DOX/SAX による XML 処理と XML データベースの処理性能

テクノバン㈱　○豊谷　純
日大生産工　渡邊　昭廣
日大生産工　角田　和彦
日大生産工　亀井　光雄

１．はじめに

Web 関連の様々な技術が発展し、インターネット上ではオンラインチケット購買システムやショッピングモールなど、ありとあらゆるサービスが展開されている。これは Web アプリケーション開発技術の進展によるものであるが、このような機能に対応した開発言語は Java や C#.NET, VisualBasic.NET といったオブジェクト指向型言語である。

そしてそれらの言語では、データ形式は汎用性の高い XML(eXtensible Markup Language)が標準で利用できるようになっており、事実上、データ交換は XML 形式を利用するというのが業界の標準となって定着している。

この XML は W3C(World Wide Web Consortium)のワーキンググループが仕様を策定しているマークアップ言語である。そして XML はデータの構造を自由に定義することが出来るため、各種システム間においてデータフォーマットを統一すれば、企業内や企業間などの既存のシステムを、他のシステムと連携させることが出来る。

ここで XML ドキュメントの操作は DOM や SAX を利用したプログラミングが行われており、この場合は XML ファイルとして管理することが多い。また XML 専用の XML データベースもいくつかの製品があるが処理速度の遅さを理由に、実用例は稀である。

一般的には更新の必要の無いデータや、システム間のデータ交換や設定ファイルなどは XML 形式を適用し DOM や SAX で処理を行い、ログやデータ更新が頻繁に行われる部分に関しては、従来のリレーショナルデータベース(Relational Data Base)が適用されている。またこの XML 処理に関しては、DOM や SAX の他にも、テーブルの各項目を、XML ドキュメントの各要素やノードにマッピングして、RDB を利用して XML ドキュメントを処理するケースも非常に多く見受けられる。

本報告では、DOM や SAX を用いたプログラミングによる処理と、XML データベースの処理性能を比較し、それぞれの特性を明らかにする。

２．XML ドキュメントの処理技術

XML ドキュメントは XML パーサがデータを解析してアプリケーションに情報を提供するが、この際にアプリケーション側からは XML プロセッサというインターフェースを介して利用することが出来る。このインターフェースには API として、W3C が勧告している DOM (Document Object Model)や、既に XML を扱う際のデファクトスタンダードとなっている SAX(The Simple API for XML)がある。

Java 言語を利用する際は、J2SE(Java 2 Standard Edition) 1.4 以降では以下の２つの機能を持つ API が提供されており、XML ドキュメントに対して、DOM と SAX の両方が標準で利用できるようになっている。また必要な操作を全てプログラミングする必要があるため、後述の RDB に比較してデータ量が多くなる程、処理速度が低下し、コーディングが煩雑になりやすいという欠点がある。

Process Comparison of DOM and SAX Programming for XML Document and XML Database Procedure

Jun TOYOTANI, Akihiro WATANABE, Kazuhiko KAKUDA and Mitsuo KAMEI

３．テスト方法

処理時間の計測方法は、処理開始時刻と終了時刻を Java のプログラム内でシステム時間を参照して計測した。処理の実施方法は全てバッチ形式で、パラメータで指定した件数を連続して全件登録、全件更新、全件検索してそれぞれの処理時間をプログラム内で算出した。

ただし、SAX に関しては基本的には、データの参照機能しか無いため、簡単な検索機能を IF 文を用いてプログラミングした。

テスト回数については、表１のようにデータ件数を 1000, 5000, 7500, 10000, 12500, 15000 件として、全く同じ処理を行い、実施の順序は同じプログラムを連続して起動するのではなく、各回毎に XML-DB、DOM、SAX の順に異なるプログラムを各 10 回実施した。これは同じ処理を連続して行えば、コンピュータメモリ内のキャッシュ機能が働き、１回目よりも２回目の方がプログラムのロード時間やファイルの読込時間が、短縮されることを避けるためである。

初期化に関しては RDB の場合は SQL で Delete 文を実行して内容を削除し、XML ファイルに関しては必要最低限のルートタグのみを記述した XML ファイルを準備しておき、毎回コピーして初期化を行った。

調査機能

登録(追加)、更新(変更)、検索

データ件数

100,1000,2500,5000,10000,15000 件

表１　テスト用のデータベーステーブル

データ名	フィールド名	データ型
番号	ID	INT(10)
名前	NAME	CHAR(50)

調査対象と使用 API / Driver

- DOM(XML)：
  Java2 SDK1.4 標準のドライバを利用
- SAX(XML)：
  Java2 SDK1.4 標準のドライバを利用
- Xindice Ver1.1b(XML-DB)：
  Xindice 標準添付のドライバを利用

```
<?xml version="1.0" encoding="UTF-8"?>
<people>
   <user id="0">
      <name>Sample 氏名</name>
   </user>
   <user id="1">
      <name>Sample 氏名</name>
   </user>
</people>
```

図１テスト用の XML ドキュメント例

```
<?xml version="1.0" encoding="Shift_JIS"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2004/XMLSchema">
    <xsd:element name="people">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="user" type="userType"
                                                    minOccurs="0" maxOccurs="unbounded" />
                        <xsd:attribute name="id" type="xsd:positiveInteger" />
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>

    <xsd:complexType name="userType" >
        <xsd:sequence>
            <xsd:element name="name" type="xsd:string />
            <xsd:element name="phone" type="xsd:string />
        </xsd:sequence>
    </xsd:complexType>
```

図 2　テスト用の XML Schema

４．テストプログラム

プログラミング言語は汎用性の高さから Java を利用した。これはコマンド上で実行させる簡単なものであるが、Web アプリケーション構築の際にも、殆どそのまま Servlet や JSP(Java Server Pages)に流用できるため汎用性の高いものと言える。SAX と XML-DB のコードは紙面の都合上、掲載できなかったため割愛した。

```
import javax.xml.parsers.*; import org.w3c.dom.*; import javax.xml.transform.*;
import javax.xml.transform.stream.StreamResult;  import javax.xml.transform.dom.DOMSource;
import java.util.*;import java.util.Date; import java.io.*;

class RegistTime{                              // 時間を計算するためのクラス
      static long startTime, endTime;
      public static void setTime(){ startTime = Calendar.getInstance().getTime( ).getTime( );}
      public static long getTime(){ endTime = Calendar.getInstance().getTime( ).getTime( );
            return (endTime-startTime);}
}
class XML_DOM{
      public static void main(String argv[]){
            int queryCounter = 0;
            try {
                  int maxLoop = Integer.parseInt(argv[0]);
                  // XML 文書の読み込み
                  DocumentBuilder db = DocumentBuilderFactory.newInstance().newDocumentBuilder();
                  RegistTime.setTime(); Document rootTag = db.parse(new File("userdata.xml") );
                  NodeList userTag = rootTag.getDocumentElement().getElementsByTagName("user");
                  System.out.println("XML 読込　所要時間(ミリ秒)"+RegistTime.getTime() );

                  RegistTime.setTime();
                  for(int i=0; i<maxLoop ; i++){                        // データ追加処理
                        Element newUser = rootTag.createElement("user");
                        Attr id = rootTag.createAttribute("id");              // id 属性の生成
                              id.setNodeValue( Integer.toString(i) );
                        Element name = rootTag.createElement("name");  // <name>要素の生成
                              name.appendChild(rootTag.createTextNode("サンプル氏名"+i));
                        newUser.getAttributes().setNamedItem(id);       //生成した子要素を追加
                        newUser.appendChild(name);
                        rootTag.getFirstChild().appendChild(newUser);
                  } System.out.println(maxLoop + "件追加　所要時間(ミリ秒)"+RegistTime.getTime() );

                  RegistTime.setTime();
                  for (int i=0; i<maxLoop ; i++) {                      // データ更新処理
                        Element user_elem = (Element) userTag.item(i);
                        Element nChange = (Element) user_elem.getElementsByTagName("name").item(0);
                        nChange.replaceChild(rootTag.createTextNode("Updated"), nChange.getFirstChild());
                  } System.out.println(maxLoop + "件更新 DOM 所要時間(ミリ秒)"+RegistTime.getTime() );

                  RegistTime.setTime();
                  for (int i=0; i<maxLoop; i++) {                       // データ検索処理
                        Element newUser = (Element) userTag.item(i);     // data タグの id を取得
                        String id = newUser.getAttribute("id");
                        if (id.equals("50")) { queryCounter++; }
                  } System.out.println(queryCounter + "件検索 DOM 所要時間(ミリ秒)"+RegistTime.getTime() );
            } catch (Exception e) { e.printStackTrace(); }
      }
      public static String getChildNode(Element uerElement, String tagName) {
            NodeList nodeList = uerElement.getElementsByTagName(tagName);
            Element childElement = (Element) nodeList.item(0);
            return childElement.getFirstChild().getNodeValue();
      }
}
```

図３ DOM による登録・更新・検索用テストプログラム

５．テスト結果

データの登録では Xindice が DOM と比較にならない程遅いが、更新処理ではむしろ Xindice が DOM と比較にならない程高速に処理が行われた。

処理時間を見ると、DOM の場合は更新件数が増えれば増えるほど、更新処理の時間は増大しているが、Xindice の更新時間は、件数が増えても 3 m sec 程度で一定していた。

データ検索では図６を見ると SAX が最も時間が掛かっているが、これは DOM の場合では処理を始める前に、データをメモリ上にツリー構造で展開してから処理を開始するため、ファイルからのデータ読込みやツリー構造構築に関わる時間は計測されていない。

これに対し、SAX では逐一ファイルを読み込み、タグの開始や終了などのイベント発生時に記述された処理を行うため、SAX の実際のデータ処理以外に行われているファイル読込み処理などの時間が大きく影響した結果であると考えられる。この結果をみると SAX による処理が

遅く感じられるが、XML のデータサイズがメインメモリにロード出来ない程大きくなると、状況は反転するものと考えられる。

DOM では情報をメインメモリ内に構築するため、扱えるデータサイズはメインメモリに依存する。そのため、膨大なサイズの XML データを扱う場合は、むしろ DOM では処理出来なくなってしまう。

６．おわりに

本報告では DOM、SAX による XML ドキュメントの処理 XML データベース(Xindice)による処理時間の比較を行った。Xindice はデータの登録は時間が掛かるが、更新や検索は高速に処理されることが明らかになった。

従って汎用性に富んだ XML ドキュメントは異なるシステム間でのデータ交換に利点があるが、Xindice はログ出力のように連続してデータを登録・追加するような処理には向いていないと言える。ただし、一度登録すれば、検索や更新など、情報の管理は高速に行うことが出来るため、その特性を生かして活用すれば、システムの可用性を向上させることが出来る。

今回は、DOM や SAX、XML-DB を利用した時のプログラミング方法や処理時間を検討したが、どれかを択一で選択するのではなく、

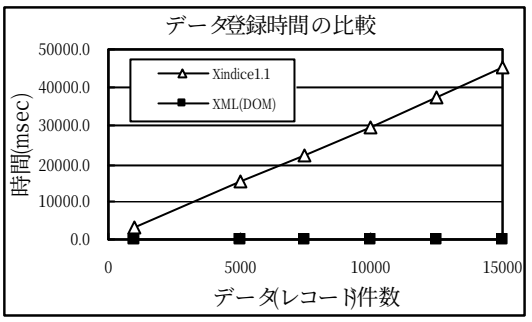

図４　データ登録時間

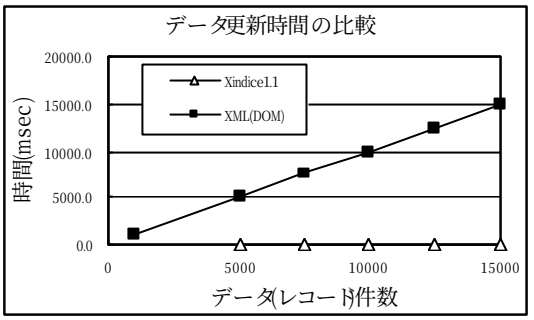

図５　データ更新時間

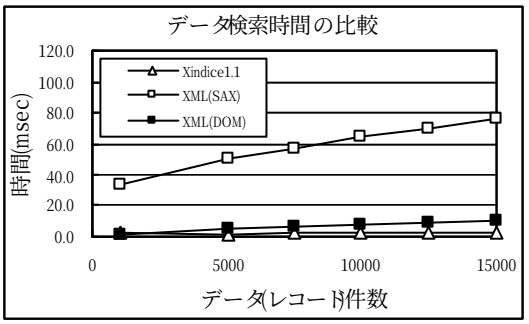

図６　データ検索時間

それぞれの特性を生かし、適所に複合してシステムを設計及び構築するのが望ましい。

今後はインターネットサーバーとして利用した場合の処理性能を検討するため、Servlet, JSP 等でシステムを構築した場合の処理時間の比較も実施したい。

参考文献

1) Dan Chang, Dan Harkey, Java & XML データアクセスガイド, 翔泳社, 2000 年
2) Apache Xindice の公式サイト http://xml.apache.org/xindice/
3) Apache Xindice の日本語紹介サイト http://www.theylive.jp/apache/xindice.html