構造型 P2P ネットワークにおける負荷分散を考慮した XML データ処理

Storage and Retrieval of XML Data in Structured P2P Networks with Load Balancing

呉 俊輝♥ 天笠 俊之♦ 北川 博之♦
Chunhui WU Toshiyuki AMAGASA
Hiroyuki KITAGAWA

近年, P2P 技術を利用した情報コンテンツの流通に関する研究，開発が盛んに行われている．XML はコンテンツやデータの記述に標準的に用いられてようになっており, P2P 環境における XML データ管理は重要な問題である．このような背景から，我々はこれまで分散ハッシュ表を利用した XML データの効率的な格納・検索手法を提案してきた．しかし，ピア毎に格納されるデータ量に偏りがあるという問題があった. そこで本論文では, ピア間の負荷分散を図るため，拡張可能ハッシュを利用した XML データの処理手法を提案する．具体的には，ピアに格納されたデータを，拡張可能ハッシュに基づき他ピアに再配分する．これによって，各ピアが保持するデータ量を均衡化する．本論文では，提案手法の詳細を述べ，実験による評価によってその有効性を示す.

To date, there has been extensive work on information management in P2P networks. In such systems, XML plays an important role for data representation and exchange, and XML data management in P2P networks is therefore an important research issue. For this reason, we have proposed a scheme for storage and retrieval of XML data using DHT (Distributed Hash Table). However, it has been pointed out that it has a drawback that storage-load distribution might be skewed. To cope with this problem, in this paper, we propose a scheme for storage-load balancing method based on extendible hashing. Specifically, for each peer, we redistribute overflowed data to other peers by means of extendible hashing. We show the effectiveness of the method by experiments.

1. はじめに

Peer-to-Peer (P2P) ネットワークは, インターネットあるいはモバイル環境における分散コンピューティングの基盤技術として活発に研究されている．P2Pネットワークは，従来のクライアントサーバ型システムとの対比で，完全分散システムであるピュア型と，両者の中間的な性質を持つハイブリッド型に分類される．ハイブリッド型P2Pシステムでは，データやピアの情報を管理する中央サーバを仮定しており，全ての検索処理は仲介サーバを通じて行われる．このため，システムはシンプルであるが，中央サーバの故障やネットワークの局所的な故障がサービス全体の停止の要因となることがある．一方，ピュア型P2Pシステムでは，ピア同士がサーバを介さずに動作するため，局所的な障害があっても全体サービスは持続するという特徴を持つ．ピュア型P2Pシステムは，さらに非構造型と構造型に分類することができる．

構造型P2Pネットワークを実現する手法の一つに，分散ハッシュ表（Distributed Hash Table; DHT）がある．DHTでは, あらゆるピアやデータはハッシュ関数によってID空間に写像され, ピアはID空間のうち特定の部分空間に写像されたデータを管理する．データの検索は，各ピアが保持するルーティング表に基づいて検索要求を転送することで行われる．モデルが単純であることや，検索効率が良いことなどから，近年活発に研究されている．

他方，XML (Extensible Markup Language) [1] は，標準のデータ記述，交換フォーマットとして，様々な分野で広く用いられるようになった．P2Pネットワークにおいても，XML形式で記述されたデータ, あるいはメタデータを利用することも少なくない．例えば，いくつかの科学分野では，実験や観測データをXMLで記述して, P2Pネットワークを通じて交換するという応用が出始めている．このことから，今後はP2P環境におけるXMLデータの効率的な格納・検索が重要になることが予想される．

このような背景から, 我々はP2P環境におけるXMLデータの格納・検索手法を提案し，実装や実験によりその有効性を検証してきた[2][3]．基本的には，XMLデータからテキストノードと（テキスト以外の）リーフノードを抽出する．それぞれのノードについて，要素名をハッシュキー，その要素に至るパス式およびテキスト値等の関連情報を値としてDHTに格納する．さらに，XMLデータの構造を格納するために，Strong DataGuideに基づく構造情報もDHTに格納する．これにより，任意のCore XPath式を効率的に処理することができる．ただし一般に，ある要素名を持つノード数には偏りがあるため，大規模なデータにおいては，データが少数のピアに偏ってしまう欠点を持つ．

そこで本論文は，上記手法におけるストレージ負荷の均衡化を目的として，拡張可能ハッシュ法を利用した方式を提案する．基本的には，前述の方式で各ピアにデータを格納する際に，拡張可能ハッシュに基づき一定量を超えるデータを他ピアに分散する．

2. 基本的事項

2.1 分散ハッシュ表（DHT）

DHTは，構造型P2Pネットワークに分類される，P2Pネットワーク上のオブジェクト管理手法である．DHTでは，ピアと検索対象となるオブジェクトを，オーバレイネットワークと呼ばれる仮想的なネットワークの上に配置する．その位置は，各オブジェクト（ピア）のファイル名やIPアドレス等のオブジェクト識別子にハッシュ関数を適用して得られる値に基づいて決められる．検索処理は，各ピアがルーティング表を利用し，検索したいオブジェクトを管理するピアへ検索リクエストを転送することで実現される．DHTには，オーバレイネットワークや経路表の構成法に様々な提案があるが，本研究ではChord [4] を用いる（図1）．

♥ 学生会員 筑波大学システム情報工学研究科．現在，日本電気（NEC）勤務 chunhuiwu@kde.cs.tsukuba.ac.jp
♦ 正会員 筑波大学システム情報工学研究科／計算科学研究センター {amagasa, kitagawa}@cs.tsukuba.ac.jp

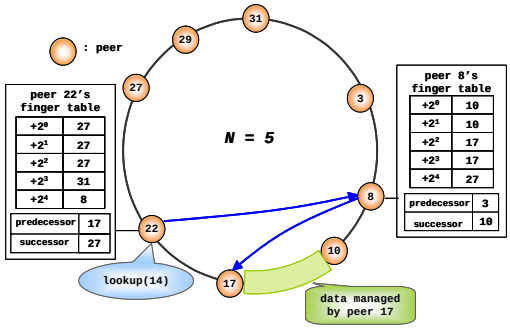


図 1　Chord の ID サークル.
Fig. 1　An ID Circle of Chord.

Chord では，オーバレイネットワークとして，2^N 個の ID から構成される ID サークルを用いる．ここで N はスケールファクタと呼ばれる．オブジェクトの位置はハッシュ関数によって決められ，ID = i のピアは，ID サークル上で直前に位置するピア(predecessor)から i−1 までのオブジェクトを担当する．各ピアは，担当するデータの ID，ルーティング表（フィンガーテーブル），前後に位置するピア(predecessor, successor) の ID を保持している．また，フィンガーテーブルには ID = $i + 2^k$ (k = 0, 1, 2, …, N−1) のオブジェクトを管理するピアへのリンクが存在する．Chord における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．

2.2 構造概要

XMLデータ検索の効率化のため，XMLの構造を考慮した索引手法が研究されている．これらは，XMLデータの構造概要 (Structural Summary) と呼ばれる．本論文では，その中で最も単純な一つであるStrong DataGuide [5] を利用する．Strong DataGuideは，基本的には，XMLデータのルートノードからリーフノードに至る全てのパス式を集約して得られる木構造である．以下ではStrong DataGuideをDGと呼ぶ．

2.3 拡張可能ハッシュ

拡張可能ハッシュ (Extendible Hashing) [6] は，外部記憶探索に使われるデータ管理手法の一つである（図 2）．ディレクトリとデータページ（以下は単にページと呼ぶ）の 2 階層のデータ構造と，キーを引数とするハッシュ関数から構成される．ディレクトリはページへのポインタを持つ配列で，キーの数に応じて配列の大きさが動的に変化する．d 次の拡張可能ハッシュにおけるディレクトリは，2^d 個のページへの参照を持つ．参照される各ページにおいては，格納されるオブジェクトのキーの先頭 d ビットがディレクトリの対応するキーと一致するようになっている．

キーの挿入は以下のように行われる．まずハッシュ関数によって固定長のビット列であるキーを生成し，上位ビットを取り出す．ビット数 d は，ディレクトリの「深さ」とも呼ばれる．取り出したビット列をディレクトリ配列の引数として使い，その配列のもつポインタが指すページにオブジェクトを挿入する．検索および削除も同様の方法でページを特定する．キー数が増加するとページが溢れるが，その場合はディレクトリの大きさを深さを増やすことによって変化させ，ページを分割することで対処する．また反対に，キー数が減少し空き領域ができると，溢れて分割されたページを統合する．

3. DHT による XML データの格納・検索

我々が[2]で提案した，DHT を用いた XML データの格納・検索手法の概要を説明する．基本的には，XML データから全てのテキストノード，およびテキストノード以外のリーフノードを抽出する．その各ノードについて，ノード値，木ラベル等の関連情報を，要素名をキーとして DHT に格納する．ただし，このままでは中間ノードに関する問合せが処理できないので，XML データから構造情報として DG を構築し，別途 DHT に格納する．

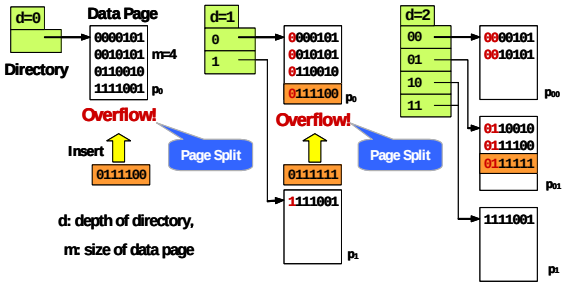


図 2　拡張可能ハッシュ.
Fig.2　Extensible Hashing.

3.1 C-DHT による XML データの格納

XMLデータを構成するノードのうち，1) テキストノード，2) 属性ノード，3) テキストノードでないリーフノードを，要素名をキーとして DHT に格納する（図 3）．ここで，属性ノードの場合は，“@”+属性名をキーとする．例えば name 属性なら “@name”となる．このための DHT をここではC-DHT (Content-DHT) と呼ぶ．各ノードについて格納する情報としては〈ピア ID，文書 ID，パス式，木ラベル，テキスト値〉を用いる．要素名（属性名）をハッシュキーとすることで，その内容を保持しているピア ID および関連情報を取得することができる．なお，木ラベルには Dewey オーダ[7]を利用しているが，その他の木ラベルも利用可能である．

3.2 S-DHT による構造概要の格納

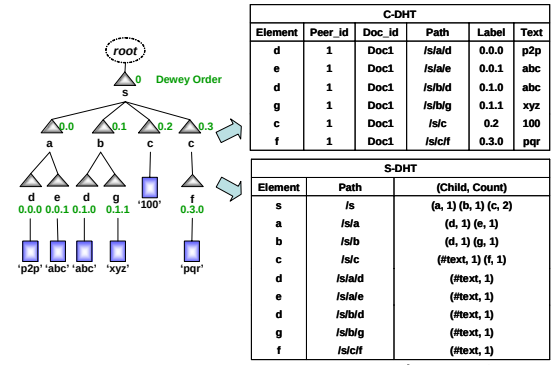
S-DHT (Structure-DHT) は，P2P ネットワーク上のピアが保持している全 XML データを元に導出される DG を格納する DHT である（図 3）．具体的には，DG の各ノードの要素名（属性名）をキー，〈パス式，子ノード集合〉を値として格納する．子ノード集合には，要素名，属性名に加えて，ノードの出現回数も記録しておく．これは，問合せ最適化の際の統計情報として用いることができるが，詳細は割愛する．

3.3 検索処理

C-DHT と S-DHT による検索処理の概要を述べる（図 4）．簡単のため，child 軸 (/) および descendant 軸 (//) だけを含む Core XPath 式の場合を説明する．検索式は $q = |e_1|e_2|\ldots|e_n$ の形で表される．ここで，e_iは要素名，“|” はロケーションステップの分離記号であり，“/” または “//” である．まず，q の末端要素名 e_n をキーして C-DHT を参照し，候補ノード集合を取得する．その中で，パス式が q とマッチするものが検索式に該当するノードである．続いて，e_n の子ノードを探索するため，S-DHT を参照する．もし子ノードが見つかれば，その結果を用いて検索パス式を拡張し，再度 C-DHT を参照する．これを部分 XML データの再構築が完了するまで繰り返す．

3.4 本手法の問題点

上記手法を評価したところ，良好な検索性能を持つことが示された [3]，しかしながら，要素名（属性名）をハッシュキーとしてデータを配置するため，同一の要素名（属性名）を持つノードは全て同じピアに格納される．これは，ある限られた文書スキーマに従う大量の XML データを格納する場合，大量のデータが特定のピアに偏って配置されてしまうことが予想される．



C-DHT

Element	Peer_id	Doc_id	Path	Label	Text
d	1	Doc1	/s/a/d	0.0.0	p2p
e	1	Doc1	/s/a/e	0.0.1	abc
d	1	Doc1	/s/b/d	0.1.0	abc
g	1	Doc1	/s/b/g	0.1.1	xyz
c	1	Doc1	/s/c	0.2	100
f	1	Doc1	/s/c/f	0.3.0	pqr

S-DHT

Element	Path	(Child, Count)
s	/s	(a, 1) (b, 1) (c, 2)
a	/s/a	(d, 1) (e, 1)
b	/s/b	(d, 1) (g, 1)
c	/s/c	(#text, 1) (f, 1)
d	/s/a/d	(#text, 1)
e	/s/a/e	(#text, 1)
d	/s/b/d	(#text, 1)
g	/s/b/g	(#text, 1)
f	/s/c/f	(#text, 1)

図 3　C-DHT, S-DHT による XML データの格納.
Fig. 3　Storing XML Data in C-DHT and S-DHT.

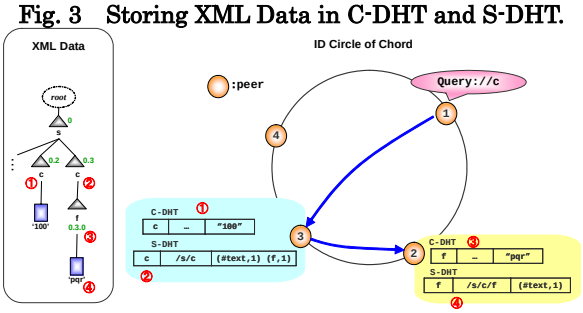


図 4　検索式 q = //c の処理過程.
Fig.4　Query processing of q = //c.

4. 拡張可能ハッシュに基づくストレージ負荷分散方式

4.1　提案手法

上で述べた問題に対処するため，拡張可能ハッシュを利用した負荷分散方式を提案する（図 5）．まず，各ピアは決められたサイズのデータ領域を複数持つものとする．これがデータページに相当する．ディレクトリから参照されるページは，ピアをまたがっても良い点が通常の拡張可能ハッシュとは異なる．

初期状態では，ピアはディレクトリとページを各 1 つ持つ．3 節で述べた手法に基づき，要素 e に関する情報を DHT に格納するとする．このとき，e の絶対パス式 p_e からハッシュ関数 f によって n ビットのビット列を生成し，この値を元に拡張可能ハッシュへの挿入を行う．同様の処理が繰り返され，ページが溢れるとページが分割される．具体的にはインデックスの先頭の 1 ビットが 0 か 1 かによってエントリを分割する．このとき，0 に相当するページはそのピアに保持するが，1 に相当するページを他のピアに配分することで，ストレージ負荷の分散を図る．具体的には，ページの先頭のエントリのパス式をキーとして C-DHT のピアを探索する．見つかったピアにページを転送するとともに，ディレクトリのエントリにキーとして与えたパス式とページ内のエントリ数を記録する．後者は実際にリモートのピアにあるページを参照せずページ溢れを検出するために必要である．仮に，見つかったピアのストレージが過負荷の場合，ページ内の 2 番目，3 番目のエントリのパス式の順に，そのページを引き受けることができるピアが見つかるまで探索を続ける．

4.2　検索処理

検索処理は，基本的に 3.3 節で説明した処理手順に基づくが，拡張可能ハッシュによって他ピアに配置されたデータを取得する手順が増えるところが異なる．検索式を q = |e1|e2|...|en とする．まず，末端要素 en を用いて 3.3 節の手順で en に関するデータを保持しているピア p を特定する．次に，p のディレクトリの各エントリのうち en のパス式とマッチングするものを探し，これらをキーとして実際のページを保持するピアを特定する．

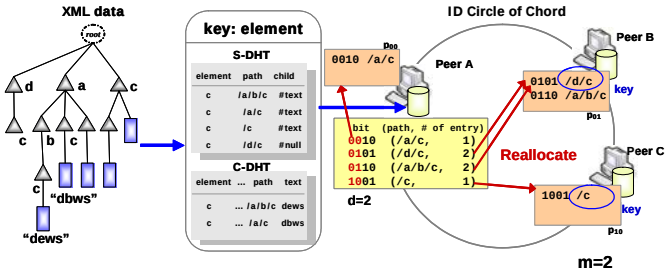


図 5　拡張可能ハッシュによるデータの再配置.
Fig.5　Data-reallocation by extensible hashing.

5. 実験

提案手法の有効性を検証するため，実験による評価を行った．具体的には，3 節で述べた手法，4 節で述べた手法，関連研究の XP2P [8] を用いて，それぞれ 1) P2P ネットワーク内のデータ分布，2) 検索処理する必要なメッセージ数を比較した．

5.1　実験環境

Overlay Weaver [9] のネットワークシミュレータにより，仮想的な DHT ネットワークを構築して実験を行った．データセットは XMark [10] による人工データ（サイズ：5MB，タグの種類:83 個）を用いた．C-DHT のエントリ数は 78,837 件，S-DHT のエントリ数は 517 件であった．

5.2　データ分布の比較

P2P ネットワークのピア数(N)と，ページ領域のサイズ(m)を変化させたときのピア間のデータ分布を測定した(図 6)．N = 100 のとき，要素名をハッシュキーとする手法では，あるピアは約 15,800 件のデータを保持し，4 つのピアが約 8,000 件のデータを保持していた．また，全体の 1/20 のピアが総データ量の 2/3 を保持しており，分布に偏りがあることが分かった．また，3/4 のピアはデータを保持しなかった．これに対して，拡張可能ハッシュによる方式では，m = 8 のとき，1/3 のピアが全データ量の半分を保持し，何も持たないピアは 59 個であった．m = 4 の場合，1/3 のピアが全データの 1/3 を保持し，何も持たないピアは半数以下に低減した．一方，XP2P では一意のパス式をハッシュキーとし，得られた値が重複することが少ないため，ピア間のデータ分布は均等であった．N = 200 の場合でも同様の傾向を示した．

5.3　検索処理性能

各方式の検索性能を，検索処理に必要なメッセージ数によって測定した．簡単のため，検索式を child 軸 (/) と descendant 軸 (//) を含むものに限定した．

図 7 に child 軸のみを含む検索式 Q_1 の処理結果を示す．Q_1 では，元データでは name 要素はテキストのみを含むため，要素名をハッシュキーとする提案手法および拡張可能ハッシュ方式との組合せでは，少ないネットワーク参照回数で問合せ処理を行うことができる．これに対して，XP2P では，（問合せパス式と DHT のハッシュキーが一致しない場合）全てのピアに対して全数検索が必要となる．このため，問合せ処理のためにより多くのメッセージ数が必要となることが示されている．

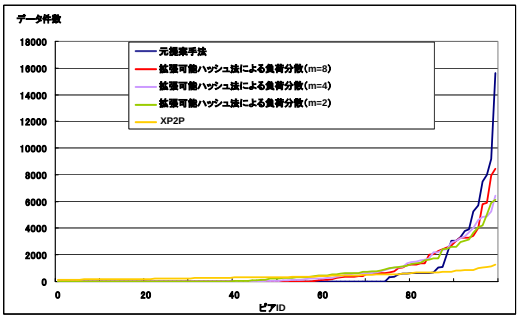


図 6　ピア間のデータ分布（N=100）
Fig. 6　Load distribution among peers (N=100)

表1　ベンチマーククエリ．
Tab. 1　Benchmark query.

項目		ベンチマーククエリ
child 軸	Q_1	/site/people/person/name
descendant 軸	Q_2	//asia

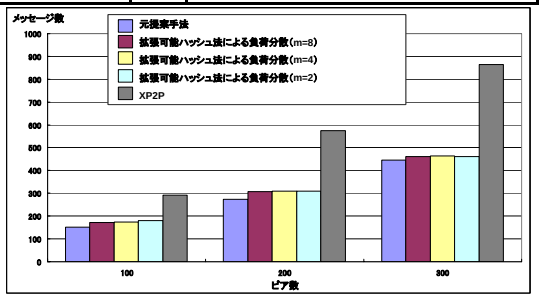


図 7　メッセージ数：Q_1
Fig. 7　# of messages: Q_1

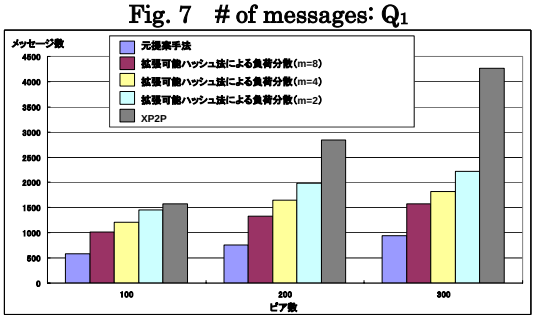


図 8　メッセージ数：Q_2
Fig. 8　# of messages: Q_2

図 8 に descendant 軸の検索式 Q_2 の処理結果を示す．元々の提案手法では，検索式の末端要素による DHT の参照を一度行うだけで済むが，拡張可能ハッシュによる方式では，ページが複数のピアに分散しているだめ，検索処理に必要なメッセージ数が多くなる．ページサイズ m を小さくすると，ピア間のデータ量に偏りが緩和されるが，検索に必要なメッセージ数はさらに増加する傾向がみられる．

一方，XP2P は，上でも指摘したように，decendant 軸を含む問合せでは，全ピアにアクセスする必要があるため，我々の手法に比べてメッセージ数が多いことが分かる．

6. まとめと今後の課題

本論文では，拡張可能ハッシュによるストレージ負荷分散方式を提案し，実験によりその有効性や性能について検証した．実験結果から，ピア間のデータ量の偏りが緩和されることが分かった．しかしながら，データがより多くのピアへ配分されているため，検索処理に必要な DHT の参照回数が増え，ネットワークのメッセージ数が増大してしまう．今後は，より大規模なネットワーク環境や多様なデータによる性能評価，XQuery への対応や問合せ最適化手法の検討などが挙げられる．

[謝辞]

本論文の一部は，科学技術振興機構・戦略的創造研究推進事業（CREST）「自律連合型基盤システムの構築」，および科学研究費補助金　萌芽研究（#18650018），若手研究（B）(#19700083)，基盤研究（A）(#17200002)によるものである．

[文献]

[1] World Wide Web Consortium: Extensible Markup Language (XML) 1.0, http://www.w3.org/TR/REC-xml. W3C Recommendation 04 February. 2004
[2] C. Wu and T. Amagasa and H. Kitagawa: “XML Retrieval Using Structural Summary in Peer-to-Peer Environment”, Proc. DEWS2006 (2006).
[3] 呉俊輝，天笠俊之，北川博之：“P2P 環境における構造概要を利用した XML データの検索手法の実装について", 夏のデータベースワークショップ(DBWS2006).
[4] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek and H. Balakrishna: “Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications”, Proc. ACM SIGCOM, pp.149-160, Aug. 2001.
[5] R. Goldman and J. Wido: “DataGuides: Query Formulation and Optimization in Semistructured Database”, Proc. VLDB, pp.436-445, Aug. 1997.
[6] R. Fagin, J. Nievergelt, N. Pippenger and H. R. Strong: “Extendible Hashing-A Fast Access Method for Dynamic File”, ACM Transactions on Database Systems (TODS), 4(3), pp.315-344, Sep. 1979.
[7] I. Tatarinov, S. Viglas, K. Beyer, J. Shanmugasundaram, E. Shekita and C. Zhang: “Storing and Querying Ordered XML Using a Relational Database System”, Proc. SIGMOD, pp.204-215, Jun. 2002.
[8] A. Bonifati, U. Matrangolo, A. Cuzzocrea, and M. Jain. “XPath Lookup Quries in P2P Networks”. Proc. WIDM, pp. 48–55, Nov. 2004.
[9] 首藤一幸，独立行政法人産業技術総合研究所. http://overlayweaver.sourceforge.net, 2006.
[10] XMark - An XML Benchmark Project: http://monetdb.cwi.nl/xml, 2003

呉 俊輝　Chunhui WU
筑波大学大学院システム情報工学研究科博士前期課程修了．P2P 環境における XML データ管理に関する研究に従事．日本データベース学会学生会員．現在，日本電気（NEC）勤務．

天笠 俊之　Toshiyuki AMAGASA
筑波大学大学院システム情報工学研究科，計算科学研究センター講師．データベースシステムの研究に従事．情報処理学会，電子情報通信学会，日本データベース学会各会員．

北川 博之　Hiroyuki KITAGAWA
筑波大学大学院システム情報工学研究科，計算科学研究センター教授．理学博士（東京大学）．異種情報源統合，データマイニング，文書データベース，情報検索などの研究に従事．情報処理学会および電子情報通信学会フェロー．日本データベース学会副会長．電子情報通信学会，日本ソフトウェア科学会，ACM，IEEE CS 各会員．著書に「データベースシステム」（昭晃堂）など．