3K-1

XML 処理の類型化に基づく高速化手法の検討

増塩 智宏† 渡邉 圭輔† 今村 誠†

三菱電機株式会社 情報技術総合研究所‡

1. はじめに

電子商取引や電子申請などにおけるデータ交換フォーマットとして XML はすでに事実上標準の地位を確立しており、近年では数分おきにデータ交換を行うリアルタイム処理へのニーズも高まっている。例えば、交換したデータをシステム内部の RDB にリアルタイムに反映させるといった処理では、従来は CSV やバイナリなどの軽量なフォーマットが利用されてきたが、近年ではシステムの互換性を高めるために XML が利用されるようになってきた。一方で、XML 処理の遅さがシステム全体のボトルネックとなる問題が指摘されている[1]。

このような課題への対処として、文献[2]に示される方法では、XML 文書中の処理対象外のデータ記述を XSLT 変換により簡略化することにより、XML 中の一部のデータにのみアクセスする用途における XML 処理時間を削減している。

しかし、この方式では XML 中の全データにアクセスするような用途ではデータを削除することができないため、高速化効果を得ることができないと考えられる。

そこで、我々は XML 中の全データにアクセスする場合でも高速化が可能な方法として、XML の木構造に対する処理を部分的な表構造に対する処理の組み合わせとして類型化し、各表構造のデータを配列に保持することで、複雑な XML ツリーデータアクセスを単純な配列アクセスに置き換えて高速化する方式を提案した。

本稿では、XML 処理を高速化する上での課題を整理し、類型化に基づく XML 処理の高速化手法について述べる。そして提案手法の有効性を検証するために行った実験結果を述べ、本手法の有効性を示す。

2. XML 処理における課題

多くの B2B システムでは社内に抱える既存処理を活用すべく、他システムから受信した XML を既存処理のデータ形式に変換する処理を行っている。このため、XML データ交換を用いた B2B システムにおける XML 処理高速化の主たる対象は、XML データ変換処理と捉えることができる。

変換処理の実装では、XML データアクセスに W3C 標準の DOM を用いる方法が一般的かつ容易な方法である。しかし、DOM では XML と同じ木構造でデータを保持しており、末端データへのアクセスでは多数の木構造探索処理が発生してしまうため、処理速度低下を招いてしまう。

DOM 以外の XML データアクセス方法としては、SAX が一般的である。SAX は XML の先頭から逐次処理を行う方式であり、XML パースと同時にデータ変換処理が行えるため DOM よりも高速な処理が可能である。しかし、SAX を用いた実装では、一般的なデータパース処理と同様に状態遷移を用いた複雑な設計が必要であり、開発コストの増加という別の課題が発生してしまう。

3. XML 処理の類型化に基づく高速化手法

我々は、2 章で述べた課題に対する解決方法として、XML 処理の類型化に基づく XML 変換処理の高速化手法を提案した。この手法では、入力 XML のスキーマと出力データ形式のテンプレートを用い、XML 文書の部分構造に適した高速処理プログラムを自動生成するものである。

はじめに XML のスキーマを解析し、シーケンシャルに要素が並ぶ部分を S テーブル(1 次元配列)、反復要素部分を R テーブル(n 次元配列)として類型化する。そして、各要素の内容をテーブルに対応する配列に格納する。また、XML 文書中の要素を特定するための XPath 式と、データ格納先テーブルの変数名との対応関係を記録しておく。例えば、図 1に示す XML データは、図 2のようなテーブルに格納される。

```
<予約>
  <ID>A9482</ID>
  <月>1</月>
  <日>14</日>
  <項目>
    <施設>会議室</施設>
    <利用者>三菱太郎</利用者>
  </項目>
  <項目>
    <施設>プロジェクタ</施設>
    <利用者>三菱太郎</利用者>
  </項目>
<予約>
```

図 1 XML 文書の例

sTable

A9482
1
14

rTable

会議室	三菱太郎
プロジェクタ	三菱太郎

図 2 テーブルへのデータの格納方法

また、XPath 式と変数名との対応関係は、図 3の

Accelerating XML Access by XML Structure Clustering

† MASUSHIO Tomohiro, WATANABE Keisuke, IMAMURA Makoto

‡ Information Technology R&D Center Mitsubishi Electric Corporation

ようになる。

/予約/ID	sTable[0]
/予約/月	sTable[1]
/予約/日	sTable [2]
/予約/項目	rTable
/予約/項目[n]/施設	rTable[n][0]
/予約/項目[n]/利用者	rTable[n][1]

図 3 XPath と変数名の対応表

次に変換先テンプレートを解析し、変換プログラムを生成する。テンプレートは図 4のように変換先テキスト中に XML データの XPath 式が"{$"と"}"に囲まれて記述されている。また、反復などの処理命令は{%REPEAT などで記述されている。自動生成される変換プログラムでは、テンプレート中のテキスト部分をそのまま出力し、XPath 部分を図 3 に示した対応表を参照して変数名に置き換える。図 5は、自動生成されたプログラムの例である。

```
CSV Data, {$/予約/ID}\n
{$/予約/月}, {$/予約/日}\n
{%REPEAT n /予約/項目}
{$/予約/項目[n]/施設},{$/予約/項目[n]/利用者}\n
{%%REPEAT n}
```

図 4 テンプレートの例

```
writer.print("CSV Data, ");
writer.print(sTable[0] + "\n");

writer.print(sTable[1] + ", ");
writer.print(sTable[2] + "\n");

for(int n=0; n<rTable.length; n++){
  writer.print(rTable[n][0]);
  writer.print(", ");
  writer.print(rTable[n][1]);
  writer.print("\n");
}
```

図 5 生成されたプログラムの例

4. 評価実験

提案方式の評価実験として、XML ファイルから CSV 形式への変換処理時間を測定した。処理対象 XML には S テーブルのみ持つものと、R テーブルのみ持つものとを用意し、各テーブル内の要素数が異なる 7 種類、計 14 個のファイルを用いた。また、提案方式の比較対象として、上記 XML と図 4 に示した形式のテンプレートを入力として与え、CSV ファイルを出力する機能を DOM、および SAX を用いて実装した。実験では、提案方式により自動生成した変換プログラムと DOM 版、SAX 版の 3 種類の変換プログラムについて、各 XML ファイルの変換処理時間を測定した。

5. 考察

評価実験の結果として、入力に S テーブルの XML を与えた場合を図 6に、R テーブルの場合を図 7に示す。実験結果より、提案方式の処理時間は DOM や SAX を用いた方式によりも 3～4 割ほど少ないことを確認できる。また、今回の実験では R テーブルのほうが提案方式の効果が高くなった。

R テーブルのような多次元構造データへのアクセスではツリー走査処理が必要であるが、提案方式では R テーブルへのデータアクセスは単純な 2 次元配列アクセスであり、ツリー走査が不要なため高速に処理できたためと考えられる。

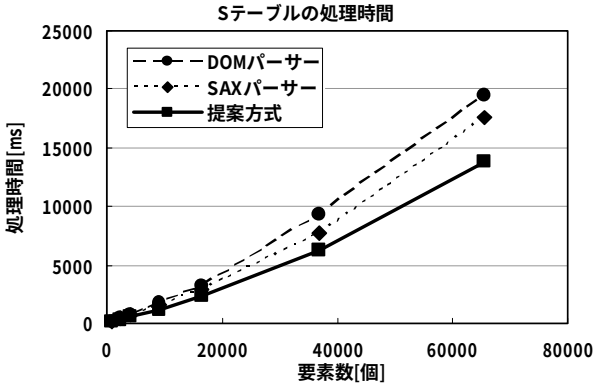

図 6 S テーブルの変換処理時間

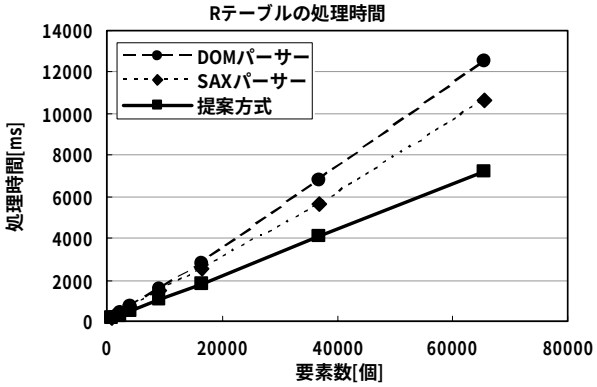

図 7 R テーブルの変換処理時間

6. おわりに

データ交換処理で受け取った XML をシステム内部で使用するデータ形式に変換する処理において、従来方式よりも高速に処理できる方式として XML 処理類型化に基づく方法を提案した。また、評価実験により、従来の DOM や SAX よりも 3～4 割処理時間を削減できることを確認した。

本方式は、入出力のデータ構造を与えて、変換処理を自動生成する方式であり、システム開発工数の削減にも寄与する。工数については、今後定量的な評価を行う予定である。

参考文献

[1] M. Nicola and J. John, “XML parsing: a threat to database performance,” Proc. of the 12th international conference on Information and knowledge management, pp. 175-178, 2003.

[2] 吉田茂, 矢作裕紀, 井谷宣子: XML 文書の事前形式変換によるデータ処理性能改善の検討, 第 1 回 情報科学技術フォーラム, D-29, pp.57-58, 2002