(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5079110号
(P5079110)
(45)発行日　平成24年11月21日(2012.11.21)　(24)登録日　平成24年9月7日(2012.9.7)

(51)Int.Cl.　　F I
H03M 7/40 (2006.01)　　H03M 7/40
G06F 12/00 (2006.01)　　G06F 12/00　511A

請求項の数 12　外国語出願　(全 21 頁)

(21)出願番号　特願2011-30873(P2011-30873)
(22)出願日　平成23年2月16日(2011.2.16)
(65)公開番号　特開2011-193451(P2011-193451A)
(43)公開日　平成23年9月29日(2011.9.29)
審査請求日　平成23年2月18日(2011.2.18)
(31)優先権主張番号　12/707582
(32)優先日　平成22年2月17日(2010.2.17)
(33)優先権主張国　米国(US)

(73)特許権者　501216023
コンピューター　アソシエイツ　シンク
インク
Ｃｏｍｐｕｔｅｒ　Ａｓｓｏｃｉａｔｅｓ
Ｔｈｉｎｋ，　Ｉｎｃ．
アメリカ合衆国，ニューヨーク州　１１７
４９，アイランディア，ワン　コンピュー
ター　アソシエイツ　プラザ
(74)代理人　110000110
特許業務法人快友国際特許事務所
(72)発明者　マーラー，スティーブン　ダグラス
アメリカ合衆国　９４０８０　カリフォル
ニア州　サウスサンフランシスコ　ショア
ラインコート　６０００　スイート３００
コンピューター　アソシエイツ内
最終頁に続く

(54)【発明の名称】圧縮された整数データを記憶・転送するためのシステム

(57)【特許請求の範囲】
【請求項１】
ソフトウエアアプリケーションにおいて使われる整数値であって、前記整数値のバイナリ表現が、前記ソフトウエアアプリケーションにて扱われるバイナリ数の最大桁数に等しい桁数を有する整数値をエンコードする方法であり、
ａ）前記整数値のバイナリ表記における桁数Ｎを決定するステップ；
ｂ）Ｎのバイナリ表記における桁数Ｎ’を決定するステップ；
ｃ）次の要素を有するように前記整数値をエンコードするステップ；
１）Ｎ’－１個のゼロに等しい第１要素；
２）第１要素に付け加えられる第２要素であり、前記整数値のバイナリ表記から最上位ビットを除いた値に等しい第２要素；
ｄ）エンコードされた値を出力するステップ；
を備える方法。
【請求項２】
エンコードされた値を出力する前記ステップ（ｄ）は、エンコードされた値を、ネットワーク接続を介して第１コンピュータデバイスから第２コンピュータデバイスへ転送するステップを備える請求項１に記載の方法。
【請求項３】
エンコードされた値を出力する前記ステップ（ｄ）は、エンコードされた値を永続性記憶媒体へ記憶するステップを備える請求項１又は２に記載の方法。

【請求項４】
エンコードされた値を出力する前記ステップ（ｄ）は、サブコンポーネントを一つずつ出力して前記エンコードされた値を出力するステップであって、最初に前記第１サブコンポーネントを、第２に前記第２サブコンポーネントをＦＩＦＯデータ構造体に出力するステップを備える請求項１から３のいずれか１項に記載の方法。
【請求項５】
エンコードされた値を出力する前記ステップ（ｄ）は、サブコンポーネントを一つずつ出力して前記エンコードされた値を出力するステップであって、最初に前記第２サブコンポーネントを、第２に前記第１サブコンポーネントをＬＩＦＯデータ構造体に出力するステップを備える請求項１から３のいずれか１項に記載の方法。
【請求項６】
バイナリ表現における桁数が前記最大桁数を上限とする整数域に含まれる整数の個数が、2^i（ただし、ｉは２のべき乗）に等しいことを特徴とする請求項１から５のいずれか１項に記載の方法。
【請求項７】
バイナリ表現における桁数が前記最大桁数を上限とする整数域に含まれる整数の個数が、４、１６、２５６、及び、６５５３６のいずれかであることを特徴とする請求項１から５のいずれか１項に記載の方法。
【請求項８】
（ｅ）前記整数値の少なくとも一部を既定の定数分だけ上位又は下位へシフトすることによって、データシステム内の整数域であってバイナリ表現における桁数が前記最大桁数を上限とする整数域を正規化するステップ；
をさらに備える請求項１から７のいずれか１項に記載の方法。
【請求項９】
ウエブアプリケーションとウエブサイトのパフォーマンスの問題を検知し診断するモニタリングシステムにおいて実行される整数値のエンコード方法であって、
当該システムは、モニタリングソフトウエアアプリケーションと、当該モニタリングソフトウエアアプリケーションによってモニタされるモニタ対象ソフトウエアアプリケーションを含んでおり、
当該モニタ対象とモニタリングソフトウエアアプリケーションは、一つのコンピュータデバイス内の１つのプロセッサ上で動作するものであり、
前記１つのプロセッサは、前記モニタリングシステム内で用いられる整数値であって、その整数値のバイナリ表現が、前記モニタリングソフトウエアアプリケーションにて扱われるバイナリ数の最大桁数に等しい桁数を有する整数値をエンコードする当該方法を実行するものであり、
当該方法が、
ａ）前記整数値のバイナリ表記における桁数Ｎを決定するステップ；
ｂ）Ｎのバイナリ表記における桁数Ｎ’を決定するステップ；
ｃ）次の要素を有するように前記整数値をエンコードするステップ；
１）Ｎ’－１個のゼロに等しい第１要素；
２）第１要素に付け加えられる第２要素であり、前記整数値のバイナリ表記から最上位ビットを除いたものに等しい第２要素；
ｄ）エンコードされた値を出力するステップ；
を備えるエンコード方法。
【請求項１０】
前記１つのプロセッサが、前記ステップａ）からｄ）によってエンコードされた値をデコードする方法を実行するものであり、前記デコードする方法は、
ｅ）エンコードされた値の先頭のゼロの個数を決定するステップ；
ｆ）先頭のゼロの個数が、前記データシステムの前記最大整数値の前記最大バイナリ桁数の$\log_2$ の値以上であるか否かを決定するステップ；

ｇ）前記ステップ（ｅ）にて、前記先頭のゼロの個数が、前記データシステムの前記最大整数値の前記最大バイナリ桁数の$\log_2$の値以上であると決定された場合、2^{N-1}に等しい第１中間十進数Ｊを決定するステップ、ただし、Ｎは、前記最大整数値の前記最大バイナリ桁数に等しい値である；

ｈ）エンコードされた値からその最上位ビットを除いた値の十進数変換値に等しい値Ｋであって第２中間十進数Ｋを決定するステップ；

ｉ）第１中間十進数Ｊを第２中間十進数Ｋに加え、エンコードされた値によって示される整数値を提供するステップ；

を備える請求項９に記載の方法。

【請求項１１】

前記先頭のゼロの個数が、前記データシステムの前記最大整数値の前記最大バイナリ桁数の$\log_2$の値よりも小さい場合に、以下のステップ、

ｊ）エンコードされた値の中から前記先頭のゼロに続くＮ’ビットを読み出すステップ；ここで、Ｎ’は前記先頭のゼロの個数に１を加えた値であってＮのバイナリ表記の桁数に等しい値である；

ｋ）読み出したＮ’ビットの十進数変換値に等しい値Ｎを決定するステップ；

ｌ）前記ステップｈ）で決定された値Ｎを用いて、Ｊ＝2^{N-1}の式によって値Ｊを決定するステップ；

を備える請求項９又は１０に記載の方法。

【請求項１２】

請求項９から１１のいずれか１項に記載の方法であって、さらに、

（ｍ）前記整数値の少なくとも一部を既定の定数分だけ上位又は下位へシフトすることによって、モニタリングシステムによって使われている整数域であってバイナリ表現における桁数が前記最大桁数を上限とする整数域を正規化するステップ；

を備える方法。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、整数データを圧縮する技術に関する。

【背景技術】

【０００２】

インターネット人口が増加するにつれて、多くのビジネスがインターネット上で存在感を確立している。それらのビジネスは、典型的には、１つあるいはそれ以上の数のウエブアプリケーションが走るウエブサイトを立ち上げる。インターネット上でビジネスを行うことの一つの短所として、ウエブサイトがダウンしてしまうと、応答ができなくなる、あるいは、顧客に適切なサービスを提供できなくなってしまい、潜在的な顧客／売り上げを逃してしまうということがある。イントラネットやエキストラネットにも同様の課題がある。それゆえ、ウエブアプリケーションあるいはウエブサイトが適切に動作するようにそれらのパフォーマンスについての問題を積極的に検知し診断するという、サービス志向アーキテクチャ（Service Oriented Architecture: SOA）アプリケーションマネジメントソリューションが開発されている。そのようなアプリケーションマネジメントシステムの一つに、カリフォルニア州サウスサンフランシスコのCA, Inc., Wily Technology DivisionによるIntroscope（登録商標）がある。

【０００３】

Introscope（登録商標）のようなアプリケーションは、その動作において膨大な量のデータを取得し記憶するので、記憶容量の低減と転送帯域の確保のためにデータ圧縮技術が欠かせない。一般に、データを圧縮する手法は、記憶や転送に先立ってデータをエンコードする（符号化する）エンコードエンジンと、データ受信や記憶装置からのデータ取り出しに際してデータをデコードする（復号する）ための逆操作を行うことのできるデコードエンジンを備える。圧縮技術の一つのタイプとして、ロッシーデータ圧縮という技術が知

られている。その技術は、高倍率の圧縮ができる反面、デコードに際して、エンコードされる元データの解像度をいくらか犠牲にするものである。ロッシーデータ圧縮技術は、高帯域の転送レートが要求されるマルチメディアデータの転送にしばしば用いられるが、そこでは、デコードされたメディアにおけるデータ忠実性の低下は知覚できるほどではない。

【０００４】

他の状況においては、デコードされたデータがエンコードされたデータの完全な復元であるという、無損失のデータ圧縮が要求される。正値整数の無損失圧縮技術として、１９７０年代にＭＩＴ教授のPeter Eliasによって開発されたエリアスデルタコード化（符号化）アルゴリズムが良く知られている。図１の従来技術のフローチャートを参照し、整数値ｘ＝１３のエリアスデルタコード化を一例として、自然数ｘIN＝｛１，２，３，・・・｝をコード化する従来のエリアスデルタ技術を説明する。ステップ４０では、整数データ値をバイナリフォーマットで受け取る。

$13_{base10} = 1101_{base2}$

ステップ４４では、ステップ４２におけるバイナリ値の桁数Ｎを決定する。

1011はＮ ＝ ４桁を有する。

Ｎは、元の整数値ｘに含まれる最も大きな２のべき乗値の指数の整数値がＮ－１であるということから、数学的に決定される。別言すると、N-1＝ Log_2 x に含まれる最大の整数、ということである。このことは、一般に、Ｎ－１＝ Log_2 x と表記され、本明細書でもそのような表記を用いる。

$N\text{-}1 = \mathrm{Log}_2\ 13 \rightarrow\ N\text{-}1 = 3;\ N = 4.$

Ｎを記述する他の手法は次のとおりである。即ち、Ｎは、ｘのバイナリ表記における桁数であり、Ｎ－１は、ｘのバイナリ表記から最上位ビット（ＭＳＢ）を除外したものの桁数である。

【０００５】

ステップ４６では、Ｎをバイナリ数に変換する。

$4_{base10} = 100_{base2}$

さらにステップ４８では、Ｎのバイナリ表記の桁数Ｎ’を決定する。

100 は、N’ = 3 の桁数を有する。 $N'\text{-}1 = 2$

繰り返すが、Ｎ’は、Ｎに含まれる最も大きな２のべき乗値の指数の整数値がＮ’－１であることから、数学的に決定される。

$N'\text{-}1 = \mathrm{Log}_2\ 4 \rightarrow\ N'\text{-}1 = 2,\ N' = 3$

【０００６】

元の整数ｘに対するエリアスデルタコードE_δ（ｘ）は、次の３つの部分から得られる。

$E_\delta(x)_1$ ＝ Ｎ’－１個のゼロ（ステップ４８）

$E_\delta(x)_2$ ＝ Nのバイナリ表記（ステップ４２）

$E_\delta(x)_3$ ＝ ｘのバイナリ表記の残りＮ－１桁（即ち、ｘのバイナリ表記からMSBを除いたもの）

それゆえ、$E_\delta(13)$は次の（数１）のとおりである。

【数１】

$$E_d(13) =$$
$$E_d(13)_1 = 00$$
$$E_d(13)_2 = 100$$
$$E_d(13)_3 = 101 = 101$$
$$E_d(13) = 00100101.$$

図２の従来技術は、整数０から１６まで、及び、３２、６４、１２８に対するエリアスデルタコードを示している。

【０００７】

整数値に対するエリアスデルタコードが得られたら、その値はステップ５０にて出力される。出力は、プロセッサから永続記憶媒体へ出力される場合があれば、プロセッサから、他のコンピュータデバイスへ送信するための通信インタフェイスへ出力される場合もある。整数値のエリアスデルタコードは、永続記憶媒体に記憶されることもあれば、単一の値として、ネットワーク接続を介して送信される場合もある。あるいは、図３の従来技術に示すように、サブコンポーネント（サブ要素）$E_{\delta}(x)_1$，$E_{\delta}(x)_2$ 及び $E_{\delta}(x)_3$ の形式で送信される場合もある。即ち、ステップ５２において、$E_{\delta}(x)_1$がFIFOデータ構造体に記憶されるか又は送信される。ステップ５４において、$E_{\delta}(x)_2$がFIFOデータ構造体に記憶されるか又は送信される。そしてステップ５６において、$E_{\delta}(x)_3$がFIFOデータ構造体に記憶されるか又は送信される。サブコンポーネントは上記した順で記憶され送信され、それらがFIFO構造体から取り出されるか受信されたときには、エリアスデルタコードの他のサブコンポーネントをデコードする必要があるそれらサブコンポーネントは、適切な順序で調べられる。エリアスデルタコード値から元の整数値を復元するために、そのエリアスデルタコード値に対して逆の操作が実行される。

【発明の概要】

【発明が解決しようとする課題】

【０００８】

エリアスデルタコード化技術の一つの課題は、あらゆる入力整数値に対して機能しなければならないことであり、それゆえ、あらゆる整数に対してコードのための容量を確保しなければならないことである。図２の例に示されているように、例えばｘ＝６４のような大きな入力値に対して、$E_{\delta}(64)$は、もとのバイナリ値そのものよりも多くの記憶容量／帯域を必要とする（バイナリ値が１０００００００であるのに対して$E_{\delta}(64)$＝００１１１０００００００である）。しなしながら、現実のアプリケーションの多くにおいては、入力される整数値には制限がある。例えば、レジスタには、例えば４ビットというように与えられたサイズがある。他の事例として、多くのデータクラスでは、工業規格によって定められたサイズがある。例えば、マルチメディアのストリームにおけるカラー値の送信は０から２５５の範囲であり、８ビットに格納される。エリアスデルタコード化は、データシステムにはしばしば制限が決められており、入力値は既知の最小値と既知の最大値を有する帯域の範囲であるという、この知見を利用してはいない。

【課題を解決するための手段】

【０００９】

本発明は、概略すると、送信すべき／記憶すべき整数値を、エリアスデルタコード化などの従来の圧縮技術と比較してより高い圧縮率で圧縮するシステムに関する。特に、データシステムが、最小値だけでなく最大値も有する既定の整数域に亘る整数値を記憶及び／又は送信することを用いて、最大値の、あるいは最大値に近い整数値を、データ解像度の損失なく、従来システムよりも高比率で圧縮する。本技術は、エリアスデルタコード化とパラレルの関係にある。しかしながら、記憶すべき、あるいは、送信すべきバイナリ整数値の桁数が、与えられたデータシステムに予め定められた最大桁数に等しいと判定された場合、整数値の$\log_{(2)}$値のバイナリ表記の記憶及び／又は送信が、記憶又は送信から除外され、これによって、従来のエリアスデルタコード化と比較して高い圧縮率が得られる。

【００１０】

本技術の圧縮技術は、広範囲の様々なデータシステムにおける幅広いアプリケーションに適用することができる。しかしながら、一つのアプリケーションとして、本技術は、ウエブアプリケーションやウエブサイトが適切に動作することを確保するためにそれらウエブアプリケーションやウエブサイトのパフォーマンスの問題を積極的に検知し診断するアプリケーションマネージメントシステムに関連するデータを圧縮するのに用いられる。

【００１１】
本技術は、ハードウエア、ソフトウエア、あるいは、ハードとソフトの組み合わせで実現されてよい。本発明のためのソフトウエアは、ハードディスクドライブ、ＣＤ－ＲＯＭ、ＤＶＤ、光学ディスク、フロッピー（登録商標）ディスク、テープドライブ、ＲＡＭ、ＲＯＭ、あるいは、他の適切な記憶デバイスを含む、一つあるいはそれ以上のプロセッサ読み出し可能な記憶媒体に格納される。別の態様では、そのソフトウエアの一部あるいは全部は、カスタムＩＣ、ゲートアレイ、ＦＰＧＡ、ＰＬＤ、及び、特定目的のためのコンピュータを含む、専用ハードウエアで置き換えられ得る。一つの態様では、本発明が実装されたソフトウエアは、一つあるいはそれ以上のプロセッサをプログラムするのに用いられる。そのプロセッサは、一つあるいはそれ以上の、記憶デバイス、周辺装置、及び／又は、通信インタフェイスと通信し得る。
【図面の簡単な説明】
【００１２】
【図１】エリアスデルタコード化スキームによる整数値符号化の方法を示す、従来技術フローチャートである。
【図２】様々な整数に対するエリアスデルタコード化値を示す従来技術のテーブルである。
【図３】エリアスデルタコード化値のサブコンポーネントを送信する、あるいは、記憶するステップを示す従来技術フローチャートである。
【図４】本システムによる高値圧縮技法に基づく整数値コード化のフローチャートである。
【図４Ａ】本システムの別の実施態様による高値圧縮技法に基づく整数値コード化のフローチャートである。
【図５】本システムの高次圧縮技法によって得られたエンコード値のサブコンポーネントを出力するステップの実施形態を示すフローチャートである。
【図６】本システムの高次圧縮技法とエリアスデルタ技法によってエンコードされた０から１５の範囲の異なる整数値の比較を示すテーブルである。
【図７】本技術による高次圧縮技法とエリアスデルタコード化技法によってエンコードされた０から２５５の範囲のなかの幾つかの整数値の比較を示すテーブルである。
【図８】本技術による高次圧縮技法によってエンコードされた値をデコードするためのフローチャートである。
【図９】整数値とそれに対応する高次圧縮値を含む第１テーブルと、整数値とそれに対応する正規化された高次圧縮値を含む第２テーブルを示す。
【図１０】高次圧縮技法を用いてエンコードされた整数値を記憶及び／又は送信する実施例のブロック図である。
【図１１】本システムを実装可能なコンピュータシステム環境のブロック図である。
【発明を実施するための形態】
【００１３】
図４～１１を参照して本技術を説明する。それは、一般的に、既知の可能な整数域内の範囲に亘る整数データを圧縮するシステムに関する。本技術の実施例は、特に、予め定められた整数域の上限に近い高次整数値を圧縮するのに効果的であるので、本技術の実施例を、本明細書では、「高次圧縮技法」と称する。ウエブアプリケーションとウエブサイトが適切に実行されるようにそれらのパフォーマンスについての問題を検知し診断するＳＯＡアプリケーションを参照し、以下、高次圧縮技法を用いる一つの特定システムを説明する。そのようなアプリケーションマネジメントシステムの一つに、カリフォルニア州サウスサンフランシスコのCA, Inc., Wily Technology DivisionによるIntroscope（登録商標）がある。図１０のブロック図を参照してそのようなシステムを以下にて説明する。しかしながら、本明細書で説明する高次圧縮技法は、予め定められた整数域の範囲に亘る整数値が送信及び／又は記憶される広範囲に及ぶ様々なソフトウエアアプリケーションに適用することができる。

【００１４】
図４に、本技術の高次圧縮技法の実施例による整数値をエンコードするためのフローチャートを示す。この技法は、記憶／送信される整数値の取り得る範囲が最小値（たとえばゼロ）と所定の最大値で制限されているデータシステムにおいて整数値を圧縮するのに用いることができる。その範囲の最小整数値の典型はゼロであるが、他の実施形態では最小値はゼロである必要はない。
【００１５】
最大整数値に関していえば、多くのデータシステムやソフトウエアアプリケーションは、システムの制約によって通常は制限されるデータ値の範囲を扱う。例えば、データシステムは、レジスタ、アドレスバス、データバスを用いており、それらは、例えば、０から１５までの間の１６個の整数値を扱い得る４ビットといった、予め決められたサイズ、セット数を扱う。他の例として、多くのアプリケーションプログラムは、工業規格によって定められているサイズとフォーマットのデータを送信し記憶する。多くの例の一つとしては、ビットマップ画像やビデオフレームバッファにおけるピクセルの色を表すデータは、８ビットフォーマットでの２５６個の値の一つとしてしばしば送信され記憶される。他の通信、オーディオ、及び、ビデオのデータは、標準的な８ビット、１６ビット、などのデータ長で記憶される。
【００１６】
高次圧縮技法の実施例では、送信され記憶される整数値は、既知の最大値を有する整数域内のものであると仮定する。この最大値の情報は、以下で説明するように、高次圧縮技法を実装するエンコードエンジンとデコードエンジンにて用いられ、圧縮されたデータの圧縮率を向上する。実施例において、本システムは、2^iに等しい最大整数値の元で動作する。ここで、ｉは、２のべき乗である（ｉ＝２、４、８、１６、３２、６４・・・）。しかしながら、本システムの他の実施形態は、別の整数値範囲のデータを圧縮／デコードするように動作する。
【００１７】
図４のフローチャートを参照して、本システムの実施例を説明する。この処理は、４ビットで送信／記憶される１６個までの異なる整数値を送信あるいは記憶するデータシステムにて用いられる。それゆえ、その整数域内の整数値をバイナリ表記で表す際のビットの最大数は４ビットである。上述したように、本技術は、バイナリの桁数の最大数がそれよりも大きい、あるいは小さい他の実施形態のデータシステムでも用いることができる。本明細書においては、バイナリ表記したときの桁数が、与えられたデータシステムにて許容される最大桁数と同じである整数値を「高次整数値」と称する。例えば、この高次圧縮技法は、最大４ビットを占める１６個の整数値を扱うシステムで用いられる。そのようなシステムでは、８（$1000_{base\,2}$）から１５（$1111_{base\,2}$）までの整数値が高次の整数値と称される。その同じ整数値であっても、例えば８ビットで２５６個の整数値を扱うシステムにおいては、高次の整数値とは見なされない。
【００１８】
以下で説明する高次圧縮技法は、エンコードエンジン２３０（図１０）にて実行される。説明の目的をもって、最大１６個の整数値を許容するデータシステムにおいて任意に選択された整数値ｘ＝１４をエンコードする場合について、図４のフローチャートを説明する。しかしながら、そのようなデータシステムにおいて、本技術は、１６個の取り得る整数値のいずれであっても受け取ってエンコードできることが理解されるであろう。ステップ１００では、エンコードエンジンはバイナリ形式の整数値ｘを受信する。この例では、

$$x=14_{base(10)}=1110_{base(2)}$$

である。
【００１９】
ステップ１０４では、エンコードエンジン２３０は、その整数のバイナリ値の桁数Ｎを決定する。「背景」の欄で説明したように、このときＮは、$N-1 = \log_2 x$ より決定される。

$N-1 = \log_2 14 \rightarrow N-1 = 3, N = 4$

本技術分野の当業者であれば、整数値のバイナリ表記の桁数を決定する代替方法あるいは付加的な方法を理解することもあろう。

【００２０】

ステップ１０６では、エンコードエンジン２３０は、Nが既知の整数域の$\log_2$の最大値に等しいか否かを判断する。この例では、Ｎの最大値は４である。整数値ｘ＝４の場合Ｎは４に等しいので、エンコードエンジンは、ステップ１０８にて、受信した整数値に対するＮが既定の最大値に等しいことを示すフラグを設定する。このフラグを設定することは、図４では、「CHK = max」で表されている。ステップ１０６におけるＮの値が最大値よりも小さい場合は、ステップ１０８はスキップされる。ステップ１１４では、Ｎのバイナリ表記（$4_{base10} = 100_{base2}$）を用い、Ｎのバイナリ表記における桁数Ｎ’を決定する。

$N'-1 = \log_2 N$

$N'-1 = \log_2 4 \rightarrow N'-1 = 2, N' = 3.$

繰り返すが、本技術分野の当業者であれば、Ｎのバイナリ表記の桁数を決定する代替方法あるいは付加的な方法を理解することもあろう。

【００２１】

ステップ１１４以降、エンコードエンジンは高次圧縮値を決定する。ここでは高次圧縮値をM(x)で表す。整数値ｘが既定の最大値よりも小さい桁数Ｎを有する場合（即ち、ステップ１１８においてCHKがmaxに等しくない場合）、高次圧縮値M(x)は、ステップ１２０にてエリアスデルタ値と同じように決定される。即ち、高次圧縮値$M(x) = M(x)_1$に（$M(x)_1$の後ろに）$M(x)_2$を付加し、さらに（$M(x)_2$の後ろに）$M(x)_3$を付加したものである。ここで、

$M(x)_1$ = Ｎ’－１個のゼロ（ステップ１１４）；

$M(x)_2$ = xのバイナリ表示におけるビット数N（ステップ１０４）；

$M(x)_3$ = xのバイナリ表記の残りN-1桁（即ち、xのバイナリ表記からMSBを除いたもの）

である。

【００２２】

しかしながら、ステップ１１８において、整数値ｘが既定の整数値の範囲における最大ビット数を用いるものである場合、本システムは、エリアスデルタコード化技法のような従来の圧縮技法に比較して、エンコードされた値のサイズを低減する。ステップ１２４では、フラグが最大値に等しい場合、高次圧縮値$M(x) = M(x)_1$に（$M(x)_1$の後に）$M(x)_3$を付加したものとなる。ここで、

$M(x)_1$ = Ｎ’－１個のゼロ（ステップ１１４）

$M(x)_3$ = xのバイナリ表記の残りN-1桁（即ち、xのバイナリ表記からMSBを除いたもの）

である。

【００２３】

理解されるように、受信した整数値のバイナリ表記が、既定の範囲についての最大桁数を有する場合、最上位ビットを表す符号化値（即ち$M(x)_2$）のための容量を確保する必要がない。

【００２４】

ｘ＝１４とするこの事例では、バイナリ表記におけるビット数（即ち、４）は、既定の最大値に等しい。従って、

$M(14) = M(14)_1$に$M(14)_3$を付加したもの

$M(14)_1$ = Ｎ’－１個のゼロ、即ち２個のゼロ $\rightarrow$ $M(14)_1 = 00$;

$M(14)_3$ = xのバイナリ表記からMSBを除いた残りN-1桁

x = 14のバイナリ表記は 1110 $\rightarrow$ $M(14)_3 = 110$

$M(14) = M(14)_1$ に $M(14)_3$を付加したもの

$M(14) = 00110$

となる。

【００２５】

ステップ１３０では、高次圧縮値M(x)が出力される。この出力は、プロセッサから永続的記憶媒体への出力であったり、プロセッサから、他のコンピュータデバイスへの転送のための通信インタフェイスへの出力であったりする。整数値に対する高次圧縮値は、永続的記憶媒体に記憶されたり、単一の値としてネットワーク接続を介して転送され得る。あるいは、図５に示されているように、それはサブコンポーネント$M(x)_1$, $M(x)_2$ （場合によっては） $M(x)_3$.として転送／記憶されることもある。即ち、$M(x)_1$は、ステップ１３４にて、転送されるか、あるいはFIFOデータ構造体に格納される。その整数値のバイナリ表記が既定の最大桁数のものではない場合、ステップ１３６にて、$M(x)_2$が転送されるかあるいはFIFOデータ構造体に格納される。そうでない場合は、ステップ１３６はスキップされる。ステップ１３８にて、$M(x)_3$が送信されるかあるいはFIFOデータ構造体に格納される。サブコンポーネントは上記した順で記憶され送信され、それらがFIFO構造体から取り出されるか受信されたときに、高次圧縮値の他のサブコンポーネントをデコードする必要があるそれらサブコンポーネントは、適切な順序で調べられる。

【００２６】

あるいはサブコンポーネントは、転送されるか又はLIFOデータ構造体に格納され得る。その場合、サブコンポーネントは、上述した場合とは逆順で送信又は格納される。

【００２７】

さらに、上記説明した図４のステップは、上記したステップの結果と同じ結果が得られる他の概念的方法によって行われてもよい。そのような代替方法を図４Ａに示す。図４Ａは、図４と類似しているが、Ｎが既定の最大値と同じか否かのチェックが省かれている。その代り、上記説明したように、システムは、ステップ１０６にて、Ｎが高次のものであるか否かをチェックする。もしそうであれば、システムは、上記したとおり、ステップ１０７にて、$N'-1 = \log_2 N$ を用い、Ｎのバイナリ表記における桁数Ｎ’を決定し、ステップ１０９にて、$M(x)_1$の後に $M(x)_3$ を付け加えて $M(x)$ を設定する。ここで、$M(x)_1$ と $M(x)_3$は上記説明したとおりである。ステップ１０６にて、Nが高次のものでないと判定された場合、システムは、上記したとおり、ステップ１１１にて、$N'-1 = \log_2 N$ を用い、Ｎのバイナリ表記における桁数Ｎ’を決定し、ステップ１１３にて、$M(x)_1$の後に $M(x)_2$ を付けさらにその後に $M(x)_3$を付け加えて $M(x)$ を設定する。ここで、$M(x)_1$ と $M(x)_2$と $M(x)_3$は上記説明したとおりである。次いで上記説明したとおりステップ１３０にてM(x) が出力される。繰り返すが、この技術分野の当業者であれば、上記開示した方法のさらに別の方法を見出し得るであろう。

【００２８】

図６は、整数値が０から１５の範囲に属するシステムのテーブルを示す。図に示されているように、下位の整数値（例えば、ｘ≦７、及び、N＜最大値４）については、高次圧縮値M(x)はエリアスデルタ値$E_\delta(x)$と同じである。しかしながら、バイナリ表記が最大ビット数である整数値（８≦ｘ≦１５、及び、N＝最大値４）については、本圧縮技法M(x)は、整数値の転送及び／又は記憶に際して、エリアスデルタコードよりも容量の低減を提供する。

【００２９】

図７は、０から２５５の範囲に属する整数値（バイナリ表記の最大桁数＝８）に対する高次圧縮技法の例を示す。この例では、０から１２７の範囲の整数値は、エリアスデルタ値と同じ方法で計算された高次圧縮値となる。しかしながら、１２８から２５５の範囲の整数値（そのいくつかが図７に示されている）については、それらの整数は最大ビット桁数をとり、それらの高次圧縮値は、エリアスデルタコード化法と比較して桁数の低減が図れる。図に示されているとおり、１２８から２５５の範囲の整数に対してはエリアスデルタコードは１４桁を要するが、１２８から２５５の範囲の整数に対する高次圧縮値はたった９桁で済む。このことは、記憶容量と転送帯域の顕著な節約を提供する。

【００３０】

エンコードされた値が転送にて受信されるか、永続記憶装置から取り出されると、エンコードされた値はデコードされる。本システムは、さらに、デコードエンジン２３２（図１０）を備えている。デコードエンジン２３２は、上記した高次圧縮技法の逆を行うことができ、エンコードされた値を元の整数値に変換する。図８のフローチャートを参照してデコードエンジン２３２の処理を説明する。デコードエンジン２３２は、エンコードエンジン２３０と同じ既定の最小値と最大値を使って動作する。それゆえ、図８の例において、デコードエンジン２３２は、エンコードエンジン２３０と同じ整数域に対して動作するように構成されている。上記の例において、値は０から１５の範囲であり、バイナリ表記における最大桁数Nは４である。図４において、M(14)=00110であった。図８にて、記憶／受信された値00110のデコード（復号）を説明する。しかしながら、上記と同様に、デコードエンジンは、既定の整数範囲のどの値であってもデコードできる点が理解されるであろう。

【００３１】

デコードエンジン２３２は、扱う高次圧縮値M(x)の全部を受信することがあれば、FIFO又はLIFOデータ取得構造体から第１のコンポーネント$M(x)_1$を受信することもある。いずれの場合であっても、ステップ１４０において、デコードエンジンは、受信したあるいは記憶された値M(x) における先頭のゼロの並びである第１のコンポーネント$M(x)_1$を受信する。前の説明と同じように、先頭に並ぶゼロの個数を以下、N’-1と称することにする。

M(x)=00110の場合、先頭に２個のゼロの並びがある → N’-1=2，及び N’=3

【００３２】

高次圧縮値において、先頭に並ぶゼロの個数N’-1 は、次の関係によってデコードされるバイナリ値の最上位ビットNを示す。

$$N' - 1 = \mathrm{Log}_2\ N$$

整数値が０から１５の間に制限されているシステムにおいて、先頭にゼロが付いていないことは、整数値が０と１の間であることを示し、先頭にゼロが１個の場合は整数値が２から７の間であることを示し、先頭にゼロが２個（あるいはそれ以上）並んでいる場合は整数値が８から１５の間であることを示す。

【００３３】

上記の例では、整数値は０から１５の間に制限されている（Nの最大値は４である）。しかしながら、上記で示したように、本発明は、いかなる既知の整数域に対しても動作する。実施例においては、既知の整数域における最大の整数値は、2^iに等しい。ここで、ｉは、２のべき乗である（ｉ＝２，４，８，１６，３２，６４・・・）。それゆえ、実施例においては、この高次圧縮技法は、２値、４値、１６値（上記の例のとおりである）、２５６値、６５５３６値の整数域を扱うものである。

【００３４】

高次圧縮値の先頭に並ぶゼロの個数が、そのデータシステムにおける最大整数値のバイナリ表記の桁数の $\log_2$ の値以上である場合はいつでも、その高次圧縮値は、そのデータシステムにおけるバイナリ表記最大桁数を要する整数値を表す。従って以下のことが言える。

・０から３までの整数域の場合、与えられた高次圧縮値がその先頭に１個のゼロを有することは、その与えられた高次圧縮値がバイナリ表記で最大桁数N=2ビットを要する整数値であることを示す。

・０から１５までの整数域の場合（上述の場合）、与えられた高次圧縮値がその先頭に２個のゼロを有することは、その与えられた高次圧縮値がバイナリ表記で最大桁数N=4ビットを要する整数値であることを示している。

・０から２５５までの整数域の場合、与えられた高次圧縮値がその先頭に３個のゼロを有することは、その与えられた高次圧縮値がバイナリ表記で最大桁数N=8ビットを要する整数値であることを示している。

・０から６５５３５までの整数域の場合、与えられた高次圧縮値がその先頭に４個のゼロを有することは、その与えられた高次圧縮値がバイナリ表記で最大桁数N=16ビットを要す

る整数値であることを示している、等々。
このような技法と整数域の情報を用いることによって、デコードエンジン２３２は、高次圧縮値の先頭のゼロの個数だけから、その高次圧縮値がバイナリで最大桁数を要する数値を表していることを判断することができる。

【００３５】

整数値が０から１５の範囲である図８の例を続ける。ステップ１４０では、デコードエンジンは先頭のゼロの個数を読む。ステップ１４４では、先頭に２個以上のゼロが存在すると判定された場合（N’-1 が２以上である場合）、デコードエンジンは、自動的に、その整数値は８から１５の間であり、その整数値のバイナリ表記のビット数Nが４であると判断する。従って、ステップ１４４にて、N’-1が２以上であると判定された場合、ステップ１４６にてNに４がセットされ、１４８－１５０（後述）のステップはスキップされる。

【００３６】

ステップ１５２では、第１の中間十進数値Ｊが、2^{N-1}として決定される。Ｊは、最上位ビットの十進数表記を表す。M(x) = 00110の例では、N=4である。それゆえ、

$$J = 2^{N-1} = 2^3 = 8$$

である。ステップ１５８では、受信／記憶された値M(x) から最上位ビットを除いたものを十進数へ変換し、第２の中間十進数値Ｋが決定される。これは、M(x)の後ろのN-1ビットである。上記のM(x) = 00110の例では、N=4と決定される。それゆえ、N-1 = 3 であり、M(x)の後ろの３桁は110である。従って、この例におけるM(x) のN-1桁の十進数変換値Kは次のとおり与えられる。

$$110_{base2} = K_{base10} \rightarrow \quad K = 6$$

【００３７】

最後にステップ１６０にて、元の整数値ｘは、中間値ＪとＫを加算して決定される。

$$x = J + K$$
$$x = 8 + 6 = 14$$

図４から理解されるように、高次圧縮技法によって整数値１４はM(14) = 00110にエンコードされる。図８から理解されるように、高次圧縮値 00110 は、元の整数値１４にデコードされる。

【００３８】

上記の説明において、先頭のゼロの並びが、デコードするバイナリ値が最大ビット数を有していることを示している場合、ステップ１４８と１５０はスキップされ、このことは処理時間の低減をもたらすことに留意されたい。ただし、バイナリ表記における桁数が最大値よりも少ないと判定された場合には、ステップ１４８と１５９が実行される。例えば、最大ビット数が４であるシステム（その整数域内で１６個の整数値が取り得るシステム）において、デコードエンジン２３２が高次圧縮値M(x)=01101を受信した場合を仮定する。その場合、ステップ１４０にて先頭に１個のゼロがあり、N’-1は２以上ではない。従って、デコードエンジンは、ステップ１４８を実行し、M(x)において先頭のゼロの並びに続くN’個のビットを読み出す。01101を復号化するこの例の場合、先頭に１個のゼロがあり、従ってN’-1 = 1であり、N’ = 2である。従って、M(x)において先頭のゼロに続く２ビットが読まれる。M(x) = 01101をデコードするこの例の場合、それら桁は「11」である。ステップ１５０では、Nは、読み出したN’個のビットの十進法への変換に等しい。この例の場合は、

$$11_{base2} = 3_{base10} \rightarrow \quad N = 3$$

である。

【００３９】

前述したように次いでステップ１５２が実行される。第１の中間値Ｊは、N=3として、次のとおり決定される。

$$J = 2^{N-1} = 2^2 = 4$$

【００４０】

ステップ１５８では、第２の中間地Ｋが決定される。上記に示したように、Ｋは、受信／記憶された値 M(x) から最上位ビットを除いた残りの部分、即ち、M(x)の最後のN-1桁、の十進法への変換である。M(x) = 01101である上記例では、N=3であることが決定されている。即ち、N-1 = 2 であり、M(x)の最後の２桁は「01」である。従って、この例におけるM(x)のN-1桁の十進法への変換であるＫは次のとおり与えられる。

$$01_{base2} = K_{base10} \rightarrow \quad K = 1$$

【００４１】

最後に、ステップ１６０にて、中間値ＪとＫが加算され、元の整数値ｘが求まる。

$$x = J + K$$

$$x = 4 + 1 = 5$$

【００４２】

上記説明した図８のステップは、上記したステップの結果と同じ結果が得られる、他の概念的な方法によって実行されてもよいことが理解されるであろう。

【００４３】

上記説明したように、実施例では、いかなる既知の整数域であっても、Ｎ＝2^iを満足する最大バイナリ桁数Nを有する。ここで、ｉは２のべき乗である。この制約によって、デコーダ（復号器）が、エンコードされた高次圧縮値がその既知の整数域における高次整数値を表しているか否かを自動的に判定することができるようになる。ｉが２のべき乗であるという制約のないシステムでは、先頭のゼロの数に基づいたＮに関する仮定が成立しないので、高次圧縮値の復号化はより複雑になるであろう。例えば、Ｎが2^iに制限されており、取り得る整数値が１６個であるシステムにおいては、M(x)の先頭に２個のゼロが存在すれば、N’-1 = 2 であり、N’ は、100_{base2} あるいは $N = 4_{base\ 10}$ に等しい３ビットのバイナリ値であることが判る。

【００４４】

しかしながら、取り得る整数値は１６個ではあるがNが2^iに制限されていないシステムにおいては、先頭に２個のゼロが存在すればN = 4であると仮定することはできないのである。例えば、M(x)の先頭に２個のゼロが存在する場合、N’-1 = 2 であり、N’ は３ビットのバイナリ値であるが、それは、100 (N = 4)、101 (N = 5)、 110 (N = 6)、或いは、 111 (N = 7)のいずれかを取り得る。

【００４５】

しかしながら、より複雑であるにしても、この高次圧縮技法の実施例は、Ｎが2^i（ｉは２のべき乗である）に制限されていないＮの値を扱うことはできる。これは、そのような値をエンコードしたM(x) が、Nの取り得る値の夫々に対して一意に定まるからである。そのような実施例においては、符号化された値M(x)における先頭のゼロだけではNの値を決定することはできないが、M(x)の他の桁を調べる付加的なステップを実行して、M(x)を元の整数値にデコードすることはできる。

【００４６】

実施例においては、高次圧縮技法は、高次整数値を符号化するには効率がよいが、高次整数値よりも下位の整数値に対してはそうでもない。他の実施形態では、高次圧縮技法によるエンコードに先立って、所定の整数域内の整数値に所定の定数を加算及び／又は除算するという、既知の正規化（regularization）技術を採用することによって、この課題に対処し得る。その定数は、その整数域のサイズに依存する。１６値の整数域である上記した図６の例においては、その整数域の最上位において値をエンコードするビット数が、下位の値を符号化するのに必要なビットよりも小さくないので、この正規化ステップは、その利益を生じないことに留意されたい。この状況は、整数域が大きい場合ではない。

【００４７】

例えば、正規化された高次圧縮技法の利点は、図９から明らかである。図９は、８ビット整数域（２５６個の整数値）に亘る整数値符号化の一部を示している。正規化されていないバージョンでは、高次圧縮技法は、６４から１２７の範囲で値を符号化するのに、１２８を符号化する場合よりも多くのビットを使う。例えば、所定の値の範囲では、これは

準最適である。正規化プロセスでは、例えば、１２８から２５５を表すビットコードを３２から１５９を表すコードの代わりに用い、３２から１２７を表すビットコードを１６０から２５５を表すコードの代りに用いることによって、この問題に対処し得る。このことは、エンコード／デコードの過程で定数（その値は、その特定の整数域の範囲に依存する）を加算及び／又は減算することによって達成し得る。図９の例では、３２から１５９の間のいずれの値にも、正規化していない高次圧縮エンコードに先立って９６が加算され、１６０から２５５の間のいずれの値にも、正規化していない高次圧縮エンコードに先立って１２８が減算されている。このシフトの後、上記した高次圧縮技法が実行される。デコードに際してはその逆が実行される。

【００４８】

上述したように、高次圧縮技法の実施例は、整数データが記憶及び／又は転送される広範囲の様々な状況で用いられる。図１０のブロック図を参照して、高次圧縮技法が用いられる１実施例を以下で説明する。このシステムは、“Transaction Tracer” と題する、２００２年１２月１２日に出願された米国特許出願第10/318,272号により詳しく記載されている。その出願は、本技術の所有者に譲渡されている。その出願は、参照によりその全てが本明細書に組み込まれる。一般に、そのようなシステムは、コンピューティング環境におけるトランザクションをモニタする。１以上のトランザクションの組についてのデータが収集される。このデータは次いで、一組の評価に対してテストされる。評価に適合するトランザクションが報告される。一実施形態では、評価に適合しないトランザクションのデータは破棄される。報告されたデータは、ソフトウエアのどの部分が実行が遅いのか、あるいは、適切に機能していないのか、を特定するために用いられる。

【００４９】

一実施形態では、ユーザは、閾値追跡時間を設定するとともに、ソフトウエアシステム上で実行されている一つ、幾つか、あるいは全てのトランザクションについてトランザクション追跡を起動することができる。グラフィカルユーザインタフェイスを使って、実行時間が閾値追跡時間を超えているトランザクションがユーザに報告される。グラフィカルユーザインタフェイスは、追跡されたトランザクションにおいてどこで時間が費やされているか、ユーザが直ちに理解できるように、トランザクションについての報告をビジュアライズすることを含む。

【００５０】

図１０はアプリケーションパフォーマンスマネジメントツールのコンポーネントの概念図である。この図は、トランザクションモニタリングシステムによってモニタされる、管理対象アプリケーション２００を示している。トランザクションモニタリングシステムは、管理対象アプリケーション２００のバイトコードにプローブ２０２、２０４を付加することによって、そのバイトコードを計測する（即ち修正する）。プローブは、管理対象アプリケーションのビジネスロジックを変えることなく、そのアプリケーションについての情報の特定部分を計測する。エージェント２０８もまた、管理対象アプリケーション２００の同じ機器に実装される。バイトコードを修正することについての詳しい情報は、“System for Modifying Object Oriented Code” と題する米国特許第6,260,187号に記載されている。その特許は、本技術の所有者に譲渡されている。その特許は、参照によりその内容の全てが本明細書に組み込まれる。トランザクションモニタリングシステムは、さらに、管理対象のアプリケーションの方法が完了したときに、その方法が追跡メカニズムを開始あるいは終了するという、新しいコードを追跡メカニズムを起動する管理対象アプリケーションに付加することによって、バイトコードを計測する。

【００５１】

図１０は、また、エンタープライズマネージャ２２０、データベース２２２、ワークステーション２２４、及び、ワークステーション２２６を示している。管理対象のアプリケーションが動作すると、プローブ（即ち２０２、及び／又は２０４）がデータをエージェント２０８へ中継する。エージェント２０８は、データを収集し要約し、それをエンタープライズマネージャ２２０へ送信する。エンタープライズマネージャ２２０は、エージェ

ント２０８を介して管理対象アプリケーションからパフォーマンスデータを受信し、要求された計算を実行し、パフォーマンスデータをワークステーション（例えば、２２４、２２６）で使えるようにする。オプションではあるが、さらに後の分析に備えてパフォーマンスデータをデータベース２２２に送る。ワークステーション（例えば２２４、２２６）は、パフォーマンスデータを見るためのグラフィカルユーザインタフェイスである。ワークステーションは、人のオペレータがモニタできるようにパフォーマンスデータを見るための特別な画面を作成するのに使われる。一実施形態では、ワークステーションは、２個の主画面、即ちコンソールとエクスプローラを表示する。コンソールは、カスタマイズ可能な１組の画面にパフォーマンスデータを表示する。エクスプローラは、パフォーマンスデータを有意義に見ることができるようにパフォーマンスデータをフィルタする計算器と警報器を表示する。パフォーマンスデータを組織化し、操作し、フィルタリングし、表示するワークステーションの要素には、動作、計測、計算、計器類、継続的な収集、測量的分類、比較、スマートトリガ、及び、SNMP収集が含まれる。

【００５２】

図１０に示されているコンポーネント２００、２２０、２２２、２２４、及び、２２６は、それぞれ、分離されたコンピューティングシステム上で動作し、あるいは、２個以上のコンポーネントが一つのコンピューティングシステム内で組み合わされてもよい。これらのコンポーネントのいくつか、あるいは、全てに対するコンピューティングシステムは、様々な異なるタイプのコンピュータデバイスであってよく、それらは、パーソナルコンピュータ、ミニコンピュータ、メインフレーム、サーバー、携帯デバイス、モバイルデバイスなどを含む。コンポーネント２００、２２、２２２、２２４、及び、２２６のいずれかを記述する例示的なコンピューティング環境３００を、図１１に示し、以下、説明する。この創造的なシステムは、数々の他の汎用或いは専用コンピュータシステム、環境、あるいは形態とともに動作する。この創造的なシステムとともに用いられるのに適したコンピューティングシステム、環境、及び／又は形態の良く知られた例には、パーソナルコンピュータ、サーバコンピュータ、マルチプロセッサシステム、マイクロプロセッサベースのシステム、セットトップボックス、プログラマブルコンシューマエレクトロニクス、ネットワークＰＣ、ミニコンピュータ、メインフレームコンピュータ、ラップトップ及びパームトップコンピュータ、ハンドヘルドデバイス、それらのシステムやデバイスのいくつかを含む分散コンピューティング環境、等々が含まれるが、それらに限定されるものでもない。

【００５３】

図１１に参照される創造的なシステムを実装するための例示的なシステムは、コンピュータ３１０の形式として、汎用コンピューティングデバイスを含む。コンピュータ３１０のコンポーネントには、プロセッシングユニット３２０、システムメモリ３３０、及び、システムメモリを含む様々なシステムコンポーネントをプロセッシングユニット３２０に接続するシステムバス３２１を含むが、それらに限られるものではない。システムバス３２１には、様々なバスアーキテクチャを採用したローカルバス、周辺バス、メモリコントローラ、メモリバスを含む、様々なタイプのバス構造を採用し得る。例として、そのようなアーキテクチャには、工業規格アーキテクチャ（ＩＳＡ）バス、マイクロチャネルアーキテクチャ（ＭＣＡ）バス、エンハンスドＩＳＡ（ＥＩＳＡ）バス、ビデオエレクトロニクススタンダードアソシエーション（ＶＥＳＡ）ローカルバス、及び、メザニンバスとして知られているペリフェラルコンポーネントインターコネクト（ＰＣＩ）バス、を含むが、それらに限定されるものではない。

【００５４】

コンピュータ３１０は、様々なコンピュータ読取可能な媒体を含む。コンピュータ読取可能媒体は、コンピュータ３１０がアクセス可能ないかなる有用な媒体であってもよく、揮発性あるいは不揮発性媒体、着脱可能あるいは着脱不可媒体のいずれであってもよい。例として、コンピュータ読取可能媒体は、コンピュータストレージ媒体や通信媒体を備えるものであってもよいが、それらに限定されるものではない。コンピュータストレージ媒

体は、揮発性あるいは不揮発性、着脱可能あるいは着脱不可の媒体のいずれであってもよく、それらは、コンピュータ読取可能な命令、データ構造、プログラムモジュール、あるいは他のデータなどの情報記憶に関する如何なる方法あるいは技術によって実装されてもよい。コンピュータストレージ媒体には、ランダムアクセスメモリ（ＲＡＭ）、リードオンリイメモリ（ＲＯＭ）、ＥＥＰＲＯＭ、フラッシュメモリ、他のメモリテクノロジ、ＣＤ－ＲＯＭ、デシタル多用途ディスク（ＤＶＤ）、あるいは他の光学ディスク媒体、磁気カセット、磁気テープ、磁気ディスク装置、あるいは他の磁気記憶デバイス、あるいは、情報を記憶することができ、コンピュータ３１０からアクセス可能な他の媒体が含まれるがそれらに限定されるものではない。通信媒体には、典型的には、コンピュータ読取可能な命令、データ構造体、プログラムモジュールや、キャリアウエーブや他の転送機構による変調データ信号による他のデータがあり、さらにはいかなる情報配信媒体も含む。「変調データ信号」は、情報を信号に符号化する方式であり、１以上の特性の組あるいは変化を含む信号を意味する。例として、通信媒体には、有線ネットワークや直接ワイヤ接続などの有線媒体、音波、ＲＦ、赤外線などの無線媒体、あるいは他の無線媒体が含まれるがそれらに限定されるものではない。上記した媒体のいかなる組み合わせも、コンピュータ読取可能媒体の範囲に含まれる。

【００５５】

システムメモリ３３０は、ＲＯＭ３３１やＲＡＭ３３２のような揮発性や不揮発性のメモリの形態のコンピュータ記憶媒体を含む。スタートアップの際などにコンピュータ３１０内の要素間での情報の転送を補助する基本的なルーチンを含むベーシック入力／出力システム（ＢＩＯＳ）３３３は、典型的にはＲＯＭ３３１に記憶されている。ＲＡＭ３３２は、典型的には、プロセッシングユニット３２０上で実行される、及び／又は、直ちにアクセスし得るデータ及び／又はプログラムモジュールを含む。例として、図１１には、オペレーティングシステム３３４、アプリケーションプログラム３３５、他のプログラムモジュール３３６、及び、プログラムデータ３３７が示されているが、それらに限られるものではない。

【００５６】

コンピュータ３１０は、また、他の着脱式／着脱不可の、揮発性／不揮発性のコンピュータ記憶媒体を含む。例として、図１１には、着脱不可の不揮発性磁気媒体にデータを書き込むあるいはデータを読み出すハードディスクドライブ３４１、着脱式の不揮発性磁気ディスク３５２にデータを書き込むあるいはデータを読み出す磁気ディスクドライブ３５１が示されている。コンピュータ３１０は、光学媒体にデータを書き込む／データを読み出す光学媒体読取りデバイス３５５を含むこともある。

【００５７】

この例示的なコンピュータ環境において用いられる他の着脱式／着脱不可、揮発性／不揮発性のコンピュータ記憶媒体には、磁気テープカセット、フラッシュメモリカード、ＤＶＤ、デシタルビデオテープ、ソリッドステートＲＡＭ、ソリッドステートＲＯＭ、等々が含まれ得るが、それらに限られるものではない。ハードディスク３４１は、典型的には、インタフェイス３４０などの着脱不可のメモリインタフェイスを介してシステムバス３２１に接続されており、磁気ディスクドライブ３５１と光学媒体読み取りデバイス３５５は、典型的には、インタフェイス３５０などの着脱式メモリインタフェイスによってシステムバス３２１に接続されている。

【００５８】

図１１に示されており上記で説明したドライブとそれに関連するコンピュータ記憶媒体は、コンピュータ３１０のためのコンピュータ読取り可能な命令、データ構造体、プログラムモジュール、及び、他のデータの格納場所を提供する。例えば、図１１において、ハードディスク３４１は、オペレーティングシステム３４４、アプリケーションプログラム３４５、他のプログラムモジュール３４６、及び、プログラムデータ３４７を記憶するものとして描かれている。それらのコンポーネントは、オペレーティングシステム３３４、アプリケーションプログラム３３５、他のプログラムモジュール３３６、及び、プログラ

ムデータ３３７と異なっていてもよいし同じであってもよい。オペレーティングシステム３４４、アプリケーションプログラム３４５、他のプログラムモジュール３４６、及び、プログラムデータ３４７には、図に描かれるのに異なる番号が与えられており、それらは少なくとも異なるコピーである。ユーザは、キーボード３６２、通常マウスと呼ばれるポインティングデバイス３６１、トラックボールやタッチパッドなどの入力デバイスを介して命令や情報をコンピュータ３１０に入力する。他の入力デバイス（不図示）として、マイクロホン、ジョイスティック、ゲームパッド、サテライトディッシュ、スキャナなどがある。それらの、あるいは他の入力デバイスは、システムバス３２１に接続しているユーザ入力インタフェイス３６０を介してプロセッシングユニット３２０に接続されているが、パラレルポート、ゲームポート、あるいは、ユニバーサルシリアルバス（ＵＳＢ）などの他のインタフェイスやバス構造を介して接続されるものであってもよい。モニタ３９１や他のタイプのディスプレイデバイスもまた、ビデオインタフェイス３９０などのインタフェイスを介してシステムバス３２１に接続されている。モニタに加えてコンピュータは、スピーカ３９７やプリンタ３９６などの他の周辺出力デバイスを含んでおり、それらは、出力周辺インタフェイス３９５を介して接続されている。

【００５９】

コンピュータ３１０は、リモートコンピュータ３８０のような、１つあるいはそれ以上のリモートコンピュータとの論理的な接続を使ったネットワーク環境にて動作するものであってもよい。リモートコンピュータ３８０は、パーソナルコンピュータ、サーバ、ルータ、ネットワークＰＣ、ピアデバイス、あるいは、他の一般的なネットワークノードであってよく、典型的には、図１１ではメモリ記憶デバイス３８１しか描かれていないが、コンピュータ３１０のように上記説明した要素の多くあるいは全てを含んでいてよい。図１１に示した論理的な接続には、ローカルエリアネットワーク（ＬＡＮ）３７１と広域ネットワーク（ＷＡＮ）３７３が含まれるが、他のネットワークを含んでもよい。そのようなネットワーク環境には、オフィスにて一般的である、エンタープライズワイドコンピュータネットワーク、イントラネット、及び、インターネットがある。

【００６０】

ＬＡＮネットワーク環境を使う場合は、コンピュータ３１０は、ネットワークインタフェイスあるいはアダプタ３７０を介してＬＡＮ３７１に接続される。ＷＡＮネットワーク環境を使う場合は、コンピュータ３１０は、典型的には、モデム３７２や、インターネットのようなＷＡＮ３７３との通信を確立するための他の手段を有する。内線あるいは外線のモデム３７２は、ユーザ入力インタフェイス３６０や他の適切なメカニズムを介してシステムバス３２１に接続される。ネットワーク環境においては、コンピュータ３１０に描かれているプログラムモジュール、あるいはその一部は、リモート記憶デバイスに記憶されていてもよい。例として、図１１には、メモリデバイス３８１に格納されているリモートアプリケーションプログラム３８５が描かれているが、これに限定されるものではない。図に示しているネットワーク接続は一例であって、コンピュータ間の通信リンクを確立する他の手段を用いてもよい。

【００６１】

いくつかの実施形態では、トランザクションマネジメントシステムの一部あるいは全ては、ソフトウエアとして実装されてよい。そのソフトウエアは、１つあるいは複数のプロセッサ読取り可能な記憶デバイスに格納され、１つあるいは複数のプロセッサをプログラムするのに用いられる。図１０に戻り、エンタープライズマネージャ２２０は、さらに、エンコードエンジン２３０とデコードエンジン２３２を有するＣＯＤＥＣを備える。エンコードエンジン２３０とデコードエンジン２３２は、エンタープライズマネージャ２２０と送受信されるデータ、あるいは、エンタープライズマネージャに記憶されているデータに対して上記した高次圧縮技法を実行する。エンジン２３０と２３２を含むＣＯＤＥＣは、管理対象のアプリケーション２００、ワークステーション２２４、２２６、及び／又は、データベース２２２を搭載する１つあるいは複数のコンピュータシステムに代替的にあるいは付加的に備えられていてもよく、その場合、それらのコンピュータシステムに記憶

されるデータあるいは送受信されるデータを符号化する。

【００６２】

一般に、エンコードエンジン２３２は、コンポーネント２００、２２０、２２２、２２４、及び、２２６のいずれかに備えられた記憶媒体から整数値を受け取り、その値をエンコードし、エンコードした値を、コンポーネント２００、２２０、２２２、２２４、及び、２２６の１つあるいは複数の記憶媒体に記憶する。一実施形態では、エンコードエンジン２３２は、エンコードされた値を出力し、それらの値を格納するために新たなファイル及び／又はデータ構造体が作成される。同様に、デコードエンジンは、コンポーネント２００、２２０、２２２、２２４、及び、２２６のいずれかに備えられた記憶媒体からエンコードされた値を受け取り、それらの値をデコードし、コンポーネント２００、２２０、２２２、２２４、及び、２２６の１つあるいは複数に、デコードした値を出力する（表示することを含む）。一実施形態では、デコードエンジン２３２は、整数値を出力し、それらの値を記憶媒体に格納する、及び／又は出力するために新たなファイル及び／又はデータ構造体が作成される。

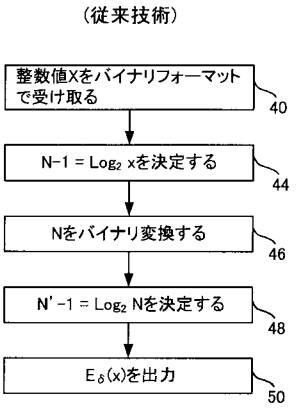
【００６３】

一実施形態では、図１１のシステムのユーザは、閾値追跡時間を指定することによって、エンタプライズマネージャに管理されているエージェントの全てあるいは幾つかについてトランザクション追跡を起動することができる。エージェント内のトランザクションでありその実行時間がその閾値レベルを超えてしまった全てのトランザクションは、追跡され、エンタープライズマネージャ２２０に報告される。エンタープライズマネージャ２２０は、追跡情報の要求を登録した適切なワークステーションにその情報を配信する。ワークステーションは、閾値を超えた全てのトランザクションをリストアップするＧＵＩを備えている。リストアップされたトランザクションの夫々に対して、追跡されたトランザクションにおいてどこで時間が費やされているかをユーザが直ちに理解できるように可視化が行われる。

【００６４】

以上、以上、本発明の具体例を詳細に説明したが、これらは例示に過ぎず、本発明の範囲を限定するものではない。上記した技術には、以上に例示した具体例を様々に変形、変更したものが含まれる。上記した実施形態は、本発明の原理をベストに説明するために選定されたものであり、その実用に際しては、本発明が最適にその有用性を発揮するように、その特定用途に適するように当業者が様々に変形し得る。本発明の範囲は、以下に記載した特許請求の範囲によって定義される。

【図１】

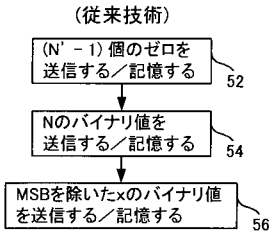


【図２】

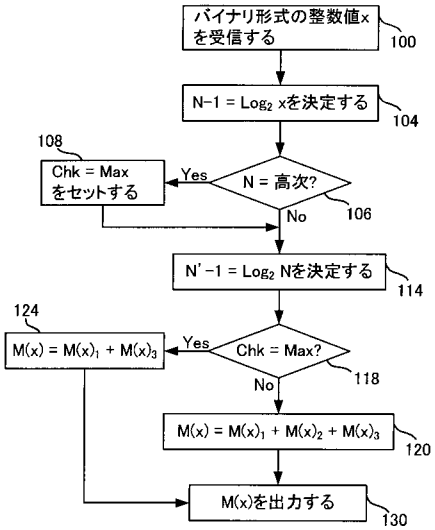
(従来技術)

整数 x	$E_\delta(x)$
0 →	10
1 →	11
2 →	0100
3 →	0101
4 →	01100
5 →	01101
6 →	01110
7 →	01111
8 →	00100000
9 →	00100001
10 →	00100010
11 →	00100011
12 →	00100100
13 →	00100101
14 →	00100110
15 →	00100111
16 →	001010000
...	...
32 →	0011000000
...	...
64 →	00111000000
...	...
128 →	00010000000000

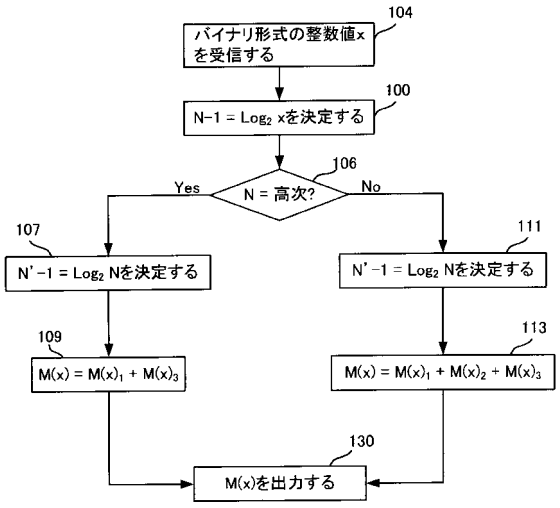
【図３】



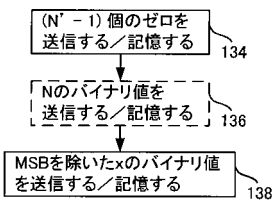
【図４】



【図４Ａ】



【図５】



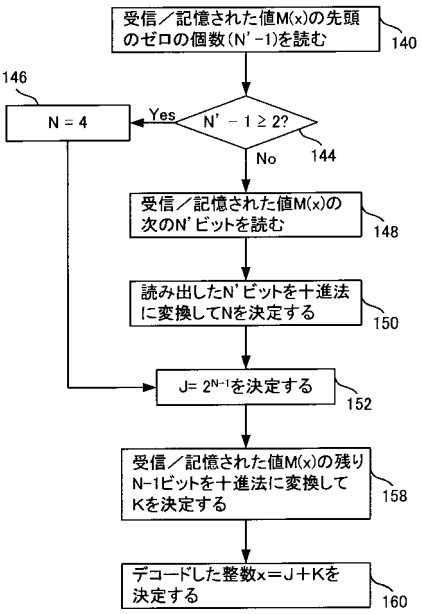
【図６】

整数	$E_8(x)$	M(x)
0:	10	10
1:	11	11
2:	0100	0100
3:	0101	0101
4:	01100	01100
5:	01101	01101
6:	01110	01110
7:	01111	01111
8:	00100000	00000
9:	00100001	00001
10:	00100010	00010
11:	00100011	00011
12:	00100100	00100
13:	00100101	00101
14:	00100110	00110
15:	00100111	00111

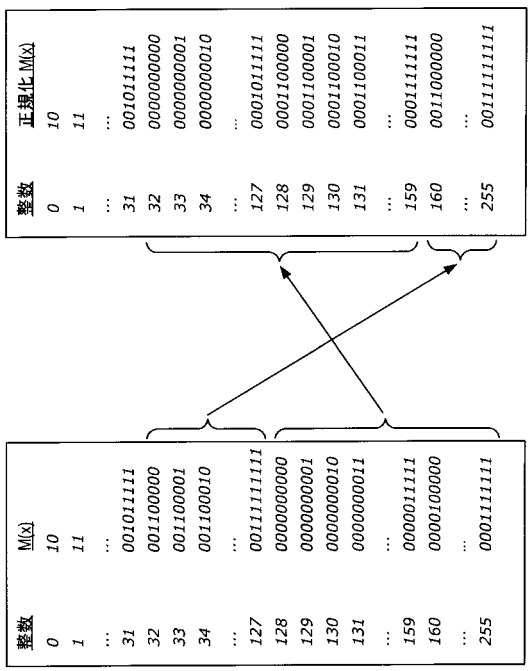
【図７】

整数	$E_8(x)$	$M_8(x)$
128	00010000000000	0000000000
129	00010000000001	0000000001
...	...	...
254	00010001111110	0001111110
255	00010001111111	0001111111

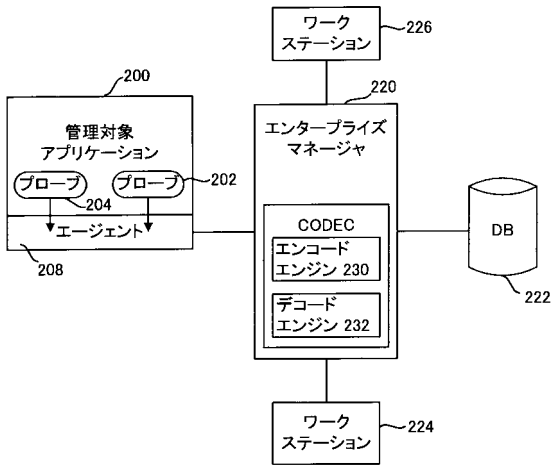
【図８】



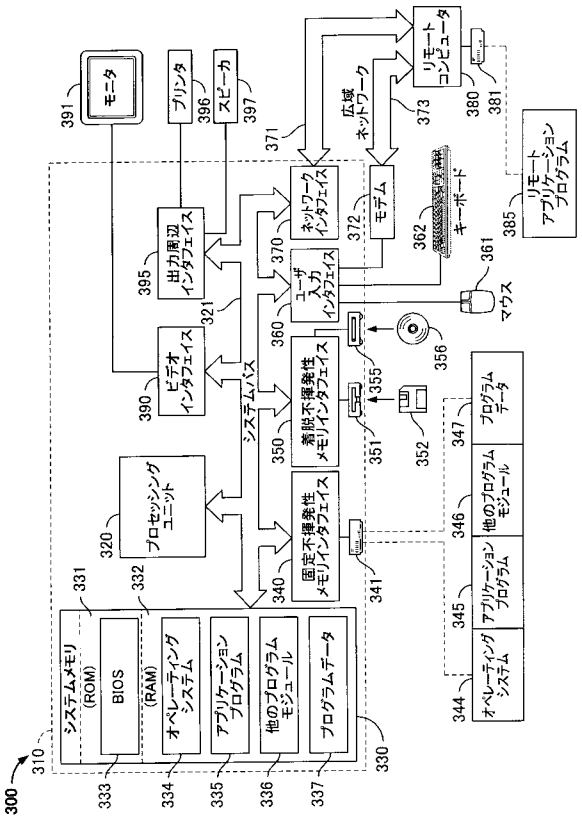
【図９】



【図１０】



【図１１】


300
310
システムメモリ
(ROM)
BIOS
(RAM)
オペレーティングシステム
アプリケーションプログラム
他のプログラムモジュール
プログラムデータ
330
331
332
333
334
335
336
337
プロセッシングユニット
320
システムバス
321
ビデオインタフェース
390
出力周辺インタフェース
395
固定不揮発性メモリインタフェース
340
着脱不揮発性メモリインタフェース
350
ユーザ入力インタフェース
360
ネットワークインタフェース
370
341
351
355
356
352
361
マウス
362
キーボード
モデム
372
広域ネットワーク
371
リモートコンピュータ
373
380
381
リモートアプリケーションプログラム
385
オペレーティングシステム
344
アプリケーションプログラム
345
他のプログラムモジュール
346
プログラムデータ
347
モニタ
391
プリンタ
396
スピーカ
397

フロントページの続き

審査官　北村　智彦

(56)参考文献　特開平１１－０８８１８９（ＪＰ，Ａ）
特開平１０－１５０３６６（ＪＰ，Ａ）
PETER ELIAS, Universal Codeword Sets and Representations of the Integers, IEEE TRANSACTIONS ON INFORMATION THEORY, １９７５年　3月, VOL.IT-21,NO.2

(58)調査した分野(Int.Cl., ＤＢ名)
Ｈ０３Ｍ　　３／００－１１／００
Ｇ０６Ｆ　　１２／００