M-010

インターネットを利用した 大規模ソフトウェアのバージョンアップ方式に関する検討

A Study of Updating Large Size Software via Internet

栗原　まり子† 清原　良三† 渡辺　拓‡ 古嶋　寛之‡ 橘高　大造†
Mariko Kurihara Ryozo Kiyohara Taku Watanabe Hiroyuki Furushima Taizo Kittaka

１．まえがき

業務用パッケージなど比較的サイズの大きなソフトウェア（数十～数百 MB 程度，以後大規模ソフトウェアとする）は，製品出荷後もバージョンアップを定期的に繰り返す運用が多い．そのためこのような大規模ソフトウェアでは，開発だけでなく，バージョンアップ業務も効率的に運用することが要求される．

バージョンアップの方法としては， CD などオフラインでのメディア配布方式の他，インターネットの普及した近年では，バージョンアップ用の配布データを web サイト上に公開し，ユーザがインターネットを介してダウンロードする方式も広く普及するようになった．

本論文では，大規模ソフトウェアにおいてこのようなインターネットを利用したバージョンアップを行う際の運用上の課題，課題の解決方式及び評価結果について述べる．

2．運用上の課題

インターネットを利用したダウンロードによるバージョンアップ方式では，ダウンロードする配布データのサイズが大きくなると，ダウンロード時間も長時間となり，ユーザへの負担が増す．バージョンアップ用の配布データサイズの削減方法としては，新規インストールのようにソフトウェアを構成する全ファイルを含めるのではなく，新規に追加されたファイル，更新の発生したファイル（以後，これらのファイルを差分ファイルとする）のみを集めて配布データとする方式（差分ファイル方式）が一般的である．

しかしこのような差分ファイル方式を採用した場合でも，バージョンアップを行えば，追加ファイルの数や更新ファイルのサイズが増えるので，結果的に配布データのサイズも増加することになる．特に大規模ソフトウェアにおいては，小規模なソフトウェアに比べ，ファイル数が多く，ファイルサイズも大きいため，バージョンアップ時の増加サイズも大きくなる．このため，バージョンアップを繰り返せば，配布データサイズも増加を続け，インターネット経由でのダウンロードを短時間で行えないレベルのサイズに達することが懸念される．

近年では通信回線の低コスト化が進み，安価で高速な回線も普及しつつあるが，ユーザの使用する通信環境は多様であり，依然，旧式の低速な回線を利用するユーザも存在する．

† 三菱電機 情報技術総合研究所
Mitsubishi Electric Corp．Information Technology R&D Center
‡ 三菱電機 名古屋製作所
Mitsubishi Electric Corp．Nagoya Works

このため特に大規模ソフトウェアの場合，ダウンロード時に低速回線ユーザであってもストレスを感じさせないよう，バージョンアップを繰り返しても配布データのサイズ増加を抑制する機構が必要となる．

3. 実現方式

前章で述べた差分ファイル方式は，OS やウィルスソフトなどのバージョンアップにも採用されており，汎用的な方式となっている．

しかし差分ファイル方式では，サイズの大きなファイルで変更箇所が数バイトしかないような更新でも，バージョンアップ時にはファイル全体を配布しなくてはならず，配布データのサイズ増加に繋がる．

そこで，更新ファイルに関しては新旧のファイル間のバイナリ差分を取り，この差分データをバージョンアップ時の配布データとする方式（以後バイナリ差分方式とする）について検討した．バイナリ差分方式は，配布データのダウンロード後，差分データを旧版ソフトウェアに適用して新版に書き換える機構を別途用意する必要があるが，上記のようなファイル更新の場合，差分ファイル方式に比べ，配布データのサイズを大幅に削減することが可能である．

また大規模ソフトウェアの場合，サイズの大きなファイルの存在する割合も高いと思われ，バイナリ差分方式による配布データサイズの削減効果はより大きくなる．

4.バイナリ差分方式の評価

ある業務用パッケージソフトウェアを対象に，バージョンアップ時の配布データを，従来の差分ファイル方式と今回検討したバイナリ差分方式でそれぞれ作成し，バイナリ差分方式でのサイズ削減効果について検証した．

4.1　調査方法

今回調査に使用したソフトウェアの概要を以下に示す．

- ソフトウェアは windows プラットフォーム上で動作するもので，調査した版のソフトウェアサイズ（構成ファイルの合計サイズ）は，約 130～140MB である．
- バージョンアップの発生頻度は年に３～４回程度である．
- 配布データのサイズがより大きくなる最悪値に近いケースでの調査を目指し，マイナーチェンジ（リビジョンアップ）相当の改修を 12 回実施した場合のバージョンアップケースを対象とする．
- 調査に使用した旧版・新版各版のファイル数・サイズを表 1 に示す．

表 1．対象ソフトウェアの概要

	旧版	新版	新版の増分
ファイル数	741	803	62
ファイル サイズ合計	133,559,583 (133MB)	140,748,714 (140MB)	7,189,131 (7MB)

バイナリ差分方式による配布データサイズ削減効率を検証するため，以下の項目を調査した．

- 差分ファイル方式で作成した配布データのサイズ（新規追加ファイルの合計サイズ＋更新ファイルの合計サイズ）
- バイナリ差分方式で作成した配布データのサイズ（新規追加ファイルの合計サイズ＋更新ファイルのバイナリ差分データの合計サイズ）

バイナリ差分データは，更新ファイル毎に新旧バージョン間で差分抽出を行い，バイナリ差分ファイルとして作成した．差分抽出には，rsync・GDiff のアルゴリズムを利用した [1][2].

4.2 調査結果

新版ファイルの旧版からの変更状況を表 2 に示す．

表 2.新版ファイルの変更種別

変更種別	ファイル数	ファイルサイズ (byte)
同一ファイル（A）	588 (73%)	87,943,162 (62%)
更新ファイル（B）	159 (19%)	47,799,071 (34%)
追加ファイル（C）	66 (8%)	5,006,481 (4%)
合計	803	140,748,714 (**圧縮時** 82,639,371)

配布データ中の追加ファイル(C)のサイズについては，バイナリ差分方式の場合もファイルそのもののコピー処理が必要なので，差分ファイル方式と同じサイズとなる．

なお新版の全ファイルサイズ中，更新ファイルと追加ファイルのサイズ比率が約 4 割を占めており，前章で述べた通り，今回使用したサンプルは更新量が多いと言える．

次に上記の更新ファイル（B）に関して作成した，バイナリ差分ファイルの合計サイズを以下の表 3 示す．

表 3．バイナリ差分方式の差分ファイルサイズ

	(byte)
差分ファイル（D）	13,278,879

これよりバイナリ差分方式の場合，更新ファイル分の配布データサイズは，従来の差分ファイル方式の約 28%となり，約 3 分の 1 に削減出来ることを確認した．なお今回調査したサンプルは，改修量が多いケースであるため，改修量が少ない場合はさらなる削減効率を期待できると考える．

次に，両方式の配布データのサイズを表 4 に示す．配布データのサイズは，バイナリ差分方式とした場合，非圧縮時で差分ファイル方式の約 34%,圧縮時で約 57%という結果であった．また配布データを全ファイル送信とする方式に比べ，約 10 分の 1 となることを確認した．

今回は更新量の多いケースであるため，更新量の少ないケースでは，さらなる削減効果を期待出来るものと考える．

表 4．配布データのサイズ

上段：非圧縮時　下段：圧縮時

	バイナリ差分方式	差分ファイル方式
追加ファイル（B）	5,006,481 (1,583,917)	5,006,481 (1,583,917)
更新ファイル（C）		47,799,071 (12,796,422)
差分ファイル（D）	13,278,879 (6,627,189)	
配布データサイズ	18,285,360 (8,211,106)	52,805,552 (14,380,339)
配布データを全ファイルとした場合とのサイズ比率	10% (13%)	38% (17%)

5. 今後の課題

本論文では，大規模ソフトウェアのバージョンアップをインターネット経由で配布データをダウンロードして行う場合の問題点に着目し，バイナリ差分方式による配布データのサイズ削減方式の検討・評価を行った．今後は，本方式による配布データサイズの削減効果について，更新量の少ないケースなども含めた詳細評価を行う予定である．

また今回検証したパッケージソフトウェアに関し，以下のような運用面の課題も考慮し，方式検討・評価を進める予定である．

- ユーザサイト内での配布データの２次配布

１つのユーザサイト内で多数の端末にインストールして使用するケースが多く，インストール端末はインターネットに接続されていない場合が多いので，バージョンアップは別のインターネットに接続された端末で配布データをダウンロードし，その後サイト内の複数の端末に配布/適用する２次配布による運用を行っている．

- バイナリ差分方式特有の運用方式の検討

バイナリ差分方式とした場合，旧版バージョン毎に，対応する配布データが異なる．このため上記の２次配布も含め，バージョンアップに関する運用モデルの整理，効率的なダウンロード・インストール方式，及びバージョン管理方式などについても検討する必要がある．

参考文献

[1] W3C, “Generic Diff Format Specification”, http://www.w3.org/TR/NOTE-gdiff

[2] A. Tridgell, P.Mackerras,"The rsync algorithm", http://cs.anu.edu.au/techreports/1996/TR-CS-96-05