國立高雄大學資訊工程學系

碩士論文

基於微服務架構的智慧管理系統之設計

Design of Intelligent Management System Based on Microservice Architecture

研 究 生： 張俊彥

指導教授： 張保榮 教授

中華民國一百一十一年十二月

基於微服務架構的智慧管理系統之設計

指導教授：張保榮 教授
國立高雄大學資訊工程學系

研究生：張俊彥
國立高雄大學資訊工程學系碩士在職專班

摘要

在資訊科技快速變動的時代，數位轉型已成為企業不可忽視的課題，數位轉型需處理複雜的數據資料，雲原生平台的微服務架構技術是主流的應用。本研究以微服務架構為基礎，運用容器虛擬化技術，設計智慧管理系統，在不影響現有系統運作的條件前提下，使用有限的硬體資源作為規劃，架設 Proxmox VE 建立 Kernel-based Virtual Machine 叢集，並使用 Kubernetes 為容器管理平台，提供高可靠度和即時遷移的相關應用服務，並透過現有資料庫資料整合及分析，實現數據指標呈現、資料統計與服務監控的功能。

關鍵字：微服務、Kubernetes、容器、智慧管理系統、雲原生、數位轉型。

Design of Intelligent Management System Based on Microservice Architecture

Advisor: Dr. Bao Rong Chang
Department of Computer Science and Information Engineering
National University of Kaohsiung

Student: Chun-Yen Chang
Department of Computer Science and Information Engineering
National University of Kaohsiung

ABSTRACT

In the era of rapid changes in information technology, digital transformation has become a topic that cannot be ignored by enterprises. Digital transformation needs to process complex data, and the microservice architecture technology of cloud-native platforms is the mainstream application. Based on the microservice architecture, this research uses container virtualization technology to design a intelligent management system. Under the premise of not affecting the operation of the existing system, using limited hardware resources as a plan, set up Proxmox VE to establish a Kernel-based Virtual Machine cluster , and use Kubernetes as the container management platform to provide high reliability and real-time migration related application services, and through the integration and analysis of existing database data, the functions of data indicator presentation, data statistics and service monitoring are realized.

Keywords:Microservices, Kubernetes, Container, Intelligent Management System, Cloud Native,Digital Transformation.

誌謝

轉眼間，我的研究所生涯將告一段落，而入學彷彿還是昨天的事，初來乍到時的場景猶歷歷在目。本論文能順利完成，首先誠摯感謝我的指導教授張保榮博士，老師的悉心教導，在論文的研究方向及內容撰寫過程中，為我指點迷津，幫助我開拓研究思路，並給予大力指導與協助，使我能克服各種困難順利完成學習任務。

此外，老師嚴謹治學的態度、豐富的知識、宏觀的眼界、誨人不倦的精神都讓我留下了深刻的印象，是我學習的榜樣。求學期間經歷了工作上的轉換及學術研究的迷茫，老師總是能提供適時的建議與指導，使我無論在心態上、學識上都能有所成長，讓我受益匪淺，在此致上深深的敬意及感謝。

最後，我要感謝培養我長大含辛茹苦的父母，謝謝您們的悉心栽培，支持鼓勵我繼續完成碩士學位。我要感謝我的太太，在我的求學生涯中，全心全意為家庭奉獻，給予我極大的支持與包容，使我可以無後顧之憂的投入於研究之中。我還要謝謝我兩個可愛的女兒，妳們是我完成艱辛困難任務的動力來源。

目錄

圖目錄

表目錄

第一章 緒論

近年來企業數位轉型已成為顯學，尤其在 2020 年 COVID-19 世紀疫情爆發後，疫情期間大家生活型態及工作型態轉變所帶來的衝擊，更加速了各國產業推動數位轉型的腳步。雲原生平台是數位專案的重要核心，微服務架構不但可以解決傳統實體機器設定管理的運維問題，經由統一自動化的部署建置、升級、管理機制，高可用度的設計大幅降低硬體設備及系統運維的成本，也有助於應用軟體服務的開發速度。對於一般企業來說，資訊系統常遇到的問題，不外乎是系統太多帳號密碼難以管理、系統資料無法互通及系統老舊與實際需求不符、新舊系統環境不相容等問題，本研究所設計之智慧管理系統，除了希望管理者能夠更有效率的監控各系統運作，經由各系統蒐集而來的資料進行分析及研究，用以增進營運效益，並制定標準 API，保留未來新建系統介接的擴充性，幫助企業循序漸進的完成數位轉型。

本研究目標在基於微服務架構基礎上，設計一智慧管理系統，內容包含新建置防火牆及交換器，整合現有網路架構、.使用現有硬體伺服器，架設 Proxmox VE 建立虛擬機叢集，並使用 Kubenetes 為容器管理調度平台、整合既有前台及外部系統資料庫，以提供指標之運算與應用、.設計標準應用程式介面服務功能，建置一標準 API，介接現有系統功能、建置資料指標監控儀表板管理平台，內容包含指標呈現、統計資訊與服務監控功能以及預留新系統擴充開發空間。

本章旨在說明學習背景、學習動機和學習目的。 第二章將討論相關的文獻和技術。 在第三章中將詳細解釋本文使用的技術和方法。 第四章將提供實驗數據和相關實驗結果。 最後，第五章將對論文進行總結，並提出未來改進和功能擴展的可能性。

第二章 相關研究

2.1 雲原生

2013 年 Pivotal 公司的 Matt Stine 首次提出雲原生的概念。2015 年由 Goole 主導成立的雲端原生運算基金會(CNCF)對於雲原生的定義包含以下三個方面的技術運用:應用容器化、微服務架構、可動態編排式容器應用。隨著該生態系不斷的發展及各大雲端主機供應商加入 CNCF 後，於 2018 年重新定義: 雲端原生技術可讓組織在現代化、動態環境中建置和執行可調整的應用程式，例如公用、私人和混合式雲端。 容器、服務網格、微服務、不可變的基礎設施，以及宣告式 API 會示範此方法[1]。

2.2 微服務

2014 年，Martin Fowler 與 James Lewis 共同提出了微服務的概念[2]，定義了微服務是由以單一應用程式構成的小服務，自己擁有自己的行程與輕量化處理，服務依業務功能設計，以全自動的方式部署，與其他服務使用 HTTP API 通訊。同時服務會使用最小的規模的集中管理 (例如 Docker) 能力,服務可以用不同的程式語言與資料庫等元件實作。微服務架構是一種應用程式架構，在這種架構之下，由許多服務會一起運作而構成整個應用程式，這些服務為鬆散耦合狀態，它們可以獨立部署、測試、建構及擴展，並透過定義好的標準化應用程式介面 API 進行通訊，各自負責單一任務，分層式獨立區塊而不互相影響[3]，在和客戶端也能使用 API 閘道，提供更多的安全驗證[4]。

微服務可以透過容器化技術將應用程式部署在容器中，並使用 Kubernetes 來有效率管理調度容器，結合 DevOps 與持續整合/持續交付(CI/CD)達到快速開發的目的[5]。

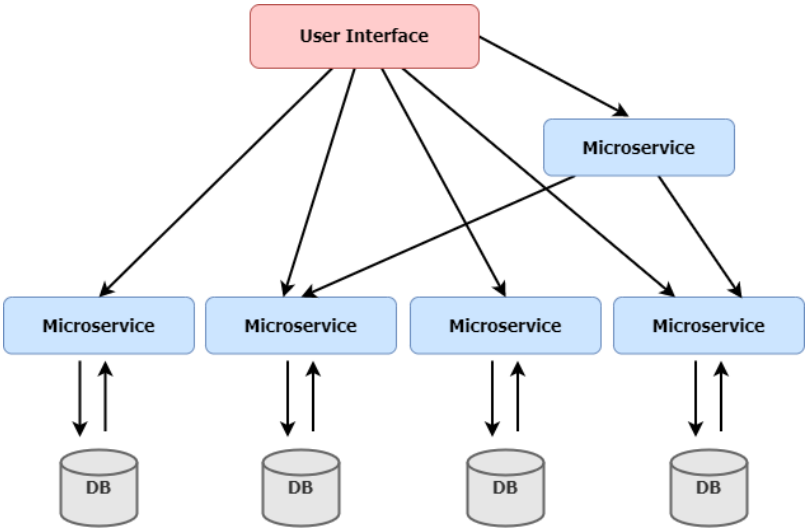

Figure 1. 微服務架構

2.3 Kubernetes

Kubernetes 是一個可以用自動化部署、擴充、管理多個容器化應用程式及服務叢集的開源系統，由 Google 工程師 Joe Beda、Brendan Burns 和 Craig McLuckie 所開發，並於 2014 年發表，前身是 Borg 系統，因為名稱的 K 與 s 之間有 8 個英文字母，所以又被稱為 K8s[6]。

Kubernetes 可以執行容器自動部署、服務自動擴展及自動化管理，也具備負載平衡和完整備份還原的災難復原機制[7]。K8s 架構主要由 Cluster、Node 與 Control Plane 所組成。Control Plane 是指揮中心，扮演大腦的角色，負責管理集群內的所有的節點，透過 kubectl 指令呼叫 API，與各個元件及節點間的行為透過 API Server 互相溝通。Controller Manager 負責監控集群運作的狀態。Scheduler 負責集群的資源調配，確保工作負載不會超過可用資源 c 提供負載均衡的機制。etdc 是負責存放紀錄集群運行狀態的資料檔，屬於分散式 key-value 方式儲存，並確保資料永久性，當發生異常時，可透過 etdc 還原狀態。Node 可以是 VM 或實體機，Pod 是抽象層，是節點裡面的最小單位，也是放置容器服務的地方。kubelet 是 Node 元件中，負責建構 Pod 及維護容器的工作。Kube-proxy 主要負責網路代理，提供 Pod 之間溝通的管道。Cloud-controller-manager 負責連接到雲端服務商的 API，操作雲端資源的控制。

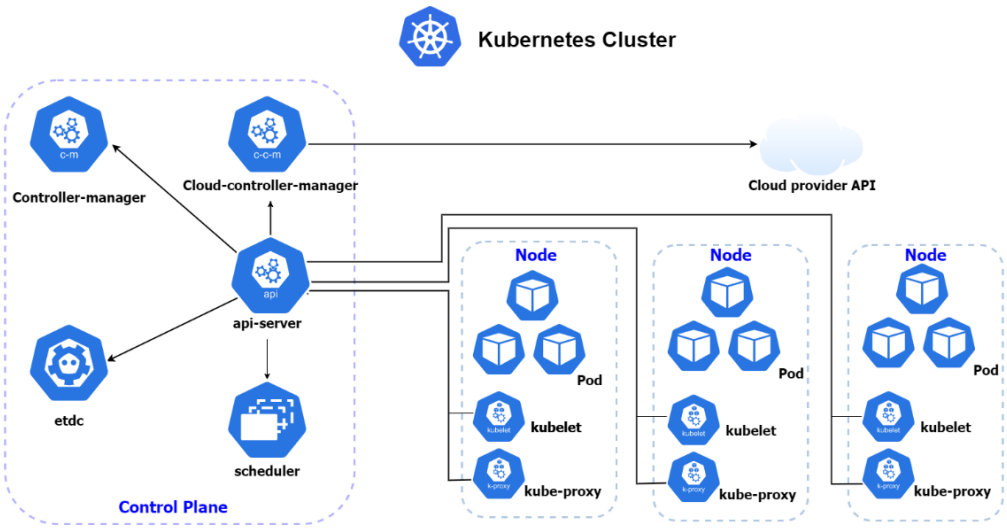

Figure 2. Kubernetes 架構

2.4 容器

Container 容器是一個可提供應用程式掛載及所需資源的空間，並可以將其程式碼及所依賴的資源項目打包在一起，使應用程式可以快速地運行到不同環境中。和傳統虛擬機比較，容器實現了作業系統虛擬化，具有輕量化、可攜度高、獨立隔離性、啟動速度快、消耗資源少、便於安裝等優點[8]。Docker 是目前最著名的容器化軟體開源平台，可以讓開發人員快速的建立、測試和部署應用程式[9]。

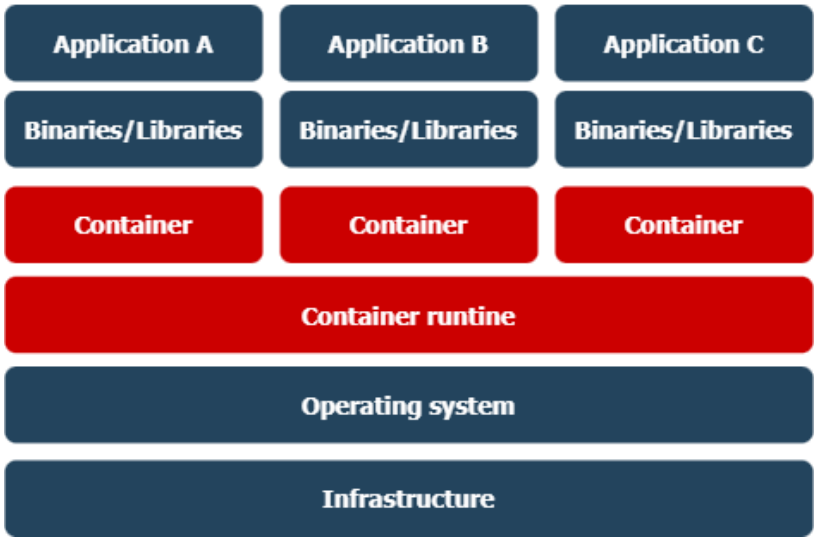

Figure 3. 容器架構

第三章 研究方法

3.1 目標與範圍

本研究智慧管理系統之設計，以現有網路環境資源為背景，使用既有三台伺服器，擴充記憶體及硬碟空間，並增加既有同型號防火牆與交換器各一台，與既有環境整合為 Full HA 架構，建置微服務架構，新建置之 Oracle 資料庫整合既有各系統的資料庫，並透過各種技術組合，開發 API 程式進行資料交換，呈現各項公司營運指標，管理者能用網頁儀表板觀看並查詢想要的資料，也保留未來新系統的擴充空間，新開發的系統可經由設定，快速使用此架構上的資源，達到快速部署的目地，包含以下內容:

- 新建置防火牆及交換器，整合現有網路架構
- 使用現有硬體伺服器，架設 Proxmox VE 建立虛擬機叢集，並使用 Kubenetes 為容器管理調度平台
- 整合既有前台及外部系統資料庫，以提供指標之運算與應用
- 設計標準應用程式介面服務功能，建置一標準 API，介接現有系統功能
- 建置資料指標監控儀表板管理平台，內容包含指標呈現、統計資訊與服務監控功能
- 預留新系統擴充開發空間

3.2 網路架構設計

本研究整體網路架構如 Figure 4 所示

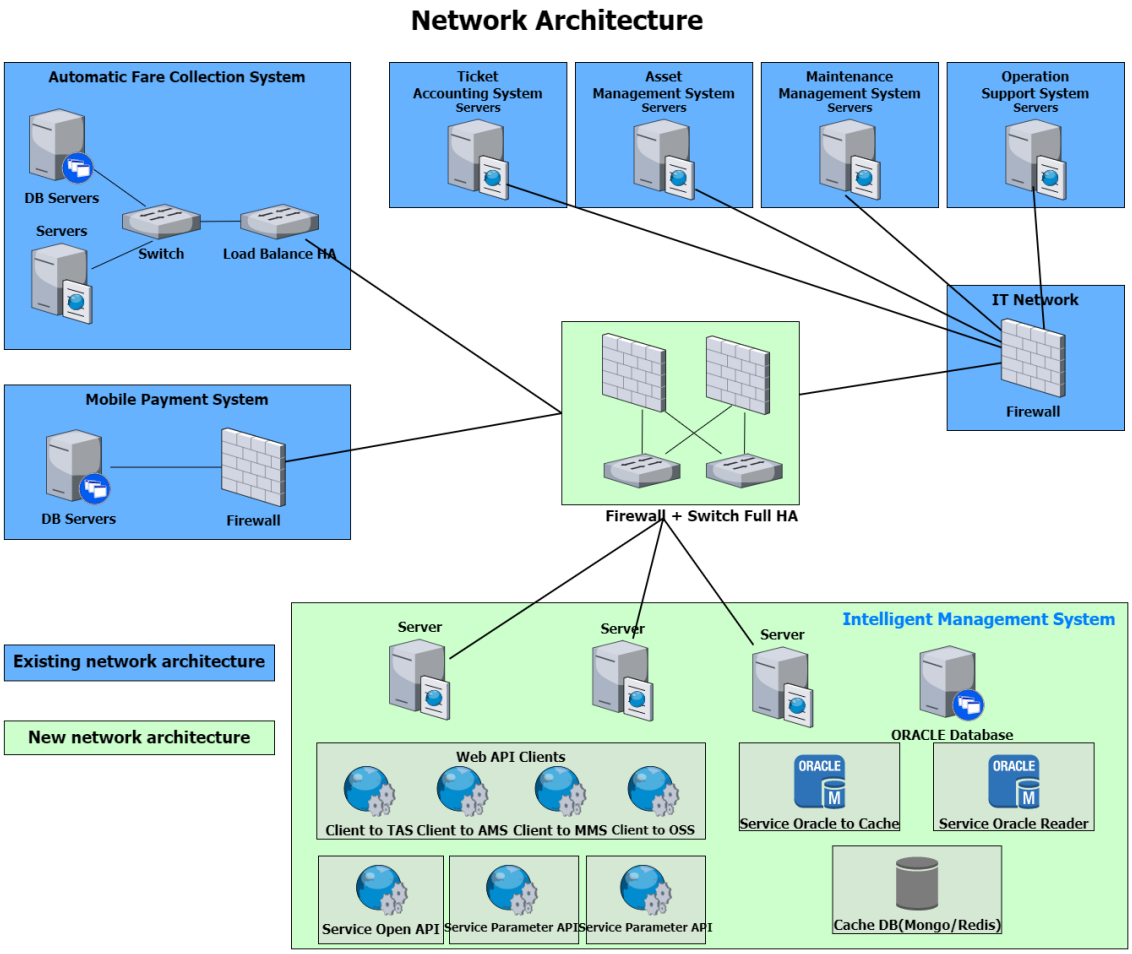

Figure 5. 網路架構

增加既有同型號防火牆與交換器各一台，與既有環境整合為 Full HA 架構，並向外串接各系統介面，並透過防火牆設定 NAT 隔離避免架構資訊揭露，再加上 Policy 僅允許必要之存取流量通過，確保資訊安全。

針對既有三台 HPE 伺服器上四個 1000Base-T 網路埠，以兩埠為一組進行合併（每埠介接至不同網路交換器上）設定成兩組 Bond，並配合交換器端設定為 LACP 或 Active/Passive(A/P)運行模式，以提供交換器或單一網路埠故障下的容錯(Fault Tolerance)機制。

3.3 系統硬體架構設計

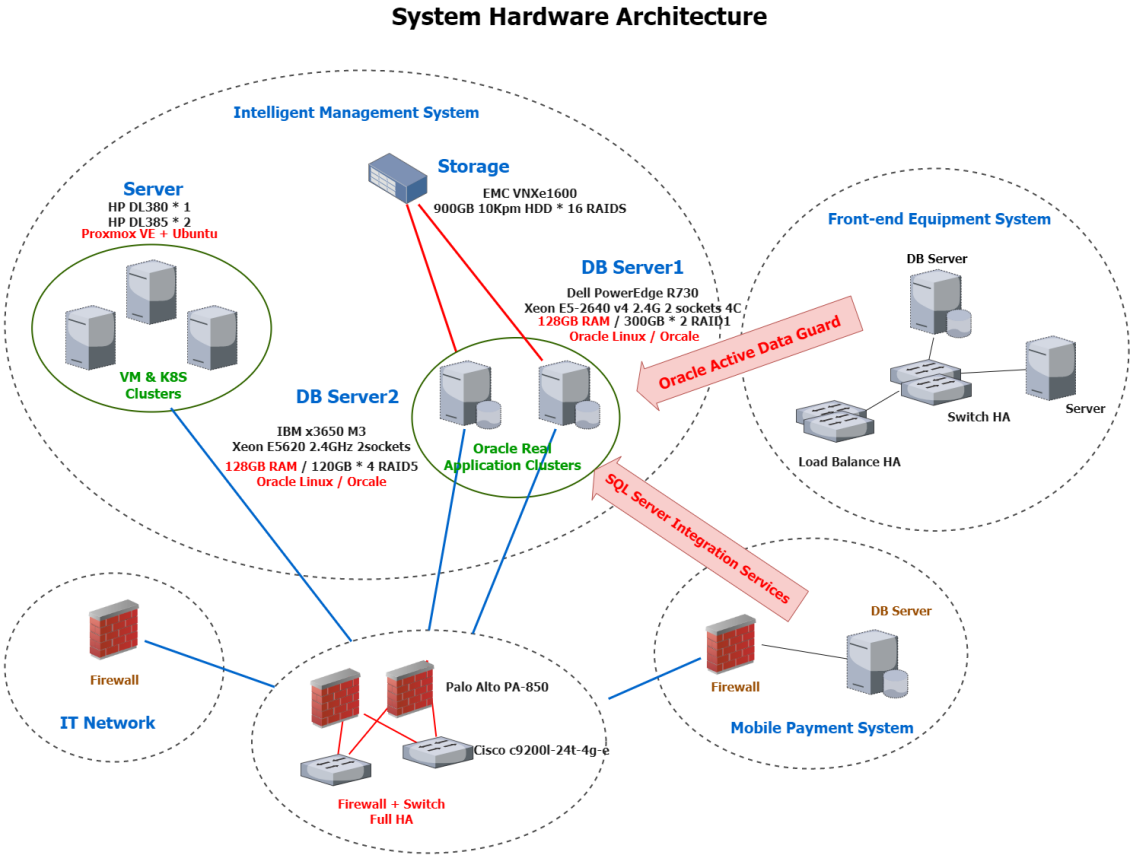

Figure 6. 系統硬體架構

運用現行既有三台 HPE 伺服器，架設 Proxmox Virtual Environment 建立 Kernel-based Virtual Machine 叢集，為安裝 Kubernetes(K8s)環境之 Linux 虛擬機提供高可靠度(High-Availability)和即時遷移(Live Migrate)相關應用服務[10]。

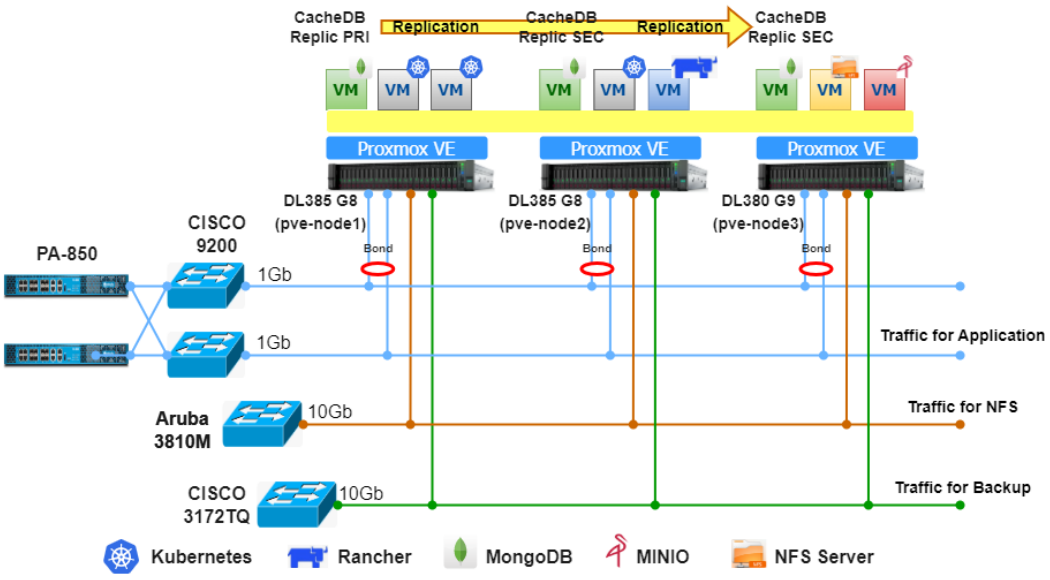

Figure 7. 虛擬平台系統

3.3.1 儲存空間架構

- 將兩台 HPE DL385 G8 伺服器內接硬碟設定為 RAID 1 提供容錯儲存空間。
- 將 HPE DL380 G9 伺服器內接硬碟設定為 RAID 6 以提供容錯儲存空間，並以 NFS export 提供服務。
- NFS export 主要擔任 2 台 HPE DL385 G8 伺服器之網路共享(NFS share)。

儲存空間設計如 Figure 7 所示：

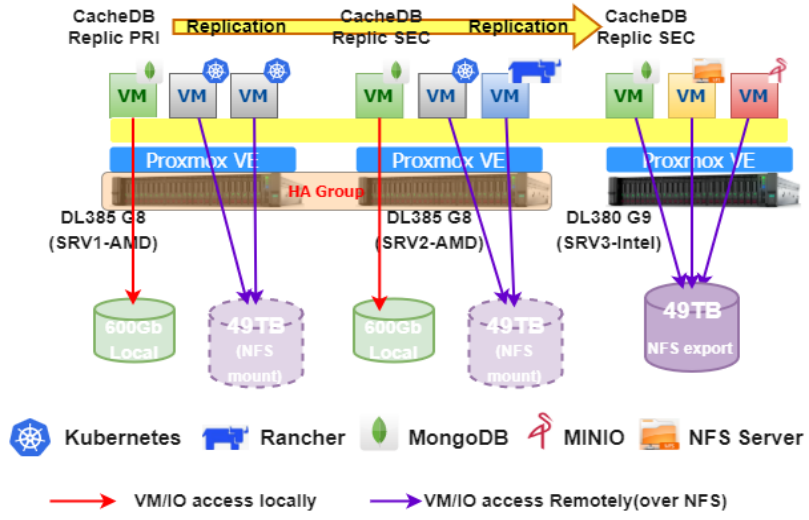

Figure 8. 儲存空間

3.3.2 網路容錯架構

- 針對既有三台 HPE 伺服器上四個 1000Base-T 網路埠，其中兩埠為一組進行合併（每埠介接至不同網路交換器上）設定成 Bond，並配合交換器端設定為 LACP 或 Active/Passive(A/P)運行模式，以提供交換器或單一網路埠故障下的容錯(Fault Tolerance)機制；擔任 Application data traffic 之用。
- 當配合交換器端設為 LACP 時，單一 Bond 線路總頻寬將提升為 2Gb/s。
- 當純粹設定為 A/P 模式時，單一 Bond 線路總頻寬將維持 1Gb/s。
- 將針對既有三台 HPE 伺服器上擴充一個 10Gb 網路埠，並介接至 Aruba 3810M 交換器，擔任三台伺服器間 NFS traffic 之用（Private LAN）。
- 針對既有三台 HPE 伺服器上四個 1000Base-T 網路埠，其中一埠介接至 CISCO 3172TQ 交換器，提供 Veritas Appliance 以 NFS export 讀取儲存於 DL380 G9 上之所有 VM 備份映像檔，擔任 Backup traffic 之用。
- 針對既有三台 HPE 伺服器上四個 1000Base-T 網路埠，其中一埠保留以備未來需求使用。

3.3.3 虛擬機容錯架構

- 將既有兩台 HPE DL385 G8 伺服器設定為 HA Group。
- HA Group 主要針對儲存於網路共享儲存空間(NFS Share)上之 K8s 環境 Linux 虛擬機提供高可靠度(High-Availability)機制[11]；當任一台 HPE DL385 G8 伺服器未預警停止服務時，該伺服器上相關之 K8s 環境 Linux 虛擬機將被轉移至仍然存活的另一台 HPE DL385 G8 伺服器上重新啟動運行。

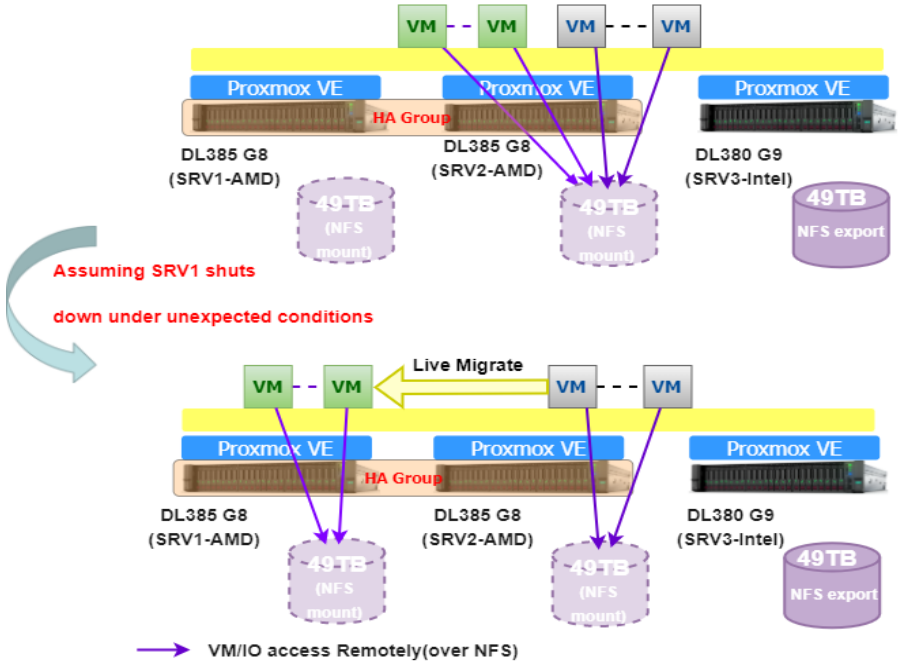

Figure 9. 虛擬機 Live Migrate 設計

- 針對儲存於三台 HPE 伺服器本機儲存空間(Local Volume)內提供 MongoDB 服務之 Linux 虛擬機，由 MongoDB Replica 提供高可靠度機制。

3.3.4 備份機制架構

將 Backup Client 安裝於各 Linux 虛擬機內，透過原架構提供之本研究獨立 VLAN 串接至 Figure 6.所述之 CISCO3172TQ 交換器，應用既有 Veritas Netbackup 5250 Appliance 針對 Linux 作業系統、Database 以及 Application 執行備份及還原作業。

3.4 軟體功能架構設計

軟體建置微服務之容器所採用作業系統預計預設為 Ubuntu 20.04 以上版本，資料庫則將採用 MongoDB /Redis(NoSQL)及 Oracle。整體架構使用前後端分離，核心程式部分預計採用.Net 6/EF Core 6，前端網頁預計使用 Vue.js 開發。

本系統軟體架構圖及系統資料流程圖如 Figure 9：

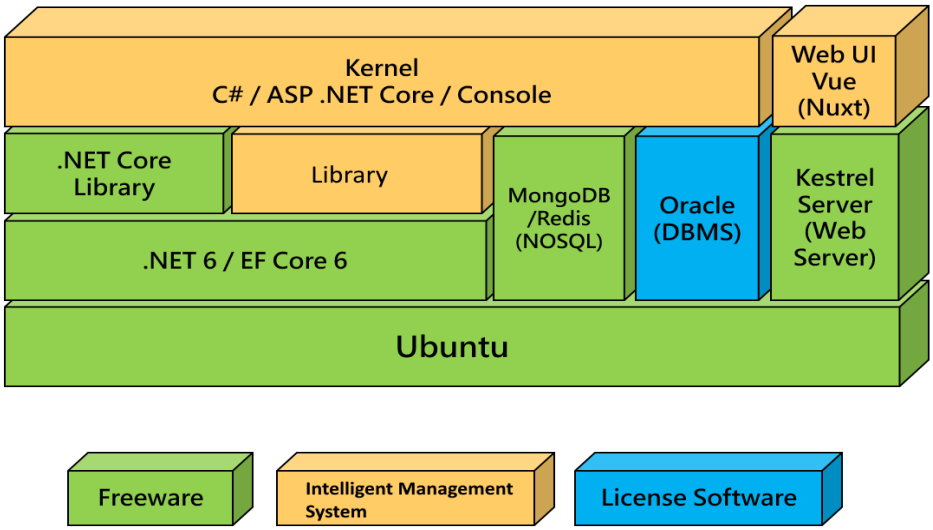

Figure 10. 軟體架構

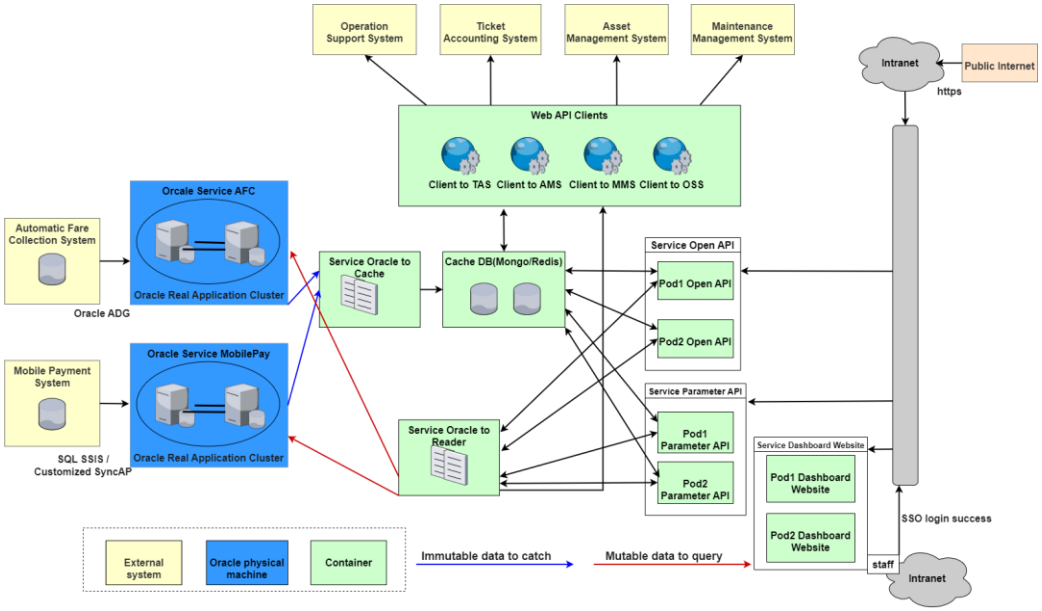

Figure 11.系統資料流程 (System Dataflow)

3.4.1 微服務架構

微服務環境的基礎建設方面，以 PVE(Proxmox Virtual Environment)建立 Kernel-based Virtual Machine 叢集作為安裝 Kubernetes(K8s)環境，並在 Kubernetes 的系統上管理部署的容器化微服務。其中 Kurbernetes 採用 3 台虛擬機建立 HA 叢集服務，並透過重要元件在多個節點各執行一組(通常最小為 3 個主節點)，因此有主節點故障時，就會有正常的主節點來確保叢集的運作。

另 MongoDB 使用 3 台虛擬機建置叢集服務，而 Rancher 因為管理 Kurbernetes

使用故採獨立 VM 建置，MINIO、NFS Server 因儲存設定管理需求，採獨立 VM 建置。

Table 1. 虛擬機用途說明

項次	安裝服務	數量	用途
1.	Kubernetes	3 (1 組叢集)	Data Reader, API Server, Web Server, …等服務
2.	MongoDB	3 (1 組叢集)	MongoDB Cluster
3.	Rancher, DNS Server	1	Kubernetes 管理工具，並安裝 DNS 服務
4.	MINIO	1	MINIO 檔案儲存系統，可提供安裝其他服務
5.	NFS Server	1	網路檔案系統

各微服務功能說明如下：

1. Oracle to Cache

   外部資料庫彙整至本系統建置之 Oracle DB 後，處理將資料庫中不可變資料(如交易資料、每日交易紀錄彙整、每月交易紀錄彙整)同步至 Cache DB。透過自行開發元件，以 Oracle Data Provider for .NET (ODP.NET) 串接 Oracle Database，建立其應用程式自訂命令列介面（Command Line Interface，CLI）提供運行環境的排程機制定期執行相關工作，將 Oracle 資料庫交易資料及相關聚合分析資料同步至 Cache DB。

2. Oracle Reader

   服務提供本系統建置之 Oracle DB 中可變資料統一查詢介面。透過自行開發元件，以 Oracle Data Provider for .NET (ODP.NET) 串接 Oracle Database，並使用 gRPC 當統一的 Oracle 資料介接介面提供各系統取得資料庫即時資料。

3. Web API Client

   整合運務支援系統、票務帳務系統、帳務管理系統、維修帳務系統，以供本系統進行資料萃取，因應四個異質系統，分別開發 WebAPI Client 程式，將其資料存放於 Cache DB，以便 OpenAPI 及儀表板作應用。

4. Open API

服務提供外部服務的查詢介面，並搭配 Kubernetes 的 Ingress 作分流。

5. Parameter API

   服務為提供本系統儀表板網頁的數據資料來源。

6. Dashboard Website

   架置網站 Server Nginx 搭配雲端前端網頁資源，供專案網路限制性存取。

7. Rancher

   作為 Kubernetes 叢集管理。

8. MongoDB

   提供 Oracle to Cache、Web API Client、Open API、Parameter API 等應用程式服務存放資訊之資料庫。

9. MINIO

   作為系統報表(Excel)或其他靜態檔案儲存媒介。

10. NFS Server

    NFS(Network File System)是一個 RPC Service，使檔案能夠共享，而 NFS 的設計是為了在不同的系統、不同的機器都可透過網路的方式而使用共享的檔案。

3.4.2 Single Sign-On(SSO)

使用者須登入內部 Portal，點選智慧管理系統連結，此時將 Portal 回傳使用者 Token 以網址參數方式代入智慧管理系統，後台即刻詢問內部 Portal Token 有效性，確認有效即會載入該使用者在智慧管理系統的相關權限及使用者資料，封裝成 JWT 傳至使用者瀏覽器，登入期間系統服務會依據 JWT 作功能權限驗證。

3.4.3 系統監控與告警

本系統使用 Prometheus 作為服務監控與異常告警應用。Prometheus 是一套開源的系統監控與告警工具，搭配 Grafana 視覺化工具，將所收集的數據用儀

表板呈現，透過 kube-state-metrics 向 api server 詢問 kubernete 內的狀態，並設定 node exporter 可以用來監控 kubernetes 叢集內 Node 的運作狀況。

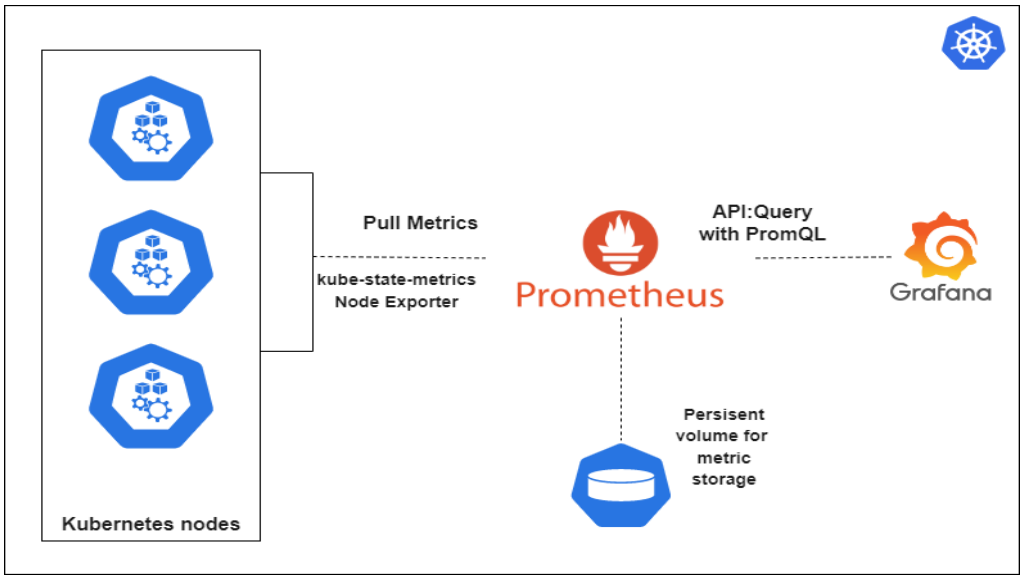

Figure 12. Prometheus 架構

3.4.4 日誌管理

本系統除採本地日誌配合定期清理機制保存系統日誌資訊，亦可透過 filebeat 或 fluentd 套件將日誌資訊推送至原系統自建 ELK 服務。

3.5 系統功能設計

3.5.1 使用者介面

本系統儀表板設計採用 Vuetify UI 框架設計，Vuetify 是一個基於 Vue 的 CSS 框架，風格依循 Google Material Design 開發，藉由該框架下之 UI 組件庫可創建重複使用的簡潔模組，並可透過這些模組實現了本系統指標圖形化目標。儀表板 UI 設計畫面如 Figure 13。

Figure 14. Vuetify UI 介面設計

Table 2. 儀錶板顯示列表

標的名稱	功能說明
參數未進版設備	儀表板顯示參數未進版之設備。顯示資訊須含： 1. 設備編號 2. 最新版本 3. 當前設備版本
離線設備	儀表板顯示離線設備。顯示資訊須含： 1. 設備編號 2. 前次連線時間

3.5.2 資料庫同步

1. Oracle RAC

Oracle Real Application Cluster (RAC) 是「共用一切」的資料庫架構，在這個架構中，兩個或更多 Oracle RAC 節點會叢集起來，共用相同的儲存體。 RAC 節點會通過高速交互連接而連接起來，以便在 Oracle 節點之間進行高速通訊。這些節點可以在啟動期間交換各種資料區塊所有權資訊、鎖定資訊、交換交易資訊和資料。

本案建置 Oracle RAC 環境依循下列項目：

(1) 安裝兩台 Linux 伺服器，並設定連接同一台儲存設備且能讀取同一儲存空間。

(2) 分別於兩台 Linux 伺服器中安裝並設定資料庫 RAC 連線開發環境 (RAC node1、RAC node2)。

(3) 建立一套 11g 版本的 Oracle 資料庫，並具備高可用性(High availability)及未來可擴充性(Scalability)的架構。

(4) 未來資料庫有補丁(Patch)需要安裝時，能有最小的服務中斷時間。

(5) Oracle Active Data Guard 11g(ADG)連線建置。

(6) 設定新建置的 Oracle 資料庫(即建置之 Oracle 11g RAC)為備援資料庫 (Standby Database)。

2. Oracle ADG

系統是一種具有高可用性的災變復原(Disaster Recovery, DR)解決方案，由一個主要資料庫(Primary Database)提供服務，且同時維護並監控一個或多個備援資料庫(Standby Database)，備援資料庫透過持續更新 Redo 資訊與主要資料庫保持同步。

藉由提供資料保護和使用狀態的進階功能，以及從實際環境執行資料庫卸載唯讀工作負載和快速增量備份功能。

3. SQL to Oracle

欲移轉至本案新增之 Oracle 資料庫內，擬使用 SQL Server 本身的 SSIS (SQL Server Integration Services)技術，透過 SQL Server 本身之設定，將其目的地指定為新建 Oracle 資料庫。

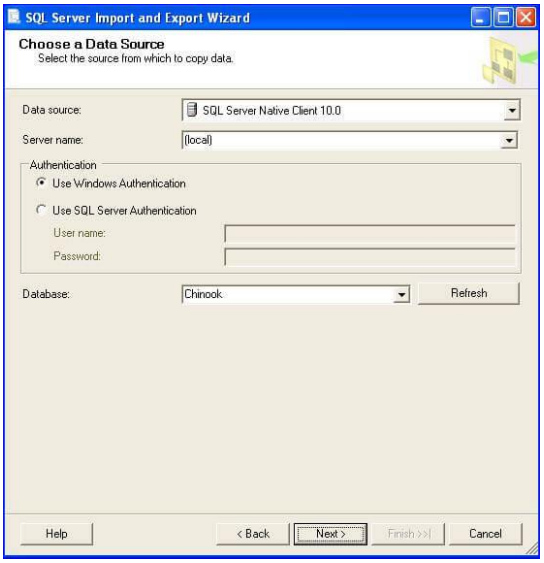

Figure 15. SQL Server SSIS 設定界面

3.5.3 功能說明

本系統提供五大類管理指標，透過整合前台設備系統、行動支付系統之交易資料，搭配串接票務帳務系統(TAS)、帳務管理系統(AMS)與維修管理系統(MMS)，提供各關鍵指標運算，以利有效且即時整合子系統相關資料。系統主要項目含：

1. API 服務：與內/外部系統(TAS 系統、AMS 系統及 MMS 系統)資料串接入口，並具備告警資訊推播給外部系統。

2. 服務指標運算：計算報表、告警資訊及計算外部系統所需資料，資料來源為 Oracle 資料庫及 API 服務，查詢資料儲存需求需建立快取資料庫。

3. 監控儀表版：使用者圖形化介面(UI)供使用者操作，且具備帳號管理、報表產出、 資料查詢及告警/監控項目設定。

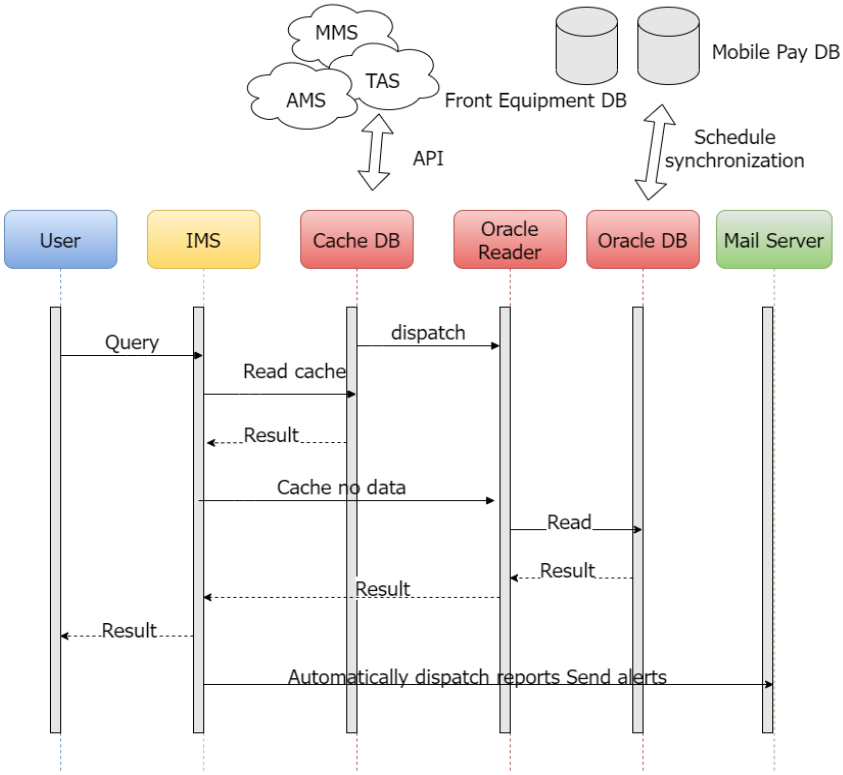

Figure 16. 系統工作流程 (System Workflow)

A. 交易運量指標

系統設計交易運量指標提供交易查詢、系統資訊、運量分析、銷售分析與產製營運日(月)報表功能，作為行銷規劃與營運決策使用。

Figure 17. 交易運量指標清單

1. 運量比較分析

Table 3. 使用者案例描述-「運量比較分析」

使用者案例名稱	運量比較分析
使用者案例編號	TV0315

描述	統計兩組時間條件對比各站日均運量變化
參與者	系統操作者
前提	查詢條件含以下： 1. 日期區間1 (必填) 2. 日期區間2 (必填)
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，統計不同日期條件下出站人次、人次變化量、變化量排序與變化百分比
衍伸或 替代流程	無輸入查詢條件即進行查詢，需顯示提示訊息
結束狀態	【畫面呈現】 顯示下載連結 【匯出】 提供報表匯出功能，圖表透過 Excel 模板方式產製
備註	

2. 系統營收統計

Table 4. 使用者案例描述-「系統營收統計」

使用者案例 名稱	系統營收統計
使用者案例 編號	TV0601
描述	統計指定月份各站/各票種營收
參與者	系統排程
前提	查詢條件含以下： 1. 營運月份 (必填)
觸發時機	依排程設定觸發時間執行
基本流程	系統自動計算營收額並製檔儲存
衍伸或 替代流程	無
結束狀態	【畫面呈現】 顯示報表下載連結 【匯出】 排程定時產製報表，含 5 報表： 營收月報表、票卡營收月報表、各家票卡營收月報表、行動支付營收月報表
備註	

3. 交易證明申請

Table 5. 使用者案例描述-「交易證明申請」

使用者案例名稱	交易證明申請
使用者案例編號	TV0103
描述	實作 API，根據卡片內碼、乘車日期區間查詢自發行票卡交易紀錄
參與者	系統操作者
前提	須具備以下查詢條件： 1. 卡片內碼 (必填) 2. 乘車日期區間 (必填)
觸發時機	收到呼叫端 API 請求
基本流程	根據使用者請求資訊，檢索符合之交易後回傳
衍伸或替代流程	無輸入乘車日期、卡片內碼進行查詢，需回覆錯誤訊息
結束狀態	【API 回傳】 Response 格式分： 1. 搭乘證明 2. 購票紀錄

4. 優待票交易查詢

Table 6. 使用者案例描述-「優待票交易查詢」

使用者案例名稱	優待票交易查詢
使用者案例編號	TV0104
描述	查詢優待票交易資料
參與者	系統操作者
前提	查詢條件含以下： 1. 交易時間區間 (必填) 2. 交易車站別 3. 票卡業者別 + 票種別 4. 卡片內碼 5. 交易類別
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，檢索符合之交易
衍伸或	無輸入任何搜尋條件進行查詢，需顯示提示訊息

替代流程	
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能
備註	單一搜尋條件(除必填欄位)若為空白，則為不比對項目

5. 定期票交易查詢

Table 7. 使用者案例描述-「定期票交易查詢」

使用者案例名稱	定期票交易查詢
使用者案例編號	TV0105
描述	查詢票交易資料
參與者	系統操作者
前提	查詢條件含以下： 1. 交易時間區間 (必填) 2. 交易車站名稱 3. 定期票銷售天期 4. 定期票起站與訖站 5. 定期票起始日 6. 定期票到期日 7. 卡片內碼 8. 交易類別
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，檢索符合之交易
衍伸或替代流程	無輸入任何搜尋條件進行查詢，需顯示提示訊息
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能
備註	單一搜尋條件(除必填欄位)若為空白，則為不比對項目

6. 出入證使用紀錄查詢

Table 8. 使用者案例描述-「出入證使用紀錄查詢」

使用者案例名稱	出入證使用紀錄查詢
使用者案例	TV0107

編號	
描述	查詢出入證交易資料
參與者	系統操作者
前提	查詢條件含以下： 1. 出入證卡號 (必填) 2. 交易時間區間 (必填)
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，檢索符合之交易
衍伸或 替代流程	無輸入卡號進行查詢，需顯示提示訊息
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能

7. 發票紀錄查詢

Table 9. 使用者案例描述-「發票紀錄查詢」

使用者案例 名稱	發票紀錄查詢
使用者案例 編號	TV0109
描述	根據輸入條件，匯出發票交易明細(含開立、作廢等)
參與者	系統操作者
前提	查詢條件含以下： 1. 交易時間區間 (必填) 2. 交易車站別 3. 發票號碼 4. 發票交易類型：多選 5. 卡號
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，檢索符合之交易
衍伸或 替代流程	無輸入任何搜尋條件進行查詢，需顯示提示訊息
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能
備註	單一搜尋條件(除必填欄位)若為空白，則為不比對項目

8. 10年內未使用Token清單

Table 10. 使用者案例描述-「10 年內未使用 Token 清單」

使用者案例名稱	10 年內未使用 Token 清單
使用者案例編號	TV0110
描述	查詢 10 年內有售出但未使用 Token 清單
參與者	系統操作者
前提	查詢條件含以下： 1. 售出交易時間區間 (必填)
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，檢索符合之交易
衍伸或替代流程	1. 無輸入交易時間區間進行查詢，需顯示提示訊息 2. 資料保存以當下時間前推 10 年為限
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能
備註	

9. 電子票證交易資料

Table 11. 使用者案例描述-「電子票證交易資料」

使用者案例名稱	電子票證交易資料
使用者案例編號	TV0210
描述	排程定時匯出每日電子票證進出站交易資料
參與者	系統排程
前提	查詢條件含以下： 1. 交易時間區間 (必填)
觸發時機	依排程設定觸發時間執行
基本流程	檢索符合之交易並製檔儲存
衍伸或替代流程	須計入 PAG 資料
結束狀態	【畫面呈現】顯示報表下載連結 【匯出】排程定時產製報表
備註	

B. 工單績效指標

透過串接維修管理系統(MMS)資料，設計工單績效指標提供維護績效分析、設備可用度分析、設備故障統計、設備維護履歷管理、維護人力資訊管理與工單資訊管理功能，作為資源盤點、服務完整性與人力統籌分析使用。

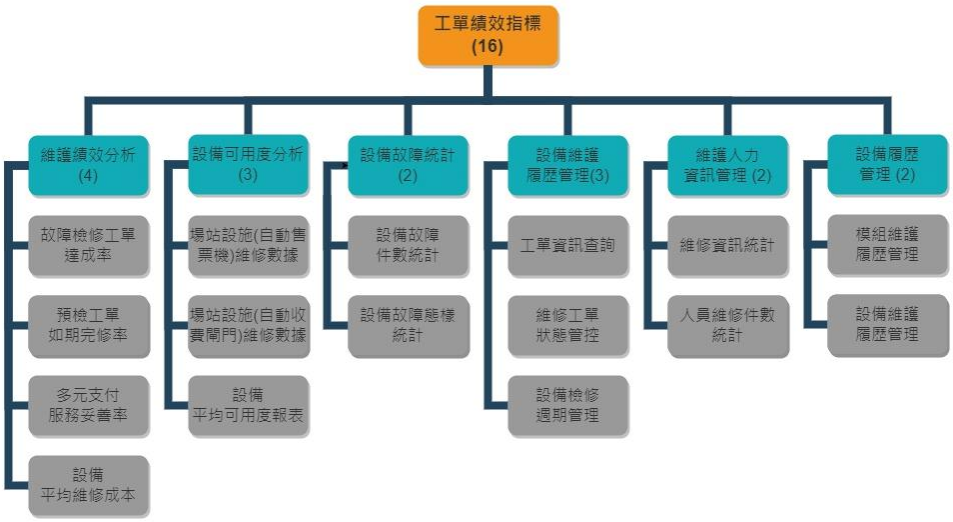

Figure 18. 工單績效指標清單

1. 故障檢修工單達成率

Table 12. 使用者案例描述-「故障檢修工單達成率」

使用者案例名稱	故障檢修工單達成率
使用者案例編號	WO0101
描述	查詢每季故檢工單完成率(當季故檢工單結案數/當季故檢工單總數)
參與者	系統操作者
前提	查詢條件含以下： 1. 年度 (必填) 2. 季度 (必填)
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，統計故障檢修工單達成率
衍伸或替代流程	1. 無輸入查詢條件即進行查詢，需顯示提示訊息 2. 查詢以百分比資料呈現
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能

2. 設備故障樣態統計

Table 13. 使用者案例描述-「設備故障樣態統計」

使用者案例名稱	設備故障態樣統計
使用者案例編號	WO0302
描述	統計月份區間內各類設備故障態樣 1. 每月/年各設備故障態樣統計 2. 態樣數據比對檢核 3. 類樞紐及趨勢統計分析
參與者	系統操作者
前提	查詢條件含以下： 1. 月份區間 (必填) 2. 設備類別 (必填)
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入條件，統計設備故障態樣
衍伸或替代流程	無輸入查詢條件即進行查詢，需顯示提示訊息
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能

C. 物料管理指標

透過串接維修管理系統(MMS)資料，設計物料管理指標提供物料管理與採購規劃功能，作為資源盤點、單位財產管控、成本控管與財務規劃分析使用。

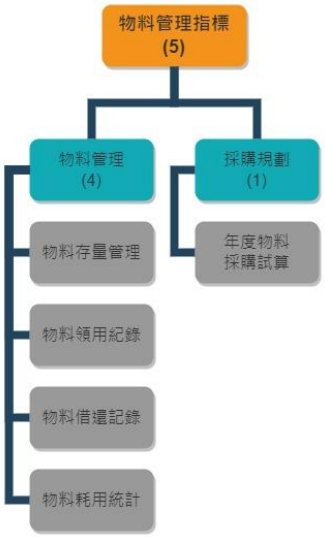

Figure 19. 物料管理指標清單

1. 物料存量管理

Table 14. 使用者案例描述-「物料存量管理」

使用者案例名稱	物料存量管理
使用者案例編號	ML0101
描述	根據總倉庫存資料，加計其他地點庫存(可手動調整)，彙整現物料總庫存
參與者	系統操作者
前提	須建立 UI 供使用者輸入其他倉位物料庫存資訊
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據總倉庫存資料，加計由使用者輸入之其他倉位數量產生統計報表
衍伸或替代流程	由 MMS API 工單內容提供沖銷資訊(例:該次工單物料應自倉庫庫存扣除/修復品/廠商提供何者數量扣除)，系統自動勾稽庫存數量與工單記錄品項呈現消耗
結束狀態	【畫面呈現】 顯示查詢結果 【匯出】 提供報表匯出功能

D. 設備可靠性指標

系統整合前台設備系統、行動支付系統之設備狀態與事件資料，設計設備可靠性指標提供前台設備與中央系統設備監控功能，作為預知提醒與即時監控使用。

Figure 20. 設備可靠性指標清單

1. 設備離線通報

Table 15. 使用者案例描述-「設備離線通報」

使用者案例名稱	設備離線通報
使用者案例編號	EQ0102
描述	提供監控設備連/離線與告警通報
參與者	系統排程
前提	無
觸發時機	依排程設定觸發時間執行
基本流程	定期檢核設備是否離線，當發現設備離線須發送 mail 告警，並於儀表板顯示離線設備。告警與顯示資訊須含： 1. 設備編號 2. 前次連線時間
衍伸或替代流程	1. Mail發送對象，需有介面維護 2. 須可指定監控標的設備 3. 須含告警解除機制
結束狀態	【畫面呈現】儀表板顯示離線設備資訊 【Mail 告警】將離線設備資訊透過信件派送

2. 監控中央伺服器及網路狀態

Table 16. 使用者案例描述-「監控中央伺服器及網路狀態」

使用者案例名稱	監控中央伺服器及網路狀態
使用者案例編號	EQ0201
描述	提供監控中央伺服器及網路狀態
參與者	系統操作者
前提	無
觸發時機	使用者點選查詢
基本流程	網路打通行政網路接連到 MXView or ELK，連接到 MXView 或 ELK 查看資訊
衍伸或替代流程	MXView or ELK 端提供資料格式欄位定義(需為 Web API)，由 5G Server 接收資料作呈現和告警，若資料源不足則無法達成
結束狀態	【畫面呈現】 顯示監控資訊

儀器及手工具管理指標

透過系統介面匯入既有管理資料(Excel)，規劃儀器及手工具管理指標提供儀器與手工具管理功能，作為資源盤點、單位財產管控與效驗履歷管理。

Figure 21. 儀器及手工具管理指標清單

1. 手工具管理

Table 17. 使用者案例描述-「手工具管理」

使用者案例名稱	手工具管理
使用者案例編號	EH0102
描述	提供手工具資訊建檔，並可提供查詢
參與者	系統操作者
前提	查詢條件含以下： 1. 編號 2. 類別 3. 個人/共用 4. 名稱及規格關鍵字 5. 校驗需求 6. 年度 7. 採購案源 8. 股別：機廠、主維修廠 9. 存放位置
觸發時機	輸入查詢條件，按下【查詢】
基本流程	根據使用者輸入搜尋條件，檢索手工具資訊
衍伸或替代流程	無輸入查詢條件即進行查詢，需顯示提示訊息
結束狀態	【畫面呈現】顯示查詢結果 【匯出】提供報表匯出功能

第四章 實驗結果與討論

本系統以三台實體伺服器搭建 Proxmox VE 與 Kubenetes 叢集為架構平台，兩台資料庫伺服器安裝 Oracle 資料庫並建置 RAC(Oracle Real Application Cluster)，使用 ADG(Oracle Active Data Guard)與既有 Oracle 資料庫同步，既有 SQL Server 的部分則運用 SSIS (SQL Server Integration Services)技術進行資料轉移，測試過程中發現，SSIS 因受限於同步二端之 SQL Server 與 Oracle DB 版本之相容性，於同步設定之過程中有所限制，導致無法順利同步，故另自行開發資料同步應用程式，經由設定作業系統排程定期每五分鐘觸發執行，執行功能為檢索來源地資料庫，並參考 SQL Server 與 Oracle DB 之間之欄位資料型別對應關係，以程式方式將來源資料寫入目的地端之 Oracle 資料，並經由 API 的串接，在本系統可將舊系統的資料做整合，依數據指標分類將其結果直覺式的顯示在儀表板資訊上，與過往需要分別從各系統撈出資料再匯出後人工比對相比，輕鬆快速許多。

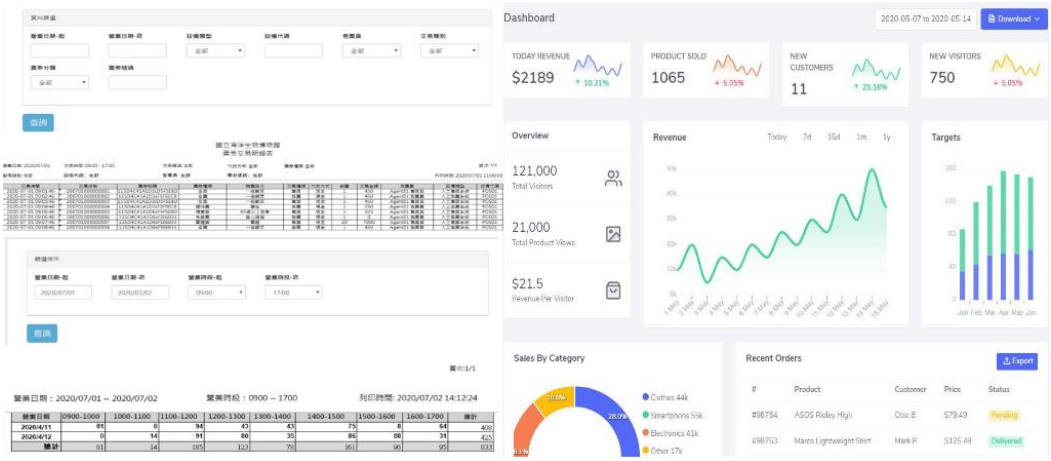

Figure 22. 新舊系統對照

微服務 API 應用程式使用.NET 6 C#為主要程式開發語言，前端使用 Vue.js，採前端與後端各自獨立進行開發設計，透過 Http RESTful API 進行溝通。後端提供 2 大類 API，ParameterAPI 為提供系統網頁調用使用，OpenAPI 以提供異質系統呼叫為目的。並以 Rancher 作為 K8s 維護容器管理平台，進行微服務應用程式開發，Rancher 提供的友善的儀表板 UI 與直覺式的功能操作，可讓管理者清楚

監控 K8s 叢集狀態。

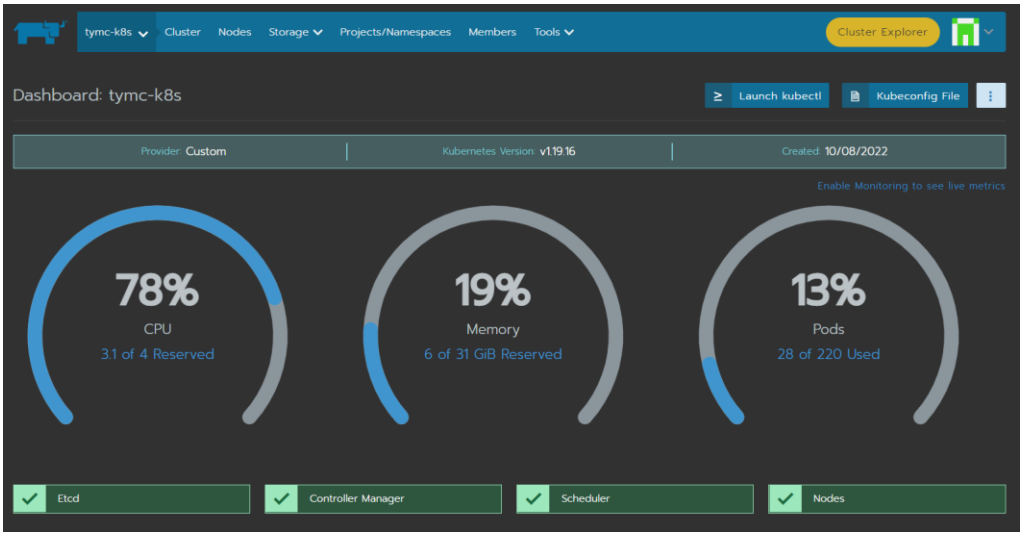

Figure 23. Rancher 儀表板

系統監控與告警使用 Prometheus+Grafana，資源即時監控畫面呈現如 Figure 24 所示。

Figure 25. Grafana 儀表板

本研究使用的技術資訊規劃如下表:

Table 18. 系統使用的技術

系統名稱	種類	部署位置	技術
TAS	WebAPI Client	K8s Pod	.NET 6 C#
AMS	WebAPI Client	K8s Pod	.NET 6 C#
MMS	WebAPI Client	K8s Pod	.NET 6 C#
OSS	WebAPI Client	K8s Pod	.NET 6 C#
監控儀錶板 Dashboard	Web	K8s Pod	Vue、Nuxt、Javascript、Typescript
監控儀錶板	Web API Server	K8s Deploy	.NET 6 C#、

系統名稱	種類	部署位置	技術
ParameterAPI			EF Core 6
OpenAPI	Web API Server	K8s Pod	.NET 6 C#、 EF Core 6
Oracle DB	Database Infrastructure		SQL
Oracle DB資料轉換	Pipeline	K8s Pod	.NET 6 C#、 EF Core 6、 OtherServicePipeline
MongoDB	Database Infrastructure Storage	VM Bare	NoSQL
Redis	Database Infrastructure Storage	VM Bare	NoSQL
Nginx	Infrastructure	K8s	Script
Rancher	Container management Cluster management	VM K8s	Script、 Linux、 Docker
MINIO	Infrastructure Storage	VM Bare	MinIO Client
Promethus + Grafana + AlertManager	Infrastructure Chart Alert Metric	K8s Helm	PromQL、 Script、 Data sources Plugin
Docker Harbor	Docker Image	K8s Helm	Script
Gitlab Runner	CI/CD	K8s Helm	Script
Gitlab	CI/CD	VM Docker	Script
ArgoCD	CI/CD	K8s Helm	Script
Sonarqube	CI/CD	VM Docker	
Fluentd	Log Monitor	K8s	Script
Elasticsearch + Kibana	Log Monitor	VM Docker	Query DSL (Domain Specific Language)

容器化是現今的趨勢，已經建置在虛擬機上的舊系統是否一定都要轉成容器，可從規模及負載來當作判斷依據，如果這個系統，它的負載的量是可預測的，因已經事先分派給它那麼多資源，運作正常也不會過載，這種可照原本的 VM 跑即可，例如:基礎架構的服務，像是 DNS、DHCP。如果這個系統有一定規模、需要快速部署以及尖峰負載過高，這種就可以選擇容器化。

第五章 結論

企業內部資訊系統從最早期的實體主機 Client-server 架構，到後來作業系統虛擬化技術的虛擬機，近年來私有雲和公有雲的架構化及軟體層的容器化已成為主流趨勢。新建的應用或服務，越來越多公司選擇在微服務架構上開發，並透過容器化封裝，達到高可用性、易維護與自動化，循序漸進的達到數位轉型目的。本研究以微服務為基礎之架構規劃設計，在有限制的環境條件下，利用擴充現有設備，使用排程同步的方式進行新舊統資料交換，做為各項關鍵指標呈現的資料來源，並保留彈性，未來新開發的程式只要部署在 K8s Pod，經由設定即可快速部署上線，透過 API 可快速完成資料串接，同步使用叢集內所有的資源。本研究 Hypervisor 使用 Proxmox VE，建置成本遠低於 VMware，可供預算有限的專案參考。

數位轉型可能遇到很多問題，例如:舊系統不易重構翻新、雲端原生容器技術本身的複雜性、運維與開發人才培養難度、資安議題之外，轉型之餘如何為企業內外部創造更大的效益，有賴於經營策略的調整及雲端資料的運用，這是一門值得深入探討的議題。

參考文獻

[1] D. Gannon, R. Barga and N. Sundaresan, "Cloud-Native Applications," in IEEE Cloud Computing, vol. 4, no. 5, pp. 16-21, September/October 2017, doi: 10.1109/MCC.2017.4250939.

[2] M. Fowler and J. Lewis, "Microservices: Nur ein weiteres Konzept in der Softwarearchitektur oder mehr," Objektspektrum, 2015, pp. 14-20.

[3] F. Rossi, V. Cardellini and F. L. Presti, "Hierarchical Scaling of Microservices in Kubernetes," 2020 IEEE International Conference on Autonomic Computing and Self-Organizing Systems (ACSOS), 2020, pp. 28-37, doi: 10.1109/ACSOS49614.2020.00023.

[4] T. Yarygina and A. H. Bagge, "Overcoming Security Challenges in Microservice Architectures," 2018 IEEE Symposium on Service-Oriented System Engineering (SOSE), 2018, pp. 11-20, doi: 10.1109/SOSE.2018.00011.

[5] A. Balalaie, A. Heydarnoori and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," in IEEE Software, vol. 33, no. 3, pp. 42-52, May-June 2016, doi: 10.1109/MS.2016.64.

[6] D. Bernstein, "Containers and Cloud: From LXC to Docker to Kubernetes," in IEEE Cloud Computing, vol. 1, no. 3, pp. 81-84, Sept. 2014, doi: 10.1109/MCC.2014.51.

[7] L. Abdollahi Vayghan, M. A. Saied, M. Toeroe and F. Khendek, "Deploying Microservice Based Applications with Kubernetes: Experiments and Lessons Learned," 2018 IEEE 11th International Conference on Cloud Computing (CLOUD), 2018, pp. 970-973, doi: 10.1109/CLOUD.2018.00148.

[8] V. Singh and S. K. Peddoju, "Container-based microservice architecture for cloud applications," 2017 International Conference on Computing, Communication and Automation (ICCCA), 2017, pp. 847-852, doi: 10.1109/CCAA.2017.8229914.

[9] Preeth E N, F. J. P. Mulerickal, B. Paul and Y. Sastri, "Evaluation of Docker containers based on hardware utilization," 2015 International Conference on Control Communication & Computing India (ICCC), 2015, pp. 697-700, doi: 10.1109/ICCC.2015.7432984.

[10] A. Idrus, "Designing high availability cluster using server virtualization on Cloud Storage Services," IJISTECH (International Journal of Information System & Technology), vol. 5, no. 2, pp. 93, 2021.

[11] A. Agarwal and S. Raina, "Live migration of virtual machines in cloud," International Journal of Scientific and Research Publications, vol. 2, no. 6, pp. 1-5, 2012.