P2P を利用した地球物理データのネットワーク横断検索・共有システムの実現に向けて

佐藤　麻美†　　渡辺知恵美††

† お茶の水女子大学理学部情報科学科　〒 112-8610 東京都文京区大塚 2-1-1
E-mail: †asamin@db.is.ocha.ac.jp, ††chiemi@is.ocha.ac.jp

あらまし　我々は地球流体科学者を対象に，科学者個人が管理するデータのアーカイブサーバを容易に構築することの出来る Ruby on Rails ベースのパッケージ「Gfdnavi」を開発している．これを用いることにより，データの公開から検索，分析，可視化までの一連のサービスを Web 上にて容易に提供することが出来るようになる．本稿では，Gfdnavi を用いて公開される多くのサーバ間の横断検索を実現するための一検討を述べる．本研究ではセンターサーバを設置せず，P2P ネットワークを利用した自律分散的な検索を目指す．Gfdnavi で共通に定義されているリレーショナルスキーマを元に，DHT を利用した絞り込み検索を行うことで結果候補を所有するノードを求め，Gfdnavi の Web サービスを介して該当ノードに問合せを行うことによって複数のノードから検索結果を取得する．

キーワード　科学 DB，P2P，オーバーレイネットワーク

A cross-search mechanism across data archiving servers for earth physical data using P2P network

Asami SATO† and Chiemi WATANABE††

† Department of Information Sciences, Faculty of Science, Ochanomizu University
E-mail: †asamin@db.is.ocha.ac.jp, ††chiemi@is.ocha.ac.jp

Abstract　We are developing a package "Gfdnavi", which can facilitate construction of data archiving servers for the earth fluid scientists. This package is based on "Ruby on Rails", which is a framework for developing database-backed web applications. Gfdnavi provide utility functions, such as metadata extraction tool, highly-functional query interface and data analysis/visualization libraries. When Gfdnavi spread thoughout scientists, a lot of Gfdnavi servers appears on the Web. In this paper, we describe one examination for a cross-search mechanism among a lot of Gfdnavi servers on the Web. The system aims at an autonomous, decentralized retrieval system using the P2P network without setting up a central server. The system is based on PIER [9] which is a cross search mechanism for RDBMSs on P2P network using Distributed Hash Table(DHT), and we improve the methanism for reducing DHT entries and query messages by using Gfdnavi query interaction pattern.

Key words　Scientific DB，P2P，Overlay Network

1. はじめに

近年の地球観測と計算機の進展により大気や水質などの数値データは爆発的に増加している．これらの数値データは一般的に多次元・多量であり解析・可視化して初めて意味を持つ．NASA などはデータセンタを設置し場合によっては数 PB にも及ぶ地球観測データを管理し web 上で公開しており，世界中の科学者がダウンロードし利用できるようになっている．

このような背景の下で科学者個人が管理するデータは飛躍的に増加し，また研究者個人で観測したデータそのものも大きな容量を持っている．そこで効率の良い研究のために気軽に自分の計算機に蓄積されたデータを検索したり，同様の研究をする科学者に観測データを公開したりしたいという要求は高まっているが，個人レベルではストレージの確保や自主的なデータの分類作業など様々な手間が掛かり，二の足を踏む科学者も多い．

そこで我々は地球流体科学者を対象に，科学者個人が管理するデータのアーカイブサーバを容易に構築することの出来る Ruby on Rails ベースのパッケージ「Gfdnavi」を開発している [5], [6], [7]．これを用いることにより，データの公開から検索，分析，可視化までの一連のサービスを Web 上にて容易に

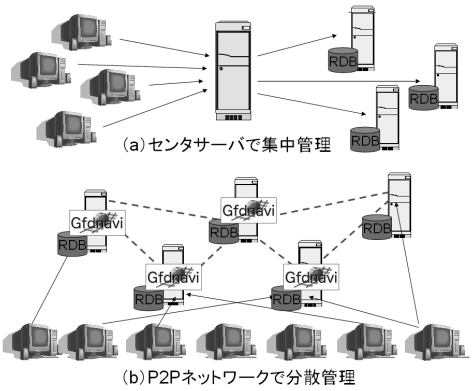

図 1　通信方式の対比

提供することが出来るようになる．

本稿では，Gfdnavi を用いて公開される多くのサーバ間の横断検索を実現するための一検討を述べる．本研究ではセンターサーバを設置せず，P2P ネットワークを利用した自律分散的な検索を目指す．Gfdnavi で共通に定義されているリレーショナルスキーマを元に，DHT を利用した絞込み検索を行うことで結果候補を所有するノードを求め，Gfdnavi の Web サービスを介して該当ノードに問合せを行うことによって複数のノードから検索結果を取得する．

P2P ネットワークにおける検索手法及びリレーショナルデータベース (RDB) の横断検索についてのサーベイを行った結果をもとに考案した我々の構築する検索手法に関して，2 節で本研究の目的である P2P を利用したネットワーク横断検索について，3 節で P2P ネットワークと検索ルーティングの種類や仕組みについて，4 節で P2P ネットワークを用いた RDB 分散検索ソフトウェアについて述べる．5 節でそれらを踏まえた横断検索機能の検討とその問題点，改善手法の提案，6 節でまとめと今後の課題を提示する．

2. P2P を利用した Gfdnavi システムの横断的検索

現在開発中である Gfdnavi は RDB を用いた地球流体データアーカイブサーバを比較的容易に作成することが出来るパッケージである．Gfdnavi が一般配布されれば，多くの研究者によるアーカイブサーバが web 上に出現することになるだろう．その時，個々のサーバでの検索だけでなく，他の公開サーバとの横断的検索ができれば研究者同士の活発な知識共有が期待出来る．

愛媛大学の村田ら [2] は 200 近くのデータ公開サーバの横断的検索をするために中央サーバを運用し各公開サーバより XML 形式によるデータカタログを定期的に収集している．しかし全てのアクセスが中央サーバに集中することから，その維持管理に膨大なコストがかかる (図 1(a)) そこで我々は中央サーバを置かず，各 Gfdnavi サーバ同士が自立的にデータ管理・検索を行う P2P ネットワークを用いた横断的検索を実現する (図 1(b))

実際の利用では Gfdnavi のユーザがある 1 つの Gfdnavi サーバを検索する時「横断検索モード」を選ぶとそのサーバ以外の様々な Gfdnavi サーバも検索する．この際システムでは P2P ネットワークを用いて検索を実行しようと考えている．

3. P2P ネットワーク

P2P とは Peer-to-Peer（端末＝ Peer から Peer へ）の略語であり端末同士の繋がりを意識したオーバーレイネットワークのことである．

クライアント-サーバ形式のネットワークでは，データの検索や管理などはサーバ側に集中するが，P2P ではノード同士の直接のやり取りが基本である．この性質を利用すると各ノード（端末）の持つデータを特定のサーバにアップロードすることなくデータのやり取りをすることも可能になる．これはストレージの節約やデータ公開側の手間の削減に繋がる．

P2P の通信形態はハイブリッド，ピュアなどの区分に分けられる．本研究ではピュア型 P2P ネットワークでのデータ検索・共有を行う．

ピュア P2P ファイル共有ソフトウェアの検索ルーティング方法は大きく二つ，非構造的検索のフラッディング，構造的検索の分散ハッシュテーブル（DHT）を用いた手法に分けられる．

フラッディングは各ノードが P2P ネットワークでつながっているほかのノードへ連鎖的に検索依頼を広める方式である．シンプルな手法であるがネットワーク内の全ノードのデータを検索することができず，その分検索が失敗する可能性も高い．また多くのノードが参加する P2P ネットワークでこの方式を工夫なしに実行すると検索依頼だけで帯域がパンクしてしまうことも考えられる．

DHT はハッシュ関数を通して算出したキーと値の組を格納したハッシュテーブルを P2P ネットワークに参加したノードで分散して管理し，分散ハッシュテーブルを辿って効率的に検索を行う手法である．

ノードが P2P ネットワークに参加すると一意な ID が付与され，ノードが公開するデータに対するハッシュテーブルエントリが生成される．エントリはデータを一意に同定する値のハッシュ値をキーとし，IP アドレス等のデータの所在を値とする 2 項組である．

各ノードは自分の ID と同じ・もしくは近隣のキー値のハッシュテーブルを管理する．またネットワーク上に存在する幾つかの他ノードの所在情報 (スキップリスト) を管理する．検索時は検索したいデータのハッシュ値を算出し，所在を知っているノードの内探しているハッシュ値に最も近い値の ID を持つノードへ検索依頼を送る．ID やキー値の付与の仕方やハッシュテーブルの分散管理法，ルーティングについては Chord, CAN，Tapestry，Kademlia など様々なアルゴリズムがある．

DHT は効率的に全ノードのデータを検索でき，ネットワークへの負荷を分散する効果があるため本研究では DHT を採用する．

4. P2P ネットワークを利用したデータベース検索

RDB の P2P 検索を設計するに当たって，我々は P2P ネットワークを利用したデータベース検索の代表的研究である PeerDB [11] と PIER [9] について分散環境での RDB 検索・共

有の構造を調査した．

PeerDB はフラッディング方式を用いた検索方式であり，一方 PIER は DHT ベースの RDB 分散検索方式である．PeerDB は各ノードのスキーマが異なる場合に対してスキーママッチングを行うことで柔軟に対応しているが，フラッディングの欠点である検索キューの多さや検索結果入手の不確実性は解決できない．Gfdnavi では同一のスキーマを用いることから，DHT のデータ検索の効率性を重視して PIER をベースにすることとした．以下 PIER について簡潔に述べる．

4.1 PIER

PIER は DHT を利用してネットワーク上のノードが持つデータベースを検索するシステムである．各ノード上にあるデータは全て共通のグローバルスキーマで定義されていることを想定する．これはネットワーク監視システムのようにデータの標準フォーマットがある程度定まっているものや，P2P 上の各ノードが同じアプリケーションを利用することを想定しているからである．

具体的な検索方法は，まずノードが持つテーブルの各タップルについて，テーブル名，属性名，属性値をまとめてハッシュ化したものをキー，テーブル名，タップル番号，所持ノードの IP アドレスといったタップルの所在情報（もしくはタップルそのもの）を値としてその組を DHT に分散させておく．

例えば IP アドレスが 123.45.67.89 であるノードのリレーション 社員 (ID，名前，出身地) に (012，佐藤有美，奈良) という挿入タップルがある時，以下のキーと値を分散ハッシュテーブルに登録する．

```
key = hash(" 社員; ID; 012"),
value = (" 123.45.67.89; 社員; 012")
key = hash(" 社員; 名前; 佐藤"),
value = (" 123.45.67.89; 社員; 012")
key = hash(" 社員; 出身地; 奈良"),
value = (" 123.45.67.89; 社員; 012")
```

PIER の問合せ処理では，DHT による選択演算に加え，等結合のための基本アルゴリズムに対称ハッシュ結合とフェッチ結合を提案し，またいくつかの最適化手法を提案している．

前述の社員テーブルと給与 (ID，社員 ID，給与) テーブルを「社員.id=給与.id」という条件で対称ハッシュ結合で結合する場合，結合した後の中間テーブルを Nq とし，結合する属性値を用いてハッシュ値を DHT に登録する．この時，結合する社員テーブルのタップルと給与テーブルのタップルは同じ key 値を持つため同じノードで管理される．そこで各ノードは各々のハッシュテーブルにある同一のハッシュ値のタップルを結合して問合せを発行したノードへ返すことにより結合結果が求められる．

5. Gfdnavi における横断検索機能の検討

本節では 4 節に述べた P2P による横断的 RDB 検索手法を参考に，Gfdnavi における横断検索手法について検討する．今回は P2P による RDB 検索機能開発の初期的試みであることから，2 点の限定条件を設けることにした．

（1） 全てのノードに Gfdnavi がインストールされていることを前提とし，Gfdnavi で定義されている以下のスキーマをグローバルリレーショナルスキーマとして用いる．

```
variables(int id, text name, text path)
spatial_attributes(int id,
                   int variable_id,
                   float longitude_lb,
                   float longitude_rt,
                   float latitude_lb,
                   float latitude_rt)
keyword_attributes(int id,
                   int variable_id,
                   text name,
                   text value)
```

以上のスキーマは本提案手法の説明のため実際の Gfdnavi で用いられているスキーマより簡素化し，本手法で用いる重要なテーブルおよび属性のみを記載している．

variables は多次元配列データセットに対するメタデータであり，実際の生データは path 属性に示されるファイルに保存されている．

spatial_attributes は *variables*（多次元配列データセット）がカバーする空間領域の最小外接矩形を記述しており，(longitude_lb, latitude_lb) が矩形左下の座標，(longitude_rt, latitude_rt) が矩形右上の座標を表している．

keyword_attributes は *variables* に関する空間属性以外の属性であり，属性名 name と属性値 value のペアで記述される．

時間属性も Gfdnavi では重要であるが本手法における扱いは空間属性とほぼ同様であり，本節の例では省略している．

なお，各スキーマの詳細やスキーマ間の関連については [6] を参照されたい．

（2） Gfdnavi の空間検索インタフェース（図 2）で発行される下記の問合せのみを対象とする．

空間インタフェースにて，空間領域 ((lon_lb, lat_lb), (lon_rt, lat_rt)) ，キーワード属性名 keyname ，キーワード属性値 keyvalue が指定された場合に発行される問合せは

```
Select   v.*
From     variables v,
         keyword_attributes k,
         spatial_attributes s
Where    v.id = k.variable_id
  and    v.id = s.variable_id
  and    overlap(
          Rectangle(s.longitude_lb,
                    s.latitude_lb,
                     s.longitude_rt,
                    s.latitude_rt),
          Rectangle(lon_lb, lat_lb,
                    lon_rt, lat_rt)
           )
  and    k.name = key_name
  and    k.value = key_value
```

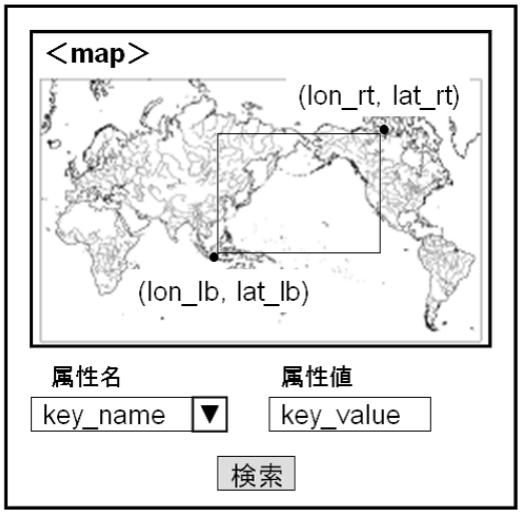

図 2　Gfdnavi の空間検索画面イメージ

である．これを P2P で検索することを考える．

5.1　素　朴　法

Gfdnavi がグローバルリレーショナルスキーマを持つことから，我々は DHT 方式の RDB 検索手法である PIER をベースに Gfdnavi 向けに特化した素朴法を考え，その手法に対する問題点を議論した後，改良を行うこととした．

まず素朴法の概要を示す．前節に述べた問合せにおける処理は以下の手順で行われると考えられる．

（1） *spatial_attributes* に対する問合せ条件を元に　選択演算を行う

（2） *keyword_attributes* に対する問合せ条件を元に　選択演算を行う

（3） *spatial_attributes*, *keyword_attributes* の *variables* への外部キー variable_id を用いて 3 つのテーブルを結合する

（4） *variables* の属性のみを射影して検索結果とする

これに対し，我々は 2 段階の処理を経て問合せを行うこととした．

Step1:　*spatial_attributes* 及び *keyword_attributes* に対する選択演算を DHT を用いて行い，該当タップルを所有するノード (以下該当ノードと呼ぶ) の URL を取得する．

Step2:　該当ノードに対して問合せを発行し，検索結果を収集し表示する．

以下，各 Step における具体的な処理について述べる．

【Step1：DHT による選択演算】

まず *spatial_attributes* および *keyword_attributes* に対する選択演算を DHT を用いて行う．そのために，あらかじめ各ノードは *keyword_attributes* 及び *variables* テーブルの各タップルに対し，

key =hash(”テーブル名；属性名；属性値”)

value =”URL；variable_id”

の規則に従って分散ハッシュテーブルにエントリを登録する．*spatial_attributes* における問合せ条件文は範囲検索によるものであるが，ハッシュ値による検索は完全一致の問合せには対応出来るものの範囲検索等には対応出来ないという問題点がある．そこで我々は図 3 に示すような緯度経度の 2 次元空間を 20 度ずつに分割したブロックを作成し，緯度経度の範囲をブロック番号で表してハッシュテーブルに登録することとした．図 3 の例は，実際には空間検索で指定した 2 点の座標が囲む範囲をブロック番号に変換し，ハッシュテーブルに登録する考え方を示している．

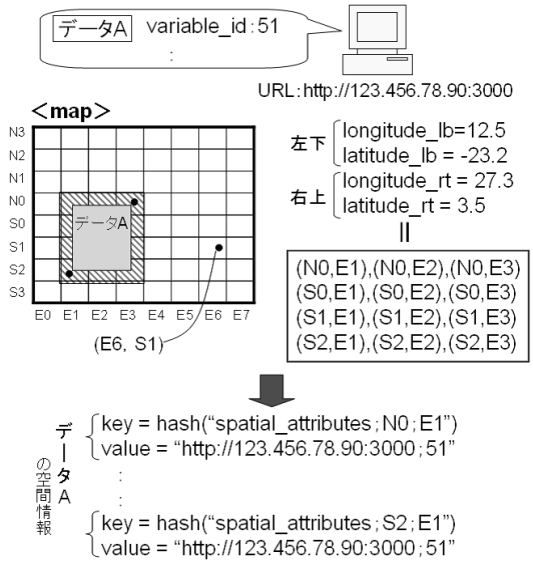

図 3　緯度経度の分割と座標の交換

Step1 では *spatial_attributes* 及び *keyword_attributes* の選択処理をするために，分散ハッシュテーブルから *hash*(”テーブル名；属性名；属性値”) を持つノードの IP アドレス及びタップル ID を取得する．

図 4 は問合せ処理の手順を示したものである．この例では *spatial_attributes* に対する問合せ条件から，*hash*(”*spatial_attributes*; *N*0; *E*1”) , *hash*(”*spatial_attributes*; *N*0; *E*2”)…と図中の＊に関する全てのブロックに対応するキーについて検索を行う．この例ではその結果から，ノード A,B,E,F が図中に示す該当タップルを所持していることが判明する．同様に *keyword_attributes* に対する問合せ条件から *hash*(”*keyword_attributes*; *description*; *ozone*”) というキーを用いて検索をし，その結果からノード A,D,E が図中に示す該当タップルを所持していることが求められる．この問合せでは *spatial_attributes* と *keyword_attributes* の条件は AND 関係であるため，どちらの条件にも当てはまるノード A,E を該当ノードと判断する．

【Step2: 該当ノードへの問合せ】

Step1 で求められた該当ノードに対し問合せを発行し検索結果を収集する．図 4 の例では Step1 にてノード A,E が検索に該当するタップルを持っていることがわかっているので，それらのノードに対し前節に述べた問合せを発行し結果を取得する．

Step2 において PIER が提案している結合方式を用いていないのは，対象としている問合せで用いられている結合処理は外部参照関係にある属性動詞の等結合であるため，ノードをまたいだ結合処理を行う必要がない単純なものだからである．つまりいくつかの選択処理を行うことによって該当ノードの数を絞

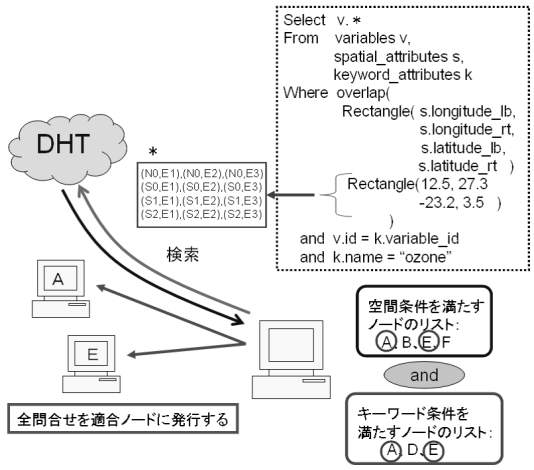

図 4　問合せ処理の流れ

り込み，後は該当ノードに直接検索をする手法をとっている．

また，対象としている問合せの場合には，Step1 で登録したエントリの value 値に variable_id だけでなく，*variables* のタップル全部を含めてしまえばもっと効率よく処理出来る．本手法でそうしない理由は今回対象としている問合せ以外に今後対応していく時に，必ずしも検索結果を value 値に入れられるとは限らないためである．

また，Step1 で variable_id までわかっているのに，Step2 にてもう一度問合せを各ノードで処理するのは，ある 1 ノードに該当タップルが多数あった場合にそれらを id から呼び出すよりも，もう一度問合せを処理してその結果を受け取ったほうが通信の負荷がかからなくなるためである．

5.2　素朴法の問題点

素朴法を用いた場合に，考えられる問題点は以下の 2 点がある．

1　Step1 にて該当ノードがほとんど絞り込めない場合，多くのノードに問合せを発行しなければならない．

ユーザの行う問合せは，とりあえず最初に空間属性だけ指定して何件帰ってくるか様子を見る，というようなことも考えられる．そのような問合せの処理を素朴法で行う場合，ほとんどのノードが該当タップルを持つことになり，ノードが絞られないままほぼすべてのノードに問合せを発行するという事態になりかねない．

2　DHT に登録するエントリ数が膨大になる恐れがある．

これは空間情報の切り分け方や 1 つのデータが持つキーワード属性の多さが原因で起こると予測される．たとえば，ネットワーク上に 100 台の Gfdnavi サーバが存在し，各ノードの所持データの平均を 20 ファイル，データがカバーしている空間範囲のブロック数の平均を 6 ブロックとすると DHT に登録されるエントリ数は *spatial_attributes* テーブルのものだけでも 12000 個に及ぶ．

5.3　改良法の提案

前段の素朴法の問題点を緩和する方法をそれぞれ提案する．

（1）　ノード数が絞り込めないことに関する対処法

Gfdnavi ではユーザが入力した問い合わせの結果が多数ある場合結果をそのままリスト表示するのではなく，そこから更に詳細な絞込検索を促すような検索インタラクションを設計しており [6]，空間属性に対する検索における該当データのグループ化表示，更にグループを細分化しての再表示を GoogleMap 上で行える機能を持っている．

これを利用し問合せを発行するべきノード数が絞り込めない場合への対処案を提案したい．まず素朴法で検索し取得した value にはデータを持っているノードに対応する URL，そのノード内でデータに対応している一意な variable_id の情報がある．それらを用いて検索条件に当てはまるデータが各ノードに幾つあるのか，総じて検索条件に当てはまるデータがネットワーク全体に幾つあるのかを調べることが可能である．

絞込みで該当データが一定数以下でなければ Step2 に進まず，データ数をインタラクティブに表示しユーザに更なる絞込みを促すようにする．十分に該当データが絞られたところで Step2 へ進むことにより該当ノードも絞られることになる．

上記の処理のイメージを表したものが図 5 である．

spatial_attributes の緯度経度情報はブロック化され DHT に登録されているため空間条件を含む絞込みをした場合，検索インタラクションを用い地図上に該当データの数をブロックごとに数字で表示する．

ブロックをクリックすると細分化した分布を表示したり検索条件範囲として指定出来たりするようにする．キーワード条件のみの絞込みをした場合はキーワードの属性名，属性値，該当データ件数を表示する．

ユーザが入力した検索条件から key を生成し DHT に検索をかける．

図 5 では返って来た value から算出したデータ数が一定数以下でなかったので，検索インタフェースの地図上にユーザが設定した空間条件内のブロックごとのデータ数の分布を表示する．

この時実際に DHT に登録したブロックよりも大きな区切りでのデータ分布が表示されているイメージである．こうすることにより DHT に登録したブロックの区切りまで選択細分表示が出来るようになっている．

ここでは丸で囲まれた空間とキーワード条件を検索条件として再設定し再び DHT に検索を掛けている．

このように素朴法の基本的な手法は変えず，問合せを発行するノードの数をある程度減らす働きかけをするために Step1 を繰り返す．十分ノード数が絞り込めた後 Step2 に移行し問合せを発行する．

（2）　エントリ数の多さに関する対処法

素朴法のように *spatial_attributes* の空間情報を個々のデータごとに登録するのではなく，あるブロックをカバーしているデータが幾つあるかを求め，value を”URL；その URL 内でブロックをカバーしているデータの個数” として，つまりブロックごとのデータ分布数の情報を登録する．

図 6 は改良法 2 での空間情報の DHT への登録の流れを示している．あるノードが所持している 4 つのデータについて空間情報をブロックに書き換え，ブロックごとにカバーしているデータの個数を数えると図のようになっている．各ブロックを DHT に登録する時はデータ所持ノードの URL とその個数を value にする．

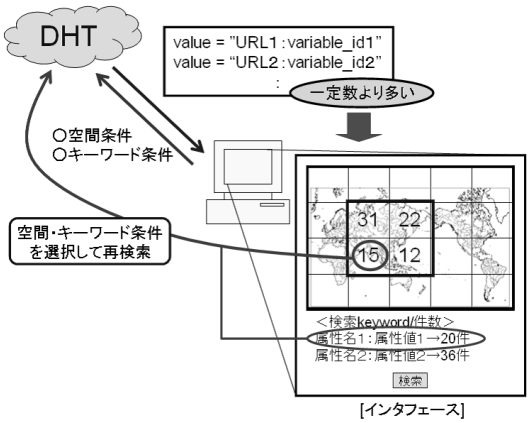

図 5　改良法 1 での処理例

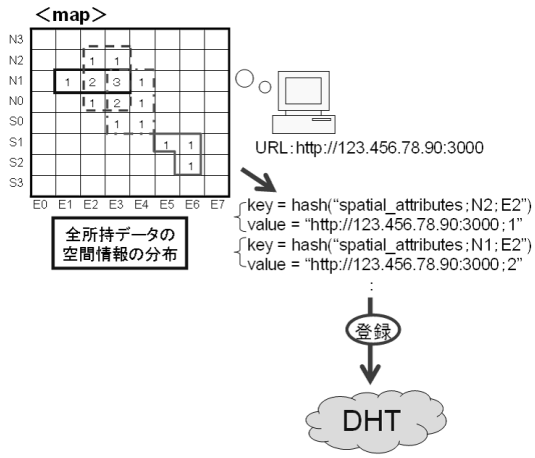

図 6　改良法 2 における空間情報の DHT への登録

検索条件として緯度経度を与えると value から該当データを所持しているノードの URL と該当データの最大数が分かる．この手法では 1 つのノードが DHT へ登録する空間情報のエントリ数が空間ブロックの切り分け方に依存しているのでエントリ数の上限が分かる．緯度経度を 10 度ごとのブロックに分けたとすると地図全体では 18 × 36=648 ブロックになるのでどんなにノードに所持データが増え *spatial_attributes* テーブルの空間属性に関するタップルが増えたとしても DHT に登録する空間情報のエントリ数は常に 648 個以下になる．予測の範囲内でエントリ数を調節出来るので対処法 1 と比較すると容易に細かいブロックの区分けが出来る．対処法 1 のような検索インタフェースを利用した表示や選択細分化表示には，より少ない DHT のエントリ数で細かいレベルまでの表示が可能となる対処法 2 は向いていると言える．

しかし素朴法の *keyword_attributes* との複合絞込みにおいて必要だった variable_id をこの対処法では value に載せていないので複合絞込みがノード URL の比較でしか出来ない．このため該当ノードを絞り込む精度は素朴法や改良法 1 より低くなる．

AND 条件で絞り込んだ時あるノードが持っているデータ A にのみキーワード条件が当てはまり，データ B にのみ空間条件が当てはまる場合，このノードは該当ノードとして数えられてしまう．

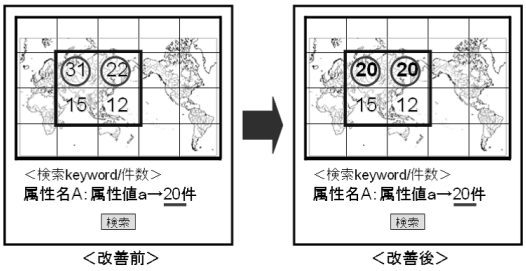

図 7　インタフェース表示時の工夫

また表示する件数はあくまでもデータの最大数である．ネットワーク上にあるノードがそれぞれ多くのデータを持ち，そのデータのほとんどがカバーしているブロックを主要な検索条件にすると，問合せを発行するノード数の絞込みという点での目安が薄れてしまう．

対処法 2 で空間条件の絞込みを行うと地図上に表示される数字が実際のノードやデータ数に対し大きくなり過ぎてしまうこともあるという懸念に関する改善案として，空間条件とキーワード条件の AND 検索であった場合ブロックごとの該当データ数を地図上に表示する際，キーワード条件に該当したデータ件数までしか表示しないようにするという方法を考えている．ブロック上に表示されている該当件数がキーワード条件の該当件数よりも多い場合は，両方の条件で絞込みをした場合に当然キーワード条件の該当件数以下になることを利用し，正確な数の表示ではないものの，どちらかの条件だけを用いた場合よりは該当件数を減らすことができる．

図 7 の例ではキーワード条件に該当したデータ数は 20 件である．空間条件とキーワード条件に該当した value の URL を比較して AND をとった結果，検索範囲にはそれぞれ 31，22，15，12 件該当するものがあると表示されているが，実際該当するデータの上限は 20 件なので 31，22 件該当するということは有り得ない．よってそれぞれを 20 に修正し表示する．

6.　まとめと今後の課題

Gfdnavi で扱おうとしているデータは膨大であり，従来のクライアント-サーバ方式ではサーバを設けるコストや帯域の圧迫の問題がある．それらの問題を解消するため Gfdnavi の横断検索機能を P2P で実現しようと考えた．その具体的な内容は Gfdnavi で定義されているリレーショナルスキーマを元に DHT を用いた絞り込み検索を行った後，改めて求めるデータを所持しているノードに問合せを発行するというものである．

今後は本システムの実用性と懸念事項への耐性を調べるために実験を行う予定である．実験では個々にデータを持ち Gfdnavi サーバを立ち上げているノードを数台用意し，5 節に示したそれぞれの手法を採用した時 DHT に登録したエントリ数やユーザが欲しいと思うデータに辿り着くまでに問合せを発行したノードの総数を求め比較する．この結果を元に 5.2 節で懸念していた素朴法の問題点を改良法がどれだけ改善出来ているかを検証する予定である．

実験結果をふまえどの機能を取り入れ強化するべきなのか模索し，将来的な Gfdnavi への組み込みを意識してバランスの取

れた動作をするシステムを構築したい.

謝辞

本研究は，文部科学省科研費特定領域「情報爆発時代に向けた新しい IT 基盤技術の研究」の課題 A01-14（課題番号 18049043）により行われた.

本研究遂行にあたって様々な協力やコメントを頂いた堀之内武，西澤誠也，森川晴大，林祥介，塩谷雅人氏ら地球流体電脳倶楽部の各氏に感謝する.

文　　献

[1] 金子勇: “Winny の技術”, 株式会社アスキー.

[2] 木村映善，村田健史：“RSS/RDF を利用した太陽地球系物理観測データのメタデータ配信の検討”，情報処理学会論文誌, Vol.47, No.4, pp.1051–1062, 2006.

[3] 首藤一幸, 田中良夫, 関口智嗣: “オーバレイ構築ツールキット Overlay Weaver”，情報処理学会論文誌:コンピューティングシステム, Vol.47, No.SIG12(ACS 15), pp.358–367, September 2006.

[4] 地球流体電脳倶楽部: http://www.gfd-dennou.org/

[5] 堀之内武，西澤誠也，渡辺知恵美，森川晴大，神代剛，林祥介，塩谷雅人：“地球流体データベース・解析・可視化のための新しいサーバー兼デスクトップツール Gfdnavi の開発”，データ工学ワークショップ (DEWS2007) (投稿中)

[6] 柳平有美：“Gfdnavi:地球流体物理科学者のためのデータアーカイブサーバ構築支援ツールの開発”，増永研・渡辺研中間発表会予稿.

[7] 渡辺知恵美：“地球惑星科学研究者のためのデスクトップサーチツールの開発に向けて”，情報処理学会研究報告 2006-DBS-140(2), Vol.2006, No.78, pp.429-436, 2006.

[8] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan.: “Chord: Scalable Peer-To-Peer lookup service for internet applications”, In *Proc. 2001 ACM SIGCOMM Conference*, pp 149–160, 2001.

[9] R. Huebsch, J. Hellerstein, N. Lanham, B. T. Loo, S. Shenker, and I. Stoica.: “Querying the Internet with PIER”, In *Proc. the 29th International Conference on Very Large Data Bases*, pp. 321–332, September 2003.

[10] S. Ratnasamy, P.Francis, M.Handley, R. Karp, and S. Shenker.: “A scalable content addressable network”, In *Proc. 2001 ACM SIGCOM Conference*, Berkeley, CA, August 2001.

[11] W. S. Ng, B. C. Ooi, Kian-Lee Tan, and A. Zhou.: “PeerDB: A P2P-based System for Distributed Data Sharing”, In *Proc. the 19th International Conference on Data Engineering*, pp. 633–644, 2003.