บทความวิจัย (Research Article)

การสเกลพอดแกนแนวนอนด้วยปฏิทินออนไลน์สำหรับคูเบอร์เนติส

The Horizontal Pod Autoscaler with an Online Calendar for Kubernetes

สิทธินนท์ ศรีสวัสดิ์ และ ทรงฤทธิ์ กิติศรีวรพันธุ์*

สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนครพนม
* ผู้รับผิดชอบบทความ
songrit@npu.ac.th

Received: 31 Dec 2020
Revised: 9 Apr 2021
Accepted: 29 Apr 2021

บทคัดย่อ

คูเบอร์เนติสเป็นซอฟต์แวร์แจกจ่ายฟรีใช้สำหรับบริหารจัดการคอนเทนเนอร์หรือพอด โดยมีคุณสมบัติรองรับความต้องการพื้นฐานสำหรับนักพัฒนาซอฟต์แวร์และผู้ดูแลระบบ อย่างไรก็ตามในส่วนเพิ่ม-ลดจำนวนพอดแบบอัตโนมัติตามขั้นตอนวิธีการสเกลพอดแกนแนวนอนนั้นมีการตอบสนองช้า ความล่าช้าเกิดจากระบบรอปริมาณโหลดปรากฏแล้วจึงเพิ่มปริมาณทรัพยากร งานวิจัยนี้จึงได้พัฒนาระบบคูเบอร์เนติสให้มีคุณสมบัติจัดเตรียมพอดได้ล่วงหน้าผ่านระบบปฏิทินออนไลน์ ซอฟต์แวร์มีคุณสมบัติอ่านกำหนดการจากปฏิทินออนไลน์และส่งข้อมูลควบคุมทรัพยากรคูเบอร์เนติสผ่านโพรโทคอลเอ็มคิวทีที อันประกอบด้วยจำนวนผู้ใช้และกำหนดวันนัดหมาย เมื่อระบบได้รับข้อมูลจะทำการขยายพอดโดยอัตโนมัติตามกำหนดนัดหมายของปฏิทินออนไลน์ จากการทดลองพบว่าเมื่อถึงกำหนดนัดหมายแล้วระบบจะเริ่มสร้างพอดโดยอัตโนมัติภายในระยะเวลาน้อยกว่า 10 นาที

คำสำคัญ: คูเบอร์เนติส การควบคุมพอด เอชพีเอ การสเกลพอดแกนแนวนอน

Abstract

Kubernetes is free distribution software for managing containers or Pods. It has a function to support the basic needs for developers and system administrators as well. However, the automatic scaling of Pods based on the horizontal pod autoscaler is a slow to response. This is caused by the process of waiting for the load to appear in the system and then adding resources, and the container preparation time causes a slow response from the unavailable time interval. In this research, developed the Kubernetes system to be able to prepare Pods in advance via an online calendar. The software has a feature to read the schedule from the online calendar and send it to control the Kubernetes resources via the MQTT protocol. This includes the number of users and the appointment date. When the system receives the information, it will automatically expand the Pods according to the online calendar appointments. Experiments have shown that once the appointment is due, the system will automatically begin creating a Pods in less than 10 minutes.

Keywords: Kubernetes, pod control, HPA, horizontal pod autoscaler

1. บทนำ

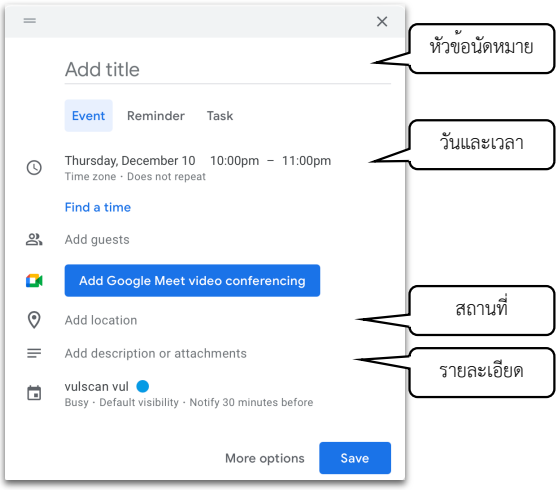
คอนเทนเนอร์ (container) เป็นเทคโนโลยีบรรจุซอฟต์แวร์และไลบรารีมารวมอยู่ภายในชุดเดียวกัน ซึ่งช่วยให้นักพัฒนาซอฟต์แวร์ส่งมอบซอฟต์แวร์ได้โดยไม่ขึ้นกับไลบรารีที่ติดตั้งใน

เครื่องลูกค้า นอกจากนั้นแล้วยังถ่ายโอนคอนเทนเนอร์ไปเครื่องแม่ข่ายเครื่องอื่นได้โดยไม่กระทบต่อการทำงาน ซึ่งเป็นประโยชน์ต่อการรองรับผู้ใช้จำนวนมากได้โดยง่าย

ลักษณะปัญหาด้านการควบคุมคอนเทนเนอร์เพื่อให้บริการได้ทันท่วงทีถือเป็นงานที่ยังได้รับความสนใจในวงการวิจัยซึ่งได้กล่าวถึงในหัวข้อ 3 จากการศึกษา [1-2] พบว่าการจัดสรรทรัพยากรในส่วนคอนเทนเนอร์ตอบสนองได้ช้า สาเหตุมาจากระบบรอให้มีการใช้ทรัพยากรเกินเงื่อนไข ทำให้เกิดช่วงเวลาในการสร้างพอด จึงได้เสนอวิธีการสเกลพอดแบบแกนแนวนอนเพื่อให้เพิ่มจำนวนคอนเทนเนอร์ได้ล่วงหน้าผ่านการควบคุมด้วยปฏิทินออนไลน์ จากการทดลองพบว่าช่วยให้ระบบตอบสนองต่อการขอใช้บริการได้รวดเร็วและเป็นการเพิ่มประสิทธิภาพเครือข่ายได้ดีกว่าใช้วิธีการสเกลพอดเพียงอย่างเดียว

1.1 ระบบปฏิทินออนไลน์

ระบบปฏิทินออนไลน์มีข้อดีหลายประการ เช่น ใช้บันทึกประวัติการนัดหมายและสืบค้นได้ง่ายในภายหลัง ใช้แจ้งเตือนได้ล่วงหน้าเป็นต้น ทำให้ผู้ใช้งานวางแผนงานได้โดยง่าย ระบบปฏิทินออนไลน์ที่เป็นที่รู้จักเช่น Google Calendar [3] หรือ ปฏิทินที่ติดตั้งมาพร้อมกับระบบปฏิบัติการ ปัจจุบันพบการใช้ปฏิทินทั้งในส่วนบุคคลและองค์กร ฟังก์ชันพื้นฐานของปฏิทินออนไลน์ประกอบด้วย หัวข้อนัดหมาย กำหนดวันและเวลา สถานที่และรายละเอียด รูปที่ 1 มีการกำหนดปฏิทินสำหรับ Google Calendar เป็นตามคุณสมบัติขั้นพื้นฐานของปฏิทินออนไลน์ดังที่กล่าวมาข้างต้น



รูปที่ 1 หน้าต่างกำหนดนัดหมายกูเกิลคาเลนดาร์

ปฏิทินออนไลน์เป็นแอปพลิเคชันพื้นฐานที่มีใช้โดยทั่วไป การประยุกต์ให้สามารถควบคุมคูเบอร์เนติสจึงช่วยลดการเรียนรู้วิธีใช้ซอฟต์แวร์ใหม่และไม่ต้องเปลี่ยนแปลงพฤติกรรมการทำงานจากเดิม ในการควบคุมดังกล่าวทำได้ผ่านซอฟต์แวร์บริหารคอนเทนเนอร์ ซอฟต์แวร์ที่นิยมใช้บริหารคอนเทนเนอร์ ได้แก่ คูเบอร์เนติส [4]

1.2 ระบบคูเบอร์เนติส

เครื่องคอมพิวเตอร์ในระบบคูเบอร์เนติสมี 2 ประเภท ได้แก่ มาสเตอร์โหนด (master node) และเวิร์คเกอร์โหนด (worker node) มาสเตอร์โหนดทำหน้าที่ควบคุมเวิร์คเกอร์โหนดซึ่งภายในเวิร์คเกอร์โหนดมีพอดที่บรรจุคอนเทนเนอร์ โดยโหนดทั้ง 2 เป็นสมาชิกภายในคลัสเตอร์ คลัสเตอร์อาจเป็นสมาชิกของระบบคลาวด์หรือไม่ก็ได้ เรียกกลุ่มของคลัสเตอร์ว่า เนมสเปซ (namespaces) เนมสเปซถือเป็นอ็อบเจกต์ (object) ประเภทหนึ่ง

1.3 การสร้างอ็อบเจกต์

การสร้างอ็อบเจกต์ทำได้สองรูปแบบได้แก่ วิธีป้อนคำสั่งโดยตรง (imperative) และวิธีประกาศค่า (declarative) การสร้างอ็อบเจกต์แบบป้อนคำสั่งโดยตรงทำได้ผ่านคำสั่ง kubectl ตามด้วยพารามิเตอร์ที่เกี่ยวข้อง ขณะที่การสร้างอ็อบเจกต์ด้วยวิธีประกาศค่าทำผ่านไฟล์ชนิด YAML (YAML ain't markup language) ตัวอย่างในโค้ดที่ 1 ถึง 3 เป็นตัวอย่างการสร้างเนมสเปซด้วยคำสั่ง kubectl และ YAML ตามลำดับ

```
kubectl create namespace example
```

โค้ดที่ 1 คำสั่งสร้าง namespace ด้วย kubectl ผ่านเชลล์

สำหรับโค้ดที่ 2 และ 3 ใช้อธิบายตัวอย่างการสร้างเนมสเปซแบบวิธีประกาศค่า

```
apiVersion: v1
kind: Namespace
metadata:
    name: example
```

โค้ดที่ 2 สร้าง namespace ด้วย YAML

```
kubectl apply -f example.yaml
```

โค้ดที่ 3 สร้างอ็อบเจกต์จากไฟล์ YAML

การสร้างอ็อบเจกต์ด้วยวิธีประกาศค่ามีความสะดวกว่าวิธีป้อนคำสั่งโดยตรง ซึ่งในงานวิจัยนี้ใช้วิธีสร้างอ็อบเจกต์ผ่าน YAML เป็นหลัก

1.4 การเพิ่ม-ลดขนาดพอด

คูเบอร์เนติสมีคุณสมบัติเพิ่ม-ลดขนาดพอดอัตโนมัติอยู่ 3 รูปแบบ ได้แก่ การสเกลพอดแกนแนวนอน (horizontal pod autoscaler, HPA) การสเกลพอดแกนแนวตั้ง (vertical pod autoscaler, VPA) และ การสเกลคลัสเตอร์(cluster autoscaler, CA) การสเกลพอดแกนแนวตั้งเป็นวิธีเพิ่ม-ลดทรัพยากรสำหรับพอดใดๆ โดยเพิ่มทรัพยากรในโหนดที่พอดนั้นรันอยู่ เช่น เพิ่มหน่วยประมวลผลกลาง หน่วยความจำ เนื้อที่จัดเก็บข้อมูลเป็นต้น การสเกลพอดแกนแนวนอน เป็นการปรับจำนวนพอด โดยพอดนั้นอาจติดตั้งที่โหนดเดียวกันหรือโหนดอื่นได้ โดยการเพิ่ม-ลดพอดเป็นเสมือนการเพิ่มจำนวนเครื่องคอมพิวเตอร์จึงเทียบได้กับการขยายตามแกนแนวนอนซึ่งระบบจะตรวจสอบการใช้ทรัพยากรหน่วยประมวลผลกลางเป็นเกณฑ์ควบคุม เมื่อค่าที่กำหนดเกินข้อกำหนด ตัวควบคุมจะปรับจำนวนแบบจำลอง ในตัวควบคุมแบบจำลองจะทำการปรับจำนวนพอด การสเกลคลัสเตอร์เป็นการเพิ่มจำนวนเวิร์คเกอร์โหนดเมื่อไม่อาจเพิ่มพอดในเวิร์คเกอร์โหนดได้อีก คุณสมบัตินี้พบเฉพาะในคลาวด์ภาคธุรกิจ [5] เช่น Google cloud platform และ Amazon web services

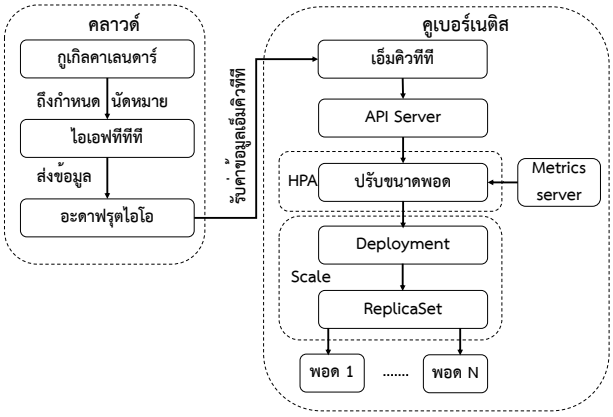
1.5 การเกิดเหตุการณ์ผู้ใช้หนาแน่นบางช่วงเวลา

เหตุการณ์ผู้ใช้หนาแน่นบางช่วงเวลาเป็นเหตุการณ์ที่ผู้ใช้จำนวนมากต้องการเข้าถึงบริการในช่วงเวลาเดียวกัน ทำให้ในช่วงเวลาดังกล่าวเกิดภาระโหลดกับเครื่องแม่ข่าย ซึ่งเป็นเหตุให้เครื่องแม่ข่ายปฏิเสธบริการ การแก้ปัญหาขั้นต้นทำได้โดยการจัดเตรียมทรัพยากรแม่ข่ายจำนวนมากไว้รองรับผู้ใช้ เช่น เครื่องแม่ข่ายหนึ่งเครื่องรองรับผู้ใช้ได้ 50 การเชื่อมต่อ เมื่อต้องการให้รองรับผู้ใช้จำนวน 1,000 การเชื่อมต่อ ทำได้โดยเตรียมเครื่องแม่ข่ายจำนวน 20 เครื่อง เป็นต้น ขณะที่เวลาในการให้บริการส่วนใหญ่ไม่มีการเข้าถึงหนาแน่น วิธีเปิดเครื่องแม่ข่ายไว้ตลอดเวลาจึงเป็นการใช้ทรัพยากรไม่เต็มประสิทธิภาพ ระบบคูเบอร์เนติสใช้วิธีเพิ่ม-ลดขนาดพอดแบบอัตโนมัติ ซึ่งได้กล่าวในหัวข้อ 1.4 นั้นตอบสนองช้าเมื่อมีความต้องการใช้ทรัพยากรมากในช่วงเวลาหนึ่ง โครงการวิจัยนี้จึงได้พัฒนาการสเกลพอดด้วยปฏิทินออนไลน์ขึ้นเพื่อให้ควบคุมพอดได้ล่วงหน้าและยังคงคุณสมบัติการสเกลพอดแกนแนวนอนได้เช่นเดิม

2. แนวคิดและทฤษฎี

ในการศึกษานี้ออกแบบให้ระบบมีคุณสมบัติสร้างพอดโดยผู้ดูแลระบบกำหนดจำนวนผู้ใช้บริการได้ล่วงหน้าผ่านทางปฏิทินการปฏิบัติงาน ข้อมูลที่ป้อนเข้าระบบประกอบด้วย วัน เวลา และค่าประมาณผู้ใช้งานในวันเปิดให้บริการ

จากข้อมูลดังกล่าวช่วยให้ผู้ดูแลระบบจัดเตรียมเครื่องแม่ข่ายให้พร้อมต่อความต้องการเข้าถึงแม่ข่ายได้ดีกว่าการใช้ระบบการสเกลพอดแกนแนวนอนเพียงอย่างเดียว โดยปฏิทินออนไลน์ใช้กำหนดพอดล่วงหน้า และยังคงใช้วิธีการสเกลพอดแกนแนวนอนเพื่อทำหน้าที่ปรับขนาดพอดในช่วงเวลาขณะให้บริการ ระบบเริ่มทำงานจากการกำหนดนัดหมายผ่านกูเกิลคาเลนดาร์



รูปที่ 2 ระบบสเกลพอดแกนแนวนอนด้วยปฏิทินออนไลน์

อธิบายการทำงานรูปที่ 2 ดังนี้ ที่คลาวด์ผู้ดูแลระบบป้อนกำหนดการผ่านกูเกิลคาเลนดาร์ เมื่อถึงกำหนดการข้อมูลจะส่งต่อไปให้ระบบไอเอฟทีทีที [12] และส่งต่อข้อมูลไปยังระบบอะดาฟรุตไอโอ [13] ระบบอะดาฟรุตไอโอจะสร้างการสื่อสารรูปแบบเอ็มคิวทีที [14] เมื่อคูเบอร์เนติสได้รับข้อมูลจำนวนผู้ใช้จากการประเมินโดยผู้ดูแลระบบผ่านโพรโทคอลเอ็มคิวทีที ได้แก่

จำนวนผู้ใช้งาน จึงเริ่มสร้างพอดผ่านทาง Deployment และ ReplicaSet การทำงานนี้รองรับการปรับขนาดพอดแกนแนวนอน (HPA) ผ่านทาง Metric Server ตามเดิม

3. งานวิจัยที่เกี่ยวข้อง

การศึกษา [2] ได้ศึกษาวิธีปรับปรุงประสิทธิภาพด้านการคำนวณหน่วยประมวลผลที่ต้องการจำนวนมาก ซึ่งโหลดจำนวนมากนั้นแบ่งการทำงานได้ (high-throughput computing (HTC) workloads) โดยเสนอวิธีการตรวจสอบและออกแบบตัวปรับขนาดแบบป้อนกลับอัตโนมัติ(high throughput autoscaler (HTA)) โดยวิเคราะห์ผ่านตัวต้นแบบพบว่าปรับปรุงประสิทธิภาพด้านการจัดการทรัพยากรดีขึ้น 5.6 เท่าเมื่อเทียบกับระบบคูเบอร์เนติสปกติ

การศึกษา [6] ศึกษาการกระจายโหลดของการสเกลคลัสเตอร์ โดยเปรียบเทียบระหว่าง CA และ CA-NAP พบว่า CA-NAP ให้ประสิทธิภาพดีกว่า CA นอกจากการศึกษาการทำงานคลาวด์สาธารณะมีนักวิจัย [1] ศึกษาด้านการทำโหลดบาลานซ์ พบว่าการกระจายงานหรือเรียกว่าโหลดบาลานซ์เซอร์ (load balancer) ใช้วิธี leader election algorithm มีโอกาสทำงานไม่เต็มประสิทธิภาพได้เมื่อโหนดที่ทำหน้าที่เป็นด่านหน้าก่อนการกระจายงาน (Leader) มีภาระโหลดสูง การศึกษานี้ได้ปรับปรุงวิธีกระจายโหลด HPA โดยศึกษาตัวชี้วัดที่มาพร้อมกับคูเบอร์เนติส (kuernetes resource metrics, KRS) กับตัวชี้วัดภายนอก (prometheus custom metris)

การศึกษา [7] ศึกษากระบวนการพัฒนาซอฟต์แวร์โดยใช้เครื่องมือเพื่อช่วยสร้างความต่อเนื่องในการพัฒนาซอฟต์แวร์ ทำให้มีการทำงานอัตโนมัติมากขึ้นกว่าเดิม โดยเครื่องมือนี้ช่วยทีมพัฒนาซอฟต์แวร์ และทีมปฏิบัติการปรับปรุงการทำงานประสานกันได้ราบรื่นเรียกว่าเดฟออปส์ (DevOps) กระบวนการเดฟออปส์ทำให้ขั้นตอนการพัฒนาซอฟต์แวร์เป็นแบบอัตโนมัติ ซึ่งการบรรจุซอฟต์แวร์บนคอนเทนเนอร์สามารถทำได้อัตโนมัติผ่านกระบวนการเดฟออปส์ ด้วยกระบวนการทางซอฟต์แวร์สมัยใหม่นี้จึงผลักดันให้แนวคิดการเขียนซอฟต์แวร์แบบไมโครเซอร์วิสทำได้ในกระบวนการเดฟออปส์

การศึกษา [8] ศึกษาการสร้างระบบการปรับขนาดพอดแกนแนวนอนที่มีประสิทธิภาพเหมาะสำหรับแอปพลิเคชันโดยไม่จำเป็นต้องมีการแทรกแซงของมนุษย์ โดยจะพูดถึงพื้นฐานการใช้เครื่องมือคูเบอร์เนติส และอธิบายเมตริกแต่ละประเภทเช่น Kubernetes Resource Metrics (KRM) และ Prometheus Custom Metrics (PCM) และอธิบายผลกระทบต่อประสิทธิภาพของการปรับขนาดพอดแกนแนวนอน

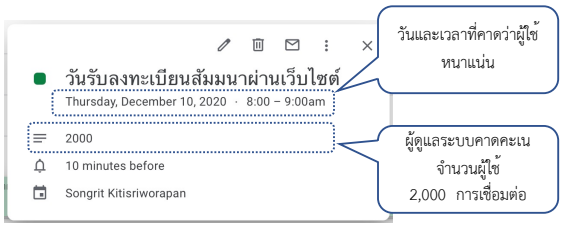
การศึกษา [9] เสนอเฟรมเวิร์กสำหรับจัดเตรียมและจัดการการปรับใช้คลัสเตอร์ สำหรับระบบขนาดใหญ่ โดยใช้แพลตฟอร์มคูเบอร์เนติส และบริการตรวจสอบการทำงานผ่านตัวดำเนินการ Prometheus นอกจากนี้เฟรมเวิร์กนี้ยังทำหน้าที่เป็นฐานข้อมูลให้กับเครื่องมือ Grafana เพื่อแสดงภาพการทำงานของระบบแบบเรียลไทม์

การศึกษา [10] ศึกษากรอบการทำงานสำหรับตรวจสอบและจัดการการดำเนินงานของศูนย์ข้อมูลในเชิงรุกโดยการผสานรวมเทคโนโลยีล้ำสมัยเช่น คูเบอร์เนติส, Prometheus, Grafana และแพลตฟอร์มการคาดการณ์ กับข้อมูลจากเมตริกเซ็นเซอร์และเครื่องมือวิเคราะห์ โดยระบบปรับขนาดคลัสเตอร์คูเบอร์เนติสแบบไดนามิก สามารถลดการสิ้นเปลืองทรัพยากรจากการรับประกัน Quality of service (QoS) ได้

การศึกษา [11] ศึกษาระบบเอพีไอเกตเวย์เป็นทางเข้าสู่บริการแบ็กเอนด์ สามารถลดปริมาณการติดต่อระยะไกลระหว่างแอปพลิเคชัน และบริการแบ็กเอนด์ได้อย่างมีประสิทธิภาพ และลดความซับซ้อนของบริการภายในที่ติดต่อหากัน โดยออกแบบระบบปรับขนาดอัตโนมัติสำหรับระบบเอพีไอเกตเวย์บนคูเบอร์เนติส และ Prometheus ซึ่งสามารถปรับขนาดจำนวนแอปพลิเคชันได้ตลอดเวลา ปรับปรุงการใช้ทรัพยากรระบบในขณะที่บริการของแอปพลิเคชันมีคุณภาพและมีความพร้อมให้บริการ

4. วิธีดำเนินการ

การศึกษานี้เลือกกูเกิลคาเลนดาร์เป็นแอปพลิเคชันปฏิทินออนไลน์ แอปพลิเคชันไอเอฟทีทีที และอะดาฟรุตไอโอ ทำงานร่วมกัน เพื่ออ่านปฏิทินและสร้างรูปแบบการสื่อสารแบบโพรโทคอลเอ็มคิวทีทีเพื่อปรับขนาดพอดแกนแนวนอน ในลำดับต่อไประบบนี้อ่านปฏิทินออนไลน์จากข้อมูลอธิบายในรูปที่ 3



รูปที่ 3 การกำหนดจำนวนผู้ใช้ล่วงหน้าผ่านกูเกิลคาเลนดาร์

เมื่อถึงกำหนดนัดหมายแอปพลิเคชันไอเอฟทีทีจะอ่านข้อมูลจากกูเกิลคาเลนดาร์ และส่งต่อไปยังอะดาฟรุตไอโอที่ได้ตั้งค่าฟีด (feeds) และส่งข้อมูลไปยังเครื่องเซิร์ฟเวอร์ที่ได้ติดตั้งคูเบอร์เนติส ซึ่งรับค่าด้วย mosquitto_sub ลำดับการทำงานเป็นไปตามอัลกอริทึม 1

Algorithm 1: การสเกลพอดด้วยปฏิทินออนไลน์

Input:
U_r ← รับจำนวนผู้ใช้จากเอ็มคิวทีที
Output:
สร้างพอดโดยใช้ข้อมูลจากปฏิทิน
begin
// U_p Number of users per pod 50
// H Number of spare pods for HPA
U_p ← 50
H ← 5
while exist(Ur) do
minPod = $\lceil U_r / U_p \rceil$
maxPod = minPod + H
Deployment (minPod)
ReplicaSet (minPod, maxPod)
end while
end

จากรูปที่อัลกอริทึม 1 รับข้อมูลโพรโทคอลเอ็มคิวทีทีผ่านโปรแกรม mosquitto_sub จาก io.adafruit.com ถ้าไม่มีข้อมูลจะรอจนกระทั่งได้รับข้อมูล เมื่อมีข้อมูลจะนำข้อมูลไปเก็บในตัวแปร U_r นำค่าที่ได้คำนวณจำนวนพอด เมื่อหนึ่งพอดรองรับผู้ใช้ได้จำนวน U_p ดังนั้นจำนวนพอดที่ระบบต้องสร้างจึงเท่ากับ U_r/U_p โดยกำหนดค่าเริ่มต้น U_p เท่ากับ 50 ผู้ใช้งานต่อหนึ่งพอด ในกรณีผลคำนวณไม่จำนวนเต็มปรับด้วยฟังก์ชันเพดาน (ceil)

ผลลัพธ์ที่ได้เป็นจำนวนเริ่มต้นของพอดแทนด้วย minPod ซึ่งในที่นี้คือจำนวนพอดต่ำสุด และคำนวณค่า maxPod ซึ่งเป็นจำนวนพอดสูงสุด ด้วยการนำตัวแปร minPod มาบวกเพิ่ม H เพื่อมารองรับทรัพยากรในกรณีที่มีผู้ใช้เข้ามาเพิ่ม และนำตัวแปรทั้งสองมาทำการเพิ่มลงในไฟล์ YAML เพื่อทำการปรับขนาดพอด

ผลทดสอบการทำงานของระบบสเกลพอดแกนแนวนอนด้วยปฏิทินออนไลน์อธิบายในหัวข้อต่อไป

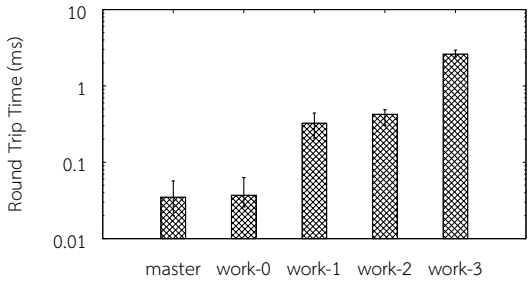
5. ผลการวิจัย

สำหรับการทดลองได้ติดตั้งซอฟต์แวร์คูเบอร์เนติสรุ่น v1.20.5 (kubeadm) มีมาสเตอร์โหนดและเวิร์คเกอร์โหนดรวมเป็นจำนวนสี่เครื่อง เครื่องมาสเตอร์ทำหน้าที่เป็นเวิร์คเกอร์มีรายละเอียดฮาร์ดแวร์ในตารางที่ 1

ตารางที่ 1 คุณสมบัติคอมพิวเตอร์แม่ข่าย

Worker	CPU	Core	*vCPU	Mem.
work-0	2*Xeon X3670	12	24	32G
work-1	2*Xeon X3670	12	24	32G
work-2	Xeon E3-1220	4	4	8G
work-3	I5-7500T	2	2	8G

*vCPU = Thread



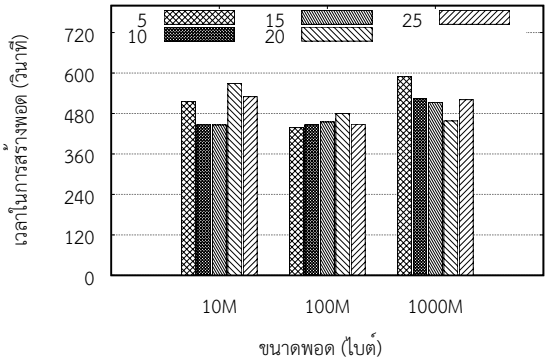
รูปที่ 4 เวลาติดต่อระหว่างมาสเตอร์และเวิร์คเกอร์

เวิร์คเกอร์ในระบบมีทั้งหมด 4 โหนดมี work-3 เป็นเครื่องอยู่ภายนอกเครือข่าย รูปที่ 4 ได้รายงานผลการทดสอบคำสั่ง ping ไปทุกโหนดในระบบ

ในหัวข้อต่อไปได้ทดลองการทำงานของระบบ โดยแบ่งการทดลองออกเป็น 3 การทดลองประกอบด้วย การทดลองระยะเวลาการทำงานของระบบ การทดลองการจัดสรรทรัพยากร และ การทดลองระยะเวลาการทำงานของระบบคลาวด์ มีรายละเอียดดังนี้

5.1 ระยะเวลาการทำงานของระบบ

เมื่อถึงกำหนดนัดหมายระบบจะอ่านจำนวนผู้ใช้ผ่านทางโพรโทคอลเอ็มคิวทีที เมื่อได้ข้อมูลจำนวนผู้ใช้จะประเมินจำนวนพอดโดยใช้อัลกอริทึม 1 ในการทดสอบนี้กำหนดให้จำนวนผู้ใช้ต่อพอดเท่ากับ 50 (U_p=50) การทดลองนี้ได้ทดสอบสร้างพอดมีขนาดพอดแตกต่างกันทั้งหมด 3 ขนาดได้แก่ 10M, 100M และ 1,000Mbytes อธิบายได้ในผลการทดลองในรูปที่ 5 จากผลการทดลอง การสร้างพอดจำนวน 5 ถึง 25 พอด สำหรับระบบสเกลพอดแกนแนวนอน ในกรณีกำหนดจำนวนพอดล่วงหน้าผ่านปฏิทินออนไลน์ ผลการทดลองนี้เกิดจากการสร้าง Replica โดยเริ่มบันทึกเวลานับจากกำหนดนัดหมายบนกูเกิลคาเลนดาร์จนกระทั่งสร้างพอดครบใช้เวลาประมาณ 480 วินาที และใช้เวลาสูงสุดประมาณ 600 วินาที (10 นาที)



รูปที่ 5 การจัดสรรทรัพยากรซีพียูบนโหนดเครื่องเดียว

สำหรับการทำงานแบบสเกลพอดแกนแนวนอน การศึกษา [2] พบว่าการสร้างพอดจาก 0 ถึง 16 พอดใช้เวลาประมาณ 13 นาที ซึ่งต้องรอระยะเวลาในการปรับค่า Metric server ค่ามาตรฐานอยู่ที่ 15 วินาที [16]

5.2 ระยะเวลาการจัดสรรทรัพยากร

เหตุการณ์ผู้ใช้หนาแน่นบางช่วงเวลาเป็นเหตุการณ์ที่ผู้ใช้จำนวนมากต้องการเข้าถึงบริการในช่วงเวลาเดียวกัน ทำให้ในช่วงเวลาดังกล่าวเกิดภาระโหลดกับเครื่องแม่ข่าย ซึ่งเป็นเหตุให้เครื่องแม่ข่ายปฏิเสธบริการ

การทดลองนี้มีวัตถุประสงค์เพื่อทดสอบการกระจายโหลดเมื่อมีการใช้ทรัพยากรซีพียูเกินข้อกำหนด ในการทดลองนี้ได้สร้างพอดให้บริการเว็บเซิร์ฟเวอร์และทำหน้าที่คำนวณรากที่สองตามสมการที่ (1) จำนวนหนึ่งล้านรอบต่อหนึ่งการเชื่อมต่อ หรือเรียกว่า 1-เซสชัน เมื่อ F_n แทนผลจากการคำนวณที่ n รอบ เริ่มต้นที่ F_0 = 0.001 คำนวณแบบเรียกตัวเองจำนวนหนึ่งล้านรอบ (n=10^6) [15]

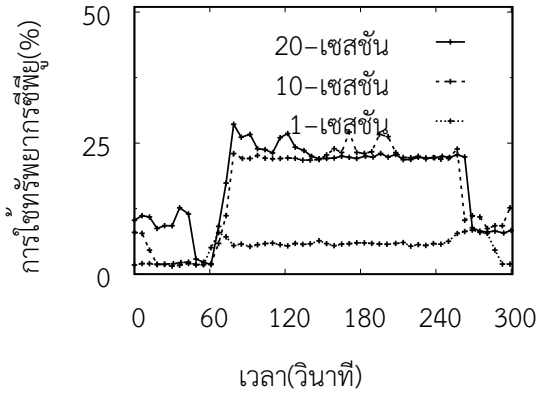
$$F_n = \begin{cases} 0.0001, & n = 0 \\ \sqrt{F_n + \sqrt{F_{n-1}}}, & n < 10^6 \end{cases} \quad (1)$$

จำลองโหลดด้วยการกำหนดให้ลูกข่ายส่งข้อมูลเชื่อมต่อเว็บเซิร์ฟเวอร์ 3 รูปแบบได้แก่ 1-เซสชัน 10-เซสชัน และ 20-เซสชัน รันเป็นเวลา 2 นาที การทดลองนี้ได้เปรียบเทียบระหว่างการมีเครื่องแม่ข่ายเครื่องเดียว และการใช้ระบบสเกลพอดแกนแนวนอน โดยสังเกตการแชร์โหลดไปยังโหนดที่เหลือ 3 โหนดในระบบ ผลทดลองในรูปที่ 5 อธิบายผลการใช้งานทรัพยากรซีพียูของ work-0 เพียงเครื่องเดียว นับจาก 0 ถึง 50%

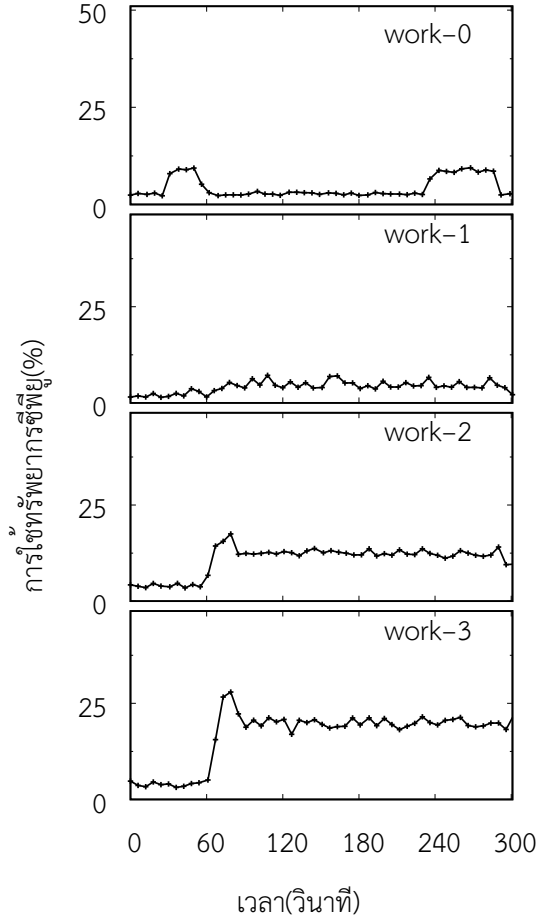
ผลการทดลองในรูปที่ 6 ทดลองรันโหลด 1 10 และ 20-เซสชัน ตามลำดับ พบว่าการสร้างโหลดจำนวน 20-เซสชัน ทำให้มีการใช้ทรัพยากรซีพียูบนเครื่อง work-0 ประมาณ 25% ใกล้เคียงกับ 10-เซสชัน และการทดลอง 1-เซสชัน ใช้ทรัพยากรซีพียูน้อยที่สุด จากกราฟเห็นได้ว่าเครื่อง work-0 สามารถรองรับโหลดได้ 10-เซสชัน ขีดจำกัดการใช้ทรัพยากรซีพียูของ work-0 อยู่ที่ประมาณ 25%

จากการทดลองรันโปรแกรมบน work-0 เพียงเครื่องเดียว ลำดับต่อมาการทดลองรูปที่ 7 ได้ส่งโหลดเข้าสู่ระบบที่รองรับการทำงานแบบสเกลพอดแกนแนวนอนโดยให้ระบบกระจายโหลดไปโหนดที่ว่าง ซึ่งในระบบนี้มีโหนดทั้งหมดสี่โหนด จากผลการทดลองพบว่าในช่วงแรก work-0 ได้รับโหลดและมีการกระจายการทำงานไปโหนดที่เหลือ ซึ่งทรัพยากรของเครื่อง

work-2 และ work-3 ใช้ทรัพยากรซีพียูมากที่สุด มีทรัพยากรหน่วยประมวลผล 2,000 มิลลิคอร์



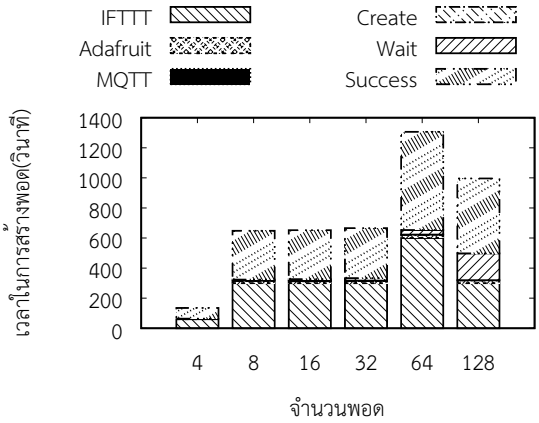
รูปที่ 6 การจัดสรรทรัพยากรซีพียูบนโหนดเครื่องเดียว



รูปที่ 7 การจัดสรรทรัพยากรซีพียูวิธีสเกลพอดแนวนอนจำนวน 20-เซสชัน บนทุกโหนด

5.3 ระยะเวลาการทำงานของระบบคลาวด์

จากผลการทดลองรูปที่ 8 แบ่งเวลาเป็นสองกลุ่ม ได้แก่ การทำงานภายในของคูเบอร์เนติส ได้แก่ Create, Wait, Success และการทำงานของเซอร์วิส ได้แก่ MQTT, Adafruit, IFTTT สำหรับการทำงานเซอร์วิสพบว่าระยะเวลาส่วนใหญ่เกิดจากการทำงานของ IFTTT และขั้นตอนการสร้างพอด



รูปที่ 8 ระบบสเกลพอดแกนแนวนอนด้วยปฏิทินออนไลน์

6. สรุป

งานวิจัยนี้ใช้กูเกิลคาเลนดาร์สร้างกำหนดนัดหมาย แล้วใช้บริการคลาวด์ไอเอฟทีทีที ในการเชื่อมโยงข้อมูลจากกูเกิลคาเลนดาร์ไปยังอะดาฟรุตไอโอ บริการคลาวด์มีเพื่อสร้างการเชื่อมต่อโพรโทคอลเอ็มคิวทีที ข้อมูลที่ส่งผ่านเอ็มคิวทีทีประกอบด้วยจำนวนผู้ใช้และกำหนดวันนัดหมาย

ระบบนี้สร้างพอดได้จากการกำหนดล่วงหน้าผ่านปฏิทินออนไลน์ โดยทำงานร่วมกับระบบสเกลพอดแกนแนวนอน เพื่อให้สามารถรองรับกรณีผู้ใช้หนาแน่นบางช่วงเวลา และได้มีการแบ่งการทดลองออกเป็นสามรูปแบบ คือ การทำงานของระบบ การทดลองการจัดสรรทรัพยากร และการทดลองระยะเวลาการทำงานของระบบคลาวด์ จากการทดลองพบว่าระบบสามารถทำงานได้ด้วยการกำหนดปฏิทินออนไลน์ล่วงหน้า โดยไม่ต้องการการเปลี่ยนพฤติกรรมการทำงานของผู้ดูแลระบบ เมื่อถึงกำหนดนัดหมายแล้วระบบจะเริ่มสร้างพอดโดยอัตโนมัติ ช่วยให้ระบบแม่ข่ายพร้อมให้บริการได้ทันต่อความต้องการ

7. เอกสารอ้างอิง

[1] N. Nguyen and T. Kim, “Toward Highly Scalable Load Balancing in Kubernetes Clusters,” in IEEE Communications Magazine, Vol. 58, no. 7, pp. 78-83, July 2020.

[2] C. Zheng, N. Kremer-Herman, T. Shaffer and D. Thain, “Autoscaling high-throughput workloads on container orchestrators,” in 2020 IEEE International Conference on Cluster Computing (CLUSTER), pp. 142-152, Sep 2020.

[3] Google. (12 September 2020). Google calendar. [Online] Available : https://www.google.com/calendar/about/

[4] E. Truyen, D. V. Landuyt, D. Preuveneers, B. Lagaisse and W. Joosen, A comprehensive feature comparison study of open-source container orchestration frameworks, Feb 2020.

[5] T. T. Nguyen, Y. J. Yeom, T. Kim, D. H. Park and S. Kim, “Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration,” Sensors (Switzerland), Vol. 20, no. 16, pp. 1–18, Aug 2020.

[6] M. Tamiru, J. Tordsson, E. Elmroth and G. Pierre, “An Experimental Evaluation of the Kubernetes Cluster Autoscaler in the Cloud,” in CloudCom 2020 - 12th IEEE International Conference on Cloud Computing Technology and Science, pp.1-9, Dec 2020.

[7] สนิธพรรณ จิตตั้งสมบูรณ์ และ มฑุปายาส ทองมาก. “ปัจจัยที่ส่งผลต่อความตั้งใจใช้แนวคิดการพัฒนาซอฟต์แวร์แบบ DevOps,” จุฬาลงกรณ์ธุรกิจปริทัศน์, 40(158), 53-110, 2561.

[8] T. Nguyen, Y. Yeom, T. Kim, D. Park and S. Kim, “Horizontal Pod Autoscaling in Kubernetes for Elastic Container Orchestration,” Sensors (Basel), Vol. 20, no. 16, pp. 4621, Aug 2020.

[9] N. Sukhija and E. Bautista, “Towards a framework for monitoring and analyzing high performance computing
environments using kubernetes and prometheus,” in 2019 IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computing, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ ATC/CBDCom/IOP/SCI), pp. 257-262, Aug 2019.

[10] Q. Wu, J. Yu, L. Lu, S. Qian and G. Xue, “Dynamically adjusting scale of a kubernetes cluster under qos guarantee,” in 2019 IEEE 25th International Conference on Parallel and Distributed Systems (ICPADS), pp. 193-200, Dec 2019.

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

[12] IFTTT. (9 December 2020). Ifttt helps every thing work better together. [Online] Available : https://ifttt.com/

[13] Adafruit. (10 December 2020). Welcome to adafruit io. [Online] Available : https://io.adafruit.com/

[14] R. A. Light, “Mosquitto: server and client implementation of the MQTT protocol,” The Journal of Open Source Software, Vol. 2, no. 13, pp. 265, 2017.

[15] T. K. Authors. (11 December 2020). Horizontal pod autoscaler walkthrough | kubernetes. [Online] Available : https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/

[16] The Kubernetes Authors. (17 March 2020). Horizontal pod autoscaler | kubernetes. [Online] Available : https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/