การเปรียบเทียบประสิทธิภาพของพื้นที่จัดเก็บในแพลตฟอร์มบิ๊กดาต้า
Comparison of Storages Performance in Big Data Platform

เกียรติศักดิ์ อุทัยศรี[1*], วรภัทร ไพรีเกรง[1], ทศพล พันธุ์กำแหง[2], วีรพงศ์ ตั้นเจริญ[2], ณรงค์ ภูมิสุข[2]

Kiattisak Utaisri[1*], Worapat Paireekreng[1], Tosapon Pankumhang[2], Weerapong Tuncharoen[2], Narong Phoomsuk[2]

[1]สาขาวิชาเทคโนโลยีสารสนเทศ วิทยาลัยนวัตกรรมด้านเทคโนโลยี และวิศวกรรมศาสตร์ มหาวิทยาลัยธุรกิจบัณฑิตย์ กรุงเทพฯ 10210 ประเทศไทย

[1]Information Technology, College of Innovative Technology and Engineering, Dhurakij Pundit University, Bangkok 10210, Thailand

[2]กองวิชาคณิตศาสตร์และคอมพิวเตอร์ ส่วนการศึกษา โรงเรียนนายร้อยพระจุลจอมเกล้า, นครนายก, 26001

[2]Mathematics and Computer Department, Academic Division, Chulachomklao Royal Military Academy, Nakhon Nayok, 26001, Thailand

**Corresponding Author. E-mail: worapat.png@dpu.ac.th*

(Received: May 31, 2024, Revised: July 8, 2024, Accepted: July 25, 2024)

บทคัดย่อ: ระบบสารสนเทศเพิ่มขึ้นอย่างรวดเร็ว โดยข้อมูลที่มาจากแหล่งข้อมูลต้นทางจะเก็บแบบกระจัดกระจายในระบบภายนอก ก่อนจะถูกเชื่อมโยงมาจัดเก็บในแพลตฟอร์มบิ๊กดาต้า ซึ่งถูกออกแบบมาให้สามารถรองรับการจัดเก็บและประมวลผลข้อมูลขนาดใหญ่ได้ทุกประเภทอย่างมีประสิทธิภาพ สำหรับนำไปใช้ประโยชน์ในมิติต่าง ๆ เช่น การวิเคราะห์ข้อมูล การให้บริการและแลกเปลี่ยนข้อมูล และการทำรายงาน เพื่อให้ผู้บริหารสามารถนำข้อมูลและรายงานเหล่านี้ ไปใช้ในการวิเคราะห์และวางแผนการขับเคลื่อนองค์กรด้วยข้อมูลได้อย่างแท้จริง

อย่างไรก็ตาม ข้อมูลเชื่อมโยงเข้ามาจัดเก็บในแพลตฟอร์มบิ๊กดาต้านั้น มาจากระบบภายนอกที่มีรูปแบบการจัดเก็บที่หลากหลาย ทำให้ระบบมีรูปแบบข้อมูลบนหน่วยจัดเก็บข้อมูลที่ต่างกัน เช่น โครงสร้างแบบเรียงตามแถวเป็นหลัก หรือแบบเรียงตามคอลัมน์เป็นหลัก การจัดเก็บข้อมูลเป็นไฟล์แบบไบนารี หรือเป็นไฟล์แบบข้อความ รวมทั้งรองรับการบีบอัดข้อมูล เป็นต้น เนื่องจากแต่ละรูปแบบต่างก็มีทั้งข้อดีและข้อเสียที่ต่างกันทำให้ไม่มีรูปแบบใดที่ดีที่สุด คณะผู้วิจัยจึงได้ทำการศึกษาเปรียบเทียบประสิทธิภาพของรูปแบบไฟล์ข้อมูลที่จัดเก็บบนแพลตฟอร์มบิ๊กดาต้า เพื่อหารูปแบบไฟล์ข้อมูลที่เหมาะสมสำหรับการทำงานในกรณีต่าง ๆ ได้อย่างมีประสิทธิภาพมากที่สุด โดยผลการทดลองชี้ให้เห็นว่ารูปแบบไฟล์ข้อมูลแบบเรียงตามแถวเหมาะกับการนำเข้าและจัดเก็บข้อมูลที่นำมาจากต้นทางมากกว่า ส่วนรูปแบบไฟล์ข้อมูลแบบเรียงตามคอลัมน์จะเหมาะกับการเรียกดูข้อมูลหรือนำข้อมูลไปวิเคราะห์ด้วยชุดคำสั่งที่ซับซ้อนมากกว่า

คำสำคัญ: รูปแบบไฟล์ ข้อมูลขนาดใหญ่ แพลตฟอร์มบิ๊กดาต้า

Abstract: From the continuous application of information technology in various fields today. This causes the amount of data in information systems to increase rapidly. The data that comes from the data source

is stored sparsely in external systems. Before being imported and linked to store in the Big Data Platform, which is designed to support the storage and processing of large data. Get all types efficiently before being put to use in various dimensions such as data analysis, data service and sharing, and reporting. So that executives can use these data and reports to truly analyze and plan to drive the organization with data.

However, the imported data is stored in the big data platform coming from different external systems resulting in a variety of data storage formats. Each format has a different structure on the storage, such as a row-based or column-based data structure, storing as a binary or text file, and supporting data compression. Because each format has both advantages and disadvantages, we therefore studied and compared the efficiency of data file formats stored on the big data platform in order to find the most suitable data file format for working in various cases as efficiently as possible. The results of the experiment indicated that column-based data structures are better suited for importing and storing data taken from external sources, while row-based data structures are suitable for querying or analyzing the data with more complex commands.

Keywords: File Format, Big Data, Big Data Platforms

1. บทนำ

ปัจจุบันเทคโนโลยีสารสนเทศมีส่วนสำคัญในการดำเนินการในด้านต่าง ๆ ทำให้เกิดความสะดวกสบายในการทำกิจกรรมและการใช้ชีวิตประจำวันของมนุษย์มากขึ้น นอกจากนี้เรายังสามารถเข้าถึงเทคโนโลยีเหล่านี้ได้ง่ายกว่าเดิม ด้วยการให้บริการผ่านทางแพลตฟอร์มต่าง ๆ ไม่ว่าจะเป็นการติดต่อสื่อสาร การเดินทาง ร้านอาหาร การเกษตร การศึกษา การท่องเที่ยว การกีฬา แม้กระทั่งการแพทย์ เช่น ระบบอัจฉริยะทางการแพทย์ ได้นำเทคโนโลยีเข้ามาช่วยในการรักษาผู้ป่วยแบบครบวงจร ตั้งแต่การติดตั้งอุปกรณ์แบบสวมใส่ (Wearable Device) เพื่อใช้ตรวจจับและติดตามอาการของผู้ป่วย การนำระบบปัญญาประดิษฐ์ (Artificial Intelligence) หรือ AI มาทำโมเดลวิเคราะห์ข้อมูลและสามารถให้คำแนะนำในการรักษาขั้นต้น ไปจนถึงการรักษาที่ซับซ้อนโดยไม่ต้องเดินทางไปโรงพยาบาลได้

จากการนำเทคโนโลยีสารสนเทศ มาประยุกต์ใช้ในงานด้านต่าง ๆ ที่มีมาอย่างต่อเนื่อง ทำให้เกิดข้อมูลที่หลากหลายจำนวนมหาศาล ซึ่งถูกจัดเก็บในรูปแบบที่แตกต่างกันไปตามแหล่งข้อมูลต้นทาง (Data Source) การนำข้อมูลมาใช้ในการวิเคราะห์นั้นจะต้องดำเนินการเชื่อมโยงบูรณาการข้อมูล (Data Integration) จากแหล่งข้อมูลต่าง ๆ มาจัดเก็บไว้ที่ทะเลสาบข้อมูล (Data Lake) ซึ่งเป็นพื้นที่รวบรวมข้อมูลดิบ (Raw Data) ของแพลตฟอร์มบิ๊กดาต้า (Big Data Platforms) ซึ่งถูกออกแบบมาให้สามารถรองรับการจัดเก็บและประมวลผลข้อมูลขนาดใหญ่ได้ทุกประเภทอย่างมีประสิทธิภาพก่อนจะนำข้อมูลเหล่านี้ไปใช้ในการวิเคราะห์ หรือใช้ประโยชน์ในด้านอื่น ๆ ต่อไป

อย่างไรก็ตามรูปแบบการจัดเก็บข้อมูลบนแพลตฟอร์มบิ๊กดาต้านั้นมีการพัฒนาและปรับปรุงมาอย่างต่อเนื่อง จนในปัจจุบันมีรูปแบบให้เลือกใช้ที่หลากหลาย เช่น ไฟล์แบบข้อความ และไฟล์แบบไบนารี ไฟล์ที่มีโครงสร้างการจัดเก็บแบบเรียงตามแถว และไฟล์แบบเรียงตามคอลัมน์ เป็นต้น

เพื่อให้การเลือกรูปแบบในการจัดเก็บข้อมูลบนแพลตฟอร์มให้เหมาะสมและมีประสิทธิภาพมากที่สุด คณะผู้วิจัยจึงได้ศึกษาวิจัย “การวิจัยศึกษาการเปรียบเทียบ ประสิทธิภาพของพื้นที่จัดเก็บในแพลตฟอร์มบิ๊กดาต้า”[1] [2]

2. วัตถุประสงค์

2.1 เพื่อเปรียบเทียบประสิทธิภาพของการนำเข้าและเรียกใช้ไฟล์ในรูปแบบต่าง ๆ ในงานแพลตฟอร์มบิ๊กดาต้า

2.2 เพื่อวิเคราะห์ไฟล์ในรูปแบบต่าง ๆ ว่าไฟล์ประเภทใดมีความเหมาะสมในการประมวลผล และวิเคราะห์ ในแพลตฟอร์มบิ๊กดาต้า

3. เอกสารและงานวิจัยที่เกี่ยวข้อง

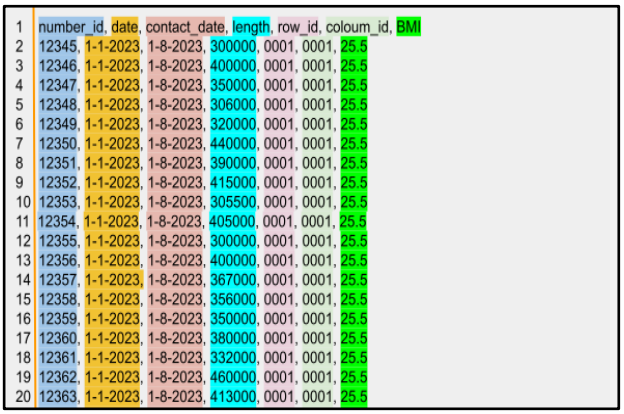
รูปแบบไฟล์ในแพลตฟอร์มบิ๊กดาต้า

การวิจัยการเปรียบเทียบประสิทธิภาพของพื้นที่จัดเก็บในแพลตฟอร์มบิ๊กดาต้า (Big Data Platforms) จะใช้ชุดข้อมูลสำหรับใช้ในการประมวลผลด้วยคำสั่งภาษา Hadoop Query Language (HQL) ซึ่งเป็นคำสั่งที่ใช้ในการสืบค้นและเรียกดูข้อมูลที่จัดเก็บอยู่ในแพลตฟอร์มบิ๊กดาต้า ซึ่งชุดข้อมูลเหล่านี้จัดเก็บไว้หลายรูปแบบและมีขนาดต่างกัน เพื่อให้ใช้ในการศึกษาวิจัย สำหรับรูปแบบของชุดข้อมูลที่เลือกมาใช้ในการศึกษาวิจัยนี้ เลือกมาจากชุดข้อมูลที่มีนิยมใช้ในปัจจุบัน ประกอบด้วย ชุดข้อมูลจำนวน 4 ประเภท ได้แก่ รูปแบบไฟล์ข้อความ (Text), รูปแบบไฟล์ AVRO, รูปแบบไฟล์ Parquet, และรูปแบบไฟล์ ORC

รูปแบบไฟล์ข้อความ (Text)

ไฟล์ข้อมูลที่อยู่ในรูปแบบไฟล์ข้อความ (Text File Format) เป็นรูปแบบพื้นฐานที่มีในทุกระบบ ซึ่งยังเป็นที่นิยมใช้ในแพลตฟอร์มบิ๊กดาต้าในปัจจุบัน เนื่องจากเป็นไฟล์ข้อมูลที่เราสามารถเปิดอ่านเนื้อหาในไฟล์ได้โดยตรง โดยอาจเป็นไฟล์แบบไม่มีโครงสร้าง (Unstructured File) เช่น ไฟล์ Log หรือไฟล์ข้อความทั่วไป แต่ไฟล์ข้อความที่นิยมจัดเก็บมากที่สุดในแพลตฟอร์มบิ๊กดาต้าจะเป็นไฟล์แบบกึ่งโครงสร้าง (Semi-Structured File) เพราะมีประโยชน์ในการใช้งานหลายอย่าง และที่ใช้มากที่สุดในปัจจุบัน ได้แก่ ไฟล์แบบ CSV และไฟล์แบบ JSON

Shafranovich ได้กำหนดไฟล์แบบ CSV (Comma-Separated Values) ให้เป็นมาตรฐานตาม RFC 4180 [3] เพื่อใช้เป็นมาตรฐานในการแลกเปลี่ยนข้อมูลและแปลงข้อมูลระหว่างโปรแกรม Spreadsheet เช่น MS Excel หรือ Google Sheet เป็นต้น โดยไฟล์ข้อมูลแบบ CSV จะประกอบด้วย ส่วนหัว (Header) ซึ่งอาจจะมีหรือไม่มีก็ได้ แล้วตามด้วยส่วนระเบียนข้อมูล (Record) ซึ่งเป็นแถวของข้อมูล (Row) ในตารางของ Spreadsheet โดยแต่ละแถวจะถูกแบ่งเขตข้อมูล (Field) หรือคอลัมน์ (Column) ในตาราง ด้วยเครื่องหมายจุลภาค (,) และจะปิดท้ายข้อมูลในแถวนั้นด้วยอักขระ ขึ้นบรรทัดใหม่ (CRLF) ข้อมูลในแต่ละคอลัมน์อาจมีเครื่องหมายอัญประกาศ (“) กั้นหน้าและกั้นหลังส่วนข้อมูลของคอลัมน์นั้น โดยเฉพาะกรณีที่ข้อมูลในคอลัมน์มีช่องว่างหรือ Space คั่นระหว่างตัวอักษร เมื่อมีบล็อคของอัญประกาศจึงทำให้เกิดความชัดเจนในการระบุข้อมูลในคอลัมน์ ตัวอย่าง โครงสร้าง ของไฟล์แบบ CSV แสดงในภาพที่ 3.1

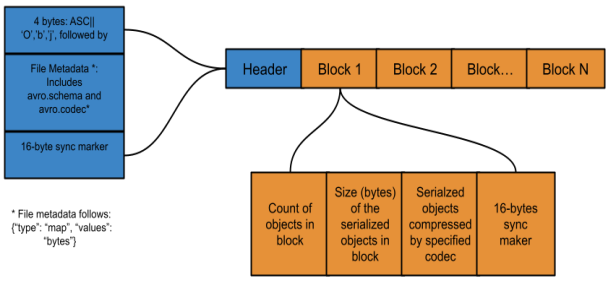


	number_id, date, contact_date, length, row_id, coloum_id, BMI
2	12345, 1-1-2023, 1-8-2023, 300000, 0001, 0001, 25.5
3	12346, 1-1-2023, 1-8-2023, 400000, 0001, 0001, 25.5
4	12347, 1-1-2023, 1-8-2023, 350000, 0001, 0001, 25.5
5	12348, 1-1-2023, 1-8-2023, 306000, 0001, 0001, 25.5
6	12349, 1-1-2023, 1-8-2023, 320000, 0001, 0001, 25.5
7	12350, 1-1-2023, 1-8-2023, 440000, 0001, 0001, 25.5
8	12351, 1-1-2023, 1-8-2023, 390000, 0001, 0001, 25.5
9	12352, 1-1-2023, 1-8-2023, 415000, 0001, 0001, 25.5
10	12353, 1-1-2023, 1-8-2023, 305500, 0001, 0001, 25.5
11	12354, 1-1-2023, 1-8-2023, 405000, 0001, 0001, 25.5
12	12355, 1-1-2023, 1-8-2023, 300000, 0001, 0001, 25.5
13	12356, 1-1-2023, 1-8-2023, 400000, 0001, 0001, 25.5
14	12357, 1-1-2023, 1-8-2023, 367000, 0001, 0001, 25.5
15	12358, 1-1-2023, 1-8-2023, 356000, 0001, 0001, 25.5
16	12359, 1-1-2023, 1-8-2023, 350000, 0001, 0001, 25.5
17	12360, 1-1-2023, 1-8-2023, 380000, 0001, 0001, 25.5
18	12361, 1-1-2023, 1-8-2023, 332000, 0001, 0001, 25.5
19	12362, 1-1-2023, 1-8-2023, 460000, 0001, 0001, 25.5
20	12363, 1-1-2023, 1-8-2023, 413000, 0001, 0001, 25.5

ภาพที่ 3.1 โครงสร้างไฟล์ข้อมูลแบบ CSV

รูปแบบไฟล์ AVRO

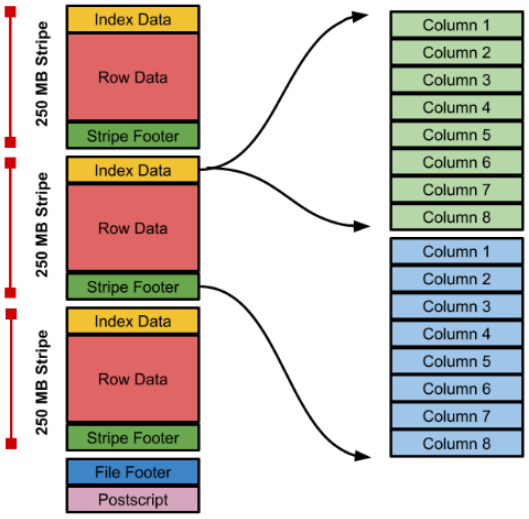
Vohra ได้คิดรูปแบบไฟล์ใหม่ โดยแยกส่วนการจัดเก็บเนื้อข้อมูล (Content) และส่วนคำอธิบายข้อมูล [4] (Schema) เรียกว่า AVRO โดยแยกการจัดเก็บข้อมูลเป็น 2 ส่วนอย่างชัดเจน โดยจัดเก็บข้อมูลอยู่ในรูปแบบไบนารี ซึ่งประหยัดเนื้อที่มากกว่าไฟล์แบบข้อความ และมีโครงสร้างการจัดเก็บข้อมูลแบบเรียงตามแถวเป็นหลัก (Row-based Data Structure) โดยใช้การเรียงลำดับข้อมูลไปทีละบล็อคตามลำดับ [5] AVRO เหมาะสำหรับการจัดเก็บข้อมูลที่ซับซ้อน (Nested Data) โดยใช้ระบบการจัดลำดับข้อมูลตามโครงสร้างข้อมูล ซึ่งส่วนของคำอธิยายข้อมูลใช้ไฟล์ข้อมูลแบบ JSON ในการจัดเก็บทำให้สามารถเข้าใจความหมายของข้อมูลได้ง่ายและยังสนับสนุนการแปลงชนิดข้อมูลระหว่างระบบฐานข้อมูลต้นทางกับระบบฐานข้อมูลปลายทาง ในการนำเข้าข้อมูล (Data Ingestion) มาจัดเก็บในแพลตฟอร์มบิ๊กดาต้าได้ง่าย ตัวอย่างโครงสร้างไฟล์ข้อมูลแบบ AVRO แสดงในภาพที่ 3.2



ภาพที่ 3.2 โครงสร้างไฟล์ข้อมูลแบบ AVRO

รูปแบบไฟล์ ORC

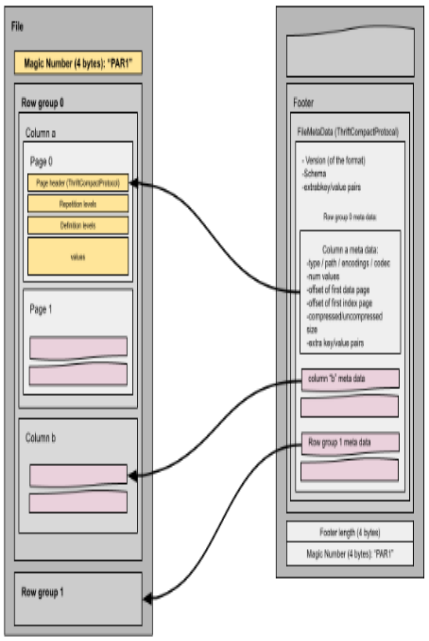
Optimized Row Columnar (ORC) เป็นรูปแบบไฟล์ ที่มีโครงสร้างการจัดเก็บข้อมูลแบบเรียงตามคอลัมน์เป็นหลัก (Column-based Data Structure) เพื่อเน้นประสิทธิภาพในการประมวลผลคำสั่งเรียกดูข้อมูลแบบ SQL และสามารถรองรับการบีบอัดข้อมูลให้การจัดเก็บข้อมูลมีขนาดเล็กลงได้ [6] ไฟล์ ORC ได้พัฒนาต่อจาก รูปแบบไฟล์ Record Columnar File (RCFile) ซึ่งมีโครงสร้างการจัดเก็บข้อมูลแบบเรียงตามคอลัมน์เช่นกัน โดยไฟล์ RCFile ถูกออกแบบมารองรับการประมวลผลแบบขนานด้วย MapReduce เป็นหลัก โดยการแบ่งข้อมูลออกเป็นส่วน ๆ เรียกว่า บล็อค (Block) กระจายไปจัดเก็บในหน่วยจัดเก็บข้อมูล (Storage) บนเครื่องคอมพิวเตอร์แม่ข่ายหลายเครื่อง ทำให้สามารถอ่านข้อมูลมาประมวลผลคำสั่งแบบ SQL พร้อม ๆ กันได้ [7] นอกจากส่วนของสไตรพ์แล้วไฟล์แบบ ORC ยังมีส่วนท้ายของไฟล์ (File Footer) ซึ่งมีคำอธิบายข้อมูล (Metadata) อยู่ภายใน ซึ่งสามารถปรับแก้โครงสร้างข้อมูลที่จัดเก็บไว้ภายหลังโดยไม่กระทบกับข้อมูลที่จัดเก็บไว้เดิมได้ ทำให้ไฟล์ ORC เหมาะกับการเชื่อมโยงข้อมูลที่อาจเปลี่ยนแปลงโครงสร้างข้อมูลได้ภายหลัง และส่วนสุดท้ายเรียกว่า โพสสคริปต์ (Postscript) เป็นส่วนที่ใช้ระบุขนาดของส่วนท้ายข้อมูล ขนาดของคำอธิบายข้อมูล วิธีการบีบอัดข้อมูล ที่สำคัญคือการเชื่อมโยงข้อมูลแต่ละสไตรพ์และช่วยให้การรวมข้อมูลทั้งหมดเข้าด้วยกันทำได้ง่าย โครงสร้างไฟล์ ORC แสดงในภาพที่ 3.3



ภาพที่ 3.3 โครงสร้างไฟล์ข้อมูลแบบ ORC

รูปแบบไฟล์ปาร์เก้ (Parquet)

รูปแบบไฟล์ปาร์เก้ (Parquet) ได้ถูกออกแบบและพัฒนาขึ้นโดย Cloudera ร่วมกับ Twitter ในปี ค.ศ. 2013 ซึ่งเป็นรูปแบบการจัดเก็บไฟล์บน HDFS ของ Apache Hadoop โดยมีโครงสร้างการจัดเก็บข้อมูลคล้ายกับแบบเรียงตามคอลัมน์เป็นหลัก (Column-Based Data Structure) แต่เป็นโครงสร้างข้อมูลแบบผสม (Hybrid Data Structure) โดยมีการเรียงข้อมูลตามคอลัมน์ติดกัน แต่มีการแยกส่วนเป็นบล็อค ซึ่งแต่ละบล็อคจะเป็นกลุ่มของแถวข้อมูล (Row Group) มีการจัดเก็บข้อมูลเป็นแบบไบนารี ซึ่งประหยัดเนื้อที่ในการจัดเก็บข้อมูลมากกว่าการจัดเก็บแบบข้อความ และสนับสนุนการบีบอัดข้อมูลได้หลายรูปแบบ เช่น Snappy, GZip และ LZO เป็นต้น ทำให้ไฟล์ปาร์เก้มีประสิทธิภาพทั้งการจัดเก็บข้อมูลและการเรียกค้นข้อมูล [8] โครงสร้างไฟล์ปาร์เก้ แสดงในภาพที่ 3.4



ภาพที่ 3.4 โครงสร้างไฟล์ข้อมูลแบบปาร์เก้ (Parquet)

งานวิจัยที่เกี่ยวข้อง

Ivanov และ Pergolesi [9] ได้ทำการศึกษาวิจัยเปรียบเทียบผลกระทบจากรูปแบบไฟล์ ที่ใช้การจัดเก็บข้อมูลแบบเรียงตามคอลัมน์เป็นหลัก (Column-Based Data Structure) ระหว่างไฟล์ ORC และไฟล์ปาร์เก้ (Parquet) โดยการทดสอบประสิทธิภาพด้วยการใช้คำสั่ง SQL เรียกดูข้อมูลในระบบฐานข้อมูลขนาดใหญ่ที่จัดเก็บบนแพลตฟอร์มที่พัฒนาจาก Apache Hadoop โดยคำสั่ง SQL ที่ใช้มีการเปรียบเทียบระหว่างข้อมูลในบริการ 2 ชนิด ได้แก่ Apache Hive [10] และ Apache Spark [11] ที่อยู่ใน Apache Hadoop โดย Hive เป็นบริการเรียกดูข้อมูลในระบบฐานข้อมูล ที่เน้นการจัดเก็บและประมวลผลแบบขนานบนหน่วยจัดเก็บข้อมูล (Storage) ที่มีขนาดใหญ่กว่าหน่วยความจำ (Memory) มาก โดยจะแปลงคำสั่ง SQL เป็น Map Reduce Job เพื่อแบ่งการประมวลผลเป็นส่วนย่อย เรียกว่าฟังก์ชัน Map และรวมผลลัพธ์ที่ประมวลผลเสร็จแล้วเรียกว่าฟังก์ชัน Reduce สำหรับบริการอีกชนิด เรียกว่า Spark ซึ่งเน้นการใช้หน่วยความจำมาประมวลผลข้อมูลเป็นหลัก ทำให้มีประสิทธิภาพสูงกว่า แต่ก็มีข้อจำกัดที่หน่วยความจำจะมีขนาดไม่มากเท่ากับหน่วยจัดเก็บข้อมูล จึงเหมาะกับการประมวลผลข้อมูลที่ไม่ใหญ่มากจนเกินไป โดยผลการทดลองชี้ให้เห็นว่าไฟล์ ORC จะมีประสิทธิภาพโดยรวมสูงกว่าไฟล์ปาร์เก้ เมื่อใช้การเรียกดูข้อมูลจาก Hive ส่วนไฟล์ปาร์เก้จะมีประสิทธิภาพสูงกว่าไฟล์ ORC เมื่อใช้การเรียกดูข้อมูลจาก Spark

4. วิธีดำเนินการศึกษา

ผลการทดลองเปรียบเทียบประสิทธิภาพของรูปแบบไฟล์ข้อมูล ทั้งหมด 4 รูปแบบ ได้แก่ รูปแบบไฟล์ Text (CSV.) รูปแบบไฟล์ AVRO รูปแบบไฟล์ ORC และรูปแบบไฟล์ Parquet ซึ่งไฟล์แต่ละรูปแบบที่นำมาทดลองได้มีการเตรียมชุดข้อมูลที่มีขนาดต่างกัน 3 ระดับ ได้แก่ ชุดข้อมูลขนาดเล็ก (Small Dataset) ที่มีข้อมูลเกิน 100,000 แถว แต่ไม่เกิน 500,000 แถว ชุดข้อมูลขนาดกลาง (Medium Dataset) ที่มีข้อมูลมากกว่า 600,000 แถว แต่ไม่เกิน 1,000,000 แถว และชุดข้อมูลขนาดใหญ่ (Large Dataset) ที่มีข้อมูลมากกว่า 1,000,000 แถว (ชุดข้อมูลที่นำมาทดลองได้มาจาก เว็บไซต์ https://www.kaggle.com/datasets/shivamb/netflix-shows) [12] โดยนำชุดข้อมูลที่เตรียมไว้แต่ละประเภทมาประมวลผลกับระบบฐานข้อมูลของ Apache Hive ซึ่งเป็นบริการที่รองรับการสร้าง (Create) ตารางข้อมูล การสืบค้นข้อมูล (Query) การแก้ไขข้อมูล (Update) และการลบข้อมูล (Delete) ซึ่งจัดเก็บอยู่ในแพลตฟอร์มบิ๊กดาต้า (Big Data Platform)

4.1 ซอฟต์แวร์ที่ใช้ในงานวิจัย

การติดตั้งซอฟต์แวร์ที่ใช้ในการพัฒนาแพลตฟอร์มบิ๊กดาต้า สำหรับเป็นเครื่องมือในการวิจัย ซึ่งเป็นบริการพื้นฐานของฮาร์ดูป (Hadoop foundation services) ทั้งนี้ซอฟต์แวร์ Cloudera ที่นำมาใช้นี้ เป็นซอฟต์แวร์ที่พัฒนามาจากฮาร์ดูป และได้รับการยอมรับในระดับโลกในการใช้พัฒนาเป็นแพลตฟอร์มบิ๊กดาต้า ซึ่งมีทั้งแบบที่มีลิขสิทธิ์และไม่มีลิขสิทธิ์ให้เลือกใช้ ดังนั้น ในการทดลองนี้คณะผู้วิจัยจึงเลือกใช้ซอฟต์แวร์ Cloudera แบบ Open source ที่ไม่มีลิขสิทธิ์ เพื่อใช้เป็นเครื่องมือสำหรับใช้ในการทดลองในงานวิจัยนี้เท่านั้น รายละเอียดการติดตั้งซอฟต์แวร์บนเครื่องคอมภาษาโปรแกรมที่ใช้ในงานวิจัย

นี้ คณะผู้วิจัยใช้ Structured Query Language (SQL) ซึ่งนำมาใช้งานบน Hadoop User Experience (Hue) ซึ่งเป็นเครื่องมือตัวนึงที่ช่วยในการนำเข้าข้อมูลของ Cloudera Data Platform (CDP) และใช้เป็นเครื่องมือในการเตรียมข้อมูลในการทดลองและหาผลการทดลองในงานวิจัยนี้ (Query) รายละเอียดการติดตั้งซอฟต์แวร์บนเครื่องคอมพิวเตอร์แม่ข่าย แสดงในภาพที่ 4.1

ลำดับ	เครื่องคอมพิวเตอร์แม่ข่าย	ระบบปฏิบัติการ	บริการพื้นฐาน
1	Master Node	CentOS 7.9	- Cloudera Manager - HDFS NameNode - Hue - Hive Metastore - Hive Server 2 - YARN Resource Manager
2	Data Node 1	CentOS 7.9	- HDFS DataNode - YARN Node Manager - ZooKeeper
3	Data Node 2	CentOS 7.9	- HDFS DataNode - YARN Node Manager - ZooKeeper
4	Data Node 3	CentOS 7.9	- HDFS DataNode - YARN Node Manager - ZooKeeper

ภาพที่ 4.1 ซอฟต์แวร์บนเครื่องคอมพิวเตอร์แม่ข่าย

4.2 ผลการทดลอง

4.2.1 การเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการสร้างตารางข้อมูลใหม่ การทดลองที่ 1 เป็นการเปรียบเทียบประสิทธิภาพ ในการสร้างตารางข้อมูลใหม่ (Create a New Table) ของไฟล์ทั้ง 4 รูปแบบ ซึ่งในแต่ละแบบมีขนาดต่างกัน 3 ระดับ เพื่อหาว่าไฟล์รูปแบบใดเหมาะสมกับการสร้างตารางข้อมูลใหม่มากที่สุด (ใช้เวลาในการสร้างตารางใหม่น้อยที่สุด) โดยการประมวลผลนี้จะใช้ Apache Hive ในการสร้างตารางข้อมูลขึ้นมาใหม่ด้วยภาษา SQL ได้แก่

```
CREATE TABLE <table_name> (
        <type1> <column1>;
        <type2> <column2>;
        ...
) STORED AS <file_format>;
```

ในส่วนของรูปแบบไฟล์ที่จัดเก็บจะแยกแยะด้วย STORED AS <file_format> ได้แก่ คำสั่ง STORED AS TEXTFILE สำหรับไฟล์แบบข้อความที่ใช้ CSV ไฟล์ในการจัดเก็บ คำสั่ง STORED AS AVRO สำหรับไฟล์ AVRO คำสั่ง

STORED AS ORC สำหรับไฟล์ ORC และคำสั่ง STORED AS PARQUET สำหรับไฟล์ปาร์เก้ โดยมีผลการทดลองแสดงในภาพที่ 4.2

No.	TEXT			AVRO			ORC			PARQUET		
	S	M	L	S	M	L	S	M	L	S	M	L
1	21.49	25.85	46.50	19.69	21.16	43.36	20.88	30.91	45.20	28.26	27.63	93.00
2	19.21	25.68	46.78	18.82	20.40	45.61	20.16	31.11	46.86	27.36	27.79	94.00
3	19.39	25.53	47.34	18.97	20.63	44.55	22.51	30.96	45.56	27.18	27.81	93.00
4	19.22	25.69	47.65	18.90	20.87	44.81	20.77	30.96	45.20	27.22	27.54	94.00
5	20.03	25.89	46.78	19.08	20.70	45.07	20.23	30.92	45.22	27.36	27.65	94.00
6	20.17	25.78	46.78	19.79	20.63	44.30	21.24	31.13	45.30	27.55	27.70	93.00
7	19.87	25.53	46.80	18.85	21.12	44.35	21.25	30.93	46.32	28.16	27.79	94.00
8	19.21	25.85	46.78	18.98	21.15	43.81	20.83	30.98	46.70	28.19	27.63	93.00
9	20.11	25.57	47.30	19.12	20.92	45.32	22.45	31.11	46.63	27.90	27.56	93.00
10	20.12	25.68	47.25	18.90	20.70	45.18	22.34	31.10	46.69	27.90	27.77	93.00
Min	19.21	25.53	46.50	18.82	20.40	43.36	20.16	30.91	45.20	27.18	27.54	93.00
Max	21.49	25.89	47.65	19.79	21.16	45.61	22.51	31.13	46.86	28.26	27.81	94.00
Ave	19.88	25.71	47.00	19.11	20.83	44.64	21.27	31.01	45.97	27.71	27.69	93.40

ภาพที่ 4.2 ผลการเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการสร้างตารางข้อมูลใหม่

4.2.2 การเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการอ่านข้อมูลทั้งตาราง การทดลองที่ 2 เป็นการเปรียบเทียบประสิทธิภาพ ในการอ่านข้อมูลที่จัดเก็บในตารางทั้งหมด (Select All Data) ของไฟล์ทั้ง 4 รูปแบบ ซึ่งแต่ละแบบมีขนาดต่างกัน 3 ระดับ เพื่อหาว่าไฟล์รูปแบบไหนเหมาะสมกับการอ่านข้อมูลทั้งหมดที่จัดเก็บในตาราง (อ่านทุกแถวและทุกคอลัมน์) มากที่สุด (ใช้เวลาในการอ่านข้อมูลน้อยที่สุด) โดยการประมวลผลนี้จะใช้ Apache Hive ในการอ่านข้อมูลด้วยภาษา SQL ได้แก่

```
SELECT *
FROM <table_name>;
```

ในส่วนของการประมวลผลนี้ จะใช้เครื่องมือ Hive Query ผ่านบริการ Apache Hue ซึ่งเป็นเครื่องมือสนับสนุนการบริหารจัดการข้อมูลขนาดใหญ่บนแพลตฟอร์มบิ๊กดาต้า สำหรับคำสั่งที่ทดสอบข้างต้นนั้น

จะคล้ายกันสำหรับทุกชุดข้อมูลที่รับการทดสอบ ยกเว้นชื่อของตารางข้อมูล (Data Table) ที่ถูกเรียกมาประมวลผล โดยมีผลการทดลองแสดงในภาพที่ 4.3

No.	TEXT			AVRO			ORC			PARQUET		
	S	M	L	S	M	L	S	M	L	S	M	L
1	00.22	00.25	00.21	00.19	00.21	00.18	00.21	00.31	00.31	00.18	00.17	00.18
2	00.20	00.22	00.22	00.14	00.16	00.20	00.22	00.23	00.24	00.15	00.20	00.21
3	00.18	00.18	00.22	00.16	00.20	00.20	00.23	00.28	00.21	00.20	00.21	00.20
4	00.22	00.24	00.21	00.15	00.21	00.19	00.23	00.30	00.30	00.16	00.18	00.21
5	00.20	00.20	00.21	00.16	00.17	00.20	00.24	00.29	00.28	00.17	00.18	00.21
6	00.19	00.18	00.22	00.14	00.17	00.20	00.22	00.29	00.27	00.15	00.19	00.23
7	00.19	00.19	00.23	00.19	00.18	00.19	00.21	00.25	00.27	00.17	00.20	00.19
8	00.20	00.18	00.23	00.19	00.17	00.20	00.24	00.28	00.28	00.16	00.19	00.20
9	00.21	00.18	00.23	00.15	00.16	00.19	00.23	00.28	00.29	00.16	00.18	00.20
10	00.19	00.19	00.22	00.15	00.16	00.20	00.24	00.27	00.30	00.17	00.20	00.21
Min	00.18	00.18	00.21	00.14	00.16	00.18	00.21	00.23	00.21	00.15	00.17	00.18
Max	00.22	00.25	00.23	00.19	00.21	00.20	00.24	00.31	00.31	00.20	00.21	00.23
Ave	00.20	00.20	00.22	00.16	00.18	00.20	00.23	00.28	00.28	00.17	00.19	00.20

ภาพที่ 4.3 ผลการเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการอ่านข้อมูลทั้งตาราง

4.2.3 การเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการนับจำนวนแถวในตาราง การทดลองที่ 3 เป็นการเปรียบเทียบประสิทธิภาพ ในการนับจำนวนแถวข้อมูลที่จัดเก็บในตารางทั้งหมด (Count the Number of Rows) ซึ่งเป็นการคำนวณเชิงสถิติพื้นฐานอย่างหนึ่ง โดยนำไฟล์ทั้ง 4 รูปแบบ มาทดสอบในแพลตฟอร์ม ซึ่งแต่ละแบบมีขนาดต่างกัน 3 ระดับ เพื่อหาว่าไฟล์รูปแบบไหนเหมาะสมกับการนับจำนวนแถวข้อมูลในตารางมากที่สุด (ใช้เวลาในการอ่านข้อมูลน้อยที่สุด) โดยการประมวลผลนี้จะใช้ Apache Hive ในการอ่านข้อมูลด้วยภาษา SQL ได้แก่

```
SELECT COUNT(*)
FROM <table_name>;
```

ในส่วนของการทดสอบนี้จะต่างจากคำสั่ง SELECT * FROM <table_name> ที่ใช้ในการทดลองก่อนหน้าในข้อ 4.2 เนื่องจาก คำสั่ง count(*) จะให้ผลลัพธ์เป็นจำนวนเต็มของแถวในตารางข้อมูล แทนที่จะเป็นการแสดงผลข้อมูลทั้งหมด โดยมีผลการทดลองแสดงในภาพที่ 4.4

No.	TEXT			AVRO			ORC			PARQUET		
	S	M	L	S	M	L	S	M	L	S	M	L
1	00.26	00.35	30.98	00.29	0.330	40.99	00.29	00.26	24.70	00.31	00.27	34.40
2	00.33	00.30	31.70	00.29	0.360	37.40	00.27	00.28	26.26	00.32	00.31	36.93
3	00.38	00.33	30.89	00.28	0.370	37.32	00.26	00.26	24.98	00.29	00.32	35.90
4	00.35	00.34	30.95	00.29	0.342	38.12	00.28	00.27	24.88	00.30	00.29	35.46
5	00.37	00.34	31.12	00.29	0.340	38.02	00.28	00.28	24.98	00.30	00.30	35.42
6	00.36	00.30	31.16	00.28	0.336	38.78	00.29	00.26	25.13	00.29	00.30	35.47
7	00.27	00.31	31.24	00.29	0.338	37.65	00.27	00.27	24.78	00.31	00.27	35.71
8	00.27	00.32	31.11	00.30	0.339	38.26	00.27	00.27	24.77	00.31	00.28	35.63
9	00.28	00.31	31.19	00.31	0.342	38.12	00.28	00.26	25.08	00.29	00.28	35.42
10	00.27	00.30	31.15	00.31	0.338	38.66	00.27	00.27	24.94	00.31	00.27	35.66
Min	00.26	00.30	30.89	00.28	00.33	37.32	00.26	00.26	24.70	00.29	00.27	34.40
Max	00.38	00.35	31.70	00.31	00.37	40.99	00.29	00.28	26.26	00.32	00.32	36.93
Ave	00.31	00.32	31.15	00.29	00.34	38.33	00.28	00.27	25.05	00.30	00.29	35.60

ภาพที่ 4.4 ผลการเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการนับจำนวนแถวในตาราง

4.2.4 การเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการเรียงลำดับข้อมูลในตาราง การทดลองที่ 4 เป็นการเปรียบเทียบประสิทธิภาพ ในการเรียงลำดับข้อมูลที่จัดเก็บในตารางทั้งหมด (Sort the Data) ซึ่งเป็นการคำนวณเชิงสถิติที่ใช้การประมวลผลสูงกว่าการนับจำนวนแถวในตาราง โดยนำไฟล์ทั้ง 4 รูปแบบ มาทดสอบในแพลตฟอร์ม ซึ่งแต่ละแบบมีขนาดต่างกัน 3 ระดับ เพื่อหาว่าไฟล์รูปแบบไหน เหมาะสมกับการเรียงลำดับข้อมูลในตารางมากที่สุด (ใช้เวลาในการอ่านข้อมูลน้อยที่สุด) โดยการประมวลผลนี้จะใช้ Apache Hive ในการอ่านข้อมูลด้วยภาษา SQL ได้แก่

```
SELECT <column 1>, <column 2>, ..., <column N>
FROM <table_name>
ORDER BY <colum_name>;
```

ในส่วนของการทดสอบนี้ จะใช้เวลาในการประมวลผล

ข้อมูลมากที่สุดเมื่อเทียบกับการทำงานประเภทอื่นที่ผ่านมาทั้งหมด เนื่องจากเป็นชุดคำสั่งที่มีความซับซ้อนในการประมวลผลมากกว่าการใช้ชุดคำสั่งหรือการทำงานประเภทอื่นก่อนหน้านี้ โดยมีผลการทดลองแสดงในภาพที่ 4.5

No.	TEXT			AVRO			ORC			PARQUET		
	S	M	L	S	M	L	S	M	L	S	M	L
1	00.32	00.43	02.55	00.31	00.53	02.55	00.31	00.43	02.35	00.29	00.37	02.24
2	00.31	00.42	02.56	00.30	00.53	02.58	00.30	00.43	02.33	00.30	00.40	02.27
3	00.32	00.43	02.61	00.31	00.48	02.59	00.29	00.43	02.41	00.29	00.40	02.25
4	00.32	00.44	02.57	00.31	00.52	02.56	00.29	00.43	02.38	00.29	00.39	02.25
5	00.31	00.45	02.57	00.30	00.52	02.56	00.30	00.44	02.37	00.29	00.39	02.25
6	00.34	00.44	02.55	00.33	00.53	02.55	00.30	00.44	02.37	00.30	00.38	02.26
7	00.32	00.43	02.60	00.32	00.49	02.57	00.29	00.43	02.36	00.29	00.40	02.25
8	00.31	00.43	02.60	00.33	00.50	02.57	00.31	00.43	02.40	00.30	00.39	02.24
9	00.31	00.42	02.60	00.32	00.49	02.58	00.30	00.43	02.39	00.29	00.37	02.24
10	00.33	00.44	02.61	00.33	00.50	02.59	00.31	00.44	02.37	00.29	00.37	02.25
Min	00.31	00.42	02.55	00.30	00.48	02.55	00.29	00.43	02.33	00.29	00.37	02.24
Max	00.34	00.45	02.61	00.33	00.53	02.59	00.31	00.44	02.41	00.30	00.40	02.27
Ave	00.32	00.43	02.58	00.32	00.51	02.57	00.30	00.43	02.37	00.29	00.39	02.25

ภาพที่ 4.5 ผลการเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ในการเรียงลำดับข้อมูลในตาราง

4.2.5 สรุปผลการทดลอง

ในการสร้างตารางข้อมูลขึ้นมาใหม่ด้วยภาษา SQL ผลการทดลองสรุปได้ว่ารูปแบบไฟล์ชนิด AVRO มีค่าเฉลี่ยของเวลาดีที่สุดจากการทดลองสร้างข้อมูลทั้ง 3 ขนาด เล็ก, กลาง, ใหญ่

ในการอ่านตารางข้อมูลด้วยภาษา SQL ผลการทดลองสรุปได้ว่ารูปแบบไฟล์ชนิด AVRO มีค่าเฉลี่ยของเวลาดีที่สุดจากการทดลองอ่านข้อมูลทั้ง 3 ขนาด เล็ก, กลาง, ใหญ่

No.	Detail	TEXT			AVRO			ORC			PARQUET		
		S	M	L	S	M	L	S	M	L	S	M	L
1	CREATE TABLE												
AVRO	*Min*	19.21	25.53	46.50	18.82	20.40	43.36	20.16	30.91	45.20	27.18	27.54	93.00
	Max	21.49	25.89	47.65	19.79	21.16	45.61	22.51	31.13	46.86	28.26	27.81	94.00
	Ave	19.88	25.71	47.00	19.11	20.83	44.64	21.27	31.01	45.97	27.71	27.69	93.40
2	SELECT *												
AVRO	*Min*	00.18	00.18	00.21	00.14	00.16	00.18	00.21	00.23	00.21	00.15	00.17	00.18
	Max	00.22	00.25	00.23	00.19	00.21	00.20	00.24	00.31	00.31	00.20	00.21	00.23
	Ave	00.20	00.20	00.22	00.16	00.18	00.20	00.23	00.28	00.28	00.17	00.19	00.20
3	SELECT COUNT(*)												
ORC	*Min*	00.26	00.30	30.89	00.28	00.33	37.32	00.26	00.26	24.70	00.29	00.27	34.40
	Max	00.38	00.35	31.70	00.31	00.37	40.99	00.29	00.28	26.26	00.32	00.32	36.93
	Ave	00.31	00.32	31.15	00.29	00.34	38.33	00.28	00.27	25.05	00.30	00.29	35.60
4	ORDER BY												
PARQUET	*Min*	00.31	00.42	02.55	00.30	00.48	02.55	00.29	00.43	02.33	00.29	00.37	02.24
	Max	00.34	00.45	02.61	00.33	00.53	02.59	00.31	00.44	02.41	00.30	00.40	02.27
	Ave	00.32	00.43	02.58	00.32	00.51	02.57	00.30	00.43	02.37	00.29	00.39	02.25

ภาพที่ 4.6 ผลการเปรียบเทียบประสิทธิภาพรูปแบบไฟล์ข้อมูลต่าง ๆ ในตาราง

ในการนับจำนวนแถวในตารางข้อมูลด้วยภาษา SQL ผลการทดลองสรุปได้ว่ารูปแบบไฟล์ชนิด ORC มีค่าเฉลี่ยของเวลาดีที่สุดจากการทดลองในการนับจำนวนแถวในตารางข้อมูลทั้ง 3 ขนาด เล็ก, กลาง, ใหญ่

ในการเรียงลำดับข้อมูลในตารางด้วยภาษา SQL ผลการทดลองสรุปได้ว่ารูปแบบไฟล์ชนิด PARQUET มีค่าเฉลี่ยของเวลาดีที่สุดจากการทดลองในการเรียงลำดับข้อมูลในตารางทั้ง 3 ขนาด เล็ก, กลาง, ใหญ่ ผลการทดลองแสดงในภาพที่ 4.6

4.3 ฮาร์ดแวร์ที่ใช้ในงานวิจัย

ฮาร์ดแวร์ที่ใช้ในงานวิจัยประกอบด้วย เครื่องคอมพิวเตอร์ Workstation ประสิทธิภาพสูง จำนวน 1 เครื่อง เพื่อใช้เป็น Master Node สำหรับเป็นส่วนติดต่อกับผู้ใช้และกำหนดงานไปประมวลผลบน Data Node และใช้เครื่องคอมพิวเตอร์เสมือน (Virtual Machine: VM) สำหรับ Data Node จำนวน 3 เครื่อง สำหรับจัดเก็บและประมวลข้อมูลแบบขนาน (Parallel computing)

แสดงในภาพที่ 4.7

ลำดับ	เครื่องคอมพิวเตอร์แม่ข่าย	CPU	Memory	Disk
1	Master Node	8 cores	32 GB	1 TB
2	Data Node 1	8 cores	16 GB	1 TB
3	Data Node 2	8 cores	16 GB	1 TB
4	Data Node 3	8 cores	16 GB	1 TB

ภาพที่ 4.7 คุณลักษณะทางเทคนิคของเครื่องคอมพิวเตอร์แม่ข่ายสำหรับแพลตฟอร์มบิ๊กดาต้า

5. ผลการศึกษา

จากผลการทดลอง เพื่อเปรียบเทียบประสิทธิภาพในการทำงานกับไฟล์รูปแบบต่าง ๆ ที่นิยมใช้ในแพลตฟอร์มบิ๊กดาต้า ที่มีการจัดเก็บและประมวลผลข้อมูลแบบกระจาย (Distributed Data Storing and Computing) สามารถแบ่งปัจจัยที่มีผลต่อประสิทธิภาพการประมวลผลข้อมูลในแพลตฟอร์มดังกล่าว ได้เป็น 2 กลุ่ม คือ ปัจจัยที่มีผลกระทบจากโครงสร้างการจัดเก็บข้อมูล และปัจจัยที่มีผลกระทบจากขนาดของข้อมูลที่จัดเก็บ

5.1 ปัจจัยที่มีผลกระทบจากโครงสร้างในการจัดเก็บข้อมูล

จากผลการทดลองประมวลผลข้อมูลกับการทำงานที่ต่างกันข้างต้น บนแพลตฟอร์มบิ๊กดาต้า จะเห็นได้ว่าโครงสร้างข้อมูลแต่ละแบบมีทั้งข้อดีและข้อเสียแตกต่างกัน ขึ้นกับการทำงานที่เกี่ยวข้องกับข้อมูลในขณะนั้น เช่น หากเป็นการสร้างตารางข้อมูลใหม่หรือการแทรกข้อมูลแถวใหม่เข้าไปในไฟล์ข้อมูลเดิม โครงสร้างการจัดเก็บข้อมูลแบบเรียงตามแถวเป็นหลัก (Row-based Data Structure) เช่น ไฟล์แบบข้อความ (CSV หรือ JSON) และไฟล์แบบ AVRO จะมีประสิทธิภาพสูงกว่าโครงสร้างการจัดเก็บข้อมูลแบบเรียงตามคอลัมน์เป็นหลัก (Column-based Data Structure) เช่น ไฟล์ ORC และไฟล์ปาร์เก้ เนื่องจากโครงสร้างการจัดเก็บข้อมูลแบบเรียงตามแถวเป็นหลักจะนำข้อมูลแถวใหม่ไปต่อท้ายข้อมูลแถวสุดท้ายในตารางเดิมเท่านั้น ในทางตรงกันข้ามถ้าเป็นข้อมูลแบบเรียงตามคอลัมน์เป็นหลักจะทำได้ยากกว่าในกรณีนี้ เพราะต้องนำข้อมูลในแต่ละคอลัมน์ของแถวใหม่ไปแทรกในส่วนท้ายของแต่ละคอลัมน์นั้น ๆ นอกจากนี้การแสดงผลข้อมูลทั้งหมดในตารางซึ่งเป็นการอ่านข้อมูลทุกคอลัมน์ในแต่ละแถวตามลำดับ โครงสร้างการจัดเก็บข้อมูลแบบเรียงตามแถวเป็นหลัก ซึ่งในที่นี้ได้แก่ไฟล์ AVRO และไฟล์ข้อความ จะเหมาะสมกว่าไฟล์ ORC และไฟล์ปาร์เก้ ที่มีโครงสร้างข้อมูลแบบเรียงตามคอลัมน์เป็นหลักเช่นกัน ในทางตรงกันข้ามการประมวลผลข้อมูลที่ไม่ได้อ่านหรือเขียนข้อมูลทุกแถวแบบตามลำดับ ไฟล์ ORC และไฟล์ปาร์เก้ ซึ่งเป็นโครงสร้างข้อมูลแบบเรียงตามคอลัมน์เป็นหลัก จะมีประสิทธิภาพที่สูงกว่าไฟล์แบบข้อความและไฟล์ AVRO ซึ่งเป็นโครงสร้างการจัดเก็บข้อมูลแบบเรียงตามแถวเป็นหลัก โดยเฉพาะการคำนวณทางคณิตศาสตร์หรือค่าสถิติพื้นฐาน เช่น การนับจำนวน (Count) การหาผลรวม (Sum) การหาค่าสูงสุด (Max) การหาค่าต่ำสุด (Min) และ การหาค่าเฉลี่ย (Average) ของข้อมูลเฉพาะบางคอลัมน์ยิ่งกว่านั้นถ้าเป็นการประมวลผลข้อมูลที่ซับซ้อนกว่าการคำนวณค่าสถิติพื้นฐาน เพื่อหาผลลัพธ์ตามที่ต้องการ เช่น การเรียงลำดับข้อมูล (Sort) ตามค่าที่กำหนด จะยิ่งเห็นความแตกต่างของประสิทธิภาพการประมวลผลข้อมูล ว่าโครงสร้างไฟล์แบบเรียงตามคอลัมน์นั้นเหมาะสมกับการทำงานประเภทนี้มากกว่าโครงสร้างไฟล์แบบเรียงตามแถวมากยิ่งขึ้น

5.2 ปัจจัยที่มีผลกระทบจากขนาดของข้อมูลที่จัดเก็บ

นอกจากประเภทของโครงสร้างไฟล์ข้อมูลที่มีการเรียงลำดับต่างกันแล้ว ขนาดของข้อมูลที่จัดเก็บบนแพลตฟอร์มบิ๊กดาต้าก็มีผลกระทบต่อประสิทธิภาพในการทำงานกับข้อมูลเช่นกัน จากผลการทดลองนี้ เราจะเห็นว่าการสร้างตารางจากไฟล์ข้อมูลที่มีขนาดใหญ่ (Large data) จะมีผลกระทบต่อประสิทธิภาพของไฟล์ปาร์เก้สูงมากกว่าไฟล์ข้อมูลรูปแบบอื่น เนื่องจากไฟล์ปาร์เก้จะเสียเวลาในการแปลงข้อมูลที่จัดเก็บจากข้อความ (Text) ให้เป็นข้อมูลแบบไบนารี (Binary) ซึ่งมี

ข้อดีที่ใช้เนื้อที่ในการจัดเก็บน้อยกว่าแบบข้อความมาก แต่ผลกระทบของไฟล์ขนาดใหญ่จะทำให้เห็นข้อเสียของไฟล์ปาร์เก้ในการสร้างไฟล์ใหม่ได้อย่างชัดเจน นอกจากผลกระทบในการสร้างไฟล์ใหม่หรือการเขียนข้อมูลลงบนไฟล์แล้ว ขนาดของไฟล์ขนาดใหญ่ยังมีผลกระทบกับไฟล์ที่มีโครงสร้างข้อมูลแบบเรียงตามคอลัมน์เป็นหลักเหมือนกันจากผลการนับจำนวนแถวในไฟล์ขนาดเล็กและไฟล์ขนาดกลางไฟล์ปาร์เก้ได้ผลต่างจากไฟล์ ORC เล็กน้อย แต่เมื่อไฟล์มีขนาดใหญ่ขึ้นไฟล์ ORC จะมีประสิทธิภาพดีที่สุดกับการคำนวณด้านสถิติพื้นฐานเนื่องจากไฟล์ ORC มีฟังก์ชันในการคำนวณค่าสถิติเหล่านั้นไว้อยู่แล้ว ทำให้ใช้เวลาในการประมวลผลน้อยกว่าไฟล์ชนิดอื่น อย่างไรก็ตาม เมื่อเป็นการประมวลผลที่ไม่เกี่ยวกับการคำนวณค่าสถิติพื้นฐาน แต่เป็นการประมวลผลข้อมูลที่ซับซ้อนเช่น การเรียงลำดับข้อมูลทั้งหมดในไฟล์ขนาดใหญ่ ไฟล์ปาร์เก้กลับมีประสิทธิภาพดีกว่าไฟล์ ORC ที่มีโครงสร้างข้อมูลแบบเรียงตามคอลัมน์เป็นหลักเหมือนกัน เนื่องจากการประมวลผลแบบนี้ไม่สามารถนำค่าทางสถิติพื้นฐานที่มีมาใช้ได้โดยตรง แต่ต้องใช้การประมวลผลข้อมูลทั้งหมดมาแสดงผลลัพธ์ ซึ่งโครงสร้างไฟล์ปาร์เก้ทำได้ดีกว่าเนื่องจากไฟล์ปาร์เก้ใช้การจัดเก็บแบบผสมทั้งแบบเรียงตามคอลัมน์และเรียงตามแถว นอกจากนี้การจัดเก็บแบบไบนารียังช่วยให้การเคลื่อนย้ายข้อมูลหรือการอ่านข้อมูลมาประมวลผลมีประสิทธิภาพดียิ่งขึ้น

6. สรุปผลและอภิปรายผล

งานวิจัยนี้เป็นการเปรียบเทียบประสิทธิภาพของรูปแบบไฟล์ข้อมูล เพื่อวิเคราะห์หารูปแบบไฟล์ที่เหมาะสมกับข้อมูลในการทำงานประเภทต่าง ๆ โดยเฉพาะข้อมูลขนาดใหญ่ ที่มีผลกระทบสูงต่อประสิทธิภาพในการประมวลผลข้อมูลบนแพลตฟอร์มบิ๊กดาต้า (Big Data Platforms) ในการทดลองนี้ได้นำรูปแบบไฟล์ข้อมูลที่นิยมใช้บนแพลตฟอร์มบิ๊กดาต้ามากที่สุดในปัจจุบัน โดยสามารถแบ่งรูปแบบไฟล์ข้อมูลได้เป็น 2 กลุ่ม ได้แก่ กลุ่มไฟล์ที่มีโครงสร้างการจัดเก็บแบบเรียงตามแถวเป็นหลัก (Row-Based Data Structure) ซึ่งคณะผู้วิจัยได้นำไฟล์ข้อความ (Text) และไฟล์ AVRO มาใช้ในการทดลอง และกลุ่มไฟล์ข้อมูลที่มีโครงสร้างการจัดเก็บแบบเรียงตามคอลัมน์เป็นหลัก (Column-Based Data Structure) โดยเรานำไฟล์ ORC และไฟล์ปาร์เก้ มาใช้เปรียบเทียบประสิทธิภาพกับไฟล์กลุ่มแรก นอกจากนี้ในการทดสอบเพื่อเปรียบเทียบประสิทธิภาพ คณะผู้วิจัยได้ทดสอบการทำงานกับข้อมูลที่หลากหลาย เช่น การสร้างตารางข้อมูลใหม่ การอ่านข้อมูลทั้งหมดในตารางการนับจำนวนแถวข้อมูลในตาราง ซึ่งเป็นเป็นการคำนวณสถิติพื้นฐาน และการเรียงลำดับข้อมูลที่มีการประมวลผลซับซ้อน โดยการทดสอบจะนำไฟล์ข้อมูลที่มีขนาดต่างกัน 3 ระดับ มาใช้ประเมินผล เพื่อพิจารณาปัจจัยจากขนาดของไฟล์ข้อมูลว่ามีผลต่อประสิทธิภาพการทำงานแต่ละประเภทหรือไม่ จากผลการทดลองสรุปได้ว่าการทำงานที่เกี่ยวกับการอ่านหรือเขียนข้อมูลทั้งหมดในตาราง แบบตามลำดับ ไฟล์ข้อมูลที่มีโครงสร้างการจัดเก็บแบบเรียงตามแถวเป็นหลัก จะมีประสิทธิภาพดีกว่าไฟล์ข้อมูลที่มีโครงสร้างการจัดเก็บแบบเรียงตามคอลัมน์เป็นหลัก ส่วนการทำงานที่ใช้การคำนวณเชิงสถิติ หรือการประมวลผลข้อมูลที่มีความซับซ้อนมากขึ้น ไฟล์ข้อมูลที่มีโครงสร้างการจัดเก็บแบบเรียงตามคอลัมน์เป็นหลัก จะมีประสิทธิภาพที่ดีกว่าไฟล์ข้อมูลที่มีโครงสร้างการจัดเก็บแบบเรียงตามแถวเป็นหลัก นอกจากนี้ผลการทดลองยังพบว่า ขนาดของข้อมูลที่ใหญ่ขึ้นส่งผลกระทบต่อประสิทธิภาพการทำงานกับข้อมูลด้วย โดยเฉพาะการสร้างตารางใหม่ไฟล์ปาร์เก้จะมีประสิทธิภาพต่ำที่สุดและมีความแตกต่างจากไฟล์ประเภทอื่นอย่างมีนัยสำคัญ

7. กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จลุล่วงไปด้วยดีเนื่องจากคณะผู้วิจัยได้รับความช่วยเหลือดูแลเอาใจใส่เป็นอย่างดีจากหลาย ๆ ฝ่ายโดยเฉพาะอาจารย์ที่ปรึกษาคือ ผศ.ดร.วรภัทร ไพรีเกรง ในการแนะนำ ตรวจแก้ไข ให้ข้อเสนอแนะ ติดตามความก้าวหน้าในการดำเนินการวิจัย คณะผู้วิจัยรู้สึกซาบซึ้งในความกรุณาของอาจารย์เป็นอย่างยิ่ง และขอขอบพระคุณเป็นอย่างสูงไว้ ณ โอกาสนี้

ขอขอบคุณ คณะกรรมการและเจ้าหน้าที่กองทุนพัฒนาโรงเรียนนายร้อยพระจุลจอมเกล้าทุกท่านที่กรุณาช่วย

พิจารณากลั่นกรองโครงการวิจัยและช่วยประสานงานต่าง ๆ ให้สำเร็จลุล่วงไปได้ด้วยดี และขอขอบคุณกองทุนพัฒนาโรงเรียนนายร้อยพระจุลจอมเกล้า ที่สนับสนุนงบประมาณในการดำเนินการวิจัย ประจำปี 2565

ขอขอบคุณผู้เชี่ยวชาญที่สละเวลาในการตรวจทานแก้ไขข้อบกพร่องของงานวิจัย เรื่องการวิจัยการเปรียบเทียบประสิทธิภาพของพื้นที่จัดเก็บในแพลตฟอร์มบิ๊กดาต้า ตรวจทานความถูกต้องของภาษาและพิจารณาความเชิงตรงเนื้อหาของเครื่องมือที่ใช้ในงานวิจัย

นอกจากนี้คณะผู้วิจัยยังได้รับการช่วยเหลืออีกจำนวนมากที่ผู้วิจัยไม่สามารถกล่าวนามได้หมดในที่นี้ ผู้วิจัยรู้สึกซาบซึ้งในความกรุณาและความปรารถนาดีของทุกท่านเป็นอย่างยิ่งจึงกราบขอบพระคุณและขอบคุณไว้โอกาสนี้

8. บรรณานุกรม

[1] Cutting, D., & Cafarella, M. (2007). Apache hadoop., 203-214.

[2] Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. Communications of the ACM, 51(1), 107-113.

[3] Shafranovich, Y. (2005). Rfc 4180: Common format and mime type for comma-separated values (csv) files., 562-604.

[4] Pezoa, F., Reutter, J. L., Suarez, F., Ugarte, M., & Vrgoč, D. (2016, April). Foundations of JSON schema. In Proceedings of the 25th international conference on World Wide Web (pp. 263-273).

[5] Vohra, D. (2016). Apache avro. Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools, 303-323.

[6] Apache, O. R. C. (2018). Apache ORC: High-Performance Columnar Storage for Hadoop., 67-108.

[7] He, Y., Lee, R., Huai, Y., Shao, Z., Jain, N., Zhang, X., & Xu, Z. (2011). RCFile: A fast and space-efficient data placement structure in MapReduce-based warehouse systems. In 2011 IEEE 27th International Conference on Data Engineering (pp. 1199-1208). IEEE.

[8] Vohra, D., & Vohra, D. (2016). Apache parquet. Practical Hadoop Ecosystem: A Definitive Guide to Hadoop-Related Frameworks and Tools, 325-335.

[9] Ivanov, T., & Pergolesi, M. (2020). The impact of columnar file formats on SQL-on-hadoop engine performance: A study on ORC and Parquet. Concurrency and Computation: Practice and Experience, 32(5), e5523.

[10] Bansal, H., Chauhan, S., & Mehrotra, S. (2016). Apache Hive Cookbook. Packt Publishing Ltd.

[11] Salloum, S., Dautov, R., Chen, X., Peng, P. X., & Huang, J. Z. (2016). Big data analytics on Apache Spark. International Journal of Data Science and Analytics, 1, 145-164.

[12] Anonymous.: Dataset web services. Available: https://www.kaggle.com/datasets/shivamb/netflix-shows. February 5, 2024