I496007

(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 I496007 B**

(45)公告日： 中華民國 104 (2015) 年 08 月 11 日

(21)申請案號：102144215 (22)申請日： 中華民國 102 (2013) 年 12 月 03 日

(51)Int. Cl.： *G06F15/16 (2006.01)* *G06F9/455 (2006.01)*

(71)申請人：國立交通大學 (中華民國) NATIONAL CHIAO TUNG UNIVERSITY (TW)
新竹市大學路 1001 號

(72)發明人：林盈達 LIN, YING DAR (TW) ；蔡 明俊 THAI, MINH TUAN (VN) ；王志強 WANG, CHIH CHIANG (TW) ；賴源正 LAI, YUAN CHENG (TW)

(74)代理人：詹銘文；葉璟宗

(56)參考文獻：

TW	I260144	TW	201128407A
US	2007/0195356A1	US	2013/0111035A1

審查人員：高元良

申請專利範圍項數：10 項 圖式數：5 共 26 頁

(54)名稱

雙階排程方法、電腦程式產品

TWO-TIER SCHEDULING METHOD AND COMPUTER PROGRAM PRODUCT

(57)摘要

一種雙階排程方法與電腦程式產品，用以排程多個專案。排程規劃包括第一專案，第一專案包括第一工作。此雙階排程方法包括：根據第一專案的運轉時間與離開時間、第一工作的服務時間、以及鬆弛因子來計算第一工作的最晚開始時間；取得包括多個第二工作的第二專案；對於每一個第二工作，取得至少一個回填時間；對於每一個第二工作，從對應的回填時間中挑選一個回填時間，使得被延遲的第一工作的最晚開始時間大於等於第一工作的開始時間；以及將每一個第二工作在選擇的第一回填時間加入至排程規劃。藉此可以減少平均的運轉時間。

A two-tier scheduling method and a computer program product are provided for scheduling projects. A scheduling plan includes a first project which has at least one first job. The method includes: calculating a latest starting time of each first job according to a turn-around time and depart time of the first project, a service time of the first job and a slack factor; obtaining a second project having multiple second jobs; for each second job, obtaining at least one backfilling time; for each second job, selecting one of the corresponding backfilling times, such that the latest starting time of delayed first job is greater or equal to a starting time of the first job; adding each second job into the scheduling plan at the selected backfilling time. Consequently, the mean turn-around time is decreased.

110・・・雲端系統

120・・・專案

122、124・・・工作

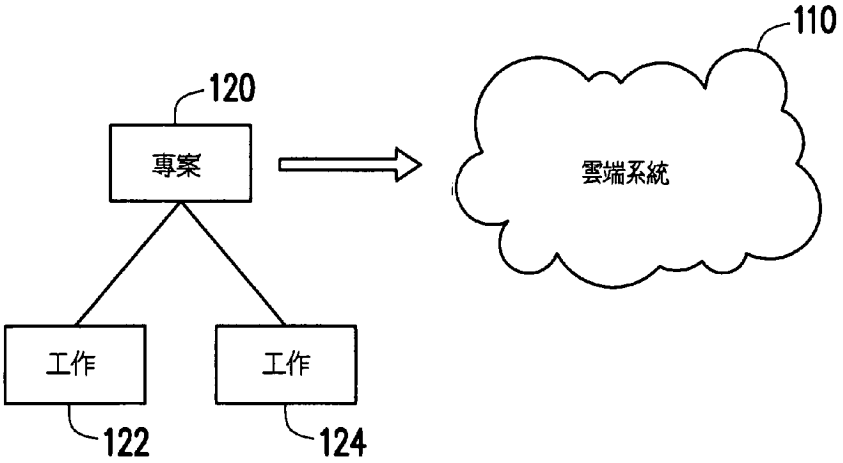


圖1A

I496007

公告本

發明摘要

※ 申請案號：102144715

※ 申請日：102.12.03 ※IPC 分類：G06F 15/16 (2006.01)
9/455 (2006.01)

【發明名稱】雙階排程方法、電腦程式產品 / TWO-TIER SCHEDULING METHOD AND COMPUTER PROGRAM PRODUCT

【中文】

一種雙階排程方法與電腦程式產品，用以排程多個專案。排程規劃包括第一專案，第一專案包括第一工作。此雙階排程方法包括：根據第一專案的運轉時間與離開時間、第一工作的服務時間、以及鬆弛因子來計算第一工作的最晚開始時間；取得包括多個第二工作的第二專案；對於每一個第二工作，取得至少一個回填時間；對於每一個第二工作，從對應的回填時間中挑選一個回填時間，使得被延遲的第一工作的最晚開始時間大於等於第一工作的開始時間；以及將每一個第二工作在選擇的第一回填時間加入至排程規劃。藉此可以減少平均的運轉時間。

【英文】

A two-tier scheduling method and a computer program product are provided for scheduling projects. A scheduling plan includes a first project which has at least one first job. The method includes: calculating a latest starting time of each first job according to a turn-around time and depart time of the first project, a service time of the first job and a slack factor; obtaining a second project having multiple second jobs; for each second job, obtaining at

least one backfilling time; for each second job, selecting one of the corresponding backfilling times, such that the latest starting time of delayed first job is greater or equal to a starting time of the first job; adding each second job into the scheduling plan at the selected backfilling time. Consequently, the mean turn-around time is decreased.

【代表圖】

【本案指定代表圖】：圖 1A。

【本代表圖之符號簡單說明】：

110：雲端系統

120：專案

122、124：工作

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

無

least one backfilling time; for each second job, selecting one of the corresponding backfilling times, such that the latest starting time of delayed first job is greater or equal to a starting time of the first job; adding each second job into the scheduling plan at the selected backfilling time. Consequently, the mean turn-around time is decreased.

【代表圖】

【本案指定代表圖】：圖 1A。

【本代表圖之符號簡單說明】：

110：雲端系統

120：專案

122、124：工作

【本案若有化學式時，請揭示最能顯示發明特徵的化學式】：

無

發明專利說明書

【發明名稱】雙階排程方法、電腦程式產品 / TWO-TIER SCHEDULING METHOD AND COMPUTER PROGRAM PRODUCT

【技術領域】

【0001】 本發明是有關於一種排程方法，且特別是有關於一種雙階排程方法與電腦程式產品。

【先前技術】

【0002】 雲端運算是用以讓使用者將運算透過網路傳遞到一個雲端系統上，藉此使用者端可以減少運算的設備。雲端運算的一個好處是可以分享運算或儲存等相關資源，並且可以根據需要來分配這些資源，以達到經濟規模而降低成本。當有多個使用者將運算傳遞到雲端系統上時，雲端系統會排程這些運算的順序。在一些情況下為了有更好的資源利用，已經進入排程的工作可能被另外一個工作所延遲。然而，被延遲的工作會因此有更高的運轉(turn-around)時間。因此，如何適當的排程，以降低平均的運轉時間，為此領域技術人員所關心的議題。

【發明內容】

【0003】 本發明提供一種雙階排程方法與電腦程式產品，可以減少平均的運轉時間。

【0004】 本發明一範例實施例提出一種雙階排程方法，用於一電腦以排程多個專案。排程規劃包括第一專案，第一專案包括第一

工作。此雙階排程方法包括：根據第一專案的運轉時間、第一專案的離開時間、第一工作的服務時間、與鬆弛因子來計算第一工作的最晚開始時間；取得第二專案，其中第二專案包括多個第二工作；對於每一個第二工作，取得至少一個回填時間；對於每一個第二工作，從對應的回填時間中挑選第一回填時間，使得被延遲的第一工作的最晚開始時間大於等於第一工作的開始時間；以及將每一個第二工作在對應的第一回填時間加入至排程規劃。

【0005】 在一範例實施例中，上述計算第一工作的最晚開始時間的步驟包括：將第一專案的運轉時間乘上鬆弛因子以取得第一數目；以及將第一專案的離開時間加上第一數目，再減去第一工作的服務時間以取得第一工作的最晚開始時間。

【0006】 在一範例實施例中，在將每一個第二工作在對應的第一回填時間加入至排程規劃以後，雙階排程方法更包括：對於每一個第二工作，根據第二專案的運轉時間、第二專案的離開時間、第二工作的服務時間、與鬆弛因子來更新第二工作的最晚開始時間。

【0007】 在一範例實施例中，上述的雙階排程方法更包括：若可被延遲第一工作的數目大於 1，從第一工作中挑選並延遲有最大的最晚開始時間的第一工作。

【0008】 在一範例實施例中，上述挑選第一回填時間步驟更包括：從所述的回填時間中挑選第一回填時間，使得被延遲的專案的個數小於一搶佔限制個數。

【0009】 在一範例實施例中，上述的雙階排程方法更包括：判斷第二專案是屬於高權限專案或是低權限專案；若第二專案是屬於低權限專案，對每一個第二工作執行第一程序。第一程序包括：判斷第二工作是否可立即被執行；若第二工作可立即被執行，則執行第二工作；以及若第二工作不可立即被執行，在一開始時間將第二工作加入至排程規劃，使得排程規劃中沒有工作被延遲。

【0010】 在一範例實施例中，上述的雙階排程方法更包括：若第二專案是屬於低權限專案，在對每一個第二工作執行第一程序以後，根據第二專案的一權限來更新鬆弛因子；以及根據更新後的鬆弛因子來更新每一個第二工作的最晚開始時間。

【0011】 在一範例實施例中，上述的雙階排程方法更包括：若第二專案是屬於高權限專案，根據第二專案的權限來更新鬆弛因子；以及根據更新後的鬆弛因子執行所述挑選第一回填時間的步驟。

【0012】 在一範例實施例中，上述根據第二專案的權限更新鬆弛因子的步驟包括：將一權重變數減去第二專案的權限以取得第三數目；以及將第三數目乘上鬆弛因子以取得更新後的鬆弛因子。

本發明一範例實施例提出一種電腦程式產品，被電腦載入以執行多個步驟以排程多個專案。排程規劃包括第一專案，第一專案包括第一工作。上述的步驟包括：根據第一專案的運轉時間、第一專案的離開時間、第一工作的服務時間、與鬆弛因子來計算第一工作的最晚開始時間；取得第二專案，其中第二專案包括多個第

二工作；對於每一個第二工作，取得至少一個回填時間；對於每一個第二工作，從對應的回填時間中挑選第一回填時間，使得被延遲的第一工作的最晚開始時間大於等於第一工作的開始時間；以及將每一個第二工作在對應的第一回填時間加入至排程規劃。

【0013】 基於上述，本發明範例實施例提出的雙階排程方法與電腦程式產品可以設定鬆弛因子與專案的權限，藉此設定工作的最晚開始時間並根據此最晚開始時間來回填工作。藉此可以減少平均的運轉時間。

【0014】 為讓本發明的上述特徵和優點能更明顯易懂，下文特舉實施例，並配合所附圖式作詳細說明如下。

【圖式簡單說明】

【0015】

圖 1A 是根據一範例實施例繪示雲端系統的示意圖。

圖 1B 是根據一範例實施例繪示執行工作的示意圖。

圖 1C 與圖 1D 是根據一範例實施例繪示回填工作的示意圖。

圖 2 是根據一範例實施例繪示雙階嚴格回填演算法的虛擬碼。

圖 3 是根據一範例實施例繪示雙階彈性回填演算法的虛擬碼。

圖 4 是根據一範例實施例繪示雙階權重回填演算法的虛擬碼。

圖 5 是根據一範例實施例繪示排程規劃的資料結構的示意圖。

【實施方式】

【0016】 圖 1A 是根據一範例實施例繪示雲端系統的示意圖。

【0017】 請參照圖 1A，雲端系統 110 會接收專案 120，其中專案 120 會包括多個工作(例如，工作 122 與工作 124)。雲端系統 110 中會包括一或多個電腦、伺服器、儲存單元、處理器或虛擬機器，本發明並不限制雲端系統 110 中的硬體或軟體架構。

【0018】 雲端系統 110 中包括了多種不同的資源，並且每一種資源的個數可能相同或是不相同。例如，虛擬機器便是一種資源，且雲端系統 110 中可能包括多個虛擬機器。當接收了專案 120，雲端系統 110 會檢視工作 122 與 124 所需要的資源種類與個數，並且分配適當的資源給工作 122 與 124。圖 1B 是根據一範例實施例繪示執行工作的示意圖。請參照圖 1B，當專案 120 抵達雲端系統 110，專案 120 會被加入至專案佇列 140 當中，並且工作 122、124 會被加入至工作佇列 150。雲端系統 110 會撿視目前可用的資源，藉此從工作佇列 150 中挑選並執行一或多個工作。具體來說，在此假設雲端系統 110 共有 N 種資源，其中第 i 種資源有 m_i 份，i 為正整數。在圖 1B 中是以 $r_{1,2}$ 表示第 1 種資源中的第 2 份，以此類推，而形狀不同的資源是代表不同的種類。每一份資源在同一時間只能分配給某一個工作。例如，圖 1B 中工作 141 佔據了資源

$r_{1,1}$、資源 $r_{1,2}$、以及資源 $r_{N,1}$；而工作 151 佔據了資源 $r_{1,3}$與 $r_{2,2}$，因此這些被佔據的資源無法分配給其他的工作。在圖 1B 的實施例中，工作 141 與 151 已經得到了所需要的資源，即工作 141 與 151 是執行中的工作。值得注意的是，雖然工作佇列 150 中的工作尚未被執行，但雲端系統 110 可預先安排資源，藉此將工作加入至一個排程規劃中(scheduling plan)。

【0019】 圖 1C 與圖 1D 是根據一範例實施例繪示回填工作的示意圖。請參照圖 1C，在此以某一種資源為例，但值得注意的是一個工作可能需要多種資源。圖 1C 中橫軸為時間，縱軸為資源。在目前的排程規劃中具有工作 131~133，圖 1C 繪示了在不同的時間內各工作所需要的資源個數。在雲端系統 110 接收了工作 122 以後，可以根據目前的排程規劃試著將工作 122 回填(backfilling)。例如，如圖 1D 所示，雖然工作 122 是新接收的，但雲端系統 110 可將工作 122 安排在工作 132 之前，但卻不會影響工作 132 的執行。以下將說明如何將工作加入至排程規劃中。

【0020】 首先，定義 $P=\{p_u \mid 1 \le u \le |P|\}$為多個專案所組成的集合。專案 p_u是在送達時間 ta_u被傳遞到雲端系統 110。此外，專案 p_u所包括的所有工作定義為集合 J_u，其中 $J_u=\{j_{u,v} \mid 1 \le v \le |J_u|\}$。工作 $j_{u,v}$具有服務時間 $te_{u,v}$，表示處理工作 $j_{u,v}$所需要的時間。雲端系統 110 開始處理工作 $te_{u,v}$的時間定義為開始時間 $ts_{u,v}$。而雲端系統 110 完成工作 $j_{u,v}$的時間定義為完成時間 $tf_{u,v}$。在此假設每一個專案被傳遞到雲端系統 110 時，其中的每一個工作所需要的服務時間與資源都可事先

估測出。例如，若雲端系統 110 曾經執行過一個工作，雲端系統 110 便可以得知處理此工作需要多少服務時間以及需要哪些資源。然而，本發明並不限制如何得知工作所需要的服務時間與資源。此外，工作之間具有非程序(non-precedence)的限制，也就是說雲端系統 110 可以用任意的順序來執行這些工作。在此也假設工作是不能被搶佔(preempt)的，即一旦工作開始執行，直到該工作執行完畢，該工作都不能被停止。然而，在其他範例實施例中，工作之間可以不具有非程序限制，或者是部分的工作是可以被搶佔，本發明並不在此限。

【0021】 在此，專案 p_u 的開始時間 tc_u 是以下列方程式(1)來定義，表示專案 p_u 開始被處理的時間。

$$tc_u = Min(J_u.ts), J_u.ts = \{ts_{u,v} \mid 1 \le v \le |J_u|\} \ldots (1)$$

【0022】 專案 p_u 的離開時間 td_u 是以下列方程式(2)來定義，表示專案 p_u 被處理完畢的時間。

$$td_u = Max(J_u.tf), J_u.tf = \{tf_{u,v} \mid 1 \le v \le |J_u|\} \ldots (2)$$

【0023】 專案 p_u 的等待時間 tw_u 是以下列方程式(3)來定義，表示從專案 p_u 的送達時間 ta_u 到專案 p_u 被處理所經過的時間。

$$tw_u = tc_u - ta_u \ldots (3)$$

【0024】 專案 p_u 的執行時間 tr_u 是以下列方程式(4)來定義，表示從開始時間 tc_u 到離開時間 td_u 所經過的時間。

$$tr_u = td_u - tc_u \ldots (4)$$

【0025】 專案 p_u 的運轉時間 tn_u 是以下列方程式(5)來定義，表示從

送達時間 ta_u 到離開時間 td_u 所經過的時間。

$$tn_u = td_u - ta_u \ldots (5)$$

【0026】 當使用者將專案傳送給雲端系統 110 時，雲端系統 110 都會給使用者此專案預定的離開時間。基本上，該專案必須要在預定的離開時間之前被處理完畢。在此範例實施例中，雲端系統 110 還會設定一個鬆弛因子 SF，使得專案可以稍微地被延遲，其中鬆弛因子可為實數。具體來說，雲端系統 110 會將專案的運轉時間乘上鬆弛因子以取得一個乘數，並且將此乘數加上離開時間以取得延遲的上限。換言之，專案真正的離開時間可以在範圍 $[td_u, td_u + tn_u \cdot SF]$ 之間。另一方面，由於專案中所有的工作都要被處理完畢以後，該專案才算是處理完畢，因此可以計算出專案中每一個工作的最後開始時間 $lts_{u,v}$，可定義為下列方程式(6)。

$$lts_{u,v} = td_u + tn_u \cdot SF - te_{u,v} \ldots (6)$$

【0027】 以下依照是否可搶佔(preemptive)，專案是否有不同的權重，討論四種排程方法。在此假設排程規劃中的專案與工作是已經分配資源且已決定開始時間的工作。排程規劃中有一第一專案，並且第一專案有至少一個第一工作。對於每一個第一工作，雲端系統 110 會根據第一專案的運轉時間、第一專案的離開時間、第一工作的服務時間、與鬆弛因子來計算第一工作的最晚開始時間。具體來說，雲端系統 110 會將第一專案的運轉時間乘上鬆弛因子以取得第一數目，將第一專案的離開時間加上第一數目，再減去第一工作的服務時間以取得第一工作的該最晚開始時間(如

以上方程式(6))。

【0028】 首先討論若不能搶佔且專案只有一種權重的排程方法，稱為雙階嚴格回填演算法。在此演算法中，專案的離開時間 td_u 與工作的完成時間 $tf_{u,v}$ 都會在專案 p_u 抵達雲端系統 110 時被計算出。舉例來說，圖 2 是根據一範例實施例繪示雙階嚴格回填演算法的虛擬碼。請參照圖 2，假設雲端系統 110 接收到了一個第二專案，標記為 p_u。第二專案包括多個第二工作，標記為 $j_{u,v}$。在圖 2 的第 3-4 行中，雲端系統 110 會判斷每一個第二工作是否可立即被執行。若目前有足夠的資源以立即執行第二工作，雲端系統 110 會執行第二工作。在第 5-8 行中，若第二工作不可立即被執行，雲端系統 110 在一開始時間將第二工作 $j_{u,v}$ 加入至排程規劃，使得排程規劃中沒有工作被延遲。此外，由於第二工作 $j_{u,v}$ 也不能被搶佔，因此第二工作 $j_{u,v}$ 的開始時間與最晚開始時間相同。

【0029】 第二種排程方法是可以搶佔且專案只有一種權重的排程方法，稱為雙階彈性回填演算法，其中設定鬆弛因子為 0。圖 3 是根據一範例實施例繪示雙階彈性回填演算法的虛擬碼。請參照圖 3，同樣以第二專案 p_u 與第二工作 $j_{u,v}$ 為例。在第 3-4 行中，若可以立即執行第二工作 $j_{u,v}$，雲端系統 110 會執行第二工作 $j_{u,v}$。在第 6-7 行中，雲端系統 110 對於每一個第二工作 $j_{u,v}$ 都會取得至少一個回填時間(成為集合 BT)。回填時間所指的是在該時間下有足夠的資源來處理第二工作 $j_{u,v}$，因此回填時間可能不只一個。在第 8-15 中，對於集合 BT 中的每一個回填時間，雲端系統 110 都會嘗試加

入對應的第二工作 $j_{u,v}$，以判斷是否可以在對應的回填時間加入第二工作 $j_{u,v}$。判斷的基準在於若有第一工作被延遲，則被延遲的第一工作的最晚開始時間必須大於等於該第一工作的開始時間。換句話說，雲端系統 110 會在所有的回填時間中，挑選出一個回填時間(亦稱第一回填時間)，藉此被延遲的第一工作的最晚開始時間必須大於等於該第一工作的開始時間。值得注意的是，由於鬆弛因子為 0，因此並不會有專案被延遲，這也表示第二工作 $j_{u,v}$ 只能延遲第一專案中最後一個工作以外的其他工作。在一範例實施例中，若可被延遲的第一工作的數目大於 1，則雲端系統 110 會從這些第一工作中，挑選並延遲有最大的最晚開始時間的第一工作。接下來，雲端系統 110 便會把第二工作 $j_{u,v}$ 在對應的第一回填時間加入至排程規劃。在圖 3 中是先把第二工作 $j_{u,v}$ 加入排程規劃，再判斷是否可行，若不可行則回復舊的排程規劃以找到第一回填時間，然而本發明並不限制挑選第一回填時間的實作。在第 19-21 行，對於每一個第二工作 $j_{u,v}$，雲端系統 110 會根據第二專案的離開時間 td_u、第二專案的運轉時間 tn_u、第二工作的服務時間 $te_{u,v}$、與鬆弛因子 SF 來更新第二工作 $j_{u,v}$ 的最晚開始時間 $lts_{u,v}$。

【0030】 第三種排程方法是相同於上述的雙階彈性回填演算法，但鬆弛因子會被設定大於 0。換言之，當加入第二工作 $j_{u,v}$ 時，原本在排程規劃中的第一專案可能會被延遲，但不會被延遲超過上限 $td_u + tn_u \cdot SF$。換言之，第二工作 $j_{u,v}$ 的開始時間會在範圍 $[ts_{u,v}, td_u + tn_u \cdot SF - te_{u,v}]$ 之間。在此排程演算法中，雲端系統 110 還會

設定一個搶佔限制個數，其是一個正整數。當在挑選第一回填時間的時候，除了讓被延遲的第一工作的最晚開始時間大於等於該第一工作的開始時間，也必須讓被延遲的專案的個數小於此搶佔限制個數。

【0031】 第四種排程方法稱為雙階權重回填演算法。圖 4 是根據一範例實施例繪示雙階權重回填演算法的虛擬碼。請參照圖 4，在此一樣以第二專案 p_u 與第二工作 $j_{u,v}$ 為例。此外，上述的搶佔限制個數標記為 PL。首先，雲端系統 110 會判斷第二專案 p_u 是屬於高權限專案或是低權限專案。若第二專案 p_u 是屬於高權限專案，在第 3-4 行中，雲端系統 110 會根據第二專案 p_u 的權限 pi_u 來更新鬆弛因子 SF，並且根據更新後的鬆弛因子執行圖 3 所述的雙階彈性回填演算法(即，包括如何挑選第一回填時間)。具體來說，雲端系統 110 會將一權重變數減去權限 pi_u 以取得一第三數目，並且將第三數目乘上鬆弛因子 SF 以取得更新後的鬆弛因子。在圖 4 中權重變數為 1，權限 pi_u 是介於 0 與 1 之間，但在其他範例實施例中權重變數與權限 pi_u 也可以為其他的實數，本發明並不在此限。

【0032】 若第二專案 p_u 是屬於低權限專案，則雲端系統 110 會執行圖 2 所述的雙階嚴格回填演算法。接下來，在第 7-10 行中，雲端系統 110 會根據第二專案 p_u 的權限 pi_u 來更新鬆弛因子 SF；以及根據更新後的鬆弛因子來更新每一個第二工作 $j_{u,v}$ 的最晚開始時間 $lts_{u,v}$。然而，更新鬆弛因子與更新最晚開始時間的步驟已詳細說明如上，在此並不再贅述。

【0033】 圖 5 是根據一範例實施例繪示排程規劃的資料結構的示意圖。

【0034】 請參照圖 5，在一範例實施例中，排程規劃是實作為以連結串列(linked list)為基礎的資料結構。在串列 510 上的每一個節點都記錄了時間、可用資源以及預定工作。例如，在節點 511 記錄了時間 0、並且資源 A 沒有可用的資源，而資源 B 還有一份資源可以用。節點 511 是連接到節點 512，而節點 512 是連結到節點 513，其中節點 511 與節點 512 都表示一個預定要執行的工作。詳細來說，節點 511 與節點 512 都記錄了需要資源、服務時間、專案編號、工作編號與最後開始時間。例如，節點 512 指示在時間 0 所要執行的是工作 $J_{1,2}$，並且佔據了一份資源 A 與兩份資源 B。此外，工作 $J_{1,2}$ 的服務時間為 5 個時間單位。節點 513 指示在時間 0 所要執行的是工作 $J_{2,1}$，並且佔據了兩份資源 A 與一份資源 B。工作 $J_{2,1}$ 的服務時間為 2 個時間單位。依照串列 510 上的資料結構，可以得到圖 5 右側的排程規劃。舉例來說，在時間 2 工作 $J_{1,1}$ 會被加入，並且節點 514 記錄了工作 $J_{1,1}$ 的相關資訊。

【0035】 以另外一個角度來說，本發明一範例實施例提供一種電腦程式產品，其中包括多個程式碼，用以被雲端系統 110 中的電腦載入以執行上述的各個排程演算法。此電腦程式產品可儲存在一個電腦可讀取記錄媒體。電腦可讀取記錄媒體例如為硬式磁碟機、軟式磁碟機、記憶卡、隨身碟、韌體、光碟、唯讀記憶體（read only memory, ROM）、隨機存取記憶體（random access memory，

RAM）或任何可儲存程式指令（或可儲存程式碼）的記錄媒體。或者，電腦程式產品也可以透過網路、任意的有線或無線介面來傳輸。

【0036】 綜上所述，本發明範例實施例提出的雙階排程方法與電腦程式產品，可以藉由設定最晚開始時間來將一個工作回填至排程規劃中。此外，鬆弛因子與搶佔限制個數都可以用來控制雲端系統的排程。藉此，可以減少平均的運轉時間。

【0037】 雖然本發明已以實施例揭露如上，然其並非用以限定本發明，任何所屬技術領域中具有通常知識者，在不脫離本發明的精神和範圍內，當可作些許的更動與潤飾，故本發明的保護範圍當視後附的申請專利範圍所界定者為準。

【符號說明】

【0038】

110：雲端系統

120：專案

122、124、131~133、141、151、$J_{1,1}$、$J_{1,2}$、$J_{2,1}$、$J_{2,2}$：工作

140：專案佇列

150：工作佇列

510：串列

511~514：節點

申請專利範圍

1. 一種雙階排程方法，用於一電腦以排程多個專案，其中一排程規劃包括該些專案中的一第一專案，該第一專案包括第一工作，該雙階排程方法包括：

根據該第一專案的一運轉時間、該第一專案的一離開時間、該第一工作的一服務時間、與一鬆弛因子來計算該第一工作的最晚開始時間；

取得該些專案中的一第二專案，其中該第二專案包括多個第二工作；

對於每一該些第二工作，取得至少一回填時間；

對於每一該些第二工作，從對應的該至少一回填時間中挑選一第一回填時間，使得被延遲的該第一工作的該最晚開始時間大於等於該第一工作的一開始時間；以及

將每一該些第二工作在對應的該第一回填時間加入至該排程規劃。

2. 如申請專利範圍第 1 項所述的雙階排程方法，其中計算該第一工作的最晚開始時間的步驟包括：

將該第一專案的該運轉時間乘上該鬆弛因子以取得一第一數目；以及

將該第一專案的該離開時間加上該第一數目，再減去該第一工作的該服務時間以取得該第一工作的該最晚開始時間。

3. 如申請專利範圍第 2 項所述的雙階排程方法，其中在將每

一該些第二工作在對應的該第一回填時間加入至該排程規劃以後，該雙階排程方法更包括：

對於每一該些第二工作，根據該第二專案的一運轉時間、該第二專案的一離開時間、該第二工作的一服務時間、與該鬆弛因子來更新該第二工作的該最晚開始時間。

4. 如申請專利範圍第 1 項所述的雙階排程方法，更包括：

若可被延遲該第一工作的數目大於 1，從該些第一工作中，挑選並延遲有最大的該最晚開始時間的該第一工作。

5. 如申請專利範圍第 1 項所述的雙階排程方法，其中從該至少一回填時間中挑選該第一回填時間步驟更包括：

從該至少一回填時間中挑選該第一回填時間，使得該些專案中被延遲的專案的個數小於一搶佔限制個數。

6. 如申請專利範圍第 1 項所述的雙階排程方法，更包括：

判斷該第二專案是屬於一高權限專案或是一低權限專案；

若該第二專案是屬於該低權限專案，對每一該些第二工作執行一第一程序，其中該第一程序包括：

判斷該第二工作是否可立即被執行；

若該第二工作可立即被執行，則執行該第二工作；以及

若該第二工作不可立即被執行，在一開始時間將該第二工作加入至排程規劃，使得該排程規劃中沒有工作被延遲。

7. 如申請專利範圍第 6 項所述的雙階排程方法，更包括：

若該第二專案是屬於該低權限專案，在對每一該些第二工作

執行該第一程序以後，根據該第二專案的一權限來更新該鬆弛因子；以及

根據更新後的鬆弛因子來更新每一該些第二工作的該最晚開始時間。

8. 如申請專利範圍第 6 項所述的雙階排程方法，更包括：

若該第二專案是屬於該高權限專案，根據該第二專案的一權限來更新該鬆弛因子；以及

根據更新後的鬆弛因子執行所述挑選該第一回填時間的步驟。

9. 如申請專利範圍第 8 項所述的雙階排程方法，其中根據該第二專案的該權限更新該鬆弛因子的步驟包括：

將一權重變數減去該第二專案的該權限以取得一第三數目；以及

將該第三數目乘上該鬆弛因子以取得更新後的該鬆弛因子。

10. 一種電腦程式產品，被電腦載入以執行多個步驟以排程多個專案，其中一排程規劃包括該些專案中的一第一專案，該第一專案包括第一工作，該些步驟包括：

根據該第一專案的一運轉時間、該第一專案的一離開時間、該第一工作的一服務時間、與一鬆弛因子來計算該第一工作的最晚開始時間；

取得該些專案中的一第二專案，其中該第二專案包括多個第二工作；

對於每一該些第二工作，取得至少一回填時間；

對於每一該些第二工作，從對應的該至少一回填時間中挑選一第一回填時間，使得被延遲的該第一工作的該最晚開始時間大於等於該第一工作的一開始時間；以及

將每一該些第二工作在對應的該第一回填時間加入至該排程規劃。

圖式

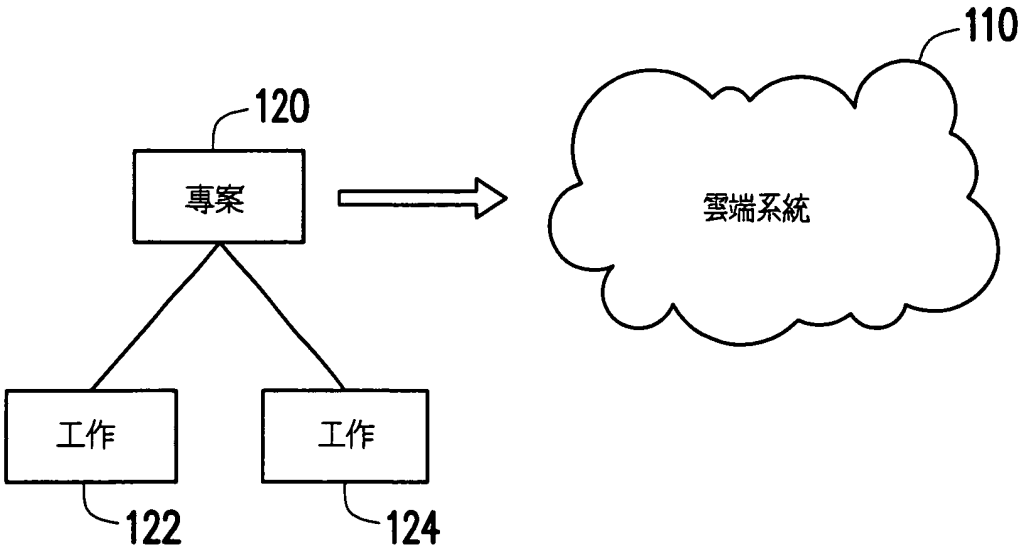


圖1A

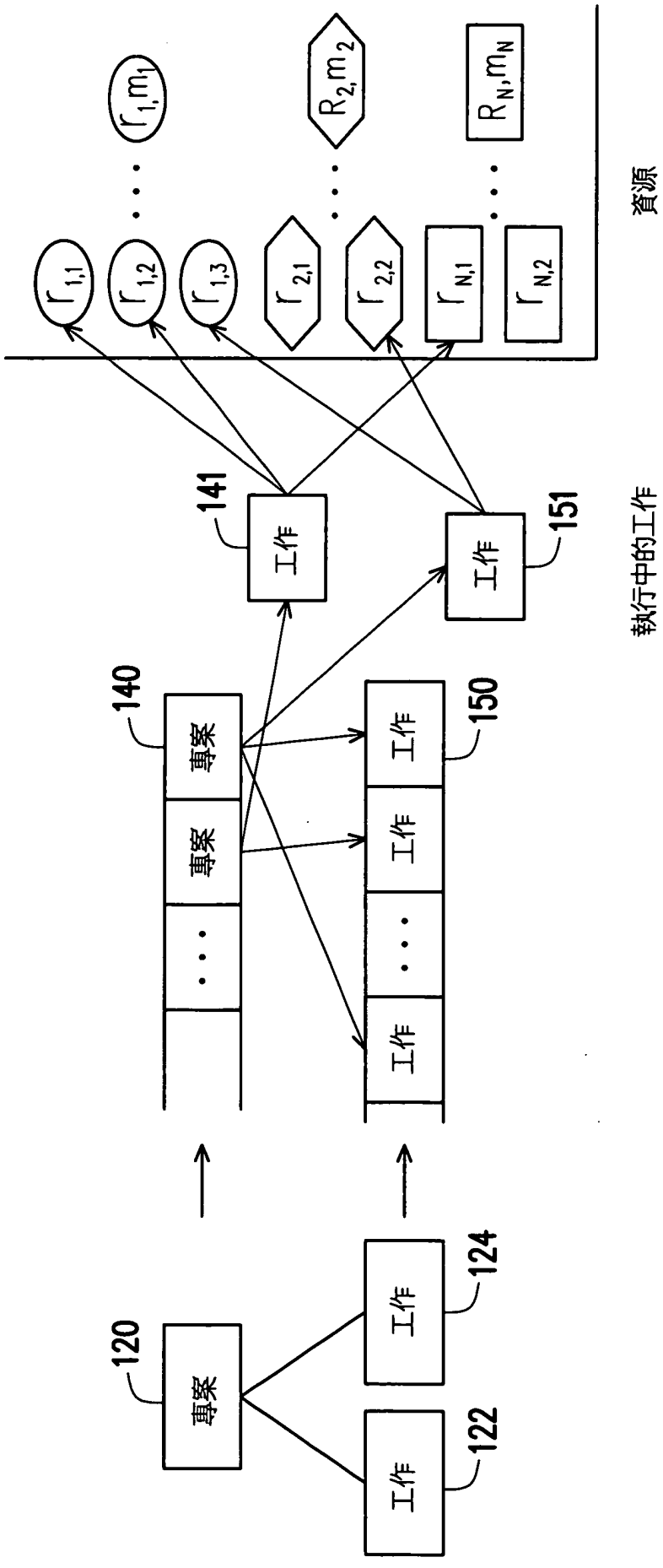

120
專案
122
工作
124
工作
140
專案
專案
141
工作
150
工作
工作
工作
151
工作
執行中的工作
$r_{1,1}$
$r_{1,2}$
$r_{1,3}$
r_{1,m_1}
$r_{2,1}$
$r_{2,2}$
R_2,m_2
$r_{N,1}$
$r_{N,2}$
R_N,m_N
資源

圖1B

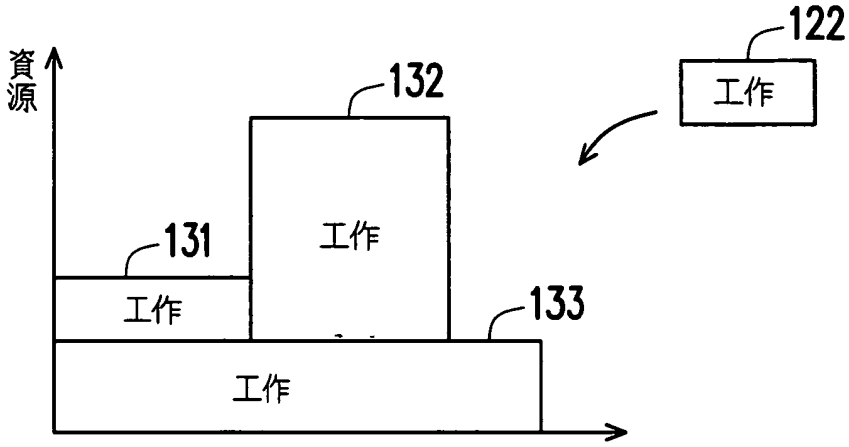


圖 1C

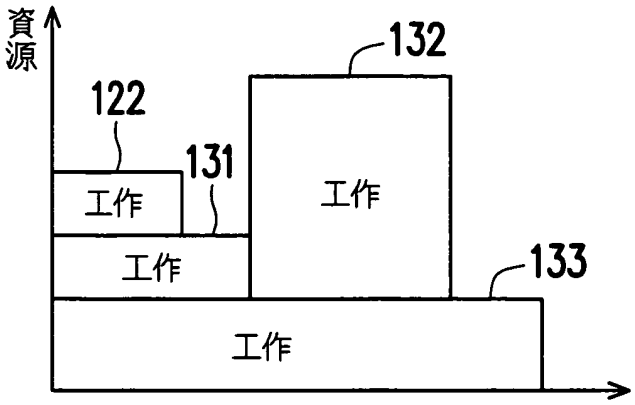


圖 1D

Two_Tier_StrictBackfilling(p_u)

```
1   Begin
2     for cach j_{u,v} ∈ J_u do
3       If (j_{u,v} can start immediately) then
4         start  j_{u,v}
5       else
          # Find the  earliest  starting time for job j_{u,v}
6         ts_{u,v} ← earliestStartingTime(j_{u,v})
          # Set up a resource reservation for the job  j_{u,v}
7         lts_{u,v} ← ts_{u,v}
8         addReservation(j_{u,v})
9       end-if
10    end-for
11  End
```

圖2

Two_Tier_FlexibleBackfilling(p_u, SF, PL)

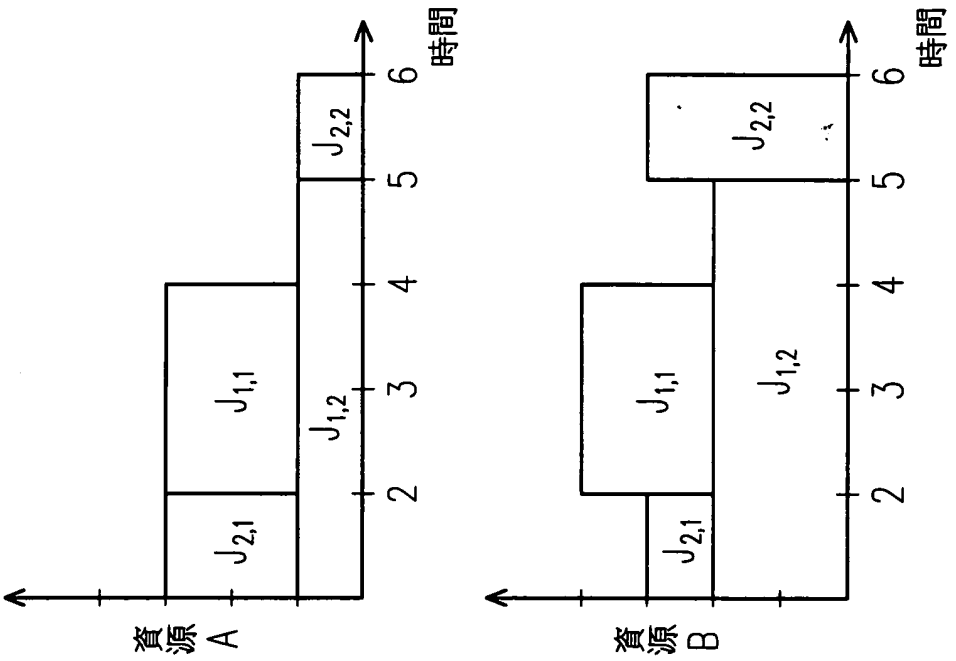
```
1   Begin
2     for cach j_{u,v} ∈ J_u do
3       If (j_{u,v} can start immediately) then
4         start j_{u,v}
5       else
          # Find the feasible backfilling time for job j_{u,v}
6         BT ← feasible Backfilling Times(j_{u,v})
7         Let S_old be the current scheduling plan
8         for each bt ∈ BT do
9           ts_{u,v} ← bt
            # Set up a resource reservation for the job j_{u,v}
10          addReservation(j_{u,v})
          # Check the availability of system resource and
            # delay existing reservations in order to
            # accommodate job j_{u,v} if necessary.
11        succeed ← shiftReservations(j_{u,v}, PL)
12        if (! succeed) then
13          restore S_old
14        else
15          break
16        end-if
17      end-for
      end-if
18  end-for
19  for each j_{u,v} ∈ J_u do
      # Update the latest starting time for # the resource
      # reservation of job j_{u,v}
20    lts_{u,v} ← td_d + (tn_u • SF) - te_{u,v}
21    updateLatestStartingTime(j_{u,v})
22  end-for
23 End
```

圖3

Two_Tier_PriorityBackfilling(p_u ,SF,PL)

```
1   Begin
2     if (p_u is high_proiority )
3         SF ←(1- pi_u )*SF
4         Two_Tier_FlexibleBackfilling(p_u ,SF,PL)
5   else
6         Two_Tier_StrictBackfilling(p_u)
7         for cach j_u,v ∈ J_u do
8           SF ← (1 - pi_u ) • SF
9           lts_u,v ← td_u,v + (tn_u • SF) - te_u,v
10          updateLatestStartingTime(j_u,v)
11    end-for
12  end-if
13 End
```

圖4


資源 A
$J_{2,1}$
$J_{1,1}$
$J_{1,2}$
$J_{2,2}$
2 3 4 5 6
時間
資源 B
$J_{2,1}$
$J_{1,1}$
$J_{1,2}$
$J_{2,2}$
2 3 4 5 6
時間

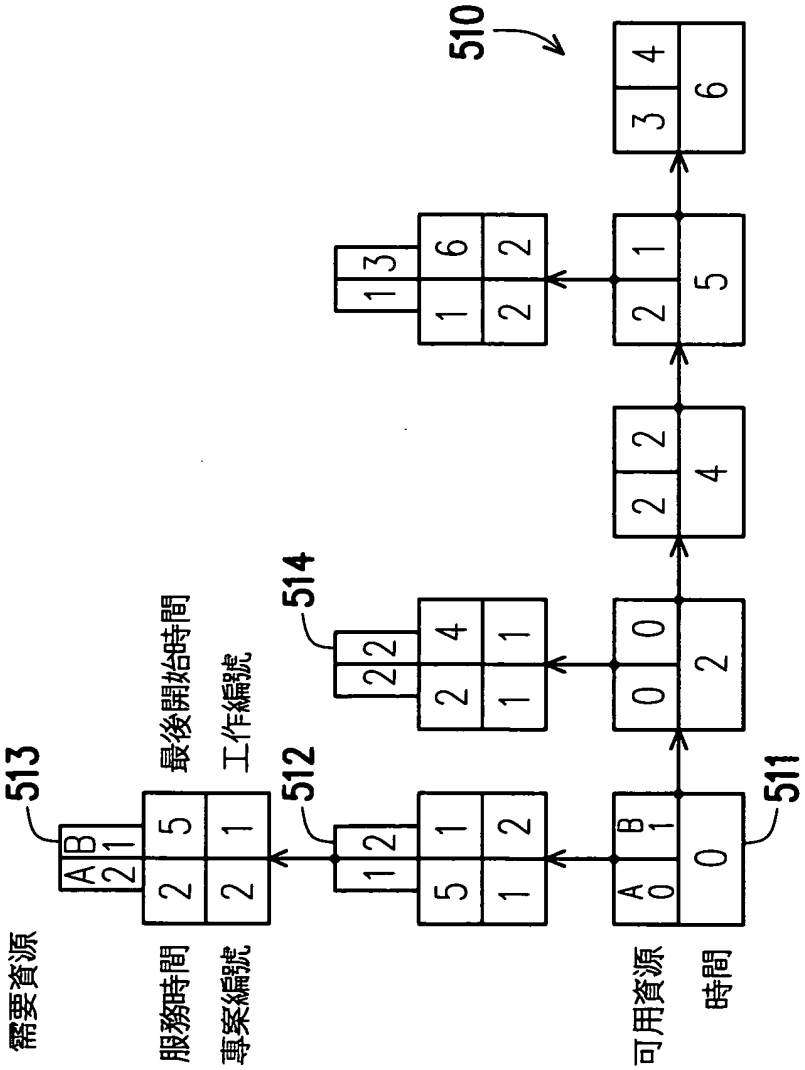

510
513
512
514
511
需要資源
服務時間
專案編號
最後開始時間
工作編號
可用資源
時間
A B
2 1
2 5
2 1
1 2
5 1
1 2
2 2
2 4
1 1
A B
0 1
0 0
0 2
2 2
2 4
2 1
2 5
1 3
1 6
2 2
3 4
6

圖5