ตัวจำลองการทำงานในสำนักงานบนระบบจัดการกระแสงานอูซซี
An Office Work Simulator Based on the Oozie Workflow Management System

ขนิษฐา พรหมสกุล

Kanittha Promsakul

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์

A Thesis Submitted in Fulfillment of the Requirements for the Degree of Master of Engineering in Computer Engineering
Prince of Songkla University

2560

ตัวจำลองการทำงานในสำนักงานบนระบบจัดการกระแสงานอูซซี
An Office Work Simulator Based on the Oozie Workflow Management System

ขนิษฐา พรหมสกุล
Kanittha Promsakul

วิทยานิพนธ์นี้สำหรับการศึกษาตามหลักสูตรปริญญา
วิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์
มหาวิทยาลัยสงขลานครินทร์
A Thesis Submitted in Fulfillment of the Requirements for the
Degree of Master of Engineering in Computer Engineering
Prince of Songkla University
2560

ชื่อวิทยานิพนธ์ ตัวจำลองการทำงานในสำนักงานบนระบบจัดการกระแสงานอูซซี
ผู้เขียน นางสาวขนิษฐา พรหมสกุล
สาขาวิชา วิศวกรรมคอมพิวเตอร์

อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

.....
(ดร.สมชัย หลิมศิโรรัตน์)

คณะกรรมการสอบ

.....ประธานกรรมการ
(ดร.อนันท์ ชกสุริวงค์)

.....กรรมการ
(ดร.สมชัย หลิมศิโรรัตน์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.พิชญา ตัณฑัยย์)

.....กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.เดือนเพ็ญ กชกรจารุพงศ์)

บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ อนุมัติให้นับวิทยานิพนธ์ฉบับนี้สำหรับการศึกษา ตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์

.....
(รองศาสตราจารย์ ดร.ดำรงศักดิ์ ฟ้ารุ่งสาง)
คณบดีบัณฑิตวิทยาลัย

ขอรับรองว่า ผลงานวิจัยนี้มาจากการศึกษาวิจัยของนักศึกษาเอง และได้แสดงความขอบคุณบุคคลที่มีส่วนช่วยเหลือแล้ว

ลงชื่อ .....
(ดร.สมชัย หลิมศิโรรัตน์)
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก

ลงชื่อ .....
(นางสาวขนิษฐา พรหมสกุล)
นักศึกษา

ข้าพเจ้าขอรับรองว่า ผลงานวิจัยนี้ไม่เคยเป็นส่วนหนึ่งในการอนุมัติปริญญาในระดับใดมาก่อน และไม่ได้ถูกใช้ในการยื่นขออนุมัติปริญญาในขณะนี้

ลงชื่อ .....
(นางสาวขนิษฐา พรหมสกุล)
นักศึกษา

ชื่อวิทยานิพนธ์ ตัวจำลองการทำงานในสำนักงานบนระบบจัดการกระแสงานอูซซี
ผู้เขียน นางสาวขนิษฐา พรหมสกุล
สาขาวิชา วิศวกรรมคอมพิวเตอร์
ปีการศึกษา 2560

บทคัดย่อ

วิทยานิพนธ์ฉบับนี้ ได้จัดทำระบบการจำลองกระแสงานอัตโนมัติ ที่สามารถรองรับการขยายตัวหรือเติบโตขึ้นของระบบงานของสำนักงานหรือในองค์กร เพื่อให้ระบบสามารถยังคงใช้งานได้เมื่อมีการปรับขนาดใหญ่ขึ้น โดยนำระบบการประมวลผลแบบกระจายมาใช้ในระบบการจำลอง ทั้งการเก็บข้อมูล เครื่องมือสำหรับจัดการกระแสงานตามเวลา และการประมวลผล ซึ่งงานวิจัยนี้จะจำลองบุคลากรและตำแหน่งการทำงาน เพื่อใช้ในการดำเนินงานและประมวลผลกระแสงาน ผลลัพธ์ของงานวิจัยมุ่งเน้นถึงการลดเวลาความล่าช้าลงของเวลาที่แต่ละบุคคลใช้ในการรอก่อนการประมวลผลกระแสงาน เพื่อให้การประมวลผลการกระทำของกระแสงานในทุกขั้นตอนไม่เกิดความล่าช้าและประมวลผลสำเร็จได้รวดเร็วขึ้น

ระบบการจำลองมีฟังก์ชันการทำงานที่ใช้ในการควบคุมการประมวลผลของกระแสงาน คือฟังก์ชันเริ่ม หยุด หยุดการทำงานชั่วขณะ บันทึกการประมวลผลและการส่งออกข้อมูล โดยผู้ใช้หรือผู้ดูแลระบบสั่งการทำงานผ่านเว็บส่วนติดต่อผู้ใช้งาน ส่วนการออกแบบการเก็บข้อมูลเพื่อให้สามารถรองรับข้อมูลที่มีจำนวนมากขึ้น จึงเลือกใช้ Apache HBase เป็นสถาปัตยกรรมฐานข้อมูลที่เป็นแบบไม่มีความสัมพันธ์กัน ทำให้มีความยืดหยุ่นและสามารถขยายขนาดได้ การประมวลผลกระแสงานเลือกใช้เครื่องมือบนระบบการประมวลผลแบบกระจายคือ Apache Oozie และ Apache Hadoop ในการจัดการกระแสงานตามลำดับเวลาและประมวลผลกระแสงาน ผลลัพธ์ของการจำลอง 100 กระแสงาน พบความล่าช้า เมื่อเพิ่มบุคคลในบทบาทงานที่ล่าช้าเป็น 2 คน ทำงานได้ทันเวลาและค่าความล่าช้าเป็น 0%

Thesis Title An Office Work Simulator Based on the Oozie Workflow Management System

Author Miss Kanittha Promsakul

Major Program Computer Engineering

Academic Year 2017

ABSTRACT

This thesis presents a simulation of automatic workflow management system to support any office or organization expansion. The goal of this research is the scalability of workflow simulation which can be utilized even the system of organization is scaled. Distributed processing system is used for both data collection and workflow execution of the simulation system. The number of person and work roles is considered to be applied in this proposed workflow simulator for the purpose of execution and processing workflow to make the simulation more accorded to business conditions. This research focused on decreasing of delay time which is waiting time of each person before working. Therefore, the overall process of workflow has less delay time and succeed.

We proposed the simulation system to control the workflow simulation processing. The main functions are start, stop, pause, log, and export data. User or administrator controls the simulator via the web user interface. Data storing is designed and implemented to support big data processing. Therefore, the Apache HBase was adopted to store the workflows and workers data for the flexibility and scalability. The Apache HBase is a database which has non-relation of storing data architecture. Furthermore, we considered to adopt both the Apache Oozie and the Apache Hadoop which are the workflow engine for workflow simulation execution on distributed system. We tested by running 100 workflow simulation. The results showed that it causes the delay time in some of the work roles in workflow process. In addition, we added 2 peoples into the work roles which has maximum delay time, and then retesting. The retesting result showed delay time is 0%.

กิตติกรรมประกาศ

ขอขอบพระคุณ ดร.สมชัย หลิมศิโรรัตน์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ที่กรุณาเสียสละเวลาให้คำปรึกษา ชี้แนะแนวทางการดำเนินงานวิจัย ให้กำลังใจและกระตุ้นให้ข้าพเจ้ามีความมุ่งมั่นในการทำงานวิจัย พร้อมทั้งสนับสนุนอุปกรณ์ในการดำเนินงานเป็นอย่างดี ตลอดจนตรวจสอบแก้ไขการเผยแพร่การประชุมวิชาการและวิทยานิพนธ์ฉบับนี้เป็นอย่างดี จนกระทั่งวิทยานิพนธ์นี้สำเร็จสมบูรณ์

ขอขอบพระคุณ ดร.อนันท์ ชกสุริวงค์ ที่กรุณาเสียสละเวลาเป็นประธานกรรมการสอบวิทยานิพนธ์ ให้คำแนะนำที่เป็นประโยชน์ ตลอดจนตรวจสอบแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร.พิชญา ตัณฑัยย์ และ ผู้ช่วยศาสตราจารย์ ดร.เดือนเพ็ญ กชกรจารุพงศ์ ที่กรุณาเสียสละเวลาเป็นกรรมการสอบวิทยานิพนธ์ ให้คำแนะนำที่เป็นประโยชน์ ตลอดจนตรวจสอบแก้ไขวิทยานิพนธ์ให้มีความสมบูรณ์

ขอขอบพระคุณ บัณฑิตวิทยาลัย มหาวิทยาลัยสงขลานครินทร์ วิทยาเขตหาดใหญ่ ที่ให้การสนับสนุนทุนในการทำวิจัยในครั้งนี้

ขอขอบพระคุณ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ที่กรุณาให้ทุนสนับสนุนการเดินทาง และเผยแพร่ผลงานการประชุมวิชาการในงานวิจัยแก่ข้าพเจ้า

ขอขอบพระคุณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยสงขลานครินทร์ ที่กรุณาให้ความช่วยเหลือและประสานงาน ให้ความสะดวกในการดำเนินงานวิจัย จนกระทั่งงานวิจัยนี้สำเร็จสมบูรณ์

ขอขอบพระคุณ คณาจารย์ บุคลากร นักศึกษาปริญญาเอก นักศึกษาปริญญาโท ภาควิชาวิศวกรรมคอมพิวเตอร์ทุกคน ที่ให้คำปรึกษาและการช่วยเหลือ พร้อมทั้งเป็นกำลังใจในการทำงานตลอดมาเป็นอย่างดี

และสุดท้ายนี้ ข้าพเจ้าขอน้อมรำลึกถึงพระคุณ คุณตา คุณยาย มารดา และครอบครัวที่ส่งเสริม สนับสนุน และให้กำลังใจแก่ข้าพเจ้าเสมอมาจนสำเร็จการศึกษา

ขนิษฐา พรหมสกุล

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

รายการตาราง

หน้า

รายการภาพประกอบ

หน้า

รายการภาพประกอบ (ต่อ)

หน้า

สัญลักษณ์คำย่อและตัวย่อ

DAG	Directed Acyclic Graph
DBMS	Database Management System
HDFS	Hadoop Distributed File System
UIMS	User Interface Management System
WfMC	Workflow Management Coalition
WfMS	Workflow Management System
XML	Extensible Markup Language

บทที่ 1
บทนำ

1.1 ที่มาและความสำคัญของการวิจัย

การจะทำให้องค์กรประสบความสำเร็จและไปถึงเป้าหมายอย่างรวดเร็ว มีประสิทธิภาพและประสิทธิผล ต้องอาศัยการบริหารจัดการที่ดี เช่น การบริหารจัดการทรัพยากรบุคคล การบริหารจัดการกระบวนการผลิต และการบริหารจัดการเวลา เป็นต้น ซึ่งการบริหารจัดการงานต่างๆ เหล่านี้ จากเดิมใช้เอกสารเพียงอย่างเดียวในการดำเนินงาน ซึ่งต่อมาคอมพิวเตอร์ได้เข้ามามีบทบาทในการทำงานมากขึ้น มีโปรแกรมประยุกต์ที่ใช้ร่วมกันในองค์กร และได้มีการพัฒนาการเก็บข้อมูลโดยใช้การจัดการระบบฐานข้อมูล (Database Management System: DBMS) (Wil van der Aalst 1998) แต่การทำงานยังไม่มีประสิทธิภาพในเรื่องของการกำหนดบทบาทหน้าที่และการไหลของข้อมูล (dataflow) ในช่วงปี คศ. 1970 เริ่มมีการนำแนวความคิดของระบบการจัดการกระแสงาน (Workflow Management System: WfMS) มาช่วยในการจัดการกระบวนการทำงานขององค์กร เพื่อเพิ่มประสิทธิภาพและประสิทธิผลให้กับระบบการทำงานในองค์กร (Edward A.Stohr และ J. Leon Zhao 2001) ซึ่งแนวคิดของระบบการจัดการกระแสงานคือ การส่งข้อมูลอัตโนมัติผ่านบุคคลหนึ่งไปยังบุคคลหนึ่ง หรือผ่านเครือข่ายคอมพิวเตอร์ภายในขององค์กร ตามกฎระเบียบที่ได้กำหนดไว้ขององค์กรนั้นๆ โดยมีจุดประสงค์ในเรื่องของการจัดลำดับขั้นตอนการทำงาน การเรียกใช้งานที่เหมาะสำหรับบุคคล หรือเป็นข้อมูลที่เกี่ยวข้องกับกิจกรรมที่กระทำ ระบบการจัดการกระแสงานที่ดีจะทำให้การบริหารเวลามีคุณภาพดีขึ้น ใช้เวลาทำงานลดลง ประหยัดค่าใช้จ่าย ควบคุมความถูกต้องได้ และผู้ใช้งานมีความพึงพอใจในการปฏิบัติหน้าที่ของตนเอง (SHI Meilin และคณะ 1998; Edward A.Stohr และ J. Leon Zhao 2001)

ในปัจจุบันนี้เทคโนโลยีกระแสงานโดยใช้ระบบการจัดการกระแสงานอัตโนมัติ มีการขยายและการนำไปใช้เพิ่มมากขึ้นในหลายขอบเขตงาน เนื่องจากกระแสงานอัตโนมัติเป็นส่วนสำคัญในการดำเนินงาน สำหรับพัฒนาธุรกิจขององค์กรใดๆ เพื่อให้บรรลุเป้าหมายสูงสุดที่กำหนดไว้ เพื่อดำเนินการเพิ่มผลผลิตของอุตสาหกรรมให้มากขึ้น หรือกระทั่งใช้ระบบจัดการกระแสงานเพื่อพัฒนาระบบสนับสนุนการตัดสินใจ (Decision Support System: DSS) สำหรับการจัดการทรัพยากรทางธรรมชาติหรือพลังทดแทน (Sergey V. Ivanov และคณะ 2013; Adela Bara และคณะ 2014) นอกจากนี้ยังต้องการระบบกระแสงานที่มีความยืดหยุ่นสูง เนื่องจากระบบงานในปัจจุบันเป็นแบบไดนามิก หรือมีการเปลี่ยนแปลงโครงสร้างไปตามธรรมชาติขององค์กรที่มีความเจริญก้าวหน้าทาง

เทคโนโลยี และการเปลี่ยนไปตามวัฒนธรรมของยุคสมัย ได้มีการนำระบบจัดการกระแสงานประยุกต์รวมกับการจัดการองค์ความรู้ของบุคคลหรือองค์กร (Knowledge Management: KM) เพื่อเป็นแนวทางในการจัดการกระบวนการทางธุรกิจและสร้างความพึงพอใจให้กับบุคลากรในองค์กร (Ricardo Anderson และ Gunjan Mansingh 2017) และนอกจากนี้เทคโนโลยีได้มีการพัฒนาเพิ่มมากขึ้น อุปกรณ์อิเล็กทรอนิกส์หรืออินเตอร์เน็ตเข้ามามีบทบาทและมีการใช้งานมากขึ้นในปัจจุบัน การเติบโตขององค์กรทำให้ระบบกระแสงานขององค์กรขยายขึ้นเช่นกัน จึงทำให้การจัดเก็บข้อมูลมีจำนวนมากขึ้น ข้อมูลมีขนาดใหญ่ขึ้น และมีการขยายการเก็บข้อมูลเพื่อให้รองรับการเพิ่มขึ้นของปริมาณข้อมูล เทคโนโลยีของเครื่องการคำนวณเพื่อประมวลผลข้อมูลแบบกลุ่มเมฆ (Cloud Computing) หรือเครือข่ายของสิ่งที่เป็นตัวตนจับต้องได้ (Internet of things: IoT) (Li Liu และคณะ 2014) ถูกนำมาใช้ในการวิเคราะห์และแก้ปัญหาของข้อมูลได้ เช่น แก้ปัญหาเรื่องคอขวดเมื่อมีการประมวลผลข้อมูลขนาดใหญ่และจำนวนมาก หรือเพื่อเพิ่มความเร็วในการประมวลผลที่เป็นแบบกระจาย โดยอาศัยระบบจัดการกระแสงานที่มีหลักเกณฑ์ทางวิทยาศาสตร์ (Scientific Workflow Management System: SWfMS) ซึ่งเป็นเครื่องมือที่ช่วยเพิ่มประสิทธิภาพสำหรับการประมวลผลกระแสงานและใช้จัดการชุดของข้อมูลที่มีจำนวนมากขึ้นในปัจจุบัน (Ji Liu และคณะ 2015; Ewa Deelman และคณะ 2015; Xiu Li และคณะ 2016) นอกจากนี้ระบบการจำลองเป็นอีกเครื่องมือที่ช่วยในการวิเคราะห์คุณภาพ และเพิ่มประสิทธิภาพของกระบวนการดำเนินงานได้ โดยเฉพาะการจำลองระบบจัดการกระแสงานเพื่อใช้ในงานด้านต่างๆ เช่น การจำลองจัดการกระแสงานเพื่อสนับสนุนระบบการตัดสินใจ จำลองการทำงานของทรัพยากรบุคคลในองค์กรให้มีประสิทธิภาพสูงสุด และการจำลองระบบจัดการกระแสงานที่มีหลักเกณฑ์ทางวิทยาศาสตร์ ในการประมวลผลบนระบบที่เป็นแบบกระจาย เป็นต้น (Anne Rozinat และคณะ 2009; Ying Liu และคณะ 2012; Weiwei Chen และ Ewa Deelman 2012; Sergey V. Ivanov และคณะ 2013) ซึ่งระบบการจำลองกระแสงานที่มีอยู่ในปัจจุบันนั้น ยังไม่สามารถรองรับการเติบโตหรือขยายตัวของระบบงานในองค์กรหรือหน่วยงานได้

วิทยานิพนธ์นี้มีความสนใจที่จะออกแบบและพัฒนาระบบการจำลองกระแสงานขององค์กรใดๆ เพื่อเป็นเครื่องมือในการวิเคราะห์ภาระงานของบุคคลที่รับผิดชอบอยู่ ให้เหมาะสมในเรื่องบทบาทหน้าที่และความพร้อมในการทำงาน แก้ปัญหาภาระงานที่รับผิดชอบมากหรือน้อยไปสำหรับบุคคลนั้นๆ กระจายงานตามหน้าที่ความรับผิดชอบให้กับบุคลากรในองค์กรได้อย่างเท่าเทียม และเมื่อมีการขยายงานของระบบสามารถทำได้ง่ายขึ้น ซึ่งในปัจจุบันงานวิจัยระบบจัดการกระแสงานส่วนมากเป็นแบบระบบผู้รับและผู้ให้บริการ (Client-Server System) ทำให้ยังมีข้อจำกัดเมื่อองค์กรขยายขนาดและระบบกระแสงานก็ขยายเพิ่มขึ้น งานวิจัยนี้จึงได้จัดทำขึ้นเพื่อแก้ปัญหาดังกล่าว เพื่อให้

สามารถรองรับระบบกระแสงานที่ขยายขนาดขององค์กรในอนาคตได้ (Scalable and Extensible System)

งานวิจัยนี้มีจุดประสงค์ เพื่อพัฒนาการจำลองระบบจัดการกระแสงานอัตโนมัติสำหรับองค์กรหรือหน่วยงานใดๆ ที่รองรับการปรับขยายขนาดหรือเปลี่ยนแปลงการบริหารจัดการกระแสงานขององค์กร ผู้วิจัยจึงได้ออกแบบและพัฒนาการจำลองระบบจัดการกระแสงานอยู่บนระบบการคำนวณแบบกลุ่มเมฆ การประมวลผลแบบกระจาย (Distributed Processing) ได้ออกแบบการเก็บข้อมูลในการจำลองและผลลัพธ์ของการประมวลผลกระแสงานเป็นแบบไม่มีความสัมพันธ์กัน เพื่อให้มีความยืดหยุ่นของระบบและสามารถเก็บข้อมูลได้จำนวนมาก ในการพัฒนาการจำลองระบบจัดการกระแสงานอัตโนมัตินี้ สำหรับใช้ประมวลผลการกระทำของกระแสงาน การจำลองจะพิจารณาการจัดการกระแสงานจากตำแหน่งหน้าที่ที่รับผิดชอบของบุคลากร ให้ตรงบทบาทที่ระบุไว้ในการกระทำของกระแสงาน พิจารณาเรื่องของเวลาที่ใช้ในการดำเนินงาน เช่น เวลาที่แต่ละบุคคลใช้ในการรอเพื่อจะประมวลผลการกระทำของกระแสงาน และเวลารวมในการประมวลผลของทั้งกระแสงาน นอกจากนี้จะพิจารณาลักษณะของกระแสงานประเภทต่างๆ ในการจำลอง เพื่อใช้ในการวิเคราะห์การทำงานของบุคคลในเรื่องเวลาในการประมวลผลกระแสงานนั้น สำหรับการพิจารณาปรับเปลี่ยนให้บุคคลอื่นที่มีหน้าที่ความรับผิดชอบเดียวกันมาทำงานนั้นแทนในกรณีการประมวลผลงานเกิดความล่าช้า หรือพิจารณาเพิ่มคนทำงานของตำแหน่งงานนั้นในกรณีที่บุคคลไม่เพียงพอ เพื่อก่อให้เกิดประสิทธิภาพและประสิทธิผลสูงสุดในการดำเนินงานของสมาชิกในองค์กร เพราะฉะนั้นการจำลองระบบจัดการกระแสงานอัตโนมัตินี้ สามารถใช้เป็นแนวคิดพื้นฐานสำหรับการพัฒนาองค์กรให้มีประสิทธิภาพ และรองรับระบบงานขององค์กรที่ขยายขึ้นได้ในอนาคต

1.2 วัตถุประสงค์

1. เพื่อจำลองระบบจัดการกระแสงานในองค์กรอัตโนมัติ และรองรับการขยายขนาดของระบบงานในองค์กรได้
2. เพื่อจำลองบุคคลในแต่ละบทบาทการทำงานด้วยระบบจัดการกระแสงาน และการวิเคราะห์ค่าเวลาที่ใช้รอก่อนการประมวลผล

1.3 ประโยชน์ที่คาดว่าจะได้รับ

1. เป็นแนวคิดพื้นฐานในการจัดการเส้นทางของระบบกระแสงานในองค์กร
2. ช่วยในการวิเคราะห์เพื่อเพิ่มหรือลดจำนวนบุคลากรขององค์กรได้

บทที่ 2
ทฤษฎีและหลักการ

ในบทนี้จะกล่าวถึงระบบจัดการกระแสงาน ประวัติในอดีตจนถึงปัจจุบันของระบบจัดการกระแสงาน มาตรฐานและประเภทของระบบจัดการกระแสงาน และระบบการประมวลผลแบบกระจายของข้อมูลที่มีขนาดใหญ่ รวมไปถึงการเปรียบเทียบเครื่องมือสำหรับประมวลผลระบบกระแสงาน และการเลือกใช้เครื่องมือในการประมวลกระแสงานที่เหมาะสมสำหรับงานวิจัยนี้

2.1 ระบบจัดการกระแสงาน

องค์กรจะไปถึงเป้าหมายที่ตั้งไว้อย่างมีประสิทธิภาพ จะต้องมีระบบการจัดการกระแสงานที่ดีซึ่งช่วยจัดการกระบวนการทางธุรกิจ เพื่อลดเวลาลง ประหยัดค่าใช้จ่าย ควบคุมความถูกต้องได้มากขึ้น และสร้างความพึงพอใจของผู้ปฏิบัติงานมากขึ้น อับดับแรกควรจะทราบถึงภาพรวมของระบบกระแสงานจากอดีตถึงปัจจุบัน มาตรฐานของการจัดการกระแสงาน รวมถึงการแบ่งประเภทของระบบกระแสงาน และแนวโน้มงานวิจัยของระบบจัดการกระแสงาน

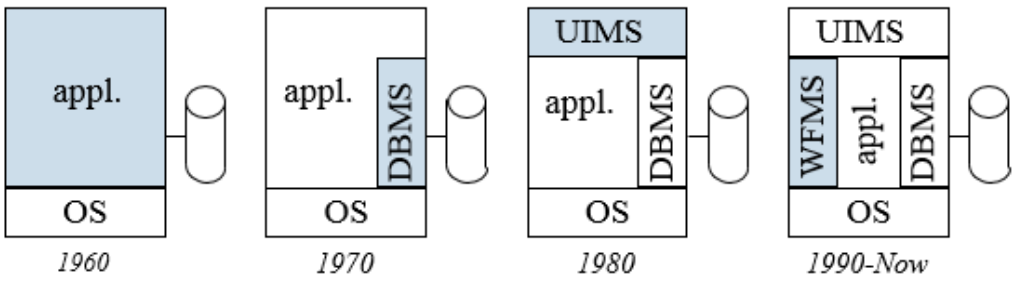
2.1.1 คำนิยาม

ระบบจัดการกระแสงาน (Workflow Management System หรือ WfMS) คือขั้นตอนการประมวลผลงานอัตโนมัติของระบบงาน ซึ่งประกอบด้วยลำดับขั้นตอนของกิจกรรมตั้งแต่เริ่มต้นจนสิ้นสุดกระบวนการ มีเป้าหมายที่ใช้ภายในองค์กรหรือระหว่างองค์กร ข้อมูลหรืองานของแต่ละกิจกรรมจะถูกประมวลผลโดยบุคคลหรือคอมพิวเตอร์ ตามกฎระเบียบที่ได้ตั้งไว้ขององค์กรนั้นๆ เพื่อให้เกิดประโยชน์สูงสุดและบรรลุเป้าหมายขององค์กร (Edward A. Stohr และ J. Leon Zhao 2001; SHI Meili และคณะ 1998)

2.1.2 ประวัติของระบบจัดการกระแสงาน

ภาพประกอบที่ 2-1 แสดงถึงวิวัฒนาการของระบบสารสนเทศ (Information System) จุดเริ่มต้นก่อนจะมีระบบจัดการกระแสงานเกิดขึ้น จากในอดีตตั้งแต่ปี ค.ศ. 1970 จนถึงปัจจุบัน ซึ่งจะเป็นประโยชน์และสนับสนุนกระบวนการทางธุรกิจในปัจจุบัน แบ่งโครงสร้างส่วนประกอบของระบบข้อมูลออกเป็น 4 ยุคด้วยกัน ดังนี้

- ปีค.ศ. 1960 ใช้โปรแกรมประยุกต์ (Application) ที่ประมวลผลอยู่บนระบบปฏิบัติการ (OS) และไม่มีการเชื่อมต่อกับเครื่องคอมพิวเตอร์อื่นๆ ในการประมวลผลข้อมูล จะจัดเก็บและเรียกใช้ข้อมูลใน Data Storage หรือเรียกว่า Standalone Application
- ปีค.ศ. 1970 ได้มีการพัฒนาระบบจัดการฐานข้อมูล Database Management System (DBMS) มาใช้ร่วมกับระบบโปรแกรมประยุกต์
- ปีค.ศ. 1980 ได้มีการพัฒนาส่วนของการเชื่อมต่อกับผู้ใช้งานหรือที่เรียกว่า User Interface Management System (UIMS)
- ปีค.ศ. 1990 จนถึงปัจจุบันได้พัฒนาส่วนชุดคำสั่ง (Software) กระแสงานโปรแกรมประยุกต์ถูกพัฒนาขึ้นเพื่อการดำเนินงานทางธุรกิจ ทำให้เกิดระบบจัดการกระแสงานขึ้น



ภาพประกอบที่ 2-1 ประวัติระบบจัดการกระแสงาน (Wil van der Aalst 1998)

2.1.3 มาตรฐานระบบจัดการกระแสงาน

มาตรฐานของระบบจัดการกระแสงานแต่ละองค์กรมีความแตกต่างกัน เพื่อให้กระแสงานระหว่างองค์กรสามารถทำงานร่วมกันได้ จึงมีกลุ่มความร่วมมือในการสร้างมาตรฐานการจัดการกระแสงานขึ้น ซึ่งประกอบไปด้วยมาตรฐานขององค์กรได้แก่ WfMC, BPMI, OMG, W3C และ OASIS (Jan Mendling 2006)

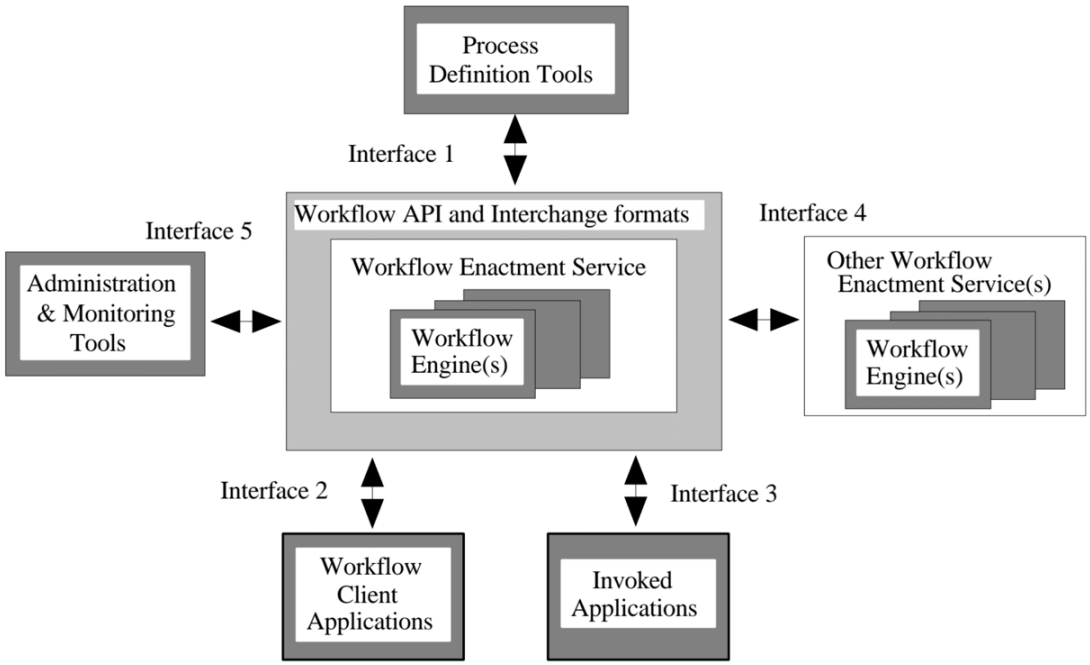
- WfMC (Workflow Management Coalition) เป็นองค์กรแรกที่มีการส่งเสริมมาตรฐานกระแสงาน และมีภาษา XPDL (XML Process Definition Language) เป็นมาตรฐาน

- BPMI (Business Process Management Interface) เป็นองค์กรที่จัดตั้งมาตรฐานการจัดการกระบวนการทางธุรกิจ มีการเผยแพร่ภาษา Business Process Modelling Language (BPML) และ XML-based ขึ้น
- OMG (Object Management Group) ได้รวมตัวกับ BPMI และมีภาษามาตรฐานคือ BPMN (Business Process Modelling Notation)
- W3C (World Wide Web Consortium) มีการเผยแพร่หลายๆ มาตรฐานสำหรับผู้ให้บริการเว็บไซต์
- OASIS (Organization for the Advancement of Structured Information Standards) มีสมาชิกเป็นบริษัทขนาดหลายบริษัทเช่น IBM, Microsoft, Adobe และ Hewlett-PackardOracle เป็นต้น โดยที่กำหนดมาตรฐานที่เรียกว่า BPEL (Business Process Execution Language) ซึ่งเป็นมาตรฐานแบบแสวงหากำไร การใช้งานจึงต้องมีค่าใช้จ่าย

สำหรับงานวิจัยนี้สนับสนุนมาตรฐาน WfMC เนื่องจากเป็นมาตรฐานแบบเปิด (Open Standard) ที่ถูกก่อตั้งขึ้นมาเพื่อให้เป็นสากล ให้สามารถติดต่อสื่อสารและทำงานร่วมกันได้ และเป็นองค์กรที่ไม่แสวงหาผลกำไร และเพื่อให้สมาชิกภายในกลุ่มสามารถทำงานระหว่างกันได้ (WfMC 2017) สมาชิกขององค์กร WfMC ประกอบด้วย IBM, Hewlett-Packard, Fujitsu, ICL, Staffware และบริษัทอื่นๆ อีก 300 กว่าบริษัท ได้มีการออกแบบความร่วมมือของมาตรฐานในองค์กร ที่มีเครื่องมือทางกระแสงานและฐานข้อมูลส่วนกลาง เชื่อมต่อกับส่วนต่อประสาน (Interface) 5 ส่วน ดังแสดงในภาพประกอบที่ 2-2 ดังต่อไปนี้

- Interface 1 Process Definition Tool เป็นการนิยามเครื่องมือที่ใช้พัฒนากระบวนการทำงานของกระแสงาน
- Interface 2 Workflow Client Application ซึ่งเป็นโปรแกรมที่ผู้ใช้งานร้องขอการให้บริการเพื่อติดต่อกับกระแสงานในระบบ และติดต่อกับเครื่องประมวลผลกระแสงาน (Workflow Engine) เพื่อที่จะใช้ทำงานของแต่ละคน
- Interface 3 Invoked Application Interface ซึ่งเป็นชุดคำสั่งสำหรับติดต่อเพื่อเรียกใช้โปรแกรมหรือบริการภายนอกระบบ
- Interface 4 Other Workflow Enactment Services ซึ่งเป็นมาตรฐานในการทำงานร่วมกันของเครื่องประมวลผลกระแสงาน ที่ทำให้บริษัทหรือองค์กรที่แตกต่างกันสามารถแลกเปลี่ยนกระแสงานกันได้

- Interface 5 Administration & Monitoring Tool เป็นชุดคำสั่งเครื่องมือของผู้ดูแลระบบสำหรับดูแลและเฝ้าสังเกตการณ์



ภาพประกอบที่ 2-2 มาตรฐานระบบกระแสงานของ WfMC (David Hollingsworth 1995)

2.1.4 ประเภทของระบบกระแสงาน

การพัฒนาระบบกระแสงานมีจุดมุ่งหมายเพื่อใช้ในกระบวนการและดำเนินการทางธุรกิจโดยอัตโนมัติ ซึ่งมีพื้นฐานมาจากระบบจัดการเอกสาร การประมวลผลภาพ การประมวลผลแบบฟอร์มและด้านอื่นๆ ในงานวิจัย (SHI Meilin และคณะ 1998) ได้กล่าวถึงประเภทของระบบการจัดการกระแสงาน และจัดประเภทกระแสงานได้ดังต่อไปนี้

- กระแสงานแบบโครงสร้าง (Structured) และกระแสงานแบบเฉพาะกิจ (Ad-hoc) กระแสงานแบบโครงสร้างเป็นกระแสงานที่มีโครงสร้างแน่นอนของข้อมูลทั้งหมดในกระบวนการประมวลผล สามารถหาได้จากการวิเคราะห์แบบจำลอง ส่วนกระแสงานแบบเฉพาะกิจเป็นระบบกระแสงานที่ลดความซ้ำซ้อนของข้อมูลเป็นกระบวนการเปลี่ยนแปลงแบบพลวัต ส่วนมากจะใช้ในระบบกระแสงานแบบง่าย

- กระแสงานที่ใช้เอกสารเป็นหลัก (Document-centric) และกระแสงานที่ใช้การประมวลผลเป็นหลัก (Process-centric) กระแสงานที่ใช้เอกสารเป็นหลักหรือโดยการประมวลผลกระแสงานด้วยเอกสารอิเล็กทรอนิกส์ การประมวลภาพดิจิตอล และกระแสงานที่ใช้การประมวลผลเป็นหลัก จะเน้นไปที่กระบวนการของการดำเนินการทางธุรกิจ
- Email-based และ Database-based กระแสงานแบบ Email-based จะเป็นการกระจายข้อมูลในการประมวลผลของกระแสงาน โดยจะเป็นการส่งข้อความเพื่อแจ้งเตือน ส่วนกระแสงานแบบ Database-based การประมวลผลของทุกขั้นตอนจะเก็บข้อมูลลงบนระบบการจัดการฐานข้อมูล
- Task-pushed และ Goal-pulled กระแสงานแบบ Task-pushed เป็นกระบวนการทำงานตามลำดับขั้นตอน จะไม่สามารถทำขั้นตอนต่อไปได้ถ้าหากขั้นตอนก่อนหน้ายังไม่เสร็จ ส่วนมากจะใช้กับกระแสงานโดยใช้เอกสารเป็นหลัก ส่วนกระแสงานแบบ Goal-pulled สามารถกระจายงานแต่ละขั้นตอนไปประมวลผลพร้อมกันได้ ทำให้ประหยัดเวลาในการประมวลผล เมื่อทุกขั้นตอนประมวลผลจบถือว่าเสร็จการทำงาน

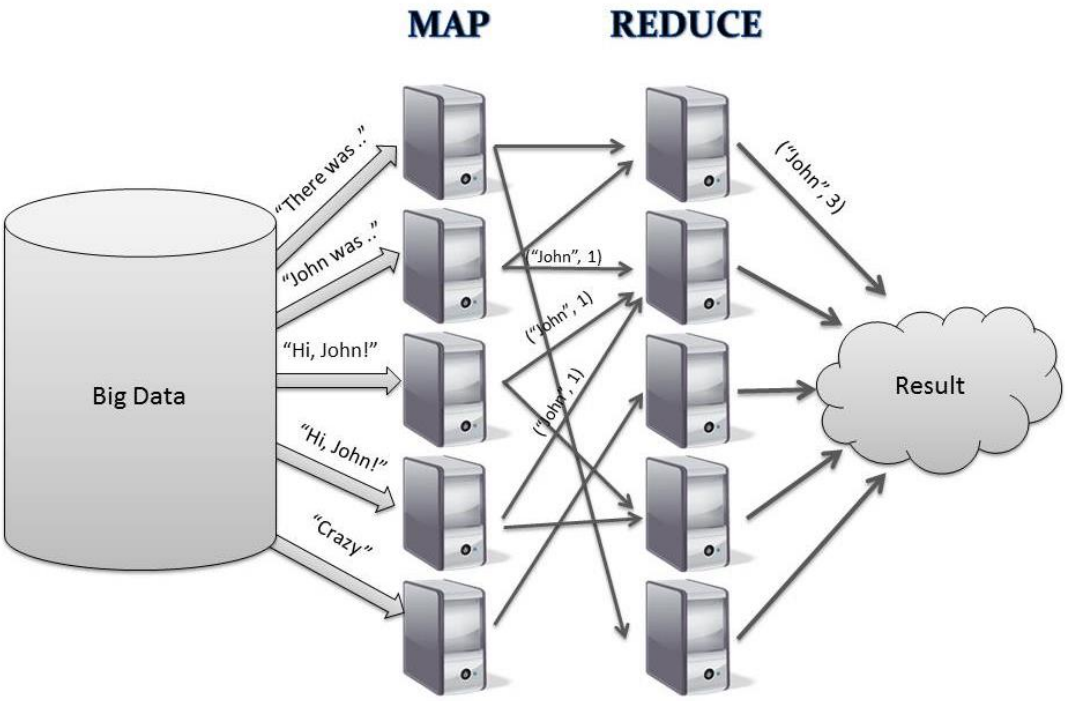
ในปัจจุบันงานวิจัยทางด้านกระแสงานมีแนวโน้มเพิ่มขึ้น เนื่องจากระบบเทคโนโลยีคอมพิวเตอร์เข้าไปมีบทบาทในการทำงานเกือบทุกองค์กร ซึ่งแนวโน้มการวิจัยระบบจัดการกระแสงานในปัจจุบัน เช่น การสร้างระบบกระแสงานให้เป็นแบบพลวัต (Wanjun Huang และคณะ 2004) กระแสงานมีความยืดหยุ่นสามารถที่จะขยายขนาดของระบบงานได้ การจัดการกระแสงานที่เป็นระบบแบบกระจาย (G. Alonso และคณะ 1994) เป็นต้น ซึ่งวิทยานิพนธ์นี้จะสร้างเป็นระบบกระแสงานแบบโครงสร้างแน่นอน แต่มีการบริหารจัดการงานให้เหมาะสมกับภาระงานที่มีอยู่ของผู้ใช้งาน กระแสงานเอกสารเป็นหลัก มีการเก็บข้อมูลบนฐานข้อมูลแบบกระจาย และการจำลองสามารถประมวลผลการกระทำพร้อมกันได้

2.2 Apache Hadoop

เนื่องจากระบบกระแสงานในปัจจุบันส่วนใหญ่เป็นสถาปัตยกรรมแบบเครื่องรับ-ให้บริการ (Client-Server) เมื่อองค์กรขยายขนาดมีผู้ใช้งานเพิ่มขึ้น ทำให้ภาระงานของระบบเพิ่มขึ้น ระบบจัดการกระแสงานจึงตอบสนองได้ช้าลง จึงได้มีการแก้ปัญหาการประมวลผลของข้อมูลที่มีขนาดใหญ่ขึ้น โดยนำสถาปัตยกรรมกระแสงานแบบกลุ่ม (Cluster) เข้ามาใช้งานแทนระบบสถาปัตยกรรมแบบเครื่องรับ-ให้บริการ (Kwang-Hoon Kim และคณะ 2005)

การประมวลผลข้อมูลขนาดใหญ่มีการใช้งานอย่างกว้างขวางมากขึ้น เช่น ทางด้านการเงิน อินเทอร์เน็ต เป็นต้น ซึ่งมีความท้าทายเป็นอย่างมาก ในการเก็บข้อมูลขนาดใหญ่ที่มีการเติบโตอย่างหลีกเลี่ยงไม่ได้ ได้มีเฟรมเวิร์คคือ Apache Hadoop ซึ่งเป็นเฟรมเวิร์คที่ใช้ในการประมวลผลข้อมูลที่มีขนาดใหญ่ โดยการประมวลผลเป็นแบบกระจาย (Anand Loganathan และคณะ 2014; The Apache Software Foundation 2017) มีข้อดีคือ Hadoop เป็นซอฟต์แวร์ที่สามารถนำมาใช้ในการพัฒนาโปรแกรมประยุกต์ได้ฟรี มีการซ่อนความซับซ้อนในการประมวลผลและมีความทนทานต่อความล้มเหลว และสามารถจัดการข้อผิดพลาดได้เองโดยอัตโนมัติ นอกจากนี้ยังมีความน่าเชื่อถือและสามารถปรับขยายขนาดได้ โดย Hadoop มีการจัดการเก็บไฟล์แบบกระจายที่เรียกว่า HDFS (Hadoop Distributed File System) ซึ่งจะเก็บข้อมูลและรองรับการส่งข้อมูลจำนวนมากระหว่างกลุ่ม ทำให้การคำนวณหาผลลัพธ์มีความรวดเร็วขึ้น เนื่องจากแบ่งกลุ่มการประมวลผลเป็นหลายกลุ่ม นอกจากนี้มีวิธีการประมวลผลแบบขนานของชุดข้อมูลขนาดใหญ่ที่เรียกว่า MapReduce (Jeff Dean และ Sanjay Ghemawat 2008; Seema Maitreya และ C.K. Jhab 2015) ซึ่งมีงานวิจัยที่นำ MapReduce มาใช้ในการสร้างระบบกระแสงานแบบกระจาย (Yang Ruan และคณะ 2012)

การจัดการไฟล์แบบ HDFS จะแบ่งเก็บข้อมูลขนาดใหญ่ไปยังโหนดย่อยๆในระบบของ Hadoop เมื่อทำการประมวลผลข้อมูลจะเป็นการประมวลแบบขนานคือ MapReduce ซึ่งมีขั้นตอนในการประมวลผลอยู่ 2 ขั้นตอนด้วยกันคือ ขั้นตอนการ Map เมื่อแบ่งข้อมูลขนาดใหญ่ไปเก็บยังโหนดต่างๆ แล้ว ก็จะส่งฟังก์ชันหรือโปรแกรมที่จะประมวลผลไปยังโหนดที่เก็บข้อมูล เพื่อทำการประมวลผลสิ่งที่ต้องการ เป็นการป้องกันไม่ให้เกิดคอขวดในระบบ ซึ่งในขั้นตอน Map นี้จะเป็นการกรองข้อมูลที่ต้องการหรือประมวลผลลัพธ์ของแต่ละโหนดออกมา หลังจากนั้นจะส่งผลลัพธ์ที่กรองข้อมูลออกมา ไปยังเครื่องคอมพิวเตอร์ในการรวบรวมข้อมูลทั้งหมด แล้วประมวลผลหาคำตอบสุดท้ายที่ต้องการออกมา ซึ่งขั้นตอนนี้เรียกว่าการ Reduce ข้อมูลออกมาเป็นผลลัพธ์สุดท้ายที่ต้องการ ดังภาพประกอบที่ 2-3 ตัวอย่างการประมวลผลแบบขนาน MapReduce ซึ่งต้องการค้นหาคำศัพท์คำว่า “John” จากข้อมูลขนาดใหญ่ (Big Data) ขั้นตอนการ Map จะมีเครื่องคอมพิวเตอร์สำหรับใช้ในการประมวลผลทั้งหมด 5 เครื่อง ซึ่งข้อมูลขนานใหญ่นั้นจะถูกแบ่งเป็น 5 ส่วนให้กับเครื่องคอมพิวเตอร์ทั้ง 5 เครื่อง ช่วยกันประมวลผลหาคำศัพท์ “John” ส่วนขั้นตอนของการ Reduce ก็มีเครื่องคอมพิวเตอร์ทั้งหมด 5 เครื่อง สำหรับประมวลผลรวมคำตอบที่ได้รับจากขั้นตอนการ Map จึงได้ผลลัพธ์ของการค้นหาคำศัพท์คำว่า “John” ซึ่งข้อดีของการประมวลผลแบบ MapReduce จะทนทานต่อความล้มเหลวของระบบได้ดี



ภาพประกอบที่ 2-3 การประมวลผลแบบขนาน MapReduce (Thomas Seidl 2009)

2.3 เครื่องมือสำหรับประมวลผลกระแสงาน

ในการดำเนินการพัฒนาระบบกระแสงาน มีเครื่องมือที่สามารถดำเนินการระบบกระแสงานอยู่หลายตัวด้วยกัน มีทั้งเครื่องมือที่ใช้ในเชิงพาณิชย์ ต้องมีค่าใช้จ่ายและไม่ต้องมีค่าใช้จ่าย ซึ่งเครื่องมือที่ผู้วิจัยสนใจคือ เครื่องมือกระแสงานที่จะพัฒนาอยู่บนระบบการประมวลผลแบบกระจาย

ตารางที่ 2-1 เป็นการเปรียบเทียบเครื่องมือในการสร้างระบบกระแสงาน ซึ่งได้เปรียบเทียบคุณสมบัติต่างๆ แต่ละเครื่องมือมีจุดเด่นจุดด้อยที่แตกต่างกัน เมื่อเปรียบเทียบข้อมูลในเรื่องภาษาของกระแสงาน XML เป็นภาษาที่เข้าใจง่ายและสามารถนำไปใช้ได้กับทุกโปรแกรมแพลตฟอร์มในกลุ่มเมฆเลือกใช้ Hadoop เนื่องจากเป็นซอฟต์แวร์แบบเปิดเสรี มีเครื่องยนต์ที่ใช้แน่นอน และมีการให้บริการเว็บซึ่งจะสะดวกในการติดตามสถานะการประมวลผลของกระแสงาน ในงานวิจัยจึงได้เลือกเครื่องมือสำหรับใช้ในการประมวลผลกระแสงาน คือ Apache Oozie เนื่องจากเป็นเครื่องมือที่ช่วยจัดระบบการจัดตารางของกระแสงาน ซึ่งเป็นซอฟต์แวร์สามารถใช้พัฒนากระแสงานได้เสรี มีข้อดีคือสามารถปรับขยายขนาดได้เมื่อระบบมีขนาดใหญ่ขึ้น และมีความน่าเชื่อถือของระบบซึ่งทนทานต่อความล้มเหลวได้ดี (The Apache Software Foundation 2017)

ตารางที่ 2-1 เปรียบเทียบเครื่องมือกระแสงาน (Anju Bala และ Inderveer Chana 2012)

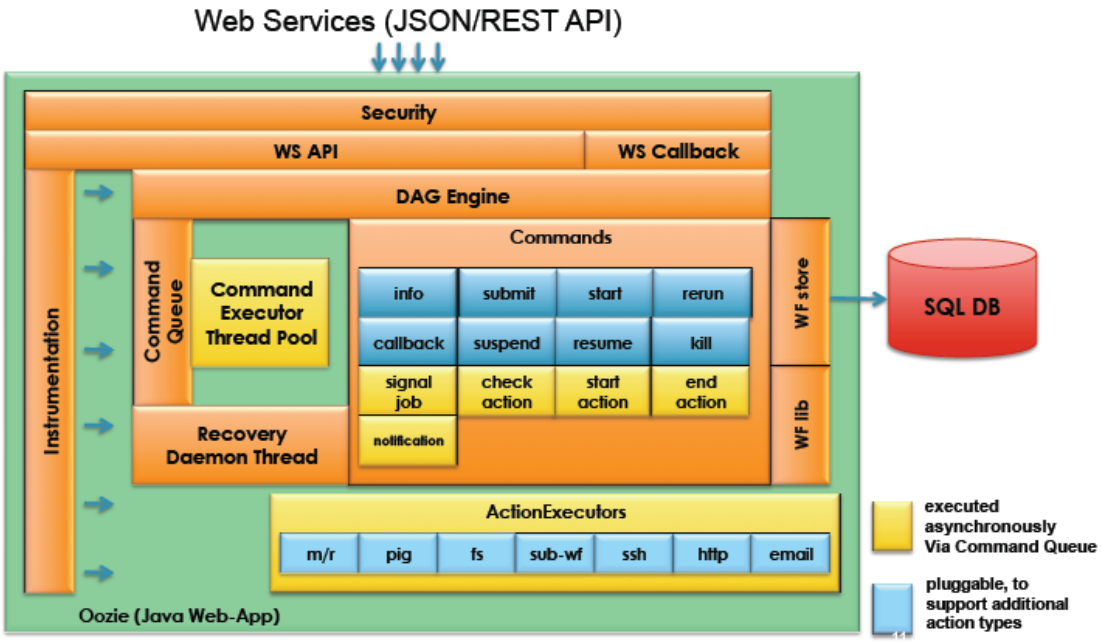
Workflow Engines	Workflow specification language	Cloud Platform	Dependency mechanism	Event notification	Requires installation	Web service
Hamake	XML	Hadoop	Data-driven	No	No	Console/log message
Oozie	XML	Hadoop	Explicit	No	Yes	Web Pages
Azkaban	Text file with key/value pairs	Hadoop	Explicit	No	Yes	Web pages
Cascading	Java API	Hadoop	Explicit	Yes	No	Java API
Orangescape Studio	MYSQL, Oracle, DB2	Google app Engine, Microsoft Azure, IBM	Data driven	Yes	No	Web pages
JBOSS	Java	Octopus	Explicit	Yes	No	Java API
Pegasus	Java	Eucalyptus, Amazon EC2, Open Nebula and so on	Explicit	Yes	Yes	Java API
YAWL	XML	None	Data driven	Yes	Yes	Web Pages

2.3.1 Apache Oozie

Apache Oozie หรือเรียกว่า Oozie (Mohammad Islam และคณะ 2012) เป็นเครื่องมือสำหรับประมวลผลกระแสงานตามเวลา รองรับการประมวลผลงานแบบกระจายบน Hadoop ซึ่งการกระทำของกระแสงานถูกอธิบายด้วยกราฟแบบระบุทิศทาง (Directed Acyclic Graph; DAG) นอกจากนั้น Oozie เป็นระบบที่รองรับการขยายตัว มีความน่าเชื่อถือ และมีความยืดหยุ่นของระบบสูง การเรียกใช้งานในการส่งกระแสงานไปประมวลผลผ่านเครื่องให้บริการเว็บ (Web Service) โดยเรียกใช้ชุดคำสั่ง JSON/REST และการประมวลผลกระแสงานโดยสั่งคำสั่งผ่าน Command line

กระแสงานที่ต้องการประมวลผลเรียกว่างานของกระแสงาน (Workflow Job) และโหนดสำหรับการประมวลผลเรียกว่าการกระทำของกระแสงาน (Workflow Action) สำหรับการออกแบบสถาปัตยกรรมของ Oozie ดังภาพประกอบที่ 2-4 ทุกคำสั่งสามารถที่จะทำงานพร้อมๆ กันทั้งระบบ (Instrumentation) เมื่อมีงานของกระแสงานเข้ามาเครื่องมือกราฟ (DAG Engine) ก็จะตรวจสอบไฟล์กระแสงานว่าอยู่ในรูปแบบที่ได้กำหนดไว้หรือไม่ ถ้าหากไฟล์กระแสงานไม่ได้อยู่ในรูปแบบที่กำหนดของ Oozie ก็จะแจ้งข้อผิดพลาด HTTP Status 400 และถ้าไฟล์กระแสงานถูกต้อง

ระบบก็จะประมวลผลกระแสงานนั้น โดยลักษณะการทำงานจะเป็นคำสั่ง Command ในกระบวนการดำเนินงาน และมี Command Queue ไว้เก็บคิวการทำงาน จึงทำให้สามารถประมวลผลพร้อมกันได้หรือตามเวลาที่กำหนด และ Oozie มี SQL DB ในตัวเองสำหรับเก็บข้อมูลการประมวลผลกระแสงาน



ภาพประกอบที่ 2-4 สถาปัตยกรรมของ Apache Oozie (Mohammad Islam และคณะ 2012)

ประเภทของโหนดการทำงานในกระแสงานของ Oozie ประกอบด้วย 2 โหนดหลักด้วยกันคือ โหนดควบคุมการไหล (Control flow nodes) และโหนดของการกระทำ (Action nodes) สำหรับโหนดควบคุมการไหลของกระแสงานจะประกอบด้วยโหนดดังต่อไปนี้

- Start Control Node คือ โหนดที่ควบคุมสำหรับเริ่มต้นงานของกระแสงาน ซึ่งจะต้องกำหนดในไฟล์ของกระแสงานเพียง 1 โหนดเท่านั้น
- End Control Node คือ โหนดที่ควบคุมการสิ้นสุดงานของกระแสงาน และต้องกำหนดในไฟล์ของกระแสงานเพียง 1 โหนดเท่านั้น
- Kill Control Node คือ โหนดที่งานของกระแสงานยินยอมให้ทำลายโหนดด้วยตัวเอง เนื่องจากเกิดการล้มเหลวกระบวนการของการประมวลผล เพื่อเป็นการจบการประมวลผลงานของกระแสงานนั้น จะมีหรือไม่มีโหนดนี้ก็ได้

- Fork and Join Control Nodes สำหรับ Fork Control คือโหนดทางแยกที่จะประมวลผลโหนดของกระทำต่อไป และ Join Control คือโหนดที่รวมกันของการกระทำ การสร้างโหนด Fork และ Join นั้นจำเป็นต้องสร้างคู่กัน เพราะต้องใช้คู่กัน

สำหรับโหนดของการกระทำคือ โหนดที่เรียกใช้สำหรับการคำนวณหรือประมวลผลงาน ซึ่งใน Oozie จะมีโหนดของการกระทำให้เรียกใช้งาน เช่น MapReduce, Pig, FS, Sub-Workflow, SSH เป็นต้น และนักพัฒนาสามารถที่จะสร้างโหนดการกระทำเพิ่มขึ้นได้ เพื่อให้สามารถประมวลผลกระแสงานตามที่ต้องการเรียกว่า Custom ActionExecutor จะกล่าวรายละเอียดการสร้าง Custom ActionExecutor ในหัวข้อที่ 3.3.1

สำหรับ ActionExecutor ของระบบ Oozie ประกอบด้วยโหนด 2 แบบด้วยกันคือ แบบ Synchronous และ Asynchronous โหนดแบบ Synchronous คือการที่โหนดการกระทำของกระแสงานจะรอการทำงาน งานก่อนหน้าต้องเสร็จก่อนที่จะประมวลผลงานถัดไป มีข้อดีคือไม่สิ้นเปลืองทรัพยากรของระบบในการประมวลผล ส่วนโหนดแบบ Asynchronous เป็นโหนดการกระทำของกระแสงานสามารถดำเนินการไปพร้อมๆ กันได้ตามรูปแบบงานของกระแสงานที่ส่งมาประมวลผล ซึ่งจะมีฟังก์ชันในการตรวจสอบสถานะการทำงานของแต่ละงาน มีข้อดีคือสามารถประมวลผลแบบขนานกันได้ สำหรับในงานวิจัยระบบการจำลองจะสร้าง ActionExecutor แบบ Asynchronous โหนดเพื่อทำการประมวลผลกระแสงาน

2.4 WebObject API

เมื่อองค์กรหรือหน่วยงานขยายระบบงานให้มีขนาดใหญ่ขึ้น ข้อมูลของระบบเพิ่มขนาดขึ้น การเก็บข้อมูลขององค์กรในระบบฐานข้อมูลก็จะมีจำนวนเพิ่มขึ้นตามขนาดของหน่วยงานนั้นๆ เพื่อรองรับการเก็บข้อมูลจำนวนมหาศาลที่จะเพิ่มขึ้นในอนาคต NoSQL เป็นอีกหนึ่งสถาปัตยกรรมฐานข้อมูลที่เป็นแบบไม่มีความสัมพันธ์กัน มีความยืดหยุ่นและสามารถเพิ่มจำนวนแถวของข้อมูลได้ถึงพันล้านแถว Apache HBase (Mehul Nalin Vora 2011; Dorin Carstoiu และคณะ 2010) คือระบบฐานข้อมูลแบบกระจายที่รองรับการเก็บข้อมูลขนาดใหญ่ (Big Data) และเก็บข้อมูลแบบ NoSQL จะเก็บข้อมูลโดยใช้ HDFS ของ Hadoop ซึ่ง HBase จะรองรับการเก็บข้อมูลขององค์กรที่ขยายขนาดขึ้นอย่างหลีกเลี่ยงไม่ได้ นอกจากนี้ยังสามารถจัดการข้อมูลได้ง่ายและรวดเร็วขึ้น ในงานวิจัยนี้จึงเลือกใช้ HBase ในการเก็บข้อมูลระบบการจำลองกระแสงาน และได้มีงานวิจัยสร้างโปรแกรมชุดคำสั่งให้ผู้ใช้สามารถเรียกใช้งานเพื่อเก็บข้อมูลบน HBase เรียกว่า WebObject

API (Nitiwat Thongkao และ Somchai Limsiroratana 2014) WebObject API เป็นชุดคำสั่งเซอร์วิสแบบ REST ที่ทำงานอยู่ด้านบนของ HBase โดยการเก็บข้อมูลจะอยู่ในรูปแบบ key-value หรือรูปแบบ JSON

2.5 Directed Acyclic Graph

การอธิบายขั้นตอนการทำงานของกระแสงานบน Oozie นั้นจะอธิบายโดยใช้ Directed Acyclic Graph หรือ DAG เป็นกราฟแบบระบุทิศทางการทำงานที่แน่นอน ซึ่งสามารถเข้าใจการทำงานได้ง่ายเนื่องจากมีทิศทางของโหนดที่แน่นอน DAG สามารถสร้างกราฟได้หลายรูปแบบ เช่น กราฟที่ระบุการทำงานตามลำดับขั้นตอน (Sequentials) จะทำงานเป็นลำดับขั้นตอนในเส้นทางเดียว กราฟที่ระบุเส้นทางการทำงานขนานกัน (Parallels) จะสามารถกระจายการทำงานให้ประมวลผลพร้อมๆ กันได้ และกราฟที่ระบุการทำงานเป็นแบบวัฏจักรหรือวนรอบ (Cycles) แต่สำหรับเครื่องมือที่ใช้ประมวลผลงานอย่าง Oozie จะไม่รองรับกราฟที่ระบุการทำงานเป็นแบบวัฏจักรหรือวนรอบ เพราะจะเกิดการวนไม่รู้จบ ดังนั้นกราฟหรือ DAG ที่ใช้ในงานวิจัยจะเป็นกราฟที่ระบุการทำงานตามลำดับขั้นตอนและกราฟที่ระบุเส้นทางการทำงานขนานกันเท่านั้น

2.6 Workflow XML

การทำงานในระบบการจำลองกระแสงานของ Oozie จะอธิบายด้วยไฟล์ workflow.xml ซึ่งจะสามารถอธิบายถึงขั้นตอนการดำเนินงานของกระแสงานทุกขั้นตอนได้ ใน workflow.xml ไฟล์สามารถระบุแท็ก (Tag) เพื่อบอกประเภทของโหนดควบคุมการไหลและโหนดของการกระทำที่ต้องการดำเนินงาน หรือระบุแท็กเพื่อเป็นพารามิเตอร์สำหรับการประมวลผลในโหนดการกระทำได้ สำหรับแท็กที่ระบุใน workflow.xml นี้ จะต้องสอดคล้องกับประเภทของโหนดการทำงานของ Oozie ซึ่งได้กล่าวถึงในข้อที่ 2.3.1 เรื่อง Apache Oozie จะอธิบายแท็กเพิ่มเติมคือส่วนที่ใช้เป็นพารามิเตอร์ในการประมวลผลของโหนดการกระทำ ดังภาพประกอบที่ 2-5 จะประกอบด้วย actionname คือ ชื่อโหนดของการกระทำ, role คือ หน้าที่ของงาน, actiontime คือเวลาที่ใช้ประมวลผลงาน, transfertime, คือเวลาที่ใช้ระหว่างเส้นทาง, objectid_account คือบัญชีผู้ใช้ในทำงาน และ project คือโปรเจคที่เลือกจะทำงาน ซึ่ง objectid_account และ project เป็นค่าตัวแปรสำหรับผู้ใช้ที่ทำการประมวลผลกระแสงาน

```
<action name="....">
      <async xmlns="uri:oozie:async-action:0.1">
           <actionname>....</actionname>
           <role>....</role>
           <actiontime>...</actiontime>
           <transfertime>...</transfertime>
           <objectid_account>
                ${objectid_account}
           </objectid_account>
           <project>${project}</project>
       </async>
       <ok to="..."/>
       <error to="fail"/>
</action>
```

ภาพประกอบที่ 2-5 พารามิเตอร์ที่ใช้ใน 1 โหนดการกระทำของกระแสงาน

บทที่ 3
วิธีดำเนินการวิจัย

ในบทนี้ จะกล่าวถึงแนวคิดการจำลองการทำงาน สถาปัตยกรรมของระบบการจำลองกระแสงานอัตโนมัติ เครื่องมือสำหรับดำเนินการและประมวลผลกระแสงานตามเวลา การเตรียมข้อมูลที่จำเป็นในระบบการจำลองกระแสงาน การออกแบบและการดำเนินการสร้างระบบการจำลองสำหรับใช้ในกระประมวลผลกระแสงาน และสุดท้ายวิธีการทดสอบระบบการจำลองและการประมวลผลกระแสงาน

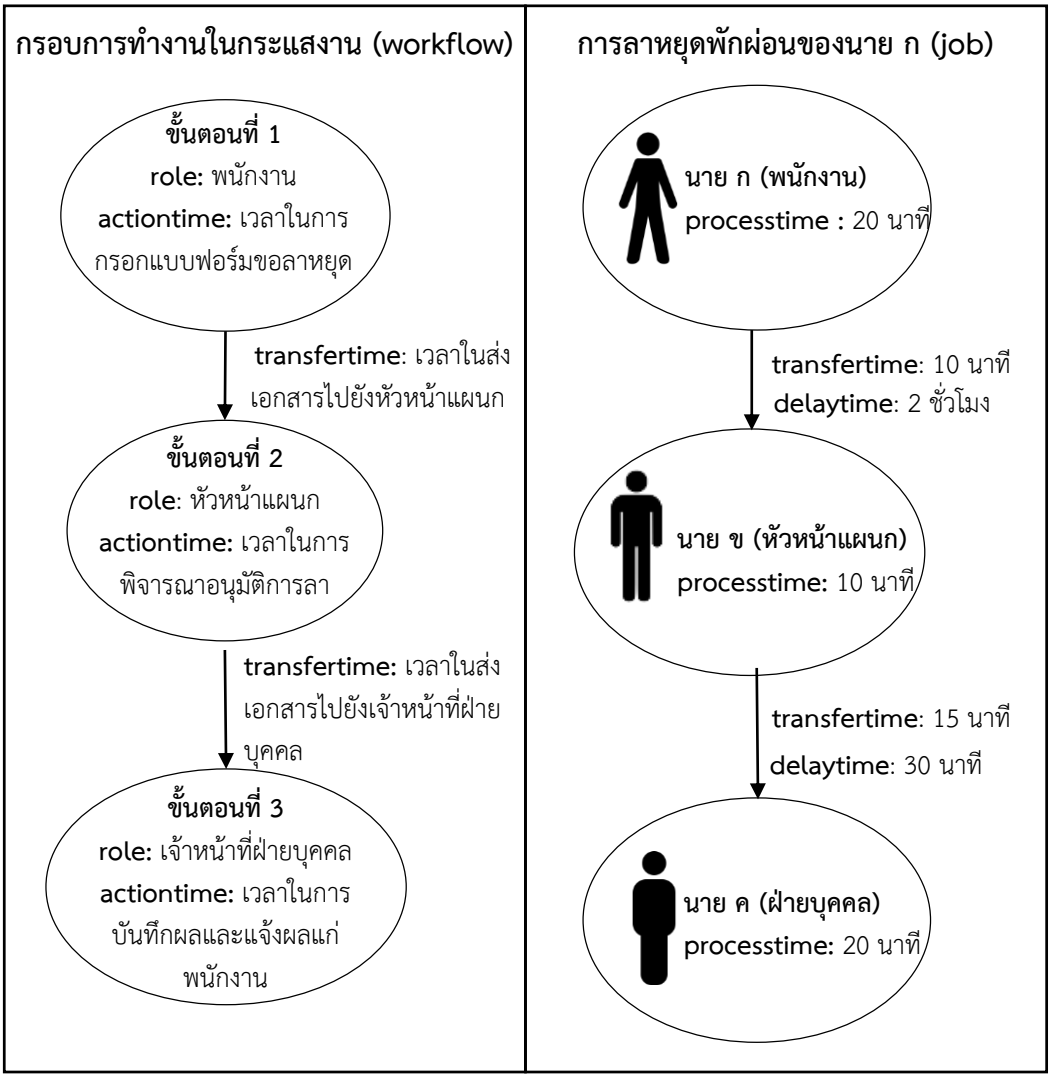
3.1 แนวคิดการจำลองการทำงาน

การดำเนินงานขององค์กรใดๆ จะประกอบด้วยขั้นตอนการทำงานที่แตกต่างกันหลายขั้นตอน ซึ่งในแต่ละขั้นตอนก็จะมีบุคคลที่เกี่ยวข้องกับบทบาทของงานและสามารถทำงานนั้นได้ ตัวอย่าง เช่น การขอลาหยุดพักผ่อนของพนักงาน ดังภาพประกอบที่ 3-1 มีขั้นตอนดังนี้

- ขั้นตอนที่ 1 พนักงานที่ต้องการลาพักผ่อนกรอกแบบฟอร์มขอลาหยุด โดยระบุจำนวนวันที่ต้องการลาหยุดจากวันเริ่มต้นถึงวันสุดท้ายของการลาหยุด แล้วยื่นเอกสารการขอลาหยุดพักผ่อนกับหัวหน้าแผนกหรือผู้มีอำนาจอนุมัติลาหยุดในขั้นตอนที่ 2
- ขั้นตอนที่ 2 หัวหน้าแผนกตรวจสอบสิทธิ์คงเหลือ และตรวจสอบกำลังคนในตำแหน่งงาน ความเร่งด่วนของงาน ในช่วงวันเวลาที่ขอว่าเพียงพอหรือไม่ เมื่อพิจารณาเรียบร้อยก็ลงชื่ออนุมัติหรือไม่อนุมัติ หากอนุมัติก็จะส่งเรื่องการขอลาหยุดไปให้ฝ่ายบุคคล เพื่อบันทึกข้อมูลในระบบในขั้นตอนที่ 3
- ขั้นตอนที่ 3 ฝ่ายบุคคลลงบันทึกรายละเอียดของวันลาหยุดให้กับพนักงาน ในระบบงานของหน่วยงาน และแจ้งผลการขอลาหยุดพักผ่อนให้กับพนักงาน

จากตัวอย่างขั้นตอนการขอลาหยุดของพนักงาน องค์กรจะสร้างกรอบการทำงานหรือแนวปฏิบัติขึ้นมาเรียกว่ากระแสงาน (workflow) โดยกำหนดให้มี 3 ขั้นตอน (action) ในแต่ละขั้นตอนกำหนดบทบาทหรือหน้าที่ของงานนั้นๆ (role) เช่น ในกรณีนี้มีบทบาทคือ พนักงาน หัวหน้าแผนก และเจ้าหน้าที่ฝ่ายบุคคล นอกจากนี้ยังต้องกำหนดกรอบเวลาที่ใช้ในการทำงานของแต่ละ

ขั้นตอน (actiontime) กรอบเวลาที่ใช้ในการส่งผ่านแต่ละขั้นตอน (transfertime) เพื่อเป็นแบบแผนในการทำงานและควบคุมคุณภาพการทำงาน



ภาพประกอบที่ 3-1 ตัวอย่างขั้นตอนการลาหยุดพักผ่อน
กรอบการทำงานในกระแสงาน (ซ้าย) และการลาหยุดของนาย ก (ขวา)

เมื่อพนักงาน (user) เช่น นาย ก เริ่มกรอกแบบฟอร์มการขอลาหยุด ก็คือการทำขั้นตอนที่ 1 จึงเป็นการเริ่มทำงาน (job) ตามกระแสงานที่วางไว้ ซึ่งระยะเวลาในการส่งแบบฟอร์มไปยังหัวหน้าแผนก (ระหว่างขั้นตอนที่ 1 ไปยังขั้นตอนที่ 2) หรือจากหัวหน้าแผนกไปยังเจ้าหน้าที่ฝ่ายบุคคลคือ (ระหว่างขั้นตอนที่ 2 ไปยังขั้นตอนที่ 3) นั้นหากการส่งเป็นรูปแบบเอกสารกระดาษจะมีเวลาคลาดเคลื่อนมากกว่าการส่งเป็นเอกสารอิเล็กทรอนิกส์ในปัจจุบัน และเพื่อไม่ให้ส่งผลต่อการ

วิเคราะห์การทำงานของคน จึงเลือกจำลองเวลาการส่งผ่านแต่ละขั้นตอนตามค่า transfertime ที่กำหนดไว้ในกระแสงาน เสมือนเป็นการส่งเอกสารอิเล็กทรอนิกส์

การกรอกแบบฟอร์มขอลาหยุดของนาย ก การพิจารณาเพื่ออนุมัติวันลาหยุดให้นาย ก ของหัวหน้าแผนก และการบันทึกวันลาหยุดให้นาย ก ของเจ้าหน้าที่ฝ่ายบุคคล คือเวลาจริงที่ใช้ในการทำงานของแต่ละขั้นตอน (processtime) ซึ่งอาจจะไม่เป็นไปตามกรอบเวลาที่ได้กำหนดไว้ในกระแสงาน (actiontime) ในการทำงานจริงจะมีปัจจัยหลายอย่างที่เกี่ยวข้องกับเวลาทำงาน เช่น ความสามารถของคน ประสบการณ์การทำงาน ความเหนื่อยล้า และอารมณ์ของผู้ทำงาน ฯลฯ ในการจำลองเพื่อวิเคราะห์โครงสร้างของกระแสงานที่เหมาะสม จึงตัดปัจจัยเหล่านี้ออกไปและกำหนดให้เวลาการทำงานเท่ากับที่กำหนดไว้ในกระแสงาน เสมือนคนทำงานมีประสิทธิภาพคงที

ในบางเหตการณ์คนไม่พร้อมที่จะทำงานนั้นในทันทีเนื่องจากติดภาระกิจอื่น จึงเกิดการรอในระหว่างขั้นตอนการทำงาน (delaytime) ในกรณีตัวอย่างนี้ ถ้าหัวหน้าแผนกมีการประชุมหรือเดินทางไปต่างจังหวัด ขั้นตอนในการอนุมัติวันลาหยุดของนาย ก ก็จะล่าช้าขึ้น และเนื่องจากแต่ละขั้นตอนในกระแสงานมีเวลาที่ทำงานไม่เท่ากัน เพราะเป็นงานคนละประเภท ไม่สามารถเปรียบเทียบเวลาของการทำงานโดยตรงได้ จึงได้คำนวณเทียบเป็นเปอร์เซ็นต์ของเวลาที่ทำงาน กับเวลาที่ใช้รอเรียกว่า เปอร์เซ็นต์ความล่าช้าของงาน (%delay) ดังสมการ (1)

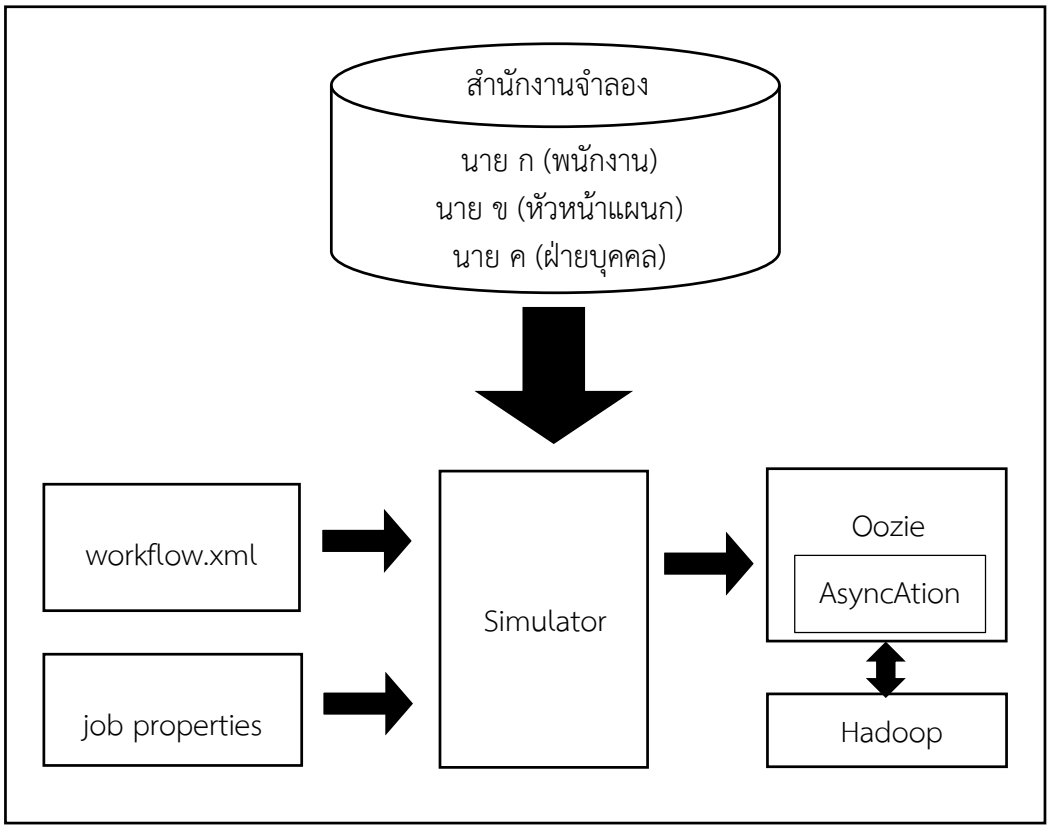
$$\%\text{delay} = \frac{\textbf{delaytime x 100}}{\textbf{processtime}} \qquad (1)$$

การจำลองการทำงานในแต่ละขั้นตอน ในงานวิจัยนี้ได้สร้างโหนดการกระทำแบบพิเศษเพื่อจำลองการทำงานในสถานการณ์ทำงานเสมือนจริง โดยยึดตามกรอบการทำงานในกระแสงานเรียกว่า AsyncAction (ดูรายละเอียดในหัวข้อ 3.3.1) สำหรับใช้ประมวลผลงานในแต่ละขั้นตอนดังภาพประกอบที่ 3-2 ก่อนการจำลองจะต้องเตรียมข้อมูลโครงสร้างขององค์กรโดยจัดเก็บในฐานข้อมูล ข้อมูลที่เก็บจะประกอบด้วยบุคลากรขององค์กรและคุณสมบัติต่างๆ ของบุคลากร ประเภทของบทบาทการทำงานหรือตำแหน่งต่างๆ ในองค์กร เช่น นาย ก มีบทบาทเป็นพนักงาน นาย ข มีบทบาทเป็นหัวหน้าแผนก และนาย ค มีบทบาทเป็นเจ้าหน้าที่ฝ่ายบุคคล เป็นต้น

นอกจากนี้ในการอธิบายกรอบการทำงานของกระแสงาน จะต้องเตรียมให้อยู่ในรูปแบบ XML ไฟล์ (workflow.xml) เพื่ออธิบายลำดับของขั้นตอนการทำงานในรูปแบบที่ Oozie เข้าใจและนำไปประมวลผลได้ ซึ่งในการจำลองเหตการณ์หนึ่งๆ ดังเช่นการลาหยุดของนาย ก นี้จะต้องมีการระบุข้อมูลคุณสมบัติของงาน (job properties) ในการประมวลผลด้วย เพื่อให้

AsyncAction ทราบถึงข้อมูลในองค์กรที่กำลังประมวลผล ว่ามีบทบาทอะไรบ้างหรือมีใครที่เกี่ยวข้องกับงานที่กำลังประมวลผลอยู่บ้าง

จากกลไกการจำลองที่กล่าวมานี้ จะต้องมีตัวกลางเพื่อใช้จัดการเตรียมข้อมูลดังกล่าว และจัดการส่งงานไปจัดลำดับการประมวลตามขั้นตอนบน Oozie นั้นก็คือ Simulator จากนั้น Oozie ก็จะจำลองงานด้วย AsyncAction ด้วยการส่งงานไปประมวลผลบน Hadoop ต่อไป



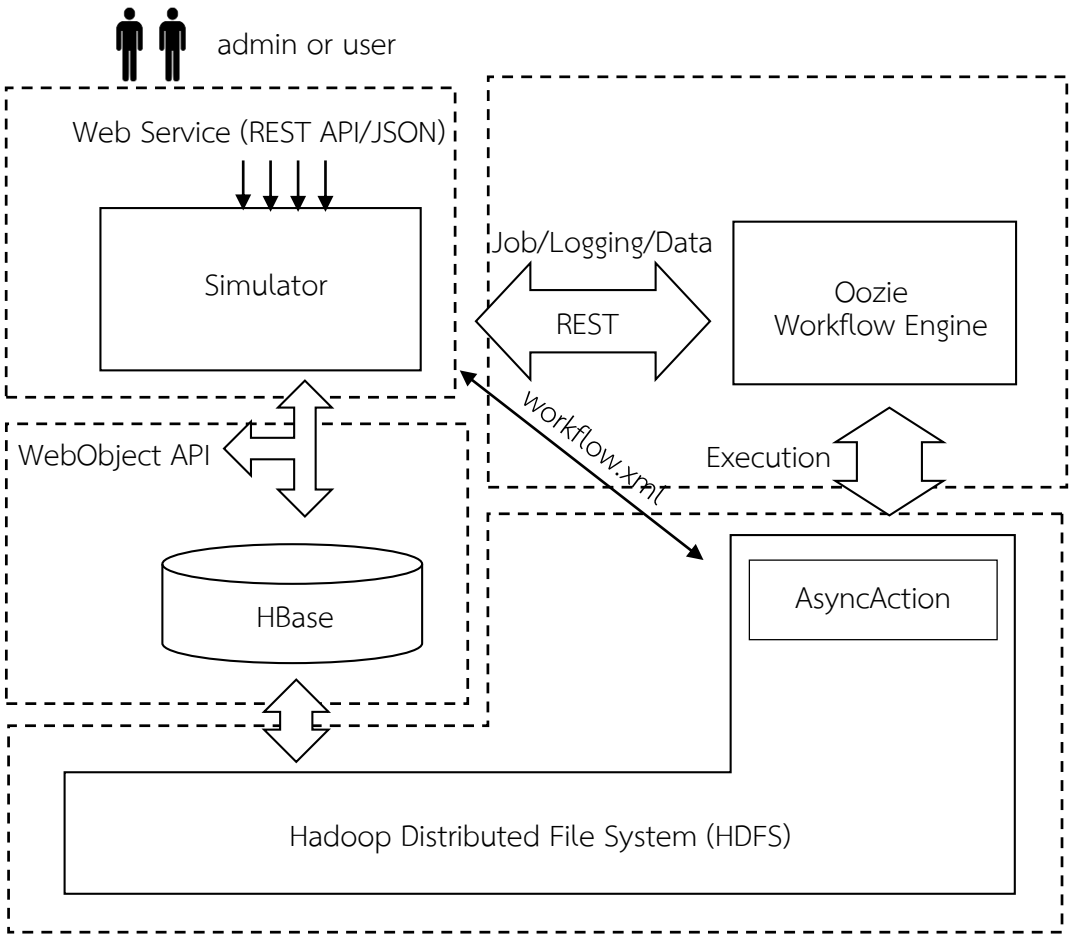
ภาพประกอบที่ 3-2 แนวคิดของระบบการจำลอง

3.2 สถาปัตยกรรมของระบบการจำลอง

เพื่อให้การจำลองเป็นการทำงานเสมือนสำนักงานขององค์กรจริง จึงได้ออกแบบสถาปัตยกรรมของระบบประกอบด้วย 4 ส่วนด้วยกัน สำหรับดำเนินการประมวลผลกระแสงาน คือ Simulator, Oozie workflow engine, HBase และ HDFS ดังภาพประกอบที่ 3-3

สถาปัตยกรรมของระบบการจำลองส่วนของ Simulator จะเป็นส่วนที่ติดต่อกันระหว่างผู้ดูแลระบบหรือผู้ใช้งาน ติดต่อกับระบบการจำลอง เพื่ออัพโหลดไฟล์กระแสงานไปเก็บไว้บน

HDFS เก็บข้อมูลของบุคลากรในสำนักงานขององค์กร และมีฟังก์ชันเพื่อจะสั่งประมวลผลกระแสงาน ผ่านทางเว็บส่วนติดต่อกับผู้ใช้ โดยที่ผู้ใช้งานมีการติดต่อกับ Simulator ผ่านทางเว็บเซอร์วิซแบบ REST มีการแลกเปลี่ยนข้อมูลแบบ JSON ส่วนของ Oozie worfklow engine เป็นเครื่องมือสำหรับจัดการ การดำเนินงาน และประมวลผลการกระทำของกระแสงาน โดยการติดต่อของ Oozie กับ Simulator ผ่านทางเซอร์วิซแบบ REST ของ Oozie เพื่อเตรียมข้อมูลของที่จะประมวลผลกระแสงาน และบันทึกข้อมูลในการประมวลผล นอกจากนั้น Oozie จะติดต่อกับ HDFS เพื่อประมวลผลกระแสงาน โดยไฟล์กระแสงาน (workflow.xml) ที่จะใช้ในการประมวลผล จะต้องถูกอัพโหลดไปเก็บไว้ใน HDFS โดยผู้ใช้งานก่อนจะมีการประมวลผลกระแสงานนั้น ข้อมูลของระบบการจำลองทั้งหมดจะถูกเก็บอยู่ในฐานข้อมูลคือ HBase การติดต่อกับฐานข้อมูลของ Simulator โดยอาศัยชุดคำสั่ง WebObject API และส่วนสุดท้ายคือ HDFS เป็น MapReduce สำหรับการประมวลผลงานแบบกระจายของ Oozie และได้สร้างฟังก์ชันการทำงานของ Oozie เพิ่มขึ้น คือ AsyncAction เพื่อที่จะใช้ประมวลผลตามเวลาของกระแสงาน



ภาพประกอบที่ 3-3 สถาปัตยกรรมของระบบการจำลอง

3.3 เครื่องมือสำหรับประมวลผลกระแสงาน

เครื่องมือในการประมวลผลกระแสงานใช้ Oozie เป็นเครื่องมือสำหรับจัดการกระแสงานตามเวลางาน และสามารถให้นักพัฒนาสร้างโหนดการกระทำใน Oozie เพิ่มขึ้นได้ เพื่อใช้ประมวลผลกระแสงานในแบบที่ต้องการ นอกเหนือจากที่ระบบของ Oozie มีให้เรียกใช้ โหนดการกระทำที่สามารถสร้างเพิ่มได้เรียกว่า Custom ActionExecutor ซึ่งจะกล่าวรายละเอียดในข้อ 3.3.1

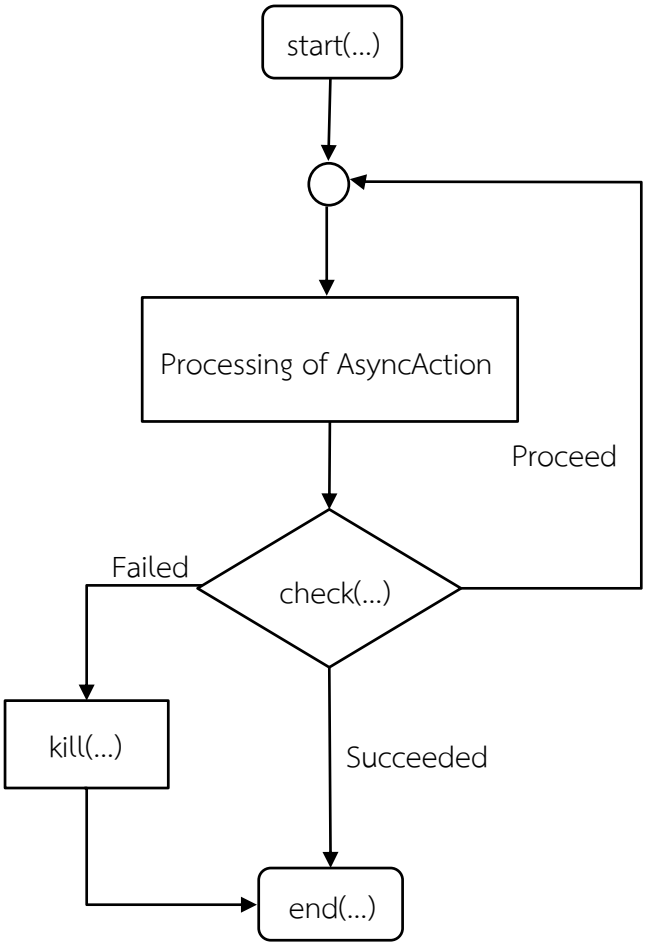
3.3.1 Custom ActionExecutor

สำหรับการจำลองการทำงานในองค์กร จะมีการกำหนดบทบาทและหน้าที่ของบุคลากรไว้ในกระแสงาน ซึ่งระบบที่ออกแบบจะจำลองบุคลากรและระบุการกระทำไว้ในกระแสงาน โดยจะต้องส่งผ่านค่าพารามิเตอร์ต่างๆ ที่จำเป็นเช่น บทบาท เวลาที่ใช้ในการทำงาน เป็นต้น (รายละเอียดเพิ่มในข้อ 3.4.1 เรื่องของการสร้างไฟล์กระแสงาน) สำหรับการสร้างจะใช้กลไก Custom ActionExecutor ของ Oozie เพื่อดำเนินงานและประมวลผลกระแสงาน ในงานวิจัยนี้ได้ออกแบบคลาสสำหรับการจำลองการกระทำของกระแสงานเรียกว่า AsyncAction ซึ่งถูกออกแบบให้เป็นโหนดการกระทำแบบ Asynchronous เนื่องจากกระแสงานที่ได้สร้างขึ้นเป็นกระบวนการของการกระทำที่เกิดขึ้นพร้อมกันได้ AsyncAction สืบทอดมาจากคลาส ActionExecutor ของ Oozie แสดงขั้นตอนการดำเนินงานดังภาพประกอบที่ 3-4 และมีฟังก์ชันการทำงานและสามารถเพิ่มการดำเนินการไปได้ ดังต่อไปนี้

- start(...) เป็นฟังก์ชันแรกที่ AsyncAction เพื่อเลือกบุคคลในบทบาทงานที่กำหนดมาทำงานและคนต้องพร้อมจะทำงาน ถ้าบุคคลไม่ว่างในการทำงานก็รอจนกว่าคนนั้นจะว่างพร้อมทำงาน เมื่อเลือกบุคคลสำหรับมาทำงานแล้ว จะสั่งคำสั่ง context.setStartData(externalId, trackerUri, consoleUrl) เพื่อกำหนดเป็น Asynchronous และ context.setExecutionData(external Status, actionData) เพื่อประมวลผลและอัพเดทสถานะงาน
- check(...) ขณะประมวลผลขั้นตอนของกระแสงานอยู่นั้น ระบบก็จะตรวจสอบสถานะการทำงานว่าเสร็จแล้วหรือไม่ ถ้างานยังไม่เสร็จก็รอการตรวจรอบถัดไป กรณีถ้างานเสร็จแล้วก็จะสั่งคำสั่ง context.setExecutionData (external Status, actionData) เพื่อส่งไปทำงานฟังก์ชัน end() ต่อไป

- end(...) เมื่อฟังก์ชัน check() ตรวจสอบการทำงานเสร็จ ฟังก์ชัน end() ก็จะประมวลผลงานต่อโดยสั่งคำสั่ง context.setEndData(status, signalValue) เพื่อสิ้นสุดการประมวลผลในขั้นตอนของกระแสงาน และเก็บข้อมูลการประมวลผลของกระแสงานนั้น
- kill(...) ถ้าฟังก์ชัน check() ตรวจสอบสถานะแล้วพบว่ามีการเปลี่ยนแปลงแต่เป็นสถานะงานไม่สำเร็จ ก็จะจบการทำงานด้วยฟังก์ชัน kill() โดยสั่งคำสั่ง context.setEndData(status,signalValue) เพื่อจบการประมวลผลในขั้นตอนนั้นและจบการทำงานกระแสงาน

เนื่องจากการทำงานแบบ Asynchronous เป็นการทำงานแบบขนานกัน จึงได้มีการสร้าง Thread ขึ้นมาเพื่อใช้เตรียมข้อมูล สำหรับรอบุคคลที่จะมาทำงาน และ Thread ที่ถูกสร้างมาจะเก็บอยู่ในชุดเก็บข้อมูล เพื่อที่จะจัดการเริ่มและจบการทำงานของ Thread ได้สะดวกและง่ายขึ้น



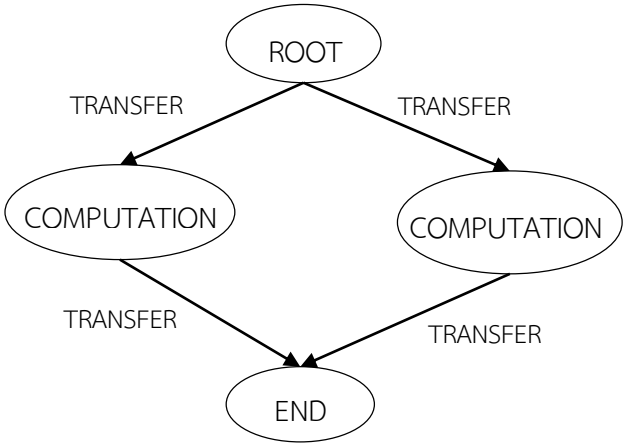
ภาพประกอบที่ 3-4 ขั้นตอนการดำเนินงานของคลาส ActionExecutor

3.4 การเตรียมข้อมูลในการจำลอง

สำหรับการประมวลผลกระแสงานจะประกอบด้วย ข้อมูลของกระแสงานคือ workflow.xml และการตั้งค่าในไฟล์ job.properties สำหรับการตั้งค่าจะประกอบด้วยข้อมูลของบุคคลสำหรับทำงาน (user) บทบาทตำแหน่งของงาน (role) ข้อมูลจัดลำดับคิวงานของกระแสงานที่ต้องการประมวลผล (queue) ซึ่งจะต้องเตรียมข้อมูลเหล่านี้ก่อนการส่งกระแสงานไปดำเนินการประมวลผล และสุดท้ายเมื่อกระบวนการดำเนินงานเสร็จแล้ว ก็จะเก็บข้อมูลในการประมวลผลกระแสงานงานนั้นๆ ไว้ (log)

3.4.1 การสร้างไฟล์กระแสงาน

ในการจำลองเพื่อส่งกระแสงานไปประมวลผลงานนั้น รูปแบบไฟล์ของกระแสงานจะเป็น XML ไฟล์และกระแสงานอยู่ในมาตรฐาน WfMC ซึ่งไฟล์กระแสงานสร้างจากโปรแกรม DAGGEN (DAGGEN 2017) DAGGEN เป็นโปรแกรมช่วยในการสังเคราะห์กราฟของงาน กราฟที่สร้างขึ้นเป็นกราฟแบบ DAG เหมาะสำหรับใช้ในการประมวลผลกระแสงาน เพื่อประเมินอัลกอริทึมการไหลของกระแสงานและเวลาที่ใช้ในการทำงาน จะมีโหนดการทำงาน 4 แบบด้วยกันคือ ROOT, COMPUTATION, TRANSFER และ END สำหรับโหนด ROOT และ END เป็นโหนดเริ่มต้นและสิ้นสุดการทำงานของกราฟตามลำดับ โหนด COMPUTATION คือโหนดที่ระบุเวลาใช้ในการประมวลผลของงาน และโหนด TRANSFER เป็นโหนดที่ระบุเวลาในเส้นทางเดินของกราฟ ดังภาพประกอบที่ 3-5



ภาพประกอบที่ 3-5 รูปแบบโหนดกราฟของ DAGGEN

ในการสร้างไฟล์กระแสงานนั้นเพื่อนำมาใช้ในการจำลอง จะกำหนดโหนดที่ระบุเวลาสำหรับการประมวลผลจำนวน 10 โหนด และกำหนดค่าคุณสมบัติอย่างอื่นเป็นค่าเริ่มต้นของโปรแกรม ไฟล์กราฟที่สร้างเสร็จแล้วดังแสดงในภาพประกอบที่ 3-7 โดยรูปแบบแต่ละบรรทัดของไฟล์กราฟจะถูกอธิบายโหนดการทำงานดังภาพประกอบที่ 3-6 ต่อไปนี้

```
NODE <ลำดับที่> <โหนดลูก> <ประเภทของโหนด> <ค่าเวลา> <ค่าparallelization overhead>
```

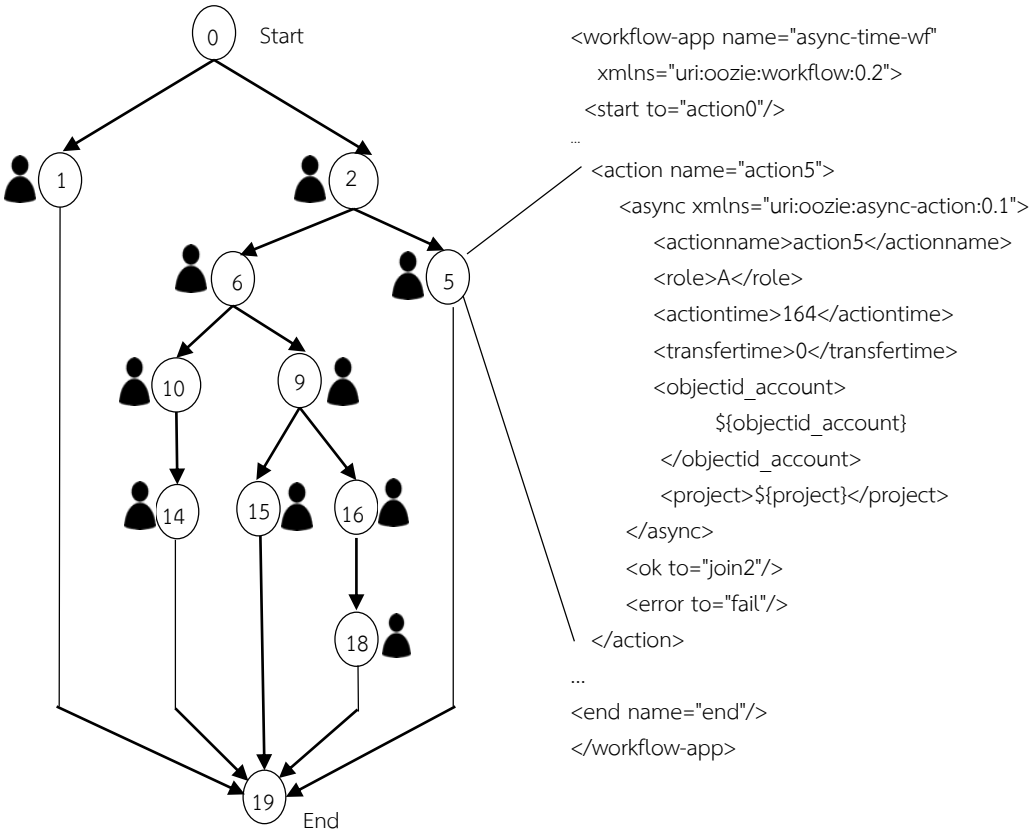
ภาพประกอบที่ 3-6 รูปแบบไฟล์ผลลัพธ์ที่ได้จากการสังเคราะห์ของ DAGGEN

```
NODE 0 1,2 ROOT 0.0 0.0
NODE 1 19 COMPUTATION 1798485812 0.16
NODE 2 3,4 COMPUTATION 549755813888 0.09
NODE 3 5 TRANSFER 536870912 0.0
NODE 4 6 TRANSFER 536870912 0.0
NODE 5 19 COMPUTATION 1640177522 0.17
NODE 6 7,8 COMPUTATION 2297570555 0.19
NODE 7 9 TRANSFER 209715200 0.0
NODE 8 10 TRANSFER 209715200 0.0
NODE 9 11,12 COMPUTATION 68719476736 0.04
NODE 11 15 TRANSFER 134217728 0.0
NODE 12 16 TRANSFER 134217728 0.0
NODE 10 13 COMPUTATION 1073741824000 0.01
NODE 13 14 TRANSFER 838860800 0.0
NODE 14 19 COMPUTATION 887851624 0.14
NODE 15 19 COMPUTATION 134217728000 0.15
NODE 16 17 COMPUTATION 170520209604 0.03
NODE 17 18 TRANSFER 134217728 0.0
NODE 18 19 COMPUTATION 28991029248 0.14
NODE 19 - END 0.0 0.0
```

ภาพประกอบที่ 3-7 ตัวอย่างไฟล์กราฟที่ได้จากการสังเคราะห์ของโปรแกรม DAGGEN

การแปลงไฟล์กราฟจาก DAGGEN เป็นไฟล์ workflow.xml จะต้องอยู่ในรูปแบบโหนดควบคุมการไหลและโหนดการกระทำของ Oozie ต้องมีข้อมูลในองค์ประกอบ (element) เพิ่มเข้าไปในกระแสงานคือ actionname เป็นชื่อของโหนดที่จะประมวลผล role เป็นบทบาทหน้าที่ในการทำงาน actiontime เวลาที่ใช้ในการประมวลผลซึ่งก็คือโหนดที่ระบุเวลาในการประมวลผล transfertime คือเวลาของค่าในเส้นทางเดินของโหนด objectid_account คือรหัสบัญชีผู้ใช้งานหรือผู้ดูแลระบบ และ project เป็นโปรเจคที่ต้องการใช้ในการประมวลผลกระแสงาน

ตัวอย่างกราฟของกระแสงาน การสมมติบุคคลเพิ่มเข้าไปสำหรับการทำงาน และคำสั่งโหนดการกระทำบางส่วนขององค์ประกอบ workflow.xml แสดงในภาพประกอบที่ 3-8 ยกตัวอย่างโหนดการกระทำที่ 5 actionname คือ action5, role คือ A, actiontime คือ 164, transfertime คือ 0, objectid_account และค่า project เป็นตัวแปรของผู้ใช้งานหรือผู้ดูแลระบบคนนั้นๆ



ภาพประกอบที่ 3-8 ตัวอย่างรูปแบบของกระแสงาน (ซ้าย) และ คำสั่งแต่ละบรรทัดของ workflow.xml (ขวา)

3.4.2 หน่วยของเวลาในการจำลอง

สำหรับหน่วยของเวลาที่ใช้ในระบบการจำลองกระแสงาน และเวลาของเครื่องคอมพิวเตอร์ในโลกความเป็นจริง จะสมมติให้หน่วยเวลาของระบบการจำลองกระแสงานภาพประกอบที่ 3-9 ซึ่งค่าเวลาที่ได้มาจากไฟล์กราฟของ DAGGEN เป็นเวลาการประมวลผลของ CPU ในหน่วยนาโนวินาทีทำให้ตัวเลขที่ได้มานั้นมีจำนวนมาก ซึ่งไม่สอดคล้องกับการจำลองจึงลดตัวเลขลงจำนวน 10 ไมโครวินาที ก่อนที่จะแปลงไฟล์กราฟเป็น workflow.xml เพื่อใช้ในการประมวลผลกระแสงานต่อไป

1 วินาทีของระบบการจำลอง = 10 มิลลิวินาทีของเครื่องคอมพิวเตอร์

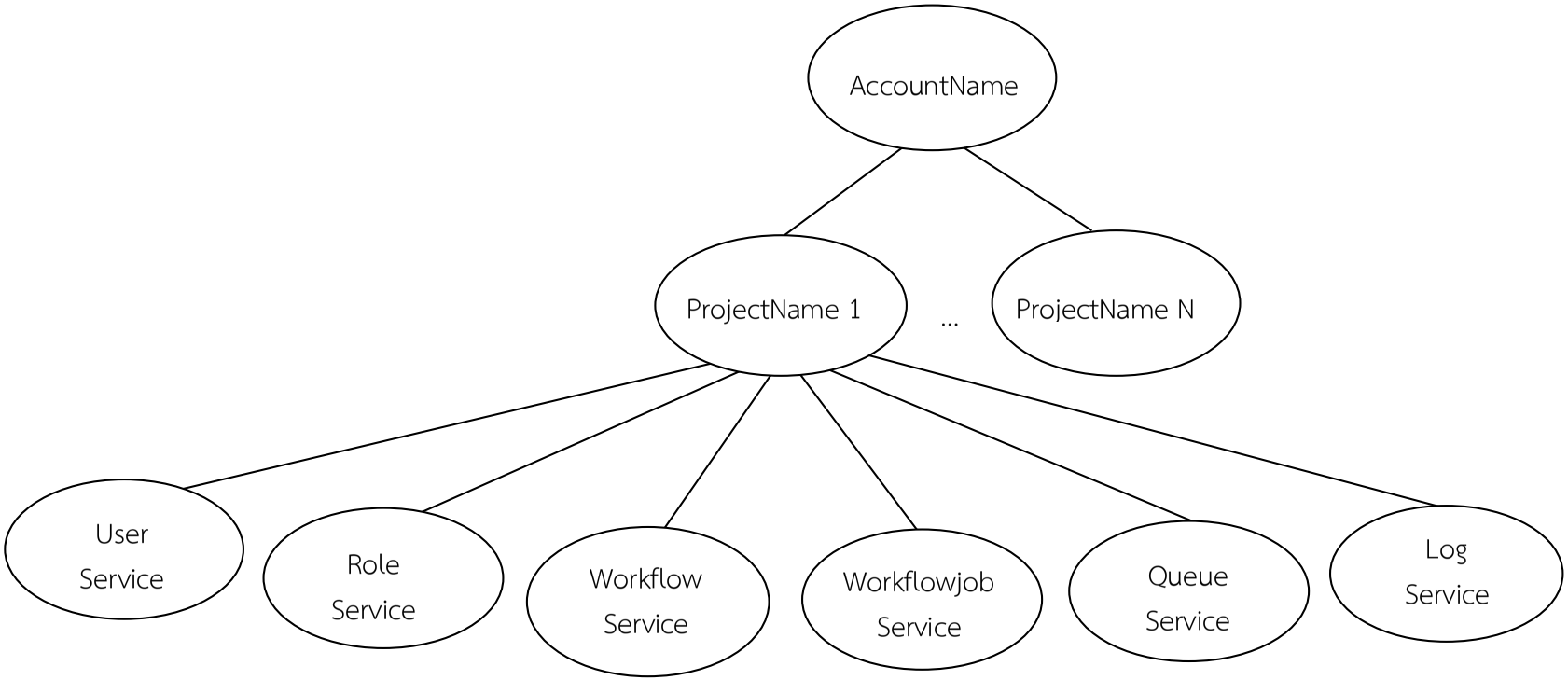
ภาพประกอบที่ 3-9 หน่วยเวลาในระบบการจำลอง

3.4.3 การออกแบบการเก็บข้อมูล

เพื่อให้ระบบรองรับการเก็บข้อมูลที่มีความยืดหยุ่นได้ในอนาคต ซึ่งใช้ฐานข้อมูล HBase เก็บข้อมูลที่ต้องเตรียมในการประมวลผลกระแสงาน เรียกใช้ผ่านทางชุดคำสั่ง WebObject API โดยสร้างเป็นเซอร์วิสแบบ REST เพื่อให้สามารถเรียกใช้บริการได้ง่ายขึ้น เริ่มต้นการเก็บข้อมูลจะต้องสร้างบัญชีผู้ใช้ (AccountName) สำหรับการใช้งานในระบบการจำลอง และสร้างโปรเจคของการทำงาน (ProjectName) ระบบการจำลองจะสร้างเซอร์วิสต่างๆ ขึ้นมาให้อัตโนมัติ ประกอบด้วยเซอร์วิสดังต่อไปนี้ User Service, Role Service, Workflow Service, WorkflowJob Service, Queue Service และ Log Service แสดงในภาพประกอบที่ 3-10

- AccountName เนื่องจากชุดคำสั่ง WebObject API มีฟังก์ชันที่สามารถลงทะเบียนการใช้งานของผู้ใช้แต่ละบุคคลได้ ระบบการจำลองจะได้สร้างบัญชีผู้ใช้ขึ้นสำหรับระบบการจำลองกระแสงาน
- ProjectName ในแต่ละบัญชีผู้ใช้งานจะสามารถสร้างหรือจัดการโปรเจคในการจำลองได้ ซึ่งในแต่ละโปรเจคมีการทำงานที่แยกจากกัน

- User Service เป็นเซอร์วิสที่ให้เรียกใช้เพื่อการเก็บข้อมูลบุคลากรในองค์กร เช่น บทบาทการทำงาน เวลาที่ใช้ทำงาน เงินเดือน และสถานะของการทำงาน
- Role Service เป็นเซอร์วิสที่ระบุค่าตำแหน่งการทำงานของบุคลากรในองค์กร
- Workflow Service เป็นเซอร์วิสที่ใช้สำหรับอัพโหลดกระแสงานไปเก็บไว้ใน HDFS
- WorkflowJob Service เป็นการเตรียมข้อมูลเกี่ยวกับบัญชีผู้ใช้และโปรเจคที่จะประมวลผลของกระแสงาน และเก็บข้อมูลเส้นทางที่อยู่ของกระแสงานนั้นบน HDFS เพื่อรอเรียกใช้สำหรับการประมวลผลกระแสงานนั้น
- Queue Service ผู้ใช้งานสร้างคิวของกระแสงานก็จะมาเรียกใช้เซอร์วิสคิว เพื่อเก็บค่า job.properties และเวลาเริ่มการในการประมวลผลงานของกระแสงานนั้น เมื่อผู้ใช้งานเลือกคิวงานมาประมวลผล การดำเนินการจะเริ่มตามเวลาที่กำหนดไว้ในคิวงานนั้น
- Log Service เมื่อประมวลผลกระแสงานข้อมูลในการประมวลผลก็จะถูกเก็บอยู่ในเซอร์วิสนี้ เพื่อนำข้อมูลไปวิเคราะห์การดำเนินงานต่อไป



ภาพประกอบที่ 3-10 โครงสร้างการเก็บข้อมูลลง Apache HBase

ในการเรียกใช้งานเพื่อที่จะเตรียมข้อมูลนั้น ผู้วิจัยได้สร้างเว็บที่ติดต่อกับผู้ใช้งาน (Web User Interface) เพื่อให้ผู้ใช้งานหรือผู้ดูแลระบบสะดวกในการเรียกใช้เซอร์วิส ซึ่งฟังก์ชันต่างๆ ของทุกเซอร์วิส (AllService) ประกอบด้วย CREATE, SET, GET, DELETE และ LIST และเซอร์วิสกระแสงาน (WorkflowService) จะมีฟังก์ชัน UPLOAD FILE เพิ่ม เพื่ออัพโหลดไฟล์ workflow.xml ไปเก็บไว้ที่ HDFS แสดงเซอร์วิสแบบ REST ดังภาพประกอบที่ 3-11

AllService
- CREATE - SET - GET - DELETE - LIST

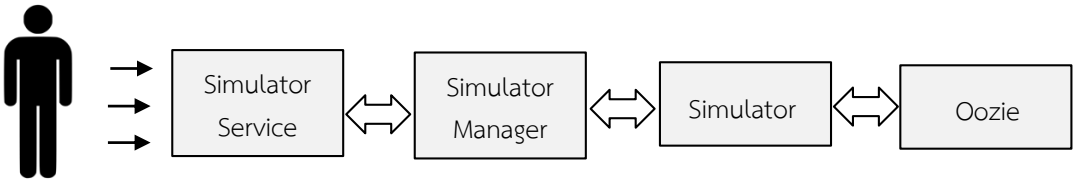
WorkflowService
- CREATE - SET - GET - DELETE - LIST - UPLOAD FILE

ภาพประกอบที่ 3-11 ชุดคำสั่งเซอร์วิสแบบ REST สำหรับเตรียมข้อมูลการจำลองกระแสงาน

3.5 การออกแบบและการดำเนินการสร้าง Simulator

Simulator ถูกออกแบบเป็นเว็บเซอร์วิสที่ดำเนินการอยู่บน Apache Tomcat 7 และติดต่อกับ Oozie ผ่าน REST API เมื่อมีการสั่งทำงานของผู้ใช้เพื่อส่งกระแสงานไปประมวลผล การเก็บข้อมูล และเก็บผลลัพธ์ของการประมวลผล เซอร์วิสของ Simulator เป็น Servlet แบบ Application scope ซึ่งจะถูกสร้างขึ้นในครั้งแรกที่ผู้ใช้งานเรียกใช้งาน HTTP request และทำงานอยู่เบื้องหลังตลอดเวลา จนกระทั่งการจำลองนั้นจะถูกทำลายหรือจบการทำงานไป

เมื่อผู้ใช้งานจำลองกระแสงานในโปรเจคเดิมที่เคยถูกสร้างไว้แล้ว เพื่อไม่ให้เกิดการซ้ำซ้อนกันของการใช้ทรัพยากรจึงออกแบบ Simulator เป็น Singleton Pattern นอกจากนี้ยังออกแบบเป็นเซอร์วิสแบบ REST เรียกว่า SimulatorService ให้ผู้ใช้เรียกผ่านเว็บส่วนติดต่อของผู้ใช้งานและมี SimulatorManager สำหรับจัดการและควบคุม Simulator ที่ดำเนินงานเพื่อจะสั่งการให้ Oozie ประมวลผลกระแสงาน แสดงในภาพประกอบที่ 3-12 และจะอธิบายขั้นตอนและกระบวนการจำลองของกระแสงานดังต่อไปนี้



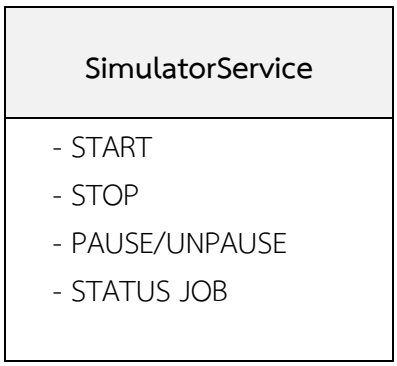
ภาพประกอบที่ 3-12 ขั้นตอนการใช้งานเซอร์วิสการจำลอง

3.5.1 Singleton Pattern

Singleton pattern ถูกนำมาใช้ในระบบการจำลองกระแสงาน เมื่อผู้ใช้เรียก URL หน้าเว็บในส่วนติดต่อของผู้ใช้งาน และสร้างโปรเจคสำหรับจะจำลองดำเนินการของกระแสงาน ระบบก็จะสร้าง Thread ขึ้นมาเมื่อเริ่มการจำลอง โดยการได้รับค่าคงที่ (getInstance) ใน Singleton pattern เพื่อเป็นการป้องกันการสร้าง Thread การทำงานของโปรเจคที่ถูกสร้างไว้แล้ว การจำลองกระแสงานของโปรเจคนั้นจะถูกสร้างขึ้นตอนที่ยังไม่มีข้อมูลคือในครั้งแรกเพียงครั้งเดียวเท่านั้น

3.5.2 SimulatorService

สำหรับ SimulatorService เป็นเซอร์วิสแบบ REST ที่ให้บริการผู้ใช้เรียกใช้งานเพื่อสั่งระบบการจำลองกระแสงาน จะประกอบด้วยฟังก์ชันที่ให้ใช้งานคือ เริ่ม (START) หยุด (STOP) หยุดชั่วคราว (PAUSE/UNPAUSE) และสถานะการทำงาน (STATUS JOB) เพื่อควบคุมในส่วนของการจำลอง ฟังก์ชันเริ่มเป็นฟังก์ชันสำหรับเริ่มต้นการทำงานของระบบการจำลอง เพื่อที่สั่งการส่งกระแสงานในคิวงานที่ผู้ใช้เลือกไปประมวลผลใน Oozie ฟังก์ชันเริ่มยังสามารถเรียกใช้เพื่อดูข้อมูลสถานะของการประมวลผลงานกระแสงานได้ เมื่อต้องการหยุดการส่งกระแสงานไปประมวลผล หรือต้องการสิ้นสุดการจำลองจะต้องเรียกใช้ฟังก์ชันหยุด และฟังก์ชันการหยุดการจำลองในการหยุดส่งกระแสงานไปประมวลผลชั่วขณะ หรือต้องการดำเนินการส่งกระแสงานไปประมวลผลต่อ เรียกใช้ฟังก์ชันหยุดชั่วคราว สำหรับฟังก์ชันสถานะงานเพื่อต้องการตรวจสอบสถานะของกระแสงาน และดูรายละเอียดของการประมวลผลของกระแสงานนั้น เซอร์วิสการเรียกใช้งานดังภาพประกอบที่ 3-13



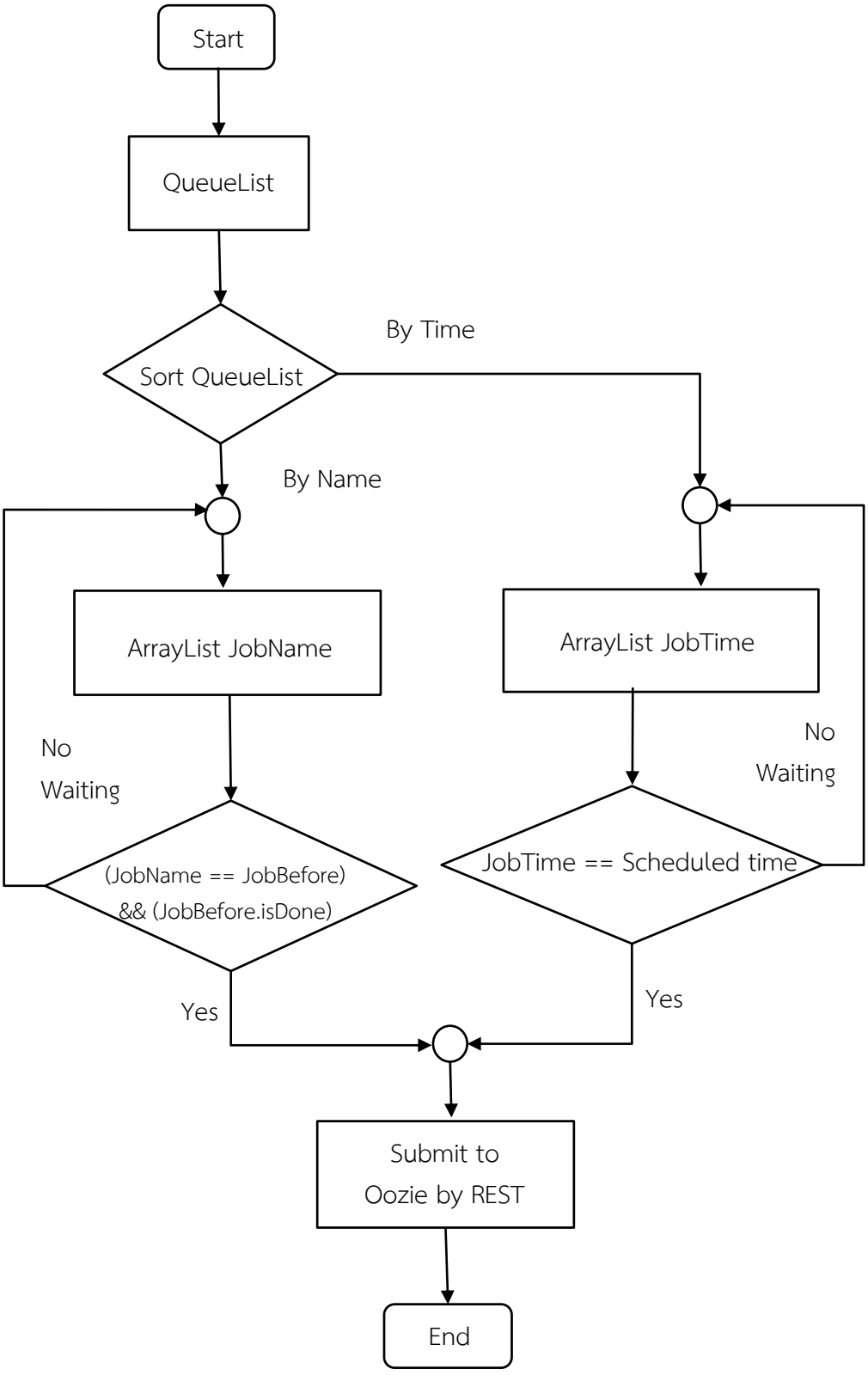
ภาพประกอบที่ 3-13 เซอร์วิสแบบ REST สำหรับสั่งงานการจำลองกระแสงาน

3.5.3 SimulatorManager

SimulatorManager สำหรับไว้จัดการและควบคุมการสร้างโปรเจคของระบบการจำลองกระแสงาน และการสร้างค่าคงที่ของ Singleton pattern ของโปรเจคก็จะถูกสร้างในส่วนของการจัดการนี้ นอกจากนี้ SimulatorManager ยังเป็นตัวคอยจัดการสำหรับรับคำสั่งมาจาก SimulatorService เพื่อสั่งให้ Simulator ดำเนินการตามคำสั่งนั้นต่อไป

3.5.4 Simulator

Simulator เป็นส่วนหลักของการทำงานเพื่อดำเนินการส่งกระแสงานไปประมวลผลใน Oozie สำหรับขั้นตอนกระบวนการดำเนินงานของ Simulator เริ่มจากการตรวจสอบข้อมูลกระแสงานในคิวที่ผู้ใช้งานส่งมาเพื่อประมวลผล หลังจากนั้นเรียงลำดับกระแสงานในคิวตามเวลากำหนดเริ่มต้นที่จะใช้ส่งไปประมวลผลกระแสงาน หรือเรียงลำดับตามกระแสงานหลังจากการประมวลผลของกระแสงานก่อนหน้าเสร็จสิ้นสมบูรณ์ เมื่อเรียงลำดับกระแสงานในคิวเรียบร้อยแล้ว ก็เริ่มต้นการส่งกระแสงานไปประมวลผลตามเงื่อนไข โดยใช้เซอร์วิสแบบ REST ติดต่อกับ Oozie เพื่อให้ Oozie ประมวลผลตามที่ระบุคำสั่งไว้ในงานของกระแสงาน และเมื่อทุกๆ งานของกระแสงานในคิวประมวลผลเสร็จ ก็จะสิ้นสุดการทำงานของ Simulalor ดังภาพประกอบที่ 3-14

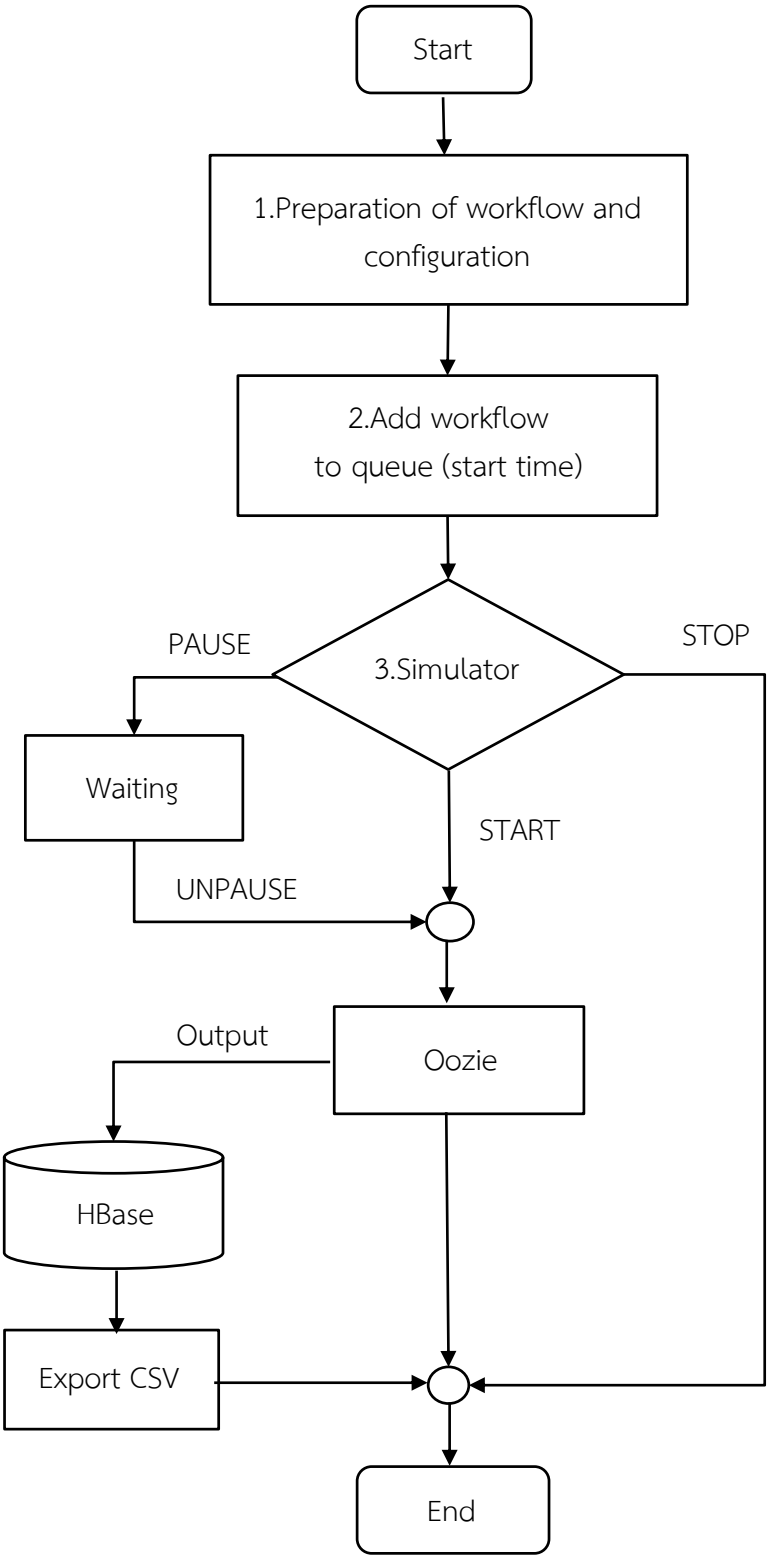


ภาพประกอบที่ 3-14 ขั้นตอนการส่งกระแสงานในคิวไปประมวลผลบน Oozie

3.5.5 กระบวนการระบบจำลองกระแสงาน

ขั้นตอนและกระบวนการของระบบจำลองกระแสงาน เมื่อผู้ใช้ลงทะเบียนหรือเข้าสู่ระบบการจำลองกระแสงานทางเว็บส่วนติดต่อผู้ใช้งาน ครั้งแรกที่เข้าระบบจะต้องสร้างโปรเจคเพื่อใช้ในการจำลองกระแสงาน หรือเมื่อต้องการสร้างโปรเจคเพื่อมาจำลองบุคลากรขององค์กรในรูปแบบอื่นๆ ระบบการจำลองสร้างเซอร์วิสต่างๆ ขึ้นมาอัตโนมัติ และเริ่มต้นกระบวนการในระบบการจำลองกระแสงานดังภาพประกอบที่ 3-15

- ขั้นตอนที่ 1 จะมีการเตรียมข้อมูลของบุคลากร ข้อมูลตำแหน่งหน้าที่ในการทำงานในองค์กร และระบุหน้าที่ให้กับบุคลากร เพื่อใช้สำหรับการตั้งค่าในไฟล์ job.properties และเตรียมไฟล์กระแสงานเพื่อรอในการประมวลผล
- ขั้นตอนที่ 2 เมื่อเตรียมข้อมูลบุคลากรและกระแสงานเรียบร้อยแล้ว สร้างคิวงานและเพิ่มกระแสงานไปเก็บไว้ในคิว พร้อมทั้งตั้งค่าเวลาเริ่มการทำงานในกระแสงานนั้นๆ สามารถเพิ่มกระแสงานได้มากกว่า 1 กระแสงานสำหรับ 1 คิวงาน
- ขั้นตอนที่ 3 ขั้นตอนการจำลอง เรียกใช้เซอร์วิสของ Simulator เพื่อสั่งให้เริ่มต้นส่งกระแสงานไปประมวลผลใน Oozie และผลลัพธ์ของการ ดำเนินงานทั้งหมดจะเก็บข้อมูลใน HBase และสามารถนำออกของข้อมูลการประมวลผลเป็นไฟล์ CSV ขณะการจำลองสามารถสั่งหยุดการจำลองชั่วคราวได้ จนกระทั่งผู้ใช้งานสั่งอีกครั้งเพื่อดำเนินการทำงานต่อไป เมื่อทุกงานในคิวประมวลผลกระแสงานเสร็จก็จะจบระบบการจำลอง หรือผู้ใช้งานสั่งจบระบบการจำลองกระแสงาน



ภาพประกอบที่ 3-15 กระบวนการจำลองของกระแสงาน

3.6 วิธีการทดสอบระบบการจำลองกระแสงาน

ในหัวข้อนี้ประกอบด้วยเครื่องมือที่ใช้ทดสอบระบบ และการทดสอบของระบบการจำลองกระแสงานอัตโนมัติซึ่งมี 2 ส่วน คือ ส่วนแรกการทดสอบระบบการจำลอง และส่วนที่ 2 การทดสอบและวิเคราะห์การประมวลผลกระแสงาน

3.6.1 เครื่องมือที่ใช้ทดสอบระบบ

เครื่องมือที่ใช้ในการจำลอง จะใช้เฉพาะซอฟต์แวร์ในการพัฒนาโปรแกรมของการจำลองเท่านั้น ซึ่งซอฟต์แวร์ทุกตัวที่เลือกใช้เป็นแบบเสรี สามารถใช้พัฒนาได้ฟรี รองรับการประมวลผลแบบกระจาย และสามารถใช้งานร่วมกันได้ ซึ่งประกอบด้วย 5 ซอฟต์แวร์ ดังนี้

1. ระบบปฏิบัติการ Ubuntu เนื่องจากเป็นระบบปฏิบัติการที่เป็นแบบเสรี และรองรับซอฟต์แวร์ที่ใช้ในการประมวลผลกระแสงาน
2. ซอฟต์แวร์ Hadoop สำหรับการประมวลผลแบบกระจาย
3. ซอฟต์แวร์ Oozie สำหรับเป็นเครื่องมือการประมวลผลกระแสงาน
4. ซอฟต์แวร์ HBase สำหรับใช้เก็บข้อมูลบน HDFS ของ Hadoop
5. ซอฟต์แวร์ Tomcat เพื่อใช้เป็นเครื่องให้บริการสำหรับเว็บส่วนติดต่อผู้ใช้งาน

3.6.2 การทดสอบระบบการจำลอง

เนื่องจากระบบการจำลองนั้น ถูกสร้างขึ้นเพื่อเตรียมบุคลากร ตำแหน่งงานในองค์กร และการเตรียมไฟล์กระแสงาน สำหรับเป็นข้อมูลใช้ในการประมวลผลกระแสงาน การทดสอบของระบบการจำลองประกอบด้วย 3 ส่วนด้วยกันคือ ส่วนแรกเป็นการทดสอบไฟล์ workflow.xml ไฟล์กระแสงานที่สร้างขึ้นเพื่อใช้ในการประมวลผล จะต้องมีโหนดควบคุมที่ถูกต้องตามรูปแบบของ Oozie และจะต้องมีโหนดการกระทำที่ถูกต้องตามรูปแบบที่กำหนดใน AsyncAction จึงต้องมีตรวจสอบไฟล์จากกราฟก่อนสร้างเป็นไฟล์ workflow.xml ส่วนที่ 2 เป็นการทดสอบฟังก์ชันการใช้งานของ SimulatorService ที่มีให้บริการของ Simulator สำหรับการเรียกใช้งานฟังก์ชันเริ่ม หยุด หยุดชั่วคราว และสถานะการทำงาน และส่วนที่ 3 เป็นการทดสอบการเตรียมข้อมูล เซอร์วิสแบบ REST ที่มีให้เรียกใช้เพื่อเก็บข้อมูลลงในฐานข้อมูล HBase โดยเรียกผ่านชุดคำสั่ง WebObject API เพื่อเป็นการตรวจสอบการทำงานของฟังก์ชัน การสร้าง อ่าน แก้ไข และลบข้อมูล ซึ่งผลการทดสอบแสดงในบทที่ 4 ข้อที่ 4.2.1 เรื่องผลการทดสอบไฟล์ workflow.xml ข้อที่ 4.2.2 เรื่องผลการทดสอบฟังก์ชันของ Simulator และข้อที่ 4.3 เรื่องผลการทดสอบการใช้งานเซอร์วิสสำหรับเตรียมข้อมูลในการประมวลผล ตามลำดับ

3.6.2.1 การทดสอบไฟล์ workflow.xml

เนื่องจากไฟล์กราฟหรือ DAG ที่ได้สร้างขึ้นจากโปรแกรม DAGGEN จะสร้างกราฟการทำงานที่เป็นแบบวัฏจักรหรือวนรอบให้มาด้วย จึงต้องสร้างโปรแกรมเพื่อตรวจสอบและคัดกรองเฉพาะกราฟการทำงานแบบเส้นตรง และกราฟการทำงานแบบขนานเท่านั้น เพื่อจะได้ถูกต้องตามรูปแบบโหนดควบคุมของกระแสงานใน Oozie และต้องตรวจสอบองค์ประกอบของโหนดการกระทำ ได้กำหนดไว้ถูกต้องแล้วหรือไม่ตามที่ระบุไว้ใน AsyncAction ที่ได้สร้างขึ้น ก่อนที่จะแปลงไฟล์เป็น workflow.xml

3.6.2.2 ฟังก์ชันการใช้งานของ Simulator

สำหรับการทดสอบฟังก์ชันการใช้เริ่ม หยุด หยุดชั่วคราว และสถานะการทำงานที่มีให้เรียกใช้ใน SimulatorService เพื่อการทดสอบการทำงานของเซอร์วิสให้ถูกต้อง โดยการสร้างเว็บส่วนติดต่อของผู้ใช้เพื่อใช้งานฟังก์ชันต่างๆ ดังกล่าว นอกจากนี้ส่วนของ SimulatorManager เพื่อใช้จัดการและควบคุมการจำลองเพื่อให้โปรเจคที่สร้างขึ้นไม่เกิดความซ้ำซ้อนกัน จึงได้ทดสอบโดยสร้างโปรเจคขึ้นมาใช้ในการจำลองหลากหลายโปรเจคด้วยกัน ส่วนการทำงานของ Simulator ได้สร้างคิวงานและเพิ่มกระแสงานที่จะส่งไปประมวลผลไปเก็บไว้ในคิว เพื่อทดสอบการทำงานของ Simulator เรื่องการจัดการส่งกระแสงานไปประมวลผลบน Oozie ตามลำดับของคิวที่ได้กำหนด

3.6.2.3 การเตรียมข้อมูล

สำหรับการเตรียมข้อมูลโครงสร้างขององค์กร คือบุคลากรและบทบาทตำแหน่งการทำงาน เพื่อใช้ในการจำลองการประมวลผลกระแสงาน ผู้วิจัยได้สร้างเซอร์วิสแบบ REST สำหรับเรียกใช้ในการเก็บข้อมูลเรียกว่า AllService เนื่องจากโครงสร้างการเก็บข้อมูล ต้องมีการลงทะเบียนการใช้งานและได้ออกแบบการจำลองให้สามารถสร้างโปรเจคสำหรับจำลองเป็นองค์กรหรือสำนักงานใดๆ ซึ่งสามารถสร้างโปรเจคการจำลองได้หลายโปรเจคเพื่อที่จำลองบุคคลที่แตกต่างกัน จึงจำเป็นต้องตรวจสอบความถูกต้องของ AllService ที่ใช้ในการเก็บบันทึกข้อมูลของแต่ละโปรเจค โดยการตรวจสอบฟังก์ชัน สร้าง อ่าน แก้ไข และลบ แสดงผลลัพธ์การทำงานในหน้าเว็บส่วนของผู้ใช้งาน

3.6.3 การทดสอบการประมวลผลกระแสงาน

ในการทดสอบและวิเคราะห์การประมวลผลของกระแสงาน ผู้วิจัยได้สร้างโปรเจคในการจำลองสร้างโครงสร้างขององค์กรขนาดกลาง เพื่อใช้ระบบจำลองนี้ในการวิเคราะห์กระแสงานว่ามีความล่าช้าในขั้นตอนใดบ้าง องค์กรจำลองนี้มีจำนวน 5 แผนก ในแต่ละแผนกมีกระแสงานประมาณจำนวน 20 กระแสงาน ซึ่งเท่ากับว่าจะต้องจำลองกระแสงานทั้งหมดจำนวน 100 กระแสงาน โดยในแต่ละกระแสงานที่มีความซับซ้อนปานกลาง จะมีจำนวนขั้นตอนการทำงานประมาณ 10 ขั้นตอนด้วยกัน และออกแบบโครงสร้างองค์กรให้มีบุคลากรและบทบาทการทำงานคือ มีตำแหน่งการทำงาน 5 บทบาท

ในการทดสอบจะเริ่มจากจำลองบุคลากรตำแหน่งละ 1 คน จำลองการทำงานทั้ง 100 กระแสงานแล้วคำนวณหาค่าเปอร์เซ็นต์ความล่าช้า (สมการที่ 1 ในหน้าที่ 18) เพื่อวิเคราะห์ว่าจำนวนคนเพียงพอหรือไม่ และกระแสงานใดที่มีเส้นทางทำให้เกิดความล่าช้าบ้าง จากนั้นจะทดลองเพิ่มจำนวนบุคลากรในบทบาทที่ล่าช้ามากที่สุด เพื่อแบ่งเบาภาระงาน แล้ววัดผลการจำลองเพื่อดูว่าได้ผลตามสมมติฐานหรือไม่

บทที่ 4
ผลการวิเคราะห์ข้อมูล

ในบทนี้จะกล่าวถึงรายละเอียดของการทดสอบระบบการจำลองกระแสงาน ผลการทดสอบการทำงานระบบจำลอง ฟังก์ชันเริ่ม หยุด หยุดชั่วคราว และสถานะการทำงาน ผลการทดสอบเซอร์วิสการเก็บข้อมูลในแต่ละโปรเจคของการจำลอง และผลการทดสอบการประมวลผลของกระแสงานทั้งหมด เพื่อใช้วิเคราะห์การเพิ่มบุคคลในบางบทบาทของตำแหน่งงาน รายละเอียดการทดสอบเป็นดังต่อไปนี้

4.1 รายละเอียดการทดสอบ

รายละเอียดในการดำเนินการทดสอบ ระบบการจำลองกระแสงานอัตโนมัติสำหรับองค์กรใดๆ นั้น ได้จัดทำขึ้นบนเครื่องคอมพิวเตอร์เสมือน Oracle VM VirtualBox จำนวน 1 เครื่อง โดยใช้ระบบปฏิบัติการ Ubuntu 14.04.5 LTS 64 Bit ใช้ CPU Intel® Core™ i5-2500 ความเร็ว 3.30GHz หน่วยประมวลผล 4 Core และหน่วยความจำ 8 GBs ติดตั้งซอฟต์แวร์การประมวลผลแบบกระจาย Hadoop เวอร์ชัน 1.2.1 ติดตั้ง HBase เวอร์ชัน 0.96.2 ติดตั้งเครื่องให้บริการ Tomcat เวอร์ชัน 1.7.0_151 และติดตั้งซอฟต์แวร์สำหรับเป็นเครื่องมือการประมวลผลกระแสงาน Oozie เวอร์ชัน 3.3.2

รายละเอียดการทดสอบและวิเคราะห์การประมวลผลของกระแสงาน จำนวนกระแสงานที่ใช้สำหรับทดสอบการประมวลผล รูปแบบองค์กรหรือรูปแบบสำนักงานกล่าวคือจำนวนบุคลากรในบทบาทหน้าที่งานที่ใช้ในการจำลอง จำนวนครั้งของการทดสอบการประมวลผลกระแสงาน ซึ่งจำนวนกระแสงานในการทดสอบใช้ไฟล์กระแสงานทั้งหมด 100 กระแสงาน และในการทดสอบสมมติบทบาทโหนดการกระทำของกระแสงานมี 5 บทบาทด้วยกัน สมมติคนสำหรับทำงานในแต่ละบทบาทเพียงแค่ 1 คน การทดสอบประมวลผลกระแสงานทั้ง 100 กระแสงาน โดยการสร้างคิวงานและเพิ่มกระแสงานทั้ง 100 กระแสงานไปเก็บไว้ในคิว ตั้งค่าเริ่มต้นการทำงานของแต่ละกระแสงานเป็นแบบดำเนินการต่อจากงานของกระแสงานที่กำหนด และกระแสงานก่อนหน้าต้องประมวลผลเสร็จ จำนวนครั้งในการทดสอบการประมวลกระแสงานทั้งหมดในคิวนั้นคือ 1 ครั้ง วิเคราะห์ผลการดำเนินงานในโหนดการกระทำในแต่ละงานของกระแสงานทั้ง 100 กระแสงาน วิเคราะห์เปอร์เซ็นต์ความล่าช้าในแต่ละบทบาทงานของทุกกระแสงาน หาค่าเปอร์เซ็นต์ความล่าช้าสูงสุดในแต่ละบทบาททั้งหมดทุกกระแสงานที่ประมวลผล เพื่อใช้พิจารณาเพิ่มบุคคลในบทบาทที่มีเปอร์เซ็นต์ความล่าช้าสูงสุด แล้ว

ประมวลผลกระแสงานนั้นอีกครั้ง เพื่อวิเคราะห์เปอร์เซ็นต์ความล่าช้าในแต่ละบทบาทของกระแสงานที่ประมวลผลซ้ำนั้นอีกครั้ง

4.2 ผลการทดสอบการทำงานระบบจำลอง

ก่อนเริ่มต้นการจำลองและประมวลผลกระแสงาน สั่งการเพื่อเริ่มต้นเซิร์ฟเวอร์ของ Tomcat ใช้เป็นเครื่องให้บริการสำหรับดำเนินการในการจำลอง เนื่องจากระบบการจำลองออกแบบเป็นเว็บส่วนติดต่อกับผู้ใช้ซึ่งจะดำเนินการอยู่บน Tomcat และผลการทดสอบของการทำงานในระบบการจำลองกระแสงานเป็นดังต่อไปนี้

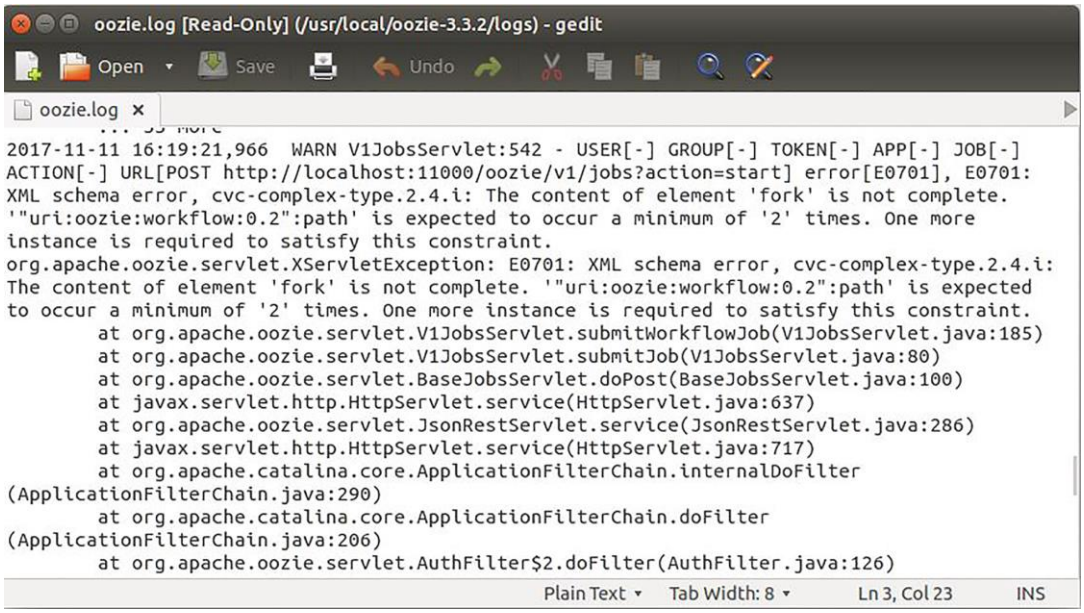
4.2.1 ผลการทดสอบไฟล์ workflow.xml

เนื่องจาก Oozie ไม่รองรับกราฟของกระแสงานที่เป็นแบบวัฏจักร แต่โปรแกรม DAGGEN ที่ใช้ในการสังเคราะห์กราฟนั้น สุ่มและสร้างกราฟของ DAG ให้ออกมาทุกรูปแบบ ทำให้ไฟล์กราฟที่ได้แบบวัฏจักรเป็นผลลัพธ์ออกมาด้วย จำนวนไฟล์ workflow.xml ที่ต้องการใช้ในระบบการจำลองเป็นจำนวนทั้งหมด 100 ไฟล์กระแสงาน เมื่อได้ทดสอบสุ่มสร้างการสังเคราะห์ไฟล์กราฟครั้งละ 100 ไฟล์ ได้แปลงไฟล์เป็น workflow.xml ผลของการแปลงไฟล์ได้เป็นจำนวนประมาณ 25 เปอร์เซ็นต์ ที่สามารถใช้งานได้และไฟล์นั้นอยู่ในเงื่อนไขกราฟของ Oozie คือเป็นกราฟแบบเส้นตรงและกราฟแบบขนานกัน จึงได้ทดสอบสังเคราะห์ไฟล์กราฟเพิ่มเติมเพื่อให้ได้ workflow.xml ครบ 100 ไฟล์กระแสงานตามที่ต้องการ ผลการทดสอบต้องสุ่มสร้างไฟล์กราฟถึงจำนวน 405 ไฟล์ จึงจะได้ครบและอยู่ในเงื่อนไขการทำงานของ Oozie สรุปจำนวนเปอร์เซ็นต์ไฟล์กระแสงานที่แปลงออกมาจากการสังเคราะห์ของ DAGGEN และสามารถใช้ในการประมวลผลได้คือ 24.69 เปอร์เซ็นต์

เมื่อนำไฟล์กระแสงานที่เป็นแบบวัฏจักรมาทดสอบ โดยสั่งการทำงานของฟังก์ชันเริ่มต้นใน SimulatorService เพื่อส่งกระแสงานนั้นไปประมวลผล จะเกิดความผิดพลาดขึ้นดังภาพประกอบที่ 4-1 คือผู้ใช้งานส่งข้อมูลเพื่อให้ Oozie ประมวลผลไม่ถูกต้อง และในขณะเดียวกันเมื่อตรวจสอบการสั่งประมวลผลกระแสงานดังกล่าวในระบบของ Oozie จะพบความผิดพลาดโดยมีข้อความคือ E0701:XML Schema error ดังแสดงในภาพประกอบที่ 4-2 ซึ่งเกิดจากไฟล์กระแสงาน workflow.xml เป็นแบบวัฏจักร ไม่มีความถูกต้องตรงตามระบบที่ Oozie สามารถประมวลผลได้ เนื่องจากโหนดควบคุมการไหลของกระแสงาน Fork และ Join ไม่ครบคู่ในเงื่อนไขของโหนดการทำงาน เมื่อระบบของ Oozie ตรวจแล้วพบข้อผิดพลาดดังกล่าว จึงแจ้งเกิดข้อผิดพลาดกลับมายังเซอร์วิสเป็น status 400 และไม่สามารถประมวลผลกระแสงานดังกล่าวได้

```
POST http://localhost:11000/oozie/v1/jobs?action=start returned a response status of 400 Bad Request
```

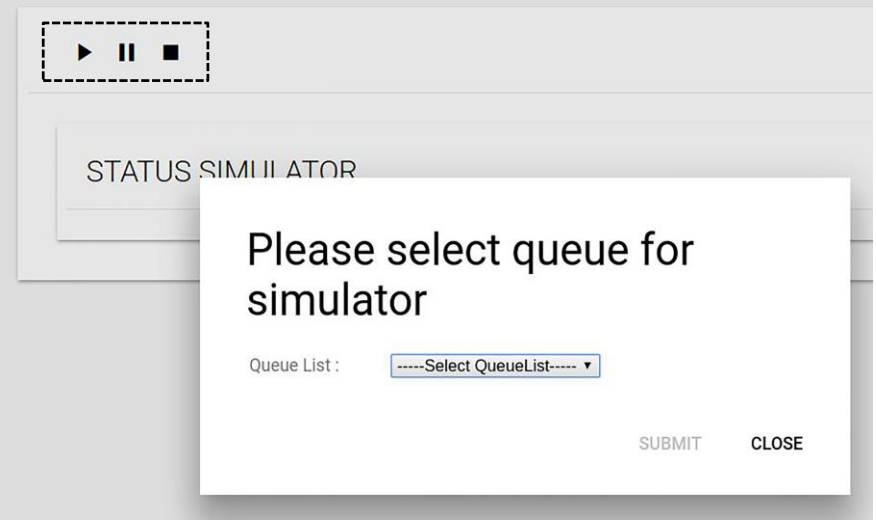
ภาพประกอบที่ 4-1 ข้อผิดพลาดของ SimulatorService เมื่อประมวลผลกราฟที่เป็นแบบวัฏจักร



ภาพประกอบที่ 4-2 ข้อผิดพลาดของ Oozie ที่เกิดจากการประมวลผลกราฟที่เป็นแบบวัฏจักร

4.2.2 ผลการทดสอบฟังก์ชันของ Simulator

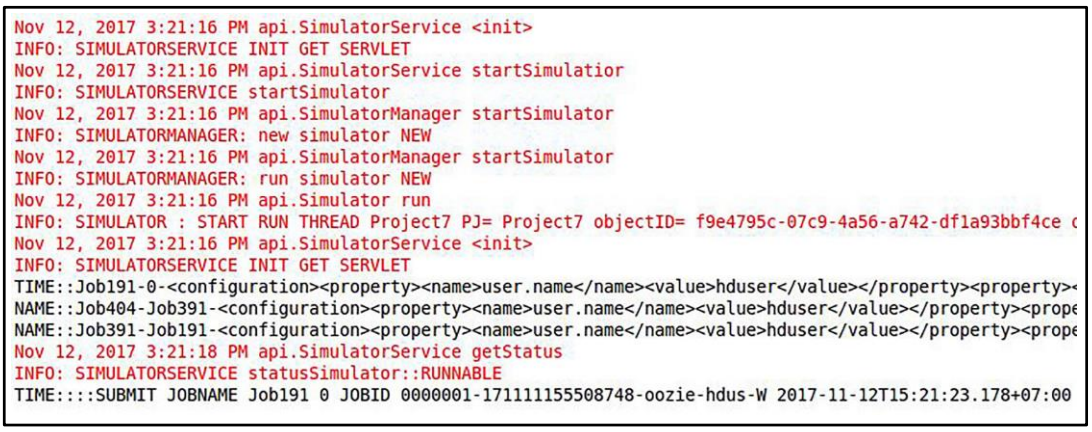
สำหรับผลการทดสอบการดำเนินงานของการจำลอง SimulatorService ในการควบคุมเพื่อให้ Oozie ดำเนินการประมวลผลกระแสงานนั้น จะมีฟังก์ชันเริ่มสำหรับผู้ใช้งานเพื่อส่งคิวที่ได้เพิ่มกระแสงานไปประมวลผล ฟังก์ชันสำหรับหยุดชั่วคราวและดำเนินการส่งกระแสงานในคิวที่เหลือไปประมวลผลต่อไป และฟังก์ชันหยุดการทำงานเป็นการหยุดส่งกระแสงานในคิวไปประมวลผลใน Oozie ภาพประกอบที่ 4-3 แสดงการเรียกใช้งานของเซอร์วิส ซึ่งอยู่กรอบเส้นประด้านบนซ้ายของรูปภาพ และเรียงลำดับฟังก์ชันการทำงานจากซ้ายไปขวา (เริ่ม หยุด และหยุดชั่วคราว)



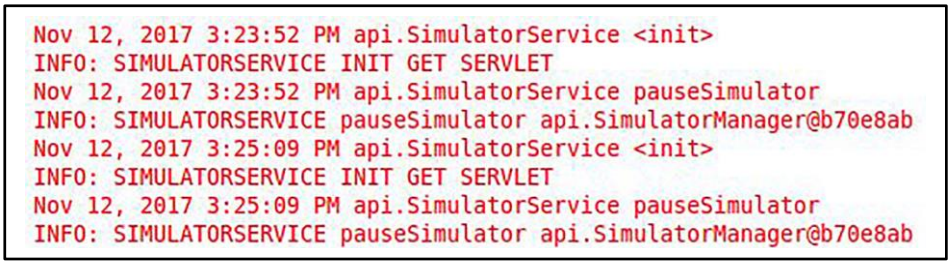
ภาพประกอบที่ 4-3 ฟังก์ชันการทำงานของ Simulator

เมื่อผู้ใช้งานเลือกคิวของกระแสงานและส่งไปประมวลผล โดยผ่านฟังก์ชันใน SimulatorService แล้ว SimulatorManager ก็จะตรวจสอบโปรเจคที่ส่งมาเพื่อดำเนินการ ถ้ามีการทำงานของโปรเจคนั้นอยู่แล้วก็จะส่งค่าคงที่ของโปรเจคกลับมาให้เซอร์วิส แต่ถ้าโปรเจคถูกส่งมาครั้งแรกก็สร้างค่าคงที่ขึ้นมาใหม่ตามการทำงานของ Singleton pattern เพื่อกำหนดค่าคงที่ให้กับโปรเจค หลังจากนั้นใน Simulator จะทำงานตามคำสั่งที่ได้รับมาจากเซอร์วิส ถ้าหากผู้ใช้เริ่มต้นส่งกระแสงานในคิวมาประมวลผล ก็จะตรวจสอบคิวมีกระแสงานอยู่จำนวนเท่าไหร่ แล้วเรียงลำดับกระแสงานในคิว เพื่อรอเวลาในการส่งกระแสงานนั้นไปประมวลผลบน Oozie (รายละเอียดในภาพประกอบที่ 3-14 ขั้นตอนการส่งกระแสงานในคิวไปประมวลผลบน Oozie) และตัวอย่างผลลัพธ์การเริ่มต้นประมวลผลกระแสงานแสดงในภาพประกอบที่ 4-4 เป็นการประมวลผลกระแสงานในโปรเจคซึ่งมีจำนวนกระแสงานทั้งหมด 3 กระแสงานในคิว

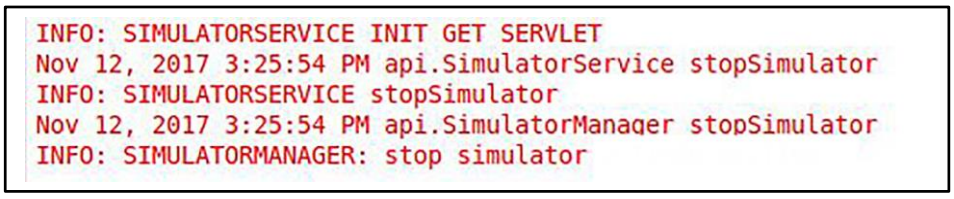
ภาพประกอบที่ 4-5 แสดงผลของการทดสอบฟังก์ชัน เพื่อจะให้ระบบการจำลองหยุดการดำเนินงานชั่วคราว และผู้ใช้สามารถเรียกใช้ฟังก์ชันเดิมอีกครั้งเพื่อให้ระบบการจำลองดำเนินงานต่อไป และภาพประกอบที่ 4-6 แสดงผลของการทดสอบการทำงานเมื่อเรียกใช้ฟังก์ชันเพื่อหยุดการดำเนินงานในระบบจำลอง



ภาพประกอบที่ 4-4 ฟังก์ชันเริ่มการทำงานของ Simulator



ภาพประกอบที่ 4-5 ฟังก์ชันหยุดการทำงานชั่วคราวของ Simulator



ภาพประกอบที่ 4-6 ฟังก์ชันหยุดการทำงานของ Simulator

นอกจากนี้ในขณะที่ประมวลผลกระแสงานในคิวนั้น สามารถดูสถานะงานของกระแสงานในคิวได้ โดย SimulatorService และเรียกผ่านหน้าเว็บส่วนติดต่อของผู้ใช้งาน เพื่อใช้ในการตรวจสอบจำนวนของกระแสงานที่ประมวลผลเสร็จเรียบร้อยแล้ว และเพื่อดูสถานะการทำงานของกระแสงานที่กำลังประมวลผล ผลลัพธ์การเรียกดูสถานะการประมวลผลของกระแสงานดังแสดงในภาพประกอบที่ 4-7

STATUS SIMULATOR

JobID:0000000-171111155508748-oozie-hdus-W	Status	RUNNING
	McreatedTime	Sat, Nov 11 2017, 16:30:26.4770
	MstartTime	Sat, Nov 11 2017, 16:30:26.5700
	MendTime	

Name	Status	StartTime	EndTime	Role	User	Delay	UWTime	PTime
action7	RUNNING	Nov 11 2017, 16:41:04.3110		C	user3	0	1	142060
action6	OK	Nov 11 2017, 16:32:12.6600	Nov 11 2017, 16:33:25.1440	C	user3	0	1	68710
action2	OK	Nov 11 2017, 16:30:33.1390	Nov 11 2017, 16:41:03.8480	B	user6	0	1	621980
action1	OK	Nov 11 2017, 16:30:33.0450	Nov 11 2017, 16:32:12.0970	D	user4	0	1	92010
action3	OK	Nov 11 2017, 16:30:33.1690	Nov 11 2017, 16:31:06.8930	E	user5	0	1	28990
action0	OK	Nov 11 2017, 16:30:26.9590	Nov 11 2017, 16:30:32.3290	E	user5	0	1	0

ภาพประกอบที่ 4-7 สถานะการประมวลผลของกระแสงาน

4.3 ผลการทดสอบการใช้งานเซอร์วิสสำหรับเตรียมข้อมูลในการประมวลผล

ผลการทดสอบการเก็บข้อมูลใน AllService เพื่อเตรียมข้อมูลสำหรับใช้ในการทดสอบและวิเคราะห์การดำเนินงานของกระแสงานในองค์กร ในการทดสอบเมื่อผู้ใช้ลงทะเบียนระบบก็จะสร้าง AccountName เป็นรหัสเพื่อใช้ใน URL การเก็บข้อมูล และผู้ใช้งานต้องสร้างโปรเจคคือ ProjectName ขึ้นมาสำหรับใช้ในการจำลอง

ตัวอย่างการทดสอบเก็บข้อมูลบุคคล สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-8 โดย object_name คือชื่อบุคคลที่ต้องการ ในกรณีนี้คือ user1 ซึ่งได้บันทึกค่าคุณสมบัติของ user1 มี status = “free” คือพร้อมที่จะทำงาน salary = “15000” คือค่าเงินเดือน workingtime = “1” ค่าน้ำหนักของการทำงาน และ role = “A” มีตำแหน่งการทำงานที่เกี่ยวกับบทบาท A เมื่อเรียกดูข้อมูลได้ผลลัพธ์ดังภาพประกอบที่ 4-9 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

```
URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/
{AccountName}/{ProjectName}/user/{object_name}
```

ภาพประกอบที่ 4-8 เซอร์วิสเก็บข้อมูลบุคลากร


```
{       "object_name": "user1",
        "property": {
           "status": "free",
           "salary": "15000",
           "workingtime": "1",
           "role": "A"  },
       "path": "/Project1/user/user1",
       "created": 1507482894139, "modified": 1509365369175
}
```


ภาพประกอบที่ 4-9 ผลลัพธ์เซอร์วิสเก็บข้อมูลบุคลากร

ตัวอย่างการทดสอบเก็บข้อมูลบทบาทตำแหน่งงาน สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-10 โดย object_name คือบทบาทที่ต้องการ ในกรณีคือ A ซึ่งได้บันทึกค่าคุณสมบัติของ A มี user1 = “6f4ad664-e38b-4de9-931c-3512a1ab521d” คือมี user1 และรหัสของ user1 อยู่ในบทบาท A เมื่อเรียกดูข้อมูลได้ผลลัพธ์ดังภาพประกอบที่ 4-11 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/{AccountName}/{ProjectName}/role/{object_name}

ภาพประกอบที่ 4-10 เซอร์วิสเก็บข้อมูลบทบาทตำแหน่งงาน


```
{
        "object_name": "A",
        "object_id": "297f2bcf-d99d-4c8a-ad00-e3ee6913c5ec",
        "is_file": false,
        "object_type": "",
        "property": {
            "user1": "6f4ad664-e38b-4de9-931c-3512a1ab521d"
        },
        "path": "/Project1/role/A",
        "created": 1507482952734,
        "modified": 1507482959842
}
```


ภาพประกอบที่ 4-11 ผลลัพธ์เซอร์วิสเก็บข้อมูลบทบาทตำแหน่งงาน

ตัวอย่างการทดสอบเก็บข้อมูลกระแสงาน สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-12 โดย object_name คือชื่อกระแสงานที่ต้องการ ในกรณีนี้คือ Workflow002 ซึ่งได้บันทึกค่าคุณสมบัติของ Workflow002 มี path = “/user/hduser/apps/ f9e47 95c-07c9-4a56-a742-df1a93bbf4ce/Project1/workflow/Workflow002/” คือเส้นทางของไฟล์กระแสงาน และ workflow_name = “workflow.xml” คือไฟล์ที่อธิบายขั้นตอนการทำงาน เมื่อเรียกดูข้อมูลได้ผลลัพธ์ดังภาพประกอบที่ 4-13 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/
{AccountName}/{ProjectName}/workflow/{object_name}

ภาพประกอบที่ 4-12 เซอร์วิสการเก็บข้อมูลของกระแสงาน


```
{
        "object_name": "Workflow002",
        "object_id": "e009dd0c-158e-44ee-8367-887501d4ef5f",
        "is_file": false,
        "object_type": "",
        "property": {
            "path": "/user/hduser/apps/f9e4795c-07c9-4a56-a742-
            df1a93bbf4ce/Project1/workflow/Workflow002/",
            "workflow_name": "workflow.xml"
        },
        "path": "/Project1/workflow/Workflow002",
        "created": 1507483646634,
        "modified": 1507483646747
}
```


ภาพประกอบที่ 4-13 ผลลัพธ์เซอร์วิสการเก็บข้อมูลกระแสงาน

ตัวอย่างการทดสอบเก็บข้อมูลงานของกระแสงาน สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-14 โดย object_name คือชื่องานของกระแสงานที่ต้องการ ในกรณีนี้คือ Job002 ซึ่งได้บันทึกค่าคุณสมบัติของ Job002 มี objectid_account = “f9e4795c-07c9-4a56-a742-df1a93bbf4ce” คือรหัสของผู้ใช้งาน path = “hdfs://localhost:8020/user/hduser/apps/f9e4795c-07c9-4a56-a742df1a93bbf4ce/Prject1/workflow/Workflow002/” คือเส้นทางของไฟล์กระแสงานบน HDFS และ project = “Project1” คือชื่อโปรเจคที่สร้างการจำลอง ดังภาพประกอบที่ 4-15 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/{AccountName}/{ProjectName}/workflowjob/{object_name}

ภาพประกอบที่ 4-14 เซอร์วิสการเก็บข้อมูลงานของกระแสงาน


```
{
        "object_name": "Job002",
        "object_id": "1de8b825-0d9b-453b-b65c-03e13ea803c7",
        "is_file": false,
        "object_type": "",
        "property": {
           "objectid_account": "f9e4795c-07c9-4a56-a742-df1a93bbf4ce",
           "oozie.wf.application.path": "hdfs://localhost:8020/
          user/hduser/apps/f9e4795c-07c9-4a56-a742-df1a93bbf4ce/
          Project1/workflow/Workflow002/",
           "project": "Project1"
        },
        "path": "/Project1/workflowjob/Job002",
        "created": 1507483646885,
        "modified": 1507483646972  }
```


ภาพประกอบที่ 4-15 ผลลัพธ์เซอร์วิสการเก็บข้อมูลงานของกระแสงาน

ตัวอย่างการทดสอบเก็บข้อมูลคิว สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-16 โดย object_name คือชื่อคิวที่ต้องการ ในกรณีนี้คือ QueueAll ซึ่งได้บันทึกค่าคุณสมบัติของ QueueAll มี starttimesim = “2017-09-09T13:42:24.239” คือเวลาที่เริ่มต้นประมวลผลกระแสงานในคิว ดังภาพประกอบที่ 4-17 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

```
URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/
{AccountName}/{ProjectName}/queue/{object_name}/
```

ภาพประกอบที่ 4-16 เซอร์วิสการเก็บข้อมูลคิว


```
{
        "object_name": "QueueAll",
        "object_id": "3c634074-6ceb-4884-ba7f-45b277d2303d",
        "is_file": false,
        "object_type": "",
        "property": {
            "starttimesim": "2017-09-09T13:42:24.239"
        },
        "path": "/Project1/queue/QueueAll",
        "created": 1507531350078,
        "modified": 1507531350159
}
```


ภาพประกอบที่ 4-17 ผลลัพธ์เซอร์วิสการเก็บข้อมูลคิว

ตัวอย่างการทดสอบเก็บข้อมูลบุคคลโหนดลูกของคิว สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-18 โดย queuename คือชื่อคิว QueueAll และ object_name คือชื่อโหนดลูกของคิว ซึ่งเป็นงานของกระแสงานที่ต้องการเก็บในคิว ในกรณีนี้คือ Job002 ซึ่งได้บันทึกค่าคุณสมบัติของ Job002 มี starttimejob = “0” คือเวลาเริ่มต้นที่กระแสงานนี้เริ่มทำงานในคิว jobproperty คือคุณสมบัติของงานใช้กำหนดเพื่อประมวลผล ซึ่งจะประกอบด้วย <property>

กับ <value> ในที่นี้ระบุอยู่ใน Job002 ตามภาพประกอบที่ 4-15 ที่ได้กล่าวไปแล้ว และ jobId = "0000000-171026142235986-oozie-hdus-W" คือหมายเลขของการประมวลผลกระแสงานนั้น และจะมีการอัพเดทเมื่อเริ่มต้นการทำงานของกระแสงาน ดังภาพประกอบที่ 4-19 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

```
URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/
{AccountName}/{ProjectName}/queue/{queuename}/{object_name}
```

ภาพประกอบที่ 4-18 เซอร์วิสการเก็บข้อมูลงานของกระแสงานในโหนดลูกของคิว


```
{       "object_name": "Job002",
        "property": {
            "jobId": "0000000-171026142235986-oozie-hdus-W",
            "starttimejob": "0",
            "jobproperty": "<configuration>
                    <property><name>user.name</name>
                    <value>hduser</value></property>
                    <property><name>objectid_account</name>
                    <value>f9e4795c-07c9-4a56-a742-
                            df1a93bbf4ce</value></property>
                    <property><name>oozie.wf.appliction.path</name>
                    <value>hdfs://localhost:8020/user/hduser
                    /apps/f9e4795c-07c9-4a56-a742-df1a93bbf4ce
                    /Project1/workflow/Workflow002/</value>
                    </property><property><name>project</name>
                    <value>Project1</value></property></configuration>" },
        "path": "/Project1/queue/QueueAll/Job002",
        "created": 1507818500610,
        "modified": 1509003615546
}
```


ภาพประกอบที่ 4-19 ผลลัพธ์เซอร์วิสการเก็บข้อมูลงานของกระแสงานในโหนดลูกของคิว

ตัวอย่างการทดสอบเก็บข้อมูลการประมวลผลกระแสงาน สามารถเรียกผ่านเซอร์วิสด้วยรูปแบบ URL ดังภาพประกอบที่ 4-20 โดย object_name คือชื่อหมายเลขของการประมวลผลกระแสงานที่ต้องการ ในกรณีนี้คือ 0000000-171026142235986-oozie-hdus-W ซึ่งได้บันทึกค่าคุณสมบัติมี starttimejob และ endtimejob คือเวลาที่เริ่มต้นและเวลาที่สิ้นสุดการประมวลผลกระแสงาน status คือสถานการประมวลผล appname คือการประมวลผลในโหนด AsyncAction apppath คือเส้นของไฟล์กระแสงานที่ใช้ในการประมวลผล project_name คือโปรเจคที่ใช้ประมวลผล jobid คือหมายเลขของการประมวลผลกระแสงาน ดังภาพประกอบที่ 4-21 หมายความว่าเซอร์วิสนี้ใช้งานได้ถูกต้อง

URL: http://{hostname}/OozieWebServicesAPI/simulate/AllService/{AccountName}/{ProjectName}/log/{object_name}

ภาพประกอบที่ 4-20 เซอร์วิสการเก็บข้อมูลของการประมวลผลกระแสงาน


```
{
        "object_name": "0000000-171026142235986-oozie-hdus-W",
        "object_id": "b836c6a4-5990-479f-881c-9bd0d4b7d7c9",
        "property": {
            "starttimejob": "2017-10-26T14:40:15.267+07:00",
            "endtimejob": "2017-10-26T15:10:07.009+07:00",
            "status": "SUCCEEDED",
            "appname": "async-time-wf",
            "apppath": "hdfs://localhost:8020/user/hduser/apps/f9e4795c-
          07c9-4a56-a742-df1a93bbf4ce/Project1/workflow/Workflow002/",
            "project_name": "Project1",
            "jobid": "0000000-171026142235986-oozie-hdus-W"  },
        "path": "/Project1/log/job/0000000-171026142235986-oozie-hdus-W",
        "created": 1509005405220, "modified": 1509005407278 }
```


ภาพประกอบที่ 4-21 ผลลัพธ์เซอร์วิสการเก็บข้อมูลของการประมวลผลกระแสงาน

4.4 ผลการทดสอบและการวิเคราะห์การประมวลผลกระแสงาน

เพื่อทดสอบการประมวลผลกระแสงาน ในสภาพแวดล้อมการจำลองที่กำหนดไว้ในหัวข้อ 3.6.3 นั้นคือ จำลององค์กรที่มี 5 บทบาทและมี 100 กระแสงาน ผลลัพธ์การประมวลผลกระแสงานมีรายละเอียดดังต่อไปนี้

4.4.1 ผลการประมวลกระแสงาน 100 กระแสงาน

จากผลลัพธ์การประมวลผลกระแสงานทั้งหมด โดยจำลองบุคลากรตำแหน่งละ 1 คน สามารถคำนวณค่าเปอร์เซ็นต์ความล่าช้าของแต่ละบุคคลในแต่ละขั้นตอนได้จากสมการที่ 1 แต่เนื่องจากใน 1 กระแสงานบุคคลนั้นๆ อาจจะทำหลายขั้นตอน จึงต้องคำนวณค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้า (Avg. %delay) ในแต่ละกระแสงาน ได้จากสมการที่ 2 เมื่อ n คือ จำนวนขั้นตอนที่บทบาทนั้นต้องทำงาน เพื่อใช้ในการวิเคราะห์บุคคลในการทำงาน

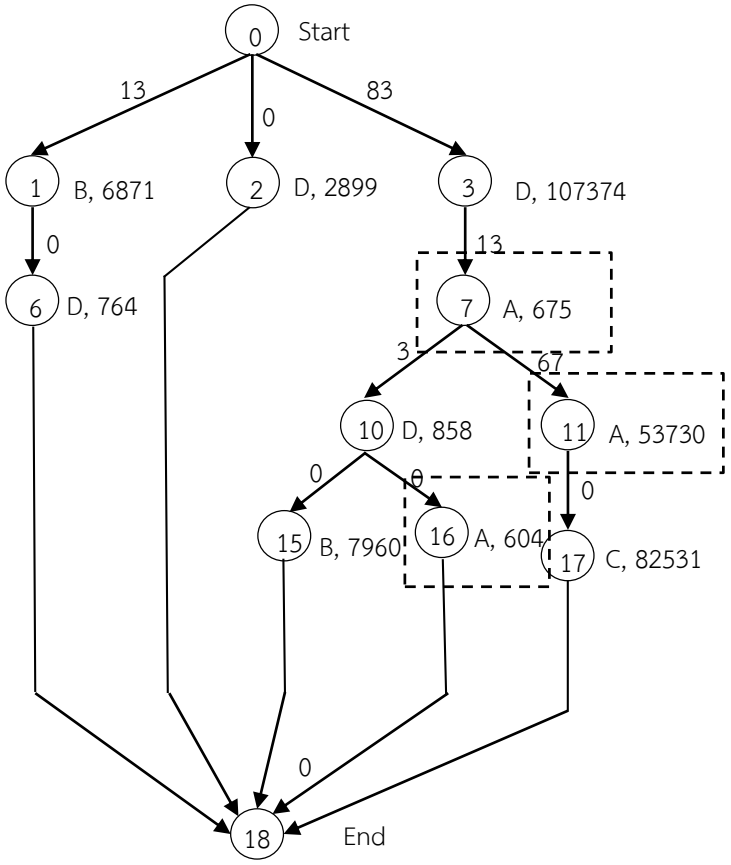
$$\text{Avg. \%delay} = \frac{\sum_{i=0}^{n} \%delay}{n} \qquad (2)$$

ผลลัพธ์ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของในแต่ละบทบาททั้ง 100 กระแสงาน แสดงในตารางที่ ข - 1 อยู่ในหน้า 78 พบว่าค่าที่สูงสุดของค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าในแต่ละบทบาทการทำงาน สำหรับบทบาท A เกิดขึ้นที่งาน Job147 บทบาท B เกิดขึ้นที่งาน Job391 บทบาท C เกิดขึ้นที่งาน Job380 บทบาท D เกิดขึ้นที่งาน Job144 และบทบาท E เกิดขึ้นที่งาน Job178 ดังตารางที่ 4-1

ตารางที่ 4-1 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าที่สูงสุดของแต่ละบทบาทในงานของกระแสงาน

Jobname	Role A	Role B	Role C	Role D	Role E
Job144	0.40	0.00	0.00	79.38	0.00
Job147	32.84	0.00	0.00	0.01	0.00
Job178	0.00	0.00	1.69	0.00	101.32
Job380	0.00	0.00	136.54	0.00	0.00
Job391	0.00	119.80	0.00	0.00	2.67

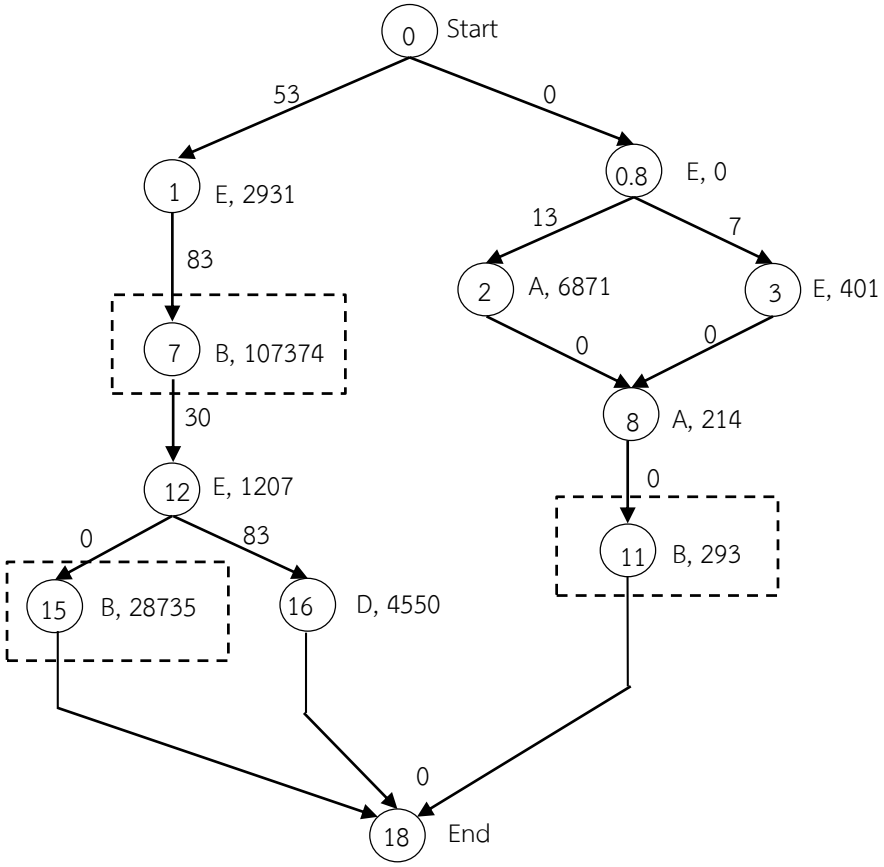
บทบาท A งานของกระแสงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคือ Job147 มีค่าเท่ากับ 32.84% ในภาพประกอบที่ 4-22 แสดงกรอบการทำงานของกระแสงาน Job147 เมื่อพิจารณาขั้นตอนการทำงานโหนดที่ต้องใช้คนในบทบาท A มาทำงานคือ โหนดที่ 7 11 และ 16 จากกราฟเส้นทางการทำงานในโหนดที่ 11 ใช้เวลา 53730 วินาที และทำงานคู่ขนานกับโหนดที่ 10 ซึ่งจะทำงานเสร็จก่อน แล้วทำให้ขั้นตอนถัดไปคือโหนดที่ 16 ใช้เวลา 604 วินาที แต่ต้องรอให้โหนดที่ 11 เสร็จเพราะมีบทบาท A เพียงคนเดียว เปอร์เซ็นต์ความล่าช้าในโหนดที่ 16 จึงสูงส่งผลให้ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของบทบาท A สูง ซึ่งเป็นค่าที่สูงสุดเมื่อเทียบกับงานของกระแสงานอื่นๆ



ภาพประกอบที่ 4-22 กรอบการทำงานของกระแสงาน Job147

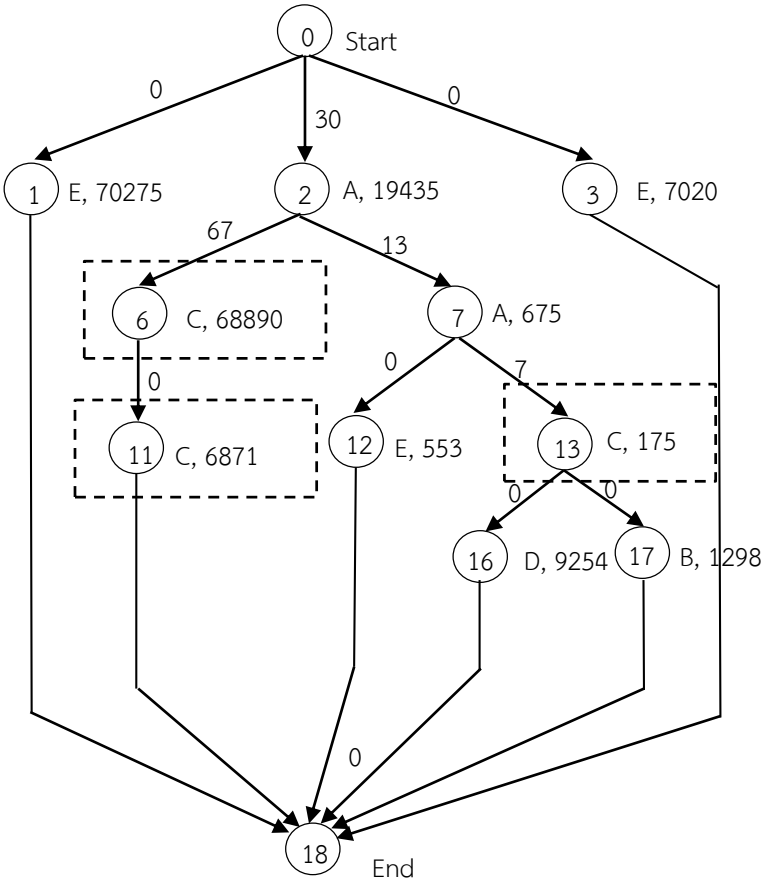
บทบาท B งานของกระแสงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคือ Job391 มีค่าเท่ากับ 119.80% ในภาพประกอบที่ 4-23 แสดงกรอบการทำงานของกระแสงาน Job391 เมื่อพิจารณาขั้นตอนการทำงานโหนดที่ต้องใช้คนในบทบาท B มาทำงาน คือโหนดที่ 7 11 และ 15 จากกราฟเส้นทางการทำงานในโหนดที่ 7 ที่ใช้เวลา 107374 วินาที ซึ่งใช้เวลานาน แต่เส้นทางของโหนดที่ 11 ที่ใช้เวลา 293 วินาที จะเสร็จเร็วกว่าทำให้ต้องรอให้โหนดที่ 7 เสร็จ เพราะมีบทบาท B เพียงคน

เดียว เปอร์เซ็นต์ความล่าช้าในโหนดที่ 11 จึงสูง ส่งผลให้ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของบทบาท B สูง ซึ่งเป็นค่าที่สูงสุดเมื่อเทียบกับงานของกระแสงานอื่นๆ



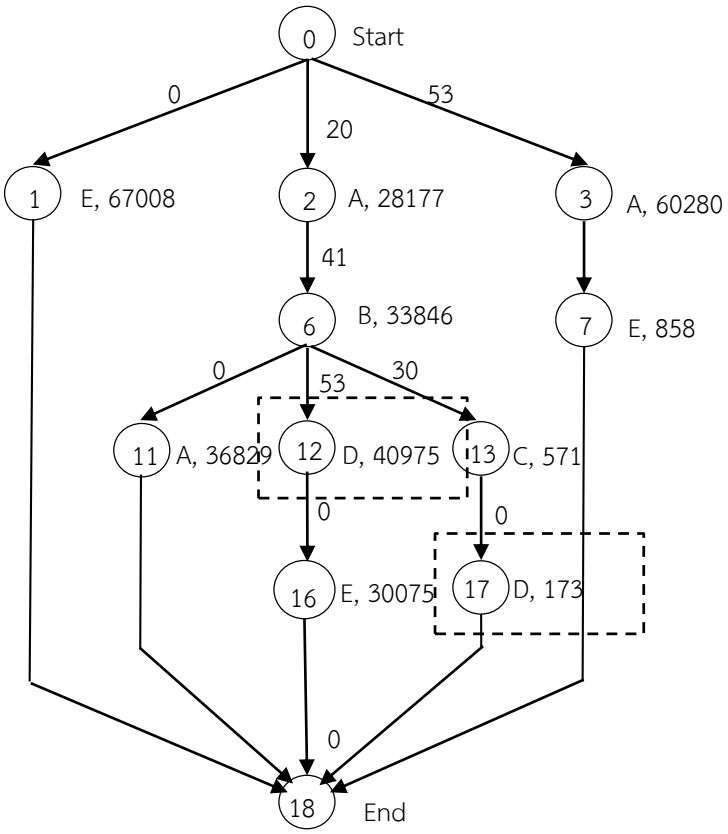
ภาพประกอบที่ 4-23 กรอบการทำงานของกระแสงาน Job391

บทบาท C งานของกระแสงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคือ Job380 มีค่าเท่ากับ 136.54% ในภาพประกอบที่ 4-24 แสดงกรอบการทำงานของกระแสงาน Job391 เมื่อพิจารณาขั้นตอนการทำงานโหนดที่ต้องใช้คนในบทบาท C มาทำงาน คือ โหนดที่ 6 11 และ 13 จากกราฟเส้นทางการทำงานในโหนดที่ 6 ที่ใช้เวลา 68890 วินาที ซึ่งใช้เวลานาน แต่เส้นทางของโหนดที่ 13 ที่ใช้เวลา 175 วินาที จะเสร็จเร็วกว่าทำให้ต้องรอให้โหนดที่ 6 เสร็จ เพราะมีบทบาท C เพียงคนเดียวทำให้มีเปอร์เซ็นต์ความล่าช้าสูง เมื่อคนในบทบาท C ทำงานโหนดที่ 13 ที่รออยู่แล้ว ทำให้โหนดที่ 11 ที่ใช้เวลา 6871 วินาที ซึ่งเป็นโหนดถัดจากโหนดที่ 6 ต้องรอ ทำให้มีเปอร์เซ็นต์ความล่าช้าสูงขึ้นอีก ส่งผลให้ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของบทบาท C สูง ซึ่งเป็นค่าที่สูงสุดเมื่อเทียบกับงานของกระแสงานอื่นๆ และสูงกว่าบทบาทอื่นๆ



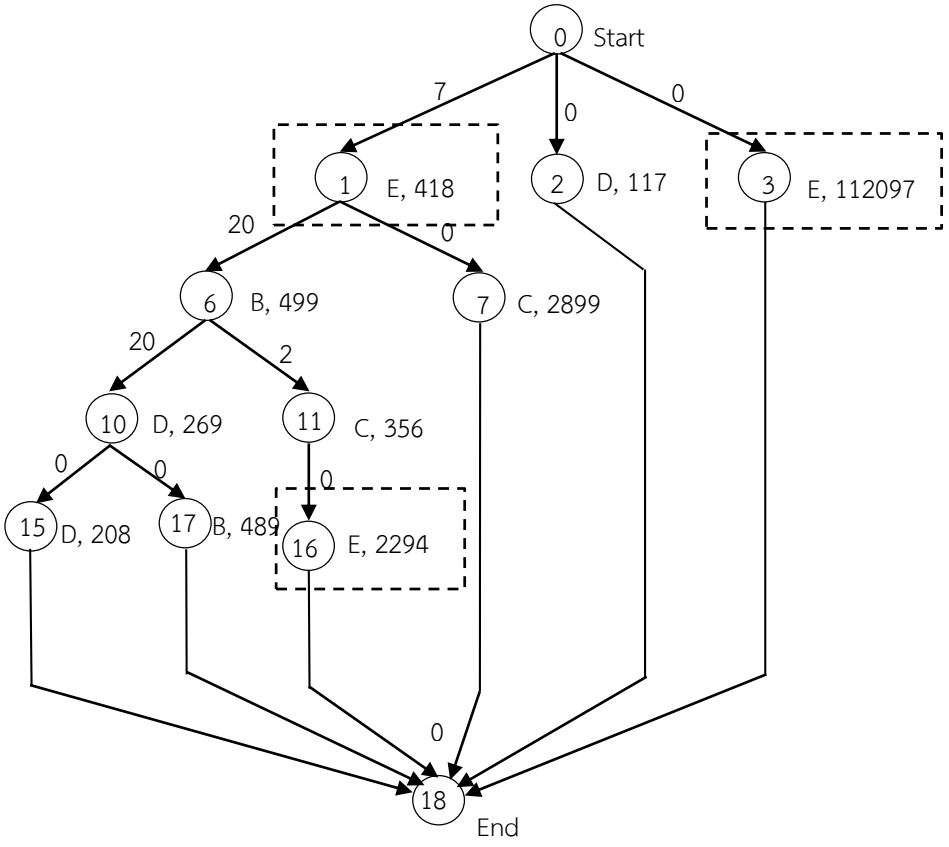
ภาพประกอบที่ 4-24 กรอบการทำงานของกระแสงาน Job380

บทบาท D งานของกระแสงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคือ Job144 มีค่าเท่ากับ 79.38% และภาพประกอบที่ 4-25 แสดงกรอบการทำงานของกระแสงาน Job144 เมื่อพิจารณาขั้นตอนการทำงานโหนดที่ต้องใช้คนในบทบาท D มาทำงาน คือโหนดที่ 12 และ 17 จากกราฟเส้นทางการทำงานในโหนดที่ 12 ที่ใช้เวลา 40975 วินาที ซึ่งใช้เวลานาน แต่เส้นทางของโหนดที่ 17 ที่ใช้เวลา 173 วินาที จะเสร็จเร็วกว่าทำให้ต้องรอให้โหนดที่ 12 เพราะมีบทบาท D เพียงคนเดียว เปอร์เซ็นต์ความล่าช้าในโหนดที่ 17 จึงสูง ส่งผลให้ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของบทบาท D สูง ซึ่งเป็นค่าที่สูงสุดเมื่อเทียบกับงานของกระแสงานอื่นๆ



ภาพประกอบที่ 4-25 กรอบการทำงานของกระแสงาน Job144

บทบาท E งานของกระแสงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคือ Job178 มีค่าเท่ากับ 101.32% และภาพประกอบที่ 4-26 แสดงกรอบการทำงานของกระแสงาน Job178 เมื่อพิจารณาขั้นตอนการทำงานโหนดที่ต้องใช้คนในบทบาท E มาทำงาน คือโหนดที่ 1 3 และ 16 จากกราฟเส้นทางการทำงานของโหนดที่ 3 ที่ใช้เวลา 112097 วินาที จะได้ทำงานก่อนโหนดที่ 1 ที่ใช้เวลาทำงาน 418 วินาที เพราะโหนดที่ 1 เสียเวลาในการส่งงาน จึงทำให้ต้องรอ เพราะมีบทบาท E เพียงคนเดียว เปอร์เซ็นต์ความล่าช้าในโหนดที่ 1 จึงสูง ส่งผลให้ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของบทบาท E สูง ซึ่งเป็นค่าที่สูงสุดเมื่อเทียบกับงานของกระแสงานอื่นๆ



ภาพประกอบที่ 4-26 กรอบการทำงานของกระแสงาน Job178

จากการวิเคราะห์ที่กล่าวมานี้ พบว่าปัญหาทั้งหมดเกิดจากการรอเนื่องจากมีคนในบทบาทนั้นเพียงคนเดียว แสดงว่าการทำงานในกระแสงานนั้นมีจำนวนคนไม่เพียงพอ จึงเกิดเวลาความล่าช้าที่มีเปอร์เซ็นต์สูง เพื่อที่จะลดเวลาความล่าช้าลง จึงทดลองเพิ่มคนสำหรับทำงานเป็น 2 คน ในแต่ละบทบาทที่มีเปอร์เซ็นต์สูงสุด และประมวลผลกระแสงานในตารางที่ 4-1 ซ้ำอีกครั้ง ดังผลลัพธ์การประมวลผลในข้อ 4.4.2

4.4.2 การประมวลผลกระแสงานเมื่อเพิ่มบุคคลในแต่ละบทบาท

การทดสอบแก้ปัญหากระแสงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุด โดยเพิ่มบุคคลเฉพาะในบทบาทที่เกิดความล่าช้าสูงสุดจาก 1 คนเป็น 2 แล้วประมวลผลกระแสงานนั้นใหม่ ได้ผลลัพธ์ดังในตารางที่ 4-2 พบว่าการเพิ่มจำนวนคนส่งผลให้ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าหมดไป หมายความว่า มีคนในบทบาทนั้นเพียงพอแล้ว

ตารางที่ 4-2 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าเมื่อเพิ่มบุคคลในแต่ละบทบาท

Jobname	Role A	Role B	Role C	Role D	Role E
Job144	0.38	0.00	0.00	0.00	0.00
Job147	0.00	0.00	0.00	0.01	0.00
Job178	0.00	0.00	1.69	0.00	0.00
Job380	0.00	0.00	0.00	0.00	0.00
Job391	0.00	0.00	0.00	0.00	0.06

4.4.3 การประมวลกระแสงาน 100 กระแสงานเมื่อเพิ่มบุคคลทำงานในแต่ละบทบาท

หลังจากที่ได้เพิ่มบุคคลเฉพาะในบทบาทที่เกิดความล่าช้าสูงสุดจาก 1 คนเป็น 2 และประมวลผลเฉพาะงานที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุด ได้ผลลัพธ์ที่แก้ปัญหาการรอได้ ที่ได้แสดงให้เห็นไปแล้วในหัวข้อที่ 4.4.2 จึงได้ทดสอบประมวลผลกระแสงานทั้ง 100 กระแสงานซ้ำอีก 5 ครั้งตามจำนวนบทบาท โดยเพิ่มคนในบทบาทนั้นๆจาก 1 คนเป็น 2 คน ผลลัพธ์ที่ได้แสดงดังตารางที่ 4-3 ถึงตารางที่ 4-7 พบว่าแนวโน้มของค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของแต่ละบทบาทลดลง ดังรายละเอียดต่อไปนี้

4.4.3.1 เพิ่มบุคคลทำงานในบทบาท A

เมื่อเพิ่มบุคคลทำงานในบทบาท A เป็น 2 คน โดยบทบาทงานอื่นยังคง 1 คนเหมือนเดิม และประมวลผลกระแสงานทั้งหมด ได้ผลลัพธ์ดังแสดงในตารางที่ 4-3 ซึ่งจากเดิมบทบาท A มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคืองาน Job147 (ในตารางที่ 4-1) มีค่า 32.84% หลังจากเพิ่มบุคคลทำงานเข้าไปทำให้ค่าลดลงเป็น 0% และทำให้ค่าสูงสุดในบทบาท A เปลี่ยนเป็น Job029 ซึ่งมีค่าน้อยมากคือ 0.07% แสดงว่าจำนวนคนในบทบาท A นั้น เพียงพอต่อการทำงานแล้วในทุกกระแสงาน แต่สำหรับบทบาทอื่นยังคงไม่เพียงพอ เพราะค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้ายังสูงอยู่ แม้บางบทบาทจะมีค่าลดลงก็ตาม แต่เมื่อวิเคราะห์แล้วการลดลงนี้ไม่เกี่ยวข้องกับการเพิ่มคนในบทบาท A เพราะเป็นงานคนละหน้าที่ไม่ได้เกิดจากการรอบทบาท A แต่เกิดจากการที่มีขั้นตอนการทำงานบทบาทเดียวกัน คู่ขนานกัน และค่าเวลาที่ใช้ส่งงานใกล้เคียงกันมาก มีความเป็นไปได้ว่าระบบสามารถเลือกงานใดก็มาประมวลผล หากเลือกโหนดที่ใช้เวลาทำงานน้อยกว่ามาประมวลผลก่อน ค่าเปอร์เซ็นต์ความล่าช้าก็จะลดลง เช่น Job144 Job370 และ Job391 ในทางกลับกันหากเลือกโหนดที่ใช้เวลาทำงานมากกว่ามาประมวลผลก่อน ค่าเปอร์เซ็นต์ความล่าช้าก็จะเพิ่มขึ้น เช่น Job319

ตารางที่ 4-3 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดของกระแสงานเมื่อเพิ่มบุคคลในบทบาท A

Jobname	Role A	Role B	Role C	Role D	Role E
Job029	0.07	0.00	1.38	0.00	0.00
Job144	0.00	0.00	0.00	54.24	1.52
Job319	0.00	0.00	291.04	0.00	0.00
Job370	0.00	0.00	0.00	0.00	24.99
Job391	0.00	80.49	0.00	0.00	1.40
งานเดิมที่ล่าช้าสูงสุดและเพิ่มบุคคลในบทบาทงาน A					
Job147	0.00	0.00	0.00	0.00	0.00
Job178	0.00	0.00	1.08	0.00	0.00
Job380	0.00	0.00	88.21	0.00	0.00

4.4.3.2 เพิ่มบุคคลทำงานในบทบาท B

เมื่อเพิ่มบุคคลทำงานในบทบาท B เป็น 2 คนทำงาน โดยบทบาทงานอื่นยังคง 1 คน เหมือนเดิม และประมวลผลกระแสงานทั้งหมด ได้ผลลัพธ์ดังแสดงตารางที่ 4-4 ซึ่งจากเดิมบทบาท B มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคืองาน Job391 (ในตารางที่ 4 1) มีค่า 119.80% หลังจากเพิ่มบุคคลทำงานเข้าไปทำให้ค่าลดลงเป็น 0% และทำให้ค่าสูงสุดในบทบาท B เปลี่ยนเป็น Job049 ซึ่งมีค่าน้อยมากคือ 0.01% แสดงว่าจำนวนคนในบทบาท B นั้น เพียงพอต่อการทำงานแล้วในทุกกระแสงานแต่สำหรับบทบาทอื่นยังคงไม่เพียงพอ เพราะค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้ายังสูงอยู่ แม้บางบทบาทจะมีค่าลดลงก็ตาม แต่เมื่อวิเคราะห์แล้วการลดลงนี้ไม่เกี่ยวข้องกับการเพิ่มคนในบทบาท B เพราะเป็นงานคนละหน้าที่ไม่ได้เกิดจากการรอบทบาท B แต่เกิดจากการที่มีขั้นตอนการทำงานบทบาทเดียวกัน คู่ขนานกัน และค่าเวลาที่ใช้ส่งงานใกล้เคียงกันมาก มีความเป็นไปได้ว่าระบบสามารถเลือกงานใดก็มาประมวลผล หากเลือกโหนดที่ใช้เวลาทำงานน้อยกว่ามาประมวลผลก่อน ค่าเปอร์เซ็นต์ความล่าช้าก็จะลดลง เช่น Job144 Job147 Job159 และ Job380

ตารางที่ 4-4 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดของกระแสงานเมื่อเพิ่มบุคคลในบทบาท B

Jobname	Role A	Role B	Role C	Role D	Role E
Job049	0.02	0.01	0.00	0.00	0.00
Job144	0.32	0.00	0.00	63.01	0.00
Job147	25.61	0.00	0.00	0.01	0.00
Job159	0.00	0.00	0.00	0.00	50.88
Job380	0.00	0.00	107.93	0.00	0.00
งานเดิมที่ล่าช้าสูงสุดและเพิ่มบุคคลในบทบาทงาน B					
Job178	0.00	0.00	1.40	0.00	0.00
Job391	0.00	0.00	0.00	0.00	1.65

4.4.3.3 เพิ่มบุคคลทำงานในบทบาท C

เมื่อเพิ่มบุคคลทำงานในบทบาท C เป็น 2 คนทำงาน โดยบทบาทงานอื่นยังคง 1 คน เหมือนเดิม และประมวลผลกระแสงานทั้งหมด ได้ผลลัพธ์ดังแสดงในตารางที่ 4-5 ซึ่งจากเดิมบทบาท C ที่มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคืองาน Job380 (ดูในตารางที่ 4-1) มีค่า 136.54% หลังจากเพิ่มบุคคลทำงานเข้าไปทำให้ค่าลดลงเป็น 0% และทำให้ค่าสูงสุดในบทบาท C เปลี่ยนเป็น Job070 ซึ่งมีค่าน้อยมากคือ 0.83% แสดงว่าจำนวนคนในบทบาท C นั้น เพียงพอต่อการทำงานแล้วในทุกกระแสงาน แต่สำหรับบทบาทอื่นยังคงไม่เพียงพอ เพราะค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้ายังสูงอยู่ แม้บางบทบาทจะมีค่าลดลงก็ตาม แต่เมื่อวิเคราะห์แล้วการลดลงนี้ไม่เกี่ยวข้องกับการเพิ่มคนในบทบาท C เพราะเป็นงานคนละหน้าที่ไม่ได้เกิดจากการรอบทบาท C แต่เกิดจากการที่มีขั้นตอนการทำงานบทบาทเดียวกัน คู่ขนานกัน และค่าเวลาที่ใช้ส่งงานใกล้เคียงกันมาก มีความเป็นไปได้ว่าระบบสามารถเลือกงานใดก็มาประมวลผล หากเลือกโหนดที่ใช้เวลาทำงานน้อยกว่ามาประมวลผลก่อน ค่าเปอร์เซ็นต์ความล่าช้าก็จะลดลง เช่น Job144 Job147 Job206 และ Job391

ตารางที่ 4-5 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดของกระแสงานเมื่อเพิ่มบุคคลในบทบาท C

Jobname	Role A	Role B	Role C	Role D	Role E
Job070	0.00	0.00	0.83	0.00	0.00
Job144	0.35	0.00	0.00	69.08	0.00
Job147	27.62	0.00	0.00	0.01	0.00
Job206	0.00	0.00	0.00	0.15	33.71
Job391	0.00	109.73	0.00	0.00	1.87
งานเดิมที่ล่าช้าสูงสุดและเพิ่มบุคคลในบทบาทงาน C					
Job178	0.00	0.00	0.00	0.00	0.00
Job380	0.00	0.00	0.00	0.00	0.00

4.4.3.4 เพิ่มบุคคลทำงานในบทบาท D

เมื่อเพิ่มบุคคลทำงานในบทบาท D เป็น 2 คนทำงาน โดยบทบาทงานอื่นยังคง 1 คน เหมือนเดิม และประมวลผลกระแสงานทั้งหมด ได้ผลลัพธ์ดังแสดงในตารางที่ 4-6 ซึ่งจากเดิมบทบาทงาน D มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคืองาน Job144 (ดูในตารางที่ 4-1) คือมีค่า 79.38% หลังจากเพิ่มบุคคลทำงานเข้าไปทำให้ค่าลดลงเป็น 0% และทำให้ค่าสูงสุดในบทบาท D เปลี่ยนเป็น Job187 ซึ่งมีค่าน้อยมากคือ 1.57% แสดงว่าจำนวนคนในบทบาท D นั้น เพียงพอต่อการทำงานแล้วในทุกกระแสงาน แต่สำหรับบทบาทอื่นยังคงไม่เพียงพอ เพราะค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้ายังสูงอยู่ แม้บางบทบาทจะมีค่าลดลงก็ตาม แต่เมื่อวิเคราะห์แล้วการลดลงนี้ไม่เกี่ยวข้องกับการเพิ่มคนในบทบาท D เพราะเป็นงานคนละหน้าที่ไม่ได้เกิดจากการรอบทบาท D แต่เกิดจากการที่มีขั้นตอนการทำงานบทบาทเดียวกัน คู่ขนานกัน และค่าเวลาที่ใช้ส่งงานใกล้เคียงกันมาก มีความเป็นไปได้ว่าระบบสามารถเลือกงานใดก็มาประมวลผล หากเลือกโหนดที่ใช้เวลาทำงานน้อยกว่ามาประมวลผลก่อน ค่าเปอร์เซ็นต์ความล่าช้าก็จะลดลง เช่น Job147 Job206 Job380 และ Job391

ตารางที่ 4-6 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดของกระแสงานเมื่อเพิ่มบุคคลในบทบาท D

Jobname	Role A	Role B	Role C	Role D	Role E
Job147	22.08	0.00	0.00	0.00	0.00
Job187	0.00	0.00	0.00	1.57	17.53
Job206	0.00	0.00	0.00	0.00	26.43
Job380	0.00	0.00	92.11	0.00	0.00
Job391	0.00	86.06	0.00	0.00	1.50
งานเดิมที่ล่าช้าสูงสุดและเพิ่มบุคคลในบทบาทงาน D					
Job144	0.29	0.00	0.00	0.00	0.00
Job178	0.00	0.00	1.17	0.00	0.00

4.4.3.5 บุคคลทำงานในบทบาท E

เมื่อเพิ่มบุคคลทำงานในบทบาท E เป็น 2 คนทำงาน โดยบทบาทงานอื่นยังคง 1 คน เหมือนเดิม และประมวลผลกระแสงานทั้งหมด ได้ผลลัพธ์ดังแสดงในตารางที่ 4-7 จากเดิมบทบาทงาน E มีค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดคืองาน Job178 (ดูในตารางที่ 4-1) มีค่า 101.32% หลังจากเพิ่มบุคคลทำงานเข้าไปทำให้ค่าลดลงเป็น 0% และทำให้ค่าสูงสุดในบทบาท E เปลี่ยนเป็น Job251 ซึ่งมีค่าน้อยมากคือ 0.01% แสดงว่าจำนวนคนในบทบาท E นั้น เพียงพอต่อการทำงานแล้วในทุกกระแสงาน แต่สำหรับบทบาทอื่นยังคงไม่เพียงพอ เพราะค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้ายังสูงอยู่ แม้บางบทบาทจะมีค่าลดลงก็ตาม แต่เมื่อวิเคราะห์แล้วการลดลงนี้ไม่เกี่ยวข้องกับการเพิ่มคนในบทบาท E เพราะเป็นงานคนละหน้าที่ไม่ได้เกิดจากการรอบทบาท E แต่เกิดจากการที่มีขั้นตอนการทำงานบทบาทเดียวกัน คู่ขนานกัน และค่าเวลาที่ใช้ส่งงานใกล้เคียงกันมาก มีความเป็นไปได้ว่าระบบสามารถเลือกงานใดก็มาประมวลผล หากเลือกโหนดที่ใช้เวลาทำงานน้อยกว่ามาประมวลผลก่อน ค่าเปอร์เซ็นต์ความล่าช้าก็จะลดลง เช่น Job144 Job147 Job380 และ Job391

ตารางที่ 4-7 ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าสูงสุดของกระแสงานเมื่อเพิ่มบุคคลในบทบาท E

Jobname	Role A	Role B	Role C	Role D	Role E
Job144	0.28	0.00	0.00	56.26	0.00
Job147	23.32	0.00	0.00	0.00	0.00
Job251	0.00	0.00	5.10	0.00	0.01
Job380	0.00	0.00	99.64	0.00	0.00
Job391	0.00	89.76	0.00	0.00	0.00
งานเดิมที่ล่าช้าสูงสุดและเพิ่มบุคคลในบทบาทงาน E					
Job178	0.00	0.00	1.31	0.00	0.00

ผลลัพธ์การทดสอบการประมวลผลทั้ง 100 กระแสงาน เมื่อเพิ่มบุคคลทำงานในแต่ละบทบาทเข้าไปเป็น 2 คนทำงาน พบว่าผลลัพธ์ค่าเฉลี่ยเปอร์เซ็นต์ความล่าช้าของบทบาทในกระแสงานทั้งหมดโดยรวมมีแนวโน้มลดลง คือเวลาที่รอคนสำหรับทำงานน้อยลง ซึ่งสรุปได้ว่าจำนวนคนที่เพิ่มไปในแต่ละบทบาทเป็น 2 คน นั้นเพียงพอต่อการทำงานในกระแสงานที่ใช้ในการทดสอบการจำลองนี้ ซึ่งจากข้อมูลการประมวลผลดังกล่าวสามารถนำไปใช้เป็นแนวทางเพื่อวิเคราะห์กำลังคนขององค์กรต่อไปได้

บทที่ 5
สรุปผลการวิจัยและข้อเสนอแนะ

ในบทนี้จะกล่าวถึง สรุปผลการวิจัยของระบบจำลองและการประมวลผลกระแสงานที่ได้จัดทำขึ้น และข้อเสนอแนะในงานวิจัย ดังต่อไปนี้

5.1 สรุปผลการวิจัย

งานวิจัยนี้ ได้ออกแบบและพัฒนาตัวจำลองการทำงานในสำนักงาน เพื่อเป็นเครื่องมือสำหรับใช้วิเคราะห์ขั้นตอนการทำงานในสำนักงานใดๆ โดยออกแบบและพัฒนาบนระบบจัดการกระแสงานอูซซี ซึ่งทำงานอยู่บนการประมวลผลแบบกระจาย Hadoop ทำให้รองรับการขยายตัวของระบบงานหรือขนาดขององค์กรที่เติบโตในอนาคตได้ ระบบนี้มีความสามารถที่ได้ออกแบบและพัฒนาคือ มีเซอร์วิสสำหรับจัดการกับข้อมูลของระบบประกอบด้วย User Service, Role Service, Workflow Service, WorkflowJob Service, Queue Service และ Log Service มีฟังก์ชันเพื่อใช้ควบคุมตัวจำลองการทำงานคือ ฟังก์ชันเริ่ม หยุด หยุดชั่วคราว และสถานะการทำงาน นอกจากนี้ยังออกแบบและพัฒนาโหนดการกระทำที่ใช้สำหรับประมวลผลขั้นตอนการทำงานเรียกว่า AsyncAction เพื่อใช้จำลองการทำงานของบุคคลตามบทบาทหน้าที่

จากผลลัพธ์การทดสอบประมวลผล 100 กระแสงาน ที่มี 5 บทบาทการทำงาน พบความล่าช้าเกิดขึ้นในบทบาทการทำงานเนื่องจากจำนวนคนไม่เพียงพอจึงเกิดการรอการทำงาน ซึ่งแก้ปัญหาได้โดยการเพิ่มบุคลากรในบทบาทนั้นจนไม่เกิดความล่าช้า สาเหตุของความล่าช้าของกระแสงานอื่นๆ ส่วนใหญ่เกิดจากมีงานในบทบาทเดียวกัน ทำคู่ขนานกัน แล้วเลือกทำงานที่ใช้เวลานานก่อน งานที่เสร็จเร็วกว่าจึงต้องรอ ซึ่งเป็นทางเลือกที่ไม่มีประสิทธิภาพ หากวิจัยเพิ่มการตัดสินใจลำดับการทำงาน ณ จุดนี้ ก็จะช่วยเพิ่มประสิทธิภาพได้

ทั้งหมดนี้แสดงให้เห็นว่าตัวจำลองที่พัฒนาขึ้น สามารถใช้เป็นเครื่องมือในการพิจารณาและวิเคราะห์ระบบกระแสงานได้ แต่ในสภาพแวดล้อมการทำงานจริง ยังมีอีกหลายปัจจัยที่มีผลกับคุณภาพและประสิทธิภาพการทำงาน เช่น ในแง่ของเวลา มีเวลาที่คนใช้ทำงาน ขึ้นอยู่กับประสบการณ์ ความสามารถ อารมณ์ หรือเวลาในการส่งงานขึ้นอยู่กับช่องทางการสื่อสารที่เลือกใช้ ในแง่ของค่าใช้จ่าย มีค่าใช้จ่ายของค่าจ้างบุคลากร ค่าเครื่องมือในการทำงาน ค่าสาธารณูปโภค เป็นต้น ซึ่งงานวิจัยนี้ยังไม่สามารถครอบคลุมปัจจัยการทำงานทั้งหมดได้ เพราะมีความซับซ้อนสูงและนอกเหนือจากขอบเขตงานวิจัยนี้ แต่หากต้องการดัดแปลง ระบบได้ออกแบบให้รองรับการเก็บข้อมูล

ปัจจัยใดๆ ก็ได้ไว้แล้ว มีเพียงการประมวลผลภายใน AsyncAction เท่านั้น ที่ต้องเอาปัจจัยอื่นๆ มาใช้เป็นเงื่อนไขการจำลองเพิ่ม

5.2 ข้อเสนอแนะ

5.2.1 หากต้องการจำลององค์กรที่มีขนาดใหญ่ขึ้น ก็สามารถเพิ่มจำนวนครื่องประมวลผลในระบบการจายของ Hadoop ได้โดยไม่ต้องปรับปรุงโปรแกรม

5.2.2 หากต้องการจำลองปัจจัยอื่นๆ ที่มีผลกระทบต่อประสิทธิภาพของงาน สามารถทำได้โดยปรับปรุงส่วนของเงื่อนไขใน AsyncAction ได้

5.2.3 สำหรับประเด็นที่น่าสนใจวิจัยต่อในอนาคตคือ การเพิ่มการวิเคราะห์เพื่อตัดสินใจเลือกทำงานใดก่อนหลังให้ได้ประสิทธิภาพสูงสุด หรือการจำลองให้ 1 คนทำงานได้หลายบทบาท

บรรณานุกรม

Adela Bara, Ion Lungu, Simona Vasilica Oprea, George Carutasu, Cornelia Paulina Botezatu and et al. "*Design Workflow for Cloud Service Information System for Integration and Knowledge Management based in Renewable Energy,"* Journal of Information Systems & Operations Management; Bucharest, Vol.6, No.1, 2014.

Anand Loganathan, Ankur Sinha, Muthuramakrishnan V., and Srikanth Natarajan, *"A Systematic Approach to Big Data Exploration of the Hadoop Framework,"* International Journal of Information & Computation Technology, Vol.4, No.9, pp.869-878, 2014.

Anju Bala and Inderveer Chana, *"Design and Deployment of Workflows in Cloud Environment,"* International Journal of Computer Applications, Vol.51, No.11, pp. 9–15, Aug. 2012.

Anne Rozinat, Moe Thandar Wynn, Wil MP van der Aalst, Arthur HM ter Hofstede and Colin J Fidge *"Workflow simulation for operational decision support,"* Sixth International Conference on Business Process Management (BPM 2008), Vol.68, Issue 9, pp. 834-850, September 2009.

DAGGEN: A synthetic task graph generator, 2017 https://github.com/frs69wq/daggen.

David Hollingsworth, *"Workflow Management Coalition The Workflow Reference Model,"* The Workflow Management Coalition, 1995.

Dorin Carstoiu, Elena Lepadatu, Mihai Gaspar, *"Hbase - non SQL Database, Performances Evaluation,"* International Journal of Advancements in Computing Technology, Vol.2, No.5, pp. 42-52, 2010.

Edward A.Stohr and J. Leon Zhao, "*Workflow Automation: Overview and Research Issues,*" Vol. 3, Issue.3, pp. 281-296, 2001.

Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J. Maechling, RajivMayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny and Kent Wenger, "*Pegasus, a workflow management system for science automation,*" Future Generation Computer Systems, Vol.46, pp. 17-35, May 2015.

G. Alonso, M. Kamath, Dl Agrawal, and etc. "*Failure handling in large-scale workflow management systems,"* IBM Research Report RJ9913, November 1994.

Jan Mendling, *"Business Process Execution Language for Web Services (BPEL)*", Aktuelles Schlagwort, EMISA Forum, vol. 26, no 2, pp. 5-8, August 2006.

Jeff Dean and Sanjay Ghemawat, *"MapReduce: Simplified Data Processing on Large Clusters,"* Communications of the ACM - 50th anniversary issue: 1958 - 2008, Vol.51, No 1, pp. 107-113, Jan. 2008.

Ji Liu, Esther Pacitti, Patrick Valduriez and Marta Mattoso *"A Survey of Data-Intensive Scientific Workflow Management,"* Journal of Grid Computing, Vol.13, Issue 4, pp. 457–493, December 2015.

Kwang-Hoon Kim, Hyung-Jin Ahn, and Chang-Min Kim, "*Performance estimations of clustered workflow architectures,"* Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05), pp. 288–293.

Li Liu, Miao Zhang, Yuqing Lin and Liangjuan Qin *"A Survey on Workflow Management and Scheduling in Cloud Computing*," 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 26-29 May 2014.

Mehul Nalin Vora, *"Hadoop-HBase for Large-Scale Data,"* International Conference on Computer Science and Network Technology, 2011.

Mohammad Islam, Angelo K. Huang, Mohamed Battisha, Michelle Chiang, Santhosh Srinivasan, Craig Peters, Andreas Neumann, *"Oozie: Towards a Scalable Workflow Management System for Hadoop,"* 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies Article, No. 4, 2012.

Nitiwat Thongkao and Somchai Limsiroratana, *"Design of cloud-based university data structure,"* 2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Bangkok, pp. 186-191, 6-8 May 2014.

Ricardo Anderson and Gunjan Mansingh "*A Workflow Management System for Knowledge Management Initiatives,"* Twenty-third Americas Conference on Information Systems, Boston, 2017.

Seema Maitreya and C.K. Jhab, *"MapReduce: Simplified Data Analysis of Big Data,"* 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015), Vol.57, pp. 563-571, 2015.

Shi Meilin, Yang Guangxin, Xiang Yong, and Wu Shangguang, "*Workflow management systems: a survey*," Communication Technology Proceedings, 1998. ICCT '98. 1998 International Conference, Vol.2, p. 6, 22-24 Oct. 1998.

Sergey V. Ivanov, Sergey V. Kovalchuk and Alexander V. Boukhanovsky "*Workflow-based Collaborative Decision Support for Flood Management Systems,"* International Conference on Computational Science, Vol.18, pp. 2213-2222, 2013.

The Apache Software Foundation 2017, "Hadoop" http://hadoop.apache.org/.

The Apache Software Foundation 2017, "Oozie" http://oozie.apache.org/.

Thomas Seidl 2009, "mapreduce" http://dme.rwth-aachen.de/en/research/projects/mapreduce/.

Wanjun Huang, Xinhua Zhang, U. Roth, and C. Meinel, *"Routing based workflow for construction of distributed applications,"* Vol. 1, pp. 80–85, 2004.

Weiwei Chen and Ewa Deelman, *"WorkflowSim: A toolkit for simulating scientific workflows in distributed environments,"* IEEE 8th International Conference on E-Science, 8-12 Oct. 2012.

Workflow Management Coalition 2017, "*WfMC,"* http://www.wfmc.org/.

Wil van der Aalst, *"The Application of Petri Nets to Workflow Management,"* Journal of Circuits, Systems, and Computers. 1998 Vol. 8, pp. 21-66.

Xiu Li, Jingdong Song and Biqing Huang *"A scientific workflow management system architecture and its scheduling based on cloud service platform for manufacturing big data analytics,"* The International Journal of Advanced Manufacturing Technology, Vol. 84, Issue 1–4, pp. 119–131, April 2016.

Yang Ruan, Zhenhua Guo, Yuduo Zhou, Judy Qiu, Geoffrey Fox *"HyMR: a Hybrid MapReduce Workflow System"* The 3rd International Emerging Computational Methods for the Life Sciences Workshop (ECMLS'12), 2012.

Ying Liu, Hui Zhang, Chunping Li and Roger Jianxin Jiao, "Workflow simulation for operational decision support using event graph through process mining," Decision Support Systems, Vol.52, Issue 3, pp. 685-697, February 2012.

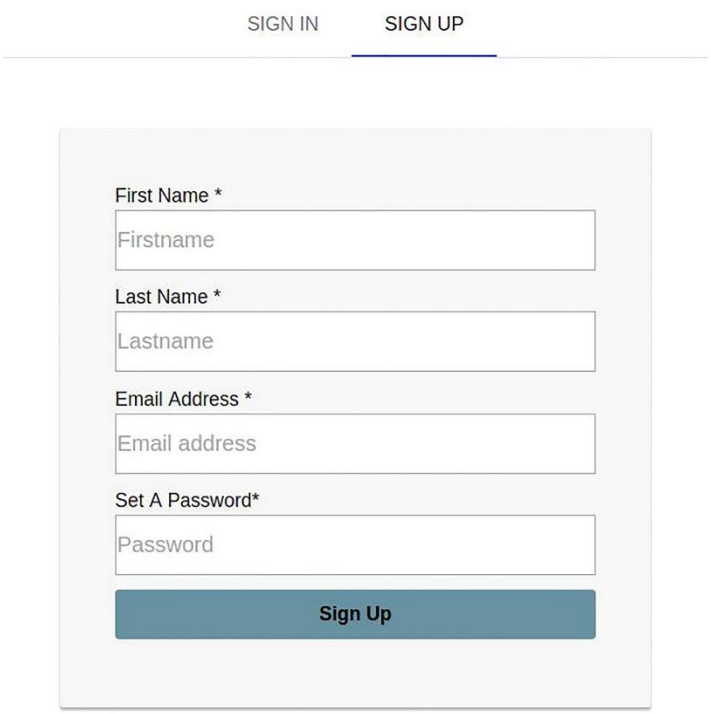
ภาคผนวก

ภาคผนวก ก
คู่มือการใช้งานเว็บส่วนติดต่อของผู้ใช้เพื่อเตรียมข้อมูล

การใช้งานเว็บส่วนติดต่อการใช้งานของผู้ใช้งานหรือผู้ดูแลระบบ เริ่มต้นจะมีการลงทะเบียนผู้ใช้ เพื่อสร้างโปรเจคสำหรับการจำลองประมวลผลกระแสงานอัตโนมัติ เตรียมข้อมูลบุคลากร บทบาทหน้าที่การทำงาน เตรียมไฟล์กระแสงาน เตรียมคิวงานและเพิ่มกระแสงานในคิวนั้นๆ หลังจากนั้นเริ่มระบบจำลองประมวลผลกระแสงาน และแสดงผลลัพธ์การประมวลผลงานของกระแสงาน

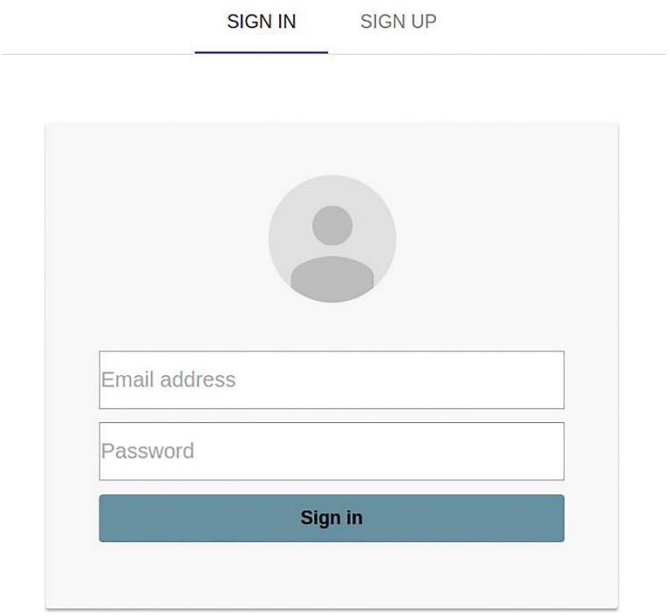
1. ลงทะเบียนผู้ใช้งาน

การลงทะเบียน URL http://{hostname}/OozieWebServicesAPI/login.html กรอกข้อมูลพารามิเตอร์สำหรับการลงทะเบียน ดังภาพประกอบที่ ก - 1



ภาพประกอบที่ ก - 1 การลงทะเบียนผู้ใช้งาน

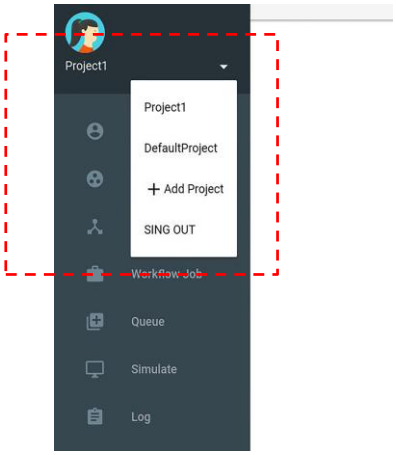
เมื่อลงทะเบียนเสร็จแล้ว จะสามารถลงชื่อเข้าใช้งานในภาพประกอบที่ ก - 2



ภาพประกอบที่ ก - 2 การลงชื่อเข้าใช้งาน

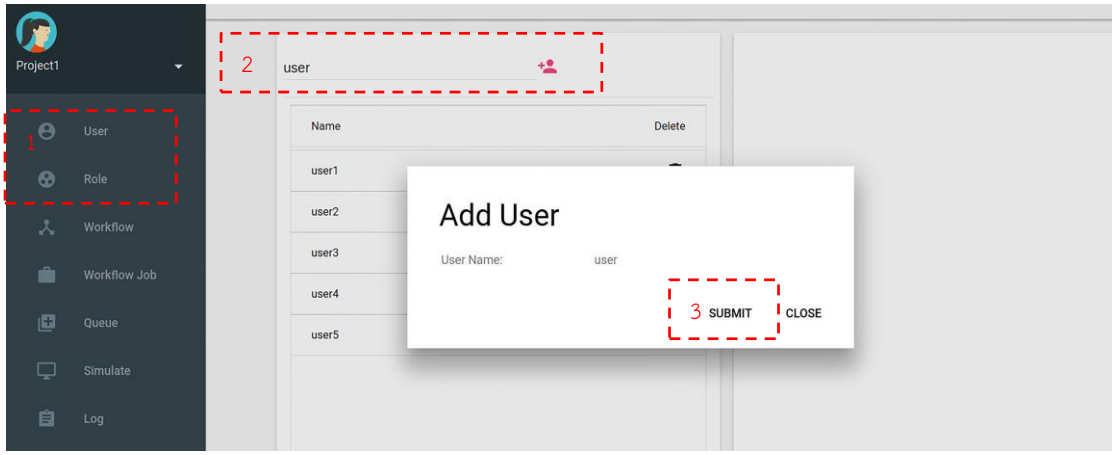
2. การเตรียมข้อมูลการจำลอง บุคลากรและบทบาทหน้าที่

ระบบการจำลองกระแสงานต้องสร้างโปรเจคเพื่อใช้ในการเตรียมข้อมูล ผ่านเซอร์วิสในการเตรียมข้อมูล User, Role ดังภาพประกอบที่ ก - 3



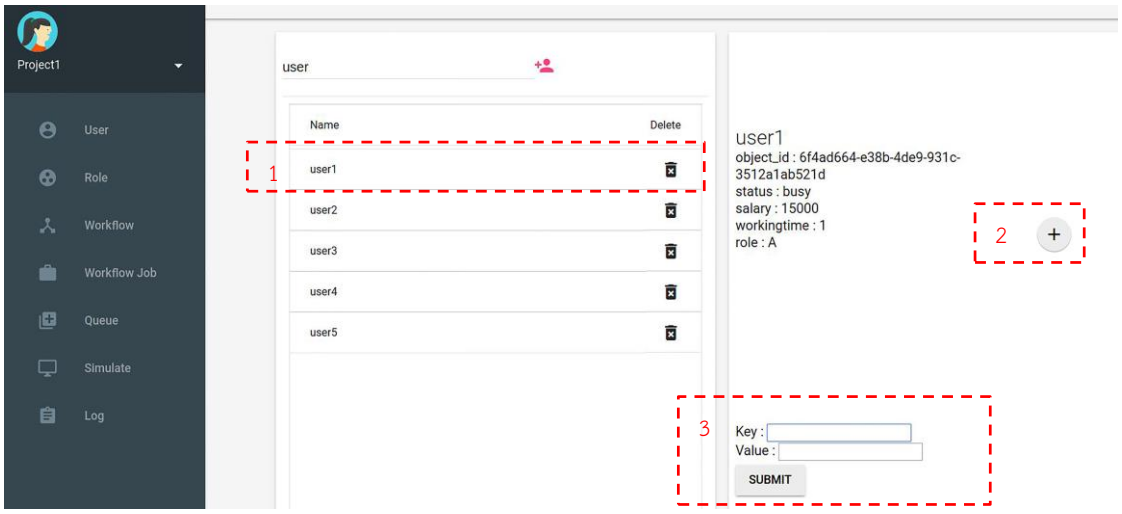
ภาพประกอบที่ ก - 3 สร้างโปรเจคสำหรับการจำลอง

เซอร์วิส User และ Role สำหรับเตรียมข้อมูลบุคลากรและข้อมูลบทบาทหน้าที่งาน มีขั้นตอนการเหมือนกัน ตามขั้นตอนในภาพประกอบที่ ก - 4

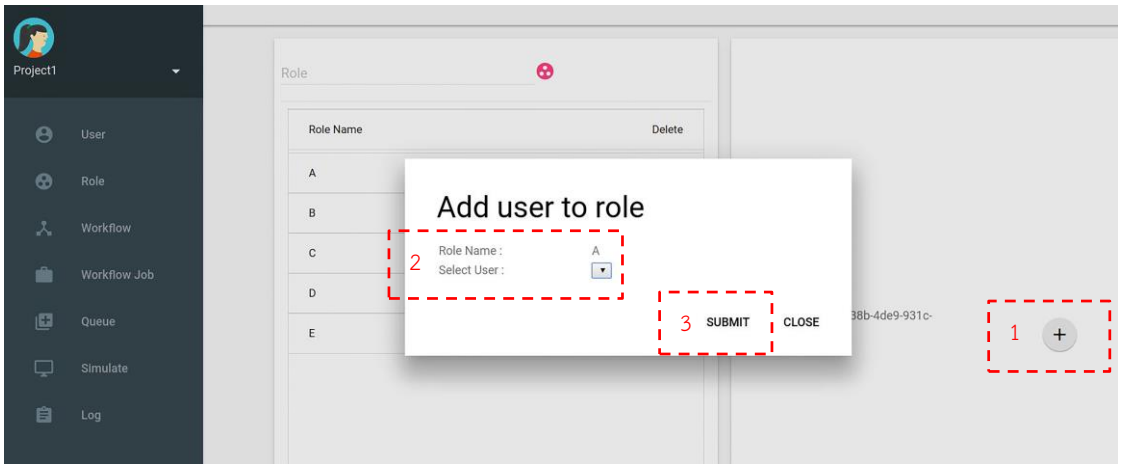


ภาพประกอบที่ ก - 4 การเพิ่มข้อมูลบุคคลในการจำลอง

สามารถเพิ่มคุณสมบัติของบุคคลนั้น เช่น เงินเดือน สถานะการทำงาน และเวลาที่ใช้เป็นค่าน้ำหนักในการทำงาน เป็นต้น ดังภาพประกอบที่ ก - 5 ส่วนบทบาทหน้าที่ของการทำงานจะถูกกำหนดในเมนู Role ซึ่งขั้นตอนการกำหนดบทบาทหน้าที่การทำงานของบุคคลนั้น แสดงในภาพประกอบที่ ก - 6



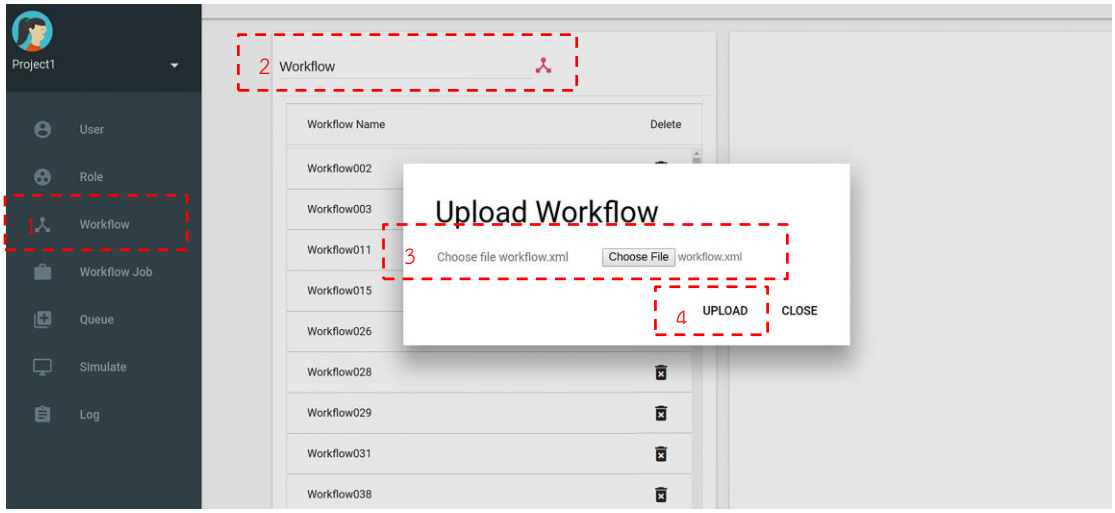
ภาพประกอบที่ ก - 5 การกำหนดค่าคุณสมบัติสำหรับบุคคล



ภาพประกอบที่ ก - 6 การกำหนดค่าคุณสมบัติบทบาทให้กับบุคคล

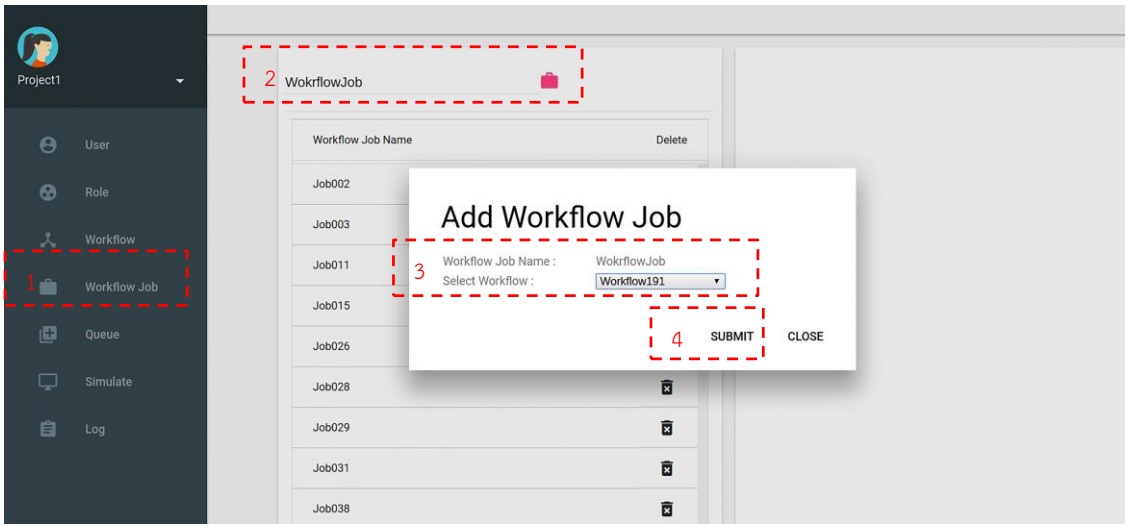
3. การเตรียมไฟล์กระแสงานและคิวงาน

เซอร์วิส Workflow และ WorkflowJob สำหรับการเตรียมไฟล์กระแสงานที่ต้องการจะอัพโหลดไปบน HDFS เพื่อรอประมวลผลกระแสงานโดย Oozie ขั้นตอนการใช้งานเซอร์วิส workflow ดังภาพประกอบที่ ก - 7



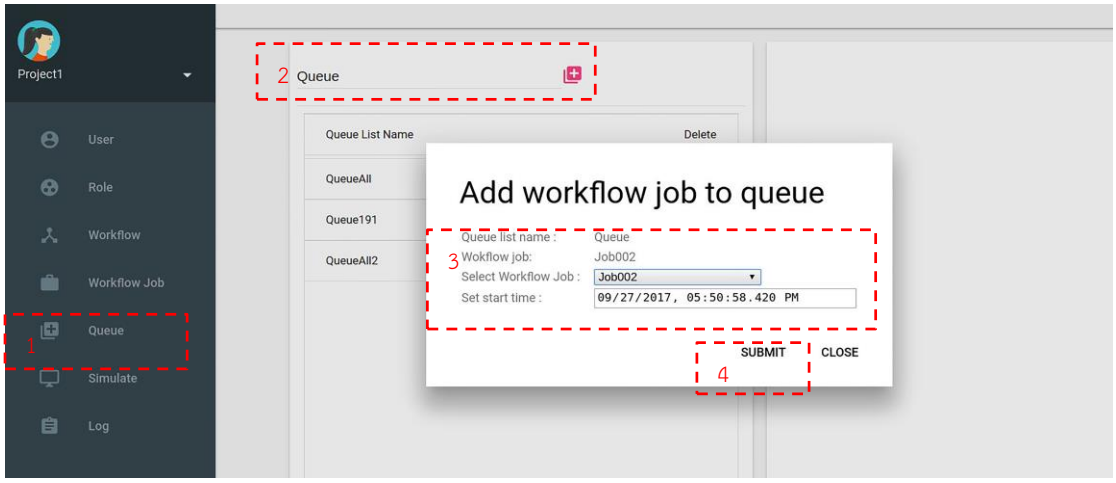
ภาพประกอบที่ ก - 7 การอัพโหลดไฟล์กระแสงาน

เมื่ออัพโหลดไฟล์กระแสงานไปบน HDFS แล้ว เซอร์วิส WorkflowJob ก็เตรียมข้อมูลกระแสงานสำหรับใช้ในการประมวลผลดังแสดงภาพประกอบที่ ก - 8



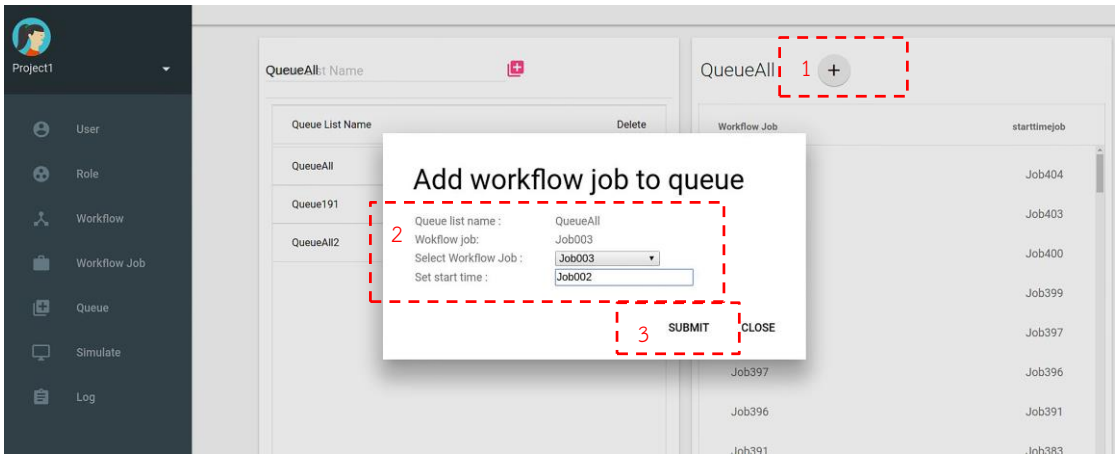
ภาพประกอบที่ ก - 8 การเตรียมข้อมูลกระแสงาน

สำหรับเซอร์วิส Queue เพื่อเตรียมคิวงานและเพิ่มกระแสงานในคิว กำหนดค่าเวลาที่เริ่มในการประมวลผลกระแสงาน เพื่อที่จะส่งคิวงานประมวลผลต่อไป ดังภาพประกอบที่ ก - 9



ภาพประกอบที่ ก - 9 การเพิ่มคิวการดำเนินงาน

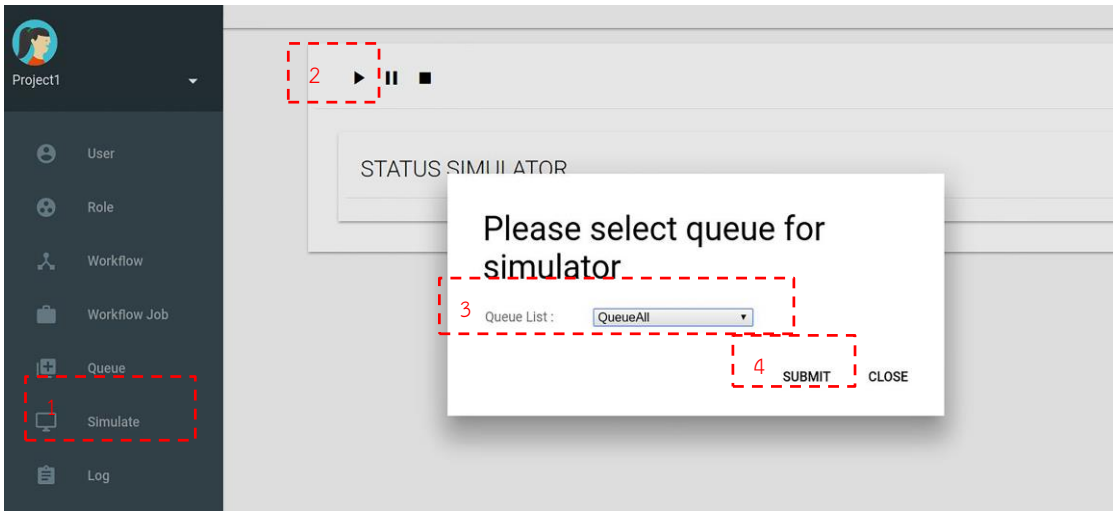
ในคิวสามารถเพิ่มกระแสงานขึ้นได้ โดยทำตามขั้นตอนในภาพประกอบที่ ก - 10 และสามารถกำหนดค่าเริ่มต้นได้ในประมวลผลกระแสงานได้



ภาพประกอบที่ ก - 10 การเพิ่มกระแสงานในคิว

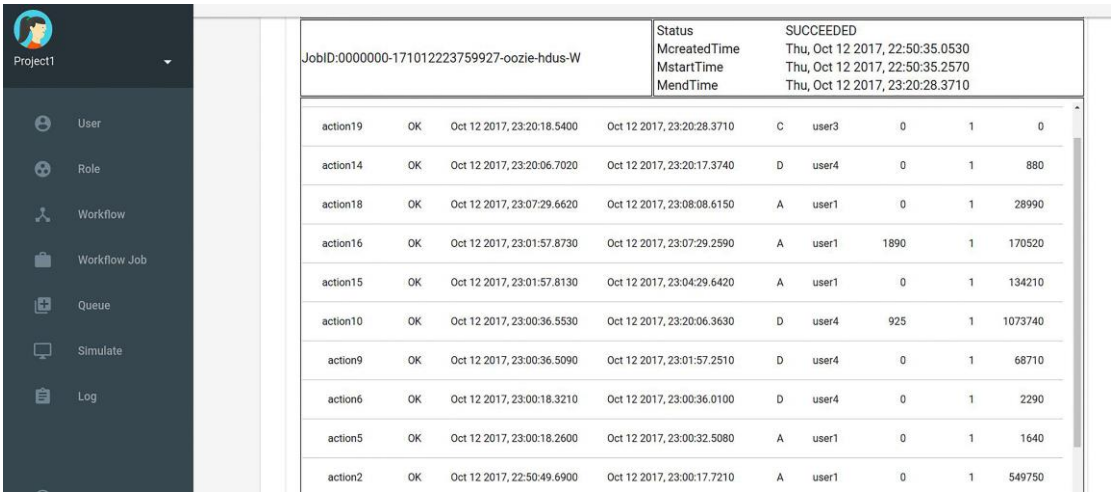
4. การประมวลผลกระแสงานและผลลัพธ์

ในการดำเนินการประมวลผลกระแสงานจะใช้เซอร์วิส Simulator เพื่อที่เริ่มต้นกระบวนการส่งคิวงานที่เลือกเพื่อประมวลผลกระแสงานใน Oozie ต่อไป ดังแสดงในภาพประกอบที่ ก - 11



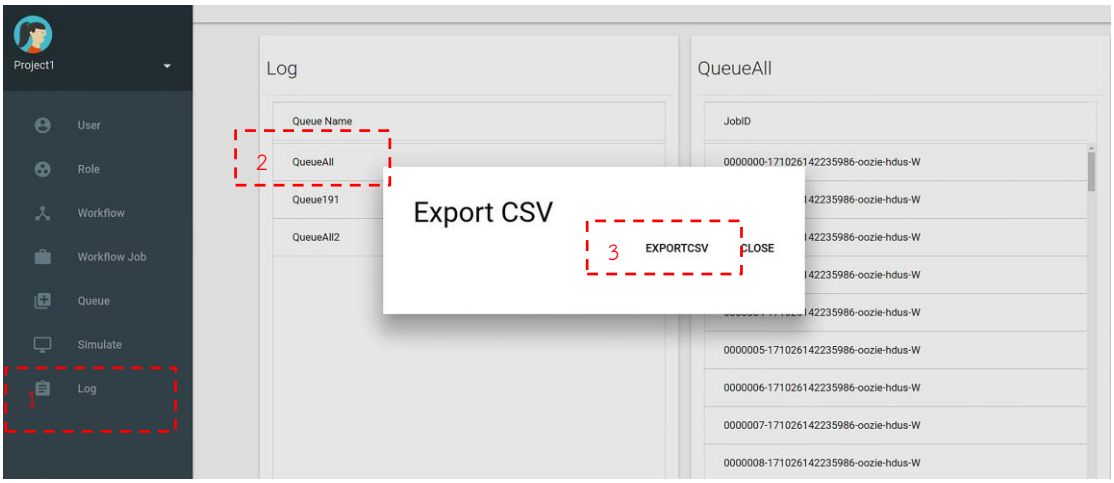
ภาพประกอบที่ ก - 11 ขั้นตอนการประมวลผลกระแสงานในคิว

สำหรับเซอร์วิส Simulator ยังสามารถติดตามสถานะของการประมวลผลกระแสงานได้อีกด้วย ซึ่งจะแสดงในภาพประกอบที่ ก - 12



ภาพประกอบที่ ก - 12 สถานะการประมวลผลกระแสงาน

เซอร์วิส Log การแสดงผลลัพธ์ของการประมวลผลกระแสงาน โดยสามารถเลือกแปลงไฟล์ข้อมูลสถานะของการประมวลผลเป็นไฟล์ CSV ได้ เพื่อใช้ในการวิเคราะห์ผลต่อไป และสามารถเลือกการแสดงเป็นคิวงานหรือครั้งละ JobID ได้ ดังแสดงในภาพประกอบที่ ก - 13



ภาพประกอบที่ ก - 13 ไฟล์ผลลัพธ์ของการประมวลผลกระแสงาน

ภาคผนวก ข
ผลการทดสอบประสิทธิภาพการประมวลผลกระแสงาน

5. ผลการทดสอบการประมวลผลกระแสงาน

สำหรับผลการทดสอบการประมวลผลของกระแสงานจำนวน 100 กระแสงาน รายละเอียดเปอร์เซ็นต์ความล่าช้าของการประมวลผลในแต่ละบทบาทของกระแสงาน แสดงในตารางที่ ข - 1

ตารางที่ ข - 1 เปอร์เซ็นต์ความล่าช้าของแต่ละบทบาทในงานของกระแสงาน

Jobname	Role A	Role B	Role C	Role D	Role E
Job002	0.20	0.00	0.00	0.02	0.00
Job003	0.00	0.00	0.00	0.00	16.00
Job011	0.00	0.00	32.94	0.00	0.00
Job015	0.49	0.00	0.00	0.00	0.00
Job026	0.00	0.00	0.00	0.00	0.00
Job028	0.00	0.00	0.00	0.17	0.00
Job029	0.50	0.00	1.96	0.00	0.00
Job031	0.00	0.00	8.42	0.00	0.00
Job038	2.98	6.62	0.00	0.00	0.00
Job039	0.19	0.00	0.00	0.00	0.00
Job042	0.04	0.02	0.00	0.00	0.00
Job047	0.00	0.00	0.06	0.00	0.00
Job049	0.05	0.27	0.00	0.00	0.00
Job058	0.34	0.00	0.25	4.07	0.00
Job061	0.66	0.00	0.00	0.00	0.00
Job070	0.00	0.00	1.43	0.00	0.00
Job078	0.00	0.00	0.00	0.00	0.31
Job079	0.00	0.76	0.00	0.00	0.00
Job083	0.00	0.00	2.40	0.00	0.00
Job086	0.00	0.00	0.00	4.45	0.00
Job095	0.00	0.00	0.00	0.00	0.23
Job100	0.00	0.00	71.86	0.00	0.00
Job103	9.34	0.00	0.01	0.00	0.00
Job107	0.00	0.00	9.35	0.00	0.00
Job108	0.00	6.51	0.00	0.00	0.00

Jobname	Role A	Role B	Role C	Role D	Role E
Job110	0.00	0.00	39.68	0.00	0.00
Job112	0.00	0.06	14.80	0.00	0.83
Job120	0.00	0.00	0.00	0.22	0.00
Job121	0.00	0.00	0.00	1.47	0.00
Job122	0.00	0.01	0.00	0.00	0.00
Job125	0.00	0.00	0.00	0.00	0.00
Job129	0.00	0.00	0.10	0.00	0.00
Job135	0.00	0.00	0.00	0.00	0.00
Job144	0.40	0.00	0.00	79.38	0.00
Job147	32.84	0.00	0.00	0.01	0.00
Job149	0.37	0.00	0.00	0.00	0.00
Job151	0.00	0.00	0.00	0.82	0.00
Job153	0.02	0.47	0.00	0.00	0.00
Job157	0.00	0.00	0.00	0.00	0.00
Job159	0.00	13.98	0.00	0.00	65.04
Job171	0.00	0.00	0.00	0.00	0.00
Job174	0.06	0.16	0.03	0.00	0.00
Job178	0.00	0.00	1.69	0.00	101.32
Job184	0.00	0.00	0.00	0.00	4.74
Job187	0.00	0.00	0.00	3.99	25.28
Job191	0.00	0.00	0.00	0.00	0.00
Job193	0.00	0.00	0.00	0.00	0.02
Job195	0.00	0.00	0.00	0.00	0.36
Job198	0.01	0.00	0.01	0.00	0.00
Job201	0.00	0.00	0.00	0.00	0.03
Job202	0.00	0.00	0.24	0.00	0.00
Job206	0.00	0.00	0.00	0.15	37.94
Job215	0.01	0.00	0.00	11.92	0.00
Job218	0.00	0.04	0.00	0.00	1.70
Job221	0.10	0.00	0.00	0.00	0.00
Job223	0.00	0.00	0.00	0.00	0.00
Job228	0.00	0.00	0.00	0.00	0.00
Job234	0.00	0.00	0.00	1.55	0.00
Job243	0.00	0.00	0.00	0.00	0.03

Jobname	Role A	Role B	Role C	Role D	Role E
Job246	0.00	0.00	0.00	0.00	6.55
Job251	0.00	0.00	6.77	0.00	0.08
Job253	0.00	0.00	0.00	2.44	1.49
Job263	0.00	0.00	0.00	0.00	3.47
Job264	0.00	0.00	0.00	0.00	7.15
Job267	0.00	0.00	0.00	0.00	0.00
Job277	1.16	0.00	0.00	0.00	0.02
Job278	0.00	0.02	0.00	1.45	0.00
Job282	0.00	0.19	0.00	0.00	0.00
Job292	0.00	0.00	0.00	0.00	0.00
Job295	7.28	7.74	0.00	0.00	0.00
Job296	0.00	0.34	0.00	0.00	0.00
Job304	0.00	0.00	0.00	0.00	0.00
Job313	0.16	0.00	0.00	0.00	0.00
Job316	0.00	0.00	0.00	0.00	0.00
Job317	0.00	0.00	0.03	0.00	0.00
Job319	12.68	3.48	0.00	0.00	0.00
Job323	0.34	0.00	0.00	7.45	0.00
Job327	0.00	0.00	5.79	0.07	0.00
Job332	0.00	0.00	3.99	0.00	0.00
Job334	0.00	0.00	0.00	14.67	0.00
Job336	0.34	0.00	0.00	0.00	0.00
Job337	0.00	0.00	0.00	0.00	0.00
Job348	0.00	1.58	0.00	0.00	0.00
Job349	0.00	0.00	0.00	0.00	0.00
Job351	0.00	0.00	0.00	0.00	0.00
Job353	0.00	0.00	0.00	0.52	0.00
Job363	0.00	0.06	0.00	0.00	0.00
Job366	0.00	0.24	4.17	0.00	40.21
Job369	0.00	0.50	1.61	0.00	0.00
Job370	0.00	0.00	0.00	0.00	37.18
Job380	0.00	0.00	136.54	0.00	0.00
Job383	0.06	0.00	0.00	0.00	1.05
Job391	0.00	119.80	0.00	0.00	2.67

Jobname	Role A	Role B	Role C	Role D	Role E
Job396	0.00	0.00	0.00	0.00	8.81
Job397	0.00	0.00	0.00	0.00	0.41
Job399	0.00	0.00	0.00	0.00	0.00
Job400	23.53	0.00	0.00	0.00	1.10
Job403	0.00	0.17	0.00	0.00	0.00
Job404	0.00	0.00	11.25	0.00	0.00
Job405	0.00	0.00	0.61	0.00	5.92

ภาคผนวก ค
ผลงานตีพิมพ์เผยแพร่จากวิทยานิพนธ์

JCSSE 2017

14th International Joint Conference on Computer Science and Software Engineering (JCSSE 2017)

IEEE Catalog Number: CFP1732P-ART

ISBN: 978-1-5090-4834-2

Workflow Simulation based on Cloud Platform for Office Automation System

Kanittha Promsakul
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University
Hat Yai, Songkhla 90112, Thailand
5510120112@psu.ac.th

Somchai Limsiroratana
Department of Computer Engineering, Faculty of Engineering, Prince of Songkla University
Hat Yai, Songkhla 90112, Thailand
somchai.l@psu.ac.th

***Abstract*— This research focuses on improving business process worked under workflow simulation. In the business process, there are several properties to be considered such as person, cost, time. The current workflow simulation cannot combine these properties to simulate the situation for business decision making. In this paper, the improving business workflow simulation is proposed by adding the business properties to the workflow simulation. Then, our simulation is able to simulate his/her work and monitor/collect status data for analysis. Both Apache Oozie and Apache Hadoop are used in this work as fundamental framework. Oozie is a workflow engine job scheduler system which is performed by the Apache Hadoop. Apache Hadoop is big data framework that provides distributed processing and storage. As a result, our workflow simulation can increase the efficiency of workflow management and scalable for solving the big business workflow process.**

Keywords- workflow; simulation; distributed system; Apache Oozie; Apache Hadoop;

I. INTRODUCTION

Workflow is a process of operation from starting to ending that follows the business rule of organization. Present organization works base on computer and manage by workflow management system. Workflow simulation is widely used as a tool to analyze and improve business process [1][2][3][4]. Simulation can analyze the past data and use its result to develop current workflow system. Advantages of workflow simulation are to reduce cost and to manage workflow system for optimal efficiency.

Workflow is almost used by organization to manage human and work. The workflow simulation can demonstrate how the system flows. Manager can attain maximum benefit by using the workflow simulation without performing the real work on the person. Workflow systems are complex, particularly when it needs multiple different workflow simulators to solve a problem. The work in [1] are presented using the YAWL workflow management system and the ProM used process mining framework in simulation for analyzing to simulate operational decision making. Research of [2] based on the Extended Xinpai Driven Workflow Model to descripted and analyzed quantitative the business process. WorkflowSim [3] used CloudSim (framework for simulating cloud computing) for providing fault tolerant clustering of workflow. Finally, in research using RealView™ [4] is business process tools to automate maintenance and analysis task of workflow. However, these workflow simulators cannot fully support adding more properties, scalable and extensible organization system.

Workflow engine is a tool for processing workflow. This research requires flexible, scalable and extensible workflow engine. The paper in [5] compared a workflow engine working on cloud platform including Hamake, Apache Oozie, Azkaban, Cascading, Orangescape Studio, JBOSS, Pegasus and YAWL. Apache Oozie is selected to use in this work because it is an active open source project, scalable and extensible base on Hadoop, developed by well known Apache Foundation, active project on website and good performance [6]. Moreover, large organizations use the Oozie for their workflows system such as Yahoo, IBM BigInsights, Cloudera, Hortonworks and Amezon EMR etc.

This paper presents simulation of a business workflow management system. To create simulated person working models that has role in the work and supports scalable system. The simulation can store job status data in order to analyze and optimize workflow system. This paper uses both Apache Oozie and Apache Hadoop for workflow scheduler and distributed processing respectively. The Oozie engine manages the workflow and sends action executor of each workflow node to Hadoop for execution [7]. Hadoop is a library framework for distributed processing of large data sets across clusters of cluster computing. The advantages of Hadoop is that if an error or failure occurs, Hadoop can handle it automatically. It can also be used to sort out complexity and hence is tolerant to failure. In addition, Oozie is scalable, reliable, extensible system and its architecture can be developed into a real organizational system.

The proposed workflow simulation system in this paper is expected to be a tool for optimizing organization's workflow suitable for individual allocated capacity by reducing its working time and cost. Section II presents architecture of simulator on Oozie engine. Designed and implemented of the simulation system is described in Section III. Results of the simulation system are in Section IV. Section V is discussion

978-1-5090-4834-2/17/$31.00 ©2017 IEEE

performance of workflow simulation. Finally, in Section VI, conclusions and feature works on the usage of this simulation are discussed.

II. Simulation System Architecture

Architecture design shown in figure 1 is the proposed simulation system which has 4 parts including simulator, Oozie workflow engine, HBase and Hadoop distributed file system (HDFS). The simulator is web interface to establish connection between the system and users. The administrator or users can connect to the interface by REST API with JSON data. Oozie workflow engine manages task activity according base on Oozie by simulator through Oozie REST service. All system data are stored in HBase through the WebObject API [8] running on HDFS. WebObject API is NoSQL API service for interfacing with HBase. It provides storage of hierarchical structure of key-value base objects on very large table which supports a big general organization. Finally, in last part, HDFS has the MapReduce that parallel execute jobs form Oozie engine. The flow of Oozie job compose of action nodes. The main important action node that designs for person working simulation is SyncAction. The SyncAction was developed by deriving from custom action executor, ActionExecutor class in Oozie which described detail in section III.A.

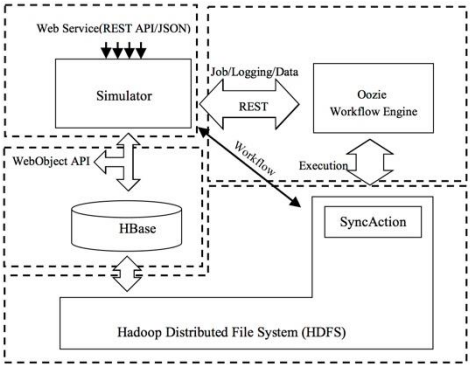


Figure 1. Simulation system architecture.

III. Design and Implementation

In this section, the detail of our design and implementation are described including SyncAction, system data storage, workflow execution and monitoring. The SyncAction supports adding properties to execute in Oozie. The system data storage provides the WebObject API to store the necessary data and configuration in HBase. The workflow execution is used to prepare and execute the custom workflow and the monitor is a tracking module of simulator.

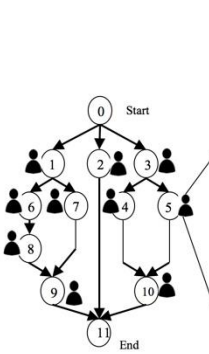
A. SyncAction

Oozie [9] has a collection of actions for containing control flow nodes and action nodes of workflow. Control flow nodes are nodes to define starting, ending (end, kill node), execution route (decision, fork and join node), execution of a computation and processing task (Hadoop MapReduce, HDFS, SSH, Pig, and JAVA action) and Oozie also support additional type of actions by custom action executor.

In this work, we develop custom action executor called SyncAction for purpose of simulate person working models and deployed on Oozie server. Each person working model has role of person, duration time, cost and other properties etc. Role of person is condition to choose the work. If the person is not in role, that means the person is not eligible in work. Duration time is amount of working time of person. Person has ability difference to work. Amount of cost is communication and expenses of system. These properties assigned in workflow and configuration file to submit job for Oozie server. When simulator start, workflow in queue is sent to Oozie server. Oozie manages and processes workflow and sends to Hadoop for job execution, SyncAction. The SyncAction simulates the addition parameter and record a workflow result in Hbase.

B. System Data Storage

HBase is a NoSQL distributed database of big tables. It is developed on top of Hadoop HDFS. The work in [8] implemented WebObject API based on HBase for storing data and information of university. The WebObject API is REST service to collect tree structure data. The advantage of the WebObject API is to support storing large data. It is easily designed to create, read, update and delete (CRUD) general objects. We used the WebObject API to store all system data and parameters for simulator such as configuration, workflow file, queue and status of job. Configuration consists of person, role, property to be parameter for SyncAction. Workflow files are definition action nodes for execution on HDFS. Queue is a list of workflow for submitting to simulate on Oozie. The last information is status for showing the state of the job in processing and completion from Oozie.




```
<workflow-app name="sync-time-wf"
xmlns="uri:oozie:workflow:0.2">
  <start to="action0"/>
  …

  <fork name="fork3">
      <path start="action4"/>
      <path start="action6"/>
  </fork>
  <action name="action5">
      <sync xmlns="uri:oozie:sync-action:0.1">
        <actionname>action5</actionname>
        <role>manager</ role>
        <duration>1640177522</ duration >
        <cost>209715200</cost>
        <objectid_account>
              ${objectid_account}
        </objectid_account>
        <project>${project}</project>
      </sync>
      <ok to="join10"/>
      <error to="fail"/>
  </action>
  <join name="join10" to="action10"/>
  …

  <kill name="fail">
      <message>Java failed, error
message</message>
  </kill>
  <end name="end"/>
</workflow-app>
```

Figure 2. Workflow model (Left) and outline of workflow.xml (Right).

C. Workflow Execution

1) Workflow Preparation

Workflow management system (WfMS) [10][11] is recently used to support management and execution for workflow processes. There are several of standardization for workflow description such as, WfMC, W3C, OASIS, OMG and BPMI [12]. In this research supported WfMC because it is open standard, non-profit, standard organize by workflow management coalition (WfMC) and Oozie also support. Workflow model defines actions from beginning to ending for processing. Oozie workflow is directed acyclic graph (DAG).

Preparation of workflow and configuration, the flow of actions (person working) must be described in workflow.xml file with WfMC format and upload into HDFS. The parameter for each action must be prepared and stored in HBase. For examples, the workflow of 10 actions show in figure 2 (left) can be described as XML file shown in figure figure 2 (right). For simulate a person working No.5, the parameters for execution action No. 5 such as role 'manager', duration 1640177522, cost 209715200, etc. are prepared and store in HBase.

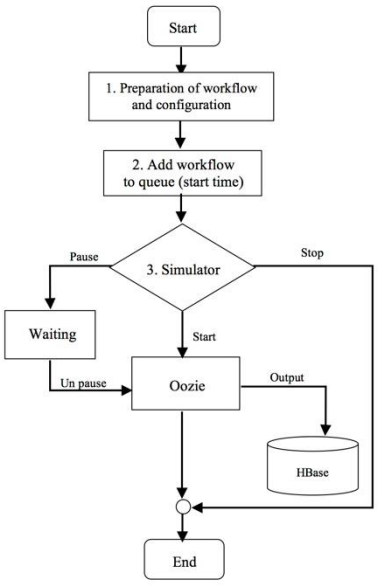


Figure 3. Process of workflow simulation.

2) Executor

This part describes simulator and process of simulation. The simulator is a web service running on Apache Tomcat 7 and connects to Oozie by REST API. The simulation service has application scope, that has thread away running on background. Therefore, the simulator design uses a single thread for protecting thread safety. The simulator prepares workflow execution by loading properties from HBase and send to Ooze with execution command.

Process of simulation from beginning to ending shown figure 3. The first step is a preparation of workflow and configuration which already described topic in workflow preparation above. The system allow user to prepare/store multiple workflows. The 2nd step, we can choose any prepared workflow to add in queue and assign the start time. Then, 3rd step is starting simulator. Simulator load workflow in order of start time and send each workflow and parameters to Oozie for workflow execution. Each SyncAction execute and store output to HBase that can be read from simulator for monitoring and tracking or using REST API to download data for analysis later. We can control the started simulator by pausing or stopping for more real-time tracking and diagnosis problems. After finish execution all workflow in queue, simulator and status.

D. Monitoring

Simulator starting and running can be tracked the status of job (create time, start time and end time), status of SyncAction (action name, status, start time, and end time). The status shows in real time on web service. All of Monitoring values of are store in HBase which will be very useful for analysis and optimization of person working model.

IV. RESULTS

In this simulation system, we test the system by random generated workflow using Daggen [13]. The Daggen supports DAGs and proposes synthetic graph to reach of simulation. Output files from Daggen were converted to workflow.xml files, 100 workflows were generated.

User or administrator can select and upload workflows file to HDFS prepare for simulation shown in figure 4.

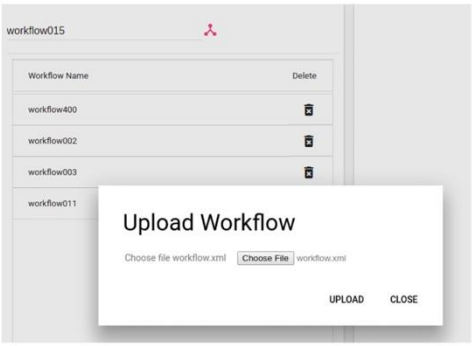


Figure 4. Upload workflow file to HDFS.

Adding workflows to queue and set start time for execution in simulation shown in figure 5.

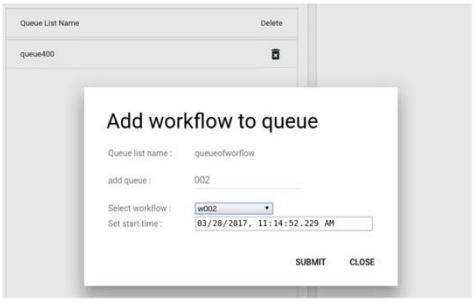


Figure 5. Adding workflow to queue.

Selected list of workflows in queue, sent to Oozie for processing shown in figure 6.

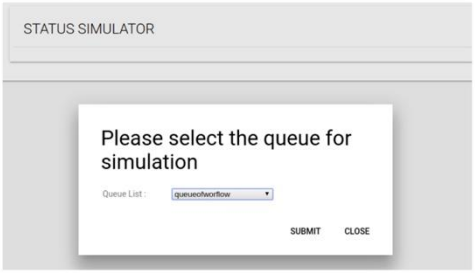


Figure 6. Selected workflow in queue to Oozie.

Figure 7 shows a status and a tracking of the job in simulation. Start time, end time and status parameters can be used for analysis. The user can request to the web service for detail of job. Figure 8 shows the detail of job in JSON format. The user can also request the detail of each action from web service shown in Figure 9.

Figure 7. Status of workflow simulation.


```
"object_id": "32cd2c55-e4c8-48ef-ba6f-80b54bb1336e",
"parent_objectid": "9fd89f38-a9b8-485a-82c1-100d90f981b2",
"object_name": "0000004-170428100217135-oozie-hdus-W",
"object_type": "",
"path": "/Test/log/job/0000004-170428100217135-oozie-hdus-W",
"is_file": false,
"created": 1493353624399,
"modified": 1493353624800,
"property": {
  "status": "RUNNING",
  "createtimejob": "Fri Apr 28 11:17:42 ICT 2017",
  "starttimejob": "Fri Apr 28 11:17:42 ICT 2017",
  "appname": "sync-time-wf",
  "apppath": "hdfs://localhost:8020/user/hduser/apps/f0b247d0-84ad-49f3-9(
  "jobid": "0000004-170428100217135-oozie-hdus-W"
},
```


Figure 8. JSON file of job.


```
{
  "object_name": "action7",
  "object_id": "a15fdce1-2506-49e8-9a61-17a663983841",
  "is_file": false,
  "object_type": "",
  "property": {
    "getactiontime": "Fri Apr 28 11:22:14 ICT 2017",
    "cost": "25000",
    "actionname": "action7",
    "endactiontime": "",
    "status": "PREP",
    "accountname": "test5-37 0",
    "role": "student",
    "accountid": "a15c9259-188f-4c54-b81a-b88d93728486",
    "duedate": "-",
    "workingtime": "25000",
    "startactiontime": "Fri Apr 28 11:22:14 ICT 2017",
    "priority": "-",
    "actionid": "0000004-170428100217135-oozie-hdus-W@action7"
  },
  "path": "/Test/log/job/0000004-170428100217135-oozie-hdus-W/action7",
  "created": 1493353343428,
  "modified": 1493353345025
},
```


Figure 9. JSON file of action7.

V. Discussion

Since our simulation system is designed to support the large workflow scale. The organization which requires to enlarge its workflow can reach the goal by using our simulation system. Our simulation system bases on Oozie engine conducting on Hadoop which is big data framework. Although this work does not provide a performance testing because of lack large resources. However, the performance is related to the Oozie performance which already proved that it has good enough performance [6] on Yahoo production system. The results showed that the Oozie can process 1250 jobs per minutes with 640 threads. There were 52k job to be submitted to the Oozie and the jobs finished with in 8 hours. The number of action can increase to 50 actions per workflow and the large number of actions per workflow did not affect to the Oozie performance. The overhead per action decreased when the number of action increased because the overhead of action to prepare and launch is more than the overhead to transit the action between nodes.

VI. Conclusion and Future Work

Workflow simulation presents relevant information about process which is useful for analyzing to diagnose business process, especially research on large scale office automation. In this paper, we proposed and developed workflow simulation system which supports scalability and extension based on Oozie and Hadoop jobs. The workflow simulation fully supports the

workflow represented in DAG under WfMC format and adds working person models of workflow simulation system.

In the future work, we would like to improve efficiency of our workflow simulation system by optimizing the person property. Since the current person property has only few parameters. The extra parameters such as potentiality or working queue of each person will be added to the person property, then the simulation efficiency can be increased.

REFERENCES

[1] A. Rozinat, M.T. Wynn, W.M.P. van der Aalst, A.H.M. ter Hofstede, C.J. Fidge, "Workflow simulation for operational decision support," Data & Knowledge Engineering, vol. 68, pp. 834-850, September 2009

[2] C. Xiang-qun and G. Wei, "Research on Workflow Model simulation technology," 2010 International Conference on Computer Application and System Modeling (ICCASM 2010), Taiyuan, 2010, pp. V10-181-V10-184.

[3] W. Chen and E. Deelman, "WorkflowSim: A toolkit for simulating scientific workflows in distributed environments," 2012 IEEE 8th International Conference on E-Science, Chicago, IL, 2012, pp. 1-8.

[4] J.G. Everton and R.D. Stafford, "Using workflow business process tools in simulation modeling," In Proceedings of the 37th conference on Winter simulation (WSC '05). Winter Simulation Conference pp.2063-2067, 2005.

[5] A. Bala, and I. Chana, "Design and deployment of workflows in cloud environment," International Journal of Computer Applications, vol.51, pp. 0975 – 8887, August 2012.

[6] M. Islam, A. K. Huang, M. Battisha, M. Chiang, S. Srinivasan, C. Peters, and A. Neumann, "Oozie: towards a scalable workflow management system for hadoop," In Proceedings of the 1st ACM SIGMOD Workshop on Scalable Workflow Execution Engines and Technologies, p. 4. ACM, May 2012.

[7] C. Olston, G. Chiou, L. Chitnis, F. Liu, Y. Han, M. Larsson, A. Neumann, V.B.N. Rao, V. Sankarasubramanian, S. Seth, C. Tian, T. ZiCornell, and X. Wang, "Nova: continuous Pig/Hadoop workflows," In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data (SIGMOD '11), ACM, New York, NY, USA, pp.1081-1090, June 2011.

[8] N. Thongkao and S. Limsiroratana, "Design of cloud-based university data structure," 2014 Fourth International Conference on Digital Information and Communication Technology and its Applications (DICTAP), Bangkok, 2014, pp. 186-191.

[9] Apache Oozie, Retrieved March 25, 2013, https://oozie.apache.org/.

[10] S. Meilin, Y. Guangxin, X. Yong and W. Shangguang, "Workflow management systems: a survey," Communication Technology Proceedings, 1998. ICCT '98. 1998 International Conference *on,* Beijing, 1998, pp. 6 vol.2.

[11] E.A. Stohr, and J.L. Zhao, "Workflow automation: Overview and research issues," Information Systems Frontiers, 2001, pp.281-296.

[12] J. Mendling, "Business process execution language for web service (BPEL)," In EMISA Forum, vol. 26, no. 2, pp. 5-8, 2006.

[13] Daggen a synthetic task graph generator, Retrieved December 10, 2016, https://github.com/frs69wq/daggen.

ประวัติผู้เขียน

ชื่อ สกุล นางสาวขนิษฐา พรหมสกุล

รหัสประจำตัวนักศึกษา 5510120112

วุฒิการศึกษา

วุฒิ	ชื่อสถาบัน	ปีที่สำเร็จการศึกษา
วิศวกรรมศาสตรบัณฑิต (วิศวกรรมคอมพิวเตอร์)	มหาวิทยาลัยสงขลานครินทร์	2554

การตีพิมพ์เผยแพร่ผลงาน

Kanittha Promsakul and Somchai Limsiroratana *"Workflow Simulation based on Cloud Platform for Office Automation System,"* The 14th International Joint Conference on Computer Science and Software Engineering (JCSSE), 12-14 July 2017