การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์

พรรักษา กมลเวชช์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์และโทรคมนาคม วิทยาลัยนวัตกรรมด้านเทคโนโลยี
และวิศวกรรมศาสตร์
มหาวิทยาลัยธุรกิจบัณฑิตย์
พ.ศ. 2561

Performance evaluation of K8s container networking

Pornraksa Kamolvage

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering

Department of Computer and Telecommunication Engineering

College of Innovative Technology And Engineering,

Dhurakij Pundit University

2018

ใบรับรองวิทยานิพนธ์

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

มหาวิทยาลัยธุรกิจบัณฑิตย์

ปริญญา วิศวกรรมศาสตรมหาบัณฑิต

หัวข้อวิทยานิพนธ์ การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์

เสนอโดย นางสาวพรรักษา กมลเวชช์

สาขาวิชา วิศวกรรมคอมพิวเตอร์และโทรคมนาคม

อาจารย์ที่ปรึกษาวิทยานิพนธ์ อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์

ได้พิจารณาเห็นชอบโดยคณะกรรมการสอบวิทยานิพนธ์แล้ว

.....ประธานกรรมการ

(อาจารย์ ดร.ประศาสน์ จันทราทิพย์)

.....กรรมการและอาจารย์ที่ปรึกษาวิทยานิพนธ์

(อาจารย์ ดร.ชัยพร เขมะภาตะพันธ์)

..... กรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์)

..... กรรมการ

(อาจารย์ ดร.เจนจบ วีระพานิชเจริญ)

วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์รับรองแล้ว

..... คณบดีวิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรมศาสตร์

(ผู้ช่วยศาสตราจารย์ ดร.ณรงค์เดช กีรติพรานนท์)

วันที่ 19 เดือน ก.ค. พ.ศ. 2561

หัวข้อวิทยานิพนธ์ การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์
ชื่อผู้เขียน พรรักษา กมลเวชช์
อาจารย์ที่ปรึกษา ดร.ชัยพร เขมะภาตะพันธ์
สาขาวิชา วิศวกรรมคอมพิวเตอร์และโทรคมนาคม
ปีการศึกษา 2560

บทคัดย่อ

คูเบอร์เนตส์หรือ K8s เป็นระบบจัดการประมวลผลแบบเสมือนอีกแบบที่พัฒนาโดยกูเกิ้ลกำลังได้รับความนิยมเป็นอย่างมากสำหรับการประมวลผลคลาวด์ โดยมีพื้นฐานมาจากการใช้งานและจัดการคอนเทนเนอร์ของดอคเกอร์บนโหนดประมวลผลหลายโหนดที่เป็นคลัสเตอร์ อย่างไรก็ตามคูเบอร์เนตส์มีวิธีการการเชื่อมต่อเครือข่ายบนคลัสเตอร์ของคอนเทนเนอร์หลายวิธี ดังนั้นงานวิจัยนี้จึงทำการเปรียบเทียบประสิทธิภาพทางด้านเวลาในการตอบสนองที่ใช้ในการตอบกลับ เปอร์เซ็นต์ทำงานของ CPU ปริมาณการดาวน์โหลดข้อมูล และค่า Throughput ของการให้บริการเว็บ ของการเชื่อมต่อเครือข่ายคอนเทนเนอร์ 3 แบบได้แก่ Flannel, Calico และ Cannel จากผลการทดลองพบว่า Calico มีประสิทธิภาพดีที่สุดจากทั้ง 3 วิธี โดยเฉพาะอย่างยิ่งกรณีที่มีปริมาณทราฟฟิกเป็นจำนวนมาก สำหรับวิธี Flannel และ Canal ให้ประสิทธิภาพใกล้เคียงกัน เนื่องจากทั้ง 2 วิธีอยู่บนพื้นฐานของ VxLAN นั่นเอง อย่างไรก็ตาม Flannel มีการนำมาใช้งานที่ง่ายที่สุด

Thematic Paper Title	Performance evaluation of K8s container networking
Author	Pornraksa Kamolvage
Thematic Paper Advisor	Dr. Chaiyaporn Khemapatapan
Department	Computer and Telecommunication Engineering
Academic Year	2017

ABSTRACT

Kubernetes, K8s, is a new virtualized computing orchestration developed by Google and being popular cloud computing. K8s is designed for deployment and management Docker containers on clustered computer nodes. However, there are many container networking schemes for the clustering. Thus, in this paper, performance comparison based on throughput, response time, cpu performance and data download quality from Web server which connected via 3 various container networking: Flannel, Calico and Canal will be evaluated. From the experimental results, Calico achieved best performance especially for heavy traffic requirement. Flannel and Canal stand the similar performance because both methods used VxLAN technology. However, Flannel is most simply for implemented.

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความอนุเคราะห์ของบุคคลหลายท่าน โดยเฉพาะอย่างยิ่ง ดร.ชัยพร เขมะภาตะพันธ์ ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ซึ่งให้ความรู้ แนะนำแก้ไขข้อบกพร่องต่างๆ ตลอดจนให้คำชี้แนะช่วยเหลือ พร้อมทั้งให้ประสบการณ์ต่างๆ แต่ผู้วิจัย กราบขอบพระคุณเป็นอย่างสูง

ทางผู้วิจัยขอขอบพระคุณ ดร.ประศาสน์ จันทราทิพย์ ผศ.ดร.ณรงค์เดช กีรติพรานนท์ ดร.เจนจบ วีระพานิชเจริญ ที่ได้สละเวลามาเป็นกรรมการสอบวิทยานิพนธ์ และขอขอบพระคุณอาจารย์และเจ้าหน้าที่ทุกท่านของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์ โดยเฉพาะอย่างยิ่งพี่แวว ที่ให้คำแนะนำและช่วยเหลืออย่างดีเสมอมา

ขอขอบคุณพี่ภาณุวง เมฆไพบูลย์ และวรานนท์ จินาหยิน ที่คอยให้คำแนะนำ การแก้ปัญหาต่างๆ ในการทำวิทยานิพนธ์

ท้ายที่สุดนี้ ขอกราบขอบคุณพระคุณบิดา มารดา ผู้อยู่เบื้องหลังความสำเร็จ และนายธรรมรักษ์ เติมทองสุขสกุล ที่เป็นที่ปรึกษาและได้ให้ความช่วยเหลือ ให้กำลังใจ และสนับสนุนในการศึกษาตลอดมา

พรรักษา กมลเวชช์

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จได้ด้วยความอนุเคราะห์ของบุคคลหลายท่าน โดยเฉพาะอย่างยิ่ง ดร.ชัยพร เขมะภาตะพันธ์ ซึ่งเป็นอาจารย์ที่ปรึกษาวิทยานิพนธ์ ผู้ซึ่งให้ความรู้ แนะนำแก้ไขข้อบกพร่องต่างๆ ตลอดจนให้คำชี้แนะช่วยเหลือ พร้อมทั้งให้ประสบการณ์ต่างๆ แต่ผู้วิจัย กราบขอบพระคุณเป็นอย่างสูง

ทางผู้วิจัยขอขอบพระคุณ ดร.ประศาสน์ จันทราทิพย์ ผศ.ดร.ณรงค์เดช กีรติพรานนท์ ดร.เจนจบ วีระพานิชเจริญ ที่ได้สละเวลามาเป็นกรรมการสอบวิทยานิพนธ์ และขอขอบพระคุณอาจารย์และเจ้าหน้าที่ทุกท่านของคณะวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์ โดยเฉพาะอย่างยิ่งพี่แวว ที่ให้คำแนะนำและช่วยเหลืออย่างดีเสมอมา

ขอขอบคุณพี่ภาณุวง เมฆไพบูลย์ และวรานนท์ จินาหยิน ที่คอยให้คำแนะนำ การแก้ปัญหาต่างๆ ในการทำวิทยานิพนธ์

ท้ายที่สุดนี้ ขอกราบขอบคุณพระคุณบิดา มารดา ผู้อยู่เบื้องหลังความสำเร็จ และนายธรรมรักษ์ เติมทองสุขสกุล ที่เป็นที่ปรึกษาและได้ให้ความช่วยเหลือ ให้กำลังใจ และสนับสนุนในการศึกษาตลอดมา

พรรักษา กมลเวชช์

สารบัญ

หน้า

สารบัญ (ต่อ)

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญภาพ

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

สารบัญภาพ (ต่อ)

ภาพที่ **หน้า**

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของปัญหา

ทุกวันนี้เทคโนโลยีการประมวลผลเสมือนโดยเฉพาะด็อกเกอร์คอนเทนเนอร์ (Docker Container) มีบทบาทในระบบประมวลผลบนอินเตอร์เน็ตหรือคลาวด์ (Cloud) มาก เนื่องมาจากการใช้ทรัพยากรที่น้อยกว่า และสามารถย้ายการติดตั้งได้ง่ายและทำงานได้เร็ว การทำงานร่วมกันเป็นคลัสเตอร์ของคอนเทนเนอร์กลายเป็นปัจจัยสำคัญในการสร้างระบบให้บริการบนระบบคลาวด์ ซึ่งสามารถจัดการได้โดยใช้งาน Orchestrator เช่น Docker Swarm หรือ Kubernetes ซึ่งเป็นระบบควบคุมกำกับการทำงานในการประมวลผลขนาดเล็กที่เป็นระบบเสมือนได้ อย่างไรก็ตาม Kubernetes หรือ K8s มีแนวโน้มการใช้งานอย่างกว้างขวางกว่า

ในการเชื่อมโยงระหว่างโหนดคอนเทนเนอร์ของ Kubernetes สามารถเลือกใช้งานได้จากหลายวิธี ซึ่งสิ่งนี้ส่งผลต่อความเร็วและเวลาในการส่งข้อมูลระหว่างคอนเทนเนอร์และจากคอนเทนเนอร์มายังผู้ใช้งาน ซึ่งโดยปกติวิธีการแบบ Flannel จะเป็นวิธีที่ถูกแนะนำให้ใช้งานโดยปริยายจาก Google ดังนั้นในงานวิจัยชิ้นนี้ได้ทำการเปรียบเทียบประสิทธิภาพทางด้านเวลาในการตอบสนองที่ใช้ในการตอบกลับ (Response Time) ไปยังผู้ใช้บริการของ 3 วิธีด้วยกัน คือ Flannel, Calico และ Canal เพื่อให้สามารถเลือกวิธีการเชื่อมโยงได้เหมาะสมกับการใช้งานและมีประสิทธิภาพสูงสุด

ทั้งนี้ในบทที่ 2 จะกล่าวถึงทฤษฎี และงานวิจัยที่เกี่ยวข้อง บทที่ 3 จะกล่าวถึงรายละเอียดของระเบียบวิธีวิจัย บทที่ 4 จะกล่าวถึงผลการวิจัย และในบทที่ 5 จะหล่าวถึงการสรุปผลการวิจัยและข้อเสนอแนะ

1.2 วัตถุประสงค์ของงานวิจัย

1. ศึกษาเทคโนโลยีทางด้านด็อกเกอร์คอนเทนเนอร์
2. ศึกษาการใช้งานคูเบอร์เนตส์ และวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค
3. เพื่อประเมินและวิเคราะห์ประสิทธิภาพวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธีด้วยกัน คือ Flannel, Calico และ Canal

1.3 ขอบเขตของงานวิจัย

1. สร้างระบบประมวลผลคลาวด์ด้วยคูเบอร์เนต โดยใช้วิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธีด้วยกัน คือ Flannel, Calico และ Canal

2. สร้างเครื่องจักรเสมือน 3 เครื่อง แต่ละชุดใช้วิธีการเชื่อมโยงของคอนเทนเนอร์แตกต่างกัน โดยแต่ละชุดประกอบไปด้วยเครื่องมาสเตอร์ 1 เครื่อง และเครื่องมินเนี่ยน 2 เครื่อง

3. เปรียบเทียบประสิทธิภาพทางด้านเวลาในการตอบสนองที่ใช้ในการตอบกลับ (Response Time) ของวิธีการเชื่อมโยงคอนเทนเนอร์เน็ตเวิร์คทั้ง 3 วิธี

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ทราบถึงหลักการทำงาน และใช้งานคูเบอร์เนตส์ รวมถึงวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค

2. ทราบถึงประสิทธิภาพ และข้อจำกัดในการใช้งานของวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์คทั้ง 3 วิธี คือ Flannel, Calico และ Canal

3. สามารถเลือกใช้และสร้างคอนเทนเนอร์เน็ตเวิร์คที่เหมาะสมกับการใช้งานได้

1.5 แผนการดำเนินงาน

ตารางที่ 1.1 แผนการดำเนินงาน

	หัวข้อ	ช่วงเดือน / ปี (พ.ศ.)				
		ม.ค. 2561	ก.พ. 2561	มี.ค. 2561	เม.ย. 2561	พ.ค. 2560
1	ค้นคว้า ศึกษาและรวบรวมงานวิจัยที่เกี่ยวข้องกับการเปรียบเทียบประสิทธิภาพของวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์					
2	ศึกษาเทคโนโลยีด็อกเกอร์คอนเทนเนอร์					
3	ศึกษาเครื่องมือที่ใช้ในการจัดการด็อกเกอร์คอนเทนเนอร์					

ตารางที่ 1.1 แผนการดำเนินงาน (ต่อ)

	หัวข้อ	ช่วงเดือน / ปี (พ.ศ.)				
		ม.ค. 2561	ก.พ. 2561	มี.ค. 2561	เม.ย. 2561	พ.ค. 2560
4	ศึกษาวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ของคูเบอร์เนตส์					
5	ออกแบบโครงสร้างของระบบการประมวลผลคลาวด์					
6	เปรียบเทียบ วิเคราะห์ผลที่ได้ และสรุป					
7	รวบรวมข้อมูลที่ได้ทั้งหมดจัดทำวิทยานิพนธ์					

1.6 ผลงานตีพิมพ์

งานวิจัยนี้ได้รับการตีพิมพ์ในงานประชุมวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 14 (NCCIT 2018) ในวันที่ 5-6 กรกฎาคม พ.ศ. 2561 จัดโดยมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ ชื่อผลงานตีพิมพ์ “การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์”

บทที่ 2
แนวคิด ทฤษฎี และผลงานวิจัยที่เกี่ยวข้อง

การวิจัยเรื่อง การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์ จำเป็นต้องมีความรู้เกี่ยวกับเทคโนโลยีคอนเทนเนอร์ และคูเบอร์เนตส์ ซึ่งในบทนี้จะกล่าวถึงเนื้อหาดังต่อไปนี้

2.1 การประมวลผลแบบกลุ่มเมฆ (Cloud Computing)

2.2 เทคโนโลยีเวอร์ชวลไลซัน

2.3 ด็อกเกอร์ (Docker)

2.4 คูเบอร์เนตส์ (Kubernetes)

2.5 คอนเทนเนอร์เน็ตเวิร์คกิ้ง (Container Networking)

2.6 Web Stress Simulator

2.7 โปรแกรม Apache JMeter

2.8 งานวิจัยที่เกี่ยวข้อง

2.1 การประมวลแบบกลุ่มเมฆ (Cloud Computing)

การประมวลแบบกลุ่มเมฆ หมายถึง รูปแบบการให้บริการทรัพยากรทางด้านเทคโนโลยีให้กับผู้ใช้บริการผ่านทางระบบเครือข่าย โดยสามารถเข้าใช้งานได้หลากหลายอุปกรณ์ ผู้ใช้งานสามารถปรับเพิ่มหรือลดการใช้ทรัพยากรได้ง่ายและรวดเร็วตามความต้องการทั้งยังสามารถเข้าถึงทรัพยากรได้ตลอดเวลา ซึ่งทรัพยากรที่จัดสรรให้นั้นสามารถมาได้จากหลายแหล่ง (Mell & Grance, 2011) ผู้ใช้บริการไม่จำเป็นต้องรับรู้ถึงความซับซ้อนของการทำงานภายในระบบหรือมีความเชี่ยวชาญทางเทคนิคสำหรับการทำงานนั้นๆ ผู้ใช้บริการสามารถทำงานได้โดยอาศัยไฮเพอร์ไวเซอร์ (Hypervisor) หรือเวอร์ชวลไลเซชัน (Virtualization) เช่น Microsoft Hyper-V, VMware เป็นต้น

2.1.1 คุณลักษณะที่สำคัญของการประมวลผลแบบกลุ่มเมฆ มี 5 ประการ ดังนี้

1. การบริการตนเองเมื่อต้องการ (On-demand self-service) โดยผู้ใช้บริการระบุความต้องการ และขอใช้ทรัพยากรได้ด้วยตนเอง และสามารถทำได้ตลอดเวลา โดยไม่จำเป็นต้องให้ผู้บริการดำเนินการให้

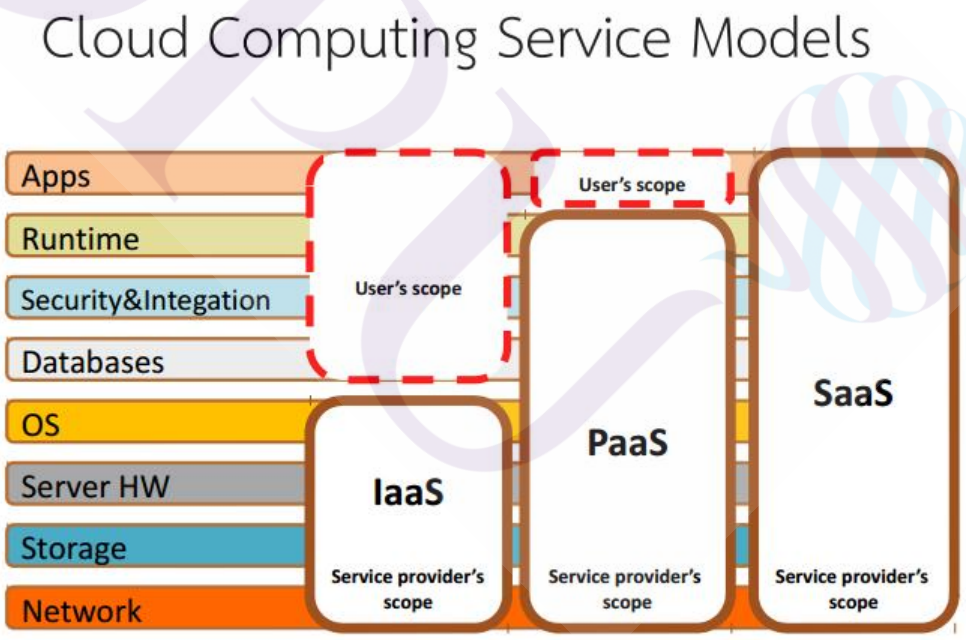
2. การเข้าถึงทรัพยากรผ่านทางเครือข่าย (Broad network access) ผู้ใช้บริการสามารถเข้าถึงบริกการได้โดยผ่านระบบเครือข่ายและเข้าใช้งานได้หลากหลายอุปกรณ์

3. การรวบรวมทรัพยากรจากที่ต่างๆ (Resource polling) ทรัพยากรทางด้านเทคโนโลยีที่ผู้บริการจัดสรรให้กับผู้ใช้บริการสามารถมาจากหลายแหล่ง นอกจากนี้ทรัพยากรชุดเดียวกันสามารถให้ผู้ใช้บริการหลายรายใช้บริการได้โดยไม่รบกวนกัน

4. การปรับตัวและความยืดหยุ่น (Rapid elasticity) การให้บริการสามารถเพิ่มและลดขนาดของทรัพยากรที่ให้บริการได้ตลอดเวลา และตามความต้องการ พร้อมทั้งสามารถจัดสรรทรัพยากรได้โดยอัตโนมัติ

5. ระบบที่ให้บริการสามารถวัดปริมาณการใช้ได้ (Measured Service) การให้บริการต้องสามารถวัดปริมาณการใช้งานได้ตามความเป็นจริง สามารถติดตามและควบคุมการใช้ทรัพยากรได้ตลอดเวลา โดยสามารถแสดงรายงานการใช้บริการให้แก่ผู้ใช้บริการ และผู้รับบริการได้อย่างถูกต้อง

2.1.2 รูปแบบการให้บริการ มี 3 รูปแบบ ดังนี้



ภาพที่ 2.1 รูปแบบการให้บริการของระบบประมวลแบบกลุ่มเมฆ

ที่มา: http://www.similantechnology.com/news&article/Cloud-computing-tech.html

1. Cloud Software as a Service: SaaS การให้บริการใช้ระบบ หรือโปรแกรมประยุกต์ที่ผู้ให้บริการจัดหาให้ โปรแกรมประยุกต์ที่ให้บริการจะทำงานกับโครงสร้างพื้นฐานของระบบประมวลผลแบบกลุ่มเมฆ ผู้ใช้บริการสามารถใช้งานผ่านเว็บเบราว์เซอร์จากหลากหลายอุปกรณ์ หรือผ่านทางโปรแกรมประยุกต์ นอกจากนี้ผู้ใช้บริการสามารถตั้งค่าการใช้งานบางอย่างของโปรแกรมประยุกต์เพื่อความเหมาะสมกับการใช้งานของผู้ใช้งาน แต่ผู้ใช้งานไม่สามารถจัดการหรือควบคุมโครงสร้างพื้นฐาน เช่น ระบบปฏิบัติการ อุปกรณ์จัดเก็บข้อมูล ระบบเครือข่าย เครื่องแม่ข่าย เป็นต้น เนื่องจากถูกจัดการโดยผู้ให้บริการ

2. Cloud Platform as a Service: PaaS การใช้บริการ โครงสร้างพื้นฐานเพื่อพัฒนาโปรแกรมประยุกต์ที่ผู้ให้บริการจัดเตรียมให้เท่านั้น โดยที่ผู้ใช้บริการไม่สามารถจัดการโครงสร้างพื้นฐานทาง เช่น ระบบปฏิบัติการ อุปกรณ์จัดเก็บข้อมูล ระบบเครือข่าย เครื่องแม่ข่าย เป็นต้น แต่ผู้ใช้บริการสามารถตั้งค่าเครื่องคอมพิวเตอร์เพื่อใช้ในการทดสอบโปรแกรมประยุกต์ได้

3. Infrastructure as a Service: IaaS การใช้บริการโครงสร้างพื้นฐานระดับฮาร์ดแวร์ (Hardware) ในรูปแบบของเวอร์ชวลแมชชีน (Virtual Machine) เช่น ระบบเครือข่าย เครื่องแม่ข่า อุปกรณ์จัดเก็ข้อมูล รวมถึงทรัพยากรอื่นๆ ที่ใช้ในโครงสร้างพื้นฐาน เป็นต้น ซึ่งผู้ใช้บริการสามารถปรับเปลี่ยนจัดการระบบปฏิบัติการและโปรแกรมประยุกต์ต่างๆ แต่ไม่สามารถจัดการโครงสร้างพื้นฐานของระบบประมวลแบบกลุ่มเมฆได้

2.1.3 รูปแบบการใช้งาน มี 4 รูปแบบ ดังนี้

1. รูปแบบที่เป็นส่วนตัว (Private Cloud) เป็นการใช้ระบบโครงสร้างพื้นฐานของระบบประมวลแบบกลุ่มเมฆ ที่มีความเฉพาะภายในองค์กร โดยมีองค์กรเป็นเจ้าของ และเป็นผู้ดูแลเอง หรือการจ้างบุคคลภายนอกเข้ามาดูแล การติดตั้งโครงสร้างพื้นฐานอาจติดตั้งในสถานที่ของผู้ใช้บริการ หรืออาจติดตั้งภายนอก

2. รูปแบบที่เป็นชุมชน (Community Cloud) เป็นการใช้ระบบโครงสร้างพื้นฐานของระบบประมวลแบบกลุ่มเมฆ ที่มีการใช้งานเฉพาะด้าน หรือทำงานในลักษณะเดียวกัน โดยอนุญาตให้หลายองค์กรสามารถเข้าถึงแซะทำงานร่วมกัน ผู้ใช้บริการอาจมีผู้ดูแลและจัดการด้านทรัพยากรเอง หรือการจ้างบุคคลภายนอกเข้ามาดูแล การติดตั้งโครงสร้างพื้นฐานอาจะติดตั้งในสถานที่ภานในหรือภายนอกสถานที่ของสมาชิกได้

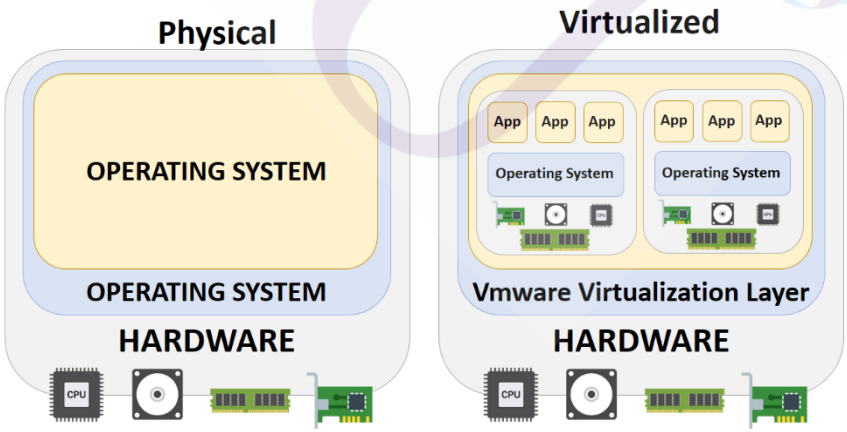
3. รูปแบบที่เป็นสาธารณะ (Public Cloud) เป็นการใช้ระบบโครงสร้างพื้นฐานของระบบประมวลผลแบบกลุ่มเมฆ ที่มีการจัดเตรียมสำหรับให้บุคคลทั่วไปใช้งาน โดยมีเจ้าของเป็นผู้บริหารจัดการเอง ซึ่งอาจจะเป็นหน่วงานภาคธุรกิจ ราชการ หรือสถาบันการศึกษา การติดตั้งโครงสร้างจะติดตั้งในสถานที่ของผู้ให้บริการ

4. รปแบบที่เป็นการผสม (Hybrid Cloud) เป็นการบริการที่ผสมผสานความแตกต่างของโครงสร้างพื้นระหว่างรูปแบบที่เป็นส่วนตัว รูปแบบที่เป็นชุม หรือรูปแบบที่เป็นสาธารณะ ให้สามารถทำงานร่วมกันได้

2.2 เทคโนโลยีเวอร์ชวลไลเซชัน

เทคโนโลยีเวอร์ชวลไลเซชัน เป็นเทคโนโลยีที่ทำให้คอมพิวเตอร์หนึ่งเครื่องสามารถแบ่งทรัพยากรของเครื่องคอมพิวเตอร์ให้เกิดสภาพแวดล้อมขึ้นใหม่หรือจำลองระบบเสมือนให้ใช้งานได้หลายระบบพร้อมกันบนทรัพยากรเดียวกัน เรียกว่าระบบคอมพิวเตอร์เสมือน (Changanti, 2007) โดยอาศัยหลักการสร้างอุปกรณ์เสมือนขึ้นมา คือ หน่วยประมวลผล หน่วยความจำ อุปกรณ์เครือข่าย ดิกส์ และอุปกรณ์ทางด้านเครือข่ายเสมือน เพื่อให้ระบบคอมพิวเตอร์หลักมองเห็นอุปกรณ์ดังกล่าว เสมือนเป็นอุปกรณ์ที่มีอยู่จริง นอกจากนี้ยังมีกลไกช่วยให้ระบบคอมพิวเตอร์เสมือนหลายระบบทำงานพร้อมกันได้อย่างต่อเนื่อง

แนวคิดเทคโนโลยีเวอร์ชวลไลเซชันเกิดขึ้นตั้งแต่ปี 1996 มีการใช้งานอย่างกว้างขว้างบนระบบเมนเฟรม (Mainframe) ของ IBM มีกระบวนการทำมัลติทาสก์ (Multitask) ที่สามารถทำงานหลายๆ แอพพลิเคชันในเวลาเดียวกัน โดยที่แต่ละสภาพแวดล้อมการประมวลผลนี้แต่ละตัวจะเรียกว่า เกส (Guest) และเรียกเครื่องคอมพิวเตอร์แม่ข่ายที่ใช้ประมวลผลว่า โฮสต์ (Host) ซอฟต์แวร์ที่ทำงานอยู่บนโฮสต์ทำหน้าที่เชื่อมต่อระหว่างโฮสต์และเกส ทำให้สามารถประมวลผลข้อมูลในสภาพแวดล้อมเรียกว่า เวอร์ชวลแมชชีน หรือ ไฮเปอร์ไวเซอร์ (Hypervisor)



ภาพที่ 2.2 เปรียบเทียบสถาปัตยกรรมระหว่างระบบคอมพิวเตอร์แบบปกติ และแบบใช้เทคโนโลยีเวอร์ชวลไลเซชัน

จากภาพที่ 2.2 แสดงให้เห็นถึงความแตกต่างระหว่างสถาปัตยกรรมคอมพิวเตอร์แบบปกติ และแบบเทคโนโลยีเวอร์ชวลไลเซชัน จะเห็นได้ว่า เครื่องคอมพิวเตอร์เสมือนแต่ละเครื่องจะมีการจำลองฮาร์ดแวร์เสมือนเป็นของแต่ละเครื่อง เพื่อใช้งานภายในฮร์ดแวร์ที่มีอยู่จริง

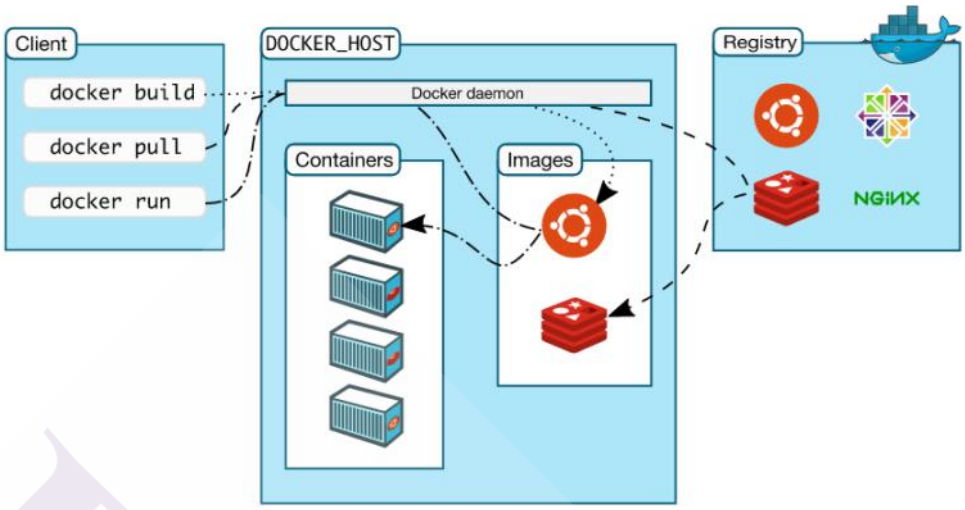
เวอร์ชวลแมทชีน คือโปรแกรมประยุกต์ที่ทำหน้าที่ในการจำลองเครื่องคอมพิวเตอร์เสมือนมากกว่าหนึ่งบนบนเครื่องคอมพิวเตอร์จริงเพียงเครื่องเดียว โดยเป็นการจำลองทรัพยากรต่างๆ เช่น หน่วยความจำ (CPU) หน่วยความจำหลัก (Main Memory) อุปกรณ์อินพุต เอาท์พุต (I/O Device) การทำงานของเวอร์ชวลแมชชีนต่างๆ สามารถที่จะทำงานพร้อมกันในเครื่องเดียวกัน และในการเข้าใช้ทรัพยากรเครื่องของเวอร์ชวลแมชชีนจะถูกควบคุมด้วยโปรแกรมที่เรียกว่า เวอร์ชวลแมทชีนมอนิเตอร์ (Virtual Machine Monitor: VMM)

เวอร์ชวลแมชชีนมอนิเตอร์ คือซอฟต์แวร์ที่ช่วยในการจัดการและจัดสรรการใช้ทรัพยากรของระบบร่วมกัน รวมถึงการแปลคำสั่งจากเวอร์ชวลแมชชีนไปเป็นคำสั่งระบบของเครื่อง

2.3 ด็อกเกอร์ (Docker)

ด็อกเกอร์ คือ ลินุกซ์คอนเทนเนอร์ (Linux Container: LXC) ชนิดหนึ่ง เป็นรูปแบบเทคนิคหนึ่งของการจำลองระบบเสมือน โดยไม่ต้องทำการติดตั้งโปรแกรมเวอร์ชวลแมทชีนหรือไฮเปอร์ไวเซอร์ ในการใช้งานของ LXC ทำให้สามารถสร้างแอพพลิเคชันให้ทำงานภายในคอนเทนเนอร์ได้จากการที่เนมสเปซ (namespace) ถูกแยกเป็นอิสระจากโปรเซสการทำงานอื่นๆ จุดประสงค์หลักของด็อกเกอร์คือช่วยให้นักพัฒนาสามารถที่จะสร้างแอพพลิเคชัน และนแอพพลิเคชันไปติดตั้งและให้ทำงานได้ง่ายขึ้น โดยที่คอนเทนเนอร์แต่ละตัวจะแยกการทำงานออกจากกัน ไม่ว่าจะเป็นโปรเซสการทำงาน และระบบเครือข่าย โดยใช้ทรัพยากรฮาร์ดแวร์เดียวกัน

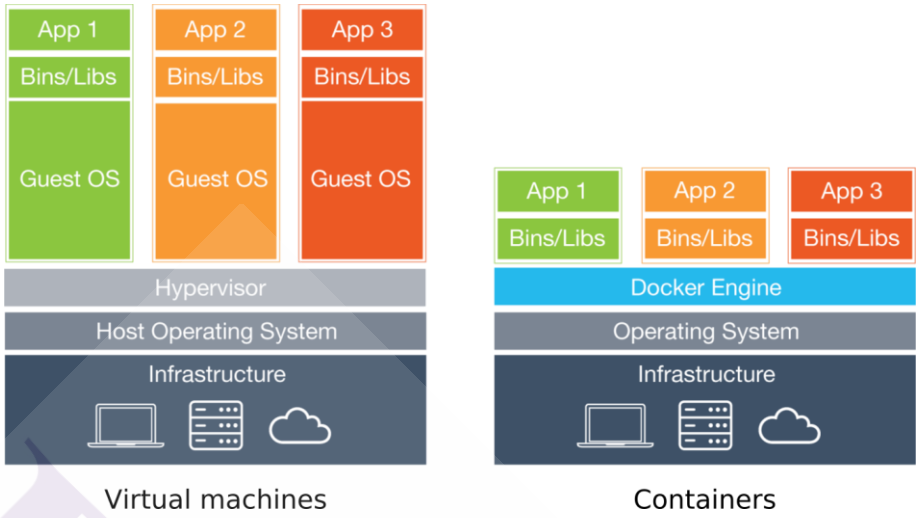
สถาปัตยกรรมของด็อกเกอร์เป็นแบบแม่ข่ายและลูกข่าย (Client Server) โดยมี Docker Client ทำหน้าที่เชื่อมต่อกับ Docker Engine เพื่อสิ่งการแอพพลิเคชันต่างๆ นอกจากนี้ Docker Engine สามารถรองรับการสั่งงานผ่านทางเซอร์วิท RESTful APIs ทำให้ง่ายต่อการสั่งการต่างๆ นอกจากนี้ด็อกเกอร์ยังมีส่วนประกอบที่สำคัญดังนี้ Docker Host, Docker Client, Docker Registry, Docker Daemon, Docker Images และ Docker Containers ดังภาพที่ 2.3



ภาพที่ 2.3 รูปแบบการให้บริการของระบบประมวลแบบกลุ่มเมฆ

ที่มา: https://docs.docker.com/engine/docker-overview/#docker-architecture

การทำงานของด็อกเกอร์ที่มีลักษณะการจำลองสภาพแวดล้อมเพื่อจำลองเครื่องคอมพิวเตอร์เสมือน มีลักษณะการทำงานคล้ายคลึงกับเวอร์ชวลแมชชีน เช่น VMWare. Virtual Box, XEN, KVM แต่มีข้อแตกต่างที่เห็นได้ชัดเจน คือ ด็อกเกอร์มีการใช้เทคโนโลยีคอนเทนเนอร์ในการจำลองสภาพแวดล้อมขึ้นมาเพื่อใช้งานสำหรับ 1 บริการ (Service) โดยไม่ต้องมีการติดตั้งระบบปฏิบัติการ แต่สำหรับเวอร์ชวลแมชชีนจะมีการจำลองสภาพแวดล้อมขึ้นมารวมถึงระบบปฏิบัติการด้วย



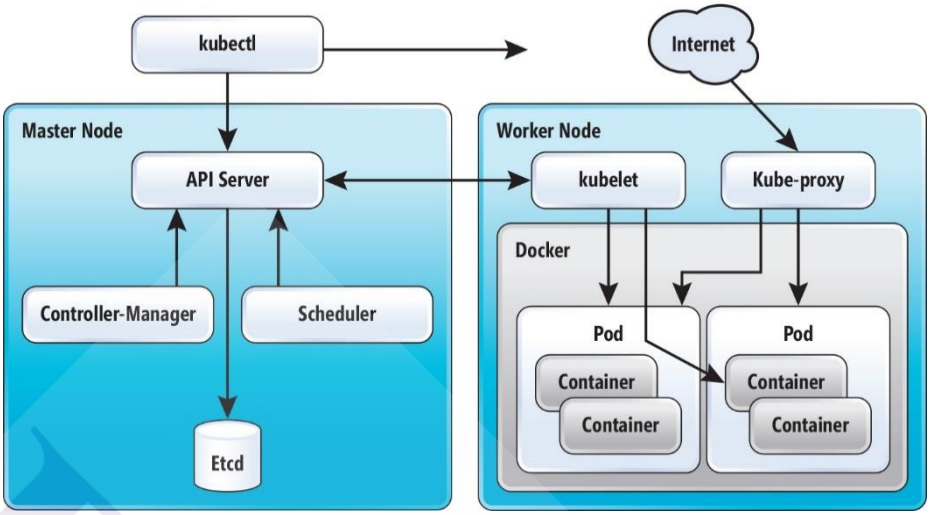
ภาพที่ 2.4 เปรียบเทียบสถาปัตยกรรมระหว่างเวอร์ชวลแมชชีนกับคอนเทนเนอร์

ที่มา: https://crs4.github.io/Galaxy4Developers/lectures/06.docker_and_galaxy/

ในการเปรียบเทียบประสิทธิภาพระหว่างคอนเทนเนอร์และเวอร์ชวลแมชชีนในเรื่องของความทนทานในการรองรับการเรียกใช้งานและการขยายการรองรับการใช้งาน (Scalable) พบว่าคอนเทนเนอร์สามารถรองรับการเรียกใช้งาน พร้อมทั้งมีประสิทธิภาพการขยายการรองรับการใช้งานและสามารถทำงานได้เร็วกว่าเวอร์ชวลแมชชีน (Ann Mary Joy, 2015) นอกจากนี้ประสิทธิภาพในการอัพเดตข้อมูลพบว่าคอนนเทนเนอร์ให้ประสิทธิภาพเทียบเท่าหรือมากกว่าเวอร์ชวลแมชชีน และ Overhead ในส่วนของหน่วยประมวลผล และหน่วยความจำของคอนเทนเนอร์มีค่าน้อยกว่าค่าของเวอร์ชวลแมชชีน (Wes Alex Ram Juan, 2015)

2.4 คูเบอร์เนตส์ (Kubernetes)

คูเบอร์เนตส์ หรือ K8s เป็นเครื่องมือช่วยบริหารจัดการคอนเทนเนอร์ที่ทำงานอยู่ในระบบสภาพแวดล้อมแบบคลัสเตอร์ (Cluster) โดยเป็นเครื่องมือโอเพ่นซอร์สที่บริษัทกูเกิ้ล (Google) พัฒนาขึ้น ถูกนำมาใช้ในขั้นตอนการนำคอนเทนเนอร์ไปใช้งาน (Deployment) สำหรับคูเบอร์เนตส์จะจัดการระบบเครือข่ายและจัดภาระการทำงานให้กับคอนเทนเนอร์ โดยที่ผู้ใช้งานจะสั่งการทางโปรแกรม kubectl และเซอร์วิสต่างๆ ซึ่งทำงานอยู่บนเครื่องมาสเตอร์ (Master Server)



ภาพที่ 2.5 โครงสร้างของคูเบอร์เนตส์

ที่มา: https://redmondmag.com/articles/2017/08/01/container-orchestration-with-kubernetes.aspx

โครงสร้างของคูเบอร์เนตส์มีส่วนประกอบดังนี้

2.4.1 เครื่องมาสเตอร์ (Master Server)

เครื่องมาสเตอร์เป็นส่วนที่ควบคุมการทำงานหลักของคูเบอร์เนตส์ โดยที่ผู้ใช้งานสามารถสั่งการต่างๆ ผ่านทางเครื่องมาสเตอร์ไปยังเครื่องมินเนี่ยนหรือเครื่องวอคเกอร์ (Worker/Minion Server) โดยเครื่องมาสเตอร์จะต้องมีโปรแกรมเพื่อทำงานกับคูเบอร์เนตส์ดังนี้

1. Etcd เป็นส่วนที่ให้บริการเก็บข้อมูลแบบ Key-Value ซึ่งแต่ละเครื่องเซิร์ฟเวอร์จะใช้ Etcd เป็นตัวกลางในการติดต่อสื่อสารภายในคลัสเตอร์ เช่น สภาพแวดล้อต่างๆ ของเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ด้วยกัน โดยการทำงานของ Etcd จะมีการเก็บข้อมูลในรูปแบบ HTTP/JSON เมื่อต้องการเรียกดูข้อมูลสามารถเรียกข้อมูลของ Key ที่ต้องการ โดย Etcd จะทำการตอบค่ากลับมา

2. API Service Server เป็นส่วนสำคัญของเครื่องมาสเตอร์ และเป็นส่วนที่จัดการงานต่างๆ ของเครื่องที่อยู่ภายในคลัสเตอร์ โดยจะทำการอ่านค่าสถานะต่างๆ จาก Etcd ในการเรียกใช้งาน API Service จะสามารถเรียกใช้งานผ่าน RESTful ซึ่งเป็นข้อได้เปรียบที่ทำให้ผู้พัฒนาสามารถพัฒนาเครื่องมือขึ้นมาติดต่อ API Service Server ได้เอง

3. Controller Manager Server เป็นส่วนที่ใช้ในการจัดการงานเกี่ยวกับการสร้างคอนเทนเนอร์ โดยจะทำการอ่านค่าสถานะต่างๆ ของคอนเทนเนอร์ผ่านทาง Etcd และเป็นส่วนที่ใช้ในการจัดการลดหรือการขยายการให้บริการ

4. Scheduler Server เป็นส่วนในการจัดการบริหารงานในแต่ละเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ โดยจะทำการตรวจสอบไม่ให้มีการใช้งานทรัพยากรเกินกว่าทรัพยากรที่มีว่างอยู่ และเมื่อมีการสร้างคอนเทนเนอร์ขึ้นมาใหม่ ส่วนนี้จะเป็นส่วนที่จัดการหาเครื่องมินเนี่ยนเพื่อรองรับคอนเทนเนอร์ใหม่ที่เกิดขึ้น

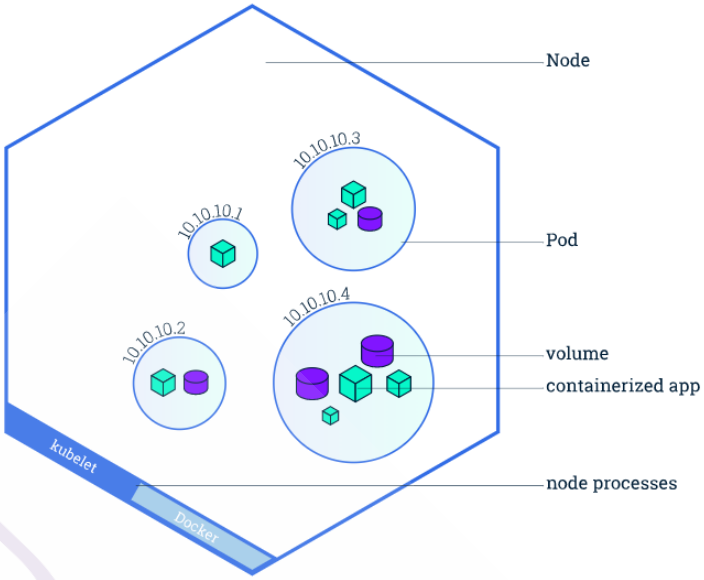
2.4.2 เครื่องมินเนี่ยนหรือเครื่องวอคเกอร์ (Worker/Minion Server)

เครื่องมินเนี่ยนจะทำหน้าที่รองรับงานจากเครื่องมาสเตอร์ โดยเชื่อมต่อกับเครื่องมาสเตอร์ผ่านทางเครือข่าย ซึ่งเครื่องมินเนี่ยนจะมีโปรแกรมด็อกเกอร์ทำงานอยู่ เมื่อมีการสร้างงานขึ้นมาใหม่จากเครื่องมาสเตอร์ เครื่องมินเนี่ยนจะดำเนินการสร้างด็อกเกอร์คอนเทนเนอร์ขึ้นมาตามที่ได้รับงานมา ซึ่งเครื่องมินเนี่ยนจะมีส่วนของการทำงานดังต่อไปนี้

1. Kubelet Service เป็นส่วนที่ติดต่อกับเครื่องมาสเตอร์เพื่อรับข้อมูลหรือรับคำสั่งต่างๆ เช่น การรับคำสั่งสร้างด็อกกเกอร์คอนเทนเนอร์

2. Proxy Service เป็นส่วนที่ทำหน้าที่ส่งการร้องขอการให้บริการจากด็อกเกอร์โฮสต์ไปยังด็อกเกอร์คอนเทนเนอร์ และทำการส่งค่าผลลัพท์กลับ ซึ่งในส่วนของ Proxy Service จะสามารถจัดการให้บริการในรูปแบบการกระจายงานได้ (Load Balancing)

3. หน่วยทำงานของคูเบอร์เนตส์ เป็นส่วนการทำงานของคอนเทนเนอร์ที่ทำงานอยู่บนคูเบอร์เนตส์ โดยประกอบไปด้วยหน่วยการทำงานดังนี้

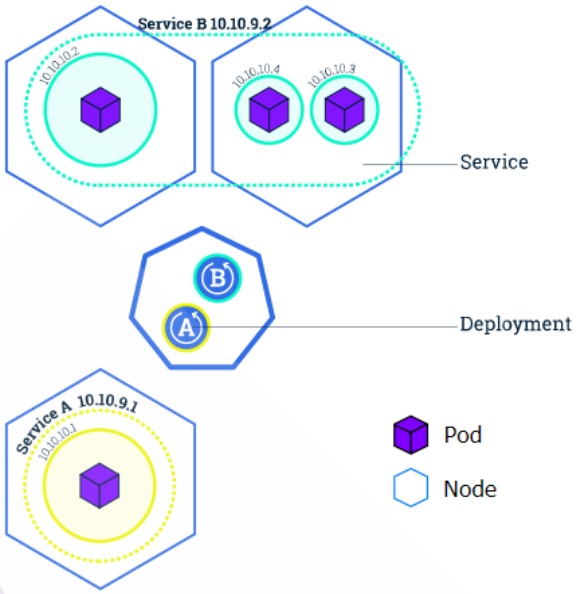


ภาพที่ 2.6 ลักษณะ Pods ของคูเบอร์เนสต์

ที่มา: https://kubernetes.io/docs/tutorials/kubernetes-basics/explore-intro/

3. หน่วยทำงานของคูเบอร์เนตส์ เป็นส่วนการทำงานของคอนเทนเนอร์ที่ทำงานอยู่บนคูเบอร์เนตส์ โดยประกอบไปด้วยหน่วยการทำงานดังนี้

- Pods คือชื่อเรียกของกลุ่มคอนเทนเนอร์ ในหนึ่ง Pods สามารถมีคอนเทนเนอร์ได้ตั้งแต่ 1 คอนเทนเนอร์ขึ้นไป แต่โดยทั่วไปนิยมให้มีแค่หนึ่งตัวในหนึ่ง Pods เพื่อใช้ในการรองรับการให้บริการแอพพลิเคชันที่ติดตั้งบนคอนเทนเนอร์ และสำหรับแอพพลิเคชันสามารถมีจำนวนมากกว่าหนึ่ง Pods โดยขึ้นอยู่กับความสามารถในการรองรับการให้บริการของแอพพลิเคชันนั้น



ภาพที่ 2.7 ลักษณะ Services ของคูเบอร์เนสต์

ที่มา: https://kubernetes.io/docs/tutorials/kubernetes-basics/expose-intro/

- Services คือส่วนของการเชื่อมต่อกับคอนเทนเนอร์ โดยส่วนนี้จะทำการส่งการร้องขอใช้บริการต่างๆ ไปยังคอนเทนเนอร์ที่อยู่ใน Pods หรือเรียกได้ว่า Services คือส่วนของอินเตอน์เฟสที่ใช้เชื่อมต่อระหว่างผู้ใช้บริการกับคอนเทนเนอร์ที่ให้บริการ

- Replication Controllers คือส่วนจัดการเพิ่มหรือลด Pods เป็นในลักษณะของการขยายขนาดหรือลดขนาดการให้บริการ ซึ่งการขยายหรือลดจะเป็นในลักษณะรูปแบบแนวนอน (Horizontal) และหน้าที่อีกส่วนคือจัดการในเรื่องรุ่นของแอพพลิเคชัน (Version) ที่ทำงานในคอนเทนเนอร์ เช่น เดิมมี Pods ที่ให้บริการแอพพลิเคชันรุ่นที่หนึ่ง ต่อมาผู้ให้บริการต้องการนำแอพพลิเคชันรุ่นที่สองขึ้นมาทำงานแทน และเมื่อแอพพลิเคชันรุ่นที่สองสามารถทำงานได้แล้ว Replication Controllers จะทำการปิดแอพพลิเคชันรุ่นที่หนึ่งลงโดยอัตโนมัติ

2.5 คอนเทนเนอร์เน็ตเวิร์คกิ้ง (Container Networking)

คอนเทนเนอร์เน็ตเวิร์คกิ้งถูกนำมาใช้ในการแก้ไขปัญหาการเชื่อมโยงระหว่างคอนเทนเนอร์ ซึ่งมีอยู่หลายวิธีการ สำหรับในงานวิจัยนี้นำเสนอวิธีการ 3 วิธี ดังนี้

2.5.1 Flannel

วิธีการหนึ่งที่ออกแบบมาสำหรับคูเบอร์เนตส์ เป็นวิธีที่ไม่ซับซ้อนในการติดตั้ง โดยการติดตั้ง Flannel ในแต่ละโฮสต์จะมีเอเจนต์ที่ชื่อว่า Flanneld มีหน้าที่ในการดูแลจัดสรรซับเน็ต (Subnet) ในแต่ละเครื่อง

การทำงานของ Flannel จะเรียกใช้ Kubernetes API กรือ Etcd เพื่อเก็บค่าการตั้งค่าของเน็ตเวิร์ค จัดสรรซับเน็ต และข้อมูลอื่นๆ เช่น Public IP ของโฮสต์ เป็นต้น และในการส่งต่อแพ็คเก็ตจะถูกส่งออกโดยใช้กระบวนการหลายอย่างรวมถึง VxLAN Encapsulation

Flannel ทำหน้าที่ในการจัดการเครือข่ายในระดับเลเยอร์ 3 ระหว่างหลายๆ โหนดในคลัสเตอร์ แต่ไม่สามารถจัดการคอนเทนเนอร์ที่เชื่อมต่อกับโฮสต์ได้ ทำได้เพียงเฉพาะวิธีการรับส่งข้อมูลระหว่างโฮสต์ สำหรับคูเบอร์เน็ตมองแต่ละ Pods มีความเป็นเอกลักษณ์ (Unique) และมีการทำ Routable IP ในคลัสเตอร์ ข้อดีของวิธีการนี้คือกำจัดความซับซ้อนในการทำ Port Mapping ที่มาจากการแชร์ IP จากโฮสต์เดียว

2.5.2 Calico

วิธีการที่เน้นทางด้านความปลอดภัยในการเชื่อมต่อสำหรับคอนเทนเนอร์และปริมาณงานที่เกิดขึ้นในเวอร์ชวลแมชชีน สำหรับ Calico มีการสร้างและจัดการเครือข่ายในระดับเลเยอร์ 3 ซึ่งจะมีการทำ Routable IP เต็มรูปแบบ สำหรับการส่งข้อมูลจะไม่ต้องใช้ IP Encapsulation หรือการแปลงที่อยู่ของเครือข่าย ทำให้สามารถทำงานร่วมกันได้ง่ายขึ้น

ในสภาพแวดล้อมที่เป็นในลักษณะซ้อนทับ (Overlay) วิธี Calico ใช้ IP-in-IP tunneling หรือสามารถทำงานกับเครือข่ายที่ตั้งค่าคอนเทนเนอร์เน็ตเวิร์คกิ้งประเภทอื่นได้ เช่น Flannel นอกจากนี้ Calico มีการประยุกต์ใช้กฎทางด้านความปลอดภัย (Network Policy) กับเวอร์ชวลอินเตอร์เฟสสำหรับควบคุมคอนเทนเนอร์และเวอร์ชวลแมชชีน รวมทั้งมีการใช้งานกับโฮสต์อินเตอร์เฟสสำหรับเซิร์ฟเวอร์และเวอร์ชวลแมชชีน

2.5.3 Canal

วิธีการที่รวมระหว่าง Calico กับ Flannel โดยในปัจจุบัน Canal เป็นเพียงรูปแบบการติดตั้งและการตั้งค่าเพื่อให้สามารถทำงานร่วมกันบนเครือข่ายเดียวกันได้ ซึ่งในอนาคตอาจจะมีการปรับเปลี่ยนรูปแบบรวมกับ Calico และ Flannel เพื่อลดความซับซ้อนในการติดตั้งและการตั้งค่า

2.6 Web Stress Simulator

โปรแกรม Web Stress Simulator เป็นโปรแกรมที่ใช้สำหรับจำลองสถานะการณ์ของโครงสร้างพื้นฐานของเว็บแอพพลิเคชัน ไม่ว่าจะเป็น การโหลดหน่วยประมวลผลซีพียูให้สูงขึ้น การใช้หน่วยความจำที่สูงขึ้น หรือ การรับส่งข้อมูลภายในเน็ตเวิร์คที่สูงขึ้น ตัวอย่างการใช้งาน โดยจะเรียกผ่าน URL พร้อมทั้งส่งค่าที่ต้องการ ดังนี้

- http://localhost:8080/web-stress-simulator-1.0.0/output?nbytes=3000

ใช้กรณีผู้ใช้งานต้องการไฟล์ข้อมูลขนาด 3 Kbytes โดยที่ Web Stress จะทำการสุ่มข้อมูลขนาด 3 Kbytes กลับมา

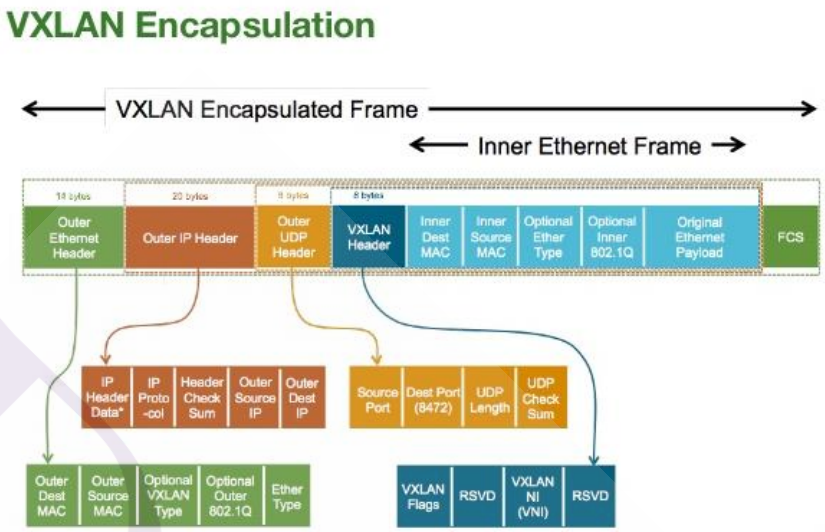
- http://localhost:8080/web-stress-simulator-1.0.0/output?nbytes=3000 &time=60000

ใช้กรณีผู้ใช้งานต้องการไฟล์ข้อมูลจาด 3 Kbytes มีอัตราการส่งข้อมูล 0.5 b/s ภายในเวลา 60 วินาที ซึ่ง URL นี้เหมาะกับการทดสอบเครื่องภายใต้เงื่อนไขที่ความเร็วในการเชื่อมต่อต่ำ

- http://localhost:8080/web-stress-simulator-1.0.0/memory?nbytes=10000000 &time=5000

ใช้กรณีผู้ใช้งานต้องการจำลองให้มีการใช้งานหน่วยความจำขนาด 10 MB เป็นเวลา 5 วินาที

2.7 VxLAN Encapsulation



ภาพที่ 2.8 VxLAN Encapsulation Frame

ที่มา: https://image.slidesharecdn.com/2014-06-10-nsx-preso-140610164242-phpapp01/95/an-introduction-to-vmware-nsx-18-638.jpg?cb=1402418606

กระบวนการ Vxlan (Virtual Extensible LAN) Encapsulation เป็นกระบวนการที่จะช่วยให้สามารถทำการส่งข้อมูลในระดับเลเยอร์ 2 ข้ามไประหว่างเครือข่ายเลเยอร์ 3 ได้ โดยหลักการทำงานเป็นการห่อหุ้ม (encapsulation) แพ็คเก็ตด้วยแพ็คเก็ตของเลเยอร์ 3 เพื่อทำการส่งข้อมูลบนเครือข่าย เลเยอน์ 3 ซึ่งส่งผลให้แพ็คเก็ตนั้นมีเฮดเดอร์ 2 ชั้น โดยชั้นนอกจะเป็น IP ของ VxLAN ที่ใช้สำหรับส่งข้อมูลจากต้นทางไปยังปลายทางข้ามระหว่างเครือข่ายเลเยอร์ 3 ส่วนเฮดเดอร์ชั้นที่ 2 จะเป็นข้อมูลเดิมของแพ็คเก็ตที่ต้นทางต้องการส่งข้อมูลไปยังปลายทาง

2.8 โปรแกรม Apache JMeter

โปรแกรม Apache JMeter เป็นโปรแกรมโอเพนซอร์สที่พัฒนาโดย Apache Software Foundation พัฒนาโดยภาษา Java และใช้เพื่อวัดประสิทธิภาพของระบบที่บริการอยู่บนเครื่องเซิร์ฟเวอร์ โดยสามารถจำลองการร้องขอการใช้บริการจากเครื่องเซิร์ฟเวอร์ได้หลากหลายโปรโตคอล เช่น HTTP, SMTP หรือ POP 3 เป็นต้น พร้อมทั้งยังสามารถกำหนดจำนวนผู้ร้องขอช่วงเวลาการร้องขอ ซึ่งผลลัพท์ที่ได้จะแสดงเป็นข้อมูลผลลัพท์ที่เครื่องเซิร์ฟเวอร์สามารถให้บริการได้

2.9 งานวิจัยที่เกี่ยวข้อง

2.9.1 งานวิจัยเรื่อง Measurement and Evaluation for Docker Container Networking (Hao Zeng, Baosheng Wang, Wenping Deng, Weiqi Zhang, 2017) ได้ทำการวัดประสิทธิภาพของคอนเทนเนอร์เน็ตเวิร์คระหว่าง Flannel, Swarm Overlay และ Calico โดยมีการวัดประสิทธิภาพทางด้านเวลาในการตอบกลับของการ Ping พบว่า Calico ใช้เวลาในการตอบกลับเร็วที่สุด ตามมาด้วย Swarm Overlay และFlannel ใช้เวลาตอบกลับช้าที่สุด นอกจากนี้ในงานวิจัยได้ทำการวัดประสิทธิภาพทางด้าน TCP และ UDP Throughput พบว่า Calico ให้ผลลัพท์ดีที่สุด

จากการวิเคราะห์งานวิจัยนี้ พบว่างานวิจัยนี้ไม่ได้กล่าวถึงการวัดประสิทธิภาพของคอนเทนเนอร์เน็ตเวิร์คทางด้านเวลาที่ใช้ในการตอบสนองกลับจากระบบเว็บแอพพลิเคชันในการร้องขอไฟล์ที่มีขนาดแตกต่างกัน

2.9.2 งานวิจัยเรื่อง Docker Based Overlay Network Performance Evaluation in Large Scale Streaming (Bin Xie, Guanyi Sun, Guo Ma, 2016) ได้ทำการวัดประสิทธิภาพระหว่าง Host Mode กับ Overlay Network Mode บนแอพพลิเคชันที่เป็นรูปแบบ Streaming พบว่าในด้านการทำงานของ Host Mode ให้ประสิทธิภาพดีกว่า แต่ประเภท Overlay Network Mode ให้การทำงานที่มีเสถียรภาพมากกว่า ในด้านการถ่ายโอนข้อมูลพบว่า Overlay Network Mode ให้ผลลัพท์ที่ดีกว่า

จากการวิเคราะห์งานวิจัยนี้ พบว่างานวิจัยนี้ไม่ได้กล่าวถึงประเภทของ Overlay Network Mode ที่ใช้ในการทดลอง

2.9.3 งานวิจัยเรื่อง Assessing the Value of Containers for NFVs: A Detailed Network Performance Study (Struye, Spinnewyn, Spaey, Bonjean, Latre, 2017) ได้ทำการวัดประสิทธิภาพเครือข่ายของคอนเทนเนอร์ระหว่างด็อกเกอร์ RKT และ LXC โดยทำการประเมินค่า Throughput และ Latency จากการกำหนดเน็ตเวิร์คเป็น Host Mode, Bridge (NAT) Mode และ Macvlan พบว่า

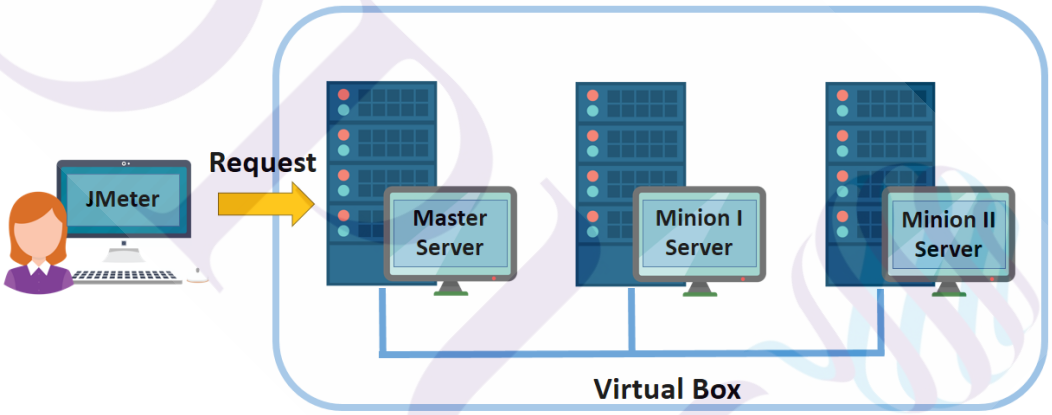
ในด้านของ Throughput ทางด้าน LXC ให้ประสิทธิภาพดีที่สุด แต่ทางด้านเครือข่าย Macvlan ให้ประสิทธิภาพดีที่สุด

จากการวิเคราะห์งานวิจัยนี้ พบว่างานวิจัยนี้เป็นการวัดประสิทธิภาพเทียบระหว่างตัวจัดการคอนเทนเนอร์ โดยไม่ได้กล่าวถึงประสิทธิภาพในด้านการตอบสนองทางด้านเวลา

บทที่ 3
ระเบียบวิธีวิจัย

ในบทนี้จะกล่าวถึง ขั้นตอนการใช้ด็อกเกอร์คอนเทนเนอร์สร้างระบบการประมวลผลคลาวด์ด้วยคูเบอร์เนตส์ เพื่อใช้ในการทดสอบประสิทธิภาพทางด้านเวลาในการตอบสนองกลับในการให้เว็บจากวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ของคอนเทนเนอร์ รวมทั้งอธิบายแนวทางการวิจัยและพัฒนา เครื่องมือที่ใช้ในงานวิจัย แผนการดำเนินงาน ขั้นตอนและวิธีการดำเนินงาน

3.1 โครงสร้างภาพรวมของระบบการประมวลผลคลาวด์ด้วยคูเบอร์เนตส์



ภาพที่ 3.1 โครงสร้างภาพรวมของระบบการประมวลผลคลาวด์

จากภาพที่ 3.1 โครงสร้างภาพรวมของระบบการประมวลผลคลาวด์ โดยการจำลองการเข้าใช้บริการระบบในระบบการประมวลผลคลาวด์ เพื่อทำการทดสอบประสิทธิภาพของวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ โดยแต่ละส่วนมีรายละเอียดดังนี้

3.1.1 โครงสร้างของระบบการประมวลผลคลาวด์ประกอบไปด้วยเครื่องเวอร์ชวลแมชีน (Virtual Machine: VM) จำนวน 3 เครื่องที่ได้ติดตั้งอยู่บนเครื่องหลัก โดยแบ่งเป็นเครื่องมาสเตอร์เซิร์ฟเวอร์ (Master Server) จำนวน 1 เครื่อง และเครื่องมินเนี่ยนเซิร์ฟเวอร์ (Minion Server) จำนวน 2 เครื่อง

3.1.2 ส่วนที่ใช้ในการจำลองพฤติกรรมการใช้งานของผู้ใช้บริการระบบประมวลผลคลาวด์ โดยเป็นการติดตั้ง Apache JMeter ที่เครื่องหลัก (Host)

3.2 แนวทางการวิจัยและพัฒนา

3.2.1 ค้นคว้า ศึกษาและรวบรวมงานวิจัยที่เกี่ยวข้องกับการเปรียบเทียบประสิทธิภาพของวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์

ศึกษาวิธีการดำเนินงานของงานวิจัยที่เกี่ยวกับทางด้านการเปรียบเทียบประสิทธิภาพของวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ เพื่อนำมาเป็นแนวทางในการจัดทำงานวิจัย

3.2.2 ศึกษาเทคโนโลยีด็อกเกอร์คอนเทนเนอร์

เพื่อเป็นความรู้พื้นฐานในการนำมาพัฒนาระบบการประมวลผลคลาวด์

3.2.3 ศึกษาเครื่องมือที่ใช้ในการจัดการด็อกเกอร์คอนเทนเนอร์

ศึกษาและรวบรวมเครื่องมือที่ใช้ในการจัดการด็อกเกอร์คอนเทนเนอร์เพื่อพัฒนาระบบจัดการประมวลผลคลาวด์ โดยในงานวิจัยชิ้นนี้ใช้คูเบอร์เนตส์เป็นตัวจัดการ

3.2.4 ศึกษาวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ของคูเบอร์เนตส์

ศึกษาหลักการทำงาน และวิธีการตั้งค่า ของแต่ละวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ โดยในงานวิจัยชิ้นนี้เลือกวิธีการเชื่อม 3 วิธี คือ Flannel Calico และ Canal

3.2.5 ออกแบบโครงสร้างของระบบการประมวลผลคลาวด์

ดำเนินการออกแบบ และจัดทำระบบการประมวลผลคลาวด์ด้วยคูเบอร์เนตส์เพื่อใช้ในการทดสอบ โดยใช้ข้อมูลต่างๆ ที่ได้กล่าวไปแล้วข้างต้น

3.2.6 เปรียบเทียบ วิเคราะห์ผลที่ได้ และสรุป

เมื่อทำการสร้างระบบที่ใช้ในการทดสอบแล้วเสร็จ ทำการทดสอบและเก็บข้อมูลประสิทธิภาพของวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ทั้ง 3 วิธี เพื่อนำผลที่ได้มาวิเคราะห์ พร้อมทั้งสรุปผลงานวิจัย

3.2.7 รวบรวมข้อมูลที่ได้ทั้งหมดจัดทำวิทยานิพนธ์

ดำเนินการรวบรวมข้อมูลทั้งหมดที่ได้มาจากการดำเนินงานตั้งแต่ต้น มาเรียบเรียงเพื่อจัดทำเป็นวิทยานิพนธ์ โดยมีแผนการดำเนินงานในตารางที่ 3.1

ตารางที่ 3.1 แผนการดำเนินงาน

	หัวข้อ	ช่วงเดือน / ปี (พ.ศ.)				
		ม.ค. 2561	ก.พ. 2561	มี.ค. 2561	เม.ย. 2561	พ.ค. 2560
1	ค้นคว้า ศึกษาและรวบรวมงานวิจัยที่เกี่ยวข้องกับการเปรียบเทียบประสิทธิภาพของวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์					
2	ศึกษาเทคโนโลยีด็อกเกอร์คอนเทนเนอร์					
3	ศึกษาเครื่องมือที่ใช้ในการจัดการด็อกเกอร์คอนเทนเนอร์					
4	ศึกษาวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ของคูเบอร์เนตส์					
5	ออกแบบโครงสร้างของระบบการประมวลผลคลาวด์					
6	เปรียบเทียบ วิเคราะห์ผลที่ได้ และสรุป					
7	รวบรวมข้อมูลที่ได้ทั้งหมดจัดทำวิทยานิพนธ์					

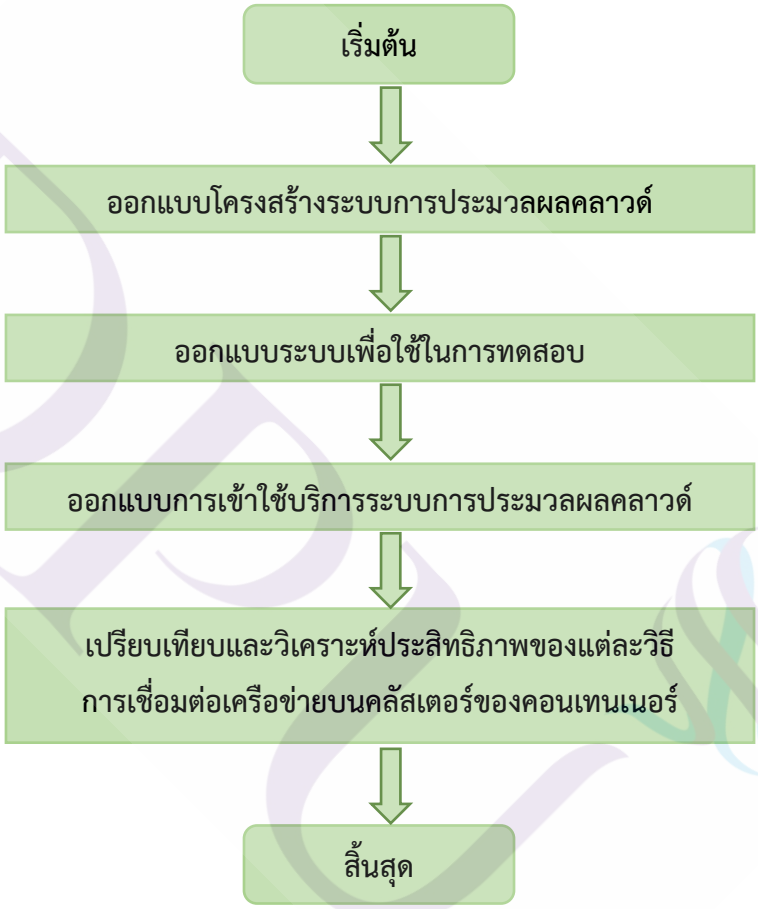
3.3 เครื่องมือที่ใช้ในงานวิจัย

ซอฟต์แวร์ที่ใช้ในงานวิจัย มีรายละเอียดดังนี้

1. โปรแกรม Oracle VM VirtualBox 5.1.22
   - CPUs
   - Memory 2 GB
   - Hard Disk 50 GB
2. Ubuntu Desktop 16.04 LTS

3. Kubernetes
4. Web Stress Simulator
5. Apache Jmeter 4.0

3.4 ขั้นตอนและวิธีการดำเนินงาน



ภาพที่ 3.2 แผนผังการดำเนินงานโดยรวมของระบบ

3.4.1 ออกแบบโครงสร้างระบบการประมวลผลคลาวด์

ในการออกแบบโครงสร้างระบบการประมวลผลคลาวด์ สำหรับงานวิจัยนี้ใช้เครื่องคอมพิวเตอร์พกพาจำนวน 1 เครื่อง ทำการติดตั้งโปรแกรม Oracle VM VirtualBox และสร้างเวอร์ชวลแมชีนจำนวน 3 เครื่อง เป็นจำนวน 3 ชุด เพื่อให้บริการระบบการประมวลผลคลาวด์ในแต่ละวิธีการเชื่อมต่อเครือข่ายบนคลัสเตอร์ แต่ละเครื่องกำหนดให้มี CPU 2 Core, Ram 2 GB และ Interface Network ขนาด 100 Mb/s พร้อมทั้งติดตั้งระบบปฏิบัติการ Ubuntu Server 16.04 LTS โดยมีการตั้งค่าหมายเลข IP ดังตารางที่ 3.2

ตารางที่ 3.2 ตาราง IP ของโครงสร้างระบบการประมวลผลคลาวด์ที่ใช้ในงานวิจัย

คอนเทนเนอร์เน็ตเวิร์ค	Master Server	Minion Server I	Minion Server II
Flannel	192.168.56.11	192.168.56.12	192.168.56.13
Calico	192.168.56.21	192.168.56.22	192.168.56.23
Canal	192.168.56.31	192.168.56.32	192.168.56.33

จากนั้นทำการติดตั้งด็อกเกอร์คอนเทนเนอร์ และติดตั้งเครื่องมือคูเบอร์เนตส์เพื่อสร้างระบบการประมวลผลคลาวด์ โดยตั้งค่าเวอร์ชวลแมชีนให้เป็นเครื่องมาสเตอร์ 1 เครื่อง และเครื่องมินเนี่ยน 2 เครื่อง ซึ่งในเครื่องมาสเตอร์จะเกิด Pods หลักๆ คือ Etdc, Proxy, Controller และ Scheduler ในเครื่องมินเนี่ยนจะเกิด Proxy Pods ดังแสดงในภาพที่ 3.3

ภาพที่ 3.3 โครงสร้างของระบบประมวลผลคลาวด์ด้วยคูเบอร์เนตส์ที่ออกแบบ

หลังจากสร้างเครื่องมาสเตอร์และเครื่องมินเนี่ยน ทำการตั้งค่าคูเบอร์เน็ตคลัสเตอร์ (Kubernetes Cluster) หรือคอนเทนเนอร์เน็ตเวิร์คระหว่างเครื่องมาสเตอร์และเครื่องมินเนี่ยน ในงานวิจัยนี้เลือกวิธีการเชื่อมต่อ 3 วิธี คือ

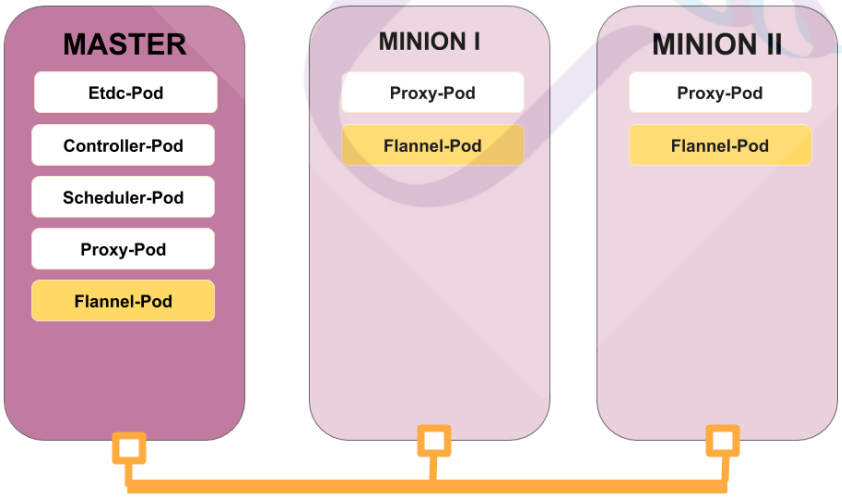
1. Flannel กำหนด Subnet Network เป็น 10.244.0.0/16
2. Calico กำหนด Subnet Network เป็น 192.168.0.0/16
3. Canal กำหนด Subnet Network เป็น 10.244.0.0/16

Flannel Calico และ Canal หลังจากติดตั้งเสร็จสามารถเชื่อมต่อระหว่างเครื่องมาสเตอร์และเครื่องมินเนี่ยนทั้ง 2 เครื่องได้ โดยสามารถเรียกดูการเชื่อมต่อได้ ดังภาพที่ 3.4

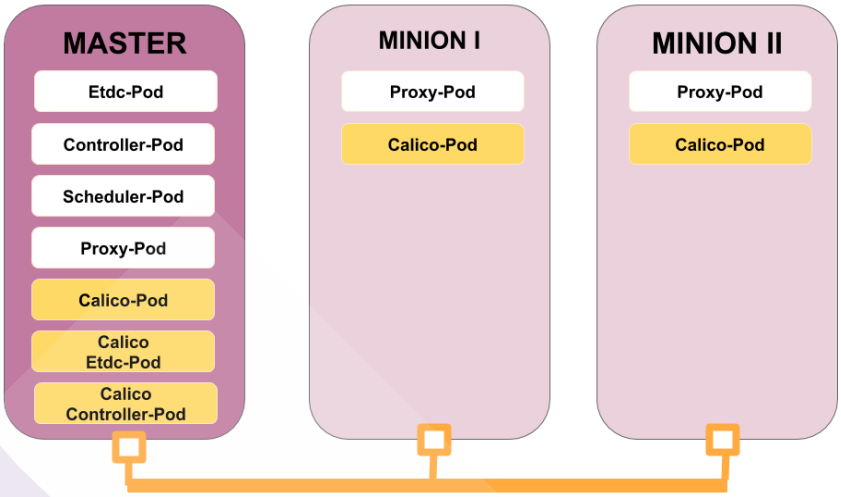
```
root@master-node:~# kubectl get nodes
NAME           STATUS    ROLES     AGE       VERSION
master-node    Ready     master    3h        v1.10.0
worker-node1   Ready     <none>    3m        v1.10.0
worker-node2   Ready     <none>    37s       v1.10.0
```

ภาพที่ 3.4 หน้าจอแสดงผลการเชื่อมต่อระหว่างเครื่องมาสเตอร์และเครื่องมินเนี่ยน 2 เครื่อง

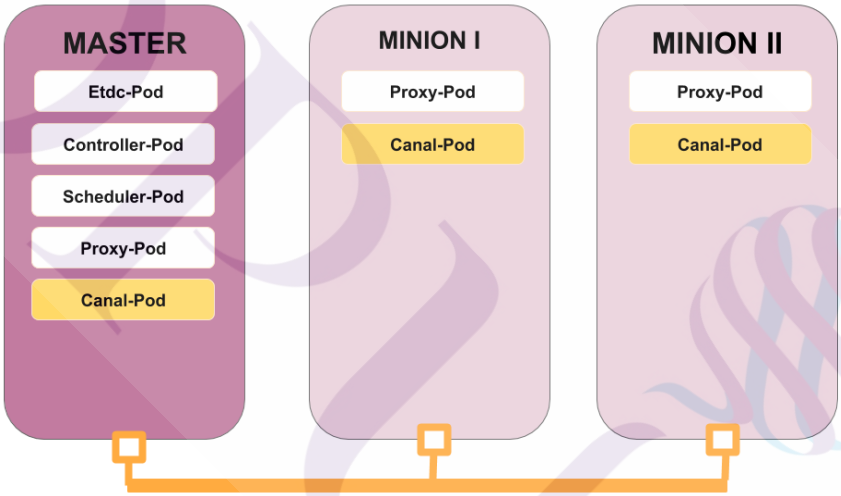
การเชื่อมต่อแต่ละวิธีจะเกิด Pods ของคอนเทนเนอร์เน็ตเวิร์คเพิ่มขึ้นมา ในแต่ละเครื่องมาสเตอร์และเครื่องมินเนี่ยน ดังภาพต่อไปนี้



ภาพที่ 3.5 รูปแบบการใช้งานของคอนเทนเนอร์เน็ตเวิร์ควิธี Flannel



ภาพที่ 3.6 รูปแบบการใช้งานของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico



ภาพที่ 3.7 รูปแบบการใช้งานของคอนเทนเนอร์เน็ตเวิร์ควิธี Canal

สามารถเรียกดู Pods ทั้งหมดที่เกิดขึ้นจากเครื่องมาสเตอร์ของแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์ค ได้ดังนี้

```
root@kubeadm-master1:~# kubectl get pods --namespace=kube-system -o wide
NAME                                    READY     STATUS    RESTARTS   AGE       IP             NODE
etcd-kubeadm-master1                    1/1       Running   13         43d       10.0.5.11      kubeadm-master1
kube-apiserver-kubeadm-master1          1/1       Running   13         43d       10.0.5.11      kubeadm-master1
kube-controller-manager-kubeadm-master1 1/1       Running   13         43d       10.0.5.11      kubeadm-master1
kube-dns-6f4fd4bdf-pdg7m                3/3       Running   39         43d       10.244.0.15    kubeadm-master1
kube-flannel-ds-57w99                   1/1       Running   13         43d       10.0.5.13      kubeadm-minion2
kube-flannel-ds-dtf69                   1/1       Running   18         43d       10.0.5.11      kubeadm-master1
kube-flannel-ds-h4rt9                   1/1       Running   15         43d       10.0.5.12      kubeadm-minion1
kube-proxy-6fs75                        1/1       Running   13         43d       10.0.5.11      kubeadm-master1
kube-proxy-fhtb4                        1/1       Running   12         43d       10.0.5.12      kubeadm-minion1
kube-proxy-jtg5b                        1/1       Running   12         43d       10.0.5.13      kubeadm-minion2
kube-scheduler-kubeadm-master1          1/1       Running   13         43d       10.0.5.11      kubeadm-master1
kubernetes-dashboard-5bd6f767c7-pgbmj   1/1       Running   12         43d       10.244.2.81    kubeadm-minion2
root@kubeadm-master1:~#
```

ภาพที่ 3.8 หน้าจอแสดง Pods ที่เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Flannel

```
root@calico-master:~# kubectl get pods --namespace=kube-system -o wide
NAME                                    READY     STATUS    RESTARTS   AGE       IP              NODE
calico-etcd-l5csc                       1/1       Running   1          32d       10.0.5.21       calico-master
calico-kube-controllers-559b575f97-pg4hs 1/1      Running   13         41d       10.0.5.21       calico-master
calico-node-g9jxk                       2/2       Running   31         41d       10.0.5.23       calico-minion2
calico-node-mb9wr                       2/2       Running   21         41d       10.0.5.21       calico-master
calico-node-w8s97                       2/2       Running   27         41d       10.0.5.22       calico-minion1
etcd-calico-master                      1/1       Running   7          41d       10.0.5.21       calico-master
kube-apiserver-calico-master            1/1       Running   7          41d       10.0.5.21       calico-master
kube-controller-manager-calico-master   1/1       Running   7          41d       10.0.5.21       calico-master
kube-dns-6f4fd4bdf-qwz6m                3/3       Running   18         41d       192.168.255.14  calico-master
kube-proxy-c9lbd                        1/1       Running   10         41d       10.0.5.23       calico-minion2
kube-proxy-gkc7d                        1/1       Running   9          41d       10.0.5.22       calico-minion1
kube-proxy-lqk5t                        1/1       Running   7          41d       10.0.5.21       calico-master
kube-scheduler-calico-master            1/1       Running   7          41d       10.0.5.21       calico-master
kubernetes-dashboard-5bd6f767c7-lrzl6   1/1       Running   7          41d       192.168.255.15  calico-master
root@calico-master:~#
```

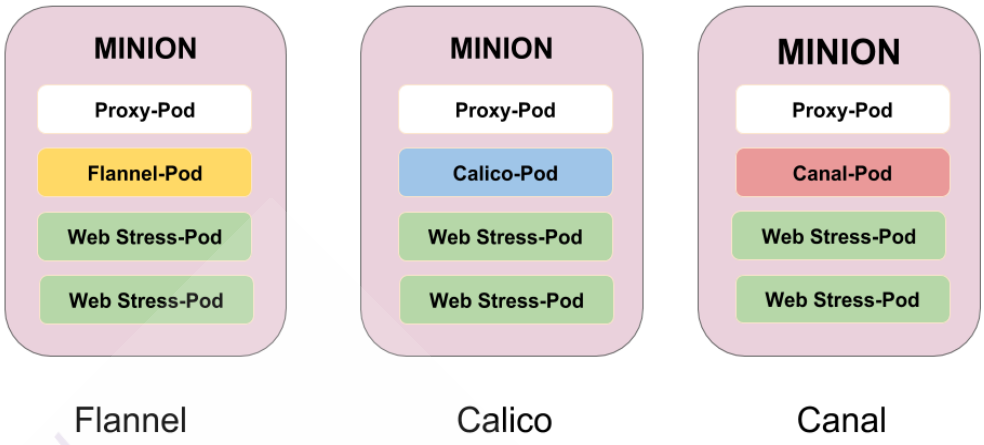
ภาพที่ 3.9 หน้าจอแสดง Pods ที่เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico

```
root@canel-master1:~# kubectl get pods --namespace=kube-system -o wide
NAME                                    READY     STATUS    RESTARTS   AGE       IP            NODE
canal-d2pps                             3/3       Running   16         41d       10.0.5.32     canel-minion1
canal-twgjh                             3/3       Running   12         41d       10.0.5.31     canel-master1
canal-z62wm                             3/3       Running   13         41d       10.0.5.33     canel-minion2
etcd-canel-master1                      1/1       Running   4          41d       10.0.5.31     canel-master1
kube-apiserver-canel-master1            1/1       Running   4          41d       10.0.5.31     canel-master1
kube-controller-manager-canel-master1   1/1       Running   4          41d       10.0.5.31     canel-master1
kube-dns-6f4fd4bdf-vmmlk                3/3       Running   12         41d       10.244.0.6    canel-master1
kube-proxy-7n9w7                        1/1       Running   5          41d       10.0.5.32     canel-minion1
kube-proxy-97frs                        1/1       Running   4          41d       10.0.5.33     canel-minion2
kube-proxy-l5c8n                        1/1       Running   4          41d       10.0.5.31     canel-master1
kube-scheduler-canel-master1            1/1       Running   4          41d       10.0.5.31     canel-master1
root@canel-master1:~#
```

ภาพที่ 3.10 หน้าจอแสดง Pods ที่เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Canal

3.4.2 ออกแบบระบบเพื่อใช้ในการทดสอบ

หลังจากติดตั้งระบบการประมวลผลคลาวด์ด้วยคูเบอร์เนตส์แล้ว ทำการติดตั้ง Web Application เพื่อให้บริการข้อมูลด้วย Web-Stress-Simulator เพื่อใช้ในการทดสอบ โดยทำการติดตั้งที่เครื่องมินเนี่ยนทั้ง 2 เครื่อง ของแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์ค เครื่องละ 2 Pods ดังแสดงในภาพที่ 3.11



ภาพที่ 3.11 โครงสร้างของระบบแต่ละวิธีคอนเทนเนอร์ที่ใช้ในการทดสอบ

สามารถเรียกดู Pods ที่ได้จากการติดตั้ง Web Stress ที่เกิดขึ้นจากเครื่องมาสเตอร์ของแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์ค ได้ดังนี้

```
root@kubeadm-master1:~# kubectl get pod -o wide
NAME                                    READY   STATUS    RESTARTS   AGE     IP            NODE
webstress-deployment-6f84c56594-4r76m   1/1     Running   8          41d     10.244.2.82   kubeadm-minion2
webstress-deployment-6f84c56594-kfjld   1/1     Running   8          41d     10.244.2.83   kubeadm-minion2
webstress-deployment-6f84c56594-n7fvx   1/1     Running   7          41d     10.244.1.37   kubeadm-minion1
webstress-deployment-6f84c56594-ngjrr   1/1     Running   7          41d     10.244.1.38   kubeadm-minion1
```

ภาพที่ 3.12 หน้าจอแสดง Webstress Pods ที่เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Flannel

```
root@calico-master:~# kubectl get pod -o wide
NAME                                    READY   STATUS    RESTARTS   AGE     IP               NODE
webstress-deployment-6f84c56594-779lp   1/1     Running   1          32d     192.168.168.67   calico-minion2
webstress-deployment-6f84c56594-8j6dk   1/1     Running   1          32d     192.168.168.127  calico-minion2
webstress-deployment-6f84c56594-8jjp9   1/1     Running   1          32d     192.168.84.251   calico-minion1
webstress-deployment-6f84c56594-dczgj   1/1     Running   1          32d     192.168.84.252   calico-minion1
root@calico-master:~# kubectl get pod
```

ภาพที่ 3.13 หน้าจอแสดง Webstress Pods ที่เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico

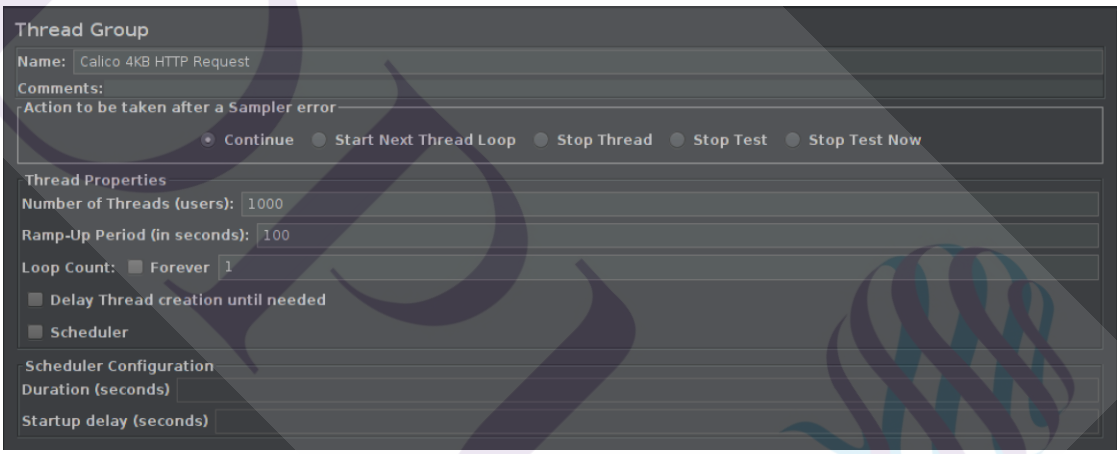
```
root@canel-master1:~# kubectl get pod -o wide
NAME                                    READY   STATUS    RESTARTS   AGE     IP            NODE
webstress-deployment-6f84c56594-2np9z   1/1     Running   4          41d     10.244.1.10   canel-minion1
webstress-deployment-6f84c56594-49tpg   1/1     Running   3          40d     10.244.2.11   canel-minion2
webstress-deployment-6f84c56594-4mmz8   1/1     Running   4          41d     10.244.1.11   canel-minion1
webstress-deployment-6f84c56594-p2cbf   1/1     Running   3          41d     10.244.2.10   canel-minion2
```

ภาพที่ 3.14 หน้าจอแสดง Webstress Pods ที่เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Canal

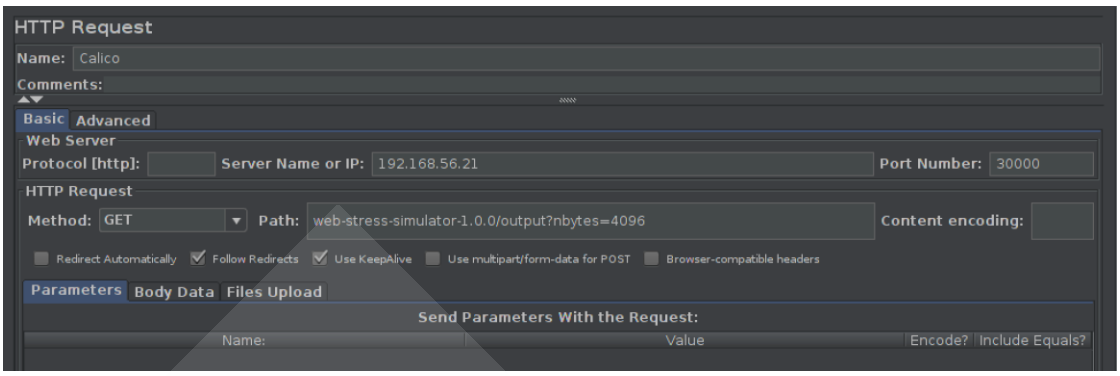
3.4.3 ออกแบบการเข้าใช้บริการระบบการประมวลผลคลาวด์

หลังจากติดตั้งระบบการประมวลผลคลาวด์ด้วยคูเบอร์เนตส์ และติดตั้งระบบเพื่อใช้ในการทดสอบ ดำเนินการติดตั้งโปรแกรม Apache JMeter ไว้ที่เครื่องโฮสต์เพื่อใช้เป็นเครื่องมือที่ใช้สำหรับจำลองพฤติกรรมการร้องขอเข้าใช้บริการ (Request) จากเครื่องโฮสต์ไปยังเครื่องมาสเตอร์ผ่านทางโปรโตคอล HTTP ในการจำลองการร้องขอเข้าใช้บริการในแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์ค ซึ่งมีรายละเอียดการจำลองดังนี้

1. กำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 1,000 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 100 วินาที ตัวอย่างดังภาพที่ 3.15 โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 4 Kbytes ตัวอย่างดังภาพที่ 3.16 และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 4 Kbytes กลับมาให้จนครบ



ภาพที่ 3.15 หน้าจอ JMeter กำหนดจำนวนร้องขอเข้าใช้บริการ 1,000 ครั้ง



ภาพที่ 3.16 หน้าจอ JMeter กำหนด IP Address และพาธที่เข้าถึงเว็บไซต์

จากภาพที่ 3.16 เป็นการสร้าง HTTP Request เพื่อเป็นโปรโตคอลที่ใช้ในการร้องขอใช้บริการ โดยมีการกำหนด Server Name or IP เป็นเครื่องมาสเตอร์ตามแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คที่กำหนดไว้ ตัวอย่าง IP Address ที่แสดงเป็น IP Address เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico และการร้องขอข้อมูลขนาด 4 Kbytes จาก Web Stress Simulator ทำการกำหนดพาธเป็น web-stress-simulator-1.0.0/output?nbytes=4096

ผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 4 Kbytes เป็นจำนวน 1,000 ครั้งภายในเวลา 100 วินาที แต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คได้ให้ผลการทดลองทางด้านเวลาในการตอบสนองดังรูปที่ 3.17, 3.18 และ 3.19

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
Flannel	1000	2	2	52	1.88	0.00%
TOTAL	1000	2	2	52	1.88	0.00%

ภาพที่ 3.17 หน้าจอ JMeter แสดงเวลาในการตอบสนองกลับของคอนเทนเนอร์เน็ตเวิร์ควิธี Flannel

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
Calico	1000	3	2	79	4.40	0.00%
TOTAL	1000	3	2	79	4.40	0.00%

ภาพที่ 3.18 หน้าจอ JMeter แสดงเวลาในการตอบสนองกลับของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
Canal	1000	3	2	68	5.57	0.00%
TOTAL	1000	3	2	68	5.57	0.00%

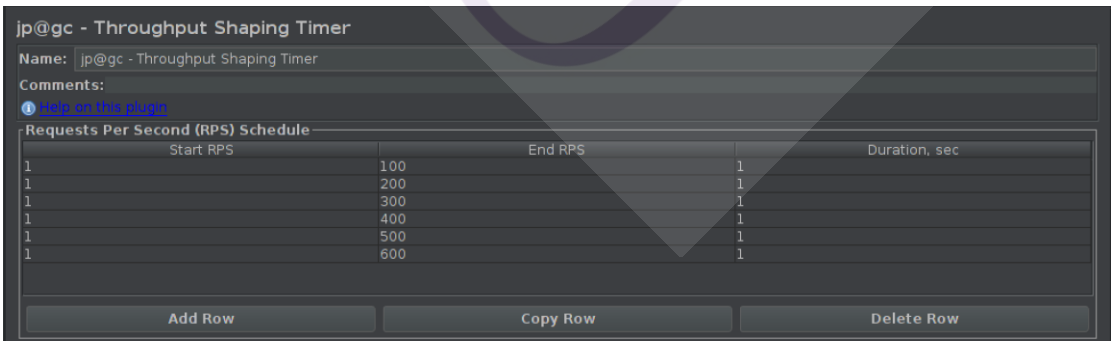
ภาพที่ 3.19 หน้าจอ JMeter แสดงเวลาในการตอบสนองกลับของคอนเทนเนอร์เน็ตเวิร์คิวธี Canal

ผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 4 Kbytes เป็นจำนวน 1,000 ครั้งภายในเวลา 100 วินาที แต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คได้ให้ผลการทดลองทางด้านเวลาในการตอบสนองดังรูปที่ 3.17, 3.18 และ 3.19

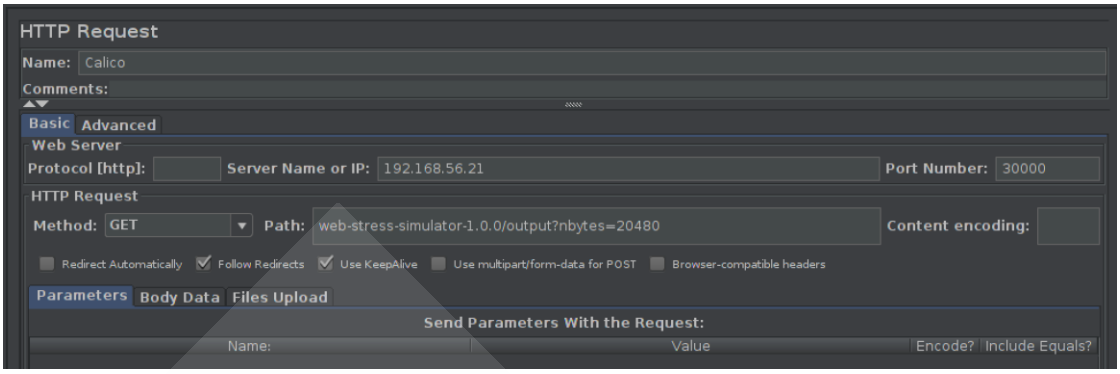
2. กำหนดให้มีการร้องขอเข้าใช้บริการ (Number of Threads (users)) จำนวนรวมแล้ว 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที ทำการร้องขอไฟล์ขนาด 20 Kbytes โดยมีลักษณะการร้องขอดังนี้

- วินาทีที่ 1: 100 การร้องขอ
- วินาทีที่ 2: 200 การร้องขอ
- วินาทีที่ 3: 300 การร้องขอ
- วินาทีที่ 4: 400 การร้องขอ
- วินาทีที่ 5: 500 การร้องขอ
- วินาทีที่ 6: 600 การร้องขอ

แสดงตัวอย่างการตั้งค่าการร้องขอที่ JMeter ในภาพที่ 3.20 โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 20 Kbytes ตัวอย่างดังภาพที่ 3.21 และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 20 Kbytes กลับมาให้จนครบ



ภาพที่ 3.20 หน้าจอ JMeter กำหนดจำนวนร้องขอเข้าใช้บริการ 2,100 ครั้ง



ภาพที่ 3.21 หน้าจอ JMeter กำหนด IP Address และพาธที่เข้าถึงเว็บไซต์

จากภาพที่ 3.21 เป็นการสร้าง HTTP Request เพื่อเป็นโปรโตคอลที่ใช้ในการร้องขอใช้บริการ โดยมีการกำหนด Server Name or IP เป็นเครื่องมาสเตอร์ตามแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คที่กำหนดไว้ ตัวอย่าง IP Address ที่แสดงเป็น IP Address เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico และการร้องขอข้อมูลขนาด 20 Kbytes จาก Web Stress Simulator ทำการกำหนดพาธเป็น web-stress-simulator-1.0.0/output?nbytes=20480

ผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 20 Kbytes เป็นจำนวน 2,100 ครั้งภายในเวลา 6 วินาที แต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คได้ให้ผลการทดลองทางด้านเวลาในการตอบสนองดังรูปที่ 3.22, 3.23 และ 3.24

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
Flannel	2100	275	3	1470	233.69	0.00%
TOTAL	2100	275	3	1470	233.69	0.00%

ภาพที่ 3.22 หน้าจอ JMeter แสดงเวลาในการตอบสนองกลับของคอนเทนเนอร์เน็ตเวิร์ควิธี Flannel

Label	# Samples	Average	Min	Max	Std. Dev.	Error %
Calico	2100	275	2	1722	230.63	0.00%
TOTAL	2100	275	2	1722	230.63	0.00%

ภาพที่ 3.23 หน้าจอ JMeter แสดงเวลาในการตอบสนองกลับของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico

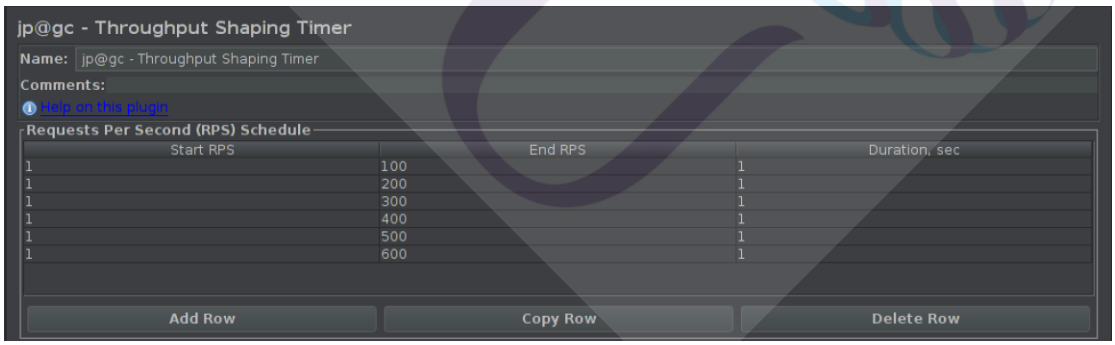
Label	# Samples	Average	Min	Max	Std. Dev.	Error %
Canal	2100	354	3	3167	332.07	0.00%
TOTAL	2100	354	3	3167	332.07	0.00%

ภาพที่ 3.24 หน้าจอ JMeter แสดงเวลาในการตอบสนองกลับของคอนเทนเนอร์เน็ตเวิร์ควิธี Canal

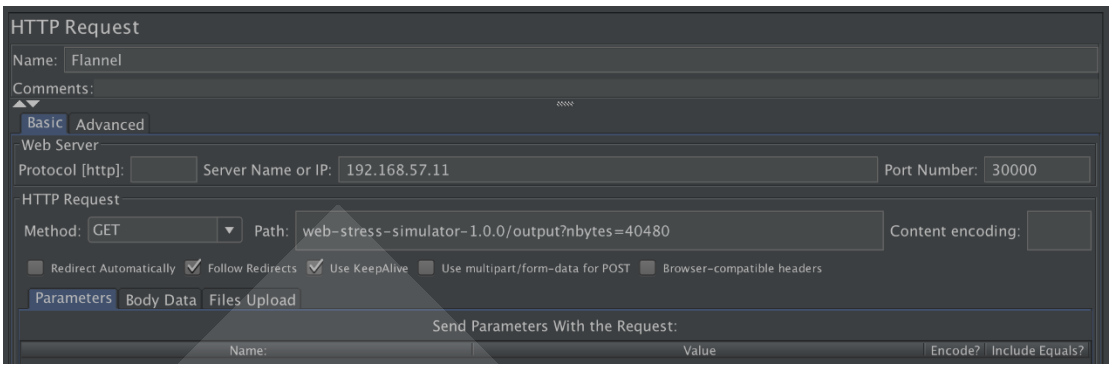
3. กำหนดให้มีการร้องขอเข้าใช้บริการ (Number of Threads (users)) จำนวนรวมแล้ว 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที ทำการร้องขอไฟล์ขนาด 40 Kbytes เพื่อวัดการทำงานของ CPU ของเครื่องมาสเตอร์และเครื่องมินเนี่ยน พร้อมทั้งวัดปริมาณการดาวน์โหลดข้อมูลของเครื่องทดสอบ และปริมาณ Throughput โดยมีลักษณะการร้องขอดังนี้

- วินาทีที่ 1: 100 การร้องขอ
- วินาทีที่ 2: 200 การร้องขอ
- วินาทีที่ 3: 300 การร้องขอ
- วินาทีที่ 4: 400 การร้องขอ
- วินาทีที่ 5: 500 การร้องขอ
- วินาทีที่ 6: 600 การร้องขอ

แสดงตัวอย่างการตั้งค่าการร้องขอที่ JMeter ในภาพที่ 3.25 โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 40 Kbytes ตัวอย่างดังภาพที่ 3.26 และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 40 Kbytes กลับมาให้จนครบ



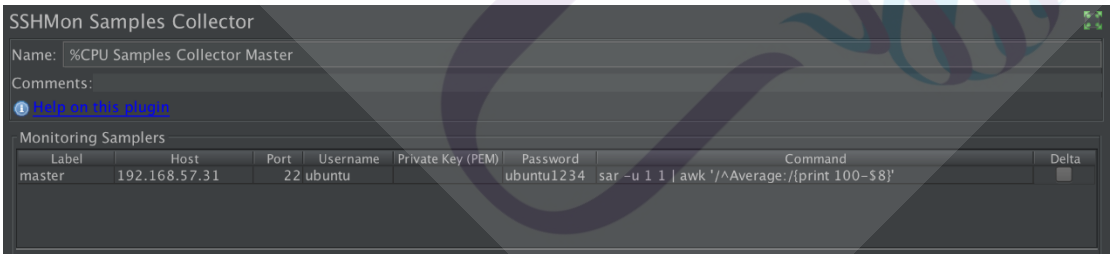
ภาพที่ 3.25 หน้าจอ JMeter กำหนดจำนวนร้องขอเข้าใช้บริการ 2,100 ครั้ง



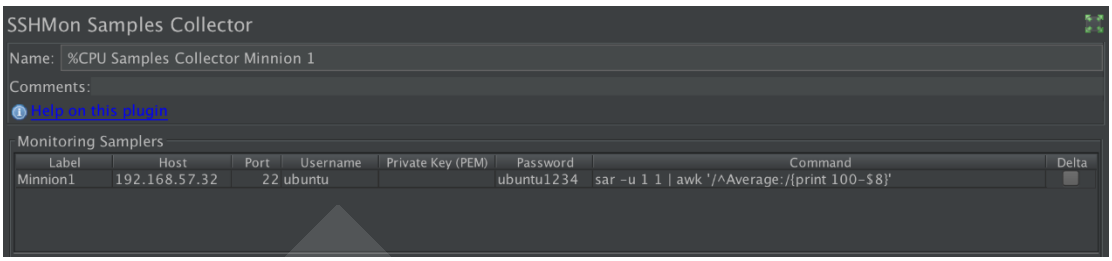
ภาพที่ 3.26 หน้าจอ JMeter กำหนด IP Address และพาธที่เข้าถึงเว็บไซต์

จากภาพที่ 3.26 เป็นการสร้าง HTTP Request เพื่อเป็นโปรโตคอลที่ใช้ในการร้องขอใช้บริการ โดยมีการกำหนด Server Name or IP เป็นเครื่องมาสเตอร์ตามแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คที่กำหนดไว้ ตัวอย่าง IP Address ที่แสดงเป็น IP Address เครื่องมาสเตอร์ของคอนเทนเนอร์เน็ตเวิร์ควิธี Flannel และการร้องขอข้อมูลขนาด 40 Kbytes จาก Web Stress Simulator ทำการกำหนดพาธเป็น web-stress-simulator-1.0.0/output?nbytes=40480

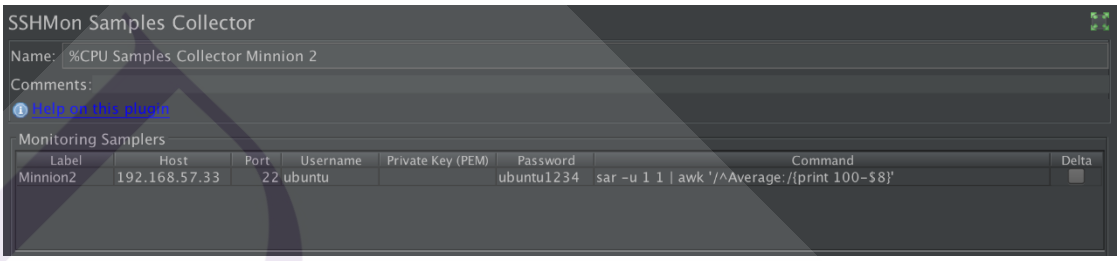
ในการกำหนด JMeter เพื่อทำการวัดการทำงานของ CPU ของเครื่องมาสเตอร์และเครื่องมินเนี่ยน แสดงตัวอย่างการตั้งค่าเพื่อวัดการใช้งาน CPU ที่ JMeter ของวิธี Flannel ดังภาพที่ 3.27-3.29



ภาพที่ 3.27 หน้าจอ JMeter กำหนด IP และคำสั่งอ่านข้อมูล CPU ของเครื่องมาสเตอร์



ภาพที่ 3.28 หน้าจอ JMeter กำหนด IP และคำสั่งอ่านข้อมูล CPU ของเครื่องมินเนี่ยน 1



ภาพที่ 3.29 หน้าจอ JMeter กำหนด IP และคำสั่งอ่านข้อมูล CPU ของเครื่องมินเนี่ยน 2

ตัวอย่างจากภาพที่ 3.27 เป็นการใช้งานในลักษณะ SSH ผ่านทาง JMeter เพื่อดึงข้อมูล CPU โดยมีการกำหนด IP Port พร้อมทั้ง Username และ Password ของเครื่องที่ต้องการ นอกจากนี้จะต้องมีการกำหนดคำสั่ง sar เพื่อดึงข้อมูล CPU ซึ่งในการทดลองนี้ใช้คำสั่ง sar –u 1 1 | awk '/^Average:/{print 100-$8}'เป็นการดึง % ปริมาณการใช้งาน CPU ณ ขณะที่ JMeter ทำงาน

ผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 40 Kbytes เป็นจำนวน 2,100 ครั้งภายในเวลา 6 วินาที แต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คได้ให้ผลการทดลองดังรูปที่ 3.30-3.32

timeStamp	elapsed	label	responseCode	responseMessage
1.52747E+12	0	master		12.73
1.52747E+12	0	master		16.19
1.52747E+12	0	master		20.54
1.52747E+12	0	master		28.82

ภาพที่ 3.30 ตัวอย่างไฟล์เปอร์เซ็นต์การใช้งาน CPU ของเครื่องมาสเตอร์ที่ JMeter เขียนบันทึก

timeStamp	elapsed	label	responseCode	responseMessage
1.52747E+12	0	Minnion1		64.95
1.52747E+12	0	Minnion1		35.9
1.52747E+12	0	Minnion1		39.9
1.52747E+12	0	Minnion1		46.32

ภาพที่ 3.31 ตัวอย่างไฟล์เปอร์เซ็นต์การใช้งาน CPU ของเครื่องมินเนี่ยน 1 ที่ JMeter เขียนบันทึก

timeStamp	elapsed	label	responseCode	responseMessage
1.52747E+12	0	Minnion2		73.82
1.52747E+12	0	Minnion2		35.2
1.52747E+12	0	Minnion2		37.95
1.52747E+12	0	Minnion2		46.7

ภาพที่ 3.32 ตัวอย่างไฟล์เปอร์เซ็นต์การใช้งาน CPU ของเครื่องมินเนี่ยน 2 ที่ JMeter เขียนบันทึก

สำหรับการเปรียบเทียบประสิทธิภาพระหว่างคอนเทนเนอร์เน็ตเวิร์คแต่ละวิธี รวมถึงการวิเคราะห์จะนำเสนอและแสดงรายละเอียดไว้ในบทถัดไป

บทที่ 4
การทดลองและผลการวิจัย

ในบทนี้จะกล่าวถึงผลการศึกษาวิจัยและการอภิปรายผลการวิจัย เพื่อให้ทราบถึงประสิทธิภาพสมรรถนะของรูปแบบการเชื่อมโยงเครือข่ายระหว่างคอนเทนเนอร์ 3 วิธี โดยมีหัวข้อในการวิจัย ดังนี้

1. การเปรียบเทียบเวลาในการตอบสนองกลับมายังผู้ใช้บริการ โดยที่ผู้ใช้บริการได้รับข้อมูลในการร้องขอครบถ้วน
2. การเปรียบเทียบการทำงานของ CPU ของเครื่องมาสเตอร์ และเครื่องมินเนี่ยนระหว่างการให้บริการ
3. การเปรียบเทียบปริมาณการดาวน์โหลดข้อมูล
4. การเปรียบเทียบปริมาณ Throughput

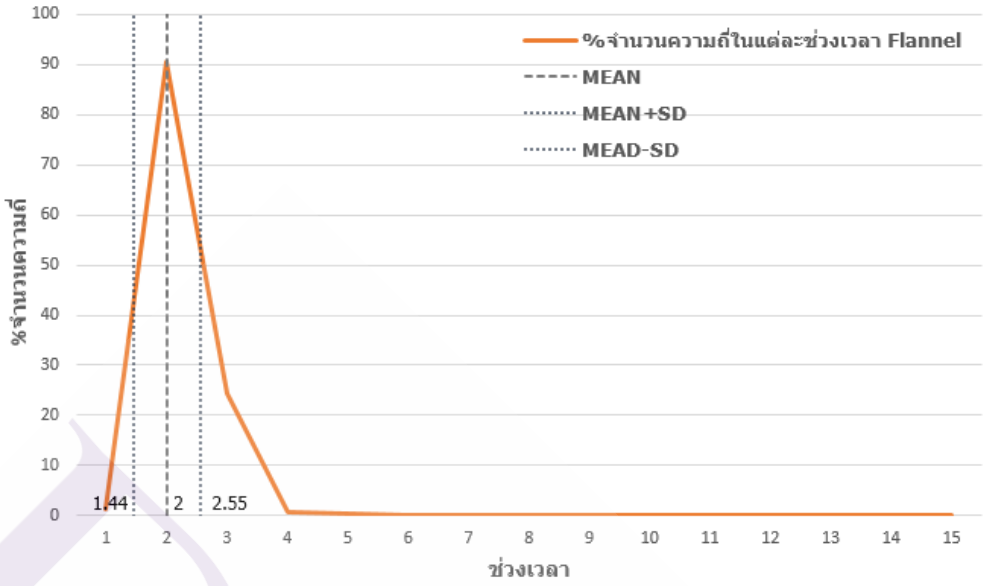
4.1 การประเมินประสิทธิภาพทางด้านเวลาในการตอบสนองกลับมายังผู้ใช้บริการ โดยที่ผู้ใช้บริการได้รับข้อมูลในการร้องขอครบถ้วน

4.1.1 รายละเอียดการทดลองร้องขอไฟล์ขนาด 4 Kbytes

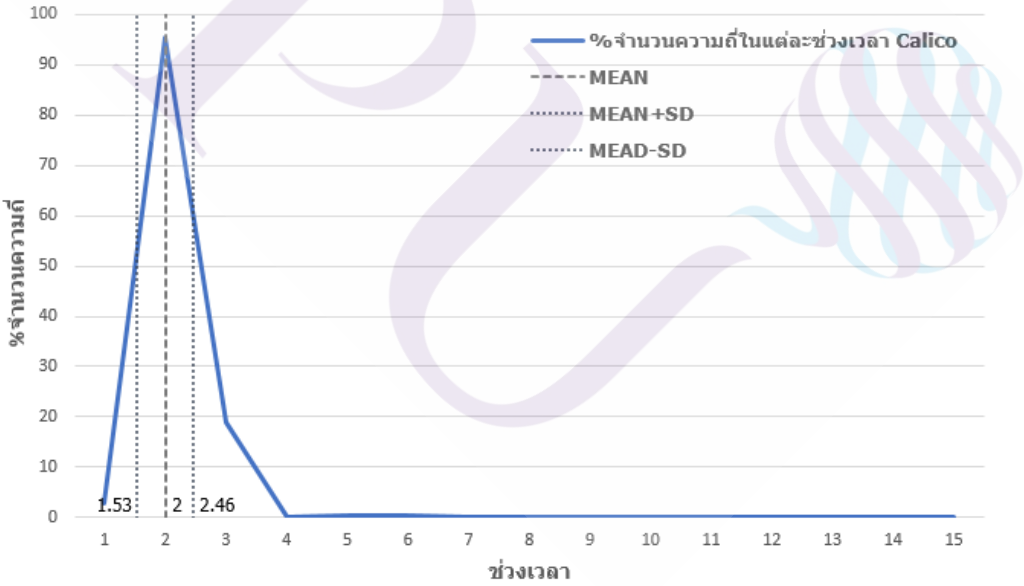
การทดลองกำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 1,000 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 100 วินาที โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 4 Kbytes และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 4 Kbytes กลับมาให้จนครบ โดยเป็นการเปรียบเทียบเวลาที่ใช้ในการตอบสนองระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal โดยผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 4 Kbytes แสดงดังตารางที่ 4.1 และภาพที่ 4.1-4.3

ตารางที่ 4.1 เวลาที่ใช้ในการตอบสนองกรณีร้องขอจำนวน 1,000 ครั้ง ใน 100 วินาที

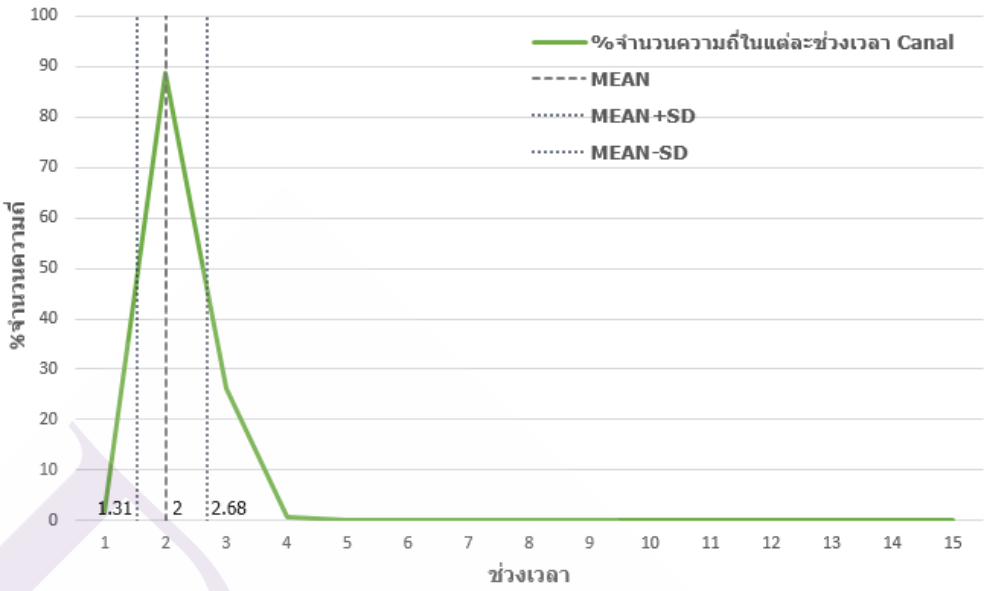
	คอนเทนเนอร์เน็ตเวิร์ค		
	Flannel	Calico	Canal
เวลา (ms)	2.232	2.151	2.251



ภาพที่ 4.1 กราฟแสดงการแจกแจงความถี่ในแต่ละช่วงเวลาวิธี Flannel กรณีร้องขอจำนวน1,000 ครั้ง ใน 100 วินาที



ภาพที่ 4.2 กราฟแสดงการแจกแจงความถี่ในแต่ละช่วงเวลาวิธี Calico กรณีร้องขอจำนวน1,000 ครั้ง ใน 100 วินาที



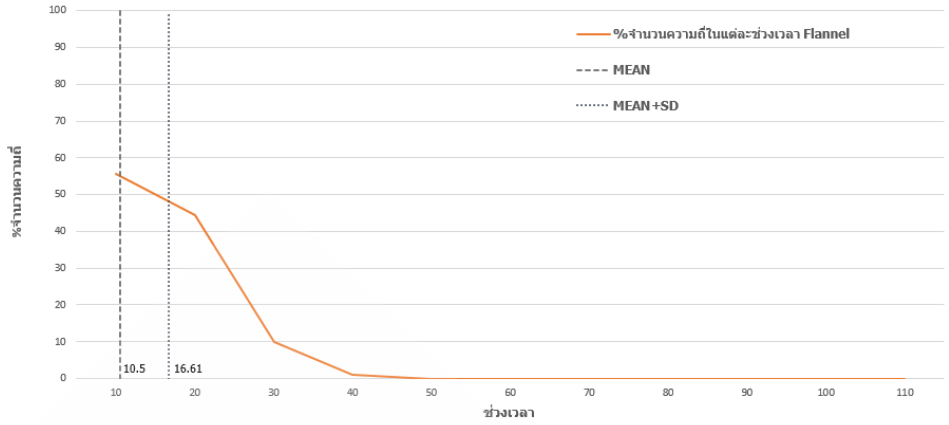
ภาพที่ 4.3 กราฟแสดงการแจกแจงความถี่ในแต่ละช่วงเวลาวิธี Canal กรณีร้องขอจำนวน1,000 ครั้ง ใน 100 วินาที

4.1.2 รายละเอียดการทดลองร้องขอไฟล์ขนาด 20 Kbytes

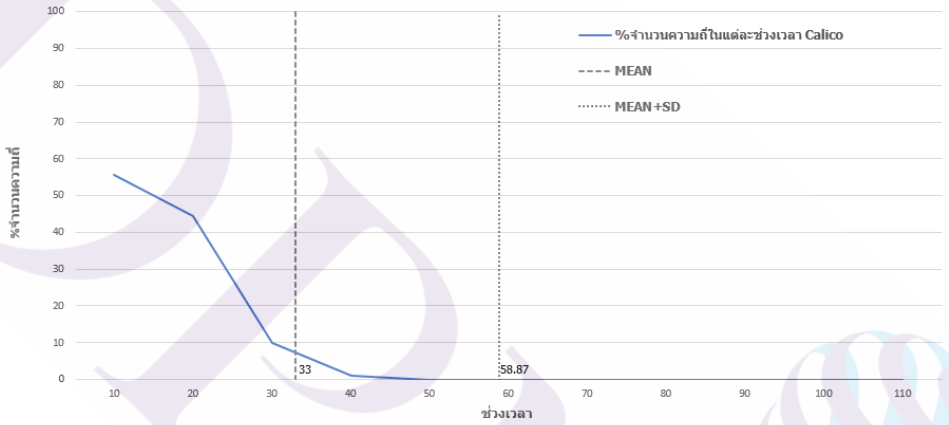
การทดลองกำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 20 Kbytes และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 20 Kbytes กลับมาให้จนครบ โดยเป็นการเปรียบเทียบเวลาที่ใช้ในการตอบสนองระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal โดยผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 20 Kbyte แสดงดังตารางที่ 4.2 และภาพที่ 4.4 – 4.9

ตารางที่ 4.2 เวลาที่ใช้ในการตอบสนองกรณีร้องขอจำนวน 2,100 ครั้ง ใน 6 วินาที

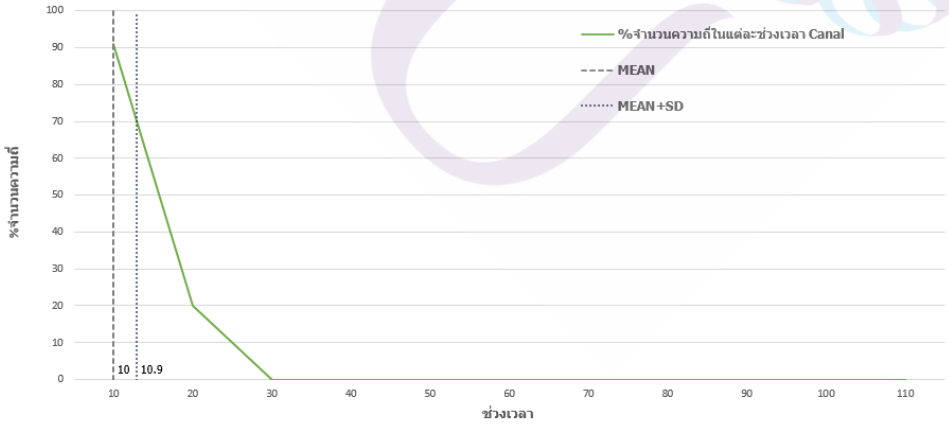
	คอนเทนเนอร์เน็ตเวิร์ค		
	Flannel	Calico	Canal
เวลาเฉลี่ยรวม	382.56	365.18	380.10



(a)



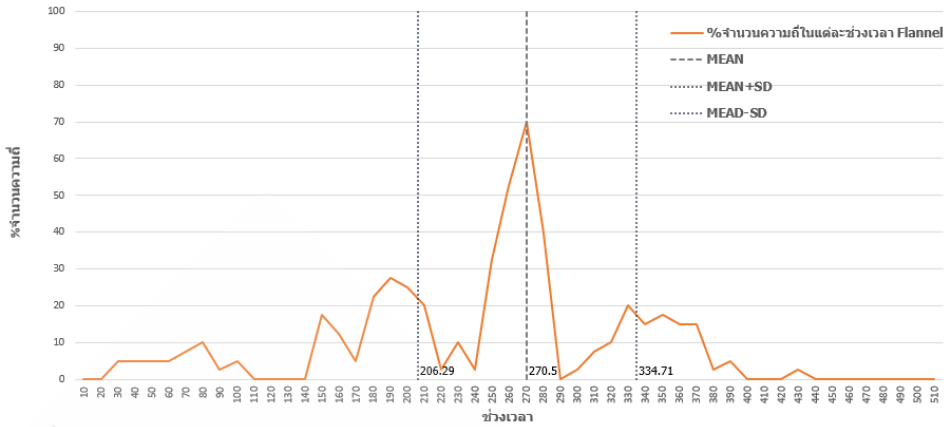
(b)



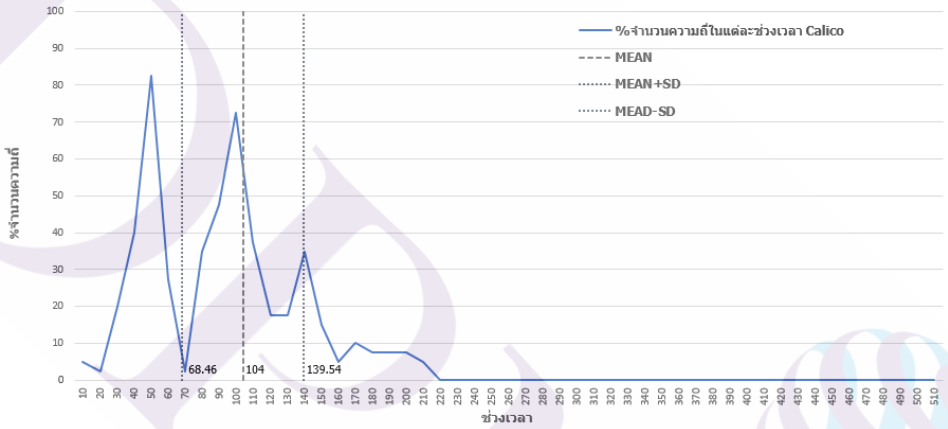
(c)

ภาพที่ 4.4 กราฟแสดงการแจกแจงความถี่ในวินาทีที่ 1 ในแต่ละวิธี

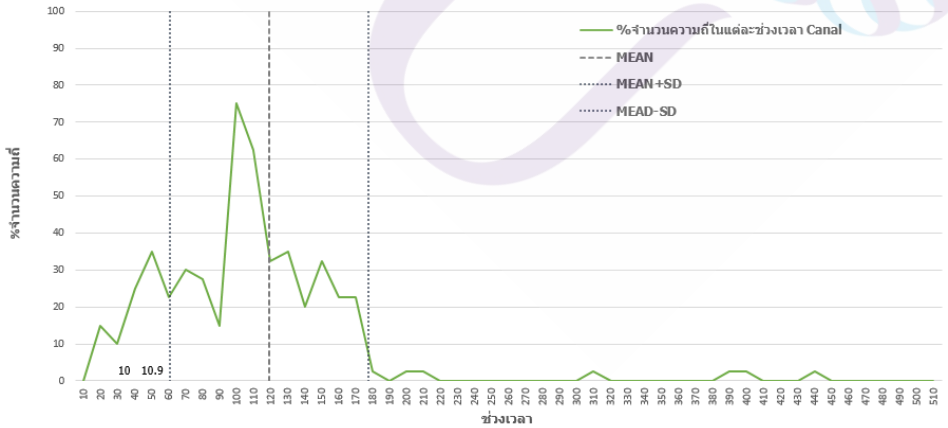
(a) Flannel (b) Calico (c) Canal



(a)



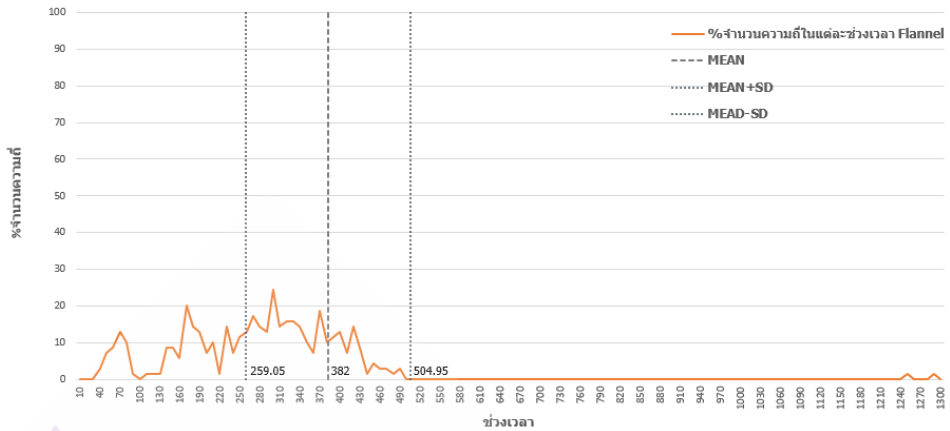
(b)



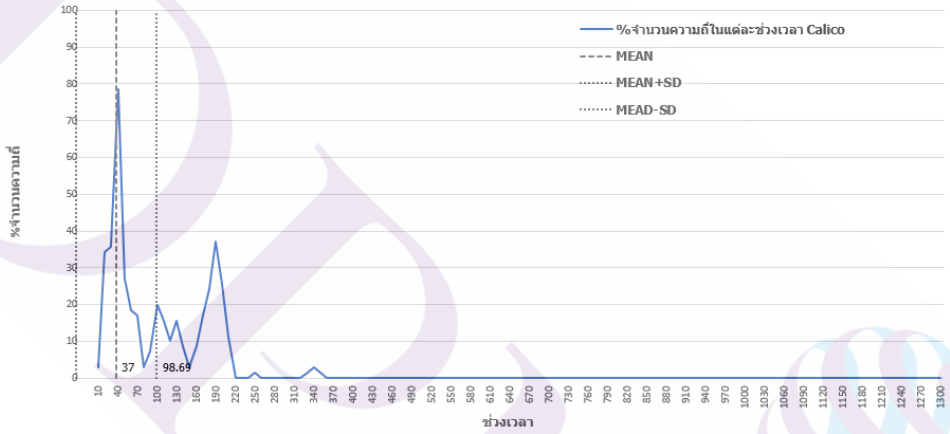
(c)

ภาพที่ 4.5 กราฟแสดงการแจกแจงความถี่ในวินาทีที่ 2 ในแต่ละวิธี

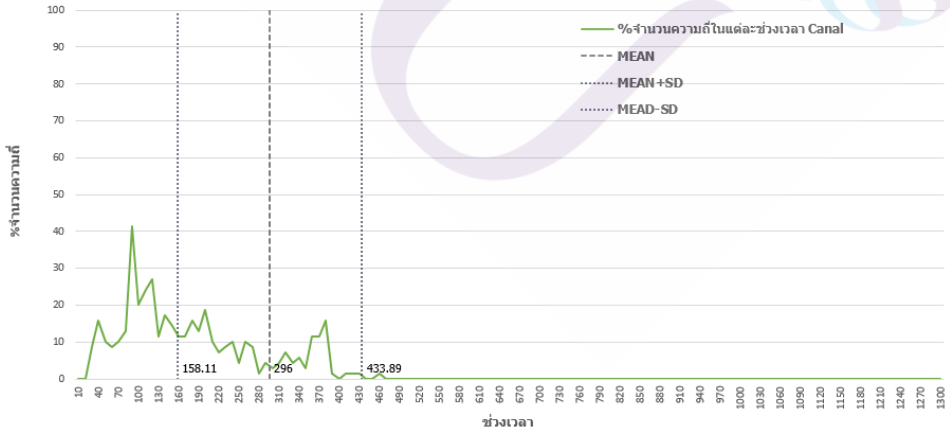
(a) Flannel (b) Calico (c) Canal



(a)



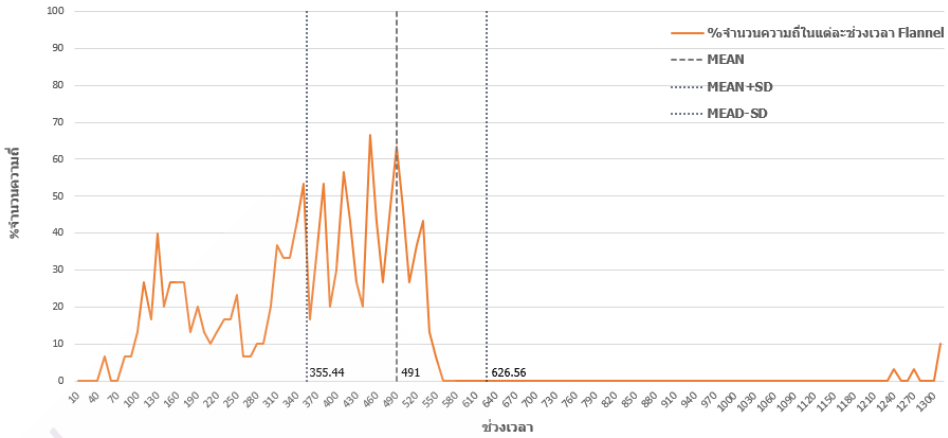
(b)



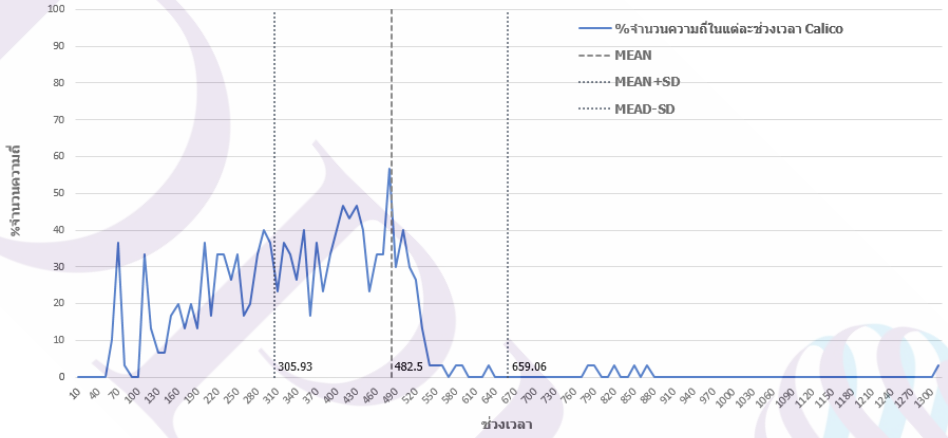
(c)

ภาพที่ 4.6 กราฟแสดงการแจกแจงความถี่ในวินาทีที่ 3 ในแต่ละวิธี

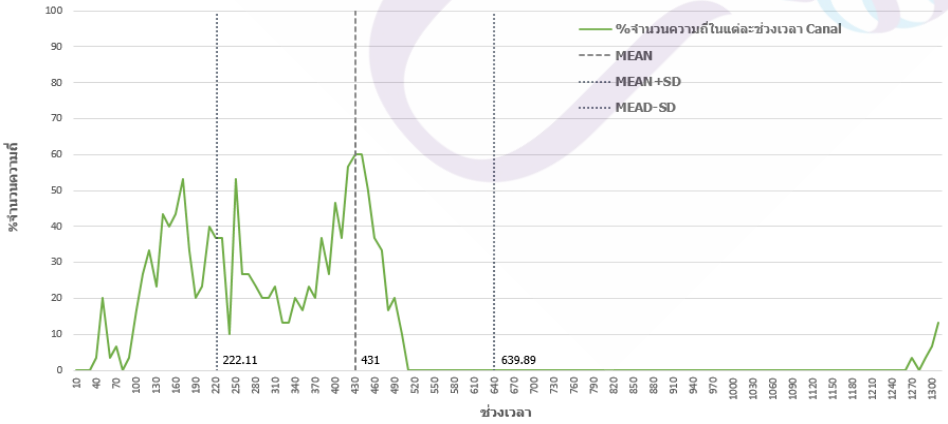
(a) Flannel (b) Calico (c) Canal



(a)



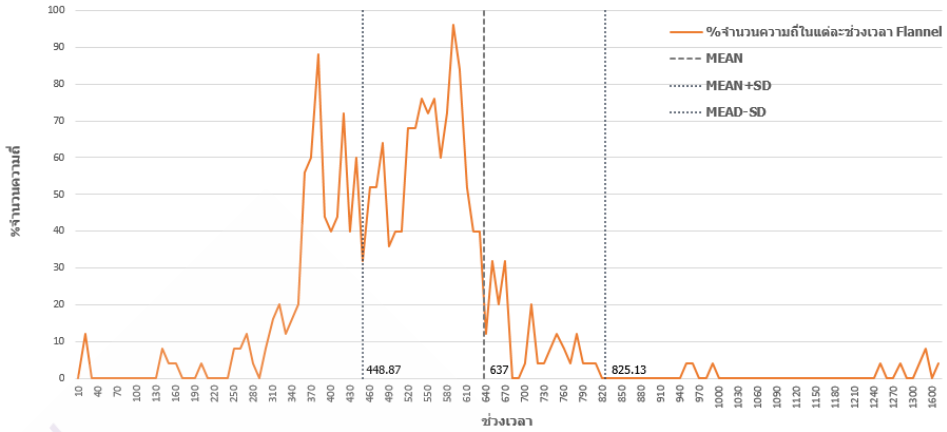
(b)



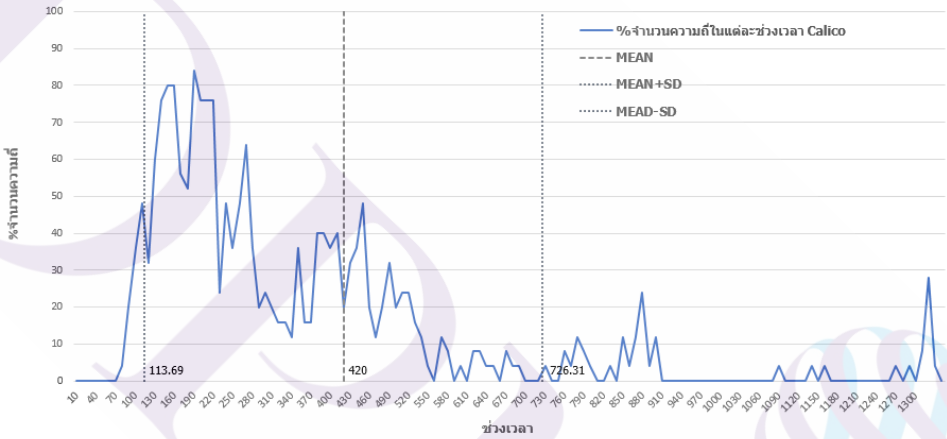
(c)

ภาพที่ 4.7 กราฟแสดงการแจกแจงความถี่ในวินาทีที่ 4 ในแต่ละวิธี

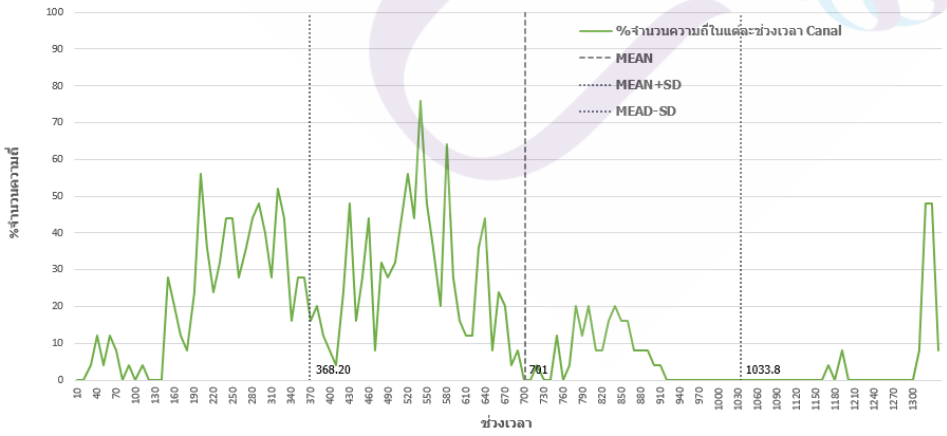
(a) Flannel (b) Calico (c) Canal



(a)



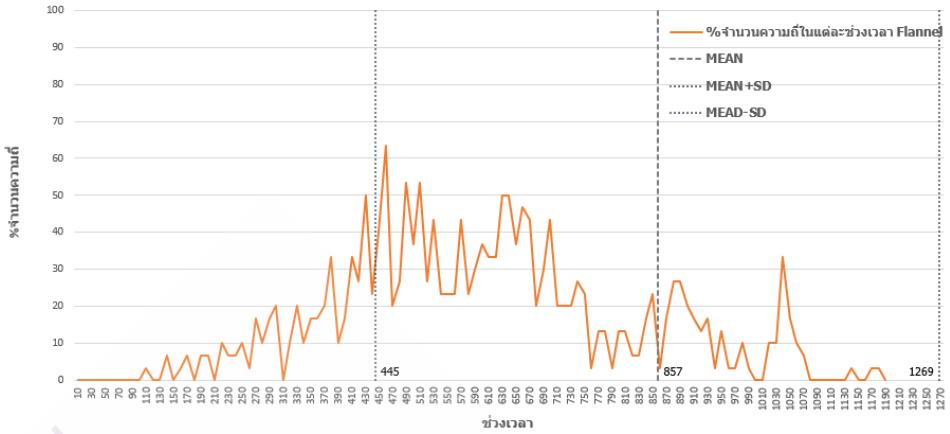
(b)



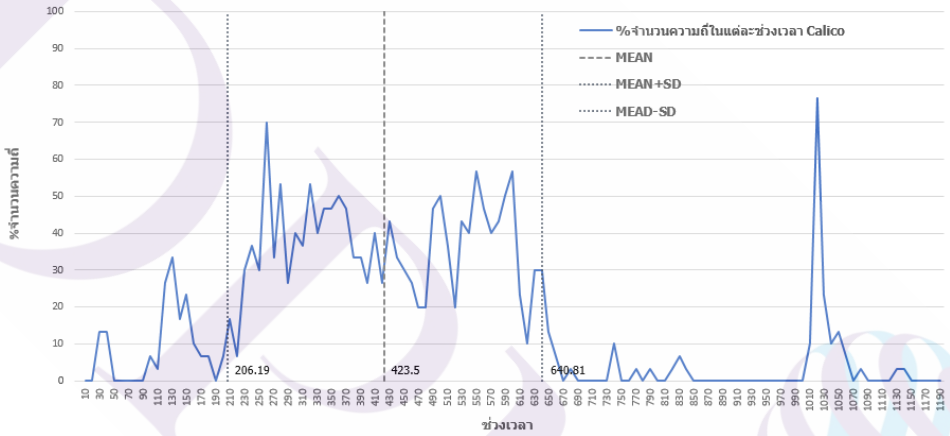
(c)

ภาพที่ 4.8 กราฟแสดงการแจกแจงความถี่ในวินาทีที่ 5 ในแต่ละวิธี

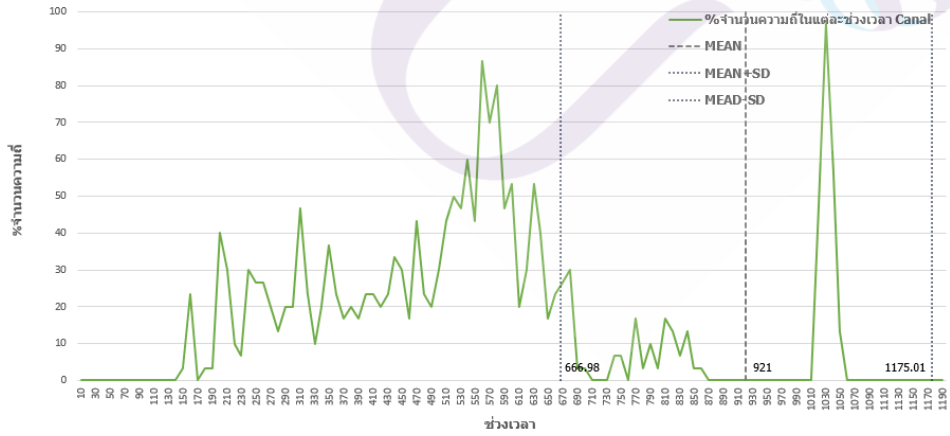
(a) Flannel (b) Calico (c) Canal



(a)



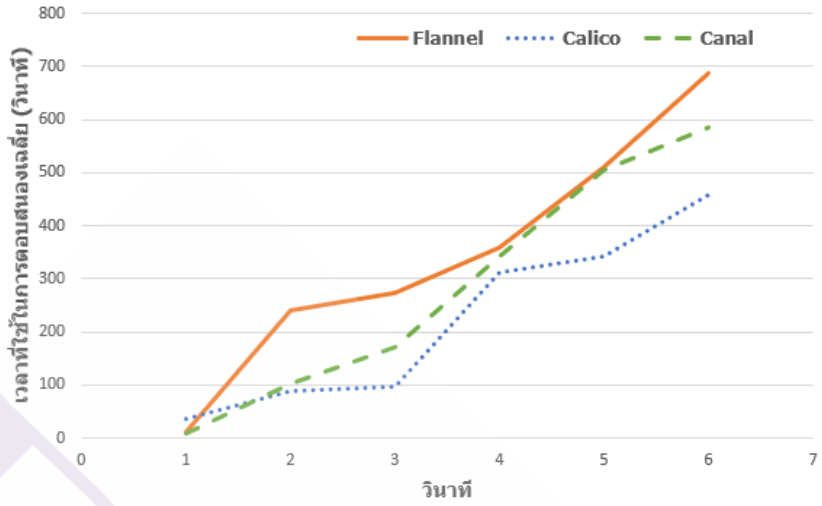
(b)



(c)

ภาพที่ 4.9 กราฟแสดงการแจกแจงความถี่ในวินาทีที่ 6 ในแต่ละวิธี

(a) Flannel (b) Calico (c) Canal



ภาพที่ 4.10 กราฟแสดงเปรียบเทียบเวลาที่ใช้ในการตอบสนองในแต่ละวิธี

4.1.3 วิเคราะห์ผลการทดลอง

จากตารางที่ 4.1 และ 4.2 จะเห็นได้ว่าคอนเทนเนอร์เน็ตเวิร์ควิธี Calico ให้เวลาในการตอบสนองน้อยว่าอีก 2 วิธี และวิธี Flannel และ Canal ให้ผลการทดลองไม่แตกต่างกัน และจากภาพที่ 4.1 -4.9 ที่แสดงการแจกแจงความถี่ของเวลาที่ใช้ในการตอบสนองของแต่ละวิธีคอนเทนเนอร์เน็ตเวิร์คมาแจกแจง พบว่าวิธี Calico มีความกว้างของกราฟน้อยกว่า และมีการกระจายตัวของข้อมูลแคบกว่า การแจกแจงความถี่ดีกว่าอีก 2 วิธี

และจากภาพที่ 4.10 เมื่อพิจารณาเวลาที่ใช้ในการตอนสนองในแต่ละวิธี จะพบว่าเวลาในการตอบสนองของวิธี Calico ในช่วงวินาทีแรก จะใช้เวลาในการตอบสนองมากกว่าอีก 2 วิธี แต่เมื่อเวลาผ่านไปและมีการร้องขอที่สูงขึ้นอย่างต่อเนื่อง จะเห็นได้ว่าวิธี Calico ใช้เวลาในการตอบสนองน้อยลง ซึ่งจะเห็นได้ว่าประสิทธิภาพทางด้านเวลาในการตอบสนองของคอนเทนเนอร์เน็ตเวิร์ควิธี Calico ให้ผลลัพท์ดีกว่า Flannel และ Canal เนื่องจากการทำงานของ Flannel และ Canal มีการทำงานในรูปแบบ VxLAN Encapsulation ทำให้ต้องใช้เวลาในการ Encapsulation เพิ่มส่วนของ Header เข้าไปในแพ็กเกจ ซึ่งส่งผลให้ใช้เวลามากกว่า

4.2 การเปรียบเทียบการทำงานของ CPU ของเครื่องมาสเตอร์ และเครื่องมินเนี่ยนระหว่างการให้บริการ

4.2.1 รายละเอียดการทดลอง

การทดลองกำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 40 Kbytes และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 40 Kbytes กลับมาให้จนครบ โดยเป็นการเปรียบเทียบการทำงานของ CPU ของเครื่องมาสเตอร์ และเครื่องมินเนี่ยนระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal โดยผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 40 Kbytes แสดงดังตารางที่ 4.3

ตารางที่ 4.3 การทำงานของซีพียูของเครื่องมาสเตอร์และเครื่องมินเนี่ยน

	% การทำงานของ CPU		
	Master Server	Minion Server 1	Minion Server 2
Flannel	32.93	44.16	46.43
Calico	30.89	42.15	42.19
Canal	33.44	45.06	42.69

4.2.2 วิเคราะห์ผลการทดลอง

จากตารางที่ 4.3 เมื่อพิจารณาการทำงานของ CPU ของเครื่องมาสเตอร์ และเครื่องมินเนี่ยน พบว่าวิธี Calico มีการทำงานของ CPU น้อยกว่า Flannel และ Canal 1% เนื่องจากการทำงานของ Flannel และ Canal มีการใช้กระบวนการ VxLAN Encapsulation ทำให้มีปริมาณข้อมูลในส่วนของเฮดเดอร์ (Header) ที่ใช้ส่งข้อมูลระหว่างกัน ทำให้ต้องมีการใช้งาน CPU มากขึ้นเพื่อทำการ Encapsulation

4.3 การเปรียบเทียบปริมาณการดาวน์โหลดข้อมูล

4.3.1 รายละเอียดการทดลองร้องขอไฟล์ขนาด 40 Kbytes

การทดลองกำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 40 Kbytes และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 40 Kbytes กลับมาให้จนครบ โดยเป็นการเปรียบเทียบปริมาณการดาวน์โหลดข้อมูลบนเครื่องผู้ใช้บริการ ระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal โดยผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 40 Kbytes แสดงดังตารางที่ 4.4

ตารางที่ 4.4 การเปรียบเทียบปริมาณการดาวน์โหลดข้อมูล ขนาดไฟล์ 40 Kbyte

	Flannel	Calico	Canal
Download (kB/Sec)	13,879.23	13,925.78	13,851.06

จากตารางที่ 4.4 แสดงให้เห็นปริมาณการดาวน์โหลดข้อมูลพบว่าปริมาณการดาวน์โหลดข้อมูลของ Calico สูงกว่า Flannel และ Canal 1%

4.3.2 รายละเอียดการทดลองร้องขอไฟล์ขนาด 100 Kbytes

การทดลองกำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 100 Kbytes และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 100 Kbytes กลับมาให้จนครบ โดยเป็นการเปรียบเทียบปริมาณการดาวน์โหลดข้อมูลบนเครื่องผู้ใช้บริการ ระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal โดยผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 100 Kbytes แสดงดังตารางที่ 4.5

ตารางที่ 4.5 การเปรียบเทียบปริมาณการดาวน์โหลดข้อมูล ขนาดไฟล์ 100 Kbytes

	Flannel	Calico	Canal
Download (kB/Sec)	16,631.74	16,833.91	16,549.17

จากตารางที่ 4.5 แสดงให้เห็นปริมาณการดาวน์โหลดข้อมูลพบว่าปริมาณการดาวน์โหลดข้อมูลของ Calico สูงกว่า Flannel และ Canal ประมาณ 1%

4.3.3 วิเคราะห์ผลการทดลอง

จากตารางที่ 4.4-4.5 เมื่อพิจารณาการเปรียบเทียบปริมาณการดาวน์โหลดข้อมูลพบว่าทั้งวิธีของ Calico ให้ผลลัพท์ดีกว่า Flannel และ Canal

4.4 การเปรียบเทียบปริมาณ Throughput

4.4.1 รายละเอียดการทดลอง

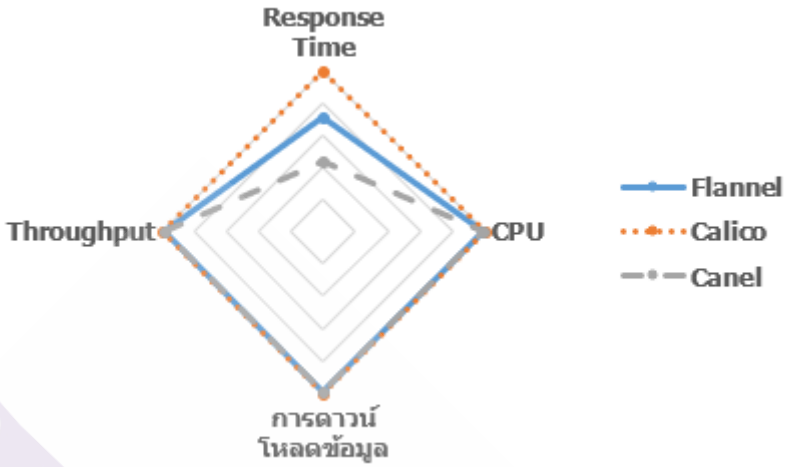
การทดลองกำหนดให้มีการร้องขอเข้าใช้บริการจำนวน (Number of Threads (users)) 2,100 ครั้ง ภายในเวลา (Ramp-up Period (in seconds)) 6 วินาที โดยแต่ละการร้องขอไปยัง Web Stress Simulator ร้องขอไฟล์ขนาด 40 Kbytes และระบบจะตอบสนองด้วยการส่ง HTTP Response ขนาด 40 Kbytes กลับมาให้จนครบ โดยเป็นการเปรียบเทียบปริมาณ Throughput ระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal โดยผลที่ได้จากการจำลองการร้องขอข้อมูลขนาด 40 Kbytes แสดงดังตารางที่ 4.6

ตารางที่ 4.6 การเปรียบเทียบปริมาณ Throughput

	Flannel	Calico	Canal
Throughput (Transaction /Sec)	345.6	349.1	341.9

จากตารางที่ 4.5 แสดงให้เห็นปริมาณการ Throughput ว่าปริมาณการ Throughput ของ Calico มีค่าสูงกว่า Flannel และ Canal ประมาณ 1%

จากผลการทดลองที่กล่าวมาสามารถเปรียบเทียบประสิทธิภาพระหว่างคอนเทนเนอร์เน็ตเวิร์ค Flannel Calico และ Canal ได้ดังภาพที่ 4.11 ซึ่งแสดงให้เห็นว่าวิธีของ Calico ให้ผลลัพท์ที่ดีทั้งทางด้านเวลาที่ใช้ในการตอบสนอง ปริมาณการดาวน์โหลดข้อมูล ปริมาณ Throughput และทางด้านการทำงานของซีพียู



ภาพที่ 4.11 กราฟแสดงจุดอ่อน-จุดแข็งของแต่ละวิธี

บทที่ 5
สรุปผล และข้อเสนอแนะ

5.1 สรุปผลการวิจัย

ในงานวิจัยนี้ทำการประเมินประสิทธิภาพของประสิทธิภาพการทำงานของวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธีด้วยกัน คือ Flannel, Calico และ Canal พบว่าวิธีการเชื่อมโยงแบบ Calico มีประสิทธิภาพการทำงานดีกว่าอีก 2 วิธีทางด้านเวลาในการตอบสนองกลับมายังผู้ใช้บริการ ปริมาณการดาวน์โหลดข้อมูล ปริมาณ Throughput ยังเว้นการทำงานของ CPU ที่เครื่องมาสเตอร์และเครื่องมินเนี่ยนที่วิธี Calico มีการทำงานที่มากกว่าอีก 2 วิธีเพียง 1% เนื่องจากลักษณะของ Calico เป็นการจำลองเน็ตเวิร์คในระดับเลเยอร์ 3 และไม่มีการทำ Encapsulation หรือ Tunnel ทำให้มี Overhead น้อยกว่าอีก 2 วิธี ส่งผลให้มีประสิทธิภาพการทำงานที่ดีกว่า และยังเป็นวิธีที่เหมาะกับกรณีที่ผู้ใช้งานมีเป็นจำนวนมาก สำหรับวิธี Flannel และ Canal ให้ผลทางด้านประสิทธิภาพคล้ายกัน เนื่องจากทั้ง 2 วิธีอยู่บนพื้นฐานของ VxLAN เหมือนกัน

สรุปผลตามวัตถุประสงค์ของงานวิจัย สามารถสรุปผลได้ดังนี้

1. สามารถเข้าใจถึงวิธีการทำงานของเทคโนโลยีทางด้านด็อกเกอร์คอนเทนเนอร์
2. สามารถเข้าใจถึงวิธีการใช้งานคูเบอร์เนตส์ และวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธี คือ Flannel Calico และ Canal
3. ทำการประเมินและวิเคราะห์ประสิทธิภาพวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธี คือ Flannel Calico และ Canal

สรุปผลตามขอบเขตของงานวิจัย สามารถสรุปผลได้ดังนี้

1. สามารถเข้าใจถึงวิธีการทำงานของเทคโนโลยีทางด้านด็อกเกอร์คอนเทนเนอร์ และสามารถสร้างระบบการประมวลผลคลาวด์ด้วยคูเบอร์เนต ด้วยวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธี คือ Flannel Calico และ Canal
2. สร้างระบบประมวลผลคลาวด์ด้วยคูเบอร์เนต บนเครื่องจักรเสมือน 3 เครื่อง ประกอบไปด้วยเครื่องมาสเตอร์ 1 เครื่อง และเครื่องมินเนี่ยน 2 เครื่อง และตั้งค่าวิธีการเชื่อมโยงคอนเทนเนอร์เน็ตเวิร์ค 3 วิธี คือ Flannel Calico และ Canal
3. ประเมินและวิเคราะห์ประสิทธิภาพวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์ค 3 วิธี คือ Flannel Calico และ Canal

5.2 ข้อจำกัดและแนวทางแก้ไขของงานวิจัย

เนื่องจากการจำลองทดสอบประสิทธิภาพวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์คด้วยการติดตั้งระบบการประมวลผลคลาว์ด้วยคูเบอร์เนตส์บนเครื่องจักรเสมือนในเครื่องคอมพิวเตอร์ของผู้วิจัยซึ่งมีหน่วยความจำและหน่วยประมวลผลจำกัด ทำให้การใช้งานระบบประมวลผลคลาวด์อาจจะไม่สามารถทำงานได้อย่างเต็มที่ ทางแก้อย่างหนึ่งคือติดตั้งระบบการประมวลผลคลาวด์บนเซิร์ฟเวอร์จริง หรือเครื่องที่มีทรัพยากรสูงเพื่อให้สามารถทำงานได้อย่างมีประสิทธิภาพ และทำให้สามารถวัดประสิทธิภาพของวิธีการเชื่อมโยงของคอนเทนเนอร์เน็ตเวิร์คแต่ละประเภทได้อย่างสมบูรณ์มากยิ่งขึ้น

5.3 ข้อเสนอแนะ

ในงานวิจัยชิ้นนี้เป็นการทดลองบนเครื่องคอมพิวเตอร์ของผู้วิจัยซึ่งมีข้อจำกัด ทางผู้วิจัยจึงขอเสนอแนะแนวทางในการวิจัยต่อยอดสำหรับผู้ที่สนใจนำไปพัฒนาต่อดังนี้

1. ทดสอบในสภาพแวดล้อมอื่น เช่น การใช้งานบนเครื่องเซิร์ฟเวอร์จริง การทดสอบผ่านทางอุปกรณ์อื่น หรือการทดสอบบนอุปกรณ์ IoT

2. ทดสอบทางด้านปัยจัยอื่น เพื่อสามารถวิเคราห์ประสิทธิภาพได้ถูกต้องมากขึ้น เช่น ด้านทางความปลอดภัย ความยืดหยุ่นในการใช้งาน

บรรณานุกรม

บรรณานุกรม

ภาษาต่างประเทศ

Apache JMeter (2018). *Apache JMeter.* Retrieved March 20, 2018 from https://jmeter.apache.org/

Bin Xie. (2016). Docker Based Overlay Network Performance Evaluation in Large Scale Streaming System. *Proceedings of the 2016 Advanced Information Management, Communicate (IMCEC)* (pp. 366-369). Xi'an, China.

C. Boettiger. (2015). An introduction to docker for reproducible research. *Communications of the ACM SIGOPS Operating Systems*, (49), 71-79, Retrieved March 15, 2018, from ACM Digital Library.

Flannel. (2018). *Flannel is a network fabric for containers.* Retrieved March 20, 2018 from https://github.com/coreos/flannel

Hao Zeng. (2017). Measurement and Evaluation for Docker Container Networking. *Proceedings of the 2017 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)* (pp. 105-108). Nanjing, China.

Jakob Struye. (2017). Assessing the Value of Containers for NFVs. *Proceedings of the International Conference on Network and Service Management (CNSM)* (pp. 1-7). Tokyo, Japan.

Kubernetes. (2018). *Deploy on Kubernetes.* Retrieved February 5, 2018 from https://docs.docker.com/docker-for-windows/kubernetes/

Rancher Labs. (2017). *Comparing Orchestration Engines in Rancher, A Closer look at Docker Swarm and Kubernetes.* Retrieved March 3, 2018, from https://cdn2.hubspot.net/hubfs/468859/eBooks/Comparing%20Kubernetes%20and%20Docker%20%20Rancher%20Labs.pdf?t=1502218959497

Tigera. (2018). *Project Calico v3.0*. Retrieved March 20, 2018 from https://www.projectcalico.org/wpcontent/uploads/2018/01/ProjectCalico.v3.datasheet

What is Kubernetes. (2018). *Kubernetes. What is Kubernetes.* Retrieved February 5, 2018 from http://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

ภาคผนวก


nccit 2018
The 14th National Conference on Computing and Information Technology
Proceedings of NCCIT 2018
The 14th National Conference on Computing and Information Technology
5th- 6th July 2018
at Shangri-La Hotel, Chiang Mai, Thailand
www.nccit.net
Faculty of Information Technology
King Mongkut's University of Technology North Bangkok
บทความวิจัย
การประชุมทางวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 14
5-6 กรกฎาคม 2561
โรงแรม แชงกรี-ลา เชียงใหม่
KMUTNB
คณะเทคโนโลยีสารสนเทศ
มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

The Fourteenth National Conference on Computing and Information Technology NCCIT2018

การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์
Performance evaluation of K8s container networking

พรรักษา กมลเวชช์ (Pornraksa Kamolvage)[1] และชัยพร เขมะภาตะพันธ์ (Chaiyaporn Khemapatapan)[2]
สาขาวิชาวิศวกรรมศาสตร์คอมพิวเตอร์และโทรคมนาคม
วิทยาลัยนวัตกรรมด้านเทคโนโลยีและวิศวกรรม มหาวิทยาลัยธุรกิจบัณฑิตย์
[1]npicmy@outlook.com, [2]chaiyaporn@dpu.ac.th

บทคัดย่อ

คูเบอร์เนตส์หรือ K8s เป็นระบบจัดการประมวลผลแบบเสมือนอีกแบบที่พัฒนาโดยกูเกิ้ลกำลังได้รับความนิยมเป็นอย่างมากสำหรับการประมวลผลคลาวด์ โดยมีพื้นฐานมาจากการใช้งานและจัดการคอนเทนเนอร์ของดอคเกอร์บนโหนดประมวลผลหลายโหนดที่เชื่อมต่อเป็นคลัสเตอร์ อย่างไรก็ตามการเชื่อมต่อเครือข่ายบนคลัสเตอร์ของคอนเทนเนอร์ทำได้หลายวิธี ดังนั้นงานวิจัยนี้จึงทำการเปรียบเทียบประสิทธิภาพทางด้านเวลาในการตอบสนองที่ใช้ในการตอบกลับและอัตราการส่งผ่านข้อมูลของการให้บริการข้อมูลด้วยเว็บเซิร์ฟเวอร์ของการเชื่อมต่อเครือข่ายคอนเทนเนอร์ 3 แบบได้แก่ Flannel, Calico และ Canal จากผลการทดลองพบว่า Calico มีประสิทธิภาพดีที่สุดจากทั้ง 3 วิธี โดยเฉพาะอย่างยิ่งกรณีที่มีปริมาณทราฟฟิกเป็นจำนวนมาก สำหรับวิธี Flannel และ Canal ให้ประสิทธิภาพใกล้เคียงกัน เนื่องจากทั้ง 2 วิธีอยู่บนพื้นฐานของ VxLAN นั่นเอง อย่างไรก็ตาม Flannel มีการนำมาใช้งานที่ง่ายที่สุด

คำสำคัญ: คอนเทนเนอร์ การเชื่อมต่อเครือข่าย คูเบอร์เนตส์ ฟาแนล คาลิโค แคแนล

Abstract

Kubernetes, K8s, is a new virtualized computing orchestration developed by Google and being popular cloud computing. K8s is designed for deployment and management Docker containers on clustered computer nodes. However, there are many container networking schemes for the clustering. Thus, in this paper, performance comparison based on throughput and response time from Web server which connected via 3 various container networking: Flannel, Calico and Canal will be evaluated. From the experimental results, Calico achieved best performance especially for heavy traffic requirement. Flannel and Canal stand the similar performance because both methods used VxLAN technology. However, Flannel is most simply for implemented.

Keyword: Docker Container, Container Networking, Kubernetes, Flannel, Calico, Canal

1. บทนำ

ทุกวันนี้เทคโนโลยีการประมวลผลเสมือนโดยเฉพาะด็อกเกอร์คอนเทนเนอร์ (Docker Container) มีบทบาทในระบบประมวลผลบนอินเตอร์เน็ตหรือคลาวด์ (Cloud) มากเนื่องมาจากการใช้ทรัพยากรที่น้อยกว่า และสามารถย้ายการติดตั้งได้ง่ายและทำงานได้เร็ว การทำงานร่วมกันเป็นคลัสเตอร์ของคอนเทนเนอร์กลายเป็นปัจจัยสำคัญในการสร้างระบบให้บริการบนระบบคลาวด์ ซึ่งสามารถจัดการได้โดยใช้งาน Orchestrator เช่น Docker Swarm หรือ Kubernetes [1] ซึ่งเป็นระบบควบคุมกำกับการทำงานในการประมวลผลขนาดเล็กที่เป็นระบบเสมือนได้ อย่างไรก็ตาม Kubernetes หรือ K8s มีแนวโน้มการใช้งานอย่างกว้างขวางกว่า ในการเชื่อมโยงระหว่างโหนดคอนเทนเนอร์ของ Kubernetes สามารถเลือกใช้งานได้จากหลายวิธี [2] ซึ่งสิ่งนี้ส่งผลต่อความเร็วและเวลาในการส่งข้อมูลระหว่างคอนเทนเนอร์และจากคอนเทนเนอร์มายังผู้ใช้งาน ซึ่งโดยปกติวิธีการแบบ Flannel จะเป็นวิธีที่ถูกแนะนำให้ใช้งานโดยปริยายจาก Google ดังนั้นในงานวิจัยชิ้นนี้ได้ทำการเปรียบเทียบประสิทธิภาพทางด้านเวลาในการ

ตอบสนองที่ใช้ในการตอบกลับ (Response Time) ไปยังผู้ใช้บริการของ 3 วิธีด้วยกัน คือ Flannel, Calico และ Canal

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 Kubernetes

Kubernetes เป็นเครื่องมือช่วยบริหารจัดการคอนเทนเนอร์ที่ทำงานอยู่ในระบบสภาพแวดล้อมแบบคลัสเตอร์ (Cluster) เป็นเครื่องมือโอเพ่นซอร์สที่บริษัท Google พัฒนาขึ้น ถูกนำมาใช้ในขั้นตอนการนำคอนเทนเนอร์ไปใช้งาน (Deployment) สำหรับ Kubernetes จะจัดการระบบเครือข่ายและจัดภาระการทำงานให้กับคอนเทนเนอร์ [3] โดยที่ผู้ใช้งานจะสั่งการทำงานผ่านทางโปรแกรม kubectl และเซอร์วิสต่างๆ จะทำงานอยู่บนเครื่องมาส์เตอร์ (Master Server) การทำงานของ Kubernetes มีส่วนประกอบดังนี้

2.1.1 Master Server

ส่วนควบคุมการทำงานหลักของ Kubernetes ผู้ใช้งานจะสั่งงานต่างๆ ผ่านทางเครื่องมาสเตอร์ไปยังเครื่องมินเนี่ยน (Minion Server) โดยเครื่องมาสเตอร์จะต้องมีโปรแกรมเพื่อทำงานกับ Kubernetes ดังนี้

2.1.1.1 Etcd

เป็นการให้บริการเก็บข้อมูลแบบ Key-Value ซึ่งแต่ละเครื่องเซิร์ฟเวอร์จะใช้ Etcd เป็นตัวกลางในการติดต่อสื่อสารภายในคลัสเตอร์ เช่น สภาพแวดล้อมต่างๆ ของเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ โดยการทำงานของ Etcd จะมีการเก็บข้อมูลในรูปแบบของ HTTP/JSON เมื่อต้องการเรียกดูข้อมูลสามารถเรียกข้อมูลของค่าที่ต้องการ โดย Etcd จะทำการตอบกลับค่ามาให้

2.1.1.2 API Service Server

ส่วนนี้เป็นส่วนสำคัญของเครื่องมาสเตอร์ และเป็นส่วนที่จัดการงานต่างๆ ของเครื่องที่อยู่ภายในคลัสเตอร์ โดยจะทำการอ่านค่าสถานะต่างๆ จาก Etcd

การเรียกใช้งาน API Service Server จะสามารถเรียกใช้งานผ่าน RESTful ซึ่งเป็นข้อได้เปรียบที่ทำให้ผู้พัฒนาสามารถพัฒนาเครื่องมือขึ้นมาติดต่อ API Service Server ได้เอง

2.1.1.3 Controller Manager Server

ส่วนที่ใช้ในการจัดการงานเกี่ยวกับการสร้างคอนเทนเนอร์ โดยจะทำการอ่านค่าสถานะต่างๆ ของคอนเทนเนอร์ผ่านทาง Etcd และเป็นส่วนที่ใช้ในการจัดการลดหรือการขยายการให้บริการด้วยเช่นกัน

2.1.1.4 Scheduler Server

ส่วนในการจัดการบริหารงานในแต่ละเครื่องเซิร์ฟเวอร์ภายในคลัสเตอร์ โดยจะทำการตรวจสอบไม่ให้มีการใช้งานทรัยากรเกินกว่าทรัพยากรที่มีว่างอยู่ และเมื่อมีการสร้างคอนเทนเนอร์ขึ้นใหม่ ส่วนนี้จะเป็นส่วนที่จัดการหาเครื่องมินเนี่ยนเพื่อรองรับคอนทนเนอร์ใหม่ที่เกิดขึ้น

2.1.2 Minion Server หรือ Worker Server

เครื่องมินเนี่ยนจะทำหน้าที่รองรับงานจากเครื่องมาสเตอร์ โดยเชื่อมต่อกับเครื่องมาสเตอร์ผ่านทางเครือข่าย ซึ่งเครื่องมินเนี่ยนจะมีโปรแกรมด็อกเกอร์ทำงานอยู่ เมื่อมีการสร้างงานขึ้นมาใหม่จากเครื่องมาสเตอร์ เครื่องมินเนี่ยนจะดำเนินการสร้างด็อกเกอร์คอนเทนเนอร์ขึ้นตามที่ได้รับงานมา ซึ่งเครื่องมินเนี่ยนจะมีส่วนของการทำงานดังต่อไปนี้

2.1.2.1 Kubelet Service

ส่วนที่ติดต่อกับเครื่องมาสเตอร์เพื่อรับข้อมูลหรือรับคำสั่งต่างๆ เช่น การรับคำสั่งสร้างคอนเทนเนอร์

2.1.2.2 Proxy Service

ส่วนที่ทำหน้าที่ส่งการร้องขอการให้บริการจากด็อกเกอร์โฮสต์ไปยังด็อกเกอร์คอนเทนเนอร์ และทำการส่งผลลัพท์กับคืน ซึ่งส่วนของ Proxy Service นี้จะสามารถจัดการการให้บริการในรูปแบบการกระจายงานได้ (Load Balancing)

2.1.3 หน่วยการทำงานของ Kubernetes

การทำงานของคอนเทนเนอร์ที่ทำงานอยู่บน Kubernetes ประกอบไปด้วยหน่วยการทำงานดังนี้

2.1.3.1 Pods

ชื่อเรียกของกลุ่มคอนเทนเนอร์ ในหนึ่ง Pods สามารถมีคอนเทนเนอร์ได้ตั้งแต่ 1 ตัวขึ้นไป แต่โดยทั่วไปนิยมให้มีแค่หนึ่งตัวในหนึ่ง Pods เพื่อใช้ในการรองรับการให้บริการแอพพลิเคชั่นที่ติดตั้งในคอนเทนเนอร์ และสำหรับแอพพลิเคชั่นสามารถมีจำนวนมากกว่าหนึ่ง Pods โดยขึ้นอยู่กับความสามารถในการรองรับการให้บริการของแอพพลิเคชั่นนั้น

2.1.3.2 Services

ส่วนของการเชื่อมต่อกับคอนเทนเนอร์ โดยส่วนนี้จะทำการส่งการร้องขอใช้บริการต่างๆ ไปยังคอนเทนเนอร์ที่อยู่ใน Pods หรือเรียกอีกอย่างได้ว่า Serviccs คือส่วนของอินเตอร์เฟสที่ใช้เชื่อมต่อระหว่างผู้ใช้บริการกับคอนเทนเนอร์ที่ให้บริการ

2.1.3.3 Replication Controllers

ส่วนจัดการเพิ่มหรือลด Pods เป็นในลักษณะของการขยายขนาดหรือลดขนาดการให้บริการ ซึ่งการขยายหรือลดจะเป็นในลักษณะรูปแบบแนวนอน (Horizontal) และหน้าที่อีกส่วนหนึ่งคือการจัดการในเรื่องรื้นของแอพพลิเคชัน (Version) ที่ทำงานในคอนเทนเนอร์ เช่น เดิมมี Pods ที่ให้บริการแอพพลิเคชั่นรุ่นที่หนึ่ง ต่อมาผู้ให้บริการต้องการนำแอพพลิเคชั่นรุ่นที่สองขึ้นมาทำงานแทน และเมื่อแอพพลิเคชั่นรุ่นที่สองสามารถทำงานได้แล้ว Replication Controllers จะทำการปิดแอพพลิเคชั่นรุ่นหนึ่งลงโดยอัตโนมัติ

2.2 Container Networking

Container Networking [4][5][6] ถูกนำมาใช้ในการแก้ไขปัญหาการเชื่อมโยงระหว่างคอนเทนเนอร์ ซึ่งมีอยู่หลายวิธีการสำหรับในงานวิจัยนี้นำเสนอวิธีการ 3 วิธี ดังนี้

2.2.1 Flannel

วิธีการหนึ่งที่ออกแบบมาสำหรับ Kubernetes เป็นวิธีที่ไม่ซับซ้อนในการติดตั้ง โดยการติดตั้ง Flannel ในแต่ละโฮสต์จะมีเอเจนต์ที่ชื่อว่า Flanneld มีหน้าที่ในการดูแลจัดสรร subnet ในแต่ละเครื่อง การทำงานของ Flannel จะเรียกใช้ Kubernetes API หรือ Etcd เพื่อเก็บค่าการตั้งค่าของเน็ตเวิร์ค จัดสรร Subnet และข้อมูลอื่นๆ เช่น Public IP-ของโฮสต์ เป็นต้น และในการส่งต่อแพ็คเก็ตจะถูกส่งออกโดยใช้กระบวนการหลายอย่างรวมถึง VxLAN

Flannel ทำหน้าที่ในการจัดการเครือข่ายในระบบเลเยอร์ 3 ระหว่างหลายๆ โหนดในคลัสเตอร์ แต่ไม่สามารถจัดการคอนเทนเนอร์ที่เชื่อมต่อกับโฮสต์ได้ ทำได้เพียงเฉพาะวิธีการรับส่งข้อมูลระหว่างโฮสต์

สำหรับ Kubernetes มองว่าแต่ละ Pods มีความเป็นเอกลักษณ์ (Unique) และมีการทำ Routable IP ในคลัสเตอร์ ข้อดีของวิธีการนี้กำจัดความซับซ้อนในการ Port Mapping ที่มาจากการแชร์ IP จากโฮสต์เดียว

2.2.2 Calico

เป็นวิธีการที่เน้นทางด้านความปลอดภัยในการเชื่อมต่อสำหรับคอนเทนเนอร์และปริมาณงานที่เกิดขึ้นในเวอร์ชวลแมชีน

Calico สร้างและจัดการเครือข่ายในระดับเลเยอร์ 3 ซึ่งจะมีการทำ Routable IP เต็มรูปแบบ สำหรับการส่งข้อมูลจะไม่ต้องใช้ IP encapsulation หรือการแปลที่อยู่ของเครือข่าย ทำให้สามารถทำงานร่วมกันได้ง่ายขึ้น ในสภาพแวดล้อมที่เป็นในลักษณะซ้อนทับ (Overlay) Calico ใช้ IP-in-IP tunneling หรือสามารถทำงานกับเครือข่ายซ้อนทับอื่นๆ ได้ เช่น Flannel

Calico มีการประยุกต์ใช้กฎทางด้านความปลอดภัย (Network Policy) กับเวอร์ชวลอินเตอร์เฟสสำหรับควบคุมคอนเทนเนอร์และเวอร์ชวลแมชีน รวมทั้งมีการใช้งานกับโฮสต์อินเตอร์เฟสสำหรับเซิร์ฟเวอร์และเวอร์ชวลแมชีน

2.2.3 Canal

เป็นวิธีการที่รวมระหว่าง Calico กับ Flannel โดยในปัจจุบัน Canal เป็นเพียงรูปแบบการติดตั้งและการตั้งค่าเพื่อให้ทำงานร่วมกันบนเครือข่ายเดียวกันได้ ซึ่งในอนาคตอาจจะมีการปรับเปลี่ยนรูปแบบรวมกับ Calico และ Flannel เพื่อลดความซับซ้อนในการติดตั้งและการตั้งค่า

2.3 งานวิจัยที่เกี่ยวข้อง

Hao Zeng และคณะ [9] ได้ทำการวัดประสิทธิภาพของ Containcr Nctwork ระหว่าง Flannel, Swarm Overlay และ Calico โดยมีการวัดค่าเวลาในการตอบกลับของการ Ping, TCP Throughput และ UDP Throughput พบว่า Calico มีประสิทธิภาพสูงสุด แต่การตั้งค่าในการใช้งานมีความซับซ้อนมากกว่าตัวอื่น โดยงานวิจัยนี้ไม่ได้พิจารณาทางด้านเวลาที่ใช้ในการตอบสนอง และ Container Network โดยวิธี Calico

Bin Xie และคณะ [10] ได้ทำการตรวจสอบประสิทธิภาพระหว่าง Host Mode กับ Overlay Network Mode บนแอพพลิเคชั่นที่เป็นรูปแบบ Streaming พบว่าในด้านการทำงาน Host Mode ให้ประสิทธิภาพดีว่าประมาณ 8% แต่ Overlay Network Mode ให้การทำงานที่มีเสถียรภาพมากกว่า

ด้านการถ่ายโอนข้อมูลระหว่างแพ็กเกจพบว่า Overlay Network Mode ให้ผลลัพท์ที่มากกว่า 50%

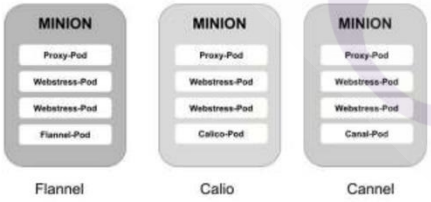
Jakob Struye และคณะ [11] ได้ทำการประเมินประสิทธิภาพเครือข่ายของคอนเทนเนอร์ระหว่างดือกเกอร์ RKT และ LXC โดยทำการประเมินค่า Throughput และ Latency จากการกำหนดเน็ตเวิร์คเป็น Host, Bridge (NAT) และ Macvlan พบว่าในด้านของ Throughput ทางด้าน LXC. ให้ประสิทธิภาพดีที่สุด ทางด้านเครือข่าย Macvlan ให้ประสิทธิภาพดีที่สุด

3. การดำเนินงานวิจัย

1. ดำเนินการออกแบบโครงสร้างของระบบที่ทำงานด้วย Kubernetes เพื่อให้บริการข้อมูล Web Server ด้วย Webstress บนเครื่องคอมพิวเตอร์จำนวน 3 เครื่อง โดยเป็นเครื่อง Master 1 เครื่อง และเครื่อง Minion 2 เครื่อง ซึ่งแต่ละโหนดกำหนดให้มี CPU 2 Core, Ram 2 GB และกำหนด Network Interface Card (NIC) ขนาด 100 Mbps

2. ตั้งค่า Container Network ให้กับระบบที่ได้ออกแบบไว้ โดยเมื่อทำการตั้งค่าแล้วเสร็จ จะเกิด Pods ของแต่ละ Container Network ที่แต่ละ Minion Server

ภาพที่ 1: โครงสร้างของระบบดือกเกอร์คูเบอร์เนตส์ที่ออกแบบ



ภาพที่ 2: รูปแบบการใช้งาน Container Network

3. ติดตั้ง Webstress :ซึ่งเป็นเครื่องมือที่ใช้ในการทดสอบประสิทธิภาพการทำงานของเซิร์ฟเวอร์ที่เครื่อง Minion ทั้ง 2 เครื่อง เครื่องละ 2 Pod

4. ติดตั้ง Apache JMeter [12] ที่เครื่องโฮสต์ ซึ่งเป็นเครื่องมือที่ใช้ในการจำลองผู้ใช้บริการทำการร้องขอจากเครื่องโฮสต์ไปยังเครื่อง Master ผ่านทางโปรโตคอล http โดยจำลองให้มีการร้องขอข้อมูลเว็บจากระบบที่ทดสอบดังนี้

4.1 วินาทีที่ 1: 100 requests

4.2 วินาทีที่ 2: 200 requests

4.3 วินาทีที่ 3: 300 requests

4.4 วินาทีที่ 4: 400 requests

4.5 วินาทีที่ 5: 500 requests

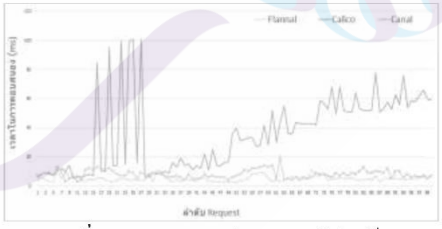
4.6 วินาทีที่ 6: 600 requests

โดยแต่ละ request ระบบจะตอบสนองด้วยการส่ง http response ขนาด 20 Kbytes จนครบทุก request

5. ประเมินประสิทธิภาพ โดยจะเป็นการประเมินประสิทธิภาพทางด้านเวลาในการตอบสนองที่ใช้ในการตอบกลับไปยังผู้ใช้บริการ และประสิทธิภาพทางด้านความเร็วในการรับและส่งข้อมูล (Upload/Download)

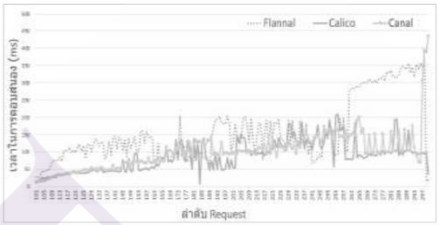
4. ผลการทดสอบและวัดประสิทธิภาพ

ผลการทดสอบประเมินประสิทธิภาพทางด้านเวลาในการตอบสนองและส่งกลับข้อมูล http response ของ Container Networking หรือรูปแบบการเชื่อมโยงเครือข่ายระหว่างคอนเทนเนอร์แต่ละประเภท ที่ได้จากการจำลองการทำงานและทดสอบในแต่ละช่วงเวลาแสดงดังรูปที่ 3 ถึง 6



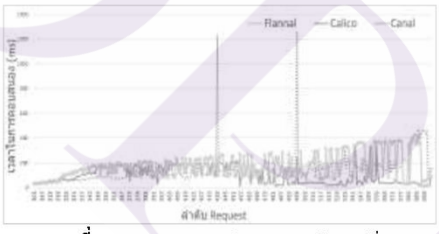
ภาพที่ 3: เวลาตอบสนอง กรณี 100 requests ในวินาทีที่ 1

จากภาพที่ 3 ผลการทดสอบเวลาที่ใช้ในการตอบสนองในกรณีมีการร้องขอจำนวน 100 requests จะพบว่าวิธีการของ Calico ให้ประสิทธิภาพทางด้านเวลาในการตอบสนองไม่ดี ใช้เวลาในการตอบสนองมากกว่าวิธี Flannel และ Canal

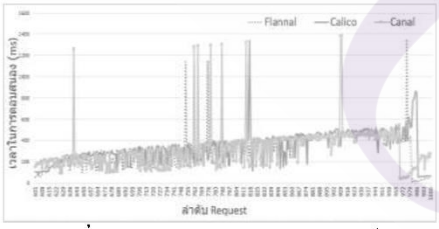


ภาพที่ 4: เวลาตอบสนอง กรณี 200 requests ในวินาทีที่ 2

จากภาพที่ 4 ผลการทดสอบเวลาที่ใช้ในการตอบสนองในกรณีมีการร้องขอจำนวน 200 requests ในวินาทีที่ 2 พบว่าวิธีการของ Calico ให้ประสิทธิภาพดีกว่าวิธีการของ Flannel และ Canal อย่างชัดเจน



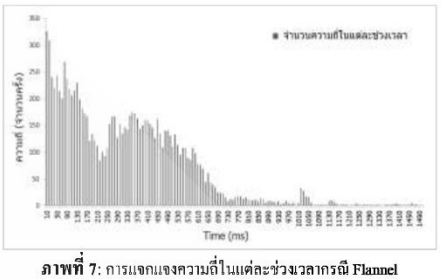
ภาพที่ 5: เวลาตอบสนอง กรณี 300 requests ในวินาทีที่ 3



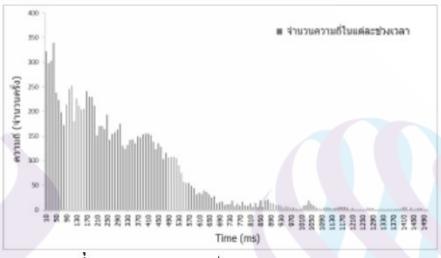
ภาพที่ 6: เวลาตอบสนอง กรณี 400 requests ในวินาทีที่ 4

จากภาพที่ 5 และ 6 ผลการทดสอบเวลาที่ใช้ในการตอบสนองในกรณีมีการร้องขอจำนวน 300 requests ในวินาทีที่ 4 และ 400 requests ในวินาทีที่ 4 ตามลำดับ ซึ่งมีทราฟิกจำนวนมากขึ้นมาก พบว่าวิธีการของ Calico ให้ประสิทธิภาพดีกว่าวิธีการของ Flannel และ Canal จึงกล่าวได้ว่าเมื่อมีการร้องขอเป็นจำนวนมากหรือ heavy traffic ตัว Calico มีแนวโน้มการตอบสนองได้เร็วกว่าวิธีการอื่น

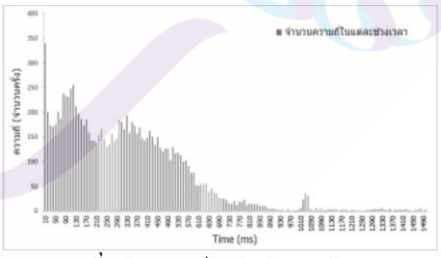
พิจารณาทางด้านการแจกแจงความถี่เวลาในการตอบสนองกับผู้ใช้บริการของการร้องขอข้อมูลทั้งหมดในแต่ละวิธี แสดงผลได้ดังภาพที่ 7 ถึง 8



ภาพที่ 7: การแจกแจงความถี่ในแต่ละช่วงเวลากรณี Flannel



ภาพที่ 8: การแจกแจงความถี่ในแต่ละช่วงเวลากรณี Calico



ภาพที่ 9: จำนวนความถี่ในแต่ละช่วงเวลากรณี Canal

จากภาพที่ 7, 8 และ 9 พบว่าวิธี Calico มีความถี่เวลาที่ใช้ในการตอบสนองที่ดีกว่าอีก 2 วิธี มีการกระจายตัวที่ค่อนข้างแคบกว่า แสดงให้เห็นถึงความมีประสิทธิภาพทางด้านเวลาที่ใช้ในการตอบสนองกับผู้ใช้งาน

อัตราการส่งผ่านข้อมูลหรือ throughput จากเว็บเซิร์ฟเวอร์ไปยังเครื่องที่ทำการร้องขอ ซึ่งคำนวณได้จาก Apache JMeter สรุปได้ดังตารางที่ 1

ตารางที่ 1: เปรียบเทียบอัตราการส่งผ่านข้อมูล (Kbyte/s)

Flannel	Calico	Canal
6,797.40	6,820.88	6,693.53

จากตารางพบว่าอัตรการส่งผ่านข้อมูลสำเร็จนั้นวิธีการ Calico จะให้ผลลัพธ์ที่ดีกว่าวิธีการอื่น สอดคล้องกับระยะเวลาที่ใช้ในการตอบสนองดังที่ได้กล่าวไปแล้ว

5. สรุป

จากผลการทดลองจะเห็นได้ว่า Container Networking ของ Kubernetes แบบ Calico มีประสิทธิภาพทางด้านเวลาที่ใช้ในการตอบสนองต่อการร้องขอดีที่สุด เมื่อเทียบกับ Flannel และ Canal เนื่องจากลักษณะของ Calico เป็นการจำลองเน็ตเวิร์คในระดับเลเยอร์ 3 และไม่มีการทำ Encapsulation หรือ Tunnel ทำให้มี Overhead น้อยกว่าอีก 2 วิธี ส่งผลให้ใช้เวลาในการตอบสนองน้อยกว่า และเป็นวิธีที่เหมาะกับกรณีที่ผู้ใช้งานมีเป็นจำนวนมาก สำหรับวิธี Flannel และ Canal ให้ผลทางด้านประสิทธิภาพคล้ายกัน เนื่องจากทั้ง 2 วิธีอยู่บนพื้นฐานของ VxLAN ซึ่งต้องทำการ Encapsulate เฟรมข้อมูลทำให้มี overhead เกิดขึ้น

สำหรับงานวิจัยในอนาคตสามารถทำการศึกษาปัจจัยทางด้านอื่นๆ ไม่ว่าจะเป็นทางด้านความปลอดภัย ความยุ่งยากในการติดตั้ง การตั้งค่า ความยืดหยุ่นในการใช้งาน หรือทดสอบในสภาพแวดล้อมอื่นๆ เพื่อให้สามารถวิเคราะห์ได้ว่าวิธีการต่างๆ นั้นเหมาะสมหรือไม่

เอกสารอ้างอิง

[1] "Comparing Orchestration Engines in Rancher, A Closer look at Docker Swarm and Kubernetes", Rancher Labs, July, 2017.

[2] Deploy on Kubernetes. Available online at https://docs.docker.com/docker-for-windows/kubernetes/

[3] Kubernetes. What is Kubernetes? Available online at http://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

[4] C. Boettiger, "An introduction to docker for reproducible research" *Acm Sigops Operating Systems Review*, Vol 49, Issue 1, pp. 71-79, 2015.

[5] D. Vohra, *Kubernetes microservices with docker*, Apres, Canada, 2016.

[6] E. Brewer, "Kubernetes and the path to cloud native," Proceedings of the Sixth ACM Symposium on Cloud Computing, New York, USA, August 27-29, 2015.

[7] Flannel is a network fabric for containers. Available online at https://github.com/coreos/flannel

[8] Project Calico v3.0. Available online at https://www.projectcalico.org/wp-content/uploads/2018/01/ProjectCalico.v3.datasheet.pdf

[9] Hao Zeng, "Measurement and Evaluation for Docker Container Networking" *IEEE Trans. on Cyber-Enabled Distributed Computing (CyberC)*, Nanjing, China, 12-14 Oct, 2017, pp.105-108.

[10] Bin Xie, "Docker Based Overlay Network Performance Evaluation in Large Scale Streaming System" *IEEE Trans. on Advanced Information Management Communicates (IMCEC)*, Xi'an, China, 3-5 Oct, 2016, pp.366-369.

[11] Jakob Struye, "Assessing the Value of Containers for NFVs" *International Conference on Network and Service Management* (CNSM), Tokyo, Japan, 26-30 Nov, 2017, pp.1-7.

[12] Apache JMeter. Available online at https://jmeter.apache.org/

ประวัติผู้เขียน

ชื่อ-นามสกุล	พรรักษา กมลเวชช์
ประวัติการศึกษา	พ.ศ. 2553 วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์) สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ตำแหน่งและสถานที่ทำงานปัจจุบัน	นักคอมพิวเตอร์ การประปานครหลวง
ประสบการณ์ ผลงานทางวิชาการ	พ.ศ. 2561 การประเมินสมรรถนะการเชื่อมต่อเครือข่ายคอนเทนเนอร์คูเบอร์เนตส์ งานประชุมวิชาการระดับชาติด้านคอมพิวเตอร์และเทคโนโลยีสารสนเทศ ครั้งที่ 14 (NCCIT 2018) พ.ศ. 2560 การปรับปรุงอัลกอริทึมสำหรับการจัดการการทำงานของซีพียูแบบอัตโนมัติของโปรแกรมตรวจสอบแรงดันน้ำในท่อประปาของการประปานครหลวง ที่มีการใช้งานโนดด็อทเจเอสในการรับ-ส่งข้อมูล