y 長を交互に増加させ分割領域の範囲を広げるため，部分領域を大きくとることができる．以下に，基本型分割のアルゴリズムを説明する．

(1) 分割対象領域の有無を調べる．
(2) 分割対象領域の上を優先した左上から分割を開始する．
(3) 部分領域の x 長を 1 マス増加させる．
(4) 部分領域の y 長を 1 マス増加させる．
(5) 部分領域が内包するプレイヤ数が領域プレイヤ数を超えない限り (3)，(4) の動作を繰り返す．
(6) (5) の処理終了後，残った分割対象領域の左上から再び (2)〜(5) の動作を繰り返し，分割対象領域を完全に分割する．

2.4.2 基本型動的領域分割手法のアルゴリズム

本項では，以下に基本型動的領域分割手法のアルゴリズムを示す．

(1) プレイヤの移動 (または新規参入) によって，部分領域内のプレイヤ数が分割閾値を超える，または結合閾値を下回る場合，その部分領域を担当する管理ノードは，自身が管理する部分領域に接する部分領域の管理ノードに分割メッセージを送信する．
(2) メッセージを受け取った各管理ノードは，自身が管理するプレイヤの座標データと部分領域の範囲をメッセージを送ってきた管理ノードへ送信する．
(3) プレイヤの座標データなどを受け取った管理ノードは，自身が管理する部分領域を中

心に分割対象領域を作成する．

(4) 分割対象領域に対して基本型分割を行う．

(5) 分割対象領域を作成した管理ノードは新たに分割された各部分領域に管理ノードを割り当て，新たな部分領域のデータとその部分領域に属するプレイヤノードのデータを送信する．

(6) 新たに割り当てられた各管理ノードは自分の管理している部分領域内のプレイヤに分割後の領域データを送信する．

(7) プレイヤに領域データ送信後，再びプレイヤの移動や新規参入が繰り返し行われる．各部分領域内のプレイヤ数が分割閾値を超える，または結合閾値を下回る度，(1)〜(6) の操作を繰り返し行う．

2.5 改良型動的領域分割手法

前節では閾値を設けることでプレイヤの移動を考慮した動的な分割手法を提案している．しかし，基本型動的領域分割法による分割は部分領域が歪な形になる場合がある．歪な形の部分領域はプレイヤの領域間の移動の頻度を増加させ，管理ノード間の通信負荷を増大させる原因となる．そこで，前節の基本型動的領域分割法を基に，分割領域の形ができるだけ正方形に近づくように分割し，かつプレイヤの移動を考慮した改良型動的領域分割法を提案する．本提案法の流れを以下に示す．

(1) 分割閾値，結合閾値を決める．また，分割閾値と結合閾値の平均を分割プレイヤ数とする．

(2) 部分領域内のプレイヤ数が分割閾値を超える場合，または結合閾値を下回る場合，その領域を中心に分割対象領域を作成する．

(3) 分割対象領域に対して分割判定により，分割を開始するマス (以下，分割開始マス) と分割を進める方向 (以下，分割方向) を決める．

(4) 分割判定で決定した分割開始マスと分割方向を基に，分割対象領域に対して領域分割 (後述の改良型分割) を行う．

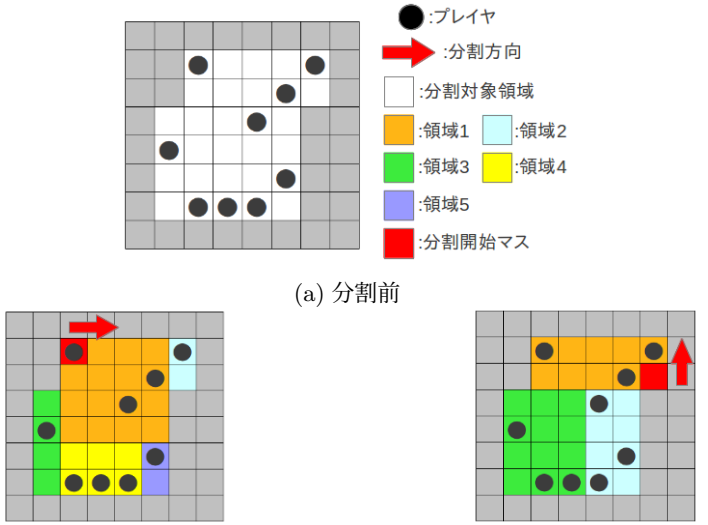
(5) 内包するプレイヤ数が統合領域を下回る，または部分領域の縦横の長さの比が 1:2 あるいは 2:1 以上の部分領域に対して領域矯正を行う．領域矯正については後述する．

(6) 新しく作成された部分領域に管理ノードを割り当てる．

(7) 分割閾値を超える，または結合閾値を下回る度，(2)〜(6) を繰り返し行う．

2.5.1 分 割 判 定

分割対象領域の形によっては，分割すると細長い部分領域ができやすいものがある．細長い部分領域は，プレイヤが移動する際に異なる管理ノードが管理する領域を跨ぐ頻度を増やすので，管理ノード間の通信負荷の増加に起因するという問題点がある．そこで，分割対象領域の形によって分割開始マスを変更し，分割する際の部分領域を広げていく順番も変えることで，ある程度細長い領域をできにくくすることができる．尚，x 長，y 長の順に部分領域を広げ，x 軸方向に分割を進めていくことを横方向とし，y 長，x 長の順に部分領域を広げ，y 軸方向に分割を進めていくことを縦方向とする．分割開始マスと分割方向の違いによる改良型分割後の部分領域の違いを図 2 で示す．改良型分割については後で述べる．



(a) 分割前

(b) 最も上の左から横方向に改良型分割を行った場合

(c) 最も右の下から縦方向に改良型分割を行った場合

図 2 分割開始マスと分割方向
Fig. 2 Starting place and dividing direction.

図 2 より，分割の開始マスと分割方向によって同じ形の分割対象領域でも，分割後の部分領域の形が大きく異なることがわかる．以下に分割判定のアルゴリズムを示す．

(1) 分割対象領域の範囲を調べる．

(2) 分割対象領域を全て内包する最小の四角形を作成する．

(3) 作成した四角形から分割対象領域分を削除する．

(4) 残ったそれぞれの領域の x 長と y 長の差を比較し，一番差の大きい領域 (領域 A と仮定) を選択する．
(5) 領域 A の x 長，y 長を比べ，短い辺と接する分割対象領域の角のマスを分割開始マスとする．また分割方向は，x 長が y 長以上の場合は横方向，y 長の方が大きい場合は縦方向とする．この例では，上に進む．

2.5.2 改良型分割

分割対象領域に対して領域分割を行う．領域プレイヤ数を超えない範囲で x 長，y 長を交互に増加させる．また，部分領域の x 長を増加中に他の部分領域に接した場合，x 長の増加を停止し y 長のみを増加させ部分領域の範囲を広げる．同様に，部分領域の y 長を増加中に他の部分領域に接した場合，y 長の増加を停止し x 長のみを増加させ部分領域の範囲を広げる．このような条件を加えることで，部分領域を必ず四角形に保つことができる．

(1) 分割対象領域の有無を調べる．
(2) 分割対象領域の左上から分割を開始する．
(3) 部分領域の x 長を 1 マス増加させる．
(4) 部分領域の y 長を 1 マス増加させる．
(5) 部分領域が内包するプレイヤ数が領域プレイヤ数を超えない，または他の領域と接さない限り (3)，(4) の動作を繰り返す．
(6) 仮に x 軸で部分領域同士が接した場合は x 長の増加を停止し，y 長のみを内包するプレイヤ数が領域プレイヤ数を超えない，または他の領域と接さない限り増加させる．y 軸が接した場合も同様である．
(7) 分割対象領域が完全に分割されるまで，(2)〜(6) を繰り返す．

2.5.3 領 域 矯 正

基本的領域分割は内包するプレイヤ数のみを考慮して部分領域を広げるため，内包するプレイヤ数が領域プレイヤ数を超さない限り部分領域を広げることができる．しかし，改良型分割は内包するプレイヤ数に加えて部分領域の形も考慮するため，場合によっては内包するプレイヤ数が結合閾値を下回る部分領域や，細長い部分領域ができてしまう．そこで，結合閾値を下回る領域と，部分領域の x 長と y 長の比が 1:2 あるいは 2:1 以上の部分領域を矯正する領域矯正を行う．以下に，領域矯正の流れを示す．

(1) 結合閾値を下回る領域，あるいは部分領域の x 長と y 長の比が 1:2，または 2:1 以上の部分領域を検知する．
(2) 統合する領域によっては，統合後の領域の形が六角形になることがある．そこで，検知した部分領域が縦長の場合は，統合後にできる六角形の領域数が少ない，左右のどちらかの領域と統合する．統合後にできる六角形の数が同じ場合は左の領域と統合する．同様に横長は，上下の六角形の数を比べ，少ない方と統合する．統合後にできる六角形の数が同じ場合は上の領域と統合する．x 長，y 長が等しい場合は，上下左右の統合後の六角形のできる数をそれぞれ調べ，最も少ない方向と統合する．六角形の数が同じ場合は左の領域と統合する．
(3) 統合後の部分領域が内包するプレイヤ数が分割閾値より多い場合，その部分領域の形が縦長か横長かを調べる．
(4) 横長の場合，部分領域が内包するプレイヤ数が半分ずつになるように領域を縦に分割する．同様に，縦長なら部分領域が内包するプレイヤ数が半分ずつになるように領域を横に分割する．
(5) 検知した全ての部分領域に対して (2)〜(4) を行う．

2.5.4 改良型動的負荷分散手法のアルゴリズム

本項では，以下に改良型動的領域分割手法のアルゴリズムを示す．

(1) プレイヤの移動 (または新規参入) によって，部分領域内のプレイヤ数が分割閾値を超える，または結合閾値を下回った場合，その部分領域を担当する管理ノードは，自身が管理する部分領域に接する部分領域の管理ノードに分割メッセージを送信する．
(2) メッセージを受け取った各管理ノードは，自身が管理するプレイヤの座標データと部分領域の範囲をメッセージを送ってきた管理ノードへ送信する．
(3) プレイヤの座標データなどを受け取った管理ノードは，自身が管理する部分領域を中心に分割対象領域を作成する．
(4) 分割対象領域に対して分割判定を行う．
(5) 分割判定で決定した分割開始マスと分割方向を元に改良型分割を行う．
(6) 領域矯正を行う．
(7) 分割対象領域を作成した管理ノードは新たに分割された各部分領域に管理ノードを割り当て，新たな部分領域のデータとその部分領域に属するプレイヤノードのデータを送信する．
(8) 新たに割り当てられた各管理ノードは自分の管理している部分領域内のプレイヤに分割後の領域データを送信する．
(9) プレイヤに領域データ送信後，再びプレイヤの移動や新規参入が繰り返し行われる．各部分領域内のプレイヤ数が分割閾値を超える，または結合閾値を下回った場合，(1)〜

(8) の操作を繰り返し行う．

3. 性能評価

提案法である基本型動的領域分割法 (以下，基本型) と改良型動的領域分割法 (以下，改良型) の有効性を示すため，コンピュータシミュレーションによる性能評価を行う．シミュレーションでは，第 2 章で説明した四分木手法 (以下，四分木) による領域分割法と，提案法である基本型，改良型の 3 つをそれぞれ比較検討し，それらの性能について評価する．

3.1 シミュレーションモデル

シミュレーションに際し，仮想空間，閾値，プレイヤ，相互干渉領域，通信間隔，負荷の設定を以下に示す．

(1) 仮想空間

仮想空間は 64 × 64 マスの正方形の 2 次元平面とし，領域の最小単位 1 マスとする．また，現実世界で人が不快なく存在するためには，ある程度空間が必要である．その快適な空間のことをパーソナル・スペース[9] とよび，一般的に 45〜75cm とされている．よって仮想空間内の 1 マスは 60cm と定義する．

(2) 閾値

分割閾値を 30，結合閾値を 10，領域プレイヤ数を 20 とする．

(3) プレイヤ

- MMORPG では通常，グループで集まるため，プレイヤは一様に分布しにくい．このことから，プレイヤの振る舞いを完全なランダムではなく，ある程度の動作パターンを持たせるために，プレイヤノードの新規参入及び，移動が集中する部分 (以下，ホットスポット) を 5ヶ所設定する．
- タイムスロット (1[s]) 毎に移動できる範囲は，人間の歩く速度が秒速約 1 mなので 2 マス (120cm) とする．また，プレイヤの移動は，確率に従い上下左右の移動または移動しないの 5 パターンの行動をランダムに 2 回行う．各々の行動の確率は 2/11 である．また，1/11 の確率であらかじめプレイヤに決められたホットスポットへ向かう方向に移動するようにする．
- プレイヤの参入及び離脱は，増加時と減少時でデータに関係性を持たせるために，増加時は毎秒 2 人参入し，1 人離脱，減少時は参入無しの 1 人離脱とする．
- ポワソン分布に従って，離脱するプレイヤノードの新規参入間隔と離脱間隔をそれぞれ同じ値 (3[s]) に設定することで，プレイヤノード数を任意の値に収束させている．
- MMORPG ではリアルタイム性が重視されるので，管理ノードとプレイヤノード間の通信，および管理ノード間の通信をタイムスロット毎に行うものとする．

表 1 シミュレーションのパラメータ

仮想空間全体	64 × 64[マス]
結合閾値	10
分割閾値	30
領域プレイヤ数	20
移動速度	2[/s]
プレイヤ増加時の新規参入人数	2[/s]
プレイヤ増加時の離脱人数	1[/s]
プレイヤ減少時の新規参入人数	0[/s]
プレイヤ減少時の離脱人数	1[/s]
相互干渉領域	2[マス]
分割管理ノードと管理ノード間の通信間隔	1[s]
管理ノード間の通信間隔	1[s]
管理ノードとプレイヤノード間の通信間隔	1[s]
領域情報を書き換える処理負荷	2[bit]
プレイヤの座標情報を扱う通信負荷	2[byte]
部分領域の領域情報を扱う負荷	32[byte]

(4) 相互干渉領域

相互干渉領域でプレイヤ情報を管理ノード間で通信するためには，管理ノード間の通信中にプレイヤが相互干渉領域内にいる必要がある．管理ノード間の通信間隔が 1[s]，プレイヤ移動速度が 2[マス/s] なので，相互干渉領域を 2 マスとする．

本シミュレーションのパラメータを表 1 に示す．

本シミュレーションの評価項目を以下に示す．

(1) 領域数

仮想空間全体の領域数を測定する．

(2) 総負荷

本シミュレーション中に発生する，プレイヤ情報やプレイヤの位置を管理する際に発生する管理負荷，領域情報を計算する際に発生する処理負荷，領域を分割する際に発生する分割負荷，相互干渉領域による負荷の総計を測定する．相互干渉領域による負荷は，(3) で説明する．

(3) 相互干渉領域による負荷

相互干渉領域内にプレイヤが存在する場合に発生する負荷の合計を測定する．

これらの項目を比較検討するために，プレイヤノード数を 100 にセットした後，毎秒 1 ずつ増加させて，200〜2000 まで 100 増加毎に測定する．また，プレイヤノード数を 2000 にセットした後，毎秒 1 人ずつ減少させて，1900〜200 まで 100 減少毎に測定する．各シミュレーションでは，プレイヤノードは設定したホットスポットを中心に新規参入し，プレイヤノード数が 100 人となった時点からシミュレーションを開始する．シミュレーションを 100 回行い，その平均をとったものを評価する．

3.2 シミュレーション結果

まず，各手法におけるプレイヤ数の増減に対する領域数の変化を図 3，4 に示す．図 3 より，提案分割手法は，四分木に対して最大半分程度まで領域数を削減することに成功している．また，図 4 より，四分木はプレイヤ数が減少しても小さくなった領域を統合できないために，領域数を削減できていないことがわかる．それに対して，提案分割手法は共に無駄な領域の削減や小さくなった領域の統合が可能なため，プレイヤの減少に対応することができている．また，プレイヤ数に関係なく，基本型に比べ改良型の方が領域数が少ないこともわかる．

次に，プレイヤ数に対する総負荷を図 5，6 に示す．図 5，6 より，改良型が最も総負荷を抑えれることがわかる．基本型は四分木に比べ領域数を抑えることができたが，総負荷は四分木以上の値になった．基本型は領域分割後に領域の矯正を行わないので，領域分割後小さな領域や歪な形の領域が残ってしまう．また，プレイヤ減少時に領域を統合する負荷が発生する．この 2 つが主な原因となり，総負荷が増加したと考えられる．

次に，相互干渉領域による負荷を図 7，8 を示す．正方形に近い形で分割されていれば，相互干渉領域の負荷が低く抑えられると思われる．図 7，8 より，改良型の相互干渉負荷が最も軽減できたことがわかる．これは，領域を正方形に近い形に分割していくことで，相互干渉領域の範囲を少ない範囲に抑えれたことが要因と考えられる．また，図 8 の四分木は領域数を減らすことができないため，必要以上の相互干渉領域が残ってしまい，提案法に比べ負荷の減少速度が遅くなっていることがわかる．

4. おわりに

本研究では，大規模仮想空間における動的領域分割手法を提案した．また，シミュレーションにユーザの密集が起こりやすいホットスポットや，データの整合性を維持するために相互干渉領域を導入することで，より実際の MMORPG に近い環境でシミュレーション

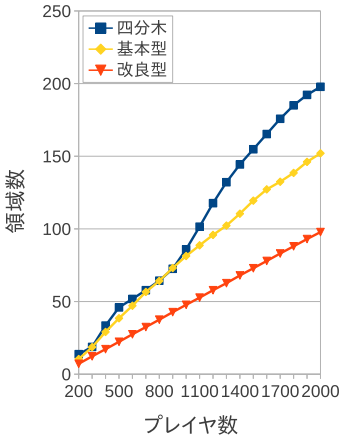


図 3 領域数 (プレイヤ増加時)
Fig. 3 Number of regions(Increasing players.)

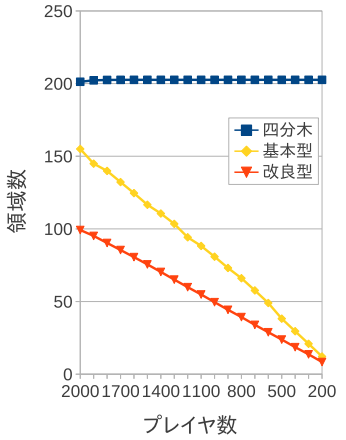


図 4 領域数 (プレイヤ減少時)
Fig. 4 Number of regions(Decreasing players.)

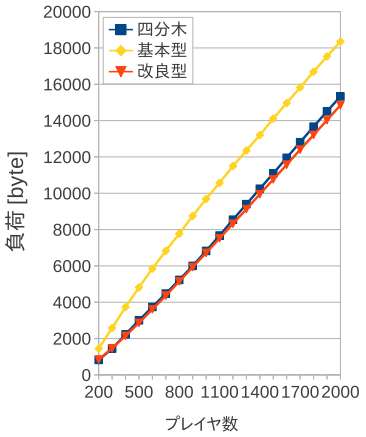


図 5 総負荷 (プレイヤ増加時)
Fig. 5 The total load(Increasing players.)

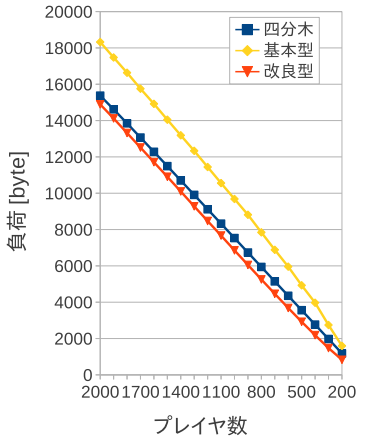


図 6 総負荷 (プレイヤ減少時)
Fig. 6 The total load(Decreasing players.)

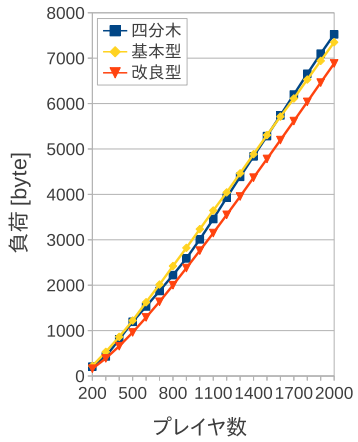


図 7 相互干渉負荷 (プレイヤ増加時)
Fig. 7 Mutual interference Load(Increasing players.)

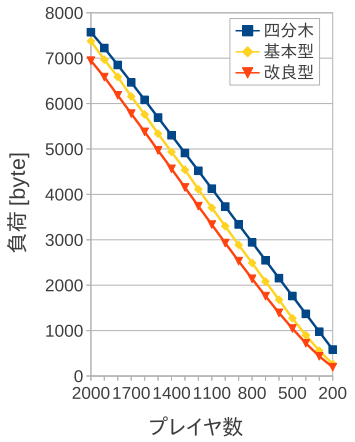


図 8 相互干渉負荷 (プレイヤ減少時)
Fig. 8 Mutual interference Load(Decreasing players.)

し，どの程度負荷を軽減できるかを評価した．また，提案法である改良型動的領域分割手法は，動的な領域分割を行うだけでなく，歪な形の領域を少なくすることにより負荷の軽減ができることを示した．結果より，プレイヤの移動を考慮した動的な領域分割をすることで，プレイヤの密集や過疎に対応した負荷分散ができることを示した．

現在，管理ノードは離脱しないものとして，シミュレーション実験を行っている．管理ノードの離脱に関しては，データの損失を避けるため，バックアップを保存するプレイヤを選ばなければならない．管理ノードの離脱は今後の課題である．

参 考 文 献

1) Entertainment, B.: World of warcraft, Blizzard Entertainment (online), available from ⟨http://www.worldofwarcraft.com/index.xml/⟩ (accessed 2012-04-19).
2) PhilippeDavid, A.V.: Improving scalability in MMOGs - ScalaMo: a new architecture -, Cite SeerX (online), available from ⟨http://search.yahoo.co.jp/search?p=Improving+scalability+in+MMOGs+-+ ScalaMo%3A+a+new+architecture+-&search.x=1&fr= top_ga1_sa&tid=top_ga1_sa&ei=UTF-8&aq=&oq=⟩ (accessed 2012-04-19).
3) 遠藤　伶，高木健士，北　望，重野　寛：MMO 仮想環境におけるユーザ密度の変化に対応した負荷分散手法 (セッション B-8:P2P・オーバーレイネットワーク (2))，情報処理学会研究報告 (CSEC)， No.21, pp.213–218 (2008).
4) R.Thawonmas, M.K. and Chen, K.: Detection of landmarks for clustering of online-game players, *International Journal of Virtual Reality*, Vol.6, No.3, pp.11–16 (2007).
5) Dewan TanvirAhmed, S.S.: A Dynamic Area of Interest Management and Collaboration Model for P2P, *Proceedings of the 2008 12th IEEE/ACM International Symposium on Distributed Simulation and Real-Time Applications*, pp.27–34 (2008).
6) 山崎孝裕，金田憲二，大山恵弘，米澤明憲：柔軟性と拡張性を備えた大規模多人数オンラインゲームのための枠組み，SWOPP 武雄 2005，pp.61–66 (2005).
7) 小林基成，鈴木俊博，永田智大，カーンアシック，趙　晩熙：オンラインゲームのための仮想ネットワークの資源割り当て方法の検討，電子情報通信学会技術研究報告. NS, ネットワークシステム，pp.73–78 (2008).
8) MariosAssiotis, V.T.: A Distributed Architecture for MMORPG, *NetGames '06 Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games*, pp.73–78 (2006).
9) Future.net, P. F.T.: Passion For The Future.net, Passion For The Future.net (online), available from ⟨http://www.ringolab.com/note/daiya/archives/001278.html⟩ (accessed 2012-04-19).