การวิเคราะห์ประสิทธิภาพการประมวลผลของ Hive-QL ด้วย ORCfile บน Hadoop Cluster

Analyzing the Query Performance of Hive-QL with ORCfile on Hadoop Cluster

พันธิการ์ วัฒนกุล[1] กฤษณ์วรา รัตนโอภาส[2 *] และ สุรีรัตน์ แก้วคีรี[3]

[1] โปรแกรมวิชาคอมพิวเตอร์ธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยราชภัฏนครปฐม
85 ถนนมาลัยแมน ตำบลนครปฐม อำเภอเมือง จังหวัดนครปฐม 73000

[2] โปรแกรมวิชาคอมพิวเตอร์ คณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสงขลา

[3] โปรแกรมวิชาคอมพิวเตอร์ธุรกิจ คณะวิทยาการจัดการ มหาวิทยาลัยราชภัฏสงขลา
160 ถนนกาญจนวนิช ตำบลเขารูปช้าง อำเภอเมือง จังหวัดสงขลา 90000

Phantiga Wattanakul[1] Kritwara Rattanaopas[2*] and Sureerat Keawkeeree[3]

[1] Department of Business Computer, Faculty of Management Science, Nakhon Pathom Rajabhat University
85 Malaiman Road, Muang, Nakhon Pathom, Thailand, 73000

[2] Computer Department, Faculty of Science and Technology, Songkhla Rajabhat University

[3] Department of Business Computer, Faculty of Management Science, Songkhla Rajabhat University
160 Karnjanawanich Road, Kaoroopchang District, Ampur Muang, Songkhla, Thailand, 90000

[*] ผู้นิพนธ์ประสานงาน: kritwara.ra@skru.ac.th เบอร์โทรศัพท์ 089-733-3779

บทคัดย่อ

ข้อมูลทางอุตุนิยมวิทยาถือเป็นข้อมูลที่สำคัญในการนำมาประมวลผลเพื่อการพยากรณ์ ด้วยแนวคิดของการทำ Big data ที่มีความนิยมในปัจจุบันและโครงสร้าง Hadoop Cluster ที่มีโปรแกรม Hive สำหรับประมวลผลแบบฐานข้อมูลเชิงสัมพันธ์ ผู้วิจัยจึงมีแนวคิดในการศึกษาปัจจัยที่ทำให้การประมวลผลดังกล่าวมีประสิทธิภาพสูงขึ้น ด้วยสมมติฐานการเพิ่มขึ้นของ จำนวน Data node และจำนวนการสำเนาของชุดข้อมูล โดยผลการวิจัยพบว่า ค่าของ Map-Reduce ที่ถูกกำหนดด้วยโปรแกรม Hive เมื่อทำการประมวลผลมีผลกระทบต่อประสิทธิภาพในการประมวลผล ด้วยค่า Map=5/Reduce=1 สอดคล้องกับจำนวน Data node ที่ดีที่สุดคือ 5 Data node กับสำเนาข้อมูล 3 ชุด หากมีการเพิ่มจำนวน Data node และจำนวนสำเนาข้อมูล พบว่าไม่มีผลกระทบต่อเวลาในการประมวลผลและทำให้มีเวลาที่สูงขึ้นในทุกกรณีของชุดคำสั่ง SQL

คำสำคัญ ข้อมูลปริมาณมาก โปรแกรมโอเพ่นซอร์สจัดเก็บข้อมูลขนาดใหญ่ ระบบคลังข้อมูล การวัดประสิทธิภาพ

Abstract

The weather forecast data is one of the most important datasets in Big Data. The Hive application was the first relational database that runs on Hadoop cluster. This paper presents a performance analysis of HiveQL on Hadoop cluster with varying number of data node and data replication. The results show that the best performing Map-Reduce configuration for distributed nodes in Hadoop cluster is Map=5/Reduce=1. This ratio is consistent with the best query performance setup which is 3 replications per 5 data nodes. Meanwhile, increasing the number of data nodes and replications did not affect the result in anyway.

Keywords: Big Data, Hadoop, Hive, performance analysis.

1. บทนำ

ข้อมูลสารสนเทศที่ถูกบันทึกในแต่ละกิจกรรมในชีวิตประจำวันจากการใช้งานระบบอินเตอร์เน็ต หรือ ข้อมูลอื่นๆ ที่มีการเก็บเพื่อใช้นำมาประมวลผล ได้แก่ ข้อมูลธนาคาร ข้อมูลการซื้อขายสินค้า ข้อมูลเครือข่ายสังคมออนไลน์ เป็นต้น เหล่านี้ล้วนมีปริมาณข้อมูลจำนวนมากมาย โดยการนิยามภาพลักษณะของคำว่า “Big data” [1] กล่าวไว้ด้วยข้อมูลทางสถิติที่สามารถสะท้อนภาพถึงปริมาณข้อมูลขนาดใหญ่ หากเรารู้จักเว็บที่มีทั่วไปมากกว่า 100 พันล้านที่มีอัตราการเพิ่มขึ้นในแต่ละวันสูงกว่า 20,000 เว็บไซต์ และในกรณีของ Facebook ที่มีการโพสมากว่า 300 พันล้านครั้ง และการอัพเดตกว่า 60 พันล้านครั้ง เมื่อพิจารณาถึงปริมาณข้อมูลที่ไม่มากนักในแต่ละครั้งกลับกลายเป็นข้อมูลขนาดมหาศาล ด้วยเทคโนโลยีของการประมวลผลข้อมูลปริมาณมากนี้ จึงจำเป็นต้องใช้เครื่องมือที่เหมาะสม เพราะการใช้ระบบจัดการฐานข้อมูลเชิงสัมพันธ์ที่มีอยู่ในอดีตอาจไม่ใช่คำตอบ แต่ปัจจุบันมีเครื่องมือในการจัดการข้อมูลลักษณะ Big data นั้นคือ โปรแกรม Hadoop ที่มีความสามารถในการขยายเครื่องลูกข่ายหรือโหนดเก็บข้อมูลตามต้องการร่วมกับโปรแกรม Hive ที่สามารถนำข้อมูลเชิงสัมพันธ์มาประมวลผลด้วยกระบวนการ Map-Reduce ที่ได้รับความนิยมในปัจจุบัน เพื่อประมวลผลข้อมูลแบบ Big data ด้วยตัวอย่างข้อมูลการวัดอุณหภูมิจากสถานีวัดอุณหภูมิจากเว็บไซต์ open-big data [2] ภายใต้โครงสร้างของ Hadoop ที่แตกต่างกัน ซึ่งผู้วิจัยต้องการนำเสนอประสิทธิภาพของการประมวลผลข้อมูลแบบ Big data ด้วยจำนวนโหนดที่แตกต่างกันและจำนวนการสำเนาข้อมูลที่แตกต่างกันบนหน่วยบันทึกข้อมูลของ Hadoop

ดังนั้นเนื้อหาของงานวิจัยส่วนต่อไปกล่าวถึงทฤษฎีและงานวิจัยพื้นฐานที่เกี่ยวข้องกับ Hadoop และ Hive พร้อมด้วยวิธีการดำเนินการวิจัยพร้อมแนวคิดในการออกแบบการทดสอบ สำหรับส่วนท้ายกล่าวถึงผลและสรุปการวิจัย

2. ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

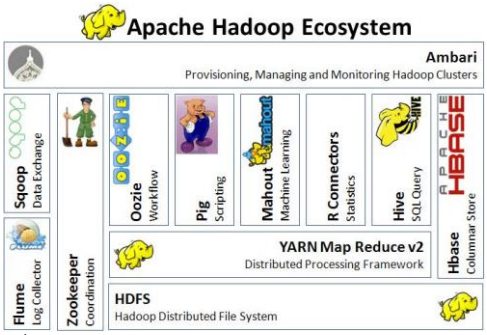
งานวิจัยนี้นำเทคโนโลยี Apach Hadoop และ Apache Hive ที่ได้รับการพัฒนาอย่างต่อเนื่อง มาประมวลผลข้อมูลขนาดใหญ่อย่าง Big data จึงขอแนะนำโครงสร้างและความสามารถของโปรแกรมในเนื้อหาส่วนนี้ พร้อมงานวิจัยในลักษณะเดียวกันที่นำ Hadoop มาใช้ในการวิเคราะห์ข้อมูลในปัจจุบัน

2.1 งานวิจัยที่เกี่ยวข้อง

การศึกษาวิจัยด้าน Big data ด้วยโปรแกรม Hadoop มีความนิยมแพร่หลายในปัจจุบัน ด้วยงานวิจัยของ Daniel Abadi 0 ได้เขียนแนวทางในการใช้ภาษา SQL บนโครงสร้างของ Hadoop พร้อมด้วยงานของ K. Jayasri [4] ที่วัดประสิทธิภาพการประมวลผลเช่นเดียวกันแต่เน้นในส่วนของการทดสอบบน NoSQL ด้วยโปรแกรม Mongodb กับข้อมูลที่เกี่ยวข้องกับภูมิศาสตร์ ด้วยวิธีการ SPARQL query บนโครงสร้างของโปรแกรม Hadoop สำหรับงานวิจัยในประเทศไทยมีการนำมาใช้ในด้านการเรียนรู้ด้วยคอมพิวเตอร์ (Machine learning) บน HDFS แบบ Hadoop cluster ของอดิศร อดิศรพันธ์กุล และศุภฤกษ์ มานิตพรสุทธ์ [4] ด้วยเครื่อง Name node จำนวน 1 เครื่อง และเครื่องData node จำนวน 3 เครื่อง ซึ่งมีการเชื่อมต่อเครือข่ายที่ความเร็วเพียง 100 Mbps ด้วยภาระงานการนับคำ พบว่า การเพิ่ม Data nodeในการประมวลผลไม่ช่วยในการเพิ่มประสิทธิภาพ แต่ในงานวิจัยนี้ เลือกใช้ภาระงานของฐานข้อมูลอุณหภูมิร่วมกับโปรแกรม Hive เพื่อนำเสนอประสิทธิภาพการประมวลผลของ Map-Reduce บนโครงสร้าง Hadoop Cluster ในลักษณะเดียวกัน

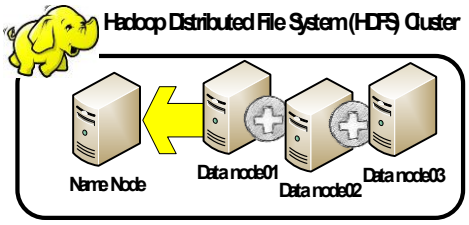
2.2. Hadoop

Hadoop ถือเป็นโอเพ่นซอร์สที่ได้รับความนิยมสำหรับการนำมาประมวลผลข้อมูล และกล่าวถึงกันในกลุ่ม Big data ถูกออกแบบใช้งานบนระบบปฏิบัติการลินุกส์หรือระบบปฏิบัติการวินโดว์ ลักษณะการทำงานแบบอิสระเพียงคัดลอกไว้ในเครื่องที่ต้องการ และมี Java เป็นตัวสนับสนุนเบื้องหลัง เพื่อสร้างเป็นกลุ่มคลัสเตอร์ได้ง่าย โดยมีการพัฒนากระบวนการที่เรียกว่า Map-Reduce เพื่อช่วยประมวลผลโปรแกรมหรือข้อมูลในรูปแบบกระจายสำหรับอุปกรณ์คอมพิวเตอร์ทั่วไปตามรูปที่ 1



รูปที่ 1 สถาปัตยกรรมของกลุ่มโปรแกรม Hadoop [6]

จากสถาปัตยกรรมมีโปรแกรมจำนวนมากที่รองรับบริการ บน Apache Hadoop Ecosystem ได้แก่ Oozie Pig Hive และ Mahout ซึ่งสามารถใช้โปรแกรม R Connectors เพื่อประมวลผลข้อมูลทางสถิติด้วยภาษา R โดยในงานวิจัยนี้เลือกโปรแกรม Hive ที่สนับสนุน SQL query เพื่อนำมาใช้ประมวลผลข้อมูลด้วยโครงสร้างของ HDFS Cluster ตามรูปที่ 2 ที่มีการให้บริการหลักด้วย Name node เพื่อบริหารระบบ HDFS Cluster จากเครื่อง Data node ที่สามารถเพิ่มจำนวนได้ตามต้องการในรูปแบบกระจายภาระงานตามรูปที่ 2 ซึ่งโครงสร้างของ HDFS Cluster สามารถเพิ่มเครื่อง Data node ได้ตามความต้องการและสามารถสร้างเครื่อง Name node แบบ Secondary หรือ Name node สำรองเพื่อเพิ่มความเสถียรภาพให้กับ HDFS Cluster



รูปที่ 2 โครงสร้างของ HDFS Cluster

โดย HDFS Cluster มีคุณสมบัติการสำเนาข้อมูล (Replication) ซึ่งสามารถสำเนาข้อมูลได้หลายชุดแบบกระจายไปตาม Data node มีค่าตั้งต้นที่ 3 ชุด ซึ่งหากมีจำนวน Data node น้อยกว่าค่าสำเนา ระบบ HDFS Cluster จะบันทึกเท่าจำนวนของ Data node ขณะนั้น หากมีการเพิ่ม Data node ใหม่เข้าสู่คลัสเตอร์ ระบบจะเพิ่มการสำเนาทันที จึงสามารถกล่าวได้ว่ามีความเสถียรภาพสูงในการบันทึกข้อมูลและการกู้คืนหากมี Data node เสียหายไป

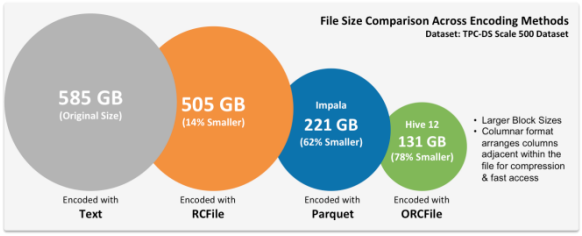
2.3 Hive

โปรแกรมนี้ถูกออกแบบให้ติดตั้งแบบอิสระเหมือนโปรแกรม Hadoop เพื่อสนับสนุนการทำงานของ Structured Query Language (SQL) ซึ่งยังคงได้รับความนิยมในปัจจุบันและถือเป็นความท้าทายในการพัฒนาโปรแกรมให้สนับสนุนการทำงาน โดยสามารถสนับสนุนคำสั่งพื้นฐานอย่าง “SELECT...WHERE” “SELECT...ORDER BY” “GROUP BY” และ “JOIN” พร้อมด้วยคำสั่งเกี่ยวกับการคำนวณอย่าง ผลรวม คือ sum( ) ค่าเฉลี่ย คือ avg( ) นับจำนวน คือ count( ) ค่าน้อยที่สุด คือ min( ) และ ค่าสูงที่สุด คือ max( ) โดยสามารถกระจายงานในลักษณะ Map-Reduce และถูกกำหนดชื่อเรียกว่า Hive Query Language (ย่อว่า HiveQL หรือ HQL) เพื่อเรียกหรือค้นหาข้อมูลภายใน HDFS Cluster จึงสามารถรองรับข้อมูลขนาดใหญ่และมีความสามารถยืดหยุ่นกว่าฐานข้อมูลเชิงสัมพันธ์อย่าง MySQL และ MariaDB ด้วยโครงสร้างการเก็บข้อมูลของ HDFS Cluster ที่ประกอบด้วย Data node สำหรับจัดเก็บข้อมูลและกระจายการประมวลผล

พื้นฐานของโปรแกรม Hadoop Hive ทำงานด้วยส่วนที่เหมือนกับ MySQL ที่มีชื่อว่า Hive console แต่หากต้องการใช้งาน Hive จากเครื่องภายนอกมีส่วนเสริมที่สามารถสั่งให้ทำงานแทนที่ Hive console ที่มีชื่อว่า “hive.server2” หรือที่เรียกว่า “Thrift Hive Server” ซึ่งมีความสามารถในการเชื่อมต่อผ่าน Java Database Connectivity (JDBC) ทำให้สามารถเข้าถึงจากโปรแกรมภายนอก

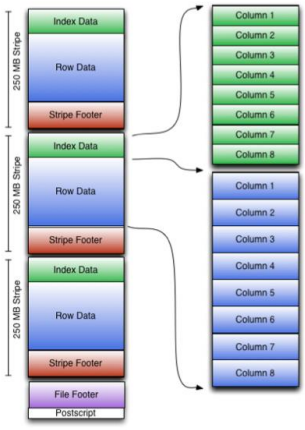
2.4 Optimized Record Columnar File (ORCFile)

ทีมพัฒนา Hadoop ได้เพิ่มขั้นตอนวิธีการบีบอัดข้อมูลเพื่อช่วยลดปริมาณของเนื้อที่การเก็บข้อมูล เนื่องจากเมื่อออกแบบให้รองรับข้อมูลขนาดใหญ่ (Big data) มีความจำเป็นต้องมีพื้นที่รองรับการเก็บข้อมูลดังกล่าวได้ โดยชุดบีบอัดข้อมูลที่ถูกนำเสนอได้แก่ Zlib LZO และ Snappy เป็นต้น ชุดบีบอัดข้อมูลดังกล่าวถูกพัฒนาร่วมกับรูปแบบการเก็บข้อมูล ซึ่งในช่วงเริ่มต้นการพัฒนา Hive ได้นำเสนอตารางแบบ Record Columnar File (RCFile) ที่มีประสิทธิภาพเพียง 14% ของการลดขนาดพื้นที่เก็บข้อมูล ต่อมาได้มีการปรับแต่งจนกลายเป็น Optimized Row Columnar File (ORCFile) ที่ลดพื้นที่การเก็บข้อมูลได้สูงถึง 78% ตามรูปที่ 3



รูปที่ 3 ตัวอย่างรูปแบบไฟล์สำหรับเก็บข้อมูลบน Hive [7]

สำหรับในงานวิจัยนี้ผู้วิจัยเลือกรูปแบบการเก็บข้อมูลไฟล์แบบ ORCFile สำหรับตารางข้อมูลบน Hive [8] จาก 3 รูปแบบคือ Text RCFile และ ORCFile ที่มีประสิทธิภาพสูงสุด โดยมีโครงสร้างของการเก็บข้อมูลที่ถูกเรียกว่า Strips ที่มีค่าเริ่มต้นที่ 250 MB ละการปรับเปลี่ยนขนาดมีผลต่อประสิทธิภาพการอ่านบน HDFS ที่มีค่าของ Block size และมีคุณสมบัติการบีบอัดข้อมูลด้วยขั้นตอนวิธีแบบ Snappy ที่นำมาใช้เก็บข้อมูลและใช้ประมวลผลด้วยภาระงานสำหรับวัดประสิทธิภาพการประมวลผลแบบ Map-Reduce บน Hadoop Cluster



รูปที่ 4 โครสร้างแบบ Strip [9]

3. วิธีการดำเนินการวิจัย

การออกแบบในการวัดประสิทธิภาพของการประมวลผลด้วยโปรแกรม hive ด้วยสมมติฐานการทดสอบประสิทธิภาพของการประมวลผลที่ควรแปรผันตรงกับ “จำนวน Data node และการสำเนาข้อมูล (Replication)” ซึ่งผู้วิจัยเลือกข้อมูลจากตัวอย่างข้อมูลการวัดอุณหภูมิของสถานีอุตุนิยมวิทยา จากเว็บไซต์ “www.open-bigdata.com” [2] เพื่อสร้างข้อมูลขนาดใหญ่หรือ Big data ที่มีขนาด 207,466,688 ระเบียนด้วยวิธีการสำเนาซ้ำ 32 รอบจากตัวอย่างข้อมูล 6,483,334 ระเบียน ที่ประกอบด้วยโครงสร้างข้อมูล รหัสสถานี (Stationcode) ชื่อสถานี (station) วันที่ (datefield) อุณหภูมิสูงสุด(tmax) และอุณหภูมิต่ำสุด(tmin) ตามตารางที่ 1 ในไฟล์รูปแบบ ORC ที่บีบอัดข้อมูลด้วยวิธีการ SNAPPY เพื่อนำมาประมวลผลด้วย SQL Query ที่ออกแบบในการทดสอบด้วยโครงสร้างของระบบในรูปที่ 5

ตารางที่ 1 ตารางข้อมูลของสถานีอุตุนิยมวิทยา

Field	Stationcode (Primary Key)	station	datefield	tmax	tmin
Type	STRING	STRING	Date/time	DOUBLE	DOUBLE

3.1 โครงสร้างในการทดสอบ

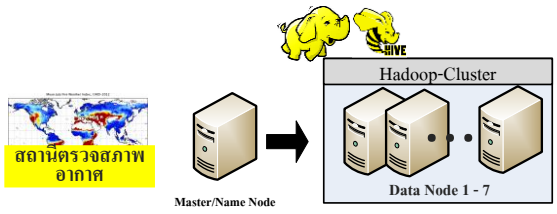
การติดตั้ง Hadoop เพื่อให้บริการ HDFS และ Map-Reduce ทำการทดสอบด้วยเทคโนโลยีเวอร์ชวลไลเซชั่น เพื่อสร้างเครื่องเสมือนจริงที่ได้รับความนิยมในปัจจุบันที่มีการบริการด้วย Cloud computing โดยมีเครื่องคอมพิวเตอร์ที่มีคุณสมบัติหน่วยประมวลผลกลางแบบ Intel i7 หน่วยความจำ 4GB หน่วยบันทึกข้อมูล 1TB และ การ์ดเครือข่ายแบบ 1 Gbps Ethernet จำนวน 4 เครื่อง เพื่อติดตั้งโปรแกรม Hadoop และ Hive แบบคลัสเตอร์ พร้อมโปรแกรม Java openJDK รุ่น 1.8 เพื่อรองรับการทำงาน บนระบบปฏิบัติการ CentOS 7.2 แบบ 64 bits ทั้งเครื่องเสมือนจริงที่เป็น Name node และ Data node ที่บรรจุไว้ภายในเครื่องแม่ข่ายที่ใช้ทดสอบด้วยระบบปฏิบัติการ CentOS 7.2 แบบ 64 bits และโปรแกรมบริหารจัดการเครื่องเสมือนจริง Kernel-based Virtual Machine (KVM) เพื่อใช้บริหารจัดการเครื่องเสมือนจริงภายในตามรูปที่ 5

การติดตั้ง Hadoop cluster ที่ประกอบด้วย Name node/Master node และ Data node ต้องมีซอฟต์แวร์ขั้นพื้นฐานในงานวิจัยนี้พร้อมคำสั่งติดตั้งดังนี้

- ระบบปฏิบัติการ CentOS 6.8 แบบ 64 bit สำหรับทั้ง Name node และ Data node
- Java openjdk รุ่น 1.8.0 : “sudo yum install java-1.8.0-openjdk* -y” พร้อมกำหนดค่าในไฟล์ “.bashrc” โดยระบุ “export JAVA_HOME=/usr/lib/jvm/jre-1.8.0-openjdk” สำหรับทั้ง Name node และ Data node

- Hadoop 2.6.1 ที่ดาวน์โหลดจากเว็บไซต์ http://apache.arvixe.com/hadoop/common/hadoop-2.6.1/hadoop-2.6.1.tar.gz โดยทำการติดตั้งทั้ง Name node และ Data node
- Hive รุ่น 1.2.2 ดาวน์โหลดจากเว็บไซต์ http://www.eu.apache.org/dist/hive/hive-1.2.2/apache-hive-1.2.2-bin.tar.gz ซึ่งมีวามแตกต่างจากตัวโปรแกรม Hadoop โดยติดตั้งไว้เฉพาะที่ Name node เท่านั้น

หลังการติดตั้งโปรแกรมพื้นฐานทั้งหมดแล้วการสร้างกลุ่ม Hadoop cluster ด้วยการกำหนดค่าในไฟล์ slave ของ Name node ด้วยชื่อของ Data node เดียวกับในไฟล์ Host และต้องทำคำสั่ง “hdfs namenode -format” เพื่อให้ล้างค่าและจัดรูปแบบข้อมูลบน Name node ก่อน จากนั้นสั่งเริ่มการทำงานของ Hadoop cluster ด้วยคำสั่ง “start-all.sh”



รูปที่ 5 โครงสร้างสำหรับการวิจัย

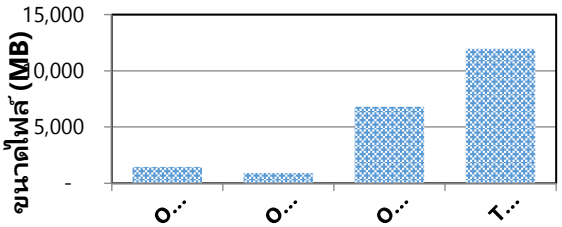
สำหรับโครงสร้างทดสอบตามรูปที่ 5 ประกอบด้วยเครื่องเสมือนจริงที่มีคุณสมบัติ 2 vCPU RAM 2 GB HD 60 GB จำนวน 8 เครื่อง ประกอบด้วยเครื่องเสมือนจริง Name node 1 เครื่อง และ เครื่องเสมือนจริง Data node 7 เครื่อง กำหนดรูปแบบการทดสอบเริ่มจาก Name node 1 เครื่อง และ Data node 3 เครื่อง โดยปรับเพิ่ม Data node ครั้งละ 2 เครื่อง เนื่องจากเครื่อง Host บรรจุ Data node ได้ 2 เครื่อง ควบคู่ไปกับการกำหนดการสำเนา (Replication-Rep) ที่ 3, 5 และ 7 ไปพร้อมกันตามตารางที่ 6 เพื่อเปรียบเทียบประสิทธิภาพของ Data node และการสำเนาข้อมูล ซึ่งผลเวลาเฉลี่ยของการประมวลผลจะถูกนำเสนอในหัวข้อผลวิจัยต่อไป

การทำงานของ Hive ในงานวิจัยนี้เลือกทดสอบผ่าน Hive console ด้วยคำสั่ง “hive” บน Name node โดยพื้นที่เก็บข้อมูลของ Hive ถูกสร้างไว้บน HDFS ของ Hadoop cluster ที่ “/user/hive/warehouse” ด้วยคำสั่ง “hadoop fs –mkdir -p”

สำหรับพื้นฐานของ Hive-QL มีฐานข้อมูลเริ่มต้นชื่อว่า default ซึ่งการสร้างตารางมีรูปแบบเหมือนกับ SQL เช่น “CREATE EXTERNAL TABLE weather1 (stationcode STRING, station STRING, datefield STRING, prcp DOUBLE, tmax DOUBLE, tmin DOUBLE) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';);” แต่มีความพิเศษของรูปแบบส่วนท้ายคือ “FORMAT DELIMITED FIELDS TERMINATED BY ','” เนื่องจากตารางนี้เป็นข้อมูลเริ่มต้นที่จะนำเข้าจากไฟล์ในรูปแบบ CSV หากเชื่อมต่อโดยตรงจาก Java Database Connectivity (JDBC) สามารถสร้างตารางแบบ ORC ได้ทันที ในกรณีที่นำเข้าไฟล์ CSV ไฟล์ดังกล่าวต้องนำเข้าไปวางใน HDFS ของ Hadoop cluster และใช้คำสั่ง “LOAD DATA INPATH '/user/hdfs/Weather.csv' OVERWRITE INTO TABLE weather1;” เมื่อได้ตารางข้อมูลพื้นฐานนี้ การเพิ่มประสิทธิภาพ หรือการสร้างตารางแบบ ORC ทำได้โดยคำสั่งต่อไปนี้

- ORC แบบ none : “CREATE EXTERNAL TABLE weather_orc0 1 (stationcode STRING, station STRING, datefield) STORED AS ORC TBLPROPERTIES ("orc.compress"="**NONE**") ;”
- ORC แบบ snappy : “CREATE EXTERNAL TABLE weather_orc0 1 (stationcode STRING, station STRING, datefield) STORED AS ORC TBLPROPERTIES ("orc.compress"=" **snappy**") ;”
- ORC แบบ zlib : “CREATE EXTERNAL TABLE weather_orc0 1 (stationcode STRING, station STRING, datefield) STORED AS ORC TBLPROPERTIES ("orc.compress"=" **zlib**") ;”

การสร้างตารางแบบ ORC file ผู้วิจัยทำการสร้างไฟล์ต้นฉบับที่มีขนาด 12 GB และสร้างตาราง ORC file ทั้ง 3 แบบตามรูปที่ 6



รูปที่ 6 ขนาดไฟล์ของตารางข้อมูลบน Hive

จากรูปที่ 6 สำหรับพื้นที่จัดเก็บข้อมูลไฟล์แบบ ORC ทั่วไปสามารถลดพื้นที่จัดเก็บได้ถึง 50% และชุดบีบอัดที่เลือกแบบ Snappy สามารถลดขนาดพื้นที่เหลือเพียง 12% ของไฟล์ข้อมูลปกติที่จำนวนข้อมูล 207,466,688 ระเบียน เพื่อนำไปวัดประสิทธิภาพของการประมวลผลต่อไปด้วย HiveQL เพื่อหาค่าเฉลี่ย ค่าสูงสุด รวมถึงการเชื่อมตารางสำหรับประมวลผลข้อมูลต่อไป และผู้วิจัยได้ตั้งสมมติฐานของจำนวนการสำเนาข้อมูลไว้ตามตารางที่ 6 บนไฟล์ HDFS-site.xml ที่มีคุณสมบัติที่ชื่อว่า “dfs.replication” ที่ปรับค่า value ตามต้องการ และสามารถตรวจสอบการสำเนาข้อมูลไฟล์บน HDFS Cluster ด้วยคำสั่ง “hadoop dfs -ls” จะปรากฏหมายเลขการสำเนาหน้าชื่อไฟล์นั้นๆ

3.2 ภาระงานสำหรับการวิจัย

ตัวอย่างข้อมูลภาระงานที่นำมาใช้ในการประมวลผลแบบ Big data เพื่อมาทดสอบด้วย SQL ที่มีคำสั่ง “SELECT.. FROM..WHERE” และเลือกใช้การคำนวณ avg( ) max( ) พร้อมคำสั่ง “GROUP BY” และการ JOIN จากข้อมูลผลลัพธ์ในชุดก่อนหน้า โดยกำหนดการทดสอบด้วย 4 กรณีตามตารางที่ 1-4

กรณีที่ 1 CASE01:

ตารางที่ 2 ชุดคำสั่งของ CASE 01

CASE 01	- SELECT station, datefield, tmax FROM weatherorc WHERE tmax <> -9999 ORDER BY tmax DESC;

กำหนดชุดคำสั่งที่มีการเรียงข้อมูลกลับจากมากไปหาน้อยของชุดข้อมูลทั้งหมด เพื่อทดสอบประสิทธิภาพของการเรียงข้อมูลจำนวนมาก โดยให้ผลลัพธ์ตามตารางที่ 3

ตารางที่ 3 ตารางข้อมูลของ CASE 01

Field	station	datefield	tmax
Type	STRING	Date/time	DOUBLE

กรณีที่ 2 CASE02:

ตารางที่ 4 ชุดคำสั่งของ CASE 02

CASE 02	- INSERT OVERWRITE TABLE weather_temp2 SELECT station, cast(SUBSTR(datefield,1,4) as INT), tmax FROM weatherorc WHERE tmax <> -9999; - SELECT station, datefield, MAX(tmax) FROM weather_temp2 GROUP BY station, datefield;

มีขั้นตอนคือทำการกรองข้อมูลเข้าสู่ตาราง “weather_temp2” จำนวน 67,167,328 ระเบียนและนำมาประมวลผลเพื่อหาค่าสูงที่สุด (Max temperature) ในแต่ละปีแยกตามสถานีตรวจอากาศด้วยคำสั่ง Group by จะได้ผลลัพธ์ตามโครงสร้างตารางที่ 5

ตารางที่ 5 ตารางข้อมูลของ weather_temp2

Field	station	datefield	tmax
Type	STRING	INT	DOUBLE

กรณีที่ 3 CASE03:

ตารางที่ 6 ชุดคำสั่งของ CASE 03

CASE 03	- INSERT OVERWRITE TABLE weather_temp3 SELECT station, cast(SUBSTR(datefield,1,4) as INT), tmax FROM weatherorc WHERE tmax <> -9999; - SELECT station, datefield, AVG(tavg) FROM weather_temp3 GROUP BY station, datefield;

มี 2 ขั้นตอนด้วยกันคือ กรองข้อมูลเข้าสู่ตาราง “weather_temp3” แล้วจึงนำมาหาค่าเฉลี่ยของสถานีตรวจอากาศแยกตามปี เพื่อทดสอบการประมวลผลทางสถิติด้วยการคำนวณค่าเฉลี่ย และนำไปเก็บที่ตารางที่ 7 เพื่อศึกษาความสามารถของการหาค่าเฉลี่ย

ตารางที่ 7 ตารางข้อมูลของ weather_temp3

Field	station	datefield	tmax
Type	STRING	INT	DOUBLE

กรณีที่ 4 CASE04:

ตารางที่ 8 ชุดคำสั่งของ CASE 04

CASE 04	- INSERT OVERWRITE TABLE weather4_temp SELECT station, cast(SUBSTR(datefield,1,4) as INT), tmax, tmin FROM weatherorc WHERE tmax > -9999 and tmin > -9999; - INSERT OVERWRITE TABLE weather4_temp2 SELECT station, year, AVG((tmax-tmin)/2) FROM weather4_temp GROUP BY station, year; - INSERT OVERWRITE TABLE weather4_temp3 SELECT station, MAX(meanTemp) FROM weather4_temp2 GROUP BY station; INSERT OVERWRITE TABLE weather_res4 SELECT w.station, w.year, ww.maxMeanTemp FROM weather4_temp2 w JOIN weather4_temp3 ww ON w.meanTemp = ww.maxMeanTemp;

ถือว่าเป็นกรณีที่ซับซ้อนมากที่สุดเพราะใช้วิธีกรองข้อมูลแยกเป็นรายปีพร้อมหาค่าเฉลี่ยของผลต่างอุณหภูมิสูงและต่ำที่สุดบันทึกในตาราง “weather4_temp2” แล้วจึงหาค่าเฉลี่ยสูงสุดของทุกสถานีบันทึกใน “weather4_temp3” เพื่อนำมาเชื่อมข้อมูลระหว่าง 2 ตารางด้วยการ JOIN เพื่อนำเสนอข้อมูลของสถานีในแต่ละปี ด้วยค่าสูงสุดของอุณหภูมิเฉลี่ยที่เกิดจากการนำเอาอุณหภูมิสูงสุดและต่ำสุดมาเฉลี่ยกัน เพื่อบันทึกในตารางที่ 9

ตารางที่ 9 ตารางข้อมูลของ weather_res4

Field	station	year	maxMeanTemp
Type	STRING	INT	DOUBLE

ในส่วนที่ 2 ผู้วิจัยพิจารณาความแตกต่างของประสิทธิภาพได้อย่างชัดเจน โดยแยกตาม Query ที่เลือกจากชุดคำสั่งของ SQL ที่สำคัญเพื่อนำเสนอเวลาประมวลผลของ Query ดังกล่าวไว้ดังนี้

1) SELECT station, datefield, ***MAX(tmax)*** FROM **weather_temp2** *GROUP BY station, datefield;*
2) SELECT station, datefield, ***AVG(tavg)*** FROM **weather_temp3** *GROUP BY station, datefield;*
3) SELECT station, year, ***AVG((tmax-tmin)/2)*** FROM **weather4_temp** *GROUP BY station, year;*
4) SELECT w.station, w.year, ww.maxMeanTemp FROM ***weather4_temp2 w JOIN weather4_temp3 ww*** *ON w.meanTemp = ww.maxMeanTemp;*

การทดสอบวัดค่าการประมวลผลแบ่งตามจำนวนเครื่อง Data node และการสำเนาข้อมูลไว้ 5 แบบตามตารางที่ 6 ดังนี้

ตารางที่ 10 รูปแบบของ Hadoop Cluster

ชื่อแบบ	การกำหนดค่า
3 node(Rep-3)	มี Data node 3 เครื่องและกำหนดสำเนาข้อมูล 3 ชุด
5 node(Rep-3)	มี Data node 5 เครื่องและกำหนดสำเนาข้อมูล 3 ชุด
5 node(Rep-5)	มี Data node 5 เครื่องและกำหนดสำเนาข้อมูล 5 ชุด
7 node(Rep-3)	มี Data node 7 เครื่องและกำหนดสำเนาข้อมูล 3 ชุด
7 node(Rep-7)	มี Data node 7 เครื่องและกำหนดสำเนาข้อมูล 7 ชุด

* Rep คือ การสำเนาข้อมูล (Replication)

โดยมีสมมติฐานในการสำเนาข้อมูลให้มีจำนวนคงที่ และเพิ่มเท่ากับจำนวน Data node เช่น ในกรณี 5 Data node มีการสำเนาที่ 3 และ 5 ชุด เช่นเดียวกับกรณี 7 Data node มีการสำเนาที่ 3 และ 7 ชุด เพื่อต้องการศึกษาปัจจัยของการสำเนาข้อมูลนั้นมีผลกระทบต่อการประมวลผลข้อมูลหรือไม่ ซึ่งการเพิ่มการสำเนาต้องใช้พื้นที่ในการเก็บข้อมูลเพิ่มขึ้น

การเพิ่มชุดการสำเนาทำการกำหนดค่าเฉพาะที่ Name node เท่านั้น ซึ่งค่าดังกล่าวเก็บไฟล์ในไฟล์ hdfs-site.xml ด้วยคุณสมบัติ

```
<property>
        <name>dfs.replication</name>
        <value>1</value>
</property
```

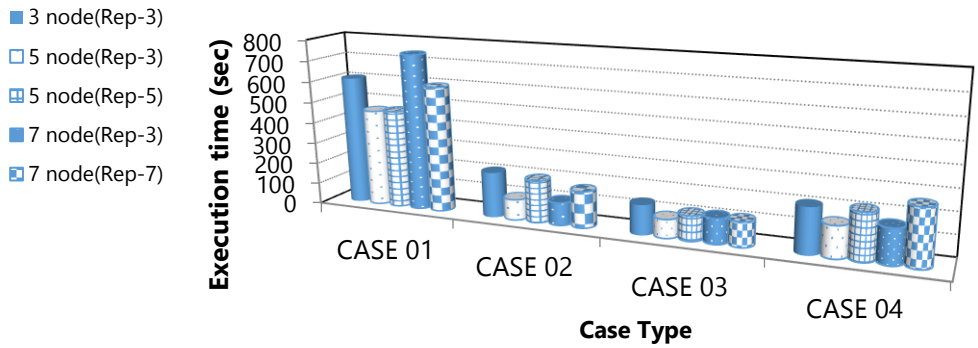
เพียงระบุจำนวนที่ต้องการสำเนา โดยใช้คำสั่ง stop-all.sh และ start-all.sh ใหม่ทุกครั้งเพื่อให้ Hadoop cluster ปรับค่าจำนวนการสำเนา

4. ผลการวิจัย

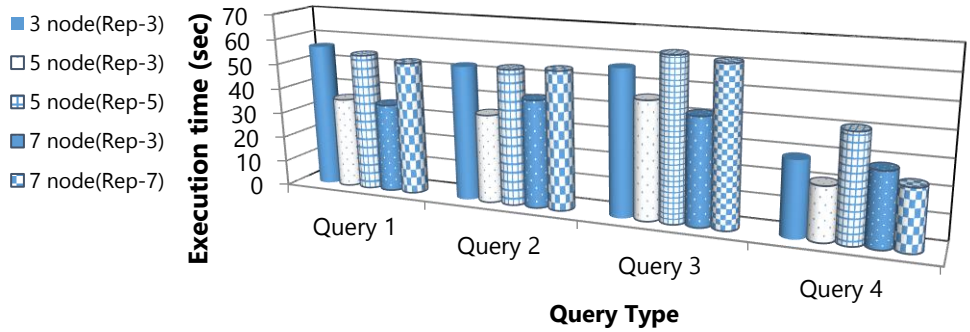
ผลจากการประมวลผลที่วัดเป็นวินาทีแยกตามการทดสอบแต่ละแบบ โดยผู้วิจัยนำเสนอแยกตาม Hadoop Cluster ในตารางที่ 10 มีภาพรวมของการประมวลผลตามรูปที่ 7 และตาม query ในรูปที่ 8 เพื่ออธิบายผลกระทบตามสมมติฐานในตอนต้น

จากผลการประมวลผลแต่ละชุดคำสั่ง SQL ชุดคำสั่ง CASE 01 ใช้เวลาประมวลผลนานที่สุดด้วยผลลัพธ์ข้อมูล 67,167,328 ระเบียน ด้วยเงื่อนไขของการเรียงลำดับจากมากไปน้อย ระเบียนในส่วนของ CASE 02 - 04 มีค่าเฉลี่ยเวลาในการประมวลผลที่ใกล้เคียงกัน หากพิจารณาแยกตามรูปแบบ Hadoop Cluster และจำนวนการสำเนาข้อมูล แบบ 5 node (Rep-3) สามารถประมวลผลได้ดีที่สุด และรองลงมาคือ 5 node (Rep-5) และ 7 node (Rep-3) เนื่องจากการทำ Map-Reduce ที่โปรแกรม Hive จะคำนวณแบบอัตโนมัติอยู่ที่ Map=5/Reduce=1 ทุกชุดคำสั่ง

เมื่อพิจารณาแยกตามแต่ละ Query ที่แยกออกจากชุดคำสั่งแต่ละ CASE พบว่ารูปแบบของ Hadoop Cluster ที่



รูปที่ 7 ค่าเวลาเฉลี่ยของการประมวลผลแยกตามชุดคำสั่ง SQL



รูปที่ 8 ค่าเวลาเฉลี่ยของการประมวลผลแยกตาม Query

เหมาะสมกับการประมวลผลได้แก่ แบบ 5 node (Rep-3) และ 7 node (Rep-5) สำหรับรูปแบบ Hadoop Cluster อื่นๆ มีค่าใกล้เคียงกันยกเว้นการใช้คำสั่ง JOIN ใน Query 4 มีความแตกต่างใน 5 node (Rep-5) โดยผลของค่า อยู่ที่ 40.89 ที่ Map=5/Reduce=1 เช่นกัน

5. สรุปผลการวิจัย

จากการศึกษาการประมวลผลของข้อมูลอุณหภูมิของสถานีอุตุนิยมวิทยาด้วยชุดข้อมูลจำนวน 207,466,688 ระเบียนที่ประกอบด้วยข้อมูลของ รหัสสถานี ชื่อสถานี วันที่ อุณหภูมิสูงสุด และอุณหภูมิต่ำสุด ในรูปแบบฐานข้อมูลเชิงสัมพันธ์ บนโครงสร้างโปรแกรม Hadoop Cluster ที่มีจำนวน Data node และจำนวนการสำเนาชุดข้อมูลที่แตกต่างกัน โดยเปรียบเทียบประสิทธิภาพด้วย Query ที่มีความหลากหลายด้วยคำสั่งค่าเฉลี่ย avg( ) และค่าสูงสุด max( ) ควบคู่กับการรวมกลุ่ม “GROUP BY” และการเรียงลำดับ “ORDER BY” สามารถสรุปได้ว่า Hadoop Cluster 5 node(Rep-3) ใช้เวลาประมวลผลต่ำที่สุด เนื่องจากค่าของ Map-Reduce ที่ถูกกำหนดด้วยโปรแกรม Hive ที่ Map=5/Reduce=1 หมายถึงการ map อยู่ที่ 5 และ reduce อยู่ที่ 1 ซึ่งสอดคล้องกับจำนวน Data node ของชุดดังกล่าว ดังนั้นการเพิ่มขึ้นของจำนวน Data node ถือ

ว่ามีผลทำให้การประมวลผลรวดเร็วขึ้น แต่ต้องมีการปรับค่า Map-Reduce ให้เพิ่มขึ้นเพื่อกระจายภาระงาน เพราะประสิทธิภาพการกระจายงานนั้นถูกแลกมาด้วยเวลาที่ใช้ส่งข้อมูลระหว่าง Data node สำหรับประเด็นด้านเครือข่าย ซึ่งถือเป็นประเด็นวิจัยที่ต้องทำต่อไป พร้อมกับการกำหนดค่าคุณสมบัติอื่นๆ ของกลุ่มคลัสเตอร์ โดยการประมวลผลของ MapReduce ที่สนับสนุน Hadoop Hive มีกระบวนการบีบอัดข้อมูลระหว่างการ Map และ การ Reduce ซึ่งถือเป็นคุณสมบัติที่อาจเพิ่มประสิทธิภาพ สำหรับงานวิจัยนี้ในส่วนของการทดสอบการเพิ่มชุดสำเนาข้อมูลนั้นพบว่าไม่ส่งผลต่อเวลาที่ใช้ในการประมวลผลด้วยโปรแกรม Hive อย่างชัดเจน ตามค่าเวลาการประมวลผลในรูปที่ 7 และรูปที่ 8 เมื่อเปรียบเทียบที่จำนวน Data node เท่ากัน โดยการเพิ่มการสำเนานั้นสามารถกระจายข้อมูลอยู่ในหลาย Data node แต่ด้วยคุณสมบัติของคลัสเตอร์ที่มีการเชื่อมต่อระหว่าง Data node ด้วยการประมวลผลของ Hadoop นั้นมีข้อจำกัดในส่วนของการกระจายภาระงานที่มีค่าน้อยกว่าจำนวน Data node จึงส่งผลให้เกิดกรณีที่คาดไม่ถึง เช่น ข้อมูลที่ต้องเรียกใช้ อยู่ใน Data node ที่ไม่ถูกประมวลผล หรือเกิดปัญหาในการรวมข้อมูลจาก Data node หลายตัวมาประมวลผล ซึ่งข้อดีของการสำเนาจำนวนหลายชุดให้ผลดีโดยตรงกับความเสถียรภาพและป้องกันข้อผิดพลาดในกรณี Data node เสียหายหรือชำรุด แต่หากกำหนดจำนวนมากเกินความจำเป็นจะส่งผลกระทบกับการประมวลผลตามตัวอย่างของ Hadoop Hive ในงานวิจัยนี้

6. กิตติกรรมประกาศ

ขอขอบคุณสำนักวิทยบริการและเทคโนโลยีสารสนเทศและคณะวิทยาศาสตร์และเทคโนโลยี มหาวิทยาลัยราชภัฏสงขลาที่ให้โอกาส สนับสนุนเครื่องมือและทุนแก่นักศึกษาในการทำงานวิจัยและนำเสนอผลงาน และแนวคิดของงานวิจัยจากเว็บไซต์ open-bigdata [2]

7. เอกสารอ้างอิง

[1] V. Reynolds. Big Data For Beginners: Understanding SMART Big Data, Data Mining & Data Analytics For improved Business Performance, Life Decisions & More! . Kindle Edition, 2016.

[2] Hadoop's open source query tools. Performance test of Pig vs Hive with code examples. Available From: http: / / www. open-bigdata. com/ performance-test-pig-vs-hive-code-examples/ [Accessed 5th Fab 2017].

[3] D. Abadi, S. Babu, F. Ozcan, and I Pandis. Tutorial: SQL-on-Hadoop Systems. Proceedings of the VLDB Endowment. 2015 Aug 31-Sep 4; Kohala Coast, Hawaii. p. 2050-2051.

[4] K. Jayasri, R. Rajmohan, and D. Dinagaran. Analyzing the Query Performances of Description Logic based Service Matching using Hadoop. Proceeding of International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology. 2015 May 6-8; Chennai, T.N., India. p. 1-7.

[5] Adisorn G, Suparerk M. Performance of the Apache Mahout on Apache Hadoop Cluster. Proceeding of Electrical Engineering Conference 38th. 2015 Nov 18-20; Pranakornsrya, Ayutthaya, Thailand, p. 858-861.Thai.

[6] The Big Data Blog. Hadoop Ecosystem Overview. Available from: http:/ / thebigdata blog.weebly.com/blog/the-hadoop-ecosystem-overview/ [Accessed 5th Fab 2017].

[7] The Hortonworks Blog. ORCFile in HDP 2: Better Compression, Better Performance. Available from: http://hortonworks.com/blog /orcfile-in-hdp-2-better-compression-better-performance/.

[8] MAPR.blog. What Kind of Hive Table is Best for Your Data. Available From: https:/ / www.mapr

.com/ blog/ what-kind-hive-table-best-your-data/ [Accessed 5th Fab 2017].

[9] Apache Hive. LanguageManual ORC. Available From: https:/ / cwiki. apache. org/ confluence/ display/ Hive/ LanguageManual+ ORC [Accessed 5th Fab 2017].