IBM WebSphere Portal：效能測試與分析

等級：中級

IBM 入口網站資深工程師 Alex Lang (alex_lang@us.ibm.com)

2008 年 7 月 1 日

本文提供建議和方法 ，以尋找和解決 IBM® WebSphere® Portal 的一般效能問題。WebSphere Portal SEAL 團隊會與遭遇重大 WebSphere Portal 部署問題的知名客戶接觸，其中約有 50% 的個案主要涉及效能的相關問題。在系統正式作業之後，主要任務通常會變成尋找和解決這些效能問題。

就許多層面來說，在正式作業環境中尋找和修正效能問題是一項挑戰。在最理想的情況下，系統在正式作業之前，應該先找出並修正大部分的瓶頸。本文說明一項經過測試的程序，可確保系統在正式作業之前，找出並解決大部分重要效能問題，且成功率相當高。效能測試與分析的主要目的有三個：

1. 判定受測試的系統是在哪一個負載層次失敗。
2. 找出系統中阻斷作業的瓶頸，並儘快在可實行的情況下，移除瓶頸。
3. 產能規劃，例如，預測在「服務水準協定 (SLA)」內，已定義使用者負載量所需之馬力。

系統是定義爲一套完整的端對端元件，需要傳遞所要求的網頁至使用者瀏覽器。最明顯且最常發生問題的元件爲 WebSpherePortal 本身、Websphere Portal 資料庫、輕量型目錄存取通訊協定 (LDAP)，以及提供內容給 Portlet 的後端系統（資料庫、應用程式伺服器等等）。

在 WebSphere Portal 部署中，許多系統的後端系統最常出現風險，因爲後端系統是由不同的組織維護。就效能目的而言，這種維護方式會阻礙 WebSphere Portal 部署團隊與後端團隊之間的通訊管道。

本文所提供的方法若執行成功，可以同時達成上述三項目的。

環境

爲達成上述的效能測試目的，效能測試環境必須是正式作業環境或其鏡映，若是鏡映，則儘可能具有相同的硬體、拓蹼，以及相同的後端系統。如果此複合測試拓蹼有任一環節與正式作業的對應項目不同，則必須推斷測試環境中的結果，以預測於正式作業環境中的影響。這些推斷通常需要對入口網站和部署之應用程式有詳細的實作知識，而測試組織通常在這方面也很缺乏。若能讓測試環境與正式作業環境相同，您就可以有信心，正式作業中實際發生的情況會與測試結果相同。

測試環境的一項重要目標，就是結果的可重複性。系統有輕微變更時，可重複性可確保您能夠精確地測定這些變更的影響。因此，在效能測試期間，最好是將系統放在隔離的網路上。在正式作業網路上執行效能測試會造成變異（例如，使用者資料流量），可能扭曲網頁轉譯回應時間等測量值。將效能測試網路隔離還有另一個實務上的理由。加重 WebSphere Portal 的負擔，也很可能增加公司網路的負擔，在平常的上班時間經常會造成問題。

如果沒辦法將效能測試放在隔離網路上，至少也要確保測試的元件全放在網路路由器的相同子網路上。一般 WebSphere Portal 最佳實務建議在入口網站及其資料庫之間使用 Gigabit 乙太網路連線，最好延伸至 LDAP 伺服器、Web 伺服器及重要的後端服務。負載產生器也要在 Web 伺服器及／或入口網站本身的本端區段上，這是很重要的。

將負載產生器放在與入口網站本身相同的本端 LAN 區段上，一般客戶也會有所顧慮。在此情況下，「此測試無法呈現系統效能的真實情況，因爲將資料中心至使用者的網路排除。」而客戶通常難以接受這個回覆。這裡說明的程序是爲了調整及解決入口網站及其週邊元件的問題。嘗試調整使用者（或負載產生器）與入口網站之間的網路，會讓分析及問題解決的過程更爲複雜，因此，我們在測試中移除這個項目。相較於這裡使用的程序，有許多工具和程序更適合用來調整網路。

入口網站架構基準線

相對於鏡映的正式作業環境，強烈建議同時進行一套漸進式的基準線測試，以操練該架構。進行測試時，後續測試應以客戶撰寫的程式碼逐漸擴大入口網站。因此，測試計劃應從簡單的拓蹼移至最終的正式作業拓蹼，這樣比較容易隔離有問題的元件。

首先，使用立即可用的入口網站，測試完整的 WebSphere Portal 架構。傳送資料庫，並啓用安全功能。確定所有前端 Web 伺服器、防火牆及負載平衡器都已就位。同時，也應採用並正確配置安全管理程式（例如，Computer Associates SiteMinder 或 IBM Tivoli® Access Manager）。使用不會存取任何後端系統的 Portlet 來建立簡單的首頁（例如，World Clock Portlet）。建立一個簡單的負載測試 Script，以存取未經鑑別的首頁，然後登入（鑑別），並在未登入的情況下閒置。從這時候開始，要新增模擬使用者 (Vuser) 直到系統飽和。利用以下的瓶頸分析技巧，找出並修正架構中的任何瓶頸。請注意此系統的效能基準線。

現在，將任何自訂的主題及面板新增至系統，並重複先前的測試。在修改過的系統中，找出並修正任何重要的瓶頸。最後，依照下面的說明，將要使用的實際 Portlet 新增在首頁上，並執行瓶頸分析。

在沒有應用程式支援的架構中尋找瓶頸時，此基準線環境非常有效率。此外，在應用程式分析額外負載超過基本 WebSphere Portal 架構的程度時，還可提供參照。

您的策略是進行下列的相同測試，以在這個基準線環境中執行瓶頸分析、將環境最佳化，然後以實際的應用程式來執行瓶頸分析。

Portal Tuning Guide 的應用建議

進行任何效能測試之前，請先將 WebSphere Portal Tuning Guide 的建議套用至所有系統，該指南的內容是很好的著手點，因爲可以修正預設 WebSphere Portal 安裝中已知的效能障礙。雖然瓶頸分析可能找到相同的問題，但最好在最初即解決那些問題。

負載產生

適當的效能測試也需要使用負載產生器，以產生使用者對網頁提出的模擬要求。最重要的是，此工具會產生回應時間和每秒檢視頁數等測量值，您可以藉以判斷系統什麼時候不符合 SLA 合約，或是達到飽和點，即使在每單位時間投入更多頁面要求，也不會產生更多頁面。本文稍後會再討論飽和的相關事項。產生器彙總資料的能力，例如彙總入口網站和 HTTP 伺服器上的 CPU 使用率，以及來自 HTTP 伺服器的 mod_status 資料，都有助於判斷問題。

有數項工具經常用來建立測試系統的 Web 資料流量（也稱爲磁碟機負載）。較常用的工具包括 Mercury LoadRunner、Borland SilkPerformer 及 IBM Rational® Performance Tester。

負載產生器要有足夠的虛擬使用者 (vUser) 來促使系統達到飽和狀態。請注意，虛擬使用者不會直接對映至實際使用者。虛擬使用者代表負載產生器上的作用中通道。一個虛擬使用者可模擬多個實際使用者，但是，每個虛擬使用者只能有一個作用中的實際使用者。

除此之外，在 WebSphere Portal 環境定義中，系統是否需要經鑑別才能存取測試中的 WebSphere Portal 應用程式、LDAP 目錄中要有足夠的唯一測試使用者 ID，以及 Script 要確保在測試期間，只有合理數量的重複登入。（在此環境定義中，合理數量是指有些使用者可能開啓了數個瀏覽器，每個實例均使用相同的 WebSphere Portal 登入 ID。）WebSphere Portal 的入口網站構件有很大的快取架構，通常會依據使用者來快取這些構件。如果負載模擬在所有測試中也使用相同的使用者 ID，效能就會顯得異常地高，因爲不需要從 LDAP 目錄和資料庫載入那些構件。

一般方法

下列各節說明用來實際分析系統的反覆程序。「程序」之前的段落會定義該程序執行期間所必須瞭解的概念。

使用者情境

若要調整 WebSphere Portal 系統，以處理大量使用者，以及精準地預測其正確處理特定使用者數量的能力，那麼判斷出系統使用者最適合的情境就很重要。該測試必須使用負載產生器，精準地模擬那些使用者情境。若要有效執行此步驟，可以列出最可能使用的案例。爲每個使用案例或是依實際需求撰寫 Script。現在，請指定所有使用者可能執行該情境的機率。執行測試時，以預期的一般人數比例來指定使用案例給 Vuser。Vuser 數量突然增加時（稍後討論），請嘗試維持此比例。

附註：「Vuser」是 LoadRunner 的術語，代表提出及傳回要求的一個作用中通道。

思考時間

思考時間是使用 WebSphere Portal 的過程中，每次按下滑鼠或鍵盤按鍵之間的平均暫停時間。在負載產生工具中，這個時間通常可供程式設計，在預先定義的範圍內產生隨機的時間。

思考時間減少時，每秒的要求數會增加，進而增加系統上的負載量。減少思考時間通常會增加 WebSphere Portal 登入及轉換瀏覽頁面的平均回應時間。因此，若要在正式作業中產生準確的系統模型，精準地估算真正的使用者思考時間是很重要的。

在大部分的使用案例中，若入口網站的使用者均具有經驗，則 10 秒的思考時間加減 50% 是合理的範圍。若入口網站的使用者沒有經驗，則接近 30 秒的數值較爲合理。

Cookie 及階段作業

一般來說，大部分的真正使用者都會登入至入口網站，並執行需要完成的工作；然而，他們很少會用登出按鈕來登出，而是讓瀏覽器閒置，直到階段作業逾時。通常記憶體中會有很多階段作業等待清除，而擱置 WebSphere Application Server 階段作業逾時，導致 Java™ Virtual Machine (JVM) 資料堆工作集增加，進而增加 JVM 資料堆耗盡的機率。資料堆耗盡會造成效能瓶頸，並導致 JVM 故障。

有效的模擬作業必須模擬使用者沒有明確登出。每個模擬作業執行特定的使用案例時，應以進入閒置狀態的方式來結束使用案例，而非登出。Script 循環回來且在此特定 Vuser 登入新使用者時，舊階段作業（通常是 JSESSIONID）及「輕量型協力廠商鑑別 (LTPA)」的 cookie，以及任何應用程式特定的 cookie，都必須先適當地清除，才能使用該 Script 來登入下一個使用者。此模型也意味著，必須要有足夠的測試 ID，測試 ID 的閒置時間才能一直維持到 WebSphere Application Server 階段作業逾時，而不會在前一個階段作業逾時之前，出現重複使用的風險。

測量值

Script 必須具有測量功能，最重要的測量值是登入回應時間，以及頁面轉換回應時間。大部分的負載產生器都有提供「每秒檢視頁數 (PV/s)」測量值。PV/s 是判斷系統飽和點最重要的測量值。

在每項測試結束時，需要 Vuser 上升率與三項測量值的圖表來進行分析。

除了負載產生工具所收集的測量值，也應採用系統監視工具，例如，IBM Tivoli Composite Application Manager for WebSphere 或 Computer Associates Wily IntroScope 產品。這些工具會在 WebSphere Portal 實例上執行，並直接做爲 JVM 的工具，在偵測及解決系統、vUser 或思考時間的瓶頸方面，都很有用。

有不少人認爲，若要精準地模擬以特定速率產生要求的大量人數，只要用較少思考時間來產生要求的較少使用者即已足夠，這種觀念其實是錯的。

以較少使用者及思考時間來執行結果，會讓快取命中率過高，即已建立的階段作業太少。對許多 **Portlet** 應用程式來說，階段作業大小是嚴重的問題，所以這個方法會讓您看到很好但不實際的系統效能，以至在正式作業時出現狀況。

另一種不恰當的做法，就是在沒有思考時間的情況下執行一小組 **vUser**。

可重複性原則

在大量人數中，使用者在瀏覽入口網站時，大部分的使用者動作均會假設爲隨機進行。然而，有經驗的使用者通常會不斷重複使用相同的模式。此外，從測試工程的觀點來看，使用者情境必須相當固定，才能從每次的測試有效地測量出系統變更。

因此，可重複性原則的定義是，對特定情境的所有測試而言，如果執行的時間夠長，則所產生的測量值（平均回應時間、**PV/s**、飽和點等等）全都會綜合出相同的結果。需要注意的是，測試 **Script** 的變化愈多（即獨一的情境），綜合結果所需的平均時間就愈長。

爲效能測試而撰寫的模擬 **Script** 應遵守可重複性原則。

達到飽和

飽和的定義爲，**Vuser** 增加而 **PV/s** 數目沒有增加時，作用中 **Vuser** 的數目。此飽和點適用於給定的模擬作業，每個不同的模擬作業可能有不同的飽和點。飽和點會依據使用模式而有所不同。

若要有效讓系統達到飽和，請每次新增些許 **Vuser**，讓系統保持穩定，觀察 **PV/s** 是否增加，再新增更多 **Vuser**。（在本文中，「穩定」的意思是在數分鐘內，回應時間能保持穩定。）在 **LoadRunner** 上，如果繪製 **Vuser** 與傳輸量 **(PV/s)** 的圖表，**PV/s** 一開始會隨 **Vuser** 的數量直線上升，然後達到上限，再從該點稍爲回落。飽和點是 **PV/s** 達到上限時的 **Vuser** 數目。

瓶頸分析

瓶頸分析的目標是要消除系統提高負載量的障礙。爲較高負載量定義的測量值，是飽和時較高的 **PV/s** 數。因此，瓶頸分析可消除提高飽和點的障礙。

WebSphere Portal 環境中的負載瓶頸，通常都是因爲競用共用資源所致。造成競用的原因，可能是 **Java** 類別、方法或資料結構的同步化、連續的資源競用（例如 **SystemOut.log**），或是後端資料庫或 **Web** 伺服器的回應時間過長。您也必須注意像網路本身這種瓶頸。路由器和防火牆等元件可能會增加擁塞控制的負擔，或是難以調整。

隨著負載量增加，資源競用的情況也會增加，較容易偵測及更正競用鎖定。正是因爲這個小細節，瓶頸分析會需要有效負載測試。

常見的錯誤是只專注在頁面回應時間，很多效能測試人員喜歡將轉譯回應時間最佳化，因爲這項延遲是最明顯的使用者需求。這類型的效能分析需要減少客戶 **Portlet** 應用程式的路徑長度。一般而言，在無負載的系統中，使用作業特定的工具（例如 **JProbe**）來執行回應時間最佳化比較適當。

程序

瓶頸分析的執行程序很簡單，若要進行特定的效能分析（例如 **LoadRunner**）模擬，請進行下列步驟：

1. 讓單一 **WebSphere Portal JVM** 提高至飽和點。
2. 判斷飽和時的瓶頸。
3. 解除瓶頸。
4. 除非滿意系統產能，否則請回到步驟 **1** 找出下一個瓶頸。

請注意，此程序是反覆執行的。主要概念是修正一個瓶頸，再尋找下一個瓶頸。

本案例只使用單一 **JVM**，因爲這樣偵測瓶頸會簡單許多。尋找和解決跨 **JVM** 競用可以相當複雜。在調整單一 **JVM** 之後，可進行多節點的產能規劃分析，而本文稍後將有說明。

注意上升率

執行效能測試時，有一個常見的問題，就是 **Vuser** 應該以哪個速率登入系統。

請不要在短時間內登入好幾百個使用者，讓系統垮掉。這個方法並不實際，而且不會提供可重複的結果。

您應模擬實際情況，預測或測量預期入口網站可承受的實際最高登入率。此速率通常會在使用者最常登入入口網站的時間出現，例如，他們早上到達辦公室後。建議您在設定好的時段內（例如，**5** 分鐘），登入固定且少量的使用者（例如，每分鐘二個 **Vuser**），在某一時段內不加入任何使用者，讓系統穩定下來（例如，**5** 分鐘），然後以相同的模式，重新加入另一批 **Vuser**。

這個技巧可以讓入口網站有時間將各種快取依序填入，並且可以更精準地偵測飽和點。

預載入口網站

入口網站重新啓動之後，在執行主要測試之前，應先執行簡短的 **Script** 來預載特定快取（例如，在真正測試開始之前，先預載 **WebSphere Portal** 存取控制項及匿名頁面快取）。如果沒有這麼做，起始回應時間的差異會過大。

分析技巧

入口網站飽和之後，您可以對測試中的入口網站 **Java** 程序，執行 **Java** 執行緒傾出（使用 **kill -3** 指令），以判斷造成系統瓶頸的原因。執行緒傾出會顯示 **JVM** 中所有執行緒的狀態。一般程序是尋出遭相同情況封鎖的所有執行緒，或是在相同類別的方法中等待的所有執行緒，

通常會搜尋被封鎖或處於等待狀態的執行緒。確定某些類別在統計上顯示爲封鎖的原因之後，您就可以著手排除那個原因，進而解除瓶頸。下一節將討論一些常見的瓶頸問題。深入探究所有問題是 **WebSphere Portal** 瓶頸分析的精髓，而且需要時間和經驗。

如果瓶頸並非 **WebSphere Portal JVM** 本身，偵測和解決技巧會有所不同，而且不在本文的討論範圍。

常見問題

本節列出許多客戶在效能測試期間所遇到的常見問題。

記載

利用直接寫入 **SystemOut.log** 或利用記載類別（如 **log4j**）來記載的方式，會導致執行中的執行緒之間序列化，並且會大幅降低入口網站效能。在正式作業的入口網站系統中，請只記載絕對需要的項目。使用 **log4j** 時，應只記載錯誤，請勿記載警告或參考訊息。如果是爲了稽核而需要進行記載，請考慮使用入口網站服務，或是在不同 **JVM** 中執行的其他服務。

關閉所有記載功能，並移除所有寫入檔案的除錯程式碼之後，再執行效能測試。

Java 類別及變數同步化

在使用方法等級的同步化區塊中，若有一個方法處於監視等待 **(MW)** 狀態，而有一個方法保留鎖定，則可能發生問題。在此情況下，您的 **Java** 程式碼已同步化，並導致系統序列化。

使用同步化的類別變數或同步化的 **HashMap** 也會造成這個問題。

在這兩種情況下（方法或變數同步化），都會因爲執行入口網站的 **WebSphere Application Server** 傳輸執行緒數目隨意增加，讓問題惡化。執行緒數目增加，就會增加以此方式同步化之入口網站程式碼命中率，最終讓所有執行緒序列化。

資料庫競用

如果執行緒傾出顯示很多執行緒在 **Socket.read()** 方法中的「**Java** 資料庫連線功能 **(JDBC)**」類別等待，則資料庫本身可能有回應時間的問題。

執行緒在 **WebSphere Application Server** 中的 **JDBC** 儲存區資源上等待時，您會看到執行緒在 **WebSphere Application Server** 連線儲存區 **(J2C)** 類別中，處於條件等待 **(CW)** 狀態。在此情況下，您可能需要增加這個資料來源的儲存區大小，或是增加資料庫可同時伺服器處理的連線數目。

LDAP 回應性

如果有數個執行緒在「**Java** 命名和目錄介面 **(JNDI)**」類別的 **Socket.read()** 方法中，則可能是在等待 **LDAP** 目錄的結果。

階段作業過大

如果客戶撰寫的 **Portlet** 在階段作業儲存太多資料，即會導致記憶體和效能問題。

出現的異常狀況

雖然這個問題好像很明顯，但是很多客戶還是會不斷於日誌出現異常狀況的系統中，試圖進行效能分析和瓶頸削減。**JVM** 在處理未經檢查的異常狀況時，會減緩 **JVM** 的速度，並促使序列 **I/O**（列印）至 **SystemOut.log** 列印串流，進而將 **WebSphere Application Server** 傳輸執行緒序列化。

更常見的問題是嘗試找出本來已有錯誤之系統，並加以調整。在這種環境下產生的所有結果，都必須標示爲不可重覆，並隨著錯誤解除而變更（可能有很大的改變）。

最後，您應該堅持，**WebSphere Portal** 系統進入的高負載正式作業環境中，不得有任何錯誤記錄。

關於 DRS 抄寫模式的動態快取

WebSphere Portal 需要啓用「**WebSphere Application Server** 動態快取服務」。動態快取（**dynamic cache**，或通常稱爲 「**dynacache**」）是一種資料結構，通常用來從 **WebSphere Portal** 中的後端服務快取資料（例如，資料庫結果）。動態快取可確保整個叢集的 **WebSphere Portal** 成員快取同步化。爲了在叢集中正確運作，**WebSphere Portal** 需要啓用快取抄寫功能，但是抄寫的預設模式 **PUSH** 會在 **WebSphere Portal** 環境中造成效能問題。此設定會依據各入口網站叢集成員 **(JVM)**，包含在部署管理程式中。

就絕大多數的 **WebSphere Portal** 配置而言，均強烈建議使用「未共用」。您需要進行三個動作，確定已爲 **WebSphere Portal** 完全啓用該節點。首先，在每個叢集成員的 **WebSphere Application Server** 主控台，將抄寫模式設成「未共用」。接著是安裝 **PK64925**。最後是安裝 **PK62457**。依據所安裝的 **WebSphere Portal** 服務層次，這二個 **APAR** 可能已經安裝。

WebSphere Content Manager 的動態快取也應設成「未共用」。若要完成作業，請在「部署管理程式」主控台中，瀏覽至「資源」-「快取實例」-「物件快取實例」，並變更各個快取實例。撰寫本文時，**WebSphere Content Manager** 有 **11** 個實例。

動態快取逐出考量

從 **WebSphere Portal 5.1.0.2** 版開始，**WebSphere Portal** 動態快取的大小已預設增加至適合大部分客戶之 **WebSphere Portal** 應用程式。可是，仍然會發生預設值不足的情況，造成重大的效能問題。例如，若入口網站有大量衍生頁面在 **WebSphere Portal 5.1.0.x** 版中有共同母項，入口網站存取控制項 **(PAC)** 快取會太小，造成快取猛移 **(thrashing)**（使用的資源增多，執行的工作減少）。同樣地，如果入口網站 **objectID** 快取太小，也會發生猛移 **(thrashing)**。

客戶必須安裝及使用進階動態快取監視器，監視所有快取。如果有一或數個快取似乎有大量的「近來最少使用 **(LRU)**」逐出，可能需要增加該快取的大小。**WebSphere Portal** 快取的大小通常位在 **CacheManagerService.properties** 檔案中。

客戶 Portlet

過去接觸客戶時，經常遇到的部分問題：

1. 使用同步化的類別變數。
2. 資料庫呼叫過多。考慮使用 **DB** 快取層次或動態快取來減少應用程式資料庫或後端服務的負載量。
3. 非同步使用 **HashMap**。如果不同的執行緒命中相同的 **HashMap**，而沒有同步化，則會導致類別進入無限循環的計時情境。

產能規劃

產能規劃的目的，是要在進入正式生產作業之前，估算在預先決定的 **SLA** 測量值範圍內，滿足特定使用者人數所需的 **WebSphere Portal JVM** 總數。

一般測量值包括：

1. 入口網站登入回應時間（一般大約 **4** 秒）
2. 登入之後的頁面轉換回應時間（一般大約 **2** 秒）

程序

執行負載測試的程序，類似瓶頸分析所執行的測試程序，但在停止測試時有第二個準則。第一個準則是已定義的飽和。第二個準則是任何 **SLA** 測量值的失敗。

如果在超過任何 **SLA** 測量值之前，測試達到飽和狀態，或是已判定出沒有可消除瓶頸，則可立刻計算所需的節點數目。

如果在達到飽和狀態之前，超過 **SLA** 測量值，則必須分析失敗原因，以判定下一個動作的做法。如果判定不需要解決回應時間問題，則直接繼續計算節點數目，此部分在本文的下一節將有討論。

推斷結果

一般而言，如果單一 **WebSphere Portal** 節點可支援 **n** 個使用者在給定的 **SLA** 測量值中，則 **2** 個節點可支援 **1.95 * n** 個使用者。所接受的入口網站水平縮放係數爲 **.95**。因此，如果單一 **WebSphere Portal** 節點可支援 **n** 個使用者在給定的 **SLA** 測量值中，則 **m** 個節點可支援：

n (1 + .95 + .952 + .953 + ⋯ + .95 m)

所以水平縮放係數均比線性係數較小。

此縮放係數假設資料庫產能不會造成系統瓶頸。事實上，此縮放係數主要是用來登入使用者的 **WebSphere Portal** 資料庫衰退測量值。

垂直複製（縮放）則有些不同。如果單一 **JVM** 在處理器使用率大約達 **80%**，或以下時讓節點飽和，就需要垂直複製。請注意，在大多情況下，瓶頸分析通常可以解決問題。若沒有 **Java** 資料堆的問題，單一 **JVM** 通常可以進行調整，讓節點在處理器使用率達 **85%** 到 **90%** 時飽和。本文稍後會更完整地討論垂直調整。

以完整叢集進行測試

如果有足夠的負載產能（包括測試 **ID**），進行最後一系列的測試是很明智的做法，這些測試會針對完整的叢集來模擬整個使用者社群，以確保整個系統的可行性。

失效接手測試

如果在失效接手期間，有完整效能的系統需求，也應撰寫此情境的 **Script**，並加以測試。

在執行此情境之前，請檢閱 **HTTP** 伺服器上的 **plugin-cfg.xml** 檔案，確定叢集定義是正確的。考慮新增 **ServerIOTimeOut** 參數至叢集成員。此參數會增加 **ConnectIOTimeout** 參數。**ConnectIOTimeout** 是在遠端伺服器無法依要求開啓 **Socket** 連線時，將叢集成員標示爲關閉之前的時間。該參數通常會在 **plugin-cfg.xml** 檔案中，預設值爲 **0**，表示其仰賴作業系統傳回逾時狀態給外掛程式，而不是由外掛程式明確地計算連線本身的時間。

ServerIOTimeout 參數預設不包含在 **plugin-cfg.xml** 中，此參數會依實際的 **HTTP** 要求來設定逾時。如果入口網站沒有在指定的時間內回應，伺服器就會標示爲關閉。此步驟很有用，因爲 **WebSphere Portal** 叢集成員依要求開啓 **Socket** 時，有幾種類別的失敗狀況，但是 **JVM** 已當機，而且不會回應 **HTTP** 要求。若沒有 **ServerIOTimeout**，

外掛程式就不會將叢集成員標示爲關閉，但也無法處理要求。這種情況會導致要求遞送到當掉的伺服器。

在此測試期間，請先從可完全操作的叢集開始。在模擬作業中，啓用 **Vuser** 至 **SLA** 指定的最大數。然後，停止一或數個叢集成員。您可以按正常程序執行此步驟，從部署管理程式停止叢集成員，或是將叢集節點的乙太網路纜線拔除，以模擬網路失效的狀況。還有其他失效模式也值得您一探究竟（例如，資料庫失效、**Web** 服務失效等等）。模擬叢集成員運行中斷之後，請確定仍然運行的叢集成員會依據您的系統需求，處理其餘的負載量。然後，重新啓動離線的叢集成員，以確定負載量隨著時間回復到平衡狀態。

持續的產能規劃

如果系統已進入正式作業，並符合其目前的 **SLA** 目標，您也會想要規劃系統使用者數目未來的成長。假設 **WebSphere Portal** 上的應用程式沒有重大變更，您可以從執行中的正式作業系統，衍生必要的測量值和計算值，但是，您需要適當的工具來取得測量值。

簡單而言，如果 **n** 個 **JVM** 可支援 **x** 個使用者，則每個 **JVM** 可支援 **(x/n)^(1/.95)** 個使用者。您可以利用先前說明的公式，輕鬆規劃未來的成長。

垂直叢集的考量

其中一項常用技巧以增進效能，就是在相同的實體系統上垂直複製 **WebSphere Portal JVM**。工程師一開始會假設，如果一個 **JVM** 表現良好，兩個 **JVM** 一定會更好。

垂直複製的終極目標，就是增加單一節點上叢集成員（複製項）總和的每秒總傳輸量淨值。此目標通常只在負載之下執行時，且調整良好的叢集成員沒有消耗該節點中大部分的可用 **CPU**，才可以達成。事實上，在調整良好的 **WebSphere Portal** 中，垂直複製有一定的成本。獲益大於成本時，就需要垂直複製。

WebSphere Application Server 叢集有二種。第一種是水平類型，在這種排列中，具有相同功能的應用程式伺服器複製項會在另一個節點上建立。此複製作業是以一種叫做部署管理程式的 **WebSphere** 元件來完成。所建立的那一組相同節點稱爲叢集。結果就是前端 **HTTP** 伺服器可以將要求從用戶端轉送至任一叢集成員（複製項），且結果相同。

同樣地，您也可以垂直建立叢集成員，即在同一節點上建立多個 **JVM**。每個叢集成員都可以提供相同的內容，如同水平叢集成員的例子一樣。

在 **WebSphere Portal** 的例子中，叢集成員會共用一個（只有一個）**WebSphere Portal** 資料庫。在 **WebSphere Portal** 第 **6** 版中，此說法會有些微改變，但是在 **5.x** 版中確是如此。因此，隨著叢集成員數目增加，**WebSphere Portal** 資料庫更有可能因爲可用產能減少而變成瓶頸。

垂直叢集的成本

同一實體節點上有額外的作用中叢集成員，就會產生相關成本。首先，會有程序環境定義交換。作業系統現在必須管理額外的程序 **(JVM)**。

其次，處理器資源的競用會更加激烈。一般而言，如果作用中的叢集成員數目超過節點中的處理器數目減一，均不適合選擇垂直叢集。例如，在配備三個處理器的節點上，不應該有三個叢集成員。在某些情況下，可以接受在配備三個處理器的節點上，有二個叢集成員。

垂直叢集作業的意義

本節說明垂直叢集成員可發揮價值的一些狀況。

可靠性

除了效能考量之外，僅就可靠性來說，擁有額外的叢集成員是有意義的。如果 **WebSphere Portal** 安裝在單一節點，則在軟體損毀一個 **JVM**（但沒有損毀作業系統）的情況下，您可以新增垂直叢集成員，減輕損毀的影響。這是假設大部分的軟體故障都只會影響單一個 **JVM**，並不會影響同一節點上的其他 **JVM**。因此，故障的 **JVM** 重新啓動時，該叢集會繼續服務要求。

記憶體使用率

在 **32** 位元作業系統中，程序位址空間的記憶體限制爲 **4 GB**。大部分的作業系統會將此空間分割成 **2 GB** 的使用者空間和 **2 GB** 的核心空間。可是，也有例外的情況將使用者空間增加到 **3 GB**，而核心空間降至 **1 GB**（例如，**Solaris**、**AIX®** 及 **Microsoft® Windows® 2003 Enterprise**）。

如果 **JVM** 可用的位址空間是 **2 GB**，則 **JVM** 可配置大約 **1.5 GB** 的資料堆空間。

在工作量大的時間，執行 **WebSphere Portal** 基本記憶體工作集，以及所有 **Portlet** 所需的總記憶體組合，可能會使用並接近用盡 **1.5 GB** 的資料堆。發生此情況時，如果仍有大量的處理器資源可用（**20%** 到 **30%** 或以上），則垂直複製會有效率地建立 **3 GB** 的 **JVM** 資料堆，並且在兩個 **1.5 GB** 的資料堆 **JVM** 之間平均分配工作量，以增加該機器的總傳輸量。

Java 同步化方法及類別變數

如果 **WebSphere Portal** 應用程式（及入口網站本身）使用足夠的同步化方法或類別變數，則在負載之下，應用程式伺服器中最後會經常出現大量的封鎖執行緒。若要識別此情況，您可以在負載之下擷取執行緒傾出，並注意有許多 **Web** 儲存器執行緒處於 **MW** 狀態，等待這些已同步化的構件。

在此情況下，依據各叢集成員來減少 **Web** 儲存器執行緒的數量上限，可減少這些分區。如果在該變更之後，處理器的耗用情形與上述不同，則垂直複製可增加整個節點的總傳輸量。

結論

將 **WebSphere Portal** 送進正式作業之前，先進行適當的測試，可以消除許多常見的效能問題，進而提供更爲順暢的使用者經驗。本文提供了建立測試規劃的架構，以及必要的執行程序，以確保將系統部署至正式作業時，可以取得令人滿意且可以預測的效能。

資源

- 參加論壇。
- 進一步瞭解 **IBM WebSphere Extended Cache Monitor**。
- 參考 **IBM WebSphere Portal Information Center** 中有關效能調整的要訣。
- 參考 **IBM WebSphere Portal Performance and Tuning Guide**。

關於作者

Alex Lang 於 **1982** 年加入 **IBM**，一直從事網路、數位信號處理、**Java** 支援及 **IBM WebSphere** 方面的各種技術和管理工作，目前是 **WebSphere Portal SEAL** 團隊的技術團隊領導人，主要專中於爲客戶解決 **WebSphere Portal** 架構、部署和運作的重要狀況。