NUEJ
NARESUAN UNIVERSITY
ENGINEERING JOURNAL

การบูรณาการข้อมูลแบบกระจายบนเครื่องลูกข่าย
Distributed Data Integration on Client Side

เชาวรัตน์ แตงเรือง[1*], วรลักษณ์ คงเด่นฟ้า[2] และพงศ์พันธ์ กิจสนาโยธิน[3]
Chaowarat Tangrueng[1*], Woralak Kongdenfha[2] and Phongphun Kijsanayothin[3]

[1*,2,3]ภาควิชาวิศวกรรมไฟฟ้าและคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ มหาวิทยาลัยนเรศวร
[*]Corresponding author : chaowaratt@gmail.com
E-mail: woralakk@gmail.com[2], kphongph@nu.ac.th[3]

บทคัดย่อ

การบูรณาการข้อมูลเป็นสิ่งจำเป็นและมีความสำคัญมากกับองค์กรต่าง ๆ ทั้งขนาดเล็กจนถึงขนาดใหญ่ เนื่องจากหน่วยงานจะต้องนำข้อมูลต่าง ๆ จากหน่วยงานที่เกี่ยวข้องกันมาทำการวิเคราะห์ ประเมินผล เพื่อใช้ในการกำหนดแผนหรือนโยบาย แต่เนื่องจากข้อมูลที่ถูกจัดเก็บนั้นมีขนาดใหญ่และกระจายตัว จึงอาจส่งผลให้เกิดปัญหาในเรื่องของการประมวลผล หรือพื้นที่ในการจัดเก็บข้อมูลจากการบูรณาการ ส่งผลให้การตอบสนองผลลัพธ์ล่าช้า อีกทั้งค่าใช้จ่ายในการดูแลจัดการระบบและฐานข้อมูลเพิ่มมากขึ้น ดังนั้นงานวิจัยนี้จึงได้นำเสนอแนวคิดในการพัฒนาระบบเพื่อแก้ไขปัญหาดังกล่าว โดยการย้ายภาระงานมาไว้ที่เครื่องลูกข่าย มีการแบ่งส่วนการทำงานตามความต้องการของผู้ใช้งาน คำนวณแบบขนานและกระจายตามที่ข้อมูลอยู่เพื่อเพิ่มประสิทธิภาพ จากการวัดผลการทำงานของระบบที่พัฒนาตามแนวคิดการวิจัยนี้ แสดงให้เห็นว่าระบบตัวอย่างสามารถรองรับจำนวนข้อมูลและบริการได้มากขึ้นอย่างมีประสิทธิภาพ (เครื่องที่พัฒนาตามแนวคิดการวิจัยยังคงสามารถทำงานได้เมื่อมีเครื่องลูกข่ายร้องขอข้อมูลมากกว่า 32 เครื่องในขณะที่เครื่องแบบทั่วไปนั้นไม่สามารถทำงานได้)

คำสำคัญ: การบูรณาการข้อมูล ฐานข้อมูลแบบกระจาย การรองรับภาระงานที่เพิ่มขึ้น

Abstract

The data integration is very important to both small and large organizations. The organizations have to use the information from multi-stakeholder to analyzed and evaluate plan or policy. The problems in the data processing or storage/ memory occurs because the data information is large and distributed for the example, long time response to requester, take more cost of systems or databases maintenance. This research proposes a concept to develop a system for large and distributed data integration by moving the workload to the client. We separate tasks based on the needs of users and parallelizing distributed information for optimizing results. The result shows this concept can support the larger data and service that make system is scalable and improve a lot of performance compare with general concept at more than 32 concurrent clients request.

Keywords: data integration, distributed database, scalability

1. บทนำ

ข้อมูลสารสนเทศเป็นสิ่งที่องค์กรให้ความสำคัญมาก เนื่องจากข้อมูลเหล่านี้ถูกนำมาใช้ประโยชน์ในการพัฒนาองค์กร ช่วยในเรื่องการวางแผนนโยบายต่างๆ ซึ่งเทคโนโลยีสารสนเทศที่ก้าวหน้าขึ้นช่วยให้บุคลากรนั้นสามารถเข้าถึงข้อมูลเหล่านี้และใช้ประโยชน์ได้มากขึ้น แต่เนื่องด้วยข้อมูลสารสนเทศเหล่านี้นั้นมีขนาดใหญ่ อีกทั้งถูกจัดเก็บในแหล่งข้อมูลที่แตกต่างกันและมีการกระจายตัว ซึ่งอาจส่งผลให้เกิดความล่าช้าในการประมวลผลข้อมูลขนาดใหญ่นี้หรือจากการนำข้อมูลมารวมกัน อีกทั้งระบบที่ใช้ในการดูแลจัดการข้อมูลต้องมีประสิทธิภาพ

และสามารถรองรับขนาดข้อมูลและผู้ใช้งานจำนวนมากขึ้นด้วย ซึ่งส่งผลกระทบต่อการดำเนินการและค่าใช้จ่ายขององค์กรเป็นอย่างมาก เช่น ข้อมูลสารสนเทศขององค์กรหนึ่งนั้นมีความเกี่ยวข้องและเชื่อมโยงกับข้อมูลสารสนเทศของอีกองค์กรหนึ่ง หรือจากหน่วยงานในสังกัด ต้องทำการบูรณาการข้อมูลเข้าด้วยกันและทำการสรุปผลหรือวิเคราะห์เพื่อนำไปใช้ประโยชน์เป็นต้น ดังนั้นจึงจำเป็นต้องมีการบูรณาการข้อมูลสารสนเทศเหล่านี้เข้าด้วยกัน เพื่อให้สามารถนำไปใช้ประโยชน์ได้ง่ายขึ้น

งานวิจัยนี้ได้นำเสนอแนวคิดในการจัดการกับข้อมูลสารสนเทศที่มีข้อมูลขนาดใหญ่และมีแหล่งข้อมูลมาจากหลายแหล่งข้อมูล ซึ่งช่วยในเรื่องการรองรับการใช้งานจากผู้ใช้งานเป็นจำนวนมากและมีการทดสอบความเหมาะสมของระบบที่พัฒนาตามแนวคิดการวิจัยเปรียบเทียบกับระบบที่ไม่ได้พัฒนาตามแนวคิดการวิจัยนี้

ระบบสารสนเทศเพื่อการบูรณาการข้อมูลสำหรับผู้ใช้งานที่ไม่มีทักษะในการพัฒนาชุดคำสั่งนั้นได้รับความนิยมและมีความน่าสนใจมาก เนื่องจากเป็นระบบที่ใช้งานง่ายใช้ทักษะพื้นฐานทางคอมพิวเตอร์ทั่ว ๆ ไปเท่านั้น ซึ่งเหมาะกับผู้ใช้งานที่ไม่มีทักษะในการพัฒนาชุดคำสั่งเนื่องจากไม่จำเป็นต้องพัฒนาชุดคำสั่งหรือเขียนคำสั่งเพื่อสืบค้นข้อมูล (Query) ระบบสารสนเทศเพื่อการบูรณาการข้อมูลสำหรับผู้ใช้งานที่ไม่มีทักษะในการพัฒนาชุดคำสั่งที่พัฒนาในรูปแบบของเว็บเพื่อรวบรวมข้อมูลต่าง ๆ เข้าไว้ด้วยกัน (Web Mashup) สามารถแบ่งออกเป็นสองประเภทได้แก่ ระบบที่ต้องกำหนดลำดับขั้นตอนในการทำงาน (Workflow) ซึ่งผู้ใช้งานจำเป็นต้องสร้างลำดับการทำงานในแต่ละขั้นตอนให้เสร็จสิ้นก่อนจากนั้นจึงให้ระบบทำการประมวลผลขั้นตอนต่าง ๆ ที่สร้างขึ้นทั้งหมด เมื่อระบบประมวลผลเสร็จก็จะได้ผลลัพธ์ออกมาในภายหลัง ซึ่งงานวิจัยที่มีการพัฒนาในรูปแบบนี้ได้แก่งานวิจัย [1,2,3] อีกประเภทหนึ่งคือระบบที่ไม่จำเป็นต้องมีการสร้างลำดับขั้นตอนการทำงาน ผู้ใช้งานสามารถเลือกตัวดำเนินการ (Operations) เพื่อใช้จัดการกับข้อมูลตามที่ต้องการและสามารถเห็นผลลัพธ์จากการทำงานในแต่ละขั้นตอนได้ทันที เช่นตัวอย่างงานวิจัย [4-8]

ระบบที่ต้องกำหนดลำดับขั้นตอนในการทำงานนั้น เริ่มแรกผู้ใช้งานต้องทำการสร้างหรือกำหนดการทำงานในแต่ละขั้นตอนตั้งแต่เริ่มต้นจนเสร็จสิ้น เช่น ลำดับแรกกำหนดให้ระบบทำการร้องขอข้อมูลจากแหล่งข้อมูลที่หนึ่ง ลำดับถัดมาให้นำข้อมูลที่ได้มาทำการกรองข้อมูลที่ไม่จำเป็นออก และสุดท้ายนำมาแสดงผล เป็นต้น หลังจากนั้นนำลำดับการทำงานที่สร้างนี้ให้ระบบทำการประมวลผล เมื่อประมวลผลเสร็จก็จะได้ผลลัพธ์ออกมา ในงานวิจัยของ IBM Damia [2] นั้นได้มีการพัฒนาระบบด้วยแนวคิดนี้เพื่อจัดการกับข้อมูลที่มีขนาดใหญ่ในระดับองค์กร ส่วนงานวิจัย [1,3] นั้นใช้แนวคิดเดียวกันแต่จัดการกับข้อมูลที่มีขนาดเล็กกว่า งานวิจัยที่กล่าวถึงนี้มีข้อเสียในเรื่องของการตอบสนองกับผู้ใช้งาน ผู้ใช้งานไม่สามารถเห็นผลลัพธ์ในแต่ละขั้นตอนได้ในทันที ต้องรอให้สร้างลำดับการทำงานและส่งให้ระบบประมวลผลเสร็จก่อนจึงจะเห็นผลลัพธ์ หากผลลัพธ์ไม่ตรงตามที่ต้องการ ผู้ใช้ต้องแก้ไขลำดับการทำงานใหม่และส่งไปประมวลผลอีกจนกว่าจะได้ผลลัพธ์ที่ต้องการ

ซึ่งปัญหาที่เกิดจากระบบที่ต้องกำหนดลำดับขั้นตอนในการทำงานนั้น สามารถแก้ไขได้โดยการใช้ระบบที่ไม่จำเป็นต้องมีการสร้างลำดับขั้นตอนการทำงานนั้นสามารถประมวลผลและได้ผลลัพธ์ทันทีที่ผู้ใช้เลือกตัวดำเนินการ เนื่องจากผู้ใช้งานเห็นผลลัพธ์ได้ในเวลาอันสั้นทำให้สามารถทำการแก้ไขการทำงานในแต่ละขั้นตอนได้อย่างรวดเร็ว ซึ่งส่งผลให้ได้รับผลลัพธ์ที่ตรงตามความต้องการได้เร็วขึ้น ตัวอย่างงานวิจัยในลักษณะนี้ได้แก่ งานวิจัย Intel Mash Maker [4] ซึ่งเป็นการพัฒนาส่วนขยายของเบราว์เซอร์ Firefox เป็นส่วนที่ช่วยให้ผู้ใช้งานสามารถจัดการกับข้อมูลต่าง ๆ บนหน้าเว็บปัจจุบัน จากหน้าเว็บอื่นรวมถึงบริการต่าง ๆ ได้ง่ายขึ้น ในการพัฒนาระบบที่ไม่มีลำดับการทำงานนี้จำเป็นต้องใช้แนวคิดหรือวิธีการเพิ่มเติมเพื่อทำการพัฒนาระบบให้กับผู้ใช้งานได้ใช้งานง่ายขึ้นโดยไม่จำเป็นต้องมีความรู้ทักษะในการพัฒนาชุดคำสั่ง

ในงานวิจัย [9,10] ได้นำเสนอแนวคิดในการแก้ปัญหากับข้อมูลขนาดใหญ่นี้ โดยที่งานวิจัย [9] จะใช้วิธีการประมวลผลแบบก้าวหน้า (Progressive Processing) โดยจะทำการประมวลผลข้อมูลทีละส่วนเพิ่มขึ้นไปเรื่อย ๆ จนครบทั้งหมด ในระหว่างการประมวลผลแต่ละส่วนก็จะมีการแสดงผลลัพธ์ให้กับผู้ใช้งาน ระบบนี้ได้นำฟังก์ชันในการคำนวณต่าง ๆ ไว้เครื่องแม่ข่ายหรือเซิฟเวอร์ (Server) ในการประมวลผลที่ต้องใช้เวลานาน ๆ นั้นหากมีผู้ใช้งานเป็นจำนวนมากอาจก่อให้เกิดปัญหาในการใช้งานระบบได้ ในงานวิจัย [10] นั้นมีลักษณะคล้ายกันแต่เน้นที่การแสดงผลการทำงานในลักษณะก้าวหน้า (Incremental Visualization) ซึ่งมีการประมวลผลจากการสุ่มชุดข้อมูลจากฐานข้อมูลขึ้นมาทำงานไปเรื่อย ๆ จนครบทั้งหมด ส่วนแสดงผลจะเปลี่ยนแปลงตลอดเวลามีเส้นแสดงขอบเขตข้อมูลที่ถูกประมวลผล มีข้อมูลแสดงค่าความถูกต้องของข้อมูลปัจจุบันกับข้อมูลทั้งหมดซึ่งหากผู้ใช้งานคิดว่าข้อมูลที่ใช้เพียงพอต่อค่าความถูกต้องแล้วสามารถหยุดประมวลผลได้ แต่

งานวิจัยนี้มีเพียงตัวดำเนินการของการคำนวณเท่านั้น ไม่มีส่วนของการรวมข้อมูลจากหลาย ๆ แหล่งข้อมูล

จากปัญหาในการบูรณาการข้อมูลที่กระจายและมีการใช้งานเป็นจำนวนมากนั้น งานวิจัยนี้จึงได้นำเสนอแนวความคิดในการแก้ไขปัญหาดังต่อไปนี้ (1) ปัญหาในการจัดการกับข้อมูลที่มีขนาดใหญ่ที่ข้อมูลนั้นไม่ได้อยู่ที่แหล่งข้อมูลเดียวกันสามารถแก้โดยวิธีการแบ่งส่วนข้อมูลและทำการประมวลผลแบบขนานโดยใช้ตัวดำเนินการและอาศัยวิธีการผลคูณคาร์ทีเซียน (Cartesian Product) [11] ดังที่นำเสนอในบทถัดไป เพื่อรวมข้อมูล และ (2) แก้ไขปัญหาการรองรับการขยายระบบ (Scalability) และผู้ใช้งานเป็นจำนวนมาก โดยการการย้ายภาระงานจากเซิฟเวอร์มาไว้ที่เครื่องลูกข่ายหรือไคลเอนต์ (Client) แทน

2. วิธีดำเนินการวิจัย

ในส่วนนี้จะอธิบายถึงหลักการการประมวลผลของตัวดำเนินการพื้นฐานที่จำเป็น ที่ถูกย้ายมาทำงานที่ฝั่งไคลเอนต์ ซึ่งแต่ละตัวดำเนินการนั้นจะมีการกำหนดนิยามเฉพาะขึ้นมาเพื่อให้สามารถทำงานบนฝั่งไคลเอนต์ได้และมีประสิทธิภาพ โดยตัวดำเนินการพื้นฐานที่นำเสนอประกอบด้วย 2 กลุ่มด้วยกัน คือ ตัวดำเนินการเดี่ยวซึ่งสามารถทำงานและให้ผลลัพธ์จากการทำงานภายในหนึ่งตัวดำเนินการ และตัวดำเนินการร่วมคือการนำตัวดำเนินการเดี่ยวมาทำงานร่วมกันเพื่อสร้างผลลัพธ์ขึ้นมาใหม่

ตัวดำเนินการ (Operatiors)

ตัวดำเนินการเดี่ยวประกอบด้วยตัวดำเนินการสำหรับ การเลือกข้อมูล (Select) การแปลงข้อมูล (Convert) การกรอง (Filter) ค่ามากสุด (Max) ค่าน้อยสุด (Min) ชุดข้อมูลที่ไม่ซ้ำกัน (Distinct) และตัวดำเนินการร่วม ได้แก่การจัดกลุ่ม (Group) การเรียงลำดับข้อมูล (Sort) โดยแนวคิดการวิจัยนี้ได้นำเสนอวิธีการรวมข้อมูลด้วยการใช้ผลคูณคาร์ทีเซียน (Cartesian product) ซึ่งนำมาใช้ในการสร้างผลลัพธ์จากการนำข้อมูลมารวมกัน อีกทั้งใช้เป็นชุดข้อมูลในการสร้างผลลัพธ์ของตัวดำเนินการอื่น ๆ เช่น Left-join Right-join และ Inner-join เป็นต้น และมีการใช้ตารางเสมือน (Virtual Table) ที่ซึ่งมีโครงสร้างเหมือนตารางจริงแต่จะยังไม่มีข้อมูลเข้ามาช่วยในการคำนวณและการทำงานแบบขนาน และการนำข้อมูลมาเติมลงตารางเสมือนของตัวดำเนินการเติมข้อมูล (Fill) ซึ่งตัวอย่างจะใช้ฐานข้อมูลของนักเรียนเป็นหลัก แต่สามารถประยุกต์ใช้ตัวดำเนินการเหล่านี้กับฐานข้อมูลใด ๆ ก็ได้

นิยามที่ 1 ตาราง Table T=(C,R) โดยที่ C คือเซตของชื่อคอลัมน์ในตารางและ R คือเซตของแถวทุกแถวในตาราง หรือกล่าวคือเซตที่มีลำดับของ {(c,v)} โดยที่ c ทุกตัวเป็นสมาชิกของ C และ v เป็นค่าของ c ซึ่ง r_i ทุกตัวเป็นสมาชิกของ R ใช้สำหรับแสดงข้อมูลแถวที่ i ใน R และฟังก์ชัน val(r_i, c) ใช้สำหรับการหาค่าของข้อมูลที่อยู่ในแถวที่ r_i และคอลัมน์ชื่อ c ตัวอย่างของตารางแสดงข้อมูลนักเรียนซึ่งประกอบด้วยคอลัมน์ รหัส ชื่อและอายุ ซึ่งสามารถเขียนได้ดังนี้

1) T = (C, R)
2) C = {รหัส, ชื่อ, อายุ }
3) R = <r_1, r_2, r_3>
4) r_1 = {(รหัส, 1100000000001),(ชื่อ, ธนัตพร สกุลหนึ่ง),(อายุ, 17)}
5) r_2 = {(รหัส, 1100000000002),(ชื่อ, ธันยารัตน์ สกุลสอง),(อายุ, 16)}
6) r_2 = {(รหัส, 1100000000003),(ชื่อ, นภัยพัต สกุลสาม),(อายุ, 17)}

ตารางที่ 1 ตัวอย่างตารางแสดงข้อมูลนักเรียน

รหัส	ชื่อ	อายุ
1100000000001	ธนัตพร สกุลหนึ่ง	17
1100000000002	ธันยารัตน์ สกุลสอง	16
1100000000003	นภัยพัต สกุลสาม	17

นิยามที่ 2 การเลือก Select คือการเลือกคอลัมน์ที่ต้องการแสดงข้อมูลโดย $T_2 \leftarrow$ Select(T_1, C) จะทำการสร้างผลลัพธ์ออกมาเป็น T_2 = (C, R_2) โดยการเลือกเซตของชื่อของคอลัมน์จาก C ของตาราง T_1 = (C_1, R_1) ทุก c เป็นสมาชิกของ C ถ้า c เป็นสมาชิกของ C_1 แล้ว val($r_i \in R_2$, c) = val($r_i \in R_1$, c) แต่ถ้า c ไม่เป็นสมาชิกของ C_1 แล้ว val($r_i \in R_2$, c) = null

นิยามที่ 3 การแปลงชนิดของข้อมูล Convert $T_2 \leftarrow$ Convert(T_1, func, c) โดยตัวดำเนินการนี้จะให้ผลลัพธ์เป็นตาราง T_2 = (C_1, R_2) โดยรับฟังก์ชัน func ซึ่งเป็นฟังก์ชันที่จะใช้ทำงานกับคอลัมน์ c ในตาราง T_1 = (C_1, R_1) ซึ่งทุกๆ $c_j \in C_1$

ถ้า $c_j = c$ แล้ว $val(r_i \in R_2, c_j) = func(val(r_i \in R_1, c_j))$ ถ้าไม่ใช่ $val(r_i \in R_2, c_j) = val(r_i \in R_1, c_j)$

นิยามที่ 4 การกรองข้อมูล Filter $T_2 \leftarrow Filter(T_1, c, o, v)$ ตัวดำเนินการนี้จะทำการกรองข้อมูลในคอลัมน์ c ของตาราง $T_1 = (C_1, R_1)$ ด้วยตัวเปรียบเทียบ $o \in \{ =, >, <, != \}$ ซึ่งจะให้ผลลัพธ์ตาราง $T_2 = (C_1, R_2)$ ที่มีเฉพาะข้อมูลแถวที่ $r_i \in R_1$ ที่ผลลัพธ์จากการเปรียบเทียบข้อมูลของ $val(r_i, c)$ กับ v ด้วยตัวเปรียบเทียบ o มีค่าเป็นจริงเท่านั้น

นิยามที่ 5 การหาค่าน้อยสุด Min $v \leftarrow Min(T_1, c)$ ให้ผลลัพธ์เป็นค่าของข้อมูล v จากตาราง $T_1 = (C_1, R_1)$ โดยที่ $v = min\{ val(r, c) \mid \forall r \in R_1 \}$

นิยามที่ 6 การหาค่ามากสุด Max $v \leftarrow Max(T_1, c)$ ให้ผลลัพธ์เป็นค่าของข้อมูล v จากตาราง $T_1 = (C_1, R_1)$ โดยที่ $v = max \{ val(r, c) \mid \forall r \in R_1 \}$

นิยามที่ 7 การหาชุดข้อมูลที่ไม่ซ้ำกัน Distinct $S \leftarrow Distinct(T_1, c)$ ให้ผลลัพธ์เป็นเซตของข้อมูล S จากตาราง $T_1 = (C_1, R_1)$ โดยที่ $S = \{ val(r, c) \mid \forall r \in R_1 \}$

นิยามที่ 8 การจัดกลุ่มข้อมูล Group $W \leftarrow Group(T_1, c)$ จะทำการสร้างเซตของตาราง $W = \{T_1', T_2', ..., T_n' \}$ จากตาราง $T_1 = (C_1, R_1)$ โดยที่ตารางแต่ละตัว (T_i') จะเป็นสามาชิกที่อยู่ในเซต W ซึ่งนิยามโดย $T_i' = (C_1, R_i') \in W$ และแต่ละ $r \in R_i'$ ถ้า $r_j \in R_i'$ แล้ว $val(r_i, c) = val(r_j, c)$ ซึ่งสามารถทำการสร้าง W ได้จากการเรียกใช้ตัวดำเนินการ Distinct หลังจากนั้นใช้ตัวดำเนินการ Filter ทำการดึงข้อมูลตามผลลัพธ์ของ Distinct ดังนี้ โดยทุก ๆ $s \in S$ และ $S = Distinct(T_1, c)$ ตาราง $T_i' = Filter(T_1, c, =, s)$

นิยามที่ 9 การจัดเรียงลำดับของข้อมูล Sort $T_2 \leftarrow Sort(T_1, c, t)$ ให้ผลลัพธ์เป็นตาราง $T_2 = (C_1, R_2)$ โดยการเลือกข้อมูลและทำการจัดเรียงข้อมูลที่ได้จากตาราง $T_1 = (C_1, R_1)$ โดยกำหนดการเรียงข้อมูลด้วย $t \in \{ASC, DESC\}$ ถ้า t = ASC คือเรียงข้อมูลจากน้อยไปมากแล้วทุกค่า $val(r_i, c) \leq val(r_{i+1}, c)$ แต่ถ้า t = DESC คือเรียงข้อมูลจากมากไปน้อยแล้วทุก ๆ ค่า $val(r_i, c) \geq val(r_{i+1}, c)$ ในการเรียงข้อมูลสามารถทำได้โดยการหาค่าที่แตกต่างกันทั้งหมดจาก Distinct และทำการเรียงข้อมูลที่ได้นี้ตามรูปแบบที่ต้องการ หลังจากนั้นเรียกใช้ตัวดำเนินการ Group ซึ่งจะทำให้ได้เซตของตารางซึ่งแต่ละตารางในคอลัมน์ c จะมีผลลัพธ์เป็นค่าเดียวกันทั้งหมด และนำข้อมูลของทุกตารางมาต่อกันด้วยตัวดำเนินการ Append ตามลำดับ หากข้อมูลมีค่าเหมือนกัน ข้อมูลทั้งสองตัวนั้นจะอยู่ในลำดับเดียวกับข้อมูลก่อนถูกจัดเรียง เช่น $1_1, 3_2, 4_3, 3_4, 1_5$ หลังจากผ่านขั้นตอนการจัดลำดับที่ stable จะได้ $1_1, 1_5, 3_2, 3_4, 4_3$ เมื่อเลขดรรชนีล่างบ่งบอกลำดับเริ่มต้นก่อนการจัดเรียง

นิยามที่ 10 การรวมกันของข้อมูล Join $T_3 \leftarrow Join(T_1, T_2)$ ในการรวมข้อมูลนั้นมีตัวดำเนินการต่าง ๆ มากมาย เช่น Left-join Right-join และ Inner-join เป็นต้น ซึ่งมีความแตกต่างกันและยากต่อการเข้าใจของผู้ใช้งานทั่ว ๆ ไปซึ่งอาจทำให้เกิดความสับสนในการใช้งาน งานวิจัยนี้จึงเสนอวิธีการรวมข้อมูลด้วยผลคูณคาร์ทีเซียน (Cartesian Product) ซึ่งจะสร้างตาราง $T_3 = (C_3, R_3)$ โดยที่ขนาดของ $|C_3| = |C_1| + |C_2|$ และ $|R_3| = |R_1| * |R_2|$

ตารางที่ 2 ตารางผลคูณคาร์ทีเซียน

colum1	colum2	...	colum(n)
row_{11}	row_{12}	...	row_{1n}
...	...	...	...
row_{m1}	row_{m2}	...	row_{mn}

จากตารางผลคูณคาร์ทีเซียนข้างต้นเป็นตารางที่เกิดจากผลคูณคาร์ทีเซียนของสองตารางที่มีผลลัพธ์จากการคูณจำนวนแถวของสองตารางเป็นจำนวน m แถวและผลลัพธ์การรวมคอลัมน์เป็น n คอลัมน์ หากนำข้อมูลจริงมาใส่ตารางผลคูณนี้จะทำให้ขนาดของข้อมูลที่ต้องจัดเก็บมีขนาดใหญ่มาก ในงานวิจัยจึงนำเสนอตารางเสมือน (Virtual Table) ซึ่งเป็นตารางที่มีโครงสร้างเหมือนกับตารางผลคูณคาร์ทีเซียนแต่ยังไม่มีข้อมูลซึ่งจะนำข้อมูลจริงมาใส่จากฟังก์ชัน Fill

นิยามที่ 11 การนำข้อมูลมาใส่ตารางเสมือน Fill $T_2 \leftarrow Fill(T_1, startRow, endRow)$ ตัวดำเนินการนี้จะทำการสร้างข้อมูลจริงในตารางเสมือนที่เกิดจากผลคูณคาร์ทีเซียน ซึ่งจะนำค่าจริงจากตารางที่นำมาคูณกัน เช่น ข้อมูลในแถวที่ 1 คอลัมน์ที่ 1 ของตารางแรกมาใส่ลงในในตารางเสมือน ในตัวดำเนินการนี้จะต้องระบุแถว startRow และ endRow ที่ต้องการนำข้อมูลมาใส่ตาราง โดยในแต่ละแถวของตาราเสมือนนั้นไม่สัมพันธ์กับ

แถวอื่นๆ กล่าวคือ แต่ละแถวนั้นจะดึงข้อมูลคนละชุดและข้อมูลจากแถวก่อนหน้าจะไม่มีผลกระทบกับแถวอื่นเลย ซึ่งเป็นข้อดีที่สามารถนำกระบวนการทำงานแบบขนานมาปรับใช้ได้ ในการคำนวณตำแหน่งของข้อมูลที่ต้องร้องขอจากแต่ละแหล่งข้อมูลนั้นสามารถทำได้จากสมการต่อไปนี้

$$\left\lceil \frac{m - \left(\prod_{i=x}^{1} |S_i|\right)\left(\left\lfloor \frac{m}{\prod_{i=x}^{1} |S_i|} \right\rfloor\right)}{\prod_{i=x-1}^{1} |S_i|} \right\rceil$$

โดยตัวแปร x นั้นจะต้องมีค่ามากกว่า 1 ซึ่งคือตำแหน่งของแหล่งข้อมูลจากจำนวนแหล่งข้อมูลทั้งหมด ตัวแปร m คือ ตำแหน่งของแถวของข้อมูลในตารางเสมือนที่ต้องการนำข้อมูลมาใส่ เช่น ต้องนำข้อมูลตำแหน่งใดจากแหล่งข้อมูลที่ 2 เพื่อนำมาสร้างแถวที่ 3 ของตารางเสมือน ดังนั้นต้องกำหนดให้ m = 3 และ x = 2 เป็นต้น

3. การทดลองและการวัดผล

ในงานวิจัยนี้ได้แบ่งการวัดผลออกเป็นสองส่วนด้วยกันคือ (1) การวัดประสิทธิภาพการทำงานของเซิฟเวอร์แบบทั่วไปกับเซิฟเวอร์ที่มีการพัฒนาตามแนวคิดการวิจัย (2) เปรียบเทียบประสิทธิภาพการประมวลผลระหว่างการประมวลผลบนเซิฟเวอร์กับการประมวลผลบนเครื่องไคลเอ็นต์ ซึ่งทั้งสองระบบทำงานแบบขนานและกระจายตามแนวคิดที่นำเสนอ ซึ่งในการทดลองนั้น ทดลองกับเครื่องในเครือข่ายเดียวกัน ทำการร้องขอข้อมูลผ่านบริการ (REST Service) โดยชุดข้อมูลที่นำมาทำการทดลองนั้น ประกอบด้วย ข้อมูลจากฐานข้อมูลระบบจัดเก็บข้อมูลนักเรียนรายบุคคลซึ่งนำตาราง students มาใช้ มีข้อมูลมากกว่าหนึ่งล้านคน และข้อมูลจากระบบการจัดการข้อมูลของโรงเรียน ซึ่งมีข้อมูลมากกว่าสามหมื่นโรงเรียน ข้อมูลทั้งสองแหล่งไม่ได้อยู่ที่เดียวกัน และเมื่อนำมารวมกันแล้วสามารถสร้างชุดข้อมูลได้หลายล้านแถว แต่ช่วงข้อมูลที่นำมาทดสอบจะเริ่มตั้งแต่ 20,496 แถวจนถึง 1,140,560 แถวซึ่งให้ผลลัพธ์เพียงพอต่อการวิจัย ผลการทดสอบแรกได้ผลดังแสดงในตารางที่

ตารางที่ 3 ผลการเปรียบเทียบการวัดประสิทธิภาพของเซิฟเวอร์แบบทั่วไปกับงานวิจัย

ประเภท	เซิฟเวอร์แบบทั่วไป			เซิฟเวอร์ตามแนวคิดการวิจัย		
จำนวนแถว	20,496 แถว	173,604 แถว	1,140,560 แถว	20,496 แถว	173,604 แถว	1,140,560 แถว
จำนวนเครื่องที่ร้องขอข้อมูล/เวลาเฉลี่ย	เวลาเฉลี่ย (วินาที)	เวลาเฉลี่ย (วินาที)	เวลาเฉลี่ย (วินาที)	เวลาเฉลี่ย (วินาที)	เวลาเฉลี่ย (วินาที)	เวลาเฉลี่ย (วินาที)
1	3.69	15.06	ไม่ได้ผลลัพธ์	4.82	12.08	60.23
2	2.94	21.16	ไม่ได้ผลลัพธ์	5.58	17.48	110.33
4	5.26	33.14	ไม่ได้ผลลัพธ์	9.58	34.60	223.38
8	10.07	33.07	ไม่ได้ผลลัพธ์	18.083	96.01	451.12
16	17.41	87.95	ไม่ได้ผลลัพธ์	36.17	161.20	ไม่ได้ผลลัพธ์
32	31.31	ไม่ได้ผลลัพธ์	ไม่ได้ผลลัพธ์	70.41	340.16	ไม่ได้ผลลัพธ์

เซิฟเวอร์แบบทั่วไปมีหน่วยประมวลผลกลางเป็น **Intel core(TM) 2 Duo CPU E8500 3.16GHz** และหน่วยความจำขนาด **8GB** เป็นเครื่องเซิฟเวอร์เดี่ยว และไม่มีการเพิ่มเครื่องมืออื่น ๆ เพื่อช่วยในการประมวลผลลัพธ์ การทำงานจะเป็นการดึงข้อมูลจากแหล่งข้อมูลต่าง ๆ ที่มีอยู่มาทำการรวมกันและประมวลผล หลังจากนั้นจึงส่งผลลัพธ์ไปยังเครื่องที่ร้องขอข้อมูล ซึ่งภายในเครื่องเซิฟเวอร์จะมีการเก็บผลลัพธ์เพื่อช่วยในการร้องขอข้อมูลที่มีชุดข้อมูลเดียวกัน **(Cachable Result)**

เซิฟเวอร์ที่พัฒนาตามแนวคิดการวิจัยมีประสิทธิภาพในการประมวลผลและการจัดเก็บข้อมูลเท่ากับเซิฟเวอร์แบบทั่วไป ต่างกันที่วิธีการในการบูรณาการข้อมูล ซึ่งใช้วิธีการแบ่งส่วนข้อมูลและประมวลผลด้วยตัวดำเนินการต่าง ๆ ที่นิยามขึ้นในงานวิจัยนี้

ในการเปรียบเทียบประสิทธิภาพการทำงานของสองเซิฟเวอร์นี้ ผู้วิจัยได้ทำการร้องขอข้อมูลในเวลาพร้อม ๆ กันไปยังทั้งสองเซิฟเวอร์ โดยเริ่มจากการร้องขอข้อมูลจาก 1 เครื่องขึ้นไปจนถึง 32 เครื่อง และทำการหาค่าเวลาเฉลี่ยของการได้รับข้อมูลจากเซิฟเวอร์ จากการร้องขอทั้งหมด 10 ครั้ง ในรอบแรกของการร้องขอข้อมูลจำใช้ข้อมูลที่เกิดการรวมกันของข้อมูล **20,496** แถว หลังจากนั้นทำเช่นเดิมซ้ำอีก แต่เพิ่มจำนวนข้อมูลมากขึ้น เป็น **173,604** แถว และ **1,140,560** แถว ตามลำดับ

จากตารางที่ **3** แสดงให้เห็นว่าที่จำนวนข้อมูลขนาด **20,496** แถวนั้น ทั้งสองระบบสามารถตอบสนองต่อผู้ร้องขอข้อมูลได้ครบถ้วน และเซิฟเวอร์แบบทั่วไปสามารถตอบสนองได้เร็วกว่าเนื่องจากมีการเก็บผลลัพธ์เอาไว้ หลังจากมีการร้องขอข้อมูลเพิ่มขึ้นเป็น **173,604** แถว เครื่องเซิฟเวอร์แบบทั่วไปเริ่มตอบสนองไม่ได้ เมื่อร้องขอข้อมูลมากกว่า 16 เครื่องพร้อม ๆ กัน เนื่องจากหน่วยความจำของเซิฟเวอร์ไม่เพียงพอ ส่วนเครื่องที่พัฒนาตามแนวคิดการวิจัยยังสามารถตอบสนองได้ครบทุกเครื่อง แต่เมื่อมีการร้องขอข้อมูลมากถึง **1,140,560** แถว ปรากฏว่า เครื่องเซิฟเวอร์แบบทั่วไปนั้นไม่สามารถตอบสนองได้เลย ส่วนเครื่องที่พัฒนาตามแนวคิดการวิจัยยังสามารถตอบสนองได้บางส่วน ซึ่งสามารถตอบสนองเครื่องที่ร้องขอข้อมูลพร้อม ๆ กันได้ 8 เครื่อง จากข้อมูลส่วนนี้สามารถสรุปได้ว่า ประสิทธิภาพการทำงานของเครื่องที่พัฒนาตามแนวคิดการวิจัยนี้ สามารถรองรับภาระงานได้มากกว่าเซิฟเวอร์แบบทั่วไป และการพัฒนาระบบบนฝั่งเซิฟเวอร์นั้นยังมีข้อจำกัดในเรื่องของทรัพยากรเครื่อง ในกาทำงานกับข้อมูลที่มีขนาดใหญ่ และการร้องขอข้อมูลจากเครื่องที่มีมากขึ้น จึงเป็นที่มาของการทดลองส่วนที่สอง เพื่อทำการวิจัยเปรียบเทียบประสิทธิภาพการทำงานเทียบกับเครื่องไคลเอนต์

ในการทดลองที่สอง จะทำการวัดประสิทธิภาพระหว่างการประมวลผลบนเซิฟเวอร์ที่พัฒนาตามแนวคิดการวิจัยกับเครื่องไคลเอนต์ที่พัฒนาตามแนวคิดการวิจัย โดยที่เครื่องเซิฟเวอร์นั้นเป็นเครื่องชุดเดียวกับการทดลองแรก และเครื่องไคลเอนต์นั้นมีหน่วยประมวลผล 3.16**GHz** และมีหน่วยความจำ 4**GB** โดยจะทำการเปรียบเทียบที่ระดับข้อมูล **1,140,560** แถว ซึ่งเป็นจำนวนข้อมูลที่ทำให้เครื่องเซิฟเวอร์ที่พัฒนาตามแนวคิดการวิจัย เริ่มตอบสนองเครื่องที่ร้องขอได้ไม่ครบถ้วน โดยทำการร้องขอข้อมูลพร้อม ๆ กัน เริ่มที่ 1 เครื่องจนถึง 32 เครื่องเช่นเดียวกัน ทำชุดละ 10 ครั้งและหาค่าเฉลี่ยของเวลาในการได้รับผลลัพธ์ ซึ่งผลลัพธ์จากการทดลองแสดงในตารางที่ 4

ตารางที่ 4 ผลลัพธ์การเปรียบเทียบการประมวลผลบนเซิฟเวอร์และไคลเอ็น

จำนวนที่ร้องขอข้อมูล	เวลาที่ได้ผลลัพธ์เฉลี่ย (วินาที)		เวลานานสุดที่ได้ผลลัพธ์(วินาที)	
	เซิฟเวอร์	ไคลเอนต์	เซิฟเวอร์	ไคลเอนต์
1	60.23	100.13	60.23	100.13
2	110.33	100.56	116.05	101.11
4	223.38	99.27	230.25	111.71
8	451.12	97.98	468.69	109.70
16	ไม่ได้ผลลัพธ์	106.39	ไม่ได้ผลลัพธ์	116.85
32	ไม่ได้ผลลัพธ์	121.54	ไม่ได้ผลลัพธ์	142.87

จากตารางที่ 4 แสดงผลการเปรียบเทียบการประมวลผลบนเซิฟเวอร์ที่พัฒนาตามแนวคิดการวิจัยและไคลเอนต์ที่พัฒนาตามแนวคิดการวิจัย จากเวลาเฉลี่ยของการได้รับผลลัพธ์นั้นแสดงให้เห็นว่าไคลเอนต์นั้นตอบสนองได้ดีกว่า โดยที่เวลาค่อนข้างคงที่แม้จะมีการร้องขอข้อมูลด้วยจำนวนเครื่องที่เพิ่มขึ้น และเวลาที่เครื่องร้องขอรอนานสุดนั้นก็ใกล้เคียงกับเวลาเฉลี่ย ทุกเครื่องที่ร้องขอสามารถได้รับผลลัพธ์ครบทั้ง 32 เครื่อง สรุปได้ว่าการพัฒนาระบบด้วยแนวคิดการวิจัยบนเครื่องไคลเอนต์นั้นช่วยรองรับภาระงานและเพิ่มประสิทธิภาพได้มากขึ้น

4. สรุปผลและข้อเสนอแนะ

งานวิจัยนี้ได้นำเสนอแนวคิดในการแก้ปัญหาเรื่องความซับซ้อนของการพัฒนาชุดคำสั่งเพื่อตอบสนองการสืบค้นข้อมูลปริมาณมากที่กระจัดกระจายกันอยู่โดยที่ไม่จำเป็นต้องมีความเชี่ยวชาญด้านการพัฒนาชุดคำสั่งที่ซับซ้อน แนวคิดดังกล่าวตั้งอยู่บนพื้นฐาน การสืบค้นข้อมูลจากตัวอย่าง (Query-by-Example) และ การประมวลผลแบบก้าวหน้า (Progressive Processing Query) เพื่อรองรับปริมาณงานจำนวนมาก แนวคิดนี้ยังเสนอการย้ายส่วนประมวลผลไปยังฝั่งของไคลเอนต์ ซึ่งลดการใช้ทรัพยากรบนเซิฟเวอร์ทำให้เซิฟเวอร์ยังคงให้บริการข้อมูลได้แม้จะมีเครื่องผู้ใช้เพิ่มมากขึ้นก็ตาม แต่ประสิทธิภาพส่วนหนึ่งก็ยังคงขึ้นอยู่กับเครื่องผู้ใช้งาน เนื่องจากต้องทำการประมวลผลเองแทนเซิฟเวอร์ งานวิจัยนี้ได้นำเสนอผลการทดลองเพื่อวัดประสิทธิภาพการประมวลผลของระบบที่พัฒนาโดยแนวคิดนี้ ซึ่งผลการทดลองแสดงให้เห็นว่าระบบที่

พัฒนาด้วยแนวคิดนี้สามารถรองรับภาระงานได้ดีขึ้นเมื่อเปรียบเทียบกับระบบที่พัฒนาด้วยหลักการทั่วไป อีกทั้งช่วยในเรื่องการตอบสนองการทำงานกับผู้ใช้ได้ดีขึ้น เนื่องจากสามารถสร้างผลลัพธ์ได้รวดเร็ว ซึ่งจะช่วยให้ผู้ใช้งานลดเวลาในการทำงานและลดจำนวนงานที่จะต้องทำลง

นอกจากนี้เรื่องความปลอดภัยของข้อมูลซึ่งเป็นสิ่งที่มีสำคัญมากต่อผู้ให้และผู้ใช้ข้อมูล เนื่องจากการย้ายภาระงานมาไว้ที่ฝั่งไคลเอนต์นั้นข้อมูลต่าง ๆ จะถูกเก็บไว้ที่ฝั่งไคลเอนต์แทนซึ่งง่ายต่อการเข้าถึง จึงจำเป็นต้องมีการจัดการเรื่องความปลอดภัยของข้อมูลที่ดี รวมถึงการหาเทคนิคหรือเครื่องมือที่จะช่วยในการแปลงโครงสร้างข้อมูลจากแหล่งข้อมูลต่าง ๆ ที่มีความแตกต่างกันให้เป็นไปตามแนวทางเดียวกัน

5. เอกสารอ้างอิง

[1] Yahoo. (2016) . Yahoo! Pipes. Retrieved from https://en.wikipedia.org/wiki/Yahoo!_Pipes

[2] Altinel, M., Brown, P., Cline, S., Kartha, R., Louie, E., Markl, V., Mau, L., Ng, YH., Simmen, D., & Singh, A. (2007). Damia: A Data Mashup Fabric for Intranet Applications. *Proceedings of the International Conference on Very Large Data Bases. 07*, 1370–1373.

[3] Wong, J., & Hong, J. (2006). Marmite: End-user Programming for the Web. *Extended Abstracts on Human Factors in Computing Systems. 06*, 1541–1546.

[4] Ennals, R., Brewer, E., Garofalakis, M., Shadle, M., & Gandhi, P. (2007, December). Intel Mash Maker: Join the Web. *SIGMOD. 36*(4), 27–33.

[5] Tuchinda, R., Szekely, P., & Knoblock, C.A. (2007). Building Data Integration Queries by Demonstration. *Proceedings of the 12th International Conference on Intelligent User Interfaces. IUI 07*, 170–179.

[6] Tuchinda, R., Szekely, P., & Knoblock, C.A. (2008). Building Mashups by Example. *Proceedings of the 13th International Conference on Intelligent User Interfaces. IUI 08*, 139–148.

[7] Tuchinda, R., Szekely, P., & Knoblock, C.A. (2008). Building Mashups by Example. *Proceedings of the 13th International Conference on Intelligent User Interfaces. IUI 08*, 139–148.

[8] Sugiura, A., & Koseki, Y. (1998). Internet Scrapbook: Automating Web Browsing Tasks by Demonstration.*Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology. UIST 98*, 9–18.

[9] Barnett, M., Chandramouli, B., DeLine, R., Drucker, R., Fisher, D., Goldstein, J., Morrison, P., & Platt, J. (2013). Stat!: An Interactive Analytics Environment for Big Data. *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data. SIGMOD 13*, 1013–1016.

[10] Fisher, D., Popov, I., Drucker, S., & schraefel. (2012). Trust Me, I'M Partially Right: Incremental Visualization Lets Analysts Explore Large Datasets Faster. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. CHI 12*, 1673–1682.

[11] Seth Warner and Mathematics. (1990, June). Compositions Induced on Cartesian Products and Function Spaces. *Modern Algebra*, 13, 90- 100