(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2005-18738
(P2005-18738A)
(43)公開日　平成17年1月20日(2005.1.20)

(51)Int.Cl.[7]	F I		テーマコード（参考）
G06F 12/00	G06F 12/00	531J	5B082
	G06F 12/00	501B	

審査請求 未請求　請求項の数 31　ＯＬ　外国語出願　（全 45 頁）

(21)出願番号　特願2004-85961 (P2004-85961)
(22)出願日　平成16年3月24日 (2004.3.24)
(31)優先権主張番号　10/608391
(32)優先日　平成15年6月26日 (2003.6.26)
(33)優先権主張国　米国 (US)

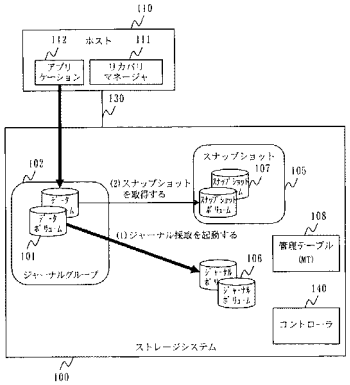
(71)出願人　000005108
株式会社日立製作所
東京都千代田区丸の内一丁目６番６号
(74)代理人　100075096
弁理士　作田　康夫
(74)代理人　100100310
弁理士　井上　学
(72)発明者　山神　憲司
アメリカ合衆国カリフォルニア州ロスガトス　カイルニベル　１０８

Ｆターム(参考)　5B082 CA14　DD08

(54)【発明の名称】ストレージベースのジャーナリングを用いてバックアップ及びリカバリを行う方法と装置

(57)【要約】
【課題】
然しながら、データベースシステムを除くと、データを任意時点に回復する手段は用意されていない。データベースシステムにしても、ジャーナルを適用するには下記の手順の為に時間が必要である。
・ストレージ（例えばディスク）からジャーナルデータを読み込む。
・ジャーナル内で目的のデータを特定するのに解析が必要である。
・データになされた活動を再生するには、ジャーナルデータを適切なデータイメージに適用しなければならない。このことは、イメージへのアクセスが通常必要でジャーナルが適用される度にデータの書き出しが必要である。
【解決手段】
ストレージシステムは、複数のジャーナルエントリと一つ以上のデータボリュームの少なくとも一つのスナップショットのジャーナルを維持する。ジャーナルとスナップショットの各々に発生順に固有な順序番号を割り当

図１　バックアップの概要



【特許請求の範囲】
【請求項１】
アプリケーションデータの保存において、ホストからの書き込み要求の受信によって保存されるアプリケーションデータに対する少なくとも最初のスナップショットを生成するステップと、前記ホスト要求による各書き込み動作に対応してジャーナルエントリを生成するステップと、各ジャーナルエントリをジャーナルデータ領域に保存し、該ジャーナルエントリのリストを集積するステップと、前記ジャーナルデータ領域内の未使用容量をモニタするステップと、前記未使用容量が第一の限界値より減少したら、該未使用容量が第二の限界値より増大するまで、前記ジャーナルデータ領域から１ないしはそれ以上のジャーナルエントリを除去することで、未使用容量を増大させるステップと、から成ることを特徴とするデータ処理方法。
【請求項２】
１ないしはそれ以上のジャーナルエントリを除去する為に、前記最初のスナップショットにその時の最古のジャーナルエントリより始めて順にジャーナルエントリを適用することにより該最初のスナップショットを更新し、前記最初のスナップショットに適用されたジャーナルエントリが、前記ジャーナルエントリリストから除去されることで前記ジャーナルデータ領域の未使用容量を増大させることを特徴とする請求項１の方法。
【請求項３】
１ないしはそれ以上のジャーナルエントリを除去する為に、その時の最古のジャーナルエントリより古くてかつ最も近いスナップショットを特定して、該スナップショットを該最古のジャーナルエントリより始めて順に１ないしはそれ以上のジャーナルエントリを適用して更新することを特徴とする請求項１の方法。
【請求項４】
１ないしはそれ以上のジャーナルエントリを除去する為に、最新のスナップショットを見つけて、該最新のスナップショットより古いジャーナルエントリを前記ジャーナルデータ領域より除去することを特徴とする請求項１の方法。
【請求項５】
前記第一の限界値と第二の限界値は異なった値であることを特徴とする請求項１の方法。
【請求項６】
各ジャーナルエントリは固定長のヘッダ部分と可変長のデータ部分から構成され、前記ジャーナルデータ領域は複数のヘッダ部分が定義される第一のストレージ領域と複数のデータ部分を保存する第二のストレージ領域で構成され、ジャーナルエントリを生成する為にジャーナルヘッダのひとつを割り当て書き込み動作に対応するデータを収容するためにデータ部に十分なスペースを割り当てることを特徴とする請求項１の方法。
【請求項７】
各ジャーナルエントリを保存する為に、ジャーナルヘッダが開始位置から始まって常に順番に割り付けられ、最終位置にジャーナルヘッダが割り当てられたら、該開始位置に戻って再度順番に割り当てられることを特徴とする請求項６の方法。
【請求項８】
ホストからの書き込み要求の受信によって保存されるアプリケーションデータの少なくとも一部分に対する少なくとも最初のスナップショットを生成するステップと、ホストからの書き込み要求動作に対応してジャーナルデータ領域に保存され、該ジャーナルデータ領域の未使用容量を消費する複数のジャーナルエントリを記録するステップと、少なくとも前記最初のスナップショットを少なくとも一つのジャーナルエントリで更新して、該少なくとも一つのジャーナルエントリが消費している記憶スペースを未使用化して前記ジャーナルデータ領域の未使用容量を増大させるステップと、から成ることを特徴とするデータの処理方法。
【請求項９】
前記更新ステップは、前記ジャーナルデータ領域の未使用容量を定期的にモニタして、

該未使用容量が第一の限界値以下になったら、少なくとも前記最初のスナップショットを更新するステップを含むことを特徴とする請求項８の方法。

【請求項１０】

少なくとも前記最初のスナップショットを更新するステップを前記未使用容量が第二の限界値を超えるまでジャーナルエントリに対して繰り返すことを特徴とする請求項９の方法。

【請求項１１】

更に、前記最初のスナップショットとジャーナルエントリに各々順序番号を割り当てるステップと、目標時刻を受信するステップと、前記ジャーナルエントリと前記最初のスナップショットの各々に割り当てられた順序番号に基づいて、開始ジャーナルエントリを決定するステップと、前記最初のスナップショットに前記開始ジャーナルエントリを適用することにより前記最初のスナップショットを更新するステップと、前記最初のスナップショットに、前記開始ジャーナルエントリに引き続く前記目標時刻以前のジャーナルエントリを適用することにより、前記最初のスナップショットの追加更新を実行するステップと、を含むことを特徴とする請求項８の方法。

【請求項１２】

更に、追加のスナップショットを生成して、複数のスナップショットを集積するステップと、前記スナップショットとジャーナルエントリに各々順序番号を割り当てるステップと、目標時刻を受信するステップと、前記目標時刻に基づいて一つのスナップショットを選択するステップと、前記ジャーナルエントリと前記選択されたスナップショットの各々の順序番号に基づいて、一つの開始ジャーナルエントリを決定するステップと、前記選択されたスナップショットに前記開始ジャーナルエントリを適用することにより前記選択されたスナップショットを更新するステップと、前記開始ジャーナルエントリに引き続き、前記目標時刻以前のジャーナルエントリを前記選択されたスナップショットに適用することにより、前記選択されたスナップショットの追加更新を実行するステップと、から成ることを特徴とする請求項８の方法。

【請求項１３】

前記選択されたスナップショットは、最初のスナップショットであることを特徴とする請求項１２の方法。

【請求項１４】

前記選択されたスナップショットは前記目標時刻以前でかつ最も近いスナップショットであることを特徴とする請求項１２の方法。

【請求項１５】

ホストからの書き込み要求の受信によって保存されるアプリケーションデータの１ないしはそれ以上の部分で構成される複数のスナップショットを生成するステップと、ホストから要求される各書き込み動作に対応して生成され、ジャーナルデータ領域に保存されリスト状に集積され、該ジャーナルデータ領域の未使用容量を消費するジャーナルエントリを記録するステップと、前記ジャーナルデータ領域内の残未使用容量をモニタするステップと、前記残未使用容量が第一の限界値より少なくなったら、一つのスナップショットを選択し該選択されたスナップショットより古いジャーナルエントリを除去することにより１ないしはそれ以上のジャーナルエントリを除去するステップと、から成ることを特徴とするデータ処理方法。

【請求項１６】

前記選択されたスナップショットは、その時の最古のジャーナルエントリより少なくとも決められた時間だけ新しいスナップショットであることを特徴とする請求項１５の方法。

【請求項１７】

更に、前記スナップショットとジャーナルエントリに各々順序番号を割り当てるステップを含み、前記選択されたスナップショットは、その時の最古のジャーナルエントリの順序番号より決められた量だけ大きい順序番号を持つスナップショットであることを特徴と

する請求項１５方法。
【請求項１８】
ホストからの書き込み要求の受信によって保存されるアプリケーションデータの複数のスナップショットを生成するステップと、ホストから要求される各書き込み動作に対応してジャーナルエントリを生成するステップと、各ジャーナルエントリをジャーナルデータ領域に保存し、ジャーナルエントリのリストを集積するステップと、前記ジャーナルデータ領域内の未使用容量をモニタするステップと、前記未使用容量が第一の限界値より少ないときには、その時の最古のジャーナルエントリより古くてかつ最も近いスナップショットを選択するステップと、前記最古のジャーナルエントリより始まる１ないしはそれ以上のジャーナルエントリにより前記選択したスナップショットを更新し、前記ジャーナルエントリのリストを減少させるステップと、前記未使用容量が第二の限界値を超えるまで追加のジャーナルエントリによる更新を繰り返すステップとから成ることを特徴とするデータ処理方法。
【請求項１９】
更に、前記スナップショットとジャーナルエントリに各々順序番号を割り当てるステップを含み、前記選択されたスナップショットは、該スナップショットの順序番号とその時の最古のジャーナルエントリの順序番号とに基づいて決定されることを特徴とする請求項１８の方法。
【請求項２０】
各ジャーナルエントリは、ジャーナルエントリのシーケンシャルリストで表現されるように、順番に連続してジャーナルデータ領域に保存され、ジャーナルエントリのリストが終端に達したら、最初のリスト部分に折り重ねて保存し該領域を再使用することを特徴とする請求項１８の方法。
【請求項２１】
ホストからの書き込み要求の受信によって保存されるアプリケーションデータの複数のスナップショットを生成するステップと、ホストから要求される各書き込み動作に対応して生成され、ジャーナルデータ領域に保存され、該ジャーナルデータ領域内でリスト上に集積されるジャーナルエントリを記録するステップと、複数の順序番号を生成するステップと、前記スナップショットとジャーナルエントリの各々に、前記順序番号を順に割り当てるステップと、前記ジャーナルデータ領域内の未使用容量をモニタするステップと、前記未使用容量が第一の限界値より少なくなったら、各スナップショットの順序番号をその時点の最古のジャーナルエントリの順序番号と比較して、一つのスナップショットを選択するステップと、前記選択したスナップショットを前記最古のジャーナルエントリに始まる１ないしはそれ以上のジャーナルエントリで順に更新するステップと、前記未使用容量が第二の限界値を超えるまで、追加のジャーナルエントリによる更新を繰り返すステップと、から成ることを特徴とするデータ処理方法。
【請求項２２】
ジャーナルエントリを記録するステップはスナップショット生成に先立って起動され、最初のスナップショットが生成されている期間に発生した各書き込み動作はジャーナルエントリに記録されることを特徴とする請求項２１の方法。
【請求項２３】
前記選択されたスナップショットは、前記最古のジャーナルエントリより以前のスナップショットであることを特徴とする請求項２１の方法。
【請求項２４】
前記選択されたスナップショットは、前記最古のジャーナルエントリより以前でかつ他のスナップショットに比べて前記最古のジャーナルエントリに最も近いスナップショットであることを特徴とする請求項２１の方法。
【請求項２５】
前記第一の限界値は前記第２の限界値と異なっていることを特徴とする請求項２１の方法。

【請求項２６】
データストレージシステムに送られた各書き込み動作に対応してジャーナルエントリを生成し、複数個のジャーナルエントリをバックアップストレージシステムに保存する、ジャーナル処理を起動するステップと、前記ジャーナル処理起動に引き続いて、前記データストレージシステムに保存されたデータの一部分に対して一ないしはそれ以上のスナップショットを取得するステップと、各ジャーナルエントリとスナップショットが生成される度に、同じシーケンス系列で構成される順序番号を各々順に割り当てるステップと、前記バックアップストレージシステム内の使用可能ストレージ容量に基づいて、該バックアップストレージシステム内で前記ジャーナルエントリによって消費されているスペース量を減少させる為に、一つのスナップショットに幾つかのジャーナルエントリを適用すべきかどうか決定するステップと、から成ることを特徴とするデータストレージシステムにおけるデータ処理方法。
【請求項２７】
一つのスナップショットに幾つかのジャーナルエントリを適用すると決定されたら、該ジャーナルエントリとスナップショットの各々に割り当てられた順序番号に基づいて、一つのスナップショットを選択するステップと、前記選択されたスナップショットより新しくかつ最古のジャーナルエントリを特定するステップと、前記最古のジャーナルエントリから始まる一つ以上のジャーナルエントリで前記選択されたスナップショットを更新するステップと、
から成ることを特徴とする請求項２６の方法。
【請求項２８】
ホストからの書き込み動作を受信するように構成された生産用データ領域と、前記生産用データ領域の少なくとも一部分のスナップショットを一つ以上保存するように構成されたスナップショットデータ領域と、ホストからの各書き込み動作に対応してジャーナルエントリを生成し、一つ以上のジャーナルエントリを保存するように構成されたジャーナルデータ領域と、前記生産用データ領域とスナップショットデータ領域にアクセスするステップと、前記ジャーナルデータ領域にアクセスするステップと、全体の中の幾つかのジャーナルエントリにより一つのスナップショットを更新し、幾つかのジャーナルエントリをデータ回復の為に温存するステップとを実行するコントローラと、から構成されることを特徴とするデータ処理の為のストレージシステム。
【請求項２９】
前記コントローラは更に、前記スナップショットとジャーナルエントリに各々順序番号を割り当てるステップと、目標時刻を受信するステップと、前記目標時刻に基づき一つのスナップショットを選択するステップと、前記ジャーナルエントリと選択されたスナップショットの各順序番号に基づき一つの開始ジャーナルエントリを決定するステップと、前記開始ジャーナルエントリを前記選択されたスナップショットに適用することにより前記選択されたスナップショットを更新するステップと、前記開始ジャーナルエントリより新しく、前記目標時刻より以前のジャーナルエントリを前記選択されたスナップショットに適用することにより、前記選択されたスナップショットに追加の更新を実行するステップと、を実行するように構成されていることを特徴とする請求項２８のストレージシステム。
【請求項３０】
一つ以上の書き込み動作を含む交信をホストから受信する手段と、生産用データを保存する為の第一の保存手段と、スナップショットを保存する為の第二の保存手段と、ジャーナルエントリを保存する為の第三の保存手段と、前記第二の保存手段に、生産用データの少なくとも一部の一つ以上のスナップショットを保存し、各書き込み動作に対応してジャーナルエントリを前記第三の保存手段に記録し、複数のジャーナルエントリを生成し、該第三の保存手段の未使用容量を減少させ、一つ以上のジャーナルエントリに従ってスナップショットの一つを更新して、前記第三の保存手段の未使用容量を増大させる、コントロール手段と、から構成されることを特徴とするデータ処理の為のストレージシステム。

【請求項３１】
前記更新が、周期的又は未使用容量に基づいた間隔で実施されることを特徴とする請求項３０のストレージシステム。
【発明の詳細な説明】
【技術分野】
【０００１】
本発明はコンピュータストレージに関わり、特に具体的にはデータのバックアップ及びリカバリに関連する。
【背景技術】
【０００２】
データ損失を防止する為に幾つかの方法が慣用的に実施されている。典型的には、データはシステム管理者によって定期的に（例えば毎日一回）バックアップされる。ベリタス社のNetBackup、レガート社のNetworker等の多数のシステムがデータのバックアップとリカバリの為に出回っている。もう一つの技術はボリュームシャドウィングである。本技術では、データがプライマリストレージシステムに書き込まれているときにセカンダリストレージシステムにも書き込まれ、データのミラーイメージが生成される。
【０００３】
ジャーナリングは、データベースシステムで一般に使用されているバックアップとリストアの技術である。バックアップされるデータのイメージが取得される。次いで、データが更新される度に更新ジャーナルが維持される。データのリカバリは、ジャーナルを回復対象データのイメージに適用する事によって行うことができ、任意時点のデータに回復される。オラクル等の典型的なデータベースシステムはジャーナリングを使用することができる。
【発明の開示】
【発明が解決しようとする課題】
【０００４】
然しながら、データベースシステムを除くと、データを任意時点に回復する手段は用意されていない。データベースシステムにしても、ジャーナルを適用するには下記の手順の為に時間が必要である。
・ストレージ（例えばディスク）からジャーナルデータを読み込む。
・ジャーナル内で目的のデータを特定するのに解析が必要である。
・データになされた活動を再生するには、ジャーナルデータを適切なデータイメージに適用しなければならない。このことは、イメージへのアクセスが通常必要でジャーナルが適用される度にデータの書き出しが必要である。
【０００５】
任意時点でのデータ回復は、次のような管理上の要求に応えなければならない。例えば、典型的な要求は、“誤って昨日１０：００ＡＭ頃にファイルを消去してしまった。そのファイルを消去直前に回復しなければならない”といったことである。
【０００６】
データがデータベースシステムに存在しなければ、この種類の要求は都合よく解決されているとは言えない。従って、損失データの回復を容易にするデータ処理方法のニーズが存在する。更にまた、データベースアプリケーション以外のユーザ環境でも、損失データの回復を容易にするデータ処理を可能とするニーズが存在する。
【課題を解決するための手段】
【０００７】
本ストレージシステムが、スナップショット動作とジャーナリングを実行して、損失データを回復させる追加のデータ処理を実行する。スナップショットとジャーナルエントリは、ユーザの生産用データボリュームとは別に保存される。新しいジャーナルエントリを作る為に、より古いジャーナルエントリは除去される。本除去は、一つ以上の古いジャーナルエントリを対応するスナップショットに適用して、本スナップショットを更新するこ

とにより達成される。その後の損失データの回復は、所定のスナップショットにアクセスして、本スナップショットにジャーナルエントリを適用して希望するデータ状態を再生することにより行われる。

【発明の効果】

【０００８】

ストレージシステムが、ユーザがアプリケーションを実行する為のデータストレージサービスを可能にする。

【発明を実施するための最良の形態】

【０００９】

図１は本発明に従うバックアップとリカバリシステムの一実施例を説明する上位階層の一般的ブロックダイアグラムである。システムが立ち上がると、スナップショットが生産用データボリューム（ＤＶＯＬ）１０１に対して取得される。ここで、“スナップショット”とは、本明細書では慣例に従い、ある指定時点に於けるデータボリュームのデータイメージのことを言う。システムに対する要求や実装等に対応して、スナップショットは、データボリュームの全体、又はその或る部分、又は複数の部分に対してなされる。本発明では、システムの正常運用の間にジャーナルエントリがホストから本データボリュームに発せられた全ての書き込み動作に対して作成される。以下に説明するように、所定のスナップショットに対して一連のジャーナルエントリを適用する事により、データは如何なる時点に対しても回復され得る。

【００１０】

図１のバックアップとリカバリシステムには、少なくとも一式のストレージシステム１００が存在する。図示されてはいないものの、本ストレージシステムには適切なプロセッサ、メモリ、及びホスト１１０とそのストレージ媒体（例えばディスク）との間の入出力動作を実行するコントロール回路が存在する事は当業者には明らかである。バックアップとリカバリのシステムには、更に少なくとも一式のホスト１１０が必要である。適切な通信パス１３０がホストとストレージシステムの間に存在する。ホスト１１０には、通常一つ以上のユーザアプリケーション（ＡＰＰ）１１２が稼動している。これらのアプリケーションは本ストレージシステム１００のデータボリューム１０１に存在するストレージ媒体に対してデータの読み書きを実行する。

【００１１】

かくして、アプリケーション１１２とデータボリューム１０１が保護されるべき対象リソースとなる。ユーザのアプリケーションで使用されるデータは、一つあるいはそれ以上のデータボリュームに保存され得るのは明らかである。本発明では、ジャーナルグループ（ＪＮＬＧ）１０２を定義する。データボリューム１０１は本ジャーナルグループの中に編成される。本発明では、ジャーナルグループは、ホスト１１０からデータボリュームへの書き込み動作のジャーナリングが保証されるデータボリュームの最小ユニットである。本関連するジャーナルは、ホストからデータボリュームへの書き込み動作を適切な順序で記録する。ジャーナリング処理によって生成されたジャーナルデータは、一つ以上のジャーナルボリューム（ＪＶＯＬ）１０６に保存される。ホスト１１０には更にリカバリマネージャ（ＲＭ）１１１が存在する。本コンポーネントは、バックアップとリカバリ動作の上位階層での協調を可能にする。本リカバリマネージャに関しては更に後に記す。

【００１２】

ストレージシステム１００はジャーナルグループを構成するデータボリュームのスナップショット（ＳＳ）１０５を生成する。例えば、スナップショット１０５は、本スナップショットが取得された時点のジャーナルグループ１０２内のデータボリューム１０１を反映している。スナップショットイメージを取得する為の慣用的方法は公知である。スナップショットデータを含むストレージシステムの中には、一つ以上のスナップショットボリューム（ＳＶＯＬ）１０７が存在する。一つのスナップショットを一つ以上のスナップショットボリュームに保存することも出来る。本発明の実施例では、ジャーナルデータとスナップショットデータを別々のストレージコンポーネントに保存しているが、これらデー

タを単一のストレージコンポーネントに保存しても良い事は勿論である。

【００１３】

管理テーブル（ＭＴ）１０８はジャーナルグループ１０２、スナップショット１０５、及びジャーナルボリューム１０６に関する情報を保存する。管理テーブルの更なる詳細は図３とこれに関する以下の記述で説明する。コントローラコンポーネント１４０は、書き込み動作のジャーナリングとデータボリュームのスナップショット、及びこれに対応する各異なったストレージコンポーネント１０１、１０６、１０７間のデータ移動を調整する。ここで、コントローラコンポーネントは、ストレージシステム１００内に分散する一つ以上のサブコンポーネントを物理的に含んだものの論理的表現である事に注意願いたい。

【００１４】

図２はジャーナル実装で使用されるデータを示す。書き込み要求がホスト１１０からストレージシステム１００に到達すると、これに対応してジャーナルが生成される。ジャーナルはジャーナルヘッダ２１９とジャーナルデータ２２５で構成される。ジャーナルヘッダ２１９は、対応するジャーナルデータ２２５に関する情報を保持する。ジャーナルデータ２２５は書き込み動作で用いられるデータ（書き込みデータ）を含む。この種類のジャーナルは“事後ジャーナル”とも呼ばれる。ジャーナルヘッダ２１９にはオフセット番号（ＪＨ＿ＯＦＳ）２１１が存在する。本オフセット番号はジャーナルグループ１０２内の本ジャーナルに対応するデータボリューム１０１を指定する。今回の実装では、本データボリュームは、データボリューム０、データボリューム１、データボリューム２・・・と番号付けられる。これに対応して本オフセット番号は０、１、２・・・等となる。

【００１５】

ジャーナルヘッダ２１９内のアドレス（ＪＨ＿ＡＤＲ）２１２フィールドは、（オフセット番号２１１で指定され、）データが書き込まれるデータボリューム内の開始アドレスを保存する。例えば、本アドレスはブロック番号ＬＢＡ（Logical Block Address）で表現される。ジャーナルヘッダ２１９内のデータ長（ＪＨ＿ＬＥＮ）２１３フィールドは、書き込みデータのデータ長を保存する。典型的にはこれはブロック数で表す。ジャーナルヘッダ２１９内の書き込み時刻（ＪＨ＿ＴＩＭＥ）２１４フィールドは、書き込み要求がストレージシステム１００に到着した時刻を保存する。本時刻はカレンダ日、時、分、秒、更にはミリ秒を含む。本時刻はデイスクコントローラ１４０又はホスト１１０により提供される。例えば、メインフレーム環境では複数のメインフレームホストが、タイマを共有し書き込みコマンドを発行する時刻を提供する。

【００１６】

各書き込み要求には順序番号が割り当てられ、ジャーナルヘッダ２１９のフィールド（ＪＨ＿ＳＥＱ）２１５に保存される。ジャーナルグループ１０２内では各順序番号は固有である。本順序番号は、ジャーナルエントリが生成された時に本ジャーナルエントリに対して割り当てられる。ジャーナルヘッダ２１９内のジャーナルボリューム識別子（ＪＨ＿ＪＶＯＬ）２１６は、ジャーナルデータ２２５を保存するジャーナルボリューム１０６を指定する。即ち、本識別子はジャーナルデータを保持しているジャーナルボリュームを示す。ジャーナルデータは、ジャーナルヘッダを保存するジャーナルボリュームとは異なったジャーナルボリュームに保存され得る事に注意願いたい。ジャーナルヘッダ２１９内のジャーナルデータアドレス（ＪＨ＿ＪＡＤＲ）２１７フィールドは、ジャーナルデータを保持している本ジャーナルボリューム１０６内でのジャーナルデータ２２５の開始アドレスを保存する。

【００１７】

図２は、ジャーナルボリューム１０６は、ジャーナルヘッダ領域２１０とジャーナルデータ領域２２０の二つのデータ領域で構成される事を示す。ジャーナルヘッダ領域２１０はジャーナルヘッダ２１９のみを、ジャーナルデータ領域２２０はジャーナルデータ２２５のみを保存する。各ジャーナルヘッダは固定長のデータ構造である。ジャーナルヘッダはジャーナルヘッダ領域の開始点より順番に割り当てられる。本順番はジャーナルエントリの発生順に対応して割り当てられる。後に説明するが、各時点の“最古の”ジャーナル

エントリであるリスト中の最初のジャーナルエントリをポイントするデータが用意される。指定された順序番号（順序番号フィールド２１５内に存在）又は指定された書き込み時刻（時刻フィールド２１４内に存在）に対するジャーナルヘッダ２１９を検出する事が、一般的に必要になる。

【００１８】

ジャーナルヘッダ２１９とジャーナルデータ２２５はジャーナルボリューム１０６の対応領域に発生順に収容されている。かくして、ジャーナルヘッダとジャーナルデータがジャーナルボリュームに保存される順番は、割り当てられた順序番号と同じ順番である。後に説明する様に、本発明の一態様として、本ジャーナル情報２１９、２２５は各対応領域２１０，２２０内で折り返される。

【００１９】

図３は管理テーブル１０８（図１）の詳細を示す。ジャーナルヘッダ領域２１０とジャーナルデータ領域２２０を管理するには、各領域に対するポインタが必要である。既に述べたように、本管理テーブルは、ジャーナルグループ１０２についての攻勢情報及び、ジャーナルグループと本ジャーナルグループに所属するジャーナルボリューム１０６及びスナップショットイメージ１０５間の関係についての構成情報を維持する。図３に示す管理テーブル３００は管理テーブルの一例とその内容を示す。本管理テーブルは、ストレージシステム１００内の特定の一つのジャーナルグループ１０２を指定するジャーナルグループＩＤ（ＧＲＩＤ）３１０を保持する。ジャーナルグループ名（ＧＲＮＡＭＥ）３１１はジャーナルグループを指定する、人が認識できる識別子である。

【００２０】

ジャーナルグループ属性（ＧＲＡＴＴＲ）３１２がジャーナルグループ１０２に対して定義される。本実装例ではマスタ及びリストアの二つの属性が定義される。本マスタ属性はジャーナルグループがジャーナルされていること示す。本リストア属性はジャーナルグループがジャーナルによりリストアされていることを示す。ジャーナル状態（ＧＲＳＴＳ）３１５がジャーナルグループ１０２に対して定義される。これには活性及び非活性の二つの状態がある。管理テーブルには順序カウンタ（ＳＥＱ）３１３を保持するフィールドが存在する。本カウンタはジャーナルヘッダ２１９内で使用される順序番号の源となる。新しいジャーナルが生成される時に、本順序番号３１３が読み込まれ本新ジャーナルに割り当てられる。次いで本順序番号は加算され本管理テーブルに戻される。管理テーブル内の数（ＮＵＭ＿ＤＶＯＬ）３１４は、ジャーナルグループ１０２内に存在するデータボリューム１０１の数を示す。データボリュームリスト（ＤＶＯＬ＿ＬＩＳＴ）３２０はジャーナルグループ内のデータボリュームをリストする。

【００２１】

一実装例では図３に見られる通り、ＤＶＯＬ＿ＬＩＳＴはデータボリューム情報を保持するデータ構造への最初のエントリのポインタである。各データボリューム情報にはオフセット番号（ＤＶＯＬ＿ＯＦＦＳ）３２１が存在する。例えば、ジャーナルグループ１０２が三式のデータボリュームを持つなら、本オフセット値は０、１、２となる。データボリューム識別子（ＤＶＯＬ＿ＩＤ）３２２は全ストレージシステム１００内でデータボリュームを固有に識別する。ポインタ（ＤＶＯＬ＿ＮＥＸＴ）３２４は、ジャーナルグループ内で次のデータボリューム情報を保持しているデータ構造をポイントする。該当する情報が無ければナル値が与えられる。本管理テーブルは、ジャーナルグループ１０２に属するデータ（ジャーナルヘッダとジャーナルデータ）を保持する為に用いられているジャーナルボリューム数を保存する（ＮＵＭ＿ＪＶＯＬ）３３０フィールドを含んでいる。

【００２２】

図２で述べたように、ジャーナルヘッダ領域２１０は、各ジャーナルに対するジャーナルヘッダ２１９を保持する。ジャーナルデータ領域２２０についても同様である。既に述べたように、本発明の一態様として本データ領域２１０、２２０は折り返される。これにより、各データ領域のスペースは有限でもジャーナリングを続ける事が出来る。本管理テーブルは、データ領域２１０、２２０の異なった部分に対するポインタを保存する為のフ

ィールドを持ち、折り返しを可能にする。各フィールドは次のジャーナルエントリの格納場所を指示する。フィールド（ＪＩ＿ＨＥＡＤ＿ＶＯＬ）３３１は、次の新ジャーナルヘッダ２１９を保存するジャーナルヘッダ領域２１０を有するジャーナルボリューム１０６を指定する。フィールド（ＪＩ＿ＨＥＡＤ＿ＡＤＲ）３３２は、次のジャーナルヘッダを保存するジャーナルヘッダ領域の位置についてジャーナルボリューム１０６上のアドレスを指定する。

【００２３】

フィールド（ＪＩ＿ＤＡＴＡ＿ＶＯＬ）３３５は、次のジャーナルデータを保存するジャーナルデータ領域２２０を有するジャーナルボリュームを指定する。フィールド（ＪＩ＿ＤＡＴＡ＿ＡＤＲ）３３６は、次のジャーナルデータを保存するジャーナルボリューム上のアドレスを指定する。かくして、次に書き込まれるジャーナルエントリは、“ＪＩ＿”フィールド３３１、３３２、３３５、３３６の情報で“ポイント”される。管理テーブルは更に、その時の“最古の”ジャーナルエントリを識別するフィールドを有する。

【００２４】

本情報の使用法を以下に記す。フィールド（ＪＯ＿ＨＥＡＤ＿ＶＯＬ）３３３は、最古のジャーナルヘッダ２１９を保持するジャーナルヘッダ領域２１０を保存するジャーナルボリュームを指定する。フィールド（ＪＯ＿ＨＥＡＤ＿ＡＤＲ）３３４は、ジャーナルヘッダ領域内の最古のジャーナルのジャーナルヘッダの位置のアドレスを指定する。フィールド（ＪＯ＿ＤＡＴＡ＿ＶＯＬ）３３７は、最古のジャーナルデータを保持するジャーナルデータ領域２２０を保存するジャーナルボリュームを指定する。本データを保存するジャーナルボリューム内のジャーナルデータ領域のデータアドレスは、フィールド（ＪＯ＿ＤＡＴＡ＿ＡＤＲ）３３８に保存される。管理テーブルには、特定のジャーナルグループ１０２に属するジャーナルボリュームのリスト（ＪＶＯＬ＿ＬＩＳＴ）３４０が存在する。本実装例では、ＪＶＯＬ＿ＬＩＳＴはジャーナルボリューム情報のデータ構造へのポインタである。

【００２５】

図３に示すように、各データ構造は、所定のジャーナルグループ１０２に属する特定のジャーナルボリューム１０６を指定するオフセット番号（ＪＶＯＬ＿ＯＦＦＳ）３４１を持つ。例えば、ジャーナルグループが二式のジャーナルボリューム１０６を持っていたら、各ジャーナルボリュームは０又は１で指定される。ジャーナルボリューム識別子（ＪＶＯＬ＿ＩＤ）３４２は、ストレージシステム１００内でのジャーナルボリュームを固有に指定する。最後にポインタ（ＪＶＯＬ＿ＮＥＸＴ）３４４は、本ジャーナルグループ内の次のジャーナルボリュームに関係する次のデータ構造のエントリをポイントする。存在しなければナル（ＮＵＬＬ）値が指定される。管理テーブルには、ジャーナルグループ１０２に属するスナップショットイメージ１０５のリスト（ＳＳ＿ＬＩＳＴ）３５０が存在する。

【００２６】

本実装例では図３に示すように、ＳＳ＿ＬＩＳＴはスナップショット情報のデータ構造へのポインタである。各スナップショット情報のデータ構造には、スナップショットが取得された時の順序番号（ＳＳ＿ＳＥＱ）３５１が与えられる。既に述べたように本番号は順序カウンタ３１３よりもたらされる。時刻値（ＳＳ＿ＴＩＭＥ）３５２は本スナップショットが取得された時の時刻を示す。状態（ＳＳ＿ＳＴＳ）３５８は各スナップショットの状態（有効及び無効）の値を示す。ポインタ（ＳＳ＿ＮＥＸＴ）３５３は次のスナップショット情報のデータ構造をポイントする。存在しなければナル（ＮＵＬＬ）値が示される。各スナップショット情報のデータ構造は、更にスナップショットイメージ１０５を保存するスナップショットボリューム１０７（図１）のリストを有する。

【００２７】

図３で分かるように、スナップショットボリューム情報のデータ構造へのポインタ（ＳＶＯＬ＿ＬＩＳＴ）３５４が、各スナップショット情報のデータ構造に存在する。各スナップショットボリューム情報のデータ構造には、スナップショットイメージの少なくとも

一部を含む各スナップショットボリュームを指定するオフセット番号（ＳＶＯＬ＿ＯＦＦＳ）３５５が存在する。一式のスナップショットイメージはセグメント化又は分割されて、一つ以上のスナップショットボリュームに保存することが出来る。本実装例では、オフセット値はスナップショットイメージの一部分（セグメント又は分割）を含むｉ番目のスナップショットボリュームを指定する。ｉ番目のスナップショットイメージのセグメントは、ｉ番目のスナップショットボリュームに保存しても良い。各スナップショットボリューム情報のデータ構造には更に、ストレージシステム１００内でのスナップショットボリュームを固有に指定するスナップショットボリューム識別子（ＳＶＯＬ＿ＩＤ）３５６が存在する。ポインタ（ＳＶＯＬ＿ＮＥＸＴ）３５７は、指定されたスナップショットイメージに対する次のスナップショットボリューム情報のデータ構造をポイントする。

【００２８】

図４は本発明の実装例に従う、リカバリマネージャ１１１とストレージシステム１００がバックアッププロセスを開始するプロセスを説明するフローチャートである。スナップショットが取得されていても、ジャーナルエントリが記録されていない書き込み動作は、データ回復処理中にデータは損失するか破損する。従って、本発明では最初のスナップショット取得前にジャーナリングプロセスを起動する。これを行うことにより、スナップショット取得後に発生する全ての書き込み動作がジャーナルされることが保障される。スナップショット完成前に記録されたジャーナルエントリは全て無視してもよい事を注記する。更に本発明に従えば、単一系列の順序番号（ＳＥＱ）３１３が各スナップショットとジャーナルエントリに、生成される毎に割り当てられる。各スナップショットとジャーナルエントリに同一系列の順序番号を割り当てる目的は以下に述べる。

【００２９】

図４に戻って、リカバリマネージャ１１１は、ジャーナルグループ（ＪＮＬＧ）１０２が定義済でなければ、これをステップ４１０で定義する。

【００３０】

図１で示した通り、本定義では、ジャーナルの対象となる一台以上のデータボリューム（ＤＶＯＬ）１０１を指定し、ジャーナル関連情報を保存する一台以上のジャーナルボリューム（ＪＶＯＬ）１０６を指定する。リカバリマネージャは、目的を達成する為に、ストレージシステム１００との間で適切な一連のやり取りを実行する。

【００３１】

ステップ４１５にて、ストレージシステムは図３のテーブル３００の詳細図に示される多様な情報を収容する管理テーブル１０８（図１）を生成する。とりわけ、本プロセスでは、ジャーナルグループ１０２を構成するジャーナルボリュームをリストするＪＶＯＬ＿ＬＩＳＴ３４０の初期化を行う。同様に、データボリュームのリストＤＶＯＬ＿ＬＩＳＴ３２０を生成する。次のジャーナルエントリ（又はテーブルが最初に生成される場合には最初のジャーナルエントリ）を指定するフィールドを初期化する。かくして、ＪＩ＿ＨＥＡＤ＿ＶＯＬ３３１はジャーナルボリュームリスト内の最初を指定し、ＪＩ＿ＨＥＡＤ＿ＡＤＲ３３２は、最初のジャーナルボリュームに存在するジャーナルヘッダ領域２１０の最初のエントリをポイントする。

【００３２】

同様に、ＪＩ＿ＤＡＴＡ＿ＶＯＬ３３５は、ジャーナルボリュームリスト内の最初を指定し、ＪＩ＿ＤＡＴＡ＿ＡＤＲ３３６は、最初のジャーナルボリュームに存在するジャーナルデータ領域２２０の開始点をポイントする。ヘッダ領域２１０とデータ領域２２０は異なるジャーナルボリュームに収容でき、ＪＩ＿ＤＡＴＡ＿ＶＯＬは最初のジャーナルヘッダボリュームとは異なったジャーナルボリュームを指定することがあり得ることに注意していただきたい。ステップ４２０にて、リカバリマネージャ１１１はジャーナリングプロセスを起動する。ジャーナリングを実行する為に、ストレージシステム１００に対して適切な交信がなされる。ステップ４２５にて、ストレージシステムはホスト１１０からの今後の各書き込み動作に対して、ジャーナルエントリ（これは、“事後ジャーナル”とも呼ばれる）を作成する。

【００３３】
図３を参照するに、ジャーナルエントリを作成するには、とりわけ、次のジャーナルエントリを保存するロケーションを指定することが必要になる。フィールドＪＩ＿ＨＥＡＤ＿ＶＯＬ３３１とＪＩ＿ＨＥＡＤ＿ＡＤＲ３３２は各々、次のジャーナルヘッダ２１９のジャーナルボリューム１０６とジャーナルヘッダ領域２１０内での位置を指定する。管理テーブルからの順序カウンタ（ＳＥＱ）３１３は、次のヘッダのＪＨ＿ＳＥＱ２１５フィールドにコピーされ割り当てられる。本順序カウンタは次いで加算され、管理テーブルに戻される。勿論、本順序カウンタは最初に加算し、ついでＪＨ＿ＳＥＱにコピーし、最後に管理テーブルに戻してもよい。

【００３４】
管理テーブルのＪＩ＿ＤＡＴＡ＿ＶＯＬ３３５とＪＩ＿ＤＡＴＡ＿ＡＤＲ３３６フィールドは、各々書き込み動作に付随するデータを保存する、ジャーナルボリュームとジャーナルデータ領域２２０の開始点を指定する。ＪＩ＿ＤＡＴＡ＿ＶＯＬとＪＩ＿ＤＡＴＡ＿ＡＤＲは各々、ジャーナルヘッダのＪＨ＿ＪＶＯＬ２１６とＪＨ＿ＡＤＲ２１２にコピーされ、この結果ジャーナルヘッダがジャーナルデータに対応するポインタを得る事になる。書き込み動作のデータは保存される。

【００３５】
ＪＩ＿ＨＥＡＤ＿ＶＯＬ３３１とＪＩ＿ＨＥＡＤ＿ＡＤＲ３３２フィールドは、次のジャーナルエントリに対する次のジャーナルヘッダ２１９をポイントするべく更新される。このことはジャーナルヘッダ領域２１０内の次に続くジャーナルヘッダエントリを用いる結果となる。同様に、ＪＩ＿ＤＡＴＡ＿ＶＯＬフィールドとＪＩ＿ＤＡＴＡ＿ＡＤＲフィールドは、各々次のジャーナルエントリに対するジャーナルデータボリュームとジャーナルデータ領域での開始点を反映すべく更新される。このことは、ジャーナルデータ領域における次の保存可能位置に進めることを意味する。従ってこれらのフィールドは、ジャーナルエントリのリストをポイントすると見なすことが出来る。リストのジャーナルエントリは、ジャーナルヘッダ領域２１０内のジャーナルヘッダ２１９のシーケンシャル構成によって、互いにリンクすることが出来る。

【００３６】
ジャーナルヘッダ領域２１０の最終点に達したら、次のジャーナルエントリに対するジャーナルヘッダ２１９は、ジャーナルヘッダ領域の開始点に戻される。ジャーナルデータ２２５に対しても同様である。以前のジャーナルエントリに上書きする事を防止する為に、本発明はジャーナルボリューム１０６のエントリを開放する処置を実施する。本件については、以下に述べる。ジャーナル起動後の最初のジャーナルエントリに対しては、ＪＯ＿ＨＥＡＤ＿ＶＯＬ３３３、ＪＯ＿ＨＥＡＤ＿ＡＤＲ３３４、ＪＯ＿ＤＡＴＡ＿ＶＯＬ３３７、及びＪＯ＿ＤＡＴＡ＿ＡＤＲ３３８の各フィールドは、各対応する“ＪＩ＿”フィールドの初期値がセットされる。後に説明するように、“ＪＯ＿”フィールドはその時の最古のジャーナルエントリをポイントする。かくして、新しいジャーナルエントリが作成されると、“ＪＩ＿”フィールドは前進するが“ＪＯ＿”フィールドは前進しない。“ＪＯ＿”フィールドの更新については以下で説明する。

【００３７】
図４のフローチャートに戻って、ジャーナリングプロセスが起動している場合には、ホストからのすべての書き込み動作はジャーナルされる。そこでステップ４３０に於いて、リカバリマネージャ１１１はデータボリューム１０１のスナップショットの取得を起動する。ストレージシステム１００は、リカバリマネージャからスナップショットを取得する指示を受け取る。ステップ４３５にて、ストレージシステムはデータボリュームのスナップショットを取得するプロセスを開始する。この結果特に、管理テーブル（図３）上のＳＳ＿ＬＩＳＴ３５０にアクセスすることになる。次のスナップショットを扱う為に必要な量のメモリがフィールド３５１－３５４に割り当てられる。順序カウンタ（ＳＥＱ）３１３は、上記のＪＨ＿ＳＥＱ２１５で説明したように、ＳＳ＿ＳＥＱ３５１フィールドにコピーされ、加算される。

【００３８】
かくて、順序番号はＳＥＱ３１３より生成され、順序カウンタ中の各番号はジャーナルエントリ又はスナップショットエントリのいずれかに処理中を通して割り当てられる。スナップショットは一台（以上）のスナップショットボリューム（ＳＶＯＬ）１０７に保存される。フィールド３５５－３５７に必要な量のメモリが割り当てられる。スナップショットを保存する為のＳＶＯＬに関連する情報がフィールド３５５－３５７に保存される。スナップショット保存に追加のボリュームが必要なら、フィールド３５５－３５７に追加のメモリが割り当てられる。

【００３９】
図５は、ジャーナルエントリとスナップショットの関係を示す。スナップショット５２０は、ジャーナルグループ１０２に属するデータボリューム１０１の最初のスナップショットイメージを表す。順序番号ＳＥＱ０とＳＥＱ１のジャーナルエントリ（５１０）が作られ、これらは二つの書き込み動作に対するジャーナルエントリを表していることに注意していただきたい。これらのエントリは、ジャーナリングはスナップショットが取得（ステップ４２０）されるに先立って起動されている事を示す。かくして、順序番号ＳＥＱ２に対応する時点で、リカバリマネージャ１１１はスナップショット取得を開始し、更にジャーナル採取は既に起動されているため、スナップショット取得開始後の全ての書き込み動作は、ジャーナルされている。かくして、順序番号ＳＥＱ３以上の書き込み動作は、全てジャーナルされている。見て分かる通り、順序番号ＳＥＱ０とＳＥＱ１のジャーナルエントリは破棄するか無視しなければならない。

【００４０】
データのリカバリでは、指定時点のデータボリューム１０１の少なくとも一部分のデータ状態に回復する事が通常要求される。このことは必要量のジャーナルエントリを本ジャーナルエントリより以前に取得されたスナップショットに適用する事で一般的に実行される。本実施例で開示する方法では、順序番号ＳＥＱ３１３は、ジャーナルエントリ又はスナップショットに割り当てられる度に、加算される。従って、所定のスナップショットに対して適用可能なジャーナルエントリを特定することは単純である。即ち、（ＪＨ＿ＳＥＱ）２１５の順序番号が本スナップショットの順序番号（ＳＳ＿ＳＥＱ）３５１より大きなジャーナルエントリを適用すればよい。

【００４１】
管理者は或る時点、例えば、データボリュームのデータが失われたか又は破損した時（“目標時刻”）以前の時刻を、データ回復の時点に指定することが出来る。目標時刻以前の時刻が見つかるまで、各スナップショットのＳＳ＿ＴＩＭＥ３５２の時刻フィールドが検索される。次に、その時の“最古の”ジャーナルヘッダから始めて、ジャーナルヘッダ領域２１０内のジャーナルヘッダ２１９が検索される。本最古のジャーナルヘッダは、管理テーブルの“ＪＯ＿”フィールド３３３、３３４、３３７及び３３８によって指定されている。ジャーナルヘッダは、順序番号ＪＨ＿ＳＥＱ２１５が対象スナップショットの順序番号ＳＳ＿ＳＥＱ３５１より大きい最初のヘッダを目指して領域２１０内を順番に検索される。選択されたスナップショットには、一時には一ずつ、各ジャーナルエントリを順次適用する事により、順次加算更新され一連の書き込み動作が再現される。本処理は、ジャーナルエントリのＪＨ＿ＴＩＭＥ２１４フィールドの値が目標時刻より以前である限り継続される。本更新作業は時刻フィールド２１４の値が目標時刻を過ぎると停止する。

【００４２】
本発明の一実施例では、単一のスナップショットが取得される。本スナップショットに引き続く全てのジャーナルエントリは、所定時刻におけるデータ状態を再構築する為の適用対象となる。本発明の他の実施例として、複数スナップショットが取得される。本件は図５Ａに示され、複数スナップショット５２０’が取得される。本発明では、各スナップショットとジャーナルエントリは各々が記録される順に順序番号が割り当てられる。各スナップショット５２０’の間には通常、多数のジャーナルエントリ５１０が存在する事に注意願いたい。複数スナップショットを使用する事により、データ回復の時間を短縮する

事が出来る。目標回復時刻に対して最寄りのスナップショットが選択される。本スナップショットに続くジャーナルエントリが所望のデータ回復の為に適用可能である。

【００４３】

図６は本発明のもう一つの態様を説明する。本発明によれば、ジャーナルエントリはホストからの書き込み動作の度に作成され、これはかなり大量のジャーナルエントリ数になる。ジャーナル採取が繰り返されると時間の経過と共に、ジャーナルグループ１０２に対してリカバリマネージャ１１１が定義した各ジャーナルボリューム１０６は、何時かは満杯になってしまう。このままでは、この時点でこれ以上のジャーナルエントリは作成できなくなり、結果として、後続する書き込み動作はジャーナルされず、ジャーナルボリュームが満杯になった以降のデータ状態の回復は不可能になってしまう。

【００４４】

図６は、ストレージシステム１００が、“オーバーフロー”条件の検出を契機に、相応しいスナップショットへジャーナルエントリを適用する状況を示している。ここで、“オーバーフロー”は、ジャーナルボリュームでの未使用容量が設定された限界値を下回った場合に発生したと見做される。オーバーフロー限界値の設定には多様な基準があることに注意願いたい。一つの分かりやすい限界値は、ジャーナルグループに割り当てられたジャーナルボリュームの合計ストレージ容量を基準にするものである。未使用容量が全ストレージ容量のある割合、例えば１０％、に達したら、オーバーフローとする。他の限界値としては各ジャーナルボリューム対応に決めても良い。本発明の一態様では、ジャーナルボリューム内の未使用容量は定期的にモニタされる。反対に未使用容量は非定期的にモニタしても良い。例えば、モニタ間隔はランダムであっても良い。更に他の方法として、例えば、モニタ間隔は未使用容量の関数として変動させても良い。或いは又、ジャーナル採取を契機にモニタしても良い。

【００４５】

図７は、ストレージシステム１００内でオーバーフロー条件の検出の為に行うプロセスを説明する。かくして、ステップ７１０にて、ストレージシステムはジャーナルボリューム１０６の全未使用容量を定期的に、例えば１０秒毎にチェックする。未使用容量は、管理テーブル３００内のポインタ（ＪＩ＿ＨＥＡＤ＿ＶＯＬ３３１、ＪＩ＿ＨＥＡＤ＿ＡＤＲ３３２等）がジャーナルボリュームで消費しているストレージの現在の状態を維持しているので、容易に計算可能である。未使用容量が限界値以上であれば、次の周期を待って未使用容量のチェックを繰り返すのみである。未使用容量が前もって決定した限界値以下になったら、ステップ７２０にて、幾つかの、特に最古のジャーナルエントリがスナップショットに適用され、スナップショットを更新する。

【００４６】

図３を参照するに、本最古のジャーナルエントリのジャーナルヘッダ２１９は、ＪＯ＿ＨＥＡＤ＿ＶＯＬフィールド３３３とＪＯ＿ＨＥＡＤ＿ＡＤＲフィールド３３４で指定される。これらのフィールドは、本最古のジャーナルエントリを保存するジャーナルボリュームとジャーナルヘッダ領域２１０に対する本ジャーナルボリューム内の位置を指定する。同様に、本最古のジャーナルエントリのジャーナルデータは、ＪＯ＿ＤＡＴＡ＿ＶＯＬフィールド３３７とＪＯ＿ＤＡＴＡ＿ＡＤＲフィールド３３８で指定される。これらのフィールドで指定されるジャーナルエントリはスナップショットに適用される。ここで選択されるスナップショットは、ジャーナルエントリより以前でかつ本ジャーナルエントリに順序番号が最も近いスナップショットである。

【００４７】

かくして、本実装例では、順序番号が加算される度に、ジャーナルエントリに対して最も近くてより小さい（時刻的にはより早い）順序番号を持つスナップショットが選択される。ジャーナルエントリがスナップショットに適用されてスナップショットが更新されると、適用済のジャーナルエントリは開放される。このことは単に、ＪＯ＿ＨＥＡＤ＿ＶＯＬフィールド３３３、ＪＯ＿ＨＥＡＤ＿ＡＤＲフィールド３３４、ＪＯ＿ＤＡＴＡ＿ＶＯＬフィールド３３７、及びＪＯ＿ＤＡＴＡ＿ＡＤＲフィールド３３８を次のジャーナルエ

ントリをポイントすべく更新するのみである。順序番号は最終的には折り返され０から再度スタートすることは当業者には明らかである。順序番号を比較する時に、本再スタートを把握するための相応しいメカニズムも当業者の技術範囲である。

【００４８】

図７に戻って、スナップショットにジャーナルエントリを適用してスナップショットを更新した後には、この結果として適用済ジャーナルエントリが開放され（ステップ７３０）て、ジャーナルボリューム内の未使用容量が増大した事をチェックする。本未使用容量は、更にステップ７３０にて限界値と比較される。この場合第二の異なった限界値が使用できる。例えば、本プロセスを終了するには開始する時よりより高いレベルの未使用容量を必要としても良い。

【００４９】

これにより本プロセスが過剰に使用される事を防止する事が可能になるが、一旦開始されたら、第二の高いレベルの限界値により合理的な範囲で出来るだけ多くの未使用容量を確保して回復を促進する。これらの限界値を決めるには管理者の時間をかけた経験が要求される事に注意願いたい。かくして、ステップ７３０により未使用容量が第二の限界値を超え停止条件に達したら、本プロセスは終了する。そうでなければ、ステップ７２０に戻り次に古いジャーナルエントリに対して繰り返される。ステップ７３０と７２０が、未使用容量がステップ７３０にて本限界値に達するまで繰り返される。

【００５０】

図７Ａは、図７のステップ７２０に対する代替の実施例のサブステップを示す。ステップ７２０では、ジャーナルエントリを、本ジャーナルエントリより以前の最新のスナップショットに適用する事によって、開放する。然しながら、複数のスナップショットが使用可能なら、スナップショットを更新する為にジャーナルエントリを適用するプロセスで消費する時間を省略する事が出来る。

【００５１】

図７Ａは、図７のステップ７２０に対する代替ステップ７２０’の詳細を示す。ステップ７２１にて、本最古のジャーナルエントリより新しいスナップショットが存在するかどうかを判定する。本判定は、順序番号が本最古のジャーナルエントリより大きい最初のスナップショットを検索する事により、可能である。代わって本判定は、その時の最古のジャーナルエントリより所定時間（例えば少なくとも１時間）だけ新しいスナップショットを見つけてこれを基準にしても良い。更に又他の方法として、スナップショットとジャーナルエントリの時間差に代わって、順序番号を用いても良い。例えば、最古のジャーナルエントリの順序番号より、Ｎ分だけ進んだ順序番号を持つスナップショットを選択しても良い。

【００５２】

該当するスナップショットがステップ７２１で見つかったら、これより古いジャーナルエントリはスナップショットに適用しないまま除去しても良い。かくして、ステップ７２２にて、“ＪＯ＿”フィールド（ＪＯ＿ＨＥＡＤ＿ＶＯＬ３３３，ＪＯ＿ＨＥＡＤ＿ＡＤＲ３３４，ＪＯ＿ＤＡＴＡ＿ＶＯＬ３３７，及びＪＯ＿ＤＡＴＡ＿ＡＤＲ３３８）は、選択されたスナップショットより新しい最初のジャーナルエントリをポイントする値に単純に書き換えられる。該当するスナップショットが見つからなかったら、ステップ７２０で議論したように、ステップ７２３にて、残った最古のジャーナルエントリがこれより古いスナップショットに適用される。

【００５３】

ステップ７２１に対する更なる代替案として、単純に最も新しいスナップショットを選択する案がある。本スナップショットより小さな順序番号を持つジャーナルエントリは開放可能である。このことはさらに単純に、最新のスナップショットより大きな順序番号を持つ最初のジャーナルエントリをポイントするように、“ＪＯ＿”フィールドを更新するのみである。ここで、本発明の一態様はデータを希望する時点の状態に回復する事が出来る事を思い出してほしい。このことは、生成されたジャーナルエントリを必要範囲は保存

して、本ジャーナルエントリを所定のスナップショットに適用して、書き込み動作を再現する事により達成される。この最後の実施例は大量のジャーナルエントリを除去する事が出来るが、これによってデータ状態が回復できる時間帯を制限する結果になってしまう。然しながら、構成によっては、大量のジャーナルエントリを除去する事は与えられた動作環境にとっては望ましい事もあり得る。

【００５４】

これまで述べたステップは全体として、コントローラ１４０（例えばデイスクコントローラ）で実現される事に注意願いたい。このことは、特定の実装次第で純粋のソフトウエア、カスタムロジック、ソフトウエアとハードウエアの相応しい組合せで可能である。もっと一般的に言えば、前記の実現は典型的に、ソフトウエアとハードウエアの組合せでなされると言ってよい。当業者には、最終的に採用する技術はシステムコスト、システム性能、慣用的ソフトウエアとハードウエア、動作環境その他を含めた要素等によって決まる事、しかしこれらに限定、制約されるものでない事を容易に理解してもらえるであろう。これまで述べられ意図された実施例は、当業者により過大な実験を行う事無く具体的実装に直ちに適用可能である。

【図面の簡単な説明】

【００５５】

本発明の態様、効果と新規な機能が、本発明の以下の付随図面と共になされる説明によって明らかになる。

【図１】は、本発明の一実施例を説明する上位階層の一般的ブロックダイアグラムである。

【図２】は、本発明に従うジャーナルエントリを保存するデータ構造の一実施例を一般的に説明する。

【図３】は、本発明に従うスナップショットボリュームとジャーナルエントリボリュームを管理するためのデータ構造の一実施例を一般的に説明する。

【図４】は、リカバリマネージャとストレージシステム内のコントローラとの間での処理を説明する上位階層のフローチャートである。

【図５】は、一つのスナップショットと複数のジャーナルエントリ間の関係を説明する。

【図５Ａ】は、複数のスナップショットと複数のジャーナルエントリ間の関係を説明する。

【図６】は、オーバーフロー状態が発生した場合のデータフローを説明する上位階層の図である。

【図７】は、オーバーフロー状態を処理する為のストレージシステム内のコントローラ動作を説明する上位階層のフローチャートである。

【図７Ａ】は、図７で示す処理ステップの別案を示す。

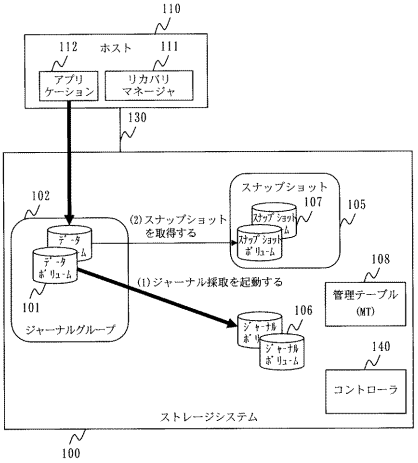
【符号の説明】

【００５６】

１００・・・ストレージシステム
１０１・・・データボリューム（ＤＶＯＬ）
１０５・・・スナップショット（ＳＳ）
１０６・・・ジャーナルボリューム（ＪＶＯＬ）
１０７・・・スナップショットボリューム（ＳＶＯＬ）
１０８・・・管理テーブル（ＭＴ）
１０２・・・ジャーナルグループ（ＪＮＬＧ）
１１０・・・ホスト
１１１・・・リカバリマネージャ（ＲＭ）
１１２・・・アプリケーション（ＡＰＰ）
１４０・・・コントローラ

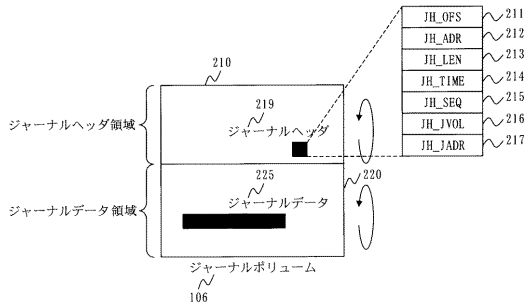
【図１】

図１　バックアップの概要



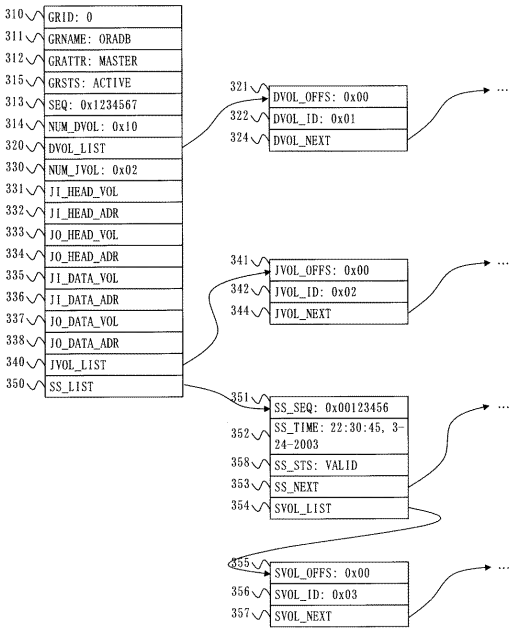
【図２】

図２　ジャーナル用のコントロールデータ



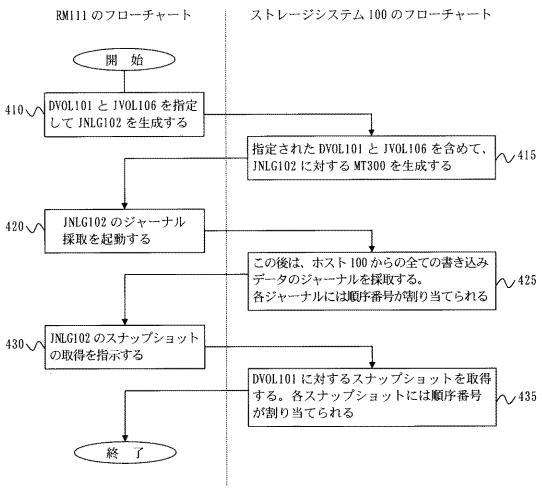
【図３】

図３　ＭＴ（管理テーブル）３００



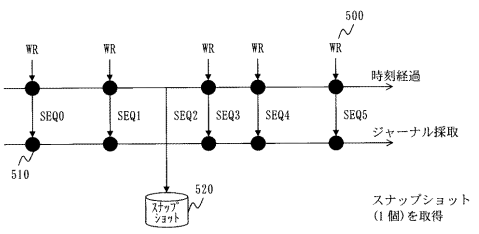
【図４】

図４　ジャーナルの起動



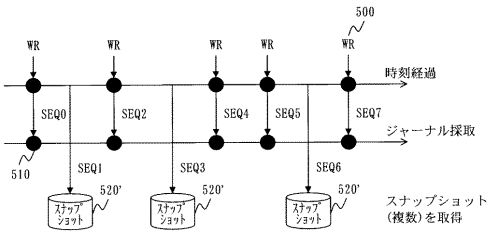
【図５】

図５　スナップショットとジャーナルの関係



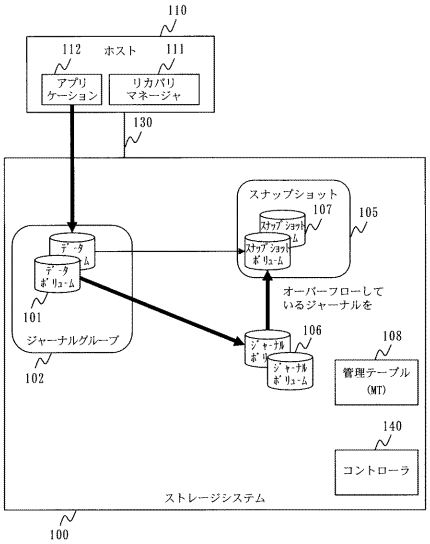
【図５Ａ】

図５Ａ　複数スナップショットとジャーナルの関係



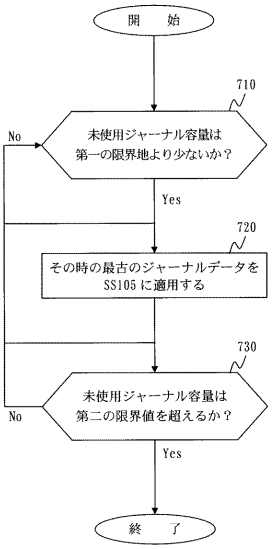
【図６】

図６　ジャーナルのオーバーフロー



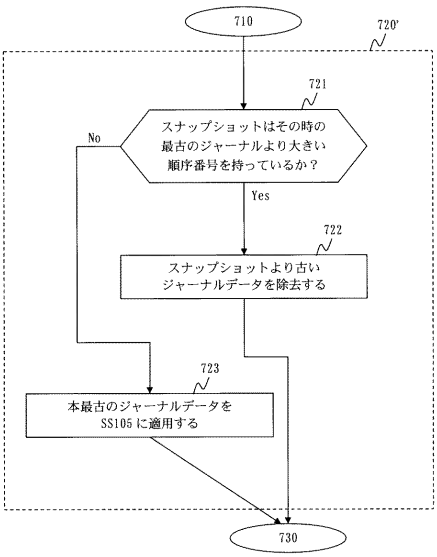
【図７】

図７　オーバフロージャーナル



【図７Ａ】

図７Ａ



フロントページの続き

【要約の続き】
てることにより、スナップショットに適用すべきジャーナルを見つける事が容易になる。ジャーナルスペースが枯渇するオーバーフロー条件を検出し、ジャーナルスペースを回復する技術が述べられる。
【選択図】図１

【外国語明細書】

PATENT

Attorney Docket No.: 16869B-082700US
Client ref. No.: HAL 279

METHOD AND APPARATUS FOR BACKUP AND RECOVERY USING STORAGE BASED JOURNALING

CROSS-REFERENCES TO RELATED APPLICATIONS

[01] This application is related to the following commonly owned and co-pending U.S. applications:

"Method and Apparatus for Data Recovery Using Storage Based Journaling," Attorney Docket Number 16869B-082800US, and

"Method and Apparatus for Synchronizing Applications for Data Recovery Using Storage Based Journaling," Attorney Docket Number 16869B-082900US,

both of which are herein incorporated by reference for all purposes.

BACKGROUND OF THE INVENTION

[02] The present invention is related to computer storage and in particular to backup and recovery of data.

[03] Several methods are conventionally used to prevent the loss of data. Typically, data is backed up in a periodic manner (e.g., once a day) by a system administrator. Many systems are commercially available which provide backup and recovery of data; e.g., Veritas NetBackup, Legato/Networker, and so on. Another technique is known as volume shadowing. This technique produces a mirror image of data onto a secondary storage system as it is being written to the primary storage system.

[04] Journaling is a backup and restore technique commonly used in database systems. An image of the data to be backed up is taken. Then, as changes are made to the data, a journal of the changes is maintained. Recovery of data is accomplished by applying the journal to an appropriate image to recover data at any point in time. Typical database systems, such as Oracle, can perform journaling.

[05] Except for database systems, however, there are no ways to recover data at any point in time. Even for database systems, applying a journal takes time since the procedure includes:

- reading the journal data from storage (e.g., disk)
- the journal must be analyzed to determine at where in the journal the desired data can be found

- apply the journal data to a suitable image of the data to reproduce the activities performed on the data - this usually involves accessing the image, and writing out data as the journal is applied

[06] Recovering data at any point in time addresses the following types of administrative requirements. For example, a typical request might be, "I deleted a file by mistake at around 10:00 am yesterday. I have to recover the file just before it was deleted."

[07] If the data is not in a database system, this kind of request cannot be conveniently, if at all, serviced. A need therefore exists for processing data in a manner that facilitates recovery of lost data. A need exists for being able to provide data processing that facilitates data recovery in user environments other than in a database application.

SUMMARY OF THE INVENTION

[08] A storage system provides data storage services for users and their applications. The storage system performs additional data processing to provide for recovery of lost data, including performing snapshot operations and journaling. Snapshots and journal entries are stored separately from the production data volumes provided for the users. Older journal entries are cleared in order to make for new journal entries. This involves updating a snapshot by applying one or more of the older journal entries to an appropriate snapshot. Subsequent recovery of lost data can be provided by accessing an appropriate snapshot and applying journal entries to the snapshot to reproduce the desired data state.

BRIEF DESCRIPTION OF THE DRAWINGS

[09] Aspects, advantages and novel features of the present invention will become apparent from the following description of the invention presented in conjunction with the accompanying drawings:

Fig. 1 is a high level generalized block diagram of an illustrative embodiment of the present invention;

Fig. 2 is a generalized illustration of a illustrative embodiment of a data structure for storing journal entries in accordance with the present invention;

Fig. 3 is a generalized illustration of an illustrative embodiment of a data structure for managing the snapshot volumes and the journal entry volumes in accordance with the present invention;

Fig. 4 is a high level flow diagram highlighting the processing between the recovery manager and the controller in the storage system;

Fig. 5 illustrates the relationship between a snapshot and a plurality of journal entries;

Fig. 5A illustrates the relationship among a plurality of snapshots and a plurality of journal entries;

Fig. 6 is a high level illustration of the data flow when an overflow condition arises;

Fig. 7 is a high level flow chart highlighting an aspect of the controller in the storage system to handle an overflow condition; and

Fig. 7A illustrates an alternative to a processing step shown in Fig. 7.

DESCRIPTION OF THE SPECIFIC EMBODIMENTS

[10] Fig. 1 is a high level generalized block diagram of an illustrative embodiment of a backup and recovery system according to the present invention. When the system is activated, a snapshot is taken for production data volumes (DVOL) 101. The term "snapshot" in this context conventionally refers to a data image of at the data volume at a given point in time. Depending on system requirements, implementation, and so on, the snapshot can be of the entire data volume, or some portion or portions of the data volume(s). During the normal course of operation of the system in accordance with the invention, a journal entry is made for every write operation issued from the host to the data volumes. As will be discussed below, by applying a series of journal entries to an appropriate snapshot, data can be recovered at any point in time.

[11] The backup and recovery system shown in Fig. 1 includes at least one storage system 100. Though not shown, one of ordinary skill can appreciate that the storage system includes suitable processor(s), memory, and control circuitry to perform IO between a host 110 and its storage media (e.g., disks). The backup and recovery system also requires at least one host 110. A suitable communication path 130 is provided between the host and the storage system.

[12] The host 110 typically will have one or more user applications (APP) 112 executing on it. These applications will read and/or write data to storage media contained in the data volumes 101 of storage system 100. Thus, applications 112 and the data volumes 101 represent the target resources to be protected. It can be appreciated that data used by the user applications can be stored in one or more data volumes.

[13] In accordance with the invention, a journal group (JNLG) 102 is defined. The data volumes 101 are organized into the journal group. In accordance with the present invention,

a journal group is the smallest unit of data volumes where journaling of the write operations from the host 110 to the data volumes is guaranteed. The associated journal records the order of write operations from the host to the data volumes in proper sequence. The journal data produced by the journaling activity can be stored in one or more journal volumes(JVOL) 106.

[14] The host 110 also includes a recovery manager (RM) 111. This component provides a high level coordination of the backup and recovery operations. Additional discussion about the recovery manager will be discussed below.

[15] The storage system 100 provides a snapshot (SS) 105 of the data volumes comprising a journal group. For example, the snapshot 105 is representative of the data volumes 101 in the journal group 106 at the point in time that the snapshot was taken. Conventional methods are known for producing the snapshot image. One or more snapshot volumes (SVOL) 107 are provided in the storage system which contain the snapshot data. A snapshot can be contained in one or more snapshot volumes. Though the disclosed embodiment illustrates separate storage components for the journal data and the snapshot data, it can be appreciated that other implementations can provide a single storage component for storing the journal data and the snapshot data.

[16] A management table (MT) 108 is provided to store the information relating to the journal group 102, the snapshot 105, and the journal volume(s) 106. Fig. 3 and the accompanying discussion below reveal additional detail about the management table.

[17] A controller component 140 is also provided which coordinates the journaling of write operations and snapshots of the data volumes, and the corresponding movement of data among the different storage components 101, 106, 107. It can be appreciated that the controller component is a logical representation of a physical implementation which may comprise one or more sub-components distributed within the storage system 100.

[18] Fig. 2 shows the data used in an implementation of the journal. When a write request from the host 110 arrives at the storage system 100, a journal is generated in response. The journal comprises a Journal Header 219 and Journal Data 225. The Journal Header 219 contains information about its corresponding Journal Data 225. The Journal Data 225 comprises the data (write data) that is the subject of the write operation. This kind of journal is also referred to as an "AFTER journal."

[19] The Journal Header 219 comprises an offset number (JH_OFS) 211. The offset number identifies a particular data volume 101 in the journal group 102. In this particular implementation, the data volumes are ordered as the 0^{th} data volume, the 1^{st} data volume, the 2^{nd} data volume and so on. The offset numbers might be 0, 1, 2, etc.

[20] A starting address in the data volume (identified by the offset number 211) to which the write data is to be written is stored to a field in the Journal Header 219 to contain an address (JH_ADR) 212. For example, the address can be represented as a block number (LBA, Logical Block Address).

[21] A field in the Journal Header 219 stores a data length (JH_LEN) 213, which represents the data length of the write data. Typically it is represented as a number of blocks.

[22] A field in the Journal Header 219 stores the write time (JH_TIME) 214, which represents the time when the write request arrives at the storage system 100. The write time can include the calendar date, hours, minutes, seconds and even milliseconds. This time can be provided by the disk controller 140 or by the host 110. For example, in a mainframe computing environment, two or more mainframe hosts share a timer and can provide the time when a write command is issued.

[23] A sequence number(JH_SEQ) 215 is assigned to each write request. The sequence number is stored in a field in the Journal Header 219. Every sequence number within a given journal group 102 is unique. The sequence number is assigned to a journal entry when it is created.

[24] A journal volume identifier (JH_JVOL) 216 is also stored in the Journal Header 219. The volume identifier identifies the journal volume 106 associated with the Journal Data 225. The identifier is indicative of the journal volume containing the Journal Data. It is noted that the Journal Data can be stored in a journal volume that is different from the journal volume which contains the Journal Header.

[25] A journal data address (JH_JADR) 217 stored in the Journal Header 219 contains the beginning address of the Journal Data 225 in the associated journal volume 106 that contains the Journal Data.

[26] Fig. 2 shows that the journal volume 106 comprises two data areas: a Journal Header Area 210 and a Journal Data Area 220. The Journal Header Area 210 contains only Journal Headers 219, and Journal Data Area 220 contains only Journal Data 225. The Journal Header is a fixed size data structure. A Journal Header is allocated sequentially from the beginning of the Journal Header Area. This sequential organization corresponds to the chronological order of the journal entries. As will be discussed, data is provided that points to the first journal entry in the list, which represents the "oldest" journal entry. It is typically necessary to find the Journal Header 219 for a given sequence number (as stored in the sequence number field 215) or for a given write time (as stored in the time field 214).

[27] Journal Header 219 and Journal Data 225 are contained in chronological order in their respective areas in the journal volume 106. Thus, the order in which the Journal Header and the Journal Data are stored in the journal volume is the same order as the assigned sequence number. As will be discussed below, an aspect of the present invention is that the journal information 219, 225 wrap within their respective areas 210, 220.

[28] Fig. 3 shows detail about the management table 108 (Fig. 1). In order to manage the Journal Header Area 210 and Journal Data Area 220, pointers for each area are needed. As mentioned above, the management table maintains configuration information about a journal group 102 and the relationship between the journal group and its associated journal volume(s) 106 and snapshot image 105.

[29] The management table 300 shown in Fig. 3 illustrates an example management table and its contents. The management table stores a journal group ID (GRID) 310 which identifies a particular journal group 102 in a storage system 100. A journal group name (GRNAME) 311 can also be provided to identify the journal group with a human recognizable identifier.

[30] A journal attribute (GRATTR) 312 is associated with the journal group 102. In accordance with this particular implementation, two attributes are defined: MASTER and RESTORE. The MASTER attribute indicates the journal group is being journaled. The RESTORE attribute indicates that the journal group is being restored from a journal.

[31] A journal status (GRSTS) 315 is associated with the journal group 102. There are two statuses: ACTIVE and INACTIVE.

[32] The management table includes a field to hold a sequence counter (SEQ) 313. This counter serves as the source of sequence numbers used in the Journal Header 219. When creating a new journal, the sequence number 313 is read and assigned to the new journal. Then, the sequence number is incremented and written back into the management table.

[33] The number (NUM_DVOL) 314 of data volumes 101 contained in a give journal group 102 is stored in the management table.

[34] A data volume list (DVOL_LIST) 320 lists the data volumes in a journal group. In a particular implementation, DVOL_LIST is a pointer to the first entry of a data structure which holds the data volume information. This can be seen in Fig. 3. Each data volume information comprises an offset number (DVOL_OFFS) 321. For example, if the journal group 102 comprises three data volumes, the offset values could be 0, 1 and 2. A data volume identifier (DVOL_ID) 322 uniquely identifies a data volume within the entire storage

system 100. A pointer (DVOL_NEXT) 324 points to the data structure holding information for the next data volume in the journal group; it is a NULL value otherwise.

[35] The management table includes a field to store the number of journal volumes (NUM_JVOL) 330 that are being used to contain the data (journal header and journal data) associated with a journal group 102.

[36] As described in Fig. 2, the Journal Header Area 210 contains the Journal Headers 219 for each journal; likewise for the Journal Data components 225. As mentioned above, an aspect of the invention is that the data areas 210, 220 wrap. This allows for journaling to continue despite the fact that there is limited space in each data area.

[37] The management table includes fields to store pointers to different parts of the data areas 210, 220 to facilitate wrapping. Fields are provided to identify where the next journal entry is to be stored. A field (JI_HEAD_VOL) 331 identifies the journal volume 106 that contains the Journal Header Area 210 which will store the next new Journal Header 219. A field (JI_HEAD_ADR) 332 identifies an address on the journal volume of the location in the Journal Header Area where the next Journal Header will be stored. The journal volume that contains the Journal Data Area 220 into which the journal data will be stored is identified by information in a field (JI_DATA_VOL) 335. A field (JI_DATA_ADR) 336 identifies the specific address in the Journal Data Area where the data will be stored. Thus, the next journal entry to be written is "pointed" to by the information contained in the "JI_" fields 331, 332, 335, 336.

[38] The management table also includes fields which identify the "oldest" journal entry. The use of this information will be described below. A field (JO_HEAD_VOL) 333 identifies the journal volume which stores the Journal Header Area 210 that contains the oldest Journal Header 219. A field (JO_HEAD_ADR) 334 identifies the address within the Journal Header Area of the location of the journal header of the oldest journal. A field (JO_DATA_VOL) 337 identifies the journal volume which stores the Journal Data Area 220 that contains the data of the oldest journal. The location of the data in the Journal Data Area is stored in a field (JO_DATA_ADR) 338.

[39] The management table includes a list of journal volumes (JVOL_LIST) 340 associated with a particular journal group 102. In a particular implementation, JVOL_LIST is a pointer to a data structure of information for journal volumes. As can be seen in Fig. 3, each data structure comprises an offset number (JVOL_OFS) 341 which identifies a particular journal volume 106 associated with a given journal group 102. For example, if a journal group is associated with two journal volumes 106, then each journal volume might be

identified by a 0 or a 1. A journal volume identifier (JVOL_ID) 342 uniquely identifies the journal volume within the storage system 100. Finally, a pointer (JVOL_NEXT) 344 points to the next data structure entry pertaining to the next journal volume associated with the journal group; it is a NULL value otherwise.

[40] The management table includes a list (SS_LIST) 350 of snapshot images 105 associated with a given journal group 102. In this particular implementation, SS_LIST is a pointer to snapshot information data structures, as indicated in Fig. 3. Each snapshot information data structure includes a sequence number (SS_SEQ) 351 that is assigned when the snapshot is taken. As discussed above, the number comes from the sequence counter 313. A time value (SS_TIME) 352 indicates the time when the snapshot was taken. A status (SS_STS) 358 is associated with each snapshot; valid values include VALID and INVALID. A pointer (SS_NEXT) 353 points to the next snapshot information data structure; it is a NULL value otherwise.

[41] Each snapshot information data structure also includes a list of snapshot volumes 107 (Fig. 1) used to store the snapshot images 105. As can be seen in Fig. 3, a pointer (SVOL_LIST) 354 to a snapshot volume information data structure is stored in each snapshot information data structure. Each snapshot volume information data structure includes an offset number (SVOL_OFFS) 355 which identifies a snapshot volume that contains at least a portion of the snapshot image. It is possible that a snapshot image will be segmented or otherwise partitioned and stored in more than one snapshot volume. In this particular implementation, the offset identifies the i^{th} snapshot volume which contains a portion (segment, partition, etc) of the snapshot image. In one implementation, the i^{th} segment of the snapshot image might be stored in the i^{th} snapshot volume. Each snapshot volume information data structure further includes a snapshot volume identifier (SVOL_ID) 356 that uniquely identifies the snapshot volume in the storage system 100. A pointer (SVOL_NEXT) 357 points to the next snapshot volume information data structure for a given snapshot image.

[42] Fig 4 shows a flowchart highlighting the processing performed by the recovery manager 111 and Storage System 100 to initiate backup processing in accordance with the illustrative embodiment of the invention as shown in the figures. If journal entries are not recorded during the taking of a snapshot, the write operations corresponding to those journal entries would be lost and data corruption could occur during a data restoration operation. Thus, in accordance with an aspect of the invention, the journaling process is started prior to taking the first snapshot. Doing this ensures that any write operations which occur during the

taking of a snapshot are journaled. As a note, any journal entries recorded prior to the completion of the snapshot can be ignored.

[43] Further in accordance with the invention, a single sequence of numbers (SEQ) 313 are associated with each of one or more snapshots and journal entries, as they are created. The purpose of associating the same sequence of numbers to both the snapshots and the journal entries will be discussed below.

[44] Continuing with Fig. 4, the recovery manager 111 might define, in a step 410, a journal group (JNLG) 102 if one has not already been defined. As indicated in Fig. 1, this may include identifying one or data volumes (DVOL) 101 for which journaling is performed, and identifying one or journal volumes (JVOL) 106 which are used to store the journal-related information. The recovery manager performs a suitable sequence of interactions with the storage system 100 to accomplish this. In a step 415, the storage system may create a management table 108 (Fig. 1), incorporating the various information shown in the table detail 300 illustrated in Fig. 3. Among other things, the process includes initializing the JVOL_LIST 340 to list the journal volumes which comprise the journal group 102 Likewise, the list of data volumes DVOL_LIST 320 is created. The fields which identify the next journal entry (or in this case where the table is first created, the first journal entry) are initialized. Thus, JI_HEAD_VOL 331 might identify the first in the list of journal volumes and JI_HEAD_ADR 332 might point to the first entry in the Journal Header Area 210 located in the first journal volume. Likewise, JI_DATA_VOL 335 might identify the first in the list of journal volumes and JI_DATA_ADR 336 might point to the beginning of the Journal Data Area 220 in the first journal volume. Note, that the header and the data areas 210, 220 may reside on different journal volumes, so JI_DATA_VOL might identify a journal volume different from the first journal volume.

[45] In a step 420, the recovery manager 111 will initiate the journaling process. Suitable communication(s) are made to the storage system 100 to perform journaling. In a step 425, the storage system will make a journal entry (also referred to as an "AFTER journal") for each write operation that issues from the host 110.

[46] With reference to Fig. 3, making a journal entry includes, among other things, identifying the location for the next journal entry. The fields JI_HEAD_VOL 331 and JI_HEAD_ADR 332 identify the journal volume 106 and the location in the Journal Header Area 210 of the next Journal Header 219. The sequence counter (SEQ) 313 from the management table is copied to (associated with) the JH_SEQ 215 field of the next header. The sequence counter is then incremented and stored back to the management table. Of

course, the sequence counter can be incremented first, copied to JH_SEQ, and then stored back to the management table.

[47] The fields JI_DATA_VOL 335 and in the management table identify the journal volume and the beginning of the Journal Data Area 220 for storing the data associated with the write operation. The JI_DATA_VOL and JI_DATA_ADR fields are copied to JH_JVOL 216 and to JH_ADR 212, respectively, of the Journal Header, thus providing the Journal Header with a pointer to its corresponding Journal Data. The data of the write operation is stored.

[48] The JI_HEAD_VOL 331 and JI_HEAD_ADR 332 fields are updated to point to the next Journal Header 219 for the next journal entry. This involves taking the next contiguous Journal Header entry in the Journal Header Area 210. Likewise, the JI_DATA_ADR field (and perhaps JI_DATA_VOL field) is updated to reflect the beginning of the Journal Data Area for the next journal entry. This involves advancing to the next available location in the Journal Data Area. These fields therefore can be viewed as pointing to a list of journal entries. Journal entries in the list are linked together by virtue of the sequential organization of the Journal Headers 219 in the Journal Header Area 210.

[49] When the end of the Journal Header Area 210 is reached, the Journal Header 219 for the next journal entry wraps to the beginning of the Journal Header Area. Similarly for the Journal Data 225. To prevent overwriting earlier journal entries, the present invention provides for a procedure to free up entries in the journal volume 106. This aspect of the invention is discussed below.

[50] For the very first journal entry, the JO_HEAD_VOL field 333, JO_HEAD_ADR field 334, JO_DATA_VOL field 337, and the JO_DATA_ADR field 338 are set to contain their contents of their corresponding "JI_" fields. As will be explained the "JO_" fields point to the oldest journal entry. Thus, as new journal entries are made, the "JO_" fields do not advance while the "JI_" fields do advance. Update of the "JO_" fields is discussed below.

[51] Continuing with the flowchart of Fig. 4, when the journaling process has been initiated, all write operations issuing from the host are journaled. Then in a step 430, the recovery manager 111 will initiate taking a snapshot of the data volumes 101. The storage system 100 receives an indication from the recovery manager to take a snapshot. In a step 435, the storage system performs the process of taking a snapshot of the data volumes. Among other things, this includes accessing SS_LIST 350 from the management table (Fig. 3). A suitable amount of memory is allocated for fields 351 - 354 to represent the next snapshot. The sequence counter (SEQ) 313 is copied to the field SS_SEQ 351 and

incremented, in the manner discussed above for JH_SEQ 215. Thus, over time, a sequence of numbers is produced from SEQ 313, each number in the sequence being assigned either to a journal entry or a snapshot entry.

[52] The snapshot is stored in one (or more) snapshot volumes (SVOL) 107. A suitable amount of memory is allocated for fields 355 - 357. The information relating to the SVOLs for storing the snapshot are then stored into the fields 355 - 357. If additional volumes are required to store the snapshot, then additional memory is allocated for fields 355 - 357.

[53] Fig. 5 illustrates the relationship between journal entries and snapshots. The snapshot 520 represents the first snapshot image of the data volumes 101 belonging to a journal group 102. Note that journal entries (510) having sequence numbers SEQ0 and SEQ1 have been made, and represent journal entries for two write operations. These entries show that journaling has been initiated at a time prior to the snapshot being taken (step 420). Thus, at a time corresponding to the sequence number SEQ2, the recovery manager 111 initiates the taking of a snapshot, and since journaling has been initiated, any write operations occurring during the taking of the snapshot are journaled. Thus, the write operations 500 associated with the sequence numbers SEQ3 and higher show that those operations are being journaled. As an observation, the journal entries identified by sequence numbers SEQ0 and SEQ1 can be discarded or otherwise ignored.

[54] Recovering data typically requires recover the data state of at least a portion of the data volumes 101 at a specific time. Generally, this is accomplished by applying one or more journal entries to a snapshot that was taken earlier in time relative to the journal entries. In the disclosed illustrative embodiment, the sequence number SEQ 313 is incremented each time it is assigned to a journal entry or to a snapshot. Therefore, it is a simple matter to identify which journal entries can be applied to a selected snapshot; i.e., those journal entries whose associated sequence numbers (JH_SEQ, 215) are greater than the sequence number (SS_SEQ, 351) associated with the selected snapshot.

[55] For example, the administrator may specify some point in time, presumably a time that is earlier than the time (the "target time") at which the data in the data volume was lost or otherwise corrupted. The time field SS_TIME 352 for each snapshot is searched until a time earlier than the target time is found. Next, the Journal Headers 219 in the Journal Header Area 210 is searched, beginning from the "oldest" Journal Header. The oldest Journal Header can be identified by the "JO_" fields 333, 334, 337, and 338 in the management table. The Journal Headers are searched sequentially in the area 210 for the first header whose sequence number JH_SEQ 215 is greater than the sequence number SS_SEQ 351 associated

with the selected snapshot. The selected snapshot is incrementally updated by applying each journal entry, one at a time, to the snapshot in sequential order, thus reproducing the sequence of write operations. This continues as long as the time field JH_TIME 214 of the journal entry is prior to the target time. The update ceases with the first journal entry whose time field 214 is past the target time.

[56] In accordance with one aspect of the invention, a single snapshot is taken. All journal entries subsequent to that snapshot can then be applied to reconstruct the data state at a given time. In accordance with another aspect of the present invention, multiple snapshots can be taken. This is shown in Fig. 5A where multiple snapshots 520' are taken. In accordance with the invention, each snapshot and journal entry is assigned a sequence number in the order in which the object (snapshot or journal entry) is recorded. It can be appreciated that there typically will be many journal entries 510 recorded between each snapshot 520'. Having multiple snapshots allows for quicker recovery time for restoring data. The snapshot closest in time to the target recovery time would be selected. The journal entries made subsequent to the snapshot could then be applied to restore the desired data state.

[57] Fig. 6 illustrates another aspect of the present invention. In accordance with the invention, a journal entry is made for every write operation issued from the host; this can result in a rather large number of journal entries. As time passes and journal entries accumulate, the one or more journal volumes 106 defined by the recovery manager 111 for a journal group 102 will eventually fill up. At that time no more journal entries can be made. As a consequence, subsequent write operations would not be journaled and recovery of the data state subsequent to the time the journal volumes become filled would not be possible.

[58] Fig. 6 shows that the storage system 100 will apply journal entries to a suitable snapshot in response to detection of an "overflow" condition. An "overflow" is deemed to exist when the available space in the journal volume(s) falls below some predetermined threshold. It can be appreciated that many criteria can be used to determine if an overflow condition exists. A straightforward threshold is based on the total storage capacity of the journal volume(s) assigned for a journal group. When the free space becomes some percentage (say, 10%) of the total storage capacity, then an overflow condition exists. Another threshold might be used for each journal volume. In an aspect of the invention, the free space capacity in the journal volume(s) is periodically monitored. Alternatively, the free space can be monitored in an aperiodic manner. For example, the intervals between monitoring can be randomly spaced. As another example, the monitoring intervals can be

spaced apart depending on the level of free space; i.e., the monitoring interval can vary as a function of the free space level.

[59] Fig. 7 highlights the processing which takes place in the storage system 100 to detect an overflow condition. Thus, in a step, 710, the storage system periodically checks the total free space of the journal volume(s) 106; e.g., every ten seconds. The free space can easily be calculated since the pointers (e.g., JI_CTL_VOL 331, JI_CTL_ADDR 332) in the management table 300 maintain the current state of the storage consumed by the journal volumes. If the free space is above the threshold, then the monitoring process simply waits for a period of time to pass and then repeats its check of the journal volume free space.

[60] If the free space falls below a predetermined threshold, then in a step 720 some of the journal entries are applied to a snapshot to update the snapshot. In particular, the oldest journal entry(ies) are applied to the snapshot.

[61] Referring to Fig. 3, the Journal Header 219 of the "oldest" journal entry is identified by the JO_HEAD_VOL field 333 and the JO_HEAD_ADR field 334. These fields identify the journal volume and the location in the journal volume of the Journal Header Area 210 of the oldest journal entry. Likewise, the Journal Data of the oldest journal entry is identified by the JO_DATA_VOL field 337 and the JO_DATA_ADR field 338. The journal entry identified by these fields is applied to a snapshot. The snapshot that is selected is the snapshot having an associated sequence number closest to the sequence number of the journal entry and earlier in time than the journal entry. Thus, in this particular implementation where the sequence number is incremented each time, the snapshot having the sequence number closest to but less than the sequence number of the journal entry is selected (i.e., "earlier in time). When the snapshot is updated by applying the journal entry to it, the applied journal entry is freed. This can simply involve updating the JO_HEAD_VOL field 333, JO_HEAD_ADR field 334, JO_DATA_VOL field 337, and the JO_DATA_ADR field 338 to the next journal entry.

[62] As an observation, it can be appreciated by those of ordinary skill, that the sequence numbers will eventually wrap, and start counting from zero again. It is well within the level of ordinary skill to provide a suitable mechanism for keeping track of this when comparing sequence numbers.

[63] Continuing with Fig. 7, after applying the journal entry to the snapshot to update the snapshot, a check is made of the increase in the journal volume free space as a result of the applied journal entry being freed up (step 730). The free space can be compared against the threshold criterion used in step 710. Alternatively, a different threshold can be used. For

example, here a higher amount of free space may be required to terminate this process than was used to initiate the process. This avoids invoking the process too frequently, but once invoked the second higher threshold encourages recovering as much free space as is reasonable. It can be appreciated that these thresholds can be determined empirically over time by an administrator.

[64] Thus, in step 730, if the threshold for stopping the process is met (i.e., free space exceeds threshold), then the process stops. Otherwise, step 720 is repeated for the next oldest journal entry. Steps 730 and 720 are repeated until the free space level meets the threshold criterion used in step 730.

[65] Fig. 7A highlights sub-steps for an alternative embodiment to step 720 shown in Fig. 7. Step 720 frees up a journal entry by applying it to the latest snapshot that is not later in time than the journal entry. However, where multiple snapshots are available, it may be possible to avoid the time consuming process of applying the journal entry to a snapshot in order to update the snapshot.

[66] Fig. 7A shows details for a step 720' that is an alternate to step 720 of Fig. 7. At a step 721, a determination is made whether a snapshot exists that is later in time than the oldest journal entry. This determination can be made by searching for the first snapshot whose associated sequence number is greater than that of the oldest journal entry. Alternatively, this determination can be made by looking for a snapshot that is a predetermined amount of time later than the oldest journal entry can be selected; for example, the criterion may be that the snapshot must be at least one hour later in time than the oldest journal entry. Still another alternate is to use the sequence numbers associated with the snapshots and the journal entries, rather than time. For example, the criterion might be to select a snapshot whose sequence number is N increments away from the sequence number of the oldest journal entry.

[67] If such a snapshot can be found in step 721, then the earlier journal entries can be removed without having to apply them to a snapshot. Thus, in a step 722, the "JO_" fields (JO_HEAD_VOL 333, JO_HEAD_ADR 334, JO_DATA_VOL 337, and JO_DATA_ADR 338) are simply moved to a point in the list of journal entries that is later in time than the selected snapshot. If no such snapshot can be found, then in a step 723 the oldest journal entry is applied to a snapshot that is earlier in time than the oldest journal entry, as discussed for step 720.

[68] Still another alternative for step 721 is simply to select the most recent snapshot. All the journal entries whose sequence numbers are less than that of the most recent snapshot can

be freed. Again, this simply involves updating the "JO_" fields so they point to the first journal entry whose sequence number is greater than that of the most recent snapshot. Recall that an aspect of the invention is being able to recover the data state for any desired point in time. This can be accomplished by storing as many journal entries as possible and then applying the journal entries to a snapshot to reproduce the write operations. This last embodiment has the potential effect of removing large numbers of journal entries, thus reducing the range of time within which the data state can be recovered. Nevertheless, for a particular configuration it may be desirable to remove large numbers of journal entries for a given operating environment.

[69] It can be appreciated that the foregoing described steps can be embodied entirely in the controller 140 (e.g., a disk controller). This can take on the form of pure software, custom logic, or some suitable combination of software and hardware, depending on the particular implementation. More generally, the foregoing disclosed embodiments typically can be provided using a combination of hardware and software implementations. One of ordinary skill can readily appreciate that the underlying technical solution will be determined based on factors including but not limited or restricted to system cost, system performance, legacy software and legacy hardware, operating environment, and so on. The described current and contemplated embodiments can be readily reduced to specific implementations without undue experimentation by those of ordinary skill in the relevant art.

WHAT IS CLAIMED IS:

1. A method for processing data in an application data store comprising:

producing at least a first snapshot of an application data store, the application data store configured to receive data by way of write operations issued from a host device;

producing a journal entry for each write operation issued from the host device;

storing each journal entry in a journal data store, thereby accumulating a list of journal entries;

monitoring an amount of free space on the journal data store; and

when the free space falls below a first threshold value, then removing one or more journal entries from the journal data store, thereby increasing the free space, wherein enough of the journal entries are removed so that the free space rises above a second threshold value.

2. The method of claim 1 wherein removing one or more journal entries includes updating the first snapshot by applying one or more journal entries to the first snapshot, beginning with an oldest journal entry, wherein journal entries applied to the first snapshot are removed from the list of journal entries thereby increasing the free space of the journal data store.

3. The method of claim 1 wherein removing one or more journal entries includes identifying a selected snapshot that is earlier in time than an oldest journal entry and is closest in time to the oldest journal entry than other snapshots and updating the selected snapshot with one or more journal entries beginning with the oldest journal entry.

4. The method of claim 1 wherein removing one or more journal entries includes looking for a most recent snapshot, removing journal entries in the journal data store that are earlier in time than the most recent snapshot.

5. The method of claim 1 wherein the first threshold value and the second threshold value are different.

6. The method of claim 1 wherein each journal entry comprises a fixed-size header portion and a variable-size data portion, wherein the journal data store comprises a first storage area within which a plurality of header portions are defined and a second storage area for storing a plurality of data portions, wherein producing a journal entry

includes allocating one of the journal headers and allocating a sufficient amount of space in the data portion to contain data associated with the write operation.

7. The method of claim 6 wherein storing each journal entry includes allocating one of the journal headers from the first portion such that the journal headers are always allocated in sequential manner and when the last journal header is allocated then allocating journal headers from the beginning of the first portion in sequential manner.

8. A data processing method comprising:

producing at least a first snapshot of at least a portion of an application data store, the application data store being configured to receive write operations issued from a host device;

recording a plurality of journal entries, each journal entry being recorded for a write operation issued from the host device, each journal entry being stored in a journal data store thereby consuming an amount of free space of the journal data store; and

updating the at least first snapshot with at least one journal entry, including associating space consumed by the at least one journal entry with the free space thereby increasing the free space of the journal data store.

9. The method of claim 8 wherein the step of updating includes periodically monitoring the free space of the journal data store and if the free space falls below a first threshold then updating the at least first snapshot.

10. The method of claim 9 wherein the step of updating the at least first snapshot is repeated for a number of journal entries so that the free space rises above a second threshold value.

11. The method of claim 8 further comprising:

associating sequence numbers to the first snapshot and to the journal entries;

receiving a target time;

determining a beginning journal entry based on the sequence numbers associated with the journal entries and the sequence number associated with the first snapshot;

updating the first snapshot by applying the beginning journal entry to it; and

performing additional updates to the first snapshot by applying journal entries that are subsequent in time to the beginning journal entry and prior in time to the target time.

12. The method of claim 8 further comprising:

producing additional snapshots thereby accumulating a plurality of snapshots;

associating sequence numbers to the snapshots and to the journal entries;

receiving a target time;

determining a selected snapshot based on the target time;

determining a beginning journal entry based on the sequence numbers associated with the journal entries and the sequence number associated with the selected snapshot;

updating the selected snapshot by applying the beginning journal entry to it; and

performing additional updates to the selected snapshot by applying journal entries that are subsequent in time to the beginning journal entry and prior in time to the target time.

13. The method of claim 12 wherein the selected snapshot is a first snapshot.

14. The method of claim 12 wherein the selected snapshot is closest in time to the target time and prior in time to the target time.

15. A method for processing data comprising:

producing a plurality of snapshots of one or more portions of an application data store, the data volume configured to receive data by way of write operations issued from a host device;

recording a journal entry for each write operation issued from the host device, each journal entry being stored in a journal data store thereby accumulating a list of journal entries and consuming free space on the journal data store;

monitoring a remaining free space on the journal data store; and

when the remaining free space falls below a first threshold value, then removing one or more journal entries by identifying a selected snapshot and removing journal entries earlier in time than the selected snapshot.

16. The method of claim 15 wherein the selected snapshot is at least a predetermined length of time later than an oldest journal entry.

17. The method of claim 15 further comprising associating sequence numbers to the snapshots and to the journal entries, wherein the selected snapshot is associated with a sequence number that is greater than a sequence number associated with an oldest journal entry by a predetermined amount.

18. A method for processing data comprising:

producing a plurality of snapshots of an application data store, the data volume configured to receive data by way of write operations issued from a host device;

producing a journal entry for each write operation issued from the host device;

storing each journal entry in a journal data store, thereby accumulating a list of journal entries;

monitoring an amount of free space on the journal data store; and

when the free space falls below a first threshold value, then:

identifying a selected snapshot that is earlier in time than an oldest journal entry and is closest in time to the oldest journal entry than other snapshots;

updating the selected snapshot with one or more journal entries beginning with the oldest journal entry, thereby reducing the list of journal entries; and

repeating the updating step for additional journal entries until the free space rises above a second threshold value.

19. The method of claim 18 further comprising associating sequence numbers to the snapshots and to the journal entries, wherein the selected snapshot is determined based on the sequence numbers of the snapshots and the sequence number of the oldest journal entry.

20. The method of claim 18 wherein journal entries are sequentially allocated and contiguously arranged in the journal data store in a manner representative of a sequential list of journal entries, wherein the step of storing includes wrapping to a beginning of the list of journal entries to reuse earlier journal entries when an end of the list of journal entries is reached.

21. A method for processing data comprising:

producing a plurality of snapshots of an application data store, the application data store configured to receive data by way of write operations issued from a host device;

recording a journal entry for each write operation issued from the host device, the journal entry being stored in a journal data store, thereby accumulating a list of journal entries in the journal data store;

producing a plurality of sequential numbers;

for each snapshot and for each journal entry, associating one of the numbers therewith;

monitoring an amount of free space on the journal data store; and

when the free space falls below a first threshold value, then:

identifying a selected snapshot by comparing sequence numbers associated with the snapshots against a sequence number associated with an oldest journal entry;

updating the selected snapshot with one or more journal entries beginning with the oldest journal entry; and

repeating the updating step with additional journal entries until the free space rises above a second threshold value.

22. The method of claim 21 wherein the step of recording is initiated prior to producing a snapshot, so each write operation that issues during a period of time that a first snapshot is being produced will be recorded in a journal entry.

23. The method of claim 21 wherein the selected snapshot is earlier in time than the oldest journal entry.

24. The method of claim 21 wherein the selected snapshot is earlier in time than the oldest journal entry and closest in time to the oldest journal entry than other snapshots.

25. The method of claim 21 wherein the first threshold is different from the second threshold.

26. A method for processing data in a data storage system comprising:

initiating journal processing wherein a journal entry is created for each write operation that is sent to the data storage system, thus producing a plurality of journal entries, the journal entries being stored in a backup storage system;

subsequent to the step of initiating journal processing, taking one or more snapshots of a portion of the data contained in the data storage system;

associating a sequence number to each journal entry when it is created and to each snapshot when it is taken, wherein sequence numbers associated with the journal entries and with the snapshots are part of the same sequence; and

determining whether to apply some of the journal entries to one of the snapshots in order to reduce the amount of space in the backup storage system that is consumed by the journal entries, based on the amount of available storage capacity in the backup storage system.

27. The method of claim 26 wherein if a determination is made to apply some of the journal entries to one of the snapshots, then:

identifying an oldest journal entry;

identifying a selected snapshot based on a sequence number associated with the oldest journal entry and on sequence numbers associated with the snapshots; and

updating the selected snapshot with one or more journal entries, beginning with the oldest journal entry.

28. A storage system for processing data comprising:

a production data store configured to receive write operations from a host device;

a snapshot data store configured to store one or more snapshots of at least a portion of the production data store;

a journal data store configured to store one or more journal entries; and

a controller configured to:

access the production data store and the snapshot data store to store one or more snapshots of at least a portion of the production data store on the snapshot data store;

access the journal data store to record a journal entry for each write operation from the host device; and

update one of the snapshots with some, but not all, of the journal entries thereby leaving some journal entries for a data recovery operation.

29. The storage system of claim 28 wherein the controller is further configured to:

associate sequence numbers to the snapshots and to the journal entries;

receive a target time;

select a snapshot based on the target time;

determine a beginning journal entry based on the sequence numbers associated with the journal entries and the sequence number of the selected snapshot;

update the selected snapshot by applying the beginning journal entry to it; and

perform additional updates to the selected snapshot by applying journal entries that are later in time than the beginning journal entry and earlier in time than the target time.

30. A storage system for processing data comprising:

means for receiving communications from a host device, the communications including one or more write operations;

first storage means for storing production data;

second storage means for storing snapshots;

third storage means for storing journal entries;

control means for:

storing onto the second storage means one or more snapshots of at least a portion of the production data;

recording onto the third storage means a journal entry for each write operation thereby producing a plurality of journal entries and reducing an amount of free space on the third storage means; and

updating one of the snapshots according to one or more journal entries thereby increasing the amount of free space on the third storage means.

31. The storage system of claim 30 wherein the updating can be performed periodically or at intervals based on the amount of free space.

PATENT

Attorney Docket No.: 16869B-082700US
Client ref. No.: HAL 279

METHOD AND APPARATUS FOR BACKUP AND RECOVERY USING STORAGE BASED JOURNALING

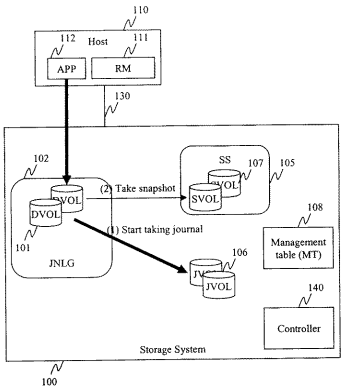
ABSTRACT OF THE DISCLOSURE

A storage system maintains a journal of journal entries and at lease one snapshot of one or more data volumes. By assigning a unique sequence number to journal and snapshot, it is easy to find a journal which can be applied to the snapshot. A technique is described for detecting an overflow condition of running out of journal space and recovering the journal space.

PA 3309031 v1

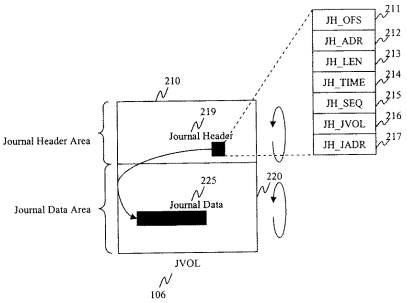
【図１】

Fig. 1 Overview of backup



【図２】

Fig. 2 Control Data for Journal



【図３】

Fig. 3 MT(Management Table) 300

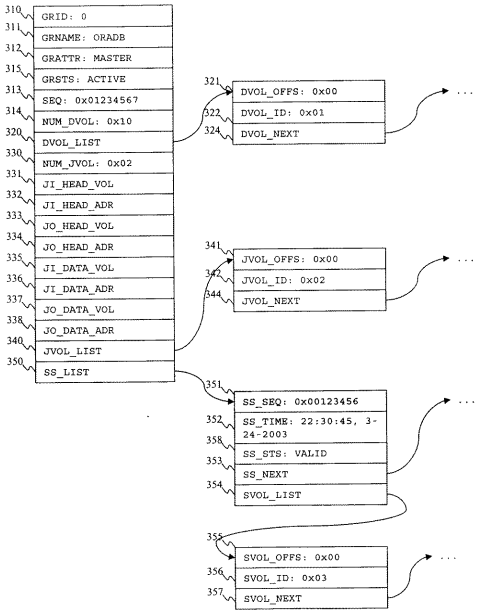
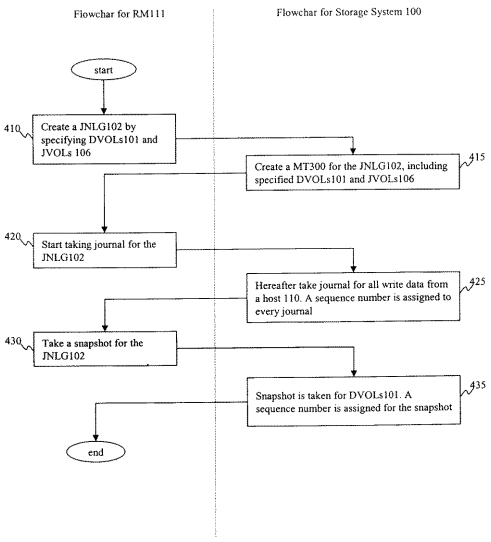


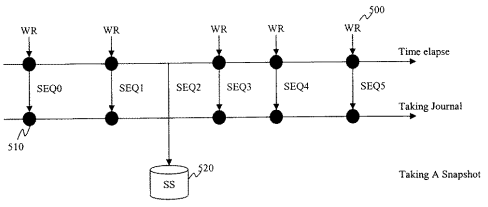
Fig. 4 Starting journal



【図４】

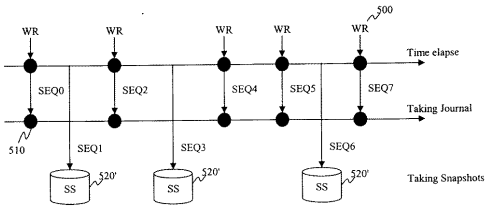
【図５】

Fig. 5 Relationship between snapshot and journal



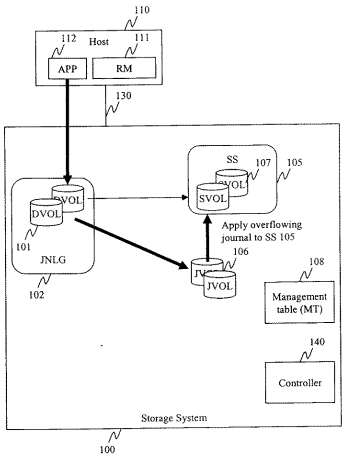
【図５Ａ】

Fig. 5A Relationship between multiple snapshots and journal

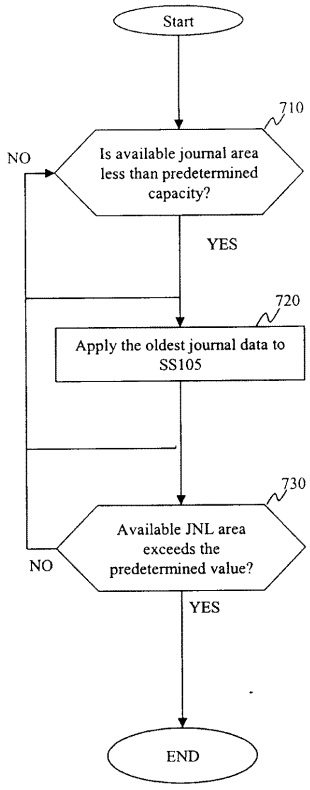


【図６】

Fig. 6 Overflowing JNL



【図７】



【図７Ａ】

Fig. 7 Overflowing JNL

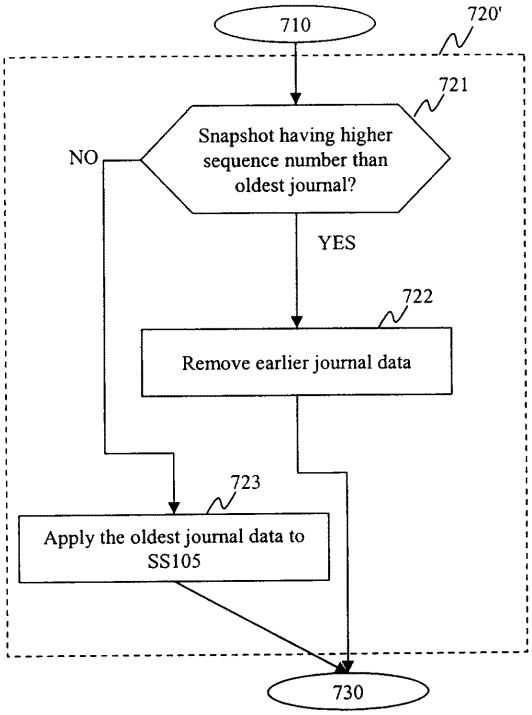

710
720'
721
Snapshot having higher sequence number than oldest journal?
NO
YES
722
Remove earlier journal data
723
Apply the oldest journal data to SS105
730

Fig. 7A