DHT における XPath に基づく XML キーワード検索の性能評価

李　暁晨† 天笠　俊之†,†† 北川　博之†,††

† 筑波大学システム情報工学研究科　〒 305–8573 茨城県つくば市天王台 1-1-1
†† 筑波大学計算科学研究センター
E-mail: †babala0851@kde.cs.tsukuba.ac.jp, ††{amagasa,kitagawa}@cs.tsukuba.ac.jp

あらまし　近年，P2P 環境における XML 文書の分散管理が注目され，様々な研究や応用が進められている．我々の研究グループはこれまで分散ハッシュ表を利用した XML 文書の検索手法として，XPath を利用する手法と，転置リストを利用したキーワード検索手法を提案した．また，これらの手法を結合し，XML 文書の構造情報とテキスト情報の双方を考慮し，XPath に基づく XML キーワード検索手法を提案してきた．本論文では，この結合手法の性能評価を報告し，手法の有効性を実験により検証する．

キーワード　XML 文書，P2P 環境，全文検索，XPath，DHT，転置リスト

An Performance Evaluation of Searching XML Documents using XPath Full-Text in Structured P2P Networks

Xiaochen LI†, Toshiyuki AMAGASA†,††, and Hiroyuki KITAGAWA†,††

† Graduate School of Systems and Information Engineering, University of Tsukuba
1-1-1 Tennoudai, Tsukuba-shi, 305-8573, Japan
†† Center for Computational Sciences, University of Tsukuba
E-mail: †babala0851@kde.cs.tsukuba.ac.jp, ††{amagasa,kitagawa}@cs.tsukuba.ac.jp

Abstract　Recently, Peer-to-Peer systems and XML are attracting increasing attention. There have been several attempts to deal with XML documents in P2P networks. Our research group has been proposed two methods for XML document processing in DHT-based P2P networks. One uses XPath and the other uses inverted list to realize keyword search. Then, to realize XPath full-text search on XML document, we combine the two proposed methods. In this paper, we evaluate the effectiveness of our proposed method by an experimentation

Key words　XML, P2P network, full-text search, XPath, DHT, inverted list

1. はじめに

Peer-to-Peer (P2P) 技術は，オーバレイネットワークの基盤技術として，近年活発に研究されている．P2P システムとは，システム内のピア同士が対等な関係を持ちながら処理を行う分散システムのことである．P2P システムの一手法として，分散ハッシュテーブル（DHT）がある．DHT を用いることにより，検索対象のデータを効率的に検索することが可能となる．

一方，XML はタグによって文書やデータにおける意味や構造を記述するためのメタ言語である．1998 年に W3C 勧告となってから現在までの間に急速に普及し，標準的なフォーマットとして，様々な分野に広く応用されている．こういった XML データを P2P 環境で利用することも一般的に行われるようになっているので，P2P 環境で XML データの格納と検索を効率的に行うための技術基盤がますます重要になっている．

一般に，XML 文書の問合せには，構造型問合せ，非構造型問合せ，半構造型問合せの三つのタイプがあると考えられる．構造型問合せでは，利用者が XML 問合せ言語（XQery [7], XPath [10]）を用い検索要求を問合せ式に記述する．この方式は，利用者が XML 問合せ言語の専門知識を有し，文書構造を事前に把握していることが前提になる．我々の研究グループは分散ハッシュ表に基づき XPath による XML 文書の効率的な格納・検索を可能にする手法 [4] を提案している．

しかし，実際の P2P 環境では異なる利用者がさまざまな XML スキーマを用いる可能性があり，全ての XML 文書構造を把握することが現実的ではないため，非構造型問合せのニーズが高まっている．そこで，我々は転置リストを利用した XML 文書のキーワード格納・検索手法を提案してきた [5].

一方，両者の中間的な問合せ方法として，半構造型問合せも考えることができる．この場合，利用者は検索対象の XML 文

書の部分的な構造とキーワードの双方を問合せ式に記述する．このような問合せは，利用者が部分的な構造しか知らない時に使用される．最近新たな XML 問合せ言語として公表された XPath Full-Text [9] では，ftcontains なる演算子を用いて XPath による XML 文書のキーワード検索を表現することができる．

そこで我々は，これまでに提案してきた二つの提案した手法を統合し，キーワードを含む XPath による XML 文書検索手法を提案した [17]．具体的には，XML の構造情報とテキスト値の双方を検索するため，問合せ式を二つの部分（パス部分とキーワード部分）に分けて，拡張した KW-DHT によって処理を行う．本論文では，評価実験により，提案手法の有効性を検証する．

本論文の構成は以下の通りである．まず第 2 章で関連研究について述べ，第 3 章で本研究に関する基本的事項を説明する．次に第 4 章で提案手法の概要を述べ，第 5 章で提案手法による評価実験について説明する．最後に，第 6 章でまとめと今後の課題について述べる．

2. 関 連 研 究

P2P 環境における XML 文書の格納・検索には，いくつかの手法が提案されている．Reynolds 等 [2] は DHT を用いて転置リストを実現し，(XML でない）文書の全文検索を行う手法を提案している．そこでは，AND 検索におけるコンテンツ ID の転送時にブルームフィルタを用いてトラフィックを低減する手法を用いている．本研究は，この手法を XML に拡張したものとして位置付けられる．

Abiteboul 等は，DHT に基く XML 検索手法として，KadoP [13] を提案している．KadoP において，XML は要素あるいはテキスト中のキーワードに基いて分解され，DHT 上に格納される．検索処理としては XPath がサポートされ，XPath 問合せを構成する各要素名あるいはキーワードのリストが DHT によって検索され，それらは Holistic Twig Join によって問合せ結果へと変換される．さらに，[14] では，DHT を構成するピア間のストレージにおける負荷分散のために，DHT 上に B-tree に類似した構造を構築する手法を提案している．また，問合せ結果の候補の効率的な絞り込みのために，ブルームフィルタに基づくフィルタリング手法も提案している．

しかし，これらの手法は，キーワードを用いた XML 文書の半構造問合せを実現していない．また，転置リストにの実現にピア ID を使っているため，万が一ピアがオフラインの場合は，検索処理を行うことができないといった点で本研究と異なる．

3. 基本的事項

本論に入る前に，提案手法の前提となる基本的事項について伸べる．

3.1 分散ハッシュ表 (DHT)

分散ハッシュ表 (DHT; Distributed Hash Table) は，構造型 P2P ネットワークの一種である．DHT では，システムで共通の一つのハッシュ関数を用意し，このハッシュ関数から得られる値を元に，オブジェクト（ピア，データ）をオーバレイネットワーク上に配置する．ピアが DHT システムに参加する際，担当すべきハッシュ値の範囲が与えられ，その範囲に当たるデータをピア間で分割して担当することになる．ある検索キーで登録されたオブジェクトを検索する場合は，その検索キーのハッシュ値を計算し，このハッシュ値を担当するピアにオブジェクトの所在を尋ねる．DHT には，代表的な方式として，Chord [3]，Pastry [6] と CAN [8] などがある．以下では本研究で用いる Chord について紹介する．

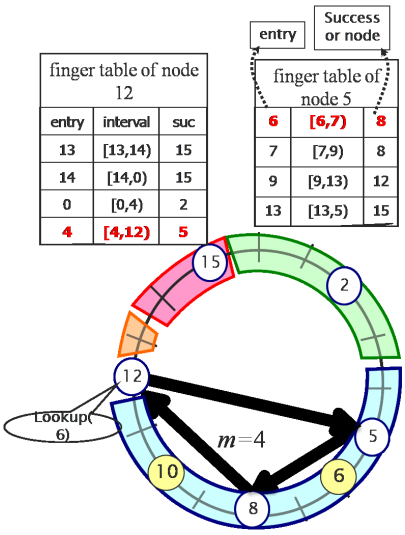


図 1　Chord の ID サークル
Fig. 1　An example of Chord.

Chord では，2^N 個のオブジェクトからなる円構造のハッシュ空間（ID サークル）を用いる．N を ID のビット数とする．オブジェクトはハッシュ関数で ID サークル上にマップされる．ID が i のピアは，担当するデータの ID，フィンガーテーブル，前後に位置するピアの ID 情報 ($predecessor$, $successor$) を保持している．またピア i が担当するデータは，前ピア ($predecessor$) から i までのハッシュ値を持つオブジェクトである．フィンガーテーブルには，N 個の他ピアへのリンクがあり，リンク先は $i + 2^k$ $(k = 0, 1, 2, ..., N - 1)$ 位置を担当するピアである．このため，オブジェクトの検索に要するホップ数は $O(\log_2 N)$ である．

図 1 に Chord の構造とオブジェクト検索の例を示す．ピア 12 がキー 6 のデータを検索したいとする．ピア 12 のフィンガーテーブル内で最もキー 6 に近いのはピア 5 であるため，まずピア 5 にアクセスする．これを繰り返し行うことで，キー 6 の情報を保持するピアへアクセスする．この例では，キー 6 を保持しているのがピア 8 であるため，最終的にピア 12 はピア 8 にアクセスし，キー 6 に対応するデータを取得する．

3.2 XPath Full-Text

XPath Full-Text は，XML 問合せ言語 XPath [10](XML Path Language) に全文検索機能を追加した問合せ言語である．XPath Full-Text は 2008 年の 5 月に W3C 勧告として公表された，XML 文書の構造を考慮した全文検索をするための

標準的な方法を提供している．例えば，従来の XPath で提供されていた述語である contains に対応する述語として，全文検索のマッチングを示す ftcontains を利用することができる．

`//book[@no = "1"]`/title ftcontains (”usability” ftand ”testing”)

は，号数が 1 の書籍において，タイトルが “usability” と “testing” に関連するかどうかを真偽値によって返す．本研究では，この ftcontains に着目し，DHT 上での実現方法を検討する．

4. 提案手法

ここで我々が文献 [17] で提案した，DHT における XPath に基づく XML キーワード検索手法を説明する．

具体的には，DHT を利用した XML 問合せを実現するために，まず XML データを分解し DHT に格納する．XML データの構造に関する部分は [4] に従い，キーワード検索に関わる XML テキストの部分の格納方法は [5] に従う．

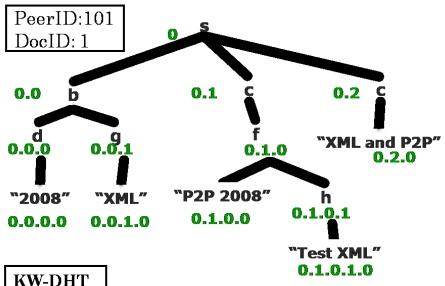
問合せ処理は以下のように行う: 1) 問合せのパス部分が述語を含まない場合，例えば，`s/c ftcontains ("2008" ftand "XML" ftand "P2P")` のような問合せの場合は，従来手法 [5] を若干拡張することによって処理可能である．2) 問合せのパス部分が述語を含む場合，例えば，`s/c[f] ftcontains ("2008" ftand "XML" ftand "P2P")` のような問合せの場合，問合せをパス部分 `s/c[f]` とキーワード部分 `s/c ftcontains ("2008" ftand "XML" ftand "P2P")` に分けて処理を行う．基本的な戦略としては，キーワード部分は 1) と同様の処理を行う．パス部分については，[4] によって該当するノードの候補を取得し，最後に両者をマッチングすることによって最終的な解を得る．

以下，まずは DHT を利用した転置インデックスの実現方法から説明する．

4.1 パス式を導入した KW-DHT

KW-DHT とは，[5] で我々が提案した，DHT を利用した XML のための転置インデックスの構築方法である．本研究では，パス式を含むキーワード検索を扱うため，KW-DHT を拡張する必要がある．具体的には，従来はある単語に対応するエントリの値が，その単語を含む要素名とそのノードの LocationID（ピア ID + 文書 ID + XML ノード ID）だったのに対し，パス式によるフィルタリングを可能にするために，単語の所在パスを KW-DHT の転置リストに格納するようにした点である．XML データの各葉ノード（テキストノード）と属性ノードについて，テキスト値または属性値の中の各単語をハッシュキー，（単語所在パス, LocationID）の転置リストを値とする．単語所在パスは XPath Full-Text クエリのパス部分を処理するために利用する．LocationID は単語が出現する場所の位置情報を保持する ID であり，ピア ID，文書 ID，Dewey Number の連結で構成される．

図 2 に実際の XML データを KW-DHT に格納する例を示す．例えば，ピア ID が 101 であるノードは文書番号が 1 である文書を保持しているとする．



KW-DHT

キーワード	パス式	LocationID
2008	/s/b/d	101.1.0.0.0.0
2008	/s/c/f	101.1.0.1.0.0
P2P	/s/c/f	101.1.0.1.0.0
P2P	/s/c	101.1.0.2.0
Test	/s/c/f/h	101.1.0.1.0.1.0
XML	/s/b/g	101.1.0.0.1.0
XML	/s/c/f/h	101.1.0.1.0.1.0
XML	/s/c	101.1.0.2.0

図 2 KW-DHT の例
Fig. 2 An example of KW-DHT.

4.2 パス部分が述語を含まない問合せ

次に，問合せ処理について述べる．上で説明したように，問合せのパス部分が述語を含まない単純パスかそうでないかによって処理手順が異なる．パス式のマッチングによるフィルタリングを行うかどうかの違い以外は，[5] と同一であるので，ここでは概要のみ説明する．

4.2.1 問合せ式とその答え

与えられたクエリは q=$|e_1|e_2|\cdots|e_n$ ftcontain k_1 ftand $\ldots k_m$ であるとする．ここで，e_i は要素名，k_j はキーワード，|はロケーションステップの分離記号で，/または//であるとする．

XML は木構造を有するため，キーワード検索の際には，キーワードにマッチする部分文書を決定する必要がある．与えられたキーワードにマッチする部分文書を決定するために，いくつかの概念を導入する．最小共通祖先 LCA(Lowest Common Ancestors) とは，木においていくつかのノードの共通の祖先の内，最も葉に近いノードである．LCA の中で与えられたキーワードを全て含む最小の LCA に対応する SLCA(Smallest LCA) を紹介する．

- LCA の内，与えられたノードを全てその子孫ノードに含み
- その子孫に同様に全てのノードを含むノードを持たない最小のノード である

すなわち，与えられたパス式で配下にある SLCA を検索すれば良い．

4.2.2 キーワードによる DHT の検索

まず，クエリに含まれているキーワード k_j を KW-DHT によって検索する．原理的には，各キーワードに対応するピアにおいて，そのキーワードが出現する場所の情報が得られるので，それらをまとめた上で SLCA を計算すれば良い．得られたキーワードの転置リストにおいて，クエリのパス式と単語所在パス式がマッチするかどうかを確認し，マッチした転置リストの LocationID を全て抽出する．

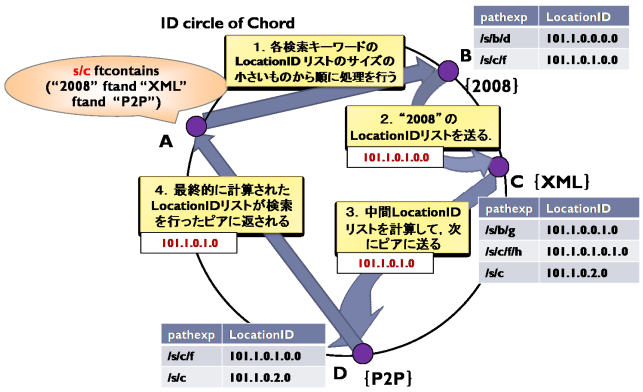


図 3　キーワードを利用した検索処理の例
Fig. 3　An example of keyword search based on DHT.

例えば，クエリは s/c ftcontains (“ 2008 ”ftand“ XML ”ftand “ P2P ”) であるとする．図 2 から，キーワード 2008 の転置リスト: (/s/b/d, 101.1.0.0.0.0), (/s/c/f, 101.1.0.1.0.0); XML の転置リスト: (/s/b/g, 101.1.0.0.1.0), (/s/c/f/h, 101.1.0.1.0.1.0), (/s/c, 101.1.0.2.0); P2P の転置リスト: (/s/c/f, 101.1.0.1.0.0), (/s/c, 101.1.0.2.0) が得られる．その中で，クエリのパス部分 s/c とマッチする転置リストの LocationID を抽出し，2008 の LocationID リスト (101.1.0.1.0.0)，XML の LocationID リスト (101.1.0.1.0.1.0, 101.1.0.2.0) と P2P の LocationID リスト (101.1.0.1.0.0, 101.1.0.2.0) が得られる．得られた LocationID を用い，P2P ネットワーク上でキーワードの SLCA を計算する．

4.2.3　LocationID による SLCA の計算方法

P2P ネットワーク上では，キーワードの転置リストが分散配置されている．与えられたキーワード集合を含む部分文書から SLCA を計算するには，LocationID をピア間でやり取りする必要がある．図 3 に例を示す．検索手順について，ピア間での通信量を最小限にするため，問合せが各検索キーワードの LocationID リストのサイズの小さいものから順に処理を行う．まず，最初のピアにおいて，KW-DHT による検索キーワードの転置リスト中の LocationID リストを抽出し，次のピアに送信する．次のピアは，送られた LocationID リストと自分が保持するキーワード集合にマッチして，LCA を探す．得られた LCA から親子もしくは祖先子孫関係にない最小部分木 (SLCA) を計算する．これを全てのキーワードについて行うことで，全キーワードを含む SLCA を計算することができる．

しかし，複数のキーワードによる SLCA の算出を行う際，転置リストの数が多くなると LocationID を送信するためのトラフィックが大量に発生してしまうという問題があるため，大規模なデータに対応できない恐れがある．

4.2.4　ブルームフィルタによる通信速度の改善

上で指摘した問題を解決するため，ブルームフィルタを利用する通信速度の改善手法について述べる．

a)　ブルームフィルタ

ブルームフィルタとは，複数の要素から構成される集合の中に，ある要素が含まれているかどうかを判定するアルゴリズムである．巨大なビット列に対し，コンテンツを表すキーワードを複数のハッシュ関数に通し，得られた値と等しい位置のビットをそれぞれ立てることでキーワードを表現する．

b)　ブルームフィルタによる転置リストのマッチング

ピア間での通信速度の改善についてブルームフィルタを XML データへどのように適用するかについて述べる．基本的には，二つ問題点がある．

- SLCA を計算するため，ブルームフィルタどのような情報を登録すれば良いか．
- ブルームフィルタでは，ハッシュ関数を用いるため，ハッシュ関数による誤検出（False Positive）がある

c)　ブルームフィルタの構成

まず，問題点 1 についての解決手法を説明する．XML データは木構造を有するため，ノード間の先祖・子孫関係を考慮して SLCA を算出しなければならない．このため，各 LocationID だけではなく，それらの全ての先祖ノードをブルームフィルタに格納する．例えば，LocationID が 101.2.0.1.1.2 とすると，0.1.1.2 による親を抽出して，ブルームフィルタに集合 (101.2.0, 101.2.0.1, 101.2.0.1.1, 101.2.0.1.1.2) を入れる．これにより，先祖・子孫関係を考慮した SLCA のマッチングが可能となる．

d)　誤検出の解消

続いて，ブルームフィルタの誤検出の解消について述べる．基本的な考え方は，以下の式が示す通りである．

$$A \cap (B \cup F(A)) = A \cup B$$

ここで，A と B は集合，$F(A)$ は誤りを含む集合 A の部分集合である．仮に $F(A)$ に偽陽性の要素が含まれていたとしても，再度 A との論理積を取ってやれば，正しい答えが得られる．

実際のキーワードを検索する場合は，まず，ブルームフィルタのみで SLCA を計算し，(誤りを含む)SLCA を求める．いったん求まった SLCA を今度は全てのピアに再度送信し，各ピアで論理積を計算することで，誤検出した LocationID を排除する．

SLCA を算出した後は一般的に LocationID の数が前より少なくなるため，例え候補解を伝送しても，LocationID のサイズがかなり小さいことが期待できる．よって，全体の検索効率を上げることができる．

4.3　パス部分が述語を含む問合せ処理

パス部分が述語を含む場合では，KW-DHT のみで処理できない．そこで，本研究では，[4] の手法を用い，パス部分が述語を含む問合せ処理を行う．

4.3.1　C-DHT と S-DHT による XML データの格納

まず，C-DHT を説明する．具体的には，XML データのノードのうち，(1) テキストノードだけを含む要素ノード，(2) 属性ノード，(3) テキストノードを含まない葉ノードなどを DHT に格納する．この際，要素名をハッシュキー，(ピア ID，文書 ID，パス式，DeweyOrder，テキスト) の組を値とする．

図 4 に XML データを C-DHT に格納した例を示す．例えば，c ノードはテキスト値“ XML and P2P ”を持ち，それについてのパス式や順序ラベルを取得することができる．さら

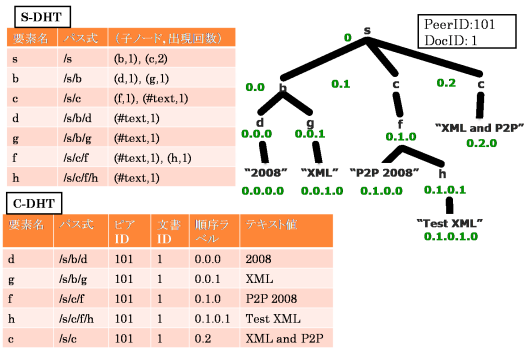


図 4　XML データから C-DHT，S-DHT を生成する
Fig. 4　Storing XML in C-DHT and S-DHT

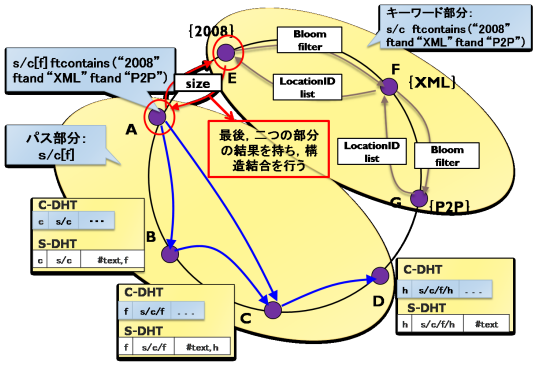


図 5　C-DHT と S-DHT による述語を含むキーワード検索処理の例
Fig. 5　Keyword search by C-DHT and S-DHT

に，任意の部分 XML データを検索，再構築するための手がかりとして，P2P ネットワーク上の全 XML データから Strong DataGuide [11] を構築し，別途 DHT に格納する．これを S-DHT と呼ぶ．具体的には，DG の各要素名をハッシュキー，（パス式，子ノード集合）の組を値として格納する．子ノード集合には，子ノードとして現れるノードを列挙するのに加えて，ノードの出現回数も記録しておく．図 4 に実際の DG を C-DHT と S-DHT に格納した例を示す．例えば，c ノードは子要素とするテキストノード と f を持ち，それぞれ一つずつ存在することが分かる．

4.3.2　C-DHT と S-DHT による XPath 検索

C-DHT と S-DHT による検索処理の概要を述べる．まず，`/`,`//`だけを含む単純パス式の場合を説明する．与えられた検索式は q=$|e_1|e_2|\cdots|e_n$ であるとする．まず，q の末端要素名 en を手がかりに，テキスト値を直接含む情報を C-DHT に探索する．続いて en を根とする部分 XML データを辿るために，S-DHT の情報を手がかりにパス式を拡張し，再度 C-DHT を参照する．具体的には，S-DHT から en の子要素である要素名を取得し，それをキーとして再度 C-DHT を参照することで，テキストを得る．これを部分 XML データの再構築が完了するまで繰り返す．

C-DHT と S-DHT を用い，パス式が述語を含む検索処理を説明する．例として，q=/a/b[c] なる問合せを考える．まず q から全てのパス式を抽出する（この例の場合は/a/b と/a/b/c になる）．得られたそれぞれのパス式に対して，上記で説明した C-DHT と S-DHT を用いる単純パスの問合せ処理を行い，解の候補集合を得る（例では，それぞれ R1, R2 とする）．次に，R1 と R2 が XML データ上で関連をもっているかどうか（先祖子孫関係にあるかどうか）を構造結合 (Structural Join) [12] によって調べる．

4.3.3　C-DHT と S-DHT による述語を含むキーワード検索

C-DHT と S-DHT によって述語を含む XPath を処理することができる．そこで，問合せののパス部分が述語を含む場合，例えば，`s/c[f] ftcontains ("2008" ftand "XML" ftand "P2P")` のような問合せの場合，問合せをパス部分 `s/c[f]` とキーワード部分 `s/c ftcontains ("2008" ftand "XML" ftand "P2P")` に分けて処理を行なうことにする．キーワード部分は上で述べたのと同様の処理を行えばよい．パス部分については，C-DHT と S-DHT により平行して処理を行ない，両者の結果が得られた後，両者の間で実際に XML データ上で関連があるかどうかを構造結合によって判定する．図 5 に実際の P2P ネットワーク上での C-DHT と S-DHT による述語を含むキーワード検索の例を示す．図 5 のように検索ピア A がパス部分とキーワード部分の二つの結果を持ち，構造結合を行う．

5.　評 価 実 験

拡張した KW-DHT の有効性を検証するために，実験評価を行った．

5.1　データセット

データセットには，サイズの大きさによって，2 種類のデータセットを用いた（データセット A と B）．データセット A は比較的小さいサイズの 2MB，23MB と 56MB の XML 文書を用いた．2MB と 23MB の XML 文書は UM XML Repository [15] で提供されているものであり．56MB の XML 文書は合成 XML 文書である．それぞれのデータセットについて，KW-DHT, C-DHT, S-DHT に記録されたデータ数は以下の通りであった．

データセット	KW-DHT	C-DHT	S-DHT
2MB	103,174	40,216	66,729
23MB	1,959,882	335,611	476,646
56MB	5,916,136	700,156	832,910

一方，大きいサイズのデータセット（データセット B）は UM XML Repository [15] で提供される Protein Sequence Database の 683MB の XML 文書を用いた．KW-DHT に記録されたキーワードは 38,215,158 個であった．

5.2　実 験 方 法

データセット A では，ネットワーク規模（ピア数）を 400 個に設定して，パス部分が述語を含まない問合せと述語を含む問合せの双方について，問合せ中のキーワードを異なる出現頻度とキーワード数によって変化させ，問合せ処理に必要となる時間コストとメッセージ数を測定した．

データセット B の場合，ピア数を 40 個とし，パス部分が述語を含まない問合せのみを実行し，時間コストとメッセージ数を測定した．

実験で用いた問合せは以下の通りである．

- 683MB・出現頻度 高
  - /ProteinDatabase/ProteinEntry/reference/accinfo /status ftcontains "GB"
  - /ProteinDatabase/ProteinEntry/reference/accinfo /status ftcontains ("GB" ftand "preliminary")
  - /ProteinDatabase/ProteinEntry/reference/accinfo /status ftcontains ("GB" ftand "preliminary" ftand "sequence")
- 683MB・出現頻度 低
  - /ProteinDatabase/ProteinEntry/reference/refinfo /title ftcontains "resolution"
  - /ProteinDatabase/ProteinEntry/reference/refinfo /title ftcontains ("resolution" ftand "biological")
  - /ProteinDatabase/ProteinEntry/reference/refinfo /title ftcontains ("central" ftand "resolution" ftand "biological")

また，パス部分が述語を含む問合せについては，データセット A のみを用いて，実験を行った．具体的には，高頻度キーワードと低頻度キーワードのそれぞれを用い，以下のクエリを試した．

- 2MB
  - /root/course_listing/restrictions [A] ftcontains "INSTR"
  - /root/course_listing//A [@HREF] ftcontains "#instructors"
- 23MB
  - /datasets/dataset/descriptions[abstract] ftcontains (" data " ftand " from ")
  - /datasets/dataset/reference[source] ftcontains " interpretation "
- 56MB
  - /site/closed_auctions/closed_auction/annotation/description[parlist] ftcontains " preventions "
  - /site/open_auctions/open_auction/annotation/description[text] ftcontains " century "

5.3 実験結果

5.3.1 パス部分が述語を含まない場合

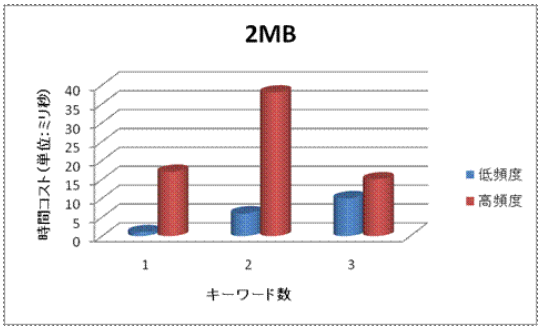


図 6　2MB 文書の問合せ処理に必要となる時間コスト
Fig. 6　The time cost needed for queries in 2MB's XML

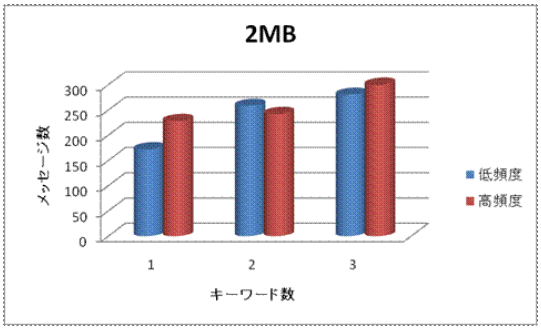


図 7　2MB 文書の問合せ処理に必要となるメッセージ数
Fig. 7　The sum of messages needed for queries in 2MB's XML

図 6 と 7，8 と 9，10 と 11，12 と 13 に異なるサイズのデータセットにおいて，問合せ処理に必要となる時間コストとメッセージ数を示す．

まず，処理時間について検討する．図 6，8，10，12 のよう

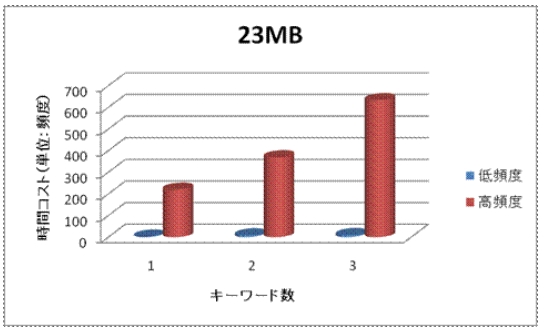


図 8　23MB 文書の問合せ処理に必要となる時間コスト
Fig. 8　The time cost needed for queries in 23MB's XML

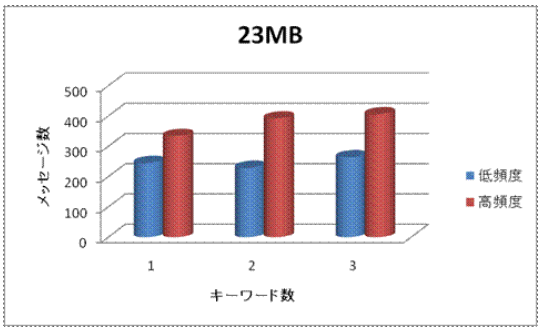


図 9　23MB 文書の問合せ処理に必要となるメッセージ数
Fig. 9　The sum of messages needed for queries in 23MB's XML

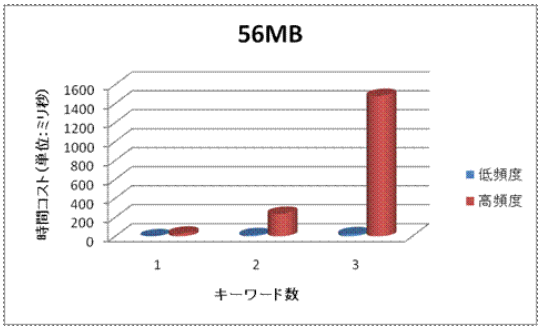


図 10　56MB 文書の問合せ処理に必要となる時間コスト
Fig. 10　The time cost needed for queries in 56MB's XML

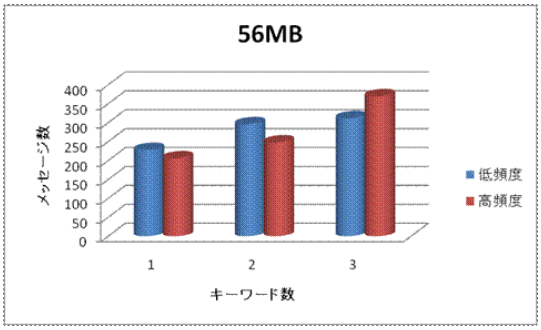


図 11　56MB 文書の問合せ処理に必要となるメッセージ数
Fig. 11　The sum of messages needed for queries in 56MB's XML

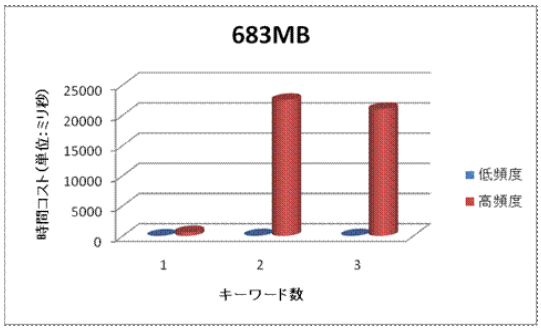


図 12　683MB 文書の問合せ処理に必要となる時間コスト
Fig. 12　The time cost needed for queries in 683MB's XML

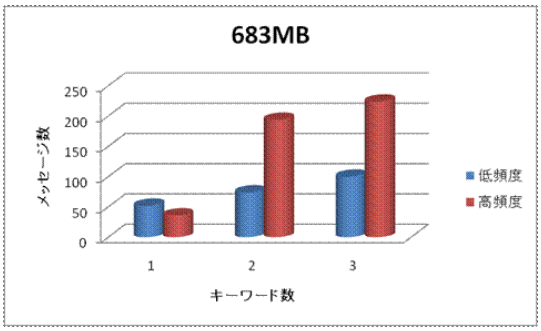


図 13　683MB 文書の問合せ処理に必要となるメッセージ数
Fig. 13　The sum of messages needed for queries in 683MB's XML

に，高頻度キーワードの方が処理時間が多くかかった．これは，高頻度キーワードの転置インデックスのサイズが低頻度の方より大きいため，そのデータ転送量の差が反映されている．次に，メッセージ数については，図 7，9，11，13 から分かるように，高頻度キーワードと低頻度キーワードの場合で大きな差はなかった．これは，問合せ式にパスが入っているため，検索キーワードの転置インデックスはそのパスに限定され，SLCA の計算には，パス部分とマッチする転置インデックスのみを用いるためである．すなわち，パス部分が述語を含まない問合せ処理に必要となるメッセージ数は単語の頻度に関係せず，キーワードがパスで指定される部分木に出現する頻度に依存するということが示唆される．

683MB のデータセット B において，クエリが `/ProteinDatabase/ProteinEntry/reference/accinfo /status ftcontains "GB"` の場合，キーワード GB の出現頻度は 291,355 であり，GB が出現するパスは以下の二つであった．

- `/ProteinDatabase/ProteinEntry/reference/accinfo /xrefs/xref/db`
- `/ProteinDatabase/ProteinEntry/reference/accinfo /status`

その中で，GB はパス 1 の出現頻度は 241,419 であり，パス 2 の出現頻度は 49,936 である．そのため，クエリ `/ProteinDatabase/ProteinEntry/reference/accinfo /status ftcontains "GB"` は 49,936 個の GB に対する検索である．

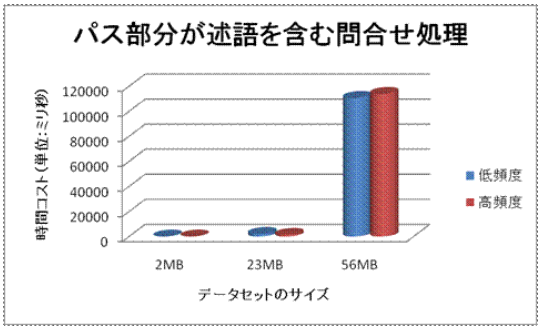


図 14　パス部分が述語を含む問合せ処理に必要な時間
Fig. 14　The time cost needed for queries whose paths contain predicates

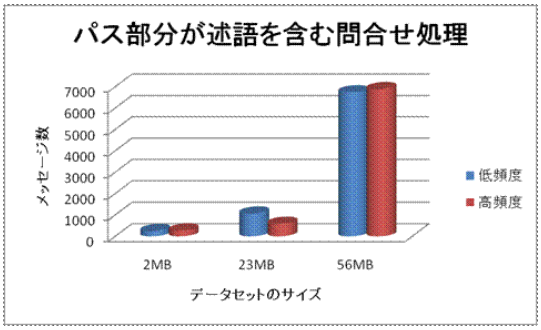


図 15　パス部分が述語を含む問合せ処理に必要なメッセージ数
Fig. 15　The sum of messages needed for queries whose paths contain predicates

5.3.2 パス部分が述語を含む場合

図 14 と 15 にパス部分が述語を含む問合せ処理に必要となる時間とメッセージ数を示す．述語を含む場合の図 14 と 15 と述語を含まない場合の図 6 と 7，8 と 9，10 と 11 を比べると，パス部分が述語を含む場合は述語を含まない場合より大量の時間コストとメッセージ数が発生してしまったことが分かる．これは，パス部分が述語を含む問合せにおいて，C-DHT と S-DHT によるパス部分の処理コストもあるためである．この際は，パス部分の出現頻度が高いなら，KW-DHT によるキーワード部分の処理コストが低くても，全体のコストが高くなってしまう可能性がある．すなわち，問合せ処理のコストはキーワード部分の出現頻度のみとは関わらず，パス部分の出現頻度とも強い関連がある．

6. ま と め

本論文では，XPath を利用する XML 文書検索手法と転置リストを利用したキーワード検索手法を統合し，キーワードを含む XPath による XML 文書検索手法を説明した．本手法の評価実験を行った．実験の結果から本手法の有効性を示した．

今後，上記の XPath に基づく XML キーワード検索では，パス部分とキーワード部分の頻度情報を利用した性能改善について検討する．また，提案手法では，ftcontains と ftand 以外の述語を含む問合せやフレーズ検索を取り扱っていないため，それらへの対応も今後の課題である．

謝辞 本研究の一部は科学研究費補助金若手研究（# 19700083），科学技術振興機構 CREST「自律連合型基盤システムの構築」による．

文 献

[1] Y. Xu and Y. Papakonstantinou. Efficient keyword search for smallest LCAs in XML databases. In Proceedings of the 2005 ACM SIGMOD international conference on Management of data, pages 527-538, 2005.

[2] Reynolds, P., Vahdat, A. Efficient peer-to-peer keyword searching. Technical Report 2002, Duke University, CS Department, Feb. 2002.

[3] Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H. Chord: A scalable peer-to-peer lookup service for Internet applications. In Proc. ACM SIGCOMM '01, San Diego, CA, Aug. 2001.

[4] T. Amagasa, C. Wu, and H. Kitagawa. Retrieving arbitrary XML fragments from structured peer-to-peer networks. In Joint Conference of the 9th Asia-PacificWeb Conference and the 8th International Conference on Web-Age Information Management (APWeb/WAIM 2007), pages 317-328, 2007.

[5] Xiaochen Li, Toshiyuki Amagasa, and Hiroyuki Kitagawa, "Searching XML Documents by Keywords in Structured P2P Networks", 3rd International Workshop on XML Data Management Tools and Techniques (XANTEC 2008), Turin, Italy, Sept 1 - 5, 2008.

[6] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In In the 18th IFIP/ACM International Conference on Distributed Systems Platforms(Middleware), pages 329- 350, 2001.

[7] W3C. XQuery 1.0: An XML Query Language, http://www.w3.org/TR/xquery/, January 2007, Recommendation.

[8] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In ACM SIGCOMM 2001, pages 161-172, 2001.

[9] W3C. XQery and XPath Full-Text Language Version 1.0. http://www.w3.org/TR/xpath-full-text-10, May 2008. Recommendation.

[10] W3C. XML Path Language (XPath) Version 1.0. http://www.w3.org/TR/xpath.html, November 1999. Recommendation.

[11] R. Goldman and J. Widom, DataGuides:Enabling Query Formulation and Optimization in Semistructured Databases. Proc. V LDB, pp.436-445(1997).

[12] S. Al-Khalifa, H. Jagadish, N. Koudas, J. Patel, D. Srivastava and Y. Wu, Structural Joins: A Primitive for Efficient XML Query Pattern Matching. Proc. ICDE, 2002.

[13] S. Abiteboul, I. Manolescu, and N. Preda, " Constructing and querying peer-to-peer warehouses of XML resources. " in SWDB, 2004.

[14] Abiteboul, S. Manolescu, I. Polyzotis, N. Preda, N. Chong Sun. XML processing in DHT networks. Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference. pp.606-615, 7-12 April 2008

[15] UW XML Repository: http://www.cs.washington.edu/ research/xmldatasets/www/repository.html

[16] Overlay Weaver: http://overlayweaver.sourceforge.net

[17] 李暁晨，天笠俊之，北川博之「構造型 P2P ネットワークにおけるキーワードを含む XPath による XML 文書検索」，iDB フォーラム 2008．2008 年 9 月 21 日〜23 日．