支援調適性行動計算之事件引擎
Event Engine for Adaptive Mobile Computing

吳秀陽　趙弘舜
國立東華大學資訊工程學系
花蓮縣壽豐鄉大學路二段一號
電話:+886-3-8662500-22115
傳真:+866-3-8662781
Email: showyang@csie.ndhu.edu.t

摘要

由於行動計算環境本身的不穩定與各種資源的限制，增加操作行動電腦時的不便。為了克服這樣的問題，一個稱之為調適性行動計算（Adaptive Mobile Computing）的研究領域正快速發展。所謂調適，就是應用程式可以知道環境與資源目前的狀況，然後採取適當的對策。我們以調適為發展方向，藉助 Active Databas 的概念，採用 Application-aware Adaptatio 的方式，實際開發一個系統稱之為事件引擎。事件引擎可以協助監測目前資源與環境的狀態，應用程式便能根據這些資料採取適當的反應動作進行調適，將行動環境所帶來的困擾減至最低。本文將介紹事件引擎系統，並展示調適性能與效果。同時進一步可以與其他軟體結合發展，提供應用程式在行動電腦上一個安定的執行環境。

關鍵詞：行動計算、調適、事件語言、事件偵測、反應動作、主動式資料庫

1. 簡介

在電腦資訊領域裡，不論是公司企業或個人，透過網際網路獲取資訊已成為一個重要的管道。例如，我們習慣每天上網收信、發表研究成果、提出自己的意見、甚至學生選課註冊等等，網際網路的確縮短了資訊傳遞的時間與距離。然而現今的有線網路由於缺乏移動性（Mobilit ），電腦只能固定在某一地點作業，這樣已經無法滿足我們的需求。所以無線網路可以提供我們不受時間與空間的限制，隨時隨地都可以透過網路完成資訊交換的目的。以現今社會的狀況來說，行動電腦與無線網路可以運用在很多領域。例如，醫生可以帶著筆記型電腦到每個病房為病人做最即時的診斷；企業家在火車上利用筆記型電腦收發郵件並與公司幕僚聯繫等。

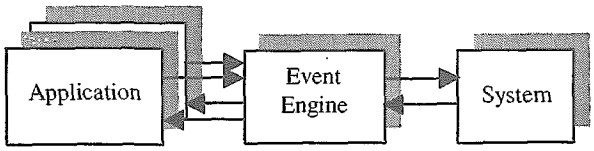
透過行動電腦與無線網路的結合應用，我們將它稱之為「行動計算」（Mobile Computing）。近幾年來，由於軟硬體技術快速發展，使得行動計算領域的研究得以實現。雖然行動計算環境帶來相當的便利，但仍有些問題需克服與解決[2][9]。例如：電力有限（batter limitation）、頻寬不穩定（bandwidth variation）、磁碟空間不足（disk space）等許多不確定、不穩定的變因，影響了行動計算領域的發展。因此，如何解決類似問題與減少這些問題帶來的負面影響是重要的課題！

為了改善這種問題，我們提出一個構想稱之為「適應」－Adaptation。發展一套稱為「事件引擎」－（Event Engine）的系統，讓電腦上的應用程式來適應行動計算環境的改變。透過系統偵測，了解現在可用資源與所處環境的即時狀況是否良好，讓行動電腦上的應用程式根據目前得知的狀況做出適當的反應、自我調整，然後發揮應有的效能。

簡言之，我們的研究目標包括：

- 提出 Event Engine 事件引擎系統架構，幫助應用程式適應行動計算環境。
- 定義 Event Language 事件語言，包括事件分類、運算元、格式等。
- 提出 Event Engine API 事件引擎應用程式發展介面，供新開發的應用程式使用。
- 提供 Change Detector 異動偵測器，能隨時監測行動電腦本身與外在環境的變化。
- 設計 Actio n反應動作，讓應用程式能選取適當的反應動作進行調適。

綜合上述各項，最終目的是希望當程式設計者在開發新軟體時，只需簡單利用我們所提供的介面，即可達到減少資源的消耗與適應環境的功能（圖一）。



【圖一】事件引擎與應用程式及系統之間的關係

2. 相關研究

行動系統依據本身以及環境變化進行調適，已被視為行動資訊管理一項相當重要的基本能力[8]。多個研究群均提出類似觀念，像是 context-aware [14]，application-aware [13]，environment-directed [15]，以及 adaptive information systems [5]等。調適的時機與方式，可以由作業系統或是應用程式本身來決行。前者稱之為 application-transparent adaptatio ，而後者則是 application-aware adaptation [12]。Application-transparent 的好處是現存的應用程式不需做任何修改即可獲益，但也意味著系統無法根據不同應用程式的特色，進行最佳的調適。另一方面，application-aware 雖然具備更高的彈性與效能等優點，其伴隨而來的軟體發展複雜度以及應用程式改寫，仍然是

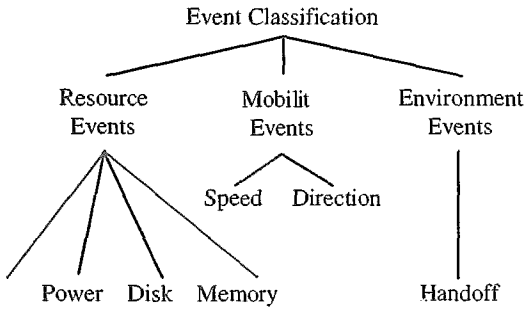
相當高的花費。我們所提出的架構[17]以及事件引擎系統，只需應用程式透過簡單的宣告，即可自動地幫所有應用程式偵測資源與環境狀態改變，並依不同應用程式的特色，進行適當的調適動作，可謂兩全其美的做法。

3. 事件語言

所謂事件，在主動式資料庫裡指的是資料庫本身或是被使用的狀態改變。針對行動計算，我們將資源與環境的改變，例如電力的強弱、頻寬的變化、記憶體的大小等稱為事件。事件亦可分為單一事件與合成事件兩種[3][4]。單一事件是只考慮一種資源或狀況的異動。為了提供給使用者更多的功能，我們透過運算元組合單一事件成為合成事件。

3.1. 事件分類

針對行動計算的特性與問題，詳加分類之後，我們提出資源事件（Resource Events）、行動事件（Mobilit Events）、與環境事件（Environment Events）等三種主要事件分類。即有關資源的可使用程度、行動電腦的位置改變、以及行動環境狀態的即時轉變與否等，我們定義合適的單位，做即時監測，增加應用程式在行動環境的調適能力，以下將一一說明（圖二）。



【圖二】行動計算事件分類

資源事件

行動計算最大限制之一就是資源有限。在資源事件中我們提出：硬碟（disk）、記憶體（memory）、電源（power）、頻寬（bandwidt ）等四種最重要的資源做為偵測的對象，分述如下：

- 硬碟：（單位：Disk Free MB ）。相較於桌上型電腦，由於重量、體積與抽換便利性等因素，可攜式電腦的硬碟容量受到限制。因此我們必須留意硬碟容量剩下多少空間可以使用。
- 記憶體：（單位：Memory Free M 或 Memory Free Percentage）。雖然標準桌上型電腦的記憶體容量越來越大，但對於行動電腦來說，記憶體容量還是有限，擴充不易。因此，我們在意的是記憶體還剩下多少容量可以使用。
- 電源：（單位：Power Remain Seco n 或 Power Remain Percentage）。由於行動電腦的移動性，不可能長時間依賴固定電源，電池供電時間的長短便影響行動電腦能有多少時間可以運作，因此我們關心電池可以使用的時間有多長。
- 頻寬：（單位：KB Per Second）。現今網際網路的頻寬使用已達飽和，對於原本頻寬小的行動電腦來說，更是一項嚴酷的限制。所以隨時偵測網路的流量，了解網路現在通暢與否是必須的。

行動事件

行動電腦必強調其移動性（Mobilit ）。由於電腦隨處在移動，因此，如何讓其他電腦能選擇最短路徑傳送資訊到正確位置是重要的。我們將行動事件分為速度（speed）、方向（direction）等兩種重要的事件。

- 速度：（單位：km/secon 或 High / Low）。我們可以帶著電腦在交通工具上或在其他移動的狀況下做操作。速度的快慢會影響收訊的穩定與否。在訊號不穩定時，便要隨時調整。
- 方向：（單位：無）。判斷往哪個方向移動，訊號強度是否增強或減弱，可以用來判斷電腦是正在遠離或是接近我們，並可以追蹤到目前電腦的位置。

環境事件

影響行動計算的因素中，以環境狀態的隨時改變最為直接。所以在這裡以換手（hand-off）為主要監測的環境事件。

- 換手：（單位：無）。行動電腦由一個 cell 跨越至另一個 cell，這個動作稱為換手。我們可以藉由行動電腦是否經過換手，了解電腦目前的位置與該位置的無線傳輸狀況是否良好。另一方面進行換手時，可能必須改變電腦內部的一些設定值，來配合新 cell 的環境。由於換手為一種過程，所以我們並不特別定義標準單位。

3.2. 運算元

前述各項直接因為資源或環境改變所引發的事件，稱之為單一事件或是原始事件(Primitive Events)。由於單一事件不能完全涵蓋所有可能發生的情況，所以利用運算元，將原本的單一事件組合成合成事件(Composite Events)，如此可以掌握更多的狀態改變是否發生。在事件引擎中，我們提出 AND 與 OR 兩種運算元，形成合成事件。

- AND（且）：定義當“ E1 AND E2 ”時，表示 E1 與 E2 都發生時，事件才成立。
- OR（或）：定義當“ E1 OR E2 ”時，表示 E1 或 E2 兩者之中只要有一個發生，事件便成立。

3.3. 事件宣告格式

針對事件的定義，我們提出三種語法格式：

```
ON <Event_Name> <OP> <Value> <Unit>
ON <Event_Name> LEVEL <Level>
ON <Condition>
```

其中，＜Event_Name＞是資源事件或是行動事件的名稱。＜OP＞是關係運算元，像是大於(>)、等於(=)、小於(<)等。＜Value＞則搭配與＜Unit＞使用。＜Level＞

定義有 Hig 、Normal、Low 三種不同的程度。<Level>與<Value>之間主要的差別在於<Value>格式提供較為詳細的偵測值，對於環境的任何改變能提供立即的反應，屬於絕對的偵測值。而<Level>則提供程度上的差異性，主要目的是希望使用者依據自己所需，在不同的程度中，自己定義（user-defined）適合的條件值，例如可以自己定義 100-60 的範圍值為 Hig ，59-0 的範圍值為 Low 等，是屬於相對的偵測值。例如：資源事件

ON Bandwidth < 100 kbps

表示當目前電腦與目的地之間的網路頻寬小於 100kpbs 時，便執行反應動作。同樣的：

ON Speed LEVEL Hig

表示行動電腦移動的速度已高於之前所定義的程度時，我們需做觸發的動作。

對於 ON <Condition>這個語法格式，由於並不需要有任何數值做為判斷的依據，基本上，我們只要是對有興趣的環境狀況都可以使用。目前只包括 Hand-Off。例如：

ON Hand-off

表示行動電腦在換手時，我們需做觸發的動作。

資源事件、行動事件、環境事件，透過以上的格式可以形成單一事件。我們可以再加入之前所提出的運算元 AND 與 OR，組合成為合成事件。例如：

ON Disk < 100 mb AND ON Hand-off

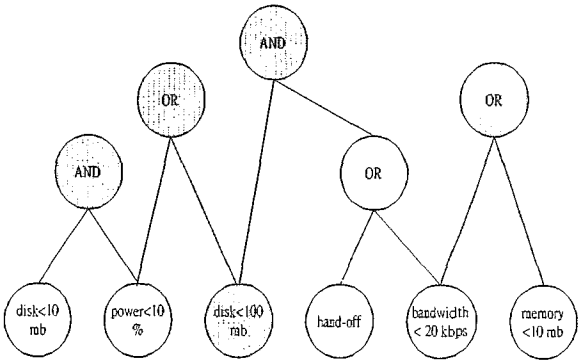
表示當電腦硬碟容量小於 100MB 而且在做換手動作時，我們需做觸發的動作。

3.4. 事件林

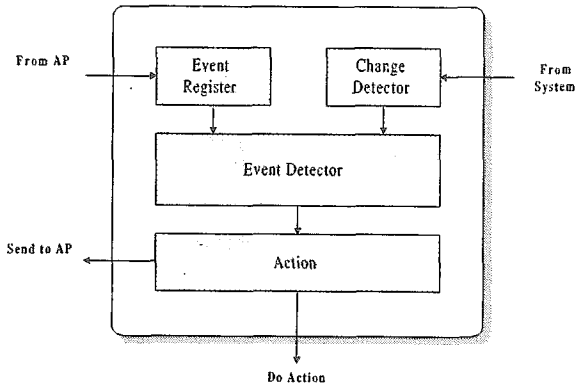
當應用程式要使用事件引擎時，必須先註冊所需偵測的事件以及選擇採行的反應動作。事件可為單一事件或合成事件。我們同時要面對一個、兩個、甚至三個以上的應用程式，這麼多個應用程式在事件引擎中註冊時，需要一個良好的方法來管理這些事件，並避免對應用程式之間相同的事件進行重複偵測。所以，在這裡我們提出一個方法，稱為「事件林」(Event Forest)。圖三為一個事件林。最底層的樹葉節點代表單一事件，例如 "Disk < 10 mb" 。若應用程式只註冊一個單一事件時，我們便將應用程式的識別值與反應動作存入這個節點。除了最底層之外，以上的每個節點為運算元 AND 或 OR。這是因為當事件為合成事件時，我們利用運算元來表示每個節點之間的關係，如圖三 "(Disk < 10 mb) AND (Power < 10 %)" 。這時候，若應用程式註冊合成事件，我們便將應用程式的識別值與反應動作存入 AND 這個節點。如果有應用程式註冊相同的事件，則節點裡便存在一個以上的識別值與反應動作。利用上述的方法，我們將每個應用程式註冊的事件做完整的分類。單一應用程式所註冊的事件稱為「事件樹」(Event Tree)，將許多應用程式的事件樹組合起來便成為一個事件林。

4. 事件引擎

事件引擎設計的靈感來自於主動式資料庫架構[16]，其構成包括：事件註冊單元（Event Register）、異動偵測器（Change Detector）、事件偵測器（Event Detector）、反應動作（Action）等四大部份。簡單來說，應用程式透過事件註冊單元將需要偵測的事件，以及當事件發生時應採取的調適反應動作告知事件引擎；異動偵測器協助監測行動電腦資源與所處環境的任何變化；事件偵測器將異動偵測結果與事件註冊單元中的資料做分析比較，判斷事件是否成立。當事件成立時，再啟動系統預設或是應用程式所註冊的反應動作進行調適。



【圖三】事件林



【圖四】事件引擎的實體架構

4.1. 設計目標

如前所述，在行動計算環境裡，應用程式必須有調適的能力。主要有兩種方式來達成[12]：一種稱為 Application-aware Adaptation，這是讓應用程式自行選擇對自己最有利的方式來做適應；另一種為 Application-transparent Adaptation，所有調適的責任由作業系統中央統一處理。後者的好處是應用程式不需要了解目前資源使用的狀況。但對行動計算環境的電腦與應用程式來說，則缺乏依據應用領域或是程式特性的不同而隨時機動進行最佳化調適的能力。因此，我們所提出的事件引擎採用 Application-aware Adaptatio 策略，對調適性行動計算提供解決方法。

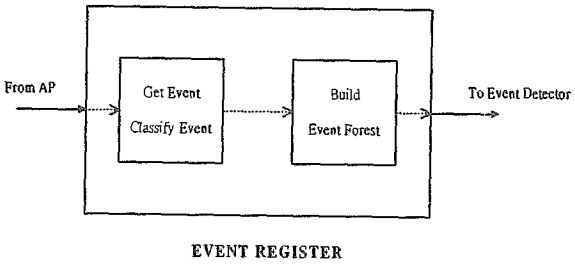
4.2. 系統架構

如圖四，事件引擎的實體架構包含四大單元：事件註冊單元、異動偵測器、事件偵測器、反應動作。我們同時提供事件引擎介面函式，供新開發的應用程式使用。下

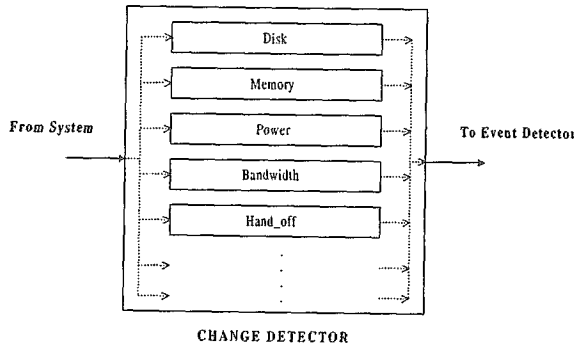
面將介紹各單元的功能與組成。

事件註冊單元

如圖五，事件註冊單元是事件引擎與應用程式間，最先接觸的溝通管道。應用程式註冊的事件必須從這裡進入事件引擎，然後逐一分門別類建置。註冊單元的設計以事件林為基礎，構成一綿密的事件控制網。



【圖五】事件註冊單元



【圖六】異動偵測器

異動偵測器

如圖六，異動偵測器隨時偵測環境狀態的改變。當偵測值與事件值相比較之後，若事件成立了，便通知應用程式做適當的反應動作。例如，當應用程式要求在電池的電力小於 10% 的時候，要做 power suspend 的動作。所以，我們要能監測與判斷目前的電力值是否小於 10% 。至於監測系統資源值（即硬碟剩餘空間、記憶體剩餘容量、電力剩餘量等），主要是利用開發工具 Visual C++所提供的函式來完成。

事件偵測器

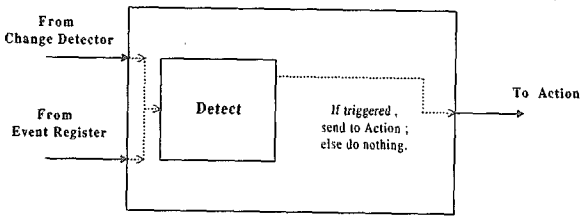
圖七為事件偵測器，我們每隔固定時間將偵測到的資料與事件林中的樹葉節點做查詢（pollin ）比較的動作。若這時有某個事件成立的話，這個節點便被記錄下來（marked）。然後開始查看本節點是否有應用程式的識別值，若有的話，便執行相對應的反應動作；若沒有，則忽略不管。接著，往上通知與本節點相連的節點，這個時候將本節點記錄取消（unmarked），然後繼續對下一個節點重複相同的判斷動作。最後，當應用程式要結束執行時，我們便根據識別值把相對應的反應動作消除。我們所設計的事件林，可以完整的處理所有應用程式註冊的事件，並帶來相當程度的便利與效率。

反應動作

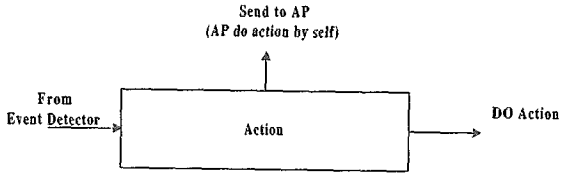
反應動作需等待事件偵測器通知才進行運作。這時，必須知道應用程式是使用何種反應動作，然後執行如圖八。我們將反應動作的宣告格式定義如下：

DO <Action> [parameters]

其中＜Action＞為反應動作，〔 parameters 〕為參數，參數這一欄可視反應動作的形式為何，再決定加入或不



【圖七】事件偵測器



【圖八】反應動作

加入參數。例如：

DO power_suspend

表示執行電源暫時切斷的動作，進入省電模式。此反應動作不需要參數。又例如：

DO print c:\log\syslog.txt

表示列印出在硬碟 C 槽 log 目錄下的 syslog.tx 檔案。其中，c:\log\syslog.tx 為參數。

4.3. 事件引擎函式介面

在完成系統架構建立後，我們提出應用程式發展介面，以利新的應用程式使用事件引擎：

```
Registration(AP_id, Event, Action)
```

Registrat 是事件引擎函式的名稱，其中包括三個參數：AP_id、Event、Action，皆以字串處理。

- **AP_id**：應用程式識別值。每個應用程式都有屬於自己唯一的識別值，彼此都不相同，所以我們用識別值來代表每個不同的應用程式。
- **Even** ：註冊事件。Ev e n參數是我們事件引擎的核心部份，應用程式對任何有興趣的事件都由這裡填入。例如，應用程式 AP1 想在硬碟容量小於 20mb 且電力剩餘小於 10%時，做 power suspend 反應動作，則 Ev e n 參數以 preorder 的方式填入，

即“AND (disk < 20mb) (power < 10%)”，一方面直接避免曖昧語意的情形發生，另一方面減低系統運算與判斷的負擔。

- **Actio** ：反應動作。這部份根據我們所提供的反應功能，應用程式可自行選擇所需，例如：“(defrag)”、“(power_suspen)”等。

以上三個參數簡單便利且提供完整功能給應用程式運用。由函式可明顯看出，只要把事件與希望執行的反應動作填入，事件引擎即開始幫助處理，達到調適的目的。事件引擎函式是我們發展本系統中相當重要的一部份。應用程式透過函式即可便利的使用事件引擎系統，達到適應行動環境的目的。

5. 效能分析

我們設計了一個類似 Word Pad 記事本的文書編輯程式，稱為 Super Cinatit Editor（SCE）使用事件引擎，並設計不同的事件來展示事件引擎所帶來的效益。

電力調適

行動電腦相當倚賴電池，為了延長電腦的可使用時間，我們採用暫停電源的方式來做調適。由實驗數據得知，電源暫停的方法可以延長電池壽命，達到數小時以上。如圖九，橫軸代表在電力分別剩多少時，啟動調適動作。縱軸代表電腦從開機到電力耗盡的時間。我們發現，在電力剩下 80%的時候啟動電源暫停，可以讓電力延長至十小時以上！所以，採用電源暫停的方式是延長電腦使用時間很好的辦法。

記憶體調適

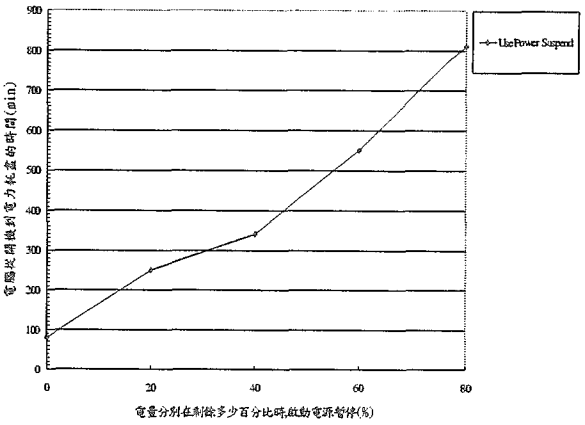
在這裡我們主要觀察實體記憶體的容量變化，其中 Physical Memory 為實體記憶體目前的使用量；Other Memory 為硬碟快取頁、記憶體固定配置檔、記憶體對應檔目前的使用量；Allocated Memor 為系統中所有記憶體配置量。

為了顯示記憶體的變化狀況，本實驗的進行採用一步步操作，順序如下：1.尚未開始執行任何程式、2.開啟檔案總管、3.開啟註冊了"(Disk < 10mb)" 事件的 Super Cinatit Edito（SCE）、4.複製圖檔文字資料、5.開啟 ACDSee32、6.事件觸動，採用關閉系統列調適、7. 事件觸動，採用清除剪貼簿調適。由圖十的數據可以發現，當我們一步步開啟應用程式時，記憶體也開始消耗。尤其是開啟 ACDSee32 時，使用相當多的記憶體。這時候事件成立，執行反應動作，關閉系統列中所有程式，然後清除剪貼簿中的資料，釋放出部份記憶體，最後達成調適目的。特別是在操作順序 4、5、6、7 中，記憶體容量在調適前後有明顯的差別，先從使用 50MB 增加到 56MB，然後經過調適降為 49MB。在調適後，記憶體可以使用的容量回升到先前未開啟任何程式時的水準。圖十一是利用 Windows95 資源監視程式觀察可使用資源的變化，在調適之後，資源明顯增加。

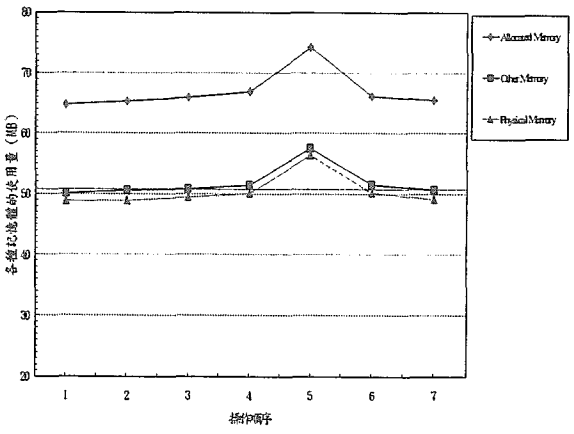
頻寬調適

行動計算環境的頻寬不足，所以必須針對頻寬做適當的調適。一般來說，在傳輸較大的檔案會耗費較多時間，較小的檔案則耗費較少時間，所以本實驗以瀏覽器為例，在低頻寬時，我們只傳送文字資料而忽略圖檔不傳送，如此可以減少傳輸的時間，並且在相同的單位時間內完成較多的工作，達到調適的目的。詳細實驗結果如圖十二。我們分別在有線網路與無線網路環境中，讀取四個網站的資料來做比較。我們可以明顯發現，無線網路平均所花費的傳輸時間比有線網路的傳輸時間長，而且傳送全部資料比只傳送文字資料所花的時間長。所以相比較之下，當頻寬不足時，若我們進行調適的動作，則單位時間內將完成較多的工作，而且可以減少傳輸時間。

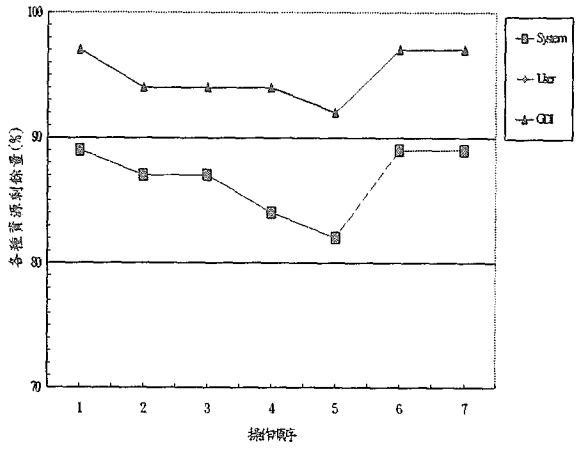
綜合以上每個實驗的結果發現，不論是可使用量或可使用時間，每種資源在調適後均有明顯增加，表示我們提出的調適架構與機制的確能達到相當不錯的效能。



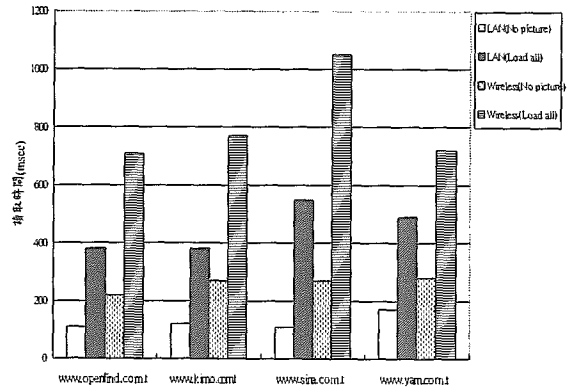
【圖九】採用電源暫停方式進行調適



【圖十】記憶體調適過程中,容量的變化狀況



【圖十一】在記憶體調適過程中,SYSTEM、USER、GD 資源的使用狀況



【圖十二】頻寬調適比較

6. 結論與展望

行動計算所帶來的問題[1]，造成在操作行動電腦的時候，會增加許多不穩定的因素。我們藉助 Active Database 的概念，以 Application-aware Adaptatio 為方向，實際開發一個系統稱為事件引擎，幫助應用程式適應多變的行動環境。在實際運用事件引擎並比較系統資源調適前後的差異之後，整體結果令我們相當的滿意，各方面的資源可使用量均明顯增加，達到我們預期的調適結果。

調適性行動計算是一個正在發展的領域，許多學術單位正致力於這方面的研究，並提出相關的解決方法[7]。而事件引擎在處理資源事件上已達不錯的成果，未來可進一步與其他程式結合發展。使系統能更準確的篩選資訊，提高執行效率。另外，我們的目標是發展可適應行動環境的資訊系統[17]，可以運用在校園或醫院等環境。在這套系統中，最重要的部份是對於環境與資源狀況的掌握，而事件引擎正可以幫助達成這個目的。所以，事件引擎可以運用在很多方面，幫助其他程式適應行動環境的狀況，讓整個系統效能提升。

參考文獻

[1] B. R. Badrinath and T. Imielinski, "Mobile Wireless Computing : Solutions and Challenges in Data Management" Rutgers University, New Brunswick, NJ 08903

[2] G. H. Forman and J. Zahorjan, "The Challenges of Mobile Computing" IEEE Computer. Vol.27 , No.4, 1994.

[3] S. Gatziu, K. R. Dittrich, O. Shmueli, "Composite Event Specification in Active Database : Model and Implementation" AT&T Bell Lab., Murray Hill, New Jersey, 1992.

[4] N. H. Gehani, H. V. Jagadish, "Event Specification in an Active Object -Oriented Database " Universit Zurich, Switzerland, 1993.

[5] T. Imielinski and S. Vishwanathan. "Adaptive Wireless Information Systems". Tech Report, De pt. Computer Science, Rutgers University, 1994.

[6] H. Jasper "Active Database for Active Repositories" Proc. 10th Intl. Conf. On Data Engineering, Feb. 1994.

[7] A. Joshi, S. Weerawarana, R. A. Weerasinghe, T. T. Drashansky, N. Ramakrishnan, E. N. Houstis, " Survey of Mobile Computing Technologies and Applications" Department of Computer Science, Purdue University, October 1995.

[8] R. H. Katz. "Adaptation and Mobility in Wireless Information Systems", IEEE Personal Communication, 1(1):6-17, 1994.

[9] B. Marsh, "System Issues on Mobile Computing" Matsushita Information Technology Laboratory, Technical Report MITL-TR-50-93, 1993.

[10] B. D. Noble, M. Price, M. Satyanarayanan, "A Programming Interface for Application-Aware Adaptation in Mobile Computing" Carnegie Mellon University, CMU-CS-95-119, February 1995.

[11] M. Satyanarayanan, "Fundamental Challenges in Mobile Computing". School of Computer Science, Carnegie Mellon University, 1996.

[12] M. Satyanarayanan, "Mobile Information Access". IEEE Personal Communications, 3(1), Feb. 1996.

[13] M. Satyanarayanan, B. D. Noble, P. Kumar, M. Price, "Application-Aware Adaptation for Mobile Computing" Operating System Review 29, Jan. 1995.

[14] B. Schilit, N. Adams, and R. Want. "Context -aware Mobile Applications". IEEE Workshop on Mobile Computing Systems and Applications, Dec. 1994.

[15] G. Welling and B. R. Badrinath. "Mobjects Programming Support for Environment Directed Application Policies in Mobile Computing". ECOOP'95 Workshop on Mobility and Replication , Aug. 1995.

[16] J. Widom and S. Ceri. Active Database Systems Triggers and Rules for Advanced Database Processing. Morgan Kaufmann, San Francisco, CA, 1996.

[17] S. Wu, and C. S. Chang, "An Adaptive Database Framework for Adaptive Mobile Data Access." Workshop on Mobile Data Access, 17th International Conference on Conceptual Modeling, Singapore, 1998.