PCクラスタを用いたXMLデータ並列処理方式の評価

An Evaluation of a Scheme for Parallel Processing of XML Data using PC Clusters

城戸 健太郎♥　天笠 俊之♦
北川 博之♦

Kentarou KIDO　Toshiyuki AMAGASA
Hiroyuki KITAGAWA

近年，XMLは標準のデータ記述フォーマットとして急速に普及しており，数百MBから数GB サイズの大規模なデータをXMLで記述することが一般的に行われるようになってきている．このような巨大なXMLデータに対して効率的に検索を行う手法として，我々はこれまでPCクラスタを用いたXMLデータの並列処理方式を提案してきた．本論文では，提案方式におけるXPath問合せのコスト計算の詳細を定めるとともに，MPIとlibpqを用いて実装を行い，より詳細な性能評価を行う．

Recently, with the rapid spread of XML format, it has become popular that large-scale data, whose sizes range from several hundreds of MB to several GB, are described by XML. For the purpose of efficient query processing of huge XML data, we have proposed a scheme for parallel processing of XML data using PC-clusters. In this paper, we attempt to give a concrete cost model by which we can choose efficient query execution plans, discuss an implementation using MPI and libpq, and report an experimental evaluation.

1. はじめに

XML (Extensible Markup Language) [1]は，その仕様の簡明さからデータ記述フォーマットとして急速に普及し，関連技術の研究及び実装が盛んに行われている．最近では，大規模なデータをXMLで記述することも多くなっており，数百MBから数GBのデータサイズを持つ大規模なものも少なくない．天文学での観測記録，バイオインフォマティクスにおける遺伝子データなどがその記述フォーマットの例として挙げられる．そのような背景から，大規模なXMLデータを効率的に扱うことのできるXMLデータベースが重要となってきている．

XMLデータベースの実現方法には何通りかの方法が提案されている．中でも関係データベースを利用する方法は，すでに稼動しているシステムが多数あることや，既存の情報資源との連携が取りやすいことなどから，主要な実現方法のひとつである．例えば，関係XMLデータベースを構築する方法として，XMLデータをノード単位に分解し，それぞれをルートノードからの経路式とともに関係表にマッピングする手法（経路アプローチ）がある[2]．この手法の利点は，実用的なXPathの部分クラスをSQLの機能だけで処理可能な点である．商用システムでもMicrosoft SQLServer 2005 がこの手法を採用している．しかし，関係データベースにおけるXMLデータの処理はいくつかの理由から高コストであるため，大規模なXMLデータでは処理効率が悪化することが指摘されている[3]．

この問題に対処するため，我々の研究グループは大規模XMLデータの高速な処理を目的として，PCクラスタによるXMLデータの並列処理方式を提案してきた[4]．具体的には，XMLデータの分割方式を定義し，それに基づいて分割を行う．得られたXMLデータフラグメントは，問合せ（XPath）のコスト関数をもとに計算ノードへ配置する．本論文では，XPath問合せのコスト計算の詳細を定めるとともに，MPIとlibpq を用いて実装を行い，より詳細な性能評価を行う．また，本手法の台数効果を調べ，処理のボトルネックを調べるために問合せ処理時間の内訳について調査を行った．

2. 準備

2.1 DataGuide

XMLデータの問合せ処理を効率的に行うためには，XMLデータ全体の構造を知る手がかりが必要になる．このため，本論文では構造概要と呼ばれる半構造データのための索引構造を利用する．構造概要はこれまで多くが提案されているが，ここではStrong DataGuide[5]を用いる．Strong DataGuideとは，情報源において共通のラベル経路を持つノードを一つのノードに集約し，木構造表現したものである．定義等の詳細については文献[5]を参照されたい．本論文ではStrong DataGuideに各経路の出現回数を統計情報として格納し，3.3節で述べるコストの計算の際に利用する．

2.2 関係データベースへのXMLデータの格納

XMLを関係データベースに格納する手法はこれまで多数提案されている．中でもXMLデータを経路式に基づき関係データベースにマッピングすることで，DTDや要素の型に依存することなくXMLデータを格納，検索することができる経路アプローチがある[2]．この手法ではXMLデータを解析して得られる木構造をノード単位に分解し，関係表に格納する．このとき，XMLデータに対してラベル付けを行うことで，マップされた関係表からもとのXMLデータの構造を再構築することが可能となる．本研究ではXRelを参考に，XMLデータをNodeTable(did, pid, nodeid, nodenum, type, value)，PathTable(pid,pexp)の2つの表に格納する．NodeTableの各属性は，XML文書ID，経路式ID，ノードID，Dewey Order[6]によるノードラベル，要素，属性などのノードの種類，テキスト値を表している．PathTableはすべての経路式とそのIDを格納する．XMLデータを関係データベースに格納した例を図1に示す．

問合せ処理については，XPath式をSQLに変換することにより，関係データベースの機能を用いて評価する．具体的には，XPathを解析することによって得られる問合せグラフをSQLに変換する．ここでは話を簡略化するため，述語の入れ子を含まない問合せについて考える．本論文ではchild軸，descendant軸，および述語を含む問合せを処理の対象とする．問合せグラフはラベルノード，述語ノード，出力ノードの三つのノードからなる．ラベルノードは経路式に対応し，[]に対応するノードを述語ノードと呼ぶ．最終的に出力されるノードを出力ノードと呼び，問合せグラフ中には必ず一つ，出

♥学生会員　筑波大学システム情報工学研究科
kido@kde.cs.tsukuba.ac.jp
♦正会員　筑波大学システム情報工学研究科　筑波大学計算科学研究センター
{amagasa,kitagawa}@cs.tsukuba.ac.jp

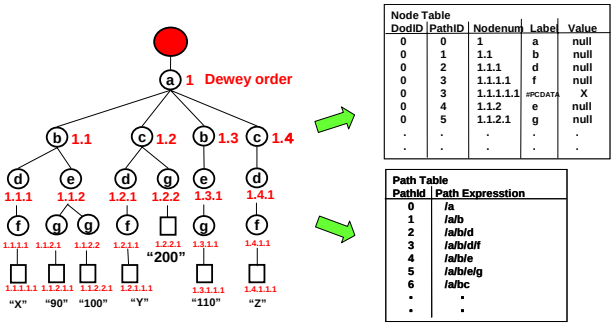

図1 経路式に基づく関係表へのマッピング
Fig.1 Mapping XML data to relational table by the path approach.

力ノードが存在する．

3. 提案手法

3.1 提案手法の概要

図2に提案手法の概要を示す．システムは無共有型のPCクラスタを想定し，各計算ノードには関係データベースが搭載されているとする．システムには入力として検索対象となるXMLデータと，問合せワークロードが与えられる．システムは，まず経路アプローチに基づきXMLデータを関係表にマップする．その際，XMLデータの構造概要であるDataGuideを同時に生成する．次に，問合せワークロードとDataGuideの情報に基づき関係表を分割し，各計算ノードに割り当てる．問合せ処理を行う際には，まず問合せから問合せプランを生成する．次に，問合せの内容からデータを保持する計算ノードを特定し，その計算ノードに処理オペレータを転送することによって，並列的な処理を行う．

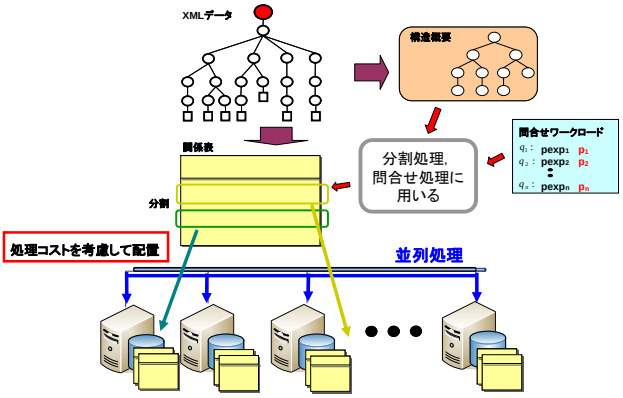

図2 システムの概要
Fig.2 System overview.

3.2 問合せワークロード

本論文では，問合せワークロードをXPath式とその発行頻度の対の集合であると定義する．発行頻度は問合せの発行履歴の統計から算出した確率を元に，上位n件の問合せを典型的な問合せセットとし，その発生確率を合計が1になるように正規化する．

3.3 XPath問合せの処理プランとコスト算出

次に，XPath問合せの処理プランとコスト計算について説明する．本システムでは，問合せプランは問合せグラフから導出される．図3の問合せグラフを例として説明すると，まず，問合せグラフの各ラベルノードを，関係表から対応するXMLノードを選択する処理に置き換える．次に，親子間の

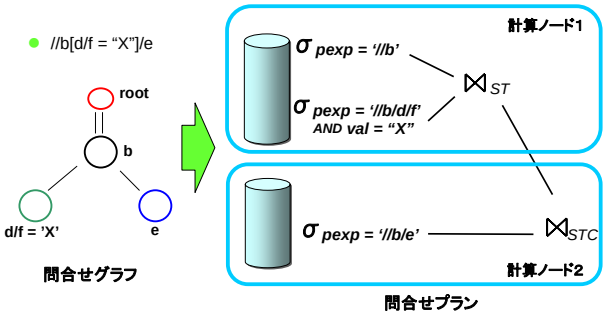

図3 問合せ処理プランの生成
Fig.3 Query plan generation.

表1 処理オペレータに対するコスト
Table 1 Cost function for operators.

オペレーション	コスト	
	送信側	受信側
$\sigma_{pathexp}$(R)	|R|	−
$\bowtie_{ST}$(R1,R2)	|R1|x|R2|	−
$\bowtie_{STC}$(R1,R2)	|R1| x log|R1| + Trans(|R1|)	|R2| x log|R2| + Trans(|R1|) + |R1| + |R2|
Trans(R)	α|R|	α|R|

エッジをノード集合間の構造結合に置き換える．構造結合とは，候補タプルのうちXMLデータ上で実際に結合されているペアのみを残す操作である．このとき，図ように，あるラベルノードが2つの子を持つ場合は2段階の構造結合が必要になるが，その順番は後述のコスト関数に基づいてよりコストが少さく見積もられる方のプランを選択することになる．出力ノードが得られるまで同様の処理を繰り返すことによって，問合せプランが得られる．

ここで，得られたXPath問合せプランから，実際にコストを算出する方法を説明する．

選択処理のコスト：この処理は，関係表の走査を行うと仮定して，それぞれ選択対象の関係表のサイズ（全タプル数）として見積もる．選択演算の結果得られるタプル数（次の演算子の入力として使われる）は，選択の条件で与えられた経路式に該当するノード数であるので，DataGuideによってその数を推定する．なお，選択演算に[]による条件がついている場合は，既存のヒストグラム等の統計情報を用いた推定手法を用いることが考えられる．今回の実装では簡単のため，PostgreSQLで用いられているコスト関数を参考にして，等号の場合は0.5%，不等号の場合は33.3%といった定数係数を用いることにする．

構造結合のコスト：構造結合は，結合の対象である関係表が同じ計算ノードにある場合とない場合とで処理が異なる．

(ア) 結合の対象となる関係表が同じ計算ノードにある場合，処理コストはアルゴリズムが入れ子ループ結合であると仮定して，二つの関係表のタプル数の積で表す．また，結果のノード数は構造結合しようとするノード集合のうち，子孫にあたるノード集合のサイズであるとする．

(イ) 同じ計算ノードに無い場合は，結合のコストに加えて関係表を転送するための通信コストが必要となる．この場合，通信コストを最小化するために，二つの関係表のうちどちらかデータ量の少ない方を転送する．通信コスト($Trans(R)$)は送信するタプル数に重みをかけたものとし($\alpha|R|$)，送信，受信する両方のノードに加算する．また，このときの結合演算は，データベースの結合演算ではなく，後述するMPIでの実装を我々で行うため，併

合結合で実装することができる．そのため，関係表を文書順でソートするコスト($|R1|\log|R1|, |R2|\log|R2|$)，結合演算を行うコスト($|R1|+|R2|$)を加算する．ここで$|R1|, |R2|$は，それぞれ結合する二つの関係表のタプル数を表す．

各処理オペレータに対するコスト定義をまとめたものを表1に示す．

3.4 XML 実装データの分割，配置処理の概要

提案手法における XML データの分割，配置処理の概要は以下の通りである．

1. まず，ワークロードの XPath 式から，全ての経路式を抽出する．なお，述語を含む問合せの場合は，二つ以上の経路式が抽出されることもある．
2. 1で得られた経路式を元にXMLデータをフラグメント化し，初期フラグメントを構成する．ここで基本的なルールとして，初期フラグメントに対応するノードを根ノードとする部分木を一つのフラグメントとして扱うこととする．このとき，いつくかの経路式がXML木上で重複部分を持つことがあるかもしれないが，その場合は，重複が起こらないような分割を求める．具体的には，文献[4]で提案したスキーマグラフの部分分割によるXMLデータの水平分割，垂直分割を用いる．
3. 2 で得られたフラグメントを計算ノードに配置する．この問題は本質的には組み合わせ最適化問題なので，グリーディアルゴリズムや遺伝的アルゴリズムなどの方法を用いて解を探索する．このとき，どのような配置方法が最適とするかには，いくつかの考え方がある．例えば，ワークロード全体の処理速度が最高となるものを最適とするという考え方，または，各計算ノードの負荷が均等となるものを最適とするといった考え方があり得る．本研究では，まず前者のアプローチをとることで，処理の効率化を狙う．

4. 実装

実装については，Javaからデータベースを操作するAPIであるJDBCを用いる方法，MPI (Message Passing Interface) [7]を利用する方法等が考えられる. MPIとは分散メモリ型の並列計算機上で並列演算を実現するために，メッセージ通信のプログラムを記述するための仕様のことである．両者を比較するため，ネットワーク転送速度を比較したところ，MPIの方が約40%性能が高いことがわかったので，本論文ではMPIによる実装を行った．MPIにはいくつかの実装があるが，MPICH2を用いた．ここで，MPIプログラムから関係データベースへ問合せを行うためのインターフェースが必要となるが，関係データベースにPostgreSQLを用いており，PostgreSQLサーバへ問合せを行うインターフェースであるlibpq[8]を用いた．

5. 実験

3章で述べた提案手法において，転送コストがなるべく低くなるようにフラグメントを配置した場合（ワークロード全体の処理効率の向上を狙った場合）に，比較手法に対してどれだけの性能向上が得られるかを検証するために実験による評価を行った．

表2 実験環境
Table 2 Experimental environment.

CPU	Intel® Xeon™ 3.0GHz x 4
OS	Red Hat Enterprise Linux 3.0
メモリ	1GB
RDBMS	PostgreSQL 8.1.0
MPI	MPICH2

表3 問合せワークロード
Table 3 Query workload.

	問合せ式	Wl1	Wl2	Wl3	Wl4	Wl5
Q1	//authors/email	0.1	0.16	0.04	0.5	0.02
Q2	/article/body/abstract	0.1	0.16	0.04	0.2	0.02
Q3	/article/body/section/@heading	0.1	0.16	0.04	0.1	0.02
Q4	//title	0.1	0.16	0.04	0.05	0.03
Q5	//author	0.1	0.16	0.04	0.03	0.03
Q6	//prolog[keywords/keyword= 'permanent']	0.1	0.04	0.16	0.03	0.03
Q7	//author[name= 'Joanna']	0.1	0.04	0.16	0.03	0.05
Q8	/article[@id = '2']/prolog/title	0.1	0.04	0.16	0.02	0.1
Q9	/article[@lang = 'en']/prolog/title	0.1	0.04	0.16	0.02	0.2
Q10	/article[prolog/authors/author/name= 'Akira']/body	0.1	0.04	0.16	0.02	0.5

5.1 実験環境

用いた実験環境は，表2の通りである．データセットはXBenchプロジェクトで公開されているデータ生成プログラムにより，サイズがそれぞれ10M，100M，1G であるニュース記事データを生成した．この際，XBenchからは複数のXML文書が生成され，ファイル数はそれぞれ，26, 266, 2666であった

5.2 実験方法

提案手法と比較するためのベースラインとして，XML文書をその数が均等になるように各計算ノードに割り当て，経路アプローチによって関係データベースに格納する方法を考える．これは，単純な経路アプローチにおいて，ノード表を水平分割して各計算ノードに割り当て，並列処理させる手法に相当する．

下の 3 つの場合について実験を行った．

1. ワークロードの発生頻度を5種類与え，それぞれについて提案手法と比較手法の処理時間を調べる．ワークロードの種類は表3の通りである．
2. ワークロードがwl1のとき，計算ノードの台数を1, 5, 10台と変化させた場合に提案手法の処理時間を調べる．
3. また，処理のボトルネックを検証するため，問合せ /article[prolog/authors/author/name='Akira']/body について処理時間の内訳を調べる．

1，2においては，表3のワークロードに従う100個の問合せを生成し，すべての問合せが実行し終える間での時間を計測した．ワークロード中の問合せに関しては，Q1～Q5は述語を含まない単純問合せで，結果サイズは比較的大きくなる．一方, Q6～Q10では述語を含み結果のサイズは比較的小さい．

5.3 実験結果

5.3.1 実験 1

実験 1 の結果を図4に示す．左の図が比較手法，右の図が提案手法である．それぞれ，縦軸は処理時間[second]を，横軸はデータサイズを表す．

まず，比較手法と提案手法の処理時間を比較すると，提案手法は，比較手法より約 4 倍～30 倍ほど処理が高速であることがわかる．これは，提案手法により，各問合せに必要なデータがフラグメントとしてまとめられているため，各計算ノードに対して，すべての問合せを実行している比較手法より問合せごとの並列度が上がったためであると考えられる．

また，発生頻度の違いに着目すると，比較手法は，比較的

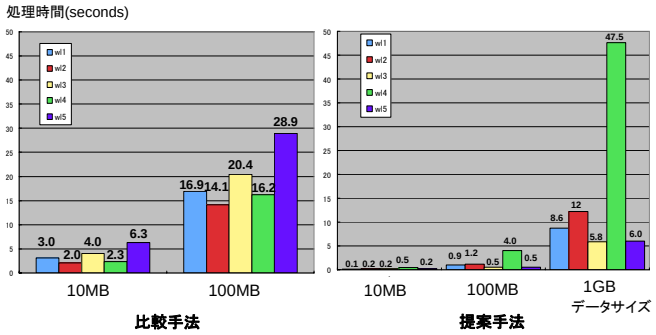

図4 提案手法と比較手法の比較

Fig.4 Comparison of proposed and baseline method.

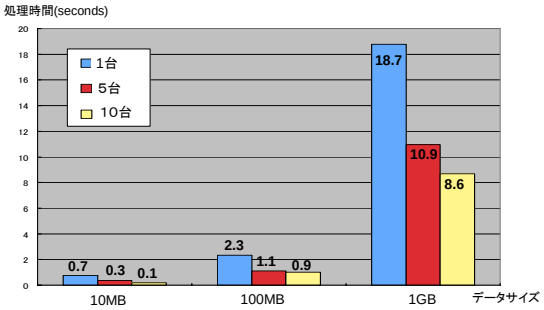

図5 台数効果

Fig.5 Cluster nodes number effect.

複雑な問合せが多く含まれる場合に，処理の効率が悪くなっている．提案手法については，処理結果のタプル数が多い問合せの発生頻度が高い場合，処理の効率が悪くなっていることがわかる．

5.3.2 実験 2

実験2の結果を図5に示す．縦軸は処理時間[second]を，横軸はデータサイズをそれぞれ表す．計算ノードの台数が多くなるにつれて，処理速度が上がっていることがわかる．しかし，1台と10台の場合を比較すると，速度は約2倍程度しか向上していないことがわかる．このため，さらに台数効果を追求する必要があると考えられる．

5.3.3 実験 3

実験3の結果を図6に示す．提案手法について，それぞれ選択処理，データ転送処理，結合処理が処理全体の中で占める割合を示す．この結果からは，選択処理が処理の比較的多くの部分を占めていることがわかる．このことから，タプル数が多いフラグメントをさらに水平分割によって計算ノードに配置し，並列的に処理することで選択処理の処理効率を高めることなどが考えられる．

6. まとめ

本研究では，PCクラスタを用いたXMLデータの並列処理方式について評価を行った．具体的には，提案方式におけるXPath問合せのコスト計算の詳細を定めるとともに，MPIとlibpqを用いて実装を行い，詳細な性能評価を行った．実験結果より，提案手法は比較手法より約4倍～30倍程度高速であることがわかった．また，本手法の台数効果を調べ，処理のボトルネックを調べるために問合せ処理時間の内訳について調べた．

今後は，さらに台数効果を高めるため，XMLデータの水平・垂直分割の効果的な組み合わせの検討，ワークロードの変化への対応などが挙げられる．

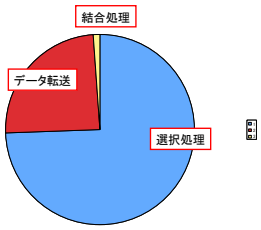

図 6 問合せ処理時間の内訳

Fig.6 Query processing time breakdown.

[謝辞]

本研究の一部は，科学研究費補助金特定領域研究(#18049005)，若手研究(B)(#17700110)の支援により行われた．

[文献]

[1] World Wide Web Consortium: Extensible Markup Language (XML) 1.0 (Third Edition), http://www.w3.org/TR/REC-xml. W3C Recommendation 04 February 2004.

[2] M. Yoshikawa, T. Amagasa, T. Shimura and S. Uemura: “XRel: Apath-based approach to storage and retrieval of XML documents using relational databases”, ACM Transactions on Internet Technology(TOIT), 1(1): 110–141 (2001).

[3] 天笠俊之，植村俊亮：“リージョンディレクトリを用いた関係データベースによる大規模XML データ処理”，日本データベース学会Letters, 3(2):33–36 (2004).

[4] 城戸健太郎，天笠俊之，北川博之：“PC クラスタを用いたXML データ並列処理方式の提案”, Proc. DEWS2006 (2006).

[5] R. Goldman and J. Widom: “DataGuides: Query Formulation and Optimization in Semistructured Databases”, Proc. ICDE (2002).

[6] Online Computer Libraly Center. Introduction to the Dewey Decimal Classification,http://www.oclc.org/dewey/versions/abridgededition14/ntro.pdf.

[7] The Message Passing Interface (MPI) standard. http://www.unix.mcs.anl.gov/mpi/.

[8] libpq. http://www.postgresql.jp/document/pg814doc/html/libpq.html.

城戸 健太郎 Kentarou KIDO

筑波大学大学院システム情報工学研究科在学中．XML データベースの研究に従事．日本データベース学会学生会員．

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．