(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号

特許第6408568号
(P6408568)

(45)発行日　平成30年10月17日(2018.10.17)　(24)登録日　平成30年9月28日(2018.9.28)

(51)Int.Cl.		F I		
G06F 3/06	(2006.01)	G06F	3/06	301J
G06F 12/00	(2006.01)	G06F	12/00	514M
G06F 13/14	(2006.01)	G06F	3/06	302A
G06F 12/02	(2006.01)	G06F	13/14	320H
		G06F	12/00	597U

請求項の数 20　(全 46 頁)　最終頁に続く

(21)出願番号　特願2016-521827(P2016-521827)
(86)(22)出願日　平成26年6月20日(2014.6.20)
(65)公表番号　特表2016-524250(P2016-524250A)
(43)公表日　平成28年8月12日(2016.8.12)
(86)国際出願番号　PCT/US2014/043299
(87)国際公開番号　WO2014/205298
(87)国際公開日　平成26年12月24日(2014.12.24)
審査請求日　平成29年6月13日(2017.6.13)
(31)優先権主張番号　13/924,567
(32)優先日　平成25年6月22日(2013.6.22)
(33)優先権主張国　米国(US)

(73)特許権者　314015767
マイクロソフト　テクノロジー　ライセンシング，エルエルシー
アメリカ合衆国　ワシントン州　98052　レッドモンド　ワン　マイクロソフト　ウェイ
(74)代理人　100140109
弁理士　小野　新次郎
(74)代理人　100075270
弁理士　小林　泰
(74)代理人　100101373
弁理士　竹内　茂雄
(74)代理人　100118902
弁理士　山本　修

最終頁に続く

(54)【発明の名称】複数のアクセスメソッドのためのラッチフリーのログ構造化ストレージ

(57)【特許請求の範囲】
【請求項１】
少なくとも１つのプロセッサーを含むデバイスであって、前記少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含むデバイスを備え、前記データマネージャーは、
任意に選択されたページ指向型アクセスメソッドに、ページデータストレージに対するラッチフリーのアクセスを含む前記ページデータストレージに対するインターフェースアクセスをもたらすように構成されたデータ不透明型インターフェースであって、前記データ不透明型インターフェースは、アトミック操作を介して、キャッシュ層ストレージと、二次ストレージとを含むデータストレージの管理のために共通で使用される間接アドレスマッピングテーブル内のページ状態を表すストレージアドレスエントリーに対して、ラッチフリーのページ更新をもたらすデータ不透明型インターフェースと、
二次ストレージバッファーの中に第１ページのページ状態をコピーすることを開始すること、および前記ページ状態の先頭にフラッシュデルタレコードを付加することを開始することに基づいて、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１ページをフラッシュする操作を開始するよう構成されたページマネージャーであって、前記フラッシュデルタレコードは、二次ストレージ内の第１のページのストレージロケーションを示す二次ストレージアドレスと、呼び出し元に関連付けられた注釈とを含む、ページマネージャーと
を含むシステム。

【請求項２】
前記データ不透明型インターフェースは、前記任意に選択されたページ指向型アクセスメソッドに、ページデータストレージに対するログ構造化アクセスを含む前記ページデータストレージに対するインターフェースアクセスをもたらすように構成される請求項１に記載のシステム。
【請求項３】
キャッシュ層マネージャーをさらに備え、前記キャッシュ層マネージャーは、前記データ不透明型インターフェースに関連付けられた間接アドレスマッピングテーブルに対するテーブル操作を開始するように構成されたマップテーブルマネージャーであって、前記テーブル操作は、前記間接アドレスマッピングテーブル内のエントリーに対してアトミック・コンペア・アンド・スワップ操作を開始して、前記ページデータストレージに関連付けられたページのそれまでの状態を、前記ページの新たな状態で置き換えるステップを含むマップテーブルマネージャーを含む請求項１に記載のシステム。
【請求項４】
前記マップテーブルマネージャーは、前記データ不透明型インターフェースに関連付けられた前記間接アドレスマッピングテーブルに対する前記テーブル操作を開始するように構成され、前記間接アドレスマッピングテーブルは、キャッシュ層ストレージと、二次ストレージとを含むデータストレージの管理のために共通で使用される請求項３に記載のシステム。
【請求項５】
前記間接アドレスマッピングテーブルは、ページの論理ロケーションを、前記ページの対応する物理ロケーションから分離し、
前記ページデータストレージのユーザーは、前記ページに関する物理ロケーションアドレス値の代わりにページ識別子値を、前記ページデータストレージを参照するデータ構造における別の場所に記憶させる請求項３に記載のシステム。
【請求項６】
前記間接アドレスマッピングテーブル内のエントリーに対するラッチフリーのコンペア・アンド・スワップ操作を使用して、ページフラッシュからもたらされるログ構造化に関連するページロケーション変更を制御するように構成されたログ構造化ストレージ層マネージャーをさらに備える請求項３に記載のシステム。
【請求項７】
アップデートデルタレコード操作と、置換アップデート操作に基づいてアップデートを制御するように構成されるレコードマネージャをさらに備える請求項３に記載のシステム。
【請求項８】
第１のプロセッサー操作によってページ情報にアクセスすることに先立って、第１のエポックに関連づけられた第１のエポック登録リスト内で、第１のプロセッサー操作の登録を開始するよう構成された第１のエポックマネージャーをさらに備える請求項１に記載のシステム。
【請求項９】
コンペア・アンド・スワップ（ＣＡＳ）操作を介して、マッピングテーブルの中にフラッシュデルタレコードに対するポインターをインストールするステップに基づいて、二次ストレージにページ状態をフラッシュするように構成されたページマネージャーであって、前記フラッシュデルタレコードは、前記ＣＡＳ操作を介して前記マッピングテーブルの中で置き換えられる既存のページ状態の先頭に付加される、ページマネージャーをさらに備える請求項１に記載のシステム。
【請求項１０】
前記ページマネージャーは、
前記ＣＡＳ操作が成功したかどうかを判定し、前記ＣＡＳ操作が成功したと判定された場合、二次ストレージフラッシュバッファーに前記既存のページ状態を書き込む書込み操

作を開始するように構成される請求項９に記載のシステム。
【請求項１１】
前記ページマネージャーは、
前記ＣＡＳ操作が失敗したかどうかを判定し、前記既存のページにそれまで割り当てられていたストレージスペースを無効にする操作を開始するよう構成される請求項１０に記載のシステム。
【請求項１２】
ラッチフリーのアップデート操作を介して、ログ構造化二次ストレージバッファーに対するアップデートを制御するよう構成されたバッファーマネージャーをさらに備える、請求項１に記載のシステム。
【請求項１３】
前記バッファーマネージャーは、前記ラッチフリーの操作を介して、前記ログ構造化二次ストレージバッファーを同時にアップデートするために、複数のスレッドを可能にするよう構成される、請求項１２に記載のシステム。
【請求項１４】
前記バッファーマネージャーは、第１の二次ストレージアドレス引数までの、より下位のアドレスを有する、前記ログ構造化二次ストレージバッファーにフラッシュされたページが、ログ構造化二次ストレージの中で安定していることを判定するための安定化操作を開始するよう構成される、請求項１２に記載のシステム。
【請求項１５】
前記ページマネージャーは、さらに

コンペア・アンド・スワップ（ＣＡＳ）操作を介して、マッピングテーブルの中にフラッシュデルタレコードのアドレスをインストールすることに基づいて前記ページ状態をアップデートすることを開始することに基づいて、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページをフラッシュする操作を開始するように構成される、請求項１に記載のシステム。
【請求項１６】
少なくとも１つのプロセッサーを含むデバイスであって、前記少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含むデバイスを備え、前記データマネージャーは、
任意に選択されたページ指向型アクセスメソッドに、ページデータストレージに対するラッチフリーのアクセスを含む前記ページデータストレージに対するインターフェースアクセスをもたらすように構成されたデータ不透明型インターフェースであって、前記データ不透明型インターフェースは、アトミック操作を介して、キャッシュ層ストレージと、二次ストレージとを含むデータストレージの管理のために共通で使用される間接アドレスマッピングテーブル内のページ状態を表すストレージアドレスエントリーに対して、ラッチフリーのページ更新をもたらす、データ不透明型インターフェースと、
第１のページに関連付けられるページ状態の先頭に部分的スワップデルタレコードを付加することを開始することに基づいて、二次ストレージ内のロケーションに対してキャッシュ層ストレージ内の第１のページの一部分をスワップする操作を開始するよう構成されたページマネージャーであって、前記部分的スワップデルタレコードは、前記第１のページの欠落した部分の二次ストレージ内のロケーションを示すフラッシュデルタレコードのストレージロケーションを示すメインメモリーアドレスを含む、ページマネージャーとを含む、システム。
【請求項１７】
少なくとも１つのプロセッサーを含むデバイスであって、前記少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含むデバイスを備え、前記データマネージャーは、

二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページをフラッシュする操作を、

二次ストレージバッファーの中に前記第１のページのページ状態をコピーすることを開始するステップと、

前記ページ状態の先頭にフラッシュデルタレコードを付加することを開始するステップであって、前記フラッシュデルタレコードは、二次ストレージ内の前記第１のページのストレージロケーションと、呼び出し元に関連付けられた注釈とを示す二次ストレージアドレスを含む、ステップと、

コンペア・アンド・スワップ（ＣＡＳ）操作を介して、マッピングテーブルの中にフラッシュデルタレコードのアドレスをインストールすることに基づいて前記ページ状態に対するラッチフリーのアップデートを開始するステップと、に基づいて、開始するように構成されたページマネージャーを含む、システム。

【請求項１８】

任意に選択されたページ指向型アクセスメソッドに、ページデータストレージに対するラッチフリーのアクセスを含むページデータストレージに対するインターフェースアクセスをもたらすように構成されたデータ不透明型インターフェースと、

ラッチフリーのアップデート操作を介してログ構造化二次ストレージバッファーに対するアップデートを制御するように構成されたバッファーマネージャーと

をさらに含む、請求項１７に記載のシステム。

【請求項１９】

少なくとも１つのプロセッサーを含むデバイスであって、前記少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含むデバイスを備え、

前記データマネージャーは、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページの一部分をラッチフリーでスワップする操作を、アトミック操作を介して、前記第１のページに関連付けられたページ状態の先頭に部分的スワップデルタレコードを付加することを開始するステップにより開始するように構成されたページマネージャーであって、前記部分的スワップデルタレコードは、前記第１のページの欠落した部分の二次ストレージ内のロケーションを示すフラッシュデルタレコードのストレージロケーションを示すメインメモリーアドレスを含む、ページマネージャーを含む、システム。

【請求項２０】

任意に選択されたページ指向型アクセスメソッドに、ページデータストレージに対するラッチフリーのアクセスを含むページデータストレージに対するインターフェースアクセスをもたらすように構成されたデータ不透明型インターフェースをさらに備え、前記データ不透明型インターフェースは、前記任意に選択されたページ指向型アクセスメソッドに、前記ページデータストレージに対するログ構造化アクセスを含むページデータストレージへのインターフェースアクセスをもたらすように構成される、請求項１９に記載のシステム。

【発明の詳細な説明】

【背景技術】

【０００１】

[0001]電子デバイスのユーザーは、しばしば、様々なタイプの情報を獲得するのにデータベースシステムにアクセスする必要がある。データ項目を記憶するため、および取り出すための多くの異なる技法が考案されてきた。例えば、一部の最新のハードウェアプラットフォームは、より高いパフォーマンスをもたらそうとしてマルチコアプロセッサー、多層メモリー階層、ならびにフラッシュなどの二次ストレージデバイスなどの最新のハードウェア開発を活用してきた。このことは、潜在的なシステムパフォーマンスを向上させてきたが、システムが、新たに開発されたプラットフォーム態様、および従来のプラットフォーム態様を有効に利用することは困難であった。

【発明の概要】

【課題を解決するための手段】
【０００２】
[0002]１つの一般的な態様によれば、システムが、少なくとも１つのプロセッサーを含むデバイスを含むことが可能であり、そのデバイスは、その少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含む。データマネージャーは、任意に選択されたページ指向型アクセスメソッドに、ページデータストレージに対するラッチフリー（latch-free）のアクセスを含むページデータストレージに対するインターフェースアクセスをもたらすように構成されたデータ不透明型インターフェース（data opaque interface）を含み得る。
【０００３】
[0003]別の態様によれば、システムが、少なくとも１つのプロセッサーを含むデバイスを含むことが可能であり、そのデバイスは、その少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含む。データマネージャーは、コンペア・アンド・スワップ（ＣＡＳ：Compare and swap）操作を介して、マッピングテーブルの中でフラッシュデルタレコードにポインターをインストールすることに基づいて、二次ストレージにページ状態をフラッシュするように構成されたページマネージャーを含むことが可能であり、そのフラッシュデルタレコードは、ＣＡＳ操作を介してマッピングテーブルの中で置き換えられる既存のページ状態の先頭に付加される。
【０００４】
[0004]別の態様によれば、システムが、少なくとも１つのプロセッサーを含むデバイスを含むことが可能であり、そのデバイスは、その少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含む。データマネージャーは、コンペア・アンド・スワップ（ＣＡＳ）操作を介して、二次ストレージバッファーの中に第１のページのページ状態をコピーすることを開始すること、そのページ状態の先頭にフラッシュデルタレコードを付加することを開始することであって、そのフラッシュデルタレコードは、二次ストレージ内の第１のページのストレージロケーションを示す二次ストレージアドレスと、呼び出し元に関連付けられた注釈とを含むこと、およびマッピングテーブルの中にフラッシュデルタレコードのアドレスをインストールすることに基づいてページ状態をアップデートすることを開始することに基づいて、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページをフラッシュする操作を開始するように構成されたページマネージャーを含み得る。
【０００５】
[0005]別の態様によれば、システムが、少なくとも１つのプロセッサーを含むデバイスを含むことが可能であり、そのデバイスは、その少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含む。データマネージャーは、ラッチフリーのアップデート操作を介してログ構造化二次ストレージバッファーに対するアップデートを制御するように構成されたバッファーマネージャーを含み得る。
【０００６】
[0006]別の態様によれば、システムが、少なくとも１つのプロセッサーを含むデバイスを含むことが可能であり、そのデバイスは、その少なくとも１つのプロセッサーによって実行されるようにコンピューター可読記憶媒体上に実体化された命令を備えるデータマネージャーを含む。データマネージャーは、第１のページに関連付けられたページ状態の先頭に部分的スワップデルタレコードを付加することを開始することであって、その部分的スワップデルタレコードは、第１のページの欠落した部分の二次ストレージ内のロケーションを示すフラッシュデルタレコードのストレージロケーションを示すメインメモリーアドレスを含むことに基づいて、二次ストレージ内のロケーションに対してキャッシュ層ストレージ内の第１のページの一部分をスワップする操作を開始するように構成されたページマネージャーを含み得る。

【０００７】
[0007]この概要は、詳細な説明において後段でさらに説明される選定された概念を、簡略化された形態で概説するように与えられる。この概要は、主張される主題の重要な特徴または不可欠な特徴を特定することは意図しておらず、主張される主題の範囲を限定するのに使用されることも意図していない。１つまたは複数の実施形態の詳細は、添付の図面、および後段の説明において示される。その他の特徴が、説明および図面、ならびに特許請求の範囲から明白となろう。
【図面の簡単な説明】
【０００８】
【図１】[0008]キャッシュ／ストレージ層に関するアクセスメソッドのための例示的なアーキテクチャー階層化を示す図である。
【図２】[0009]複数のアクセスメソッドのためのラッチフリーのログ構造化ストレージのための例示的なアーキテクチャーを示すブロック図である。
【図３】[0010]例示的なマッピングテーブルを示す図である。
【図４ａ】[0011]例示的なマッピングテーブル上の例示的なデルタアップデートを示す図である。
【図４ｂ】例示的なマッピングテーブル上の例示的なデルタアップデートを示す図である。
【図５】[0012]例示的な部分的ページスワップアウトおよび例示的な部分的スワップデルタを示す図である。
【図６】[0013]例示的なエポック、およびそれらのエポックのそれぞれのガベージコレクションリストを示す図である。
【図７ａ】[0014]フラッシュ上の例示的なログ構造化ストレージ編成を示す図である。
【図７ｂ】フラッシュ上の例示的なログ構造化ストレージ編成を示す図である。
【図７ｃ】フラッシュ上の例示的なログ構造化ストレージ編成を示す図である。
【図８】[0015]例示的なフラッシュバッファー状態を示す図である。
【図９】[0016]例示的なトランザクションテンプレートを示す図である。
【図１０】[0017]例示的なチェックポイントデータを示す図である。
【図１１】[0018]複数のアクセスメソッドのためのラッチフリーのログ構造化ストレージのための例示的なシステムを示すブロック図である。
【図１２ａ】[0019]図１１のシステムの例示的な動作を示すフローチャートである。
【図１２ｂ】図１１のシステムの例示的な動作を示すフローチャートである。
【図１２ｃ】図１１のシステムの例示的な動作を示すフローチャートである。
【図１２ｄ】図１１のシステムの例示的な動作を示すフローチャートである。
【発明を実施するための形態】
【０００９】
[0020]Ｉ．概説
ハードウェアプラットフォームの最新の開発は、より高いパフォーマンスをもたらそうとしてマルチコアプロセッサー、多層メモリー階層、ならびにフラッシュなどの二次ストレージデバイスを活用してきた。例えば、中央処理装置（ＣＰＵ）変更は、マルチコアプロセッサー、ならびに複数のレベルのキャッシングが関与するメインメモリーアクセスを含む。例えば、フラッシュストレージ、ならびに容量を低下させるアップデート・インプレース（update-in-place）を行うハードディスクベンダ認識は、ログ構造化をより多く使用することにつながってきた。例えば、クラウドデータセンターは、システムスケールを増大させ、コモディティハードウェアの使用は、高い利用可能性技法により大きな重きを置く。
【００１０】
[0021]しかし、潜在的なシステムパフォーマンスが向上する一方で、システムが、これらの最新のプラットフォーム態様を有効に利用することは困難であり得る。例えば、大量のデータにアクセスする複数のユーザーをサポートするデータ指向型システムは、過去に

長年、存在してきたとおりのハードウェアのために設計されたソフトウェアアーキテクチャーを活用する可能性がある（例えば、それらのシステムは、単一レベルメモリー（プロセッサーキャッシングをほとんど行わず、メインメモリーに対して控えめな潜時しか有さない）上で動作し、磁気ディスクにアクセスして、ユニプロセッサーをターゲットにすることが可能である）。

【００１１】

[0022]そのアプローチを変えようとする取組みは、環境を改善してきたが、相変わらず、相当な潜在的パフォーマンス利得を逃している。例えば、データ競合にアクセスする際、ブロッキングを生じさせるラッチを回避する取組みが行われてきたが、これらの取組みには、スレッドがそのような競合を回避するようにパーティションすることが関与してきた可能性があり、このことは、相当なオーバーヘッドをもたらし得る。例えば、データのアップデート・インプレースを行うことは、メモリーパフォーマンスに悪影響を及ぼす可能性があり、このことは、キャッシュ・ライン・アラインメント（cache line alignment）を配慮すること、およびベクトルにわたる二分探索の代わりにローカルツリーを使用することにつながり得る。しかし、不利に多くのアップデート・インプレースが相変わらず行われ、このことが、例えば、キャッシュ無効化を介して、プロセッサー・キャッシング・パフォーマンスに悪影響を及ぼす。さらに、実施形態が、フラッシュを、フラッシュのより高い毎秒アクセス数、およびより低いアクセス潜時のために活用し始めている。しかし、ランダムなアップデートは、フラッシュ変換層を使用してさえ、比較的高価であり得る。

【００１２】

[0023]Ｊ．Ｌｅｖａｎｄｏｓｋｉ他、「Ｄｅｕｔｅｒｏｎｏｍｙ：Ｔｒａｎｓａｃｔｉｏｎ Ｓｕｐｐｏｒｔ ｆｏｒ Ｃｌｏｕｄ Ｄａｔａ（Ｄｅｕｔｅｒｏｎｏｍｙ：クラウドデータのためのトランザクションサポート）」、Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｉｎｎｏｖａｔｉｖｅ Ｄａｔａ Ｓｙｓｔｅｍｓ Ｒｅｓｅａｒｃｈ（ＣＩＤＲ）（２０１１年１月）、１２３～１３３ページ、およびＤ．Ｌｏｍｅｔ他、「Ｕｎｂｕｎｄｌｉｎｇ Ｔｒａｎｓａｃｔｉｏｎ Ｓｅｒｖｉｃｅｓ ｉｎ ｔｈｅ Ｃｌｏｕｄ（クラウドにおけるトランザクションサービスのアンバンドリング）」、Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｉｎｎｏｖａｔｉｖｅ Ｄａｔａ Ｓｙｓｔｅｍｓ Ｒｅｓｅａｒｃｈ（ＣＩＤＲ）、２００９年が、クラウド環境において一貫性（すなわち、トランザクション）をもたらすための例示的な技法を説明する。本明細書で説明される例示的な技法は、例示的なＤＥＵＴＥＲＯＮＯＭＹデータコンポーネント（ＤＣ）、および現在のハードウェア上でそのＤＣのパフォーマンスを最大にすることに焦点を合わせる。例えば、ＤＣが、ＣＲＵＤ（作成、読取り、アップデート、削除）アトミック操作を介してアクセスされるデータの記憶および取出しを管理することが可能である。例えば、ＤＣは、ＤＣの上のソフトウェア層（例えば、ＤＥＵＴＥＲＯＮＯＭＹトランザクションコンポーネント（ＴＣ）および／またはクエリエンジン）を介して融合されて分散システムにされ得るローカル機構を代わりに使用して、非分散型であってもよい。

【００１３】

[0024]本明細書でさらに説明されるとおり、アクセスメソッド（例えば、Ｂツリー、ハッシング、マルチ属性、時間的、その他）に影響を及ぼし得る、現在のハードウェアによってもたらされる問題が存在するように思われる。さらに、本明細書で説明されるとおり、これらの問題は、ほとんどの（例えば、任意に選択された）アクセスメソッドに適用可能である例示的な一般的な機構を用いて解決され得る。

【００１４】

[0025]例えば、本明細書で説明される例示的な技法によれば、ラッチフリーの技法は、マルチコアプロセッサーを用いて有利なプロセッサー利用およびスケーリングを実現するのに利用され得る。例えば、本明細書で説明されるとおり、キャッシュ無効化を少なくするデルタアップデートが、マルチレベルキャッシュベースのメモリーシステムを用いて有利なパフォーマンスを実現するのに利用され得る。例えば、ランダム書込みの限られたパ

フォーマンス、およびフラッシュ書込み限度を有する書込み制限されたストレージが、ログ構造化を介して克服され得る。

【００１５】

[0026]例えば、ＢＷ－ＴＲＥＥ（例えば、Ｊ．Ｌｅｖａｎｄｏｓｋｉ他、「Ｔｈｅ Ｂｗ－Ｔｒｅｅ：Ａ Ｂ－ｔｒｅｅ ｆｏｒ Ｎｅｗ Ｈａｒｄｗａｒｅ Ｐｌａｔｆｏｒｍｓ（Ｂｗ－Ｔｒｅｅ：新しいハードウェアプラットフォームのためのＢツリー）」、２９ｔｈ ＩＥＥＥ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｄａｔａ Ｅｎｇｉｎｅｅｒｉｎｇ（ＩＣＤＥ ２０１３）、２０１３年４月８～１１日参照）、Ｂツリーといくぶん似通ったインデックス（例えば、Ｒ．Ｂａｙｅｒ他、「Ｏｒｇａｎｉｚａｔｉｏｎ ａｎｄ Ｍａｉｎｔｅｎａｎｃｅ ｏｆ Ｌａｒｇｅ Ｏｒｄｅｒｅｄ Ｉｎｄｉｃｅｓ（大次数インデックスの編成および管理）」、Ａｃｔａ Ｉｎｆｏｒｍａｔｉｃａ、Ｖｏｌ．１、Ｉｓｓｕｅ ３、１９７２、１７３～１８９ページ、およびＤ．Ｃｏｍｅｒ、「Ｔｈｅ Ｕｂｉｑｕｉｔｏｕｓ Ｂ－ｔｒｅｅ（ユビキタスＢツリー）」、ＡＣＭ Ｃｏｍｐｕｔｉｎｇ Ｓｕｒｖｅｙｓ（ＣＳＵＲ）、Ｖｏｌ．１１、Ｉｓｓｕｅ ２、１９７９年６月、１２１～１３７ページ参照）が、これらの例示的な技法を活用することが可能なＤＣまたはキー・バリューストア（key-value store）の例である。例示的なＢＷ－ＴＲＥＥには、ラッチフリーダムであることおよびログ構造化をより一般的に実現するための技法のためのパラダイムが関与し得る。本明細書で説明される例示的な技法によれば、ラッチフリーの技法およびログ構造技法は、従来のキャッシュ／ストレージサブシステムが、アップデート・インプレースとしてディスクにライトバック（write back）される固定サイズページに対するラッチアクセス（latched access）を扱い得るのといくぶん似通った様態で複数のアクセスメソッドをサポートすることができるキャッシュ／ストレージサブシステムにおいて実施され得る。

【００１６】

[0027]本明細書で説明される例示的な技法によれば、ＬＬＡＭＡ（Ｌａｔｃｈ－ｆｒｅｅ，Ｌｏｇ－ｓｔｒｕｃｔｕｒｅｄ Ａｃｃｅｓｓ Ｍｅｔｈｏｄ Ａｗａｒｅ）と本明細書で呼ばれ得る例示的なシステムが、最近、開発されたハードウェア環境（例えば、フラッシュ、マルチコア）のためのキャッシング－ストレージサブシステムを（少なくとも）含み、ただし、そのような例示的な技法は、最近、開発されたハードウェアだけに限定されないことがデータ処理の分野の業者には理解されよう。

【００１７】

[0028]例えば、ＬＬＡＭＡは、プロセッサーキャッシュおよび二次ストレージを最適化して、キャッシュ管理とストレージ管理の両方をもたらす任意に選択されたページ指向型アクセスメソッドのためのアプリケーションプログラミングインターフェース（ＡＰＩ）をサポートすることが可能である。例えば、キャッシング（ＣＬ）層とストレージ（ＳＬ）層が、ページの論理ロケーションと物理ロケーションを分離する共通のマッピングテーブルを使用することが可能である。例えば、キャッシュ層（ＣＬ）が、キャッシュ層（ＣＬ）のマッピングテーブル上のラッチフリーのコンペア・アンド・スワップ・アトミック状態変化を介してデータアップデートおよび管理アップデート（例えば、インデックス再編成のための）をサポートすることが可能である。

【００１８】

[0029]例えば、ストレージ層（ＳＬ）が、ページフラッシュごとにログ構造化することによってもたらされるページロケーション変化を扱うのに同一のマッピングテーブルを使用することが可能である。例えば、ラッチフリーのＢＷ－ＴＲＥＥ実施形態（例えば、順序付けられたＢツリー型インデックスの例としてＢＷ－ＴＲＥＥを使用する実施形態）が使用され得る。この脈絡において、「フラッシュする」操作とは、メインメモリー（例えば、キャッシュストレージ）からのページを、そのページを出力バッファーにコピーすることによって、二次ストレージに転送することを指すことが可能である。

【００１９】

[0030]本明細書で説明される例示的な技法は、ページのロケーションとサイズの両方を

視覚化し得るマッピングテーブルをもたらすことが可能である。例えば、そのような視覚化は、本明細書でさらに説明されるとおり、メインメモリー設計と安定したストレージ設計（例えば、ログ構造化ストレージ設計）の両方のために利用され得る。

【００２０】

[0031]この脈絡において、「ページ」とは、物理ストレージアドレスを介してアクセスされ得るストレージ内のオブジェクトを指すことが可能である。本明細書で使用される「ページ」は、柔軟性のあるサイズに関連付けられることが可能であり、ストレージの複数の不連続に記憶されたセグメントにわたって分散することが可能なストレージのページ単位を表すことが可能である。ストレージには、揮発性ストレージおよび／または安定したストレージが含まれ得る。

【００２１】

[0032]本明細書で説明される例示的な技法は、アクセスメソッド層をキャッシュ／ストレージ管理から分離することが可能である。例として、本明細書で説明される技法は、先行書込みログプロトコル（write-ahead log protocol）を執行するのに使用され得る。例えば、ページをフラッシュするのに先立って、従来のデータベースカーネルは、ページログシーケンスナンバー（ＬＳＮ）を調べて、トランザクションログにおいてまだ安定していないアップデートが存在するかどうかを判定することが可能である。例えば、ＬＬＡＭＡキャッシュ管理は、例示的なデルタアップデートを活用して、部分的ページを「スワップアウトする」ことが可能である。例えば、ＬＬＡＭＡキャッシュ管理は、二次ストレージ上に既に存在するページの部分（最近のデルタアップデートを含まない）をキャッシュからドロップすることができる。例えば、アクセスメソッド層が、トランザクション・ログ・チェックポインティングのために定期的にフラッシュされる。このため、キャッシュマネージャーは、任意のバッファーサイズ制約を満足させるのに十分な候補（場合により、部分的な）ページを見出す。

【００２２】

[0033]本明細書で説明される例示的な技法は、相当な数のアクセスメソッド（すなわち、単一のインスタンスだけでない）がこれらの技法を活用することを、これらの技法をもたらすサブシステム層を実施することによって可能にするフレームワークを提供することが可能である。さらに、有利な効率をもたらす二次ストレージにデータを書き込むためのログ構造化ストアが実施されることが可能である。したがって、アクセスメソッドは、アクセスメソッドのインデックスのメインメモリー態様に焦点を合わせることが可能であり、本明細書で説明される例示的な技法は、ＢＷ－ＴＲＥＥのパフォーマンスメトリックと同様のパフォーマンスメトリックを実現するためのフレームワークをもたらすことが可能である。

【００２３】

[0034]例えば、ＬＬＡＭＡなどの技法が、その技法のＡＰＩを介して、ラッチフリーのページアップデートをもたらすことが可能であり、このことは、マッピングテーブル上のコンペア・アンド・スワップ（ＣＡＳ）アトミック操作を介してメインメモリーの中で達せられる。

【００２４】

[0035]例えば、キャッシュを管理する際に、ＬＬＡＭＡなどの技法は、ページの以前にフラッシュされた部分だけをメモリーからドロップすることによって、メインメモリーを取り戻すことが可能であり、このため、「汚れた」ページをスワップアウトする場合でさえ、入出力（Ｉ／Ｏ）操作がまったく関与しない。このため、ＬＬＡＭＡなどの技法は、その技法のアクセスメソッドユーザーからの入力なしに、その技法のバッファー・キャッシュ・メモリー・サイズを制御できることが可能である。

【００２５】

[0036]例えば、二次ストレージの効果的な管理のために、ＬＬＡＭＡなどの技法が、ログ構造化を利用することが可能である。例えば、ＬＬＡＭＡなどの技法が、部分的ページフラッシュ、および実質的に空きスペースのない、すなわち、実質的に１００％ストレー

ジ利用率のページを使用することによって、従来のログ構造化と比べてパフォーマンスを向上させることが可能である。これらは、ページがフラッシュされる際の入出力操作（Ｉ／Ｏ）の回数、および１ページ当たり消費されるストレージの量を減らすことが可能であり、したがって、ログ構造化が使用される場合に経験され得るライトアンプリフィケーション（write amplification）を減らすことが可能である。さらに、すべてのストレージ関連の操作が、完全にラッチフリーであることが可能である。

【００２６】

[0037]例えば、ＬＬＡＭＡなどの技法が、（少なくとも）制限された形態のシステムトランザクションをもたらすことが可能である。この意味で、システムトランザクションは、ユーザーレベルトランザクションではなく、むしろ、ログ構造化ストアを活用して、もっぱら、アクセスメソッドの「プライベート使用」のため（例えば、インデックス構造変更（ＳＭＯ）のため）にアトミック性をもたらす。例えば、このことは、同時に行われるアップデートが続けられる間にインデックスが成長するにつれ、インデックスが適応することを可能にし得る。

【００２７】

[0038]例えば、ＢＷ－ＴＲＥＥ構造は、あるタイプのラッチフリーのＢツリー構造を含み得る。例えば、ＢＷ－ＴＲＥＥノードに対するアップデートが、それまでのページ状態の先頭にアップデートデルタを付加することに基づいて実行され得る。このため、ＢＷ－ＴＲＥＥは、複数のスレッドによるページに対する同時に行われるアクセスを許すことが可能であるので、ラッチフリーであり得る。そのようなデルタアップデートは、ページのそれまでの状態を保存するため、向上したプロセッサーキャッシュパフォーマンスをもたらすことも可能である。

【００２８】

[0039]ＢＷ－ＴＲＥＥを使用する例示的な技法は、やはりラッチフリーであり、Ｂリンクツリー型サイドポインターを使用することが可能なページ分割技法をさらにもたらし得る。分割（および他の構造変更操作）は、メインメモリー内部でも、安定させられる際もともにアトミックであり得る。例えば、アトミックレコードストアが、ＢＷ－ＴＲＥＥアーキテクチャーに基づいて実施され得る。

【００２９】

[0040]本明細書の説明の趣旨を逸脱することなく、本明細書で説明されるラッチフリーのログ構造化ストレージを実現する多くの方法が存在し得ることが、データ処理の分野の業者には理解されよう。

【００３０】

[0041]ＩＩ．例示的な動作環境

本明細書で説明される特徴は、本明細書の説明の趣旨を逸脱することなく、データ処理の分野の業者によって理解され得る多くの異なる方法において実施され得る例示的な実施形態として与えられる。そのような特徴は、例示的な実施形態特徴としてのみ解釈されるべきであり、それらの詳細な説明だけに限定されるものと解釈されることは意図していない。

【００３１】

[0042]図１は、キャッシュ／ストレージ層に関するアクセスメソッドのための例示的なアーキテクチャー階層化を示す。アクセスメソッド層１０２は、図１に示されるとおり、最上位層である。アクセスメソッド層１０２は、中位層であるキャッシュ層１０４と対話する。アプリケーションプログラミングインターフェース（ＡＰＩ）１０６が、アクセスメソッド層１０２とキャッシュ層１０４の間の活動のために使用され得る。例示的なストレージ層１０８が、マッピングテーブル１１０と対話することが可能であり、マッピングテーブル１１０は、キャッシュ層１０４とストレージ層１０８の間で共有され得る。例えば、ＬＬＡＭＡ１１２が、キャッシュ層１０４およびストレージ層１０８を含む。例えば、ストレージ層が、ログ構造化フラッシュストアをサポートすることが可能である。本明細書で説明される例示的な技法によれば、ログ構造化ストアが、フラッシュとディスクス

トレージの両方を管理することが可能である。この設計は、アーキテクチャー上、既存のデータベースカーネルと共存することが可能である一方で、スタンドアロン型またはＤＥＵＴＥＲＯＮＯＭＹ型のアトミックレコードストア（ＡＲＳ）としても適している。

【００３２】

[0043]例えば、ＬＬＡＭＡなどの技法が、キャッシュ／ストレージ層に関するアクセスメソッド実施形態をサポートして、ページ抽象化をサポートすることが可能である。さらに、トランザクションコンポーネント（例えば、ＤＥＵＴＥＲＯＮＯＭＹ型トランザクションコンポーネント）が上に追加され得る。図２は、複数のアクセスメソッドのためのラッチフリーのログ構造化ストレージのための例示的なアーキテクチャーを示すブロック図である。図２に示されるとおり、トランザクションコンポーネント２０２が、トランザクションキー・バリューストアをサポートすることが可能であり、アトミックキー・バリューストアを含み得るデータコンポーネント２０４と一緒に動作することが可能である。図２に示されるとおり、データコンポーネント２０４は、ラッチフリーの順序付けられたインデックス２０６および／またはラッチフリーの線形ハッシングインデックス２０８を含み得る。図２に示されるとおり、データコンポーネント２０４は、例示的なラッチフリーのログ構造化アクセスメソッドアウェア（ＬＬＡＭＡ）ストレージエンジン２１０（例えば、図１のＬＬＡＭＡ１１２）をさらに含み得る。

【００３３】

[0044]例示的なＡＰＩ１０６は、「データ不透明型」であることが可能であり、すなわち、例示的なＬＬＡＭＡ実施形態は、アクセスメソッド（例えば、アクセスメソッド層１０２の）が何をページまたはデルタレコードに入れているかを「見る」ことがなく（例えば、検査する、または分析する、または依存することがなく）、アクセスメソッドによってページまたはデルタレコードの中で何が提供されるかとは無関係に動作する。このため、本明細書で説明される例示的なＬＬＡＭＡ実施形態は、前述したとおり、アクセスメソッドによって何が提供されるかに依存しない特定の操作に応答して動作することが可能である。

【００３４】

[0045]図３に示されるとおり、ページ３０２が、メインメモリーキャッシュ３１２内で、または二次ストレージ３１４上で、ページ識別子（ＰＩＤ）３０６を状態３０８にマップする（例えば、マッピングテーブル３０４に記憶された「物理アドレス」３１０を介して）マッピングテーブル３０４を介してアクセスされ得る。例えば、メインメモリーキャッシュ３１２は、ランダムアクセスメモリー（ＲＡＭ）を含み得る。例えば、二次ストレージ３１４は、フラッシュメモリーを含み得る。例えば、ページ３０２は、オンデマンドで二次ストレージ３１４からメインメモリーキャッシュ３１２に読み取られることが可能であり、ページ３０２は、二次ストレージ３１４にフラッシュされることが可能であり、ページ３０２は、キャッシュ３１２内にある間、ページ状態を変更するようにアップデートされることが可能である。例えば、実質的にすべてのページ状態変更（データ状態と管理状態の両方）は、本明細書で説明される例示的な技法により、アトミック操作として与えられ得る。図３に示されるとおり、例示的な物理アドレス３１０が、その物理アドレスがフラッシュまたはメモリー（例えば、キャッシュ）に関連付けられているかどうかを示すフラッシュ／メモリーフラグ３１６（例えば、この例に示されるとおり、１ビットに関する）を、（少なくとも）そのアドレス自体に関するアドレスフィールド３１８（例えば、この例に示されるとおり、６３ビットに関する）と一緒に含み得る。本明細書の説明の趣旨を逸脱することなく、「物理アドレス」を表す多くの方法（例えば、６４ビット表現以外の）が存在することが、データ処理の分野の業者には理解されよう。

【００３５】

[0046]本明細書で説明される例示的な技法によれば、ＬＬＡＭＡが、ＬＬＡＭＡのＡＰＩを通じて、（例えば、スレッドをブロックすることによって同時に行われるアクセスからページを保護する従来のラッチの代わりに）マッピングテーブル３０４上でコンペア・アンド・スワップ（ＣＡＳ）アトミック操作を介してラッチフリーのページアップデート

をもたらすことが可能である。例えば、ＣＡＳ戦略は、プロセッサー利用率を有利に高めること、およびマルチコアスケーリングを向上させることが可能である。

【００３６】

[0047]本明細書で説明される例示的な技法によれば、キャッシュを管理する際、ＬＬＡＭＡは、ページの以前にフラッシュされた部分だけをメモリーからドロップすることによって、メインメモリーを取り戻すことが可能であり、このため、「汚れた」ページをスワップアウトする場合でさえ、入出力（Ｉ／Ｏ）操作をまったく使用しない。このため、ＬＬＡＭＡなどの例示的なアーキテクチャーは、そのアーキテクチャーのアクセスメソッドユーザーによってページの中に記憶されたデータを検査する必要なしに、そのアーキテクチャーのバッファー・キャッシュ・メモリー・サイズを制御することが可能である（例えば、ＬＬＡＭＡなどの例示的なアーキテクチャーは、トランザクションおよび先行書込みロギングを意識しないので）。

【００３７】

[0048]ＬＬＡＭＡなどの例示的なアーキテクチャーは、ログ構造化を使用して二次ストレージを管理することが可能である（例えば、ランダムな書込みを回避すること、大きいマルチページバッファーを介する書込みの回数を減らすこと、およびフラッシュメモリーに関係するウェアレベリングの利点を提供して）。さらに、ＬＬＡＭＡなどの例示的なアーキテクチャーが、部分的ページフラッシュ、および実質的に空きスペースのない、すなわち、実質的に１００％ストレージ利用率のページを用いてパフォーマンスを有利に向上させる（例えば、従来のログ構造化と比較して）ことが可能である。例えば、これらは、ページがフラッシュされる際のＩ／Ｏの回数、および１ページ当たり消費されるストレージの量を減らすことが可能であり、したがって、さもなければ、ログ構造化が使用される場合に経験され得るライトアンプリフィケーションを減らすことが可能である。さらに、実質的にすべてのストレージ関連の操作が、完全にラッチフリーであることが可能である。

【００３８】

[0049]さらに、ＬＬＡＭＡなどの例示的なアーキテクチャーが、（少なくとも）制限された形態のシステムトランザクションをサポートすることが可能である（システムトランザクションに関しては、例えば、Ｄ．Ｌｏｍｅｔ他、「Ｕｎｂｕｎｄｌｉｎｇ Ｔｒａｎｓａｃｔｉｏｎ Ｓｅｒｖｉｃｅｓ ｉｎ ｔｈｅ Ｃｌｏｕｄ（クラウドにおけるトランザクションサービスのアンバンドリング）」、Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｉｎｎｏｖａｔｉｖｅ Ｄａｔａ Ｓｙｓｔｅｍｓ Ｒｅｓｅａｒｃｈ（ＣＩＤＲ）、２００９参照）。例えば、システムトランザクションは、ユーザートランザクションではないことが可能であり、むしろ、もっぱら、アクセスメソッドの「プライベート使用」のため（例えば、インデックス構造変更（ＳＭＯ）（Ｃ．Ｍｏｈａｎ他、「ＡＲＩＥＳ／ＩＭ：Ａｎ Ｅｆｆｉｃｉｅｎｔ ａｎｄ Ｈｉｇｈ Ｃｏｎｃｕｒｒｅｎｃｙ Ｉｎｄｅｘ Ｍａｎａｇｅｍｅｎｔ Ｍｅｔｈｏｄ Ｕｓｉｎｇ Ｗｒｉｔｅ－Ａｈｅａｄ Ｌｏｇｇｉｎｇ（ＡＲＩＥＳ／ＩＭ：先行書込みロギングを用いた効率的な高同時実行性インデックス管理方法）」、Ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ １９９２ ＡＣＭ ＳＩＧＭＯＤ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｍａｎａｇｅｍｅｎｔ ｏｆ Ｄａｔａ（ＳＩＧＭＯＤ’９２）、１９９２、３７１～３８０ページ）のため）にアトミック性をもたらすことが可能である。例えば、トランザクションログとは別個に記録されたシステムトランザクションが効果的であり得るという特性が、データベースカーネルを分解することに対するＤＥＵＴＥＲＯＮＯＭＹアプローチの有利な洞察の例である。

【００３９】

[0050]後段の説明は、ＬＬＡＭＡなどの例示的なアーキテクチャーを使用する際にアクセスメソッド実施者が経験し得る例示的な操作インターフェースのさらなる説明を、そのアーキテクチャーがどのように使用され得るかのさらなる説明とともに含む。後段の説明は、本明細書で説明される例示的な技法による、例示的なキャッシュ層、およびログ構造

化ストレージ層の例示的な設計のさらなる説明を含む。さらに、本明細書で説明される例示的な技法による、アトミック性をもたらすものと理解され得る例示的なシステムトランザクション機構および例示的な測定に関する説明が与えられる。さらに、本明細書で説明される例示的な技法による、システムクラッシュからの例示的なログ構造化ストレージ回復に関する説明が与えられる。

【００４０】

[0051]ＬＬＡＭＡなどの例示的なシステムを設計する際、設計目標は、目標が可能な限り「汎用」であることを含むことが可能であり、このことは、ときとして、目標が可能な限り「低レベル」であることにつながる可能性がある。しかし、ＬＬＡＭＡなどの例示的なシステムが「汎用」であるには、そのシステムのファシリティを使用する際にアクセスメソッドが何を行うかについて可能な限りわずかしか知らずに、効果的に動作することが望ましい可能性がある。このため、ＬＬＡＭＡなどの例示的なシステムの動作は、キャッシュ管理、およびページのアップデートをターゲットとして、「原始的」であり得る。例えば、ＬＬＡＭＡなどの例示的なシステムは、ＳＭＯ（例えば、ページ分割およびページマージ）のために有利に含められ得る原始的トランザクション機構をサポートするいくつかのさらなるファシリティを含み得る。

【００４１】

[0052]本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムが、ログシーケンスナンバー（ＬＳＮ）、先行書込みロギング、またはトランザクションログに関するチェックポイントに関するインターフェースに何も含めないことが可能である。本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムは、ユーザー操作に関する冪等性試験（idempotence test）をまったく含まないことが可能である。さらに、本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムが、トランザクション回復（例えば、本明細書で説明される例示的な技法により、ＬＬＡＭＡなどの例示的なシステムを使用してアクセスメソッドによって扱われ得る）をまったく含まないことが可能である。

【００４２】

[0053]本明細書で説明される例示的な技法によれば、例示的なアクセスメソッドが、ユーザー操作に応答して状態を変更することが可能である。例えば、ユーザーが、レコードを作成すること（Ｃ）、読み取ること（Ｒ）、アップデートすること（Ｕ）、または削除すること（Ｄ）（例えば、ＣＲＵＤ操作）を所望する可能性がある。本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムは、これらの操作を直接にはサポートしない可能性がある。むしろ、例示的なアクセスメソッドは、これらの操作を、ＬＬＡＭＡページの状態に対するアップデートとして実施することが可能である。

【００４３】

[0054]例えば、例示的なアクセスメソッド操作の一部である構造変化が存在する可能性もある。例えば、ＢＷ－ＴＲＥＥページ分割には、元のページＯに分割デルタを書き込んで、探索者が、新たなページが現時点で、Ｏの中のキーの部分的範囲に関するデータを包含することを知るようにすることが関与し得る。例えば、これらも、ＬＬＡＭＡページＯに対するアップデートとして扱われ得る。

【００４４】

[0055]本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムが、２つの形態のアップデート、例えば、デルタアップデートおよび置換アップデートをサポートすることが可能である。例えば、アクセスメソッドが、ユーザーの所望に応じて、これらの形態のアップデートを活用することを選択することが可能である。例えば、ＢＷ－ＴＲＥＥが、一連のデルタアップデートを行い、何らかの時点で、それらのデルタアップデートを基本ページに適用することによって、そのページを「統合」し、最適化することを決定することが可能である。例えば、ＢＷ－ＴＲＥＥが、その後、置換アップデートを使用して、その新たな基本ページを生成することが可能である。

【００４５】

[0056]本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムが、本明細書で説明されるアップデート操作および置換操作の全体にわたって、ページの物理ロケーションに関する情報を二次ストレージの中に保持することが可能であり、したがって、システム１００が、本明細書でさらに説明されるとおり、ページがメインメモリーキャッシュからスワップアウトされるような場合にページを再度読み取るために、およびガベージコレクションのために、二次ストレージ・ページ・ロケーション情報を有する。このため、システム１００は、以前のページロケーション、および安定ページ状態情報を記憶していることが可能である。

【００４６】

[0057]例えば、デルタアップデートが、Ｕｐｄａｔｅ－Ｄ（ＰＩＤ，ｉｎ－ｐｔｒ，ｏｕｔ－ｐｔｒ，ｄａｔａ）として示されることが可能である。例えば、デルタアップデートは、ページのそれまでの状態に対する変更を記述するデルタを先頭に付加することが可能である。例えば、ＢＷ－ＴＲＥＥの場合、Ｕｐｄａｔｅ－Ｄに対する「ｄａｔａ」パラメーターは、少なくとも＜ｌｓｎ，ｋｅｙ，ｄａｔａ＞を含むことが可能であり、ここで、ｌｓｎは、冪等性を可能にする。例えば、「ｉｎ－ｐｔｒ」は、ページのそれまでの状態をポイントし、「ｏｕｔ－ｐｔｒ」は、ページの新たな状態をポイントする。

【００４７】

[0058]例えば、置換アップデートが、Ｕｐｄａｔｅ－Ｒ（ＰＩＤ，ｉｎ－ｐｔｒ，ｏｕｔ－ｐｔｒ，ｄａｔａ）として示され得る。例えば、置換アップデートが、ページに関する完全に新たな状態をもたらすことが可能である。Ｕｐｄａｔｅ－Ｄを使用している場合に保存されたそれまでの状態が、「ｄａｔａ」パラメーターによって取って代わられることが可能である。このため、「ｄａｔａ」パラメーターは、デルタが「折り込まれた」ページの状態全体を包含する。

【００４８】

[0059]例えば、「読取り」が、Ｒｅａｄ（ＰＩＤ，ｏｕｔ－ｐｔｒ）として示され得る。例えば、読取りは、「ｏｕｔ－ｐｔｒ」を介して、ページのメインメモリー内のアドレスを戻すことが可能である。そのページがメインメモリーの中にない場合、マッピングテーブルエントリーが、二次ストレージアドレスを包含することが可能である。例えば、その場合、そのページが、メインメモリーの中に読み込まれることが可能であり、マッピングテーブルが、その新たなメインメモリーアドレスでアップデートされることが可能である。

【００４９】

[0060]データ操作をサポートすることに加えて、本明細書で説明される例示的なシステム（例えば、ＬＬＡＭＡ）は、ページの存在、ロケーション、および永続性を管理する操作を提供することが可能である。記憶されたデータの量を調整するのに、アクセスメソッドは、そのアクセスメソッドによって管理されるコレクションにページを足す、またはそれらのコレクションからページを引くことが可能である。状態永続性をもたらすのに、アクセスメソッドは、時々、ページを二次ストレージにフラッシュすることが可能である。この永続性を管理するのに、ページは適切に（例えば、ログシーケンス番号（ｌｓｎ）を用いて）注釈を付けられることが可能である。例えば、ページマネージャーが、ページに対するフラッシュ操作、割当て操作、および解放操作を制御するように構成され得る。

【００５０】

[0061]例えば、フラッシュ操作が、Ｆｌｕｓｈ（ＰＩＤ、ｉｎ－ｐｔｒ，ｏｕｔ－ｐｔｒ，ａｎｎｏｔａｔｉｏｎ）として示され得る。例えば、Ｆｌｕｓｈは、ページ状態をログ構造化ストア（ＬＳＳ）Ｉ／Ｏバッファーに入れるようにコピーすることが可能である。Ｆｌｕｓｈは、デルタ（注釈を伴う）をそれまでの状態の先頭に付加するので、メインメモリーに及ぼす影響においてＵｐｄａｔｅ－Ｄといくぶん似通っていることが可能である。このデルタには、「フラッシュ」というタグが付けられることが可能である。本明細書で説明される例示的な技法によれば、ＬＬＡＭＡなどの例示的なシステムが、ページが位置するＬＳＳ二次ストレージアドレス（フラッシュオフセットと呼ばれる）、およびフ

ラッシュデルタの中の呼び出し元「ａｎｎｏｔａｔｉｏｎ」を記憶することが可能である。例えば、Ｆｌｕｓｈは、戻す際、Ｉ／Ｏバッファーが安定していることをユーザーに保証しない可能性がある。

【００５１】

[0062]例えば、バッファーマネージャーが、ラッチフリーのアップデート操作を介してログ構造化二次ストレージバッファーに対するアップデートを制御するように構成され得る。このため、例えば、複数のスレッドが、ラッチフリーの操作を介してログ構造化二次ストレージバッファーを同時にアップデートすることが可能である。

【００５２】

[0063]例えば、「安定させる」操作が、Ｍｋ－Ｓｔａｂｌｅ（ＬＳＳ　ａｄｄｒｅｓｓ）として示され得る。例えば、Ｍｋ＿Ｓｔａｂｌｅ操作は、ＬＳＳアドレス引数までの、ＬＳＳバッファーにフラッシュされたページが二次ストレージ上で安定していることを確実にすることが可能である。Ｍｋ＿Ｓｔａｂｌｅが戻す際、与えられるＬＳＳ　ａｄｄｒｅｓｓ、およびすべてのより下位のＬＳＳアドレスは、二次ストレージ上で安定していることが確実にされる。

【００５３】

[0064]例えば、「高安定」操作が、Ｈｉ－Ｓｔａｂｌｅ（ｏｕｔ－ＬＳＳ　ａｄｄｒｅｓｓ）として示され得る。例えば、Ｈｉ＿Ｓｔａｂｌｅ操作が、二次ストレージ上で現在、安定している最上位のＬＳＳ　ａｄｄｒｅｓｓを戻すことが可能である。

【００５４】

[0065]例えば、ページマネージャーが、第１のページのページ状態を第２のストレージバッファーに入れるようにコピーすることを開始すること、そのページ状態の先頭にフラッシュデルタレコードを付加することを開始することであって、そのフラッシュデルタレコードは、二次ストレージ内の第１のページのストレージロケーションを示す二次ストレージアドレスと、呼び出し元に関連付けられた注釈とを含むことに基づいて、キャッシュ層ストレージ内の第１のページを二次ストレージ内のロケーションにフラッシュする操作を開始するように構成され得る。

【００５５】

[0066]例えば、バッファーマネージャーが、第１の二次ストレージアドレス引数までの、より下位のアドレスを有する、二次ストレージバッファーにフラッシュされたページが、二次ストレージの中で安定していることを判定するための安定化操作を開始するように構成され得る。

【００５６】

[0067]例えば、「割り当てる」操作が、Ａｌｌｏｃａｔｅ（ｏｕｔ－ＰＩＤ）として示され得る。例えば、Ａｌｌｏｃａｔｅ操作は、マッピングテーブルの中で割り当てられた新たなページのＰＩＤを戻すことが可能である。すべてのそのようなページは、永続的に記憶されていることが可能であり、したがって、Ａｌｌｏｃａｔｅは、含められた操作を自動的にフラッシュすることが可能なシステムトランザクション（後段でさらに説明される）の一部として含められることが可能である。

【００５７】

[0068]例えば、「解放する」操作が、Ｆｒｅｅ（ＰＩＤ）として示され得る。例えば、Ｆｒｅｅ操作は、ＰＩＤによって識別されたマッピングテーブルエントリーを再使用のために利用可能にすることが可能である。メインメモリーの中で、ＰＩＤは、現在のエポック（後段でさらに説明される）に関するＰＩＤに関する待ち状態の解放リスト上に置かれ得る。この場合も、アクティブなページは記憶されていることが可能であるため、Ｆｒｅｅは、システムトランザクションの一部として含められることが可能である。

【００５８】

[0069]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡシステムトランザクションが、構造変更（例えば、ＳＭＯ）に関する相対的な持続性およびアトミック性（すべてまたは無）をもたらすのに使用され得る。例えば、ＬＳＳ、およびＬＳＳのペ

ージ指向型レコードが、「ログレコード」として使用され得る。例えば、トランザクション内のすべての操作が、キャッシュ内のページ状態を変更することに加えて、メモリー内ＬＳＳ　Ｉ／Ｏバッファーに自動的にフラッシュされ得る。例えば、各ＬＳＳエントリーが、厳密に「ページ」ストアである例示的なＬＳＳに関して、ページの状態を含み得る。

【００５９】

[0070]メインメモリーの中で、トランザクション内のすべてのそのような操作が、後段でさらに説明されるとおり、トランザクションコミットまで、隔離されたままに保たれ得る。例えば、コミット時に、トランザクションにおけるすべてのページ変更が、ＬＳＳバッファーに自動的にフラッシュされ得る。例えば、中途終了時に、すべての変更が破棄され得る。例えば、システムトランザクションマネージャーが、トランザクションをコミットすること、およびトランザクションを中途終了することを行うように構成され得る。

【００６０】

[0071]例えば、システムトランザクションが、ＬＬＡＭＡによってサポートされる操作を介して開始されること、および終了されることが可能である。

[0072]例えば、「トランザクション開始」操作が、ＴＢｅｇｉｎ（ｏｕｔ－ＴＩＤ）として示され得る。例えば、トランザクションＩＤ（ＴＩＤ）によって識別されたトランザクションが開始され得る。このことには、ＴＩＤを、例示的なＬＬＡＭＡキャッシュ層（ＣＬ）マネージャーによって保持されるアクティブトランザクションテーブル（ＡＴＴ）に入力することが関与し得る。

【００６１】

[0073]例えば、「トランザクションコミット」操作が、ＴＣｏｍｍｉｔ（ＴＩＤ）として示され得る。例えば、そのトランザクションが、アクティブトランザクションテーブルから取り除かれることが可能であり、そのトランザクションが、コミットされることが可能である。例えば、そのトランザクションにおけるページ状態変更が、マッピングテーブルにインストールされて、ＬＳＳバッファーにフラッシュされることが可能である。

【００６２】

[0074]例えば、「トランザクション中途終了」操作は、ＴＡｂｏｒｔ（ＴＩＤ）として示され得る。例えば、そのトランザクションが、アクティブトランザクションテーブルから取り除かれることが可能であり、変更されたページが、キャッシュの中で「トランザクション開始」にリセットされることが可能であり、変更は、まったくフラッシュされない。

【００６３】

[0075]本明細書で説明される例示的な技法によれば、ＡｌｌｏｃａｔｅおよびＦｒｅｅに加えて、Ｕｐｄａｔｅ－Ｄ操作が、ページ状態を変更することをトランザクション内で許され得る。例えば、Ｕｐｄａｔｅ－Ｒは、後段でさらに説明されるとおり、トランザクションを元に戻すことを複雑にする可能性があるので、使用されないことが可能である。

【００６４】

[0076]本明細書で説明される例示的な技法によれば、トランザクション操作はすべて、入力パラメーター、すなわち、ＴＩＤおよびａｎｎｏｔａｔｉｏｎを有し得る。例えば、ＴＩＤは、キャッシュ内のデルタに追加されることが可能であり、ａｎｎｏｔａｔｉｏｎは、トランザクションにおいてアップデートされた各ページに追加されることが可能である（例えば、ａｎｎｏｔａｔｉｏｎがフラッシュされているかのように）。フラッシュバッファーの中にインストールされ、コミットされる場合、キャッシュ内のすべてのアップデートされたページは、それらのページのロケーションを記述するフラッシュデルタが先頭に付加されることが可能である（例えば、それらのフラッシュデルタが、トランザクションとは無関係にフラッシュされるかのように）。

【００６５】

[0077]ＢＷ－ＴＲＥＥ（例えば、Ｊ．Ｌｅｖａｎｄｏｓｋｉ他、「Ｔｈｅ　Ｂｗ－Ｔｒｅｅ：Ａ　Ｂ－ｔｒｅｅ　ｆｏｒ　Ｎｅｗ　Ｈａｒｄｗａｒｅ　Ｐｌａｔｆｏｒｍｓ（Ｂｗ－Ｔｒｅｅ：新しいハードウェアプラットフォームのためのＢツリー）」、２９ｔｈ

ＩＥＥＥ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｄａｔａ Ｅｎｇｉｎｅｅｒｉｎｇ（ＩＣＤＥ ２０１３）、２０１３年４月８～１１日参照）が、ユーザートランザクションがサポートされる（例えば、トランザクションコンポーネント２０２のために）ことを可能にし得る例示的なキー・バリューストアをもたらすことが可能である。例えば、ＢＷ－ＴＲＥＥは、ＬＳＮを管理し、先行書込みログ（ＷＡＬ）プロトコルを執行し、ＤＥＵＴＥＲＯＮＯＭＹデータコンポーネント（ＤＣ）（Ｊ．Ｌｅｖａｎｄｏｓｋｉ他、「Ｄｅｕｔｅｒｏｎｏｍｙ：Ｔｒａｎｓａｃｔｉｏｎ Ｓｕｐｐｏｒｔ ｆｏｒ Ｃｌｏｕｄ Ｄａｔａ（Ｄｅｕｔｅｒｏｎｏｍｙ：クラウドデータのためのトランザクションサポート）」、Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｉｎｎｏｖａｔｉｖｅ Ｄａｔａ Ｓｙｓｔｅｍｓ Ｒｅｓｅａｒｃｈ（ＣＩＤＲ）（２０１１年１月）、１２３～１３３ページ、およびＤ．Ｌｏｍｅｔ他、「Ｕｎｂｕｎｄｌｉｎｇ Ｔｒａｎｓａｃｔｉｏｎ Ｓｅｒｖｉｃｅｓ ｉｎ ｔｈｅ Ｃｌｏｕｄ（クラウドにおけるトランザクションサービスのアンバンドリング）」、Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｉｎｎｏｖａｔｉｖｅ Ｄａｔａ Ｓｙｓｔｅｍｓ Ｒｅｓｅａｒｃｈ（ＣＩＤＲ）、２００９年参照）によって予期されるとおり、チェックポインティング要求に応答することが可能である。本明細書の説明は、ＢＷ－ＴＲＥＥが、ＬＬＡＭＡなどの例示的なシステムを使用している場合に、どのように前述のことを実現するかを扱うことを含む。

【００６６】

[0078]Ｕｐｄａｔｅ－ＤおよびＵｐｄａｔｅ－Ｒ ＬＬＡＭＡ操作に対する「データ」コンテンツは、キー、ＬＳＮ、およびキー・バリューストアの「データ部分」を含み得る。例えば、ＢＷ－ＴＲＥＥは、このため、これらの操作を介して、キー・バリューストアを実施し、ＬＳＮを介して冪等性をもたらし、Ｕｐｄａｔｅ－Ｄを介してインクリメンタルアップデートを実行し、Ｕｐｄａｔｅ－Ｒを介してＢＷ－ＴＲＥＥのページ統合を実行し、ＬＬＡＭＡ Ｒｅａｄ操作またはＦｌｕｓｈ操作を使用して読取りまたは書込みのためにページにアクセスすることが可能である。例えば、システムは、アップデートデルタレコード操作および置換アップデート操作に基づいてアップデートを制御するように構成され得るレコードマネージャーを含み得る。

【００６７】

[0079]例えば、アクセスメソッドが、そのアクセスメソッドがアップデート操作を介してＬＬＡＭＡにもたらすデータの中にＬＳＮを記憶することが可能である。さらに、フラッシュデルタの中に記憶されたＦｌｕｓｈ操作ａｎｎｏｔａｔｉｏｎパラメーターが、ページコンテンツを記述するさらなる情報をもたらすことが可能である。例えば、これらは、ＢＷ－ＴＲＥＥが先行書込みロギング（ＷＡＬ）を執行することを可能にし得る。例えば、安定させる操作（例えば、Ｍｋ－Ｓｔａｂｌｅ）が、ページをフラッシュした後、トランザクションログチェックポインティングのためにアップデートを安定させることが可能である。

【００６８】

[0080]例えば、Ａｌｌｏｃａｔｅ操作およびＦｒｅｅ操作が、例示的なＢＷ－ＴＲＥＥ実施形態がその実施形態のツリーを成長させること、または縮小することを可能にし得る。例えば、ＢｅｇｉｎＴｒａｎｓ（例えば、ＴＢｅｇｉｎ）およびコミット／中途終了（例えば、ＴＣｏｍｍｉｔ／ＴＡｂｏｒｔ）が、構造変更操作（ＳＭＯ）を実行している際に予期されるアトミック性を可能にし得る。

【００６９】

[0081]例えば、アップデート操作（例えば、Ｕｐｄａｔｅ－Ｄ／Ｕｐｄａｔｅ－Ｒ）は、「ユーザーレベル」データに限定されないことが可能である。例えば、ＢＷ－ＴＲＥＥが、Ｕｐｄａｔｅ－Ｄを使用して、システムトランザクションに関して、後段でさらに説明されるとおり、ＳＭＯを実施している際にＢＷ－ＴＲＥＥの「マージ」デルタおよび「分割」デルタを書き込むことが可能である。

【００７０】

[0082]本明細書で説明される例示的な技法によれば、キャッシュ層データ操作に関して

、ページアップデートは、図４に示されるとおり、デルタアップデートであれ、置換アップデート（例えば、図７に関連して後段でさらに説明される）であれ、コンペア・アンド・スワップ（ＣＡＳ）操作を使用してマッピングテーブル３０４の中に新たなページ状態ポインター４０２をインストールすることによって達せられることが可能である。例えば、置換アップデート（例えば、Ｕｐｄａｔｅ－Ｒ（ＰＩＤ，ｉｎ－ｐｔｒ，ｏｕｔ－ｐｔｒ，ｄａｔａ））が、所望される新たな状態と、ＬＳＳ内のそのページのそれまでの状態のロケーションの両方を含み得る。例えば、新たなアップデートデルタ４０４（例えば、Ｕｐｄａｔｅ－Ｄ（ＰＩＤ，ｉｎ－ｐｔｒ，ｏｕｔ－ｐｔｒ，ｄａｔａ））が、このＬＳＳロケーションを既に含む、ページ３０２のそれまでの状態４０６をポイントする。

【００７１】

[0083]例えば、そのようなラッチフリーアプローチは、ラッチすることによって生じる遅延を回避することが可能であるが、「楽観的」同時実行制御メソッドと同様に、そのアプローチ独自のペナルティを受ける可能性があり、すなわち、ＣＡＳが失敗する可能性があり、その場合、アップデートは、再試行される。例えば、ＣＡＳの操作を適宜、再試行することは、障害が生じた場合に例示的なＬＬＡＭＡ実施形態が示し得るとおり、例示的なＬＬＡＭＡユーザーに任せられる可能性がある。

【００７２】

[0084]本明細書で説明される例示的な技法によれば、データがキャッシュ（例えば、３１２）内にある場合、ブロックする操作は存在しない可能性があるが、二次ストレージからページを読み取ることには、キャッシュ内にページが出現するのを待つことが関与する可能性がある。マッピングテーブル（例えば、マッピングテーブル３０４）が、前述したとおり、キャッシュされたページに関してさえ、ＬＳＳページをポイントして、ページが、効果的なキャッシュ管理のためにキャッシュとＬＳＳの間で移動されることを可能にする。

【００７３】

[0085]本明細書で説明される例示的な技法によれば、ページがフラッシュされる際、例示的なＬＬＡＭＡ実施形態が、キャッシュ（例えば、３１２）内で代表されているものが、ＬＳＳ（例えば、３１４）内にあるものと合致することを確実にすることが可能である。このため、フラッシュデルタは、ＰＩＤとＬＳＳオフセットの両方をフラッシュデルタの中に含めることが可能であり、そのデルタを、ページ３０２の先頭に付加することによってＬＳＳバッファーおよびキャッシュ（例えば、３１２）の中に含めることが可能である。

【００７４】

[0086]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態は、デルタアップデートをサポートし得るため、ページ状態は、不連続の断片を含み得る。この特徴をフラッシュする活動と組み合わせることにより、最近のアップデートがキャッシュ内だけに存在し得る一方で、キャッシュ内ページがページの状態の一部をＬＳＳの中に有する（以前にフラッシュされていて）ことがもたらされ得る。このことが生じると、次のフラッシュのストレージ費用を低減することが可能であり得る。

【００７５】

[0087]このため、例示的なＬＬＡＭＡ実施形態が、以前のフラッシュ以来の変更だけを含むデルタを書き込むことによって、そのようなページをフラッシュすることが可能である。例えば、キャッシュ内の複数のアップデートデルタがすべて、隣接する形態のデルタ（本明細書で「Ｃデルタ」と呼ばれ得る）を、ＬＳＳ内のページのリマインダーに対するポインターと一緒に書き込むことによって、フラッシュするために隣接するようにされることが可能である。このため、そのページ全体が、ＬＳＳ内で、ただし、場合により、いくつかの断片で、アクセス可能であり得る。

【００７６】

[0088]本明細書で説明される例示的な技法によれば、Ｆｌｕｓｈ操作は、このようにしてときとともにフラッシュされてきたいくつかの部分を有し得るキャッシュされたページ

状態を見ることが可能であり、別々の断片およびそれらの断片のＬＳＳアドレスが代表された、キャッシュされたページをもたらす。本明細書で説明される例示的な技法によれば、任意の時点で、Ｆｌｕｓｈが、不連続なページ断片のコンテンツを隣接して（かつ冗長に）書き込むことによって、ＬＳＳストレージの中にこれらの断片を一緒にもたらすことが可能である。例えば、ユーザーが、異なる読取りアクセス費用のため、ＬＳＳがディスクストレージを使用する場合、隣接性を所望する一方で、ＬＳＳがフラッシュストレージを使用する場合、断片を別々のままにしておくことに積極的である可能性がある。

【００７７】

[0089]本明細書で説明される技法によれば、ページがフラッシュされる際、システムが、そのフラッシュに先立って、ページのどのような状態がフラッシュされているかを知ることが望ましい可能性がある。例えば、このことは、システムが、ページを単にラッチし、フラッシュを実行することが可能であるので、ラッチを使用して容易に確かめられ得る。しかし、ラッチフリーのアプローチにおいて、システムは、ページに対するアップデートがフラッシュされることを、そのページがフラッシュされることに先立って、防止することに相当な困難を有し得る。例えば、このことは、先行書込みログプロトコルの執行の際、またはそのフラッシュが構造変更の一環として行われる場合に、問題をもたらし得る。例えば、不適切なフラッシュが、それらのフラッシュのＣＡＳを実行する際、失敗することが望ましい可能性がある。このため、本明細書で説明される例示的な技法によれば、ＣＡＳの中にフラッシュされるべきページ状態に対するポインターが使用されることが可能であり、そのポインターは、その場合、その特定の状態をキャプチャするに過ぎないことが可能であり、フラッシュが完了する前に状態がアップデートされている場合、失敗することが可能である。しかし、このことは、他の問題を生じる可能性がある。

【００７８】

[0090]本明細書で説明される例示的な技法を研究する際、キャッシュ管理を実行する場合、およびＬＳＳにページをフラッシュする場合に有利であり得る強い不変条件(strong invariant)の種類を決定することに困難を経験した。例えば、不変条件は、以下のような特性を含み得る。すなわち、

ＬＳＳに正常にフラッシュされたページは、フラッシュされたものとしてキャッシュ内で即時に見られ、そのページのフラッシュされた状態は、すべての後の状態のフラッシュに先行してＬＳＳ　Ｉ／Ｏバッファーの中に入っている。フラッシュが失敗したページは、キャッシュ内でフラッシュされたものとして出現せず、ＬＳＳを見る際、そのフラッシュが成功しなかったことが明らかとなる。

【００７９】

[0091]例えば、２つの代替のアプローチが、以下を含み得る。すなわち、

１）フラッシュの成功が、ＣＡＳをまず実行することによって確実にされ得る。ＣＡＳが成功すると、ページが、ＬＳＳに書き込まれ得る。例えば、前述のことが行われた場合、競合条件が、信頼できるＬＳＳ回復を損なう可能性がある。例えば、より早期のフラッシュに依存するページが、その後、フラッシュされる可能性があり、ただし、この「後の」フラッシュは、システムクラッシュより前にＬＳＳに書き込まれることに成功する一方で、「より早期の」フラッシュは、完了するのが遅すぎ、安定したＬＳＳ内に出現しない。この状況は、ある種の因果関係を損なう可能性がある。

２）フラッシュされることが所望されるページのページ状態が、キャプチャされ、ＬＳＳバッファーに書き込まれることが可能である。次に、ＣＡＳが試みられることが可能であり、ＣＡＳが失敗することが可能である。このため、ページは、システムがクラッシュした場合にそのフラッシュが成功したか、または失敗したかを区別するための指示をまったく伴わずに、ＬＳＳに書き込まれる。例えば、様々な時点でＬＳＳに書き込まれた複数のそのようなページが存在し得る。例えば、失敗したＣＡＳよりも早期にＬＳＳ内に出現するページの後の状態が書き込まれ得る。前述したとおり、その状態は、より後に始まったが、より早期のフラッシュより前にバッファースロットを獲得している。

【００８０】

[0092]本明細書で説明される例示的な技法によれば、前述したジレンマが、後段で説明されるとおり解決され得る。例えば、ＣＡＳが十分に早期に実行された場合、ページの状態をログバッファーにコピーすることに先立って、そのフラッシュが成功するか否かが判定され得る。このため、例示的なフラッシュ手順が、以下のとおり実行され得る。すなわち、
ステップ１：フラッシュされることが意図されるページの状態を識別する
ステップ２：その状態を書き込むべきＬＳＳバッファー内のスペースを差し押さえる
ステップ３：ＣＡＳを実行して、そのフラッシュが成功するかどうかを判定する。このことを行うためにフラッシュデルタにおけるＬＳＳオフセットが獲得される（前述のステップ２において規定されるとおり）
ステップ４：ステップ３が成功した場合、保存されるべき状態をＬＳＳに書き込む。この状態がＬＳＳに書き込まれている間、本明細書で説明される例示的なＬＬＡＭＡ技法が、バッファーがＬＳＳ二次ストレージに書き込まれるのを防止することが可能である。
ステップ５：ステップ３が失敗した場合、「失敗したフラッシュ」を示す指示をバッファー内の確保されたスペースに書き込む。このことは、ストレージを消費し得るが、いずれのフラッシュが成功したか、または失敗したかについての曖昧さを解決する。

【００８１】

[0093]この例示的な手順の結果は、ＬＳＳが、回復中、失敗したＣＡＳの結果であるページを見ない可能性があることである。例えば、このことは、ＬＳＳ内に後に出現する（「ログ」におけるページの位置の点で）ページが、ＬＳＳログにおいてそのページのより早期のすべてのインスタンスよりも、そのページの後の状態であるという特性を保つこともする。

【００８２】

[0094]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、その実施形態のメモリー制約を満たすようにキャッシュ・アンド・スワップアウト・データを管理することが望ましい可能性がある。例えば、例示的なＬＬＡＭＡ実施形態は、デルタアップデート、置換アップデート、およびフラッシュを意識していることが可能であり、以上のそれぞれを認識することが可能である。しかし、例示的なＬＬＡＭＡ実施形態は、汎用であるものとされる場合、ページのコンテンツについて何も知らない。このため、例示的なＬＬＡＭＡ実施形態は、ページの中にＬＳＮを保持することによって、アクセスメソッド層がトランザクションをサポートしているかどうかを意識していない。このため、もたらされ得る問題は、例示的なＬＬＡＭＡ実施形態が、ＬＳＮを見ることなく、先行書込みログプロトコルを執行する可能性がある場合、どのようにキャッシュスペース管理（ページを強制退去させることを含む）を提供し得るかに関する潜在的な疑問を含む。

【００８３】

[0095]本明細書で説明される例示的な技法によれば、既にフラッシュされているデータは、キャッシュからドロップされ得る。例えば、ページのアップデート・インプレースが行われるシステムが、最近、アップデートされた、「汚れた」ページをスワップアウトすること（キャッシュからドロップすること）を防止され得る。しかし、デルタアップデートのため、例示的なＬＬＡＭＡ実施形態は、ページのいずれの部分が既にフラッシュされているかを判定することが可能である。例えば、それぞれのそのような部分が、フラッシュデルタを用いて記述されることが可能であり、それらのフラッシュされた部分が、キャッシュから「スワップアウト」されることが可能である。

【００８４】

[0096]ページの一部を「スワップアウト」する際、ストレージを単に割当て解除し、再使用することは、スワップアウトされた部分に対するダングリングリファレンスを残し得るので、望ましくない可能性がある。このため、本明細書で説明される例示的な技法によれば、ページのいずれの部分がスワップアウトされているかを記述するデルタが使用され得る。

【００８５】

[0097]例えば、完全にスワップアウトされたページに関して、マッピングテーブル３０４内のそのページのメインメモリーアドレスが、そのページの最新のフラッシュデルタからのＬＳＳポインターで置き換えられることが可能である。

【００８６】

[0098]図５は、例示的な部分的ページスワップアウト、および例示的な部分的スワップデルタを示す。例えば、部分的にスワップアウトされたページに関して、ＣＡＳが、「部分的スワップ」デルタレコード５０２を挿入するのに使用され得る。例えば、このデルタレコード５０２は、そのページが部分的にスワップアウトされている（例えば、したがって、そのページのいずれの部分も正常にアクセスされ得ない）ことを示すことが可能であり、そのページの欠落した部分５０６を探し出すためのＬＳＳ内のロケーション情報を示すフラッシュデルタレコード５０４をポイントすることが可能である。例えば、「部分的スワップ」デルタ５０２がＣＡＳを用いてインストールされると、ドロップされているページの部分に関するメモリーは、後段でさらに説明されるとおり、例示的なエポック機構を使用して解放され得る。

【００８７】

[0099]例えば、ページマネージャーが、第１のページに関連付けられたページ状態の先頭に部分的スワップデルタレコードを付加することであって、その部分的スワップデルタレコードは、第１のページの欠落した部分の二次ストレージ内のロケーションを示すフラッシュデルタレコードのストレージロケーションを示す二次ストレージアドレスを含むことに基づいて、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページの一部分をスワップする操作を開始するように構成され得る。

【００８８】

[0100]例えば、ページマネージャーは、エポック機構を使用して、第１のページのその部分に関連付けられたキャッシュ層ストレージに関する解放操作を開始するようにさらに構成され得る。

【００８９】

[0101]本明細書で説明される例示的な技法によれば、このアプローチは、いくつかの有用な特徴をユーザーに有利に提供することが可能である。例えば、そのような例示的なＬＬＡＭＡ実施形態のキャッシュ層（例えば、３１２）が、ページの実際のコンテンツに関する知識なしにメモリーを取り戻すことが可能である。例えば、フラッシュされたページ、およびページのフラッシュされた部分をドロップすることに、Ｉ／Ｏ操作がまったく関与しないことが可能である。例えば、部分的にフラッシュされたページをメインメモリーの中に戻すことに、ＬＳＳの中に複数の部分を有する完全にフラッシュされたページに関する場合よりも少ない回数のＬＳＳ読取りが関与することが可能である。

【００９０】

[0102]例えば、いくつかの例示的なキャッシュ管理戦略が、キャッシュストレージを管理するのに使用され得る（例えば、最も長く使われていない（ＬＲＵ）、ＬＲＵ（ｋ）、Ｃｌｏｃｋなど（例えば、Ｗ．Ｅｆｆｅｌｓｂｅｒｇ他、「Ｐｒｉｎｃｉｐｌｅｓ ｏｆ ｄａｔａｂａｓｅ ｂｕｆｆｅｒ ｍａｎａｇｅｍｅｎｔ（データベースバッファー管理の原理）」、ＡＣＭ Ｔｒａｎｓａｃｔｉｏｎｓ ｏｎ Ｄａｔａｂａｓｅ Ｓｙｓｔｅｍｓ（ＴＯＤＳ）、Ｖｏｌ．９、Ｉｓｓｕｅ ４（１９８４年１２月）、５６０～５９５ページ、およびＥ．Ｏ’Ｎｅｉｌ他、「Ｔｈｅ ＬＲＵ－Ｋ ｐａｇｅ ｒｅｐｌａｃｅｍｅｎｔ ａｌｇｏｒｉｔｈｍ ｆｏｒ ｄａｔａｂａｓｅ ｄｉｓｋ ｂｕｆｆｅｒｉｎｇ（データベースディスクバッファリングのためのＬＲＵ－Ｋページ交換）」、Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ｔｈｅ １９９３ ＡＣＭ ＳＩＧＭＯＤ Ｉｎｔｅｒｎａｔｉｏｎａｌ Ｃｏｎｆｅｒｅｎｃｅ ｏｎ Ｍａｎａｇｅｍｅｎｔ ｏｆ Ｄａｔａ（ＳＩＧＭＯＤ ’９３）、２９７～３０６ページ参照））。これらの例は、さらなる帳簿処理（bookkeeping）を含み得るが、実質的な困難はもたらさないことが可能である。

【００９１】

[0103]本明細書で説明される例示的な技法によれば、そのような例示的なラッチフリー

のアプローチを使用して、操作は、ページとページ状態の両方を、ページとページ状態の両方が「ガベージ」と指定された後でさえ、検査していることが可能である。例えば、従来の「ラッチ」を使用していない場合、システムは、１）ページ状態全体を置き換え、それまでの状態を、別の操作が読取っている間に割当て解除するＵｐｄａｔｅ－Ｒ操作、または２）マッピングテーブル内のページを、別の操作が検査している間に「解放」するＤｅ－ａｌｌｏｃａｔｅ操作のいずれかを防止することに失敗する可能性がある。

【００９２】

[0104]本明細書で説明される例示的な技法によれば、ストレージもＰＩＤも、別の操作がストレージにもＰＩＤにもアクセスしている可能性がまったくなくなるまで、再使用されることを許されないことが可能である。このため、「解放されたリソース」と「再使用可能なリソース」の間の区別が確立され得る。例えば、「解放されたリソース」が、操作によってガベージと指定されている。例えば、「再使用可能なリソース」が、解放されており、他のいずれの操作によってもアクセス可能でないことが確実にされることが可能である。例えば、エポックが、割当て解除されたオブジェクトがあまりにも早く再使用されるのを防ぐのに使用され得る（例えば、Ｈ．Ｋｕｎｇ他、「Ｃｏｎｃｕｒｒｅｎｔ ｍａｎｉｐｕｌａｔｉｏｎ ｏｆ ｂｉｎａｒｙ ｓｅａｒｃｈ ｔｒｅｅｓ（バイナリーサーチツリーの同時操作）」、ＡＣＭ Ｔｒａｎｓａｃｔｉｏｎｓ ｏｎ Ｄａｔａｂａｓｅ Ｓｙｓｔｅｍｓ（ＴＯＤＳ）、Ｖｏｌ．５、Ｉｓｓｕｅ ３（１９８０年９月）、３５４～３８２ページ参照）。

【００９３】

[0105]本明細書で説明される例示的な技法によれば、すべての操作は、ＰＩＤまたはページ状態にアクセスするのに先立って、現在のエポックＥに登録することが可能であり、そのようなアクセスが完了すると、Ｅを抜けることが可能である。例えば、操作が、解放されたリソースを、現在のエポックのリスト上に常に書き込むことが可能であり、現在のエポックは、Ｅ（その操作が参加したエポック）であること、または現在のエポックが先に進んでいる場合、後のエポックであることが可能である。例えば、Ｅのリスト上のいずれのリソースも、Ｅに登録されたすべての操作が抜けるまで、再使用されないことが可能である。

【００９４】

[0106]例えば、エポックに番号が付けられることが可能であり、時々、新たなエポックＥ＋１が「現在の」エポックになることが可能である。このため、新たな操作が、今やＥ＋１である現在のエポックに引き続き登録することが可能である。例えば、エポック機構不変条件は、以下のとおりである。すなわち、エポックＥ＋１または後のエポックにおけるいずれの操作も、エポックＥにおいて解放されたリソースを見ていること、および使用していることはあり得ない。

【００９５】

[0107]このため、この不変条件に基づいて、すべての操作がＥを抜けると、Ｅにおいて解放されたリソースにアクセスすることができるアクティブな操作は存在しない。図６は、例示的な２つのエポック６０２、６０４、およびそれらのエポック６０２、６０４のそれぞれのガベージコレクションリスト６０６、６０８を示す。図６に示されるとおり、ガベージコレクション項目６１０が、エポック１（６０２）における「スレッド１」に関連付けられ、ガベージコレクション項目６１２が、エポック１（６０２）における「スレッド２」に関連付けられ、ガベージコレクション項目６１４が、エポック２（６０４）における「スレッド３」に関連付けられる。図６に示されるとおり、エポック２（６０４）のガベージコレクションリスト６０８内のガベージコレクション項目６１６が、エポック１（６０２）の「スレッド１」に関連付けられる。

【００９６】

[0108]例えば、「スレッド１」および「スレッド２」がエポック１（６０２）を抜けると、エポック１（６０２）において解放されたリソース（例えば、ガベージコレクション項目６１０およびガベージコレクション項目６１２）にアクセスすることができるアクテ

ィブな操作は存在しない。

【００９７】

[0109]例えば、第１のエポックマネージャーが、第１のプロセッサー操作によってページ情報にアクセスすることに先立って、第１のエポック登録リスト内に第１のプロセッサー操作を登録することを開始するように構成され得る。

【００９８】

[0110]第１のエポックマネージャーは、第１のプロセッサー操作によって解放された１つまたは複数のリソースを、第１のエポックガベージコレクションリスト内に書き込むように構成され得る。第１のエポックマネージャーは、第１のエポック登録リストが、現在、登録されているプロセッサー操作を含むようになるまで、第１のエポックガベージコレクションリスト内に書き込まれた、書き込まれたリソースの再使用をブロックすることが可能である。

【００９９】

[0111]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、ログ構造化ファイルシステム（ＬＦＳ）と似通ったログ構造化の様態で二次ストレージ（例えば、フラッシュストレージ）上のデータを編成することが可能である（例えば、Ｍ．Ｒｏｓｅｎｂｌｕｍ他、「Ｔｈｅ　Ｄｅｓｉｇｎ　ａｎｄ　Ｉｍｐｌｅｍｅｎｔａｔｉｏｎ　ｏｆ　ａ　Ｌｏｇ－Ｓｔｒｕｃｔｕｒｅｄ　Ｆｉｌｅ　Ｓｙｓｔｅｍ（ログ構造化ファイルシステムの設計および実施）」、ＡＣＭ　Ｔｒａｎｓａｃｔｉｏｎｓ　ｏｎ　Ｃｏｍｐｕｔｅｒ　Ｓｙｓｔｅｍｓ（ＴＯＣＳ）、Ｖｏｌ．１０、Ｉｓｓｕｅ　１、１９９２年２月、２６～５２ページ参照）。このため、各ページフラッシュは、フラッシュ上のページの位置を再配置する。例えば、このことが、本明細書で説明される例示的なマッピングテーブル３０４を使用するさらなる理由を与えることが可能である。例えば、ログ構造化ストレージが、１ページ当たりの書込みの回数を有利に減らし、それらの書込みを「順次」にすることが可能である。このため、多くのランダムな書込みが、１回の大きなマルチページ書込みに変換され得る。

【０１００】

[0112]前述したとおり、「論理ページ」が、基本ページと、そのページに対するアップデートを示す０または１つ以上のデルタレコードとを含むことが可能であり、これにより、ページが、フラッシュされる際に断片でフラッシュに書き込まれることを可能にする。このため、フラッシュ上の論理ページは、潜在的に、ポインターとしてファイルオフセットを使用して一緒にリンクされた異なる物理デバイスブロック上にあるレコードに対応することが可能である。さらに、物理ブロックが、複数の論理ページからのレコードを含み得る。図７ａは、フラッシュ３１４上の例示的なログ構造化ストレージ編成７００ａを示す。

【０１０１】

[0113]例えば、論理ページが、フラッシュ上の連鎖の先頭（順次ログ７０２内のオフセットがマッピングテーブル３０４から獲得され得る）から開始して、リンクされたレコードをたどることによって、フラッシュ３１４からメモリー（例えば、ＲＡＭ３１２）の中に読み込まれ得る。例えば、フラッシュ３１４からメモリー３１２の中に対応する「論理ページ」を読み込むために、オフセット７０４が、デルタレコード７０６にアクセスするためにマッピングテーブル３０４から獲得されて、現在の状態、および基本ページ７０８が獲得されることが可能である。

【０１０２】

[0114]例えば、フラッシュ３１４から対応する「論理ページ」をメモリー３１２の中に読み込むために、オフセット７１０が、デルタレコード７１２にアクセスするためにマッピングテーブル３０４から獲得されて、デルタおよびリンクが獲得され、第２のデルタレコード７１４にアクセスが行われ、その後、基本ページ７１６にアクセスが行われることが可能である。

【０１０３】

[0115]例えば、フラッシュプロセスが、同一の論理ページの複数のデルタレコードを、それらのデルタレコードが一緒にフラッシュされる場合に有利に統合して、フラッシュ上の隣接するＣデルタにすることが可能である。さらに、論理ページが、メモリー内で統合された後にフラッシュされる場合、フラッシュ上で統合されることが可能であり、このことは、ページ読取りパフォーマンスを有利に向上させ得る。

【０１０４】

[0116]図７ｂは、第１のストレージオブジェクト７４６（例えば、図７ｂにおいて、複数の以前に先頭に付加されたデルタレコードを有する基本ページ７４２を含む）の物理アドレスを、ページ７４２の新たな状態７４４（例えば、ページ７４２を、以前に先頭に付加されたデルタレコードと統合することからもたらされる）の物理アドレスで置き換えることに基づいて、ページ７４２のそれまでの状態７４０をページ７４２の新たな状態で置き換えることを示す例示的なマッピングテーブル３０４を示す。

【０１０５】

[0117]例えば、図７ｃに示されるとおり、ページ７４２のそれまでの状態７４０をページ７４２の新たな状態７４４で置き換えることは、複数のデルタレコードを統合して、隣接するＣデルタ７５０にすることを含むことが可能であり、そのＣデルタ７５０が、次に、基本ページ７４２と一緒にフラッシュされ得る。

【０１０６】

[0118]例えば、ページ７４２のそれまでの状態７４０をページ７４２の新たな状態７４４で置き換えることは、マッピングテーブル３０４上のアトミック・コンペア・アンド・スワップ操作を介して、現在のページ７４２の変更されたバージョンを生成すること、または現在のページ７４２を置き換えるための別のページを決定すること、および現在のページ７４２の物理アドレスを、ページ７４２の新たな状態７４４（例えば、置換のための変更されたバージョンまたその他のページ）の物理アドレスで置き換えることを含み得る。

【０１０７】

[0119]例えば、図７ｂの特徴と図７ｃの特徴の間の区別として、二次ストレージにページを書き込む際、ＬＬＡＭＡは、図７ｃに示される統合を実行することが可能であるが、図７ｂの統合を実行することは、Ｕｐｄａｔｅ－Ｒを実行しているアクセスメソッドに依存する。

【０１０８】

[0120]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、完全にラッチフリーであり得る。さらに、専用のスレッドは、Ｉ／Ｏバッファーをフラッシュするのに使用されないことが可能である。というのは、そうすることが、スレッド作業負荷をバランスがとれた状態に保つことを複雑にする可能性があるからである。このため、すべてのスレッドが、このバッファーを管理することに参加することが可能である。例えば、従来のアプローチは、ラッチを使用してきた。しかし、そのような従来の技法は、バッファー内のスペースを割り当てている間、ラッチすることに限られて、データ転送に先立ってそのラッチを解放する可能性があり、データ転送は、その後、並行に進められる可能性がある。

【０１０９】

[0121]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、本明細書で説明される例示的なシステムの別の場所で行われるとおり、アトミック性のためにＣＡＳを代わりに使用して、バッファースペース割当てのための従来のラッチを回避することが可能である。例えば、このことには、ＣＡＳが実行される状態を定義することが関与する。例えば、バッファー状態の不変の部分は、その部分のアドレス（Ｂａｓｅ）と、サイズ（Ｂｓｉｚｅ）とを含み得る。例えば、バッファー内で使用されるストレージの現在の最高水位線が、Ｂａｓｅを基準としたＯｆｆｓｅｔを用いて追跡され得る。例えば、バッファーの使用を求める各要求が、ページフラッシュのためのスペースＳｉｚｅを確保しようとする作業から始まることが可能である。

【０１１０】
[0122]本明細書で説明される例示的な技法によれば、バッファー内でスペースを確保するのに、スレッドが、現在のＯｆｆｓｅｔを獲得し、Ｏｆｆｓｅｔ＋Ｓｉｚｅを計算することが可能である。例えば、Ｏｆｆｓｅｔ＋Ｓｉｚｅ≦Ｂｓｉｚｅの場合、要求はバッファー内に記憶され得る。例えば、スレッドは、比較値としての現在のＯｆｆｓｅｔ、および新たな値としてのＯｆｆｓｅｔ＋Ｓｉｚｅと一緒にＣＡＳを発行することが可能である。そのＣＡＳが成功した場合、Ｏｆｆｓｅｔは、その新たな値に設定されることが可能であり、そのスペースは、確保されることが可能であり、バッファー書込みは、データをバッファーに転送することが可能である。
【０１１１】
[0123]本明細書で説明される例示的な技法によれば、このロジックは、バッファー内のスペース割当てを扱うことが可能である。例えば、バッファーに書き込むこと、および複数のバッファーを管理することには、後段でさらに説明される、ＣＡＳ状態におけるより多くのことが関与する可能性がある。
【０１１２】
[0124]二次ストレージにバッファーを書き込む際、Ｏｆｆｓｅｔ＋Ｓｉｚｅ＞Ｂｓｉｚｅである場合、スレッドのレコードを保持するのにバッファー内に不十分なスペースしか存在しない。この場合、スレッドは、バッファーを封印して、これにより、バッファーを、もはや使用されるべきでないものにし、二次ストレージに書き込まれるべく準備されたものにすることが可能である。この条件は、フラッシュバッファー状態における「Ｓｅａｌｅｄ」ビットを用いて追跡され得る。例えば、ＣＡＳが、「Ｓｅａｌｅｄ」ビットをＦ（例えば、偽）からＴ（例えば、真）に変更することが可能である。例えば、封印されたバッファーが、もはやアップデートされないことが可能であり、封印されたバッファーに出会ったスレッドは、異なる（封印されていない）バッファーを探す。
【０１１３】
[0125]本明細書で説明される例示的な技法によれば、封印されたバッファーは、新たなアップデート要求をもはや受け付けないことが可能である。しかし、例示的なシステムは、バッファースペースを獲得することにすべて成功しているそれまでの書込み側が、書込み側のデータをバッファーに転送することを終えていることをまだ保証されていない可能性がある。本明細書で説明される例示的な技法によれば、「Ａｃｔｉｖｅ」カウントが、バッファーにデータを転送している書込み側の数を示すことが可能である。例えば、バッファー内にスペースを確保する際、書込み側のＣＡＳは、Ｏｆｆｓｅｔ、Ｓｅａｌｅｄ、およびＡｃｔｉｖｅを表す値を含み得る。例えば、書込み側のＣＡＳが、この構造を獲得し、そのＣＡＳのペイロードサイズをＯｆｆｓｅｔに追加し、「Ａｃｔｉｖｅ」を１だけインクリメントすることが可能であり、～Ｓｅａｌｅｄである場合、この状態をアップデートして、スペースを確保するようにＣＡＳを実行することが可能である。例えば、書込み側が終了している場合、書込み側は、この状態を再獲得し、「Ａｃｔｉｖｅ」を１だけデクリメントすることが可能であり、その変更を生じさせるようにＣＡＳを実行することが可能である。例えば、操作は、失敗した場合、必要に応じて再び行われ得る。
【０１１４】
[0126]例えば、バッファーは、Ｓｅａｌｅｄであり、かつＡｃｔｉｖｅ＝０である場合、フラッシュ可能であり得る。例えば、この条件をもたらす書込み側が、Ｉ／Ｏを開始することを担うことが可能である。例えば、Ｉ／Ｏが完了すると、バッファーのＯｆｆｓｅｔおよびＡｃｔｉｖｅユーザーがともに０に設定されることが可能であり、バッファーは、ｕｎＳｅａｌｅｄであり得る。
【０１１５】
[0127]本明細書で説明される例示的な技法によれば、複数のバッファーに関して、複数のバッファーのセットの中のそれらのバッファーの各々が、前述した状態を有する。図８は、例示的な完全なフラッシュバッファー状態８００を示す。図８の例に示されるとおり、バッファー当たりの状態８０２が、３２ビットを含むことが可能であり、次の書込みの

ためのオフセットに関する２４ビット８０４と、アクティブな書込み側の数に関する７ビット８０６と、「封印ビット」標識のための１ビット８０８（例えば、封印されたバッファーを示す）とを含む。例えば、現在のアクティブなバッファー番号（ＣＵＲＲＥＮＴ）８１０が、現在、アクティブなバッファーを示すことが可能である（例えば、図示されるとおり、８ビットに関する）。

【０１１６】

[0128]例えば、バッファーは、ラウンドロビンスタイルでアクセスされて、使用されることが可能であり、したがって、１つのバッファーが封印されるにつれ（封印ビット標識８０８によって示される）、本明細書の例示的な技法は、バッファー「リング」の中の次のバッファーに進むことが可能である（例えば、ＣＵＲＲＥＮＴ８１０を使用して）。本明細書で説明される例示的な技法によれば、ＣＵＲＲＥＮＴ８１０が、セットのバッファーのうちのいずれが現在、新たな書込み要求を受け付けているかを示すのに使用され得る。

【０１１７】

[0129]本明細書で説明される例示的な技法によれば、現在、アクティブなバッファーをＳＥＡＬする（例えば、「封印ビット」標識８０８を介して）スレッドは、そのバッファーをＳＥＡＬする際、ＣＵＲＲＥＮＴ８１０をアップデートすることもする。例えば、このスレッドは、その後、次のＣＵＲＲＥＮＴバッファーを選択することが可能である。例えば、バッファーＩ／Ｏが完了すると、Ｉ／Ｏスレッドは、バッファーを封印解除することが可能であるが、現在のバッファーの役割をしている別のバッファーが存在し得るので、ＣＵＲＲＥＮＴ８１０を設定しない可能性がある。

【０１１８】

[0130]ＬＳＳは、ログ構造化ストアであり、したがって、概念的に「付加専用(append only)」である。例えば、ＬＳＳの実現には、任意の通常のログ構造化ファイルシステム（ＬＦＳ）の場合と同様に、ページの新たなバージョンを付加するために絶えずスペースを取り戻すことが関与し得る。例えば、この技法は、本明細書で「クリーニング」（例えば、前掲のＭ．Ｒｏｓｅｎｂｌｕｍ他参照）と呼ばれ得る。

【０１１９】

[0131]例示的なページの異なるバージョンは、異なる寿命を有し得るので、再使用することが望ましい可能性がある例示的な「ログ」の古い部分が現在のページ状態を含むことが可能である。例えば、例示的なログのこの「古い」セクションを再使用するのに、依然として最新のページ状態が、ログのアクティブな末尾に移動されることが可能であり、それらの状態をその末尾に付加して、そのより古い部分がその後の使用のためにリサイクルされ得るようにする。例えば、クリーニングすることのこの副作用は、書込みの回数を増加させ得る（このことは、本明細書で「ライトアンプリフィケーション」と呼ばれ得る（例えば、Ｘ．－Ｙ．Ｈｕ他、「Ｗｒｉｔｅ ａｍｐｌｉｆｉｃａｔｉｏｎ ａｎａｌｙｓｉｓ ｉｎ ｆｌａｓｈ－ｂａｓｅｄ ｓｏｌｉｄ ｓｔａｔｅ ｄｒｉｖｅｓ（フラッシュベースの固体状ドライブにおけるライトアンプリフィケーション分析）」、Ｉｎ Ｐｒｏｃｅｅｄｉｎｇｓ ｏｆ ＳＹＳＴＯＲ ２００９：Ｔｈｅ Ｉｓｒａｅｌｉ Ｅｘｐｅｒｉｍｅｎｔａｌ Ｓｙｓｔｅｍｓ Ｃｏｎｆｅｒｅｎｃｅ（ＳＹＳＴＯＲ ’０９）、Ａｒｔｉｃｌｅ Ｎｏ．１０参照））。

【０１２０】

[0132]例えば、クリーニングする作業は、単に編成され得る。例えば、ログは、最も古い部分（例えば、ログの先頭）が「クリーニング」されて、新たなページ状態が書き込まれるログのアクティブな末尾に新たなスペースとして追加される大きい「循環バッファー」として管理され得る。

【０１２１】

[0133]本明細書で説明される例示的な技法によれば、再配置される各ページは、そのページが再書込みされる際、隣接させられる（例えば、ページが、ＬＳＳストアに再付加されると、「再書込みされた」内容は隣接する）。このため、どれほど多くのインクリメン

トフラッシュを経たとしても、そのページのすべての部分は、現時点で隣接させられ、このため、ＬＳＳ内のページのアクセス可能性を有利に最適化する。

【０１２２】

[0134]本明細書で説明される例示的な技法によれば、デルタ（本明細書で「再配置デルタ」と呼ばれ得る）に対するＣＡＳが、そのページに関するマッピングテーブルエントリーにおいて実行されて、新たなロケーションをもたらし、そのページのいずれの部分が再配置されたかを記述する（すなわち、その新たなロケーション情報をインストールするようにキャッシュを管理する）ことが可能である。例えば、同時に行われるアップデートまたはフラッシュが、このＣＡＳを失敗させる可能性があり、その場合、そのＣＡＳは、再び試みられる。

【０１２３】

[0135]ストレージ効率は、ログ構造化ストレージシステムに有利な良い影響を及ぼすことが可能である。本明細書で説明される例示的な技法によれば、ＬＳＳに割り当てられた任意の所与の量のスペースに関して、この技法がそのスペースをより効率的に使用するほど、この技法は、クリーニングを実行することが少なくなることが可能であり、このことには、より少ないページ移動が関与し得る。例えば、ページ移動は、ストレージに対するさらなる書込み（例えば、ライトアンプリフィケーション）をもたらし得る。

【０１２４】

[0136]潜在的なＬＳＳストレージ効率に関して、フラッシュされるページ内に空いたスペースは存在しない。例えば、それらのページは、パックされた可変長ストリングとして書き込まれることが可能である（例えば、平均で、従来のＢ－ＴＲＥＥページは、６９％だけしか利用されない可能性がある）。さらに、前回のフラッシュ以来のデルタだけが頻繁にフラッシュされる可能性があるため、１回のページフラッシュ当たり消費されるスペースがより少ないことが可能である。さらに、アップデートされたページをキャッシュからスワップアウトすることには、さらなるフラッシュが関与しない。というのは、キャッシュ内のメインメモリーが、それまでにフラッシュされたページの部分に関してだけ取り戻され得るからである。

【０１２５】

[0137]アクセスメソッドの１つの例示的な態様は、アクセスメソッドが、そのような構造が成長すること、および縮小することを可能にする構造変更操作（ＳＭＯ）を行うことである。例えば、ＳＭＯは、通常のアップデートが、進行中のＳＭＯが存在する状況で正しく実行され、アトミック（すべてまたは無）であることが可能であるようにインデックスのアトミック変更を行う方法が存在するものと予期する。例えば、例示的なＢＷ－ＴＲＥＥが、そのＢＷ－ＴＲＥＥのＳＭＯのための機構としてシステムトランザクションを活用することが可能である。

【０１２６】

[0138]本明細書で説明される例示的な技法によれば、システムトランザクションの持続性が、ログを介して実現され得る。しかし、本明細書で説明される一部のログは、トランザクションログではなく、例示的なＬＳＳ「ページ」ストアであり、このことは、トランザクションシステムが通常、操作をログ記録するだけであり得ることから、いくぶん不十分であるように見える可能性がある。しかし、デルタアップデートを用いて、ページ状態は、前回のページフラッシュ以来のデルタアップデートだけをログ記録することによってログ記録され得る。コミット時に持続性は関与せず、したがって、コミットは、ＬＳＳバッファーを「強制」しない。しかし、本明細書で説明される例示的な技法によれば、トランザクションの結果を使用するすべての後の操作は、ＬＳＳ内のトランザクションコミット後に生じることが確実にされ得る。

【０１２７】

[0139]本明細書で説明される例示的な技法によれば、非トランザクション操作と同様に、すべてのトランザクション操作も、マッピングテーブル内のページポインターに対するＣＡＳを介してインストールされ得る。本明細書で説明される例示的な技法は、キャッシ

ュ内のコンテンツがＬＳＳ内で忠実に表されることを確実にすること、およびＬＳＳ内のコンテンツがキャッシュ内で忠実に表されることを確実にすることが可能である。このため、システムトランザクション内の実質的にすべてのアップデートが、フラッシュ操作を含み得る。例えば、すべてのシステムトランザクションアップデートが、ＬＳＳバッファー内に記録されることが可能であり、このため、「ログ記録」されることが可能である。例えば、情報のその２つの表現は等価であることが可能であり、これにより、システムクラッシュが生じた場合に、キャッシュの状態が、ＬＳＳによって安定してキャプチャされた最終のバッファー時点で忠実に再構築され得ることが確実にされる。

【０１２８】

[0140]この等価性は、ＳＭＯの場合のように、アクションに複数のページが関与する場合、従来、問題がある可能性がある。例えば、Ｂ－ＬＩＮＫツリーにおけるノード分割ＳＭＯは、新たなページを割り当てることと、その新たなページを参照するその新たなページの兄弟ページリンクポインターをアップデートすることをともに行う。例えば、ラッチベースのシステムにおけるＳＭＯは、通常、ラッチを使用して隔離をもたらして、マルチページＳＭＯの内部状態が、そのＳＭＯが完了するまで、キャッシュマネージャー内で可視でないようにすることが可能である。例えば、ラッチフリーの設計は、アクティブな（したがって、コミットされていない）トランザクションアップデートを隔離する能力が制限され得ることを意味する可能性がある。

【０１２９】

[0141]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、ページに対する実質的に任意のアクセスを許すトランザクションインターフェースをもたらすことが可能である（すなわち、任意のページに対する操作がトランザクション内に入れられることが可能である）。しかし、トランザクション中にアップデートされたページが、そのトランザクションの外部の操作によるアクセスから保護されない可能性がある。しかし、本明細書で説明される例示的な技法によれば、完全に一般的な隔離能力が関与しないＳＭＯが設計され得る。例えば、図９が、ＳＭＯトランザクションをキャプチャするために使用され得る例示的なトランザクションテンプレート９００を示す。

【０１３０】

[0142]例えば、ステップ１（９０２）で、ページがマッピングテーブル内で割り当てられる、または解放される。ステップ２（９０４）で、ページが、必要に応じてアップデートされる。ステップ３（９０６）で、既存のページが、新たなページをインデックスの残りの部分に接続するように、または別のページをアップデートしている間に既存のページを取り除くようにアップデートされる。

【０１３１】

[0143]本明細書で説明される例示的な技法によれば、ノード分割（図９の例示的なテンプレートを使用する）のための新たなノードは、そのノードがツリーに接続され、トランザクションがコミットされる際、図９のステップ３まで、他のスレッドに可視ではない。このため、そのようなＳＭＯトランザクションは、アトミック性と隔離をともにもたらすことが可能である。

【０１３２】

[0144]従来のトランザクションシステムといくぶん同様に、アクティブトランザクションテーブル（ＡＴＴ）と本明細書で呼ばれ得るアクティブトランザクションテーブルが、システムトランザクションに関して保持され得る。例えば、ＡＴＴは、アクティブなシステムトランザクションごとに、そのトランザクションのトランザクションｉｄ（ＴＩＤ）と、そのトランザクションの最新の操作のメモリーアドレスをポイントする（またはそれ以外で参照する）、そのトランザクションの直前の操作に対するポインター（「ＩＰ」（「直前」を表す）と本明細書で呼ばれ得る）とを含むエントリーを含み得る。

【０１３３】

[0145]例えば、ＢｅｇｉｎＴｒａｎｓ操作（例えば、ＴＢｅｇｉｎ）が、任意の先行するトランザクションよりも高いトランザクションｉｄ（ＴＩＤ）を有し、ＩＰがＮＵＬＬ

の値に設定された新たなエントリーをＡＴＴに追加することが可能である。例えば、トランザクション操作の実行が、そのＩＰによって識別されるその操作に関するログレコードをポイントする、その操作に関する「ログレコード」を作成することが可能であり、ＩＰが、その新たな操作を参照するようにアップデートされることが可能である。例えば、このことは、すべての「ログレコード」がメインメモリー内に存在して、トランザクションの操作に関する「ログレコード」をバックリンクする(backlink)役割をすることが可能である。さらに、本明細書で説明される例示的な技法によれば、システムトランザクション内の操作は、マッピングテーブルアップデートを介してキャッシュ状態しか（すなわち、ＬＳＳバッファー状態ではなく）変更しないことが可能である。本明細書で説明される例示的な技法によれば、これらのページは、トランザクションコミット時にフラッシュされ得る。本明細書で説明される例示的な技法によれば、トランザクションの終了（コミットまたは中途終了）が生じると、トランザクションは、ＡＴＴから取り除かれ得る。

【０１３４】

[0146]例えば、システムトランザクションマネージャーが、キャッシュ層マネージャーによって保持されるアクティブトランザクションテーブル（ＡＴＴ）に第１のトランザクションのトランザクション識別子（ＴＩＤ）を追加するように構成され得る。例えば、トランザクションコミットマネージャーが、ＡＴＴからＴＩＤを取り除くこと、第１のトランザクションに関連付けられたページ状態変更をマッピングテーブルの中にインストールすること、および二次ストレージバッファーに第１のトランザクションに関連付けられたページ状態変更をフラッシュすることを開始することに基づいて、第１のトランザクションをコミットするように構成され得る。

【０１３５】

[0147]本明細書で説明される例示的な技法によれば、コミット操作の時点で、トランザクションによって変更されたページが、アトミック様態でＬＳＳバッファーにフラッシュされる。例示的な技法として、これらのページ書込みは、ＬＳＳ内のそのトランザクションに関する開始レコードおよび終了レコードで囲まれ得るが、このことには、中断されたトランザクションを元に戻すクラッシュ回復が関与し得る。例えば、そのような元に戻す回復には、元に戻す情報をＬＳＳに書き込むことが関与し得る。本明細書で説明される例示的な技法によれば、このことは、後段でさらに説明されるとおり、トランザクションによって変更されたすべてのページのアトミックフラッシュを、コミット時に実行することによって回避され得る。

【０１３６】

[0148]本明細書で説明される例示的な技法によれば、ＳＭＯに依存する後のアクションは、ＳＭＯトランザクションを記述する情報よりもＬＳＳバッファー内で後に出現する。このため、ＳＭＯの状態が、システムトランザクションを扱っているスレッド以外のスレッドにキャッシュ内で可視になると、それらの他のスレッドは、ＬＳＳにコミットされており、ＬＳＳバッファー内に既に存在するＳＭＯに依存し得る。

【０１３７】

[0149]図９に示されるとおり、ステップ３は、「新たなページをインデックスの残りの部分に接続するように、または別のページをアップデートしている間に既存のページを取り除くように既存のページをアップデートする」を示す。このため、本明細書で説明される例示的な技法は、アトミックフラッシュを介してメインメモリー内でアップデートすること（トランザクション状態を可視にすること）とＬＳＳバッファー内でトランザクションをコミットすることの両方をカプセル化することを、これを実現するのにＵｐｄａｔｅ－Ｄに関する例示的な「コミット」能力を使用して（すなわち、アップデートをトランザクションコミットと組み合わせて）行うことが可能である。

【０１３８】

[0150]本明細書で説明される例示的な技法によれば、ＬＳＳは、アップデート、およびアップデートのＣＡＳインストールを、トランザクションにおいて変更されたすべてのページのアトミックフラッシュと組み合わせることによって、トランザクションＵｐｄａｔ

ｅ－Ｄ「コミット」操作を可能にし得る。例えば、複数のページのコミット時のこのフラッシュは、個々のページフラッシュの場合と同様に実行され得る。例えば、ＬＳＳバッファースペースが、トランザクションにおいて変更されたすべてのページに割り当てられることが可能である。次に、フラッシュデルタが先頭に付加されたＵｐｄａｔｅ－Ｄデルタをインストールするｃａｓが、実行され得る。そのｃａｓが成功した場合、トランザクションにおいてアップデートされたページは、ＬＳＳフラッシュバッファーに書き込まれ得る。そのトランザクションに関するすべてのページのフラッシュが完了した後、フラッシュプロセスは、フラッシュバッファーの書込み側の数をデクリメントすることが可能である。例えば、トランザクションにおけるすべてのページに単一のユニットとしてスペースを割り当て、ＬＳＳバッファー上の書込み側デクリメントまで保持することが、ＬＳＳストア内のそのトランザクションに関するアトミック性を確実にし得る。

【０１３９】

[0151]例えば、トランザクションコミットマネージャーが、コンペア・アンド・スワップ（ｃａｓ）操作を介して、トランザクションに関連付けられたアップデートデルタレコードをマッピングテーブルの中にインストールするように構成されることが可能であり、そのアップデートデルタレコードの先頭には、フラッシュデルタレコードが付加される。例えば、トランザクションコミットマネージャーが、そのｃａｓ操作が成功したかどうかを判定するように構成され得る。トランザクションコミットマネージャーが、そのｃａｓ操作が成功したと判定した場合、トランザクションコミットマネージャーは、トランザクションにおいてアップデートされたページを二次ストレージフラッシュバッファーに書き込む書込み操作を開始することが可能である。

【０１４０】

[0152]本明細書で説明される例示的な技法によれば、そのｃａｓが失敗した場合、応答は、他のフラッシュ失敗の場合と同様に進められ得る。例えば、ＬＳＳが、回復中、そのスペースを別のものと混同しないように割り当てられていたスペースが、ＶＯＩＤにされ得る。このため、例示的な回復プロセスは、システムトランザクションをまったく意識しないことが可能である。むしろ、システムトランザクションは、例示的なキャッシング層の能力のみであり得る。このため、システムクラッシュまたはシステム再起動の前後でＴＩＤ一意性を確実にすることなしに進めることが許容可能であり得る。

【０１４１】

[0153]本明細書で説明される例示的な技法によれば、中途終了されたシステムトランザクションの操作は、回復が不完全なトランザクションを見ないので、キャッシュ内で元に戻され得る。このため、メインメモリー内で一緒にリンクされた、そのトランザクションに関するログレコードのバックチェーンがたどられることが可能であり、元に戻すことは、そのトランザクションに関するＡＴＴリスト上の操作の性質に基づいてもたらされ得る。例えば、デルタアップデートは、そのデルタを取り除くことによって元に戻されることが可能であり、割当ては、「解放する」ことで元に戻されることが可能であり、「解放する」は、「解放する」に先立つページの状態にページを復元することによって元に戻されることが可能である。「解放する」を元に戻すことを除いて、操作成功を記述する情報を超えた、これらの操作に関する特別の情報は、まったく所望されない可能性がある。

【０１４２】

[0154]本明細書で説明される例示的な技法によれば、トランザクション内で行われるアクションは、ストレージおよびマッピングテーブル・ページ・エントリー（ＰＩＤ）を割り当てること、および解放することを含め、暫定的である。例えば、トランザクション実行中、ＰＩＤが、割り当てられること、または解放されることが可能であり、Ｕｐｄａｔｅ－Ｄデルタが、生成されることが可能である。例えば、これらのリソースの管理は、本明細書で説明されるとおり、エポック機構に基づいて実現され得る。例えば、ＳＭＯは、単一のユーザー操作要求内で実行されるので、スレッドは、そのトランザクションの持続時間にわたってそのスレッドのエポック内に留まり得る。

【０１４３】

[0155]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、トランザクションコミットまたはトランザクション中途終了に依存して、リソースを取り戻すことが可能である。例えば、コミット操作に関して、ＦｒｅｅＰａｇｅ　ＰＩＤが、現在のエポックに関するＰＩＤ解放待ち状態リストに追加され得る。例えば、中途終了操作に関して、ＡｌｌｏｃａｔｅＰａｇｅ　ＰＩＤが、元に戻す間に解放されることが可能であり、ＰＩＤ解放待ち状態リストに同様に追加され得る。例えば、Ｕｐｄａｔｅ－Ｄ操作に関して、アップデートデルタは、トランザクションが中途終了された場合、現在のエポックに関するストレージ解放待ち状態リストに追加され得る。

【０１４４】

[0156]本明細書で説明される「クラッシュ回復」とは、一般に、「トランザクション回復」を指す。本明細書で説明される「チェックポインティング」とは、一般に、トランザクションログを管理するのに使用されるチェックポインティングを指す。むしろ、本明細書で説明される「クラッシュ回復」とは、ＬＳＳ（例えば、ログ構造化ストア）が、ページのＬＳＳのマッピングテーブル、およびそれらのページの状態を、システムクラッシュの時点に回復する例示的な技法を指し得る。この特定のタイプの回復ステップは、通常、従来のアップデート・インプレース・ストレージ・システムの関心事ではない。

【０１４５】

[0157]本明細書で説明される「クラッシュ回復」に関して、マッピングテーブルは、ある種の「データベース」と見なされ得る。例えば、このデータベースに対するアップデートは、ＬＳＳにフラッシュされるページ状態を含み得る。このため、すべてのページフラッシュは、その「マッピングテーブルデータベース」をアップデートすることが可能である。システムがクラッシュした場合、ＬＳＳ「ログ」は、フラッシュされたページを、マッピングテーブルをアップデートするやり直しログレコードとして使用して、「マッピングテーブルデータベース」を回復するように再実行され得る。

【０１４６】

[0158]前述の戦略をサポートして、マッピングテーブルは、ＬＳＳアップデートを無期限に保持することを回避するように、周期的にチェックポイント処理されることが可能である。例えば、前述したＬＦＳクリーニング技法が、この目的で使用され得る（すなわち、回復ログを短くして）が、そのような技法は、高速回復のために望ましいとされ得るよりも、相当に大きい回復ログ（ＬＳＳログ構造化ストア）を残す可能性がある。

【０１４７】

[0159]本明細書で説明される例示的な技法によれば、チェックポインティングのために有利な方策が使用され得る。例えば、例示的なＬＬＡＭＡ実施形態が、チェックポイント処理中に完全なマッピングテーブルを、２つの交互のロケーションのうちの１つに非同期に、インクリメンタルに書き込むことが可能である。図１０は、本明細書で説明される例示的な技法による例示的なチェックポイントデータ１０００を示す。例えば、２つの交互のロケーションが、様々なエンティティのロケーションに関するその他の「最新の」情報を失う可能性があるシステムクラッシュの後でさえ、システムがそれらのロケーションを知っているように、異なる２つの「よく知られたロケーション」（ＷＫＬ）として選択され得る。このため、クラッシュの時点で存在していたとおりのシステムの状態に関する情報をポイントするポインターが、保存され得る（例えば、ＷＫＬを使用して）。例えば、２つのチェックポイントを使用することによって、ユーザーが、「ライブの」チェックポイントのアップデート・インプレースを行わないことが可能である。

【０１４８】

[0160]例えば、各ロケーションは、図１０に示されるとおり、完全なマッピングテーブルに加えて、フラッシュログ１００６の中に回復開始位置（ＲＳＰ）１００２およびガベージコレクションオフセットＧＣ１００４を記憶することが可能である。例えば、ＲＳＰ１００２は、マッピングテーブル３０４をコピーすることを開始した時点におけるＬＳＳストア内の終了オフセットを含み得る。例えば、ＧＣオフセット１００４は、ガベージコレクション「フロンティア」に印を付けることが可能である。

【０１４９】
[0161]本明細書で説明される例示的な技法によれば、より後のチェックポイントは、ＬＳＳオフセットが、仮想化されることによって単調に増加するので、より高いＲＳＰ１００２を有する。例えば、システムクラッシュの後、最高のＲＳＰ１００２で完了したチェックポイント処理が、回復されたマッピングテーブル３０４の状態を初期設定するのに使用され得る。例えば、ＲＳＰ１００２は、やり直し回復を開始するためのＬＳＳ「ログ」（１００６）内の位置を示す。最新の完了したチェックポイント処理を識別するのに、ＲＳＰ１００２は、マッピングテーブル３０４が完全にキャプチャされてしまうまで、チェックポイントに書き込まれない。このため、現在のチェックポイント処理が完了するまで、前の高いＲＳＰ（代替のロケーションからの）が最高のＲＳＰ１００２である。

【０１５０】
[0162]本明細書で説明される例示的な技法によれば、マッピングテーブル３０４をチェックポイントの一部として書き出すことは、キャッシュ内に存在するとおりのマッピングテーブル３０４のバイトごとに対応するコピーではない。例えば、キャッシュされた形態のマッピングテーブル３０４は、キャッシュされたページに関してマッピングテーブルエントリーの中にメインメモリーポインターを有するのに対して、本明細書で説明される例示的な所望されるチェックポイントには、それらのページのＬＳＳアドレスをキャプチャすることが関与する。別の例として、現在、割り当てられていないマッピングテーブルエントリーが、それらのマッピングテーブルエントリーをリスト項目として使用する空きリスト上に保持される。このため、空きマッピングテーブルエントリーは、０、または直前の空きマッピングテーブルエントリーのアドレス（その０またはアドレスが空きリストに追加された時刻に基づく時間順序で）を有する。例えば、使用可能な空きリストは、本明細書で説明されるとおり、マッピングテーブルを非同期で「コピー」している間、キャプチャされないことが可能である。例えば、本明細書で説明されるマッピングテーブル３０４のコピーは、非同期で、インクリメンタルに書き込まれ、このことは、通常の実行に対する影響を最小限に抑えるのを助けることが可能である。

【０１５１】
[0163]本明細書で説明される例示的な技法によれば、例示的なＬＬＡＭＡ実施形態が、ＬＳＳの現在の終了オフセットをＲＳＰ１００２としてまず保存することが可能であり、現在のＬＳＳクリーニングオフセットをＧＣ１００４として保存することが可能である。例えば、マッピングテーブル３０４が、スキャンされる（例えば、進行中の操作と同時に）ことが可能であり、各ＰＩＤエントリー（最新のフラッシュデルタの中に記憶された）に関するページの最新のフラッシュが識別されることが可能であり、そのＬＳＳアドレスが、そのマッピングテーブル３０４エントリーに関する例示的なチェックポイントの中に記憶され得る。例えば、エントリーが空いている場合、そのエントリーは、チェックポイントコピーにおいてゼロ化され得る。例えば、やり直し回復の終わりに空きリストが再構築され得る。さらに、マッピングテーブル３０４をコピーすることが完了すると、それまでに保存されたＲＳＰ１００２およびＧＣ１００４は、安定したチェックポイント領域に書き込まれることが可能であり、これにより、チェックポイント処理が完了する。

【０１５２】
[0164]本明細書で説明される例示的な技法によれば、回復は、最高のＲＳＰ１００２を有するチェックポイント（すなわち、最新の完全なチェックポイント）に関するマッピングテーブル３０４をキャッシュ内３１２にコピーすることによって開始され得る。例えば、ログ１００６が、次に、ＲＳＰ１００２からＬＳＳの終わりまで順方向に読み取られ得る。例えば、出会う各ページフラッシュが、そのページフラッシュがページ読取りの結果であるかのように、キャッシュ３１２内にもたらされ得る。

【０１５３】
[0165]例えば、ページのコンテンツが読み取られることが可能であり、デルタは、ＬＳＳ内のロケーションがフラッシュデルタの中で参照されるように設定されることが可能である。例えば、ＡｌｌｏｃａｔｅＰａｇｅ操作に出会うと、割り当てられたＰＩＤに関す

るマッピングテーブル３０４エントリーが、ＡｌｌｏｃａｔｅＰａｇｅ操作によって予期されるとおり「空」に初期設定され得る。例えば、ＦｒｅｅＰａｇｅ操作に出会うと、マッピングテーブル３０４エントリーは、ＺＥＲＯに設定され得る。例えば、ＬＳＳクリーナが、チェックポイントから読み取られたＧＣオフセット（１００４）からログに対するガベージコレクションを再開することが可能である。

【０１５４】

[0166]本明細書で説明される例示的な技法によれば、回復中、すべての空きマッピングテーブル３０４エントリーが、ＺＥＲＯに設定され得る。例えば、再構築されたマッピングテーブル３０４がスキャンされ得る。例えば、ＺＥＲＯエントリーに出会うと、そのＺＥＲＯエントリーは、空きリストに追加されることが可能であり、空きリストは、スタックとして管理され得る（すなわち、再使用されるべき最初のエントリーは、リストに追加された最後のエントリーである）。これらの例示的な技法によれば、下位のＰＩＤは、再使用されることが可能であり（再使用の際の選好として）、このことは、テーブルサイズを、クラスタ化されて、小さく保つ傾向があり得る（少なくとも回復の結果）。さらに、それまでに使用された最高のＰＩＤを示す最高水位線が、マッピングテーブル内に保持され得る。例えば、空きリストが尽きると、テーブルの使用されていない部分からＰＩＤが追加されて、最高水位線がインクリメントされることが可能である。

【０１５５】

[0167]本明細書でさらに説明されるとおり、図１１は、ラッチフリーのログ構造化ストレージを管理するためのシステム１１００のブロック図である。システム１１００は、ハードウェア実施形態、ソフトウェア実施形態、またはハードウェア実施形態とソフトウェア実施形態の組合せとして実現され得ることが、データ処理の分野の業者には理解されよう。図１１に示されるとおり、システム１１００が、少なくとも１つのプロセッサー１１０４を含むデバイス１１０２を含み得る。デバイス１１０２は、任意に選択されたページ指向型アクセスメソッド１１１０に、ページデータストレージ１１１２に対するラッチフリーのアクセスを含むページデータストレージ１１１２に対するインターフェースアクセスをもたらすように構成され得る。データ不透明型インターフェース１１０８を含み得るデータマネージャー１１０６を含み得る。例えば、ページ指向型アクセスメソッド１１１０は、任意の恣意的なアクセスメソッドであり得る。例えば、ページデータストレージ１１１２は、メインメモリーなどの（少なくとも）揮発性ストレージ、ならびにフラッシュストレージ、および他のタイプのディスクドライブなどを含み得る「二次ストレージ」などのより安定したストレージ（例えば、より不揮発性のストレージ）を含め、任意のタイプのページデータストレージを含み得る。本明細書の説明の趣旨を逸脱することなく、本明細書で説明される技法と一緒に使用され得る多くのタイプのページデータストレージが存在することが、データ処理の分野の業者には理解されよう。

【０１５６】

[0168]例示的な実施形態によれば、データマネージャー１１０６、またはデータマネージャー１１０６の１つまたは複数の部分が、後段でさらに説明されるとおり、有形のコンピューター可読記憶媒体上に記憶され得る実行可能命令を含み得る。例示的な実施形態によれば、コンピューター可読記憶媒体は、分散デバイスを含め、任意の数のストレージデバイス、および任意の数の記憶媒体タイプを含み得る。

【０１５７】

[0169]この脈絡において、「プロセッサー」は、コンピューティングシステムに関連する命令を処理するように構成された単一のプロセッサー、または複数のプロセッサーを含み得る。このため、プロセッサーは、並行に、かつ／または分散された様態で命令を処理する１つまたは複数のプロセッサーを含み得る。デバイスプロセッサー１１０４は、図１１のデータマネージャー１１０６の外部に描かれるものの、デバイスプロセッサー１１０４は、データマネージャー１１０６および／またはデータマネージャー１１０６の要素のいずれかの内部または外部に配置され得る単一のコンポーネントとして、かつ／または分散ユニットとして実装され得ることが、データ処理の分野の業者には理解されよう。

【０１５８】
[0170]例えば、システム１１００は、１つまたは複数のプロセッサー１１０４を含み得る。例えば、システム１１００は、１つまたは複数の１１０４によって実行可能な命令を記憶する少なくとも１つの有形のコンピューター可読記憶媒体を含むことが可能であり、それらの実行可能な命令は、少なくとも１つのデータ処理装置に、本明細書で説明されるとおり、システム１１００に含まれる様々な例示的な構成要素に関連付けられた操作を実行させるように構成される。例えば、その１つまたは複数のプロセッサー１１０４は、少なくとも１つのデータ処理装置の中に含められ得る。本明細書の説明の趣旨を逸脱することなく、本明細書の説明により構成され得るプロセッサーおよびデータ処理装置の多くの構成が存在することが、データ処理の分野の業者には理解されよう。

【０１５９】
[0171]この脈絡において、「コンポーネント」とは、ある操作を実行するように構成され得る命令またはハードウェアを指すことが可能である。そのような命令は、命令のコンポーネントグループ内に含められることが可能であり、または複数のグループにわたって分散されることが可能である。例えば、第１のコンポーネントの操作に関連するいくつかの命令が、第２のコンポーネント（またはより多くのコンポーネント）の操作に関連する命令のグループに含められてもよい。例えば、本明細書の「コンポーネント」は、単一のエンティティ内に配置され得る命令によって実施され得るある種の機能を指すことが可能であり、または複数のエンティティにわたって散在させられる、もしくは分散させられることが可能であり、他のコンポーネントに関連する命令および／またはハードウェアと重なり合うことが可能である。

【０１６０】
[0172]例示的な実施形態によれば、データマネージャー１１０６は、１つまたは複数のユーザーデバイスに関連して実施され得る。例えば、データマネージャー１１０６は、後段でさらに説明されるとおり、サーバーと通信することが可能である。

【０１６１】
[0173]例えば、１つまたは複数のデータベースが、データベースインターフェースコンポーネント１１２２を介してアクセスされ得る。様々なタイプのデータベース構成（例えば、リレーショナルデータベース、階層データベース、分散データベース）および非データベース構成などの、本明細書で説明される情報を記憶するための多くの技法が存在することが、データ処理の分野の業者には理解されよう。

【０１６２】
[0174]例示的な実施形態によれば、データマネージャー１１０６は、即時の結果などのオブジェクトを記憶することが可能なメモリー１１２４を含み得る。この脈絡において、「メモリー」は、データおよび／または命令を記憶するように構成された単一のメモリーデバイスまたは複数のメモリーデバイスを含み得る。さらに、メモリー１１２４は、複数の分散ストレージデバイスにわたることが可能である。さらに、メモリー１１２４は、複数のプロセッサーの間に分散され得る。

【０１６３】
[0175]例示的な実施形態によれば、ユーザーインターフェースコンポーネント１１２６が、ユーザー１１２８とデータマネージャー１１０６の間の通信を管理することが可能である。ユーザー１１２８は、ディスプレイ１１３２および他の入出力デバイスに関連付けられ得る受信デバイス１１３０に関連付けられ得る。例えば、ディスプレイ１１３２は、内部デバイスバス通信を介して、または少なくとも１つのネットワーク接続を介して受信デバイス１１３０と通信するように構成され得る。

【０１６４】
[0176]例示的な実施形態によれば、ディスプレイ１１３２は、フラットスクリーンディスプレイ、印刷形態のディスプレイ、２次元ディスプレイ、３次元ディスプレイ、静的ディスプレイ、移動ディスプレイ、触覚出力、オーディオ出力、およびユーザー（例えば、ユーザー１１２８）とコミュニケーションをとるための他の任意の形態の出力などの知覚

ディスプレイとして実施され得る。

【０１６５】

[0177]例示的な実施形態によれば、データマネージャー１１０６は、データマネージャー１１０６と、少なくとも１つのネットワーク１１３６を介してデータマネージャー１１０６と通信することが可能な他のエンティティとの間のネットワーク通信を管理することが可能なネットワーク通信コンポーネント１１３４を含み得る。例えば、ネットワーク１１３６は、インターネット、少なくとも１つのワイヤレスネットワーク、または少なくとも１つの有線ネットワークのうちの少なくとも１つを含み得る。例えば、ネットワーク１１３６は、データマネージャー１１０６のためにデータの伝送をサポートすることが可能なセルラーネットワーク、無線ネットワーク、または任意のタイプのネットワークを含み得る。例えば、ネットワーク通信コンポーネント１１３４は、データマネージャー１１０６と受信デバイス１１３０の間のネットワーク通信を管理することが可能である。例えば、ネットワーク通信コンポーネント１１３４は、ユーザーインターフェースコンポーネント１１２６と受信デバイス１１３０の間のネットワーク通信を管理することが可能である。

【０１６６】

[0178]例えば、データ不透明型インターフェース１１０８が、任意に選択されたページ指向型アクセスメソッド１１１０に、ページデータストレージ１１１２に対するログ構造化アクセスを含むページデータストレージ１１１２に対するインターフェースアクセスをもたらすように構成され得る。

【０１６７】

[0179]例えば、キャッシュ層マネージャー１１３８が、データ不透明型インターフェース１１０８に関連付けられた間接アドレスマッピングテーブル１１４２に対するテーブル操作を開始するように構成され得るマップテーブルマネージャー１１４０を含むことが可能であり、テーブル操作は、間接アドレスマッピングテーブル１１４２の中のエントリーに対してアトミック・コンペア・アンド・スワップ（ＣＡＳ）操作を開始して、ページデータストレージ１１１２に関連付けられたページのそれまでの状態を、それらのページの新たな状態で置き換えることを含む。

【０１６８】

[0180]例えば、マップテーブルマネージャー１１４０は、データ不透明型インターフェース１１０８に関連付けられた間接アドレスマッピングテーブル１１４２に対するテーブル操作を開始するように構成されることが可能であり、間接アドレスマッピングテーブル１１４２は、キャッシュ層ストレージ１１４４と、二次ストレージ１１４６とを含むデータストレージの管理のために共通で使用される。

【０１６９】

[0181]例えば、間接アドレスマッピングテーブル１１４２は、ページの論理ロケーションを、それらのページの対応する物理ロケーションから分離し、ページデータストレージのユーザーは、それらのページの物理ロケーションアドレス値の代わりにページ識別子値を、ページデータストレージを参照するデータ構造における別の場所に記憶させる。

【０１７０】

[0182]例えば、アップデートマネージャー１１４８が、間接アドレスマッピングテーブル１１４２の中のエントリーに対してラッチフリーのコンペア・アンド・スワップ操作を使用して、間接アドレスマッピングテーブル１１４２上でアトミック状態変更を行うようデータアップデートおよび管理アップデートを制御するように構成され得る。

【０１７１】

[0183]例えば、ストレージ層１１４９が、間接アドレスマッピングテーブル１１４２の中のエントリーに対してラッチフリーのコンペア・アンド・スワップ操作を使用して、ページフラッシュからもたらされるログ構造化に関連するページロケーション変更を制御するように構成され得るログ構造化ストレージ層マネージャー１１５０を含み得る。

【０１７２】

[0184]例えば、バッファーマネージャー１１５１が、ラッチフリーのアップデート操作を介してログ構造化二次ストレージバッファーに対するアップデートを制御するように構成され得る。このため、例えば、複数のスレッドが、ラッチフリーの操作を介してログ構造化二次ストレージバッファーを同時にアップデートすることが可能である。

【０１７３】

[0185]例えば、バッファーマネージャー１１５１は、第１の二次ストレージアドレス引数までの、下位のアドレスを有する、ログ構造化二次ストレージバッファーにフラッシュされたページが、ログ構造化二次ストレージ内で安定していることを判定するための安定操作を開始するように構成され得る。

【０１７４】

[0186]例えば、ページマネージャー１１５２が、ページに対するフラッシュ操作、割当て操作、および解放操作を制御するように構成され得る。例えば、ページマネージャー１１５２は、コンペア・アンド・スワップ（ＣＡＳ）操作に基づいて、二次ストレージバッファー内に第１のページのページ状態をコピーすることを開始すること、そのページ状態の先頭にフラッシュデルタレコードを付加することであって、そのフラッシュデルタレコードは、二次ストレージ内の第１のページのストレージロケーションと、呼び出し元に関連付けられた注釈とを示す二次ストレージアドレスを含むこと、およびマッピングテーブルの中にフラッシュデルタレコードのアドレスをインストールすることに基づいてページ状態に対するアップデートを開始することに基づいて、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページをフラッシュする操作を開始するように構成され得る。

【０１７５】

[0187]例えば、ページマネージャー１１５２は、第１のページに関連付けられたページ状態の先頭に部分的スワップデルタレコードを付加することであって、その部分的スワップデルタレコードは、第１のページの欠落した部分の二次ストレージ内のロケーションを示すフラッシュデルタレコードのストレージロケーションを示すメインメモリーアドレスを含むことに基づいて、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページの部分をスワップする操作を開始するように構成され得る。

【０１７６】

[0188]例えば、システムトランザクションマネージャー１１５４が、トランザクションをコミットすること、およびトランザクションを中途終了することを行うように構成され得る。

【０１７７】

[0189]例えば、レコードマネージャー１１５６が、アップデートデルタレコード操作および置換アップデート操作に基づいてアップデートを制御するように構成され得る。

[0190]例えば、エポックマネージャー１１６０が、第１のプロセッサー操作によってページ情報にアクセスすることに先立って、第１のエポックに関連付けられた第１のエポック登録リスト内に第１のプロセッサー操作を登録することを開始するように構成され得る。例えば、第１のプロセッサー操作は、スレッドであり得る。

【０１７８】

[0191]例えば、ページマネージャー１１５２は、コンペア・アンド・スワップ（ＣＡＳ）操作を介して、マッピングテーブルの中にフラッシュデルタレコードに対するポインターをインストールすることであって、そのフラッシュデルタレコードは、ＣＡＳ操作を介してマッピングテーブル内で置き換えられる既存のページ状態の先頭に付加されることに基づいて、二次ストレージにページ状態をフラッシュするように構成され得る。

【０１７９】

[0192]例えば、ページマネージャー１１５２は、そのＣＡＳ操作が成功したかどうかを判定し、そのＣＡＳ操作が成功したと判定された場合、二次ストレージフラッシュバッファーに既存のページ状態を書き込む書込み操作を開始するように構成され得る。

【０１８０】

[0193]例えば、ページマネージャー１１５２は、そのＣＡＳ操作が失敗したと判定された場合、既存のページにそれまでに割り当てられていたストレージスペースを無効にする操作を開始するように構成され得る。

【０１８１】

[0194]本明細書の説明の趣旨を逸脱することなく、多くの異なる技法が、ラッチフリーのログ構造化ストレージシステムのために使用され得ることが、データ処理の分野の業者には理解されよう。

【０１８２】

[0195]ＩＩＩ．フローチャート説明

本明細書で説明される特徴は、本明細書の説明の趣旨を逸脱することなく、データ処理の分野の業者によって理解され得る多くの異なる様態で実施され得る例示的な実施形態として提供される。そのような特徴は、例示的な実施形態特徴としてのみ解釈されるべきであり、それらの詳細な説明だけに限定されるものと解釈されることは意図していない。

【０１８３】

[0196]図１２ａ～図１２ｄは、例示的な実施形態による、図１１のシステムの例示的な動作を示すフローチャートである。図１２ａの例において、ページデータストレージに対するラッチフリーのアクセスを含むページデータストレージに対するインターフェースアクセスが、任意に選択されたページ指向型アクセスメソッドにもたらされ得る（１２０２）。例えば、前述したとおり、任意に選択されたページ指向型アクセスメソッド１１１０に、ページデータストレージ１１１２に対するラッチフリーのアクセスを含むページデータストレージ１１１２に対するインターフェースアクセスをもたらすことが可能なデータ不透明型インターフェース１１０８。

【０１８４】

[0197]例えば、ページデータストレージに対するインターフェースアクセスは、安定したページデータストレージに対するログ構造化アクセスを含み得る（１２０４）。例えば、データ不透明型インターフェース１１０８が、前述したとおり、任意に選択されたページ指向型アクセスメソッド１１１０に、ページデータストレージ１１１２に対するログ構造化アクセスを含むページデータストレージ１１１２に対するインターフェースアクセスをもたらすことが可能である。

【０１８５】

[0198]例えば、データ不透明型インターフェースに関連付けられた間接アドレスマッピングテーブルに対するテーブル操作が開始されることが可能であり、それらのテーブル操作は、間接アドレスマッピングテーブル内のエントリーに対してアトミック・コンペア・アンド・スワップ操作を開始して、ページデータストレージに関連付けられたページのそれまでの状態を、それらのページの新たな状態で置き換えることを含む（１２０６）。例えば、マップテーブルマネージャー１１４０が、前述したとおり、データ不透明型インターフェース１１０８に関連付けられた間接アドレスマッピングテーブル１１４２に対してテーブル操作を開始することが可能であり、テーブル操作は、間接アドレスマッピングテーブル１１４２内のエントリーに対してコンペア・アンド・スワップ（ＣＡＳ）操作を開始して、ページデータストレージ１１１２に関連するページのそれまでの状態を、それらのページの新たな状態で置き換えることを含む。

【０１８６】

[0199]例えば、間接アドレスマッピングテーブルが、図１２ｂに示されるとおり、キャッシュ層ストレージと、二次ストレージとを含むデータストレージの管理のために共通で使用され得る（１２０８）。例えば、マップテーブルマネージャー１１４０が、前述したとおり、データ不透明型インターフェース１１０８に関連付けられた間接アドレスマッピングテーブル１１４２に対するテーブル操作を開始することが可能であり、間接アドレスマッピングテーブル１１４２は、キャッシュ層ストレージ１１４４と、二次ストレージ１１４６とを含むデータストレージの管理のために共通で使用される。

【０１８７】

[0200]例えば、ページの論理ロケーションが、前述したとおり、それらのページの対応する物理ロケーションから分離されることが可能であり、ページデータストレージのユーザーは、それらのページに関する物理ロケーションアドレスの代わりにページ識別子値を、ページデータストレージを参照するデータ構造における別の場所に記憶させる（１２１０）。例えば、間接アドレスマッピングテーブル１１４２が、前述したとおり、ページの論理ロケーションを、それらのページの対応する物理ロケーションから分離し、ページデータストレージのユーザーは、それらのページに関する物理ロケーションアドレスの代わりにページ識別子値を、ページデータストレージを参照するデータ構造における別の場所に記憶させる。

【０１８８】

[0201]例えば、データアップデートおよび管理アップデートが、間接アドレスマッピングテーブル内のエントリーに対してラッチフリーのコンペア・アンド・スワップ操作を使用して、間接アドレスマッピングテーブル上でアトミック状態変更を行って、制御され得る（１２１２）。例えば、アップデートマネージャー１１４８が、前述したとおり、間接アドレスマッピングテーブル１１４２内のエントリーに対してラッチフリーのコンペア・アンド・スワップ操作を使用して、間接アドレスマッピングテーブル１１４２上でアトミック状態変更を行って、データアップデートおよび管理アップデートを制御することが可能である。

【０１８９】

[0202]例えば、ページフラッシュからもたらされるログ構造化に関連するページロケーション変更が、間接アドレスマッピングテーブル内のエントリーに対してラッチフリーのコンペア・アンド・スワップ操作を使用して、制御され得る（１２１４）。例えば、ログ構造化ストレージ層マネージャー１１５０が、前述したとおり、間接アドレスマッピングテーブル１１４２内のエントリーに対してラッチフリーのコンペア・アンド・スワップ操作を使用して、ページフラッシュからもたらされるログ構造化に関連するページロケーション変更を制御することが可能である。

【０１９０】

[0203]例えば、図１２ｃの例において、第１のエポックに関連付けられた第１のエポック登録リストに第１のプロセッサー操作を登録することが、第１のプロセッサー操作によってページ情報にアクセスすることに先立って開始され得る（１２１６）。

【０１９１】

[0204]例えば、ページ状態が、コンペア・アンド・スワップ（ＣＡＳ）操作を介して、マッピングテーブルの中にフラッシュデルタレコードに対するポインターをインストールすることに基づいて、二次ストレージにフラッシュされることが可能であり、そのフラッシュデルタレコードは、ＣＡＳ操作を介してマッピングテーブル内で置き換えられる既存のページ状態の先頭に付加される（１２１８）。

【０１９２】

[0205]例えば、ログ構造化二次ストレージバッファーに対するアップデートが、ラッチフリーのアップデート操作を介して制御され得る（１２２０）。

[0206]例えば、図１２ｄの例において、コンペア・アンド・スワップ（ＣＡＳ）操作を介して、二次ストレージ内のロケーションにキャッシュ層ストレージ内の第１のページをフラッシュする操作が、第２のストレージバッファーの中に第１のページのページ状態をコピーすることを開始すること、そのページ状態の先頭にフラッシュデルタレコードを付加することを開始することであって、そのフラッシュデルタレコードは、二次ストレージ内の第１のページのストレージロケーションと、呼び出し元に関連付けられた注釈とを示す二次ストレージアドレスを含むこと、およびマッピングテーブルの中にフラッシュデルタレコードのアドレスをインストールすることに基づいて、そのページ状態に対するアップデートを開始することに基づいて、開始され得る（１２２２）。

【０１９３】

[0207]例えば、二次ストレージ内のロケーションに対してキャッシュ層ストレージ内の

第１のページの一部分をスワップする操作が、第１のページに関連付けられたページ状態の先頭に部分的スワップデルタレコードを付加することを開始することであって、その部分的スワップデルタレコードは、第１のページの欠落した部分の二次ストレージ内のロケーションを示すフラッシュデルタレコードのストレージロケーションを示すメインメモリーアドレスを含むことに基づいて、開始され得る（１２２４）。

【０１９４】

[0208]本明細書の説明の趣旨を逸脱することなく、ラッチフリーのログ構造化ストレージシステムのために多くの異なる技法が使用され得ることが、データ処理の分野の業者には理解されよう。

【０１９５】

[0209]顧客プライバシーおよび顧客機密保持が、長年にわたるデータ処理環境における継続的な配慮事項である。このため、ラッチフリーのログ構造化ストレージシステムのための例示的な技法は、そのような分析に関連付けられた、関連付けられたアプリケーションまたはサービスを有する１つまたは複数の契約合意（例えば、「サービス利用規約」ＴＯＳ合意）を介して許可を与えたユーザーによって提供されるユーザー入力および／またはユーザーデータを使用することが可能である。例えば、ユーザーが、ユーザーの入力データが送信され、デバイス上に記憶されることに同意を与えることが可能であり、ただし、（例えば、ユーザーが受諾した合意を介して）各関係者が、送信および／または記憶がどのように行われるか、および記憶が保持される場合、どのようなレベルもしくは持続時間の記憶が保持され得るかを管理することが可能である。

【０１９６】

[0210]本明細書で説明される様々な技法の実施形態は、デジタル電子回路として、またはコンピューターハードウェア、ファームウェア、ソフトウェア、またはそれらの組合せ（例えば、様々な機能を実行する命令を実行するように構成された装置）として実施され得る。

【０１９７】

[0211]実施形態は、純粋な伝播される信号などの純粋な信号として実現されるコンピュータープログラムとして実施され得る。そのような実施形態は、「コンピューター可読伝送媒体」を介して実施されるものとして本明細書に記載され得る。

【０１９８】

[0212]代替として、実施形態は、データ処理装置、例えば、プログラマブルプロセッサー、コンピューター、または複数のコンピューターによって実行されるように、またはデータ処理装置の動作を制御するようにマシン使用可能デバイスもしくはマシン可読ストレージデバイス（例えば、ユニバーサルシリアルバス（ＵＳＢ）ストレージデバイス、テープ、ハードディスクドライブ、コンパクトディスク、デジタルビデオディスク（ＤＶＤ）などの磁気媒体もしくはデジタル媒体）として実現されたコンピュータープログラムとして実施され得る。そのような実施形態は、「コンピューター可読記憶媒体」または「コンピューター可読ストレージデバイス」を介して実施されるものとして本明細書に記載されることが可能であり、このため、純粋な伝播される信号などの純粋に信号である実施形態と異なる。

【０１９９】

[0213]前述したコンピュータープログラムなどのコンピュータープログラムは、コンパイラー型言語、インタープリター型言語、または機械語を含め、任意の形態のプログラミング言語で書かれることが可能であり、スタンドアロンのプログラムとして、またはモジュール、コンポーネント、サブルーチンとして、あるいはコンピューティング環境において使用されるのに適した他のユニットとして展開されることを含め、任意の形態で展開され得る。コンピュータープログラムは、マシン使用可能デバイス上、またはマシン可読ストレージデバイス（例えば、コンピューター可読媒体）上の実行可能コード（例えば、実行可能命令）として実体化され得る。前述した技法を実施することが可能なコンピュータープログラムは、１つの場所における、または複数の場所にわたって分散され、通信ネッ

トワークによって互いに接続された１つのコンピューターまたは複数のコンピューターの上で実行されるように展開され得る。

【０２００】

[0214]方法ステップが、入力データを操作すること、および出力を生成することによって機能を実行するようコンピュータープログラムを実行する１つまたは複数のプログラマブルプロセッサーによって実行され得る。１つまたは複数のプログラマブルプロセッサーは、命令を並行に実行することが可能であり、かつ／または分散処理のために分散構成で配置されることが可能である。また、本明細書で説明される例示的な機能は、ハードウェア論理コンポーネントによって実行されることも可能であり、装置は、少なくとも部分的に、１つまたは複数のハードウェア論理コンポーネントとして実施されることが可能である。例えば、限定なしに、使用され得るハードウェア論理コンポーネントの例示的なタイプには、フィールド・プログラマブル・ゲート・アレイ（ＦＰＧＡ）、Ｐｒｏｇｒａｍ－ｓｐｅｃｉｆｉｃ Ｉｎｔｅｇｒａｔｅｄ Ｃｉｒｃｕｉｔ（ＡＳＩＣ）、Ｐｒｏｇｒａｍ－ｓｐｅｃｉｆｉｃ Ｓｔａｎｄａｒｄ Ｐｒｏｄｕｃｔ（ＡＳＳＰ）、Ｓｙｓｔｅｍ－ｏｎ－ａ－ｃｈｉｐシステム（ＳＯＣ）、Ｃｏｍｐｌｅｘ Ｐｒｏｇｒａｍｍａｂｌｅ Ｌｏｇｉｃ Ｄｅｖｉｃｅ（ＣＰＬＤ）などが含まれ得る。

【０２０１】

[0215]コンピュータープログラムの実行に適したプロセッサーには、例として、汎用マイクロプロセッサーおよび専用マイクロプロセッサー、ならびに任意の種類のデジタルコンピューターの任意の１つまたは複数のプロセッサーが含まれる。一般に、プロセッサーは、読取り専用メモリーから、またはランダムアクセスメモリーから、または読取り専用メモリーとランダムアクセスメモリーの両方から命令およびデータを受け取る。コンピューターの要素には、命令を実行するための少なくとも１つのプロセッサー、ならびに命令およびデータを記憶するための１つまたは複数のメモリーデバイスが含まれ得る。一般に、コンピューターは、データを記憶するための１つまたは複数の大容量ストレージデバイス、例えば、磁気ディスク、光磁気ディスク、または光ディスクを含むことも可能であり、あるいはそのような大容量ストレージデバイスからデータを受け取るように、もしくはそのような大容量ストレージデバイスにデータを送るように、またはその両方を行うように動作上、結合されることも可能である。コンピュータープログラム命令およびデータを実体化するのに適した情報キャリアには、例として、半導体メモリーデバイス、例えば、ＥＰＲＯＭ、ＥＥＰＲＯＭ、およびフラッシュメモリーデバイス、磁気ディスク、例えば、内部ハードディスクもしくはリムーバブルディスク、光磁気ディスク、ならびにＣＤ ＲＯＭディスクおよびＤＶＤ－ＲＯＭディスクを含め、すべての形態の不揮発性メモリーが含まれる。プロセッサーおよびメモリーは、専用論理回路によって捕捉されること、または専用論理回路に組み込まれることが可能である。

【０２０２】

[0216]ユーザーとの対話を可能にするのに、実施形態は、ユーザーに情報を表示するためのディスプレイデバイス、例えば、陰極線管（ＣＲＴ）、液晶ディスプレイ（ＬＣＤ）、またはプラズマモニターと、ユーザーがコンピューターに入力を与えることができるキーボードおよびポインティングデバイス、例えば、マウスもしくはトラックボールとを有するコンピューター上で実施され得る。他の種類のデバイスが、ユーザーとの対話を可能にするのに使用されることも可能であり、例えば、ユーザーに与えられるフィードバックは、任意の形態の知覚フィードバック、例えば、視覚フィードバック、聴覚フィードバック、または触覚フィードバックであり得る。例えば、視覚出力（例えば、視覚的ジェスチャー、ビデオ出力）、オーディオ出力（例えば、音声、デバイスサウンド）、触覚出力（例えば、タッチ、デバイスの動き）、温度、匂いなどを含め（ただし、以上には限定されない）、任意の形態の知覚出力を介して与えられ得る。

【０２０３】

[0217]さらに、ユーザーからの入力は、音響入力、音声入力、または触覚入力を含め、任意の形態で受け取られることが可能である。例えば、入力は、視覚入力（例えば、ジェ

スチャー、ビデオ入力）、オーディオ入力（例えば、音声、デバイスサウンド）、触覚入力（例えば、タッチ、デバイスの動き）、温度、匂いなどを含め（ただし、以上には限定されない）、任意の形態の知覚入力を介してユーザーから受け取られ得る。

【０２０４】

[0218]さらに、ナチュラルユーザーインターフェース（ＮＵＩ）が、ユーザーとインターフェースをとるのに使用され得る。この脈絡において、「ＮＵＩ」とは、ユーザーが、マウス、キーボード、リモコンなどの入力デバイスによって課せられる人為的な制約を免れて、「自然な」様態でデバイスと対話することを可能にする任意のインターフェース技術を指すことが可能である。

【０２０５】

[0219]ＮＵＩ技法の例には、音声認識、タッチ認識およびスタイラス認識、画面上のジェスチャー認識と画面に隣接したジェスチャー認識の両方、エアジェスチャー、頭部および目追跡、音声および発話、視覚、タッチ、ジェスチャー、ならびにマシンインテリジェンスに依拠する技法が含まれ得る。例示的なＮＵＩ技術には、すべてよりナチュラルなインターフェースを提供することが可能な、タッチセンシティブディスプレイ、音声および発話認識、意図および目標理解、デプスカメラ（例えば、立体カメラシステム、赤外線カメラシステム、ＲＧＢ（赤、緑、青）カメラシステム、および以上の組合せ）を使用するモーションジェスチャー検出、加速度計／ジャイロスコープを使用するモーションジェスチャー検出、顔認識、３Ｄディスプレイ、頭部、目、および視線追跡、没入型拡張現実システムおよび仮想現実システム、ならびに電界検出電極を用いた脳活動を検知するための技術（例えば、脳波計（ＥＥＧ）技法および関連する技法）が含まれ得るが、以上には限定されない。

【０２０６】

[0220]実施形態は、例えば、データサーバーとしてバックエンドコンポーネントを含む、または、ミドルウェアコンポーネント、例えば、アプリケーションサーバーを含む、またはフロントエンドコンポーネント、例えば、ユーザーが実施形態と対話することができるグラフィカルユーザーインターフェースまたはウェブブラウザを有するクライアントコンピューターを含む、あるいはそのようなバックエンドコンポーネント、ミドルウェアコンポーネント、またはフロントエンドコンポーネントの任意の組合せを含むコンピューティングシステムとして実施され得る。コンポーネントは、任意の形態または媒体のデジタルデータ通信、例えば、通信ネットワークによって互いに接続され得る。通信ネットワークの例には、ローカルエリアネットワーク（ＬＡＮ）およびワイドエリアネットワーク（ＷＡＮ）、例えば、インターネットが含まれる。

【０２０７】

[0221]主題は、構造上の特徴および／または方法上の動作に特有の言い回しで説明されてきたが、添付の特許請求の範囲において規定される主題は、以上に説明される特定の特徴または動作に必ずしも限定されないことを理解されたい。むしろ、以上に説明される特定の特徴および動作は、特許請求の範囲を実施する例示的な形態として開示される。説明される実施形態のいくつかの特徴が、本明細書で記載されるとおり例示されてきたが、今や、多くの変形形態、置換形態、変更形態、および均等形態が当業者には想起されよう。したがって、添付の特許請求の範囲は、実施形態の範囲に含まれるすべてのそのような変形形態および変更形態に及ぶものとする。

【図１】

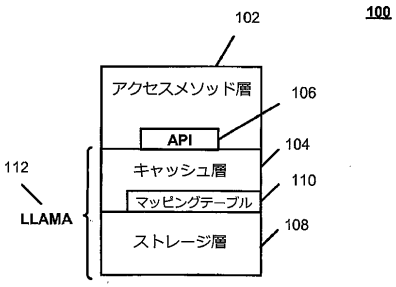

FIG. 1

【図２】

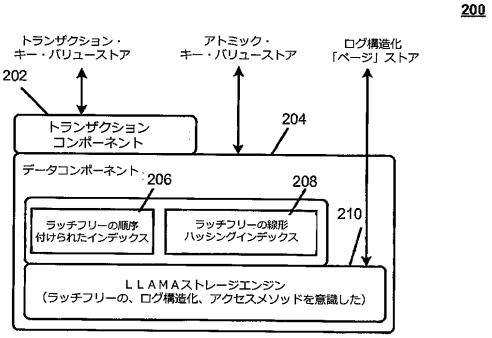

FIG. 2

【図３】

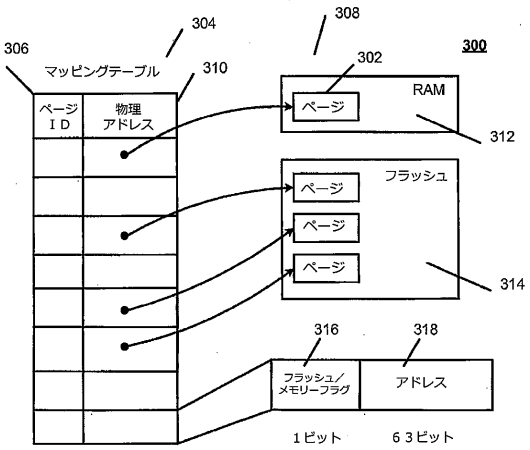

FIG. 3

【図４ａ】

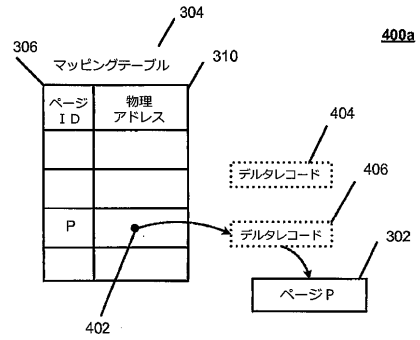

FIG. 4a

【図４ｂ】

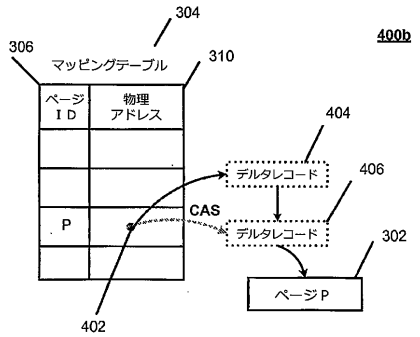

FIG. 4b

【図５】

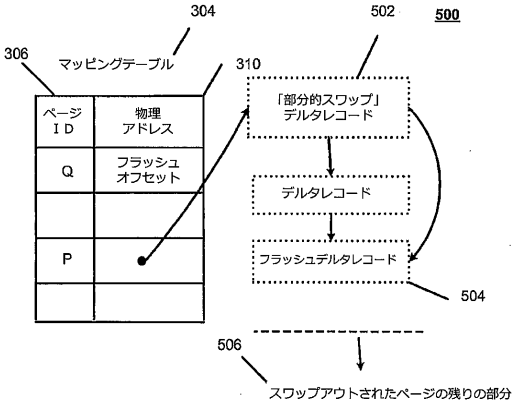

FIG. 5

【図６】

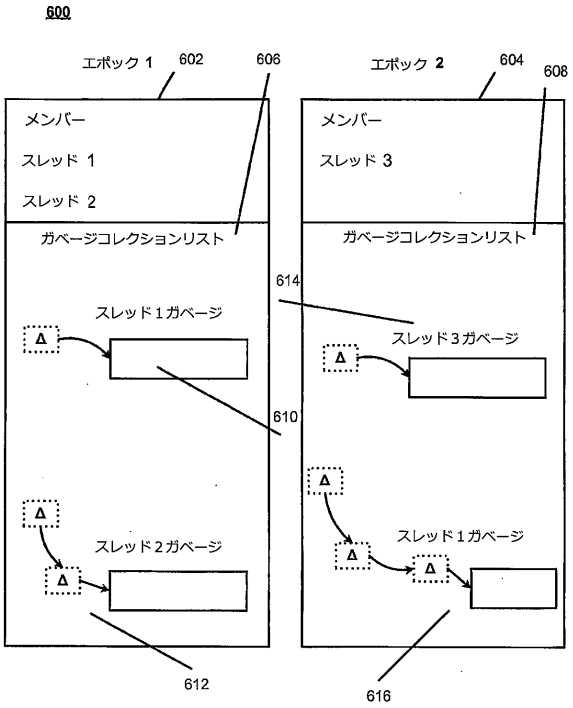

FIG. 6

【図７ａ】

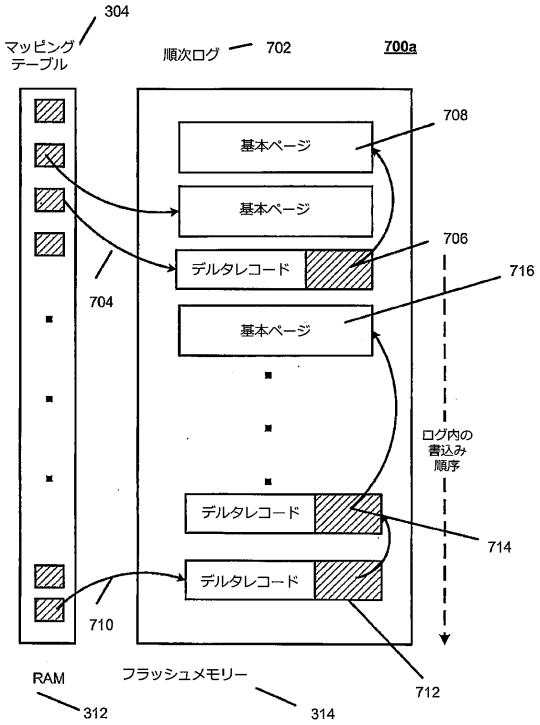

FIG. 7a

【図７ｂ】

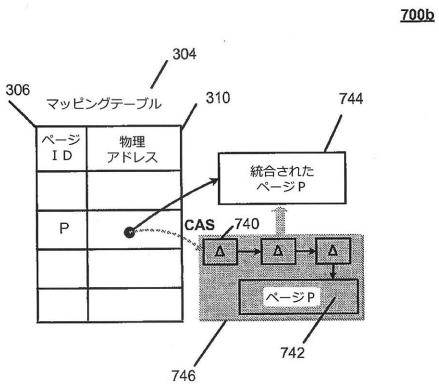

FIG. 7b

【図７ｃ】

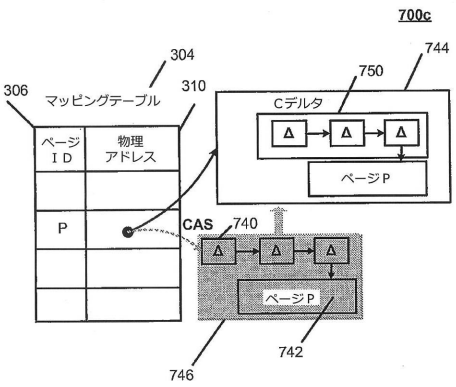

FIG. 7c

【図８】

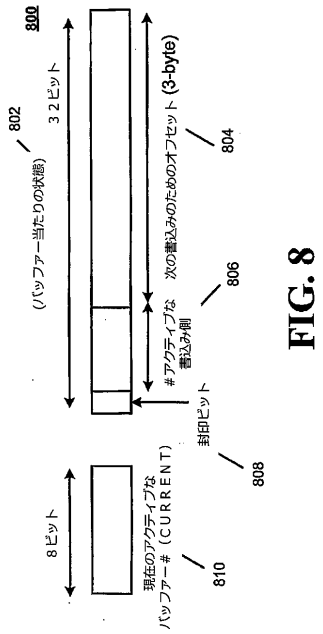

FIG. 8

【図９】

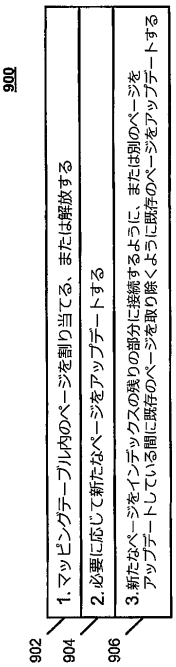

FIG. 9

【図１０】

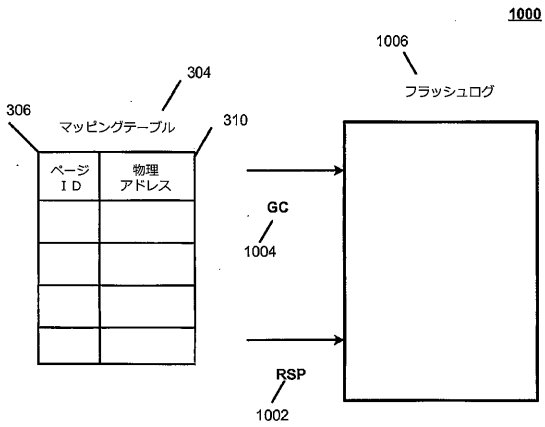

FIG. 10

【図１１】

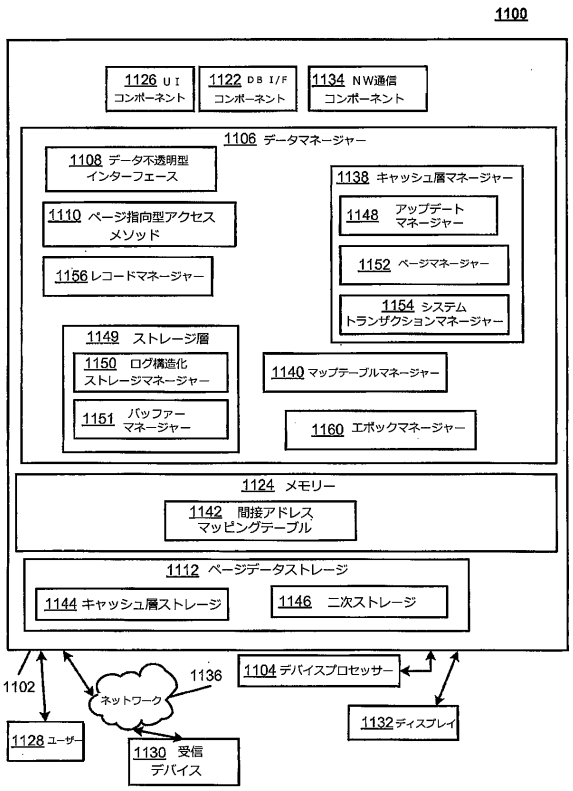

FIG. 11

【図１２ａ】

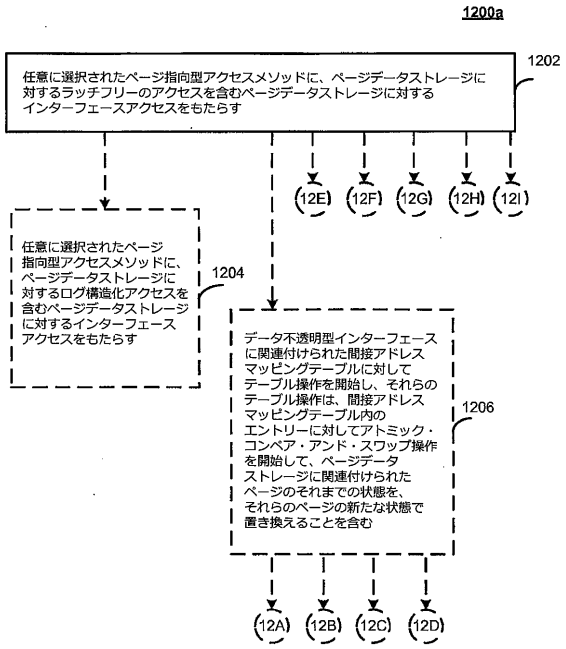

FIG. 12a

【図１２ｂ】

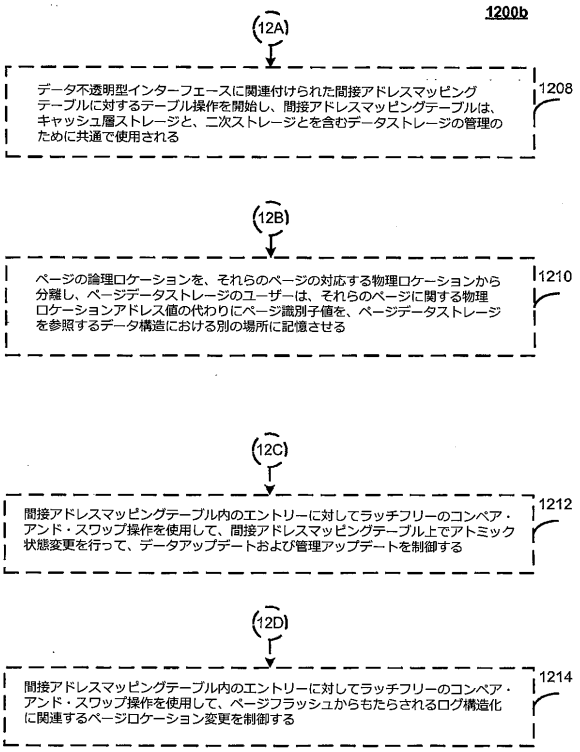

FIG. 12b

【図１２ｃ】

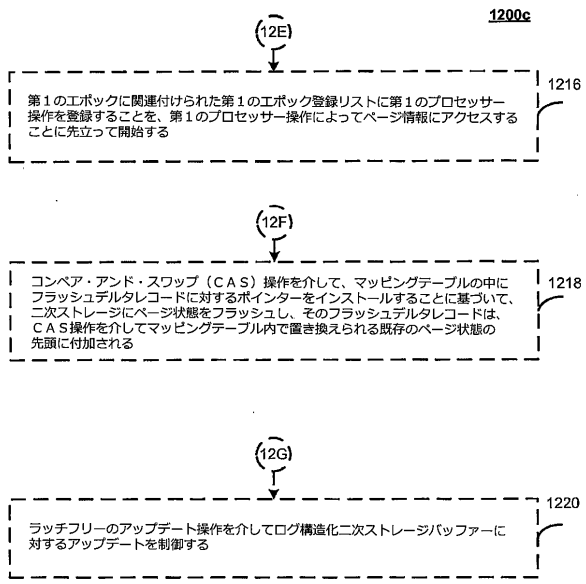

FIG. 12c

【図１２ｄ】

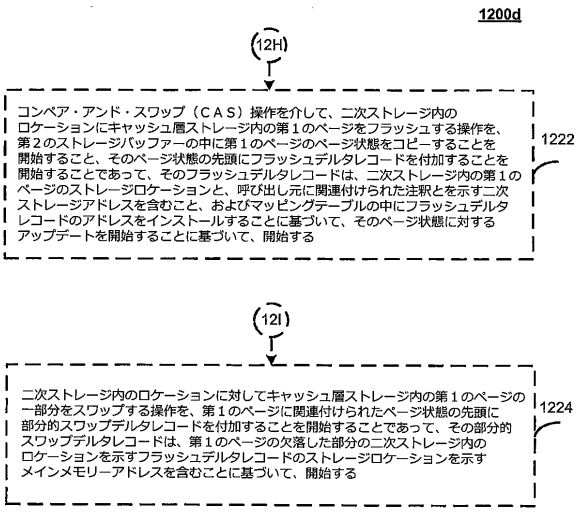

FIG. 12d

フロントページの続き

(51)Int.Cl. FI
G06F 12/02 520A

(74)代理人 100162846
弁理士 大牧 綾子
(72)発明者 ロメット, デーヴィッド・ビー
アメリカ合衆国ワシントン州９８０５２－６３９９, レッドモンド, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ（８／１１７２）
(72)発明者 レヴァンドスキ, ジャスティン
アメリカ合衆国ワシントン州９８０５２－６３９９, レッドモンド, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ（８／１１７２）
(72)発明者 セングプタ, スディプタ
アメリカ合衆国ワシントン州９８０５２－６３９９, レッドモンド, マイクロソフト コーポレーション, エルシーエイ－インターナショナル・パテンツ（８／１１７２）

審査官 田中 啓介

(56)参考文献 特表２００２－５２４８０１（ＪＰ, Ａ）
特開２０１０－２１８５２９（ＪＰ, Ａ）
米国特許出願公開第２００８／００６５６７０（ＵＳ, Ａ１）
米国特許出願公開第２００９／０２４０６６４（ＵＳ, Ａ１）
米国特許出願公開第２０１０／０１９１７１３（ＵＳ, Ａ１）
特開２００７－０１２０６１（ＪＰ, Ａ）
JUSTIN J LEVANDOSKI; DAVID B LOMET; SUDIPTA SENGUPTA, THE BW-TREE: A B-TREE FOR NEW HARDWARE PLATFORMS, DATA ENGINEERING (ICDE), 2013 IEEE 29TH INTERNATIONAL CONFERENCE ON, [ONLINE], IEEE, ２０１３年 ４月 ８日, pp.302 - 313, ＵＲＬ, http://dx.doi.org/10.1109/ICDE.2013.6544834

(58)調査した分野(Int.Cl., ＤＢ名)
Ｇ０６Ｆ３／０６－３／０８
Ｇ０６Ｆ１２／００－１２／１２８
Ｇ０６Ｆ１３／００－１３／１８、１７／３０