東海大學

資訊工程研究所

碩士論文

指導教授: 楊朝棟博士

在 Docker 上以 MongoDB 實作雲端半導體測試資料庫系統

The Implementation of Semiconductor Test Information Cloud Database System with MongoDB on Docker

研究生: 謝祥議

中華民國一〇四年六月

東海大學碩士學位論文考試審定書

東海大學資訊工程學系 研究所

研究生 謝 祥 議 所提之論文

在 Docker 上以 MongoDB 實作雲端半導體測試資料庫系統

經本委員會審查，符合碩士學位論文標準。

學位考試委員會

召 集 人 楊武 簽章

委 員 劉榮春

許慶賢

張玉山

指 導 教 授 楊朝棟 簽章

中華民國 104 年 7 月 7 日

摘　要

半導體測試資料 STDF（Standards Test Data Format）檔案中的 Data Log 資料，無法如同一般半導體測試結果的等級別（Bin）資料，將它儲存進關聯式資料庫（RDBMS）後進行分析，原因為單一個測試資料 STDF 檔案之中，包含的測試資料就有數百萬筆，半導體封裝測試工廠進行半導體晶圓或者積體電路的測試後，每日獲得之 STDF 檔案超過數千個。如果建立傳統資料庫進行儲存將耗費大量成本且不彈性。本研究利用 NoSQL Document-Oriented（MongoDB）資料庫，建構一個儲存半導體測試資料雲端資料系統，這是個將半導體測試資料 STDF 轉為 JSON 格式存入私有雲中的資料庫，並利用 Docker Container 快速彈性的方式，進行佈署所需要的 NoSQL 資料庫。讓半導體測試工程師可以方便連結到 NoSQL 資料庫查詢測試結果，且公司可以在因應需要投入相對應的軟硬體資源。系統驗證部分則使用 Yahoo Cloud Serving Benchmark (YCSB) 對 MongoDB 進行壓力測試驗證。

關鍵字：私有雲、NoSQL、MongoDB、Docker 容器

Abstract

The testing of the data log file in the STDF archive containing the current semiconductor test information cannot complete as Bin (levels) of chips stored in the traditional relational database for analysis, the reason being that a single STDF archive with test data contains test data with millions of record, and the STDF files acquired through semiconductor tests of chips and integrated circuits executed every day in the factory amount to several thousands. The building of a traditional database for storage is very costly and in no way elastic. This research uses the NoSQL Document-Oriented (MongoDB) database to construct a cloud data system for semiconductor chip test data, and transforms STDF into JSON format stored in a private cloud, uses Docker Container flexible and fast approach to rapid deployment of NoSQL database, with which it enables the semiconductor test engineers to make fast connections with semiconductor materials querying databases, and the company can invest in corresponding hardware and software resources. This is used to resolve the company's problem of not being able to store the semiconductor test data log in a traditional relational database. The system is used to verify parts of Yahoo Cloud Serving Benchmark (YCSB) for stress testing to verify MongoDB.

Keyword:Private Cloud, NoSQL, MongoDB, Docker Container

致謝詞

對我來說，獲得碩士學位一直是我夢想中想要完成的一個目標。能順利拿到學位對我來說是令我無比開心的事情。在白天要寫程式及開會，半夜要 oncall 的忙碌的 CIM 工作環境中，週末以及晚上還要抽出時間去上課真的對自己的一大考驗，時常是蠟燭多頭燒的狀況，對我來說一切的考驗，都是為了完成自己多年來的夢想而必經的歷程。

十年來都專注於半導體封測產業當中，接觸的都是關於電腦製成整合職務所必須要的程式設計，然而到了研究所接觸到的雲端運算可真是讓我大開眼界，讓多年後再度踏入校園的我真的感覺到不虛此行。

在求學期間，非常非常感謝我的指導老師楊朝棟教授，無私的指導我這老學生，教導我雲端運算的相關領域知識，這些知識讓我剛好可以運用到工作之中，不會說只是來拿個文憑而已。在這個大數據的時代，因為來東海學習這些，讓我帶回去跟主管們分享，真的對工作產生了幫助，現在公司已經開始大量使用虛擬機器來整合利用公司資源。

另外就是同學們，除了無私分享知識與技術外，還有一起學習成長的共患難感情，讓我深深感受到可以來念東海資工，我真的是超級幸運的。

Table of Contents

List of Figures

List of Tables

Chapter 1

簡介

1.1 研究動機

雲端運算是現今十分熱門的議題，不管是軟體或是硬體廠商均開始將產品逐漸移轉到雲端，發展雲端相關產品。然而其中雲端儲存（Cloud storage）的部份，就是將資源或是資料放到網際網路上供人隨時隨地存取的一種模式。也就是資料存放在雲端上，使用者透過任何可上網的裝置快速容易地存取資料。現有的公有雲雲端儲存服務資料安全性是否能夠確保無虞，相對於私有雲其實還有很大一段路要走，因為資料都是交由提供雲端服務之廠商儲存，如果遇到不肖的雲端廠商或是遭到駭客攻擊，都有可能發生資料外洩的情況。但在設備與軟體的購買與維運上，會對一般中小企業來說是一項不小的負擔，畢竟設備都需要定期維護，才能確保在存取資料中不會發生異常狀況。在現今的雲端服務爆發的時代，雲端儲存的方案處處可見，但對於中小型公司要儲存一些特殊的資料數據的雲端儲存方案就比較少。

為了讓 IT 資源得以滿足組織目標，提高資源的使用率與靈活性；跳脫過去加碼採購用以滿足企業業務需求的思維，在不增加企業支出的前提下，針對企業原有的架構來規劃，整併、汰換與升級企業的 IT 環境，在需要的時候，可利用資源使用的靈活性去動態配置所需的資源而達成 IT 資源最佳化以及改善系統及管理的效率的最重要技術之一，就是虛擬化技術。虛擬化技術的最初用意就

是為了要實現更高的設備利用率，使用戶能夠盡可能地利用系統資源。因此如果可以在單一伺服器上虛擬多個系統，就能夠以少數幾台電腦設備完成所有更多工作。也就是說以一台電腦的耗電量，等同於開啟多台電腦的耗電量，故虛擬化技術在節能省電上以及減少碳排放是效果是驚人的。

然而傳統的虛擬化技術，需要用戶先安裝系統，然後才可於系統之中安裝軟體，其實某程度上整個安裝的過程都需要花上一定時間，而且假如我只是在虛擬機之中運行單一服務的話，卻仍需先安裝整個系統（如 Windows、Ubuntu），然後進行系統的配置設定之後才可以使用。而 Docker 是一種針對應用本身的虛擬化，而非傳統虛擬化方式；可以讓你於一般的伺服器之中，可以快速動態配置多個虛擬化應用。

雖然現在有很多提供雲端服務的公司，但半導體測試的結果對於客戶是其付費後獲得的資產，必須小心加以保護，若能夠使用 Open Source 軟體來建置私有雲的雲端儲存平台，將可以有效控制與降低購買與建置費用並且保護其資料安全性，並可以根據企業本身之特殊儲存需求來作動態配置資源及規劃，來滿足企業需求。

1.2 論文目標與貢獻

積體電路的製造，是要將設計的電路布局，經由反覆地曝光、顯影、離子植入、蝕刻等幾百道迴流的複雜製造程序，和超過一個月以上的生產週期時間，直到把多達三十層以上的每一層電路，都準確成形於一片片如圓餅般的薄片晶圓上，最後經過後段的封裝測試而成為一顆顆晶片（Chips）[26, 27]。其中半導體封裝測試，是半導體製造最後一段關卡。測試工程師像老師一樣出問題 (測試項目) 給晶圓或者晶片進行測試，各種題目都會打出分數來，並且記錄到測試結果檔案中，獲得分數的高低將直接影響該晶片的價值 (依其電性功能作分類，作為 IC 不同等級產品的評價依據)。

現在晶片的功能非常強大，所以它要進行驗證的項目非常多從數百乃至千個，而一片晶圓上面可能有數萬個晶片，每個晶圓每經過一次測試，將產生一

個測試資料檔案 STDF（Standard Test Data Format）。STDF 是由 Teradyne 公司已經開發出來一種簡單、靈活的測試資料格式文件，裡面可以儲存數百萬筆測試項目的 Data Log。大數據對半導體廠封裝測試廠而言，必須先具有能隨時儲存大量測試資料的能力 [13]，其次更要能在短時間從這些龐大資料中找出所要查詢條件的資料，然後才能拿來分析應用。目前半導體廠大多採用傳統的關聯式資料庫來分析儲存資料，因為受到容量限制，如果資料量一多，就得先將資料切成好幾段分開運算，需要資料進行分析，只能片段或者看單一的 STDF 檔案來進行分析。

隨著網際網路及雲端運算的發展，現今企業將應用程式及資料存放在資料中心，可動態提供服務及共用運算，使用者可以隨時隨地透過各式連線裝置存取資料。STDF 本身是有其特殊架構的文件，如果可以將他完整存入資料，對於後續工程師去分析使用將會有很大幫助。半導體封測的低毛利，讓如何以現有電腦主機以及不花費其他軟體費用，來建立提供這個服務變成了一個我們實驗必須面對的限制。因此 NoSQL 的可擴充性以及高效能可提供巨量資料需要的儲存規模和分析能力。在硬體系統建置方面，以工廠內舊有主機為主要硬體設備，畢竟半導體封測一向是低毛利的產業，對於控制預算方面十分嚴格，故本實驗有二個實驗限制:

1、沿用原工廠電腦主機：主要是不增加公司購買硬體負擔，若確實需要時候再行購置。

2、使用 Open Source 軟體建構平台：使用 Open Source 即可不增加軟體購買費用，舉例來說 Oracle 1 CPU 一年授權費用大約要花費新台幣一百萬。

本研究使用 MongoDB 與 Docker 來建立半導體測試資料儲存雲端資料系統，MongoDB 系統是 MongoDB, Inc 這家公司用 C++ 開發的一個 Open Source NoSQL [19] 分散式文件 Database。MongoDB 它提供了高效能 (high performance)、高可用度 (high availability) 以及自動擴充 (automatic scaling) 等特色。故研究進行利用 MongoDB 來做半導體測試資料雲端庫。過去虛擬化技術是建立一個可以執行整套作業系統的沙箱獨立執行環境，一般我們稱呼為虛擬機器（Visual Machine）。然而 Container 技術提供了一種以應用程式為中心

的虛擬化技術，Container 技術將應用程式所需的相關程式碼、環境設定都包裹在一個 Docker File 後，接下來可依需要於數分鐘就可以建立好一個沙箱執行環境，可以相當彈性的去配置硬體資源。

1.3 論文架構

本論文主要在介紹雲端儲存系統的效能評估，並運用 Docker 虛擬化技術來建置 MongoDB2.6.9 與 MongoDB 3.0.X Wired Tiger 使用 YCSB 進行效能評估。第一章針對雲端運算及雲端儲存作介紹，並說明研究動機。第二章說明本論文的研究背景，並介紹 YCSB。第三章描述系統的建置方法及實驗方法。第四章說明實際實驗的結果，並依照實驗的結果說明兩個版本的優劣。第五章說明本論文依照實驗結果做出結論及未來的研究方向。

Chapter 2

研究背景與相關研究

2.1 研究背景

2.1.1 雲端運算

雲端運算是一種基於網際網路的運算程序，雲端就是指網路，透過網際網路連線至遠端主機，提供軟體與硬體給不同地理環境的使用者。雲端運算詞源於，2006 年 8 月 9 日，Google 執行長 Eric Schmidt 在搜尋引擎大會 (SES San Jose) 上首次提出的運算概念。

依據美國國家標準與技術研究院 (National Institute of Standards and Technology, NIST) 於 2012 年 5 月發佈 SP 800-146 建議書，定義雲端運算為「使用無所不在、便利、隨需應變的網路，共享廣大的運算資源 (如網路、伺服器、儲存、應用程式與服務)，可透過最少的管理工作及服務供應者互動，快速提供各項服務」。此模型能提供一個便利 (convenient) 且可按需求 (on-demand) 來透過網路存取與配置的共享運算資源池 (如網路、伺服器、儲存裝置、應用程式與各類服務)。這些共享運算資源池可以被迅速的提供並發佈，同時將管理成本或服務供應商協助最小化。這種雲端模型提升了服務可用性，此雲端模型由五個基本特徵、三個服務模式及四種佈署模型組成，其模型架構如圖 2-1 所示：

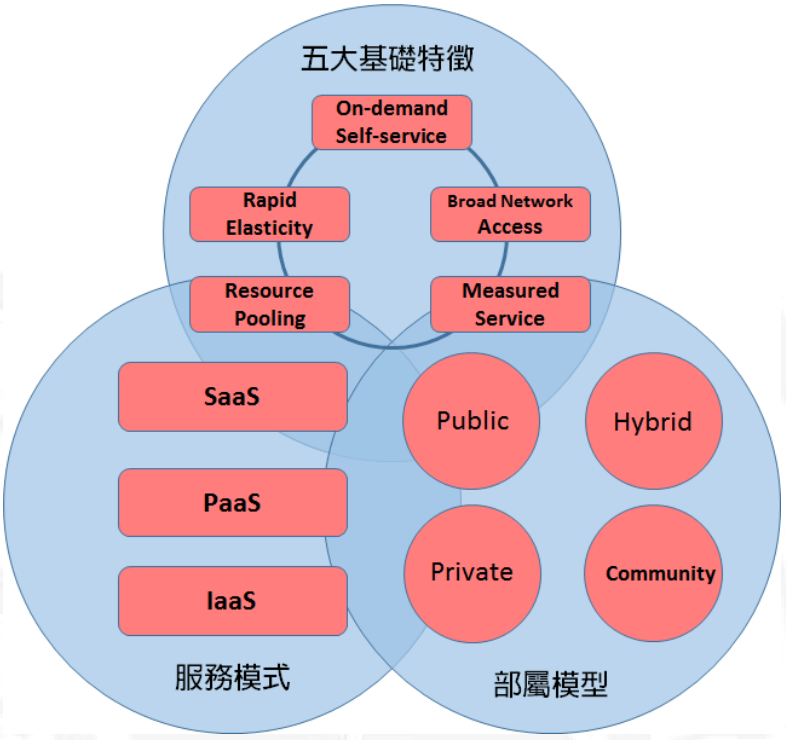

FIGURE 2.1: 美國國家標準與技術研究所 NIST －雲端運算定義

美國國家標準與技術研究所對於雲端運算定義中，雲端服務架構可依服務類型指標劃分為基礎架構、平台以及應用三大層次，分別為基礎架構即服務 (IaaS)、平台即服務 (PaaS) 以及軟體即服務 (SaaS)。

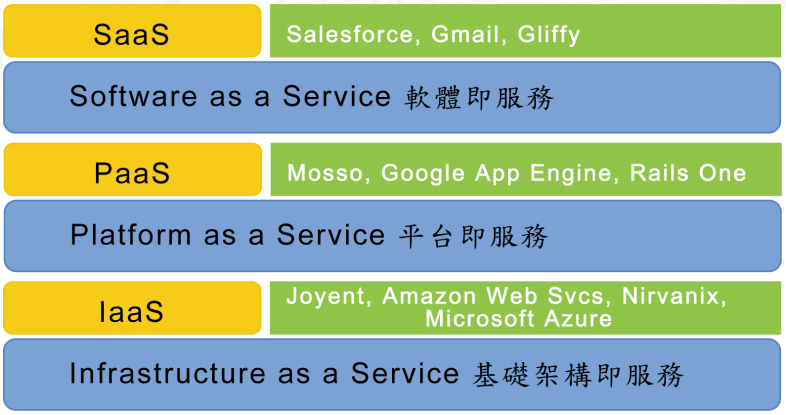

FIGURE 2.2: 雲端的三種服務模式

- Software as a Service (SaaS): 希望使用者使用應用程式，但並不接觸作業系統、硬體或運作的網路基礎架構。是一種以提供服務的觀念為基礎，軟體服務供應商，以租賃的概念提供使用者服務，而非購買，將商用軟體存

放在資料中心，以網路存取的方式提供訂閱 (Subscription) 服務或計次付費 (Pay-per-use) 的型式收費。例如：Google GMail 與 Salesforce.com。

- Platform as a Service (PaaS): 希望使用者透過使用主機操作應用程式。使用者掌控運作應用程式的環境 (也擁有主機部分掌控權)，但並不掌控作業系統、硬體或運作的網路基礎架構。平台通常是應用程式基礎架構。例如: Google App Engine 與 Microsoft Azure。其運作通常有以下兩種方式：

- Infrastructure as a Service (IaaS): 希望使用者使用「基礎設施運算資源」，如運算能力、儲存空間、網路元件或中介軟體。消費者能掌控作業系統、儲存空間、已部署的應用程式及網路元件 (如防火牆、負載平衡器等)，但並不掌控雲端基礎架構。企業用戶可以依照所需要的電腦與網路設備等資源 (CPU、OS、Storage)，隨時向服務提供廠商訂購服務，並可即時更改需求設定，並依處理器 (CPU ／ Hr)、儲存空間 (GB) 或網路流量 (Gbps) 使用量計費。例如: Amazon EC2、Microsoft Azure。

2.1.2 NoSQL

關聯式資料庫，主要是以 ISO 文件中定義的 ACID [8] 為標準，也就是說資料庫交易 (Transcation) 有以下四種特性:

- 1. Atomicity: 一個交易中的所有操作, 若執行就需全部執行完畢。
- 2. Consistency: 在操作執行前和執行後, 並不影響整體資料庫的完整性。
- 3. Isolation: 兩個交易操作互相不影響。
- 4. Durability: 而當操作結束後, 所做的更動將完整且持久地保存至資料庫中。

根據 Michael Stonebraker 的研究指出，若能減少以上其中一項, 就能有效的提高效能。在近年來雲端技術的發展下，對資料庫的要求，對於 Consistency、Isolation 的要求不再那麼高，而是轉向分散式以及快速。其中利用 CAP [6] 理論而發展的 NoSQL(Not Only SQL) 是一種解決雲端時代資料爆量的方法策略，它沒有 Schema 限制 (schemaless) 可自由添加新欄位，而 nosql-database.org 對它定義的為: 帶來非關聯式、分散式、開放原始碼、高可擴充性。NoSQL 是多種非使用 SQL 語法查詢的資料庫總稱。大致可以分成以下 4 種:

- Key-Value 資料庫，它打破以往關聯式資料庫常用的 Schema 將每筆各自獨立，且具有高度水平擴充能力地特性，能按照需求增加資料庫。常見的有 Google BigTable、Hadoop HBase、Apache Cassandra。
- 記憶體資料庫 (In-Memory Database)，它將資料庫儲存在記憶體當中來加快讀取速度。適合用在快取網頁，減少讀取硬碟的次數，常見的有 Redis、Tuple Space、Velocity。
- 文件資料庫 (Document Database)，用來儲存結構鬆散或者非結構化資料，一般來說常見的非結構化資料為 HTML 網頁，每個標籤段落內可以包含文字、圖片、音樂、影響等等。常見的有 CouchDB、MongoDB、Riak。

- 圖學資料庫 (Graph Database)，運用圖學架構來儲存結點間關係的資料結扣，包括各節點 (Node)、關係 (Relation)、屬性 (Property)。常見的有 Neo4j、InfoGrid。

過去我們花費大量的金錢購買昂貴的設備，來保持不要發生硬碟、伺服器、網路不會失效 (Failure)，但現在隨著硬體成本的不斷降低。但 NoSQL 的特性是假設失效是必然的，我們要思考的就變成當失效是必然的時候，如何去應付此一問題。我們可以利用以下手段:

- 對資料進行切割 (Partition the Data): 通過此一方式最小化了失效的影響範圍，也將讀寫的工作分配到不同節點，可以提升速度或者當資料遺失時，只是部分而不是全部。
- 對同一資料保持多個副本 (對同一資料保持多個副本): 大部分 NOSQL 實作都基於資料副本的熱備份 (hot-backup) 來保證連續的高可用性 (high availability)。
- 動態伸縮 (Dynamic Scaling): 由於資料的成長速度是呈現行方式快速成長，而 NoSQL 提供了不停機的方式來做擴展的方式。
- 對查詢的支援 (Query Support): 文件資料庫的儲存的資料是屬於 BLOB，採用的是一個關鍵字對應一個屬性列表 (List)，這樣子的好處是不須受限於 Schema 的限制，也就是你要刪除一個文檔的屬性或者砍掉整個檔案都是很方便的。

NoSQL DB 的特徵如下:

- Schema-less: 不需預先定義 schema，每一筆記錄的欄位數量與結構也可以不一樣。
- Shared nothing architecture: 通常採用儲存於本機硬碟、而非共同儲存設備（如 SAN 或 NAS)。本地硬碟的存取速度遠較透過網路傳輸快，再者 NAS 空間昂貴一般主機硬碟便宜。

- Elasticity: 只需增加更多主機，便能立即擴充儲存容量與負載能力，並且利用建立虛擬機器來彈性增加調配資源。
- Sharding: 不將儲存視為龐大的空間，取而代之的是以「分片」(shard) 方式來分割資料集。分片可在主機間 (或者 Container) 進行複製 (replication)，但一個分片至少會由一部主機 (或者 Container) 來管理。
- Asynchronous replication: 相較於 RAID 或同步複製機制，NoSQL DB 採用的是非同步的複製。這種方式較不會受到額外網路流量影響，能使寫入動作更快完成，但也因資料不是同步複製可能會發生資料遺失問題。
- BASE instead of ACID :NoSQL 強調的是效能與可用度，因此它注重的是 CAP。

NoSQL DB 的理論基礎 - CAP(Table 2.1)、BASE(Table 2.2)，BASE [7] 是基於 CAP 理論逐步演化而來，核心思想是：即便不能達到「強一致性」(Strong consistency)，但可以採用適當的方式來達到「最終一致性」(Eventual consistency) 的效果。

Table 2.1: CAP Theorem

Consistency(一致性)	在分散式環境說是多個節點的資料內容是否一致
Availability (可用性)	服務能一直保證是可用的狀態
Partition Tolerance (分區容錯性)	指的是網路的分區

Table 2.2: BASE 特性

Basically Available	基本可用
Soft-state	資料狀態可以有一段時間的不同步
Eventual consistency	資料狀態最終一致性

2.1.3 Dockers

Docker 是一個遵從 Apache 2.0 協議的開源專案。它利用於 Google 的 Go 語言實作，誕生於 2013 年初。最初是 dotCloud 公司內部的一個業餘專案，此專案後來加入了 Linux 基金會。Docker 專案的目標是實作輕量級的作業系統虛擬化解決方案。Docker 的基礎是 Linux 容器（LXC）等技術。在 LXC 的基礎上 Docker 進行了進一步的封裝，讓使用者不需要去關心容器的管理。使用者操作 Docker 的容器就像操作一個快速輕量級的虛擬機一樣簡單便捷 [23, 24]。

Docker 和傳統虛擬化方式的不同之處，可見容器是在作業系統層面上實作虛擬化，直接使用本地主機的作業系統，而傳統方式則是在硬體層面實作，由圖 Figure 2.3我們可以看出傳統虛擬化方式，我們還要多花費時間與硬體資源去建立 Guest OS，但 Docker 不需要。

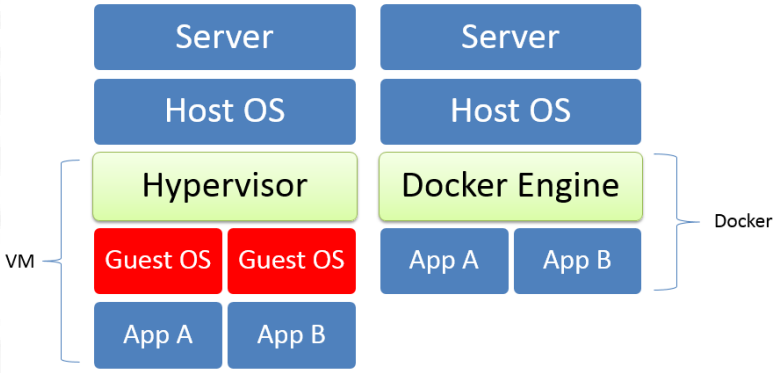

FIGURE 2.3: Docker 與 Virtual Machines 架構對比

Docker 容器的啟動可以在秒級實作，Docker 對系統資源的使用率很高，Docker 容器除了執行其中應用外，基本不消耗額外的系統資源，使得應用的效能很高。一台主機上可以同時執行數千個 Docker 容器。再者，程式設計師可以使用一個標準的映像檔來建立一套開發容器，待開發完成之後，測試或者上線人員可以直接使用這個容器來部署程式碼相當節省時間與精力。Docker 容器內可以安裝多個程式，例如同時安裝 Ubuntu、Apache、Nodejs 產生一個 Microsevices 的新軟體架構，日後若是服務有問題，我們可以很快去抽換其中一個 Docker 容器，不需要經歷過去需要重新安裝系統環境以及應用程式的麻煩。

2.1.4 MongoDB

MongoDB 是 MongoDB, Inc 在 2007 年十月所開發的文件式儲存無關聯式資料庫 [20, 21]，2009 年二月釋出第一個版本。MongoDB 使用 BSON 的資料格式，它是 JSON(JavaScript Object Notation) 的二進制型式，而 JSON 是一個以文字為基礎的資料交換語言，資料在寫入文件時會以 JSON [25] 的方式寫入，資料在傳輸時以 BSON(Binary Serialized Document Format) 的方式進行傳遞。另外和傳統的關聯式相比，MongoDB 少了綱要 (Schema) 的限制也就是 Schema-Free，使其可以在大量的資料下，自由決定資料的儲存欄位，而不會對效能造成影響 [29, 31]。

MongoDB 提供的 Schema-Free 的儲存模式，可以將不同結構的文件儲存在資料庫當中，形成一個 Collection(集合)。資料被以 Key-Value 方式形成一個 Document(文檔) 儲存於 Collection 中，其中 Document 中可以有 Embedded Documents(子文檔)，這種類似 JSON 的儲存格式稱之為 BSON。

BSON 支援豐富的層次結構，以及可利用簡易且功能強大的語句來進行提供動態查詢，相當適合查詢 STDF 這種本身文件即有複雜關係的資訊，雖然沒有傳統 RDBMS 之 JOIN 操作，也可以實現類似於關聯式資料庫查詢的絕大部份功能。

Table 2.3: MongoDB 與 RDBMS 對照

RDBMS	MongoDB
Database	Database
Table	Collection
Column	Field
Index	Index
Table joins	Embedded documents and linking
Specify any unique column or column combination as primary key	the primary key is automatically set to the _id field
Aggregation (e.g.group by)	Aggregation pipeline

MongoDB 以 Document 作為基本的資料儲存單位，有下列好處：

- JSON document 可對應於許多物件導向程式語言的原生資料類型 (如：物件)，這是近年來 JSON 和 ORM (Object-Relational Mapping) 日漸風行的原因。
- RDBMS 的 Join 操作其實是成本高昂的，Join 越多越慢越吃資源，但如果採用嵌入式文件 (embedded document) 的方式，可減少 Join 操作的使用。

MongoDB, Inc 於 2014 年 12 月併購了 WiredTiger，獲得了新的儲存層（Storage Layer）技術 WiredTiger，讓 MongoDB 進行儲存層的強化，進而擁有文檔層級 (Document-level) 的並行控制 (Concurrency Control)。讓系統即使處理頻繁寫入的工作，資料庫依然能維持效能一定的穩定度和可預測性，並且提供最高達 80% 的壓縮率。

MongoDB 的叢集架構，是由三大部分組成：

- 分割節點 (Shards)：負責儲存資料，每個分割節點為一複寫叢集 (Replica set)，由多台機器組成。因為有複寫的機制，所以位於同一個複寫叢集的每一台機器都擁有相同的資料。在複寫的叢集中有單點損壞時，可以將資料自動移轉到沒有損壞的機器上而達到高可用性。
- 設定伺服器 (Config Servers)：儲存所有分割節點的配置資訊，包含每筆資料分割鍵 (Shard Key) 的分佈。
- 路由伺服器 (Routing Process Servers, Mongos)：負責接收外部的查詢，當有請求進來時，路由伺服器會到設定伺服器依照鍵值來查詢資料，再將結果回傳。

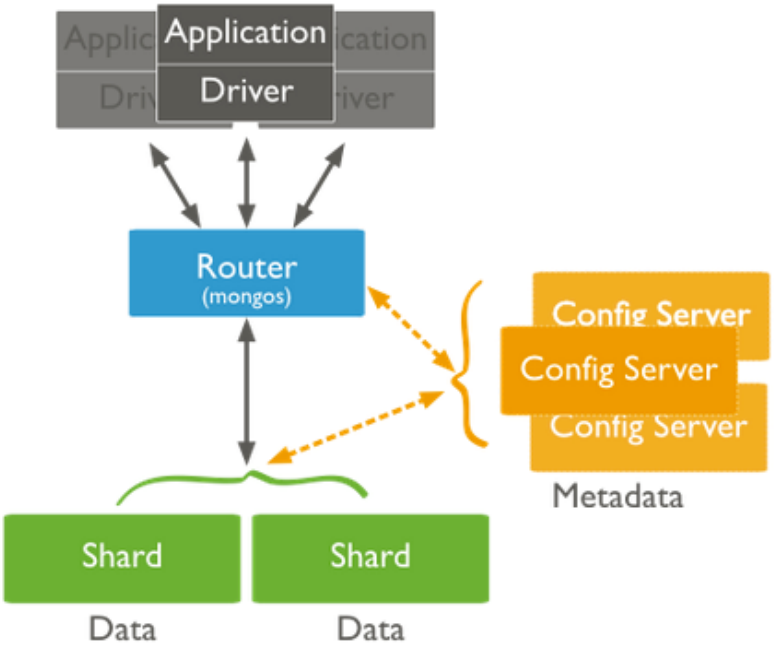

FIGURE 2.4: MongoDB 叢集架構

2.1.5 YCSB

YCSB(Yahoo！Cloud Serving Benchmark) 是 Yahoo 針對各式各樣的雲端服務資料庫所設計的 Benchmark 框架 [14–18]。

YCSB 的目標是為了制訂一個框架，和一套通用的計算方式，用作不同的資料庫測試彼此之間的速度比較。現在 NoSQL 的設計理念有很多的不同，但主要的設計趨勢，仍是以下列三項特色為主。

- 大量的資料流量：由於網路速度的提升以及服務應用的轉型，急需處理大量的資料流量。
- 彈性增加節點：讓管理者可以方便的增加雲端服務的節點數量，用以提供更多的儲存容量。
- 容錯性：應付所有可能發生的主機或網路的軟硬體問題。

YCSB 利用以下四個層次來做主要 Benchmark 的區分 [28]：

- Performance：利用逐漸增加 Loading，看是否會有異常或是效能有沒有出現不同的趨勢。
- Scaling：利用增加雲端節點的數目，同時也等比的增加 Loading 去比較是否有變動的幅度。
- Availability：測試當發生硬體或網路問題會對效能產生多大的負擔。
- Replication：資料庫提供雲端服務節點的數量是可以各自調整的。

2.1.6 YCSB 的架構

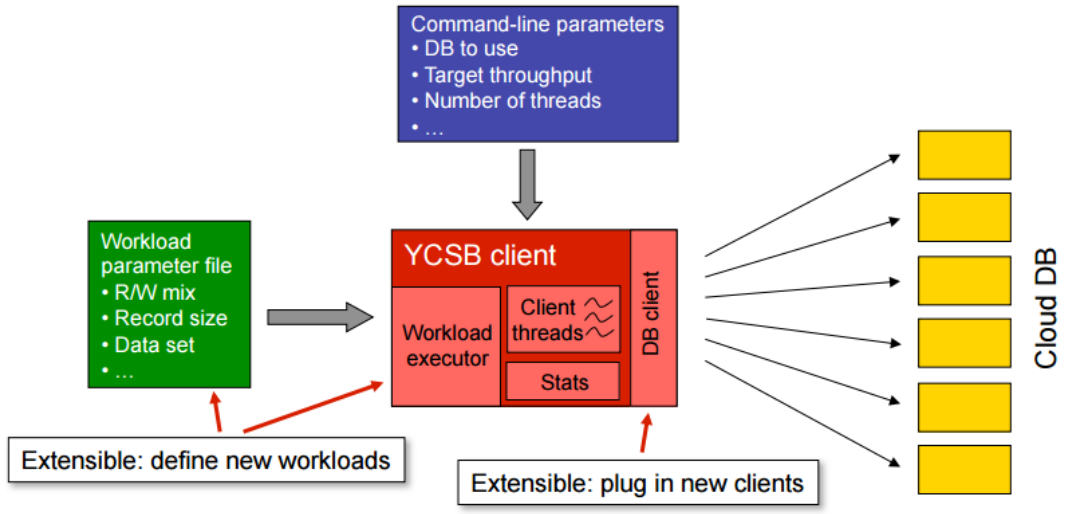

FIGURE 2.5: YCSB 與 Database 的架構

如圖 Figure 2.5，YCSB 主要可分為兩大部分，分別為 YCSB Client 與 Workload。

YCSB Client：負責跟各式各樣資料庫溝通，是 YCSB 最重要的一部分，它掌管了整個結構連結的方式。透過 Workload 的設定檔案，選擇對應的 Workload executor 執行。例 Read/Write、Record size、Popularity distribution、並利用多 Thread 模擬多台電腦同時進行資料新增狀態。

YCSB 使用方法：使用呼叫 DB Client 的 java 去讀取 workload 來依據所設定的測量方式，來做插入、讀取等使用方式。其步驟如下：

- 建立叢級資料庫。
- 選擇適當的 DB interface。
- 挑選測試的 workload。
- 每次測試需事先寫入資料庫，以及測試完畢需移除資料。
- 執行 workload，並將測試結果以時間的差別來進行呈現。

Workload：依據指定的 DB interface Layer 搭配使用者指令的要求，選擇搭配的 Workload 參數如表 Table 2.4來進行不同的數據測試。其中特別的是 Request Distribution，此一設定將會影響到 Read 獲得的結果，它有下列幾種設定：

- Uniform：任何一筆紀錄等概率隨機選擇。
- Hotspot：任何一筆紀錄隨機選擇。
- Zipfian：在選擇紀錄的同時，有些數據會比較常被選到。
- Lastest：類似 Zipfian 的選擇方式，但常被選到的那些紀錄將是比較新插入的那些資料。

TABLE 2.4: Workload 參數

Properties	Description	Default
Fieldcount	the number of fields in a record	10
Fieldlength	the size of each field	100
Readallfields	should reads read all fields (true) or just one (false)	True
Writeallfields	should updates and read/modify/writes update all fields (true) or just one (false)	False
Readproportion	what proportion of operations should be reads	0.95
Updateproportion	what proportion of operations should be updates	0.05
Insertproportion	what proportion of operations should be inserts	0
Scanproportion	what proportion of operations should be scans	0
readmodifywriteproportion	what proportion of operations should be read a record, modify it, write it back	0
Requestdistribution	what distribution should be used to select the records to operate on - uniform, zipfian, hotspot, or latest	Uniform
Maxscanlength	for scans, what is the maximum number of records to scan	100
scanlengthdistribution	for scans, what distribution should be used to choose the number of records to scan, for each scan, between 1 and maxscanlength	Uniform
Insertorder	should records be inserted in order by key (”ordered”), or in hashed order (”hashed”)	hashed
mongodb.writeConcern	describes the guarantee that MongoDB provides when reporting on the success of a write operation.options	acknowledged

2.2 相關研究

近年來，隨著半導體製程的日益精進，積體電路所具備功能益發強大。每個晶片的單價也就更加昂貴，因此驗證晶片功能的測試項目資料也就越來越多。在本論文中，我們的目標在提供儲存半導體測試資料 (STDF) 的雲端半導體測試資料庫系統。在傳統的關聯式資料庫，這些數量龐大 Data log 是無法處理應用的，要面對的挑戰項是分析、查詢、資料共享、儲存、傳遞、視覺化分析。為了解決這些難題，我們就必須引入巨量資料相關技術來解決。首先要解決的課題就是儲存，因此本論文提出一個建立於 Docker 上的使用 MongoDB 來彈性儲存測試資料的架構來解決此一問題。

文件導向 (Document-oriented) 儲存的 NoSQL 資料庫被證實了比傳統關聯式資料庫具有更多優點 [1]。Barbierato 等人 [5] 的研究發現，使用 NoSQL 來儲存或者查詢資料的效能都優於關聯式資料庫。美國軟體協會利用 YCSB 做出的測試報告 [2]，顯示出 MongoDB 比起其他的 NoSQL 資料庫 Cassandra 和 Couchbase 有更好的擴展性，MongoDB 評比效果明顯優於其他 NoSQL 資料庫。有一研究 [3]，利用了 MongoDB 來進行了電子病歷的儲存，證明了 MongoDB 面對這種非常複雜 (多種資料格式) 的資料，效能遠遠超過關聯式資料庫。另一個研究指出，使用 MongoDB 來進行風力發電渦輪機的資訊，他的儲存速度與查詢效能也比傳統關聯性資料庫好很多 [4] 。

Chapter 3

系統設計與實作

3.1 建置環境

本實驗研究在使用 VMware vSphere5.1.0 與 VMware Fusion6 作為雲端基礎架構即服務 (IaaS) 平台，兩種平台上均使用 Ubuntu Server 為作業系統，並安裝 Docker，使用 Dockerfile 來切換使用 MongoDB 二種版本（2.6.9、3.0.2）Container。

實驗環境配置如 Table 3.1，Table 3.2，兩種虛擬化環境平台差異在於，VMware vSphere 是建立一個伺服器虛擬化環境，本身就包含一個 Linux 內核的作業系統，故不需要建置於其他作業系統上，而 VMware Fusion 需要建置於其他作業系統上才能運行，所以十分依賴主作業系統，若主作業系統出問題時，虛擬機也會受到影響，故較不穩定，而 VMware vSphere 是直接控制及支援實體硬體，則穩定性會比 VMware Fusion 好。

3.1.1 Notebook 實驗環境 (Hosted Hypervisor)

使用 VMware Fusion 6 建置虛擬機環境平台，硬體的規格如下 Table 3.1：

Table 3.1: VMware Funsion 實驗環境 (單機)

CPU	Intel(R)CPU @ 2.00GHz 之 2 核
Memory	4GB
Hard Disk	100GB

3.1.2 Server 實驗環境 (Bare-Metal Hypervisor)

使用 VMware vSphere 5.1.0 建置兩台虛擬機環境平台環境，兩台 Server 硬體使用一樣的硬體規格如下 Table 3.2：

Table 3.2: VMWare vSphere 實驗環境 (多機)

CPU	Intel(R)CPU @ 3.50GHz 之 8 核
Memory	32GB
Hard Disk	300GB

3.1.3 Client 實驗環境

實驗 Client 端硬體配置如 Table 3.3

Table 3.3: Client 實驗環境

CPU	Intel(R) i7-3667U 2.00GHz 之 2 核
Memory	8GB
Hard Disk	256GB

3.1.4 Cloud Database 的研究

Cloud Database 通常是運行在 Cloud Computing 平台的資料庫，在 NoSQL 的資料模式下常見有兩種模式：

- Virtual Machine Image Deployment ：例如 MongoDB/Hadoop/CouchDB/ Neo4J on Amazon EC2。
- Database as a Server(DBaaS)：例如 Google App Engine Datastore、Amzaon SimpleDB。

Virtual Machine Image 是 Cloud 平台供應商允許使用者購買虛擬機的時間或者資源來使用，使用者可以自己上傳自己建立的機器 Image 檔或者使用 Cloud 平台供應商已經優化過的機器 Image。平台供應商負責資料庫的整體健康以及底層的軟體修補與更新，並且提供高可用性承諾 (例如 99.9%)。然而半導體測試後產生的 STDF 數據為客戶資產，且檔案龐大 (單一檔案大小超過 5GB) 不適合上傳至外部平台之上。

本研究建立一套在 Docker 上實作利用 MongoDB 來儲存半導體測試數據 STDF，且考量 Docker 與上層 MongoDB 儲存層新舊技術的進行效能評估，效能評估是本論文著重的部分，故效能評估有三個階段分別是效能模型 (Performance Model)、效能測試 (Performance Test)、效能分析 (Performance Analyzer)。

- 第一階段：效能模型主要是由五種 Open Source(Linux ubuntu 14.04 Server、Docker、MongoDB、MongoDB java Client、YCSB) 來建構半導體測試資料儲存效能測試系統。
- 第二階段：效能測試首先是將進行類 STDF 資料 Workload config 的設定，並使用 YCSB 進行不同數量的 Shard Server(增加 Container) 的測試; 接下來實際使用開發的 STDF Parser 將 JSON 存入 MongoDB 的效能測試。

- 第三階段：效能分析是額外再多利用 (Google GSON,STDF4j) 將 STDF 傳送與寫入效能數據進行分析。

3.1.5 半導體測試資料軟體架構

本系統架構圖主要在說明將測試數據 STDF 轉換成 JSON 經由 MongoDB client 端將資料傳送到 MongoDB 的 Mongos，再過 MongoDB 內部演算法將數據寫入 MongoDB 的 MongoD 中儲存，並於 MongoDB Config Server 紀錄儲存的 MongoD 位置。

Docker 與 MongoDB 之架構如圖，Figure 3.1:

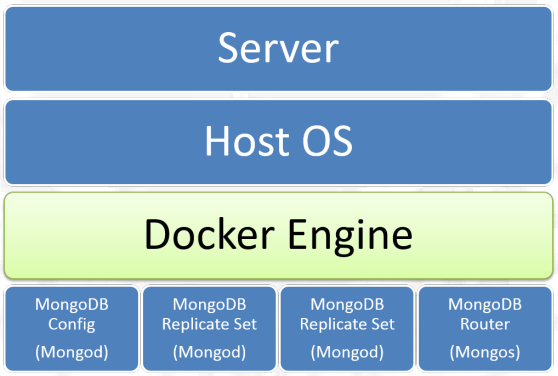

FIGURE 3.1: 建構於 Docker 之上的 MongoDB 系統環境圖

半導體晶圓測試資料雲端資料系統是透將雲端服務與半導體測試資料結合，透過網際網路的便利性，提供工程師可以隨時隨地查詢放在雲端上的測試資料建立 GUI 半導體測試資料查詢系統。系統流程如圖 Figure 3.2 顯示：

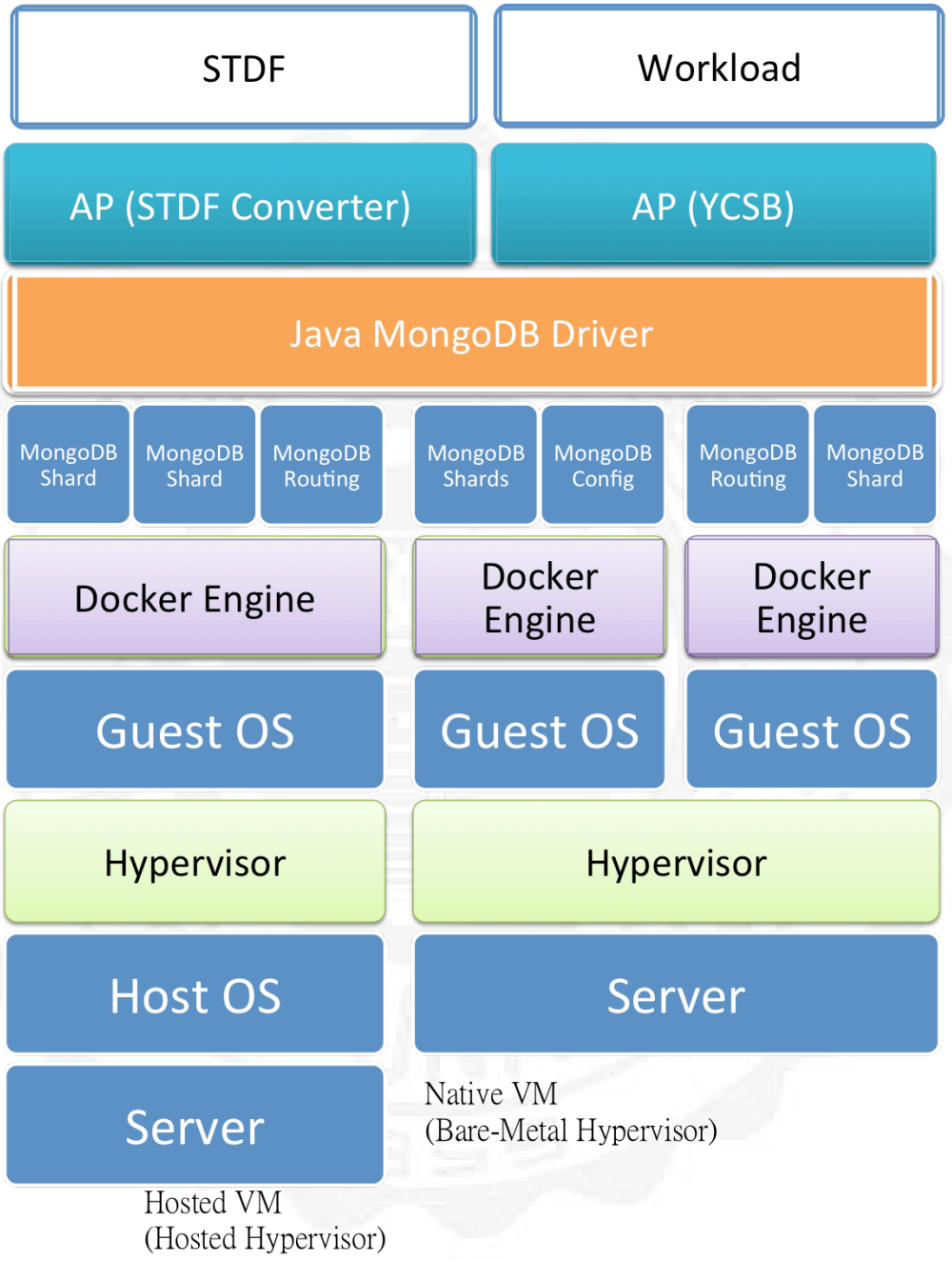

FIGURE 3.2: 半導體測試資料軟體架構圖

- STDF：半導體測試數據檔案。
- STDF Converter 包含 STDF4j(解析 STDF 之 Java Open Source) 以及 Google GSON(Google 提供的將 Java Object 轉換為 JSON)。
- MongoDB Shards Server：負責儲存資料，每個分割節點為一複寫叢集 (Replica set)，由多台機器（Contianer）組成。

- MongoDB Config Server：儲存所有分割節點的配置資訊，包含每筆資料分割鍵 (Shard Key) 的分佈。
- MongoDB Routing Server：負責接收外部的查詢，當有請求進來時，路由伺服器會到設定伺服器依照鍵值來查詢資料，再將結果回傳。
- MongoDB java Client：用來訪問 MongoDB 並利用指令操作 MongoDB。
- Docker：MongoDB 的分割節點 (Shards 的 Replica set,Mongod)、設定伺服器 (Config Servers,Mongod)、路由伺服器 (Routing Process Servers, Mongos) 都使用 Container 來建立。

3.2 系統架構

3.2.1 系統架構圖

作業員操作機台進行生產，產生 STDF 檔案於機台 Direct Mount 的 NAS 空間，常駐的 STDF Converter 去 Monitor 特定 Direct Mount 的 NAS 空間，若有 STDF 產生時，AP 利用 STDF4j、Google GSON 將 STDF 轉成 JSON，然後將 JSON 資料透過 Java MongoDB driver 去 insert 到 MongoDB。測試工程師透過 GUI 去進行查詢，獲得儲存於 MongoDB 資料庫的資料來進行分析動作。系統流程如圖 Figure 3.3 顯示：

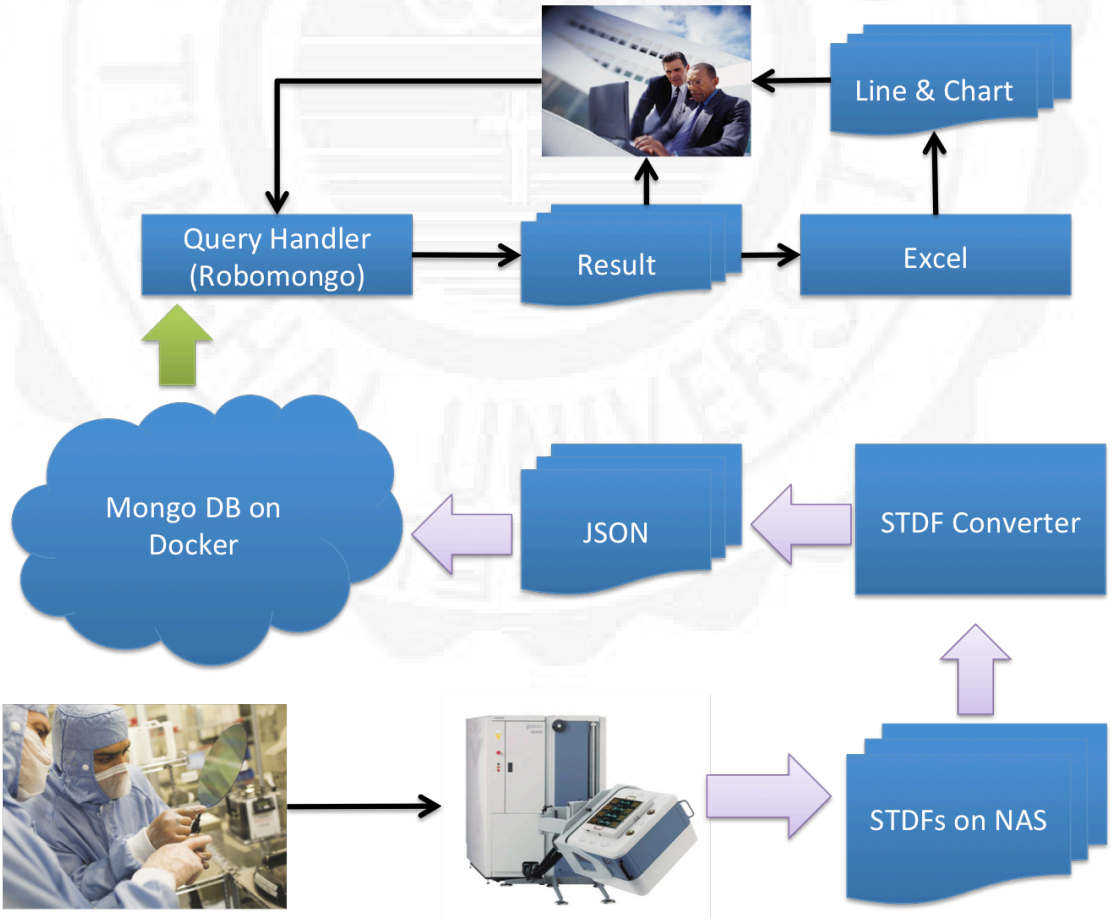

FIGURE 3.3: 系統架構圖

3.3 研究材料

3.3.1 半導體測試

依設計產品的所使用的規格, 使用電性偵測的方式對此產品進行驗證稱之為測試，其大致輪廓如 Figure 3.4所示。

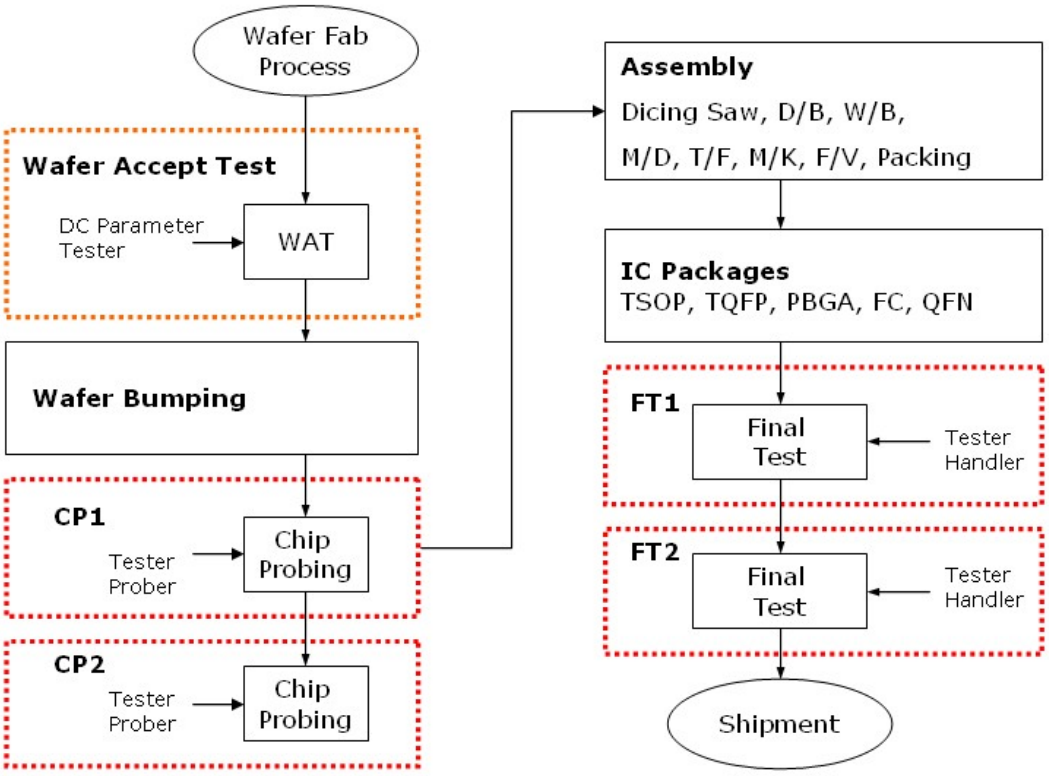

FIGURE 3.4: General of Testing Profile

半導體元件的測試通常可區分為裸晶測試 (封裝前測試) CP(chip probe) 與最終測試 (封裝後測試) FT(final test) 兩種：

- 裸晶針測:CP(Chip Probing)：將晶片上的晶粒根據設計的電性標準規格，以針測方式檢測其良劣的作業稱之為 CP。又由於是晶圓級的測試，故又稱之為 Wafer Sorting。若良品率過低，則表示在晶圓製造的過程中，有某些步驟出現問題，必須儘快通知工程師檢查。

- 最終測試:FT(Final Testing)：針對封裝後的成品，再做一次電性針測的作業，稱之為 FT，有時也稱為成品測試。這是為確保產品在封裝後仍符合設計的規格。

STDF 的介紹：STDF（Standard Test Data Format）文件 [9–12] 是由各種資料記錄組成的一種 Binary 檔案。每一個資料記錄含有頭 (Header) 資訊和資料本身。其中頭資訊又包括資料長度 (REC_LEN)（佔 2 個 Byte）、資料類型 (REC_TYPE)（佔 1 個 Byte）和子類型 (REC_SUB)（佔 1 個 Byte）。在頭資訊中資料類型和子類型是成對的出現的整數。我們可以根據 REC_TYPE+ REC_SUB 的組合找出這種資料記錄的含義。

Table 3.4: STDF Record Types and Subtypes 部份列表

RecType	RecSubType	Record	Description
0			Information about the STDF File
	10	FAR	File attribute record
	20	ATR	Audit trail record
1			Data collected on a per lot basis
	10	MIR	Master information record
	20	MRR	Master results record
	30	PCR	Part count record
	40	HBR	Hardware Bin record
	50	SBR	Software Bin record
	60	PMR	Pin map record
	62	PGR	Pin group record
	63	PLR	Pin list record
	70	RDR	Retest data record
	80	SDR	Site description record
10			Data Collected per test in the test
	30	TSR	Test synopsis record
15			Data Collected per test execution
	10	PTR	Parametric test record
	15	MPR	Multiple-result parametric record
	20	FTR	Functiona test reocrd

STDF 中最主要的一筆資料紀錄為 Master Information Record(MIR)，用來識別每個 STDF 檔案，他記錄著該次測試的通用資訊包括貨批資訊、開始/結束測試時間、使用的測試機、使用的測試程式、生產的用具代號、或者是由哪位作業員所生產，其欄位資訊如下表。

TABLE 3.5: Master Information Record

Field Name	Data Type	Field Description
REC_TYPE	U*1	Record type (1)
REC_SUB	U*1	Record sub-type (10)
SETUP_T	U*1	Date and time of job setup
START_T	U*1	Date and time first part tested
STAT_NUM	U*1	Tester station number
MODE_COD	C*1	Test mode code (e.g. prod, dev)
RTST_COD	C*1	Lot retest code
PROT_COD	C*1	Data protection code
BURN_TIM	C*1	Burn-in time (in minutes)
CMOD_COD	C*1	Command mode code
LOT_ID	C*n	Lot ID (customer specified)
PART_TYP	C*n	Part Type (or product ID)
NODE_NAM	C*n	Name of node that generated data
TSTR_TYP	C*n	Tester type
JOB_NAM	C*n	Job name (test program name)
JOB_REV	C*n	Job (test program) revision number
SBLOT_ID	C*n	Sublot ID
OPER_NAM	C*n	Operator name or ID (at setup time)
EXEC_TYP	C*n	Tester executive software type
EXEC_VER	C*n	Tester exec software version number
TEST_COD	C*n	Test phase or step code
TST_TEMP	C*n	Test temperature
USER_TXT	C*n	Generic user text
	be omitted	

STDF 佔絕大多數的資料記錄為 Parametric Test Record(PTR)3.6。STDF 中有數千乃至數萬個晶片資料，每一個晶片都有數百 PTR 資料紀錄，因此一個檔案將會有幾百萬個 PTR，PTR 於 STDF 檔案首次出現時還會儲存該測試的預設值，比如限制、單位和數值範圍。

Table 3.6: Parametric Test Record

Field Name	Data Type	Field Description
REC_TYPE	U*1	Record type (15)
REC_SUB	U*1	Record sub-type (10)
TEST_NUM	U*4	Test number
HEAD_NUM	U*1	Test head number
SITE_NUM	U*1	Test site number
TEST_FLAG	B*1	Test flags (fail, alar, etc.)
PARM_FLAG	B*1	Parametric test flags (drift,etc.)
RESULT	R4	Test Result
TEST_TXT	C*n	Test description text or label
ALARM_ID	C*n	Name of alarm
OPT_FLAG	B*1	Optional data flag
RES_SCAL	I*1	Test results scaling exponent
LLM_SCAL	I*1	Low limit scaling exponent
HLM_SCAL	I*1	High limit scaling exponent
LO_LIMIT	R*4	Low test limit value
HI_LIMIT	R*4	High test limit value
UNITS	Cn	Test units
C_RESFMT	C*n	ANSI C result format string
C_LLMFMT	C*n	ANSI C low limit format string
C_HLMFMT	C*n	ANSI C high limit format string
	be omitted	

Data Code 說明則如3.7

TABLE 3.7: Data Type Codes in STDF

Code	Description
Cn	Character string of length n
U1,U2,U4	One, two and four bytes unsigned integer
I1,I2,I4	One, two and four bytes signed integer
R4,R8	Four,eight bytes floating point number
Vn	Variable data type field, data type specified in the first byte(max 255)
Bn	Variable length bit encoded field;first byte denotes the count;first data bit in LSB of second byte
Dn	Variable length bit-encoded field;first two bytes specify the count
N1	Unsigned integer stored in nibble

3.3.2 半導體測試資料 JSON

半導體測試資料 JSON 資料範例：本次實驗僅先將 STDF 中佔多數的 PTR 轉入 MongoDB 資料庫。因此測試資料內容之 JSON 架構僅紀錄 MIR/PTR 等兩項 Record 之部份，如 Figure 3.5所示。其中 PTR 部分為 n 筆，其數量視 STDF 內測試資料數量而定。

{ "DEVICE_NAME":"DEVICE01"，"PROGRAM_NAME":"PGM01"，
{ "ITEM_NAME":"TEST1"，"TESTER_ID":"TXX01"，
"PROBER_HANDER":"P1234"，"P/C_ID":"AT001"，
"START_TIME":["\/Date(1330444800000)\/"]，
"END_TIME":["\/Date(1330444800000)\/"]，
"PTR": [
{"x":"1"，"Y":"3"，"SITE":"1"，"BIN_PF":"P"，"ITEM_PF":"F"，"VALUES":"0.22"}，
{"x":"2"，"Y":"3"，"SITE":"1"，"BIN_PF":"P"，"ITEM_PF":"F"，"VALUES":"0.23"}，
…]}

FIGURE 3.5: 半導體測試資料 Data Log JSON 資料

3.4 系統實作

本實驗研究在使用 VMware Fusion 與 VMware vSphere 來架設虛擬機環境平台，並在此平台安裝 Docker 以及使用 Docker 建立 MongoDB 的 Container，虛擬環境均使用 Ubuntu Server 為作業系統，並利用 Mongo Dockerfile 建立 MongoDB Docker image，並利用上述的 Container 建立的 MongoDB Shard Server 作為半導體測試資料儲存系統。

使用的軟體相關規格如下 Table 3.8：

TABLE 3.8: Software Specification

	Version
OS	Ubuntu 14.04 LTS
MongoDB Version	3.0.2/2.6.9
Java JDK	Oracle JDK8 Update45
VMware Fusion	6.0
VMware vSphere	5.1
Docker	1.6
Google GSON	2.3.1
STDF4j	0.0.2

系統建置步驟：

- 1. 安裝 Ubuntu 作業系統。
- 2. 更新及升級作業系統套件，並安裝 SSH 服務。
- 3. 安裝 JDK。
- 4. 安裝 Docker。
- 5. 建立 Docker files (Mongod、Mongos)。
- 6.Create MongoDB Replica Sets Server。

- 7.Initialize Replica Set Server。
- 8.Create Some MongoDB Config Server。
- 9.Create MongoDB Router Server。
- 10.Initialize MongoDB Shard。
- 11. 輸入 docker 命令確認 MongoDB Shard Server 完成架設如圖 Figure 3.6 顯示。

```
craig@ubuntu:~$ sudo docker ps -a
CONTAINER ID        IMAGE                    COMMAND                  CREATED             STATUS              PORTS                     NAMES
d11b474f5ba7        dev24/mongos:latest      "usr/bin/mongos --po   7 days ago          Up 7 days           0.0.0.0:27017->27017/tcp  mongos1
9a44c8ac1e61        dev24/mongodb:latest     "usr/bin/mongod --no   7 days ago          Up 7 days           0.0.0.0:32783->27017/tcp  cfg4
f092a6047673        dev24/mongodb:latest     "usr/bin/mongod --no   7 days ago          Up 7 days           0.0.0.0:32782->27017/tcp  cfg3
d487f5d01e59        dev24/mongodb:latest     "usr/bin/mongod --no   7 days ago          Up 7 days           0.0.0.0:32781->27017/tcp  cfg2
aba93e78495e        dev24/mongodb:latest     "usr/bin/mongod --no   7 days ago          Up 7 days           0.0.0.0:32780->27017/tcp  cfg1
7b5d8c4f5966        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32779->27017/tcp  rs4_srv3
80df3cfb1647        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32778->27017/tcp  rs4_srv2
3ae0d2c920d4        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32776->27017/tcp  rs3_srv3
529080d2f8fa        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 2 seconds        0.0.0.0:32784->27017/tcp  rs4_srv1
bb71017f0305        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32775->27017/tcp  rs3_srv2
b414b7bed0a0        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32774->27017/tcp  rs3_srv1
cb2102da45c1        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32773->27017/tcp  rs2_srv3
3844959f5fa3        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32772->27017/tcp  rs2_srv2
ba4dd5e56beb        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32771->27017/tcp  rs2_srv1
3346f5398659        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32770->27017/tcp  rs1_srv3
cdd4fc2e1838        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32769->27017/tcp  rs1_srv2
1735af43ff8f        dev24/mongodb:latest     "usr/bin/mongod --re   7 days ago          Up 7 days           0.0.0.0:32768->27017/tcp  rs1_srv1
craig@ubuntu:~$
```

FIGURE 3.6: MongoDB Container

- 12. 下載與安裝 Apache Maven
- 13.download YCSB for MongoDB for MongoDB Driver、MongoDB Driver
- 14. 修改 POM.xml

經由上述步驟建立的系統即可執行 YCSB 進行效能的測試，以及準備好提供 STDF Converter 將 stdf 轉入資料庫當中。STDF Converter 可以經由 Config 設定下列資訊，STDF 轉入資料庫後，我們可以於 Robomongo GUI 看到結果如圖 Figure 3.7 顯示。

- 1. Monitor 的 STDF 檔案路徑
- 2. File Size Check 的 Interval 秒數
- 3. File Size 最多檢查次數
- 4. Monitor 的 STDF File Name 的黑白名單

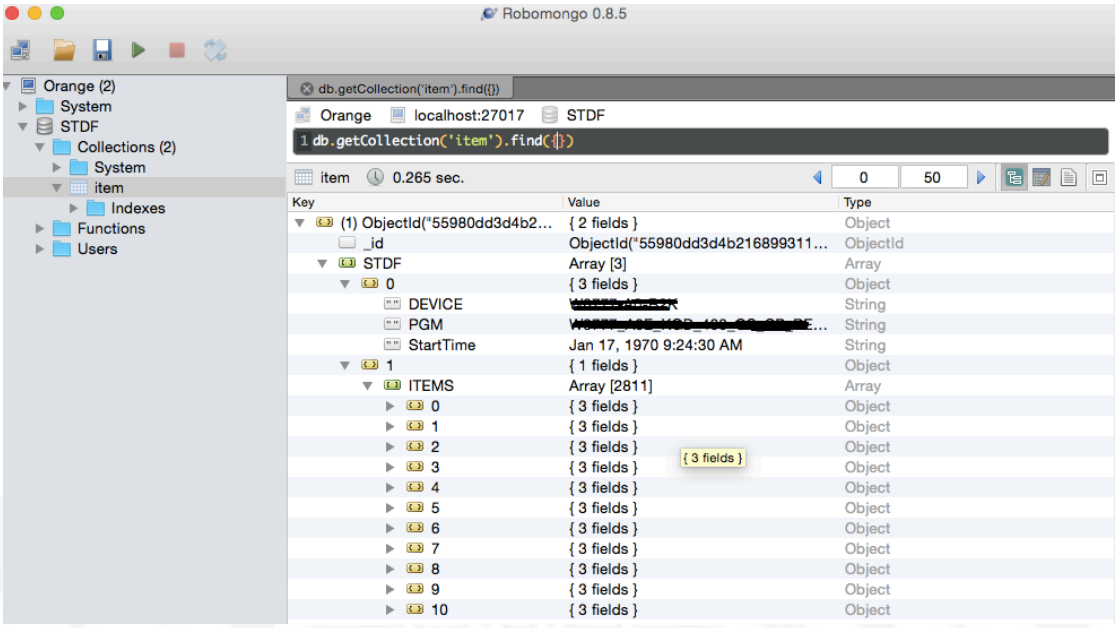

FIGURE 3.7: STDF 資料查詢於 Robomongo 結果

Chapter 4

實驗環境與結果

4.1 實驗環境

在進行實驗的時候，使用者透過執行 Shell 的 Script 檔案，並且透過參數指定使用的 workload 以及 Interface (e.g MongoDB)。接著系統會先把資料寫入測試的資料庫當中 (一般預設名稱為 YCSB，使用 workloadA)，然後再進行測試的動作。其中要注意的是，每次測試完畢都要把原先測試後殘留的資料進行清除，重新輸入測試的資料數據，用以確定每次測試的公平，其架構如 Figure 2.5。

本論文使用的 workload 讀寫參數如 Table 4.1：

Table 4.1: 測試 workload 設定

Workload	Property	Read	Update	Insert
WorkloadA	輸入資料	0%	0%	100%
WorkloadB	寫多讀少	10%	90%	0%
WorkloadC	混合讀寫	65%	10%	25%
WorkloadD	讀多寫少	90%	10%	0%
WorkloadE	全讀	100%	0%	0%

不同測試 workload，但為相同的參數如下：

- 1. recordcount=10000000
- 2. operationcount=100000000
- 3. fieldlength=100
- 4. fieldcount=15
- 5. threadcount=10
- 6. Distribution=uniform
- 7. mongodb.writeConcern=acknowledged

4.2 實驗方法

4.2.1 實驗測試 - Throughput

於 Hosted VM 之 Throughput 的比較單一 Server 設定 4 Shard Server，MongoDB (2.6.9、3.0.2) Figure 4.1，我們可以看到 3.0.2 有較 2.6.9，有比較佳的 Throughput 測試結果。

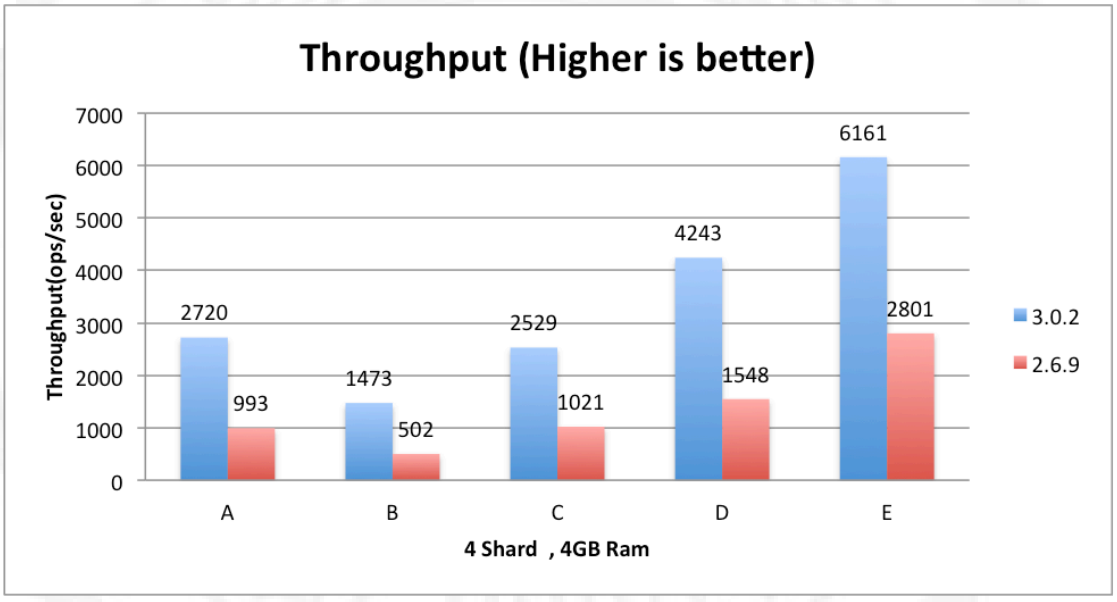

FIGURE 4.1: NB 環境測試的 Throughput 結果

於 Native VM 下之單一 Server 下，我們可以看到 MongoDB 3.0.2 有較 MongoDB2.6.9 有比較佳的 Throughput 測試結果 (2.6.9、3.0.2)Figure 4.2。

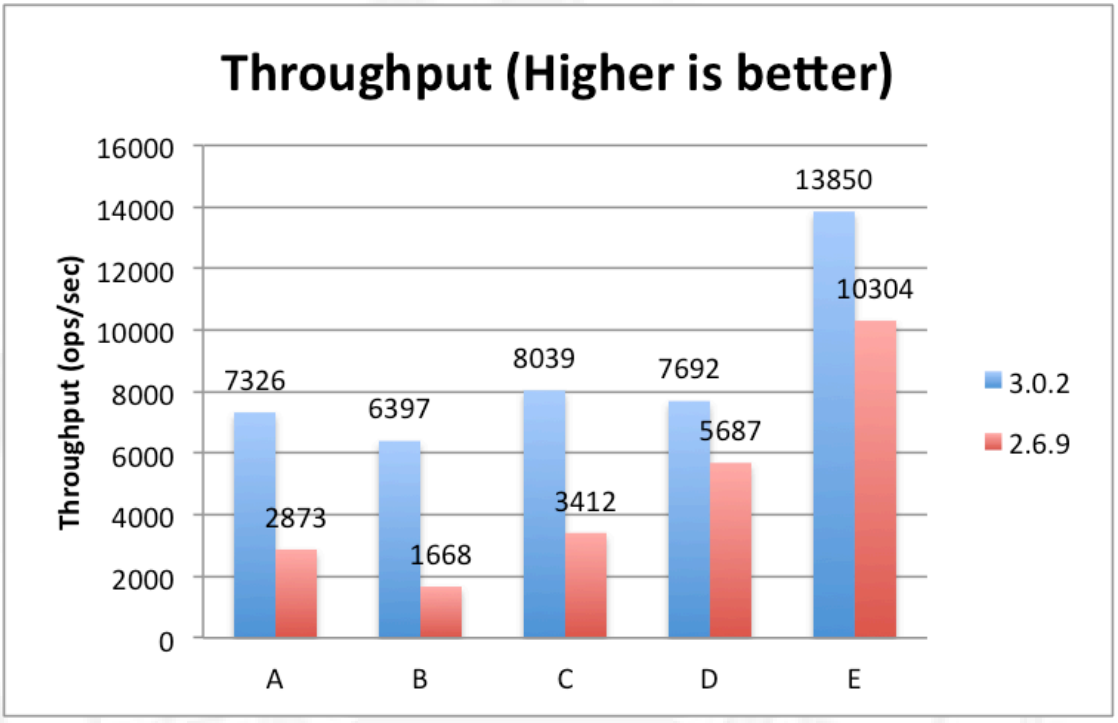

FIGURE 4.2: 單一 Server 環境測試的 Throughput 結果

於 Native VM 下之二台 Server 下在同樣的軟硬體環境, 進行 8 個 Shard Server 的 Throughput 的比較測試，我們可以看到 3.0.2 有較 2.6.9，有比較佳的測試結果 Figure 4.3。

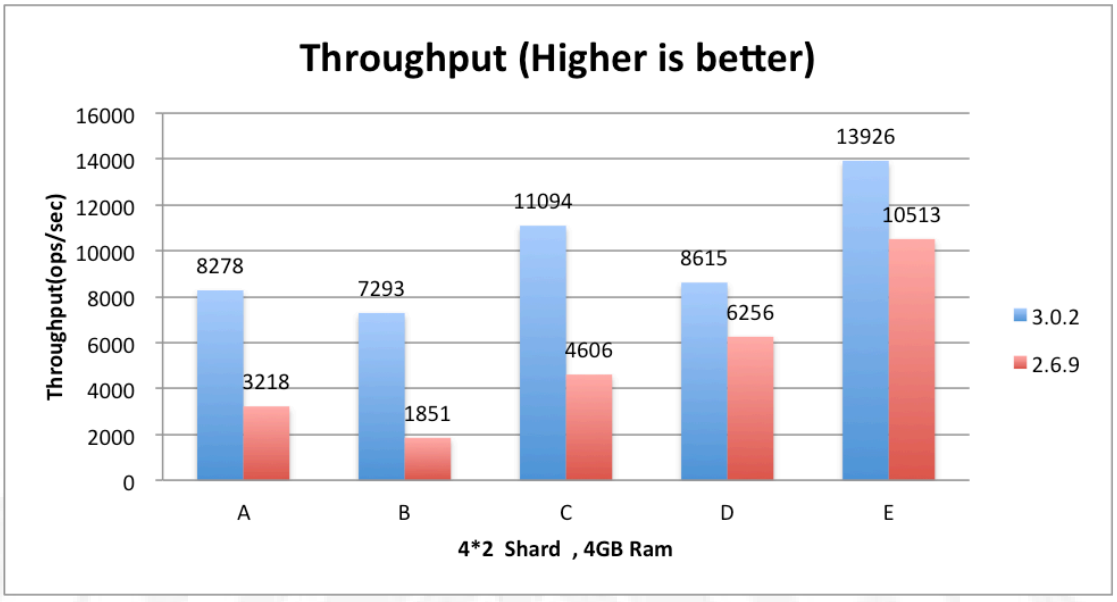

FIGURE 4.3: 二台 Server 環境測試的 Throughput 結果

MongoDB 2.6.9 的寫入會影響 Read 的操作，因而影響 Throughput 的輸出，綜合以上測試，我們可以發現 MongoDB 3.0.2 可以獲得更好的 Throuphput 結果。

4.2.2 實驗測試 - Update Latency

Hosted VM 之 Update Latency 的比較，在 4 Shard Server，MongoDB (2.6.9、3.0.2) Figure 4.4，我們可以看到 3.0.2 有較 2.6.9，於 NoteBook 環境 3.0.2 大約低了 80% 有比較佳的 Update Latency 測試結果，在 WorkloadC(讀寫混合) 情境中，MongoDB 3.0.2 顯著改善 Throughput，將給用戶更好的使用回饋，因為 STDF 資料常常會大量匯入與大量讀取。

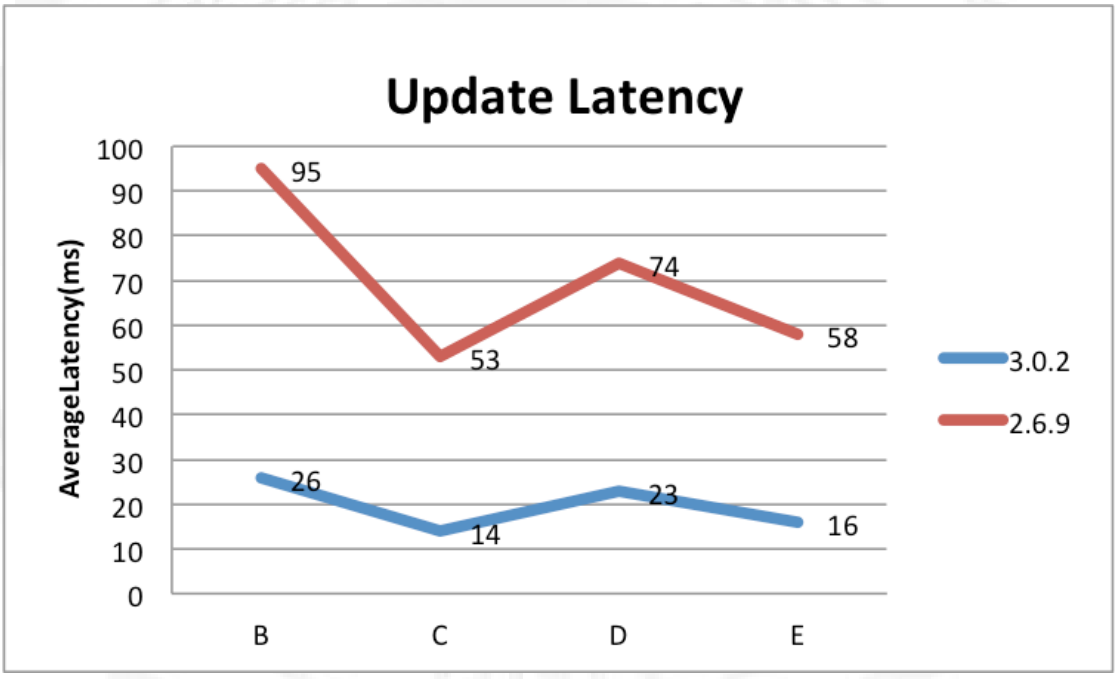

FIGURE 4.4: NB 環境測試的 Update Latency 結果

Native VM 之 Update Latency 的比較，在 8 (4*2) Shard Server，MongoDB (2.6.9、3.0.2) Figure 4.5，我們可以看到 3.0.2 有較 2.6.9，於二台 Server 有比較佳的 Update Latency 測試結果。

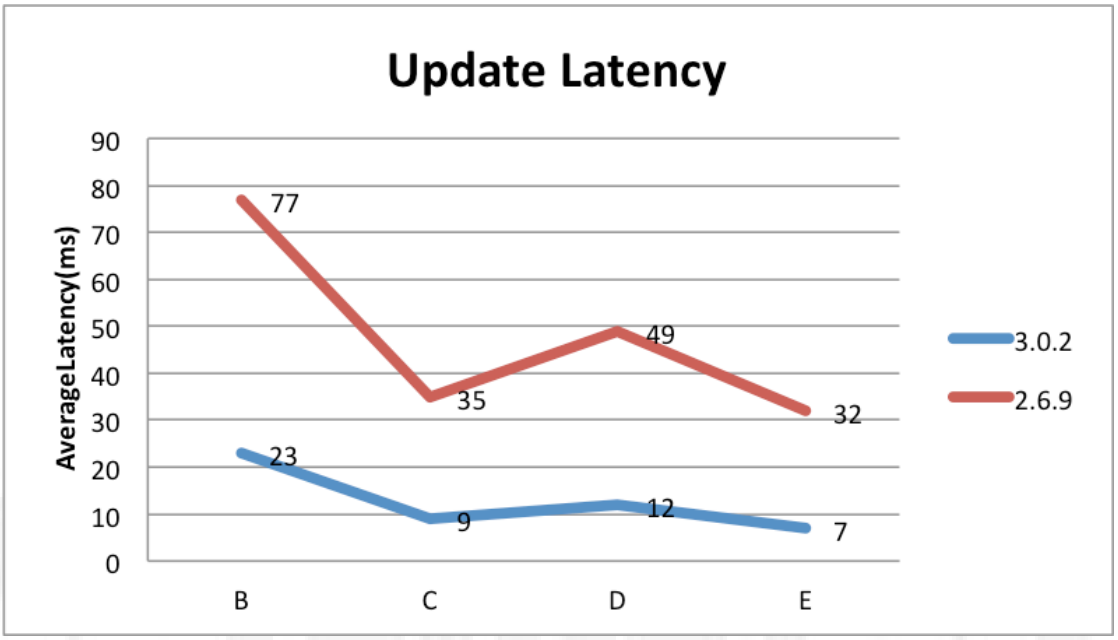

FIGURE 4.5: 二台 Server 環境測試的 Update Latency 結果

4.2.3 實驗測試 - Throughput(不同數量之 Shard)

Hosted VM 之 Throughput 的比較，在同樣的硬體環境與不同數量的 Shard Server，MongoDB (3.0.2) 越多數量的 Shard Server 可帶來更好的 Throughput 效果 Figure 4.6，於 NoteBook 環境，3.0.2 版本隨著 Shard Server 數量的增加，Throughput 逐步有更佳測試結果，

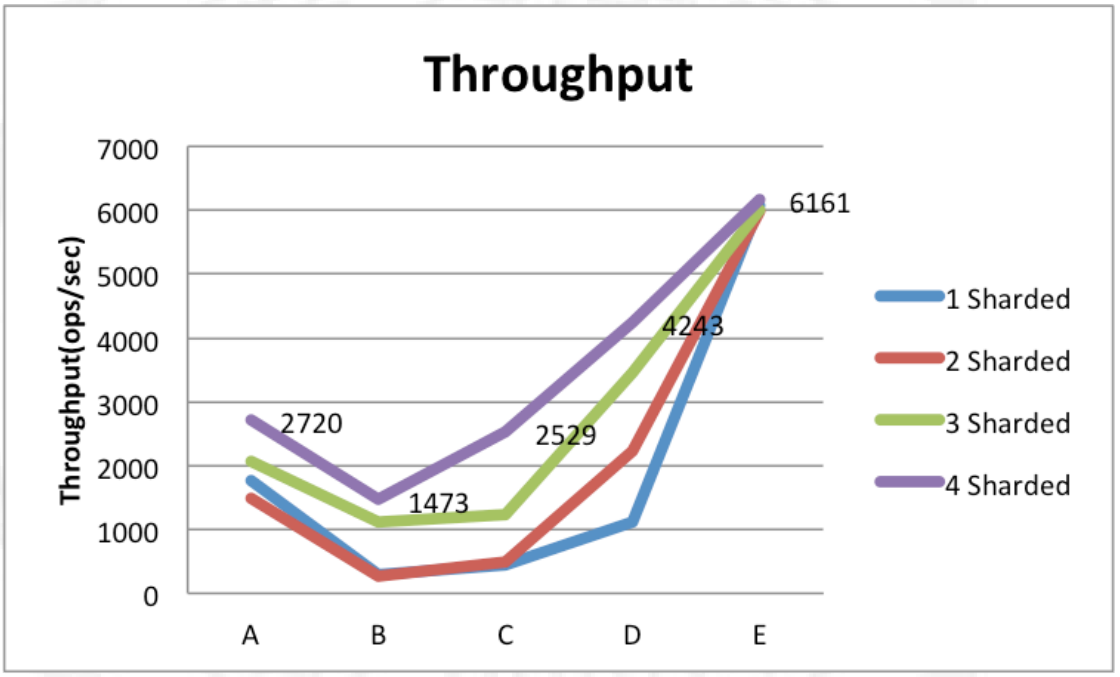

FIGURE 4.6: NB 環境測試的不同 Shard Server 數量 Throughput 結果

Native VM 之 Throughput 的比較，在同樣的硬體環境與不同數量的 Shard Server，MongoDB (3.0.2) 越多數量的 Shard Server 可帶來更好的 Throughput 效果 Figure 4.7，於二台 Server 環境，3.0.2 版本隨著 Shard Server 數量的增加，Throughput 逐步有更佳測試結果。

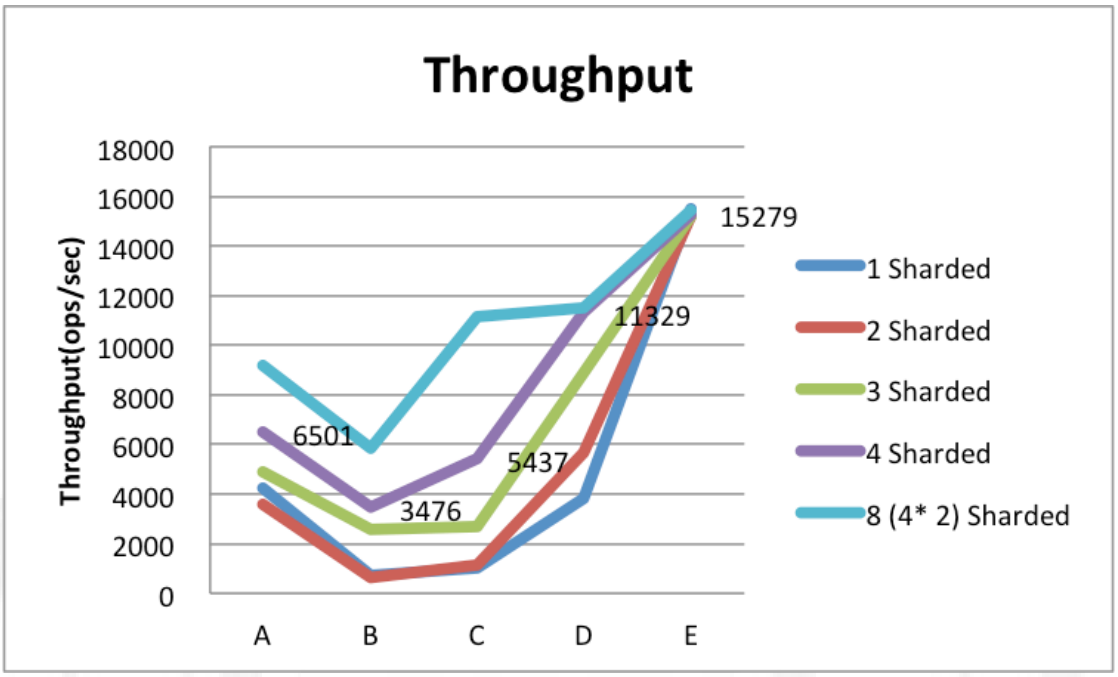

FIGURE 4.7: 二台 Server 環境測試的不同 Shard Server 數量 Throughput 結果

隨著 Shard Server 增加會看到在 Read 的應用情況不會有太大幫助，因此會看到 WorkloadE(僅讀取)，並沒有大改進。對於增加 Shard Server 來增加 Throughput，最有幫助的是在對於 WorkloadC(混合讀寫) 的情況。

4.2.4 實驗測試 - STDF 資料新增至 MongoDB

實際將 STDF 資料轉換為 JSON 後塞入至 MongoDB 3.0.2 ，使用 RAM 24GB、INETL XEON E5-2640 CPU * 4、Shard Server *2、Config Server * 1 Figure 4.8，我們可以看到隨著資料量的增加其效能依舊是不錯的：

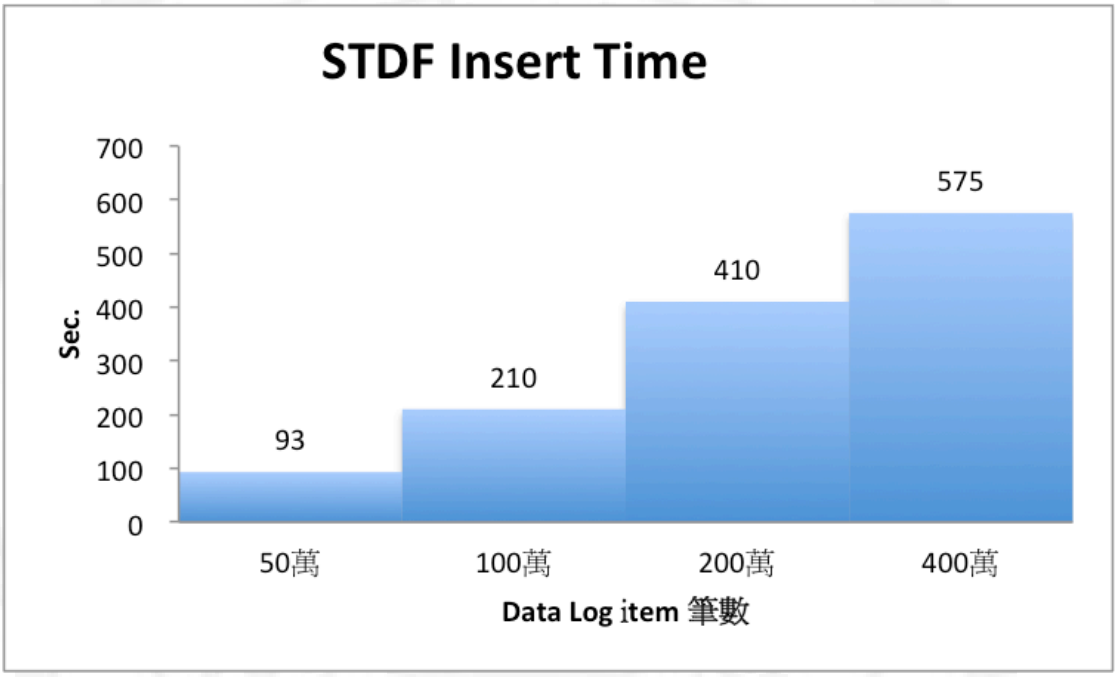

FIGURE 4.8: 實際塞入 STDF Data Log 花費時間

4.3 實驗結果

實驗表明，我們可以確定 MongoDB 架構在 Docker 來存放半導體測試資料有很好的存儲速度與查詢時間。綜合以上，我們可以確定 MongoDB3.0.2 使用 WiredTiger 儲存層後，讀取效能較 2.6.9 來的好很多，不僅提供了資料壓縮功能，也帶來了效能的提升。而 Document level 讓它即時面對大量存取也可以有一定的穩定度。

Chapter 5

結論與未來方向

5.1 結論

在大數據的時代，資料以及速度就是一切，過去受限於技術及硬體條件限制，要花費大量時間才可以獲得分析的結果，如何運用大數據資料在半導體生產過程中，提升生產良率、製成品質是半導體大數據分析很重要的工作。使用者可以快速連上公司伺服器，利用網路查詢已經事先存入 NoSQL 資料庫的半導體測試資料，他們將可以快速又有效率的找出可能的製程變異問題。另外，使用免費開放原始碼以及雲端的系統，將可以彈性的讓過去沒有進資料庫的數據一起加入分析的資料來源。

故本論文建構的雲端儲存系統環境，並測試其效能，如果只需要建置小型雲端儲存系統，可以利用現有 NB 或 PC 主機來建置此系統，因使用 Docker 的 Container 方式來建置 MongoDB，將可以大大縮短一般使用虛擬機器花費的時間。考慮到系統穩定性與安全性，則建議使用 Sharded Server 環境來建置系統，將可以提供公司穩定存取資料的基礎。

5.2 未來方向

半導體測試數據不只有 STDF 檔案，未來可以將其他生產相關數據一併存入雲端儲存系統，妥適地選用 Shard Key 來進行資料的 Sharding，將巨量資料進行適當切割，那麼查詢時將能直接對應所要查詢的 Shard，以提升查詢上的速度。未來可以實驗之方向如下:

- 進行 Shard Key 的挑選層別，本研究將 STDF By ITEM 去進行 JSON to MongoDB 之研究，未來可以進行相關 Shard Key 的研究分析。
- 加入設備的 Life Cycle Time 數據，如: Probe Card(探針卡) 的使用數據，獲得生產設備的逐漸劣化對於測試結果的影響。
- 將篩選出來的查詢結果運用資料視覺化（Data Visualiztion）方式，清楚地表達出半導體測試資料所提供的資訊。
- 嘗試使用商業的 PaaS 服務，如: Azure Document DB（同為 Document DB 但可使用 SQL 查詢)，來進行測試資料儲存的效能分析。

參考文獻

[1] B.G. Bucur C. Tudorica. A comparison between several nosql databases with comments and notes. In *Roedunet International Conference (RoEduNet) 2011 10th*. IEEE, 2011.

[2] United Software Associates. Comparative benchmarks: Mongodb vs. couchbase vs. cassandra https://www.mongodb.com/collateral/comparative-benchmarks-mongodb-vs-couchbase-vs-cassandra, March 2015.

[3] Wei Xu, Zhonghua Zhou, Hong Zhou, Wu Zhang, and Jiang Xie. Mongodb improves big data analysis performance on electric health record system. In *Life System Modeling and Simulation*, pages 350–357, 2014.

[4] Qile Wang, Zhu Shen, Long Ma, and Shi Yin. In W.Eric Wong and Tingshao Zhu, editors, *Computer Engineering and Networking*, volume 277 of *Lecture Notes in Electrical Engineering*, pages 403–409. Springer International Publishing, 2014.

[5] Enrico Barbierato, Marco Gribaudo, and Mauro Iacono. Performance evaluation of nosql big-data applications using multi-formalism models. *Future Generation Computer Systems*, 37:345 – 353, 2014.

[6] Seth Gilbert and Nancy Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2): 51–59, June 2002.

[7] Eric A. Brewer. Towards robust distributed systems (abstract). In *Proceedings of the Nineteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '00, pages 7–, New York, NY, USA, 2000. ACM.

[8] Theo Haerder and Andreas Reuter. Principles of transaction-oriented database recovery. *ACM Comput. Surv.*, 15(4):287–317, December 1983.

[9] Teradyne Inc. Standard test data format specification version 4.0. Teradyne, 2007.

[10] A. Khoche, P. Burlison, J. Rowe, and G. Plowman. A tutorial on stdf fail datalog standard. In *Test Conference, 2008. ITC 2008. IEEE International*, pages 1–10, Oct 2008.

[11] A. Khoche, J. Katz, S. Landini, Kochen Liao, N. Agrawal, G. Plowman, Song lin Zuo, Liyang Lai, J. Rowe, and T. Zanon. Stdf memory fail datalog standard. In *VLSI Test Symposium, 2009. VTS '09. 27th IEEE*, pages 209–214, May 2009.

[12] M. Seuring, M. Braun, A. Ma, G. Eide, K. Yang, and Huaxing Tang. Employing the stdf v4-2007 standard for scan test data logging. *Design Test of Computers, IEEE*, 29(6):91–99, Dec 2012.

[13] Ming-Ju Wu, J.-S.R. Jang, and Jui-Long Chen. Wafer map failure pattern recognition and similarity ranking for large-scale data sets. *Semiconductor Manufacturing, IEEE Transactions on*, 28(1):1–12, Feb 2015.

[14] Brian F. Cooper, Adam Silberstein, Erwin Tam, Raghu Ramakrishnan, and Russell Sears. Benchmarking cloud serving systems with ycsb. In *Proceedings of the 1st ACM Symposium on Cloud Computing*, SoCC '10, pages 143–154, New York, NY, USA, 2010. ACM.

[15] Swapnil Patil, Milo Polte, Kai Ren, Wittawat Tantisiriroj, Lin Xiao, Julio López, Garth Gibson, Adam Fuchs, and Billie Rinaldi. Ycsb++: Benchmarking and performance debugging advanced features in scalable table stores. In

Proceedings of the 2Nd ACM Symposium on Cloud Computing, SOCC '11, pages 9:1–9:14, New York, NY, USA, 2011. ACM.

[16] V. Abramova, J. Bernardino, and P. Furtado. Testing cloud benchmark scalability with cassandra. In *Services (SERVICES), 2014 IEEE World Congress on*, pages 434–441, June 2014.

[17] Yahoo cloud serving benchmark(ycsb) http://labs.yahoo.com/news/yahoo-cloud-serving-benchmark, November 2015.

[18] Ycsb github repository https://github.com/brianfrankcooper/YCSB, November 2015.

[19] Your ultimate guide to the non-relational universe! (nosql)http://nosql-database.org, November 2015.

[20] Mongodb https://www.mongodb.org, November 2015.

[21] K. Chodorow and M. Dirolf. Mongodb: The definitive guide, November 2010.

[22] Mongodb https://www.mongodb.org, November 2015.

[23] Sebastian Voss. Docker and mongodb sharded cluster https://sebastian-voss.com/docker-mongodb-sharded-cluster.html, November 2015.

[24] Philipzheng. Docker——從入門到實踐 https://www.gitbook.com/book/philipzheng/docker_practice/details, November 2015.

[25] introducing json http://json.org, November 2015.

[26] 余至浩. 台積電運用大資料分析創造半導體製程技術優勢 http://www.ithome.com.tw/news/92290, September 2014.

[27] 簡禎富. 台積電以大數據提升製造智慧 http://www.hbrtaiwan.com/article_content_AR0002794.html, June 2014.

[28] 黃珣. 植基於 ycsb 系統之叢集資料庫效能分析. Master's thesis, 國立中山大學, june 2012.

[29] 周耿達. Mongodb 與 mysql cluster 效能分析. Master's thesis, 國立臺北商業技術學院, june 2014.

[30] 呂欣汶. 雲端醫療紀錄之巨量資料存取與處理平台建置. Master's thesis, 東海大學, june 2014.

[31] 洪健恆. Linux 網路應用框架之研究：以 mongodb 為例之實作. Master's thesis, 國立中山大學, July 2011.

附錄 A

安裝實驗基礎環境

1. 必要安裝的軟體套件及說明：

安裝 Java JDK/Apache Maven/Git 是使用 YCSB 先必須安裝的 Package。

原因為 YCSB 為 Java 開發，程式原始碼以及相關的 Libary 的獲取與開發必須透過 Git 與 Maven 來獲得。

安裝 SSH，則是為了我們可以遠端登入 Server 進行相關設定與使用。

安裝 Docker, 則是為了可以使用 Docker container。

2. 進行 Ubuntu 套件更新、升級以及安裝 SSH

sudo apt-get update – 套件更新

sudo apt-get -y upgrade – 套件升級

sudo apt-get -y install ssh – 安裝 SSH

3. 進行安裝 Oracle Java 8

sudo add-apt-repository ppa:webupd8team/java

sudo apt-get update

sudo apt-get install oracle-java8-installer

java -version – 確認是否安裝完成

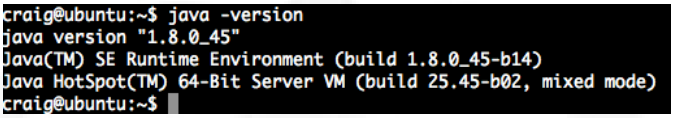

FIGURE A.1: Java 版本確認

4. 進行安裝 Git

sudo apt-get update

sudo apt-get install git

git –version

5. 進行安裝 Apache Maven

cd

wget http://ftp.heanet.ie/mirrors/www.apache.org/dist/maven/maven-3/3.1.1/binaries/apache-maven-3.1.1-bin.tar.gz

sudo tar xzf apache-maven-*-bin.tar.gz -C /usr/local

cd /usr/local

sudo ln -s apache-maven-* maven

sudo vi /etc/profile.d/maven.sh

編輯 maven.sh 檔案內容, 將下列內容寫入

export M2_HOME=/usr/local/maven

export PATH=${M2_HOME}/bin:$PATH

編輯完畢後進行重新開機

sudo reboot

mvn -version

6. 進行安裝 Docker

安裝最後一版的 Docker package

wget -qO- https://get.docker.com/ | sh

進行確認是否正確安裝 Docker

sudo docker run hello-world

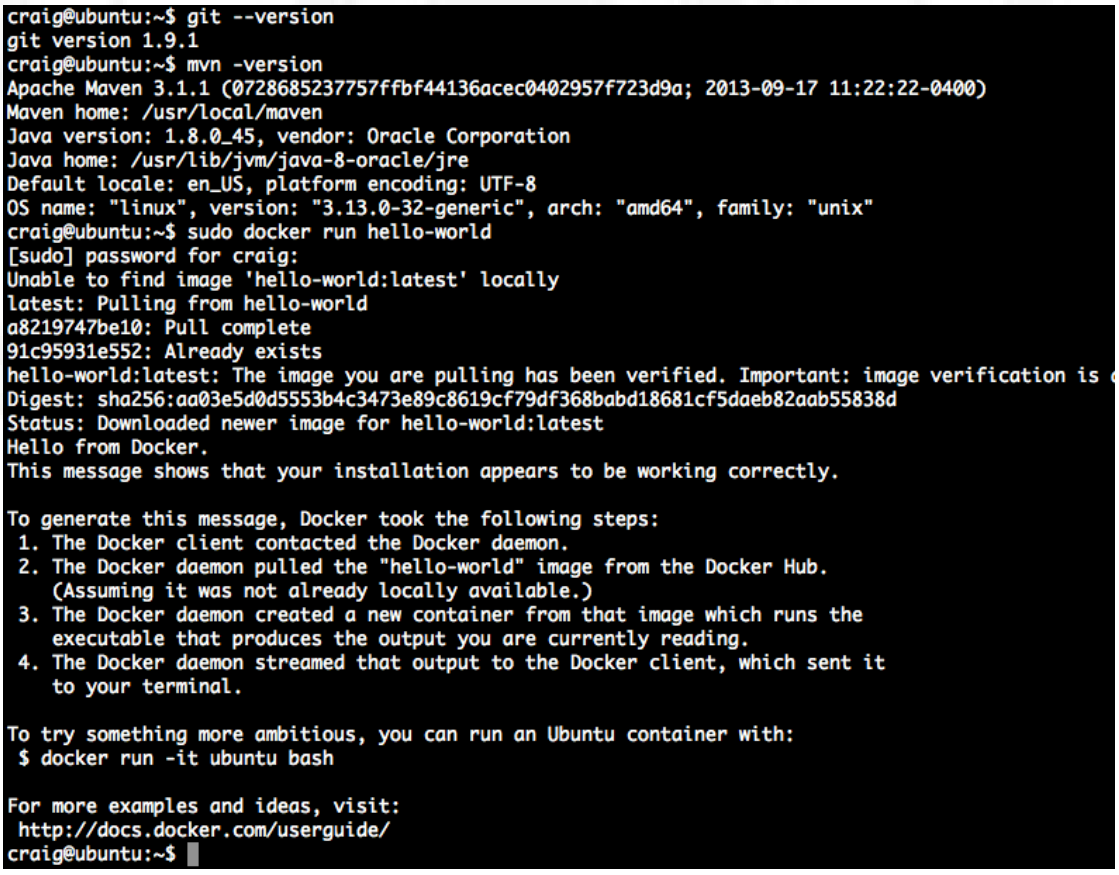

FIGURE A.2: Git / Maven / Docker 安裝結果確認

附錄 B

Docker 以及 MongoDB Sharded Cluster 安裝步驟

1. 建立 Docker files

cd

~ mkdir -p 7docker_mongodb_cluster/mongod

mkdir -p 7docker_mongodb_cluster/mongos

針對所需要的 MongoDB, 需使用不同的 MongoDB Dockerfiles

3.0.2 版本, Mongod container 需使用如圖 Figure B.1 顯示的內容

Mongod Dockerfile (MongoDB 3.0.2)

```
FROM ubuntu:latest
RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
RUN echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | tee
/etc/apt/sources.list.d/10gen.list
RUN apt-get install mongodb-10gen
RUN mkdir -p /data/db
EXPOSE 27017
ENTRYPOINT ["usr/bin/mongod"]
```

FIGURE B.1: Mongod Dockerfile (MongoDB 3.0.2

2.6.9 版本, Mongod container 需使用如圖 Figure B.2 顯示的內容

Mongod Dockerfile (MongoDB 2.6.9)

```
FROM ubuntu:latest
RUN apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
RUN echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-upstart dist 10gen' | tee
/etc/apt/sources.list.d/10gen.list
RUN apt-get update
RUN apt-get install -y mongodb-org=2.6.9 mongodb-org-server=2.6.9 mongodb-org-shell=2.6.9
mongodb-org-mongos=2.6.9 mongodb-org-tools=2.6.9
RUN echo "mongodb-org hold" | sudo dpkg --set-selections
RUN echo "mongodb-org-server hold" | sudo dpkg --set-selections
RUN echo "mongodb-org-shell hold" | sudo dpkg --set-selections
RUN echo "mongodb-org-mongos hold" | sudo dpkg --set-selections
RUN echo "mongodb-org-tools hold" | sudo dpkg --set-selections
RUN mkdir -p /data/db
EXPOSE 27017
ENTRYPOINT ["usr/bin/mongod"]
```

FIGURE B.2: Mongod Dockerfile (MongoDB 2.6.9)

然後將內容填到/docker_mongodb_cluster/mongod/Dockerfile

Mongos container 基本上市依賴上述 Mongod 建立出來的 image, 所以不需進行區分都是使用如圖 Figure B.3 顯示內容

然後將內容填到/docker_mongodb_cluster/mongos/Dockerfile

Mongos Dockerfile

```
FROM dev24/mongodb:latest
EXPOSE 27017
ENTRYPOINT ["usr/bin/mongos"]
```

FIGURE B.3: Mongos Dockerfile

2. 進行 Docker image 的建立

cd /docker_mongodb_cluster

sudo docker build -t dev24/mongodb mongod

sudo docker build -t dev24/mongos mongos

3. 建立 Replica Sets, 這邊要注意的是 2.6.9 與 3.0.2 有些微差距, 因為 3.0.2 要額外開啟 Wired Tiger

針對 MongoDB 2.6.9 的如下

sudo docker run -P -name rs1_srv1 -d dev24/mongodb –replSet rs1 –noprealloc –smallfiles

sudo docker run -P -name rs1_srv2 -d dev24/mongodb –replSet rs1 –noprealloc –smallfiles

sudo docker run -P -name rs1_srv3 -d dev24/mongodb –replSet rs1 –noprealloc –smallfiles

針對 MongoDB 3.0.2 的如下

sudo docker run -P -name rs1_srv1 -d dev24/mongodb –replSet rs1 –storageEngine wiredTiger –noprealloc –smallfiles

sudo docker run -P -name rs1_srv2 -d dev24/mongodb –replSet rs1 –storageEngine wiredTiger –noprealloc –smallfiles

sudo docker run -P -name rs1_srv3 -d dev24/mongodb –replSet rs1 –storageEngine wiredTiger –noprealloc –smallfiles

4. 初始化 Replica Sets

先查詢 Container 的 ip address

sudo docker inspect rs1_srv1 | grep IPAddress

sudo docker inspect rs1_srv2 | grep IPAddress

sudo docker inspect rs1_srv3 | grep IPAddress

假設分別獲得 ip address 如下

rs1_srv1 172.17.0.2

rs1_srv2 172.17.0.3

rs1_srv3 172.17.0.4

設定 Replica Set 1 (rs1)

A.sudo docker exec -ti rs1_srv1 bash

B.mongo

C.rs.initate()

D.rs.add("172.17.0.3:27017") 將上面查到的 ip 代入 rs.add("<IP_of_rs1_srv2>:27017")

E.rs.add("172.17.0.3:27017") 將上面查到的 ip 代入 rs.add("<IP_of_rs1_srv3>:27017")

F.cfg = rs.conf()

G.cfg.members[0].host = "172.17.0.2:27017" 將上面查到的 ip 代入 cfg.members[0].host = "<IP_of_rs1_srv1>:27017"

H.rs.reconfig(cfg)

I.rs.status()

5. 建立 Config Servers

sudo docker run -P -name cfg1 -d dev24/mongodb –noprealloc –smallfiles –configsvr –dbpath /data/db –port 27017

sudo docker run -P -name cfg2 -d dev24/mongodb –noprealloc –smallfiles –configsvr –dbpath /data/db –port 27017

sudo docker run -P -name cfg3 -d dev24/mongodb –noprealloc –smallfiles –configsvr –dbpath /data/db –port 27017

查詢 Config Server ip address

sudo docker inspect cfg1 | grep IPAddress

假設獲得的 ip address 如下：

cfg1 172.17.0.13

cfg2 172.17.0.14

cfg3 172.17.0.15

6. 建立 Router Server

代入上面查得的 Config Server IP

sudo docker run -d -p 27017:27017 -name mongos1 -d dev24/mongos –port 27017 –configdb 172.17.0.13:27017,172.17.0.14:27017,172.17.0.15:27017

7. 初始化 Shard Server A.sudo docker exec -ti monogs1 bash

B.mongo

C.sh.addShard(”rs1/172.17.0.2:27017”) 帶入上面 rs1_srv1 的 IP 到 sh.addShard(”rs1/<IP_of_rs1_srv1>:27017”)

D.sh.addShard(”rs2/<IP_of_rs2_srv1>:27017”) 帶入 rs2_srv1 的 IP

E.sh.addShard(”rs3/<IP_of_rs3_srv1>:27017”) 帶入 rs3_srv1 的 IP

F.sh.addShard(”rs4/<IP_of_rs4_srv1>:27017”) 帶入 rs4_srv1 的 IP

G.sh.enableSharding(”ycsb”)

H.sh.shardCollection(”ycsb.usertable”, _id:”hashed”)

I.sh.status() 確認 Shard 設定結果

FIGURE B.4: Replica Set 設定結果確認

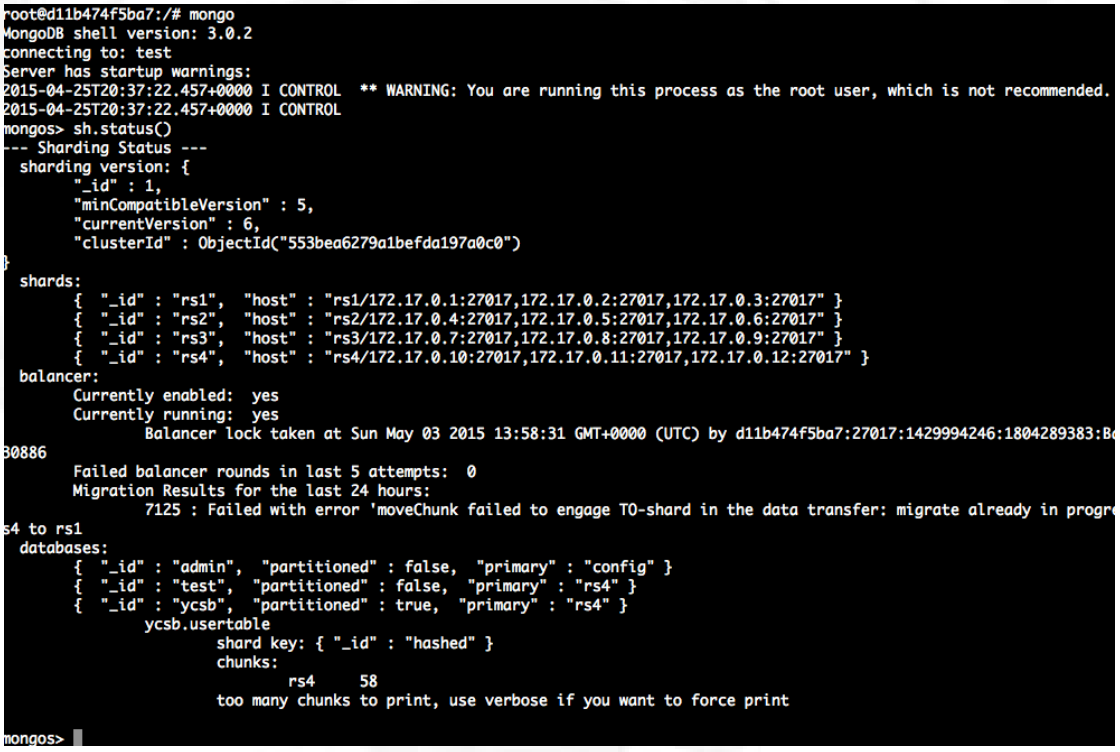

FIGURE B.5: Shard 設定結果確認

附錄 C

YCSB 下載安裝步驟

下載可以測試 async 的 YCSB for MongoDB

cd $HOME

git clone https://github.com/allanbank/YCSB.git

mv $HOME/YCSB /async-YCSB

下載一般版本的 YCSB for MongoDB

git clone https://github.com/10gen-labs/YCSB.git

cd $HOME/async-YCSB/mongodb/src/main/java/com/yahoo/ycsb/db

cp MongoDbClient.java 7YCSB/ycsb-mongodb/mongodb/src/main/java/com/yahoo/ycsb/db

```
craig@ubuntu:~/YCSB/ycsb-mongodb/mongodb/src/main/java/com/yahoo/ycsb/db$ ll
total 48
drwxrwxr-x 2 craig craig  4096 Apr 25 15:51 ./
drwxrwxr-x 3 craig craig  4096 Apr 25 15:47 ../
-rw-rw-r-- 1 craig craig 18967 Apr 25 15:51 AsyncMongoDbClient.java
-rw-rw-r-- 1 craig craig 16747 Apr 25 15:47 MongoDbClient.java
craig@ubuntu:~/YCSB/ycsb-mongodb/mongodb/src/main/java/com/yahoo/ycsb/db$ pwd
/home/craig/YCSB/ycsb-mongodb/mongodb/src/main/java/com/yahoo/ycsb/db
craig@ubuntu:~/YCSB/ycsb-mongodb/mongodb/src/main/java/com/yahoo/ycsb/db$
```

FIGURE C.1: Async MongoDB Client 放置處

修改 POM.XML 一 vim $HOME/YCSB/ycsb-mongodb/pom.xml

修改 POM.XML 二 vim $HOME/YCSB/ycsb-mongodb/mongodb/pom.xml

FIGURE C.2: pom.xml 修改內容一

修改 ycsb 增加支援 async vim $HOME/YCSB/ycsb-mongodb/bin/ycsb

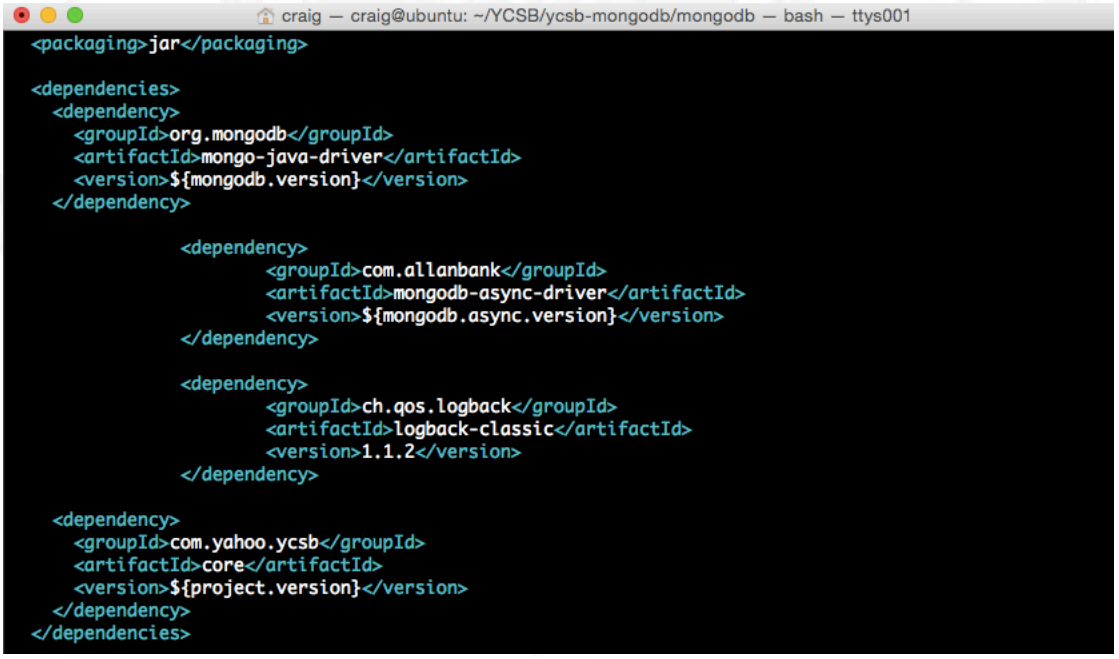

FIGURE C.3: pom.xml 修改內容二

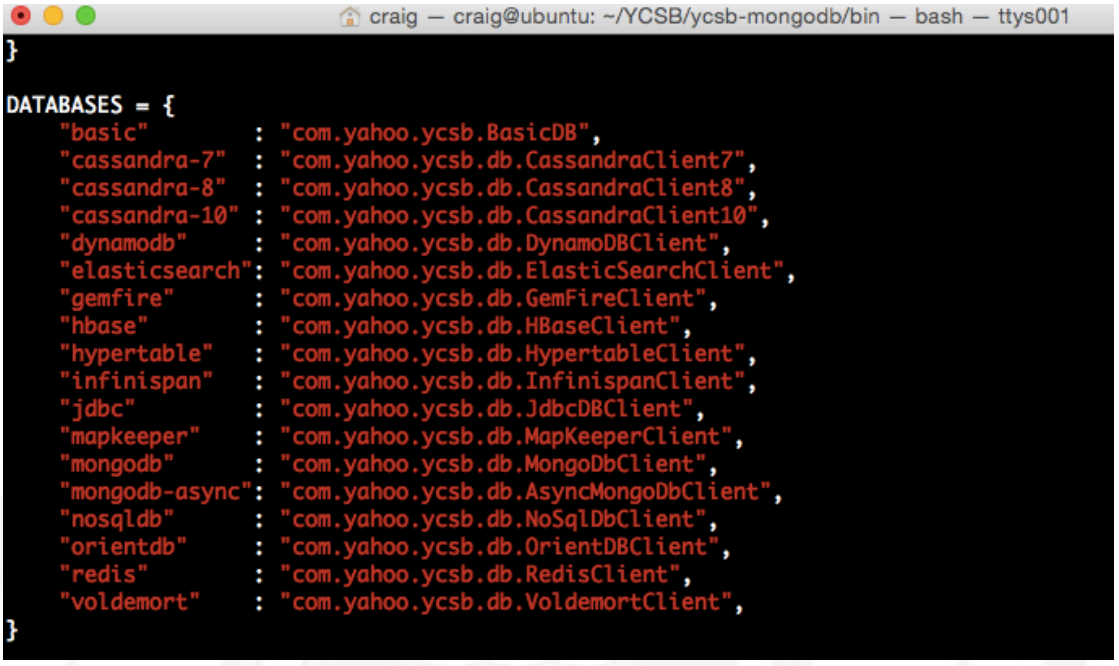

FIGURE C.4: ycsb 啟動程式修改內容

將相關的 workload 檔案放到 $HOME/YCSB/ycsb-mongodb/workload 目錄

程式碼的編譯與引用的 library 下載

cd $HOME/YCSB/ycsb-mongodb

mvn -pl com.yahoo.ycsb:core,com.yahoo.ycsb:mongodb-binding clean package

接下來就可以使用以下指令去進行測試

./bin/ycsb load mongodb -P workload/workloada -s > output/outputload.txt

./bin/ycsb run mongodb-async -s -P workload/workloada > output/asyncoutputRuna.txt

./bin/ycsb run mongodb -P workload/workloada -s > output/outputRuna.txt