I465934

(19)中華民國智慧財產局

(12)發明說明書公告本 **(11)證書號數：TW I465934 B**

(45)公告日： 中華民國 103 (2014) 年 12 月 21 日

(21)申請案號：098140587 (22)申請日： 中華民國 98 (2009) 年 11 月 27 日

(51)Int. Cl.： ***G06F15/177 (2006.01)***

(30)優先權：2008/12/02 美國 12/315,331

(71)申請人：英特爾公司 (美國) INTEL CORPORATION (US)
美國

(72)發明人：哈森勞夫 威廉 HASENPLAUGH, WILLIAM (US)；艾姆 喬爾 EMER, JOEL (US)；佛舒姆 崔格夫 FOSSUM, TRYGGVE (NO)；傑利爾 艾梅兒 JALEEL, AAMER (US)；史泰利 賽門 STEELY, SIMON (US)

(74)代理人：惲軼群；陳文郎

(56)參考文獻：

TW	I297832	US	5394531
US	6341331B1	US	2008/0235457A1

審查人員：高元良

申請專利範圍項數：24 項 圖式數：5 共 23 頁

(54)名稱

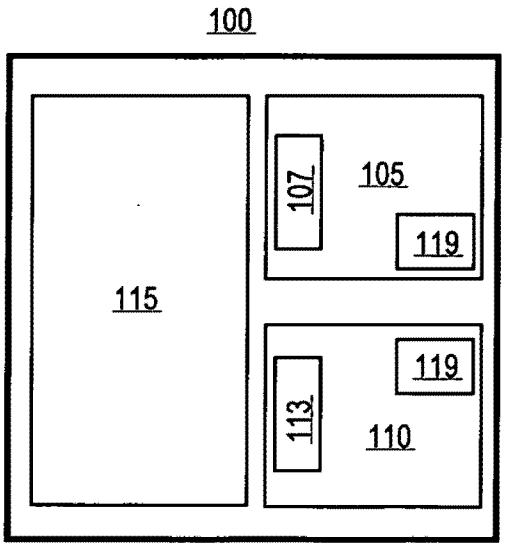
用以控制運算資源分配之裝置、系統及方法

APPARATUS, SYSTEM AND METHOD FOR CONTROLLING ALLOCATION OF COMPUTING RESOURCES

(57)摘要

用以在一電腦系統內致能資源分配最佳化的一技術。在一實施例中，一梯度分區演算法(GPA)模組用以持續量測性能及調整在多個資源類別之間的共享資源的分配，以達到最佳性能。

A technique to enable resource allocation optimization within a computer system. In one embodiment, a gradient partition algorithm (GPA) module is used to continually measure performance and adjust allocation to shared resources among a plurality of data classes in order to achieve optimal performance.



100・・・微處理器

105、110・・・處理器核心

107、113・・・區域快取部

115・・・共享快取記憶體

119・・・邏輯元件

第 1 圖

發明專利說明書

（本說明書格式、順序，請勿任意更動，※記號部分請勿填寫）

※ 申請案號：98140587

※ 申請日：98.11.27　　※IPC 分類：G06F 15/177 (2006.01)

一、發明名稱：（中文/英文）

用以控制運算資源分配之裝置、系統及方法
APPARATUS, SYSTEM AND METHOD FOR CONTROLLING ALLOCATION OF COMPUTING RESOURCES

二、中文發明摘要：

用以在一電腦系統內致能資源分配最佳化的一技術。在一實施例中，一梯度分區演算法(GPA)模組用以持續量測性能及調整在多個資源類別之間的共享資源的分配，以達到最佳性能。

三、英文發明摘要：

A technique to enable resource allocation optimization within a computer system. In one embodiment, a gradient partition algorithm (GPA) module is used to continually measure performance and adjust allocation to shared resources among a plurality of data classes in order to achieve optimal performance.

104年1月7日修正替換頁

四、指定代表圖：

(一)本案指定代表圖為：第（ 1 ）圖。

(二)本代表圖之元件符號簡單說明：

100...微處理器
105、110...處理器核心
107、113...區域快取部
115...共享快取記憶體
119...邏輯元件

五、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

六、發明說明：

【發明所屬之技術領域】

發明領域

本發明之實施例大體上有關於資訊處理領域及更特定地有關於在運算系統與微處理器中分配資源之領域。

【先前技術】

發明背景

在一電腦系統或處理器中分配資源是很困難的。例如，在一些電腦系統中，其中諸如快取空間、記憶體、執行資源等的資源，以一「靜態的」方式來分配 (即沒有根據變化的資源需求來改變)，該電腦系統或處理器可能對某些程序或執行緒服務不足或對其他程序或執行緒過度服務。但是，即使在一些習知的「動態的」資源分配方案中(即那些試著回應其等服務的執行緒、程序等的需求變化)，與該等資源的動態分配相關聯之額外負擔可能不值得此分配所帶來的性能利益。因此，改進型資源分配機制可提高處理器或運算系統性能，而不會導致可能削減該分配方案優勢的額外負擔。

【發明內容】

依據本發明之一實施例，係特地提出一種裝置，其包含：一梯度分區演算法(GPA)模組，用以持續控制至少一處理資源對至少一指令執行緒之分配，其中該GPA模組係用以持續對該處理資源執行測試，以決定可最佳滿足該至少一指令執行緒之處理需求的一動態調節器設定。

圖式簡單說明

本發明之實施例在附圖的圖中藉由範例方式說明，而非藉由限制方式說明，且其中相同的參考數字代表相同的元件且其中：

第1圖說明一微處理器的一方塊圖，其中本發明至少一實施例可被使用；

第2圖說明一共享匯流排電腦系統的一方塊圖，其中本發明至少一實施例可被使用；

第3圖說明一點對點互連電腦系統的一方塊圖，其中本發明至少一實施例可被使用；

第4A-4C圖說明邏輯元件的一方塊圖，其中本發明至少一實施例可被實施；

第5圖是一操作流程圖，其可用以執行本發明至少一實施例。

【實施方式】

較佳實施例之詳細說明

本發明之實施例是有關於一動態的資源分配技術，其可提高整個系統或處理器性能而不會導致將會削減本文描述的資源分配技術優勢的額外負擔。在一些實施例中，一動態的資源分配方案可用於多個不同的處理資源，諸如在執行緒之間的快取分配、記憶體資源，及執行資源。就說明的目的而言，依據至少一實施例，以下描述將主要集中在應用資源分配技術到由兩個或多個執行緒共享之快取資源的範例中。但是，本文描述之實施例也可被應用於其他

運算資源及除了這裡所特別描述外的其他數目的執行緒。

在一實施例中，一基於梯度的分區演算法(GPA)用以在多個執行緒之間分配快取空間。在一實施例中，該GPA使用一個狀態機，每一資料類別(例如，串流化資料、重複使用資料等)三個暫存器及不取決於資料類別的四個全域暫存器。在一實施例中，一GPA使用梯度下降(或「登山」)演算法的一變形以找到共享資源之最佳分配，針對每個資料類別，實施一實驗，假定此資料類別對於討論中的資源遞增之後遞減。GPA可接著針對兩種情況，量測一「全域良度」度量，且依據該實驗，將該資源之標稱定額分配給該討論中的資料類別。在一實施例中，GPA使用丘諾夫界限(Chernoff bound)來決定何時調整該資源之分區，而在其他實施例中，其他演算法也可用以決定何時調整資源之分區。此外，在一實施例中，藉由橫跨處理週期劃分該實驗流程(例如，時間多工頻寬或功率管理，快取或預取管理記憶體空間分區)，該上述實驗可同時針對每個資料類別來實施。

在一些實施例中，GPA應用到共享資源的多處理器系統或多核心處理器中可獲得比習知系統更高的性能及更低的功率消耗，在某種程度上是由於一些實施例在一持續基礎上主動最佳化資源管理。此外，實施例可使電池壽命更長，使每一刀鋒具有更高性能，產生高密度雲端運算等。

例如，在諸如此等使用伺服器處理器的一些運算環境中，一些資料被重複使用(例如，封包標頭、路由表、指令

資料、作業系統狀態、其他如統計的元資料等)且一些資料被串流化(例如，封包主體資訊)。依據一些實施例，使用未被管理的一快取部可能使該重複使用的資料無用，因為所有被重複使用的資料可能在其有機會被重複使用之前被該串流化資料逐出。在一實施例中，GPA可動態地決定何資料可保持在一快取部內且何資料被串流化，甚至在沒有以架構知覺最佳化來寫的應用程式中，其在應用程式中可能是有用的，諸如虛擬機群組(farm)，在其中的應用程式不感知該處理架構，其他在相同機器上執行之應用程式也是。

第1圖說明一微處理器，其中本發明至少一實施例可被使用。特別地，第1圖說明具有一個或多個處理器核心105及110之微處理器100，每個處理器核心105及110分別與一區域快取部107及113相關聯。在第1圖中還說明一共享快取記憶體115，其可儲存至少一些儲存於每個該等區域快取部107及113中的資訊版本。在一些實施例中，微處理器100還可包括其他未在第1圖中顯示的邏輯元件，諸如一整合記憶體控制器、整合圖形控制器，還有其他邏輯元件，以執行一電腦系統內的其他功能，諸如I/O(輸入/輸出)控制。在一實施例中，一多處理器系統中的每個微處理器或一多核心處理器中的每個處理器核心可包括邏輯元件119或與之相關聯，以致能運算資源分配技術，依據至少一實施例。該邏輯元件可包括電路、軟體(包含於一有形媒體中)或兩者，以比一些習知的實施態樣在多個核心或處理器間達到更有效率的資源分配。

例如，第2圖說明一前端匯流排(FSB)電腦系統，其中本發明的一實施例可被使用。任一處理器201、205、210或215可從該等處理器核心223、227、233、237、243、247、253、257其中之一內或與之相關聯的任一區域一階(L1)快取記憶體220、225、230、235、240、245、250、255存取資訊。此外，任一處理器201、205、210或215可從任一共享二階(L2)快取部203、207、213、217存取資訊或經由晶片組265從系統記憶體260存取資訊。依據至少一實施例，第2圖中的一個或多個該等處理器可包括邏輯元件219或與之相關聯以致能資源分配技術。

除在第2圖中說明之該FSB電腦系統之外，其他系統架構結合本發明之各種實施例也可被使用，包括點對點(P2P)互連系統及環狀互連系統。例如，第3圖之該P2P系統，可包括數個處理器，其中僅兩個處理器370、380以範例顯示。每個處理器370、380可包括與記憶體32、34連接的一區域記憶體控制中樞(MCH)372、382。處理器370、380可經由一點對點(PtP)介面350使用PtP介面電路378、388交換資料。每個處理器370、380可經由各自PtP介面352、354使用點對點介面電路376、394、386、398與一晶片組390交換資料。晶片組390還可經由一高性能圖形介面339與一高性能圖形電路338交換資料。本發明之實施例可設定在任意具有任意數目的處理器核心的處理器中，或設定在第3圖的每個PtP匯流排代理器中。在一實施例中，任一處理器核心可包括一區域快取記憶體(未顯示)或與之相關聯。此外，一共享

快取部(未顯示)可被包括於兩個處理器中的任一處理器外，經由p2p互連體與該等處理器相連接，使得若一處理器被置於一低功率模式，一個或兩個處理器的區域快取資訊可儲存於該共享快取部內。依據至少一實施例，第3圖中的一個或多個該等處理器或核心可包括邏輯元件319或與之相關聯以致能資源分配技術。

依據本發明至少一實施例，第4a圖說明一資源管理型快取部的概念。在一實施例中，該快取部400a邏輯上被劃分為兩部分401a及405a，每部分被分成三個區塊410a-412a，對應三個不同的執行緒。在一實施例中，每個分區中的該三區塊被控制響應對應一特定資料類別的每個執行緒之需求變化，且是透過使用一調節器(「T」)來控制。在一實施例中，一調節器代表被給定最高優先順序以存取該共享資源的一資料類別的時間比例。在一實施例中，給定該組調節器下，偏袒高優先順序存取勝於低優先順序存取的資源分配可確保共享資源的最佳分配。在一實施例中，調節器420a及425a對應在該快取部之兩個邏輯部分中有效的調節器，且藉由+Δ與-Δ在每部分中分別被遞增或遞減，使得一初始調節器(T_0)+Δ的資源分配將高於T_0-Δ的資源分配。在第4a圖中，說明給定一組調節器下，三個執行緒(T_0,T_1,T_2)的快取容量。

在第4b圖中，依據一實施例，給定一資源分配測試的說明。在一實施例中，第4b圖說明針對該快取部400a的兩部分401a與405a的兩性能量測，其中該調節器(在此例中，

T_1)在一第一方向(在此例中，正向)針對第一分區401a儲存於區塊411a中的該資料類別，其中該調節器(T_1)在一第二方向(在此例中，負向)移動是針對第二分區405a儲存於區塊411a中的該資料類別。依據各種性能量測技術(包括使用性能計數器)，產生的性能度量接著被決定(其結果顯示於第4b圖中)。此等資源分配測試可持續運行，適於程式中的相變、執行緒遷移/調換等，以連續最佳化該調節器，且因此最佳化分配給一個特定的資料類別或多個特定的資料類別的該快取容量大小。例如，在一實施例中，儲存於快取部400a內的每個資料類別410a-412a對應一多重執行緒程式中的不同執行緒。

第4c圖說明依據至少一實施例可被使用的邏輯元件。第4c圖中的邏輯元件400c說明一處理核心401c、介面連接該核心至一互連體的一介面邏輯元件405c及一快取部410c，該快取部410c可以是一包含式末階快取部，L2快取部等。在一實施例中，依據一連續基礎上的測試結果，GPA模組415c用以執行針對該快取部410c的上述資源分配測試，及將快取部容量分配給該等資料類別(及對應的執行緒)。在一實施例中，該介面邏輯元件405c可以不出現或可包括在該核心或其他邏輯元件中。

在一實施例中，每個針對該快取部的測試結果被發送至該GPA模組，該GPA模組決定如何調整該等快取部分配以最佳化一全域度量，諸如快取失敗率。此外，每次該快取部將對一特定的資料類別分配快取部中的一額外快取部區

塊大小時，該快取部可向該GPA模組請求建議(信號「建議」)，該GPA模組將以一優先順序(信號「優先順序」)回覆。例如，由該GPA模組的一「高優先順序」的指示可使該區塊大小增加，而來自該GPA模組的一「低優先順序」的指示可使該快取部不增加一區塊大小或以一較低程度增加該區塊大小或設定替換位元，因此使得一區塊或部分接下來被替換。

為指示及維持合適的資源分配資訊，該GPA模組可包括每個資料類別之狀態位元及橫跨多個資料類別之全域位元。例如，針對每個資料類別，該GPA模組可儲存總計38位元的狀態，其包括：

18-bits -- 對應一第一性能計數器
(「referencecounter」)

12-bits -- 對應一第二性能計數器
(「randomWalkCounter」)

8-bits -- 對應調節器控制。

此外，四個全域暫存器(總計24位元)可用以參數化最佳化演算法，其包括：

4-bits -- 對應分配給每個資料類別的不同快取區
(「numGradientRegionsLog」)

8-bits -- 對應一第一調節器Δ
(「gradientDelta」)

8-bits -- 對應一第二調節器Δ
(「updateDelta」)

4-bits -- 對應到敏感度，藉由該敏感度該GPA模組將響應可能的性能增益randomWalkThresholdMultiplier。

在一實施例中，該GPA模組使用此等位元尋找共享資源之最佳化分配，使用該梯度下降(或「登山」)演算法的一變形，針對每個資料類別，實施一實驗，假定此資料類別對於討論中的資源遞增之後遞減。針對兩種情況，GPA可接著使用這些位元去量測一「全域良度」度量，且依據該實驗，將該資源之標稱定額分配給該討論中的資料類別。在一實施例中，GPA使用丘諾夫界限來決定何時調整該資源之分區，而在其他實施例中，其他演算法也可用以決定何時調整資源之分區。

在一實施例中，其中上述GPA模組位元用以實施一狀態機，狀態轉換可依據以下程式碼範例描述：

```
void GRADIENT_PARTITION_ALGORITHM_CLASS::Reference(
    bool _hitEqualsTrue,
    UINT32 _setIndex )
{
    UINT32 setIndex = SetIndexHash( _setIndex );
    UINT32 gradientRegionIndex = setIndex & ( ( 1 <<
numGradientRegionsLog ) - 1 );
    setIndex = setIndex >> numGradientRegionsLog;
    bool setMap;
    for( UINT32 i = gradientRegionIndex; i < numDataClasses; i = i + ( 1
<< numGradientRegionsLog ) )
    {
```

```
        setMap = XOR( setIndex & ( ( i >> numGradientRegionsLog ) +
1 ) ); // +Δ 或 -Δ
        data[ i ].referenceCounter++;
        data[ i ].randomWalk += ( _hitEqualsTrue == setMap ) ? 1 : -1;
        if( ( data[ i ].referenceCounter * randomWalkThresholdMultiplier )
<
            data[ i ].randomWalk * data[ i ].randomWalk )
        { //越過動態臨界值了嗎？如果是，在勝利的方向移
動該調節器。
            data[ i ].throttle += ( data[ i ].randomWalk > 0 ) ? updateDelta :
-updateDelta;
            data[ i ].throttle = ( data[ i ].throttle > throttleMask ) ?
throttleMask : data[ i ].throttle;
            data[ i ].throttle = ( data[ i ].throttle < 0 ) ? 0 : data[ i ].throttle;
            Reset( i ); // 針對第 i 個資料類別重置 referenceCounter
及 randomWalk
        }
        else if( data[ i ].referenceCounter >= maxReferenceCount )
        { //嵌入過程，當從先前的實驗中保存一些傾向時，
其被重置
            data[ i ].referenceCounter = data[ i ].referenceCounter >> 2;
            data[ i ].randomWalk = data[ i ].randomWalk >> 1;
        }
    }
}

bool GRADIENT_PARTITION_ALGORITHM_CLASS::InstallAdvice(
    UINT32 _dataClass,
    UINT32 _setIndex )
{
    UINT32 setIndex = SetIndexHash( _setIndex );
```

```
    UINT32 gradientRegionIndex = setIndex & ( ( 1 <<
numGradientRegionsLog ) - 1 );
    setIndex = setIndex >> numGradientRegionsLog;
    bool setMap = XOR( setIndex & ( ( _dataClass >>
numGradientRegionsLog ) + 1 ) );
    INT32 randomThrottle = ( INT32 ) ( rand( ) & throttleMask );
    if( gradientRegionIndex == ( _dataClass & ( ( 1 <<
numGradientRegionsLog ) - 1 ) ) )
    {  //在一梯度區中
        return randomThrottle <= ( data[ _dataClass ].throttle +
                                    ( ( setMap == true ) ? gradientDelta :
-gradientDelta ) );
    }
    else
    {
        return clock <= data[ _dataClass ].throttle;
    }
}
```

第5圖說明一操作流程圖，其可用以與本發明至少一實施例相結合，不管該實施例中使用的處理器或系統架構。在操作501中，一快取部被劃分為至少兩部分，依據一實施例，以執行資源分配實驗。在操作505中，每部分被劃分為對應到被最佳化的特定資料類別的區塊(例如，串流化資料及重複使用資料)。在一實施例中，每個快取分區中的每個資料類別分區對應一不同的執行緒。在操作510中，依據該等上述實施例，執行針對每個快取分區的該資源分配測試。在操作515中，在一程式執行的各個時點上，該快取部

將資訊傳遞至一GPA模組，該GPA模組可持續量測性能且為該快取部提供最佳化參數以提高性能。

至少一實施例中的一個或多個層面可藉由儲存在一機器可讀媒體上的表示資料來實施，其表現出該處理器內各種邏輯元件，當被一機器讀取時會使得該機器來產生邏輯以執行本文描述的該等技術。此等表示資料，已知為「IP核心」，可被儲存於一有形的、機器可讀媒體(「帶子」)上，且供給各種客戶或製造設備以載入到實際製造該邏輯元件或處理器的該等製造機中。

因此，針對管理微架構記憶體區存取的一方法及裝置已被描述。應了解，上述描述欲為說明性的而非限制性的。在熟於此技者閱讀及理解該上述描述後，許多其他實施例將顯而易見。因此，本發明之範圍應該參考後附申請專利範圍及等效於此等申請專利範圍所享有的權利的全範圍來決定。

【圖式簡單說明】

第1圖說明一微處理器的一方塊圖，其中本發明至少一實施例可被使用；

第2圖說明一共享匯流排電腦系統的一方塊圖，其中本發明至少一實施例可被使用；

第3圖說明一點對點互連電腦系統的一方塊圖，其中本發明至少一實施例可被使用；

第4A-4C圖說明邏輯元件的一方塊圖，其中本發明至少一實施例可被實施；

第5圖是一操作流程圖，其可用以執行本發明至少一實施例。

【主要元件符號說明】

32、34...記憶體

100...微處理器

105、110、223、227、237、243、247、253、257、401c...處理器核心

107、113...區域快取部

115...共享快取記憶體

119、219、400c...邏輯元件

201、205、210、215、370、380...處理器

203、207、213、217...共享二階快取部(L2)

220、225、230、235、240、245、250、255...區域一階(L1)快取記憶體

260...系統記憶體

265、390...晶片組

338...高性能圖形

339...高性能圖形介面

350...點對點(PtP)介面

352、354...獨立PtP介面

376、378、386、388、394、398...PtP介面電路

400a、410c...快取部

401a、405a...快取部400a的兩部分

401a...第一分區

405a...第二分區

405c...介面邏輯元件

410a、411a、412a...區塊/資料類別

415c...GPA模組

501、505、510、515...操作

七、申請專利範圍：

1. 一種用以控制運算資源分配之裝置，其包含：

一梯度分區演算法(GPA)模組，用以控制至少一處理資源對至少一指令執行緒之分配，其中被分配給該至少一執行緒之各執行緒的該處理資源的各部分對應於多個資料類別中之一者並且被劃分成為至少兩個分區，以及該GPA模組在對應於該至少一執行緒之各者的該經劃分處理資源上執行測試，用以藉由在經測試的對應分區間之性能度量差異來識別一梯度方向並且用以在該經識別之梯度方向上調整可較佳滿足該至少一指令執行緒之處理需求的一動態調節器設定。

2. 如申請專利範圍第1項所述之裝置，其中該處理資源包括一快取記憶體。

3. 如申請專利範圍第2項所述之裝置，其中該快取記憶體係邏輯上被分割成至少兩部分，該GPA模組可對該至少兩部分執行該等測試。

4. 如申請專利範圍第3項所述之裝置，其中該等邏輯上被分割的兩部分中的每部分包含對應於該至少一指令執行緒的至少一區塊。

5. 如申請專利範圍第1項所述之裝置，其中該GPA模組係用以依據被設定於一第一設定與第二設定的該調節器設定為函數來測試該至少一處理資源之性能，其中該第一設定相較於該第二設定使該處理資源於一較低的位準執行。

6. 如申請專利範圍第5項所述之裝置，其中該GPA模組係用以為該處理資源提供資訊以指示在藉由該GPA模組所使用以測試該處理資源的性能之該等調節器設定中可導致該處理資源具有最高性能位準的該調節器設定。

7. 如申請專利範圍第4項所述之裝置，其中該GPA模組包括對應於該至少一指令的類別資料位元。

8. 如申請專利範圍第7項所述之裝置，其中該GPA模組包括對應於多個指令的全域資料位元。

9. 一種用以控制運算資源分配之系統，其包含：

一處理器核心；

耦接至該處理器核心的一快取部，使得被分配給任何個別執行緒之該快取部的一部分被劃分成為至少兩個分區；

一梯度分區演算法(GPA)模組，用以依據此劃分來對該快取部執行測試，以藉由在該快取部的該部分之經測試的對應分區間之性能度量差異來識別一梯度方向並且在該經識別之梯度方向上調整一動態調節器設定。

10. 如申請專利範圍第9項所述之系統，其中該快取部係邏輯上被分割成至少兩部分，該GPA模組可對該至少兩部分執行該等測試。

11. 如申請專利範圍第10項所述之系統，其中該等邏輯上被分割的兩部分中的每部分包含對應於該等多個指令的多個區塊。

12. 如申請專利範圍第11項所述之系統，其中該等多個區塊

係取決於該調節器的值，而在大小上增加或減少。

13. 如申請專利範圍第12項所述之系統，其中當該區塊大小的該等至少兩部分中之一部分藉由該調節器增加時，該等測試包含量測對應於一第一區塊的一第一執行緒的性能。

14. 如申請專利範圍第13項所述之系統，其中當該區塊大小的該等至少兩部分中之一第二部分藉由該調節器減少時，該等測試也包含量測對應於該第一區塊的該第一執行緒的性能。

15. 如申請專利範圍第14項所述之系統，其中該GPA模組係用以依據該等測試中達到的該性能來設定該第一區塊之大小。

16. 如申請專利範圍第15項所述之系統，其中該GPA模組包括對應於該等多個指令中之每一個的類別資料位元。

17. 如申請專利範圍第16項所述之系統，其中該GPA模組包括對應於所有該等多個指令的全域資料位元。

18. 一種用以控制運算資源分配之方法，其包含以下步驟：

將一快取部劃分為至少兩部分，在該至少兩部分中係用以執行資源分配測試；

將該等至少兩部分的每部分劃分為對應於多個資料類別的多個對應區塊；

對該等至少兩部分的每部分之該等區塊對應於一個資料類別持續執行該等測試且互相比較對應性能結果以識別一梯度方向，其中係以一第一調節器設定對該

等至少兩部分中之一部分執行該測試，且以一第二調節器設定對該等至少兩部分中之一第二部分執行該測試；以及

響應於該等性能結果之比較，在該經識別之梯度方向上調整該等多個區塊中的至少一個區塊的大小。

19. 如申請專利範圍第18項所述之方法，其中該等多個區塊係依據該調節器的設定而在大小上增加或減少。

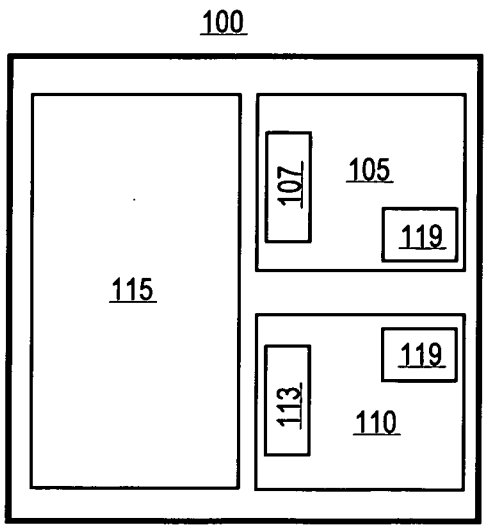
20. 如申請專利範圍第19項所述之方法，其中當該區塊大小的該等至少兩部分中之一部分藉由該調節器設定增加時，該等測試包含量測對應於一第一區塊的一第一執行緒的性能。

21. 如申請專利範圍第20項所述之方法，其中當該區塊大小的該等至少兩部分中之該第二部分藉由該調節器減少時，該等測試也包含量測對應於該第一區塊的該第一執行緒的性能。

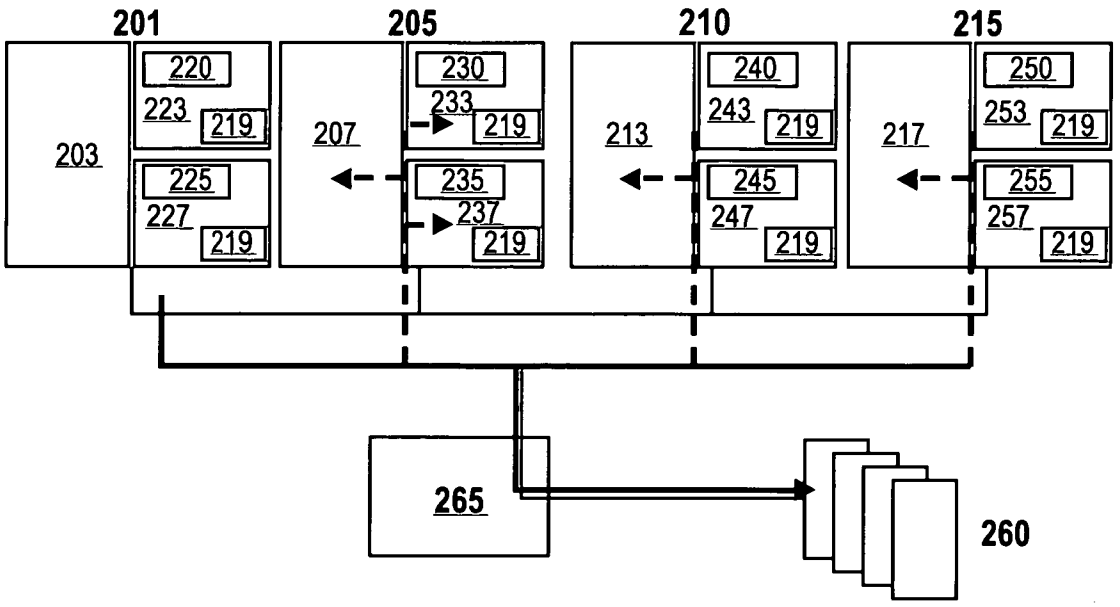
22. 如申請專利範圍第21項所述之方法，其包含依據在該等測試中達到的該性能設定一第一區塊大小。

23. 如申請專利範圍第22項所述之方法，其中對應於該等多個資料類別中每個類別的類別資料位元被維持。

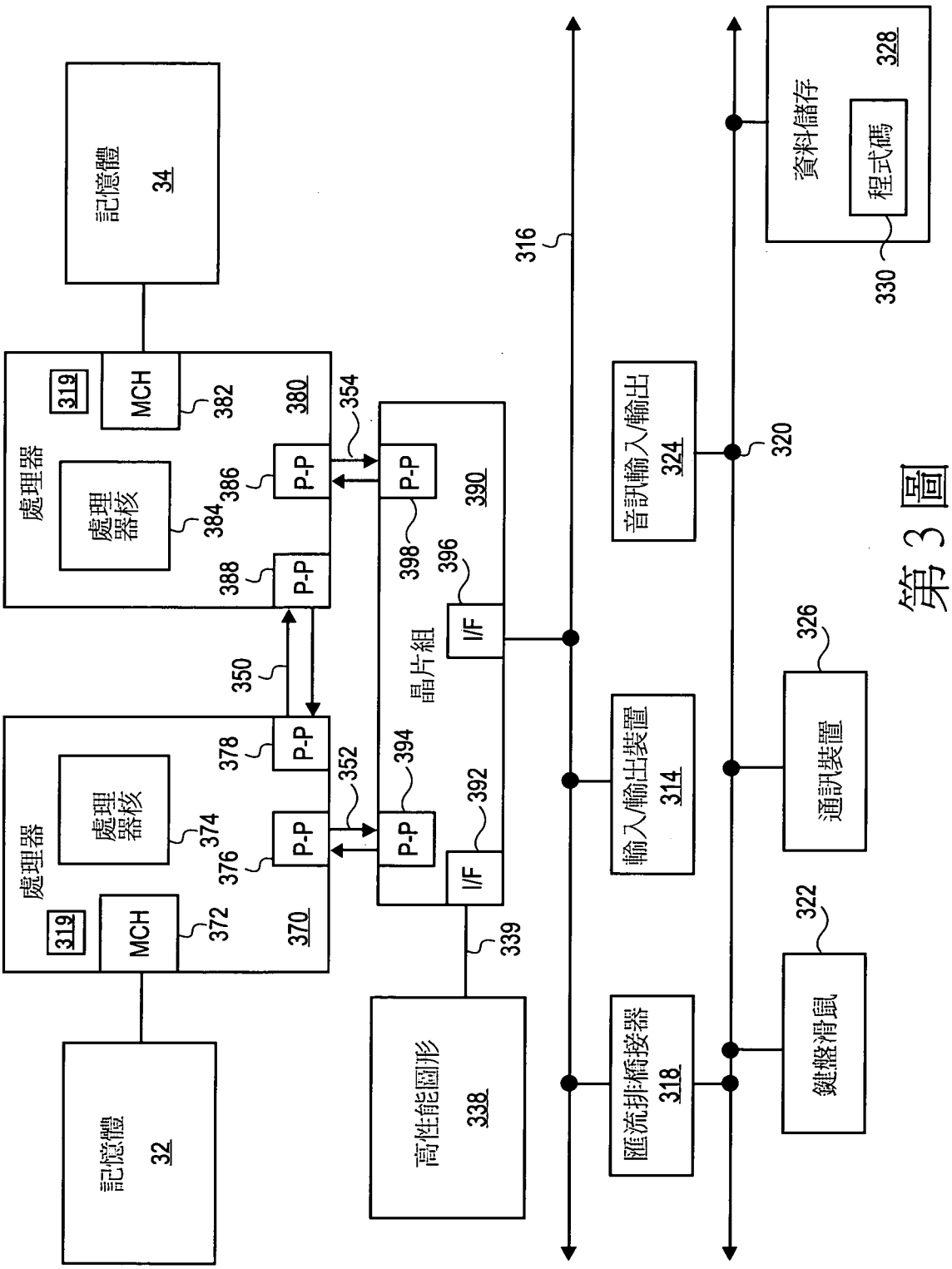
24. 如申請專利範圍第23項所述之方法，其中對應於所有該等多個資料類別的全域資料位元被維持。



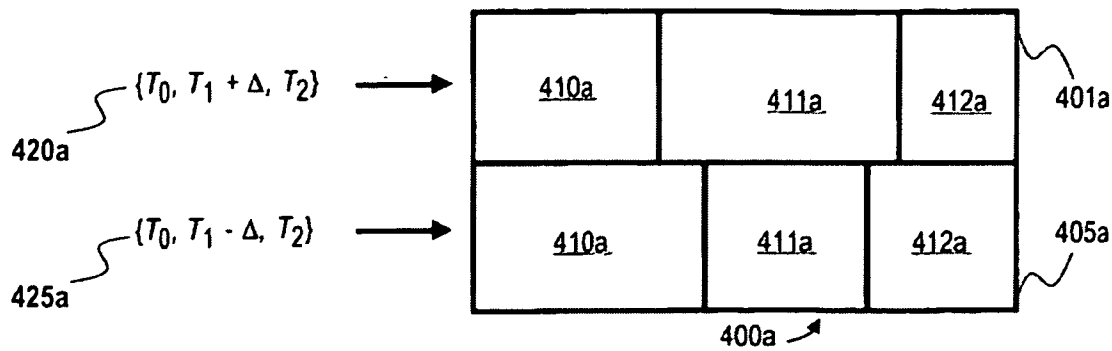
第 1 圖



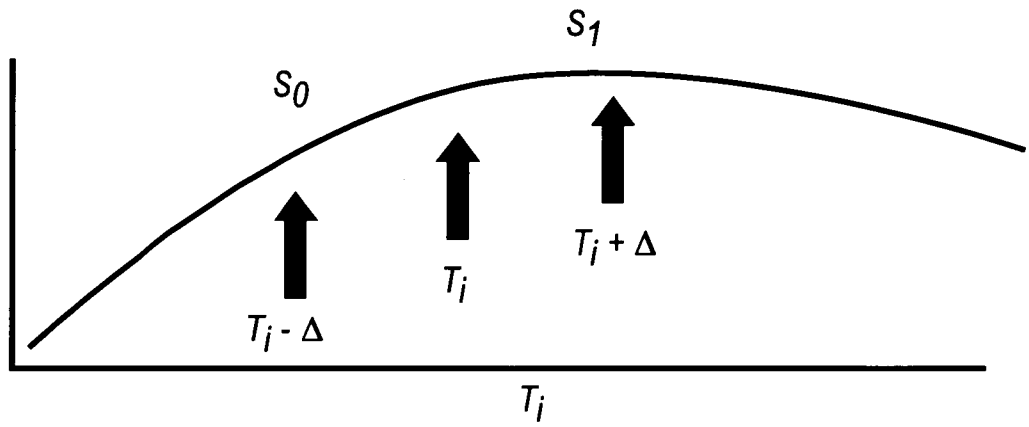
第 2 圖


記憶體
34
處理器
319
MCH
382
380
處理器核
384
386
P-P
388
P-P
354
350
352
378
P-P
處理器核
374
376
P-P
處理器
319
MCH
372
370
記憶體
32
晶片組
P-P
398
396
I/F
390
P-P
394
392
I/F
339
高性能圖形
338
316
音訊輸入/輸出
324
320
輸入/輸出裝置
314
匯流排橋接器
318
資料儲存
328
程式碼
330
326
通訊裝置
322
鍵盤滑鼠

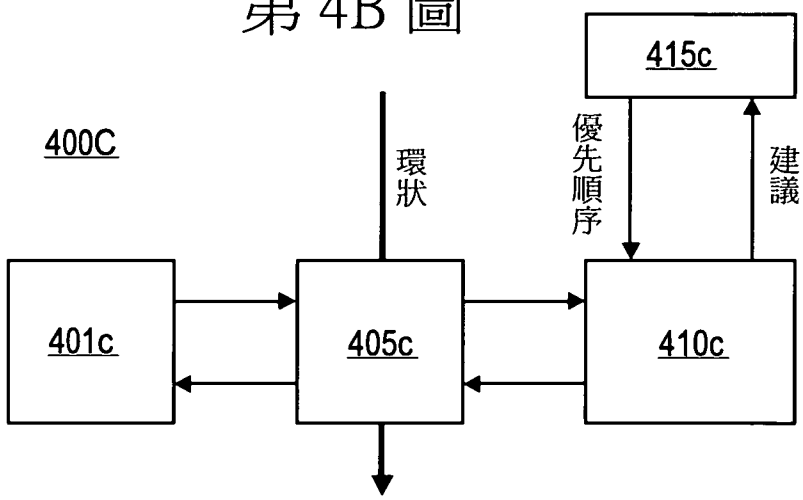
第3圖



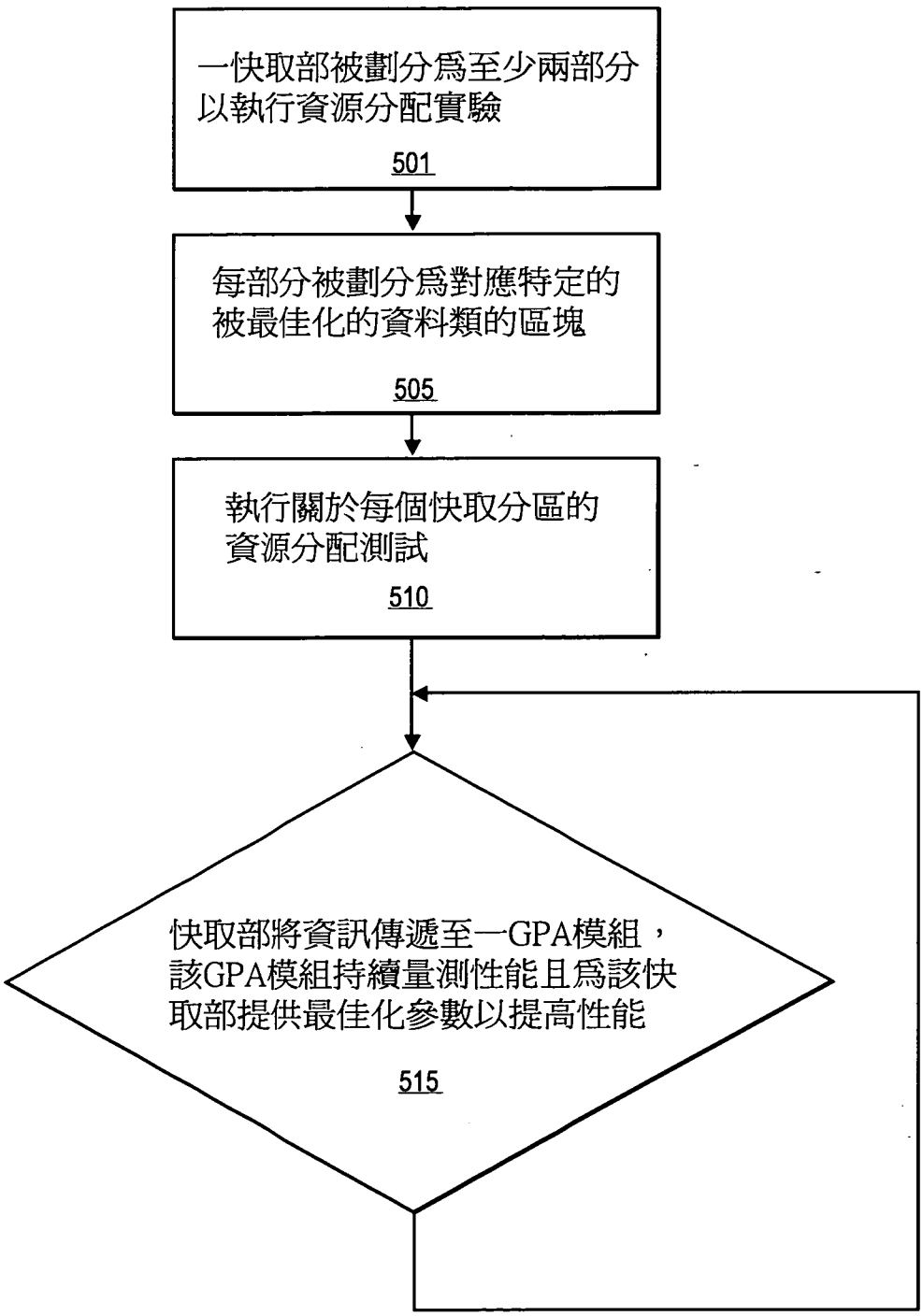
第 4A 圖



第 4B 圖



第 4C 圖


一快取部被劃分爲至少兩部分以執行資源分配實驗
501
每部分被劃分爲對應特定的被最佳化的資料類的區塊
505
執行關於每個快取分區的資源分配測試
510
快取部將資訊傳遞至一GPA模組，該GPA模組持續量測性能且爲該快取部提供最佳化參數以提高性能
515

第 5 圖