

發明專利說明書

(本說明書格式、順序及粗體字，請勿任意更動，※記號部分請勿填寫)

※ 申請案號：94106055

※ 申請日期：94 年 3 月 1 日

※IPC 分類：G06F 9/46 (2006.01)

一、發明名稱：(中文/英文)

在多節點電腦系統中啟動作業系統資源配置的機制

Mechanism for Enabling the Distribution of Operating System
Resources in a Multi-Node Computer System

二、申請人：(共壹人)

姓名或名稱：(中文/英文)(簽章)

萬國商業機器公司

INTERNATIONAL BUSINESS MACHINES CORPORATION

代表人：(中文/英文)(簽章)

傑羅 羅森梭

Gerald Rosenthal

住居所或營業所地址：(中文/英文)

美國紐約州 10504 亞芒克市新奧爾察德路

New Orchard Road, Armonk, NY 10504, U.S.A.

國籍：(中文/英文)

美國

US

三、發明人：(共六人)

姓名(中文/英文)，國籍(中文/英文)

凌 基特 張，Chung, Lynn Keat，美國 US

克里斯多弗 弗朗科斯，Francois, Christopher，美國 US

理察 卡爾 科克蒙，Kirkman, Richard Karl，美國 US

派崔克 喬瑟夫 麥克卡斯，McCarthy, Patrick Joseph，美國 US

道恩 達瑞 李德，Reed, Don Darrell，美國 US

肯尼士 查爾斯 佛森，Vossen, Kenneth Charles，美國 US

四、聲明事項：

☐ 主張專利法第二十二條第二項 ☐ 第一款或 ☐ 第二款規定之事實，其事實發生日期為： 年 月 日。

☒ 申請前已向下列國家（地區）申請專利：

【格式請依：受理國家（地區）、申請日、申請案號 順序註記】

☒ 有主張專利法第二十七條第一項國際優先權：

1. 美國 US、西元 2004 年 3 月 04 日、10/793,476

☐ 無主張專利法第二十七條第一項國際優先權：

☐ 主張專利法第二十九條第一項國內優先權：

【格式請依：申請日、申請案號 順序註記】

☐ 主張專利法第三十條生物材料：

☐ 須寄存生物材料者：

國內生物材料 【格式請依：寄存機構、日期、號碼 順序註記】

國外生物材料 【格式請依：寄存國家、機構、日期、號碼 順序註記】

☐ 不須寄存生物材料者：

所屬技術領域中具有通常知識者易於獲得時，不須寄存。

九、發明說明：

【發明所屬之技術領域】

本發明相關於資源管理，尤其是關於在多節點電腦系統中的資源管理。

【先前技術】

電腦系統廣泛被用來儲存和操作資料。資料儲存於電腦系統記憶體中，且由在電腦系統的處理器上執行的電腦系統程式所操作。如眾所習知，處理器常常被認為是電腦系統的「大腦」，因為它是電腦系統內執行電腦系統之程式的元件，係允許電腦系統進行實際工作。記憶體被用來保存正在被執行的電腦程式，且保存資料以供正執行電腦程式的處理器所存取。

為了有競爭力，電腦系統的設計者不斷努力讓電腦系統更強大並維持或減少電腦系統大小。常見的方法為藉由增加所用處理器的數目來增加電腦系統的整體處理能力。為了生產效率，處理器和記憶體常常被封裝在一起以形成所謂的節點，且電腦系統由一或更多個節點所組成。在這些多節點電腦系統內，任何處理器可以存取任何節點上的記憶體，但處理器一般存取本身節點上的記憶體（本地端存取）比存取任何其他節點上的記憶體（遠端存取）更有效率。

電腦系統包括一系列由電腦系統的一或更多個處理器所執行的指令。藉由執行這些指令，處理器被稱為在執行電腦程式。作業系統（主要負責為其他程式的利益操作

電腦系統的程式) 透過工作 (有時稱為任務或程序) 的使用, 來控制這些程式的執行。大多數處理器只能夠一次執行一個指令串流, 但因為他們運作很快, 所以他們看起來似乎執行許多工作且同時服務許多使用者。電腦作業系統給予每一工作執行的「回合 (turn)」, 且然後要求該工作當另一工作獲得回合時要等待。在工作需要等待一些事情發生才能繼續 (例如, 存取次級儲存) 或多重處理器可用的情況, 工作可以建立執行緒 (有時稱為子程序或子任務) 來非同步地繼續或加速處理。未建立任何執行緒的工作本身可以被視為有單一執行緒。因此, 工作可以被稱為由一或更多個執行緒所組成。

從節點的觀點, 作業系統可以分派執行緒以任何數目的方式執行。舉例來說, 一個工作的執行緒可以被選取在特定節點上執行, 而另一工作的執行緒可以被選取在不同節點上執行。類似地, 來自相同工作的執行緒可執行於不同節點上, 且選取在特定節點上執行的執行緒也可在結束前, 被選取在一或更多個其它節點上執行。儘管此彈性在一些方面為有利的, 但從資料存取觀點是有問題的。如上面所述, 節點由處理器和記憶體所組成, 且處理器可以比在另一節點上更有效率地存取本身節點上的記憶體。因此, 為了有效率地執行, 作業系統必須保證每一個執行緒於其正在執行之同樣節點上的記憶體中存取其資料。

作業系統解決此問題的一種方式為藉由關聯每一執行緒至對其有偏好執行和存取資料的節點。然後, 當到了執行特定執行緒的時候, 作業系統盡可能選取在其偏好節

點上的處理器。類似地，當資料需要代表該執行緒被帶入記憶體時，其偏好節點上的記憶體會盡可能被選取。此方法一般而言對最小化遠端記憶體存取有幫助，假設由該執行中的執行緒所做的工作平衡遍佈於電腦系統的節點。

具有一或更多個節點的電腦系統也可被分隔為二或更多個邏輯上分開的系統。邏輯分隔可能被分派處理器和記憶體，而不管其所屬的節點（們）。更進一步來說，處理器且／或記憶體可以因為組態變動或容量升級或降級，被動態地增加至或移除自分隔且／或電腦系統。關於電腦系統內本地端對比遠端記憶體存取的效率問題也適用於每一邏輯分隔內。在本文的描述當中，術語「系統」被用來指一個完整未分隔的電腦系統，或是電腦系統的邏輯分隔。

作業系統所使用的節點平衡的一種方式包括機制，以依照每個節點上可供系統使用的處理器及／或資源的相同分例，來分派工作（執行緒）給較佳節點。

雖然這些機制在平衡工作負載上是有同的，而且可以概略地將遠端存取最小化，但是習知的作業系統本身並未對其提供的相關服務，進行遠端存取的最小化。舉例來說，考慮一包括有整合式檔案系統的作業系統，其提供各種檔案服務給客戶端執行緒。與此檔案系統相關的各種資源需要執行其功能。在這些資源之中，可能會有一組核心執行緒（僅為作業系統所知與管理）在佇列上等待提出要求的客戶端所提的信息，以指示他們非同步地處理檔案的特

定部份。其它可能的資源為信息集合 (pool of messages)，其可被傳送至這些執行緒。除非這些信息集合與佇列剛好常駐在客戶端執行緒所偏好節點的記憶體上，且除非對這些請求提供服務的核心執行緒也偏好相同的節點，否則在客戶端執行緒與提供服務的核心執行緒之間，會很有可能牽涉到遠端記憶體存取。其它許多的例子可以用來描述相同的概念。

在沒有方法使作業系統服務以節點為基礎來分佈其資源，並使得與客戶端執行緒的偏好節點相關的特定資源被用來服務此執行緒之請求的情況下，作業系統的使用會因為遠端存取而變得沒有效率。此外，除非是以系統上工作負載的相同比例，並依節點來平衡資源的分佈，否則資源將不會被平均地使用，並且在特定的資源上仍會產生各種其它無效率的結果。

【發明內容】

本發明係有關於一種裝置、方法和程式產品，用期望的系統工作負載的相同比例，以節點為基礎來達成作業系統資源的分佈。在系統開始時，當系統的處理與記憶體資源以節點為基礎而被識別後，資源的初始分佈會被建立。而在處理器及／或記憶體被加入系統或由系統中被移除時，資源可被重新分佈以配合新期望的工作平衡。隨後，在實際的工作負載逐漸地轉移至其新的節點比例後，便可逐漸地對資源進行重新分佈。

在本發明的較佳實施例中，藉由分派各種不同種類的

權重給每個節點，以代表他們在系統中整體平衡的比例。目標權重代表期望的工作負載分佈，其係基於每個節點上處理器與記憶體資源的現存比例。目標權重在系統被開始之初就會被建立，並且會保持固定，除非後來有處理器及／或記憶體被加入或被移除。系統上實際的工作負載則是由目前權重來代表，而作業系統將其持續地保持儘可能接近目標權重。當系統開始時，作業系統服務是以目標權重相同的比例，來依節點(nodally)而分佈他們的資源，並且可以要求在目標權重改變時被通知。若處理器及／或記憶體在後續被加入或移除，則此時會計算新的目標權重，而且有提出請求的所有服務都會被通知，因此他們可以依照新的目標權重或其逐步的改進，來重新分佈他們的資源。

【實施方式】

現在參照圖式，圖 1A 顯示一些本發明較佳實施例之計算裝置中所使用的運作元件。電腦裝置 100 為增強的 IBM iSeries 電腦系統，然而其他電腦也可被使用。被描述的元件包括處理器節點 130、網路介面 125、使用者介面 150 和大量儲存裝置 155。較佳實施例的電腦裝置 100 包括四個處理器節點，然而更多或更少的節點都可被使用。網路介面 125 被用來與其它電腦系統進行通訊。使用者介面 150 被用來接受命令，且轉遞(relay)資訊至電腦系統 100 的一或更多個使用者。

節點 130 其中之一的展開圖也被顯示（即，節點 131）。節點 131 包含處理器 132 和記憶體 134。在較佳實施例中，每一節點可以包括多達八個處理器 132，然而

在其他組態中也可包括更多或更少的處理器。處理器 132 也被用來執行儲存於記憶體 134 中的執行緒，然而應該了解的是，在有些時刻，這些執行緒可能部份或全部位於大量儲存裝置 155 中。雖然處理器存取本身節點上的記憶體是較有效率的，但處理器仍然可以存取任何節點上的記憶體。

圖 1B 顯示較佳實施例之計算環境之更詳細的示意圖。如圖所示，較佳實施例能夠運作於邏輯分隔的電腦系統。在此情況下，核心服務 137 之實例(instance)被設定位於每一分隔（於圖 1B 顯示為系統 1 至 3）。較佳實施例的機制將每一分隔視為完整的系統。因此，「系統」這一個術語在此單獨使用時，應該被解釋為一個未分隔電腦裝置，或是電腦裝置內的單一邏輯分隔。系統可以只使用整個電腦裝置的部份節點，例如系統 1 和 2 之情況。而一個節點也可能支援一個以上的系統，例如節點 1 和 2 的情況。較佳實施例的許多機制涉及分派和調整工作（執行緒）如何被分佈於系統的節點之間。因此，當系統組態改變時，較佳實施例的機制決定是否需要對工作的分佈進行調整，且如果需要，則進行這些調整。

每一系統被顯示包括執行緒 135 和作業系統 136。核心服務 137 被整合至作業系統 136。核心服務 137 提供作業系統等級的服務給其他作業系統個體(entities)和其他程式。不過應該注意的是，儘管較佳實施例對重新平衡的需要提供核心服務通知，但本發明不被限制於核心或作業系統等級的服務，但反而能適用於能夠利用重新平衡通知的

所有服務。

作為最後前言內容，應該被了解的是儘管本發明的實施例於此被描述於在電腦裝置的背景中，然而一些程式機制，例如核心服務 137，能夠以程式產品的形式被散佈。當然，程式產品可以使用其他類型的信號承載媒體來散佈，包括但不限於：可記錄型媒體，例如軟碟和光碟；和傳輸型媒體例如數位和類比通訊連結。

圖 2 為較佳實施例之核心服務 137 中一些方面的展開視圖。如圖所示，核心服務 137 包括執行緒管理者 200、節點管理者 220 和組態管理者 240。執行緒管理者 200 更進一步被顯示包括執行緒建立機制 (Thread Creation Mechanism, TCM) 205、執行緒分配機制 (Thread Dispatch Mechanism, TDM) 215 和執行緒記憶體管理者 (Thread Memory Manager, TMM) 217。TCM 205 在較佳實施例中被用來建立可執行的執行緒以及必要的執行緒控制結構。做為該建立程序的一部份，TCM 205 也分派主節點給這些可執行執行緒。主節點的分派是重要的，因為若可能則 TDM 215 係偏向(biased)以執行其被分派的主節點上的執行緒，且若可能則 TMM 217 係偏向以配置其主節點的記憶體。此偏向係意指相關於整體系統利用和資源配置。TCM 205 的主節點分派方面被描述圖式 26 至 28 伴隨的文字中。

在執行緒被建立且由 TCM 205 分派至主節點後，他們由 TDM 215 進行分發來執行。一般來說，執行緒可以

被執行於系統之各個節點上許多處理器的其中任一之上。TCM 205 為較佳實施例內的實例，係負責將執行緒分組以對可用的處理器和系統資源做最佳利用。關於 TCM 205 的功能和內部工作的額外細節被提供於下面圖 3A 到 5 的討論中。

圖 2 上也顯示有節點管理者 220。節點管理者 220 係由三個離散的個體所組成，被用於較佳實施例中來決定系統中各種節點的相對工作負載。節點管理者 220 額外地被用於較佳實施例中來識別和更正系統內的工作負載失衡，並通知其他核心服務節點工作負載的改變，使得這些服務可以自己相應地調整。失衡識別和處理被描述於本專利的工作重分佈章節中（圖 11 至 14 和 16 至 25），其中核心元件平衡通知被描述於圖 7 至 11、14 和 15 所伴隨的文字中。

顯示於節點管理者 220 內的是分發處理者(Dispatch Handler)230、監視處理者 (WatchDog Handler, WDH) 225 和節點平衡者 235。這三個實例運作於較佳實施例的系統內做為非同步執行緒。習知本技術者認知這些執行緒的非同步本質隱含基本的 (atomic) 資料存取的需要。因此儘管於此偶爾參照到基本性議題，本專利並不包含底層基本性機制的廣泛探討，因為其已如同上述為熟此技術者所習知。如同其名稱所暗示，分發處理者 230 負責處理來自 TDM 215 的分發要求。因為執行緒不斷地被分發，分發處理者 230 相當頻繁地執行。在此背景下，其責任之一為以主節點為基礎累加處理器執行週期計數（分發處理者 230 用來獲得此週期資訊的確切方式對於理解此較佳實施

例來說並不是很重要，因而並不於此詳述)。WDH 225 係排定每秒執行，並每七秒收取這些累加週期。此七秒期間於此稱為目前權重期間 (Current Weight Interval)。在收取該資訊後，WDH 225 對此資訊進行各種計算，最終獲得目前權重。(較佳實施例內目前權重的使用被解釋於下面段落中)。在一些時候，WDH 225 也負責在每一次目前權重期間終了時，通知節點平衡者 235。節點平衡者 235 利用此通知來決定是否需要核心元件平衡通知。

也顯示在圖 2 上的是組態管理者 240。組態管理者 240 是較佳實施例內的實例，係發出組態變更信號給節點平衡者 235。節點平衡者 235 然後考慮該組態變更，且相對於工作分佈和核心元件平衡通知，採取適當的動作。組態變更的範例包括增加或減少系統上處理器或記憶體數目。

基於記憶體親和性之執行緒分組

為了允許多於一個執行緒對共享資料的本地端存取之目的，本發明提供將兩個或更多個執行緒分組為記憶體親和性群組 (Memory Affinity Group, MAG)。藉由具有相互依賴資料之存取需要的執行緒，允許其在相同節點上執行，因而達到本地端記憶體存取的效能增加。在較佳實施例中，執行緒個別地或以工作為基礎被放進 MAG 中。若以工作為基礎被放進 MAG 中，特定工作的所有執行緒全部被放置於相同的 MAG 中。然而習知本技術者應認知存在其它方法。舉例來說，可以在兩個等級都載明 MAG 參

與，而其中一個載明優先於另一個。

圖 3A 至 5 於此被用來描述本發明一方面的記憶體親和性分組。在較佳實施例中，MAG 決定和邏輯節點分派係發生於正在執行的執行緒要求建立另外一個執行緒的時候。圖 3A 於此被用來說明 TCM 205，其負責與發出要求之執行緒直接進行互動的部分。此互動的結果最終導致執行緒建立，係被描述於圖 3B 所關聯的文字中。首先看到圖 3A，方塊 300 顯示接收到要建立執行緒之要求。接下來，在方塊 302 中，TCM 205 決定是否執行緒建立要求載明是否期望參與 MAG。在較佳實施例中，期望 MAG 參與的需求係載明於 TCM 205 所存取的持續執行緒建立屬性 (Thread Creation Attributes, TCA)。若此執行緒期望 MAG 參與，TCM 205 接下來決定是否 TCA 載明特定的 MAG [方塊 304]。在較佳實施例中，此係透過使用 MAG 識別符 (即群組 ID) 來載明。若群組 ID 在 TCA 內被載明，TCM 205 將允許該要求，且單純地建立具有該載明之群組 ID 的執行緒 [方塊 308]。若 TCA 尚未有載明的群組 ID，其將會被分派下一個可用的群組 ID (且隨後此群組將由 TCA 所用) [方塊 306]。

若方塊 302 中 TCM 205 決定未被要求參與 MAG，TCM 205 接下來將決定是否 TCA 中已載明使用「邏輯節點」[方塊 310]。在較佳實施例中，邏輯節點不同於實體節點，其係用來容許發出要求的執行緒，載明被建立的執行緒會被分派相同或不同的主節點。實體節點係為執行緒的實際主節點，則不被發出要求的執行緒所察覺。此方法

具有允許發出要求者載明邏輯節點，而又保留使用具有不同識別符的實體節點來允許要求的彈性。因此，邏輯節點的載明具有載明執行緒主節點的效果。

若在方塊 310 中邏輯節點的使用已經被載明，TCM 205 決定是否 TCA 包括特定邏輯節點的載明〔方塊 312〕。若沒有，TCM 205 分派下一個可用的邏輯節點 ID，且儲存它於 TCA 中〔方塊 314〕。最終，TCM 205 使用 TCA 中所載明的邏輯節點 ID，或是新分派的邏輯節點 ID，以繼續建立執行緒〔方塊 316〕。現在回到方塊 310，若 TCM 205 藉由檢視 TCA 來決定邏輯節點的使用未被載明，TCM 205 將在方塊 318 中決定是否 TCA 包括參考執行緒的識別符。參考執行緒為對於將被建立的執行緒來說，相對應具有記憶體親和性的一執行緒。在較佳實施例中，參考執行緒的載明藉由包括執行緒 ID 於 TCA 中來達成。若參考執行緒 ID 未被載明，TCM 205 建立沒有邏輯節點或 MAG ID 的執行緒〔方塊 320〕。此將有允許 TCM 205 為被要求的執行緒選擇主節點的效果。

若參考執行緒被載明，TCM 205 將接下來決定是否被載明的參考執行緒已經被關聯於 MAG〔方塊 322〕。如果如此，TCM 205 將提取參考執行緒的 MAG ID〔方塊 324〕，且使用被載明的 MAG ID 繼續建立執行緒〔步驟 326〕。若方塊 322 中 TCM 205 決定參考執行緒還未被關聯至 MAG，TCM 205 決定是否參考執行緒已被分派有邏輯節點。如果如此，TCM 205 提取參考執行緒的邏輯節點〔方塊 330〕，且使用被載明的邏輯節點建立執行緒〔方塊 332〕。

〕。若方塊 328 中 TCM 205 決定邏輯節點事實上尚未被分派至參考執行緒，TCM 205 將分派下一個可用的 MAG ID〔方塊 334〕，且從參考執行緒中提取主節點〔方塊 336〕。一旦這被達成，新的 MAG 被建立且被加至 MAG 列表，參考執行緒也被加至 MAG〔方塊 338〕。TCM 205 然後使用被載明的 MAG ID 建立執行緒〔方塊 340〕。

圖 3B 顯示較佳實施例之 TCM 205 中所用來實際實現執行緒建立的步驟。有三個進入點到圖 3B 的邏輯。執行緒的建立可以用 MAG ID 當作輸入〔方塊 354〕、用邏輯節點 ID 當作輸入〔方塊 352〕、或沒有載明 MAG ID 或邏輯節點 ID〔方塊 350〕。

若 TCM 205 透過方塊 350 進入圖 3B 的邏輯，TCM 205 將繼續決定執行緒的主節點〔方塊 356〕。如同稍早所提及，執行緒的主節點為執行緒將會盡可能被分發到的節點，且其記憶體將會盡可能被配置給執行緒的行為。在決定主節點時，系統會考慮數個因素，包括目前和期望的工作負載。關於主節點決定的詳情被提供在圖 26 至 28 所關聯的文字中。TCM 205 然後繼續為執行緒建立核心執行緒結構〔方塊 372〕。在較佳實施例中，核心執行緒結構被用來代表執行緒。因此，建立核心執行緒結構同義於建立執行緒。TCM 205 接下來決定是否最近建立的執行緒為 MAG 的成員。這部分解釋的現行假設為執行緒建立要求並未包括 MAG 或邏輯節點 ID 的載明（見上面方塊 350 的討論）。因此，方塊 380 和 384 的處理中產生否（NO）的決定，係致使 TCM 205 在方塊 388 中結束處理。在此

時點被要求的執行緒已經被建立且被分派至主節點。

若 TCM 205 透過方塊 352 進入圖 3B 的邏輯，TCM 205 繼續將載明的邏輯節點轉換為實體節點，其即為執行緒的主節點。在較佳實施例中，決定性的演算法被用來維護特定邏輯節點和特定實體節點間一致的關聯，而這個對應可以是多對一的對應。然而應該注意的是，其他對應方法亦可以被使用。方塊 372 和 380 的處理是相同於上面所已經描述。不過在此情形中，邏輯節點 ID 已經被載明，係導致 TCM 205 標記執行緒為不可被移動的〔方塊 386〕，意指其主節點不可被變更（見下面的工作重分佈的章節）。發出要求之執行緒所載明的特定邏輯節點係據此以達成。

若 TCM 205 透過方塊 354 進入圖 3B 的邏輯，TCM 205 將首先嘗試找到關聯於群組 ID 的 MAG〔方塊 360〕。在較佳實施例中，每一 MAG 由群組物件所代表，且每一群組物件包含群組 ID、MAG 的主節點、以及成員執行緒的列表。物件被鏈結在一起以形成 MAG 的列表。TCM 205 藉由訪遍 MAG 的列表決定是否群組 ID 可找到匹配〔方塊 362〕。若匹配被找到〔方塊 362〕，TCM 205 從群組物件中提取 MAG 的主節點〔方塊 364〕，其係在方塊 372 中建立核心執行緒結構之前。

現在回到處理方塊 362，若 MAG 的列表中沒有匹配被找到，TCM 205 需要為將被建立的執行緒和 MAG 決定主節點。TCM 205 繼續在方塊 370 中決定此執行緒的主

節點。(如所提及，主節點決定更詳細被解釋在圖 26 至 28 所關聯的文字中)。一旦主節點被決定，TCM 205 建立新的 MAG (即，群組物件)，且儲存主節點和群組 ID 於 MAG 中[方塊 378]，且增加群組物件至列表中[方塊 374]。當新的 MAG 被加入至列表中時，執行緒則接著被建立 (即，核心執行緒結構)[方塊 372]。

如上面所描述，一但核心執行緒結構被建立，TCM 205 決定是否剛建立的執行緒為 MAG 成員。在此部份的解釋中，執行緒為 MAG 的成員。因此，TCM 205 在方塊 382 中繼續增加執行緒至 MAG。在較佳實施例中，此藉由增加執行緒至 MAG 的執行緒列表中而達成。

圖 4 顯示用於較佳實施例中，當執行緒終結時處理 MAG 管理之強調的步驟。在較佳實施例中，當執行緒終結時，若其為 MAG 的成員，則有責任將自己從 MAG 移除。因此，若在方塊 405 中終結的執行緒決定其不是 MAG 的一部分，正常執行緒終結處理則會繼續[方塊 417]。若在方塊 405 中執行緒決定其為 MAG 的成員，則執行緒繼續將自己從 MAG 的執行緒列表中移除[方塊 410]。若終結中的執行緒不是 MAG 中的最後一個執行緒[方塊 419]，終結程序再次繼續。不過，若終結中的執行緒是 MAG 中的最後一個執行緒[方塊 419]，在方塊 412 中 MAG 從 MAG 列表中被移除。然後在正常終結程序於方塊 417 中繼續之前，MAG 本身在方塊 423 中被摧毀。

MAG 管理的另一方面為呈現關於 MAG 的資訊於

MAG 之存在相當短暫的環境中（即，MAG 來去的環境）。圖 5 顯示用於較佳實施例中來達成此任務的步驟。當資訊要求被接收〔方塊 500〕，首先做一個嘗試來找到該資訊所找尋的 MAG〔方塊 505〕。若關聯的 MAG 在 MAG 列表中未被找到，在方塊 535 中發出通知給要求者，且在方塊 530 中繼續正常處理的步驟。若 MAG 被找到，被要求的資訊則在方塊 520 中提供。處理然後繼續〔方塊 530〕。

核心元件平衡通知

本發明的此章節係關於核心元件平衡通知。如上面所描述，較佳實施例的機制運作以將系統上的整體工作負載進行最佳化的平衡。與此平衡相關聯的是要通知其他核心元件（即，服務）有關工作分佈之改變，使得這些元件可以採取適當步驟的需求。為接收通知，元件必須與節點平衡者 235 進行登記。節點平衡者 235 可以存取元件登錄，係顯示於圖 31 中。用於較佳實施例中的登錄，其邏輯上看起來為鏈結列表（linked-list），其每一元件包含元件名稱、元件方法、和指向登錄中下一元素的指標。元件方法被設立在通知的時刻由節點平衡者 235 所呼叫。應該注意的是，儘管較佳實施例以作業系統服務的核心服務而來說明本發明的運作，然而本發明的應用不限於核心或作業系統服務，而可應用於能從系統內工作分佈改變的通知受益的任何類型的元件或服務。

系統起始程式讀取（IPL）

最初應該注意的是，儘管在系統 IPL 章節中數個較佳實施例概念被介紹於此，許多這些概念的完整解釋（即，目前權重、轉換權重、權重差異門檻值和起始權重）直到整體被討論時才被提供。目前權重(Current Weight)、轉換權重(Transition Weight)、權重差異門檻值(Weight Differential Threshold)和起始權重(Initiation Weight)的概念被討論在圖 7 至 9 所關聯的文字中，而起始權重的概念則連同圖 26 至 28 的說明中討論。採取此方式是因為透過下面段落所提供的背景可以達到較佳的理解。

現在參考圖 6，節點平衡者 235 在每一次當系統開機時被呼叫，其係可能發生於任何時刻。節點平衡者 235 將節點管理者狀態（見圖 29A 的 2902）設定為「目前權重」〔方塊 605〕，且將所有儲存區域初始化為零（見圖 29A 至 31 的 2904 至 3100）〔方塊 608〕。節點平衡者 235 繼續建立一組權重。這些權重係以主節點為基礎而被建立，並被用於較佳實施例中來管理系統上工作的分配。第一個被建立的權重為所謂的目標權重。每一節點的目標權重代表該特定節點期望的工作份量。較佳實施例之機制的目標為依照各種目標權重的比例分配工作至整個系統。圖 6 的方塊 610 參照到圖 10A，係描述目標權重的計算。圖 10A 在本發明的組態變更章節被詳細解釋。

從圖 10A 的邏輯返回後，起始和目前權重 2930 和 2906 被設定為目標權重〔方塊 625〕。對每一節點，其目前和轉換權重被設定為相等於在目前 KCBN 集合（見圖 30 的 3015）中的目標權重〔方塊 635〕。轉換權重、目前

權重和目標權重的交互作用被詳細解釋在圖 7 至 9 所關聯的文字中。起始權重係關於主節點分派，其係被描述於伴隨圖式 26 至 28 的文字中。所有上面的處理必須發生在剩下的 IPL 處理前〔方塊 640〕，其係包括執行緒的分發和由作業系統 136 所啟動之主節點移動。此外，其可包括基於目標權重之作業系統／核心資源的節點分配。IPL 處理結束於方塊 645。

組態變更

如將於隨後的段落被更詳細地描述，目標權重係以節點為基礎，由系統可利用的處理器和記憶體所驅動。在特定系統啟動期間，係針對每一系統建立目標權重。其不會發生改變，除非／直到系統中發生組態變更時才會改變。在此讀者則是需要假設組態變更已經發生。此變更由圖 7 的方塊 700 所代表。變更的詳情對較佳實施例的解釋而言不是很重要，但如此變更的範例可以是系統的重新組態，以將節點利用三個處理器改變為二個。當節點平衡者 235 接收組態變更的通知時〔方塊 700〕，節點平衡者 235 繼續獲得目前權重的複本〔方塊 710〕。

目前權重代表系統內實際的工作分佈。較佳實施例中用來獲得目前權重的機制被顯示於圖 10C 和 10D。圖 10C 中所採取的第一個步驟是目前權重識別符（Current Weight Identifier, CWID）的提取〔方塊 1048〕，接著為目前權重期間編號（Current Weight Interval Number, CWIN）的提取〔方塊 1049〕。目前權重僅在一個期間內有效，且在那之後就到期，而必須基於最近的期間重新計

算。CWID 2911 包含目前權重為有效的期間編號。若 CWID 2911 等於 CWIN 2908，則目前權重為有效；若否，其已經過期且必須被重新計算。簡短地參考圖 29A，關於目前權重處理的資訊被顯示。連同 CWID 2911 和 CWIN 2908，目前權重呼叫計數 (Current Weight Invocation Count, CWIC) 2910 和集合 0 (SET0) 和集合 1 (SET1) 被顯示。每一集合包括分發者許諾百分比累加器 (Dispatcher Commitment Percentage Accumulators, DCPA) 和最近執行週期計數 (Recently Executed Cycle Counts, RECC)。CWIC 2910 被用來決定何時 CWI 已經過期。DCPA 資訊相關於較佳實施例之工作重新分配和主節點分派方面，而因此被描述於那部份的討論。圖 29A 上也被顯示的是目前權重 2906 和 DCP 2907。目前權重 2906 被用於較佳實施例的此一方面 (現在正描述的 KCBN 邏輯)，而 DCP 2907 被用於較佳實施例的工作重新分配和主節點分派方面。

現在回到圖 10C，在方塊 1049 中提取 CWIN 2908 之後，節點平衡者 235 首先決定是否 CWID 2911 等於 CWIN 2908。若其相等，節點平衡者 235 繼續至圖 10D 的方塊 1061 (此處理在隨後的段落中被描述)。若其不相等，節點平衡者 235 則測試 CWIN 2908，以決定是否其為奇數或偶數 [方塊 1050]。若為奇數，RECC [方塊 1051] 和 DCPA [方塊 1054] 則從集合 1 中被提取 (見圖 29A)，但若 CWIN 2908 為偶數，RECC [方塊 1052] 和 DCPA [方塊 1056] 則從集合 0 中被提取。如上所述，分發處理者 230 執行非常頻繁且以主節點為基礎累積處理器執

行週期當作其功能之一部分。這些週期計數和 DCP 則由監視處理者 225 所收取，且以節點為基礎在每個 CWI 的結束時儲存在 RECC 和 DCPA 中。監視處理者 225 也會增加 CWIN，其係將集合於活動中和非活動中之間切換 (flip flops)，而反之亦然（此被詳細描述於圖 11 和 14 的討論中）。節點平衡者 235 接著再次提取 CWIN 2908 [方塊 1058]，且決定從先前提取以來，它是否已經已改變 [方塊 1062]。基本性 (atomicity) 會是一個議題，因為當目前 RECC 和 DCPA 正在被提取時，CWI 可能已經過期。因此，這些步驟是必要的，以確保 RECC 和 DCPA 資訊從活動中的集合被提取。若 CWIN 2908 已經改變，上述程序重複（見圖 10D 的方塊 1062 和 1063），直到節點平衡者 235 決定 CWIN 2908 未被改變為止。

現在移到圖 10D，當節點平衡者 235 決定 CWIN 2908 從先前提取以來未被改變 [方塊 1062]，則節點平衡器 235 藉由將每一節點的 RECC 除以全部的 RECC，來計算每一節點的目前權重 [方塊 1064]。此產生由每一節點實際正在進行之目前工作的比例，且因此，其代表每一主節點承受之整體工作負載的實際份量。目前權重然後儲存於目前權重集合 2906 且 DCP 儲存於圖 29 的 DCP 集合 2907。一旦目前權重被計算且儲存，CWID 2911 被設定等於被提取的 CWIN。目前權重然後從目前權重集合 2906 被提取，以被傳回至圖 10C 和 10D 之邏輯的呼叫者 [方塊 1066]。在方塊 1070 中 DCP 也從 DCP 2907 中被提取，其再次是為了將 DCP 傳回給圖 10C 與 10D 之邏輯的呼叫者。最後，在方塊 1072 中 CWIN 2908 再次被提取，

且在方塊 1074 中用先前提取的版本測試。若兩個值不同，處理藉由在圖 10C 的方塊 1057 中繼續而重複。

現在回到圖 7，節點平衡者 235 將接下來決定新的目標權重〔方塊 715〕。當然，此處的假設為目標權重因為組態改變而改變。在圖 10A 的第一步驟中〔方塊 1002〕，系統內每一節點處理器的數目和記憶體的数量被決定。在較佳實施例中，此資訊在系統啟動期間被儲存於基於節點的資料結構中（見圖 30 的 3015 和 3020）。讀者要回想較佳實施例中的節點可以有零或更多個處理器。橫跨系統之處理器的全部數目和記憶體的全部數量在方塊 1004 中被決定。一旦節點的總數被決定，目標權重的非活動中集合將被識別（即，若 TWMN(Target Weight Modification Number) 2920 為奇數則是集合 0，若 TWMN 2920 為偶數則是集合 1）。非活動中集合將會被用來計算新的一組目標權重（即當它們改變時）。然後當 TWMN 2920 被增加時，係將有改變活動中的集合的效果。

一旦目標權重的非活動中集合已經被識別，節點平衡者 235 接下來決定是否系統是設定為專屬或共享處理器。若系統為未分隔電腦裝置，處理器依照定義為專屬的（如圖 1A 中），係因為沒有其他系統存在，若有則他們可能為共享的。然而，為邏輯分隔的系統可能使用共享或是專屬的處理器。簡短地參照到圖 1B，節點橫跨多重分隔未必表示此分隔在節點上共享處理器。特定節點上的處理器可以在多於一個分隔間被共享，或此分隔可能由一個或更多專屬處理器所支援。具有專屬處理器的分隔，其係獨

占地使用這些處理器，直到例如其被通知組態變更時。具有共享處理器的分隔，則動態地被授與使用電腦裝置上任何可用處理器而沒有如此的通知。系統之處理器的本質由組態管理者 240 使其可用於節點平衡者 235。若系統使用共享處理器，在非活動中集合中每一節點的目標權重，則成為等於節點上對於系統可用的記憶體，對上橫跨所有節點對於系統可用的記憶體總量的比例〔方塊 1006〕。TWMN 2920 然後在方塊 1020 中被增加，係有改變非活動中目標權重集合為活動中目標權重集合的效果。

若系統未使用共享處理器〔方塊 1008〕，節點平衡者 235 決定是否系統有任何具有處理器和記憶體兩者的節點〔方塊 1012〕。一個否的決定（即，沒有節點具有處理器和記憶體兩者），雖然可能但並不是典型的。舉例來說，可能將較佳實施例之電腦系統 100 的分隔設定為，具有一些節點有處理器而沒有記憶體，以及具有其他節點有記憶體而沒有處理器。若情況確實如此，節點平衡者 235 將非活動中的目標權重集合中每一節點的目標權重，設定為等於節點上處理器對上橫跨所有節點對系統可用之處理器之全部數量的比例，並將沒有處理器的節點的目標權重設為零〔方塊 1016〕。TWMN 2920 然後在方塊 1020 中被增加，其係（再次）有改變非活動中目標權重集合為活動中目標權重集合的效果〔方塊 1020〕。

若在方塊 1012 中，節點平衡者 235 決定至少一個節點具有至少一個處理器和一些記憶體，節點平衡者 235 將接下來決定位於有至少一個處理器和一些記憶體之節點

上的處理器之整體數目〔方塊 1014〕。目標權重的非活動中集合然後被更新，使得具有至少一個處理器和一些記憶體之節點的目標權重，被設定為其個別處理器對比前面所提及處理器之整體數目的比例〔方塊 1018〕。非活動中集合內沒有處理器也沒有記憶體之節點，其目標權重則被設定為零〔方塊 1022〕。TWMN 2920 然後在方塊 1020 中被增加，其係（再次）有改變非活動中目標權重集合為活動中目標權重集合的效果〔方塊 1020〕。

不管經過圖 10A 的流程圖所走的路徑為何，節點平衡者 235 將最終地複製目標權重、處理器數目和記憶體數量至每一節點的節點錨(Node Anchor)結構〔方塊 1024〕。如將在隨後段落中所解釋，做此以使有意義的核心元件平衡通知成為可能。圖 30 顯示用於較佳實施例中的節點錨結構。每一節點錨結構包括識別符〔3005〕、目前權重、處理器和記憶體〔3015〕和先前權重、處理器和記憶體〔3020〕。也顯示在節點錨結構 3000 的為一系列資源錨。這些錨被已登錄元件用來儲藏資源或指向資源的指標，係由已登錄元件以節點為基礎基於遍及系統之工作分佈所配置。後者觀念將在隨後段落更詳細被解釋。

現在回到圖 7，節點平衡者 235 接著計算目前權重和目標權重間的權重差異〔方塊 730〕。該差異為每一節點差異之絕對值的總合。節點平衡者 235 的處理在圖 8 的方塊 800 中繼續進行。權重差異門檻值 (Weight Differential Threshold, WDT) 被設定為剛計算出之差異的 3/4。圖 29B 顯示 WDT 2924。WDT 為適當的因為認知到實際工作分

佈可能與新期望的工作分佈相當不同。儘管系統將動作以調整目前工作分佈（如目前權重所示）來接近目標權重，而如此可能花費相當的時間。因此，已登錄核心元件僅根據新目標權重來重新調整其資源可能是不適當的。因此，WDT 連同所謂的轉換權重被用來提供已登錄元件更漸進地調整其資源的機會。如將被討論的，轉換權重被設定在目前權重和目標權重之間，以提供已登錄元件之中間的平衡點。WDT 被用於較佳實施例中來決定目前和目標權重間的差異何時足夠大，以批准轉換權重的使用。如將會在接踵而至的討論中所見，較佳實施例的 WDT 的值，可以大到權重差異的 $3/4$ ，以及小到 $1/64$ (.015625)。

在設定 WDT 2924 為差異的 $3/4$ 後，節點平衡者 235 決定是否差異少於 $1/32$ [方塊 805]。若否，節點平衡者 235 從差異減去 $1/32$ ，且決定是否結果少於 WDT 2924。如果結果少於 WDT 2924，則 WDT 2924 被重設為比差異少 $1/32$ 。不管方塊 805 中測試的結果為何，節點平衡者 235 最終地決定是否 WDT 2924 為 $1/64$ 或更少[方塊 820]。若 WDT 2924 為 $1/64$ 或更少，轉換權重的使用不被批准，所以 WDT 2924 被設為零且目前集合 (3015) 中的 KCBN 轉換權重被設為等於目標權重。若節點平衡者 235 在方塊 820 中決定 WDT 2924 大於或等於 $1/64$ ，則節點平衡者 235 將目前集合 (3015) 中的 KCBN 轉換權重設定為目前權重和 (新) 目標權重之間的值。在較佳實施例中，此係根據下列公式來達成。

$$\text{轉換} = \text{目標} + (\text{目前} - \text{目標}) * \text{WDT} / \text{權重差異}$$

節點平衡者 235 的處理然後繼續到圖 9 的方塊 900。在此，KCBN 目前權重的目前集合 (3015) 被設定為目前權重。每一已登錄元件 (3105) 所載明的核心元件方法 (見圖 31) 接替被呼叫 [方塊 905]。

被呼叫之後 (見圖 9 的方塊 905)，核心元件方法存取儲存於每一節點上的 KCBN 資訊。目前、轉換和目標權重連同這些權重之先前值的複本，可在被呼叫的方法中使用。目前和先前處理器數目以及記憶體數量也可在被呼叫的方法中使用。全部的這些資訊允許被呼叫的方法來決定如何最佳利用其資源。能夠利用上述通知服務之核心元件的範例為整合檔案系統 (IFS)。檔案系統典型地利用事先被建立之執行緒的集合來處理不同功能。舉例來說，IFS 可有一組執行緒進行硬碟機 (HDD) 的讀取，以及另一組執行緒進行 HDD 的寫入。為了最佳利用這些執行緒，IFS 會使用較佳實施例以將執行緒的最佳分佈方式與工作分佈一致。在被呼叫時，關聯於 IFS 的核心元件方法則透過考慮 KCBN 權重來決定工作正在從第一節點被移動到第二節點，此方法將摧毀一些或全部在第一節點上其事先建立的執行緒，並建立一些於第二節點上。在 KCBN 資訊的目前集合已經被複製到 KCBN 資訊的先前集合後，核心元件平衡通知之組態變動方面的處理然後結束。

重新計算轉換權重

此專利的先前章節相關於當組態變更發生時，在較佳

實施例中轉換權重如何被計算和設定。有了先前的敘述，本專利的此章節則相關於轉換權重如何被週期性地被重新計算，以透過週期性通知而提供漸進的重新平衡。圖 11 至 14 顯示用於較佳實施例的監視處理者 225 中的步驟。如同上面所提及，較佳實施例的監視處理者 225 每秒執行一次，並且其有相關於較佳實施例之各種方面的功能。然而，為了此特定討論之目的，在此只詳述其中的某些方面（即，轉換權重重新計算）。因此，只有一些圖示和處理方塊被探討，而其他的部份暫時被略過，而會在隨後的段落中討論。在圖 11 中，處理方塊 1105 至 1114 相關於較佳實施例的其他方面。在圖 11 的方塊 1116 中，監視處理者 225 增加目前權重呼叫計數 (CWIC)，且因為監視處理者 225 每秒執行，CWIC 每秒被增加 (CWIC 2910 顯示於圖 29A)。監視處理者 225 接下來決定是否已達到該期間的最大呼叫計數 [方塊 1120]。在較佳實施例中，最大呼叫計數為七。計數到七則發出目前權重期間結束的信號。

圖 14 中顯示已完成之目前權重期間的監視處理者 225 處理。在步驟 1405 中，非活動中集合中之 DCP 累加值被正規化 (見圖 29A 之 DCPA 集合 2912 和 2916)。(決定 DCPA 之中的哪個集合為非活動中，是透過上述檢視 CWIN 2908 來決定，奇數 CWIN 2908 為集合 0 而偶數 CWIN 2908 為集合 1)。在較佳實施例中，此正規化的處理係藉由將累加值除以在該期間中的呼叫之數目 (即，七)。此處應該注意的是圖 14 中所顯示的一些步驟 (即，方塊 1405 和 1435) 並不相關於核心元件平衡通知。然而，

這些內容於此被討論，因為這些步驟在目前權重期間的結束時被進行。

監視處理者 225 接下來提取由分發處理者 230 所收集的目前權重週期計數〔方塊 1410〕。在較佳實施例中，這些計數係基於主節點而被儲存於以處理器為基礎的資料結構（未顯示）。監視處理者 225 然後從剛獲得的週期計數減去在非活動中集合中先前所儲存的週期計數（見圖 29A 的 RECC），以達成每一主節點在到期的 CWI 期間所執行的週期計數。此結果然後被回存在非活動中集合中，取代先前的值〔方塊 1415〕。在方塊 1420 中，來自活動中集合的 RECC 被提取，除以二，且然後被加至在非活動中集合的值（以節點為基礎）。此產生之值在較佳實施例中被視為最近被執行之週期計數。

CWIN 2908 然後在方塊 1425 中被增加，以將非活動中集合（即，具有最新計數的集合）切換至活動中集合。CWIC 2910 然後在方塊 1430 中被設定為零，當作新 CWI 的開始。監視處理者 225 然後將在方塊 1410 中所收集的計數，儲存至新被切換的非活動中集合，以形成這些相同計算的基礎（即，在下一個 CWI 的結束）〔方塊 1440〕。WDT 2924 然後在方塊 1450 中以零做測試。非零 WDT 的集和以及目前權重期間之結束發出需求信號，以決定是否轉換權重需要被重新計算。圖 15 和 8 概述用於較佳實施例之轉換權重的重新計算步驟。首先參考到圖 15，目標權重在方塊 1500 中被獲得。

在較佳實施例中，用來獲得目標權重的機制由圖 10B 所代表。在圖 10B 的方塊 1030 中，目標權重修改編號（Target Weight Modification Number, TWMN）被提取。現在參考至圖 29A，所顯示的為目標權重集合 2922 和目標權重修改編號（TWMN）2920。如同所示，有兩組目標權重用於較佳實施例中（集合 0 和集合 1）。TWMN 2920 被用來決定哪一個集合為活動中集合，以及哪一個集合為非活動中集合。若 TWMN 2920 為奇數，則集合 1 為活動中集合，且集合 0 為非活動中集合。若 TWMN 2920 為偶數，則集合 0 為活動中集合，且集合 1 為非活動中集合。節點平衡者 235 接著測試 TWMN 2920 以決定是否其為奇數或偶數〔方塊 1032〕。若 TWMN 2920 為偶數，每一節點的目標權重從集合 0 中被提取〔方塊 1034〕，但若 TWMN 2920 為奇數，每一節點的目標權重從集合 1 中被提取〔方塊 1036〕。TWMN 2920 然後在方塊 1038 中被再次提取，且在方塊 1040 中測試變更。若 TWMN 2920 沒有改變，先前提取的目標權重集合在方塊 1042 中被傳回。若 TWMN 2920 已經改變，則測試奇／偶、集合提取、和變更測試的步驟被重複，直到 TWMN 2920 被發現沒有改變為止。TWMN 2920 被要求於未改變之狀態是為了基本性(atomicity)的目的。本專利的重新平衡工作和主節點分派章節，概述當 TWMN 2920 可能正在被增加時，節點平衡者 235 正在擷取目標權重值的情況。

現在回到圖 15，在方塊 1505 中獲得目標權重。對於較佳實施例中來獲得目前權重之步驟的細節，請參考至先前圖 10C 之討論。目前權重和目標權重間的權重差異然

後被計算。如同先前所述，差異為每一節點權重差異之絕對值的總和。差異然後用 WDT 2924 測試。若差異大於 WDT 2924，轉換權重不需要被重新計算。然而，若差異小於或等於 WDT 2924，轉換權重使用上述與圖 8 和 9 關聯的處理被重新計算。更多資訊請參考至這些圖式的描述。

主節點分派

短暫地回到圖 2，TCM 205 被用於較佳實施例中來建立可執行的執行緒，連同必要的執行緒控制結構。建立程序的一部分是分派主節點給這些可執行的執行緒。主節點分派是重要的，因為若可能則 TDM 215 係偏向(biased)以執行其被分派的主節點上的執行緒，且若可能則 TMM 217 係偏向以配置其主節點的記憶體。此偏向係意指相關於整體系統利用和資源配置。。在較佳實施例中，分派主節點至執行緒意謂系統將最佳地在主節點上執行執行緒，以及在主節點上分派記憶體。此在這裡被稱為節點親和性。當然，若特定節點被設定只有一個處理器，此親和性變成處理器親和性。此外，習知本技術者將認知本發明不限於由較佳實施例所提供之特定類型的親和性。舉例來說，其它能達到總是在主節點上執行執行緒之要求的緊密親和性，也落在本發明的精神和範圍內。

如上所討論，目標權重被用於較佳實施例中來特徵化整個系統中所期望的工作分佈。目前權重被用來特徵化整個系統中目前的工作分佈。如果每一執行緒做相同份量的

工作，則為新執行緒決定主節點將很直接。即，主節點將僅基於保持分派至每一節點之現存執行緒的數量，與目標權重相同的比例。但現實的情形為不同執行緒做不同份量的工作，且不可能事先知道執行緒將做多少工作。因此，較佳實施例基於目標權重和目前權重兩者建立起始權重，且分派主節點以便保持分派至每一者的執行緒數目為與起始權重相同的比例。目前權重超過目標權重的節點，讓其起始權重被設定為稍微小於其目標權重（即，因為其已經有超過其份量的工作負載），且目前權重低於目標權重的節點，讓其起始權重被設定為稍微大於其目標權重（即，因為目前其有少於其份量的工作負載）。當計算每一節點的起始權重時，系統越忙碌，目前權重被給予越多的影響。

圖 26 顯示主節點分派的上層步驟。TCM 205 在處理方塊 2605 中獲得每一節點的起始權重。圖 27 和 28 概述此程序如何運作於較佳實施例中。現在參考至圖 27，TCM 205 首先在方塊 2705 中提取起始權重識別符（Initiation Weight Identifier, IWID）。圖 29B 顯示 IWID 2928 和起始權重 2930。然後在 TCM 205 在方塊 2715 中決定是否 IWID 符合 CWIN 前，CWIN 2908 在方塊 2710 中被提取。若 IWID 不符合 CWIN，TCM 205 繼續至圖 28 的方塊 2800 以計算新起始權重，因為現存起始權重是過時的。

目前權重連同 CWID 和 DCP 在方塊 2805 中被獲得。這些步驟的解釋請參考至上面圖 10C 的描述。在方塊 2710 中被提取之 CWIN 的值然後被設定為在方塊 2810

中的 CWID，使得其代表與目前權重關聯的期間編號。目標權重然後在方塊 2815 中被獲得。這些步驟的解釋請參考至上面圖 10B 的描述。

開始於方塊 2820 中的第一個節點，TCM 205 現在繼續結合目標和目前權重，以產生每一節點的起始權重。在結合目標和目前權重時，特定節點的 DCP 越高，目前權重對所產生之該節點的起始權重將有越大的影響。

結合權重的第一部分為計算將主導目前權重之影響的乘數。若 DCP 為 75% 或更少 [方塊 2825]，目前權重的影響將被抑制，其係藉由在方塊 2850 中設定乘數等於 DCP 立方除以 75% 的平方。否則若 DCP 少於 100% [方塊 2835]，目前權重的影響將被放大，其係藉由在方塊 2855 中設定乘數等於 100% 減去 ((100% 減去 DCP 的立方) 除以 (25% 的平方))。否則，若 DCP 為 100% 或更大 [方塊 2835]，目前權重的影響將被最大化，其係藉由在方塊 2840 中設定乘數等於 100%。不管所採取的路徑為何，乘數將在零到 100% 的範圍中且被用於下面的第二部分。

結合權重的第二部分為計算除數，容許在當目前權重小於目標權重時，起始權重係大於目標權重，而當目前權重大於目標權重時，起始權重係小於目標權重，依照目前權重被允許有的影響量。TCM 205 在方塊 2860 中計算除數為目標權重加上 ((目前權重減去目標權重) 乘上乘數)。所產生的除數的範圍將在目標權重和目前權重之間

某處（取決於乘數），且被用於第三部分。

第三部分為藉由將目標權重的平方除以除數來計算未正規化的起始權重〔方塊 2875〕。這樣的效果將使得目前權重較小時，讓起始權重較大，而反之亦然。為了避免除以零的可能，TCM 205 檢查除數是否為零〔方塊 2870〕，且如果是，其在方塊 2865 中僅設定起始權重等於目標權重。TCM 205 然後檢查是否有更多節點要處理〔方塊 2890〕，且如果是，其為每一剩下的節點重複上面處理。當 TCM 205 的確決定所有節點已經被考慮，其繼續至圖 27 的方塊 2750。在經由連接方塊 2750 回到圖 27 的邏輯之後，TCM 205 將未正規化的起始權重進行正規化，且儲存他們於起始權重 2930〔方塊 2745〕。在較佳實施例中，起始權重藉由將每一權重除以所有節點的所有權重的總合而被正規化。IWID 然後被設定為方塊 2735 中的 CWID，以載明起始權重為有效的目前權重期間。

目前所描述的執行路徑和來自方塊 2715「是」的分支（見上面描述）在方塊 2725 會合。在該處，TCM 提取起始權重且然後繼續再次提取 CWIN。若新提取的 CWIN 不同於先前提取的 CWIN，於是起始權重為過期的，且先前描述之起始權重計算步驟重新開始於圖 28 的方塊 2805。若新提取的 CWIN 相同於先前提取的 CWIN，被提取的起始權重為最新的，且被傳回至圖 26 的邏輯。

現在回到圖 26 的方塊 2610，TCM 205 決定每一節點目前現存執行緒的數目。在較佳實施例中，核心服務 137

以節點為基礎追蹤現存執行緒，係藉由當執行緒被建立時增加計數器，和當執行緒被摧毀時將其減少。執行緒的整體數目然後藉由加總計數器被決定〔方塊 2615〕。然後在方塊 2620 中，執行緒的目標數目以節點為基礎被決定。在較佳實施例中，此藉由將執行緒的整體數目乘上每一節點的起始權重而達成。實際執行緒和目標執行緒間有最大落差的節點，為被選為剛建立執行緒之主節點的節點〔方塊 2622〕。

工作重新分配

概論

我們在上面討論了本發明的較佳實施例能夠偵測系統內的工作不平衡且做適當調整。在較佳實施例中，這是藉由一系列在監視處理者 225、分發處理者 230 和節點平衡者 235 之間狀態驅動的互動所達成。為了工作重新平衡目的用於較佳實施例中的狀態被命名為觀察平衡 (Observe Balance)、記錄週期 (Record Cycles) 和移動執行緒 (Move Threads)。觀察平衡狀態，係為起始點，由系統管理者所設定。因此，系統管理者讓系統能夠決定是否存在工作平衡問題。當狀態被設為觀察平衡，監視處理者 225 追蹤五十三 (53) 秒的週期，於此被稱為觀察平衡期間 (Observe Balance Interval, OBI)。在一定數目的期間已經經過之後，在較佳實施例的例子為六，監視處理者 225 通知節點平衡者 235。

當節點平衡者 235 醒來時，其注意到狀態被設定為觀察平衡。節點平衡者 235 藉由進行是否系統失去平衡的上層分析來回應。儘管此決定在隨後的段落中被更詳細解釋，此處應該注意的是分發處理者 230 係執行的非常頻繁，正在以主節點為基礎追蹤所執行的週期數目。此原始週期計數被節點平衡者 235 用來做其上層平衡決定。當節點平衡者 235 決定系統工作負載可能失去平衡時，其設定狀態至記錄週期，且相應地通知監視處理者 225。

監視處理者 225 然後認知狀態已經被改變至記錄週期，且在追蹤經過單一記錄週期期間(Record Cycles Interval)後，設定狀態至移動執行緒且通知節點平衡者 235。同時，分發處理者 230 認知狀態已經被節點平衡者 235 設定至記錄週期。分發處理者 230 藉由以執行緒為基礎，和以主節點為基礎兩者收集週期資料來回應。分發處理者 230 當稍後決定狀態已經被改變至記錄週期以外者時，停止以執行緒為基礎收集週期資料。

當節點平衡者 235 接收來自監視處理者 225 的上述通知時，其繼續進行由分發處理者 230 所收集之基於執行緒之資料的下層分析。節點平衡者 235 然後仔細檢查系統的平衡／未平衡狀態，且使用基於執行緒的資料來決定在系統的工作負載沒有被適當地平衡的情況時，何執行緒要移動。若節點平衡者 235 決定系統工作負載沒有被平衡，其繼續在節點間移動執行緒，以更適當地平衡系統的工作負載。最後，節點平衡者 235 設定狀態回到觀察平衡以重新開始處理。

在觀察平衡狀態的分發處理者 230

剛才所提到的是分發處理者 230，當在觀察平衡狀態時，保有以主節點為基礎之所執行之週期之數目的原始計數。此邏輯顯示於圖 16A。顯示於圖 16A 的是分發處理者 230 內相關於本發明之較佳實施例的步驟。習知本技術者認知存在其他未相關之執行緒分發處理。在方塊 1600 中，分發處理者 230 獲得將要被分發之執行緒的主節點 ID。上一次此特定執行緒執行所消耗的週期，然後在方塊 1605 中被獲得。所消耗的週期計數然後以主節點為基礎被加至目前權重週期計數累加器。此計數，係由分發處理者 230 所保有，被儲存於基於處理器的資料結構（未顯示）。分發者 230 然後決定是否其在觀察平衡狀態或更以後。若是，上述所消耗週期被加至觀察平衡週期計數累加器，係為由分發處理者 230 所維護之另一個基於處理器的資料儲存（未顯示）。分發處理者 230 然後決定是否其在記錄週期狀態〔方塊 1645〕。因為此處我們在觀察平衡狀態，分發處理者 230 在方塊 1640 中僅返回，係為剩下的執行緒分發處理發生之處。圖 16B 的邏輯連同記錄週期狀態在下面被描述。

在觀察平衡狀態的監視處理者 225

圖 11 關於核心元件平衡通知在稍早被討論。在上述討論，被提及的是相關於工作重新分配的一些特定處理方塊（即，監視處理者 225）。這些方塊將現在被描述。在

方塊 1105 中，目前分發者貢獻比率被獲得。在較佳實施例中，其系藉由系統介面來獲得，此系統介面對本發明的益處和優點來說並不重要，且因此未於此更進一步被描述。百分比代表以節點為基礎之系統之活動之等級的指示。類似於 RECC，非活動中集合透過 CWIN 2908 的檢視被識別（見圖 29A 的 DCPA 集合 2912 和 2916。）〔方塊 1110〕。若 CWIN 2908 為奇數，DCP 被加至集合 0 中的 DCPA；若 CWIN 2908 為偶數，DCP 被加至集合 1 中的 DCPA。監視處理者 225 然後決定是否 CWIC 2910 已經到達其最大值〔方塊 1120〕。若已經到達最大值，監視處理者 225 繼續處理已完成的目前權重期間。參考至圖 14，此處理已經在先前被描述（見上）。讀者被要求在這裡假定已完成之目前權重期間的處理不在討論中。因此，監視處理者 225 將決定是否目前狀態是觀察平衡或更以後〔方塊 1122〕。

此處目前狀態是觀察平衡，所以監視處理者 225 移動至圖 12 的方塊 1205。於該處，監視處理者 225 計算至最舊的儲存區域（見圖 29B，儲存區域 SA0-SAN，2932-2936）的索引。在較佳實施例中，此計算涉及提取圖 29B 的觀察期間編號（Observe Interval Number, OIN）2950，且將 OIN 2950 除以儲存區域的數目（即， $N+1$ ），且然後使用產生的餘數當作索引。監視處理者 225 然後以節點為基礎把 DCP 加至最舊之儲存區域的 DCPA〔方塊 1210〕。最舊之儲存區域的呼叫計數（IC）然後被增加〔方塊 1215〕。監視處理者 225 然後決定是否最大呼叫計數已經達到〔方塊 1220〕。若是如此，觀察期間已經被完成，

係需要處理。此處理被顯示在圖 17 中。若觀察期間尚未完成，處理繼續於圖 13。

首先看到圖 17，監視處理者 225 首先從分發處理者 230 獲得觀察平衡週期計數，且以節點為基礎儲存它們在第二最舊的儲存區域〔方塊 1700〕。(再次，見圖 29B，儲存區域 SA0-SAN，2932-2936。)這些值將最終變為在下一個觀察期間所被執行的週期數目的基礎。然後，在方塊 1705，監視處理者 225 將計算在剛完成的觀察期間所執行的週期。在較佳實施例中，其係藉由從在方塊 1700 中獲得的計數，減去在為剛完成觀察期間的基礎之最舊之儲存區域的週期計數。此計算的結果然後回存至最舊的儲存區域。在方塊 1710 中，監視處理者 225 設定第二最舊之儲存區域之 DCPA 為零。做此以準備下一個觀察期間的累加。監視處理者 225 然後設定第二最舊之儲存區域的呼叫計數為零，再次以準備下一個觀察期間的累加〔方塊 1715〕。OIN 2950 然後被增加，係有基本性地改變最舊的儲存區域為最新的和第二舊的儲存區域為最舊的之效果〔方塊 1720〕。

在方塊 1725，監視處理者 225 決定是否 OIN 2950 已經到達觀察平衡門檻值。此值被用於較佳實施例中，以決定何時節點平衡者 235 應該檢查從工作負載的觀點系統的節點是否平衡。在較佳實施例中，此值為六(6)。若觀察平衡門檻值尚未到達，在方塊 1740 中監視處理者 225 回到圖 12 的邏輯。若觀察平衡門檻值已經到達，監視處理者 225 決定是否目前狀態是觀察平衡(見圖 29A 的節

點管理者狀態 2902)。若目前狀態不是觀察平衡，在方塊 1740 中監視處理者 225 回到圖 12 的邏輯。此處狀態為觀察平衡，所以在回到圖 12 的邏輯前，監視處理者 225 通知節點平衡者 235，以進行一上層決定，其在返回圖 12 的邏輯之前，系統正在進行的工作是否適當地在系統的節點間被平衡〔方塊 1740〕。此邏輯連同圖 18 被描述。

在觀察平衡狀態的節點平衡者 235

現在看到圖 18，節點平衡者 235 繼續加總六個最近之儲存區域的呼叫計數〔方塊 1800〕。節點平衡者 235 然後計算每一節點的平均 DCP〔方塊 1805〕。在較佳實施例中，此透過首先從最近六個儲存區域以節點為基礎加總 DCPA 而達成。每一節點的總和然後除以呼叫計數的總和（即，在方塊 1800 中被決定的值）。結果為每一節點的平均 DCP。節點平衡者 235 然後以節點為基礎加總六個最近儲存區域的觀察平衡週期計數〔方塊 1810〕。（見圖 29B，週期計數和儲存區域 SA0-SAN，2932-2936。）在方塊 1815 中系統總和然後被決定。做出已平衡／不平衡決定的步驟繼續於圖 19。圖 19 的邏輯被節點平衡者 235 對每一節點使用一次。每次圖 19 的邏輯被使用時，下面的輸入會被考慮：1) 系統全部週期計數、2) 考慮中節點的 DCP、3) 節點的目標權重（對於目前權重如何計算的解釋見圖 10B 和關聯的文字）、和 4) 特定節點的週期計數。

現在移動至圖 19，在方塊 1900 中節點平衡者 235 繼

續計算目標週期計數。在較佳實施例中，此藉由將橫跨所有節點所被執行的全部週期，乘上特定節點的目標權重而達成。結果是討論中節點的目標週期計數。然後從節點的週期計數中減去目標週期計數〔方塊 1910〕。

在較佳實施例中， $1/32$ 的目標週期計數被用來當做容忍限度，使得若實際週期計數在目標週期計數的 $1/32$ 之內，節點被稱為被平衡，即使其可能稍微失去平衡。此容忍限度的應用可見於處理方塊對 1905/1920 和 1915/1925 的邏輯中。若實際週期計數僅稍微少於目標週期計數〔方塊 1905 和 1920〕，零的結果被傳回〔方塊 1935〕。零的結果指示對特定節點沒有看到不平衡。類似地，若實際週期計數僅稍微多於目標週期計數〔方塊 1915 和 1925〕，零的結果被傳回〔方塊 1935〕。當然，若方塊 1910 之計算的結果為零，在方塊 1935 中零被傳回。若實際週期計數和目標週期間的差異不只稍微低，圖 19 的邏輯將傳回實際和目標週期計數間的差異為負數〔方塊 1930〕，係指示特定節點能夠接受而不超過其公平比例之額外工作的量。若實際週期計數和目標週期間的差異不只稍微高，圖 19 的邏輯將測試是否特定節點的 DCP 大於 75%。超額的週期計數和大於 75% 的 DCP，指示節點有多於其公平份量的工作，且實際上正受超額工作之苦。當發現情況為此，圖 19 的邏輯將傳回實際週期和目標週期計數間的差異為正數〔方塊 1945〕，係指示應該從討論中節點被移走的工作份量。若在方塊 1940 中，圖 19 的邏輯決定儘管週期計數的超額，特定節點的 DCP 並沒有超過 75%，該節點並不被當作正在過度工作〔方塊 1935〕。

回想圖 19 的邏輯係被用於每一節點，且現在回到圖 18 而假定所有節點已經被考慮，如果圖 19 之邏輯的結果顯示沒有工作的不平衡（即，工作適當地分佈遍及系統），在方塊 1840 中節點平衡者 235 僅結束執行。然而，如果邏輯顯示不平衡（即，至少一個節點具有太多工作且至少一個節點具有太少工作），節點平衡者 235 繼續至方塊 1830，在該處記錄中狀態值被初始化如下：1) 圖 29C 的記錄中期間 Id 2940 被增加、2) 週期計數 2945 以主節點為基礎被設定為零、3) 記錄中期間呼叫計數 2942 被設定為零、4) 每一節點的 DCPA 2948 被設定為零。然後在節點平衡者 235 於方塊 1840 中結束執行之前，節點管理者狀態 2905 被設定為記錄週期〔方塊 1825〕。

在記錄週期狀態的分發處理者

現在狀態已經被節點平衡者 235 設定至記錄週期（見上面），所以分發處理者 230 的處理方塊 1645 下一次執行緒被分發時將評估為是。相對應地，分發處理者 230 將接下來決定執行緒的主節點是否可以移動〔圖 16B 的方塊 1609〕。若否，在回到圖 16A 的邏輯前，分發處理者 230 在方塊 1660 中僅增加記錄週期的週期累加器 (cycle accumulator)。若如此，分發處理者 230 決定是否記錄中期間 ID 相符〔方塊 1610〕。此決定藉由比較保存於執行緒控制結構（未顯示）中的記錄中期間 ID，和圖 29C 的記錄中期間 ID 2940 而做成。若 ID 相符，分發處理者 230 增加所計數的原始週期至已經儲存在執行緒結構（未顯

示) 中的現存總數 [方塊 1655]。當 ID 不相符時，目前週期的原始週期計數尚未被儲存於此特定執行緒的控制結構中，所以執行緒控制結構中的記錄中期間 ID，被設定為儲存在記錄中期間 ID 2940 中的值 [方塊 1650]，且週期計數被設定等於最近所被執行的週期 [方塊 1665]。此為方塊 1605 中所確定的值。不管處理方塊 1610 如何評估，已記錄週期的主節點總數在方塊 1660 中也被增加。

在記錄週期狀態的監視處理者 225

現在回到圖 13 以繼續，下一次監視處理者 225 執行處理方塊 1305 將評估為是，因為狀態 2902 現在被節點平衡者 235 設定至記錄週期。監視處理者 225 將 DCP 加至記錄中狀態 DCPA (見圖 29C 之 2948 之 DCPA) [方塊 1310]。記錄中期間 IC 2942 然後在方塊 1315 中被增加。監視處理者 225 然後測試記錄中期間 IC 2942 以決定是否最大值已經被達到。在較佳實施例中最大值為五十三。若最大值尚未被達到，監視處理者 225 在方塊 1340 中僅離開。若在方塊 1320 中，監視處理者 225 決定最大值已經被達到，其繼續從分發處理者 230 獲得已記錄週期計數，且儲存它們於週期計數儲存區域 2948 中。監視處理者 225 然後在方塊 1335 中設定狀態為移動執行緒，且在方塊 1340 中離開前，通知節點平衡者 235 要移動執行緒。

在移動執行緒狀態的節點平衡者 235

圖 20 於此用來描述在較佳實施例內的執行緒移動。節點平衡者 235 首先計算每一節點的平均 DCP [方塊 2000]。在較佳實施例中，此藉由將 DCPA 2948 中所持有的值除以在週期內期間的總數（即，記錄週期 IC 2942）。節點平衡者 235 然後加總 2948 之已累加週期計數，係產生系統在記錄中期間內的總數 [方塊 2005]。

節點平衡者 235 接下來決定工作是否在記錄中期間被公平地分佈遍及系統。在較佳實施例中，此透過圖 19 中的邏輯達成。如同讀者可能回想起，圖 19 已在上面被描述，且細節將不在此重述。然而，回想圖 19 的邏輯對每一節點使用，且現在回到圖 20 而假定所有的節點已經都被考慮，若圖 19 之邏輯的結果顯示沒有工作的不平衡（即，工作被適當地分佈遍及系統），節點平衡者 235 在方塊 2035 中僅設定狀態至觀察平衡，且在方塊 2040 中結束執行。然而，若圖 19 的邏輯顯示不平衡（即，至少有一個節點具有太多工作，且至少有一個節點具有太少工作，且相同的節點在觀察平衡狀態被觀察到相對應地有太多工作和太少工作），節點平衡者 235 繼續至方塊 2020，在該處要移動之週期的數目被決定。讀者被要求首先回想起週期短缺的節點和具有過量週期的節點在方塊 2010 中被識別。要移動之週期的數目然後由節點平衡者 235 設定至最大過量和最大短缺之較小者的 $1/2$ 。節點平衡者 235 然後以此為基礎移動執行緒。

圖 21A 至 23 顯示節點平衡者 235 邏輯被用於較佳實施例中來進行此任務。最初，在方塊 2100，空白的列表

被建立。此列表將最終記載為將被移動之候選者的執行緒。然後，在方塊 2105 中，節點平衡者 235 從由系統所維護之執行緒的列表中選擇第一個（或下一個）執行緒。節點平衡者 235 然後決定是否執行緒有目前記錄中期間 ID 與其關聯〔方塊 2110〕。若否，節點平衡者 235 決定是否有更多執行緒要考慮〔方塊 2111〕，且如果是，繼續選擇下一個執行緒〔方塊 2105〕，並再次決定是否有目前記錄中期間 ID 與其關聯〔方塊 2110〕。圖 21B 顯示被用來處理具有目前記錄中期間 ID 之執行緒的邏輯。節點平衡者 235 首先在方塊 2110 中決定討論中之執行緒的主節點，是否為已經在先前決定有過量之週期的節點。若否，在圖 21A 的方塊 2111 中處理繼續，如同已經被描述。若討論中之執行緒的主節點的確是有過量週期者，節點 235 決定是否特定執行緒為 MAG 的成員，其中群組的所有執行緒被要求要有相同的主節點〔2125〕。若否，節點平衡者 235 提取由此執行緒所記錄的週期〔方塊 2130〕。節點平衡者 235 然後將增加此執行緒至候選者列表，且基於其所記錄週期的數目做排名（最多的為第一）。節點平衡者 235 然後在圖 21A 的方塊 2111 中繼續處理。

若在圖 21B 的方塊 2125 中，節點平衡者 235 決定討論中的執行緒的確是群組的一部分，節點平衡者 235 決定是否該群組為在此記錄中期間已經在先前被遇到者〔方塊 2140〕。如果如此，在圖 21A 的方塊 2111 中處理繼續。若否，節點平衡者 235 計算群組的總週期計數（即，群組之執行緒之已記錄週期的總和）〔方塊 214〕，且然後基於群組的總週期計數增加群組至移動候選者列表〔方塊 2150〕。

]。在達成之後，在圖 21A 的方塊 2111 中處理繼續。

當所有執行緒都已經被考慮時，處理方塊 211 評估為否，且節點平衡者 235 繼續來實際移動執行緒以重新平衡系統。用於較佳實施例中來達成此任務的邏輯被顯示在圖 22 中。在方塊 2220，節點平衡者 235 在候選者列表中選擇第一個（下一個）執行緒或 MAG。假定列表的結束尚未達到〔方塊 2205〕，節點平衡者 235 決定是否討論中的執行緒或 MAG 有比必要來移動更多或更少的週期（見處理方塊 2020 和關聯的文字）〔方塊 2215〕。若執行緒或 MAG 消耗比所需要之更多的週期，節點 235 在方塊 2020 中僅繼續處理。若否，節點處理者 235 決定正在被考慮者是否為執行緒或群組〔方塊 2225〕。若其為執行緒，節點平衡者 235 繼續至圖 23 的邏輯。若其為群組，在繼續來對群組中的每一執行緒使用圖 23 的邏輯之前，群組的主節點在方塊 2330 中被改變。

現在看到圖 23，節點平衡者 235 改變討論中執行緒的主節點〔方塊 2300〕。只要特定節點保留此特定主節點，TDM 215 將嘗試且分發它在它的新主節點上執行，且 TMM 217 將嘗試在它的新主節點上獲得它的儲存。最後，在方塊 2305 中，節點平衡者 235 將重新組成與執行緒關聯的重要記憶體頁面。在較佳實施例中，此係藉由使得在舊的主節點上的頁面暫時無法使用、複製那些頁面到新的主節點且使那些頁面在新的主節點上為可用。現在回到圖 22，每一次執行緒或執行緒的群組被移動到新的主節點，需要移動的週期便減少了已移動的週期數〔方塊 2240〕。

在方塊 2200 中處理然後繼續直到所有執行緒已經被考慮。當所有執行緒已經被考慮，處理方塊 2205 評估為是，使得節點平衡者 235 重新初始化 OIN。節點平衡者 235 也藉由提取節點週期計數，和儲存他們於最舊的儲存區域，來建立下一個觀察期間的基礎〔方塊 2210〕。這些初始化具有效果能使得下一個觀察期間完全重新開始，以致在更進一步的平衡被考慮前至少六個期間被觀察。

在進行初始化之後，節點平衡者 235 回到圖 20 的方塊 2035，於該處在於方塊 2040 中結束處理前，其設定狀態為觀察平衡。

在本專利文件之工作重新分佈章節的開端，我們指出狀態起初被系統管理者設定為觀察平衡。此為必要的，因為，如同使用者可能回想起，節點管理者狀態 2902 在 IPL 期間被設定至目前權重（見圖 6 的方塊 605）。圖 24 和 25 分別顯示系統和系統管理者間啟動主節點移動（即，重新平衡）和關閉主節點移動的互動。在較佳實施例中，系統管理者透過使用者介面 150 與電腦系統 100 互動。在該處，系統管理者能夠輸入命令，以指示節點管理者 220 將自己設定至觀察平衡狀態的指令，且因此啟動主節點移動。在圖 24 的方塊 2400 中，節點管理者 220 決定是否它已經在觀察平衡或再往後的狀態。如果如此，節點管理者 220 僅通知系統管理者（透過使用者介面 150）主節點移動已經被啟動〔方塊 2405〕。若節點管理者 220 不在觀察平衡狀態，節點管理者 220 初始化 OIN，如同在與圖 22 之方塊 2210 關聯的文字中所描述。節點管理者 220 然

後在返回前在方塊 2415 中設定其狀態至觀察平衡。

系統管理者能夠輸入命令，以指示節點管理者 220 將自己設定至目前權重狀態的指令，且因此關閉主節點移動。在圖 25 的方塊 2500 中，節點管理者 220 決定是否它已經在目前權重狀態。如果如此，節點管理者 220 僅通知系統管理者（透過使用者介面 150）主節點移動已經被關閉〔方塊 2505〕。若節點管理者 220 不在目前權重狀態，節點管理者 220 然後在返回前，於方塊 2510 中設定其狀態至目前權重。

在此處所提出的實施例和範例，是為了能最佳地解釋本發明和其實際應用，且因此使習知本技術者能夠製作和使用本發明。然而，習知本技術者將認知前述描述和範例只是為了說明和舉例的目的而被呈現。如所提出的敘述並非是徹底的或限制本發明於所揭示的特定形式。依照上面的教示，仍有許多的修改和變化，而不離開後附申請專利範圍的精神和範圍。

【圖式簡單說明】

圖 1A 為方塊圖，係顯示能夠支援本發明較佳實施例的計算環境；

圖 1B 為方塊圖，係顯示能夠支援本發明較佳實施例的另一計算環境；

圖 2 為方塊圖，係顯示本發明較佳實施例之核心服務之示意圖；

圖 3A 和 3B 為流程圖，係顯示用來進行較佳實施例

之執行緒建立機制中，重要的處理步驟；

圖 4 為流程圖，係顯示用來從親近群組中解除執行緒關聯之較佳實施例之執行緒建立機制內的重要步驟；

圖 5 為流程圖，係顯示較佳實施例之執行緒建立機制內，用來使記憶體親近群組資訊可用於其他系統元件之重要步驟；

圖 6 為流程圖，係顯示起始程式載入（IPL）序列內的重要步驟，用來起始用於較佳實施例中的各種權重，以達成核心元件平衡通知（KCBN）；

圖 7 至 11、14 和 15 為流程圖，係顯示用於較佳實施例內的重要步驟，以因應系統內的組態變動，而調整各種 KCBN 權重；

圖 11 至 14 和 16 至 25 為流程圖，係顯示用於較佳實施例之監視處置器(Watch Dog Handler)與節點平衡器(Node Balancer)中的重要步驟，用來進行工作重分佈。

圖 26 至 28 為流程圖，係顯示主節點分派的重要步驟，係有關於較佳實施例之執行緒建立機制；以及

圖 29A 至 31 顯示用於本發明較佳實施例中的資料結構。

【主要元件符號說明】

100	電腦裝置	125	網路介面
130	節點	131	節點
132	處理器	134	記憶體
135	執行緒	136	作業系統
137	核心服務		
150	使用者介面	155	大量儲存裝置

200 執行緒管理者	205 執行緒建立機制
215 執行緒分配機制	217 執行緒記憶體管理者
220 節點管理者	225 監視處理者
230 分發處理者	235 和節點平衡者
240 組態管理者	
2902 節點管理者狀態	2904 元件登錄指標
2906 目前權重	2907 分發者許諾百分比
2908 目前權重期間編號(CWIN)	
2910 目前權重呼叫計數(CWIC)	
2911 目前權重識別符(CWID)	
2912 分發者許諾百分比累加器(DCPA)	
2914 最近執行週期計數(RECC)	
2916 分發者許諾百分比累加器(DCPA)	
2918 最近執行週期計數(RECC)	
2920 目標權重修改編號(TWMN)	
2922 目標權重	
2924 權重差異門檻值(WDT)	
2928 起始權重識別符(IWID)	
2930 起始權重	
2932、2934、2936 儲存區域	
2940 記錄中期間識別符(Id)	
2942 記錄中期間呼叫計數(IC)	
2948 儲存區域	2950 觀察期間編號(OIN)
3000 節點錨結構	3005 節點識別符
3010 節點狀態	3015 目前狀態
3020 先前狀態	3100 元件登錄
3105 已登錄元件	

五、中文發明摘要：

所揭示的一種裝置、方法和記錄媒體，用期望的系統工作負載的相同比例，以節點為基礎來達成作業系統資源的分佈。在本發明的較佳實施例中，係藉由分派各種不同種類的權重給每個節點，以代表他們在系統中整體平衡的比例。目標權重代表期望的工作負載分佈，其係基於每個節點上處理器與記憶體資源的現存比例。系統上實際的工作負載則是由目前權重來代表，而作業系統將其持續地保持儘可能接近目標權重。當系統開始時，作業系統服務是以目標權重相同的比例，來依節點(nodally)而分佈他們的資源，並且可以要求在目標權重改變時被通知。若處理器及/或記憶體在後續被加入或移除，則此時會計算新的目標權重，而且有提出請求的所有服務都會被通知，因此他們可以依照新的目標權重或其逐步的改進，來重新分佈他們的資源。

六、英文發明摘要：

Disclosed is an apparatus, method, and program product that enables distribution of operating system resources on a nodal basis in the same proportions as the expected system workload. The preferred embodiment of the present invention accomplishes this by assigning various types of weights to each node to represent their proportion of the overall balance within the system. Target Weights represent the desired distribution of the workload based on the existing proportions of processor and memory resources on each node. The actual workload balance on the system is represented by Current Weights, which the

operating system strives to keep as close to the Target Weights as possible, on an ongoing basis. When the system is started, operating system services distribute their resources nodally in the same proportions as the Target Weights, and can request to be notified if the Target Weights ever change. If processors and/or memory are subsequently added or removed, new Target Weights are calculated at that time, and all services which requested notification are notified so they can redistribute their resources according to the new Target Weights or a stepwise refinement thereof.

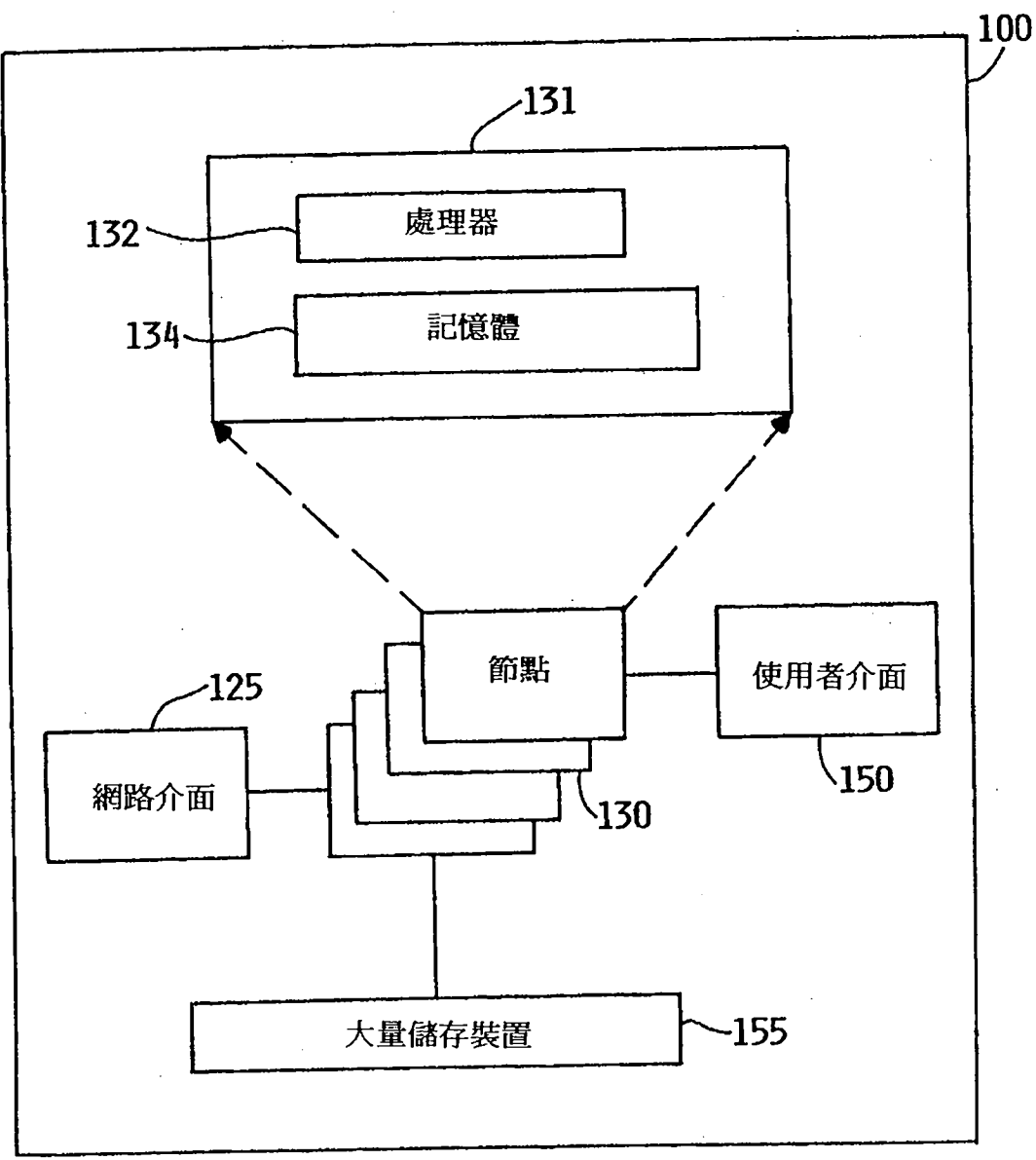


圖1A

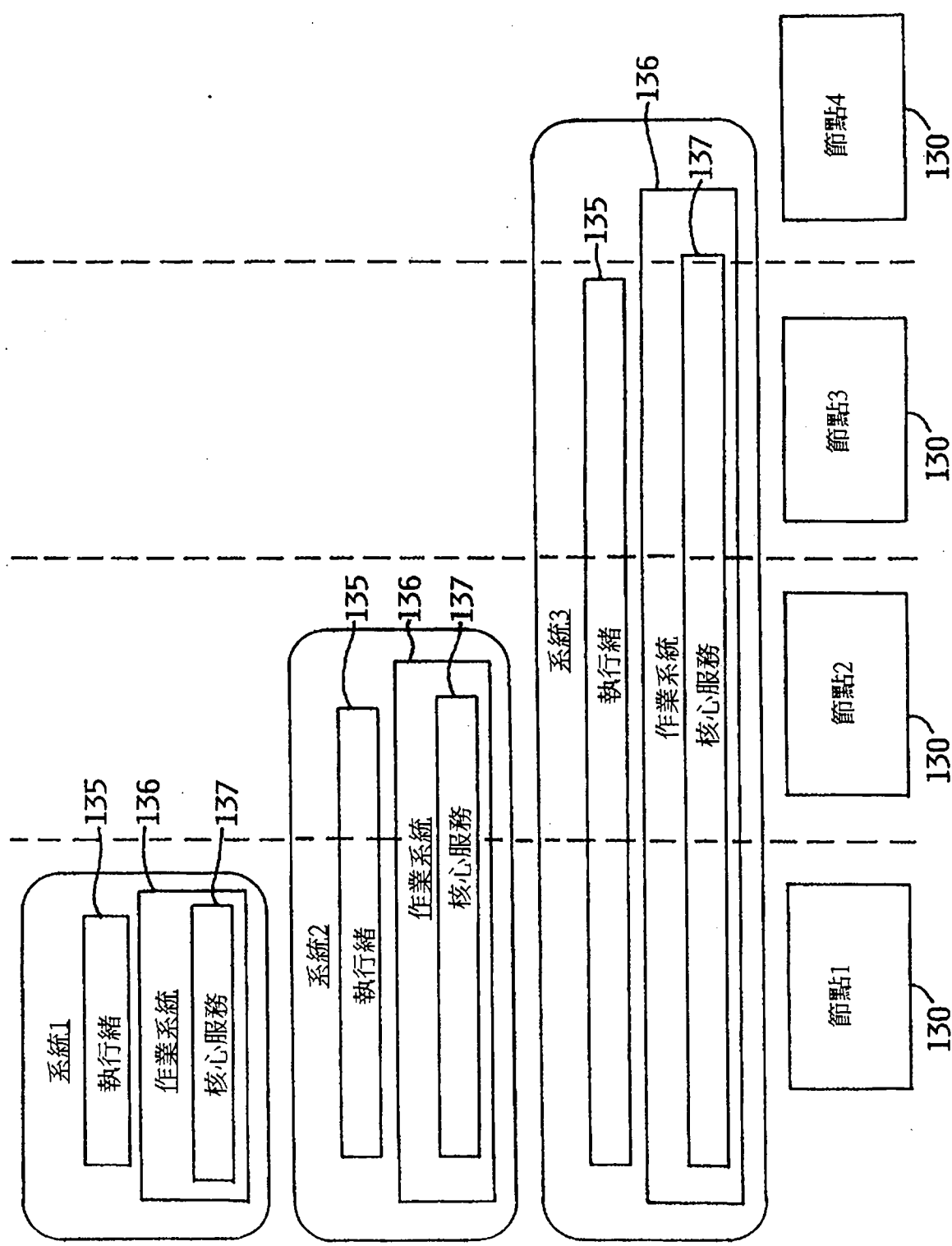


圖1B

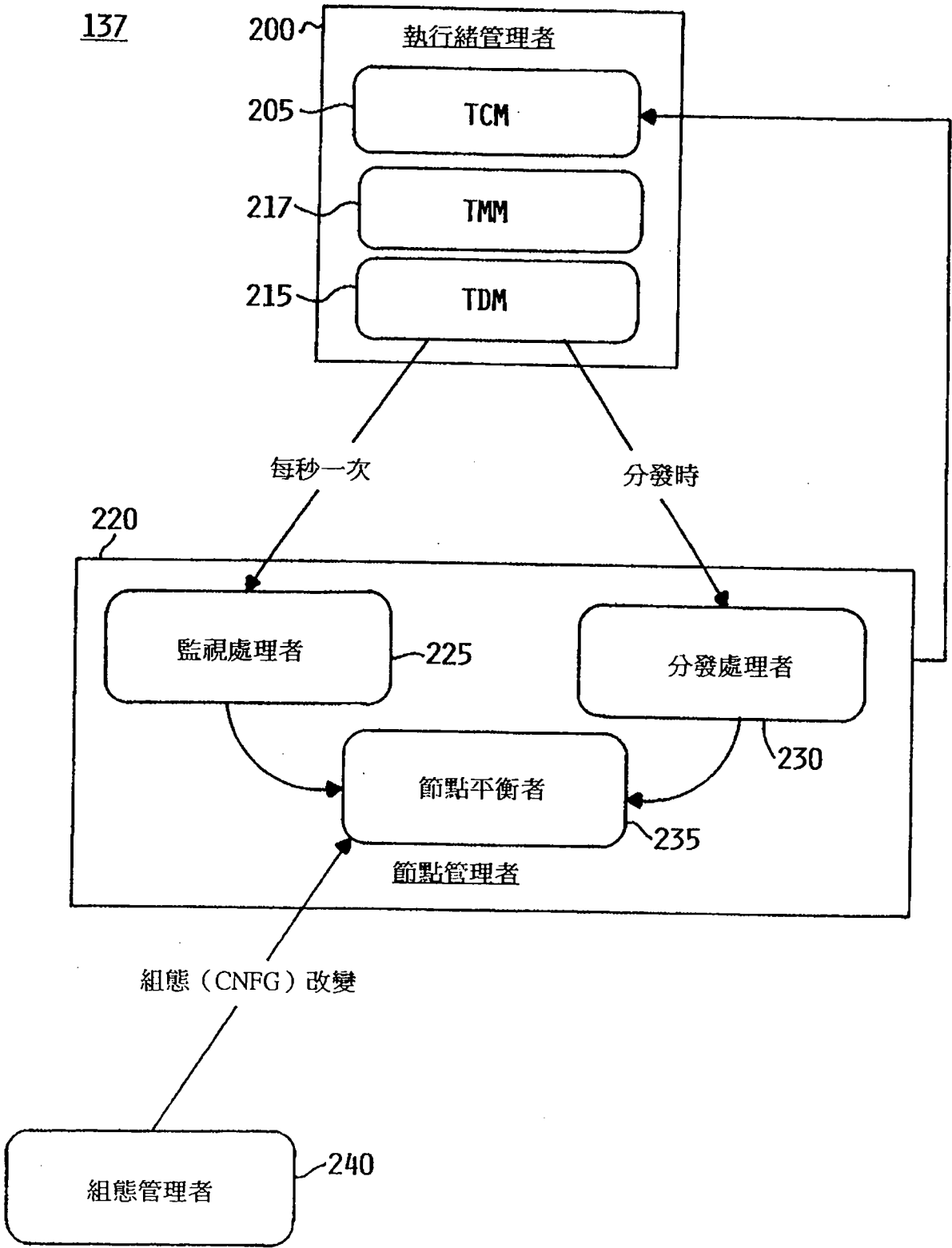


圖2

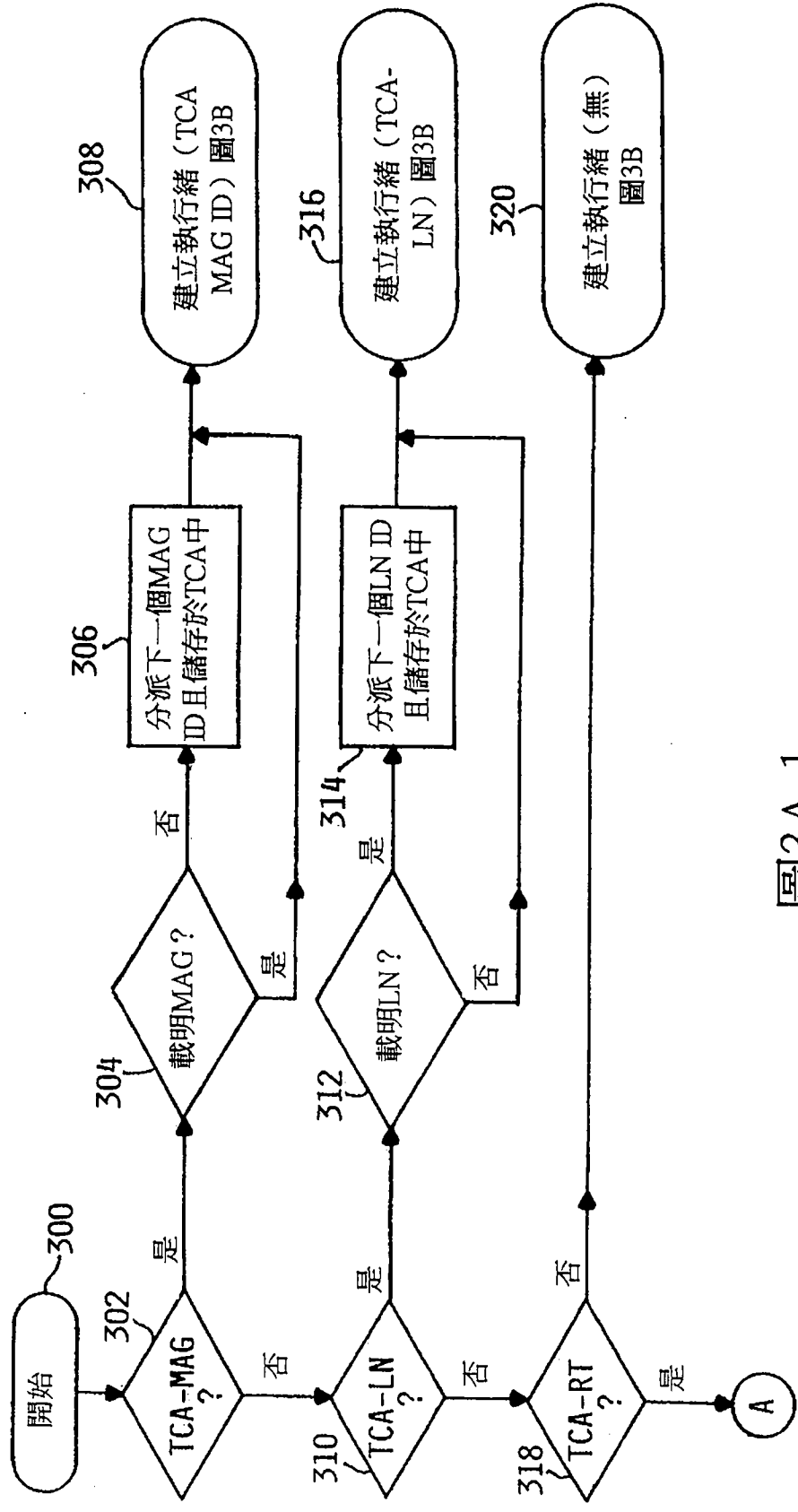


圖3A-1

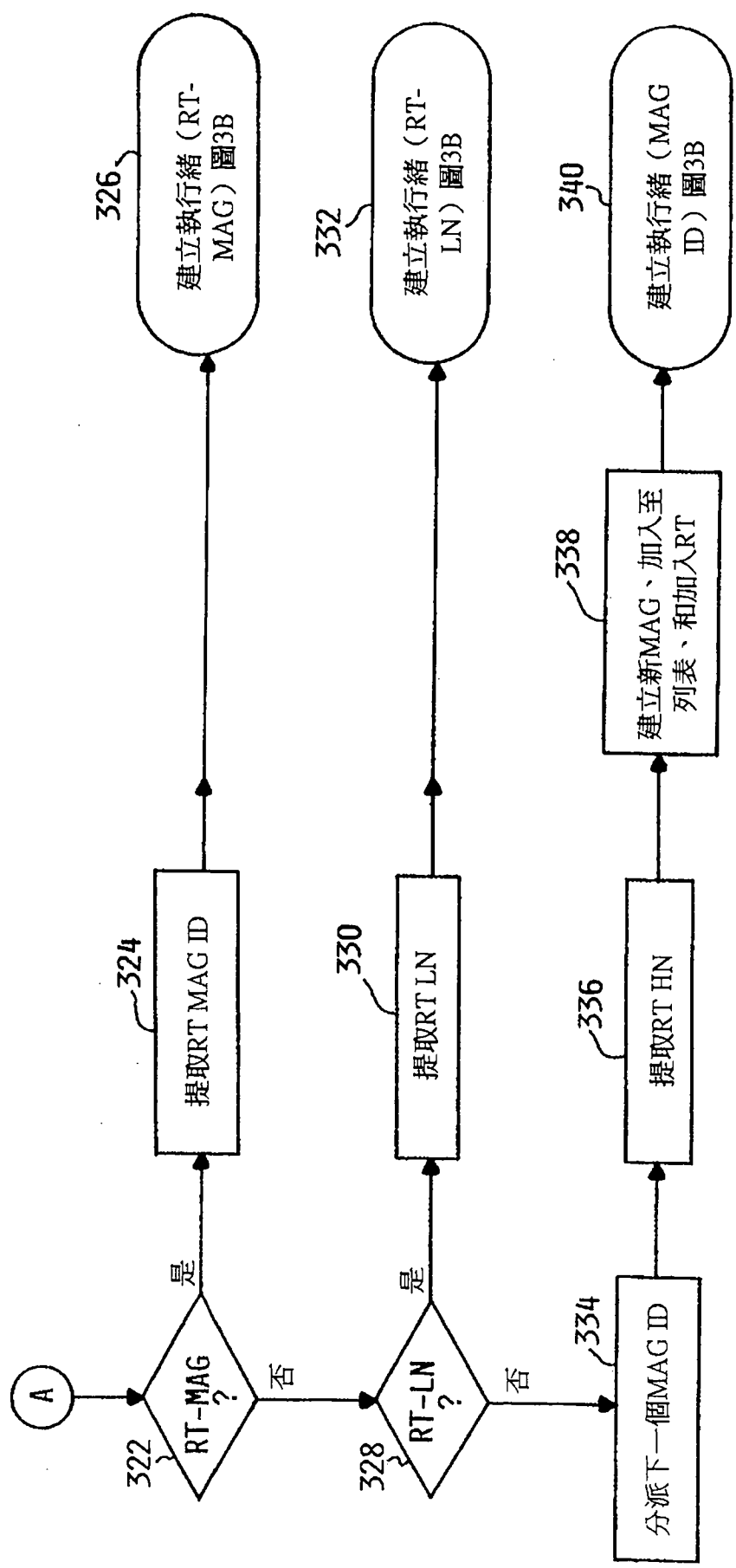


圖3A-2

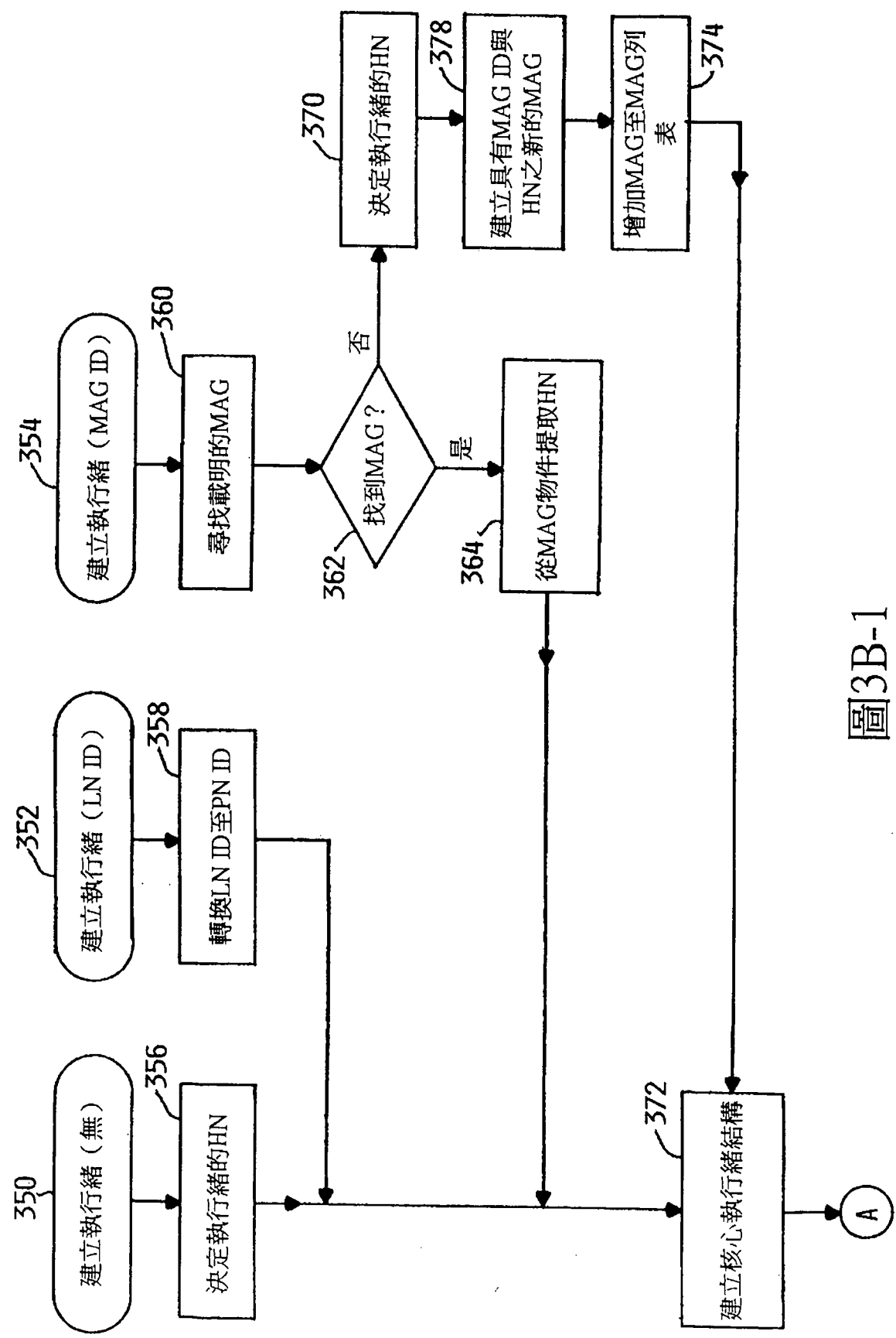


圖3B-1

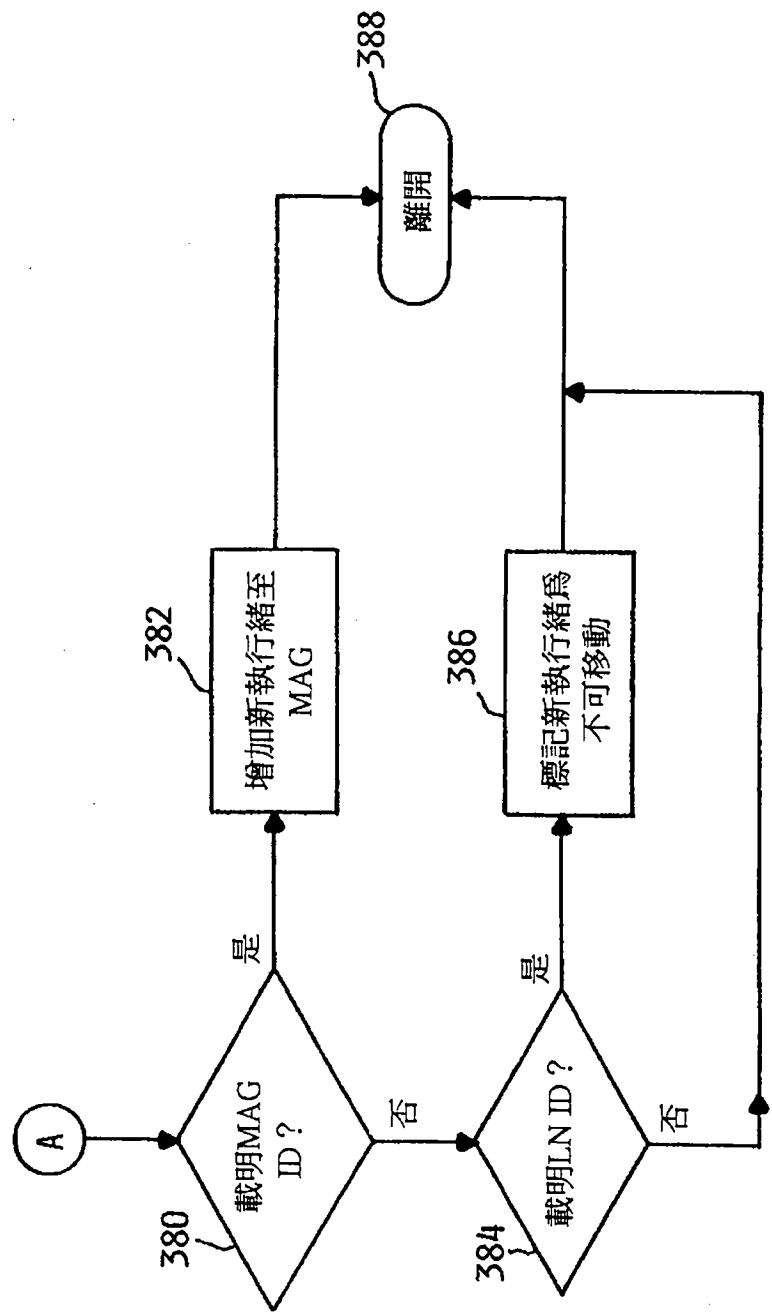


圖3B-2

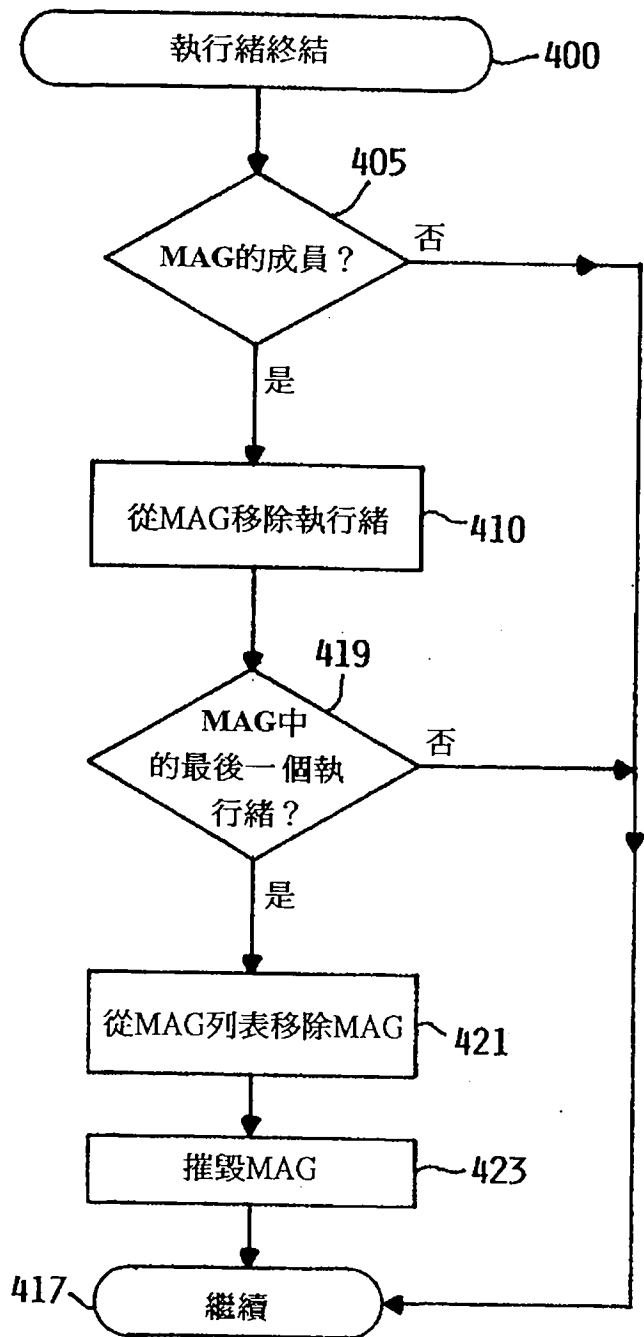


圖4

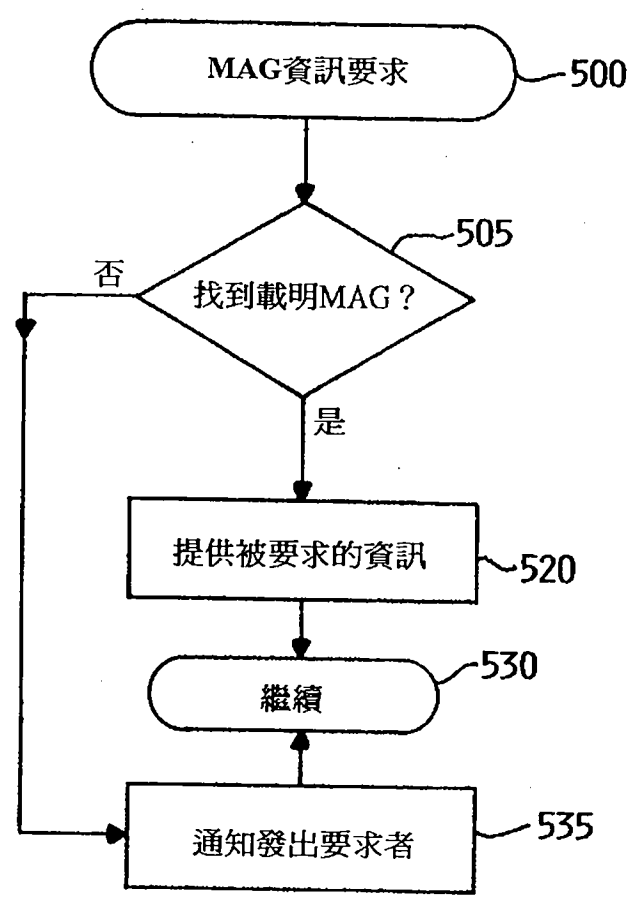


圖5

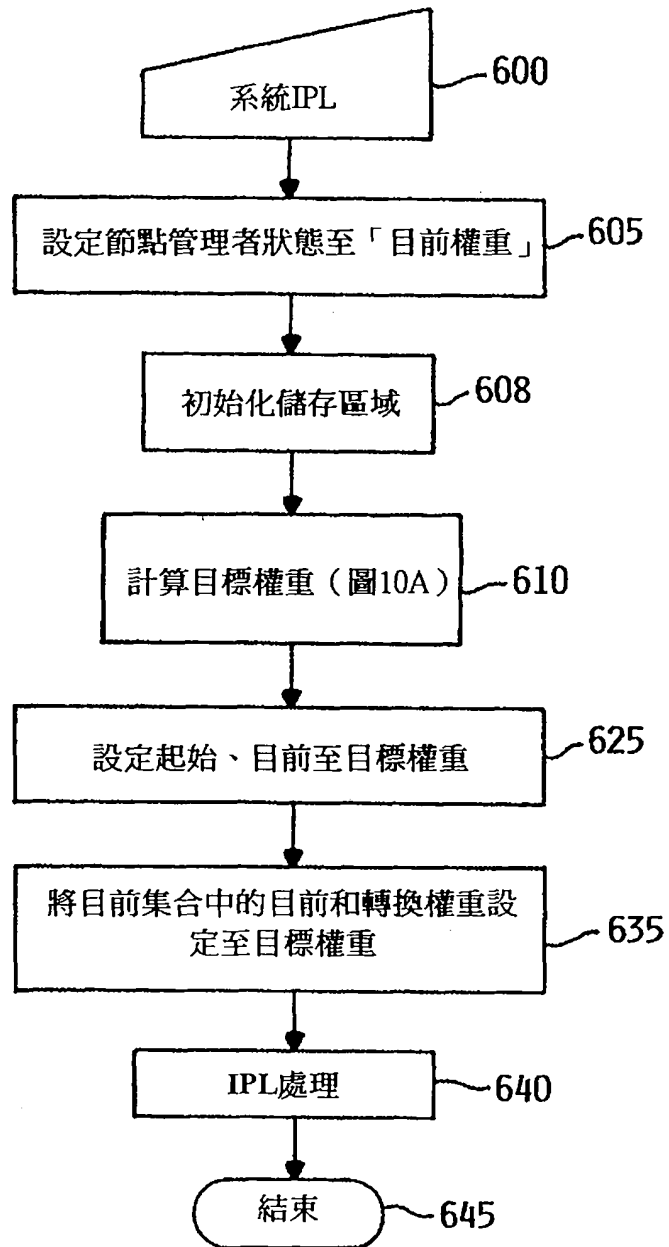


圖6

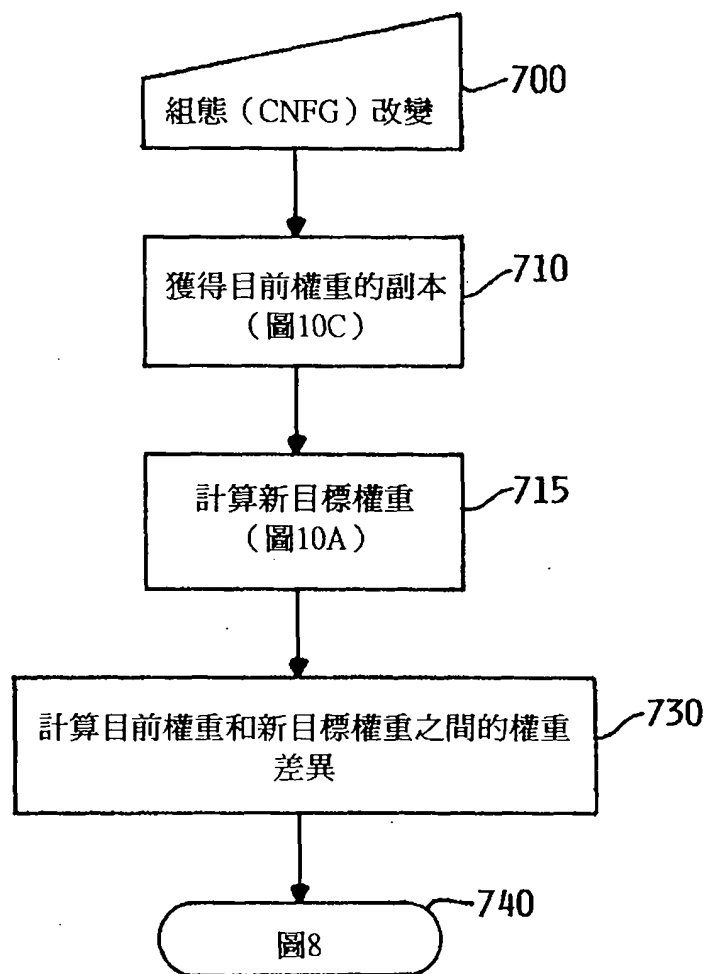


圖7

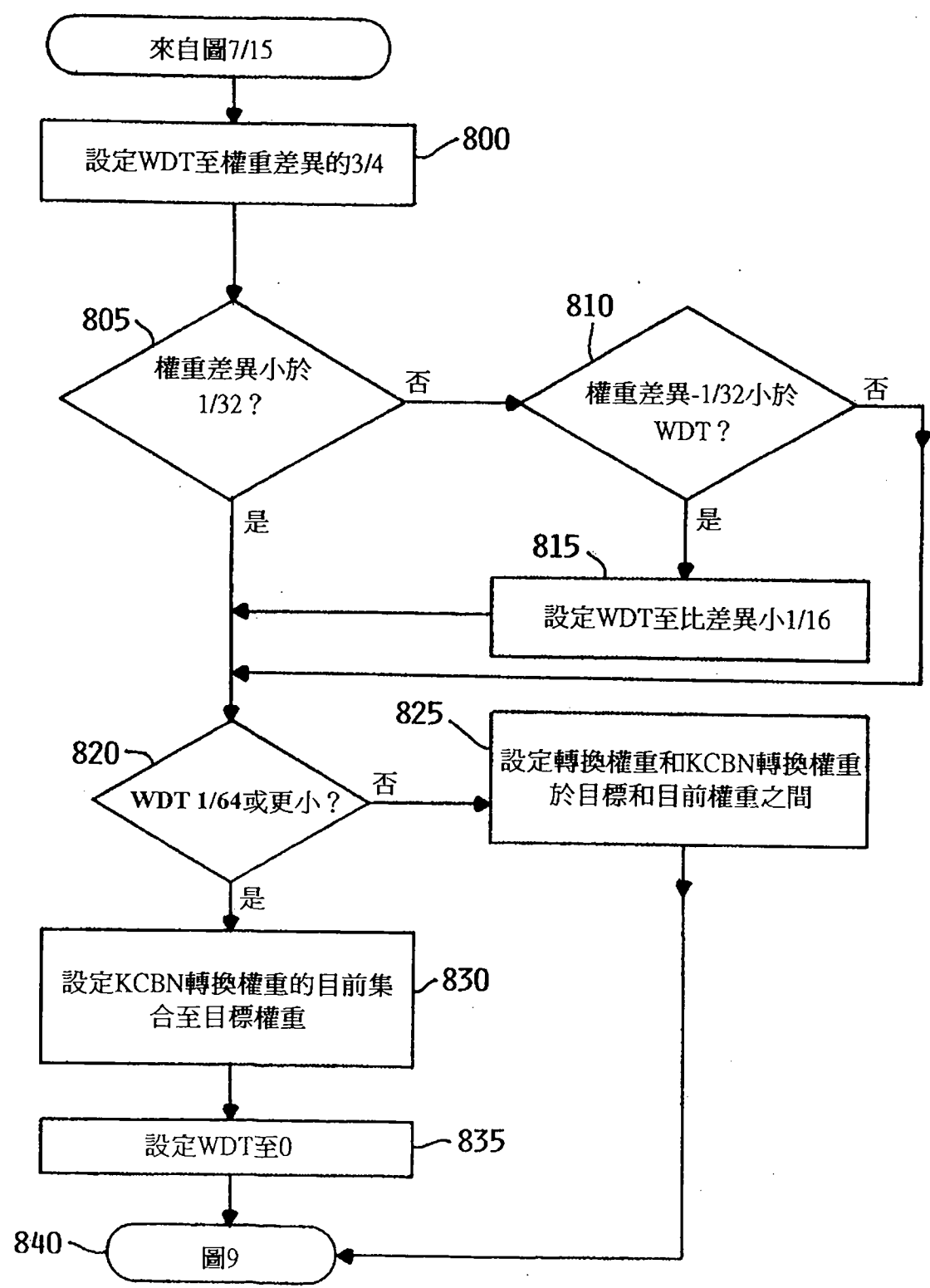


圖8

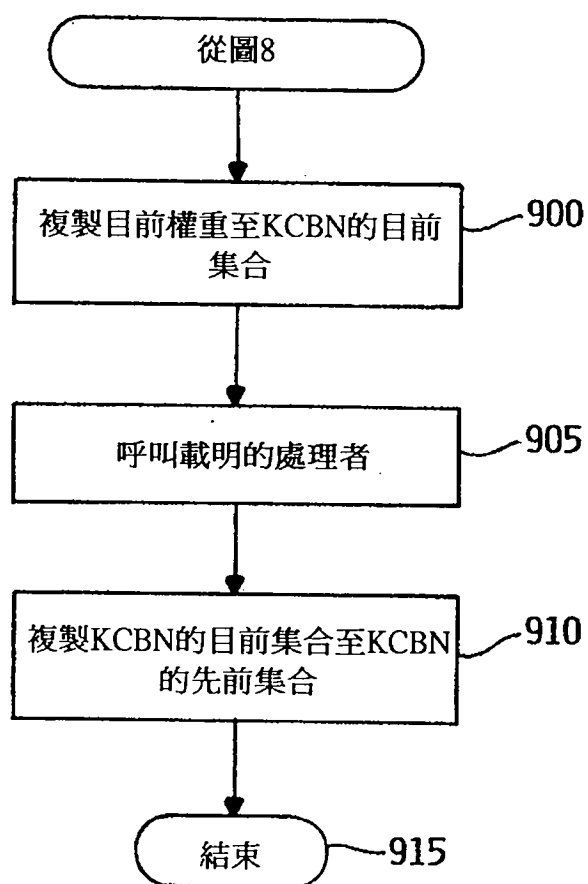


圖9

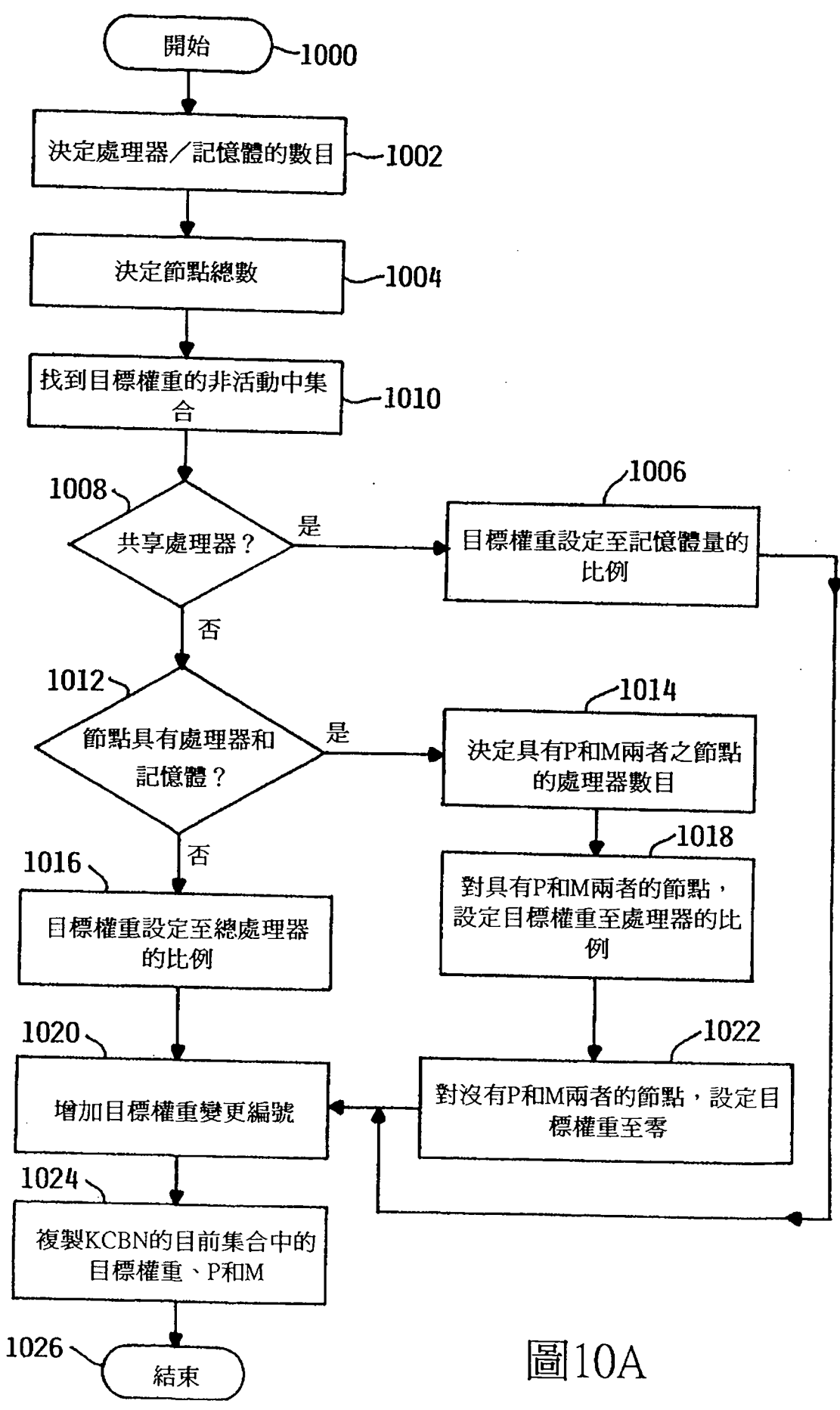


圖10A

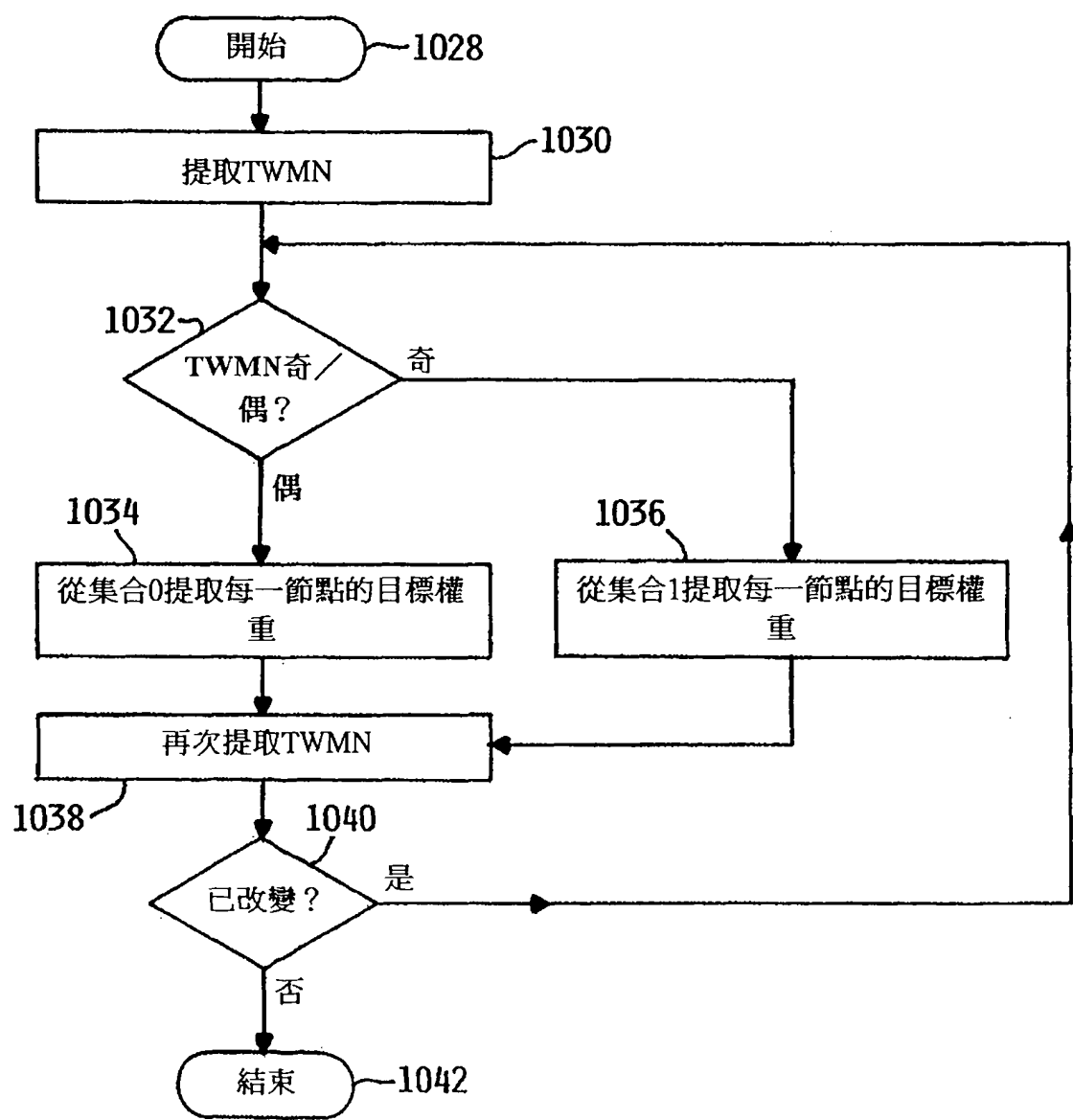


圖10B

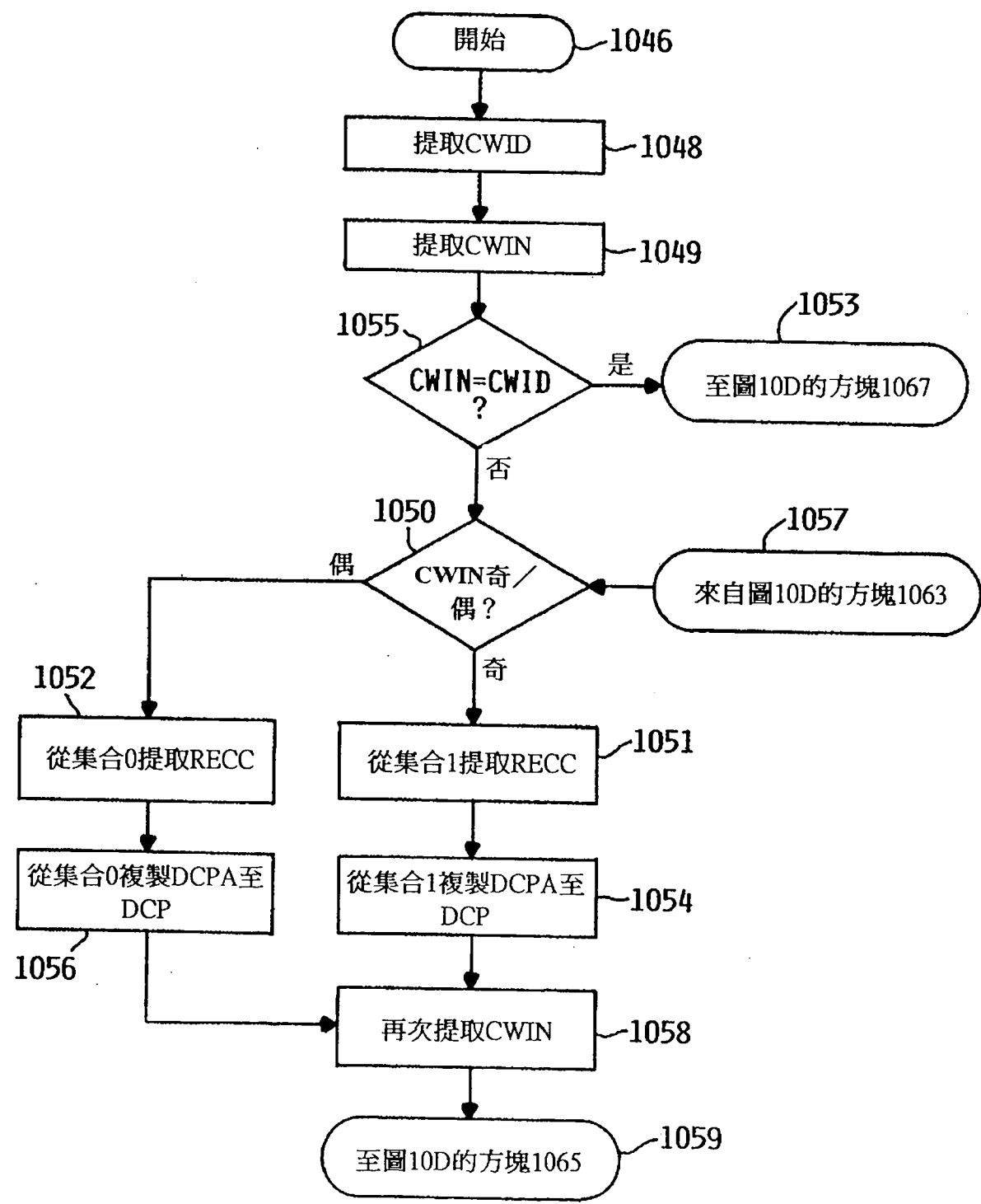


圖10C

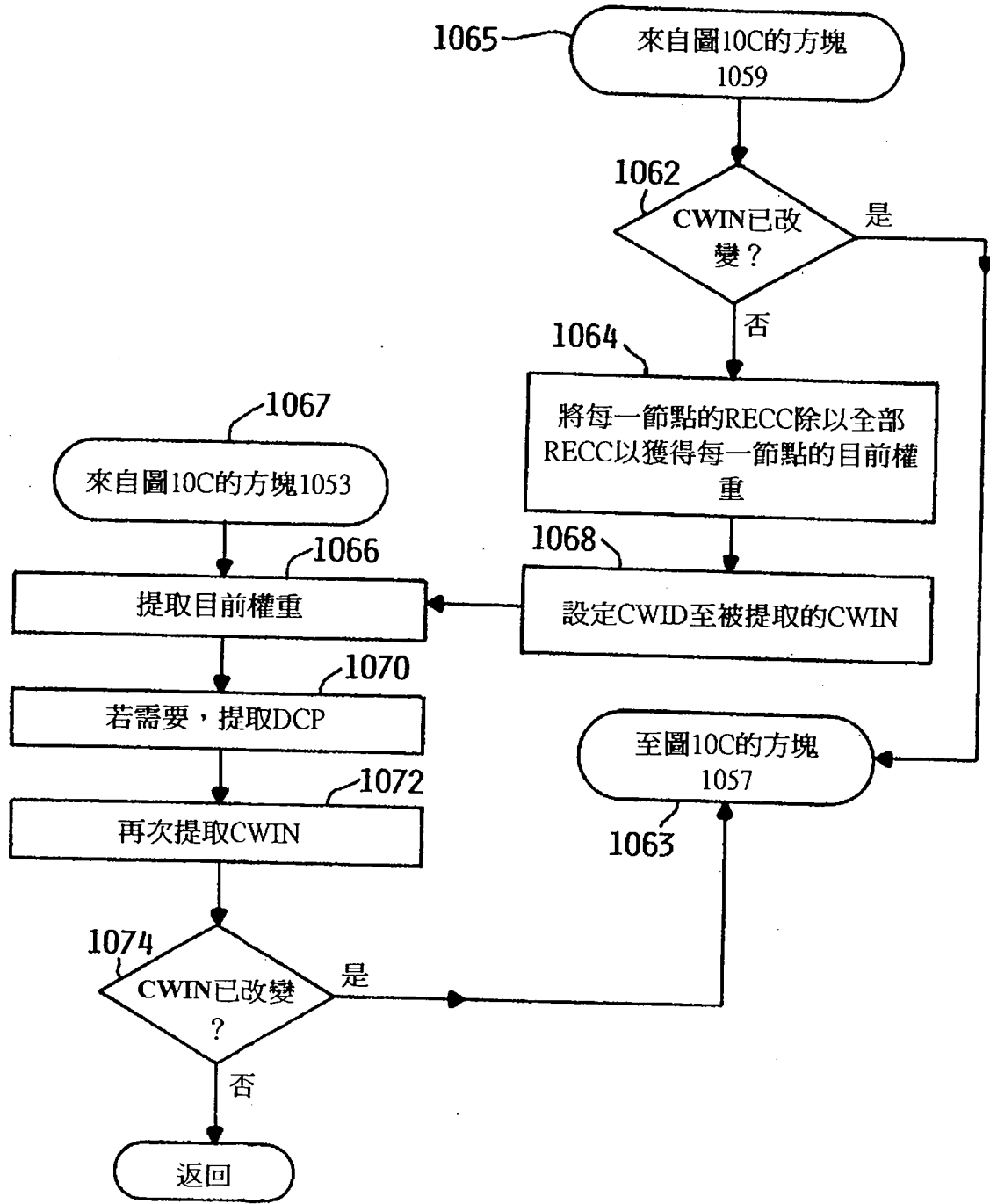


圖10D

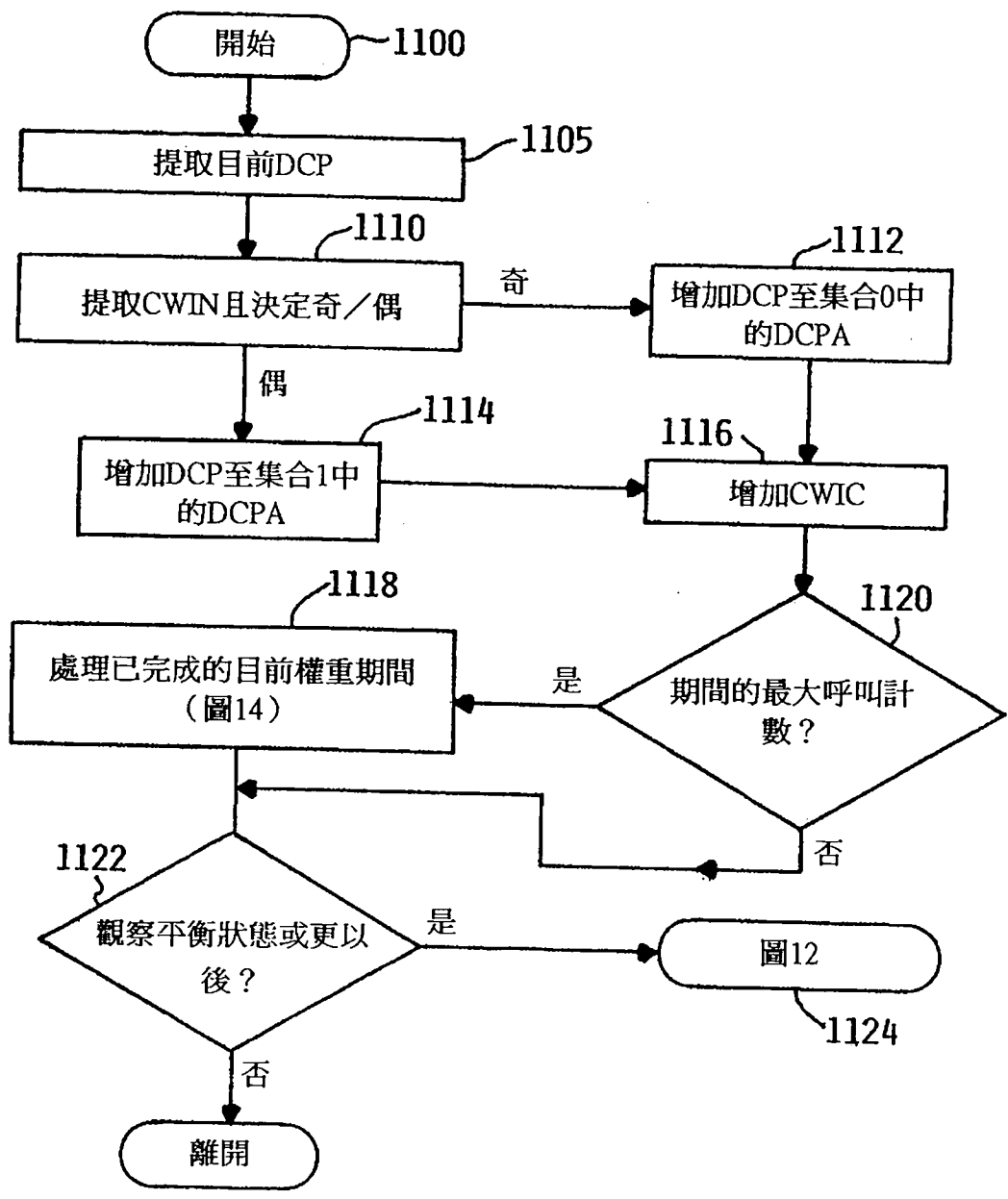


圖11

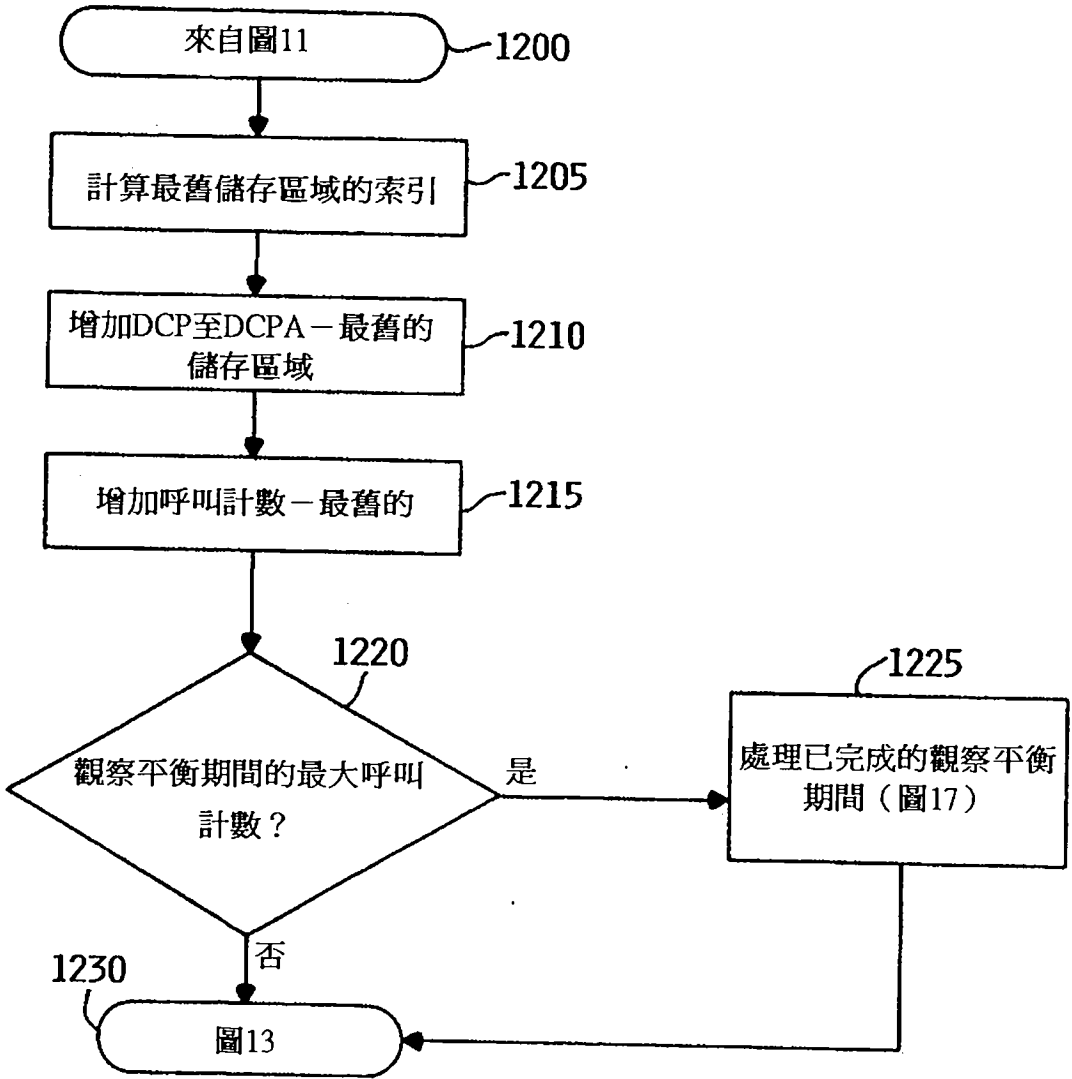


圖12

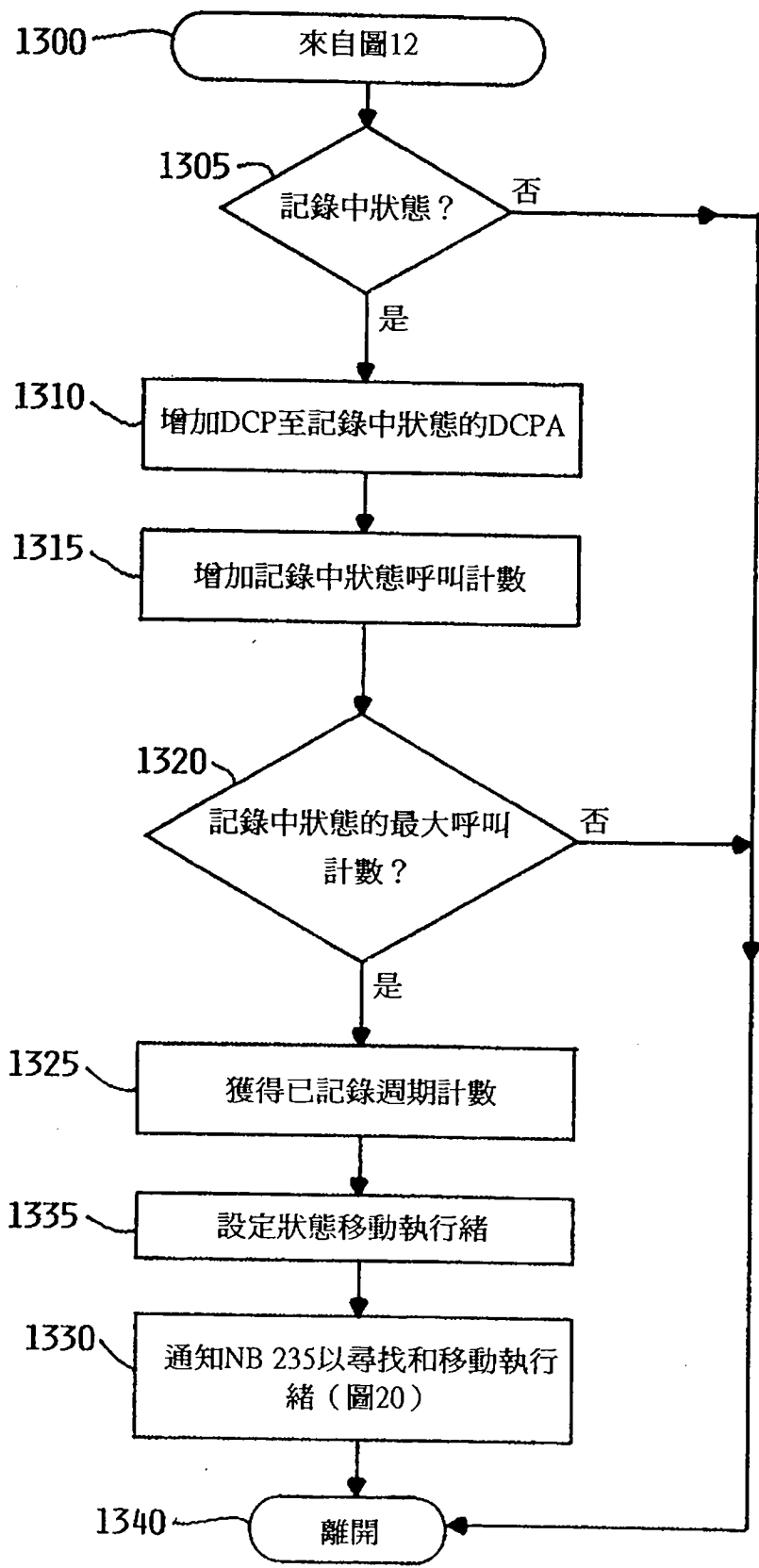


圖13

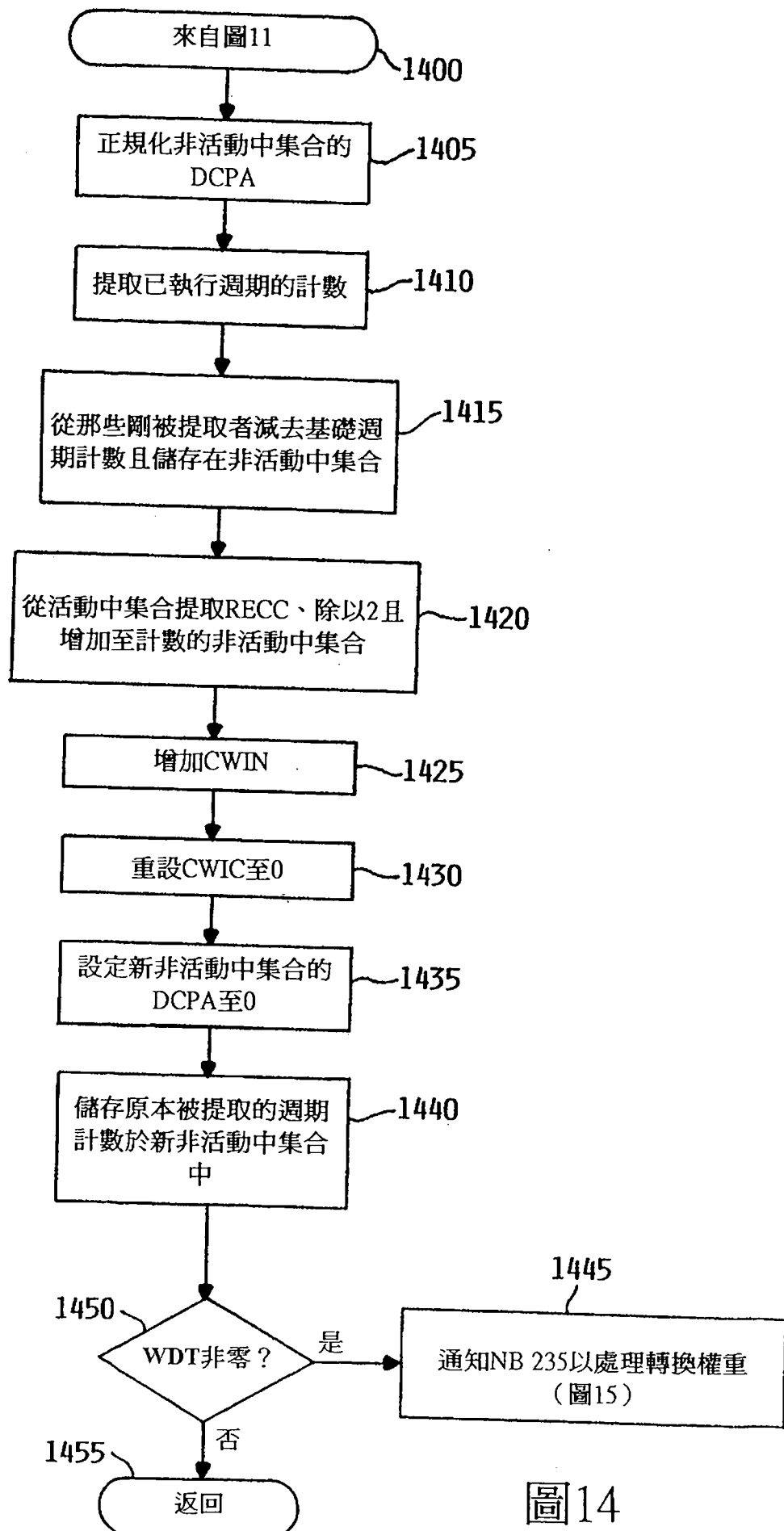


圖14

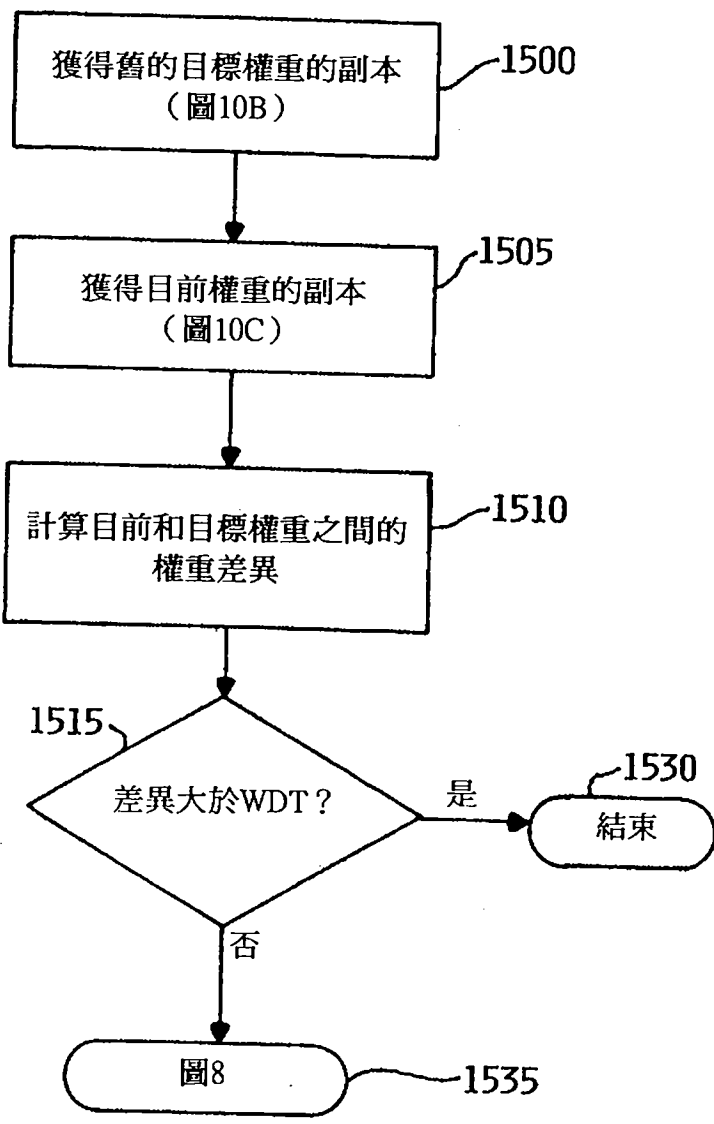


圖15

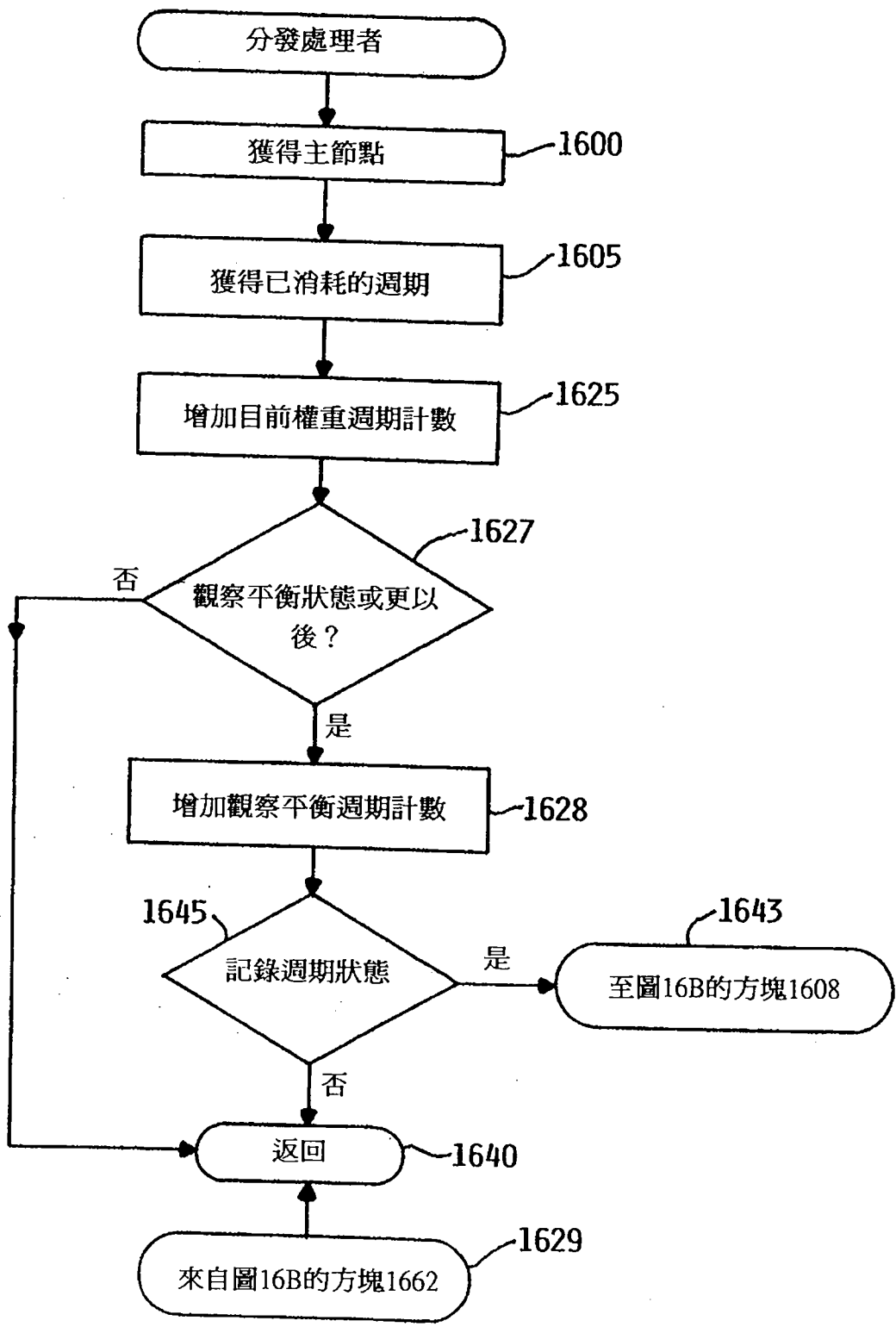


圖16A

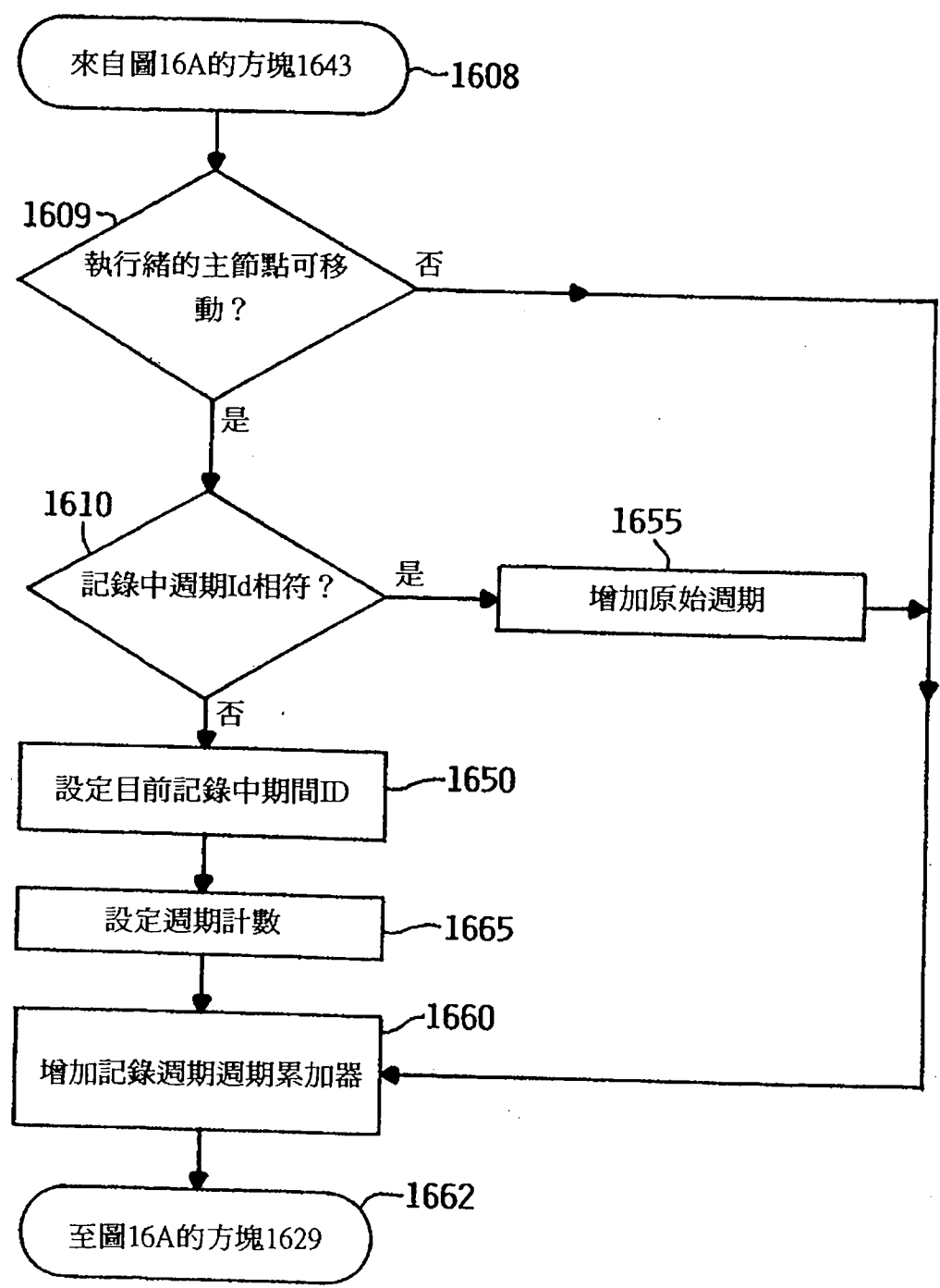


圖16B

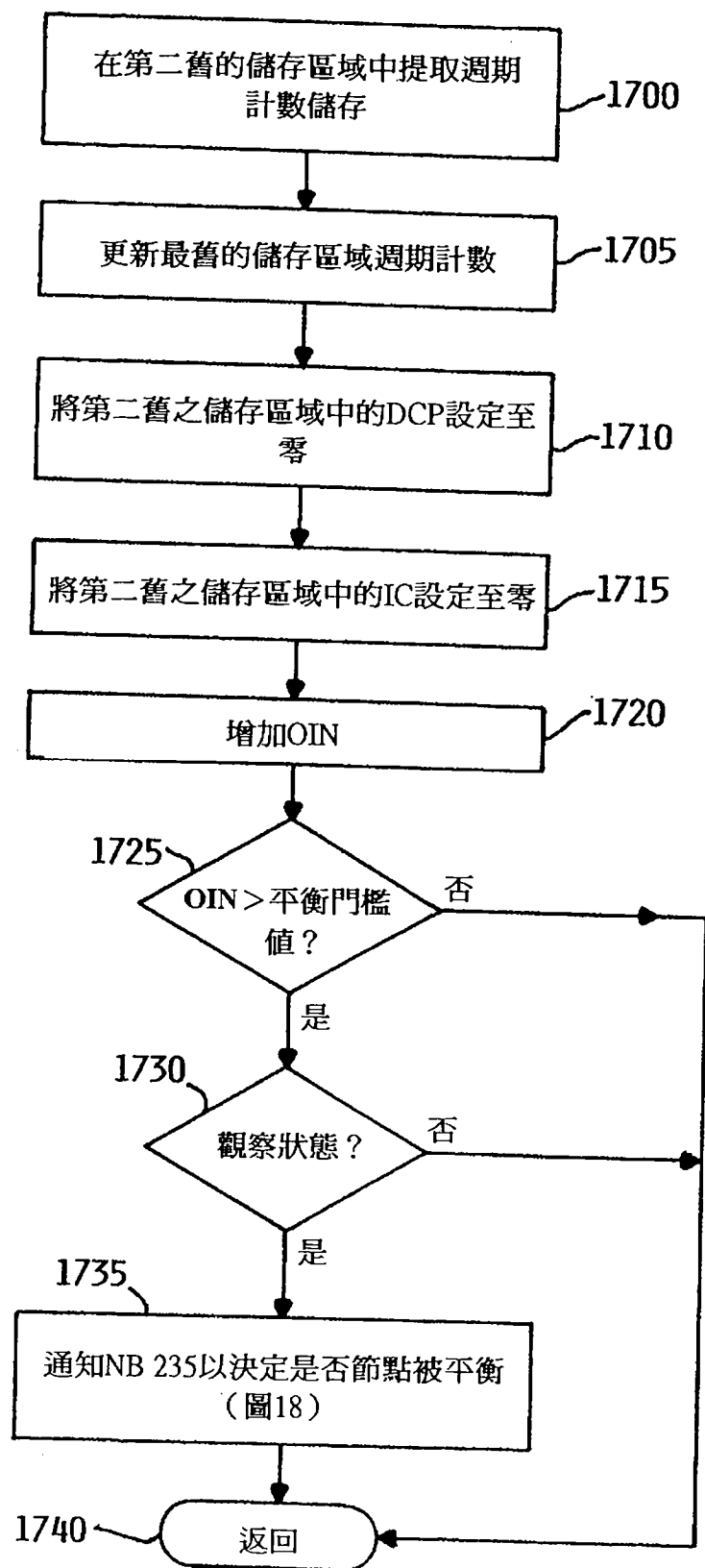


圖17

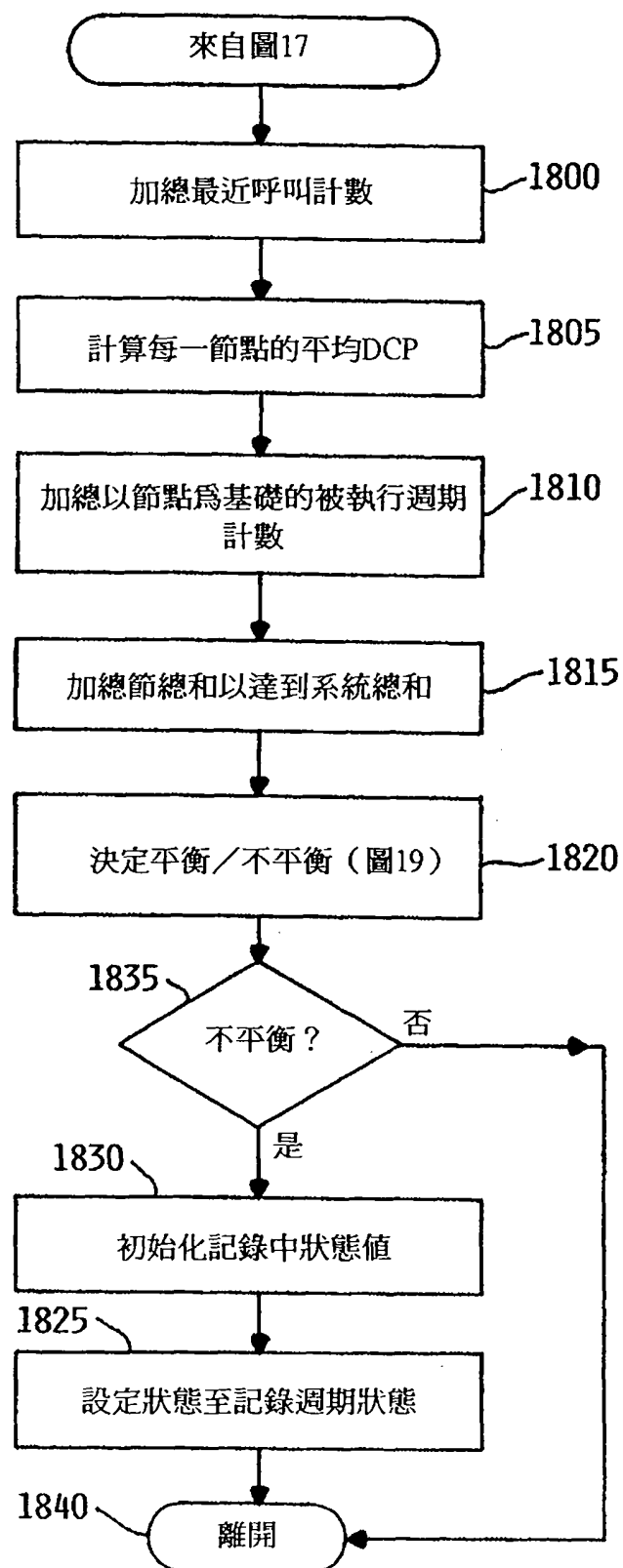


圖18

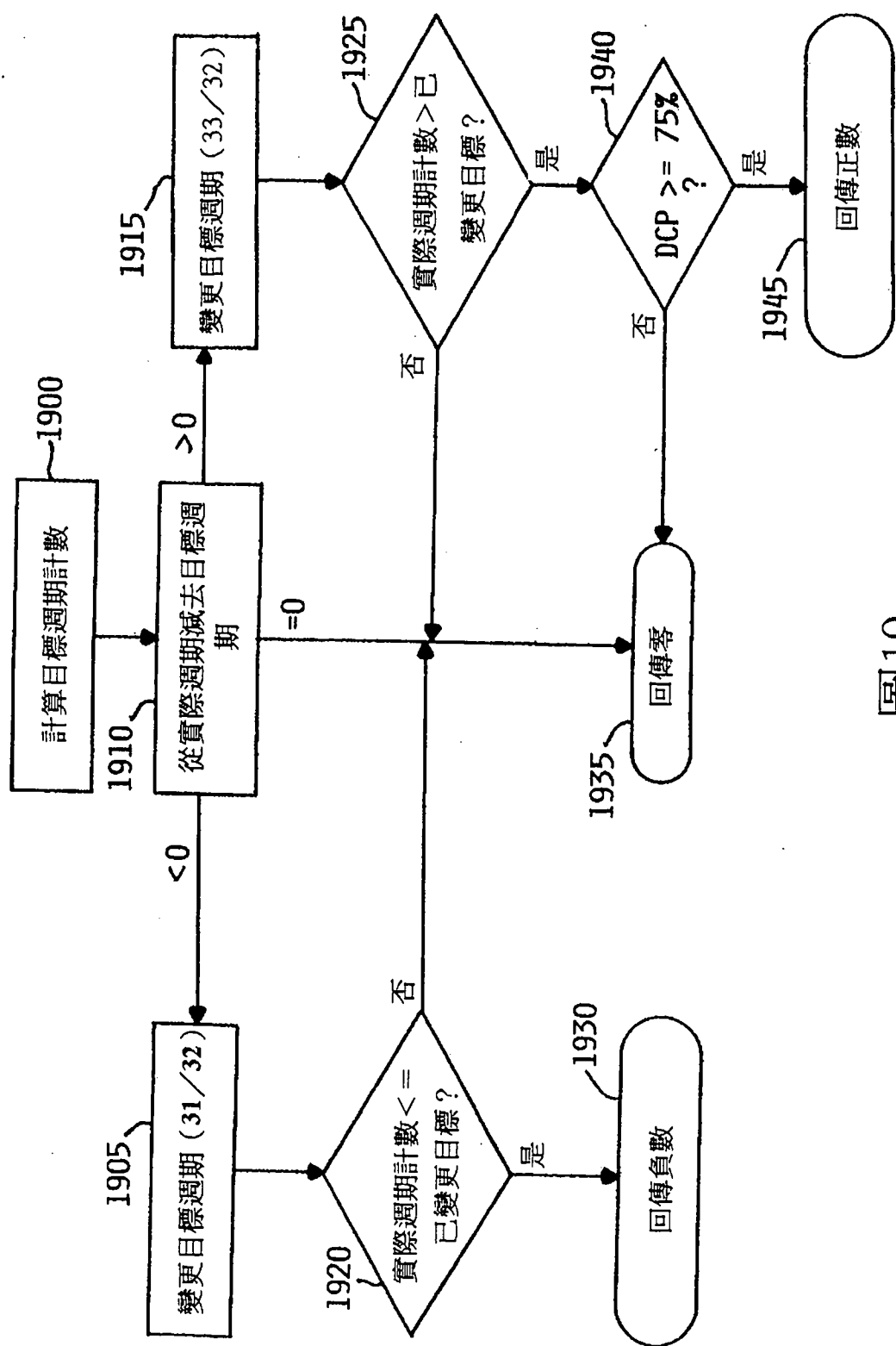


圖19

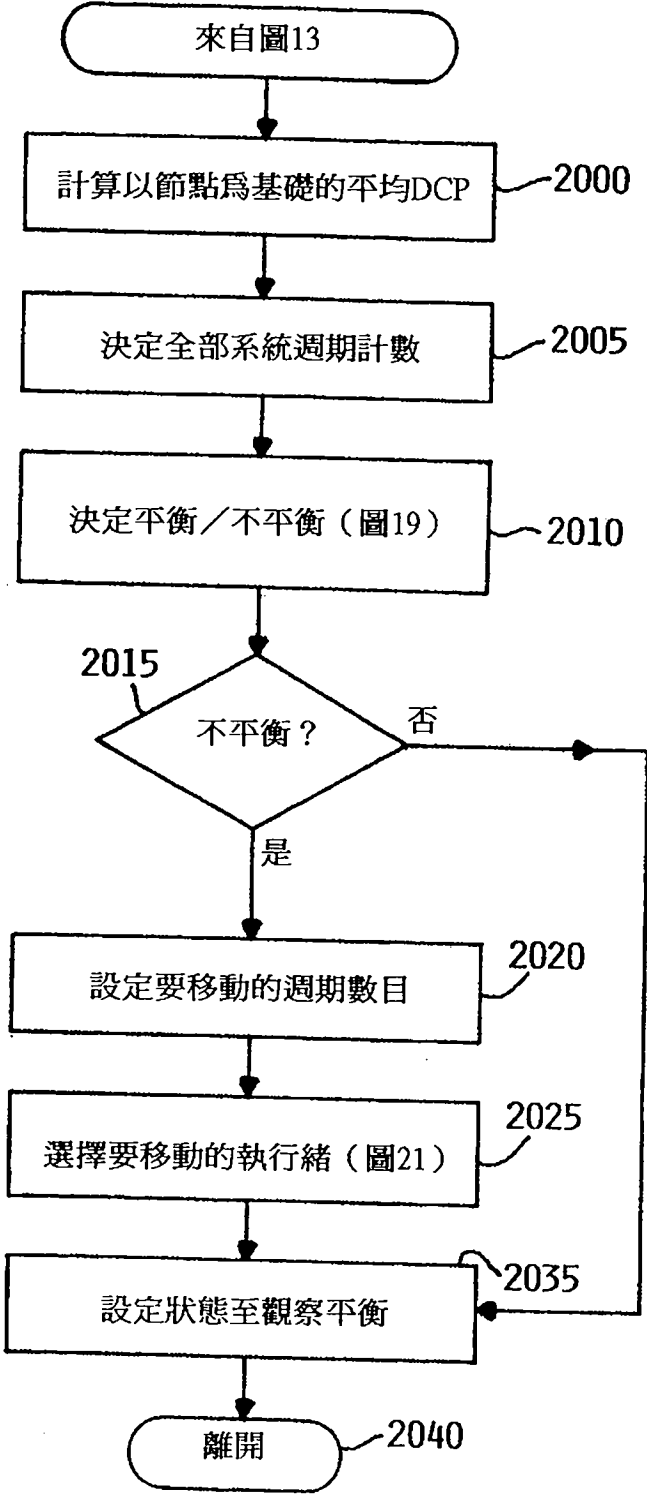


圖20

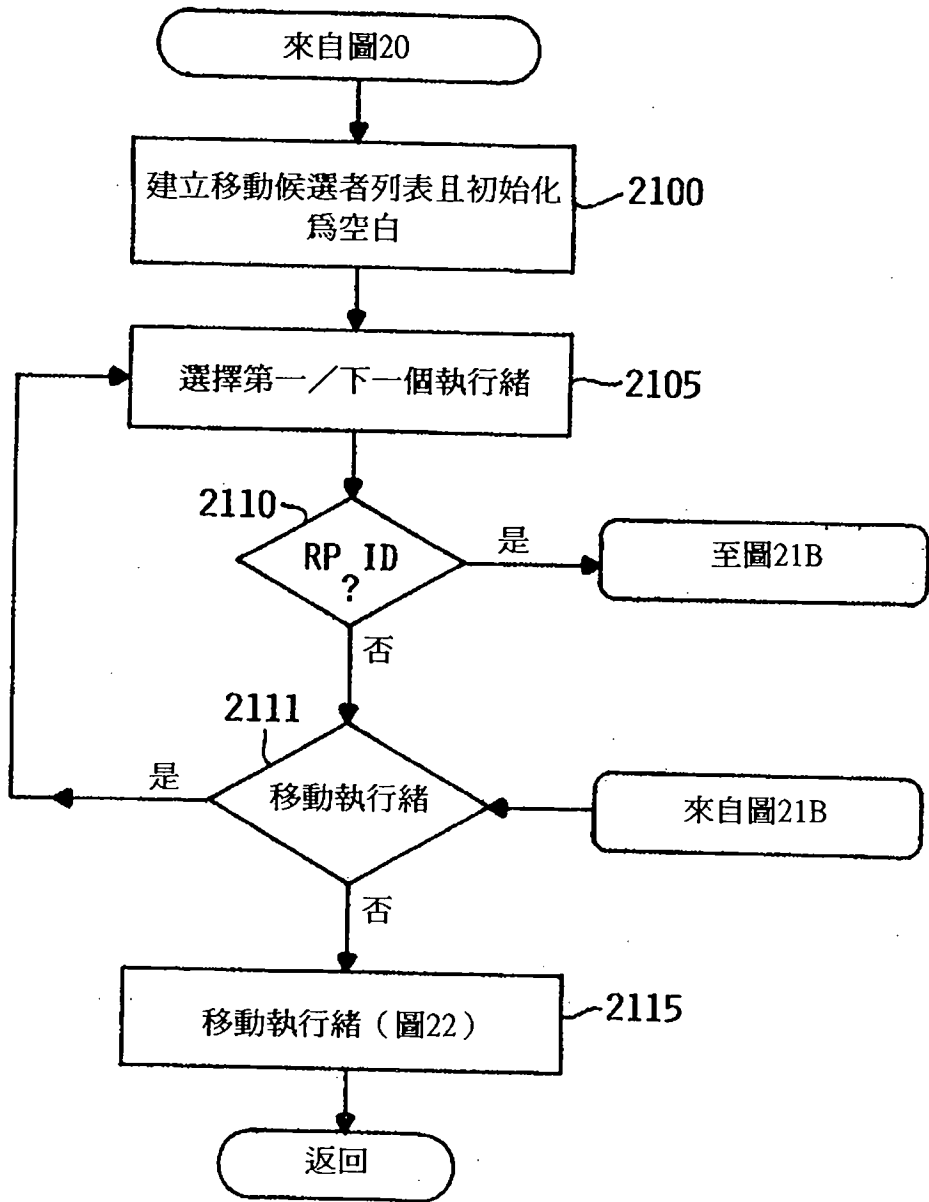


圖21A

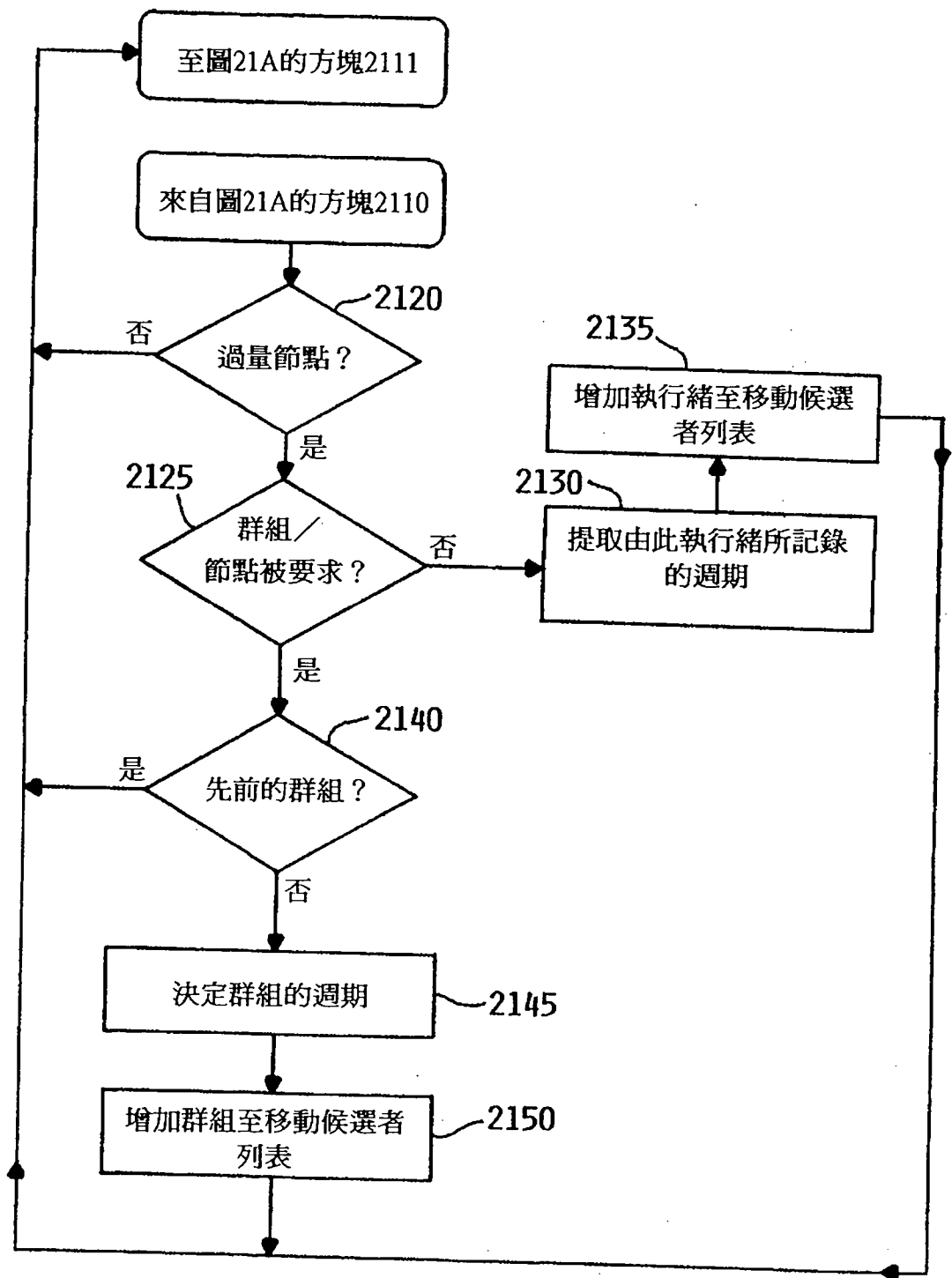


圖21B

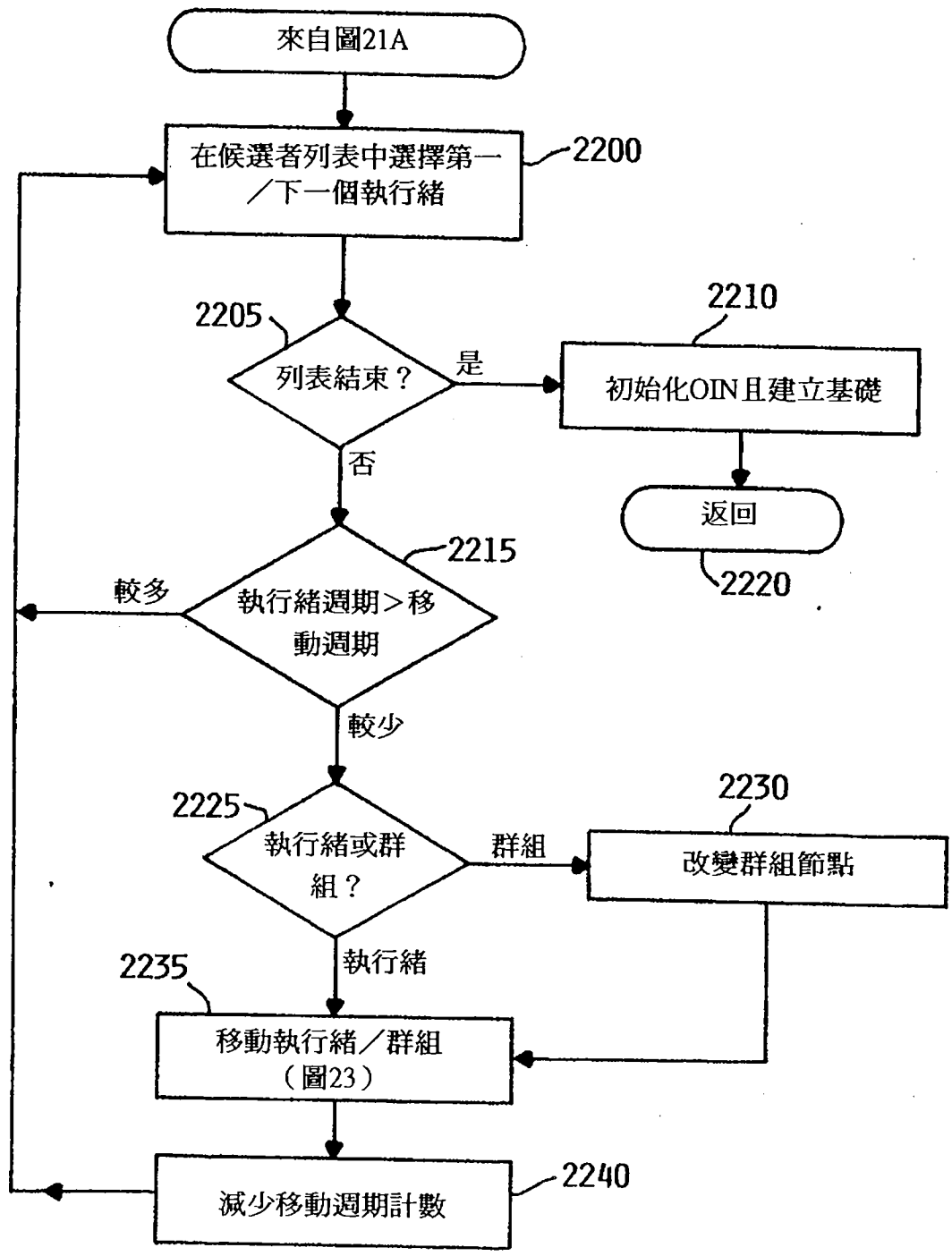


圖22

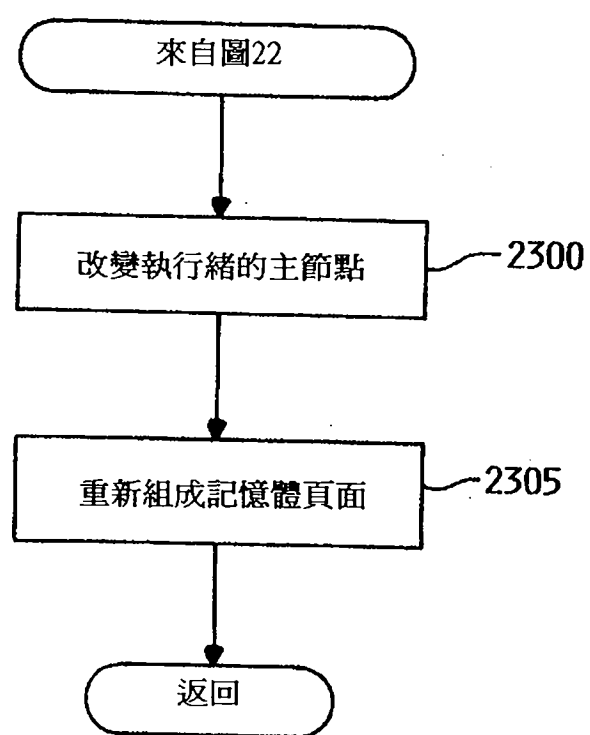


圖23

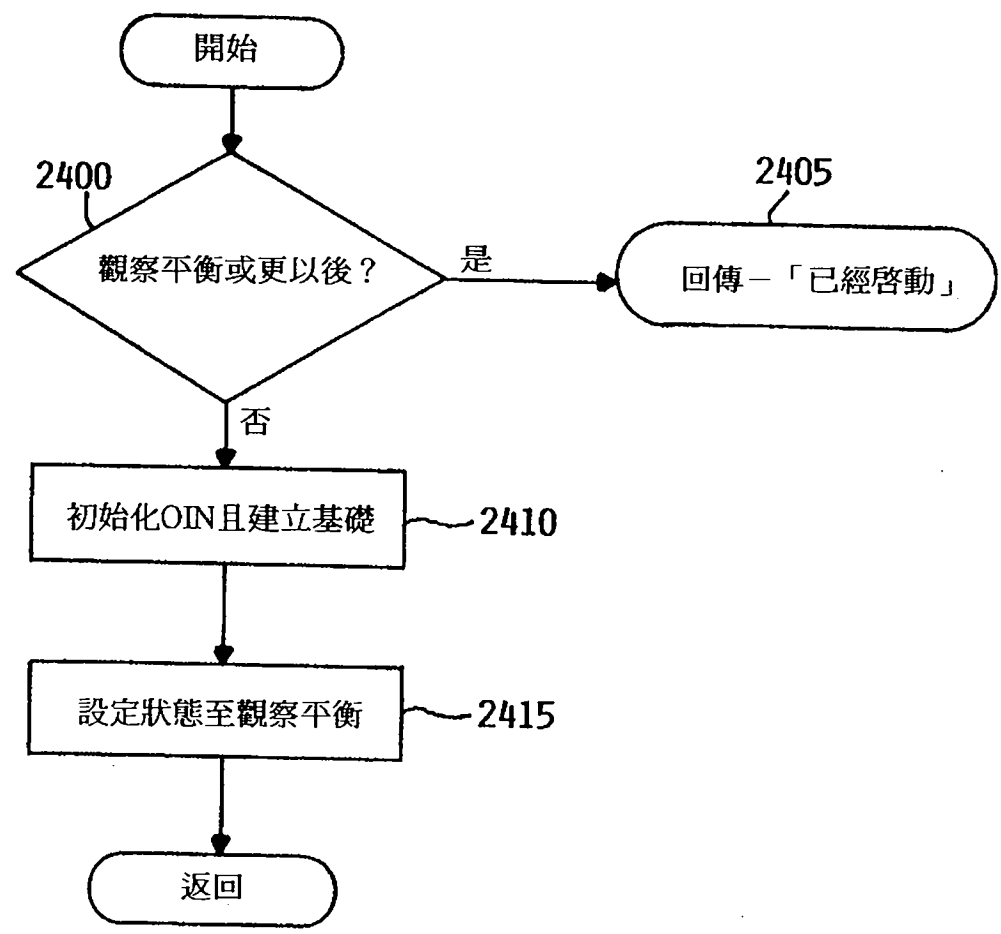


圖24

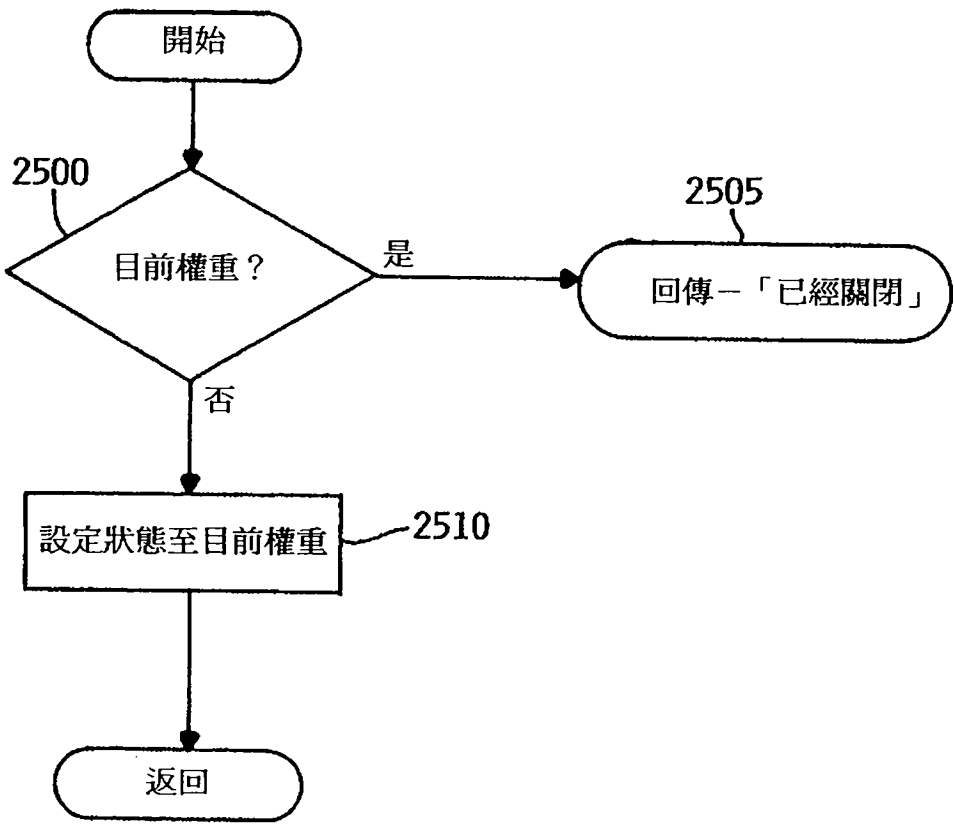


圖25

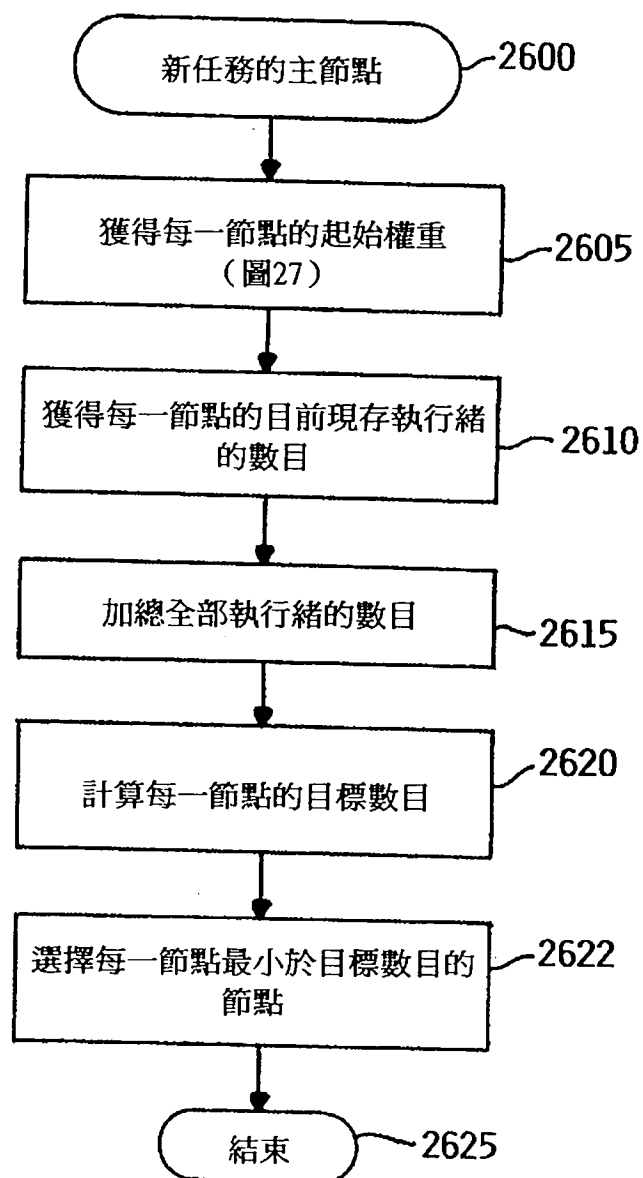


圖26

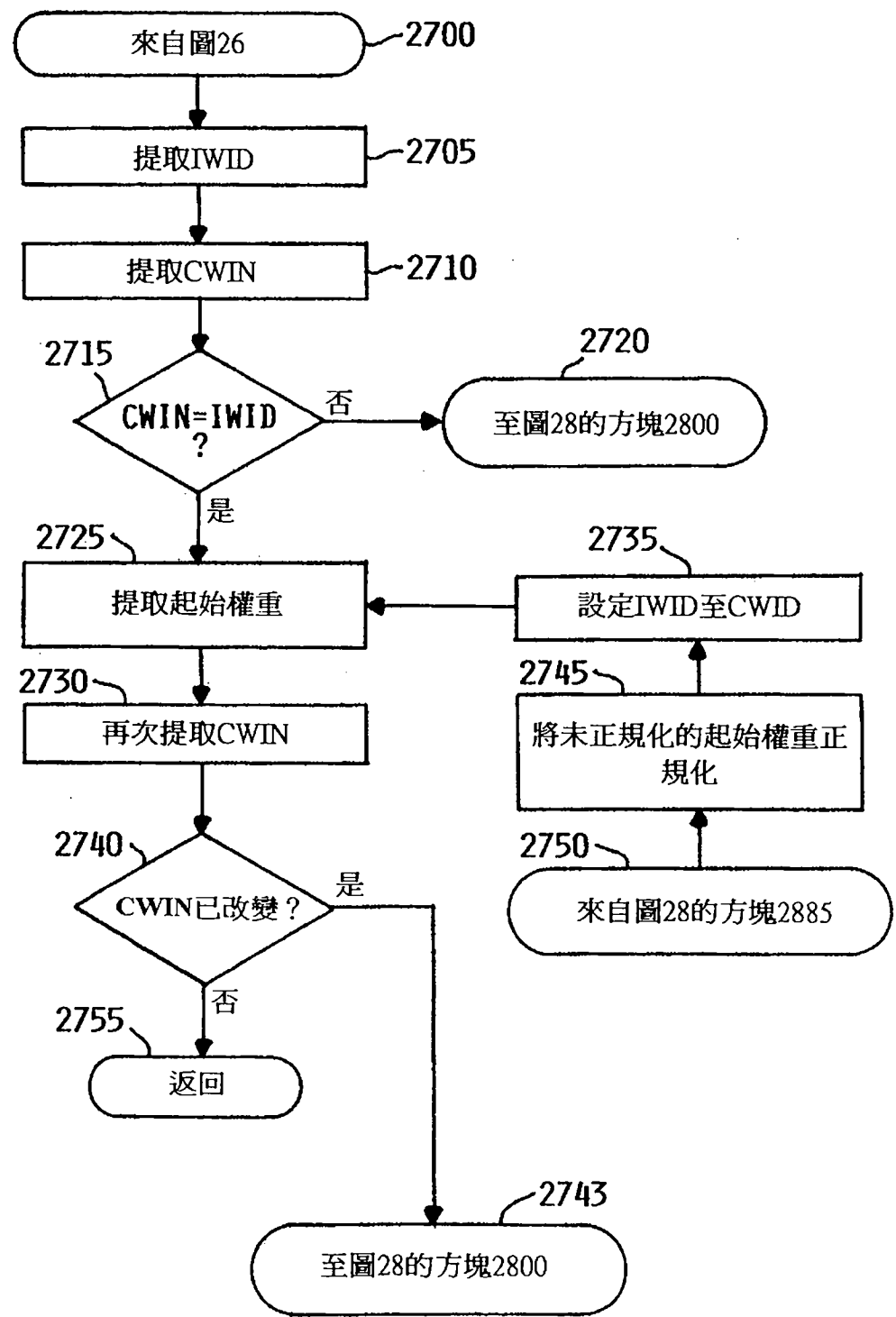


圖27

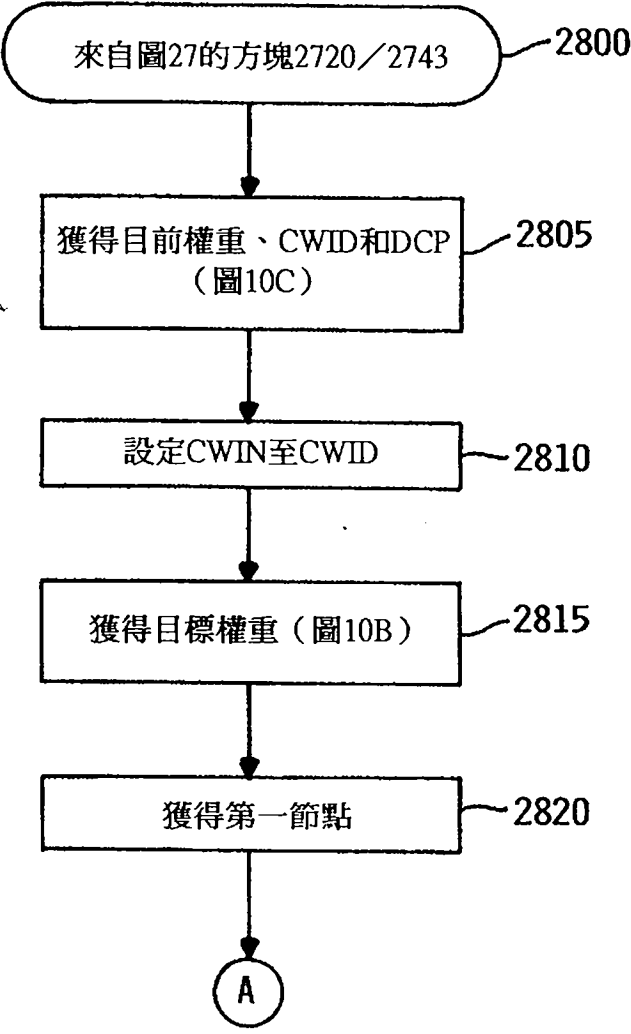


圖28A

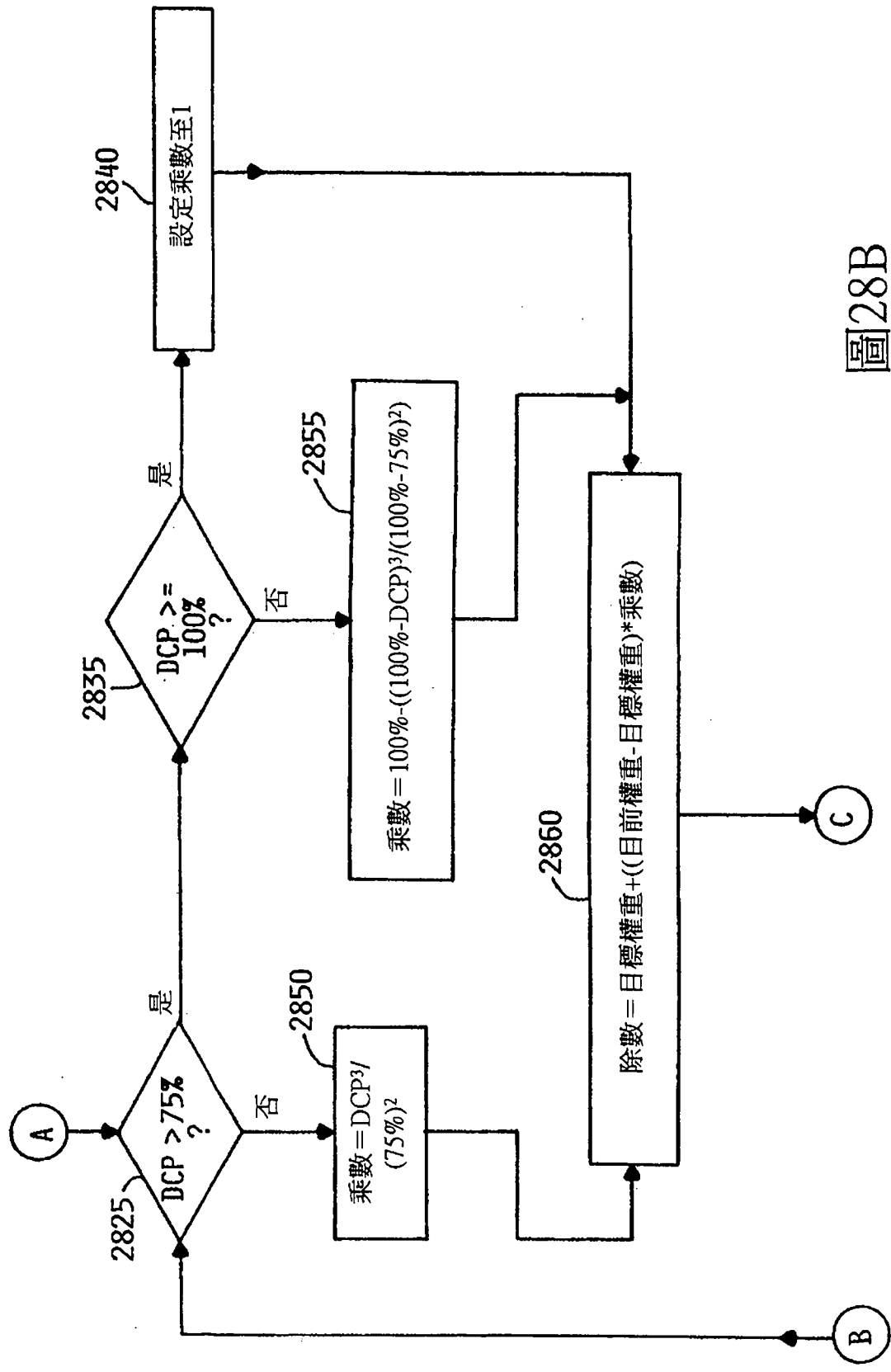


圖28B

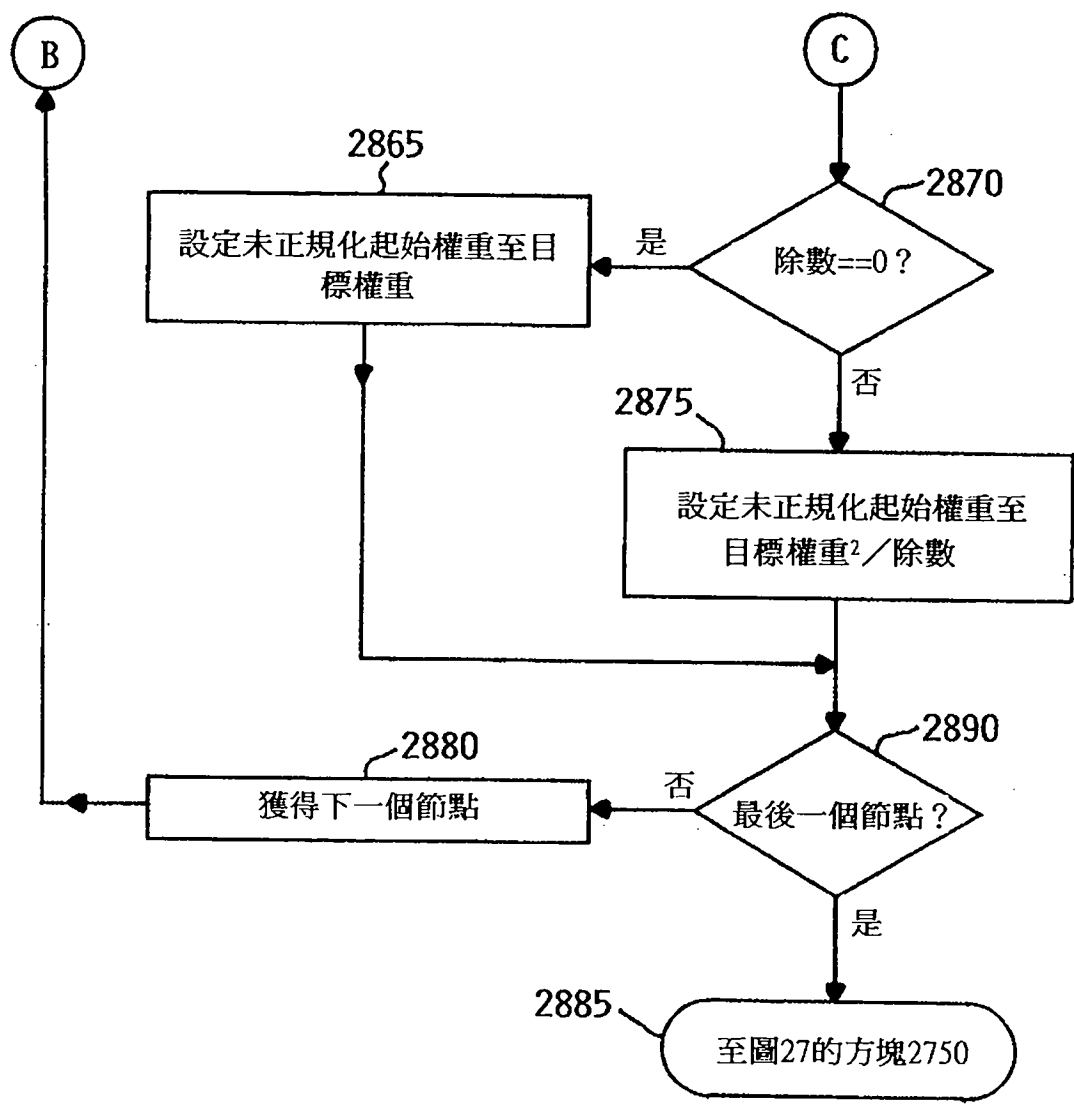


圖28C

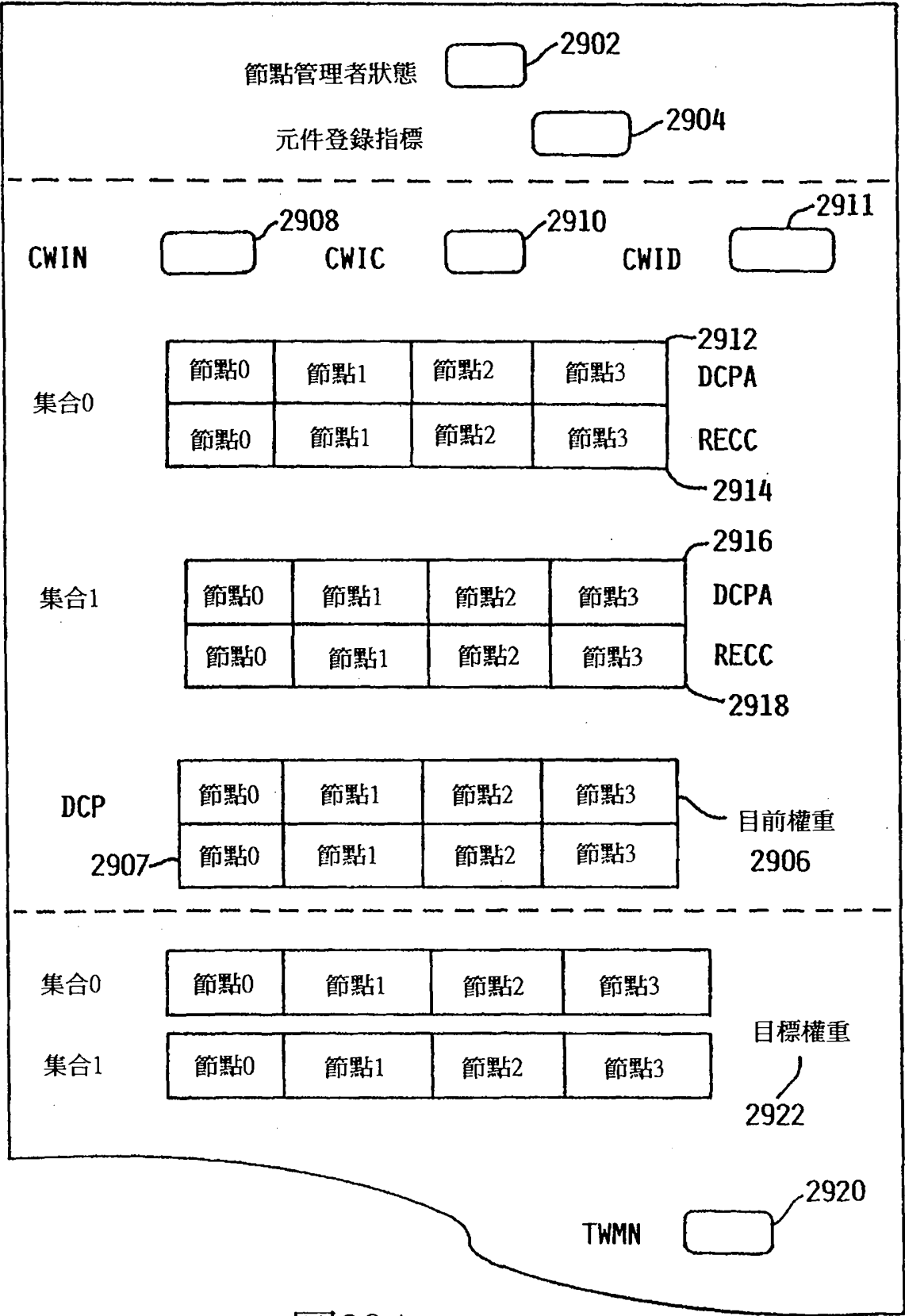


圖29A

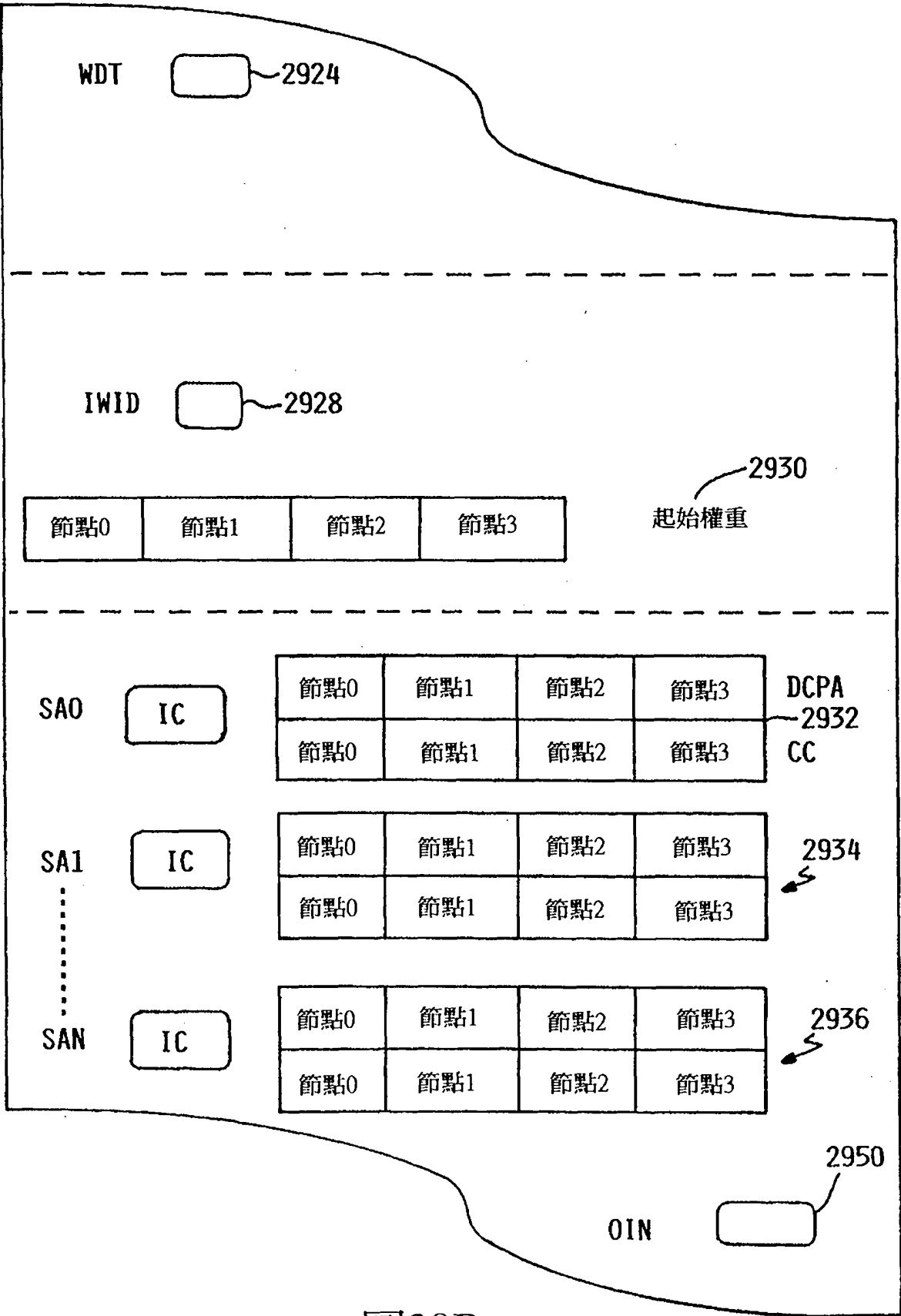


圖29B

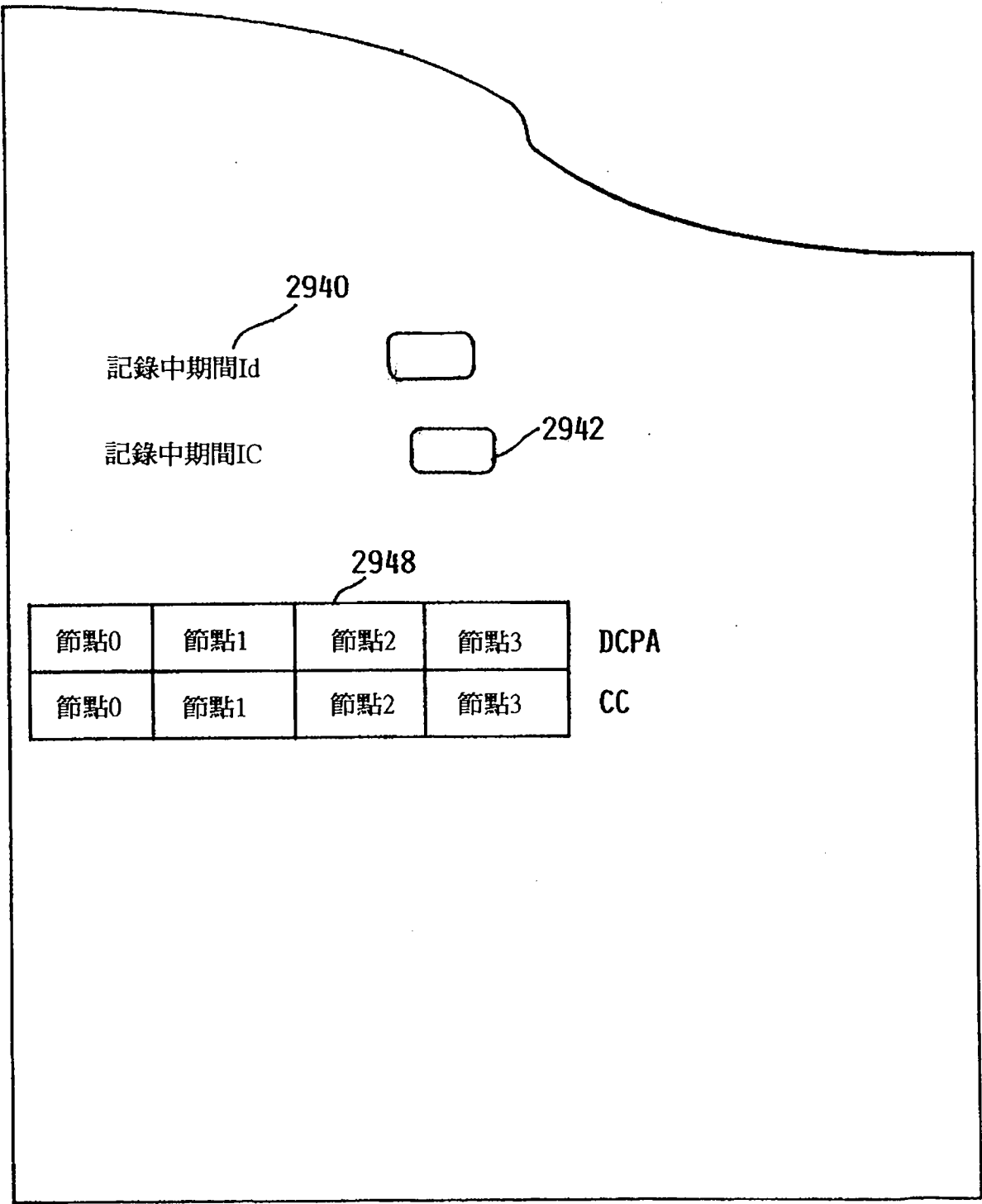


圖29C

節點錨

3005	節點識別符		
3010	節點狀態		
3015	目前目標權重		
	目前目前權重		
	目前轉換權重		
	目前記憶體量		
	目前處理器數目		
3020	先前目標權重		
	先前目前權重		
	先前轉換權重		
	先前記憶體量		
	先前處理器數目		
	資源錨		

3000

圖30

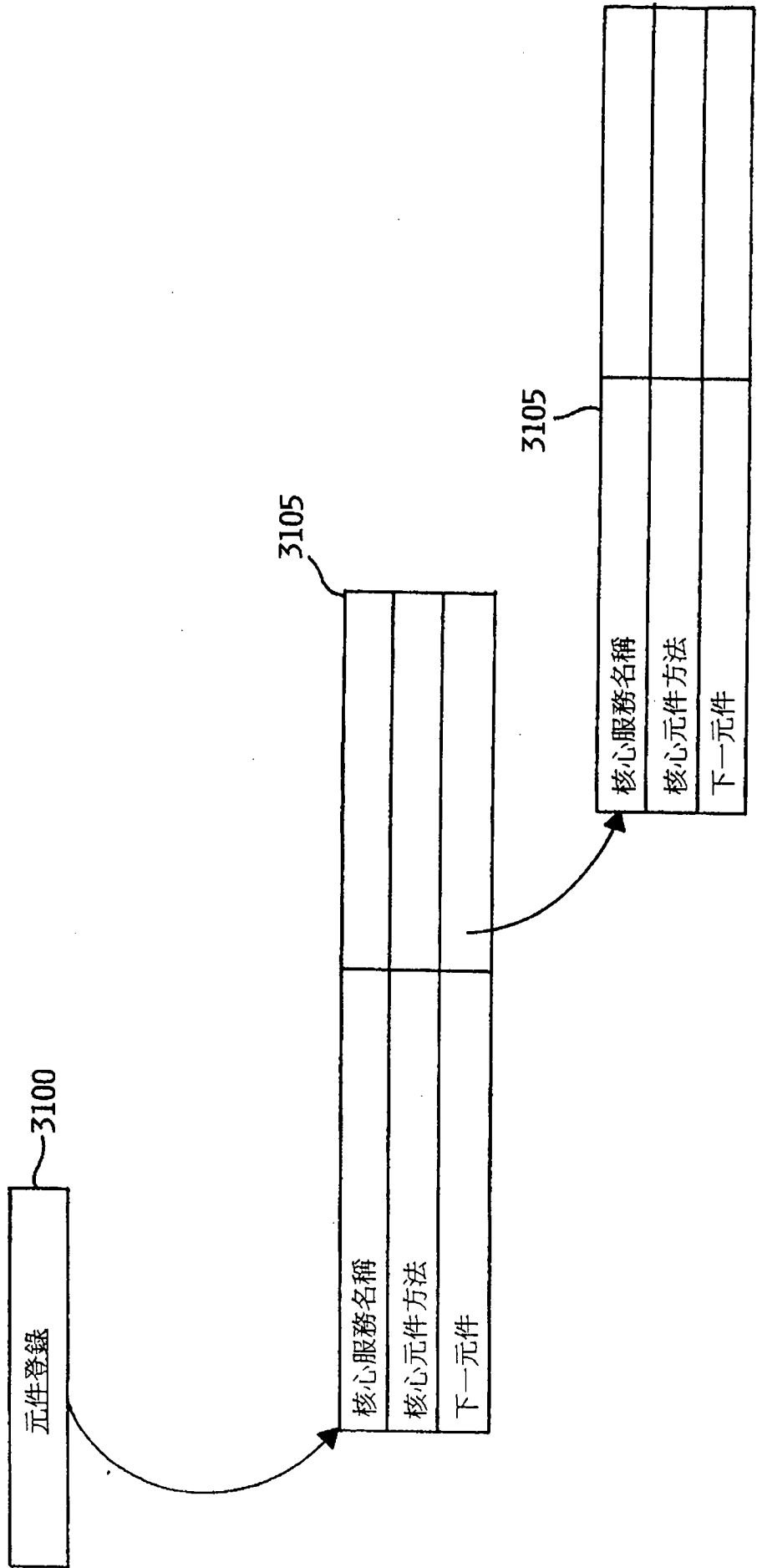


圖31

七、指定代表圖：

(一)本案指定代表圖為： 圖 1A

(二)本代表圖之元件符號簡單說明：

100	電腦裝置	125	網路介面
130	節點	131	節點
132	處理器	134	記憶體
150	使用者介面	155	大量儲存裝置

八、本案若有化學式時，請揭示最能顯示發明特徵的化學式：

十、申請專利範圍：

1. 一種以電腦實施的方法，該方法包含下列步驟：

確認與一電腦裝置之一第一處理器和一第二處理器對應之個別至少一個工作負載測量，該第一處理器關聯至一第一記憶體，以及一第二處理器關聯至一第二記憶體；其中該第一處理器和該第一記憶體形成該電腦裝置之一第一節點且其中該第二處理器和該第二記憶體形成該電腦裝置之一第二節點，其中該個別至少一個工作負載測量係由一個別第一量值與一個別第二量值得到，其中該個別第一量值代表包含該相對應處理器之該節點的一個別期望工作負載，而該個別第二量值代表包含該相對應處理器之該節點所經歷之一個別目前的工作負載，其中該個別目前的工作負載代表該電腦裝置內該工作實際分佈；以及

將該個別工作負載通知給一服務提供者，該服務提供者提供該電腦裝置之一作業系統的複數個核心服務，該複數個核心服務係由複數個服務提供者執行緒執行，每一個該服務提供者執行緒可在該第一處理器和該第二處理器任一者上執行，且與一個別之至少一個處理器關聯，每一個該服務提供者執行緒優先分配去執行該服務提供者執行緒所關聯之一處理器，該通知一服務提供者之步驟允許該服務提供者藉由依照該個別工作負載改變該複數個服務提供者執行緒之至少一個服務提供者執行緒的一處理器關聯，而調整服務提供者資源，該服務提供者提供該複數個核心服務給執行於該第一處理器和該第二處理器上的實體(entities)。

2. 如請求項1所述之方法，其中該個別期望工作負載係由該第一節點上的一第一目標權重，以及該第二節點上的一第二目

標權重所代表。

3. 如請求項 1 所述之方法，其中該個別目前的工作負載係由該第一節點上的一第一目前權重，以及該第二節點上的一第二目前權重所代表。

4. 如請求項 1 所述之方法，其中該個別期望的工作負載係由該第一節點上的一第一目標權重，以及該第二節點上的一第二目標權重所代表，且其中該個別目前的工作負載係由該第一節點上的一第一目前權重，以及該第二節點上的一第二目前權重所代表。

5. 一種以電腦實施的方法，該方法包含下列步驟：

確認與一電腦裝置之一第一處理器和一第二處理器對應的個別期望工作負載，其中該第一處理器和一第一記憶體形成該電腦裝置之一第一節點且其中該第二處理器和一第二記憶體形成該電腦裝置之一第二節點，該期望工作負載係一第一量值，其中該第一量值代表包含該相對應處理器之該節點的一個個別期望工作負載；

確認該第一處理器和包含該第二處理器之該節點所經歷的個別目前工作負載，該個別目前工作負載為代表該電腦裝置內該工作實際分佈的第二量值；

由該個別期望工作負載以及由該個別目前工作負載，得到 (derive) 一個別中介工作負載 (respective intermediate workload)；以及

將該個別中介工作負載通知給一服務提供者，該服務提供者提供該電腦裝置之一作業系統的複數個核心服務，該複數個

核心服務係由複數個服務提供者執行緒執行，每一個該服務提供者執行緒可在該第一處理器和該第二處理器任一者上執行，且與一個別之至少一個處理器關聯，每一個該服務提供者執行緒優先分配去執行該服務提供者執行緒所關聯之一處理器，該通知一服務提供者之步驟允許該服務提供者藉由依照該中介工作負載改變該複數個服務提供者執行緒之至少一個服務提供者執行緒的一處理器關聯，而調整服務提供者資源，該服務提供者提供該複數個核心服務給執行於該第一處理器和該第二處理器上的實體(entities)。

6. 如請求項 5 所述之方法，其中該個別目前工作負載係隨時間而調整，使得該個別目前工作負載更接近該個別期望工作負載，且其中該個別中介工作負載係週期地更新，以反應出對該個別目前工作負載的調整。

7. 如請求項 5 所述之方法，其中該個別期望工作負載係由該第一節點上的一目標權重，以及該第二節點上的一目標權重所代表，且其中該個別目前工作負載係由該第一節點上的一第一目前權重，以及該第二節點上的一第二目前權重所代表，且其中該個別中介工作負載係由該第一節點上的一第一轉換權重，以及該第二節點上的一第二轉換權重所代表。

8. 一種以電腦實施的方法，該方法包含下列步驟：

確認與一電腦裝置之一第一處理器和一第二處理器對應之個別至少一個工作負載測量，該第一處理器關聯至一第一記憶體，以及一第二處理器關聯至一第二記憶體；其中該個別至少一個工作負載測量係由該相對應處理器之一個別期望工作

負載以及該相對應處理器所經歷之一個別目前的工作負載得到；以及

將該個別至少一個工作負載測量通知給一服務提供者，該服務提供者提供該電腦裝置之一作業系統的複數個核心服務，該複數個核心服務係由複數個服務提供者執行緒執行，每一個該服務提供者執行緒可在該第一處理器和該第二處理器任一者上執行，且與一個別之至少一個處理器關聯，每一個該服務提供者執行緒優先分配去執行該服務提供者執行緒所關聯之一處理器；

回應該通知一服務提供者該個別至少一個工作負載測量之步驟，藉由依照該個別至少一個工作負載測量改變該複數個服務提供者執行緒之至少一個服務提供者執行緒的一處理器關聯，而調整服務提供者資源，該服務提供者提供該複數個核心服務給在該第一處理器和該第二處理器上執行的實體(entities)；其中該第一處理器和該第一記憶體形成一第一節點，且其中該第二處理器和該第二記憶體形成一第二節點，且其中該個別期望工作負載係由該第一節點上的一第一目標權重以及該第二節點上的一第二目標權重所代表，且其中該個別目前工作負載係由該第一節點的一目前權重以及該第二節點的一第二目前權重所代表。

9. 一種電腦裝置，該電腦裝置包含：

複數個處理器；

一服務提供者，提供該電腦裝置之一作業系統的複數個核心服務給在該複數個處理器上執行的實體(entities)，該複數個核心服務係由複數個服務提供者執行緒執行，每一個該服務提供者執行緒可在該些處理器任一者上執行，且與該複數個處理

器之複數個互斥（disjoint）子集合之一個別處理器子集合關聯，每一個該處理器子集合包含至少一個別處理器；

其中一第一處理器子集合與該電腦裝置之記憶體的一第一子集合形成該電腦裝置之一第一節點，且其中一第二處理器子集合與該電腦裝置之記憶體的一第二子集合形成該電腦裝置之一第二節點，記憶體的該第一與第二子集合互斥；

一工作負載管理機制，確認處理器之每一該子集合所經歷的個別目前工作負載，該個別目前工作負載為代表該電腦裝置內該工作實際分佈的第一量值，該工作負載管理機制將與每一該節點關聯的一個別至少一工作負載測量通知該服務提供者，該個別至少一工作負載測量係由一個別至少一第二量值得到，其中該個別至少一第二量值代表該相對應節點之一個別期望工作負載以及每一該個別目前工作負載；

其中，回應該工作負載管理機制所做之該個別至少一工作負載測量通知，該服務提供者依照該個別工作負載改變該複數個服務提供者執行緒之至少一個服務提供者執行緒的一處理器子集合關聯，而調整服務提供者資源；以及

一分配機制，將該些服務提供者執行緒優先分配至與每一個該服務提供者執行緒關聯之該個別處理器子集合內的處理器上執行。

10. 如請求項 9 所述之電腦裝置，其中該服務提供者根據該個別目前工作負載與該個別期望工作負載間之差異而調整該服務提供者資源。

11. 如請求項 10 所述之電腦裝置，其中該個別期望工作負載係由該第一節點的一第一目標權重(target weight)以及該第二

節點的一第二目標權重所代表。

12. 如請求項 11 所述之電腦裝置，其中該個別目前工作負載係由該第一節點的一第一目前權重以及該第二節點的一第二目前權重所代表。

13. 如請求項 9 所述之電腦裝置，其中該工作負載管理機制由該個別期望工作負載以及由該個別目前工作負載，得到 (derive) 一個別中介工作負載 (respective intermediate workload)，該工作負載管理機制將該個別中介工作負載通知給該服務提供者，該服務提供者依照該個別中介工作負載而調整服務提供者資源。

14. 如請求項 13 所述之電腦裝置，其中該個別目前工作負載係隨時間而調整，使得該個別目前工作負載更接近該個別期望工作負載，且其中該個別中介工作負載係週期地更新，以反應出對該個別目前工作負載的調整。

15. 如請求項 14 所述之電腦裝置，其中該個別期望工作負載係由該第一節點的一第一目標權重以及該第二節點的一第二目標權重所代表，且其中該個別目前工作負載係由該第一節點的一第一目前權重以及該第二節點的一第二目前權重所代表，且其中該個別中介工作負載係由該第一節點的一第一轉換權重 (transition weight) 以及該第二節點的一第二轉換權重所代表。

16. 一種程式產品，該程式產品包含：

實體 (tangible) 電腦可讀取儲存媒體；以及

一工作負載管理機制，置於該實體電腦可讀取儲存媒體上，該工作負載管理機制確認一電腦裝置之複數個處理器之複數個互斥處理器子集合之每一個處理器子集合所經歷的一個別目前工作負載，該目前工作負載為代表該電腦裝置內該工作實際分佈的第一量值，每一該處理器子集合包含至少一個別處理器，其中一第一處理器子集合與一電腦裝置之記憶體的一第一子集合形成該電腦裝置的一第一節點，且其中一第二處理器子集合與該電腦裝置之記憶體的一第二子集合形成該電腦裝置的一第二節點，記憶體的該第一與第二子集合互排斥，該工作負載管理機制將與每一該處理器子集合關聯之至少一個別工作負載測量通知給一服務提供者，該個別至少一工作負載測量係由一個別至少一第二量值得到，其中該個別至少一第二量值代表該相對應節點之工作的一個別期望共享 (share) 以及每一該個別目前工作負載，因此允許該服務提供者根據該個別至少一工作負載測量改變複數個服務提供者執行緒之至少一服務提供者執行緒的一處理器子集合關聯而調整服務提供者資源；

其中，該服務提供者提供該電腦裝置之一作業系統的複數個核心服務至在該複數個處理器上執行的實體 (entities)，該複數個核心服務係由該複數個服務提供者執行緒執行，每一該服務提供者執行緒可在該些處理器任一者上執行，且與該複數個互斥處理器子集合之一個別處理器子集合關聯，每一該服務提供者執行緒優先分配去執行該服務提供者執行緒所關聯之處理器之該個別子集合內的該至少一處理器。

17. 如請求項 16 所述之程式產品，其中該個別期望工作負載

係由該第一節點的一第一目標權重(target weight)以及該第二節點的一第二目標權重所代表。

18. 如請求項 17 所述之程式產品，其中該個別目前工作負載係由該第一節點的一目前權重以及該第二節點的一第二目前權重所代表。

19. 如請求項 16 所述之程式產品，其中該工作負載管理機制從與該處理器子集合關聯之該個別期望工作負載以及該處理器子集合所經歷之該個別目前工作負載得到與每一該處理器子集合關聯的一個別中介工作負載，且該工作負載管理機制將該個別中介工作負載通知給一服務提供者，因此允許該服務提供者根據該個別中介工作負載調整服務提供者資源。

20. 如請求項 19 所述之程式產品，其中該個別目前工作負載係隨時間而調整，使得該個別目前工作負載更接近該個別期望工作負載，且其中該個別中介工作負載係週期地更新，以反應出對該個別目前工作負載的調整。

21. 如請求項 19 所述之程式產品，其中該個別期望工作負載係由該第一節點的一第一目標權重以及該第二節點的一第二目標權重所代表，且其中該個別目前工作負載係由該第一節點的一第一目前權重以及該第二節點的一第二目前權重所代表，且其中該個別中介工作負載係由該第一節點的一第一轉換權重(transition weight)以及該第二節點的一第二轉換權重所代表。