ระบบการประเมินและเปรียบเทียบประสิทธิภาพการทำงานของอัลกอริทึมการขยายตัวแบบอัตโนมัติของซอฟต์แวร์ควบคุมคอนเทนเนอร์

PERFORMANCE EVALUATION AND COMPARISON SYSTEM OF AUTO-SCALING ALGORITHM IN A CONTAINER ORCHESTRATION SOFTWARE

วรเทพ อหันทริก[1]
ดร.ธนัญ จารุวิทยโกวิท[2]

บทคัดย่อ

โครงการวิจัยนี้จัดทำขึ้นเพื่อศึกษาและการประเมินสมรรถนะและเปรียบเทียบประสิทธิภาพของการขยายตัวของ Service บนคอนเทนเนอร์ 2 แพลตฟอร์ม Kubernetes และ Docker Swarm ซึ่งจะประเมินประสิทธิภาพในด้านของ ซีพียู และเวลาในการขยายตัวของ Service ทั้ง 2 คือ NGINX และ Web Service (VITE) เพื่อลดปัญหาการใช้งานจากผู้ใช้งานที่มีปริมาณมาก งานวิจัยนี้ได้ใช้ Public Cloud ของ Google Cloud Platform จัดการโครงสร้าง Kubernetes ด้วย Google Kubernetes Engine (GKE) เป็นเครื่องมือช่วยบริหารจัดการ container ทำงานบน Cloud ช่วยในเรื่องของการสร้าง (build) Service, การนำ Service ไปใช้ (deploy) และการรองรับการขยายตัว (scale) ส่วนของ Docker Swarm จะสร้าง VM Instances บน Google Container Engine โดยบนคอนเทนเนอร์ 2 แพลตฟอร์ม จะติดตั้ง Workload 2 ชนิด คือ NGINX เป็น Web Service ทั่วไป และ VITE เป็น Web Service โดยใส่เนื้อหาวีดีโอ เป็น Full HD:ความละเอียดของภาพ 1080p เพื่อใช้เปรียบเทียบประสิทธิภาพการใช้งาน CPU ของแต่ละ Service ซึ่งกำหนดไว้สถานะไว้ ถ้าสถานะปกติคือ CPU ต่ำกว่า 80% .ให้ Service ทำงานแต่ 1 Service แต่ถ้ามีปริการใช้งาน CPU มากกว่า 80% จะทำการขยายตัวเพื่อให้รองรับปริมาณที่มาใช้มากขึ้น โดยตั้งไว้ให้ขยายได้มากสุดถึง 5 Services เพื่อให้รองรับการทำงานของผู้ใช้ได้อย่างมีประสิทธิภาพ

จากการที่ได้ทดลองเปรียบเทียบทั้ง 2 แพลตฟอร์ม พบว่าการทำงานของ Kubernetes มีความสะดวกสบาย สามารถทำการขยายตัว Service แบบอัตโนมัติได้ และยังขยายตัวได้เร็วกว่า ถึง 93% ในส่วนของ Docker Swarm ต้องอาศัยการ Monitor จากผู้ดูแลระบบเนื่องจากไม่มีตัวช่วยในการขยาย Service แบบอัตโนมัติ ทำให้เกิดความล่าช้ากว่ามาก สรุปคือ Kubernetes สามารถมีการขยายตัวได้เร็วกว่า Docker Swarm

คำสำคัญ : การขยายตัว, คอนเทนเนอร์, เว็บเซอร์วิซ

---

[1] นักศึกษาหลักสูตรวิศวกรรมศาสตร์มหาบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์

[2] ที่ปรึกษาสารนิพนธ์หลัก

ABSTRACT

This research project was conducted to study and evaluate the competence and performance comparison of scaling services on two container platforms, Kubernetes and Docker Swarm. It aims to assess the efficiency in terms of CPU and the time taken for the expansion of two services, namely NGINX and Web Service (VITE). The objective is to address user-related issues that arise due to high user volumes.

This research project utilized the Public Cloud of Google Cloud Platform to manage the Kubernetes infrastructure using Google Kubernetes Engine (GKE) as a tool for container management on the cloud. It facilitated the process of building, deploying, and scaling services. For Docker Swarm, VM instances were created on Google Container Engine. On both container platforms, two types of workloads, NGINX (a general web service) and VITE (a video content web service with a Full HD resolution of 1080p), were installed to compare the CPU performance of each service. The services were set to a standby state, where they would operate as a single service if the CPU usage was below 80%. However, if the CPU usage exceeded 80%, the services would scale up to accommodate the increased workload. The maximum scalability was set to five services to efficiently handle user operations.

After conducting a comparative test on both platforms, it was found that the operation of Kubernetes is more convenient as it allows for the automatic scaling of services and achieves faster scalability, up to 93%. On the other hand, Docker Swarm does not have auto-scaling functionality and relies on system administrators for monitoring. Manual scaling is required, resulting in significant delays. In conclusion, Kubernetes has the advantage of faster scalability compared to Docker Swarm. From the conducted comparison between the two platforms, it was found that Kubernetes offers convenience in managing and automatically scaling services. It allows for fast and automatic scaling. On the other hand, Docker Swarm requires manual monitoring from system administrators as it lacks built-in automatic service scaling capabilities. This leads to significant delays compared to Kubernetes. In conclusion, Kubernetes has the advantage of faster scalability compared to Docker Swarm.

Keywords: Scaling, Container, Web Service

1.บทนำ

เทคโนโลยีสารสนเทศ (Information Technology) เป็นเครื่องมือสำคัญ ประการหนึ่งใน การดำเนินธุรกิจของหน่วยงานบริษัท หรือผู้ให้บริการการประยุกต์ใช้เทคโนโลยีสารสนเทศได้ อย่างเหมาะสม คอนเทนเนอร์จึงถูกสร้างขึ้นมาเพื่อแก้ปัญหาข้างต้น โดยมีฮาร์ดแวร์และ OS เพียงชุดเดียวกัน ลดความซ้ำซ้อนของการใช้ทรัพยากรลง ส่วนตัวแอพพลิเคชันและซอฟต์แวร์ซึ่งเป็นจุดที่แตกต่างกันไปก็จะมี "container" (เทียบได้กับ VM) มาครอบเพื่อแบ่งส่วนทรัพยากรไว้ไม่ให้ยุ่งกัน จุดเด่นของคอนเทนเนอร์จึงเป็นเรื่องการใช้ทรัพยากรที่น้อยกว่า virtualization มาก อิมเมจของคอนเทนเนอร์อาจมีขนาดเพียงกี่ไม่กี่สิบ MB ในขณะที่อิมเมจของ VM ต้องใช้พื้นที่ระดับหลาย GB นอกจากนี้ ระยะเวลาที่ใช้บูต, พลังซีพียูและปริมาณแรมที่ต้องใช้ ก็ลดลงตามไปด้วย ส่งผลให้เซิร์ฟเวอร์หนึ่งเครื่องสามารถยัดคอนเทนเนอร์จำนวนมากกว่าการรัน VM ที่ให้ผลแบบเดียวกันถึง 2-3 เท่าตัว บางครั้งคอนเทนเนอร์ถูกเรียกชื่อในทางเทคนิคว่า Operating-system-level virtualization หรือการสร้าง VM ที่ระดับ OS โดยเราไม่ต้องสร้างเครื่องคอมพิวเตอร์เสมือนขึ้นมาทั้งตัว

1.1 ปัญหาและแรงจูงใจ

ปัญหาที่พบบ่อยในการใช้งาน

1.1.1 ปัญหาการทำ Web Application แต่การออกแบบระบบในตอนแรกไม่ได้ออกแบบไว้ให้ผู้ใช้ Request เข้ามาใช้งานพร้อม ๆ กันเป็นจำนวนมาก พอระบบเริ่มขยายมีคนเข้ามาใช้งานจำนวนมาก ๆ Web Server จึงรับไม่ไหว จึงทำให้ Serverล่มได้

1.1.2 ปัญหา Internet ช้าเป็นเรื่องที่พบเจอ โดยเกิดจากระบบเครือข่ายโดยตรง ไม่ว่าจะเป็นระบบที่ออกแบบมาไม่ดี การใช้อุปกรณ์ไม่มีประสิทธิภาพเนื่องจากงบประมาณหรือการประเมินสเปคผิดพลาด หรือมีการดาวน์โหลดไฟล์เยอะเกินไป

1.1.3 การเลือกใช้ Serverแบบประกอบเองทำให้อุปกรณ์ที่ใช้ไม่ได้มาตรฐานที่ดี

1.1.4 ปัญหาการจัดทำระบบ Database ไม่สอดคล้องเพราะเมื่อระบบ Input และ Output ประมวลผลไม่ทันก็เกิดการจัดการระบบข้อมูลไม่ได้ ซึ่งเป็นอีกสาเหตุที่ทำให้ระบบล่ม

1.1.5 ปัญหาระบบภายในเครื่อง Server จะใช้วลานานในการตรวจสอบสถานะของ server และการตรวจพบปัญหา

1.2 วัตถุประสงค์ของงานวิจัย

1.2.1 เพื่อทดสอบและเปรียบเทียบประสิทธิภาพของการทำงานของการขยายตัวของคอนเทนเนอร์ 2 แพลตฟอร์มคือ Kubernetes และ Docker Swarm

1.2.2 เพื่อเปรียบเทียบความแตกต่างระหว่าง CPU load ของ 2 Service คือ NGINX และ Webtest ที่เขียนด้วย React VITE

1.3 ขอบเขตของโครงงาน

งานวิจัยนี้จะทำการทดสอบบนระบบคอนเทนเนอร์คลัสเตอร์ Kubernetes และ Docker Swarm ที่ทำงานบน Public Cloud (Google Cloud Platform) โดยจะเปรียบเทียบอุปกรณ์หลักของระบบ เช่น CPU และการขยายตัวของ Web Service NGINX และ Webtest ที่เขียนด้วย React VITE โดยมีรายละเอียดขอบเขตงานวิจัยดังนี้

1.3.1 ใช้ระบบ Kubernetes รุ่นเสถียร รหัสรุ่น 1.24.12 (GKE)

1.3.2 ใช้ระบบ Docker Engines รุ่น 24.0.2

1.3.3 ใช้ระบบ Container Runtime เป็น Docker

1.3.4 Public Cloud ที่ใช้ในการทดสอบ ใช้บริการจาก Google Cloud Platform (GCP) โซนของ ประเทศสิงคโปร์ รหัสโซน asia-southeast1-a และ ประเทศอินโดนีเซีย รหัสโซน asia-southeast2-a

1.3.5 คอมพิวเตอร์ Workstation ติดตั้งระบบปฏิบัติการ Ubuntu Linux Server 18.04.5

1.3.6 ซอฟต์แวร์ที่ใช้ในการทดสอบ ใช้โปรแกรม wrk

2. ทฤษฎีเอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 การประมวลผลแบบกลุ่มเมฆ (Cloud computing) [1]

Cloud Computing เป็นเสมือนคอมพิวเตอร์เครื่องหนึ่งที่เราใช้งานกันอยู่คือมีทั้งฮาร์ดแวร์และซอฟต์แวร์ที่ใช้ในการประมวลผลและเก็บข้อมูล แต่ Cloud Computing เป็นระบบคอมพิวเตอร์ที่ใหญ่มากๆ รองรับการใช้งาน การประมวลผล ตลอดจนการจัดเก็บข้อมูลได้อย่างมหาศาล

Cloud Computing สามารถจำแนกตามการใช้งานได้ 3 ประเภท

2.1.1 Infrastructure as a Service (IaaS)

2.1.2 Platform as a Service (PaaS)

2.1.3 Software as a Service (SaaS)

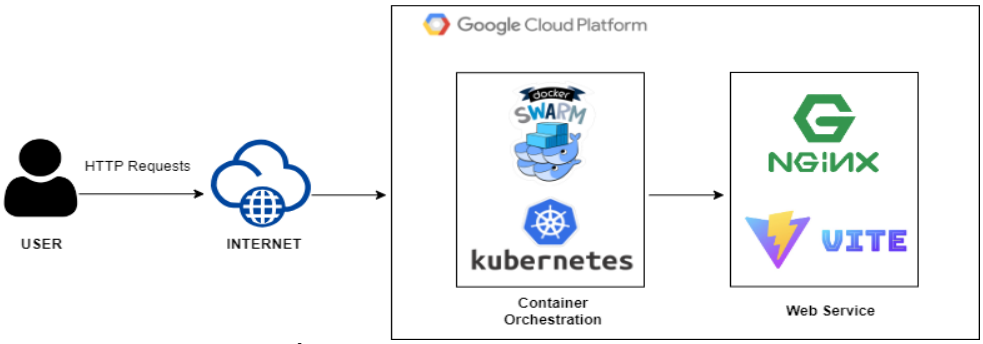
2.2 การประมวลผลแบบคอนเทนเนอร์ (Container) [2]

Containers คือการจัดเก็บแอพพลิเคชั่นให้อยู่ในรูปแบบที่ง่ายต่อการโยกย้ายและนำขึ้นระบบการปรับใช้ โดยประกอบไปด้วยตัวแอปพลิเคชั่น library หรือ binary จำเป็นในการรันแอพพลิเคชั่นนั้นและการกำหนดค่าของแอปพลิเคชั่นนั้น ไม่ว่าจะย้ายตัวแอพพลิเคชั่นที่ถูกจัดเก็บเป็น คอนเทนเนอร์ไปรันที่ไหนจะสามารถทำงานได้เหมือนเดิมโดยไม่คำนึงถึงระบบปฏิบัติการ ซึ่งเป็นเทคโนโลยีการจำลองเสมือนของระบบปฏิบัติการคอมพิวเตอร์เช่น CPU Memory แต่ละคอนเทนเนอร์ทำหน้าที่เป็นแพ็กเกจแยกต่างหากซึ่งสามารถเรียกใช้แอปพลิเคชันหรือบริการได้โดยไม่ต้องคำนึงถึงสภาพแวดล้อมการประมวลผลพื้นฐาน ภายในหนึ่ง คอนเทนเนอร์สามารถรันแอปพลิเคชั่นทั้งตัวหากเราแบ่งแบบ Microservice ก็ได้แอพพลิเคชั่นในรูปแบบ คอนเทนเนอร์ โดยจะถูกรันผ่าน Software ที่เรียกว่า Container engine/runtime จะมีชั้นของแอปพลิเคชันที่เก็บอยู่รวมกัน มีแพคเกจหรือชุดคำสั่งที่ใช้ทำงานร่วมกัน ตั้งแต่หนึ่งตัวขึ้นไปสามารถรันบนคอมพิวเตอร์เครื่องเดียวกันและใช้เคอร์เนล ของระบบปฏิบัติการร่วมกันกับคอนเทนเนอร์ตัวอื่น ๆ แต่ละตัวทำงานหรือประมวลผลแยกในพื้นที่ของตัวเองและสามารถจัดการแอปพลิเคชันได้มากขึ้น ใช้จำนวน Virtual Machine และจำนวนระบบปฏิบัติการน้อยลง

3. การออกแบบและพัฒนาระบบ

3.1 การวิเคราะห์และออกแบบระบบ

องค์ประกอบของการทำงานของการเปรียบเทียบการขยายตัวจะมีภาพรวมการทำงานตั้งแต่มีการเรียกใช้งานจากผู้ใช้ ด้วย HTTP Request ผ่าน Internet เข้าไปที่ คอนเทนเนอร์ที่อยู่บน Public Cloud และไปถึง Web Service ที่อยู่บนคอนเทนเนอร์แสดงได้ดังภาพที่1



ภาพที่1 แสดงโครงสร้างการทำงานของระบบ

3.2 ขั้นตอนการทดลอง

3.2.1 ออกแบบและติดตั้งระบบคอนเทนเนอร์ (Master node, Worker Node) ของ Kubernetes และ Docker Swarm บน Google Cloud

3.2.2 ออกแบบและติดตั้ง Web Service มี NGINX, VITE บนคอนเทนเนอร์ ทั้ง 2 ให้เหมือนกัน

3.2.3 เริ่มทำการทดสอบโดย Generate Load ส่งไปยัง Web Service ด้วย wrk HTTP Benchmarking Tool เพื่อดูพฤติกรรมของ CPU

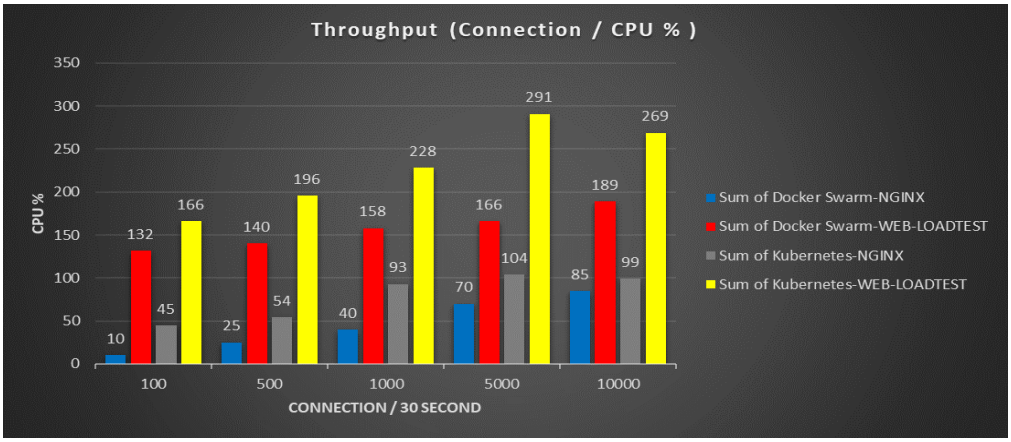
3.2.4 เฝ้าดูพฤติกรรมของ CPU โดยที่มีการตั้งค่าของแต่ละ Service ให้มีการใช้ปกติอยู่ที่ 1 pod และขยายได้สูงสุด 5 pod มีการทำงานของ CPU เกิน 80% จะมีการขยายตัวออก เพื่อรองรับการการบริการของ Web service และถ้า CPU ต่ำกว่า 25% จะขยายตัวเข้า เพื่อลดการใช้ทรัพยากรของระบบ

3.2.5 บันทึกผลการทดลอง เพื่อนำไปวิเคราะห์ผลการทดลอง ประเมินสมรรถนะและเปรียบเทียบประสิทธิภาพ

4. ผลการดำเนินงานวิจัย

เพื่อให้ผลการทดลองบรรลุวัตถุประสงค์ของการทำวิจัยจึงได้ทดสอบ CPU โดยแบ่งเป็น HTTP requests จำนวน 100,500, 1,000, 5,000 และ 10,000 request ต่อ 30 วินาที

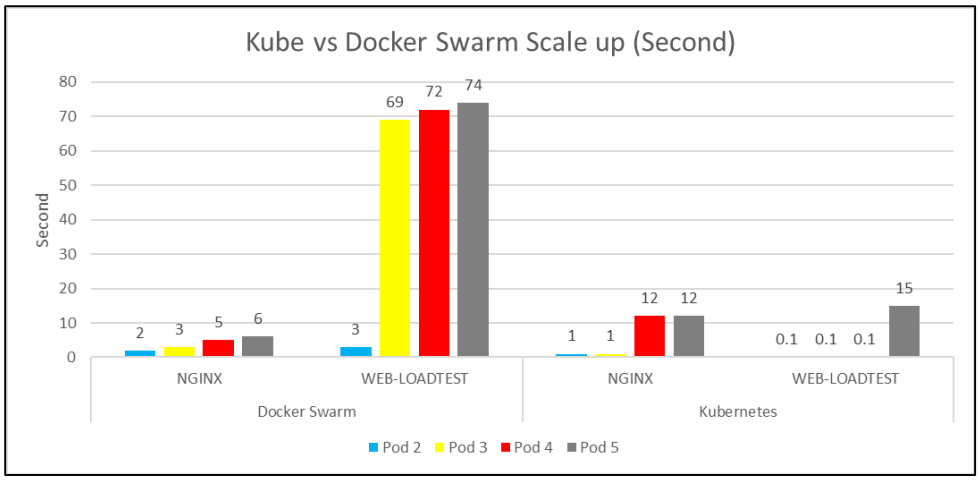
4.1 ผลการทดสอบการทำงานของ CPU



ภาพที่2 แสดงผลการทดสอบการทำใช้งานของซีพียู

จากภาพที่ 2 เป็นการเปรียบเทียบทำงานของ CPU เมื่อมีการเรียกใช้งาน Web Service จะเห็นได้ว่า CPU ของแต่ละคอนเทนเนอร์ และของแต่ Web Service ที่มีการเรียกใช้ Connection ที่มีปริมาณงานที่เท่ากัน แต่การใช้งานงานปริมาณ CPU แตกต่างกัน ผลที่ออกมาจะแสดงให้เห็นว่า Service ของ NGINX ที่ไม่มีเนื้อหา จึงมีการใช้งานของ CPU ที่น้อย เมื่อเทียบกับ Webtest เพราะ Webtest มีวีดีโอ Full HD (ความละเอียดของภาพ 1080p) จึงทำให้ CPU มีการทำงานมากขึ้นอย่างเห็นได้ชัดทั้งที่อยู่ในคอนเทนเนอร์เดียวกัน

4.2 ผลการเปรียบเทียบการขยายตัวของคอนเทนเนอร์ของ Kubernetes และ Docker Swarm



ภาพที่3 กราฟแสดงผลการทดสอบเปรียบเทียบของ ด็อกเกอร์ สวอม และ คูเบอร์เนทิส

จากภาพที่ 3 จะได้แสดงให้เห็นอย่างชัดเจนว่าคอนเทนเนอร์ของ Kubernetes การขยายตัวที่เร็วกว่ามากถึง 93% ถ้าดูจากกราฟแล้ว Docker swarm จะใช้เวลามากกว่า เนื่องจาก Docker Swarm ไม่มี Auto scaling จำเป็นต้องให้ผู้ดูแลระบบทำ manual scaling จึงทำให้เกิดความล่าช้า และ มีการขยายตัวของ Service ที่มี content ที่มีเนื้อหาเยอะได้ช้าจึงสรุปได้ว่าการที่จะเลือกใช้คอนเทนเนอร์เพื่อมาใช้ในการให้บริการทั้งภายในหรือบริการให้ลูกค้าควรเลือกคอนเทนเนอร์ที่มีประสิทธิภาพการรองรับการให้บริการที่ดีกว่าอย่าง Kubernetes

สรุปผลตามวัตถุประสงค์ของงานวิจัย

จากการที่ได้ทดลองเปรียบเทียบทั้ง 2 แพลตฟอร์ม พบว่าการทำงานของ Kubernetes มีความสะดวกสบาย สามารถทำการขยายตัว Service แบบอัตโนมัติได้ และยังขยายตัวได้เร็วกว่า Docker Swarm ถึง 93% ในส่วนของ Docker Swarm ไม่มี Auto scaling ต้องอาศัยการ Monitor จากผู้ดูแลระบบ จำเป็นต้องให้ผู้ดูแลระบบทำ manual scaling ทำให้เกิดความล่าช้ากว่ามาก สรุปคือ Kubernetes สามารถมีการขยายตัวได้เร็วกว่า Docker Swarm

บรรณานุกรม

[1]https://www.aosoft.co.th/article/342/Cloud-computing.html

[2]https://www.mindphp.com/%E0%B8%84%E0%B8%B9%E0%B9%88%E0%B8%A1%E0%B8%B7%E0%B8%AD/73%E0%B8%84%E0%B8%B7%E0%B8%AD%E0%B8%AD%E0%B8%B0%E0%B9%84%E0%B8%A3/9187-containers.html

[3] สิทธินนท์ศรีสวัสดิ์และทรงฤทธิ์กิติศรีวรพันธุ์ การสเกลพอดแกนแนวนอนด้วยปฏิทินออนไลน์สำหรับคูเบอร์เนติส2021 https://ph02.tci-thaijo.org/index.php/ectiard/article/view/243951/165521

[4] ธีรภัทร์ขุนเพ็ชร์และสุชาสมานชาต การขยายขนาดการให้บริการแบบอัตโนมัติในระบบด็อกเกอร 2020 https://nccit.net/wp-content/uploads/2016/05/Example_Paper.pdf

[5] M. Song, C. Zhang and E. Haihong, “An autoscaling system for api gateway based on kubernetes,” In 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), pp. 109-112, Nov 2018.