(19)日本国特許庁(JP)　(12)特　許　公　報(B1)　(11)特許番号
特許第6306275号
(P6306275)
(45)発行日　平成30年4月4日(2018.4.4)　(24)登録日　平成30年3月16日(2018.3.16)

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

請求項の数 11 (全 41 頁)

(21)出願番号　特願2017-549547(P2017-549547)
(86)(22)出願日　平成28年10月20日(2016.10.20)
(86)国際出願番号　PCT/JP2016/081144
審査請求日　平成29年9月21日(2017.9.21)

早期審査対象出願

(73)特許権者　399037405
楽天株式会社
東京都世田谷区玉川一丁目１４番１号
(74)代理人　100116942
弁理士　岩田　雅信
(74)代理人　100167704
弁理士　中川　裕人
(72)発明者　山本　陽司
東京都世田谷区玉川一丁目１４番１号　楽天株式会社内

審査官　大桃　由紀雄

最終頁に続く

(54)【発明の名称】情報処理装置、情報処理方法、プログラム、記憶媒体

(57)【特許請求の範囲】
【請求項１】
１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得部と、
前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定部と、
一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定部と、を備えた
情報処理装置。
【請求項２】
前記判定部によって圧縮対象と判定された記事について圧縮を行い、既に圧縮した記事に対してのアクセスが発生した場合に当該記事の解凍を行う圧縮解凍部を備えた
請求項１に記載の情報処理装置。
【請求項３】
前記圧縮解凍部は、前記圧縮を行う場合に圧縮対象とされた記事に含まれる画像データの圧縮を行う
請求項２に記載の情報処理装置。
【請求項４】
前記圧縮解凍部は、前記圧縮を行う場合に圧縮対象とされた記事に含まれるデータのう

ち冒頭の所定のデータ以外のデータについて圧縮を行う
請求項２に記載の情報処理装置。
【請求項５】
前記ブログに記事を投稿する投稿者の利用状況に応じて当該投稿者に許可する記憶容量を増減させるブログ管理部を備えた
請求項１に記載の情報処理装置。
【請求項６】
アクセス要求に応じて既に圧縮された記事の解凍が行われた場合、前記判定部は、解凍から所定期間、当該記事を圧縮対象と判定しない
請求項２に記載の情報処理装置。
【請求項７】
前記判定部は、既に圧縮された記事について、同一のブログに属する他の記事に対するページビューの増加傾向を示す値に応じた解凍の是非を判定し、
前記圧縮解凍部は前記解凍の是非に応じて前記既に圧縮された記事を解凍しておく
請求項２に記載の情報処理装置。
【請求項８】
前記判定部は、記事内容に基づいて、既に圧縮した記事を解凍するか否かを判定する
請求項１に記載の情報処理装置。
【請求項９】
１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得ステップと、
前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定ステップと、
一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定ステップと、を
情報処理装置が実行する情報処理方法。
【請求項１０】
１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得機能と、
前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定機能と、
一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定機能と、を
情報処理装置に実行させるプログラム。
【請求項１１】
１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得機能と、
前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定機能と、
一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定機能と、を
情報処理装置に実行させるプログラムを記憶した記憶媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は情報処理装置、情報処理方法、プログラム、記憶媒体に関し、特にブログを管理するサーバ装置への適用に好適な技術に関する。

【背景技術】
【０００２】
インターネットを利用したサービスの一つとしてブログ環境を提供するサービスが知られている。ユーザはブログとしての任意の文章や画像による記事をサーバにアップロードし、当該ユーザの書き込み記事として保存してもらう。保存されたブログは、例えばウェブページの形式で一般に公開される。また、公開範囲を限定したものや、非公開のものなどがある。
多くのユーザは、自己発信のためのツールや、或いは個人的な日記代わりなどとして、このようなブログサービスを利用している。
特許文献１には、ブログに関する技術、例えば記事のアップロードに関する技術が開示されている。
特許文献２には、サーバの残容量不足に応じたコンテンツ削除に関する技術が開示されている。
【先行技術文献】
【特許文献】
【０００３】
【特許文献１】特開２００７－３２８７５０号公報
【特許文献２】特開２０１０－４４４６８号公報
【発明の概要】
【発明が解決しようとする課題】
【０００４】
サーバの残容量不足を解消するためには、ユーザごとに一定容量の記憶リソースを貸与する形式を取り、記憶リソースを超えたユーザに対しては新たな記事投稿の制限や自身のブログの記事を削除させるなどの方法が考えられる。
しかし、ユーザにとって記事投稿の制限や記事の削除を強いられることは、煩わしいと感じる虞がある。特に投稿した記事を削除しなければならないことに関しては、サービスの質が悪いと受け取られかねない。
【０００５】
そこで、アクセス可能性の低いブログや記事を圧縮することにより記憶リソースを確保し、圧縮された記事に対するアクセス要求があった場合には記事を解凍して配信することが考えられる。
但し、圧縮や解凍の処理は負荷が高い。アクセスの際に記事の解凍が必要となると、その処理時間により閲覧ユーザにウェブページのレスポンスの遅さを感じさせるなど閲覧時のパフォーマンス低下を生じさせる恐れがある。これらのことから、閲覧時における圧縮や解凍の処理回数は、可能な限り減らしたい。
そこで本発明は、圧縮するための条件を適切に設定することにより、閲覧時の解凍処理を可能な限り生じさせないようにしつつ、記憶リソースの有効な利用を図ることを目的とする。
【課題を解決するための手段】
【０００６】
本発明に係る情報処理装置は、１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得部と、前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定部と、一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定部と、を備えている。
ブログに含まれる記事によって当該ブログを記憶するためのリソースが圧迫された場合に、当該ブログに属する記事を圧縮することを考える。このときに、圧縮するか否かを判定するための閾値をブログごとに必要な記憶容量の増加傾向に基づいて決める。
【０００７】

上記した情報処理装置においては、前記判定部によって圧縮対象と判定された記事について圧縮を行い、既に圧縮した記事に対してのアクセスが発生した場合に当該記事の解凍を行う圧縮解凍部を備えていてもよい。

この圧縮解凍部は、判定部の判定に沿って適切に選択された記事の圧縮を行う。またアクセス可能性が小さいとして圧縮された記事であっても、アクセスが発生することは当然あり得る。そのような場合、解凍処理を行うことで、アクセスしたユーザに対して記事を適切に提供する。

【０００８】

上記した情報処理装置の前記圧縮解凍部においては、前記圧縮を行う場合に圧縮対象とされた記事に含まれる画像データの圧縮を行ってもよい。

画像データはテキストデータに対して容量が大きい場合が多く、テキストデータの圧縮率に対して画像データの圧縮率が低かったとしても圧縮効果、即ち圧縮した際に確保される空き容量が大きくなる可能性が高い。そこで、画像データを圧縮対象とする。

【０００９】

上記した情報処理装置の前記圧縮解凍部においては、前記圧縮を行う場合に圧縮対象とされた記事に含まれるデータのうち冒頭の所定のデータ以外のデータについて圧縮を行ってもよい。

これにより、閲覧者が記事の閲覧を行う際には、圧縮されていない冒頭のデータからユーザ端末に送信することが可能とされる。

【００１０】

上記した情報処理装置の前記圧縮解凍部においては、前記圧縮を行う場合に圧縮対象とされた記事に含まれる画像データを圧縮する際に非可逆圧縮を行ってもよい。

これにより、閲覧される可能性の低い記事を効率よく圧縮することが可能となる。

【００１１】

上記した情報処理装置の前記圧縮解凍部においては、前記圧縮対象とされた記事に含まれる画像データを可逆圧縮した後に当該記事が再度圧縮対象となった場合に当該記事に対して前記非可逆圧縮を行ってもよい。

これにより、閲覧される可能性が低い記事であっても、偶然そのような状態となっているに過ぎない記事もある。そこで、先ずは可逆圧縮を行う。

【００１２】

上記した情報処理装置においては、前記ブログに記事を投稿する投稿者の利用状況に応じて当該投稿者に許可する記憶容量を増減させるブログ管理部を備えていてもよい。

ブログの使用態様は様々であり、ブログの総データ量の増加が速い投稿者もいれば、遅い投稿者もいる。総データ量が増加してきたブログに対しては、記事の圧縮を行うことで記憶リソースの管理を行うが、それでも記憶リソースが確保しきれない場合には、投稿者のブログに許可された記憶容量の上限値を変更することで対応する。

【００１３】

上記した情報処理装置において、アクセス要求に応じて既に圧縮された記事の解凍が行われた場合、前記判定部は、解凍から所定期間、当該記事を圧縮対象と判定しなくてもよい。

つまりアクセス時に解凍した記事は、所定期間の間は解凍したままとする。

【００１４】

上記した情報処理装置の前記判定部においては、既に圧縮された記事について、同一のブログに属する他の記事に対するページビューの増加傾向を示す値に応じた解凍の是非を判定し、前記圧縮解凍部は前記解凍の是非に応じて前記既に圧縮された記事を解凍しておいてもよい。

或るブログが何らかの人気記事によってアクセス数が顕著に上昇したような場合、そのブログに含まれる別の記事は、これまでアクセスされていなかったとしても、今後アクセスされる可能性が高まる。そこで解凍するように判定する。

【００１５】

上記した情報処理装置の前記判定部においては、記事内容に基づいて、既に圧縮した記事を解凍するか否かを判定してもよい。

例えば圧縮した記事の内容として、或る設定したキーワードや時事語を含む記事、特定のテーマの記事などを抽出し、これらを解凍対象とする。

【００１６】

本発明に係る情報処理方法は、１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得ステップと、前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定ステップと、一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定ステップと、を行う。

この情報処理方法により、情報処理装置によって適切に圧縮する記事を判定することができる。

本発明に係るプログラムは、上記各ステップに相当する手順を情報処理装置に実行させるプログラムである。本発明に係る記憶媒体は、上記プログラムを記憶したものである。これらにより上述の情報処理装置の処理を実現する。

【発明の効果】

【００１７】

本発明によれば、圧縮する記事を適切に選択することができるため、閲覧時の解凍処理を可能な限り生じさせないようにしつつ、記憶リソースの有効な利用を図ることができる。

【図面の簡単な説明】

【００１８】

【図１】本発明の実施の形態のブログサーバを含むネットワークの説明図である。
【図２】実施の形態で利用できるコンピュータ装置のブロック図である。
【図３】実施の形態のブログサーバの機能構成の説明図である。
【図４】実施の形態の管理データベースの説明図である。
【図５】第１の実施の形態の圧縮判定処理のフローチャートである。
【図６】第１の実施の形態の圧縮処理のフローチャートである。
【図７】実施の形態のアクセス要求受信時の処理のフローチャートである。
【図８】実施の形態の解凍後の処理例のフローチャートである。
【図９】第２の実施の形態の圧縮判定処理及び圧縮処理のフローチャートである。
【図１０】第３の実施の形態の総データ量上限値設定処理のフローチャートである。
【図１１】第４の実施の形態のアクセス可能性指数算出処理のフローチャートである。
【図１２】第５の実施の形態の圧縮処理のフローチャートである。
【図１３】第６の実施の形態の圧縮判定処理のフローチャートである。
【図１４】第７の実施の形態の圧縮解凍判定処理のフローチャートである。
【図１５】第７の実施の形態の圧縮解凍処理のフローチャートである。
【図１６】第８の実施の形態の解凍判定処理のフローチャートである。
【図１７】第８の実施の形態の解凍処理のフローチャートである。
【図１８】第９の実施の形態の圧縮判定処理のフローチャートである。

【発明を実施するための形態】

【００１９】

以下、実施の形態を次の順序で説明する。

＜１．システム構成＞
＜２．ブログサーバ及びデータベース＞
＜３．第１の実施の形態＞
＜４．第２の実施の形態＞
＜５．第３の実施の形態＞
＜６．第４の実施の形態＞

＜７．第５の実施の形態＞
＜８．第６の実施の形態＞
＜９．第７の実施の形態＞
＜１０．第８の実施の形態＞
＜１１．第９の実施の形態＞
＜１２．まとめ及び変形例＞
＜１３．プログラム及び記憶媒体＞

【００２０】
なお、以下の説明において、“ブログ”とは、ウェブログあるいは単にブログと呼ばれる日記形式のウェブページのことである。より具体的には、ブログサーバはユーザに対してブログを形成する環境（記憶容量やウェブページ）を提供し、ユーザは投稿等の形式で文章や画像による記事を自分のブログにアップロードする。ブログサーバは通常、当該記事を一般（または限定した範囲）の閲覧に供する。但し非公開のものとしてもよい。
記事の内容については特に限定されない。ユーザが情報発信に用いる内容でも、個人的な日記等の内容でもよい。また「ブログ」と呼ばれていないものであっても、同等のものを含む。

【００２１】
“記事”とは、ブログを構成する要素であって、文章や画像によって構成された一つの単位（例えば投稿単位）のことを指す。内容には関わらない。なお、必ずしも一つの話題としての記事ではなく、１または複数の話題について一つのＵＲＬで閲覧される記事群を指すものと考えてもよい。

【００２２】
“ユーザ”とは、自己のブログに記事を書き込む記述者としてのユーザ（いわゆるブロガー）と、他人または自己のブログを閲覧する閲覧者としてのユーザが想定される。これらを区別して「記述者」「閲覧者」と表記する。もちろん一人のユーザがある時点では記述者となりある時点では閲覧者となることが通常に想定される。

【００２３】
“圧縮”とは、いわゆるデータ圧縮のことであり、テキストデータや画像データ等の各種データを、そのデータの実質的な性質を保ったまま、データ量を減らした別のデータに変換することである。
“解凍”とは、圧縮されたデータを圧縮前の状態に戻すことである。但し、圧縮時にいわゆる非可逆圧縮が行われた場合など、データが完全に圧縮前の状態に戻らない場合も含む。本明細書では、少なくとも記事の内容の閲覧等が可能な状態とすることを“解凍”ということとする。

【００２４】
＜１．システム構成＞
図１に実施の形態のブログサーバ１を含むネットワークシステムの構成例を示す。
本実施の形態に係るネットワークシステムは、ブログサーバ１と複数のユーザ端末５がネットワーク２により相互に通信可能に接続されている。
またブログサーバ１は各種データベースにアクセス可能とされている。なお、以下「データベース」については「ＤＢ」と表記する。図ではブログサーバ１がアクセス可能なＤＢとしてブログＤＢ５１、画像ＤＢ５２、管理ＤＢ５３を例示している。

【００２５】
ネットワーク２の構成は多様な例が想定される。例えば、インターネット、イントラネット、エキストラネット、ＬＡＮ（Local Area Network）、ＣＡＴＶ（Community Antenna TeleVision）通信網、仮想専用網（Virtual Private Network）、電話回線網、移動体通信網、衛星通信網等が想定される。
またネットワーク２の全部または一部を構成する伝送媒体についても多様な例が想定される。例えばＩＥＥＥ（Institute of Electrical and Electronics Engineers）１３９４、ＵＳＢ（Universal Serial Bus）、電力線搬送、電話線等の有線でも、ＩｒＤＡ（In

frared Data Association）のような赤外線、ブルートゥース（登録商標）、８０２．１１無線、携帯電話網、衛星回線、地上波デジタル網等の無線でも利用可能である。

【００２６】

ブログサーバ１は、ユーザに対するブログサービスの管理運営を行う組織によって用いられる情報処理装置である。ブログサーバ１は、ユーザ（記述者）へのブログ環境の提供やユーザ（閲覧者）のアクセス要求に応じたブログ記事ページ等のウェブページデータの配信を行う。

具体的にはブログを開設したい記述者に対しては、その記述者のブログとしてのウェブページの設定やユーザ情報の登録などを行う。既にブログを開設済みの記述者に対しては、記述者が投稿する記事の保存を行う。

また一般の閲覧者となるユーザからのアクセス要求に応じて、該当のウェブページに係るウェブページデータを配信する。

このブログサーバ１が、本発明請求項の情報処理装置の実施の形態に相当する。

【００２７】

ユーザ端末５は、記述者や閲覧者としてのユーザが使用する端末である。このユーザ端末５は、例えば、通信機能を備えたＰＣ（Personal Computer）やフィーチャーフォンやＰＤＡ（Personal Digital Assistant）、或いは、スマートフォンやタブレット端末などのスマートデバイスなどが想定される。

ユーザ端末５では、必要に応じて各種の送受信処理や表示処理などが実行される。

閲覧者は、ユーザ端末５においてウェブブラウザを介して、関心のあるブログの閲覧を任意に行うことができる。

記述者は、ユーザ端末５により自分のブログページにアクセスし、閲覧したり、新規に記事を投稿したりすることができる。

ユーザ端末５ではこれらの動作のための通信処理や表示処理等を行うことになる。

【００２８】

図１に示したブログサーバ１やユーザ端末５を構成する情報処理装置のハードウエア構成を図２に示す。ブログサーバ１やユーザ端末５として示した各装置は、情報処理および情報通信が可能な図２に示すようなコンピュータ装置によって実現される。

【００２９】

図２において、コンピュータ装置のＣＰＵ（Central Processing Unit）１０１は、ＲＯＭ(Read Only Memory)１０２に記憶されているプログラム、または記憶部１０８からＲＡＭ(Random Access Memory)１０３にロードされたプログラムに従って各種の処理を実行する。ＲＡＭ１０３にはまた、ＣＰＵ１０１が各種の処理を実行する上において必要なデータなども適宜記憶される。

ＣＰＵ１０１、ＲＯＭ１０２、およびＲＡＭ１０３は、バス１０４を介して相互に接続されている。このバス１０４には、入出力インタフェース１０５も接続されている。

入出力インタフェース１０５には、入力装置１０６、出力装置１０７、記憶部１０８、通信部１０９が接続されている。

入力装置１０６はキーボード、マウス、タッチパネルなどにより構成される。

出力装置１０７はＬＣＤ（Liquid Crystal Display）、ＣＲＴ（Cathode Ray Tube）、有機ＥＬ（Electroluminescence）パネルなどよりなるディスプレイ、並びにスピーカなどにより構成される。

記憶部１０８はＨＤＤ（Hard Disk Drive）やフラッシュメモリ装置などにより構成される。

通信部１０９はネットワーク２を介しての通信処理や機器間通信を行う。

入出力インタフェース１０５にはまた、必要に応じてメディアドライブ１１０が接続され、磁気ディスク、光ディスク、光磁気ディスク、或いは半導体メモリなどのリムーバブルメディア１１１が適宜装着され、リムーバブルメディア１１１に対する情報の書込や読出が行われる。

【００３０】

このようなコンピュータ装置では、通信部１０９による通信によりデータやプログラムのアップロード、ダウンロードが行われる。またリムーバブルメディア１１１を介したデータやプログラムの受け渡しが可能である。

ＣＰＵ１０１が各種のプログラムに基づいて処理動作を行うことで、ブログサーバ１やユーザ端末５としての必要な情報処理や通信が実行される。

なお、ブログサーバ１やユーザ端末５を構成する情報処理装置は、図２のようなコンピュータ装置が単一で構成されることに限らず、複数のコンピュータ装置がシステム化されて構成されてもよい。複数のコンピュータ装置は、ＬＡＮ等によりシステム化されていてもよいし、インターネット等を利用したＶＰＮ等により遠隔地に配置されたものでもよい。複数の情報処理装置には、クラウドコンピューティングサービスによって利用可能なサーバ群（クラウド）としての情報処理装置が含まれてもよい。

【００３１】

＜２．ブログサーバ及びデータベース＞

１または複数の情報処理装置で構成されるブログサーバ１としての機能構成および各種のＤＢを図３に示す。

ブログサーバ１としての各機能は、情報処理装置においてＣＰＵ１０１でプログラムに応じて実行される処理により実現される機能である。但し以下説明する全部または一部の各構成の処理をハードウエアにより実現してもよい。

また各機能をソフトウエアで実現する場合に、各機能がそれぞれ独立したプログラムで実現される必要はない。一つのプログラムにより複数の機能の処理が実行されてもよいし、一つの機能が複数のプログラムモジュールの連携で実現されてもよい。

また各機能は複数の情報処理装置に分散されていてもよい。更に機能の一つが、複数の情報処理装置によって実現されてもよい。

【００３２】

ブログサーバ１は、図示するようにブログ管理部１１、傾向取得部１２、閾値設定部１３、判定部１４、圧縮解凍部１５としての機能を備える。

【００３３】

ブログ管理部１１は、ブログサービスを提供するサーバとして必要な処理を実行する。例えばユーザへのブログ環境の提供、記述者としてのユーザの情報の管理、作成されたブログの記憶管理、各ブログに関する情報管理、アクセス要求に応じたブログ（記事）のウェブページの配信などを行う。

ブログ管理部１１は、ブログごとに許可する総データ量の上限値（以降、総データ量上限値）を設定する。総データ量上限値は、例えば２ＧＢ（Giga Byte）のように全てのブログで一律同じデータ量であってもよいし、人気のあるブログに多くのデータ容量を許可するためにブログごとに異なるデータ量であってもよい。

またブログ管理部１１は管理ＤＢ５３の情報の更新や読み出しを逐次行う。

【００３４】

傾向取得部１２は、ブログの総データ量の増加傾向や減少傾向を示す値を取得する。値の取得は、既に算出された値をＤＢなどから取得してもよいし、複数の値から傾向を示す値を算出することによって取得してもよい。なお、傾向を示す値は数値である必要はなく、「高（増加傾向が高い）／中（増加傾向が中程度）／低（増加傾向が低い）」などのように複数の段階を示すものであってもよい。以降の説明においては、ブログの総データ量の増加傾向や減少傾向を示す値のことをまとめて「増加傾向指数」と記載する。

【００３５】

増加傾向指数は、ブログごとに求められる指標であり、当該ブログに含まれる全ての記事の総データ量から導出されることが望ましいが、そうでなくてもよい。例えば、投稿記事数の増加傾向からブログ全体の総データ量を推量することにより増加傾向指数を求めてもよい。

【００３６】

傾向取得部１２は、他にも、記事に対するアクセス可能性を示す値（以降、アクセス可

能性指数）を取得する。
記事に対するアクセス可能性指数は、例えば、当該記事についてのページビュー数、アクセスしたユニークユーザ数、総被リンク数、記事のランクを示す値、記事に対するアクセス要求が無い期間の長さ、所定期間における記事についてのページビュー数、記事についてのページビューの増加傾向を示す値、記事ページ上に掲載される広告に対するクリック数などの中から、アクセス可能性指数として採用する情報などを用いて取得する。具体的には、これらの値のうちの一つをアクセス可能性指数として選択して取得してもよいし、複数の情報から算出したアクセス可能性指数を取得してもよい。
【００３７】
更に傾向取得部１２は、ブログについての人気傾向を示す値（人気度指数）を取得する。ブログの人気度指数は、ブログ全体についての総ページビュー数、記事ごとのページビュー数、ブログにアクセスしたユニークユーザ数、ブログに設定された総被リンク数、ブログに投稿された総コメント数、ブログにコメントを投稿したユニークユーザ数、ブログのページランクを示す値、ブログに対するアクセス要求が無い期間の長さ、所定期間におけるブログ全体についての総ページビュー数、ブログ全体についてのページビューの増加傾向を示す値、ブログ更新頻度（記事投稿頻度）を示す値、ブログページ上に掲載される広告に対するクリック数、ブログ全体についてのデータ量の増加傾向を示す値などのうちから、１または複数の値に基づいて算出される。
【００３８】
閾値設定部１３は、ブログのデータ容量の圧縮をするか否かを判定するための閾値（以降、ブログ圧縮判定閾値）をブログごとに設定する処理を行う。
閾値の設定には傾向取得部１２が取得した増加傾向指数を用いる。増加傾向指数が高い（即ちブログの総データ量の増加速度が速い）ブログほど閾値を低く設定することで、総データ量上限値を超えにくくし、延いては新たな記事投稿が制限されてしまうことを防止する。
【００３９】
判定部１４は、ブログ圧縮判定閾値に基づきブログに含まれる記事を圧縮するか否かを判定する処理を行う。換言すれば、当該ブログを圧縮対象ブログとするか否かを判定する処理を行う。判定部１４は、圧縮対象ブログと判定した場合には、各記事について何れの記事を圧縮するかについての判定も行う。
また判定部１４は、記事の内容に応じて解凍するか否かを判定する処理や、記事のアクセス可能性指数に応じて記事を解凍するか否かを判定する処理なども行う。
具体例は各実施の形態の処理として後述する。
【００４０】
圧縮解凍部１５は、判定部１４によって圧縮すると判定された記事を圧縮する処理を行う。
また圧縮解凍部１５は、既に圧縮された記事に対してのアクセス要求が発生した場合に、圧縮に対する解凍処理を行う。
更に圧縮解凍部１５は、判定部１４によって解凍すると判定された圧縮済みの記事を解凍する処理を行う。
なお、記事の圧縮処理を行った結果、それ以上の記事の圧縮処理が不要となったブログは、圧縮対象ブログではなくなる。
【００４１】
図３にはブログサーバ１がアクセスするＤＢとしてブログＤＢ５１、画像ＤＢ５２、管理ＤＢ５３を示している。
ブログＤＢ５１は、各記述者についてのブログデータをウェブページデータとして保存するＤＢである。各ブログについては、記述者の投稿に応じて記事が追加されていく。
ブログを形成するウェブページのデータは、例えば、ＨＴＭＬ（HyperText Markup Language）やＸＨＴＭＬ（Extensible HyperText Markup Language）などの構造化文書ファイルである。構造化文書ファイルには、記述者が投稿した記事のテキストデータや各種画

像等の画像データの指定情報と、それらの配置や表示態様（文字色やフォントや大きさや装飾など）が記述されている。
またブログに対しては閲覧者がコメントを投稿することもできる。そのような閲覧者からのコメントデータもブログやブログ内の個々の記事に紐づけてブログＤＢ５１に保存される。
ブログサーバ１は、ユーザ端末５から或るブログについてのアクセス要求があった場合に、要求されたブログページをブログＤＢ５１から読み出してユーザ端末５に配信することになる。
【００４２】
画像ＤＢ５２は、ブログに添付された画像データ（静止画データや動画データ）を保存するＤＢである。
ブログ内の記事には、画像を添付することができるが、例えばブログＤＢ５１には記事データ及び記事データに対応した画像の指定情報（リンク情報）が記憶される。そして画像データ自体は画像ＤＢ５２に保存される。
画像が添付されたブログ記事へのアクセス要求の場合、そのウェブページデータがユーザ端末５においてブラウザにより表示されるが、その際、ユーザ端末５はウェブページ上のリンク設定により画像データをブログサーバ１に要求する。ブログサーバ１は当該要求に応じて画像データを画像ＤＢ５２から読み出し、ユーザ端末５に配信する。これによりユーザ端末５上で、画像付きのブログ記事が表示される。
なおこれは一例であり、予め画像データを含むウェブページデータをブログＤＢ５１に格納するようにしてもよい。
【００４３】
管理ＤＢ５３は、各ブログを管理するための情報を格納するＤＢである。
管理ＤＢ５３の内容の一例を図４に示す。
一つのブログについてはブログＩＤ（Identification）が設定され、ブログＩＤにより付随する情報が管理される。例えばブログ（ブログＩＤ）ごとに、ユーザ情報、ブログ管理情報、ブログ実績情報、サイズ情報、判定情報、圧縮解凍情報、圧縮記事タグなどが、逐次更新されながら管理される。
【００４４】
ユーザ情報は、ブログを開設した記述者としてのユーザ（ブロク管理者）の情報である。例えばユーザ情報としては、ユーザＩＤ、管理者としてのログインパスワード、ユーザの住所、氏名、年齢等の属性情報、管理者としてのログイン日時、などの情報が含まれる。
【００４５】
ブログ管理情報は、ブログ自体の属性情報である。例えばブログのＵＲＬ（Uniform Resource Locator）、ブログのジャンル情報、ブログの開設日時、ブログに含まれる記事数、更新日時情報、ブログのレイアウト情報、リンク設定情報等が含まれる。
【００４６】
ブログ実績情報としては、ブログの増加傾向指数となる情報や、ブログの人気度指数となる情報や、各記事のアクセス可能性の指標となる情報が記憶される。
具体的には、ブログ全体について、総ページビュー数、アクセスしたユニークユーザ数、総被リンク数、ブログのページランクを示す値、ブログに対するアクセス要求が無い期間の長さ、所定期間におけるブログ全体についての総ページビュー数、ブログ全体についてのページビューの増加傾向を示す値、ブログ更新頻度を示す値、ブログページ上に掲載される広告に対するクリック数などが記憶され、逐次更新される。
これらの値はブログごとの人気度に応じた値となるため、人気度指数算出の際に用いる値として適している。なお、算出された人気度指数もブログ実績情報として記憶される。
算出したブログの増加傾向指数についてもブログ実績情報として記憶される。
【００４７】
また、ブログ実績情報として、記事ごとに、ページビュー数、アクセスしたユニークユ

ーザ数、総被リンク数、記事のランクを示す値、記事に対するアクセス要求が無い期間の長さ、所定期間における記事についてのページビュー数、記事についてのページビューの増加傾向を示す値、記事ページ上に掲載される広告に対するクリック数などが記憶され、逐次更新される。

これらの値は、記事ごとのアクセス可能性に応じた値となるため、アクセス可能性の度合い（以降、アクセス可能性指数）を算出する際に用いる値として適している。なお、算出されたアクセス可能性指数もブログ実績情報として記憶される。

【００４８】

サイズ情報は、そのブログ全体に含まれる記事の総データ量の情報である。また各記事のサイズ情報を記憶してもよい。サイズ情報は、ブログの更新に応じて更新される。

なお、サイズ情報としてブログＤＢ５１に記憶したデータ量と画像ＤＢ５２に記憶した画像データのデータ量を合わせて管理してもよいし、それぞれ別個に管理してもよい。

【００４９】

判定情報は、そのブログを圧縮対象ブログとするか否かの判定情報や、各記事について判定部１４が増加傾向指数や人気度指数やその他の情報に基づいて圧縮するか否かを判定した情報である。また圧縮した記事について解凍するか否かを判定部１４が判定した情報も含む。即ち圧縮可否や解凍可否を示す情報である。これらは例えばフラグデータとして更新される。

圧縮可否情報として圧縮可を示す情報が記憶された記事は、圧縮処理の対象とされた記事であることを示す。そして、圧縮処理の後には当該フラグデータがクリアされ、圧縮否（即ち圧縮処理の対象とはしないことを示す状態）を示す情報が上書きされる。

他にも、判定情報としてブログ圧縮判定閾値が記憶される。

【００５０】

圧縮解凍情報は、ブログ内の各記事について、オリジナル状態／圧縮状態／圧縮から解凍した状態など、その時点でどのような状態のデータが記憶されているかを示す情報である。圧縮解凍情報は、これらを識別するステータス情報として構成されればよい。

また圧縮や解凍の履歴情報として、圧縮や解凍の実行日時も記憶される。

以降の記載においては、オリジナル状態の記事を「未圧縮記事」、圧縮状態の記事を「圧縮記事」、圧縮から解凍した状態の記事を「解凍記事」と記載する。

なお、解凍記事のデータは、未圧縮記事のデータと同一である場合もあるが、非可逆圧縮を用いているために元の記事データよりも低品質となる場合もある。つまり必ずしも未圧縮記事と解凍記事のデータは同一ではない。そこで圧縮後に解凍した記事データについては「未圧縮記事」と区別するために「解凍記事」と表記する。

【００５１】

圧縮記事タグは、各記事について内容に応じて設定されたタグである。

例えば記事に出現するキーワード的な語句、時事的な語句、記事のジャンルなどがタグとして設定され、登録される。例えば圧縮を行う際に記事内容に応じたタグが作成されて圧縮記事タグとして登録される。圧縮記事タグは、解凍せずに圧縮記事の内容を推定するために利用される。

【００５２】

以上の各ＤＢ（ブログＤＢ５１、画像ＤＢ５２、管理ＤＢ５３）は、ブログサーバ１がアクセス可能とされていればどのような形態で実現されていてもよい。例えばブログサーバ１と同一システム内の記憶部に各ＤＢのすべてが形成されていてもよいし、各ＤＢの一部または全部が別体、遠隔地等のコンピュータシステムに設けられていてもよい。もちろん各ＤＢが一つの装置（例えば一つのＨＤＤ等）内に形成されている必要はない。また各ＤＢのそれぞれが、それぞれ一つのＤＢとして構成される必要もない。例えば管理ＤＢ５３として記憶される情報が、複数のＤＢ（例えばブログに関するユーザ管理用のＤＢとブログ管理用のＤＢなど）により記憶管理されてもよい。以上の各ＤＢは、実施の形態の処理に関連する情報の記憶部を、それぞれ一つのＤＢの形態で例示したものに過ぎない。

【００５３】

＜３．第１の実施の形態＞
ブログサーバ１が実行する第１の実施の形態としての処理例を説明していく。
現在、一般ユーザにとってブログは容易に始められるが、少しの記事をアップロードした後、飽きてしまうユーザもいれば、長く続けるユーザもいる。また、アクセス要求の多い人気ブログもあれば、ほとんど閲覧者がいないブログもある。
ブログサーバ１としては、これら多様なユーザに対して、分け隔て無くブログを維持する必要があるが、そのために記憶リソースの負担が大きくなりがちである。従って、ユーザごとに総データ量上限値を設け、許可された記憶リソースの中でブログの投稿の受け付けや管理を行う。また、総データ量上限値に近付いてきたブログに対しては、適切な記事の圧縮を行うことにより、記憶リソースの確保に努めるが、圧縮をするか否かをユーザごとに適切に判定することが重要である。
そこで本実施の形態において、ブログサーバ１は、ブログごとの増加傾向指数に基づいて、圧縮するブログ及び記事を決定する。また、圧縮記事に対するアクセス要求があった場合には、解凍して配信する。
【００５４】
但し、圧縮処理・解凍処理もある程度の処理負担がかかるため、あまり頻繁に行いたくない。また、圧縮した記事を解凍して配信することは、処理負担とともに応答時間も増えることになり、ユーザにパフォーマンス低下を感じさせる可能性もあるため、なるべく圧縮した記事へのアクセス要求が発生しないようにしたい。
そこで本実施の形態では、圧縮対象ブログに属する各記事の中でも、アクセス可能性が小さい記事をより的確に選択して圧縮対象記事とする。
【００５５】
図５はブログサーバ１が実行する圧縮判定処理の例を示している。圧縮判定処理では、圧縮対象ブログを判定するとともに、当該圧縮対象ブログに含まれる記事の中から圧縮対象記事を判定する。
なお、この例では、ブログ実績情報として記憶された増加傾向指数を取得している。増加傾向指数の算出例については後述する。また、記事ごとのアクセス可能性指数については、ブログ実績情報の中の一つの値をそのままアクセス可能性指数として用いる。
図５以降、後述する図１８までのフローチャートで示す各処理は、ブログサーバ１が図３に示したブログ管理部１１、傾向取得部１２、閾値設定部１３、判定部１４、圧縮解凍部１５としての機能により実行する処理である。
【００５６】
ブログサーバ１は図５の圧縮判定処理を、逐次、ブログＤＢ５１に保存している全部または一部のブログに対して実行する。これは、今回圧縮対象とするブログを圧縮対象ブログとして判定するとともに、当該圧縮対象ブログに含まれる記事のうち何れの記事を圧縮対象記事とするかを判定する処理である。
先ず、ステップＳ１０１でブログサーバ１は、圧縮判定処理の対象とする一つのブログを特定する。例えばブログＩＤ順に従って一つのブログを選択するものとすればよい。
【００５７】
ステップＳ１０２でブログサーバ１は、処理対象として特定したブログについての増加傾向指数を取得する。本例では、管理ＤＢ５３にブログ実績情報として記憶された増加傾向指数を取得する。
なお、増加傾向指数は他のブログに対して記憶容量がどの程度の速度で増加していくかを相対的に示すものであるため、ブログごとに異なる方法によって算出された値を取得することは好ましくない。即ち、１日あたりに増加する記憶容量の平均値を算出して増加傾向指数とする場合は、何れのブログも同様に算出した増加傾向指数を取得することが望ましい。
【００５８】
次に、ステップＳ１０３でブログサーバ１は、増加傾向指数に応じたブログ圧縮判定閾値を決定する処理を実行する。

例えば、一人あたりの記述者に割り当てられているブログの総データ量が２ＧＢである場合に、ブログ圧縮判定閾値は、２ＧＢよりも少ない値とされる。一例として、増加傾向指数の高／中／低に基づいて１．９ＧＢ／１．９５ＧＢ／１．９８ＧＢのように決定する。

増加傾向指数を高／中／低に分類する一例を説明する。

例えば、増加傾向指数として一日あたりに増加する記憶容量の平均値（以降、増加容量平均値）を採用し、

・増加容量平均値が０～５０ＫＢ（Kilo Byte）＝増加傾向指数（増加速度）が低い（遅い）

・増加容量平均値が５０ＫＢ～３００ＫＢ＝増加傾向指数が中くらい

・増加容量平均値量が３００ＫＢ以上＝増加傾向指数（増加速度）が高い（速い）

とする。このとき、増加容量平均値が１００ＫＢのブログは、増加傾向指数が中であるとして、ブログ圧縮判定閾値が１．９５ＧＢに設定される。

なお、増加傾向指数によってブログ圧縮判定閾値を三段階に分けるのはあくまで一例であり、２段階に分けてもよく、４段階以上に分けてもよい。

【００５９】

ブログ圧縮判定閾値を４段階以上に分類する一例として、数式を用いる例を説明する。

例えば、全てのブログの中で増加容量平均値の最大値が４００ＫＢであり、今回の処理対象とされたブログの増加容量平均値が２００ＫＢである場合に、ブログ圧縮判定閾値＝１．９８ＧＢＧＢ－（２００ＫＢ／４００ＫＢ×０．０８ＧＢ）＝１．９４ＧＢなどのように算出してもよい。この数式によれば、増加容量平均値＝４００ＫＢのブログのブログ圧縮判定閾値は１．９ＧＢとなり、増加容量平均値＝０ＫＢのブログのブログ圧縮判定閾値は１．９８ＧＢとなる。即ち、ブログ圧縮判定閾値は、増加傾向指数に応じて細かく設定される。なお、この場合には、増加傾向指数を高／中／低の３段階などに分ける必要は無い。

他にも、ブログ圧縮判定閾値は、全てのブログの増加傾向指数の分布を加味して算出してもよいし、処理対象とされたブログの増加傾向指数のみを用いて算出してもよい。

【００６０】

続いて、ステップＳ１０４でブログサーバ１は、処理対象のブログにおける総データ量がブログ圧縮判定閾値を超えているか否かを判定する処理を実行する。

ブログにおける総データ量がブログ圧縮判定閾値を超えていない場合、ブログサーバ１は後述するステップＳ１１１の処理へと遷移する。

一方、ブログにおける総データ量がブログ圧縮判定閾値を超えている場合、ブログサーバ１は、処理対象のブログを圧縮対象ブログと判定し、当該判定結果を管理ＤＢ５３の判定情報として記憶した後、ブログ内の各記事について圧縮可否を判定するためにステップＳ１０５～Ｓ１１０の各処理を行う。

【００６１】

ステップＳ１０５でブログサーバ１は、ブログ内の一つの記事を選択する。

ステップＳ１０６でブログサーバ１は、選択した記事が既に圧縮済であるか否かを確認する。これは例えば管理ＤＢ５３における圧縮解凍情報を参照すればよい。

もし圧縮済であれば、当該記事の圧縮可否判定は不要なためステップＳ１１１に進んだ後、ステップＳ１１０で全ての記事について処理を終えたか否かを確認し、終えていなければステップＳ１０５に戻って次の記事を選択する。

なお、ステップＳ１１１の全記事とは、今回処理対象とする記事の全てという意味である。ブログ内の全ての記事という場合もあるし、ブログ内の一部（例えば特定の期間に投稿された記事など）でもよい。

【００６２】

ステップＳ１０５で処理対象に選択した記事が圧縮されていない状態の記事である場合、ブログサーバ１はステップＳ１０６からＳ１０７に進み、当該記事についてのアクセス可能性指数としてのブログ実績情報を取得する。本例におけるアクセス可能性指数は、管

理ＤＢ５３にブログ実績情報として記憶された情報からアクセス可能性指数として採用する情報を一つ取得する処理である。具体的には、管理ＤＢ５３に記憶された当該記事についてのページビュー数、アクセスしたユニークユーザ数、総被リンク数、記事のランクを示す値、記事に対するアクセス要求が無い期間の長さ、所定期間における記事についてのページビュー数、記事についてのページビューの増加傾向を示す値、記事ページ上に掲載される広告に対するクリック数などの中から、アクセス可能性指数として採用する情報を一つ選択して取得する。

なお、アクセス可能性指数は、記事ごとに同じ指標を取得することが望ましい。

【００６３】

続いて、ブログサーバ１はステップＳ１０８で、当該記事についての圧縮可否判定をアクセス可能性指数に基づいて行う。即ち、アクセス可能性が大きければ圧縮すべきでないと判定し、アクセス可能性が小さければ圧縮してもよい記事（或いは圧縮すべき記事）と判定する。

例えば、アクセス可能性指数として当該記事に対するアクセス要求が無い期間の長さを選択した場合、当該期間長が３年以上であるか未満であるかを判定し、３年以上アクセス要求が無い記事はアクセス可能性が低いとして圧縮対象記事と判定する。なお、３年という数値（判定のための閾値）はあくまで一例である。この数値は記述者ごとに変えてもよいし、一律同じ値であってもよい。記述者ごとに変える場合は、処理対象のブログに属する各記事のアクセス可能性指数の分布を鑑みて決定することが望ましい。即ち、アクセス可能性指数が低い記事が多いブログであれば、上記「３年」という判定のための閾値を「５年」のように長くすることが考えられるし、アクセス可能性指数が高い記事が多いブログであれば、上記「３年」という判定のための閾値を「１年」のように短くすることが考えられる。

【００６４】

このように、まずステップＳ１０３及びＳ１０４においてブログの増加傾向指数を考慮して圧縮対象ブログを判定した後、ステップＳ１０７及びＳ１０８においてアクセス可能性指数を考慮して圧縮可否を判定する。

これによってブログの記憶リソースの増加傾向を考慮したうえで、あまりアクセスされていない記事の圧縮可否を判定できることになる。

例えば単純に、「記事に対して１度もアクセスがない期間が３年以上続いていること」のように、記事に対して１度もアクセスがない期間を指標値として、指標値が一定の条件を満たす記事について圧縮することも考えられるが、この場合、必ずしもアクセス可能性が適切に判断されているとは言いがたい面がある。

例えば、増加記憶容量が大きなブログと同じように増加記憶容量の小さなブログを圧縮対象としてしまうと、まだ圧縮する必要性が小さいにも関わらず記事の圧縮を行うこととなる。すると、アクセス要求のあった記事が圧縮されてしまっている可能性が高まることとなり、圧縮及び解凍の機会が増えることとなる。従って、なるべく圧縮や解凍の機会を少なくしたいという観点からは、増加記憶容量によらずに圧縮対象ブログとしてしまうことは必ずしも妥当ではない。

そこでステップＳ１０４でブログサーバ１は、ブログの増加傾向指数を反映させたブログ圧縮判定閾値を用いることにより、ブログが占有する記憶容量の増加傾向が小さなブログが圧縮対象ブログとなり難いようにしている。

【００６５】

続くステップＳ１０９ではブログサーバ１は、当該記事についての圧縮可否判定の結果を判定情報として記憶する。例えば、当該記事についての判定情報として管理ＤＢ５３に記憶された圧縮可／否のフラグを更新または維持する。

【００６６】

以上で一つの記事について圧縮可否判定を終えたら、ブログサーバ１はステップＳ１１０で、現在の処理対象のブログについて今回処理対象としている記事の全てについて判定を終えたか否かを確認し、終えていなければステップＳ１０５に戻って次の記事を選択す

る。そしてステップＳ１０６～Ｓ１０９を実行する。

【００６７】

或るブログについて今回対象の全ての記事について圧縮可否判定を終えたら、ステップＳ１１１で、続いて他のブログについても処理を実行するかを確認する。他のブログについても同様の処理を行う場合は、ステップＳ１０１に戻って、他の一つのブログを処理対象として特定し、以下同様の処理を行う。

例えば今回処理対象とする全てのブログについて処理を終えていた場合は、ステップＳ１１１から図５の圧縮判定処理を終える。

【００６８】

ブログサーバ１は、この図５のような圧縮判定処理を逐次実行する。例えば定期的に、ブログＤＢ５１に保存されている全てのブログについて実行することが考えられる。これにより、各ブログについて、各記事が、圧縮してよいものか否かが判定され、その判定情報が管理ＤＢ５３に記憶される。

なお、各ブログについて、図５のような圧縮判定処理をある時点で１回行うだけで無く、期間をおいてくり返し実行することが望ましい。時期ごとにブログの増加傾向指数は変動することが想定されるからである。

【００６９】

ブログサーバ１は以上のような圧縮判定処理を適宜行うとともに、図６に例示する圧縮処理を適宜行う。例えば全部または一部のブログを対象として定期的に圧縮処理を行う。

図６の圧縮処理としてブログサーバ１は、まずステップＳ２０１で処理対象のブログを一つ特定する。

ステップＳ２０２でブログサーバ１は、処理対象と特定したブログについての判定情報を取得する。即ち管理ＤＢ５３において当該ブログのブログＩＤに対応して記憶されている判定情報である。具体的には例えば、図５の圧縮判定処理において判定した圧縮対象ブログであるか否かを示す情報や、記事ごとの圧縮可否のフラグ情報を確認する処理となる。

判定情報により、当該ブログが圧縮対象ブログであるか否か、及び、ブログにおける各記事について圧縮対象記事とされているか否かを確認できる。

そこでステップＳ２０３でブログサーバ１は、処理対象のブログが圧縮対象ブログであるか否かを判定する。

【００７０】

もし処理対象のブログが圧縮対象ブログでなければ、ブログサーバ１はステップＳ２０３からＳ２１０に進み、当該ブログについての圧縮処理を終了する。そして続いて他のブログについても圧縮処理を実行するかを確認する。他のブログについても圧縮処理を行う場合は、ステップＳ２０１に戻って、他の一つのブログを処理対象として特定する。

【００７１】

圧縮対象とされた１以上の記事が存在する場合は、ブログサーバ１はステップＳ２０３から２０４に進み、圧縮対象ブログに属する各記事から、圧縮対象記事を特定する。複数の記事が圧縮対象記事とされている場合は、全ての圧縮対象記事を特定する。

【００７２】

続いて、ブログサーバ１はステップＳ２０５において、記事圧縮を行う。即ち、ステップＳ２０４で特定した１または複数の記事のデータ圧縮を行う。そして圧縮データ化された記事を、当該ブログに対応づけてブログＤＢ５１や画像ＤＢ５２に記憶する。

【００７３】

このステップＳ２０５でどのような圧縮を行うかは多様に考えられる。

まず圧縮対象部分の設定、即ち記事データにおけるどの部分を圧縮するかという種別として、

・記事におけるテキストデータと画像データの両方を圧縮する
・記事におけるテキストデータの全部を圧縮する
・記事におけるテキストデータの一部を圧縮する

・記事に含まれる画像データの全部を圧縮する
・記事に含まれる画像データの一部を圧縮する
ということが想定される
【００７４】
記事のテキストデータと画像データの両方を圧縮することによれば、圧縮効果を高くすることができ、必要記憶容量の削減効果を高くすることができる。
記事におけるテキストデータの全部を圧縮することによれば、テキストデータ量や圧縮率にもよるが、圧縮効果（容量削減効果）を高くすることができる。特に記事内容としてテキストデータが中心であるブログで有効である。
テキストデータの一部を圧縮することによれば、もし圧縮後にアクセスが生じたときの配信対応を迅速化することも可能である。例えばブログの冒頭の所定部分（閲覧時のファーストビューに現れない部分）を圧縮しておく。圧縮部分は後述のように解凍して配信することが想定されるが、ファーストビュー部分は圧縮しないことで、ユーザ端末５に対して迅速に（解凍処理を経ずに）配信できる。そしてユーザ端末５でファーストビュー表示を行っている間に後続部分の解凍を行って配信すれば、閲覧者にとって応答の遅れが生じていないように感じさせることができる。
またテキストデータのみの圧縮を行うことは、画像データ圧縮を行う場合に比較して処理負担が小さく、処理時間も短いという利点も得られる。
【００７５】
記事における画像データの全部を圧縮することによれば、データ量の大きい部分であるため、圧縮効果（容量削減効果）を高くすることができる。画像データの圧縮は、画像の解像度を低下させる圧縮とすれば、容量削減効果は特に高い。複数の画像データが存在する場合、全部の画像データに限らず、一部の画像データを圧縮するものでもよい。
記事における一部の画像データを圧縮する場合、閲覧時のファーストビューに現れない画像を選んで圧縮するとよい。その場合、もし圧縮後にアクセスが生じたときには、まず解凍不要な画像データを配信すればよいため、閲覧者にとって応答の遅れが生じていないように感じさせることができる。そしてユーザ端末５でファーストビュー表示を行っている間に後続の画像データを解凍して配信すればよい。
【００７６】
以上のような記事内における圧縮対象部分の設定は、固定的でもよいし、状況に応じて変更できるようにしてもよい。例えばブログＤＢ５１や画像ＤＢ５２の記憶リソース状況などに応じて自動的に選択できるようにしてもよい。
例えばブログＤＢ５１の記録可能なリソースが所定量以下になったら、テキストデータ圧縮を選択し、画像ＤＢ５２の記録可能なリソースが所定量以下になったら画像データ圧縮を選択する。ブログＤＢ５１と画像ＤＢ５２の両方の記憶可能容量が低下した状態であれば、テキストデータと画像データの両方を圧縮するなどである。一例として、テキストデータ用のブログ圧縮判定閾値と画像データ用のブログ圧縮判定閾値を設けることにより実現可能である。
【００７７】
また、圧縮対象部分の設定は、ブログごとや記事ごとに自動的に選択するようにしてもよい。
記事内容に応じて圧縮処理内容を決定する例としては、
・記事においてテキストデータが所定量以上ならテキストデータのみの圧縮を行い、所定量未満ならテキストデータと画像データの全体圧縮を行う。
・記事において画像データが存在すれば画像データのみの圧縮を行う。
などである。
更にブログごとに圧縮対象部分を選択する例としては、ブログの全体のテキスト／画像の比率からテキスト中心ブログか画像中心ブログかを判定し、テキスト中心ブログの場合はテキストデータの圧縮、画像中心ブログの場合は画像データの圧縮を行うということも考えられる。

逆に、アクセス時にユーザが感じる配信速度を重視する場合、テキスト中心ブログの場合は画像データの圧縮、画像中心ブログの場合はテキストデータの圧縮を行うということも考えられる。

【００７８】

なお、画像データとして動画が含まれている場合、更に動画圧縮と音声圧縮の両方、一方を選択することも考えられる。

【００７９】

以上のような圧縮対象部分の設定のほかに、圧縮方式の設定も多様に考えられる。公知の通り、画像データやテキストデータの圧縮には多様な方式が存在し、また圧縮率も多様に選択できる。可逆圧縮、非可逆圧縮という選択も可能である。

この圧縮方式についても、或る圧縮方式を固定的に用いてもよいし、状況に応じて選択するようにしてもよい。

例えばブログＤＢ５１や画像ＤＢ５２の記録可能なリソースが所定量以下になったら、より圧縮率の高い圧縮方式に切り替えるということが考えられる。

またブログごとや記事ごとに自動的に圧縮方式を選択するようにしてもよい。

例えば増加傾向指数が高いブログほど圧縮率が高くなるようにしたり、記事のアクセス可能性の低さの程度によって圧縮率の異なる圧縮方式を選択するなどである。

【００８０】

ブログサーバ１は図６のステップＳ２０５で、現在処理対象のブログにおける圧縮対象の１または複数の記事について圧縮処理を行った後、圧縮処理の対象となった記事に関する圧縮可否の情報（フラグ情報）を更新（圧縮可→圧縮否へ更新）する。

続いて、ブログサーバ１はステップＳ２０６で管理ＤＢ５３における圧縮解凍情報を更新する。ここでは当該ブログ内の圧縮を行った記事について、圧縮状態であることを示すように例えばフラグ情報を更新する。また圧縮履歴を追加する。

【００８１】

ステップＳ２０７でブログサーバ１は、圧縮した各記事についてタグ設定を行う。ここでいうタグとは、記事内容を示すキーワードや記事のジャンル情報などを示す情報で、記事の検索、抽出に利用できるようにするものである。

圧縮記事については、記事を対象とするテキスト検索がやりにくくなる。即ち圧縮記事も検索範囲に含めるようにしたい場合、検索時に、わざわざ解凍を行わなければならない。そこで、タグを設定して登録しておく。

ステップＳ２０７の処理を実行時は、処理対象の記事について、圧縮記事と圧縮前の元の記事（即ち未圧縮記事）との双方が記憶されている状態である。

従って、ステップＳ２０７でブログサーバ１は、未圧縮記事のデータから、頻出語の抽出、品詞解析による名詞抽出、ジャンル情報の取得などを行い、タグとして登録する１または複数の語句を設定する。

そしてブログサーバ１ステップＳ２０８で圧縮記事タグとして管理ＤＢ５３に登録する。即ち圧縮した記事のそれぞれに対応させて、キーワード等の１または複数の語句を登録する。

【００８２】

なお、この圧縮記事タグの設定及び登録は、図６の例の場合、或る記事を圧縮した際に、その記事について行うようにしているが、予め全ての記事について行っておいてもよい。特に記事の検索のために、タグを用いており、前記事についてタグ登録を行っているのであれば、タグを本実施の形態の圧縮記事タグとして用いればよいため、このステップＳ２０７，Ｓ２０８を実行する必要は無い。

但し、特に全記事についてタグ登録を行っていないシステムの場合、図６のステップＳ２０７、Ｓ２０８として圧縮記事タグ登録を行うことで、必要な記事についての必要最小限の処理となるため、ブログサーバ１の処理負担の増大防止に有効である。

【００８３】

ステップＳ２０９でブログサーバ１は、圧縮した記事について、元の圧縮前の記事デー

タを削除する。もちろん記事の一部のみの圧縮を行った場合は、圧縮した部分のみの元のデータを削除する。また、解凍記事を再圧縮する処理を行った場合は、元の圧縮前の記事データとしての解凍記事を削除する。

【００８４】

以上の処理により一つのブログについての圧縮処理を終えたら、ブログサーバ１はステップＳ２１０で他に処理対象となっているブログがあるか否かを確認する。

そして今回処理対象とする全てのブログについて処理を終えていた場合は、ステップＳ２１０から図６の圧縮処理を終える。

以上の図６の圧縮処理を行うことで、図５の圧縮判定処理で圧縮可と判定された記事についての実際の圧縮処理が行われ、記憶リソースの回復が図られる。

【００８５】

続いて、ブログやブログ内の記事に対するアクセス要求があった場合のブログサーバ１の処理を図７，図８で説明する。

ステップＳ３０１において、ユーザ端末５からのアクセス要求の受信を確認したブログサーバ１は、続くステップＳ３０２において、まず要求された記事が、圧縮記事であるか否かを判定する。

圧縮記事でなければ、ブログサーバ１はステップＳ３０２からＳ３０３に進み、通常に要求された記事を配信する。即ち該当の記事のウェブページデータをブログＤＢ５１から読み出し、ユーザ端末５に送信する。これによりユーザ端末５を使用している閲覧者は所望の記事を閲覧することができる。

【００８６】

アクセス要求された記事が圧縮記事であった場合、ブログサーバ１はステップＳ３０４に進み、解凍処理を行う。即ち該当の記事の圧縮状態のデータをブログＤＢ５１から読み出し、解凍処理を行う。そしてステップＳ３０５で解凍後のウェブページデータをユーザ端末５に送信する。これにより、圧縮されていた記事であっても、ユーザ端末５を使用している閲覧者は所望の記事を閲覧することができる。

【００８７】

なお、上述のように記事データの一部、特にウェブページデータにおけるファーストビューとして現れる領域以外のデータを圧縮するようにしていた場合、ブログサーバ１は、まず非圧縮記事部分をユーザ端末に送信しておき、その間に圧縮部分の解凍処理を行って、解凍でき次第、送信を行うという手法をとることができる。このようにすると解凍処理の時間を閲覧者に感じさせないような配信を実行でき、ブログサーバ１のサービスとしてのパフォーマンス維持が実現できる。

またファーストビュー以外の部分の圧縮に限らず、記事の一部を圧縮している場合は、記事内の非圧縮部分を先に送信するようにすることが同様の理由で望ましい。

【００８８】

圧縮記事を解凍して配信した後は、図８Ａ、図８Ｂ、図８Ｃに示すような処理例が考えられる。

まず図７のステップＳ３０５から図８Ａのステップ３１０に進む例では、ブログサーバ１は解凍記事、即ち圧縮を解凍した記事のデータ、即ち配信したウェブページデータを保存しておかずに消去する例である。

これは、当該記事への今回のアクセス要求は、あくまで例外的にアクセスが生じてしまったもので、この記事へのアクセス可能性が低いために圧縮された記事であることに代わりはないという考え方で、圧縮記事のまま保存しておくものである。

もしその後にアクセスが発生したら、その都度、解凍処理を行うことになる。アクセス要求に応じて解凍処理負担が発生するが、そもそもアクセスが少ないと考えれば、圧縮状態で保存しておくことで記憶リソース的に有利な状態を維持できる。

【００８９】

一方で、圧縮記事に対するアクセスが生じたということは、その圧縮記事（アクセス可能性が低いと判定された記事）について、アクセス可能性が高まってきた可能性があると

考えることもできる。
そこで図７のステップＳ３０５から図８Ｂのステップ Ｓ３２０に進む例が考えられる。ステップＳ３２０でブログサーバ１は、解凍した記事データをブログに組み込む。そしてステップＳ３２１で圧縮記事を削除する。即ち、それまでブログに組み込んでいた圧縮記事に代えて、解凍した記事データ（解凍記事）をブログに組み込むようにする。
ステップＳ３２２では、ブログサーバ１は管理ＤＢ５３における圧縮解凍情報を更新する。即ち当該ブログの該当の記事について、圧縮から解凍した状態の記事データ（即ち解凍記事）であることを示すように情報を更新する。また解凍の日時等の履歴情報を追加する。
【００９０】
このように解凍した場合、圧縮記事を解凍記事に置き換えることで、その後、アクセス要求があった場合に、解凍を行わずに配信できる。
なお、もし当該解凍記事に対して、その後もアクセスが少なく、図５の圧縮判定処理でアクセス可能性が低いと判定された場合は、図６の圧縮処理で再び圧縮されることになる。従って、相変わらず不人気な記事であった場合、再び圧縮されるため、記憶リソース維持の観点で図８Ａの処理と比べて大きな不利とはならない。
また、非可逆圧縮を用いて圧縮処理を行った場合には、その後の解凍処理によって解凍した解凍記事のデータを記憶しておいたとしても、未圧縮記事を記憶しておくよりもブログＤＢ５１において占有する記憶領域（必要記憶容量）が小さいため有利である。
【００９１】
図８Ｃは、解凍記事を圧縮記事とともに保存しておく例である。図７のステップＳ３０５から図８Ｃのステップ Ｓ３３０に進んだ場合、ブログサーバ１は、解凍した記事データをブログに対応づけて登録しておく。但し、圧縮記事はそのまま保持する。そしてステップＳ３３１でブログサーバ１は、管理ＤＢ５３における圧縮解凍情報を更新する。即ち当該ブログの該当の記事について、解凍記事が存在することを示すように情報を更新する。また解凍の日時等の履歴情報を追加する。
【００９２】
この場合、解凍記事を保存しておくことで、その後にアクセス要求があった場合に、解凍処理を行わずに配信できる。
但し、解凍記事と圧縮記事の両方を保存しておくことで、記憶リソースの負担を大きくする。そこで例えば解凍記事は、或る一定期間を経過したら削除することが考えられる。このようにすれば、アクセス要求が生じた場合、一定期間は、再度アクセスがあったときに解凍処理を経ずに配信できる状態を作ることができる。
また、このように圧縮記事と解凍記事を併存させた場合、当該記事がその後の図５の処理で圧縮対象となった場合、図６の圧縮処理では、解凍記事を削除するという処理を行えばよい。
【００９３】
＜４．第２の実施の形態＞
記憶リソースの確保の観点からは、画像や動画などの圧縮の際に圧縮率が高い非可逆圧縮を用いることが好ましい。しかし、記事の閲覧を考えると、解凍の際に画像等が元の状態に完全に戻らないため、画像等が見難くなってしまう虞がある。
そこで、本実施の形態では、記事の圧縮の際に可逆圧縮と非可逆圧縮を適切に選択する。
【００９４】
ブログサーバ１が実行する圧縮判定処理及び圧縮処理について、図９Ａ及び図９Ｂに示す。但し、ブログサーバ１は図５に示す圧縮判定処理の一部の処理を図９Ａに示す処理に置き換え、図６に示す圧縮処理の一部の処理を図９Ｂに示す処理に置き換えることで、本実施の形態を実現する。
【００９５】
先ず、圧縮判定処理について、図５及び図９Ａを参照して説明する。

ブログサーバ１は、ステップＳ１０１乃至Ｓ１０４の処理を行うことにより、圧縮対象ブログの一つを適切に選択する。
ブログサーバ１は、ステップＳ１０５を実行することにより、圧縮対象ブログと判定したブログに属する一つの記事を選択する。
そして、ブログサーバ１はステップＳ１０６の代わりに図９Ａのステップ１２０を実行することにより、選択した記事が二次圧縮済みであるか否かを判定する。

【００９６】

本実施の形態では、圧縮を２段階に分ける。具体的には、未圧縮記事及び解凍記事に対して１段階目の圧縮を行う際は、一次圧縮として可逆圧縮を適用する。そして、一次圧縮によって圧縮された圧縮記事に対して更に圧縮を行う際は、二次圧縮として非可逆圧縮を適用する。
即ち、図９Ａのステップ１２０の処理は、非可逆圧縮を適用した記事であるか否かを判定することと同じ処理である。
ステップＳ１２０において二次圧縮済みでないと判定した場合、これ以上の圧縮は行わないため、ステップＳ１１０に進む。

【００９７】

一方、二次圧縮済みでないと判定した場合、ブログサーバ１はステップＳ１２０からステップＳ１０７に進む。図５のステップＳ１０７乃至Ｓ１１１の各処理は前述した通りであり、詳述を省く。

【００９８】

次に、ブログサーバ１が行う圧縮処理について、図６及び図９Ｂを参照して説明する。
ブログサーバ１は、ステップＳ２０１乃至Ｓ２０３を実行することにより、圧縮判定処理において圧縮対象ブログと判定されたブログを一つ特定する。
そして、続くステップＳ２０４によって、ブログサーバ１は当該圧縮対象ブログの中で圧縮対象とされた記事を全て特定する。

【００９９】

ブログサーバ１は、ステップＳ２０５の代わりに図９Ｂに示すステップＳ２２０，Ｓ２２１を実行する。
先のステップＳ２０４で特定した圧縮対象記事は、未圧縮記事か解凍記事か一次圧縮（可逆圧縮）を適用された圧縮記事の何れかである。即ち、これ以上圧縮できない状態とされる二次圧縮（非可逆圧縮）を適用された圧縮記事は圧縮対象記事とはされていないため、ステップＳ２０４で特定した圧縮対象記事には含まれない。そして、一次圧縮済みである記事は可逆圧縮を適用された圧縮記事のみであり、一次圧縮済みでない記事は未圧縮記事若しくは解凍記事である。

【０１００】

ブログサーバ１は、ステップＳ２２０を実行することにより、既に可逆圧縮を適用された記事であるにも関わらず更なる圧縮を行うと判定された記事に対して非可逆圧縮を適用する。これにより、更なる記憶リソースの確保が図られる。
そして、ブログサーバ１は更にステップＳ２２１を実行することにより、未圧縮記事若しくは解凍記事であって圧縮を行うと判定された記事に対して可逆圧縮を適用する。これにより、記憶リソースの確保が図られる。

【０１０１】

ステップＳ２２０及びＳ２２１を実行することにより圧縮対象記事の圧縮をした後、ブログサーバ１はステップＳ２０６で管理ＤＢ５３における圧縮解凍情報を更新する。具体的には、当該ブログ内で今回圧縮を行った記事について、圧縮状態であることを示すように例えばフラグ情報を更新する。また圧縮履歴を追加する。このとき、圧縮状態であることを示すだけでなく、可逆圧縮が適用されているか非可逆圧縮が適用されているかを区別して記憶することが望ましい。これにより、何れの圧縮形式が適用されているかを容易に判別することができる。

【０１０２】

なお、一つの記事に対して非可逆圧縮及び解凍を繰り返した場合には、当該記事に含まれる画像や動画などが過剰に粗くなってしまう可能性がある。これを防止するために、非可逆圧縮の適用の上限回数を定めてもよい。その場合には、圧縮解凍情報に非可逆圧縮を適用した回数を記憶することが望ましい。そして、ステップＳ２２０の処理では、一次圧縮済みであり且つ非可逆圧縮の適用回数が上限未満となっている記事に対して、非可逆圧縮を適用することとなる。

【０１０３】

以上の処理を行えば、例えばアクセス可能性が低い記事に対しては先ず可逆圧縮が適用されることにより記憶リソースの確保が図られる。そこで当該記事に対するアクセス要求があった場合には、未圧縮記事と同じ品質の記事へと復元することが可能である。

そして、可逆圧縮が適用された記事に対するアクセス要求が一定期間なかった場合などには、更なる記憶リソースの確保のために非可逆圧縮が適用される。

これにより、偶然一定期間アクセス要求がなかった記事に対しては未圧縮記事と同一品質の記事へと復元可能な可逆圧縮が適用されるため、その後のアクセス要求の受信により閲覧者に記事の見にくさなどを感じさせることのない状態へと回復することができる。

また、長期間に亘ってアクセス要求がない記事に対しては、可逆圧縮を経て非可逆圧縮が適用されるため、最低限の品質を保ちつつ最大限の記憶容量の確保を図ることができる。

【０１０４】

なお、記事のアクセス可能性が著しく低いと判定した場合には、可逆圧縮を経ずに非可逆圧縮を適用することにより、記憶リソースの確保を優先してもよい。また、ブログに必要な記憶リソースの確保が難しい場合などにも、可逆圧縮を経ずに非可逆圧縮を適用することが考えられる。

【０１０５】

＜５．第３の実施の形態＞

本実施の形態では、ブログごとに許可する総データ量の上限値である総データ量上限値を適切に設定する例を説明する。

少しの記事をアップロードした後、飽きてしまうユーザもいれば、長く続けるユーザもいる中で、総データ量上限値を一律同じ設定とすることは適切ではない場合がある。特に、多くの記事が投稿されており閲覧者がたくさんいるようなブログにおいて、総データ量上限値に達してしまった場合などは、ユーザの利便性を考慮すると好ましくない状況である。

そこで、ユーザごとのブログの利用状況に応じて総データ量上限値を設定する。

【０１０６】

ブログサーバ１はステップＳ４０１で、処理対象のブログを一つ特定し、当該ブログに対して以降に示すステップＳ４０２乃至Ｓ４０６の処理を実行する。

ブログサーバ１はステップＳ４０２で処理対象のブログに総データ量上限値が設定されているか否かを判定する。総データ量上限値が未設定の場合は、ステップＳ４０３で総データ量上限値のデフォルト値（例えば２ＧＢなど）を設定して、ステップＳ４０６の処理へと進む。

【０１０７】

総データ量上限値が未設定でない場合、ブログサーバ１はステップＳ４０４で、管理ＤＢ５３に記憶された各種情報から記述者の利用状況を示す情報を取得する。

ここでいう記述者の利用状況を示す情報とは、例えば、ブログを頻繁に利用している記述者か否かを示す情報や、当該ブログが人気のあるブログか否かを示す情報や、当該ブログに対して現在設定されている総データ量上限値に対する余裕度合いを示す情報などである。これらの情報の一つを取得してもよいし、複数の情報を取得してもよい。

ここでは、記述者が自身のブログを頻繁に利用しているか否かを示す情報としてブログ更新頻度を示す値と、総データ量上限値に対する余裕度合いを示す情報としてブログの総データ量及び総データ量上限値を取得する。

【０１０８】
続いて、ブログサーバ１はステップＳ４０５で利用状況に応じた総データ量上限値の設定を行う。
一例を示す。例えば、ブログ更新頻度を示す値に基づいて以下のようにブログを３段階で評価する。
・ブログ更新頻度が１日１回以上＝更新頻度高
・ブログ更新頻度が１週間に１回以上かつ１日１回未満＝更新頻度中
・ブログ更新頻度が１週間に１回未満＝更新頻度低
更に、ブログの総データ量と総データ量上限値を取得し、そこから当該ブログの空き容量（「ブログに設定された総データ量上限値」－「ブログの総データ量」）を算出する。
【０１０９】
そして、更新頻度高であって且つ空き容量が５０ＭＢ（Mega Byte）未満であるブログに対して、総データ量上限値を一段階上げる処理を行う。また、更新頻度中であって且つ空き容量が３０ＭＢ未満であるブログに対して、総データ量上限値を一段階上げる処理を行う。更に、更新頻度低であって且つ空き容量が１０ＭＢ未満であるブログに対して、総データ量上限値を一段階上げる処理を行う。
総データ量上限値を一段階上げる処理とは、例えば、現在の総データ量上限値に１００ＭＢを加えた容量を新たな総データ量上限値として設定する処理である。
【０１１０】
次に、ブログサーバ１はステップＳ４０６において、次の処理対象のブログがあるか否かを判定し、次の処理対象のブログがある場合にはステップＳ４０１に戻り、次のブログを特定する。一方、次の処理対象のブログが無い場合には、ステップＳ４０６から図１０に示す総データ量上限値設定処理を終了する。
【０１１１】
以上の処理を実行することにより、空き容量が少ないブログであっても、更新頻度が少なく総データ量上限値を変更する緊急性が低いブログに対しては、総データ量上限値を変更しない。従って、無駄な記憶リソースを提供してしまう可能性を抑制することができる。
一方、空き容量が少ないブログであり、且つ更新頻度が高いブログに対しては、総データ量上限値を変更することによりそれまでよりも高い総データ量上限値が設定される。従って、ブログに新たな記事を投稿する際に空き容量が足らずに投稿できない事態を容易に回避することができる。
【０１１２】
なお、総データ量上限値をユーザのブログの利用状況に応じて変更する場合には、図１０に示す一連の処理を定期的に行うことが望ましい。定期的に行うことにより、例えばそれまで頻繁に記事を投稿していたユーザが投稿しなくなった場合などのようにユーザの利用状況が変化した場合にも適切な総データ量上限値を設定することができる。
【０１１３】
なお、上記では総データ量上限値を増加させる例について述べたが、総データ量上限値を減少させる例も考え得る。例えば、デフォルトの総データ量上限値が２ＧＢである場合において、２ＧＢよりも高い総データ量上限値が設定されており且つブログ更新頻度低と判定されたブログに対しては、総データ量上限値を引き下げることにより記憶リソースを確保する。この場合に、ブログの総データ量が新規に設定し直した総データ量上限値を超えてしまった場合には、圧縮判定処理及び圧縮処理を行いブログの総データ量を総データ量上限値以下に抑えることが望ましい。
特に、ブログにアクセスしたユニークユーザ数が減少しているブログに対しては、総データ量上限値を下げることによって記事の圧縮処理を実行せざるを得なくなったとしても、アクセス要求による解凍処理を実行する可能性が低いため、有効である。
【０１１４】
また、ユーザの利用状況に応じて総データ量上限値を撤廃して無制限に使わせることも

考え得る。例えば、記述者の更新頻度が著しく高い上に閲覧者が多いブログなどの総データ量上限値を撤廃する。こうすれば、以降において総データ量上限値の再設定や更新などの処理対象から当該ブログを外すことができるため、ブログサーバ１の処理負担の軽減を図ることができる。

【０１１５】

＜６．第４の実施の形態＞

本実施の形態では、取得するアクセス可能性指数が複数の情報から算出された値とされる。即ち、第１の実施の形態では、管理ＤＢ５３にブログ実績情報として記憶された情報のうちの一つをアクセス可能性指数として取得するのに対し、本実施の形態では、管理ＤＢ５３にブログ実績情報として記憶された複数の情報からアクセス可能性指数を算出して取得する。

【０１１６】

このために、ブログサーバ１は図５に示す圧縮判定処理を行う前にアクセス可能性指数を算出しておくことが考えられる。もちろん、その都度算出してもよい。

【０１１７】

図１１は、アクセス可能性指数を算出する処理を示すものである。

ブログサーバ１はステップＳ５０１において、処理対象となるブログを一つ特定する。以降の処理では、特定したブログに対してアクセス可能性指数を算出する。

ステップＳ５０２でブログサーバ１は、処理対象のブログに含まれる複数の記事ごとのアクセス可能性指数を算出するために用いる情報をブログＤＢ５１のブログ実績情報から取得する。即ち、記事が１０個あるブログが処理対象であれば、１０個の記事ごとに情報を取得する。

具体的には、ブログ実績情報として記事ごとに記憶された記事ごとのページビュー数、記事にアクセスしたユニークユーザ数、記事ごとの総被リンク数、記事のランクを示す値、記事に対するアクセス要求が無い期間の長さ、所定期間における記事についてのページビュー数、記事についてのページビューの増加傾向を示す値、記事ページ上に掲載される広告に対するクリック数などの情報から複数の情報を取得する。

【０１１８】

ステップＳ５０３でブログサーバ１は、取得した複数の情報からアクセス可能性指数を記事ごとに算出する。このとき、情報の種類ごとに正規化した数値を加算することにより当該記事のアクセス可能性指数を求めてもよいし、情報の種類ごとの重要性に応じて重みを付けて算出してもよい。

続いて、ブログサーバ１はステップＳ５０４において、算出した記事ごとのアクセス可能性指数を管理ＤＢ５３に記憶する。アクセス可能性指数を管理ＤＢ５３に記憶しておくことにより、図５に示す圧縮判定処理におけるステップＳ１０７の処理では、記憶されたアクセス可能性指数そのものを取得すればよい。

【０１１９】

次に、ステップＳ５０５でブログサーバ１は、次の処理対象となるブログがあるか否かを判定し、次の処理対象のブログがある場合にはステップＳ５０１に戻り、次の処理対象のブログを特定する。一方、次の処理対象となるブログが無い場合には、ステップＳ５０５から図１１に示すアクセス可能性指数算出処理を終了する。

なお、処理対象のブログはブログＤＢ５１に記憶された全てのブログとしてもよいし、一部としてもよい。一部とする例としては、例えば、全ての記事を圧縮しなくてはならないほどブログ圧縮判定閾値を大幅に超えているブログは、記事ごとにアクセス可能性指数を算出する必要がなくなるため、処理対象外としてもよい。換言すれば、ブログ圧縮判定閾値を少し超えている程度のブログは、何れの記事を圧縮するべきかを検討する必要があるため、アクセス可能性指数の算出対象のブログとする。

【０１２０】

なお、図１１のアクセス可能性指数算出処理をバッチ処理で定期的に実行する場合において、アクセス可能性指数算出処理の処理対象をブログＤＢ５１に記憶された全てのブロ

グの全ての記事とした場合には、アクセス可能性指数を事前にバッチ処理などで算出しておくことにより、処理のスケジュールの自由度を高めることができる。

【０１２１】

＜７．第５の実施の形態＞

本実施の形態では、処理対象のブログの総データ量が所定値以下となるまで、アクセス可能性指数が低い記事から圧縮をしていく例を説明する。

圧縮処理の流れを図１２に示す。処理のおおまかな流れは図６に示す圧縮処理と同様であるため、説明を適宜省略する。

【０１２２】

ブログサーバ１は、ステップＳ２０１乃至Ｓ２０３の処理を実行することにより、圧縮対象ブログの一つを選択する。そして、選択したブログに対して、以降の各処理を実行する。

先ず、ブログサーバ１はステップＳ２３０において、選択したブログに属する各記事のうちアクセス可能性指数の低い記事を選択する。そして、当該記事に対してステップＳ２０５乃至Ｓ２０９の処理を行うことにより記事の圧縮等を行う。ステップＳ２３０の処理は、図６におけるステップＳ２０４の処理に代えて実行する処理である。

【０１２３】

一つの記事の圧縮を終えたブログサーバ１は、続くステップＳ２３１において、当該ブログの全体の総データ量が所定値以下であるか否かを判定する。

所定位置よりも大きい場合、このブログの各記事に対する圧縮は十分ではないと判定し、ステップＳ２３０の処理に戻って次の記事を選択する。

なお、全ての記事を圧縮したために次に選択するべき記事が存在しない場合は、ステップＳ２３０の処理に遷移せずにステップＳ２１０へ遷移して、次のブログに対する処理を開始するか否かを判定してもよい。

【０１２４】

選択したブログの総データ量が所定値以下となった場合、ブログサーバ１は当該ブログに対する圧縮処理を終了し、次の処理対象のブログを選択するための処理としてステップＳ２１０を実行する。

【０１２５】

ステップＳ２３１の処理で用いる所定値としては、例えば、ブログ圧縮判定閾値を用いることが考えられる。ブログ圧縮判定閾値を用いることにより、判定に用いる数値を新たに算出したり記憶したりする必要が無いため、処理負担の軽減や記憶リソースの確保に寄与することができる。

また、他にも、所定値としてブログ圧縮判定閾値に０．８などの係数を乗算した値を用いてもよい。圧縮対象ブログが記事を頻繁に投稿するユーザのものであった場合、これにより１回の投稿でブログ圧縮判定閾値を再度超えてしまうことを抑制できる。従って、当該ブログが再び圧縮対象ブログとなるまでの時間を長くすることができるため、図１２に示す圧縮処理（或いは図５のステップＳ１０５乃至Ｓ１１０の処理）の実行頻度を下げることができる。これにより、ブログサーバ１の処理負担の軽減を図ることができる。

【０１２６】

本実施の形態によれば、処理対象のブログの総データ量が所定値以下となったことに応じて即座に各記事の圧縮処理が終了するため、過剰となり得る圧縮処理を抑制することができ、ブログサーバ１の処理負担の軽減を図ることができる。

なお、状況に応じて所定値を設けたり設けなかったりしてもよい。例えば、処理対象のブログの全ての記事のアクセス可能性指数が低い場合などは、所定値を設けずに全ての記事に対する圧縮処理（即ち図１２のステップＳ２０５～Ｓ２０９の処理）を行ってしまってもよい。このようなブログは、閲覧者のアクセスによる記事の解凍処理を実行する可能性が低いため、解凍処理の実行による処理負担の増加を懸念することなく記憶リソースの確保を行うことができる。

【０１２７】

＜８．第６の実施の形態＞
本実施の形態では、当該ブログの総データ量がブログ圧縮判定閾値をどの程度超えているかに基づいて、当該ブログに属する記事に対する圧縮可否判定を行う例について説明する。
圧縮判定処理について図１３に示す。処理のおおまかな流れは図５に示す圧縮判定処理と同様であるため、説明を適宜省略する。
【０１２８】
ブログサーバ１は、ステップＳ１０１乃至Ｓ１０４において、ブログが圧縮対象ブログであるか否かを判定する。このとき圧縮対象ブログとして判定されたブログに関しては、総データ量がブログ圧縮判定閾値をどの程度超えたのか（即ち超過容量）を把握しておく。
続いて、ブログサーバ１は、ステップＳ１０５において選択した記事について、続くステップＳ１０６～Ｓ１０９を実行することにより当該記事を圧縮すべきか否かを判定する。このとき、アクセス可能性指数の大きさにより圧縮すべきか否かを判定するが、本実施の形態では、上記超過容量も加味して圧縮すべきか否かの判定を行う。
具体的に例えば、先の第１の実施の形態において、図５のステップＳ１０８の説明の際に、アクセス可能性指数として当該記事に対するアクセス要求が無い期間の長さを選択した場合、当該期間長が３年以上であるか未満であるかを判定し、３年以上アクセス要求が無い記事はアクセス可能性が低いとして圧縮対象記事と判定する例を挙げた。
【０１２９】
本実施の形態によれば、この「３年」という閾値を超過容量に応じて変えるということである。一例を挙げると、超過容量が多いブログ、即ち多くの記事を圧縮しなければならないブログに関しては、「２年」という閾値を設ける。これにより、２．５年間アクセス要求が無いような記事も圧縮処理の対象となるため、記憶リソースの確保を促進させることができる。また、超過容量の少ないブログ、即ち少しの記事を圧縮するだけでよいブログに関しては、「５年」という閾値を設ける。これにより、３年間アクセス要求が無い記事は圧縮処理の対象外となるため、圧縮記事に対する解凍処理が発生する可能性を低減させつつ効率的な記憶リソースの確保を行うことができる。
【０１３０】
そのために、ブログサーバ１はステップＳ１３０においてアクセス可能性指数を取得した後、ステップＳ１３１において処理対象のブログの総データ量の超過容量に基づく判定閾値を取得する。ステップＳ１３１においては、判定閾値の取得の代わりに判定閾値を算出する処理、即ち上述した２年や３年や５年などの閾値を算出する処理を実行してもよい。
【０１３１】
本実施の形態によれば、各ブログの総データ量に応じて適切な判定閾値が設定されることにより、好適な記憶リソースの確保を行うことができる。
【０１３２】
＜９．第７の実施の形態＞
本実施の形態では、ブログサーバ１は、各記事について、増加傾向指数及びアクセス可能性指数に基づいた圧縮可否判定を実行するだけでなく、既に圧縮した記事の解凍可否判定も実行する。
【０１３３】
ブログサーバ１は図１４の圧縮解凍判定処理を例えば定期的、或いは所定のタイミングで実行する。この図１４の処理は第１の実施の形態における図５の圧縮判定処理に代わる処理と考えればよい。図５と同一の処理については同一のステップ番号を付して重複説明を避ける。
【０１３４】
ブログサーバ１は処理対象のブログを特定し（Ｓ１０１）、増加傾向指数と人気度指数を取得（Ｓ１４０）した後、ステップＳ１０５、Ｓ１４１、Ｓ１４２、Ｓ１４３、Ｓ１１

０で各記事について判定を行う。
ブログサーバ１は、ステップＳ１０５で一つの記事を選択したら、圧縮済か否かに関わらずステップＳ１４１に進んでアクセス可能性指数を取得する。この処理では、図５のステップＳ１０７のようにアクセス可能性指数としてのブログ実績情報を取得してもよいし、図１０で算出したアクセス可能性指数を取得してもよい。
そしてブログサーバ１はステップＳ１４２で圧縮可否及び解凍可否の判定を行う。
即ちブログサーバ１は、未圧縮記事もしくは解凍記事については、図５のステップＳ１０８と同様に、増加傾向指数に基づいた圧縮可否判定を行う。
【０１３５】
一方、圧縮記事についてはブログサーバ１は、人気度指数及びアクセス可能性指数を用いて解凍可否判定を行う。
例えば、総ページビューなどの情報を用いて、人気度指数を高／中／低の３段階などに分ける。
そしてアクセス可能性指数として、例えば当該記事のページビュー数Ｎを用い、
・人気度指数が低のブログの場合・・・記事のページビュー数ＮがＮ１以上であれば解凍
・人気度指数が中のブログの場合・・・記事のページビュー数ＮがＮ２以上であれば解凍
・人気度指数が高のブログの場合・・・記事のページビュー数ＮがＮ３以上であれば解凍
とする。但しＮ１＞Ｎ２＞Ｎ３である。
つまり人気ブログの圧縮記事であれば、ページビュー実績が少々高くなっていたら解凍可とするが、不人気ブログの圧縮記事であれば、ページビュー数により解凍可とする閾値を上昇させて判定を行う。人気ブログの記事ほど、一旦圧縮されても解凍可とされやすいことになる。これは人気ブログほど、アクセス可能性は上昇しやすいと考えられることによる。
【０１３６】
他の例として、アクセス可能性指数として、例えば当該記事のページビューの増加傾向を示す値Ｋを用い、
・人気度指数が低のブログの場合・・・増加傾向値ＫがＫ１以上であれば解凍
・人気度指数が中のブログの場合・・・増加傾向値ＫがＫ２以上であれば解凍
・人気度指数が高のブログの場合・・・増加傾向値ＫがＫ３以上であれば解凍
とすることも考えられる。但しＫ１＞Ｋ２＞Ｋ３である。
つまり人気ブログの圧縮記事であれば、ページビューの少々の増加傾向が観測されたら解凍可とするが、不人気ブログの圧縮記事であれば、ページビューの大幅な増加傾向が観測されたら解凍可するというような判定を行う。この場合も、人気ブログの記事ほど、一旦圧縮されても解凍可とされやすいことになる。
【０１３７】
そしてステップＳ１４３ではブログサーバ１は、当該記事についての圧縮可否判定または解凍可否判定を判定情報として記憶する。例えば管理ＤＢ５３の判定情報として当該記事について圧縮可否のフラグまたは解凍可否のフラグを更新または維持する。
ステップＳ１１０，Ｓ１１１は図５と同様である。
【０１３８】
ブログサーバ１はまた、図１５の圧縮解凍処理を逐次実行する。これは第１の実施の形態における図６の圧縮処理に代わるものである。図６と同一処理は同一のステップ番号を付し、詳細な説明は避ける。
【０１３９】
ブログサーバ１はステップＳ２０１で処理対象のブログを特定し、ステップＳ２０２で判定情報を取得する。
ブログサーバ１はステップＳ２４０で、圧縮可否の判定情報を参照して圧縮する記事があるかどうかを判定する。そしてステップＳ２０４で圧縮する記事を特定する。ステップＳ２４０では、圧縮記事があるか否かを判定してもよいが、処理対象のブログが圧縮対象ブログであるか否かを判定してもよい。

１または複数の記事が圧縮する記事として特定された場合、ブログサーバ１はステップＳ２０５で、１または複数の各記事データの圧縮処理を行い、図５の場合と同様にステップＳ２０７，Ｓ２０８，Ｓ２０９で、圧縮した記事についてのタグ設定、タグ登録、及び元の記事データの削除を行う。

【０１４０】

以上のステップＳ２０５～Ｓ２０９を行った後、もしくはステップＳ２４０で圧縮する記事が存在しないとされた後、ブログサーバ１はステップＳ２４１で、解凍する記事があるか否かを判定する。即ちステップＳ２０２で取得した判定情報を参照して、圧縮されている記事の内で解凍すべき記事があるか否かを判定する。解凍する記事が存在しなければステップＳ２４５に進む。

【０１４１】

１または複数の解凍すべき記事が存在する場合、ブログサーバ１はステップＳ２４２で解凍する記事の特定を行い、続くステップＳ２４３で、特定した記事の解凍処理を行う。この場合、解凍した記事データは、それまでの圧縮記事に代えてブログに組み込むようにする。またステップＳ２４４でブログサーバ１は圧縮記事を削除する。

これにより、ブログ内の圧縮されていた記事が、アクセス可能性の上昇に応じて、解凍記事に回復されたことになる。

【０１４２】

ブログサーバ１はステップＳ２４５で管理ＤＢ５３における圧縮解凍情報を更新する。ここでは当該ブログ内の圧縮を行った記事について、圧縮状態であることを示すようにフラグ情報を更新する。また圧縮履歴を追加する。更に当該ブログ内の解凍を行った記事について、解凍状態であることを示すようにフラグ情報を更新する。また解凍履歴を追加する。

【０１４３】

以上の処理により一つのブログについての圧縮処理及び解凍処理を終えたら、ブログサーバ１はステップＳ２１０で他のブログについて処理を行うか否かを確認する。

そして今回処理対象とする全てのブログについて処理を終えていた場合は、ステップＳ２１０から図１５の圧縮解凍処理を終える。

以上の図１５の圧縮解凍処理を行うことで、図１４の圧縮解凍判定処理で圧縮可と判定された記事についての実際の圧縮処理や、解凍可と判定された記事についての実際の解凍処理が行われる。これにより記憶リソースの回復が図られるとともに、アクセス可能性が高くなった記事については解凍が行われ、アクセス時のパフォーマンスを向上させる。

【０１４４】

例えば或るブログの記事について、そのブログの記述者によって内容の変更があったり、その記事に対して閲覧者からコメントが投稿されたりした場合など、その記事自体に更新があった場合には、その記事に対するアクセス可能性は高くなる。本実施の形態によれば、このような記事は、圧縮を解いておくという動作が実現される。

そこで、このような動きを事前に察知して、既に圧縮したもののアクセス可能性の高まったブログ記事については、圧縮状態を解凍しておく。

このようにすれば、実際にアクセスしてきた場合に、その時点で解凍処理をせずにすぐに記事を配信することができる。

【０１４５】

また、人気上昇傾向のブログにおける圧縮記事について解凍対象とする判定を行うことも可能となる。

人気が上昇したブログについては、どの記事もアクセス可能性が高くなるため、圧縮記事については予め解凍しておくことで、閲覧時の解凍処理を不要とし、応答性の向上やその際の処理負担の削減を実現できる。

【０１４６】

＜１０．第８の実施の形態＞

本実施の形態として、ブログサーバ１が記事内容に基づいて、既に圧縮した記事を解凍

するか否かを判定する例を説明する。
例えば、あるテーマについて言及した記事について、当初にアップロードされた時点ではあまり話題にならなかったものの、数年後に、その記事で扱ったテーマに関する何らかの事件や事象が世の中で起きたために、その記事が掘り起こされる可能性が高まる（つまり、アクセス可能性が高まる）ことや、その記事に対する被リンク数が増加するといったことが起きうる。
そこで、このような動きを事前に察知して、既に圧縮したもののアクセス可能性の高まった記事については、圧縮状態を解凍しておくようにする。
このようにすれば、実際にアクセス要求があった場合に、その時点で解凍処理をせずにすぐに記事を配信することができる。

【０１４７】
図１６に解凍判定処理を示す。ブログサーバ１は、第１の実施の形態の各処理に加え、例えば定期的などに逐次この解凍判定処理を行う。
ステップＳ６０１でブログサーバ１は１または複数の抽出語句を設定する。例えば時事語を抽出語句とする。例えば新聞やニュースで頻出する言葉や、芸能関係でよく使われ出した言葉、流行語などである。或いは、話題になっているジャンルや関連語などを抽出語句としてもよい。例えば、オリンピック開催期間であれば、「スポーツ」というジャンルや、各種競技の名称、選手名などを抽出語句とする。

【０１４８】
ブログサーバ１はステップＳ６０２で処理対象のブログを一つ特定する。そしてステップＳ６０３で、特定したブログに圧縮記事が存在するか否かを確認する。例えば管理ＤＢ５３の圧縮解凍情報を確認すればよい。
もし圧縮記事が存在しなければ、解凍判定の必要はないためステップＳ６０７に進む。
圧縮記事が存在するブログであった場合は、ブログサーバ１はステップＳ６０４に進み、当該ブログの圧縮記事タグの情報を管理ＤＢ５３から取得する。圧縮記事タグとしては、少なくとも現在圧縮状態である１または複数の圧縮記事について設定されたタグ情報を含む。即ち、図６のステップＳ２０８などで登録されたタグ情報である。

【０１４９】
ステップＳ６０５でブログサーバ１は、ステップＳ６０１で設定した抽出語句と管理ＤＢ５３から取得した圧縮タグ情報を比較し、解凍する記事を判定する。
つまり抽出語句と同一または類似の圧縮記事タグが登録されている記事を抽出し、解凍する記事と判定する。
ステップＳ６０６でブログサーバ１は、判定情報を管理ＤＢ５３に記憶する。つまり解凍すると判定した記事の情報を記憶する。

【０１５０】
以上の処理により一つのブログについての解凍判定処理を終えたら、ブログサーバ１はステップＳ６０７で次の処理対象のブログがあるか否かを確認する。ある場合はステップＳ６０２に戻り、次の処理対象のブロクを特定して上記同様のステップＳ６０３以降の処理を行う。
今回処理対象とする全てのブログについて処理を終えていた場合は、ステップＳ６０７から図１６の解凍判定処理を終える。

【０１５１】
この解凍判定処理とともに、ブログサーバ１は図１７の解凍処理を逐次実行する。
ブログサーバ１はステップＳ２０１で処理対象のブログを特定し、ステップＳ２０２で判定情報を取得する。
ブログサーバ１はステップＳ２４１Ａで、取得した判定情報を参照して、圧縮されている記事のうちで解凍すべき記事があるか否かを判定する。解凍すべき記事が存在しなければステップＳ２１０に進む。

【０１５２】
１または複数の解凍すべき記事が存在すると判定した場合、ブログサーバ１はステップ

Ｓ２４２Ａで、解凍する記事を特定した後、ステップＳ２４３Ａで当該特定された１または複数の記事の解凍処理を行う。この場合、解凍した記事データは、それまでの圧縮記事に代えてブログに組み込むようにする。またステップＳ２４４Ａでブログサーバ１は圧縮記事を削除する。これにより、ブログ内の圧縮されていた記事が解凍記事に回復されたことになる。

ブログサーバ１はステップＳ２４５Ａで管理ＤＢ５３における圧縮解凍情報を更新する。ここでは当該ブログ内の解凍を行った記事について、解凍状態であることを示すようにフラグ情報を更新する。また解凍履歴を追加する。

【０１５３】

以上の処理により一つのブログについての解凍処理を終えたら、ブログサーバ１はステップＳ２１０で次の処理対象のブログがあるか否かを確認する。ある場合はステップＳ２０１に戻って、他の処理対象のブログを特定し、同様の処理を実行する。

今回処理対象とする全てのブログについて処理を終えていた場合は、ステップＳ２１０から図１７の解凍処理を終える。

【０１５４】

以上の図１７の処理により、図１６の解凍判定処理で解凍可と判定された記事についての実際の解凍処理が行われる。

これにより、時事語、流行語、流行のテーマなどを含む記事が、アクセス可能性が高くなるであろうと予測して解凍が行われる。これによりその後のアクセス増加時に解凍処理を不要にすることができる。

【０１５５】

＜１１．第９の実施の形態＞

本実施の形態は、圧縮記事についてアクセスに応じて解凍が行われた場合、ブログサーバ１は解凍から所定期間、当該記事を圧縮する記事と判定しないようにする例である。例えば図８Ｂまたは図８Ｃで説明したようにアクセスに応じて解凍が行われた後、その解凍記事を記憶しておく場合に、図５の処理に代えて用いることができる圧縮判定処理である。

【０１５６】

図１８に本実施の形態としての圧縮判定処理を示す。図５と同一の処理については同一のステップ番号を付して重複説明を避ける。

【０１５７】

図１８に示す圧縮判定処理でも、ブログサーバ１は、ステップＳ１０１でブログを特定し、ステップＳ１０２で増加傾向指数を取得した後、ステップＳ１０３で増加傾向指数に応じた圧縮対象ブログ判定閾値を設定したら、ステップＳ１０４で処理対象のブログにおける総データ量が圧縮対象ブログ判定閾値を超えているか否かを判定することにより、当該ブログが圧縮対象ブログであるか否かを確認する。

【０１５８】

続いてブログサーバ１は、ステップＳ１０５で記事を一つ選択し、ステップＳ１０６で当該記事が圧縮済みであるか否かを判定する。記事が圧縮済みである場合は、ステップＳ１１０の処理へ遷移する。

当該記事が圧縮済みでない場合、ブログサーバ１はステップＳ１５０で当該記事が解凍された記事であるか否かを判定する。解凍された記事ではない場合、即ち未圧縮状態の記事である場合には、ブログサーバ１はステップＳ１０７乃至Ｓ１０９の各処理を当該記事に対して行う。

当該記事が解凍された記事である場合、ブログサーバ１はステップＳ１５１において解凍から所定時間経過しているか否かを判定する。例えば、当該記事の解凍日時を確認し、現時点で所定期間を経過しているか否かを確認する。所定期間は例えば１ヶ月などとする。解凍日時の確認は管理ＤＢ５３の圧縮解凍情報における履歴情報を参照して行う。なお、圧縮／解凍が複数回行われている記事の場合は、最新の解凍日時を確認する。

そして解凍から所定時間が経過している場合、ブログサーバ１は当該記事を再圧縮可能

と判断し、ステップＳ１０７乃至Ｓ１０９の各処理を当該記事に対して行う。即ち、アクセス可能性指数に応じた圧縮可否判定を行い、判定情報を記憶する。
一方、解凍から所定時間経過していない場合には、ブログサーバ１は当該記事の圧縮可否判定は行わずに、即ち圧縮は行わないとされて、ステップＳ１１０の処理に進む。
【０１５９】
従って、一旦圧縮した後に解凍した解凍記事については、解凍から所定期間経過するまでは、アクセス可能性の判定は行われず、圧縮可とは判定されないことになる。
このため解凍してから所定期間の間は、アクセス要求があった場合に、必ず解凍処理を行うことなく配信が可能となる。
【０１６０】
なおステップＳ１５０の処理は、図７のようにアクセス要求の受信時に解凍された解凍記事であるか否かを判定することを前提としているが、アクセス要求の受信としては、ユーザ端末５からの閲覧のためのアクセス要求を受信する以外にも、例えばクローラによるアクセス要求を受信することもあり得る。ここでいうクローラとは、各種のウェブサイトにおけるテキストデータや画像データなどを周期的に取得して自動的にＤＢ化するプログラム（或いはそのプログラムに基づく動作を行う情報処理装置）のことである。このクローラによるアクセス要求は人気度や閲覧のためのアクセス可能性指数の変化には関与しない。
そこでクローラのアクセス要求に応じて解凍したような場合は、ステップＳ１５０で解凍記事と判断しないようにすることが望ましい。つまりその場合は、解凍直後であっても圧縮可否判断の対象とする。
或いはクローラによるアクセス要求に応じて解凍したような場合は、図８Ａのように解凍記事を保存しないようにすることも考えられる。
更には、クローラからのアクセス要求を受信した場合、圧縮記事をそもそも解凍しないようにするという手法も考えられる。
ここではクローラを例に挙げたが、一般の閲覧者たるユーザの意思と考えられるアクセス以外のアクセスは、以上のクローラの場合と同様の対応をとるとよい。
【０１６１】
＜１２．まとめ及び変形例＞
以上の実施の形態によれば次のような効果が得られる。第１～第９の実施の形態の情報処理装置としてのブログサーバ１は、１または複数の記事が含まれるブログが利用する記憶容量の容量（必要記憶容量）の増加傾向（例えば増加容量平均値などの増加傾向指数）を取得する傾向取得部１２と、増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値（ブログ圧縮判定閾値）を当該ブログに設定する閾値設定部１３と、一つのブログに含まれる記事の総データ量と閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合い（アクセス可能性指数）に応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定部１４と、を備えている。
これらの機能により図５，図９Ａ，図１３，図１４，図１８の処理で圧縮可否の判定を行うようにしている。
ブログの記憶リソースが圧迫されてきた場合に、ブログごとに一定の閾値を設けて記事の圧縮を行うことが考えられる。本構成によれば、一律同じ閾値を設けるのではなく、ブログが利用するブログＤＢ５１の記憶容量の増加傾向を示す指標（例えば増加容量平均値）に基づいて決定されたブログ圧縮判定閾値によって圧縮対象のブログとするか否かを決定することにより、適切なブログを圧縮対象とすることができる。
例えば、圧縮判定処理を１日に一度の割合で定期的に行う場合、必要記憶容量の増加傾向が高い（増加速度の速い）ブログは、圧縮判定処理の間に投稿される記事で当該ブログに割り当てられた総データ量上限値を超えてしまう可能性が相対的に高い。つまり、総データ量の増加傾向の高いブログは低いブログに対して空き容量に余裕が無い可能性が高いと言える。本実施の形態の処理によれば、ブログの総データ量の増加傾向を考慮した判定

ができ、これにより圧縮対象を適切に選択できる。
また、圧縮や、圧縮に対する解凍は、処理負荷が高いが、本実施の形態の場合、圧縮や解凍をなるべく行わないようにできる。
もし、圧縮した記事に対するアクセス要求があった場合、ブログサーバ１はその記事を解凍して配信することが考えられるが、記憶リソースのために圧縮する記事は、アクセスされる可能性が低いブログに属する記事となるため、アクセス時に解凍処理が必要となる事態を極力少なくできる。これによってブログ提供のためのブログサーバ１の処理負担を軽減できる。
またアクセス可能性が低い記事を選択して圧縮することで、圧縮処理の負担も軽減される。
以上から、本実施の形態によれば、記事の圧縮によって記憶リソースの圧迫を抑えるとともに、なるべく圧縮や解凍の処理機会が少なくなるように、圧縮する記事を適切に選択することができ、サーバの処理負担の低減や、閲覧時のパフォーマンス向上を図ることができる。

【０１６２】
第１～第９の実施の形態のブログサーバ１は、圧縮対象ブログに属する記事のうち圧縮対象記事の圧縮を行い、既に圧縮した記事に対するアクセス要求が発生した場合に当該記事の解凍を行う圧縮解凍部１５を備えている。
即ち圧縮解凍部１５の機能により、図６，図９Ｂ，図１２，図１５に示す処理で判定部１４の判定に沿って適切に選択された記事の圧縮を行う。またアクセスされる可能性が小さいとして圧縮された記事であっても、アクセス要求が発生することは当然あり得るが、そのような場合、図７の処理で解凍を行うことで、アクセス要求をした閲覧者に対して記事を適切に提供する。
これにより記憶リソースの圧迫防止や処理負担の削減のための適切な圧縮・解凍を行うことができる。

【０１６３】
第１～第９の実施の形態のブログサーバ１の圧縮解凍部１５は、第１の実施の形態で説明したように、圧縮を行う場合に圧縮対象とされた記事に含まれる画像データの圧縮を行ってもよい。
ブログによっては、画像データを多く含む記事が投稿されやすいものやテキストデータのみの記事が投稿されやすいものがある。画像データはテキストデータに対して容量が大きい場合が多く、テキストデータの圧縮率に対して画像データの圧縮率が低かったとしても圧縮効果、即ち圧縮した際に確保される空き容量が大きくなる可能性が高い。例えば、５ＫＢのテキストデータと３０ＭＢの画像データを含む記事の場合、テキストデータの圧縮ではどんなに圧縮率が高くても５ＫＢ以上の空き容量を確保することは不可能であるが、画像データの圧縮によって５ＫＢ以上の空き容量の確保することは十分可能である。
この構成によれば、効果の高い画像データの圧縮を行う機会を増やすことができる。また、テキストデータ及び画像データの双方を適切に圧縮することにより、圧縮効果を最大限に高めることができる。更に、画像データを圧縮対象とする場合には、圧縮の形式として非可逆圧縮を利用することができるため、効果の高い圧縮を行うことが可能である。

【０１６４】
第１～第９の実施の形態のブログサーバ１の圧縮解凍部１５は、圧縮を行う場合に圧縮対象とされた記事に含まれるデータのうち冒頭の所定のデータ以外のデータについて圧縮を行ってもよい。
ユーザ端末５上に一つの記事を表示した場合に、ユーザ端末５の表示部の大きさや表示文字の大きさ、或いは、記事の長さなどによっては、画面内に収まりきらない可能性がある。例えば、記事の前半部分が画面のファーストビューに表示され後半部分が画面外に位置する場合には、記事の後半部分は前半部分が読まれた後に閲覧される可能性が高い。本構成によれば、記事における冒頭の所定のデータは圧縮されない状態でブログＤＢ５１に記憶されているため、解凍処理を行わずに速やかに送信される。従って、閲覧者は記事の

前半部分をストレス無く閲覧することができる。そして、閲覧の間に記事の後半部分が解凍されてユーザ端末５に配信されるため、閲覧者は記事の後半部分も問題なく閲覧することができる。

記事に含まれるデータのうちどの程度を圧縮せずに記憶しておくかは、複数の態様が考えられる。例えば、圧縮部分の分量が多い場合には、解凍時間に通常よりも時間が掛かると想定して、解凍処理が終わって送信されるまでに一般的な閲覧者が閲覧し終わらないだけの分量を圧縮せずに記憶しておいてもよい。即ち、記事に含まれるデータ量に応じて圧縮せずに記憶するデータ量を決めてもよい。

また、複数の記事が一つのウェブページデータに含まれる状態で配信される場合には、複数の記事の圧縮していない部分が最初に配信されるようにしてもよい。この場合には、ユーザ端末５の画面上に、複数の記事の圧縮していない部分のみが最初に連なって表示され、各記事の残りの部分の解凍が終わって配信されると共に既に表示されている各記事の間に残りの部分が挿入されて表示される。これによれば、閲覧者は最初に複数の記事の冒頭部分を一つの画面で読むことができるため、興味のある記事を探しやすくなる可能性がある。

なお、複数の記事が一つのウェブページデータに含まれる状態で配信される場合において、例えば中程に位置する記事がファーストビューとしてユーザ端末５の画面に表示されるような場合には、冒頭部分が表示された記事の残りの部分の解凍及び配信から行うことが望ましい。これにより、閲覧者はファーストビューに表示されている閲覧中の記事の続きを速やかに読むことができる。

【０１６５】

第１～第９の実施の形態のブログサーバ１における圧縮解凍部１５は、圧縮を行う場合に圧縮対象とされた記事に含まれる画像データを圧縮する際に非可逆圧縮を行ってもよい。

閲覧される可能性の低い、即ちアクセス可能性が低い記事は、解凍される機会が少ないため、圧縮処理において画像の非可逆圧縮を用いることが望ましい。これにより、記憶リソースの確保を最大限行うことができる。

また、非可逆圧縮を用いて圧縮した画像データは、解凍しても未圧縮状態の画像データよりもデータ量が小さい。従って、ある記事がアクセス可能性が低いために非可逆圧縮を用いた圧縮処理が行われた場合には、その後に何らかの理由で注目されアクセス可能性が上昇して解凍されたとしても、必要記憶容量は未圧縮状態の記事よりも小さくなる。従って、記憶リソースの確保の点で有利である。

【０１６６】

特に、第２の実施の形態で説明したように、ブログサーバ１の圧縮解凍部１５は、圧縮対象とされた記事に含まれる画像データを可逆圧縮した後に当該記事が再度圧縮対象となった場合に当該記事に対して非可逆圧縮を行うように構成されていてもよい。即ち、非可逆圧縮を適用するまでの段階を一次圧縮と二次圧縮の２段階に分けてもよい。可逆圧縮を用いる一次圧縮により、画像が不用意に見にくくなってしまうことを防止すると共に、真にアクセス可能性の小さい記事に対しては二次圧縮としての非可逆圧縮を用いて記憶リソースの確保を最大限に行うことが可能となる。

【０１６７】

第３の実施の形態で説明したように、ブログサーバ１は、ブログに記事を投稿する投稿者の利用状況に応じて当該投稿者に許可する記憶容量を増減させるブログ管理部１１を備えていてもよい。

ブログ更新頻度が高いユーザは増加傾向指数が高くなる傾向にあるため、圧縮判定処理において圧縮対象ブログと判定される可能性が高く、圧縮処理によって記事が圧縮されやすい。換言すれば圧縮状態にある記事が多くなる。このような状況で更に記事が投稿され続けると、圧縮すべき記事が無くなり、ブログの総データ量が総データ量上限値を超えてしまう事態に陥る。そこでそれ以上の記事投稿を制限してしまうと、記述者にとってブログサービスを利用する魅力が無くなってしまうと共に、当該記述者のブログを楽しみにし

ている閲覧者にとっても好ましくない。
　このような状況においては、記述者（投稿者）のブログ利用状況に応じて、総データ量上限値を設定することが好ましい。そして、ブログに対する記述者の利用態様は、時間の経過と共に変化していくため、例えば定期的などに総データ量上限値を見直し再設定することで、ブログの記述者に応じた適切なサービスを提供することができる。
　なお、記述者だけでなく、閲覧者の利用状況も鑑みて総データ量上限値を設定することも考えられる。例えば記述者が亡くなってしまい新たな記事の投稿はされなくなったが、その記述者が投稿してきた記事を熱心に閲覧してきた閲覧者によるコメントの投稿が止まない場合などは、当該ブログの必要記憶容量が増加し続ける。このようなブログの利用状況（即ちコメントだけが投稿され続けるような利用状況）も考慮して総データ量上限値を設定し、記事が圧縮の対象とされにくくすることで、ブログの閲覧時に解凍処理を行う必要性を低下させ、閲覧時のパフォーマンス低下を感じさせることを抑制することができる。
【０１６８】
　第９の実施の形態のブログサーバ１の判定部１４は、アクセス要求に応じて既に圧縮した記事の解凍が行われた場合、解凍から所定期間、当該記事を圧縮対象記事と判定しない。
　第９の実施の形態では、圧縮された記事についてアクセスに応じて解凍が行われた場合、解凍から所定期間、当該記事を圧縮する記事と判定しないようにしている（図１８参照）。
　圧縮した後にアクセスがあって解凍して配信した場合、その記事についてはアクセス要求がなされる可能性が高まったと考えることができる。そこで、解凍したままとし、次のアクセス要求を受信した際に解凍処理を不要とする。
　但し、アクセスが単発的なものであって、引き続き不人気記事であることもある。そこで所定期間経過後は、ブログの増加傾向指数や記事ごとのアクセス可能性指数に基づいて適宜圧縮を行えばよい。これにより、解凍したままとしておくことが無駄な記憶リソース消費となっている場合に対処でき、記憶に必要な容量を低減させ、記憶リソースの圧迫を回避することができる。
【０１６９】
　第７の実施の形態のブログサーバ１の判定部１４は、既に圧縮した記事について、同一の圧縮対象ブログの他の記事に対するページビューの増加傾向を示す値に応じた解凍の是非を判定し、圧縮解凍部１５は解凍の是非に応じて圧縮対象記事を解凍しておく。
　第７の実施の形態では、ブログの人気度指数の変化を監視し、或るブログについて人気度指数の上昇傾向を検知した場合に、該ブログに含まれる圧縮済の記事を解凍する記事と判定するようにしている（図１４参照）。
　或るブログが何らかの人気記事によってアクセス数が顕著に上昇したような場合、そのブログに含まれる別の記事は、これまでアクセスされていなかったとしても、今後アクセスされる可能性が高まる。そこで当該ブログ内の全ての圧縮記事を解凍する記事と判定する。
　このようにすれば、実際にアクセスがあった際に、その時点での解凍処理は不要となり、レスポンスよく記事を配信することができる。
【０１７０】
　なお第７の実施の形態の図１４の処理の変形例として、人気上昇と判定されたブログにおける全ての圧縮記事ではなく、一部の圧縮記事を解凍する記事と判定してもよい。例えば投稿日時が現在から過去に所定期間内の記事を解凍する記事とすることで、むやみに多数の記事を解凍せず、記憶リソースの維持に好適である。例えば長期にわたって多数の記事が投稿されており、比較的古い記事の多数が圧縮されているようなブログについては、全部ではなく一部の圧縮記事を解凍対象とすることが好ましい。
　或いは、圧縮記事の数が所定数以上の場合は、一部（例えば半分）のみを解凍対象とすることも考えられる。

また解凍する記事数の上限を決め、上限数以上の圧縮記事が存在する場合は、投稿日時の新しい記事から上限数の記事を解凍する記事と判定してもよい。

【０１７１】

第８の実施の形態のブログサーバ１の判定部１４は、記事内容に基づいて、既に圧縮した記事を解凍するか否かを判定する。

第８の実施の形態では、記事内容に基づいて、既に圧縮した記事を解凍するか否かを判定するようにしている（図１６参照）。

例えば圧縮した記事の内容として、或る設定したキーワードや時事語を含む記事、特定のテーマの記事などを抽出し、これらを解凍対象とする。

例えば、あるテーマについて言及したブログ記事について、当初にアップロードされた時点ではあまり話題にならなかったものの、数年後に、その記事で扱ったテーマに関する何らかの事件や事象が世の中で起きたために、その記事が掘り起こされる可能性が高まる（つまり、アクセス可能性が高まる）ことや、その記事に対する被リンク数が増加するといったことが起きうる。

そこで、このような動きを記事内容、具体的には時事語、設定したキーワード、記事のテーマ等により事前に察知して、既に圧縮したもののアクセス可能性の高まった記事については、圧縮状態を解凍しておく。

このようにすれば、実際にアクセスがあった際に、その時点での解凍処理を不要とし、すぐに記事を配信することができる。

【０１７２】

なお、圧縮した記事については、内容を検索することが困難にもなる。そこで、管理ＤＢ５３に圧縮記事タグとしてキーワードやテーマ等を示すタグ情報を記憶しておくようにしている。これにより、解凍判定を容易かつ適切に行うことができる。

またタグ設定及び管理ＤＢ５３へのタグ登録は、圧縮時に行っている（図６，図９Ｂ，図１２，図１５参照）。これによりタグ設定及び登録の処理の実行を必要最小限にでき、処理負担の増大を抑止し、またＤＢにおける記憶リソースをむやみに圧迫しないようにできる。

【０１７３】

上記した実施の形態の処理例は一例であり、他にも各種の変形例が想定される。

ブログの増加傾向指数について例えば３段階以上の多段階に分けた場合、最も増加傾向指数が高い低いランクであると判定されたブログについては、全記事を圧縮可と判定するようなことも考えられる。つまり増加傾向指数のｎ段階の第１レベル（増加傾向指数が小）から第（ｎ－１）レベル（増加傾向指数が大）については増加傾向指数及びアクセス可能性指数を用いて記事ごとの圧縮可否判定を行うが、第ｎレベル（増加傾向指数が最大）のブログは、個々の記事を判定することなく全記事を圧縮可とするような例である。

特に増加傾向指数は高いが閲覧者のアクセス（閲覧やコメント書き込み）が低いようなブログであり、自分のためだけの記録を綴っているようなブログは、閲覧者の利便性を阻害する可能性も低いことから、個々の記事を判定することなく全記事を圧縮可と判定することが望ましい。

これによりブログサーバ１の処理効率の向上、処理負担の削減、記憶リソースの確保を促進できる。

【０１７４】

圧縮処理については段階的に複数回行うようにしてもよい。

例えば図１３のように圧縮記事についても引き続きアクセス可能性の判定を行うようにする。そして圧縮済であるが、更に所定期間、アクセスがない記事は、より高い圧縮率で圧縮するなどである。

例えば１回目の圧縮では圧縮率２０％の圧縮、２回目は圧縮率５０％の圧縮、３回目は圧縮率８０％の圧縮というようにする。

また、１回目は可逆圧縮、２回目は非可逆圧縮を行うというような例もある。

また、１回目は記事の一部圧縮、２回目は記事の全体圧縮としてもよい。

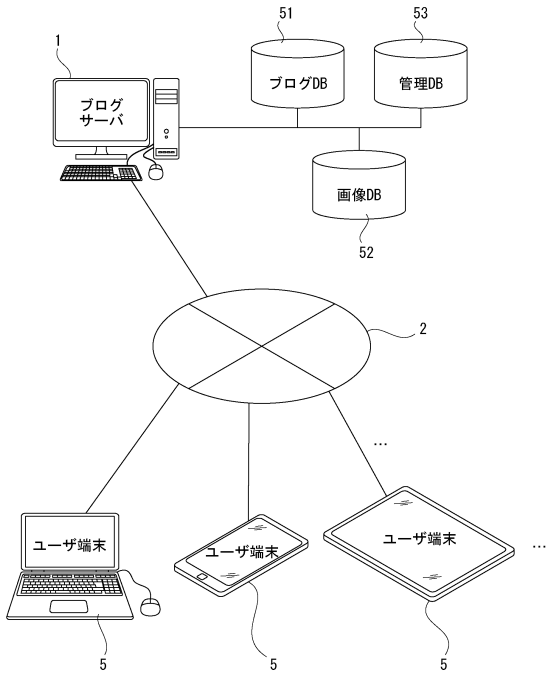
また、１回目は記事のテキストのみ圧縮、２回目は加えて画像も圧縮としてもよい。
また、１回目は記事の画像のみ圧縮、２回目は加えてテキストも圧縮としてもよい。
【０１７５】
圧縮判定処理の対象について、一部のブログを除外することも考えられる。
例えば長期にわたって高い人気と判定されるブログについては、そのブログに含まれる記事については圧縮判定の対象外としてしまうことが考えられる。これにより図５，図９Ａ，図１３，図１４，図１８等の処理の対象とするブログの数を削減でき、処理を効率化できる。
同様に、記事全体を圧縮済のもの、特に上述のように非常に増加傾向指数が高く、閲覧者もいないようなブログは、解凍判定の対象外としてしまうことで、図１４，図１６等の処理の対象とするブログの数を削減でき効率化が図られる。
【０１７６】
なお、実施の形態はいわゆるブログとブログに含まれる記事を対象とし、記事の圧縮を行う処理について説明したが、このような技術は、ファイルシステムにおけるフォルダと、フォルダに含まれるファイルについても適用できる。
つまりフォルダの増加傾向指数は、フォルダ内のファイルを記憶しておくために必要とされる記憶リソースの増加傾向を示す値となり、それを反映させて、ファイルのアクセス可能性を基準にファイルの圧縮可否を判断するものである。
【０１７７】
またブログは、いわゆるクラウドストレージとして実現されるシステムであってもよい。
【０１７８】
＜１３．プログラム及び記憶媒体＞
実施の形態のプログラムは、ブログサーバ１における少なくとも傾向取得部１２、閾値設定部１３、判定部１４の処理を情報処理装置（ＣＰＵ等）に実行させるプログラムである。
【０１７９】
実施の形態のプログラムは、１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得機能と、前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定機能と、一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定機能と、を情報処理装置に実行させるプログラムである。
即ちこのプログラムは、情報処理装置に対して図５，図９Ａ，図１３，図１４，図１８等で説明した処理を実行させるプログラムである。
【０１８０】
このようなプログラムにより、上述したブログサーバ１としての１または複数の情報処理装置を実現できる。
そしてこのようなプログラムはコンピュータ装置等の機器に内蔵されている記憶媒体としてのＨＤＤや、ＣＰＵを有するマイクロコンピュータ内のＲＯＭ等に予め記憶しておくことができる。あるいはまた、半導体メモリ、メモリカード、光ディスク、光磁気ディスク、磁気ディスクなどのリムーバブル記憶媒体に、一時的あるいは永続的に格納（記憶）しておくことができる。またこのようなリムーバブル記憶媒体は、いわゆるパッケージソフトウェアとして提供することができる。
また、このようなプログラムは、リムーバブル記憶媒体からパーソナルコンピュータ等にインストールする他、ダウンロードサイトから、ＬＡＮ、インターネットなどのネットワークを介してダウンロードすることもできる。
【符号の説明】
【０１８１】

１　ブログサーバ、２　ネットワーク、５　ユーザ端末、１１　ブログ管理部、１２　傾向取得部、１３　閾値設定部、１４　判定部、１５　圧縮解凍部、５１　ブログＤＢ、５２　画像ＤＢ、５３　管理ＤＢ

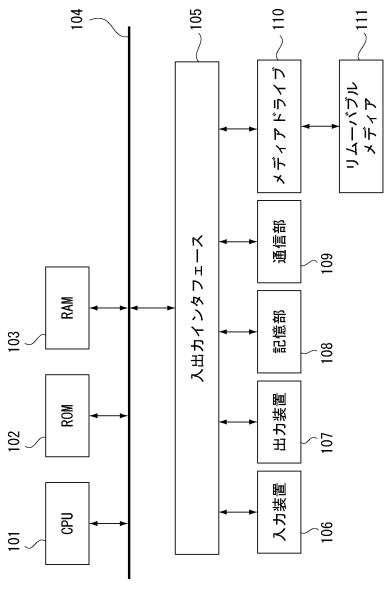
【要約】

ブログサービスを提供するサーバにおいてブログの数やサイズの増大によりメモリリソースが徐々に圧迫されていくため、このメモリ負担を低減する。そこで１または複数の記事が含まれるブログが利用する記憶容量の増加傾向を取得する傾向取得部と、前記増加傾向に応じてブログに含まれる記事の少なくとも一部を圧縮するか否かを判定するための閾値を当該ブログに設定する閾値設定部と、一つの前記ブログに含まれる記事の総データ量と前記閾値とに基づいて当該ブログを圧縮対象とするか否かを判定し、アクセス可能性の度合いに応じて当該ブログに含まれる記事ごとに圧縮対象とするか否かを判定する判定部と、を備えるものとする。

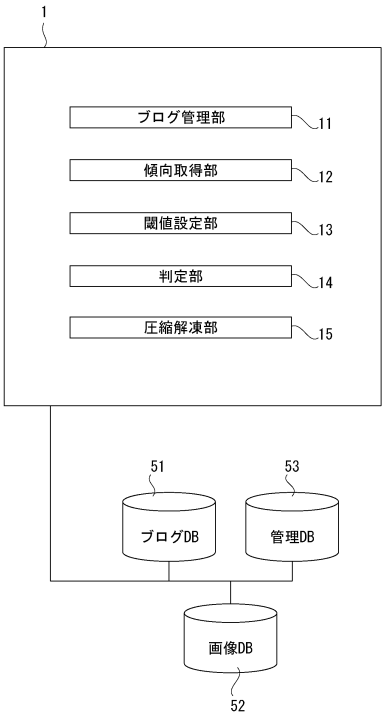
【図１】



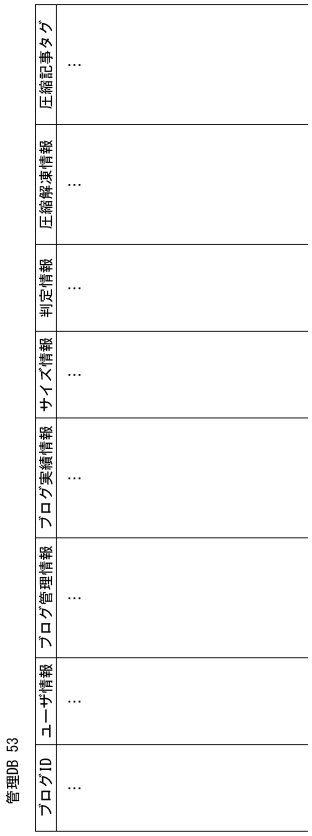
【図２】



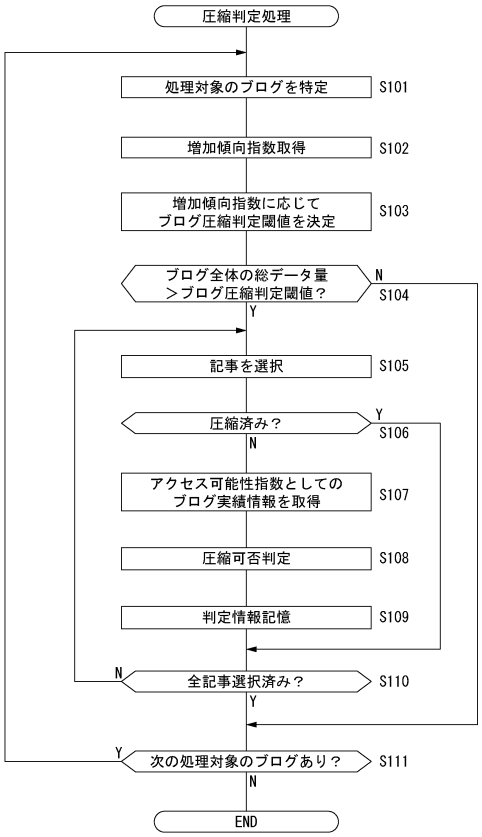
【図３】


1
ブログ管理部
11
傾向取得部
12
閾値設定部
13
判定部
14
圧縮解凍部
15
51
53
ブログDB
管理DB
画像DB
52

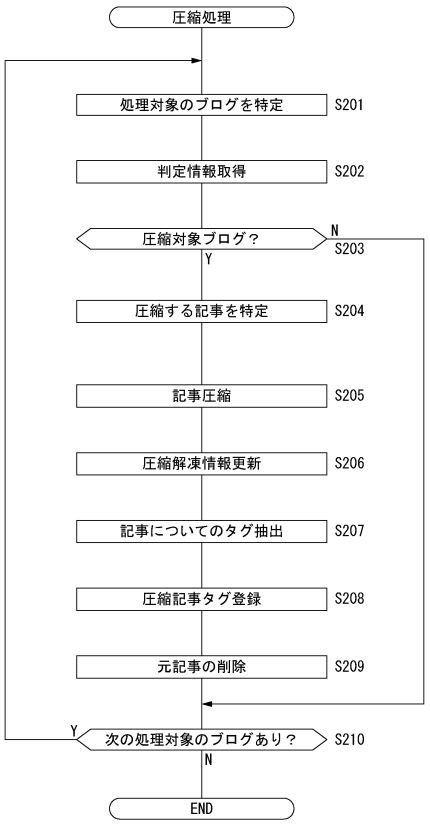
【図４】


管理DB 53
ブログID
ユーザ情報
ブログ管理情報
ブログ実績情報
サイズ情報
判定情報
圧縮解凍情報
圧縮記事タグ
…

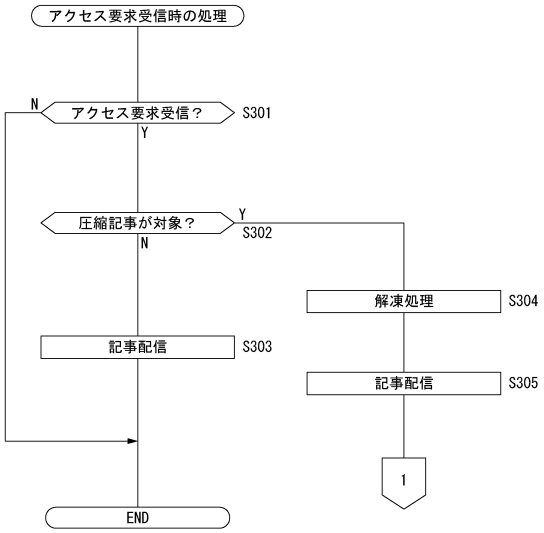
【図５】


圧縮判定処理
処理対象のブログを特定
S101
増加傾向指数取得
S102
増加傾向指数に応じて
ブログ圧縮判定閾値を決定
S103
ブログ全体の総データ量
＞ブログ圧縮判定閾値？
S104
N
Y
記事を選択
S105
圧縮済み？
S106
Y
N
アクセス可能性指数としての
ブログ実績情報を取得
S107
圧縮可否判定
S108
判定情報記憶
S109
全記事選択済み？
S110
N
Y
次の処理対象のブログあり？
S111
Y
N
END

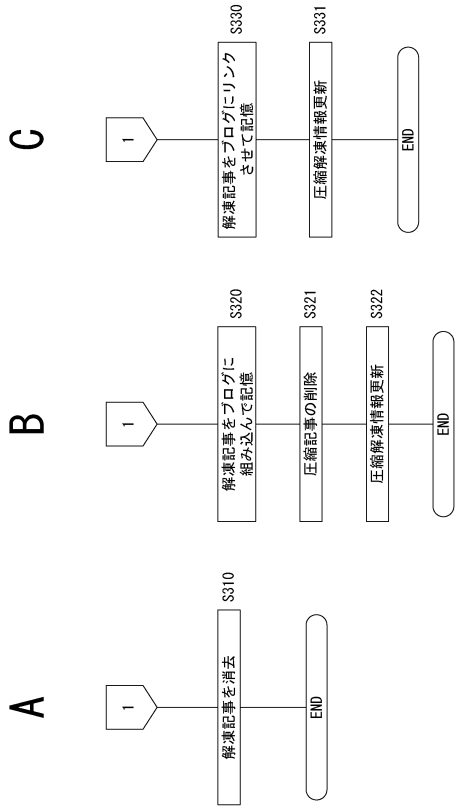
【図６】


圧縮処理
処理対象のブログを特定
S201
判定情報取得
S202
圧縮対象ブログ？
S203
N
Y
圧縮する記事を特定
S204
記事圧縮
S205
圧縮解凍情報更新
S206
記事についてのタグ抽出
S207
圧縮記事タグ登録
S208
元記事の削除
S209
次の処理対象のブログあり？
S210
Y
N
END

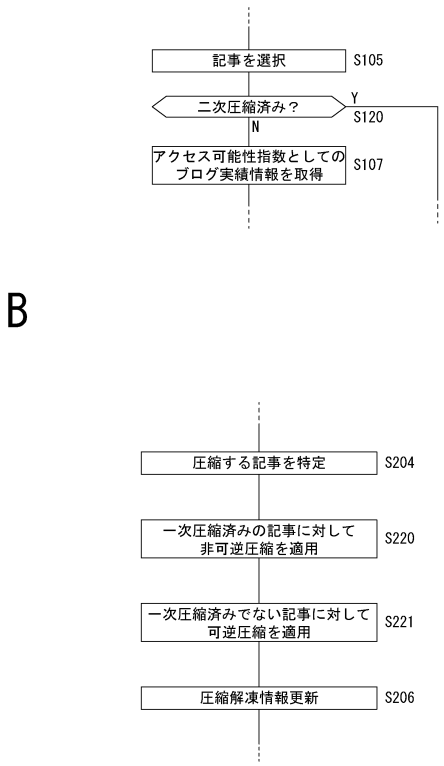
【図７】


アクセス要求受信時の処理
アクセス要求受信？
S301
N
Y
圧縮記事が対象？
Y
S302
N
解凍処理
S304
記事配信
S303
記事配信
S305
1
END

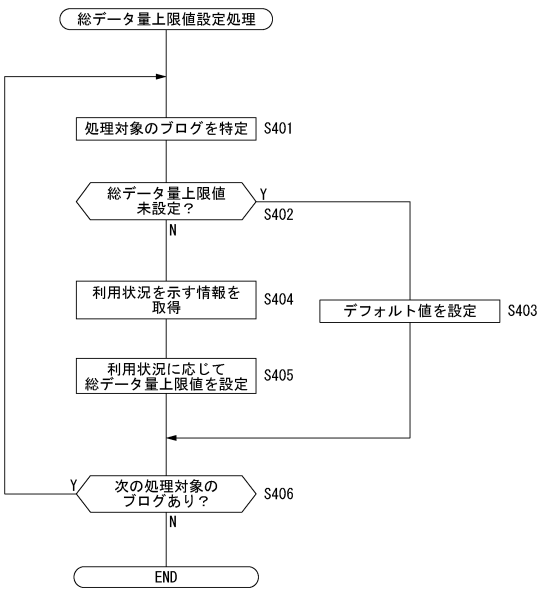
【図８】


C
1
解凍記事をブログにリンクさせて記憶
S330
圧縮解凍情報更新
S331
END
B
1
解凍記事をブログに組み込んで記憶
S320
圧縮記事の削除
S321
圧縮解凍情報更新
S322
END
A
1
解凍記事を消去
S310
END

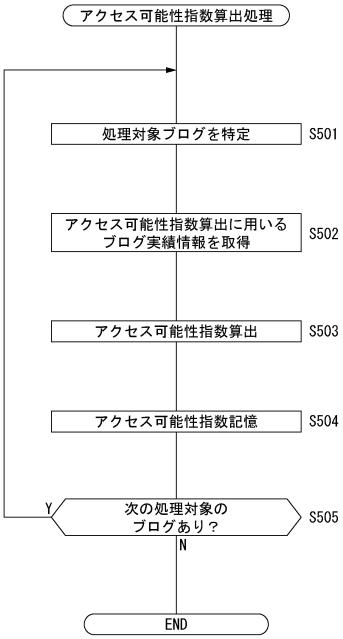
【図９】


A
記事を選択
S105
二次圧縮済み？
Y
S120
N
アクセス可能性指数としてのブログ実績情報を取得
S107
B
圧縮する記事を特定
S204
一次圧縮済みの記事に対して非可逆圧縮を適用
S220
一次圧縮済みでない記事に対して可逆圧縮を適用
S221
圧縮解凍情報更新
S206

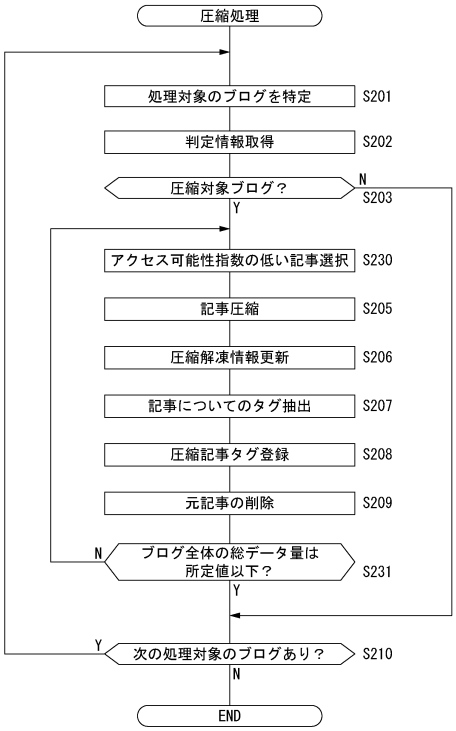
【図１０】


総データ量上限値設定処理
処理対象のブログを特定
S401
総データ量上限値未設定？
Y
S402
N
利用状況を示す情報を取得
S404
デフォルト値を設定
S403
利用状況に応じて総データ量上限値を設定
S405
Y
次の処理対象のブログあり？
S406
N
END

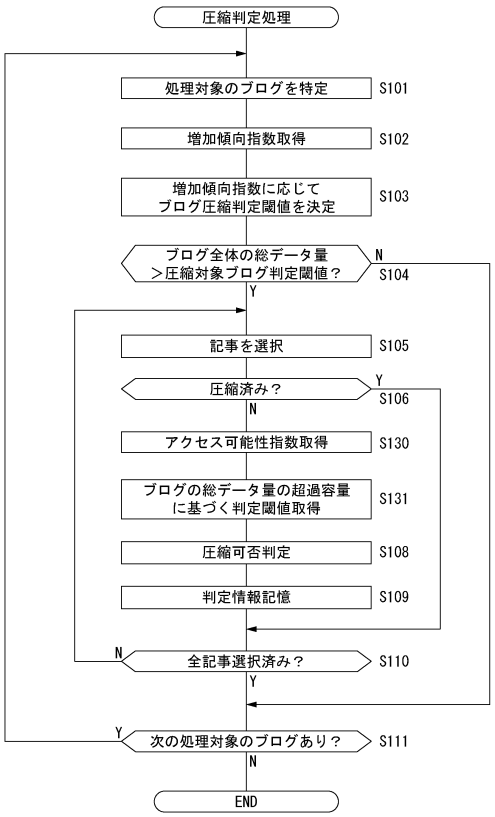
【図１１】



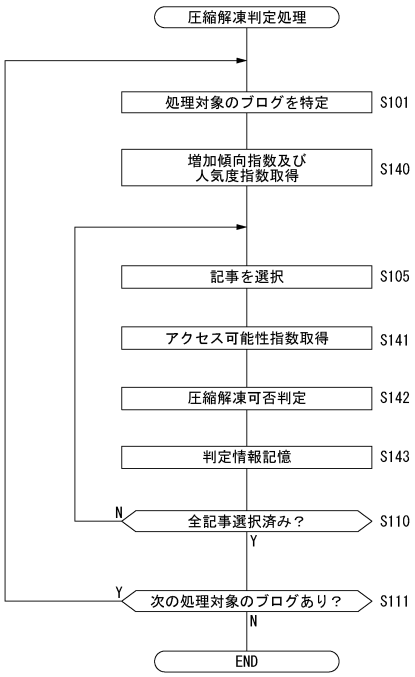
【図１２】



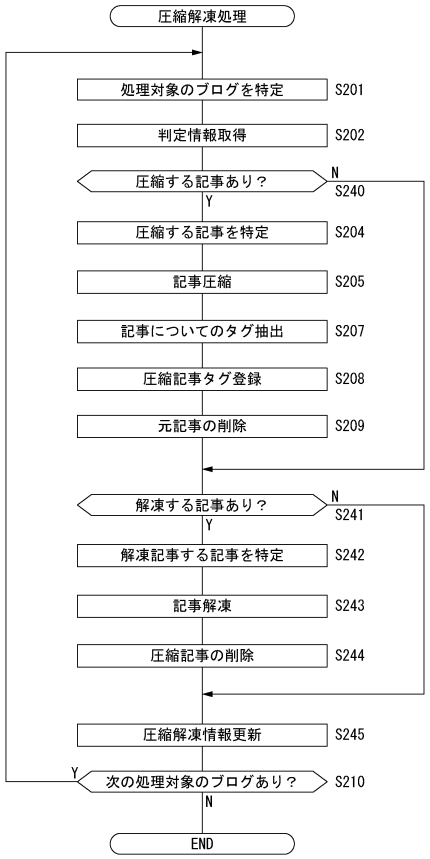
【図１３】



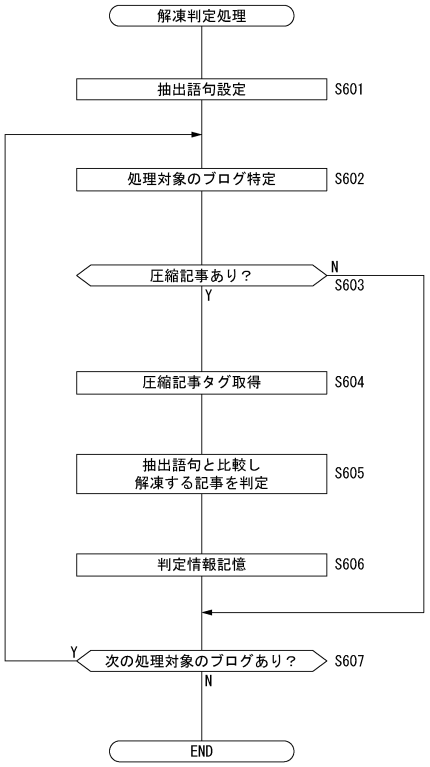
【図１４】



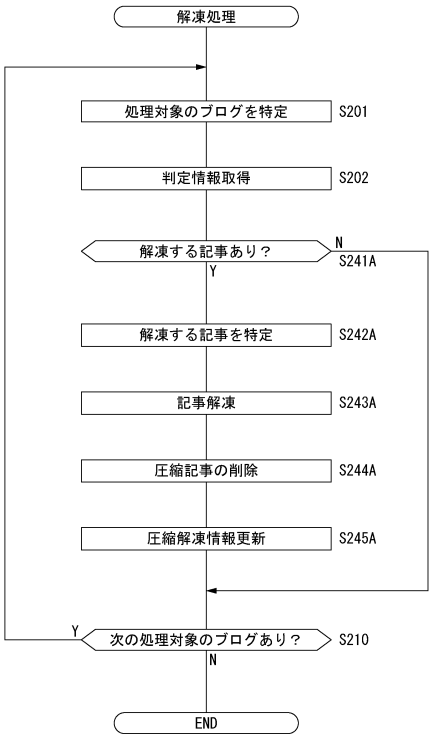
【図１５】



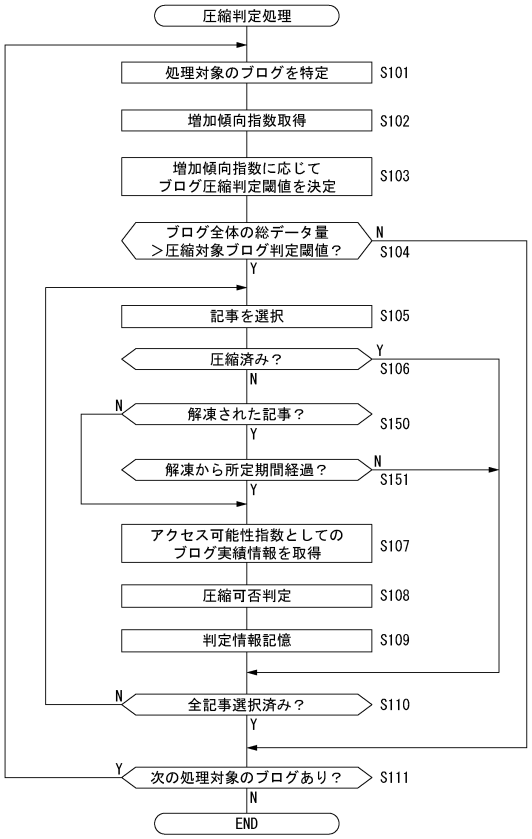
【図１６】



【図１７】



【図１８】



フロントページの続き

(56)参考文献 特開２００９－０７０３６１（ＪＰ，Ａ）
米国特許出願公開第２０１１／０１３８２７０（ＵＳ，Ａ１）
特開２０１２－２２２５０４（ＪＰ，Ａ）
特開２０１２－０１４２３６（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ １２／００