対話的な絞込み操作を考慮したP2PによるGfdnaviの横断検索

A cross-search mechanism for Gfdnavi on the P2P overlay network using Faceted Navigation

齋藤 真衣♥ 堀之内 武♦
渡辺 知恵美♠

Mai SAITO Takeshi HORINOUCHI
Chiemi WATANABE

近年増加している科学データに対し，個人レベルでのストレージの確保やデータの分類をすることなくデータ検索・公開・解析・可視化をサポートし，リレーショナルデータベースを用いたアーカイブサーバを構築できるGfdnaviを開発している.Gfdnaviが一般配布され多くのアーカイブサーバがWeb上に出現した時，個々のサーバでの検索だけでなく，P2Pを用いて他の公開サーバとの横断的検索をすることで研究者同士の知識共有を可能にする.そこで我々はGfdnaviの横断検索の開発を進めてきた.Gfdnaviで共通に定義されているリレーショナルスキーマをもとに，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.実用段階に向けGfdnaviの探索的検索インタフェースからシームレスに横断検索ができるようビットマップを用いた絞込みを可能にして拡張を行った.

In recent years, scientific data archives on geophysical and environmental fluids have increased in an explosive manner; therefore, it has become necessary to search for required data appropriately from a large amount of scientific datasets stored on PCs in order to share them among users. Hence, we have developed Gfdnavi, a data archiving server construction support tool for geophysical fluid databases.
Gfdnavi provides utility functions such as a metadata extraction tool, highly functional query interface, and data analysis/visualization libraries. Due to the widespread popularity of Gfdnavi among scientists, a large number of Gfdnavi servers have appeared on the Web. In this paper, we describe a cross-search mechanism for Gfdnavi servers on the Web. This mechanism aimed at establishing an autonomous, decentralized retrieval system by using a P2P network without setting up a central server.
Gfdnavi provides a faceted navigation interface to search scientific data, and we propose a P2P search mechanism based on the user-computer interaction flow on the faceted navigation interface.

♥ 学生会員 お茶の水女子大学大学院人間文化創成科学研究科博士前期課程 maisaito@db.is.ocha.ac.jp
♦ 京都大学生存圏研究所 horinout@rish.kyoto-u.ac.jp
♠ 正会員 お茶の水女子大学大学院人間文化創成科学研究科 chiemi@is.ocha.ac.jp

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．この現状を改善するべく，個人やグループ内におけるデータ解析・可視化作業からデータ検索・大規模データ公開までをカバーする Gfdnavi を共同で開発している．
Gfdnavi はリレーショナルデータベース(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できる．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，P2P を用いて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できる．そこで我々は Gfdnavi の横断検索の開発を進めてきた．先行研究[1]では空間条件とキーワードによる横断検索を試験的に実装したが，実用段階に向け Gfdnavi の探索的検索インタフェースからシームレスに横断検索ができるように拡張を行った．

2. 前提知識

本節では，Gfdnavi のピュア P2P 横断検索にて採用している分散ハッシュテーブルと Gfdnavi で想定される利用状況，さらに DHT を用いて Gfdnavi における横断検索を検討した先行研究について述べる．

2.1 DHT(Distributed Hash Table)

DHT はハッシュ関数を通して算出したキーとそれに対応する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，その分散された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，ノードが公開したデータの情報(メタデータ)がハッシュ関数を用いて生成される．メタデータは「データを特定するもの」をキーとし，「該当するデータもしくはそのデータの所在地(ノード id，オブジェクト id)」を値とする 2 項の組である．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うち探しているハッシュ値に最も近い値の ID を持つノードへ探索依頼を送る．ID やキー値の付与の仕方やメタデータの管理，ノードの辿り方(ルーティング)のアルゴリズムには Chord，Pastry などがある．DHT でユーザが

使う関数は put と get である．put は(キー，値)のペアを指定して DHT に登録し，get はキーを指定することでそれに対応する値を返す．

2.2 先行研究

先行研究[1]にて佐藤らは,2 段階の処理を経て問合せを実行した．まず想定する問合せを限定し，ユーザが指定する空間条件とキーワードをキー，その条件を満たすノードを値として DHT に登録して，それぞれの条件に該当するデータを所持するノードを DHT に問い合わせて特定する．また各 Gfdnavi サーバには SQL を受け付けその結果を返す Web サービスを起動しておき，DHT 検索で該当データを所持しているとされた全ノードに対し，Web サービスを用いて SQL を発行する．DHT で行われるのはデータを所持するノードを特定するだけの検索であり，ほとんどノードが絞られない場合は多くのサーバに SQL を発行しなければならない．さらに Web サービスでは任意の SQL を実行できてしまい，データの削除も可能となるためセキュリティ上に問題がある．そして対象ノードがネットワークに参加していない場合には問合せに対処できない．また先行研究では試験的に横断検索を導入したため Java によるコマンドプログラムを実行しており，Gfdnavi から動作させることまで具体的に想定していなかった．

3. Gfdnavi における横断検索

Gfdnavi では図 1 のようなメタデータに対する属性が定義されており，ユーザは各属性リストから検索条件を指定することで検索を進める．各属性の詳細や属性間の関連については[2]を参照されたい．

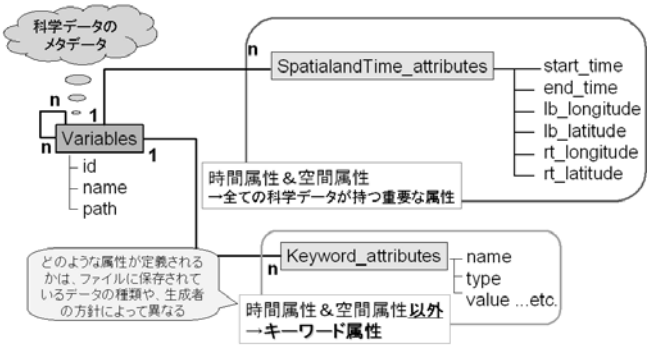

図 1　Gfdnavi における属性
Fig.1　Attributes of Gfdnavi

前節で述べた先行研究の問題点を踏まえ，Gfdnaviの横断検索機能の実用化に向け本研究ではGfdnaviに新しく加わる機能として重川[2]により実装されている「探索的検索インタフェース(以下Gfdexplorerと呼ぶ)」を，そのまま横断検索でも利用できることを目的とする．図2 はGfdexplorer のスクリーンショットである．①はキーワード属性リスト，②は結果データセット情報，③は該当オブジェクトの位置情報，④は時間属性リストを示している．ここで探索的検索とはあるデータ全体の中から必要としているデータを求める際に1度検索をかけるだけでなく，条件によってデータを絞込んでいく検索を繰り返したり，検索結果を比較したり，また検索結果同士を組み合わせたりしながら徐々にユーザの必要とするデータに近づいていく検索方法を指す．探索的検索の代表的な手法にはFaceted Navigationが挙げられる．既存のGfdnaviでは空間属性に対してのみGoogleMapを用いた探索的検索ができるが，時間属性やキーワード属性に対しても同様に探索的検索が行えるよう現在新しいインタラクションモデルが開発されている．新しい検索インタフェースではデータの属性値をもとにグルーピングをし，その結果をリスト表示する．ユーザがリスト中のグループの一つをクリックするとそれを絞込み条件として問合せをし，データを絞り込んで結果を出力する．このインタフェースで横断検索をする際，ユーザは図2 の右上部分のenable cross searchにチェックを入れるだけで良いものとする．

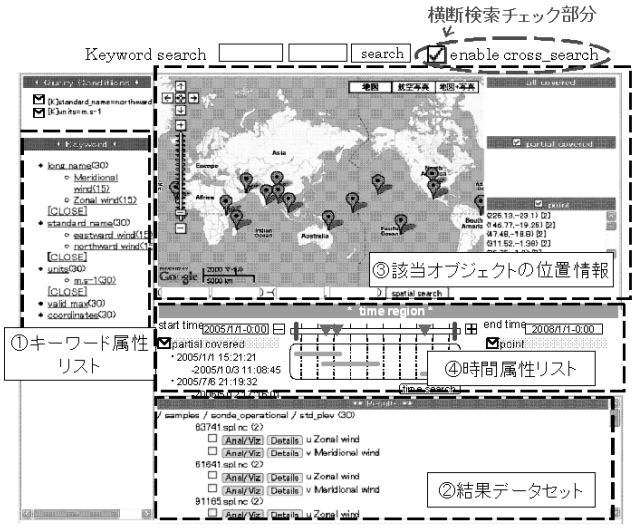

図2　Gfdexplorerのスクリーンショット
Fig.2　Screenshot of Gfdexplorer

4. 探索的検索を実現するための DHT のデータ構造と処理

本節では探索的検索を横断検索において実現するため，各属性におけるDHTのデータ構造と処理について述べる．Faceted Navigationでの問合せでは，単一条件による絞込みと複数条件による絞込みの方法について考える必要がある．

4.1　単一条件による絞込み

本項ではGfdexplorer で横断検索をする際の単一条件による絞込みについて基本アイデアを述べる．2.1 節で述べたように DHT ではキーで検索し，対応する値を結果として返す．そこで DHT での検索のために考えるべきことは何をキーにし，何を値とするかに帰結する． Gfdexplorer ではある条件 C をユーザが指定したときにその結果として表示される検索結果集約情報(主要なキーワード属性名と各々のキーワード属性のとり得る主要な値，ファイルパス，空間属性，時間属性)がリストアップされる．その検索結果集約情報をAggr(C)とすると，Aggr(C)からユーザが指定する次の条件C’で絞込みを行う．DHT を用いた横断検索でもユーザが指定する条件 C をキー,条件 C について検索した結果の Agr(C)を値とし，キーと値の組合せを登録する．Aggr(C)は以下の情報を含む．

・キーワード属性リスト（図 2①）
・結果データセット情報（図 2②）

このキーと値の組合せによりリンク先をクリックして DHT から取得したデータをもとに HTML のページを作成し，Faceted Navigation による検索が可能となる．1 つの条件による問合せは 1 回の get で完了し，複数ノードが同じ条件で検索結果を持っていたならば，検索条件すなわちキーが同じであるため同じノードに値が置かれる．例えばリストから属

性名 standard_name を選択すると，その属性値 Dewpoint temperature や Elevation の他にそれぞれの該当件数が表示される．さらに属性値 Elevation を選択すると属性名 standard_name の属性値 Elevation で検索され，次の絞込み条件の候補となる属性名リストが返される．基本アイデアに基づき，検索結果集約情報を以下のように定義する．

・属性名もしくは属性名と属性値のペアによる検索条件をキーとする．
・キーの条件によって検索された variable が持つ属性の属性名をその属性名を持つ variable の件数上位 p1 件分，およびさらに上位 p2 (p2<p1)件の属性名に対しては該当する属性名のリスト q 件をキーに対する値として持つ．

つまり検索条件 C に対して，該当する variable が持つ属性の属性名を ai (0≦i≦p1-1), 属性名 ai に対する属性値を vij(0≦j≦ai の取りうる属性数)とすると検索集約情報 Aggr(C)は以下の通りとなる．

$Aggr(C)=((a_0,(v_{00},\ldots,v_{0q})),\ldots,(a_{p2},(v_{p20},\ldots,v_{p2q})),a_{p2+1},\ldots,a_{p1})$

4.2 複数条件による絞込み

4.1 節で述べた手法では単一の検索条件には対応できるが，複数条件による絞込みを行う場合に対処できない．複数条件による絞込みでは以下の 2 つの方法が挙げられる．

① 2 つの単一条件による絞込み結果から件数を見積もる．
② ①の結果から再び DHT に問合せ，正しい件数を求める．

①の方法では正しい件数は得られないが，最大で何件かという近似値を見積もることができる．例えば条件 1 による検索結果が 10 件，条件 2 による検索結果が 5 件であった場合，まず①の方法で件数の少ない条件 2 による検索結果をとって見積もることができる．さらにユーザが正しい件数を求めたい場合は②の方法で件数を取得する．このように大まかな件数を求めたい場合は見積もり値で対処し，ユーザの要求があった場合は正しく再計算を行う．正しい件数を取得する手法として，本研究ではある検索条件に対する検索結果集約情報に加えて各 variable が検索条件を満たすか満たさないかを 1 と 0 だけのビット値で表したビットマップを DHT にて共有することとする．例として図 3 に示されるキーワード属性における絞込みでは，属性名 A の属性値 a3 を満たす variable に対応するビットは 1 で表されている．

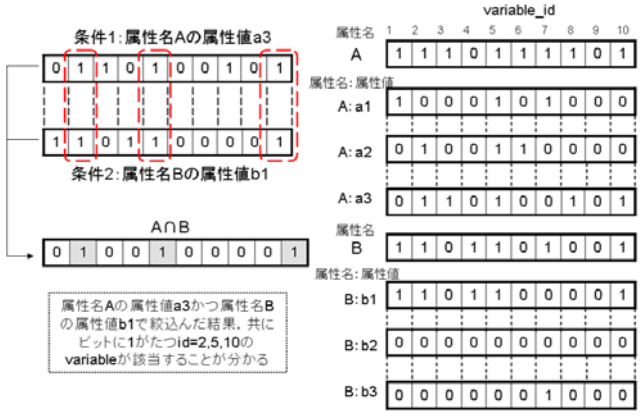

図3 キーワード属性における絞込み
Fig.3 Example of narrowing down variables by keyword_attributes

データの有無を 1 ビットで表現でき，ビット間の論理積や論理和の計算は高速であることから，ビットマップを用いて絞込み検索を実現する．まずキーワード属性における検索ではユーザが検索条件として指定し得る属性名，さらにその属性値それぞれに対して variable 数の分だけビットマップを用意し，各ビットは各 variable_id に順に対応させる．例えば図 3 では属性名 A，属性名 A の属性値 a1,a2,a3，同様に属性名 B，属性名 B の属性値 b1,b2,b3 に対してビットマップを用意し，属性名 A の属性値 a3 かつ属性名 B の属性値 b1 で絞り込まれた場合，共にビットに 1 がたつ id=5 の variable が該当する．その結果をもとにキーワード属性のリストとその件数を求める必要がある．絞込みの際の該当件数取得の流れを図 4 に示す．クライアント側で属性名 A と B のビットマップ，キーワードリストを所持し，そのリストから共通する属性 C,D,E について絞込み可能であることから，属性 C で絞り込むこととする．キーワードリストとその件数から A かつ C は 3 件，B かつ C は 4 件であり，A かつ B かつ C は 3 件と見積もることはできるが，実件数は求めることができない．この場合 A かつ B のビットマップをキーとして DHT に問い合わせ，A かつ B かつ C のビットマップをもとに該当件数を計算する必要があるが，クライアント側でビットマップをすべて取得してくるのはコストが高い．そこで我々は，ビットマップを置いてあるノードで該当件数処理をする getcount 関数を用意する．件数取得のチェック項目を設け必要に応じて件数取得処理をすることにより，DHT で取得するデータ量を減らす．

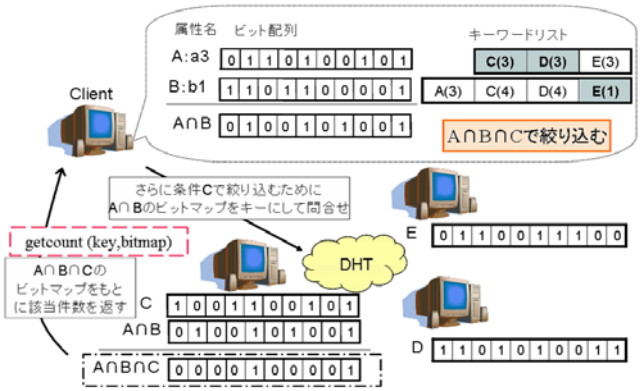

図4 該当件数取得処理の流れ
Fig.4 Example of getcount

getcount は以下のように定義する．

int getcount(key, bitmap)

検索条件とクライアントが所持するビットマップをキーとし，次の絞込み条件を満たすビットマップをもとに該当件数を値として返すものとする．ZU4 の例では最初の見積もりとは異なり，該当件数 2 件と分かる．このように各キーに対して検索結果リスト，ビットマップを DHT の値とすることで，キーワード属性における絞込み検索ができる．ここでは説明のために単一ノードから結果を得る場合を例としたが，横断検索においては複数ノードから検索結果が返ってくることが考えられる．その場合，複数ノード分のビットマップを取得してクライアント側で所持しておき，ノードごとに getcount で件数を計算してその合計を該当件数として返す．Gfdnavi 全体では大きなノード数を扱うが，すべてのノードに対して横断検索するのではなく実際は共有ノード間と一般公開ノードによるものと想定されるため，そのビットマップを取得するコストはそれほど高くないと考えられる．また

該当ノードをビットマップで表すときはデータの挿入，削除，更新に対してコストがかかるという問題がある．ただしGfdnaviの横断検索では現段階ではそれらの更新は頻繁に行われないものと仮定する．

4.3 オブジェクト

本項では結果データセット情報を表示させるためにDHTに登録するデータ構造について述べる．まずvariable_idをキー，オブジェクトのidを値としてDHTに登録し，オブジェクトの基本情報を取得できるようにする．各属性について検索を進めるとビットマップから該当variable_idが割り出されるので，ビットの位置情報をキーとし，基本情報，ノードid，名前，パス，descriptionを値として登録する．ブラウザの表示範囲の大きさから最大5件ずつ表示するものとし，1で表されるビットのうち5つを選んでオブジェクト情報を取得する．

4.4 空間属性・時間属性

空間属性においては緯度経度をそれぞれ10度ずつに切り分け，各々ビットマップを用意する．ユーザが図5のように検索したい空間を指定した場合，その空間の緯度経度の範囲を一部でも含めば条件を満たすものとし，経度は10～30度，緯度は20～50度の範囲のビットマップを参照する．緯度経度ごとにビットの論理和をとると，経度はid=3,8,10のvariable，緯度はid=1,5,8のvariableが該当する．最後に緯度経度共に条件を満たすvariableを論理積で求めるとid=8のvariableが該当し，これにより空間絞込みができる．また検索の初期段階では該当データ数が多く，ビットマップによる絞込みのコストが高くなってしまうため，まず大まかに60度ずつに切り分けて絞込みが進んでから細かく切り分けることとする．

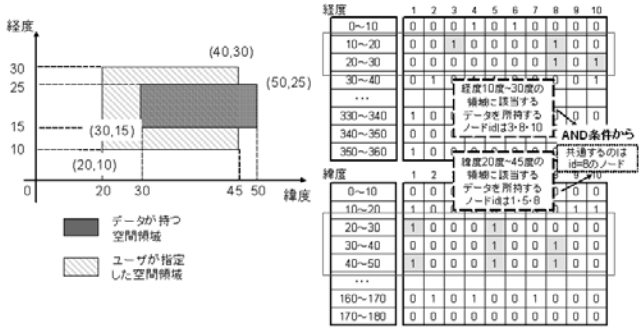

図5 空間属性における絞込み
Fig.5 Example of narrowing down variables by spatial_attributes

時間属性においても年，月，日毎にビットマップを用い，年月日毎に対応ビットマップを参照すると該当ビットが分かる．

5．実装

Gfdnaviは Ruby 言語による Web アプリケーション開発フレームワークである Ruby on Rails を拡張し，地球流体科学者を対象とした高機能なデータカイブサーバ構築を支援するパッケージである．横断検索部では P2P ネットワークルーティング及び分散ハッシュテーブルの管理・制御に Java 言語によって実装された OverlayWeaver[3]を利用するため，Ruby から Java のクラスを呼び出すライブラリである RJB(Ruby Java Bridge)[4]を用いてデータ検索部との連携を図る．我々は RDB に格納されているテーブルの索引情報を OverlayWeaver の分散ハッシュテーブル管理モジュールに登録するハッシュ検索登録モジュールと，検索インタフェースにより実行された問合せをもとに横断的検索を行う問合せ処理モジュールを実装する．

5.1 DHT への登録

まず 4.1 節で述べた基本アイデアに基づき，RDB に格納されている検索対象データセットからユーザが検索条件に指定する DHT のキーと，それに対応する値を生成する．ここで DHT に登録する値のうちブラウザ上で表示される検索結果リストはこの段階では複数の情報をまとめて格納している型であるため，これを DHT の値として put するためには文字列に変換する必要がある．結果生成モジュールで文字列化した値を OverlayWeaver で put することで，キーと値の組が DHT へ登録される．

5.2 DHT を用いた検索

図 6 に DHT を用いた検索の流れを示す．横断検索時，まず step1 としてユーザが一覧から指定した属性名がインタフェースからパラメタとして取得され，横断検索モジュールに渡されて DHT のキーが生成される．そこで Ruby で実装されている横断検索モジュールから RJB を用いて OverlayWeaver を呼び出し，get 関数を実行してキーに対応する DHT の値を取得する．取得された値は結果解凍モジュールで検索結果として表示できるように変換される．次に step2 では step1 によって表示された結果に対してユーザが該当件数を取得したい場合，チェックの有無によって該当件数を求める処理が行われる．このような 2 段階の問合せを繰り返すことにより，インタフェースにはユーザが指定した条件で絞り込んだ検索結果が表示される．

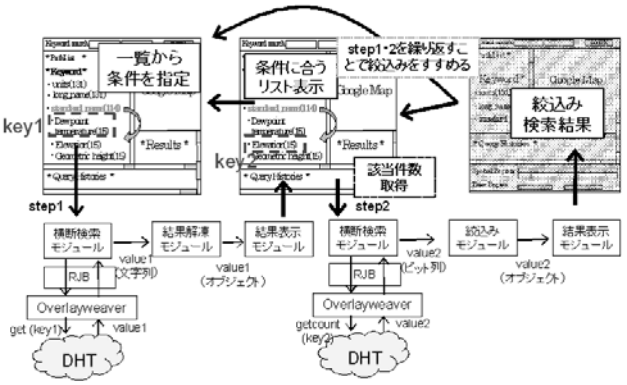

図6 DHT への問合せ処理モジュール
Fig.6 query process module

6.評価

本節では提案手法を用いた評価実験とその考察について述べる．実験は OverlayWeaver を用いて 1 台のマシンで複数ノードをエミュレートし，DHT ネットワークを構築した．ルーティングアルゴリズムは Chord を用いた．本節では以下の項目に関して評価を行う．

・検索（get）にかかる時間
・該当件数取得（getcount）にかかる時間
・該当件数の見積もり値と実際の値の差

本論文にて提案した手法では，各検索条件における集約結果を予め求め DHT で公開するため，各ノードが所有するデータ量が大きくなることが予想される．その反面 Faceted

Navigation による問合せのために最適化されているため，検索にかかる時間およびオーバーレイネットワーク上の通信量は少なくなる．

6.1 共有データ量及び問合せコストの見積もり

まずは各ノードが所有するデータ量及び検索時間を見積もり，文献[7]による手法（あるタプルについて属性名と属性値のペアをキーとし，該当するタプルを値とする）と比較する．なおこの手法を以下従来手法と呼ぶこととする．

本提案手法にて一つのノードが DHT にて公開するデータは以下の 4 点である．

・検索結果集約情報
・空間・時間集約情報
・該当オブジェクト(一部)
・ビットマップ

本節では特に空間・時間属性以外の属性（キーワード属性）による横断検索を想定し見積もりを行う．まずノード Ni における公開データ量は

（検索キー数）×（検索結果集約情報＋ビットマップ＋該当オブジェクト）

となる．検索キー数は属性名と属性値のペアの数である．あるノード Ni 上の variables が持つ属性の属性名の集合を Ai, n(Ai)をノード Ni における属性名の数とし, 属性名 aij∈Ai (0≦j<n(Ai))における属性値の集合を Vij としその属性値数を n(Vij)とする．その時検索キー数は

$$n(Ai)+\sum_{j=0}^{n(Ai)-1} n(Vij)$$

となる．また検索結果集約情報は 4.1 節をもとに属性名および属性値のデータ量をそれぞれ 64bytes, 4bytes とすると 64p1+4qp2, ノード Ni における variables を Di としその数を n(Di)とすると，ビットマップの大きさは n(Di)/8 となる．以上のことから，ノード Ni における公開データ量は

$$P(Ni)=(n(Ai)+\sum_{j=0}^{n(Ai)-1} n(Vij))(64p_1+4qp_2+\frac{n(Di)}{8})$$

となる. n 個のノードで同程度のデータ量を公開し m 個のノード(m≧n)で共有した場合には, 一つのノードで所有するハッシュテーブルの大きさは平均

$$\sum_{i=0}^{n-1} P(Ni)\Big/m$$

と見積もることができる．

たとえば公開ノード数=参加ノード数とし，各ノードが持つ属性名数の平均を 30, 各属性の属性値数の平均を 10, p1=p2=5, q=3, variables 数の平均を 40 としたとき 1 つのノードで所有するデータの大きさは平均 127KB となる．

一方，従来手法では属性テーブルをすべて DHT にて共有するとして見積もる. ノード Ni において各属性名 aij および属性値 vij のペアを持つ variable 数を n(Var(aij,vij))とすると全タプル数は Σn(Var(aij,vij))となる．1 タプルの大きさを属性名の大きさ+属性値の大きさ=68 バイトとすると P(Ni)=68Σn(Var(aij,vij))となる．上記の計算例をこちらにも当てはめてみると各ノードで所有するデータの大きさは 408KB となり従来手法よりもコンパクトにまとまった．

次に本研究における検索時のコストを見積もる．4 節で述べたように問合せは以下の 3 つの場合に分けることができる．

(1) 単一の条件で検索をした場合
(2) 複数の条件で絞り込みをした場合（該当件数は見積もり値を取る）
(3) 複数の条件で絞り込みをした場合(該当件数は bitmap を用いて再計算する）

ここでの検索コストは 1 回の検索に対し DHT への get 関数を発行した回数を見積もる．まず単一の条件で検索した場合はその条件をキーとして検索をするためコストは 1 である．

複数の条件で絞り込みをし該当件数は見積もり値を取る場合も絞り込みのための検索を 1 回行い，絞り込み前のデータと合わせて該当件数の見積もりをするため問合せ自体のコストは 1 である．該当件数を再計算する場合は，ある 1 つの該当件数表示を再計算するのであればその条件をキーにして getcount 関数を実行すれば良いのでコストは 1 である．

ただし絞り込みを行いその結果の属性名および該当件数一覧を表示したとき表示する該当件数をすべて再計算するのであれば属性名の数 n(Ni)だけの検索コストが，またある属性名（aij）を指定したときにその属性値一覧と該当件数を表示するためには n(Vij)の検索コストがかかることになる．

従来手法では一度条件を満たす variable の属性名および属性値を調べる必要があるため，属性名・属性値リストを求めるには(1+該当する variable 数)回もの get 関数発行を行わなくてはならない．

以上のことから従来手法に比べて比較的コンパクトな大きさのメタデータを DHT にて共有することによって Faceted Navigation による横断検索を効率よく行うことができると考えられる．

6.2 実験結果と考察

前節の見積もりで使用したパラメタ値を用いて実験を行った．参加ノード数=データ公開ノード数=10，各ノードでの variable 数，属性名数，各属性の属性値数はそれぞれ[0, 平均の 2 倍の値]の範囲からランダムに設定した．つまり variables 数は 0~80，属性名数は 0~60，属性値数は 0~20 の範囲からランダムに設定されることとなる．表 1 に検索に用いた Query の詳細，表 2 に表 1 の Query による get 及び getcount の実行時間を示す．

	1 回目の問合せ（属性名，属性値）	2 回目の絞込み（属性名，属性値）
Query1	(name7, value1)	(name10, value8)
Query2	(name10, value5)	(name4, value0)
Query3	(name5, value10)	(name21, value1)
Query4	(name11, value2)	(name5, value1)
Query5	(name2, value7)	(name21, value2)

表 1 クエリ一覧
Table.1 List of query

	1 回目の get	2 回目の get	getcount
Query1	0.019868	0.009790	0.129537
Query2	0.017437	0.014225	0.115597
Query3	0.044467	0.006752	0.121185
Query4	0.018673	0.014176	0.125047
Query5	0.018184	0.011294	0.110514
平均	0.023726	0.011247	0.120376

単位：秒

表 2 get/getcount の実行時間の比較
Table.2 Comparison of get/getcount execution times

表 2 より，DHT の get で複数回の絞込みを行っても実行時間が大幅に増加することはない．また getcount では絞込みで該当するすべての属性に対して実際の件数を取得してい

るため通常の get より実行時間は増加しているが，それほどのコストではないと考えられる．次に，表 1 の Query5 における該当件数の見積もり値と実際の値との比較結果を図 7 に示す.

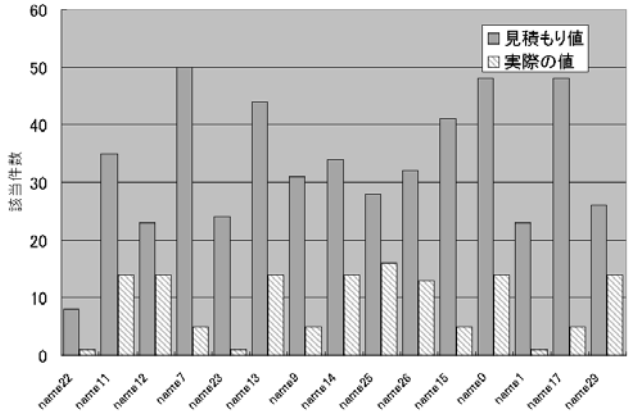

図 7　見積もり値と実際の値との比較
Fig.7 Comparison of estimation with actual number

図 7 より，どの属性においても見積もり値と実際の値とでは該当件数の差が大きいことが分かる．この差を小さくすべく見積もり値をより正確にとるため，variable を分割してグループ化しグループごとの件数を記録して，そのグループ内での絞込みを検討したい.
以上の考察より，Gfdnavi の横断検索において DHT を用いた絞込み操作が可能であり，また get と getcount による該当件数取得処理によって横断検索していないときと同様の検索方法及び結果をユーザに提供できる.

7. 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．DHT のキーと値のペアを問合せの全てのパターンで登録した場合，ネットワーク上に分散するデータ量の増加が考えられる．その解決策として，まず登録した値を文字列圧縮することでデータ量を減らす方法が挙げられる．さらに最初は上位数件のみを DHT に登録し，ノードに詳細な問合せが生じた場合のみユーザの検索頻度が高いものとして新たに登録することで，データ量を抑えた上で DHT での大まかな横断検索を行うことが可能になると考えられる．また絞込み操作のためのビットマップはデータ量が多くなり圧縮する必要があるため，variable 同士のパス関係を考慮した上でのデータ圧縮を検討している．Gfdnavi ではデータの更新はほとんどないものと考えられるが，実際に更新がある際には絞込みに用いるビットマップを該当部分だけ書き換えたり，put の TTL を短くしたりすることで対応できると考えられる．今後は Gfdnavi への組込みに向け更なる実験，考察を進めていきたい.

[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(課題番号 19024039)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] 佐藤麻美, 渡辺知恵美：“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,” 第18 回データ工学ワークショップ,D1-9 2007.
[2] 重川美咲子, 西澤誠也, 堀之内武, 渡辺知恵美：“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 データ属性の探索的検索を利用する検索,” 第19 回データ工学ワークショップ,C9-4 2008.
[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] Ruby Java Bridge: http://rjb.rubyforge.org/
[5] 地球流体電脳倶楽部: http://www.gfd-dennou.org/
[6] 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兼デスクトップツールGfdnavi の開発, ” 第18 回データ工学ワークショップ,D2-8 2007.
[7] 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.

齋藤 真衣 Mai SAITO
お茶の水女子大学大学院人間文化創成科学研究科博士前期課程在学中．2008 年お茶の水女子大学理学部情報科学科卒業．データベースシステムの研究・開発に従事．日本データベース学会学生会員.
堀之内 武 Takeshi HORINOUCHI
京都大学生存圏研究所助教．博士(理学)．1997 年京都大学大学院理学研究科博士後期課程修了．気象学の研究および気象学のためのシステム開発に従事．情報処理学会・日本気象学会・米国地球物理学連合・地球電磁気地球惑星圏学会会員.
渡辺 知恵美 Chiemi WATANABE
お茶の水女子大学大学院人間文化創成科学研究科講師. 2003 年お茶の水女子大学大学院人間文化研究科修了. 博士(理学). e-サイエンスおよびデータベースセキュリティの研究・開発に従事．日本データベース学会正会員.