การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจร เว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala

นายจันตพงษ์ บุตรลักษณ์

นักวิชาการคอมพิวเตอร์ ฝ่ายวิศวกรรมระบบเครือข่าย

สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

พ.ศ. ๒๕๕๘

หัวข้อวิจัย : การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala

โดย : นายจันตพงษ์ บุตรลักษณ์

ที่ปรึกษา : ผู้ช่วยศาสตราจารย์ ดร.ชูพันธุ์ รัตนโภคา

แหล่งทุน : สำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศ

ชื่อทุน : ทุนสนับสนุนการทำวิจัยและเสนอผลงานวิจัย

ปีงบประมาณ : 2556

บทคัดย่อ

ปัจจุบันเครื่องคอมพิวเตอร์แม่ข่ายที่ให้บริการระบบสารสนเทศต่างๆผ่านเว็บ มีผู้ใช้เป็นจำนวนมาก ทำให้ข้อมูลจราจรมีขนาดใหญ่ขึ้นเรื่อยๆ ส่งผลให้มีความจำเป็นต้องมีที่จัดเก็บข้อมูลขนาดใหญ่เพื่อการวิเคราะห์ต่างๆ ระบบฐานข้อมูลแบบเชิงสัมพันธ์ที่นิยมใช้งานกันอยู่ในปัจจุบันยังมีประสิทธิภาพไม่ดีมากนักกับการทำงานกับข้อมูลขนาดใหญ่ รวมถึงการเก็บข้อมูลขนาดใหญ่จำเป็นต้องใช้อุปกรณ์ที่มีราคาสูงอีกด้วย

ดังนั้นงานวิจัยนี้ต้องการนำเสนอการออกแบบและพัฒนาเว็บไซต์วิเคราะห์การให้บริการของเครื่องแม่ข่ายเว็บไซต์ เพื่อแสดงกราฟและข้อมูลการให้บริการต่างๆ โดยจะนำเทคโนโลยีของ Hadoop ในส่วน HDFS มาประยุกต์ใช้กับเทคโนโลยีอื่นๆที่เกี่ยวข้องเพื่อช่วยในการเก็บข้อมูลจราจรของเครื่องคอมพิวเตอร์แม่ข่ายเว็บ และใช้ Cloudera Impala ในการสืบค้นข้อมูลเพื่อนำมาใช้ในการวิเคราะห์

ผลการดำเนินงานวิจัย ระบบสามารถแสดงข้อมูลจราจรต่างๆที่ได้จากเครื่องแม่ข่ายเว็บไซต์ สามารถแสดง IP ของผู้เข้าชมเว็บไซต์มากที่สุด 10 อันดับ, เว็บไซต์ที่มีผู้เข้าชมมากที่สุด 10 อันดับ, หน้าเว็บเพจที่มีผู้เข้าชมมากที่สุด 10 อันดับ และ URL ของเว็บไซต์ที่เกิดข้อผิดพลาดจากการเข้าชม 10 อันดับ ให้แสดงออกมาในรูปแบบของกราฟ ซึ่งจากการทดสอบบนคอมพิวเตอร์ 8 เครื่อง ด้วยข้อมูลขนาด 50 กิกะไบต์ พบว่าระบบใช้เวลาในการสืบค้นข้อมูลเพียง 14.4 วินาที

Research Title : Design and Development of Data Traffic Analysis of Web Server on HDFS System using Impala.
Researcher : Mr. Jantapong Boodluck
Advisor : Asst. Professor Choopan Rattanapoka
Funding Sources : Institutes of Computer and Information Technology
Funding Category : Grants for research projects and presentation
Year : 2013

Abstract

Nowadays an extensive usage of web server for Internet Information Services result in a large amount of traffic. Thus very large data storage devices for efficient database operations, analysis and organization are indeed necessary. Relational Database Systems presently employed are not efficient at performing operations that apply to the immense size databases, which require a bigger and more expensive server.

The study aims at the scope of web development and web server analysis via graph representation and data reports of the services. Hadoop HDFS operations were applied with related technology for the server's traffic data storage. Cloudera Impala was applied in the retrieval system for further analysis.

As results, the developed system can display the network traffic on server; a list of top 10 visitors' IP addresses; a list of the top 10 most visited websites; and the top 10 configuration mistakes after a URL reference. These were generated in graph visual representations. The trials were conducted on eight 50-GB desktops while data retrieval takes just 14.4 seconds.

กิตติกรรมประกาศ

งานวิจัยฉบับนี้สำเร็จได้ ด้วยความกรุณาของ ผู้ช่วยศาสตราจารย์ ดร.ชูพันธุ์ รัตนโภคา อาจารย์ที่ปรึกษางานวิจัย ซึ่งได้ให้คำปรึกษา ข้อเสนอแนะ และความช่วยเหลือในหลายสิ่งหลายอย่าง จนกระทั่งลุล่วงไปได้ด้วยดีผู้จัดทำขอกราบขอบพระคุณเป็นอย่างสูงมา ณ ที่นี้

ขอขอบพระคุณ คณะเจ้าหน้าที่และสำนักคอมพิวเตอร์และเทคโนโลยีสารสนเทศที่ได้เอื้อเฟื้ออุปกรณ์ เครื่องมือ สถานที่และให้ความช่วยเหลือในด้านต่างๆ เป็นอย่างดี ขอบคุณ เพื่อนๆ พี่ๆ น้องๆ ทุกคนที่ช่วยทดสอบระบบ ให้คำปรึกษา ข้อเสนอแนะและช่วยเป็นกำลังใจมาโดยตลอด ขอขอบพระคุณ ทุกท่านและผู้ที่มีส่วนร่วมเกี่ยวข้องกับความสำเร็จ แต่มิได้เอ่ยนามทุกท่านมา ณ ที่นี้ด้วย

ท้ายนี้ผู้จัดทำใคร่ขอกราบขอบพระคุณบิดา-มารดา คณะผู้บริหารที่คอยเป็นแรงผลักดัน เป็นกำลังใจ อีกทั้งให้การสนับสนุนในทุกๆด้าน ตลอดจนคณะครู-อาจารย์ทุกท่านที่ได้เคยประสิทธิ์ประสาทวิชาความรู้ รวมทั้งความหวังดีอื่นๆ กระทั่งผู้จัดทำประสบความสำเร็จในการทำงานวิจัย

ผู้จัดทำ

สารบัญ

สารบัญ (ต่อ)

สารบัญตาราง

สารบัญภาพ

สารบัญภาพ (ต่อ)

สารบัญกราฟ

บทที่ 1
บทนำ

1.1 **หลักการและเหตุผล**

เนื่องจากพระราชบัญญัติว่าด้วยการกระทำความผิดเกี่ยวกับคอมพิวเตอร์ พ.ศ.2550 มาตรา 26 ผู้ให้บริการเครือข่ายคอมพิวเตอร์ต้องเก็บรักษาข้อมูลจราจรทางคอมพิวเตอร์ไว้ไม่น้อยกว่าเก้าสิบวัน ซึ่งผู้ให้บริการจะต้องเก็บรักษาข้อมูลของผู้ใช้บริการเท่าที่จำเป็นเพื่อให้ระบุตัวผู้ใช้บริการได้ ดังนั้นภาระจึงเกิดกับผู้ให้บริการเครือข่ายคอมพิวเตอร์ เนื่องด้วยการเก็บข้อมูลที่มีขนาดใหญ่จำเป็นต้องใช้อุปกรณ์ประเภท Network-attached storage (NAS) หรือ Storage area Network (SAN) ที่มีราคาสูง และกรณีที่ต้องการค้นหาข้อมูลของผู้ที่กระทำผิดต้องใช้เวลานานในการสืบค้น เนื่องจากข้อมูลที่เก็บไว้มีขนาดใหญ่

ปัญหาเกี่ยวกับการจัดการข้อมูลขนาดใหญ่ กำลังได้รับความสนใจเป็นอย่างมาก เนื่องจากข้อมูลต่างๆ ที่จัดเก็บมีความสำคัญเพื่อใช้ในการสืบค้น วิเคราะห์ รวมทั้งประมวลผลข้อมูล อีกทั้งข้อมูลที่จัดเก็บยังมีขนาดใหญ่ขึ้นเรื่อยๆ ทำให้การจัดการกับข้อมูลเหล่านั้นมีความลำบากและยุ่งยากมากขึ้น หากจะเลือกข้อมูลเพื่อจัดเก็บบางส่วน สามารถเลือกได้จากความรู้หรือปัจจัยที่สำคัญ ณ เวลานั้นๆ แต่พอเวลาผ่านไปปัจจัยที่สำคัญก็เปลี่ยนไป จึงทำให้ค่อนข้างยากที่จะคาดการอนาคต ซึ่งอาจจะก่อให้เกิดปัญหาตามมา เช่น ข้อมูลที่เก็บไว้ใช้ไม่ได้หรือไม่เพียงพอเพื่อใช้วิเคราะห์ ดังนั้นจำเป็นที่จะต้องจัดเก็บข้อมูลทุกอย่างไว้ ปัจจุบันได้มีการนำเทคโนโลยีที่เรียกว่า HDFS (Hadoop Distributed File System) มาประยุกต์ใช้งานในการเก็บข้อมูลขนาดใหญ่ หลักการทำงานของ HDFS คือแบ่งข้อมูลขนาดใหญ่ที่ต้องการจะเก็บออกเป็นส่วนย่อยๆ แล้วกระจายข้อมูลส่วนย่อยๆ นั้นไปยังเครื่องคอมพิวเตอร์ที่ทำหน้าที่เก็บข้อมูล(DataNode) ส่วนของการค้นหาข้อมูลภายในเทคโนโลยี HDFS วิธีที่เหมาะสมที่สุดเรียกว่าวิธี Map/Reduce ซึ่งเป็นการส่งคำสั่งค้นหาจากเครื่องคอมพิวเตอร์หลัก(NameNode) กระจายไปยังเครื่องคอมพิวเตอร์ที่ทำหน้าที่เก็บข้อมูล(DataNode) โดยที่ไม่จำเป็นต้องมีการย้ายข้อมูลระหว่างการประมวลผล จึงทำให้สามารถค้นหาข้อมูลได้อย่างรวดเร็ว อย่างไรก็ตาม HDFS ทำหน้าที่ประมวลผลแบบคู่ขนาน (Parallel Processing) และจัดเก็บข้อมูลแบบกระจายเท่านั้น การเข้าถึงข้อมูลบน HDFS ซึ่งมีลักษณะที่เป็นแบบ Batch ไฟล์นั้น HDFS ไม่ได้เตรียมการจัดการส่วนนี้ไว้ให้ ดังนั้นนักพัฒนาระบบต้องออกแบบวิธีการเข้าไปจัดการข้อมูลเอง ซึ่งต้องใช้เวลามากพอสมควรและอาจจะได้วิธีการที่ไม่มีประสิทธิภาพมากนัก Impala เข้ามาช่วยนักพัฒนาระบบให้ทำงานกับ HDFS ได้สะดวกขึ้น โดยมีสามารถในการสุ่มการเข้าถึง การอ่าน/เขียนข้อมูลบน HDFS ได้อย่างง่ายดายและมีประสิทธิภาพ ในลักษณะการทำงานแบบระบบฐานข้อมูล SQL ขนาดใหญ่ที่ใช้ HDFS ในการเก็บข้อมูล

จากปัญหาและข้อดีของ HDFS กับ Impala ดังกล่าวข้างต้น งานวิจัยนี้ต้องการนำเสนอการออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala โดยพัฒนาระบบจัดเก็บข้อมูลจราจรเว็บเซิร์ฟเวอร์โดยใช้ Impala ในการจัดการกับ

ข้อมูลที่เก็บไว้บน HDFS เพื่อให้การบริหารและจัดการฐานข้อมูลจราจรเว็บเซิร์ฟเวอร์ขนาดใหญ่มีประสิทธิภาพด้านความเร็วในการค้นหามากยิ่งขึ้น

1.2 **วัตถุประสงค์**

1.2.1 เพื่อศึกษาวิธีการจัดเก็บข้อมูลขนาดใหญ่ด้วย HDFS และการจัดการข้อมูลบน HDFS ด้วย Impala

1.2.2 เพื่อออกแบบและพัฒนาระบบจัดการข้อมูลจราจรเว็บเซิร์ฟเวอร์บนฐานข้อมูลขนาดใหญ่

1.2.3 เพื่อวิเคราะห์ประสิทธิภาพการทำงานของการเก็บข้อมูลแบบกระจาย

1.3 **ขอบเขตของการวิจัย**

1.3.1 ระบบสามารถวิเคราะห์ข้อมูลได้ดังนี้ จำนวนการเข้าใช้เว็บไซต์ในแต่ละวัน แต่ละเดือน และแต่ละปี

1.3.2 ข้อมูลที่ใช้สำหรับทดสอบ 1-100 ล้านเรคคอร์ด(~50 Gb)

1.3.3 ระบบสามารถแสดงผลเป็นกราฟได้ตามการวิเคราะห์ข้อมูล

1.3.4 ระบบสำหรับวิจัยมี 1 NameNode และ 8 DataNode

1.3.5 ระบบที่พัฒนาสามารถจัดการกับข้อมูลจราจรเว็บเซิร์ฟเวอร์ Apache เท่านั้น

1.4 **ประโยชน์ที่คาดว่าจะได้รับ**

1.4.1 ได้วิธีการในการจัดการกับข้อมูลขนาดใหญ่ เพื่อเป็นแนวทางในการพัฒนาระบบใช้งานด้านอื่นๆ ต่อไป

1.4.2 ได้ระบบวิเคราะห์ข้อมูลสำหรับเว็บเซิร์ฟเวอร์ในด้านสถิติการใช้งานเว็บไซต์ เพื่อเป็นข้อมูลสำหรับปรับปรุงระบบการให้บริการได้ดียิ่งขึ้น

1.4.3 ได้ความรู้และความเข้าใจสำหรับการจัดการกับข้อมูลขนาดใหญ่ และเป็นข้อมูลให้คนอื่นๆ ได้ศึกษาพัฒนาต่อไป

บทที่ 2
ทฤษฎีที่สำคัญและงานวิจัยที่เกี่ยวข้อง

ในการดำเนินการวิจัย “ การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala (Design and Development of Data Traffic Analysis of Web Sever on HDFS System using Impala) ” มีการศึกษาข้อมูลเกี่ยวกับโปรแกรมและทฤษฎีที่เกี่ยวข้องกับการสร้างซอฟต์แวร์ดังต่อไปนี้

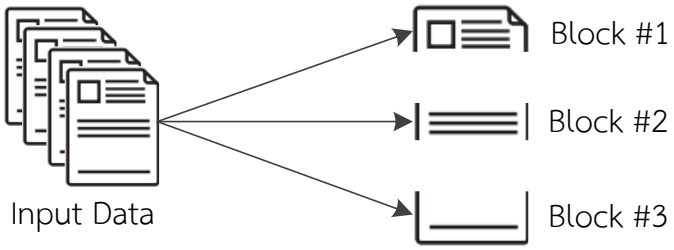
2.1 Cloudera
2.2 Hadoop Distributed File System (HDFS)
2.3 Cloudera Impala
2.4 Apache Flume
2.5 Apache Thrift
2.6 PHP
2.7 HTML
2.8 JavaScript
2.9 CSS

2.1 Cloudera

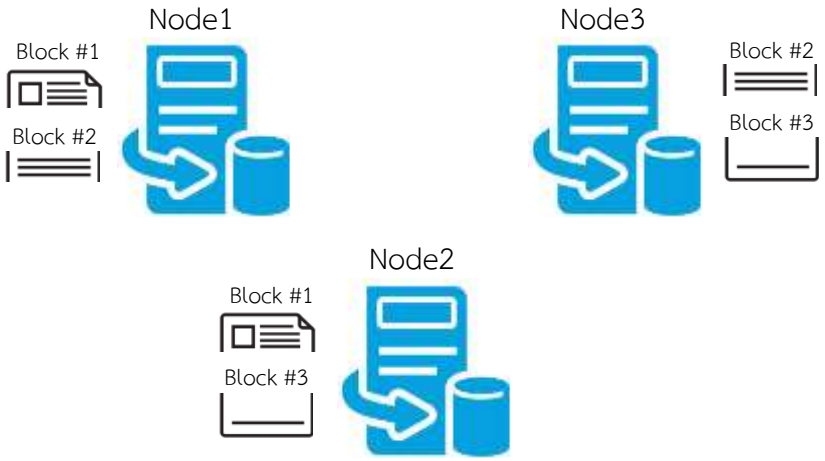
Cloudera คือแพลตฟอร์มสำหรับข้อมูลขนาดใหญ่โดย Cloudera นั้นจะมีสองผลิตภัณฑ์ คือ Cloudera's Distribution สำหรับ Hadoop (CDH) และ Cloudera Enterprise โดยที่ CDH เป็นแพลตฟอร์มสำหรับการกระจายข้อมูล ที่ประกอบด้วย HDFS, Hadoop, MapReduce, Hive, Pig, Hbase, Sqoop, Flume, Oozie, Zookeeper และ HUE นอกจากนี้ยังสามารถ ใช้งานได้ฟรีภายใต้ใบอนุญาตของ Apache ส่วน Cloudera Enterprise นั้นจะมีการสนับสนุนทางด้านการผลิต และการออกแบบเครื่องมือที่ออกแบบมาเพื่อให้ง่ายต่อการใช้งานของ Hadoop ในระบบของการผลิต และนอกจากนี้ Cloudera ยังมีการให้บริการทางด้านการให้การสนับสนุน การบริการให้คำปรึกษาและการฝึกอบรมอีกด้วย แต่จะมีค่าใช้จ่ายในการใช้งาน

2.2 Hadoop Distributed File System (HDFS)

Hadoop Distributed File System (HDFS) คือระบบจัดเก็บข้อมูลหลักที่ใช้ในซอฟต์แวร์ Hadoop ซึ่ง HDFS จะสร้างแบบจำลองเป็นบล็อกข้อมูลบนคลัสเตอร์เพื่อให้เกิดความน่าเชื่อถือและการคำนวณผลที่รวดเร็ว ใช้งานในการเก็บข้อมูลขนาดใหญ่โดย HDFS นั้นเป็นซอฟต์แวร์ที่สามารถนำมาใช้งานได้โดยไม่เสียค่าใช้จ่ายใดๆ หลักการทำงานคือ HDFS จะรันอยู่บนระบบแฟ้มข้อมูลของระบบปฏิบัติการแบ่งข้อมูลขนาดใหญ่ที่ต้องการจะเก็บออกเป็นส่วนย่อยๆเก็บไฟล์เป็น Block แล้วกระจายข้อมูลส่วนย่อยๆ ดังภาพที่ 2-1 และจะทำสำเนาของแต่ละ Block เก็บไว้หลายที่เพื่อป้องกันการสูญหายของข้อมูล ดังภาพที่ 2-2



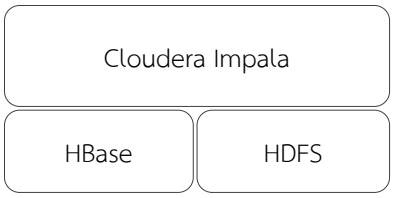
ภาพที่ 2-1 รูปแบบการแบ่งไฟล์เป็น Block ของ HDFS



ภาพที่ 2-2 การทำสำเนาข้อมูลในแต่ละ block เก็บไว้หลายที่เพื่อป้องกันการสูญหายของข้อมูล

2.3 Cloudera Impala

Cloudera Impala คือโปรแกรมระบบจัดการฐานข้อมูลที่รันอยู่บน Hadoop ซึ่ง Cloudera Impala จะใช้รูปแบบคำสั่งต่างๆเหมือน MySQL เช่น Select, Insert เป็นต้น และมีความเร็วในการค้นหาข้อมูล โดยการค้นหาข้อมูลของ Cloudera Impala นั้นจะค้นหาข้อมูลใน Hadoop Distributed File System (HDFS) หรือ Hbase ดังภาพที่ 2-3



ภาพที่ 2-3 ลักษณะการทำงานของ Cloudera Impala

2.3.1 คุณสมบัติหลักของ Cloudera Impala

2.3.1.1 ประมวลผลในแบบ Interactive ผ่าน SQL และ BI Application

2.3.1.2 Query Data แบบ Real-Time

2.3.1.3 Optimized เพื่องานคล้าย กับ Data warehouse สำหรับ Operational Analytics และ Reports.

2.3.1.4 Query Data ผ่าน Hbase หรือ HDFS

2.3.1.5 รูปแบบไฟล์ HDFS เช่น Text file, Sequence File, RCFile และ Parquet.

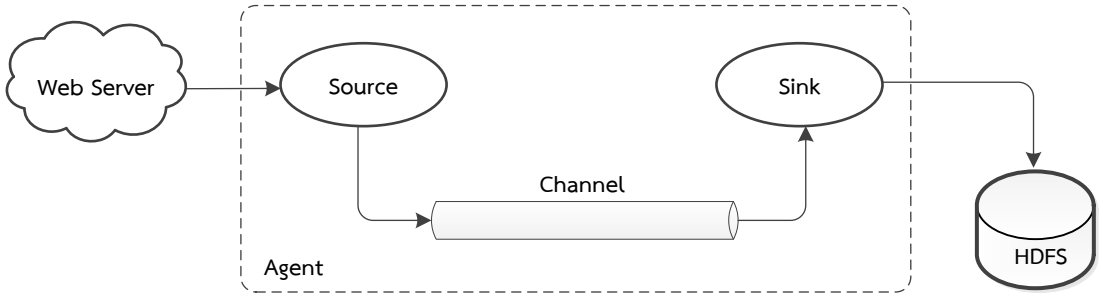
2.3.1.6 สามารถบีบอัดไฟล์เป็นแบบ Snappy, GZIP, Deflate และ BZIP

2.3.1.7 การติดต่อเพื่อใช้งานของ Impala ผ่านด้วย JDBC driver, ODBC driver และ

Hue WebGUI

2.4 Apache Flume

Apache Flume เป็นโปรแกรมส่งข้อมูลที่รันอยู่บน Hadoop ซึ่ง Flume นั้นจะทำหน้าที่คอยส่งข้อมูลจำนวนมากไปยังที่เก็บข้อมูล โดยโครงสร้างภายในของ Flume จะประกอบไปด้วย Source, Channel และ Sink ดังภาพที่ 2-4



ภาพที่ 2-4 รูปแบบการส่งข้อมูลของ Apache Flume

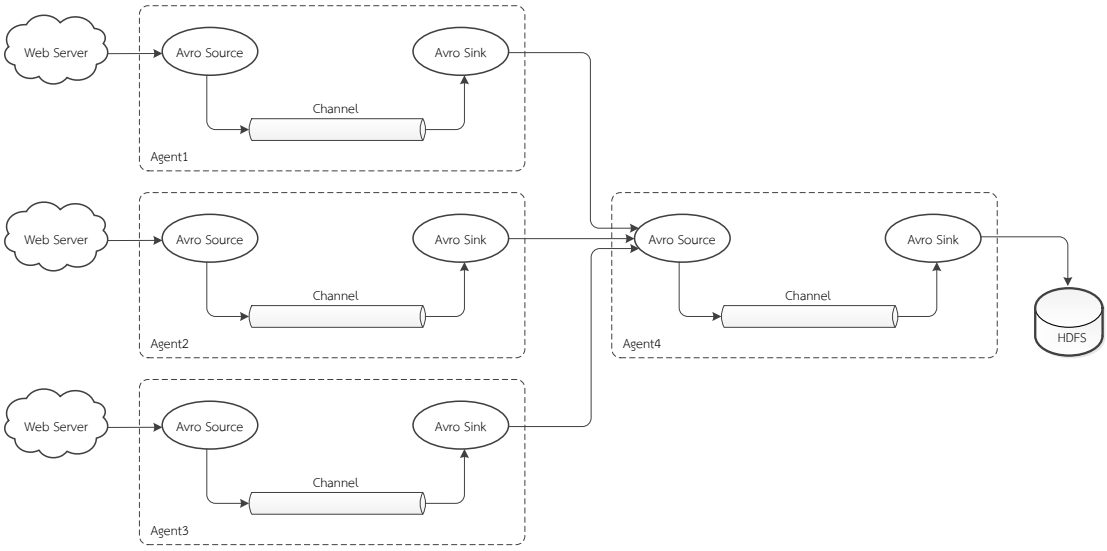
2.4.1 คุณสมบัติหลักของ Flume

2.4.1.1 Flume Sink เก็บข้อมูล ผ่าน Hadoop Distributed File System (HDFS) หรือ Hbase, Logger, Avro, Thrift และ IRC

2.4.1.2 Flume Source รวบรวมข้อมูลผ่าน Avro, Thrift, Exec, Http, syslog และ netcut

2.4.1.3 Flume Channel ช่องทางที่เก็บเมื่อมีการรวบรวมข้อมูลเก็บผ่าน Memory, JDBC และ File

ตัวอย่าง การเก็บรวบรวมข้อมูลจราจร จาก เครื่อง Web Server หลายๆเครื่องที่ส่งไปยัง Flume Agent อีกตัวหนึ่งเพื่อรวบรวมข้อมูลแล้วเขียนไปยัง Hadoop Distributed File System (HDFS) ดังภาพที่ 2-5

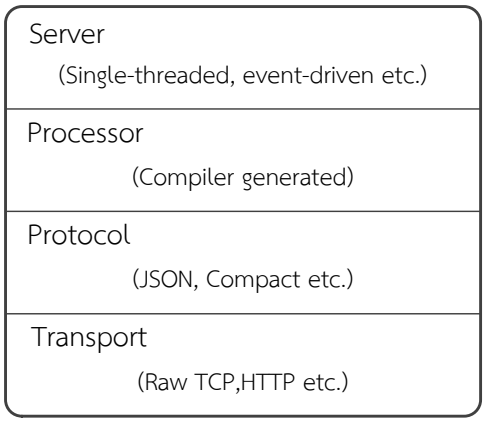


ภาพที่ 2-5 การเก็บรวบรวมข้อมูลจราจรจากเครื่อง Web Server หลายๆ เครื่อง

2.5 Apache Thrift

Apache Thrift เป็นสื่อกลางในการเชื่อมต่อระหว่างทางภาษาคอมพิวเตอร์ ใช้สำหรับสร้างความเข้าใจสำหรับภาษาคอมพิวเตอร์ต่าง ๆ ให้สามารถติดต่อสื่อสารกันระหว่างภาษาคอมพิวเตอร์ ที่ต่างกันได้และเข้าใจได้ตรงกันโดย Apache Thrift จะสนับสนุนภาษาคอมพิวเตอร์ได้แก่ C++, Java, Python, PHP, Ruby, Perl, C#, JavaScript, Delphi และภาษาอื่นๆ เป็นต้น

โครงสร้าง Stack เครือข่ายของ Apache Thrift จะประกอบไปด้วย Server layer, Protocol layer, Processor layer และ Transport layer ดังภาพที่ 2-6



ภาพที่ 2-6 โครงสร้าง Stack เครือข่ายของ Apache Thrift

ชั้น Transport ทำหน้าที่สำหรับอ่านและเขียนจากเครือข่ายหนึ่งไปยังอีกเครือข่ายหนึ่ง โดยชั้น Transport จะมี methods ซึ่งประกอบไปด้วย open, close, read, write นอกจากนี้ยังใช้ส่วนการติดต่อ Server Transport ใช้ในการยอมรับหรือสร้าง Objects ทำให้ Server Transport ส่วนใหญ่ใช้ฝั่งเซิร์ฟเวอร์ในการสร้าง Objects สำหรับการเชื่อมต่อเข้ามา โดย Objects ที่สร้างนั้นประกอบไป

ด้วยเมธอด open, listen, accept, close

ชั้น Protocol ทำหน้าที่กำหนดกลไกในการ map โครงสร้างข้อมูลในหน่วยความจำไปยังรูปแบบของสาย โดยโปรโตคอลจะระบุว่าประเภทข้อมูลของ Transport ในการเข้ารหัส/ถอดรหัส ดังนั้นภาษาโปรโตคอลที่สนับสนุนใน Apache Thrift คือ JSON, XML, Plain text, compact binary และอื่นๆ

ชั้น Processor ทำหน้าที่ ประมวลผลความสามารถในการอ่านข้อมูลจากสตรีมอินพุตและเขียนไปยังสตรีมเอาท์พุท

สตรีมอินพุทและเอาท์พุทเป็น Objects ของโปรโตคอลจึงทำให้ส่วนการติดต่อชั้นประมวลผลนั้นติดต่อได้ง่ายดังภาพที่ 2-7

```
interface TProcessor {
    bool process(TProtocol in, TProtocol out) throws TException
}
```

ภาพที่ 2-7 ส่วนติดต่อของชั้นประมวลผล

ชั้น Server ทำหน้าที่ สร้างชั้น Transport, สร้างชั้นโปรโตคอลอินพุต/เอาต์พุตสำหรับ Transport, สร้างชั้นประมวลผลบนพื้นฐานของโปรโตคอลอินพุต/เอาต์พุตและรอรับการเชื่อมต่อเข้าและส่งให้ออกไปยังชั้นประมวลผล

2.6 PHP

PHP ย่อมาจากคำว่า “PHP Hypertext Preprocessor” ใช้สำหรับจัดทำเว็บไซต์เป็น Server Side Script ที่มีการทำงานที่ฝั่งของเครื่องคอมพิวเตอร์ Server ซึ่งรูปแบบในการเขียนคำสั่งการทำงานนั้นจะมีลักษณะคล้ายกับภาษา Perl หรือภาษา C และแสดงผลออกมาในรูปแบบ HTML ได้อย่างมีประสิทธิภาพ ซึ่งจะทำให้รูปแบบเว็บเพจมีความสามารถเพิ่มขึ้นในด้านของการเขียนโปรแกรมในการสร้างเว็บจะใช้ Script อยู่ 2 แบบด้วยกันคือ

- Server-Side Script เป็นลักษณะของภาษาที่ทำงานบนเครื่อง Server เช่น CGI, ASP
- Client-Side Script เป็นลักษณะของภาษาที่ทำงานบนเครื่องผู้ใช้เช่น JavaScript, VBScript

ความสามารถของ PHP นั้น สามารถที่จะทำงานเกี่ยวกับ Dynamic Web ได้ทุกรูปแบบเหมือนกับ CGI หรือ ASP ไม่ว่าจะเป็นการดูแลจัดการระบบฐานข้อมูล ระบบรักษาความปลอดภัยของเว็บเพจการ รับ–ส่ง Cookies

ภาษา PHP จะเป็นส่วนประกอบภายในเว็บเพจโดยคำสั่งจะปรากฏระหว่าง <?php ... ?> ดังภาพที่ 2-8

```
<?php
    echo "Hello HDFS";
?>
```

ภาพที่ 2-8 ตัวอย่างภาษา PHP

การเขียน PHP แบบ OOP ดังภาพที่ 2-9

```php
<?php
    Class EasyClass{
        private $VarA = "Hello HDFS";
        public function getHello(){
            return $this->VarA;
        }
    }
    $object = new EasyClass();
    echo $object->getHello();
?>
```

ภาพที่ 2-9 ตัวอย่างภาษา PHP แบบ OOP

2.7 HTML

HTML ย่อมาจาก Hyper Text Markup Language คือภาษาคอมพิวเตอร์ที่ใช้ในการแสดงผลของเอกสารบน website หรือที่เรียกกันว่าเว็บเพจ ถูกพัฒนาและกำหนดมาตรฐานโดยองค์กร World Wide Web Consortium (W3C) และจากการพัฒนาทางด้าน Software ของ Microsoft ทำให้ภาษา HTML เป็นอีกภาษาหนึ่งที่ใช้เขียนโปรแกรมได้ หรือที่เรียกว่า HTML Application

HTML เป็นภาษาประเภท Markup สำหรับการการสร้างเว็บเพจ โดยใช้ภาษา HTML สามารถทำโดยใช้โปรแกรม Text Editor ต่างๆ เช่น Notepad, Editplus หรือจะอาศัยโปรแกรมที่เป็นเครื่องมือช่วยสร้างเว็บเพจ เช่น Coda(MAC OSX), Notepad++(Windows) ซึ่งอำนวยความสะดวกในการสร้างหน้า HTML ส่วนการเรียกใช้งานหรือทดสอบการทำงานของเอกสาร HTML จะใช้โปรแกรม web browser เช่น Microsoft Internet Explorer(IE), Mozilla Firefox, Safari, Opera, และ Google chrome เป็นต้น

คำสั่ง HTML แต่ละตัวนิยมเรียกกันว่า "แท็ก" ซึ่งแท็กทุกตัวจะมีเครื่องหมาย < และ > ปิดหัวท้าย ตัวอย่างเช่น
, <P>, <HEAD> ฯลฯ ลักษณะการทำงานของคำสั่ง HTML ส่วนใหญ่จะต้องมีการกำหนดจุดเริ่มต้นและจุดลงท้าย เช่น หากต้องการทำให้อักษรแสดงผลเป็นตัวหนาต้องใช้คำสั่ง <B> แล้วพิมพ์ข้อความที่ต้องการทำให้เป็นตัวหนา จากนั้นปิดคำสั่งด้วย </B> เช่น <B>Hello</B> เป็นต้น

2.8 JavaScript

JavaScript เป็นภาษาโปรแกรม (Programming Language) ประเภทหนึ่ง ที่เรียกกันว่า "สคริปต์" (Script) ซึ่งมีวิธีการทำงานในลักษณะ "แปลความและดำเนินงานไปทีละคำสั่ง" (Interpreter) ภาษานี้เดิมมีชื่อว่า Live Script ได้รับการพัฒนาขึ้นโดย Netscape ด้วยวัตถุประสงค์เพื่อที่จะช่วยให้เว็บเพจสามารถแสดงเนื้อหา ที่มีการเปลี่ยนแปลงไปได้ ตามเงื่อนไขหรือสภาพแวดล้อมต่างๆกัน หรือสามารถโต้ตอบกับผู้ชมได้มากขึ้น ทั้งนี้เพราะภาษา HTML แต่เดิมนั้นเหมาะสำหรับใช้แสดงเอกสารที่มีเนื้อหาคงที่แน่นอน และไม่มีลูกเล่นอะไรมากมายนักเนื่องจาก JavaScript ช่วยให้ผู้พัฒนา สามารถสร้างเว็บเพจได้ตรงกับความต้องการ และมีความน่าสนใจมากขึ้น

ประกอบกับเป็นภาษาเปิด ที่ใครก็สามารถนำไปใช้ได้ ดังนั้นจึงได้รับความนิยมเป็นอย่างสูง มีการใช้งานอย่างกว้างขวาง รวมทั้งได้ถูกกำหนดให้เป็นมาตรฐานโดย ECMA ซึ่งจะพบว่าปัจจุบัน จะหาเว็บเพจที่ไม่ใช้ JavaScript เลยนั้น ได้ยากเต็มที

การทำงานของ JavaScript จะต้องมีการแปลความคำสั่ง ซึ่งขั้นตอนนี้จะถูกจัดการโดยบราวเซอร์ ดังนั้น JavaScript จึงทำงานได้ เฉพาะบนบราวเซอร์ที่สนับสนุน ซึ่งปัจจุบันบราวเซอร์เกือบทั้งหมดก็สนับสนุน JavaScript แล้ว อย่างไรก็ดี สิ่งที่ต้องระวังคือ JavaScript มีการพัฒนาเป็นเวอร์ชันใหม่ๆออกมาด้วย ถ้านำโค้ดของเวอร์ชันใหม่ ไปรันบนเบราว์เซอร์รุ่นเก่าก็อาจจะทำให้เกิด Error ได้

2.9 CSS

CSS ย่อมาจาก Cascading Style Sheet มักเรียกโดยย่อว่า "สไตล์ชีต" คือภาษาที่ใช้เป็นส่วนของการจัดรูปแบบการแสดงผลเอกสาร HTML โดยที่ CSS กำหนดกฎเกณฑ์ในการระบุรูปแบบ (หรือ "Style") ของเนื้อหาในเอกสาร ได้แก่ สีของข้อความ สีพื้นหลัง ประเภทตัวอักษร และการจัดวางข้อความ ซึ่งการกำหนดรูปแบบ หรือ Style นี้ใช้หลักการของการแยกเนื้อหาเอกสาร HTML ออกจากคำสั่งที่ใช้ในการจัดรูปแบบการแสดงผล กำหนดให้รูปแบบของการแสดงผลเอกสาร ไม่ขึ้นอยู่กับเนื้อหาของเอกสาร เพื่อให้ง่ายต่อการจัดรูปแบบการแสดงผลลัพธ์ของเอกสาร HTML โดยเฉพาะในกรณีที่มีการเปลี่ยนแปลงเนื้อหาเอกสารบ่อยครั้ง หรือต้องการควบคุมให้รูปแบบการแสดงผลเอกสาร HTML มีลักษณะของความสม่ำเสมอทั่วกันทุกหน้าเอกสารภายในเว็บไซต์เดียวกัน โดยกฎเกณฑ์ในการกำหนดรูปแบบ (Style) เอกสาร HTML ถูกเพิ่มเข้ามาครั้งแรกใน HTML 4.0 เมื่อปีพ.ศ. 2539 ในรูปแบบของ CSS level 1 Recommendations ที่กำหนดโดย องค์กร W3C

ประโยชน์ของ CSS

2.9.1 CSS มีคุณสมบัติมากกว่า tag ของ html เช่น การกำหนดกรอบให้ข้อความ รวมทั้งสี รูปแบบของข้อความที่กล่าวมาแล้ว

2.9.2 CSS นั้นกำหนดที่ต้นของไฟล์ html หรือตำแหน่งอื่น ๆ ก็ได้ และสามารถมีผลกับเอกสารทั้งหมด หมายถึงกำหนดครั้งเดียวจุดเดียวก็มีผลกับการแสดงผลทั้งหมด ทำให้เวลาแก้ไขหรือปรับปรุงทำได้สะดวก ไม่ต้องตามแก้ tag ต่างๆ ทั่วทั้งเอกสาร

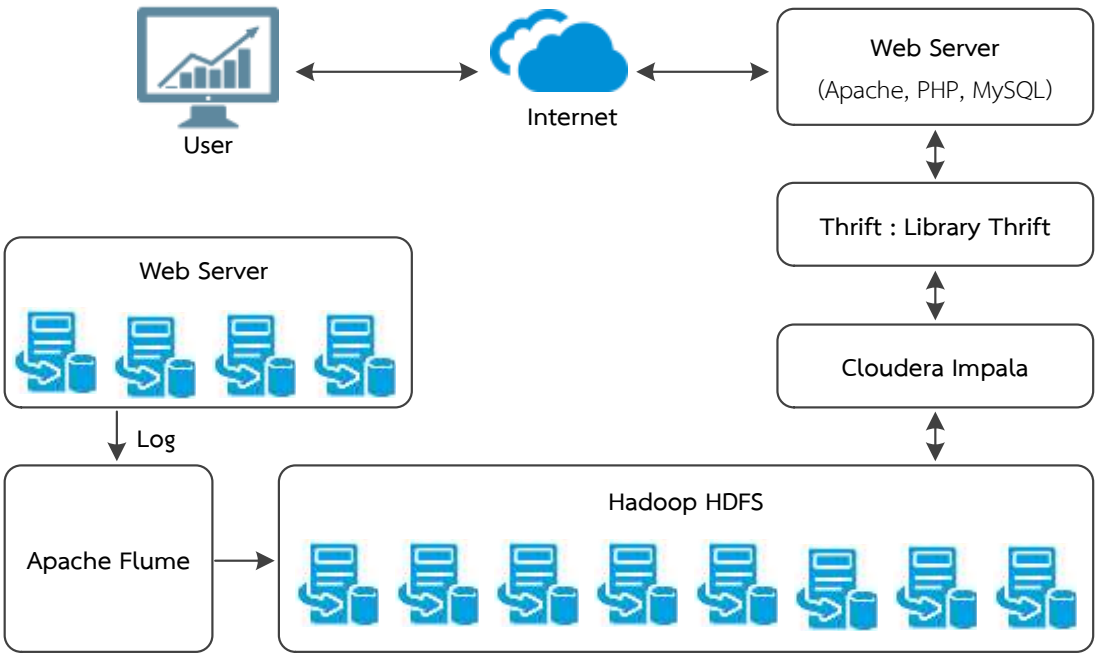
2.9.3 CSS สามารถกำหนดแยกไว้ต่างหากจากไฟล์เอกสาร html และสามารถนำมาใช้ร่วมกับเอกสารหลายไฟล์ได้ การแก้ไขก็แก้เพียงจุดเดียวก็มีผลกับเอกสารทั้งหมด

บทที่ 3
วิธีการดำเนินงาน

การดำเนินการโครงงานวิจัย “การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala (Design and Development of Data Traffic Analysis of Web Sever on HDFS System using Impala)” แบ่งออกเป็นส่วนต่างๆ ดังนี้

3.1 หลักการทำงานของโปรแกรม
3.2 การออกแบบและติดตั้งระบบเครือข่าย
3.3 การเชื่อมต่อ Web Server และ HDFS ด้วย Apache Flume
3.4 การจัดรูปแบบข้อมูลจราจร
3.5 การสร้างตาราง Cloudera Impala
3.6 การติดตั้งระบบเพิ่มเติม
3.7 การติดต่อ PHP กับ Cloudera Impala ผ่านทาง Thrift
3.8 การออกแบบส่วนติดต่อกับผู้ใช้งานบนเว็บ

3.1 หลักการทำงานของโปรแกรม



ภาพที่ 3-1 บล็อกไดอะแกรมของระบบ

จากภาพที่ 3-1 การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala (Design and Development of Data Traffic Analysis of Web Sever on HDFS System using Impala) นั้นจะนำข้อมูลจราจรของเครื่องแม่ข่ายเว็บมาวิเคราะห์ โดยมีโปรแกรม Apache Flume เป็นตัวกลางในการดึงข้อมูลจราจรของเครื่องแม่ข่ายเว็บ และแปลง

ให้อยู่ในรูปแบบที่เหมาะสมก่อนนำไปจัดเก็บใน Hadoop Distributed File System หลังจากนั้นใช้ Cloudera Impala ในการดึงข้อมูลจากฐานข้อมูล Hadoop Distributed File System เพื่อนำข้อมูลจราจรมาวิเคราะห์โดยระบบนี้จะใช้ Thrift ในการติดต่อกับเว็บไซต์ที่พัฒนาด้วยภาษา PHP โดยนำข้อมูลจราจรที่ได้วิเคราะห์แล้วมาแสดงผลในรูปแบบของกราฟบนเว็บไซต์

ส่วนประกอบต่างๆของระบบประกอบด้วย Hadoop Distributed File System, Cloudera Impala, Apache Flume และ Thrift ซึ่งส่วนประกอบต่างๆ มีหน้าที่ดังนี้

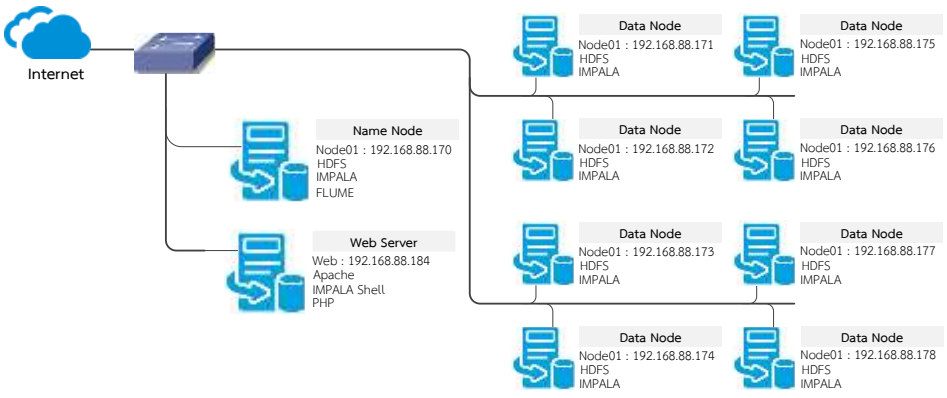
3.1.1 Apache Flume ทำหน้าที่ในการดักจับข้อมูลจราจรทางเว็บไซต์ล่าสุดของข้อมูลใน log file ซึ่งเมื่อมีข้อมูลจราจรทางเว็บไซต์ใหม่เกิดขึ้นจะทำการดึงข้อมูลจราจรนั้นจาก log file ของ Web Server เพื่อส่งต่อไปยัง Hadoop Distributed File System

3.1.2 Hadoop Distributed File System เมื่อมีข้อมูลจราจรส่งเข้ามาจาก Apache Flume ที่ส่วนของ Web Server แล้ว Apache Flume ที่ Hadoop จะส่งข้อมูลต่อไปยัง HDFS เพื่อทำหน้าที่ในการแบ่งข้อมูลออกเป็นส่วนย่อยๆ แล้วกระจายข้อมูลไปเก็บยัง DataNode

3.1.3 Cloudera Impala เป็นตัวจัดการข้อมูลในรูปแบบของตารางรองรับคำสั่งในรูปแบบ SQL ซึ่งทำหน้าที่ในการค้นหาข้อมูล log file ที่ถูกแบ่งออกเป็นส่วนย่อยๆใน HDFS

3.1.4 Thrift ทำหน้าที่เป็นสื่อกลางในการเชื่อมต่อกันระหว่าง PHP กับ Cloudera Impala โดยการแปลภาษาให้สามารถติดต่อสื่อสารกันได้เนื่องด้วย Cloudera Impala นั้นปกติจะสามารถติดต่อไปด้วย Java เพียงอย่างเท่านั้น

3.2 การออกแบบและติดตั้งระบบเครือข่าย



ภาพที่ 3-2 แผนผังการทำงานระหว่าง NameNode กับ DataNode

การวิจัยนี้ใช้เครื่องคอมพิวเตอร์ทั้งหมด 10 เครื่องเชื่อมต่อกันผ่านระบบเครือข่ายปิด ความเร็ว 1 Gbps ดังภาพที่ 3-2 ซึ่งมีรายละเอียดดังนี้

3.2.1 โหนดหลัก (NameNode)

NameNode มีจำนวน 1 เครื่อง ซึ่งจะจัดการเกี่ยวกับ Namespace ของระบบแฟ้มข้อมูล เช่น เปิด ปิด เปลี่ยนชื่อ แฟ้มข้อมูลและไดเร็กทอรี และยังทำหน้าที่ในการจดจำบล็อกที่เก็บข้อมูลว่าอยู่ที่ DataNode ตัวไหน

โฮสเนมและหมายเลขไอพีแอดเดรสของเครื่องโหนดหลัก (NameNode) คือ

namenode01.hadoop.kmutnb.ac.th(192.168.88.170)

Software ของเครื่องโหนดหลัก (NameNode) คือ

- HDFS
- Flume
- Impala

3.2.2 โหนดข้อมูล (DataNode)

DataNode มีจำนวน 8 เครื่อง หน้าที่หลักของ DataNode ก็คือจัดการเกี่ยวกับเนื้อที่ในการเก็บข้อมูลของเครื่องที่เรียกใช้ DataNode ขึ้นมาทำงานและรับผิดชอบในการบริการการเขียนและอ่านข้อมูลตามคำขอของผู้ใช้งาน รวมทั้งมีหน้าที่สร้าง ลบ และทำสำเนาบล็อกตามคำสั่งของ NameNode

โฮสเนมและหมายเลขไอพีแอดเดรสของเครื่องโหนดข้อมูล (DataNode) คือ

- node01.hadoop.kmutnb.ac.th (192.168.88.171)
- node02.hadoop.kmutnb.ac.th (192.168.88.172)
- node03.hadoop.kmutnb.ac.th (192.168.88.173)
- node04.hadoop.kmutnb.ac.th (192.168.88.174)
- node05.hadoop.kmutnb.ac.th (192.168.88.175)
- node06.hadoop.kmutnb.ac.th (192.168.88.176)
- node07.hadoop.kmutnb.ac.th (192.168.88.177)
- node08.hadoop.kmutnb.ac.th (192.168.88.178)

Software ของเครื่องโหนดข้อมูล (DataNode) คือ

- HDFS
- Impala

การทำงานภายในของระบบ HDFS แฟ้มข้อมูลที่จะถูกจัดเก็บถูกแบ่งออกเป็นบล็อกๆ และจะถูกนำไปเก็บแบบกระจายไปยัง DataNode ต่างๆ

การที่มี NameNode ในระบบเพียงเครื่องเดียวนั้นทำให้ การออกแบบและใช้งานระบบง่ายขึ้น โดยที่ข้อมูลต่างๆ เกี่ยวกับแฟ้มข้อมูลจะถูกเก็บไว้ที่ NameNode แต่เนื้อหาข้อมูลทั้งหมดจะถูกเก็บไว้ที่ DataNode ทำให้ข้อมูลที่วิ่งผ่านระบบจะไม่ผ่าน NameNode เลย

3.2.3 เว็บเซิร์ฟเวอร์ (Web Server)

เว็บเซิร์ฟเวอร์มีจำนวน 1 เครื่อง มีหน้าที่หลักคือ ให้บริการเว็บไซต์สำหรับดูข้อมูลในรูปแบบกราฟและข้อความต่างๆ

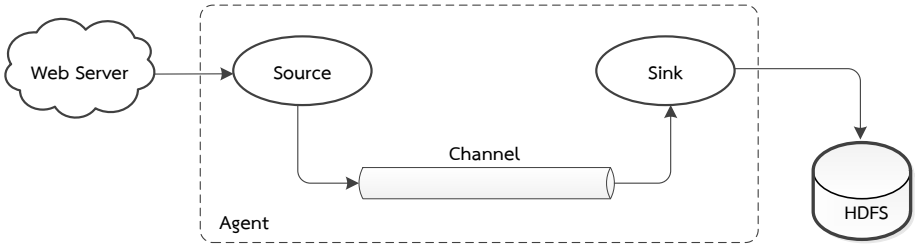
โฮสเนมและหมายเลขไอพีแอดเดรสของเครื่องเว็บเซิร์ฟเวอร์ (Web Server) คือ

- wh.hadoop.kmutnb.ac.th (192.168.88.184)

Software ของเครื่องเว็บเซิร์ฟเวอร์ (Web Server) คือ

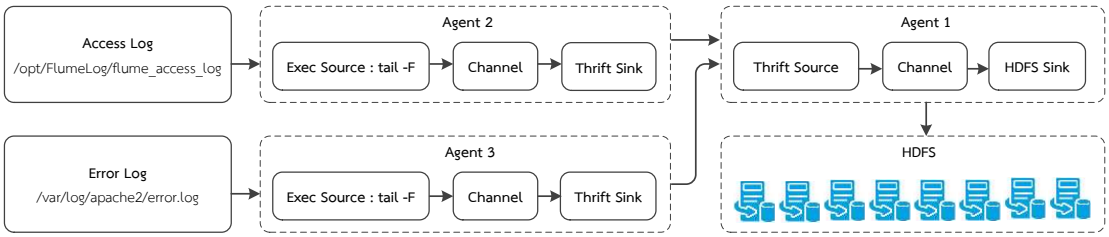
- Impala Shell
- Apache
- PHP & MySQL Server

3.3 การเชื่อมต่อระหว่าง Web Server และ HDFS ด้วย Apache Flume



ภาพที่ 3-3 การติดต่อระหว่าง Web Server กับ HDFS ด้วย Apache Flume

จากภาพที่ 3-3 เป็นการแสดงให้เห็นว่าการเชื่อมต่อระหว่าง Web Server และ HDFS ด้วย Apache Flume นั้น มี 2 ส่วนในการเชื่อมต่อ คือ ฝั่ง Web Server และ ฝั่ง Hadoop Distributed File System โดย Agent ของ Apache Flume นั้นจะประกอบด้วย 3 ส่วนคือ Source, Channel, Sink ดังแสดงในภาพที่ 3-4



ภาพที่ 3-4 Agent1 HDFS และ Agent2, Agent3 ในฝั่ง Web Server

จากภาพที่ 3-4 แสดงการทำงานของ Apache Flume ซึ่งจะเห็นว่า Agent1 จะอยู่ในฝั่ง Hadoop Distributed File System (HDFS) โดย Source ของ Agent1 นั้นจะเป็นการเชื่อมต่อแบบ Thrift และ Sink ของ Agent1 จะ Sink ไปยัง Hadoop Distributed File System (HDFS) ในการ Config Agent1 ของ Apache Flume นั้นจะ Config ตามภาพที่ 3-5 ส่วน Agent2, Agent3 จะอยู่ฝั่ง Web Server โดย Source ของ Agent3 นั้นจะค่อยรับคำสั่ง tail –f /opt/FlumeLog /flume_access_log และ Agent2 นั้นจะค่อยรับคำสั่ง tail –f /var/log/apache2/access.log และ Sink ของ Agent2, Agent3 จะ Sink ไปยัง Source ของ Agent1 ในการ Config Agent2, Agent3 ของ Apache Flume นั้นจะ Config ตามภาพที่ 3-6 และภาพที่ 3-7 ตามลำดับ

```
agent1.sources = thrift thrift2
agent1.channels = memoryChannel1 memoryChannel2
agent1.sinks = sink1 sink2

agent1.sources.thrift.type  = thrift
agent1.sources.thrift.bind = 0.0.0.0
agent1.sources.thrift.port = 44444
agent1.sources.thrift.interceptors = i1
agent1.sources.thrift.interceptors.i1.type = timestamp
agent1.sources.thrift.channels = memoryChannel1

agent1.sources.thrift2.type  = thrift
agent1.sources.thrift2.bind = 0.0.0.0
agent1.sources.thrift2.port = 55555
agent1.sources.thrift2.interceptors = i2
agent1.sources.thrift2.interceptors.i2.type = timestamp
agent1.sources.thrift2.channels = memoryChannel2

agent1.sinks.sink1.channel = memoryChannel1
agent1.sinks.sink1.type = hdfs
agent1.sinks.sink1.hdfs.path = hdfs://192.168.88.170:8020/tmp/accesslog
agent1.sinks.sink1.hdfs.fileType  = DataStream
agent1.sinks.sink1.hdfs.writeFormat = Text
agent1.sinks.sink1.hdfs.filePrefix = %Y-%m-%d-
agent1.sinks.sink1.hdfs.rollInterval = 600
agent1.sinks.sink1.hdfs.rollCount = 1000
agent1.sinks.sink1.hdfs.rollSize = 0

agent1.sinks.sink2.channel = memoryChannel2
agent1.sinks.sink2.type = hdfs
agent1.sinks.sink2.hdfs.path = hdfs://192.168.88.170:8020/tmp/errorlog
agent1.sinks.sink2.hdfs.fileType  = DataStream
agent1.sinks.sink2.hdfs.writeFormat = Text
agent1.sinks.sink2.hdfs.filePrefix = %Y-%m-%d-
agent1.sinks.sink2.hdfs.rollInterval = 600
agent1.sinks.sink2.hdfs.rollCount = 1000
agent1.sinks.sink2.hdfs.rollSize = 0

agent1.channels.memoryChannel1.type = memory
agent1.channels.memoryChannel1.capacity = 1000
agent1.channels.memoryChannel1.transactionCapacity = 100
agent1.channels.memoryChannel2.type = memory
agent1.channels.memoryChannel2.capacity = 1000
agent1.channels.memoryChannel2.transactionCapacity = 100
```

ภาพที่ 3-5 ตัวอย่างในการ config flume ฝั่ง Hadoop Distributed File System (HDFS)

```
#--------------------------- Error Log -------------------------
agent3.sources = tail
agent3.channels = memoryChannel2
agent3.sinks = thriftSink

agent3.sources.tail.type = exec
agent3.sources.tail.command = tail -f /var/log/apache2/error.log
agent3.sources.tail.channels = memoryChannel2

agent3.sinks.thriftSink.channel = memoryChannel2
agent3.sinks.thriftSink.type = thrift

agent3.sinks.thriftSink.hostname = 192.168.88.170
agent3.sinks.thriftSink.port = 55555

agent3.channels.memoryChannel2.type = memory
agent3.channels.memoryChannel2.capacity = 1000
```

ภาพที่ 3-6 ตัวอย่างในการ config flume ฝั่ง Web Server ในส่วนของ Error log

```
#--------------------------- Access Log  -------------------------
agent2.sources = tail
agent2.channels = memoryChannel1
agent2.sinks = thriftSink

agent2.sources.tail.type = exec
agent2.sources.tail.command = tail -f /opt/FlumeLog/flume_access_log
agent2.sources.tail.channels = memoryChannel1

agent2.sinks.thriftSink.channel = memoryChannel1
agent2.sinks.thriftSink.type = thrift

agent2.sinks.thriftSink.hostname = 192.168.88.170
agent2.sinks.thriftSink.port = 44444

agent2.channels.memoryChannel1.type = memory
agent2.channels.memoryChannel1.capacity = 1000
```

ภาพที่ 3-7 ตัวอย่างในการ config flume ฝั่ง Web Server ในส่วนของ Access log

```
2015-04-26 21:35:24,489 INFO org.apache.flume.sink.hdfs.HDFSDataStream: Serializer = TEXT,
UseRawLocalFileSystem = false
2015-04-26 21:35:24,548 INFO org.apache.flume.sink.hdfs.BucketWriter: Creating
hdfs://192.168.88.170:8020/tmp/accesslog/2015-04-26-.1430109324490.tmp
```

ภาพที่ 3-8 ตัวอย่าง Log ของ Apache flume เมื่อมีการเขียนข้อมูลจราจร

จากภาพที่ 3-8 เมื่อข้อมูลจราจรมีการเปลี่ยนแปลงนั้น Apache Flume จะทำการสร้างไฟล์ .tmp เพื่อรอเขียนข้อมูลลง Hadoop Distributed File System (HDFS) จนกว่าจะครบ 10 นาที ซึ่งในไฟล์ Config Agent1 ของ Apache Flume ได้ ทำการ Config agent1 sinks.sink1.hdfs.-rollInterval = 600 หลังจากนั้นเมื่อเวลาครบ 10 นาทีแล้ว Apache Flume จะทำการเขียนข้อมูลข้อมูลลง Hadoop Distributed File System (HDFS) ดังภาพที่ 3-9

```
2015-04-27 18:27:37,528 INFO org.apache.flume.sink.hdfs.BucketWriter: Renaming
hdfs://192.168.88.170:8020/tmp/accesslog/2015-04-27-.1430183857030.tmp to
hdfs://192.168.88.170:8020/tmp/accesslog/2015-04-27-.1430183857030
2015-04-27 18:27:37,560 INFO org.apache.flume.sink.hdfs.HDFSEventSink: Writer callback called.
```

ภาพที่ 3-9 ตัวอย่าง Log ของ Apache flume ที่เขียนข้อมูลลงใน HDFS เมื่อครบ 10 นาทีแล้ว

3.4 การจัดรูปแบบข้อมูลจราจร

การจัดรูปแบบข้อมูลจราจรนั้นจัดเพื่อให้ได้ข้อมูลจราจรของ access.log ที่เหมาะสมกับการวิเคราะห์ โดยแก้ไขไฟล์ /etc/apache2/apache2.conf ดังภาพที่ 3-10

```
LogFormat "%V`%h`%{%Y-%m-%d %H:%M:%S}t`%r`%D`%{User-agent}i" flume
```

ภาพที่ 3-10 รูปแบบข้อมูลจราจรของ access.log ที่ต้องการ

```
10.10.55.249`10.10.55.222`2014-03-23 18:42:04`GET /5555.html HTTP/1.1`489`Mozilla/5.0
(Windows NT 6.1; WOW64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/31.0.1650.63 Safari/537.36
```

ภาพที่ 3-11 ผลลัพธ์ของรูปแบบของข้อมูลจราจรของ access log

จากภาพที่ 3-11 ประกอบรายละเอียดด้วยส่วนต่างๆดังนี้

- %V คือหมายเลข IP หรือ Hostname ของเครื่องแม่ข่ายเว็บ
- %h คือหมายเลข IP ของเครื่องคอมพิวเตอร์ปลายทางที่เข้าชมเว็บ
- %{%Y-%m-%d %H%M%S}t คือวัน-เดือน-ปี ชั่วโมง:นาที:วินาที ที่บันทึกข้อมูลจราจร
- %r คือเว็บไซต์ที่ถูกร้องขอ
- %D คือเวลาที่ใช้ในการตอบสนองของหน้าเว็บไซต์
- %{User-agent}i คือโปรแกรมที่ใช้ในการเชื่อมต่อเว็บไซต์

ส่วนการจัดรูปแบบข้อมูลจราจรของ error log นั้นจัดเพื่อให้ได้ข้อมูลจราจรที่เหมาะสมกับการวิเคราะห์ โดยเพิ่มในส่วนของ ErrorLogFormat ดังภาพที่ 3-12

```
ErrorLogFormat "%V`%{c}t`%a`%M"
```

ภาพที่ 3-12 รูปแบบข้อมูลจราจรของ errorlog ที่ต้องการ

```
10.10.55.142`2014-03-24 17:55:02`10.10.55.222:52933`script '/var/www/index3.php'
not found or unable to stat
```

ภาพที่ 3-13 ผลลัพธ์ของรูปแบบของข้อมูลจราจรของ errorlog

จากภาพที่ 3-13 ประกอบรายละเอียดด้วยส่วนต่างๆดังนี้

- %V คือหมายเลข IP หรือ Hostname ของเครื่องที่เก็บแฟ้มข้อมูลจราจร
- %{c}t คือวัน-เดือน-ปี ที่บันทึกข้อมูลจราจร
- %a คือหมายเลข IP ของเครื่องคอมพิวเตอร์ปลายทางที่เชื่อมต่อด้วย
- %M คือเว็บไซต์ที่ถูกร้องขอ

3.5 การสร้างตารางของ Cloudera Impala

ในโครงงานปริญญานิพนธ์ได้มีการจัดเก็บข้อมูลการจราจรทางเว็บไซต์อยู่ 2 แบบคือ Acess log และ Error log ดังนั้นเพื่อให้การสืบค้นข้อมูลสะดวกขึ้นจึงจำเป็นต้องสร้างตารางที่สัมพันธ์กับข้อมูลเหล่านี้ด้วย Cloudera Impala

3.4.1 ตารางข้อมูลการจราจร Acess log

```
(1)  CREATE TABLE textaccesslog
     (
       host STRING,
       ip STRING,
       date1 STRING,
       website STRING,
       timeweb STRING,
       webrowser STRING
     )
(2)  ROW FORMAT DELIMITED FIELDS TERMINATED BY ' '
(3)  STORED AS TEXTFILE
(4)  LOCATION '/tmp/accesslog';
```

ภาพที่ 3-14 คำสั่งในการสร้างตาราง Access log ใน Cloudera Impala แบบไฟล์ Text

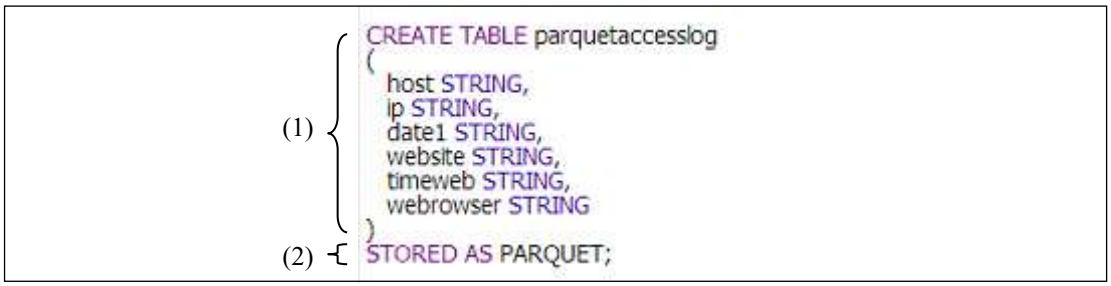
จากภาพที่ 3-14 เป็นการสร้างตารางใน Cloudera Impala แบบไฟล์ Text โดยมีรูปแบบของคำสั่งต่าง ๆดังนี้

(1) คำสั่งในการสร้างตารางและ field

(2) คำสั่งที่ใช้ในการตัดข้อความ

(3) คำสั่งจัดเก็บข้อมูลในรูปแบบ Text

(4) คำสั่งระบุไดเร็กทอรีที่เก็บไฟล์



ภาพที่ 3-15 คำสั่งในการสร้างตาราง Access log ใน Cloudera Impala แบบ Parquet

จากภาพที่ 3-15 เป็นการสร้างตารางใน Cloudera Impala แบบ Parquet โดยมีรูปแบบของคำสั่งต่าง ๆดังนี้

(1) คำสั่งในการสร้างตารางและ field

(2) คำสั่งจัดเก็บข้อมูลในรูปแบบ Parquet

ซึ่งข้อมูลแบบ Parquet นั้นจะมีการเก็บข้อมูลในรูปแบบโครงสร้างฐานข้อมูลทำให้มีการสืบค้นที่รวดเร็วกว่าแบบไฟล์ Text แต่ข้อเสียของข้อมูลแบบ Parquet คือไม่สามารถอ่านข้อมูลโดยตรงจากไฟล์ใน HDFS ได้ จำเป็นจะต้องเพิ่มข้อมูลจากตารางอื่นๆใน Cloudera Impala ดังนั้นข้อมูลใหม่ๆ จะเก็บไว้ในตาราง textaccesslog เมื่อถึงเวลาประมาณเที่ยงคืนทุกวันจะมีการนำเอาข้อมูลที่อยู่ในตาราง textaccesslog ทั้งหมดไปใส่ไว้ในตาราง parquetaccesslog แล้วลบข้อมูลออกจากตาราง textaccesslog ดังนั้นเมื่อใช้โปรแกรมที่เชื่อมต่อเพื่อสืบค้นข้อมูลไม่ต้องแก้ไขไปมาจึงสร้างการเชื่อมต่อกับ View ที่สร้างขึ้นด้วยคำสั่งดังภาพที่ 3-16

```
CREATE  VIEW accesslog as select * from parquetaccesslog union all select * from textaccesslog
```

ภาพที่ 3-16 คำสั่งในการสร้างตาราง Access log ใน Cloudera Impala แบบ View

จากภาพที่ 3-16 เป็นการสร้างตารางแบบ View โดยนำเอาตาราง parquetaccesslog มา union กับตาราง textaccesslog

ข้อมูลจราจรเข้าเว็บไซต์ที่สำเร็จจะถูกเก็บข้อมูลไว้ในตารางชื่อ accesslog มีโครงสร้างดังตารางที่ 3-1

ตารางที่ 3-1 รายละเอียดของการเก็บข้อมูลจราจรที่เข้าหน้าเว็บที่สำเร็จ

Field	Type	Null	Key	Default	Extra
host	String	No		Null	ชื่อ host ของเครื่องที่เก็บข้อมูล log
ip	String	No		Null	หมายเลข IP
date1	String	No		Null	วันที่ทำการจัดเก็บ
website	String	No		Null	เว็บไซต์ที่เข้าใช้งาน
timeweb	String	No		Null	เวลาที่ใช้ในการเข้าเว็บไซต์
webbrowser	String	No		Null	Browser ที่ใช้งาน

3.4.2 ตารางข้อมูลการจราจร Errorlog

ภาพที่ 3-17 คำสั่งในการสร้างตาราง Error log ใน Cloudera Impala แบบไฟล์ Text

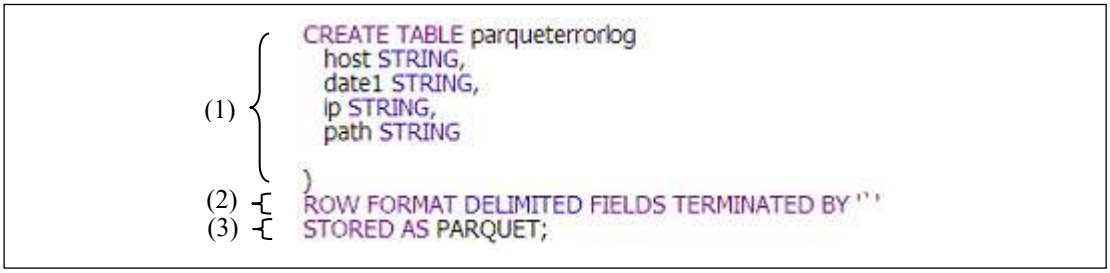
จากภาพที่ 3-17 เป็นการสร้างตารางใน Cloudera Impala แบบไฟล์ Text โดยมีรูปแบบของคำสั่งต่าง ๆดังนี้

(1) คำสั่งในการสร้างตารางและ field

(2) คำสั่งที่ใช้ในการตัดข้อความ

(3) คำสั่งจัดเก็บข้อมูลในรูปแบบ Text File

(4) คำสั่งระบุที่เก็บ files



ภาพที่ 3-18 คำสั่งในการสร้างตาราง Error log ใน Cloudera Impala แบบ Parquet

จากภาพที่ 3-18 เป็นการสร้างตารางใน Cloudera Impala แบบ Parquet โดยมีรูปแบบของคำสั่งต่าง ๆดังนี้

(1) คำสั่งในการสร้างตารางและ field
(2) คำสั่งที่ใช้ในการตัดข้อความ
(3) คำสั่งจัดเก็บข้อมูลในรูปแบบ parquet

```
CREATE  VIEW errorlog as select * from parqueterrorlog union all select * from texterrorlog
```

ภาพที่ 3-19 คำสั่งในการสร้าง View ของ Error log ใน Cloudera Impala

จากภาพที่ 3-19 เป็นการสร้างตารางแบบ View โดยนำเอา ตาราง parqueterrorlog มา union กับ ตาราง texterrorlog

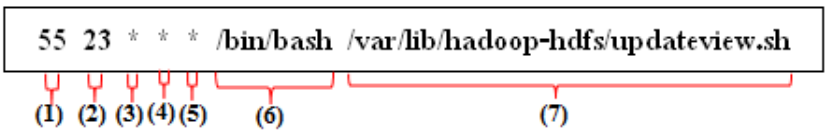
ข้อมูลจราจรเข้าเว็บไซต์ที่ไม่สำเร็จจะถูกเก็บข้อมูลไว้ในตารางชื่อ errorlog มีโครงสร้างดังตารางที่ 3-2

ตารางที่ 3-2 รายละเอียดของการเก็บข้อมูลจราจรที่เข้าหน้าเว็บไม่สำเร็จ

Field	Type	Null	Key	Default	Extra
host	String	No		Null	ชื่อ host ของเครื่องที่เก็บข้อมูล log
ip	String	No		Null	หมายเลข IP
Date1	String	No		Null	วันที่ทำการจัดเก็บ
Path	String	No		Null	เก็บ Path ของเว็บไซต์ที่เกิดข้อผิดพลาด

3.6 การติดตั้งระบบเพิ่มเติม

การติดตั้งระบบเพิ่มนี้หลักการทำงานคือทุกวัน ทุกเดือน ทุกสัปดาห์ เวลา 23 นาฬิกา 55 นาที ระบบจะนำข้อมูลจากตาราง textaccesslog เพิ่มใส่ในตาราง parquetaccesslog และจะนำข้อมูลจากตาราง texterrorlog เพิ่มใส่ในตาราง parqueterrorlog แล้วหลังจากนั้นไปลบข้อมูลในไดเร็กทอรีของ accesslog กับ errorlog ใน HDFS

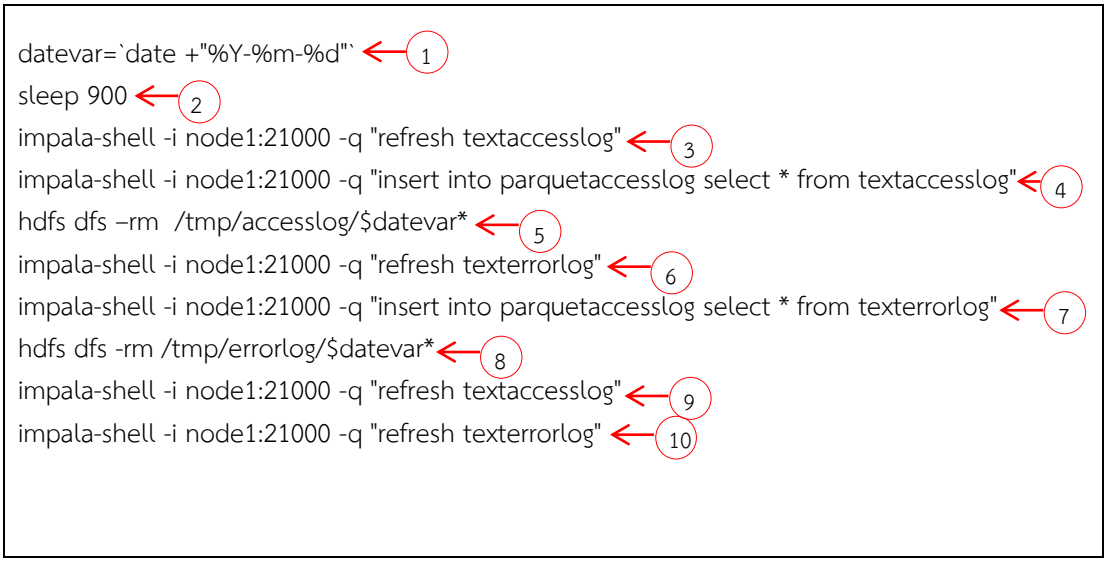


ภาพที่ 3-20 ตัวอย่างการ Config crontab

จากภาพที่ 3-20 เป็นการ Config crontab ใน linux โดยใช้คำสั่ง crontab –e เพื่อเข้าไปแก้ไข crontab แล้วเพิ่มคำสั่ง 55 23 * * * /bin/bash /var/lib/hadoop-hdfs/updateview.sh คำสั่งนี้สามารถอธิบายได้ดังนี้

1. 55 คือ 55 นาที
2. 23 คือ 23 ชั่วโมง

3. * คือ ทุกวัน
4. * คือ ทุกเดือน
5. * คือ ทุกสัปดาห์
6. /bin/bash คือ ประเภทคำสั่ง
7. /var/lib/hadoop-hdfs/updateview.sh คือ คำสั่งในการรัน



ภาพที่ 3-21 Shell script ใช้ในการแทรกข้อมูลเข้าจากตารางแบบไฟล์ Text ไปยังตารางแบบ Parquet

จากภาพที่ 3-21 เป็นการสร้างไฟล์ Shell script โดยภายในคำสั่งไฟล์ Shell script สามารถอธิบายได้ดังนี้

1. เก็บเวลาไว้ในตัวแปร datevar
2. หลับรอ 15 นาที
3. refresh ตาราง textaccesslog โดยผ่าน impala-shell
4. นำข้อมูลทั้งหมดของตาราง textaccesslog ไปใส่ในตาราง parquetaccesslog โดยผ่าน impala-shell
5. ลบข้อมูล HDFS ในfolder /tmp/accesslog/ ที่ขึ้นต้นตามค่าตัวแปร datevar ทั้งหมด
6. refresh ตาราง textaccesslog โดยผ่าน impala-shell
7. นำข้อมูลทั้งหมดของตาราง texterrorlog ไปใส่ใน ตาราง parqueterrorlog โดย impala-shell
8. ลบข้อมูล HDFS ในfolder /tmp/errorlog/ ที่ขึ้นต้นตามค่าของตัวแปร datevar ทั้งหมด
9. refresh ตาราง textaccesslog โดยผ่าน impala-shell
10. refresh ตาราง texterrorlog โดยผ่าน impala-shell

3.7 การติดต่อ PHP กับ Cloudera Impala ผ่านทาง Thrift

ภาพที่ 3-22 บล็อกไดอะแกรม PHP ติดต่อกับ Cloudera Impala

จากภาพที่ 3-22 จะได้เห็นได้ว่าการติดต่อกันระหว่าง PHP กับ Cloudera Impala นั้นไม่สามารถติดต่อกันได้โดยตรง ซึ่งหากต้องการให้ PHP สามารถติดต่อกับ Cloudera Impala ได้นั้น จำเป็นต้องอาศัย Thrift เป็นตัวกลางในการทำหน้าที่ Compile แปลงภาษาให้สามารถติดต่อกันได้ โดยเขียนคำสั่งในการติดต่อกันดังภาพที่ 3-23

```php
<?php
use Thrift\Transport\TBufferedTransport;
use Thrift\Transport\TSocket;
use Thrift\Protocol\TBinaryProtocol;

// include the phar
require_once 'build/php-impala.phar';

// grab the classmap from the phar put in globals
$GLOBALS['PHP_IMPALA_CLASSMAP'] = require_once 'phar://php-impala.phar/autoload_classmap.php';

// build a nice little autoloader using the classmap
spl_autoload_register(function ($class) {
    if(array_key_exists($class, $GLOBALS['PHP_IMPALA_CLASSMAP'])) {
        require $GLOBALS['PHP_IMPALA_CLASSMAP'][$class];
    }
});

// now to access the impala service
$socket = new TSocket('10.10.55.101', 21000); // make sure to enter your impala host ip address
$transport = new TBufferedTransport($socket);
$transport->open();
$protocol = new TBinaryProtocol($transport);
$client = new ImpalaServiceClient($protocol);
shell_exec('/usr/bin/impala-shell -q "invalidate metadata apachelog"');
?>
```

ภาพที่ 3-23 Code PHP สำหรับติดต่อกับ Cloudera Impala โดยผ่าน Thrift

จากภาพที่ 3-23 Code PHP สำหรับติดต่อกับ Cloudera Impala โดยผ่าน Thrift อธิบายได้ดังนี้

บรรทัดที่ 2 การเรียกใช้ ไฟล์ Thrift\Transport\TBufferedTransport

บรรทัดที่ 3 การเรียกใช้ ไฟล์ Thrift\Transport\TSocket

บรรทัดที่ 4 การเรียกใช้ ไฟล์ Thrift\Protocol\TBinaryProtocol

บรรทัดที่ 7 require_once เป็นฟังก์ชันในการแทรกไฟล์แบบไม่แทรกไฟล์เดิมซ้ำ โดยฟังก์ชันนี้จะเรียกไฟล์ ชื่อ build/php-impala.phar

บรรทัดที่ 10 ตัวแปร $GLOBALS['PHP_IMPALA_CLASSMAP'] มีค่าเท่ากับ require_once 'phar://php-impala.phar/autoload_classmap.php'

บรรทัดที่ 20 เป็นการสร้าง Object ให้กับตัวแปร $socket โดยใช้ TSocket ('10.10.55.101',21000) เพื่อใช้งาน

บรรทัดที่ 21 เป็นการสร้าง Object ให้กับตัวแปร $transport โดยใช้ TBuffered Transport ($socket) เพื่อใช้งาน

บรรทัดที่ 22 เปิดการเชื่อมต่อ โดยใช้ $transport->open() เพื่อเปิดใช้งาน

บรรทัดที่ 23 เป็นการสร้าง Object ให้กับตัวแปร $protocol โดยใช้ TBinaryProtocol($transport) เพื่อใช้งาน

บรรทัดที่ 24 เป็นการสร้าง Object ให้กับตัวแปร $client โดยใช้ ImpalaServiceClient($protocol)เพื่อใช้งาน

บรรทัดที่ 25 ใช้คำสั่ง linux ผ่าน php จะทำให้ Cloudera Impala การ Refresh ข้อมูลทุกครั้ง

```
1  <?php
2  include("impala_connect.inc.php");
3  $query = new Query();
4  $query->query = 'select website,count(*) from apachelog group by website order by count(*) desc limit 10';
5
6  $queryHandle = $client->query($query);
7
8  $result = $client->fetch($queryHandle,false,100);
9  $data = $result->data;
10 echo "<center><h1>¨Ó¹Ç¹¢éÍÁÙÅ : ".count($data)."</h1></center>";
11 echo "<center><table border='1'><tr><th>web</th><th>visit</th></tr>";
12 foreach ($data as $data1)
13 {
14  $parts = preg_split('/\s+/', $data1);
15  $web = $parts[0];
16  $visit = $parts[1];
17  echo "<tr><td>$web</td><td>$visit</td></tr>";
18 }
19 echo "</table></center>";
20 $transport->close();
21 ?>
```

ภาพที่ 3-24 Code PHP สำหรับการใช้งาน Cloudera Impala

จากภาพที่ 3-24 Code PHP สำหรับติดต่อกับ Cloudera Impala สามารถอธิบายได้ดังนี้

บรรทัดที่ 2 ฟังก์ชัน includeจะเรียกไฟล์ชื่อว่า impala_connect.inc.php มาใช้งาน

บรรทัดที่ 3 เป็นการสร้าง Object ให้กับ ตัวแปร ชื่อ $query เพื่อใช้งาน

บรรทัดที่ 4 ใช้ $query->query เก็บคำสั่งการค้นหาข้อมูลจากฐานข้อมูล Cloudera Impala

บรรทัดที่ 6 ใช้ตัวแปร $queryHandle เก็บค่าที่ได้จากการ ส่งค่า $query ให้ $client->query

บรรทัดที่ 8 ใช้ตัวแปร $result เก็บค่าที่ได้จากการ $client->fetch-($queryHandle,false,100) ซึ่งจะทำให้ $result เป็น Object โดยจะเก็บค่าไว้หลายค่าแต่เราจะนำมาใช้เฉพาะค่า $result->data

บรรทัดที่ 9 ใช้ตัวแปล $data เก็บค่า $result->data

บรรทัดที่ 10-11 สร้างตารางส่วนหัวตาราง

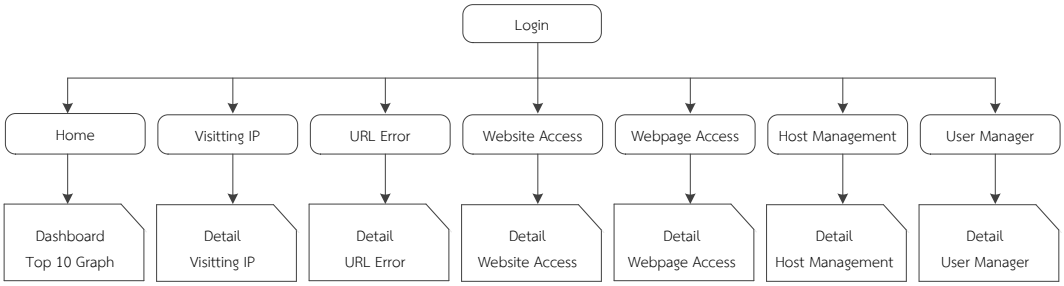
บรรทัดที่ 12 ใช้ฟังชั้น foreach ($data as $data1) เพื่อดึงอาร์เรย์ $data ใส่ $data1 ทีละตัวจนหมด

บรรทัดที่ 14 ใช้ preg_split('/\s+/', $data1) เพื่อตัดแยกข้อความในตัวแปร $data1 ออกมาโดยใช้ /S+ เป็นตัวแยกข้อความแล้วเก็บไว้ในตัวแปร $parts ในรูปแบบของอาร์เรย์

บรรทัดที่ 15 เป็นการเก็บตัวแปร $parts[0] ไปไว้ตัวแปร $web

บรรทัดที่ 16 เป็นการเก็บตัวแปร $parts[1] ไปไว้ตัวแปร $visit
บรรทัดที่ 17 สร้างตารางและใส่ข้อมูลของตัวแปร $web และ $visit
บรรทัดที่ 18 ปิด Tag ตารางและปิด Tag Center
บรรทัดที่ 20 ปิดการเชื่อมต่อ โดย $transport->close()

3.8 ออกแบบส่วนติดต่อผู้ใช้งาน



ภาพที่ 3-25 โครงสร้างเมนูและเนื้อหาต่างๆของเว็บไซต์

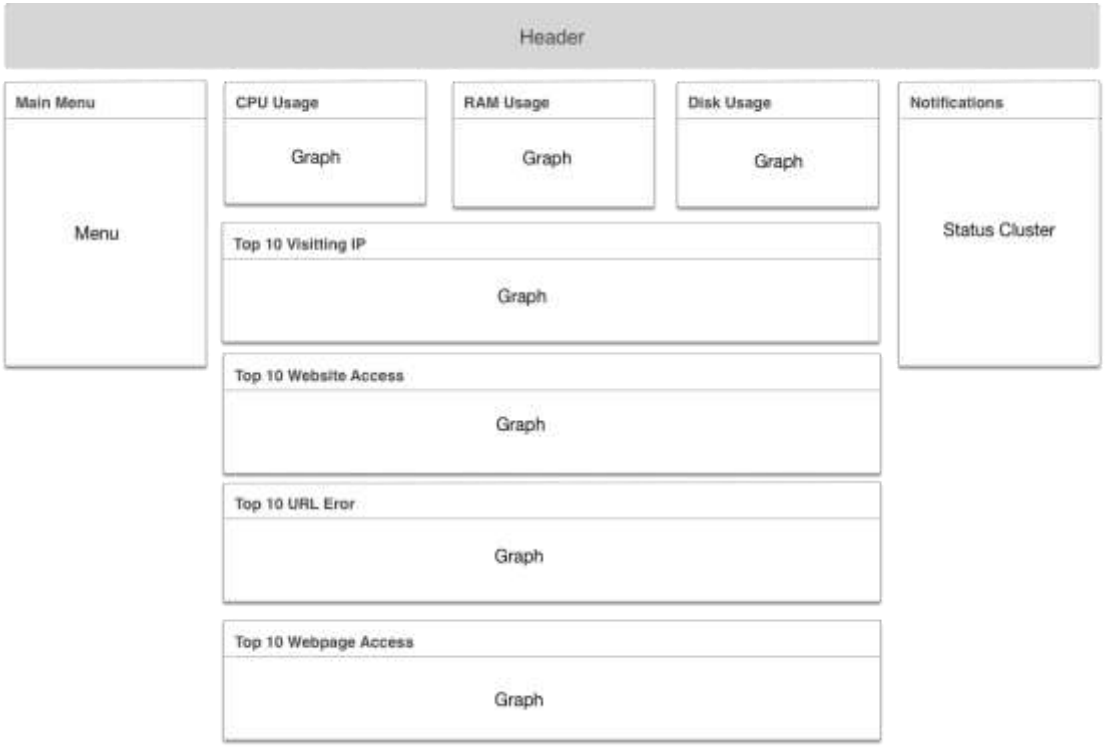
จากภาพที่ 3-25 แสดงโครงสร้างเว็บไซต์การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala ซึ่งระบบสามารถทำการประมวลผลข้อมูลจราจรจาก log file มาแสดงเป็นกราฟสถิติต่างๆ เช่น IP ของผู้เข้าชมเว็บไซต์ เป็นต้น ประกอบด้วยหน้าเว็บต่างๆดังนี้คือ หน้า Home, Visiting IP, URL Error, Website Access, Webpage Access, Host Management และหน้า User Manager

3.8.1 หน้า Login

ภาพที่ 3-26 โครงสร้างหน้า Login

จากภาพที่ 3-26 แสดงหน้าเว็บ Login สำหรับ Admin เท่านั้น เนื่องจากเว็บไซต์นี้เป็นเว็บที่จัดทำขึ้นเพื่อ Admin โดยเฉพาะสำหรับจัดการระบบ ไม่อนุญาตให้บุคคลภายนอกเข้ามาใช้งาน และแสดงหน้าต่างขึ้นมาให้พิสูจน์ตัวตน ให้ Admin กรอก Username และ Password เพื่อทำการ Login พิสูจน์ตัวตนก่อนเข้าใช้งาน

3.8.2 หน้า Home

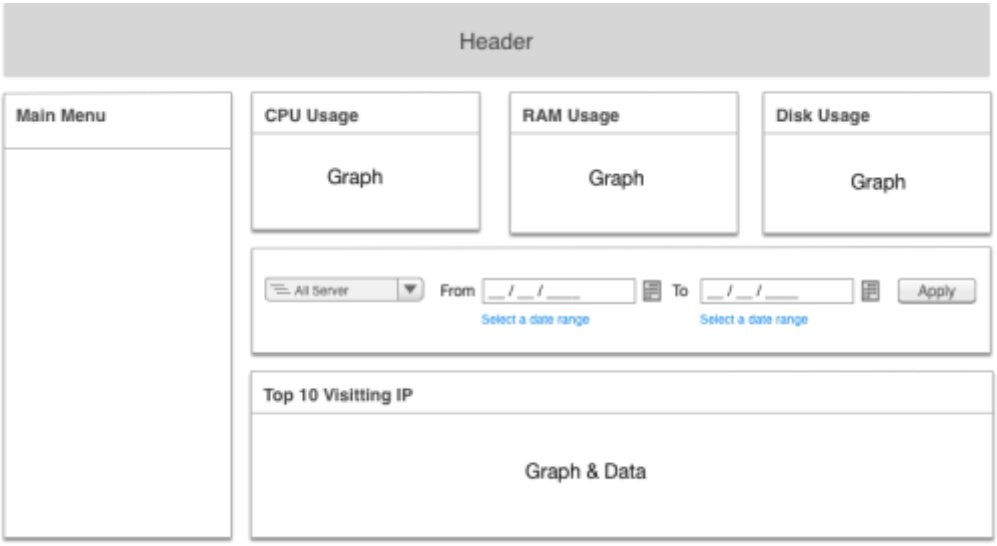


ภาพที่ 3-27 โครงสร้างหน้า HOME

จากภาพที่ 3-27 แสดงโครงสร้างหน้า Home ของเว็บไซต์ เมื่อ Admin ทำการ Login เข้ามาแล้วจะพบเว็บไซต์ดังภาพ ซึ่งส่วนหัวของเว็บไซต์ประกอบด้วยแถบเมนูสำหรับ Logout เพื่อทำการออกจากเว็บไซต์ และรายละเอียดเมนูอื่นๆ ดังต่อไปนี้

- Main Menu ประกอบด้วยเมนูย่อย Top 10 Visitting IP, Top 10 Website Access, Top 10 Webpage Access, Top 10 URL Error สำหรับดูรายละเอียดมากขึ้น
- CPU Usage แสดงการใช้งาน CPU ของเครื่อง NameNode
- RAM Usage แสดงการใช้งาน RAM ของเครื่อง NameNode
- Disk Usage แสดงการใช้งานพื้นที่เก็บข้อมูลบน HDFS
- Notifications แสดงสถานะของโปรแกรมที่รันอยู่บน NameNode
- Top 10 Visitting IP, Top 10 Website Access, Top 10 Webpage Access, Top 10 URL Error แสดงข้อมูลสรุปในรูปแบบกราฟ

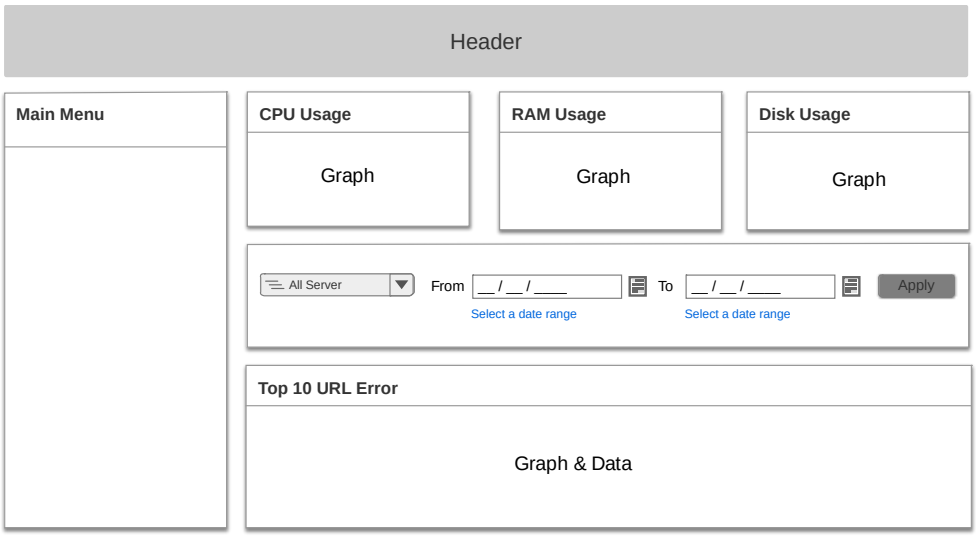
3.8.3 หน้า Visiting IP



ภาพที่ 3-28 ส่วนโครงสร้างแสดงรายละเอียดเพิ่มเติมของ Visiting IP

จากภาพที่ 3-28 เป็นโครงสร้างหน้าเว็บแสดงรายละเอียดเพิ่มเติมของกราฟ Visiting IP โดยจะแสดงกราฟสถิติหมายเลข IP พร้อมทั้งบอกว่า IP ของผู้เข้าชมเว็บไซต์ ณ ปัจจุบันหรือเวลาที่กำหนด และ ส่วนพื้นที่เนื้อหาของเว็บไซต์จะประกอบด้วย ปุ่มฟังก์ชันในการเลือกเครื่อง Server, เลือกช่วงเวลา วัน/เดือน/ปี จากปฏิทินว่าต้องการผลลัพธ์ให้แสดงเป็นช่วงเวลาใด ซึ่งเมื่อกำหนดช่วงเวลาแล้วกดปุ่ม Apply ระบบจะนำข้อมูลจราจรเฉพาะช่วงเวลาที่ต้องการมาประมวลผลแล้วแสดงออกมาเป็นกราฟในส่วนแสดงกราฟ Visiting IP โดยมีคำอธิบายกราฟอยู่ด้านล่างของกราฟ พร้อมทั้งแสดงรายการข้อมูลที่ได้ ณ ช่วงเวลานั้นๆที่เลือกในส่วนของตารางแสดงข้อมูล

3.8.4 หน้า URL Error



ภาพที่ 3-29 ส่วนโครงสร้างแสดงรายละเอียดเพิ่มเติมของ URL Error

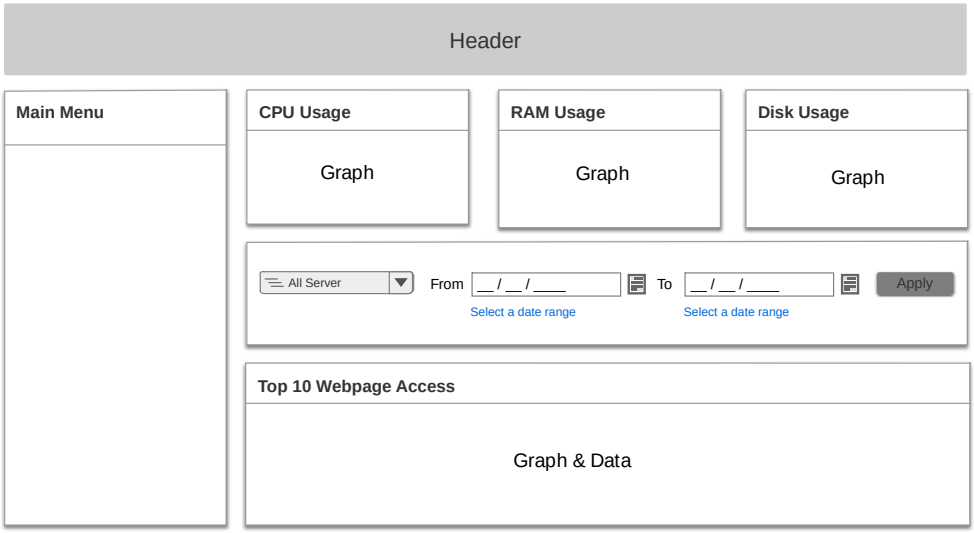
จากภาพที่ 3-29 เป็นโครงสร้างหน้าเว็บแสดงรายละเอียดเพิ่มเติมของกราฟ URL Error โดยจะแสดงกราฟสถิติ URL ของเว็บไซต์ที่เกิดข้อผิดพลาดจากการเข้าชมเว็บไซต์ ณ ปัจจุบันหรือเวลาที่กำหนดนั้นๆ ส่วนพื้นที่เนื้อหาของเว็บไซต์จะประกอบด้วย ปุ่มฟังก์ชันในการเลือกเครื่อง Server, เลือกช่วงเวลา วัน/เดือน/ปี จากปฏิทินว่าต้องการผลลัพธ์ให้แสดงเป็นช่วงเวลาใด ซึ่งเมื่อกำหนดช่วงเวลาแล้วกดปุ่ม Apply ระบบจะนำข้อมูลจราจรเฉพาะช่วงเวลาที่ต้องการมาประมวลผลแล้วแสดงออกมาเป็นกราฟในส่วนแสดงกราฟ URL Error โดยมีคำอธิบายกราฟอยู่ด้านล่างของกราฟ พร้อมทั้งแสดงรายการข้อมูลที่ได้ ณ ช่วงเวลานั้นๆที่เลือกในส่วนของตารางแสดงข้อมูล

3.8.5 หน้า Website Access

ภาพที่ 3-30 เป็นโครงสร้างหน้าเว็บแสดงรายละเอียดเพิ่มเติมของกราฟ Website Access โดยจะแสดงกราฟสถิติของเว็บไซต์มีผู้เข้าชมเว็บไซต์มากที่สุด ณ ปัจจุบันหรือเวลาที่กำหนดนั้นๆ และส่วนพื้นที่เนื้อหาของเว็บไซต์จะประกอบด้วย ปุ่มฟังก์ชันในการเลือกช่วงเวลา วัน/เดือน/ปี จากปฏิทินว่าต้องการผลลัพธ์ให้แสดงเป็นช่วงเวลาใด ซึ่งเมื่อกำหนดช่วงเวลาแล้วกดปุ่ม Apply ระบบจะนำข้อมูลจราจรเฉพาะช่วงเวลาที่ต้องการมาประมวลผลแล้วแสดงออกมาเป็นกราฟในส่วนแสดงกราฟ Website Access โดยมีคำอธิบายกราฟอยู่ด้านล่างของกราฟ พร้อมทั้งแสดงรายการข้อมูลที่ได้ ณ ช่วงเวลานั้นๆที่เลือกในส่วนของตารางแสดงข้อมูล

ภาพที่ 3-30 ส่วนโครงสร้างแสดงรายละเอียดเพิ่มเติมของ Website Access

3.8.6 หน้า Webpage Access



ภาพที่ 3-31 ส่วนโครงสร้างแสดงรายละเอียดเพิ่มเติมของ Webpage Access

จากภาพที่ 3-31 เป็นโครงสร้างหน้าเว็บแสดงรายละเอียดเพิ่มเติมของกราฟ Webpage Access โดยจะแสดงกราฟสถิติของหน้า Webpage มีผู้เข้าชมเว็บไซต์มากที่สุด ณ ปัจจุบันหรือเวลาที่กำหนดนั้นๆ สามารถเลือกเครื่อง Server, เลือกช่วงเวลา วัน/เดือน/ปี จากปฏิทินว่าต้องการผลลัพธ์ให้แสดงเป็นช่วงเวลาใด ซึ่งเมื่อกำหนดช่วงเวลาแล้วกดปุ่ม Apply ระบบจะนำข้อมูลจราจรเฉพาะช่วงเวลาที่ต้องการมาประมวลผลแล้วแสดงออกมาเป็นกราฟในส่วนแสดงกราฟ Webpage Access โดยมีคำอธิบายกราฟอยู่ด้านล่างของกราฟ พร้อมทั้งแสดงรายการข้อมูลที่ได้ ณ ช่วงเวลานั้นๆที่เลือกในส่วนของตารางแสดงข้อมูล

3.8.7 หน้า User Manager

ภาพที่ 3-32 ส่วนโครงสร้างเว็บไซต์หน้า User Manager

จากภาพที่ 3-32 เป็นโครงสร้างเว็บไซต์หน้า User Manager ใช้ในการเพิ่มผู้ใช้งานเข้าสู่ระบบฐานข้อมูล เมื่อเข้ามาจะมีแบบฟอร์มบังคับให้กรอก Name, Surname, Username, Password และ Description ให้ครบ โดยข้อมูลที่ทำการเพิ่มสำเร็จจะถูกเก็บไว้ในฐานข้อมูลและจะถูกจัดแสดงขึ้นในตาราง ซึ่งหากต้องการเปลี่ยนข้อมูล หรือลบ สามารถทำได้ในหน้านี้

3.8.8 หน้า Host Management

ภาพที่ 3-33 ส่วนโครงสร้างเว็บไซต์หน้า Host Management

จากภาพที่ 3-33 เป็นโครงสร้างเว็บไซต์หน้า Host Management แสดงข้อมูลเครื่อง Server ที่มีอยู่ในระบบ ปุ่ม Add New Server ใช้สำหรับเพิ่ม Server เข้าสู่ระบบฐานข้อมูล โดยต้องกรอกข้อมูล Name Server, URL Server และ Description ให้ครบ เมื่อทำการเพิ่มสำเร็จจะถูกเก็บไว้ในฐานข้อมูลและจะถูกจัดแสดงขึ้นในตาราง ซึ่งหากต้องการเปลี่ยนข้อมูล หรือลบ สามารถทำได้ในหน้านี้

3.8.9 ตารางฐานข้อมูล Mysql

การเก็บข้อมูลชื่อ Server จะถูกเก็บข้อมูลไว้ในตารางชื่อ tblserver มีโครงสร้างดังตารางที่ 3-3

ตารางที่ 3-3 รายละเอียดของการเก็บข้อมูลชื่อ Server ใน MySQL

Field	Type	Null	Key	Default	Extra
nameserver	Varchar(50)	No	PRI	Null	ชื่อ Hostname
url	Varchar(50)	No		Null	ชื่อ url ของเว็บไซต์
description	Varchar(50)	No		Null	รายละเอียดของ Hostname

บทที่ 4
ผลการดำเนินงาน

การดำเนินการโครงงานวิจัย “การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala (Design and Development of Data Traffic Analysis of Web Sever on HDFS System using Impala)” ระบบมีความสามารถในการทำงานตรงตามขอบเขตที่ได้กำหนดไว้

4.1 ผลการวัดประสิทธิภาพของ Cloudera Impala และ HDFS

4.2 หน้าเว็บไซต์การออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala

4.1 ผลการวัดประสิทธิภาพของ Impala และ HDFS

การทดสอบวัดประสิทธิภาพของ Cloudera Impala และ HDFS นั้นมีเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบทั้งจำนวนทั้งสิ้น 9 เครื่อง (NameNode 1 เครื่องและ DataNode 8 เครื่อง) โดยมีข้อมูลจำเพาะของเครื่องคอมพิวเตอร์ที่ใช้ทดสอบดังนี้

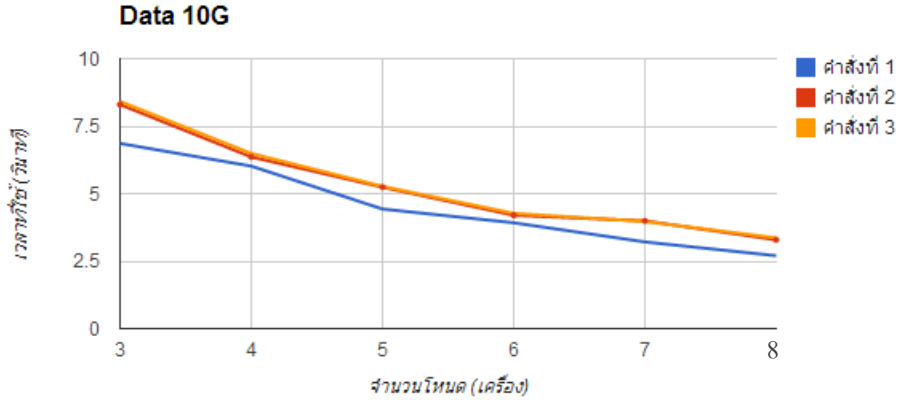
- CPU Intel® Core™2 Quad Processor Q9500
- Ram 8192 MB DDR3
- Hard disk 1TB 3G
- Graphics NVDIA® GeForce® GT220 3D

คำสั่งที่ใช้ในการสืบค้นข้อมูลในตาราง

- คำสั่งที่ 1 : select count(*) from ชื่อตาราง;
- คำสั่งที่ 2 : select ip,count(*) from ชื่อตาราง group by ip limit 10;
- คำสั่งที่ 3 : select ip,count(*) from ชื่อตาราง group by ip order by count(*) desc limit 10;

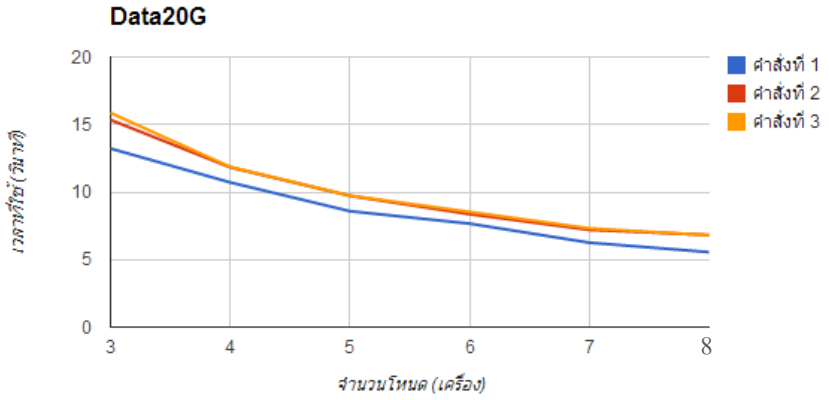
4.1.1 ทดลองการสืบค้นข้อมูลบนตารางที่เก็บข้อมูลในรูปแบบ Text

4.1.1.1 เวลาที่ใช้ดึงข้อมูล



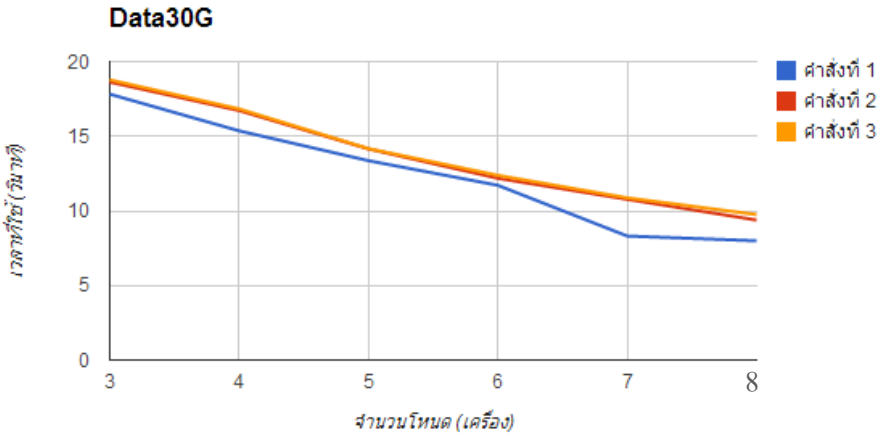
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	6.86	8.3	8.42
4	6.02	6.36	6.49
5	4.43	5.24	5.27
6	3.92	4.2	4.27
7	3.21	3.99	3.97
8	2.7	3.29	3.36

กราฟที่ 4-1 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Text ขนาด 10G



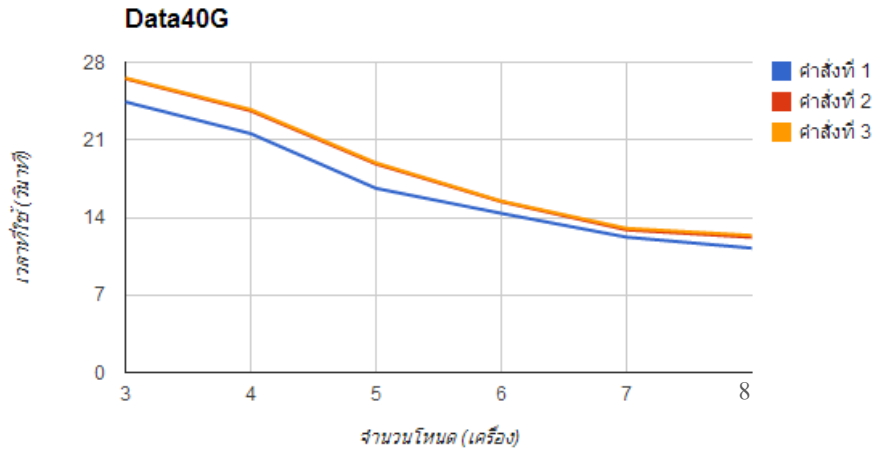
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	13.22	15.34	15.86
4	10.71	11.83	11.86
5	8.58	9.72	9.73
6	7.66	8.36	8.53
7	6.25	7.2	7.31
8	5.55	6.82	6.8

กราฟที่ 4-2 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Text ขนาด 20G



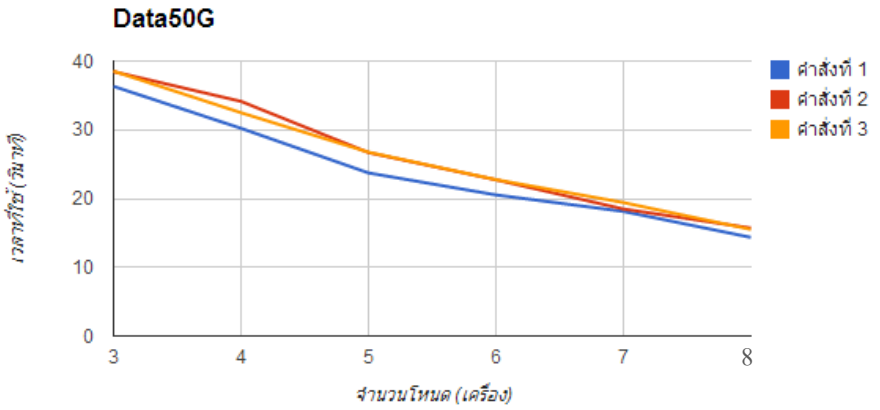
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	17.82	18.63	18.78
4	15.36	16.72	16.83
5	13.36	14.15	14.17
6	11.71	12.18	12.38
7	8.31	10.77	10.87
8	7.99	9.38	9.75

กราฟที่ 4-3 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Text ขนาด 30G



จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	24.42	26.52	26.58
4	21.56	23.62	23.73
5	16.62	18.82	18.92
6	14.36	15.42	15.47
7	12.21	12.87	13.01
8	11.22	12.2	12.37

กราฟที่ 4-4 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Text ขนาด 40G

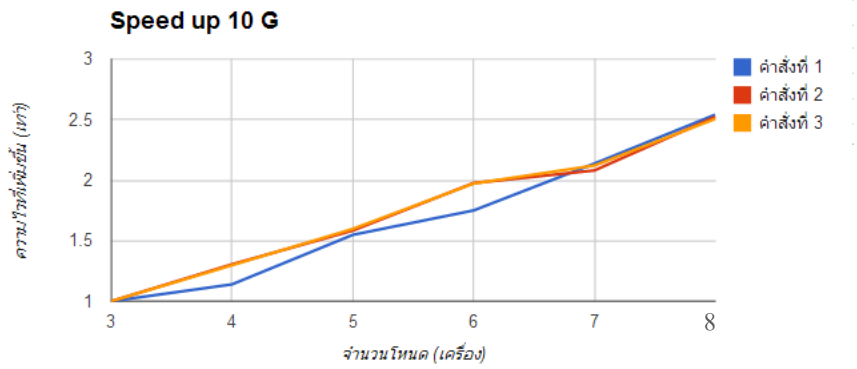


จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	36.32	38.46	38.52
4	30.21	34.13	32.48
5	23.72	26.68	26.74
6	20.52	22.72	22.74
7	18.12	18.47	19.41
8	14.34	15.7	15.5

กราฟที่ 4-5 กราฟแสดงผลการทดลองคำสั่งแบบ Text Data 50G

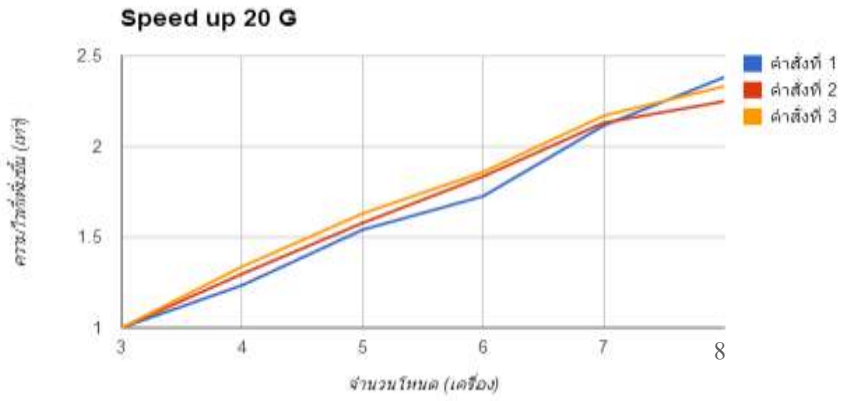
จากกราฟที่ 4-1 ถึงกราฟที่ 4-5 แสดงการเปรียบเทียบเวลาที่ใช้ในการสืบค้นข้อมูลการวัดผลโดยใช้ข้อมูลไฟล์ในรูปแบบ Text ขนาด 10 กิกะไบต์, 20 กิกะไบต์, 30 กิกะไบต์,40 กิกะไบต์ และ 50 กิกะไบต์ โดยใช้เครื่องที่ทำหน้าที่เป็น DataNode ในการทดลองเป็นจำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่องและ 8 เครื่องตามลำดับ ใช้คำสั่งทั้ง 3 คำสั่งทดลองการสืบค้นข้อมูลบนตารางที่เก็บข้อมูลในรูปแบบ Text จากการทดลองพบว่าจำนวนเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น DataNode มีผลต่อเวลาที่ใช้ในการสืบค้นข้อมูล โดยเมื่อมีจำนวนเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น DataNode มากขึ้นจะทำให้เวลาในการการสืบค้นข้อมูลลดลง จากการใช้คำสั่ง 3 คำสั่ง ผลออกมามีลักษณะที่คล้ายกันทั้ง 3 คำสั่ง คือเมื่อเพิ่มเครื่องคอมพิวเตอร์จะทำให้เวลาการสืบค้นน้อยลงตามลำดับ

4.1.1.2 ความเร็วที่เพิ่มขึ้นในการดึงข้อมูล



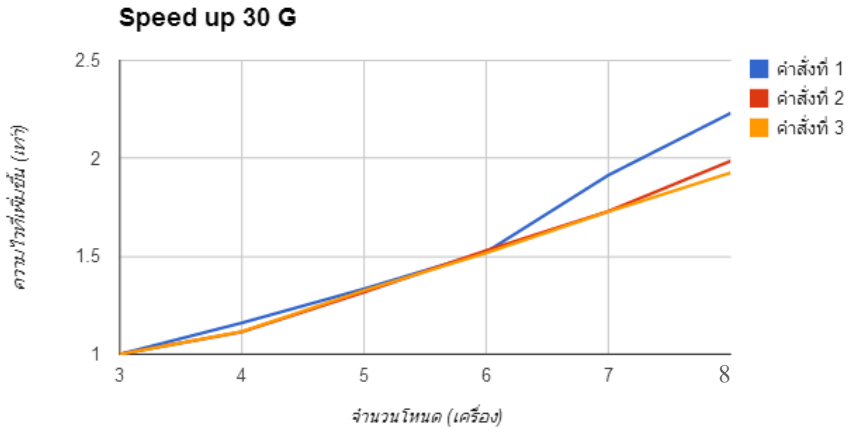
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.13953488372093	1.30503144654088	1.29738058551618
5	1.54853273137698	1.58396946564886	1.5977229601518
6	1.75	1.97619047619048	1.97189695550351
7	2.13707165109034	2.08020050125313	2.12090680100756
8	2.54074074074074	2.52279635258359	2.50595238095238

กราฟที่ 4-6 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Text ขนาด 10G



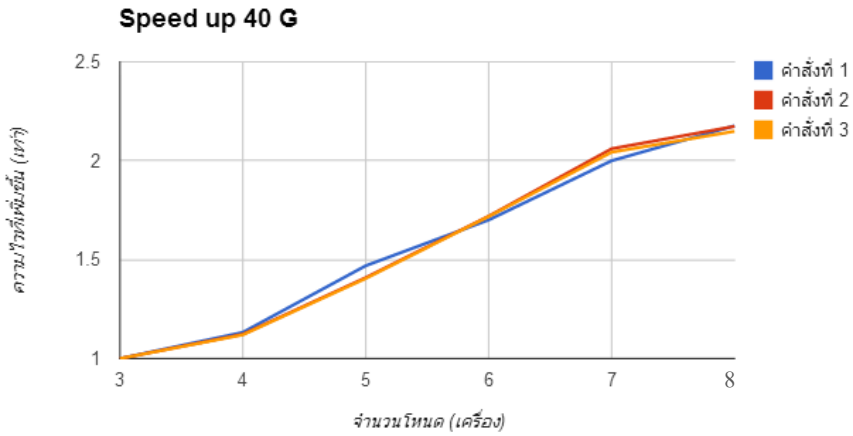
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.234360410831	1.2967032967033	1.33726812816189
5	1.54079254079254	1.57818930041152	1.63001027749229
6	1.72584856396867	1.83492822966507	1.85932004689332
7	2.1152	2.13055555555556	2.16963064295486
8	2.38198198198198	2.24926686217009	2.33235294117647

กราฟที่ 4-7 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Text ขนาด 20G



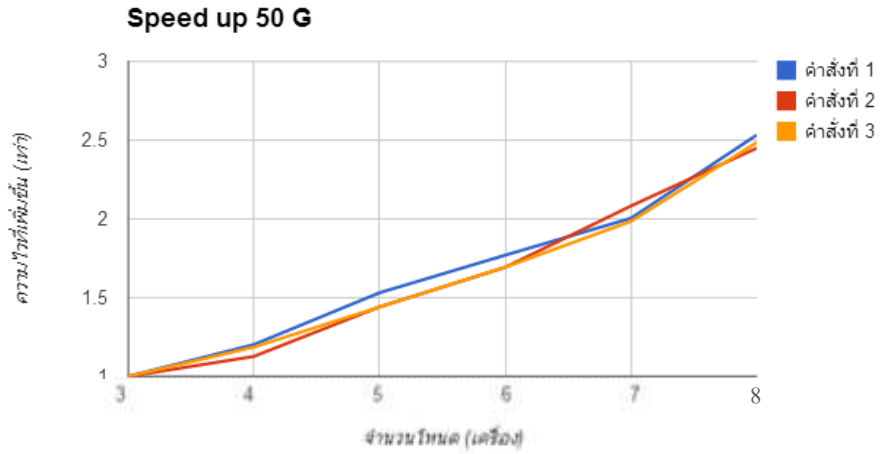
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.16015625	1.11423444976077	1.11586452762923
5	1.33383233532934	1.31660777385159	1.32533521524347
6	1.52177625960717	1.52955665024631	1.51696284329564
7	1.9140708915145	1.72980501392758	1.72769089236431
8	2.23028785982478	1.9861407249467	1.92615384615385

กราฟที่ 4-8 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Text ขนาด 30G



จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.13265306122449	1.12277730736664	1.12010113780025
5	1.46931407942238	1.40913921360255	1.40486257928118
6	1.70055710306407	1.71984435797665	1.71816418875242
7	2	2.06060606060606	2.04304381245196
8	2.17647058823529	2.17377049180328	2.14874696847211

กราฟที่ 4-9 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Text ขนาด 40G



จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.2022509102946	1.12686785818928	1.18596059113301
5	1.53119730185497	1.44152923538231	1.44053851907255
6	1.76998050682261	1.69278169014085	1.69393139841689
7	2.00441501103753	2.08229561451002	1.98454404945904
8	2.53277545327755	2.44968152866242	2.48516129032258

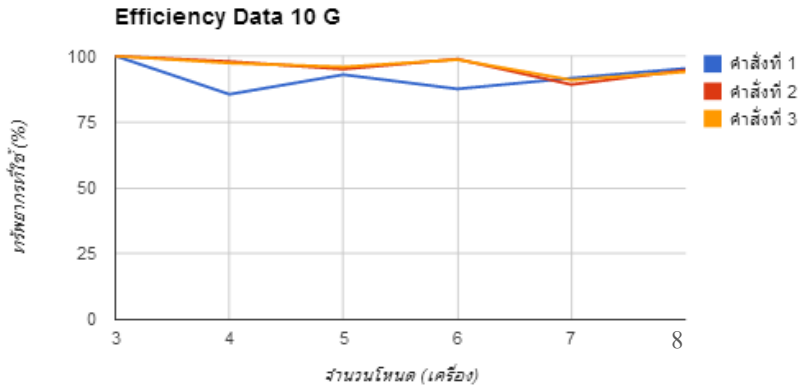
กราฟที่ 4-10 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Text ขนาด 50G

จากกราฟที่ 4-6 ถึงกราฟที่ 4-10 แสดงผลการทดสอบความเร็วในการสืบค้นข้อมูลที่เพิ่มขึ้นโดยใช้ข้อมูลไฟล์รูปแบบ Text ขนาด 10 กิกะไบต์, 20 กิกะไบต์, 30 กิกะไบต์, 40 กิกะไบต์ และ 50 กิกะไบต์ โดยใช้เครื่องในการทดลองเป็น DataNode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่องและ 8 เครื่อง เมื่อทำการเพิ่มจำนวนเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ พบว่าหากเพิ่มจำนวนเครื่องคอมพิวเตอร์ที่ใช้ทดสอบจากจำนวน 3 เครื่องเป็น 8 เครื่อง โดยหาได้จากสูตร

$$\text{ความเร็วที่เพิ่มขึ้น} = \frac{\text{เวลาของจำนวนเครื่องคอมพิวเตอร์ที่น้อยที่สุด}}{\text{เวลาของจำนวนเครื่องปัจจุบัน}} \qquad (4\text{-}1)$$

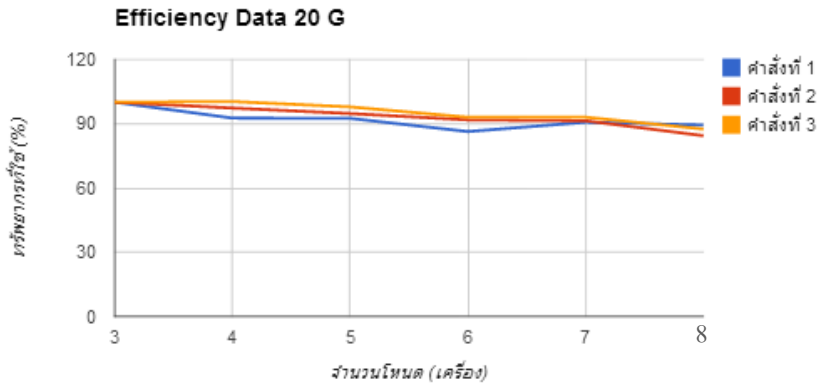
จะได้ค่าความเร็วที่เพิ่มขึ้นในการเพิ่มเครื่องคอมพิวเตอร์ในการสืบค้นข้อมูลและความเร็วในการสืบค้นข้อมูลของทั้ง 3 คำสั่งพบว่าเร็วขึ้นจากเดิมเฉลี่ยถึง 2.5 เท่าจากการใช้คอมพิวเตอร์ 3 เครื่อง เมื่อใช้คอมพิวเตอร์ 8 เครื่องโดยประมาณ

4.1.1.3 ประสิทธิภาพในการดึงข้อมูล



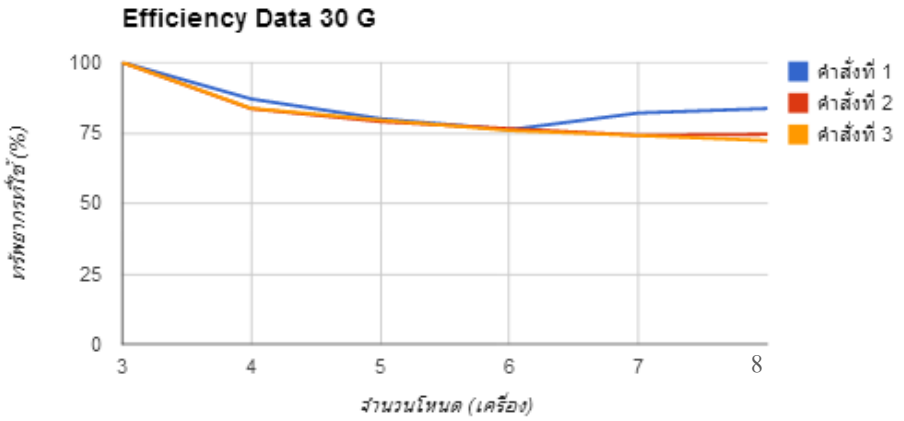
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	85.4651162790698	97.877358490566	97.3035439137134
5	92.9119638826185	95.0381679389313	95.8633776091082
6	87.5	98.8095238095238	98.5948477751756
7	91.588785046729	89.1514500537057	90.8960057574667
8	95.2777777777778	94.6048632218845	93.9732142857143

กราฟที่ 4-11 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Text ขนาด 10G



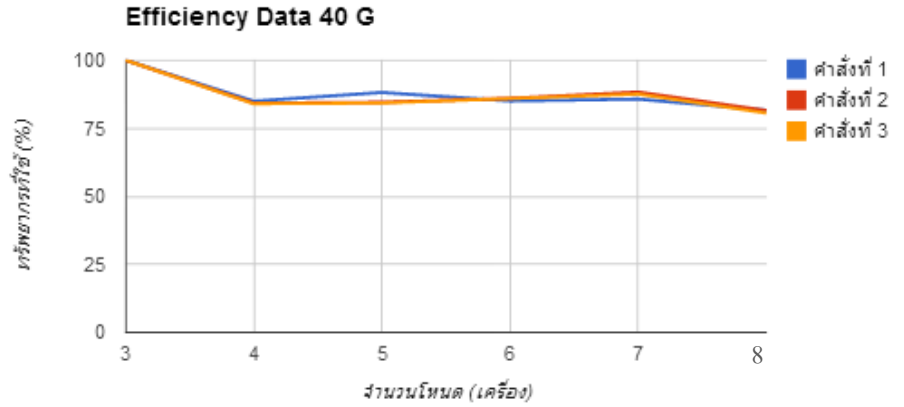
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	92.5770308123249	97.2527472527472	100.295109612142
5	92.4475524475524	94.6913580246914	97.8006166495375
6	86.2924281984334	91.7464114832536	92.9660023446659
7	90.6514285714286	91.3095238095238	92.984170412351
8	89.3243243243243	84.3475073313783	87.4632352941176

กราฟที่ 4-12 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Text ขนาด 20G



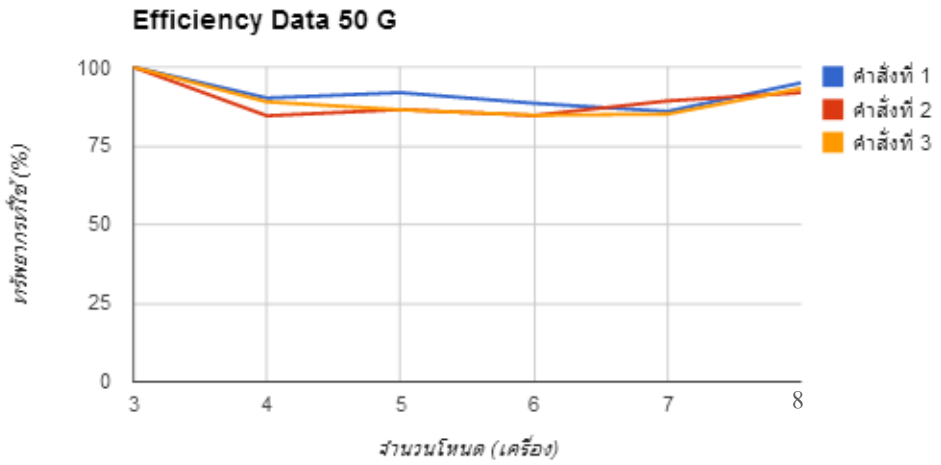
จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	87.01171875	83.5675837320574	83.6898395721925
5	80.0299401197605	78.9964664310954	79.5201129146083
6	76.0888129803587	76.4778325123153	75.8481421647819
7	82.0316096363357	74.1345005968962	74.0438953870417
8	83.6357947434293	74.4802771855011	72.2307692307692

กราฟที่ 4-13 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Text ขนาด 30G



จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	84.9489795918367	84.2082980524979	84.007585335019
5	88.158844765343	84.548352816153	84.291754756871
6	85.0278551532034	85.9922178988327	85.9082094376212
7	85.7142857142857	88.3116883116883	87.5590205336554
8	81.6176470588235	81.516393442623	80.5780113177041

กราฟที่ 4-14 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Text ขนาด 40G



จำนวนเครื่อง	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	90.1688182720953	84.5150893641957	88.9470443349754
5	91.8718381112985	86.4917541229385	86.432311144353
6	88.4990253411306	84.6390845070422	84.6965699208443
7	85.9035004730369	89.2412406218578	85.0518878339589
8	94.979079497908	91.8630573248408	93.1935483870968

กราฟที่ 4-15 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Text ขนาด 50G

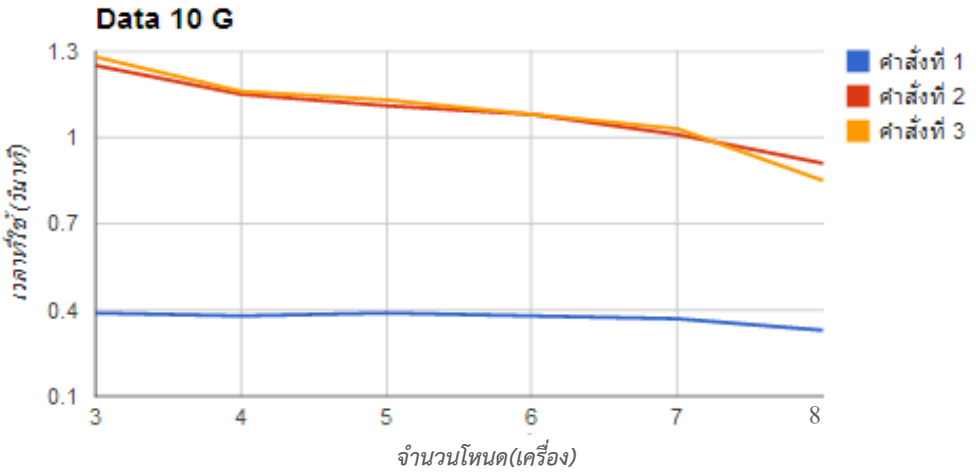
จากกราฟที่ 4-11 ถึงกราฟที่ 4-15 แสดงผลการทดสอบถึงประสิทธิภาพในการสืบค้นโดยใช้ข้อมูลไฟล์ในรูปแบบ Text ขนาด 10 กิกะไบต์, 20 กิกะไบต์, 30 กิกะไบต์, 40 กิกะไบต์ และ 50 กิกะไบต์ โดยใช้เครื่องในการทดลองเป็น DataNode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่องและ 8 เครื่อง ใช้ทรัพยากรของคอมพิวเตอร์ไปมากน้อยเพียงใด โดยหาประสิทธิภาพได้จากสูตร

$$\text{ประสิทธิภาพ} = (\text{ค่าความเร็ว} \times 100) / (\text{จำนวนเครื่อง}/3) \qquad (4\text{-}2)$$

โดยเพิ่มเครื่องและทำการเก็บผล ผลการทดสอบปรากฏว่าเมื่อเพิ่มคอมพิวเตอร์มากขึ้น ยังคงที่ใช้งานคอมพิวเตอร์แต่ละเครื่องมากกว่าได้เกือบเต็มประสิทธิภาพ ทุกขนาดข้อมูล ของคอมแต่ละเครื่อง สามารถเพิ่มคอมพิวเตอร์ได้โดยเครื่องคอมพิวเตอร์ทุกเครื่องยังคงทำงานเต็มประสิทธิภาพ

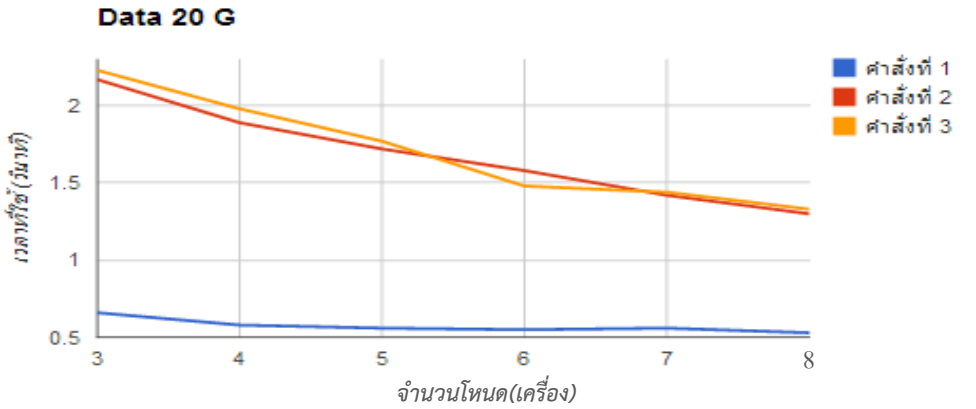
4.1.2 ทดลองการสืบค้นข้อมูลบนตารางที่เก็บข้อมูลในรูปแบบ Parquet

4.1.2.1 เวลาที่ใช้ในการดึงข้อมูล



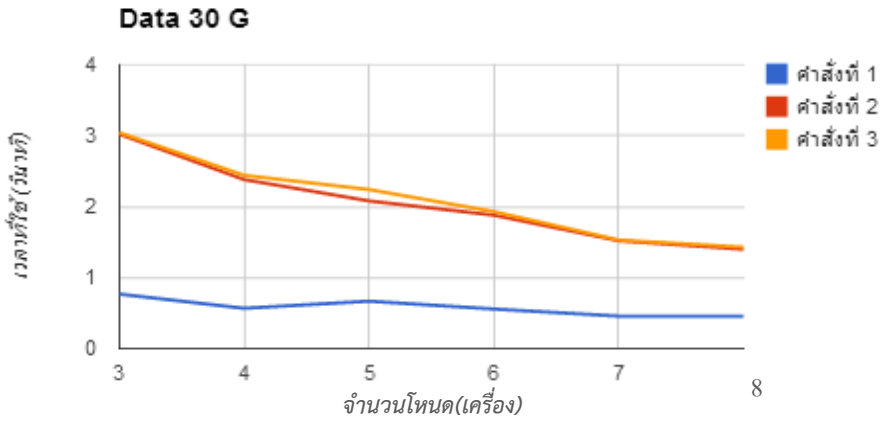
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	0.39	1.25	1.28
4	0.38	1.15	1.16
5	0.39	1.11	1.13
6	0.38	1.08	1.08
7	0.37	1.01	1.03
8	0.33	0.91	0.85

กราฟที่ 4-16 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Parquet ขนาด 10G



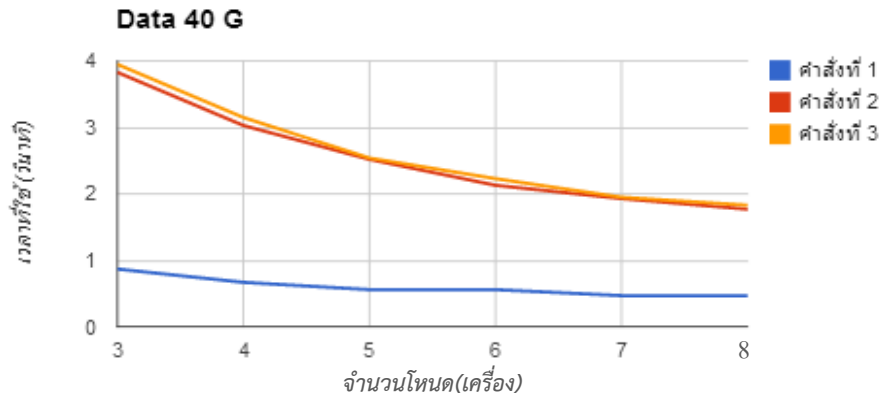
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	0.66	2.17	2.23
4	0.58	1.89	1.98
5	0.56	1.72	1.77
6	0.55	1.58	1.48
7	0.56	1.42	1.44
8	0.53	1.3	1.33

กราฟที่ 4-17 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Parquet ขนาด 20G



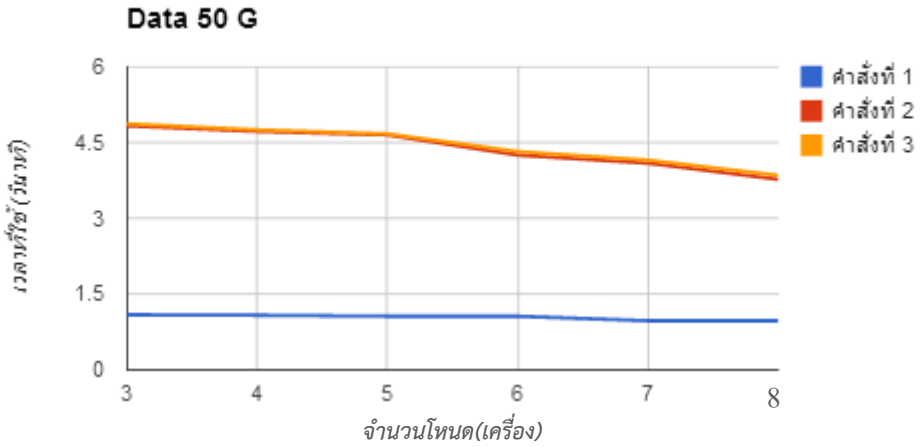
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	0.77	3.02	3.04
4	0.57	2.38	2.44
5	0.67	2.08	2.24
6	0.56	1.88	1.93
7	0.46	1.52	1.53
8	0.46	1.4	1.43

กราฟที่ 4-18 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Parquet ขนาด 30G



Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	0.87	3.83	3.95
4	0.67	3.03	3.15
5	0.56	2.52	2.54
6	0.56	2.13	2.23
7	0.47	1.93	1.95
8	0.47	1.77	1.83

กราฟที่ 4-19 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Parquet ขนาด 40G

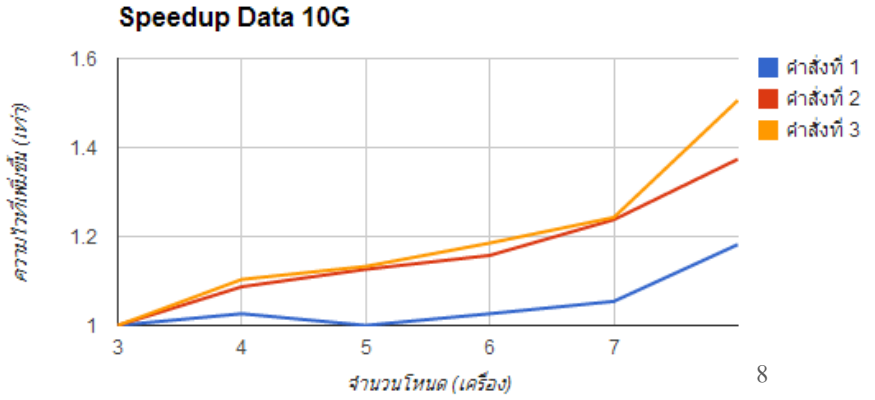


Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1.09	4.83	4.87
4	1.08	4.72	4.75
5	1.06	4.65	4.67
6	1.06	4.25	4.32
7	0.97	4.09	4.15
8	0.97	3.77	3.85

กราฟที่ 4-20 กราฟแสดงผลการทดลองคำสั่งบนข้อมูลแบบ Parquet ขนาด 50G

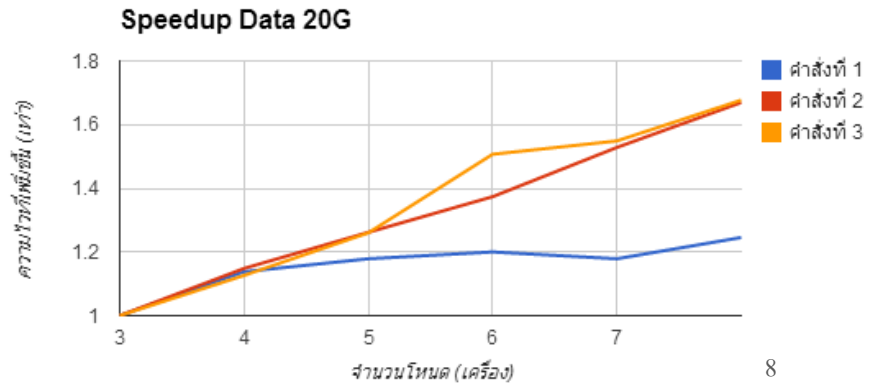
จากกราฟที่ 4-20 แสดงการเปรียบเทียบเวลาที่ใช้ในการสืบค้นข้อมูลการวัดผลโดยใช้ข้อมูลแบบไฟล์ในรูปแบบ Parquet ขนาด 10 กิกะไบต์ , 20 กิกะไบต์ , 30 กิกะไบต์ , 40 กิกะไบต์ และ 50 กิกะไบต์ โดยใช้เครื่องในการทดลองเป็น Datanode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่องและ 8 เครื่อง โดยใช้คำสั่งทดลองทีละคำสั่งตามลำดับด้วยทั้ง 3 คำสั่งทดลองการสืบค้นข้อมูลบนตารางที่เก็บข้อมูลในรูปแบบ Parquet จากการทดลองพบว่าหากใช้จำนวนเครื่องคอมพิวเตอร์จะส่งผลต่อเวลาที่ใช้ในการสืบค้นข้อมูล จำนวนเครื่องคอมพิวเตอร์มากขึ้นทำให้เวลาในการสืบค้นข้อมูลลดลง ทดสอบโดยการใช้คำสั่ง 3 คำสั่งผลออกมาคำสั่งที่ 1 ใช้เวลาในการค้นหาน้อยที่สุด

4.1.2.2 ความเร็วที่เพิ่มขึ้นในการดึงข้อมูล



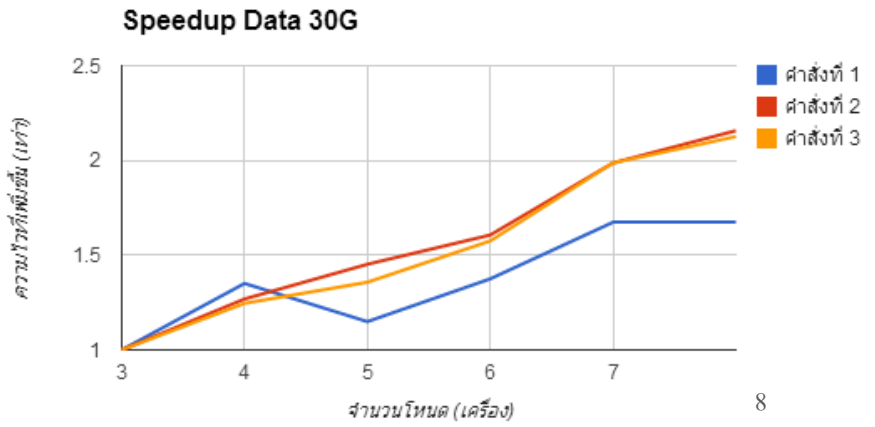
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.02631578947368	1.08695652173913	1.10344827586207
5	1	1.12612612612613	1.13274336283186
6	1.02631578947368	1.15740740740741	1.18518518518519
7	1.05405405405405	1.23762376237624	1.24271844660194
8	1.18181818181818	1.37362637362637	1.50588235294118

กราฟที่ 4-21 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Parquet ขนาด 10G



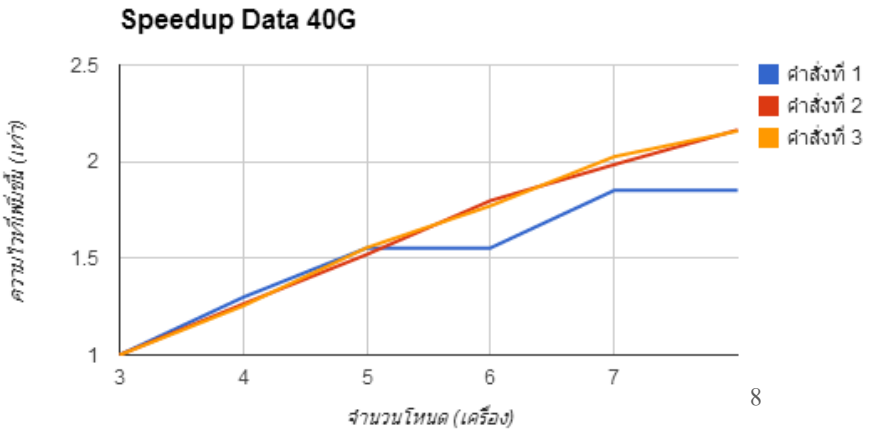
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.13793103448276	1.14814814814815	1.12626262626263
5	1.17857142857143	1.26162790697674	1.25988700564972
6	1.2	1.37341772151899	1.50675675675676
7	1.17857142857143	1.52816901408451	1.54861111111111
8	1.24528301886792	1.66923076923077	1.67669172932331

กราฟที่ 4-22 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Parquet ขนาด 20G



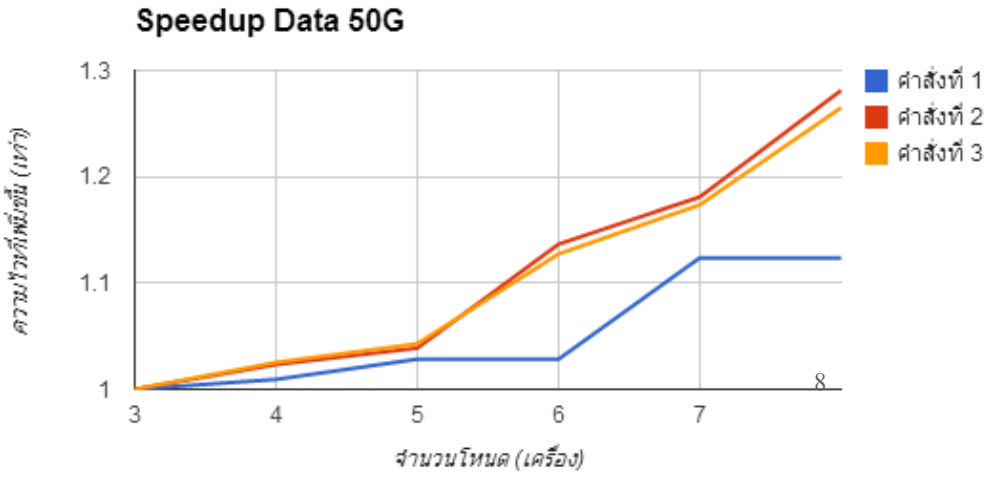
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.35087719298246	1.26890756302521	1.24590163934426
5	1.14925373134328	1.45192307692308	1.35714285714286
6	1.375	1.6063829787234	1.57512953367876
7	1.67391304347826	1.98684210526316	1.98692810457516
8	1.67391304347826	2.15714285714286	2.12587412587413

กราฟที่ 4-23 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Parquet ขนาด 30G



Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.29850746268657	1.26402640264026	1.25396825396825
5	1.55357142857143	1.51984126984127	1.55511811023622
6	1.55357142857143	1.7981220657277	1.77130044843049
7	1.85106382978723	1.98445595854922	2.02564102564103
8	1.85106382978723	2.1638418079096	2.15846994535519

กราฟที่ 4-24 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Parquet ขนาด 40G



Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	1	1	1
4	1.00925925925926	1.02330508474576	1.02526315789474
5	1.02830188679245	1.03870967741935	1.04282655246253
6	1.02830188679245	1.13647058823529	1.12731481481481
7	1.12371134020619	1.18092909535452	1.17349397590361
8	1.12371134020619	1.28116710875332	1.26493506493506

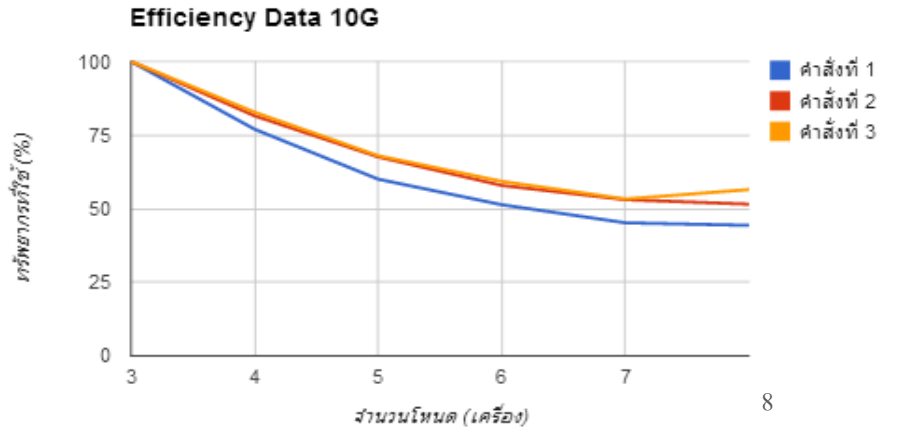
กราฟที่ 4-25 กราฟแสดงผลความเร็วที่เพิ่มขึ้นในการดึงข้อมูลแบบ Parquet ขนาด 50G

จากกราฟที่ 4-21 ถึงกราฟที่ 4-25 แสดงผลการทดสอบความเร็วในการสืบค้นข้อมูลที่เพิ่มขึ้นโดยใช้ข้อมูลแบบไฟล์ในรูปแบบ Parquet ขนาด 10 กิกะไบต์ , 20 กิกะไบต์ , 30 กิกะไบต์ , 40 กิกะไบต์ และ 50 กิกะไบต์ โดยใช้เครื่องในการทดลองเป็น Datanode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่องและ 8 เครื่อง เมื่อทำการเพิ่มจำนวนเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบ พบว่าหากเพิ่มจำนวนเครื่องคอมพิวเตอร์ที่ใช้ทดสอบจากจำนวน 3 เครื่องเป็น 8 เครื่อง โดยหาได้จากสูตร

$$\text{ความเร็วที่เพิ่มขึ้น} = \frac{\text{เวลาของจำนวนเครื่องคอมพิวเตอร์ที่น้อยที่สุด}}{\text{เวลาของจำนวนเครื่องปัจจุบัน}} \qquad (4\text{-}3)$$

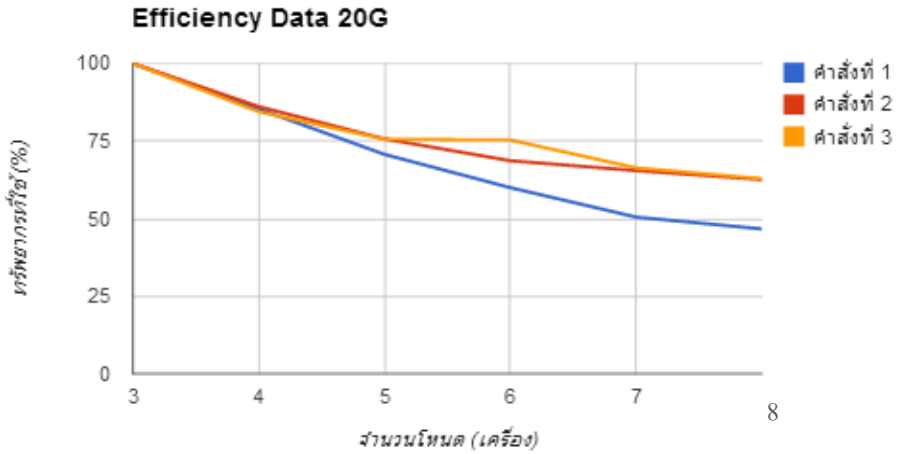
จะได้ค่าความเร็วที่เพิ่มขึ้นในการเพิ่มเครื่องคอมพิวเตอร์ในการสืบค้นข้อมูลและความเร็วในการสืบค้นข้อมูลของทั้ง 3 คำสั่งพบว่าเร็วขึ้นจากเดิมเฉลี่ยถึง 0.5 เท่าโดยประมาณ

4.1.2.3 ประสิทธิภาพในการดึงข้อมูล



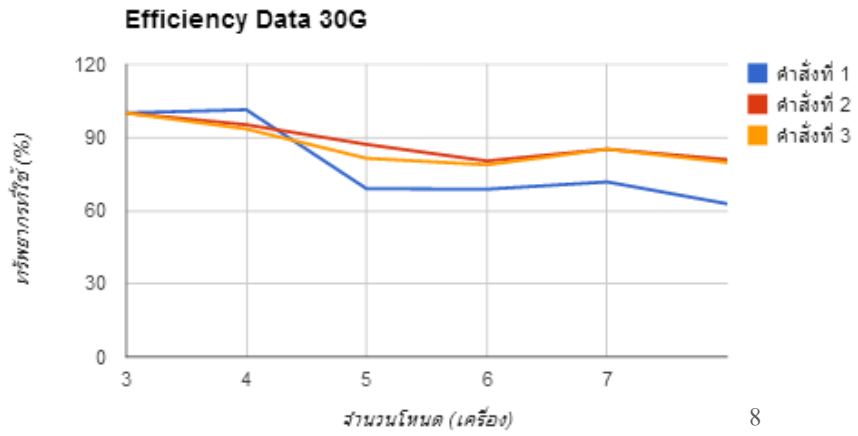
Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	76.9736842105263	81.5217391304348	82.7586206896552
5	60	67.5675675675676	67.9646017699115
6	51.3157894736842	57.8703703703704	59.2592592592592
7	45.1737451737452	53.041018387553	53.2593619972261
8	44.3181818181818	51.510989010989	56.4705882352941

กราฟที่ 4-26 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Parquet ขนาด 10G



Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	85.3448275862069	86.1111111111111	84.469696969697
5	70.7142857142857	75.6976744186046	75.593220338983
6	60	68.6708860759494	75.3378378378378
7	50.5102040816326	65.4929577464789	66.3690476190476
8	46.6981132075472	62.5961538461538	62.875939849624

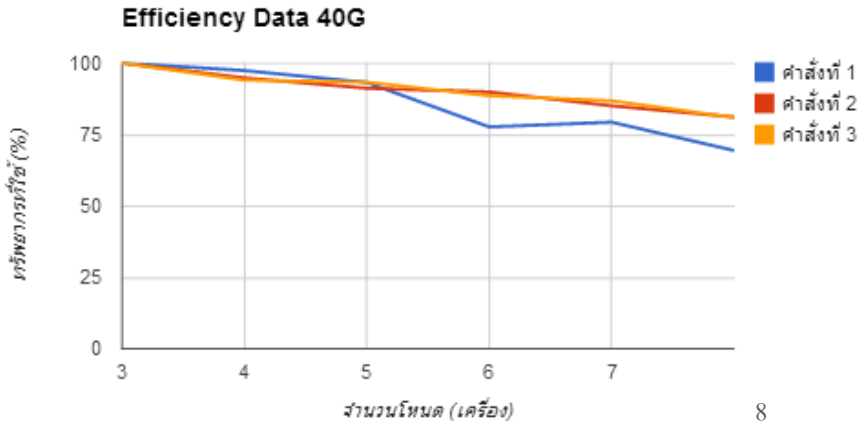
กราฟที่ 4-27 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Parquet ขนาด 20G



8

Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	101.315789473684	95.1680672268908	93.4426229508197
5	68.955223880597	87.1153846153846	81.4285714285714
6	68.75	80.3191489361702	78.7564766839378
7	71.7391304347826	85.1503759398496	85.1540616246498
8	62.7717391304348	80.8928571428572	79.7202797202797

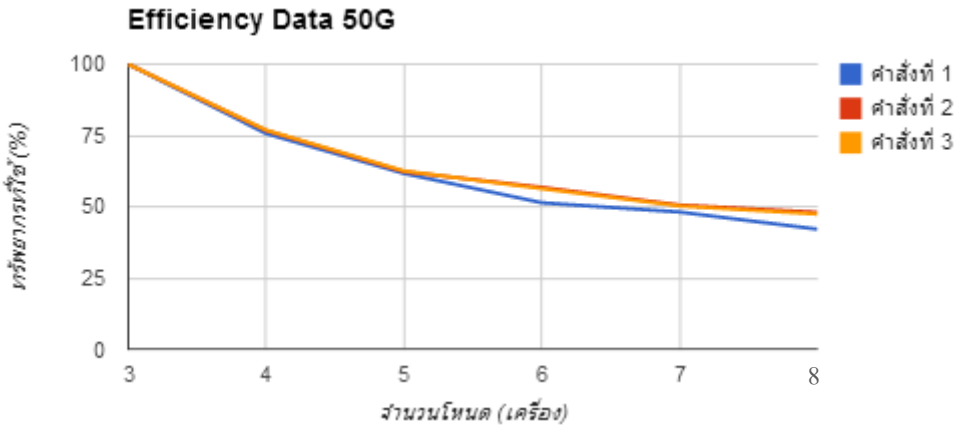
กราฟที่ 4-28 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Parquet ขนาด 30G



8

Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	97.3880597014925	94.8019801980198	94.047619047619
5	93.2142857142857	91.1904761904762	93.3070866141732
6	77.6785714285714	89.906103286385	88.5650224215247
7	79.3313069908815	85.0481125092524	86.8131868131868
8	69.4148936170213	81.1440677966102	80.9426229508197

กราฟที่ 4-29 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Parquet ขนาด 40G



Node	คำสั่งที่ 1	คำสั่งที่ 2	คำสั่งที่ 3
3	100	100	100
4	75.6944444444444	76.7478813559322	76.8947368421053
5	61.6981132075472	62.3225806451613	62.5695931477516
6	51.4150943396226	56.8235294117647	56.3657407407407
7	48.159057437408	50.6112469437653	50.2925989672978
8	42.139175257732	48.0437665782493	47.4350649350649

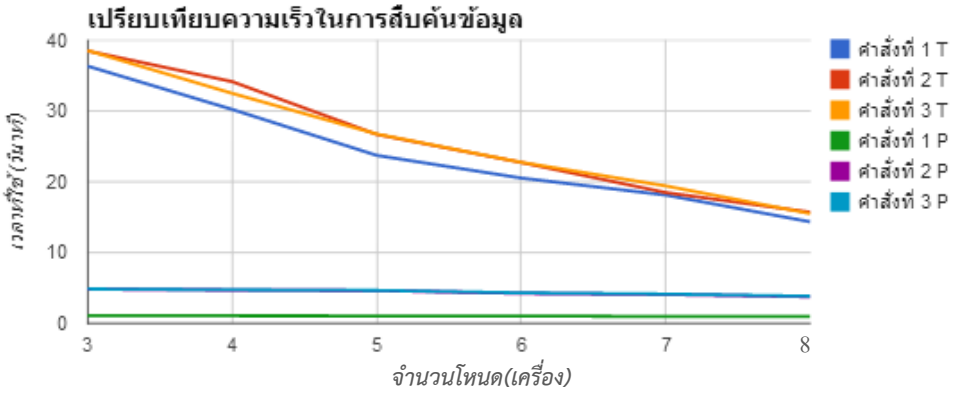
กราฟที่ 4-30 กราฟแสดงผลการทดลองประสิทธิภาพของข้อมูลแบบ Parquet ขนาด 50G

จากกราฟที่ 4-26 ถึงกราฟที่ 4-30 แสดงผลการทดสอบถึงประสิทธิภาพในการสืบค้นโดยใช้ข้อมูลในรูปแบบ Parquet ขนาด 10 กิกะไบต์ , 20 กิกะไบต์ , 30 กิกะไบต์ , 40 กิกะไบต์ , และ 50 กิกะไบต์ โดยใช้เครื่องในการทดลองเป็น Datanode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่อง และ 8 เครื่อง ใช้ทรัพยากรของคอมพิวเตอร์ไปมากน้อยเพียงใด โดยหาประสิทธิภาพได้จากสูตร

ประสิทธิภาพ = (ค่าความเร็วที่เพิ่มขึ้น x 100) / (จำนวนเครื่อง/3) (4-4)

โดยเพิ่มเครื่องและทำการเก็บผล ผลการทดสอบปรากฏว่าเมื่อเพิ่มคอมพิวเตอร์มากขึ้น ประสิทธิภาพการใช้งานคอมพิวเตอร์แต่ละเครื่องลดลง ทุกขนาดข้อมูล มีผลเหมือนกัน โดยที่ยิ่งเพิ่มจำนวนคอมพิวเตอร์มากขึ้น การทำงานของคอมพิวเตอร์ไม่ได้ทำงานอย่างเต็มประสิทธิภาพ จากผลการทดลองจะเห็นได้ว่าไฟล์ข้อมูลแบบ Parquet สามารถสืบค้นข้อมูลได้เร็วอยู่แล้วการเพิ่มคอมพิวเตอร์มากขึ้นส่งผลให้ใช้งานเครื่องคอมพิวเตอร์ไม่คุ้มค่ากับขนาดข้อมูลที่เพิ่มขึ้นเท่าไรนัก

4.1.3 การเปรียบเทียบประสิทธิภาพผลการทดลองการเก็บข้อมูลแบบ Text และ Parquet

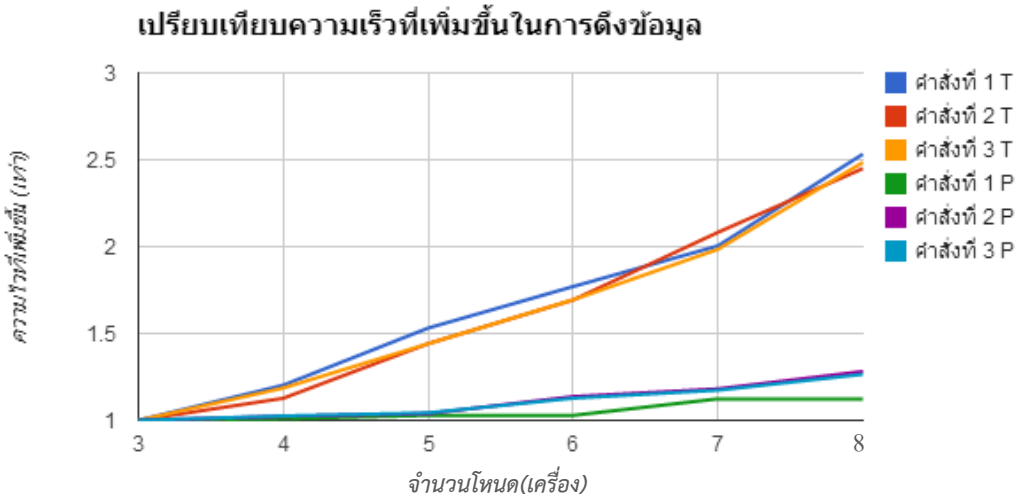


แบบ Text Data 50 G			
Node	คำสั่งที่ 1 T	คำสั่งที่ 2 T	คำสั่งที่ 3 T
3	36.32	38.46	38.52
4	30.21	34.13	32.48
5	23.72	26.68	26.74
6	20.52	22.72	22.74
7	18.12	18.47	19.41
8	14.34	15.7	15.5

แบบ parquet Data 50G			
Node	คำสั่งที่ 1 P	คำสั่งที่ 2 P	คำสั่งที่ 3 P
3	1.09	4.83	4.87
4	1.08	4.72	4.75
5	1.06	4.65	4.67
6	1.06	4.25	4.32
7	0.97	4.09	4.15
8	0.97	3.77	3.85

กราฟที่ 4-31 กราฟแสดงผลการทดลองการเปรียบเทียบ Data 50G

จากกราฟที่ 4-31 แสดงการค้นหาข้อมูลแบบ ไฟล์รูปแบบ Text และไฟล์ในรูปแบบ Parquet เป็นการเทียบความเร็วในการค้นหาข้อมูลในเครื่องคอมพิวเตอร์ที่เป็น DataNode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่อง และ 8 เครื่อง โดยใช้ข้อมูลจำนวน 50 กิกะไบต์จะเห็นได้ว่า การค้นหาข้อมูลแบบ Parquet จะใช้เวลาน้อยกว่าแบบ Text มากและหากเพิ่มเครื่องคอมพิวเตอร์มากขึ้นในขนาดข้อมูล 50 กิกะไบต์ การค้นหาข้อมูลแบบ Text จะใช้เวลาน้อยลง แต่ก็ยังใช้เวลาเยอะกว่าแบบ Parquet ดังนั้นจากผลการทดลองพบว่าข้อมูลแบบ Parquet จะใช้เวลาในการสืบค้นข้อมูลภายในที่น้อยกว่าไฟล์แบบ Text อย่างมาก

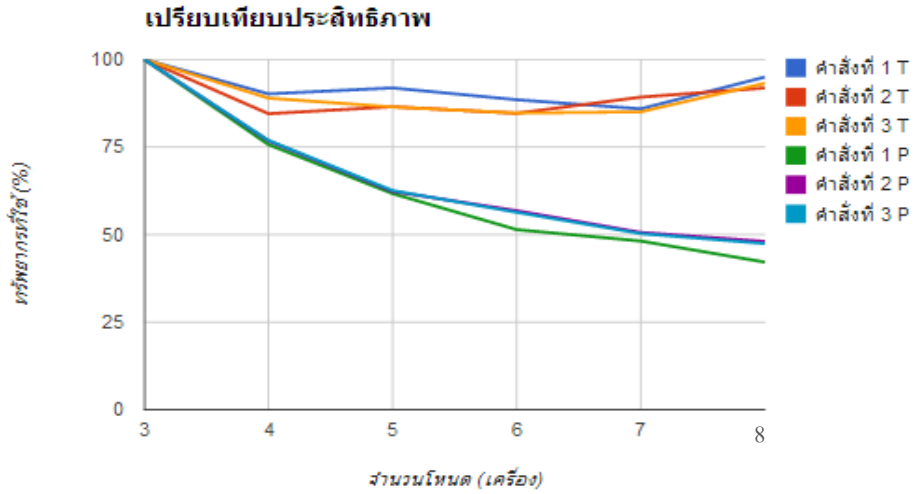


แบบ Text Data 50 G			
Node	คำสั่งที่ 1 T	คำสั่งที่ 2 T	คำสั่งที่ 3 T
3	1	1	1
4	1.2022509102946	1.12686785818928	1.18596059113301
5	1.53119730185497	1.44152923538231	1.44053851907255
6	1.76998050682261	1.69278169014085	1.69393139841689
7	2.00441501103753	2.08229561451002	1.98454404945904
8	2.53277545327755	2.44968152866242	2.48516129032258

แบบ parquet Data 50G			
Node	คำสั่งที่ 1 P	คำสั่งที่ 2 P	คำสั่งที่ 3 P
3	1	1	1
4	1.00925925925926	1.02330508474576	1.02526315789474
5	1.02830188679245	1.03870967741935	1.04282655246253
6	1.02830188679245	1.13647058823529	1.12731481481481
7	1.12371134020619	1.18092909535452	1.17349397590361
8	1.12371134020619	1.28116710875332	1.26493506493506

กราฟที่ 4-32 กราฟแสดงผลการทดลองการเปรียบเทียบ Data 50G

จากกราฟที่ 4-32 แสดงการเปรียบเทียบความเร็วที่เพิ่มขึ้นในการค้นหาข้อมูลในเครื่องคอมพิวเตอร์ที่เป็น DataNode จำนวน 3 เครื่อง, 4 เครื่อง, 5 เครื่อง, 6 เครื่อง, 7 เครื่อง และ 8 เครื่อง โดยใช้ข้อมูลจำนวน 50 กิกะไบต์จากผลการทดลองเพิ่มเครื่องคอมพิวเตอร์มากขึ้นจะเห็นได้ว่าการสืบค้นข้อมูลแบบ Text เร็วขึ้นในขณะที่การค้นหาข้อมูลในรูปแบบ Parquet นั้นมีความไวในการค้นหาอยู่แล้ว ดังนั้นการเพิ่มเครื่องคอมพิวเตอร์มากขึ้นไม่ค่อยมีผลต่อความไวในการค้นหาข้อมูลแบบ Parquet มากนัก



แบบ Text Data 50 G			
Node	คำสั่งที่ 1 T	คำสั่งที่ 2 T	คำสั่งที่ 3 T
3	100	100	100
4	90.1688182720953	84.5150893641957	88.9470443349754
5	91.8718381112985	86.4917541229385	86.432311144353
6	88.4990253411306	84.6390845070422	84.6965699208443
7	85.9035004730369	89.2412406218578	85.0518878339589
8	94.979079497908	91.8630573248408	93.1935483870968

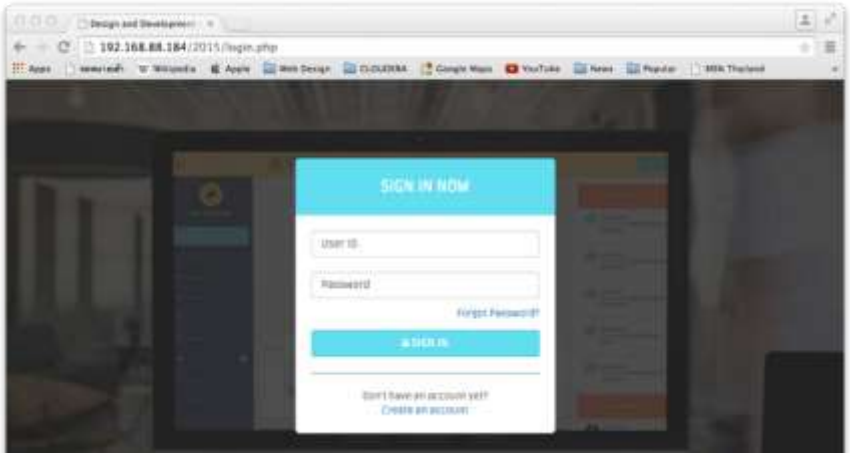
แบบ parquet Data 50G			
Node	คำสั่งที่ 1 P	คำสั่งที่ 2 P	คำสั่งที่ 3 P
3	100	100	100
4	75.6944444444444	76.7478813559322	76.8947368421053
5	61.6981132075472	62.3225806451613	62.5695931477516
6	51.4150943396226	56.8235294117647	56.3657407407407
7	48.159057437408	50.6112469437653	50.2925989672978
8	42.139175257732	48.0437665782493	47.4350649350649

กราฟที่ 4-33 กราฟแสดงผลการทดลองการเรียบเทียบ Data 50G

กราฟที่ 4-33 แสดงการกราฟและตาราง เปรียบเทียบประสิทธิภาพ ในการค้นหาข้อมูลในเครื่องคอมพิวเตอร์ที่เป็น DataNode จำนวน 3 เครื่อง 4 เครื่อง 5 เครื่อง 6 เครื่อง 7 เครื่อง และ 8 เครื่อง โดยใช้ข้อมูลจำนวน 50 กิกะไบต์ ซึ่งในการเพิ่มจำนวนคอมพิวเตอร์ในรูปแบบ Text ยังคงใช้ประสิทธิภาพของเครื่องคอมพิวเตอร์ได้อย่างเต็มประสิทธิภาพ แต่การค้นหาข้อมูลแบบ Parquet ในการเพิ่มเครื่องคอมพิวเตอร์จะทำให้ประสิทธิภาพลดลง แสดงให้เห็นถึงความไม่จำเป็นต้องเพิ่มเครื่องคอมพิวเตอร์ในขณะที่มีข้อมูลจำนวน 50 กิกะไบต์

4.2 หน้าเว็บไซต์

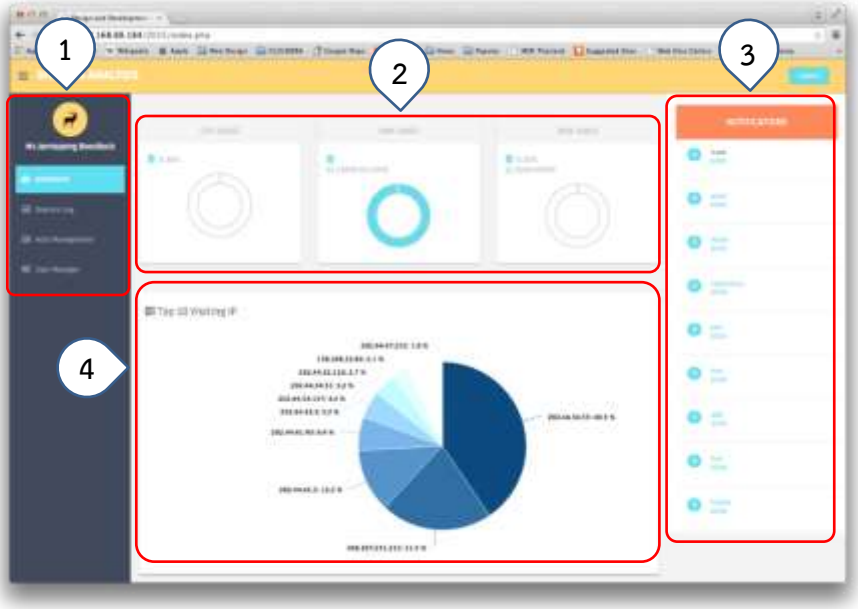
4.2.1 หน้า Login



ภาพที่ 4-1 หน้า Login

จากภาพที่ 4-34 แสดงหน้าเว็บ Login สำหรับ Admin เท่านั้น เนื่องจากเว็บไซต์นี้เป็นเว็บที่จัดทำขึ้นเพื่อ Admin โดยเฉพาะสำหรับจัดการระบบ ไม่อนุญาตให้บุคคลภายนอกเข้ามาใช้งานหน้าต่างให้พิสูจน์ตัวตน ให้ Admin กรอก Username และ Password เพื่อทำการพิสูจน์ตัวตนก่อนเข้าใช้งาน

4.2.2 หน้า Home



ภาพที่ 4-2 หน้า Home

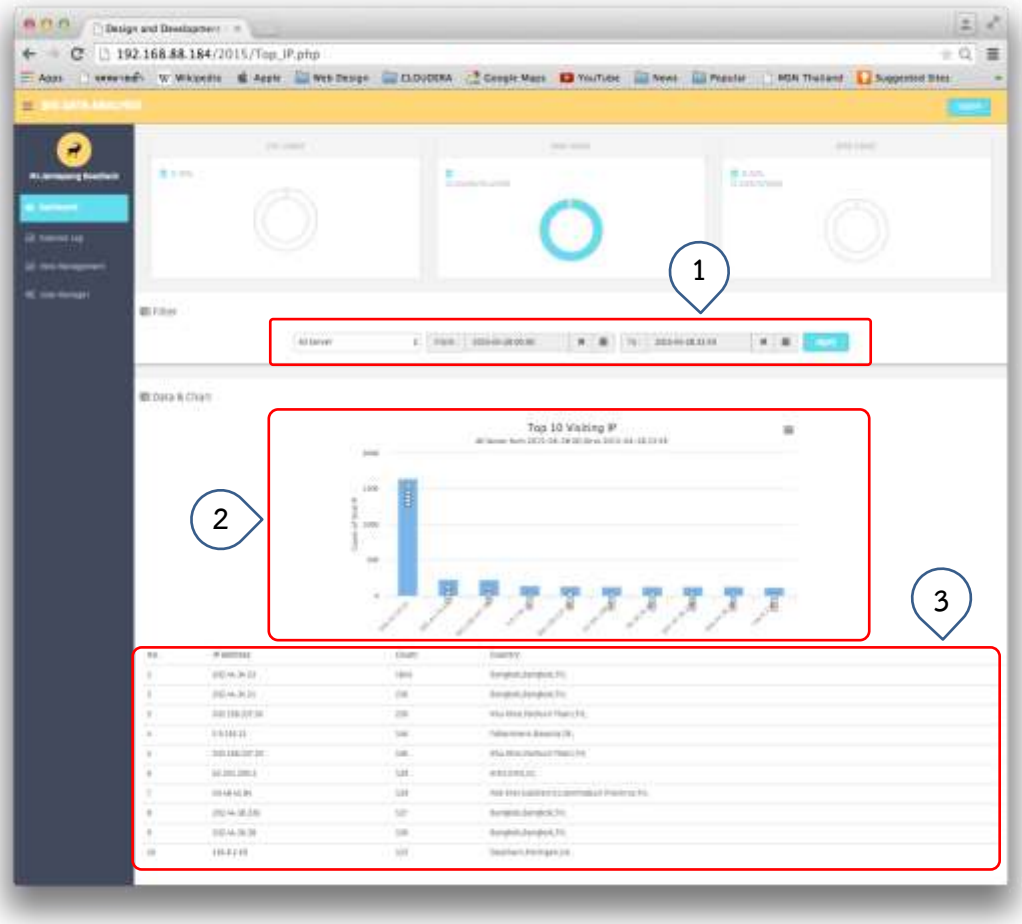
จากภาพที่ 4-35 มีส่วนประกอบต่างๆดังนี้ คือ

- หมายเลขที่ 1 เป็นส่วนเมนูหลัก ซึ่งประกอบไปด้วยเมนู Top 10 Visiting IP, Top 10

Top 10 Website Access, Top 10 URL Error, Top 10 Webpage Access, User Manager และ Host Management

- หมายเลขที่ 2 เป็นส่วนแสดงการใช้งาน CPU, RAM และ Disk ของเครื่อง NameNode
- หมายเลขที่ 3 เป็นส่วนแสดงสถานะการทำงานของ Cluster
- หมายเลขที่ 4 ส่วนแสดงกราฟ Top 10 ของ Visiting IP, Website Access, URL Error และ Webpage Access

4.2.3 หน้า Visitting IP

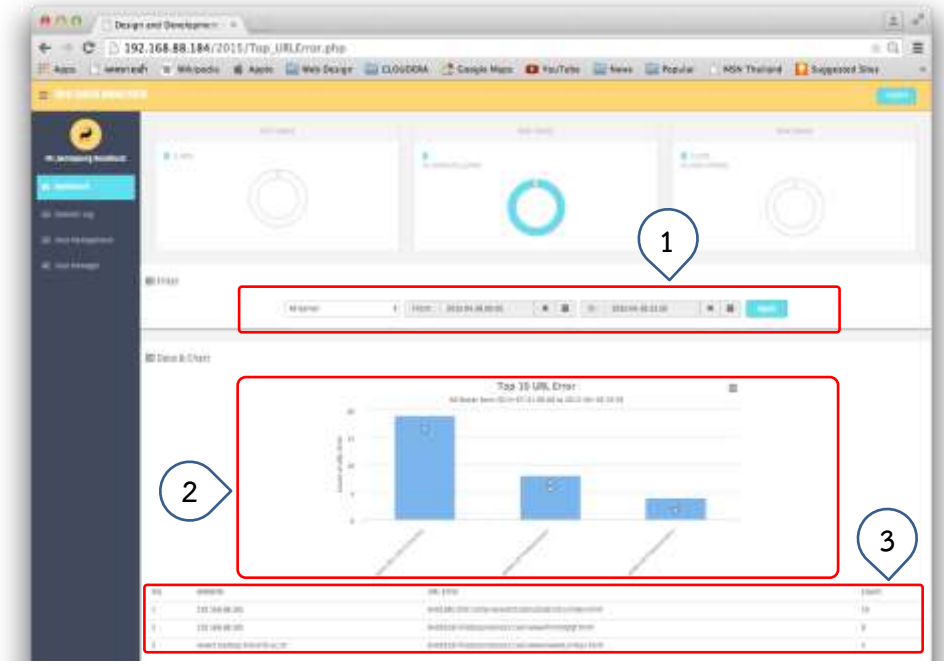


ภาพที่ 4-3 หน้า Visiting IP

ภาพที่ 4-36 แสดงหน้าเว็บในส่วนของ Visiting IP มีส่วนประกอบต่างๆ ดังนี้คือ

- หมายเลข 1 มี Dropdown menu สำหรับเลือกเครื่อง Server และเมนูเลือกช่วงเวลาวันที่ สำหรับเลือกระยะเวลาที่ต้องการให้ข้อมูลแสดง สามารถคลิกเพื่อเลือกวันที่/เดือน/ปี และเวลาในการแสดงข้อมูลได้
- หมายเลข 2 แสดงกราฟ 10 อันดับแรกของ IP ที่เข้าชมเว็บไซต์
- หมายเลข 3 ส่วนตารางแสดงข้อมูล IP ของผู้ที่เข้าชมเว็บไซต์มากสุด 10 อันดับในช่วงเวลาที่ต้องการ

4.2.4 หน้า URL Error



ภาพที่ 4-4 หน้า URL Error

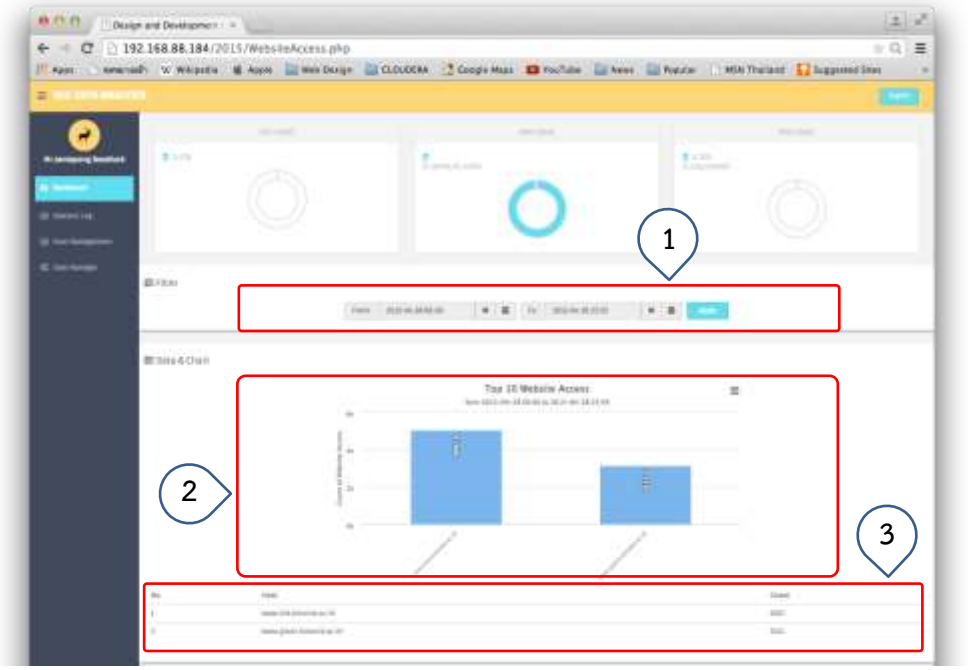
จากภาพที่ 4-37 แสดงหน้าเว็บไซต์ URL Error ประกอบด้วยส่วนต่างๆ ดังนี้

- หมายเลขที่ 1 สำหรับเลือกเครื่อง Server ที่มีอยู่ในระบบ เลือกช่วงเวลาวันที่/เดือน/ปี ที่ต้องการให้แสดง และปุ่ม Apply เพื่อสืบค้นข้อมูลที่ต้องการมาแสดง
- หมายเลขที่ 2 พื้นที่แสดงกราฟ URL Error
- หมายเลขที่ 3 ตารางแสดงข้อมูลรายการ URL ที่เกิดข้อผิดพลาดจากการเข้าชมในช่วงเวลาที่ต้องการ 10 อันดับ

4.2.5 หน้า Website Access

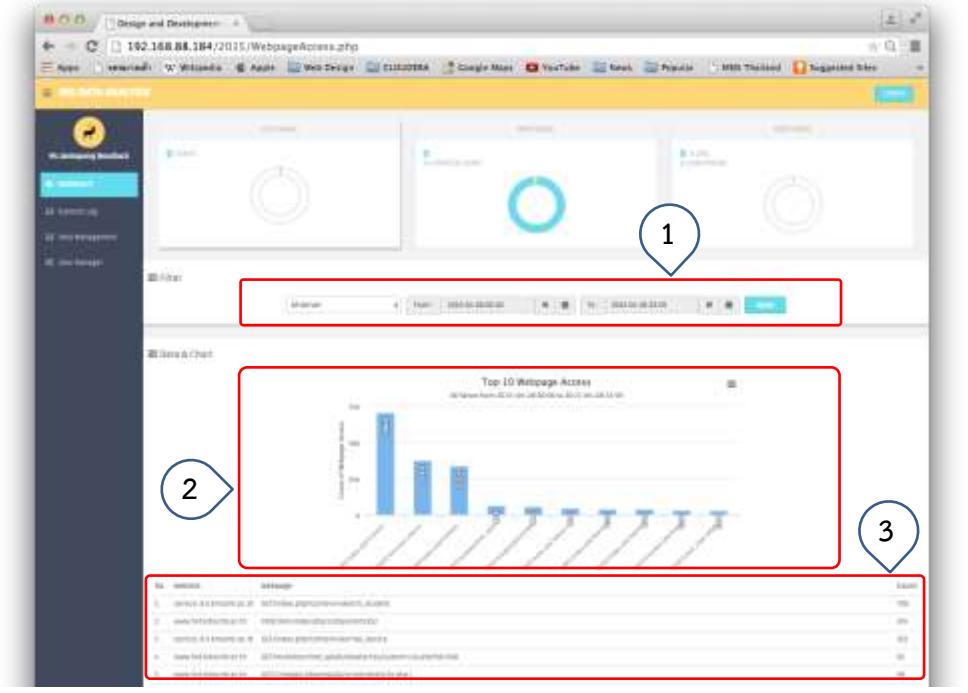
ภาพที่ 4-38 แสดงหน้าเว็บไซต์ Website Access ประกอบด้วยส่วนต่างๆ ดังนี้

- หมายเลข 1 สำหรับเลือกช่วงเวลาวันที่/เดือน/ปี ที่ต้องการให้แสดง และปุ่ม Apply เพื่อสืบค้นข้อมูลที่ต้องการมาแสดง
- หมายเลข 2 พื้นที่แสดงกราฟ 10 อันดับเว็บไซต์ที่มีผู้เข้าชมมากที่สุด
- หมายเลข 3 ตารางแสดงข้อมูลรายการเว็บไซต์และจำนวนที่มีผู้เข้าชมในช่วงเวลาที่กำหนด



ภาพที่ 4-5 หน้า Website Access

4.2.6 หน้า Webpage Access

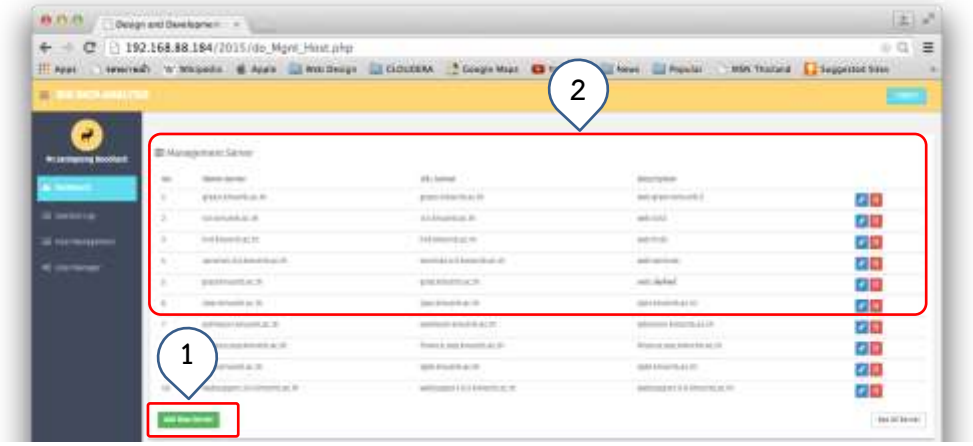


ภาพที่ 4-6 หน้า Webpage Access

ภาพที่ 4-39 แสดงหน้าเว็บไซต์ Webpage Access ประกอบด้วยส่วนต่างๆ ดังนี้

- หมายเลข 1 สำหรับเลือกเครื่อง Server ที่มีอยู่ในระบบ เลือกช่วงเวลาวันที่/เดือน/ปี ที่ต้องการให้แสดง และปุ่ม Apply เพื่อสืบค้นข้อมูลที่ต้องการมาแสดง
- หมายเลข 2 พื้นที่แสดงกราฟ Webpage Access
- หมายเลข 4 ตารางแสดงข้อมูลรายการหน้าเว็บเพจและจำนวนผู้เข้าชมในช่วงเวลาที่กำหนด

4.2.7 หน้า Host Management



ภาพที่ 4-7 หน้า Host Management

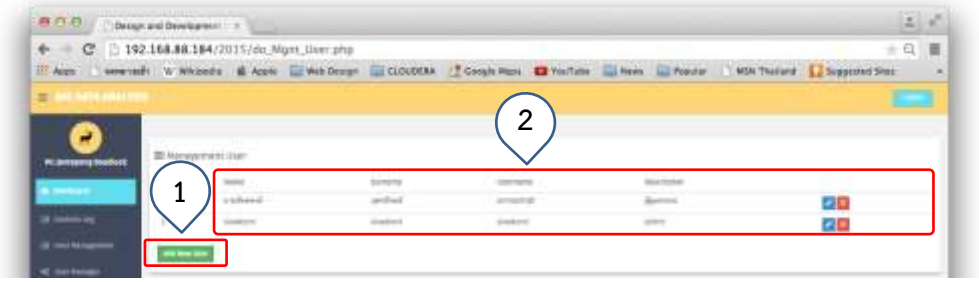
ภาพที่ 4-40 แสดงหน้าเว็บไซต์ Host Management ประกอบด้วยส่วนต่างๆ ดังนี้

- หมายเลข 1 ปุ่มฟังก์ชันสำหรับ Add New Server เข้ามาในฐานข้อมูล
- หมายเลข 2 ตารางแสดงรายการ Name Server, URL Server และ Description ที่มีอยู่ในระบบ ซึ่งสามารถทำการ ลบ หรือแก้ไขข้อมูลในตารางฐานข้อมูลได้

4.2.8 หน้า User Manager

ภาพที่ 4-41 แสดงหน้าเว็บไซต์ User Manager ประกอบด้วยส่วนต่างๆ ดังนี้

- หมายเลข 1 ปุ่มฟังก์ชันสำหรับ Add New User เข้ามาในฐานข้อมูล
- หมายเลข 2 ตารางแสดงรายละเอียดของผู้ใช้ที่มีอยู่ในระบบ ซึ่งสามารถทำการ ลบ หรือแก้ไขข้อมูลในตารางฐานข้อมูลได้



ภาพที่ 4-8 หน้า Add Server

บทที่ 5
สรุปผลการดำเนินงาน

การจัดทำวิจัยการออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala สามารถสรุปการทำงาน ปัญหาและอุปสรรคต่างๆที่เกิดขึ้นระหว่างการทำวิจัย รวมถึงแนวทางแก้ไขปัญหาและข้อเสนอแนะเพื่อที่จะได้นำวิจัยนี้พัฒนาต่อไปในอนาคต

5.1 สรุป
5.2 ปัญหาและแนวทางแก้ไขปัญหา
5.3 ข้อเสนอแนะและแนวทางพัฒนา

5.1 สรุป

ในการจัดทำวิจัยการออกแบบและพัฒนาระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala เป็นวิจัยที่ทำเกี่ยวกับการวิเคราะห์ข้อมูลจราจร Access Log และ Error Log ของเครื่องคอมพิวเตอร์แม่ข่ายเว็บโดยใช้ Apache Flume ในการส่งข้อมูลจราจรผู้เข้าชมเว็บไซต์ของเครื่องคอมพิวเตอร์แม่ข่ายเว็บเก็บไปไว้ใน HDFS หลังจากนั้นใช้ Cloudera Impala ในการสืบค้นข้อมูลจราจรผู้เข้าเยี่ยมชมเว็บไซต์ของเครื่องคอมพิวเตอร์แม่ข่ายเว็บที่จัดเก็บอยู่ใน HDFS และข้อมูลที่ได้จะถูกนำมาประมวลผลออกมาแสดงบนเว็บไซต์และพัฒนาเว็บไซต์ด้วยภาษา PHP เพื่อแสดงกราฟและข้อมูลการให้บริการต่างๆ ของเครื่องแม่ข่ายเว็บอีกทั้งระบบนี้ใช้ Thrift เป็นตัวกลางในเชื่อมต่อระหว่าง Cloudera Impala และเว็บไซต์ PHP อีกทั้งในวิจัยนี้ได้ทำการทดลองวัดผลการสืบค้นข้อมูลจราจรขนาดต่างๆ ตั้งแต่ขนาด 10G ถึง 50G พร้อมกับการเพิ่มจำนวนเครื่องคอมพิวเตอร์ที่ใช้ในการทดสอบจาก 3 เครื่องถึง 8 เครื่องเพื่อเปรียบเทียบเวลาที่ใช้ในการสืบค้นข้อมูลความเร็ว และค่าประสิทธิภาพให้สามารถเห็นผลลัพธ์ได้อย่างชัดเจน ซึ่งพบว่าในส่วนของความเร็วในการสืบค้นข้อมูลในรูปแบบ Parquet ใช้เวลาในการสืบค้นข้อมูลน้อยกว่าข้อมูลในรูปแบบ Text และเมื่อมีการเพิ่มจำนวนเครื่องคอมพิวเตอร์ที่ทำหน้าที่เป็น DataNode มากขึ้นเวลาที่ใช้ในการสืบค้นข้อมูลจะลดลง

โดยระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala นี้สามารถทำงานได้ครบถ้วนตามวัตถุประสงค์และขอบเขตของการวิจัย

5.2 ปัญหาและแนวทางแก้ไขปัญหา

5.2.1 หาขนาดข้อมูล Log File จริงขนาดใหญ่เพื่อใช้ทดสอบยากและขนาดข้อมูลมีจำกัด

แนวทางแก้ไข ให้ทำการรวบรวมเพื่อให้ได้ปริมาณมากขึ้นโดยการทำซ้ำข้อมูลเดิมเพื่อให้ได้ขนาดที่ใหญ่ขึ้น

5.2.2 รูปแบบของ Access log ไม่ตรงกับความต้องการและมีปัญหาในการตัดข้อความไม่ได้

แนวทางแก้ไข ให้จัดรูปแบบ Access log ใหม่ให้ตรงกับความต้องการ

5.2.3 การกำหนดรูปแบบ Error log ใน Apache เวอร์ชัน 2.2 ไม่ได้ เนื่องจากการกำหนดรูปแบบ Error log ได้เฉพาะ Apache เวอร์ชั่น 2.4 ขึ้นไป

แนวทางแก้ไข ให้ update เวอร์ชันของ Apache จาก 2.2 เป็น 2.4

5.2.4 ถ้าเกิดมีข้อมูลใหม่ใน HDFS ข้อมูลในตาราง Impala จะไม่เปลี่ยนแปลงเนื่องจากข้อมูลใน Impala อ้างอิงกับกับข้อมูล HDFS

แนวทางแก้ไข ให้ใช้คำสั่ง Refresh ตารางก่อนทุกครั้งเมื่อมีข้อมูลมาใหม่

5.2.5 หน้าจัดการ Host Management และ User Manager ไม่สามารถเก็บข้อมูลลง Cloudera Impala ได้เนื่องจาก Cloudera Impala ไม่รองรับคำสั่ง ลบ, แก้ไข ข้อมูลในตาราง

แนวทางแก้ไข ให้ใช้ MySQL แทน Cloudera Impala เฉพาะหน้าจัดการ Host Management และ User Manager

5.2.6 Apache Flume ฝั่ง Web Server ส่งข้อมูลจราจรไปยัง Apache Flume ฝั่ง Hadoop ไม่ได้เนื่องจากติดระบบ Firewall ของฝั่ง Hadoop

แนวทางแก้ไข ให้จัดการระบบ Firewall ของเครื่อง Hadoop

5.3 ข้อเสนอแนะและแนวทางพัฒนา

5.3.1 ระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala นี้ยังสามารถวิเคราะห์ข้อมูลจราจรได้มากกว่านี้

5.3.2 ระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala นี้ยังสามารถเพิ่มเครื่องได้มากกว่านี้

5.3.3 นำระบบวิเคราะห์ข้อมูลจราจรเว็บเซิร์ฟเวอร์บนพื้นฐานการเก็บข้อมูลแบบ HDFS โดยใช้ Impala นี้ สามารถนำไปใช้กับระบบวิเคราะห์อื่นๆ ได้

เอกสารอ้างอิง

ภาษาอังกฤษ

Cloudera, Inc. **Cloudera Impala Release Notes.** [Online]. Available form: http://www.cloudera.com/content/cloudera-content/cloudera-docs/Impala/latest /Cloudera-Impala-Release-Notes/Cloudera-Impala-Release-Notes.html. (20 June 2013).

The Apache Software Foundation. **HDFS Architecture Guide.** [Online]. Available form: http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html (25 June 2013)**.**

The Apache Software Foundation. **Apache Flume**. [Online]. Available from: http://flume.apache.org (27 June 2013).

The Apache Software Foundation. **Apache Thrift.** [Online]. Available form: http://thrift.apache.org/ (7 July 2013).

MDO. **Bootstrap.** [Online]. Available form: http://getbootstrap.com/customize (10 August 2013)

W3Schools**. CSS Tutorial.** [Online]. Available form: http://www.w3schools.com/css/default.asp. (2 January 2014).