携帯電話を想定したXML文書の高速DOM操作方式

小林 亜令　村松 茂樹　太田 慎司　西山 智
(株) KDDI研究所

携帯電話のようにCPUやメモリの処理能力が低い端末では，DOMパース処理，及びDOM APIによるXML文書操作処理における文字列解析処理時間が，端末操作性を損ねてしまうという問題がある．そこで本稿では，この問題を解決するために，携帯電話におけるXML文書の高速操作を目的とした，DOMパース処理方式，DOM APIの改良仕様，及びDOM APIによる文書操作方式を提案する．性能評価実験の結果，提案手法を用いればDOMパース処理，及び改良型DOM APIによるDOM操作処理を，現状の携帯電話上で十分高速動作できることが分かった．

A high speed DOM operation method of XML document for cellular phone

AREI KOBAYASHI , SHIGEKI MURAMATSU, SHINJI OTA, and SATOSHI NISHIYAMA
KDDI R&D Laboratories Inc.

A terminal with poor processing performance such as a cellular phone has a problem of processing time of XML document operation for string analysis. To solve this problem, we propose a high speed DOM parse method, an improved DOM API, and a document operation method. And we show the effectiveness of this method through performance evaluation.

1. はじめに

近年，PCを対象としたWWWベースの情報システム構築の際には，開発コストの低減，汎用性の確保，拡張性の確保の観点からXML形式のデータを採用するケースが一般的になってきている．反面，テキストデータのためデータ格納効率が悪く，低メモリかつ狭帯域な携帯電話システムでの活用では，データ圧縮を行うことが効果的である．

そこで筆者らは，携帯電話システム向けのXMLデータ符号化方式として，「XEUS(XML document Encoding with Uniformed Sheet，ゼウス)[1][2]」を提案した．XEUSは，「XEUSシート」と呼ばれるXML形式の符号化テーブルを定義して符号化を行うことを特徴としており，任意のXML文書を対象とし，そのスキーマ情報を利用した圧縮を行うだけでなく，符号化時にパース処理も行うことで，受信側の復号処理負荷を低減できる．

一般的にアプリケーションがXML文書を利用する際には，DOM APIを利用して一部分のノードを検索・抽出したり，編集したりという文書操作処理が発生する．携帯電話のようにCPUやメモリの処理能力の低い端末では，DOM パース処理、及びDOM APIによるXML文書操作処理における、文字列解析に処理時間がかかるという問題がある。

従来のXEUSでは，SAX APIによる出力を前提とした高速復号処理を実現したが，DOM APIによる文書操作処理の高速化については考慮していなかった．

そこで本稿では，この問題を解決するために、携帯電話におけるXML文書の高速操作を目的とした、DOMパース処理方式、DOM APIの改良仕様、及びDOM APIによる文書操作方式を提案する。また，処理時間の観点から性能評価実験を行い，提案手法の有効性を示す．

2. 従来の XEUS

2.1 符号化規則

XEUS は，2.2 節で述べる XEUS シートと呼ぶ XML 文書のノードのツリー構造と符号化テーブルをエンコーダとデコーダであらかじめ共有することを前提とし，エンコーダは符号化対象 XML 文書をパースして，ノードのツリー構造を符号化する．また要素名，属性名，要素値，属性値を XEUS シートに定義された符号化テーブルに則って符号化し，デコーダに送ることを基本とする．

送られるバイナリデータは，図 1 に示したとおりヘッダー部とボディ部に分けられ，ヘッダー部に符号化処理の際に参照した XEUS シート情報等を格納する．ボディ部には，XML 文書のノード部分と要素値/属性値部分を分離して符号化した結果を格納する．各ノード符号は，各要素に対する属性名群，子ノード有無の情報を組にして符号化する．また各要素値/属性値符号の占有する符号長（各数値型・選択型符号の占有ビット数符号，各文字列型符号のの占有ビット数符号）や各数値型・選択型の要素値/属性値符号に対してハフマン符号化を行い，ハフマンテーブル（各数値型・選択型符号の占有ビット数のハフマンテーブル，各文字列型符号の占有ビット数のハフマンテーブル，各数値型・選択型値のハフマンテーブル）も格納する．文字列型の要素値/属性値は，連結し一箇所にまとめた上で，gzip による符号圧縮を行い，格納する（文字列型の要素値/属性値符号）．

2.2 XEUS シート

XEUS シートは XML 形式で記述され，符号化対象文書のノードのツリー構造と符号化テーブルを定義する．すなわち，各要素について，要素名，属性名，属性値，要素値，子要素名のリスト，親子関係と符号化テーブル，また属性値，要素値について，データ型（数値型，文字列型，選択型），符号長，個数を定義する．また，要素値/属性値のセマンティクス（特に利用頻度が高いと考えられる座標値型とハフマンテーブル ID 付データ型）を用いた符号化規則も定義できる．

2.3 エンコーダとデコーダの処理フロー

2.1 節の符号化規則に従ったバイナリデータを生成するための符号化処理フロー及びバイナリデータの復号処理フローを図 2 と図 3 に示す．エンコーダでは，まず初期化処理を行い(1)，XEUS シートの読み込み，解析を行う(2)．その後，符号化対象文書のパース処理，符号化処理を行い，ボディ部の生成を行う(3-4)．その後，要素値/属性値のハフマン符号化を行う(5)．最後に文字列の gzip 処理を行い，出力する(6)．

デコーダは，まず初期化処理を行い(1)，XEUS シートの解析を行う(2)．次に XEUS バイナリデータを読み込み gunzip 処理を行う(3)．次にボディ部の復号処理を行い，SAX イベントを発行する(4-6)．最後に endDocument イベントを発行した時点で復

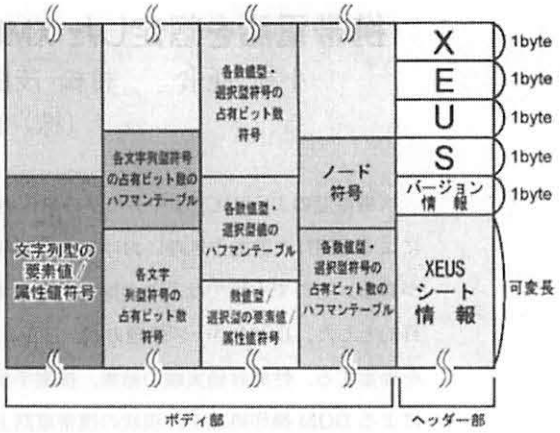

図 1：符号化後のバイナリデータ構造

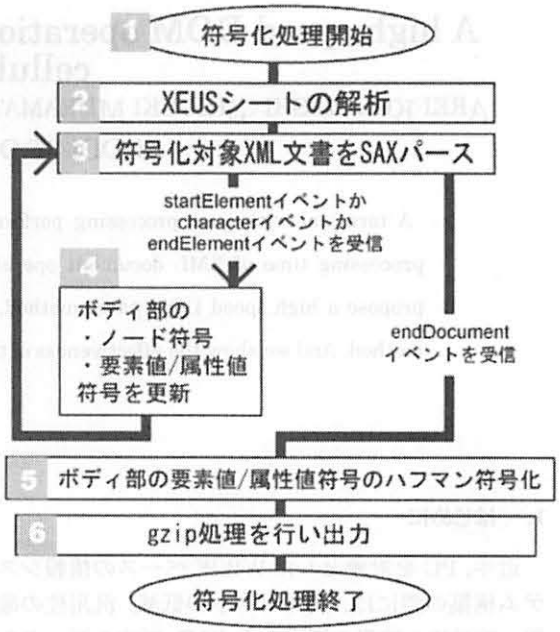

図 2：符号化処理フロー

号処理終了となる(7).

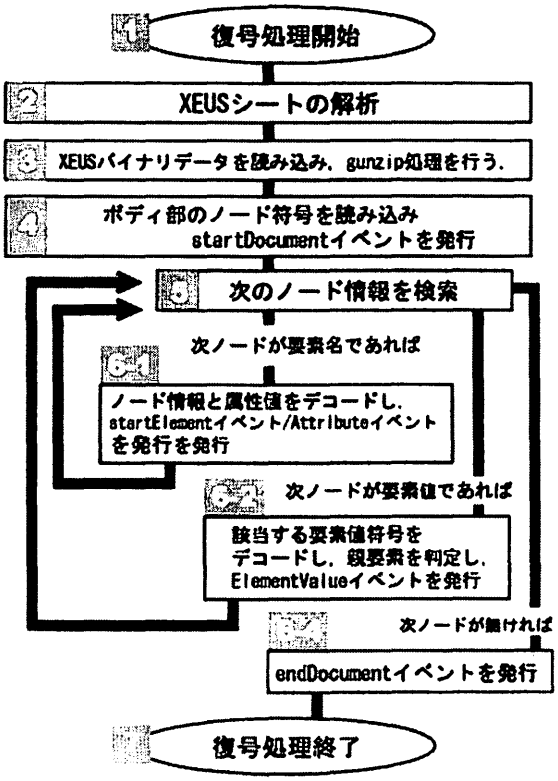

図3：復号処理フロー

3. 提案手法

3.1 従来の XEUS の問題点

DOM API による文書操作を行うためには，受信したデータを復号した後，DOM API による操作を可能とするデータ構造に展開（DOM パース処理）する必要がある．この際に，一般的な XML DOM パーサのような文字列操作やメモリの動的確保を携帯電話上で頻繁に行うと，処理時間がかかってしまう（課題1）．また DOM パース処理後，文書操作を行う際にも，一般的な DOM API のような文字列処理による文書操作を携帯電話上で行うと処理時間がかかってしまう（課題2）．

そこで上記の課題を解決し，携帯電話における XML 文書の高速操作を目的とした提案方式を次節に示す．

3.2 DOM パース処理方式

本節では，3.1 節で述べた課題1を解決するための DOM パース処理方式を提案する．提案方式は，図3の復号処理とは異なり，図1のバイナリデータからノード符号を XEUS シート情報を参照しながら復号した後，以下の6種類の変数を持つ DOM 操作可能なデータ構造に展開する．本データは要素，要素値，属性ノード毎に生成される．

(1)階層 ID

ルート要素を0階層目として，ルート要素から何階層目に存在するノードであるかを格納する．この情報によりノードの親子関係を把握する．

(2)1つ前後のノードのポインタ

1つ前後のノードに対するデータへのポインタ情報を格納する．この情報によりノードを辿る．

(3)ノード種別

ノード種別（要素ノード，属性ノード，要素値ノード）を格納する．この情報によりノードを辿る．

(4)要素名/属性名

ノード種別が要素，属性である場合，その要素名/属性名（XEUS シート情報に定義されるインデックス符号）を格納する．

(5)要素値/属性値のデータ型，個数（文字数）

ノード種別が要素値，属性である場合，その要素値/属性値のデータ型(char,int,double)，及び数値型値の個数/文字列型値の文字数を格納する．

(6)要素値/属性値符号のポインタ

ノード種別が要素値，属性である場合，その要素値/属性値符号のポインタ情報を格納する．

このノード符号展開データは，一般的な DOM パース処理後のデータ構造と異なり，要素値/属性値符号の実体を含まないため（ポインタのみ保持），符号長を固定長とすることができる．そのため，DOM パース処理を実行しながら，データ格納用メモリを動的確保するのではなく，デコーダ起動時にあらかじめメモリ確保することが可能となり，メモリ動的確保回数を低減することができる．また提案方式は，XEUS 符号化されたバイナリデータから直接，上記データ構造に変換（復号）するため，DOM パース処理中に文字列操作も発生しない．これらの特徴から DOM パース処理の高速化が期待できる．

3.3 改良型 DOM API による文書操作方式

本節では，3.1 節で述べた課題2を解決するための

改良型 DOM API, 及び DOM 操作方式を提案する. 提案方式では, DOM 操作時の文字列処理を低減するために, 一般的な DOM API とは異なり, 文字列型(String 型)以外のデータ型を持つ API を定義した. 以下に改良型 API の例と DOM 操作方法を示す.

(1)ノード遷移処理 API

(例 : XEUS_getFirstChild(),XEUS_getNextSibling(), XEUS_getParentNode())

引数, 返り値共にノード指定は, 全て 3.2 節で述べたノード符号展開データのポインタ情報で操作を行う.

(2)値の取得/更新 API

(例 : XEUS_getNodeCharValue(),XEUS_getNodeIntValue(), XEUS_getNodeDoubleValue())

ノード値の取得 API はデータ型毎に定義されており, 数値型値の取得 API が呼ばれた際には, 適宜 2.1 節図 1 に示したハフマン符号を復号し, int,double 型の値を返す. 文字列型値の取得 API は 2.1 節図 1 に示した文字列型符号を gunzip 処理し, 復号後の値を返す.

(3)値の更新 API

(例 : XEUS_setNodeCharValue(),XEUS_setNodeIntValue(), XEUS_setNodeDoubleValue())

ノード値の更新 API もデータ型毎に定義されており, API が呼ばれた際には, 引数に指定されている値が要素値/属性値符号内に存在していれば, そのポインタ情報で, 存在しなければ新たに要素値/属性値符号を追加定義して, そのポインタ情報で, 該当するノード符号展開データを更新する.

(4)ノードの挿入/削除 API

(例 : XEUS_insertBefore(),XEUS_removeChild(), XEUS_appendChild())

ノード挿入 API が呼ばれた際には, ノード符号展開データを新規作成し, 前のノードに該当するノード符号展開データの 1 つ後のノードのポインタ情報を新規ノード符号展開データのポインタで更新する. ノード削除 API が呼ばれた際には, 該当する符号展開データの削除は行わず, 1 つ前のノードに該当するノード符号展開データの 1 つ後のノードのポインタ情報を, 削除されるノードの 1 つ後のノードに該当するノード符号展開データのポインタで更新する. (削除対象ノードを skip する)

このような改良型 API を用いた DOM 操作を行うことにより, 一般的な DOM 操作とは異なり, 文字列操作とメモリ動的確保を低減することができ, DOM 操作処理の高速化が期待できる.

4. 提案手法の性能評価

3 章で述べた提案方式について DOM 操作ソフトウェアとベンチマークアプリケーションソフトウェアの実装を行い, 処理時間の観点から性能評価実験を行った.

4.1 評価コンテンツ

評価コンテンツには, 1 銘柄の 1 年分の株価情報を格納した RSS[3]コンテンツ 3 種類(XML 形式 : 129336byte,129339byte,129349byte,XEUS 形式 : 7098byte,7150byte,7237byte)を用いた. サンプルを以下に示す.

```
<rdf:RDF>
  <channel rdf:about="http://.../7203.html">
    <title>...</title>
    <description>...</description>
    <link>http://...</link>
    <items><rdf:Seq>
            <rdf:li rdf:resource="http://..."/>
            ...
  </rdf:Seq></items></channel>
  <item rdf:about="http://...">
    <title>2005-09-07</title>
    <link>http://...</link>
<description>4580,4600,4560,4590,5643300
</description>
  </item>
  ...
</rdf:RDF>
```

<description>要素の値に, 日毎の初値, 終値, 高値, 安値, 出来高を格納している.

4.2 性能評価システム

本実験は, 3 銘柄分の RSS を予め XEUS 符号化し, 端末内に保存した状態から以下のシナリオに基づいて各処理の処理時間を計測した.

①DOM 操作ソフトウェア及びベンチマークアプリ

ケーションソフトウェアの起動.
②1 銘柄の RSS ファイル（XEUS バイナリ形式）の読み込み，RSS データの DOM パース処理.
③改良型 DOM API により 1 年分の日付/始値/高値/安値/終値を取得.
④最近 1 か月分の株価の変動から，ローソク足と 13 日/26 日移動平均を生成し，グラフとして画面に表示.
⑤ユーザが Right キーを押したタイミングで，再度改良型 DOM API により 1 年分の日付/始値/高値/安値/終値を取得.
⑥最近 3 か月分の株価の変動から，ローソク足と 13 日/26 日移動平均を生成し，グラフとして画面に表示.
⑦ユーザが Down キーを押したタイミングで，別銘柄の RSS ファイルを読み込み，②-④の処理を行う.

ベンチマークアプリケーションの画面表示例を図 4 に示す.

図 4：画面表示例

4.3 DOM パース/DOM 操作処理時間

表 1 に，提案方式の各処理時間を示す．DOM 操作ソフトウェア及びベンチマークアプリケーションソフトウェアの動作環境は以下の通りである.
○仕様端末：携帯電話(au W21T)
○デコーダ動作環境：BREW 2.1 Ja
DOM パース処理時間は RSS ファイル(XEUS バイナリ形式)の読み込みを開始した時点からノード符号の展開処理を完了した時点までとして計測した．また，DOM 操作処理時間は，文書にアクセスする（XEUS_getDocumentElement()を呼ぶ）直前から，1 年分の日付/始値/高値/安値/終値を取得完了するまでの時間として計測した．表 1 から分かる通り，提案方式では，130KB 程度の RSS ファイルの読み込みを開始してから，グラフが表示されるまで，200msec 以内で処理を完了することができている．特に，DOM パース処理後の，ユーザイベントをトリガーとした DOM 操作と表示内容変更処理は，75msec 程度で完了できる．これは，現状の携帯電話でも十分な性能で XML 文書の DOM 操作ができていると言える.

表 1：平均処理時間

ソフトの処理内容	平均処理時間[msec]
① 初期化処理	40.0 msec
②⑦DOM パース処理	121.9 msec
③⑤⑦DOM 操作処理	56.4 msec
④⑥⑦グラフ生成・表示	18.7 msec

また DOM 操作処理の中で DOM API は合計 5551 回呼ばれており，1 回の DOM API 当たり 0.01msec 程度の時間で処理を完了していることが確認できた.

5. まとめ

本稿では，携帯電話における XML 文書の高速操作を目的とした，DOM パース処理方式，DOM API の改良仕様，及び DOM API による文書操作方式を提案し，処理時間の観点から行った性能評価実験結果を示した．その結果，RSS データの DOM パース処理に 120msec 程度，DOM 操作処理に 56msec 程度で処理を完了することを確認した．つまり提案方式を用いることにより，現状の携帯電話でも十分高速 DOM 操作できることが示された.

本研究は、独立行政法人情報通信研究機構からの委託研究「ユビキタス ITS」に基づき行われたものである。

参考文献

[1] 小林,松本,井ノ上” XML 文書汎用符号化方式「XEUS」”，信学技報 DE2001-9
[2] 小林他,” 汎用 XML 文書符号化方式「XEUS」の性能評価”,FIT2003.
[3] http://web.resource.org/rss/1.0/spec