社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－DBS－139 (14)
2006－ＦＩ－ 83 (14)
2006／5／31

インクリメンタルに更新可能なXPushマシンにおけるフィルタ交換のコスト削減

武川 肇[†1] 片山 薫[†2] 石川 博[†3]

フィルタリングは多くのXMLシステムで主要な機能である．以前に我々はフィルタの部分交換が可能なフィルタエンジン用オートマトンを提案した．しかしその交換方法では，フィルタの分離や結合を同時に数多く行うと，部分交換は再構築よりも時間コストが高くなる．この問題を解決するため，フィルタの部分交換時のオートマトン更新をXMLデータ評価時まで遅延させる方法を提案する．

Efficient Filter Exchange Method in an Incrementally Updatable XPush Machine

Hajime TAKEKAWA,[†1] Kaoru KATAYAMA[†2] and Hiroshi ISHIKAWA[†3]

Filtering is the principal function in many XML systems. Recently we proposed the automaton for a filter engine in which filters can be exchanged. However, time cost in this exchange method will become higher rather than that of reconstruction if many separations and additions of filters are performed simultaneously. The method of delayed automaton updating for the partial exchange filters until the moment of XML data evaluation is proposed to solve this problem.

1. はじめに

Publish/Subscribeシステム[6][8]やXML配信システム[1]などの様々なXMLシステムにおいて，フィルタリングは主要な機能である．典型的にこれらのシステムではXMLデータごとにフィルタのパターンが含まれるか否かの判定を行うが，このときフィルタの数が問題となる．

Guptaらは，フィルタ条件の数が増えてもXMLシステムのデータ評価のスループットが維持できるように，全フィルタを，1つの決定性プッシュダウンオートマトン(XPushマシン)として，ボトムアップに結合する構築法を提案した[5]．XMLシステムはXMLデータをストリームとして処理しながら，タグや値ごとの各イベントで，XPushマシンの参照を行う．XPushマシンは，参照時に渡されるタグや値から，まとめて全フィルタの判定を行う．XPushマシンは各イベント時の参照負荷が一定に保てるようにできている．ただし未知のXMLデータ(構造)に対しては各イベントで構築処理が発生する．

XPushマシン構築処理は，XPushマシンの内部状態数の爆発を防ぐために，データ評価時まで遅延させる必要があった．そこでGuptaらはXPushマシンの参照時に少しずつ完全な(complete)XPushマシンへと近づけ，そのときの結果を状態遷移表に保持する仕組みを用いた．XPushマシンは評価したことがあるデータと同じ構造のときには，状態遷移表を参照するだけであり，構築処理を行わない．そのためXPushマシンは同じ構造のデータを効率よく処理できる．

しかしGuptaらの構築法にはオートマトンの更新機能がないため，フィルタの部分交換の実現にはXPushマシン全体の再構築が必要となる．ところがXPushマシン構築処理はフィルタ数に依存するためコストが高く，したがって再構築はXMLシステムのスループットを低下させる．

そこで我々は問合せごとに構築したサブXPushマシンの集合から全体のオートマトンを構築する手法を提案した[2]．サブXPushマシンを全体のオートマトン(統合型XPushマシン)に結合でき，その結合処理では統合型XPushマシンに対してインクリメンタルな更新が行える．また分離処理も同様に行える．

統合型XPushマシンによって，フィルタの部分交換(分離と結合)が可能となる．しかしその更新方法では，フィルタの部分交換を同時に数多く行うと，部分交換は再構築よりも時間コストが高くなる．この問題を解決するため，フィルタの部分交換時のオートマトン更新をXMLデータ評価時まで遅延させる方法を提案する．

†1 職業能力開発総合大学校，静岡大学創造科学技術大学院
†2 首都大学東京システムデザイン学部
†3 静岡大学情報学部情報科学科

本稿の構成は，続く 2 章で提案方式について述べ，3 章でフィルタ部分交換の実験概要と実験結果を示す．

2. 提 案 方 式

この節では統合型XPushマシンの概要および本提案のフィルタ部分更新法について述べる．

2.1 統合型 XPush マシンの概要

統合型 XPush マシンの概要を図 1 に示す．統合型 XPush マシンにおけるフィルタ条件は XPath[3]で記述し，この条件を XPath フィルタという．XPath フィルタの定義はGupta らのXPushマシン[5]のものと同じである．例 1 に統合型 XPush マシンで利用する XPath フィルタの例を示す．

例 1 [Running Example]
P1 : //a[@b>=10 and @b<20]
P2 : //a[@b<20]

XPath フィルタはコンパイラによってサブ XPush マシンへと変換される．図 1 で M1～M5 はサブ XPush マシンを表す．サブ XPush マシンはフィルタグループ単位に作成される．現在フィルタグループには問合せ単位（例えば XSLT[4] などのテンプレートや multiple predicate[7]）を想定している．なおサブ XPush マシンは従来の XPush マシンとほぼ同じ構造であるため，サブ XPush マシン内の部分フィルタ交換はできない．しかしサブ XPush マシンは統合型 XPush マシンに結合でき，その結合処理では統合型XPushに対してインクリメンタルな更新が行える．また外部記憶を利用したサブ XPush マシン単位の分離と再結合処理が行える．

統合型 XPush マシンはフィルタグループごとに分割管理されたサブ XPush マシンとインクリメンタルに更新可能な統合結合状態表と 4 種類の統合結合状態遷移表で構成されている．統合結合状態表の X と統合結合状態遷移表の Tpop，Tadd，Tvalue，Taccept はサブ XPush マシンを集計するために用意された表である．

2.2 分離処理の遅延方法

本方式では従来方式の分離処理[2]を次のよう改良する．

「サブ XPush マシンの分離指示時が行われたときは，その id だけ記録して，システムが忙しくないときにまとめて分離処理を行う．」

2.3 結合処理の遅延方法

本方式では従来方式の結合処理[2]を次のよう改良する．

「統合結合状態表 Tadd に構築済みの値は更新しない．」

なぜ Tadd を更新しないのかについて以下に説明する．

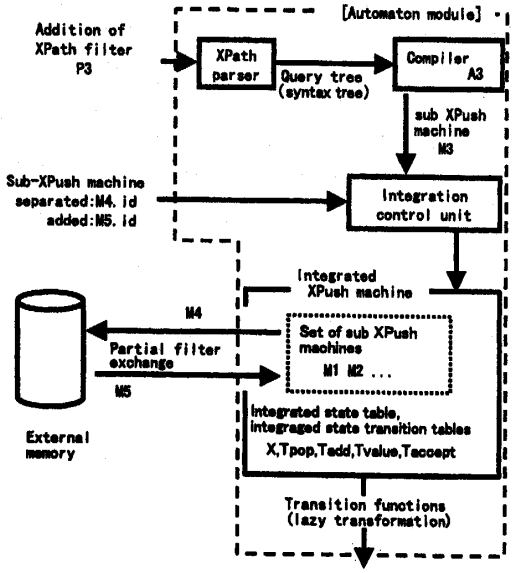

図 1 統合型 XPush マシンの概要

例 1 の P1 のサブ XPush マシン M1 を図 2a に示す．図 2a の A1 は P1 をコンパイルして得られる非決定性有限オートマトンである．これを葉から遷移させ，葉を持つ部分木を全て導き出したものがサブ XPush マシンとなる．Mi.Q は内部状態（XPush 状態という）のコレクションである．Mi.Tvalue は XML データの値を評価するときに参照される．Mi.Tpop は葉から根方向への遷移を評価するときに参照される．Mi.Tadd は XML データのある要素の全ての子（要素および属性）の総和を計算するときに参照される．図 2b には M1 だけが結合された統合状態表と統合結合状態遷移表が示されている．なお実際には M1.Tvalue と Tvalue のそれら以外の表に記載されている値のほとんどは，XML データ評価時まで構築が遅延される．

ここで図 2c に示された例 2 の P2 のサブ XPush マシン M2 を図 2b の表に結合する際の，インクリメンタルな更新について説明する．結合アルゴリズム[2]によって，前世代（ここでは図 2b の表）および M2 を入力として，図 2d の世代関係が導かれる．世代関係は前世代の内部状態と結合中のサブ XPush マシンの内部状態の組を表す．結合アルゴリズムはこの世代関係を利用して，インクリメンタルに図 2b の統合結合状態遷移表を更新し，図 2e へと変化させる．このとき Tpop は前世代の値が上書きされる可能性がある．たとえば Tpop[x3][a]は，x0（空欄の場合*を参照する決まりとなっている）から x10 へと変化している．一方，Tadd は結合処理によって前世代の値を上書きされることはない．このことを本方式に応用した．なおTaddが上書きされないことについての証明は今後の課題とする．

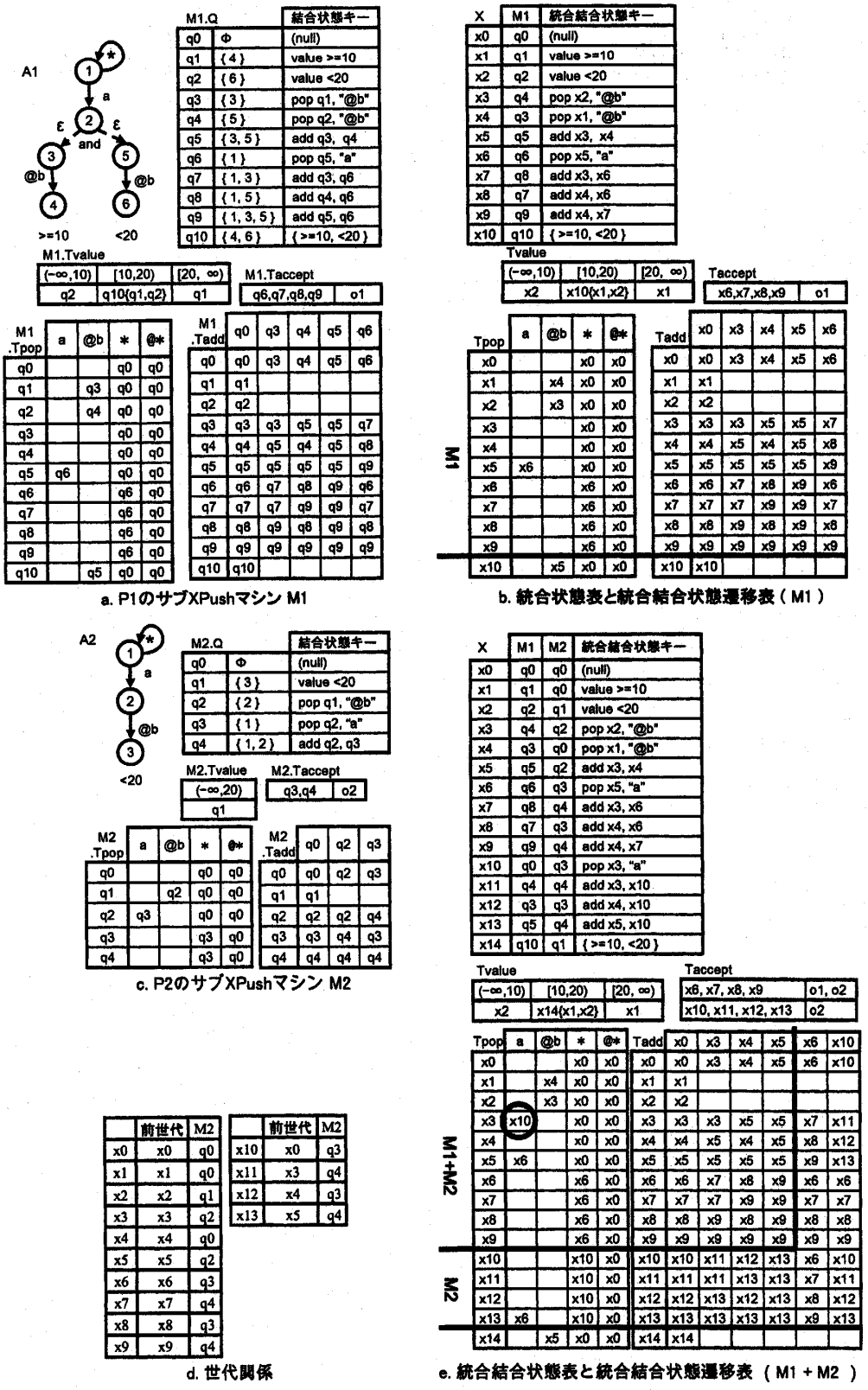

図2　サブXPushマシンを統合型XPushマシンに結合するときの統合結合状態表と統合結合状態遷移表の変化

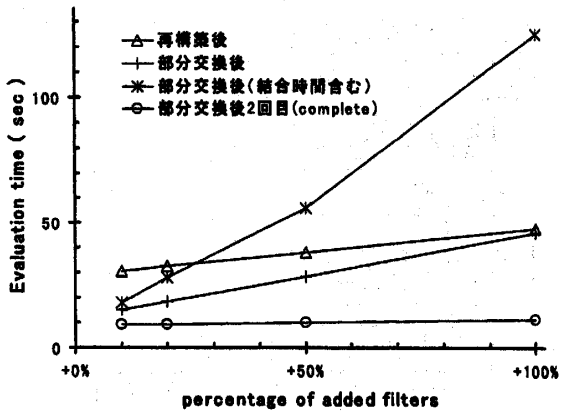

図3 従来方式によるデータ評価時間（フィルタ総数=1,000）

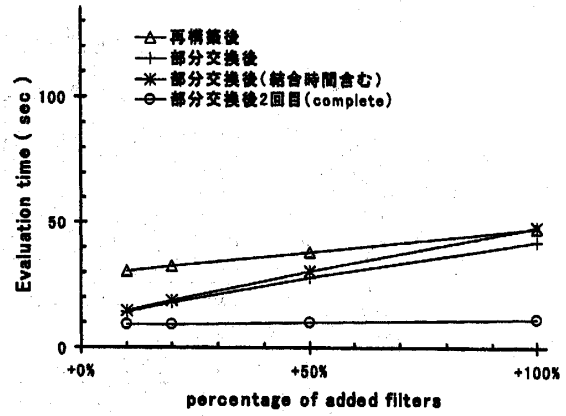

図4 提案方式によるデータ評価時間（フィルタ総数=1,000）

3. 実　　験

ここでは統合型XPushマシンのフィルタ部分交換に関する実験概要と実験結果を示す.

3.1 実験概要

実験は提案方式のフィルタエンジンをJavaで実装し，これをRed Hat Linux 8.0上で使用した．使用したコンピュータのCPUはPentiumIV3.3GHz，メモリ2Gである．XMLパーサにはxerces2.6.2（http://xml.apache.org）を利用した.

実験データにはProtein（http://pir.georgetown.edu）を利用した.Proteinは700MBあるデータセットであり，これを9.12MBのサイズに分割したものを用いた.XPathフィルタはXMLデータからランダムに作成するプログラムで生成し，各実験において必要数分だけ利用した.

予備実験1：まず1フィルタグループあたりのフィルタ数（以降，グループサイズ）が20で，総フィルタ数が1,000+10%（=20×50組+20×5組）のXPathフィルタをコンパイルした後，1つに結合して従来方式の統合型XPushマシンを再構築した．そしてデータ9.12MBを1回評価させ，その際のデータ評価時間を測定した．次に全フィルタの10%を分離し，その分離したフィルタを結合させ，同じデータを2回評価させた．その際に結合処理時間とデータ評価時間を測定した．同様に総フィルタサイズが1,000+20%, +50%, +100%に対しても行った.

実験1：提案方式で，予備実験1と同じ実験をした.

実験2：まずグループサイズが10で，総フィルタ数が1,000+10%（=10×100組+10×10組）のXPathフィルタをコンパイルした後，1つに結合して提案方式の統合型XPushマシンを再構築し，予備実験1と同じデータを1回評価させた．次に全フィルタの10%を分離し，その分離したフィルタを結合させ，そのときの結合処理時間を測定した.同様に総フィルタサイズが1,000+20%, +50%, +100%に対しても測定した.さらにグループサイズを20, 50に対しても行った.

実験3：まずグループサイズが10で，総フィルタ数が500+20%（=10×50組+10×10組）のXPathフィルタをコンパイルした後，1つに結合して提案方式の統合型XPushマシンを再構築し，予備実験1と同じデータを評価させた．次に全フィルタの20%を分離し，その分離したフィルタを結合させ，そのときの結合処理時間を測定した．同様に総フィルタサイズが1,000, 2,000, 5,000に対しても行った．さらにグループサイ 20, 50で同じ実験をした.

実験4：まず総フィルタ数が1,000のXPathフィルタをグループサイズが20（×50組）でコンパイルした後，1つに結合して提案方式の統合型XPushマシンを再構築し，予備実験1と同じデータを評価させた．次にフィルタ総数の10%（=20×5組）のフィルタを分離し，そのときの分離処理時間を測定した.同様に90%まで10%間隔で分離割合を増やし測定した．さらにグループサイズを20, 50に対しても同じ実験をした.

3.2 実験結果

予備実験1の測定結果を図3に示す．再構築後のデータ評価時間はフィルタ結合割合が増えると増加する傾向があり，+10%のときに30秒，+100%のときに47秒であった．一方，部分交換後のデータ評価時間は+10%のときに15秒，+100%のときに45秒であり，部分結合の方が，再構築よりも，時間コストが低い結果となった．しかし結合処理時間を含めると，+10%のときに18秒，+100%のときに125秒であり，図3のグラフでは+25%以上では再構築よりも,時間コストが高い結果となった.

実験1の測定結果を図4に示す．再構築後のデータ評価時間は従来方式と同じであった．一方，提案方式の部分交換後のデータ評価時間は+10%のときに14秒,+100%のときに42秒であり，従来方式と比べ，時間コストは低

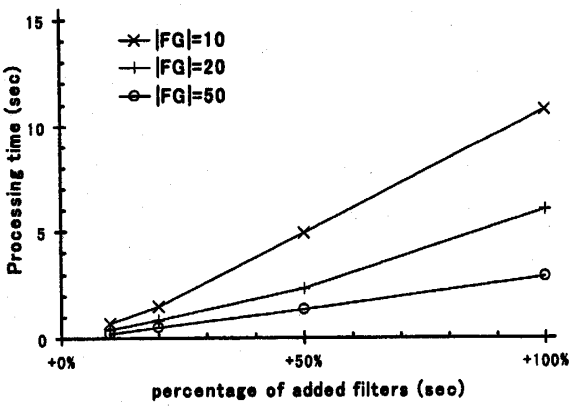

図5 結合処理時間（フィルタ総数=1,000）

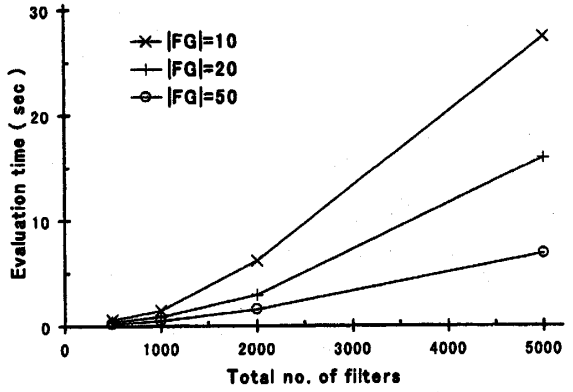

図6 結合処理時間（フィルタ総数の+20%）

い．さらに結合処理時間を含めても，+10%のときに 15 秒，+100%のときに 48 秒であり，図 3 のグラフでは+95%以下で再構築よりも，時間コストが低い結果となった．

実験 1 について考察する．結合処理を遅延させれば，データ評価時の XPush マシン構築処理コストは増すはずであるが，実験 1 の結果は違った．従来方式は，XML データには利用されていない遷移パスまでも構築が行われていたと考えられる．

実験 2 の結合処理時間の測定結果を図 5 に示す．グループサイズが 10 の結合処理時間は，+10%のとき 1 秒，+100%のとき 11 秒であった．グループサイズが大きいほど結合処理時間のコストは低く，結合割合が+100%で，グループサイズが 10，20，50 はそれぞれ 11 秒，6 秒，3 秒であった．

実験 3 の結合処理時間の測定結果を図 6 に示す．グループサイズが 10 の結合処理時間は，フィルタ総数 500 のとき 1 秒，5,000 のとき 28 秒であった．グループサイズが大きいほど結合処理時間のコストは低く，結合割合が+100%で，グループサイズが 10，20，50 はそれぞれ 28 秒，16 秒，7 秒であった．

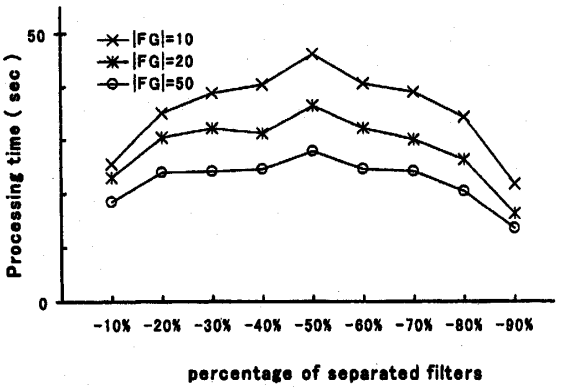

図7 分離処理時間

実験 4 の分離処理時間の測定結果を図 7 に示す．分離処理時間は，フィルタ総数の 50%まではフィルタ分離割合が増えると増加する傾向にあったが，50%以上では減少する傾向にあった．グループサイズ 10 で，分離割合が -10%，-50%，-90%はそれぞれ 3 秒，68 秒，30 秒であった．グループサイズが大きいほど結合処理時間のコストは低く，分離割合が-50%で，グループサイズが 10，20，50 はそれぞれ 58 秒，46 秒，36 秒であった．

実験 4 について考察する．総フィルタ数の 50%以上まとめて分離させると，分離処理の時間コストが減少する傾向にあった．このことは削除量が増えると状態遷移表の更新する範囲が狭くなるためと考えられる．

4. おわりに

本研究では，統合型 XPush マシンの結合処理と分離処理を遅延させる方式によって，統合型 XPush マシンの更新コストを，従来方式に比べて削減することができた．

参考文献

1) 内山寛之,鬼塚真,芳西崇：XML フィルタ配信システムにおける XPath の特徴量を用いた負荷分散方式, DEWS, 2004.
2) 武川藝,片山薫,石川博：インクリメンタルに更新可能な XPush マシン，情報処理学会:データベース, Vol.46, No.SIG18(TOD28), pp.116–128, 2005.
3) XML path language(XPath). http://www.w3.org/Tr/xpath.
4) Xsl transformations (xslt). http://www.w3.org/TR/xslt.
5) Gupta,A. Suciu,D. : Stream Processing of XPath Queries with Predicates. SIGMOD, pp. 419–430 , 2003.
6) Kwon,J. Rao,P. Moon,B. L,Sukho. : FiST: Scalable XML Document Filtering by Sequencing Twig Patterns, VLDB, pp. 217–228, 2005.
7) Peng,F. Chawathe,S.S.: XPath Queries on Streaming Data, SIGMOD, pp. 431–442 , 2003.
8) Tian,F. : Implementing A Scalable XML Publish/Subscribe System Using Relational Database Systems. SIGMOD, pp. 479–490, 2004.