5Y-1　マルチウインドウ上でのイメージデータ圧縮/伸張制御方式

坂本 幸夫
NTT　電気通信研究所

1.はじめに

多機能ワークステーション上では、マルチウインドウをベースに種々の図形、イメージデータの作成/編集が可能になっている。[1]

本操作を行うアプリケーション(AP)では、ウインドウ内に表示されたイメージデータを蓄積(ファイル化)し、本ファイルから自ノードおよび他ノードで再度表示する、等の処理が必要となる。

蓄積においては、データ圧縮の処理が施され、結果としてファイル量の節約、およびネットワーク間でのデータ転送におけるトラフィック量の削減/転送時間の短縮が図られる。

データ圧縮の効果は大きいが、イメージデータの圧縮処理において、圧縮後の結果量(例えば、バイト長)は圧縮される各々のデータが保有する情報量に依存し、圧縮前には予測できないため、どの程度のバッファ量を用意すべきか判断できないというプログラム記述上の問題がある。

本論文では、この問題点を解決する制御方式を検討したので報告する。本報告では、伸張処理は圧縮処理の逆操作であるため、圧縮処理の場合に例をとり論ずる。

2.処理の流れ

画面上のデータを圧縮/伸張する操作を図1に示す。APは、ウインドウ内に表示されている矩形領域を圧縮し、ファイル化する。再表示には、圧縮されたファイル(圧縮ファイル)から伸張処理を行い復元化する。また、圧縮ファイルはLAN経由で他ノードに転送して同様に操作される。

3.想定モデルと問題点

二次元のイメージデータを圧縮する際のモデルを、図2に示す。構成としては、AP、OS、二次元の被圧縮データ、圧縮ファイル、からなる。OS内には圧伸を制御する管理部があり、圧伸の処理を実施する。APには、一つのプロセスが保有できる最大のバッファUBUFがあり、そのバッファから圧縮用のバッファbufが切り出される。APは、システムコール等の命令によりOSに圧縮処理を依頼する。正常な処理時のフローを、以下にしめす。

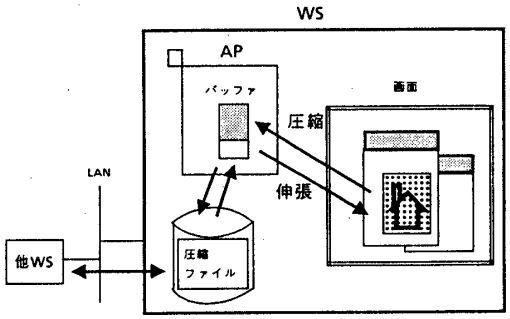

図1.圧縮/伸張処理の流れ

(i)APは,UBUFから圧縮用のバッファbufを切り出す。
(ii)APはbuf,および圧縮すべき領域、等をパラメータとしてOSに命令を発行する。
(iii)OS内の圧伸管理部では、その情報をもとに二次元のイメージデータを圧縮し、bufに格納する。
(iv)圧縮処理後、APにリターンする。
(v)buf内のデータを、ファイルに格納する。

本処理においては、以下の問題点が内在する。

(1)バッファのサイズ予測:上記処理(i)でバッファbufの切り出し時、サイズをどの程度に見積もるか圧縮前には予測出来ない。

(2)バッファ不足:上記処理(iii)で圧縮データがbufのサイズをオーバーしたときの対処をどうするか。

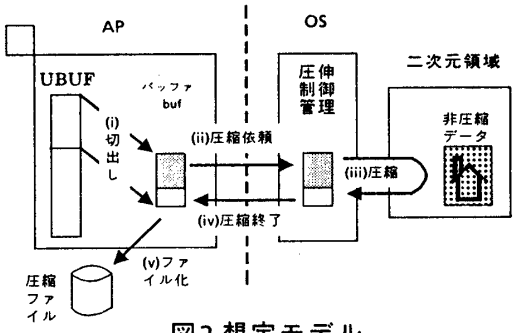

図2.想定モデル

Method of Controlling Compressed/Expressed Image Data on Multiple Window System
Yukio SAKAMOTO
NTT Electical Communications Laboratories

(1)に関しては、一般的な解決策がないため適当なサイズ(例えば、二次元データの数分の一等)に見積もるしかない。(2)に関して、以降に制御方式を述べる。

4.方式案

前記した問題点(2)の解決方式としては、以下の2案が考えられる。

(i)単独操作方式:図2中の圧縮操作において、1回のAP/OS間の会話処理(インタラクションと呼ぶ)で完結させる方式である。バッファ不足の時は、再度バッファサイズを大きくして、再実行させる。サイズの拡大量は、目安として現サイズの数分の一をとる。

(ii)連続操作方式:図2中の圧縮操作において、複数回のインタラクションにより、継続的に処理を実行する。バッファ不足の時は、既に処理したデータをファイル化し、次のインタラクションでは現処理からの継続として圧縮処理を続ける。

両案の処理概要を、図3に示す。

5.方式比較

圧縮処理時間と被圧縮される二次元のデータ量の観点から、両方式の比較を表1に示す。

表中、バッファ不足時における連続方式の処理時間T3は、ファイル化に要する時間fが小さいことから殆どT1に等しくなる。単独方式の処理時間T2は、圧縮処理に失敗した時間分$\sum_{\lambda=1}^{n-1} C(i)$だけT3($\simeq$T1)より大きくなり、本差分はT1に比べて無視できない。そのため、T2>>T3となり連続方式の方が失敗処理がない分だけ処理時間は早くなり効率的である。

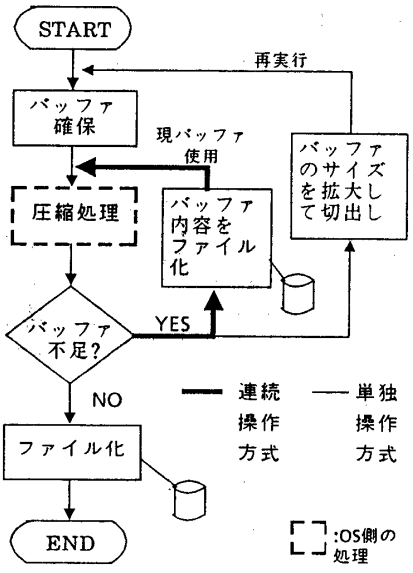

図3.処理概要(フロー)

表1.方式案の比較

項目＼方式		単独操作方式	連続操作方式
圧縮処理時間	バッファ充足時	被圧縮データ分の圧縮に要する時間T1	
	バッファ不足時	n回(n≧2)目に成功した場合の時間T2 $T2 = T1 + \sum_{\lambda=1}^{n-1} C(i) + f$	n回(n≧2)目に成功した場合の時間T3 T3 = T1 + f (T3<T2)
		C(i):i回目のインタラクション中の圧縮に要する時間 f:ファイル化に要する時間	
圧縮データ量		1回のインタラクションで処理を終わらせるため、バッファbuf内に圧縮データが格納される必要がある。そのため、被圧縮されるデータの領域が制限される。	圧縮データを継続しながら、分割してファイル化できることから、左記の制限はない
ソフト規模		少	大
評価		△	○

インプリメントのソフト規模から見ると、単独方式は簡明なトライアンドエラー方式であるため、小規模で実現可能である。連続方式では、処理の継続状態を保持するため、バッファ不足時にOSからAPにリターンしてから再度圧縮命令をコールするまでの間に該APが不意に終了させられた場合、該APの圧縮命令待ちとなり他APの処理が受け付けられないウエイト状態が生ずる。そのため連続方式では、処理途中の状態保持や上記ウエイト状態の対処、等の複雑な制御機構が必要となるため、規模的に大きくなる。

両方式とも上記のように一長一短がある。連続方式の規模は、高々1Kステップ程度であり、上記の制御機構もOS内の基本機能を用いることにより実現可能である。そのため、総合的にみると、処理時間の効率性、被圧縮データ量に制約がなく分割的にファイル化して格納できる等の操作の良好さ、およびメモリ量の有効利用、等から連続方式の方が優れていると考える。

6.まとめ

上記した問題点は、圧縮処理過程において圧縮後のサイズが予測できないという不確定要素に起因する。本論文ではその一解決方式(連続方式)を示したが、本方式は、一般的に二次元イメージ領域をシーケンシャルなメモリ上に圧縮して格納する際にも適用できる方式である。

<参考文献>

[1]坂本、鈴木、谷口、横山、"EINSにおける分散処理システムの構成"、「LAN/マルチメディアの応用と分散処理」シンポジウム、P151,1984,10