200900930

（此處由本局於收文時黏貼條碼）

854203

發明專利說明書

（本申請書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※申請案號：97102657

※申請日期：97年01月24日　　※IPC分類：G06F 12/06 (2006.01)

一、發明名稱：

(中) 階層式不可改變的內容可定址的記憶體處理器

(英) Hierarchical immutable content-addressable memory processor

二、申請人：(共 1 人)

1.姓　名：(中) 大衛　奇里頓

(英) CHERITON, DAVID R.

代表人：(中)

(英)

地　址：(中) 美國加州帕羅奧多考伯街一三一號

(英) 131 Cowper St., Palo Alto, CA 94301, U.S.A.

國籍：(中英) 加拿大　　CANADA

三、發明人：(共 1 人)

1.姓　名：(中) 大衛　奇里頓

(英) CHERITON, DAVID R.

國　籍：(中) 加拿大

(英) CANADA

四、聲明事項:

◎本案申請前已向下列國家（地區）申請專利□主張國際優先權：

【格式請依：受理國家（地區）；申請日；申請案號數　順序註記】

1.美國　　；2007/01/26；60/897,773 ☑有主張優先權

（5）

200900930

（此處由本局於收文時黏貼條碼）

854203

發明專利說明書

（本申請書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫）

※申請案號：97102657

※申請日期：97年01月24日　　※IPC分類：G06F 12/06 (2006.01)

一、發明名稱：

(中) 階層式不可改變的內容可定址的記憶體處理器

(英) Hierarchical immutable content-addressable memory processor

二、申請人：(共 1 人)

1.姓　名：(中) 大衛　奇里頓

(英) CHERITON, DAVID R.

代表人：(中)

(英)

地　址：(中) 美國加州帕羅奧多考伯街一三一號

(英) 131 Cowper St., Palo Alto, CA 94301, U.S.A.

國籍：(中英) 加拿大　　CANADA

三、發明人：(共 1 人)

1.姓　名：(中) 大衛　奇里頓

(英) CHERITON, DAVID R.

國　籍：(中) 加拿大

(英) CANADA

四、聲明事項:

◎本案申請前已向下列國家（地區）申請專利□主張國際優先權：

【格式請依：受理國家（地區）；申請日；申請案號數　順序註記】

1.美國　　；2007/01/26；60/897,773 ☑有主張優先權

（5）

九、發明說明

【發明所屬之技術領域】

本發明係相關於電腦記憶體管理。

【先前技術】

在習知 Von Neumann（馮紐曼）電腦架構中，記憶體被構造成以循序位址索引之固定尺寸的胞元（cell）之線性陣列。圖 1 圖示此習知架構的例子。指令 102 的執行產生如所示之記憶體 104 的內容。雖然此方法實施簡易且相當容易用於固定尺寸的應用程式資料單元，但是軟體結構和技術需要處理可變尺寸和構造資料。

欲處理可變尺寸資料，軟體典型上實施動態記憶體配置器，此配置器定位至少與所要求的一般大之記憶體的連續區域。然而，在延伸執行過程中，記憶體空間可變得分段成多個較小區域，使得即使可利用的記憶體總量仍充裕，記憶體配置要求還是會失敗。諸如世代垃圾收集器等機構可藉由拷貝區域使它們彼此連續以週期性重新緊密（re-compact）記憶體，但是此種垃圾收集在召喚時可能與正在執行的應用程式相互干擾，此係即時系統中尤其無法接受的觀點或係通常需要可預測反應者。而且，若可變尺寸資料項目長度變長，則軟體必須配置該所需尺寸的新連續區域，及拷貝該資料到新位置，及將指向舊位置的所有參考改成指向新位置的目前點（now point）。爲了有助於後者之此一動作，某些軟體經由提供實際指標給可變尺寸資

料的確定尺寸定位來採用額外位準的間接指標，如此具有單一位置更新，但是以各個存取上的額外的間接指標爲代價。

可變尺寸資料的另一方式係使用指標（即、使用構造資料）從非連續記憶體單元來建構該可變尺寸資料類型。構造資料是處理的一大挑戰，因爲其在複雜的構造資料之指標參考特徵存在時，難以決定何時記憶體區是有空的。構造資料代表中的資料存取亦導致須間接經由指標以確定可變尺寸資料項目中的下一登錄點的負擔。

就在多個分開處理中執行的應用程式而言，構造資料採用更多的負擔，因爲其通常需要串列化構造資料且拷貝結果到分開的位址空間，然後將其去串列化以共享結構資料。用於提供處理之間之隔離的虛擬位址轉譯，使得用於構造該資料的位址對各位址空間須是唯一，係此之產生原因。與典型應用程式資料單元（如、32-128 位元組）比較，用於位址轉譯的記憶體頁（如、8 千位元組或更大）的大顆粒進一步阻礙共享。結果，應用程式被組織成一處理內的多個執行緒（thread），須放棄保護分離的位址或爲串列化、拷貝、和去串列化位址空間之間的結構資料而付出相當大的代價。

近來可預期的技術發展使此標準 Von Neumann 模型的不利點更加有問題。首先，記憶體系統性能無法跟上日益增加的處理器性能，使得記憶體性能成爲電腦性能逐漸侷限的因素。因此，諸如拷貝和垃圾收集等記憶體密集的操

作在比例上變得更加昂貴。快取已變成處理處理器/記憶體速度失配的主要機制。然而，隨著日益增加的記憶體尺寸、更大和更複雜的應用程式物件、及更資料密集的應用程式，使用此習知記憶體模型，快取已變得明顯較無效率。

當作第二方向，電腦硬體逐漸依賴平行執行以達成性能優點。尤其是，在單一微晶片上實施多個"核心"，在共享記憶體控制器和快取記憶體的同時提高成本效率是可行的。然而，由於構造資料所造成的額外拷貝，導致重複資料而造成快取記憶體的使用無效率。而且，諸如參考計數更新和使用更新之快取線之錯誤共享等額外更新導致記憶體和快取記憶體更加無效率。

當作最後的趨勢，增加的記憶體尺寸和處理器性能使應用程式能夠變得較大且較複雜，但是增加維持軟體正確性的困難，尤其是不斷改良和特徵的需求。同時，應用程式在時間、任務、甚至使用壽命上的急迫性功能日益要求，使其可靠性更爲重要。

就這些和其他理由，已考慮其他記憶體架構。例如，在 US4,989,137 中，使用者處理器藉由記憶體管理系統的一部分之連結暫存器單元，專門地存取記憶體。以此方式，記憶體管理系統可對使用者處理器隱藏實體記憶體組織的低階細節，取而代之，以邏輯記憶體抽象化呈現給使用者處理器。在 US4,695,949 中，說明區塊取向記憶體，其中爲各個區塊維持參考計數，藉以減輕經常性垃圾收集的

需要。

在 US5,784,699 中，考慮具有幾個標準區塊尺寸之區塊取向記憶體系統。聚集記憶體配置要求到最接近的標準區塊尺寸。在 US5,950,231 中，考慮以堆疊指標控制的區塊取向記憶體管理系統。在 US5,247,634 及 US5,864,867 中，考慮依據使用樹的記憶體管理。

然而，這些方式及標準 Von Neumann 快取方式二者都有困難度，特別是構造資料而言。因此，提出改良的記憶體管理，尤其是針對可變尺寸和構造資料在技術上應是一大發展。

【發明內容】

根據階層式不可改變的內容可定址之記憶體處理器（HICAMP）架構來提供改良的記憶體管理。在 HICAMP 中，實體記憶體被組織成兩或更多個實體記憶體區塊，各個實體記憶體區塊具有固定儲存容量。提供任一時間點中哪一些實體記憶體區塊是有效的指示。記憶體控制器提供非重複寫入能力，其中在寫入時將欲寫入至實體記憶體之資料與所有有效實體記憶體區塊的內容比較，藉以確保在完成非重複寫入之後沒有兩有效記憶體區塊具有相同資料。在重複的案例中，則使用現存的區塊而不產生具有相同值的另一區塊。

【實施方式】

為了更加瞭解本發明，首先以比較抽象、實施獨立的方式來考慮本發明的實施例之主要型態，然後經由圖解例子提供一些其他的實施細節是有幫助的。

1)主要型態

為了簡潔，參考本發明的各種實施例當作階層式不可改變的內容可定址之記憶體處理器（HICAMP）架構之例子是方便的。該 HICAMP 架構與標準的 Von Neumann 架構不同於幾個基本方面。

第一，在 HICAMP 中，實體記憶體被組織成兩或更多個實體記憶體區塊，各個實體記憶體區塊具有固定儲存容量。第二、提供任一時間點中哪一些實體記憶體區塊是有效的指示。第三、記憶體控制器提供非重複寫入能力，其中在寫入時將欲寫入至實體記憶體之資料與所有有效實體記憶體區塊的內容比較，藉以確保在完成非重複寫入之後沒有兩有效記憶體區塊具有相同資料。

從 HICAMP 架構的這些型態可具有幾個主要的優點。

1)消除有關記憶體分段儲存的習知問題，因為記憶體的配置係經由固定尺寸的實體記憶體區塊。重複抑制使這些區塊能夠被有效定位和管理。

2)因為重複抑制，指定量的應用程式資料所需之記憶體的量能夠依照應用程式資料的尺寸被分析和連結。

3)因為資料區塊在有效之時無法被修改，所以無論在位址空間內和在位址空間之間都能夠安全地共享資料使得

可以減少記憶體拷貝。

4)可有效實施記憶體中的兩物件之相等/不相等比較，因爲它們只有代表相同實體區塊才是相等的。

5)重複的抑制使可利用的實體記憶體更有效的使用。

6)使同時程式化簡化且更有效率，因爲在許多共有的資料結構上可微量地執行非區塊更新。

在較佳的 HICAMP 實施例中，將非重複寫入的概念延伸到亦包括需要一個以上的實體記憶體區塊來儲存之資料項目（簡稱多區塊資料）。此可藉由提供一多區塊資料規約加以完成，該多區塊資料規約替任何需要二或更多實體區塊以儲存的資料項目指定一唯一代表。然後，當將多區塊資料寫入至記憶體時可執行該多區塊資料規約，使得非重複寫入可確保沒有多區塊資料的任何實例之重複存在於該組有效實體記憶體區塊中。

當作多區塊非重複寫入的效果之一簡單例子，試想在一系統中之一字串"abcde"的代表，該系統中之實體區塊具有三字元的儲存容量。由總共三區塊來代表該字串，其中區塊 1 包含字串的頭，區塊 2 包含字串的尾，及區塊 3 包含到區塊 1 及 2 的指標。沒有多區塊唯一性提供規約，則該例示字串可被代表爲（"abc", "de"）或代表爲（"ab", "cde"），其中第一項目是區塊 1 的內容而第二項目是區塊 2 的內容。藉由執行（或支援）此種多區塊唯一性規約（如、首先填充頭，或首先填充尾），可去掉此種多種代表的可能性。結果，具有到區塊 1 及 2 的指標之區塊 3 是

記憶體中字串"abcde"的唯一代表，藉以提供多區塊重複的抑制。

較佳的是，各個實體記憶體區塊具有相同儲存容量，以簡化記憶體管理。再者，相對於諸如藉由磁性或光媒體所實行之檔案系統的持續記憶體，該實體記憶體以揮發性記憶體爲較佳（例如、電腦主記憶體，諸如動態 RAM 等）。

圖 2 爲藉由設置與實體記憶體 202 通訊之記憶體控制器 220 來實施 HICAMP 的較佳實施例圖。一或多個處理器 230 可經由記憶體控制器 220 來存取實體記憶體 202，記憶體控制器 220 提供 HICAMP 組織的記憶體模型給處理器。在此例中，具有三個處理器，210、212、及 214。因此，HICAMP 可應用到單一處理器或多處理器內文。下面將說明此例中有關記憶體控制器 220 的其他細節。

可利用硬體及/或軟體的組合來提供哪一些實體記憶體區塊是有效的指示。此外，在實施本發明時可利用"有效區塊"的各種定義。例如，若在整個系統初始化後，已由 HICAMP 系統配置或初始化，則實體區塊可被視作有效的。利用此種方式，無法回收利用先前已使用過但目前未使用中之實體記憶體區塊。因爲某些電腦系統被設計成當記憶體變低時重新初始化，所以此種相對較小的"有效"指示已可滿足此種系統。

提供"有效"指示的另一方式係爲各個記憶體區塊維持指示其是否有效的旗標。可根據前述之 HICAMP 法的初始

化/配置來設定此種旗標。另一選擇是，可以分開操作來設定用於區塊活動性的旗標，藉以決定哪一些區塊是有效的（例如、在標記清掃垃圾收集中所進行的一般）。

提供"有效"指示的另一方式係爲各個實體記憶體區塊維持參考計數，其中有效區塊具有對應的參考計數> 0，及失效區塊具有參考計數=0。在某些例子中，此種參考計數可區別具有各種位置的參考，諸如處理器暫存器中的參考，實體記憶體區塊中的參考，及/或虛擬對實體區塊 ID 映射中（VPBIM）的參考等。

在較佳實施例中，每一區塊提供一組標記，指示哪一些區塊包含相對於原始資料的實體和虛擬區塊參考，使此種區塊參考能夠被處理成參考計數，標記清掃垃圾收集或類似"有效"決定計畫的一部分，及能夠防止應用程式位準處理在沒有 HICAMP 實施例的知識之下就製造區塊參考。其他標記可支援偵測循環參考和提供專門更新語意，諸如用於高內容資料區塊的更新時合併等。

在 HICAMP 的較佳實施例中，提供 VPBIM 以替實體記憶體的一些或所有內容映射虛擬區塊 ID 至實體區塊 ID。藉由提供此種映射，能夠在限制負擔的同時自動更新構造資料的代表。例如，物件中的字元字串描述欄位可被實施當作區塊中的記憶體胞元，以儲存虛擬區塊 ID。藉由以根實體區塊來產生修正的字元字串資料項目且將用於此實體區塊的識別符號儲存在對應於虛擬區塊 ID 的映射登錄點中，以更新此敘述。

圖 3a-d 爲與虛擬和實體區塊 ID（在圖式中分別縮寫成 VID 和 PID）有關的 HICAMP 記憶體使用之一些例子圖。爲了說明簡單，此例子每一區塊儲存單一值。在圖 3a 上，指令 302 的執行產生如所示之記憶體 304 的內容。具體而言，指令 302 中的第一指定使值 37 儲存在新的實體區塊（即、PID1）中，及使 VID1 與 PID1 相關，因爲指定係到 VID，而非 PID。同樣地，指令 302 中的第二指定使值 125 儲存在新的實體區塊（即、PID2）中，及使 VID2 到 PID2 相關。指令 302 中的第三指定只使 VID3 相關到 PID1，因爲用於 VID3 的值是已儲存之值的重複。此重複的抑制與圖 1 的習知記憶體模型成強烈對比，在圖 1 中係使用 PID 做記憶體存取，且重複值係儲存在記憶體中。

在圖 3b 上，指令 306 的執行產生如所示之記憶體 308 的內容。指令 306 不同於指令 302 之處只在於藉由 VID1 的最後指定的添加以具有值 25。此第四指令使值 25 儲存在新的實體區塊（即、PID3）中，及使 VID1 與 PID3 相關。

在圖 3c 上，指令 310 的執行產生如所示之記憶體 312 的內容。指令 310 不同於指令 306 之處只在於 VID3 的最後指定的添加以具有值 125。此第五指令只使 VID3 與 PID2 相關，因爲值 125 已在記憶體中。

在圖 3d 上，指令 314 的執行產生如所示之記憶體 316 的內容。指令 314 不同於指令 310 之處只在於用於 VID4

的最後指令的添加以指向 VID3。此第六指令使位址 VID3 儲存在新的實體記憶體區塊 PID4 中，及使 VID4 相關到 PID4。

此例子顯示出藉由參考具有 VID 的記憶體內容，在消除所有重複的實體儲存同時，可提供指定陳述和指標的通常邏輯。一旦產生實體區塊，則其內容不可改變。在圖 3a-d 的例子中，字母"A"出現在有效的實體記憶體區塊旁邊，而字母"F"出現在釋放的（即、可用於配置）記憶體區塊旁邊。在圖 3c-d 上，實體區塊 PID1 被標記成釋放，因爲沒有參考到它。

圖 4 爲用以實施 VPBIM 的適當邏輯結構之例子。在此例中，VPBIM 中的各個登錄點包括 VID，對應 PID，及那 VID 的參考數目。圖 5a-d 分別圖示具有此結構和對應於圖 3a-d 的例子之 VPBIM 例子。VPBIM 通常是 VID 對 PID 的多對一映射，因爲同一 PID 可對應於幾個 VID，如這些例子所示一般。因此，VID 的使用可被視作管理多個連至同一實體資料區塊的不同參考之系統方法，這是由於在 HICAMP 的記憶體特性中消除實體重複所自然產生的情況。在另一實施例中，VID 不需要是 VPBIM 中的明確登錄點。取而代之的是，映射可以是不言明的。例如，（PID、參考計數）配對的陣列可當作具有充當 VID 之陣列索引的 VPBIM。可利用與任何其他多區塊資料項目相同的方式將 VPBIM 儲存在 HICAMP 組織記憶體中。

在較佳實施例中，HICAMP 非重複寫入能力係經由以

內容來區塊擷取（BFBC）指令來提供的，此指令具有區塊資料當作輸入且提供區塊識別符號當作輸出。可考慮兩種情況。若輸入區塊資料是存在於有效實體記憶體區塊中之資料的重複，則由 BFBC 指令所傳回的識別符號是此現存的有效記憶體區塊之位址。

若輸入區塊資料非存在於有效實體記憶體區塊中之資料的重複，則由記憶體控制器配置新的實體資料區塊，其內容被設定成輸入區塊資料，及此新的有效實體記憶體區塊之區塊位址係由 BFBC 指令傳回。

藉由經由 BFBC 指令來建構記憶體存取，可藉由記憶體控制器來執行管理重複抑制的處理，及不需要在應用程式位準中分開考慮上述兩情況。具體而言，經由 BFBC 記憶體存取所代表的應用程式演算法不需要考慮記憶體重複抑制的細節，因爲係由 BFBC 抽象的實施所處理。在利用 VPBIM 映射的例子中，由 BFBC 指令所傳回的區塊位址可以是實體區塊 ID 或虛擬區塊 ID。典型上，對應用程式位準 BFBC 指令而言，傳回虛擬區塊 Id 較爲有用。

在較佳實施例中，有效實體記憶體區塊被組織成兩或更多個直接非環式圖形（DAGs）。在此種例子中，各個多區塊資料項目具有其自己的 DAG，由非環式的條件排除 DAG 內部之參考的閉環，及 DAG 的方向性提供清楚的根區塊以供整個多區塊資料項目參照。爲了排除總體的循環參考迴圈，一組 DAGs 不包括多 DAG 循環參考迴圈較佳（如、包括兩或更多的 DAGs 之指標參考的閉環）。

HICAMP 的實施依賴提供實體記憶體的內容可定址性之各種方法。其中一方式係提供循序讀取和比較能力給實體記憶體的一些或所有內容。另一方式係提供平行讀取和比較能力給實體記憶體的一些或所有內容。例如，可利用單一比較器來實施循序方法，該單一比較器配置成循序讀取有效實體記憶體區塊的內容和比較它們以輸入資料。同樣地，可藉由設置（如、在硬體中）對應各個實體記憶體區塊的分開比較器以實施平行方法，使得可同時執行有效區塊內容的所有比較以輸入資料。因此，在速度和成本之間要有所權衡以考慮何時決定利用循序或平行比較（或混合方式）來提供內容可定址性。

在一較佳實施例中，藉由將實體記憶體分成 N 個記憶庫來提供用於 HICAMP 的內容可定址性，其中 N 是整數 >1。當將資料寫入至此記憶體時，將具有 N 個可能輸出的雜湊函數應用到區塊資料以提供雜湊值。雜湊值被用於選擇對應的記憶庫，此對應的記憶庫將根據慣常的 HICAMP 規約（即、只有在記憶庫中不會因此而產生重複的情況，才可以在該相關的記憶庫中產生新的區塊）來儲存資料。以此方式，用於內容定址的讀取和比較能力一次只要處理一記憶庫，而非整個實體記憶體。以此種規劃，如果各個記憶庫具有 M 個區塊和 M 個比較器，則可提供快速平行比較，如此實際上比實施全區塊位準平行（即、NM 比較器）來得不昂貴。

例如，設想根據上述規劃將字串"abc"儲存到記憶體

。假設"abc"雜湊成 3，則在實體記憶體中，只有在記憶庫 3 中，可將"abc"儲存在一有效實體記憶體區塊。因此，避免重覆，只要檢查記憶庫 3 中的區塊即可。

圖 6a-b 爲多區塊資料物件的簡單例子。圖 6a 爲在本發明的一實施例中由指令 602 指定三元素串列[1,2,3]給 VID1 之結果圖。記憶體 604 的最後內容可瞭解如下。各個串列元素得到它自己的實體區塊，因位在此例中沒有元素被重複，及虛擬 ID VID3、VID4、及 VID5 對應於陣列元素。串列的根是在 VID1 所參考的區塊中，及其內容是到串列的第一元素之指標（即、VID3），及到串列的剩餘部分（即、到串列[2,3]）之指標（即、VID2）。在 VID2 中，具有到串列[2,3]的第一元素之指標（即、VID4），及到串列[2,3]的第二元素之指標（即、VID5）。

圖 6b 的例子類似於圖 6a 的例子。圖 6b 的記憶體內容 606 和圖 6a 的 604 之間不同處只在 VID1 及 VID2 的區塊參考之內容中。具體而言，在圖 6a 的例子中從其尾端建立串列，而圖 6b 的例子中從其一開始建立串列。爲了消除多區塊重複，對多區塊唯一性提供規約（如上述）而言，指定唯一組織給任何指定系統所支援的所有多區塊資料結構是重要的。適當利用此種規約，同一物件不可能具有如圖 6a-b 所示一般的兩種不同記憶體代表。

2)實施細節

區塊組織

在一例示實施例中 HICAMP 實體記憶體區塊可被建構如下：

refCount | tags | inLevel | exLevel | data

refCount（參考計數）欄位包含此區塊的充分參考之數目。下面所說明的 backRef 並不視作"充分參考"。零的 refCount 代表區塊釋放且可用於重新使用。

tags 欄位用以指示相關 data 欄位的各自的子欄位：

i) 00-資料

ii) 01-intraRef-包含代表在多區塊數結構中是在內部且隸屬於此區塊的區塊之區塊 Id。即子樹參考，可計入此區塊中的參考計數的增加。若任一子欄位是 intraRef，則區塊中的所有子欄位需要都是 intraRef。

iii) 11-extraRef-包含代表另一區塊的區塊 Id，可計入此種另一區塊中的參考計數的增加。

iv) 10-backRef-包含代表另一物件的區塊 Id，但是不代表可作爲此另一物件的參考計數增加。

extraRef 和 backRef 值是虛擬區塊 Id。tags 欄位可包括單一"合併更新"旗標，此旗標指示區塊應在更新時與目前區塊內容合併，而非取代這些內容。

inLevel 欄位指示從此節點藉由 intraRefs 直到一不包含 intraRefs 的節點之最大距離。例如，在可變尺寸物件的典型階層式（樹或 DAG）代表中，此位準（level）是

此樹或 DAG 中的高度。

exLevel 欄位需要是比其所連接至的任何具有 extraRef 之節點的 exLevel 大一者，經由節點直接或間接到由 intraRef 可到達者。例如，若此節點是代表多區塊物件之樹的根，則該之 exLevel 大於此物件所有具有 extraRef（等同習知程式化中的智慧型指標）的節點之 exLevel。執行 exLevel 的此限制是確保在 HICAMP 記憶體組織中沒有有害的循環參考迴圈之一方法。

分析

爲了評估 HICAMP 對各種情況的可應用性，限制添加到記憶體系統和控制器之欄位的尺寸是有幫助的，如下述。

不同於習知記憶體系統，其藉由系統中的各個內容至多只能具有單一拷貝來約束指定區塊的參考數目。尤其是，最糟的情況是所有 DAG 均具有同一區塊的單一共同字首（prefix），如此此一區塊具有最大的參考數目。各個 DAG 必須具有內節點區塊以參考此共有區塊，加上各個 DAG 必須具有至少一特有區塊以確保各個根區塊是可區別的。而且，各個內節點需要被另一內節點參考直到一些根節點。然後，利用 64 位元組區塊和兆位元組（2^{40}）位元組記憶體（及因此 2^{33} 區塊），最糟情況的參考數目被侷限在 2^{32}，此種根節點，假設它使用 2 區塊來產生最小額外唯一參考 DAG，及每一最小 DAG 使用另一區塊內節點

來參考這些根節點（即、32 位元 refCount 是足夠的）。在此結構中，樹葉是共有區塊和特有區塊的另一順序，及其他 N 使用當作整個 DAG 內的內節點。

藉由每一區塊儲存 inLevel 和 exLevel 來避免參考循環。inLevel 欄位需要足夠大以容納最大的 intraRef 建構物件。該位元數是 loglogN/B，其中 N 是最大的單一物件及 B 是每一區塊的位元組數目。因此，6 位元將容納物件達 $B*2^{63}$。exLevel 需要容納 extraRef 的深度，此深度通常遠小於 100。例如，指向具有子物件的物件之目錄基本上是在 exLevel3。因此，8 位元似乎更適合此欄位。

在一實施例中，區塊是 20 位元組的資料，4 位元組的 RefCount，2 位元組的 levels（inLevel 以及 exLevel），1 位元組的標記，導致其負擔近乎 25 百分比。較大的區塊能夠每一區塊儲存 4 區塊 Id，支援四重樹當作階層式資料代表。其他實施例可支援較大的區塊尺寸和多區塊尺寸以進一步減少負擔。

記憶體控制器

在一例子中，圖 2 的記憶體控制器 220 包括 3 主要組件。一區塊 Id/位移擷取器 204，其回應處理器要求，定位並傳回在記憶體儲存中對應該指定位移的資料區塊，該位移係以一階層式區塊結構中之藉由區塊 Id 識別的區塊爲根。一 VPBIM 206，按照維持用於 PIDs 的參考計數，轉譯虛擬區塊 IDs（v 區塊 Ids 或 VIDs）及實體區塊 IDs（p

區塊 Ids 或 PIDs）之間的彼此轉換並管理配置及釋放這些映射。一區塊資料目錄 208，實施指定資料和標記規格給區塊（如上述），若區塊尚未存在，則配置和初始化此種區塊。實施此映射的各種技術是眾所皆知的。在一實施例中，二元內容可定址的記憶體可被使用當作資料儲存體。在此例中，refCount 及 level 可分開儲存於較不昂貴的 DRAM 中。在另一實施例中，映射可使用自查閱資料結構的選出者，諸如先前說明的以樹或雜湊表爲主的實施。

在例示實施例中，記憶體控制器管理被組織成 B 位元組的區塊之記憶體的 DRAM 記憶庫，其中 B 被預期是在 16 到 64 位元組的範圍中，但是 B 在每一系統中是固定的（至少在系統操作期間）。其亦具有 32 位元參考計數欄位的陣列，每一區塊一個。其中一實例是於系統中各個實體區塊具有一登錄點的 DRAM。記憶體控制器提供用以以指定指數來自動增加和減少參考計數欄位的操作，及如下述，當參考計數被減至零時釋放該區塊。

可使用釋放區塊的參考計數欄位來處理區塊配置以將其鏈結到釋放的區塊表。也就是說，釋放的記憶體區塊之參考計數欄位包含釋放串列中之下一釋放記憶體區塊的索引。當系統初始化時，所有釋放區塊被佇列到此釋放串列上。當需要新區塊時，由記憶體控制器從釋放串列的頭解佇列，且將其參考計數重設成 1，對應於新的參考。當區塊的參考計數成爲零時，記憶體控制器增加區塊到釋放表的頭。對應地，可從釋放串列去除不良的記憶體區塊，如

此避免配置它們，此與頁位準的習知技術相同，但是具有更精密的顆粒。藉由維持釋放串列尾指標，釋放的區塊可被添加到釋放串列的尾端，如此只要可能就不會重新用到它們。在記憶體晶片的使用壽命期間支援有限的寫入次數之諸如快閃記憶體等技術來實施記憶體時，此非必要的精鍊提供遍及區塊的"穿戴整平"之形式。

結合額外位元與參考計數機構來指示其在此形式中當作釋放"下一"欄位之使用以保護免於諸如對釋放區塊的錯誤參考等錯誤行爲是可行的。

在此方式中，參考計數欄位需要足夠大到能夠儲存數個區塊 Id。另一選擇是，可具有 K 個釋放串列，使得第 i 釋放串列中的所有區塊具有 i 在其低階位元中，去除儲存它們的需要。下面進一步說明使用遍及 K 個釋放串列劃分區塊來當作實施內容可定址的查閱或以內容來區塊擷取（BFBC）之部分。

與習知記憶體相同地執行以實體區塊 id 來區塊擷取。記憶體控制器解碼到 DRAM 記憶庫的區塊 Id，及在此記憶庫內的列/行，發出此區塊的讀取和傳回資料。最不尋常的型態是支援以內容來區塊擷取（BFBC），如下述。

在理想或邏輯實施例中，主記憶體被實施當作對應於不包括參考計數欄位之區塊尺寸寬度的二元內容可定址記憶體（CAM）。因此，以內容來區塊擷取（BFBC）記憶體控制器操作傳遞區塊資料到 CAM，及若存在的話，則接

收回區塊或"列"id，否則就指示不存在。在後一例子中，其如上配置區塊及以相關資料書寫區塊。依據遍及整個記憶體系統之資料的唯一性（即、不會有兩命中（hit)），利用在各個記憶庫中平行執行的查閱可將記憶體分成多個記憶庫。藉由建立比較邏輯到個別 DRAM 晶片內，此方式是可行的。然而，二元 CAM 記憶體目前非常昂貴，且相對習知 DRAM 而言非常耗電。

能夠使用習知 DRAM 的實施可降低比較器的數量，使得在記憶庫中各個 K 列具有單一比較器，而非每列一個。然後，將區塊內容雜湊成 0 到 K-1，比如說 h，及各個比較器被要求比較資料與其第 h 相關列。若區塊匹配的話，則比較器報告命中和區塊數。爲了準確地進行此工作，在 K 個釋放串列中維持釋放區塊。雜湊成 h 之未命中資料時，由第 h 釋放串列配置區塊來儲存資料。

使用上述方式，記憶體控制器能夠包含 C 個比較器和在 C 個獨立記憶庫中存取記憶體，各個係由習知 DRAM 晶片來實施。然後，記憶體控制器值行平行記憶體擷取和比較，加上對未命中的區塊之配置。利用適當的雜湊函數，在用完所有釋放串列之前應該不可能明顯地用完一釋放串列。相反地，釋放串列的劃分不會明顯降低記憶體的有效尺寸。當如上述被使用當作釋放串列中的"下一"鏈結時，K 個釋放串列可被用於降低欲儲存在參考計數欄位中所需的位元數。

對一 HICAMP 的特定實例而言，K 的尺寸和比較器的

數目可以是特定的，對軟體而言是易懂的。其依賴記憶體控制器上可行的 I/O 接腳數目。而且，在某些例子中，每一 BFBC 要求從 DRAM 每一記憶庫發出多個讀取是可行的，藉由每一 BFBC 操作中多次使用它們可有效增加比較器數目。

VPBIM 機構

虛擬對實體區塊 Id 映射（VPBIM）可被實施當作以虛擬區塊 id（v 區塊 Id）來檢索之記憶體陣列。各個登錄點具有欄位：

[p 區塊 Id|refCount]

使用 p 區塊 Id 欄位將釋放登錄點鏈結在一起，類似於爲 p 區塊 s 所說明的規劃。利用用於區塊 Id 的 40 位元和每一參考計數的 32 位元，各個登錄點是 9 位元組。記憶體控制器被組配成相對於 p 區塊 s 支援足夠的登錄點，如此 VPBIM 登錄點不是有限的資源。假設記憶體之每 4 字元 1 指標的比例，及每區塊 4 字元的資料，則記憶體控制器可每區塊提供一 VPBIM 登錄點，如此 p 區塊 Id 欄位足夠大到用於釋放串列。理想上，記憶體控制器可被組配成支援不同尺寸的 VPBIM。

就 VPBIM 而言，記憶體控制器支援操作成：

a)配置 VPBIM 登錄點且以指定 p 區塊 Id 和參考計數 1 來初始化，增加相關 p 區塊 Id 的參考計數。此簡單包含

解佇列釋放的 VPBIM 登錄點和使其初始化。

b)傳回對應於指定 v 區塊 Id 的 p 區塊 Id。此僅由 v 區塊 Id 來索引 VPBIM 陣列及傳回那登錄點中的 p 區塊 Id。

c)爲指定 v 區塊 Id 增加參考計數。此僅由 v 區塊 Id 來索引 VPBIM 陣列及增加那位置的參考計數。

d)爲指定 v 區塊 Id 減少參考登錄點及若參考計數是零，則使登錄點釋放，減少 p 區塊 Id 和增加此 VPBIM 登錄點到釋放串列。

可使用韌體和用於諸如配置等綜合操作的內部微控制器來實施上述記憶體控制器操作。簡單的性能緊要操作可使用硬導線邏輯。記憶體控制器性能係侷限於 DRAM 性能，如同習知架構一般。因此，HICAMP 處理性能高度依賴處理器元件位準中快取的有效使用，如下述。

快取

HICAMP 處理器元件係依據實施習知暫存器對暫存器演算法、邏輯等指令的 CPU 區塊與記憶體暫存器載入操作，但是須再增加該新的 HICAMP 記憶體系統結合之特定操作。此處理器元件包括處理器快取記憶體以提供到資料的有效存取，該資料係利用時間和空間的局部性加以存取，在習知架構中被證明良好。與習知處理器元件競爭之下，HICAMP 的主要性能挑戰是提供到記憶體的快速存取。習知處理器元件的主要觀點是有效快取。

HICAMP 處理器快取記憶體被建構成快取線的集合，

其被量身定作成可與記憶體區塊尺寸相容，典型上匹配此區塊尺寸，與習知處理器快取記憶體相同。同樣地，其包括習知快取記憶體目錄，映射區塊 Id 到快取線，大概具有一些合理的設定關聯性，諸如 4 或 8 等。可利用 HICAMP 資料對區塊 Id 映射（即內容可定址性）來擴大此快取記憶體機構。此映射係藉由記憶體控制器而非要求快取記憶體中的資料，而實施類似於 BFBC 操作。快取記憶體中的區塊又構成到記憶體控制器的參考。當從快取刪除區塊時，在記憶體控制器中減少其參考計數（如此取代必須跳過由暫存器所參考的任何區塊）。

該快取記憶體支援回收一實體區塊 Id 的資料區塊之操作，若該實體區塊 Id 存在的話。其亦支援傳回指定資料區塊的 p 區塊 Id，若該實體區塊 Id 存在的話，或可依賴記憶體控制器以執行此映射。從經過比較的資料，快取記憶體分別地儲存區塊的實體區塊 Id。這是 p 區塊 Id 的寬度之個別記憶庫，具有用於各個具有 R 列的 C 個記憶庫之 R*C 登錄點，即、每一快取線一個。只要支援從區塊到 p 區塊 Id 的映射，可將此與支援 p 區塊 Id 的查閱之快取記憶體目錄加以組合。

在未命中（miss）的例子中，快取記憶體也支援在被稱爲"打開"的暫時狀態中之一區塊的配置，該狀態在快取記憶體目錄中以額外的位元來代表。如此使區塊能夠追加地寫入快取記憶體中，然後當完成時"固定（committed）"到記憶體，此操作僅在決定該系統 p 區塊 Id 對應區塊資

料之時被允許。尤其是，當從最初配置暫存器移動其 p 區塊 Id 時或當其被儲存到記憶體內時，或當其正從快取記憶體去除時，方可固定區塊到記憶體。固定必須決定資料的系統指定 p 區塊 Id 及可能須移動該線到該資料所雜湊的列。固定可以決定具有此資料的區塊已存在於快取記憶體中，並藉由現存的區塊和快取線在區段中取代原有資料使原有資料釋放。

快取記憶體支援"打開"新或現存的快取線來寫入。若新的，則新線被配置旗標爲打開。若現存且線已具有額外參考，則資料被拷貝到新快取線且此新的快取線被配置旗標爲打開。固定快取線以關上修改，及依據上述內容來決定其區塊 Id。

快取記憶體在任何指定時間中只曾經包含至多一資料區塊的拷貝之事實提高習知處理器快取記憶體的快取記憶體利用性。具有各種軟體技術，諸如在區塊邊界上校直資料和碼及使用標準化指令順序等，期待增加快取記憶體中的共享。

記憶體操作

處理器經由 v 區塊 Id 和位移量來存取資料。例如，習知程式的碼區段可參考本身，如此程式計數器是由區塊 Id 所指出的碼區段中之位移量。硬體橫貫物件的階層式結構以利用特定位移量來定位資料及傳回資料，或若沒有的話，則指出記憶體例外。當傳回資料時，其被載入到處理

器暫存器內，其保留指出資料是否爲區塊 Id 的標記。若是，則其是參考的類型。因爲硬體查閱經由 intraRef 區塊 Id 來進行，所以反應於存取所傳回的資料中之區塊 Id 不是 extraRef 就是 backRef，即、其是有效的應用程式位準指標或參考。

在硬體中可提供資料擷取當作索引載入操作，指定區塊 Id 和位移量，傳回特定位置的資料或區塊 Id，或拋出例外/中斷。此操作向下遞迴出此樹，若存在的話以傳回在特定位移量的資料不然拋出例外並代表存取失敗。

在一實施例中，處理器支援一或多個重複暫存器，一或多個重複暫存器各個維持必須的狀態，以便經由多區塊資料 DAG 有效率地映射到目前記憶體位置且有效率地映射到下一位置的增加量。類似於習知電腦處理器中位址暫存器的使用，藉由經由重複暫存器來間接讀取和寫入而使讀取和寫入存取發生。

在較佳實施例中，由內容（BFBC）操作所擷取的硬體區塊，使用指定想要之區塊的資料和標記之參數，且傳回虛擬區塊 Id 給區塊。若 BFBC 參數指定區塊 Id，則只能從已保有這些區塊 Id 的暫存器產生。當使用 BFBF 操作在新區塊中指定其區塊 Id 時，用於區塊的參考計數亦增加。

讀取包含區塊 Id 和 BFBC 的現存區塊是處理器能夠使區塊 Id 存在於暫存器中之唯一方法。因爲無法從計算產生區塊 Id。結果，無論藉由有效指定其內容或藉由傳遞（

直接或間接）區塊 Id，硬體系統可在各個區塊上維持準確的參數計數，及處理只能夠存取已接收區塊 Id 的物件。

爲了圖解說明一實施例中的基本操作，藉由產生字串的字元到區塊暫存器內，然後進行 BFBC 操作以得到用於包含那些字元的區塊之區塊 Id，以零填充在尾端，使得 HICAMP 程式產生字串。若字串長於能夠塞進區塊的資料部位之字元 B 的數目，則程式遞迴地替字串中之每 B 個字元產生一區塊，且使用 BFBC 操作有效地將區塊連在一起以取得包含用於字串的字首和字尾的區塊 Id 之區塊。任何大於單一區塊尺寸的連續資料物件能以類似方式說明，例如，一陣列。

在一實施例中，大於區塊的物件可被視作包含用於字首樹的區塊 Id 和用於字尾樹的區塊 Id 之具有根節點的二元樹。區塊 Id 0 可被保留以代表資料是 0。在另一實施例中，樹可以是四重樹，具有每一內節點達 4 子節點。我們意指區塊樹代表一狀態，其在邏輯上係連續部分而作爲一物件，無論其是單一區塊或非簡易樹。

藉由具有用於另一物件的根節點之 extraRef，被儲存當作虛擬區塊 Id，且由 VPBIM 映射到實體區塊 Id，使得物件有效包含到另一物件的指標。因此，藉由自動更新 VPBIM 以映射虛擬區塊 Id 到新物件的實體區塊 Id，使得物件可被改成指向不同物件。在一實施例中，在包含特定實體區塊 Id 的目前映射上，比較交換操作支援此自動更新條件。因此，可藉由產生欲更新之物件的”虛擬”拷貝（

到原有物件的另一實體區塊 Id 參考剛好有效），修正此拷貝，藉以產生具有新實體區塊 Id 的新物件來達成自動更新，然後若其尚未從虛擬拷貝改變，則自動更新 VPBIM 映射以映射到新實體區塊 Id，及否則中止更新。然後，軟體可再重試操作，遵循與鎖定-釋放程式化類似的模式。

當產生具有對 v 區塊 Id 的新參考之區塊時，增加對應的 v 區塊 Id 上之參考計數。同樣地，當包含 v 區塊 Id 的區塊是釋放的時，減少對應的 v 區塊 Id 上之參考計數，當參考計數變成零時，有效使其釋放並且去除其對對應實體區塊 Id 的參考。以更熟悉的字眼陳述，VPBIM 爲 HICAMP 中的"智慧型指標"同等物（即、extraRefs）維持參考計數。

I/O 需要經過上述機構。即、沒有直接寫入至記憶體。各個 DMA I/O 僅以既定資料要求區塊，建立諸如上述一般的封包緩衝程式或碟區塊等物件。此消除預先配置 I/O 緩衝器的需要。若想要的話，只需要限制所接收的量。通常，可大量去除與網路建立有關的拷貝，尤其是若在網路酬載已達區塊限界之下啓動時。

現在已根據幾個例示實施例說明本發明，這幾個例示實施例係用於圖解說明所有型態而非限制。因此，無論在精於本技藝之人士從此處所包含的說明所衍生的硬體及/或軟體中，本發明都能夠在細節實施上具有許多變化。其中一變化係相關於將 HICAMP 實現當作經由單一邏輯記憶體控制器所操作的多處理器之實施例，如圖 2 所示。亦可

藉由在區塊的配置上之不同單元之間的適當同步化及對VPBIM的更新，利用多記憶體控制器和處理器晶片來實現本發明。在另一變化中，雖然說明用於通用電腦，但是本發明亦可應用到網路封裝開關，使胞元/區塊配置具有重複抑制，在無須多重播放的特別最佳化之下就能提供內容的共享。亦可應用到具有記憶體系統的其他裝置。在另一變化中，歸屬於硬體的機構亦可取代成以微碼或受保護的軟體或甚至在可靠性上有些微耗損之未受保護的機構來實施。所有此種變化和其他變化被視作在本發明的範圍和精神內。

【圖式簡單說明】

圖1爲習知電腦記憶體使用的例子圖。

圖2爲根據本發明的一實施例之電腦系統的概要區塊圖。

圖3a-d爲根據本發明的一實施例之電腦記憶體使用的例子圖。

圖4爲與本發明的一實施例一起使用的虛擬對實體區塊ID映射（VPBIM）之適當邏輯結構的例子圖。

圖5a-d爲對應於圖3a-d的例子之VPBIM例子圖。

圖6a-d爲能夠以兩不同方式在記憶體中代表相同多區塊物件之例子圖。

【主要元件符號說明】

102：指令

104：記憶體

202：實體記憶體

204：區塊 Id/Offset（位移量）擷取器

206：虛擬對實體區塊 Id(blockId)映射

208：區塊資料目錄

210：處理器

212：處理器

214：處理器

220：記憶體控制器

230：處理器

302：指令

304：記憶體

306：指令

308：記憶體

310：指令

312：記憶體

314：指令

316：記憶體

602：指令

604：記憶體

606：記憶體內容

五、中文發明摘要

發明之名稱：階層式不可改變的內容可定址的記憶體處理器

根據階層式不可改變的內容可定址之記憶體處理器（HICAMP）架構來提供改良的記憶體管理。在 HICAMP 中，實體記憶體被組織成兩或更多個實體記憶體區塊，各個實體記憶體區塊具有固定儲存容量。提供任一時間點中哪一些實體記憶體區塊是有效的指示。記憶體控制器提供非重複寫入能力，其中在寫入時將欲寫入至實體記憶體之資料與所有有效實體記憶體區塊的內容比較，藉以確保在完成非重複寫入之後沒有兩有效記憶體區塊具有相同資料。

六、英文發明摘要

發明之名稱：

Hierarchical Immutable Content-Addressable Memory Processor

Improved memory management is provided according to a Hierarchical Immutable Content Addressable Memory Processor (HICAMP) architecture. In HICAMP, physical memory is organized as two or more physical memory blocks, each physical memory block having a fixed storage capacity. An indication of which of the physical memory blocks is active at any point in time is provided. A memory controller provides a non-duplicating write capability, where data to be written to the physical memory is compared to contents of all active physical memory blocks at the time of writing, to ensure that no two active memory blocks have the same data after completion of the non-duplicating write.

十、申請專利範圍

1.一種電腦系統，包含：

一內容可定址的實體記憶體，其包括兩或更多實體記憶體區塊，各個該實體記憶體區塊具有固定儲存容量，其中提供哪一些該等實體記憶體區塊是有效實體記憶體區塊之指示；及

一記憶體控制器，其提供非重複寫入能力，其中在寫入時將欲寫入至該實體記憶體的資料與所有有效實體記憶體區塊之內容比較，藉以確保在完成該非重複寫入之後沒有兩該等有效實體記憶體區塊具有相同資料。

2.根據申請專利範圍第 1 項之系統，其中該記憶體控制器提供多區塊非重複寫入能力，包含：

提供多區塊資料規約，其爲需要兩或更多該等實體區塊來儲存之任何資料項目指定一唯一代表；

當將該資料項目寫入至該實體記憶體時，支援該多區塊資料規約，藉以可延伸該非重複寫入能力以確保沒有多區塊資料的任何實例之重複存在於該等有效實體記憶體區塊中。

3.根據申請專利範圍第 1 項之系統，其中各個該實體記憶體區塊具有相同儲存容量。

4.根據申請專利範圍第 1 項之系統，另外包含一或多個處理器，該一或多個處理器與該記憶體控制器通訊且只有經由該非重複寫入能力才能夠寫入至該實體記憶體。

5.根據申請專利範圍第 1 項之系統，其中該實體記憶

體是揮發性的。

6.一種電腦系統記憶體管理方法，包含：

提供內容可定址實體記憶體，其包括兩或更多實體記憶體區塊，各個該實體記憶體區塊具有固定儲存容量；

提供哪一些該等實體記憶體區塊是有效實體記憶體區塊之指示；及

提供非重複寫入能力，其中在寫入時將欲寫入至該實體記憶體的資料與所有有效實體記憶體區塊之內容比較，藉以確保在完成該非重複寫入之後沒有兩該等有效實體記憶體區塊具有相同資料。

7.根據申請專利範圍第 6 項之方法，另外包含：

提供多區塊資料規約，其爲需要兩或更多該等實體區塊來儲存之任何資料項目指定一唯一代表；

當將該資料項目寫入至該實體記憶體時，支援該多區塊資料規約，藉以可延伸該非重複寫入能力以確保沒有多區塊資料的任何實例之重複存在於該等有效實體記憶體區塊中。

8.根據申請專利範圍第 6 項之方法，其中該提供哪一些該等實體記憶體區塊是有效實體記憶體區塊之指示包含：

若根據該非重複寫入能力已初始化或配置，則將實體記憶體區塊視作有效。

9.根據申請專利範圍第 6 項之方法，其中該提供哪一些該等實體記憶體區塊是有效實體記憶體區塊之指示包含

：

維持一旗標，該旗標用於各個該實體記憶體區塊，指示該實體記憶體區塊是否有效。

10.根據申請專利範圍第 6 項之方法，其中該提供哪一些該等實體記憶體區塊是有效實體記憶體區塊之指示包含：

爲各個該等實體記憶體區塊維持一參考計數，該參考計數用於指示各個該實體記憶區塊之有效參考的該數目；及

識別具有對應參考計數＞ 0 之實體記憶體區塊當作有效實體記憶體區塊。

11.根據申請專利範圍第 6 項之方法，另外包含提供虛擬區塊 IDs（識別）對實體區塊 IDs 的映射給一些或所有該等有效實體記憶體區塊。

12.根據申請專利範圍第 11 項之方法，其中該非重複寫入能力係藉由提供以內容來區塊擷取（BFBC）指令來實施，該指令具有區塊資料當作輸入且提供區塊位址當作輸出，

其中在執行該 BFBC 指令之前，若該區塊資料是該等有效實體記憶體區塊的其中之一的資料之重複，則該區塊位址是有效實體記憶體區塊的位址，及

其中在執行該 BFBC 指令之前，若該區塊資料不是該有效實體記憶體區塊的其中之一的資料之重複，則該區塊位址是重新配置的實體記憶體區塊之位址。

13.根據申請專利範圍第 12 項之方法，其中該區塊位址不是實體區塊 ID 就是對應的虛擬區塊 ID。

14.根據申請專利範圍第 6 項之方法，其中該有效實體記憶體區塊被組織成複數直接非環式圖形（DAG）。

15.根據申請專利範圍第 14 項之方法，其中該複數 DAG 被限制成未包括多 DAG 循環參考迴圈。

16.根據申請專利範圍第 6 項之方法，另外包含提供循序讀取和比較能力給該實體記憶體的一些或所有內容以提供內容可定址性。

17.根據申請專利範圍第 6 項之方法，另外包含提供平行讀取和比較能力給該實體記憶體的一些或所有內容以提供內容可定址性。

18.根據申請專利範圍第 6 項之方法，另外包含：

提供將該實體記憶體劃分成 N 記憶庫，其中 N 是整數 > 1；

根據具有 N 種可能輸出的雜湊函數來雜湊區塊資料以提供雜湊值給該區塊資料；

提供該區塊資料的讀取和比較能力給對應於該雜湊值之該實體記憶體的該記憶庫之內容；

藉以提供內容可定址性。

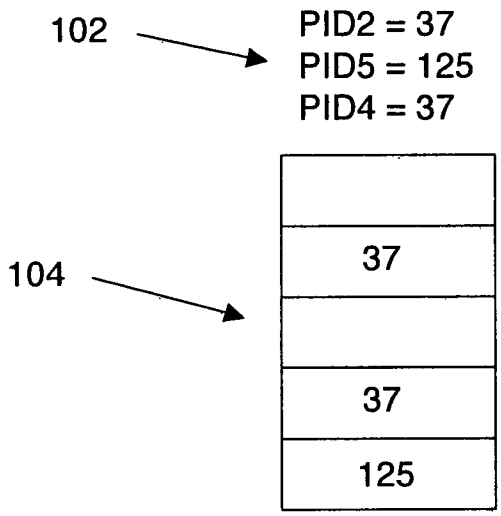
102
PID2 = 37
PID5 = 125
PID4 = 37
104
37
37
125

圖1

(先前技術)

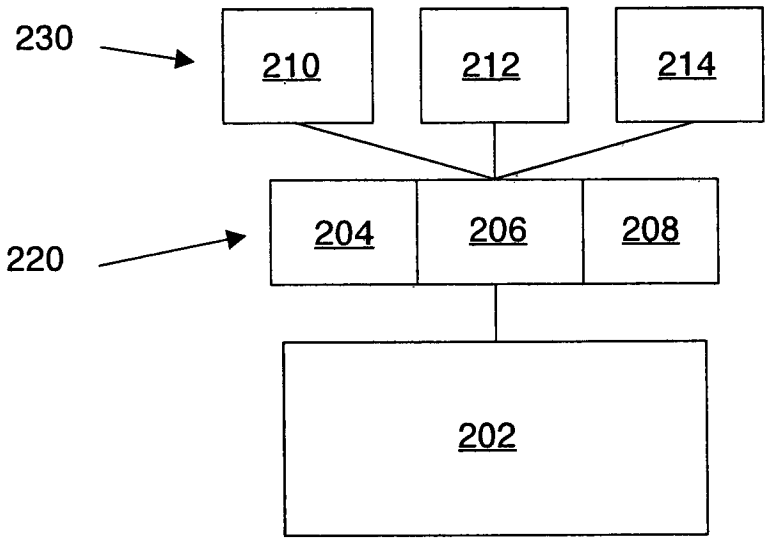
230
210
212
214
220
204
206
208
202

圖2

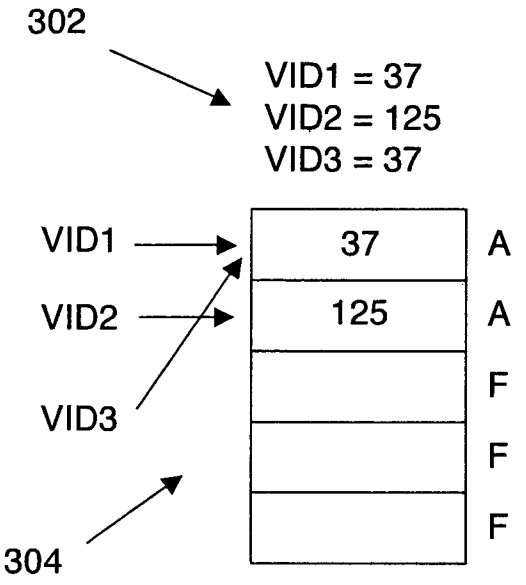

圖3a

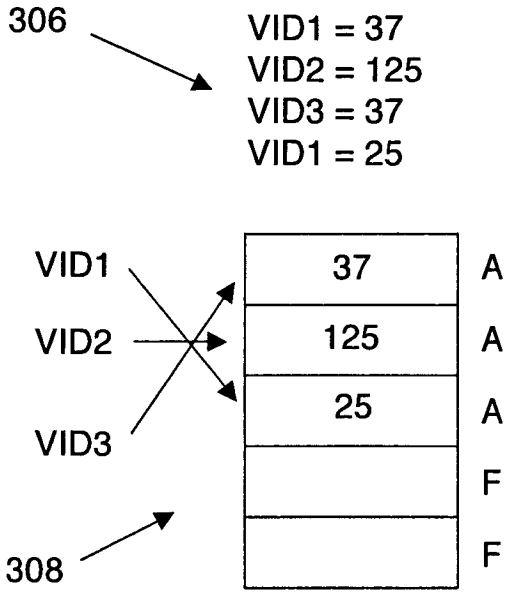

圖3b

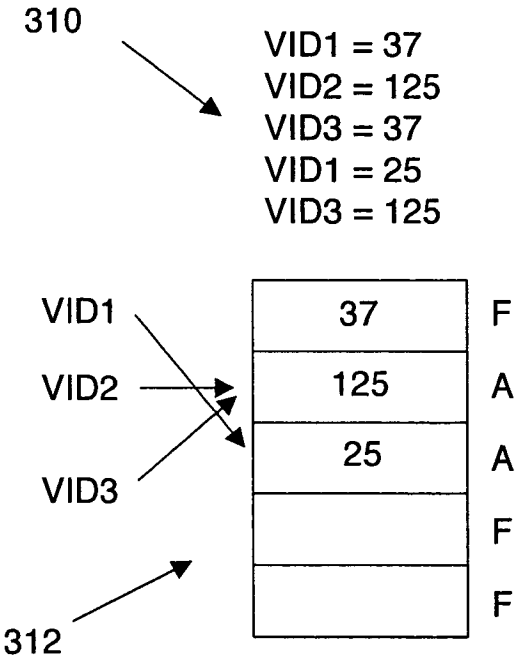

圖3c

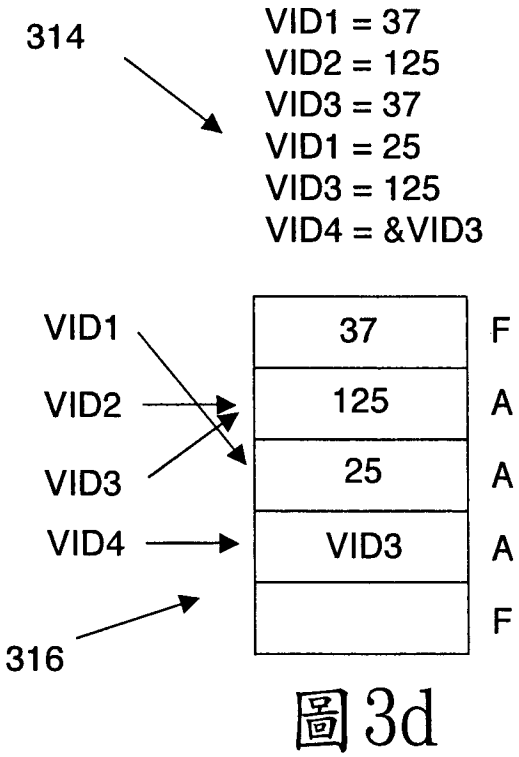

圖3d

VID	PID	計數

圖4

VID1	PID1	1
VID2	PID2	1
VID3	PID1	1

圖5a

VID1	PID3	1
VID2	PID2	1
VID3	PID1	1

圖5b

VID1	PID3	1
VID2	PID2	1
VID3	PID2	1

圖5c

VID1	PID3	1
VID2	PID2	1
VID3	PID2	2
VID4	PID4	1

圖5d

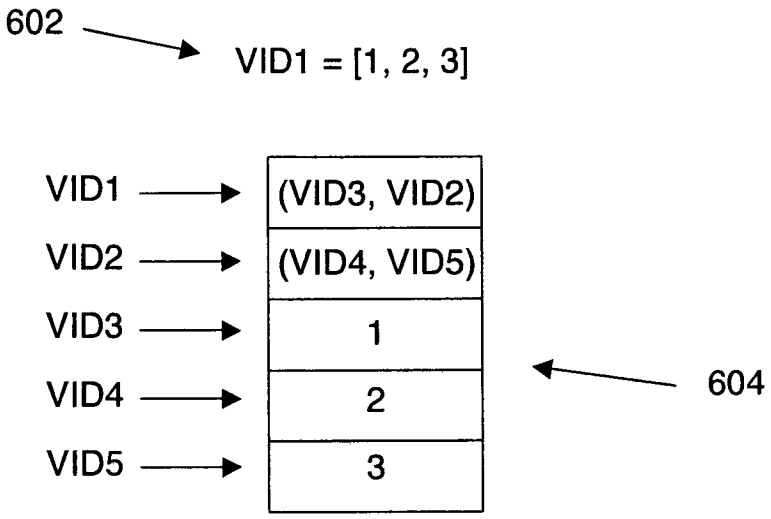
602
VID1 = [1, 2, 3]
VID1
VID2
VID3
VID4
VID5
(VID3, VID2)
(VID4, VID5)
1
2
3
604

圖6a

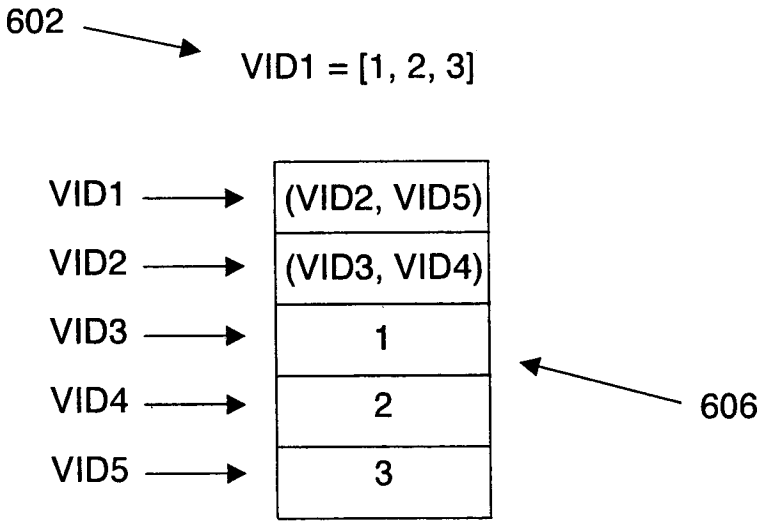
602
VID1 = [1, 2, 3]
VID1
VID2
VID3
VID4
VID5
(VID2, VID5)
(VID3, VID4)
1
2
3
606

圖6b

七、指定代表圖

（一）、本案指定代表圖為:第（3a）圖

（二）、本代表圖之元件代表符號簡單說明：

302：指令

304：記憶體

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：無

9710265?

藉由在區塊的配置上之不同單元之間的適當同步化及對VPBIM的更新，利用多記憶體控制器和處理器晶片來實現本發明。在另一變化中，雖然說明用於通用電腦，但是本發明亦可應用到網路封裝開關，使胞元/區塊配置具有重複抑制，在無須多重播放的特別最佳化之下就能提供內容的共享。亦可應用到具有記憶體系統的其他裝置。在另一變化中，歸屬於硬體的機構亦可取代成以微碼或受保護的軟體或甚至在可靠性上有些微耗損之未受保護的機構來實施。所有此種變化和其他變化被視作在本發明的範圍和精神內。

【圖式簡單說明】

圖 1 爲習知電腦記憶體使用的例子圖。

圖 2 爲根據本發明的一實施例之電腦系統的概要區塊圖。

圖 3a-d 爲根據本發明的一實施例之電腦記憶體使用的例子圖。

圖 4 爲與本發明的一實施例一起使用的虛擬對實體區塊 ID 映射（VPBIM）之適當邏輯結構的例子圖。

圖 5a-d 爲對應於圖 3a-d 的例子之 VPBIM 例子圖。

圖 6a-b 爲能夠以兩不同方式在記憶體中代表相同多區塊物件之例子圖。

【主要元件符號說明】