M_025

組込みSWの特性を用いたバージョン間差分抽出方式
A New Delta Extraction Method for Embedded Software

三井 聡† 清原 良三†
Satoshi Mii Ryozo Kiyohara

1. はじめに

近年、携帯電話をはじめとする組込み機器の機能が増大の一途をたどっている。これにつれて、機能のカスタマイズや更新、不具合の修正などに対する要求が高まっている。例えば携帯電話ではネットワークサービスと差分データを用いた不具合修正サービスが提供されている[1]。

組込み機器の多くはソフトウェアを静的リンクによって直接実行できる形にしてNOR型Flash ROMに格納している。そのため、少量の修正であっても参照先の位置ずれによって更新が必要となる箇所が多数発生し、差分データのサイズが増大するという問題がある。筆者らはこの問題に対して組込みソフトウェアの特性を用いた差分抽出方式を提案しているが、いくつかの課題が残っている[2]。本論文では従来の提案手法に対する改良案を示し、その効果を評価する。

2. 関連研究

バイナリデータを対象とした差分抽出方式に関しては多くの研究がなされており、例えばzdelta[3]はデータ圧縮の考え方を用いて差分抽出を効率的に実現している。

対象をソフトウェアに限定した方式としては、参照先の位置ずれによる影響を自動的に復元することで差分データサイズを小さくする方式[4]や、ソースコードの再コンパイル時に発生するレジスタアサインのずれに着目した方式[5]などがある。

3. 差分データの表現形式

旧版ソフトウェアから新版ソフトウェアを生成するための差分データは、一般には以下2種類のコマンドで表現される。

- 旧版からのコピー（旧版位置、長さ）
- 追加データ（データ長、データ列）

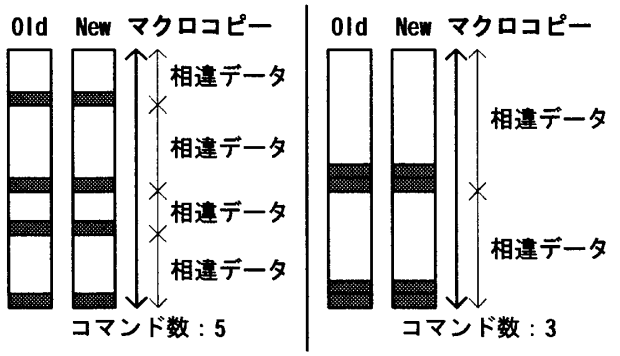

図1：相違箇所の分散と差分コマンド数

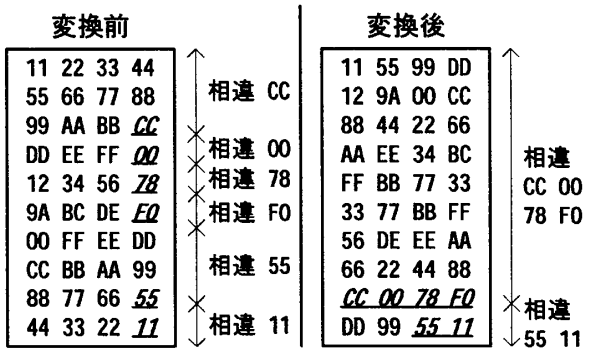

図2：アドレス変換方式の例

組込みソフトウェアでは、参照先の位置ずれによって領域全体としてはほぼ一致しているが参照元が相違箇所になっていると言うケースが多く見られる。このような領域に対しては、領域全体をコピーしておいて（マクロコピー）相違箇所のみを変更する、と言う差分データ表現方法を用いることができる。そこで、さらに以下2種類のコマンドを用いて差分データを表現することとする。

- 旧版からのマクロコピー（旧版位置、長さ）
- 相違データ（一致長、相違長、相違データ列）

マクロコピーを用いることで旧版位置の指定回数を少なくすることができ、差分データサイズを削減することができる。これらの表現形式を用いる場合、旧版と新版の間で相違箇所（byte）数が同じ場合でも、相違箇所の分散度合いによって必要となるコマンドの数、つまりは差分データのサイズが異なる（図1）。

4. アドレス変換方式

筆者らはRISCアーキテクチャCPU向けソフトウェアをターゲットとして、アドレス変換方式と呼ぶ差分抽出方式を提案している。アドレス変換方式は、ソフトウェアをワード（4byte）単位で見た場合に特定のバイト位置（例えば4byte目）に旧版ソフトウェアと新版ソフトウェアとの間で相違箇所が集中することに着目した方式である。旧版、新版の両者をバイト位置に応じて変換してか

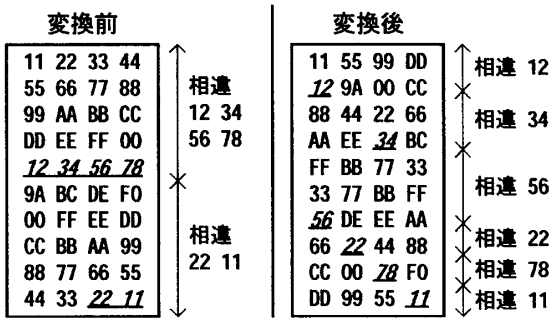

図3：アドレス変換方式が適していない例

† 三菱電機（株）情報技術総合研究所
Mitsubishi Electric Corp. Information Technology R&D Center

ら比較することで相違箇所が集中して現れるようになり、差分データの表現を効率的に行うことができるようになる（図2）。

5. 改良方式

従来方式ではソフトウェア全体に対してアドレス変換をかけた後に旧版ソフトウェアと新版ソフトウェアの比較を行って差分データを生成している。アドレス変換の目的は相違箇所が集中的に現れるようにして差分データサイズを小さくすることであるが、アドレス変換前の状態で相違箇所が集中している箇所に対してアドレス変換を行うと却って相違箇所が分散してしまい、差分データサイズが大きくなってしまう（図3）。実際にある組込みソフトウェアで評価した結果では従来方式では効果のある場合とない場合があり、その理由は上記のような相違箇所の分散が原因であることが分かっている。

そこで、ソフトウェア全体に対してアドレス変換をかけるのではなく、効果のある部分のみに対してアドレス変換をかける改良方式を提案する。

改良方式ではアドレス変換の開始と終了を意味するコマンドを用い、以下の手順で差分抽出を実行する。

① アドレス変換をかけずに旧版ソフトウェアと新版ソフトウェアを比較し、マクロコピーで表現できる領域（マクロ一致領域）を発見する。
② 各マクロ一致領域をワード単位で順次比較し、アドレス変換によって相違データコマンドの数を減らすことのできる範囲を特定する。
③ 差分データのエンコードを行う。この際、アドレス変換範囲に対してはアドレス変換を行ってから差分コマンドを表現するとともに、その前後に変換開始、終了のコマンドを出力する。

手順②においては、アドレス変換を行うことによる変換開始、終了コマンドによるコマンド数（差分データサイズ）の増加分を考慮する必要がある。

表1：差分コマンド数

	A	B	C
1→2	116,114	102,226	83,841
2→3	265,715	270,970	210,609
1→3	171,248	195,628	147,419

表2：差分データサイズ（単位:byte）

	A	B	C
1→2	403,922	367,338	334,820
2→3	936,459	927,899	818,750
1→3	610,033	646,493	559,162

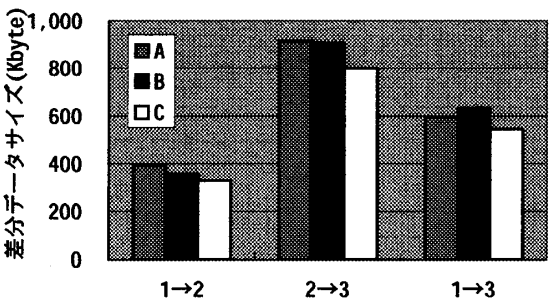

図4：差分データサイズ

6. 評価

ある組込みソフトウェアを対象に行った評価結果を示す。ソフトウェアの対象CPUはM32Rであり、ソフトウェアのサイズは約30Mbyteである。ソフトウェアは3バージョン（1、2、3）用意し、それぞれのバージョン間に対する差分抽出を行う。

評価は“コマンド数”と“差分データサイズ”の2つの観点から行う。差分データは独自に定めたフォーマットに従うものとする（フォーマットの説明は省略する）。

評価対象とする方式は以下の3種類とする。

A　アドレス変換なし
B　ソフトウェア全体のアドレス変換
C　改良方式

評価結果を表1と表2、図4に示す。ソフトウェア全体にアドレス変換をかけた場合（B）、アドレス変換なしの場合（A）よりコマンド数、差分データサイズともに劣るケースがあるのに対し、提案した改良方式では全てのケースで他方式より優れている。特に差分データに関しては8～17%のサイズ削減効果があることが分かる。

7. おわりに

本論文では組込みソフトウェアをターゲットとした差分抽出方式を提案し、その評価を行った。その結果、提案した方式が差分データサイズの削減という目的を十分に果たすことが分かった。

今後の課題としては、まずM32R以外のCPU向けソフトウェアに対しても提案した手法が有効かどうかを評価することがある。また、差分データサイズを小さくするためには効率的なフォーマットを採用することが非常に重要であることがわかっているので、提案した手法に適した差分データのフォーマットを検討、評価することも今後の課題である。

8. 参考文献

[1] S. Hoshi et al., “Software update system using wireless communication,” NTT DoCoMo Technical Journal, vol.5, no.4, pp.36-43, March 2004.
[2] 清原他, “携帯電話ソフトウェア更新のためのバージョン間差分表現方式,” 電子情報通信学会論文誌, vol.J89-B, no.4, pp.478-487, April 2006.
[3] http://cis.poly.edu/zdelta/
[4] Brenda S. Baker, Udi Manber, and Robert Muth, “Compressing Differences of Executable Code,” In Proceedings of ACM SIGPLAN 1999 Workshop on Compiler Support for System Software (WCSSS'99), May 1999.
[5] Y. Okada and K. Terazono, “A New Delta Compression Algorithm Suitable for Program Updating in Embedded Systems,” In Proceedings of IEEE Data Compression Conference (DCC'03), March 2003.