(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第5918243号
(P5918243)
(45)発行日　平成28年5月18日(2016.5.18)　(24)登録日　平成28年4月15日(2016.4.15)

(51)Int.Cl.　F I
G06F 12/00 (2006.01)　G06F 12/00 531R
G06F 12/00 545A

請求項の数 27　(全 28 頁)

(21)出願番号　特願2013-530183(P2013-530183)
(86)(22)出願日　平成23年9月13日(2011.9.13)
(65)公表番号　特表2013-544386(P2013-544386A)
(43)公表日　平成25年12月12日(2013.12.12)
(86)国際出願番号　PCT/US2011/051313
(87)国際公開番号　WO2012/039988
(87)国際公開日　平成24年3月29日(2012.3.29)
審査請求日　平成26年9月11日(2014.9.11)
(31)優先権主張番号　12/889,744
(32)優先日　平成22年9月24日(2010.9.24)
(33)優先権主張国　米国(US)

(73)特許権者　510071448
ヒタチ　データ　システムズ　コーポレーション
ＨＩＴＡＣＨＩ　ＤＡＴＡ　ＳＹＳＴＥＭＳ　ＣＯＲＰＯＲＡＴＩＯＮ
アメリカ合衆国　カリフォルニア州　９５０５０　サンタ・クララ　ラファイエット　ストリート　２８４５
(74)代理人　110000279
特許業務法人ウィルフォート国際特許事務所
(72)発明者　ブライアント，アラン　ジー．
アメリカ合衆国　０２０３２　マサチューセッツ州　イーストウォルポール，ユニオン　ストリート，２２８

最終頁に続く

(54)【発明の名称】分散型データベースにおいてインテグリティを管理するためのシステム及び方法

(57)【特許請求の範囲】
【請求項１】
（Ａ）１以上のメタデータを含んだ単位であるリージョンの正式なコピーと前記リージョンのバックアップコピーであり保護レベルと同数のバックアップコピーとで構成され複数のノードに格納されている複数のリージョンコピーのうちの前記正式なコピー又は前記バックアップコピーの損失が検出された場合、前記リージョンの不完全なコピーを生成し、
（Ｂ）ソースコピーから、前記不完全なコピーに、メタデータをコピーし、
（Ｃ）前記不完全なコピーから、前記ソースコピーからコピーされたメタデータのうち、保留の更新でありターゲットコピーに無いメタデータを、前記ターゲットコピーにコピーし、
（Ｄ）（Ｂ）の完了前に、前記ターゲットコピーの内容が前記ソースコピーの内容と同じになった場合、前記ターゲットコピーを、前記リージョンのバックアップコピーとし、
（Ｅ）（Ｃ）の完了前に、前記不完全なコピーの内容が前記ソースコピーの内容と同じになった場合、前記不完全なコピーをバックアップコピーとする、
ことをコンピュータに実行させ、
前記損失が、いずれかのバックアップコピーの消失の場合、前記ソースコピーは、前記正式なコピーであり、前記ターゲットコピーは、前記消失の後に復元したバックアップコピーである、
ことを特徴とするコンピュータプログラム。

【請求項２】
前記消失の後に復元したバックアップコピーを、（Ｂ）の完了まで、部分的なコピーとし、
（Ｄ）において、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことをコンピュータに実行させることを特徴とする請求項１記載のコンピュータプログラム。
【請求項３】
前記損失が、前記正式なコピーのダウンの場合、（Ａ）の実行前に、前記１以上のバックアップコピーのうちのいずれかのバックアップコピーを、前記リージョンの正式なコピーに昇格する、
ことを更にコンピュータに実行させ、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、前記昇格された正式なコピーであり、前記ターゲットコピーは、前記ダウンした正式なコピーである、
ことを特徴とする請求項１又は２記載のコンピュータプログラム。
【請求項４】
前記損失が、前記正式なコピーのダウンの場合、前記ダウンした正式なコピーを部分的なコピーとする、
ことを更にコンピュータに実行させ、
前記損失が、前記正式なコピーのダウンの場合、前記ターゲットコピーは、前記部分的なコピーであり、
（Ｄ）において、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項３記載のコンピュータプログラム。
【請求項５】
前記複数のノードにわたる前記正式なコピー、前記バックアップコピー及び前記不完全なコピーの位置を特定するリージョンマップを提供する、
ことをコンピュータに実行させることを特徴とする請求項１乃至４のうちのいずれか１項に記載のコンピュータプログラム。
【請求項６】
（Ａ）１以上のメタデータを含んだ単位であるリージョンの正式なコピーと前記リージョンのバックアップコピーであり保護レベルと同数のバックアップコピーとで構成され複数のノードに格納されている複数のリージョンコピーのうちの前記正式なコピー又は前記バックアップコピーの損失が検出された場合、前記リージョンの不完全なコピーを生成し、
（Ｂ）ソースコピーから、前記不完全なコピーに、メタデータをコピーし、
（Ｃ）前記不完全なコピーから、前記ソースコピーからコピーされたメタデータのうち、保留の更新でありターゲットコピーに無いメタデータを、前記ターゲットコピーにコピーし、
（Ｄ）（Ｂ）の完了前に、前記ターゲットコピーの内容が前記ソースコピーの内容と同じになった場合、前記ターゲットコピーを、前記リージョンのバックアップコピーとし、
（Ｅ）（Ｃ）の完了前に、前記不完全なコピーの内容が前記ソースコピーの内容と同じになった場合、前記不完全なコピーをバックアップコピーとする、
ことをコンピュータに実行させ、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、いずれかのバックアップコピーから昇格された正式なコピーであり、前記ターゲットコピーは、前記ダウンした正式なコピーである、
ことを特徴とするコンピュータプログラム。
【請求項７】
前記損失が、前記正式なコピーのダウンの場合、（Ａ）の実行前に、前記１以上のバッ

クアップコピーのうちのいずれかのバックアップコピーを、前記リージョンの正式なコピーに昇格する、
ことを更にコンピュータに実行させ、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、前記昇格された正式なコピーである、
ことを特徴とする請求項６記載のコンピュータプログラム。
【請求項８】
前記損失が、前記正式なコピーのダウンの場合、前記ダウンした正式なコピーを部分的なコピーとする、
ことを更にコンピュータに実行させ、
前記損失が、前記正式なコピーのダウンの場合、前記ターゲットコピーは、前記部分的なコピーであり、
（Ｄ）において、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項７記載のコンピュータプログラム。
【請求項９】
前記複数のノードにわたる前記正式なコピー、前記バックアップコピー及び前記不完全なコピーの位置を特定するリージョンマップを提供する、
ことをコンピュータに実行させることを特徴とする請求項６乃至８のうちのいずれか１項に記載のコンピュータプログラム。
【請求項１０】
１以上のメタデータオブジェクトを含んだ単位であるリージョンの正式なコピーと前記リージョンのバックアップコピーであり保護レベルと同数のバックアップコピーとで構成された複数のリージョンコピーを格納する複数のノードを含んだシステムであって、
前記複数のリージョンコピーのうちの前記正式なコピー又は前記バックアップコピーの損失が検出された場合、前記リージョンの不完全なコピーを生成する生成手段と、
ソースコピーから、前記不完全なコピーに、メタデータをコピーする第１処理を実行する第１実行手段と、
前記不完全なコピーから、前記ソースコピーからコピーされたメタデータのうち、保留の更新であり前記ターゲットコピーに無いメタデータを、ターゲットコピーにコピーする第２処理を実行する第２実行手段と、
前記第１処理の完了前に、前記ターゲットコピーの内容が前記ソースコピーの内容と同じになった場合、前記ターゲットコピーを、前記リージョンのバックアップコピーとする第１変更手段と、
前記第２処理の完了前に、前記不完全なコピーの内容が前記ソースコピーの内容と同じになった場合、前記不完全なコピーをバックアップコピーとする第２変更手段と
を備え、
前記損失が、いずれかのバックアップコピーの消失の場合、前記ソースコピーは、前記正式なコピーであり、前記ターゲットコピーは、前記消失の後に復元したバックアップコピーである、
ことを特徴とするシステム。
【請求項１１】
前記消失の後に復元したバックアップコピーを、前記第１処理の完了まで、部分的なコピーとし、
前記第１変更手段は、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項１０記載のシステム。
【請求項１２】
前記損失が、前記正式なコピーのダウンの場合、前記不完全なコピーの生成前に、前記１以上のバックアップコピーのうちのいずれかのバックアップコピーを、前記リージョン

の正式なコピーに昇格する手段
を更に備え、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、前記昇格された正式なコピーであり、前記ターゲットコピーは、前記ダウンした正式なコピーである、
ことを特徴とする請求項１０又は１１記載のシステム。
【請求項１３】
前記損失が、前記正式なコピーのダウンの場合、前記ダウンした正式なコピーを部分的なコピーとする手段
を更に備え、
前記損失が、前記正式なコピーのダウンの場合、前記ターゲットコピーは、前記部分的なコピーであり、
前記第１変更手段は、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項１２記載のシステム。
【請求項１４】
前記正式なコピー、前記バックアップコピー及び前記不完全なコピーの位置を特定するリージョンマップが、前記複数のノードの各々に格納される、
ことを特徴とする請求項１０乃至１３のうちのいずれか１項に記載のシステム。
【請求項１５】
１以上のメタデータオブジェクトを含んだ単位であるリージョンの正式なコピーと前記リージョンのバックアップコピーであり保護レベルと同数のバックアップコピーとで構成された複数のリージョンコピーを格納する複数のノードを含んだシステムであって、
前記複数のリージョンコピーのうちの前記正式なコピー又は前記バックアップコピーの損失が検出された場合、前記リージョンの不完全なコピーを生成する生成手段と、
ソースコピーから、前記不完全なコピーに、メタデータをコピーする第１処理を実行する第１実行手段と、
前記不完全なコピーから、前記ソースコピーからコピーされたメタデータのうち、保留の更新であり前記ターゲットコピーに無いメタデータを、ターゲットコピーにコピーする第２処理を実行する第２実行手段と、
前記第１処理の完了前に、前記ターゲットコピーの内容が前記ソースコピーの内容と同じになった場合、前記ターゲットコピーを、前記リージョンのバックアップコピーとする第１変更手段と、
前記第２処理の完了前に、前記不完全なコピーの内容が前記ソースコピーの内容と同じになった場合、前記不完全なコピーをバックアップコピーとする第２変更手段と
を備え、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、いずれかのバックアップコピーから昇格された正式なコピーであり、前記ターゲットコピーは、前記ダウンした正式なコピーである、
ことを特徴とするシステム。
【請求項１６】
前記損失が、前記正式なコピーのダウンの場合、前記不完全なコピーの生成前に、前記１以上のバックアップコピーのうちのいずれかのバックアップコピーを、前記リージョンの正式なコピーに昇格する手段
を更に備え、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、前記昇格された正式なコピーである、
ことを特徴とする請求項１５記載のシステム。
【請求項１７】
前記損失が、前記正式なコピーのダウンの場合、前記ダウンした正式なコピーを部分的なコピーとする手段

を更に備え、
前記損失が、前記正式なコピーのダウンの場合、前記ターゲットコピーは、前記部分的なコピーであり、
前記第１変更手段は、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項１６記載のシステム。
【請求項１８】
前記正式なコピー、前記バックアップコピー及び前記不完全なコピーの位置を特定するリージョンマップが、前記複数のノードの各々に格納される、
ことを特徴とする請求項１５乃至１７のうちのいずれか１項に記載のシステム。
【請求項１９】
１以上のメタデータオブジェクトを含んだ単位であるリージョンの正式なコピーと前記リージョンのバックアップコピーであり保護レベルと同数のバックアップコピーとで構成された複数のリージョンコピーを格納する複数のノードを含んだシステムの制御方法であって、
（Ａ）複数のリージョンコピーのうちの前記正式なコピー又は前記バックアップコピーの損失が検出された場合、前記リージョンの不完全なコピーを生成し、
（Ｂ）ソースコピーから、前記不完全なコピーに、メタデータをコピーし、
（Ｃ）前記不完全なコピーから、前記ソースコピーからコピーされたメタデータのうち、保留の更新であり前記ターゲットコピーに無いメタデータを、ターゲットコピーにコピーし、
（Ｄ）（Ｂ）の完了前に、前記ターゲットコピーの内容が前記ソースコピーの内容と同じになった場合、前記ターゲットコピーを、前記リージョンのバックアップコピーとし、
（Ｅ）（Ｃ）の完了前に、前記不完全なコピーの内容が前記ソースコピーの内容と同じになった場合、前記不完全なコピーをバックアップコピーとし、
前記損失が、いずれかのバックアップコピーの消失の場合、前記ソースコピーは、前記正式なコピーであり、前記ターゲットコピーは、前記消失の後に復元したバックアップコピーである、
ことを特徴とする制御方法。
【請求項２０】
前記消失の後に復元したバックアップコピーを、（Ｂ）の完了まで、部分的なコピーとし、
（Ｄ）において、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項１９記載の制御方法。
【請求項２１】
前記損失が、前記正式なコピーのダウンの場合、（Ａ）の実行前に、前記１以上のバックアップコピーのうちのいずれかのバックアップコピーを、前記リージョンの正式なコピーに昇格する、
ことを更に実行し、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、前記昇格された正式なコピーであり、前記ターゲットコピーは、前記ダウンした正式なコピーである、
ことを特徴とする請求項１９又は２０記載の制御方法。
【請求項２２】
前記損失が、前記正式なコピーのダウンの場合、前記ダウンした正式なコピーを部分的なコピーとする、
ことを更に実行し、
前記損失が、前記正式なコピーのダウンの場合、前記ターゲットコピーは、前記部分的なコピーであり、
（Ｄ）において、前記部分的なコピーを、前記リージョンのバックアップコピーとし、

且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項２１記載の制御方法。
【請求項２３】
前記複数のノードにわたる前記正式なコピー、前記バックアップコピー及び前記不完全なコピーの位置を特定するリージョンマップを提供する、
ことを特徴とする請求項１９乃至２２のうちのいずれか１項に記載の制御方法。
【請求項２４】
１以上のメタデータオブジェクトを含んだ単位であるリージョンの正式なコピーと前記リージョンのバックアップコピーであり保護レベルと同数のバックアップコピーとで構成された複数のリージョンコピーを格納する複数のノードを含んだシステムの制御方法であって、
（Ａ）複数のリージョンコピーのうちの前記正式なコピー又は前記バックアップコピーの損失が検出された場合、前記リージョンの不完全なコピーを生成し、
（Ｂ）ソースコピーから、前記不完全なコピーに、メタデータをコピーし、
（Ｃ）前記不完全なコピーから、前記ソースコピーからコピーされたメタデータのうち、保留の更新であり前記ターゲットコピーに無いメタデータを、ターゲットコピーにコピーし、
（Ｄ）（Ｂ）の完了前に、前記ターゲットコピーの内容が前記ソースコピーの内容と同じになった場合、前記ターゲットコピーを、前記リージョンのバックアップコピーとし、
（Ｅ）（Ｃ）の完了前に、前記不完全なコピーの内容が前記ソースコピーの内容と同じになった場合、前記不完全なコピーをバックアップコピーとし、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、いずれかのバックアップコピーから昇格された正式なコピーであり、前記ターゲットコピーは、前記ダウンした正式なコピーである、
ことを特徴とする制御方法。
【請求項２５】
前記損失が、前記正式なコピーのダウンの場合、（Ａ）の実行前に、前記１以上のバックアップコピーのうちのいずれかのバックアップコピーを、前記リージョンの正式なコピーに昇格する、
ことを更に実行し、
前記損失が、前記正式なコピーのダウンの場合、前記ソースコピーは、前記昇格された正式なコピーである、
ことを特徴とする請求項２４記載の制御方法。
【請求項２６】
前記損失が、前記正式なコピーのダウンの場合、前記ダウンした正式なコピーを部分的なコピーとする、
ことを更に実行し、
前記損失が、前記正式なコピーのダウンの場合、前記ターゲットコピーは、前記部分的なコピーであり、
（Ｄ）において、前記部分的なコピーを、前記リージョンのバックアップコピーとし、且つ、前記不完全なコピーを削除する、
ことを特徴とする請求項２５記載の制御方法。
【請求項２７】
前記複数のノードにわたる前記正式なコピー、前記バックアップコピー及び前記不完全なコピーの位置を特定するリージョンマップを提供する、
ことを特徴とする請求項２４乃至２６のうちのいずれか１項に記載の制御方法。
【発明の詳細な説明】
【関連出願の相互参照】
【０００１】
この出願は、また、次の出願と関連している：

整理番号１２／８８９，７６２、「部分的なデータベース停電中に分散オブジェクトストレージシステムの有効性を増強するためのシステム及び方法」というタイトルで２０１０年９月２４日に出願された。

【背景技術】

【０００２】

本発明は、一般に、分散コンピューターネットワークにおける可用性、信頼性、及び持続性の高いデータストーレージのための技術に関する。

【関連技術の説明】

【０００３】

従来のテープ及び光学ストレージ解決法に取って代わるかまたはそれを補う、可用性、信頼性、及び持続性の高い方式による「固定コンテンツ」のアーカイブストレージ（archival storage）に対する必要性が高まってきた。用語「固定コンテンツ」とは、通常、参照または他の目的のためにそのままで保存されることが予想される任意のタイプのデジタル情報を指す。このような固定コンテンツの例としては、多くの例の中でもとりわけ、電子メール、文書、診断画像、チェック画像、音声録音、フィルム及びビデオなどが挙げられる。従来の独立ノード冗長アレイ（ＲＡＩＮ）ストレージ手法は、このような固定コンテンツの情報資産をストレージするための大規模なオンラインアーカイブを生み出すために選択されるアーキテクチャとして登場した。ＲＡＩＮアーキテクチャは、ノードが必要に応じてクラスタに結合すること及びクラスタから退出することを可能にすることにより、ストレージクラスタを１つ以上のノードの障害から隔離する。RAINタイプのアーカイブは、データを複数のノード上で複製することにより、ノードの障害または除去を自動的に補償することができる。通常、ＲＡＩＮシステムは、閉システム内の同一コンポーネントから設計されたハードウェア機器として広く提供される。

従来知られているアーカイブストレージシステムは、典型的に、ファイル毎にメタデータをそのコンテンツと同様に格納する。メタデータは、データを説明する、そのデータのコンポーネントである。メタデータは、典型的に、内容、質、条件、及びシステムに保存されている実際のデータのその他の特性について説明する。分散ストレージのコンテキストにおいて、ファイルに関するメタデータは、例えば、ファイルの名前、どこにファイルの断片が格納されているか、ファイルの作成日及び保持値を含む。信頼できるファイルストレージは、ファイルのストレージシステム信頼性及び有効性を達成するのに必要だが、メタデータのインテグリティもまたシステムの重要な部分である。しかし、 先行技術では、潜在的に信頼性の低いノードの分散システムにわたってメタデータを分散することが可能ではなかった。本発明は、当技術分野でこの要求に対処する。

【０００４】

改善されたアーカイバルストレージシステムは、一般に所有された米国特許第７，１５５，４６６号明細書、第７，６５７，５８１号明細書及び第７，６５７，５８６号明細書に記載されている。このシステムは、ノードの分散されたセットにわたって分散オブジェクトストアを提供する。米国特許第７，６５７，５８１号明細書によれば、対称なノードのアーカイバルストレージクラスタは、できればメタデータオブジェクトのかたちでメタデータへのアクセスを組織し提供する「メタデータ管理」システムを含む。メタデータオブジェクトは、それぞれユニークな名前を持っていて、メタデータオブジェクトは、リージョンへ組織される。１つの実施例において、リージョンは、１つ以上のオブジェクト属性（例えばオブジェクトの名前）のハッシュ値を計算し、生じるハッシュ値のビットの所与の数を抽出することにより選択される。ビットの数は、配置パラメータによってコントロールされてもよい。このスキームでは、リージョンはそれぞれ重複して格納され、リージョンは、リージョンコピーのセットを含む。具体的には、リージョンの１つの正式なコピー及び０以上のバックアップコピーがある。記述されているように、コピーの数は、多くのメタデータ保護レベル（「ＭＤＰＬ」）と呼ばれることがある配置パラメータによってコントロールされても良い。従って、例えばこのスキームの１つの実施例の中で、リージョンは正式なリージョンコピー及びそのＭＤＰＬ－１バックアップコピーを含む。リー

ジョンコピーは、１つのノード当たりの合計のリージョンコピーの数と同様に１つのノード当たりの正式なリージョンコピーの数が均等になるようクラスタのノードにわたって分散される。

【０００５】

上記メタデータマネージャシステムの別の一態様は、各リージョンの各コピーの原因であるノードを識別するリージョン「マップ」と呼ばれる。リージョンマップは、メタデータ管理システムを有するプロセスによってアクセス可能である。リージョンマップ中のリージョンは、ハッシュ値のセットを表し、すべてのリージョンのセットはあらゆるハッシュ値をカバーする。リージョンは、ハッシュ値のビットの数を抽出することにより引き出される数によって識別される。ネームスペース分割スキームは、リージョンマップ中でリージョンを定義し、かつ所与のリージョンのオーナー権をコントロールするために使用される。この分割スキームは、データベース中で実装される。スキームでは、リージョンコピーは、３つの段階：「正式」、「バックアップ」そして「不完全」、のうちの１つをもつ。リージョンコピーが正式の場合、リージョンへのすべての要求はこのコピーに行き、各リージョンにつき１つの正式なコピーがある。リージョンコピーがバックアップ（あるいは不完全）である場合、コピーは更新要求（正式なリージョンマネージャプロセスからの）を受信する。メタデータがロードされているが、コピーがまだ同期されない（典型的に正式なリージョンコピーに関して）場合、リージョンコピーは不完全である。同期が完了するまで、不完全なリージョンコピーは、別の段階への昇格の資格を有さない、すなわち、その段階ではコピーはバックアップコピーになる。

【０００６】

上記メタデータ管理スキームの別の態様は、バックアップリージョンコピーが正式なリージョンコピーと同期した状態で保たれることである。同期は、更新要求が処理されている場合に、プロトコルあるいは正式なリージョンコピーとそのＭＤＰＬ－１バックアップコピーの間の「コントラクト」を強化することにより保証される。例えば、更新を局所的に委ねた後、正式なリージョンマネージャプロセスは、そのＭＤＰＬ－１バックアップコピー（それらは典型的に、他のノードに置かれる）の各々への更新要求を出す。この通常の経過の中で、更新要求の受取において、所与のバックアップコピーに関連したリージョンマネージャプロセスは、確認を出すあるいは出すことを試みる。正式なリージョンマネージャプロセスは、更新が成功したという表示を提供する前にＭＤＰＬ－１バックアップコピーのすべてからの確認を待つ。この更新処理が失敗することがあるいくつかの状況がある、例えば、正式なリージョンマネージャ（確認を待っている間）は、バックアップマネージャプロセスがしたことを示す例外に遭遇するかもしれない、あるいは、たとえそれが確認を出したとしても、バックアップマネージャプロセスは、更新要求を局所的に処理することを失敗するかもしれない、あるいは、確認を出す間のバックアップリージョンマネージャプロセスは、正式なリージョンマネージャプロセスが故障したことを示す例外に遭遇するかもしれない、等など。バックアップリージョンマネージャが更新を処理することができなければ、それはサービスから退出する。バックアップリージョンマネージャプロセスあるいは正式なマネージャプロセスのいずれかが故障の場合、新しいリージョンマップが出される。このように同期を保証することによって、バックアップコピーはそれぞれ正式なコピー用の「ホットスタンバイ」である。そのようなバックアップコピーは、正式なリージョンコピーが失われる場合あるいは現在の正式なリージョンコピーが降格されるべきである（またいくつかのバックアップリージョンコピーが昇格されるべきである）と負荷分散要求が指示するために必要になるかもしれない正式なコピーになることへの昇格の資格を有する。

【０００７】

上記設計は、多くの同時のノード故障であってもそれがメタデータの高い有効性を保証するという点で有利である。ノード故障がある場合、１つ以上のリージョンが失われる、すなわち、その後、システムはそれをリカバリする必要がある。リカバリ過程は失われたリージョンのためのリージョンコピーを作成することを含む。その修理は、ハードウェア

とデータベースのリソースを消費し、従って、性能コストがある。大きなクラスタにおいては、修理が長時間かかる場合があり、その時間の間、クラスタはＭＤＰＬ未満かもしれない。

【簡易概要】

【０００８】

クラスタリカバリタイムは、ここに記載される増分リフレッシュ技術によって短縮される。その技術の目標は、以来データベースのその部分に生じる更新だけの増分リフレッシュを行なうことにより失われた（例えば障害中に）冗長分散データベースの部分を復旧させることである。

【０００９】

１つの実施例では、増分リフレッシュは、ネットワーク化された独立ノードの冗長アレイにおいて実装され、メタデータオブジェクトは、アレイにわたって分散されたリージョンのセットに格納される。リージョンマップは、リージョンのセットの位置を識別し、マップは、いくつかの段階：正式な（Ａ）、バックアップ（Ｂ）、そして不完全（Ｉ）、のうちの１つにあるリージョンのコピーと、どこに不完全なリージョンコピーがリージョンへの保留の更新を格納するかとを、典型的に含む。その技術によれば、その段階は、更に、保留の更新から追加される部分的な段階を含んでいて、クラスタに復帰するリージョンを修理するための復元の間、保留の更新が使用される。具体的には、その修理は、次のステップを含んでいる。最初に、リージョンの部分的な（Ｐ）コピーが、好ましくは（Ａ）または（Ｂ）リージョンが期限切れであることを決定する場合に、（Ｐ）リージョンへ（Ａ）または（Ｂ）リージョンコピーを降格すること（あるいは変換すること）により作成される。その後、部分的な（Ｐ）リージョンコピーは、適切な不完全な（Ｉ）リージョン上で既に待ち行列に入れられた保留の更新の適用により最新にされる。その後、（Ｐ）リージョンは、バックアップ（Ｂ）リージョンコピーに変換される。それが変換されるとすぐに、それに更新を送ることにより（Ｐ）リージョンが、バックアップリージョンとして扱われる。ある状況では、部分的な（Ｐ）コピーは、その結果、最後の更新を取消すためにそのリージョンの正式な（Ａ）コピーからのその部分を同期させることにより最新にされる。

【００１０】

代替実施例では、上記方法は、プロセッサと、プロセッサによって実行されたときに上記方法を実行するコンピュータプログラム指示を保持するコンピュータメモリとを有する装置によって実行される。

【００１１】

別の代替実施例では、上記方法は、それ自体がバッキングストアに置かれているかもしれないバージョンファイルシステムに関連する使用のためのコンピュータ読取り可能な媒体中のコンピュータプログラム製品によって実行される。そのコンピュータプログラム製品は、プロセッサによって実行されたとき上記方法を実行するコンピュータプログラム指示を保持する。

前述のものは、発明のより適切な特徴のいくつかを概説した。これらの特徴は単に例となるために解釈されるべきである。異なるやり方で記載された発明を適用すること、あるいは今後記載される発明の変更により、他の多くの有益な成果が成し遂げられるかもしれない。

【図面の簡単な説明】

【００１２】

本発明及びその利点のより完全な理解については、参考文献が、添付図面と併用される次の記載にある。

【図１】ここでの本件をその中で実施できる固定コンテンツストレージアーカイブの簡略化されたブロック図である。

【図２】各独立ノードが対称でありアーカイブクラスタアプリケーションをサポートする独立ノード冗長アレイの簡略化された説明図である。

【図３】所与のノード上で実行されるアーカイブクラスタアプリケーションの様々なコンポーネントの高レベル説明図である。
【図４】クラスタの所与のノード上のメタデータ管理システムのコンポーネントを示す図である。
【図５】例示的なリージョンマップを示す図である。
【図６】クラスタがサイズを増すとともにリージョンマップ変更を容易にするためにネームスペース分割スキームがどのように使用されるか示す図である。
【図７】（Ｂ）リージョンの損失の次に行われるリストア中に、どのように（Ｐ）リージョンコピー（Ｐリージョン）が作成され、そして、ここで記載される技術によって（Ｂ）リージョンへ昇格されるかの例を示す図である。
【図８】（Ａ）リージョンの損失の次に行われるリストア中に、どのように（Ｐ）リージョンコピー（Ｐリージョン）が作成され、そして、ここで記載される技術によって（Ｂ）リージョンへ昇格されるかの例を示す図である。
【例示的な実施例の詳細な記載】
【００１３】
以下に記載する技術は、スケーラブルなディスクベースのアーカイブストレージ管理システム、好ましくは、独立ノード冗長アレイに基づくシステムアーキテクチャの中で好ましくは実施される。各ノードは、異なるハードウェアを備える場合があり、したがって「異種である」と考えることができる。ノードは通常、１つまたは複数のストレージディスクへのアクセスを有するが、これらのストレージディスクは、実際の物理ストレージディスクとすることもでき、またはストレージエリアネットワーク（ＳＡＮ）におけるように仮想ストレージディスクとすることもできる。各ノード上でサポートされるアーカイブクラスタアプリケーション（及び任意選択で、このアプリケーションが実行される、基礎を成すオペレーティングシステム）は、同じかまたはほぼ同じとすることができる。例示的な一実施形態では、各ノード上のソフトウェアスタック（オペレーティングシステムを含むことができる）は対称だが、ハードウェアは異種である場合がある。このシステムを使用して、図１に示すように、企業が、とりわけ文書、電子メール、衛星画像、診断画像、チェック画像、音声録音、ビデオなど、多くの異なるタイプの固定コンテンツ情報のための永続的ストレージを生み出すことができる。当然、これらのタイプは例示に過ぎない。独立サーバまたはいわゆるストレージノード上でデータを複製することによって、高レベルの信頼性が達成される。各ノードは、そのピアと対称であることが好ましい。したがって、好ましくはどんな所与のノードも全ての機能を実施することができるので、いずれか１つのノードの障害はアーカイブの可用性にほとんど影響を及ぼさない。
【００１４】
一般に所有される米国特許第７，１５５，４６６号明細書、第７，６５７，５８１号明細書、及び第７，６５７，５８６号明細書で述べられているように、各ノード上で実行される分散ソフトウェアアプリケーションが、デジタル資産を取り込み、保存し、管理し、取り出す。図２の例示する一実施形態では、個別アーカイブの物理的境界が、クラスタと呼ばれる。通常、クラスタは、単一のデバイスではなく、デバイスの集合である。デバイスは、同種または異種である場合がある。典型的なデバイスは、Linuxなどのオペレーティングシステムを実行するコンピュータまたはマシンである。コモディティハードウェア上でホストされるLinuxベースのシステムのクラスタは、少数のストレージノードサーバから、何千テラバイトものデータを記憶する多くのノードまでスケーリングできるアーカイブを提供する。このアーキテクチャにより、記憶容量が、組織のますます増加するアーカイブ要件に常に遅れを取らずにいられることが確実になる。アーカイブが常にデバイス障害から保護されるように、データがクラスタ全体で複製されることが好ましい。ディスクまたはノードに障害が発生した場合、クラスタは自動的に、クラスタ中の、同じデータのレプリカを維持する他のノードにフェイルオーバする。
【００１５】
例示的なクラスタは、以下の一般的なコンポーネントカテゴリ、すなわちノード２０２

、ネットワークスイッチ２０４のペア、電力分散ユニット（ＰＤＵ）２０６、及び無停電電源（ＵＰＳ）２０８を好ましくは有する。ノード２０２は、典型的に１つ以上のコモディティサーバを有し、ＣＰＵ（例えばインテルｘ８６、適切なランダムアクセスメモリ（ＲＡＭ）、１つ以上のハードドライブ（例えば標準のＩＤＥ／ＳＡＴＡ、ＳＣＳＩなど）及び２枚以上のネットワークインターフェイス（ＮＩＣ）カード）を有する。典型的なノードは、２.４ＧＨｚのチップ、５１２ＭＢ　ＲＡＭ及び６つの（６）２００ＧＢハードドライブを備えた２Ｕラックマウントユニットである。しかし、これに限られない。ネットワークスイッチ２０４は、典型的にノード間のピアツーピア通信を可能にする内部スイッチ２０５、及び、各ノードへの追加のクラスタアクセスを許可する外部スイッチ２０７を有する。各スイッチは、クラスタ中の全て潜在的なノードを扱うことを十分なポートに要求する。イーサネットまたはGigEのスイッチは、この目的に使用されてもよい。ＰＤＵ２０６は、全てのノード及びスイッチに動力を提供するために使用され、ＵＰＳ２０８は、全てのノード及びスイッチを保護するために使用される。制限しているつもりではないが、典型的にクラスタは、公衆インターネット、企業イントラネットあるいは他の広域か、ローカルエリアネットワークのようなネットワークに連結可能である。例となる実施例において、クラスタは、企業環境内で実施される。それは、例えばサイトの企業ドメイン名前システム（ＤＮＳ）ネームサーバによってナビゲートすることにより達するかもしれない。従って、例えば、クラスタのドメインは、既存のドメインの新しいサブドメインかもしれない。代表的な実施において、サブドメインは、企業のＤＮＳサーバの中でクラスタ自体の中のネームサーバに委託される。エンドユーザは、任意の従来のインターフェースあるいはアクセスツールを使用して、クラスタにアクセスする。従って、例えば、クラスタへのアクセスは、任意のＩＰベースのプロトコル（ＨＴＴＰ、ＦＴＰ、ＮＦＳ、ＡＦＳ、ＳＭＢ、ウェブサービスなど）上に、ＡＰＩを経由して、あるいは他の既知か、その後発展したアクセス方式、サービス、プログラムあるいはツールを通じて実行されるかもしれない。

【００１６】

クライアントアプリケーションは、標準ＵＮＩＸファイルプロトコルのような１つ以上のタイプの外部ゲートウエイによるクラスタ、またはＨＴＴＰ　ＡＰＩにアクセスする。アーカイブは、好ましくは、オプションで任意の標準ＵＮＩＸファイルプロトコル系設備の下に位置できる仮想ファイルシステムを通じて露出される。これらは、ＮＦＳ、ＦＴＰ、ＳＭＢ／ＣＩＦＳなどを含む。

【００１７】

１つの実施例において、アーカイブクラスタアプリケーションは、クラスタとして（例えば、イーサネット経由で）ネットワーク化される独立ノード（Ｈ－ＲＡＩＮ）の冗長アレイ上で作動する。所与のノードのハードウェアは異種かもしれない。最大の信頼性のために、しかし、好ましくは、各ノードは、今、図３に示されるようないくつかのランタイムコンポーネントから成る分散アプリケーション（すなわち、同じインスタンス、あるいは本質的に同じインスタンスかもしれない）のインスタンス３００を実行する。従って、ハードウェアは、異種かもしれないが、ノード（少なくともそれが本発明に関係のある）上のソフトウェアスタックは、同じである。これらのソフトウェアコンポーネントは、ゲートウエイプロトコルレイヤ３０２、アクセスレイヤ３０４、ファイルトランザクション及び管理レイヤ３０６、及び、コアコンポーネントレイヤ３０８を有する。機能が他の意味のある方法で特徴づけられるかもしれないことを通常のスキルのうちの１つが認識するので、「レイヤ」指定は、説明目的に提供される。レイヤ（あるいはその点でコンポーネント）の１つ以上は、統合されるかそうでないかもしれない。いくつかのコンポーネントは、レイヤに渡って共有されるかもしれない。

【００１８】

ゲートウエイプロトコルレイヤ３０２の中のゲートウエイプロトコルは、既存のアプリケーションに透明性を提供する。具体的には、ゲートウエイは、カスタムアプリケーションを構築するためのウェブサービスＡＰＩ同様に、ＮＦＳ３１０及びＳＭＢ／ＣＩＦＳ３

１２のようなネイティヴファイルサービスを提供する。ＨＴＴＰサポート３１４も提供される。アクセスレイヤ３０４は、アーカイブへのアクセスを提供する。具体的には、発明によれば、固定コンテンツファイルシステム（ＦＣＦＳ）３１６は、アーカイブオブジェクトへフルアクセスを提供するためにネイティヴファイルシステムをエミュレートする。あたかもそれらが通常のファイルかのように、ＦＣＦＳは、アーカイブコンテンツにアプリケーションダイレクトアクセスを与える。好ましくは、メタデータがファイルとして露出されている一方、アーカイブコンテンツは、そのオリジナルフォーマットでレンダリングされる。ＦＣＦＳ３１６は、管理者らが、使いやすい方法で固定コンテンツのデータをセットアップできるように、ディレクトリと許可についての従来のビュー及びルーチンファイルレベルコールを提供する。ファイルアクセス呼び出しは、好ましくは、ユーザスペースデーモンによって傍受され、ダイナミックに呼び出しのアプリケーションへの適切な表示を作成する適切なコアコンポーネント（レイヤ３０８に）に送られる。ＦＣＦＳ呼び出しは、好ましくは、自律的なアーカイブ管理を促進するアーカイブポリシによって抑制される。従って、一例において、管理者かアプリケーションは、保存期間（所与のポリシ）がまだ有効のアーカイブオブジェクトを削除することができない。

【００１９】

アクセスレイヤ３０４は、好ましくは、また、ウェブユーザインターフェース（ＵＩ）３１８及びＳＮＭＰゲートウエイ３２０を含む。ウェブユーザインターフェース３１８は、ファイルトランザクション及び管理レイヤ３０６での管理エンジン３２２への対話型のアクセスを提供する管理者コンソールとして好ましくは実施される。管理上のコンソール３１８は、好ましくは、アーカイブオブジェクト及び個々のノードを含むアーカイブの動的考察を提供するパスワードで保護されウェブベースのＧＵＩである。ＳＮＭＰゲートウエイ３２０は、安全にストレージ管理アプリケーションがクラスターアクティビティを監視し制御することを可能にしながらストレージ管理アプリケーションに管理エンジン３２２への容易なアクセスを提供する。管理エンジンモニタは、システムとポリシイベントを含むアクティビティをクラスタする。ファイルトランザクションと管理レイヤ３０６は、また、要求マネージャプロセス３２４を含む。要求マネージャ３２４は、コアコンポーネントレイヤ３０８の中のポリシマネージャ３２６からの内部要求と同様に外界（アクセスレイヤ３０４を通じて）からの全ての要求を統合する。

【００２０】

コアコンポーネントは、ポリシマネージャ３２６に加えて、メタデータマネージャ３２８及びストレージマネージャ３３０の１つ以上のインスタンスを含む。メタデータマネージャ３２８は、各ノードに好ましくはインストールされる。クラスタ中のメタデータマネージャは、集合的に、全てのアーカイブオブジェクトを管理しながら、分散型データベースとして作動する。所与のノードにおいて、メタデータマネージャ３２８は、好ましくは、アーカイブオブジェクトのサブセットを管理し、各オブジェクトが、好ましくは、外部ファイル（「ＥＦ」（ストレージ用にアーカイブに入ったデータ））とアーカイブデータが物理的に検索される内部ファイル（各々「ＩＦ」）のセット間をマップする。同じメタデータマネージャ３２８は、また、他のノードから複製されたアーカイブオブジェクトのセットを管理する。従って、全ての外部ファイルの現状は、いくつかのノード上の複数メタデータマネージャに常に利用可能である。ノード障害の場合には、他のノード上のメタデータマネージャが、機能不全のノードによって以前管理されたデータへのアクセスを提供し続ける。このオペレーションは、以下により詳細に説明される。ストレージマネージャ３３０は、分散アプリケーション中の他の全てのコンポーネントに利用可能なファイルシステムレイヤを提供する。好ましくは、それはノードのローカルファイルシステムにデータオブジェクトを格納する。所与のノード中の各ドライブは、好ましくは、それぞれ自身のストレージマネージャを持っている。これは、ノードが個別のドライブを削除し、処理能力を最適化することを可能にする。ストレージマネージャ３３０は、また、システム情報、データの一貫性チェック及び直接ローカル構造をトラバースする能力を提供する。

【００２１】

さらに図３で示されるように、クラスタは、通信ミドルウェアレイヤ３３２及びＤＮＳマネージャ３３４を通じて内部及び外部通信を管理する。インフラストラクチャ３３２は、アーカイブコンポーネント中の通信を可能にする効率的で信頼できるメッセージベースのミドルウェアレイヤである。図で示した実施例において、レイヤは、マルチキャストとポイントツーポイント通信をサポートする。ＤＮＳマネージャ３３４は、企業サーバに全てのノードを接続する分散型ネームサービスを行う。好ましくは、ＤＮＳマネージャ（単独であるいはＤＮＳサービスと共に）ロードバランスは、最大のクラスタ処理能力及び有効性を保証することを全てのノードに渡って要求する。

【００２２】

図で示した実施例において、ＡｒＣアプリケーションインスタンスは、Red Hat Linux 9.0、Fedora Core 6などのような基礎オペレーティングシステム３３６上で実行する。通信ミドルウェアは、任意の便利な分散型通信メカニズムである。他のコンポーネントは、固定コンテンツファイルシステム（ＦＣＦＳ）３１６のために使用されてもよいＦＵＳＥ（USErspaceの中のファイルシステム）を含むかもしれない。ＮＦＳゲートウエイ３１０は、標準のnfsd LinuxカーネルＮＦＳドライバによって実施されるかもしれない。各ノード中のデータベースは、実施されるかもしれない、例えば、オブジェクト関係データベース管理システム（ＯＲＤＢＭＳ）であるPostgreSQL（またここにPostgresとして引用される）である。ノードは、Java HTTPサーバ及びservletコンテナーであるジェティのようなウェブサーバを含むかもしれない。もちろん、上記のメカニズムは、単に例となる。

【００２３】

所与のノード上のストレージマネージャ３３０は、物理的な記憶デバイスを管理する責任がある。好ましくは、各ストレージマネージャインスタンスは、全てのファイルがその配置アルゴリズムによって入れられる単一のルートディレクトリの責任がある。複数のストレージマネージャインスタンスは、ノード上で同時に作動することができ、各々は、通常、システムで異なる物理的なディスクを表わす。ストレージマネージャは、システムの残りから使用されているドライブ及びインターフェース技術を抽象する。ストレージマネージャインスタンスがファイルを書くように依頼される場合、それはそのために責任を負う表現用のフルパス及びファイル名を生成する。代表的な実施例において、ストレージマネージャ上に格納される各オブジェクトは、それが異なるタイプの情報を追跡するデータを格納する場合、ファイルにそれ自身のメタデータを加えて、そのときストレージマネージャと共に保存されるローデータとして受信される。例として、このメタデータは、限定無しで含む：ＥＦ長さ（バイトでの外部ファイルの長さ）、ＩＦセグメントサイズ（内部ファイルのこの部分のサイズ）、ＥＦ保護表現（ＥＦ保護モード）、ＩＦ保護役割（この内部ファイルの表現）、ＥＦ生成タイムスタンプ（外部ファイルタイムスタンプ）、シグネチャ（シグネチャタイプを含む書き込み（ＰＵＴ）の時間の内部ファイルのシグネチャ）及びＥＦファイル名（外部ファイルファイル名）。内部ファイルデータでこの追加のメタデータを格納することは、追加のレベルの保護を提供することである。具体的には、スカビンジングは、内部ファイルに保存されたメタデータからデータベースに外部ファイルレコードを作成することができる。他のポリシは、内部ファイルが元の状態のままになることを有効にするために内部ファイルに対する内部ファイルハッシュを有効にすることができる。

【００２４】

前述のように、内部ファイルは、アーカイブオブジェクト中でオリジナルの「ファイル」の一部を表わす、データの「チャンク」かもしれない、また、それらはストライピングと保護ブロックを達成するために異なるノードに置かれるかもしれない。典型的に、１つの外部ファイルエントリは各アーカイブオブジェクトのためのメタデータマネージャの中にあり、その一方で、個々の外部ファイルエントリのための多くの内部ファイルエントリがあるかもしれない。典型的に、内部ファイルレイアウトは、システムに依存する。所与の実施において、ディスク上のこのデータの実際の物理フォーマットは一連の可変長レコードに格納される。

【００２５】
要求マネージャ３２４は、システム内の他のコンポーネントとのやりとりによりアーカイブアクションを行なうために必要とされるオペレーションのセットを実行する責任がある。要求マネージャは、異なるタイプの多くの同時のアクションをサポートし、機能不全のトランザクションをロールバックすることができ、実行するのに長い時間かかるトランザクションをサポートする。要求マネージャは、更に、アーカイブの読取り書き込みオペレーションが適切に扱われる事を保証し、全ての要求がいつでも既知の状態である事を保証する。更に、それは、所与のクライアント要求を満たすためにノードに渡って複数の読取り書き込みオペレーションを調整するためにトランザクション制御を提供する。更に、要求マネージャは、最近使われたファイルのためのメタデータマネージャエントリをキャッシュに格納し、データブロックと同様にセッションのためのバッファリングを提供する。

【００２６】
クラスタの主要な責任は、ディスク上に無制限のファイルを確実に格納することである。それが何らかの理由で手が届かないか、そうでなければ利用不可能かもしれないという意味で、所与のノードは「信頼性が低い」と見なされるかもしれない。そのような潜在的に信頼性の低いノードのコレクションは、確実で、高度に利用可能なストレージを作成することに協力する。一般に、格納される必要のある２つのタイプの情報がある：ファイル自体及びファイルに関するメタデータ。

【００２７】
メタデータ管理

【００２８】
米国特許第７，６５７，５８１号明細書（その開示は参考文献によってここに組込まれる）に記載のように、メタデータマネジメントシステムは、システムメタデータのような所与のメタデータへのアクセスを組織し提供する責任がある。このシステムメタデータは、構成情報、管理ＵＩに表示された情報、メトリクス、復元不能なポリシ違反についての情報などに表示された情報等と同様にアーカイブに置かれたファイルについての情報を含む。詳細に示されていないが、他のタイプのメタデータ（例えばアーカイブしたファイルに関連したユーザメタデータ）も今説明されるメタデータ管理システムを使用して管理されるかもしれない。

【００２９】
代表的な実施例では、メタデータ管理システムは、次のオブジェクトタイプ（それらは単に例となる）の１つ以上を含んでいるかもしれないメタデータオブジェクトのセットのための持続性を提供する。

・ExternalFile：アーカイブのユーザによって知覚されるようなファイル、
・InternalFile：ストレージマネージャによって格納されたファイル。典型的には、外部ファイルと内部ファイルの間に一対多数の関係があるかもしれない、
・ConfigObject：クラスタを構成するのに使われる名前／値ペア、
・AdminLogEntry：管理者ＵＩに表示されるメッセージ、
・MetricsObject：ある時点でのアーカイブ（例えばファイルの数）のある測定を表わす、タイムスタンプされたキー／値ペア、そして
・PolicyState：あるポリシの違反。

【００３０】
各メタデータオブジェクトは、好ましくは、変わらないユニークな名前を持っているかもしれない。メタデータオブジェクトは、リージョンに組織される。リージョンは、正式なリージョンコピーとメタデータ保護レベル（ＭＰＤＬ）数（０以上のセット）バックアップリージョンコピーを有する。０のコピーで、メタデータマネジメントシステムは、計量可能であるが、高度に利用可能ではないかもしれない。リージョンは、１つ以上のオブジェクト属性（例えばフルパス名やその一部のようなオブジェクトの名前）をハッシュ及びハッシュ値のビットの所与数の抽出により選択される。これらのビットは、リージョン

番号から成る。選択されたビットは、低位ビット、高位ビット、中位ビットあるいは個々のビットの任意のコンビネーションかもしれない。代表的な実施例において、所与のビットはハッシュ値の低位ビットである。オブジェクトの属性か属性（複数）は、任意の便利なハッシュ関数を使用してハッシュされるかもしれない。これらは制限なしで、java.lang.string.hashCode等のようなＪａｖａベースのハッシュ関数を含む。好ましくは、リージョン番号から成るビットの数は、ここでregionMapLevelと呼ばれ、構成パラメータによってコントロールされる。この構成パラメータが６にセットされる場合、例えば、これは$2^6=64$リージョンが得られる。もちろん、以下に説明されるように、多くのリージョンは許され、リージョンの数はネームスペース分割スキームを使用して自動的に調節されるかもしれない。

【００３１】

米国特許第７，６５７，５８１号明細書に記載通り、各リージョンは、重複して格納されるかもしれない。上記の通り、リージョンの１つの正式なコピー及び０以上のバックアップコピーがある。前述のように、バックアップコピーの数は、メタデータ・データ保護レベル（あるいは「ＭＤＰＬ」）構成パラメータによってコントロールされる。好ましくは、リージョンコピーは、１つのノード当たりの正式なリージョンコピーの数の平衡を保ち、かつ１つのノード当たりの合計のリージョンコピーの数の平衡を保つようにクラスタの全てのノードに渡って分散される。

【００３２】

メタデータ管理システムは、各ノード上で作動するデータベースにメタデータオブジェクトを格納する。このデータベースは、リージョンマップをサポートするために使用される。典型的なデータベースは、オープンソースとして利用可能であるPostgreSQLを使用して実施される。好ましくは、各リージョンコピーのスキーマがあり、各スキーマでは、各タイプのメタデータオブジェクト用のテーブルがある。スキーマは、単にテーブル、インデックス、手順及び他のデータベースオブジェクトを所有することができるネームスペースである。各リージョンは、好ましくは、それ自身のスキーマを持っている。各スキーマは、テーブル一式、すなわち各メタデータオブジェクトに１つ持っている。これらのテーブルのうちの１つの列は、単一のメタデータオブジェクトに相当する。Postgresが好ましいデータベースであると同時に、任意の便利なリレーショナルデータベース（例えばオラクル、IBM DB/2など）が使用されてもよい。

【００３３】

図４で示されるように、各ノード４００は、プロセスあるいはコンポーネント、すなわち、１つ以上のリージョンマネージャ（ＲＧＭ）４０２ａ－ｎ、メタデータマネージャ（ＭＭ）４０４、少なくとも１つのメタデータマネージャクライアント（ＭＭＣ）４０６、及び１つ以上のスキーマ４１０ａ－ｎがある１つのデータベース４０８、のセットを有する。ＲＧＭ（ｓ）、ＭＭ及びＭＭＣコンポーネントは、Ｊａｖａ仮想マシンのようなバーチャルマシン４１２で実行する。各リージョンコピーにつき１つのＲＧＭがある。従って、正式なリージョンコピー用のＲＧＭ、各バックアップリージョンコピー用のRGM及びそれぞれ不完全なリージョンコピー用のＲＧＭがある。ＲＧＭ４０２のスキーマを管理する各ＲＧＭ４０２用のデータベーススキーマ４１０もある。データベースは、また、リージョンマップ４０５を格納する。上述の特許の開示によれば、各ノードは、好ましくは、同期スキームによって強化されている要求と共に、リージョンマップの同じ全体的な見解を持っている。リージョンマネージャＲＧＭ４０２は、リージョンコピー（それが正式な、バックアップ、あるいは不完全な場合によっては）上で作動し、メタデータマネージャクライアント４０６、及び他のリージョンマネージャ４０２によって提出された要求の実行に責任がある。要求は、図３で示された通信ミドルウェアあるいは他のメッセージングレイヤのような任意の便利な手段を通じて所与のRGMに提供される。リージョンマネージャは、これらの要求が実行する実行環境を提供する、例えば、スキーマのＲＧＭによって管理されているスキーマ上で作動するように構成されているデータベースへの接続を提供することによって。各リージョンマネージャは、データベース４０８にそのデータを格納す

る。メタデータマネージャ４０４は、ノード上のメタデータ管理の責任があるトップレベルのコンポーネントである。それは、リージョンマネージャ（ＲＧＭ）を作成し破壊し、そして、ＲＧＭ、例えばクラスタ構成情報、データベース接続のプールによって必要とされるリソースを組織する責任がある。好ましくは、所与のメタデータマネージャ（所与のノード中の）は、リーダーとして働き、どのメタデータマネージャ（ノードのセット又はサブセットに渡った）がどのリージョンコピーに責任を負うかを決める責任がある。賛成アルゴリズム又はその変形のようなリーダー選挙アルゴリズムは、メタデータマネージャリーダーを選ぶために使用されるかもしれない。好ましくは、１つのノード当たり複数のＭＭを実行することは可能であるが、各ノードは、１つのメタデータマネージャを持っている。一旦リージョンオーナー権がネームスペース分割スキーム（下記に述べられるように）によって確立されたならば、各メタデータマネージャは、１つ以上のリージョンマネージャのそのセットに従って調節することに責任がある。システムコンポーネント（例えば管理エンジン、ポリシマネージャなど）は、メタデータマネージャクライアントを通じてメタデータマネージャＭＭとやりとりをする。ＭＭＣは、所与の要求を実行するためにＲＧＭを見つける事、選択されたＲＧＭに要求を出す事、及び選択されたＲＧＭが利用不可能な場合に（例えば、ノードが機能しなくなったので）要求を再試行することに責任がある。後者の場合は、新しいリージョンマップがノードで受信される場合、再試行要求が成功するであろう。

【００３４】

上記の通り、リージョンマップは、各リージョンの各コピーに責任のあるノードを識別する。バーチャルマシン４１２（またその中での各ＲＧＭ、ＭＭ、及びＭＭＣ構成要素）は、リージョンマップ４０５へのアクセスを持っている；リージョンマップのコピー４２０も、それがＪＶＭにコピーされた後、図４に示される。リージョンマップは、従って、所与のノード中のＪＶＭ及びデータベースの両方に利用可能である。このインスタンスとなる実施例において、各メタデータオブジェクトは、0x0と0x3fffffff合計間の整数を産出するためにハッシュされる、つまり３０ビットの値の属性（例えば名前）を持っている。これらの値は、オーバーフロー問題（例えば範囲の高域に１を加える時）にぶち当たる事なく、符号付き３２ビット整数中で快適に表わす事ができる。３０ビットは、大きなクラスタにさえ十分であるおよそ１０億までのリージョンを考慮に入れる。リージョンは、１セットのハッシュ値を表わし、全てのリージョンのセットは、あらゆるハッシュ値をカバーする。各リージョンのための異なるビット位置があり、異なるビット位置は、好ましくは固定順になっている。従って、各リージョンは、ハッシュ値のRegionLevelMapビットの抽出により好ましくは引き出される数によって識別される。６４リージョンを考慮に入れて、構成パラメータが６にセットされる場合、生じるハッシュ値は、0x0から0x3fの数である。

先述の通り、リージョンコピーは、３つの（３）段階、すなわち、「正式な」（Ａ）、「バックアップ」（Ｂ）そして「不完全」（Ｉ）のうちの１つにある。リージョンコピーが正式の場合、リージョンへの全ての要求がこのコピーに行き、また、各リージョンにつき１つの正式なコピーがある。リージョンコピーがバックアップである場合、コピーは、バックアップ要求（正式なリージョンマネージャプロセスからの）を受信する。メタデータがロードされているが、コピーがまだ同期されない（典型的に他のバックアップコピーに関して）場合、リージョンコピーは、不完全である。同期が完了するまで、不完全なリージョンコピーは、別の段階への昇進の資格を有さない、すなわち、そのポイントではコピーは、バックアップコピーになる。各リージョンは、１つの正式なコピー、所与の数（ＭＤＰＬ構成パラメータによってセットされた）バックアップあるいは不完全なコピーを持っている。

【００３５】

米国特許第７，６５７，５８１号明細書に記載通り、バックアップリージョンコピーは、正式なリージョンコピーとそのＭＤＰＬバックアップコピー間で所与のプロトコル（あるいは「契約」）を強化することにより、正式なリージョンコピーと同期され続ける。こ

のプロトコルは、今説明される。

【００３６】

説明されたように、リージョンマップは、各リージョンの各コピーのオーナー権について説明する。例えば、図５は、４つのノードクラスタ用のリージョンマップをmetadataMDPL=2で例証する。この例において、示されるように、ノード１はリージョン０は正式であり、ノード２及び３はバックアップとして指定され、ノード２はリージョン１は正式であり、ノード３及び４はバックアップとして指定されるなどである。ネームスペース分割スキームは、クラスタが増大するとともに、特定のリージョンのコントロール（オーナー権）を変更するために使用されても良い。動的成長を許可する１つの方法は、ハッシュ値番号を有するットの数を決定するregionMapLevel構成パラメータをインクリメントすることである。クラスタが増大するとともに、リージョンマップの１つ以上のパーティションが「分離した」工程を経る。分離は、ハッシュ値のもう１つのビットを使用し、その結果メタデータを再分散することを引き起こす。例えば、レベル６のマップ、及びハッシュ値0x1000002a及び0x1000006aを備えた２つのメタデータオブジェクトを考慮してほしい。これらのハッシュ値（２進法の「0010」である「2」、及び２進法の「0110」である「6」を備えた１６進法0x2a）の最後の６ビットは、同じである：したがって、両方のオブジェクトは、リージョン0x2aに分類される。その後、マップレベルが７に増加される場合、リージョンは0から0x7fであり、それにより、異なるリージョン、すなわち0x2a、0x6aに入ることを２つのオブジェクトに強いる。

【００３７】

このアプローチは使用されてもよいが、それは同時に分離していることをすべてのリージョンに要求する。よりよい技術は、リージョンを増加的に分離することである。これをするために、ネームスペース分割スキームは、リージョン0でスタートし、現在のレベルの最後のリージョンで終了する順番でリージョンを分離する。リージョンは、ハッシュ値のもう１つのビットの使用により分離される。図６はこのプロセスを示す。この例において、マップレベル１では、２つのリージョン６０２（ノード０）及び６０４（ノード１）があると仮定する。ノード番号は、２進法で示される。マップが増大する必要がある場合、分割スキームは、ハッシュ値のもう１つのビットの使用によりリージョン０を分離する。これは、３つのリージョン６０６、６０８及び６１０を作る。それらがリージョン６０６（ノード００）にあり、残りのオブジェクトが新しい最後のリージョン６１０（ノード１０）へ行く場合、新しいビットが０であるオブジェクトはとどまる。分裂により加えられるビットはイタリック体にされる、すなわち00と10．注目すべきは、最初のリージョン６０６及び最後のリージョン６１０が２ビットを使用し、その一方で中間（分割されていない）リージョンは１つだけ使用する、けれども、左から右まで見られた場合、番号付けをするスキームはそれでも、正確に機能する、すなわち、{0、1、2}。 更なる増大については、リージョン１は４つのリージョン６１２（ノード００）、６１４（ノード０１）、６１６（ノード１０）及び６１８（ノード１１）を作るために分割される。これは2レベルを完成させる。リージョンマップが再び増大する必要がある場合、スキームは、リージョン００～０００（つまりハッシュ値のもう１つのビットを加えることによって）を分割し、最後に、新しいリージョン１００（さらにハッシュ値のもう１つのビットを加えることによって）を加える。 その後、リージョンマップは、示されるように５つのリージョン６２０、６２２、６２４、６２６及び６２８を持つことになる。

【００３８】

リージョンの数がノードの数に相当するという必要はない。より一般に、リージョンの数は、独立したノードのアレイ中のノードの数に関連しない。

【００３９】

従って、１つの実施例によれば、リージョンに対するコントロールは、リージョンにメタデータオブジェクトを割り当て、次にリージョンを増加的に分けることで果たされる。リージョンコピー（正式、バックアップ、あるいは不完全である）は、各ノード上のデータベースに格納される。記述されたように、メタデータオペレーションは、正式なＲＧＭ

によって実行される。しかしながら、ノードが失敗する場合、いくつかの数のリージョンコピーは失われる。記述されたように、有効性は、正式なリージョンのバックアップコピーのうちの１つを昇格させることにより復元される、すなわちそれは数秒で通常できる。バックアップが昇格される短い間隔中に、ＭＭＣによってリージョンに提出される要求は失敗する。この障害は、遅れの後に再試行を引き起こすＭＭＣによって見つかる例外として現われる。要求が再試行される時には、しかしながら、ＭＭＣユーザに対してサービスが中断されないことをもたらしながら最新のマップが適所にあるようになる。記述されたように、このアプローチは、同期されたままであるリージョンのコピー（好ましくはそれらのすべて）に依存する。

【００４０】

下記は、メタデータ管理システムの追加導入詳細を提供する。

【００４１】

上記の通り、ノードがクラスタを残す場合、あるいはノードがクラスタを連結する場合、あるいは不完全なリージョンコピーがロードを終える場合、ＭＭリーダーはリージョンマップを作成する。最初のケースでは、ノードがクラスタを残す場合、一時的にあるいは永久に、そのノード上のＭＭによって管理されるリージョンを再び割り当てなければならない。ノードがサービスに戻る場合、あるいはノードが初めてクラスタを連結する場合、第２のケースはその状況を含む：そのような場合、クラスタ中の他のＭＭのためのロードを軽くするために、リージョンはそれに割り当てられる。新しいノード上で作られたリージョンは、すべて不完全である。これらのリージョンは、一旦それらがデータをロードし終えると、バックアップに昇格される。 不完全なリージョンがそのデータをロードすることを完了すると、３番目の状況が起こる。この時に、リージョンは、バックアップになる。マップ生成アルゴリズムは、好ましくは、所与のノードが正式なリージョンがクラスタにわたって平衡を保たれる、また、全てのリージョンがクラスタにわたって平衡を保たれるいかなるリージョンの１つを超えるコピーを決して含まないことを、保証する。全てのＲＧＭが全てのメタデータ更新を処理し、このようにクラスタにわたって広げられるので、後者の２つの制約は必要である。正式なＲＧＭは、また、検索要求を処理する、したがって、それらもまたよく分散される。

【００４２】

下記に、マップ生成アルゴリズムに関する追加の詳細を提供する。

【００４３】

ＭＭリーダーが新しいマップを作成する必要がある場合、それが最初にすることはリージョンセンサスである。これは、現在クラスタ中の各ノード上のＭＭに要求を送信しながら要求／応答メッセージパターンを使用して行われる。要求／応答パターンは、好ましくは、どのリージョンの全体像がアーカイブに存在するかを形成して全ての応答が組み合わせられる集合ステップを含む。リージョンセンサスによって提供される情報は、好ましくは、各リージョンコピーの用に下記を含む：リージョンコピーを所有するノード、リージョンマネージャ（もしあれば）によって処理される最後の更新、及びリージョンのデータベーススキーマに格納されるタイムスタンプ。リージョンタイムスタンプは、センサスから削除される無効のリージョンを識別するために使用される。これは、無効のリージョンが形成されているマップから外され、また、無効のリージョンスキーマが削除されることを保証する。ほとんどの場合、無効のリージョンコピーは、現在のリージョンコピーからのマップ番号より低いマップバージョン番号を持つ。しかしながら、これは必ずしもそうだとは限らないかもしれない。例えば、新しいマップがノードクラッシュにより作成されていると仮定する。リージョンセンサスは、残りのリージョンを発見し、新しいマップを形成する。機能不全のノードがリージョンセンサスに応答するのに間に合うように再開すれば、あたかもうまく行かないものもなかったかのように、ノードはそのリージョンを報告する。しかしながら、これらのリージョンはすべてノードが下がっていた間に更新を逃したため、無効になるかもしれない。この問題の解決策は、リージョンセンサスで含まれるリージョンタイムスタンプを検査することである。リージョンコピーは、最後の更新の

タイムスタンプが処理されたことを表すそれぞれのリージョンタイムスタンプを報告する。リージョンコピーが同期された状態になるので、有効なタイムスタンプは、マップバージョン変更及び最初のマップを考慮に入れなければならない。機能不全のリージョンが最新の又は無効のマップバージョン番号を持っているかどうか、これが無効のリージョンを識別する。ノードが機能しなくなり、サービスに速く戻り、次に、無効のリージョンに基づいた要求を処理し始める危険はない。この理由は、ノードがリブートでリージョンマップを持たない、また、マップが受信されるまで、ＲＧＭは存在しないからである。ＲＧＭが作成されるまで、ＭＭＣからの要求は処理できない。したがって、新しいマップを得るまで、素早く再開する機能不全のノードは、要求を処理することができない、そして新しいマップはノードにその古いリージョンを廃棄させる。

【００４４】

リージョンセンサスの後、最初のリージョンマップは以下のように生成される。リージョンセンサスが全くリージョンを放棄しない場合、クラスタは初めてスタートしているに違いない。この場合、正式なリージョンの所有者が最初に割り当てられる。各割り当てについては、アルゴリズムは、最小の使用中のノードを選択する。最小の使用中のノードは、最も少ないリージョンコピーを備えたノードである。つながりは、所有された正式なコピーの数に基づいて解決される。正式なリージョンの所有者が割り当てられた後、バックアップリージョンの所有者は、バランスのとれた権限があり合計のリージョンオーナー権を追い求めながら割り当てられる。新しいマップはすべてのＭＭに送られる、すなわち、その後、マップによって説明されたリージョンを作る。

【００４５】

一旦クラスタがスタートしたならば、マップ変更は、好ましくは、順番に次のマップ変更を行うことにより実施される：（１）リージョンに正式なコピー（ノード機能不全による）がない場合、バックアップを昇格させる；（２）、リージョンがＭＤＰＬバックアップ以上に持っている場合、超過バックアップを削除する；（３）リージョンがＭＤＰＬバックアップ（ノード機能不全、あるいは正式なことへの昇格による）より少なく持っている場合、新しい不完全なリージョンコピーを作成する；（４）オーナー権のバランスを再び取る、そして（５）正式なオーナー権のバランスを再び取る。ステップ（４）は、最も忙しいノードを見つけ、オーナー権計算が少なくとも２低いノードにそのリージョンのうちの１つを再び割り当てることを含む。（目標ノードのオーナー権計算が1低い場合、再配分は、仕こと量の平衡を保つのを助けない）好ましくは、これは、新しい不完全なリージョンを作ることにより行われる。これがいかなるノードによって所有されたリージョンの最大数を縮小し続ける限り、この操作が継続される。ステップ（５）は、権厳のあるリージョンの最大数を所有するノードを見つけることと、正式なオーナー権計算が少なくとも２低いバックアップを見つけることを含む。このステップは、例えば、バックアップを昇格させる及び正式なものを降格させることで責任を交換する。この操作は、いかなるノードによって所有される正式なリージョンの最大数を縮小し続ける限り、継続される。

【００４６】

ノードがクラスタを残す場合、その後、ステップ（１）及び（３）は、ノードの離脱によって残されたリージョンマップ中のどんなギャップも満たす。必要ならば、その後、ステップ（４）と（５）が仕事量を一定にするために使われる。

【００４７】

ノードがクラスタを連結する場合、ステップ（１）－（３）は、何も変更しない。ステップ（４）は、対照的に、新しいノードに割り当てられながら不完全なリージョンをセットもたらす。不完全なリージョンがそのデータをロードすることを完了する場合、それはＭＭリーダーに通知する。マップは、バックアップに不完全なリージョンを昇格させる。その後、ステップ（５）は、新しいノードに正式なリージョンを割り当てる効果がある。

【００４８】

不完全なリージョンがその同期を終了する場合、それはバックアップリージョンに変わり、ＭＭリーダーに通知する。その後、ＭＭリーダーは、少なくとも１つのリージョン用

のＴＰＯＦバックアップ以上に含んでいる新しいマップを発行する。ステップ（２）は、最も極度にロードしたＭＭに対する負担を軽くすることを選びながら、超過しているバックアップリージョンを削除する。

【００４９】

ＭＭが新しいマップを受信する場合、それは、新しいマップと現在のものとを比較し、ＭＭによって管理される各リージョンのためにいかなる変更をも適用する必要がある。可能な変更は、以下の通りである：リージョンを削除する、リージョンを作る、バックアップリージョンを正式なものに昇格させる、バックアップに不完全なリージョンを昇格させ、バックアップに正式なリージョンを降格させる。最初のタイプの変更に関して、ロードバランスは、コピーの削除をもたらせながら、あるノードから別のノードにリージョンコピーのコントロールを移動させることができる。そのような場合、ネットワークとデータベース資源がリージョンのデータを格納するスキーマの削除を含めながら返される。正式なそしてバックアップリージョンが作られるとともに、リージョンを作る第2のタイプの変更が典型的に新しいクラスタに生じる。その後、不完全なリージョンだけが作られる。リージョン生成は、各タイプのメタデータオブジェクト用のテーブルを含んでいるデータベーススキーマを作成することを含む。各リージョンのスキーマは、リージョン（正式、バックアップ、あるいは不完全）の役割を識別する情報を含んでいる。３番目のタイプの変更、すなわち、バックアップから正式への昇格は、リージョンの役割の修正を必要とする。他の変更タイプは、それらの名前が意味するように、不完全からバックアップへあるいは正式からバックアップへのリージョンの役割を変更することを含む。

【００５０】

ノードのメタデータマネージャはそれぞれ、全部のクラスタ用のメタデータの所与の部分をコントロールする。したがって、所与のノードに格納されるメタデータは、クラスタ中のすべての（あるいは所与のサブセットの）ノード中に理論上平等に分散されているデータベースと共に、分散型データベース（メタデータの）の一部から成る。メタデータマネージャは、説明されてきたように、この機能を達成するために協力する。新しいノードがクラスタに加えられる場合、個々のノード責任は、新しいキャパシティに調節される；これは新メンバーが等しい割り当てを仮定するように、すべてのノードにわたってメタデータを再分散することを含む。反対に、ノードが機能しなくなるか、クラスタから取り除かれる場合、他のノードメタデータマネージャは、より大きな割り当てを仮定することにより縮小されたキャパシティを補う。データロスを防ぐために、ノードがそれぞれ直接すべてのクラスタメタデータのあるパーセンテージを管理する責任があり、他のノードのセット番号にこのデータをコピーする場合、メタデータ情報は、好ましくは、複数のノードにわたって再現される。

【００５１】

新しいマップが生成される場合、ＭＭリーダーは、他のノードへそのマップの分散を始め、全てのノードがそれを持つまで処理の保留を要求する。 一旦ノードがすべて新しいマップを持っていることをシステムが確認すれば、通常の処理が再開される。

【００５２】

増分リフレッシュ

【００５３】

上記の通りシステムにおいて、メタデータは、システムでノードにわたって冗長格納されているリージョンへ分散される。メタデータマネージャは、これらのリージョンの位置を含んでいるリージョンマップを持っていて、システムはこれによって適切に要求を送ることができるようになる。リージョンの数は、メタデータロードがすべてのノードにわたって分割される粒度を決定する。マップは、以下の段階、すなわち、正式なリージョンコピー（（Ａ）リージョン）、バックアップリージョンコピー（（Ｂ）リージョン）、及び、スクラッチからあるいは（Ａ）または（Ｂ）のリージョンから復元される過程中にある不完全なリージョンコピー（（Ｉ）リージョン）、におけるリージョンのコピーを含んでいる。（Ａ）または（Ｂ）のリージョンがクラスタを残す場合、マップがリフレッシュさ

れる。（Ｂ）リージョンがクラスタを残している場合、（Ｉ）リージョンは、（Ａ）リージョンからメタデータをすべてコピーすることにより作られ追加される。（Ａ）リージョンがクラスタを残す場合、対応する（Ｂ）リージョンは、（Ａ）に昇格され、次に、（Ｉ）リージョンが作られる。（Ｉ）リージョンコピーが完全な場合、それは（Ｂ）リージョンへ昇格される。リージョンがすべて完成した場合、マップは再度ＭＤＰＬにある。このプロセス中に、失われたリージョンが戻る場合、そのタイムスタンプが無効であるので典型的に捨てられる。

【００５４】

この開示によれば、リージョンへの最後に適用された変更を格納する概念に基づくエンハンスされたリカバリスキームが説明される。現在のタイムスタンプだけでなくその値は戻るリージョンをどうするか決めるために使用される。

【００５５】

したがって、（Ａ）または（Ｂ）のリージョンがすぐに（例えば単純なノードリブートにより）戻る場合、ほとんどの場合、リージョンは、ちょうど１更新だけ異なる。このケースを確認するために、この開示によれば、好ましくは、最後の更新は各リージョンのリージョンタイムスタンプテーブルで続けられる。最初のシナリオでは、正に最後の更新で見つからないリージョンがある。これは、（Ａ）リージョンから更新を受信する前に、（Ｂ）リージョンが消失する場合である。この場合、システムは、現在のマップでそのリージョンを最新にさせるために、最後の更新を適用する。更新を適用した後に、マップインストールの残りは通常通り継続する。２番目のシナリオは、システムの他のいかなる場所にも繁殖しなかったリージョンに適用された更新があるということである。これは、（Ａ）リージョンの更新がその（Ｂ）リージョンに適用される前に、（Ａ）リージョンが消失する場合である。この場合、その更新は、無効であると考えられ、そのリージョンがマップに戻される前に、そのリージョンから取り除かれる。この場合、更新の除去が行われる間に、下記の通り、システムは「Ｐ」リージョンと呼ばれるものを作成する。

【００５６】

最後に、正式なコピー（例えば戻るノードが利用不可能だった間に、リージョンへ書き込みがなかった場合）と戻るリージョンコピーが全く異ならないケースがある。このケースでは、上記のリージョンタイムスタンプの比較によって、メタデータマネージャリーダーは、戻るリージョンが完全に最新で、したがって、サービスに直ちに（バックアップ（Ｂ）リージョンとして）戻ることができることを決めることができる。リージョンタイムスタンプ比較は、ＭＭリーダーが暫くの間見つからなかったかもしれない不正確に戻るリージョンコピーを検査することを可能にする１以上の失敗したマップを説明するのに十分に柔軟である。

【００５７】

ここでは、部分的、あるいは、「Ｐ」段階は、あるリージョンタイムスタンプの時点で最新の、そして現在のマップで最新でないあらゆるリージョンに当てはまるが、別のリージョンコピー上に格納された保留の更新を当てはめることにより最新にできる。システムは、リージョンコピーが失われたことを検知する場合、（Ｉ）リージョンが直ちに作られるので（前に述べたように）及びその（Ａ）リージョンが受信する（リージョンコピーが失われたマップからの最後の更新で始まる）全ての更新をそれらの（Ｉ）リージョンが受信するので、（Ｉ）リージョンコピーは、（Ｐ）リージョンを最新にするために必要とされる保留の更新を正確に含んでいる。

【００５８】

したがって、この開示によれば、（Ｐ）リージョンは、先在する（Ａ）か（Ｂ）リージョンから復元されているリージョンを表わす。（Ｐ）リージョンはそれぞれ、（Ｉ）リージョン（（Ｐ）リージョンコピーを最新にさせるために必要とされる保留の更新を正確に含んでいる）に格納された保留の変更から更新される。（Ｂ）または（Ｉ）リージョンのように、（Ｐ）リージョンが作られるとすぐに、（Ａ）リージョンは、すべての更新のためにバックアップリージョンとしてそれを扱う。（Ｐ）リージョン（それが作成されると

すぐに）は、バックアップ要求を受信し、pending_updateテーブル（あるいは他の便利な持続メカニズム中で）にそれらを格納する。（Ｐ）リージョンが更新し終えた場合、それは（Ｂ）リージョンに変換される。この変換に際して、（Ｉ）リージョンは撤去される。

【００５９】

最後の更新を削除する必要のある場合、Ｐリージョンは、それが他のリージョンコピー上で適用されなかった場合に適用された最後の更新を最初に取り消す。

【００６０】

図７は、第１の例のシナリオを説明する。ここで、（Ｂ）リージョン７００は失われており、（Ａ）リージョン７０２から更新（１００，４）を一度も受信していない。この場合に、（Ｉ）リージョン７０４は作られる。（Ａ）リージョン７０２からメタデータをコピーすることにより（Ｉ）リージョンは追加される（示されるように）；（Ｉ）リージョン７０４は、また、（Ａ）リージョン７０２へのどんなそれに続く更新も受信し始める。（Ｂ）リージョン７００が戻る場合、それを含んでいた最後のマップにおいて最新だったため、それは（Ｐ）リージョン７０６に変換される。（Ｐ）リージョン７０６は、（Ｉ）リージョン７０４の保留の更新テーブルからその見つからない更新を追加する。（Ｉ）リージョン７０４が最新である前に、このプロセスが終わる場合、（Ｐ）リージョン７０６は、（Ｂ）リージョンへ昇格され、（Ｉ）リージョンが廃棄される。

【００６１】

図８は、第２の例を説明する。ここで、更新（１００，４）（B）リージョン８０２を適用する前に、（Ａ）リージョン８００はダウンする。リージョン（Ａ）がクラスタを残す場合、（Ｂ）リージョンは新しい（Ａ）リージョンへ昇格される。新しい（Ｉ）リージョン８０４は、リージョン（Ａ）から新しい更新を受信するために作られる。前の（Ａ）リージョンが戻る場合、それは（Ｐ）リージョン８０６に変換される。上記の通り、（Ｐ）リージョン８０６は、リージョン（Ｉ）８０４の保留の更新テーブルからのその見つからない更新を追加する。（Ｐ）リージョンは、また、有効でない最後の更新を取り消す。リージョン（Ｉ）が最新である前に、このプロセスが終わる場合、リージョン（Ｐ）８０６は、（Ｂ）リージョンへ昇格され、（Ｉ）リージョンが落とされる。

【００６２】

したがって、図７は、どのように一例、（Ｂ）リージョンのリストアの次の損失中に、（Ｐ）リージョンコピー（Ｐリージョン）が作成され、次に、（Ｂ）リージョンへ昇格されるかを説明する。図８は、どのように一例、（Ａ）リージョンのリストアの次の損失中に、（Ｐ）リージョンコピーが作成され、次に、（Ｂ）リージョンへ昇格されるかを説明する。

【００６３】

マップが存在するとそのマップが示す場合、（Ｐ）リージョンが作られ、既存のリージョンは、（Ｐ）リージョンに変換される。その場合に、リージョンも、それがその最後の更新を後退させる必要があり、どの（Ｉ）リージョンからコピーするかをマップ中で伝えられる。

【００６４】

クラスタが、既存の正式なリージョンからバックアップリージョンを移動又は作る必要がある場合、リフレッシュタスク（RefreshTaskと呼ばれる）が始められる。そのタスクの中心となるのは、ソース及びターゲットマシンの両方におけるコピーテーブルスクリプトの起動である。スクリプトはSQLに基づくデータベースと直接交信する。RefreshTaskは、（Ｐ）リージョンの機能性を実施する。その他のオペレーションの前に、（Ｐ）リージョンは、適用された最後の更新が他のリージョンコピー上で適用されなかった場合、それを最初に取り消さなければならない。これは、Undo Last Updateと呼ばれるサブタスクである。そのため、RefreshTaskは、更新又は（Ａ）リージョンから同等の列のコピーによって影響されたかもしれないデータベーステーブル中のいかなる列のローカルコピーを削除する。単一の保留の更新中に複数の更新があるかもしれないので、このプロセスは、保留の更新中ですべての個々の更新を処理する。各更新については、タスクは、影響される

データベーステーブルと影響を受けた列のSQL（構造化照会言語）WHERE句を決定する。このWHERE句は、次に、その更新用の影響を受けたテーブル中のいかなるローカルの列を最初に削除するのに使われ、（Ａ）リージョンソース（copytableスクリプトを使用して）からのすべての同様の列をコピーする。Undo Last Updateによって修正される必要のあるメタデータのサブセットを選択するために、その句は、メタデータに述語を適用する。

【００６５】

その後、RefreshTaskは、Copy Pending Updatesと呼ばれるサブタスクを始める。これは、マップ中で指定される（Ｉ）リージョンからpending_updateテーブルをコピーする。このコピーは、正常なバックアップ要求から入って来るpending_updatesとの対立を避けるためにpending_updateテーブルではなくpending_update_copyテーブルをターゲットとする。コピーが完全な場合、RefreshTaskは、2つのpending_updateテーブルを統合させる。これは、pending_updatesをソートし、どんな重複も除外する。

【００６６】

その後、RefreshTaskは、Apply Pending Updatesと呼ばれるサブタスクを始める。具体的には、一旦pending_updateテーブルがフル実装されると、（Ｐ）リージョンは、好ましくは、２パスに、pendings_updatesを適用する。このプロセスの終わりに、リージョンのregion_timestampは通常更新される。

【００６７】

一旦、保留の更新が適用されると、RefreshTaskは、バックアップリージョンへConvertと呼ばれるサブタスクを行なう。ここで、（Ｐ）リージョンは、それ自体を（Ｂ）リージョンに変換する。

【００６８】

（Ｐ）リージョンが、（Ｉ）リージョンをリフレッシュするコストを下げるので、（Ｉ）リージョンは、（Ｐ）→（Ｂ）変換が完了するまで前進し続ける。これを遂行するために、マップは、変換が起こる場合にＩリージョンを落すように指定する使い捨てとしてそれらをマークするフラグを含む。Ａリージョンは、そのＩリージョンにバックアップ更新を送るのをやめることを知る必要があるので、この変換は、Ａリージョンによって統合される。（Ｐ）→（Ｂ）変換が起こる場合、（Ａ）リージョンは、ロックを解除し、なんらかの使い捨てのIリージョン自体を除外するためにそれらのリージョンのそれぞれに１つの新しいメッセージを送りながら、発表されるリージョン用のローカルマップからそれらを撤去することによって、このメッセージに反応するために１つの新しいリスナーを持つ。（Ｐ）又は（Ｉ）リージョンRefreshTaskのいずれかのステップが機能しなくなる場合、そのリージョンは、再度使用されることができないようになる。

【００６９】

上記の通り、増分リフレッシュは、多数の利点をもたらす。主要な利点は、（Ｉ）リージョンの長く高価なリフレッシュを回避することである。具体的には、リージョンセンサス（マップ生成プロセスの一部としての）がＭＤＰＬコピーより少なく持っているリージョンを識別する場合、（Ｉ）リージョンは、新しいコピーとして作られる。この（Ｉ）リージョンは、リージョンの（Ａ）コピーから完全なスキーマをコピーするためにRefreshTaskを起動する。このRefreshTaskは、データをコピーし、インデックスの作成を含むいくつかの高価なステップを持っている。ここに記述される技術の使用によって、失われたデータベースの部分に生じる更新だけが適用され、このオペレーションは、はるかに少ない時間を要し、冗長性を通じてその部分を再構築するよりはるかに少数のシステムリソースを消費する。修復する時間の短縮は、また、クラスタがＭＤＰＬ及びピーク性能以下である時間を短縮する。

【００７０】

ここに記述されるようなアーカイブ管理ソリューションは、デジタル資産のキャプチャー、保存、管理及び検索を可能にする。設計は、多数の必要条件のアドレスを指定する：既存のアプリケーションを備えた統合の無制限のストレージ、高い信頼度、自主管理と各種規格との適合、ハードウェア・インディペンデンス及び軽減。

【００７１】
Linux（例えば）を実行するコモディティハードウェアのクラスタは、ロバストプラットフォーム及びこと実上無制限のアーカイブを提供する。システムは、計ることができる、例えば、少数のストレージノードサーバから何千ものテラバイトデータを格納する多くのノードまで 。アーキテクチャは、ストレージ容量が、組織の増加するアーカイブ要求と歩調を常に合わせることができることを保証する。
【００７２】
システムは、ファイルを決して失わないことを指定される。アーカイブがデバイス障害から常に保護されるように、それは、クラスタにわたってデータを複製する。ディスク又はノードが機能しなくなると、クラスタは、同じデータの複製を維持するクラスタ中の他のノードにわたって自動的に機能しなくなる。
【００７３】
システムは、自律的処理を通じてアーカイブストレージのコストを下げる。例えば、ノードがクラスタ化されたアーカイブを連結するか離れる場合、システムは自動的にクラスタのロードバランスを調節し、メンバーノードにわたってファイルを再分散することによりパフォーマンスを最適化する。
【００７４】
システムは、ユーザ定義の保存政策への従順を促進する。
【００７５】
システムは、オープンプラットフォーム上で展開することによりハードウェア依存を排除する。コモディティプラットフォーム及び所有者の記憶デバイス間のコストギャップが大きくなると、ＩＴバイヤは、もはや高いコストの電気器具ベンダーとの関係にはまりたくない。所与のノードがコモディティハードウェア及びオープンソース（例えばLinux）オペレーティングシステムソフトウェア上で典型的に動作するので、好ましくは、バイヤは、最良のソリューションのために多くのハードウェア選択肢の中で買い物をすることができる。
【００７６】
システムは、また、ファイルを格納し検索するNFS、HTTP、FTP及びCIFSのような業界基準インターフェースを提供する。これは、システムが、カスタマイズされたアーカイブアプリケーションと同様に、ほとんどの標準コンテンツ管理システム、検索システム、ストレージ管理ツール（ＨＳＭとバックアップシステムのような）に容易に接続することができることを保証する。
【００７７】
上記のものは、ある実施例によって行なわれたオペレーションの特定な順序について記述しているが、代替実施例が異なる順にオペレーションを行い、あるオペレーションを組み合わせ、あるオペレーションをオーバーラップさせる等を理由にそのような順序が例となるということを承知してほしい。所与の実施例へのスペック中の参照は、記述される実施例が特別な特徴、構造あるいは特性を含むかもしれないことを示し、しかし、すべての実施例が必ずしも特別な特徴、構造あるいは特性を含むとは限らないかもしれない。
【００７８】
公開された技術は、方法又はプロセスのコンテキストに述べられているが、ここでの内容は、また、ここでのオペレーションを行なうための装置に関係がある。 この装置は、所要の目的のために特に構築されてもよい、もしくは、それは、コンピュータに格納されたコンピュータプログラムによって選択的に活性化されるか再構成される汎用計算機から成ってもよい。そのようなコンピュータプログラムは、例えば、光ディスクを含む任意のタイプのディスク、ＣＤ－ＲＯＭ及び光磁気ディスク、読み出し専用メモリ（ＲＯＭ）、ランダムアクセスメモリ（ＲＡＭ）、磁気又は、光カード、あるいは電子マニュアルを格納するのにふさわしい任意のタイプの媒体、そしてそれぞれがコンピュータシステムバスに接続されているだがこれに限らないは、コンピュータ読取り可能な記憶媒体に格納されてもよい。

【００７９】
システムの所与のコンポーネントは、別々に説明されているが、通常のスキルのうちの１つは、所与の指示、プログラムシーケンス、コード部分などにおいて機能のうちのいくつかが組み合わせられる又は共有されるかもしれないことを認識する。
【００８０】
「固定コンテンツ」のためのアーカイブのコンテキストで本発明を述べたが、これもまた限定ではない。本明細書に述べた技術は、コンテンツに対する付加タイプ及び置換タイプの修正を可能にするストレージシステムにも等しく適用することができる。
本発明について述べてきたが、次に以下のとおり特許請求する。

【図１】

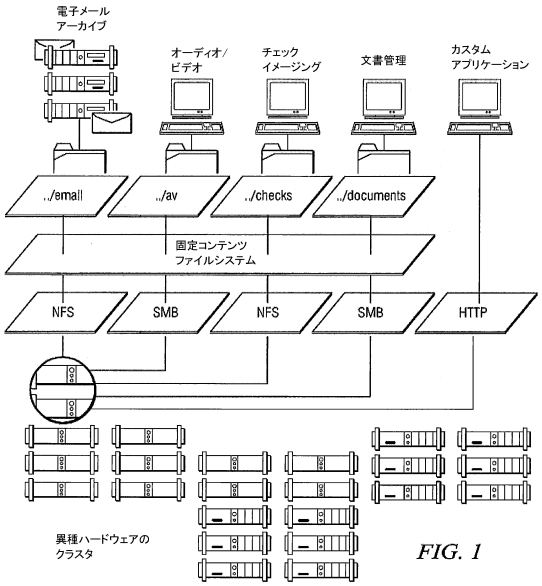

FIG. 1

【図２】

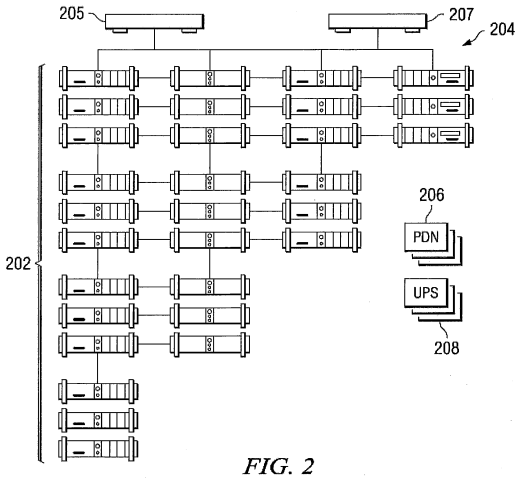

FIG. 2

【図３】

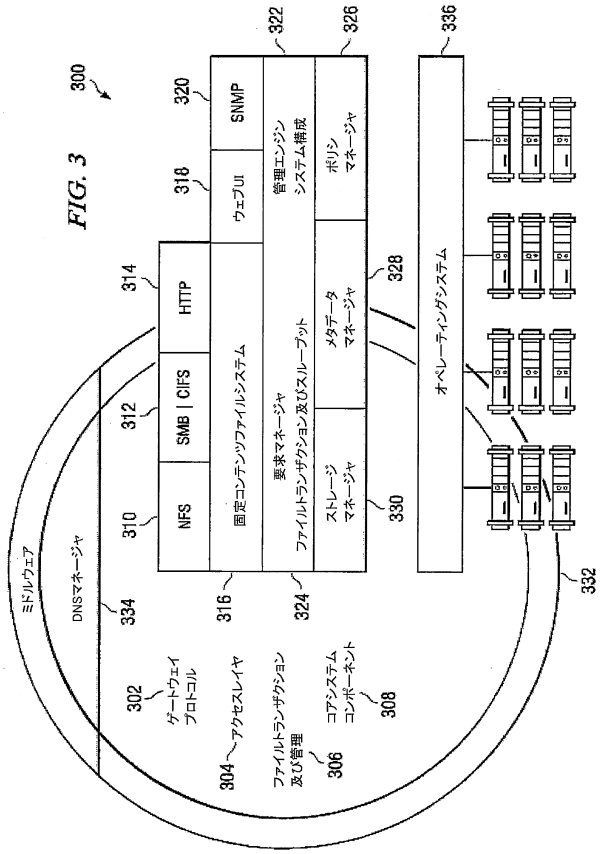

FIG. 3

【図４】

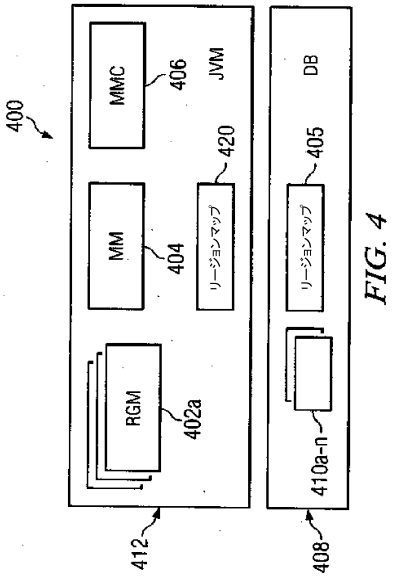

FIG. 4

【図５】

リージョン	正式	BACKUP 1	BACKUP 2
0	n1	n2	n3
1	n2	n3	n4
2	n3	n4	n1
3	n4	n1	n2
4	n1	n2	n3
...	...	...	...
63	n4	n1	n2

FIG. 5

【図６】

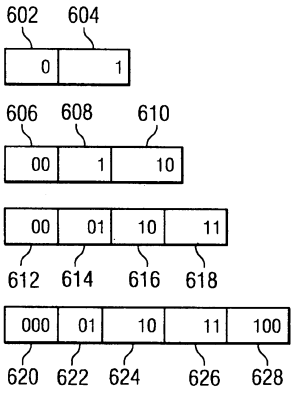

FIG. 6

【図７】

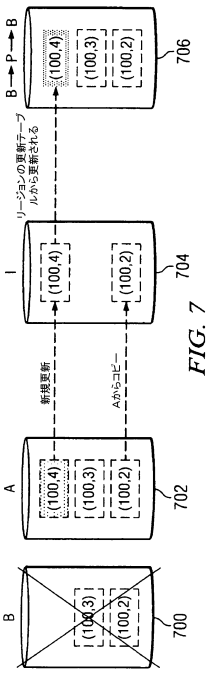

FIG. 7

【図８】

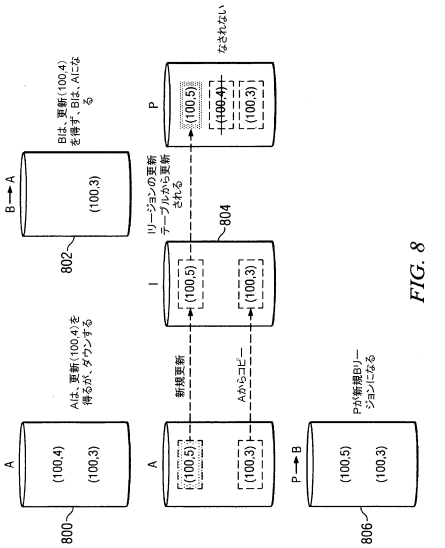

FIG. 8

フロントページの続き

(72)発明者　グリマルディ，ケビン　エス.
アメリカ合衆国　０２１４４　マサチュセッツ州　サマービル，ベイステートアベニュー２番，３８
(72)発明者　パーマー，トレック
アメリカ合衆国　０２１３９　マサチュセッツ州　ケンブリッジ，パットナムアベニュー１番，２７
(72)発明者　ピンクニー，デビッド
アメリカ合衆国　０１８１０　マサチュセッツ州　アンドーバー，タングルウッドウェイ　エヌ.，８

審査官　金沢　史明

(56)参考文献　特表２００８－５１８２８４（ＪＰ，Ａ）
特開２００１－２９０６８７（ＪＰ，Ａ）
特開２００６－０７２４５０（ＪＰ，Ａ）
特開２００５－１４１５２８（ＪＰ，Ａ）

(58)調査した分野(Int.Cl.，ＤＢ名)
Ｇ０６Ｆ　１２／００