การขยายตัวจัดเก็บบันทึกจราจรเครือข่ายด้วยสถาปัตยกรรมไมโครเซอร์วิส

นายชาคริต ผาอินทร์

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2560

SCALING NETWORK TRAFFIC LOGGER WITH MICROSERVICE ARCHITECTURE

Mr. Chakrit Phain

A Thesis Submitted in Partial Fulfillment of the Requirements
for the Degree of Master of Science Program in Computer Science
Department of Computer Engineering
Faculty of Engineering
Chulalongkorn University
Academic Year 2017

หัวข้อวิทยานิพนธ์	การขยายตัวจัดเก็บบันทึกจราจรเครือข่ายด้วยสถาปัตยกรรมไมโครเซอร์วิส
โดย	นายชาคริต ผาอินทร์
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์

(รองศาสตราจารย์ ดร.สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ

(ผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

(รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์)

.....กรรมการภายนอกมหาวิทยาลัย

(อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต)

ชาคริต ผาอินทร์ : การขยายตัวจัดเก็บบันทึกจราจรเครือข่ายด้วยสถาปัตยกรรมไมโครเซอร์วิส (SCALING NETWORK TRAFFIC LOGGER WITH MICROSERVICE ARCHITECTURE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร.ญาใจ ลิ่มปิยะกรณ์, 57 หน้า.

ในช่วงไม่กี่ปีที่ผ่านมา องค์กรขนาดใหญ่จำนวนมากได้วิวัฒนาการเทคโนโลยีสแต็คองค์กรไปสู่ไมโครเซอร์วิส หรือที่รู้จักกันว่า สถาปัตยกรรมไมโครเซอร์วิส ถึงแม้ว่าสถาปัตยกรรมแบบดั้งเดิมโมโนลิทิกยังคงเป็นตัวเลือกที่ดีสำหรับแอปพลิเคชันจำนวนมาก แต่ยังมีข้อจำกัดในเรื่องการขยายตัวของระบบ การเปลี่ยนแปลงที่เกิดขึ้นกับส่วนเล็กๆ ส่วนหนึ่งในแอปพลิเคชันสามารถทำให้ต้องพัฒนาซ้ำระบบโมโนลิทิกทั้งระบบ ส่งผลให้ยากต่อการบำรุงรักษาโครงสร้างความเป็นมอดูลที่ดีในระยะยาว งานวิจัยนี้จึงได้นำเสนอการใช้สถาปัตยกรรมไมโครเซอร์วิสสำหรับการออกแบบตัวจัดเก็บบันทึกจราจรเครือข่าย บนพื้นฐานของแบบจำลองขยายตัวที่เรียกว่า ลูกบาศก์การขยายตัว แต่ละเซอร์วิสสามารถขยายตัวได้แบบปัจเจก ส่งผลให้แอปพลิเคชันขยายตัวอย่างมีประสิทธิภาพมากยิ่งขึ้น การศึกษาเบื้องต้นเพื่อประเมินสมรรถนะแนวทางการออกแบบที่นำเสนอ จากผลการทดลองแสดงให้เห็นว่า ล็อกเกอร์ที่พัฒนาบนพื้นฐานไมโครเซอร์วิสใช้เวลาค้นถามข้อมูลบนเรเดียสล็อกน้อยกว่าเมื่อเปรียบเทียบกับล็อกเกอร์ที่พัฒนาแบบโมโนลิทิก

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิทยาศาสตร์คอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาหลัก .....

ปีการศึกษา 2560

5970917221 : MAJOR COMPUTER SCIENCE
KEYWORDS: TRAFFIC LOG MANAGEMENT / INFORMATION SECURITY / MICROSERVICES / SCALABILITY

CHAKRIT PHAIN: SCALING NETWORK TRAFFIC LOGGER WITH MICROSERVICE ARCHITECTURE. ADVISOR: ASSOC. PROF. YACHAI LIMPIYAKORN, Ph.D., 57 pp.

Over the past few years, many large organizations evolve their technology stack to Microservices, also known as the microservice architecture. Although the traditional monolithic architecture is still a good choice for many applications, it does have limitations in scalability. A change made to a small part of the application also requires the entire monolith to be rebuilt and deployed. This makes it harder to maintain a good modular structure over time. This paper thus presents a design of network traffic logger using the microservice architecture. Based on the scaling model, scale cube, each service can be individually scaled, with the result being that the application is scaled more efficiently. The preliminary study was carried out for performance evaluation. The results show that the microservice-based logger yields less query times on Radius log retrieval, compared to the monolithic.

Department: Computer Engineering Student's Signature .....
Field of Study: Computer Science Advisor's Signature .....
Academic Year: 2017

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความอนุเคราะห์อย่างดียิ่งของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งทางได้สละเวลาให้ความรู้ ให้คำปรึกษา ตรวจสอบ ให้คำแนะนำแนวทางการวิจัย และสนับสนุนในทุกๆด้านจนทำให้การวิจัยนี้สำเร็จได้โดยสมบูรณ์ยิ่ง ข้าพเจ้าขอกราบระลึกถึงพระคุณของรองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ ไว้ ณ โอกาสนี้

ขอขอบพระคุณ คุณบิดา มารดา และครอบครัวของข้าพเจ้า ซึ่งเป็นกำลังใจที่ดีเสมอ รวมถึงขอบพระคุณผู้บังคับบัญชาในสายงาน เพื่อนร่วมงาน และมิตรสหาย ที่คอยให้กำลังใจ ให้การสนับสนุนและความช่วยเหลือในด้านต่างๆ

ท้ายสุด ข้าพเจ้าขอขอบพระคุณผู้ที่เกี่ยวข้องท่านอื่นๆที่ไม่ได้กล่าวมาในข้างต้น ซึ่งมีผลให้วิทยานิพนธ์นี้สำเร็จลุล่วงไปได้ด้วยดี ผู้วิจัยหวังเป็นอย่างยิ่งว่า วิทยานิพนธ์ฉบับนี้จะเป็นประโยชน์บ้างไม่มากก็น้อย สำหรับผู้ที่สนใจจะศึกษารายละเอียดต่อไป

สารบัญ

หน้า

หน้า

หน้า

สารบัญตาราง

สารบัญรูป

บทที่ 1
บทนำ

1.1. ที่มาและความสำคัญของปัญหา

พระราชบัญญัติว่าด้วยการกระทำผิดทางคอมพิวเตอร์ พ.ศ.2550 ได้ระบุในมาตรา 26 ว่า “ผู้ให้บริการ ต้องเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ไว้ไม่น้อยกว่าเก้าสิบวัน นับแต่วันที่ข้อมูลนั้นเข้าสู่ระบบคอมพิวเตอร์ แต่ในกรณีจำเป็น พนักงานเจ้าหน้าที่ จะสั่งให้ผู้ให้บริการผู้ใด เก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ ไว้เกินเก้าสิบวัน แต่ไม่เกินหนึ่งปีเป็นกรณีพิเศษเฉพาะราย และเฉพาะคราวก็ได้” [1] จึงเป็นที่มาของข้อบังคับการจัดทำระบบจัดเก็บข้อมูลการจราจรคอมพิวเตอร์ ซึ่งการออกแบบระบบโดยทั่วไปจะเป็นแบบรวมศูนย์ และใช้ซิสล็อก (Syslog) เป็นมาตรฐานการรับข้อมูลโดยเป็นโปรโตคอลในการรับส่งล็อก ทั้งนี้ การจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์แบบรวมศูนย์นั้นจะรวบรวมล็อกประเภทต่างๆ เก็บไว้ยังที่เดียวกัน จุดประสงค์เพื่อให้ง่ายต่อการรวม (Aggregate) และสอบถาม (Query) อย่างไรก็ตาม เมื่อปริมาณข้อมูลการจราจรคอมพิวเตอร์เพิ่มขึ้น จึงมีความต้องการการขยายตัว (Scalability) ซึ่งระบบจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์แบบรวมศูนย์ จะมีข้อจำกัดในการขยายตัวแบบแนวนอน (Horizontal Scaling) ซึ่งการขยายตัวแบบแนวนอนนั้น จะสามารถเพิ่มจำนวนโหนดได้ เพื่อกระจายการทำงานของซอฟต์แวร์ให้สามารถรับภาระ (Load) ได้มากขึ้น

ปัจจุบัน เทคโนโลยีข้อมูลขนาดใหญ่ (Big data) เข้ามามีบทบาทในการจัดเก็บโดยใช้เทคโนโลยีฐานข้อมูล NoSQL รองรับการขยายตัวแบบแนวนอน ซึ่งมีส่วนช่วยให้การขยายตัวมีประสิทธิภาพมากขึ้น แต่การจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์ ประกอบด้วยหลายส่วนเช่น Radius log หรือ Firewall log ซึ่งหากจัดเก็บในรูปแบบสถาปัตยกรรมโมโนลิทิก (Monolithic Architecture) จะยังไม่มีความเป็นอิสระต่อกัน และเมื่อมีการค้นคืน พบว่าประสิทธิภาพจะถูกใช้งานอย่างเต็มที่ โดยไม่มีความจำเป็น แต่หากมีการจำแนกประเภทของข้อมูลตามจุดประสงค์การใช้งานจะช่วยลดภาระในการค้นคืนให้มีประสิทธิภาพอย่างเหมาะสม

งานวิจัยนี้จึงได้นำเสนอการใช้สถาปัตยกรรมไมโครเซอร์วิส (Microservice Architecture) สำหรับการจัดการระบบจัดเก็บข้อมูลจราจรทางคอมพิวเตอร์แทนการจัดการแบบรวมศูนย์เดิม โดยแยกส่วนบริการแต่ละบริการให้เป็นอิสระต่อกัน เพื่อเป็นประโยชน์ในการจัดการของแต่ละเซอร์วิส และเพื่อให้การขยายตัวของแต่ละเซอร์วิสกระทำได้ง่ายขึ้น

1.2. วัตถุประสงค์ของงานวิจัย

เพื่อศึกษาค้นคว้าและออกแบบการพัฒนาระบบด้วยสถาปัตยกรรมไมโครเซอร์วิส เพื่อเป็นแนวทางการจัดเก็บข้อมูลขนาดใหญ่ และมีการขยายตัวอย่างรวดเร็ว

1.3. ขอบเขตการดำเนินงาน

1) ใช้ระบบการจัดเก็บบันทึกจราจรเครือข่ายเป็นกรณีศึกษา
2) ข้อมูลที่ใช้คือล็อกของ Radius และ ล็อกของ Firewall ยี่ห้อใดยี่ห้อหนึ่งเท่านั้น
3) ส่วนประกอบต่างๆที่ถูกจำลองนั้นรวมถึงส่วนประกอบที่เป็นเว็บแอปพลิเคชัน
4) ประเมินผลโดยเปรียบเทียบการใช้งานระหว่างแบบ Monolithic และ แบบMicroservice ในด้านการประสิทธิภาพขยายตัว และ ประสิทธิภาพการใช้งาน

1.4. ขั้นตอนการวิจัย

1) ศึกษาค้นคว้าทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2) ออกแบบระบบการขยายตัวเพื่อรองรับข้อมูลจราจรเครือข่ายปริมาณมากในอนาคต
3) พัฒนาระบบจัดเก็บบันทึกจราจรเครือข่าย
4) ทดสอบระบบที่ได้จัดเตรียมไว้และบันทึกข้อมูลผลลัพธ์
5) วิเคราะห์ข้อมูลที่ได้จากการทดสอบทั้งในส่วนของประสิทธิภาพของการขยายและข้อมูลการรับส่งระหว่างเซอร์วิส
6) สรุปผลและเรียบเรียงวิทยานิพนธ์

1.5. ประโยชน์ที่คาดว่าจะได้รับ

1) เป็นต้นแบบสำหรับการพัฒนาระบบจัดเก็บและค้นคืนข้อมูลจราจรเครือข่ายด้วยไมโครเซอร์วิส
2) ลดการผูกติดระหว่างเซอร์วิสทำให้ง่ายต่อการดูแลรักษาระบบ
3) รองรับการขยายตัวของระบบในอนาคต
4) นำโมเดลไปประยุกต์ใช้กับระบบอื่น ๆ ที่มีการขยายตัวสูงได้

1.6. ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

เนื้อหาวิทยานิพนธ์ฉบับนี้ถูกจัดแบ่งออกเป็น 6 บท โดยเริ่มจาก บทที่ 1 บทนำได้กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของงานวิจัย ขอบเขตงานวิจัย ประโยชน์ที่คาดว่าจะได้รับ วิธีดำเนินการวิจัย และ ผลงานตีพิมพ์จากวิทยานิพนธ์ บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้องที่นำมาใช้ในงานวิจัยนี้ บทที่ 3 แนวคิดและวิธีวิจัย บทที่ 4 การพัฒนาระบบ สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา บทที่ 5 การทดสอบระบบ ทดสอบการทำงานของระบบ และการวิเคราะห์ผล และบทที่ 6 สรุปผลการวิจัย ข้อจำกัด และแนวทางสำหรับการทำวิจัยต่อในอนาคต

1.7. ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์ได้นับการตีพิมพ์บทความวิชาการ 2 บทความ ประกอบด้วย

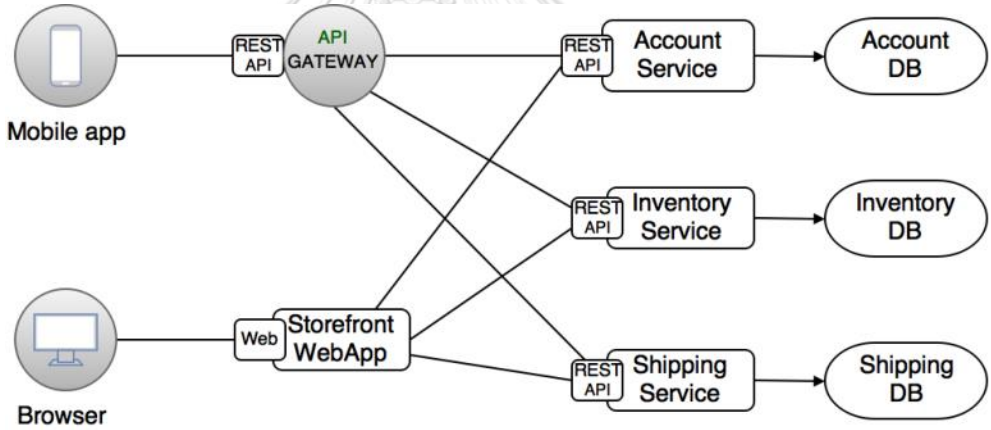
1) C. Phain and Y. Limpiyakorn, “Scaling Network Traffic Logger with Microservice Architecture” ในรายงานประชุมวิชาการนานาชาติสืบเนื่องจาก The 9th IEEE International Conference on System Science and Engineering (ICSSE 2018), Taipei Taiwan.
2) C. Phain and Y. Limpiyakorn, “Implementation of Microservice based Network Traffic Logger” ในรายงานการประชุมวิชาการนานาชาติสืบเนื่องจาก The 8th International Workshop on Computer Science and Engineering (WCSE 2018), Bangkok Thailand.

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1. ทฤษฎีที่เกี่ยวข้อง

2.1.1. สถาปัตยกรรมไมโครเซอร์วิส

ไมโครเซอร์วิส คือ สถาปัตยกรรมซอฟต์แวร์ที่รองรับการพัฒนาแอปพลิเคชันที่เปลี่ยนจากเดิม ซึ่งมีความซับซ้อนมาเป็นส่วนประกอบต่างๆ ที่ย่อยให้เล็กลง และไม่ผูกติดกับระบบ เรียกว่าเซอร์วิส แต่ละเซอร์วิสจะมีขนาดเล็กและโฟกัสในหน้าที่ของตนเอง โดยหน้าที่จะมีขนาดไม่ใหญ่มาก ทั้งนี้ แต่ละเซอร์วิสจะสื่อสารกันผ่าน REST APIs ดังรูปที่ 1 แสดงตัวอย่างแอปพลิเคชันซอฟต์แวร์ที่ออกแบบด้วยสถาปัตยกรรมไมโครเซอร์วิส



รูปที่ 1 ตัวอย่างการออกแบบแอปพลิเคชันด้วยสถาปัตยกรรมไมโครเซอร์วิส [2]

แต่ละเซอร์วิสจะถูกออกแบบให้ทำงานในส่วนงานของตนเอง และติดต่อสื่อสารผ่าน HTTP API แต่ละเซอร์วิสสามารถเขียนด้วยภาษาใดก็ได้ โดยไม่จำเป็นต้องใช้ภาษาเดียวกัน และฐานข้อมูลก็ไม่จำเป็นต้องใช้ฐานข้อมูลประเภทเดียวกัน ในแต่ละเซอร์วิสสามารถ สร้าง (Build) ทดสอบ (Test) และ ปล่อย (Release) โดยไม่ขึ้นต่อกัน ซึ่งประโยชน์ของการใช้ไมโครเซอร์วิสมีข้อดีคือ [3]

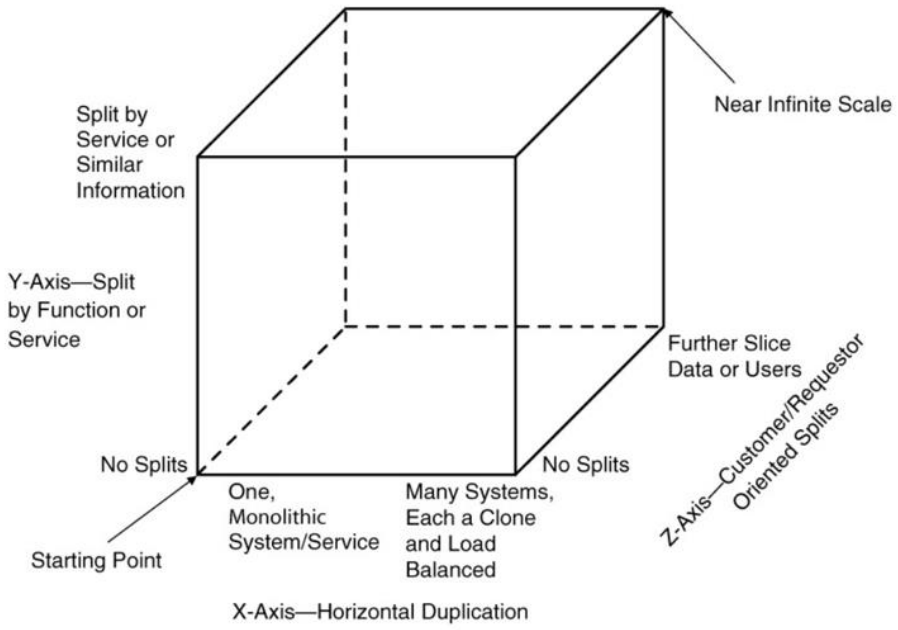
1. ลดความซับซ้อนของระบบ
2. แต่ละเซอร์วิสมีการพัฒนาโดยไม่ขึ้นตรงต่อกันทำให้ทีมสามารถโฟกัสไปยังเซอร์วิสที่ตนดูแลอยู่
3. แต่ละเซอร์วิสสามารถปรับใช้งานได้โดยไม่ขึ้นตรงต่อกันและกัน

4. แต่ละเซอร์วิสสามารถขยายตัวได้อย่างอิสระและไม่ขึ้นตรงต่อกันและกัน

สถาปัตยกรรมไมโครเซอร์วิสถูกออกแบบมาให้กระจายศูนย์ (Decentralized) [4] และถูกออกแบบมาให้มีการส่งมอบแบบต่อเนื่อง (Continuous delivery) และรวดเร็ว ซึ่งต้องถูกจัดเตรียมเครื่องมือต่าง ๆ ให้การ Release ไม่กระทบกับฟังก์ชันเดิมที่มีอยู่

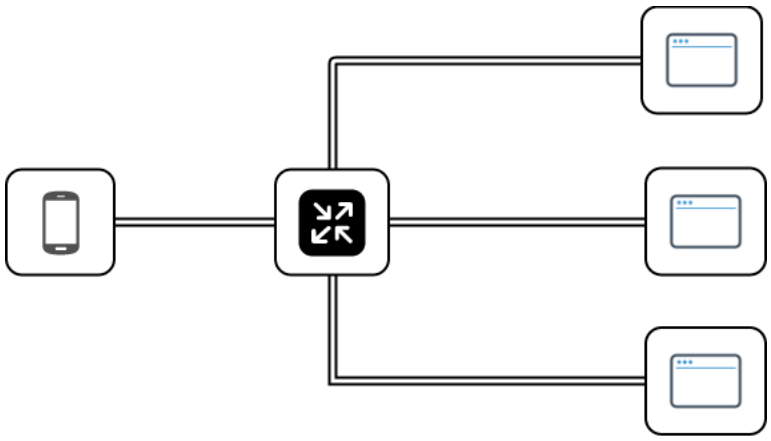
2.1.2. ลูกบาศก์การขยาย (Scale Cube)

โมเดลได้ออกแบบความสามารถการขยายเป็น 3 ประเภท ประกอบด้วย การขยายบนแกน X แกน Y และแกน Z ดังรูปที่ 2



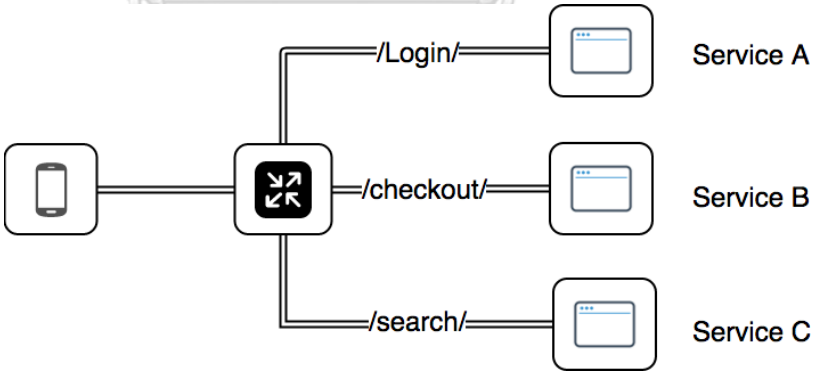
รูปที่ 2 ลูกบาศก์การขยาย [5]

โดยแกน X คือ การขยายแบบโหลดบาลานซ์ (Load Balancer) นิยมเรียกว่า Scale out เป็นการขยายแบบแนวนอน (Horizontal scaling) สร้างโดยการโคลน โดยมีคุณลักษณะที่เหมือนกัน N จำนวน ข้อมูลที่จะต้องจัดการในแต่ละ Node คือ 1/N สามารถแสดงได้ดังรูปที่ 3



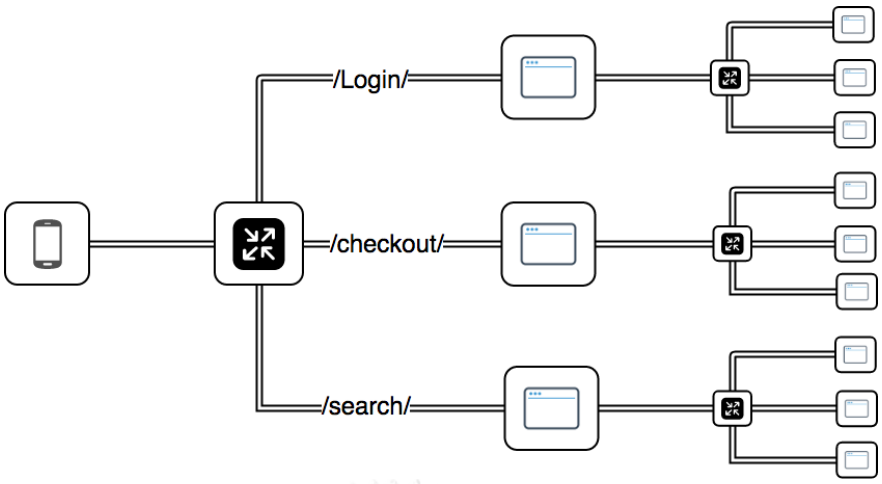
รูปที่ 3 การขยายตัวในแกน X

การขยายในแกน X และแกน Z สามารถสำเนาแอปพลิชันของตัวเองได้ แต่การขยายในแกน Y จะแตกต่าง กล่าวคือ เป็นการแบ่งเซอร์วิสออกเป็นหลายๆ ส่วน โดยแต่ละเซอร์วิสมีหน้าที่รับผิดชอบเพียงหนึ่งฟังก์ชัน หรือฟังก์ชันที่เกี่ยวข้องกัน นิยมจำแนกออกเป็นสองประเภท คือ อิงกริยา (verb-based) ที่จำแนกด้วยคำกริยา เช่น Checkout เป็นต้น และ อิงนาม (noun-based) ที่จำแนกด้วยคำนามหรือการดำเนินงานที่เฉพาะเจาะจง เช่น การจัดการลูกค้า เป็นต้น รูปที่ 4 แสดงตัวอย่างการขยายในแกน Y คือ การจำแนกเป็น 3 ส่วน คือ “login” “checkout” และ “search”



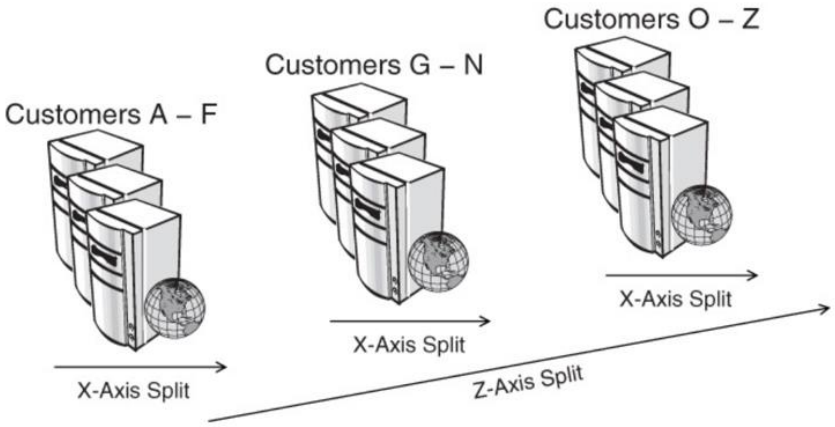
รูปที่ 4 ตัวอย่างการขยายในแกน Y

ซึ่งหากนำมารวมกันระหว่างแกน X และแกน Y จะเป็นการผสานการขยาย ทำให้สามารถขยายได้ทั้งแบบฟังก์ชัน และแบบ load balancer แสดงได้ดังรูปที่ 5



รูปที่ 5 ตัวอย่างการผสานการขยายตัวในแกน X และ แกน Y

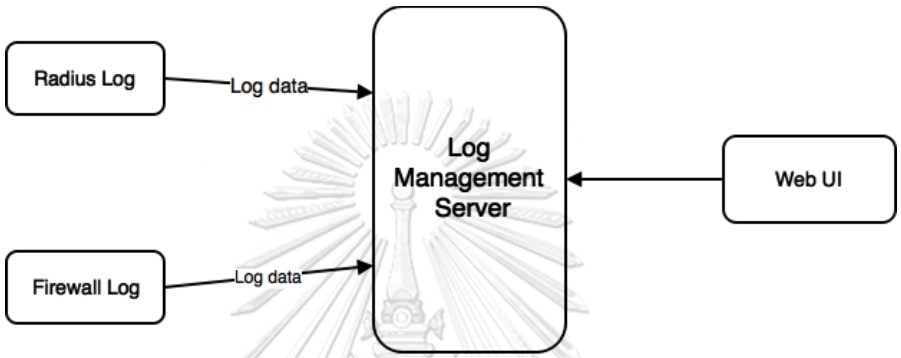
ในส่วนของการขยายในแกน Z นั้น แต่ละเซิร์ฟเวอร์จะเรียกใช้โค้ดที่เหมือนกัน ซึ่งมีความคล้ายคลึงกับการปรับมาตราส่วนแกน X ส่วนที่แตกต่างกัน คือ แต่ละเซิร์ฟเวอร์จะรับผิดชอบเพียงส่วนย่อยของข้อมูลเท่านั้น โดยมีเกณฑ์การกำหนดเส้นทาง โดยทั่วไปคือแอตทริบิวต์ (attribute) เกณฑ์การกำหนดเส้นทางทั่วไปอื่นๆ ได้แก่ ประเภทลูกค้า ตัวอย่างเช่น แอปพลิเคชันที่ถูกกำหนดโดย SLA กล่าวคือ ที่สูงกว่าจะอยู่เซิร์ฟเวอร์หนึ่ง และลูกค้าที่มี SLA ที่ต่ำกว่า จะอยู่อีกเซิร์ฟเวอร์หนึ่ง รูปที่ 6 แสดงการขยายในแกน Z โดยใช้เกณฑ์การแบ่งแยกตามตัวอักษร เช่น A-G, H-O, และ P-Z ตามลำดับ



รูปที่ 6 ตัวอย่างการขยายในแกน Z ตามเกณฑ์ตัวอักษร [5]

2.1.3. ระบบจัดเก็บบันทึกจราจรเครือข่าย

การจัดเก็บบันทึกจราจรเครือข่ายส่วนใหญ่จะถูกนำเข้าจากอุปกรณ์ต่างๆ เข้ามายังศูนย์กลางเดียวกัน ซึ่งจะถูกเรียกว่า การจัดเก็บบันทึกข้อมูลแบบรวมศูนย์ (Centralized log management) ดังแสดงในรูปที่ 7. Firewall log และ Radius log จะถูกส่งมาจัดเก็บไว้ยังที่เดียวกันคือ Log Management Server และสามารถค้นคืนได้ผ่าน Web UI โดยระบบจะให้บริการจัดเก็บ Log ที่ถูกส่งมาจากอุปกรณ์ต่าง ๆ และสามารถค้นคืนได้เมื่อต้องการ



รูปที่ 7 การจัดเก็บข้อมูลจราจรด้วยระบบรวมศูนย์

บันทึกจราจรเครือข่าย แบ่งออกเป็น 2 ประเภทหลัก กล่าวคือ

2.1.3.1 เรเดียสล็อก (Radius log) เป็นข้อมูลที่จัดเก็บการล็อกอินและล็อกเอ้าท์ ของผู้ใช้บริการโดยรูปแบบของเรเดียสล็อกจะมีลักษณะของล็อกเป็นแนวตั้งดังรูปที่ 8 และมีข้อมูลสำคัญดังตารางที่ 1

ตารางที่ 1 รายละเอียดข้อมูลของเรเดียสล็อก

Attribute	Description
Time stamp	บันทึกเวลาของกิจกรรมนั้นๆ
User-Name	ชื่อของผู้ที่ทำการล็อกอิน
Framed-IP-Address	ไอพีแอดเดรสของผู้ที่ทำการล็อกอิน
Acct-Status-Type	สถานะของผู้ใช้งาน เช่น เริ่มใช้ (Start) หรือสิ้นสุด (Stop)

```
Fri Dec 15 18:00:24 2000
      Acct-Session-Id = "2193976896017"
      User-Name = "e2"
      Acct-Status-Type = Start
      Acct-Authentic = RADIUS
      Service-Type = Framed-User
      Framed-Protocol = PPP
      Framed-IP-Address = 11.10.10.125
      Calling-Station-Id = "+15678023561"
      NAS-IP-Address = 11.10.10.11
      NAS-Port-Id = 8
      Acct-Delay-Time = 0
      Timestamp = 976896024
      Request-Authenticator = Unverified

Fri Dec 15 18:32:09 2000
      Acct-Session-Id = "2193976896017"
      User-Name = "e2"
      Acct-Status-Type = Stop
      Acct-Authentic = RADIUS
      Acct-Output-Octets = 5382
      Acct-Input-Octets = 7761
      Service-Type = Framed-User
      Framed-Protocol = PPP
      Framed-IP-Address = 11.10.10.125
      Acct-Session-Time = 1905
      NAS-IP-Address = 11.10.10.11
      NAS-Port-Id = 8
      Acct-Delay-Time = 0
      Timestamp = 976897929
      Request-Authenticator = Unverified
```

รูปที่ 8 ตัวอย่างเรเดียสล็อก

2.1.3.2 ไฟร์วอลล์ล็อก (Firewall log) หรือเรียกว่า แนทล็อก (NAT log: Network Address Translation log) เป็นล็อกที่ประกอบด้วยไอพีต้นทาง ไอพีปลายทาง พอร์ตต้นทาง พอร์ตปลายทางดังรูปที่ 9 และแสดงรายละเอียดดังตารางที่ 2

```
2006-09-19 03:04:29 OPEN TCP 11.10.10.125 10.20.72.204 3599 445
2006-09-19 03:04:29 OPEN TCP 11.10.10.125 10.20.72.204 3600 139
```

รูปที่ 9 ตัวอย่างไฟร์วอลล์ล็อก [6]

ตารางที่ 2 รายละเอียดข้อมูลของไฟร์วอลล์ล็อก

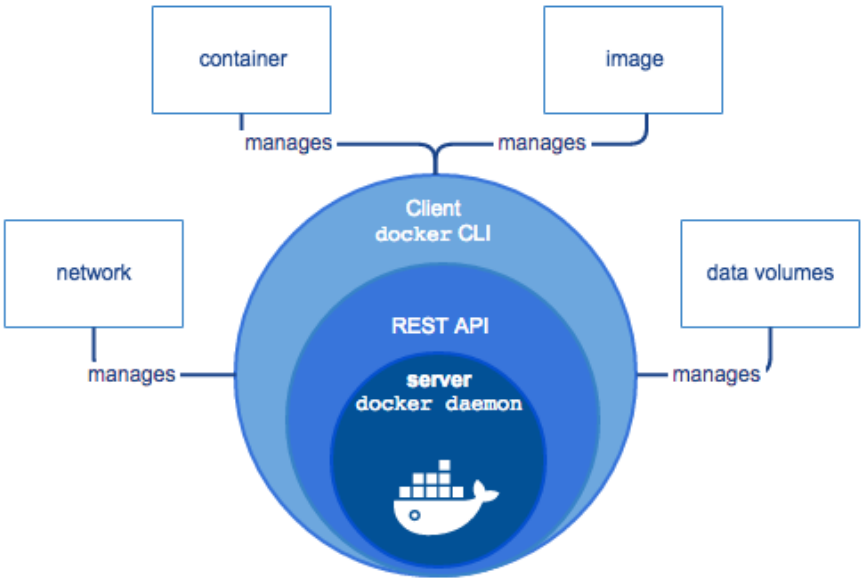
Attribute	Description
Time stamp	บันทึกเวลาของกิจกรรมนั้นๆ
Source IP	ไอพีต้นทางของผู้ใช้งาน
Source Port	พอร์ตต้นทางของผู้ใช้งาน
Destination IP	ไอพีปลายทางที่ผู้ใช้งานเรียกใช้บริการ
Destination Port	พอร์ตปลายทางที่ผู้ใช้งานเรียกใช้บริการ

2.1.4. ด็อคเกอร์ (Docker)

ด็อคเกอร์(Docker) [7] คือ ผู้นำด้านแพลตฟอร์มซอฟต์แวร์คอนเทนเนอร์ (Software Container Platform) นักพัฒนาซอฟต์แวร์นิยมใช้เพื่อลดปัญหาเรื่องความสามารถใช้งานได้บนเครื่องของตนเอง ในขณะที่เมื่อนำไปใช้งานเครื่องจริงกลับใช้งานไม่ได้ และในระดับองค์กรนิยมนำด็อคเกอร์มาใช้สร้างเอไจล์ (Agile) ซอฟต์แวร์ในการส่งงานลักษณะไปป์ไลน์

Docker Engine คือ แอปพลิเคชันไคลเอ็นต์เซิร์ฟเวอร์พร้อมกับส่วนประกอบสำคัญต่างๆ ดังนี้ เซิร์ฟเวอร์ เป็นโปรแกรมชนิดหนึ่งเรียกว่ากระบวนการ daemon ในที่นี้คือคำสั่ง dockerd

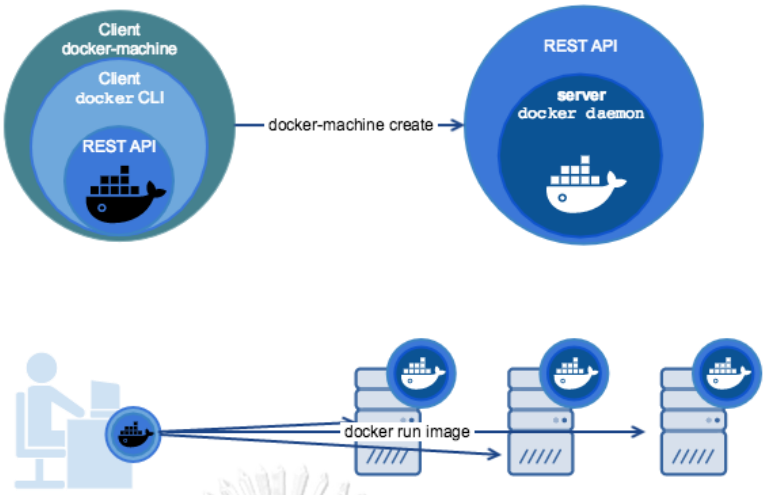
REST API คือ อินเทอร์เฟซที่โปรแกรมสามารถใช้เพื่อพูดคุยกับ daemon ส่วนติดต่อบรรทัดคำสั่ง (command line interface) เป็นเครื่องมือสำหรับ ไคลเอนต์เรียกว่า docker command



รูปที่ 10 ส่วนประกอบของ Docker engine

จากรูปที่ 10 ส่วนติดต่อบรรทัดคำสั่งจะทำการส่งคำสั่งไปยัง REST API เพื่อส่งต่อไปยัง docker daemon (dockerd) ซึ่งแอปพลิเคชันอื่นๆจะทำการเรียกผ่านเอพีไอและซีแอลไอเช่นกัน

Docker Machine คือซอฟต์แวร์สำหรับติดตั้ง Docker Engine บน virtual hosts ด้วยคำสั่ง docker-machine สามารถสร้างได้บน Mac หรือ วินโดวส์บ็อกส์ (Windows box) รวมถึงสามารถส่งคำสั่งไปยัง cloud providers เช่น Azure , AWS หรือ Digital Ocean ได้อีกด้วย ดังรูปที่ 11



รูปที่ 11 รูปแบบจำลองการสั่งงานด้วยคำสั่ง docker-machine

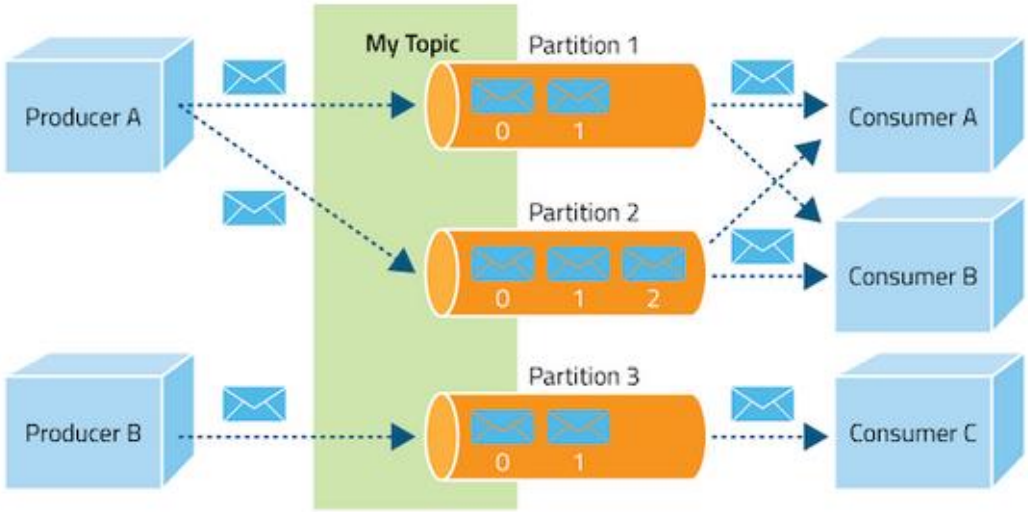
Docker Swarm คือซอฟต์แวร์ที่นำเซอร์วิสมารวมกันเพื่อให้เกิดการคลัสเตอร์

Docker Compose เป็นซอฟต์แวร์ที่ใช้บอกหน้าที่และการใช้งานของแต่ละคอนเทนเนอร์

2.1.5. อาปาเช่ คาฟคา (Apache Kafka)

Apache Kafka เป็นส่วนหนึ่งของระบบนิเวศน์ของ Hadoop ซึ่ง Apache Kafka เป็นบริการบันทึกข้อมูลแบบกระจายที่ทำหน้าที่คล้ายกับระบบการส่งข้อความแบบ publish / subscribe แต่มีความสามารถในการแบ่งส่วนการจำลองแบบและความผิดพลาดได้ดีขึ้น ส่วนใหญ่นิยม สำหรับการเก็บบันทึกและการประมวลผลแบบสตรีมมักใช้ ควบคู่กับ Apache Hadoop, Apache Storm และ Spark Streaming

ล็อก (Log) สามารถเก็บได้ง่าย เนื่องจากรายการบันทึกถูกเรียงด้วยเวลา เป็นพร็อกซีที่สะดวกสำหรับข้อมูลประเภทเวลาต่อเนื่อง (Time series) บันทึกสามารถเรียงลำดับเวลา ดังรูปที่ 12



รูปที่ 12 สถาปัตยกรรม Kafka [8]

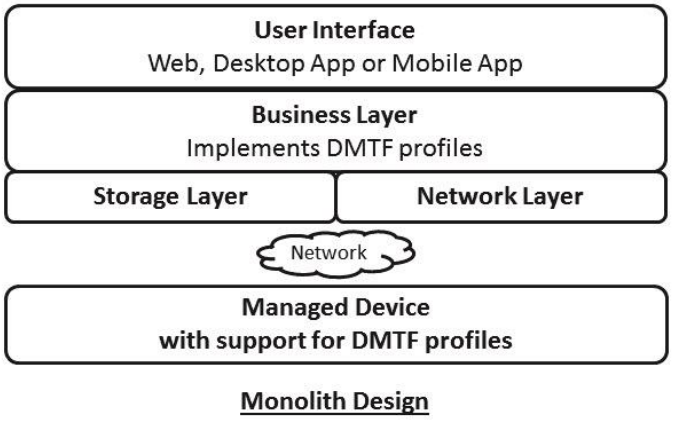
1. การส่งข้อความแบบถาวรด้วย 0 (1) เนื่องจากโครงสร้างพื้นฐานของดิสก์ที่ให้ประสิทธิภาพการทำงานของเวลาที่คงที่แม้จะมีข้อมูลระดับเทราไบต์ก็ตาม
2. มีอัตราการส่งข้อมูลสูงรองรับข้อความนับร้อยนับพันข้อความต่อวินาทีแม้จะมีฮาร์ดแวร์เพียงเล็กน้อยก็ตาม
3. สนับสนุนการแบ่งพาร์ติชันข้อความผ่านเซิร์ฟเวอร์ Kafka อย่างชัดเจนและกระจายการบริโภคผ่านกลุ่มเครื่องอุปโภคบริโภคขณะที่ยังคงรักษาความหมายของคำสั่งต่อพาร์ติชัน
4. รองรับการโหลดข้อมูลแบบขนานไปยัง Hadoop

2.2. งานวิจัยที่เกี่ยวข้อง

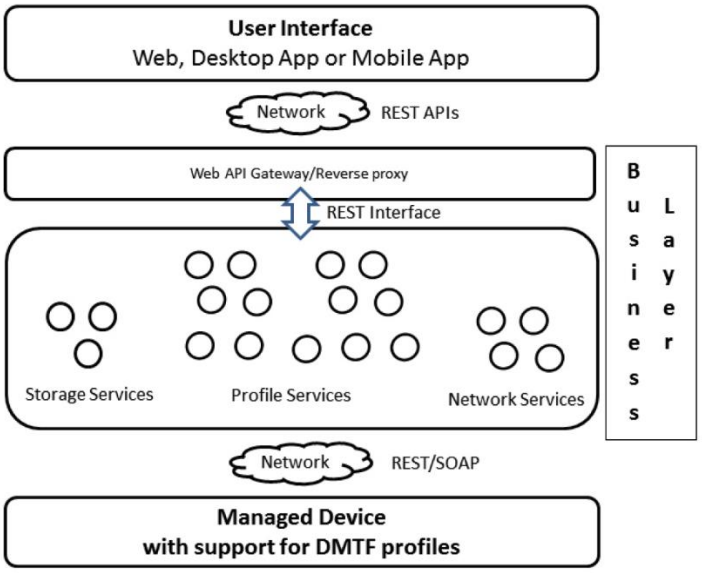
2.2.1. Scalable microservice based architecture for enabling DMTF profiles

ปี ค.ศ. 2015 Malavalli และ Sathappan [9] ได้ทำการทดสอบการขยาย Microservice โดยอธิบายถึงการออกแบบโมโนลิทิก (รูปที่ 13) ซึ่งแบ่งระบบเป็นชั้น (layer) ประกอบด้วย ชั้นส่วนต่อประสานผู้ใช้งาน (User Interface layer) ชั้นตรรกะทางธุรกิจ (Business logic layer) ชั้นจัดเก็บข้อมูล (Storage layer) และชั้นเครือข่าย (Network Layer) และพบว่าการพัฒนาซอฟต์แวร์โดยไม่ให้กระทบกับผู้ใช้งานนั้นเป็นไปได้ยาก เพื่อแก้ไขปัญหาดังกล่าว จึงได้นำเสนอการออกแบบระบบ

ด้วยสถาปัตยกรรมไมโครเซอร์วิส โดยเปลี่ยนแปลงให้อยู่ในรูปแบบไมโครเซอร์วิสในชั้นของตรรกะทางธุรกิจดัง รูปที่ 14



รูปที่ 13 สถาปัตยกรรมการออกแบบโมโนลิทิกสำหรับ DMTF profiles [9]

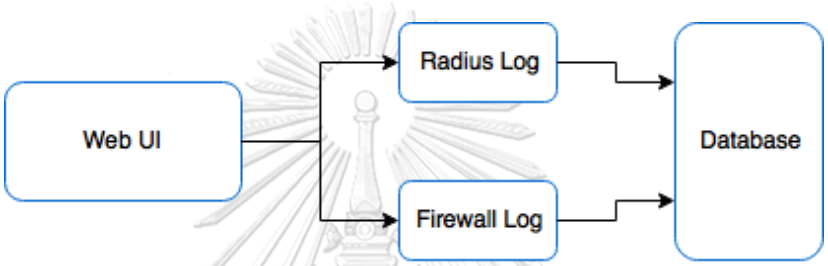


รูปที่ 14 สถาปัตยกรรมการออกแบบไมโครเซอร์วิสสำหรับ DMTF profiles [9]

บทที่ 3
แนวคิดและวิธีวิจัย

3.1 ภาพรวมแนวคิดงานวิจัย

งานวิจัยนี้ได้ประยุกต์ใช้สถาปัตยกรรมไมโครเซอร์วิสมาปรับใช้กับการค้นคืน ให้กับระบบจัดการข้อมูลจราจรเครือข่าย โดยมุ่งเน้นการเพิ่มความสามารถการขยายทั้ง 3 แกน จากเดิมที่ระบบได้ถูกออกแบบด้วยสถาปัตยกรรมโมโนลิทิก [10] ดังแสดงในรูปที่ 15

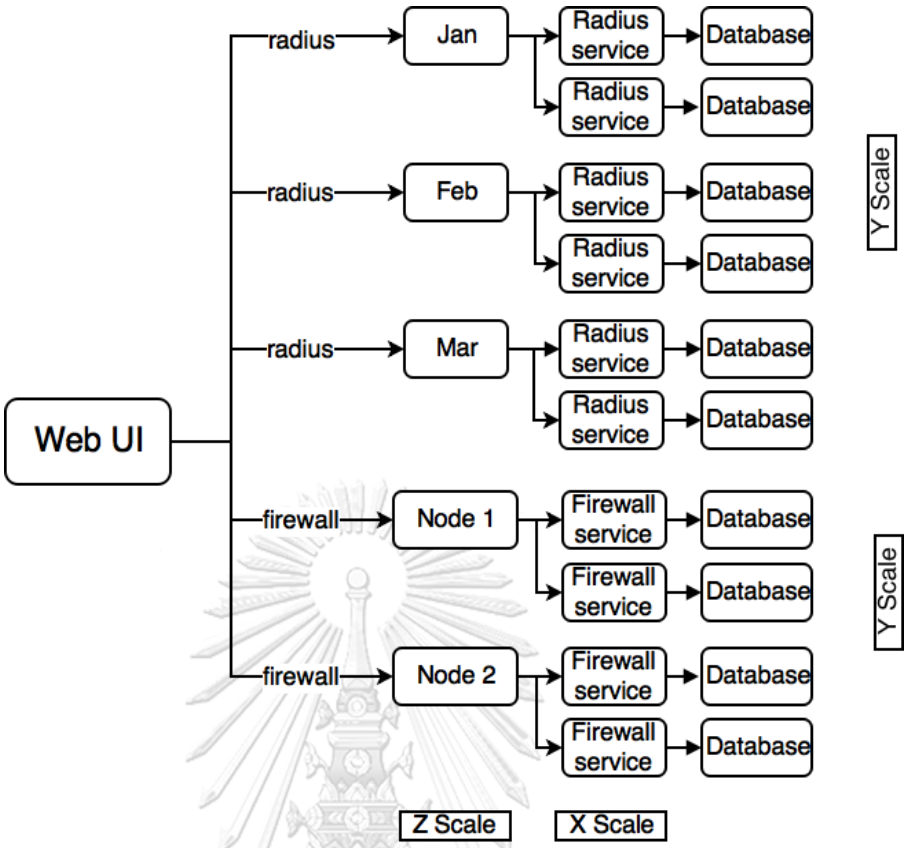


รูปที่ 15 การออกแบบระบบบันทึกจราจรเครือข่ายเดิมด้วยสถาปัตยกรรมโมโนลิทิก

เรเดียสล็อก คือ บันทึกที่บ่งบอกถึงสถานการณ์ล็อกอิน และ ล็อกเอาต์ สู่ระบบของผู้ใช้งาน โดยถูกจัดเก็บเข้าสู่ฐานข้อมูล ส่วน ไฟร์วอลล์ล็อก คือ บันทึกที่แสดงถึงพฤติกรรมการเข้าถึงของไอพีแอดเดรส ซึ่งสามารถบ่งบอกได้ว่าไอพีแอดเดรสใดออกไปยังไอพีแอดเดรสใด โดยผ่านพอร์ตอะไร

สำหรับระบบโมโนลิทิก หากเมื่อเครื่องเซิร์ฟเวอร์ไม่สามารถให้บริการได้ อาจเนื่องมาจากฮาร์ดแวร์หรือซอฟต์แวร์ก็ตาม จะส่งผลให้ไม่สามารถรับล็อกและไม่สามารถค้นคืนข้อมูลใดๆได้ อีกทั้งระบบไม่ได้ถูกออกแบบให้สามารถขยายตัว ได้จึงมีข้อจำกัดในการขยายตัว เมื่อมีปริมาณข้อมูลเพิ่มขึ้นในอนาคตรวมถึงระยะเวลาในการค้นคืนก็จะใช้เวลานานขึ้นเมื่อมีปริมาณข้อมูลมากขึ้น

การออกแบบระบบดังกล่าวด้วยสถาปัตยกรรมไมโครเซอร์วิส เพื่อให้สามารถรองรับการขยายทั้ง 3 แกน แสดงดังรูปที่ 16



รูปที่ 16 การออกแบบระบบด้วยสถาปัตยกรรมไมโครเซอร์วิสที่รองรับการขยายทั้ง3แกน

3.1.1. ด้านแกน X แต่ละเซอร์วิสจะได้รับการ Balance เพื่อกระจายการทำงานในลักษณะ Round-Robin

3.1.2.ด้านแกน Y จะทำการแยกเซอร์วิสออกเป็น 2 ส่วน ประกอบด้วย

3.1.2.1 Radius Service คือ เซอร์วิสที่มุ่งเน้นการให้บริการค้นคืนข้อมูลการล็อกอิน/ล็อกเอ้าต์ของผู้ใช้งาน และมีฟังก์ชันคือ

3.1.2.1.1 เมื่อสอบถามช่วงเวลาใดเวลาหนึ่งโดยแจ้ง IP ให้ทราบ ต้องสามารถบอกได้ว่า ณ เวลาช่วงนั้น คือใคร (User)

3.1.2.1.2 เมื่อสอบถามเวลาที่ระบุได้พร้อมกับ IP สามารถบอกได้ว่าคนที่ใช้ IP ดังกล่าว ได้ทำการ login ล่าสุดเมื่อเวลาใด และคือใคร

3.1.2.2 Firewall Service คือ เซอร์วิสที่มุ่งเน้นการให้บริการค้นคืนข้อมูลกิจกรรมที่ IP ต้นทางใด ๆ ได้ติดต่อไปยัง IP ปลายทางใด ๆ ผ่านโปรโตคอลใด และมีฟังก์ชันคือ

3.1.2.2.1 เมื่อสอบถามช่วงเวลาใดเวลาหนึ่งโดยแจ้ง IP ต้นทางต้องสามารถบอกได้ว่าไปยัง IP ปลายทางใดบ้างในช่วงเวลานั้นผ่านโปรโตคอลอะไร

3.1.2.2.2 เมื่อสอบถามช่วงเวลาใดเวลาหนึ่งโดยแจ้ง IP ปลายทาง ต้องสามารถบอกได้ว่า มีต้นทางใดบ้างที่ติดต่อกับ IP ปลายทางนี้บ้างผ่านโปรโตคอลอะไร

3.1.3.ด้านแกน Z คือ การแบ่งการทำงานของเซอร์วิสตามเกณฑ์ที่กำหนด ในที่นี้สามารถจำแนกจากแกน Y เป็น 2 ประเภท คือ Radius Service และ Firewall service สามารถแยกย่อยได้อีก คือ

3.1.3.1 Radius Service ทำการแบ่งแยกด้วยช่วงเวลา โดยแบ่งเป็น 3 กลุ่มตามรายเดือน

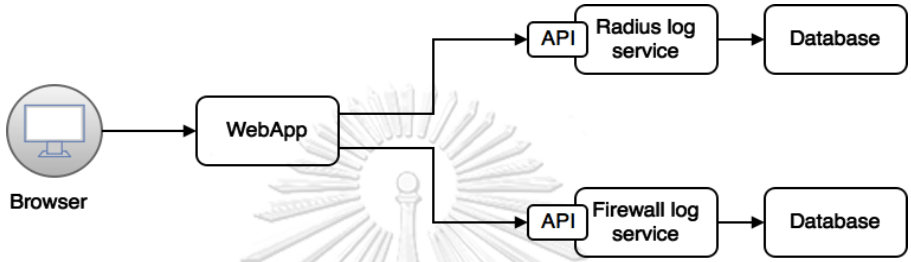
3.1.3.2 Firewall Service ทำการแบ่งแยกด้วย Node หรือจำนวนของอุปกรณ์ เช่น Firewall ปัจจุบันมี 2 อุปกรณ์ จึงทำการแยกเป็น 2 เซอร์วิสย่อย

3.1.4.เว็บแอปพลิเคชัน ถูกออกแบบให้รองรับการค้นคืนแม้ Firewall และ Radius จะมีการเพิ่มจำนวนก็ตาม

บทที่ 4
การพัฒนาระบบ

4.1. สถาปัตยกรรมระบบ

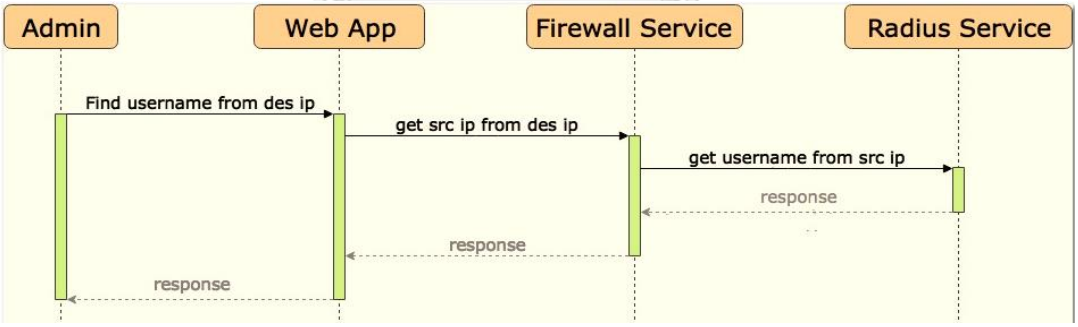
สถาปัตยกรรมไมโครเซอร์วิสได้ถูกเลือกใช้เพื่อให้จัดการระหว่างเรเดียสล็อกและไฟร์วอลล์ล็อกได้โดยง่าย โดยเซอร์วิสเรเดียสและไฟร์วอลล์ต่างก็มีเอพีไอของตนเอง ดังรูปที่ 17



รูปที่ 17 การค้นคืนข้อมูลล็อกของเรเดียสและไฟร์วอลล์ด้วยไมโครเซอร์วิส

4.1.1. ความถูกต้องตรงกันข้อมูล (Data consistency)

รูปแบบการค้นคืนจะเน้นความถูกต้องตรงกันของข้อมูล ซึ่งหากการค้นคืนล้มเหลวที่เซอร์วิสใดจะมีการแจ้งเตือนให้กับเว็บแอปเพื่อทำการยกเลิกงานนั้นๆ ดังรูปที่ 18

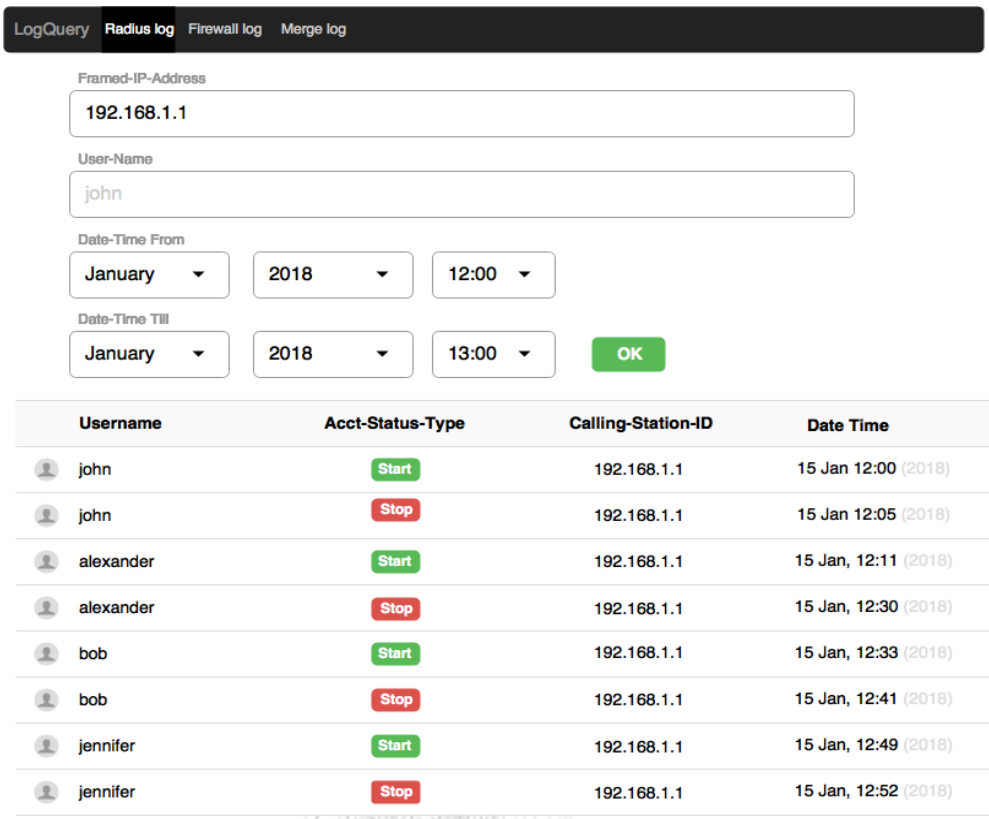


รูปที่ 18 กระบวนการควบคุมความถูกต้องตรงกันของข้อมูล

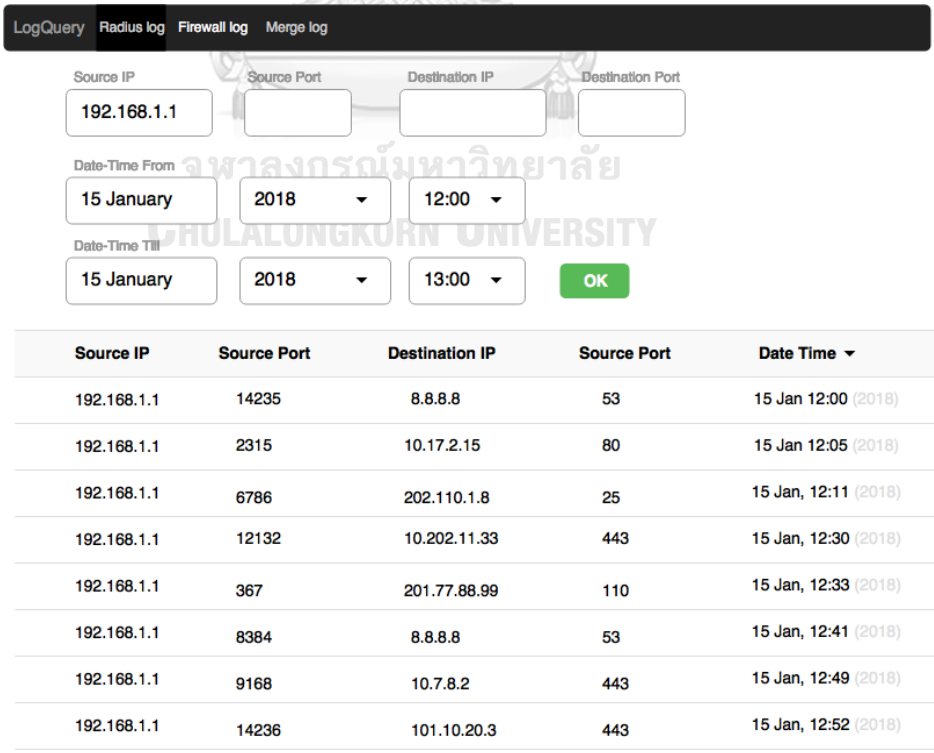
4.1.2. ส่วนต่อประสานผู้ใช้

ส่วนต่อประสานผู้ใช้ประกอบได้ด้วยสามส่วนหลัก คือ

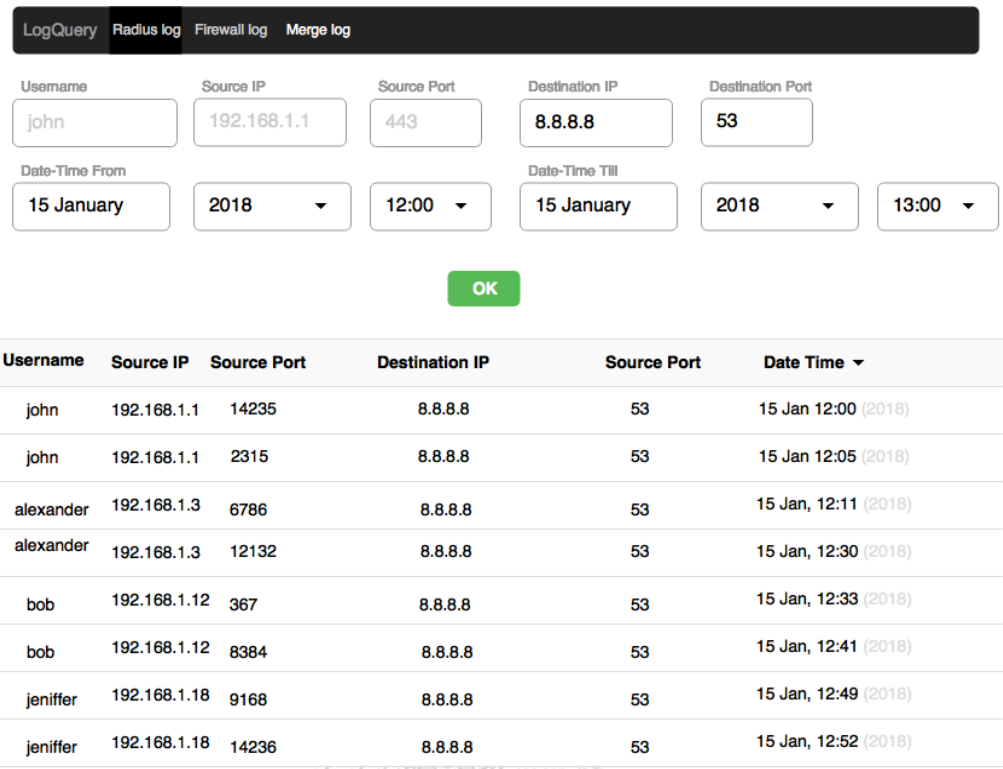
1. หน้าค้นคืนล็อกประเภทเรเดียส ดังแสดงในรูปที่ 19
2. หน้าค้นคืนล็อกประเภทไฟร์วอลล์ ดังแสดงในรูปที่ 20
3. หน้าค้นคืนรวม ดังแสดงในรูปที่ 21



รูปที่ 19 หน้าค้นคืนล็อกประเภทเรเดียส



รูปที่ 20 หน้าค้นคืนล็อกประเภทไฟร์วอลล์



รูปที่ 21 หน้าค้นคืนรวม

4.1.3. โครงสร้างฐานข้อมูล

การออกแบบโครงสร้างฐานข้อมูลสำหรับ MongoDBของเรเดียส (Radius) และไฟร์วอล์ (Firewall) มีความแต่กต่างกันดัง รูปที่ 22 และ รูปที่ 23


```
{
        "_id" : ObjectId("5b1773cd72514e5e9ae0d9ef"),
        "timestamp" : Timestamp(1528263629000, 1),
        "user" : "chakrit",
        "frame_ip_address" : "192.168.1.106",
        "acct_status_type" : "start"
}
```


รูปที่ 22 โครงสร้างฐานข้อมูลเรเดียสล็อก


```
{
        "_id" : ObjectId("5b17749f72514e5e9ae0d9f0"),
        "timestamp" : Timestamp(1528263839000, 1),
        "srcip" : "192.168.1.6",
        "sport" : "3599",
        "dstip" : "122.155.12.44",
        "dstport" : "443"
}
```


รูปที่ 23 โครงสร้างฐานข้อมูลไฟร์วอล์ล็อก

4.2. สภาพแวดล้อมและเครื่องมือที่ใช้พัฒนา

สภาพแวดล้อมและเครื่องมือที่ใช้พัฒนาระบบประกอบด้วย รายการฮาร์ดแวร์ และซอฟต์แวร์ ดังต่อไปนี้

4.2.1. สภาพแวดล้อม

4.2.1.1 หน่วยประมวลผลอินเทล คอร์ ไอ 7 2.7 กิกะเฮิร์ต (CPU Intel Core I7 2.7 Ghz)

4.2.1.2 หน่วยความจำ 16 กิกะไบต์ (16 GB RAM)

4.2.1.3 ฮาร์ดดิสก์ความจุ 500 กิกะไบต์ (500GB HDD)

4.2.1.4 ระบบปฏิบัติการ macOS High Sierra

4.2.2. เครื่องมือที่ใช้ในการพัฒนา

4.2.2.1 VI text editor 7.4.576

4.2.2.2 Atom text editor 1.14.3

4.2.2.3 PHP 7.1.18

4.2.2.4 MongoDB v3.6.5

4.2.2.5 Docker 18.05.0-ce

4.2.2.6 Syslog-ng 3.16.1

4.2.2.7 Kafka 2.12-1.1.0

4.3. การออกแบบสถาปัตยกรรม

งานวิจัยนี้มุ่งเน้นที่จะนำเสนอวิธีการออกแบบสถาปัตยกรรมให้สามารถขยายตัวได้ทั้งขาเข้าซึ่งเริ่มจากการรับ log ให้สามารถขยายตัวได้โดยใช้โมเดลการขยายตัวแบบลูกบาศก์ ซึ่งมีประโยชน์มากในการออกแบบในงานวิจัยนี้ อันประกอบด้วย การขยายตัวแบบแกน การขยายตัวแบบลูกบาศก์ ได้กล่าวถึง Cloning หมายถึง คุณลักษณะของสิ่งที่จะ Clone จะต้องมีความเหมือนกัน จึงเรียกว่าการ Scale แบบแกน X ในมุมมองของการจัดเก็บล็อกได้เลือกใช้การ Scale แบบแกน X ด้วย Docker เนื่องจาก Docker มีคุณสมบัติการขยายตัวในลักษณะ Cloning ได้

4.3.1.การออกแบบการนำเข้าด้วย Syslog-ng

ทำหน้าที่รับ log ด้วย port udp 514 และ tcp 514 เพื่อส่งไปยังแฟ้มโครงแบบคาฟคา (Kafka Configuration file) แสดงดังรูปที่ 24

```
source s_sys {
  udp(ip(0.0.0.0) port(514));
  tcp(ip(0.0.0.0) port(514));
};

destination d_kafka{
  program("/var/www/html/kafka -broker kafka1 -topic syslog");
};

log { source(s_sys); destination(d_kafka); };
```

รูปที่ 24 โครงแบบของ syslog-ng เพื่อส่งไปยัง Kafka

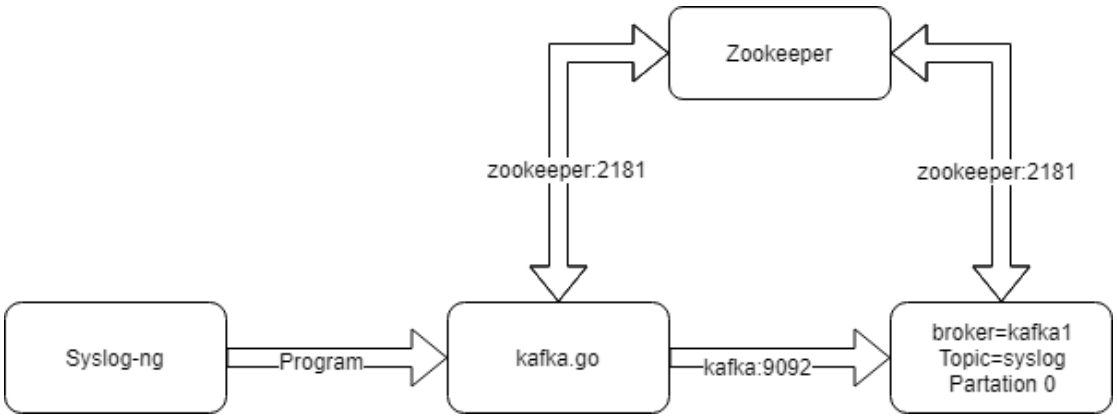
โครงแบบ (Configuration) ของ syslog-ng สามารถอธิบายได้ดังตารางที่ 3

ตารางที่ 3 อธิบายรายละเอียดคุณสมบัติที่ระบุใน syslog-ng

หัวข้อ	**รายละเอียด**
s_sys	ชื่อของ source
d_kafka	ชื่อของ target ที่ต้องการส่ง
Program	ชื่อคำสั่งของ syslog-ng ใช้ในการทำ data pipeline
/var/www/html/kafka	คือโปรแกรมเขียนด้วยภาษาโก(Golang) ทำหน้าที่รับข้อมูลในรูปแบบ Standard input (stdin) เพื่อส่งไปยัง Kafka brokers โดย script Golang สามารถแสดงในรูปที่
-Broker kafka1	ชื่อ hostname kafka1 ในที่นี่คือชื่อ service ของ docker ที่ถูกประกาศเพื่อใช้ swarm ในการกระจายการทำงาน
-topic Syslog	ชื่อ topic ซึ่งถูกสร้างโดย Kafka

4.3.2.การออกแบบ Script Kafka.go

จุดประสงค์เขียนขึ้นเพื่อนำส่ง data จาก syslog ไปยัง Kafka โดยมีภาพรวมขั้นตอนการทำงาน ดังรูปที่ 25



รูปที่ 25 อธิบายการทำงานของ kafka.go

การทำงานของ kafka.go ถูกเขียนและแปลโดยภาษา golang ชื่อไฟล์ Kafka เพื่อประสิทธิภาพการ stream data ที่เร็วกว่าภาษาในรูปแบบ interpreter มีหลักการทำงานคือ เริ่มจากการสอบถามไปยัง zookeeper เพื่อสอบถามถึงจำนวน broker ของ Kafka ว่ามีกี่เครื่อง หลังได้จำนวนเครื่องจาก zookeeper แล้วจะทำการเชื่อมต่อไปยัง Kafka broker ทุกเครื่องเพื่อเตรียมส่งข้อมูล เมื่อ syslog-ng ส่งข้อมูลเข้ามาในรูปแบบท่อส่ง (data pipeline) จะทำการรับข้อมูลและส่งไปยัง broker แต่ละ broker ดังรูปที่ 26 แสดงการทำงานของโปรแกรม Kafka และ รูปที่ 27 คือผลจากดึง Kafka มาใช้งาน script kafka.go ได้ถูกนำแสดงในภาคผนวก

```
root@8def6ad50c9e:/var/www/html# cat /tmp/test.log
Test send to kafka line 1
Test send to kafka line 2
root@8def6ad50c9e:/var/www/html# cat /tmp/test.log|./kafka -broker kafka1 -topic syslog
2018/06/21 06:10:40 Connected to 10.0.1.16:2181
2018/06/21 06:10:40 Authenticated: id=100240410049380398, timeout=4000
2018/06/21 06:10:40 Re-submitting `0` credentials after reconnect
2018/06/21 06:10:40 Connected to 10.0.1.16:2181
2018/06/21 06:10:40 Authenticated: id=100240410049380399, timeout=4000
2018/06/21 06:10:40 Re-submitting `0` credentials after reconnect
Connect brokers kafka1
Connect partition [0]
2018/06/21 06:10:40 Connected to 10.0.1.16:2181
2018/06/21 06:10:40 Authenticated: id=100240410049380400, timeout=4000
2018/06/21 06:10:40 Re-submitting `0` credentials after reconnect
Update max partition: 1
Delivered message to syslog[0]@1
Delivered message to syslog[0]@2
root@8def6ad50c9e:/var/www/html#
```

รูปที่ 26 การทดสอบเรียกใช้งาน ./kafka ในลักษณะ data pipeline

```
root@8cf3c3d56abb:/var/www/html# php roundrobin_get.php 0
Add broker d19038f9648e:9092 success.
Add broker e4a4309b305e:9092 success.
Add broker ada3982a41fe:9092 success.
0.11109304428101 sec.
0 msg.
Test send to kafka line 1
Test send to kafka line 2
0.10138416290283 sec.
2 msg.
```

รูปที่ 27 ผลลัพธ์จากการรับข้อมูลจาก kafka

4.3.3.การออกแบบ Kafka

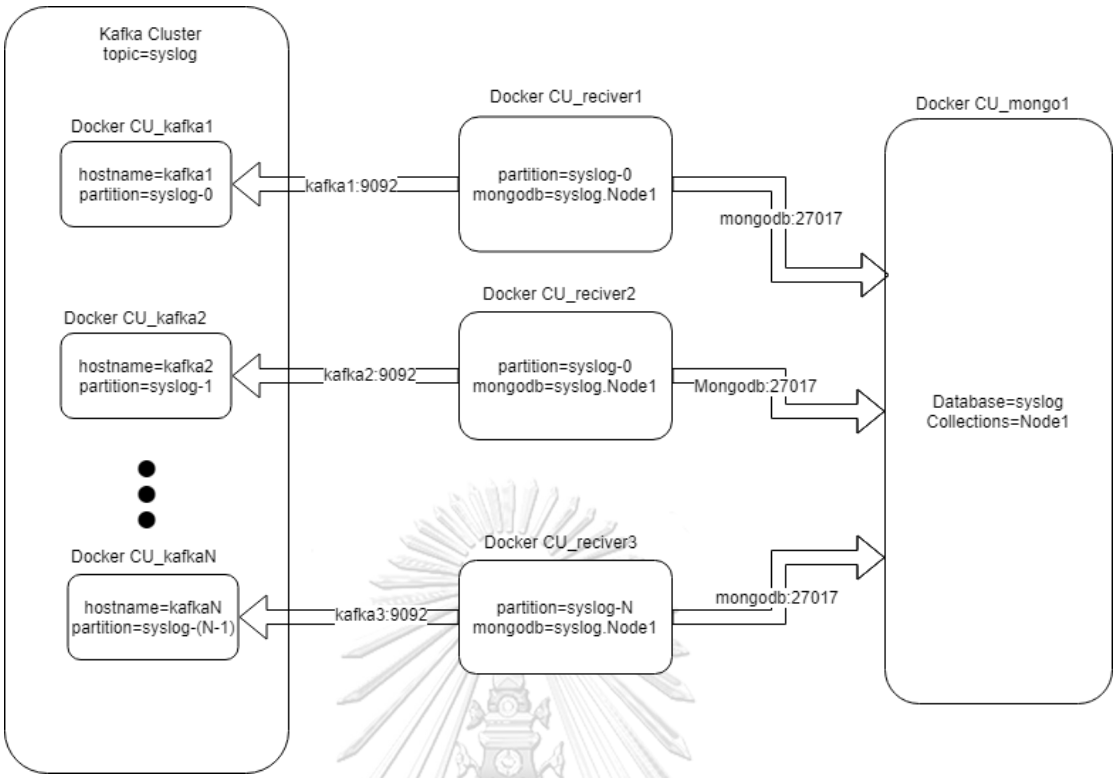
จุดประสงค์เพื่อทำหน้าที่รับ log จาก syslog-ng และจัดการเรื่องแถวคอย (Queue) ทำงานในลักษณะ Publish/Subscribe โดยรับจาก syslog-ng และชุดคำสั่ง kafka.go ในงานวิจัยนี้ได้ออกแบบไว้ ดังตารางที่ 4

ตารางที่ 4 อธิบายค่าพารามิเตอร์สำหรับ Kafka ในแต่ละเครื่อง

พารามิเตอร์	**ค่าที่ถูกกำหนด**
Broker Name,Hostname	Kafka1...KafkaN
Topic	Syslog
Partition	Syslog-0...SyslogN

4.3.4.การออกแบบมอนโกดีบี (MongoDB)

MongoDB ถูกออกแบบในลักษณะที่ implement โดยจัดให้มี Receiver ขึ้นเพื่อดึงข้อมูลจาก Kafka และส่งต่อไปยัง MongoDB ซึ่งใช้วิธีการส่งแบบท่อส่ง (pipeline)

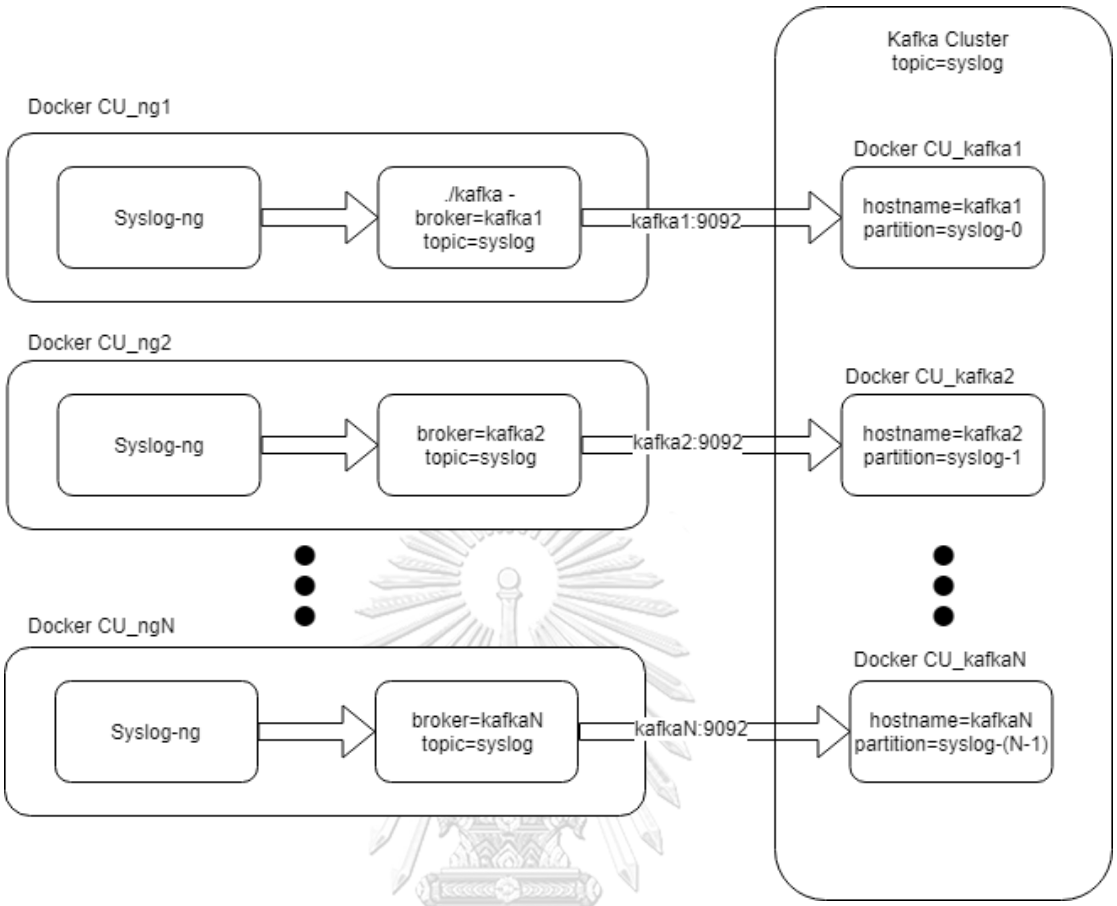


รูปที่ 28 กระบวนการดึงข้อมูลล็อกจาก Kafka ส่งต่อไปยัง MongoDB ด้วย receiver แบบท่อส่ง

จาก รูปที่ 28 Docker CU_reciever1 ทำหน้าที่อ่านข้อมูลจาก Kafka และส่งต่อไปยัง MongoDB โดยใช้ script php เขียนในรูปแบบท่อส่ง ตั้งชื่อ script ว่า kk2mongo.php

4.3.5.ความสามารถการขยายตัว (Scalibility)

การขยายตัวใช้ความสามารถของ docker ในการจัดการ เนื่องจากมีความยืดหยุ่นในการขยายตัวและมีความยืดหยุ่นในการกำหนดค่าพารามิเตอร์ต่างๆ ดังแสดงในรูปที่ 29



รูปที่ 29 สถาปัตยกรรมการขยายตัวของระบบนำเข้าล็อก

4.3.6. สถาปัตยกรรม Docker

จากการออกแบบเพื่อให้สามารถขยายตัวได้นั้น จึงได้ทำการทดลองโดยใช้ Docker Container ช่วยในการจำลองระบบ โดยมีรายละเอียดดังรูปที่ 30 ซึ่งประกอบไปด้วยคอนเทนเนอร์ต่างๆ ดังอธิบายในตารางที่ 5

รูปที่ 30 การจำลองด้วย Docker container

ตารางที่ 5 รายชื่อคอนเทนเนอร์ทั้งหมดในระบบ

Container	Description
CU_ng	syslog-ng และภายในประกอบไปด้วย php,go และ Kafka library
CU_kafka	Kafka Server
CU_zookeeper	Zookeeper Server ใช้สำหรับเก็บ configuration ของ Kafka
CU_reciever	php scripts ถูกจัดทำไว้ในคอนเทนเนอร์ เพื่อรับจาก Kafka และส่งต่อไปยัง MongoDB
CU_mongodb	ฐานข้อมูลที่ใช้สำหรับจัดเก็บ Log
CU_www	webserver ถูกสร้างโดย ngnix
CU_php	php server เชื่อมต่อกับ webserver ผ่าน port 9000

4.3.7. การออกแบบ CU_ng

ภายในประกอบด้วย syslog-ng verions 3.15 ถูก build โดย balabit [11] สำหรับงานวิจัยนี้ได้ทำการเพิ่ม library สำหรับเชื่อมต่อกับ Kafka และ ภาษา Golang ใช้สำหรับเขียนและคอมไพล์ชื่อไฟล์ kafka.go ชุดคำสั่งสามารถตรวจสอบได้จากภาคผนวก ในส่วนของ Docker file และสคริปต์ที่เกี่ยวข้อง แสดงดังรูปที่ 31 รูปที่ 32 และ รูปที่ 33

```
1  FROM php:7.1-fpm-jessie
2  MAINTAINER Andras Mitzki <andras.mitzki@balabit.com>
3  MAINTAINER Chakrit Phain <chakrit@softnix.co.th>
4
5  RUN apt-get update -qq && apt-get install -y wget gnupg2
6  RUN wget -qO - https://packages.confluent.io/deb/4.1/archive.key | apt-key add -
7  RUN echo "deb http://packages.confluent.io/deb/4.1 stable main"| tee --append
   /etc/apt/sources.list.d/confluent.io.list
8  RUN wget -qO -
   https://download.opensuse.org/repositories/home:/laszlo_budai:/syslog-ng/Debian_8.
   0/Release.key | apt-key add -
9  RUN echo "deb
   http://download.opensuse.org/repositories/home:/laszlo_budai:/syslog-ng/Debian_8.0
   ./" | tee --append /etc/apt/sources.list.d/syslog-ng-obs.list
10 RUN apt-get update -qq && apt-get install -y syslog-ng && apt-get install -y
   mongodb-clients
11
12 ADD openjdk-libjvm.conf /etc/ld.so.conf.d/openjdk-libjvm.conf
13 RUN ldconfig
14
15 EXPOSE 514/udp
16 EXPOSE 601/tcp
17 EXPOSE 6514/tcp
18
19 RUN apt -y  install iputils-ping && apt -y  install telnet && apt -y  install
   net-tools && apt -y install librdkafka-dev && apt -y install librdkafka1 && apt -y
   install libzookeeper-mt2 && apt -y install vim
20 ADD www.conf /usr/local/etc/php-fpm.d/
21 ADD rdkafka.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
22 ADD zookeeper.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
23 ADD stomp.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
24 RUN docker-php-ext-enable stomp
25 RUN docker-php-ext-enable rdkafka
26 RUN docker-php-ext-enable zookeeper
27 RUN docker-php-ext-install sockets
28 ADD ./html/ /var/www/html/
29 ADD ./goroot.tar.gz /opt/
30 ADD ./golang/src/github.com/ /opt/goroot/src/github.com/
31 RUN mkdir /opt/kafka/
32 RUN mkdir /opt/kafka/libs/
33 ADD ./libs/ /opt/kafka/libs/
34 RUN echo 'export PATH="/opt/goroot/bin/:$PATH"' >> /etc/profile
35 ENTRYPOINT ["/usr/sbin/syslog-ng", "-F"]
```

รูปที่ 31 สคริปต์ Dockerfile เพื่อทำการ build CU_ng

```yaml
version: '3.5'
configs:
  ng1_conf:
    file: ./ng1/app1.conf
services:
  ng1:
    image: krmonline/syslog-ng-softnix:0.8dev
    networks:
      - stomp
    configs:
      - source: ng1_conf
        target: /etc/syslog-ng/conf.d/app1.conf
    volumes:
      - type: bind
        source: ./ng1/html
        target: /var/www/html
```

รูปที่ 32 การเรียกใช้งาน CU_ng จาก docker-composer.yml

```
source s_sys {
  udp(ip(0.0.0.0) port(514));
  tcp(ip(0.0.0.0) port(514));
};

destination d_php{
  program("/var/www/html/kafka -broker kafka1 -topic syslog");
};

log { source(s_sys); destination(d_php); };

```


รูปที่ 33 App1.conf ใช้สำหรับ listen port 514 udp และ 514 tcp

4.3.8.การออกแบบ CU_kafka และ Zookeeper

ในการเรียกใช้งานจะใช้ควบคู่กับคอนเทนเนอร์ของ zookeeper ร่วมด้วยเสมอ เนื่องจาก zookeeper ทำหน้าที่เก็บโครงแบบให้กับ CU_kafka และกำหนดให้ default ของจำนวน partition มีค่าเท่ากับ 1 ในการเพิ่มจำนวน partition สามารถทำได้โดยใช้คำสั่งที่คอนเทนเนอร์ของ CU_kafka เครื่องใดเครื่องหนึ่งโดย Docker-compose ไฟล์ แสดงดังรูปที่ 34

“/opt/kafka/bin/kafka-topics.sh --zookeeper zookeeper:2181 --alter --partitions=3 --topic syslog”

```
version: '3.2'
services:
  zookeeper:
    image: wurstmeister/zookeeper
    networks:
      - stomp
    ports:
      - "2181:2181"
  kafka:
    image: wurstmeister/kafka:latest
    deploy:
      replicas: 1
    networks:
      - stomp
    ports:
      - target: 9094
        published: 9094
        protocol: tcp
        mode: host
    environment:
      HOSTNAME_COMMAND: "hostname"
      BROKER_ID_COMMAND: "date +%s|cut -c8-10"
      KAFKA_ZOOKEEPER_CONNECT: zookeeper:2181
      KAFKA_LISTENER_SECURITY_PROTOCOL_MAP: INSIDE:PLAINTEXT,OUTSIDE:PLAINTEXT
      KAFKA_ADVERTISED_LISTENERS: INSIDE://:9092,OUTSIDE://_{HOSTNAME_COMMAND}
:9094
      KAFKA_LISTENERS: INSIDE://:9092,OUTSIDE://:9094
      KAFKA_INTER_BROKER_LISTENER_NAME: INSIDE
      KAFKA_CREATE_TOPICS: "syslog:1:1"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
networks:
  stomp:
```

รูปที่ 34 การเรียกใช้งาน CU_kafka

4.3.9.การออกแบบ CU_receiver

CU_receiver ทำหน้าที่สำหรับดึงข้อมูลจาก Kafka เพื่อส่งไปยังฐานข้อมูล MongoDB โดยข้อมูลจาก Kafka จะอยู่ในลักษณะข้อมูลดิบและจะถูกส่งไปยัง script เพื่อแปลงให้อยู่ในรูป Json format หลังจากนั้นจะถูกส่งไปยัง MongoDB ผ่าน command mongoimport โดยใช้ลักษณะท่อส่ง ในส่วนของ image file สามารถใช้ image เดียวกันกับ CU_ng เนื่องจากใช้ php script ในการทำงาน

ตัวอย่างสคริปต์ที่ใช้สำหรับส่งไปยังฐานข้อมูล MongoDB ดังตารางที่ 6 และชุดคำสั่งของแต่ละสคริปต์แสดงดังรูปที่ 35 ถึง รูปที่ 39 ตามลำดับ

ตารางที่ 6 สคริปต์และคำสั่งที่ถูกใช้งานในคอนเทนเนอร์ CU_reciever

สคริปต์หรือคำสั่ง	**รายละเอียด**
Kk2mongo.php	ใช้สำหรับดึงข้อมูลจาก Kafka และส่งออกด้วย stdout
A102json.php	ใช้สำหรับแปลงข้อมูลที่ได้จาก kk2mongo.php จากข้อมูลดิบให้อยู่ในรูป json format
mongoimport	ใช้สำหรับ import ข้อมูลที่ได้จาก a102json.php และส่งต่อไปยังฐานข้อมูล MongoDB

```
root@7addcc99712d:/var/www/html# php kk2mongo.php | \
> php a102json.php | \
> mongoimport -h mongo1 -d syslog -c node1
connected to: mongo1
```

รูปที่ 35 การใช้งานสคริปต์และคำสั่งแบบท่อส่ง

```yaml
version: '3.5'

services:
  reciever1:
    image: krmonline/syslog-ng-softnix:0.8dev
    networks:
      - kafka_network
      - mongo_network
    volumes:
      - type: bind
        source: ./ng1/html
        target: /var/www/html
networks:
  kafka_network:
  mongo_network:
  bridge:
    external: true

```

รูปที่ 36 การเรียกใช้งาน CU_receiver

```php
<?php
function microtime_float(){
  list($usec,$sec) = explode(" ",microtime());
  return $usec + $sec;
}

function getbroker($zookeeper="zookeeper:2181"){
  $zk = new zookeeper($zookeeper);
  $path = "/brokers/ids";
  $r = $zk->getchildren($path);
  foreach($r as $v){
    $key = $path."/".$v;
    $value = $zk->get($key);
    $json = json_decode($value);
    $hostname =  $json->endpoints[0];
    if(preg_match("/[A-Z]+:\/\/([0-9a-f]+):([0-9]+)/",$hostname,$arr_result)){
      list($dc,$host,$port) = $arr_result;
      $arr[] = "$host:$port";
    }
  }
  return $arr;
}
$conf = new RdKafka\Conf();

// Set the group id. This is required when storing offsets on the broker
$conf->set('group.id', 'myConsumerGroup');

$rk = new RdKafka\Consumer($conf);
$arr_broker = getbroker();
foreach($arr_broker as $v){
        if($rk->addBrokers($v)){
          //echo "Add broker $v success.\n";
        }
}
//$rk->addBrokers("ff820d365ea1,1669b5f567b9,f6094acd8f95");

$topicConf = new RdKafka\TopicConf();
$topicConf->set('auto.commit.interval.ms', 100);

// Set the offset store method to 'file'
$topicConf->set('offset.store.method', 'file');
$topicConf->set('offset.store.path', sys_get_temp_dir());

// Alternatively, set the offset store method to 'broker'
// $topicConf->set('offset.store.method', 'broker');

// Set where to start consuming messages when there is no initial offset in
// offset store or the desired offset is out of range.
// 'smallest': start from the beginning
$topicConf->set('auto.offset.reset', 'smallest');
```

รูปที่ 37 สคริปต์ kafka2mongo.php ใช้สำหรับดึงล็อกจาก Kafka

```php

$topic = $rk->newTopic("syslog", $topicConf);
$queue = $rk->newQueue();
// Start consuming partition 0
$topic->consumeQueueStart($argv[1], RD_KAFKA_OFFSET_STORED,$queue);
if(isset($argv[2]))
  $topic->consumeQueueStart($argv[2], RD_KAFKA_OFFSET_STORED,$queue);
if(isset($argv[3]))
  $topic->consumeQueueStart($argv[3], RD_KAFKA_OFFSET_STORED,$queue);
$timestamp = microtime_float();
$count = 0;
$reset = false;
while (true) {
    //$message = $topic->consume(0, 120*10000);
    $message = $queue->consume(1000);
    switch ($message->err) {
        case RD_KAFKA_RESP_ERR_NO_ERROR:
            //var_dump($message);
            if($reset){
              $timestamp = microtime_float();
            }
            if($message->payload){
              echo $message->payload."\n";
                    $count++;
                    $reset = false;
            }
            break;
        case RD_KAFKA_RESP_ERR__PARTITION_EOF:
            //echo "No more messages; will wait for more\n";
            $timestampOld = $timestamp;
            $timestamp = microtime_float();
            $diff = $timestamp - $timestampOld;
            //echo $diff." sec.\n";
            //echo $count." msg.\n";
            $reset = true;
            //$count = 0;
            break;
        case RD_KAFKA_RESP_ERR__TIMED_OUT:
            //echo "Timed out\n";
            break;
        default:
            throw new \Exception($message->errstr(), $message->err);
            break;
    }
}

?>

```

รูปที่ 38 สคริปต์ kafka2mongo.php ใช้สำหรับดึงล็อกจาก Kafka (ต่อ)

```
<?php
$stdin = fopen('php://stdin', 'r');
$reg = '/([A-Z][a-z]* +[^ ]* [^ ]*) ([^ ]*).* ([^:]+:[^:]+:[^ ]+) ([0-9\.]+) ([^ ]*) - . \[(TCP|UDP)
([^:]+):([^\ ]+) ([^\:]+):([^ ]+) ([^:]+):([^ ]+) ([^:]+):([^ ]+) "([^\"]+)" .*\]/';
$count = 0;
if(!isset($argv[1])){
        echo "a102json.php [uniq keyid]";
}else{
        $tag = $argv[1];
}

$now = time();
$last_y = date("Y")-1; //fix bug
while(!feof($stdin)){
        $line = fgets($stdin);
        $line = trim($line);
        if(!$line){
                continue;
        }
        $match = preg_match($reg,$line,$r);
        if($match){
                //echo $line."\n";
                //var_dump($r);
                $datetime = $r[1];
                //`echo '$datetime' >> /tmp/debug`;
                $ts = strtotime($datetime);
                //fix bug 'Feb 19 19:29' dont know year
                //change Feb 19 19:29 to 2013-02-19 19:29:00
                if($now < $ts){
                        $date_tmp = date("-m-d H:i:s",$ts);
                        $date_tmp = $last_y.$date_tmp;
                        $ts = strtotime($date_tmp);
                }
                $ts = $ts."000";
                $a10 = $r[2];
                $ntype = $r[6];
                $src = $r[7];
                $srcp = $r[8];
                $dst = $r[11];
                $dstp = $r[12];
                $natip =  $r[13];
                $natp = $r[14];
                $msisdn = $r[15];
                $uri = "";
                $json = '{"a10" : "'.$a10.'", "ntype" : "'.$ntype.'", "srcip1" : "'.$src.'", "srcp" :
"'.$srcp.'", "dst" : "'.$dst.'", "dstp" : "'.$dstp.'", "natip" : "'.$natip.'", "natp" : "'.$natp.'",
"msisdn" : "'.$msisdn.'", "uri":"'.$uri.'" ,"time" : { "$date" : '.$ts.' } }'."\n";
                echo $json;
                $count++;
                //`echo '$json' >> /tmp/debug`;
        }else{
                //`echo 'not match $line' >> /tmp/debug`;
        }
        //die();
}
?>
```


รูปที่ 39 สคริปต์ a102json.php สำหรับแปลงข้อมูลดิบให้อยู่ในรูป json format ด้วยวิธีการท่อส่ง

4.3.10. การออกแบบ CU_php

CU_php ทำหน้าที่จัดการสคริปต์เอพีไอ (API) เพื่อใช้ค้นคืนระบบล็อกและนำเสนอผ่านเว็บเซิร์ฟเวอร์ CU_www1 โดยพีเอชพีเซิร์ฟเวอร์จะเปิดพอร์ต 9000 โดยใช้เอพีไอเฟรมเวิร์กชื่อ สลิม (Slim) มีรหัสต้นฉบับ ดังรูปที่ 40

```php
<?php

use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;
require '../vendor/autoload.php';
require 'config.php';
$app = new \Slim\App(['settings' => $config]);

$app->get('/', function () {
    $name = "/srcip\n/dstip";
    return $name;
});

$app->get('/q', function (Request $request, Response $response, array $args) {
  //var_dump($_GET);
  $mongoip = $this->get('settings')['mongo_server'];

  $response->getBody()->write("Source IP $srcip");
  $manager = new MongoDB\Driver\Manager("mongodb://$mongoip");


  //Query
  //date
  $from = isset($_GET['from'])?strtotime($_GET['from'])*1000:"";
  if($from){
    $to = isset($_GET['to'])?strtotime($_GET['to'])*1000:$from+300*1000;
    $filter["time"] = ['$gte' => new \MongoDB\BSON\UTCDatetime($from), '$lte' => new \MongoDB\BSON\UTCDatetime($to)];
  }

  //source ip
  $srcip = ISSET($_GET['srcip'])?$_GET['srcip']:"";
  $filter["srcip1"] =  $srcip;

  //var_dump($filter);
  //limit skip
  $limit = isset($_GET["limit"])?$_GET["limit"]:$this->get('settings')['limit'];
  $page = isset($_GET["page"])?$_GET["page"]:"0";
  $skip = $page*$limit;
  $options = ["limit" => $limit,"skip" => $skip];

  $query = new MongoDB\Driver\Query($filter, $options);
  $cursor = $manager->executeQuery('syslog.node1', $query);
  foreach($cursor as $doc){
    //var_dump(new MongoDB\BSON\UTCDateTime(1416445411987));
    //$datetime = $utcdatetime->toDateTime();
    $result[] = $doc;
  }
  return $response->withJson($result);
});
$app->run();
```

รูปที่ 40 รหัสต้นฉบับเอพีไอเพื่อใช้ในการค้นคืนล็อก

4.3.11. การออกแบบ CU_www

CU_www ทำหน้าที่ให้บริการเว็บเอพีไอ ซึ่งจะเชื่อมต่อไปยัง CU_php ผ่านพอร์ต 9000 โดยมีโครงแบบ ดังรูปที่ 41

default.conf 928 Bytes

```
server {
    listen       80;
    server_name  localhost;
    index index.php;
    root /var/www/html/public;
    #charset koi8-r;
    #access_log  /var/log/nginx/host.access.log  main;

    location / {
        try_files $uri /index.php$is_args$args;
    }

    #error_page  404              /404.html;

    # redirect server error pages to the static page /50x.html
    #
    error_page   500 502 503 504  /50x.html;
    location = /50x.html {
        root   /usr/share/nginx/html;
    }

    # proxy the PHP scripts to Apache listening on 127.0.0.1:80
    #
    #location ~ \.php$ {
    #    proxy_pass   http://127.0.0.1;
    #}

    # pass the PHP scripts to FastCGI server listening on 127.0.0.1:9000
    #
    location ~ \.php$ {
        fastcgi_pass   phpfw:9000;
        fastcgi_index  index.php;
              fastcgi_param      SCRIPT_FILENAME $document_root$fastcgi_script_name;
        include        fastcgi_params;
    }

}
```

รูปที่ 41 ไฟล์โครงแบบสำหรับ CU_www

4.3.12. การออกแบบ CU_receiver_rd

CU receiver ทำหน้าที่ส่ง log ของ Radius เพื่อส่งไปยังฐานข้อมูล MongoDB โดยปรกติแล้วรูปแบบล็อกของ Radius จะมีลักษณะแนวตั้ง จึงต้องทำการแปลงจากแนวตั้งให้เป็นแนวนอนโดยใช้สคริปต์ Perl ในลักษณะท่อส่งและแยกขั้นด้วยเครื่องหมาย '@@@' โดยสคริปต์มีรายละเอียดดังรูปที่ 42 และผลลัพธ์จากสคริปต์แสดงดังรูปที่ 43 หลังจากนั้นจะทำการแปลงรูปแบบให้อยู่ในรูปแบบ JSON โดยใช้สคริปต์ php ในลักษณะท่อส่งเช่นเดียวกันสคริปต์สามารถแสดงได้ดังรูปที่ 44 และผลลัพธ์แสดงดังรูปที่ 45 หลังจากนั้นจึงนำส่งยังฐานข้อมูล MongoDB ในลักษณะท่อส่งเช่นเดียวกันด้วยคำสั่ง mongoimport

```perl
#!/usr/bin/perl
my $file = shift @ARGV;
my $ifh;
my $separate = "@@@";
my $is_stdin = 0;
if (defined $file){
    open $ifh, "<", $file or die $!;
} else {
    $ifh = *STDIN;
    $is_stdin++;
}
while (<$ifh>){
    # Process
    $line = $_;
    chomp($line);
    if(!$line ){
        print $text."\n\n";
        $text = "";
    }
    $text .= ($text)?$separate.$line:$line;
}
#close $ifh unless $is_stdin;
close $ifh;
```

รูปที่ 42 สคริปต์สำรับแปลงแนวตั้งเป็นแนวนอน

```
Tue Mar 18 19:00:00 2014@@@     Acct-Status-Type = Interim-Update@@@    Acct-Input-Octets = 11011@@@    Acct-Output-
Octets = 25104@@@       Acct-Input-Packets = 140@@@     Acct-Output-Packets = 104@@@    Acct-Link-Count = 1@@@  Even
t-Timestamp = "Mar 18 2014 19:00:00 ICT"@@@     Acct-Authentic = RADIUS@@@      Acct-Delay-Time = 0@@@  Acct-Session
-Time = 522@@@  Acct-Multi-Session-Id = "9508dfe3"@@@   Calling-Station-Id = "66667811424"@@@   Framed-IP-Address =
22.203.161.221@@@       Acct-Session-Id = "b4d6ce919508dfe3"@@@ NAS-IP-Address = 22.95.56.33@@@ Framed-Protocol = GP
RS-PDP-Context@@@       Called-Station-Id = "hinternet"@@@      User-Name = "true"@@@   NAS-Port-Type = Virtual@@@ S
ervice-Type = Framed-User@@@    NAS-Port = 110001@@@    NAS-Port-Id = "UGW"@@@  3GPP-IMSI = "520002045549948"@@@   3
GPP-Charging-ID = 2500386787@@@ 3GPP-PDP-Type = 0@@@    3GPP-Charging-Gateway-Address = 22.95.13.215@@@ 3GPP-GPRS-Ne
gotiated-QoS-profile = "05-0b911f7196f7fe4401ffff006400"@@@     3GPP-SGSN-Address = 180.214.206.205@@@  3GPP-GGSN-Ad
dress = 180.214.206.145@@@      3GPP-IMSI-MCC-MNC = "520002"@@@ 3GPP-GGSN-MCC-MNC = "520000"@@@ 3GPP-NSAPI = "5"@@@3
GPP-Selection-Mode = "0"@@@     3GPP-Charging-Characteristics = "0800"@@@       3GPP-SGSN-MCC-MNC = "52000"@@@  3GPP
-RAT-Type = 0x01@@@     3GPP-MS-TimeZone = 0x8200@@@    3GPP-Negotiated-DSCP = "\n"@@@  Acct-Unique-Session-Id = "eb
e68d094f943d02"@@@      Timestamp = 1395144000

Tue Mar 18 19:00:00 2014@@@     Acct-Status-Type = Interim-Update@@@    Acct-Input-Octets = 1245081@@@  Acct-Output-
Octets = 32553813@@@    Acct-Input-Packets = 12246@@@   Acct-Output-Packets = 25619@@@  Acct-Link-Count = 1@@@  Even
t-Timestamp = "Mar 18 2014 19:00:00 ICT"@@@     Acct-Authentic = RADIUS@@@      Acct-Delay-Time = 0@@@  Acct-Session
-Time = 5691@@@ Acct-Multi-Session-Id = "54094e0c"@@@   Calling-Station-Id = "66666653887"@@@   Framed-IP-Address =
22.192.206.185@@@       Acct-Session-Id = "b4d6ce9454094e0c"@@@ NAS-IP-Address = 22.95.56.33@@@ Framed-Protocol = GP
RS-PDP-Context@@@       Called-Station-Id = "hinternet"@@@      User-Name = "true"@@@   NAS-Port-Type = Virtual@@@ S
ervice-Type = Framed-User@@@    NAS-Port = 110001@@@    NAS-Port-Id = "UGW"@@@  3GPP-IMSI = "520002042340682"@@@   3
GPP-Charging-ID = 1409895948@@@ 3GPP-PDP-Type = 0@@@    3GPP-Charging-Gateway-Address = 22.95.13.215@@@ 3GPP-GPRS-Ne
gotiated-QoS-profile = "05-0b911f7196f4fe4429ffff004a00"@@@     3GPP-SGSN-Address = 180.214.200.106@@@  3GPP-GGSN-Ad
dress = 180.214.206.148@@@      3GPP-IMSI-MCC-MNC = "520002"@@@ 3GPP-GGSN-MCC-MNC = "520000"@@@ 3GPP-NSAPI = "5"@@@3
GPP-Selection-Mode = "0"@@@     3GPP-Charging-Characteristics = "0800"@@@       3GPP-SGSN-MCC-MNC = "52004"@@@  3GPP
-RAT-Type = 0x01@@@     3GPP-MS-TimeZone = 0x8200@@@    3GPP-Negotiated-DSCP = "\n"@@@  Acct-Unique-Session-Id = "a6
7fe6c9d96ff8f0"@@@      Timestamp = 1395144000
```

รูปที่ 43 ผลลัพธ์จากการรันสคริปต์ในลักษณะท่อส่ง

```
<?php
$separate = "@@@";
$id = isset($argv[1])?$argv[1]:"";

$data = json_decode($argv[2]);
//print_r($data);
$user = $data->user;
$uuid = $data->uuid;
$type_log = $data->type_log;
//die();
$wlog = function($text) {
    if(!$text){
        return false;
    }
    $filename = "/var/log/softnix/indexing.log";
    $time = date("Y-m-d H:i:s");
    $text = $time."\t".$text;
    $text = escapeshellarg($text);
    `echo $text >> $filename`;
};

$stdin = fopen('php://stdin', 'r');
$reset = false;
$text = "";
while (!feof($stdin)) {
    $line = fgets($stdin);
    $line = trim($line);
    if ($line) {
        $text = "";
        $text .= "\"user\":\"$user\",";
        $text .= "\"uuid\":\"$uuid\",";
        $text .= "\"type_log\":\"$type_log\"";
        $arr = explode($separate,$line);
        $tmp = array_shift($arr);
        //var_dump($arr);
        foreach($arr as $v) {
            list($radius_key,$radius_value) = explode("=",$v);
            $radius_key = trim($radius_key);
            $radius_value = trim($radius_value);
            if(preg_match("/\"([^\"]+)\"/",$radius_value,$pregResult)) {
                //var_dump($pregResult);
                $radius_value = $pregResult[1];
                //echo $radius_value." is new value\n";
            }
            if($radius_key && $radius_value){
                $text .= ($text)?",":"";
                if ($radius_key == "3GPP-Negotiated-DSCP") {
                    $radius_value = "";
                }
                $text .= "\"$radius_key\":\"$radius_value\"";
            }
        }
        $json = json_decode("{".$text."}");
        echo "".json_encode($json)."\n";
    }
}
?>
```

รูปที่ 44 สคริปต์แปลงเป็น JSON ในลักษณะท่อส่ง

{"user":"","uuid":"","type_log":"","Acct-Status-Type":"Interim-Update","Acct-Input-Octets":"735229","Acct-Output-Octets":"14726893","Acct-Input-Packets":"9797","Acct-Output-Packets":"11875","Acct-Link-Count":"1","Event-Timestamp":"Mar 18 2014 19:00:00 ICT","Acct-Authentic":"RADIUS","Acct-Session-Time":"4443","Acct-Multi-Session-Id":"8409695c","Calling-Station-Id":"66668791221","Framed-IP-Address":"22.202.92.125","Acct-Session-Id":"b4d6ce938409695c","NAS-IP-Address":"22.95.56.33","Framed-Protocol":"GPRS-PDP-Context","Called-Station-Id":"hinternet","User-Name":"true","NAS-Port-Type":"Virtual","Service-Type":"Framed-User","NAS-Port":"110001","NAS-Port-Id":"UGW","3GPP-IMSI":"520002042481712","3GPP-Charging-ID":"2215209308","3GPP-Charging-Gateway-Address":"22.95.13.215","3GPP-GPRS-Negotiated-QoS-profile":"05-23931f9196f4fe944bffff004a00","3GPP-SGSN-Address":"180.214.200.97","3GPP-GGSN-Address":"180.214.206.147","3GPP-IMSI-MCC-MNC":"520002","3GPP-GGSN-MCC-MNC":"520000","3GPP-NSAPI":"5","3GPP-Charging-Characteristics":"0800","3GPP-SGSN-MCC-MNC":"52004","3GPP-RAT-Type":"0x01","3GPP-MS-TimeZone":"0x8200","3GPP-Negotiated-DSCP":"","Acct-Unique-Session-Id":"721425116cd017d8","Timestamp":"1395144000"}
{"user":"","uuid":"","type_log":"","Acct-Status-Type":"Interim-Update","Acct-Input-Octets":"11011","Acct-Output-Octets":"25104","Acct-Input-Packets":"140","Acct-Output-Packets":"104","Acct-Link-Count":"1","Event-Timestamp":"Mar 18 2014 19:00:00 ICT","Acct-Authentic":"RADIUS","Acct-Session-Time":"522","Acct-Multi-Session-Id":"9508dfe3","Calling-Station-Id":"66667811424","Framed-IP-Address":"22.203.161.221","Acct-Session-Id":"b4d6ce919508dfe3","NAS-IP-Address":"22.95.56.33","Framed-Protocol":"GPRS-PDP-Context","Called-Station-Id":"hinternet","User-Name":"true","NAS-Port-Type":"Virtual","Service-Type":"Framed-User","NAS-Port":"110001","NAS-Port-Id":"UGW","3GPP-IMSI":"520002045549948","3GPP-Charging-ID":"2500386787","3GPP-Charging-Gateway-Address":"22.95.13.215","3GPP-GPRS-Negotiated-QoS-profile":"05-0b911f7196f7fe4401ffff006400","3GPP-SGSN-Address":"180.214.206.205","3GPP-GGSN-Address":"180.214.206.145","3GPP-IMSI-MCC-MNC":"520002","3GPP-GGSN-MCC-MNC":"520000","3GPP-NSAPI":"5","3GPP-Charging-Characteristics":"0800","3GPP-SGSN-MCC-MNC":"52000","3GPP-RAT-Type":"0x01","3GPP-MS-TimeZone":"0x8200","3GPP-Negotiated-DSCP":"","Acct-Unique-Session-Id":"ebe68d094f943d02","Timestamp":"1395144000"}

รูปที่ 45 ผลลัพธ์จากการรันสคริปต์แปลงเป็น JSON ในลักษณะท่อส่ง

4.3.13. การออกแบบ CU_php_rd

CU_php_rd ทำหน้าที่จัดการสคริปต์เอพีไอ (API) เพื่อใช้ค้นคืนเรเดียสล็อกและนำเสนอผ่านเว็บเซิร์ฟเวอร์ CU_www โดยพีเอชพีเซิร์ฟเวอร์จะเปิดพอร์ต 9000 โดยใช้เอพีไอเฟรมเวิร์กชื่อสลิม (Slim) มีรหัสต้นฉบับดังรูปที่ 46

```php
<?php
//ini_set("date.timezone","Asia/Bangkok");
use \Psr\Http\Message\ServerRequestInterface as Request;
use \Psr\Http\Message\ResponseInterface as Response;
require '../vendor/autoload.php';
require 'config.php';
$app = new \Slim\App(['settings' => $config]);

$app->get('/', function () {
    $name = "/srcip\n/dstip";
    return $name;
});

$app->get('/q', function (Request $request, Response $response, array $args) {
  //var_dump($_GET);
  $mongoip = $this->get('settings')['mongo_server'];

  $response->getBody()->write("Source IP $srcip");
  $manager = new MongoDB\Driver\Manager("mongodb://$mongoip");


  //Query
  //date
  $from = isset($_GET['from'])?strtotime($_GET['from']):"";

  if($from){
    $to = isset($_GET['to'])?strtotime($_GET['to']):$from+86400;
    $filter["Timestamp"] = ['$gte' => "$from" , '$lte' => "$to"];
    $month_f = date("M",$from);
    $month_t = date("M",$to);
    if($month_f != $month_t){
      //var_dump($month_f);
      die("from and to seperate month can not query");
    }
    $month = strtolower($month_f);
  }

  //source ip
  $srcip = ISSET($_GET['srcip'])?$_GET['srcip']:"";
  $filter["Framed-IP-Address"] =  $srcip;

  //var_dump($filter);
  //limit skip
  $limit = isset($_GET["limit"])?$_GET["limit"]:$this->get('settings')['limit'];
  $page = isset($_GET["page"])?$_GET["page"]:"0";
  $skip = $page*$limit;
  $options = ["limit" => $limit,"skip" => $skip,
    'projection' => [
      'Framed-IP-Address' => 1,
      'Calling-Station-Id' => 1,
      'Timestamp' => 1
    ]
  ];

  $query = new MongoDB\Driver\Query($filter, $options);
  $cursor = $manager->executeQuery('radius.'.$month, $query);
  foreach($cursor as $doc){
    //var_dump(new MongoDB\BSON\UTCDateTime(1416445411987));
    //$datetime = $utcdatetime->toDateTime();
    $result[] = $doc;
  }
  return $response->withJson($result);
});
$app->run();
```

รูปที่ 46 รหัสต้นฉบับเอพีไอเพื่อใช้ในการค้นคืนเรเดียสล็อก

บทที่ 5
การทดสอบและการวิเคราะห์ผล

5.1. วัตถุประสงค์ของการทดสอบ

เพื่อเป็นแนวทางในการนำสถาปัตยกรรมไมโครเซอร์วิสมาใช้สำหรับรองรับการขยายตัวของข้อมูลที่เพิ่มมากขึ้น ไม่ว่าจะเป็นการเพิ่มขึ้นของข้อมูลหรือเพิ่มขึ้นของอุปกรณ์ต่างๆที่ส่งเข้ามาจัดเก็บยังระบบจัดเก็บล็อก

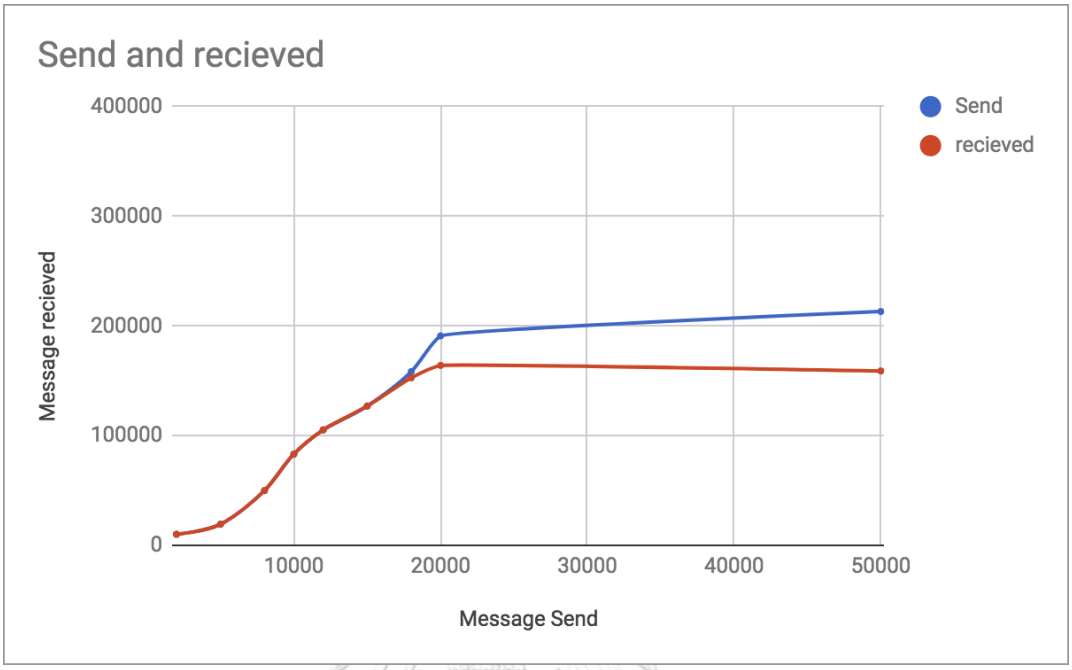
5.2. การทดสอบระบบในด้านแกน X

การทดสอบด้วยการส่ง log จาก CU_ng ไปยัง CU_kafka พบหากยังไม่มีการ scale ใดๆ หรือเริ่มที่ 1คอนเทนเนอร์ ได้ปรากฏข้อมูลดังตารางที่ 7 โดยทำการทดสอบครั้งละ 10 วินาที และเพิ่มจำนวนข้อมูลเพิ่มมากขึ้นตามลำดับดังตารางที่ 7

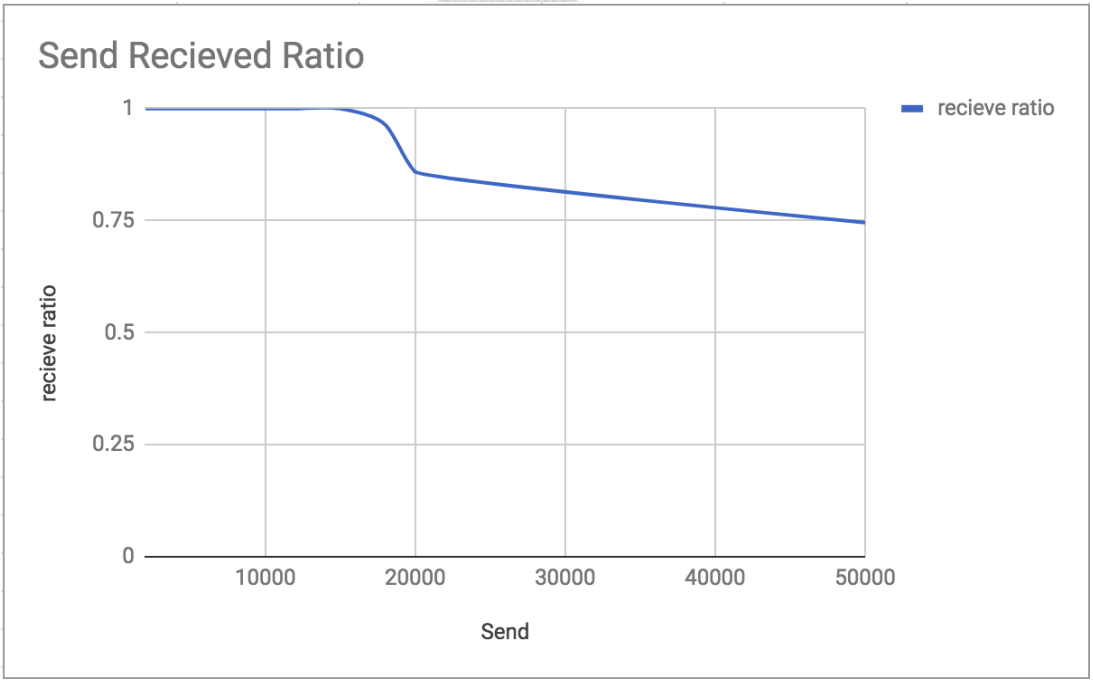
ตารางที่ 7 ผลการทดสอบเมื่อส่ง log จากคอนเทนเนอร์ CU_ng ไปยัง CU_kafka

Msg/s	Send	received	receive ratio (received/Send)
1003.82	10039	10039	1
1921.49	19216	19216	1
4987.15	49874	49874	1
8315.07	83153	83153	1
10524.06	105246	105246	1
12696.92	126983	126983	1
15835.62	158366	152664	0.9639947969
19093.11	190942	164020	0.859004305
21315.02	213173	158978	0.7457698677
32112.57	322920	158144	0.4897312028

เมื่อเปรียบเทียบระหว่างการส่งและการรับแล้วพบว่าสมรรถนะ (performance) ของการรับจะอยู่ที่ประมาณ 15000 และหากเกินค่าอัตราส่วน data log in/ out จะลดลงดังรูปที่ 47,รูปที่ 48



รูปที่ 47 ผลการทดสอบเมื่อส่งล็อกจาก CU_ng 1 คอนเทนเนอร์ ไปยัง CU_kafka 1 คอนเทนเนอร์



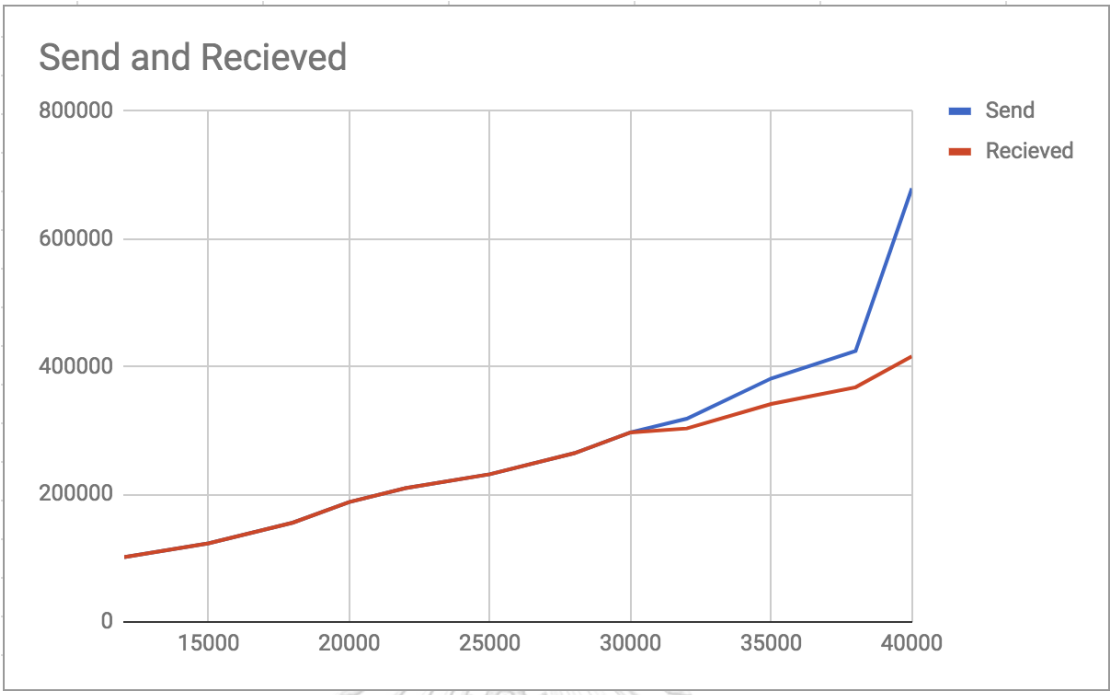
รูปที่ 48 อัตราส่วนระหว่างการรับต่อการส่งจาก CU_ng 1 คอนเทนเนอร์ไปยัง CU_kafka 1 คอนเทนเนอร์

เมื่อทำการเพิ่มคอนเทนเนอร์ของ CU_ng และ CU_kafka โดยเพิ่มเป็นอย่างละ 2 คอนเทนเนอร์ พบว่า ข้อมูลสามารถรับได้มากขึ้นดังตารางที่ 8

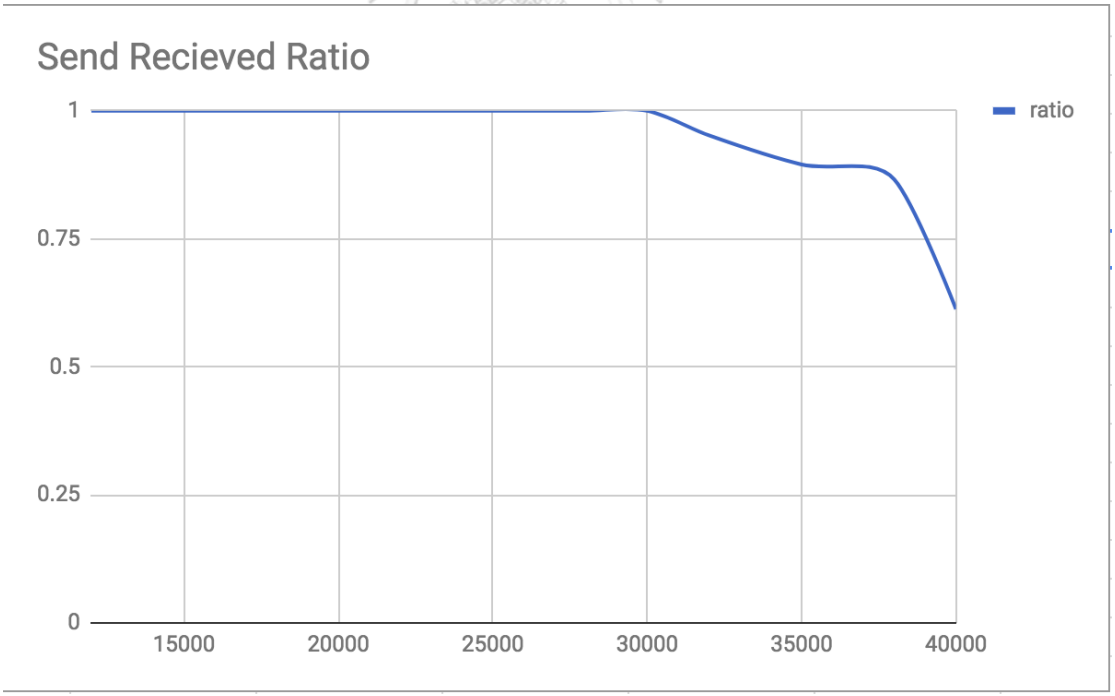
ตารางที่ 8 ผลการทดสอบเมื่อส่ง log จากคอนเทนเนอร์ CU_ng 2 ไปยัง CU_kafka 2 คอนเทนเนอร์

msg/s	Send	Received	ratio
10144.24	101447	101447	1
12298.45	122988	122988	1
15552.83	155538	155538	1
18775.55	187764	187764	1
20960.43	209622	209622	1
23137.08	231383	231383	1
26429.25	264327	264327	1
29689.7	296925	296925	1
31826.52	318281	303036	0.9521020733
38141.18	381435	341498	0.8952980193
42431.68	424357	367570	0.8661810692
67804.78	679117	416032	0.6126072532
64147.4	641586	407068	0.6344714504

จากตารางที่ 8 พบว่าอัตตราส่วนระหว่างการรับต่อการส่งเริ่มลดลงที่ปริมาณข้อมูลที่ 29000 msg/s ดังรูปที่ 49 และ รูปที่ 50



รูปที่ 49 ผลการทดสอบเมื่อส่งล็อกจาก CU_ng 2 คอนเทนเนอร์ ไปยัง CU_kafka 2 คอนเทนเนอร์



รูปที่ 50 อัตราส่วนระหว่างการรับต่อการส่งจาก CU_ng 2 คอนเทนเนอร์ ไปยัง CU_kafka 2 คอนเทนเนอร์

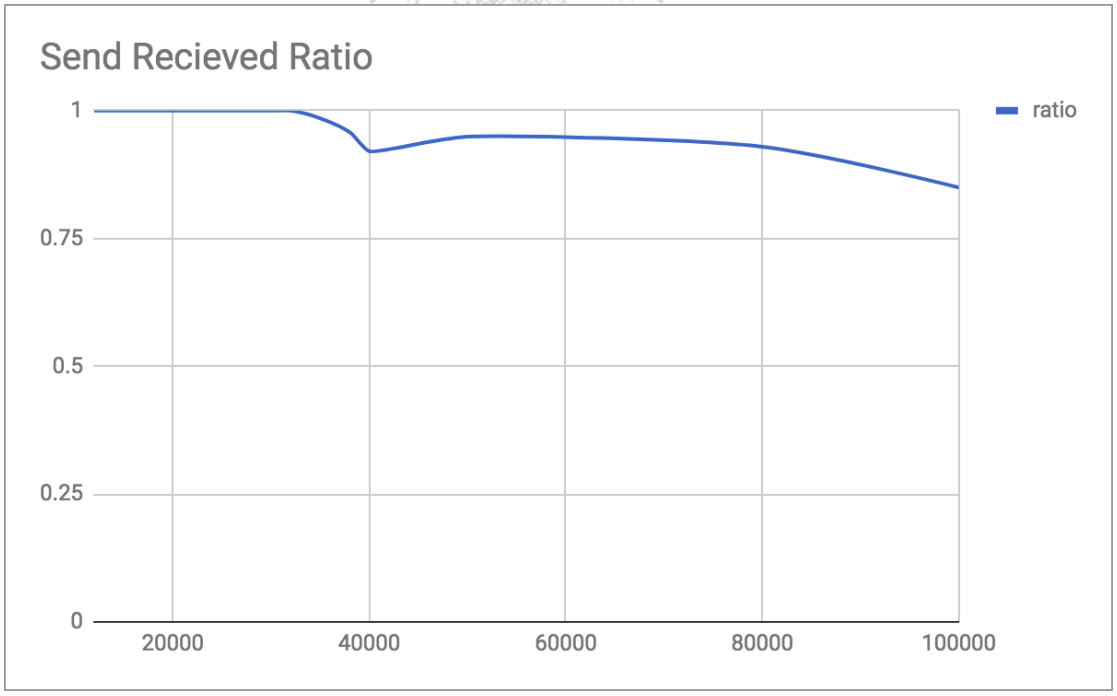
เมื่อทำการเพิ่มคอนเทนเนอร์ของ CU_ng และ CU_kafka เป็นอย่างละ 3 คอนเทนเนอร์ พบว่าข้อมูลสามารถรับได้มากขึ้น ดังตารางที่ 9

ตารางที่ 9 ผลการทดสอบเมื่อส่ง log จาก คอนเทนเนอร์ CU_ng 3 คอนเทนเนอร์ ไปยัง CU_kafka 3 คอนเทนเนอร์

msg/s	Send	Received	ratio
9768.42	97691	97691	1
11980.79	119811	119811	1
15345.88	153471	153471	1
18750.02	187515	187515	1
26363.4	263660	263660	1
0	296467	296467	1
31794.45	317971	317971	1
33994.25	339981	334958	0.985225645
36834.15	368443	352963	0.95798536
39385.82	399933	368179	0.9206017008
41415.42	420633	399230	0.9491171639
46652.09	466568	433557	0.9292471837
57784.63	584136	496272	0.8495829738
66071.26	662759	504487	0.7611922282

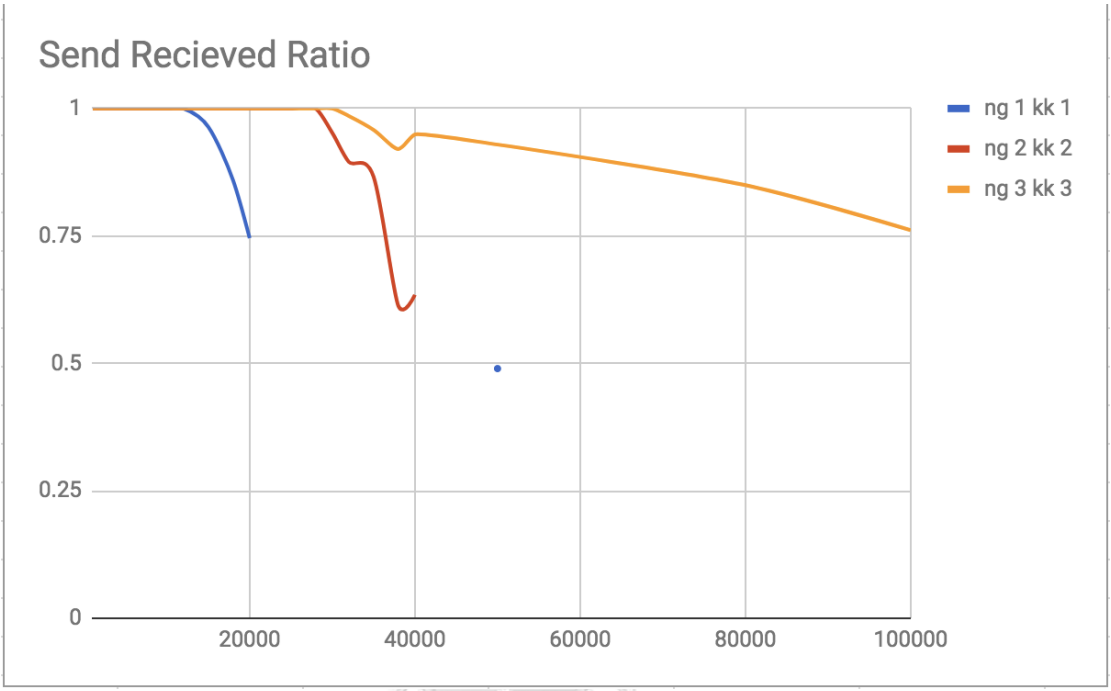
จากตารางที่ 9 พบว่าอัตราส่วนระหว่างการรับต่อการส่งเริ่มลดลงที่ปริมาณข้อมูลที่ 33000 – 45000 msg/s โดยช่วงดังกล่าวมีอัตราส่วนระหว่างการรับต่อการส่งมีค่าอยู่ที่ประมาณ 0.9 ดังรูปที่ 51 และรูปที่ 52

รูปที่ 51 ผลการทดสอบเมื่อส่งล็อกจาก CU_ng 3 คอนเทนเนอร์ ไปยัง CU_kafka 3 คอนเทนเนอร์



รูปที่ 52 อัตราส่วนระหว่างการรับต่อการส่งจาก CU_ng 3 คอนเทนเนอร์ไปยัง CU_kafka 3 คอนเทนเนอร์

เมื่อทำการเปรียบเทียบอัตราส่วนระหว่างการรับและการส่ง เมื่อเพิ่มคอนเทนเนอร์ครั้งละ 1 คอนเทนเนอร์ตามลำดับ จะได้ผลลัพธ์ดังรูปที่ 53



รูปที่ 53 อัตราส่วนระหว่างการรับต่อการส่งจาก CU_ng ไปยัง CU_kafka ตั้งแต่ 1-3 คอนเทนเนอร์ตามลำดับ

5.3. การทดสอบระบบในด้านแกน Y

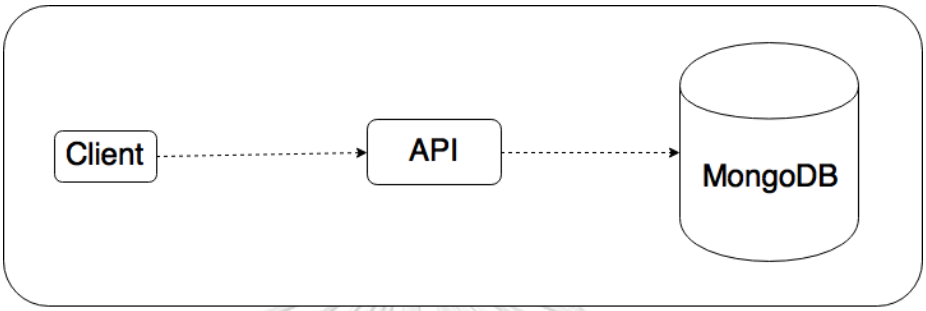
ทำการทดสอบโดยการจำแนกประเภทของล็อกประกอบไปด้วยเรเดียสล็อกและไฟร์วอล์ล็อก ซึ่งผลที่ได้จากการทดลองพบว่าการแยกประเภทของล็อกทำให้การออกแบบโครงสร้างฐานข้อมูลมีความยืดหยุ่นและมีความเหมาะสมในแต่ละประเภทและมีผลต่อประสิทธิภาพในการค้นคืนของล็อกทั้งสองประเภทซึ่งสอดคล้องตามคุณสมบัติของสถาปัตยกรรมไมโครเซอร์วิส

5.4. การทดสอบระบบในด้านแกน Z

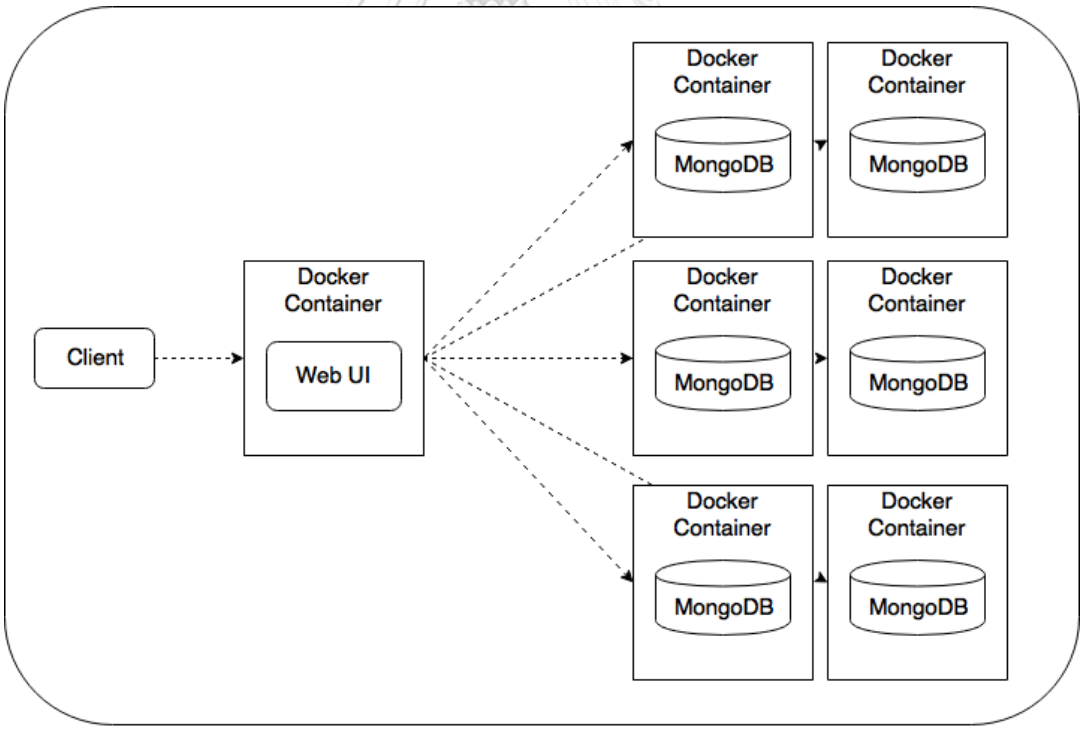
ในด้านแกน Z ได้ทำการทดสอบโดยเปรียบเทียบระหว่างแบบ Monolithic Architecture และ Microservice Architecture โดยจำลองการขยายตัวในลักษณะแกน Z และประเมินผล

5.4.1. รูปแบบการทดสอบ

ใช้วิธีการจับเวลาโดยส่งข้อมูล Query โดยใช้ IP และช่วงเวลาสอบถามไปยัง Radius Service และทำการจับเวลาเปรียบเทียบกับแบบ Monolithic และ Microservice ดังแสดงในรูปที่ 54 และ รูปที่ 55 โดยทำการเปรียบเทียบกับแบบ Microservice ซึ่งมีการขยายตัวในลักษณะแกน Z



รูปที่ 54 โครงสร้างการทดสอบโมโนลิทิก [12]



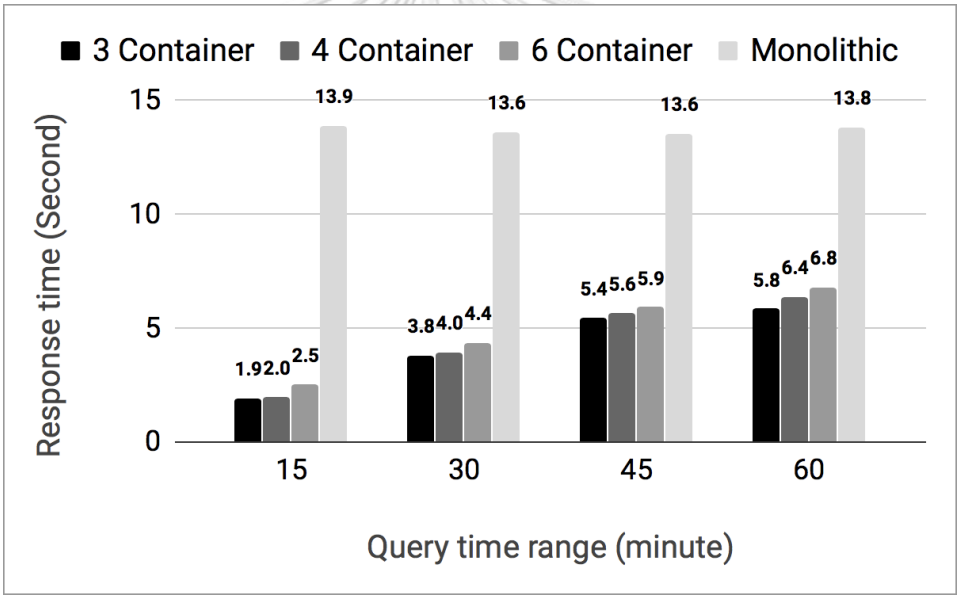
รูปที่ 55 โครงสร้างการทดสอบ Microservice Architecture โดยมีการขยายตัวแกน Z

5.4.2. ข้อมูลที่นำมาทดสอบ

ได้ทำการนำข้อมูลของ Radius Log ปริมาณ 9,000,000 Records เข้ามายังทั้งสองโครงสร้างที่ทำการทดสอบ และทำการเรียก API เพื่อขอค้นคืนข้อมูล หลังจากนั้นทำการจับเวลาและเปรียบเทียบผล รูปแบบการ Query จะสอบถามถึงข้อมูล Username โดยทำการส่ง parameter ในรูปแบบ Json ประกอบไปด้วย IP และช่วงเวลาที่ต้องการค้นหา ซึ่งการทดสอบครั้งนี้จะค้นหาภายใต้ช่วงเวลา 15,30,45,60 นาทีตามลำดับ

5.4.3. ผลการทดสอบ

พบว่าการ Query ในรูปแบบโมโนลิทิกในช่วงเวลาต่างๆ ไม่ว่าจะค้นหาระยะเวลาสั้น หรือยาวจะใช้ระยะเวลาในการค้นหาไม่ต่างกัน คือประมาณ 14 วินาทีโดยประมาณ ซึ่งต่างจากการขยายตัวในแนวแกน Z ในรูปแบบไมโครเซอร์วิส ซึ่งใช้เวลาสูงสุดเพียง 6 วินาทีเท่านั้น ดังรูปที่ 56



รูปที่ 56 เปรียบเทียบการค้นคืนระหว่างโมโนลิทิกและไมโครเซอร์วิส

จากการใช้ Docker คอนเทนเนอร์ช่วยในการขยายตัวโดยใช้ MongoDB และขยายด้วยการ Sharding พบว่าความเร็วในการค้นคืนเมื่อเปรียบเทียบกับแบบ Monolithic แล้วจะเร็วขึ้นประมาณ 50% ที่ช่วงเวลาในการค้นหาที่ 60 นาที อย่างไรก็ตาม การเพิ่มจำนวนของคอนเทนเนอร์ จาก 3 ไปยัง 6 คอนเทนเนอร์ นั้นมีค่าความเร็วในการค้นคืนไม่ต่างกันมากนัก สิ่งที่น่าสังเกตก็คือ เมื่อทำการเพิ่มจำนวนคอนเทนเนอร์มากขึ้นกลับทำให้เวลาในการค้นคืนมากขึ้น คาดว่าเนื่องมาจากการถูกจอง Resource ของ Docker และการประมวลผลของ MongoDB ซึ่งต้องรอให้ทุกคอนเทนเนอร์ทำงานเสร็จพร้อมกัน จึงจะส่งข้อมูลตอบกลับไปยัง API

5.5. สรุปผลการทดลอง

การนำสถาปัตยกรรมไมโครเซอร์วิสมาประยุกต์ใช้เพื่อช่วยในการขยายตัวให้กับระบบการค้นคืนข้อมูลจราจรคอมพิวเตอร์นั้น ทำให้แก้ไขปัญหาการขยายตัวได้จริง ถึงแม้ประสิทธิภาพจะไม่รวดเร็วขึ้นเท่าใด แต่พบว่าการแบ่งหน้าที่ให้กับแต่ละเซอร์วิสนั้น ทำให้ถูกโฟกัสไปยังตรรกะทางธุรกิจของแต่ละเซอร์วิสนั้นๆ ซึ่งจุดประสงค์ที่แตกต่างกัน จะนำไปสู่วิธีการที่เหมาะสมของแต่ละเซอร์วิสที่ไม่เหมือน และทำให้การวินิจฉัยปัญหานั้นกระทำได้ง่ายขึ้น กอปรกับความยืดหยุ่นของ MongoDB สามารถทำให้การขยายตัวในแกน Z นั้นกระทำได้ง่าย และความสามารถของ Docker Swarm จะช่วยให้การขยายตัวในแกน X ทำได้ง่ายเช่นเดียวกัน ในส่วนของการขยายตัวในแกน Y สามารถทำได้แต่ยังมีความเป็นอัตโนมัติได้ไม่มาก เนื่องจากต้องมีการ customize Web Interface เพื่อให้รองรับกับ Feature ใหม่ที่เพิ่มเข้ามาในอนาคต

บทที่ 6
สรุปผลการวิจัย

6.1. สรุปผลการวิจัย

วิทยานิพนธ์ฉบับนี้ได้นำเสนอแนวทางการพัฒนาระบบบันทึกจราจรเครือข่ายด้วยสถาปัตยกรรมไมโครเซอร์วิส ซึ่งมีข้อดีคือความสามารถการขยายตัวด้วยแบบจำลองลูกบาศก์การขยายที่รองรับการขยายตัวของระบบใน 3 มิติ จากการศึกษาและทดสอบในเบื้องต้น โดยทำการทดลองจำลองระบบด้วยโมเดลของการขยายตัวทั้งสามแกน และใช้ฐานข้อมูลไม่สัมพันธ์ MongoDB พบว่าการขยายตัวของ MongoDB มีความยืดหยุ่นสูง เว็บแอปพลิเคชันสามารถ เพิ่มจำนวน Node ของ Firewall และเพิ่มเดือนของ Radius ได้ตามตรรกะทางธุรกิจที่กำหนดไว้

สำหรับแกน X การขยายตัวทางด้านแกน X ช่วยเพิ่มความสามารถในด้านการรับข้อมูลเพื่อให้การรับข้อมูลล็อกได้ตามปริมาณที่ต้องการได้เป็นอย่างดี

สำหรับการขยายตัวด้านแกน Y ข้อดี คือ ช่วยในการแยกประเภทของข้อมูลที่ไม่เหมือนกัน สามารถแยกตามประเภทของข้อมูลเพื่อจัดเก็บในฐานข้อมูลด้วยโครงสร้างที่เหมาะสมสำหรับข้อมูลแต่ละประเภท ทำให้การค้นคืนแต่ละประเภทมีประสิทธิภาพที่ดีกว่า

ในด้านแกน Z การขยายตัวทางด้านแกน Z จะช่วยเพิ่มความเร็วในการค้นคืนอย่างเห็นได้ชัด ซึ่งทำให้เกิดประสิทธิภาพในการค้นคืนอย่างสูงสุด

6.2. ข้อจำกัดในงานวิจัย

โปรมแกรมรับล็อก Syslog-ng ยังไม่สามารถขยายตัวในลักษณะแกน X ได้การรับล็อกจึงขยายในลักษณะของแกน Y ซึ่งก็คือการแบ่งการรับล็อกของเรเดียสล็อกและไฟร์วอลล็อกทดแทน

6.3. งานวิจัยในอนาคต

งานในอนาคตที่ควรศึกษาเพิ่มเติม ได้แก่ การเติมฟังก์ชันสำหรับการวิเคราะห์พฤติกรรมการเข้าใช้งานจาก Radius และ Firewall และแจ้งเตือนเมื่อพบพฤติกรรมที่ผิดปกติต่อไป

รายการอ้างอิง

1. พระราชบัญญัติว่าด้วยการกระทำผิดเกี่ยวกับคอมพิวเตอร์ พุทธศักราช *2550*. 2550. p. 11.
2. Richardson, C. *Microservice Architecture*. [cited Nov 2017; Available from: http://microservices.io/patterns/microservices.html.
3. *Microservices from Design to Deployment*. 2016: NGINX.
4. Fowler, M. *Decentralized Data Management*. Available from: https://martinfowler.com/articles/microservices.html.
5. Fisher, M.L.A.a.M.T., *The art of scalability: Scalable web architecture, processes, and organizations for the modern enterprise*. 2009.
6. *Log Samples*. Available from: https://ossec-docs.readthedocs.io/en/latest/log_samples.
7. *Docker- Build, Ship, and Run Any App, Anywhere*. Available from: https://www.docker.com.
8. *Apache Kafka Overview*. Available from: https://www.cloudera.com/documentation/kafka/1-2-x/topics/kafka.html
9. Malavalli, D. and S. Sathappan, *Scalable microservice based architecture for enabling DMTF profiles*, in *Proceedings of the 11th International Conference on Network and Service Management, CNSM 2015*. 2015, IEEE. p. 428-432.
10. Richardson, C. *Pattern: Monolithic Architecture*. [cited Nov 2017; Available from: http://microservices.io/patterns/monolithic.html.
11. *balabit/syslog-ng*. Available from: https://hub.docker.com/r/balabit/syslog-ng/.
12. Ueda, T., T. Nakaike, and M. Ohara, *Workload characterization for microservices*, in *2016 IEEE International Symposium on Workload Characterization (IISWC*. 2016, IEEE. p. 1-10.

ภาคผนวก

ภาคผนวก ก

Kafka.go

```go
package main
//Version 0.1
//Create by Chakrit<chakrit@softnix.co.th>
import (
        "fmt"
        "time"
        "github.com/confluent-kafka-go/kafka"
        "github.com/go-zookeeper/zk"
        "github.com/tidwall/gjson"
        "strings"
        "bufio"
        "os"
        "io"
        "flag"

)

func getPartition(topic string) []string {
  c, _, err := zk.Connect([]string{"zookeeper"}, time.Second) //*10)
  if err != nil {
    panic(err)
  }
  path := "/brokers/topics/"+topic+"/partitions"
  children, _, _ , err := c.ChildrenW(path)
  if err != nil {
    panic(err)
  }
  //fmt.Println(children)
  return children
}

func getBroker() []string {
  c, _, err := zk.Connect([]string{"zookeeper"}, time.Second) //*10)
  if err != nil {
    panic(err)
  }
  path := "/brokers/ids"
  children, _, _ , err := c.ChildrenW(path)
  if err != nil {
    panic(err)
  }
  results := make([]string,len(children))

```

รูปที่ 57 ชุดคำสั่ง kafka.go (1)

```
  for key, element := range children {
    id_path := path+"/"+element
    //fmt.Printf("%+v\n", id_path)
    result_byte,_,err := c.Get(id_path)
    if err != nil {
      panic(err)
    }
    json_kafkaid := string(result_byte[:])
    value := gjson.Get(json_kafkaid, "endpoints.0")
    hostid := value.String()
    //fmt.Printf("%+v\n",hostid)
    arrip := strings.Split(hostid,"//")
    //arrip := strings.Split(arr[1],':')
    results[key] = arrip[1]
    //fmt.Printf("%+v\n",results)
  }
  return results
}

func main() {
        var brokers string
        var topic string
        var max_partition int32
        result := getBroker()
        brokerPtr := flag.String("broker", "", "a String")
        topicPtr := flag.String("topic","syslog","a String")
        flag.Parse()
        topic = *topicPtr
        partitions := getPartition(topic)
        max_partition = int32(len(partitions))
        brokers = strings.Join(result,",")
        if *brokerPtr != "" {
          brokers = *brokerPtr
        }
        p, err := kafka.NewProducer(&kafka.ConfigMap{"bootstrap.servers": brokers})
        if err != nil {
                panic(err)
        }
        fmt.Println("Connect brokers " + brokers);
        fmt.Printf("Connect partition %+v\n", partitions);
```

รูปที่ 58 ชุดคำสั่ง kafka.go (2)

```
84          defer p.Close()
85
86          // Delivery report handler for produced messages
87          go func() {
88                  partitions := getPartition(topic)
89                  max_partition = int32(len(partitions))
90                  fmt.Printf("Update max partition: %d\n", max_partition)
91                  for e := range p.Events() {
92                          switch ev := e.(type) {
93                          case *kafka.Message:
94                                  if ev.TopicPartition.Error != nil {
95                                          fmt.Printf("Delivery failed: %v\n", ev.TopicPartition)
96                                  } else {
97                                          fmt.Printf("Delivered message to %v\n", ev.TopicPartition)
98                                  }
99                          }
100                 }
101         }()
102
103         // Produce messages to topic (asynchronously)
104         //topic := "syslog"
105         reader := bufio.NewReader(os.Stdin)
106         var n int32
107         n = 0
108         for {
109           word, err := reader.ReadBytes('\n')
110           str_word := string(word[:])
111           arr_str_word := strings.Split(str_word, "\n")
112           str_word = arr_str_word[0]
113           if len(word) != 0 {
114             /*
115             if isln == byteln {
116               word2 = word[:len(word)-1]
117             }else{
118               word2 = word
119               fmt.Printf("%+v",word)
120             }
121             */
122             word2 := str_word
123             //fmt.Println(word2)
124             if n >= max_partition {
125               n = 0
126             }
127             p.Produce(&kafka.Message{
128               TopicPartition: kafka.TopicPartition{Topic: &topic, Partition: kafka.PartitionAny},
129               //TopicPartition: kafka.TopicPartition{Topic: &topic, Partition: n},
130               Value: []byte(word2),
131             },nil)
132             n++
133           }else{
134             if err != nil {
135               if err != io.EOF {
136                   fmt.Println(err)
137               }
138               break
139             }
140           }
141         }
142
143         // Wait for message deliveries
144         p.Flush(60 * 1000)
145 }
```

รูปที่ 59 ชุดคำสั่ง kafka.go (3)

ภาคผนวก ข

Dockerfile

Dockerfile สำหรับ สร้าง CU_php

```
FROM php:7.1-fpm-jessie
MAINTAINER Andras Mitzki <andras.mitzki@balabit.com>
MAINTAINER Chakrit Phain <chakrit@softnix.co.th>

RUN apt-get update -qq && apt-get install -y wget gnupg2
#kafka
RUN wget -qO - https://packages.confluent.io/deb/4.1/archive.key | apt-key add -
RUN echo "deb http://packages.confluent.io/deb/4.1 stable main"| tee --append /etc/apt/sources.list.d/confluent.io.list

ADD openjdk-libjvm.conf /etc/ld.so.conf.d/openjdk-libjvm.conf
RUN ldconfig

#Softnix
RUN apt-get update -qq && apt-get install -y mongodb-clients  && apt -y install librdkafka1 && apt -y install libzookeeper-mt2

#Dev
RUN apt -y  install iputils-ping && apt -y  install telnet && apt -y  install net-tools && apt -y install librdkafka-dev && apt -y install vim

ADD www.conf /usr/local/etc/php-fpm.d/
ADD rdkafka.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
ADD zookeeper.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
ADD stomp.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
ADD mongodb.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
RUN docker-php-ext-enable stomp
RUN docker-php-ext-enable rdkafka
RUN docker-php-ext-enable zookeeper
RUN docker-php-ext-install sockets
RUN docker-php-ext-enable mongodb
```

รูปที่ 60 Docker ใช้สำหรับสร้าง CU_php

Dockerfile สำหรับ สร้าง CU_ng

```
FROM php:7.1-fpm-jessie
MAINTAINER Andras Mitzki <andras.mitzki@balabit.com>
MAINTAINER Chakrit Phain <chakrit@softnix.co.th>

RUN apt-get update -qq && apt-get install -y wget gnupg2
RUN wget -qO - https://packages.confluent.io/deb/4.1/archive.key | apt-key add -
RUN echo "deb http://packages.confluent.io/deb/4.1 stable main"| tee --append
/etc/apt/sources.list.d/confluent.io.list
RUN wget -qO - https://download.opensuse.org/repositories/home:/laszlo_budai:/syslog-
ng/Debian_8.0/Release.key | apt-key add -
RUN echo "deb http://download.opensuse.org/repositories/home:/laszlo_budai:/syslog-ng/Debian_8.0 ./" |
tee --append /etc/apt/sources.list.d/syslog-ng-obs.list
RUN apt-get update -qq && apt-get install -y syslog-ng && apt-get install -y mongodb-clients

ADD openjdk-libjvm.conf /etc/ld.so.conf.d/openjdk-libjvm.conf
RUN ldconfig

EXPOSE 514/udp
EXPOSE 601/tcp
EXPOSE 6514/tcp

#Softnix
RUN apt -y  install iputils-ping && apt -y  install telnet && apt -y  install net-tools && apt -y install
librdkafka-dev && apt -y install librdkafka1 && apt -y install libzookeeper-mt2 && apt -y install vim
ADD www.conf /usr/local/etc/php-fpm.d/
ADD rdkafka.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
ADD zookeeper.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
ADD stomp.so /usr/local/lib/php/extensions/no-debug-non-zts-20160303/
RUN docker-php-ext-enable stomp
RUN docker-php-ext-enable rdkafka
RUN docker-php-ext-enable zookeeper
RUN docker-php-ext-install sockets
ADD ./html/ /var/www/html/
ADD ./goroot.tar.gz /opt/
ADD ./golang/src/github.com/ /opt/goroot/src/github.com/
RUN mkdir /opt/kafka/
RUN mkdir /opt/kafka/libs/
ADD ./libs/ /opt/kafka/libs/
RUN echo 'export PATH="/opt/goroot/bin/:$PATH"' >> /etc/profile

#go

ENTRYPOINT ["/usr/sbin/syslog-ng", "-F"]
```

รูปที่ 61 Dockerfile สำหรับ สร้าง CU_ng

ประวัติผู้เขียนวิทยานิพนธ์

นายชาคริต ผาอินทร์ เกิดเมื่อวันที่ 14 มิถุนายน พ.ศ. 2522 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาตรีหลักสูตรครุศาสตร์อุตสาหกรรมบัณฑิต (ค.อ.บ.) สาขาวิศวกรรมไฟฟ้า คณะครุศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าพระนครเหนือ ในปีการศึกษา 2546 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิทยาศาสตร์คอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2559