

(19)日本国特許庁(JP)

(12)公表特許公報(A)

(11)公表番号
特表2024-519674
(P2024-519674A)

(43)公表日 令和6年5月21日(2024.5.21)

(51)国際特許分類
G 0 6 F 16/28 (2019.01)

F I
G 0 6 F 16/28

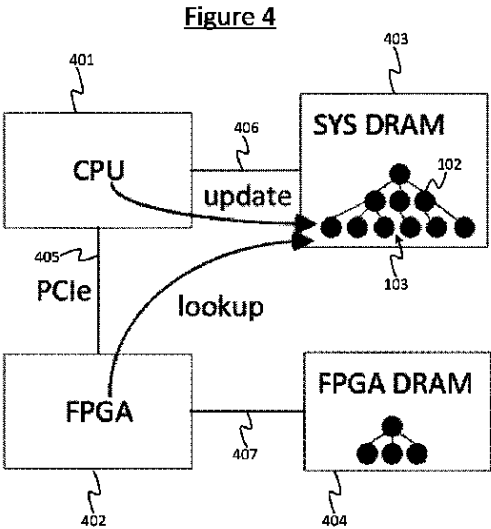
テーマコード (参考)
5 B 1 7 5

審査請求 未請求 予備審査請求 未請求 (全31頁)			
(21)出願番号	特願2023-564186(P2023-564186)	(71)出願人	314015767
(86)(22)出願日	令和4年5月18日(2022.5.18)		マイクロソフト テクノロジー ライセン
(85)翻訳文提出日	令和5年12月13日(2023.12.13)		シング, エルエルシー
(86)国際出願番号	PCT/US2022/029722		アメリカ合衆国 ワシントン州 9 8 0 5
(87)国際公開番号	WO2022/251009		2 レッドモンド ワン マイクロソフト
(87)国際公開日	令和4年12月1日(2022.12.1)		ウェイ
(31)優先権主張番号	21175921.2	(74)代理人	100079108
(32)優先日	令和3年5月26日(2021.5.26)		弁理士 稲葉 良幸
(33)優先権主張国・地域又は機関	欧州特許庁(EP)	(74)代理人	100109346
(81)指定国・地域	AP(BW,GH,GM,KE,LR,LS,MW,MZ,NA ,RW,SD,SL,ST,SZ,TZ,UG,ZM,ZW),EA(AM,AZ,BY,KG,KZ,RU,TJ,TM),EP(AL,A T,BE,BG,CH,CY,CZ,DE,DK,EE,ES,FI,FR ,GB,GR,HR,HU,IE,IS,IT,LT,LU,LV,MC, 最終頁に続く	(74)代理人	100117189
			弁理士 大貫 敏史
		(74)代理人	100134120
			弁理士 江口 昭彦
		(74)代理人	100108213
			弁理士 内藤 和彦
			最終頁に続く

(54)【発明の名称】 ツリーベースのデータ構造

(57)【要約】

ライタは、ツリーのリーフノードにアイテムを書き込み、リーダは、リーフノードからアイテムを読み出す。各ノードは、それぞれの第1のブロック及び第2のブロックを備え、第1のブロックは、キーの順序でソートされたそれぞれのリーフの複数のアイテムを備える。ライタは、リーフに新しいアイテムを書き込む際に、識別されたリーフノードの第2のブロックに対して、キーの順序でソートされるのではなく、書き込まれた順序で新しいアイテムを書き込む。リーダが、リーフから1つ又は複数のターゲットアイテムを読み出す際に、a) 第1のブロックで既にソートされたアイテムの順序と、b) リーダが第1のブロックのアイテムに関連したキーによって第2のブロックのアイテムをソートすることに基づいて、1つ又は複数のターゲットアイテムについてリーフを検索する。



【特許請求の範囲】

【請求項 1】

データ構造を格納するメモリであって、前記データ構造が、それぞれノード ID を有する複数のノードを備えたツリー構造を備え、一部のノードがリーフノードであり、他のノードが内部ノードであり、各内部ノードが、前記ツリー構造内の 1 つ又は複数の子のそれぞれのセットの親であり、各子が、リーフノード又は別の内部ノードの何れかであり、各リーフノードが、子であるが親ではなく、前記リーフノードの各々が、それぞれキーバリュペアを備えた 1 つ又は複数のアイテムのそれぞれのセットを備え、前記内部ノードの各々が、それぞれの子の各々の前記ノード ID を、前記それぞれの子によって包含されるキーの範囲にマッピングする、メモリと、

10

前記リーフノードにアイテムを書き込むように配置されたライタと、

前記リーフノードからアイテムを読み出すように配置されたリーダと、

を備えたシステムであって、

前記リーフノードの各々が、それぞれの第 1 のブロックと、それぞれの第 2 のブロックとを備え、前記第 1 のブロックが、前記メモリのアドレス空間においてキーの順序でソートされた前記それぞれのリーフノードの複数の前記アイテムを備え、

前記ライタが、前記ツリー構造に新しいアイテムを書き込む際に、前記ツリー構造を通して、ノード ID に対するキーの前記マッピングをたどることによって、どのリーフノードに書き込むべきかを識別し、前記識別されたリーフノードの前記第 2 のブロックに対して、キーの順序でソートされるのではなく、書き込まれた順序で前記新しいアイテムを書き込むように構成され、

20

前記リーダが、前記ツリー構造から 1 つ又は複数のターゲットアイテムを読み出す際に、前記ツリー構造を通して、ノード ID に対するキーの前記マッピングをたどることによって、どのリーフノードから読み出すべきかを決定し、次に、a) 前記第 1 のブロックで既にソートされた前記アイテムの前記順序と、b) 前記リーダが前記第 1 のブロックの前記アイテムに関連したキーによって前記第 2 のブロックの前記アイテムをソートすることに基づいて、前記 1 つ又は複数のターゲットアイテムについて前記決定されたリーフノードを検索するように構成される、

システム。

【請求項 2】

30

前記ライタが、前記システムのコンピュータ可読ストレージに格納され、前記システムの 1 つ又は複数のプロセッサ上で実行されるように配置されたソフトウェアにおいて実装される、請求項 1 に記載のシステム。

【請求項 3】

前記リーダが、PGA、FPGA、又は専用ハードウェア回路網において実装される、請求項 1 又は 2 に記載のシステム。

【請求項 4】

前記ツリー構造が、B ツリーの形態を取る、請求項 1 ～ 3 の何れか一項に記載のシステム。

【請求項 5】

40

前記 B ツリーが、アイテムが中間ノードに格納されない B + ツリーである、請求項 4 に記載のシステム。

【請求項 6】

前記リーフノード及び／又は内部ノードの少なくとも一部において、前記それぞれの第 1 のブロックが、セグメントに分けられ、前記ノードが、複数のショートカットを含むショートカットブロックをさらに備え、各ショートカットが、i) 前記それぞれの第 1 のブロックの対応するセグメントの境界における前記キーのインジケーションと、ii) 前記リーフノード内の前記対応するセグメントに対するオフセットとを備え、

前記リーダが、第 1 のブロックの前記検索を行う際に、キーに基づいて前記 1 つ又は複数のアイテムを含有する可能性のあるセグメントのみを検索するために前記ショートカッ

50

トを使用するように構成される、請求項 1～5 の何れか一項に記載のシステム。

【請求項 7】

前記ライタが、前記リーフノードの少なくとも 1 つに書き込む際に、前記それぞれの第 2 のブロックにおける各アイテムに関連付けられたバックポインタを書き込み時に含めるように構成され、各バックポインタが、前記第 2 のブロック内の前記関連付けられたアイテムと比較して次に大きいキー又は次に小さいキーを持つ前記第 1 のブロック内のアイテムに対するオフセットを備え、

前記リーダが、第 2 のブロックの前記ソートを行う際に、前記それぞれの第 1 のブロックに関連したキーによって前記第 2 のブロックの前記アイテムをソートするために、前記バックポインタを使用するように構成される、請求項 1～6 の何れか一項に記載のシステム。

10

【請求項 8】

前記ライタが、前記リーフノードの少なくとも 1 つに書き込む際に、前記第 2 のブロック内の前記アイテムの各々について、前記第 2 のブロックに前記アイテムを書き込む時点における、前記第 2 のブロックの前記現在のキーに対する前記アイテムのキーの順序を示す順序ヒントを書き込み時に含めるように構成され、

前記リーダが、第 2 のブロックの前記ソートを行う際に、前記それぞれの第 1 のブロックに関連したキーによって前記第 2 のブロックの前記アイテムをソートするために、前記順序ヒントを使用するように構成される、請求項 1～7 の何れか一項に記載のシステム。

【請求項 9】

20

前記リーダが、

ー前記それぞれの第 2 のブロック内の前記アイテムを、それらが格納された順序で処理することと、

ーアレイ内の前記アイテムをソートするために前記順序ヒントを使用することであって、前記アレイが、前記第 2 のブロック内の前記アイテムのオフセットを格納する、ことと、

によって前記順序ヒントを使用するように構成され、

順序ヒント i を持つ各アイテムが処理される際に、前記第 2 のブロック内のそのオフセットが、前記アレイ内の位置 i に挿入され、前記第 2 のブロック全体が処理された後に、前記アレイが前記ソートされた順序で前記アイテムのオフセットを含有するように、位置 $j \geq i$ にある前記アイテムのすべてが右に移動される、請求項 8 に記載のシステム。

30

【請求項 10】

前記データ構造が、前記ノード ID をそれぞれのメモリアドレスにマッピングするページテーブルを備え、

前記リーダが、決定されたノード ID のノードから読み出す際に、メモリ内の前記決定されたノードの前記メモリアドレスを決定するために、前記ページテーブルを使用するように構成され、

前記ライタが、ノードの書き込み、分割、又はマージを行うために前記ノードを更新する際に、前記ノードの更新されたバージョンを新しいメモリアドレスに書き込み、一旦書き込まれると、前記新しいメモリアドレスを指すように前記ページテーブル内の前記それぞれのメモリアドレスを更新するように構成される、請求項 1～9 の何れか一項に記載のシステム。

40

【請求項 11】

各第 2 のブロックが、最大サイズを有し、

前記ライタが、前記識別されたノードの前記それぞれの第 2 のブロックが前記最大サイズを超えることを生じさせる新しいアイテムを書き込む場合、前記第 2 のブロックを前記それぞれの第 1 のブロックにマージするように構成される、請求項 1～10 の何れか一項に記載のシステム。

【請求項 12】

前記データ構造が、前記ノード ID をそれぞれのメモリアドレスにマッピングするページ

50

ジテーブルを備え、

前記リーダが、決定されたノードIDのノードから読み出す際に、メモリ内の前記決定されたノードの前記メモリアドレスを決定するために、前記ページテーブルを使用するように構成され、

前記ライタが、前記それぞれの第2及び第1のブロックをマージするためにノードを更新する際に、前記マージを含む前記ノードの更新されたバージョンを新しいメモリアドレスに書き込み、一旦前記マージが完了すると、前記新しいメモリアドレスを指すように前記ページテーブル内の前記それぞれのメモリアドレスを更新するように構成される、請求項11に記載のシステム。

【請求項13】

各リーフノードが、前記それぞれのリーフノードの前記キーに隣接するキーの範囲を包含する1つ又は複数の兄弟リーフノードの前記ノードIDをさらに示し、

前記リーダが、キーの範囲からアイテムを読み出すための範囲スキャンを行うように構成され、2つ以上のリーフノードに及ぶアイテムの範囲から読み出す範囲スキャンを行う際に、

－前記ツリー構造を通して、ノードIDに対するキーの前記マッピングをたどることによって、前記スキャンされた範囲の前記キーの1つを包含する前記リーフノードの1つを決定することと、

－前記リーフノードの前記1つにおいて示される前記兄弟リーフノードの少なくとも1つの前記IDを使用することによって、前記スキャンされた範囲の少なくとも1つの他のキーを包含する少なくとも1つの他のリーフノードを決定することと、

を行うように構成される、請求項1～12の何れか一項に記載のシステム。

【請求項14】

データ構造をメモリに格納することであって、前記データ構造が、それぞれノードIDを有する複数のノードを備えたツリー構造を備え、一部のノードがリーフノードであり、他のノードが内部ノードであり、各内部ノードが、前記ツリー構造内の1つ又は複数の子のそれぞれのセットの親であり、各子が、リーフノード又は別の内部ノードの何れかであり、各リーフノードが、子であるが親ではなく、前記リーフノードの各々が、それぞれキーバリュースペアを備えた1つ又は複数のアイテムのそれぞれのセットを備え、前記内部ノードの各々が、それぞれの子の各々の前記ノードIDを、前記それぞれの子によって包含されるキーの範囲にマッピングする、ことと、

前記リーフノードにアイテムを書き込むことと、

前記リーフノードからアイテムを読み出すことと、

を備えた方法であって、

前記リーフノードの各々が、それぞれの第1のブロックと、それぞれの第2のブロックとを備え、前記第1のブロックが、前記メモリのアドレス空間においてキーの順序でソートされた前記それぞれのリーフノードの複数の前記アイテムを備え、

前記書き込むことが、前記ツリー構造に新しいアイテムを書き込む際に、前記ツリー構造を通して、ノードIDに対するキーの前記マッピングをたどることによって、どのリーフノードに書き込むべきかを識別することと、前記識別されたリーフノードの前記第2のブロックに対して、キーの順序でソートされるのではなく、書き込まれた順序で前記新しいアイテムを書き込むことと、を備え、

前記読み出すことが、前記ツリー構造から1つ又は複数のターゲットアイテムを読み出す際に、前記ツリー構造を通して、ノードIDに対するキーの前記マッピングをたどることによって、どのリーフノードから読み出すべきかを決定することと、次に、a) 前記第1のブロックで既にソートされた前記アイテムの前記順序と、b) 前記リーダが前記第1のブロックの前記アイテムに関連したキーによって前記第2のブロックの前記アイテムをソートすることとに基づいて、前記1つ又は複数のターゲットアイテムについて前記決定されたリーフノードを検索することと、を備える、方法。

【請求項15】

1つ又は複数の非一時的なコンピュータ可読媒体上に具現化されたコンピュータプログラムであって、1つ又は複数のプロセッサ上で実行される際に、請求項14に記載の前記書き込み及び／又は読み出しを行うように構成されたコードを備えた、コンピュータプログラム。

【発明の詳細な説明】

【背景技術】

【0001】

背景

Bツリーなどのツリー構造は、キーバリューペア (key-value pair) (各ペアは、格納される内容である値と、その値をインデックスするために使用されるキーとを備える) を格納するためのインデックスデータ構造として使用することができる。ツリーの各リーフノードには、キーバリューペアである複数のアイテムが含有される。リーフからツリーをさらに上へ進むと、ツリーの各内部ノードには、その子 (リーフ又は他の内部ノードである可能性がある) の各々によって包含されるキー値の範囲のインジケーションが含有される。新しいアイテムがツリーに追加される場合に、ライタは、ツリー構造を使用して、新しいアイテムのキーが入るキーの範囲をどのリーフが含有するかを決定する。このリーフが一杯でなければ、新しいアイテムは、単に既存のリーフに追加することができる。しかし、リーフが一杯である場合は (リーフ及び内部ノードは、通常バイト単位の最大サイズを有する)、リーフは分割される。すなわち、新しいリーフが作成される。これは、古いリーフ及び新しいリーフのキー範囲を正しく参照するように、これら2つのリーフの親ノードを更新することも意味する。親に対するこの更新によって、親がその最大サイズを超える場合は、その親自体が分割され、祖父母の参照が更新されるなどする。アイテムが削除される場合は、これは、リーフ又は内部ノードをマージすることも伴う可能性がある。

【0002】

あるアプリケーションにおいて、ライタ又はリーダが行うことを必要とされ得る操作が幾つかある。リーダは、特定のキーを持つ個々のアイテムを読み出すことができ、又は範囲スキャン (range scan) を行ってキーの範囲から読み出すことができる。ライタは、新しいエントリを書き込んだり、既存のエントリを修正したり、又はエントリを削除したりすることができる。

【0003】

インデックスデータ構造は、例えばデータストア及びデータベース、ファイルシステム、オペレーティングシステムなどの広範なソフトウェアシステムで使用される。インデックスデータ構造は、キーバリューペアを格納し、それは、数ある操作の中でも、キーバリューペアのルックアップ、スキャン、挿入、及び削除をサポートする。Bツリーは、そのようなインデックスデータ構造の1つである。Bツリーは、整理されたインデックスであり、つまり、それは、スキャン操作もサポートする。スキャン操作は、ツリーに格納されている、指定された範囲内のキーを持つすべてのキーバリューペアを返す。

【発明の概要】

【0004】

概要

Bツリーなどのツリーベースのデータ構造では、ライタとリーダとの間のワークロードのバランスをとることが望ましい。例えば、新しいエントリが単に時系列で追加され、キーによってソートされていない場合は、これは、ライタにとっては非常に速い。しかしリーダは、個々の読み出し又は範囲スキャンを行うために、読み出し時にアイテムをソートしなければならない (少なくとも範囲スキャンは、ソートを必要とする個々の読み出しは、単に、ルックアップ及び範囲スキャンの両方を、ソートを使用して同じように実装する方が簡単である)。一方、ライタが、書き込みを行うたびに、すべての新しいエントリをソートされた順序に並べる場合、これはリーダにとって読み出しを非常に速くするが、書き込み時にライタに対してより大きな負担をかける。これら2つのアプローチ間の妥協

10

20

30

40

50

点を提供することが望ましい。

【0005】

本明細書に開示される一態様によれば、データ構造を格納するメモリを備えたシステムが提供され、このデータ構造は、それぞれノードIDを有する複数のノードを備えたツリー構造を備える。一部のノードは、リーフノードであり、他のノードは、内部ノードであり、各内部ノードは、ツリー構造内の1つ又は複数の子のそれぞれのセットの親である。各子は、リーフノード又は別の内部ノードの何れかであり、各リーフノードは、子であるが親ではない。リーフノードの各々は、それぞれキーバリュペアを備えた1つ又は複数のアイテムのそれぞれのセットを備え、内部ノードの各々は、それぞれの子の各々のノードIDを、それぞれの子によって包含されるキーの範囲にマッピングする。システムはさらに、ソフトウェア、ハードウェア、又はそれらの組み合わせにおいて実装され、リーフノードにアイテムを書き込むように配置されたライタと、ソフトウェア、ハードウェア、又はそれらの組み合わせにおいて実装され、リーフノードからアイテムを読み出すように配置されたリーダとを備える。リーフノードの各々は、それぞれの第1のブロックと、それぞれの第2のブロックとを備え、第1のブロックが、メモリのアドレス空間においてキーの順序でソートされたそれぞれのリーフノードの複数のアイテムを備える。ライタは、ツリー構造に新しいアイテムを書き込む際に、ツリー構造を通して、ノードIDに対するキーのマッピングをたどることによって、どのリーフノードに書き込むべきかを識別し、識別されたリーフノードの第2のブロックに対して、キーの順序でソートされるのではなく、書き込まれた順序で新しいアイテムを書き込むように構成される。リーダは、ツリー構造から1つ又は複数のターゲットアイテムを読み出す際に、ツリー構造を通して、ノードIDに対するキーのマッピングをたどることによって、どのリーフノードから読み出すべきかを決定し、次に、a) 第1のブロックで既にソートされたアイテムの順序と、b) リーダが第1のブロックのアイテムに関連したキーによって第2のブロックのアイテムをソートすることとに基づいて、1つ又は複数のターゲットアイテムについて決定されたリーフノードを検索するように構成される。

10

20

【0006】

実施形態では、第1のブロックは、第2のブロックよりも大きな最大サイズを有する場合があり、よって、第1のブロックは「大ブロック」と呼ばれ、第2のブロックは「小ブロック」と呼ばれる場合がある。実施形態では、ライタは、ソフトウェアにおいて実装され得るが、リーダは、カスタムハードウェア、例えばプログラマブルゲートアレイ（PGA）又はフィールドプログラマブルゲートアレイ（FPGA）で実装されてもよい。しかし、より一般的には、ライタは、ハードウェア（例えば、ハードウェアアクセラレータ）において実装され得、及び／又はリーダは、ソフトウェアにおいて実装され得る。

30

【0007】

図面の簡単な説明

本明細書に開示される実施形態の理解を助け、そのような実施形態がどのように実施され得るかを説明するために、単なる例として、以下の図面を参照する。

【図面の簡単な説明】

【0008】

40

【図1】例示的なキー区間[14, 22]に対する包括的スキャン（inclusive scan）（返されたアイテムのキーによって形成される区間が、入力区間内に入る最大のサブセットである）の形態の範囲スキャンを概略的に図示する。

【図2】例示的なキー区間[14, 22]に対するカバリングスキャン（covering scan）（返されたアイテムのキーによって形成される区間が、入力区間を含有する最小の上位セットである）の形態の範囲スキャンを概略的に図示する。

【図3】例示的なキー区間（14, 22）に対するカバリングスキャンの形態の範囲スキャンを概略的に図示する。

【図4】本明細書に開示される実施形態によるシステムの概略ブロック図である。

【図5】ツリー構造を備えたデータ構造を概略的に図示する。

50

- 【図 6】小ブロック及び大ブロックを備えたノードを概略的に図示する。
- 【図 7】小ブロックに新しいアイテムを挿入する方法を概略的に図示する。
- 【図 8】小ブロックをソートするために順序ヒント (order hint) を含める方法を概略的に図示する。
- 【図 9】順序ヒントに基づいて小ブロックをソートする方法を概略的に図示する。
- 【図 10】4 要素小ブロックソータの概略回路図である。
- 【図 11】挿入操作中に大ブロックと小ブロックとをマージする方法を概略的に図示する。
- 【図 12】挿入時にリーフノードを分割する方法を概略的に図示する。

10

【発明を実施するための形態】

【0009】

実施形態の詳細な説明

本開示は、各ノードが第 1 及び第 2 のブロックから成るノードレイアウトを提供する。好ましくは、第 1 のブロックは、第 2 のブロックよりも大きな最大サイズ（ビット又はバイト単位）を有し、よって、第 1 及び第 2 のブロックは、それぞれ大ブロック及び小ブロックと呼ばれることがある。下記は、例として大ブロック及び小ブロックを参照して説明されるが、以下のどこでも、原理上は、これをより一般的にそれぞれ「第 1 のブロック」及び「第 2 のブロック」に置き換えることができる。

【0010】

大ブロックはソートされており、これにより、効率的なノードの検索が可能となる。この文脈においてソートされるとは、キーによるアイテムの順序が、メモリのアドレス空間（実装形態によっては仮想アドレス空間又は物理アドレス空間）においてアイテムが出現する順序と同じになるように、アイテムがキーによってソートされることを意味する。一方、小ブロックは、ノードに対する最新の挿入及び削除を備え、並びに時系列であり、これにより、高速な更新が可能となる。代替方法としては、ノード全体をソートされていない状態に保つことで、高速な更新性能がもたらされるが、検索性能は低下するか、或いはノード全体をソートされた状態に保つことで、高速な検索性能がもたらされるが、更新性能は低下する。

20

【0011】

任意選択的に、大ブロックにおいて、ショートカットキーが使用されてもよい。これは、ノード検索がノード全体のごく一部のみを調べることを可能にし、性能の向上がもたらされる。

30

【0012】

別の、代替の又は追加の最適化として、実施形態では、「バックポインタ」が小ブロックに含まれてもよい。これらは、キーを比較することなく、小ブロック及び大ブロックのアイテム間の順序を確立することを可能にする。これにより、キー全体を比較する必要がないため、（例えば、ハードウェアにおいて）効率的な実装形態がもたらされる。

【0013】

ショートカットキー及び／又はバックポインタと一緒に、又はそれらとは無関係に使用することが可能な別の任意選択的な最適化として、実施形態では、「順序ヒント」が小ブロックに含まれてもよい。小ブロック内の順序ヒントは、キーを比較することなく、小ブロック内のアイテムのソートされた順序を確立するための非常に効率的なハードウェアを可能にする。

40

【0014】

さらに別の任意選択的な最適化は、複雑な更新の同期を提供することである。この場合、ノードが分割又はマージされると、ライタは、サブツリーのコピー（例えば、スレッドプライベートコピー）を作成し、次に、ページテーブルの単一のポインタ更新でそれをスワップすることによってツリーを更新する。

【0015】

実施形態では、ライタは、1 つ又は複数のプロセッサ上で実行されるソフトウェアにお

50

いて実装されてもよいが、リーダは、特殊ハードウェアにおいて実装されてもよい。ハードウェアは、FPGA若しくはPGA、又はASIC（特定用途向け集積回路）などの専用（すなわち固定機能）回路網を備えていてもよい。

【0016】

インデックスデータ構造を実装する特殊ハードウェアは、データ構造演算を実行するための専用パイプラインを有するハードウェアである。これは、命令セットを実装し、その命令セットを使用してデータ構造演算を実装するプログラムを実行する汎用CPUとは対照的である。特殊ハードウェアは、幾つかの利点を有する。それは、ワット当たりのスループットがより高く、より予測可能で待ち時間がより少なく、CPU上でソフトウェアとして実行される同じ機能性よりも安価となり得る。特殊ハードウェアを使用する主な欠点は、設計がより難しいこと、ソフトウェアよりも構築に時間がかかること、専用ハードウェアの場合、構築後に変更できないため、通常、非常に広範に展開された機能性にのみ使用されることである。しかし、本開示の範囲は、ソフトウェアにおいて書き込みを実装し、及びハードウェアにおいて読み出しを実装することに限定されないことに留意されたい。他の可能性として、ライタは、FPGA若しくはASICなどの何らかの形態のハードウェアにおいて実装されてもよく、及び／又は、リーダは、ソフトウェアにおいて実装されてもよい。本明細書で開示する実施形態は、両方の長所から恩恵を受けるインメモリBツリーを提供する。すなわち、すべての操作をCPU上のソフトウェアにおいて実行することができ、ルックアップ及びスキャン操作は、我々が設計した特殊ハードウェア上で実行することができる。多くのワークロードにおいて、ルックアップ及びスキャンは、主要な操作であり、つまり、ほとんどの操作がハードウェアオフロードの恩恵を受ける。ルックアップ及びスキャンは更新操作よりも単純であり、つまり、システム全体をより高速に構築及び展開することができる。ある例示的な実装形態では、メモリは、DRAMベースのホストメモリでもよいが、これは限定するものではなく、他の例では、Bツリーは、例えばSSD又は3D-XpointドライブなどのストレージNVMデバイスに格納することができる。

【0017】

これより、上述した技法の例を、図を参照してより詳細に説明する。

【0018】

下記は、Bツリーの観点から説明されるが、より一般的には、異なるタイプの読み出し操作及び書き込み操作、大ブロック及び小ブロックの使用、ショートカットキー、バックポインタ、順序ヒント、及び／又は複雑な更新（分割及びマージ）の同期などの本明細書で開示するアイデアは、キーバリュースタアを格納するための任意のツリー構造において適用することができる。

【0019】

ツリー構造

Bツリーなどのキーバリュースタアは、キーバリュースタア（アイテムとも呼ばれる）としてデータを格納する。アイテムは可変サイズのものであってもよいが、実装形態によっては、データ構造は、アイテムの総許容サイズに制限を設けてもよく（例えば、512バイトまで）、又は幾つかの実装形態では、実装形態を単純にするためにそれらを固定サイズにしてもよい。各キーは、単一のそれぞれの値と関連付けられ、同じキーを持つ複数のキーバリュースタアは許可されない。

【0020】

図1、図4、及び図5は、例として、Bツリーなどのツリー構造103の要素を図示している。ツリー103は、メモリに実装されたデータ構造の一形態であり、これは、キーバリュースタアとして使用することができる。ツリー103は、複数のノード102を備え、そのうちの幾つかは、リーフノード102Lであり、他のものは、内部ノード102Iである。各ノード102は、本明細書において論理ID（LID）とも呼ばれ得る、それぞれのノードIDを有する。各リーフノード102Lは、1つ又は複数のアイテム101のそれぞれのセットを備え、各アイテム101は、キーバリュースタアを備える（実例と

して、図 1 は、範囲スキャンを受けているノード 102 を示しているが、これは限定するものではなく、読み出し操作のタイプの一例に過ぎない）。キーバリュースペアは、アイテムのインデックスとして機能し、アイテムの内容である値にマッピングされたキーを備える。図中のアイテム 101 の内部に示される数字は、例示的なキーである（実際には、キーは、はるかに大きな数字に達し得るが、これは単に説明のためである）。各内部ノード 102 I は、内部ノードの 1 つ又は複数の子のそれぞれのセットを指定する。ツリーが構築されると、内部ノードの少なくとも 1 つ又は幾つかは、それぞれ 2 つ以上の子を有することになる（ただし、ツリーにリーフが 1 つしかない場合、例えば、それが作成されたばかりのときは、ルートノードは、子を 1 つのみ有する）。各内部ノード 102 I は、キーのどの範囲がそれぞれの子の各々によって包含されるかも示す。後述する、ポイント間にキーが挟まれる図 5 に示す特定のスキームは、このマッピングを指定するための可能な一例に過ぎない。

10

【0021】

マッピングがどんな手段を用いて実装されるにせよ、内部ノード 102 I のノード ID は、このようにして、親ノードと子ノードとの間のエッジを定義し、それによってツリー構造を形成する。各内部ノード 102 I は、少なくとも 1 つのそれぞれの子の親である。ツリーが構築されると、内部ノードの少なくとも 1 つ又は幾つかの各々は、複数のそれぞれの子の親となる（ただし、上述したように、ツリーにリーフが 1 つしかない場合は、ルートは、単一の子を有する）。ある親の子は、リーフノード 102 L 又は別の内部ノード 102 I の何れかであり得る。リーフノード 102 L は、親ではなく、子のみである（すなわち、リーフは、ツリーの底部であり、別の言い方をすれば、ツリーの枝の終端である）。

20

【0022】

本明細書において、キーが子に「包含される」とは、子が、そのキーを持つアイテム（キーバリュースペア）を含有するリーフノード 102 であるか、或いは子が、ノード ID 対キーのマッピングによって、最終的に、そのキーを持つアイテムを含有するリーフノード 102 I に、ツリーの階層の 1 つ又は複数下のレベル（又は「世代」）でつながる別の内部ノード 102 I であることを意味する。

【0023】

ツリーの最上部にある内部ノード 102 I の 1 つがルートノードである。ライタ又はリーダーが特定のキーを持つアイテムに書き込みを行ったり、又はそのアイテムから読み出しを行ったりする場合、それは、ルートのどの子が必要とされるキーを包含するかを見つけるために、ルートノードで指定されたキー範囲に対するノード ID のマッピングをクエリすることから始める。その子自体が別の内部ノード 102 I である場合は、ライタ又はリーダーは、必要とされるキーを持つアイテムを含有するリーフノード 102 L を見つけるまで、そのノードで指定されたマッピングをクエリして、次の世代のどの子が必要とされるキーを包含するかを見つけるなどする。

30

【0024】

実施形態では、キー比較は、C memcmp 関数のセマンティクスに従ってもよく、つまり、整数比較のセマンティクスを維持するために、整数キーは、ビッグエンディアンフォーマットで格納されるべきである。しかし、これは、可能な一例に過ぎない。

40

【0025】

実施形態では、B ツリーは、B+ ツリーの形態を取ってもよく、これにより、リーフノード 102 L のみがキーバリュースペアを含有し、内部ノード 102 I のみが、ノード ID 対キー範囲のマッピングを含有する。

【0026】

読み出し／書き込み操作

B ツリーなどのツリー構造は、個々のルックアップ及び／又は範囲スキャンの読み出し操作タイプをサポートし得る。それは、挿入、修正、及び／又は削除の書き込み（更新）操作タイプをサポートする。このような操作の例示的なセマンティクスを以下に説明する

50

。

【0027】

ルックアップ：これは、引数としてキーを取り、ツリーにそのキーが存在すれば、そのキーに関連付けられた値を返す。そうでない場合は、それは、ステータス「見つかりません (not-found)」を返す。

【0028】

スキャン：これは、2つのキーを入力引数として取り、閉区間 [low-key, high-key] 内のキーを持つツリーからすべてのキーバリューペアを返す。実施形態では、包括的スキャン及びカバリングスキャンの2種類のスキャンがサポートされる。図1、図2、及び図3は、その違いを図示している。(l, u) は、下界 l 及び上界 u がツリー内にあっても、それらが結果に含まれないことを意味する。[l, u] は、それらがツリー内にある場合に、それらが含まれることを意味する。l は含まれるが u は含まれない [l, u)、及び l は含まれないが u は含まれる (l, u] の組み合わせも可能である。

【0029】

包括的スキャン (図1) では、返されたアイテムのキーによって形成される区間が、入力区間内に入る最大のサブセットである。包括的スキャンは、「36～80の間のキーを持つすべてのアイテムを取得する」のようなデータベースクエリを供するときに使用され得る。その結果は、キー36を持つアイテムが存在しない場合であっても、キー35を持つアイテムは含まない。カバリングスキャン (図2及び図3) では、返されたアイテムのキーによって形成される区間は、入力区間を含有する最小の上位セットである。図2の場合、キー22を持つアイテムは返されるが、図3では返されない。カバリングスキャンは、「ファイルの先頭からオフセット範囲550～800でデータを格納するディスクブロックを取得する」のようなストレージメタデータをクエリするときに有用である。その結果は、550の前の最大のキーと、800の後の最小のキー (すなわち、指定された範囲のちょうど両側のキー) を含み、つまり、要求された範囲のデータを格納するすべてのディスクブロックが返される。

【0030】

挿入：これは、挿入する新しいキーバリューペアを取る。指定されたキーを持つアイテムがツリーに存在しない場合は、そのキーバリューペアがツリーに挿入される。そうでない場合は、この操作は、ツリーを更新せず、ステータス「既に存在します (already-exists)」を返す。

【0031】

修正：これは、更新する既存のキーバリューペアを取る。指定されたキーを持つアイテムがツリーに存在する場合は、値は入力引数の値に更新される。値は以前と異なるサイズのものであり得ることに留意されたい。そうでない場合は、この操作は、ツリーを更新せず、ステータス「見つかりません」を返す。

【0032】

削除：これは、除去するアイテムのキーを取る。そのアイテムがツリーに存在する場合は、それは除去される。そうでない場合は、この操作は、ステータス「見つかりません」を返す。

【0033】

システムアーキテクチャ

図4は、実施形態に従った例示的なシステムを示す。このシステムは、ライタ401、リーダ402、システムメモリ403、及び任意選択的にリーダ402のローカルメモリ404を備える。ツリー構造103は、システムメモリ403に格納される。ライタ401及びリーダ402は、システムメモリ403に動作可能に結合される。例えば、リーダ402は、PCIeバスなどのバス405を介してライタ401に動作可能に結合されてもよく、一方、ライタ401は、バスを介した通信を必要としない別個のローカル接続又は専用接続406を介してシステムメモリ403に動作可能に結合されてもよい。ライタ401は、ローカル接続406を介してツリー103に書き込むためにシステムメモリ40

10

20

30

40

50

3にアクセスするように配置され得るが、リーダ402がツリーから読み出しを行うためには、それは、バス405を介してメモリ403にアクセスしなければならない場合がある。リーダ402は、それ自体のローカル接続407を介してそのローカルメモリ404に接続されてもよく、それによって、リーダのローカルメモリ404内のツリー103の一部をキャッシュするように配置されてもよい。

【0034】

メモリ403は、任意の適切な形態を取ることができ、1つ又は複数のメモリユニットで具現化された1つ又は複数のメモリ媒体を備えてもよい。例えば、メモリ媒体は、SRAM（スタティックランダムアクセスメモリ）若しくはDRAM（ダイナミックRAM）、EEPROM（電氣的に消去可能でプログラマブルなROM）、フラッシュメモリなどの電子シリコンベースの媒体；又は磁気ディスク若しくはテープなどの磁気媒体；又は再書き込み可能な光学媒体若しくは合成生物学的ストレージなどのさらにエキゾチックな形態；又はこれらの何れか及び／又は他のタイプの任意の組み合わせを備え得る。メモリ403が具現化される1つ又は複数のメモリユニットは、ライタ401と同じチップ上のオンチップメモリユニット、又はライタ401と同じボード上の別個のユニット、又はライタ401と同じハウジング若しくはラック内に、或いは別個のハウジング若しくはラック内の別の場所などに実装され得る、SSD（ソリッドステートドライブ）若しくはハードドライブなどの外部ユニット、又はこれらの任意の組み合わせを備え得る。ある特定の実施形態では、図示されているように、メモリ403は、ライタ401と同じボード又はチップ上に実装され得るDRAMを備える。例えば、ある特定の例では、システムは、CPUが存在するボードに差し込まれるDRAMボードを使用する。しかし、これは限定するものではない。

【0035】

リーダのローカルメモリ404は、使用される場合、任意の1つ又は複数のユニット内に任意の適切な媒体を備える任意の適切な形態を取ることにもできる。ある特定の実施形態では、図示されるように、リーダのローカルメモリ404は、リーダ402に統合されてもよい、又はリーダ402と同じチップ若しくはボード上に実装されてもよいDRAMを備える。

【0036】

実施形態では、ライタ401は、システムの非一時的なコンピュータ可読ストレージに格納され、及びシステムの1つ又は複数のプロセッサ、例えばCPU上で実行されるように配置されたソフトウェアにおいて実装され、一方、リーダ402は、FPGA、PGA、或いは専用（固定機能）ハードウェア回路網、又は組み合わせの形態を取り得るハードウェアにおいて実装される。下記は、そのような実施形態の観点から説明される場合があるが、以下に説明される技法の何れについても、ライタ401が代わりにハードウェアにおいて実装され、及び／又はリーダ402がソフトウェアにおいて実装される可能性が排除されないことが認識されるであろう。

【0037】

ライタ401（及び／又はリーダがソフトウェアにおいて実装される代替実施形態におけるリーダ）のソフトウェアを格納するために使用されるストレージは、上記のものの何れか、又は代替的に電子シリコンベースのROMなどのリードオンリーメモリ（ROM）、若しくはCD ROMなどの光学媒体などの任意の適切な形態を取り得る。1つ又は複数のプロセッサは、例えば、汎用CPU（中央処理装置）；又はGPU（グラフィック処理装置）、DSP（デジタル信号プロセッサ）、暗号プロセッサ、若しくはAIアクセラレータプロセッサなどの特定用途向けプロセッサ若しくはアクセラレータプロセッサ；又はこれらの何れか及び／又は他の種類の任意の組み合わせを備え得る。

【0038】

ある特定の実施形態では、図示されるように、ライタ401は、CPU上で実行されるソフトウェアにおいて実装され、リーダ402は、FPGAにおいて実装される。以下の実施形態は、このような例の観点から説明される場合があるが、これは限定するものではない。

ないことが認識されるだろう。

【0039】

Bツリーは、システムメモリ403にデータを格納する。実施形態では、これは、システムメモリと同程度の大きさ（通常は数百GB）のツリーを格納することを可能にするが、リーダ402（例えばFPGA）がPCIe405を通じてシステムメモリにアクセスするという代償を払う。リーダ（例えばFPGA）402は、そのローカルメモリ（例えばDRAM）404にツリーの上位層のキャッシュを保持し、実行を高速化することができる。ある例示的な実装形態では、リーダ402は、そのローカルレジスタファイルにルートノードのLID及びツリーのレベル数を記録する。

【0040】

Bツリーは、さらに大きなツリーを格納することを可能にするために、二次ストレージにもデータを格納することを可能にするように設計されてもよい。新たに出現した超低レイテンシーNVMe（Non-Volatile Memory Express）デバイスは、低レイテンシーシナリオの場合でも、これを魅力的な選択肢にする。

【0041】

実施形態では、FPGA402は、ルックアップ及びスキャン動作のみを実行し、更新動作はCPU401によって実行される。CPUとFPGAの同期は、好ましくは軽量化される。Bツリーノードが分割又はマージされる場合のみ、それは、PCIeバス405を通じた通信を必要とし得る（さらなる詳細については、データ構造のセクションを参照）。FPGAに対する読み出し操作のみをサポートすることによって、すべての操作をハードウェアにおいて実装することなく、読み出し主体のユースケースがハードウェアオフロードの恩恵を受けることが可能となる。

【0042】

データ構造

実施形態では、Bツリー103は、B+ツリーであり、これは、キーバリューペア101が、リーフ102Lにのみ格納され、内部ノード102Iは、キー指標及び子ノードへのポインタを格納することを意味する。

【0043】

図5は、二レベルツリーの一例を示すが、ここで説明する原理は、任意の数のレベル（すなわち、任意の数の親及び子のレベル）を持つツリーに適用することが可能である。

【0044】

ツリー103を備えるデータ構造は、ページテーブル501をさらに備える。ページテーブル501は、ノードID（論理ID、LIDとも呼ばれる）を、それぞれのノード102がシステムメモリ403に格納されている実際のメモリアドレスにマッピングする。このようにして、ライタ401又はリーダ402は、ページテーブル501におけるノードIDに基づいてアドレスをルックアップすることにより、ノードID（LID）に基づいて、メモリ403内のノード102の位置を特定することができる。

【0045】

前述のように、各内部ノード102Iは、その子ノードIDの各々を、それぞれの子によって包含されるキーの範囲のインジケーションにマッピングする情報を備える。図5は、これがどのように実装され得るかの一例を図示する。

【0046】

この例では、各内部ノード102Iは、ノードポインタ502及びキー指標503のセットを備える。各ノードポインタ502は、ノードID（論理ID、LID）で指定された子ノードへのポインタである。この例における各キー指標502は、子によって包含される範囲の境界でキーの1つを指定する。各内部ノード102Iにおいて、ポインタの左側にキーl、その右側にキーuの指標503を持つポインタ502は、区間[l, u)にキーを格納するサブツリーを指す。加えて、各内部ノード102Iは、その内部ノード内の最小キーよりも小さいキーを持つサブツリーへのポインタ502を格納する。この左端のポインタは、ノードのヘッダ603に格納される（例えば、図6を参照）。言い換え

10

20

30

40

50

ると、各キー指標は、一対のポインタ間に挟まれており、キー指標は、キー指標の両側のポインタが指す子によって包含される範囲間の境界を指定する。別の言い方をすれば、内部ノードは、左端の子リッドを格納する。これらは、基本的にX個のキーとX個の+1ポインタとを格納する。左端は、+1ポインタである。図5に示す特定の例では、区間〔22, 65〕内のキーを持つ新しいアイテムが中央のリーフに挿入され、22未満のキーを持つアイテムが左端のリーフに挿入され、65以上のキーが右端のリーフに挿入される。

【0047】

図5に示す、子をキーの範囲にマッピングするためのスキームは一例に過ぎないことが認識されるだろう。他の例は、例えば、まず、すべての境界キーをノードに格納し、次に、すべてのLIDがそれらの後に続くことである。

【0048】

幾つかの実施形態では、各リーフ102Lは、範囲スキャンの実装形態を単純にするために、左側にあるリーフへのポインタと、右側にあるリーフへのポインタも格納し得る。

【0049】

Bツリー103は、(他のBツリーと同様に)底部から成長する。挿入中に新しいアイテム101がマッピング先のリーフ102に収まることができない場合、リーフは2つに分割される。古いリーフのアイテムは半分に分割され、新しいリーフへのポインタが親ノードに挿入される。親ノードが一杯になると、それも分割される。分割はツリーのルートに至るまで再帰的に続けることができる。ルートが分割されると、新しいルートが作成され、これは、ツリーの高さを増加させる。リーフからすべてのアイテムが削除されたためにリーフが空になると、それは、親から削除される。代替的に、別の選択肢は、Bツリーの空間不変性を維持するために、リーフが半分より少なくなった際にそれらをマージすることである。

【0050】

ノードレイアウト

実施形態では、Bツリーノード102は、それらをDRAMページサイズに整合させるために、8KBのサイズであってもよい。これらは、リーダ402(例えば、FPGA)が物理メモリアドレスを使用してノード102にアクセスすることを可能にするために、オペレーティングシステムによって移動させることができないピン止めメモリに割り当てられてもよい。しかし、これは必須ではない。

【0051】

内部ノード102Iは、子ノードのアドレスを直接格納しないことが好ましい。その代わりに、それらは、各ノードの論理識別子(LID)を格納する。一例では、LIDのサイズは6バイトであってもよく、これにより、ツリーの最大サイズが 2^{61} バイトに制限される。ノードのLIDからノードの仮想アドレス及び/又は物理アドレスへのマッピングは、ページテーブル501に維持される。ページテーブル501は、システムメモリ403内及びFPGAアタッチドメモリ404内の両方に格納され得る。新しいノードのマッピングが作成されたとき、又はノードのマッピングが変更されたとき、ライタ401(例えば、CPU)は、システムメモリ403内のテーブル501を更新する。リーダ側にキャッシュが保持されている場合、ライタ401は、リーダ402(例えば、FPGA)に対して(例えば、PCIeを通じて)コマンドも発し、リーダのアタッチドメモリ404内のテーブルのコピーを更新する。代替的に、リーダは、更新のためにツリーをポーリングすることも可能である。

【0052】

LIDを使用したノードのアドレス指定は、NVMeデバイスへのノードの格納を可能にするなど、間接参照(indirection)のレベルを提供する。それは、場合によっては更新操作がノードのコピーオンライトを行い得るため、ツリーへのアクセスの同期にも役立つ。

【0053】

実施形態では、内部ノード102I及びリーフノード102Lのレイアウトは同じであ

10

20

30

40

50

る。例えば、図 6 を参照されたい。ノードヘッダ 6 0 3 は、例えば、ノード 1 0 2 の最初の 3 2 バイトに格納され得る。ヘッダ 6 0 3 は、ノード 1 0 2 のタイプ（内部又はリーフ）を指定するフィールド、ノード内の使用バイト数、ノードへのアクセスを同期するためのロックビット及びノードバージョン番号、及び／又は後述する幾つかの他のフィールドを含有し得る。図 5 に示すような例では、内部ノード 1 0 2 I のヘッダ 6 0 3 は、左端の子の L I D も格納する。リーフノード 1 0 2 L のヘッダは、左及び右の兄弟の L I D を格納し得る。キー及び値は、可変サイズのものであってもよい。それらは、ブラブ（blob）として格納され得る。各ブラブは、例えば、そのサイズを指定する 2 バイトのヘッダを有する。L I D は、そのサイズが既知であるため、ヘッダなしで格納され得る。値は、順次アクセス性能を向上させるために、B ツリーノード 1 0 2 L にインラインで格納されてもよい。5 1 2 バイトより大きい値は、B ツリー 1 0 3 の外部に格納されてもよく、値へのポインタは、リーフ 1 0 2 L に格納される。

10

【0054】

上記に例示したノードサイズなどの特定の実装パラメータの選択にかかわらず、本明細書に開示する実施形態によれば、各リーフノード 1 0 2 L は、大ブロック 6 0 1 及び小ブロック 6 0 2 の 2 つのブロックを備える。これを実装するために、ブロック間の境界へのポインタが、ノードのヘッダ 6 0 3 に格納され得る。このポインタは、例えば大ブロック 6 0 1 の開始位置に対するオフセットとして表現することができる。大ブロック 6 0 1 は、ソートされた順序でアイテムを格納し、小ブロック 6 0 2 は、最近の更新のログを格納する。小ブロック 6 0 2 が閾値（例えば、5 1 2 バイトに設定される）より大きくなると、それは、大ブロック 6 0 1 にマージされる。ノード 1 0 2 L を大ブロック及び小ブロックに分割することにより、読み出し操作は、更新のたびにノードをソートするオーバーヘッドなしに、ほとんどソートされたデータにアクセスすることの恩恵を受ける。小ブロック 6 0 2 のエントリは、新しく挿入されたアイテム又は削除マーカの何れかであり得る。任意選択的な最適化として、小ブロック 6 0 2 の各エントリは、大ブロック 6 0 1 のアイテムへのポインタ（例えば、2 バイト長）をさらに格納し得る。便利なラベルとして、これは「バックポインタ」と呼ばれることがある（ただし、「バック」という用語は、必ずしも物理アドレス空間などにおける特定の方向に限定することを意味しない）。ある例示的な実装形態では、新しく挿入されたアイテム 1 0 1 の場合、ポインタは、小ブロック 6 0 2 に追加された新しいアイテムのキーよりも大きいキーを持つ大ブロック 6 0 1 内の最初のアイテムを指す。或いは、削除マーカの場合、ポインタは、削除されたアイテムを指す。ポインタは、ノード 1 0 2 内のオフセットとして表現することができる。それは、大ブロック及び小ブロック内のアイテム 1 0 1 間の順序を、それらのキーを比較することなく確立するために、リーダ 4 0 2 によって使用することができ、これは、特に読み出しがハードウェアにおいて実装される場合に有用な最適化である。

20

30

【0055】

別の任意選択的な最適化として、小ブロック 6 0 2 内の各アイテムは、アイテムの挿入時にソートされた小ブロック内にそのインデックスを格納するフィールド（例えば、1 バイト長）を備えていてもよい。すなわち、これは、挿入時におけるアイテムの順序である。これは、「順序ヒント」と呼ばれることがある。これらのインデックスは、現在のソートされた順序を再構築するために、検索中に小ブロックをスキャンしながら、リーダ 4 0 2 によって「再生」される。この順序は、リーダ 4 0 2 の一時的な「間接参照」アレイ（例えば、F P G A レジスタ）に格納され、これは、ソートされた順序でアイテムにアクセスするために使用される。これにより、特にハードウェアにおいて実装された場合に、ソートがより効率的になり、大きな待ち時間が導入されない。

40

【0056】

図 7 は、小ブロック 6 0 2 に新しいアイテム 1 0 1 を挿入するためのステップの一例を示す。ステップ a) は、挿入前のノード 1 0 2 を示す。ステップ b) では、アイテム 1 0 1 は、小ブロック 6 0 2 にコピーされる。それは、リーダ 4 0 2 にはまだ見えない。ステップ c) では、ノードヘッダ 6 0 3 において示されるデータのサイズが、新しいアイテム

50

を含むように変更される。

【0057】

別の任意選択的な最適化として、ノード内のキーの検索を最適化するために、大ブロック601が、ほぼ等しいサイズのセグメントに分けられてもよい。セグメントの境界にあるキーは、境界へのポインタと共に、ノードヘッダ603の直後に格納され得るショートカットブロック604に格納される。キーの検索は、ショートカットブロックをスキャンし、キーを含有する可能性のあるセグメントを識別することから始まる。検索は、これらのセグメント及び小ブロックのみを調べ、このことは、メモリから読み出されるデータ量を減らす。例えば、性能ボトルネックがリーダ（例えば、FPGA）402とシステムメモリ403との間のPCIe帯域幅である場合は、この最適化は、性能を大幅に向上させる。32バイト以下のキーの場合、検索は、8KBのノードからおよそ1.5KBのデータを読み出し、それによって、ノード全体をスキャンする、より単純な実装形態に比べて5倍の性能向上が得られる。ショートカットキーは、大ブロック601及び小ブロック602のマージ中に選択され得る。

10

【0058】

ノードキャッシュ

FPGA402は、数ギガバイト（例えば、4GB）のDRAM404がそれに直接取り付けられている場合がある。このメモリは、すべてのユースケースにおいてBツリー103全体を格納するのに十分な大きさではないかもしれないが、それは、最初の幾つかのレベルについては十分な大きさであり得るため、実施形態では、それは、Bツリーのレイテンシーを低減し、及びBツリーのスループットを向上させるためのキャッシュとして使用することができる。例えば、キャッシュ404は、メタデータの量を削減するために、例えば8KBノード粒度で割り当てられ得るが、PCIe405を通じて読み出される不要なデータの量を削減するために、データは、256バイトチャンクでフェッチされ得る。FPGA402は、キャッシュ404内にあるチャンクの記録を取るために、各キャッシュノード102に対して32ビットの占有ビットマップを保持し得る。キャッシュ404及びシステムメモリ403間のページの一貫性は、ページマッピングが更新されるたびにキャッシュされたページを無効にすることによって維持され得る。CPU401上のソフトウェアは、ページマッピングが更新されたときにPCIe405を通じてページテーブル更新コマンドを送信することができ、FPGA402は、占有ビットマップをクリアすることによってページを無効にすることができる。

20

30

【0059】

ページテーブル

リーダ402（例えば、FPGA）は、例えばFPGAデバイス402に取り付けられ得る、そのローカルメモリ404（例えば、DRAM）に格納されたページテーブル501のローカルコピーを維持することができる。実施形態では、テーブルの各エントリは、FPGAメモリでは8バイトであるが、システムDRAMでは16バイトである。これは、実施形態では、システムDRAM内のコピーが2つのアドレス（仮想及び物理）を格納し、FPGAのDRAMメモリ内のコピーが1つのアドレス（物理のみ）を格納するためである。

40

【0060】

ページテーブル501のエントリは、それぞれのLIDにマッピングされたノード102のメモリアドレスを格納する。一般に、これらは、実装形態に応じて物理アドレス又は仮想アドレスの場合がある。実施形態では、ライタは、仮想メモリマッピングを採用し、ライタ側のページテーブル501は、それぞれのIDにマッピングされたノードの仮想アドレスを格納する。ページテーブルのコピーがFPGA402に保持される実施形態では、FPGAページテーブルコピーのエントリは、それぞれのLIDにマッピングされたノードの物理アドレスを格納し得る。例えば、8バイトのエントリ及び4GBのDRAMの場合、システムは、最大4TBのツリーをサポートし、これは、今日のサーバのメインメモリツリーにとって十分な大きさである。より大きなツリーは、リーダ402（例えば、

50

FPGA) をシステムメモリ 403 若しくは NVMe デバイスのページテーブル 501 にアクセスさせること、又は B ツリーのノードサイズを大きくすることによってサポートされ得る。

【0061】

ページテーブル 501 は、すべての可能な実施形態において絶対に必須であるというわけではない。あまり好ましくない代替形態は、内部ノード 102 I のノードポインタとして、メモリアドレスを直接使用することである。

【0062】

ルックアップ

ルックアップはルートから開始し、各レベルの内部ツリーノード 102 I を走査して、次のレベルのノード 102 へのポインタを見つける。各ルックアップは、リーフ 102 L で終了する。リーフがキーを含有する場合、それは、関連付けられた値を返す。そうでない場合は、それは、キーがツリーに格納されていないことを示すステータスを返す。

【0063】

内部ノード 102 I にアクセスする際に、ルックアップは最初に、ノードヘッダ 603 及びショートカットブロック 604 を含有するノード 102 の最初の部分（例えば、最初の 512 バイト）をフェッチし得る。ルックアップは、ショートカットブロック 604 を検索して、ターゲットキー以下の最後のキーを見つけることから始まる。それは、ショートカットアイテムのポインタをたどって、大ブロック 601 のセグメントを見つける。検索は、大きなセグメントをメモリからフェッチし、それを検索して、ターゲットキー以下の最後のキーを見つける。次に（又は並行して）、ルックアップは、小ブロック 602 をメモリからフェッチし、ターゲットキー以下の最後のキーを目的としてそれを検索する。バックポインタが使用されている場合、ルックアップは、キーを比較することなく、小ブロック及び大ブロックで見つけられたアイテム 101 のうち大きい方に格納されているポインタをたどる。小ブロックのアイテムのバックポインタが大ブロックのアイテムの直後を指している場合は、それは、小ブロックのポインタをたどるが、そうでない場合は、それは、大ブロックのポインタをたどる。ターゲットキーが大ブロック内及び小ブロック内の最初のキーの両方よりも小さい場合は、ルックアップは、左端のポインタをたどる。

【0064】

ルックアップがリーフ 102 I に達すると、ルックアップは、内部ノードと同様のやり方でそれを検索する。主な違いは、それが小ブロック及び大ブロックにおいて完全にマッチするものを探すという点であり、つまり、大ブロック及び小ブロックのアイテムを順序付ける必要がない。

【0065】

小ブロック検索：小ブロック 602 の検索は、アイテム 101 を検査する前に、それらをソートすることから始まってよい。ソートすることにより、小ブロックの検索は、それがターゲットキーよりも大きいキーを持つ最初のアイテムに遭遇するとすぐに停止することができる。実施形態では、小ブロックのソートは、キーの比較を行わない。その代わりに、それは、各小ブロックアイテムに格納されている順序ヒントフィールドを使用して順序付けを確立する。順序ヒントフィールドは、アイテムの挿入時に、ソートされた小ブロック内のアイテムのインデックスを格納する。これらのインデックスは、現在のソートされた順序を再構築するために小ブロックをスキャンしながら「再生」される。確立された順序は、アイテムをコピーすることなく、一時的な間接参照アレイ（例えば、FPGA レジスタ）に格納される。間接参照アレイは、ソートされた順序でアイテムにアクセスするために使用される。ソートは、特にハードウェアにおいて大きな待ち時間を導入せず、それは、ショートカットブロック 604 及び大ブロック 601 を検索しながら並行して行われ得る。ソフトウェアでは、ソートされた順序は、小さな間接参照アレイに保持され得る。

【0066】

図 8 は、例示的な、ノード 102 への挿入シーケンスを図示している。各アイテム 10

10

20

30

40

50

1に示された数字は、そのキーを表す。大ブロック601内の要素は、明確にするために省略されている。アイテムの上の数字は、その順序ヒントを表す。ステップa)では、ノード内の小ブロック602は、最初は空である。ステップb)では、例として、キー90を持つアイテムが挿入され、それが小ブロック内の最初のアイテムであるため、それは、順序ヒント0が付与される。次にステップc)において、キー60を持つアイテムが挿入される。それが小ブロック内の最小のアイテムであるため、それは、順序ヒント0が付与される。既存のアイテムの順序ヒントは変更されない。順序ヒントは、新しいアイテムを挿入する前に確立される小ブロックのソート順序から決定される。次に、ステップd)において、キー30を持つアイテムが挿入される。ここでも、それが小ブロック内の最小のものであるため、その順序ヒントも0である。ステップe)において、最後のアイ

10

【0067】

図9は、小ブロック602のソートを図示している。この図は、間接参照アレイ901の状態を示している。間接参照アレイは、小ブロック内のアイテムのオフセットを格納するが、図では、説明目的で、各オフセットがそのキーで表される。

【0068】

小ブロック602をソートするために、アイテム101は、それらが格納された順序で処理され、それらの順序ヒントは、間接参照アレイ内でアイテムをソートするために使用される。間接参照アレイ901は、小ブロック内のアイテムのオフセットを格納する。順序ヒントiを持つアイテムが処理されると、小ブロック内のそのオフセットが、間接参照アレイの位置iに挿入される。位置j ≥ iにあるアイテムはすべて右に移動される。小ブロック全体が処理されると、間接参照アレイは、ソートされた順序でアイテムのオフセットを含有する。図9は、図8に示した小ブロックの間接参照アレイソートを図示している。ある実装形態では、図中の各ステップは、キー比較と並行して、FPGA上の単一のサイクルで完了する。最初のステップa)では、アイテム90が最初のスロットに挿入される。次のステップb)では、すべてのアイテムが右に移動され、順序ヒント0を持つアイテム60が、最初のスロットに挿入される。次のステップd)では、順序ヒント0を持つアイテム30用の場所を空けるために、すべてのアイテムが再び右に移動される。最後のステップd)では、アイテム60及び90が右に移動され、順序ヒント1を持つアイテム45がインデックス1を持つスロットに挿入される。最終状態e)は、ステップa)～d)の後の間接参照アレイの状態を示す。

20

30

【0069】

図10は、間接参照アレイソータの例示的な実装形態を示す。それは、4アイテム幅のアレイを示す。ソータは、9ビット幅のレジスタを備える。レジスタへの入力は、その出力、その左側にあるレジスタの出力、又は入力オフセット値の何れかである。レジスタ入力は、入力インデックスをレジスタの(定数)インデックスと比較することによって選択される。それらが等しければ、入力オフセットが選択され、入力インデックスがレジスタのインデックスより小さければ、左側にあるレジスタの出力が選択され、入力インデックスの方が大きければ、レジスタの出力がそれ自体に書き込まれる。。この例示的な実装形態では、小ブロック内に削除アイテムを有するケースをカバーしていないが、レジスタへの入力が、その右側にあるレジスタの出力も取るように適応できるようにするためである。

40

【0070】

もちろん、図10に示すレイアウトは、可能なハードウェア実装形態の1つに過ぎず、当業者であれば、本明細書の開示が与えられれば、従来の設計ツールを使用して他の設計を行い得ることが認識されるだろう。また、読み出し側401でのソフトウェア実装形態も可能である。

【0071】

CPU401上でのソートの実装形態は、同様の手法に従ってもよい。間接参照アレイは小さく、それはL1キャッシュに快適に収まるため、動作は高速である。主な違いは、

50

アレイ内のアイテムを右に移動するために複数サイクルかかる点である。

【0072】

範囲スキャン

範囲スキャン操作は、上記のようにルートからツリー103を走査し、ターゲット範囲の開始キーを見つける。次に、それは、範囲の終了キー又はツリーの終わりに達するまで、開始キーから前方にスキャンする。

【0073】

実施形態では、リーフ102L間を移動するために、範囲スキャンは、兄弟ポインタを使用してもよい。そのような実施形態では、各リーフ102Lは、その左右の兄弟へのポインタを保持し、それによって、両方向に範囲スキャンを行うことが可能となる。しかし、兄弟ポインタの使用は必須ではない。兄弟ポインタが使用されない場合は、その代わりに、スキャンは、次のリーフを見つけるためにツリーを上にかのぼる。ほとんどの場合、それは、最初の親を見て、次の兄弟のLEDを取るだけでよい。しかし、それが親の終わりまで行ってしまった場合は、その兄弟を見つけるために、その親まで行くことになる。スキャンは、ルートまではるばる上がっていく必要があるかもしれない

【0074】

範囲スキャンは、キーによってソートされたアイテム101を返す。アイテムのソートされた状態を保つために、リーフ102Lのスキャンは、大ブロック601及び小ブロック602と一緒に処理してもよい。カバリングスキャンでは、スキャンはまず、大ブロック及び小ブロック両方における範囲の開始位置以下の最大のキーを持つアイテムを見つける。「標準的な」包括的スキャンは、それが範囲の開始位置以上の最小のキーから始まること以外は同様である。いずれにせよ、スキャンはその後、大ブロック及び小ブロックの両方でノードを前方にスキャンし、区間の終わりを探し、区間内にあるアイテムを返す。右の境界より大きいキーを持つアイテムが大ブロック及び小ブロックの両方で見つかり、スキャンは停止する。スキャン中、次の小さいアイテム及び次の大きいアイテムの両方が区間内にある場合、検索は、小さいアイテムのバックポインタを使用して、次にどちらを返すかを定めることができる。小さいアイテムが大きいアイテムを指す場合、小さいアイテムが返され、検索は、次の小さいアイテムに移動し、そうでない場合は、大きいアイテムが返され、検索は、次の大きいアイテムへと移動する。ショートカットキー604を扱うために、スキャンは、現在の大きいセグメントの終わりの記録を取ってもよい。この場合、次のセグメントの初めに、ショートカットブロックからキーが取り出され、大ブロックから値が取り出される。セグメントの中央では、キー及び値が、大ブロック内にある。

【0075】

挿入

挿入操作は、新しいアイテム101を挿入するノード120Lを見つけるためのツリーの走査から始まる。この走査は、ルックアップ中の走査と同様である。その違いは、実施形態では、それが、挿入操作及び修正操作のセマンティクスを実装するために、バージョンに関係なく、すべてのアイテムを読み出す点である。ノードを更新する前に、ライタ401は、それを書き込みロックでロックすることができ、それによって、ノードの状態が走査中に観察されたものであることが保証される（実施形態では、メモリ403は、異なるタイプのロック、すなわち書き込みロック及び読み出しロックをサポートすることができ、書き込みロックの使用は、必ずしも読み出しロックの使用を暗示しないことに留意されたい）。例えば、書き込みロックは、ノードのヘッダにおいて、32ビットのロックワードで格納することができ、ロックワードは、ロックビット及び31ビットのノードシーケンス番号から成り、これは、ノードが更新された際にインクリメントされる。ライタ401は、シーケンス番号を走査時と同じに保ちながら、ロックワードに対してアトミックコンペアアンドスワップ命令（atomic compare-and-swap instruction）を使用してロックビットをアトミックに設定することにより、書き込みロックを入手することができる。コンペアアンドスワップが成功すると、これは、ノードが走査以降変更されていない

10

20

30

40

50

いことを意味するので、挿入を続行することができる。コンペアアンドスワップが失敗した場合は、ノードが走査以降に変更されているため、ライタは、ツリーの走査を再び行うことによって操作を再試行する。

【0076】

高速パス挿入：あるよくあるケースでは、挿入は、ノード102Lを分割すること、又は大ブロック601及び小ブロック602をマージすることを必要としない。このよくあるケースの挿入は、所定の位置で行うことができる（図7）。ライタ401は、最初に、小ブロック602の終了後にそれをコピーし、次に新しいアイテム101を含むように小ブロックのサイズを調整することによって、新しいアイテムをアトミックに付加する。これを行うために、ライタ401は、書き込みロックの入手後に、以下のステップを行ってもよい：それは、ロックの入手後に、アイテムを小ブロックにコピーし、ノードのサイズを更新し、ノードのシーケンス番号をインクリメントし、及びノードをアンロックする。実施形態では、ヘッダ内のロックワード及び小ブロックのサイズは、同じ64ビットワードを共有し得るため、ライタ401は、単一の命令で、ノードバージョンをインクリメントし、ノードのサイズを更新し、及びロックを解除することができる。新しいアイテムは、ヘッダ603で現在指定されている小ブロックの終わりを越えて格納されるため、コピーされている間、並行リーダは、新しいアイテムを観察しない。それらは、アイテムなしで、又は完全に書き込まれたアイテムを用いてノードを観察する。

10

【0077】

実施形態では、リーダ402（例えば、FPGA）は、リーフノード102Lをキャッシュしない。もしキャッシュした場合、ライタ401は、PCIe405を通じてコマンドを発することによってキャッシュ404を無効にしなければならず、このことは、よくあるケースの挿入に追加のオーバーヘッドを導入することになる。

20

【0078】

大小マージ：小ブロック602が大きくなりすぎると、ライタ401は、大ブロック601及び小ブロック602をマージする（図11を参照）。それは、ノード102Lに対して新しいメモリバッファを割り当て、ノード内のすべてのアイテム101を新しいバッファ内の大ブロックにソートする。実施形態では、新しいアイテムは、（新しい小ブロックではなく）マージされた大ブロックに追加される。すなわち、新しいノードの小ブロックは、マージ後に空になる。原理上は、それは、他のやり方（すなわち、新しいアイテムは、新しい小ブロックに追加された最初のアイテムである）でも可能であるが、新しいアイテムを大ブロックに直ちに格納することにより、経時的に大小マージの数がわずかに少なくなる。

30

【0079】

ショートカットキー604を使用する実施形態では、ライタ401は、アイテムをソートしながらショートカットキーを選択する（それは、大ブロックが不変であるため、その時点でそれらを選択する）。処理された各アイテムについて、ライタ401は、例えば、これまでにコピーされたバイト数、コピーすべき残りのバイト数、及び／又はキー及び値の平均サイズに基づいて、ショートカット又は大ブロックのどちらにそれを入れるかを決める。それは、好ましくは、大ブロックセグメントを同様のサイズに保ちながら、ショートカットキーの数を最大にする。

40

【0080】

全てのアイテムが新しいバッファにコピーされると、ライタ401は、ページテーブル501内のノードのLIDのマッピングを新しいバッファのアドレスにアトミックに置き換える。LIDマッピングを更新するために、実施形態では、ライタ401は、ページテーブルのCPU及びFPGAコピーの両方でLIDエントリを更新する。それは、LIDエントリに対するソフトウェアロックを入手し、FPGAに対してページテーブル更新コマンドを発し、FPGAコマンドの完了後にソフトウェアロックを解除する。ノードのLIDは同じままであるため、親ノードを変更する必要はない。最後に、ライタは、古いメモリバッファをガベージコレクションリストに入れ、それをアンロックし、及び並行ライ

50

タが古いバッファを更新しないことを保証するために、ヘッダに「削除済み (deleted)」フラグを設定する。進行中の操作は、依然として古いバッファから読み出すことができるが、操作が削除されたノードを更新する必要がある場合、それは再試行する。再試行時には、それは、新しいノードマッピングを使用する。

【0081】

使用されるガベージコレクションは、並行システム用の標準的な明示的ガベージコレクション、例えばRCU、エポックベースのメモリマネージャ、又はハザードポインタなどの任意の適切なスキームであってよい。図11は、挿入操作中の大ブロック及び小ブロックのマージの一例を示している。ノード102Lに新しいアイテムを挿入すると、小ブロック602がその最大サイズを超えるため、ここで、小ブロックは、新しいアイテムと共に大ブロック601にマージされる。ステップa)は、右端のノード102Lに挿入する前のツリー103を示している。ステップb)は、新しいバッファの割り当て及びそれへのマージを示している。ノードへの古いポインタ(破線)が挿入される。ステップc)では、ページテーブルが、新しいバッファを指すように更新され、古いバッファは、進行中の操作によって必要とされなくなると、ガベージコレクション(GC)に入れられる。

10

【0082】

このアプローチの利点は、元のメモリアドレスのノードの古いインスタンスに読み出しロックをかける必要なしに、マージなどの複雑な更新を行うことができることである(ただし、一部の操作では、複数のライタを防止するだけで、1つのライタ及び1つ又は複数のリーダがノードにアクセスすることを許容する書き込みロックが依然として使用される)。ライタ401が新しいバッファでノード102Lを更新している間であるが、ページテーブルが更新される前に、リーダ402がそれを読み出そうとすると、リーダ402は、ページテーブル501で現在指定されている既存のメモリアドレスから古いインスタンスを単に読み出す。新しいバッファ(すなわち、新しいメモリ場所)への書き込みが完了すると、次に、ライタ401は、ノードのLIDに対するそれぞれのページテーブルエントリを新しいメモリアドレスに切り替えるために、ページテーブル501を迅速に更新することができる。実施形態では、これは、単一のアトミック操作で行うことができる。次に、後続の読み出しはすべて、更新されたページテーブルエントリに基づいて、新しいアドレスからノードの新しいインスタンスを読み出す。同様のアプローチは、ノードの分割又はノードのマージなどの他の複雑な更新にも使用することができる。

20

30

【0083】

LIDのために新しいメモリバッファが割り当てられると、そのバッファを削除して再利用する前に、そのバッファがいかなる操作によっても到達不能であることが確実にされるので、そのシーケンス番号は、安全に0に設定することができる。実例として幾つかのサイズ例を挙げると、小ブロックが512バイトより大きくなり、小ブロックエントリのサイズが少なくとも10バイト(削除エントリのサイズが10バイト、挿入エントリのサイズが少なくとも13バイト)であるときに、大ブロック601及び小ブロック602がマージされる場合、バッファの最大バージョンは41であり、つまり、31ビットのシーケンス番号は決してラップアラウンドしない。実際に、ある実装形態では、最小サイズは実際に平均で6バイトであるが、1バイトのエントリであっても、シーケンス番号がオーバーフローすることはない。

40

【0084】

ノード分割

挿入のための十分な空間がリーフ102Lにない場合、ライタ401は、リーフを2つに分割し、新しいアイテムへのポインタを親に挿入する(図12)。親が一杯の場合、ライタ401は、親も分割する。分割は、このようにしてルートに至るまで伝播することができる。更新されるが分割されない内部ノード102Iは、分割のルートと呼ばれる。ライタは、ページテーブル501内の分割のルートのマッピングを更新することにより、分割のルートの下の子ツリー全体を置き換える。データを更新する前に、ライタは、操作によって更新されるすべてのノード102に対する書き込みロックを入手することができ

50

る（これらは、他のライタによる妨げを防止するためのロックである）。ノードバージョンのミスマッチによりロックが失敗した場合は、ライタは入手したすべての書き込みロックを解除し、操作を再開する。ライタはまた、操作を完了するために必要なすべてのメモリバッファを割り当てる。それは、それが分割する各ノードに対して2つの新しいノード（新しいL I D及び新しいバッファを持つ）と、分割のルートに対してメモリバッファとを割り当てる。システムがメモリ不足のためメモリの割り当てに失敗した場合、ライタは、断念して、適切なステータスコードを呼び出し元に戻す。ロックの入手及びメモリの割り当ての後、ライタは、リーフから上のノードを処理し、それらを新しく割り当てられたバッファに分割する。分割の結果生じる2つのノードの各々は、元のノードのおよそ半分のデータで終わる。ライタは、分割中に大ブロック及び小ブロックをマージする。それが分割のルートに達すると、ライタは、それを新しいメモリバッファにコピーする。それは、新しい子ノードへのポインタと、ノード間の境界にあるキーとを含むように新しいバッファを修正する。2つのサブツリーをスワップするために、ライタは、ページテーブルの分割のルートのマッピングを新しいメモリバッファのアドレスで更新する。このようにサブツリーをスワップすることで、リーダが古いサブツリー又は新しいサブツリーのどちらかを観察することが保証される。次に、ライタは、分割されたすべてのノードのメモリバッファ、それらのL I D、及び分割のルートの古いメモリバッファをガベージコレクションリストに入れる。最後に、それは、すべてのメモリバッファをアンロックし、それらに削除済みとマークを付ける。

10

20

【0085】

図12は、挿入時にノードを分割する一例を示している。破線ボックスは、新しく割り当てられたノードバッファを表す。塗りつぶされたボックスは、新しいL I D及びメモリバッファを持つ新しいノードを表す。ステップa)は、分割前のツリー103を示す。右端のノードが分割される予定である。分割のルートは、ツリーのルートである。ステップb)では、分割のために2つの新しいノードが割り当てられ、分割のルートのために新しいバッファが割り当てられる。ステップc)では、分割のルートのマッピングが更新され、ノードがガベージコレクション(GC)リストに入れられる。

30

【0086】

実施形態では、ライタは、スキャン操作中に使用される兄弟リーフポインタも更新する。それは、兄弟リーフをロックし、それが新しいサブツリーをスワップした後にポインタを更新する。兄弟ポインタ及び分割のルートがアトミックに更新されないかもしれないとしても、リーダは、ツリーの一貫した状態を観察する。

40

【0087】

ツリーのルートが新しいアイテムを入れることができない場合、ライタは、それを分割して新しいルートを作成し、それによって、ツリーの高さが増加する。ライタは、分割のルート用のメモリバッファのみの代わりに、新しいルート用のL I D及びメモリバッファを割り当てることを除いては、上記のように分割を実行する。それは、左端のポインタが古いルートの左半分に設定された新しいルートと、古いルートの右半分を指す単一のアイテムとを初期化する。その後、それは、ルート及びツリーの高さを更新するためのコマンドをリーダ（例えば、F P G A）に送信してもよい。ある例示的な実装形態では、リーダ402は、ルートノードのL I D及びツリーレベルの数を、キャッシュ404にではなく、そのローカルレジスタファイルに記録する。キャッシュ内のデータコピー、例えば、より古いルートノードは、そのL I Dが別の物理ノード用に再利用されるときに最終的に無効にされるか、或いは、古いデータはしばらくするとアクセスされなくなるため、場合によっては退去させられる。そのため、キャッシュは、キャッシュされたノードデータの一貫性を維持することだけに集中すればよい。

50

【0088】

削除

アイテム101の削除は、新しいアイテムの挿入と類似している。ライタ401は、削除エントリを小ブロック602に挿入する。このエントリは、それが削除しているアイテ

ムを指すので、ルックアップ操作の間、削除されたアイテムは無視される。大ブロック 601 及び小ブロック 602 のマージ中に、削除されたアイテムによって占有された空間が取り戻される。ノード 102 が空になると、それは削除される。ノードの削除は、ノードの分割と同様に、リーフ 102 L からツリー 103 の上方に進むことができる。実施形態では、上記と同じ技法を使用して、そのアトミック性が保証される。

【0089】

修正

修正操作は、挿入及び削除の組み合わせとして行われ得る。ライタ 401 は、古いアイテムの削除エントリと新しいアイテムとを小ブロック 602 に付加し、ノードのヘッダ 603 を更新することによって、それらをアトミックに発行する。

【0090】

結論

上記の実施形態は、単なる例として説明されていることが認識されるだろう。より一般的には、本明細書に開示される一態様によれば、データ構造を格納するメモリであって、データ構造が、それぞれノード ID を有する複数のノードを備えたツリー構造を備え、一部のノードがリーフノードであり、他のノードが内部ノードであり、各内部ノードが、ツリー構造内の 1 つ又は複数の子のそれぞれのセットの親であり、各子が、リーフノード又は別の内部ノードの何れかであり、各リーフノードが、子であるが親ではなく、リーフノードの各々が、それぞれキーバリュペアを備えた 1 つ又は複数のアイテムのそれぞれのセットを備え、内部ノードの各々が、それぞれの子の各々のノード ID を、それぞれの子によって包含されるキーの範囲にマッピングする、メモリを備えたシステムが提供される。システムはさらに、リーフノードにアイテムを書き込むように配置されたライタと、リーフノードからアイテムを読み出すように配置されたリーダとを備える。リーフノードの各々は、それぞれの第 1 のブロックと、それぞれの第 2 のブロックとを備え、第 1 のブロックが、メモリのアドレス空間においてキーの順序でソートされたそれぞれのリーフノードの複数のアイテムを備える。ライタは、ツリー構造に新しいアイテムを書き込む際に、ツリー構造を通して、ノード ID に対するキーのマッピングをたどることによって、どのリーフノードに書き込むべきかを識別し、識別されたリーフノードの第 2 のブロックに対して、キーの順序でソートされるのではなく、書き込まれた順序で新しいアイテムを書き込むように構成される。リーダは、ツリー構造から 1 つ又は複数のターゲットアイテムを読み出す際に、ツリー構造を通して、ノード ID に対するキーのマッピングをたどることによって、どのリーフノードから読み出すべきかを決定し、次に、a) 第 1 のブロックで既にソートされたアイテムの順序と、b) リーダが第 1 のブロックのアイテムに関連したキーによって第 2 のブロックのアイテムをソートすることとに基づいて、1 つ又は複数のターゲットアイテムについて決定されたリーフノードを検索するように構成される。

【0091】

実施形態において、ライタは、システムのコンピュータ可読ストレージに格納され、システムの 1 つ又は複数のプロセッサ上で実行されるように配置されたソフトウェアにおいて実装されてもよい。実施形態では、リーダは、PGA、FPGA、又は専用ハードウェア回路網において実装されてもよい。しかし、代替的に、ライタがハードウェアにおいて実装されてもよく、及び／又はリーダがソフトウェアにおいて実装されてもよく、又は何れかがソフトウェア及びハードウェアの組み合わせで実装されてもよい。

【0092】

実施形態では、ツリー構造は、B ツリー（例えば、アイテムが中間ノードに格納されない B+ ツリー）の形態を取ることができる。

【0093】

実施形態では、リーフノード及び／又は内部ノードの少なくとも幾つかにおいて、それぞれの第 1 のブロックは、セグメントに分けられてもよく、そのノードは、複数のショートカットを備えたショートカットブロックをさらに備えてもよく、各ショートカットは、i) それぞれの第 1 のブロックの対応するセグメントの境界におけるキーのインジケシ

10

20

30

40

50

ョンと、 i) リーフノード内の対応するセグメントに対するオフセットとを備える。このような実施形態では、リーダは、第 1 のブロックの上記検索を行う際に、キーに基づいて 1 つ又は複数のアイテムを含有する可能性のあるセグメントのみを検索するためにショートカットを使用するように構成されてもよい。

【0094】

実施形態では、ライタは、リーフノードの少なくとも 1 つに書き込む際に、それぞれの第 2 のブロックにおける各アイテムに関連付けられたバックポインタを書き込み時に含めるように構成されてもよく、各バックポインタは、第 2 のブロック内の関連付けられたアイテムと比較して次に大きいキー又は次に小さいキーを持つ第 1 のブロック内のアイテムに対するオフセットを備える。このような実施形態では、リーダは、第 2 のブロックの上記ソートを行う際に、それぞれの第 1 のブロックに関連したキーによって第 2 のブロックのアイテムをソートするために、バックポインタを使用するように構成されてもよい。

10

【0095】

実施形態では、ライタは、リーフノードの少なくとも 1 つに書き込む際に、第 2 のブロック内のアイテムの各々について、第 2 のブロックにアイテムを書き込む時点における、第 2 のブロックの現在のキーに対するアイテムのキーの順序を示す順序ヒントを書き込み時に含めるように構成されてもよい。このような実施形態では、リーダは、第 2 のブロックの上記ソートを行う際に、それぞれの第 1 のブロックに関連したキーによって第 2 のブロックのアイテムをソートするために、順序ヒントを使用するように構成されてもよい。

【0096】

20

例えば、リーダは、それぞれの第 2 のブロック内のアイテムを、それらが格納された順序で処理することと、アレイ内のアイテムをソートするために順序ヒントを使用することと、アレイが、第 2 のブロック内のアイテムのオフセットを格納する、使用することと、によって順序ヒントを使用するように構成されてもよい。順序ヒント i を持つ各アイテムが処理される際に、第 2 のブロック内のそのオフセットが、アレイ内の位置 i に挿入され、第 2 のブロック全体が処理された後に、アレイがソートされた順序でアイテムのオフセットを含有するように、位置 $j \geq i$ にあるすべてのアイテムが右に移動される。

【0097】

実施形態では、データ構造は、ノード ID をそれぞれのメモリアドレスにマッピングするページテーブルを備えてもよい。この場合、リーダは、決定されたノード ID のノードから読み出す際に、メモリ内の決定されたノードのメモリアドレスを決定するためにページテーブルを使用するように構成される。このような幾つかの実施形態では、ライタは、ノードの書き込み、分割、又はマージを行うためにノードを更新する際に、ノードの更新されたバージョンを新しいメモリアドレスに書き込み、一旦書き込まれると、新しいメモリアドレスを指すようにページテーブル内のそれぞれのメモリアドレスを更新するように構成されてもよい。

30

【0098】

実施形態では、各第 2 のブロックは、最大サイズを有し得る。ライタは、識別されたノードのそれぞれの第 2 のブロックが最大サイズを超えることを生じさせる新しいアイテムを書き込む場合、第 2 のブロックをそれぞれの第 1 のブロックにマージするように構成されてもよい。

40

【0099】

データ構造がノード ID をそれぞれのメモリアドレスにマッピングするページテーブルを備える実施形態では、ライタは、それぞれの第 2 のブロック及び第 1 のブロックをマージするためにノードを更新する際に、マージを含むノードの更新されたバージョンを新しいメモリアドレスに書き込み、一旦マージが完了すると、新しいメモリアドレスを指すようにページテーブル内のそれぞれのメモリアドレスを更新するように構成されてもよい。

【0100】

実施形態では、各リーフノードは、それぞれのリーフノードのキーに隣接するキーの範囲を包含する 1 つ又は複数の兄弟リーフノードのノード ID をさらに示してもよい。また

50

、リーダは、キーの範囲からアイテムを読み出すための範囲スキャンを行うように構成されてもよい。このような実施形態では、リーダは、2つ以上のリーフノードに及ぶアイテムの範囲から読み出す範囲スキャンを行う際に、ツリー構造を通して、ノードIDに対するキーのマッピングをたどることによって、スキャンされた範囲のキーの1つを包含するリーフノードの1つを決定し、次に、リーフノードの上記1つにおいて示された兄弟リーフノードの少なくとも1つのIDを使用することによって、スキャンされた範囲の少なくとも1つの他のキーを包含する少なくとも1つの他のリーフノードを決定するように構成されてもよい。

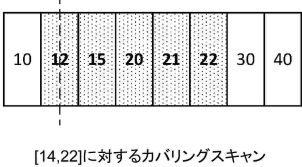
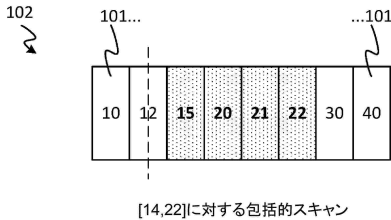
【0101】

本明細書に開示された別の態様によれば、本明細書に開示された任意の実施形態によるメモリ、ライタ、及び／又はリーダの動作を備えた方法が提供される。別の態様によれば、1つ又は複数の非一時的なコンピュータ可読媒体上に具現化されたコンピュータプログラムであって、1つ又は複数のプロセッサ上で実行される際に、ライタの書き込み及び／又はリーダの読み出しを行うように構成されたコードを備えた、コンピュータプログラムが提供され得る。開示された技法の他の変形例又は適用例は、本明細書の開示が与えられれば、当業者には明らかになり得る。本開示の範囲は、説明した実施形態によって限定されるものではなく、添付の特許請求の範囲によってのみ限定される。

【図面】

【図1】

【図2】



10

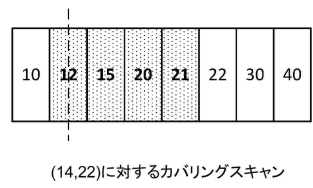
20

30

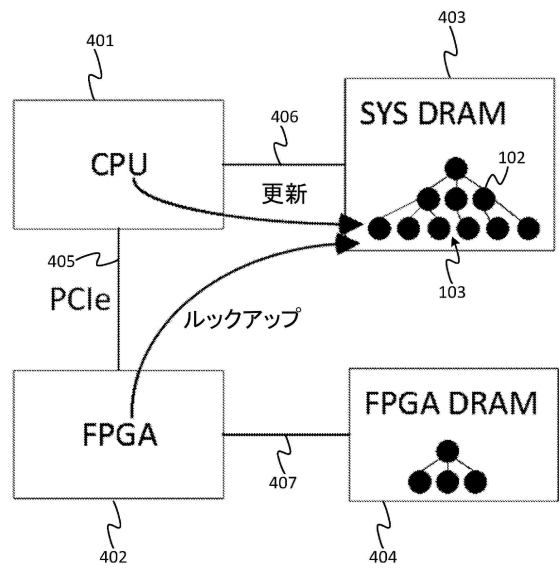
40

50

【図 3】



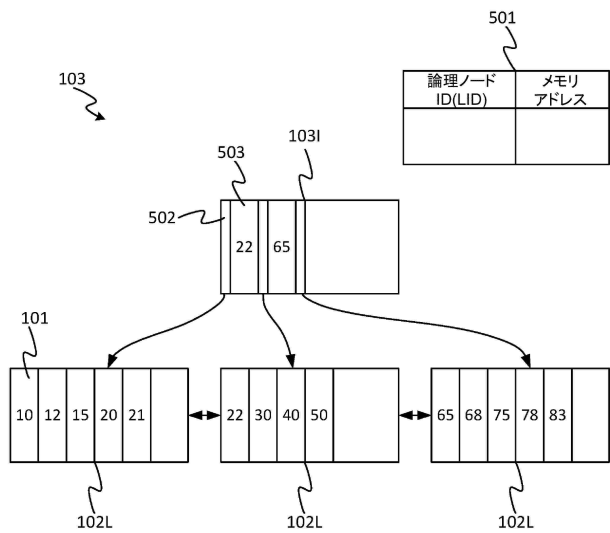
【図 4】



10

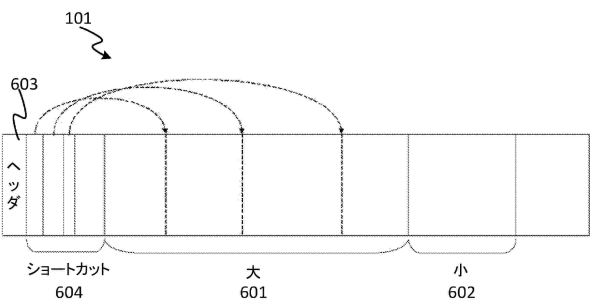
20

【図 5】



30

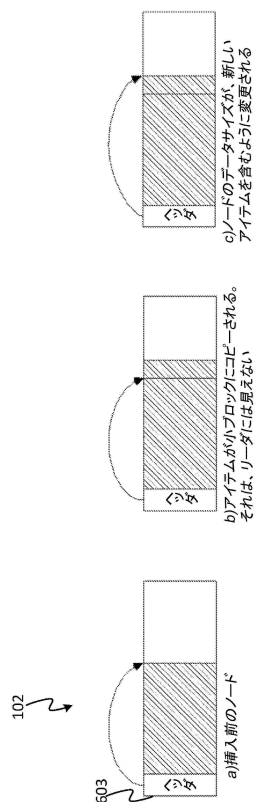
【図 6】



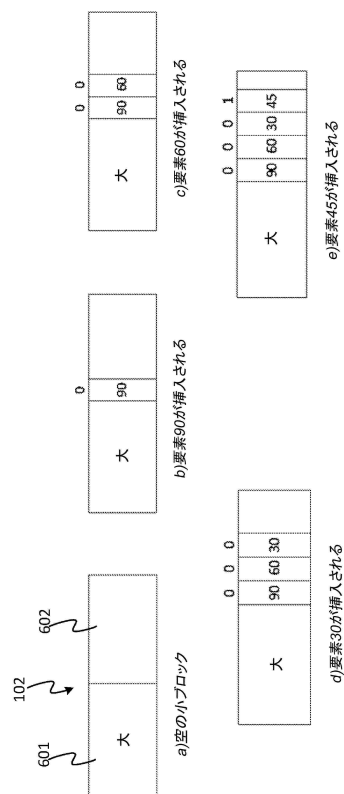
40

50

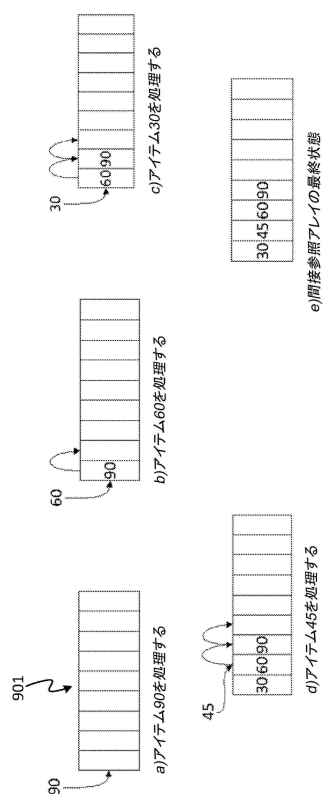
【圖 7】



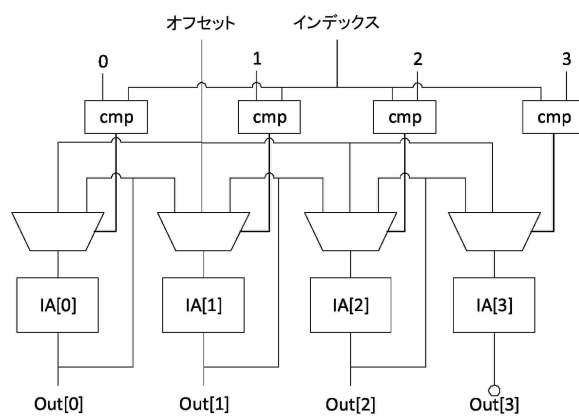
【圖 8】



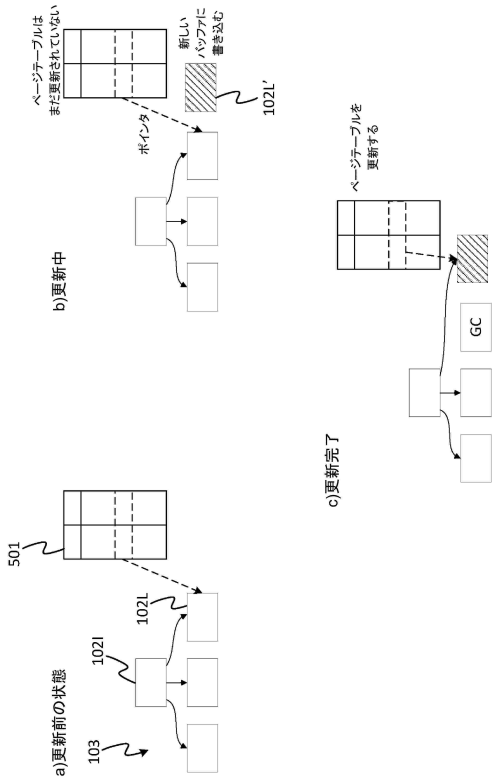
【圖 9】



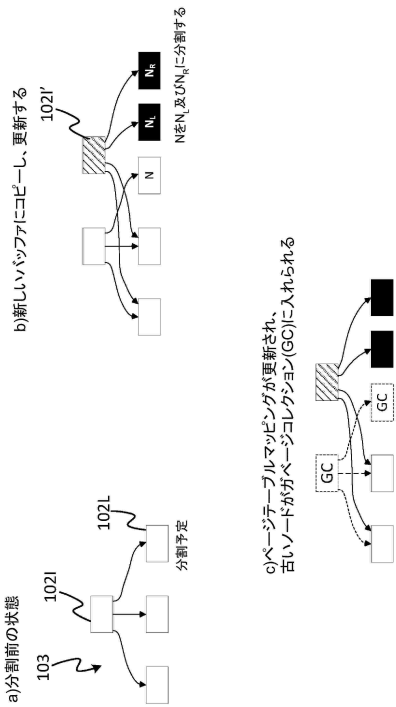
【図 10】



【図 1 1】



【図 1 2】



【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2022/029722

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F16/901
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2018/373727 A1 (DING LI [US] ET AL) 27 December 2018 (2018-12-27) abstract paragraph [0001] paragraph [0015] paragraph [0018] paragraph [0020] paragraph [0021] - paragraph [0036] figures 2, 3, 4, 5A-E -----	1-6, 10-15
X	US 2016/335299 A1 (VEMULAPATI MURALI [US] ET AL) 17 November 2016 (2016-11-17) abstract paragraph [0003] - paragraph [0006] paragraph [0015] - paragraph [0031] paragraph [0033] - paragraph [0045] figures 1, 2 ----- -/--	1-6, 10-15

☒ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

6 July 2022

Date of mailing of the international search report

14/07/2022

Name and mailing address of the ISA/
European Patent Office, P.B. 5618 Patentlaan 2
NL - 2280 HV Rijswijk
Tel. (+31-70) 340-2040,
Fax: (+31-70) 340-3016

Authorized officer

Pose Rodríguez, J

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2022/029722

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT		
Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2020/293506 A1 (GUPTA ABHISHEK [US] ET AL) 17 September 2020 (2020-09-17) abstract paragraph [0001] - paragraph [0003] paragraph [0016] - paragraph [0021] paragraph [0023] - paragraph [0032] figures 1,2 -----	1-15

4

10

20

30

40

50

INTERNATIONAL SEARCH REPORT				International application No PCT/US2022/029722
Information on patent family members				
Patent document cited in search report		Publication date	Patent family member(s)	Publication date
US 2018373727	A1	27-12-2018	NONE	
US 2016335299	A1	17-11-2016	NONE	
US 2020293506	A1	17-09-2020	NONE	

10

20

30

40

50

フロントページの続き

MK,MT,NL,NO,PL,PT,RO,RS,SE,SI,SK,SM,TR),OA(BF,BJ,CF,CG,CI,CM,GA,GN,GQ,GW,KM,ML,MR,N
E,SN,TD,TG),AE,AG,AL,AM,AO,AT,AU,AZ,BA,BB,BG,BH,BN,BR,BW,BY,BZ,CA,CH,CL,CN,CO,CR,CU,
CZ,DE,DJ,DK,DM,DO,DZ,EC,EE,EG,ES,FI,GB,GD,GE,GH,GM,GT,HN,HR,HU,ID,IL,IN,IQ,IR,IS,IT,JM,J
O,JP,KE,KG,KH,KN,KP,KR,KW,KZ,LA,LC,LK,LR,LS,LU,LY,MA,MD,ME,MG,MK,MN,MW,MX,MY,M
Z,NA,NG,NI,NO,NZ,OM,PA,PE,PG,PH,PL,PT,QA,RO,RS,RU,RW,SA,SC,SD,SE,SG,SK,SL,ST,SV,SY,TH,
TJ,TM,TN,TR,TT,TZ,UA,UG,US,UZ,VC,VN,WS,ZA,ZM,ZW

弁理士 阿部 豊隆

- (72)発明者 ドラゴイエビッチ, アレクサンダー
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
- (72)発明者 リウ, ジュンイー
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
- Fターム (参考) 5B175 CA02 KA11