สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่

นายพิทยุตม์ ตั้งสัจจะธรรม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2558

Hybrid Architecture for Large Scale Log Processing

Mr. Pittayut Tangsatjatham

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2015

หัวข้อวิทยานิพนธ์	สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่
โดย	นายพิทยุตม์ ตั้งสัจจะธรรม
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. ณัฐวุฒิ หนูไพโรจน์
อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม	ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร. สุพจน์ เตชวรสินสกุล)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. วีระ เหมืองสิน)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร. ณัฐวุฒิ หนูไพโรจน์)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม
(ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร. ภุชงค์ อุทโยภาศ)

พิทยุตม์ ตั้งสัจจะธรรม : สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่ (Hybrid Architecture for Large Scale Log Processing) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: ผศ. ดร. ณัฐวุฒิ หนูไพโรจน์, อ.ที่ปรึกษาวิทยานิพนธ์ร่วม: ผศ. ดร. เกริก ภิรมย์โสภา, 47 หน้า.

การประมวลผลบันทึกของระบบงานขนาดใหญ่ที่มาจากหลายๆ เครื่องแม่ข่ายจะพบว่า บันทึกของระบบงานของแต่ละเครื่องแม่ข่ายนั้นมีขนาดใหญ่ ถูกส่งมาตลอดเวลา และมีรูปแบบที่แตกต่างกัน ซึ่งทำให้การประมวลผลข้อมูลของบันทึกของระบบงานเหล่านี้ทำได้ยากยิ่ง ตัวอย่างเช่น การตรวจจับความผิดปกติของระบบเป็นระบบที่จะต้องวิเคราะห์ข้อมูลล่าสุดร่วมกับข้อมูลในอดีต ถ้าหากเราต้องการความแม่นยำในการตรวจจับความผิดปกติอย่างทันท่วงที เราจะต้องประมวลผลข้อมูลเหล่านี้ให้ได้ภายในระยะเวลาที่จำกัด เพื่อแก้ปัญหาเหล่านี้ งานวิจัยนี้นำเสนอ สถาปัตยกรรมแบบผสมสาหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่โดยใช้ Apache Spark สำหรับทำการประมวลผลข้อมูล และ Apache Flume สำหรับการจัดการกับข้อมูล โดยใช้การตรวจจับความผิดปกติโดยใช้แบบจำลองเวลาซาริมา มาทดสอบและประเมินระบบ ซึ่งสถาปัตยกรรมที่นำเสนอจะประมวลผลทั้งในรูปแบบ Batch และ Real-Time จากผลการทดสอบพบว่า การใช้สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่นั้นสามารถช่วยเพิ่มประสิทธิภาพในการทำงานกับบันทึกของระบบงานขนาดใหญ่ได้ดียิ่งขึ้น โดยสามารถประมวลผลข้อมูลที่มีขนาดใหญ่แล้วนำมาสร้างแบบจำลองที่มีความซับซ้อน เพื่อใช้ทำการตรวจจับความผิดปกติอย่างมีประสิทธิภาพ ภายในระยะเวลาที่จำกัด

ภาควิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อนิสิต	.....
สาขาวิชา	วิศวกรรมคอมพิวเตอร์	ลายมือชื่อ อ.ที่ปรึกษาหลัก	.....
ปีการศึกษา	2558	ลายมือชื่อ อ.ที่ปรึกษาร่วม	.....

5670311621 : MAJOR COMPUTER ENGINEERING

KEYWORDS: HADOOP / REAL-TIME / LOG PROCESSING / LARGE-SCALE / HYBRID PROCESSING

PITTAYUT TANGSATJATHAM: Hybrid Architecture for Large Scale Log Processing. ADVISOR: ASST. PROF. NATAWUT NUPAIROJ, Ph.D., CO-ADVISOR: ASST. PROF. KRERK PIROMSOPA, Ph.D., 47 pp.

Log processing can be very challenging, especially for environments with lots of servers. In these environments, log data is large, coming at high-speed, and have various formats, the classic case of big data problem. This makes big data log processing very difficult. For example, anomaly detection needs to process both latest data and historical data. To get good accuracy, large amount of data must be processed in real-time. To solve this problem, this research proposes a hybrid architecture for log anomaly detection using Apache Spark for data processing and Apache Flume for data collecting. To demonstrate the capabilities of our proposed solution, we implement a SARIMA-based anomaly detection as a case study. The experimental results clearly indicated that our proposed architecture can support log processing in large-scale environment effectively.

Department:	Computer Engineering	Student's Signature .....
Field of Study:	Computer Engineering	Advisor's Signature .....
Academic Year:	2015	Co-Advisor's Signature .....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความกรุณาอย่างสูงจาก ผศ. ดร. ณัฐวุฒิ หนูไพโรจน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ และ ผศ. ดร. เกริก ภิรมย์โสภา อาจารย์ที่ปรึกษาวิทยานิพนธ์ร่วม ที่ได้กรุณาสละเวลาให้คำปรึกษา คำแนะนำ รวมทั้งให้แนวคิดที่เป็นประโยชน์ในการทำวิจัย และช่วยชี้แนะแนวทางในการแก้ปัญหาต่างๆ ที่เกิดขึ้นระหว่างการทำวิจัย ขอขอบพระคุณอาจารย์มา ณ โอกาศนี้

ขอกราบขอบพระคุณ ผศ. ดร. วีระ เหมือง, ผศ. ดร. ภุชงค์ อุทโยภาศ, ผศ. ดร. เกริก ภิรมย์โสภา และ ผศ. ดร. ณัฐวุฒิ หนูไพโรจน์ คณะกรรมการคุมสอบวิทยานิพนธ์เป็น อย่างยิ่งที่ได้กรุณาตรวจสอบและให้ข้อเสนอแนะอันเป็นประโยชน์ ในการปรับปรุงและแก้ไขวิทยานิพนธ์นี้ให้มีความถูกต้องและสมบูรณ์มากยิ่งขึ้น

ขอกราบขอบพระคุณ คณาจารย์ทุกท่านในภาควิชาวิศวกรรมคอมพิวเตอร์ จุฬาลงกรณ์ มหาวิทยาลัย ที่ให้คำแนะนา ความรู้ และแนวทางการทำวิทยานิพนธ์

ขอขอบคุณ เพื่อนๆ พี่ๆ หลักสูตรวิศวกรรมคอมพิวเตอร์ สาหรับกำลังใจ และคำแนะนำใน การจัดทาวิทยานิพนธ์

สุดท้ายนี้ ขอกราบขอบพระคุณ บิดา มารดา รวมถึงสมาชิกในครอบครัว ที่ให้การสนับสนุน และให้กำลังใจที่ดีเสมอมา

สารบัญ

หน้า

บทคัดย่อภาษาไทย ..... ง

บทคัดย่อภาษาอังกฤษ ..... จ

กิตติกรรมประกาศ ..... ฉ

สารบัญ ..... ช

สารบัญภาพ ..... ญ

สารบัญตาราง ..... ฏ

บทที่ 1 บทนำ 1

1.1 ความเป็นมาและความสำคัญของปัญหา 1

1.2 วัตถุประสงค์ของงานวิจัย 4

1.3 ขอบเขตของงานวิจัย 4

1.4 ประโยชน์ที่คาดว่าจะได้รับ 4

1.5 แผนการดำเนินงานวิจัย 4

1.6 ผลงานตีพิมพ์ 5

บทที่ 2 ทฤษฎีและงานวิจัยที่เกี่ยวข้อง 6

2.1 บันทึกของระบบงาน (Log) 6

2.2 Big Data 7

2.3 Resilient Distributed Datasets (RDDs) 8

2.4 Discretized Streams (D-Stream) 8

2.5 Apache Hadoop [18, 19] 9

2.5.1 สถาปัตยกรรมของ HDFS 9

2.5.2 Hadoop Yarn (MapReduce version 2) 11

2.6 Apache Flume [20] 12

หน้า

2.6.1 สถาปัตยกรรมของ Apache Flume 12
2.7 Apache Spark [19, 21] 13
2.8 การแก้ปัญหาไฟล์ขนาดเล็กบน Hadoop 13
2.9 แบบจำองอนุกรมเวลาซาริมา (SARIMA) 14
2.10 Akaike information criterion (AIC) 15
2.11 Bayesian information criterion (BIC) 16
2.12 Maximum likelihood estimation (MLE) 16
2.13 งานวิจัยที่เกี่ยวข้อง 16
2.13.1 โครงสร้างระบบประมวลผลบันทึกของระบบงาน 16
2.13.1.1 โครงสร้างการประมวลผลบันทึกของระบบงาน แบบ Batch 17
2.13.1.2 โครงสร้างการประมวลผลบันทึกของระบบงาน แบบ Real-Time 18
2.13.2 สถาปัตยกรรมแบบ Three-layered lambda 19
2.13.3 Anomaly detection base on SARIMA 20
บทที่ 3 แนวคิดและวิธีดำเนินงานวิจัย 21
3.1 ปัญหาการประมวลผลบันทึกของระบบงาน 21
3.2 หลักการทำงานของสถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่ 22
3.3 หลักการทำงานของการพยากรณ์ปริมาณการใช้งานเครือข่ายโดยใช้บันทึกของระบบงาน 23
3.4 การสร้างแบบจำลองอนุกรมเวลาซาริมา 25

3.5 การตรวจหาปริมาณการใช้งานเครือข่ายที่ผิดปรกติโดยใช้แบบจำลองอนุกรมเวลาซาริมา 26

3.6 นิยามศัพท์ในงานวิจัย 28

บทที่ 4 การพัฒนาเครื่องมือและการทดสอบ 31

4.1 สภาพแวดล้อมที่ใช่ในการทดสอบ 31

4.2 ข้อมูลที่ใช้ในการทดสอบ 31

4.3การทดสอบประสิทธิภาพ 31

4.3.1 การทดสอบประสิทธิภาพในการพยากรณ์ปริมาณการใช้งานเครือข่าย 32

4.3.2 การทดสอบประสิทธิภาพในการตรวจหาความผิดปรกติของเครือข่าย 36

4.3.3 การทดสอบประสิทธิภาพในการประมวลผลและพยากรณ์ปริมาณการใช้งานเครือข่าย 39

4.3.4 การทดสอบประสิทธิภาพในการทำงานในสภาวะแบบ Real-Time 41

4.5 วิเคราะห์ผลการทดลอง 42

บทที่ 5 บทสรุปและแนวทางในการพัฒนาต่อ 43

5.1 สรุปผลการวิจัย 43

5.2 แนวทางในการพัฒนาต่อ 43

รายการอ้างอิง 44

ประวัติผู้เขียนวิทยานิพนธ์ 47

สารบัญภาพ

หน้า

ภาพที่ 1 ตัวอย่างไฟล์ของบันทึกของระบบงาน 6
ภาพที่ 2 3Vs of Big Data 7
ภาพที่ 3 RDDs Transformation 8
ภาพที่ 4 D-Streams Operation 8
ภาพที่ 5 สถาปัตยกรรมของ HDFS 9
ภาพที่ 6 สถาปัตยกรรมของ Hadoop YARN 11
ภาพที่ 7 สถาปัตยกรรมของ Apache Flume 12
ภาพที่ 8 Spark Framework 13
ภาพที่ 9 โครงสร้างการประมวลบันทึกของระบบงาน โดยใช้ Hadoop 17
ภาพที่ 10 โครงสร้างการประมวลผลบันทึกของระบบงาน แบบ Real-Time 19
ภาพที่ 11 Three-layered lambda architecture 20
ภาพที่ 12 สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่ .. 23
ภาพที่ 13 แสดงการพยากรณ์ปริมาณการใช้งานเครือข่าย 23
ภาพที่ 14 แสดงตัวอย่างการพยากรณ์ที่ละ step (a) step ที่ 146 (b) step ที่ 147 24
ภาพที่ 15 แสดงตัวอย่างการพยากรณ์ที่ละ 18 steps (a) step ที่ 145 ถึง step ที่ 162 (b) step ที่ 163 ถึง step ที่ 180 25
ภาพที่ 16 ปริมาณการใช้งานเครือข่ายจากบันทึกของระบบงาน โดยใช้ 5 นาทีเป็นหน่วยของเวลา 26
ภาพที่ 17 ปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง เทียบกับผลจากการพยากรณ์ 27
ภาพที่ 18 ตัวอย่าการแจ้งเตือนความผิดปรกติของการใช้งานอินเทอร์เน็ต 27
ภาพที่ 19 แสดงถึงช่วงเวลาที่ใช้ในการสร้างแบบจำลองและพยากรณ์ 28

ภาพที่ 20 ผลการประมวลผลโดยใช้ช่วงเวลาที่แตกต่างกันในการประมวลผล..... 29

ภาพที่ 21 กราฟแสดงเวลาที่ใช้งานการประมวลผลเมื่อเทียบระหว่างช่วงเวลาของข้อมูลและปริมาณของข้อมูลที่ใช้งานการประมวลผล (a) 2 Nodes, (b) 3 Nodes, (c) 4 Nodes, และ 5 Nodes 30

ภาพที่ 22 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ (a) ระดับ 5 นาที, (b) ระดับ 10 นาที, (c) ระดับ 20 นาที, (d) ระดับ 30 นาที, และ (e) ระดับ 60 นาที 33

ภาพที่ 23 ตัวอย่างข้อมูลของปริมาณการใช้งานเครือข่าย 34

ภาพที่ 24 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ โดยการนำวันที่มีปริมาณเท่าๆ กัน โดย (a) ใช้ข้อมูลย้อนหลัง 2 วัน (b) ใช้ข้อมูลย้อนหลัง 7 วัน 35

ภาพที่ 25 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ โดยการนำวันธรรมดามาทำนายวันหยุด โดย (a) ใช้ข้อมูลย้อนหลัง 2 วัน (b) ใช้ข้อมูลย้อนหลัง 7 วัน 35

ภาพที่ 26 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ โดยการนำวันหยุดมาทำนายวันธรรมดา โดย (a) ใช้ข้อมูลย้อนหลัง 2 วัน (b) ใช้ข้อมูลย้อนหลัง 7 วัน 36

ภาพที่ 27 เวลาที่ใช้งานการประมวลผล (a) ระดับ 5 นาที, (b) ระดับ 10 นาที, (c) ระดับ 20 นาที, (d) ระดับ 30 นาที, และ (e) ระดับ 60 นาที 40

ภาพที่ 28 แสดงการทำงานประมวลผลของระบบในสภาวะ Real-Time 41

สารบัญตาราง

หน้า

ตารางที่ 1 ตารางแสดงการแก้ไขปัญหาการไฟล์ขนาดเล็กบนระบบการเก็บข้อมูลแบบกระจาย 14

ตารางที่ 2 ผลลัพธ์ของการทดสอบประสิทธิภาพในการตรวจหาความผิดปรกติของเครือข่าย 37

ตารางที่ 3 ผลการทดสอบประสิทธิภาพในการประมวลผลแบบ Real-Time 41

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในองค์กรต่างๆ จะต้องมีระบบ IT เป็นพื้นฐานในการทำงาน ซึ่งจะประกอบไปด้วยอุปกรณ์และบริการต่างๆ โดยที่อุปกรณ์และบริการเหล่านี้จะสร้างบันทึกของระบบงาน (Log) ที่ใช้เก็บบันทึกพฤติกรรมที่เกิดขึ้นกับตัวอุปกรณ์ หรือบริการนั้นๆ ระบบประมวลผลบันทึกของระบบงาน จึงเป็นเครื่องมือที่สำคัญที่จะใช้ในการสนับสนุนการทำงานของระบบงานขนาดใหญ่ (Large Scale System) เช่น การนำบันทึกของระบบงาน มาประมวลผลเพื่อศึกษาพฤติกรรมของผู้ใช้งานภายในระบบ, การตรวจสอบด้านความปลอดภัยของระบบ, หรือตรวจสอบการโจมตี และพฤติกรรมการฉ้อโกงของมิจฉาชีพ เป็นต้น เนื่องจากในปี พ.ศ.2550 พระราชบัญญัติว่าด้วยการกะทำความผิดเกี่ยวกับคอมพิวเตอร์ ได้กำหนดให้หน่วยงานต่างๆ ทั้งในภาครัฐฯ และเอกชนจะต้องดำเนินการเก็บข้อมูลการจราจรทางคอมพิวเตอร์ (ข้อมูลการใช้งานระบบงานบนเครือข่าย) เป็นระยะเวลา 90 วัน ซึ่งส่งผลให้ทางสำนักงานจัดการระบบเทคโนโลยีสาระสนเทศของจุฬาลงกรณ์มหาวิทยาลัยจะต้องดำเนินการจัดเก็บบันทึกของระบบงาน จากเครื่องแม่ข่ายกว่า 40 เครื่อง โดยในบางกรณีอาจจะมีขนาดของข้อมูลถึง 1 TB (เป็นขนาดภายหลังจากการทำการบีบอัดแล้ว) หรืออาจมีปริมาณของข้อมูลเกิดขึ้น มากกว่า 10 GB ต่อวัน สำหรับการเก็บข้อมูลตามข้อกำหนดของพรบ.ฯ ซึ่งจำนวนของบันทึกของระบบงาน เหล่านี้อาจจะมีขนาดเพิ่มขึ้นได้ เนื่องจากการเพิ่มขึ้นของอุปกรณ์หรือบริการในอนาคต จึงทำให้เกิดปัญหาของ Big Data โดยปัญหาที่เกิดนั้นจะมีสองส่วนด้วยกันคือ Big Data และ Fast Data คำว่า Big Data [1, 2] หรือฐานข้อมูลขนาดใหญ่ ในที่นี้หมายถึง เซตของข้อมูลที่ใหญ่ขึ้นเรื่อยๆ และเกิดขึ้นอย่างรวดเร็วอยู่ตลอดเวลา จนยากที่จะใช้เครื่องมือบริหารที่ทำงานด้วยมือ ตามแบบที่เคยมีมา ส่วน Fast Data นั้นคือ Big Data ที่ถูกสร้างขึ้นด้วยความเร็วสูงเช่น ข้อมูลในการคลิกเข้า web site ต่างๆ, ข้อมูลทางการเงิน, บันทึกของระบบงาน (Log), หรือข้อมูลจาก sensor ต่างๆ

ในการทำงานของผู้ดูแลระบบของสำนักเทคโนโลยีสารสนเทศ ซึ่งจะต้องแก้ปัญหาทางเทคนิคให้แก่ผู้ใช้งาน จำเป็นอย่างยิ่งที่จะต้องใช้ข้อมูลบันทึกของระบบงาน ที่เก็บมาจากเครื่องแม่ข่ายต่างๆ ซึ่งเป็นข้อมูลประเภท Big Data ที่เก็บไว้ และ Fast Data ที่เข้ามาใหม่มาประมวลผลแบบ Real-Time อย่างไรก็ตามเนื่องจากปริมาณของข้อมูลบันทึกของระบบงาน ที่มีขนาดใหญ่

ดังกล่าว ทำให้ระบบการจัดเก็บบันทึกของระบบงาน ในปัจจุบันไม่สามารถตอบสนองความต้องการในการประมวลผลและเข้าถึงข้อมูลได้อย่างมีประสิทธิภาพ อีกทั้งการนำข้อมูลที่จัดเก็บไว้มาทำการวิเคราะห์เพื่อหาต้นตอของปัญหา (Root Cause) หรือความผิดปกติของระบบ (Failure) ก็กระทำได้ยาก ต้องใช้เวลาในการค้นหา และการคัดกรองข้อมูลเป็นเวลานาน ทำให้เกิดข้อจำกัดในการทำงานเป็นอย่างมาก

จากงานวิจัยที่ผ่านมาเราพบว่าระบบการประมวลผลบันทึกของระบบงาน ส่วนใหญ่นั้นจะเป็นการประมวลผลแบบ Batch [3-6] ซึ่งจะเน้น Throughput ของระบบ ทำให้สามารถประมวลผลข้อมูลที่มีปริมาณมากได้ แต่ว่าการประมวลผลแบบ Batch นั้นต้องใช้ระยะเวลาในการประมวลผลที่ค่อนข้างสูง ทางผู้จัดทำจึงได้นำเทคโนโลยี Hadoop ซึ่งเป็น platform ของการประมวลผลแบบกระจายที่มีความนิยม และสามารถทำให้ความเร็วในการประมวลผลแบบ Batch เพิ่มมากขึ้น มาใช้ในการแก้ปัญหาและจัดการกับข้อมูลประเภทบันทึกของระบบงาน หรือ Big Data [7-10] โดย Hadoop Distributed File System (HDFS) ใน Hadoop นั้นจะช่วยในเรื่องของการจัดเก็บข้อมูลที่มีปริมาณมากโดยที่ HDFS นั้นจะมีการทำสำรองข้อมูลอัตโนมัติ (Replicate) เพื่อเพิ่มความทนทานต่อความสูญเสียของข้อมูล (Fault Tolerance) อยู่ด้วย นอกจากนั้น Hadoop ยังมี MapReduce Framework ที่สามารถประมวลผลข้อมูลขนาดใหญ่ที่เก็บไว้แบบ parallel ได้ ทำให้สามารถประมวลผลข้อมูลได้อย่างรวดเร็ว ถึงแม้ว่าเราจะนำ Hadoop มาใช้แล้วก็ตาม การประมวลผลแบบ Batch อย่างเดียวนั้นก็ยังคงไม่เพียงพอต่อความต้องการของสำนักเทคโนโลยีสารสนเทศ ทำให้เราต้องนำการประมวลผลแบบ Real-Time มาใช้ร่วมกัน แต่การประมวลผลแบบ Real-Time ที่มีมานั้น [11-13] จะเน้นการประมวลผลให้ทันเวลาโดยที่จะประมวลผลข้อมูลที่เข้ามาให้ได้ผลลัพธ์ออกมาด้วยอัตราที่ที่เร็วพอๆ กับอัตราที่ข้อมูลไหลเข้ามาในระบบ อีกทั้งการประมวลผลบันทึกของระบบงาน แบบ Real-Time ส่วนใหญ่นั้นจะประมวลผลบันทึกของระบบงานก่อนแล้วนำผลลัพธ์นั้นๆ มาจัดเก็บลงฐานข้อมูลทำให้ไม่ต้องเก็บข้อมูลดิบหรือข้อมูลเก่าที่มีปริมาณมาก จึงทำให้การประมวลผลแบบ Real-Time ส่วนใหญ่ ไม่สามารถทำการประมวลผลข้อมูลย้อนหลังได้เหมือการประมวลผลแบบ Batch นอกจากนี้ถึงแม้ว่าจะมีระบบงานที่รองรับการบริหารจัดการบันทึกของระบบงาน ขนาดใหญ่อย่างมีประสิทธิภาพเช่น Splunk, Loggy หรือว่า ArcSight Logger แต่ก็เป็นระบบงานแบบ Commercial ซึ่งมีราคาที่ค่อนข้างสูงและมีปัญหาในเรื่องการขยายของข้อมูล (data scale) เพราะต้องรับมือกับข้อมูลที่มีปริมาณมาก

ขนาดของบันทึกของระบบงาน ที่ส่งมาจากเครื่องแม่ข่ายแต่ละเครื่องนั้นมีขนาด (Volume) และความถี่ในการส่ง (Velocity) ไม่คงที่ โดยส่วนใหญ่แล้วจะเป็นไฟล์ที่มีขนาดเล็ก [14] ซึ่งไม่

เหมาะที่จะนำมาจัดเก็บบน HDFS เพราะว่า HDFS นั้นถูกออกแบบมาเพื่อจัดเก็บข้อมูลที่มีขนาดใหญ่ โดยที่ HDFS นั้นจะแบ่งข้อมูลขนาดใหญ่ออกเป็น block ของข้อมูลที่มีขนาดเล็กแล้วเก็บลงบน HDFS แต่ถ้าเราจัดเก็บข้อมูลที่มีขนาดเล็กลงไปซึ่งจะไม่เต็ม block ส่งผลให้เราใช้ทรัพยากรของเครื่องอย่างไม่มีประสิทธิภาพ อีกทั้งยังส่งผลให้มีจำนวนของ block มากเกินความจำเป็น เนื่องจาก HDFS นั้นมีโครงสร้างการทำงานแบบ single-master-multiple-slave โดยจะประกอบไปด้วย NameNode กับ DataNode โดยที่การทำงานทั้งหมดนั้นจะต้องทำงานผ่าน NameNode ซึ่งทำหน้าที่เป็น Master Node ดังนั้นบน NameNode จึงมีการจัดเก็บ metadata เอาไว้เพื่อคอยให้บริการกับผู้ใช้เมื่อมีการร้องขอ โดยที่ matadata บน NameNode นั้นจะถูกจัดเก็บไว้บนหน่วยความจำเพื่อความเร็วในการทำงาน จึงทำให้เกิดปัญหาคอขวด หรือ Bottle Neck ตามมา กล่าวคือ เมื่อมีการร้องขอการใช้งานจำนวนมาก จะทำให้ NameNode ต้องรับการร้องขอเกี่ยวกับการจัดเก็บ หรือ การกระจาย block บ่อยๆ ส่งผลให้ NameNode ทำงานหนักและอาจจะทำให้ NameNode นั้นไม่สามารถทำงานได้อย่างมีประสิทธิภาพเท่าที่ควร นอกจากนั้นพื้นที่ที่ใช้เก็บข้อมูลบน NameNode ก็มีขีดจำกัดตาม physical mamory อีกด้วย เพาะว่า HDFS นั้นจัดเก็บ metadata เอาไว้ใน memory ของ NameNode ดังนั้น ยิ่งมีจำนวนไฟล์จัดเก็บในระบบมากเท่าไหร่ก็ยิ่งต้องใช้พื้นที่ในการจัดเก็บบน mamory ของ namenode มากเท่านั้น

ดังนั้นระบบการจัดการบันทึกของระบบงาน ที่ดีนั้นควรจะต้องรองรับการทำงานกับข้อมูลที่มีลักษณะเป็น Fast และ Big Data โดยที่ข้อมูลที่จัดเก็บลงบนระบบจะต้องมีความคงทนต่อความเสียหายหรือศูนย์หาย และระบบการจัดการบันทึกของระบบงานนั้นจะต้องมีความสามารถในการประมวลผลข้อมูลทั้งในรูปแบบของ Batch และ Real-Time เพราะว่างานแต่ละประเภทนั้นมีความต้องการไม่เหมือนกัน เช่น งานบางประเภทมีความต้องการที่จะประมวลจากข้อมูลเก่าหรือข้อมูลดิบที่เราเก็บเอาไว้เท่านั้น, งานบางประเภทมีความต้องการที่จะประมวลผลข้อมูลที่เข้ามาเพื่อให้ได้คำตอบอะไรบางอย่างทันที, หรืองานบางประเภทที่มีความต้องนำคำตอบจากการประมวลผลข้อมูลที่เก็บเอาไว้มาประมวลผลร่วมกับกับการประมวลผลข้อมูลที่เข้าใหม่ เป็นต้น

จากปัญหาที่กล่าวมาข้างต้น งานวิจัยนี้จึงได้มุ่งเน้นการพัฒนาสถาปัตยกรรมที่เหมาะสมแก่การประมวลผลบันทึกของระบบงานขนาดใหญ่ โดยอ้างอิงจากสถาปัตยกรรมแบบ Three-Lambda [15] เพื่อรองรับการทำงานแบบ Real-Time และ Batch ที่มีโครงสร้างของข้อมูลเชิง Big data ซึ่งมีความซับซ้อนและมีขนาดใหญ่ โดยจะเป็นการนำเทคโนโลยี HDFS ใน Hadoop และ In-memory Processing เช่น Spark เข้ามาต่อยอด

1.2 วัตถุประสงค์ของงานวิจัย

1. เพื่อพัฒนากระบวนการบริหารจัดการบันทึกของระบบงาน ขนาดใหญ่ที่สามารถประมวลผลได้ทั้งแบบ Batch และแบบ Real-Time
2. เพื่อพัฒนาระบบที่สามารถเก็บข้อมูลที่มีความคงทน และสามารถขยายขนาดเพื่อรองรับการเติบโตของข้อมูลได้

1.3 ขอบเขตของงานวิจัย

1. สามารถประมวลผลได้ทั้งในรูปแบบ Batch และ Real-Time
2. ตรวจสอบความผิดปรกติของการใช้งานอินเทอร์เน็ตของจุฬาลงกรณ์มหาวิทยาลัย
3. ตรวจสอบความผิดปรกติของการใช้งานอินเทอร์เน็ตจากการพยากรณ์ปริมาณการใช้งานโดยใช้แบบจำลองอนุกรมเวลาซาริมา (SARIMA)

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1. ได้เรียนรู้รูปแบบการทำงานของ HDFS
2. ได้ต้นแบบของสถาปัตยกรรมที่ใช้ในการจัดการกับบันทึกของระบบงาน ในโครงสร้างการทำงานเชิง Big data และสามารถรองรับการขยายของข้อมูลได้
3. นำเสนอวิธีการตรวจหาความผิดปรกติของการใช้งานเครือข่าย จากบันทึกของระบบงาน โดยใช้แบบจำลองอนุกรมเวลาซาริมา
4. สามารถนำความรู้จากผลการวิจัยนี้ไปประยุกต์ใช้จริงต่อไปในอนาคต

1.5 แผนการดำเนินงานวิจัย

1. ศึกษางานวิจัยที่เกี่ยวข้อง

   ทำการศึกษางานวิจัยที่เกี่ยวข้องเช่น โครงสร้างระบบการประมวลผลบันทึกของระบบงานที่เคยมีมา, ต้นแบบสถาปัตยกรรมที่เราสนใจนำมาปรับใช้ เป็นต้น
2. ออกแบบวิธีการ และการเขียนโปรแกรม

   ทำการออกแบบสถาปัตยกรรมที่ใช้สำหรับการจัดเก็บและประมวลผลบันทึกของระบบงาน
3. ทดสอบโปรแกรม และปรับปรุงวิธีการ

นำสถาปัตยกรรมไปใช้ในการประมวลผลของบันทึกของระบบงาน โดยใช้วิธี SARIMA ในการทดสอบสถาปัตยกรรม เพื่อทดสอบประสิทธิภาพการทำงาน โดยใช้ตัวชี้วัดที่นำเสนอ

4. สรุปผลการทดลอง และจัดทำวิทยานิพนธ์

นำผลการทดลองที่ได้มาวิเคราะห์ เปรียบเทียบ และสรุปผล โดยพิจารณาจากสถาปัตยกรรมที่ได้ออกแบบว่า มีความสามารถในการจัดการกับบันทึกของระบบงานได้มี ประสิทธิภาพมากน้อยเพียงใด

1.6 ผลงานตีพิมพ์

ส่วนหนึ่งของวิทยานิพนธ์ฉบับนี้ ได้นาเสนอในการประชุมวิชาการ ดังนี้

Pittayut Tangsatjatham and Natawut Nupairoj, “Hybrid Big Data Architecture for High-Speed Log Anomaly Detection”, The 13th International Joint Conference on Computer Science and Software Engineering (JCSSE 2016) , Khon Kaen, Thailand, July 13-15, 2016

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 บันทึกของระบบงาน (Log)

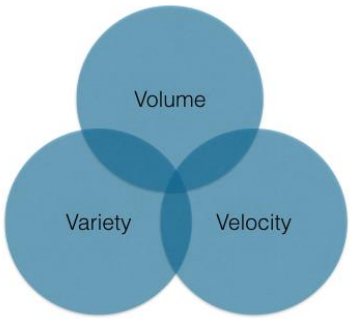
บันทึกของระบบงาน คือ ข้อมูลที่ถูกบันทึกไว้ตามลำดับเหตุการณ์ที่เกิดขึ้นบนอุปกรณ์คอมพิวเตอร์ หรือในบริการต่างๆ เพื่อที่จะสื่อถึงพฤติกรรมต่างๆ ที่เกิดขึ้นทั้งเหตุการณ์ที่ปกติ และไม่ปรกติที่เกิดขึ้นกับอุปกรณ์หรือ บริการนั้นๆ โดยปกติแล้วการสร้างบันทึกของระบบงานนั้นถูกใช้งานอย่างกว้างขวางทั้งใน Application, Operation system, Control system, Mobile devices, Super computer, รวมถึง Application component ดังนั้นเมื่อมีสิ่งที่ผิดปกติเกิดขึ้นกับระบบของเรา บันทึกของระบบงานสามารถเป็นตัวแทนที่สื่อถึงพฤติกรรมของระบบ และสามารถนำไปวิเคราะห์หา failure ในระบบได้ โดยที่บันทึกของระบบงานนั้นจะจัดเก็บในรูปแบบที่ไม่เหมือนกัน รูปแบบที่นิยมใช้คือ Syslog format ซึ่งเป็นรูปแบบมาตรฐานหนึ่งของบันทึกของระบบงานโดยมีการบันทึกวันและเวลาของเหตุการณ์ที่เกิดขึ้น

```
Sep 26 22:09:24 ubuntu ntpd[1589]: Listen normally on 3 eth0 192.168.86.133 UDP 123
Sep 26 22:09:24 ubuntu ntpd[1589]: peers refreshed
Sep 26 22:09:24 ubuntu ntpd[1589]: Listening on routing socket on fd #20 for interface updates
Sep 26 22:09:24 ubuntu ntpd[1589]: Deferring DNS for 0.ubuntu.pool.ntp.org 1
Sep 26 22:09:24 ubuntu ntpd[1589]: Deferring DNS for 1.ubuntu.pool.ntp.org 1
Sep 26 22:09:24 ubuntu ntpd[1589]: Deferring DNS for 2.ubuntu.pool.ntp.org 1
Sep 26 22:09:24 ubuntu ntpd[1589]: Deferring DNS for 3.ubuntu.pool.ntp.org 1
Sep 26 22:09:24 ubuntu ntpd[1589]: Deferring DNS for ntp.ubuntu.com 1
Sep 26 22:09:29 ubuntu ntpd_intres[859]: parent died before we finished, exiting
Sep 26 22:09:51 ubuntu ntpd_intres[1530]: parent died before we finished, exiting
Sep 26 22:10:30 ubuntu dhclient: Internet Systems Consortium DHCP Client 4.2.4
Sep 26 22:10:30 ubuntu dhclient: Copyright 2004-2012 Internet Systems Consortium.
Sep 26 22:10:30 ubuntu dhclient: All rights reserved.
Sep 26 22:10:30 ubuntu dhclient: For info, please visit https://www.isc.org/software/dhcp/
```

ภาพที่ 1 ตัวอย่างไฟล์ของบันทึกของระบบงาน

จากภาพที่ 1 เป็นตัวอย่างของบันทึกของระบบงานซึ่งปรกติแล้วจะประกอบไปด้วยข้อมูลต่างๆ เช่น วัน-เวลา, IP address หรือชื่อของ node ที่สร้างบันทึกของระบบงาน, และ message ซึ่งเป็นข้อมูลที่บอกถึงเหตุการณ์ปรกติ (regular) ไปจนถึง Error ของระบบ โดยที่เหตุการณ์ปรกติจะบอกถึงพฤติกรรมที่ไม่มีการ error เช่น Disk mount, Network status, และ User connection ส่วน error จะบอกเกี่ยวกับปัญหาของ Hardware, Software หรือการทำงานที่ผิดพลาดและไม่สำเร็จของอุปกรณ์ หรือบริการนั้นๆ

2.2 Big Data



ภาพที่ 2 3Vs of Big Data

Big data หรือฐานข้อมูลขนาดใหญ่ ในที่นี้หมายถึง เซตของข้อมูลที่ใหญ่มากขึ้นเรื่อยๆ และเกิดขึ้นอย่างรวดเร็ว จนยากแก่การใช้เครื่องมือแบบเก่าๆ ที่เคยมีมารวมถึงสภาพแวดล้อมในระบบที่มีความแตกต่างกันและสามารถเปลี่ยนแปลงได้ตลอดเวลาจนทำให้เกิดความซับซ้อนมากขึ้นเรื่อยๆ ภายในระบบ โดยที่ขนาด, ความซับซ้อน, การจัดเก็บ, การค้นหา, รวมถึงการวิเคราะห์ เป็นปัญหาที่ยากต่อการจัดการในสภาพแวดล้อมที่เป็น Big data คือ ปริมาณ (Volume), ความเร็ว (velocity), และความหลากหลาย (variety) [1, 2]

1. ข้อมูลมีปริมาณมาก (Volume) : องค์กรต่างๆ มีการสร้างข้อมูลมากขึ้นเรื่อยๆ จนถึงขนาดเทราไบต์ ไม่ว่าจะเป็นการจัดเก็บข้อมูลที่มีความซับซ้อนมากขึ้น หรือการรับข้อมูลเข้ามา จนกระทั่งมีข้อมูลจำนวนมาก ทำให้เกิดความยากลำบากในการเข้าถึงข้อมูล เนื่องจากต้องเสียเวลากับการไล่ลำดับการเข้าถึงข้อมูล
2. ข้อมูลถูกสร้างขึ้นด้วยความเร็ว (Velocity) : การที่ข้อมูลมีการไหลเข้ามาในระบบตลอดเวลา จำเป็นจะต้องใช้ประโยชน์จากข้อมูลเหล่านี้ให้ ได้มากที่สุดในลักษณะ Real time เพื่อวิเคราะห์ข้อมูลได้ถูกต้องและรวดเร็วมากยิ่งขึ้น
3. ความหลากหลายข้องข้อมูล (Variety) : ข้อมูลมีความหลากหลายจึงยากต่อการนำข้อมูลมาใช้เพื่อประมวลผล อาจเป็นข้อมูลชนิดใดก็ได้ ทั้งมีโครงสร้างและไม่มีโครสร้าง เช่น ข้อความ, ข้อมูลจากเซนเซอร์, วีดีโอ, เสียง, ข้อมูลบันทึกของระบบงาน เป็นต้น

2.3 Resilient Distributed Datasets (RDDs)

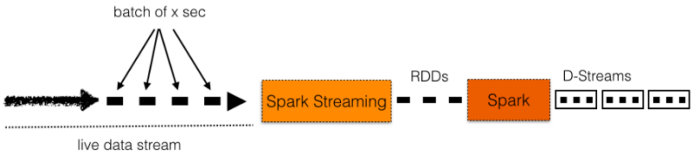
Resilient Distributed Datasets หรือ RDDs [16] เป็นสิ่งที่ช่วยให้โปรแกรมเมอร์สามารถเขียนเขียนโปรแกรมแบบขนานเพื่อประมวลผลบนหน่วยความจำ (In memory parallel programing) ของระบบคลัสเตอร์ได้ ซึ่งการสร้าง RDDs นั้น ทำได้โดยการนำข้อมูลจากภายนอกมาผ่าน parallel operations ต่างๆ โดยที่ parallel operations นั้นสามารถแบ่งออกได้เป็นสองแบบด้วยกันคือการ transformation ซึ่งเป็น operation ที่ใช้ในการสร้าง dataset และ action ซึ่งใช้ในการสั่งคำสั่งให้ประมวลผลข้อมูลบนคลัสเตอร์

RDDs นั้นเหมาะสมอย่างยิ่งกับการนำไปใช้ใน batch application ที่ทุกๆ datasets จะถูกกระทำโดยใช้คำสั่งเดียวกันในการประมวลผล และเนื่องจากการ transformation นั้นเป็นการแบ่งข้อมูลออกเป็นชิ้นเล็กๆ แล้วนำมาจัดกลุ่มเพื่อกระจายออกไปตามเครื่องต่างๆ ในคลัสเตอร์ โดยข้อมูลเหล่านี้จะถูกจัดเก็บไว้บนหน่วยความจำ ดังนั้นยิ่งหน่วยความจำมีขนาดของใหญ่เท่าไหร่ประสิทธิภาพในการประมวลผลก็จะยิ่งสูงมากเท่านั้น นอกจากนั้นการที่ RDDs ใช้การ transformation data ในการสร้าง datasets ทำให้ RDDs มีความสามารถในการ recovery partition ที่หายไปโดยไม่จำเป็นที่จะต้องเก็บข้อมูลทั้งหมด (fault-tolerant) โดยข้อมูลที่ได้รับความเสียหายนั้นจะถูก สร้างขึ้นมาใหม่ จากการ transformation แทน

ภาพที่ 3 RDDs Transformation

2.4 Discretized Streams (D-Stream)

Discretized Streams หรือ D-Stream [17] เป็น stream programming model ที่ใช้สำหรับระบบการกระจายขนาดใหญ่ ซึ่งมีความต้องกันของข้อมูล (Consistency) และมีความสามารถในการทำ fault recovery และยังสามารถทำงานรวมกับระบบ batch ได้อย่างลงตัว โดยมุ่งเน้นการจัดการกับข้อมูลที่เข้ามาในระบบ (Streaming Computation) ในรูปแบบชุดของ batch jobs ขนาดเล็ก (Batch Computation) และลดเวลาในการประมวลผลกับงานเหล่านี้ให้มากที่สุดเท่าที่จะมากได้ ซึ่งการทำแบบนี้จะส่งผลให้การทำ batch processing สามารถทำ stream processing ได้



ภาพที่ 4 D-Streams Operation

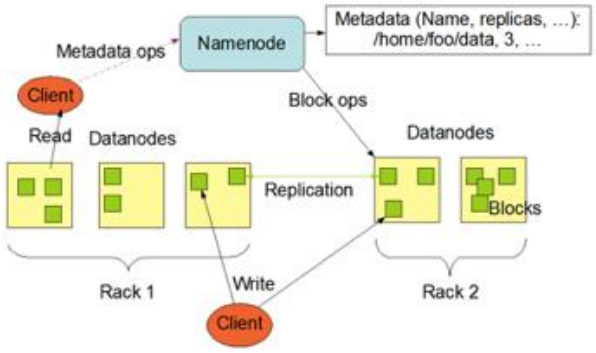
2.5 Apache Hadoop [18, 19]

Hadoop Distributed File System (HDFS) นั้นคือ ระบบการจัดเก็บข้อมูลแบบกระจายที่ออกแบบมาเพื่อทำงานบนระบบ commodity hardware ซึ้งคล้ายคลึงกับระบบการจัดเก็บข้อมูลแบบกระจายอื่นๆ แต่มีความแต่ต่างตรงที่ HDFS นั้นสามารถทนต่อการสูญเสียสูง (Highly fault-tolerant) และออกแบบมาเพื่อรองรับการทำงานของอุปกรณ์ที่มีราคาไม่สูง (low-cost hardware) ซึ่งเหมาะที่จะนำมาใช้ในงานของเรา โดยที่ HDFS นั้นมีความเหมาะสมสำหรับ application ที่มีขนาดใหญ่และสามารถเพิ่มจำนวนโหนดได้ถึงหนึ่งหมื่นโหนด, หนึ่งร้อยล้านไฟล์ และขนาดใหญ่ได้มากถึง 10 petabyte

โดยปรกติแล้ว HDFS เป็นการจัดเก็บข้อมูลแบบ write-once-read-many access model โดยที่ไฟล์จะถูกเขียนเพียงครั้งเดียวและไม่สามารถแก้ไขได้ทำให้เหมาะกับการเก็บข้อมูลประเภทบันทึกของระบบงาน ซึ่งจะทำให้ความสัมพันธ์ต่างๆ ของข้อมูลมีความง่ายมากขึ้น และสามารถเข้าถึงข้อมูลได้แบบ high throughput นอกจากนั้นยังมีการย้ายการประมวลผลให้ไปอยู่ใกล้กับที่เก็บข้อมูล จะทำให้ช่วยลดความคับคั่งของเครือข่าย (Traffic Network) และยังช่วยเพิ่ม throughput ให้กับระบบอีกด้วย

2.5.1 สถาปัตยกรรมของ HDFS

HDFS นั้นจะมีการทำงานแบบ master-slave โดยจะมีโหนดที่สำคัญอยู่สองชนิดคือ NameNode และ DataNode



ภาพที่ 5 สถาปัตยกรรมของ HDFS

NameNode ทำหน้าที่เป็น Master server ที่คอยจัดการกับ file system เช่น การเปิด-ปิดไฟร์, เปลี่ยนชื่อสาระบบ (Directories) นอกจากนั้นยังเป็นตัวที่จัดการกับการแบ่งข้อมูลไปเก็บเอาไว้ใน DataNode ต่างๆ ซึ่งจะถูกแสดงอยู่บน NameNode ในรูปของ inode ที่เก็บข้อมูลเกี่ยวกับ permissions, การแก้ไขต่างๆ, เวลาในการเข้าถึง namespace และ disk space quotas

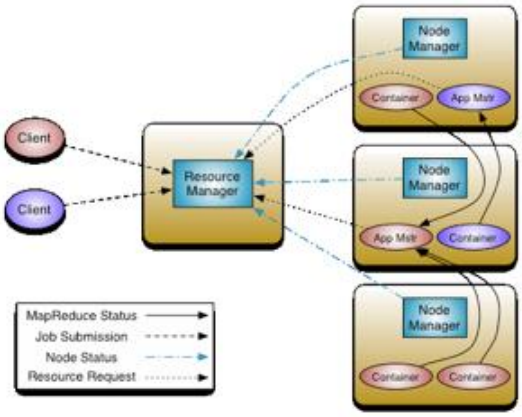
โดยไฟล์จะถูกแบ่งออกเป็น blocks ซึ่งจะถูกทำ replicate ไปจัดเก็บไว้บน DataNode ต่างๆ โดย NameNode จะเก็บ nampspace ทั้งหมดเอาไว้ใน RAM ซึ่งในแต่ละไฟล์จะมี inode data และรายชื่อของ blocks และจะเก็บ metadata ทั้งหมดเอาไว้ใน Namespace image โดยจะถูกจัดเก็บเอาไว้ใน localhost ของ file system ที่เรียกว่า checkpoint นอกจากนั้น NameNode ยังมีการเก็บ Journal ซึ่งเป็นบันทึกของระบบงานที่จะเก็บการเปลี่ยนแปลงต่างๆ ภายในระบบเอาไว้ใน localhost file system และมีการทำ redundant copies ของ journal และ check point ที่ server อื่นๆ เอาไว้ด้วยเพื่อเพิ่มความทนทานของระบบ (Fault Tolerance)

DataNode จะมีในอยู่ในทุกๆ โหนด ภายใน cluster ซึ่งจะคอยจัดการเกี่ยวกับข้อมูล เช่น การสร้าง, การลบ block และการทำ replication ซึ่งรับคำสั่งมาจาก NameNode อีกทีหนึ่ง โดยแต่ละ block replica บน DataNode จะประกอบไปด้วยไฟล์สองไฟล์ โดยไฟล์แรกจะเป็นข้อมูล และไฟล์ที่สอง คือ block ของ metadata ซึ่งรวมถึง checksum ของ block data และ block's generation stamp โดยขนาดของไฟล์ข้อมูล จะเท่ากับความกว้างของ block จริงๆ และไม่มีการร้องขอพื้นที่พิเศษเพื่อใช้ในการปัดให้ครบตามขนาดเหมือน file system แบบดั่งเดิม โดยในระหว่าง startup แต่ละ DataNode จะเชื่อมต่อไปยัง NameNode และทำการ handshake เพื่อยืนยัน namespace ID และ software virsion ของ DataNode หากไม่ตรงกับ NameNode ก็จะทำการ shutdown อัตโนมัติเพื่อไม่ให้ข้อมูลเกิดความเสียหายหรือสุญหาย หลังจากนั้นจะส่ง block report ไปยัง NameNode ซึ่งจะประกอบด้วย block id, generarion stamp และ ความยาวของแต่ละ block replicat โดย block report จะทำการส่งทุกหนึ่งชั่วโมง โดยในระหว่างการทำงานปรกติ DataNode จะสงสัญญาณ heartbeats ไปยัง NameNode เพื่อยืนยันว่า DataNode นั้นยังทำงานอยู่และ block replicas ภายในยังคงทำงานอยู่เช่นเดียวกัน ซึ่งข้อมูลที่ถูกส่งไปใน heartbeats นั้นจะประกอบไปด้วย ความจุ, หน่วยความจำที่ถูกใช้งาน และจำนวนของ data transfer ที่กำลังทำงานอยู่ ซึ่งข้อมูลเหล่านี้จะใช้สำหรับการจองพื้นที่และการตัดสินใจในการทำ load balancing ของ NameNode โดยที่ปกติจะทำ heartbeat ทุกๆ 3 วินาที ถ้าหาก NameNode ไม่ได้รับ heartbeat จาก DataNode ภายใน 10 นาที NameNode ห็จะทำการเปลี่ยนสถานะของ DataNode นั้นให้กลายเป็น out-of –service และจะสร้าง replicas ที่อยู่ใน DataNode นั้นไปไว้ใน DataNode ใหญ่

2.5.2 Hadoop Yarn (MapReduce version 2)

Hadoop Yarn เป็นตัวที่พัฒนามาจาก MapReduce ซึ่ง MapReduce คือ framework ที่ช่วยในการสร้าง application สำหรับประมวลผลข้อมูลที่มีขนาดใหญ่ในลักษณะการประมวลผลแบบขนาน (Pallel Programming) บนระบบการกระจายขนาดใหญ่ (large cluster) ที่มีความน่าเชื่อถือ (reliable) และ การคงทนต่อความเสียหาย (Fault Tolerance)

โดยกาทำงานของ MapReduce นั้นจะแบ่งข้อมูล input ออกเป็น chunks ซึ่ง chunks เหล่านี้จะถูกประมวลผลโดย map task ในรูปแบบของ parallel หลังจากนั้น framework จะนำผลที่ได้จาก map task มาจัดเรียงแล้วนำมาส่งต่อให้กับ reduce task แล้วจึงได้ผลลัพท์ออกมา (output) โดยที่ทั้ง input และ output นั้นจะถูจัดเก็บอยู่บน HDFS โดยที่ framework นั้นจะมีการการทำ scheduling tasks, monitoring task และจักการกับ failed tasks โดยการ re-executes task นั้น



ภาพที่ 6 สถาปัตยกรรมของ Hadoop YARN

จากภาพที่ 6 สถาปัตยกรรมของ Hadoop YARN ประกบด้วย

1. Resource Manager : ประกอบไปด้วย Scheduler กับ Applications Manager โดยที่ Scheduler ทำหน้าที่ในการจัดลำดับการประมวลผลงาน โดยดูจากปริมาณทรัพยากรที่งานนั้นๆ ร้องขอ และ Application Manager เป็นตัวที่รับงานมาจาก client แล้วไปสั่งให้ Application Master ทำงาน
2. Node Manager : จะมีอยู่บนทุกๆ โหนดภายใน cluster เป็นตัวที่คอยดูปริมาณทรัพยากรที่มีอยู่ของโหนดนั้นๆ แล้วรานงานไปยัง Resource Manager รวมถึงเป็นตัว execute และ monitor งานทั้ง map task และ reduce task

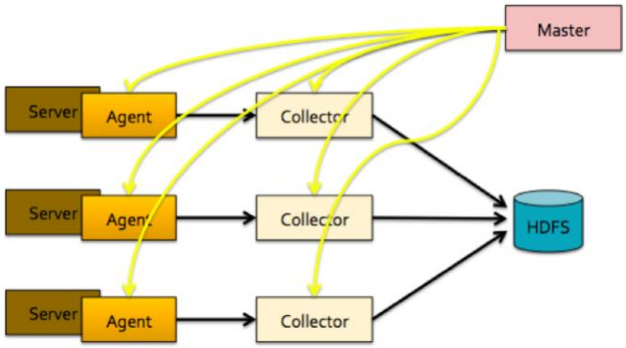
3. Application Master : เป็นตัวที่สร้างและร้องขอทรัพยากรสำหรับการทำงานไปยัง Scheduler บน Resource Manager รวมถึงติดตามและรายงานผลการทำงานไปยัง Resource Manager

2.6 Apache Flume [20]

Apache Flume เป็นโครงการของ Apache Software Foundation โดยจะนำเสนอการผสมผสานการทำงานของ Hadoop และข้อมูลที่มีลักษณะเป็น Streaming Data สำหรับการเก็บรวบรวมข้อมูลจากต้นทางหลายๆ แหล่งและส่งต่อไปยังศูนย์กลางที่เก็บข้อมูลเช่น Hadoop Distributed File System (HDFS) ด้วยความรวดเร็ว เพื่อที่จะนำข้อมูลเหล่านั้นมาวิเคราะห์ หรือตรวจสอบพฤติกรรมและความผิดปกติในการทำงานที่เกิดขึ้น โดยที่ Apache Flume นั้นมีทั้งความเป็น Distributable, Reliable และ Available service ของการรวบรวมบันทึกของระบบงาน ได้อย่างมีประสิทธิภาพและยังมีความสามารถในการเคลื่อนย้ายบันทึกของระบบงาน ที่มีขนาดใหญ่ โดยสามารถใช้ได้ทั้ง Simple และ Flexible architecture ที่เป็นแบบ Streaming dataflow

2.6.1 สถาปัตยกรรมของ Apache Flume

สถาปัตยกรรมของ Flume จะมีลักษณะ Stream-oriented data flow โดยจะมีเส้นทางส่งต่อไปยังปลายทาง และประกอบไปด้วย Logical node ที่สามารถส่งต่อหรือรวบรวม event ที่ได้รับมา ซึ่งสามารถตั้งค่าได้ที่ Flume master



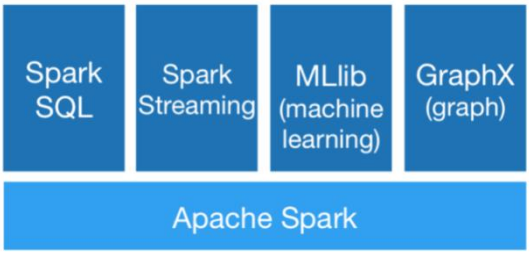
ภาพที่ 7 สถาปัตยกรรมของ Apache Flume

รูปที่ 6 แสดงถึงตัวอย่างการใช้งานของ Flume ในการจัดการกับบันทึกของระบบงาน จาก Server ซึ่งแบ่งออกเป็นสามส่วนคือ Agent, Collector และ Storage โดย Agent จะถูกติดตั้งบน node แต่ละ node ที่มีการสร้างบันทึกของระบบงาน Agent จะทำการส่งบันทึกของระบบงานไปยังส่วนต่อไปคือ Collector ซึ่งเป็นส่วนที่ติดตั้งลงบนเครื่องที่จะรวบรวมข้อมูลต่างๆ ของแต่ละ

data flow แล้วส่งต่อไปยัง storage ซึ่งเป็นส่วนสุดท้าย เช่น Agent อาจจะเก็บข้อมูล sysLog แล้วส่งข้อมูลต่อไปยัง Collector จากนั้น collector ก็จะรวบรวมข้อมูลจากทุกๆ agent และทำการ stream ข้อมูลต่อไปยังที่เก็บข้อมูล เช่น HDFS

2.7 Apache Spark [19, 21]

Apache Spark เป็น framework สำหรับการทำ large-scale data processing โดยหมายของ spark นั้นคือการเพิ่มความเร็วในการทำงานของ batch processing, การทำงานแบบ iterative ใน machine learning, การทำ interactive query และการทำ graph processing โดยที่ spark สามารถประมวลผลได้เร็วว่า hadoop mapreduce ถึง 100 เท่า และสามารถพัฒนาได้ง่ายโดยใช้ภาษา Java, Scala หรือว่า Python จุดเด่นของ Spark อยู่ที่ Resilient Distributed Datasets หรือ RDDs ซึ่งช่วยให้ Spark นั้นมีความทนต่อความเสียหาย (Fault Tolerance) นอกจากนี้ Spark ยังมี high-level APIs อย่าเช่น Java, Scala และ Python อีกทั้งยังมีเครื่องมือมาให้อย่างในรูปที่ 8



ภาพที่ 8 Spark Framework

SparkSQL: ใช้สำหรับทำ SQL และ data processing

Spark Streaming: ใช้ในการทำ streaming data

MLlib: เป็น library ที่ใช้ในการทำ machine learning

GraphX: ใช้สำหรับทำ graph processing

2.8 การแก้ปัญหาไฟล์ขนาดเล็กบน Hadoop

โดยปรกติขนาดของ block ใน HDFS นั้นจะมีขนาด 128MB (สามารถปรับเปลี่ยนได้) แต่ระบบที่ทางผู้วิจัยได้พัฒนาขึ้นนั้นจำเป็นที่จะต้องรับมือกับข้อมูลที่มีขนาดต่ำกว่า 128MB ซึ่งเป็นข้อมูลที่มีขนาดเล็ก โดยถ้าหากเราจัดเก็บไฟล์ขนาดเล็กเหล่านี้เอาไว้บน HDFS จะทำให้การทำงานภายใน HDFS นั้นไม่มีประสิทธิภาพ เนื่องจาก metadata นั้นจะถูกจัดเก็บเอาไว้บน physical memory ของ NameNode ดังนั้นยิ่งมีไฟล์ขนาดเล็กมากเท่าใด เราก็จะเสียพื้นที่ในการจัดเก็บบน NameNode มากเท่านั้น ถ้าหากเก็บไฟล์ขนาดเล็กมากเกินไป ก็จะส่งผลให้ไม่เหลือ

พื้นที่บน NameNode เลย ปัญหาไฟล์ขนาดเล็กนั้นไม่ได้เกิดขึ้นเฉพาะกับ HDFS เท่านั้น Zhang, Xiaoxue [22] ได้อธิบายวิธีในการแก้ไขปัญหาการไฟล์ขนาดเล็กบนระบบการเก็บข้อมูลแบบกระจาย (Distributed File System) โดยสามารถทำได้ 4 รูปแบบด้วยกันคือ การจัดการกับ metadata ส่งผลให้ขนาดของ metadata ลดลง, การทำ performance optimization เป็นการนำเทคนิคในการทำ cashing เข้ามาช่วยเพื่อเพิ่มประสิทธิภาพในการเข้าถึงไฟล์, การนำไฟล์ขนาดเล็กมารวมกันให้กลายเป็นไฟล์ขนาดใหญ่ ทำให้ช่วยลด metadata ให้น้อยลง และการเก็บไฟล์ขนาดเล็กบนพื้นที่ที่แยกจากระบบกระจาย ดังในตารางที่ 1

method	illustration	advantage	disadvantage
Metadata Management	Metadata compression reduces metadata size	Improve space utilization	the metadata read performance was hurt due to extra steps of lookup file .
Performance Optimization	Utilizing prefetching and caching technologies to improve access efficiency e.g. Hot Files Caching, Metadata Caching	The improvement of the Cache hit ratio	Just for particular application
Small file merging	A set of correlated files is combined into a single large file to reduce the file count. An indexing mechanism has been built to access the individual files from the corresponding combined file.	Reduce the metadata	Two indexes, affect the speed
sequence files	Form by a series of binary, where key is the name of the file, the value of the file content.	Free access for small files, nor restrict how much users and files	Platform dependent
Way to store	Small files stored separately in separate areas	Reduce disk fragmentation	Complexity of the movement

ตารางที่ 1 ตารางแสดงการแก้ไขปัญหาการไฟล์ขนาดเล็กบนระบบการเก็บข้อมูลแบบกระจาย

ทาง Apache Hadoop เองก็ได้เสนอ Hadoop Archive (HAR) ซึ่งเป็นวิธีการจัดการกับไฟล์ขนาดเล็กโดยการรวมไฟล์ขนาดเล็กให้กลายเป็น ไฟล์ขนาดใหญ่ และยังสามารถเข้าถึงไฟล์แบบ parallel transparently ได้อย่างมีประสิทธิภาพ แต่ถึงแม้ว่าการใช้งาน HAR นั้นจะสามารถลดขนาดของ metadata ลงได้ก็ตาม แต่การใช้งาน HAR นั้นก็ทำให้เพิ่ม overhead ที่ไม่จำเป็น จึงทำให้ประสิทธิภาพในการเข้าถึงช้ากว่าการเข้าถึงไฟล์ใน HDFS ปรกติ นอกจากนั้น Apache Hadoop ยังมีวิธีการแก้ปัญหาอีกวิธีหนึ่งคือ การเพิ่ม NameNode ในระบบเพื่อเพิ่มความสามารถในการรองรับ load ของระบบ โดยที่ HDFS Federation นั้นจะเพิ่มจำนวนของ NameNode โดยที่ NameNode แต่ละตัวนั้น จะทำงานเป็นอิสระจากกันโดยแต่ละ DataNode จะต้องทำการ register กับ NameNode ทั้งหมดใน cluster และต้องส่ง heartbeat และ block report ให้กับ NameNode ทุกตัวรวมถึงคอยรับคำสั่งจาก NameNode แต่ปัญหาคือต้องทำการแก้ไขระบบภายในให้รองรับการทำงานที่มี master หลายตัว อีกทั้งยังสิ้นเปลืองทรัพยากรเนื่องจากต้องเพิ่มจำนวนเครื่องในระบบ

2.9 แบบจำลองอนุกรมเวลาซาริมา (SARIMA)

แบบจำลองอนุกรมเวลาซารีมา (SARIMA) หรือ Seasonal Autoregressive Integrate Moving Average เป็นแบบจำลองที่เพิ่มส่วนของ Seasonal Part เข้าไปในในแบบจำลองอนุกรมเวลาอารีมา (ARIMA) โดยสามารถเขียนได้ในรูป $ARIMA(p,d,q) \times (P,D,Q)_s$ ซึ่ง

p, d, q, P, D, Q, และ s หมายถึงอันดับของ Autoregressive, Differencing, Moving average, Seasonal autoregressive, Seasonal differencing, Seasonal moving average , และ Seasonal period ตามลำดับ สมการของ $ARIMA(p,d,q) \times (P,D,Q)_s$ คือ

$$\phi_p(B)\Phi_p(B^s)\nabla^d\nabla_s^D x_t = \theta_q(B)\Theta_Q(B^s)e_t \qquad (1)$$

จากสมการที่ 1 ตัวแปรต่างๆ ของสมการมีดังนี้

- B เป็นตัวดำเนินการ Backward-shift ซึ่ง
  - $BX_t = X_{t-1}$
  - $B^s X_t = X_{t-s}$
- $\nabla = 1 - \mathrm{B}$ เป็นตัวดำเนินการ Differencing ซึ่ง
  - $\nabla^d = (1 - B^d)$
  - $\nabla_s^D = (1 - B^s)^D$
- ตัวดำเนินการ autoregressive
  - $\phi_p(B) = 1 - \phi_1 B - \phi_2 B^2 - \cdots - \phi_p B^p \ , p \geq 0$
- ตัวดำเนินการ moving average
  - $\theta_q(B) = 1 - \theta_1 B - \theta_2 B^2 - \cdots - \theta_q B^q \ , q \geq 0$
- ตัวดำเนินการ seasonal autoregressive
  - $\Phi_P(B) = 1 - \Phi_1 B^{1s} - \Phi_2 B^{2s} - \cdots - \Phi_P B^{Ps} \ , P \geq 0$
- ตัวดำเนินการ seasonal moving average
  - $\Theta_P(B) = 1 - \Theta_1 B^{1s} - \Theta_2 B^{2s} - \cdots - \Theta_Q B^{Qs} \ , Q \geq 0$

2.10 Akaike information criterion (AIC)

AIC คือค่าการวัดความเหมาะสมของแบบจำลองทางสถิติที่ถูกประมาณโดยใช้แนวคิดของเอนโทรปีเพื่อวัดการสูญหายของข้อมูลเมื่อใช้แบบจำลองนั้นโดยแบบจำลองที่มีค่า AIC น้อยที่สุดคือแบบจำลองที่ดีที่สุด

2.11 Bayesian information criterion (BIC)

BIC คือค่าบรรทัดฐานในการเลือกแบบจำลองภายในกลุ่มของแบบจำลองที่มีการปรับค่าตัวแปรต่างๆ โดยค่า BIC นั้นมีความเกี่ยวข้องกับค่า AIC มาก แต่ว่าค่า BIC นั้นจะมีการให้โทษสำหรับตัวแปรเพิ่มขึ้นมากกว่า

2.12 Maximum likelihood estimation (MLE)

MLE คือวิธีการทางสถิติที่นิยมใช้งานการปรับแบบจำลองทางสถิติให้เหมาะสมกับข้อมูลนั้นๆ รวมถึงหาค่าประมาณของตัวแปลต่างๆ ในแบบจำลองทางสถิติ โดย MLE จะเลือกค่าตัวแปรที่ทำให้ฟังก์ชั่นภาวะน่าจะเป็น (Likelihood function) มีค่ามากที่สุด

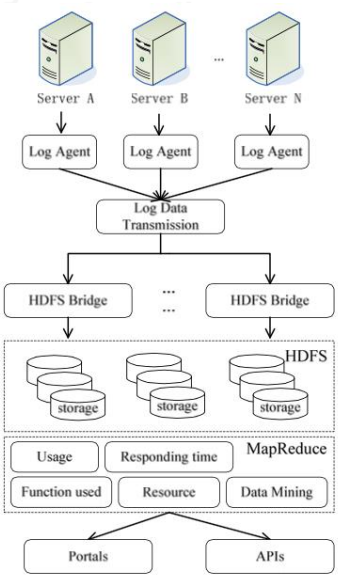
2.13 งานวิจัยที่เกี่ยวข้อง

2.13.1 โครงสร้างระบบประมวลผลบันทึกของระบบงาน

ในบันทึกของระบบงาน จะประกอบด้วยข้อมูลที่มีประโยชน์ซ่อนอยู่ โดยระบบการประมวลผลบันทึกของระบบงาน นั้นเป็นเครื่องมือที่ใช้ในการดึงเอาข้อมูลที่เป็นประโยชน์เหล่านั้นออกมา อย่างเช่น การตรวจหาปัญหาภายในระบบ, การตรวจสอบพฤติกรรมการใช้งานของผู้ใช้, และ การตรวจสอบสถานะของระบบ เป็นต้น [3, 5, 6] เป็นตัวอย่างวิธีการที่ใช้ในการประมวลผลบันทึกของระบบงาน โดยนำเทคนิคต่างๆ เข้ามาช่วย แต่ระบบประมวลผลบันทึกของระบบงานแบบที่เคยมีมานั้นส่วนใหญ่จะเป็นระบบการประมวลผลแบบ Batch ซึ่งไม่เหมาะที่จะนำมาใช้จัดการกับบันทึกของระบบงาน ในปัจจุบันเพราะบันทึกของระบบงาน ในปัจจุบันนั้นถูกสร้างขึ้นด้วยอัตราเร็วและมีปริมาณของข้อมูลมาก (Fast Data) ดังนั้นทางผู้วิจัยจึงนำระบบการประมวลผลแบบ Real-Time เข้ามาใช้ แต่ก็ไม่สามารถเลิกใช้ระบบการประมวลผลแบบ Batch ได้ เนื่องจากในงานบางงานก็ยังจำเป็นที่จะต้องใช้ระบบการประมวลผลแบบ Batch อยู่ เช่นการทำ checksum ของข้อมูล หรือ การทำสรุปยอดการเงินของธนาคารเป็นต้น ดังนั้นการที่เราจะประมวลผลบันทึกของระบบงาน ได้อย่างมีประสิทธิภาพนั้นเราจำเป็นอย่างยิ่งที่จะต้องมีระบบประมวลผลที่สามารถประมวลผลได้ทั้งแบบ Batch และแบบ Real-Time

2.13.1.1 โครงสร้างการประมวลผลบันทึกของระบบงาน แบบ Batch

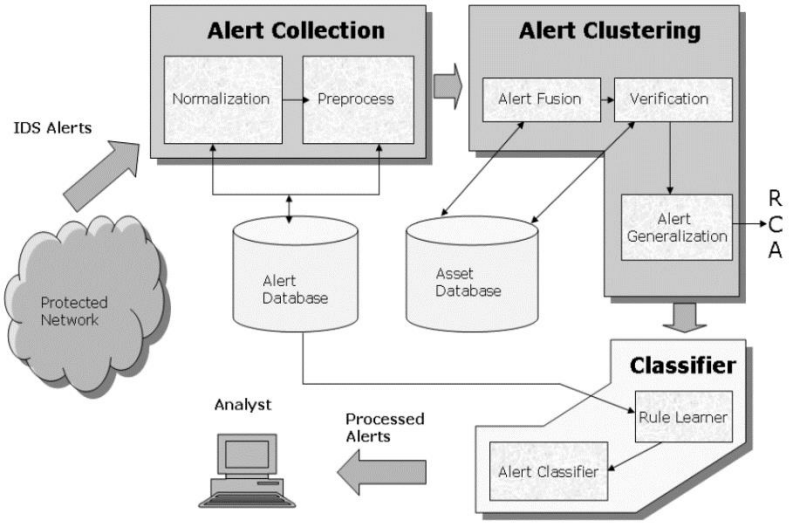
การประมวลผลแบบ Batch เป็นวิธีที่ใช้ในการประมวลผลข้อมูลที่มีปริมาณมากและสะสมมาเป็นระยะเวลาหนึ่ง โดย Yu, Hongyong และ Deshuai Wang [8] ได้นำเสนอโครงสร้างการประมวลผลบันทึกของระบบงาน โดยใช้เทคโนโลยี Hadoop เข้ามาประมวลผลเพื่อจัดการกับบันทึกของระบบงานขนาดใหญ่โดยมีสถาปัตยกรรมดังรูปที่ 8.1 เนื่องจากบันทึกของระบบงานจะถูกเก็บไว้ใน HDFS ทำให้ระบบสามารถที่จะขยายขนาดเพื่อที่จะรองรับข้อมูลที่อาจจะขยายขึ้นได้ในอนาคต Lee, Yeonhee et al [7] และ Therdphapiyanak, Jakrarin et al [9] ได้นำเทคโนโลยี Hadoop เพื่อมาประมวลผลบันทึกของระบบงาน โดยนำเทคนิค intrusion detection มาใช้ในการประมวลผลข้อมูลที่มีปริมาณมาก จากงานวิจัยของ Cheon, JeongJin และ Choe, Tae-Young [10] เทคโนโลยี Hadoop สามารถเพิ่มประสิทธิภาพในการประมวลผล message Log ได้ถึง 426% เมื่อเทียบกับ single computer แต่จะเห็นได้ว่าการประมวลผลที่กล่าวมาข้างต้นนั้นใช้เทคโนโลยี Hadoop ซึ่งเป็นการประมวลผลแบบ Batch สามารถรองรับการประมวลผลข้อมูลที่มีปริมาณมาก แต่ก็ต้องใช้เวลาในการประมวลผลพอสมควร



ภาพที่ 9 โครงสร้างการประมวลบันทึกของระบบงาน โดยใช้ Hadoop

2.13.1.2 โครงสร้างการประมวลผลบันทึกของระบบงาน แบบ Real-Time

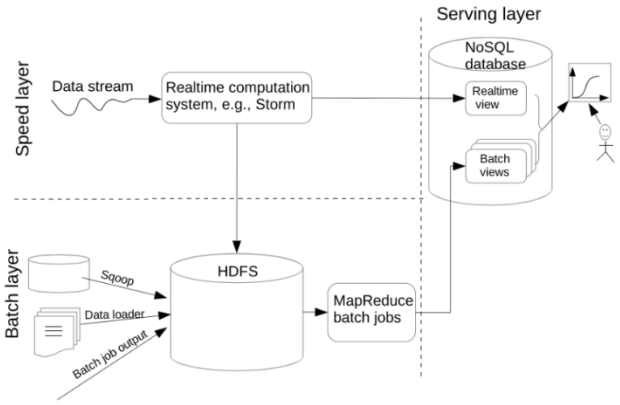
การประมวลผลบันทึกของระบบงาน แบบ Real-Time นั้นเป็นการประมวลผลที่เน้นที่ latency ของระบบ เพื่อที่จะประมวลผลข้อมูลที่มีความเร็วสูง ให้ได้ผลลัพธ์แล้วสามารถนำไปใช้ได้ทันทีเช่น การ Monitoring การใช้งาน network ในระบบเพื่อนำมาปรับ Bandwidth ให้เหมาะสมกับการใช้งาน เป็นต้น การประมวลผลแบบ Real-Time นั้นจะประมวลผลข้อมูลที่เข้ามา ให้ได้ผลลัพธ์ออกมาด้วยอัตราเร็วที่เท่ากับอัตราเร็วที่ข้อมูลไหลเข้าสู้ระบบ Subbulakshmi, T, et al. [12] ได้นำเสนอสถาปัตยกรรมระบบการแจ้งเตือนแบบ Real-Time ดังรูปที่ 8.2 โดยแบ่งการแจ้งเตือนออกเป็น 2 ระยะ แต่ใช้ได้กับบันทึกของระบบงาน ของ Intrusion Detection Systems (IDS) เท่านั้น Mohari, Bhupendra, et al. [13] ได้นำเสนอระบบการจัดการและวิเคราะห์บันทึกของระบบงาน แบบ real-time โดยการนำเอา platform ต่างๆ เข้ามารวมกัน โดยเน้นการประมวลผลให้ได้ผลลัพธ์ออกมาแล้วนำไปเก็บลงบนฐานข้อมูล (process-store) ซึ่งโดยปรกติแล้วการประมวลผลบันทึกของระบบงาน นั้นจะมีขั้นตอนอยู่ด้วยกันสามขั้นตอนคือ เก็บข้อมูลลงฐานข้อมูลก่อนแล้วนำไปประมวลผล เมื่อได้ผลลัพธ์แล้วก็จะนำมาเก็บไว้ในฐานข้อมูลอีก (store-process-store) ทำให้การประมวลผลในรูปแบบนี้สามารถช่วยลดเวลาในการประมวลผลลงได้ แต่ว่าเนื่องจากงานวิจัยนี้จำเป็นที่จะต้องเก็บบันทึกบันทึกของระบบงาน ย้อนหลังไว้เพื่อนำมาประมวลในภายภาคหน้าจึงไม่สามารถตัดขั้นตอนในการจัดเก็บบันทึกของระบบงาน ในขึ้นแรกได้



ภาพที่ 10 โครงสร้างการประมวลผลบันทึกของระบบงาน แบบ Real-Time

2.13.2 สถาปัตยกรรมแบบ Three-layered lambda

จากงานที่ผ่านๆ ระบบการประมวลผลแบบ Batch ใช้ในการจัดการกับข้อมูลที่มีขนาดใหญ่ แต่ในปัจจุบันนั้นข้อมูลถูกสร้างขึ้นด้วยอัตราที่เร็วมาก (Fast Data) แลทางผู้วิจัยมีความต้องการที่จะประมวลผลข้อมูลเหล่านั้น ทำให้การประมวลผลแบบ Batch นั้นไม่เพียงพอที่จะใช้ในการประมวลผล จึงต้องนำการประมวลผลแบบ Real-Time เข้ามาช่วยในการประมวลผล จากงานวิจัย Survey of Real-time Processing System for Big Data [15] ได้เสนอวิธีในการแก้ปัญหาโดยได้เสนอสถาปัตยกรรมแบบ Three-layered lambda ขึ้น จากรูปรูปที่ 9 สถาปัตยกรรมแบบ Three-layered lambda จะประกอบไปด้วย Batch layer เป็นส่วนที่ใช้ในการประมวลผลข้อมูลที่ถูกจัดเก็บมา โดยที่ผลลัพธ์ของ Batch layer นั้นจะเป็นผลลัพธ์ที่ล้าช้า แต่จะมี Speed layer มาจัดการกับปัญหาโดย Speed layer จะประมวลผลข้อมูลที่วิ่งเข้ามาในระบบ Real-Time สุดท้ายจะมี Serving layer ซึ่งเป็นส่วนที่ใช้เก็บผลลัพธ์จากทั้ง Speed layer และ Batch layer เพื่อให้ผู้ใช้นำข้อมูลไปใช้ได้



ภาพที่ 11 Three-layered lambda architecture

2.13.3 Anomaly detection base on SARIMA

จากงานวิจัยที่ผ่านมานั้น ได้มีการนำเอาแบบจำลองอนุกรมเวลาซารีมา (SARIMA) มาพยากรณ์ และใช้ผลจากการพยากรณ์มาตรวจสอบและจำแนกข้อมูลที่มีลักษณะเบี่ยงเบน หรือแปลกแยกไปจากข้อมูลปรกติ เพื่อตรวจสอบหาความผิดปรกติต่างๆ เช่นใน [23] ได้นำ SARIMA มาพยากรณ์การจราจรใน WLAN นอกจากนั้นใน [24] ยังมีการนำ SARIMA มาใช้พยากรณ์ปริมาณงานที่ใช้ในการทำงาน

ในงานวิจัยของ Aphichit และ Krerk [25] ได้นำเสนอการตรวจจับความผิดปรกติของการใช้งานเครือข่ายโดยใช้แบบจะลองอนุกรมเวลาซารีมา (SARIMA) โดยการพยากรณ์ปริมาณการใช้งานเครือข่ายที่จะเกิดขึ้นในอนาคต จากปริมาณการใช้งานเครือข่ายในอดีต แล้วเปรียบเทียบปริมาณการใช้งานเครือข่ายที่พยากรณ์ไว้กับปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริงว่ามีความแตกต่างมากกว่าค่า Threshold หรือไม่ ถ้าเกินจะถือว่ามีความผิดปรกติเกิดขึ้น ซึ่งงานวิจัยชิ้นนี้สามารถตรวจจับความผิดปรกติที่เกิดขึ้นได้จริง แต่ว่าก็ยังมีข้อจำกัดอยู่สองด้านด้วยกันคือ ไม่สามารถใช้ตรวจจับปริมาณการใช่งานที่น้อยผิดปรกติ และความผิดปรกติที่เกิดขึ้นต่อเนื่องเป็นเวลานานได้

บทที่ 3
แนวคิดและวิธีดำเนินงานวิจัย

3.1 ปัญหาการประมวลผลบันทึกของระบบงาน

งานวิจัยนี้จะทำการศึกษาองค์ประกอบที่ส่งผลต่อประสิทธิภาพการทำงานของระบบงานแบบ Real-Time เพื่อมาตอบสนองความต้องการของสำนักเทคโนโลยีสารสนเทศ โดยจะออกแบบสถาปัตยกรรมที่รองรับการทำงาน ทั้งในด้านการจัดเก็บบันทึกของระบบงาน และมีความสามารถในการประมวลผลข้อมูลทั้งในรูปแบบ Real-Time และแบบ Batch โดยข้อมูลที่จัดเก็บนั้นต้องสามารถเรียกออกมาดูหรือตรวจสอบได้ย้อนหลัง 90 วัน

เนื่องจากในงานบางประเภทของสำนักเทคโนโลยีสารสนเทศจำเป็นที่จะต้องประมวลผลข้อมูลภายในระยะเวลาที่จำกัด เช่น การตรวจจับผู้ที่อาจจะใช้ระบบหรือเครือข่ายของสำนักเทคโนโลยีสารสนเทศไปเข้าเว็บไซต์ที่ไม่พึงประสงค์ โดยการนำบันทึกของระบบงาน ที่บันทึกเวลาการใช้งานของผู้ใช้ที่อยู่ในระบบ และบันทึกของระบบงาน ที่บันทึกการเข้าออกขอเว็บไซต์ มาใช้เพื่อระบุ user ของผู้ที่กระทำผิด นอกจากนั้นยังมีงานอีกประเภทหนึ่งที่จะต้องนำข้อมูลซึ้งเก็บไว้มาประมวลผลในภายหลังเช่น การทำรายงานเพื่อวิเคราะห์พฤติกรรมการใช้งานอินเทอร์เนตของแต่ละหน่วยงานเพื่อที่จะนำไปควบคุมปริมาณการจ่าย bandwidth ของแต่ละหน่วยงานให้เหมาะสม เป็นต้น แต่ข้อมูลบันทึกของระบบงาน ของสำนักเทคโนโลยีสารสนเทศนั้นเป็นข้อมูลที่มีปริมาณมาก (Volume), ถูกสร้างขึ้นด้วยความเร็วสูง (Velocity), และมีความรูปแบบที่แตกต่างกัน (Varity) ซึ่งเป็นข้อมูลเชิง Big Data ทำให้ทางผู้จัดทำจะต้องออกแบบระบบการจัดการบันทึกของระบบงานที่มีความคงทน (Reliability) และยังมีความสามารถในการเพิ่มขนาดของฐานข้อมูล (Scalability) รวมถึง สามารถที่จะประมวลผลได้อย่างรวดเร็วตามความต้องการของผู้ใช้ ดังนั้นทางผู้วิจัยจึงนำเทคโนโลยี HDFS ใน Hadoop มาปรับใช้กับงานวิจัย แต่ทางผู้วิจัยพบว่า HDFS นั้นถูกออกแบบมาเพื่อจัดเก็บไฟล์ข้อมูลที่มีขนาดใหญ่ เช่น 1GB, 1TB, หรือว่า 1PB โดยที่ HDFS นั้นจะนำข้อมูลขนาดใหญ่มาแบ่งเป็น block ขนาดเล็ก ซึ่งโดยปรกติขนาดของ block จะอยู่ที่ 128MB (สามารถปรับเปลี่ยนได้) แต่ระบบที่ทางผู้วิจัยได้พัฒนาขึ้นนั้นจำเป็นที่จะต้องรับมือกับข้อมูลที่มีขนาดต่ำกว่า 128MB ซึ่งการที่จะใช้ HDFS ให้เกิดประสิทธิภาพนั้นเราจะต้องรอให้ข้อมูลถูกเขียนเต็มขนาดของ block เสียก่อนจึงจะสามารถประมวลผลได้ ซึ่งการที่เราจะนำข้อมูลเหล่านี่ออกมาใช้ จะต้องเสียเวลาในการรอค่อนข้างนาน

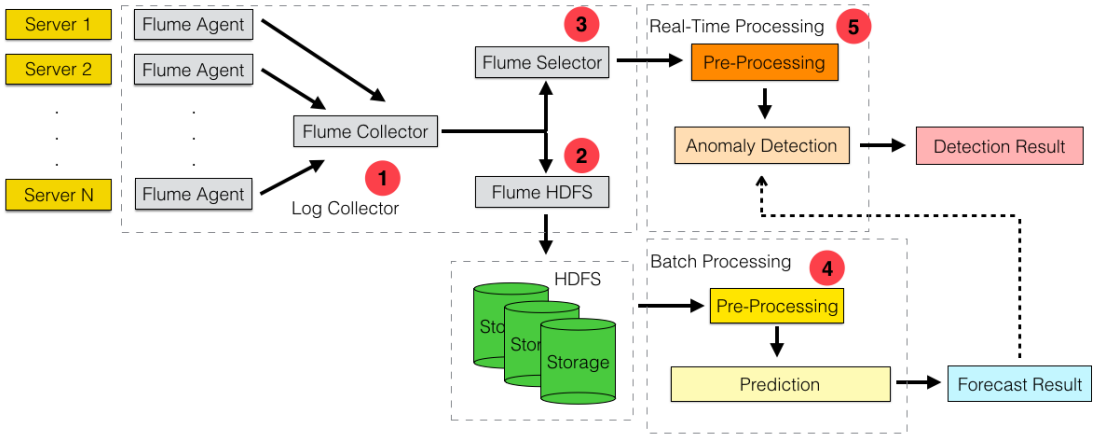
3.2 หลักการทำงานของสถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่

จากปัญหาที่กล่าวไปแล้วในขั้นต้นนั้น ทางผู้วิจัยจึงได้นำสถาปัตยกรรมแบบ Three-layered lambda มาเป็นพื้นฐานในการแก้ปัญหา โดยแนวคิดในการแก้ปัญหานี้จะแบ่งออกเป็น 2 ส่วนด้วยกันคือ

1. สร้างสถาปัตยกรรมที่สามารถประมวลผลบันทึกของระบบงานทั้งในรูปแบบ Batch และ Real-Time รวมถึงรองรับการจัดเก็บข้อมูลที่มีปริมาณมาก และมีแนวโน้มที่จะเพิ่มสูงขึ้น โดยใช้เทคโนโลยี Spark และ Spark Streaming ที่มีใน Spark มาใช้ ในการประมวลผลร่วมกับ HDFS ใน Hadoop
2. ทดลองใช้สถาปัตยกรรมที่ออกแบบ โดยใช้เทคนิคการวิเคราอนุกรมเชิงเวลา (Time series analysis) แบบ SARIMA ในการพยากรณ์ปริมาณการใช้งาน Network และตรวจจับความผิดปรกติของปริมาณบันทึกของระบบงานแบบ Real-Time

โดยสถาปัตยกรรมที่เราออกแบบนั้น สามารถแบ่งออกเป็น 4 ส่วนหลักๆ ดังภาพที่ 12 คือ

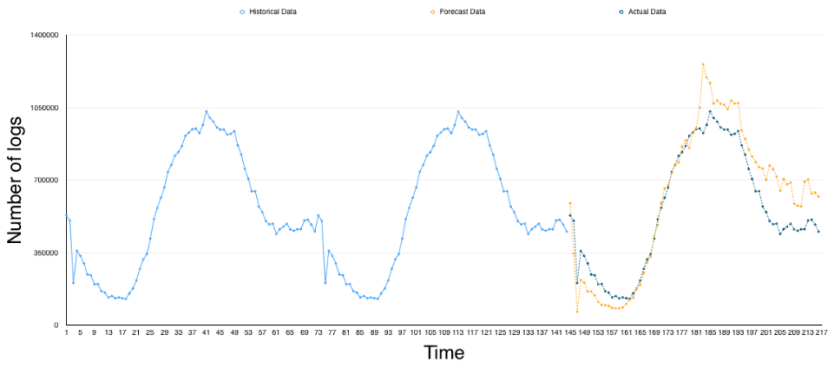
1. ส่วนที่ใช้ในการส่งและรวบรวมบันทึกของระบบงาน
2. ส่วนที่ใช้ในการเก็บข้อมูล HDFS ใช้ในการเก็บบันทึกของระบบงานที่มาจากที่ต่างๆ เพื่อใช้เป็น historical data
3. ส่วนที่ใช้ในการเลือกบันทึกของระบบงานที่เราสนใจ หรือบันทึกของระบบงานที่เป็นประเภทเดียวกัน เนื่องจากระบบที่เราออกแบบจะต้องรองรับบันทึกของระบบงานที่มาจากหลายๆ แหล่ง
4. ส่วนที่ใช่ในการประมวลผลแบบ Batch ใช้ในการประมวลผลเพื่อหาแบบจำลองที่เหมาะสมที่จะนำมาใช้ในการพยากรณ์ โดยใช้ SARIMA Time Series Analysis แล้วนำแบบจำลองที่ได้มาใช้เป็นข้อมูลในการพยากรณ์
5. ส่วนที่ใช้ในการประมวลผลแบบ Real-Time ใช้ในการประมวลผล log ที่เข้ามาใหม่จากระบบอย่างต่อเนื่อง โดยมีการทำ pre-processing แล้วนำข้อมูลที่ได้มาเปรียบเทียบกับแบบจำลองที่เราทำขึ้นจากการประมวลผลแบบ Batch เพื่อตรวจจับและแจ้งเตือนเมื่อมีการใช้งานระบบเครือข่ายมากกว่าหรือ น้อยกว่าปรกติ



ภาพที่ 12 สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่

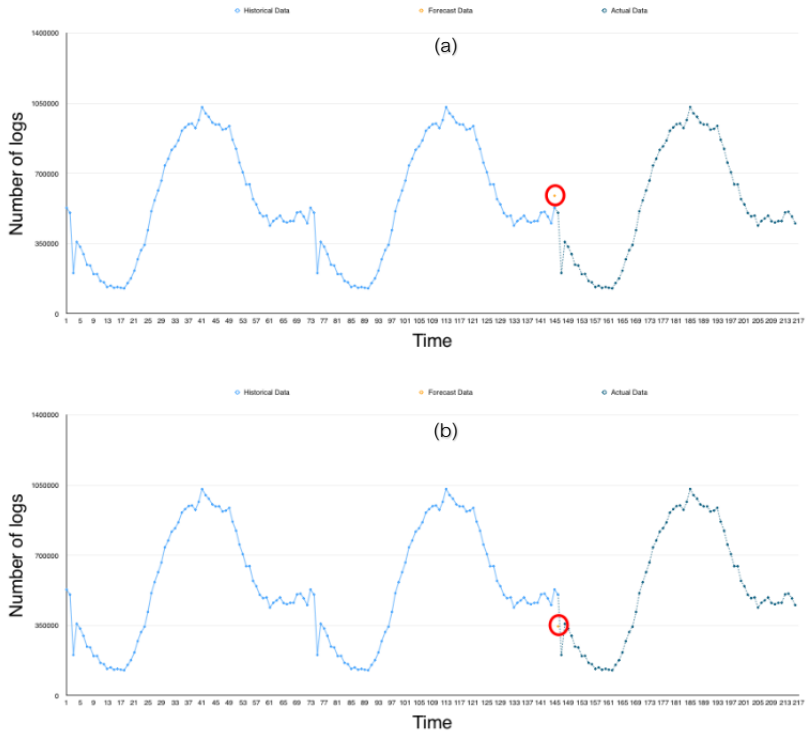
จากภาพที่ 12 ข้อมูลบันทึกของระบบงานจะถูกส่งเข้ามาในระบบโดยใช้ Apache Flume แล้วแยกออกเป็นสองทางโดยบันทึกของระบบงานทุกๆ ข้อความที่เข้ามาจะถูกนำไปเก็บไว้ใน HDFS เป็น historical data และจะถูกนำไปประมวลผลแบบ Batch โดยการนำบันทึกของระบบงานมาหาปริมาณการใช้ระบบอินเทอร์เน็ตที่ผ่านมาและนำมาสร้าง SARIMA Time Series Model ที่เหมาะสม เพื่อนำมาพยากรณ์ถึงพฤติกรรมการใช้งานอินเตอร์เน็ตโดยปรกติที่ควรจะเกิดขึ้น ส่วนอีกทางหนึ่งบันทึกของระบบงานจะถูกส่งเข้า Selector เพื่อเลือกเฉพาะบันทึกของระบบงานที่เราสนใจแล้วนำมาผ่านการ pre-processing เพื่อหาปริมาณการใช้งานระบบอินเทอร์เน็ตแบบ Real-Time และนำผลลัพธ์ที่ได้มาเปรียบเทียบกับผลการพยากรณ์จาก Time Series Model ที่เราสร้างขึ้นในการประมวลผลแบบ Batch เพื่อที่จะตรวจจับและแจ้งเตือนเมื่อตรวจพบความผิดปรกติแบบ Real-Time เช่นมีปริมาณการใช้งานอินเทอร์เน็ตมากกว่าปรกติมากเป็นต้น

3.3 หลักการทำงานของการพยากรณ์ปริมาณการใช้งานเครือข่ายโดยใช้บันทึกของระบบงาน

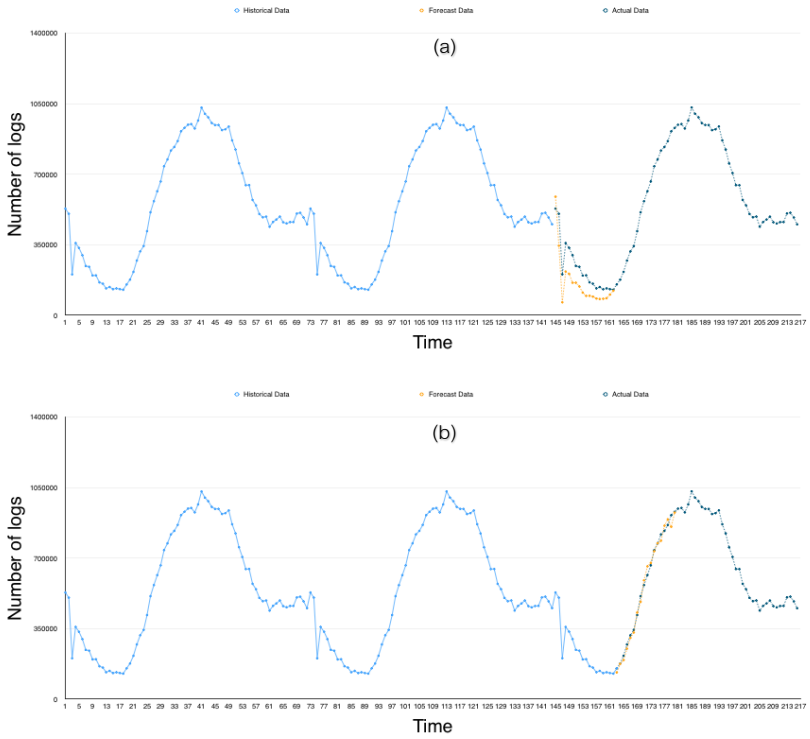


ภาพที่ 13 แสดงการพยากรณ์ปริมาณการใช้งานเครือข่าย

การพยากรณ์ปริมาณการใช้งานเครือข่ายที่จะเกิดขึ้นนั้น เราจะใช้ข้อมูลบันทึกของระบบงานในอดีต (historical log data) มาทำการประมวลผลเพื่อเปลี่ยนจากข้อมูลที่ไม่สามารถใช้การได้หรือก็คือ unstructured data ให้เป็น structure ซึ่งก็คือ ปริมาณการใช้งานเครือข่าย แล้วนำเอาปริมาณการใช้งานเครือข่ายมาสร้างแบบจำลองอนุกรมเวลาซาริมา เพื่อพยากรณ์ปริมาณการใช้งานเครือข่ายที่จะเกิดขึ้นในอนาคต ภาพที่ 13 แสดงตัวอย่างการพยากรณ์การใช้งานเครือข่ายโดยที่กราฟเส้นสีฟ้าจะแสดงถึงปริมาณการใช้งานเครือข่ายในอดีต, สีเหลืองแสดงถึงปริมาณการใช้งานเครือข่ายที่เราพยากรณ์เอาไว้, และสีน้ำเงินแสดงถึงปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง ซึ่งรูปแบบการประมวลผลที่เคยมีมานั้นอาจจะประมวลผลและพยากรณ์ได้ครั้งละ step ข้างหน้า ดังในภาพที่ 14 เนื่องจากต้องประมวลผลข้อมูลที่มีปริมาณมากให้ได้ภายในระยะเวลาที่จำกัด ซึ่งการใช้สถาปัตยกรรมแบบผสมนั้นจะสามารถทำให้เราประมวลผลและพยากรณ์ได้ครั้งละหลาย step เช่น 18 step ดังแสดงในภาพที่ 15 หรืออาจจะพยากรณ์ไว้ 1 วันล่วงหน้าดังแสดงในภาพที่ 13



ภาพที่ 14 แสดงตัวอย่างการพยากรณ์ที่ละ step (a) step ที่ 146 (b) step ที่ 147

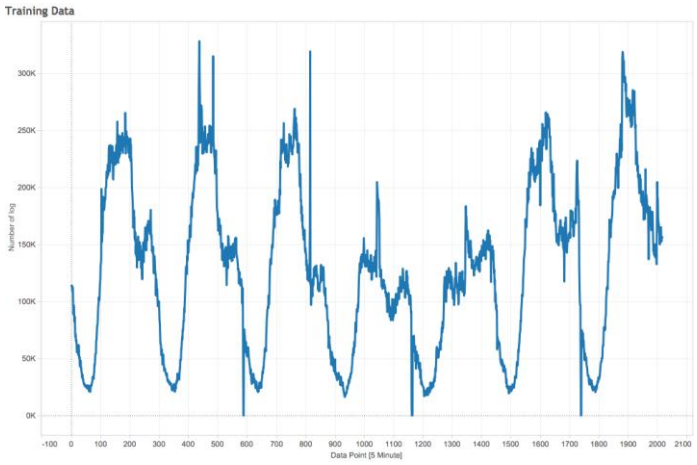


ภาพที่ 15 แสดงตัวอย่างการพยากรณ์ที่ละ 18 steps (a) step ที่ 145 ถึง step ที่ 162 (b) step ที่ 163 ถึง step ที่ 180

3.4 การสร้างแบบจำลองอนุกรมเวลาซาริมา

งานวิจัยนี้จะนำข้อมูลบันทึกของระบบงานจากสำนักเทคโนโลยีของจุฬาลงกรณ์มหาวิทยาลัย ที่จำนวนวันต่างๆ มาใช้ในการสร้างแบบจำลองอนุกรมเวลาซาริมา โดยใช้ช่วงเวลาต่างๆ เป็นหน่วยเวลาเช่น 5 นาที, 10 นาที, 20 นาที, 30 นาที, และ 60 นาที เป็นต้น นอกจากนั้นยังนำค่า BIG มาใช้ในการเลือกแบบจำลองที่มีความศูนย์เสียของข้อมูลน้อยที่สุด อัลกอริทึมในการสร้างแบบจำลองอนุกรมเวลาซาริมาที่เราใช้นั้น ถูกเสนอไว้ในงานวิจัย [25] ซึ่งงานวิจัยนี้ดำเนินตามขั้นตอนหลักๆ ดังนี้

1. หาคาบของฤดูกาล s จากการวิเคราะห์กราฟ เช่น จากภาพที่ 16 ค่าของคาบจะมีค่าเท่ากับ 288



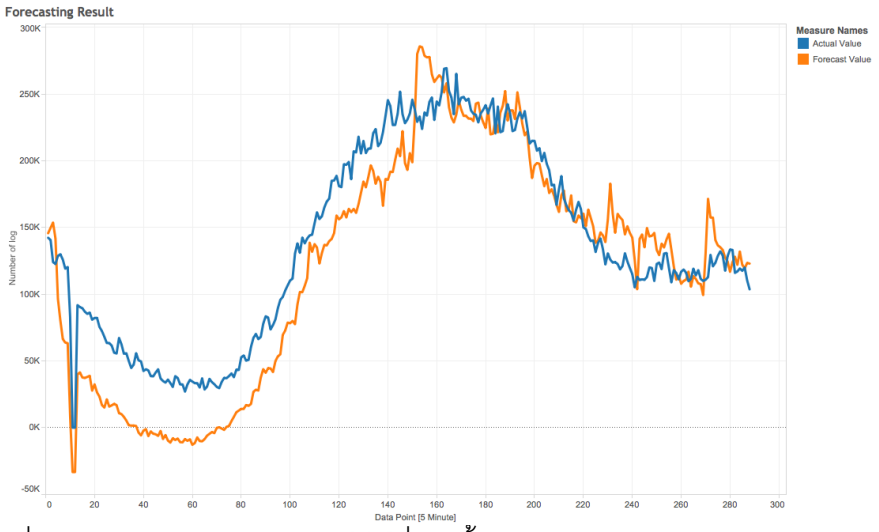
ภาพที่ 16 ปริมาณการใช้งานเครือข่ายจากบันทึกของระบบงาน โดยใช้ 5 นาทีเป็นหน่วยของเวลา

2. หาค่าของ p, d, q, P, D, และ Q โดยการทดสอบแทนค่าโดยที่ค่าของ p และ P ไม่ควรเป็น 0 พร้อมกัน รวมทั้ง q และ Q ไม่ควรเป็น 0 พร้อมกัน จากนั้นเลือกรูปแบบการแทนค่าที่เหมาะสมที่สุดจากค่าของ BIG
3. หาค่าพารามิเตอร์ที่เหลือทั้งหมด เช่น $\phi_1, \Phi_1, \theta_1,$ และ Θ_1 โดยใช้วิธี Maximum likelihood estimation [26]
4. สร้างแบบจำลองอนุกรมเวลา SARIMA ตามสมการที่ 1

3.5 การตรวจหาปริมาณการใช้งานเครือข่ายที่ผิดปรกติโดยใช้แบบจำลองอนุกรมเวลาซาริมา

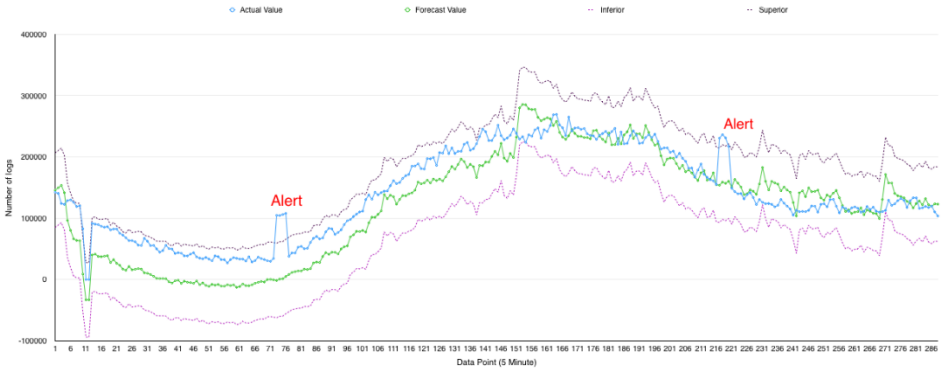
การตรวจหาความผิดปรกติของปริมาณการใช้งานเครือข่าย สามารถทำได้โดยการเปรียบเทียบปริมาณการใช่งานเครือข่ายในปัจจุบัน กับค่าขีดสุด (Threshold) โดยในงานวิจัยชิ้นนี้จะมีค่าขีดสุดซึ่งนำมาจากงานวิจัย [27] สองค่าคือค่า inferior (IT) และค่า superior (ST) ถ้าค่าการใช่งานในปัจจุบันมากว่าค่า superior หรือน้อยกว่าค่า inferior แสดงว่ามีความผิดปรกติเกิดขึ้นกับปริมาณการใช้งานเครือข่าย ซึ่งการตรวจหาปริมาณการใช้งานเครือข่ายที่ผิดปรกติโดยใช้แบบจำลองอนุกรมเวลาซาริมานั้นมีขั้นตอนดังนี้

1. สร้างแบบจำลองอนุกรมเวลาซาริมาโดยใช้วิธีที่อธิบายไปข้างต้น แล้วพยากรณ์ถึงปริมาณการใช้งานเครือข่ายที่ควรจะเกิดขึ้น ภาพที่ 17



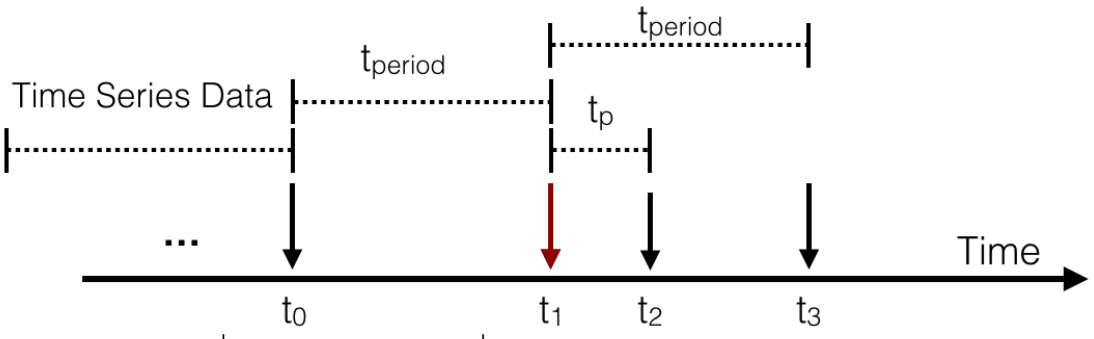
ภาพที่ 17 ปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง เทียบกับผลจากการพยากรณ์

2. นำปริมาณการใช้งานเครือข่ายที่เราพยากรณ์ไว้ มาสร้างค่าขีดสุด
3. นำค่าปริมาณการใช้งานจริงมาเปรียบเทียบกับค่าขีดสุด ถ้าปริมาณการใช้งานเครือข่ายมากกว่า หรือน้อยกว่าค่าขีดสุด แสดงว่ามีความผิดปรกติเกิดขึ้น ดังในภาพที่ 18



ภาพที่ 18 ตัวอย่างการแจ้งเตือนความผิดปรกติของการใช้งานอินเทอร์เน็ต

3.6 นิยามศัพท์ในงานวิจัย

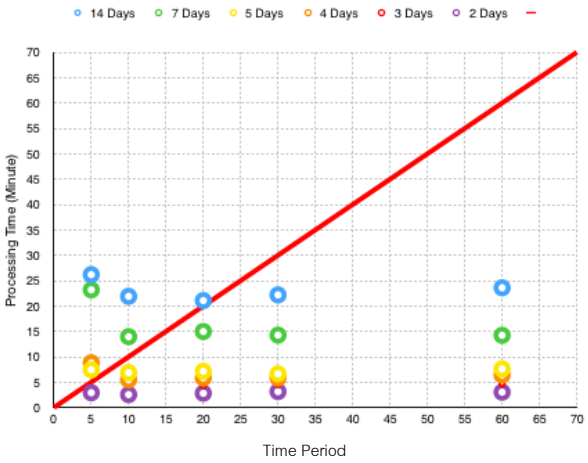


ภาพที่ 19 แสดงถึงช่วงเวลาที่ใช้ในการสร้างแบบจำลองและพยากรณ์

นิยามที่ใช้ในงานวิจัยนี้ มีดังต่อไปนี้

1. Time Series Data คือ เวลาของข้อมูลย้อนหลัง ที่เราจะใช้ในการประมวลผล อาจจะมีค่า 1 วัน, 2 วัน, 3 วัน, หรือ 7 วัน เป็นต้น
2. ช่วงเวลาสะสมข้อมูล (Time Period หรือ t_{period}) เป็นหน่วยของช่วงเวลาที่จะทำการสะสมข้อมูลเพื่อใช้ในการสร้างแบบจำลองและพยากรณ์อาจจะมีค่า 5 นาที, 10 นาที, 20 นาที, หรือ 60 นาที เป็นต้น จากรูปที่ 19 ค่า t_{period} จะเท่ากับ $t_1 - t_0$ และ $t_3 - t_1$
3. เวลาที่ใช้ในการประมวลผล (Processing Time หรือ t_p) เป็นระยะเวลาที่ระบบจะต้องใช้ในการสร้างแบบจำลอง โดยจะใช้ข้อมูล Time Series Data ผนวกกับข้อมูลที่ได้ทำการสะสมในช่วงสะสมข้อมูล (Time Period) เพื่อทำการ พยากรณ์ โดยจากรูปที่ 19 ณ เวลา t_1 ระบบจะทำการนำข้อมูล Time Series Data ที่ได้ทำการสะสมไว้ก่อนช่วง t_0 ผนวกกับข้อมูลในช่วง $t_1 - t_0$ มาทำการสร้างแบบจำลองและพยากรณ์ โดยใช้เวลา $t_2 - t_1$ หรือ t_p ในการดำเนินการ

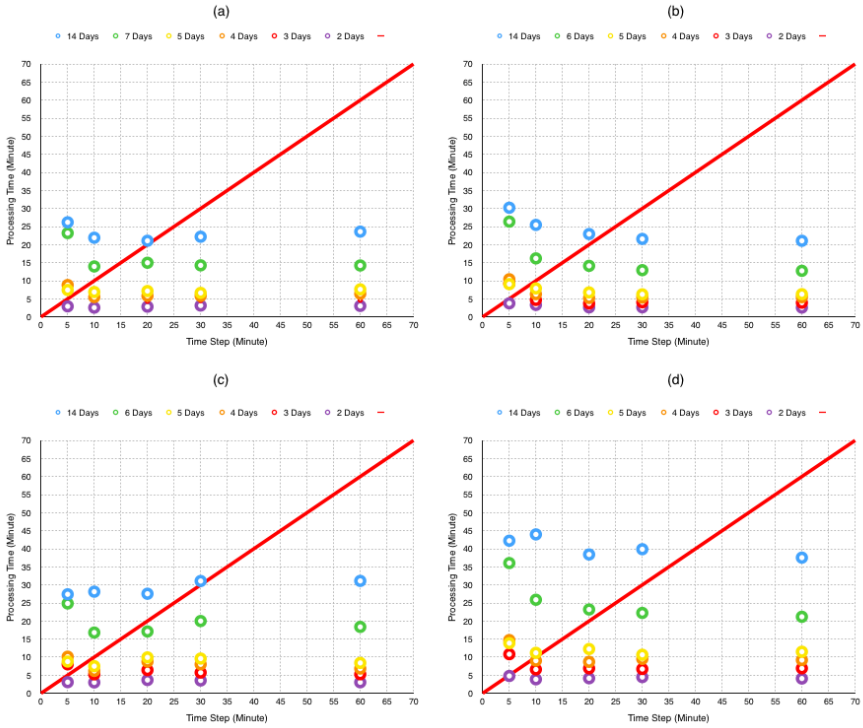
ดังนั้นถ้าหากว่าเราต้องการที่จะประมวลผลข้อมูลและพยากรณ์ปริมาณการใช้งานเครือข่ายให้ทันเวลาเราจะต้องทำให้ $t_p < t_{period}$ แต่ถ้าหากว่า $t_p > t_{period}$ แสดงว่าเราจะไม่สามารถประมวลผลข้อมูลและพยากรณ์ปริมาณการใช้งานเครือข่ายได้ทันเวลา แต่การที่เราจะประมวลผลให้ทันเวลานั้น จะไม่สามารถประมวลผลด้วยข้อมูลย้อนหลังกลับไปได้หลายวัน ซึ่งจะส่งผลให้ค่าปริมาณการใช้งานเครือข่ายที่เราพยากรณ์ออกมานั้นมีค่าผิดไปจากค่าปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง ดังนั้น ถ้าเราต้องการพยากรณ์ปริมาณการใช้งานเครือข่ายให้ได้ค่าใกล้เคียงกับปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง เราจำเป็นต้องใช้ข้อมูลย้อนหลังในปริมาณที่มากทำให้ค่า t_p มีค่ามากทำให้อาจจะทำให้ประมวลผลได้ไม่ทันเวลา



ภาพที่ 20 ผลการประมวลผลโดยใช้ช่วงเวลาที่แตกต่างกันในการประมวลผล

ในภาพที่ 20 แสดงเวลาที่ใช้ในการประมวลผลเทียบกับช่วงเวลาต่างๆ ของข้อมูล และขนาดของข้อมูลย้อนหลัง โดยเส้นสีแดงจะแสดงถึง dead line ของการประมวลผลในช่วงเวลาต่างๆ ถ้าค่าระหว่างเวลาที่ใช้ในการประมวลผลกับช่วงเวลาอยู่เหนือเส้นสีแดง แสดงว่าเราไม่สามารถประมวลผลได้ทันเวลา แต่ถ้าอยู่ใต้เส้นสีแดงแสดงว่าเราสามารถประมวลผลได้ทันเวลา ยกตัวอย่างเช่น ที่ขนาดข้อมูล 2 วัน เวลาที่ใช้ในการประมวลผลในช่วงเวลาต่างๆ ของข้อมูลนั้นไม่เกิน 5 นาทีและอยู่ใต้เส้นสีแดง แสดงว่าเราสามารถประมวลผลโดยใช้ขนาดข้อมูล 2 วันในการประมวลผลแบบ Real-Time ได้ แต่สำหรับข้อมูลที่มีขนาด 14 วัน เราจะสามารถประมวลผลได้ทันเวลา ที่ช่วงเวลา 30 นาที และ 60 นาที แต่ไม่สามารถประมวลผลได้ทันเวลาในช่วงเวลา 5, 10, และ 20 นาที ดังนั้นในงานวิจัยชิ้นนี้ที่ใช้สถาปัตยกรรมแบบผสมนั้น จะประมวลผลข้อมูลหรือพยากรณ์ปริมาณการใช้งานไว้ล่วงหน้าและต้องคอยปรับแบบจำลอง, ผลของการพยากรณ์, รวมทั้งค่าขีดสุดที่ใช้ในการแบ่งความผิดปรกติเป็น Batch แล้วจึงนำผลไปใช้ในการตรวจจับความผิดปรกติของการใช้งานเครือข่ายแบบ Real-Time

นอกจากนั้นยังได้ทำการแสดงกราฟของเวลาที่ใช้ในการประมวลผลเมื่อเทียบกับช่วงเวลาของข้อมูลและปริมาณของข้อมูลที่ใช้ในการประมวลผลโดยทำการเพิ่มจำนวน Node ดังแสดงในรูปที่ 21



ภาพที่ 21 กราฟแสดงเวลาที่ใช้งานการประมวลผลเมื่อเทียบระหว่างช่วงเวลาของข้อมูลและปริมาณของข้อมูลที่ใช้งานการประมวลผล (a) 2 Nodes, (b) 3 Nodes, (c) 4 Nodes, และ 5 Nodes

บทที่ 4

การพัฒนาเครื่องมือและการทดสอบ

ในบทนี้จะเป็นการอธิบายองค์ประกอบของการพัฒนาเครื่องมือที่ใช้เพื่อวิเคราะห์ประสิทธิภาพในการทำงานของระบบ ของสถาปัตยกรรม โดยเริ่มต้นจากสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ จากนั้นอธิบายวิธีการต่างๆ ที่ใช้ในการทดสอบ เพื่อวัดประสิทธิภาพในการทำงานของระบบดังที่นำเสนอไปในบทที่ 3 ซึ่งมีรายละเอียดดังนี้

4.1 สภาพแวดล้อมที่ใช่ในการทดสอบ

ในงานวิจัยชิ้นนี้จะทำการสร้าง cluster ภายในเครื่องเสมือน (Virtual Machine) ขึ้นมาจำนวน 8 node เพื่อใช้เป็น platform สำหรับการทดสอบโดยกำหนดให้มี Master Node 1 ตัว และมี Slave Node 8 ตัว โดยสภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือมีดังต่อไปนี้

1. ฮาร์ดแวร์ (Hardware)
    a. Intel(R) Xeon(R) CPU E5-2670 v2 @2.5GHz
    b. หน่วยความจำสำรอง (RAM) 4 GB
2. ซอร์ฟแวร์ (Software)
    a. ระบบปฏิบัติการ (Operating System)
    b. Apache Hadoop version 2.7.0
    c. Apache Spark version 1.4.1
    d. Apache Flume version 1.5.2

4.2 ข้อมูลที่ใช้ในการทดสอบ

ข้อมูลบันทึกของระบบงานของจุฬาลงกรณ์มหาวิทยาลัยในช่วงเดือน เมษายนต์ และพฤษภาคม พ.ศ. 2559

4.3การทดสอบประสิทธิภาพ

งานวิจัยชิ้นนี้ทดสอบประสิทธิภาพการประมวลผลบันทึกของระบบงาน และพยากรณ์ถึงปริมาณการใช้งานเครือข่ายที่จะเกิดขึ้น จากแบบจำลองอนุกรมเวลาซาริมา โดยแบ่งปริมาณบันทึกของระบบงานที่ใช้ในการสร้างและพยากรณ์ปริมาณการใช้งานเครือข่ายออกเป็น 2 วัน, 3

วัน, 4 วัน, 5 วัน, 10 วัน, และ 15 วัน รวมถึงใช้ช่วงเวลาต่างๆ ในการสร้างแบบจำลองและพยากรณ์ เป็นหน่วยเวลาเช่น 5 นาที, 10 นาที, 20 นาที, 30 นาที, และ 60 นาที

4.3.1 การทดสอบประสิทธิภาพในการพยากรณ์ปริมาณการใช้งานเครือข่าย

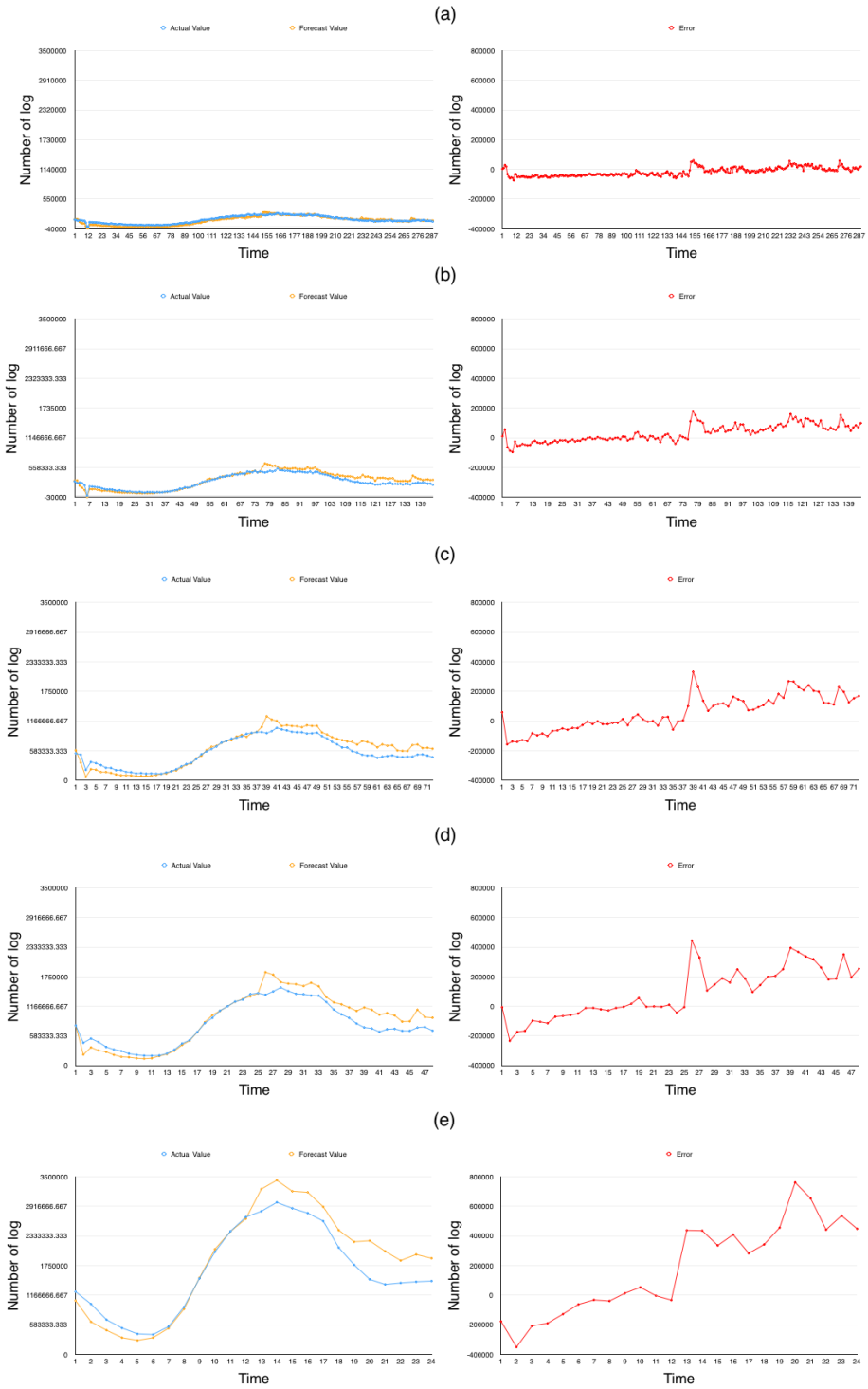
ภาพที่ 22 แสดงถึงปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์เอาไว้ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ ในภาพที่ 22(a) แสดงกราฟเปรียบเทียบระหว่างผลการพยากรณ์ของปริมาณการใช้งานเครือข่ายเทียบกับปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริงในระดับ 5 นาที พร้อมกับค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ ภาพที่ 22(b), 22(c), 22(d), และ 22(e) ในระดับ 10, 20, 30, และ 60 นาที ตามลำดับ

จากการทดสอบจะเห็นได้ว่า ถ้าหากใช้ช่วงเวลาสั้นๆ เช่น 5 นาที หรือ 10 นาที ในการสร้างแบบจำลองและพยากรณ์ ค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์จะมีค่าน้อย แต่ถ้าหากว่าเราใช้ช่วงเวลายาวๆ เช่น 30 นาที หรือ 60 นาทีในการสร้างแบบจำลองและพยากรณ์นั้น ค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ที่เกิดขึ้นก็จะสูงขึ้นตามไปด้วย ซึ่งค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ (Forecasting Error) ที่เกิดขึ้นนั้น เกิดมาจากค่าความต่างของปริมาณการใช้งานเครือข่ายที่เราพยากรณ์เอาไว้ (Forecast value) กับค่าปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง (Actual value) ดังที่แสดงในสมการที่ 2

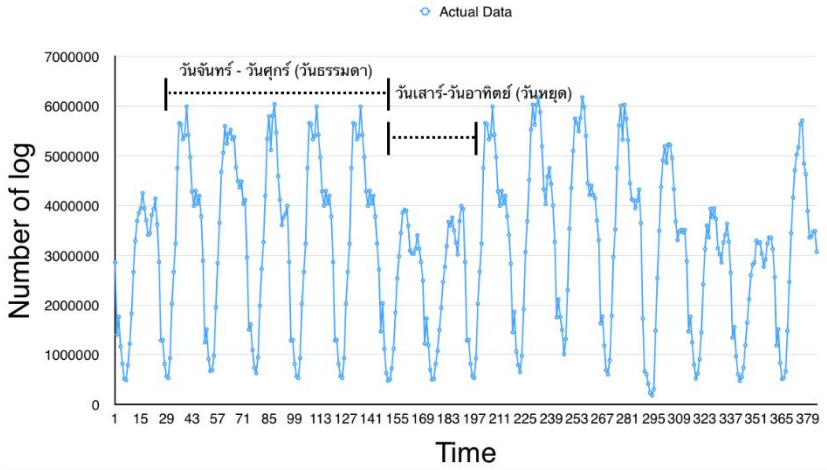
$$e(t) = \hat{y}(t) - y(t) \qquad (2)$$

จากสมการที่ 2 ตัวแปรต่างๆ ของสมการมีดังนี้

- $e(t)$ คือค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์
- $\hat{y}(t)$ คือค่าปริมาณการใช้งานเครือข่ายที่เราพยากรณ์
- $y(t)$ คือค่าปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจริง



ภาพที่ 22 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ (a) ระดับ 5 นาที, (b) ระดับ 10 นาที, (c) ระดับ 20 นาที, (d) ระดับ 30 นาที, และ (e) ระดับ 60 นาที



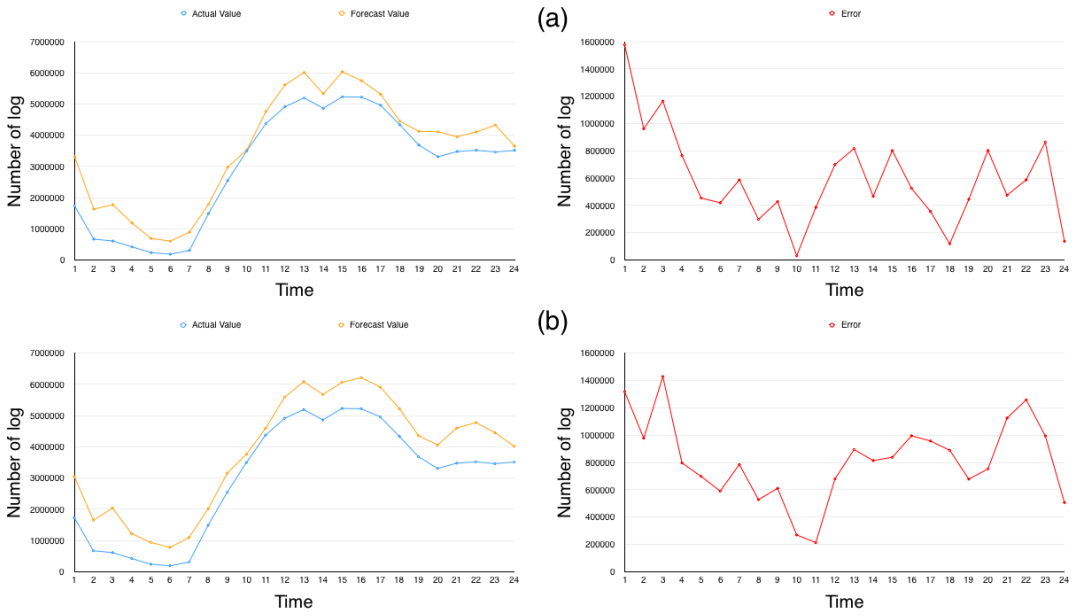
ภาพที่ 23 ตัวอย่างข้อมูลของปริมาณการใช้งานเครือข่าย

นอกจากนั้นทางผู้วิจัยยังได้ทอสอบประสิทธิภาพการพยากรณ์ปริมาณการใช้งานเครือข่าย โดนใช้ปริมาณข้อมูลที่แตกต่างกัน โดยจะแบ่งออกเป็น 2 กรณีด้วยกันคือ

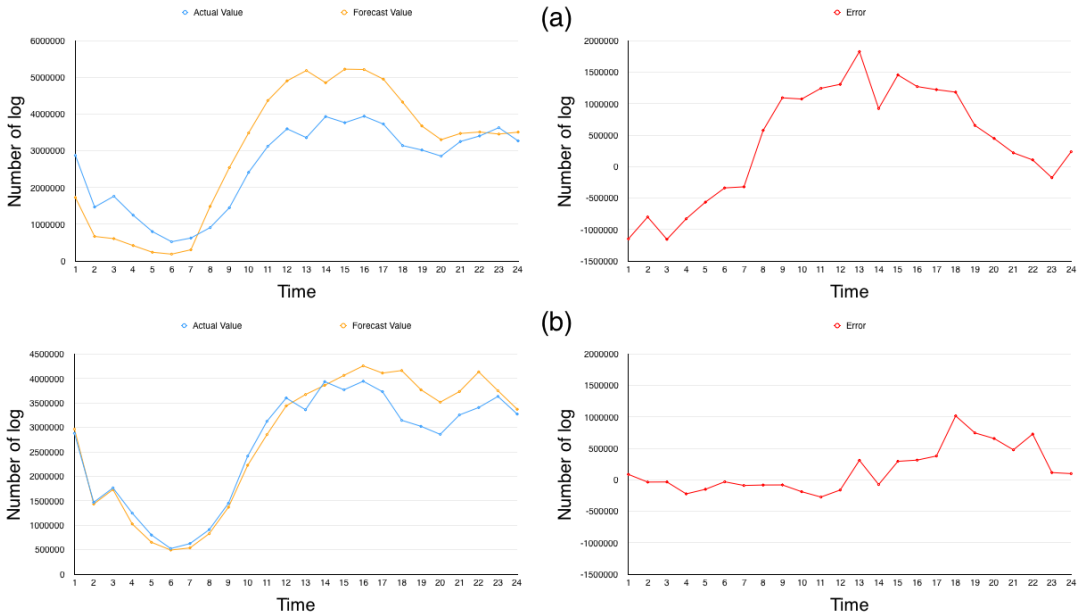
1. นำเอาข้อมูลที่มีปริมาณเท่าๆ กันมาสร้างแบบจะลองเช่น การนำเอาวันธรรมดา คือ จันทร์, อังคาร, พุธ, พฤหัสบดี, ศุกร์ มาเพื่อพยากรณ์ปริมาณการใช้งานเครือข่ายของวันธรรมดา หรือว่านำเอาวันหยุด คือ วันเสาร์และวันอาทิตย์ มาเพื่อพยากรณ์ปริมาณการใช้งานเครือข่ายของวันหยุด
2. นำเอาข้อมูลของวันธรรมดามาประมวลผลเพื่อพยากรณ์ปริมาณการใช้งานเครือข่ายในวันหยุด หรือ นำเอาข้อมูลของวันหยุดมาประมวลผลเพื่อพยากรณ์ปริมาณการใช้งานเครือข่ายในวันธรรมดา

จะเห็นได้ว่าการที่เราใช้วันที่มีปริมาณการใช้งานเครือข่ายเท่าๆ กันหรือพอๆ กัน (กรณีที่ 1) มาสร้างแบบจำลองและพยากรณ์นั้น ถึงแม้ว่าเราจะใช้ปริมาณข้อมูลย้อนหลังไปไกลค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ก็ไม่ได้แตกตากไปจากเดิมมาก ดังแสดงในภาพที่ 24

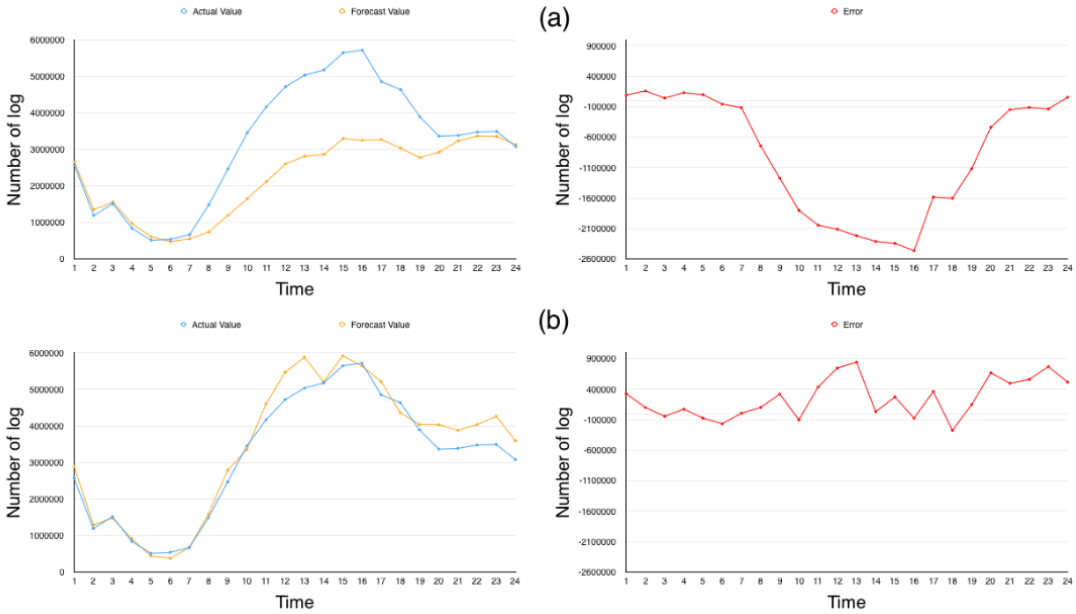
ส่วนในกรณีที่ 2 จากภาพที่ 25 การที่เราใช้ปริมาณการใช้งานเครือข่ายจากวันธรรมดามาสร้างแบบจำลองและพยากรณ์ปริมาณการใช้งานเครือข่ายของวันหยุด เราพบว่าการที่เราเพิ่มปริมาณข้อมูลย้อนหลังไปไกลขึ้นนั้น ค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ก็จะลดลง เช่นเดียวกันกับภาพที่ 26 การที่เราใช้ปริมาณการใช้งานเครือข่ายจากวันหยุดมาสร้างแบบจำลองและพยากรณ์ปริมาณการใช้งานเครือข่ายของวันธรรมดา เราพบว่าการที่เราเพิ่มปริมาณข้อมูลย้อนหลังไปไกลขึ้นนั้น ค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ก็จะลดลง



ภาพที่ 24 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ โดยการนำวันที่มีปริมาณเท่าๆ กัน โดย (a) ใช้ข้อมูลย้อนหลัง 2 วัน (b) ใช้ข้อมูลย้อนหลัง 7 วัน



ภาพที่ 25 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ โดยการนำวันธรรมดามาทำนายวันหยุด โดย (a) ใช้ข้อมูลย้อนหลัง 2 วัน (b) ใช้ข้อมูลย้อนหลัง 7 วัน



ภาพที่ 26 ปริมาณการใช้งานระบบเครือข่ายที่เกิดขึ้นจริงเทียบกับค่าปริมาณการใช้งานระบบเครือข่ายที่พยากรณ์ และค่าความผิดพลาดของปริมาณการใช้งานเครือข่ายที่เกิดขึ้นจากการพยากรณ์ โดยการนำวันหยุดมาทำนายวันธรรมดา โดย (a) ใช้ข้อมูลย้อนหลัง 2 วัน (b) ใช้ข้อมูลย้อนหลัง 7 วัน

4.3.2 การทดสอบประสิทธิภาพในการตรวจหาความผิดปรกติของเครือข่าย

งานวิจัยนี้ทดสอบประสิทธิภาพของการตรวจหาความผิดปรกติของการใช้งานเครือข่ายโดยใช้แบบจำลองอนุกรมเวลาซาริมา ด้วยการจำลองการโจมตีเพื่อให้เกิดการหยุดให้บริการ โดยปัจจัยที่ใช่ในการจำลองการโจมตี มีดังนี้

1. อัตราการใช้งานอินเทอร์เน็ตที่ใช้ในการโจมตีแบ่งออกเป็น 4 กรณีได้แก่
    a. 10 เปอร์เซ็นต์ของอัตราการใช้งานอินเทอร์เน็ตสูงสุด
    b. 25 เปอร์เซ็นต์ของอัตราการใช้งานอินเทอร์เน็ตสูงสุด
    c. 50 เปอร์เซ็นต์ของอัตราการใช้งานอินเทอร์เน็ตสูงสุด
    d. 70 เปอร์เซ็นต์ของอัตราการใช้งานอินเทอร์เน็ตสูงสุด
2. ระยะเวลาที่ใช้ในการโจมตีแบ่งเป็น 2 กรณี
    a. 5 นาที่
    b. 10 นาที
    c. 20 นาที

3. เวลาที่ใช้โจมตี ได้แก่
    a. 6.00 น.
    b. 18.00 น.
4. ช่วงเวลาของข้อมูล
    a. 5 นาที
    b. 10 นาที
    c. 20 นาที
    d. 30 นาที
    e. 60 นาที

การทดสอบจะใช้การเปรียบปริมาณการใช้งานเครือข่าย มาเปรียบเทียบกับค่าขีดสุด ถ้าค่าปริมาณการใช้งานเครือข่ายมีค่ามากกว่า หรือน้อยกว่าค่าขีดสุด แสดงว่ามีการตรวจพบความผิดปรกติของการใช้งาน

ตารางที่ 2 ผลลัพธ์ของการทดสอบประสิทธิภาพในการตรวจหาความผิดปรกติของเครือข่าย

(a)

5 Minute Period

	5 Minute Data Point		10 Minute Data Point		20 Minute Data Point		30 Minute Data Point		60 Minute Data Point	
	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M
DDOS 10%	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO
DDOS 25%	YES	YES	NO	NO	NO	NO	NO	NO	NO	NO
DDOS 50%	YES	YES	YES	NO	NO	NO	NO	NO	NO	NO
DDOS 75%	YES	YES	YES	YES	YES	NO	NO	NO	NO	NO

(b)

10 Minute Period

	5 Minute Data Point		10 Minute Data Point		20 Minute Data Point		30 Minute Data Point		60 Minute Data Point	
	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M
DDOS 10%	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO
DDOS 25%	YES	YES	YES	NO	NO	NO	NO	NO	NO	NO
DDOS 50%	YES	YES	YES	YES	YES	NO	NO	NO	NO	NO
DDOS 75%	YES	YES	YES	YES	YES	YES	YES	NO	NO	NO

(c)

20 Minute Period

	5 Minute Data Point		10 Minute Data Point		20 Minute Data Point		30 Minute Data Point		60 Minute Data Point	
	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M	6 A.M.	6 P.M
DDOS 10%	YES	NO	NO	NO	NO	NO	NO	NO	NO	NO
DDOS 25%	YES	YES	YES	NO	YES	NO	NO	NO	NO	NO
DDOS 50%	YES	YES	YES	YES	YES	YES	YES	YES	NO	NO
DDOS 75%	YES	YES	YES	YES	YES	YES	YES	YES	YES	NO

จากการทดสอบในตารางที่ 2(a) แสดงถึงผลการตรวจสอบการใช้งานเครือข่ายที่ผิดปรกติ ที่เกิดขึ้นเป็นเวลา 5 นาที โดยทำให้เกิดปริมาณการใช้งานเครือข่ายเพิ่มขึ้น 10% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 นาทีเราสามารถตรวจพบความผิดปรกติของการใช้งานเครือขายในช่วงเช้า คือ 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 25% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M., ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 50% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 10 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 75% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 และ 10 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 20 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น

จากตารางที่ 2(b) แสดงถึงผลการตรวจสอบการใช้งานเครือข่ายที่ผิดปรกติ ที่เกิดขึ้นเป็นเวลา 10 นาที โดยทำให้เกิดปริมาณการใช้งานเครือข่ายเพิ่มขึ้น 10% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 นาทีเราสามารถตรวจพบความผิดปรกติของการใช้งานเครือขายในช่วงเช้า คือ 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 25% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 10 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 50% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 และ 10 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 20 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 75% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5, 10, และ 20 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 30 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น

จากตารางที่ 2(c) แสดงถึงผลการตรวจสอบการใช้งานเครือข่ายที่ผิดปรกติ ที่เกิดขึ้นเป็นเวลา 20 นาที โดยทำให้เกิดปริมาณการใช้งานเครือข่ายเพิ่มขึ้น 10% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5 นาทีเราสามารถตรวจพบความผิดปรกติของการใช้งานเครือขายในช่วงเช้า คือ 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 25% ของการใช้งานเครือข่าย

ปรกติ ที่ช่วงเวลา 5 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 10 นาที และ 20 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น, ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 50% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5,10,20, และ 30 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M., ถ้าทำให้ความผิดปรกติของเครือข่ายเพิ่มขึ้นเป็น 75% ของการใช้งานเครือข่ายปรกติ ที่ช่วงเวลา 5,10,20, และ 30 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติของการใช้งานเครือข่ายที่เวลา 6 A.M. และ 6 P.M. ที่ช่วงเวลา 60 นาที เราจะสามารถตรวจสอบการใช้งานที่ผิดปรกติในเวลา 6 A.M. เท่านั้น

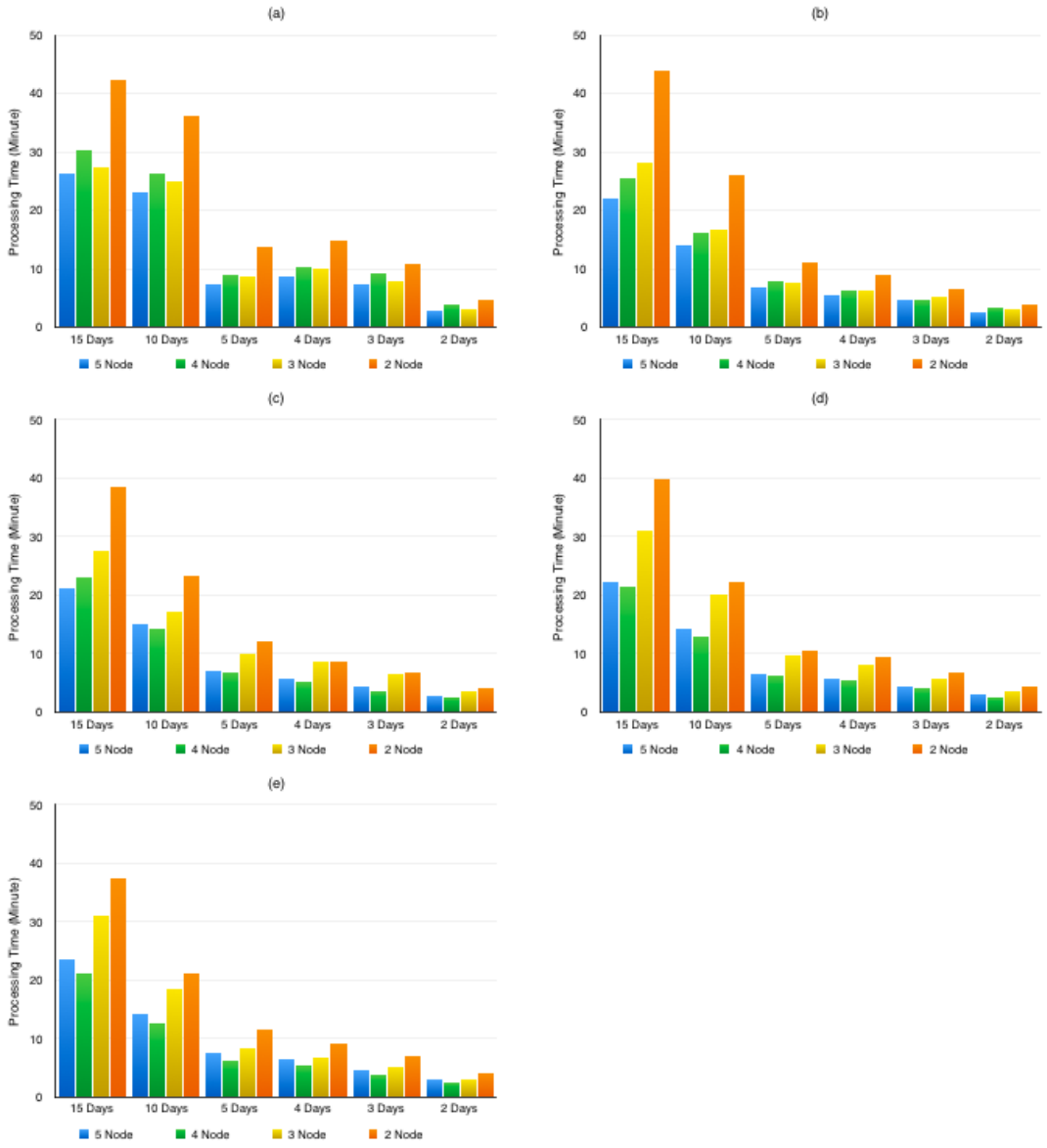
จากการทดสอบประสิทธิภาพในการตรวจจับความผิดปรกติของเครือข่าย จากตารางที่ 3 พบว่าการที่เราใช้ข้อมูลที่มีความถีต่ำๆ มาใช้ในการสร้างแบบจำลองอนุกรมเวลาและตรวจสอบความผิดปรกตินั้น ทำให้เราสามารถตรวจจับความผิดปรกติของเครือข่ายที่เกิดจากการโจมดีที่มีอัตราการใช้งานต่ำได้ดีกว่า การที่เราใช้ช่วงเวลาสูงในการสร้างแบบจำลองอนุกรมเวลาและตรวจสอบความผิดปรกติ

4.3.3 การทดสอบประสิทธิภาพในการประมวลผลและพยากรณ์ปริมาณการใช้งานเครือข่าย

ในภาพที่ 27 แสดงถึงเวลาที่ใช้งานการประมวลผลของบันทึกของระบบงาน โดยจะรวมเวลาที่ใช้งานการหาปริมาณการใช้งานเครือข่าย, สร้างแบบจำลองอนุกรมเวลาซาริมา และการพยากรณ์ปริมาณการใช้งานเครือข่ายภาพที่ 27(a) แสดงถึงเวลาที่ใช้ในการประมวลผลบันทึกของระบบงานโดยใช้ปริมาณของบันทึกของระบบงานย้อนหลัง 2 วัน, 3 วัน, 4 วัน, 5 วัน, 10 วัน, และ 15 วัน โดยใช้ช่วงเวลาของข้อมูลระดับ 5 นาที และแบ่งการประมวลผลออกเป็น 2 node, 3 node, 4 node, และ 5 node นอกจากนั้นในภาพที่ 27(b), 27(c), 27(d), และ 27(e) จะแสดงถึงการใช้ช่วงเวลาข้องข้อมูลที่แตกต่างกัน โดยที่ภาพที่ 27(b) จะใช้ 10 นาทีเป็นช่วงเวลาของข้อมูล, ภาพที่ 27(c) จะใช้ 20 นาทีเป็นช่วงเวลาของข้อมูล, ภาพที่ 27(d) จะใช้ 30 นาทีเป็นช่วงเวลาของข้อมูล, และ ภาพที่ 27(e) จะใช้ 60 นาทีเป็นช่วงเวลาของข้อมูล

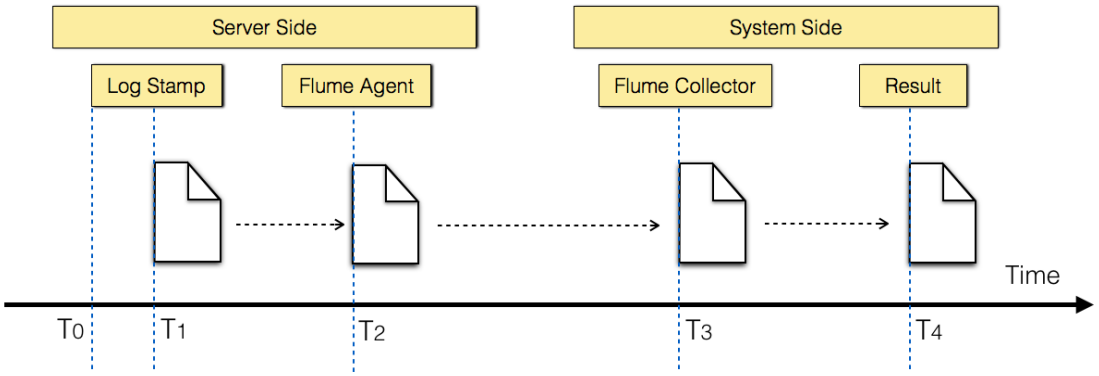
จากการทดสอบจะเห็นได้ว่าการที่เราเพิ่มปริมาณของข้อมูลที่ใช้ในการประมวลผลนั้น เวลาที่ใช้ในการประมวลผลก็จะเพิ่มขึ้นด้วย แต่ถ้าหากเราเพิ่มจำนวน node หรือหน่วยประมวลผลขึ้น จะทำให้เราสามารถลดระยะเวลาที่ใช้ในการประมวลผลให้น้อยลงได้ แต่ว่า

ในบางกรณีเช่นในภาพที่ 27(e) นั้น การเพิ่มจำนวน node ที่ใช้ในการประมวลผลกลับ ไม่ได้ทำให้การประมวลผลนั้นเร็วขึ้นแต่กลับทำให้ช้าลง



ภาพที่ 27 เวลาที่ใช้งานการประมวลผล (a) ระดับ 5 นาที, (b) ระดับ 10 นาที, (c) ระดับ 20 นาที, (d) ระดับ 30 นาที, และ (e) ระดับ 60 นาที

4.3.4 การทดสอบประสิทธิภาพในการทำงานในสภาวะแบบ Real-Time



ภาพที่ 28 แสดงการทำงานประมวลผลของระบบในสภาวะ Real-Time

งานวิจัยนี้ทดสอบประสิทธิภาพของการตรวจหาความผิดปรกติของการใช้งานเครือข่ายแบบ Real-Time โดยใช้แบบจำลองอนุกรมเวลาซาริมา ด้วยการจำลองการโจมตีเพื่อให้เกิดการหยุดให้บริการ โดยจะสร้างบันทึกของระบบงานโดยใช้ความถี่ที่แตกต่างกันคือ 1,000 messages/sec, 2,000 messages/sec, 3,000 messages/sec, 4,000 messages/sec, และ 5,000 messages/sec โดยใช้ระยะเวลา 5 นาที และแบ่งตัวชี้วัดออกเป็นสามอย่างด้วยกันคือ

1. ระยะเวลาที่ข้อมูลจะพร้อมใช้ (Delivery Time) (T_3-T_1)
2. ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติ (Detection Time) (T_4-T_1)

ตารางที่ 3 ผลการทดสอบประสิทธิภาพในการประมวลผลแบบ Real-Time

	1000	2000	3000	4000	5000
ระยะเวลาที่ข้อมูลจะพร้อมใช้	1552.2	1473.6	2808.4	1797.6	1687
ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติ	1560.8	1476.4	2873.6	1852.2	1730

จากการทดสอบพบว่า ถ้าหากใช้ความถี่ในการสร้างบันทึกของระบบงาน 1,000 messages/sec ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติมีค่าเท่ากับ 1560.8 มิลลิวินาที และระยะเวลาที่ข้อมูลจะพร้อมใช้เท่ากับ 1552.2 มิลลิวินาที ถ้าหากใช้ความถี่ในการสร้างบันทึกของระบบงาน 2,000 messages/sec ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติมีค่าเท่ากับ 1476.4 มิลลิวินาที และ ระยะเวลาที่ข้อมูลจะพร้อมใช้เท่ากับ 1473.6 มิลลิวินาที ถ้าหากใช้ความถี่ในการสร้างบันทึกของระบบงาน 3,000 messages/sec ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติมีค่าเท่ากับ 2873.6 มิลลิวินาที และ ระยะเวลาที่ข้อมูลจะพร้อมใช้เท่ากับ 2808.4

มิลลิวินาที ถ้าหากใช้ความถี่ในการสร้างบันทึกของระบบงาน 4,000 messages/sec ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติมีค่าเท่ากับ 1852.2 มิลลิวินาที และ ระยะเวลาที่ข้อมูลจะพร้อมใช้เท่ากับ 1797.6 มิลลิวินาที และถ้าหากใช้ความถี่ในการสร้างบันทึกของระบบงาน 5,000 messages/sec ระยะเวลาที่ใช้ในการตรวจจับความผิดปรกติมีค่าเท่ากับ 1730 มิลลิวินาที และระยะเวลาที่ข้อมูลจะพร้อมใช้เท่ากับ 1687 มิลลิวินาที

4.5 วิเคราะห์ผลการทดลอง

เนื่องจากการพยากรณ์ปริมาณการใช่งานเครือข่ายโดยใช้แบบจำลองเวลาซาริมา จากบันทึกของระบบงานนั้นเป็นการสร้างแบบจำลองที่เกิดขึ้นจากการนำข้อมูลในอดีตมาประมวลผลเพื่อที่จะพยากรณ์ถึงสิ่งที่จะเกิดขึ้น ดังนั้นการที่เรารู้ข้อมูลย้อนหลังกลับไปก็จะส่งผลให้ผลในการพยากรณ์นั้นแม่นยำขึ้น และจากการทดลองพบว่า ความถูกต้องของปริมาณการใช้งานเครือข่ายที่เราพยากรณ์ออกมานั้นจะเปลี่ยนไป ตามช่วงเวลาที่เราใช้ในการสร้างแบบจำลองและพยากรณ์ ถ้าหากช่วงเวลาที่ใช้ในการสร้างแบบจำลองและพยากรณ์มีค่ามาก ค่าความผิดพลาดจากการพยากรณ์ที่เกิดขึ้นก็จะสูงมาก แต่ถ้าหากว่าเราต้องการที่จะพยากรณ์ปริมาณการใช้งานเครือข่ายให้มีค่าความผิดพลาดจากการพยากรณ์ต่ำๆ นั้นเราจะต้องใช้ช่วงเวลาสั้นๆ อย่างเช่น 5 นาที มาใช้ในการประมวลผลดังแสดงในภาพที่ 22 เป็นต้น ดังนั้นถ้าหากว่าเราต้องการประมวลข้อมูลเหล่านี้เพื่อสร้างแบบจำลองและพยากรณ์ให้ได้ภายใน Real-Time หรือทันเวลา เราจะไม่สามารถประมวลผลข้อมูลย้อนหลังได้หลายวันดังแสดงในภาพที่ 20

แต่การใช้สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่นั้นจะประมวลผลข้อมูลหรือพยากรณ์ปริมาณการใช้งานไว้ล่วงหน้าเป็น Batch แล้วนำผลไปใช้ในการตรวจจับความผิดปรกติของการใช้งานเครือข่ายแบบ Real-Time ทำให้เราสามารถพยากรณ์ไปล่วงหน้าได้ ยกตัวอย่างเช่น จากภาพที่ 20 ถ้าหากว่าเราใช้ Time Period เป็น 10 นาที และใช้ข้อมูลย้อนหลัง 7 วันเราจะสามารถพยากรณ์ได้ภายใน 15 นาที โดยที่เราจะพยากรณ์ไว้ล่วงหน้าอย่างน้อย 20 นาทีหรือมากกว่านั้นก็ได้ และถ้าหากว่าเราใช้ข้อมูลย้อนหลังเป็น 14 วันเราจะสามารถพยากรณ์ได้ภายใน 22 นาที โดยที่เราจะพยากรณ์ไว้ล่วงหน้าอย่างน้อย 30 นาทีหรือมากกว่านั้นก็ได้

บทที่ 5
บทสรุปและแนวทางในการพัฒนาต่อ

5.1 สรุปผลการวิจัย

งานวิจัยนี้เสนอสถาปัตยกรรมที่มีความสามารถในการประมวลผลข้อมูลได้ทั้งในรูปแบบ Batch และแบบ Real-Time ทดสอบโดยนำวิธีการตรวจจับความผิดปรกติของเครือข่ายโดยใช้แบบจำลองอนุกรมเวลาซาริมามาประยุกต์ใช้ โดยในการประมวลผลแบบ Batch นั้นจะเป็นการประมวลผลข้อมูลที่เก็บไว้ เพื่อนำมาสร้างแบบจำลองอนุกรมเวลาซาริมาและทำการพยากรณ์ ถึงปริมาณการใช้งานเครือข่ายที่ควรจะเป็นรวมทั้งคำนวณหาค่าขีดสุดเพื่อแบ่งแยกระหว่างเครือข่ายที่ปรกติและเครือข่ายที่ผิดปรกติ ในส่วนของการประมวลผลแบบ Real-Time นั้นจะใช้ในการประมวลผลบันทึกของระบบงาน เพื่อหาปริมาณการใช้งานเครือข่ายแล้วนำมาเปรียบเทียบกับปริมาณการใช้งานเครือข่ายที่พยากรณ์เอาไว้ ถ้าปริมาณการใช้งานเครือข่ายมากกว่าค่าขีดสุด ก็แสดงว่ามีความผิดปรกติของการใช้งานเครือข่ายเกิดขึ้น

แต่การที่เราใช้การตรวจจับความผิดปรกติของเครือข่ายโดยใช้แบบจำลองอนุกรมเวลาซาริมาได้อย่างแม่นยำนั้น เราจะต้องใช้ช่วงเวลาสั้นๆ เช่น 5 นาที ในการประมวลผลและตรวจจับความผิดปรกติ นอกจากนั้นยังต้องใช้ปริมาณข้อมูลย้อนหลังหลายวัน ซึ่งทำให้เวลาที่ใช้ในการประมวลผลก็จะมากขึ้นตามไปด้วย จากการทดสอบพบว่าการใช้สถาปัตยกรรมแบบผสมสำหรับการประมวลผลบันทึกของระบบงานขนาดใหญ่นั้น สามารถช่วยเพิ่มประสิทธิภาพในการทำงานกับบันทึกของระบบงานในระบบงานขนาดใหญ่ให้มากขึ้น โดยสามารถประมวลผลข้อมูลที่มีขนาดใหญ่แล้วนำมาสร้างแบบจำลองที่มีความซับซ้อน และยังสามารถตรวจจับความผิดปรกติได้ภายในระยะเวลาที่จำกัดได้

5.2 แนวทางในการพัฒนาต่อ

งานวิจัยนี้สามารถนำไปปรับปรุงและพัฒนาให้มีประสิทธิภาพดียิ่งขึ้นโดยนำไปทดสอบหาค่าที่เหมาะสมสำหรับการตรวจจับความผิดปรกติ หรือทดสอบใช้แบบจำลองอื่นๆ เพื่อให้ผลลัพธ์ในการพยากรณ์ปริมานการใช้งานเครือข่ายนั้นมีความแม่นยำมากยิ่งขึ้น รวมทั้งประยุกต์ใช้เทคนิคหรือวิธีการอื่นๆ ที่ใช้ในการตรวจจับความผิดปรกติเช่น การทำ Clustering หรือการใช้ neural network เป็นต้น และยังสามารถนำไปทดสอบในสภาพแวดล้อมจริง

รายการอ้างอิง

1. Chen, M., S. Mao, and Y. Liu, *Big data: A survey.* Mobile Networks and Applications, 2014. **19**(2): p. 171-209.
2. Hashem, I.A.T., et al., *The rise of "big data" on cloud computing: Review and open research issues.* Information Systems, 2015. **47**: p. 98-115.
3. Xu, W., et al. *Detecting large-scale system problems by mining console logs*. in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. 2010.
4. Wei, J., et al. *Analysis farm: A cloud-based scalable aggregation and query platform for network log analysis*. in *Cloud and Service Computing (CSC), 2011 International Conference on*. 2011. IEEE.
5. Cinque, M., D. Cotroneo, and A. Pecchia, *Event logs for the analysis of software failures: A rule-based approach.* Software Engineering, IEEE Transactions on, 2013. **39**(6): p. 806-821.
6. Jayathilake, D. *Towards structured log analysis*. in *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*. 2012. IEEE.
7. Lee, Y. and Y. Lee. *Detecting ddos attacks with hadoop*. in *Proceedings of The ACM CoNEXT Student Workshop*. 2011. ACM.
8. Yu, H. and D. Wang. *Mass log data processing and mining based on Hadoop and cloud computing*. in *Computer Science & Education (ICCSE), 2012 7th International Conference on*. 2012. IEEE.
9. Therdphapiyanak, J. and K. Piromsopa. *Applying Hadoop for log analysis toward distributed IDS*. in *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*. 2013. ACM.
10. Cheon, J. and T.-Y. Choe, *Distributed processing of snort alert log using hadoop.* International Journal of Engineering and Technology, 2013. **5**(3): p. 2685-2690.
11. Gunasekaran, R., et al., *Real-Time System Log Monitoring/Analytics Framework.*

12. Subbulakshmi, T., G. Mathew, and S.M. Shalinie, *Real time classification and clustering of IDS alerts using machine learning algorithms.* International journal of Artificial & Application, 2010. **1**(1): p. 20.

13. Moharil, B., et al., *Real Time Generalized Log File Management and Analysis using Pattern Matching and Dynamic Clustering.* International Journal of Computer Applications, 2014. **91**(16).

14. Vorapongkitipun, C. and N. Nupairoj. *Improving performance of small-file accessing in Hadoop*. in *Computer Science and Software Engineering (JCSSE), 2014 11th International Joint Conference on*. 2014. IEEE.

15. Liu, X., N. Iftikhar, and X. Xie. *Survey of real-time processing systems for big data*. in *Proceedings of the 18th International Database Engineering & Applications Symposium*. 2014. ACM.

16. Zaharia, M., et al. *Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing*. in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*. 2012. USENIX Association.

17. Zaharia, M., et al. *Discretized streams: an efficient and fault-tolerant model for stream processing on large clusters*. in *Presented as part of the*. 2012.

18. Hadoop, A. *Hadoop Homepage. http://hadoop.apache.org*.

19. da Silva Morais, T., *Survey on Frameworks for Distributed Computing: Hadoop, Spark and Storm.*

20. Flume, A. *https://flume.apache.org*.

21. Spark, A. *Spark Homepage http://spark.apache.org*.

22. Zhang, X. and F. Xu. *Survey of research on big data storage*. in *Distributed Computing and Applications to Business, Engineering & Science (DCABES), 2013 12th International Symposium on*. 2013. IEEE.

23. Chen, C., Q. Pei, and L. Ning. *Forecasting 802.11 Traffic Using Seasonal ARIMA Model*. in *Computer Science-Technology and Applications, 2009. IFCSTA'09. International Forum on*. 2009. IEEE.

24. Debusschere, V. and S. Bacha. *Hourly server workload forecasting up to 168 hours ahead using Seasonal ARIMA model*. in *2012 IEEE International Conference on Industrial Technology*. 2012.

25. Hanbanchong, A. and K. Piromsopa. *SARIMA based network bandwidth anomaly detection*. in *Computer Science and Software Engineering (JCSSE), 2012 International Joint Conference on*. 2012. IEEE.

26. Haslett, J. and A. Raftery, *Space-time modeling with long-memory dependence: assessing Ireland's wind-power resource. Technical report*. 1987, Washington Univ., Seattle (USA). Dept. of Statistics.

27. Pena, E.H., et al. *Anomaly detection using digital signature of network segment with adaptive ARIMA model and Paraconsistent Logic*. in *Computers and Communication (ISCC), 2014 IEEE Symposium on*. 2014. IEEE.

ประวัติผู้เขียนวิทยานิพนธ์

นายพิทยุตม์ ตั้งสัจจะธรรม เกิดเมื่อวันที่ 8 ธันวาคม พ.ศ. 2533 ที่จังหวัดนครศรีธรรมราช สำเร็จการศึกษาระดับปริญญาบัณฑิต หลักสูตรวิศวกรรมศาตรบัณฑิต สาขาวิศวกรรมคอมพิวเตอร์ มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เมื่อปี พ.ศ. 2556 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชา วิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปี พ.ศ. 2556