資料庫測試工具

M0629010 呂毓軒

目錄

壹、資料庫系統介紹

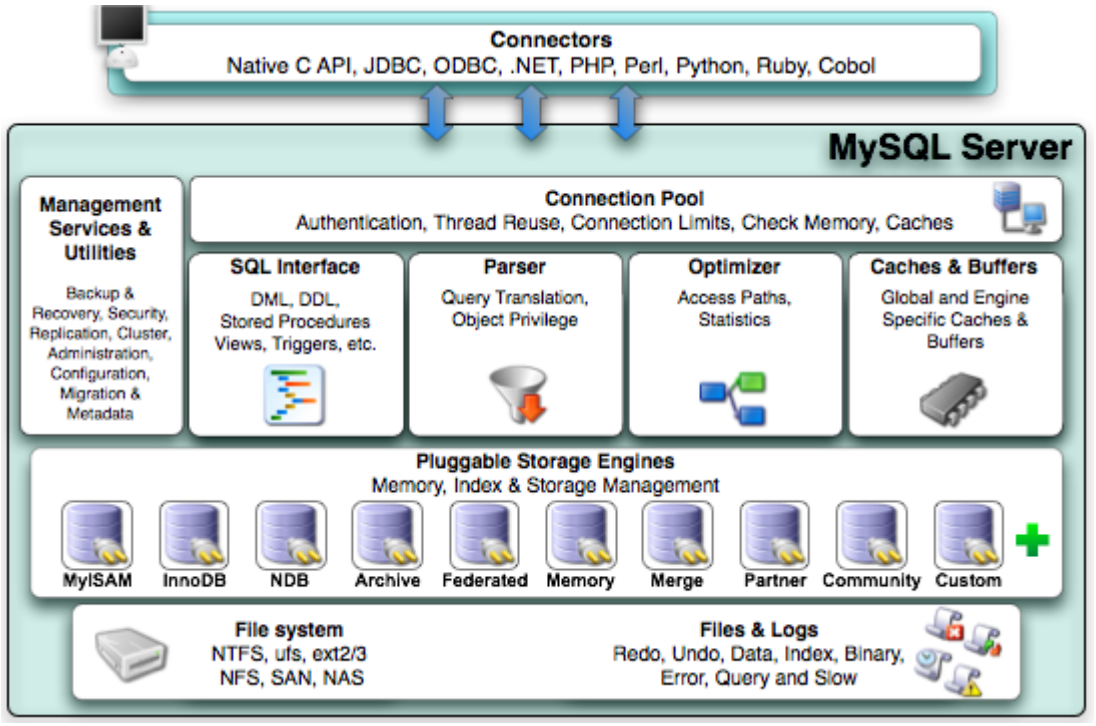
一、MySQL

1. 概念

數據庫實例是由數據庫後台進程/線程以及一個共享內存區組成。共享內存可以被運行的後台進程/線程所共享。需要注意的是，數據庫實例才是真正用來操作數據庫文件的。

2. MySQL 體系架構

如圖所示：



1. 最上層的連接服務，用於不同語言與 SQL 的交互。

   可以通過【show variables like '%connections%'】命令查看 MySQL 實例的最大連接數和單個用戶的最大連接數。

2. 第二層 MySQL Server，大多數 MySQL 的核心服務功能都在這一層，包括查詢解析、分析、優化、緩存，所有跨存儲引擎的功能都集中在這一層實現。

   - Management Serveices & Utilities：系統管理和控制工具

     備份和恢復的安全性，複製，集群，管理，配置，遷移和元數據。

   - Connection Pool：連接池

     進行身份驗證、線程重用，連接限制，檢查內存，數據緩存；管理用戶的連接，線程處理等需要緩存的需求。

   - SQL Interface：SQL 接口

進行 DML、DDL，存儲過程、視圖、觸發器等操作和管理；用戶通過 SQL 命令來查詢所需結果。

- Parser：解析器

  查詢翻譯對象的特權；SQL 命令傳遞到解析器的時候會被解析器驗證和解析。

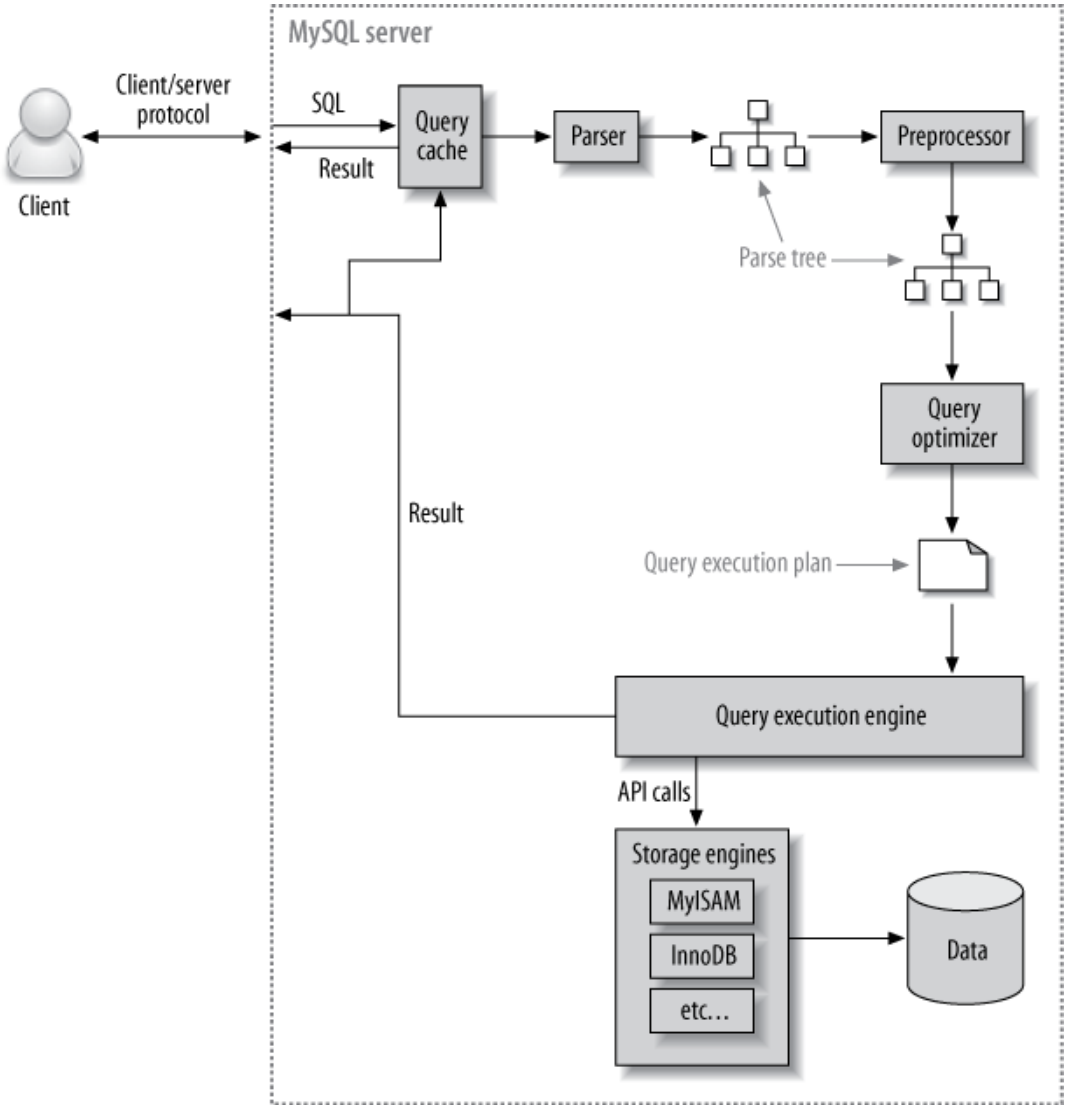
- Optimizer：查詢優化器
- Cache 和 Buffer：查詢緩存

  全局和引擎特定的緩存和緩衝區

3. 第三層為存儲引擎層，存儲引擎負責 MySQL 中數據的存儲和提取，服務器通過 API 於存儲引擎進行通信。

   可以通過【SHOW ENGINES】命令查看各個存儲引擎信息。

3. 執行流程



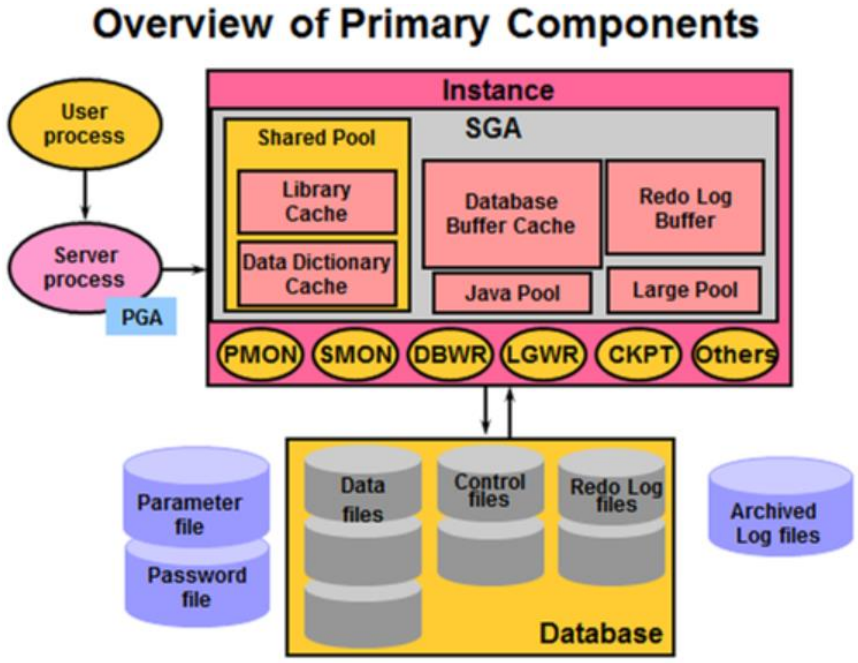
在 MySQL Server 中首先有一個 Cache，用來緩存 SQL 查詢語句的查詢結果，如果緩存命中，則直接返回結果，如果緩存沒有命中，則對 SQL 語句進行語法分析，預處理和查詢優化，最終得到該查

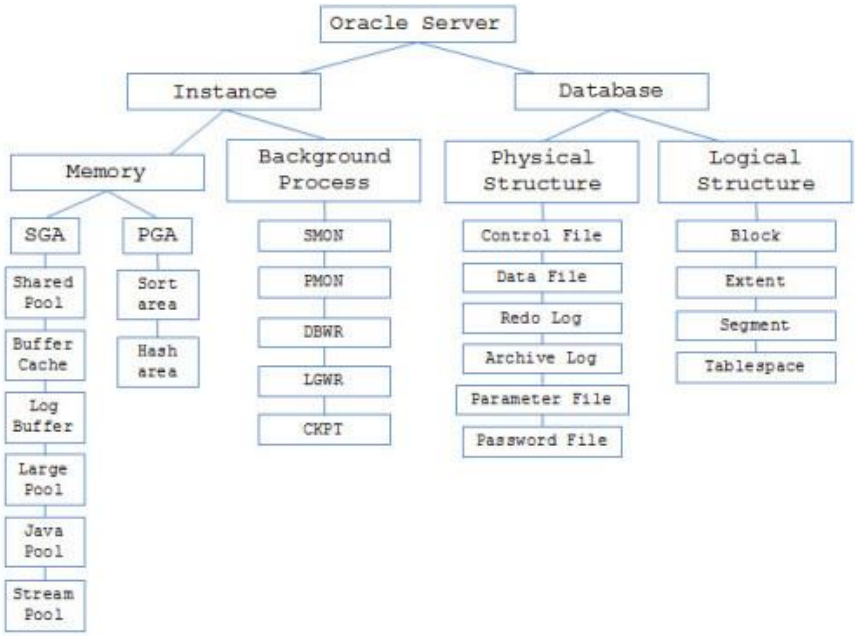
詢的執行計劃，之後，該執行計劃被送往數據庫引擎，得到最終查詢到的數據。數據庫引擎並不會嚴格按照執行計劃進行執行，而是會根據自身的架構和特性進行一些調整，以提高執行效率。

參考：[MySQL 數據庫的體系架構]

二、Oracle

1. Oracle RDBMS 架構圖





Oracle Server = Instance + Database

Instance = Memory + Background Process

Database = Physical Structure + Logical Structure

由幾行短短的公式，便可很清楚的定義出 Oracle 資料庫的架構。

2. 分析：

Oracle Server 主要是由 Instance 與 Database 兩部分所構成:

※ Instance 偏向系統面，由作業系統取得相關資源，包含 Memory 與 Background Process 兩部分，當 Instance 啟動之後，便會由作業系統獲取所需要的 Memory，同時也產生出資料庫運作所需要的 Background Process。

Memory 包括 SGA 與 PGA ，其中 SGA 的成分有 Shared Pool (用於程式快取 " Library Cache" 與 資料字典快取 " Dircetory Cache")、Buffer Cache (用於存放讀取過的 Data Block)、Log Buffer (存放交易日誌的快取)、Large Pool (資料庫特定的操作所會使用到的快取，例如使用 Shared Server 架構或是使用 RMAN 做備份還原時將使用此記憶體空間)、Jave Pool (程式有使用到 Java 相關功能時會使用此記憶體空間、Stream Pool (當資料庫有使用到 Stream 功能時才會使用到此空間)。

Background Process 主要有五個，缺一不可，若其中一個 Process 中斷，則會造成 Instance Crash，包括 SMON (用於 Instance recovery 與回收系統 Temporary 空間)、PMON (用於 rollback 失敗的交易與動態註冊 Listener)、DBWR(用於將 Buffer Cache 中的 Data Block 寫入 Data File)、LGWR(用於將 Log Buffer 中的交易資訊寫入 Redo Log)、CKPT (用於更新資料庫的 SCN)。

※ Database 可分為實體結構 (Physical Structue) 與邏輯結構 (Logical Structure)。實體結構表示實際上可看得到的，例如資料庫的控制檔 (Control File)、資料檔(Data File)、重做日誌檔(Redo Log)…等，我們可以利用作業系統的指令將這些檔案列出來，因此稱做實體結構。而邏輯結構則是比較抽象的，例如我們無法拿出一樣實體東西來說明這個就是表格空間 (Tablespace)，表格空間 (Tablespace) 這個概念是需要花費一點想像力去理解的，這些實體上看不到的東西，我們稱做邏輯結構。

另外 Oracle Server 除了 Instance 與 Database 兩大部分之外，還有一個成份為 Oracle Network，這也是資料庫架構中不可或缺的，Oracle Network 主要的成員就是 Listener，使用者必須透過 Listener 來對 Oracle 資料庫做連線，沒有了 Listener，使用者也就無法登入資料庫了。

參考：

Oracle 架构实现原理、含五大进程解析(图文详解)

Oracle 系統架構

Oracle 数据库体系架构详解

Oracle 架構與 MySQL 架構對比

三、Microsoft SQL Server

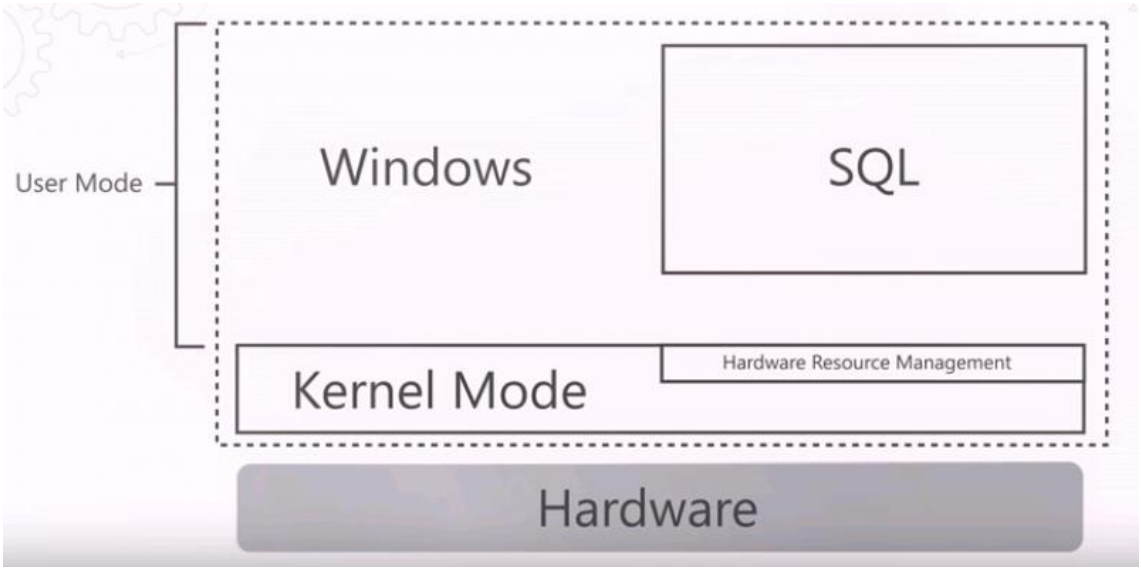
1. SQL Server 架構演進

按照以往微軟對 SQL Server 資料庫產品的釋出節奏，一般情況下是兩年一個大版本更新迭代，比如 SQL Server 2012，2014 和 2016。但是，SQL Server 2017 的釋出僅僅只用了一年時間，而且實現了 Linux 版本 SQL Server 的巨大轉變，並且所有功能和 Windows 版本對齊。很多使用者和從業者對這一點都非常好奇，微軟到底是如何做到這一點的？要回答這個問題，我們從 SQL Server 底層的架構演進來分析這個問題的答案。總結來看，到 SQL Server 2017 的出現，微軟對資料庫底層架構的演進經歷了以下幾個階段：

- 使用 Windows 對 SQL Server 系統進行資源管理：這個階段沒有一個特定的名稱叫法，這個階段的 SQL Server 服務無法突破 Windows 核心對資源的限制。
- SQL OS 階段：爲了使得 SQL Server 資料庫擁有更好的效能，SQL OS（也叫 SOS）出現了。
- Drawbridge 的出現：研究性專案，用於實現應用的沙盒（Sandbox），最開始不是為 SQL Server 專門設計的，但在 SQL Server 2017 中扮演中非常重要的角色。
- SQL PAL 的出現：SQL Server 2017 整合了 SQL OS 和 Drawbridge，進行底層封裝，形成了 SQL PAL 層。

2. SQL Server 2005 之前

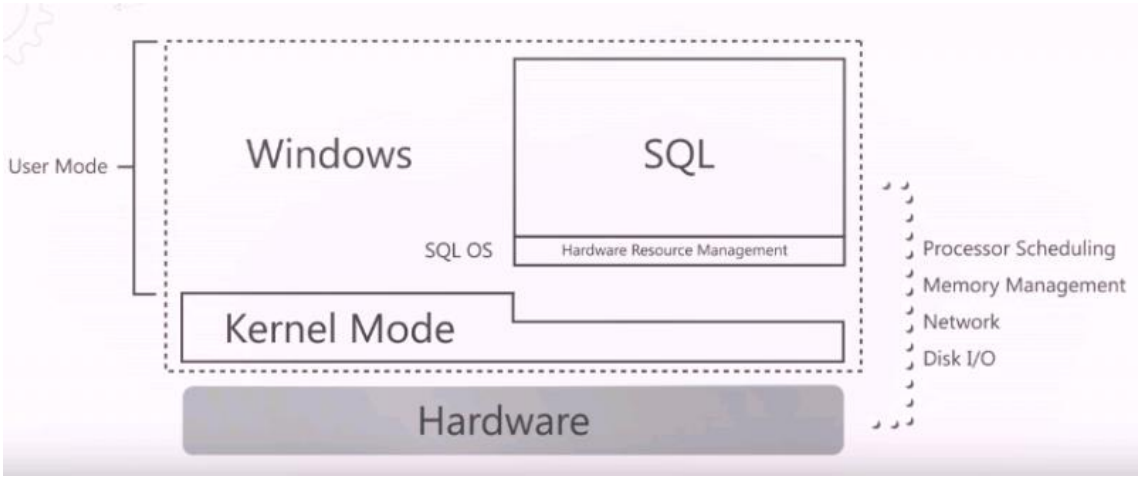
追述到 SQL Server 2005 版本之前（SQL Server 2000 及更早版本），SQL Server 服務是以一個使用者態程序執行在 Windows 作業系統中，這個服務程序與其他普通的程序沒有任何差異，它依賴於作業系統核心對底層硬體資源進行管理和互動，SQL Server 服務本身沒有對系統資源的管理能力。具體的架構圖大致如下所示：



這個架構最大的缺點是 SQL Server 服務本身無法突破 Windows 核心對系統資源使用的限制，換話句話說 SQL Server 無法榨乾系統硬體資源，加之缺乏對作業系統資源的控制能力，只能依賴於作業系統核心對底層硬體資源進行排程，因此 SQL Server 服務很難最大限度充分利用系統所有硬體資源，阻礙了 SQL Server 系統性能的進一步提升。

3. SQL OS

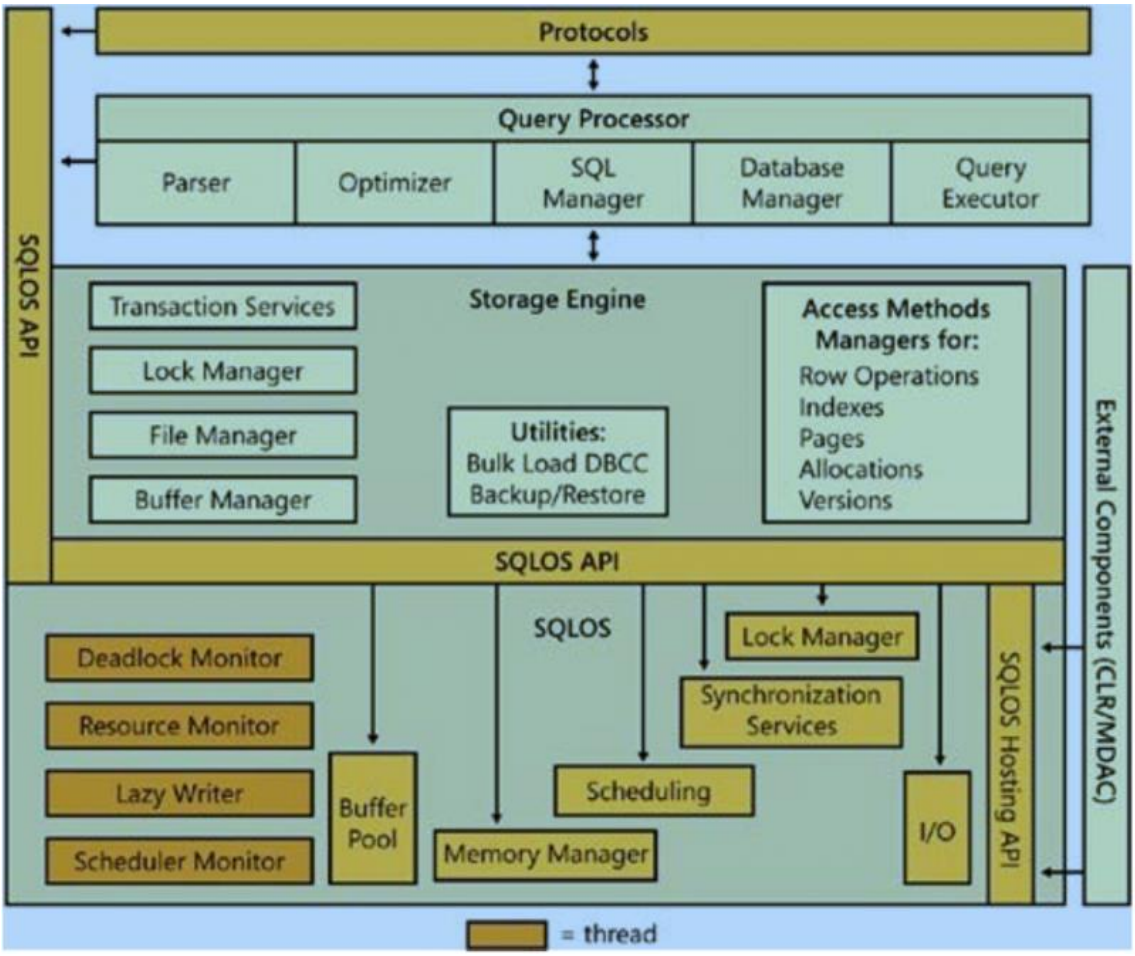
爲了解決上面的問題和獲取更好的效能，微軟花了很大的力氣來抽象一箇中間層對系統硬體資源進行排程和管理，併發布在 SQL Server 2005 版本中，也因此 SQL Server 2005 版本經歷了長達 5 年的時間才得以面世。這個對硬體資源進行集中排程和管理的中間層叫著 SQL OS（也叫著 SOS）。SQL OS 的主要職責包括：Processor Scheduling，Memory Management，Network，Disk I/O 使得 SQL Server 效能最大化（可以使得 SQL Server 使用者程序最大限度的充分利用作業系統硬體資源）。SQL Server 2005 引入了 SQL OS 後的底層架構圖如下所示：



SQL Server 2005 賦予了 SQL OS 非常全面的資源管理功能，涉及到資料庫系統核心功能的方方面面，具體包括：

- Deadlock Monitor：死鎖監控
- Resource Monitor：資源監控
- Lazy Writer：延遲寫，將隨機 I/O 寫，轉化為順序 I/O 寫
- Scheduler Monitor：排程器監控
- Buffer Pool：快取池
- Memory Manager：記憶體管理
- Scheduling：排程
- Synchronization Service：同步服務
- Lock Manager：鎖管理器
- I/O：I/O 資源管理

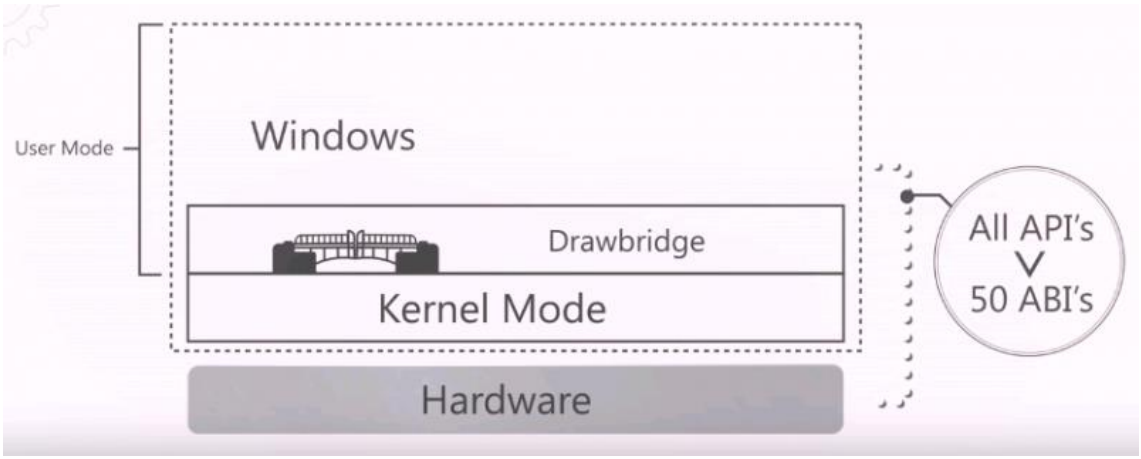
詳細架構如下圖 SQLOS API 部分所示：



有了 SQL OS 層次的抽象，得以在資料庫內部實現對系統資源的集中管理，擺脫系統核心對 SQL Server 資源使用的限制，使得 SQL Server 服務隊系統資源有了很強的控制能力，SQL Server 2005 效能有了大幅的提升，成了微軟關係型資料庫歷史上劃時代的版本，也為 SQL Server 2017 能夠提供跨平臺能力提供了可能性，可以毫不誇張的說，沒有 SQL OS 的出現，微軟不可能在如此短的時間內實現 Linux 版的 SQL Server。

4. Drawbridge

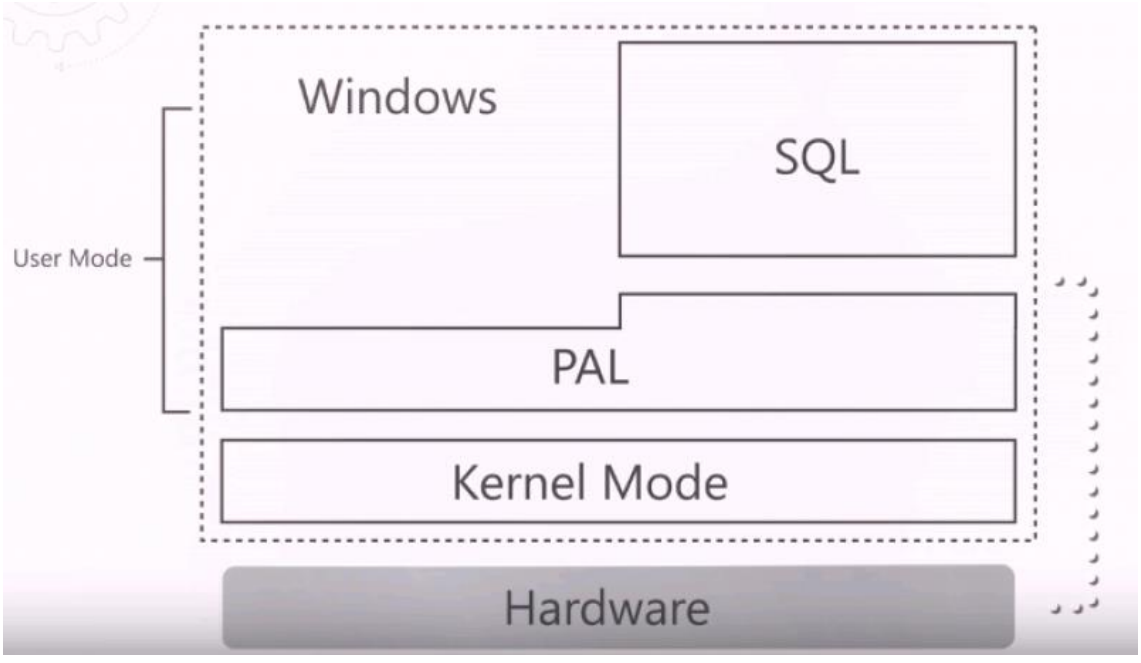
微軟研究院在 2011 年 9 月建立了一個全新的研究性專案，名稱叫 Drawbridge，目的是提供應用程式新的虛擬化資源隔離解決方案，減少虛擬資源的使用，使得在同一個硬體主機上，可以執行更多的虛擬機器（類似於 Docker 產品對硬體資源的管理）。Drawbridge 其中一個非常重要的元件 Library OS 僅依靠約 50 個底層核心應用二進制介面（ABI：Application Binary Interface）實現了一千多個常用的 Windows API，同時還具備了為其他元件提供宿主的能力，比如：MSXML 和 CLR 等元件。 在 Windows 10 版本中存在著 Drawbridge 的大量應用。



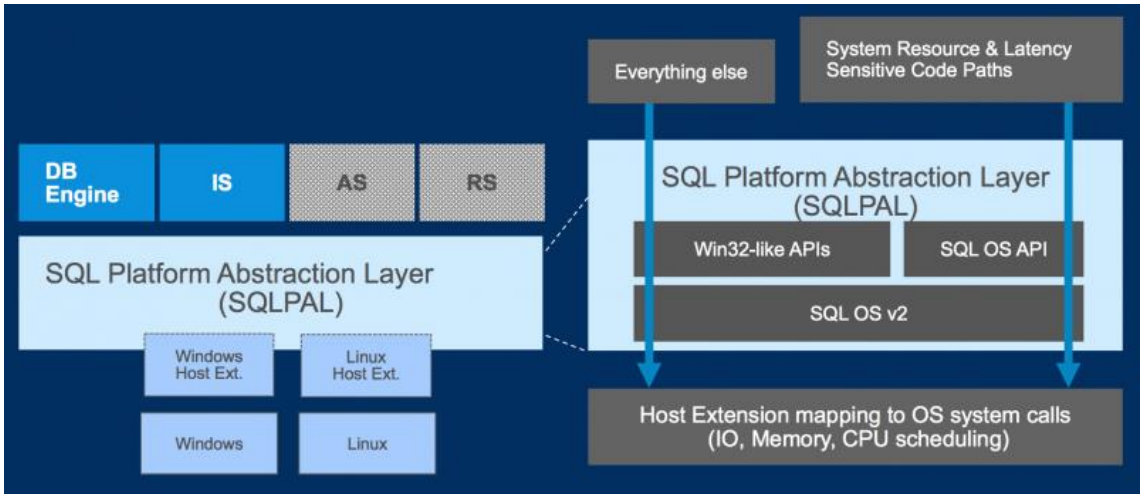
5. SQL PAL

SQL Server 資料庫團隊基於 Drawbridge 專案與 SQL OS 兩者進行了必要的重寫和充分的融合，形成了新一代資料庫底層抽象和封裝，叫 SQL PAL （Platform Abstract Layer），同時將上層邏輯程式碼移植到 SQL PAL 之上。如此，微軟只需要確保 SQL PAL 層可以在 Windows 平臺和 Linux 平臺執行良好即可。這樣 SQL Server 即使執行在 Linux 平臺，也無需修改 SQL Server 本身的程式碼，SQL Server 自己本身與平臺無關。能做到這一點完全是由 Drawbridge 中的 ABI（Application Binary Interface）來達到目的的，這些 ABI 我們叫著 Host Extension，所以爲了支援 SQL Server 2017 的跨平臺特性，微軟只需要實現基於 Windows 平臺的 Host Extension 和 Linux 平臺的 Host Extension，這樣做最大的好處是：

- 大大縮短開發週期：微軟無需對 SQL Server 本身做任何的程式碼修改就可以將 SQL Server 移植到 Linux 平臺。
- 產品功能一貫性：對 SQL Server 新功能、新特性的支援，無需對兩個平臺進行重複開發，Windows 平臺支援了，Linux 平臺也就支援了，保持了產品功能的一致性。
- 良好的後期維護性：假如 SQL Server 存在某個 Bug，只需要修復 SQL Server 本身，那麼 Windows 平臺，Linux 平臺上相應的 Bug 也同樣被修復掉了，具備良好的可維護性。



以上架構圖是比較巨集觀的層面展示，以下是 SQL PAL 功能更為詳細的描述架構圖：



從這張圖，我們可以清晰的看到 SQL PAL 層次對於 Host Extension 的呼叫，以及構建在這層次之上的 SQL Server 服務，包括：資料庫引擎、整合服務、分析服務和報表服務。

6. SQL PAL 效能影響

提到 SQL PAL 對 SQL Server 2017 資料庫服務的影響，很多使用者最為擔憂的應該就是效能的影響了。請不要擔心，根據 TPC-H 測試來看，SQL Server 2017 相對於 SQL Server 2016 來看，效能不但沒有任何損失，反而效能不降反升。

參考：

MSSQL - 架構分析 - 從 SQL Server 2017 釋出看 SQL Server 架構的演變

貳、資料庫測試流程

1. 資料庫設計測試

資料庫是應用的基礎，其性能直接影響應用軟體的性能。為了使資料庫具有較好的性能，需要對資料庫中的表進行規範化設計。規範化的範式可分為第一範式、第二範式、第三範式、BCNF 範式、第四範式和第五範式。一般來說，邏輯資料庫設計應滿足第三範式的要求，這是因為滿足第三範式的表結構容易維護，且基本滿足實際應用的要求。因此，實際應用中一般都按照第三範式的標準進行規範化。但是，規範化也有缺點：由於將一個表拆分成為多個表，在查詢時需要多表連接，降低了查詢速度。故資料庫設計的測試包括前期需求分析產生資料庫邏輯模型和後期業務系統開發中的測試兩部分(這裡指的是後者)，我在這裡稱為實體測試。

資料庫是由若干的實體組成的，包括(表，視圖，存儲過程等)，資料庫最基本的測試就是實體測試，通過對這些實體的測試，可以發現資料庫實體設計得是否充分，是否有遺漏，每個實體的內容是否全面，擴展性如何。

實體測試，可以用來發現應用軟體在功能上存在的不足，也可以發現數據冗餘的問題。經過測試，測試人員對有異議的問題要及時和資料庫的設計人員進行溝通解決。

參考：第三正規化

2. 數據一致性測試

在進行實體測試後，應進一步檢查下面的內容以保障數據的一致性：

2.1 表的主鍵測試根據應用系統的實際需求，對每個表的主鍵進行測試，驗證是否存在記錄不唯一的情況，如果有，則要重新設置主鍵，使表中記錄唯一。

2.2 表之間主外鍵關係的測試資料庫中主外鍵欄位在名稱，數據類型，欄位長度上的一致性測試。

2.3 級聯表，刪除主表數據後，相應從報表數據應同時刪除的問題例如學生表和學生成績表，學生數據已經刪除，成績表中相應學生的成績記錄應同時刪除。

2.4 存儲過程和觸發器的測試存儲過程可以人工執行，但觸發器不能人工處理，所以在對存儲過程和觸發器執行的過程中針對 SQL SERVER2005 及以上版本可以使用 Microsoft SQL Server Profiler 性能測試工具進行測試。

Microsoft SQL Server Profiler 是 SQL 跟蹤的圖形用戶介面，用於監視資料庫引擎或 Analysis Services 的實例。測試人員可以捕獲有關每個事件的數據並將其保存到文件或表中供以後分析。例如：可以對生產環境進行監視，了解哪些存儲過程由於執行速度太慢影響了性能。

3. 資料庫的容量測試

隨著資料庫系統的使用，數據量在飛速增長，如何在使用前對數據容量的增長情況進行初步估算，為最終用戶提供參考，這在資料庫使用和維護過程中，是非常重要的。可以通過對資料庫設計中基本表的數據大小，和每天數據表的數據產生量進行初步估算。

記錄數據量 = 各個欄位所占字節數的總和

表的數據量 = 記錄數據量*記錄數

資料庫大小 = 各表數據量的總和

當然，資料庫的大小不僅僅只是基本表的大小，還有系統表，視圖，存儲過程等其它實體所占的容量，但最基本的數據是表的數據。另外，資料庫的容量還包括資料庫日誌文件的容量，一般應預留資料庫文件的 2 倍左右。

4. 資料庫的性能測試

應用軟體除了功能外，很重要的一部分就是軟體的性能，而對於資料庫系統，資料庫性能的好壞會直接影響應用軟體的性能，這部分的測試，一般手工測試就顯得無能為力了，這時就要藉助自動化的測試軟體，例如：DataFactory，DataFactory 是一種強大的數據產生器，它允許開發人員和測試人員很容易產生百萬行有意義的正確的測試資料庫，該工具支持 DB2、Oracle、Sybase、SQL Server 資料庫。這樣，就可以模擬出應用軟體長期使用後，海量數據存儲的資料庫的性能狀況。從而儘早發現問題，進行資料庫性能的優化。

這裡要注意，進行性能測試的時候，一定要注意測試環境的一致性，包括：作業系統、應用軟體的版本以及硬體的配置等，而且在進行資料庫方面的測試的時候一定要注意資料庫的記錄數、配置等要一致，只有在相同條件下進行測試，才可以對結果進行比較。否則無法和用戶對軟體的性能的觀點達成一致。

參考：Azure Data Factory

5. 資料庫的壓力測試

說起測試，我們首先想到的就是軟體正確性的測試，即常說的功能測試。軟體功能正確僅是軟體質量合格指標之一。在實際開發中，還有其它的非功能因素也起著決定性的因素，例如軟體的響應速度。影響軟體響應速度的因素有很多，有些是因為算法不夠高效;還有些可能受用戶並發數的影響。

在眾多類型的軟體測試中，壓力測試正是以軟體響應速度為測試目標，尤其是針對在較短時間內大量並發用戶的訪問時，軟體的抗壓能力。但壓力測試往往是手工難以測試的，必須藉助自動化測試工具。常用的壓力測試有：Web 測試、資料庫測試等。

資料庫在大多數軟體項目中是不可缺少的，對於它進行壓力測試是為了找出資料庫對象是否可以有效地承受來自多個用戶的並發訪問。這些對象主要是：索引、觸發器、存儲過程和鎖。通過對 SQL 語句和存儲過程的測試，自動化的壓力測試工具可以間接的反應資料庫對象是否需要優化。

這些自動化的測試工具很多，各有特點，基於 Java 的項目可以使用 JMeter，.Net 項目可以採用.Net 集成開發環境中提供的測試方案。

參考：

MySQL 壓力測試經驗

參、資料庫性能及壓力測試分析

1. 分析原則

- 具體問題具體分析 (這是由於不同的應用系統，不同的測試目的，不同的性能關注點)
- 查找瓶頸時按以下順序，由易到難。

伺服器硬體瓶頸 -> 網路瓶頸 (區域網路可以不考慮) -> 伺服器作業系統瓶頸 (參數配置) -> 中介軟體瓶頸 (參數配置，資料庫，web 伺服器等) -> 應用瓶頸 (SQL 語句、資料庫設計、業務邏輯、演算法等)

※ 注：以上過程並不是每個分析中都需要的，要根據測試目的和要求來確定分析的深度。對一些要求低的，我們分析到應用系統在將來大的負載壓力（併發使用者數、資料量）下，系統的硬體瓶頸在哪兒就夠了。

- 分段排除法，分析的資訊來源：
  - 根據場景運行過程中的錯誤提示資訊
  - 根據測試結果收集到的監控指標資料

2. 分析經驗

參考：

性能測試監控點

Windows 常見性能計數器

常用效能計數器介紹

肆、壓力測試工具

1. MySQL 自帶壓力測試工具 --- mysqlslap

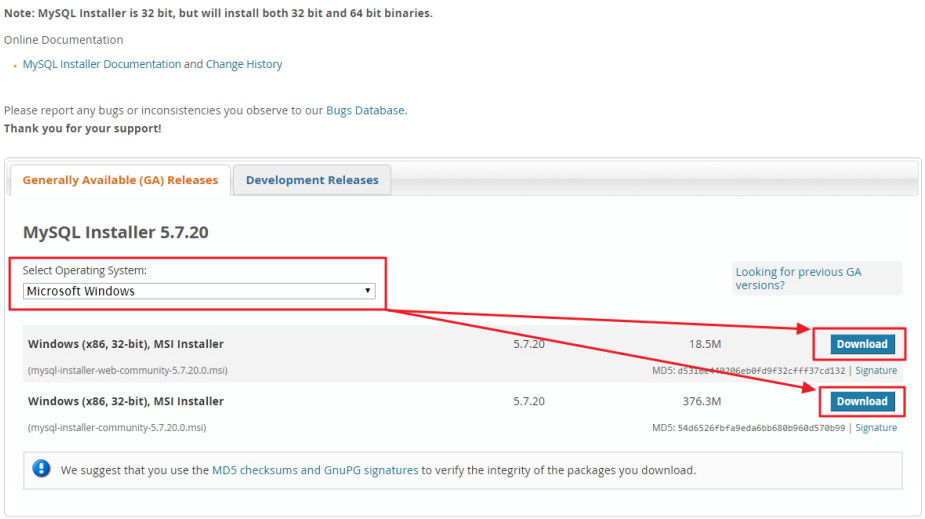
- **實驗環境：**

  Windows 10 作業系統
  MySQL v.5.7.20

- **安裝 MySQL**

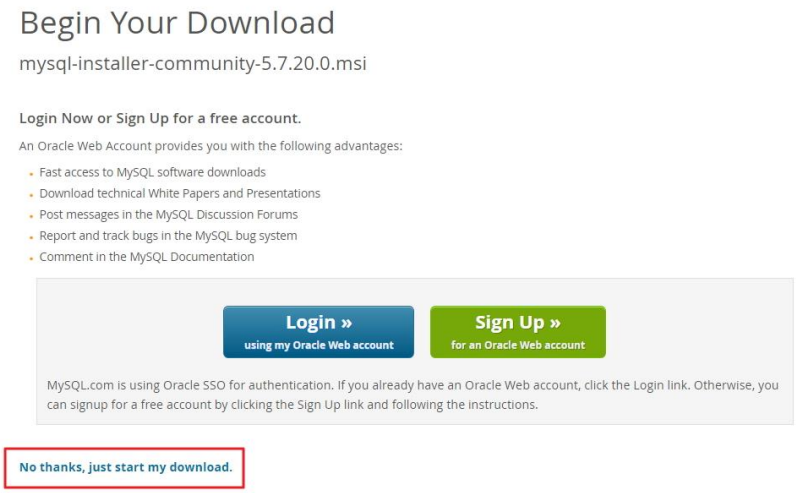
  下載 MySQL Installer 就可以一次擁有 MySQL Server, MySQL Workbench，以及連接函式庫的功能。安裝網址 ：Download MySQL Installer

進到下載頁面有兩種安裝方式，一種是用網路安裝，另外一種是完整安裝檔，這裡選擇用完整安裝檔

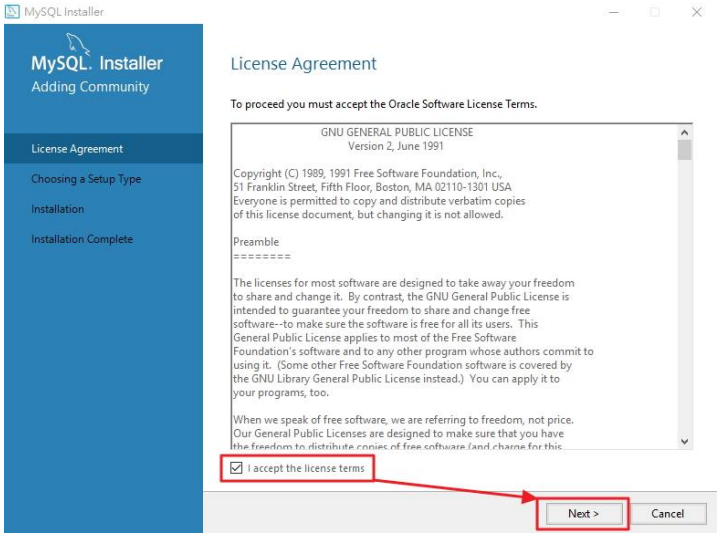


點選完「Download」後會出現註冊頁面，可以不用註冊就下載

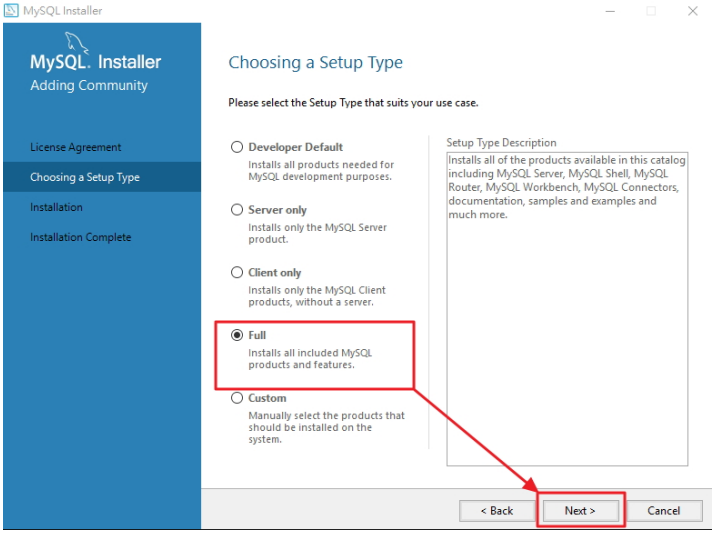
頁面最底下有「No thanks, just start my download」，點完就會直接下載



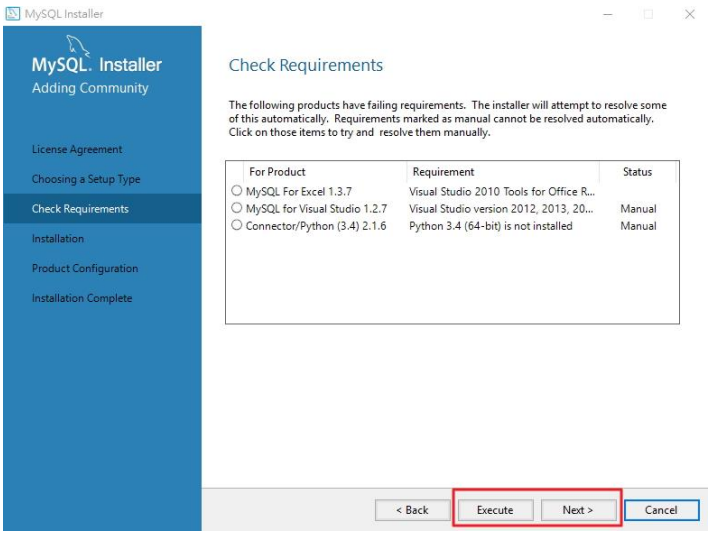
勾選同意條款就按下一步



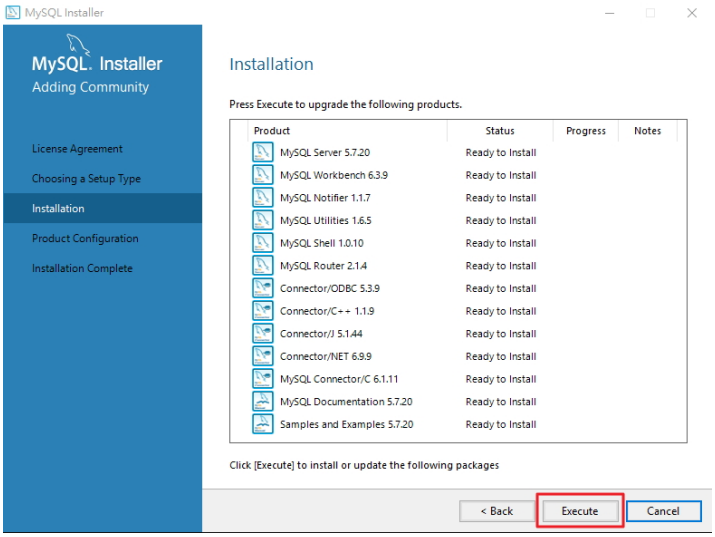
有五種版本可以選擇，這裡選擇安裝「Full」全部完整的 MySQL



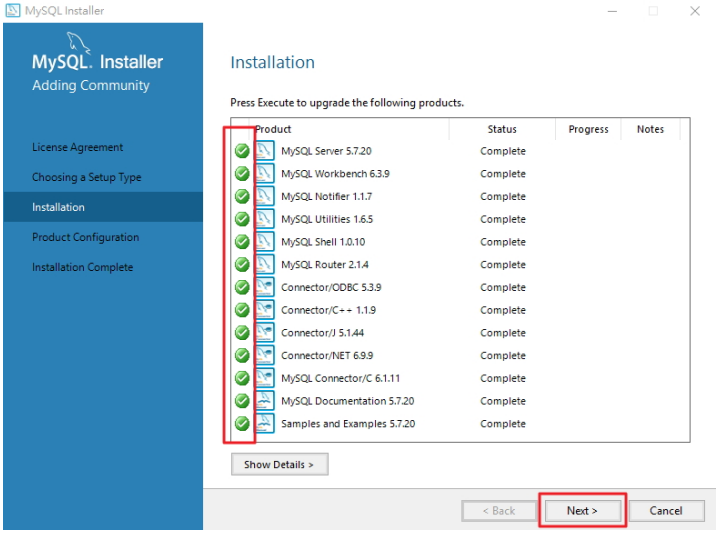
接著這裡有「Excute」可以額外安裝與 Visual Studio 資料庫做連接
也包含自動更新，可以選擇要不要安裝



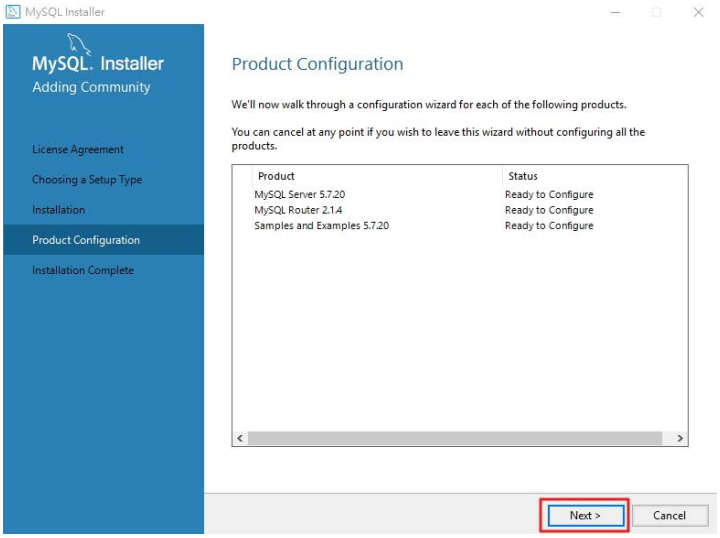
到「Installation」畫面就直接點選「Excute」安裝完所有需要的檔案



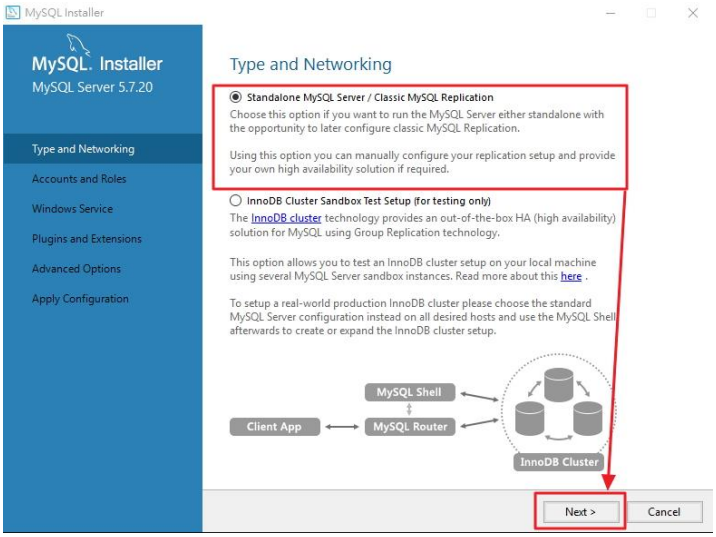
安裝完旁邊會出現「綠色」的勾勾，就可點下一步



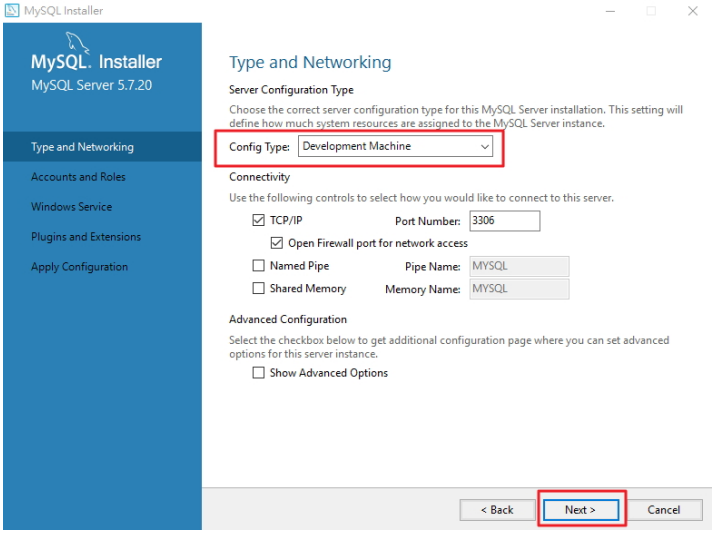
點選下一步



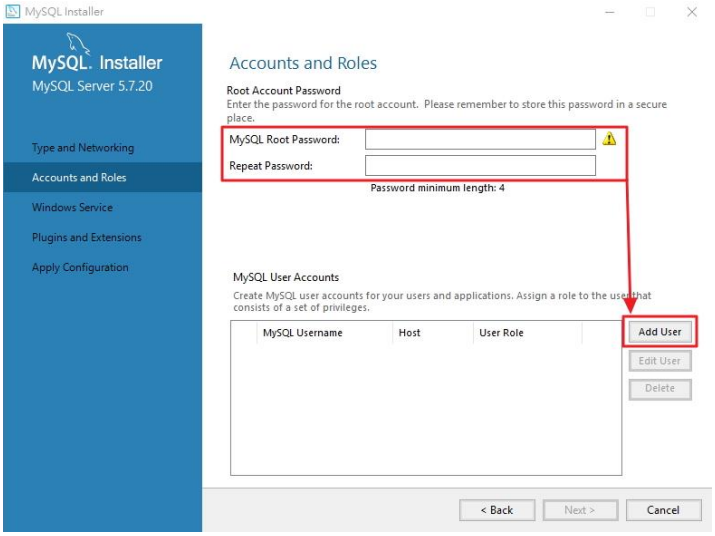
預設值會選擇第一個，就直接用預設的就可以



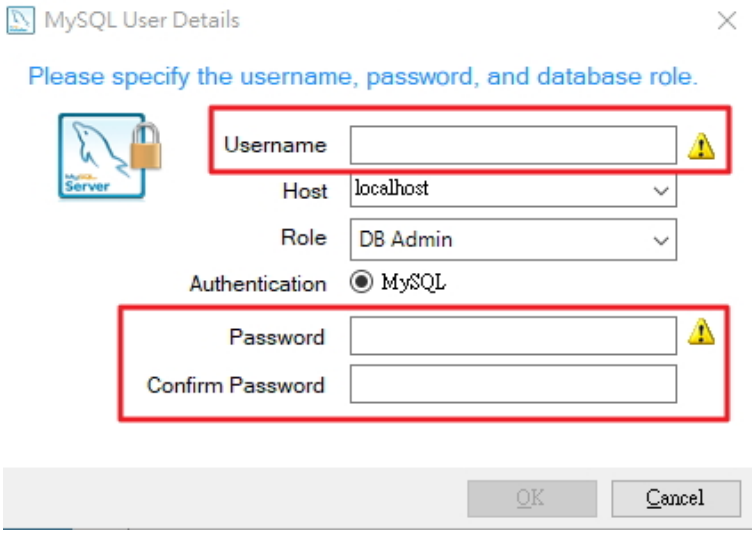
這裡要注意有沒有選擇到「Development Machine」，確認完就下一步



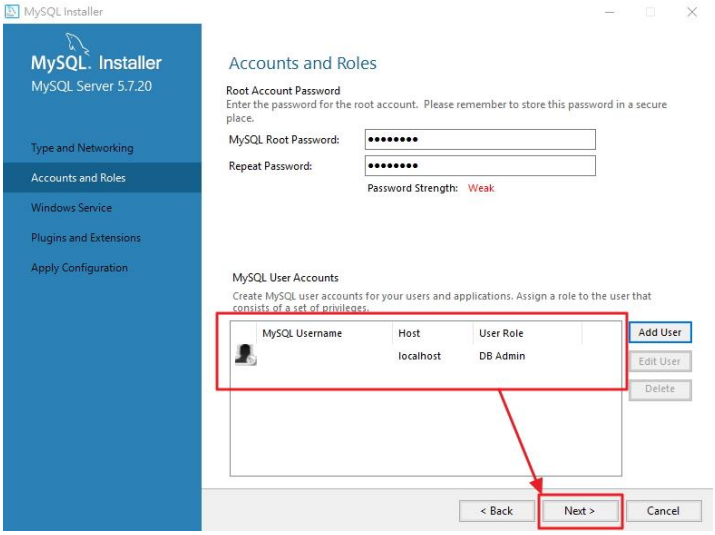
輸入自己的密碼，盡量不要設得太簡單，設定完就點「Add User」



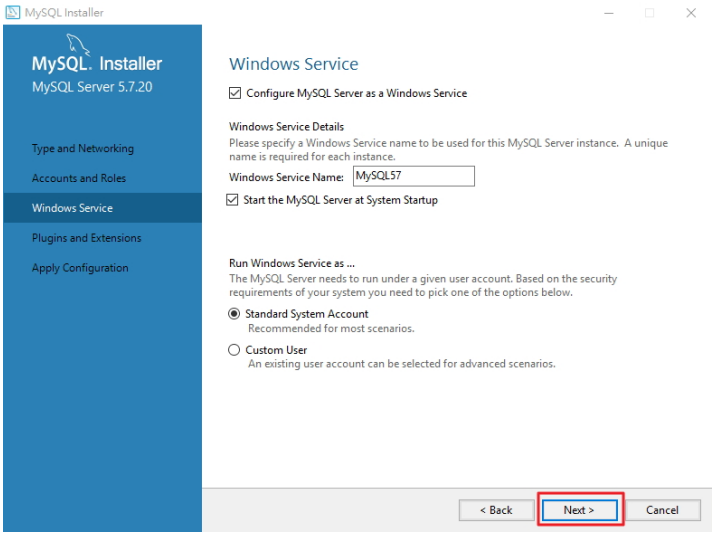
會跳出小視窗，輸入自己想要的使用者名稱跟密碼，Host 選擇「localhost」



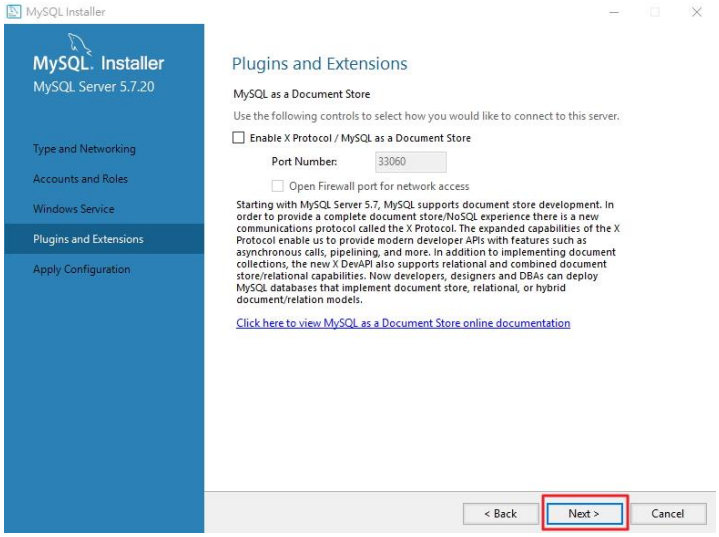
設定完就會出現自己的帳戶，點選下一步



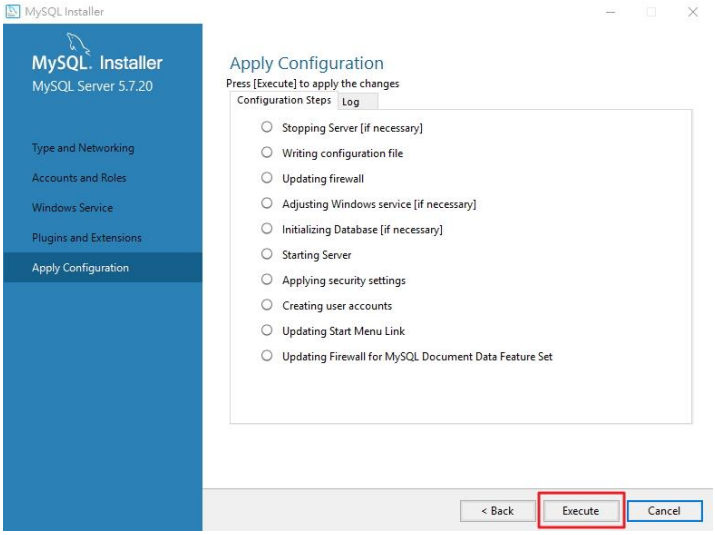
選擇「Standatd System Account」，選完就下一不



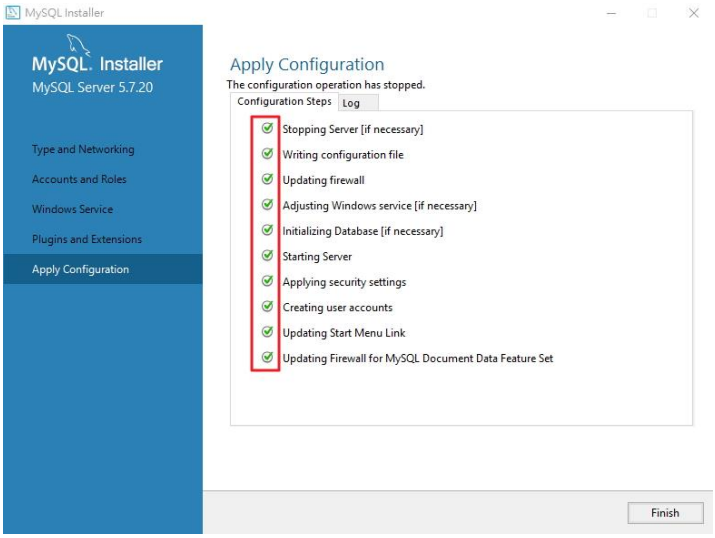
這邊可以不用做設定，直接下一步



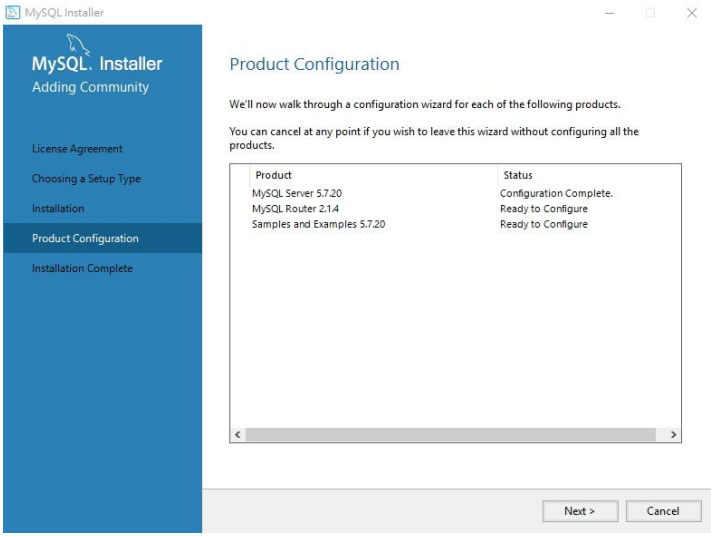
點選「Excute」安裝所有需要的配件



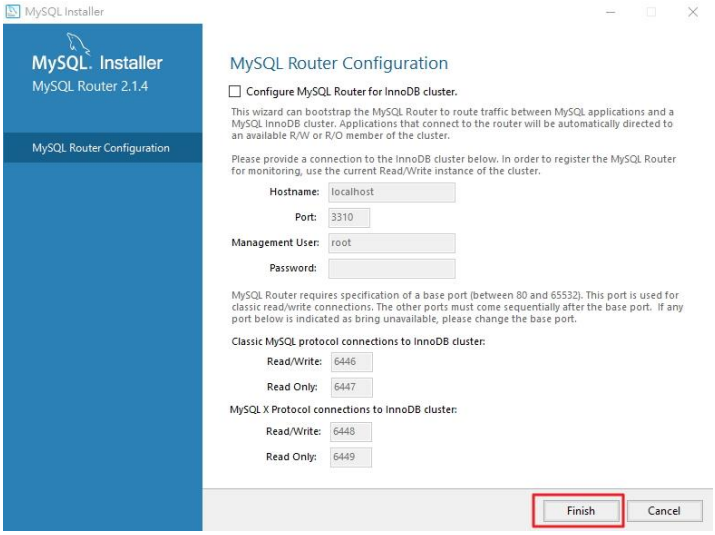
安裝完一樣旁邊會出現綠色的勾勾



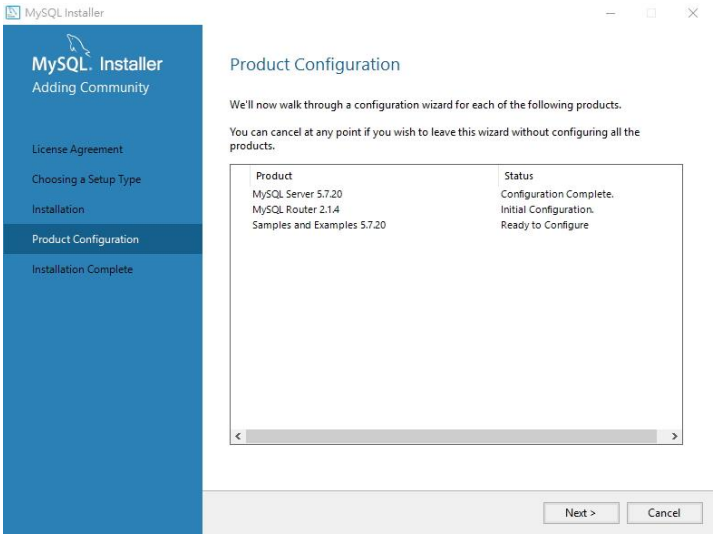
點選下一步



這裡沒有問題就直接點選下一步

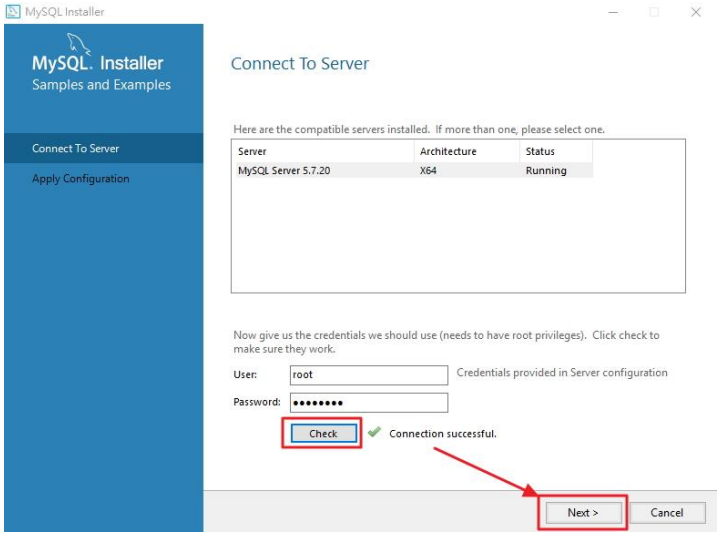


繼續下一步

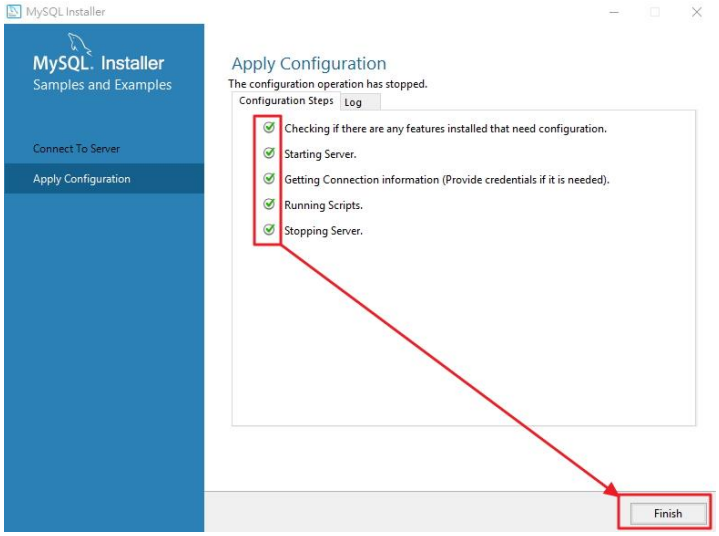


這裡要先按下「Check」旁邊出現綠色的勾勾才能點選下一步

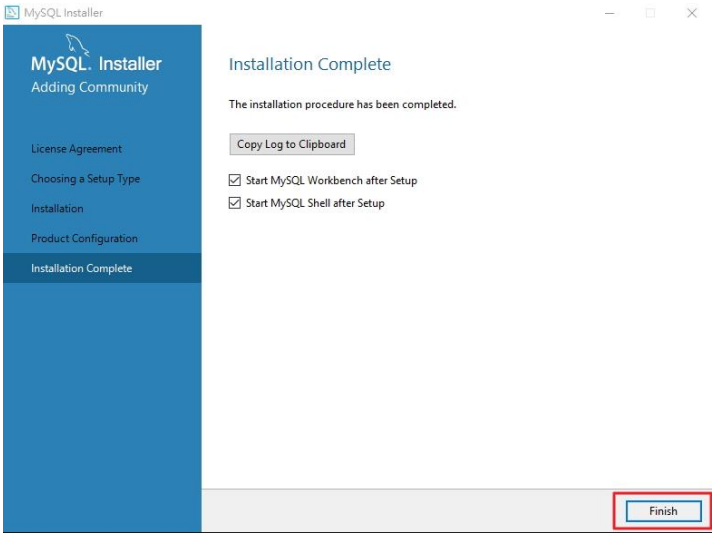
User 的名稱也可以自行做更改



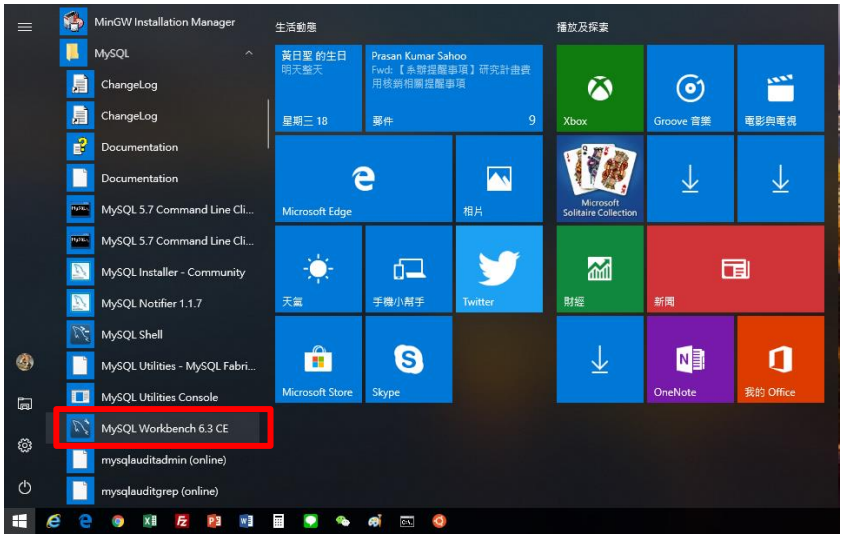
這裡一樣安裝完需要的配件



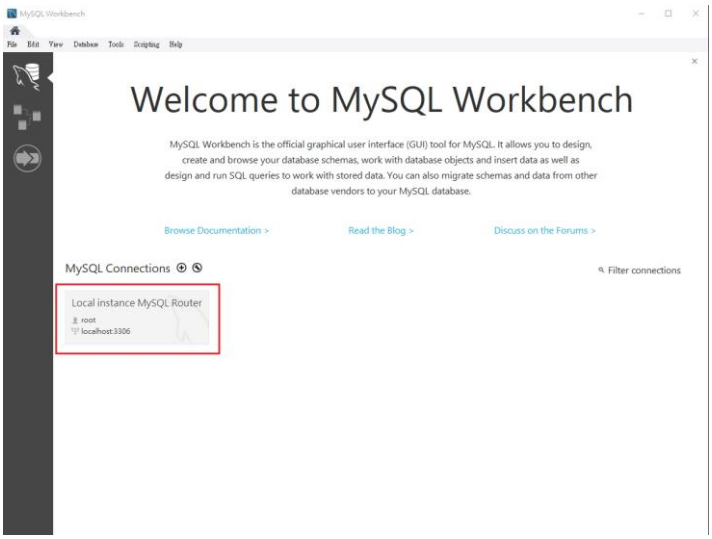
點選「Finish」就可以開始使用 MySQL



到 Windows 程式集的地方找 MySQL Workbench 可以使用圖形介面



程式首頁可以看到剛剛設定的連線帳戶，點進去就可以開始編輯



安裝完後需要將 MySQL 啟動才能正式使用，才不會出問題

- **用法**

「mysqlslap」是 Mysql 自帶的壓力測試工具，可以模擬出大量客戶端同時操作數據庫的情況，通過產出訊息來了解數據庫的性能狀態。mysqlslap 的一個主要工作場景就是對數據庫伺服器做基準測試。例如：我們拿到了一台伺服器，準備做為數據庫伺服器，那麼這台伺服器的硬體資源能夠支持多大的訪問壓力呢？最佳化操作系統的內核參數後，性能是否提升？調整 Mysql 的參數配置後，對性能有多少影響？.....

通過一系列的最佳化工作，配合基準測試，就可以把這台伺服器調整到最佳狀態，也掌握了健康狀態下的性能指標。以後在實際運行過程中，當監控的數據接近了基準指標時，說明數據庫伺服器快要過載了，需要分析數據庫結構設計、SQL 指令表示法問題，還是硬體資源不夠需要擴充，然後進行相對應的處理。數據庫伺服器也可能需要硬體升級，升級之後也需要進行基準測試，和之前的測試結果對比，確保升級後的性能是提升的，防止不恰當的升級或者錯誤的配置造成性能下降

了解 mysqlslap 的用處，下面看一下如何使用 mysqlslap。

安裝完 MySQL 後裡面有一個預設的 world database，如果沒有可以參考以下連結匯入。
[MySQL :: Setting Up the world Database :: 2 Installation]

在 Windows 作業系統下使用命令提示字元

1. 簡單用法(單線程)

mysqlslap --auto-generate-sql -uroot -p2386

```
C:\Users\user
λ  mysqlslap --auto-generate-sql -uroot -p2386
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Average number of seconds to run all queries: 0.469 seconds
        Minimum number of seconds to run all queries: 0.469 seconds
        Maximum number of seconds to run all queries: 0.469 seconds
        Number of clients running queries: 1
        Average number of queries per client: 0
```

--auto-generate-sql：作用是自動生成測試 SQL。

-uroot ：user account, (-uXXXX)

-p2386：password, (-pXXXX)

Average number of …　：運行所有語句的平均秒數

Minimum number of …：運行所有語句的最小秒數

Maximum number of …：運行所有語句的最大秒數

Number of clients …　：客戶端數量

Average number of queries per client：每個客戶端運行查詢的平均數

2. 添加並發(多線程)

mysqlslap --concurrency=100 --number-of-queries=1000 --auto-generate-sql -uroot -p2386

```
C:\Users\user
λ  mysqlslap --concurrency=100 --number-of-queries=1000 --auto-generate-sql -uroot -p2386
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Average number of seconds to run all queries: 19.937 seconds
        Minimum number of seconds to run all queries: 19.937 seconds
        Maximum number of seconds to run all queries: 19.937 seconds
        Number of clients running queries: 100
        Average number of queries per client: 10
```

--concurrency=100 ：指定同時有 100 個客戶端連接

--number-of-queries=1000：指定總共的測試查詢次數（並發客戶端數 * 每個客戶端的查詢次數）

3. 自動生成複雜表

自動測試時，創建的表結構非常簡單，只有兩列，實際的產品環境肯定會更複雜，可以使用參數指定列的數量和類型。

mysqlslap --concurrency=50 --number-int-cols=5 --number-char-cols=20 --auto-generate-sql -uroot -p2386

```
C:\Users\user
λ  mysqlslap --concurrency=50 --number-int-cols=5 --number-char-cols=20 --auto-generate-sq
l -uroot -p2386
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Average number of seconds to run all queries: 10.094 seconds
        Minimum number of seconds to run all queries: 10.094 seconds
        Maximum number of seconds to run all queries: 10.094 seconds
        Number of clients running queries: 50
        Average number of queries per client: 0
```

--number-int-cols=5　　：指定生成 5 個 int 類型的列

--number-char-cols=20　：指定生成 20 個 char 類型的列

4. 使用自己的測試庫和測試語句

自動測試可以幫助我們了解硬體層面的狀況，對於產品特定的情況，還是使用自己的資料庫來測試比較好，可以復制一份產品資料庫過來，然後對此資料庫做測試。

mysqlslap --concurrency=50 --number-of-queries=1000 --create-schema=world --query="SELECT * FROM country;" -uroot -p2386

```
C:\Users\user
λ  mysqlslap --concurrency=50 --number-of-queries=1000 --create-schema=world --query="SELE
CT * FROM country;" -uroot -p2386
mysqlslap: [Warning] Using a password on the command line interface can be insecure.
Benchmark
        Average number of seconds to run all queries: 1.203 seconds
        Minimum number of seconds to run all queries: 1.203 seconds
        Maximum number of seconds to run all queries: 1.203 seconds
        Number of clients running queries: 50
        Average number of queries per client: 20
```

--create-schema=world：用來指定測試庫名稱

--query="SELECT * FROM country;"：是自定義的測試語句

實際使用測試多個複雜的語句，可以定義一個腳本文件。把多個查詢語句寫入了一個 sql 文件，然後使用此文件執行測試。

echo "SELECT * FROM world; use world; SELECT * FROM city; SELECT * FROM country; SELECT * FROM countrylanguage;" > ~/select_query.sql

mysqlslap --concurrency=50 --number-of-queries=1000 --create-schema=world --query=select_query.sql --delimiter=";" -uroot -p2386

--query=select_query.sql：指定了 sql 文件

--delimiter=";"　　：說明 sql 文件中語句間的分隔符是什麼

mysqlslap 參考：

【原创】mysqlslap 使用总结

MySQ 技術文件

MySQL 压力测试工具 mysqlslap

2. 壓力測試工具 --- sysbench

- **實驗環境：**

  Ubuntu 16.04 LTS 作業系統
  MySQL v.5.7.21

- **支援 SQL：**

  Mysql、PostgreSQL、Oracle

- **在 MySQL 安裝 sysbench：**

  sysbench 安装、使用和测试

sysbench 主要包括以下幾種測試：CPU 性能、磁盤 IO 性能、調度程序性能、內存分配及傳輸速度、POSIX 線程性能、數據庫性能(OLTP 基準測試)。sysbench 作為業界流行的測試工具，絕大多數 DBA 都熟悉它。MySQL 等幾大主流廠商 Oracle、Percona 等在釋出效能資料時，都採用 sysbench 作為測試工具。使用相同的測試工具，更便於復現測試結果。sysbench 目前已支援 MySQL 8.0 的測試，因此從長遠來看，該工具將會持續活躍。sysbench 內嵌了 Lua 指令碼。在不需要修改核心的 C 語言程式碼的情況下，通過增添或者修改 Lua 指令碼，即可擴充套件新的測試場景，大大提高了 DBA 人員對 sysbench 的掌控能力。

1. 內建測試 1：File IO 測試

1.1 準備測試數據集，使用 prepare 命令

```
sysbench --test=fileio --file-total-size=5G prepare
```

--test=fileio　　表示測試類型為內建的 fileio 測試

--test-total-size=5G　　表示準備測試數據集為 50G 大小

prepare　　命令準備測試數據集

```
root@user:~# sysbench --test=fileio --file-total-size=5G prepare
WARNING: the --test option is deprecated. You can pass a script name or path on
the command line without any options.
sysbench 1.0.14 (using bundled LuaJIT 2.1.0-beta2)

128 files, 40960Kb each, 5120Mb total
Creating files for the test...
Extra file open flags: (none)
Reusing existing file test_file.0
Reusing existing file test_file.1
Reusing existing file test_file.2
```

1.2 使用 help 命令，查看 fileio 測試的幫助文檔：

```
sysbench --test=fileio help
```

1.3 作混合隨機讀寫測試：

```
sysbench --test=fileio --file-test-mode=rndrw --file-total-size=5G --file-rw-ratio=2 run
```

--test-fileio：測試類型為 IO 測試

--file-total-size=5G：測試文件總大小為 5G

--file-test-mode=rndrw：文件 IO 測試模式為隨機混合讀寫方式

--file-rw-ratio=2：讀寫次數比率為 2

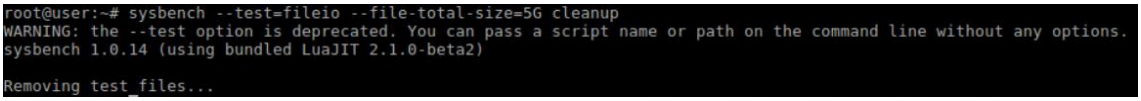
1.4 分析打印結果

```
sysbench 1.0.14 (using bundled LuaJIT 2.1.0-beta2)

Running the test with following options:
Number of threads: 1     // 測試線程數
Initializing random number generator from current time

// 測試屬性設定，可以通過選項控制
Extra file open flags: (none)
128 files, 40MiB each
5GiB total file size
Block size 16KiB
Number of IO requests: 0
Read/Write ratio for combined random IO test: 2.00
Periodic FSYNC enabled, calling fsync() each 100 requests.
Calling fsync() at the end of test, Enabled.
Using synchronous I/O mode
Doing random r/w test
Initializing worker threads...

Threads started!
// 測試過程中文件操作情況

File operations:
    reads/s:                      99.83 // 每秒鐘讀請求數
    writes/s:                     49.87 // 每秒鐘寫請求數
    fsyncs/s:                     179.08 // 每秒鐘同步次數

Throughput:
    read, MiB/s:                  1.56 // 讀， 1.56MB 每秒
    written, MiB/s:               0.78 // 寫， 0.78MB 每秒

General statistics:
    total time:                          10.0002s // 測試總時間長
    total number of events:              3290     // 事件數量

Latency (ms):
         min:                                    0.01 // 最短
         avg:                                    3.03 // 平均
         max:                                   62.49 // 最長
         95th percentile:                       11.87 // 95%以上響應時間
         sum:                                 9984.83

Threads fairness: // 線程公平性統計信息
    events (avg/stddev):           3290.0000/0.00
    execution time (avg/stddev):   9.9848/0.00
```

與文件 IO 性能評價密切相關的指標有：每秒鐘請求數、吞吐量、95%以上事件響應時間。

1.5 清除測試數據集

sysbench --test=fileio --file-total-size=5G cleanup



參考：

sysbench 基准测试工具使用

2. OLTP 的測試流程

prepare(準備資料) -> run(執行測試) -> cleanup(清理資料)

參考：

sysbench 基准测试（2）——oltp.lua 测试

sysbench-系统、数据库压力测试工具

sysbench 參考：

sysbench 在美團點評中的應用

sysbench 的一點整理

sysbench 壓力測試工具的安裝和使用

用 sysbench 實測運算讀寫圖形報表呈現主機效能

sysbench 基准测试工具使用

MySQL压力测试经验

如何避免生产环境性能瓶颈

个人介绍

- 叶金荣
  - Oracle ACE
  - 搜狐畅游福州(www.17173.com)
  - 系统部经理
  - imysql@gmail.com
  - Weibo: @yejinrong

为什么要压力测试

- 采购新设备，评估新设备性能
- 开发新项目，评估数据库容量
- 新系统上线前，预估/模拟数据库负载
- 更换数据库版本，评估性能变化

关注指标

- CPU

  %wait，%user，%sys

- 内存

  只内存读写，不产生swap

- IO

  IOPS、iowait、svctm、%util

- 数据库

  TPS/TpmC、吞吐量（QPS）、响应时长、InnoDB指标

影响因素

- 硬件
  - CPU（省电模式、超线程、多核）
  - 内存（镜像模式、xen内核限制可用总内存大小）
  - 阵列卡(BBU、CACHE、条带、读写策略、FW)
  - 硬盘（SSD/SAS）
- 系统
  - 内核参数（tcp相关）
  - 文件系统
  - IO调度器

影响因素

- MySQL
  - TRANSACTION ISOLATION LEVEL
  - Buffer pool
  - Concurrency thread
  - Redo log
  - Binlog sync
  - innodb_flush_log_at_trx_commit
  - …

注意事项

- 只在本地加压
- 压测数据量小
- 压测时间过短
- 压测模式太少
- 压力负载过大或过小
- 每轮测试完毕要净化环境

测试环境

指标	测试环境A	测试环境B
机型	DELL PE R720（2U PC Server）	DELL PE R710（2U PC Server）
CPU	Xeon E5-2620(6核，12线程，2.0GHz, L3 15MB) * 2	Xeon E5620(4核，8线程，2.4GHz，L3 12MB) * 2
内存	32G(4G * 8)	32G(4G * 8)
阵列卡及设置	PERC H710，512MB， BBU(FW：12.10.1-0001)， RAID 1+0 FORCE WB	PERC H700，512MB， BBU(FW：12.10.1-0001)， RAID 1+0 FORCE WB
硬盘	15K RPM 300G SAS * 8	15K RPM 300G SAS * 6
网卡	Intel 1GbE	Broadcom 1GbE
操作系统	RHEL 6.4	RHEL 6.4
文件系统	xfs/ext4	xfs/ext4
MySQL版本	5.5.34	5.6.14

测试工具

- sysbench
  - Primarily for MySQL OLTP benchmarking，By MySQL AB
  - cpu、threads、mutex、memory、fileio、oltp
- tpcc-mysql
  - Primarily for MySQL OLTP benchmarking，By Percona
- tpch
  - Primarily for OLAP benchmarking
- tcpcopy
  - 模拟生产环境真实请求
- 其他
  - mysqlslap
  - sql-bench

测试工具 – sysbench

- 项目地址
  - http://sysbench.sourceforge.net/
- 分支版本
  - https://code.launchpad.net/sysbench
  - 特点：支持lua，增加选项 oltp-tables-count

测试工具 – sysbench

- 安装
  - ./configure --with-mysql-includes=path --with-mysql-libs=path && make && make install
- 支持其他数据库
  - with-pgsql
  - with-oracle
- 运行
  - sysbench --test=[mode] [other_options] prepare
  - sysbench --test=[mode] [other_options] run
  - sysbench --test=[mode] [other_options] cleanup

测试工具 – sysbench

- 通用基准
  - 最大请求数：5,000,000
  - 并发线程数：8 ~ 512
- 基准 – OLTP
  - mode=complex
  - engine=innodb
  - oltp-table-size=100,000,000

测试工具 – sysbench

- 测试用例

```
sysbench --test=oltp --mysql-host=host \
--mysql-user=user --mysql-password=passwd \
--oltp-test-mode=complex --mysql-table-engine=innodb \
--oltp-table-size=100000000 --mysql-db=db \
--oltp-table-name=test_tbl --num-threads=128 \
--max-requests=5000000 run
```

--oltp-test-mode 测试模式：complex/simple/nontrx

测试工具 – sysbench

- 测试结果

```
OLTP test statistics:
    queries performed:
        read:                            70000
        write:                           25000
        other:                           10000
        total:                           105000
    transactions:                        5000   (2001.82 per sec.)
    deadlocks:                           0      (0.00 per sec.)
    read/write requests:                 95000  (38034.59 per sec.)
    other operations:                    10000  (4003.64 per sec.)

Test execution summary:
    total time:                          2.4977s
    total number of events:              5000
    total time taken by event execution: 79.6005
    per-request statistics:
         min:                            0.0051s
         avg:                            0.0159s
         max:                            0.0795s
         approx.  95 percentile:         0.0250s

Threads fairness:
    events (avg/stddev):           156.2500/3.72
    execution time (avg/stddev):   2.4875/0.00
```

测试工具 – sysbench

- 不足
  - 只能模拟简单OLTP
  - 测试表列数少
  - 测试表数据类型少
  - 标准版本只能支持一个测试表

测试工具 – tpcc mysql

- 安装
  - 下载 bzr branch lp:~percona-dev/perconatools/tpcc-mysql
  - 直接make即可
  - create_table.sql - 创建数据表
  - add_fkey_idx.sql – 创建索引及外键
- 初始化加载数据
  - tpcc_load db_host db_name db_user db_passwd db_warehouse_num
  - 例如：tpcc_load localhost tpcc1000 user passwd 1000
- 运行OLTP测试
  - ./tpcc_start -h localhost -d tpcc1000 -u root -p ' xx' -w 1000 -c 32 -r 120 -l 3600 -f ./tpcc_mysql_20120314

测试工具 – tpcc mysql

- 基准 - OLTP
  - warehouse = 1000
  - max connection = 8 ~ 512
  - warm up = 120(s)
  - run time/duration = 3600(s)

测试工具 – tpcc mysql

- 测试用例

./tpcc_start -h host -d db -u user -p passwd \
-w 500 -c 128 -r 120 -l 3600

-w warehose/仓库数
-c concurrent threads/并发线程
-r warmup/数据预热时长
-l during time/持续加压时长

测试工具 – tpcc mysql

- 测试结果

```
<Constraint Check> (all must be [OK])
 [transaction percentage]
        Payment: 43.48% (>=43.0%) [OK]
   Order-Status: 4.35% (>= 4.0%) [OK]
       Delivery: 4.35% (>= 4.0%) [OK]
    Stock-Level: 4.35% (>= 4.0%) [OK]
 [response time (at least 90% passed)]
      New-Order: 100.00%  [OK]
        Payment: 100.00%  [OK]
   Order-Status: 100.00%  [OK]
       Delivery: 100.00%  [OK]
    Stock-Level: 100.00%  [OK]

<TpmC>
                    34035.332 TpmC
```

测试工具 – tpcc mysql

- 不足
  - 测试表模式设计未优化
  - 存在外键
  - 部分索引不合理

测试工具 – tpch

- 安装
  - 下载 http://www.tpc.org/tpch/spec/tpch_2_16_0.zip
  - cp makefile.suite makefile
  - 编辑 makefile 文件

```
CC = gcc
DATABASE = MYSQL
MACHINE = LINUX
WORKLOAD = TPCH
```

  - 编辑tpcd.h，增加宏定义

```
#ifdef MYSQL
#define GEN_QUERY_PLAN ""
#define START_TRAN "START TRANSACTION"
#define END_TRAN "COMMIT"
#define SET_OUTPUT ""
#define SET_ROWCOUNT "limit %d;\n"
#define SET_DBASE "use %s;\n"
#endif
```

测试工具 – tpch

- 初始化
  - 初始化测试表数据： ./dbgen -s 100
  - 生成测试数据：mysql -f tpch < dss.ddl
  - 默认的初始化模式无主键、无索引
  - LOAD DATA INFILE导入数据
  - 注意max_binlog_cache_size限制，需要切分文件导入
  - 执行修改主键/外键/额外索引脚本
  - 数据表名全部改成小写，适应TPC-H测试SQL脚本
  - 运行qgen生成测试SQL
  - 修改部分SQL语句
  - 拆分完成测试SQL脚本成23个测试SQL
- 运行
  - 执行23个测试脚本，记录运行时长

测试工具 – tpch

- 基准 - tpch
  - warehouse = 1000
  - 单进程
  - tpch侧重OLAP模型，而MySQL并不适合OLAP，因此warehouse设定较小

测试工具 – tpch

- 测试用例

生成测试数据：./dbgen -s 1000

执行OLAP查询：time mysql -f tpch < ./queries/tpch_${NN}.sql

-s warehose/仓库数

测试工具 – tpch

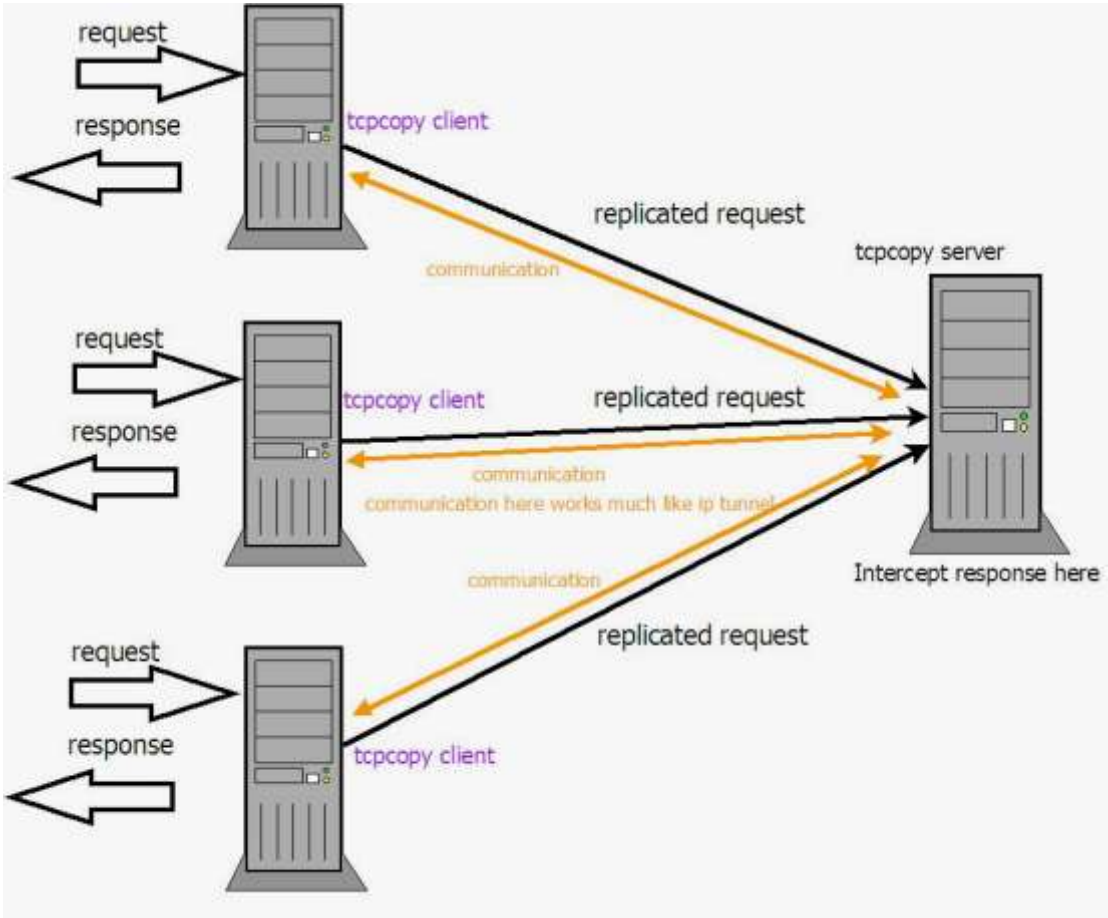
- 不足
  - 初始化比较麻烦
  - 部分索引不合理
  - MySQL本身不擅长做OLAP，测试模式有局限

测试工具 – tcpcopy

- 安装
  - 项目地址：https://github.com/wangbin579/tcpcopy
  - 下载，执行 sh autogen.sh 后直接编译安装即可
  - 按照文档配置server和client端，即可启动压测

测试工具 – tcpcopy

- 测试用例



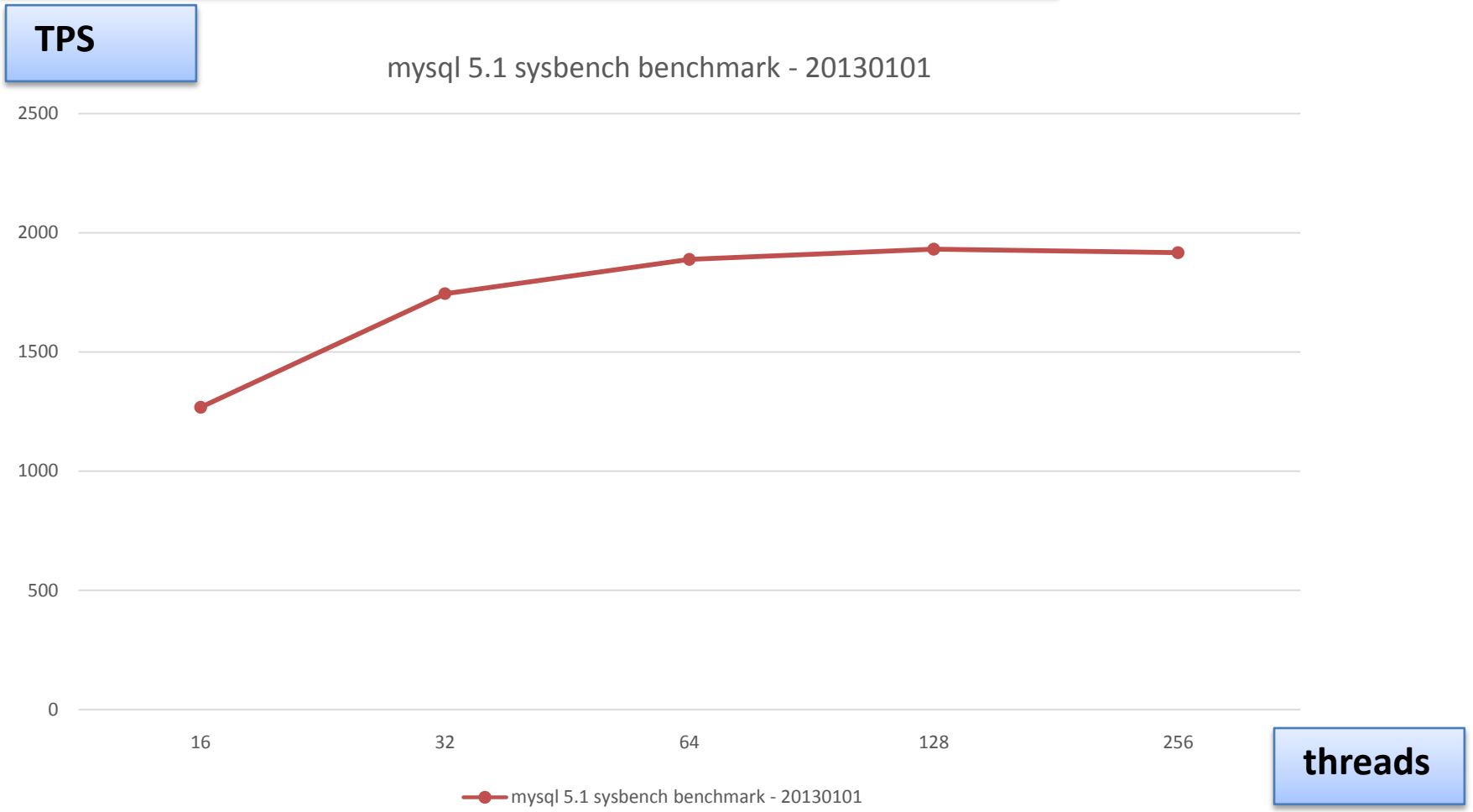
测试工具 – tcpcopy

- 不足
  - 测试MySQL时，测试机需要开启 skip-grant-tables，否则无法正常进行，因为MySQL需要进行认证；在线服务器无需调整
  - 测试过程中不能执行flush privileges，否则上述选项会失效
  - 每次启用tcpcopy时，都需要重启mysqld，不能在线直接应用，否则不能转发包
  - 不支持prepare语义
  - 可能需要调整部分内核tcp参数，避免出现queue dropped
  - 无法100%保证生产环境与测试环境数据一致性

测试报告 – sysbench

线程数	第一轮(tps)	第二轮(tps)	第三轮(tps)	均值(tps)
16	1266.867	1267.683	1267.275	1267.275
32	1744.833	1744.350	1744.592	1744.592
64	1891.250	1884.600	1887.925	1887.925
128	1938.483	1923.983	1931.233	1931.233
256	1920.350	1913.017	1916.684	1916.684

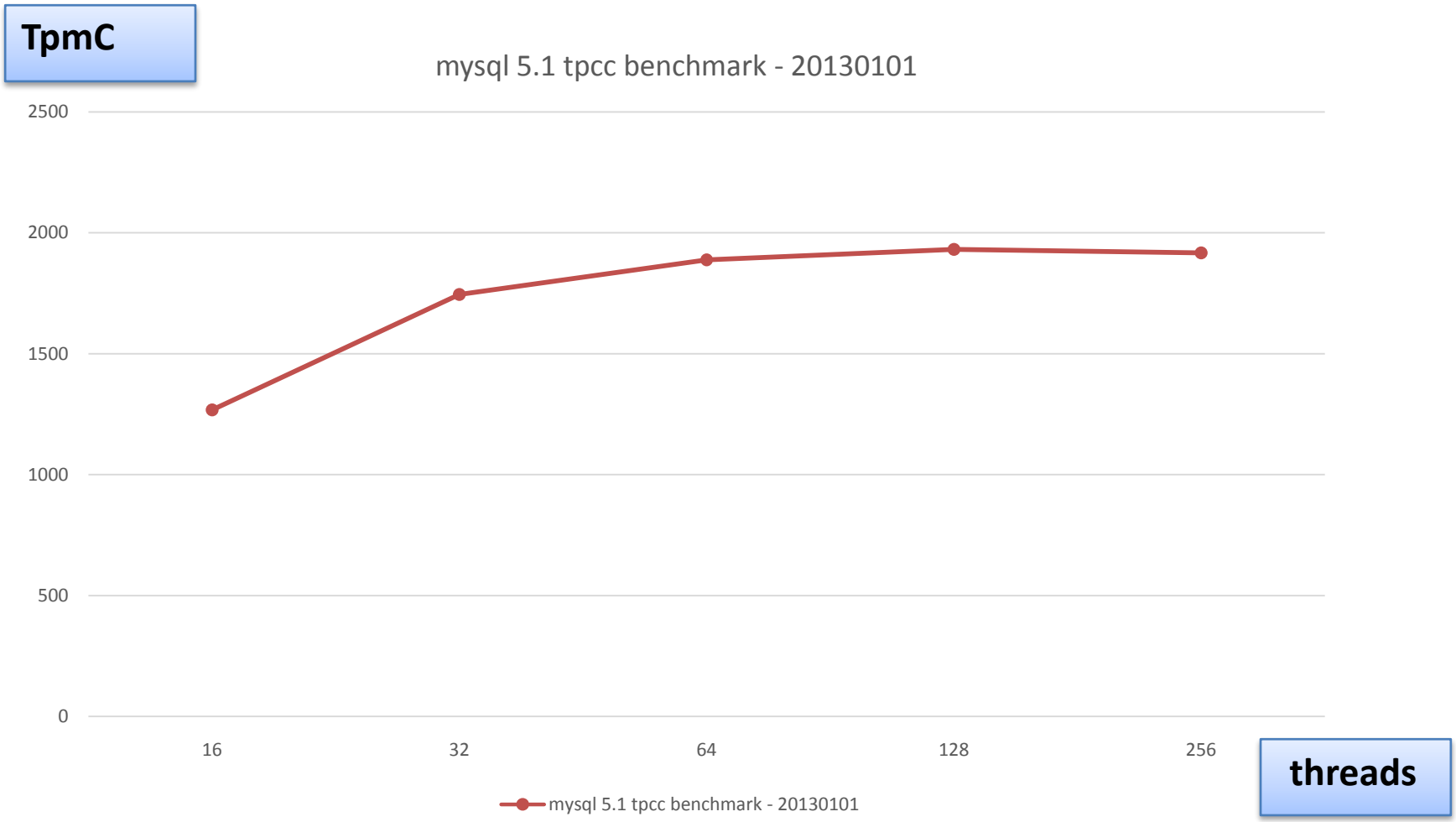
测试报告 – sysbench



测试报告 – tpcc mysql

线程数	第一轮(TpmC)	第二轮(TpmC)	第三轮(TpmC)	均值(TpmC)
16	1266.867	1267.683	1267.275	1267.275
32	1744.833	1744.350	1744.592	1744.592
64	1891.250	1884.600	1887.925	1887.925
128	1938.483	1923.983	1931.233	1931.233
256	1920.350	1913.017	1916.684	1916.684

测试报告 – tpcc mysql

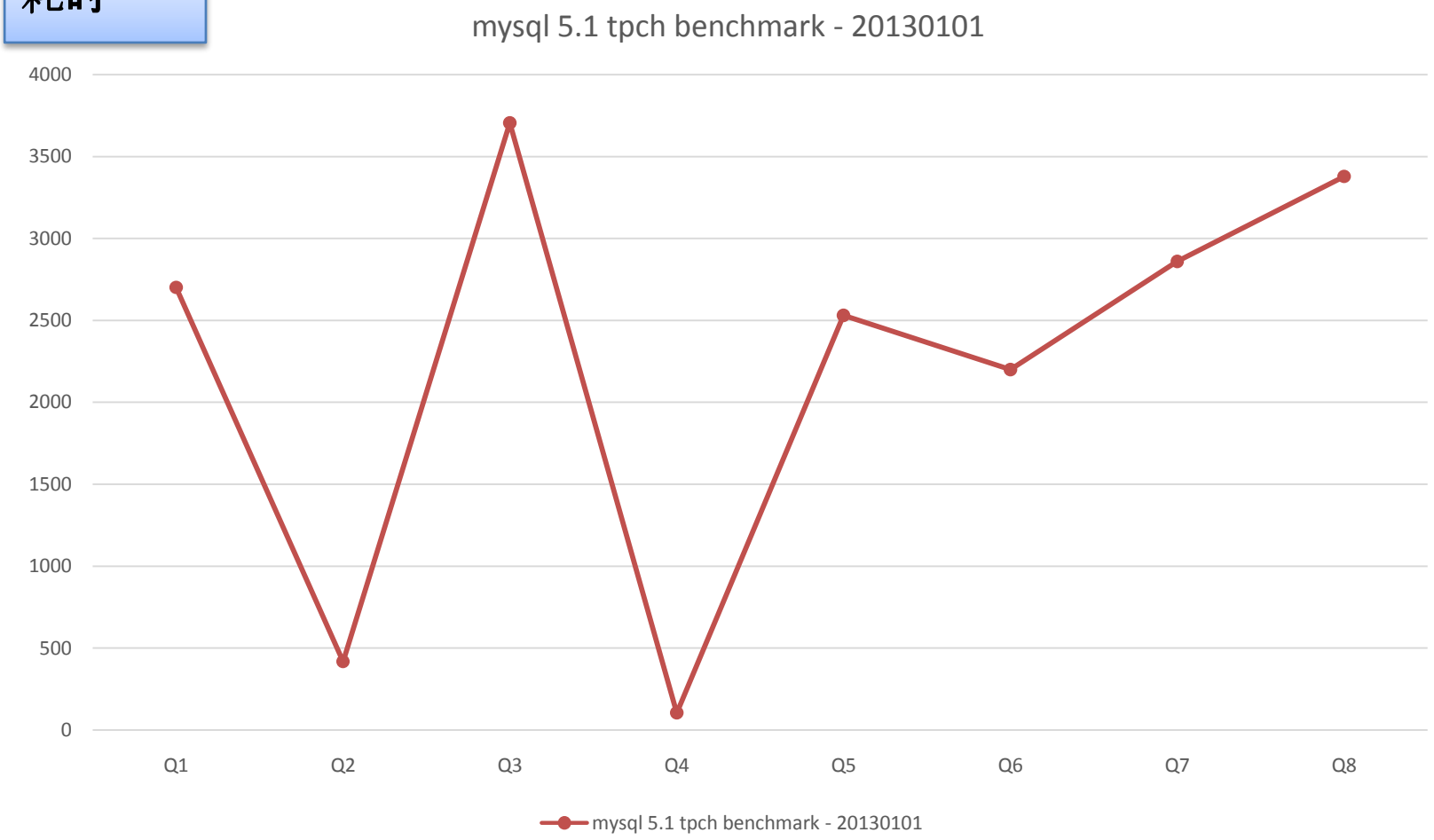


测试报告 – tpch

查询	第一轮	第二轮	第三轮	均值(s)
Q1	2701.215	2697.523	2704.187	2700.975
Q2	416.314	418.774	417.474	417.521
Q3	3667.766	3755.572	3692.900	3705.413
Q4	105.804	105.493	102.411	104.569
Q5	2525.357	2536.164	2530.909	2530.810
Q6	2202.317	2194.628	2201.490	2199.478
Q7	2829.224	2860.434	2890.436	2860.031
Q8	3335.373	3383.118	3417.130	3378.540

测试报告 – tpch

耗时


mysql 5.1 tpch benchmark - 20130101
4000
3500
3000
2500
2000
1500
1000
500
0
Q1
Q2
Q3
Q4
Q5
Q6
Q7
Q8
mysql 5.1 tpch benchmark - 20130101

可靠性测试

- 模拟意外事件
  - 断电(硬件冷重启)
  - RESET(硬件热重启)
  - 阵列卡掉线
  - 磁盘掉线
  - REBOOT(系统重启)
  - 正常关闭服务(kill -TERM)
  - 异常关闭服务(kill -9)
  - 磁盘空间满
  - 删除文件
  - 破坏性修改已打开文件
  - …

可靠性测试

- 长期极限高压
  - 持续数小时、数天、数周运行高负载计算、IO任务
  - 考验服务器在高压下的性能波动情况
  - 考验硬件设备在高压下的稳定性表现

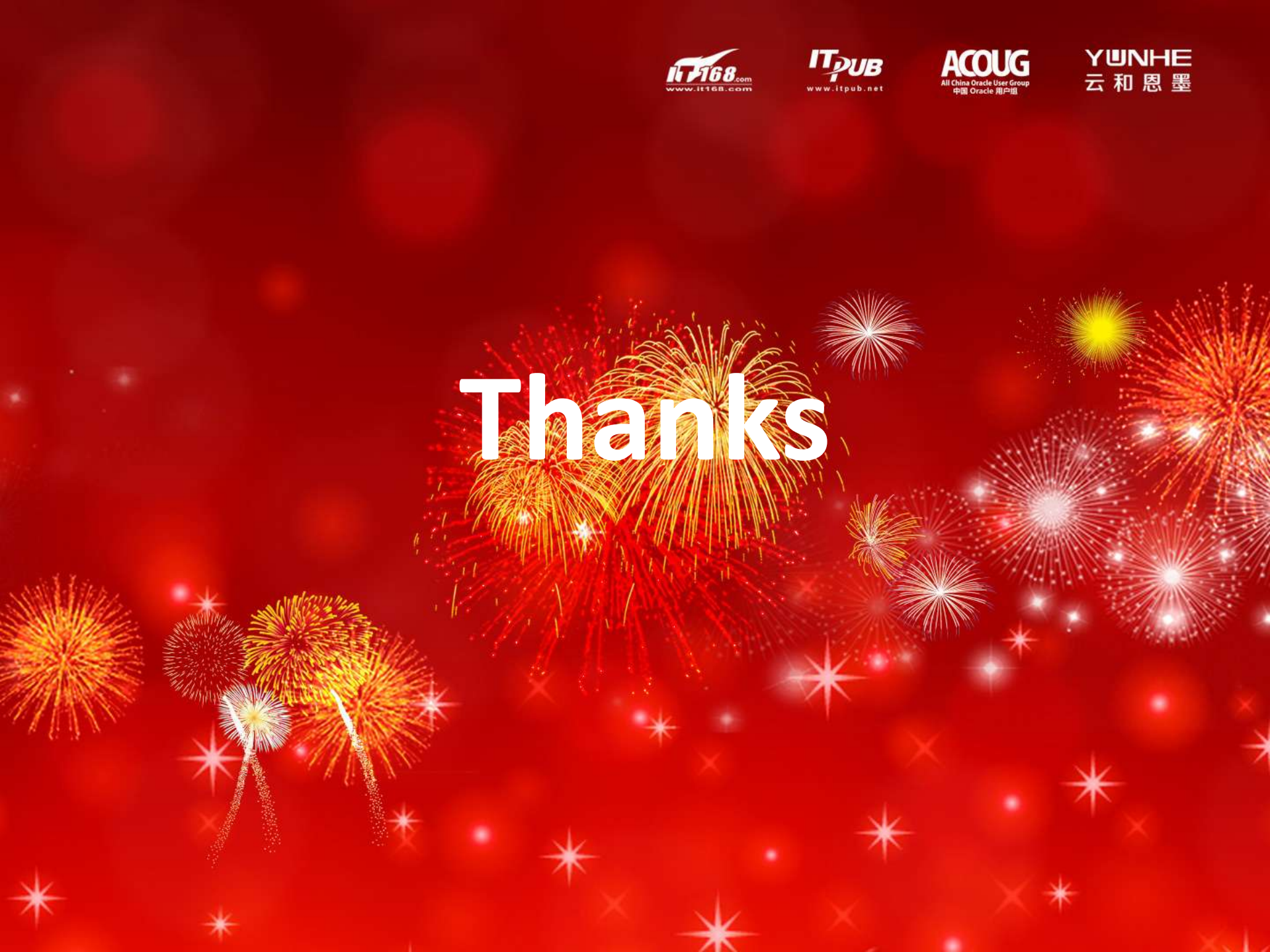
- 模拟恶劣环境
  - 供电不稳
  - 通风冷却不好
  - 湿气大、灰尘多

参考

- http://baike.baidu.com/view/2776305.htm
- https://code.google.com/p/tcpcopy
- http://imysql.com/2012/12/21/tpch-for-mysql-manual.html
- http://imysql.com/2012/08/04/tpcc-for-mysql-manual.html
- http://imysql.com/node/312
- http://imysql.com/2009/04/24/using_mysqlslap_for_load_testing.html
- http://imysql.com/2008_07_25_innodb_vs_pbxt
- http://imysql.com/2011/10/22/using-tcpcopy-for-request-replication-or-testing.html

附件

- 整合sysbench测试脚本 [下载]
- 整合tpcc-mysql测试脚本 [下载]
- 整合tpch测试脚本 [下载]
- 汇总下载 [下载]
- 测试结果表格模板 [下载]


IT168.com
www.it168.com
ITPUB
www.itpub.net
ACOUG
All China Oracle User Group
中国 Oracle 用户组
YUNHE
云和恩墨
Thanks