(19)日本国特許庁(JP)　(12)公開特許公報(A)　(11)特許出願公開番号
特開2005-276192
(P2005-276192A)
(43)公開日　平成17年10月6日(2005.10.6)

| (51)Int.Cl.[7] | F I | | | テーマコード（参考） |
|---|---|---|---|---|
| G06F 12/00 | G06F | 12/00 | 501A | 5B075 |
| G06F 17/30 | G06F | 17/30 | 110C | 5B082 |
| | G06F | 17/30 | 110F | |

審査請求 有　請求項の数 30　ＯＬ　外国語出願　（全 59 頁）

(21)出願番号　特願2005-55114(P2005-55114)
(22)出願日　平成17年2月28日(2005.2.28)
(31)優先権主張番号　10/787337
(32)優先日　平成16年2月26日(2004.2.26)
(33)優先権主張国　米国(US)

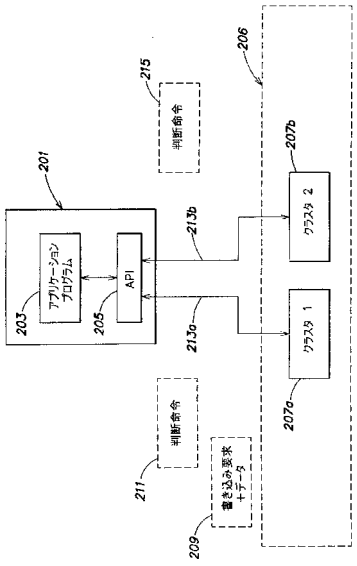
（特許庁注：以下のものは登録商標）
１．フロッピー

(71)出願人　503093224
イーエムシー　コーポレイション
ＥＭＣ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　０１７４８　マサチューセッツ州　ホプキントン　サウス　ストリート　１７６
(74)代理人　100068755
弁理士　恩田　博宣
(74)代理人　100105957
弁理士　恩田　誠
(72)発明者　スティーブン　トッド
アメリカ合衆国　０１５４５　マサチューセッツ州　シュルーズベリー　トーマス　ファーム　サークル　３６

最終頁に続く

(54)【発明の名称】データ記憶容量を増大させるための方法及び装置

(57)【要約】
【課題】　データ記憶環境の記憶容量を増大させるための装置及び方法を提供する。
【解決手段】　各記憶クラスタの性能に影響を与えることなく、さらなる記憶クラスタが記憶環境に追加され得る。データが記憶環境に書き込まれる場合は、データを記憶する記憶クラスタの選択が行われ得る。データが記憶環境から読み出される場合は、どの記憶クラスタにそのデータは記憶されているのかの判断が行われ、その記憶クラスタからデータは取得され得る。
【選択図】　図２



【特許請求の範囲】
【請求項１】
コンピュータ・システム上で実行された場合に、複数の記憶クラスタを含む記憶環境に記憶されたデータのユニットにアクセスする方法を実行する命令がエンコードされた少なくとも一つのコンピュータ可読媒体であって、
前記コンピュータ・システムは、前記記憶環境と、前記記憶環境上にデータを記憶するホスト・コンピュータとを備え、
前記方法は、
前記記憶環境に以前に記憶されたデータのユニットを確認するために、前記ホスト・コンピュータからの要求を受信する動作と、
前記要求の受信に応答して、前記データのユニットが前記複数の記憶クラスタのうちのどの一つの記憶クラスタ上に記憶されているのかを判断する動作とを備えた、少なくとも一つのコンピュータ可読媒体。
【請求項２】
請求項１に記載の少なくとも一つのコンピュータ可読媒体において、前記ホスト・コンピュータは、前記記憶環境にデータ記憶し、かつ前記記憶環境からデータを取得するアプリケーション・プログラムを実行し、前記ホストは、さらに、前記アプリケーション・プログラムを前記記憶環境で機能するようにするアプリケーション・プログラミング・インターフェイスを実行し、前記受信動作は、該アプリケーション・プログラミング・インターフェイスにより実行される、少なくとも一つのコンピュータ可読媒体。
【請求項３】
請求項１に記載の少なくとも一つのコンピュータ可読媒体において、前記コンピュータ・システムは、さらに、前記記憶環境を前記ホスト・コンピュータに接続する少なくとも一つの通信リンクと、前記記憶環境に対する前記ホスト・コンピュータからのアクセス要求を監視するアプライアンスとを備え、前記要求受信動作は、さらに、前記アプリケーションからの前記要求を、該アプライアンスにおいて受信する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項４】
請求項３に記載の少なくとも一つのコンピュータ可読媒体において、前記検出動作は前記アプライアンスによって実行される、少なくとも一つのコンピュータ可読媒体。
【請求項５】
請求項３に記載の少なくとも一つのコンピュータ可読媒体において、前記要求の受信動作は、さらに、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタにおいて前記要求を受信する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項６】
請求項５に記載の少なくとも一つのコンピュータ可読媒体において、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタは、アクセス要求を受信し、かつ処理する少なくとも一つのアクセス・ノードを含み、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタおける前記アプリケーション・プログラムからの要求受信動作は、さらに、前記少なくとも一つのアクセス・ノードにおいて前記要求を受信する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項７】
請求項１に記載の少なくとも一つのコンピュータ可読媒体において、前記判断動作は、前記複数の記憶クラスタ上の前記データのユニットの検索を実行する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項８】
請求項１に記載の少なくとも一つのコンピュータ可読媒体において、前記判断動作は、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタによって実行される、少なくとも一つのコンピュータ可読媒体。
【請求項９】

請求項１に記載の少なくとも一つのコンピュータ可読媒体において、前記判断動作は、検索を実行することなく、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタ上の前記データのユニットを確認することを備える、少なくとも一つのコンピュータ可読媒体。

【請求項１０】

請求項１に記載の少なくとも一つのコンピュータ可読媒体において、前記記憶環境は、内容アドレス記憶装置の環境であって、該環境において記憶されたデータのユニットは、該データのユニットの内容の少なくとも一部に少なくとも部分的には基づく内容アドレスによりアクセス可能であり、前記判断動作は、さらに、前記データのユニットの前記内容アドレスに基づいて、前記データのユニットが前記複数の記憶クラスタのうちのどの記憶クラスタ上に記憶されているのかを判断することを備える、少なくとも一つのコンピュータ可読媒体。

【請求項１１】

請求項１０に記載の少なくとも一つのコンピュータ可読媒体において、前記内容アドレスは、前記データのユニットが前記記憶環境に記憶された時間に基づく時間情報を含み、前記判断動作は、前記データのユニットの前記内容アドレスの前記時間情報に少なくとも部分的には基づいて、前記データのユニットが前記複数の記憶クラスタのうちのどの記憶クラスタ上に記憶されているのかを判断する動作を備える、少なくとも一つのコンピュータ可読媒体。

【請求項１２】

請求項１１に記載の少なくとも一つのコンピュータ可読媒体において、前記判断動作は、さらに、前記データのユニットの前記内容アドレスの前記時間情報のハッシュ値に少なくとも部分的には基づいて、前記データのユニットが前記複数の記憶クラスタのうちのどの記憶クラスタ上に記憶されているのかを判断する動作を備える、少なくとも一つのコンピュータ可読媒体。

【請求項１３】

請求項１０に記載の少なくとも一つのコンピュータ可読媒体において、前記判断動作は、さらに、

前記データのユニットの前記内容アドレスに基づいて、前記複数の記憶クラスタのうちのどの記憶クラスタ上に前記データのユニットを記憶するのかの選択に利用されたアルゴリズムを特定する情報にアクセスする動作と、

前記データのユニットが前記複数の記憶クラスタのうちのどの記憶クラスタ上に記憶されているのかを判断するために、前記アルゴリズムを前記データのユニットの前記内容アドレスに適用する動作とを備える、少なくとも一つのコンピュータ可読媒体。

【請求項１４】

請求項１３に記載の少なくとも一つのコンピュータ可読媒体において、前記情報は、前記記憶環境によって利用された複数のアルゴリズムを特定し、かつ、データのユニットを記憶するために、該複数のアルゴリズムのそれぞれがどの程度の期間利用されたのかを特定する、少なくとも一つのコンピュータ可読媒体。

【請求項１５】

請求項１４に記載の少なくとも一つのコンピュータ可読媒体において、前記情報は、さらに、前記複数のアルゴリズムのそれぞれについて、前記複数のアルゴリズムのうちの一つが有効であった場合、前記期間に前記記憶環境に存在した少なくとも一つの記憶クラスタを特定する、少なくとも一つのコンピュータ可読媒体。

【請求項１６】

コンピュータ・システム上で実行された場合に、複数の記憶クラスタを含む記憶環境に記憶されたデータのユニットにアクセスする方法を実行する命令がエンコードされた少なくとも一つのコンピュータ可読媒体であって、

前記コンピュータ・システムは、前記記憶環境と、前記記憶環境上にデータを記憶するホスト・コンピュータとを備え、

前記方法は、
データのユニットを記憶するために、前記ホスト・コンピュータ上で実行されるアプリケーション・プログラムからの要求を受信する動作と、
前記要求の受信に応答して、前記データのユニットを記憶するために、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作であって、前記複数の記憶クラスタから選択された一つの記憶クラスタが前記データのユニットを記憶したことを前記アプリケーション・プログラムが認識しない、つまり、前記アプリケーション・プログラムにとって透過的であるような選択動作とを備えた、少なくとも一つのコンピュータ可読媒体。
【請求項１７】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記ホストは、さらに、前記アプリケーション・プログラムを前記記憶環境で機能するようにするアプリケーション・プログラミング・インターフェイスを実行し、前記受信動作は、該アプリケーション・プログラミング・インターフェイスにより実行される、
少なくとも一つのコンピュータ可読媒体。
【請求項１８】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記コンピュータ・システムは、さらに、前記記憶環境を前記ホスト・コンピュータに接続する少なくとも一つの通信リンクと、前記記憶環境に対する前記ホスト・コンピュータからのアクセス要求を監視するアプライアンスとを備え、前記要求受信動作は、さらに、前記アプリケーションからの前記要求を、該アプライアンスにおいて受信する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項１９】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記要求受信動作は、さらに、前記アプリケーション・プログラムから直接に前記要求を受信する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２０】
請求項１９に記載の少なくとも一つのコンピュータ可読媒体において、前記アプリケーション・プログラムからの前記要求の受信動作は、さらに、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタにおいて前記要求を受信する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２１】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、前記データのユニットを記憶するために、ラウンド・ロビン技術を用いて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２２】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、前記データのユニットを記憶するために、前記複数の記憶クラスタのそれぞれの利用可能な記憶容量に少なくとも部分的には基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２３】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、前記データのユニットを記憶するために、前記複数の記憶クラスタのうちの少なくとも一つの記憶クラスタの負荷に基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、
少なくとも一つのコンピュータ可読媒体。
【請求項２４】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、前記データのユニットを記憶するために、前記データのユニットのサイズに基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、

少なくとも一つのコンピュータ可読媒体。
【請求項２５】
請求項１６に記載の少なくとも一つのコンピュータ可読媒体において、前記記憶環境は、内容アドレス記憶装置の環境であって、該環境において記憶されたデータのユニットは、該データのユニットの内容の少なくとも一部に少なくとも部分的には基づく内容アドレスによりアクセス可能であり、前記選択動作は、さらに、前記データのユニットを記憶するために、前記データのユニットの前記内容アドレスに基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２６】
請求項２５に記載の少なくとも一つのコンピュータ可読媒体において、前記内容アドレスは、前記データのユニットが前記記憶環境に記憶された時間に基づく時間情報を含み、前記選択動作は、前記データのユニットを記憶するために、前記データのユニットの前記内容アドレスの前記時間情報に少なくとも部分的には基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２７】
請求項２６に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、前記データのユニットを記憶するために、前記データのユニットの前記内容アドレスの前記時間情報のハッシュ値に少なくとも部分的には基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２８】
請求項２７に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、前記データのユニットを記憶するために、ＧＵＩＤのハッシュに少なくとも部分的には基づいて、前記複数の記憶クラスタから一つの記憶クラスタを選択する動作を備える、少なくとも一つのコンピュータ可読媒体。
【請求項２９】
請求項２５に記載の少なくとも一つのコンピュータ可読媒体において、前記選択動作は、さらに、
前記データのユニットが前記複数の記憶クラスタのうちのどの記憶クラスタ上に記憶されているのかを判断するために、アルゴリズムを前記データのユニットの前記内容アドレスに適用する動作と、
前記アルゴリズムを、前記アルゴリズムが利用されていたタイム・フレームを示すレコードに記憶する動作とを備える、少なくとも一つのコンピュータ可読媒体。
【請求項３０】
請求項２９に記載の少なくとも一つのコンピュータ可読媒体において、前記レコードは、前記記憶環境によって利用された複数のアルゴリズムを特定し、かつ、データのユニットを記憶するために、該複数のアルゴリズムのそれぞれがどの程度の期間利用されたのかを特定する、少なくとも一つのコンピュータ可読媒体。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明は、データ記憶に関し、より詳細には、データ記憶容量を増大させるための方法及び装置に関する。
【背景技術】
【０００２】
データ記憶システムの容量及び性能は、その記憶システムの物理的リソースに依存している。例えば、記憶システムが記憶可能なデータ量は、その記憶システムが保有する物理的記憶デバイスの個数及び容量に依存している。記憶システムに記憶されたデータ量が、

その記憶システムの記憶容量に近づくと、その記憶システムにさらなる物理的な記憶デバイスを追加することにより、記憶システムの容量を増大させることが望ましい。

【発明の開示】

【発明が解決しようとする課題】

【０００３】

しかしながら、記憶システムのハードウェアの構成によって、記憶システムが有し得る記憶デバイスの個数には物理的な制限が課せられる場合がある。結果として、記憶システムのデータ量が、その記憶容量に近づくか、若しくは、殆どその記憶容量である時に、その記憶システムにさらなる物理的記憶デバイスを追加することはもはや不可能であるか、若しくは、望ましくないという場合がある。寧ろ、記憶容量を増大させたい場合には、一つ以上の追加の記憶システムを利用する方がよい。

【課題を解決するための手段】

【０００４】

一つの例示の実施形態は、複数の記憶クラスタを含む記憶環境に記憶されたデータのユニットにアクセスする方法を対象としている。その方法は、記憶環境に以前に記憶されたデータのユニットを確認するためにホスト・コンピュータからの要求を受信する動作と、その要求の受信に応答して、そのデータのユニットが複数の記憶クラスタのうちのどの一つの記憶クラスタ上に記憶されているのかを判断する動作と、を備える。別の例示の実施形態は、コンピュータ・システム上で実行された場合に上述の方法を実行する命令がエンコードされた少なくとも一つのコンピュータ可読媒体を対象としている。

【０００５】

さらなる例示の実施形態は、データを記憶するための装置を対象としている。その装置は、記憶環境における複数の記憶クラスタのうちの少なくとも一つ記憶クラスタ上のデータのユニットを確認するために、ホスト・コンピュータからの要求を受信する入力と、その入力に接続され、かつその入力からの要求を受信し、その要求の受信に応答して、データのユニットが複数の記憶クラスタのうちのどの記憶クラスタ上に記憶されているのかを判断する少なくとも一つの制御装置と、を備える。

【０００６】

別の例示の実施形態は、複数の記憶クラスタを含み、かつホスト・コンピュータのデータを記憶する記憶環境に記憶されたデータのユニットにアクセスする方法を対象とする。その方法は、データのユニットを記憶するための要求を、ホスト・コンピュータ上で実行されるアプリケーション・プログラムから受信する動作と、その要求の受信に応答して、データのユニットを記憶するために、複数の記憶クラスタから一つの記憶クラスタを選択する動作であって、その複数の記憶クラスタから選択された一つの記憶クラスタがそのデータのユニットを記憶したことをアプリケーション・プログラムが認識しない、つまり、アプリケーション・プログラムにとって透過的であるような選択動作と、を備える。さらなる例示の実施形態は、コンピュータ・システム上で実行された場合に上述の方法を実行する命令がエンコードされた少なくとも一つのコンピュータ可読媒体を対象としている。

【０００７】

さらなる例示の実施形態は、データを記憶する装置を対象としている。その装置は、記憶環境における複数の記憶クラスタのうちの少なくとも一つの記憶クラスタ上にデータのユニットを記憶するために、ホスト・コンピュータ上で実行されるアプリケーション・プログラムからの要求を受信する入力と、その入力に接続され、その入力からの要求を受信し、かつその要求の受信に応答して、データのユニットを記憶するために、複数の記憶クラスタから一つの記憶クラスタを選択する少なくとも一つの制御装置であって、複数の記憶クラスタから選択された一つの記憶クラスタがそのデータのユニットを記憶したことをアプリケーション・プログラムが認識しない、つまり、アプリケーション・プログラムにとって透過的であるような少なくとも一つの制御装置と、を備える。

【０００８】

別の例示の実施形態は、複数の記憶クラスタを含む記憶環境に記憶されたデータのユニ

ットにアクセスする方法を対象としている。その方法は、データのユニットを記憶するために、ホスト・コンピュータ上で実行されるアプリケーション・プログラムからの要求を受信する動作と、そのデータのユニットを記憶するために、複数の記憶クラスタから一つの記憶クラスタを選択する動作と、複数の記憶クラスタから選択された一つの記憶クラスタを特定する情報を記憶することなく、複数の記憶クラスタから選択された一つの記憶クラスタ上にデータのユニットを記憶する動作と、を備える方法。さらなる例示の実施形態は、コンピュータ・システム上で実行された場合に上述の方法を実行する命令がエンコードされた少なくとも一つのコンピュータ可読媒体を対象としている。

【０００９】

別の例示の実施形態は、データを記憶するための装置を対象としており、その装置は、記憶環境における複数の記憶クラスタのうちの少なくとも一つの記憶クラスタ上にデータのユニットを記憶するために、ホスト・コンピュータ上で実行されるアプリケーション・プログラムからの要求を受信する入力と、その入力に接続され、その入力からの要求を受信し、データのユニットを記憶するために、複数の記憶クラスタから一つの記憶クラスタを選択し、かつ複数の記憶クラスタから選択された一つの記憶クラスタを特定する情報を記憶することなく、複数の記憶クラスタから選択された一つの記憶クラスタ上にデータのユニットを記憶する少なくとも一つの制御装置と、を備える。

【発明を実施するための最良の形態】

【００１０】

本明細書に記載の本発明の態様の用途は、相互接続された複数の独立なノードを含む分散型記憶環境における内容アドレス記憶装置に関する。そのような環境は、表１に示す同一出願人による米国特許出願に記載されている。その出願のそれぞれは、完全に本明細書に参照として組み入れられる。本明細書において、表１に列挙した出願を総称して、関連出願と呼ぶ。

【００１１】

【表１】

| 名称 | 出願番号 | 出願日 |
|---|---|---|
| 内容アドレス情報、カプセル化、表現、及び転送(Content Addressable Information, Encapsulation, Representation, And Transfer) | 09/236,366 | 1999/1/21 |
| ネットワーク上の内容アドレスデータへのアクセス(Access To Content Addressable Data Over A Network) | 09/235,146 | 1999/1/21 |
| 確実な記憶転送及び内容アドレス情報の検索のためのシステム及び方法(System And Method For Secure Storage Transfer And Retrieval Of Content Addressable Information) | 09/391,360 | 1999/9/7 |
| 記憶システムにおけるデータ保有のための方法及び装置(Method And Apparatus For Data Retention In A Storage System) | 10/731,790 | 2003/12/9 |
| データ記憶システムにおける内容へのアクセスを円滑にするための方法及び装置(Method And Apparatus For Facilitating Access To Content In A Data Storage System) | 10/731,613 | 2003/12/9 |
| データ記憶システムにおける記憶場所インデックスをキャッシュに格納するための方法及び装置(Method And Apparatus For Caching A Location Index In A Data Storage System) | 10/731,796 | 2003/12/9 |
| データ記憶システムにおける物理的な記憶場所の選択を円滑にするための内容アドレスの解析のための方法及び装置(Method And Apparatus For Parsing A Content Address To Facilitate Selection Of A Physical Storage Location In A Data Storage System) | 10/731,603 | 2003/12/9 |
| 最も近い時間に記憶システムに書き込まれたデータのユニットを表示するための内容アドレスを生成するための方法及び装置(Method And Apparatus For Generating A Content Address To Indicate Data Units Written To A Storage System Proximate In Time) | 10/731,845 | 2003/12/9 |

| | | |
|---|---|---|
| 記憶システムにおけるデータの保有期間を変更するための方法及び装置(Method And Apparatus For Modifying A Retention Period For Data In A Storage System) | 代理人整理番号: E0295.70195US00 出願番号未通知 | 2004/1/21 |
| 記憶システムにおけるデータの保有期間を延長するための方法及び装置(Method And Apparatus For Extending A Retention Period For Data In A Storage System) | 代理人整理番号: E0295.70200US00 出願番号未通知 | 2004/1/21 |
| 記憶システムにおけるデータの保有期間を間接的に特定するための方法及び装置(Method And Apparatus For Indirectly Identifying A Retention Period For Data In A Storage System) | 代理人整理番号: E0295.70201US00 出願番号未通知 | 2004/1/21 |
| 記憶システムにおけるデータの保有期間を間接的に特定するための方法及び装置(Method And Apparatus For Indirectly Identifying A Retention Period For Data In A Storage System) | 代理人整理番号: E0295.70202US00 出願番号未通知 | 2004/1/21 |

分散型記憶環境１００の一例を図１に示す。分散型記憶環境１００は、複数のアクセス・ノード（ａｃｃｅｓｓ　ｎｏｄｅ：ＡＣ）１０１ａ～１０１ｃ、及び複数の記憶ノード（ｓｔｏｒａｇｅ　ｎｏｄｅ：ＳＮ）１０３ａ～１０３ｅを含む。アクセス・ノード１０１は、ホスト・コンピュータ１０５からのアクセス要求を受信して、応答し得る。記憶ノード１０３は、ホスト・コンピュータ１０５によって記憶環境１００に送信されたデータを記憶し得る。アクセス・ノード１０１及び記憶ノード１０３は、ネットワーク（図示略）によって接続され、そのネットワーク上で通信することによって、各ノードはネットワーク上における自身の存在を他のノードに知らしめ得る。このように、それらのノードは、ホスト・コンピュータ１０５に対して、協働して、アクセス要求を処理し、データを記憶し得る。

【００１２】

各ノードは、処理リソース（例えば、プロセッサ及びメモリ）、並びに記憶デバイスを含み得る。それらのノードは、互いに通信することにより、データを記憶し、アクセス要求に応答し、他の環境機能を実行する。記憶環境のユーザ（例えば、ホスト・コンピュータ１０５又はその上で実行されるアプリケーション・プログラム）にとって、その記憶環境は単一の実体のように見なされ得る。即ち、ユーザは、その記憶環境が複数の分離したノードを含むことを、或いは、データの特定のユニットがどの記憶ノードに記憶されているのか、或いはミラー化されているのかを、認識する必要がない。

【００１３】

記憶環境１００の記憶容量を増大させるために、より多くの記憶ノードが追加され、そのネットワークに接続され得る。これらの追加の記憶ノードはネットワーク上における自身の存在を知らしめ得るため、アクセス・ノード１０１は、データを記憶するのに、その追加の記憶装置を利用することが可能になる。アクセス・ノードの個数を増大させることなく、記憶ネットワークにより多くの記憶ノードを追加すると、アクセス・ノードは、その記憶環境における隘路として作用し、性能の低下が起こり得る。このように、記憶ノードの個数を増大させた場合は、アクセス・ノードの個数も増大させることが望ましい。

【００１４】

記憶環境１００は、上記の関連出願に記載の複数の機能、例えば、ホスト１０５からの

書き込み要求に応答してデータを記憶ノード１０３のうちのどれに記憶するのかを判断すること、ホスト１０５からの読み出し要求に応答してデータが記憶ノード１０３のうちのどれに記憶されているのかを判断すること、記憶環境から消去しても構わない不要データの収集を実行すること、データが記憶環境から消去されるべきではない時間間隔を特定する保持期間を取り決めること、データをミラー化すること（即ち、一つ以上のミラー・コピーを記憶環境の異なる記憶ノード上で生成すること）、一つ以上のノードの障害を補償するために自己回復すること、及び他の機能を実行し得る。そのような機能は、記憶ノード、アクセス・ノード、又は両方のノードによって実行され得るものであり、そのような機能を実行することは、ノード間のネットワーク・トラフィックの原因となり得る。

【００１５】

例えば、自己回復の機能を実行するために、他のノードは、いつノードに障害が起きたのかを検出し得る。それに応答して、環境１００は、障害を起こしたノード上に記憶されたデータに対するアクセス要求を、そのデータのミラー・コピーを記憶する他のノードに再び方向付けし、継続する障害を容認するための別のミラーを構築し得る。ノードは、障害を起こさずに動作していることを表示する正常メッセージ（ｋｅｅｐ－ａｌｉｖｅ ｍｅｓｓａｇｅ）をネットワーク上に送信し得る。そのノードからの正常メッセージを他のノードが受信していない場合は、他のノードは、そのノードは障害を起こしていると判断し得る。記憶環境により多くのノードを追加することは、より多くの正常メッセージがネットワーク上で送信される原因となり、その結果、より多くのネットワーク・トラフィックがもたらされる。

【００１６】

別の例として、記憶環境１００は、データの確認を円滑にするために、関連出願に記載のｂｌｏｂ記憶場所インデックス（ｂｌｏｂ ｌｏｃａｔｉｏｎ ｉｎｄｅｘ：ＢＬＩ）のようなインデックスを保持し得る。ＢＬＩにより、データのユニットがどの記憶ノード上に記憶されているのかが特定され得る。ネットワークにおける各アクセス・ノード又は記憶ノードは、一部のＢＬＩの管理に関与し得る。ＢＬＩはアクセス・ノード及び／又は記憶ノードの全体にわたって分散し得るため、データのユニットが記憶環境に書き込まれた、或いは、記憶環境から削除された時に、ＢＬＩを保持し、かつ更新することは、ノード間におけるネットワーク・トラフィックの発生の原因となる。より多くのノードを追加することによって、ＢＬＩを管理する負担はより多数のノード間で共有され得るため、より多量のネットワーク・トラフィックの発生が引き起こされる。

【００１７】

関連出願に記載の一部を含む他の機能、例えば、不要データの収集を実行すること、（例えば、全てのノードに対する送信メッセージを介して）記憶環境上で内容を確認すること、（ノードが記憶環境に追加された時か、或いは、記憶環境から削除された時に）ＢＬＩを再分類することは、ノードが記憶環境に追加された際、より多量のネットワーク・トラフィックを引き起こし得る。そのような増大するネットワーク・トラフィックにより、記憶環境の性能は低下し得る。

【００１８】

上述したように、記憶環境の記憶容量を増大させるために、記憶ノード１０３をその記憶環境に追加し得る。加えて、さらなる記憶ノードを追加することによって引き起こされる性能の低下を防止するために、アクセス・ノード１０１が追加され得る。しかしながら、一旦、記憶環境におけるノードの個数が特定の数に到達すると、アクセス・ノード及び記憶ノードの追加はネットワーク・トラフィックの増大の原因となるため、さらなるノードを追加することによって被る性能の向上のうちの少なくとも一部は、記憶環境におけるノードによって発生するネットワーク・トラフィックの増大、並びに、記憶環境を支援する基幹設備の（上述したような）機能を実行する際に使用される処理リソースの量の増大によって相殺されることを、出願人は認識した。このように、記憶容量及び／又は性能を向上させるために記憶環境にさらなるノードを追加した場合に、記憶環境の全体の性能は期待した程度には向上しないかもしれない、或いは、場合によっては、低下をすることす

らあるかもしれない。この性能の影響は、以下において、性能低下と参照され、この用語は、本明細書において、実際の性能の低下のみならず、さらなる記憶リソースの追加によって達成されると期待した性能の向上に比べて低い性能の向上をも参照するものとして使用される。

【００１９】

本発明の一つの実施形態は、一体記憶を提供するように協働するが、独立している記憶クラスタをさらに提供することを対象としており、一つのクラスタにさらなる記憶ノードを追加したとしても、他のクラスタに対しては上述の性能低下の問題は発生しない。本明細書において使用されるように、記憶クラスタという用語は、相互接続された一つ以上のノードのグループであって、それらのノードを論理的に接続する少なくとも一つのソフトウェア・ユーティリティを共有するグループを参照する。例えば、記憶クラスタのノードは、自己回復のソフトウェア・ユーティリティを共有し得るものであり、クラスタにおけるノードは、そのクラスタにおける他のノードにより送信される正常メッセージは監視するが、そのクラスタ外部のノードによるものは監視しない。そのノードはさらに、共通のＢＬＩを共有し、一つの記憶クラスタのＢＬＩは、他の任意の記憶クラスタのＢＬＩとは独立であり得る。記憶クラスタのノードはさらに、或いは、代わりに、例えば、不要データ収集ユーティリティ及び／又はクラスタ内で記憶されたデータがどこにミラー化されているのかを追跡するデータ・ミラー化ユーティリティのような他のユーティリティを共有し得る。一つの実施形態において、クラスタにおける各ノードは、クラスタにおける全ての他のノードのアドレス（例えば、ＩＰアドレス）を認識しているが、本発明はこの点に関して限定されている訳ではない。

【００２０】

記憶クラスタにおけるノードは、そのクラスタ外部のノード（例えば、別の記憶クラスタにおけるノード）とはソフトウェア・ユーティリティを共有しないため、一つの記憶クラスタ内部における内部ネットワーク・トラフィック及びノードの処理リソースの使用は、他の任意の記憶クラスタの性能には影響を与えない。結果として、上述の性能低下の問題に遭遇することなく、記憶容量を増大させるために、記憶環境にさらなる記憶クラスタを追加し得る。

【００２１】

（異なるクラスタのノード間では共有されないが）一つのクラスタにおけるノード間では共有され得るソフトウェア・ユーティリティの型についての説明は、単に例示の目的で提供されたものであること、かつ、本明細書記載の本発明の態様が、記憶クラスタにおけるノードが上述の特定の型のソフトウェア・ユーティリティ又は他の任意の特定の型のソフトウェア・ユーティリティを共有する記憶環境においての利用に限定されないことは、当然のことであろう。同様に、関連出願に記載されたような特性を有する記憶環境の例は、本明細書において、例示の目的のために記載されたものであり、本明細書に記載の本発明の態様は、この点に関しても限定されないこと、かつ無数の他の構成を有する記憶環境においても利用可能であることは、当然のことであろう。例えば、ノードを参照するところで、任意のデバイス又は一つの構成要素として共に機能するデバイスの集合体を参照することが可能であるために、一つ以上の相互接続されたノードを備える記憶クラスタに対する参照は、関連出願に絡めて記載したようなアクセス・ノード及び記憶ノードの構成に限定されない。

【００２２】

本発明の一つの態様において、複数の記憶クラスタは、記憶環境のユーザ（例えば、ホスト・コンピュータ又はその上で実行されるアプリケーション・プログラム）にとって透過的であるように管理され、従って、ユーザは、特定のデータのユニットをアクセスするのに、どの記憶クラスタ上にそのデータのユニットが記憶されているのかを認識する必要がない。このように、記憶環境にデータを記憶し、かつ記憶環境からデータを取得するアプリケーション・プログラムは、記憶環境に書き込まれたデータがどのクラスタ上に記憶されているのかを認識することなく、複数の記憶クラスタを単一の記憶環境として取り扱

い得る。それゆえ、本発明の一つの実施形態において、（例えば、ホスト上の）アプリケーション・プログラムが記憶環境に対しデータのユニットの書き込み要求を送信した場合における、どの記憶クラスタ上にデータのユニットが書き込まれるべきかの判断、及び、そのアプリケーションが読み出し要求を送信した場合における、どの記憶クラスタ上にそのデータが記憶されているのかの判断、いずれの判断もアプリケーション・プログラムにとって透過的であるように、行われる。

【００２３】

本発明は、どの記憶クラスタ上にデータのユニットを記憶するのかを判断し、かつ以前に書き込まれたデータのユニットを適切なクラスタから取得するコンピュータ・システムの態様を実施することについて、任意の特定の実施技術に限定される訳ではない。一つの実施形態においては、これらの機能は、ホスト・コンピュータ上のアプリケーション・プログラミング・インターフェイス（ａｐｐｌｉｃａｔｉｏｎ ｐｒｏｇｒａｍｍｉｎｇ ｉｎｔｅｒｆａｃｅ：ＡＰＩ）によって実行され得る。この実施形態は、図２に関連して以下に説明される。

【００２４】

図２は、二つの記憶クラスタ２０７ａ，２０７ｂを含む記憶環境２０６を通信するホスト・コンピュータ２０１を示す。ホスト・コンピュータ２０１は、データのユニットの読み出し、書き込みを行うアプリケーション・プログラム２０３、及び記憶環境２０６との接続を仲介するのに適合したアプリケーション・プログラミング・インターフェイス（ＡＰＩ）２０５を実行する。ホスト２０１としては、例えば、リソース（例えば、データ記憶、電子メール、及び他のサービス）を提供するサーバ、クライアント（例えば、ユーザによって操作される汎用コンピュータ環境）、ネットワーク部品（例えば、スイッチ）、又は任意の他の型のコンピュータがあり得る。ホスト２０１は、通信リンク２１３ａ，２１３ｂによって、記憶環境２０６の記憶クラスタ２０７ａ，２０７ｂに接続され、その接続によって、ホスト２０１は、データ記憶及び取得のために記憶環境２０６を利用することが可能となる。リンク２１３ａ，２１３ｂとしては、任意の型のネットワーク又は直接接続を含む任意の型の通信リンクがあり得る。

【００２５】

アプリケーション・プログラム２０３としては、ホスト２０１のユーザ又は管理者によって使用され、記憶環境２０６にデータを記憶し、かつ／若しくは、記憶環境２０６からデータを取得する任意のコンピュータ・アプリケーション・プログラムがあり得る。アプリケーション・プログラム２０３は、ＡＰＩ２０５に接続され、その接続によって、アプリケーション・プログラム２０３は、ＡＰＩ２０５を利用して、複数の記憶クラスタ２０７と通信し得る。一つの実施形態において、アプリケーション・プログラム２０３は、記憶環境２０６が複数の記憶クラスタを含んでいることすら認識する必要は無く、複数のクラスタ２０７間のデータ記憶の管理は、アプリケーション・プログラム２０３にとって透過的であるように実行され得る。

【００２６】

ＡＰＩ２０５は、任意の適切な方法により具体化され得る。例えば、ＡＰＩ２０５は、アプリケーション・プログラム２０３のプログラマーに対して提供されるコンピュータ・オブジェクト・コードを含み得る。そのコンピュータ・オブジェクト・コードは、記憶環境と通信するのに利用され得るルーチンを含む。ＡＰＩ２０５に接続された時、アプリケーション・プログラムは、これらのルーチンを呼び出して、記憶環境２０６と通信し得る。ＡＰＩ２０５は、任意の個数のアプリケーションに接続され得るものであり、アプリケーション・プログラム２０３のプログラマーを記憶環境２０６が通信する際に利用する内部プロトコルから遮断する。

【００２７】

アプリケーション・プログラム２０３がＡＰＩ２０５の一つ以上のルーチンを呼び出して、記憶環境２０６にデータを書き込む場合、ＡＰＩ２０５は、そのデータを記憶する記憶クラスタを選択し、そのデータを含む書き込み要求２０９を適切な記憶クラスタに対し

て送信し得る。その記憶クラスタの選択は、任意の適切な技術を用いることにより、実行され得る。例えば、ＡＰＩ２０５は、ラウンド・ロビン方式を利用してもよく、その方式によって、記憶環境におけるクラスタは、各書き込み要求に対して順次選択される。図２の例において、ＡＰＩ２０５は、書き込み要求を処理するために、記憶クラスタ２０７ａと記憶クラスタ２０７ｂとの間を交互に選択することが可能である。或いは、ＡＰＩ２０５は、例えば、各記憶クラスタ上の利用可能な記憶容量、各記憶クラスタ上の負荷、記憶されるべきデータのサイズ若しくは型、又は他の任意の適切な基準に基づいて、データを記憶する記憶クラスタを選択し得る。記憶クラスタ上の負荷は、例えば、記憶クラスタによってどの程度の個数のオブジェクト（例えば、ｂｌｏｂ又はＣＤＦ）が処理されているのか、記憶クラスタのＣＰＵがどの程度使用されているのか、記憶クラスタのディスクがどの程度使用されているのか、かつ／若しくは、他の任意の適切な要素に基づいて、判断され得る。

【００２８】

アプリケーション・プログラム２０３がＡＰＩ２０５の一つ以上のルーチンを呼び出して、以前に記憶されたデータのユニットを記憶環境から読み出す場合、ＡＰＩ２０５は、どの記憶クラスタ上に要求されたデータのユニットが記憶されているかを判断し得る。これは、任意の適切な方法によって実行することが可能である。例えば、ＡＰＩ２０５は、各データのユニットがどのクラスタ上に記憶されているのかを特定する情報を記憶し得る。しかしながら、別の実施形態においては、そのような情報は、何ら記憶されておらず、ＡＰＩ２０５は、読み出し要求に応答して、そのデータが何処に記憶されているのかを判断する。

【００２９】

一つの実施形態において、ＡＰＩ２０５は、記憶環境における各記憶クラスタに対して、要求するデータのユニットを特定し、かつそのクラスタにその特定されたデータのユニットが記憶されているのか否かを判断するように要求する命令を送信する。その命令としては、読み出し命令、又は、データのユニットを読み出そうとするのではなく、どのクラスタ上にそれが記憶されているのかの判断のみを行う命令（本明細書において、「判断命令（ｅｘｉｓｔｓ　ｃｏｍｍａｎｄ）」と参照される）が可能である。

【００３０】

その命令は、記憶クラスタに対して、順次、或いは、平行して送信され得るものであり、この点に関して本発明は限定されない。例えば、その命令を順次送信する場合は、ＡＰＩ２０５は、最初に、判断命令２１１を記憶クラスタ２０７ａに送信し得る。クラスタ２０７ａがそこにデータのユニットが記憶されていると応答してきた場合は、次いで、ＡＰＩ２０５は、記憶クラスタ２０７ａに対して、そのデータのユニットの読み出し要求を送信し、他のクラスタに対して判断命令をさらに送信する必要はない。クラスタ２０７ａがそこにはデータのユニットが記憶されていないと応答してきた場合は、次いで、ＡＰＩ２０５は、別の判断命令２１５をクラスタ２０７ｂに送信し得る。クラスタ２０７ｂがそこに特定されたデータのユニットが記憶されていると応答してきた場合は、次いで、ＡＰＩ２０５は、クラスタ２０７Ｂに対して、そのデータのユニットの読み出し要求を送信し得る。ＡＰＩ２０５が判断命令を平行して送信する場合は、判断命令２１１及び判断命令２１３は、それぞれクラスタ２０７ａ及びクラスタ２０７ｂに対して、ほぼ同時に送信され得る。記憶クラスタのうちの一方がそこに特定されたデータのユニットが記憶されているとＡＰＩ２０５に対して応答してきた時は、次いで、ＡＰＩ２０５は、そのクラスタに対して、そのデータのユニットに対する読み出し要求を送信し得る。上述したように、判断命令及びそれに続く読み出し要求を利用する代わりに、ＡＰＩ２０５は、記憶クラスタに対して、順次、或いは、平行して、読み出し要求のみを送信しても（即ち、判断命令を送信しなくても）構わない。

【００３１】

判断命令又は読み出し要求を平行して送信することの利点は、データのユニットの検索が各クラスタに対して同時に実行されるため、データのユニットの確認までに費やされる

時間が低減されることである。しかしながら、これらの同時検索は、実際には多数の記憶クラスタ上でそのような検索は不成功に終わるにも拘わらず、各クラスタ上の処理リソースを浪費させる（従って、性能に影響する）。逆に、判断命令又は読み出し命令を順次送信する場合は、データのユニットの確認までに費やされる時間は増大し得る。これは、以前の記憶クラスタ上の検索が完了するまでは、判断命令は記憶クラスタに送信されないからであり、しかしながら、一部のクラスタについては、それらのクラスタに対し判断命令又は読み出し要求が送信される以前に、データのユニットの検索を実行する必要がなくなるために、記憶環境の性能が極めて影響を受けるということはないであろう。各クラスタ内での検索は、任意の適切な方法によって実行可能であって、この点に関して本発明は限定されない。検索技術の例は、上記の関連出願に記載されている。

【００３２】

データのユニットの検索の際に記憶クラスタが被る全体の処理負荷を低減するために、ＡＰＩ２０５は検索方法を階層的に構成し得る。この場合、クラスタは相対的に表面的な（計算が集中することのない）検索を最初に実行し、必要な場合（例えば、全てのクラスタにわたる表面的な検索が不成功に終わった場合）にのみ、より完全な検索を実行する。

【００３３】

例えば、関連出願に記載されているように、各クラスタは、そのクラスタ上に記憶されたデータのユニットの一部を特定するＢＬＩを含み得る。相対的に表面的な検索では、最初に、クラスタにおけるＢＬＩの検索のみが実行され得るものであり、その検索は相対的に迅速かつ比較的に低廉なものである。環境における全てのクラスタに対する（順次、或いは、平行）ＢＬＩ検索が不成功に終わった場合、より完全な検索が実行され得る。ＢＬＩ検索は、単に、相対的に表面的な検索を実行するための技術の一例に過ぎず、階層的検索方法を利用する本発明の態様として他の技術も可能であり、この点に関して限定されないことは、当然のことであろう。さらに、本明細書に記載の本発明の態様は、関連出願に示す構成を有する記憶環境における利用に限定されることも、ＢＬＩを利用する記憶クラスタに限定されることもないことは、当然のことであろう。

【００３４】

図２の例において、記憶環境２０６は二つの記憶クラスタを有するものとして示されている。これは、一例として与えたものに過ぎず、記憶環境２０６は、三つ以上の記憶クラスタを含み得るものであり、本発明がこの点に関して限定されないことは、当然のことであろう。同様に、図２では、記憶環境２０６に対しデータを送信し、かつ記憶環境２０６からデータを受信するホスト・コンピュータ２０１は唯一つ示されている。しかしながら、記憶環境２０６は、より多くのホスト・コンピュータで動作し得るものであり、本発明がこの点に関して限定されないことは当然のことであろう。

【００３５】

上述の環境において、データを記憶する記憶クラスタを選択すること、かつ／若しくは、以前に書き込まれたデータのユニットを記憶するクラスタはどれであるのかを判断することに関する本発明の態様は、ＡＰＩにより実行される。しかしながら、この説明は、単に例示の目的のために与えられたものであり、本発明がこの点に関して限定されないことは当然のことであろう。上述したように、本発明の一つの例示の実施形態の利点は、複数のクラスタ間にわたる記憶環境の利用は、ホスト上で実行されるアプリケーション・プログラムにとって透過的であり、そのため、そのアプリケーション・プログラムは、複数のクラスタの存在を認識する必要もなければ、複数のクラスタを管理することに関連するオーバーヘッドに関心を払う必要もない。しかしながら、本発明の他の態様では、この点に関して限定されず、代替案として、複数のクラスタの管理を、アプリケーション・プログラムにとって透過的ではないように実行することが可能である。加えて、複数のクラスタの管理がアプリケーション・プログラムにとって透過的であろうとなかろうと、そのような管理は、上述のようにＡＰＩにおいて実行される必要はなく、代替案として、コンピュータ・システムの他の記憶場所において実行されても構わない。例えば、本発明の一つの実施形態において、複数のクラスタの管理は、クラスタ自体の内部で実行可能である。こ

の実施形態において、ＡＰＩは単一のクラスタのみを認識し得るが、複数のクラスタは、複数のクラスタ間でのデータの分布を管理するために、それらの間を通信することが可能である。或いは、ＡＰＩは、ＡＰＩが任意のアクセス・ノードに対して、そのアクセス・ノードがどのクラスタの一部を形成しているのかに拘らず、任意のアクセス要求を送信し得るように、全てのクラスタの全てのアクセス・ノードを認識していても構わないが、これらのノードが二つ以上のクラスタ間に分割されていることは、場合によっては認識しない。本発明の別の実施形態において、複数のクラスタの管理は、任意のクラスタからも、ホストからも分離したアプライアンスにおいて実行可能である。そのアプライアンスは、多数の形態のうちの任意の形態を取ることが可能であり、例えば、ホスト・コンピュータと記憶クラスタとを相互接続するための、ネットワークの一部を形成するネットワーク部品（例えば、ルータ）があり得る。

【００３６】

図３は、（書き込み要求の場合に）どの記憶クラスタ上にデータのユニットを記憶するのかの判断、及び、（読み出し要求の場合に）どの記憶クラスタ上に以前に書き込まれたデータのユニットが記憶されているのかの判断が、アプライアンス３０３によって成される実施形態の一つの例示の表現を提供する。アプライアンス３０３は、ホスト３０１から記憶環境３０５に対するアクセス要求を受信する。アプライアンス３０３は、次いで、そのアクセス要求を適切な記憶クラスタに送信し得る。アプライアンス３０３としては、例えば、ホスト３０１及び記憶環境３０５に（例えば、ネットワークを介して）接続され、かつ、ホスト３０１上で実行されるアプリケーションが、複数の記憶クラスタを単一の記憶環境とみなして、それらの記憶クラスタにアクセスすることを可能にする機能を実行するコンピュータがあり得る。或いは、アプライアンス３０３は、記憶環境３０５の内部に含まれ得る。

【００３７】

アプライアンス３０３は、どの記憶クラスタ上にデータのユニットを記憶するのかを判断するのに、かつ、どの記憶クラスタ上に以前に書き込まれたデータのユニットが記憶されているのかを判断するのに、任意の適切な技術を利用し得るものであり、本発明はこの点に関して限定されない。

【００３８】

上記説明から当然のように、図３の実施形態の場合、ホスト・コンピュータ３０１上で実行されるアプリケーション・プログラム又はＡＰＩは、記憶環境３０５が複数の記憶クラスタを含んでいることを認識する必要すらない。さらに、そのアプリケーション・プログラム又はＡＰＩは、データのユニットが何処に記憶されるのかを認識することなく記憶環境３０５にその特定のデータのユニットを書き込むこと、或いは、データのユニットが何処に記憶されていたのかを認識することなくその記憶環境３０５からその特定のデータのユニットを取得することを実行し得る。そのホスト・コンピュータは、単にアクセス要求をアプライアンス３０３に対し送信し得るのみであって、次いで、アプライアンス３０３が何処にデータのユニットを書き込むのか、或いは、何処に以前に書き込まれたデータのユニットが記憶されているのかを判断する。

【００３９】

図３の例において、二つの記憶クラスタを有する記憶環境３０５が示されている。これは単に一例として与えられているに過ぎず、記憶環境３０５は三つ以上の記憶クラスタを含み得るものであり、本発明がこの点に関して限定されないことは、当然のことであろう。同様に、図３では、記憶環境３０５に対しデータを送信し、かつ記憶環境からデータを受信するホスト・コンピュータ３０１が唯一つ示されているのみである。しかしながら、記憶環境３０５はより多くのホスト・コンピュータで動作し得るものであり、本発明がこの点に関して限定されないことは、当然のことであろう。さらに、図３では、ホスト３０１と記憶環境３０５との間で動作するアプライアンス３０３が唯一つ示されているが、複数のアプライアンスも利用可能である。このように、記憶環境に至る全てのアクセス要求が単一のアプライアンスにより処理されることによる隘路は回避され得る。追加として、

図２の例では、ホスト・コンピュータと記憶環境との間の通信にＡＰＩが利用されているが、本発明の一部の実施形態では、ＡＰＩを利用する必要はなく、アプリケーション・プログラムは記憶環境と直接に通信し得る。

【００４０】

図４は、（書き込み要求の場合に）どの記憶クラスタ上にデータのユニットを記憶するのかの判断、及び、（読み出し要求又は消去要求の場合に）どの記憶クラスタ上に以前に書き込まれたデータのユニットが記憶されているのかの判断が、記憶クラスタ自体によって成され得る実施形態の一つの例示の表現を提供する。図４の例では、ホスト・コンピュータ４０１は、二つの記憶クラスタ４０５ａ，４０５ｂを含む記憶環境４０３と通信する。図１に関連して上述した場合と全く同様に、記憶クラスタ４０５ａは複数のアクセス・ノード４０７ａ～４０７ｃ及び複数の記憶ノード４１１ａ～４１１ｄを含み、記憶クラスタ４０５ｂは複数のアクセス・ノード４０９ａ～４０９ｃ及び複数の記憶ノード４１３ａ～４１３ｃを含む。本発明のこの実施形態において、少なくとも一つの記憶クラスタは、記憶環境における他の記憶クラスタの存在を認識するが、各クラスタは独立なクラスタである（例えば、各クラスタは独立なソフトウェア・ユーティリティを有している）。図４において示すように、クラスタ４０５ａ，４０５ｂの間には通信経路４２１が存在し、それらを通信可能にする。この通信経路は任意の適切な方法により具体化することが可能であり、本発明はこの点に関して限定されない。

【００４１】

アクセス・ノード４０７又はアクセス・ノード４０９がデータのユニットを取得するための要求を受信した時、そのアクセス・ノードは、その要求されたデータのユニットがそのクラスタに記憶されているか否かを判断し得る。記憶されていない場合は、そのアクセス・ノードは、上述した場合と全く同様に、順次、或いは、平行して、記憶環境における他の記憶クラスタに判断命令及び／又は読み出し命令を送信し得る。

【００４２】

ホスト・コンピュータ４０１上で実行されるアプリケーション・プログラムは、記憶クラスタ４０５ａ，４０５ｂのアクセス・ノード４０７，４０９に対して、（例えば、ＡＰＩを介して）アクセス要求を送信し得る。一つの実施形態において、アプリケーション・プログラム及び／又はＡＰＩは、アクセス・ノード４０７，４０９が異なる記憶クラスタに属していることを認識していない。アプリケーション・プログラム及び／又はＡＰＩの観点からは、アクセス・ノード４０７，４０９は、どの記憶クラスタが要求されたデータのユニットを記憶しているかに拘らず、アクセス要求を受信し得る。実際に、アプリケーション・プログラム及び／又はＡＰＩは記憶環境が複数の記憶クラスタを含んでいることすら認識する必要はなく、それゆえ、本発明の一つの実施形態において、データが記憶環境上の何処に記憶されているのかは、アプリケーション・プログラム及び／又はＡＰＩにとって透過的である。

【００４３】

アクセス・ノード４０７又はアクセス・ノード４０９が、ホスト４０１からデータのユニットを書き込むための要求を受信した時、そのアクセス・ノードは、そのデータのユニットをそのクラスタ上に記憶するのか、或いは、別のクラスタに対し、そのデータのユニットを記憶させるために送信するのかの判断をし得る。そのような判断は任意の適切な要素に基づいてされ得るものであり、その例は上述した（例えば、ラウンド・ロビン方式、記憶されるべきデータのサイズ又は型、各記憶クラスタ上で利用可能な容量、各記憶クラスタ上での現在の処理負荷）。

【００４４】

或いは、アプリケーション・プログラム及び／又はＡＰＩは、記憶環境における任意のアクセス・ノードに対して要求を送信するように許可されているのではなく、一つの記憶クラスタ（例えば、クラスタ４０５ａ）のアクセス・ノードに対してのみアクセス要求を送信するように許可されていても構わない。この記憶クラスタのアクセス・ノードは、次いで、それらが属する記憶クラスタ、及び／又は、記憶環境におけるそれらが属さない他

の記憶クラスタに対するデータの書き込み、並びに、それらの記憶クラスタからのデータの取得に関与する。

【００４５】

一つの実施形態において、本発明の態様を、関連出願に記載の型の内容アドレス記憶装置（ｃｏｎｔｅｎｔ　ａｄｄｒｅｓｓａｂｌｅ　ｓｔｏｒａｇｅ：ＣＡＳ）の環境で利用することが可能である。内容アドレス記憶装置の環境において、データは、そのデータ自体の内容に基づいて生成された内容アドレスを用いることにより記憶される。内容アドレスは、記憶されるべきデータに対してハッシュ関数を適用することにより生成され得る。ハッシュ関数の出力は、ホストと記憶環境との間の通信の際にデータを参照するために利用される内容アドレスの少なくとも一部として使用され得る。内容アドレスは、記憶環境内部における一つ以上の物理的記憶場所に（例えば、記憶環境内部において）マップされ得る。

【００４６】

ＣＡＳの用途において定義されたアーキテクチャにおけるデータのユニットは、ｂｌｏｂと参照される。図５に示すように、ｂｌｏｂ５０３としては、例えば、記憶環境（例えば、記憶環境１０５）上のホストにより記憶されるべき二進数データのユニット、具体例としては、患者のＸ線データ、企業の財務記録、又は他の任意の型のデータがあり得る。ｂｌｏｂ５０３が内容記憶アドレス環境に記憶された時、ｂｌｏｂ５０３に対して一意的なアドレスが、その内容に基づいて生成される。

【００４７】

各ｂｌｏｂ５０３は、それに関連する少なくとも一つの内容記述ファイル（ｃｏｎｔｅｎｔ　ｄｅｓｃｒｉｐｔｏｒ　ｆｉｌｅ：ＣＤＦ）５０１を有する。ＣＤＦ５０１はメタデータ５０５及び複数の参照５０７ａ，５０７ｂ，・・・，５０７ｎを含み得る。ＣＤＦは一つ以上のｂｌｏｂ又は他のＣＤＦを参照し得る。このように、参照５０７としては、例えば、ＣＤＦ５０１によって参照されるｂｌｏｂ及び／又はＣＤＦに対する参照があり得る。メタデータ５０５は、例えば、ＣＤＦ５０１の生成データ（例えば、ＣＤＦ５０１は記憶環境上に記憶されていたというデータ）、ｂｌｏｂ５０３の内容の記述、及び／又は、他の情報を含み得る。

【００４８】

上述のように、記憶環境は、処分しても構わないｂｌｏｂ及びＣＤＦを消去するための不要データ収集ユーティリティを利用し得る。関連出願に記載のシステムの一つの実施形態において、処分されても構わないｂｌｏｂ又はＣＤＦとは、任意のＣＤＦによって参照されていないｂｌｏｂ又はＣＤＦのことである。

【００４９】

ｂｌｏｂ及びそれに対応するＣＤＦは、別々に記憶環境に書き込まれ得る。本発明の一つの実施形態によれば、ｂｌｏｂ及びそれを参照するＣＤＦは、異なる記憶クラスタ上に記憶され得る。ｂｌｏｂを消去することが、そのｂｌｏｂをポイントするＣＤＦが存在しないことに依存しており、かつＣＤＦの消去に応答して、そのＣＤＦに対するポインタを消去するようにそれにより以前に参照されていた任意のｂｌｏｂを更新する不要データ収集ユーティリティを利用するシステム上で、この実施形態が利用された場合、困難が起こり得る。例えば、ＣＤＦが消去され、それが参照する任意のｂｌｏｂが異なる記憶クラスタ上に記憶されている場合、不要データ収集ユーティリティは、最初に、そのＣＤＦが記憶されている記憶クラスタで、参照ｂｌｏｂを検索し得る。その記憶クラスタ上で、参照ｂｌｏｂが確認されなかった場合は、（例えば、判断命令及び／又は読み出し要求を介して）他の記憶クラスタ上でそのｂｌｏｂを確認し、消去されたＣＤＦに対する参照は削除されるべきであると、そのｂｌｏｂを記憶する記憶クラスタに指示し得る。このプロセスはクラスタ間の通信において著しいオーバーヘッドを招き得る。さらに、一部の実施形態においては、各記憶クラスタは、記憶環境に他の関連する記憶クラスタが存在することを認識すらしないことが望ましく、従って、ＣＤＦが消去された場合、クラスタは、参照ｂｌｏｂを求めて他のクラスタを検索しようともしないかもしれない。

【００５０】
本発明の一つの実施形態において、上述の困難は、ｂｌｏｂとそのｂｌｏｂを参照するＣＤＦを同一の記憶クラスタ上に記憶することによって、緩和される。これは、任意の適切な方法によって達成され得る。例えば、図２から図４を参照して上述した配置のアルゴリズムと同一のものを利用することが可能であるが、ｂｌｏｂ及びそれを参照するＣＤＦの記憶環境への書き込みは、ｂｌｏｂ及びそのＣＤＦが同一のクラスタに書き込まれるように、単一の書き込みトランザクションにおいて可能である。その後、そのｂｌｏｂを参照するＣＤＦが記憶環境にさらに書き込まれる場合は、その追加のＣＤＦを記憶するための記憶場所を判断する構成要素（例えば、ＡＰＩ、アプライアンス、又は記憶クラスタ）は、最初に、どの記憶クラスタにそのｂｌｏｂが記憶されているのかを判断し、次に、それと同一の記憶クラスタに追加のＣＤＦを記憶し得る。

【００５１】
本発明の一つの実施形態において、アプリケーション・プログラムは、ＣＤＦの内容アドレスのみを記憶することによって、以前に記憶されたデータの履歴を保持し、それによって参照されたｂｌｏｂの内容アドレスについては、その情報は対応するＣＤＦから取得することが可能であるため、追跡しない。このように、ｂｌｏｂにアクセスするためには、アプリケーション・プログラムは、そのｂｌｏｂを参照するＣＤＦの取得を要求すればよい。アプリケーション・プログラムは、その要求されたＣＤＦを入手し、そのＣＤＦにより参照されたｂｌｏｂの内容アドレスを判断し得る。次いで、アプリケーション・プログラムは、この内容アドレスを利用して、そのｂｌｏｂに対する読み出し要求を送信し得る。本発明の一つの実施形態において、そのｂｌｏｂ及びＣＤＦは同一の記憶クラスタ上に記憶されているため、そのＣＤＦに対して要求を送信した構成要素は、どの記憶クラスタにそのｂｌｏｂが記憶されているのかを判断する検索を実行することを必要とせずに、そのｂｌｏｂに対する読み出し要求を同一の記憶クラスタに送信することが可能である。

【００５２】
本発明の別の実施形態において、記憶環境上に記憶されたデータを確認するのに検索技術を利用するのではなく、固有記憶場所アドレス指定（ｉｎｔｒｉｎｓｉｃ　ｌｏｃａｔｉｏｎ　ａｄｄｒｅｓｓｉｎｇ）として知られる概念を利用し得る。固有記憶場所アドレス指定は、データのユニットを、その内容アドレスに基づいて特定の記憶クラスタ上に配置することに関係する。それゆえ、データのユニットが後に読み込まれる時に、そのデータのユニットがどの記憶クラスタ上に記憶されているのかを判断するための検索を実行する必要はない。代わって、データのユニットを記憶している記憶クラスタは、データのユニットの内容アドレスに基づいて再び判断され得る。

【００５３】
例えば、内容アドレスの選択されたビットが、データのユニットをどの記憶クラスタ上に記憶するのかを判断するのに利用され得る。各記憶クラスタは、記憶されるべきアドレスの特定の範囲により指定され得る。例えば、データのユニットの内容アドレスの頭文字により、そのデータのユニットがどの記憶クラスタ上に記憶されるべきかが判断され得る。図６の記憶環境は三つの記憶クラスタを含む。頭文字が「Ａ」から「Ｉ」の場合には、そのデータのユニットは記憶クラスタ６０１上に、頭文字「Ｊ」から「Ｒ」の場合には、記憶クラスタ６０３上に、頭文字が「Ｓ」から「Ｚ」の場合には、記憶クラスタ６０５上に記憶され得る。この例が、限定ではなく例示を目的としていることは当然のことであろう。例えば、記憶クラスタを判断するのに検査されるものが頭文字である必要はなく、その範囲は、近似的に等しいサイズである必要も、連続的である必要もないような異なるものであっても構わない。

【００５４】
上述したように、データのユニットの内容アドレスは、少なくとも部分的にはデータのユニットの内容に基づき、任意の適切な方法により生成され得るものであり、本発明はこの点に関して限定されない。例えば、データのユニットは、ＭＤ５のようなハッシュ・アルゴリズムを用いることにより、ハッシュされ、内容アドレスの一部を生成し得る。関連

出願に記載されているように、内容アドレスは、そのデータのユニットが最初に記憶環境に書き込まれた時を表示するタイムスタンプ、及び／又は、その内容アドレスが一意的であることを保証する、保証一意識別子（ｇｕａｒａｎｔｅｅｄ　ｕｎｉｑｕｅ　ｉｄｅｎｔｉｆｉｅｒ：ＧＵＩＤ）のような他の情報を含み得る。このように、図７に示すように、内容アドレス７０１は、ハッシュ値、タイムスタンプ、及びＧＵＩＤを含み得る。

【００５５】

一部の状況では、書き込まれるべきデータのユニットを記憶する一つの記憶クラスタ上の記憶場所が選択される以前に、そのデータのユニットは、そのハッシュ値を計算するには大き過ぎる場合がある。例えば、記憶環境は、データのユニットについてのハッシュを計算している間に、そのデータのユニット全体を記憶するのに十分な大きさのバッファを有してはいない場合がある。このように、本発明の一つの実施形態において、内容アドレスのハッシュ値が計算される以前に、一つの記憶クラスタにおける記憶場所を選択することを可能にする技術が利用される。

【００５６】

一部の状況では、データのユニットを記憶する記憶クラスタを選択するのに、ハッシュ値を利用することが不可能である場合があるため、適切な記憶クラスタを選択するのに内容アドレスの別の部分（例えば、タイムスタンプ又はＧＵＩＤ部分）を利用することが望ましい。関連出願に説明されているように、タイプスタンプの情報を利用することは、例えば、一つのクラスタのように、記憶システム内にほぼ同時に書き込まれたデータのユニットに対する参照の一時的な局所性を提供する際に、有益となり得る。しかしながら、全てのデータのユニットが同時に同一のクラスタに書き込まれた場合、クラスタが平行して書き込み要求を処理することが抑制されるため、性能にはマイナスの影響が与えられ望ましくない。それゆえ、図８に示すように、記憶クラスタを選択するのに利用される内容アドレスの部分（例えば、タイムスタンプ）が十分にランダムであることを保証するために、この内容アドレスの部分は、アルゴリズム（ＭＤ５又は他の任意の適切なアルゴリズム）を用いることによりハッシュされ（８０３）、記憶場所識別子８０５が生成され得る。記憶場所識別子８０５の部分８０７（又は全て）が、どの記憶クラスタをデータのユニットを記憶するのに使用するべきかの判断に利用され得る。このように、ハッシュ・アルゴリズムの擬似ランダム性を利用することにより、部分８０７は十分にランダムであり、同時に書き込まれたデータのユニットは記憶クラスタにわたって一様に分散し得る。

【００５７】

一つの実施形態において、データのユニットの取得を円滑にするための記憶場所識別子８０５の保存は行われない。寧ろ、データのユニットを読み出したい場合は、記憶場所識別子８０５を生成するために、内容アドレスの部分８０１は再びハッシュし、その記憶場所識別子の部分８０７を、データのユニットがどの記憶クラスタ上に記憶されているのかの判断に利用すればよい。

【００５８】

データのユニットの読み出し要求が送信された場合に、そのデータのユニットを確認するための検索を実行する必要性は、データのユニットを記憶する記憶クラスタを選択するための固有記憶場所アドレス指定技術を利用することによって、未然に排除されている、ということは、当然のことであろう。

【００５９】

ｂｌｏｂ、ＣＤＦ、及び上述の不要データ収集ユーティリティを利用する内容アドレス記憶装置のシステムと併せて、固有記憶場所アドレス指定技術が利用された場合、ｂｌｏｂ及びそれに対応するＣＤＦは異なる記憶クラスタ上に記憶され得るものであり、これにより、上述の困難が発生し得る。従って、本発明の一つの実施形態は、データのユニットを記憶する記憶クラスタを選択するために固有記憶場所を用いる技術であって、ＣＤＦ及びその参照ｂｌｏｂが同一の記憶クラスタ上に記憶されることを保証する技術と併せた固有記憶場所を用いる技術を対象としている。一つの実施形態において、ｂｌｏｂ及びそのＣＤＦが同一の記憶クラスタ上に記憶されることを保証する技術では、ＣＤＦ及びその参

照ｂｌｏｂの内容アドレスには共通の識別子が用いられ、この共通の識別子を用いることにより、ＣＤＦ及びその参照ｂｌｏｂをどの記憶クラスタ上に記憶するかの判断が行われる。

【００６０】

例えば、ＣＤＦ及びその参照ｂｌｏｂの内容アドレスには、同一のＧＵＩＤが備っていればよい。このＧＵＩＤの部分（又はＧＵＩＤのハッシュの部分）は、どの記憶クラスタ上にデータのユニットが記憶されているのかの判断に利用され、同一のＧＵＩＤを備えたＣＤＦ及びｂｌｏｂは同一の記憶クラスタにマップされる。例えば、図９は、二つのｂｌｏｂ９０３，９０５を参照するＣＤＦ９０１を示す。本発明の一つの実施形態によれば、これらの三つのデータのユニットは以下のように同一の記憶クラスタ上に記憶される。最初に、アプリケーション・プログラム（例えば、図２における２０３）は、ｂｌｏｂ９０３を記憶環境に送信する。記憶環境との接続を仲介するためのＡＰＩをホスト・コンピュータ２０１が有しているこれらの実施形態において、ＡＰＩはｂｌｏｂ９０３の内容アドレスの一部としてＧＵＩＤを生成し得る。記憶クラスタを選択するための設備（例えば、ＡＰＩ２０５、アプライアンス３０３、又はクラスタのうちの一つ）は、ｂｌｏｂ９０３を記憶する記憶クラスタを選択し得る。次に、アプリケーション・プログラムは、ｂｌｏｂ９０５をＡＰＩに送信する。ＡＰＩがまだＣＤＦを受信していないために、ＡＰＩは、ｂｌｏｂ９０３，９０５が同一のトランザクションの一部である（即ち、いずれも同一のＣＤＦにより参照される）ことを認識し、ｂｌｏｂ９０５の内容アドレスにおいて同一のＧＵＩＤを利用し得る。結果として、ｂｌｏｂ９０３，９０５は同一の記憶クラスタ上に記憶される。次に、アプリケーション・プログラムは、記憶環境上で記憶されるべきＣＤＦ９０１をＡＰＩに送信する。ＣＤＦを受信することにより、ＡＰＩは、そのＣＤＦがそのトランザクションにおける最後のデータのユニットであることを認識する。このように、ＡＰＩはＣＤＦ９０１の内容アドレスに同一のＧＵＩＤを割り当て、従って、ＣＤＦ９０１はｂｌｏｂ９０３，９０５と同一の記憶クラスタ上に記憶される。

【００６１】

一つの実施形態において、アプリケーションが、対応するＣＤＦを介さずして、ｂｌｏｂに直接にアクセスすることは不可能である。このように、ＣＤＦは最初に読み出されなければならない。この実施形態によれば、ＣＤＦの内容アドレスにはＧＵＩＤが含まれており、ＣＤＦが記憶されている記憶クラスタを確認するのに利用され、ＣＤＦの読み出しを円滑にするが、ｂｌｏｂの内容アドレスにもそのＧＵＩＤが含まれている必要はない。これは、ｂｌｏｂが記憶されている記憶クラスタを確認する役割を担う構成要素は、既に読み出したＣＤＦと同一の記憶クラスタ上にそのｂｌｏｂは記憶されていると認識するからである。

【００６２】

記憶環境に新たな記憶クラスタが追加される場合、どの記憶クラスタ上にデータのユニットが記憶されているのかを判断するための選択アルゴリズムは変更され、その結果として、全ての記憶クラスタにわたってデータのユニットが一様に分散されることが望ましい。例えば、記憶環境に三つの記憶クラスタが存在する場合、３を法とする（内容アドレスにおける）ＧＵＩＤ（ＧＵＩＤ ｍｏｄｕｌｏ ３）のような選択アルゴリズムが、どの記憶クラスタ上にその内容が記憶されているのかを判断するのに用いられ得る。四番目の記憶クラスタが追加された場合、そのアルゴリズムを、４を法とするＧＵＩＤ（ＧＵＩＤ ｍｏｄｕｌｏ ４）に変更することが望ましい。しかしながら、新たなクラスタの追加の後にデータのユニットが読み出された時に、その新たなアルゴリズムを内容アドレスに適用した場合、その新たなアルゴリズムでは、アルゴリズムの変更の以前に記憶環境上に記憶されたデータのユニットを適切に確認することは不可能であるかもしれない。

【００６３】

このように、本発明の一つの実施形態は、以前に記憶されたデータのユニットに対してどのようなアルゴリズムによりクラスタを選択していたかを示す情報を保持することを対象としている。一つの実施形態において、この情報は、特定の期間に使用されていた選択

アルゴリズムを特定する。データのユニットの内容アドレスはタイムススタンプを含み得るため、そのタイムスタンプの情報を、そのデータのユニットが記憶された時間において、どのようなアルゴリズムが実施されていたのかを判断する情報に対するインデックスとして利用することが可能である。このアルゴリズムは、どの記憶クラスタ上にデータのユニットが記憶されているのかを判断するための内容アドレスの適切な部分に適用可能である。

【００６４】

どのようなアルゴリズムが特定の期間に使用されていたのかを示す情報の一例を、図１０に示す。図１０において、ライン１００１は特定の選択アルゴリズムに対するタイム・フレームを示し、そのアルゴリズムが時間「－１」から時間「１０１９９２２０９３０２１」まで使用されていたことを示す。値「－１」は、記憶環境の存在の開始を示す。ライン１００３は、ライン１００１により示されたタイム・フレームの間にどのようなアルゴリズムが有効であったのかを示し、図１０の例では、ファーストＢｌｏｂネーミング（Ｆａｓｔ　Ｂｌｏｂ　Ｎａｍｉｎｇ）、即ち、ＦＢＮと呼ばれるアルゴリズムが使用されていたことを示す。ファーストＢｌｏｂネーミングは、データのユニットがどの記憶クラスタ上に記憶されるべきかの判断に、内容アドレスのハッシュ値及びタイムスタンプの法演算を利用するアルゴリズムである。「法（ｍｏｄ）」指示器は、ファーストＢｌｏｂネーミングのアルゴリズムにおいて利用される法の値を指示する。従って、例えば、ライン１００３においては、１を法とするハッシュ値及びタイムスタンプのアルゴリズムが使用されている。ｒｅｓｕｌｔの領域には、いずれかの記憶クラスタに対応するアルゴリズムのいずれかの結果が示されている。ライン１００３の例では、１を法とした場合であるので、唯一の可能な結果は０である。このように、結果が０の場合、データのユニットは記憶クラスタＣ１上に記憶される。

【００６５】

ライン１００５は、新たなアルゴリズムが期間「１０１９９２０９３０２１」と「１０１９９８３８３８」との間に有効であったことを示す。ライン１００７，１００９に示すように、この期間では、２を法とするファーストＢｌｏｂネーミングのアルゴリズムが使用されており、その法の操作の結果が０であった場合は、データのユニットはクラスタＣ１上に記憶され、法の操作の結果が１であった場合は、データのユニットはクラスタＣ２上に記憶される。

【００６６】

データのユニットの内容アドレスのタイムスタンプは、データのユニットを確認する際に、どのようなアルゴリズムを利用するのかの判断に利用され得る。例えば、データのユニットのタイムスタンプが１０１９９２０９５６６であった場合、そのようなタイプスタンプは、ライン１００５に示す期間内に収まる。それゆえ、そのデータのユニットが記憶されている記憶クラスタの判断には、ライン１００７，１００９によって特定されるアルゴリズムを利用すればよい。

【００６７】

ライン１０１１は、時間「１０１９９８３８３８」において、新たなアルゴリズムが有効になったことを示し、終了時間における「－１」は、そのアルゴリズムが現在でも有効であることを示す。ライン１０１３，１０１５，１０１７，１０１９，１０２１は、ファーストＢｌｏｂネーミングのアルゴリズムにおいて、法の値として５が使用され、かつ、三つのクラスタ（即ち、クラスタＣ１，Ｃ２，Ｃ３）が現在利用されていることを示す。ライン１０１３は、５を法とするハッシュ値及びタイムスタンプの結果が０の場合は、データのユニットはクラスタＣ１上に記憶されることを示す。結果が、１又は２の場合は、そのデータのユニットはクラスタＣ２上に記憶される。結果が３又は４である場合は、データのユニットはクラスタＣ３上に記憶される。結果として、ライン１０１３～１０２１は、一部の記憶クラスタが他の記憶クラスタに比べてより多くのデータのユニットを記憶するように、記憶環境における記憶クラスタの重み付けする方法の一例を提供している。このようなクラスタの重み付けは、例えば、一つの記憶クラスタが、記憶環境におけ

る他の記憶クラスタよりも利用可能な容量が少ない場合に、或いは、他の任意の望ましい目的のために、必要である場合がある。

【００６８】

このように、図１０の例において、５を法とするハッシュ値及びタイムスタンプの計算の五つの可能な結果のうち、唯一の結果（即ち、０）はクラスタＣ１上へのデータのユニットの記憶をもたらし、他方、二つの結果（即ち、１又は２）はクラスタＣ２上へのデータのユニットの記憶をもたらし、残りの二つの結果（即ち、３又は４）は、クラスタＣ３上へのデータのユニットの記憶をもたらす。結果として、このアルゴリズムを使用して処理されたデータのユニットの約５分の１がクラスタＣ１上に記憶され、他方、データのユニットの約５分の２がクラスタＣ２上に記憶され、データのユニットの約５分の２がクラスタＣ３上に記憶される。

【００６９】

特定の記憶クラスタを選択するための本明細書に記載の種々の技術を、クラスタ内の記憶場所を選択するための任意の適切な技術と併せて利用することは可能であり、本発明がこの点に関して限定されないことは当然のことであろう。例えば、記憶クラスタを選択するのには、固有記憶場所アドレス指定技術を利用するが、クラスタ内の記憶場所には利用しない、或いは、その逆も可能であり、選択技術の任意の組み合わせが利用可能である。

【００７０】

本発明の上述の実施形態は、任意の種々の方法により具体化可能である。例えば、その実施形態は、ハードウェア、ソフトウェア、又はそれらの組み合わせを用いて具体化され得る。ソフトウェア内で具体化された場合、そのソフトウェアのコードは、単一のコンピュータ内で実現された、或いは、複数のコンピュータ間で分散して実現された、任意の適切なプロセッサ又はプロセッサの集合の上で実行可能である。上述の機能を実行する任意の部品又は部品の集合を、一般的に、上述の機能を制御する一つ以上の制御装置とみなすことが可能であることは当然のことであろう。その一つ以上の制御装置は、例えば、専用ハードウェア、或いは、上記機能を実行するためにマイクロコード又はソフトウェアを利用してプログラムされた汎用ハードウェア（例えば、一つ以上のプロセッサ）を伴う種々の方法により具体化可能である。

【００７１】

この点に関して、本発明の実施形態の一つの具体化は、プロセッサ上で実行された場合に、本発明の実施形態の上述の機能を実行するコンピュータ・プログラム（即ち、複数の命令）がエンコードされた少なくとも一つのコンピュータ可読媒体（例えば、コンピュータ・メモリ、フロッピー・ディスク、コンパクト・ディスク、テープ等）を備えていることは、当然のことであろう。そのコンピュータ可読媒体は持ち運び可能であるため、その上に記憶されたプログラムを任意のコンピュータ環境のリソースにロードして、本明細書に記載の本発明の態様を実行することが可能である。加えて、実行された場合に上述の機能を実行するコンピュータ・プログラムを参照する場合、それはホスト・コンピュータ上で動作するアプリケーション・プログラムに限定される訳ではないことは、当然のことであろう。寧ろ、本明細書において、コンピュータ・プログラムという用語は、一般的な意味において、本発明の上述の態様を実行するプロセッサをプログラムするのに利用可能な任意の型のコンピュータ・コード（例えば、ソフトウェア又はマイクロコード）を参照するのに利用される。

【００７２】

プロセスがコンピュータ可読媒体内で具体化されている本発明の幾つかの実施形態によれば、コンピュータにより具体化されたそのプロセスは、その実行の間、（例えば、ユーザから）手動による入力を受信し得ることは、当然のことであろう。

【００７３】

上述の種々の例において、内容アドレスは、英字「Ａ」から「Ｚ」を含むものであると説明された。これらの内容アドレスは例として与えたに過ぎず、内容アドレスは、任意の英数字、ビット列、又は任意の他の適切な文字を含み得るものであり、本発明がこの点に

関して限定されないことは、当然のことであろう。

【００７４】

本明細書で使用されている表現及び専門用語は説明を目的としており、限定と解釈するべきではない。「含む」、「備える」、「有する」、「包含する」、「伴う」、及びそれらの変形が使用された場合、後に列挙される構成要素及び追加の構成要素を網羅することを意味する。

【００７５】

本発明の幾つかの実施形態を詳細に説明したが、直ちに種々の改良及び改善が当業者により成されるであろう。そのような改良及び改善は、本発明の技術思想及び範囲内に収まるものであると意図される。従って、前述の説明は、例に過ぎず、限定を意図している訳ではない。本発明は、以下の請求項及びその均等物使用によって規定されるものとしてのみ限定される。

【図面の簡単な説明】

【００７６】

【図１】本発明の実施形態を実行し得る分散型記憶環境のブロック図。

【図２】本発明の一つの実施形態による、ＡＰＩを介して、複数の記憶環境又はクラスタと相互作用するホスト・コンピュータのブロック図。

【図３】本発明の別の実施形態による、外部アプライアンスを介して、複数の記憶クラスタと相互作用するホスト・コンピュータのブロック図。

【図４】本発明のさらなる実施形態による、複数の記憶クラスタを含む記憶環境と相互作用するホスト・コンピュータのブロック図。

【図５】本発明の一つの実施形態による、データを記憶環境上に記憶する例示の方法を示すブロック図。

【図６】本発明の一つの実施形態による、複数のクラスタ間からデータのユニットの記憶場所を判断するための固有記憶場所の利用を示すブロック図。

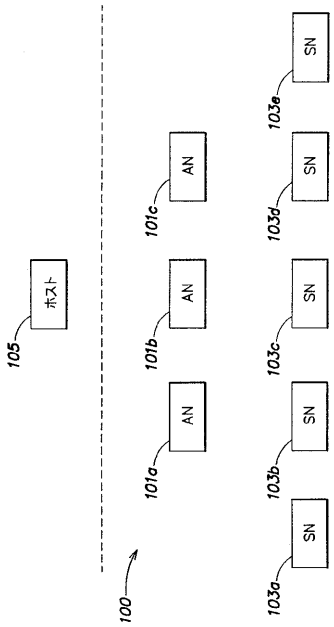
【図７】本発明の一つの実施形態による、記憶されたデータにアクセスするのに利用可能な内容アドレスを示す図。

【図８】本発明の一つの実施形態による、複数のクラスタのうちの一つの記憶クラスタに記憶されたデータを確認する際に利用される記憶場所識別子の生成を示す図。

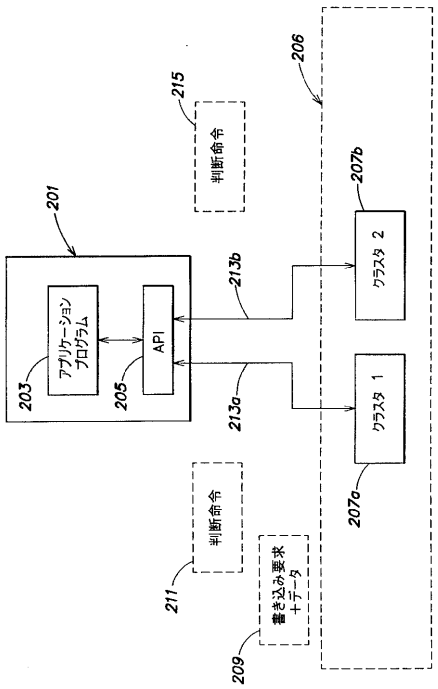
【図９】本発明の一つの実施形態による、本発明の態様が利用可能な内容アドレス環境において使用可能であり、かつ複数のｂｌｏｂを参照する内容記述ファイル（ＣＤＦ）の図。

【図１０】本発明の一つの実施形態による、複数の記憶環境にわたるデータ記憶に利用される記憶アルゴリズムを追跡するのに使用され得る例示のレコードを示す図。

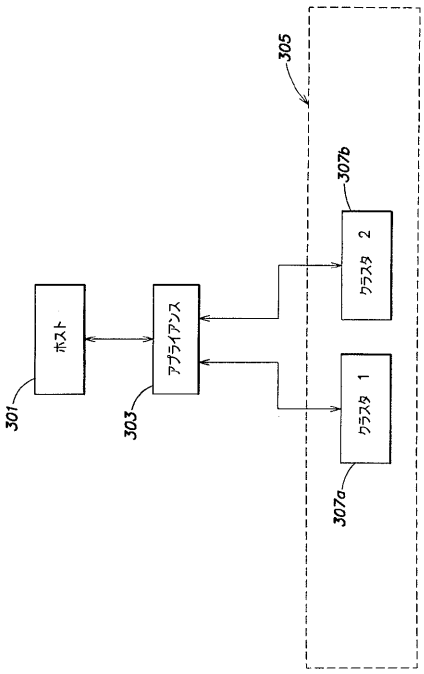
【図１】


ホスト
105
AN
101a
AN
101b
AN
101c
SN
103a
SN
103b
SN
103c
SN
103d
SN
103e
100

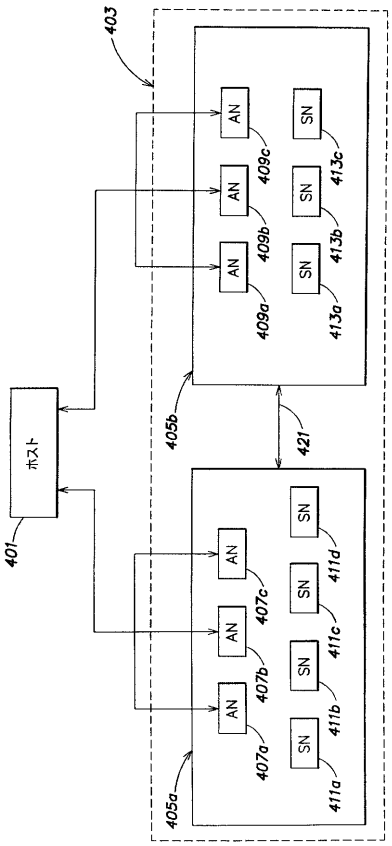
【図２】


201
アプリケーション プログラム
203
API
205
判断命令
215
206
クラスタ 2
207b
クラスタ 1
207a
213b
213a
判断命令
211
書き込み要求 +データ
209

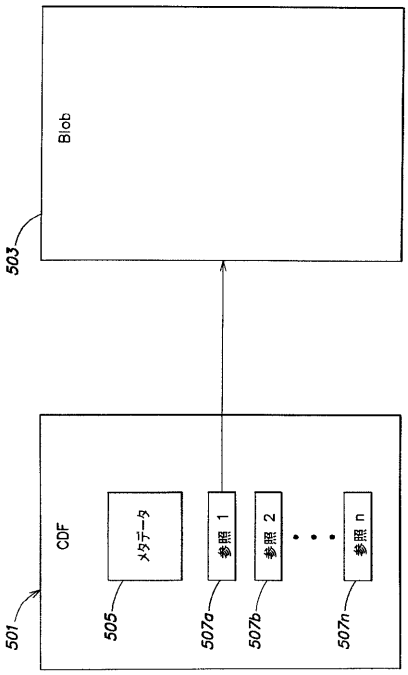
【図３】


ホスト
301
アプライアンス
303
305
クラスタ 2
307b
クラスタ 1
307a

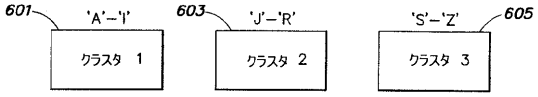
【図４】


403
ホスト
401
AN
409a
AN
409b
AN
409c
SN
413a
SN
413b
SN
413c
405b
421
AN
407a
AN
407b
AN
407c
SN
411a
SN
411b
SN
411c
SN
411d
405a

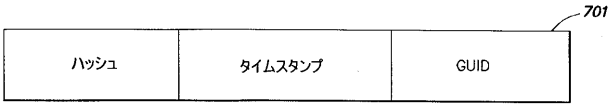
【図５】


Blob
503
CDF
メタデータ
参照 1
参照 2
参照 n
501
505
507a
507b
507n

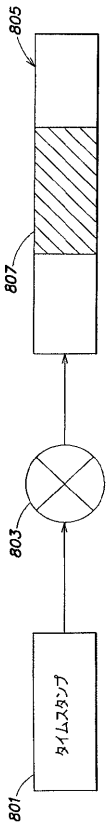
【図６】


601
'A'－'I'
クラスタ 1
603
'J'－'R'
クラスタ 2
'S'－'Z'
605
クラスタ 3

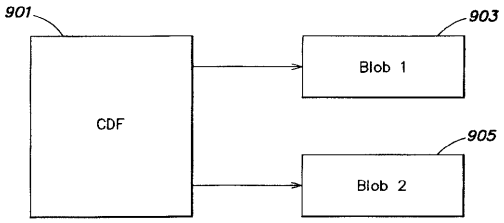
【図７】


701
ハッシュ
タイムスタンプ
GUID

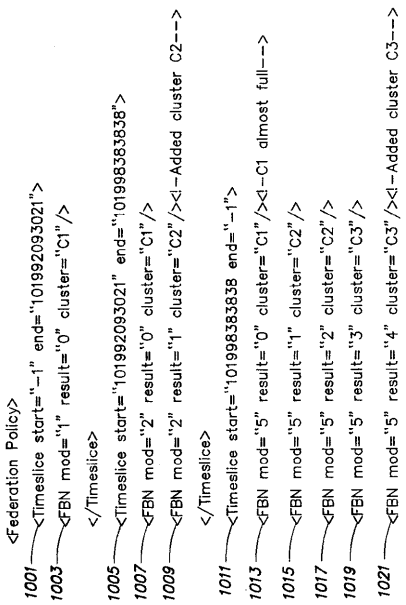
【図８】


805
807
803
タイムスタンプ
801

【図９】


901
CDF
903
Blob 1
905
Blob 2

【図１０】


<Federation Policy>
1001 <Timeslice start="-1" end="101992093021">
1003 <FBN mod="1" result="0" cluster="C1" />
</Timeslice>
1005 <Timeslice start="101992093021" end="101998383838">
1007 <FBN mod="2" result="0" cluster="C1" />
1009 <FBN mod="2" result="1" cluster="C2" /><!-Added cluster C2-->
</Timeslice>
1011 <Timeslice start="101998383838 end="-1">
1013 <FBN mod="5" result="0" cluster="C1" /><!-C1 almost full-->
1015 <FBN mod="5" result="1" cluster="C2" />
1017 <FBN mod="5" result="2" cluster="C2" />
1019 <FBN mod="5" result="3" cluster="C3" />
1021 <FBN mod="5" result="4" cluster="C3" /><!-Added cluster C3-->
</FederationPolicy>

フロントページの続き

(72)発明者 マイケル キリアン
アメリカ合衆国 ０１４５１ マサチューセッツ州 ハーバード キャンドルベリー レーン ２５
(72)発明者 トム トゥーゲルス
ベルギー国 ２９００ スコテン ローゼンラーン ２６
(72)発明者 ヤン バン リール
ベルギー国 Ｂ－２４４０ ヘール グーレイント ４３セー
(72)発明者 カール ダルイン
ベルギー国 ８５００ コルトレイク サン－エローイスドリーフ １０
(72)発明者 クリストフ ブーオン
ベルギー国 ４９１０ ラ リード ルート デュ カナダ １６５

Ｆターム(参考) 5B075 KK03 KK13 KK33 NK06 NK45 PQ05
5B082 CA01

【外国語明細書】



# METHODS AND APPARATUS FOR INCREASING DATA STORAGE CAPACITY

## Field of the Invention

The present invention relates to data storage and, more particularly, to methods and apparatus for increasing data storage capacity.

## Description of the Related Art

The capacity and performance of a data storage system depends on the physical resources of the storage system. For example, the quantity of data that a storage system is capable of storing is dependent on the number and capacity of the physical storage devices that the storage system possesses. As the quantity of data stored on the storage system approaches the storage capacity of the storage system, it may be desired to increase the storage system capacity by adding additional physical storage devices to the storage system. However, there may be physical limits imposed by the hardware configuration of the storage system on the number of storage devices that the storage system may have. Consequently, when a storage system approaches or nears it storage capacity, it may no longer be possible or desirable to add more physical storage devices to the storage systems. Rather, if it is desired to increase storage capacity, one or more additional storage systems may be used.

## Summary of the Invention

One illustrative embodiment is directed to a method of accessing a unit of data stored in a storage environment that includes a plurality of storage clusters. The method comprises acts of: receiving a request from a host computer to locate the unit of data previously stored in the storage environment; and in response to receipt of the request, determining on which one of the plurality of storage clusters the unit of data is stored. Another illustrative embodiment is directed to at least one computer readable medium encoded with instructions that, when executed on a computer system, perform the above-described method.

A further illustrative embodiment is directed to an apparatus for storing data. The apparatus comprises: an input that receives a request from a host computer to locate a unit of data on at least one of a plurality of storage clusters in a storage environment, and

at least one controller, coupled to the input, that: receives the request from the input; and in response to receipt of the request, determines on which of the plurality of storage clusters the unit of data is stored.

Another illustrative embodiment is directed to a method of accessing a unit of data stored in a storage environment that includes a plurality of storage clusters, the storage environment storing data for a host computer. The method comprises acts of: receiving a request from an application program executing on the host computer to store a unit of data; and in response to receipt of the request, selecting one of the plurality of storage clusters to store the unit of data in a manner that is transparent to the application program so that the application program is not made aware that the selected one of the plurality of storage clusters stores the unit of data. A further illustrative embodiment is directed to at least one computer readable medium encoded with instructions that, when executed on a computer system, perform the above-described method.

A further illustrative embodiment is directed to an apparatus for storing data. The apparatus comprises: an input that receives a request from an application program executing on a host computer to store a unit of data on at least one of a plurality of storage clusters in a storage environment, and at least one controller, coupled to the input, that: receives the request from the input; and in response to receipt of the request, selects one of the plurality of storage clusters to store the unit of data in a manner that is transparent to the application program so that the application program is not made aware that the selected one of the plurality of storage clusters stores the unit of data.

Another illustrative embodiment is directed to a method of accessing a unit of data stored in a storage environment that includes a plurality of storage clusters. The method comprises acts of: receiving a request from an application program executing on a host computer to store a unit of data; selecting one of the plurality of storage clusters to store the unit of data; and storing the unit of data on the selected one of the plurality of storage clusters without storing information identifying the selected one of the plurality of storage clusters. A further illustrative embodiment is directed to at least one computer readable medium encoded with instructions that, when executed on a computer system, perform the above-described method.

Another illustrative embodiment is directed to an apparatus for storing data comprising: an input that receives a request from an application program executing on a host computer to store a unit of data on at least one of a plurality of storage clusters in a

storage environment, and at least one controller, coupled to the input, that: receives the request from the input; selects one of the plurality of storage clusters to store the unit of data; and stores the unit of data on the selected one of the plurality of storage clusters without storing information identifying the selected one of the plurality of storage clusters.

## Brief Description of the Drawings

Figure 1 is a block diagram of a distributed storage environment on which embodiments of the present invention may be implemented;

Figure 2 is a block diagram of a host computer interacting with multiple storage environments or clusters via an API, in accordance with one embodiment of the invention;

Figure 3 is a block diagram of a host computer interacting with multiple storage clusters via an external appliance, in accordance with another embodiment of the invention;

Figure 4 is a block diagram of a host computer interacting with a storage environment that includes multiple storage clusters, in accordance with a further embodiment of the invention;

Figure 5 is a block diagram illustrating an exemplary manner in which data may be stored on a storage environment, in accordance with one embodiment of the invention;

Figure 6 is a block diagram illustrating the use of intrinsic locations to determine the storage location of a unit of data among multiple clusters, in accordance with one embodiment of the invention;

Figure 7 is a diagram illustrating a content address that can be used to access data stored in accordance with one embodiment of the invention;

Figure 8 is a diagram illustrating the generation of a location identifier for use in locating data stored among one of multiple clusters, in accordance with one embodiment of the invention;

Figure 9 is a diagram of a content descriptor file (CDF) that references multiple blobs and can be used in a content addressable environment on which aspects of the

present invention can be employed, in accordance with one embodiment of the invention; and

Figure 10 is a diagram illustrating an exemplary record that may be used to track storage algorithms used to store data among multiple storage clusters, in accordance with one embodiment of the invention.

## Detailed Description

On application for aspects of the present invention described herein relates to content addressable storage in a distributed storage environment that includes a plurality of interconnected independent nodes. Such an environment is described in the commonly assigned U.S. applications shown in Table 1, each of which is incorporated herein by reference in its entirety. The applications listed in Table 1 are referred to collectively herein as the Related applications.

**Table 1**

| Title | Serial No. | Filing Date |
|---|---|---|
| Content Addressable Information, Encapsulation, Representation, And Transfer | 09/236,366 | January 21, 1999 |
| Access To Content Addressable Data Over A Network | 09/235,146 | January 21, 1999 |
| System And Method For Secure Storage Transfer And Retrieval Of Content Addressable Information | 09/391,360 | September 7, 1999 |
| Method And Apparatus For Data Retention In A Storage System | 10/731,790 | December 9, 2003 |
| Methods And Apparatus For Facilitating Access To Content In A Data Storage System | 10/731,613 | December 9, 2003 |
| Methods And Apparatus For Caching A Location Index In A Data Storage System | 10/731,796 | December 9, 2003 |

| | | |
|---|---|---|
| Methods And Apparatus For Parsing A Content Address To Facilitate Selection Of A Physical Storage Location In A Data Storage System | 10/731,603 | December 9, 2003 |
| Methods And Apparatus For Generating A Content Address To Indicate Data Units Written To A Storage System Proximate In Time | 10/731,845 | December 9, 2003 |
| Methods And Apparatus For Modifying A Retention Period For Data In A Storage System | Attorney Docket No: E0295.70195US00 Serial No. Not Yet Assigned | January 21, 2004 |
| Methods And Apparatus For Extending A Retention Period For Data In A Storage System | Attorney Docket No: E0295.70200US00 Serial No. Not Yet Assigned | January 21, 2004 |
| Methods And Apparatus For Indirectly Identifying A Retention Period For Data In A Storage System | Attorney Docket No: E0295.70201US00 Serial No. Not Yet Assigned | January 21, 2004 |
| Methods And Apparatus For Indirectly Identifying A Retention Period For Data In A Storage System | Attorney Docket No: E0295.70202US00 Serial No. Not Yet Assigned | January 21, 2004 |

An example of a distributed storage environment 100 is shown in Figure 1. Distributed storage environment 100 includes a plurality access nodes 101a-101c and a plurality of storage nodes 103a-103e. Access nodes 101 may receive and respond to access requests from a host computer 105, and storage nodes 103 may store data sent to storage environment 100 by host computer 105. Access nodes 101 and storage nodes 103 may be coupled by a network (not shown) and communicate over the network such that each node may make its presence on the network known to the other nodes. In this manner, the nodes may operate together to process access requests and store data for host computer 105.

Each node may include processing resources (e.g., processor and memory) and storage devices. The nodes communicate with each other to store data, respond to access

requests, and perform other environment functions. To a user of the storage environment (e.g., the host computer 105 or an application program executing thereon), the storage environment may appear as single entity. That is, the user need not be aware that the storage environment includes a plurality of separate nodes or on which storage node a certain unit of data is stored or mirrored.

To increase the storage capacity of the storage environment 100, more storage nodes may be added and coupled to the network. These additional storage nodes may make their presence known on the network, thus allowing access nodes 101 to employ the additional storage in the storing of data. Adding more storage nodes to the storage network without increasing the number of access nodes may result in the access nodes acting as a bottleneck for the storage environment and a degradation in performance. Thus, it may desirable when increasing the number of storage nodes to also increase the number of access nodes.

Storage environment 100 may perform a number of functions described in the above-referenced Related applications, such as determining on which storage node 103 to store data in response to a write request from host 105, determining on which storage node 103 data is stored in response to a read request from host 105, performing garbage collection of data that may be deleted from the storage environment, enforcing retention periods that specify a period of time that data should not be deleted from the storage environment, mirroring data (i.e., creating one or more mirror copies on different nodes of the storage environment), self-healing to compensate for failure of one or more nodes, and other functions. Such functions may be performed by storage nodes, access nodes, or both, and performing such functions may cause network traffic between the nodes.

For example, to perform self-healing functions, other nodes may detect when a node fails. In response, the environment 100 may re-direct access requests to data stored on the failed node to other nodes that store a mirrored copy of that data and may build another mirror for continued fault tolerance. A node may broadcast keep-alive messages on the network to indicate that it is operational and has not failed. If keep-alive messages from that node are not received by other nodes, the other nodes may determine that the node has failed. Adding more nodes to the storage environment causes more keep-alive messages to be transmitted on the network and results in more network traffic.

As another example, the storage environment 100 may maintain an index such as the blob location index (BLI) described in the Related applications, to aid in locating

data. The BLI may specify on which storage node units of data are stored. Each access or storage node in the network may be responsible for administering a portion of the BLI. Because the BLI may be distributed across the access and/or storage nodes, maintaining and updating the BLI when units of data are written to or deleted from the storage environment causes network traffic to be generated among nodes. Adding more nodes may cause the administration responsibilities of the BLI to be shared among a greater number of nodes, thus casing a greater amount of network traffic to be generated.

Other functions, including some described in the Related applications, such as performing garbage collection, locating content on the storage environment (e.g., via a broadcast message to all nodes), and re-ranging the BLI (i.e., when nodes are added or removed from the storage environment), may cause a greater amount of network traffic as nodes are added to the storage environment. Such increased network traffic may result in a decrease in performance of the storage environment.

As mentioned above, storage nodes 103 may be added to the storage environment to increase the storage capacity of the storage environment. Additionally, access nodes 101 may be added to counteract degradation in performance caused by adding the additional storage nodes. However, Applicants have appreciated that because adding access nodes and storage nodes causes increased network traffic, once a certain number of nodes in the storage environment is reached, the performance benefit gained by adding additional nodes is offset at least somewhat by the increased network traffic generated by the nodes in the storage environment, as well as the increased amount of processing resources used in performing the infrastructure functions (such as those described above) that support the storage environment. Thus, as additional nodes are added to the storage environment to increase storage capacity and/or performance, the overall performance of the storage environment may increase less than expected, or might in some cases even decrease. This performance impact is referenced below as performance degradation, which term is used herein to refer to not only actual decreases in performance, but also performance improvements that are less than would be expected to be achieved through the addition of an additional storage resource.

One embodiment of the invention is directed to providing additional storage clusters that work together to provide combined storage, but that are independent so that the addition of an additional storage node in one cluster does not raise the performance degradation issues discussed above for other clusters. As used herein, the term storage

cluster refers to a group of one or more interconnected nodes that share at least one software utility that logically connects them. For example, the nodes of a storage cluster may share a self-healing software utility, such that nodes in a cluster monitor keep-alive messages transmitted by other nodes in that cluster, but not by nodes outside the cluster. The nodes may also share a common BLI so that the BLI of one storage cluster may be independent of the BLI of any other storage cluster. The nodes of a storage cluster may also, or alternatively, share other utilities such as a garbage collection utility and/or a data mirroring utility that keeps track of where data stored in the cluster is mirrored. In one embodiment, each node in a cluster knows the address (e.g., an IP address) of every other node in the cluster, although the present invention is not limited in this respect.

Because nodes in a storage cluster do not share software utilities with nodes outside the cluster (e.g., in a different storage cluster), the internal network traffic and use of processing resources of the nodes within one storage cluster does not affect the performance of any other storage cluster. Consequently, an additional storage cluster may be added to a storage environment to increase storage capacity without experiencing the above-described performance degradation issues.

It should be appreciated that the description of the types of software utilities that may be shared among nodes in a cluster (but not shared among nodes in different clusters) is provided merely for the purposes of illustration, and that the aspects of the present invention described herein are not limited to use in a storage environment wherein the nodes in a storage cluster share the particular types of software utilities discussed above, or any other particular type of software utility. Similarly, while examples of a storage environment having characteristics such as those described in the Related applications are described herein for illustrative purposes, it should be appreciated that the aspects of the present invention described herein are not limited in this respect, and can be employed in storage environments having numerous other configurations. For example, the reference to a storage cluster comprising one or more interconnected nodes is not limited to an access node and storage node configuration as described in connection with the Related applications, as the reference to a node can refer to any device or collection of devices that function together as an entity.

In one embodiment of the invention, multiple storage clusters are managed in a manner that is transparent to users (e.g., host computers or application programs executing thereon) of the storage environment, so that users need not know on which

storage cluster a particular unit of data is stored to access that unit of data. Thus, application programs that store data to and retrieve data from the storage environment may treat the multiple storage clusters as a single storage environment, without knowledge of on which cluster data written to the storage environment is stored. Thus, in one embodiment of the invention, when an application program (e.g., on a host) issues a write request for a unit of data to the storage environment, it is determined on which storage cluster the unit of data is to be written, and when the application issues a read request, it is determined on which storage cluster the data is stored, both in a manner transparent to the application program.

The present invention is not limited to any particular implementation technique as to where the aspects of the computer system that determine on which storage cluster to store a unit of data and that retrieve a previously written unit of data from the appropriate cluster are practiced. In one embodiment, these functions may be performed by an application programming interface (API) on the host computer. This implementation is described below in connection with Figure 2.

Figure 2 shows a host computer 201 that communicates with a storage environment 206 that includes two storage clusters, 207a and 207b. Host computer 201 executes an application program 203 that reads and writes units of data, and an application programming interface (API) 205 that is adapted to interface with the storage environment 206. Host 201 may be, for example, a server that provides resources (e.g., data storage, email, and other services), a client (e.g., a general-purpose computer environment operated by a user), a network component (e.g., a switch) or any other type of computer. Host 201 is coupled by a communication links 213a and 213b to storage clusters 207a and 207b of storage environment 206 so that the host 201 can use the storage environment 206 to store and retrieve data. The links 213a and 213b can be any type of communication link, including any type of network or direct connection.

Application program 203 may be any computer application program that a user or administrator of host 201 uses and that stores data to and/or retrieves data from storage environment 206. Application program 203 is linked with API 205 so that application program 203 may use API 205 to communicate with the storage clusters 207. In one embodiment, application program 203 need not be aware that storage environment 206 even includes multiple storage clusters, as the management of data storage between the clusters 207 may be done in a manner transparent to the application program 203.

API 205 may be implemented in any suitable way. For example, API 205 may include computer object code that is provided to the programmer of application program 203. The computer object code includes routines that may be used to communicate with the storage environment. When linked with API 205, an application program may call these routines to communicate with storage environment 206. API 205 may be linked with any number of applications, and shields the programmer of application program 203 from the internal protocol by which storage environment 206 communicates.

When application program 203 calls a routine or routines of API 205 to write data to storage environment 206, API 205 may select a storage cluster to store the data and send a write request 209 including the data to the appropriate storage cluster. The selection of the storage cluster may be performed using any suitable technique. For example, the API 205 may employ a round robin scheme by which the clusters in the storage environment are selected sequentially for each write request. In the example in Figure 2, API 205 could alternate between selection of storage cluster 207a and storage cluster 207b for processing write requests. Alternatively, the API 205 may select a storage cluster to store the data based on, for example, the available storage capacity on each storage cluster, the load on each storage cluster, the size or type of the data to be stored, or any other suitable criteria. The load on a storage cluster may be determined, for example, based on factors such as how many objects (e.g., blobs or CDFs) are being processed by the storage cluster, how busy the CPU(s) of the storage cluster are, how busy the disk(s) of the storage cluster are, and/or any other suitable factor.

When application program 203 calls a routine or routines of API 205 to read a previously stored unit of data from the storage environment, API 205 may determine on which storage cluster the requested unit of data is stored. This can be done in any suitable way. For example, the API 205 may store information that identifies the cluster on which each data unit is stored. However, in another embodiment no such information is stored, and the API 205 determines where the data is stored in response to a read request.

In one embodiment, API 205 sends a command that identifies the requested unit of data to each storage cluster in the storage environment and requests the storage cluster to determine if the specified unit of data is stored in that cluster. The command can be a read command, or a command (referred to herein as an "exists command") that does not seek to read the data unit but only to determine on which cluster it is stored.

The commands may be sent to the storage clusters either serially or in parallel, as the invention is not limited in this respect. For example, if sending the commands serially, API 205 may first send an exists command 211 to storage cluster 207a. If cluster 207a responds that the unit of data is stored therein, API 205 may then send a read request for that unit of data to storage cluster 207a and need not send any additional exists commands to other clusters. If cluster 207a responds that the unit of data is not stored therein, API 205 may then send another exists command 215 to cluster 207b. If cluster 207b responds that the specified unit of data is stored therein, API 205 may then send a read request to cluster 207b for the unit of data. If API 205 sends the exists commands in parallel, then exists command 211 and exists command 213 may be sent to clusters 207a and 207b, respectively, at approximately the same time. When one of the storage clusters responds to API 205 that the specified unit of data is stored therein, then API 205 may send a read request for the unit of data to that storage cluster. As mentioned above, instead of employing exists commands followed up by read requests, API 205 may only issue read requests (i.e., without issuing exists commands) to the storage clusters either serially or in parallel.

Sending the exists commands or read requests in parallel provides the advantage that the time spent in locating the unit of data is reduced because each cluster is performing a search for the unit of data simultaneously. However, these simultaneous searches consume the processing resources (and thereby impact performance) on each of the clusters, even though such searches will fail on many of the storage clusters. Conversely, sending the exists or read commands serially may increase the time spent in locating the unit of data because the exists command is not sent to a storage cluster until the search on the preceding storage cluster completes, but the performance of the storage environment may not be impacted as greatly because some clusters will not need to perform a search for units of data found before the exists command or read request is sent to those clusters. The search within each cluster can be performed in any suitable manner, as the present invention is not limited in this respect. Examples of searching techniques are described in the above-referenced Related applications.

To reduce the overall processing load incurred by the storage clusters in searching for a unit of data, the API 205 may orchestrate a layered searching approach wherein clusters initially perform a relatively cursory (and not computationally intensive)

search, and perform a more thorough search only when necessary (e.g., when the cursory search among all clusters does not result in a successful search).

For example, as described in the Related applications, each cluster may include a BLI that identifies some of the data units stored on the cluster. A relatively cursory search may involve initially searching only the BLI in a cluster, which is relatively quick and comparatively inexpensive. If a BLI search (either serially or in parallel) fails for all clusters in the environment, a more thorough search may be performed. It should be appreciated that the BLI search is merely one example of a technique for performing a relatively cursory search and that other techniques are possible, as the aspect of the present invention that employs a layered searching approach is not limited in this respect. Further, it should be appreciated that the aspects of the present invention described herein are not limited to use with a storage environment having the configuration shown in the Related applications, and is not limited to storage clusters that employ a BLI.

In the example of Figure 2, storage environment 206 is shown as having two storage clusters. It should be appreciated that this is given only as an example, as storage environment 206 may include three or more storage clusters, as the invention is not limited in this respect. Similarly, Figure 2 shows only a single host computer 201 sending data to and receiving data from storage environment 206. However, it should be appreciated the storage environment 206 may operate with many more host computers, as the invention is not limited in this respect.

In the environment discussed above, the aspects of the present invention that relate to selecting of the storage clusters to store the data, and/or the determining of which of the clusters stores a previously-written unit of data is implemented by the API. However, it should be appreciated that this explanation is provided merely for illustrative purposes, as the present invention is not limited in this respect. As mentioned above, an advantage of one illustrative embodiment of the present invention is that the implementation of the storage environment among multiple clusters is transparent to an application program executing on the host, so that the application program need not even be aware that multiple clusters exist, and need not be concerned with the overhead associated with managing multiple clusters. However, other aspects of the present invention are not limited in this respect, such that the management of the multiple clusters can alternatively be implemented in a manner that is not transparent to the application program. In addition, whether the management of multiple clusters is

transparent to the application program or not, such management need not be implemented in an API in the manner described above, as it can alternatively be implemented in other locations of the computer system. For example, in one embodiment of the present invention, the management of multiple clusters can be implemented within the clusters themselves. In this embodiment, the API may see only a single cluster, but the clusters are capable of communicating among themselves to manage the distribution of data among the multiple clusters. Alternatively, the API may see all the access nodes of all the clusters, but may not be aware that these nodes are divided amongst two or more clusters, such that the API may send any access request to any access node, regardless of which cluster the access node forms a part. In an alternative embodiment of the present invention, the management of the multiple clusters can be implemented in an appliance that it is separate from any of the clusters and from the host. The appliance can take any of numerous forms, and may be, for example, a network component (e.g. a router) that forms part of a network for interconnecting the host computer with the storage clusters.

Figure 3 provides one illustrative representation of the embodiment in which the determination of on which storage cluster to store a unit of data (in the case of a write request) and on which storage cluster a previously written unit of data is stored (in the case of a read request) is made by an appliance 303. Appliance 303 receives access requests from a host 301 to storage environment 305. Appliance 303 may then route the access request to the appropriate storage cluster. Appliance 303 may be, for example, a computer coupled to host 301 and storage environment 305 (e.g., via a network) that performs functions that allow applications executing on host 301 to access storage clusters as a single storage environment. Alternatively, appliance 303 may be included within storage environment 305.

Appliance 303 may use any suitable technique to determine on which storage cluster to store a unit of data, and for determining on which storage cluster a previously written unit of data is stored, as the present invention is not limited in this respect.

As should be appreciated from the foregoing, with the embodiment of Figure 3, an application program or API executing on host computer 301 need not even be aware that storage environment 305 includes multiple storage clusters. Further, the application program or API may write a unit of data to or retrieve a unit of data from storage environment 305 without knowing where the particular unit of data is stored. The host

computer may simply send access requests to appliance 303, which then determines where to a write a unit of data or where a previously written unit of data is stored.

In the example of Figure 3, storage environment 305 is shown having two storage clusters. It should be appreciated that this is given only as an example, as storage environment 305 may include three or more storage clusters as the invention is not limited in this respect. Similarly, Figure 3 shows only a single host computer 301 sending data to and receiving data from storage environment 305 However, it should be appreciated the storage environment 305 may operate with many more host computers and the invention is not limited in this respect. Further, although Figure 3 depicts only a single appliance 303 operating between host 301 and storage environment 305, multiple appliances may be used. In this manner, a bottleneck, wherein all access requests to the storage environment are processed by a single appliance, may be avoided. Additionally, although in the example of Figure 2, an API is employed to communication between the host computer and the storage environment, some embodiments of the invention do not require the use of an API and may allow the application program to communicate directly with the storage environment.

Figure 4 provides one illustrative representation of the embodiment in which the storage clusters themselves may make the determination of on which storage cluster to store a unit of data (in the case of a write request) and on which storage cluster a previously written unit of data is stored (in the case of a read request or a delete request). In the example of Figure 4, a host computer 401 communicates with a storage environment 403 that includes two storage clusters, 405a and 405b. Storage cluster 405a includes a plurality of access nodes 407a-c and a plurality of storage nodes 411a-d, and storage cluster 405b includes a plurality of access nodes 409a-c and a plurality of storage nodes 413a-c in much the same manner as discussed above in connection with Figure 1. In this embodiment of the invention, at least one storage cluster is aware of the presence of the other storage cluster(s) in the storage environment, but each cluster is an independent cluster (e.g., each cluster has independent software utilities). As shown at 421 in Figure 4, there is a communication path among the clusters 405a, 405b to enable them to communicate. This communication path can be implemented in any suitable way, as the present invention is not limited in this respect.

When an access node 407 or 409 receives a request to retrieve a unit of data, the access node may determine whether the requested unit of data is stored in its cluster. If it

is not, the access node may send exists commands and/or read commands to the other storage clusters in the storage environment, either serially or in parallel, in much the same manner as described above.

Application programs executing on host computer 401 may send access requests (e.g., via an API) to the access nodes 407 and 409 of storage clusters 405a and 405b. In one embodiment, the application program and/or API is unaware that access nodes 407 and 409 belong to different storage clusters. From the point of view of the application program and/or API, access nodes 407 and 409 may receive access requests, regardless of which storage cluster stores a requested unit of data. Indeed, the application program and/or API need not even be aware that storage environment includes multiple storage clusters, such that where data is stored on the storage environment is transparent to the application program and/or API in one embodiment of the invention.

When an access node 407 or 409 receives a request to write a unit of data from host 401, the access node may determine whether to store the unit of data on its cluster or send the unit of data to another cluster for storage. Such a determination may be made based on any suitable factor, examples of which were discussed above (e.g., a round robin scheme, the size or type of the data to be stored, the available capacity on each of the storage clusters, the current processing load on each of the storage clusters).

Alternatively, instead of allowing an application program and/or API to send requests to any access nodes in the storage environment, the application program and/or API may only be permitted to send access requests to the access nodes of one storage cluster (e.g., cluster 405a). The access nodes of this storage cluster are then responsible for writing data to and retrieving data from the storage cluster of which they are a part, and/or other storage clusters in the storage environment of which the access nodes are not a part.

In one embodiment, aspects of the invention can be employed with content addressable storage (CAS) environments of the type described in the Related applications. In a content-addressable storage environment, data is stored using a content address generated based upon the content of the data itself. The content address may be generated by applying a hash function to the data to be stored. The output of the hash function may be used as at least part of the content address used in communication between the host and storage environment to refer to the data. The content address can

be mapped (e.g., within the storage environment) to one or more physical storage locations within the storage environment.

A unit of data in the architecture defined in the CAS applications is referred to as a blob. As illustrated in Figure 5, blob 503 may be, for example, a unit of binary data to be stored by a host on a storage environment (e.g., storage environment 105), such as, for example, a patient x-ray, company financial records, or any other type of data. When the blob 503 is stored to the content addressable storage environment, a unique address is generated for the blob 503 based upon its content.

Each blob 503 has at least one content descriptor file (CDF) 501 associated with it. CDF 501 may include metadata 505 and a plurality of references 507a, 507b,...,507n. A CDF may reference one or more blobs or other CDFs. Thus, the references 507 may be, for example, references to the blobs and/or CDFs referenced by CDF 501. Metadata 505 may, for example, include the creation date of CDF 501 (e.g., the date that CDF 501 was stored on the storage environment), a description of the content of blob 503, and/or other information.

As discussed above, a storage environment may employ a garbage collection utility to clean up blobs and CDFs that are available for deletion. In one implementation of the system described in the Related applications, a blob or CDF is available for deletion when it is not referenced by any CDFs.

Blobs and their corresponding CDFs may be written separately to the storage environment. In accordance with one embodiment of the invention, a blob and the CDF(s) that reference(s) it may be stored on different storage clusters. This may present challenges when used on a system that employs a garbage collection utility that relies on the absence of a CDF pointing to a blob to delete the blob, and that in response to deletion of a CDF updates any blobs previously referenced thereby to delete a pointer to the CDF. For example, when a CDF is deleted and any blobs that it references are stored on different storage clusters, the garbage collection utility may first search the storage cluster on which the CDF is stored for the referenced blobs. When it does not find the referenced blobs on that storage cluster, it may locate the blobs on other storage clusters (e.g., via exists commands and/or read requests), and indicate to the storage clusters that store the blobs that the reference to the deleted CDF should be removed. This process may incur significant overhead in communication between clusters. Furthermore, in some embodiments, it may be desirable to have each storage cluster not even be aware

that there are other related storage clusters in the storage environment, such that a cluster might not know to look to other clusters for a referenced blob when a CDF is deleted.

In one embodiment of the invention, the above-described challenges are alleviated by storing a blob and the CDFs that reference that blob on the same storage cluster. This may be accomplished in any suitable way. For example, the same placement algorithms discussed above with reference to Figures 2-4 can be employed, but a blob and the CDF(s) that reference it can be written to the storage environment in a single write transaction so that the blob and the CDF are written to the same storage cluster. If, at a later time, additional CDFs that reference the blob are written to the storage environment, the entity determining the location for storing the additional CDFs (e.g., API, appliance, or storage cluster) may first determine which storage cluster stores the blob and then store the additional CDFs to that same storage cluster.

In one embodiment of the invention, application programs keep track of previously stored data by storing the content addresses only of CDFs, but do not track the content addresses of blobs referenced thereby, as that information can be retrieved from the corresponding CDF. Thus, to access a blob, an application program may request retrieval of a CDF that references the blob. The application program may receive the requested CDF and determine the content address of the blob referenced by the CDF. The application program may then use this content address to send a read request for the blob. Because the blob and the CDF are stored on the same storage cluster in one embodiment of the invention, the entity that sent the request for the CDF can send a read request for the blob to the same storage cluster, without needing to perform a search to determine which storage cluster stores the blob.

In another embodiment of the invention, instead of using a search technique to locate data stored on a storage environment, a concept known as intrinsic location addressing may be used. Intrinsic location addressing relates to placing a unit of data on a particular storage cluster based on the content address of the unit of data. Thus, when the unit of data is later read, it is not necessary to perform a search to determine on which storage cluster the unit of data is stored. Instead, the storage cluster that stores the unit of data may again be determined based on the content address of the unit of data.

For example, selected bits of the content address may be used to determine on which storage cluster to store a unit of data. Each storage cluster may be assigned a range of addresses to be stored. For example, the first character of the content address of

a unit of data may determine on which storage cluster the unit of data is to be stored. The storage environment of Figure 6 includes three storage clusters. When the first character is 'A'-'I', the unit of data may be stored on storage cluster 601, when the first character is 'J'-'R', on storage cluster 603, and when the first character is 'S'-'Z', on storage cluster 605. It should be appreciated that this example is illustrative and not limiting. For example it need not be the first character that is examined to determine the storage cluster, and the ranges can differ from those shown so that they need not be of approximately equal size and need not be contiguous.

As discussed above, the content address of a unit of data may be based, at least in part, on the content of the unit of data, and may be generated in any suitable way, as the present invention is not limited in this respect. For example, the unit of data may be hashed using a hashing algorithm, such as MD5, to generate a portion of the content address. The content address may include other information such as a timestamp that indicates when the unit of data was first written to the storage environment and/or a guaranteed unique identifier (GUID) that ensures the content address is unique, as described in the Related applications. Thus, as shown in Figure 7, a content address 701 may include a hash value, a timestamp, and a GUID.

In some situations, the unit of data to be written may be too large to compute the hash value before it is desired to have selected a storage location on one of the storage clusters to store the unit of data. For example, the storage environment may not have a buffer large enough to store the entire unit of data while the hash is computed for the unit of data. Thus, in one embodiment of the invention, a technique is employed to enable a storage location in one of the storage clusters to be selected before the hash value of the content address is computed.

Because the hash value may not be available in some situations to select a storage cluster to store the unit of data, it may be desirable to use another portion of the content address (e.g., the timestamp or GUID portion) to select the appropriate storage cluster. As discussed in the Related applications, employing timestamp information can be beneficial in providing a temporal locality of reference for data units written at approximately the same time within a storage system such as one cluster. However, it is desired to not have all data units written at around the same time be written to the same cluster, as this would negatively impact performance by preventing clusters from processing write requests in parallel. Thus, as shown in Figure 8, to ensure that the

portion of the content address used to select the storage cluster (e.g., the timestamp) is sufficiently random, this portion of the content address may be hashed at 803 using an algorithm (MD5 or any other suitable algorithm) to generate a location identifier 805. A portion 807 (or all) of the location identifier 805 may be used to determine which storage cluster should be used to store the unit of data. In this manner, the pseudo-randomness of the hashing algorithm may be employed so that portion 807 is sufficiently random to have an even distribution across the storage clusters for units of data written contemporaneously.

In one embodiment, the location identifier 805 is not saved to assist in retrieving the data unit. Rather, when it is desired to read the unit of data, the portion 801 of the content address may be hashed again to generate location identifier 805 and the portion 807 of the location identifier may be used to determine on which of the storage clusters the unit of data is stored.

It should be appreciated that the use of intrinsic location addressing techniques for selecting a storage cluster to store a unit of data obviates the need to perform a search to locate the unit of data when a read request for the unit of data is issued.

It should be appreciated that when used in conjunction with a content addressable storage system that employs blobs, CDFs and a garbage collection utility as discussed above, the use of the intrinsic location addressing technique may result in a blob and corresponding CDF being stored on different clusters, which can present challenges as discussed above. Accordingly, one embodiment of the invention is directed to a technique for using intrinsic locations to select a storage cluster to store a unit of data in conjunction with a technique to ensure that a CDF and its referenced blob(s) are stored on the same storage cluster. In one embodiment, the technique for ensuring that blobs and its CDFs are stored on the same cluster involves using a common identifier in the content addresses of the CDF and its referenced blob(s), and using this common identifier to determine on which storage cluster to store the CDF and its referenced blobs.

For example, a CDF and its referenced blobs may be provided with the same GUID in their content addresses. A portion of this GUID (or a portion of a hash of the GUID) may be used to determine on which storage cluster to store a unit of data, so that the CDFs and blobs with the same GUIDs will map to the same storage cluster. For example, Figure 9 shows a CDF 901 that references two blobs, 903 and 905. In

accordance with one embodiment of the invention, these three units of data are stored on the same storage cluster as follows. First, the application program (e.g., 203 in Figure 2) sends blob 903 to the storage environment. In those embodiments where the host computer 201 has an API to interface with the storage environment, the API may generate a GUID as part of the content address of the blob 903. The facility for selecting a storage cluster (e.g., API 205, appliance 303 or one of the clusters) may select a storage cluster on which to store blob 903. The application program then sends blob 905 to the API. Because the API has not yet received a CDF, the API recognizes that blob 903 and 905 are part of the same transaction (i.e., both referenced by the same CDF) and may use the same GUID in the content address of blob 905. Consequently, blob 903 and 905 will be stored on the same storage cluster. The application program then sends CDF 901 to the API to be stored on the storage environment. The reception of a CDF by the API indicates to the API that the CDF is the last unit of data in the transaction. Thus, API assigns the same GUID to the content address of CDF 901, such that CDF 901 is stored on the same storage cluster as blobs 903 and 905.

In one embodiment, a blob cannot be accessed directly by an application without accessing it through a corresponding CDF. Thus, the CDF must first be read. In accordance with this embodiment, the GUID may be included in the content address of the CDF and used to locate the storage cluster on which the CDF is stored to facilitate a read of the CDF, but the GUID need not be provided in the content address of the blob because the entity charged with locating the storage cluster on which the blob is stored will recognize that the blob is stored on the same storage cluster as the previously read CDF.

When new storage clusters are added to the storage environment, it may be desired to change the selection algorithm for determining on which storage cluster a unit of data is stored so that an even distribution of data units across all of the storage clusters results. For example, if there are three storage clusters in the storage environment, a selection algorithm such as the GUID (in the content address) modulo three may be used to determine on which storage cluster the content is stored. If a fourth storage cluster is added, it may be desired to change the algorithm to GUID modulo four. However, if the new algorithm is applied to content addresses when units of data are read after the addition of the new cluster, the new algorithm may not properly locate data units that were stored on the storage environment prior to the change in algorithm.

Thus, one embodiment of the invention is directed to maintaining information that indicates what algorithm was in making a selection of the cluster for a previously stored data unit. In one embodiment this information identifies the selection algorithm used during certain time periods. Because the content address of the data unit may include a timestamp, the timestamp information may be used to index into the information to determine what algorithm was in place at the time the unit of data was stored. This algorithm can be applied to the appropriate portion of the content address to determine on which storage cluster the unit of data is stored.

An example of information that indicates what algorithm was in use during certain periods of time is shown in Figure 10. In Figure 10, line 1001 indicates a time frame for a particular selection algorithm, and indicates that the algorithm was used from time "-1" to time "1019922093021." A value of -1 indicates the beginning of the existence of the storage environment. Line 1003 indicates what algorithm was in effect during the time frame indicated by line 1001, and in the example of Figure 10 indicates that an algorithm called Fast Blob Naming or FBN was in use. Fast blob naming is an algorithm where a modulus of the hash value and timestamp of the content address is used to determine on which storage cluster a unit of data should be stored. The "mod" indicator indicates the modulus value used in the Fast Blob Naming algorithm. Thus, for example, in line 1003, an algorithm of hash value and timestamp modulo one was used. The result field indicates which results of the algorithm correspond to which storage clusters. In the example of line 1003, the only possible result when using a modulus of 1 is zero. Thus, when the result is zero, the unit of data is stored on storage cluster C1.

Line 1005 indicates that a new algorithm was in effect between time period "101992093021" and "101998383838." As shown in lines 1007 and 1009, the Fast Blob Naming algorithm was used with a modulus of two during that time period, so that when the result of the modular operation was zero, the data unit was stored on cluster C1, and when the result of the modular operation was one, the data unit was stored on cluster C2.

The timestamp of the content address of a unit of data may be used to determine which algorithm to employ in locating a unit of data. For example, if the timestamp of a unit of data were 10199209566, such a timestamp falls within the time period indicated in line 1005. Thus, the algorithm specified by lines 1007 and 1009 may be used to determine which storage cluster stores that unit of data.

Line 1011 indicates that a new algorithm came into effect at time "101998383838," and the end time of "-1" indicates that the algorithm is currently in effect. Lines 1013, 1015, 1017, 1019, and 1021 indicate that a modulus value of 5 was used in the Fast Blob Naming algorithm and that three clusters are currently in use (i.e., clusters C1, C2, and C3). Line 1013 indicates that if the result of the hash value and timestamp modulo five is zero, then the unit of data is stored on cluster C1. If the result is one or two, then the unit of data is stored on cluster C2. If the result is three or four, then the unit of data is stored on cluster C3. Consequently, lines 1013-1021 provide an example of how storage clusters in a storage environment may be weighted so that some storage clusters store more units of data than other storage clusters. Weighting clusters in this manner may be desirable, for example, if one storage cluster has less available capacity than the other storage clusters in the storage environment, or for any other desirable purpose.

Thus, in the example of Figure 10, of the five possible results of computing the hash value and timestamp modulo five, only one of those results (i.e., zero) causes a unit of data to be stored on cluster C1, while two of those results (i.e., one and two) cause the unit of data to be stored on cluster C2, and the remaining two results (i.e., three and four) cause the unit of data to be stored on cluster C3. Consequently, approximately one fifth of the units of data processed using this algorithm are stored on cluster C1, while approximately two fifths of the units of data are stored on cluster C2 and approximately two fifths of the units of data are stored on cluster C3.

It should be appreciated that the various techniques described herein for selecting a particular storage cluster can be used with any suitable technique for choosing storage locations within a cluster, as the present invention is not limited in this respect. For example, intrinsic location addressing techniques can be used to select a storage cluster but not the storage location within a cluster, or vice-versa, as any pairing of selection techniques can be used.

The above-described embodiments of the present invention can be implemented in any of numerous ways. For example, the embodiments may be implemented using hardware, software or a combination thereof. When implemented in software, the software code can be executed on any suitable processor or collection of processors, whether provided in a single computer or distributed among multiple computers. It should be appreciated that any component or collection of components that perform the

functions described above can be generically considered as one or more controllers that control the above-discussed functions. The one or more controllers can be implemented in numerous ways, such as with dedicated hardware, or with general purpose hardware (e.g., one or more processors) that is programmed using microcode or software to perform the functions recited above.

In this respect, it should be appreciated that one implementation of the embodiments of the present invention comprises at least one computer-readable medium (e.g., a computer memory, a floppy disk, a compact disk, a tape, etc.) encoded with a computer program (i.e., a plurality of instructions), which, when executed on a processor, performs the above-discussed functions of the embodiments of the present invention. The computer-readable medium can be transportable such that the program stored thereon can be loaded onto any computer environment resource to implement the aspects of the present invention discussed herein. In addition, it should be appreciated that the reference to a computer program which, when executed, performs the above-discussed functions, is not limited to an application program running on a host computer. Rather, the term computer program is used herein in a generic sense to reference any type of computer code (e.g., software or microcode) that can be employed to program a processor to implement the above-discussed aspects of the present invention.

It should be appreciated that in accordance with several embodiments of the present invention wherein processes are implemented in a computer readable medium, the computer implemented processes may, during the course of their execution, receive input manually (e.g., from a user).

In various examples described above, content addresses were described to include alphabetic characters 'A'-'Z'. It should be understood that these content addresses were given only as examples, and that content addresses may include any alphanumeric character, series of bits, or any other suitable character, as the invention is not limited in this respect.

The phraseology and terminology used herein is for the purpose of description and should not be regarded as limiting. The use of "including," "comprising," "having," "containing", "involving", and variations thereof, is meant to encompass the items listed thereafter and additional items.

Having described several embodiments of the invention in detail, various modifications and improvements will readily occur to those skilled in the art. Such

modifications and improvements are intended to be within the spirit and scope of the invention. Accordingly, the foregoing description is by way of example only, and is not intended as limiting. The invention is limited only as defined by the following claims and the equivalents thereto.

What is claimed is:

## CLAIMS

1. At least one computer readable medium encoded with instructions that, when executed on a computer system, perform a method of accessing a unit of data stored in a storage environment that includes a plurality of storage clusters, wherein the computer system comprises the storage environment and a host computer that stores data on the storage environment, and wherein the method comprises acts of:

receiving a request from the host computer to locate the unit of data previously stored in the storage environment; and

in response to receipt of the request, determining on which one of the plurality of storage clusters the unit of data is stored.

2. The at least one computer readable medium of claim 1, wherein the host computer executes an application program that stores data to and retrieves data from the storage environment, wherein the host further executes an application programming interface that interfaces the application program to the storage environment, and wherein the act of receiving is performed by the application programming interface.

3. The at least one computer readable medium of claim 1, wherein the computer system further comprises at least one communication link that couples the storage environment to the host computer and an appliance that monitors access requests from the host computer to the storage environment, and wherein the act of receiving the request further comprises an act of receiving, at the appliance, the request from the application.

4. The at least on computer readable medium of claim 3, wherein the act of detecting is performed by the appliance.

5. The at least one computer readable medium of claim 3, wherein the act of receiving the request further comprises an act of receiving the request at at least one of the plurality of storage clusters.

6. The at least one computer readable medium of claim 5 wherein the at least one of the plurality of storage clusters includes at least one access node that receives and

processes access requests, and wherein the act of receiving the request from the application program at the at least one of the plurality of storage clusters further comprises an act of receiving the request at the at least one access node.

7. The at least one computer readable medium of claim 1, wherein the act of determining comprises an act of performing a search for the unit of data on the plurality of storage clusters.

8. The at least one computer readable medium of claim 1, wherein the act of determining is performed by at least one of the plurality of storage clusters.

9. The at least one computer readable medium of claim 1, wherein the act of determining comprises locating the unit of data on at least one of the plurality of storage clusters without performing a search.

10. The at least one computer readable medium of claim 1, wherein the storage environment is a content addressable storage environment in which the unit of data stored in the storage environment is accessible by a content address that is based, at least in part, upon at least a portion of the content of the unit of data, and wherein the act of determining further comprises determining on which of the plurality of storage clusters the unit of data is stored based on the content address of the unit of data.

11. The at least one computer readable medium of claim 10, wherein the content address includes time information, based on when the unit of data was stored in the storage environment, and the act of determining comprises an act of determining on which of the plurality of storage clusters the unit of data is stored based, at least in part, on the time information of the content address of the unit of data.

12. The at least one computer readable medium of claim 11, wherein the act of determining further comprises an act of determining on which of the plurality of storage clusters the unit of data is stored based, at least in part, on a hash value of the time information of the content address of the unit of data.

13. The at least one computer readable medium of claim 10, wherein the act of determining further comprises acts of:

accessing information that specifies an algorithm that was used to select on which of the plurality of storage clusters the unit of data was stored, based on the content address of the unit of data; and

applying the algorithm to the content address of the unit of data to determine on which of the plurality of storage clusters the unit of data is stored.

14. The at least one computer readable medium of claim 13, wherein the information specifies a plurality of algorithms used by the storage environment and during which period of time each of the plurality of algorithms was used to store units of data.

15. The at least one computer readable medium of claim 14, wherein the information further specifies, for each one of the plurality of algorithms, at least one storage cluster that was in the storage environment during the period of time when the one of the plurality of algorithms was in effect.

16. At least one computer readable medium encoded with instructions that, when executed on a computer system, perform a method of accessing a unit of data stored in a storage environment that includes a plurality of storage clusters, wherein the computer system comprises the storage environment and a host computer that stores data on the storage environment, and wherein the method comprises acts of:

receiving a request from an application program executing on the host computer to store a unit of data; and

in response to receipt of the request, selecting one of the plurality of storage clusters as to store the unit of data in a manner that is transparent to the application program so that the application program is not made aware that the selected one of the plurality of storage clusters stores the unit of data.

17. The at least one computer readable medium of claim 16, wherein the host further executes an application programming interface that interfaces the application program to the storage environment, and wherein the act of receiving is performed by the application programming interface.

18. The at least one computer readable medium of claim 16, wherein the computer system further comprises at least one communication link that couples the storage environment to the host computer to the storage environment and an appliance that monitors access requests from the host computer, and wherein the act of receiving the request further comprises an act of receiving, at the appliance, the request from the application.

19. The at least one computer readable medium of claim 16, wherein the act of receiving the request further comprises an act of receiving the request directly from the application program.

20. The at least one computer readable medium of claim 19, wherein the act of receiving the request from the application program further comprises an act of receiving the request at at least one of the plurality of storage clusters.

21. The at least one computer readable medium of claim 16, wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data using a round-robin technique.

22. The at least one computer readable medium of claim 16, wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data based, at least in part, on an available storage capacity of each of the plurality of storage clusters.

23. The at least one computer readable medium of claim 16, wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data based on a load of at least one of the plurality of storage clusters.

24. The at least one computer readable medium of claim 16, wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data based on a size of the unit of data.

25. The at least one computer readable medium of claim 16, wherein the storage environment is a content addressable storage environment in which the unit of data stored in the storage environment is accessible by a content address that is based, at least in part, upon at least a portion of the content of the unit of data, and wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data based on the content address of the unit of data.

26. The at least one computer readable medium of claim 25, wherein the content address includes time information, based on when the unit of data was stored in the storage environment, and the act of selecting comprises an act of selecting one of the plurality of storage clusters to store the unit of data based, at least in part, on the time information of the content address of the unit of data.

27. The at least one computer readable medium of claim 26, wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data based, at least in part, on a hash value of the time information of the content address of the unit of data.

28. The at least one computer readable medium of claim 27, wherein the act of selecting further comprises an act of selecting one of the plurality of storage clusters to store the unit of data based, at least in part, on a hash of the GUID.

29. The at least one computer readable medium of claim 25, wherein the act of selecting further comprises acts of:

applying an algorithm to the content address of the unit of data to determine on which of the plurality of storage clusters to store the unit of data;

storing the algorithm in a record that indicates a time frame in which the algorithm was in use.

30. The at least one computer readable medium of claim 29, wherein the record specifies a plurality of algorithms used by the storage environment and during which period of time each of the plurality of algorithms was used to store units of data.

## ABSTRACT

A environment and method are provided for increasing the storage capacity of a data storage environment. Additional storage clusters may be added to the storage environment without affecting the performance of each individual storage cluster. When data is written to the storage environment, a selection may be made as to which storage cluster is to store the data. When data is read from the storage environment, it may be determined which storage cluster stores the data and the data may be retrieved from that storage cluster.

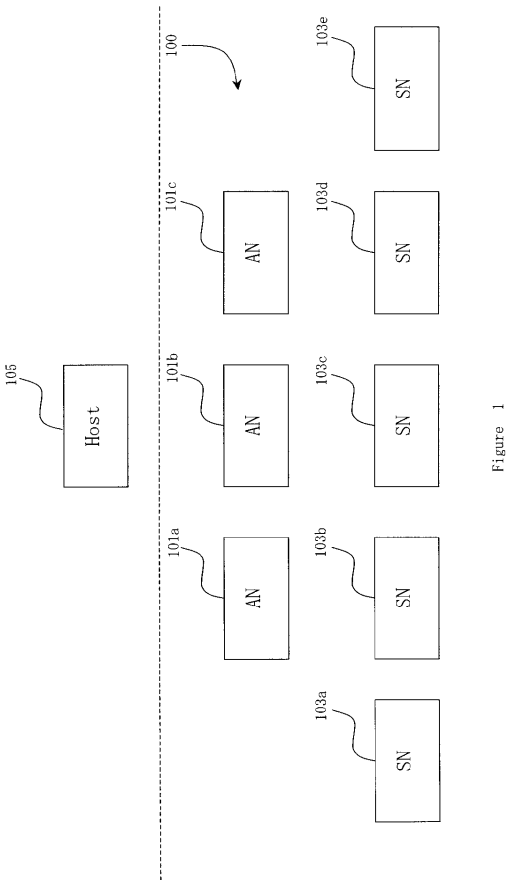



Figure 1

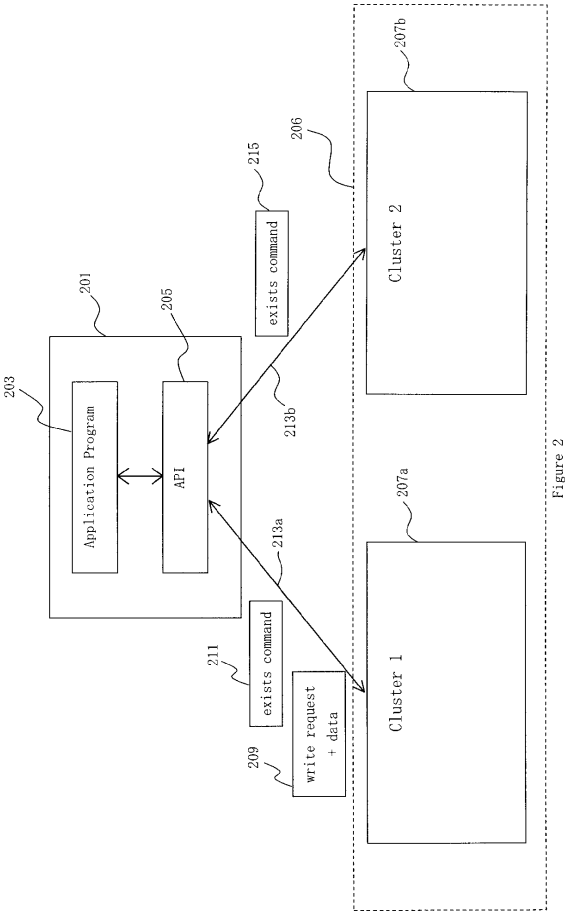



Figure 2

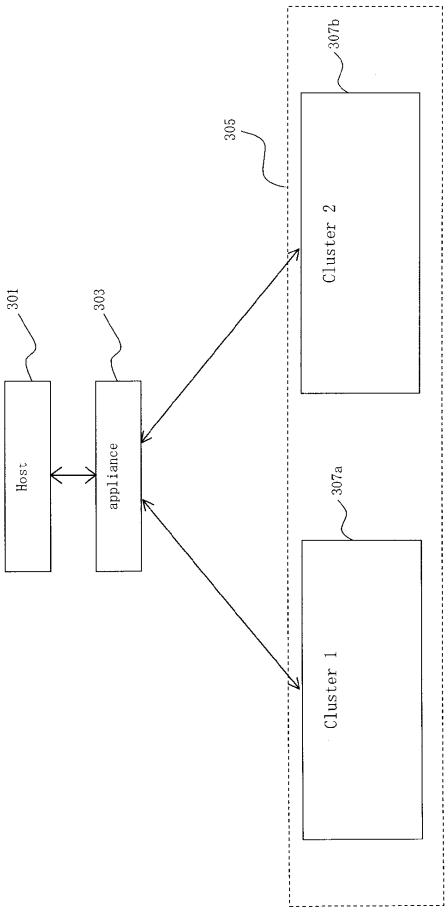



Figure 3

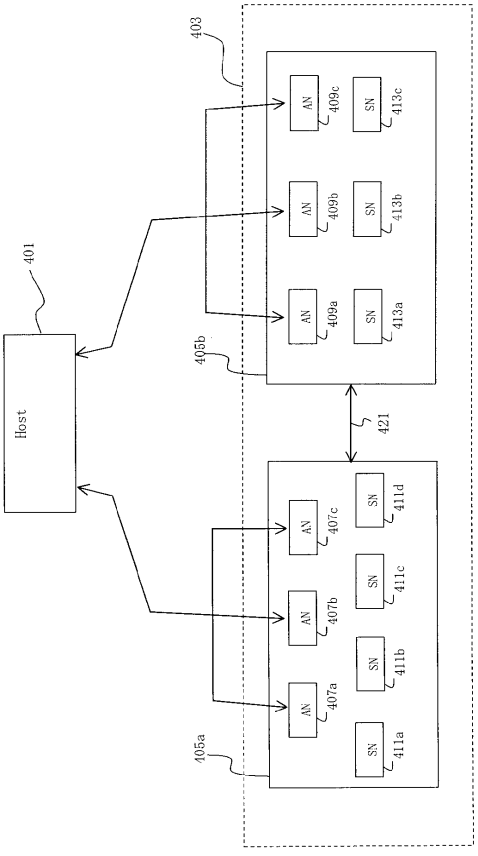



Figure 4

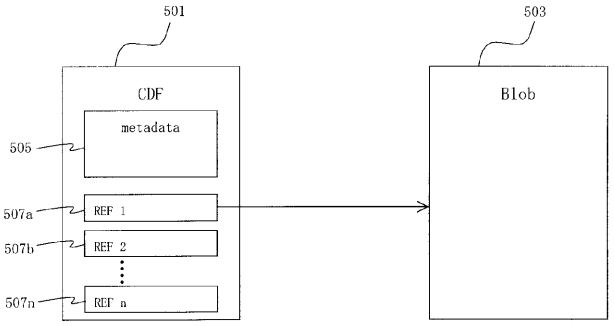



Figure 5

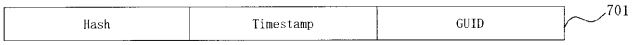



Figure 7

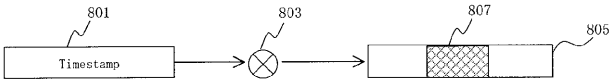



Figure 8

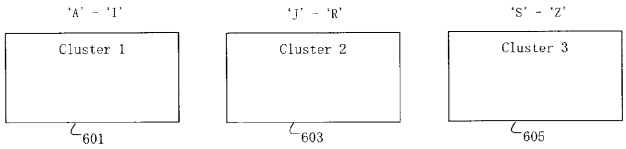



Figure 6

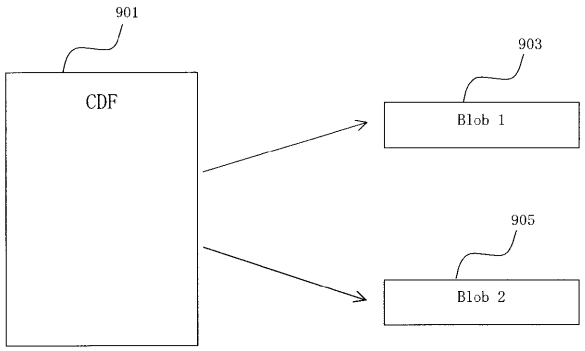



Figure 9

```
<FederationPolicy>
1001 <Timeslice start="-1" end="101992093021">
1003 <FBN mod="1" result="0" cluster="C1"/>
     </Timeslice>
1005 <Timeslice start="101992093021" end="101998383838">
1007 <FBN mod="2" result="0" cluster="C1"/>
1009 <FBN mod="2" result="1" cluster="C2"/> <!-- Added cluster C2 -->
     </Timeslice>
1011 <Timeslice start="101998383838 end="-1">
1013 <FBN mod="5" result="0" cluster="C1"/> <!-- C1 almost full -->
1015 <FBN mod="5" result="1" cluster="C2"/>
1017 <FBN mod="5" result="2" cluster="C2"/>
1019 <FBN mod="5" result="3" cluster="C3"/>
1021 <FBN mod="5" result="4" cluster="C3"/> <!-- Added cluster C3 -->
     </FederationPolicy>
```

Figure 10