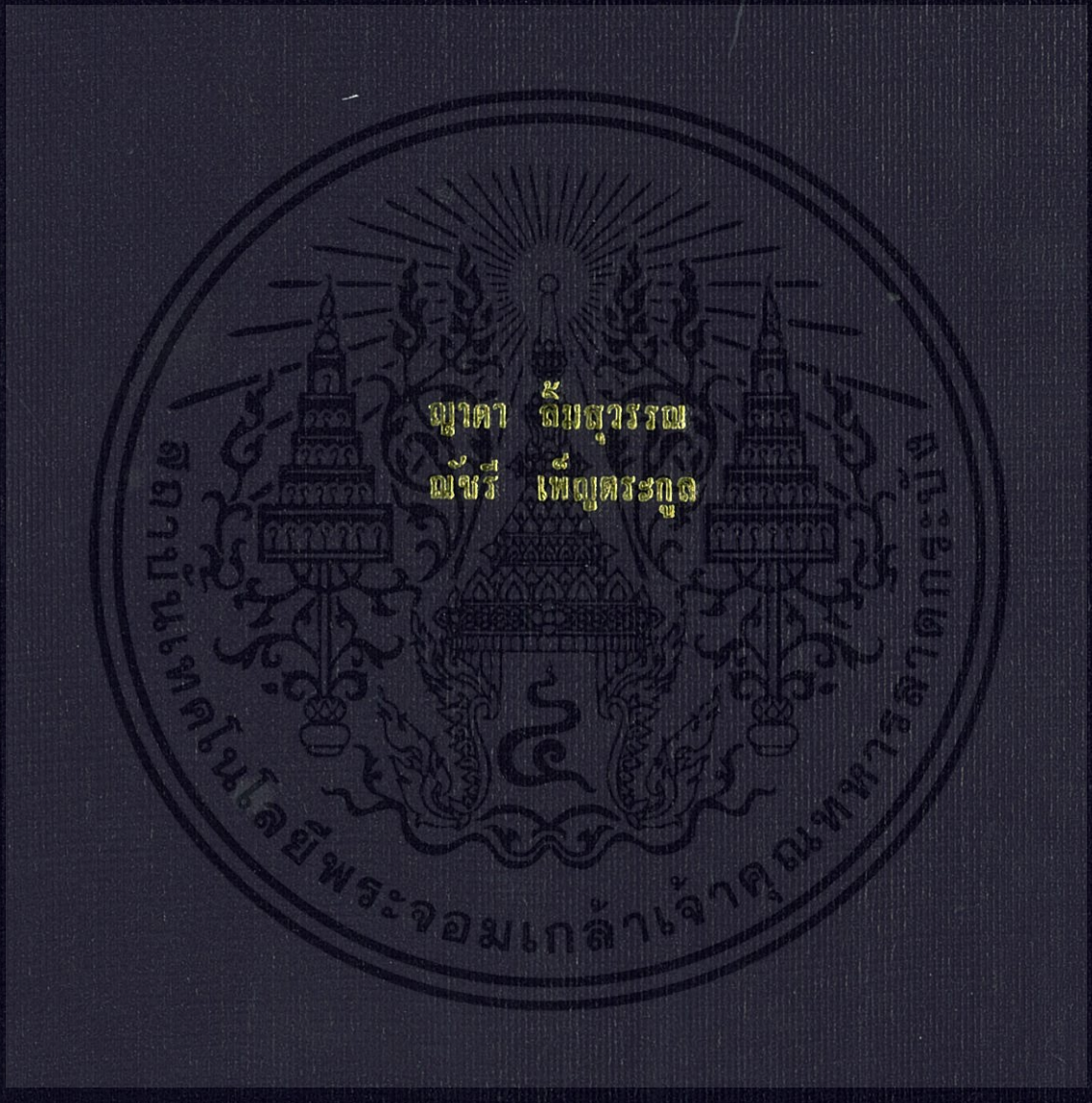
การพัฒนาฐานข้อมูลขนาดใหญ่ด้วยฮาดูป

BIG DATA DEVELOPMENT BASED ON HADOOP

ฌาดา ฉันสุวรรณ
ณัชรี เพ็ญตระกูล



ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

การพัฒนาฐานข้อมูลขนาดใหญ่ด้วยฮาดูป

BIG DATA DEVELOPMENT BASED ON HADOOP

ญาดา ลิ้มสุวรรณ
ณัชรี เพ็ญตระกูล

ปริญญานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2556

ปริญญานิพนธ์ปีการศึกษา 2556

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง การพัฒนาฐานข้อมูลขนาดใหญ่ด้วยฮาดูป

BIG DATA DEVELOPMENT BASED ON HADOOP

ผู้จัดทำ

1. นางสาว ญาดา	ลิ้มสุวรรณ	รหัสนักศึกษา	53010395
2. นางสาว ณัชรี	เพ็ญตระกูล	รหัสนักศึกษา	53010433

.....อาจารย์ที่ปรึกษา

(รองศาสตราจารย์ กฤตวัน ศิริบูรณ์)

.....อาจารย์ที่ปรึกษาร่วม

(ดร.วรวัฒน์ ลิ้มโภคา)

การพัฒนาฐานข้อมูลขนาดใหญ่ด้วยฮาดูป

นางสาว ญาดา	ลิ้มสุวรรณ	53010395
นางสาว ณัชรี	เพ็ญตระกูล	53010433
รศ.กฤตวัน	ศิริบูรณ์	อาจารย์ที่ปรึกษา
ดร.วรวัฒน์	ลิ้มโภคา	อาจารย์ที่ปรึกษาร่วม

ปีการศึกษา 2556

บทคัดย่อ

ด้วยเทคโนโลยีทางคอมพิวเตอร์ที่พัฒนามาจนถึงปัจจุบัน ทำให้พื้นที่ข้อมูลถูกขยายออกไปรูปแบบข้อมูลไร้โครงสร้างที่เริ่มเป็นที่นิยมมากขึ้นโดยเฉพาะใน Social Network ซึ่งเป็นปัจจัยหลักก่อให้เกิดข้อมูลขนาดยักษ์ Big Data ที่ความสามารถของระบบประมวลผลเชิง Database เกินจะรองรับได้ บริษัทหลายแห่งเริ่มให้ความสำคัญกับการพัฒนาข้อมูล Big Data มากขึ้นเรื่อยๆ ระบบประมวลผลแบบกระจาย (Distributed Computing) ก็เป็นหนึ่งในทางเลือกที่สามารถประมวลผลข้อมูลขนาดยักษ์ได้อย่างมีประสิทธิภาพ จึงเกิดการพัฒนาระบบจัดการข้อมูลขนาดยักษ์ภายใต้หลักการดังกล่าว

ปริญญานิพนธ์เล่มนี้กล่าวถึงการศึกษา Apache Hadoop ซึ่งเป็น Open-Source ที่ใช้จัดการพัฒนาข้อมูลขนาดยักษ์ ซึ่งเชื่อว่าจะเป็นที่นิยมและเกิดการศึกษาพัฒนาทางด้านนี้อีกมาก เพื่อนำไปรองรับข้อมูลที่ยังคงขยายตัวขึ้นอีกในอนาคต

BIG DATA DEVELOPMENT BASED ON HADOOP

Ms.Yada	Limsuwan	53010395
Ms. Natcharee	Pentrakul	53010433
Assoc.Prof. Kritawan	Siriboon	Advisor
Dr. Voravat	Limpoka	Co-advisor

Academic Year 2013

ABSTRACT

Because of the rising of computer technology, the data has been expanding until now. Unstructured data which became more popular, especially from social network data, is the main reason causes an enormous data called big data that database processing system is not enough to support. Many company starts to focus on big data development. Until found that Distributed Computing is one of the big data's performance solution. So the development that base on this theory has begin.

This thesis is study about Apache Hadoop which is open source that used to manage and develop big data. Believe it will be on trend and be more developed soon, for support enormous data which still be expanding in the future.

กิตติกรรมประกาศ

ความสำเร็จของคนหนึ่งคนเกิดจากการเดินทางผ่านเส้นทางมากมาย ย่อมต้องมีการเกื้อหนุนจากคนรอบข้าง จึงมายืน ณ จุดหนึ่งนี้ได้ เช่นเดียวกับปริญญานิพนธ์เล่มนี้ที่สำเร็จลุล่วงไปได้ด้วยดี

ขอกราบขอบคุณบิดา มารดา ผู้คอยประคับประคองข้าพเจ้ามาตั้งแต่เริ่มต้น อบรมเลี้ยงดูจนเติบโต และสนับสนุนโอกาสทางการศึกษา รวมไปถึงครูบาอาจารย์ตั้งแต่ประถมจนมัธยม เชื่อว่าพวกท่านก็เป็นคนนำพาให้ได้เข้ามาเรียน ณ สถาบันแห่งนี้เช่นกัน

ขอบคุณรุ่นพี่ เพื่อนๆ และน้องๆ ทั้งในภาควิชาวิศวกรรมคอมพิวเตอร์และนอกภาควิชา ที่คอยให้คำปรึกษาและช่วยเหลือยามลำบาก ขอบคุณอาจารย์ภายในภาควิชาผู้ถ่ายทอดวิชาเฉพาะทางอันเป็นพื้นฐานของการศึกษาปริญญานิพนธ์เล่มนี้

ท้ายที่สุด ขอขอบคุณ รศ.กฤตวัน ศิริบูรณ์ และดร.วรวัฒน์ ลิ้มโภคา ที่ปรึกษาปริญญานิพนธ์เล่มนี้ ผู้มอบโอกาสให้ศึกษาและทำวิจัยหัวข้อนี้ ซึ่งเป็นเรื่องที่น่าสนใจและยังค่อนข้างใหม่ในประเทศไทย ขอบคุณที่สละเวลามาให้คำปรึกษา ชี้แนะแนวทาง และเป็นแรงผลักดันสำคัญของการศึกษาปริญญานิพนธ์เล่มนี้

นางสาว ญาดา ลิ้มสุวรรณ

นางสาว ณัชรี เพ็ญตระกูล

สารบัญ

หน้า

บทคัดย่อภาษาไทย i

บทคัดย่อภาษาอังกฤษ ii

กิตติกรรมประกาศ iii

สารบัญ iv

สารบัญตาราง vii

สารบัญรูป ix

บทที่ 1 บทนำ 1

1.1 ความสำคัญและที่มาของโครงงาน 1

1.2 วัตถุประสงค์ของโครงงาน 1

1.3 ขอบเขตของโครงงาน 2

1.4 วิธีการดำเนินการ 2

1.5 ประโยชน์ที่คาดว่าจะได้รับ 3

1.6 ส่วนประกอบของปริญญานิพนธ์ 3

บทที่ 2 ทฤษฎีบท 5

2.1 ระบบประมวลผลแบบกระจาย (Distributed Computing System) 5

2.1.1 Message Passing 5

2.1.2 Remote Procedure Calls 7

2.1.3 ข้อดีของการประมวลผลแบบกระจาย 7

2.1.4 ข้อเสียของการประมวลผลแบบกระจาย 8

2.2 การประมวลผลแบบขนาน (Parallel Computing) 9

2.3 ระบบประมวลผลคลัสเตอร์ (Cluster Computing System) 10

2.4 ระบบจัดการข้อมูลฮาดูป (Hadoop System) 11

2.4.1 โครงสร้างของฮาดูปคลัสเตอร์ 12

2.4.2 หลักการทำงานของฮาดูป 13

สารบัญ (ต่อ)

หน้า

บทที่ 3 การออกแบบและพัฒนา 45
3.1 บทนำ 45
3.2 ความต้องการของระบบ 45
3.2.1 แพลตฟอร์มที่รองรับ 45
3.2.2 ซอฟต์แวร์ที่ต้องการ 45
3.3 โครงสร้างของระบบ 46
3.4 การตั้งค่าระบบ 47
3.4.1 การตั้งค่าพื้นฐานในไฟล์ hadoop-env.sh 47
3.4.2 การตั้งค่าในไฟล์ conf/core-site.xml 47
3.4.3 การตั้งค่าในไฟล์ conf/mapred-site.xml 47
3.4.4 การตั้งค่าในไฟล์ conf/hdfs-site.xml 48
3.4.5 การตั้งค่าในไฟล์ conf/master 49
3.4.6 การตั้งค่าในไฟล์ conf/slaves 49
3.5 การใช้งานระบบ 50
3.5.1 การเริ่มต้นฮาดูป 50
3.5.2 การใช้งาน Hadoop 51
3.6 การพัฒนาระบบด้วย Eclipse 55
3.7 การพัฒนาระบบด้วย Pig Latin 55
3.8 การพัฒนาระบบด้วย Hive 56

บทที่ 4 การทดลองและผลการทดลอง 57
4.1 การทดลองนับคำทั้งหมดที่ปรากฏ (WordCount) 57
4.1.1 ขั้นตอนการทำงานของโปรแกรมนับคำ (Word Count) 57
4.1.2 การทดลองนับคำทั้งหมดในไฟล์ 61
4.2 การทดลองโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file 64
4.2.1 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file 65
4.2.2 การทดลองโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file 69

สารบัญ (ต่อ)

หน้า

4.2.3 สรุปผลการทดลอง 90
4.2.3.1 สรุปผลการทดลองนับคำทั้งหมดที่ปรากฏ (Word Count) 90
4.2.3.2 สรุปผลการทดลองโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับไฟล์ล็อก 91
4.2.3.3 สรุปลักษณะโปรแกรมที่ถูกสร้างด้วยจาวา (JAVA) พิกละติน (Pig Latin) และไฮฟ์ (Hive) 91

บทที่ 5 บทสรุปและข้อเสนอแนะ 93
5.1 บทสรุป 93
5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข 93
5.3 แนวทางในการพัฒนาต่อ 94

บรรณานุกรม 95

ภาคผนวก ก 98
ก1 วิธีการติดตั้ง Hadoop แบบโหนดเดี่ยว (Single node) บน CentOS6.5 98

สารบัญรูป (ต่อ)

รูปที่	หน้า
รูปที่ 2.1 แสดงโครงสร้างระบบการประมวลผลแบบกระจาย	5
รูปที่ 2.2 แสดงการทำงานของ RPC	6
(ก) แสดงการติดต่อสื่อสารผ่าน Message-oriented Middleware	6
(ข) แสดงการติดต่อสื่อสารผ่าน RPC	6
รูปที่ 2.3 การส่ง Message เพื่อติดต่อระหว่าง Cluster	6
รูปที่ 2.4 แสดงโปรโตคอลของ Remote Procedure Call	7
รูปที่ 2.5 แสดงการประมวลผลแบบขนานในระดับ CPU	9
รูปที่ 2.6 แสดงโครงสร้างของระบบคลัสเตอร์	10
รูปที่ 2.7 แสดงโลโก้ของ Apache Hadoop	11
รูปที่ 2.8 โครงสร้างของ Hadoop Sever Roles	12
รูปที่ 2.9 แสดงทำการแบ่งไฟล์ File.txt	14
รูปที่ 2.10 เมื่อ Data node ขาดการติดต่อ	15
รูปที่ 2.11 แสดงการติดต่อระหว่าง Name Node และ Secondary Name Node	15
รูปที่2.12 แสดงการเขียนไฟล์เข้าสู่ HDFS	16
รูปที่ 2.13 แสดงแผนภาพตัวอย่างการอ่านไฟล์ของ Client จาก HDFS ของไฟล์ Result.txt	17
รูปที่ 2.14 แสดงแผนภาพตัวอย่างการอ่านไฟล์ของ Data node จาก HDFS ของไฟล์ File.txt	18
รูปที่ 2.15 แสดงการทำงานโดยภาพรวมของ Map	18
รูปที่ 2.16 แสดงการทำงานโดยภาพรวมของ Reduce	19
รูปที่ 2.17 แสดงการทำงานของคลัสเตอร์	20
(ก) แสดงการทำงานของคลัสเตอร์ช่วง Map	20
(ข) แสดงการทำงานของคลัสเตอร์ช่วง Reduce	20
รูปที่ 2.18 แสดงขั้นตอนการทำงานของ MapReduce ตั้งแต่นำไฟล์ข้อมูลมาจาก Block จนประมวลผลเสร็จและให้ HDFS จัดเก็บข้อมูลอีกครั้ง	22
รูปที่ 2.19 แสดง Input และ Output เมื่อผ่าน LineRecordReader	23
รูปที่ 2.20 แสดงผลลัพธ์ของ Intermediate data ที่เปลี่ยนไปเมื่อผ่าน Combiner	24
รูปที่ 2.21 แสดงขั้นตอนการ Partition และ Shuffle	25
รูปที่ 2.22 แผนผังแอพพลิเคชั่นที่เป็น Hadoop EcoSystem	26

สารบัญรูป (ต่อ)

รูปที่	หน้า
รูปที่ 2.23 Apache Ambari	27
รูปที่ 2.24 Apache Avro	27
รูปที่ 2.25 Apache Cassandra	28
รูปที่ 2.26 Apache Chukwa	28
รูปที่ 2.27 Apache Flume	29
รูปที่ 2.28 Apache HBase	29
รูปที่ 2.29 Apache Zoo Keeper	30
รูปที่ 2.30 Apache Mahout	30
รูปที่ 2.31 Apache Oozie	31
รูปที่ 2.32 ตัวอย่าง Directed Acyclical Graph (DAG) ในอูซี่	31
รูปที่ 2.33 แสดงแนวคิดการทำงานของสคูป	32
รูปที่ 2.34 Apache Pig	33
รูปที่ 2.35 แสดงการประมวลผลของคอมไพล์เลอร์จากภาษา Pig Latin สู่แมปรีดิวซ์	34
รูปที่ 2.36 Apache Hive	37
รูปที่ 2.37 ตัวอย่างข้อมูลไฟล์ล็อก	40
รูปที่ 2.38 แผนผังอีเมล์เซิร์ฟเวอร์	42
รูปที่ 2.39 ตัวอย่างไฟล์ล็อก สังเกตได้ว่าจะมีรูปแบบวันที่ขึ้นต้น ตามด้วยเวลา และข้อมูลอื่นๆ	43
รูปที่ 3.1 โครงสร้างของระบบ	46
รูปที่ 3.2 ผลการสั่งฟอร์แมตเนมโหนดก่อนการเริ่มต้นการใช้งานฮาดูป	50
รูปที่ 3.3 ผลการสั่งเริ่มต้นการใช้งาน Hadoop	50
รูปที่ 3.4 ผลการป้อนคำสั่งตรวจสอบพาธใน HDFS	51
รูปที่ 3.5 ผลการสั่งการทำงาน WordCount	52
รูปที่ 3.6 ยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ของ Namenode Daemon	53
รูปที่ 3.7 ยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ของ JobTracker Daemon	54
รูปที่ 3.8 ยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ของ TaskTracker Daemon	54
รูปที่ 3.9 โปรแกรม Eclipse (Kepler) สามารถทำงานก่อนสร้างไฟล์ JAR	55

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

รูปที่ 4.1 แสดงการแบ่งกระจายการทำงานของอินพุตไฟล์ (Input file) ให้กับแต่ละ สปริท (Split) ในโปรแกรมนับคำ (Word Count) 57

รูปที่ 4.2 แสดงการทำงานของ LineRecordReader ให้กับแต่ละสปริท (Split) ในโปรแกรม WordCount 58

รูปที่ 4.3 แสดงการทำงานของฟังก์ชันแมป (Map) ในโปรแกรม WordCount 59

รูปที่ 4.4 แสดงการทำงานของฟังก์ชัน Suffle&Sort ในโปรแกรม WordCount 60

รูปที่ 4.5 แสดงการทำงานฟังก์ชันรีดิวซ์ (Reduce) ในโปรแกรม WordCount 60

รูปที่ 4.6 การทำงานของโปรแกรมนับคำ (WordCount) 64

รูปที่ 4.7 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file ภาษาจาวา ด้วยหลักการแมปรีดิวซ์ (Map Reduce) 66

รูปที่ 4.8 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file ด้วย Pig Lantin 67

รูปที่ 4.9 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file ด้วย Hive 68

รูปที่ 4.10 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Server log ด้วยภาษา JAVA 71

รูปที่ 4.11 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Server log ด้วยภาษา Pig 71

รูปที่ 4.12 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Server log ด้วยภาษา Hive 72

รูปที่ 4.13 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Proxy log ด้วยภาษา JAVA 77

รูปที่ 4.14 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Proxy log ด้วยภาษา Pig 78

รูปที่ 4.15 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Webmail log ด้วยภาษา JAVA 82

รูปที่ 4.16 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Webmail log ด้วยภาษา Pig 83

สารบัญรูป (ต่อ)

รูปที่ **หน้า**

รูปที่ 4.17 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Webmail log ด้วยภาษา Hive 83
รูปที่ 4.18 แสดง ความถี่ที่ของเหตุการณ์ที่เกิดขึ้นของแต่ละวันของ Mail log 89
รูปที่ ก1.1 แสดง Namenode 103
รูปที่ ก1.2 แสดง Jobtracker 103
รูปที่ ก1.3 แสดง Jobtracker 104

บทที่ 1
บทนำ

1.1 ความสำคัญและที่มาของโครงงาน

ในยุคปัจจุบันที่คอมพิวเตอร์เริ่มมีบทบาทกับมนุษย์มากขึ้นเรื่อยๆ จากแต่ก่อนที่ใช้ในเพียงการคำนวณสำหรับงานวิจัยขั้นสูง จนพัฒนามาเป็นเทคโนโลยีที่อยู่รอบตัวเรา จึงไม่น่าแปลกใจที่จำนวนข้อมูลทางคอมพิวเตอร์ (bits) ทั้งหมดในแต่ละปีจะเพิ่มสูงในอัตราที่รวดเร็วและไม่มีทีท่าว่าจะชะลอตัว ในไม่กี่ปีที่ผ่านมาจึงได้เกิดคำศัพท์ที่เรียกว่า ข้อมูลขนาดยักษ์ (Big Data) ซึ่งหมายถึงข้อมูลขนาดมหาศาลที่ประสิทธิภาพของคอมพิวเตอร์ทั่วไปไม่สามารถรองรับได้ โดยเฉพาะข้อมูลไม่มีโครงสร้าง (Unstructured Data) ซึ่งเป็นชนิดข้อมูลที่นิยมใช้กับข้อมูลสมัยใหม่ ที่เห็นได้ชัดคือข้อมูลสังคมออนไลน์ (Social Media Data) และใช้พื้นที่ในการเก็บมากกว่าข้อมูลประเภทโครงสร้าง (Structured Data) ค่อนข้างมาก ท้ายที่สุดแล้ว การเกิดข้อมูลขนาดยักษ์ข้างต้นนำมาซึ่งปัญหาสองประการ

ประการแรก คอมพิวเตอร์ธรรมดาไม่สามารถประมวลผลข้อมูลขนาดยักษ์ได้อย่างมีประสิทธิภาพ ซึ่งเห็นได้ชัดในเรื่องของเวลา จึงเกิดแนวคิดที่เรียกว่าการประมวลผลแบบกระจาย (Distributed Computing) เพื่อกระจายข้อมูลไปให้คอมพิวเตอร์หลายๆเครื่องประมวลผลแทนการซื้อคอมพิวเตอร์ประสิทธิภาพสูงแต่ราคาแพง

ประการที่สอง ด้วยข้อมูลที่เพิ่มขึ้น ซ้ำทั้งยังเป็นข้อมูลแบบไร้โครงสร้าง ข้อมูลที่ได้จะไม่ได้อยู่ในรูปของตาราง (Schema) จึงทำให้ระบบจัดการข้อมูลเชิง Database เช่น MySQL ไม่สามารถนำมาใช้ได้อีกต่อไป บริษัท Apache จึงคิดระบบจัดการข้อมูลฮาดูป (Hadoop) ซึ่งอยู่ภายใต้แนวคิดของการประมวลผลแบบกระจายมารองรับข้อมูลขนาดยักษ์ โดยจะเน้นการประมวลผลข้อมูลแบบไร้โครงสร้างให้มีประสิทธิภาพมากที่สุด

อย่างไรก็ตาม แนวโน้มในอนาคตข้างหน้า การจัดการแก้ปัญหาข้อมูลขนาดยักษ์ยังคงมีบทบาทและเป็นที่ต้องการการพัฒนาอย่างต่อเนื่อง จึงเกิดความสนใจศึกษาการจัดการข้อมูลเหล่านั้นภายใต้ระบบฮาดูป เพื่อที่จะสามารถรองรับข้อมูลที่มากขึ้นในอนาคตได้

1.2 วัตถุประสงค์ของโครงงาน

1) ศึกษาหลักการทำงานของระบบจัดการข้อมูล Apache Hadoop
2) สามารถติดตั้ง Apache Hadoop บนระบบปฏิบัติการลินุกซ์ (Linux) ได้

3) ศึกษาการทำงานของ HDFS และสามารถประยุกต์การตั้งค่าได้
4) ศึกษาหลักการทำงานของแมปรีดิวซ์ (Map Reduce) และสามารถตั้งค่า เขียนโปรแกรมประยุกต์ใช้ได้
5) ศึกษาหลักการทำงานแอพพลิเคชั่นที่ทำงานบนฮาดูปเฟรมเวิร์ก สามารถเขียนโปรแกรมประยุกต์ใช้ได้อย่างเหมาะสม
6) สามารถติดตั้งระบบประมวลผล และนำข้อมูลขนาดยักษ์มาประมวลผลบนฮาดูป ได้ตามหลักการประมวลผลแบบกระจาย
7) สามารถสรุปคุณลักษณะที่แตกต่างกันระหว่างฮาดูป และแอพพลิเคชั่นที่ทำงานบนฮาดูปเฟรมเวิร์ก และระบุสถานการณ์ที่เหมาะสมในการใช้งานของทั้งสองระบบได้

1.3 ขอบเขตของโครงงาน

โครงงาน พัฒนาฐานข้อมูลขนาดใหญ่ด้วยฮาดูป ดำเนินการศึกษาและทดลองภายใต้องค์ประกอบและขอบเขตดังนี้

1) สามารถให้ฮาดูปจัดการประมวลผลแบบกระจายได้ (Multi-Node) โดยการศึกษาและทดลองบนกลุ่มทรัพยากรคอมพิวเตอร์คลาวด์ (Cloud Server)
2) ระบบจัดการข้อมูลฮาดูปที่ใช้ดำเนินการบนระบบปฏิบัติการ ลินุกส์
3) ทดลองการเขียน แมปรีดิวซ์ และ Pig ของโปรแกรมวิเคราะห์ไฟล์ล็อก (Log File) ด้วยภาษา Java และ Pig Latin โดยโปรแกรมทั้งสองภาษาจะสามารถ ค้นหาข้อมูล(query), คัดเลือกข้อมูลที่สนใจ(Filter) และนับกลุ่มเหตุการณ์ที่สนใจได้(Count)
4) นำข้อมูลขนาดใหญ่มาประมวลผลด้วยฮาดูปทั้งที่เป็นแมปรีดิวซ์ (Map Reduce) และพิก (Pig) นำผลลัพธ์มาสรุปคุณลักษณะความสามารถของทั้งสอง

1.4 วิธีการดำเนินการ

1) ศึกษาการขอบเขตและหลักการทำงานของระบบฮาดูป (Hadoop)
2) ศึกษาการขอบเขตและหลักการทำงานของ HDFS และ แมปรีดิวซ์ (Map Reduce)
3) ศึกษาการขอบเขตและหลักการทำงานของแอพพลิเคชั่นที่ทำงานบนฮาดูปเฟรมเวิร์กพิก (Pig) รวมไปถึงศึกษาภาษาสคริปท์พิกละติน (Pig Latin)
4) ติดตั้งทรัพยากรที่ใช้ จำลองระบบและนำโปรแกรมตัวอย่างมาทำการทดลองเพื่อใช้ประกอบทฤษฎีที่ทำการศึกษา

5) ศึกษาข้อมูลและคุณสมบัติของไฟล์ล็อก (Log File) รวมไปถึงประเภทข้อมูลที่สนใจจะนำมาวิเคราะห์
6) ออกแบบและพัฒนาโปรแกรมวิเคราะห์ไฟล์ล็อก โดยพัฒนาทั้งแบบแมปรีดิวซ์และ Pig
7) ติดตั้งทรัพยากรระบบจริงผ่านทางระบบประมวลผลบนกลุ่มก้อนเมฆ (Cloud Computing) นำโปรแกรมที่พัฒนาทั้งสองมาใช้กับข้อมูลขนาดใหญ่ นำผลลัพธ์ที่ได้ รวมไปถึงตัวแปรต่างๆ ของระบบมาเปรียบเทียบและหาข้อสรุป

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1) เข้าใจหลักการทำงานและขอบเขตของระบบฮาดูป
2) เข้าใจขอบเขตและหลักการทำงานของ HDFS และ แมปรีดิวซ์
3) สามารถเขียนโปรแกรมบนระบบฮาดูป เพื่อนำไปพัฒนาและจัดการข้อมูลขนาดยักษ์ได้
4) เข้าใจขอบเขตและหลักการของแอพพลิเคชั่นที่ประมวลผลบนฮาดูปเฟรมเวิร์ก ได้แก่ Pig
5) สามารถเขียนโปรแกรมบนแอพพลิเคชั่น Pig ด้วยภาษา Pig latin ได้
6) เข้าใจความแตกต่าง ประสิทธิภาพ และสถานการณ์ที่เหมาะสมสำหรับการใช้งานระหว่างการเขียนโปรแกรมแบบแมปรีดิวซ์ และ Pig

1.6 ส่วนประกอบของปริญญานิพนธ์

ปริญญานิพนธ์ฉบับนี้ได้แบ่งเนื้อหาโดยทั่วไปออกเป็น 5 บทด้วยกัน ได้แก่

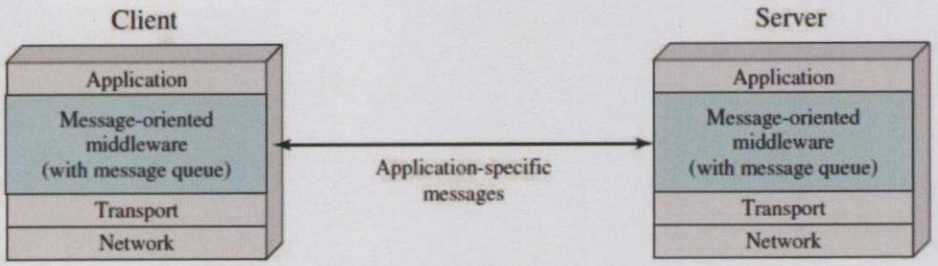
บทที่ 1 บทนำ กล่าวถึงความสำคัญและที่มาของโครงงาน วัตถุประสงค์ของโครงงาน ขอบเขตของโครงงาน วิธีการดำเนินการ ประโยชน์ที่คาดว่าจะได้รับ และส่วนประกอบของปริญญานิพนธ์

บทที่ 2 ทฤษฎีที่เกี่ยวข้อง ประกอบด้วย ความรู้ในด้านระบบประมวลแบบต่างๆที่ใช้เป็นหลักการของฮาดูป ทั้งระบบประมวลผลแบบกระจาย (Distributed Computing System) การประมวลผลแบบขนาน (Parallel Computing) และระบบประมวลผลคลัสเตอร์ (Cluster Computing System) จากนั้นจึงกล่าวระบบจัดการข้อมูลฮาดูป (Hadoop System) โดยรวม ก่อนจะแยกเข้าไปในระบบจัดการไฟล์แบบกระจายของฮาดูป (Hadoop Distributed File System) และหลักการแมปรีดิวซ์ (MapReduce) ของระบบฮาดูป แล้วจึงเข้าสู่ระบบนิเวศน์ซองฮาดูป (Hadoop EcoSystem) รวมถึงแอพพลิเคชั่นบนฮาดูปที่ชื่อว่า อาปาเช่ พิก (Apache Pig) และ อาปาเช่ ไฮฟ์ (Apache Hive) ท้ายที่สุดคือความหมายและความสำคัญของไฟล์ล็อก (Log file) รวมไปถึงแนวทางในการพัฒนาโปรแกรมวิเคราะห์ไฟล์ล็อกด้วย

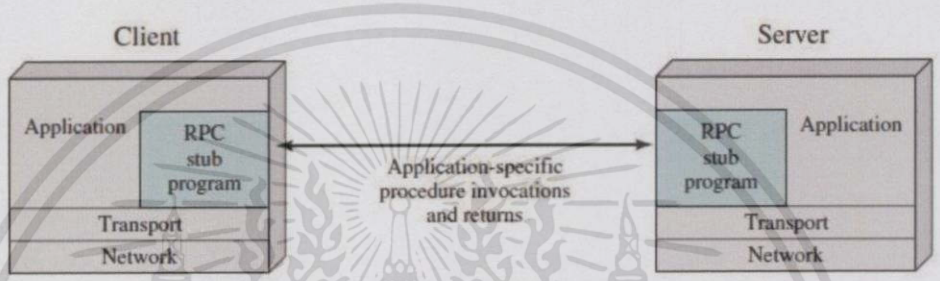
บทที่ 3 สำหรับเนื้อหาในบทนี้ จะกล่าวถึงความต้องการของระบบ อธิบายถึงโครงสร้างของฮาดูป แบบโหนดเดียว (Single-node) และแบบหลายโหนด (Multi-node) รวมไปถึงการใช้งาน Hadoop Distribute File System (HDFS) และ แมปริดิวซ์ (Map Reduce) เฟรมเวิร์ค (Map Reduce Framework) ซึ่งจะเป็นการใช้งานเบื้องต้น เสนอวิธีการพัฒนาระบบ อีกทั้งยังนำเสนอวิธีการใช้งานของพิก (Pig Latin) ซึ่งสามารถทำงาน Map reduce ในเชิงเป็นไฟล์สคริปท์สำหรับการการวิเคราะห์ไฟล์ล็อก (Log file) ได้ และนำเสนอนำข้อมูลที่นำมาใช้ในระบบ เพื่อความเข้าใจเนื้อหาในลำดับต่อไป

บทที่ 4 การทดลองและผลการทดลอง โดยในภาคการศึกษานี้นักศึกษาได้ออกแบบการทดลองไว้ 2 การทดลอง ได้แก่ โปรแกรมนับคำ (Word Count) และ โปรแกรมค้นหาและจัดกลุ่มข้อมูลให้ กับ Log file ซึ่งทั้งสองโปรแกรมจะเขียนทั้งในแบบแมปรีดิวซ์ และ Pig เพื่อเปรียบเทียบประสิทธิภาพ โดยในเนื้อหาจะกล่าวถึงวิธีการทำงานของโปรแกรมทั้งสอง โดยแบ่งเป็นทั้งสองแพลตฟอร์ม แสดงผลลัพธ์ที่ได้ รวมถึงตารางสรุปการทำงานของระบบคลัสเตอร์เมื่อทำการรันโปรแกรม

บทที่ 5 บทสรุป กล่าวถึงบทสรุปของประสิทธิภาพของทั้งสองแพลตฟอร์ม โดยอ้างอิงจากผลลัพธ์ในบท 4 วิจารณ์ผลลัพธ์ที่ได้ ข้อจำกัดที่พบ รวมถึงปัญหาอุปสรรคต่างๆ ของโครงงาน



(ก)



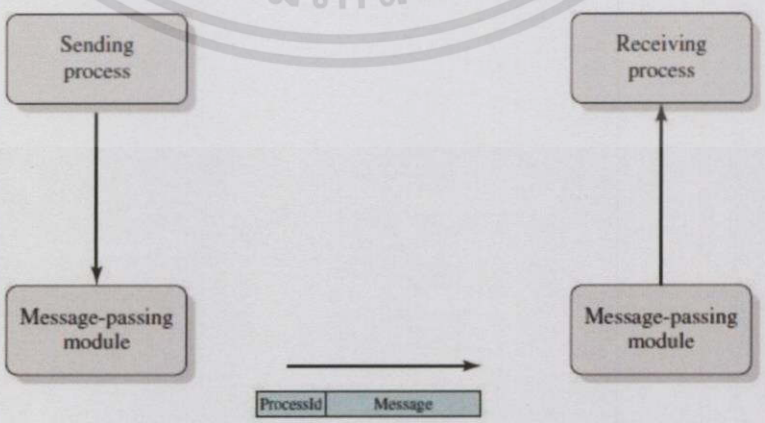
(ข)

รูปที่ 2.2 แสดงการทำงานของ RPC

(ก) แสดงการติดต่อสื่อสารผ่าน Message-oriented Middleware

(ข) แสดงการติดต่อสื่อสารผ่าน RPC

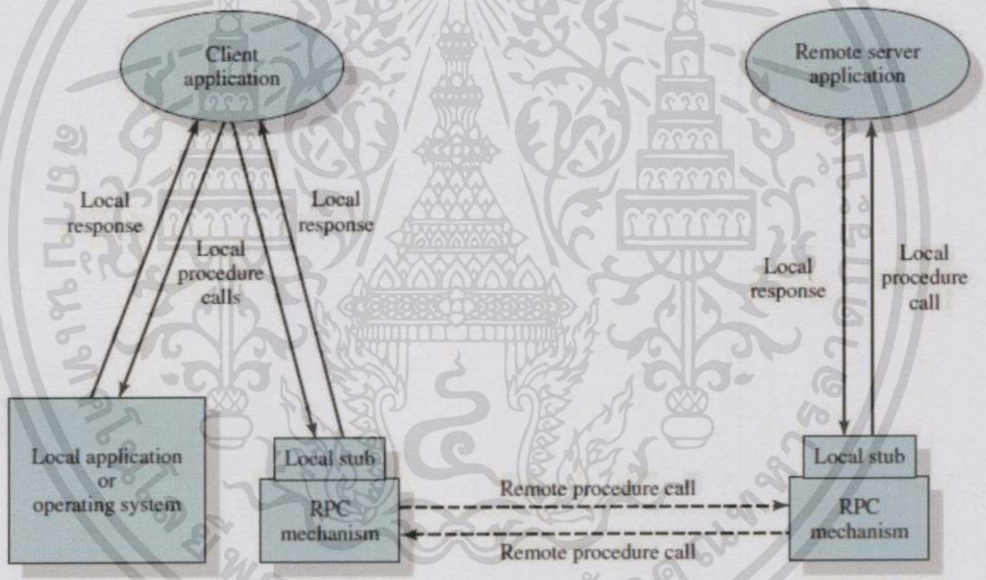
อย่างไรก็ตาม โครงสร้างของหน่วยข้อความจะไม่ต่างกันมากนัก คือภายในจะประกอบไปด้วยพารามิเตอร์ที่ระบุ process id ของเครื่องปลายทาง พร้อมทั้งเนื้อหาของข้อความ เมื่อเครื่องปลายทางได้รับ message ก็จะดูส่วน process id และเก็บข้อความ ไว้ใน buffer ของ process นั้นๆ



รูปที่ 2.3 การส่ง Message เพื่อติดต่อระหว่าง Cluster

2.1.2 Remote Procedure Calls

Remote Procedure Call (RPC) เป็นโปรโตคอลที่โปรแกรมหนึ่งสามารถใช้คำขอบริการจากโปรแกรมที่ตั้งอยู่ในคอมพิวเตอร์อีกเครื่องในเครือโดยไม่ต้องเข้าใจรายละเอียดเครือข่าย (บางครั้ง procedure call เรียกว่า function call or a subroutine call) RPC ใช้แบบจำลองลูกข่าย/แม่ข่าย โปรแกรมที่ขอเป็นลูกข่ายและโปรแกรมที่ให้บริการเป็นแม่ข่าย RPC เป็นปฏิบัติการแบบพร้อมกันที่ต้องการให้โปรแกรมที่ขอได้รับการแขวนจนกระทั่งผลลัพธ์จาก remote procedure ได้รับกลับมาเหมือนกับ procedure call ธรรมดา อย่างไรก็ตามการใช้ lightweight process หรือ thread ที่แบ่งปันพื้นที่เดียวกันจะยอมให้หลาย RPC ได้รับการทำอย่างพร้อมกัน



รูปที่ 2.4 แสดงโปรโตคอลของ Remote Procedure Call

2.1.3 ข้อดีของการประมวลผลแบบกระจาย

1) ใช้เวลาตอบสนองได้เร็วขึ้น (Quicker response time) เป็นระบบที่ตอบสนองตามคำสั่งของผู้ใช้ได้อย่างรวดเร็ว เมื่อเทียบกับระบบแบบรวมศูนย์ (Centralized systems) โดยเฉพาะเมื่อมีการใช้งานหลาย ๆ อย่างพร้อม ๆ กัน ซึ่งส่งผลให้บริการได้ดีขึ้น
2) ใช้ต้นทุนน้อยกว่า (Lower costs) ทำให้ลดปัญหาด้านปริมาณของข้อมูลที่จะส่งไปในระยะทางไกล ๆ ถือว่าเป็นการลดต้นทุนในการสื่อสารทางไกล เพราะข้อมูลบางส่วนสามารถประมวลผลจากเครื่องคอมพิวเตอร์ในเครือข่ายนั้น

3) ปรับปรุงความถูกต้องของข้อมูล (Improved data integrity) ผู้ใช้เครือข่ายเฉพาะพื้นที่ (LAN) สามารถตรวจสอบความผิดพลาดได้รวดเร็ว นอกจากนี้ยังสามารถแก้ไขได้ด้วย ซึ่งจะส่งผลไปสู่ความถูกต้องของข้อมูล เช่นชื่อสกุลที่สะกดแบบผิด ๆ ปริมาณการสั่งซื้อไม่ถูกต้อง ซึ่งจะสามารถพบได้โดยพนักงานที่ทำงานอย่างใกล้ชิดกับลูกค้าของตนเท่านั้น
4) ลดต้นทุนตัวประมวลผลหลัก (Reduced host processor costs) จะสามารถเพิ่มอายุการใช้งานของคอมพิวเตอร์ขนาดใหญ่ (Mainframe) เนื่องจากไม่มีภาระงานเกินภาวะการทำงาน
5) การเพิ่มความน่าเชื่อถือ (Increased reliability) หากคอมพิวเตอร์หลักในระบบการประมวลผลแบบรวมศูนย์ (Centralized processing systems) ล้มเหลวจะเกิดการขัดข้องทั้งระบบ แต่ในระบบการประมวลผลแบบกระจาย (Distributed processing systems) บางส่วนยังสามารถทำหน้าที่เป็นตัวเสริมหรือสนับสนุนในกรณีที่ตัวประมวลผลใดประสบปัญหาหรือล้มเหลวได้
6) การใช้ทรัพยากรร่วมกัน (Resource sharing) ประโยชน์หลักของการพัฒนาเครือข่ายคอมพิวเตอร์ขนาดเล็กในระบบแบบกระจาย คือ ประโยชน์จากการแบ่งสรรทรัพยากรที่มีราคาแพง อุปกรณ์ในการประมวลผลที่มีความเร็วสูง และอุปกรณ์ต่างๆ ในการทำงาน

2.1.4 ข้อเสียของการประมวลผลแบบกระจาย

1) 1.การขาดแคลนผู้เชี่ยวชาญด้าน MIS (Shortage of MIS professionals) ในขณะที่ความต้องการระบบในลักษณะการกระจายการประมวลผลเพิ่มมากขึ้น แต่ผู้เชี่ยวชาญในด้านนี้มีจำนวนน้อยไม่เพียงพอกับความต้องการ
2) มาตรฐานของระบบ (Standardization) มาตรฐานของระบบเป็นสิ่งที่ผู้บริหารจะต้องให้คำนึงถึงเป็นอย่างยิ่ง เพราะทรัพยากรคอมพิวเตอร์ที่นำมาต่อเชื่อมเป็นระบบในเครือข่ายเดียวกันจะทำงานร่วมกันได้ ก็ต่อเมื่อมีมาตรฐาน (Standardization) เดียวกัน มิฉะนั้นแล้วจะเป็นยากต่อการพัฒนาในองค์กรที่มีการประมวลผลแบบกระจาย
3) ความถูกต้องของข้อมูล (Data integrity) ในระบบที่ฐานข้อมูลกระจัดกระจายไม่เป็นระเบียบผู้ใช้มีความจำเป็นต้องทำการแก้ไขข้อมูล เพิ่ม หรือลบข้อมูล อย่างเป็นระบบ เป็นขั้นตอน มิฉะนั้นอาจจะเกิดความผิดพลาดได้โดยง่าย
4) ความปลอดภัยของระบบ (Security) ในระบบที่ทำการประมวลผลข้อมูลแบบกระจายจากส่วนกลางนั้น ผู้ใช้ระบบร่วมกันได้หลาย ๆ คน อาจเปิดโอกาสให้ผู้ที่ไม่หวังดีทำการบุกรุกระบบ ได้ เช่น นำไวรัสเข้าสู่ในระบบ ขโมยข้อมูล หรือแก้ไขข้อมูลโดยไม่ได้รับอนุญาต เป็นต้น

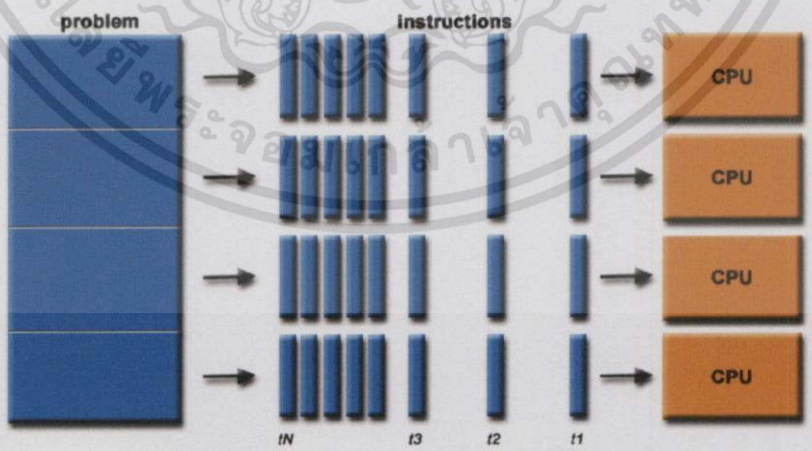
2.2 การประมวลผลแบบขนาน (Parallel Computing)

การประมวลผลแบบขนานคือรูปแบบการประมวลแบบหนึ่ง โดยจะแบ่งปัญหาขนาดใหญ่เป็นชิ้นย่อยเล็กๆ และประมวลปัญหาไปพร้อมๆกัน ซึ่งการแบ่งนั้นก็ทำได้หลายระดับ ทั้งในระดับหน่วยข้อมูล ระดับคำสั่ง ระดับข้อมูล จนไปถึงระดับ Task การประมวลผลแบบขนานนี้ถูกพัฒนาและใช้กันมาอย่างต่อเนื่อง โดยเฉพาะในคอมพิวเตอร์ประสิทธิภาพสูง แต่ในระยะหลังมานี้กลับมีนำหลักการประมวลผลแบบนี้มาพัฒนาในรูปแบบที่เป็นกายภาพมากขึ้น หรือก็คือแบ่งเป็นคอมพิวเตอร์หลายตัว เนื่องจากปัญหาทางด้านพลังงานซึ่งเริ่มเป็นที่วิตกกันในไม่กี่ปีที่ผ่านมา ในปัจจุบัน การประมวลผลแบบขนานกลายเป็นกระบวนทัศน์ที่โดดเด่นสำหรับสถาปัตยกรรมคอมพิวเตอร์ โดยเฉพาะในรูปแบบของมัลติคอร์โปรเซสเซอร์ (Multicore Processors)

การประมวลผลแบบขนานสามารถแบ่งระดับโดยหยาบได้โดยใช้ประเภทของฮาร์ดแวร์ที่แบ่งประมวลผล มัลติคอร์ (Multicore) และมัลติโปรเซสเซอร์ (MultiProcessor) เป็นการแบ่งประมวลแบบขนานที่ยังอยู่ในเครื่องเดียว ในขณะที่ คลัสเตอร์(Cluster) และ กริด(Grid) จะใช้คอมพิวเตอร์หลายๆเครื่องประมวลผลไปพร้อมกัน

สำหรับโปรแกรมสำหรับการประมวลผลแบบขนานย่อมเขียนยากกว่าแบบทั่วไป ด้วยข้อบกพร่องของซอฟต์แวร์ที่จากคลาสที่เกิดขึ้นใหม่ อีกทั้งการสื่อสารระหว่างเครื่องที่ต้องประสานกันอย่างพอดีและรวดเร็วพอที่จะทำให้ยอมรับได้ว่าประสิทธิภาพของการประมวลผลแบบขนานนั้นดีกว่า

แม้จะกล่าวว่าการเพิ่มหน่วยประมวลผลจะทำให้ประสิทธิภาพของการประมวลผลดีขึ้น แต่ที่จริงแล้วความเร็วสูงสุดจะถูกจำกัดตามกฎของอัมดัลห์ (Amdahl's law)

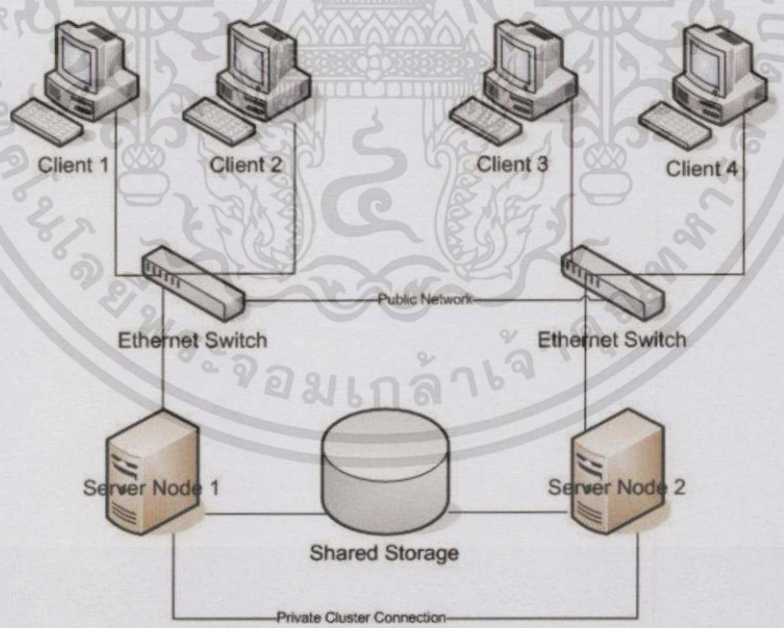


รูปที่ 2.5 แสดงการประมวลผลแบบขนานในระดับ CPU

2.3 ระบบประมวลผลคลัสเตอร์ (Cluster Computing System)

ระบบคลัสเตอร์ คือ การนำคอมพิวเตอร์หลาย ๆ ตัวมาเชื่อมต่อกันเพื่อให้ทำงานเหมือนเครื่องใหญ่เครื่องเดียว ความแตกต่างระหว่างระบบคลัสเตอร์กับระบบLAN คือ ระบบLAN เป็นการเชื่อมต่อคอมพิวเตอร์จำนวนมากเพื่อใช้ทรัพยากรร่วมกันโดยแต่ละเครื่องยังเป็นเครื่องอิสระ แต่ระบบคลัสเตอร์เป็นการรวมเครื่องหลาย ๆ เครื่องที่ประสานงานกันอย่างเหนียวแน่นจนเปรียบเสมือนเป็นเครื่องเดียวกัน ระบบคลัสเตอร์สามารถนำมาประยุกต์ใช้งานแทนระบบเซิร์ฟเวอร์ขนาดใหญ่ได้เป็นอย่างดี ดังนั้นอาจกล่าวได้ว่า ระบบคลัสเตอร์ คือคอมพิวเตอร์แบบขนานที่มีหน่วยจำแยกตามโครงสร้างของระบบคลัสเตอร์

โครงสร้างของระบบคลัสเตอร์ทั่วไปจะประกอบด้วยเครื่องพีซีสมรรถนะสูงจำนวนหนึ่งที่นำมาเชื่อมต่อกันด้วยเครือข่ายความเร็วสูง โครงสร้างพื้นฐานของระบบคลัสเตอร์และระบบ LAN ไม่แตกต่างกัน แต่ระบบคลัสเตอร์จะพึ่งพาซอฟต์แวร์พิเศษที่กระจายงานไปในกลุ่มของคอมพิวเตอร์ที่มารวมกันทำงานเป็นระบบคลัสเตอร์ดังนั้นคุณสมบัติสำคัญของเทคโนโลยี คลัสเตอร์มีอยู่สามประการ คือ เครือข่ายความเร็วสูง ระบบซอฟต์แวร์ที่สนับสนุนการทำงานแบบคลัสเตอร์และโปรแกรมประยุกต์ที่ใช้



รูปที่ 2.6 แสดงโครงสร้างของระบบคลัสเตอร์

2.4 ระบบจัดการข้อมูลฮาดูป (Hadoop System)

รูปที่ 2.7 แสดงโลโก้ของ Apache Hadoop

Apache Hadoop คือ ซอฟต์แวร์แบบโอเพ่นซอร์ส (Open- source Software) ของ Apache สำหรับการประมวลผลแบบกระจาย หรือ Distributed Computing เพื่อรองรับการจัดเก็บและประมวลผลข้อมูลขนาดใหญ่ โดยการเชื่อมต่อเครื่องแม่ข่ายจำนวนมากเข้าด้วยกัน ซึ่งฮาดูปถูกพัฒนามาจาก Google's MapReduce และ Google File System (GFS) เฟรมเวิร์ค ฮาดูปจึงเหมาะสมกับแอพพลิเคชั่นที่ต้องการความน่าเชื่อถือและมีข้อมูลเคลื่อนไหวตลอดเวลา

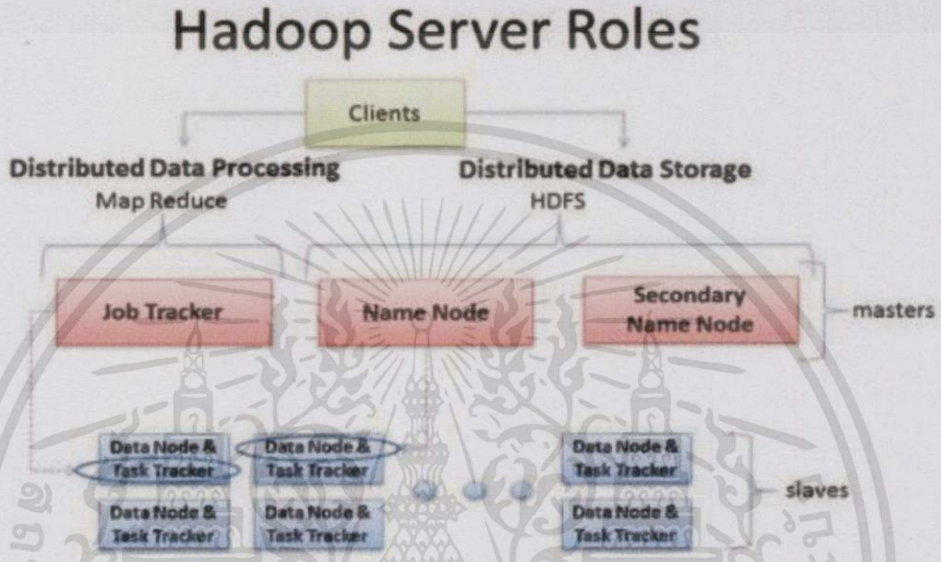
ระบบฮาดูปมีส่วนในการทำงานอยู่ 2 ส่วน คือ

1) ส่วนเตรียมระบบไฟล์แบบกระจาย (Distributed File System) ซึ่งเรียกว่า HadoopDistributed File System (HDFS) เป็นระบบจัดการไฟล์ข้อมูลในการเก็บข้อมูลบนระบบ โดยจะรองรับการส่งข้อมูลจำนวนมากระหว่างคลัสเตอร์ ด้วยวิธีการแบ่งข้อมูลออกเป็นหลายๆส่วน แล้วนำชิ้นข้อมูลเหล่านั้นกระจายออกไปเก็บยังโหนดต่างๆ นอกจากนี้ระบบเตรียมไฟล์แบบกระจาย สามารถสำรองข้อมูล (Data Replication) บนโหนดต่างๆด้วย
2) ส่วนประมวลผลข้อมูลที่เรียกว่า แมปรีดิวซ์ (Map Reduce) จะทำนำคำสั่งการประมวลผลกระจายออกไปตามโหนดต่างๆบนระบบ ซึ่งโหนดใดเก็บข้อมูลที่จะนำไปประมวลผลอยู่โหนดนั้นก็จะมีการประมวลผล ตามคำสั่ง นอกจากนี้ส่วนแมปรีดิวซ์ ยังสามารถเรียกการทำงานจากโหนดสำรอง ให้ทำการประมวลผลแทนได้ทันที เมื่อมีโหนดใดที่ทำงานล้มเหลว (Fault-Toralent)

สำหรับการส่งข้อมูลข้ามคลัสเตอร์ โดยทั้ง Map Reduce และ HDFS ได้ถูกออกแบบมาเพื่อให้เฟรมเวิร์คสามารถจัดการกับโหนดที่ทำงานล้มเหลวสามารถทำงานต่อไปได้โดยอัตโนมัติ โดยที่ข้อมูลไม่สูญหาย

2.4.1 โครงสร้างของฮาดูปคลัสเตอร์

เมื่อเราใช้งานระบบฮาดูปบนคลัสเตอร์ สามารถแบ่งเครื่องที่ทำงานได้เป็น 3 ประเภท คือ Client , Master node และ Slave node



รูปที่ 2.8 โครงสร้างของ Hadoop Sever Roles

1) Master node มีฟังก์ชั่น การทำงานอยู่ 2 ส่วนด้วยกันคือ การจัดการในการเก็บข้อมูลขนาดใหญ่ (HDFS) และส่วนที่ทำการประมวลผลข้อมูลเหล่านั้นแบบขนาน (Map Reduce) ซึ่ง มี Name Node ทำหน้าที่ดูแล และประสานการทำงานกับฟังก์ชั่น การจัดเก็บข้อมูล (HDFS) และ Job Tracker จะดูแล และประสานงานกับฟังก์ชั่น การประมวลผลข้อมูลแบบขนาน (Map Reduce)
2) Slave node เป็นตัวจัดการกับข้อมูล และประมวลผลข้อมูลทั้ง หมดจริงๆ แต่ละ Slave node จะทำงานบนทั้ง Data node และ Task tracker ซึ่ง Data node เป็นตัวที่ติดต่อสื่อสารกับ Name node และTask tracker เป็นตัวติดต่อสื่อสารกับ Job tracker ของ Master node
3) Client จะต้องทำการติดตั้ง ฮาดูปด้วยการตั้งค่าคลัสเตอร์ ไม่ว่าจะเป็น Master node หรือ Slave node ซึ่งเครื่อง Client จะทำการโหลดข้อมูลไปให้เครื่องคลัสเตอร์ แล้วส่ง MapReduce job ไปอธิบายว่า ควรจะถูกประมวลผลอย่างไร และจากนั้น เมื่อทำการประมวลผลเสร็จก็จะนำผลลัพธ์ที่ได้กลับมารวมกันสำหรับระบบที่คลัสเตอร์ที่มีขนาดเล็กโหนดหนึ่งโหนดสามารถเป็นได้ทั้ง Name node กับ Job tracker แต่สำหรับระบบคลัสเตอร์ที่มีขนาดใหญ่ต้องแยกการทำงานแต่ละงานอยู่บนคนละโหนด

2.4.2 หลักการทำงานของฮาดูป

การทำงานโดยภาพรวมของฮาดูปจะมีขั้นตอนดังนี้ ส่วนรายละเอียดของการทำงานในแต่ละส่วนจะกล่าวในหัวข้อที่ 2.5 ระบบจัดการไฟล์แบบกระจายของฮาดูป และหัวข้อที่ 2.6 หลักการแมปรีดิวซ์ของฮาดูป ต่อไป

1) นำข้อมูลเข้าสู่ระบบคลัสเตอร์ (HDFS writes)
2) วิเคราะห์ข้อมูลตามโปรแกรมที่เขียนด้วยแมปรีดิวซ์
3) เก็บผลลัพธ์ที่ได้จากการวิเคราะห์ในระบบคลัสเตอร์ของแมปรีดิวซ์ผ่าน HDFS
4) อ่านผลลัพธ์จากระบบคลัสเตอร์ (HDFS reads)

2.5 ระบบจัดการไฟล์แบบกระจายของฮาดูป (Hadoop Distributed File System)

Hadoop Distributed File system (HDFS) คือ file system ที่ถูกออกแบบมาสำหรับจัดเก็บข้อมูลที่มีขนาดใหญ่ให้เป็นรูปแบบ ซึ่งการทำงานบนระบบคลัสเตอร์ทั่ว ไป ซึ่ง มีลักษณะดังนี้

1) Streaming data access HDFS ถูกสร้างโดยใช้แนวคิดให้รูปแบบในการประมวลผลของข้อมูลให้มีประสิทธิภาพสูงสุด คือบันทึกข้อมูลหนึ่งครั้ง แต่อ่านได้หลายครั้ง ซึ่งกลุ่มข้อมูลที่ถูกสร้างขึ้นหรือถูกคัดลอกมาจากต้นแบบ
2) Commodity hardware ฮาดูปไม่จาเป็นต้องใช้เครื่องที่มีทรัพยากรที่สูงมาก เพียงแต่ใช้เครื่องที่นำไปทำงานบน ระบบคลัสเตอร์ได้ โดยปกติแต่ละโหนดนั้น มีโอกาสที่จะล่มได้ แต่เมื่อใช้ระบบคลัสเตอร์ โอกาสการเกิดความล้มเหลวของระบบคลัสเตอร์มีอยู่น้อยมาก เมื่อเครื่องใดเครื่องหนึ่งในระบบล้มเหลวไป เครื่องอื่นที่ว่างจากการทำงานอยู่ก็สามารถทำงานแทนเครื่องที่ล้มเหลวได้
3) Lots of small files มีการแบ่งไฟล์เป็นหลายๆชิ้น เพื่อทำข้อมูลที่จะส่งไปประมวลผลขนาดเล็กลง ทั้งนี้จำนวนไฟล์ที่แบ่งได้ขึ้นอยู่กับขนาดของหน่วยความจาของ Name node

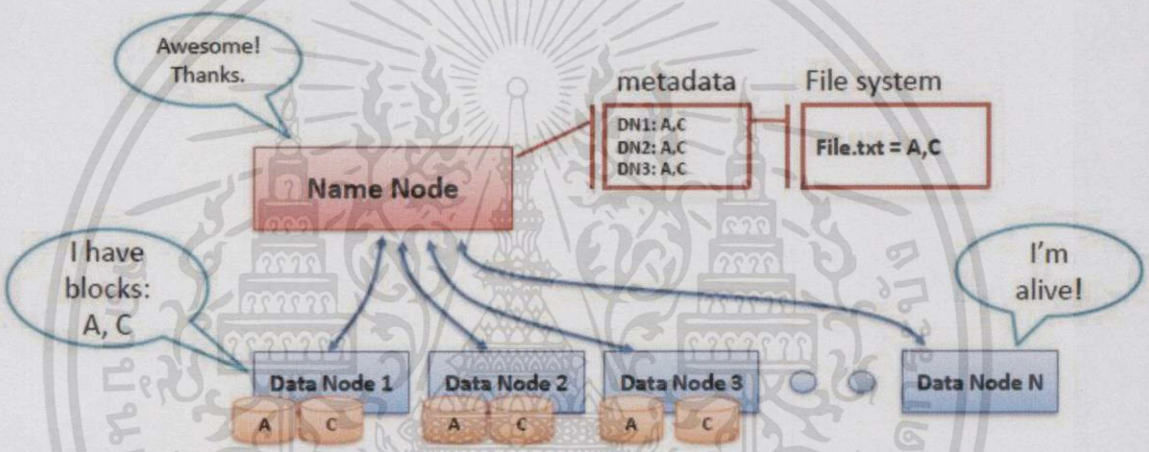
2.5.1 โครงสร้างเชิง HDFS

HDFS Cluster มีโหนดการทำงานอยู่ 2 แบบด้วยกัน คือ

1) Namenode คือ โหนดที่ทำงานในส่วน HDFS บน Master node ซึ่ง Name node ดูแล file system ทั้งหมดของคลัสเตอร์ และดูแลเครื่อง Data node และทำงานประสานงานการเข้าถึงข้อมูล โดย Name node จะเป็นศูนย์กลางการควบคุมของHDFS โดย Name node จะไม่มีการเก็บไฟล์ข้อมูลใดๆจาก client แต่ Name node จะเก็บเพียงที่อยู่ของแต่ละ กลุ่มบล็อคที่ได้จากการแบ่งไฟล์เป็นชิ้นเล็กๆเท่านั้น ว่าไปอยู่ที่โหนดใดบ้าง ซึ่ง Name node จะติดต่อกับ Client เพื่อบอกที่อยู่ของ Data node ที่เก็บกลุ่มบล็อคข้อมูลที่ต้องการ , เก็บ

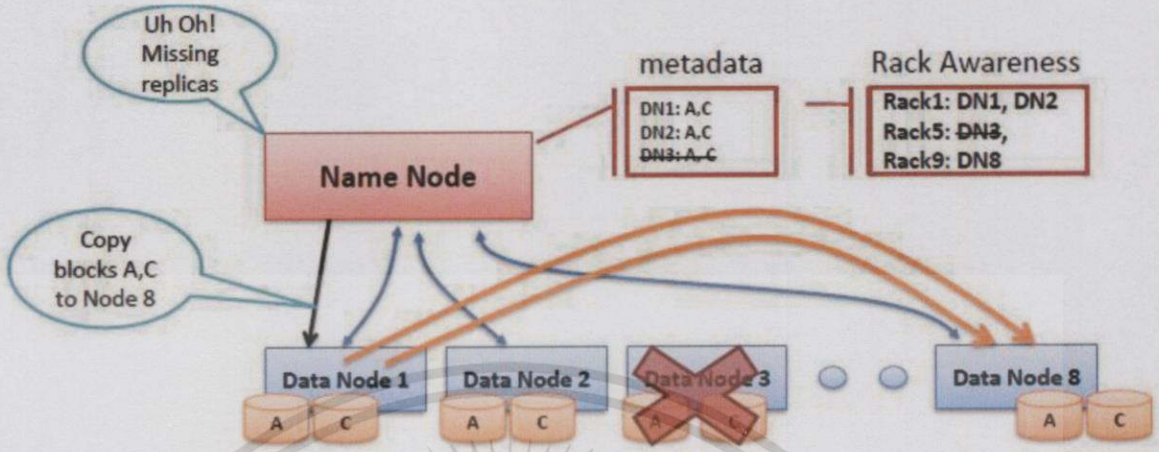
สถานะของคลัสเตอร์ , ดูแล Data node และทำให้แน่ใจว่ามีการทำสำเนากลุ่มบล๊อคแต่ละกลุ่มบล๊อคเรียบร้อยดีแล้วถ้าหากไม่มี Name node แล้ว ก็จะไม่สามารถ เขียนหรืออ่านไฟล์จาก HDFS ได้ และจะไม่สามารถทำการประมวลผลผ่านแมปรีดิวซ์ ได้อีกด้วย

2) **Data node** จะคอยติดต่อกับ Name node เพื่อบอกสถานะของโหนดนั้นๆ ยังสามารถทำงานได้ และบอกข้อมูลกลุ่มบล๊อคให้แก่ Name node ทุกๆ 3 วินาที ผ่าน TCP handshake บนหมายเลขพอร์ต TCP 9000 ทุกๆ 10 สัญญาณการติดต่อ Data node จะส่งรายงานกลุ่มบล๊อคที่อนุญาตให้ Name node สามารถสร้างสำเนา และส่งไปยังโหนดต่างๆได้



รูปที่ 2.9 แสดงทำการแบ่งไฟล์ File.txt

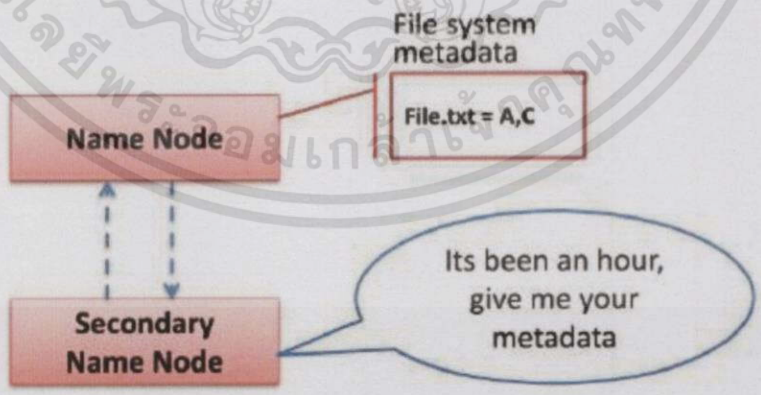
จากรูปที่ 2.9 Client ทำการแบ่งไฟล์ File.txt ออกเป็น 2 กลุ่มบล๊อคสำหรับแต่ละกลุ่มบล๊อค นั้น Client จะติดต่อกับ Name node เพื่อทราบรายชื่อของ Data node ทั้งหมด ซึ่ง Data node เหล่านั้น ควรจะมีสำเนาของแต่ละกลุ่มบล๊อคด้วย ซึ่ง Client จะสร้างกลุ่มบล๊อคแล้วส่งไป Data node โดยตรง สำหรับการสำรองข้อมูลนั้น Data node จะได้รับสำเนาของกลุ่มบล๊อคจาก Data node อื่นๆ และทำแบบนี้ซ้ำกับกลุ่มบล๊อคที่เหลือ สำหรับ Name node ไม่ได้เก็บข้อมูลไว้ แต่ Name node ให้เพียงที่อยู่ของข้อมูลและบอกว่าข้อมูลที่ Client ต้องการนั้นอยู่ที่ใดในคลัสเตอร์เท่านั้น



รูปที่ 2.10 เมื่อ Data node ขาดการติดต่อ

จากรูปที่ 2.10 ถ้า Name node ไม่ได้รับสัญญาณจาก Data node มันจะตัดสินว่า Data node ตัวนั้นไม่สามารถทำงานได้และข้อมูลทั้งหมดหายไปแล้ว ดังนั้น รายงานข้อมูลรายละเอียดการทำงานของกลุ่มบล๊อคที่มันได้รับมาจาก Datanode ที่หยุดทำงานไปนั้น Namenode จะสามารถหาที่อยู่ของสำเนาของกลุ่มบล๊อคเหล่านั้นได้ ดังนั้น Name node สามารถถ่ายทอดงานจาก Data node ที่หยุดทำงานไป ไปยัง Data node ตัวอื่นทำแทนได้

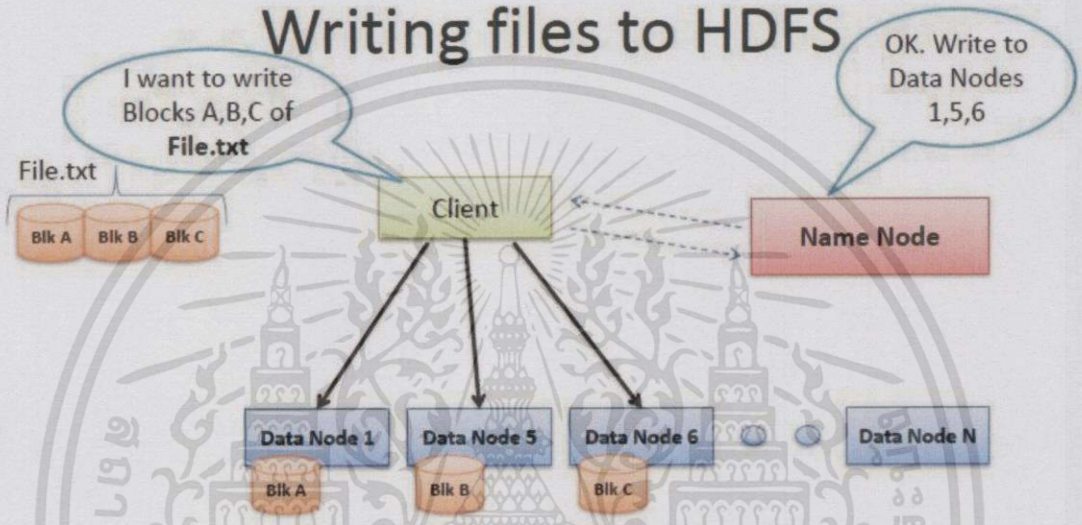
นอกจากนี้ HDFS ยังมีโหนดสำรอง (Secondary Namenode) ไว้สำหรับ Backup Name node เพื่อป้องกันการสูญหายของรายการกลุ่มบล๊อคใน Name node ดังรูปที่ 2.11



รูปที่ 2.11 แสดงการติดต่อระหว่าง Name Node และ Secondary Name Node

2.5.2 การเขียนไฟล์เข้าสู่ HDFS (HDFS writes)

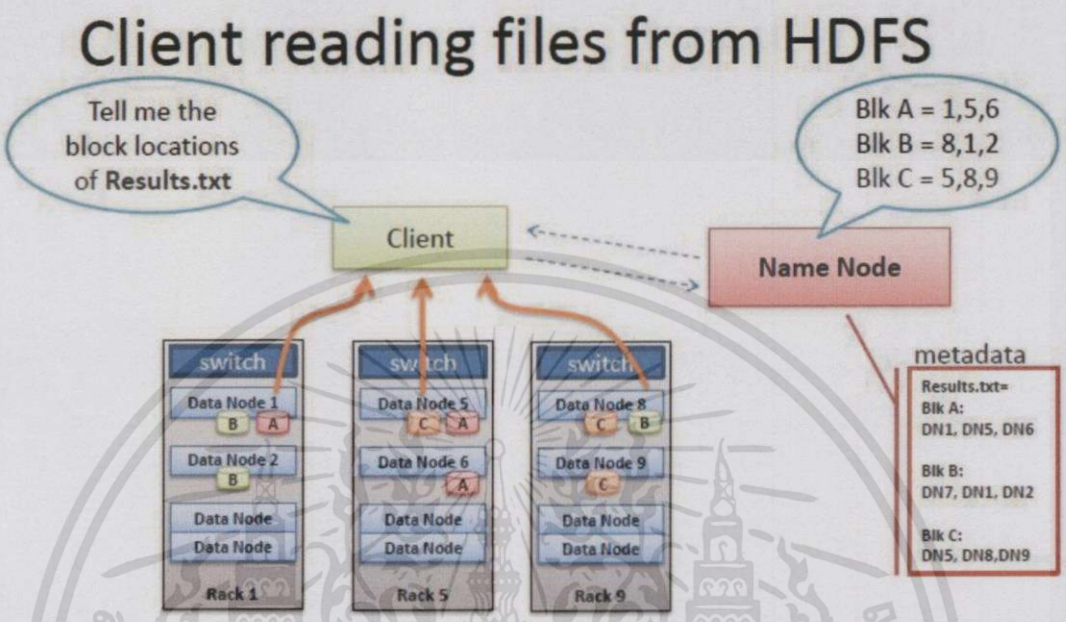
ยกตัวอย่าง Client จะส่งไฟล์ข้อมูล (File.txt) เข้าสู่คลัสเตอร์ และส่งรายละเอียดของข้อมูลที่ต้องการวิเคราะห์เป็นโปรแกรมภาษาจาวา แล้วระบบคลัสเตอร์จะส่งผลลัพธ์ที่ได้จากการประมวลผลลงเป็นในไฟล์ใหม่ (Result.txt) ให้ Client



รูปที่2.12 แสดงการเขียนไฟล์เข้าสู่ HDFS

รูปที่ 2.12 เมื่อทำการโหลดไฟล์ข้อมูลเข้าไปในคลัสเตอร์นั้นแล้ว Client จะทำการแบ่งไฟล์ข้อมูลออกเป็นชิ้นเล็กๆ เรียกว่า กลุ่มบล๊อค แล้วแจกจ่ายแต่ละกลุ่มบล๊อคเหล่านั้นให้กับแต่ละเครื่อง Slave ในระบบ เพื่อให้ทำการประมวลผลแบบขนานกันได้ ในขณะเดียวกัน แต่ละเครื่องที่ทำงานอยู่ในระบบคลัสเตอร์นั้นก็สามารถล้มเหลวเมื่อไหร่ก็ได้เช่นกัน ดังนั้นจึงมีการสำรองข้อมูลแต่ละกลุ่มบล๊อคอยู่ในเครื่องอื่นด้วยเพื่อหลีกเลี่ยงการสูญเสียข้อมูล โดยปกติยาดูปจะทำสำเนากลุ่มบล๊อคไว้กลุ่มบล๊อคละ 3 สำเนา และแจกจ่ายอยู่คนละ Data node กัน

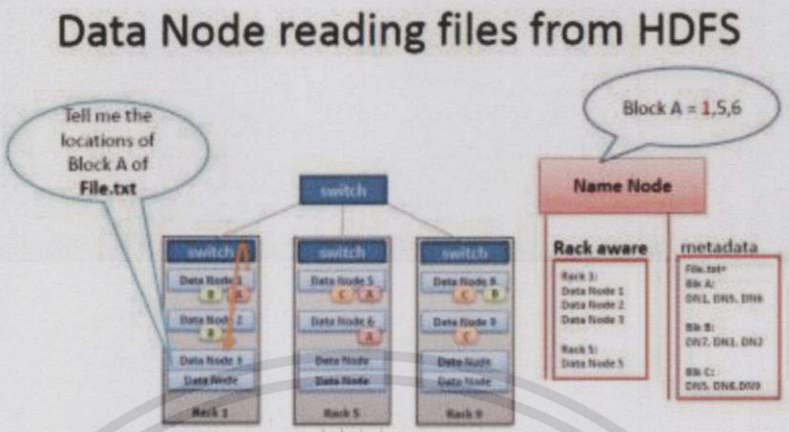
2.5.3 การอ่านไฟล์จาก HDFS (HDFS reads)



รูปที่ 2.13 แสดงแผนภาพตัวอย่างการอ่านไฟล์ของ Client จาก HDFS ของไฟล์ Result.txt

จากรูปที่ 2.13 แสดงถึงการดึงไฟล์ของ Client จาก HDFS ซึ่งอาจจะเป็นผลลัพธ์ของการประมวลผล Client จะไปติดต่อกับ Name node เพื่อตรวจหาที่อยู่ของ กลุ่มบล๊อค ข้อมูลที่เก็บไฟล์นั้น Name node จะส่งรายการ Data node ที่เก็บ กลุ่มบล๊อค ของไฟล์เหล่านั้นให้ Client ให้ Client สามารถหยิบ Data node จาก กลุ่มบล๊อค แต่ละรายการ กลุ่มบล๊อคและอ่านทีละ กลุ่มบล๊อค โดยที่มันจะไม่อ่าน กลุ่มบล๊อค ถัดไปจนกว่าจะอ่าน กลุ่มบล๊อค เดิมเสร็จ

ในบางครั้ง Data node บางตัวจำเป็นต้องการประมวลผลข้อมูล กลุ่มบล๊อค อื่นร่วมด้วย ถ้า Block ที่ต้องการอ่านจาก HDFS นั้น ที่ไม่ได้อยู่ใน Data node เดียวกัน Data node จะติดต่อกับ Name node เพื่อหาที่อยู่ของ Block นั้น ดังรูปที่ 2.14



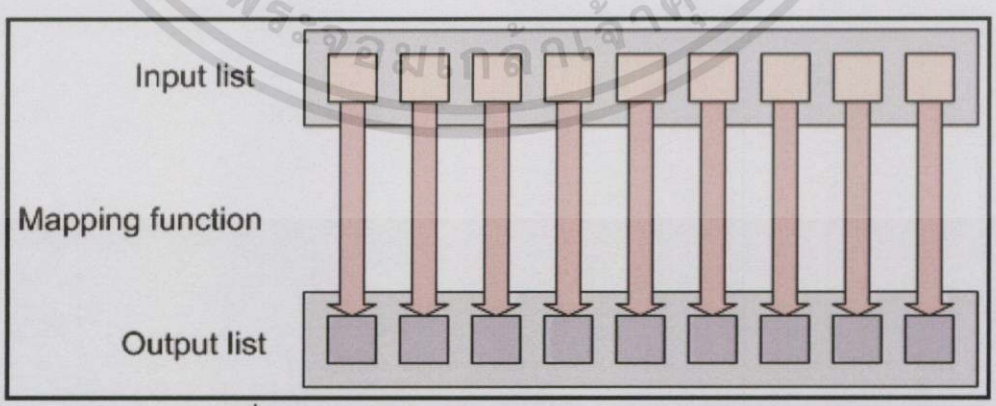
รูปที่ 2.14 แสดงแผนภาพตัวอย่างการอ่านไฟล์ของ Data node จาก HDFS ของไฟล์ File.txt

2.6 หลักการแมปรีดิวซ์ (แมปรีดิวซ์) ของระบบฮาดูป

MapReduce คือ โมเดลการเขียนโปรแกรมที่นำมาใช้ในการวิเคราะห์ข้อมูลขนาดใหญ่ และมักจะใช้ในการกระจายการประมวลผลไปยังเครื่องคอมพิวเตอร์เครื่องอื่นๆในเครือข่าย โดย MapReduce นี้มีต้นแบบมาจากการ Map และ Reduce Function ที่มักพบอยู่ในภาษาประเภท Functional Programming อย่างเช่น Ruby และ Python เป็นต้น

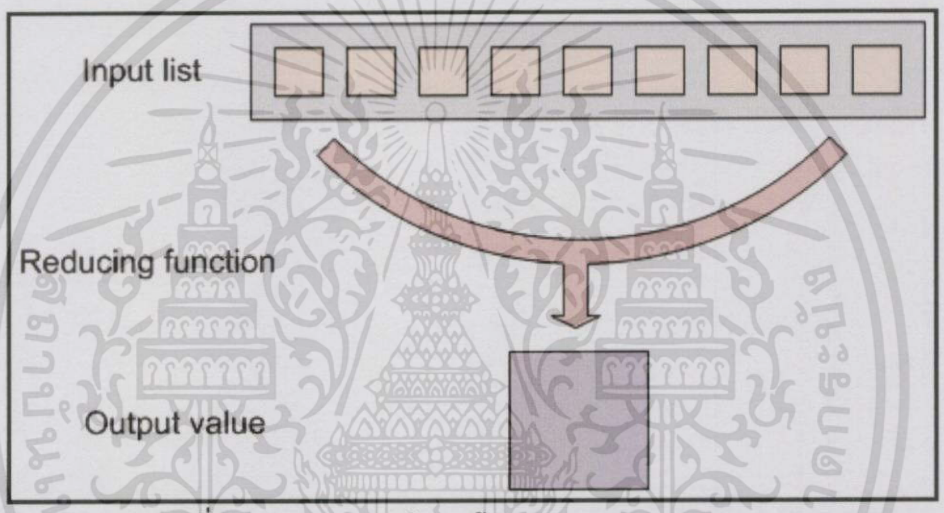
การทำงานของ แมปรีดิวซ์ แบ่งออกเป็น 2 ส่วนด้วยกัน คือ Map และ Reduce

ขั้นตอนการ Map คือ การที่เครื่อง Master node นำอินพุตคำสั่งที่ได้รับเข้ามาแบ่งเป็นส่วนๆ แล้วส่งอินพุตคำสั่งย่อยๆ เหล่านั้นกระจายไปยัง Slave node ต่างๆ ซึ่ง Slave node เหล่านั้นทำหน้าที่เป็น Worker node ที่เก็บข้อมูลในส่วนที่ต้องการจัดการ (Data node) เพื่อให้ Slave node เหล่านั้นทำการประมวลผลข้อมูลเหล่านั้น



รูปที่ 2.15 แสดงการทำงานโดยภาพรวมของ Map

ขั้นตอนการ Reduce คือ ที่ Master node จะนำเอาผลลัพธ์ที่ได้จากการประมวลผลการทำงานของ Slave node ต่างๆ มารวมกัน และสรุปเป็นผลลัพธ์สุดท้าย แล้วส่งคืนกลับไปยังเครื่องที่ร้องขอตอนแรก แมปรีดิวซ์ มักจะถูกนามาใช้ในกรณีที่ข้อมูลมีขนาดใหญ่มาก โดยที่เครื่องเซิฟเวอร์เพียงเครื่องเดียว ไม่สามารถจะจัดการข้อมูลได้ภายในเวลาที่กาหนด จึงจาเป็นต้องมีการใช้เครื่อง Workerจำนวนหลายๆเครื่องมาช่วยในการประมวลผลเรามักจะพบการใช้งาน แมปรีดิวซ์ ในหลายๆ เทคโนโลยีที่รองรับการ Scale ขนาดใหญ่ อย่างเช่น Apache Hadoop Project , MongoDB เป็นต้น

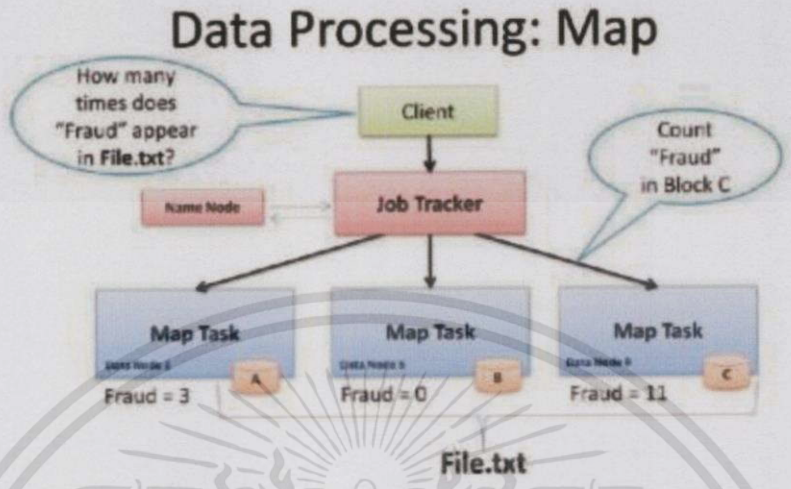


รูปที่ 2.16 แสดงการทำงานโดยภาพรวมของ Reduce

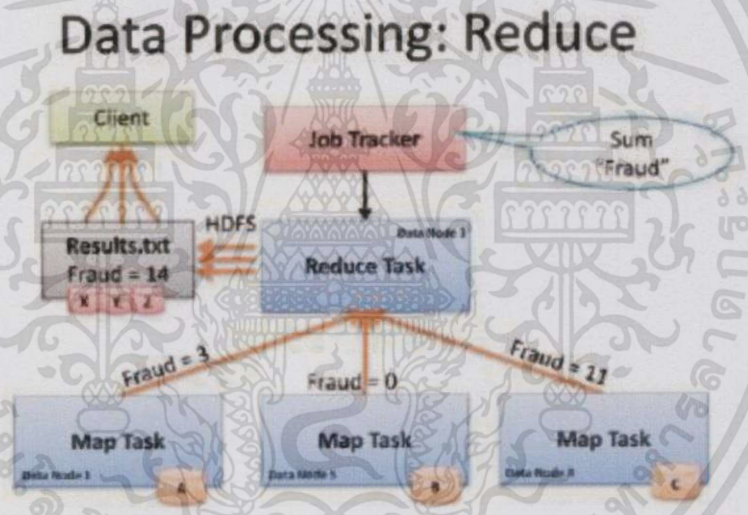
2.6.1 โครงสร้างเชิง แมปรีดิวซ์

ในขั้นตอน แมปรีดิวซ์ สามารถแบ่งโหนดทำงานได้เป็น 2 ลักษณะ ได้แก่

1) Job Tracker Node อยู่ในระดับชั้น Master Node ทำหน้าที่เป็นตัวกลางระหว่าง Client กับ Task Tracker คอยส่ง Java code ที่ใช้ให้ Task Tracker และบริหารจัดการให้งานกับโหนดใหม่ เมื่อมีโหนดที่ล้มเหลว (fault-tolerant)
2) Task Tracker Node อยู่ในระดับชั้น Slave Node รับ Java code จาก Job Tracker มาทำการประมวลผล โดยจะสร้าง Map Task ให้ทำงานกับ Block ข้อมูลที่มี และคอยส่งสัญญาณบอก Job Tracker ว่ายังทำงานอยู่



(ก)



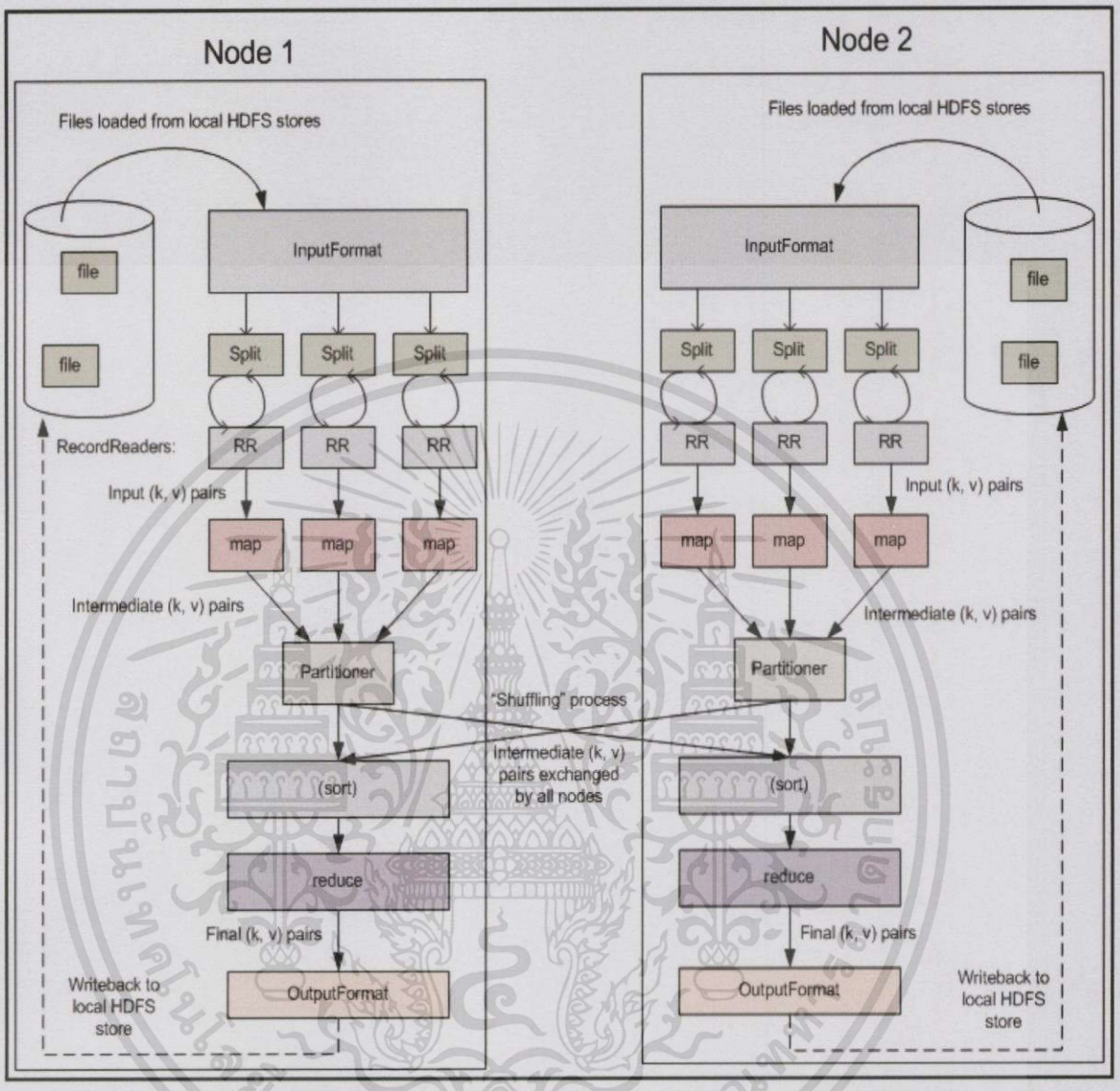
(ข)

รูปที่ 2.17 แสดงการทำงานของคลัสเตอร์
(ก) แสดงการทำงานของคลัสเตอร์ช่วง Map
(ข) แสดงการทำงานของคลัสเตอร์ช่วง Reduce

จากรูปที่ 2.17 (ก) File.txt ถูกแบ่งเป็น Block แล้วกระจายไปยัง Slave node ต่างๆ ซึ่งทำให้ระบบสามารถประมวลผลแต่ละส่วนได้ในเวลาเดียวกัน ในที่นี้ ทุกๆ node จะประมวลผล Data block ที่สนใจในระบบทั้งหมด โดยในกรณีนี้จะให้นับคำว่า Refund ใน Data block ของ File.txt เริ่มจากการร้องขอของเครื่อง Client จะส่ง MapReduce job ให้กับ Job Tracker แล้วหาว่า ในไฟล์ File.txt เจอคำว่า "Refund" กี่ครั้ง ซึ่งสิ่งที่ส่งไปจะเป็น Java code จากนั้น Job Tracker จะติดต่อกับ Name node เพื่อทราบว่าใน Data node มีอยู่กี่ Data block ที่แบ่งมาจากไฟล์ File.txt

จากนั้น Job Tracker ก็จะติดต่อกับ Task Tracker ที่ทำงานอยู่ที่ Slave node ตามคำสั่งของ Java code สำหรับทำงาน Map ฟังก์ชั่นบน local data ต่อมา Task Tracker เริ่มเปิด Map task และ Job Tracker จะคอยติดตามการประมวลผลของ Task Tracker โดย Task Tracker จะส่งสัญญาณและสถานะถามกลับไปยัง Job Tracker เพื่อจะได้รู้ว่า Map Task ยังทำงานอยู่หรือไม่

เช่นเดียวกับรูปที่ 2.17 (ข) ในส่วนของ Reduce เมื่อ Map task ทำงานเสร็จสิ้นแล้ว และได้สร้าง Intermediate data ขึ้นมา ต่อมาเป็นขั้นตอนนำ Intermediate data จากแต่ละ node มารวมกัน Job Tracker เริ่ม Reduce Task บนโหนดใดโหนดหนึ่งบนคลัสเตอร์ และสั่งให้ Reduce task ไปหยิบข้อมูลจากintermediate dataซึ่งถูกประมวลผลโดยMap taskและMap task จะตอบสนองต่อ Reducer เกือบจะพร้อมกัน Reduce task ก็จะได้รวบรวมintermediate data ทั้งหมดจากMap taskและสามารถเริ่มการประมวลผลครั้งสุดท้ายได้ เช่นการหาผลรวมของคำ Refund แล้วเขียนผลลัพธ์ลงResult.txt Output จาก Job ที่เรียกว่า Result.txt จะถูกเขียนไปที่ HDFS และเครื่อง Client ก็จะสามารถอ่าน Result.txt จาก HDFS และ Job ก็จะเสร็จ



รูปที่ 2.18 แสดงขั้นตอนการทำงานของ MapReduce ตั้งแต่นำไฟล์ข้อมูลมาจาก Block จนประมวลผลเสร็จและให้ HDFS จัดเก็บข้อมูลอีกครั้ง

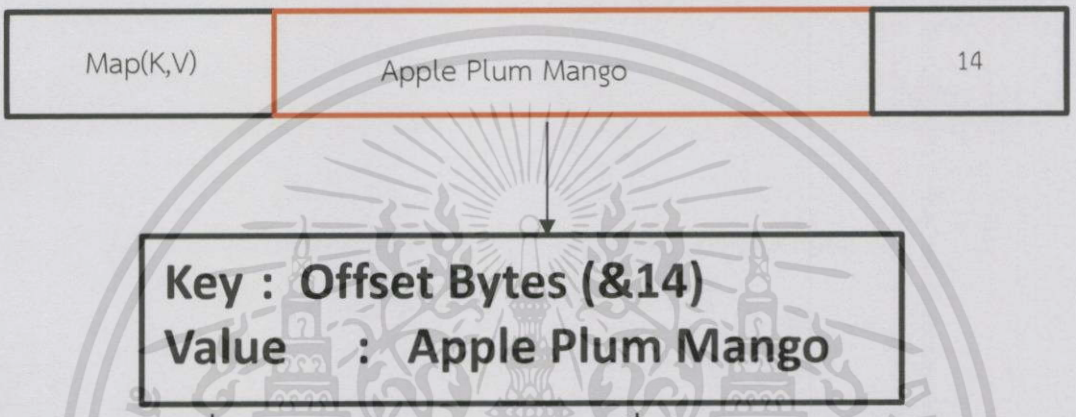
2.6.2 หลักการ Map

2.6.2.1 Input Split

Input Split คือหน่วยการทำงานที่แบ่งข้อมูลจากกลุ่มบล๊อคในแต่ละโหนด ย่อยเข้าสู่ระดับ map task ซึ่งอาจมีขนาดทั้งกลุ่มบล๊อคแต่ส่วนใหญ่จะถูกแบ่งไฟล์ย่อยๆเสียมากกว่า โดยปกติแล้ว Input split จะตัดขนาดไฟล์ที่ 64 MB (ขนาดเท่า 1 Block Size ของ HDFS) ซึ่งสามารถตั้งค่าขนาดได้ การแยกไฟล์เป็นชิ้นย่อยๆ ทำให้ map task สามารถทำงานพร้อมกันในรูปแบบขนานได้ ยิ่งไปกว่า

นั้น การที่ HDFS แบ่งไฟล์เป็นกลุ่มบล๊อคย่อยแยกไปอยู่ตามโหนดต่างๆในระบบคลัสเตอร์เป็นการยืนยันรูปแบบการประมวลผลแบบขนานได้เป็นอย่างดี นอกจากนี้ เราสามารถกำหนดจำนวน Task สูงสุดในแต่ละโหนดเพื่อประสิทธิภาพในการประมวลผลได้อีกด้วย

2.6.2.2 Record Reader



รูปที่ 2.19 แสดง Input และ Output เมื่อผ่าน LineRecordReader

จากเนื้อหาข้างต้นสมารถสรุปได้ว่า Input Split เป็นขั้นตอนการแบ่งงาน แต่ก็ไม่ได้อธิบายส่วนของการเข้าถึงข้อมูลเอาไว้ Record Reader จะทำการนำข้อมูลที่แบ่งไว้ในแต่ละ Task มาเปลี่ยนให้อยู่ในรูปแบบ <Key , Value> เพื่อนำไปประมวลผลในขั้นต่อไป

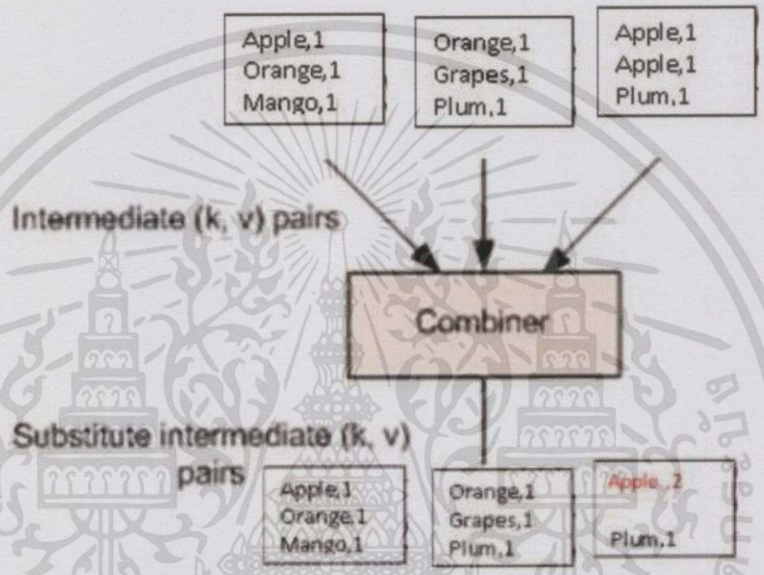
Record Reader ถูกกำหนดโดย InputFormat ที่สามารถกำหนดเองเพื่อใช้ประยุกต์ในโปรแกรมต่างๆที่ต้องการ ซึ่งพื้นฐานจะกำหนดเป็น TextInputFormat โดยใช้ LineRecordReader แบ่งไฟล์แตกแต่ละบรรทัดให้เป็น Value จับคู่กับ Key ที่เป็น Offset byte ของบรรทัดนั้นๆ แล้วเขียนลง Iterator จับคู่ไปทุกบรรทัดจนกระทั่งหมด สุดท้ายจึงส่งให้ฟังก์ชั่น Map() ดำเนินการต่อ

2.6.2.3 Mapper

หากจะกล่าวถึงขั้นตอนการ Map โดยภาพรวม ขั้นตอน Mapper จะถูกยกขึ้นมาเสมอ เริ่มจากการนำ <key , value> ที่ถูกส่งต่อมาจาก Record Reader มาประมวลผลตาม Java Code ที่ Job tracker node ส่งมาให้ทุก Task tracker node ผลลัพธ์ที่ได้(เรียกอีกชื่อว่า Intermediate data) จะถูกเขียนในรูป <key , value> อีกครั้งโดยคลาส OutputCollector และส่งต่อผลลัพธ์ไปยังส่วนถัดไป

2.6.2.4 Combiner

ขั้นตอน Combiner เป็นขั้นตอนเสริมผู้พัฒนาอาจนำมาใช้หลังจากขั้นตอน Mapper มีวัตถุประสงค์หลักเพื่อลดแบนวิธของ Intermediate Data ก่อนจะถูกนำส่งไปยัง Reducer โดยจะทำการรวมค่า Value ที่มี Key ซ้ำกันไว้ด้วยกัน ดังรูปที่ 2.X ซึ่งการกระทำนี้จะทำภายใน Intermediate Data เดียวกันเท่านั้น เพราะมีการกระทำต่อ Value ขั้นตอนนี้จึงถูกเรียกอีกชื่อหนึ่งว่า Mini-reduce



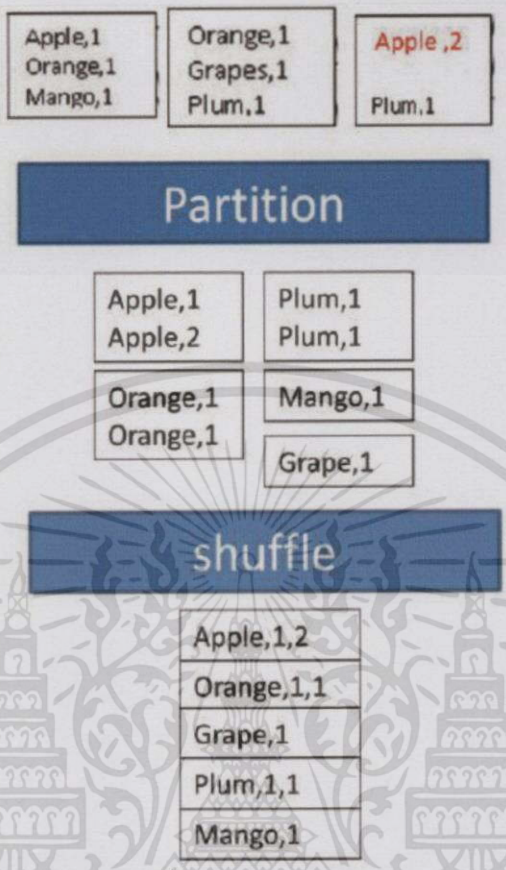
รูปที่ 2.20 แสดงผลลัพธ์ของ Intermediate data ที่เปลี่ยนไปเมื่อผ่าน Combiner

2.6.3 หลักการ Reduce

2.6.3.1 Partition & Shuffle

แม้จะอยู่ในหัวข้อ Reduce แต่แท้จริงแล้วขั้นตอนนี้จะอยู่ระหว่างการ Map และ Reduce โดยในแต่ละโหนดจะทำการจับ Intermediate data ที่มี Key เดียวกันมาไว้ด้วยกันเป็น Subset เรียกว่า Partitions ก่อนจะเริ่มขั้นตอนส่งไปยัง reducer ที่เรียกว่า Shufflling

ในขั้นตอนการส่งนั้น สามารถเริ่มได้ทันทีโดยไม่ต้องรอ Map Task ทั้งหมดของโหนดเสร็จสิ้น ซึ่งเมื่อถึงปลายทาง reducer จะทำการรวม Value ของ Intermediate data ที่ key เหมือนกันเขียนใหม่ให้อยู่ในรูป <Key, Value(Iterator)> ดังรูปที่ 2.21



รูปที่ 2.21 แสดงขั้นตอนการ Partition และ Shuffle

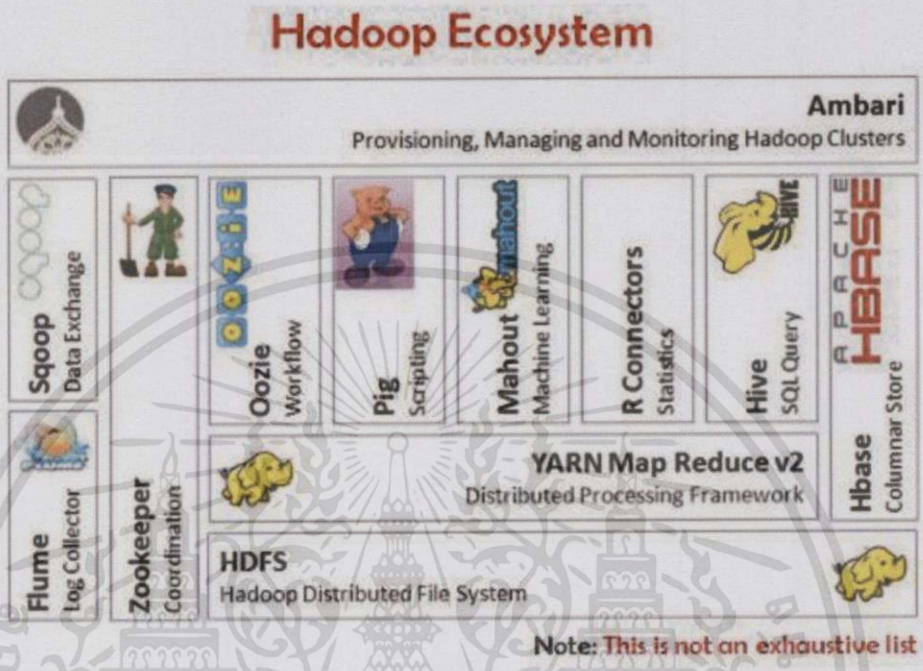
2.6.3.2 Sort

ระบบจัดการฮาดูปมีขั้นตอนการเรียงลำดับ Key ก่อนที่จะส่งให้ Reduce ประมวลผล ซึ่งผู้พัฒนาสามารถเขียน Override ประยุกต์วิธีการเรียงลำดับได้

2.6.3.3 Reduce

เป็นอีกขั้นตอนหนึ่งที่จะนำ Intermediate file ที่ได้มาประมวลผล ตาม Reduce Code ที่ผู้พัฒนาเขียนขึ้น โดยคราวนี้จะทำการจัดการกับค่า Value ทั้งหมดที่อยู่ใน Iterator เดียวกัน นับเป็นการ Reduce 1 key 1 ครั้ง และท้ายที่สุดจะออกมาเป็นค่า <Key , Value> ให้ Record Writer ทำการเขียนออกมาเป็น Output สุดท้ายเพื่อส่งไปยัง HDFS เป็นอันเสร็จสิ้น

2.7 Hadoop Ecosystem



รูปที่ 2.22 แผนผังแอพพลิเคชั่นที่เป็น Hadoop EcoSystem

ในต่างประเทศ ระบบฮาดูปกำลังเป็นที่จับตามองเพราะสามารถตอบโจทย์การประมวลผลข้อมูลขนาดใหญ่ที่กำลังขยายตัวได้อย่างรวดเร็ว แต่แน่นอนว่าย่อมไม่หยุดเพียงแค่นี้ โปรแกรมเมอร์หลายกลุ่มได้พยายามพัฒนาแอพพลิเคชั่นที่ทำงานบนระบบฮาดูป เพื่อให้ฮาดูปเข้าใกล้ผู้ใช้มากขึ้นเรื่อยๆ จากรูปที่ 2.22 แสดงถึงแผนผังระบบนิเวศน์เชื่อมโยงของฮาดูป (Hadoop EcoSystem) ซึ่งมีแอพพลิเคชั่นหลายตัวที่ทำงานติดต่อกับระบบ Hadoop หรือจะเป็น Zoo Keeper (ขวามือสุด) ที่จะคอยดูแลระบบประมวลผลฮาดูปรวมไปถึงแอพพลิเคชั่นอื่นๆด้วย

2.7.1 Ambari

รูปที่ 2.23 Apache Ambari

ซอฟต์แวร์อาปาเช่ อัมบารี่ (Apache Ambari) ถูกพัฒนาขึ้นให้สามารถจัดการระบบฮาดูปได้ง่ายขึ้น ไม่ว่าจะเป็นขั้นตอนการติดตั้งหรือตั้งค่าเครื่อง หน้าต่างแสดงสถานะของแต่ละโหนดและแจ้งเตือนเมื่อเกิดปัญหา เรียกได้ว่าสร้างอินเตอร์เฟซผู้ใช้ขึ้นมาอำนวยความสะดวก

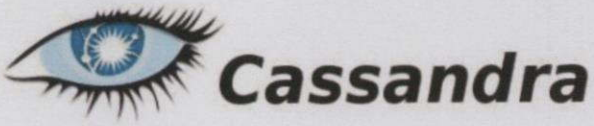
ปัจจุบันอัมบารี่รองรับการทำงานของ HDFS, MapReduce, Hive, HCatalog, HBase, ZooKeeper, Oozie, Pig และ Sqoop ซึ่งคาดว่าอนาคตจะสามารถรองรับการทำงานของระบบนิเวศน์ฮาดูปได้มากขึ้น

2.7.2 Avro

รูปที่ 2.24 Apache Avro

อาปาเช่ อาโวร่ (Apache Avro) คือการควบคุมการทำงานระยะไกล และพัฒนาการส่งในฮาดูป โดยอาโวร่จะใช้เจสัน (JSON) เพื่อระบุประเภทของข้อมูลและโปรโตคอล และส่งข้อมูลในรูปแบบไบนารี่ที่กะทัดรัด กล่าวคือ อาโวร่คือระบบส่งข้อมูลที่นำมาใช้ในฮาดูป ซึ่งเป็นไปได้ทั้งการส่งข้อมูลทั่วไป ข้อมูลการติดต่อสื่อสารระหว่างโหนด หรือ จากโปรแกรมไปยังส่วนบริการฮาดูป

2.7.3 Cassandra



รูปที่ 2.25 Apache Cassandra

อาปาเช่ แคสแซนดร้า (Apache Cassandra) เป็น Open-Source ของระบบจัดการฐานข้อมูลแบบกระจาย โดยระบบนี้ถูกออกแบบมาเพื่อรองรับข้อมูลจำนวนมากที่ถูกส่งไปมาระหว่างหลายๆเครื่องเซิร์ฟเวอร์ ซึ่งแพลตฟอร์มนี้จะสนับสนุนระบบศูนย์ข้อมูลหลายทอดที่อยู่บนกลุ่มคลัสเตอร์ โดยใช้วิธีการทำสำเนาไม่พร้อมกัน (asynchronous replication) ทำให้การดำเนินงานแอบแฝงในกลุ่มผู้ใช้ต่ำ

2.7.4 Chukwa

รูปที่ 2.26 Apache Chukwa

อาปาเช่ ชักวา (Apache Chukwa) เป็นระบบจัดเก็บข้อมูล สำหรับจัดการ การประมวลผลแบบกระจายที่มีขนาดใหญ่ ซึ่งเหมาะกับการจัดเก็บไฟล์ล็อกเพื่อให้ใช้สำหรับวิเคราะห์ข้อมูลต่อไป โดยการนำไฟล์ล็อกที่จะมาจากที่ต่างๆ (Agent) มาเก็บในบ่อข้อมูล (Data Sink) แล้วทำการแปลงข้อมูลไฟล์ล็อกที่ไร้โครงสร้าง มาเป็นข้อมูลชนิดโครงสร้าง

2.7.5 Flume

รูปที่ 2.27 Apache Flume

อาปาเช่ ฟลูม (Apache Flume) เป็นบริการสำหรับ เก็บ รวบรวม เคลื่อนย้าย ข้อมูลไฟล์ล็อกที่มีขนาดใหญ่มหาศาลได้อย่างมีประสิทธิภาพ ซึ่งง่ายและยืดหยุ่นต่อโครงสร้างการไหล (Streaming) ข้อมูล และสามารถรับข้อมูลเป็นแบบ real-time ได้ โดยจะเก็บไว้ใน hdfs ทั้งนี้ฟลูมยังมีการรองรับเหตุการณ์การเกิดความผิดพลาด (fault tolerant) ด้วยกลไก tunable reliability

อีกประการหนึ่ง ฟลูมมีการใช้รูปแบบข้อมูลที่สามารถพัฒนาต่อได้ ซึ่งสามารถนำไปพัฒนาต่อเป็นแอพพลิเคชั่นการวิเคราะห์ข้อมูลแบบออนไลน์

จากหัวข้อ 2.7.4 จะเห็นว่าชักวาและฟลูมจะมีจุดประสงค์โปรแกรมที่คล้ายกัน เพียงแต่หลักการทำงานภายในจะต่างกัน ซึ่งประสิทธิภาพของฟลูมนั้นดีกว่าทั้งในเรื่องของเวลา และความเสี่ยงการสูญหายของข้อมูลเมื่อระบบล่ม รวมไปถึงการนำไปประยุกต์ใช้ของชักวาก็ไม่ยืดหยุ่นเท่าฟลูม ซึ่งก็มีการมองแนวโน้มในอนาคตแล้วว่าฟลูมจะถูกนำไปใช้มากกว่าชักวาอย่างแน่นอน

2.7.6 HBase

รูปที่ 2.28 Apache HBase

อาปาเช่ เอชเบส (Apache HBase) เป็นฐานข้อมูล Open-source สำหรับตารางข้อมูลขนาดใหญ่ ซึ่งเป็นโมเดลที่ถูกพัฒนาหลังจาก Google's Bigtable (ระบบเก็บข้อมูลแบบกระจายแบบโครงสร้าง) ซึ่งตัว Bigtable จะจัดเก็บข้อมูลที่ถูกกระจายโดย Google File System HBase จึงเป็นการให้ Bigtable ทำงานอยู่บน HDFS ทำให้สามารถสุ่ม เข้าถึง อ่าน/เขียน ข้อมูลขนาดใหญ่ ได้อย่างรวดเร็ว

2.7.7 Zoo Keeper

รูปที่ 2.29 Apache Zoo Keeper

อาปาเช่ ซู คีปเปอร์ (Apache Zoo Keeper)เป็นศูนย์กลางการให้บริการทางด้านแอพพลิเคชั่นแบบกระจาย (distributed application)ที่มีการทำงานร่วมกันอย่างมีประสิทธิภาพสูง โดยสามารถปรับแต่งเบื้องต้นได้ เช่น Naming, Configuration management, Synchronization และ Group services

บริการเหล่านี้ถูกใช้กับบางแอพพลิเคชั่น ซึ่งในแต่ละแอพพลิเคชั่นที่ถูกสร้างและใช้งานนั้นจะมีการทำงานที่เกิดขึ้นมากมายและนำไปสู่การแก้ปัญหาของ Bugs รวมถึงการแย่งใช้งานทรัพยากร (Race conditions) ซึ่งยากต่อการจัดการ แม้ว่าจะดำเนินการถูกต้องเรียบร้อยแล้วก็ตามแต่ก็ถือว่าการจัดการเหล่านั้นก็ค่อนข้างมีความซับซ้อนมากอยู่ดี ซู คีปเปอร์จะคอยปรับแต่งระบบเหล่านั้นให้สะดวกแค่การจัดการมากยิ่งขึ้น เช่น การกำหนด Barrier ,Queue และ Locking

2.7.8 Mahout

รูปที่ 2.30 Apache Mahout

อาปาเช่ มาเฮาท์ (Apache Mahout) เป็นซอฟต์แวร์การใช้งานระบบแบบกระจายตัว หรือก็คืออัลกอริทึ่มสำหรับเครื่องที่สามารถปรับขนาดได้ โดยเน้นไปในด้านของการกรอง (filtering) การจัดกลุ่ม (clustering) และการจัดหมวดหมู่ (classification) โดยทั้งหมดนี้ดำเนินการตามภายใต้พื้นฐาน

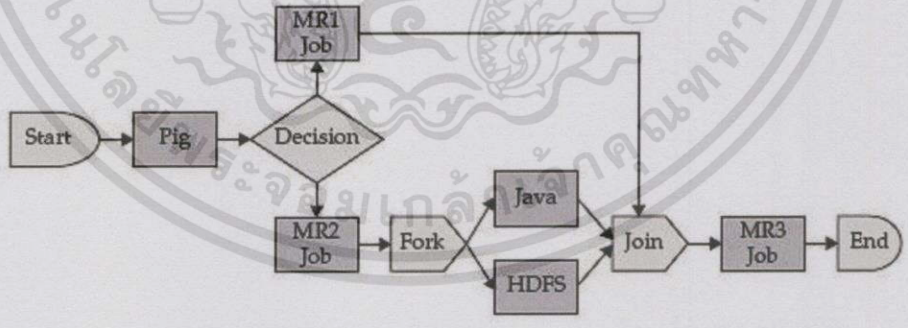
ของแมปรีดิวซ์ และยังสามารถให้บริการไลบารี่จาว่าที่สำคัญรวมถึงสำหรับการดำเนินงานทางคณิตศาสตร์ (เน้นไปทางพีชคณิตเส้นตรงและสถิติ)

2.7.9 Oozie

รูปที่ 2.31 Apache Oozie

อาปาเช่ อูซี่ เป็นซอฟต์แวร์ที่จะช่วยลดความยุ่งยากของเวิร์กโฟลว และข้องเกี่ยวกันระหว่างงาน ทั้งมีความสามารถในการกำหนดการกระทำและอ้างอิงการกระทำได้ จากนั้นก็จะกำหนดเวลากระทำงานเมื่อเงื่อนไขการอ้างอิงเป็นไปตามที่กำหนด

เวิร์กโฟลวในอูซี่จะถูกกำหนดในกราฟที่เรียกว่า Directed Acyclical Graph (DAG) ซึ่ง Acyclical แปลว่าไม่มีการวนลูปในกราฟ ซึ่งงานและจุดอ้างอิงทั้งหมดจะถูกดำเนินไปโดยไม่มีทางย้อนกลับมาทำซ้ำ ซึ่งโหนดในกราฟสามารถเป็นได้ทั้งงานแมปรีดิวซ์ พิกแอพพลิเคชั่น งานของไฟล์ระบบ หรือแม้กระทั้งจาว่า แอพพลิเคชั่น



รูปที่ 2.32 ตัวอย่าง Directed Acyclical Graph (DAG) ในอูซี่

2.7.10 Sqoop

อาปาเช่ สคูป (Apache Sqoop) เป็นเครื่องมือสำหรับการส่งข้อมูลขนาดใหญ่ระหว่างฮาดูปกับการจัดเก็บข้อมูลชนิดโครงสร้างอย่างมีประสิทธิภาพ สคูปสามารถนำเข้าข้อมูลจากตัวจัดเก็บข้อมูลชนิดโครงสร้างสู่ HDFS รวมถึงในทางตรงกันข้ามเช่นเดียวกัน

สคูปสามารถนำเข้าหรือส่งออกข้อมูลจากฐานข้อมูลชนิดโครงสร้างได้อย่างง่ายดาย ถ้าใช้สคูปนำข้อมูลภายนอกสู่ HDFS และนำไปไว้บนตารางด้วยไฮฟ์และเอชเบส หากรวมกับอูซี่ก็จะสามารถกำหนดช่วงเวลาส่งออกนำเข้าได้อย่างอัตโนมัติ สคูปใช้การเชื่อมต่อตามสถาปัตยกรรมที่รองรับปลั๊กอินที่สามารถทำการเชื่อมต่อกับระบบภายนอก

หลายคนอาจสงสัยเกี่ยวกับการทำงานภายใต้เบื้องหน้าที่ตรงไปตรงมาของสคูป ชุดข้อมูลที่ถูกส่งจะถูกหั่นไปยัง partition ที่ต่างกัน และงานแมปรีดิวซ์จะเริ่มทำงานโดยขึ้นอยู่กับ mapper ที่รับผิดชอบการถ่ายโอนข้อมูลชุดนั้นๆ บันทึกข้อมูลแต่ละชุดจะถูกจัดการขึ้นอยู่กับลักษณะความปลอดภัย เพราะสคูปใช้ข้อมูลของฐานข้อมูลในการสรุปชนิด

รูปที่ 2.33 แสดงแนวคิดการทำงานของสคูป

นอกจากที่กล่าวมาข้างต้น ยังมีระบบนิเวศน์ของฮาดูปอีกสองตัวที่ได้นำมาศึกษาและทดลองในครั้งรี้ นั่นก็คือ อาปาเช่ พิก (Apache Pig) และ อาปาเช่ ไฮฟ์ (Apache Hive) ซึ่งจะกล่าวในหัวข้อถัดไป

2.8 Apache Pig

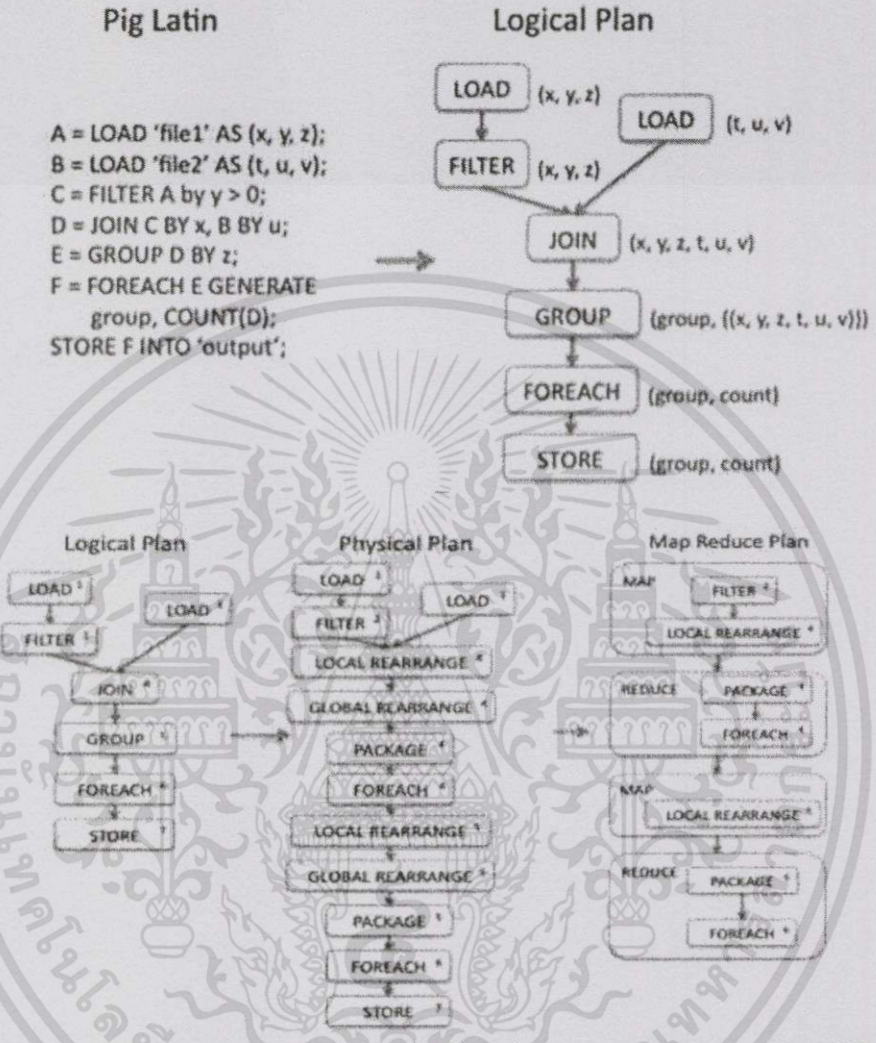
รูปที่ 2.34 Apache Pig

อาปาเช่ พิก (Apache Pig) เป็นแพลตฟอร์มสำหรับการวิเคราะห์เซ็ตข้อมูลขนาดใหญ่ ด้วยภาษาสคริปท์ขั้นสูง จุดเด่นอีกอย่างของ Pig คือโครงสร้างภายในที่ประมวลผลแบบขนาน ซึ่งทำให้สามารถรองรับข้อมูลขนาดใหญ่ได้ แน่นอนว่าโครงสร้างพื้นฐานดังกล่าวจะมีตัวคอมไพลเลอร์ที่คอยสร้างลำดับการแมปรีดิวซ์อยู่นั่นเอง

2.8.1 Pig Latin

พิก ลาติน (Pig Latin) คือภาษาสคริปท์ขั้นสูงที่ใช้ในแพลตฟอร์มของ Pig โดยมีลักษณะสำคัญสามประการคือ

1) เขียนโปรแกรมง่าย
2) คอมไพล์เลอร์มีการเพิ่มประสิทธิภาพ (Optimization) ให้โค้ดโดยอัตโนมัติ
3) สามารถเขียนฟังก์ชั่นพัฒนาด้วยตนเอง ทำให้โปรแกรมขยายออกไปได้



รูปที่ 2.35 แสดงการประมวลผลของคอมไพล์เลอร์จากภาษา Pig Latin สู่แมปรีดิวซ์

คำสั่งของ Pig latin จะนำความสัมพันธ์ระหว่างตัวแปรและโอเปอเรเตอร์ในคำสั่งมาใส่เป็นเอาท์พุท (ทุกคำสั่งของ pig latin จะเป็นเช่นนี้ยกเว้นคำสั่ง Load, Store และ write) ซึ่งในคำสั่งอาจจะมีการใช้ expression หรือ schema ก็ได้ ซึ่งคำสั่งสามารถเขียนเป็นแบบหลายบรรทัดเพื่อสะดวกในการอ่าน แต่ต้องลงท้ายด้วย semi colon (;) แสดงถึงการจบชุดคำสั่งนั้นๆ

ลำดับคำสั่งของ pig latin สามารถเรียงได้ดังนี้

1) LOAD : เพื่ออ่านข้อมูลจากระบบ
2) ชุดคำสั่งแปลง เพื่อดำเนินการทางข้อมูล
3) DUMP เพื่อแสดงข้อมูล หรือ STORE เพื่อเก็บผลข้อมูล

ไฟล์ที่ 2.1 แสดงไฟล์ student

```
John   18     4.0
Mary   19     3.8
Bill   20     3.9
Joe    18     3.8
```

ไฟล์ที่ 2.2 แสดงภาษา Pig latin ในการ Load ข้อมูลและดำเนินการทางข้อมูล

```
A = LOAD 'student' USING PigStorage() AS (name:chararray, age:int,
gpa:float);
B = FOREACH A GENERATE name;
DUMP B;
```

จากไฟล์ข้างต้น บรรทัดแรกจะเป็นการนำอินพุท student มาใส่ไว้ในโครงตารางที่มีคอลัมน์ name, age และ gpa โดยโครงตารางเหล่านี้จะถูกเก็บไว้ในตัวแปร A ทีละแถว จากนั้นจึงดำเนินการเลือกเพียงแต่คอลัมน์ name ของทุกแถวเก็บไว้ในตัวแปร B แล้วจึง DUMP แสดงค่าตัวแปร B ที่ได้ดังไฟล์ที่ 2.3

ไฟล์ที่ 2.3 แสดงผลลัพธ์ของการ DUMP B; ของไฟล์ 2.2

```
(John)
(Mary)
(Bill)
(Joe)
```

ดังตัวอย่างไฟล์ข้างต้น พื้นฐานของภาษา pig latin คือการใช้โอเปอเรเตอร์ทางความสัมพันธ์ตามลับดับเพื่อให้ได้ผลลัพธ์ที่ต้องการ ตารางที่ 2.X จะแสดงถึงโอเปเรเตอร์ที่พื้นฐานที่ถูกใช้บ่อยใน pig latin

ตารางที่ 2.1 ตารางแสดงโอเปอเรเตอร์พื้นฐานของ pig latin

Operator	ความหมาย	Syntax
Load	โหลดข้อมูลจากระบบไฟล์	LOAD 'data' [USING function] [AS schema];
Filter	เลือกทัปเปิ้ลจากเงื่อนไข	alias = FILTER alias BY expression;
Group	จัดกลุ่มข้อมูลตามความสัมพันธ์	alias = GROUP alias { ALL | BY expression} [, alias ALL | BY expression ...] [USING 'collected'] [PARALLEL n];
Foreach	สร้างการเปลี่ยนแปลงข้อมูลตามคอลัมน์ของข้อมูล	alias = FOREACH { gen_blk | nested_gen_blk } [AS schema];
Order	เรียงลำดับความสัมพันธ์จากฟิลด์	alias = ORDER alias BY { * [ASC|DESC] | field_alias [ASC|DESC] [, field_alias [ASC|DESC] ...] } [PARALLEL n];

2.9 Apache Hive

รูปที่ 2.36 Apache Hive

แอพพลิเคชั่นที่เป็นระบบนิเวศน์เชื่อมโยงของฮาดูปอีกหนึ่งตัวที่โปรเจคนำมาศึกษาคือ อาปาเช่ ไฮฟ์ เป็นซอฟต์แวร์อำนวยความสะดวกทางคลังข้อมูล (Data warehouse) ทั้งการดึงข้อมูล (query) หรือจัดการเซ็ตข้อมูลขนาดใหญ่ที่มีการเก็บแบบกระจายตัว (distributed storage) ไฮฟ์มีกลไกแสดงข้อมูลที่เก็บอยู่ให้อยู่ในรูปแบบข้อมูลโครงสร้าง และจัดการข้อมูลด้วยภาษาที่คล้าย SQL เรียกว่า HiveQL ซึ่งภาษานี้ยังอนุญาตให้โปรแกรมเมอร์สามารถปรับเปลี่ยนแก้ไขส่วนแมปรีดิวซ์เพื่อให้เหมาะสมกับโปรแกรมที่ต้องการมากที่สุดอีกด้วย

โดยพื้นฐานแล้ว ฮาดูปถูกสร้างขึ้นมาเพื่อจับระบบและจัดเก็บข้อมูลจำนวนมาก กลุ่มคอมพิวเตอร์ของฮาดูปเปรียบเสมือนบ่อเก็บข้อมูลหลากหลายชนิดโครงสร้าง ทั้งจากแหล่งที่แตกต่าง รวมไปถึงชนิดของข้อมูล ไฮฟ์จะให้ผู้ใช้เข้าไปสำรวจข้อมูลนั้นในรูปโครงสร้าง และวิเคราะห์ออกมาในเชิงฐานข้อมูล (Database)

ตารางข้อมูลในไฮฟ์ก็เหมือนกับตารางข้อมูลของฐานข้อมูลเชิงสัมพันธ์ (relational database) และจะถูกจัดระเบียบแบบอนุกรมวิธาน (taxonomy) จากขนาดใหญ่ไปสู่ขนาดย่อย ฐานข้อมูลจะประกอบไปด้วยตารางซึ่งถูกสร้างมาจากวิธีแบ่งส่วน (partitions) ซึ่งเราสามารถเข้าถึงข้อมูลด้วยภาษาที่ใช้ในการดึงข้อมูล (query) ง่ายๆเรียกว่า HiveQL โดยที่เราสามารถเขียนทับหรือเพิ่มข้อมูล แต่ไม่สามารถปรับปรุง (update) หรือลบได้

2.9.1 HiveQL เบื้องต้น

ดังที่กล่าวในข้างต้น โปรแกรมของ Hive จะสามารถแบ่งได้เป็นสองส่วนใหญ่ๆคือ ส่วนที่เป็นการสร้างตารางให้กับข้อมูล และอีกส่วนคือการดึงข้อมูลมาวิเคราะห์

2.9.1.1 Data Definition

- Creating Tables

ไฟล์ที่ 2.4 ตัวอย่างการสร้างตาราง page_view ด้วยภาษา HiveQL

```
CREATE TABLE page_view(viewTime INT, userid BIGINT,
                page_url STRING, referrer_url STRING,
                ip STRING COMMENT 'IP Address of the User')
COMMENT 'This is the page view table'
PARTITIONED BY(dt STRING, country STRING)
STORED AS SEQUENCEFILE;
```

ในตัวอย่างไฟล์ที่ 2.4 คอลัมน์ของตารางจะถูกกำหนดชนิดตามที่ได้เขียนไว้ ผู้ใช้สามารถเขียนคอมเมนต์ไว้ได้ทั้งในระดับคอลัมน์และระดับตาราง

- Loading Data

การโหลดข้อมูลเข้าไปในตารางไฮฟ์นั้นมีหลายวิธี ผู้ใช้สามารถสร้างตารางจากภายนอกฮาดูปซึ่งชี้มายังพื้นที่ใน HDFS ด้วยวิธีนี้ ผู้ใช้สามารถคัดลอกไฟล์มายังไดเรคทอรี่ที่กำหนดไว้ด้วย คำสั่ง put หรือ copy ของฮาดูป และสร้างตารางซึ่งชี้มายังไดเรคทอรี่ที่เกี่ยวกับข้อมูลเหล่านั้น เมื่อทำแบบนี้แล้ว ผู้ใช้จะสามารถแปลงข้อมูลและใส่เข้าไปยังตารางใดๆของไฮฟ์ก็ได้ ยกตัวอย่างเช่น ถ้าไฟล์ /tmp/pv_2008-06-08.txt มีการใช้จุลภาค (Comma) แยกหน้าข้อมูลของวันที่ 2008-06-08 ซึ่งต้องการจะโหลดเข้าสู่ตาราง page_view โดยต้องการให้มีการแบ่งส่วนที่เหมาะสม ทั้งหมดนี้จะสามารถเขียนได้ด้วยภาษา HiveQL ตามไฟล์ที่ 2.5

ไฟล์ที่ 2.5 แสดงตัวอย่างการใส่ข้อมูลเข้าไปในตาราง page_view

```
CREATE EXTERNAL TABLE page_view_stg(viewTime INT, userid BIGINT,
                page_url STRING, referrer_url STRING,
                ip STRING COMMENT 'IP Address of the User',
                country STRING COMMENT 'country of origination')
COMMENT 'This is the staging page view table'
ROW FORMAT DELIMITED FIELDS TERMINATED BY '44' LINES TERMINATED BY
'12'
STORED AS TEXTFILE
LOCATION '/user/data/staging/page_view';

hadoop dfs -put /tmp/pv_2008-06-08.txt /user/data/staging/page_view

FROM page_view_stg pvs
INSERT OVERWRITE TABLE page_view PARTITION(dt='2008-06-08',
country='US')
```

```
SELECT pvs.viewTime, pvs.userid, pvs.page_url, pvs.referrer_url,
null, null, pvs.ip
WHERE pvs.country = 'US';
```

2.9.1.2 Queries

ในส่วนวิเคราะห์ข้อมูลนั้นจะเป็นส่วนที่คล้ายกับภาษา SQL เป็นอย่างมาก แต่มีสิ่งหนึ่งที่พึงระลึกคือ HiveQL ต้องทำการเขียนผลลัพธ์ลงในตารางด้วยเสมอ

ไฟล์ที่ 2.6 แสดงการ query โดยเลือกแสดงข้อมูลที่ column active = 1

```
INSERT OVERWRITE TABLE user_active
SELECT user.*
FROM user
WHERE user.active = 1;
```

แน่นอนว่าจุดเด่นของไฮฟ์คือการวิเคราะห์ข้อมูลขนาดยักษ์ที่มีความสามารถหลายๆอย่างภาษา SQL เช่นการรวมข้อมูลดังไฟล์ที่ 2.7 หรือการจัดกลุ่มและนับในไฟล์ที่ 2.8

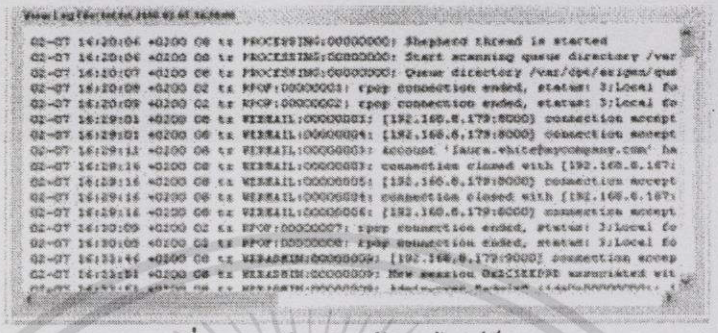
ไฟล์ที่ 2.7 แสดงการ query ตารางร่วมกันระหว่าง user และ page_view

```
INSERT OVERWRITE TABLE pv_users
SELECT pv.*, u.gender, u.age
FROM user u JOIN page_view pv ON (pv.userid = u.id)
WHERE pv.date = '2008-03-03';
```

ไฟล์ที่ 2.8 แสดงการจัดกลุ่มด้วย Group By และ การนับด้วย count

```
INSERT OVERWRITE TABLE pv_gender_sum
SELECT pv_users.gender, count (DISTINCT pv_users.userid)
FROM pv_users
GROUP BY pv_users.gender;
```

2.10 Logfile และการวิเคราะห์ Logfile



รูปที่ 2.37 ตัวอย่างข้อมูลไฟล์ล็อก

Log file คือ ข้อมูลจราจรคอมพิวเตอร์ เป็นข้อมูลเกี่ยวกับการติดต่อสื่อสารของระบบคอมพิวเตอร์ แสดงถึงแหล่งกำเนิด ต้นทาง ปลายทาง เส้นทาง เวลา วันที่ ปริมาณ ระยะเวลาชนิดของบริการ หรืออื่นๆ ที่เกี่ยวข้องกับการติดต่อสื่อสารของระบบคอมพิวเตอร์ ไฟล์ล็อกจะอยู่ในลักษณะข้อความจำนวนมาก ในหนึ่งบรรทัดคือหนึ่งเหตุการณ์ที่เกิดขึ้น ดังนั้น ไฟล์ล็อกหนึ่งไฟล์จะบันทึกเหตุการณ์จำนวนมากที่เกิดขึ้นบนระบบ

2.10.1 ประภทของไฟล์ล็อก

2.10.1.1 File Server log (syslog)

Server log เป็นไฟล์ล็อกที่ถูกสร้างขึ้นโดยอัตโนมัติและเก็บไว้ในเครื่อง Server ที่เกิดกิจกรรมนั้นๆขึ้น สามารถแยกได้เป็นหลายประเภทขึ้นอยู่กับประเภทกิจกรรมหรือหน้าที่ของเครื่อง Server เช่น Web Server log file จะเก็บประวัติการร้องขอหน้าเว็บ เป็นต้น
โดยปกติแล้วรูปแบบ(Format) มาตรฐานของ Web Server log file จะถูกกำหนดโดย W3C แต่ก็มีรูปแบบอื่นเช่นเดียวกัน กิจกรรมที่เพิ่งเกิดจะถูกเพิ่มไว้ที่ท้ายของไฟล์ล็อก ประกอบไปด้วย ชื่อลูกข่าย (Client), IP Address, วัน เวลาที่ร้องขอ, หน้าเว็บที่เรียก, โค้ด HTTP, ไบต์กำกับ, ตัวแทนลูกข่าย และตัวอ้างอิง ข้อมูลทั้งหมดนี้จะรวมกันอยู่ในไฟล์เดียว หรืออาจแยกไปอยู่ในล็อกเฉพาะก็ได้ อย่างไรก็ตาม Server log ส่วนใหญ่จะไม่เก็บข้อมูลเฉพาะของผู้ใช้

โดยปกติแล้ว ผู้ใช้ทั่วไปไม่มีความจำเป็นและไม่สามารถเข้าถึงไฟล์ล็อกได้ มีเพียง webmaster หรือผู้ดูแลระบบเท่านั้นที่สามารถเข้าถึงได้ การวิเคราะห์ล็อกเชิงสถิตินั้นอาจถูกใช้ในการตรวจสอบรูปแบบจราจรของเครือข่าย อาจดูจากช่วงของวันหรือสัปดาห์ ตัวอ้างอิง หรือตัวแทนผู้ใช้ ผู้ดูแลระบบที่เชี่ยวชาญจะใช้ไฟล์ล็อกเหล่านี้วิเคราะห์เพื่อปรับกลยุทธ์ทางการค้าทางเว็บไซต์ได้ เรียก

ได้ว่าองค์กรทางการค้าที่มีเว็บไซต์เป็นของตัวเองควรเข้าถึงความสามารถของโปรแกรมวิเคราะห์ไฟล์ล็อกได้ดีในระดับหนึ่ง

ดังที่กล่าวในข้างต้น ไฟล์ล็อกคือการบันทึกข้อมูลการติดต่อสื่อสารทั้งหมดที่เกิดขึ้น ซึ่งย่อมไม่ได้เกิดจากการใช้งานโปรแกรมหรือโปรโตคอลชนิดเดียว จึงเกิดการแบ่งประเภทไฟล์ล็อกตามโปรโตคอลที่ถูกใช้งาน เพื่อให้ง่ายต่อการค้นหามากยิ่งขึ้น ในที่นี้จะขอยกตัวอย่างถึงโปรโตคอลล็อกบางชนิดเท่านั้น

- Messages log

  จะเก็บข้อมูลหลากหลายแบบ ซึ่งไฟล์นี้สามารถใช้ในการหาการแก้ไขไฟล์ โดยมีข้อมูลของเหตุการณ์ และเวลา เพื่อช่วยตรวจสอบการบุกรุกได้

- Smb Log

  จะเก็บข้อมูลเหตุการณ์ที่เกิดขึ้นของโปรโตคอล Server Message Block Protocol สำหรับโปรแกรมประยุกต์ลูกข่ายในเครื่องคอมพิวเตอร์เพื่ออ่านและเขียนไฟล์และขอบริการจากโปรแกรมแม่ข่ายในเครือข่ายคอมพิวเตอร์ จึงสามารถเข้าถึงไฟล์ที่แม่ข่ายทางไกลและทรัพยากรอื่น เช่น เครื่องพิมพ์ เป็นต้น

- Secure log

  ในบางระบบ UNIX (RedHat Linux) ใช้เก็บ log ของ tcp wrapper messages ในไฟล์ secure log file ทุกครั้งที่มีการเชื่อมต่อกับระบบผ่าน inetd ที่ใช้ tcp wrappers, log messageจะเกิดขึ้น โดยสามารถตรวจสอบได้ว่ามีการสร้างการเชื่อมต่อจากบริการที่ผิดปรกติหรือไม่. หรือ ว่ามีการเชื่อมต่อบริการกับ host ที่ไม่ให้สิทธิหรือไม่

- Slapd Log

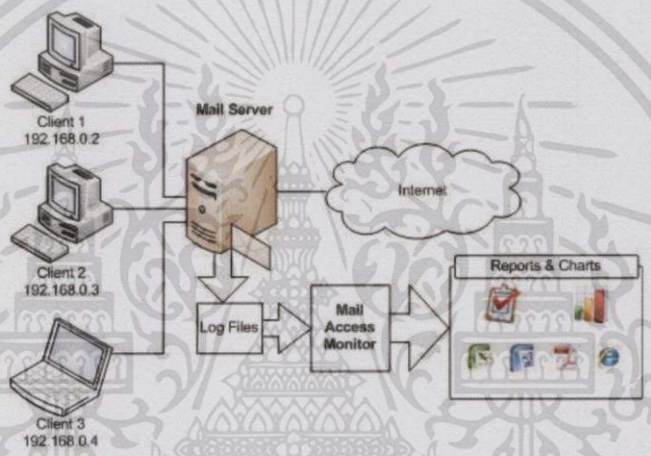
  เครื่อง Server แพลตฟอร์มยูนิกซ์ ที่ให้บริการโปรโตคอล Lapd ในการแจกจ่ายไดเรคทอรี่สำหรับเก็บข้อมูล คล้ายกับฐานข้อมูลแต่จะเป็นข้อมูลเชิงคุณลักษณะ (Attribute) เสียมากกว่า และจะถูกจัดเก็บแบบต้นไม้ลำดับขั้น (hierarchical tree) ผู้ใช้จะสามารถให้บริการแจกจ่ายไดเรคทอรี่สำหรับเก็บข้อมูล ทั้งเชื่อมต่อกับ ladp กลางหรือใช้ภายในองค์กรส่วนตัวก็ได้

2.10.1.2 Proxy log

ในเครือข่ายคอมพิวเตอร์ พร็อกซีเซิร์ฟเวอร์ (อังกฤษ: proxy server) คือเซิร์ฟเวอร์ ที่ทำงานโดยการเป็นตัวกลางในการหาข้อมูลตามคำขอของเครื่องลูกข่ายจากเซิร์ฟเวอร์อื่นๆ กล่าวคือเครื่องลูกข่ายเชื่อมต่อไปที่ พร็อกซีเซิร์ฟเวอร์เพื่อขอใช้งานบางบริการ เช่น ไฟล์ การเชื่อมต่อ เว็บเพจ หรือ

ทรัพยากรต่าง ๆ จากเซิร์ฟเวอร์อื่น จากนั้น พร็อกซีเซิร์ฟเวอร์จะทำการคัดกรองด้วยกฎที่ตั้ง ตัวอย่างเช่น คัดกรองจาก หมายเลขไอพี, Protocol หลังจากนั้นถ้าการขอผ่านการคัดกรอง พร็อกซีเซิร์ฟเวอร์จะจัดหาข้อมูลตามคำร้องขอจากเซิร์ฟเวอร์อื่นแทนเครื่องลูกข่าย จุดประสงค์หลักของพร็อกซีเซิร์ฟเวอร์คือ เพื่อให้เครื่องลูกข่ายซ่อนตัว (โดยส่วนใหญ่ เพื่อความปอดภัย) และ เพื่อความเร็วของการใช้บริการที่เพิ่มขึ้น, โดยการเก็บเว็บเพจจากเว็บเซิร์ฟเวอร์

2.10.1.3 Mail log



รูปที่ 2.38 แผนผังอีเมล์เซิร์ฟเวอร์

เมล์ล็อกจะบันทึกเหตุการณ์บริการรับส่งอีเมล์ของเครื่องเซิร์ฟเวอร์ เมื่อผู้ใช้เขียนอีเมล์โดยใช้โปรแกรมรับส่งอีเมล์ ลูกข่ายจะติดต่อกับเครื่องเซิร์ฟเวอร์ โดยใช้โปรโตคอล SMTP (Simple Mail Transport Protocol) อีเมล์จะถูกส่งมาเก็บยังเครื่องเมล์เซิร์ฟเวอร์ จัดเก็บลงในคิว เพื่อรอการจัดส่งต่อไป เมื่ออีเมล์ที่ผู้ใช้ส่งมายังเซิร์ฟเวอร์ จะถูกจัดการโดยโปรแกรมแลกเปลี่ยนอีเมล์ โดยอ่านที่อยู่อีเมล์ปลายทาง และนำโดเมนปลายทางไปตรวจสอบกับเนมเซิร์ฟเวอร์ (Name Server) เพื่อหาว่าเมล์เซิร์ฟเวอร์ใดเป็นเมล์เซิร์ฟเวอร์ที่ให้บริการโดเมนปลายทาง ผู้ใช้ปลายทางสามารถรับอีเมล์ได้โดย โดยโปรแกรมจะติดต่อมายังเครื่องเมล์เซิร์ฟเวอร์ แล้วส่งชื่อผู้ใช้และรหัสผ่าน

2.10.2 ไฟล์ล็อกกับโปรแกรมวิเคราะห์บนฮาดูป

ในไฟล์ล็อกหนึ่งประเภท จะมีรูปแบบ(Format) ที่ตายตัว หรืออาจถูกกำหนดเพิ่มเติมก็ได้ อย่างไรก็ตาม แม้ในตำแหน่งหนึ่งๆจะถูกกำหนดค่าเหมือนสดมภ์(Column) และแถว (Row) ดังรูปที่ 2.39 อาจจะดูคล้ายกับการเก็บข้อมูลแบบดาต้าเบส (Data Base) แต่ก็ไม่ได้ถูกเก็บไว้ในรูปแบบโครงกระดูก (Schema) จึงยังเรียกไฟล์ล็อกว่าเป็นข้อมูลชนิดไร้โครงสร้างได้อยู่

รูปที่ 2.39 ตัวอย่างไฟล์ล็อก สังเกตได้ว่าจะมีรูปแบบวันที่ขึ้นต้น ตามด้วยเวลา และข้อมูลอื่นๆ

โปรแกรมวิเคราะห์ไฟล์ล็อกส่วนใหญ่จะนำไฟล์ที่ได้ไปเปลี่ยนให้อยู่ในระบบฐานข้อมูลสัมพันธ์ (Relational Database) จึงค่อยนำไปประมวลผลด้วยภาษาดาต้าเบส เช่น MySQL เป็นต้น

แต่เมื่อมีการพูดถึงระบบประมวลผลฮาดูป ซึ่งเน้นใช้ประมวลกับข้อมูลไร้โครงสร้างแล้ว ความคิดที่จะนำไฟล์ล็อกมาวิเคราะห์โดยปราศจากกกระบวนการแปลงเป็นฐานข้อมูลก่อน ก็ดูน่าสนใจอยู่ไม่น้อย

ในส่วนของโปรแกรมวิเคราะห์ไฟล์ล็อกนั้นเป็น โปรแกรมสำหรับการนับเหตุการณ์ต่างๆ ที่เกิดขึ้นในไฟล์ล็อกโดยผู้ใช้งานสามารถการหาเหตุการณ์ต่างๆ ที่ผู้ใช้งานสนใจ ซึ่งผู้ใช้งานสามารถกำหนดขอบเขต ได้ในเรื่องของ คำ และขอบเขตช่วงเวลา เช่น

Messages log ผู้ใช้งานสามารถหาความถี่ของเหตุการณ์ที่ Error เกิดขึ้น ในช่วงเวลาที่สนใจได้ เพื่อตรวจสอบหาช่วงเวลาที่ระบบถูกบุกรุก หรือความผิดปกติต่างๆของระบบ และหาสาเหตุอื่นต่อไป

สำหรับ Mail log (Webmail) สามารถตรวจสอบหา ช่วงเวลาที่มีผู้พยายามบุกรุก อาจตรวจสอบได้จาก ความถี่ของเหตุการณ์ที่มีการ Login Error ถ้าพบว่า ถ้ามีการ Login Error มากกว่าปกติ ก็อาจจะเป็นผู้ต้องสงสัยที่มีการบุกรุก

หากมองในมุมของ Proxy log สามารถตรวจสอบหาว่า ผู้ใช้อินเตอร์เน็ตมีการร้องขอใช้งานหรือดาวน์โหลดข้อมูลจากเว็บไซต์ใดบ่อยมากที่สุดในช่วงเวลาที่ผู้ตรวจสอบกำหนดได้

บทที่ 3
การออกแบบและพัฒนา

3.1 บทนำ

สำหรับเนื้อหาในบทนี้ จะกล่าวถึงความต้องการของระบบ อธิบายถึงโครงสร้างของฮาดูปแบบโหนดเดียว (Single-node) และแบบหลายโหนด (Multi-node) รวมไปถึงการใช้งาน Hadoop Distribute File System (HDFS) และ แมปริดิวซ์ (Map Reduce) เฟรมเวิร์ค (Map Reduce Framework) ซึ่งจะเป็นการใช้งานเบื้องต้น เสนอวิธีการพัฒนาระบบ อีกทั้งยังนำเสนอวิธีการใช้งานของพิก (Pig Latin) ซึ่งสามารถทำงาน Map reduce ในเชิงเป็นไฟล์สคริปท์สำหรับการการวิเคราะห์ไฟล์ล็อก (Log file) ได้ และนำเสนอนำข้อมูลที่นำมาใช้ในระบบ เพื่อความเข้าใจเนื้อหาในลำดับต่อไป

3.2 ความต้องการของระบบ

เนื่องจากระบบฮาดูป นั้น มีการทำงานแบบคลัสเตอร์ร่วมกับโหนดต่างๆ เป็นจำนวนมาก จึงจำเป็นต้องมีซอฟต์แวร์, ฮาร์ดแวร์ และความต้องการต่างๆของระบบที่เหมือนกันในแต่ละโหนด ดังต่อไปนี้

3.2.1 แพลตฟอร์มที่รองรับ

แพลตฟอร์ม GNU/Linux นี้มีการสนับสนุนทั้งด้านการพัฒนาและในด้านผลิตผล อีกทั้งยังสามารถใช้งานบนระบบ Cloud Computing ได้ โดยผ่านการจัดการโอเพนแสตก (open stack management) โดยในที่นี้เลือกใช้ระบบปฏิบัติการ CENT OS 6.5 ในการทดสอบการพัฒนาระบบ ส่วนแพลตฟอร์ม Win32 นั้นสนับสนุนเพียงแค่ในด้านของการพัฒนา และการเข้าถึงระบบ เพราะการทดสอบระบบการประมวลผลแบบกระจายบน Win32 นั้นพบว่ามีการใช้งานได้ไม่มีประสิทธิภาพตามความต้องการ ดังนั้นแพลตฟอร์มนี้จึงไม่สนับสนุนในส่วนผลิตผล

3.2.2 ซอฟต์แวร์ที่ต้องการ

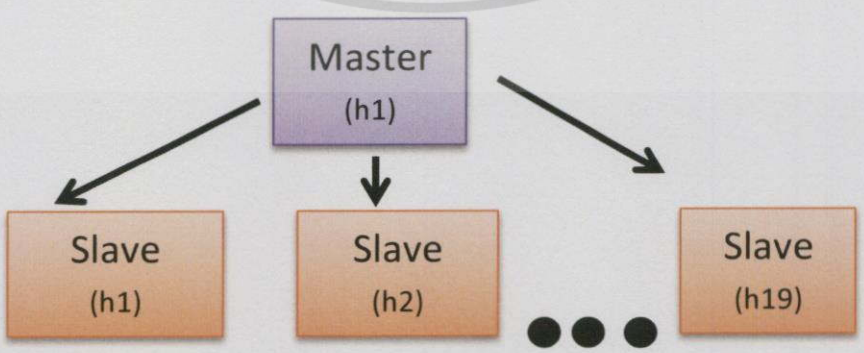
ซอฟต์แวร์ที่ต้องการสำหรับแพลตฟอร์มลินุกซ์ และวินโดว์ นั้นประกอบด้วย

1) Java™ รุ่น 1.6.x ขึ้นไป ในที่นี้เลือกใช้ Java 1.7.0 บนแพลตฟอร์มลินุกซ์
2) ssh และ sshd เพื่อให้สคริปท์ฮาดูปสามารถจัดการควบคุมฮาดูปโหนดอื่นๆได้
3) Putty บนแพลตฟอร์มวินโดว์ สามารถรีโมตผ่าน ssh ควบคุมเครื่องเซริฟเวอร์ได้

4) Eclipse บนแพลตฟอร์มลินุกซ์เป็นเครื่องมือ (Tool) สำหรับการเขียนโปรแกรม และทดลองการทำงานของโปรแกรมส่วนแมปรีดิวซ์ (Map Reduce)
5) VMWARE บนแพลตฟอร์มวินโดว์ สำหรับจำลองเครื่องเซิร์ฟเวอร์แพลตฟอร์มลินุกซ์บนแพลตฟอร์มวินโดว์
6) Apache Hadoop บนแพลตฟอร์มลินุกซ์ สำหรับในการติดตั้งเฟรมเวิร์คฮาดูป (Hadoop Framework)
7) Pig Latin บนแพลตฟอร์มลินุกซ์ และทำงานบนฮาดูป ซึ่งพิคละตินเป็นเครื่องมือสำหรับการสร้างงานของ Map reduce ในลักษณะสคริปท์ไฟล์ แทนภาษา JAVA เพื่อความสะดวก และรวดเร็วยิ่งขึ้น ในการวิเคราะห์ข้อมูลในระบบ ของฮาดูปเป็นการทำงานตามสคริปท์ที่ผู้ใช้กำหนด

3.3 โครงสร้างของระบบ

ก่อนที่เราจะสร้างคลัสเตอร์แบบหลายโหนด (multi-node cluster) นั้น เราจะต้องทำการสร้างเครื่องต้นแบบเป็นโหนดมาสเตอร์ ซึ่งสามารถสร้างโหนดเครื่องเซิร์ฟเวอร์สำรองจะต้องมีการติดตั้ง การตั้งค่า สำหรับทดสอบระบบฮาดูป ก่อนนำไปใช้งานจริง ให้เรียบร้อยเสียก่อนจากนั้นก็ทำการโคลนโหนดตามจำนวนที่ต้องการ เพื่อทดสอบการทำงานของระบบ จากนั้นจึงทำการสร้างระบบให้เป็นไปตามการออกแบบ ซึ่งจะนำเฟรมเวิร์คฮาดูปไปทำงานบนระบบประมวลผลกลุ่มก้อนเมฆ (Cloud computing) และในขั้นตอนถัดมา จะเป็นการรวมแต่ละคลัสเตอร์โหนด (single-node cluster) ทั้งหลายให้เป็นหนึ่งคลัสเตอร์แบบหลายโหนด (multi-node cluster) โดยจะมีหนึ่งโหนดทำหน้าที่เป็นโหนดมาสเตอร์ ซึ่งยังสามารถทำงานแบบสลาฟได้ในด้านการเก็บและประมวลผลข้อมูล และโหนดที่เหลือ ทำหน้าที่เป็นสลาฟเท่านั้น ซึ่งในการทดลองกำหนดให้มีโหนดมาเสตอร์ จำนวน 1 โหนด มีโหนดสลาฟจำนวน 15 โหนด (โหนดมาสเตอร์ สามารถทำงานเป็นโหนดสลาฟได้)



รูปที่ 3.1 โครงสร้างของระบบ

3.4 การตั้งค่าระบบ

3.4.1 การตั้งค่าพื้นฐานในไฟล์ hadoop-env.sh

ในไฟล์ hadoop-env.sh นี้จะเป็นไฟล์ที่เก็บค่าตัวแปรที่ฮาดูปจะเรียกใช้เพื่อให้โปรแกรมต่างๆของฮาดูปสามารถทำงานได้ ซึ่งในที่นี้ ให้แก้ไขพาธของตัวแปร JAVA_HOME ให้เป็นพาธของโปรแกรมจาวาที่เราติดตั้งลงไป เพื่อให้ฮาดูปสามารถเรียกใช้งานโปรแกรมจาวาได้ เนื่องจากฮาดูปนั้นทำงานบนแพลตฟอร์มของจาวา และพาธของไฟล์นี้จะอยู่ที่ /usr/local/hadoop/conf/hadoop-env.sh

ไฟล์ที่ 3.1 การตั้งค่าตัวแปรในไฟล์ hadoop-env.sh

```
export JAVA_HOME=/opt/jdk1.7.0_51
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

3.4.2 การตั้งค่าในไฟล์ conf/core-site.xml

ในส่วนนี้จะเป็นการกำหนดตำแหน่งของไดเรกทอรี่ที่ฮาดูปจะเก็บไฟล์ข้อมูลของมัน และพอร์ตทางเครือข่ายที่ต้องการใช้ ในส่วนของ HDFS เป็นต้น

ไฟล์ที่ 3.2 การตั้งค่าในไฟล์ conf/core-site.xml

```
<configuration>
<property>
      <name>fs.default.name</name>
      <value>hdfs://h1:9000/</value>
</property>
<property>
      <name>dfs.permissions</name>
      <value>false</value>
</property></configuration>
```

3.4.3 การตั้งค่าในไฟล์ conf/mapred-site.xml

ในส่วนนี้จะเป็นกำรกำหนดชื่อโฮส หรือไอพี และพอร์ตสำหรับจอบแทรกเกอร์ (JobTracker)

ไฟล์ที่ 3.3 การตั้งค่าในไฟล์ conf/mapred-site.xml

```
<configuration>
<property>
      <name>mapred.job.tracker</name>
      <value>h1:9001</value>
```

```
</property>
</configuration>
```

ทั้งนี้ชื่อโฮสต์สามารถกำหนดให้แต่ละเครื่องได้จาก /etc/hosts ให้แต่ละโหนดรู้จักกัน

ไฟล์ที่ 3.3 (ก) การตั้งค่าในไฟล์ /etc/hosts

```
10.0.0.5    h1
10.0.0.30   h2
10.0.0.6    h3
10.0.0.7    h4
10.0.0.8    h5
10.0.0.10   h6
10.0.0.11   h7
10.0.0.12   h8
10.0.0.13   h9
10.0.0.9    h10
10.0.0.14   h11
10.0.0.17   h12
10.0.0.16   h13
10.0.0.15   h14
10.0.0.19   h15
10.0.0.35   h16
10.0.0.33   h17
10.0.0.36   h18
10.0.0.39   h19
```

และสามารถกำหนดชื่อโฮสต์ให้กับเครื่องเซิร์ฟเวอร์ได้จาก /etc/hostname

ไฟล์ที่ 3.3 (ข) การตั้งค่าในไฟล์ /etc/sysconfig/network

```
HOSTNAME=h1
```

3.4.4 การตั้งค่าในไฟล์ conf/hdfs-site.xml

ในส่วนนี้จะเป็นการกำหนดจำนวนโหนดสูงสุดที่ใช้งานในระบบฮาดูปเพื่อตั้งค่าจำนวนไฟล์ที่ต้องการสำรองข้อมูล เช่น หากเป็นการทำงานแบบโหนดเดียว (Single-node) ก็อาจจะกำหนดค่าเป็น 1 แต่ในที่นี้มีทำงานแบบหลายโหนด (Multi-node) รวม 3 โหนด จึงกำหนดค่าโหนดทั้งหมดเป็น 3 ซึ่งหมายถึง มี 3 โหนดสามารถสำรองไฟล์ข้อมูลที่อยู่ใน 3 โหนดเหล่านั้นได้

ไฟล์ที่ 3.4 การตั้งค่าในไฟล์ conf/ hdfs-site.xml

```
<configuration>
<property>
      <name>dfs.data.dir</name>
      <value>/opt/datahadoop/dfs/name/data</value>
      <final>true</final>
```

```
</property>
<property>
      <name>dfs.name.dir</name>
      <value>/opt/datahadoop/dfs/name</value>
      <final>true</final>
</property>
<property>
      <name>dfs.replication</name>
      <value>3</value>
</property>
</configuration>
```

3.4.5 การตั้งค่าในไฟล์ conf/master

กำหนดเครื่องที่ต้องการให้เป็น Primary Namenode ในกรณีที่ต้องการทำระบบ Hadoop แบบ Multi-node โดยในที่นี้มีการกำหนดค่าไอพีแอดเดรสในไฟล์ /etc/Hosts ของระบบแล้ว จึงใช้ชื่อของเครื่องที่ต้องการได้เลย

ไฟล์ที่ 3.5 การตั้งค่าในไฟล์ conf/masters

```
h1
```

3.4.6 การตั้งค่าในไฟล์ conf/slaves

กำหนดเครื่องที่ต้องการให้เป็นเนมโหนดอันดับสอง (Secondary Namenode) โดยกำหนดบรรทัดละหนึ่งโฮสต์ และเนื่องจากเครื่อง Master ของฮาดูปสามารถทำงานเป็น Slave ได้ด้วย จึงทำการกำหนดเครื่องมาสเตอร์ node1 และเครื่องสลาฟที่เหลือทั้งหมดลงไปดังตัวอย่าง

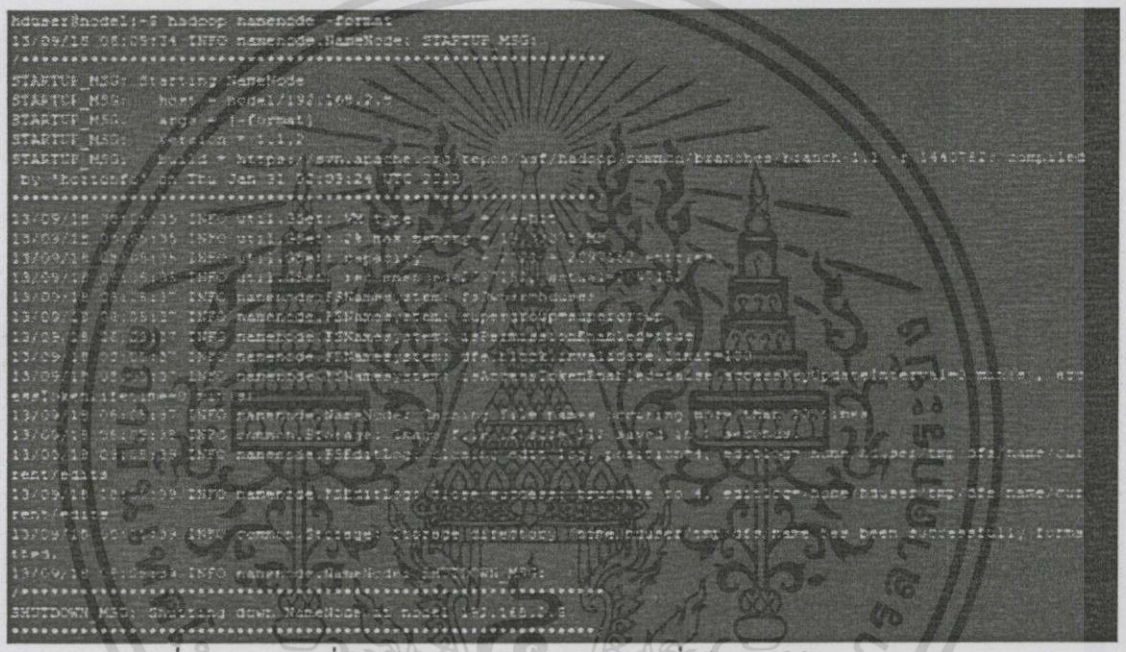
ไฟล์ที่ 3.6 การตั้งค่าในไฟล์ conf/ slaves

```
h1
h3
h4
h5
h6
h7
h8
h9
h10
h11
h12
h13
h14
h15
h16
h17
h18
h19
```

3.5 การใช้งานระบบ

3.5.1 การเริ่มต้นฮาดูป

การเริ่มต้นเปิดระบบฮาดูปคลัสเตอร์นั้นจะต้องเปิดเดมอนของ HDFS และMap/Reduce cluster ทั้งคู่เสียก่อน โดยก่อนอื่นนั้นเราจะต้องทำการฟอร์แมตระบบ HDFS เพื่อให้ระบบสร้างไฟล์ข้อมูลที่จำเป็นสำหรับการเก็บข้อมูลเสียก่อน โดยใช้คำสั่ง hadoop namenode –format



รูปที่ 3.2 ผลการสั่งฟอร์แมตเนมโหนดก่อนการเริ่มต้นการใช้งานฮาดูป

จากนั้นให้เริ่มต้นเปิด Namemode , Datanode , Jobtracker และ Tasktracker เพื่อเริ่มการทำงานของฮาดูปด้วยคำสั่ง start-all.sh

รูปที่ 3.3 ผลการสั่งเริ่มต้นการใช้งาน Hadoop

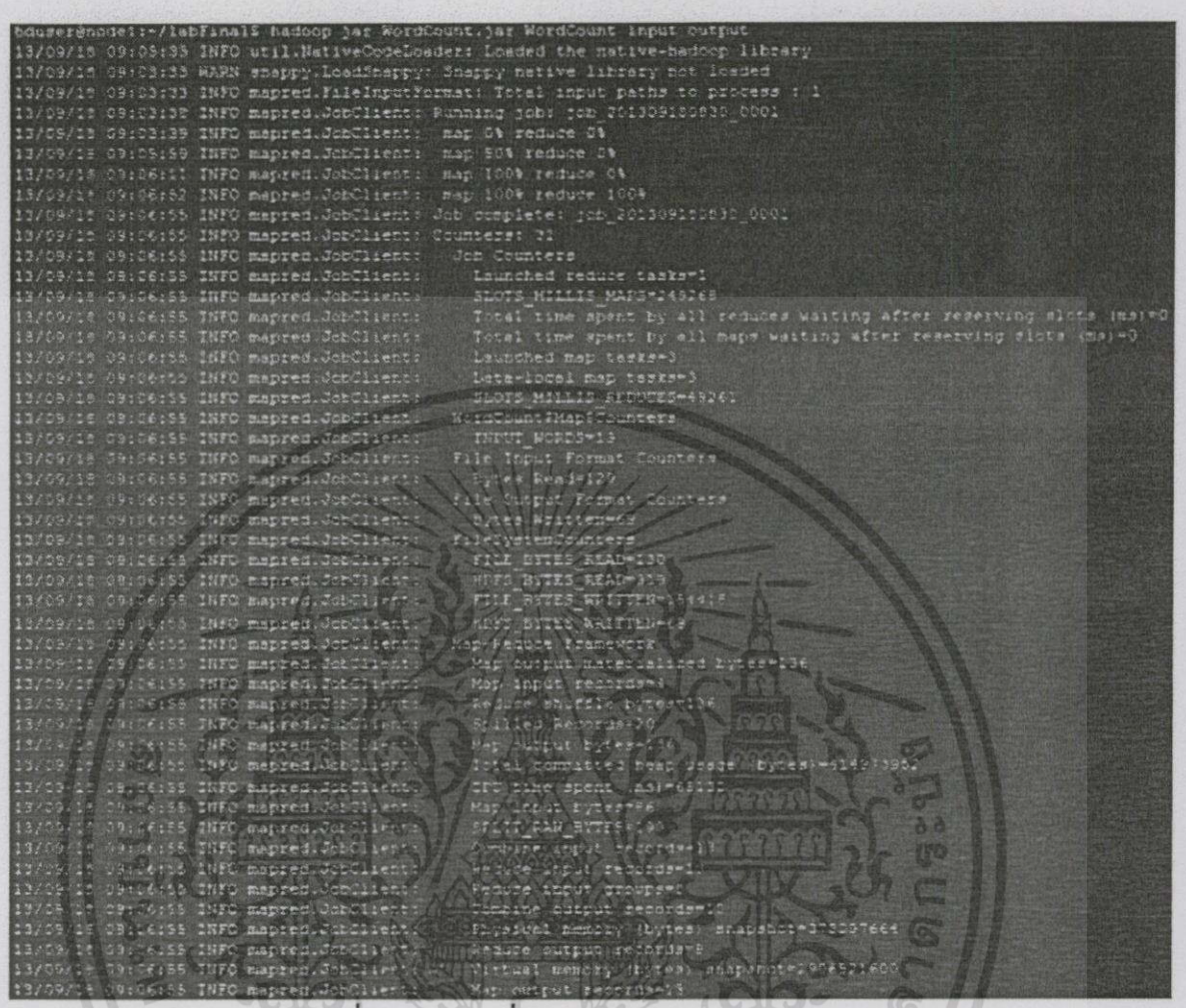
3.5.2 การใช้งาน Hadoop

ก่อนที่จะทดสอบการใช้งาน MapReuce เราจะต้องนำไฟล์ข้อมูลเข้า HDFS ก่อนโดยป้อนคำสั่ง hadoop fs –mkdir input เพื่อสร้างไดเรคทอรี่ input จากป้อนคำสั่ง hadoop dfs -put b.txt input เพื่อทำข้อมูลไฟล์ b.txt ลงใน HDFS ในพาธ input สามารถตรวจสอบว่าพาธที่เราต้องการอยู่ในระบบแล้วหรือไม่ สามารถป้อนคำสั่งได้ดังนี้ hadoop fs –lsr ซึ่งเป็นคำสั่งตรวจสอบว่าตอนนี้มีพาธ หรือไฟล์ข้อมูลใดอยู่ใน HDFS อยู่บ้าง

```
hduser@node1:~/labFinal$ hadoop dfs -lsr
drwxr-xr-x   - hduser supergroup          0 2013-09-18 08:54 /user/hduser/input
-rw-r--r--   3 hduser supergroup         86 2013-09-18 08:54 /user/hduser/input/b.txt
```

รูปที่ 3.4 ผลการป้อนคำสั่งตรวจสอบพาธใน HDFS

สำหรับการใช้งานโปรแกรม WordCount สามารถป้อนคำสั่งในรูปแบบดังนี้ hadoop jar <ชื่อไฟล์โปรแกรม> <Class program> <Inputpath> <Outputpath> ยกตัวอย่างการป้อนคำสั่งดังต่อไปนี้ hadoop jar WordCount.jar WordCount input output



รูปที่ 3.5 ผลการสั่งการทำงาน WordCount

และเรายังสามารถดูการทำงานผ่านยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ได้อีกด้วย ดังนี้

- http://node1:50070/ คือ ยูสเซอร์อินเตอร์เฟสของ NameNode
- http://node1:50030/ คือ ยูสเซอร์อินเตอร์เฟสของ JobTracker
- http://node1:50060/ คือ ยูสเซอร์อินเตอร์เฟสของ TaskTracker

NameNode 'node1:54310'

Started: Wed Sep 18 08:29:17 PDT 2013
Version: 1.1.2, r1440782
Compiled: Thu Jan 31 02:03:24 UTC 2013 by hortonfo
Upgrades: There are no upgrades in progress.

Browse the filesystem
Namenode Logs

Cluster Summary

21 files and directories, 11 blocks = 32 total. Heap Size is 30.08 MB / 966.69 MB (3%)

Configured Capacity	:	56.1 GB
DFS Used	:	384 KB
Non DFS Used	:	9.72 GB
DFS Remaining	:	46.39 GB
DFS Used%	:	0 %
DFS Remaining%	:	82.68 %
Live Nodes	:	3
Dead Nodes	:	0
Decommissioning Nodes	:	0
Number of Under-Replicated Blocks	:	0

NameNode Storage:

Storage Directory	Type	State
/home/hduser/tmp/dfs/name	IMAGE_AND_EDITS	Active

รูปที่ 3.6 ยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ของ Namenode Daemon

node1 Hadoop Map/Reduce Administration

Quick Links

State: RUNNING
Started: Wed Sep 18 08:30:43 PDT 2013
Version: 1.1.2, r1440782
Compiled: Thu Jan 31 02:03:24 UTC 2013 by hortonfo
Identifier: 201309180830
SafeMode: OFF

Cluster Summary (Heap Size is 9 MB/966.69 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	1	2	0	0	0	0	6	6	4.00	0	0	0

Scheduling Information

Queue Name	State	Scheduling Information
default	running	N/A

Filter (Jobid, Priority, User, Name)

Example: 'user:smith 3200' will filter by 'user:smith' in the user field and '3200' in all fields

Running Jobs

none

Completed Jobs

Jobid	Started	Priority	User	Name	Map % Complete	Map Total	Maps Completed	Reduce % Complete	Reduce Total	Reduces Completed	Job Scheduling Information	Diagnostic Info
job_201309180830_0001	Wed Sep 18 09:03:36 PDT 2013	NORMAL	hduser	wordcount	100.00%	2	2	100.00%	1	1	NA	NA

Retired Jobs

none

Local Logs

Log directory, Job Tracker History

รูปที่ 3.7 ยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ของ JobTracker Daemon

tracker_node1:localhost/127.0.0.1:56269 Task Tracker Status

hadoop

Version: 1.1.2, r1440782
Compiled: Thu Jan 31 02:03:24 UTC 2013 by hortonfo

Running tasks

Task Attempts	Status	Progress	Errors

Non-Running Tasks

Task Attempts	Status

Tasks from Running Jobs

Task Attempts	Status	Progress	Errors

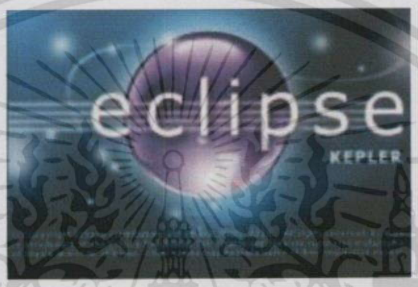
Local Logs

Log directory

รูปที่ 3.8 ยูสเซอร์อินเตอร์เฟสทางเว็บไซต์ของ TaskTracker Daemon

3.6 การพัฒนาระบบด้วย Eclipse

เนื่องจากโปรแกรมที่เขียนขึ้นมาสำหรับใช้งานในระบบนั้น จะต้องสร้างให้ไฟล์ JAVA เป็นไฟล์ JAR เสียก่อน ซึ่งการใช้งานโปรแกรม Eclipse (Kepler) บนแพลตฟอร์มลินุกซ์ซึ่งเป็นเครื่องมือ (Tool) สำหรับการเขียนโปรแกรม และทดลองการทำงานของโปรแกรมส่วน Map/Reduce ในภาษา JAVAสามารถตรวจสอบได้ว่า คลาสแมปริดิวซ์ (Map Reduce) ที่เขียนให้กับระบบนั้นจะสามารถใช้งานได้หรือไม่ก่อนนำไปประมวลผลจริง



รูปที่ 3.9 โปรแกรม Eclipse (Kepler) สามารถทำงานก่อนสร้างไฟล์ JAR

3.7 การพัฒนาระบบด้วย Pig Latin

เนื่องจากผู้จัดทำได้มีการเขียนโปรแกรมวิเคราะห์ log file ซึ่งจะมีการจัดกลุ่มข้อมูลต่างๆ เป็นรูปแบบสำเร็จรูปพอที่จะให้ผู้ใช้งานสามารถค้นหา และสามารถตีความเหตุการณ์ต่างๆ ได้อย่างสะดวกยิ่งขึ้นกับ Log นั้นๆ นอกจากผู้จัดทำศึกษาหลักการของจาวาแมปรีดิวซ์ (Java Map Reduce) แล้ว ยังได้ศึกษาไปถึงแอพพลิเคชั่น และเครื่องมือต่างๆที่ช่วยในการจัดการกับข้อมูลได้ดียิ่งขึ้นที่สามารถใช้งานบนเฟรมเวิร์คฮาดูปได้ นั่นก็คือ Pig Lantin

Pig Latin เป็นเครื่องมือสำหรับการวิเคราะห์ข้อมูล โดยลักษณะของการทำงานของ Pig Latin จะทำงานผ่านไฟล์สคริปท์ ซึ่งเบื้องหลังจะทำการแปลงเป็นภาษาJAVA เพื่อเข้าสู่ส่วนประมวลผลแมปรีดิวซ์ (Map Reduce) ดังนั้นผู้จัดทำจึงได้พัฒนาระบบด้วย Pig Latin ควบคู่กับการพัฒนาระบบด้วยภาษา JAVA ที่ให้ทำงานในลักษณะ Map Reduce ด้วยการพัฒนาระบบวิเคราะห์ Log file เพื่อเปรียบเทียบว่า กระบวนการ ลักษณะการทำงาน การประมวลผล และผลลัพธ์ที่ได้ ของทั้งสองแตกต่างกันอย่างไร และเหมาะสมนำไปใช้กับอะไร

3.8 การพัฒนาระบบด้วย Hive

นอกจากผู้จัดทำได้มีการเขียนโปรแกรมวิเคราะห์ log file ด้วย จาวาแมปรีดิวซ์ (Java Map Reduce) และ Pig Latin แล้ว ผู้จัดทำได้ศึกษาเครื่องมือต่างๆที่ช่วยในการจัดการกับข้อมูลได้ดียิ่งขึ้น ที่สามารถใช้งานบนเฟรมเวิร์คฮาดูปได้ นั่นก็คือ Hive

Hive เป็นเครื่องมือสำหรับวิเคราะห์ข้อมูล เหมาะสำหรับการทำ Data warehouse ซึ่งลักษณะของคำสั่งมีลักษณะเหมือนภาษา SQL ดังนั้นผู้จัดทำจึงได้พัฒนาระบบด้วย Hive และ Pig Latin ควบคู่กับการพัฒนาระบบด้วยภาษา JAVA ที่ให้ทำงานในลักษณะ Map Reduce ด้วยการพัฒนาระบบวิเคราะห์ Log file เพื่อเปรียบเทียบว่า กระบวนการ ลักษณะการทำงาน การประมวลผล และผลลัพธ์ที่ได้ ของทั้งสองแตกต่างกันอย่างไร และเหมาะสมนำไปใช้กับอะไร

บทที่ 4

การทดลองและผลการทดลอง

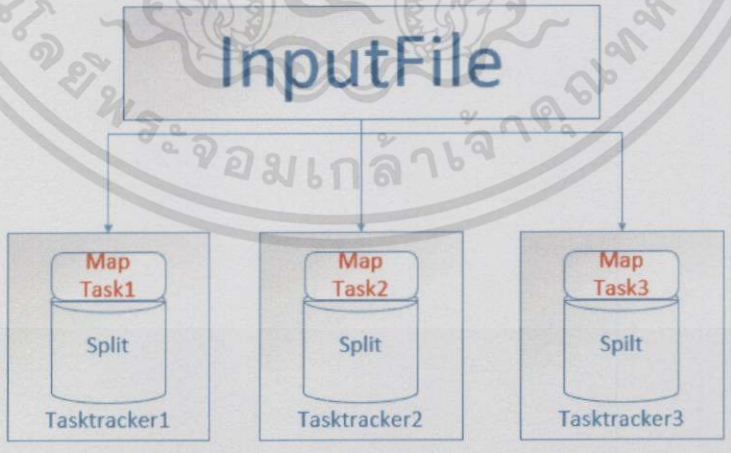
4.1 การทดลองนับคำทั้งหมดที่ปรากฏ (WordCount)

การทดลองนี้จะเป็นการประมวลผลไฟล์โดยการนับคำ เพื่ออ่านไฟล์ข้อความและนับจำนวนคำทั้งหมดที่ปรากฏอยู่ในไฟล์ โดยอาศัยหลักการของแมปริดิวซ์ (Map Reduce) เป็นองค์ประกอบของการทดลองนี้ ประกอบด้วยโปรแกรมภาษาจาวา ที่เรียกใช้งานแมปริดิวซ์ (Map Reduce) และไฟล์ที่ต้องการจะนับคำทั้งหมดเป็นจำนวนที่ต้องการ

4.1.1 ขั้นตอนการทำงานของโปรแกรมนับคำ (Word Count)

การทำงานของโปรแกรมนับคำนั้น มีการแบ่งออกเป็น 2 ส่วนหลัก คือส่วนแมป (Map) และส่วนรีดิวซ์ (Reduce)

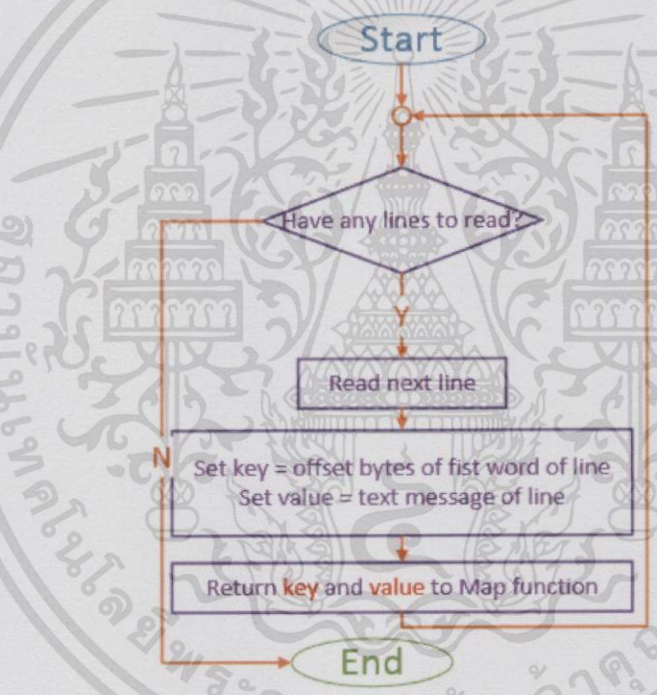
ในส่วนของการแมป (Map)นั้นจะเริ่มจากส่วนการทำงานของอินพุตไฟล์ (Input file) ซึ่งเป็นส่วนรับฟอร์แมตของไฟล์ข้อมูลที่ถูกแบ่งออกมาให้กับโหนดสลาฟในระบบ โดยผ่านเข้าอินเตอร์เฟสอินพุตสปริท (Input split) แล้วสร้างเป็นบล็อกข้อมูลที่ เรียกว่า สปริท (Split) ซึ่งการนำเข้าข้อมูลของโปรแกรมนับคำจะอยู่ในรูปแบบ TextInputFormat



รูปที่ 4.1 แสดงการแบ่งกระจายการทำงานของอินพุตไฟล์ (Input file) ให้กับแต่ละสปริท (Split) ในโปรแกรมนับคำ (Word Count)

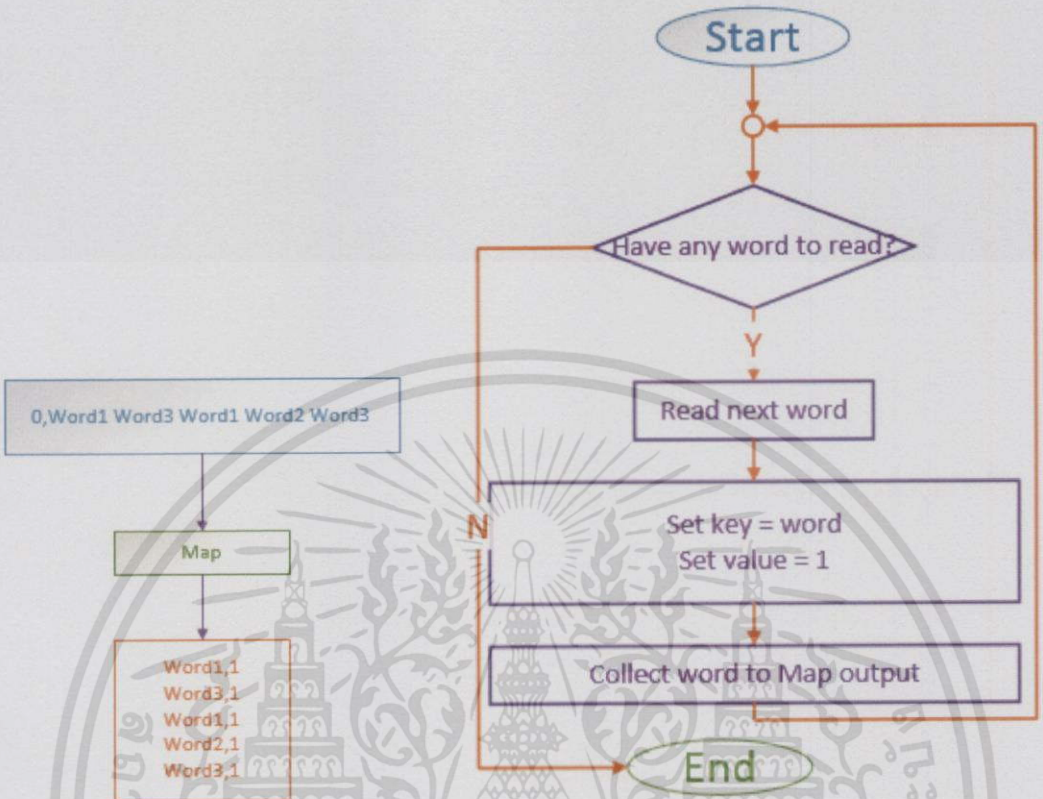
จากนั้นระบบจะนำบล็อกข้อมูลสปริท (Split) นั้นไปเข้าอินเตอร์เฟส Record reader ซึ่งทำหน้าที่นำข้อมูลที่เข้ามาแล้วสร้างค่าคีย์ Key และ Value ให้กับสปริท (Split) เหล่านั้น แล้วส่งค่า Key และ Value ส่งไปยังฟังก์ชันแมป (Map)

สำหรับอินพุตไฟล์ (Input file) ฟอร์แมต TextInputFormat จะเข้าสู่อินเตอร์เฟสของ LineRecordReader ซึ่งอินเตอร์เฟส LineRecordReader จะพิจารณาสปริท (Split) ที่ได้รับโดยการอ่านค่าในสปริท (Split) นั้นทุกๆ 1 บรรทัด แล้วส่งค่า Key และ Value ให้กับฟังก์ชันแมป (Map) ซึ่งค่า Key ที่ถูกส่งให้กับฟังก์ชันแมป (Map) คือ Offset byte ซึ่งเป็นหมายเลขที่อยู่ของคำแรกที่เริ่มต้นของบรรทัดนั้น และ Value คือข้อความที่เป็นสตริงของบรรทัดนั้น



รูปที่ 4.2 แสดงการทำงานของ LineRecordReader ให้กับแต่ละสปริท (Split) ในโปรแกรม WordCount

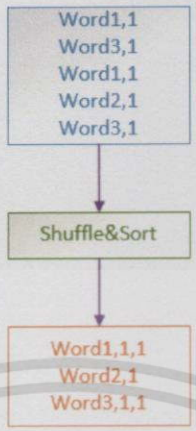
เมื่อฟังก์ชันแมป (Map) ได้รับค่า Key และ Value จาก RecordReader ฟังก์ชันแมป (Map) จะทำการตรวจสอบทีละคำในบรรทัดเดียวกัน โดยพิจารณาจากช่องว่างในบรรทัด (White space) เป็นลักษณะของ Token และทำการให้ค่าน้ำหนักของแต่ละคำเป็น 1 แล้วทำการเอาท์พุตแต่ละคำออกไปในลักษณะ Key และ Value ซึ่ง Key มีค่าเท่ากับคำ และ Value มีค่าเท่ากับค่าน้ำหนักของแต่ละคำเป็น 1 ซึ่งมีลักษณะเป็น <Word, 1> ซึ่งเอาท์พุตที่ออกจากฟังก์ชันแมป (Map) เรียกว่า Intermediate data



รูปที่ 4.3 แสดงการทำงานของฟังก์ชันแมป (Map) ในโปรแกรม WordCount

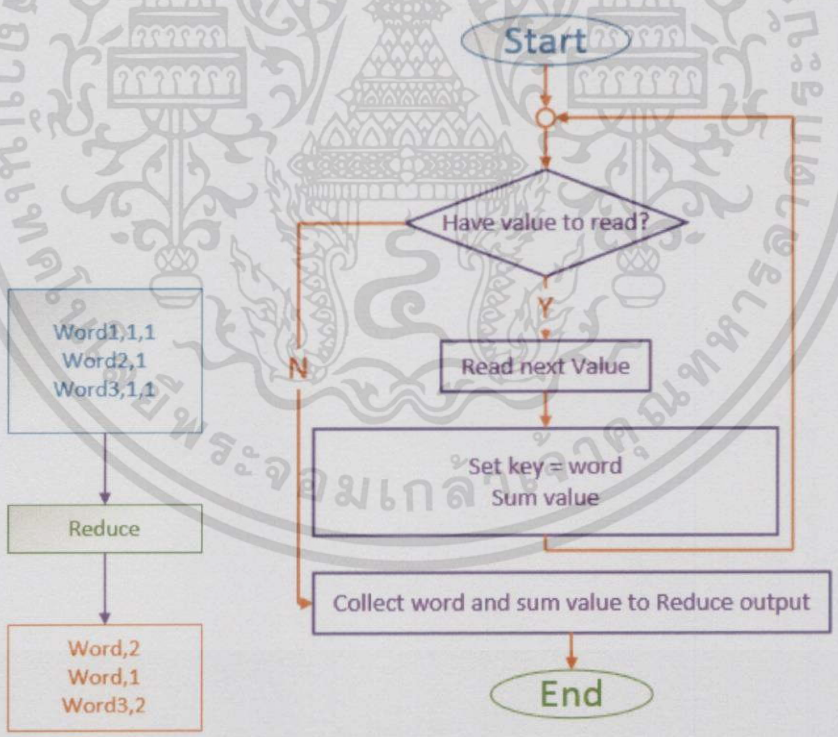
เมื่อฟังก์ชันแมป (Map) ส่งเอาท์พุตออกมา Combiner ก็จะทำการรวมค่า Value ของ Key ที่เหมือนกันก่อน แล้วจึงส่งเอาท์พุตออกไปเรียก Intermediate data เช่นกัน

เมื่อระบบได้รับ Intermediate Data มาจาก Combiner ก็จะทำการรวมทำข้อมูลเหล่านั้นออกเป็นพาทิชันโดยจะทำการจัดกลุ่มข้อมูลที่มี Key เป็นคำเดียวกันให้อยู่ในกลุ่มเดียวกัน จากนั้นจึงนำแต่ละพาทิชันเหล่านั้นมาเรียงตามพจนานุกรมแล้วรวมเป็นพาทิชันเดียวกัน ซึ่งเรียกวิธีการนี้ว่า “Shuffle&Sort”



รูปที่ 4.4 แสดงการทำงานของฟังก์ชัน Suffle&Sort ในโปรแกรม WordCount

ในส่วนของรีดิวซ์ (Reduce) จะรับ Intermediate Data มาจากเอาท์พุตของShuffle & Sort ก็จะนำค่า Value ที่มี Key เป็นคำเดียวกันมารวมค่ากัน จะได้จำนวนของคำแต่ละคำ



รูปที่ 4.5 แสดงการทำงานฟังก์ชันรีดิวซ์ (Reduce) ในโปรแกรม WordCount

จากนั้นรีดิวซ์ (Reduce) จะส่งให้ยัง HDFS เขียนไฟล์เอาท์พุตต่อไปโดย Text Record Reader จะเป็นตัวที่กำหนดฟอร์แมตไฟล์เอาท์พุตได้

4.1.2 การทดลองนับคำทั้งหมดในไฟล์

ทำการทดลองโดยสร้างไฟล์จำนวน 3 ไฟล์ ขนาด 74 ไบต์ 74 ไบต์ และ 34 ไบต์ ตามลำดับ โดยไฟล์ file1.txt และ file2.txt มีจำนวน 3 บรรทัด และ file3.txt มีจำนวน 2 บรรทัด ทั้งสามไฟล์มีจำนวนคำรวมทั้งหมด 22 โดยนับคำซ้ำ ถ้านับคำไม่รวมคำที่ซ้ำกันจะเหลือเพียง 9 คำ

ไฟล์ที่ 4.1 ไฟล์ขนาดเล็กไฟล์ที่หนึ่งที่ใช้ในการทดลอง

```
Apple Orange Pineapple
Mango Pineapple GRAPE
Greenapple Greenapple Mango
```

ไฟล์ที่ 4.2 ไฟล์ขนาดเล็กไฟล์ที่สองใช้ในการทดลอง

```
Orange grape apple
Mango greenapple Pineapple
GreenaPPle Rambutan Potato
```

ไฟล์ที่ 4.3 ไฟล์ขนาดเล็กไฟล์ที่สามใช้ในการทดลอง

```
Pineapple M@ngo Potato
Greenapple
```

ไฟล์ที่ 4.4 ผลลัพธ์ของการทดลองโปรแกรมนับคำ

```
apple 2
grape 2
greenapple 5
m@ngo 1
mango 3
orange 2
pineapple 4
potato 2
rambutan 1
```

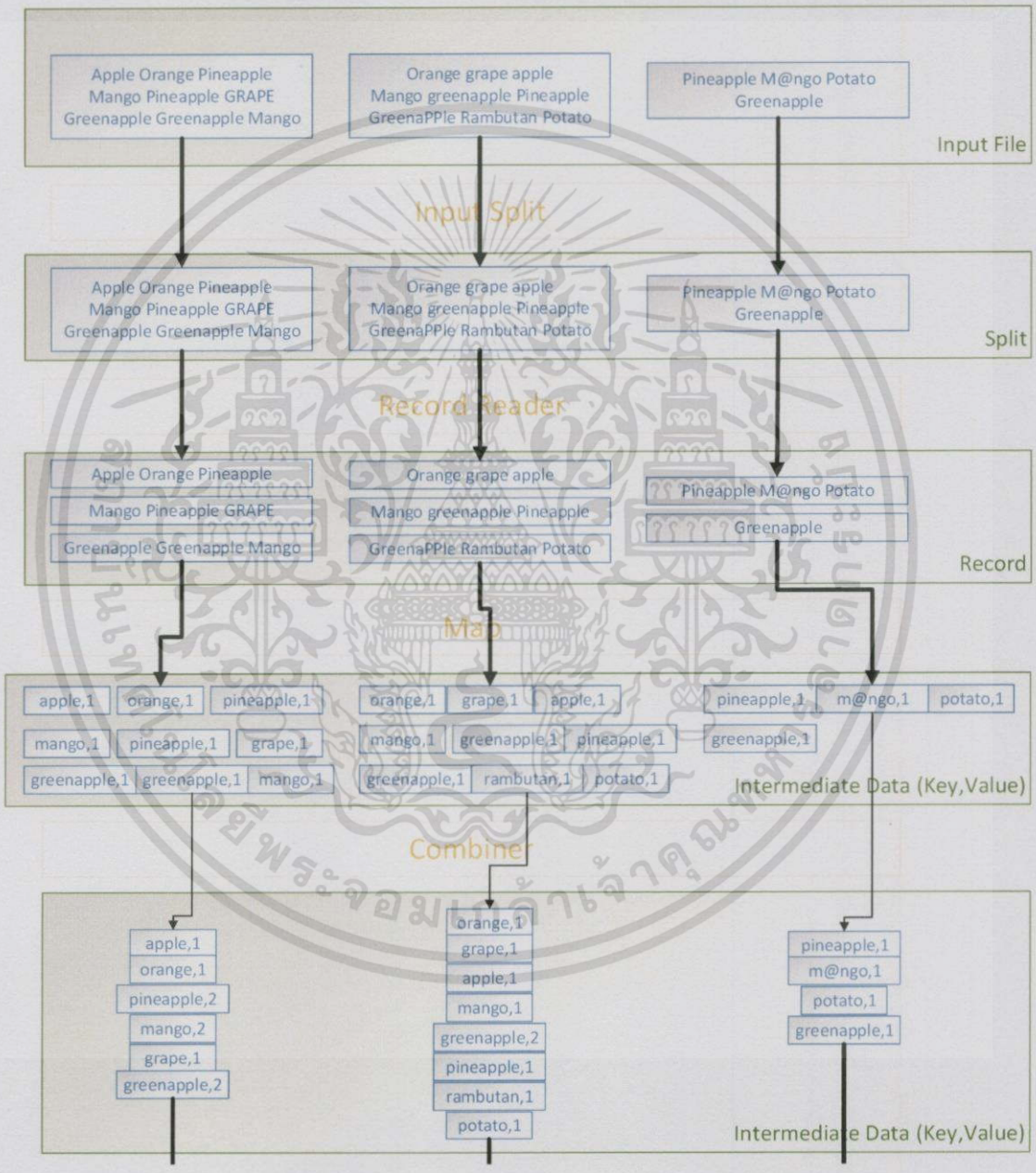
จากไฟล์ที่ 4.4 แสดงผลลัพธ์จากการทดลองใช้งานโปรแกรมนับคำ พบว่าถ้าเจอคำซ้ำกันจะมีการนับคำคำนั้นเพิ่มขึ้นหนึ่ง โดยที่ไม่สนใจว่าตัวอักษรนั้นจะเป็นพิมพ์เล็ก หรือตัวอักษรพิมพ์ใหญ่ ตัวอักษรที่อ่านเข้าไปจะถูกแปลงเป็นตัวอักษรพิมพ์เล็กทั้งหมด

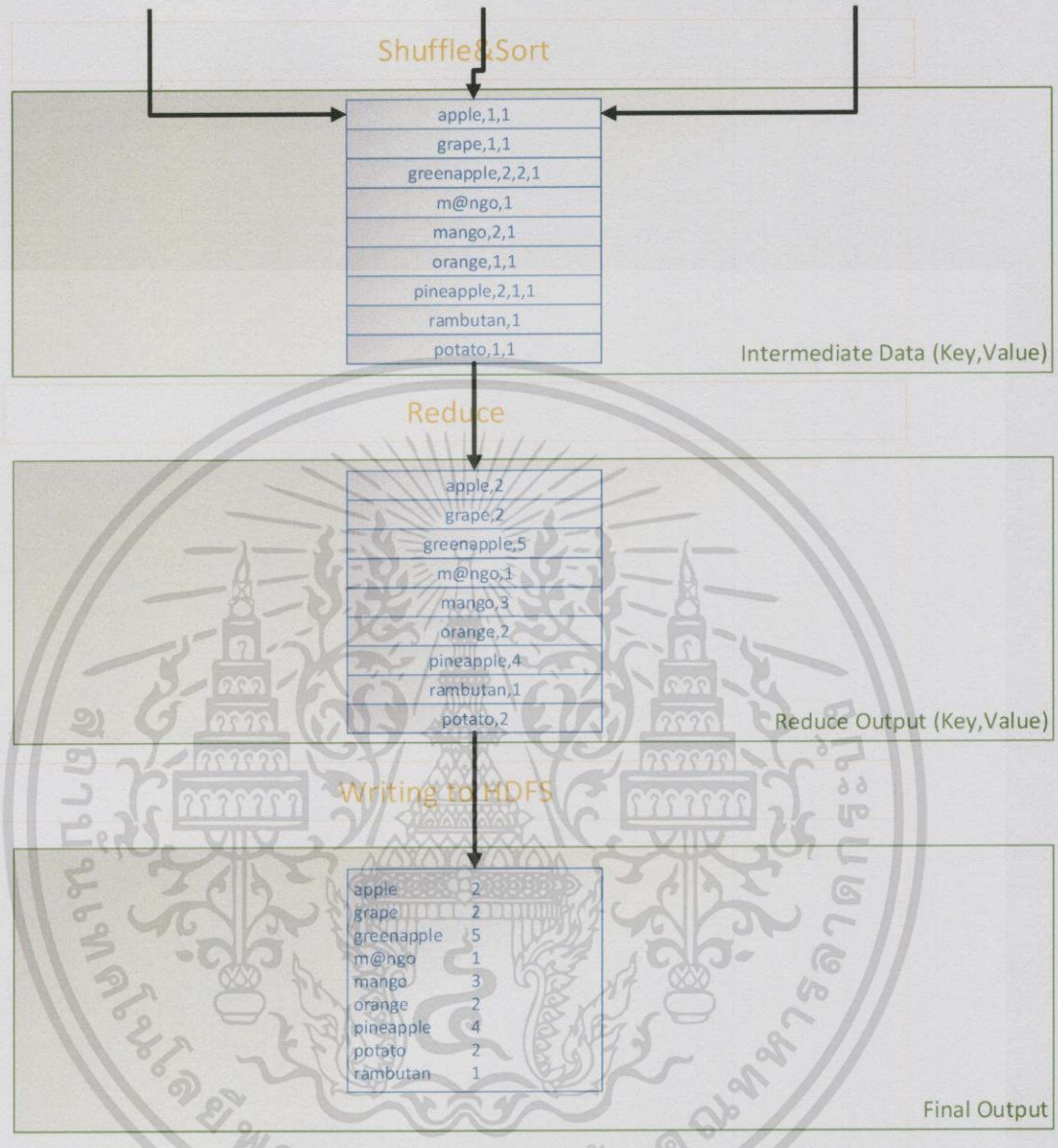
ตารางที่ 4.1 ผลการทดลองการทำงานของโปรแกรมนับคำ (WordCount)

	Counter	Map	Reduce	Total
Job Counters	Launched reduce tasks	0	0	1
	Launched map tasks	0	0	3
	Data-local map tasks	0	0	3
FileSystemCounters	FILE_BYTES_READ	0	256	256
	HDFS_BYTES_READ	479	0	479
	FILE_BYTES_WRITTEN	154,531	51,566	206,097
	HDFS_BYTES_WRITTEN	0	86	86
Map-Reduce	Reduce input groups	0	9	9
	Combine output records	22	0	22
	Map input records	8	0	8
	Reduce shuffle bytes	0	268	268
	Reduce output records	0	9	9
	Spilled Records	18	18	36
	Map output bytes	268	0	268
	Map output records	22	0	22
	Combine input records	18	0	18
	Reduce input records	0	18	18

จากตารางที่ 4.1 พบว่ามีงานที่ใช้ในการทำงานแมป (Map) จำนวน 3 งานตามจำนวนไฟล์จาก Launched map tasks ส่วน Map input records มีค่าเท่ากับ 8 ซึ่งเป็นส่วนนับ Record ที่เข้ามาในแมป (Map) โดย 1 Record จะอ่านอินพุตสปริท (Input Split) เข้ามา 1 บรรทัด ดังนั้นในระบบอ่านข้อมูลเข้ามาในระบบทั้งหมดรวม 8 บรรทัด จากนั้น Map output records มีค่าเท่ากับ 22 ซึ่งหมายความว่า เมื่อแมป (Map) อ่านค่าทั้ง 8 บรรทัดนั้นแล้วพบคำทั้งหมด 22 คำซึ่งเป็นขั้นตอนที่ยังไม่มีการนับคำซ้ำกัน จากนั้นพบว่า Combine input records มีค่าเท่ากับ 22 ซึ่งเป็นการเป็นการรวมค่าของคำสำหรับคำเดียวกันที่อยู่ในงานแมป (Map) เดียวกันก่อนส่งไปยังส่วนรีดิวซ์ (Reduce) ซึ่งเสมือนการทำรีดิวซ์ (Reduce) แต่ทำเพียงในแมป (Map) เดียวกัน จึงได้ Combine

output records มีค่าเท่ากับ 18 จากนั้นพบว่า Reduce input records มีค่าเท่ากับ 18 แสดงว่ารีดิวซ์ (Reduce) ได้รับอินพุตมาจากเอาท์พุตของ Combiner และพบว่า Reduce output records มีค่าเท่ากับ 9 ซึ่งเป็นการรวมคำทั้งหมดที่มีอยู่โดยไม่คิดคำซ้ำซึ่งแสดงการทำงานดังรูปที่ 4.6





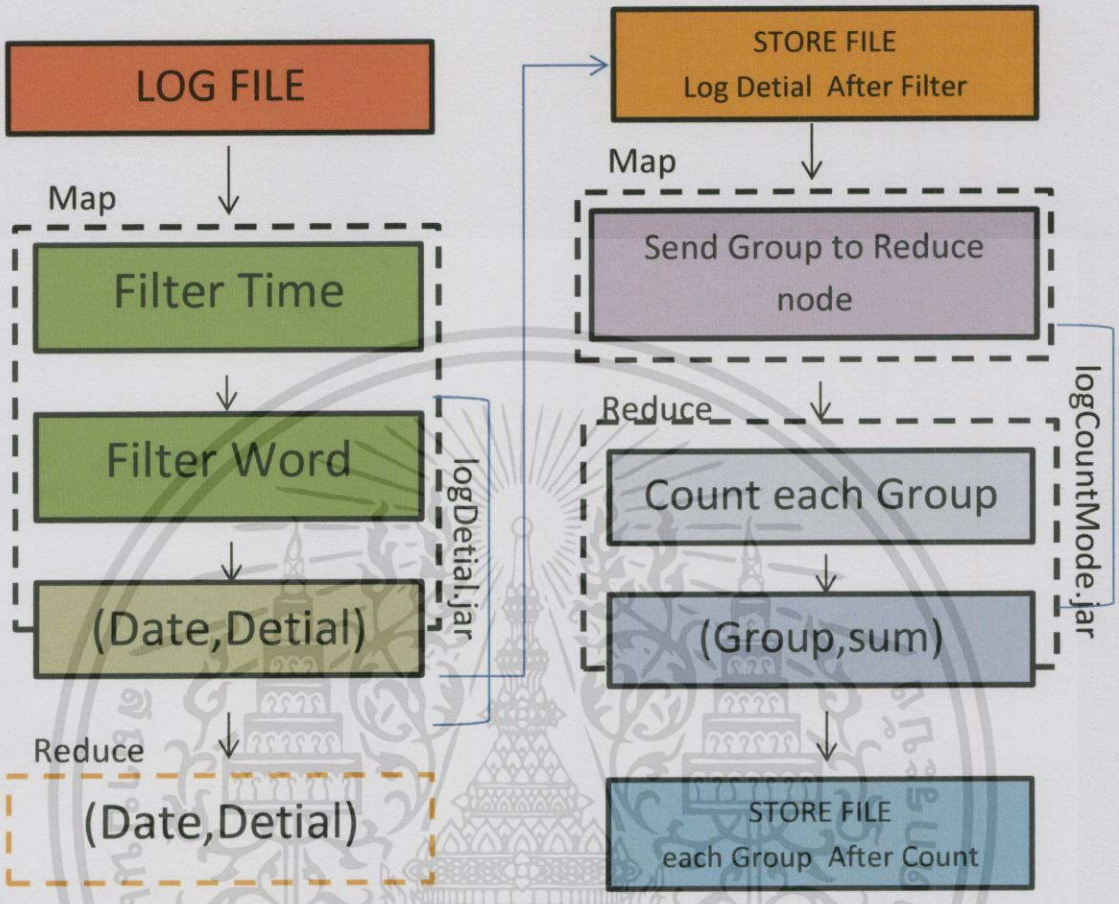
รูปที่ 4.6 การทำงานของโปรแกรมนับคำ (WordCount)

4.2 การทดลองโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file

การนำเอาข้อมูล Log file มาประมวลผลโดยมีขั้นตอนหลักๆ อยู่ 3 ขั้นตอนคือ ค้นหาคำ , ฟิลเตอร์ด้วยช่วงเวลา และจัดกลุ่มให้กับข้อมูล ซึ่งการจัดกลุ่มข้อมูลเป็นการเอาช่วงของข้อมูลที่สามารถจัดกลุ่มได้ เช่น เวลา แล้วนับความถี่ของเหตุการณ์ที่เกิดขึ้นให้ตรงตามเงื่อนไขการค้นหา จากนั้นจะทำการเปรียบเทียบหาความแตกต่างของผลที่ได้ระหว่างโปรแกรม ที่ถูกสร้างจากภาษาจาวาที่ทำงานในส่วนของแมปรีดิวซ์ (Map Reduce) และ Pig Latin

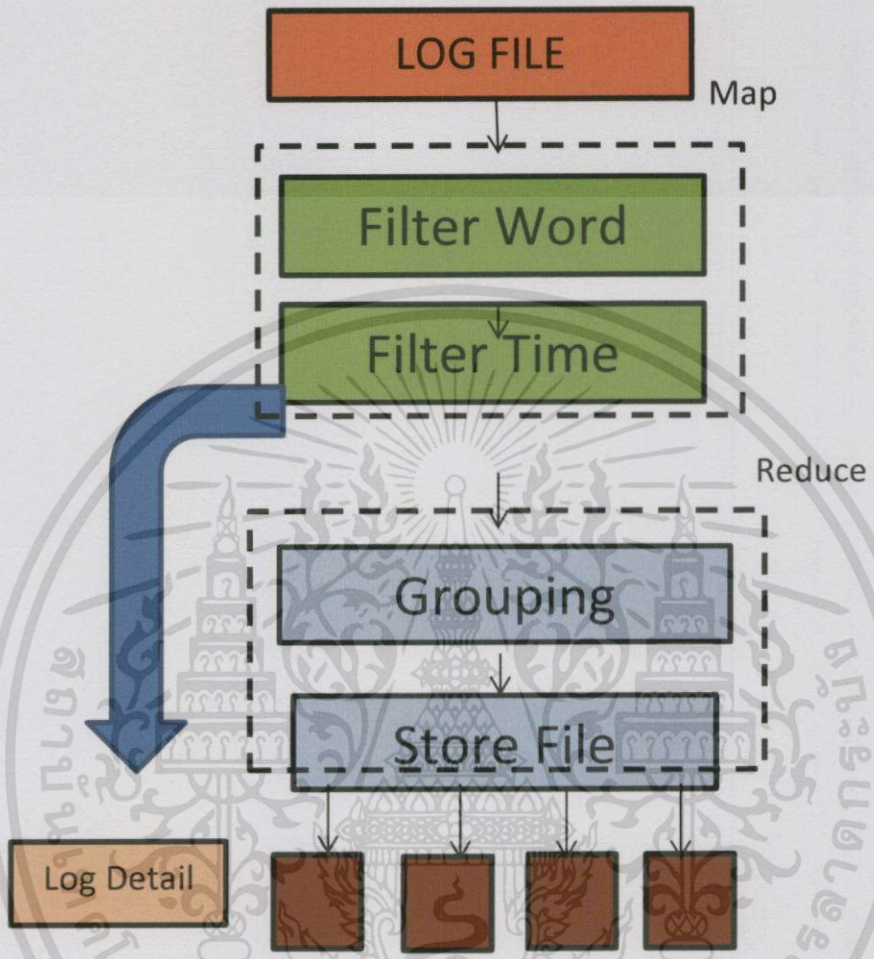
4.2.1 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file

การทำงานของโปรแกรม ค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file นั้นมีการแบ่งออกเป็น 3 ขั้นตอนหลักคือ มีการค้นหาคำ เพื่อต้องการค้นข้อมูลที่มีรายละเอียดได้ชัดเจนยิ่งขึ้น หลังจากนั้นก็จะทำการฟิลเตอร์กรองช่วงเวลาที่ผู้ใช้งานต้องเพื่อที่จะได้ตรวจสอบข้อมูลใน Log file ในช่วงเวลาที่เกิดเหตุการณ์ต่างๆ ได้รวดเร็วยิ่งขึ้น และขั้นตอนการจัดกลุ่มให้กับข้อมูล และนับความถี่ให้กับเหตุการณ์ที่เกิดขึ้น กับแต่ละกลุ่มนั้นๆ เช่น การนับจำนวน login error สำหรับผู้ที่เข้ามาใช้บริการอีเมลล์บนเว็บแอพลิเคชัน (Web Application) ทำให้สามารถตรวจหาผู้บุกรุก หรือ ทราบช่วงเวลาที่มีผู้ไม่หวังดีพยายามจะเข้ามาใช้ระบบ ก็จะสามารถหาวิธีแก้ไขได้ทันท่วงทีทั้งนี้ผู้จัดทำได้ทำการพัฒนาโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file ทั้งภาษาจาวา (JAVA) และพิคละติน (Pig Latin) โดยมีความคาดหวังที่จะให้ผลลัพธ์หลังจากทีประมวลผลได้ให้ใกล้เคียงกันมากที่สุด แต่ทั้งนี้ขึ้นอยู่กับฟังก์ชันการใช้งานต่างๆ ของเครื่องมือเหล่านั้นด้วย ซึ่งเป็นลักษณะพิเศษของเครื่องมือเหล่านั้น ดังนั้นผู้จัดทำจึงจะทำการทดสอบกับ Log แบบต่างๆ ซึ่งความแตกต่างการทำงานของโปรแกรม ระหว่างจาวา (JAVA) และพิคละติน (Pig Latin) ที่ผู้จัดทำได้ทำออกแบบ มีขั้นตอนตอนการทำงานดังนี้



รูปที่ 4.7 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้ กับ Log file ภาษาจาวา ด้วยหลักการแมปรีดิวซ์ (Map Reduce)

สำหรับขั้นตอนการทำงานของโปรแกรม ของภาษาจาวา (JAVA) ด้วย Map Reduce จะแบ่งขั้นตอนหลักๆ เป็น 2 ส่วน ได้แก่ ส่วนที่เป็นการฟิลเตอร์ และส่วนที่เป็นการนับความถี่สำคัญของเหตุการณ์ที่เกิดขึ้นที่ได้จากการแบ่งกลุ่มของข้อมูล ซึ่งถูกแบ่งออกเป็น 2 ไฟล์สำหรับการรันโปรแกรม ในส่วนแรกจะเริ่มด้วยในส่วนของแมป (Map) ฟังก์ชัน มีการทำฟิลเตอร์กรองคำที่ต้องการค้นหา และฟิลเตอร์ช่วงเวลา แล้วจากนั้นก็จะทำการส่งคู่ Key และ Value ไปยัง รีดิวซ์ (Reduce) ฟังก์ชัน เพื่อทำการจับคู่เวลากับ รายละเอียดของเหตุการณ์ที่สนใจ ของ Log file อีกส่วนหนึ่งนั้นในส่วนของแมป (Map) ฟังก์ชันจะจัดกลุ่มของประเภทข้อมูล log โดยการกำหนดให้กลุ่มเป็น ค่า key แล้วส่งคู่ key และ value ซึ่งมีค่าเป็นหนึ่ง ไปยังรีดิวซ์ (Reduce) ฟังก์ชัน ซึ่งจะเป็นส่วนที่ทำการรวมค่าของแต่ละ key ซึ่งหมายถึง การรวมค่าของแต่ละกลุ่มข้อมูลเช่นกัน ดังนั้นในขั้นตอนทำงานนี้จะทำให้จำเป็นต้องสร้างงานขึ้นมาทำงานระบบถึงสองงานด้วยกัน



รูปที่ 4.8 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file ด้วย Pig Lantin

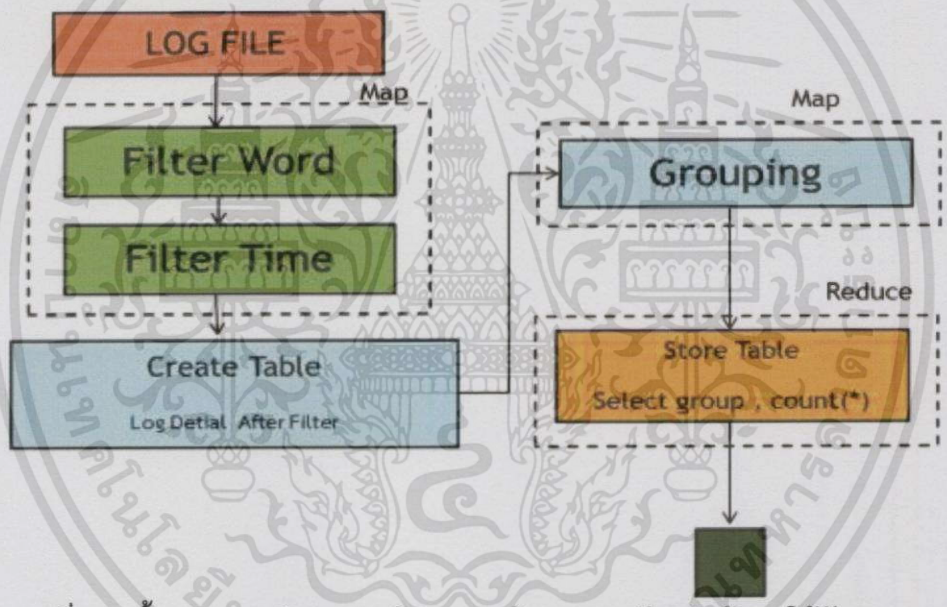
สำหรับขั้นตอนการทำงานของโปรแกรม ของ Pig Latin จะมีเพียงการทำงานเพียงครั้งเดียว ซึ่งประกอบไปด้วยขั้นตอนของการแมป (Map) ฟังก์ชัน มีการทำฟิลเตอร์กรองคำที่ต้องการค้นหา และฟิลเตอร์ช่วงเวลา แล้วจากนั้นก็จะทำการส่งคู่ Key ซึ่งก็คือชื่อกลุ่มข้อมูลที่จะใช้จัดกลุ่ม และ Value ซึ่งมีค่าเป็นหนึ่ง สำหรับไว้นับความถี่ของที่เกิดขึ้นของแต่ละกลุ่ม ไปยัง รีดิวซ์ (Reduce) เพื่อทำการจัดการรวมความถี่ของแต่ละกลุ่ม ทั้งนี้ Pig Latin สามารถทำงานได้ทั้ง interactive mode ซึ่งเป็นโหมดที่สามารถป้อนคำสั่งให้ทำงานได้โดยตรง หรือ ทำงานเป็นสคริปท์ไฟล์ได้ ทำให้สามารถสร้างไฟล์ที่ได้ผลลัพธ์จากการค้นหาได้หลากหลาย พร้อมๆกันได้ ซึ่งลักษณะภาษา Pig Latin กับ JAVA จะแตกต่างโดยสิ้นเชิง ภาษา JAVA เป็นภาษาโปรแกรมคอมพิวเตอร์ ซึ่งจะมีการเรียนใช้แบบ ออปเจ็ค

โอเรียนเต็ด (Object oriented) ซึ่ง Pig Latin จะเป็นในลักษณะที่เป็นการกำหนดตัวแปรซ้อนชั้นไปเรื่อยๆ โดยอาตัวแปรเหล่านั้นไปใส่ไว้ในฟังก์ชันในการจัดการข้อมูลที่มีมาให้สำเร็จรูป เช่น

ไฟล์ที่ 4.5 ตัวอย่างคำสั่งสคริปท์ Pig Latin

```
A = load 'passwd' using PigStorage(':');
B = foreach A generate $0 as id;
dump B;
```

จากไฟล์ที่ 4.5 เป็นตัวอย่างในการเรียกข้อมูลมาแสดง ซึ่งสามารถเรียกข้อมูลเป็น Attribute ได้เลย สำหรับข้อมูลที่ค่อนไปทางกึ่งโครงสร้าง (Semi-structure data)



รูปที่ 4.9 ขั้นตอนการทำงานของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file ด้วย Hive

สำหรับขั้นตอนการทำงานของโปรแกรม ของ Hive จะมีการทำงานถึง 2 ส่วน ในส่วนแรกประกอบไปด้วยขั้นตอนของการแมป (Map) ฟังก์ชัน มีการทำฟิลเตอร์กรองคำที่ต้องการค้นหา และฟิลเตอร์ช่วงเวลา แล้วจากนั้นก็จะทำการสร้างตารางข้อมูลขึ้นมาใหม่ แล้วบันทึก Log ที่ผ่านการฟิลเตอร์เหล่านั้นลงไปในตาราง จากนั้นจะเข้าสู่การทำงานส่วนหลัง ในส่วนหลังจะทำการจัดกลุ่มข้อมูล พร้อมนับความถี่แต่ละกลุ่มของข้อมูล การจัดกลุ่มข้อมูลจะเป็นการจัดกลุ่มโดยการเลือก Colum มาจัดกลุ่ม ซึ่งจะเริ่มจากส่งคู่ Key ซึ่งก็คือชื่อกลุ่มข้อมูลที่จะใช้จัดกลุ่ม และ Value ซึ่งมีค่าเป็นหนึ่ง สำหรับไว้นับความถี่ของที่เกิดขึ้นของแต่ละกลุ่ม ไปยัง รีดิวซ์ (Reduce) เพื่อทำการจัดการรวมความถี่ของแต่ละกลุ่ม ทั้งนี้ Hive สามารถทำงานได้ทั้ง interactive mode ซึ่งเป็นโหมดที่

สามารถป้อนคำสั่งให้ทำงานได้โดยตรง หรือ ทำงานเป็นสคริปท์ไฟล์ได้ (SQL Script) ทำให้สามารถสร้างไฟล์ที่ได้ผลลัพธ์จากการค้นหาได้หลากหลาย พร้อมๆกันได้ ซึ่งลักษณะภาษา Hive Pig Latin และ JAVA จะแตกต่างโดยสิ้นเชิง ภาษา JAVA เป็นภาษาโปรแกรมคอมพิวเตอร์ ซึ่งจะมีการเรียนใช้แบบ ออปเจ็ค โอเรียนเต็ด (Object oriented) ส่วน Pig Latin จะเป็นในลักษณะที่เป็นการกำหนดตัวแปรซ้อนชั้นไปเรื่อยๆ โดยอาตัวแปรเหล่านั้นไปใส่ไว้ในฟังก์ชันในการจัดการข้อมูลที่มีมาให้สำเร็จรูป แต่ Hive จะเป็นในลักษณะภาษา SQL

4.2.2 การทดลองโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file

ในการทดลองจะทำการเปรียบเทียบข้อแตกต่าง ของโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับ Log file ที่ถูกพัฒนามาจากระหว่าง ภาษาจาวา (JAVA) Pig Latin และ Hive ซึ่ง Log file ที่นำมาใช้ทดสอบได้แก่ Get.log.cache ซึ่งเป็น proxy log อย่างหนึ่งเพื่อเก็บรายละเอียดการใช้งานเว็บไซต์ ซึ่งขนาดข้อมูลเท่ากับ 5.2 จิกะไบต์ Server log (Sys log) รวม 1.8 จิกะไบต์ ซึ่งเป็น Log file ที่เป็นการบอกสถานะต่างๆของเครื่องเซิร์ฟเวอร์ รวม 1.8 จิกะไบต์ และ Webmail log จำนวน 5.15 เมกะไบต์ ซึ่งเป็น Log ของเว็บแอพพลิเคชันที่เปิดให้บริการใช้งานอีเมลล์บนเว็บ ซึ่งจะเก็บรายละเอียดข้อมูลที่ผู้ใช้งานเข้ามาใช้ระบบได้

ไฟล์ที่ 4.6 ไฟล์ตัวอย่าง Server log

```
Nov  3 04:08:47 auranuch snmpd[26233]: Received SNMP packet(s) from
UDP: [192.168.42.191]:4895
Nov  3 04:09:29 auranuch snmpd[26233]: Connection from UDP:
[192.168.42.191]:4930
Nov  3 04:09:29 auranuch snmpd[26233]: Received SNMP packet(s) from
UDP: [192.168.42.191]:4930
Nov  3 04:09:29 auranuch snmpd[26233]: Connection from UDP:
[192.168.42.191]:4931
```

สำหรับการนำข้อมูล Server log ไปประมวลผลด้วยโปรแกรม จะทำการประมวลผลข้อมูล 3 ขนาดด้วยกัน คือ 434 เมกะไบต์ 1.05 จิกะไบต์ และ 1.85 จิกะไบต์ แต่ละขนาดจะทำการทดลอง 3 รูปแบบด้วยกัน คือ

- ไม่มีฟิลเตอร์ค้นหา และไม่มีฟิลเตอร์วัน
- ฟิลเตอร์ค้นหา คำว่า Error และฟิลเตอร์ในช่วงเวลา 9/5/13-19/11/13
- ฟิลเตอร์ค้นหา คำว่า Connection และฟิลเตอร์ในช่วงเวลา 9/5/13-19/11/13

เมื่อทำการทดลองโปรแกรมพบว่า ผลลัพธ์ที่จากได้จากการประมวลผลมีการจัดกลุ่ม และนับความถี่ของเหตุการณ์ที่เกิดขึ้นในแต่ละประเภทของกลุ่มนั้น ซึ่งแต่ละประเภทของการจัดกลุ่มเป็นหมวดหมู่ออกมามีดังนี้

ตารางที่ 4.2 ประเภทของการจัดกลุ่ม Server log

ประเภทกลุ่ม	ความหมาย
Day	จัดกลุ่มทุกวันที่เดียวกัน
Month	จัดกลุ่มทุกเดือนเดียวกัน
Hour	จัดกลุ่มทุกเลขชั่วโมงเดียวกัน
Minute	จัดกลุ่มทุกเลขนาทีเดียวกัน
Second	จัดกลุ่มทุกเลขวินาทีเดียวกัน
Each Month	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละเดือน
Each Day	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละวัน
Each Hour	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละชั่วโมง
Each Minute	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละนาที
Hostname	จัดกลุ่มที่มีชื่อโฮสต์เนมเดียวกัน
Log Detial	แสดงรายละเอียดของ Log File หลังทำการฟิลเตอร์
Protocol	จัดกลุ่มที่มีชื่อโพรโตคอลเดียวกัน
Count All	นับความถี่ที่เกิดขึ้นทั้งหมด หลังทำการฟิลเตอร์

สำหรับ Pig และ Hive จะมีหมวดหมู่ประเภทการจัดกลุ่มที่คล้ายคลึงกับของ จาวา (JAVA) แต่ Pig จะไม่มีในส่วนของ Count All ส่วน Hive ไม่มีเก็บหมวด Count All ในตารางแต่สามารถเรียกด้วยคำสั่ง SQL ได้ แต่สำหรับ JAVA จะไม่มีการจัดกลุ่ม Seconds Hostname และ Protocol

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
all	dir				2014-03-07 06:41	rwxr-xr-x	hadoop	supergroup
day	dir				2014-03-07 06:31	rwxr-xr-x	hadoop	supergroup
eachday	dir				2014-03-07 06:37	rwxr-xr-x	hadoop	supergroup
eachhour	dir				2014-03-07 06:38	rwxr-xr-x	hadoop	supergroup
eachminute	dir				2014-03-07 06:39	rwxr-xr-x	hadoop	supergroup
eachmonth	dir				2014-03-07 06:36	rwxr-xr-x	hadoop	supergroup
hour	dir				2014-03-07 06:33	rwxr-xr-x	hadoop	supergroup
minute	dir				2014-03-07 06:35	rwxr-xr-x	hadoop	supergroup
month	dir				2014-03-07 06:32	rwxr-xr-x	hadoop	supergroup
tmpDetial	dir				2014-03-07 06:30	rwxr-xr-x	hadoop	supergroup

รูปที่ 4.10 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Server log ด้วยภาษา JAVA

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
dDay	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dFilter	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dHost	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dHour	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dMin	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dMonth	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dProtocol	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
dSec	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
deachDay	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
deachHour	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup
deachMin	dir				2014-03-10 10:03	rwxr-xr-x	hadoop	supergroup

รูปที่ 4.11 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Server log ด้วยภาษา Pig

```
hive> show tables;
ms
tablequery_dayms
tablequery_eachdayms
tablequery_eachminutems
tablequery_hostms
tablequery_hourms
tablequery_minutems
tablequery_monthms
tablequery_ms
tablequery_protocolms
tablequery_secms
view_ms
Time taken: 0.028 seconds, Fetched: 12 row(s)

hive>
```

รูปที่ 4.12 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Server log ด้วยภาษา Hive

สำหรับจัดกลุ่มข้อมูลแบบไม่มีฟิลเตอร์ค้นหา และไม่มีฟิลเตอร์ช่วงเวลา จะทำการจัดกลุ่มดูข้อมูลประเภทเวลาชั่วโมงเดียวกัน (Hour) เพื่อดูว่า ช่วงเวลาไหนเกิดข้อมูลมากที่สุด ซึ่งฝั่งซ้ายของข้อมูลผลลัพธ์เป็นเลขชั่วโมง และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.7 ผลลัพธ์จากการประมวลผลข้อมูล Server log แบบไม่มีฟิลเตอร์ค้นหา และไม่มีฟิลเตอร์ช่วงเวลา หมวด Hour

0	259544
1	254873
2	256333
3	249571
4	237398
5	251350
6	263906
7	504239
8	1582889
9	1647505
10	1558843
11	1542338
12	1419410
13	1553919
14	1546392
15	1566300
16	1563332
17	1375140
18	518771
19	345661
20	281821
21	287170
22	265186
23	263314

จากไฟล์ที่ 4.7 พบว่าในการประมวลผลข้อมูลทั้งหมดของ Server log พบว่าในช่วงเวลา 9.00 น. ถึง 9.59 น. มีการรับ Log เข้ามาในระบบมากที่สุด

สำหรับจัดกลุ่มข้อมูลแบบ Log ที่เกิด Error ในระบบ และค้นหาในช่วงวันที่ 9/5/13 ถึง 19/11/13 จะทำการจัดกลุ่มดูข้อมูลประเภทเวลาชั่วโมงเดียวกัน (Hour) เพื่อดูว่า ช่วงเวลาไหนเกิดข้อมูล Error มากที่สุด ซึ่งฝั่งซ้ายของข้อมูลผลลัพธ์เป็นเลขชั่วโมง และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.8 ผลลัพธ์จากการประมวลผลข้อมูล Server log แบบ Log ที่เป็น Error และฟิลเตอร์ในช่วงวันที่ 9/5/13 ถึง 19/11/13 หมวด Hour

0	8
1	16
2	17
3	18
4	527
5	189
6	49
7	685
8	1932
9	1242
10	1191
11	1050
12	1338
13	1068
14	1641
15	1362
16	866
17	648
18	234
19	35
20	42
21	45
22	12
23	17

จากไฟล์ที่ 4.8 พบว่าการประมวลผลข้อมูลทั้งหมดของ Server log ในช่วงวันที่ 9/5/13 ถึง 19/11/13 พบว่าในช่วงเวลา 8.00 น. ถึง 8.59 น. เกิด Error Log เข้ามาในระบบมากที่สุด ดังนั้นจึงควรตรวจสอบความผิดปกติของระบบ หรือพฤติกรรมการทำงานของระบบ แล้วตรวจสอบว่า เหตุใดจึงความผิดพลาดของระบบในช่วงเวลา 8.00 ถึง 8.59 น. บ่อยครั้งที่สุด

สำหรับจัดกลุ่มข้อมูลที่มีเหตุการณ์การทำ Connection กับระบบ และค้นหาในช่วงวันที่ 9/5/13 ถึง 19/11/13 จะทำการจัดกลุ่มดูข้อมูลประเภทเวลาชั่วโมงเดียวกัน (Hour) เพื่อดูว่าช่วงเวลาไหนเกิดข้อมูลที่เกี่ยวกับการ Connection ระบบมากที่สุด ซึ่งฝั่งซ้ายของข้อมูลผลลัพธ์เป็นเลขชั่วโมง และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.9 ผลลัพธ์จากการประมวลผลข้อมูล Server log แบบ Log ที่มีการทำ Connection และฟิลเตอร์ในช่วงวันที่ 9/5/13 ถึง 19/11/13 หมวด Hour

```
0       18226
1       18041
2       18398
3       18156
4       17934
5       18041
6       19626
7       34933
8       101353
9       84592
10      75346
11      75472
12      64122
13      76106
14      75554
15      90616
16      76256
17      63503
18      25006
19      19968
20      19734
21      26795
22      18249
23      18664
```

จากไฟล์ที่ 4.9 พบว่าการประมวลผลข้อมูลทั้งหมดของ Server log ในช่วงวันที่ 9/5/13 ถึง 19/11/13 พบว่าในช่วงเวลา 8.00 น. ถึง 8.59 น. เกิด Connection เข้ามาในระบบมากที่สุด ทั้งนี้จำนวนเหตุการณ์สูงสุดที่เกิดขึ้น Connection กับระบบ มีความสอดคล้องกับ ความผิดปกติที่เกิดขึ้นในระบบด้วย ดังนั้นจึงสรุปได้ว่า ที่เกิดความผิดปกติของระบบสูงที่สุดในเวลา 8.00 น. ถึง 8.59 น. เพราะมีการเชื่อมต่อติดต่อในระบบมากที่สุดในช่วงเวลา 8.00 น. ถึง 8.59 น.

จากนั้นในการทดลองทำการจับเวลาในการประมวลผลโปรแกรมค้นหา และจัดกลุ่มให้กับข้อมูล Server Log ของแต่ละประเภทการฟิลเตอร์ แต่ละประเภทโปรแกรม และแต่ละขนาดของข้อมูล ได้ผลลัพธ์ดังนี้

ตารางที่ 4.3 ผลการทดลองเวลาที่ใช้การประมวลผล Server log ขนาด 434 เมกะไบต์

Filter Search	Filter Time	Java	Pig	Hive
No	No	6m16s	1m 54s	7m 54s
Error	9/5/13-19/11/13	3m15s	0m 51s	7m 32s
Connection	9/5/13-19/11/14	4m03s	1m 15s	6m 37s

ตารางที่ 4.4 ผลการทดลองเวลาที่ใช้การประมวลผล Server log ขนาด 1.05 จิกะไบต์

Filter Search	Filter Time	Java	Pig	Hive
No	No	13m32s	3m 3s	18m 21s
Error	9/5/13-19/11/13	3m36s	1m 21s	10m 17s
Connection	9/5/13-19/11/14	4m11s	1m 18s	7m 56s

ตารางที่ 4.5 ผลการทดลองเวลาที่ใช้การประมวลผล Server log ขนาด 1.85 จิกะไบต์

Filter Search	Filter Time	Java	Pig	Hive
No	No	18m37s	5m 3s	24m 21s
Error	9/5/13-19/11/13	4m16s	1m 4s	11m 55s
Connection	9/5/13-19/11/14	4m51s	1m 56s	9m 41s

จากตารางที่ 4.1 4.2 และ 4.3 พบว่าเมื่อขนาดข้อมูล Server log เพิ่มขึ้น พบว่าเวลาที่ใช้ในการประมวลผลก็เพิ่มขึ้นด้วย และความเร็วที่ใช้ในการประมวลผลที่ใช้เวลาน้อยที่สุดเป็นแอพลิเคชันที่ถูกสร้างด้วย Pig Latin

ไฟล์ที่ 4.10 ไฟล์ตัวอย่าง proxy log

```
Sat Oct 12 03:29:09 2013 192.168.1.231 GET
http://www.live.com/favicon.ico - NONE/- text/html
```

สำหรับการนำข้อมูล Server log ไปประมวลผลด้วยโปรแกรม จะทำการประมวลผลข้อมูล 3 ขนาดด้วยกัน คือ 536 เมกะไบต์ 1.8 จิกะไบต์ และ 5.19 จิกะไบต์ แต่ละขนาดจะทำการทดลอง 2 รูปแบบด้วยกัน คือ

- ฟิลเตอร์ค้นหา คำว่า www.pantip.com และฟิลเตอร์ในช่วงเวลา 4/1/14-14/1/14 สำหรับข้อมูลขนาด 536 เมกะไบต์ 1.8 จิกะไบต์ และ 5.19 จิกะไบต์

เมื่อเมื่อทำการทดลองโปรแกรมพบว่า ผลลัพธ์ที่จากได้จากการประมวลผลมีการจัดกลุ่ม และนับความถี่ของเหตุการณ์ที่เกิดขึ้นในแต่ละประเภทของกลุ่มนั้น ซึ่งแต่ละประเภทของการจัดกลุ่มเป็นหมวดหมู่ออกมามีดังนี้

ตารางที่ 4.6 ประเภทของการจัดกลุ่ม Proxy log

ประเภทกลุ่ม	ความหมาย
Day	จัดกลุ่มทุกวันที่เดียวกัน
Month	จัดกลุ่มทุกเดือนเดียวกัน
Year	จัดกลุ่มทุกเลขปีเดียวกัน
Hour	จัดกลุ่มทุกเลขชั่วโมงเดียวกัน
Minute	จัดกลุ่มทุกเลขนาทีเดียวกัน
Second	จัดกลุ่มทุกเลขวินาทีเดียวกัน
Each Month	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละเดือน
Each Day	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละวัน
Each Hour	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละชั่วโมง
Each Minute	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละนาที
Name Day	จัดกลุ่มทุกชื่อวันมีชื่อเดียวกัน
Protocol	จัดกลุ่มที่มีชื่อโพรโตคอลเดียวกัน
Host name	จัดกลุ่มที่มีชื่อโฮสต์เนมเดียวกัน
IP Connect	นับความถี่ที่เกิดขึ้นทั้งหมด หลังทำการฟิลเตอร์
Website Tail	จัดกลุ่มที่มีชื่อ Url Path เดียวกัน
Type Source	จัดกลุ่มที่มีประเภทของการดาวน์โหลดข้อมูลเดียวกัน
Source IP	จัดกลุ่มประเภทที่มีหมายเลขไอพีเดียวกัน
Web Domain	จัดกลุ่มที่มีชื่อโดเมนเดียวกัน
Log Detial	แสดงรายละเอียดของ Log File หลังทำการฟิลเตอร์
Count All	นับความถี่ที่เกิดขึ้นทั้งหมด หลังทำการฟิลเตอร์

สำหรับ Pig และ Hive จะมีหมวดหมู่ประเภทการจัดกลุ่มที่คล้ายคลึงกับของ จาวา (JAVA) แต่ Pig จะไม่มีในส่วนของ Count All ส่วน Hive ไม่มีเก็บหมวด Count All ในตารางแต่สามารถเรียกด้วยคำสั่ง SQL ได้ แต่ JAVA ไม่มีส่วนของ Url Path ของ Domain เพื่อเป็นการตัดคำออกไปจากระบบทำให้ระบบสามารถประมวลผลได้เร็วยิ่งขึ้น

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
all	dir				2014-03-07 09:32	rwxr-xr-x	hadoop	supergroup
day	dir				2014-03-07 09:26	rwxr-xr-x	hadoop	supergroup
domain	dir				2014-03-07 09:29	rwxr-xr-x	hadoop	supergroup
eachday	dir				2014-03-07 09:31	rwxr-xr-x	hadoop	supergroup
eachhour	dir				2014-03-07 09:31	rwxr-xr-x	hadoop	supergroup
eachminute	dir				2014-03-07 09:32	rwxr-xr-x	hadoop	supergroup
eachmonth	dir				2014-03-07 09:30	rwxr-xr-x	hadoop	supergroup
hostc	dir				2014-03-07 09:29	rwxr-xr-x	hadoop	supergroup
hosts	dir				2014-03-07 09:30	rwxr-xr-x	hadoop	supergroup
hour	dir				2014-03-07 09:28	rwxr-xr-x	hadoop	supergroup
ip	dir				2014-03-07 09:29	rwxr-xr-x	hadoop	supergroup
minute	dir				2014-03-07 09:28	rwxr-xr-x	hadoop	supergroup
month	dir				2014-03-07 09:27	rwxr-xr-x	hadoop	supergroup
nameday	dir				2014-03-07 09:28	rwxr-xr-x	hadoop	supergroup
protocol	dir				2014-03-07 09:29	rwxr-xr-x	hadoop	supergroup
tmpDetial	dir				2014-03-07 09:26	rwxr-xr-x	hadoop	supergroup
type	dir				2014-03-07 09:30	rwxr-xr-x	hadoop	supergroup
year	dir				2014-03-07 09:27	rwxr-xr-x	hadoop	supergroup

รูปที่ 4.13 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Proxy log ด้วยภาษา JAVA

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
dDay	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dDomain	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dFilter	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dHost	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dHour	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dMin	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dMonth	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dNameday	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dProtocol	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dSec	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dSource	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dTail	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dType	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dYear	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
deachDay	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
deachHour	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
deachMin	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup
dipAdd	dir				2014-03-07 13:01	rwxr-xr-x	hadoop	supergroup

รูปที่ 4.14 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Proxy log ด้วยภาษา Pig

สำหรับจัดกลุ่มข้อมูลที่มีเหตุการณ์ที่ผู้ใช้งานเรียกใช้บริการเว็บไซต์พันทิพย์ (www.pantip.com) ในช่วงระหว่างวันที่ 4/1/14 ถึง 14/1/14 จะทำการจัดกลุ่มดูข้อมูลประเภทเวลาชั่วโมงเดียวกัน (Hour) เพื่อดูว่า ช่วงเวลาไหนมีผู้ใช้งานเข้าใช้บริการเว็บไซต์พันทิพย์มากที่สุด ซึ่งฝั่งซ้ายของข้อมูลผลลัพธ์เป็นเลขชั่วโมง และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.11 ผลลัพธ์จากการประมวลผลข้อมูล Proxy log แบบ Log ที่มีการเรียกใช้บริการ www.pantip.com และฟิลเตอร์ในช่วงวันที่ 4/1/14 ถึง 14/1/14 หมวด Hour

```
6       10
7       19
8       38
9       22
10      29
11      84
12      15
13      49
14      110
15      44
```

16	79
17	59
18	12

จากไฟล์ที่ 4.11 พบว่าการประมวลผลข้อมูลทั้งหมดของ Proxy log ในช่วงวันที่ 4/1/14 ถึง 14/1/14 พบว่าในช่วงเวลา 14.00 น. ถึง 14.59 น. มีจำนวนเหตุการณ์เรียกใช้บริการเว็บพันทิพย์จาก Proxy server มากที่สุด

นอกจากนี้ ยังสามารถทราบได้ว่า ผู้ใช้งานนิยมเข้าใช้บริการเว็บไซต์พันทิพย์ วันใดมากที่สุดในช่วงระหว่างวันที่ 4/1/14 ถึง 14/1/14 โดยดูผ่านการจัดกลุ่มข้อมูลประเภทชื่อวันเดียวกัน (Name day) เพื่อดูแนวโน้มความนิยมในการเล่นเว็บพันทิพย์ว่าผู้ใช้งาน นิยมเข้าเว็บพันทิพย์ในช่วงวันไหนมากที่สุด สำหรับผลลัพธ์จากการประมวลผลด้วยโปรแกรมภาษาจาวา ฝั่งซ้ายของข้อมูลผลลัพธ์เป็นหมายเลขของชื่อวัน และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล สำหรับผลลัพธ์จากการประมวลผลด้วยโปรแกรมภาษา Pig ฝั่งซ้ายของข้อมูลผลลัพธ์เป็นชื่อวัน และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.12 ผลลัพธ์จากการประมวลผลข้อมูล Proxy log แบบ Log ที่มีการเรียกใช้บริการ www.pantip.com และฟิลเตอร์ในช่วงวันที่ 4/1/14 ถึง 14/1/14 หมวด Nameday ด้วยภาษา JAVA

2	132
3	259
4	47
5	55
6	62
7	15

ไฟล์ที่ 4.13 ผลลัพธ์จากการประมวลผลข้อมูล Proxy log แบบ Log ที่มีการเรียกใช้บริการ www.pantip.com และฟิลเตอร์ในช่วงวันที่ 4/1/14 ถึง 14/1/14 หมวด Nameday ด้วยภาษา Pig

Mon	132
Tue	259
Wed	47
Thu	55
Fri	62
Sat	15

จากไฟล์ที่ 4.12 และ 4.13 พบว่าผลลัพธ์ที่ได้จากการค้นหา Proxy log โดยทำการค้นหาเหตุการณ์ที่มีการเรียกใช้งานเว็บไซต์พันทิพย์ ในช่วงวันที่ 4/1/14 ถึง 14/1/14 เมื่อทำการสังเกตผลลัพธ์ทั้งการประมวลผลผ่านจาวา และการประมวลผลผ่าน Pig พบว่าทั้งมีผลลัพธ์เดียว แต่ต่างกันสำหรับผลลัพธ์ที่ประมวลผลด้วยภาษาจาวาจะเป็นจัดกลุ่มชื่อวันด้วยตัวเลข แต่สำหรับการประมวลผลด้วย Pig กลับจัดกลุ่มด้วยชื่อวัน

จากไฟล์ที่ 4.12 และ 4.13 พบว่าการประมวลผลข้อมูลทั้งหมดของ Proxy log ในช่วงวันที่ 4/1/14 ถึง 14/1/14 พบว่า วันอังคาร มีจำนวนเหตุการณ์เรียกใช้บริการเว็บพันทิพย์จาก Proxy server มากที่สุด

จากนั้นในการทดลองทำการจับเวลาในการประมวลผลโปรแกรมค้นหา และจัดกลุ่มให้กับข้อมูล Proxy Log ของแต่ละขนาดของข้อมูล โดยให้มีการฟิลเตอร์เดียวกัน ได้ผลลัพธ์ดังนี้

ตารางที่ 4.7 ผลการทดลองเวลาที่ใช้การประมวลผล Proxy log

Size	Filter Search	Filter Time	Java	Pig
5.19 GB	www.pantip.com	4/1/14-14/1/14	7m 31s	25m 35s
1.8 GB	www.pantip.com	4/1/14-14/1/14	6m 11s	11m 25s
536 MB	www.pantip.com	4/1/14-14/1/14	6m 1s	4m 0s

จากตารางที่ 4.6 พบว่าเมื่อขนาดข้อมูล Proxy log เพิ่มขึ้น พบว่าเวลาที่ใช้ในการประมวลผลก็เพิ่มขึ้นด้วย และความเร็วที่ใช้ในการประมวลผลที่ใช้เวลาน้อยที่สุดเป็นแอพลิเคชันที่ถูกสร้างด้วย จาวาแมปรีดิวซ์ (JAVA Map Reduce) สำหรับขนาดข้อมูล 5.19 จิกะไบต์ และ 1.8 จิกะไบต์ แต่สำหรับข้อมูลที่มีขนาด 536 เมกะไบต์ กลับพบว่า แอพลิเคชันที่ถูกสร้างด้วย Pig Latin กลับใช้เวลาประมวลผลน้อยกว่า

จากตารางที่ 4.2-4.4 และ 4.6 พบว่า เมื่อมีการค้นหา Proxy log ด้วยคำนั้น การประมวลผลด้วย Pig จะใช้เวลาในการประมวลผลมากกว่าประมวลผลด้วยภาษา JAVA เนื่องจากการ Search ซึ่ง Pig จะสามารถค้นหาคำทุกคำที่อยู่หรือเป็นส่วนหนึ่งบนบรรทัดนั้นด้วยการกำหนดรูปแบบของ Regular Expression แต่โปรแกรมภาษา JAVA สามารถค้นหาได้เพียง คำที่ขึ้นต้น หรือ คำที่ลงท้ายของแต่ละ Token เท่านั้น พร้อมทั้งจำนวนคอลัมน์ในการค้นหาที่มากกว่า Server log และ Webmail log เป็นผลทำให้ โปรแกรมภาษา JAVA สามารถตัดคำได้เร็วกว่า แต่ไม่สามารถหาคำที่อยู่ตรงกลางของคำได้

ไฟล์ที่ 4.14 ไฟล์ตัวอย่าง Webmail log

```
03/11/2013 05:35:06 [LOGIN] amp (tpc.co.th) from 172.20.18.59:
03/11/2013 08:14:34 [LOGIN] sitawat.ka (tpc.co.th) from 172.20.18.75:
03/11/2013 09:06:00 [LOGIN_ERROR] N/A (tpc.co.th) from 27.55.0.234:
Your session has expired, but will be resumed after logging in again.
03/11/2013 09:06:25 [LOGIN] areerat.ch (tpc.co.th) from 27.55.0.234:
```

สำหรับการนำข้อมูล Webmail log ไปประมวลผลด้วยโปรแกรม จะทำการประมวลผลข้อมูลที่มีขนาด 5.15 เมกะไบต์ จะทำการทดลอง 4 รูปแบบด้วยกัน คือ

- ไม่มีฟิลเตอร์ค้นหา และไม่มีฟิลเตอร์วัน
- ฟิลเตอร์ค้นหา คำว่า [LOGIN_ERROR] และฟิลเตอร์ในช่วงเวลา 1/11/13-30/11/13

เมื่อเมื่อทำการทดลองโปรแกรมพบว่า ผลลัพธ์ที่จากได้จากการประมวลผลมีการจัดกลุ่ม และนับความถี่ของเหตุการณ์ที่เกิดขึ้นในแต่ละประเภทของกลุ่มนั้น ซึ่งแต่ละประเภทของการจัดกลุ่มเป็นหมวดหมู่ออกมามีดังนี้

ตารางที่ 4.8 ประเภทของการจัดกลุ่ม Webmail log

ประเภทกลุ่ม	ความหมาย	ประเภทกลุ่ม	ความหมาย
Each Month	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละเดือน	Login Type	จัดกลุ่มทุกประเภทล็อกอินเดียวกัน
Each Day	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละวัน	Error type	จัดกลุ่มทุกประเภทความผิดพลาดเดียวกัน
Each Hour	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละชั่วโมง	Host name	จัดกลุ่มที่มีชื่อโฮสต์เนมเดียวกัน
Each Minute	จัดกลุ่มแสดงความถี่ที่เกิดขึ้นแต่ละนาที	Log Detail	แสดงรายละเอียดของ Log File หลังทำการฟิลเตอร์
Hour	จัดกลุ่มทุกเลขชั่วโมงเดียวกัน	IP Connect	จัดกลุ่มทุกเลขไอพีเดียวกัน
Minute	จัดกลุ่มทุกเลขนาทีเดียวกัน	Domain	จัดกลุ่มทุกโดเมนเดียวกัน
Second	จัดกลุ่มทุกเลขวินาทีเดียวกัน	Count All	นับความถี่ที่เกิดขึ้นทั้งหมดหลังทำการฟิลเตอร์
Day	จัดกลุ่มทุกวันที่เดียวกัน	Year	จัดกลุ่มทุกเลขปีเดียวกัน
Month	จัดกลุ่มทุกเดือนเดียวกัน		

สำหรับ Pig และ Hive จะมีหมวดหมู่ประเภทการจัดกลุ่มที่คล้ายคลึงกับของ จาวา (JAVA) แต่ Pig จะไม่มีในส่วนของ Count All ส่วน Hive ไม่มีเก็บหมวด Count All ในตารางแต่สามารถเรียกด้วยคำสั่ง SQL ได้ แต่ JAVA ไม่มีการจัดกลุ่มประเภท Second เพื่อเป็นการตัดคำบางส่วนสำหรับการประมวลผลในส่วนการจัดกลุ่มข้อมูลให้เร็วยิ่งขึ้น

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
all	dir				2014-03-07 11:53	rwxr-xr-x	hadoop	supergroup
day	dir				2014-03-07 11:47	rwxr-xr-x	hadoop	supergroup
domain	dir				2014-03-07 11:50	rwxr-xr-x	hadoop	supergroup
eachday	dir				2014-03-07 11:52	rwxr-xr-x	hadoop	supergroup
eachhour	dir				2014-03-07 11:52	rwxr-xr-x	hadoop	supergroup
eachminute	dir				2014-03-07 11:53	rwxr-xr-x	hadoop	supergroup
eachmonth	dir				2014-03-07 11:51	rwxr-xr-x	hadoop	supergroup
error	dir				2014-03-07 11:51	rwxr-xr-x	hadoop	supergroup
host	dir				2014-03-07 11:50	rwxr-xr-x	hadoop	supergroup
hour	dir				2014-03-07 11:49	rwxr-xr-x	hadoop	supergroup
ip	dir				2014-03-07 11:50	rwxr-xr-x	hadoop	supergroup
minute	dir				2014-03-07 11:49	rwxr-xr-x	hadoop	supergroup
month	dir				2014-03-07 11:48	rwxr-xr-x	hadoop	supergroup
protocol	dir				2014-03-07 11:49	rwxr-xr-x	hadoop	supergroup
tmpDetial	dir				2014-03-07 11:47	rwxr-xr-x	hadoop	supergroup
type	dir				2014-03-07 11:51	rwxr-xr-x	hadoop	supergroup
year	dir				2014-03-07 11:48	rwxr-xr-x	hadoop	supergroup

รูปที่ 4.15 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Webmail log ด้วยภาษา JAVA

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
dDay	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dDomain	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dErrorType	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dFilter	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dHost	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dHour	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dIpConnection	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dLoginType	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dMin	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dMonth	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dSec	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
dYear	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
deachDay	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
deachHour	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup
deachMin	dir				2014-03-07 11:59	rwxr-xr-x	hadoop	supergroup

รูปที่ 4.16 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Webmail log ด้วยภาษา Pig

```
hive> show tables;
OK
ma
tablequery_dayma
tablequery_domainma
tablequery_eachdayma
tablequery_eachhourma
tablequery_eachminutema
tablequery_eachmonthma
tablequery_errortypema
tablequery_hostma
tablequery_hourma
tablequery_ipma
tablequery_logtypema
tablequery_ma
tablequery_minutema
tablequery_monthma
tablequery_secma
tablequery_yearma
view_ma
Time taken: 0.013 seconds, Fetched: 18 row(s)
hive>
```

รูปที่ 4.17 ตัวอย่างผลลัพธ์ของการประมวลผลโดยจัดออกเป็นหมวดหมู่ประเภทการจัดกลุ่มของ Webmail log ด้วยภาษา Hive

สำหรับจัดกลุ่มข้อมูลแบบไม่มีฟิลเตอร์ค้นหา และไม่มีฟิลเตอร์ช่วงเวลา จะทำการจัดกลุ่มดูข้อมูลประเภทเวลาชั่วโมงเดียวกัน (Hour) เพื่อดูว่า ช่วงเวลาไหนเกิดเหตุการณ์ในระบบมากที่สุด ซึ่งฝั่งซ้ายของข้อมูลผลลัพธ์เป็นเลขชั่วโมง และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.15 ผลลัพธ์จากการประมวลผลข้อมูล Webmail log แบบไม่มีฟิลเตอร์ค้นหา และไม่มีฟิลเตอร์ช่วงเวลา หมวด Hour

0	128
1	109
2	43
3	47
4	39
5	44
6	107
7	1774
8	9992
9	7046
10	7422
11	7515
12	2766
13	9169
14	6673
15	7457
16	8789
17	1454
18	405
19	376
20	367
21	357
22	292
23	296

จากไฟล์ที่ 4.15 พบว่าในการประมวลผลข้อมูลทั้งหมดของ Webmail log พบว่าในช่วงเวลา 8.00 น. ถึง 8.59 น. มีการรับ Log เข้ามาในระบบมากที่สุด

สำหรับจัดกลุ่มข้อมูลแบบ Log ที่มีการ Login เข้ามาใช้งานแล้วเกิด Error ในระบบ และค้นหาในช่วงวันที่ 1/11/13 ถึง 10/11/13 จะทำการจัดกลุ่มดูข้อมูลประเภทเวลาชั่วโมงเดียวกัน (Hour) เพื่อดูว่า ช่วงเวลาไหนเกิดข้อมูลที่มีการ Login เข้าใช้งานแล้วเกิดความผิดพลาดมากที่สุด ซึ่งฝั่งซ้ายของข้อมูลผลลัพธ์เป็นเลขชั่วโมง และฝั่งขวาของผลลัพธ์คือความถี่ที่เกิดขึ้นของข้อมูล

ไฟล์ที่ 4.16 ผลลัพธ์จากการประมวลผลข้อมูล Webmail log แบบ Login ที่เกิด Error และฟิลเตอร์ในช่วงวันที่ 1/11/13 ถึง 10/11/13 หมวด Hour

0	1
2	2
3	2
4	2
5	1
6	3
7	28

```
8       97
9       60
10      75
11      59
12      26
13      119
14      69
15      71
16      92
17      17
18      6
19      6
20      9
21      6
22      14
23      13
```

จากไฟล์ที่ 4.16 พบว่าการประมวลผลข้อมูลทั้งหมดของ Webmail log ในช่วงวันที่ 1/11/13 ถึง 10/11/13 พบว่าในช่วงเวลา 13.00 น. ถึง 13.59 น. เกิด Error Log เข้ามาในระบบมากที่สุด ดังนั้นจึงควรตรวจสอบความผิดปกติของระบบ หรือพฤติกรรมการทำงานของระบบ แล้วตรวจสอบว่า เหตุใดจึงความผิดพลาดของระบบในช่วงเวลา 13.00 ถึง 13.59 น. บ่อยครั้งที่สุด

นอกจากนี้สำหรับ Webmail log มีการจัดกลุ่มประเภทของ Error ที่เกิดขึ้นสำหรับการล็อกอินผิดพลาด เพื่อทำให้ผู้วิเคราะห์ความผิดพลาดของระบบ E-mail สามารถตรวจสอบได้ว่า การล็อกอินที่ผิดปกตินั้น มีสาเหตุมาจากอะไรได้บ้าง

ไฟล์ที่ 4.17 ผลลัพธ์จากการประมวลผลข้อมูล Webmail log แบบ Login ที่เกิด Error และฟิลเตอร์ในช่วงวันที่ 1/11/13 ถึง 10/11/13 หมวด Error Type

```
Unknown user or password incorrect. 228
You must be logged in to access this page.      42
Sorry, you did not provide the correct challenge response.  194
This page request could not be verified and appears to have expired.
      17
Your session has expired, but will be resumed after logging in again.
      296
Preference file, /webmail/prefs/designtpc1.pref, could not be copied
from temporary file, /webmail/prefs/designtpc1.pref.tmp. Contact your
system administrator to resolve this issue.     1
```

จากไฟล์ที่ 4.17 พบว่า ปัญหาล็อกอินผิดพลาด มากที่สุดเป็นผลมาจาก เซสชัน (Session) ของผู้ใช้งานหมดอายุ รองลงมาเป็นปัญหาจากการกรอกข้อมูลเข้าใช้งานผิดพลาด ในส่วนนี้อาจเป็นผลมาจากผู้ไม่ประสงค์ดีที่พยายามจะเจาะระบบได้

จากนั้นในการทดลองทำการจับเวลาในการประมวลผลโปรแกรมค้นหา และจัดกลุ่มให้กับข้อมูล Webmail Log ของแต่ละขนาดของข้อมูล โดยให้มีการฟิลเตอร์เดียวกัน ได้ผลลัพธ์ดังนี้

ตารางที่ 4.9 ผลการทดลองเวลาที่ใช้การประมวลผล Webmail log

Filter Search	Filter Time	Java	Pig	Hive
No	No	6m 51s	45s	7m 8s
LOGIN_ERROR	1/11/13-10/11/13	6m 35s	39s	6m 31s

จากตารางที่ 4.8 พบว่าเมื่อขนาดข้อมูล Webmail log ความเร็วที่ใช้ในการประมวลผลที่ใช้เวลาน้อยที่สุดเป็นแอพลิเคชันที่ถูกสร้างด้วย Pig Latin

จากนั้นแสดงจำนวนการสร้างงาน (job) ไปประมวลผลในระบบ

ตารางที่ 4.10 ผลการทดลองจำนวนงานที่ใช้การประมวลผล Log File

ประเภท	JAVA	Pig	Hive
Proxy Log	16	1	-
Server log	10	1	12
Maillog Log	17	1	16

เมื่อมีการประมวลผล Log เพื่อจัดกลุ่มข้อมูล แบบหลายๆ กลุ่มพร้อมกันพบว่าการประมวลผลด้วยภาษา JAVA และ Hive สร้างงาน 1 งานสำหรับ 1 ประเภทการจัดกลุ่ม แต่สำหรับ Pig นั้นสร้างงานเพียงงานเดียวก็สามารถจัดกลุ่มข้อมูลออกมาได้หลายประเภท

จากนั้นจำแนกประเภทในการจัดผลลัพธ์ ที่ได้จากการประมวลผลในระบบ โดยการกำหนดว่า “มี” หมายความว่า มีการทำการจัดกลุ่มตามประเภทกลุ่มที่กำหนด สำหรับ “ไม่มี” หมายความว่า ไม่มีการจัดกลุ่มข้อมูลตามประเภทกลุ่มตามที่กำหนด

ตารางที่ 4.11 ผลการทดลองแสดงประเภทในการจัดกลุ่มที่ใช้การประมวลผล Proxy log ของแต่ละภาษาของโปรแกรม

ประเภทการจัดกลุ่ม	JAVA	Pig
Day	มี	มี
Month	มี	มี
Year	มี	มี
Hour	มี	มี
Minute	มี	มี
Second	ไม่มี	มี
Each Month	มี	มี
Each Day	มี	มี
Each Hour	มี	มี
Each Minute	มี	มี
Name Day	มี	มี
Protocol	มี	มี
Host name	มี	มี
IP Connect	มี	มี
Website Tail	มี	มี
Type Source	มี	มี
Source IP	มี	มี
Web Domain	มี	มี
Log Detail	มี	มี
Count All	มี	ไม่มี

ตารางที่ 4.12 ผลการทดลองแสดงประเภทในการจัดกลุ่มที่ใช้การประมวลผล Server log

ประเภทการจัดกลุ่ม	JAVA	Pig	Hive
Day	มี	มี	มี
Month	มี	มี	มี
Hour	มี	มี	มี
Minute	มี	มี	มี
Second	ไม่มี	มี	มี
Each Month	มี	ไม่มี	ไม่มี
Each Day	มี	มี	มี
Each Hour	มี	มี	มี
Each Minute	มี	มี	มี
Host name	มี	มี	มี
Log Detial	มี	มี	มี
Count All	มี	ไม่มี	ไม่มี

ตารางที่ 4.13 ผลการทดลองแสดงประเภทในการจัดกลุ่มที่ใช้การประมวลผล Webmail log

ประเภทกลุ่ม	JAVA	Pig	Hive	ประเภทกลุ่ม	JAVA	Pig	Hive
Day	มี	มี	มี	Each Minute	มี	มี	มี
Month	มี	มี	มี	Login Type	มี	มี	มี
Year	มี	มี	มี	Error type	มี	มี	มี
Hour	มี	มี	มี	Domain	มี	มี	มี
Minute	มี	มี	มี	Log Detail	มี	มี	มี
Second	ไม่มี	มี	มี	Count All	มี	ไม่มี	ไม่มี
Each Month	มี	มี	มี	Host name	มี	มี	มี
Each Day	มี	มี	มี	IP Connect	มี	มี	มี
Each Hour	มี	มี	มี				

เมื่อทำการประมวลผล Log File พบว่าผลที่ได้มีการจัดกลุ่มข้อมูล พร้อมนับเหตุการณ์แต่ละกลุ่มข้อมูล แบบหลายๆ กลุ่มพร้อมกันพบว่า JAVA Pig และ Hive ซึ่ง Pig และ Hive สามารถจัดกลุ่มข้อมูลได้เหมือนกัน และสามารถเลือกจัดกลุ่มข้อมูลได้ต่างกัน กับ JAVA ยกตัวอย่างเช่น Pig และ Hive ไม่มีประเภทการจัดกลุ่มนับข้อมูลที่เกิดขึ้นทั้งหมด (Count All) และสำหรับโหมด Each Month สำหรับ Server log มีเพียง JAVA เท่านั้นที่มีการจัดกลุ่มประเภทนี้ เนื่องจากตัว Server log ไม่มีการบันทึกข้อมูลระดับปี ดังนั้นการจัดกลุ่ม Each Month จะไม่ต่างกับ Month เลย สำหรับ JAVA จะไม่มีการจัดกลุ่มประเภทวินาที (Second) เนื่องจากการใช้ JAVA มาจัดกลุ่มข้อมูลในหน่อยวินาทีย่อมทำได้ แต่มีความยุ่งยากในเรื่องของการเลือก Token ต้องมีการหาตำแหน่งของหลักวินาทีและไม่เหมาะสม เนื่องจากเหตุการณ์ที่เกิด Log ขึ้น เป็นไปได้ว่าเกิดขึ้นทุกวินาทีอยู่แล้ว

ความแตกต่างของวิธีการจัดกลุ่มทำให้ ทั้ง 3 มีความแตกต่างกันในเรื่องของการเลือกตัวแปรสำคัญมาจัดกลุ่ม สำหรับ JAVA การจัดกลุ่มนั้นขึ้นอยู่กับ การส่ง ค่า Key ที่มีการแมป (Map) กับค่าไปยังฟังก์ชันรีดิวซ์ (Reduce) แล้วทำการรวมผลลัพธ์ ซึ่งขึ้นอยู่กับผู้เขียนโปรแกรมเลือกการจัดกลุ่มได้ตามต้องการ แต่สำหรับ Pig และ Hive จะเป็นการเลือก Colum มาจัดกลุ่มแล้วนับเหตุการณ์

สำหรับผลลัพธ์ที่ได้นั้น สำหรับในการจัดประเภทกลุ่มเดียวกัน พบว่าเหมือนกัน โดยสังเกตจากผลลัพธ์ของการ ประมวลผล Mail log โดยมีการให้กรองช่วงเวลาตั้งแต่วันที่ 11/12/13 ถึงวันที่ 14/12/13 แล้วจัดกลุ่มประเภทแสดง ความถี่ที่เกิดขึ้นของแต่ละวัน (Each day) ดังภาพ

```
11 12 13 0 :   740
12 12 13 0 :   853
13 12 13 0 :   686
14 12 13 0 :   455
```

(ก)

```
(12,11)   740
(12,12)   853
(12,13)   686
(12,14)   455
```

(ข)

```
2013    12      11      740
2013    12      12      853
2013    12      13      686
2013    12      14      455
```

(ค)

รูปที่ 4.18 แสดง ความถี่ที่ของเหตุการณ์ที่เกิดขึ้นของแต่ละวันของ Mail log
(ก)ความถี่ที่ของเหตุการณ์ที่ประมวลผลด้วยภาษา JAVA
(ข)ความถี่ที่ของเหตุการณ์ที่ประมวลผลด้วยภาษา Pig Latin
(ค)ความถี่ที่ของเหตุการณ์ที่ประมวลผลด้วยภาษา Hive

จากรูปที่ 4.18 ผลลัพธ์ที่ได้จากการประมวลผลวิเคราะห์ข้อมูลล็อกจากโปรแกรมที่สร้างด้วยภาษาจาวา (JAVA) Pig และ Hive พบว่า ทั้งสามให้ผลลัพธ์เดียวกัน

ต่อจากการพิสูจน์ผลลัพธ์ที่เกิดขึ้นของทั้งสามภาษาแล้ว ผู้จัดทำได้ทำการหาเวลาที่ใช้ในการประมวลผล เมื่อมีการเพิ่มโหนดเข้ามาทำงานในระบบ โดยในที่นี้จะทำการประมวลผลข้อมูล Server log ขนาด 1.8 จิกะไบต์ โดยทำการฟิลเตอร์ในช่วงวันที่ 9/8/13 ถึง 8/11/13 แล้วจัดกลุ่มออกมาแต่ละประเภท พร้อมความถี่ของเหตุการณ์แต่ละกลุ่ม

ตารางที่ 4.14 ผลการทดลองเวลาที่ใช้ในประผลมวลสำหรับ 15 และ 19 โหนด

node	Java	Pig	Hive
15	9m 56s	4m 3s	11m 6s
19	8m 6s	2m 43s	10m 15s

จากตารางที่ 4.13 พบว่าเมื่อนำโปรแกรมไปประมวลผลข้อมูล Server log ด้วย Java Pig และ Hive เมื่อมีการเพิ่มโหนดจาก 15 โหนด เป็น19 โหนด แล้วทำการประมวลผล พบว่า เวลาที่ใช้ในการประมวลผลทั้ง Java Pig และ Hive จะใช้เวลาในการประมวลผลน้อยลง

4.2.3 สรุปผลการทดลอง

4.2.3.1 สรุปผลการทดลองนับคำทั้งหมดที่ปรากฏ (Word Count)

1) จากตารางที่ 4.2 , 4.3 และ 4.4 พบว่าเมื่อนำ Word มาจับคู่กับ Offset Bytes ของแต่ละไฟล์ ผลลัพธ์ของ Word ที่อยู่บนบรรทัดเหล่านั้นเหมือนกันกับไฟล์ที่ 4.2 4.3 และ 4.4 ที่ใช้สำหรับนำไปประมวลผลในระบบ
2) พบว่าจำนวนการกระจายการทำงานของแมป (Map) เท่ากับจำนวนไฟล์ข้อมูล และเป็นการนำทั้งไฟล์ไปทำงานทั้งแมป (Map) พิสูจน์ได้จากจำนวนของ Map input records ที่ทำหน้าที่บันทึกจำนวนบรรทัดของแต่ละแมป (Map) และ Map output records บันทึกจำนวนคำที่อยู่ในแมป (Map) นั้น ซึ่งมีค่าเท่ากับจำนวนคำในไฟล์ ดังนี้

ตารางที่ 4.14 ผลการทดลองการทำงานของโปรแกรม Log file ส่วน Reduce

	Counter	Map1	Map2	Map3
Map-Reduce	Map input records	4	3	2
	Map output records	12	9	4
Input File		File3.txt	File1.txt	File2.txt

3) ทั้งนี้การเขียนโปรแกรมระบุที่อยู่ของคำยังสามารถนำไปประยุกต์ต่อยอดได้อีกในอนาคต เช่น โปรแกรมค้นหาไฟล์

4.2.3.2 สรุปผลการทดลองโปรแกรมค้นหา และจัดกลุ่มข้อมูลให้กับไฟล์ล็อก

สำหรับการประมวลผลด้วยโปรแกรมวิเคราะห์ Log File สรุปผลลัพธ์ได้ดังนี้

1) Pig Latin ใช้เวลาประมวลข้อมูลน้อยที่สุด
2) แต่ละภาษาที่ใช้สร้างโปรแกรมต่างสามารถจัดกลุ่มได้ลักษณะเฉพาะแบบ
3) การเพิ่มโหนด ทำให้เวลาที่ใช้ในการประมวลผลน้อยลง
4) ผลลัพธ์ที่ได้จากการทดลองด้วยการประมวลผลข้อมูลผ่านโปรแกรมที่ถูกสร้างทั้ง 3 ภาษา ให้ผลลัพธ์เดียวกัน
5) การประมวลผลข้อมูลผ่านโปรแกรมที่ถูกสร้างทั้ง 3 ภาษา เบื้องหลังจะทำการแปลงเป็นจาวาแมปรีดิวซ์ (JAVA Map Reduce)

4.2.3.3 สรุปลักษณะโปรแกรมที่ถูกสร้างด้วยจาวา (JAVA) พิกละติน (Pig Latin) และไฮฟ์ (Hive)

จาวาแมปรีดิวซ์ (Java Map Reduce)

1) เหมาะสำหรับข้อมูลที่มีลักษณะไม่เป็นโครงสร้าง (Unstructure Data) สามารถใช้งานกับ ข้อมูลที่มีลักษณะกึ่งโครงสร้าง (Semi-Structure) ได้
2) เหมาะสำหรับการเขียนโปรแกรมแมปรีดิวซ์ (Map Reduce) ที่มีความซับซ้อน เช่น การเขียนคลาส (Class) การเขียนโปรแกรมเชิงลักษณะ Object Oriented การให้ค่ากับข้อความ
3) เหมาะสำหรับการดึงเอารายละเอียดเฉพาะของข้อมูลตามต้องการของผู้ใช้งานออกมาใช้คำนวณ ประมวลผล ในลักษณะของ Token

4) ไม่เหมาะสำหรับการ Join ข้อมูลจากหลายๆแหล่ง
5) ต้องมีการออกแบบโปรแกรมตั้งแต่เริ่มเขียนโปรแกรม

พิก ละติน (Pig Latin)

1) เหมาะสำหรับข้อมูลที่มีลักษณะกึ่งโครงสร้าง (Semi-Structured)
2) สามารถใช้งานกับข้อมูลที่มีลักษณะไม่เป็นโครงสร้างได้ (Unstructured)
3) เนื่องจาก Pig Latin มีการสร้างการทำงาน เพียงงานเดียว ทำให้เวลาในการประมวลผลน้อยกว่าการเขียนด้วยโปรแกรมด้วยภาษาอื่น เช่น Java Hive
4) เหมาะสำหรับผู้ที่ไม่ถนัดเขียนโปรแกรมด้วยภาษาจาวา (Java) หรือ ภาษาที่มีลักษณะเป็น Object-Oriented ตัวอื่นๆ
5) มีฟังก์ชันสำเร็จรูปที่สามารถใช้งานได้อย่างสะดวก
6) ภาษาในการเขียนเข้าใจง่าย
7) สามารถ Join ข้อมูลได้โดยใช้คำสั่งที่คล้ายภาษา SQL
8) เหมาะสำหรับการทำ Batch processing หมายถึง การประมวลผลข้อมูลขนาดใหญ่ด้วย Script file
9) แต่ไม่เหมาะสำหรับการปรับเปลี่ยนประเภทในการนำเข้าข้อมูล หรือควบคุมการอ่านข้อมูลได้
10) ไม่เหมาะกับอัลกอริทึมของโปรแกรมที่มีความซับซ้อน

ไฮฟ์ (Hive)

1) เหมาะสำหรับข้อมูลที่มีลักษณะโครงสร้าง (Structured)
2) สามารถใช้งานข้อมูลที่มีลักษณะกึ่งโครงสร้าง (Semi-Structured) ได้
3) เหมาะสำหรับผู้ที่ถนัดการใช้ภาษา SQL ในการทำ Data warehouse
4) เวลาที่ใช้ในการประมวลผลสูง เนื่องจากมีการกำหนดการแบ่งบล็อกกระจายไปยังโหนดต่างๆ ที่มีขนาดใหญ่กว่าภาษาอื่น ดังนั้นการทำงานในแต่ละโหนดจึงใช้เวลามากกว่าภาษาอื่น
5) เหมาะทำการ Join ข้อมูลกับตารางอื่นได้
6) ไม่เหมาะข้อมูลที่มีลักษณะไม่เป็นโครงสร้างได้ (Unstructured)
7) Relational Database

บทที่ 5
บทสรุปและข้อเสนอแนะ

5.1 บทสรุป

ระบบฮาดูปเป็นระบบประมวลผลแบบกระจายชนิดหนึ่ง เพื่อเพิ่มประสิทธิภาพในการประมวลผลข้อมูลขนาดใหญ่ เนื่องมาจากปัญหาของข้อมูลที่มีขนาดใหญ่ นอกจากปริมาณข้อมูลที่เพิ่มขึ้นอย่างรวดเร็วแล้ว อีกทั้งข้อมูลเหล่านั้นยังมีความหลากหลายอีกด้วย จึงทำการศึกษาการทำงานของระบบฮาดูป โดยการติดตั้งระบบฮาดูปบนระบบปฏิบัติการอูบุนตู แล้วทำการศึกษาทฤษฎีการทำงานของแมปริดิวซ์ (Map Reduce) (Map Reduce) ซึ่งเป็นส่วนหนึ่งของระบบฮาดูป หลังจากศึกษาทฤษฎีแล้ว จึงศึกษาผ่านตัวอย่างโปรแกรมนับคำ เพื่อให้เห็นภาพ และความเข้าใจที่ชัดเจนมากยิ่งขึ้น จากนั้นได้ศึกษาตัวอย่างโปรแกรมระบุที่อยู่ของคำ และเขียนโปรแกรมเพิ่มเติมส่วนที่ระบุ Offset Bytes ของ Line เพื่อให้ทราบที่อยู่ของคำที่อยู่ชัดเจนยิ่งขึ้นของไฟล์ยิ่งขึ้น ทั้งนี้จุดประสงค์การเขียนโปรแกรมแมปริดิวซ์ (Map Reduce) (Map Reduce) คือ ให้ระบบสามารถประมวลผลข้อมูลขนาดใหญ่ในเวลาเดียวกันได้มากขึ้นโดยการทำงานของโปรแกรมที่กระจายนั้นทำงานเหมือนกัน และทำงานอิสระจากกัน

นอกจากนี้ ผู้จัดทำได้ศึกษา และทดลองระบบแอพพลิเคชั่นเสริมบนฮาดูป นั่นก็คือ อาปาเช่พิก (Apache Pig) ซึ่งมีจุดเด่นในเรื่องภาษาสคริปท์ชั้นสูงที่มีความง่ายต่อการเขียนมากกว่าเมื่อเทียบกับภาษาโปรแกรมมิ่งในลักษณะ Object-Oriented และยังเหมาะสมกับโปรแกรมประเภทวิเคราะห์อีกด้วย แต่พิก (Pig) ยังมีข้อจำกัดมากกว่าจาวาแมปรีดิวซ์ (JAVA Map Reduce) อยู่ อีกทั้งผู้จัดทำได้ทำการศึกษา ไฮฟ์ (Hive) ซึ่งเป็นอีกเครื่องมือหนึ่งที่ช่วยในการประมวลข้อมูลที่เหมาะสำหรับข้อมูลที่เป็นโครงสร้าง แต่ก็ยังมีข้อจำกัดในการทำงานกับข้อมูลลักษณะที่ไม่เป็นโครงสร้าง ท้ายที่สุดแล้วนักพัฒนาต้องเลือกใช้แพลตฟอร์มให้เหมาะสมกับโปรแกรมที่ต้องการพัฒนา เพื่อประสิทธิภาพสูงสุดของการประมวลผลนั่นเอง

5.2 ปัญหาอุปสรรคและแนวทางการแก้ไข

1) ต้องใช้เวลาในการศึกษาและค้นคว้าวิธีการ Implement และ debug ตัวโปรแกรม เนื่องจากวิธีการ Implement หรือ debug มีหลากหลายวิธี จึงต้องใช้เวลาคัดเลือกวิธีการที่เหมาะสมกับชิ้นงานมากที่สุด

2) เนื่องจากเรื่องของฮาดูปเป็นเนื้อหาที่ไม่เคยทราบมาก่อน จึงต้องทำการค้นคว้าหาข้อมูลจากหลายแหล่งมาวิเคราะห์ และเชื่อมโยงเข้าด้วยกัน ทำให้ต้องใช้เวลาในการทำความเข้าใจ
3) เนื่องจากเนื้อหาที่เกี่ยวกับระบบฮาดูป มีอยู่หลายแหล่งข้อมูล มีทั้งจากประเทศไทย และต่างประเทศ ดังนั้นจึงต้องมีวิจารณญาณในการคัดเลือกข้อมูลจากแหล่งที่เชื่อถือได้ เพื่อให้ได้ข้อมูลที่ถูกต้อง
4) ระบบฮาดูปในส่วนของโปรแกรมที่ประมวลผลนั้น การทำงานเบื้องหลังของระบบมีความซับซ้อน และค่อนข้างยากต่อการทำความเข้าใจ จึงทำให้ต้องใช้เวลาในการเข้าถึง และทำความเข้าใจการทำงานเบื้องหลังเหล่านั้น
5) คำอธิบายโค้ดแมปรีดิวซ์ (Map Reduce) เปิดเผยแค่บางส่วนเท่านั้น ทำให้ผู้ศึกษาขาดความเข้าใจในบางส่วนได้
6) การติดตั้งระบบฮาดูป บนระบบประมวลผลบนกลุ่มก้อนเมฆ มีความแตกต่างกันกับการติดตั้งระบบฮาดูปบน VMWARE ดังนั้นผู้จัดทำต้องเข้าใจหลักการทำงานเบื้องต้นของระบบประมวลบนกลุ่มก้อนเมฆด้วย
7) ลักษณะการเขียนโปรแกรมจาวา (JAVA) พิก (Pig) และไฮฟ์ (Hive) มีความแตกต่างกันอย่างชัดเจน ทำให้ผู้จัดทำต้องทำความเข้าใจ และศึกษาวิธีการเขียนโปรแกรมเป็นเวลานาน

5.3 แนวทางในการพัฒนาต่อ

1) การจัดกลุ่มข้อมูล และให้ความถี่ของแต่ละกลุ่มของข้อมูล โดยการประมวลผลแบบแมปรีดิวซ์ (Map Reduce) สามารถนำไปพัฒนาเป็น แอพลิเคชันที่สามารถวิเคราะห์ข้อมูลทางการตลาดได้ โดยการกำหนดจัดกลุ่มข้อมูล
2) สำหรับโปรแกรมวิเคราะห์ล็อก ข้อมูลสามารถนำเอากลุ่มข้อมูลของล็อกต่างประเภทกัน มาเปรียบเทียบความแตกต่างของเหตุการณ์ที่เกิดขึ้น ในช่วงเวลาเดียวกันได้ เพื่อที่หาสาเหตุและปัญหาของระบบได้ละเอียดชัดเจนยิ่งขึ้น

บรรณานุกรม

[1] Tom White. **Hadoop: The Definitive Guide, Third Edition**: O'Reilly Media, Inc.2012.

[2] Edward Capriolo, Dean Wampler, and Jason Rutherglen. **Programming Hive**: O'Reilly Media, Inc.2012.

[3] Jeffrey Dean and Sanjay Ghemawat "**MapReduce: Simplified Data Processing on Large Clusters**" [Online]. Available: http://static.usenix.org/event/osdi04/tech/full_papers/dean/dean.pdf

[4] Brad Hedlund "**Understanding Hadoop Clusters and the Network**" [Online]. Available: http://bradhedlund.com/2011/09/10/understanding-hadoop-clusters-and-the-network/

[5] The Apache Software Foundation. "**Welcome to Apache™ Hadoop®!**" [Online]. Available: http://hadoop.apache.org/

[6] The Apache Software Foundation. "**Apache Hadoop**" [Online]. Available: http://hadoop.apache.org/

[7] The Apache Software Foundation. "**MapReduce Tutorial**" [Online]. Available: http://hadoop.apache.org/docs/stable/mapred_tutorial.html

[8] Michael G. Noll "**Running Hadoop on Ubuntu Linux (Single-Node Cluster)**" [Online]. Available: http://www.michael-noll.com/tutorials/running-hadoop-on-ubuntu-linux-single-node-cluster/

[9] Topgun "**การใช้ iterator**" [Online].
Available: http://www.narisa.com/forums/index.php?showtopic=8238

[10] metha wangthammang "**Java 7 JDK install on Ubuntu** " [Online].
Available: http://maykungth.blogspot.com/2013/04/java-7-jdk-install-on-ubuntu.html

[11] XIAOCHONG ZHANG "**A Simple Example to Demonstrate how does the MapReduce work**" [Online]. Available: http://xiaochongzhang.me/blog/? p=338

[12] Yannarak "**วิธี Configure IP Address บน Linux และ คำสั่งเกี่ยวกับ Network ที่น่าสนใจ**" [Online]. Available: http://www.yannarak.com/node/354

[13] Yahoo Developer Network "**Module 4: MapReduce**" [Online].
Available: http://developer.yahoo.com/hadoop/tutorial/module4.html #chaining

[14] Hadoopers "**Big data page**" [Online].
Available: https://www.facebook.com/hadoopers

[15] Lee Kyu Jae "**Platforms for Big Data**" [Online].
Available: http://www.cubrid.org/blog/dev-platform/platforms-for-big-data/

[16] Apache "**Getting Started Pig**" [Online].
Available: https://pig.apache.org/docs/r0.9.2/start.html

[17] Wikipedia "**Server Message Block**" [Online].
Available: http://en.wikipedia.org/wiki/Server_Message_Block

[18] jaja **"log file คืออะไร"** [Online].
Available: http://jaja-piyaratkeawruangrit.blogspot.com/2011/08/log-file.html

[19] ORACLE **"A Log File Types Supported by Clickstream Intelligence"** [Online].
Available: http://docs.oracle.com/cd/A97329_03/bi.902/a90500/ admin-05.htm

[20] Apache Hive **"Tutorial"** [Online].
Available: https://cwiki.apache.org/confluence/display/Hive/Tutorial#Tutorial-DataUnits

ภาคผนวก ก.

ก1 วิธีการติดตั้ง Hadoop แบบโหนดเดี่ยว (Single node) บน CentOS6.5

1. ติดตั้ง JAVA

1) ทำการดาวน์โหลด JAVA JDK เวอร์ชันที่สูงกว่า 1.6 จาก http://www.oracle.com/technetwork/java/javase/downloads/jdk7-downloads-1880260.html โดยจะเก็บไว้ที่ /opt โดยการใช้คำสั่ง

ไฟล์ ก1.1 คำสั่งเข้าสู่ไดเร็คทอรี opt บนระบบปฏิบัติการ Linux

```
cd /opt
```

2) จากนั้นทำการดาวน์โหลด JAVA ด้วยคำสั่ง wget ตามด้วยลิ้งค์ดาวน์โหลด JAVA

ไฟล์ ก1.2 คำสั่งดาวน์โหลด JAVA จากเว็บไซต์ Oracle.com

```
wget http://download.oracle.com/otn-pub/java/jdk/7u51-b13/jdk-7u51-
linux-
i586.tar.gz?AuthParam=1390449913_60183bed2462c33ac44293ffc30614d9
```

3) จากนั้นก็จะได้รับไฟล์ jdk-7u51-linux-i586.tar.gz ใน /opt ให้ทำการแตกไฟล์ออกมาโดยพิมพ์คำสั่ง

ไฟล์ ก1.3 แตกไฟล์ติดตั้ง JAVA

```
tar xzf jdk-7u51-linux-i586.tar.gz
```

4) จากนั้นทำการติดตั้ง JAVA ด้วยคำสั่ง

ไฟล์ ก1.4 ติดตั้ง JAVA

```
cd /opt/jdk1.7.0_51/
alternatives --install /usr/bin/java java /opt/jdk1.7.0_51/bin/java 2
```

5) จากนั้นทำการเลือกเวอร์ชัน JAVA เพื่อไปใช้งาน ด้วยคำสั่ง alternatives --config java แล้วตามด้วยเลือกเวอร์ชัน JAVA โดยพิมพ์ตัวเลข Selection ลงไป

ไฟล์ ก1.5 เลือกเวอร์ชัน JAVA

```
alternatives --config java

There are 4 programs which provide 'java'.

  Selection    Command
-----------------------------------------------
   1           /opt/jdk1.7.0_25/bin/java
*  2           /usr/lib/jvm/jre-1.6.0-openjdk/bin/java
 + 3           /opt/jdk1.7.0_45/bin/java
   4           /opt/jdk1.7.0_51/bin/java
Enter to keep the current selection[+], or type selection number:
```

6) จากนั้นตั้งค่า JAVA ให้เข้ากับ ตัวแปรสภาพแวดล้อมของ Linux ด้วยคำสั่ง

ไฟล์ ก1.6 ตั้งค่าตัวแปรสภาพแวดล้อมให้กับ JAVA

```
export JAVA_HOME=/opt/jdk1.7.0_51
export JRE_HOME=/opt/jdk1.7.0_51/jre
export PATH=$PATH:/opt/jdk1.7.0_51/bin:/opt/jdk1.7.0_51/jre/bin
```

2. การติดตั้ง Apache Hadoop มีการติดตั้งดังนี้

1) กำหนด User สำหรับใช้งาน Hadoop ในที่นี้จะสร้าง User ชื่อ Hadoop ด้วยคำสั่ง

ไฟล์ ก1.7 ตั้งค่าตัวแปรสภาพแวดล้อมให้กับ JAVA

```
useradd hadoop
passwd hadoop
Changing password for user hadoop.
New password:          //ใส่พาสเวิร์ดให้กับผู้ใช้ฮาดูป
Retype new password:   //ใส่พาสเวิร์ดให้กับผู้ใช้ฮาดูปอีกครั้ง
passwd: all authentication tokens updated successfully.
```

2) ทำการสร้าง Key สำหรับ SSH เพื่อให้ติดต่อสื่อสารระหว่างโหนดได้ด้วยคำสั่ง

ไฟล์ ก1.8 สร้าง ssh key

```
su - hadoop
ssh-keygen -t rsa
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
chmod 0600 ~/.ssh/authorized_keys
ssh localhost
yes
exit
```

3) นำไฟล์ติดตั้ง Hadoop ไปเก็บไว้ที่ /opt ดังนี้

ไฟล์ ก1.9 ทำการวางไฟล์ติดตั้ง Hadoop ให้อยู่ใน /opt

```
mkdir /opt/hadoop
cd /opt
wget http://apache.mesi.com.ar/hadoop/common/hadoop-1.2.1/hadoop-
1.2.1.tar.gz
tar -xzf hadoop-1.2.1.tar.gz
mv hadoop-1.2.1 hadoop
chown -R Hadoop /opt/hadoop
cd /opt/hadoop/
```

4) จากนั้นพิมพ์คำสั่ง vi /opt/hadoop/conf/core-site.xml เพื่อแก้ไฟล์ตั้งค่า core-site.xml ให้เป็นตามไฟล์ดังนี้

ไฟล์ ก1.10 การตั้งค่าไฟล์ core-site.xml

```
<property>
    <name>fs.default.name</name>
    <value>hdfs://localhost:9000/</value>
</property>
<property>
    <name>dfs.permissions</name>
    <value>false</value>
</property>
```

5) จากนั้นพิมพ์คำสั่ง vi /opt/hadoop/conf/hdfs-site.xml เพื่อแก้ไฟล์ตั้งค่า hdfs-site.xml ให้เป็นตามไฟล์ดังนี้

ไฟล์ ก1.11 การตั้งค่าไฟล์ hdfs-site.xml

```
<property>
      <name>dfs.data.dir</name>
      <value>/opt/hadoopdata/dfs/name/data</value>
      <final>true</final>
</property>
<property>
      <name>dfs.name.dir</name>
      <value>/opt/hadoopdata/dfs/name</value>
      <final>true</final>
</property>
<property>
      <name>dfs.replication</name>
      <value>3</value>
</property>
```

6) จากนั้นพิมพ์คำสั่ง vi /opt/hadoop/conf/mapred-site.xml เพื่อแก้ไฟล์ตั้งค่า mapred-site.xml ให้เป็นตามไฟล์ดังนี้

ไฟล์ ก1.12 การตั้งค่าไฟล์ mapred-site.xml

```
<property>
        <name>mapred.job.tracker</name>
      <value>localhost:9001</value>
</property>
```

7) จากนั้นพิมพ์คำสั่ง vi /opt/hadoop/conf/hadoop-env.sh เพื่อแก้ไฟล์ตั้งค่า hadoop-env.sh ให้เป็นเพิ่มเข้าไปในไฟล์ตั้งค่าดังนี้

ไฟล์ ก1.13 การตั้งค่าไฟล์ hadoop-env.sh

```
export JAVA_HOME=/opt/jdk1.7.0_51
export HADOOP_OPTS=-Djava.net.preferIPv4Stack=true
```

8) จากนั้นพิมพ์คำสั่ง mkdir /opt/hadoopdata เพื่อเป็นการสร้าง Path ทำการเก็บ Namenode และ Datanode และพิมพ์คำสั่ง chown -R Hadoop /opt/hadoopdata เพื่อเป็นการตั้งค่าเจ้าของ Path

9) ทำการ format namenode โดยการป้อนคำสั่ง

ไฟล์ ก1.14 ทำ Format Namenode

```
su - hadoop
cd /opt/hadoop/hadoop
bin/hadoop namenode -format
```

10) ถ้าไม่มีปัญหาเรื่องการทำงานของ Java จะปรากฏดังไฟล์

ไฟล์ ก1.15 ทำ Format Namenode

```
13/06/02 22:53:48 INFO namenode.NameNode: STARTUP_MSG:
/************************************************************
STARTUP_MSG: Starting NameNode
STARTUP_MSG:   host = srv1.tecadmin.net/192.168.1.90
STARTUP_MSG:   args = [-format]
STARTUP_MSG:   version = 1.2.1
STARTUP_MSG:   build =
https://svn.apache.org/repos/asf/hadoop/common/branches/branch-1.2 -r
1479473; compiled by 'hortonfo' on Mon May  6 06:59:37 UTC 2013
STARTUP_MSG:   java = 1.7.0_17
```

```
************************************************************/
13/06/02 22:53:48 INFO util.GSet: Computing capacity for map
BlocksMap
13/06/02 22:53:48 INFO util.GSet: VM type       = 32-bit
13/06/02 22:53:48 INFO util.GSet: 2.0% max memory = 1013645312
13/06/02 22:53:48 INFO util.GSet: capacity      = 2^22 = 4194304
entries
13/06/02 22:53:48 INFO util.GSet: recommended=4194304, actual=4194304
13/06/02 22:53:49 INFO namenode.FSNamesystem: fsOwner=hadoop
13/06/02 22:53:49 INFO namenode.FSNamesystem: supergroup=supergroup
13/06/02 22:53:49 INFO namenode.FSNamesystem:
isPermissionEnabled=true
13/06/02 22:53:49 INFO namenode.FSNamesystem:
dfs.block.invalidate.limit=100
13/06/02 22:53:49 INFO namenode.FSNamesystem:
isAccessTokenEnabled=false accessKeyUpdateInterval=0 min(s),
accessTokenLifetime=0 min(s)
13/06/02 22:53:49 INFO namenode.FSEditLog:
dfs.namenode.edits.toleration.length = 0
13/06/02 22:53:49 INFO namenode.NameNode: Caching file names occuring
more than 10 times
13/06/02 22:53:49 INFO common.Storage: Image file of size 112 saved
in 0 seconds.
13/06/02 22:53:49 INFO namenode.FSEditLog: closing edit log:
position=4, editlog=/opt/hadoop/hadoop/dfs/name/current/edits
13/06/02 22:53:49 INFO namenode.FSEditLog: close success: truncate to
4, editlog=/opt/hadoop/hadoop/dfs/name/current/edits
13/06/02 22:53:49 INFO common.Storage: Storage directory
/opt/hadoop/hadoop/dfs/name has been successfully formatted.
13/06/02 22:53:49 INFO namenode.NameNode: SHUTDOWN_MSG:
/************************************************************
SHUTDOWN_MSG: Shutting down NameNode at
srv1.tecadmin.net/192.168.1.90
************************************************************/
```

11) จากนั้นพิมพ์ bin/start-all.sh เพื่อสั่งให้ระบบฮาดูปเริ่มทำงาน

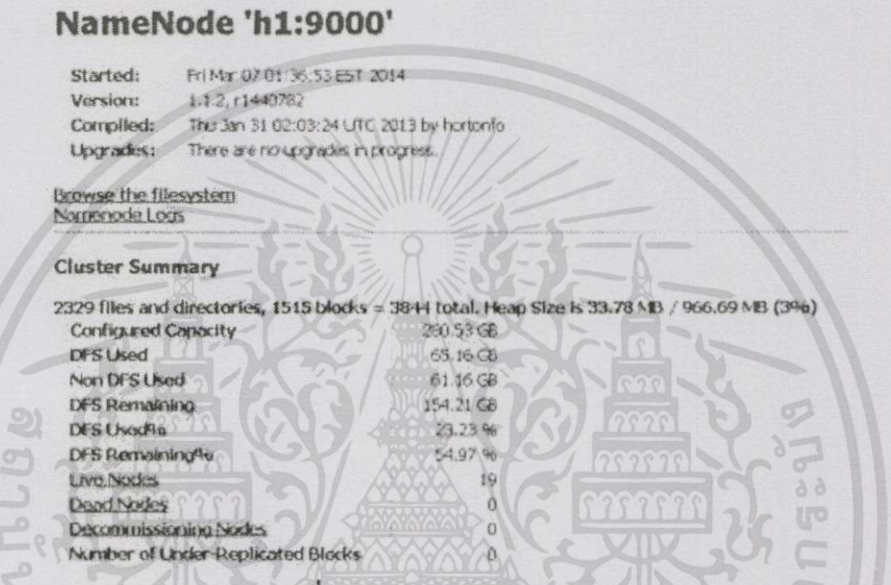
12) อยากทราบว่าระบบฮาดูปทำงานถูกต้องหรือไม่ใช้คำสั่ง jps ถ้าระบบทำงานถูกต้องจะแสดงรายละเอียดดังนี้

ไฟล์ ก1.16 ทำ Format Namenode

```
26049 SecondaryNameNode
25929 DataNode
26399 Jps
26129 JobTracker
26249 TaskTracker
25807 NameNode
```

สามารถตรวจสอบรายละเอียดการทำงานผ่าน Web UI ได้อีกด้วย

- http://localhost:50030/ สำหรับ Jobtracker
- http://localhost:50070/ สำหรับ Namenode
- http://localhost:50060/ สำหรับ Tasktracker



NameNode 'h1:9000'

Started: Fri Mar 07 01:36:53 EST 2014
Version: 1.1.2, r1440782
Compiled: Thu Jan 31 02:03:24 UTC 2013 by hortonfo
Upgrades: There are no upgrades in progress.

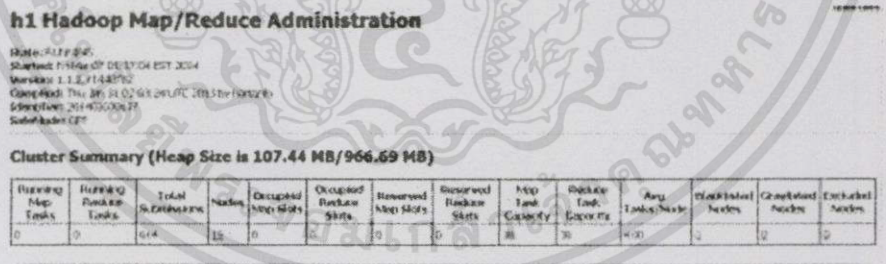
Browse the filesystem
Namenode Logs

Cluster Summary

2329 files and directories, 1515 blocks = 3844 total. Heap Size is 33.78 MB / 966.69 MB (3%)

Configured Capacity	280.53 GB
DFS Used	65.16 GB
Non DFS Used	61.16 GB
DFS Remaining	154.21 GB
DFS Used%	23.23 %
DFS Remaining%	54.97 %
Live Nodes	19
Dead Nodes	0
Decommissioning Nodes	0
Number of Under-Replicated Blocks	0

รูปที่ ก1.1 แสดง Namenode



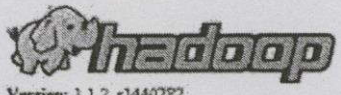
h1 Hadoop Map/Reduce Administration

Cluster Summary (Heap Size is 107.44 MB/966.69 MB)

Running Map Tasks	Running Reduce Tasks	Total Submissions	Nodes	Occupied Map Slots	Occupied Reduce Slots	Reserved Map Slots	Reserved Reduce Slots	Map Task Capacity	Reduce Task Capacity	Avg. Tasks/Node	Blacklisted Nodes	Graylisted Nodes	Excluded Nodes
0	0	644	19	0	0	0	0	38	38	4.00	0	0	0

รูปที่ ก1.2 แสดง Jobtracker

tracker_h1:localhost/127.0.0.1:33350 Task Tracker Status



Version: 1.1.2, r1440782
Compiled: Thu Jan 31 02:03:24 UTC 2013 by hortonfo

Running tasks

Task Attempts	Status	Progress	Errors

Non-Running Tasks

Task Attempts	Status

Tasks from Running Jobs

Task Attempts	Status	Progress	Errors

Local Logs

รูปที่ ก1.3 แสดง Jobtracker