4N-2

ASN.1コンパイラが生成する専用コーダのアプリケーションプロトコル実装への適用性の評価

長谷川 亨　　野村 真吾
国際電信電話株式会社 上福岡研究所

1. はじめに

OSI等のアプリケーションプロトコルでは、プロトコル要素(PDU)のデータ構造がASN.1[1,2]により定義されている。定義されるデータ構造や、転送されるバイト列の符号化/復号が複雑なため、アプリケーションプロトコルの実装には、高速なコーダ(符号化/復号)・プログラムの実現が重要である。実現法には、ASN.1の型毎にプログラムを生成する方法(専用コーダ)[3]と汎用的なプログラムにより実現する方法(汎用コーダ)[4,6,7]が考えられる。筆者らは、高速な符号化/復号が重要と考えて、専用コーダ・プログラムを生成するASN.1コンパイラを開発してきた[3,4]。本稿では、現実のアプリケーションプロトコルで実際に転送されるデータを想定して、コーダの符号化/復号速度を評価することにより、ASN.1コンパイラが生成するコーダの実際のアプリケーションプロトコルの実装への適用性を検討した。

2. コーダ・プログラムへの要求条件

ASN.1では、構造を表現する構造型と、整数やストリング等の値を表現する単純型を組み合せることにより、任意の複雑さの構造、及び値の大きさを持つデータを定義できる。しかし、現実のプロトコルにおいては、複雑さや大きさが無限のデータを扱うことは無く、アプリケーションの性質の違いにより、(1)構造は複雑だが小さなデータ、(2)構造が単純で小さなデータ、(3)構造は単純だが大きなデータ、を転送することが一般的と考えられる。要求される転送速度もそれぞれ異なるため、コーダプログラムがこれらのデータを要求される速度で符号化/復号できることが、アプリケーションプロトコルの実装に適用するための必須条件である。

(1) 複雑な構造を持つデータ

構造型の値を多数持ち複雑な構造だが、単純型の値としては小さなデータを持つ。ディレクトリやOSI管理プロトコル等の問合せを行うRPC系プロトコルで、問合せ結果を転送するデータが例として挙げられる。例えば、ディレクトリプロトコルで、10個程度の構造型の値を持つ検索結果の値を、数十個持つデータが考えられる(構造型が数百個までで、バイト列の大きさが数十Kバイト程度まで)。

(2) 構造が単純で小さなデータ

構造が単純で単純型の値としても数Kバイト程度までの値しか持たないデータ。RPC系のプロトコルで問合せ要求を転送するデータや、FTAMプロトコルでファイルのレコードを転送するデータ等が考えられる。特に、MAP/TOPのようなリアルタイム制御を行うプロトコルでは、数十ミリ秒の応答時間が要求される。

(3) 構造は単純だが大きなデータ

構造は単純だが単純型の値として100Kバイトから数メガバイトの程度の値を持つデータ。ファクシミリ・データ等を扱うテレマティック系のプロトコルが考えられる。

3. コンパイラの評価

3.1 符号化/復号方式

コンパイラが生成するC言語のコーダは、処理対象の値及びバイト列として仮想メモリ上に存在するデータを扱う。ただし、単純型の値の内、数十Kバイト以上の長さを持つストリング値を符号化/復号するために、値をメモリ上でコピーすることは無駄である。そこで以下の機構を提供することにより、コーダをシステムに組み込んだ時に、バイト列のコピーが一度だけになるようにしている。

符号化は、㋑値の中の各要素が符号化された時のバイト長の決定と、㋺各要素値を変換して結果のバイト列にコピーする処理、に分けて行う。復号は、①ストリング値をメモリ上にコピーする方式だけでなく、②バイト列上での位置を復号結果とすることによりコピーしない方式をサポートする。

3.2 処理速度

2章で述べた条件のデータの符号化/復号速度を評価するため、具体的には以下のデータに関して符号化/復号時間を測定した。実験はVAX 8700(6MIPS, VMS)とSUN-3(4MIPS,UNIX)上で行ったが、処理時間はVAX 8700での測定値を示している。

(1) 複雑な構造を持つデータ

以下のデータ[5]を対象として、符号化と復号時間を測定し、表1に示す結果を得た。筆者らの作成した汎用コーダ[4]及び文献[6]に比較して、数倍以上の処理速度であった。

(a) $2^8 - 1$個のSEQUENCEからなるバイナリツリー形式のデータ。一番ネストの深いSEQUENCEでは2個の整数を要素として持つ(合計256個)。

Applicability of Coder Programs Generated by the ASN.1 Compiler for Application Protocol Implementations

Toru HASEGAWA, Shingo NOMURA
KDD Kamifukuoka R & D Laboratories

(b) 256個のSEQUENCEの要素を持つSEQUENCE OFにより定義される1段のネストのデータ。SEQUENCEでは2個の整数を持つ(合計512個)。

	符号化	復号	バイト長
(a)	約17.2ミリ秒	約18.3ミリ秒	1,300バイト
(b)	約28.7ミリ秒	約30.0ミリ秒	2,052バイト

表1 符号化/復号時間(1)

(2) 構造が単純なデータ

MHS P2プロトコルで、ヘッダとコンテンツからなるメールを送信するIM-UAPDUを取上げた。ヘッダは構造型の値を32個、整数値を4個、10バイト以下のストリングを22個持つ、約230バイト長のデータである。コンテンツは、(a) 2Kバイトと(b) 100Kバイト長のストリングの2種類で行った。表2に符号化/復号の各処理時間を示す。この結果、コーダをシステムに組み込んだ場合、100Kバイトのストリングを持つIM-UAPDUの符号化には、㋑長さの決定に約0.18(0.16 +0.017)ミリ秒、㋺バイト列の変換に約5.95 (0.35+5.60)ミリ秒の合計約6.1ミリ秒かかる。復号には、システムの入出力において必要なバイト列のコピーを含めて約7.2 (0.85 +6.31)ミリ秒かかることが推定される。

	符号化(ミリ秒)		復号(ミリ秒) コピー	
	㋑長さ決定	㋺変換(バイト列)	①有	②無
ヘッダ	0.16	0.35	0.85	0.84
コンテンツ(a)	0.017	0.005	0.018	0.017
コンテンツ(b)	0.017	5.60	6.31	0.019

表2 符号化/復号時間(2)

3.3 プログラム規模

ASN.1コンパイラが生成するプログラムの規模を評価するため、MHS P2プロトコル及びFTAMプロトコルの抽象構文を、ASN.1コンパイラで変換した。ソースプログラムの規模と実行形式の規模を測定した結果を表3に示す。実行形式はVMSのCコンパイラにより得たものである。

プロトコル	ソース (ヘッダ)	実行形式
MHS P2	約 3.5K行 (約 0.5K行)	約69Kバイト
FTAM	約 5.8K行 (約 1.0K行)	約110Kバイト

表3 専用コーダの規模

4. 考察

ASN.1コンパイラが生成する専用コーダプログラムのアプリケーションプロトコル実装への適用性に関して、以下のことが考察される。

(1) 適用プロトコル

数MIPS程度の計算機を対象とした場合、本コンパイラが生成するコーダプログラムは、2章で述べた各種のプロトコルに対して、数十から数百Kバイト/秒程度の通信プログラム開発に適用可能であると考えられる。これは、Ethernet等のLAN上で通信を行うのに十分である。例えば、2章(1)のデータは3.2節(1)のデータに類似していると考えられるため、数十Kバイト/秒程度の通信が可能である。2章(2), (3)のデータに関しては大きさが100Kバイト程度までならば、数百Kバイト/秒程度の通信が可能になる。また、MAP/TOPのように応答速度に対して厳しい要求が課せられる通信においても、3.2節(2)の(a)のように構造が単純で大きさが数Kバイト以下のデータを扱う場合には、十分適用可能である。

(2) 適用マシン

提供されるメモリ量が限定されたパソコン等の計算機では、抽象構文の大きさに比例して規模が大きくなる専用コーダの使用は困難であると考えられている。標準化プロトコルの内で抽象構文の規模が大きいFTAMプロトコルに対して生成するプログラムは、110Kバイト程度の実行形式である。パソコン用Cコンパイラでは、さらにコンパクトになることが推定されるため、本コンパイラが生成する専用コーダをパソコンで使用することも十分可能であると考えられる。

(3) 大規模データの符号化/復号

メガバイト・オーダのストリングを持つデータ(2章の(3))に関しては、汎用コーダや専用コーダといったコーダの実現方式よりも、ストリングの処理がシステムのスループットを高くするためのキーとなる。このようなデータは外部ファイルに蓄えるのが一般的なため、大規模なデータを扱うシステムでは、外部ファイルに存在するデータの符号化/復号の仕方が重要になる。VAX VMSやMachのようにファイルを仮想メモリ空間にマッピングするマップトファイルを提供するOSでは、この機能を利用してファイルにあるデータを符号化/復号すれば良いと考えられる。それ以外のOSに対しては、3.1節で述べた無駄なコピーを避ける方式を外部ファイルにあるバイト列に対してもサポートすることにより、解決できると考えている。

5. おわりに

本稿ではASN.1コンパイラの生成するコーダの処理速度、規模を定量的に評価し、アプリケーションプロトコル実装に対する有効性を確認した。今後、改良を行うとともに、ASN.1コンパイラを用いたプログラム開発効率等に関して評価を進めていく予定である。最後に日頃御指導頂くKDD上福岡研究所通信ソフトウェア研究室 小西室長、コンピュータ通信研究室加藤主査に感謝する。

参考文献

[1]:ISO,"ASN.1", ISO / IS 8824 / 8825.
[2]:CCITT, Rec. X.208 / 209 , Nov. 1987.
[3]:長谷川, 野村, 堀内 "ASN.1支援ツールの開発 - コンパイラおよびエディタ-", 情報処理学会マルチメディアと分散処理研究会, 39-4, Sept. 1988.
[4]:野村,長谷川,堀内, "ASN.1テストツールの実用性向上に関する機能拡張", 本大会予稿, Mar. 1990.
[5]:長谷川, 野村 "ASN.1コンパイラの評価", 情報処理学会全国大会, 7M-7, Mar. 1989.
[6]:Nakakawaji,et al,"Development and Evaluation of APRICOT", The second ISIIS, pp55-62, Nov. 1988.
[7]:Ohara,et al, "ASN.1 tools for Semi-automatic Implementations of OSI Application Layer Protocols, pp63-70,The second ISIIS, Nov. 1988.