モーバイルコンピューティング　8－8
（1999．2．5）

モバイル環境を想定した地図データ操作機構の提案

桜井 鐘治　　奥村 誠司　　撫中 達司　　黒田 正博
三菱電機(株) 情報技術総合研究所

モバイル環境から地図データのアクセスを可能にすることで地図データを使用する利用者の数が増加する．本報告では，モバイル環境で少数の管理者が地図データを更新し多数の一般の利用者が地図データを参照する形態において，システムにてサポートする利用者の数の増加を目的とした地図データ操作機構を提案する．本機構では，管理者が更新するサーバと利用者が参照するサーバとの間でデータ同期機構を利用して地図データの同期を行う．各サーバでそれぞれに適した多次元データ操作アルゴリズムを用いることで，サーバがサポートする管理者及び利用者の数を増加させることができる．また本機構では，利用者向けのサーバを利用者の増加に合わせて追加することが可能で，これによりシステムにてサポートする利用者の数をスケーラブルに増加できる．

Map Handling Architecture for Mobile Environment

Shoji Sakurai, Seiji Okumura, Tatsuji Munaka, Masahiro Kuroda
Information Technology R & D Center, Mitsubishi Electric Corp.

A capability of accessing map data from mobile environment increases the number of users who access map data. This paper proposes a map handling architecture with scalability in an environment where a few operators update map data and other many users read them. This architecture connects two different types of map servers, one is for operators and the other is for users, with our data consistency mechanism to keep the consistency of map data. Using suitable multi dimensional data handling algorithms for each type of servers increases the number of operators or users supported by a server. It makes possible to install additional map servers for users with scalability of system.

1. はじめに

近年の携帯端末の小型化と携帯通信インフラの急速な普及により，屋外においても事業所やオフィスと同じ環境を利用したいという要求が高くなってきている．これにともない無線通信を利用して事業者やオフィスのサーバに接続し必要なデータにアクセスすることが盛んに行われるようになってきている．地図データもこの例外ではない．地図データに関してはむしろ実際にデータを必要とする場所はそれらを管理するサーバがある場所ではなく，屋外の現地であることが普通である．

しかしながら，現地から無線通信を使って地図データにアクセスするには無線通信上のデータ通信に関する不安定さや低い転送速度の問題を克服するとともに，一般の他のデータに比べサイズが大きな地図データを転送し，かつ利用者が満足できる応答時間を確保することが必要である．これらの要求に対して我々は，楽観的データ同期機構[1]上に地図データを対象とするデータ転送システムの開発[2]を行った．このデータ転送システムでは利用者の必要とする領域の地図データのみを抜き出して，データ同期機構を使って利用者の端末に転送する．利用者に地図データを高速に提供するには，サイズが大きくかつ多次元データである地図デ

ータから必要な領域のデータを高速に抜き出す検索アルゴリズムが必要である．地図のような多次元データの検索を高速に行うためのアルゴリズムとしてこれまでにG-tree[3]やR-tree[4]及びR*-tree[5]などが研究されている．

地図データの運用については一般的に，比較的限られた人数の管理者のみが地図データを更新し，管理者比べて多数の一般の利用者がこの地図データを参照する形態が考えられる．さらに，モバイル環境から地図データをアクセスできるようにすることでこの利用者の数は更に増加する．この利用形態では，少数の更新要求と多数の参照要求を同時に処理する必要がある．多次元データ操作アルゴリズムの並列性を高め複数の操作を同時に行うことができるようにR-treeを改良したR-link tree[6]の研究も行われているが，結果としては参照の性能は上がらないことが報告されている．このため，本報告では，前述の少数の管理者と多数の利用者が存在する利用形態において，データ同期機構を用い，システムにてサポートする利用者の数の増加を目的とした地図データ操作機構を提案する．

以下，2章では今回の地図データ操作機構で利用しているデータ同期機構について述べる．3章では，地図データとその利用形態について述べる．4章では地図データを含む多次元データの操作アルゴリズムについて述べる．5章では，データ同期機構と多次元データ操作アルゴリズムを組み合わせた地図データ操作機構を提案し，6章でそのスケーラビリティについて考察する．最後に7章でまとめと今後の課題を示す．

2. データ同期機構

データ同期機構は，複数のサイトでデータレプリカ(以降,レプリカと呼ぶ)を持ち，それらレプリカ間でデータの整合性を保つ機構である．本同期機構では,汎用性のためにJava[7]技術を利用している．本同期機構は，図1に示すように，アプリケーションから見れば，データを保存したり取り出したりするコンテナとしてのSyncStore，そのSyncStoreに入れるレプリカとしてのSynchronizableなJava Serializeableオブジェクト(以降，単にオブジェクトと呼ぶ)からなっている．オブジェクトのSyncStoreへの投入あるいはSyncStoreからの取り出しは,SyncStoreへのput()/get()メソッドで行う．オブジェクトは，Reconcilableサブクラスか Diffableサブクラスの Synchronizableオブジェクトとして表現される．Reconcilableなオブジェクトは，同期を行う相手SyncStore内に存在する同一のオブジェクト識別子を持ったオブジェクトとの間で更新時のタイムスタンプを比較し，相手オブジェクトが自オブジェクトより新しいタイムスタンプを保持している場合には，自オブジェクトを相手オブジェクトで置き換える．一方，Diffableなオブジェクトの場合には,同じく同期を行う相手SyncStore内に存在する同一オブジェクト識別子を持つオブジェクトと更新時のタイムスタンプを比較し，一方のオブジェクトの最新タイムスタンプ値までの更新ログを，他方のオブジェクトに適応する．なお，ここで更新ログはオブジェクトに対して行われた変更内容を表す．ReconcilableとDiffableのいずれの場合にも実際に送られるオブジェクトに関するデータはデータ同期機構上のアプリケーションが定義する．

SyncStore対の同期に際してはデータの転送は実際にはSynchronizerを使って行う．Synchronizerは，無線通信やファイヤウォールの有無などに合わせて適したものを選択することができる．

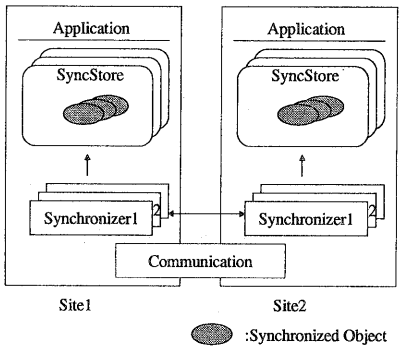

図1 データ同期機構の構成
Fig.1 A structure of data consistency mechanism

3. 地図データの特性

計算機で利用可能な電子化された地図(以降，単に地図とする)には，基本的にラスタ地図とベクタ地図の二種類が存在する．以降では，地図データとして屋外での修正が容易なベクタ地図のみを対象とする．ベクタ地図は2次元あるいは3次元の数値データの配列からなる多次元ベクトルデータにより表されるポリゴンを基本の部品とする．地理上の地形や設備・施設は地図部品としてポリゴンにより地図上に示される．同じ属性の地図部品が集まって地図レイヤを形成する．地図レイヤとしては道路レイヤや河川レイヤ，鉄道レイヤなどの地図レイヤがお互いに独立して存在する．利用者はこれら複数のレイヤの中から必要な地図レイヤのみを選択しこれらを重ねて必要な情報を含む地図を作

成し利用する．

利用者が地図を利用する場合には，そのアクセスに次の特性がある．利用者が地図を利用する場合には，まず大まかに地域を特定するために主だった地形やランドマークとなる建物のみが含まれる縮尺の大きな広域地図を利用する．次にこの広域地図に対して利用者がより詳細な情報を必要とする地域を含む場所を指定して縮尺の小さな詳細地図を利用する．この操作を繰り返して利用者は必要とする縮尺の地図を入手する．広域地図を構成する地図レイヤは，単位面積当たりのデータサイズが比較的小さいが，詳細地図を構成する地図レイヤは，単位面積当たりのデータサイズが大きくなる．利用者は各地図レイヤの部分領域にアクセスするが，地図が詳細になればなるほど利用者が必要とする領域はその一部分に限られる．このため，サイズの大きな地図データほどより狭い領域を指定してアクセスすることになる．

我々はこの地図利用時の特性を利用してモバイル環境から地図データをアクセスすることを可能にするデータ転送システムを開発した[2]．本データ転送システムでは，2章で述べたデータ同期機構を用いている．これにより，モバイル環境における通信の不安定さや低いデータ転送速度に対する解決を図っている．また，本データ転送システムでは，地図利用時の特性を利用し，データ同期機構上のアプリケーション層で転送データ削減と応答時間の短縮を行っている．具体的には，地図サーバ側で利用者が利用するモバイル端末から利用者が必要とする地図上の位置と地図レイヤ名及び範囲を無線通信を介して送信し，これに対して必要な範囲の地図データのみを取り出して，レイヤと領域により転送データに優先順位を付けてモバイル端末に送信する．このように，利用者の必要とする範囲の地図データを高速に提供するには，サイズが大きくかつ多次元データである地図データから必要な領域のデータを高速に抜き出す検索アルゴリズムが必要である．

地図データの運用については一般的に，比較的限られた人数の管理者のみが地図データを更新し，管理者に比べて多数の一般の利用者がこの地図データを参照する形態が考えられる．また地図レイヤ毎に管理者が異なり，各地図レイヤ毎に少数の管理者が各レイヤを管理することも考えられる．このような利用形態では，少数の管理者からの更新要求と管理者以外の多数の利用者からの参照要求を同時に処理する必要がある．先に述べたデータ転送システムを使ってモバイル環境からも地図データをアクセスできるようにすることでこの利用者の数は更に増加する．これにともない地図データを参照する回数も増加する．

4. 地図データアクセスの高速化手法

地図に代表される多次元データに高速にアクセスするための手法として多次元データ操作アルゴリズムがある．多次元データ操作アルゴリズムでは，地図レイヤを構成する個々の地図部品を minimum bounding rectangle(以降では，単に MBR とする)により示し，これを用いてデータ木を構成し検索を高速化する．MBR はその周囲が多次元の軸に平行で地図部品をその内部に含む最小の多次元長方形である．図 2に，XとYの2次元座標空間上で地図部品であるポリゴンとMBR の関係を示す．地図部品を囲む長方形の点線がMBR である．MBR には実際には地図部品が含まない領域も含むため，地図部品の最も基本的な情報である位置情報と各軸についての範囲のみを持つように近似したものである．

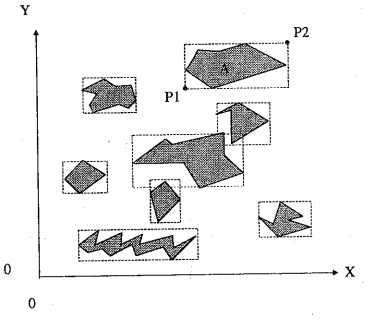

図 2 ポリゴンとMBR

Fig.2 Some polygons and corresponding MBRs

多次元データ操作アルゴリズムには，多次元の値を持つ点データを操作することができる point access methods(PAMs)と，点データだけでなくポリゴンのように範囲を持つ多次元のデータも操作することができる spatial access methods(SAMs)に大別される[8]．

PAMs の代表的なものとしては，G-tree[9]がある．G-tree はn次元の属性を持つ点を含む領域について木構造を構築し，これを使って点の検索を高速に行うアルゴリズムである．G-tree では，各属性の値を 0 から 1 の間の値に変換した値を用い，多次元属性を持つデータを，0 から 1 までの範囲を持つ n 次元単位ハイパーキューブ内の n 次元の属性を持つ点として取り扱う．G-tree は n 次元の単位ハイパーキューブを部分領域(これもハイパーキューブ)に分割し，この部分領域を葉に含む木構造を構築する．1つの領域に含まれる点の数があらかじめ決めておいた上限を越えた場合には，ハイパーキューブである部分領域をn次元の何れかの

軸に対し二分割する．n 次元の軸にはあらかじめ順番を決めておき，最初の二分割は 1 番目の軸について行い，次の分割は 2 番目の軸で，その次は 3 番目の軸と，分割を n 番目の軸まで繰り返す．n 番目の軸での二分割の次には，また 1 番目の軸からの二分割を繰り返す．分割により生成される各部分領域は，バイナリストリングと呼ばれる 0 と 1 からなる文字列により一意に示すことができる．点の最初の軸についての値が 0.5 以下の場合には点は部分領域 0 に属し，点の最初の軸についての値が 0.5 より大きい場合には部分領域 1 に属する．バイナリストリング S で示される領域 R が i 番目の軸について二つの部分領域に分割される場合には，分割する軸に関して小さい値を持つ部分領域を"S"0 で，大きい値を持つ部分領域を"S"1 で示す．

このようなバイナリストリングには絶対的な順序がある．ストリング S1 の最初のビット(MSB)がストリング S2 の最初のビットより小さいか，S1 と S2 の最初の k ビットが同じで S1 の最初の k+1 ビットが S2 の最初の k+1 ビットより小さい場合には，S1<S2 となる．ここで，null ストリングは他のどんな非 null ストリングよりも小さいと定義する．G-tree の葉ノードには，領域を示すバイナリストリング S と領域に含む点を格納するページ PG へのポインタ P の組(S,P)を含む．葉ノード以外のノードは，$(P_0;S_1,P_1;S_2,P_2;\cdots;S_m,P_m)$の形式で示す．$S_i$ はバイナリストリングで，P_i は S_i より大きいか等しいが，S_{i+1} よりは小さい全てのバイナリストリングを含むノードを指すポインタである．P0 は S1 より小さいストリングを含む．

例として，{00,0100,0101,011,01}の 5 つの領域を含む G-tree を図 3に示す．

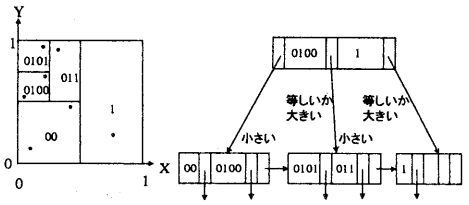

図 3 G-tree の構造
Fig. 3 Structure of G-tree

図 3の例は，葉ノードには 2 つまでの上限を 2，葉ノード以外のノードには中間ノードあるいは葉ノードへのポインタを 3 つまで含むとしたものである．

一方，SAMs の代表的なものとしては，R-tree[4]がある．R*-tree は，B+-tree の概念を SAMs に適用したバランス木で，地図部品の MBR はデータ木の葉ノードに格納される．データ木の中間ノードには，データの組(R,p)を格納する．ここで，p は子ノードを参照するポインタであり，R は中間ノードが持つ全ての子ノードの領域を合わせた領域に対する MBR である．1 つの地図部品はデータ木のどこか 1 つの葉のみに格納されるが，同じレベルのノードの持つ MBR が重なることは許されている．地図部品を格納するノードを決定する際にはデータ木を下向きに探索するが，このときノード間の重なりが最小になるようにノードを選択する．図 4に R-tree の構造とそれに対応するデータ空間の分割を示す．

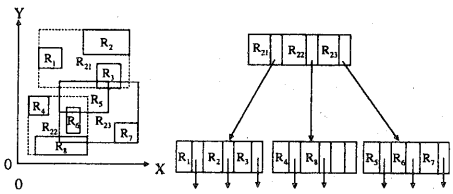

図 4 R-tree の構造
Fig. 4 Structure of R-tree

点データしか扱えない PAMs に対しては，transformation と呼ばれる手法[8]により範囲を持つ多次元のデータも操作できるように拡張することができる．transformation では，n 次元ポリゴンの MBR の対角をなす 2 点の座標を合わせて 2n 次元の点データとして扱うことにより PAMs 上でのポリゴンデータの操作を可能にする．例えば図 2の二次元の地図データを考えた場合には，地図部品 A の MBR の左下の点と右上の点を座標をそれぞれ，P1(l, b)と P2(r, t)とすると，P1 と P2 を合わせた 4 次元の点データ P3(l, b, r, t)として地図部品 A を扱う．

3章で述べた利用形態では，少数の更新要求と多数の参照要求を同時に処理できることが必要である．

G-tree や R-tree では，データ木を構成するノードに含む要素数の上限を変更することにより，データ操作アルゴリズムの性格を変えることができる．

図 5に R*-tree[5]についてノードの要素数の上限を変えた場合の挿入と検索の性能測定データを示す．R*-tree は，R-tree を改良したもので，地図部品を格納すべきノードを選択する際に，ノードが中間ノード

の場合には同じレベルのノード間の重なりが最小になるようにノードを選択し，葉ノードの場合には葉ノードの領域が最小になるようにノードを選択する．今回の測定では更新は行っていないが，R*-tree では，データを更新した際には，要素の削除と再挿入を行うため，データの更新は挿入と同じ傾向があるとみなすことができる．この R*-tree におけるノード内の要素数の上限値に対する挿入及び検索の変化の測定は，PentiumII 200Mhz，メモリ 128MB の PC 上で，OS に WindowsNT4.0 を用いて行ったものである．なお，今回はデータ同期機構との親和性を考え Java により測定プログラムを作成し，JDK1.2 を用いて実行した．測定に用いた地図データの地図部品数は，50465 である．

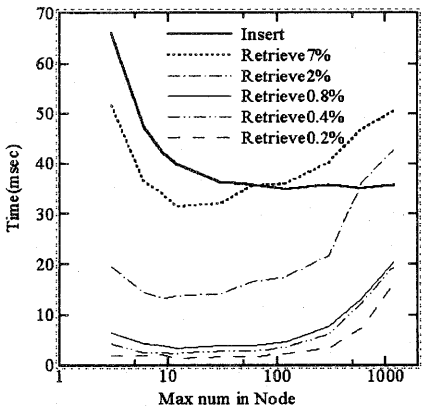

図 5 R*-tree の性格
Fig.5 Characteristic of R*-tree

図 5が示すように，ノードに含まれる要素数の上限値を大きくした場合には，データの挿入時間は短くなるが，逆にデータの検索時間が長くなる．一方，ノードに含まれる要素数の上限を小さくした場合には，データの検索時間は短くなるが，逆にデータの挿入時間は長くなる．このことは，データの挿入時にはデータを挿入すべきノードを特定する必要があるが，要素数の上限値が小さい場合にはノードが多数あり領域が細分化されているため，所定のノードを特定するのに時間を要し，逆に要素数の上限値が大きい場合に挿入時間が短いのは，ノードが少なく所定のノードを特定する時間が少なくて済むことによる．また，要素数の上限値が小さい場合には，ノードのオーバフローの発生頻度が高くなることも挿入に必要な時間を押し上げている．要素数の上限値を大きくしても，挿入に要する時間の漸近線が 0 とはなっていないが，挿入についてはメモリの割り当てが必要でありこのための処理時間が最低限必要なためと予想される．更新ついては，メモリ割り当てが必要でないため，挿入のカーブを 0 が漸近線になるように下にスライドさせた結果になることが期待される．データの検索時には，検索領域と交わりを持つノードについては，ノードの持つ個々の要素について検索条件に一致するかをチェックすることが必要である．要素数の上限値を大きくした場合に検索時間が長くなるのは，このチェックのために要する時間が長くなることによる．また，要素数の上限を小さくした場合にも検索時間が長くなるのは，R*-tree ではデータ木を下向きに探索していく場合に検索条件に一致する領域を持つノードがあるレベルで必ず一つとは限らず複数になる場合があり，この場合にその個々についてさらにその要素をチェックする必要があり，この処理が増加することによる．

一方，多次元データ操作アルゴリズムの並列性を高め複数の操作を同時に行うことができるように R*-tree を改良した R-link tree[9]の研究も行われている．

R-link tree は R-tree を改良し，同じレベルのノードに右向きリンクを導入し複数の更新が並行して行えるようにしたものである．

R-link tree では，同じレベルのノードを右向きリンクでつなぐことに加えて，logical sequence number(LSNs)と呼ばれる単調増加する数値を仮想的なタイプスタンプ値として用いる．図 6に R-link tree の構造を示す．

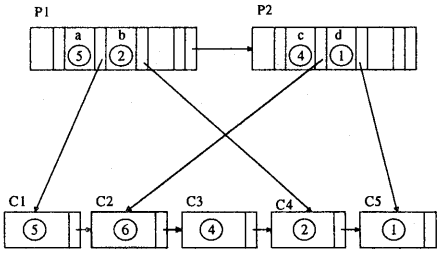

図 6 R-link tree の構造
Fig.6 Structure of R-link tree

R-link tree の各ノードは MBR と子ノードへのリンクと子ノードが持っていると期待される LSN を保持している．図 6において丸で囲まれた数値が LSN を示す．R-link tree では挿入によるノードのオーバフローでノードを分割する際には，新しくノードを割り当てて右向きリンクが元のノードを指すようにする．新しいノードには新たに LSN が割り当てられる．検索時に親ノードの保持する子ノードの LSN と実際に子

ノードの持つ LSN が一致しない場合には，ノードの分割が行われたことが検出できる．この場合には，親ノードが保持する子ノードの LSN を実際に持つ子ノードまで子ノードの右向きリンクをたどる．一致する LSN を持つ子ノードまでが元の子ノードを分割したものであることが分かる．

ノードへのアクセスは，排他制御を行うために，検索時にはリードロックを，挿入時にはライトロックを使用する． R-link tree では更新時の排他制御のやり方が R-tree と異なる．R-tree では，更新時にはまずデータ木全体の更新ロックを取得する．更新ロックは各ノードのリード／ライトロックとは異なるものである．データ木の更新ロックを取得後，実際に更新が必要なノードのライトロックを取得する．一方，検索はデータ木を下向きにノードを参照する時のみノードにリードロックをかけながら行う．このため，更新に関係のないノードに対する検索は同時に複数実行することができるが，更新は同時には１つしか実行できない．一方 R-link tree では，挿入は挿入すべき葉ノードをまず検索し，次にデータ木を下から上に親ノードのライトロックが取れるまで子ノードのライトロックを保持しながらノードのデータを更新する．これにより複数の検索に加え複数の更新を同時に行うことができるようにしている．

このように R-link tree は挿入の並列実行性を改良したものであるため，R-link tree の評価結果では，挿入に関してのシステムのスケーラビリティは向上するが，参照に関しては変わらないことが報告されている．

5. 地図データ操作機構の提案

我々は本章において，3章で述べた利用形態において，データ同期機構を用い，システムにてサポートする利用者の数の増加を目的とした地図データ操作機構を提案する．

5.1. 基本モデル

図 7に地図データ操作機構の基本モデルを示す．

管理者１は地図サーバ１が管理する地図レイヤに対して更新権限があり更新と参照を行う．一方，利用者２は地図サーバ２が管理する地図レイヤに対して更新権限はなく参照のみを行う．地図サーバ１と地図サーバ２の間ではデータ同期機構を用いて地図レイヤの同期が取られる．従って管理者１が地図サーバ１の地図レイヤに加えた更新はデータ同期に機構により地図サーバ２の地図レイヤに反映される．これにより利用者２は，管理者１が地図レイヤに加えた更新を地図サーバ２の地図レイヤにアクセスすることで参照することができる．

利用者の使用するモバイル端末では地図サーバへのアクセスにデータ同期機構を用いるか，もしくは地図サーバ自体をモバイル端末上に保持する．このような構成を取ることにより，モバイル環境における通信は全てデータ同期機構を介して行われる．

各地図サーバでは地図レイヤに含まれる地図部品を多次元データのデータ木で管理し，地図レイヤのデータへのアクセスを高速化する．特に，地図サーバ１では更新が高速なデータ木を用い，地図サーバ２では検索が高速なデータ木を用いる．このようなデータ木としては，4章で述べたように，１つの多次元データ操作アルゴリズムをノードに含む要素数の上限値を変えることでその性格を変えて使い分けることもできるし，G-tree と R-tree といった全く異なるアルゴリズムをその特長を生かして利用することもできる．

また，基本モデルは，サーバ間の地図レイヤの同期にデータ同期機構を用いており，同期の際の転送データはアプリケーション層で自由に定義できるため，基本モデルでは同期対象を各地図部品内のデータのみとし，木構造のリンク情報は転送データには含めないこととする．また，地図データの同期には Diffable なオブジェクトを用いる．これにより，データ同期時には地図部品の更新ログのみが転送データとして送られる．

管理者１が地図サーバ１の地図レイヤを更新した後に，地図サーバ１と地図サーバ２の間でデータの同期を行うと，更新された地図部品の更新ログのみが地図サーバ１から地図サーバ２に転送され地図サーバ２の地図レイヤに適用される．地図サーバ２では同期完了後に，同期処理において MBR に変更があった地図部品についてのみデータ木の更新を行う．

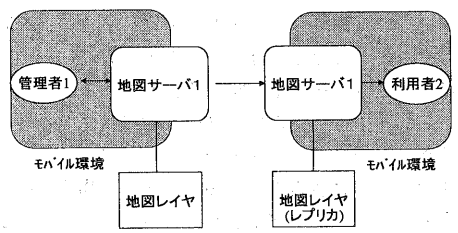

図 7 基本モデルの構成
Fig.7 Structure of basic model

5.2. システムモデル

図 8に前述の基本モデルを複数の地図レイヤに対して

適応した場合の地図データ操作機構のシステムモデルを示す．

管理者Aと管理者Bは地図レイヤaに対して更新権限があり更新を行うが，地図レイヤbに対しては更新権限はなく参照のみを行うものとする．逆に，管理者Cと管理者Dは地図レイヤaに対しては更新権限はなく参照のみであるが，地図レイヤbに対しては更新権限があり更新を行うものとする．さらに，利用者E以降のその他多数の利用者は，地図レイヤaに対しても地図レイヤbに対しても更新権限がなく，参照のみを行うものとする．

管理者Aは，地図レイヤaを介して管理者Bと最新情報を共有することができる．管理者Aが地図レイヤaに対して更新を行う際には，この更新は地図サーバα上の地図レイヤaのデータ木に対して行われる．管理者Bがこのデータにアクセスする際には地図サーバαの地図レイヤaのデータ木を使ってデータの検索を行い結果を利用する．このため，更新が加えられた最新のデータにアクセスすることができる．一方，管理者Aまたは管理者Bは，地図レイヤaに加えられた更新を利用者が参照するデータに反映したいタイミングで，明示的に地図レイヤaに関して他の地図サーバとのデータ同期を起動する．これにより地図サーバβと地図サーバγ上の地図レイヤaのデータが最新のものに更新される．

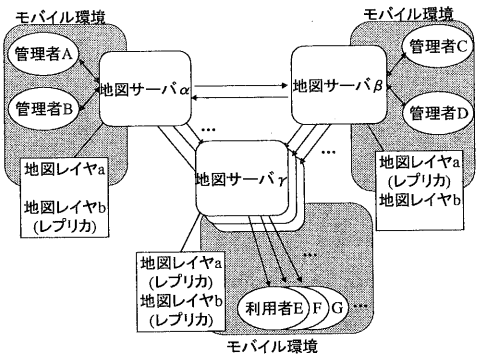

図 8 システムモデルの構成

Fig.8 Structure of system model

同期が起動された場合には，この時点でのローカルなデータと相手が保持しているデータとの差分にあたる更新ログのみが実際にデータとして転送される．これを受けた側では，これらの差分が1つのイベントにより一連の更新ログとして通知されるため，中間的な修正を省いて最終的な状態が一致するように更新ログを適用しデータの修正を行う．同期は更新を加えた管理者が更新を他の利用者がアクセスする地図データに反映したいタイミングでのみ明示的に起動するため，実際のデータ更新の頻度を少なくすることができる．

管理者Cは，地図データbに関して上記と同じように更新を行い管理者Dと最新情報を共有することができ，その他の利用者には更新を反映したいタイミングでのみ更新データを提供することが可能となる．

6. 考察

本章では，5章で提案したシステムアーキテクチャを用いた場合のシステムのスケーラビリティについて考察する．

管理者がアクセスする地図サーバα及び地図サーバβにおいいては，更新に適した多次元データ操作アルゴリズムを使うことで，これらの地図サーバにおいて地図データの更新を行える管理者の数を増加させることができる．

一方，地図サーバγでは，検索を高速化した多次元データ操作アルゴリズムを使うことでこの地図サーバがサポートする利用者の数を増加させることができる．このとき，検索を高速化することで逆に更新の速度が低下し，これにより利用者が検索にデータ木を利用することができる割合が減少することが予想されるが，このことはデータ同期機構を用いることで実際に行われる更新の回数が削減されることにより緩和される．

また，地図サーバγと同様の利用者に地図データを提供する地図サーバを追加することでさらにシステム全体でサポートする利用者の数を増加することができる．この場合には，地図サーバα及び地図サーバβが同期を取る地図サーバが増加する．このため，今度はこれらの地図サーバにおいて同期のためにデータ同期機構がデータ木をアクセスする回数が増え，このアクセスの回数の増加によりシステムに追加することができる利用者向け地図サーバの数が抑えられることが考えられる．実際の地図サーバの同期処理においては，データ同期機構が更新ログを確定するためにタイムスタンプの交換後にデータ木をロックし参照を行う．この処理は地図サーバγに相当する全ての地図サーバの地図レイヤが前回の同期で全て一致していれば，同期を取る地図サーバ数の増加に関係なく1回で良い．前回の同期時に通信エラーなどの何らかの障害の発生により同期が正常に終了しなかった地図サーバが存在する場合にのみ，これについてのみ別途同じ処理を行う．このため，データ同期機構がデータ木をアクセスすることがボトルネックとなって追加できる地図サーバの数が抑えられることはない．従って，利用者の参照する地図サーバの数を増やすことで，システムのサポー

トする利用者の数をスケーラブルに増加させることができる.

7. まとめ

本報告では，モバイル環境において限られた数の管理者が地図データを更新しその他多数の一般利用者が地図データを参照する利用形態において，システムにてサポートする利用者の数の増加を目的とした地図データ操作機構を提案した.本機構では管理者がデータを更新する地図サーバと利用者がデータを参照する地図サーバとを用い，この間でデータ同期機構を使用して地図データの同期を行う．管理者がデータを更新する地図サーバでは，更新に適した多次元データ操作アルゴリズムを使うことで，データの更新を行える管理者の数を増加させることができる．また，利用者が参照する地図サーバでは検索を高速化した多次元データ操作アルゴリズムを使うことで，地図サーバを参照する利用者数を増加させることができる．さらに，本機構では利用者が参照する地図サーバを利用者の増加に合わせて追加することによりシステムにてサポートする利用者の数をスケーラブルに増加させることができる.

実際に利用者の増加に合わせて利用者が参照する地図サーバをシステムに追加する際には，競合するリソースとして，本報告において考察した地図データのデータ木のアクセス以外にも，他の地図サーバとの同期に要する通信や CPU 消費が考えられ，これらのリソースの競合がシステムのスケーラビリティに与える影響を明らかにすることが今後の課題としてあげられる.

参考文献

1) Masahiro Kuroda, Jun Inoue, Takashi Watanabe, Tadanori Mizuno, “Wide-area messaging system on nomadic data consistency model”, ICOIN-13, Cheju Island, Korea, Jan. 1997.
2) 桜井,下間, “無線インフラを使ったマルチメディアデータ転送システムの開発” ,情報処理学会モバイルコンピューティング研究会,Vol.98,No.13, pp.51-56, 1998 年 2 月.
3) Akhil Kumar,"G-tree: A New Data Structure for Organizing Multidimensional Data", IEEE Transaction on Knowledge Data Engineering,Vol.6, No.2, APRIL 1994.
4) A. Guttman, “R-Trees: A Dynamic Index Structure for Spatial Searching”, Proc. 1984 ACM SIGMOD Conf., pp.47-57.
5) Nobert Beckmann,H.-P. Kriegel,R.Schneider and B.Seeger, "The R*-tree: An Efficient and Robust Access Method for Point and Rectangles", Proc.1990 ACM SIGMOD Conf.,pp.322-311.
6) M. Kornacker,D Banks, "High-Concurrency Locking in R-Trees", Prof. of the 21st VLDB Conf. , Zuerich Switzerland, 1995.
7) James Gosling, et al, “The Java Language Specification”, Adison-Wesley, Menlo Park, California, August 1996.
8) Bernhard Seeger, H.P. Kriegel,"Techniques for Design and Implementation of Efficient Spatial Access Methods”, Prof. of the 14th VLDB Conf., Los Angels, California, 1988.