L-011

組込み機器向け DOM モジュールの開発

羽藤淳平　佐々木幹郎　齋藤正史
三菱電機株式会社　情報技術総合研究所
{hatou, msasaki, marcy}@isl.melco.co.jp

1　はじめに

近年、インターネット上のコンテンツに限らず様々な分野で XML[2]の利用が活発となり、処理性能が飛躍的に伸びている組込み機器においても有効活用が進んでいる。XML を処理する際の内部データ表現である解析木の構成要素として DOM[1]を用いた実装が一般的であるが、動作させるために必要とするメモリ量がコンテンツサイズに強く関連して増加するという課題がある。組込み機器ソフトウェアを開発するにあたりメモリ効率向上は重要な命題の一つであり、DOM の省メモリ実装は XML の利用において重要になると考えられる。

現在、我々は組込み機器向けの XML 開発モジュール XML parser の開発を行っており、50KB 程度のコンテンツを解析する場合に、その 4 倍程度の 200KB で DOM 木を生成できる事を目標に開発を進めている。本論文では省メモリ化の方法に関して述べ、実装した DOM のメモリ消費量計測結果を示す。

2　全体構成

図 1 は我々が開発している XML parser の全体概要図である。単線領域は処理モジュールを、二重線領域はデータ領域を示す。

XML コンテンツを解析させたい Upper Level Application は DOM インスタンスを生成・格納するための Dynamic Allocating Heap 領域を生成し、DOM Factory に渡す。その次に Upper Level Application が解析したい XML コンテンツ(XML document)を XML Parser Core に渡す事で解析が開始される。

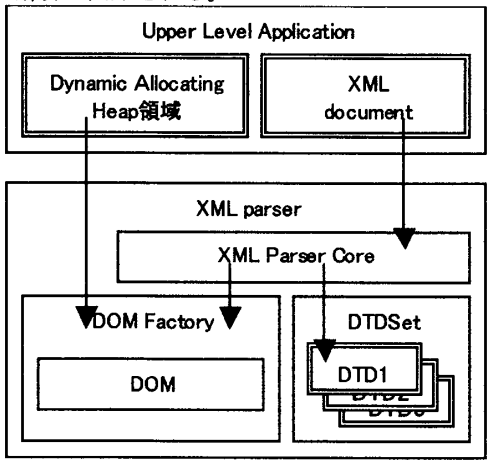

図 1XML parser の全体構成

XML Parser Core は
① Document に適合する DTD を DTDSet より取得
② DTD により妥当性検証
③ DOM Factory に DOM インスタンス生成リクエストを逐次発行
を行い、コンテンツ解析を行う。③の要求を受けた DOM インスタンス生成要求を受けた DOM Factory は、
① 整形式チェック
② 要求された DOM インスタンスを DOM 木に接続
を行い、DOM 木を構築していく。

全ての解析を終了した XML Parser Core は解析結果情報と共に、DOM 木が構築された Dynamic Allocating Heap 領域を Upper Level Application に返し、処理を終了する。

3　DOM の省メモリ化

3.1　クラス設計

本実装では DOM の省メモリ化を進めるため、各 DOM を表現するクラスは必要最低限のメンバで構成した。その際に開発効率も上げるため DOM 仕様の各 DOM Core のクラスの性質を調査し、表 1 の通りに共通した性質はそれぞれクラスとして定義し、その性質を持つ DOM Core クラスは自身特有の性質のみ保持し、共通の性質は性質クラスを継承する事で各 DOM のクラスを成立させている。

たとえば Element を表す DOMElement クラスでは、子要素・親要素共に保持でき、attributes、nodeName は可変、nodeValue が NULL であり、DOM 仕様で Node(DOMNode クラス)を継承しているため図 2 の通りの継承関係となる。

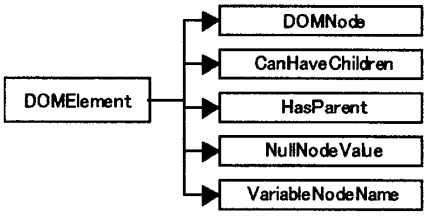

図 2 Element クラス継承関係

表 1 性質クラス一覧

性質		クラス名
子要素	保持できる	CanHaveChildren
	保持できない	NoChildren
親要素	保持できる	CanHaveParent
	保持できない	NoParent
attributes	常に NULL	NullAttributes
	可変	
nodeValue	常に NULL	NullNodeValue
	可変	VariableNodeValue
nodeName	常に固定	
	可変	VariableNodeName

Development of a DOM Module for Embedded Systems
Jumpei Hato, Mikio Sasaki, Masashi Saito
Mitsubishi Electric Corporation Information R&D center

3.2 インスタンス生成

DOMのインスタンス生成時に、必要最低限のインスタンスだけを生成する方法を取った。たとえば、ElementはNamedNodeMapをattributes属性として持つが、このNamedNodeMapは全てのElementで使用されるとは限らない。そのため、Element生成時にはNamedNodeMapは生成せずに、attribtues属性に初めて値をセットされる時にインスタンス生成を行う。この方法を取る事によって、不要なインスタンス生成を抑制する事ができる。

現時点では、インスタンスのサイズがポインタのサイズよりも大きいクラスの場合は、この方法でインスタンス生成を行う事としており、Nodeクラスの childNodes属性のNodeListと、attributes属性のNamedNodeMap等を、アクセスが行われるまでインスタンスを生成しない設計とした。

3.3 文字列のハッシュ表管理

DOMで扱う値の型は基本的に文字列である。また、XMLコンテンツでは同じタグ名や同じ属性名が複数存在する事が多い。そのため、文字列をそのままDOMに保持させるのは同じ内容の文字列が複数生成される事になり、非効率である。そこで、直接連鎖方式と解放番地方式の二通りを組み合わせたハッシュ表で文字列は管理し、DOMにはハッシュ値を保持させる事で、文字列の重複の回避と文字列比較の高速化を実現した。

逆に、テキストや属性値と言った重複頻度が低い文字列をハッシュ表で管理を行うと、ハッシュ表を圧迫する結果となるため、重複頻度が低いテキストや属性値はハッシュ表の管理外とした。

3.4 メモリ管理方式

3.2、3.3節でのメモリの確保処理で、メモリが必要になった時に必要な分だけ確保するとアドレスバウンダリで整列され、無駄な領域が頻出する可能性がある。そのため、メモリは解析開始時にまとまった量を一括確保し、その中から必要になった分を先頭から順番に使用するようDynamic Allocating Heap領域を実装する。また、解析途中で確保したメモリを使い切った場合は、その時点で再度メモリを追加で確保できる実装とする。

ただし、実装を簡単にするため、一括確保を行う動的配列テンプレートクラスを定義し、DOM生成に必要となるクラス毎に動的配列をDynamic Allocating Heap領域に定義し、インスタンスが必要となった場合には、該当クラスの動的配列から要素へのポインタを取得する方法を取る。この方法は、各動的配列が特定のクラス専用となるため、冗長なメモリ確保となる可能性が高いが、アドレスバウンダリを気にする事無く実装できる利点がある。

4 評価

4.1 実装環境

3章で述べた設計をもとにDOM実装方式の評価を行うためにMicrosoft Windows 2000上で実装を行った。プログラミング環境にはMicrosoft Visual C++6.0を使用した。ただし、ソース自体にはOSやライブラリ依存はないため、他OS上での動作も可能となっている。

4.2 使用メモリ量計測

評価プログラムを用いて、サイズの異なるXML Documentに対しDOMを生成させ、DOMに使用されるメモリ量を計測した。その際に、比較対象としてtinyXML[3]を使用した場合のメモリ使用量も計測した。

今回使用した試験コンテンツは、各DOM要素が表2の通り含まれている286byteの文字列を一単位とし、その文字列を何回連続して記述するかどうかでコンテンツサイズを調整した。生成したコンテンツのサイズは0.75KB、7.5KB、37.9KB、50.7KB、102.2KB、150KBとした。

図3がその計測結果である。X軸にコンテンツサイズ、Y軸に使用メモリ量をとり、菱形のグラフがTinyXML、四角のグラフが、本実装DOMの結果である。50KBのXML文書からDOM生成するにあたり、TinyXMLでは736KB消費するが、本実装では130KBで収まっており、目標値を達成している。また、150KBのXML文書の場合にはTinyXMLが2200KBに対して、本実装は324KBでDOM木を生成できる。

表2 試験コンテンツ内容

構成要素	個数,又は文字数
Element	5個
Attr	6個
Text	4個
Comment	2個
非DOM要素(文字列)	59文字

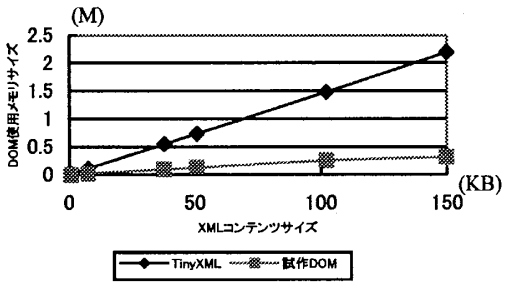

図3 メモリ使用量比較

5 おわりに

本論文で述べた方法でDOMを実装する事により、50KBコンテンツを130KB程度、150KBコンテンツを430KB程度でDOM木を作成可能である事が示された。

今後の課題として本設計の対象はDOM level1までとなっており、今後、DOM level2対応を進める必要がある。

また、3.4節で述べたメモリ管理方式でメモリ効率を上げる事ができる反面、一括管理するサイズの調整が重要なパラメータとなる。そこで今後は、不要なメモリ確保を減らす方法を現在検討中であり、その実装と効果の調査を行う予定である。

参考文献

[1] DOM level1 specification
http://www.w3.org/TR/1998/REC-DOM-Level-1-19981001/
[2] XML specification
http://www.w3.org/TR/xml11/
[3] tinyXML
http://www.grinninglizard.com/tinyxml/