在行動計算環境中使用版本集合放鬆互斥一致性的資料暫存方法

李之中
中華大學資訊管理系
leecc@chu.edu.tw

劉郁君
中華大學資訊管理系
m09410023@chu.edu.tw

摘要

在行動計算環境中，使用資料廣播方式可以解決行動客戶端的規模性問題。同時為了減少行動客戶端執行應用時的反應時間，我們也在行動客戶端上使用資料暫存技術。為了維持行動客戶端暫存資料的正確性，多數的相關方法都使用互斥一致性作為使用資料時的正確性標準。然而，使用互斥一致性作為資料正確性的標準，可能造成行動客戶端查詢的反應時間無法減短，也可能造成效能無法滿足使用者需求。然而在日常生活中有許多應用並不需要在如此嚴苛的標準下執行，像是天氣預報，因此我們希望將暫存資料的正確性標準由互斥一致性的狀態放鬆為滿足一致性即可，在此前提下，我們設計了一個版本集合的資料暫存方法，我們將行動客戶端的暫存記憶體加入了三個版本集合並將不同狀態的資料分別存放於三個版本集合中，這三個版本集合分別為新版本集合、目前版本集合、失效版本集合，除此我們設計一套控制版本集合的方法，利用此方法將行動客戶端執行應用的結果限制在某個一致性狀態下，並允許使用者可使用稍早的資料項來執行應用，且利用此控制方法可避免使用過於陳舊的資料項導致執行應用時的錯誤。我們以系統模擬的方式來評估版本集合的效能，經由實驗發現使用版本集合可提供行動客戶端更短的查詢反應時間，並允許行動客戶端更長時間的離線。

關鍵詞：行動計算環境、暫存資料、互斥一致性

一、前言

在行動計算環境中，學界利用資料廣播方式來解決行動計算環環境中的行動客戶端的規模問題[1-6]。在資料廣播方式中，資料庫伺服端透廣播伺服端，將使用者所感興趣的資料主動「推」(Push)給行動客戶端，如此一來，行動客戶端只要監聽廣播通道(Broadcast channel)，並在所需資料項被播出時下載該資料項。透過此種資料廣播方式，使得行動客戶端不需再競爭連接資料庫伺服端的通訊通道。因此，整個系統執行應用的成本與效能將不受行動客戶端數量影響。

在資料廣播方式中，為了減短使用者擷取資料項的擷取時間(Access time)，學界常使用資料暫存技術來達到此一目的[6]。使用此一技術時，在行動客戶端設置暫存記憶體(Cache)，行動客戶端可將已擷取過的資料項儲存於暫存記憶體中。當下次查詢也需要相同的資料項時，行動客戶端便可直接使用儲存於暫存記憶體中的資料項來回答，減短擷取時間。然而，但當行動客戶端與資料庫伺服端同時保有同一資料項時，便要注意到該資料項的一致性問題。如果使用者使用了不一致的資料，可能造成查詢結果不滿足使用者的需求。因此要如何有效的保留資料及確保資料的一致性問題，是本篇論文關切的重點。

在傳統資料庫系統中，序列化(Serializability)常被作為交易執行的正確性標準[7]。在分散式資料庫系統中，複本資料庫系統(Replica database)則是使用互斥一致性(Mutual consistency)作為交易執行時的資料一致性標準[8]。然而這些在有線環境下所適用的標準，在行動計算環境下，由於行動客戶端與資料庫伺服端網路連線的不可靠性而變得難以繼續適用[21,16,22,26,27]。因此，為行動計算環境尋找新的正確性標準也成為學界研究的目標之一。

已經有許多學者對行動計算環境中的資料一致性問題進行研究[1-6,]。目前學界所提出的方法，可以分成兩大類。第一類的方法為保證行動客戶端執行應用時必須使用與資料庫伺服端相同的資料值[10-15]。第二類的方法則是行動客戶端的暫存料可以與資料庫伺服端的資料有若干程度的不一致，其觀念類似由學者 Garcia-Molina 所提出的 Quasi-copy 的觀念[23]。

針對上述第一類保證行動客戶端執行應用時必須使用資料的最新值的方法，一般而言，學界大多使用由美國學者 Barbara D.與 Imielinski T.[10]所提出的驗證報告方式(Invalidation report approach)來維持資料一致性。在這個方式中，廣播伺服端定期產生並廣播驗證報告(Invalidation report)給所有的行動客戶端。當行動客戶端收到驗證報告後，再根據驗證報告刪除暫存記憶體中失效的資料項，以維持暫存資料的一致性。在此種方式下，行動客戶端執行查詢時，必須先完成一次暫存資料的資料驗證，確保暫存資料是最新值之後，才能進行查詢的執行。

對使用 Quasi-copy 觀念來維持資料一致性的第二種方式，則是針對每個暫存的資料項設定一個與正確資料之間的距離(Distance)。距離可以使用資料項的存活時間(Time to live: TTL)[26]或資料項的更新次數來表示。在此種方式下，當行動客戶端執行查詢需要使用某個暫存資料項時，行動客戶端將透過檢查該暫存資料項的存活時間來進行驗證，確定資料項仍是有效的之後，才繼續查詢的進行。

在上述所提的兩種維持資料一致性的方式中，驗證報告方式相較使用存活時間方式維持了比較強烈的一致性，但是查詢的反應時間卻是使用存活時間方式較短。造成此一現象的原因為在驗證報告方式中，行動客戶端執行查詢時，必須先完成一次暫存資料的資料驗證，確保暫存資料是最新值之後，才能進行查詢的執行。然而，當查詢所需資料項皆暫存於暫存記憶體且非陳舊資料時，此時查詢仍然無法進行，(事實上，此時暫存資料就是該查詢的正確資料)，仍須等到暫存資料經過驗證報告驗證後(確定暫存資料是最新值)，才能將查詢結果回傳給使用者，如此一來便造成查詢反應時間的增長。如果行動客戶端在等待驗證報告的過程中斷線，為了維持暫存資料的一致性，行動客戶端還是必須完成資料驗證後才能繼續查詢的進行，因此在斷線期間會造成行動客戶端處於停滯(Halt)狀態，無法處理任何查詢。在使用存活時間方式中，雖然暫存資料向只要仍在存活時間內就可被使用，相對而言並不會發生需要等待資料驗證而造成查詢延遲的情形，但是所提供資料項的一致性卻較不精確，特別是當查詢所需的資料項之間必須提供一個邏輯一致觀點(Logical consistency view)[17]時，使用存活時間方式無法滿足此種一致性的需求。所謂邏輯一致觀點要求查詢所使用的多個資料項的狀態，必須是資料庫中某個一致性狀態的部份集合。

為了克服上述缺點，使行動客戶端可以更彈性的使用暫存資料，且在真實的行動環境中，有許多情況使用者可接受使用稍早的資料來執行應用，像是天氣預報、快遞商品追蹤等等，因此我們可考慮將暫存資料的標準由互斥一致性放鬆到只要讀到過去一致性即可，這代表行動客戶端的暫存資料並非最新的狀態，但我們也不希望使用過於陳舊的資料項導致執行應用的錯誤。因此，我們在放鬆互斥一致性的前提下，提出一個在行動客戶端中使用版本集合(Version set)的方式來進行暫存資料的管理方法。我們希望利用此方法可提供執行應用程式較短的反應時間(Response time)，同時也能根據行動客戶端與資料庫伺服端的通訊狀態提，使用不同狀態的一致性資料，當通訊狀態良好時暫存資料的狀態幾乎可達到互斥一致性的狀態，當通訊狀態不好時，也能根據行動客戶端現有的暫存資料提供使用者最佳的服務。

本論文其他組織如下：第二節將對與本研究相關的研究進行討論，第三節將介紹本方法所使用的系統架構，在第四節中介紹我們所提出的使用版本集合放鬆互斥一致性的資料暫存方法，第五節為說明利用系統模擬的方式進行的效益評估，第六節為我們的結論。

二、文獻探討

在傳統主從架構下，資料庫執行交易時的並行控制與暫存資料一致性控制被視為一個整體而一起討論[18]。已有許多暫存資料並行控制與一致性控制同時考慮的方法被提出[18,19]，學者Franklin[19]對這些方法進行了分類與效益分析。由於在行動計算環境中，伺服端無法正確掌握行動客戶端的正確位置、暫存資料的內容與通訊狀態，使得這些在傳統主從架構下所設計的方法無法適用於行動式計算環境[12]。然而在行動式計算環境中，並行控制與暫存資料的一致性控制是分開進行的。先前的研究[11,12,14,15,16]大多是假設異動於伺服端資料庫確認後，如何與行動客戶端的暫存資料進行一致性控制。學者 Barbara 與 Imielinski [10]提出廣播時間戳記(Broadcasting timestamps)與健忘終端機(Amnesic terminal)兩種方法進行暫存資料的一致性控制。這兩個方法都是利用廣播驗證報告的方式進行暫存資料的一致性控制。其他同樣使用廣播驗證報告的方式進行暫存資料的一致性控制的研究有位元序列法(Bit sequence)[11]。此外，當行動客戶端的斷線時間超過驗證報告所能驗證的範圍時，[14]中的作者提出保留低異動集合群組(GCORE ：Grouping with cold update-set retention)的方法以維持維持暫存資料一致性。Tan 與 Cai[20]提出廣播基礎群組驗證法(BGI：Broadcast-based group invalidation)。

在行動計算環境中，為了維持暫存資料的互斥一致性，系統需要付出較多的成本[3,24,25]。在[24]的方法中，系統除了廣播資料外，還要廣播用來偵測與調解資料衝突的控制矩陣。在[25]中，為了偵測行動客戶端所使用的資料是否符合序列化，系統需要廣播序列化圖。在[3]中，對於在資料廣播過程中，才被更新的當週期已經被廣播的資料，系統需要重播這些資料項。無論是廣播控制矩陣、廣播序列化圖或重播已被更新的資料項，都會增加系統的成本。也因此中上先前的研究就已經有學者主張在此環境中放鬆互斥一致性[21,16,22,26,27]。在[21,27]的研究中討論了在行動式計算環境下使用 Quasi-copies 維持暫存資料一致性的想法。在[26]中，廣播伺服端不僅廣播資料的最新值，還廣播該資料之前的版本，行動客戶端可依實際需要擷取適當一致性的資料。在[16]中，學者 Wong 與 Leung 延伸多版本(Multiversions)的並行控制法的精神，提出暫存資料為伺服端資料庫的快照(Snapshot)之想法，進行暫存資料的一致性控制。學者 Young 與 Chiu[22]使用 View consistency 與 Local view consistency 作為行動計算環境中資料一致性的標準。

三、系統模型

在本節中將會介紹使用資料廣播方式的行動計算環境(在本論文稱之為資料廣播環境)的架構及運作情形，整個資料廣播環境架構如圖 1 所示分為三大部分，這三個部分分別為資料庫伺服端、廣播伺服端、行動客戶端。在介紹整個環境前，我們先

做兩項假設，第一、所有的異動交易只在資料庫伺服端產生，第二、行動客戶端只能進行查詢交易並不能產生異動，此兩項假設與過去研究[29]相同。

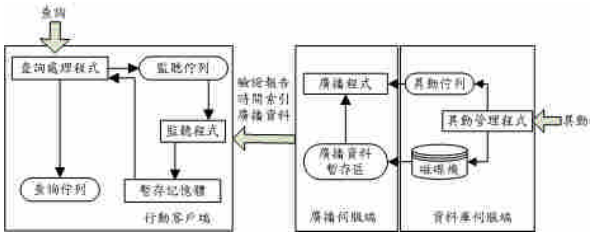

圖 1 資料廣播環境系統架構圖

(一)資料伺服端與廣播伺服端

在資料播環境當中，資料庫伺服端的主要任務是執行異動(Updates)並進行異動時的交易管理(Transaction management)。當異動發生時，資料庫伺服端使用並行控制機制決定異動的執行順序，當異動確認後(Commit)，異動管理程式便將此次異動訊息暫存於異動佇列中，以待稍後產生驗證報告。

而廣播伺服端的主要任務為根據廣播結構(Broadcast structure)將資料以週期且重複的方式廣播給行動客戶端。廣播伺服端根據廣播結構播完一次的廣播內容稱為廣播週期。廣播結構如圖 2 所示，整個廣播結構可分為驗證報告、時間索引、廣播資料三個部份，其中驗證報告為前 w 個廣播週期資料庫伺服端的異動資訊，如圖 3 所示，假設目前的時間為 T_i，L 為進行一次廣播週期所需的時間，驗證報告由 w 個廣播週期(稱為更新視窗(Update windows))異動資訊所組成，在更新視窗為 2 的條件下，更新視窗的時間為 T_{i-2} 到 T_i。而廣播資料為此次廣播週期所要廣播的資料，在廣播週期到達前廣播伺服端會決定出廣播順序，在決定好廣播順序後，資料庫伺服端便會產生廣播資料時的時間索引，等待廣播週期到達後，廣播伺服端便將驗證報告、時間索引、廣播資料傳送給行動客戶端。

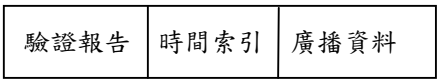

圖 2 廣播結構

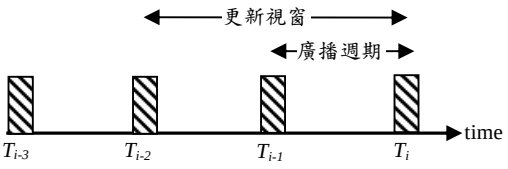

圖 3 廣播週期

資料庫伺服端與廣播伺服端的協同處理如下：當資料伺服端在成並確認異動交易完後，將此異動訊息暫存至異動佇列中。當新的廣播週期即將到達時，廣播伺服端會依據異動佇列產生驗證報告，同時將廣播資料自資料庫伺服端存放於廣播資料暫存並產生時間索引，當廣播週期到達時便將驗證報告、時間索引、廣播資料透過廣播通道傳送給行動客戶端。

上述過程中，由於廣播伺服端採用週期方式廣播資料庫中的內容，所以也採用週期的方式向資料庫伺服端讀取資料庫的最新狀態，利用此種週期的方式讀取資料庫最新狀態可維持同一廣播週期資料項之間的一致性。由此之故，當資料庫中有任何異動產生時並不會立即在廣播通道表現出來，而是等到下個廣播週期到達後，廣播伺服端向資料庫伺服端讀取資料庫最新狀態後，資料庫的最新狀態才會在廣播伺服端反應出。

(二)行動客戶端

行動客戶端的運作情形如下，當行動客戶端接收到使用者查詢資料項的要求後，查詢處理程式(Query processing program)首先找出此次查詢需要哪些資料項，再到暫存記憶體中此次查詢所需的資料項，如果此次查詢所需的資料項皆存在於暫存記憶體中則回答並結束此次查詢，如果無法結束此次查詢，便將此次查詢所需但卻不存在於暫存記憶體中的資料項識別碼加到監聽佇列中，等待下次廣播週期的到來。當下個廣播週期到達後，行動客戶端首先會根據驗證報告的內容來驗證暫存資料項的正確性，當發現暫存資料項失效後便將此失效資料項自暫存記憶體中刪除，藉由此驗證過程可將暫存記憶體中的資料項狀態更新到與資料庫伺服端相同狀態下。當暫存記憶體的正確性經驗證後，監聽程式便根據時間索引推算出監聽佇列中資料項的廣播時間，當廣播時間到達後逐一地將資料項自下傳通道載入記憶體中，接著查詢處理程式開始判斷查詢佇列中所存放的查詢，是否有某個查詢所需的資料項皆存在暫存記憶體中，如果是的話便回答並結束此次查詢

在上述過程中，由於行動客戶端的暫存資料項都來自於廣播通道中，因此只要行動客戶端的暫存資料皆來自同一廣播週期，那麼暫存資料便為一致性的資料。而一個簡單維持暫存資料一致性的方法就是當每個廣播週期結束時便刪除所有暫存資料項，並在下個廣播週期重新擷取執行應用時所需資料項，但利用此方法很可能會刪除暫存記憶體中正確的資料項，造成行動客戶端不斷地擷取這些正確的資料項，特別是使用者經常性地查詢某些特地資料項時，行動客戶端便重複的等待資料項自廣播伺服端播出，造成效能的降低。因此，我們採用驗證暫存資料項的方式僅刪除失效的資料項，而正確的資料項予以保留，供行動客戶端下次執行相同應用時使用。

四、使用版本集合放鬆互斥一致性資料暫存方法

在本節中我們將介紹使用版本集合放鬆互斥一致性的資料暫存方法。本節中的第一小節介紹版本集合，第二小節則介紹資料暫存方法。

(一)版本集合

我們將行動客戶端的暫存資料分成三個集合，我們稱這些集合為版本集合(Version set)，稍後我們將介紹如何利用版本集合來放鬆暫存資料的互斥一致性。

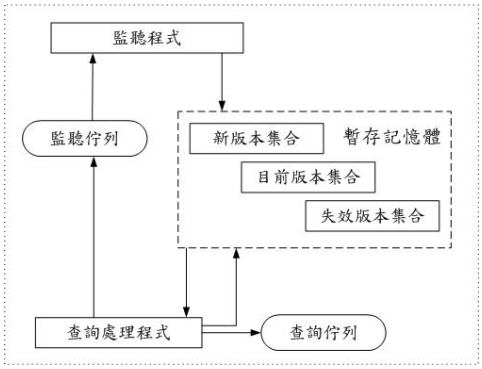

圖 4 使用版本集合的行動客戶端架構圖

圖 4 為將行動客戶端加入版本集合後的行動客戶端架構圖。在圖 4 中，行動客戶端暫存記憶體被劃分成新版本集合(New version set)、目前版本集合(Current version set)、失效版本集合(Invalid version set) 等三個版本集合。這三個版本集合分別用來存放分屬不同資料庫狀態的資料項內容－新版本集合所存放的是本次廣播週期所接收到的資料項。目前版本集合存放的是在上個廣播週期所接收同時在本次週期仍然有效的資料項。失效版本集合存放的是在上個廣播週期所接收但是在本次週期已經失效資料項。

(二)資料暫存方法

在本節中，我們將藉由事件驅動(Event driven)的方式說明使用版本集合資料暫存方法的運作方法。在我們的方法中，行動客戶端共有三個事件，分別為查詢到達(Query arrival)、驗證報告到達(Invalidation report arrival)、監聽資料到達(Listen data arrival)。本節的說明次序依序為說明所有方法所使用的符號、查詢到達方法、驗證報告到達方法、監聽資料到達方法。

1 符號

使用版本集合資料暫存方法各事件方法所使用的符號整理如表 1。在表 1 中，VS_{new} 代表暫存記憶體中的新版本集合，$VS_{current}$ 代表暫存記憶體中的目前本集合，$VS_{invalid}$ 代表暫存記憶體中的失效版本集合。*Data_Item_Set(Query)*代表查詢 *Query* 的查詢資料項集合，*Get_a_Data_Item*(*Data_Item_Set*(*Query*)) 代表自查詢到的資料項集合中，取出一個資料項的識別碼，*Get_a_Data_Item*($REPORT_{invalidation}$)代表自驗證報告 $REPORT_{invalidation}$ 中取出一個資料項的識別碼，$QUEUE_{query}$ 代表查詢佇列，用來存放尚未完成的查詢，$QUEUE_{listen}$ 代表監聽佇列，用來存放需要監聽的資料項的識別碼，$QUEUEE_{arrival}$ 代表系統中剛到達的查詢，$Data\_Item_{arrival}$ 代表系統中剛到達的資料項，*ID*(*Data_Item*)代表資料項 *Data_Item* 的識別碼。

表 1 符號的定義

名稱	描述
VS_{new}	新版本集合
$VS_{current}$	目前版本集合
$VS_{invalid}$	失效版本集合
Data_Item_Set(*Query*)	查詢 *Query* 的資料項集合
Get_a_Data_Item(*Data_Item_Set*(*Query*))	從查詢 *Query* 的資料項集合中，取出一個資料項
Get_a_Data_Item_ID($REPORT_{invalidation}$)	從驗證報告 $REPORT_{invalidation}$ 中取出一個資料項識別碼
$Queue_{query}$	查詢佇列，儲存尚未執行完成的查詢
$Queue_{listen}$	監聽佇列
$Query_{arrival}$	系統中剛到達的查詢
$Data\_Item_{arrival}$	系統中剛到達的資料項
ID(*Data_Item*)	資料項 *Data_Item* 的識別碼

2 查詢事件到達方法

處理本事件方法的演算法如圖 5 所示，而其處理流程可概分為三個步驟，分述如下：

1. 檢查新版本集合與目前版本集合的聯集中(表示為 $VS_{new} \cup VS_{curren}$)是否含有此次所需查詢的所有資料項(表示為 *Data_Item_Set*($Query_{arrival}$))，如果是的話則回答並結束此次查詢，否則進行下一個步驟。
2. 檢查目前版本集合與失效版本集合的聯集(表示為 $VS_{current} \cup VS_{invalid}$)中是否含有此次查詢所需的所有資料項，如果是的話則回答並結束此次查詢，並告知使用者此次查詢結果有可能不在互斥一致性下。反之則代表此次查詢所要求的資料項並未全部暫存於暫存記憶體中，需要進行下一個步驟。
3. 將此查詢所要求的資料項卻未存在行動客戶端暫存記憶體的資料項加入監聽佇列(表示為 $Queue_{listen}$)，也將此次查詢 $Query_{arrival}$ 加入查詢佇列(表示為 $Queue_{query}$)中，並結束此事件的執行。

```
EVENT Query_Arrival GIVEN Query_arrival
  //檢查此次查詢的資料項是否皆在 VS_current ∪ VS_new
  IF Data_Item_Set(Query_arrival) ⊂ { VS_current ∪ VS_new }
    RETURN THE RESULT OF THIS QUERY;
    END OF THIS ROUTINE；
  ENDIF

  //檢查此次查詢的資料項是否皆在 VS_current ∪VS_invalid
  IF Data_Item_Set(Query_arrival) ⊂ { VS_current ∪ VS_invalid }
    RETURN THE RESULT OF THIS QUERY
    AND NOTIFY USER THIS RESULT MAY BE NOT CURRENT;
    END OF THIS ROUTINE；
  ENDIF

  //將未在暫存記憶體中的資料項加入監聽佇列並將查詢加到查詢
   佇列中
  DO UNTIL Data_Item_Set(Query_arrival) IS EMPTY
    LET Data_Item = Get_a_Data_Item(Data_Item_Set(Query_arrival));
    //檢查資料項是否存在於記憶體中
    IF Data_Item ∉ { VS_current ∪ VS_new }AND Data_Item ∉ VS_invalid
      ADD Data_Item TO Queue_listen;
    ENDIF
  LOOP
  ADD Query_arrival TO Queue_query；
END
```

圖 5 處理查詢事件到達的演算法

3 驗證報告到達

處理本事件方法的演算法如圖 6 所示，而其處理流程可概分為二個步驟，分述如下：

1. 行動客戶端在下一廣播週期即將開始時(或目前廣播週期即將結束)，先將所有新版本集合中的資料項移至目前版本集合中，等完成此項工作後便進行下一個步驟。
2. 行動客戶端接收驗證報告，並在完成接收驗證報告(表示為 $Report_{invalidation}$)後，開始驗證目前版本集合中的資料項，並將已經失效的資料項從目前版本集合中移至失效版本集合中。

4 監聽事件到達

處理本事件方法的演算法如圖 7 所示，而其處理流程可概分為二個步驟，分述如下：

1. 當行動客戶端在接收到監聽資料項(表示為 $Data\_Item_{arrival}$)時，首先將該資料項識別碼自監聽佇列中刪除，接著檢查此資料項是否為失效版本集合中的資料項(檢查 $Data\_Item_{arrival} \in VS_{invalid}$)，如果是的話則該資料項自失效版本集合中刪除，再將剛接收到的資料項 $Data\_Item_{arrival}$ 加入新版本集合中，如果此資料項不是失效版本集合中的資料項，則直接將接收到的資料項加入新版本集合中。接著進行下一步驟。
2. 檢查查詢佇列中 $Queue_{query}$ 的查詢，確認是否已有查詢所要求的資料項皆完全存在於新版本集合與目前版本集合的聯集中($VS_{new} \cup VS_{curren}$)，如果是的話則回答並結束此次查詢，同時將此查詢自查詢佇列中刪除。

```
EVENT Invalidation_Report_Arrival GIVEN
Report_invalidation
  DO UNTIL VS_new IS EMPTY
    LET Data_Item = Get_a_Data_Item(VS_new);
    REMOVE Data_Item FROM VS_new
    ADD Data_Item TO VS_current;
  LOOP

  DO UNTIL Report_invalidation is EMPTY
    LET ID = Get_a_data_Item_ID(Report_invalidation );
    LET Data_Item = ID_To_Data_Item(ID);
    IF Data_Item∈VS_current
      REMOVE Data_Item FROM VS_current;
      ADD ID(Data_Item) TO Queue_listen;
    ENDIF
  LOOP
END
```

圖 6 處理驗證報告事件到達的演算法

```
EVENT Listen_Data_Item_Arrival GIVEN Data_Item_arrival
  REMOVE Data_Item_arrival FROM Queue_listen;
  IF Data_Item_arrival ∈ VS_invalid
    REMOVE Data_Item_arrival FROM VS_invalid;
  ELSE
    Data_Item_Replacement_Routine();
  ENDIF
  ADD Data_Item_arrival TO VS_new

  FOR EACH Queue ∈ Queue_query
    IF Data_Item_Set(Query) ⊂ { VS_current ∪ VS_new }
      RETURN THE RESULT OF THIS QUERY;
    ENDIF
  LOOP
END
```

圖 7 處理監聽事件到達的演算法

以上就是我們透過查詢到達、驗證報告到達、監聽資料到達三個事件來維持客戶端暫存資料一致性的方法。

(三)暫存資料的一致性

在本論文中，由於廣播伺服端配合廣播週期向資料庫伺服端一次讀取當廣播週期廣播所需的所有資料項，因此在此種運作方式下，同一週期內的廣播資料項是一致的，但是不同廣播週期的一致性是不同的。在廣播伺服端所提供的廣播資料的一致性具有此種特性之下，考慮以下情況。

每當新的廣播週期即將開始之時，行動客戶端首先會將新版本集合中的資料項移到目前版本集

合中。接著，在接收到新一廣播週期的驗證報告並完成驗證之後，再將目前版本集合中已經失效的資料項移至失效版本集合中。行動客戶端繼續等待監聽資料項到達時便接收並加入新版本集合中。如此一來，行動客戶端暫存記憶體中新版本集合與目前版本集合聯集的資料項，便與本廣播週期中的資料庫伺服端狀態為一致性狀態，而目前版本集合與失效版本集合，便與上個廣播週期中的資料庫伺服端狀態為一致性狀態。

在本研究方法中為了達到放鬆互斥一致性原則下，當查詢到達時，允許行動客戶端直接搜尋暫存記憶體中的資料項是否符合此次查詢，首先，查詢處理程式先至新版本集合及目前版本集合中的共同集合中做搜尋，若可在此共同集合中結束查詢，其結果為目前一致性的狀態，如果無法結束查詢，再到目前版本集合中及失效版本集合的共同集合中做搜尋，若可在此共同集合中結束查詢，代表此次查詢結果處於過去某一次一致性狀態下。若查詢到達時並無法在暫存記憶體中搜尋到資料項，便將此次查詢存放於查詢佇列中，等待下個廣播週期驗證暫存資料項的正確性並擷取查詢所需資料項後，再對查詢做處理，此時的查詢結果則符合互斥一致性的狀態。

五、效能評估

在本節中我們對使用版本集合放鬆互斥一致性的資料暫存方法進行一連串的實驗來進行效能評估。

(一)實驗設計

為了評估使用版本集合放鬆互斥一致性的暫存資料方法的效能，我們使用離散事件模擬的方式進行本實驗。我們所使用的工具是 CACI 公司所出品的套裝軟體 SIMSCRIPT II.5。而實驗中所使用的效能指標為行動客戶端執行查詢的反應時間(Response time)。反應時間指的是從行動客戶端提出查詢到得到查詢結果所經過的時間。

為了讓我們的方法有效能比較的對象，我們另外提出兩種原生(Naive)的資料暫存方法－保留互斥一致性的資料暫存方法(簡稱為方法 MC)與使用暫存完備狀態放鬆互斥一致性的資料暫存方法(簡稱為方法 CC)，與我們所提出的使用版本集合放鬆互斥一致性的暫存資料方法(簡稱為方法 VS)進行比較。實驗中的兩個用於比較效能的資料暫存方法分別簡述如下。

(1). 保留互斥一致性的資料暫存方法(MC)

此方法的運作方式為當行動客戶端提出查詢後，必須確定查詢的為結果滿足互斥一致性才能將結果回應給行動客戶端。在此條件下，行動客戶端的查詢處理程式會先將查詢暫存於查詢佇列中，等待下一廣播週期驗證暫存記憶體中的資料項後，再處理查詢佇列中的查詢。

(2). 使用暫存完備放鬆互斥一致性的資料暫存方法(CC)

此方法的運作方式，是當行動客戶端提出查詢後，查詢處理程式會先檢查該查詢是否為暫存完備(Cache complete)，(所謂暫存完備為到查詢所需的資料項都在暫存記憶體中)。若該查詢是為暫存完備，則查詢處理程式立即使用暫存記憶體中的資料項回應此一查詢。若該查詢不是暫存完備，則將該查詢存於查詢佇列中，等待下一廣播週期驗證暫存記憶體中的資料項後，再處理查詢佇列中的查詢。

表 2 實驗參數與預設值

符號	說　　明	預設值
d	伺服端資料庫中的資料項數目	100
u	伺服端資料更新的時間間隔.	3
l	伺服端的廣播週期	120
w	伺服端的更新視窗	2
cs	行動客戶端暫存記憶體大小	30
q	行動客戶端查詢到達的時間間隔.	3
q_l	行動客戶端每次查詢的資料項數目	2
dis	行動客戶端上線/離線的平均時間	120
p_{dis}	行動客戶端/離線的機率	0.5

接下來，我們說明實驗中所使用的參數與預設值。表 2 為本研究進行實驗所使用的實驗參數與預設值。在資料庫伺服端部份，資料項更新到達的間隔時間以服從平均時間為 u 指數分佈的隨機變數進行模擬，每一次更新的長度為 *1*，更新的資料項也為一個隨機變數服從均一分佈 $U(1,d)$，其中 d 為資料項數目。在行動客戶端部分，查詢到達的時間以隨機變數服從平均時間為 q 的指數分佈進行模擬，每次查詢長度服從均一分佈 $U(q_l/2,\ 3q_l\ /l)$，其中 q_l 為常數，各資料項被查詢的機率為 zipf 分佈。行動客戶端每隔一個隨機變數時間進行上線、離線狀態的轉換，這個隨機變數服從平均時間為 dis 的指數分佈，而上線、離線狀態的轉換則以白努利試驗方式進行，此試驗離線的機率為 p_{dis}。此外，暫存記憶體所使用的磁碟置換策略則是使用 LRU。

(二)實驗結果

我們將針對行動客戶端暫存記憶體的大小、行動客戶端查詢的平均到達時間、資料庫伺服端更新的平均到達時間、行動客戶端的離線機率等實驗參數的變動對反應時間的影響進行報告。實驗中所收集的實驗數據皆為系統達到穩定狀態時的結果。

1 實驗一、行動客戶端暫存記憶體大小對於反應時間的影響

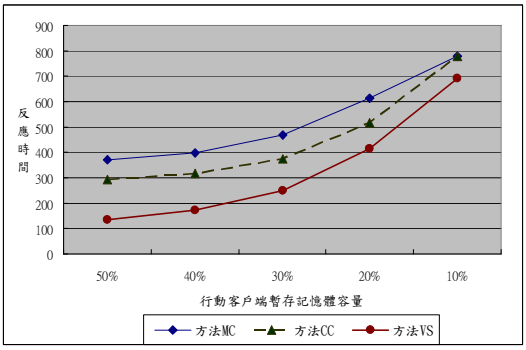

圖 8 行動客戶端暫存記憶體容量對於反應時間的影響

在本實驗中，我們在暫存記憶體的大小分別設定為資料庫容量的 50%、40%、30%、20%、 10%的條件下進行實驗。實驗結果圖示於圖 8。在圖 8 中，橫軸為行動客戶端暫存記憶體的容量，縱軸為反應時間。在圖中，我們發現在不同暫存記憶體大小下，反應時間最短的為方法 VS，其次為方法 CC，最長的是方法 MC。其原因為在方法 MC 中，行動客戶端在收到查詢之後，必須等到站存資料完成驗證才對查詢做處理。而在方法 CC 中，為了放鬆互斥一致性，對於暫存完備的查詢，行動客戶端可以不待完成資料驗證而逕行完成查詢，因此方法 CC 的反應時間比方法 MC 的反應時間來的短。在方法 VS 中，當查詢到達時，行動客戶端同樣可以不待完成資料驗證而逕行完成查詢，但是在方法 VS 中，行動客戶端暫存資料一致性較為豐富(新版本集合及目前版本集合的聯集、目前版本集合及失效版本集合的聯集)，因此，查詢在方法 VS 中完全暫存的比例高於方法 MC，也因此，方法 VS 的反應時間較方法 MC 來的短。

從圖中亦可觀察出當資料庫和暫存記憶體的大小比例由大到小時，三種方法的反應時間也跟隨著增加。其原因為當暫存記憶體的空間越小時可存放的資料項越少，當行動客戶端要求查詢時較難直接自暫存記憶體擷取資料項來回答查詢，必須等待資料項經由廣播伺服端廣播出後才能完成查詢，因此，造成暫存記憶體容量越小時效能越差。

2 實驗二、行動客戶端查詢的平均到達時間對於反應時間的影響

在本實驗中，我們將行動客戶端的查詢平均到達時間分別設定為 1、3、5、7、9、11、13 個時間單位的條件下進行實驗。實驗結果圖示於圖 9。在圖 9 中，橫軸為行動客戶端查詢的平均到達時間，縱軸為反應時間。我們從圖中發現在客戶端不同的查詢平均到達時間的情形下，反應時間最短的為方法 VS，其次為方法 CC，反應時間最長的為方法 MC。

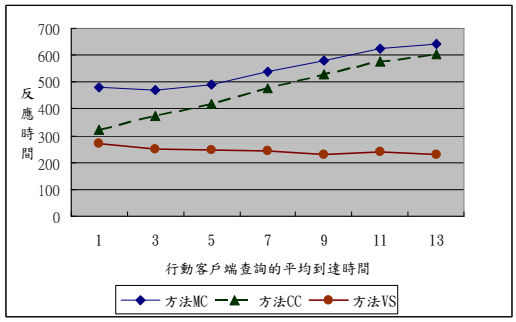

圖 9 行動客戶端查詢平均到達時間對於反應時間的影響

此外，我們也觀察出方法 MC、方法 CC 在客戶端查詢的平均到達時間間隔越大時，其反應時間也隨著增加，造成此結果原因為資料暫存方法在於監聽佇列中的內容只包含客戶端提出查詢卻未在暫存記憶體中找尋到的資料項，如此一來當查詢的平均到達時間的間隔越大時，其監聽佇列的內容也越少，間接影響行動客戶端的接收資料項數目也跟隨著減少，而行動客戶端暫存記憶體的資料項數目越少時，在暫存資料中擷取所要的資料項的機率也跟隨著降低，要等到下個廣播週期到達後才可能完全得到所要求的資料項，造成反應時間的增加。而方法 VS 中的監聽佇列如同前兩種方法的運作方式，只有在暫存記憶體中查詢不到的資料項才加入監聽，但因特別設計出的失效版本集合可存放已失效的資料項，如此一來可增加資料項在暫存記憶體中被查詢的機率，同時減短查詢的反應時間，所以當客戶端查詢的平均到達時間間隔變寬時，對於方法 VS 的反應時間影響不大。

3 實驗三、資料庫伺服端更新的平均到達時間對於反應時間的影響

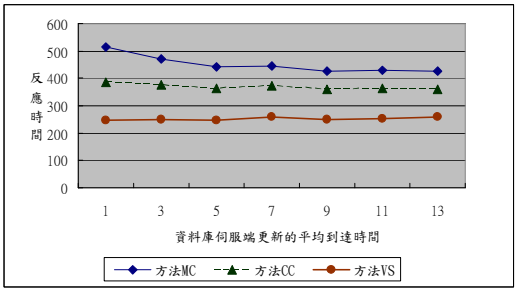

圖 10 資料庫伺服端更新平均到達時間對於反應時間的影響

在本實驗中，我們將資料伺服端的更新平均到達時間分別設定為 1、3、5、7、9、11、13 個時間單位的條件下進行實驗。實驗結果圖示於圖 10。在圖 10 中，橫軸為資料庫伺服端更新的平均到達時間，縱軸為反應時間。我們可看出在不同的資料庫伺服端更新的平均到達時間下反應時間最短的為方法 VS，其次為方法 CC，反應時間最長的為方法 MC。從圖中可看出資料庫伺服端更新的平均到達時間對於查詢反應時間影響並不大，方法 MC、方

法 CC、方法 VS 的隨著更新的平均到達時間間隔增加下反應時間差距都不大，如同前一實驗所描述，暫存記憶體中的資料項多寡主要是受行動客戶端查詢的平均到達時間所影響，雖然資料庫伺服端更新的平均到達時間會影響行動客戶端在驗證資料項時，所要刪除的資料項的多寡，但只要行動客戶端查詢的平均到達時間間隔小於資料庫伺服端更新的平均到達時間，暫存記憶體便可在短時間內放滿資料項供使用者查詢。因此造成在本實驗中方法 MC、方法 CC、方法 VS 的反應時間與資料庫伺服端更新的平均到達時間變化下並無明顯的變化。

4 實驗四、行動客戶端離線機率對於反應時間的影響

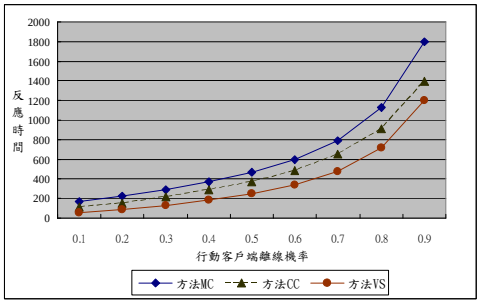

圖 11 行動客戶端離線機率對於反應時間的影響

在本實驗中，我們將行動客戶端離線機率分別設定為 0.1、0.2、0.3、0.4、0.5、0.6、0.7、0.8、0.9 的條件下進行實驗。實驗結果圖示於圖 11。在圖 11 中，橫軸為行動客戶端離線機率，縱軸為反應時間。我們可以發現在不同的行動客戶端離線機率下反應時間最短的為方法 VS，其次為方法 CC，反應時間最長的為方法 MC。從圖中可看出當離線機率不斷增加時方法 MC、方法 CC、方法 VS 的反應時間跟隨著增加，造成此結果的主要原因為當離線機率增加時，接收資料庫伺服端廣播資料的機率也同時降低，特別是方法 MC 是在接收到驗證報告後才開始處理查詢交易，若長期處於離線下會造成行動客戶端處於停滯狀態而無法處理並結束查詢，次要原因為行動客戶端在離線又重新上線後，若離線時間超出驗證報告所能驗證範圍，必須將暫存記憶體中的所有資料項刪除，等待資料庫伺服端廣播資料項時才能重新接收資料項，增加了行動客戶端在等待接收資料項的時間，並直接影響到查詢的反應時間。

綜合以上四個實驗可得知方法 VS 的反應時間皆優於方法 MC、方法 CC，其中方法 VS、方法 MC、方法 CC 的反應時間會受到行動客戶端暫存記憶體容量、行動客戶端離線機率的影響，而行動客戶端查詢的平均到達時間、資料庫伺服端更新的平均到達時間的比例不同也會影響對於方法 MC、方法 CC 反應時間，但對於方法 VS 的影響較不顯著。因此相對方法 MC、方法 CC 來說，方法 VS 的運作較為穩定，其反應時間並不受行動客戶端查詢的平均到達時間、資料庫伺服端更新的平均到達時間影響，只受行動客戶端暫存記憶體容量、行動客戶端離線機率的影響，我們可以說方法 VS 的運作較為平穩，且可比方法 MC、方法 CC 能提供行動客戶端更短的查詢反應時間，並允許行動客戶端在經常性的離線狀態下。

六、結論

在資料廣播環境當中，我們在行動客戶端設置暫存記憶體放置以減少查詢的反應時間。但是在如果查詢的結果如果還是要滿足互斥一致性的話，暫存資料需要進行資料驗證之後才能提供給查詢使用，也因此查詢的反應時間受到資料驗證的影響而無法有效減短。在本論文中，我們查詢結果從滿足互斥一致性放鬆為滿足一致性，希望藉此提供行動客戶端更快速的反應時間。在此考量下，我們提出了使用版本集合放鬆互斥一致性的資料暫存方法，我們將暫存資料區分成新版本集合、目前版本集合，失效版本集合，並設計使用這些版本集合的方法。

我們以系統模擬的方式對使用版本集合放鬆互斥一致性的資料暫存方法(VS)與保留互斥一致性的資料暫存方法(MC)、使用暫存完備放鬆互斥一致性的資料暫存方法(CC)這兩個方法比較以進行效能評估。實驗結果發現使用版本集合放鬆互斥一致性的資料暫存方法，無論在行動客戶端查詢的平均到達時間、資料庫伺服端更新的平均到時間的間隔疏密，都可維持平穩的反應時間，且對於客戶端的經常性離線，也能提供較短的反應時間。所以可證明我們所提出的使用版本集合放鬆互斥一致性的資料暫存方法，在放鬆互斥一致性的標準後能提供較短的查詢反應時間給行動客戶端，且無論行動客戶端處於連線或離線狀態時都可維持一定效能。

致謝

本論文由中華大學校內研究計畫(計畫編號 CHU-95-M-11)資助，謹表謝忱。

參考文獻

[1] T. Imielinski, S. Viswanathan and B. R. Badrinath “Data on Air: Organization and Access,” IEEE Transactions on Knowledge and Data Engineering, Vol.9, No. 3, pp.353-972, 1997.

[2] E. Pitoura and P.K. Chrysanthis, “Multiversion Data Broadcast,” IEEE Transactions on Computers, vol. 51, No. 10, October 2002.

[3] Lam K.-Y., Au M.-W. and Chan E., “Broadcasting Consistent Data to Read-Only Transactions from Mobile Clients,” The Computer Journal, vol. 45, no. 2, pp. 129-146, 2002.

[4] K.-Y. Lam, E. Chan, H.-W. Leung and M.-W. Au, “Concurrency Control Strategies for Ordered Data Broadcast in Mobile Computing Systems,” International Conference on Management of Data, Pune, India, December 2000.

[5] B. H. C. Poon, K.-W. Lam and C. S. Lee, "Broadcasting Consistent Data in Mobile Environments," 2004 IEEE International Conference on Mobile Data Management, 2004.
[6] C. R. Young and G. M. Chiu., "An Efficient Protocol for Disseminating Consistency Data in Broadcast Environments," International Workshop on Ubiquitous Data Management, 2005.
[7] J.-D. Kim and C.-K. Kim, "Response Time Analysis in a Data Broadcast System with User Cache," Telecommunication Systems, Vol. 22, pp. 119-139, January-April 2003.
[8] P. A. Bernstein, V. Hadzilacos, N. Goodman, "Concurrency Control and Recovery in Database Systems," Addison-Wesley, 1997.
[9] M. T. Ozsu and P. Valduriez, "Principles of Distributed Database Systems," 2nd Ed., Prentice Hall, 1999.
[10] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments," ACM SIGMOD Conference on Management of Data, pp.1-12, 1994.
[11] J. Jing, A. Elmagarmid, A. S. Helal and R. Alonso, "Bit-Sequences: An adaptive cache invalidation method in mobile client/server environments," Mobile Networks and Applications, pp.115-127, 1997.
[12] O. Bukhres and J. Jing, "Performance Analysis of Adaptive Caching Algorithms in Mobile Environments," Information Science, Vol.95, pp.1-27,1996.
[13] M. Carey, M. Franklin, M. Livny and E. Shekita, "Data Caching Tradeoffs in Client-Server DBMS Architectures," ACM SIGMOD International Conference on Management of Data, pp. 357-366, 1991.
[14] M. S. Chen, P. S. Yu and K. L. Wu, "Indexed sequential data broadcasting in wireless mobile computing," the 17th IEEE International Conferences Distributed Computing Systems, pages 124-131, 1997.
[15] G. Cao, "A Scalable Low-Latency Cache Invalidation Strategy for Mobile Environments," IEEE Transactions on Knowledge and Data Engineering, vol. 15, no. 5, September/October 2003.
[16] M. H. Wong and W. M. Leung, "A Caching Policy to Support Read-only Transactions in a Mobile Computing Environment," Technical Report CS TR-95-07, 1995.
[17] B. Urgaonkar, A.G. Ninan, M.S. Raunak, P. Shenoy and K. Ramamritham, "Maintaining Mutual Consistency for Cached Web Objects," the 21st International Conference on Distributed Computing Systems table of contents, pp. 371-380, 2001.
[18] Y. Wang and L. A. Rowe, "Cache Consistency and Concurrency Control in a Client-Server DBMS Architecture," ACM SIGMOD International Conference on Management of Data, 1991.
[19] Franklin M., "Client Data Caching," Kluwer Academic Publishers, 1996.
[20] K. Tan and J. Cai, "Broadcast-Base Group Invalidation: An Energy-Efficient Cache Invalidation Strategy," Information Sciences, Vol.100, pp.229-254,1997.
[21] D. Barbara and T. Imielinski, "Sleepers and Workaholics: Caching Strategies in Mobile Environments (Extend Version)", Technical Report MITL-TR-59-93, MTIL, 1993.
[22] C. R. Young and G. M. Chiu, "An Efficient Protocol for Disseminating Consistency Data in Broadcast Environments," International Workshop on Ubiquitous Data Management, 2005.
[23] R. Alonso, D. Barbara, H. Garcia-Molina, "Data caching issues in an information retrieval system," ACM Transactions on Database Systems (TODS), Volume 15, Issue 3, 1990.
[24] E. Pitoura, "Supporting Read-Only Transactions in Wireless Broadcasting", the DEXA'98 Workshop on Mobility in Databases and Distributed Systems, August 1998.
[25] J. Shanmugasundaram, A.Nithrakashyap, R. Sivasankaran, and K. Ramamritham, "Efficient Concurrency Control for Broadcast Environments", in Proceedings of ACM SIGMOD International Conference on Management of Data, Philadelphia, June 1-3, 1999.
[26] E. Pitoura, and P.K. Chrysanthis, "Exploiting Versions for Handling Updates in Broadcast Disks", in Proceedings of Very Large Data Base Conference, Sept. 1999.
[27] R Srinivasa and SH Son, "Quasi-consistency and Caching with Broadcast Disks," Second Int'l Conf. Mobile Data Management, 2001.
[28] K. L. Tan and B. C. Ooi, "On Selective Tuning in Unreliable Wireless Channels," Data and Knowledge Engineering, Vol.28, No.2, pp.209-231. 1998.
[29] S. Acharya, M. Franklin and S. Zdonik, "Disseminating Updates on Broadcast Disks," VLDB Conference, Mumbai(Bombay), 1996.