開放原始碼 MySQL 資料庫在多種同步架構下的實作與效能分析

白明宜
靜宜大學資管所
碩士
e-mail：
tonypai@hotmail.com

林茂松
靜宜大學資工所
碩士二年級
e-mail：
g9672033@pu.edu.tw

鄭介勛
靜宜大學資工所
碩士一年級
e-mail：
g9772014@pu.edu.tw

王孝熙
靜宜大學資管所
教授
e-mail：
hhwang@pu.edu.tw

摘要

開放原始碼 MySQL 資料庫，是屬於 LAMP(Linux，Apache，MySQL，PHP)堆疊中之一，在網路開放原始碼應用搭配組合上已經是趨於領先地位，且有數年之久，依照 netcraft.com 目前的統計，到 2008 年四月時，全球使用 Apache 網頁伺服器營運的網站有 82,454,415 個[01]，而使用 Apache 網頁伺服器最多的組合搭配，就是 LAMP 的組合，MySQL 在開放原始碼資料庫中，始終保持最領先的地位。

我們希望目前及未來使用 MySQL 資料庫系統管理者或系統規劃者能夠經由此研究報告資料，了解各種資料庫同步架構(Database Replication topology)下之差異，及不同條件下之解決配套方案，進而有效的協助系統建構時的決策制定。讓組織企業在各單位的承載需求，風險忍受程度及可運用之經費等，種種條件下，讓建構的系統能以最少的代價而得到系統最大之效能。

關鍵詞：MySQL 資料庫、同步架構、Vmware、資料庫系統效能調校。

1. 前言

資訊化的時代，資料庫在使用上及各種資訊系統中所扮演的角色越來越重，我們研究是希望透過技術的改善讓企業更方便的擁有資料庫系統。透過實作的方式，達到以最小的代價，讓伺服器發揮最大的承載效能。

此研究之目的是針對開放原始碼 MySQL 資料庫，在不同的同步架構中，對資料承載的能力作出統計，搭配各種不同的研究及試驗，找出各種組合條件對應到適合的系統需求上。

我們將各種方法實作，以了解不同條件及架構下，系統負載的能力比例。進一步也可知道開放原始碼 MySQL 資料庫如何在負載不斷的成長下，以最小的代價，讓伺服器發揮最大的承載效能。

同步架構在不同組合方式及改變下之各種搭配方法優缺點研究。實作所有基本組合架構，並照預定規劃過程紀錄實做測試結果。依各種架構將各別數據分析整理後，繪製實測結果圖表，解釋每種組合架構優缺點、設定過程、取樣方法、在了解各種架構下效能對應之關係後，可以針對需求，利用本論文之結果及建議規劃不同資料庫同步架構，可記錄公司資料庫系統之效能狀況，幫助企業決策或資料庫管理者改進其管理之資料庫系統架構。

2. 研究背景與動機

網際網路在日常生活中已經逐漸成為人們生活的一部份，也因如此它也成為眾多行業生意營運中重要的一環。為了使電腦能發揮功用為人們帶來的方便，不論是公司內部使用之企業資訊入口網站(Enterprise-Information-Portal)或者是企業對企業(B2B)，企業對客戶(B2C)平台等，大多都已經網路化，使資料之存取更簡單，更無地域限制。而這些平台不外乎就是處理公司內外部的資料，將資料有系統的儲存，再經由程式將資料轉成有用的資訊，供給公司內部不同的單位使用。而我們上述所討論到的資料儲存與取用大多是對資料庫作存取。

一般正常的情況，隨著資料量日積用累，資料皆呈正數的成長。或者，一般的網站系統在正常的經營模式下，使用者數量應該也是呈正數的成長。很多企業組織在面臨到資料庫繁忙，在沒有經驗或協助的情況下，大多購買更貴或更大的伺服器來解決負載的問題。許多人因為數年來一直面對相同的問題，進而在不斷的修正及嘗試錯誤中，不斷的改進、累積經驗，目前改用同步架構的網站也不斷成長。

開放原始碼 MySQL 資料庫在負載不斷的成長下，如何以最小的代價，讓伺服器發揮最大的承載效能？相信這是所有企業主的需求及系統管理者想挑戰的。也因如此我們要將各種方法實作，來了解不同條件及架構下，系統負載的能力比例為何？在伺服器發揮最大的效能背後，換來的是系統複雜度增高，相對言之，系統管理者對資料庫的管理難度增加，資料出錯之修復時間變長，系統之潛在問題節點(node of failure)增多。

2.1 研究目的

我們研究之主要目的為說明開放原始碼對許多企業或組織發揮開放原始碼的高效率、穩定及營運費用低廉之各種優點。完整收集開放原始碼 MySQL 資料庫操作相關技術背景，問題解決方案等。如資料庫儲存引擎之優缺點，資料庫版本演進之影響，技術演進之相關配合條件。相關搭配技術解決方案，各種同步架構對系統效能之改進，改進後系統複雜度提高之優點、缺點及缺點改進方法。提出對各種架構之效能測試方法，並撰寫程式實作出實際運作數據，讓不同需求者可參考此方法測試相同架構下不同數量之伺服器組合及不同硬體配置之效能改善程度。對於目前及未來的企業資訊系統決策制訂及資料庫管理人員在使用開放原始碼 MySQL 資料庫進行系統架構評估時之決策支援。

2.2 研究方法

我們的研究方法著重於實作，將目前已知的解決方案以及本篇論文中所提出的架構，以實作之方式整理出具體的測試數據，我們會使用到的軟體皆是開放原始碼，其中包含有：Fedora 作業系統平台、Apache 網頁伺服器、MySQL 資料庫管理系統、PHP 網頁直譯式程式語言及些許 MySQL 官方正式及非正式資料庫測試工具。此研究中，實體機器架構方面我們將以有限的機器資源：四台伺服器、 二台桌上型電腦、一台筆記型電腦，透過系統虛擬化的方式將機器以等比例之方式做切割，以滿足實驗時需要用之機器數量。邏輯架構上，我們會有四台資料庫伺服器、十三台網頁伺服器、五台 Windows 平台網頁資料新增發起端作用在六種不同的架構組合。在兼顧等量等比例效能測試架構下，完成我們的需求數據資料。實作所使用的平台會在後面的效能測試架構章節做詳細的說明。

2.3 研究範圍與限制

在眾多商業版本或開放原使碼版本資料庫系統中，我們選擇非商業版本及在真實網站營運上使用最多最廣泛的，開放原使碼 MySQL 資料庫做為研究的主要標的。因為各種資料庫都有各自的優缺點，研究發展的方向也不相同，不同的資料庫，在特定需求或平台上如 LAMP 架構堆疊上，去比較運作及效能並不符合客觀的要求，所以我們盡可能使用相同之硬體，以相同之資料庫及版本，相同之測試環境，相同之資料 Insert 方式及數量，唯一不同的調整變數是同步架構之更改。

本論文探討的目標為以開放原使碼 MySQL 資料庫為基礎，對資料庫同步(Database Replication)下作不同組合的架構去評估對效能的影響。

因為 Fedora 8 所附上的資料庫版本為 MySQL 5.0 ，在實作上我們只做新增(Insert)指令，所以我們實測只使用到語法記錄檔同步方式(Statement based Replication)，正式發行版 MySQL 5.0 只有支援 Statement based Replication， 經過不斷的進步與測試改版，在目前 MySQL 5.1 測試版本以後都支援語法記錄檔同步方式(Statement based Replication)、實體資料記錄檔同步方式(Row based Replication)[05]與混合方式(Mixed)。此三種選項差異為：

1. 語法記錄檔同步方式（Statement Level Replication）：不管資料是否為非可確定性語法(non-deterministic)，同步的過程中確實的複製所有 SQL 指令，後面同步架構需注意事項中有詳細說明範例。

2. 實體資料記錄檔同步方式(Row based replication)：作法是複製的內容為執行完 SQL 語法後的結果，運作時所有的同步資料都使用實體資料記錄檔記錄(Rows based logging)，只有碰到 data definition language (DDL)及 data control language(DCL) 還是使用語法記錄檔(Statements logging)。

3. 混合方式(Mixed)：MySQL 5.1 版本後預設選項使用此種模式，一般資料仍使用以語法為基礎(statements based)方式紀錄，但多了自動分辨資料是否為 非可確定性語法(non-deterministic)，如果是就會以 rows based logging and replication(RBR)執行。

本實驗主要是要觀察資料庫同步之間的效能及延遲，因為需要知道延遲時間，所以就必

須有時間的記錄。在伺服器的時間上我們使用 NPTD 讓機器與機器的時間差維持在 1 秒以下。另外在網頁資料取樣上，考慮到時間記號的新增本身也是一種負擔，我們記錄的時間啟動是由網頁每秒送出時間記錄的需求給資料庫去進行的，在時間記錄上並非同步啟動執行而是以程式控制的方式分散到不同的網頁以相同的時間差去執行。我們的取樣時間以 1 秒為單位，故此實驗數據的最大可能時間誤差值在兩秒以下。所以當資料差異秒數小於 2 秒時，我們當成 0 計算。

3. 各種同步架構及組合

在完成了各種同步架構的實測後，在本節中我們將一一的說明我們實測的六種 MySQL 資料庫同步基本架構。本節總共有七個小節，前面六個小節分別對應到六種基本同步架構：(3.1)單一 MySQL 資料庫伺服器、(3.2)單一主資料庫加單一副資料庫架構、(3.3) 樹狀同步，單一主資料庫加多台副資料庫架構、(3.4) 樹狀同步，主資料庫加多台副資料庫分佈於多階層架構、(3.5) 鏈狀同步，雙主資料庫架構、(3.6) 鏈狀同步，兩台以上多台主資料庫同步架構，最後(3.7)為多種資料庫同混合架構下的組合討論及該注意事項。

開放原始碼 MySQL 資料庫在同步架構上有多種組合方式，多年前就有人對同部架構對有高度的興趣及提出看法[23]，可是不論如何改變，MySQL 資料庫方面到目前為止基本的架構仍然沒有重大的突破，雖然資料庫系統本身一直都有在進步，但還是趕不上使用者對系統能力提昇的殷切需求，最後還是借由電腦硬體上突飛猛進的進步，利用同步的解決方案將大量的電腦組合起來進而創造出倍數效能成長的系統架構。本章的內容重點在使用者了解這幾種基本組合的優缺點後，可以再參考最後一節的其他混合架構以期讀者能依自身需求輕鬆設計及駕馭 MySQL 強大的同步功能。

3.1. 架構 A，單一 MySQL 資料庫伺服器

安裝 Apache 搭配 MySQL 於單一伺服器上或單純安裝 MySQL 於單一伺服器上接受網路連線負載。其同時為網頁伺服器與資料庫伺服器兩種角色，此種架構在網頁伺服器程式與資料庫伺服器程式直接在本機中完成溝通，算是效能高成本低的網站營運使用方式。

它最大的缺點就是當負載昇高時，不容易作系統的調校。雖然此種模式也可以多部串聯成群組。但問題是一台機器是網頁伺服器又是資料庫伺服器，在任何一方出錯或產生高負載延遲時會影響到整體運作，相對的如果我們把網頁伺服器與資料庫伺服器分開至兩台機器上，讓其各司其職。如果是這樣，這樣複雜度就會降低，以利分別針對網頁伺服器或資料庫伺服器加以調校而不去影響到對方。多付出的成本為網頁與資料庫在溝通上由原本本機中完成的溝通要改由成本較高速度較慢的 TCP/IP 網路溝通。我們之所以特別比較單一主機與多主機主要是要說明其對複雜度降低及有助於爾後的擴張性。我們後續的架構將都是基本網頁伺服器與資料庫伺服器分離的架構。我們在實測部分此架構在網頁完成時間與資料庫完成同步時間的結果如圖所示：

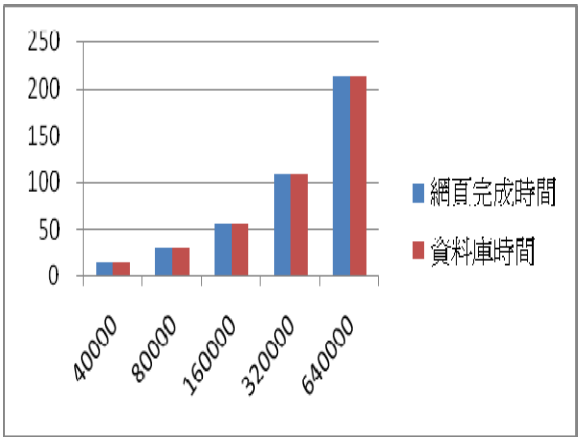

圖 1 架構 A 實測單一主資料庫網頁完成時間與資料庫完成同步時間比
(縱向單位：秒；橫向單位：資料新增筆數)

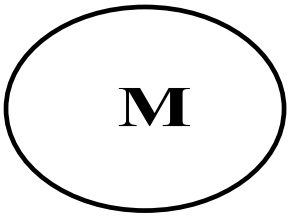

圖 1-1 單一主資料庫

3.2. 架構 B，單一主資料庫及單一副資料庫架構

這是一對將資料庫分成兩部份的基礎架構，分開後其中一台是主伺服器其可以接受所有的 SQL 語法，主伺服器因為後端有串接副

伺服器，所以在設定上需要設定為啟動同步記錄。後面串接的是副伺服器，他在透過設定帳號密碼後可以與主伺服器認證後進行資料庫同步的連線，其實他就是不停的讀取主伺服器上的同步記錄資料， 讓這些資料也產生在副伺服器中， 因為副伺服器經由同步記錄而可以在機器上得到與主伺服器相同的資料， 故副伺服器可以承受非修改性的語法如單純資料查詢或者是進階資料分析語法。在此架構下需注意幾個重點：

一、 主伺服器與副伺服器在理論上，一定會有資料同步延遲的時間差，且時間差的長短會隨著機器負載、資料量、機器數量及架構等受影響。所以在查尋即時性資料時必須要考量到資料有效性問題。

二、 在此架構下網頁的程式結構必須修改，將前端程式與資料庫互動的語法群組化，也就是分成修改資料型語法與非修改型語法。

三、 以下的每種組合都有項目(二)相同的問題。

我們在實測部分此架構在網頁完成時間與資料庫完成時間的結果如圖所示：

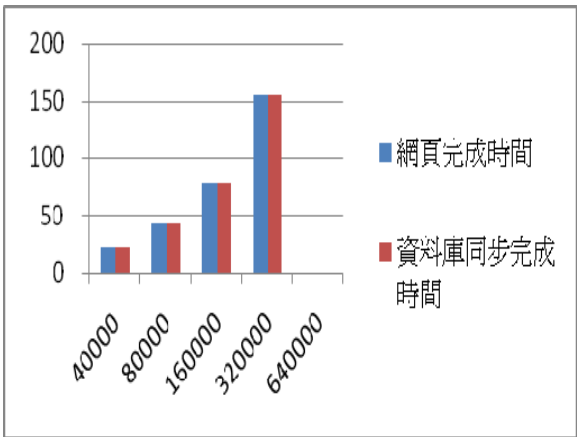

圖 2 架構 B 實測單一主資料庫伺服器及一台副資料庫伺服器網頁完成時間與資料庫完成同步時間比
(縱向單位：秒；橫向單位：資料新增筆數)

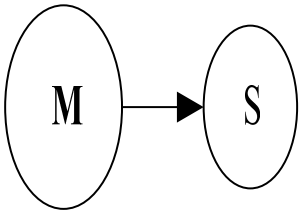

圖 2-2 單一主資料庫及單一副資料庫

3.3. 架構 C，樹狀同步、單一主資料庫及多台同階層副資料庫架構

這是一個應用很廣的架構，屬於資料庫橫向的擴充，橫向擴充的好處是全部副資料庫都屬於同一層級的機器。資料來源一樣，延遲影響變數都很接近。但唯一缺點是主伺服器只有一台，主伺服器除了本身必須承受資料新增修改的負載、記錄同步資料的負載、同時也需要對副伺服器傳輸同步資料。當副資料庫伺服器的數量過多時，主伺服器容易出現過載問題而成為此架構的一處弱點。在大部份的網站大多是執行資料查尋。基本上一台主伺服器與副伺服器對應在讀取同步資料的負載並不會很重，但在伺服器硬體相同的條件下幾乎是每增加對等數量的副伺服器，該網站可接受查詢的能力就增加一倍。

在使用合適數量的副資料庫伺服器組合架構是一種支出費用低廉與但效能增加比例高的一種組合。大多數的互動型網站都可利用這種架構來提升資料庫查詢承載能力。相關文獻中我們也看到此種架構也有人用在多人連線的遊戲平台上為的是更快速的效能及更高的可靠度，差異的是他不是使用 MySQL 資料庫、且程式上多了判斷資料是否已同步完成[28]。我們在實測部分此架構在網頁完成時間與資料庫完成時間的結果如圖所示：

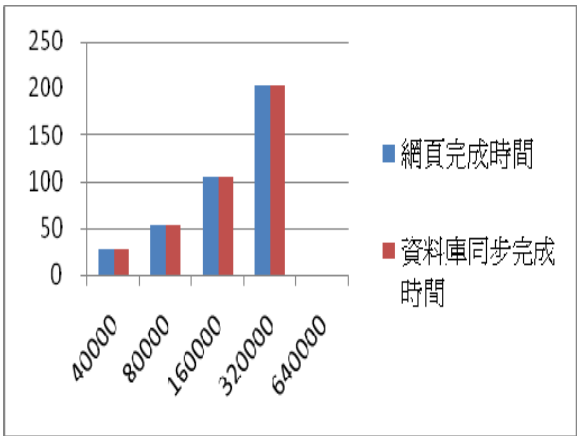

圖 3 架構 C 實測單主資料庫及三台同階層副資料庫網頁完成時間與資料庫完成同步時間比
(縱向單位：秒；橫向單位：資料新增筆數)

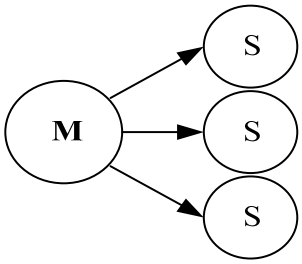

圖 3-1 單一主資料庫及三台同階層副資料庫

3.4. 架構 D，樹狀同步、主資料庫加多台副資料庫分佈於多階層架構

此架構與上個架構類似，但多加了數個延伸的層次。上個組合是屬於橫向的擴充，這個組合除了可以適量的橫向擴充，當橫向擴充出現瓶頸時再以縱向的擴充。整體運作架構不論在使用機器數量及效能上都可以有效增加。但無論怎麼增加都只能增加對非異動性查詢語法的能量。縱向擴充有幾個缺點，也就是每個縱向擴充中間的結點負載及資料儲存耗用空間都較大。因為除了最後一個點之外每個節點同時為主、副資料庫伺服器，再者延遲的時間隨著縱向延伸的階層增加而大幅累進。

在不注重資料庫整體即時性的需求下這種組合非常有彈性，除了網站系統可以增加查詢能量，也有許多 ERP 系統將不異動的封存資料庫利用離峯時間去同步到世界各地的分公司。每個大小不同的分公司，就如同資料庫架構階層般的慢慢同步起來。這樣的好處是主資料庫不會在瞬間產生大量的同步連線需求，同步的方式就像漣漪般的慢慢散發出去。我們在實測部分此架構在網頁完成時間與資料庫完成時間的結果如圖所示：

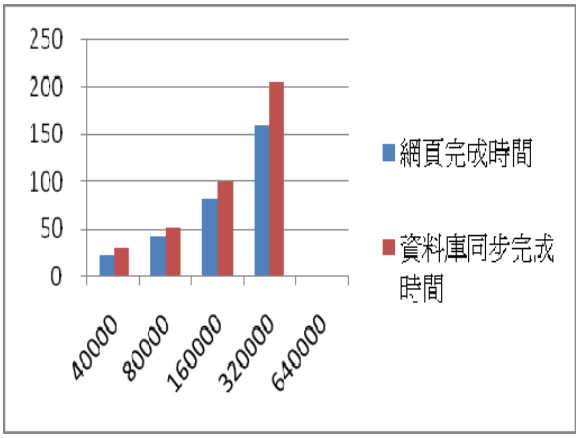

圖 4 架構 D 實測單主資料庫及三台三階層副資料庫網頁完成時間與資料庫完成同步時間比
(縱向單位：秒；橫向單位：資料新增筆數)

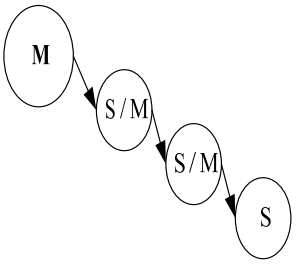

圖 4-1 實測單主資料庫及三台三階層副資料庫

3.5. 架構 E，鏈狀同步、雙主資料庫架構

此架構大多是用來讓主資料庫可以多一層保護，就如同兩台資料庫互為備援，那台出現失敗，另一台既可立刻取代。因為同步機器的數量只有兩台，在處理異動性語法的能量增幅有限。不論在同步延遲處理、資料修復、新增資料時間差等此架構都不會太複雜。雖然機器只有兩台，但是主資料庫的及時備援複製根本的解決了許多系統上的備援問題。我們在實測部分此架構在網頁完成時間與資料庫完成時間的結果如圖所示：

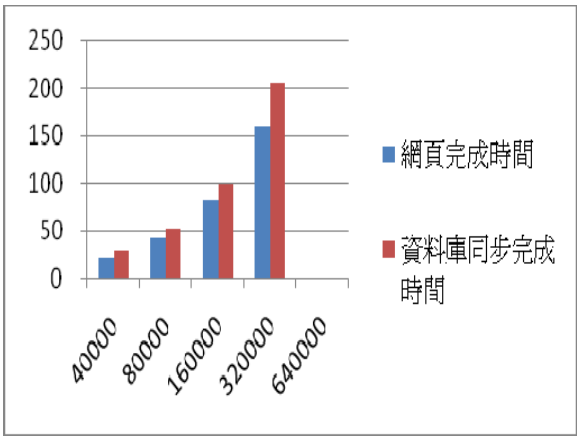

圖 5 架構 E 雙主資料庫架構網頁完成時間與資料庫完成同步時間比
(縱向單位：秒；橫向單位：資料新增筆數)

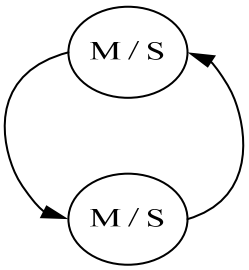

圖 5-1 雙主資料庫架構

3.6. 架構 F，鏈狀同步：兩台以上多台主資料庫同步架構架構

一般稱為 Daisy Chain [12]：這是一個 MySQL 資料庫不斷穩定發展下慢慢讓大家能接受進而廣為使用的架構，這個架構的複雜度最高、 維護最困難、相關需求的配套措失最多，但他確最能有效的強化處理特定的異動性資料執行的效能。此架構缺點有：

一、 維護的複雜度提高

主伺服器群同步的維護複雜度是最高的，單一節點出現問題會影響到整體同步，如果問題同時發在在好幾台資料庫中。此時資料被新增的時間先後判斷問題，資料的回復及捨棄會變的很難處理。另外因為困難度提高，相對也增加營運的風險。因為並不是所有的企業或系統都能夠忍受長時間的災後復原等待或資料損失。

二、 可能造成整體系統失敗的硬體節點及軟體節點增多

此架構每個節點出問題都會造成整體運作中斷，而節點又可分成硬體節點與軟體節點。如果硬體出問題，整體同步就失效。如果硬體沒問題但軟體出錯，整體同步還是一樣中斷。

三、 每個結點既是主資料庫也是副資料庫所以系統運作時的負載均衡不容易達到

因為這個架構每台資料庫伺服器同時扮演主資料庫跟副資料庫角色，所以影響資料庫的六個負載會在機器上不定期的運行，這六個負載為：

1. 資料庫本身運作負載。
2. Binary log 接收。
3. Relay log 寫入。
4. Relay log 執行。
5. 同步資料 Binary log 寫入。
6. Binary log 傳送。

資料的寫入及讀出都會造成系統很大的負載，所以當六個行程同時在其中一台機器上運作時，但其他機器因為尚未輪到進行同步，所以並沒有如此忙碌，這樣的情況有可能造成整體資料庫效能低落。所以前端如何將負載均衡的分散到各機器上會是這個架構中一個很重要的課題。

四、 在資料庫群尚未完成整體同步前，資料庫內存資料是不具時效性的

因此需要做資料是否有效的偵測(是否已完成整體同步動作)。對 Update 跟 Delete 語法使用上需謹慎，此架構因資料隨機分佈在任一機器群中，等待同步。當機器數量越多，同步的時間就會越長，而等待整體同步完成的時間也就越長。在整體資料庫同步尚未完成之前如果執行了 Delete 與 Update，其指令所影響到的筆數，將是使用者很難掌握及預估的。相對的好處是假性完成資料異動的時間大符縮短了。也可以說系統的處理資料效能加快了。

這種效能增進的優點可以在現實環境中解決掉許多原本相當困難解決或需要昂貴機器才能解決的問題。後面章節我們會以圖例再詳細說明其步驟與過程。

五、 此架構適合非常態大量 Insert 資料異動，但非持續性的大量

此系統架構因整體資料庫同步伺服器數量的多寡可以來決定可承受瞬間大量 Insert 異動的能量。它之所以能夠瞬間消耗掉大量 Insert 異動的能量是因為使用了同步的方式去達到資料分散處理的好處，資料雖然分散了但是要完整有效的可以使用這些資料，還是要讓其完成整體同步後才可以。所以如果是常態性的湧入大量資料，則系統完成同步的時間越拉越長，資料越寫效能越差，到最後終將會把系統資源耗盡而整個資料庫群停止運作。我們在實測部分此架構在網頁完成時間與資料庫完成時間的結果如圖所示：

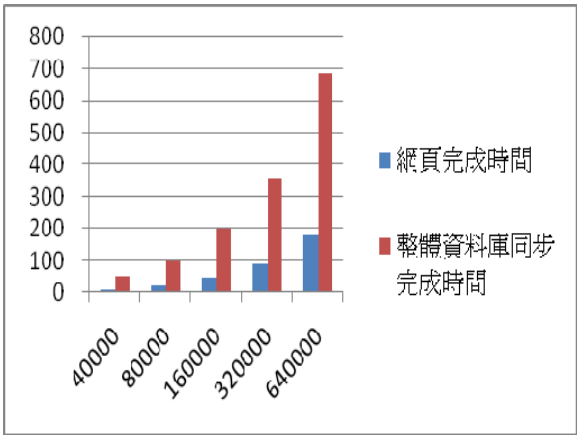

圖 6 架構 F 四台主資料庫同步架構網頁完成時間與資料庫完成時間比
(縱向單位：秒；橫向單位：資料新增筆數)

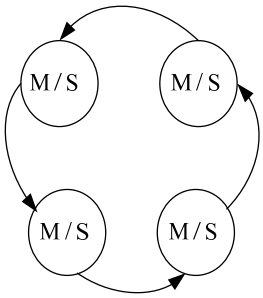

圖 6-1 四台主資料庫同步架構

我們依照實作結果及架構使用機器數量去評斷以上所有架構的優缺點比較表如下:

表格 1 架構 A-F 及 MySQL Cluster 優缺點比較表(系統方面)

	管理難易	系統容錯	系統後續擴張	系統彈性	機器數量(電力消耗)	同步網路流量需求
架構 A	簡單	低	低	低	少	少
架構 B	簡單	低	高	高	少	少
架構 C	中等	中等	高	高	多	多
架構 D	中等	中等	高	高	多	多
架構 E	中等	高	中等	中等	少	少
架構 F	困難	高	中等	中等	多	多
Cluster	困難	高	高	高	極多	多

表格 2 架構 A-F 及 MySQL Cluster 優缺點比較表(資料庫效能方面)

	Select	Insert	Update	Delete
架構 A	普通	普通	普通	普通
架構 B	優	普通	普通	普通
架構 C	強	普通	普通	普通
架構 D	超強	普通	普通	普通
架構 E	普通	優	普通	普通
架構 F	強	超強	普通	普通
Cluster	普通	強	強	強

3.7. 其他混合架構之組合

本節要討論的是基於上述六種基本架構可以再稍微巧妙整合成合成架構，或以其他方式拆解以因應不同應用需求及注意事項：

一、單一機器多子資料庫程序運作(Multi instance)：

使用者可以在一台機器上安裝數套 MySQL 程序並同時運行，用此方式來統一管理及集中副資料庫。散發到各地的主資料庫一般透過網路同步的方式進行異地資料庫備援，集中式的備份架構大致如圖所示，圖中把基隆、桃園、新竹、台中、台南、高雄的資料庫備份到台北總公司機房單一一台伺服器內，在機器效能允許下，省電又省機器。

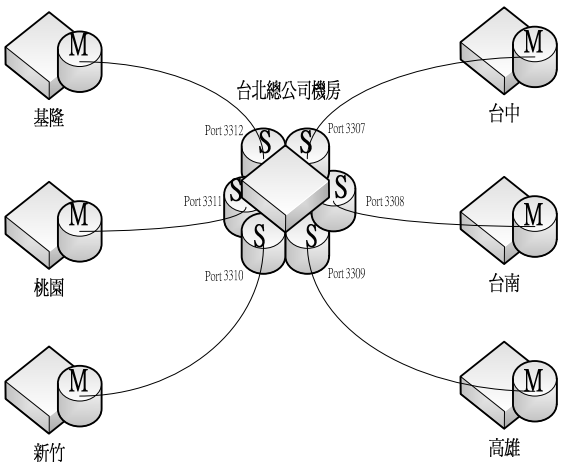

圖 7：MySQL 資料庫子行程於單一機器中運行示意圖

二、單台主資料庫同步多台副資料庫(Master with slaves)：

單一主資料庫伺服器在同等機器數量下使用單一階層的副資料庫群與使用兩階層副資料庫群的比較：使用單一主伺服器在許多資訊系統中是無法改變的現實。當資訊系統有資料

操作為即時性之需求時，單一資料庫是我們唯一的選擇。(MySQL Cluster 我們也將之視為單一資料庫)。當我們只有一台主資料庫我們在執行資料異動性語法時不需考量同步的整體進度，只需考慮當我們做即時的查詢時，副資料庫的同步狀況。一般來說主資料庫伺服器的工作能量是比副資料庫來的珍貴，所以當我們將查詢類語法轉移到副資料庫群中，可以有效的提昇高成本的主資料庫效能。

圖 8(樹狀多階)中雖然主資料庫減低了負擔但比起圖 9(樹狀單階)卻增加了大量的同步傳遞成本。圖 8(樹狀多階)雖然減低了主資料庫的同步傳遞成本，但第一階的副資料庫機器得完全承受我們之前提到的六個同步負載，也就是多了 relay log 的成本。兩個階層同步完成時間會比單階層來的久。比起主伺服器的效能與副伺服器的同步時間延長，在大多數的資訊系統中，若副資料庫節點很多的話，大多數情況下樹狀兩階層副資料庫群架構會比樹狀單一階層的副資料庫群表現來的好一些。

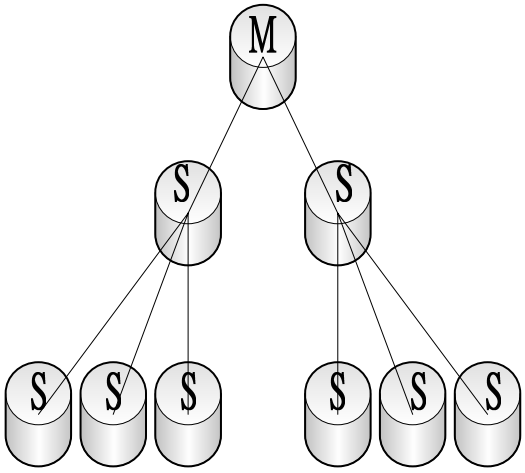

圖 8 資料庫伺服器樹狀多階同步架構

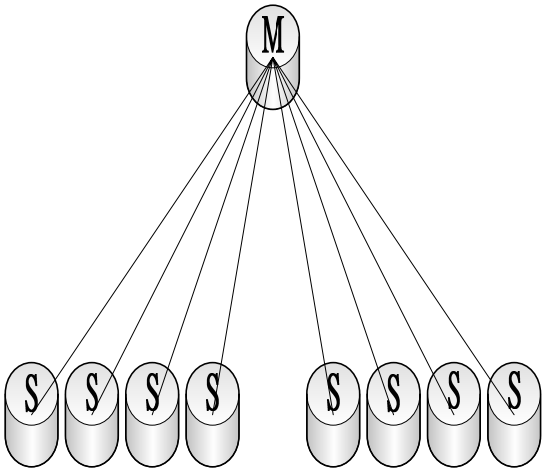

圖 9 資料庫伺服器樹狀單階同步架構

三、 多台主資料庫伺服器搭配多台副資料庫伺服器組合

現在雖然談到了結合更多種架構的組合，但筆者附上的架構圖還是一個簡單的小型組合。需求者其實可以針對自己特定的需求去調整主資料庫及副資料庫的數量及位置。不論如何組合，混合模式的可能影響變數變多了，維護也變的更困難。如果不是有可以巧妙的調整異動性資料庫語法能量與非異動性資料庫語法能量的好處，筆者其實不是很建議使用這種混合式架構。

這種架構可以說是組合了所有架構的優點同時也繼承了所有的缺點。除非該公司有經驗豐富的資料庫管理員，不然一般公司可能不容易享受到其優點反而被其缺點給弄得常常出錯。以相同伺服器硬體花費及相同的處理資料數量比較，不管是免費或收費軟體，此種架構比較能處理瞬間暴量的電子下單系統。

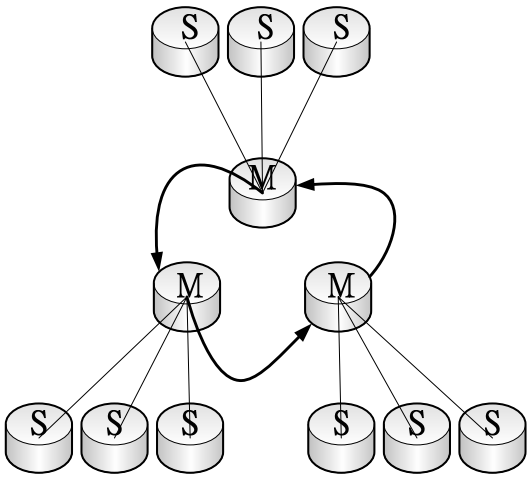

圖 10 樹狀同步與鏈狀同步架構組合

四、 群組化資料庫或拆解分散表格(Table)

由於資料搜尋或取用的時間不同，我們可以透過拆資料庫甚至 Tables 資料同步到不同硬體上，以提高搜尋之速度。如果是剛成立的留言版則用單一資料庫就可以完成所有工作。但如果他發展到一定程度，如線上人數始終保持有萬人以上，則我們可以將負載分成幾個部分來看：資料庫三階樹狀同步(圖 11)，以資料庫階層為單位群組化。在主資料庫部份純粹負責資料的新增，一般討論區，留言版新增資料的負載並不大，所以主資料庫也許使用一台就可

以了。第一層負載不重。同步的負載大多在第二層中，而我們將以第二層副資料庫作用於需要被快速查尋的資料的處理單元。如使用者登入資料、線上人數統計等。第三層副資料庫則用來作用在討論區、留言版負擔最重而且量最大的文章內容搜尋部份，綜合統計報表等。(圖 12)查詢分散是在同步下將相同屬性的資料庫群中以資料型態或資料儲存來源等方式為依據來分散資料儲存處，當要查詢時再合併分開查詢的結果來運作資料庫以加速資料庫效能。

舉個分散儲存的例子，分散的方法可以從依查詢的不同去取決儲存處，如應英文人名可依最後一個字元為 A , B 或 C 取分類，或以均等方式利用 auto_increment 造成不同的 ID 最後編碼 01、02 或 03 去分類。在未來使用上如果有簡化查詢需求，也可以將這資料庫引擎設成 Merge 型態，我們隨時可以在邏輯上再將資料合併起來但仍然不改原儲存單位分開的方式。資料區隔分散法 (Database data partitioning)：如分開資料表格(Table)、資料(Raw data)等，資料存取將複雜化，但複雜之程度跟 Partitioning 之後資料庫的效能改善程度大致成正比。相關資料區隔分散法更深入的話，可分為靜態及動態切割，但動態的複雜程度更高且往往是針對資料型態、屬性、意義去客製化設計。

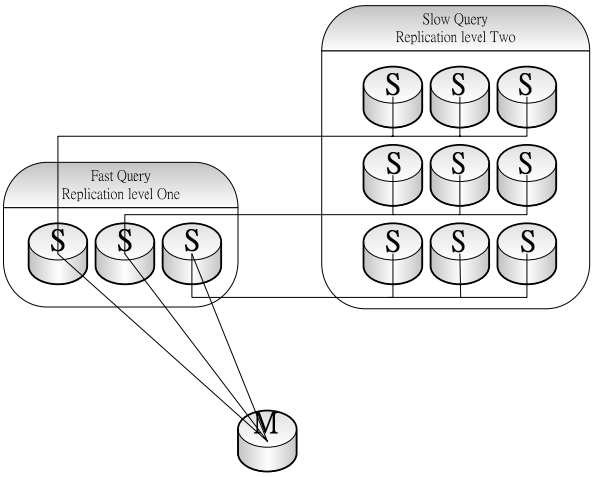

圖 11 群組化資料庫圖示

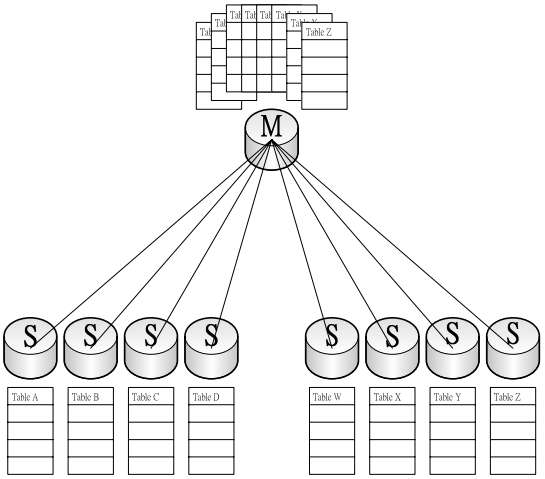

圖 12 拆解分散表格圖示

五、 不同資料庫引擎搭配

每個資料庫引擎都有他的優點或缺點，然而在不同的儲存引擎之間也可以使用同步功能將其串接起來。我們以(圖 10)為範例來看，如果主資料庫引擎使用 InnoDB，以確保資料的完整性。因為它犧牲掉了速度，但因為在第一階層已經確保資料的完整性，故同步送到第二階層時就可使用非支援 transaction supported 的儲存引擎以增加效能，甚至可啟動壓縮機制，減少空間耗損。

六、 鏈狀同步應盡量使用負載平衡機制與使用同等級之節點機器

一旦使用了同步技術，隨之而來的是資料庫的數量增加。當資料庫數量增加在前端程式要連接後端資料庫的方式大至上以程式控管連線與使用負載平衡器來支配連線兩種。以程式控管雖然不用花費額外支出，但管理上、除錯上及實際運作上負載平衡器都比程式控管靈活及簡單。因為一般如果有使用一定數量的資料庫資訊系統都希望盡可能的降低停機維護時間或人為錯誤造成資料損壞。鏈狀同步的缺點就是同步內任何一節點失敗則整體同步既告失敗。情節輕微的話任何一節點負載過重、整體同步都會卡住，嚴重的話就是整體系統停擺、資料毀損、需大量時間救援。所以如何讓每台機器在適當的負載下運作是使用鏈狀同步的重點。使用負載平衡機制技術及使用同等級之機器都是有效的方法。

七、 鏈狀資料庫同步加上多資料庫行程運作 (Multi instance)

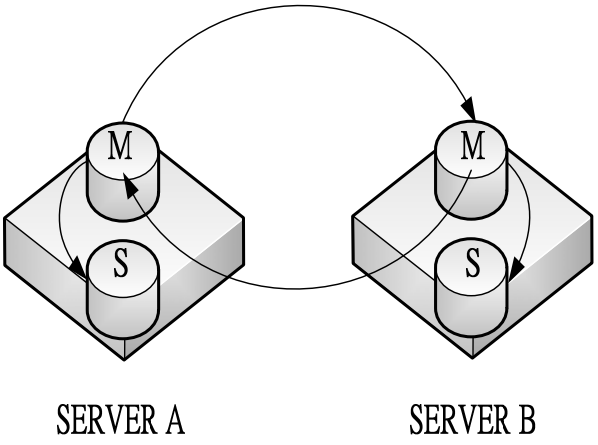

圖 13 鏈狀資料庫同步加上多資料庫行程運作示意圖

這個架構主要是讓資料庫能夠使用稍為多一點點設定及多一點點機器用量的方式，去達到資料庫服務完全不中斷的維運需求。此架構中兩台實體伺服器分別運行了兩個資料庫的程序，過程為而兩台實體伺服器中的主資料庫必須先做鏈狀同步。同步完成後再做兩台本機上的兩個資料庫的程序以串接方式完成主資料庫與副資料庫的同步。一旦所有同步完成後，則任一時間任一台機器都可以停止運作進行維護。假設 A-SERVER 的伺服器硬體損壞需維修，任何時間都可以直接停機 A-SERVER，負載由未關機的 B-SERVER 承擔。

當 A-SERVER 可以恢復時；先將 B-SERVER 的副資料庫伺機器程序停止，將副資料庫伺機器資料複製到 A-SERVER 的主資料庫程序後，此時 A-SERVER 的資料已經有了最新的同步中資料。再來重新啟動 A-SERVER 的主資料庫既可快速恢復鏈狀的同步，然後再啟動 B-SERVER 的副資料庫與重做 A-SERVER 的副資料庫。網站上 MySQL on Amazon EC2 一文中也提出這個架構同時也不少人做了許多實作上的討論[14]。

4. 研究成果

本章將整理歸納效能測試的結果並解釋其背後的意義，也就是研究後所得到的結論。將結論加上所提出的各個重點及方法，彙整成研究成果。

4.1 虛擬機器解決方案

針對機器數量及架構能量均等化之需求，此研究我們使用了 VMware(www.vmare.com) 虛擬機器解決方案(virtualization-solutions)，我們達成了以少量機器虛擬出倍數機器數量，進而測試了原本可能因機器數量不足而無法達到的測試實作，加上實作邏輯架構的設計以盡可能之方式平均主機資源以符合我們在資料庫同步中比較各伺服器角色在效能上的差異，進而提高時驗數據之效度。

4.2 MySQL 資料庫之樹狀同步測試實作

在設計的環境實作結果中，取出每個架構中資料新增(Insert) 320000 筆的等量資料，做網頁輸入完成時間與資料庫完成時間比。我們的結論是使用樹狀結構兩個階層或三個階層時，在樹根節點數量不多的情況下網頁完成時間與資料庫同步的速度很接近。資料新增完成的時間則是機器數量越少，階層數量越少新增資料的速度越快。

真實情況下機器數量與被查詢能力並非為線性成長，因為此實作並未包含查詢能量的測試，所以樹狀結構只用估算法則去推算比例。在樹狀結構上，假設把主資料庫伺服器當成可查詢能量的一個基本單位 100，每台副伺服器也同樣為一個基本單位 100。從架構 A、B、C 他的可供查詢能量分別為：A:100、B:200、C:400。而這幾個架構在完成 320000 筆網頁資料的新增所花的平均時間為：A:110 秒、B:150 秒、C:200 秒。圖 14 是依兩者比例所產生的圖表。我們的結論是針對資料新增當副資料庫節點增加，網頁新增資料平均完成時間小幅度增長，但可提供查詢資料的能量呈正向成長！

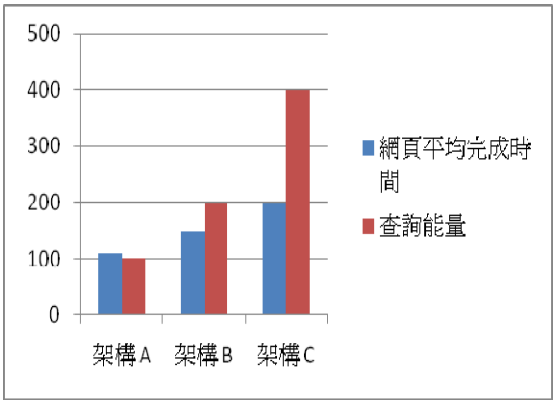

圖 14 樹狀結構網頁平均輸入完成時間與資料庫可供查詢能量比

4.3 MySQL 資料庫之鏈狀同步測試實作

針對六種同步架構進行了數種不同資料量的負載均衡新增，針對資料新增在鏈狀架構而言，資料庫數量越少同步完成時間越短，但網頁完成時間則越長。當鏈狀架構中資料庫數量增加，則新增的資料負載會快速的平均分散在每台資料庫上。資料新增的同時資料庫同時也忙著進行同步的工作，當資料分散到越多的資料庫上，雖然同步完成的時間也就越長，但網頁完成時間越短。以往大型網站常面對到會員數成長，尤其在尖峰時段網站因為無法負荷龐大的連線而整個癱瘓無法運作，大部份問題發生在前端網頁承受了大量的連線(TCP/IP Sessions)，因為後端資料庫無法及時回應所以造成連線無法消化掉而快速累積。一旦有了連線無法消化快速累積的問題，很快的前端網頁伺服器資源就耗盡，接下來就是變成無窮客訴的惡夢。

資料庫的鏈狀同步架構因為可以快速把能量分散到各資料庫節點上，所以前端網頁伺服器可以快速回應而不致累積連線，使用者在操作網頁時也不會產生延遲問題。此架構可以說是快速的把資料接收進系統中，然後再透過資料庫同步的機制把能量慢慢消化掉。所以此架構並不適合用於大量且持續性的新增資料需求系統，而是非常優異、能解決瞬間大量新增資料的解決方案。根據我們的實作結果，圖 15 中有樹狀架構(B、C、D)與鏈狀架構(E、F)，在等量資料新增及等量機器數量上鏈狀架構的網頁完成時間確實比樹狀架構來的短，鏈狀架構非常適用於資料新增，跟我們的結論相吻合。

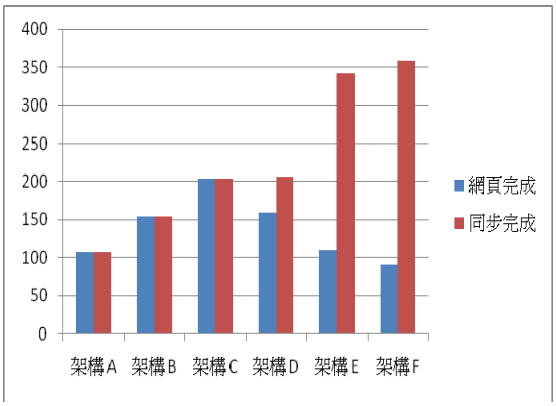

圖 15 網頁輸入完成時間與資料庫完成時間比

5. 未來展望

在程式方面可以加強參雜入其它如 Ajax、Flash 等影像互動式元素，讓資料庫的監控或效能測試表現能更自動化及人性化。第三方軟體有提供一些資料庫操作語法分析功能，將分析資料饋送給 MySQL 公司的 MySQL proxy 可以將監控的部份提昇到有調較效能的能力。當沒有實驗機器數量限制時，可以全部使用實體機器而非虛擬機器，並且將基本的同步架構作 4、8、16、32、64 台實體機器以等倍數增加的方式求得更精準顯著的實作數據。樹狀同步與鏈狀同步架構搭配組合變化後的效能消長分析。使用不同之 MySQL 資料庫引擎在相同的機器與架構下對同步效能的影響，或將數種不同的資料庫引擎混合使用在同步的架構中求得特定系統最佳解決方案的搭配。使用開放原使碼整合虛擬機器解決方案(Xen Virtual Machine)，資料庫同步(MySQL replication)，檔案系統同步(Rsync[21]，Unison[27])，負載平衡(LVS)等機制創造出一個能快速複製、瞬間放大效能、動態遷移的網站平台。

6. 結論

MySQL 研發團隊多年來一直在有關同步技術如何能好再更好、如何更快、更有效率。同步的方式一般都是手工打造，維護也是相同，使用者為了減低操作負載大多使用了第三廠商提供的軟硬體負載平衡解決方案，但第三廠商提供的軟體過於雜亂，目前並無任一單位予以有效評估，對於使用者來說風險不一、所花的測試時間也難以估計，雖然 MySQL 公司目前也在努力以自家軟體解決這個問題，讓整體運作上更有效率也易於維護。資料庫同步天生有幾個限制！來源主資料庫伺服器一定只能有一台。鏈狀只能有一個鏈，不能鏈串鏈的去倍增能量。如果這些架構可以打破，資料庫同步技術就能更上一層樓。許多使用 MySQL 的使用者是因為擁有專業技術及良好的資訊素養，所以能夠以經濟的方式去拼湊出整個資訊平台。MySQL 資料庫其實可以與很多它類軟體護相搭配。如果能有效整合這類軟體，提出有效率的完整架構，再設計成一個完整包裝，以符合好用、好安裝、容易操作的原則下，能夠讓一些懂得基本基礎操作的資訊從業人員也能輕鬆擁抱資料庫。

參考文獻

[1] 2008 Web Server Survey, Netcraft Ltd. http://news.netcraft.com/

[2] Alexander,MySQL Enterprise Solutions, page 204, Wiley, 2003.

[3] Businesswire.com, http://www.businesswire.com/portal/site/google/index.jsp?ndmViewId=news_view&newsId=20080116005349&newsLang=en

[4] Chester Gnull and Laverta Voyd, Tech Tips with Gnull and Voyd, February 1st, 2007 Linux Journal, Recover a dropped MySQL table and save partition images. http://www.linuxjournal.com/article/9488

[5] Comparison of Statement-Based and Row-Based Replication, http://dev.mysql.com/doc/refman/6.0/en/replication-sbr-rbr.html

[6] David Axmark, http://en.wikipedia.org/wiki/David_Axmark

[7] Glyn Moody, Linux Journal, An Interview with Marten Mickos of MySQL, January 1st, 2007. http://www.linuxjournal.com/article/9224

[8] Implementation of Multi-Versioning, http://dev.mysql.com/doc/refman/5.1-maria/en/innodb-multi-versioning.html

[9] Jeremy Zawodny, High Performance MySQL,Page 140, 141, O'Reilly, 2004

[10] M. D. Giacomo , D. Chote, G. Harrison, Lessons Learned in Building a Highly Scalable MySQL Database Presentation, http://en.oreilly.com/mysql2008/public/schedule/detail/625

[11] Michael Kofler,Apress.The Definitive Guide to MySQL 5, Page 382, Apress 2006.

[12] M. Kruckenberg and J. Pipes,Pro MySQL, page 614, Apress 2005.

[13] MySQL AB, MySQL® Administrator's Guide and Language Reference, Chapter 5.4. Replication Implementation Details, MySQL Press, 2006.

[14] MySQL on Amazon EC2.

[15] MyTOP in details, http://www.opensourcetutorials.com/tutorials/Server-Side-Coding/Administration/mytop/page1.html

[16] Phil Hildebrand , Applied Partitioning and Scaling Your Database System Presentation, Santa Clara Convention Center in Santa Clara, California. MySQL conference, 2008.

[17] P. McCullagh, Scalable BLOB Streaming Infrastructure for MySQL.

[18] PrimeBase technologies, PrimeBase XT Storage Engine for MySQL. http://www.primebase.org/

[19] Robert Hodges, CTO of Continuent, Continuent Tungsten: Proxies on Steroids for HA and Performance. http://en.oreilly.com/mysql2008/public/schedule/detail/2582 http://www.continuent.org/HomePage

[20] Robin Schumacher, Director of Product Management MySQL AB, Using MySQL Replication for Scale-Out and High Availability(white paper). http://www.mysql.com/news-and-events/on-demand-webinars/replication-20070208.php

[21] Rsync, Fast incremental file transfer tool. http://samba.anu.edu.au/rsync/

[22] Sasha Pachev, Understanding MySQL Internals, page214, O'Reilly, 2007.

[23] S Drake, W Hu, D. M. McInnis, M. Skold, L. Thalmann, M. Tikkanen, O. Torbjornsen, and A. Wolski, Architecture of Highly Available Databases, ISAS 2004, LNCS 3335,© Springer-Verlag Berlin Heidelberg 2005.

[24] S. Sahri,Design of a Scalable Distributed Database System:SD-SQL Server,0-7803-9521-2/06, IEEE.

[25] T. Groothuyse, S. Sivasubramanian, G. Pierre, Globetp: template-based database replication for scalable web applications May 2007 ,WWW '07: Proceedings of the 16th international conference on World Wide Web , Publisher: ACM.

[26] The BLOB and TEXT Types, MySQL AB, http://dev.mysql.com/doc/refman/5.0/en/blob.html

[27] Unison, File-synchronization (bi-directional reconciliation) tool. http://www.myoops.org/twocw/mit/Electrical-Engineering-and-Computer-Science/6-033Spring-2005/Assignments/detail/pset34.htm http://www.cis.upenn.edu/~bcpierce/unison/

[28] Y. Lin, B. Kemme,P. M. Marta, J. P. Ricardo , Applying database replication to multi-player online games ,October 2006 NetGames '06: Proceedings of 5th ACM SIGCOMM workshop on Network and system support for games ,Publisher: ACM.