ระบบแฟ้มข้อมูลรวมบนเว็บเซอร์วิสแบบเรสฟูล

นายวิชชา รตินิมิตธรรม

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2554

UNIFICATION FILE SYSTEM USING RESTFUL WEB SERVICE

Mr. Witcha Ratinimittum

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Engineering Program in Computer Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2011

หัวข้อวิทยานิพนธ์	ระบบแฟ้มข้อมูลรวมบนเว็บเซอร์วิสแบบเรสฟูล
โดย	นายวิชชา รตินิมิตธรรม
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

…….. คณบดีคณะวิศวกรรมศาสตร์
(รองศาสตราจารย์ ดร.บุญสม เลิศหิรัญวงศ์)

คณะกรรมการสอบวิทยานิพนธ์

……. ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. ณัฐวุฒิ หนูไพโรจน์)

……... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(ผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา)

……. กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. กุลธิดา โรจน์วิบูลย์ชัย)

……. กรรมการภายนอกมหาวิทยาลัย
(ดร. พงศ์ธวัช ชีพพิมลชัย)

วิชชา รตินิมิตธรรม : ระบบแฟ้มข้อมูลรวมบนเว็บเซอร์วิสแบบเรสฟูล. (UNIFICATION FILE SYSTEM USING RESTFUL WEB SERVICE) อ. ที่ปรึกษาวิทยานิพนธ์หลัก : ผศ. ดร. เกริก ภิรมย์โสภา, 51 หน้า.

อุปกรณ์เก็บข้อมูลภายนอกเป็นสิ่งที่มักจะใช้สำหรับการโอนถ่ายข้อมูลระหว่างเครื่องเมื่อผู้ใช้ต้องทำงานกับคอมพิวเตอร์หลายเครื่อง อย่างไรก็ตาม การโอนถ่ายข้อมูลผ่านทางพื้นที่เก็บข้อมูลออนไลน์จะมีความสะดวกมากกว่าการพกพาอุปกรณ์เหล่านั้น จุดประสงค์ของการวิจัยนี้คือการออกแบบและพัฒนาระบบแฟ้มข้อมูลรวมสำหรับการจัดการแฟ้มข้อมูลข้ามเครื่อง โดยเลือกใช้เว็บเซิร์ฟเวอร์เป็นพื้นที่สำหรับเก็บข้อมูลแบบออนไลน์ ระบบมีความสามารถในการขยายระบบ โดยการรองรับเว็บเซิร์ฟเวอร์ได้หลายตัวในคราวเดียวกัน ไคลเอนต์ติดต่อกับเซิร์ฟเวอร์แต่ละตัวผ่านทางส่วนต่อประสานแบบเรสฟูลเว็บเซอร์วิส แฟ้มข้อมูลภายในระบบจะถูกเก็บอยู่ในรูปของบล็อกข้อมูลหลายชิ้นเพื่อใช้ประโยชน์จากพื้นที่ว่างและแบนด์วิดท์ที่มีให้มากที่สุด นอกจากนี้ยังรองรับการแคชข้อมูลซึ่งประยุกต์ใช้อัลกอริทึมแอลอาร์ยู ประสิทธิภาพของระบบซึ่งวัดด้วยเครื่องมือวัดเปรียบเทียบสมรรถนะไอโอโซนบ่งบอกว่าระบบมีประสิทธิภาพอยู่ในเกณฑ์ที่ยอมรับได้

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต.....

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก.....

ปีการศึกษา 2554

5370489421 : MAJOR COMPUTER ENGINEERING
KEYWORDS : FILE SYSTEM / UNIFICATION FILE SYSTEM / RESTFUL WEB SERVICE / LRU

WITCHA RATINIMITTUM : UNIFICATION FILE SYSTEM USING RESTFUL WEB SERVICE. ADVISOR : ASST.PROF. KRERK PIROMSOPA, Ph.D., 51 pp.

Working with multiple machines, an external storage is usually a typical device for synchronizing data across machines. An online storage, however, provides more convenient than carrying devices. The purpose of this study is to design and implement unification file system for conveniently managing files across machines and utilizing available web servers as online storage. The proposed system provides scalability through supporting multiple web servers. Client communicates with each server using RESTful web services interfaces. Files will be kept as several data blocks in order to leverage available free space and gain more aggregation bandwidth. Moreover, cache mechanism based on Least Recently Used replacement algorithm is supported in the system as well. Evaluated by using IOzone file system benchmark, the performance suggests that the proposed system is promising.

Department : Computer Engineering Student's Signature .....
Field of Study : Computer Engineering Advisor's Signature .....
Academic Year : 2011

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงไปได้ด้วยความอนุเคราะห์อย่างยิ่งของผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา อาจารย์ที่ปรึกษา ซึ่งท่านได้ให้ความรู้ คำแนะนำทั้งในด้านงานวิจัยและด้านอื่น ๆ และให้ความช่วยเหลือ คอยชี้แนะและช่วยแก้ปัญหาในด้านต่าง ๆ เป็นอย่างดี จนทำให้การวิจัยในครั้งนี้สำเร็จลุล่วง

ขอขอบพระคุณ ผู้ช่วยศาสตราจารย์ ดร. ณัฐวุฒิ หนูไพโรจน์ ผู้ช่วยศาสตราจารย์ ดร. กุลธิดา โรจน์วิบูลย์ชัย และ ดร. พงศ์ธวัช ชีพพิมลชัย กรรมการสอบวิทยานิพนธ์ ที่กรุณาเสียสละเวลาให้คำแนะนำ ตรวจสอบ และแก้ไขวิทยานิพนธ์ฉบับนี้

ท้ายที่สุด ขอขอบคุณเพื่อนและรุ่นพี่ทุก ๆ คน รวมทั้งครอบครัว ที่คอยให้คำปรึกษา การสนับสนุน และกำลังใจ รวมถึงท่านอื่นที่มิได้กล่าวชื่อไว้ ณ ที่นี้ที่มีส่วนช่วยให้วิทยานิพนธ์สำเร็จได้ด้วยดี

สารบัญ

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

ในบทนี้แบ่งเนื้อหาออกเป็น 8 หัวข้อย่อย กล่าวถึงความเป็นมา และความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย คำจำกัดความที่ใช้ในการวิจัย ประโยชน์ที่คาดว่าจะได้รับ วิธีดำเนินการวิจัย ลำดับขั้นตอนในการเสนอผลการวิจัย และผลงานตีพิมพ์จากวิทยานิพนธ์ ตามลำดับ ดังนี้

1.1 ความเป็นมาและความสำคัญของปัญหา

ปัจจุบันผู้ใช้ทั่วไปมักจะทำงานกับคอมพิวเตอร์หลายเครื่อง ซึ่งข้อมูลที่ใช้เป็นประจำมักถูกบันทึกบนฮาร์ดดิสก์ภายในเครื่องหลักของผู้ใช้เอง แต่เมื่อผู้ใช้ต้องย้ายไปทำงานที่อีกเครื่อง จึงหลีกเลี่ยงไม่ได้ที่จะต้องถ่ายโอนข้อมูลจากเครื่องหลักไปสู่เครื่องอื่น วิธีการถ่ายโอนข้อมูลนั้นมีหลายวิธี ผู้ใช้อาจจะถ่ายโอนข้อมูลด้วยอุปกรณ์เก็บข้อมูลแบบพกพา เช่น แฟลชไดรฟ์ (flash drive) หรือฮาร์ดดิสก์ภายนอก (external hard disk) ซึ่งเป็นวิธีที่นิยมมากในปัจจุบัน แต่อาจไม่สะดวกต่อการพกพาถ้ามีอุปกรณ์เหล่านี้หลายตัว และเนื่องด้วยขนาดที่เล็ก จึงง่ายต่อการสูญหาย นอกจากนี้ผู้ใช้อาจโอนถ่ายข้อมูลไปที่อีกเครื่องผ่านทางเครือข่ายแลน (local area network, LAN) แต่จะใช้ได้เฉพาะในกรณีที่เครื่องทั้งสองอยู่ในวงแลนเดียวกัน หรืออีกวิธีคือ การฝากข้อมูลไว้กับบริการพื้นที่เก็บข้อมูลออนไลน์ โดยข้อดีของบริการเหล่านี้คือไม่ต้องพกพาอุปกรณ์ใดเพิ่มเติมก็สามารถเข้าถึงข้อมูลจากที่ต่าง ๆ ได้เพียงมีอินเทอร์เน็ตให้ใช้งาน

สำหรับบริการพื้นที่เก็บข้อมูลออนไลน์ที่โดดเด่นมากที่สุดบริการหนึ่งในปัจจุบัน คือ Amazon S3 [1] ซึ่งให้บริการพื้นที่เก็บข้อมูลแบบกลุ่มเมฆ (cloud storage) โดยคิดค่าใช้จ่ายจริงตามปริมาณการใช้งาน บริการนี้สามารถเพิ่มลดทรัพยากรในระบบให้สอดคล้องกับปริมาณที่ผู้ใช้ใช้งานโดยอัตโนมัติ จึงช่วยประหยัดค่าใช้จ่ายกว่าบริการของรายอื่นที่คิดค่าบริการแบบเหมาจ่าย Amazon S3 มีส่วนต่อประสานแบบเว็บเซอร์วิส (web services interfaces) ให้ผู้ใช้ใช้งานผ่านทางเว็บ หรือผ่านแอพลิเคชันที่พัฒนาขึ้นมาเพื่อใช้งานกับส่วนต่อประสานนี้ และด้วยบริการในลักษณะเว็บเซอร์วิสนี่เอง จึงส่งผลให้มีผู้ให้บริการเพิ่มขึ้นอีกมากมายที่พัฒนาบริการของตนเองบน Amazon S3 เช่น Dropbox [2] เป็นต้น Dropbox เป็นบริการพื้นที่เก็บข้อมูลออนไลน์ที่มีความสามารถในการซิงโครไนซ์ (synchronize) ข้อมูลข้ามเครื่องโดยอัตโนมัติ ถ้าผู้ใช้บริการมีคอมพิวเตอร์หลายเครื่อง แต่ละเครื่องก็จะมีข้อมูลชุดเดียวกันเครื่องละชุด จึงอาจทำให้เปลืองพื้นที่

นอกจากนี้ยังมีบริการประเภทเว็บเซิร์ฟเวอร์ (web server) ซึ่งเป็นบริการที่ให้พื้นที่สำหรับใช้ในการสร้างเว็บไซต์ มีทั้งแบบฟรีและเสียค่าใช้จ่าย ผู้ใช้สามารถเข้าถึงแฟ้มข้อมูลที่ฝากไว้กับเซิร์ฟเวอร์ประเภทนี้ผ่านทางเว็บเบราเซอร์โดยกำหนดยูอาร์แอล (URL) สำหรับแฟ้มข้อมูลที่ต้องการได้

บริการพื้นที่เก็บข้อมูลออนไลน์ที่กล่าวไปข้างต้นมักจะไม่ประสานงานร่วมกับระบบแฟ้มข้อมูลบนเครื่องอย่างสมบูรณ์ ดังนั้น ผู้ใช้อาจจะไม่สะดวกเมื่อต้องจัดการหรือแก้ไขข้อมูลเหล่านั้น เพราะต้องดาวน์โหลดมาไว้บนเครื่องเสียก่อน นอกจากนี้บริการส่วนใหญ่มักจะให้พื้นที่ฟรีในขนาดที่จำกัด ถ้าใช้งานพื้นมากกว่านั้นต้องเสียค่าใช้จ่ายเพิ่มเติม แต่สำหรับเว็บเซิร์ฟเวอร์จะต่างออกไป เนื่องจากผู้ใช้สามารถติดตั้งสคริปต์ที่พัฒนาด้วยภาษาพีเอชพี (PHP) หรือไพธอน (Python) เพื่อเพิ่มความสามารถอื่นได้ เช่น การเพิ่มส่วนต่อประสานแบบเว็บเซอร์วิส เป็นต้น ดังนั้นจึงมีความเป็นไปได้ที่จะพัฒนาพื้นที่เก็บข้อมูลออนไลน์ส่วนตัวทีเกิดจากการรวมตัวกันของเว็บเซิร์ฟเวอร์หลายตัวได้ โดยเซิร์ฟเวอร์ที่ใช้อาจเป็นแบบฟรีหรือเสียค่าใช้จ่ายก็ได้

โครงการวิจัยนี้จึงออกแบบ และพัฒนาระบบแฟ้มข้อมูลอาร์เอฟเอส (RFS) ซึ่งเป็นระบบแฟ้มข้อมูลที่ทำหน้าที่เป็นตัวกลางในการจัดการแฟ้มข้อมูลของผู้ใช้ที่อยู่บนพื้นที่เก็บข้อมูลออนไลน์ส่วนตัวที่เกิดจากการรวมกลุ่มกันของเว็บเซิร์ฟเวอร์หลายตัว ระบบนี้ใช้สถาปัตยกรรมแบบไคลเอนต์-เซิร์ฟเวอร์ ทางฝั่งเซิร์ฟเวอร์เลือกใช้เว็บเซิร์ฟเวอร์ที่รองรับภาษาพีเอชพี เนื่องจากแพร่หลายมากที่สุด แล้วติดตั้งสคริปต์พีเอชพีเพื่อเพิ่มส่วนต่อประสานแบบเรสฟูลเว็บเซอร์วิส (RESTful web services interfaces) ให้กับเซิร์ฟเวอร์ โดยเว็บเซิร์ฟเวอร์ในระบบจะแบ่งออกเป็น 2 ประเภท คือ เมตาเดตาโหนด (metadata node) และสตอร์เรจโหนด (storage node) ซึ่งทำหน้าที่บันทึกเมตาเดตา (Metedata) และเก็บแฟ้มข้อมูล (file) ตามลำดับ ส่วนแอพลิเคชันทางฝั่งไคลเอนต์ หรืออาร์เอฟเอสไคลเอนต์ (RFS client) จะติดต่อกับเว็บเซิร์ฟเวอร์ผ่านทางส่วนต่อประสานแบบเรสฟูลเว็บเซอร์วิสด้วยโพรโตคอลเอชทีทีพี (HTTP) ระบบแฟ้มข้อมูลอาร์เอฟเอสรองรับการขยายตัวของระบบ (scalability) โดยการเพิ่มเว็บเซิร์ฟเวอร์ที่ใช้ภายในระบบ ส่งผลให้ระบบรองรับปริมาณข้อมูลเพิ่มขึ้นได้ นอกจากนี้ยังรองรับการแคชแฟ้มข้อมูลที่ใช้งานเป็นประจำไว้ด้วย

1.2 วัตถุประสงค์ของการวิจัย

การวิจัยมีวัตถุประสงค์ ดังนี้

1. เพื่อศึกษาระบบแฟ้มข้อมูลที่มีอยู่ วิเคราะห์หาข้อดีข้อเสีย แล้วนำข้อดีของระบบเหล่านั้นมาใช้

2. เพื่อออกแบบ และพัฒนาระบบแฟ้มข้อมูลรวมที่มีประสิทธิภาพ และมีความสามารถในการขยายระบบบนเว็บเซอร์วิสแบบเรสฟูล
3. เพื่อวัด และเปรียบเทียบประสิทธิภาพของระบบที่ออกแบบไว้กับระบบแฟ้มข้อมูลอื่น

1.3 ขอบเขตของการวิจัย

ขอบเขตของการวิจัย มีดังต่อไปนี้

1. สร้างและทดสอบระบบแฟ้มข้อมูลรวมบนระบบปฏิบัติในตระกูลลินุกซ์เท่านั้น
2. รองรับเฉพาะโพรโตคอลเอชทีทีพี (HTTP) หรือเอชทีทีพีเอส (HTTPS) เท่านั้น
3. ระบบที่ออกแบบเหมาะสำหรับการใช้งานโดยทั่วไปและสำหรับผู้ใช้คนเดียวเท่านั้น
4. เซิร์ฟเวอร์ที่ใช้ต้องรองรับภาษาพีเอชพี

1.4 คำจำกัดความที่ใช้ในการวิจัย

คำจำกัดความที่ใช้ในการวิจัย มีดังต่อไปนี้

1. บล็อกข้อมูล (data block) หมายถึง ส่วนประกอบย่อยที่ถูกแบ่งออกมาจากแฟ้มข้อมูลดั้งเดิม โดยบล็อกข้อมูลของแต่ละแฟ้มข้อมูลจะมีขนาดเล็กลงและอาจมีหลายชิ้น
2. ระบบแฟ้มข้อมูล (file system) หมายถึง ระบบจัดการข้อมูล ซึ่งเป็นข้อมูลที่ผู้ใช้อาจต้องการเข้าถึงอีกครั้งหลังจากปิดโปรแกรมไปแล้ว โดยระบบแฟ้มข้อมูลมีส่วนต่อประสานเพื่อใช้ในการสร้าง บันทึก แก้ไขข้อมูลของแฟ้มข้อมูล และไดเรคทอรี (directory)
3. โพรโตคอลเอชทีทีพี (HTTP) หมายถึง โพรโตคอลชนิดหนึ่งที่เป็นมาตรฐานในการร้องขอ และการตอบรับระหว่างไคลเอนต์และเซิร์ฟเวอร์ โดยโพรโตคอลเอชทีทีพีมักใช้สำหรับการติดต่อกับเว็บเซิร์ฟเวอร์ผ่านทางเว็บเบราว์เซอร์ หรือเครื่องมืออื่น เช่น ไลบรารีเคิร์ล (cURL library) เป็นต้น

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ประโยชน์ที่คาดว่าจะได้รับ มีดังนี้

1. เข้าใจการทำงานของระบบแฟ้มข้อมูลรวม
2. เข้าใจข้อดีข้อเสียของระบบแฟ้มข้อมูลรวม และการประยุกต์ใช้กับเว็บเซอร์วิสแบบเรสฟูล

3. ได้ต้นแบบของระบบแฟ้มข้อมูลรวมบนเว็บเซอร์วิสแบบเรสฟูลที่มีความสามารถในการขยายระบบ

1.6 วิธีดำเนินการวิจัย

วิธีดำเนินการวิจัยถูกแบ่งออกเป็น 5 ขั้นตอน ดังนี้

1. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง เช่น ระบบแฟ้มข้อมูลแบบกระจาย เว็บเซอร์วิสในรูปแบบต่าง ๆ และงานวิจัยที่เกี่ยวข้องกับระบบแฟ้มข้อมูลรวม เป็นต้น
2. ศึกษาเครื่องมือที่ใช้ในการวิจัย เช่น ทดลองพัฒนาระบบแฟ้มข้อมูลอย่างง่ายโดยใช้ไลบรารีฟิวส์ (fuse library) และทดสอบการวัดประสิทธิภาพของระบบแฟ้มข้อมูลด้วยเครื่องมือวัดเปรียบเทียบสมรรถนะ (IOzone benchmark tool)
3. ออกแบบและสร้างระบบตัวต้นแบบ
4. ทดสอบความสามารและประสิทธิภาพของระบบตัวต้นแบบที่สร้างขึ้น
5. สรุปผลการวิจัยและเรียบเรียงวิทยานิพนธ์

1.7 ลำดับขั้นตอนในการเสนอผลการวิจัย

วิทยานิพนธ์นี้มีเนื้อหาทั้งหมด 5 บท ได้แก่ บทนำ กล่าวถึงความเป็นมา ความสำคัญของปัญหา และวัตถุประสงค์ของการวิจัย บทที่ 2 กล่าวถึงทฤษฎีและงานวิจัยที่เกี่ยวข้อง บทที่ 3 กล่าวถึงหลักการและแนวคิดในการออกแบบระบบ บทที่ 4 กล่าวถึงรายละเอียดในการพัฒนาระบบต้นแบบของระบบแฟ้มข้อมูลรวมบนเว็บเซอร์วิสแบบเรสฟูล บทที่ 5 กล่าวถึงวิธีประเมินการวิจัย สรุป และการอภิปรายผล บทที่ 6 กล่าวถึงบทสรุปของการวิจัยนี้

1.8 ผลงานตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตอบรับให้ตีพิมพ์เป็นบทความทางวิชาการในหัวข้อเรื่อง “An Implementation of RESTful-based Scalable File System” โดยนายวิชชา รตินิมิตธรรม และผู้ช่วยศาสตราจารย์ ดร. เกริก ภิรมย์โสภา, ในงานประชุมวิชาการ “The Ninth International Joint Conference on Computer Science and Software Engineering (JCSSE 2012)” ณ มหาวิทยาลัยหอการค้าไทย จังหวัดกรุงเทพมหานคร ระหว่างวันที่ 30 พฤษภาคม - 1 มิถุนายน 2555

บทที่ 2
ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

ในบทนี้กล่าวถึงแนวคิด ทฤษฎี และงานวิจัยที่เกี่ยวข้อง ดังนี้

2.1 แนวคิดและทฤษฎี

ในส่วนนี้จะกล่าวถึงระบบแฟ้มข้อมูลที่เป็นพื้นฐานของระบบที่ได้ออกแบบไว้ในการวิจัยนี้ รวมถึงอัลกอริธึมการแทนที่ และเว็บเซอร์วิสประเภทต่าง ๆ

2.1.1 ระบบแฟ้มข้อมูลแบบกระจาย (distributed file system)

ระบบแฟ้มข้อมูลแบบกระจาย คือ ระบบกระจายที่เกิดจากการรวมกลุ่มของระบบแฟ้มข้อมูลหลาย ๆ ระบบภายในเครือข่ายเข้าด้วยกัน ทำให้กลายเป็นระบบแฟ้มข้อมูลที่มีขนาดใหญ่ ผู้ใช้สามารถเข้าถึงแฟ้มข้อมูลที่กระจายอยู่ตามระบบแฟ้มข้อมูลต่าง ๆ ได้โดยสะดวก โดยมีระบบแฟ้มข้อมูลแบบกระจายที่เกี่ยวข้อง ดังนี้

2.1.1.1 ระบบแฟ้มข้อมูลแอนดรูว์ (Andrew file system, AFS)

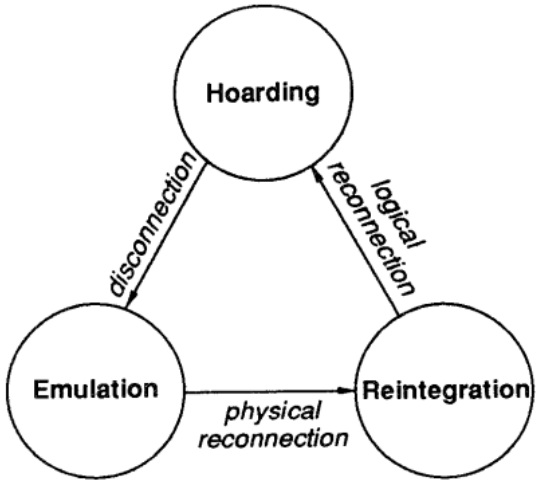
ระบบแฟ้มข้อมูลแอนดรูว์ คือระบบแฟ้มข้อมูลแบบกระจายที่พัฒนาโดยนักวิจัยของมหาวิทยาลัยคาร์เนกีเมลลอน (Carnegie Mellon University) ในช่วงคริสต์ทศวรรษ 1980 โดยมีเป้าหมายหลักคือ รองรับการทำงานของผู้ใช้จานวนมหาศาลได้ โดยการผลักภาระส่วนใหญ่ให้ไคลเอนต์ทำ ซึ่งจะมีการถ่ายโอนข้อมูลมาเก็บไว้เป็นแคช (cache) ในเครื่องของผู้ใช้ สำหรับการจัดการทางด้านความปลอดภัย มีการใช้โพรโตคอลเคอร์เบอรอส (Kerberos) สำหรับการพิสูจน์ตัวตนของของไคลเอนต์กับเซิร์ฟเวอร์

2.1.1.2 ระบบแฟ้มข้อมูลโคดา (Coda file system) [3]

ระบบแฟ้มข้อมูลโคดาถูกพัฒนาต่อจากระบบแฟ้มข้อมูลแอนดรูว์ ซึ่งถูกออกแบบมาเพื่อลดปัญหาที่มักเกิดกับเครื่องไคลเอนต์เมื่อมีการตัดการเชื่อมต่อกับระบบ ด้วยความตั้งใจหรือไม่ตั้งใจก็ตาม ระบบแฟ้มข้อมูลโคดามีคุณสมบัติที่สำคัญ 2 ประการที่ทำงานร่วมกันคือการทำสำเนาข้อมูล (server replication) และการทำงานในขณะที่ตัดการเชื่อมต่อกับระบบ (disconnected operation) ซึ่งทั้งสองคุณสมบัตินี้ทำให้ระบบมีสภาพความพร้อมใช้งานสูง (high availability)

ในระหว่างการทำงานตามปกติซึ่งเป็นช่วงกักตุน (hoarding) เครื่องทางฝั่งไคลเอนต์จะดึงข้อมูลที่ต้องใช้จากเซิร์ฟเวอร์มาเก็บเป็นแคชไว้ เมื่อถูกตัดการเชื่อมต่อจากระบบ ระบบในเครื่องไคลเอนต์จะเปลี่ยนสถานะจากช่วงกักตุนสู่ช่วงจำลอง (emulation) ซึ่งเป็นช่วงที่เสมือนว่าเครื่องของไคลเอนต์ยังเชื่อมต่อกับระบบอยู่ และสามารถทำงานต่อได้ตามปกติโดยใช้ข้อมูลที่แคชไว้ และบันทึกการเปลี่ยนแปลงต่าง ๆ ลงแฟ้มข้อมูลล็อก (log file) ไว้ เมื่อเชื่อมต่อกับระบบได้อีกครั้ง ก็จะเปลี่ยนสถานะจากช่วงจำลองสู่ช่วงผสมผสาน (reintegration) ซึ่งเป็นช่วงที่ข้อมูลที่เปลี่ยนแปลงทั้งหมดจะถูกส่งไปปรับปรุงข้อมูลที่เซิร์ฟเวอร์ให้เป็นปัจจุบัน แล้วก็จะกลับมาสู่ช่วงกักตุนอีกครั้งหนึ่งดังแสดงในภาพที่ 1

สำหรับการทำสำเนาข้อมูล ระบบแฟ้มข้อมูลโคดาจะใช้นโยบายการทำสำเนาข้อมูลแบบอ็อพทิมิสทิก (optimistic replication strategy) ซึ่งเซิร์ฟเวอร์ยอมให้ผู้ใช้อื่นอัพเดทข้อมูลที่เปลี่ยนแปลงได้ตลอดเวลาที่ยังอยู่ในเครือข่ายถ้ามีการอัพเดทข้อมูลเดียวกันจากผู้ใช้หลายคน ระบบก็มีเครื่องมือที่ใช้จัดการความขัดแย้งของข้อมูล (data conflict) ด้วย



ภาพที่ 1 แสดงสถานะภายในระบบแฟ้มข้อมูลแบบกระจายโคดา

2.1.1.3 **ระบบแฟ้มข้อมูลโมไจล์** (Mogile file system) [4]

โมไจล์เป็นระบบแฟ้มข้อมูลแบบกระจายที่เน้นความยืดหยุ่นของระบบ และสภาพพร้อมใช้งานที่สูง ระบบแฟ้มข้อมูลโมไจล์มักใช้เป็นระบบเก็บข้อมูลให้กับแอพลิเคชัน หรือบริการอื่น ๆ เพราะมีความสามารถในการทำสำเนาข้อมูล และเหมาะกับงานที่ต้องมีการดึงข้อมูลจำนวนมากจากเซิร์ฟเวอร์

ลักษณะเด่นของระบบแฟ้มข้อมูลโมไจล์คือทำงานในระดับผู้ใช้ (user space) สามารถใช้งานร่วมกับระบบแฟ้มข้อมูลที่มีอยู่ได้ ป้องกันปัญหาความล้มเหลวแบบจุดเดี่ยว (single point of failure) ได้ เพราะรองรับเซิร์ฟเวอร์ซึ่งทำหน้าที่ต่างกัน 3 ชนิด คือ สตอร์เรจเซิร์ฟเวอร์ (storage server) แทรคเกอร์เซิร์ฟเวอร์ (tracker server) เดตาเบสเซิร์ฟเวอร์ (database server) ได้คราวละหลายตัว เนมสเปซของระบบแฟ้มโมไจล์มีลักษณะเป็นโกลบอลเนมสเปซ (global namespace) ซึ่งเพิ่มเนมสเปซได้มากเท่าที่ต้องการ นอกจากนี้ยังมีการใช้ระบบจัดการฐานข้อมูลมายเอสคิวแอล (MySQL) ในการเก็บและจัดการเมตาเดตา (metadata) ของระบบ และใช้โพรโตคอลเอชทีทีพี (HTTP) ในการถ่ายโอนข้อมูล ข้อจำกัดของระบบแฟ้มข้อมูลโมไจล์ คือยังไม่สนับสนุนมาตรฐานโพสอิกซ์ (POSIX) ทำให้ไม่สามารถนำมาใช้งานได้เหมือนกับระบบแฟ้มข้อมูลทั่วไปได้ทันที

2.1.2 **อัลกอริทึมการแทนที่** (Replacement Algorithm)

อัลกอริทึมการแทนที่หรือแคชอัลกอริทึม (cache algorithm) คืออัลกอริทึมที่ใช้จัดการกับแคชของข้อมูลที่เก็บอยู่บนเครื่องคอมพิวเตอร์ เมื่อพื้นที่สำหรับเก็บแคชบนเครื่องเต็มหรือมีขนาดถึงจุดที่กำหนดไว้แล้ว จะเป็นหน้าที่ของอัลกอริทึมการแทนที่ในการเลือกว่าจะลบแคชของข้อมูลตัวใดออกไป เพื่อให้มีพื้นที่ว่างสำหรับเก็บแคชของข้อมูลอื่น ๆ ยกตัวอย่างของอัลกอริทึมการแทนที่ เช่น แอลอาร์ยู (least recently used, LRU) คืออัลกอริทึมที่จะลบแคชของข้อมูลที่ไม่ได้ใช้เป็นเวลานานหรือมีอายุมากที่สุดในแคชออกไป หลักการทำงานของแอลอาร์ยู มาจากแนวคิดที่ว่าข้อมูลที่มีการใช้งานล่าสุดมีแนวโน้มที่จะถูกใช้งานอีกในอนาคต จึงเหมาะกับงานที่ข้อมูลใหม่มีโอกาสถูกเรียกใช้บ่อย เอ็มอาร์ยู (most recently used, MRU) คืออัลกอริทึมที่ทำงานตรงข้ามกับแอลอาร์ยูโดยจะลบแคชของข้อมูลที่ถูกใช้งานล่าสุดหรือมีอายุน้อยที่สุดในแคชออกไป เหมาะกับงานที่มักจะมีการเรียกข้อมูลที่มีอยู่ในแคชอยู่แล้วหรือข้อมูลที่มีอายุมาก

2.1.3 **เว็บเซอร์วิส** (Web Services)

เว็บเซอร์วิสเป็นสถาปัตยกรรมสำหรับการพัฒนาแอพพลิเคชันในลักษณะไคลเอนต์-เซิร์ฟเวอร์ โดยเสนอวิธีการมาตรฐานสำหรับการติดต่อสื่อสารกันผ่านโพรโตคอลเอชทีทีพี (HTTP) ระหว่างโปรแกรมที่แตกต่างกัน หรือระบบที่ทำงานอยู่บนแพลตฟอร์มที่ต่างกัน ไคลเอนต์จะเป็นผู้ที่ส่งคำร้องขอ (request) โดยการส่งข้อความไปที่ยูอาร์แอล (URL) ของเซิร์ฟเวอร์ที่เปิดให้บริการส่วนทางฝั่งเซิร์ฟเวอร์ซึ่งมีส่วนต่อประสาน (application programming interface, API) สำหรับให้ไคลเอนต์เรียกใช้บริการ จะทำงานตามคำร้องขอที่ได้รับจากไคลเอนต์ แล้วส่งคำตอบกลับ

(response) พร้อมทั้งผลลัพธ์กลับไปที่ไคลเอนต์ ข้อความที่โต้ตอบกันจะอยู่ในรูปแบบที่เป็นมาตรฐาน เช่น เอ็กซ์เอ็มแอล (XML) และเจสัน (JSON) เป็นต้น การพัฒนาแอพพลิเคชันบนไคลเอนต์ และเซิร์ฟเวอร์จะใช้ภาษาหรือวิธีการใดก็ได้ตราบใดที่ยังคงรูปแบบมาตรฐานสำหรับการติดต่อสื่อสารไว้ ด้วยเหตุนี้เองเว็บเซอร์วิสจึงเพิ่มความสะดวกสำหรับการติดต่อสื่อสารกันระหว่างโปรแกรมที่เขียนด้วยภาษาที่แตกต่างกัน หรือระบบที่ทำงานอยู่บนแพลตฟอร์มที่แตกต่างกัน เว็บเซอร์วิสยังถือว่ามีแนวคิดเดียวกับสถาปัตยกรรมเชิงบริการ (service-oriented architecture, SOA) ซึ่งเปลี่ยนมุมมองของการพัฒนาแอพพลิเคชันจากการพัฒนาแอพลิเคชันทั้งหมด มาเป็นการพัฒนาแอพลิเคชันเพื่อเปิดให้บริการ แต่ละบริการจะมีหน้าที่เฉพาะตัว เราสามารถนำบริการต่าง ๆ มาประกอบกันเป็นแอพพลิเคชันใหม่หรือนำมาเพิ่มความสามารถให้กับแอพพลิเคชันที่มีอยู่เดิมได้ จะเห็นว่าแนวคิดนี้สนับสนุนการนำกลับมาใช้ใหม่ด้วย ซึ่งช่วยให้การพัฒนาแอพพลิเคชันทำได้รวดเร็วมากขึ้น เว็บเซอร์วิสในปัจจุบันมีวิธีการใช้งานที่หลากหลาย แต่ที่นิยมมากมีอยู่ 3 รูปแบบ ดังต่อไปนี้

2.1.3.1 **เอ็กซ์เอ็มแอลอาร์พีซี** (XML-RPC)

เป็นวิธีที่ใช้ข้อความในรูปแบบเอ็กซ์เอ็มแอล ในการติดต่อสื่อสารกันระหว่างไคลเอนต์และเซิร์ฟเวอร์ ทางฝั่งเซิร์ฟเวอร์จะเตรียมฟังก์ชันสำหรับให้บริการ ทางฝั่งไคลเอนต์สามารถเรียกใช้ฟังก์ชันทางไกล (remote procedure call) เหล่านั้นพร้อมทั้งส่งค่าพารามิเตอร์เหมือนกับการเขียนโปรแกรมตามปกติ แต่การทำงานหรือคำนวณของฟังก์ชันที่เรียกนั้นจะกระทำที่ฝั่งเซิร์ฟเวอร์ เมื่อทำงานเสร็จแล้วเซิร์ฟเวอร์จะส่งผลลัพธ์กลับมาเป็นข้อความเอ็กซ์เอ็มแอล ข้อดีของรูปแบบนี้คือความง่ายในการพัฒนาระบบ แต่อาจถือเป็นข้อเสียด้วยเพราะอาจไม่เหมาะกับงานที่ซับซ้อน

2.1.3.2 **โซพ** (SOAP)

เป็นโปรโตคอลที่ใช้ข้อความเอ็กซ์เอ็มแอลในการติดต่อสื่อสาร รูปแบบนี้พัฒนาต่อจากเอ็กซ์เอ็มแอลอาร์พีซี โครงสร้างของข้อความที่ใช้ติดต่อกันระหว่างไคลเอนต์และเซิร์ฟเวอร์จะมีความซับซ้อนมากกว่าของเอ็กซ์เอ็มแอลอาร์พีซี จึงช่วยให้โซพสามารถส่งข้อความที่มีรายละเอียดต่าง ๆ ได้มากกว่า โดยข้อความโซพ (SOAP message) นั้นจะอยู่ในซอง (envelope) ซึ่งภายในจะประกอบด้วยส่วนหัว (header) และส่วนเนื้อหา (body) ส่วนหัวเป็นส่วนที่มีหรือไม่มีก็ได้ ข้อมูลที่อยู่ในส่วนนี้ใช้สำหรับการให้ข้อมูลเพิ่มเติม เช่น หมายเลขข้อความ ชื่อผู้ใช้ รายละเอียดของกุญแจสาธารณะ เป็นต้น ในส่วนเนื้อหาจะเป็นข้อความที่ระบุฟังก์ชันพร้อมพารามิเตอร์ที่ต้องการเรียกใช้

จากเซิร์ฟเวอร์ หรืออาจเป็นผลลัพธ์ที่ได้จากเซิร์ฟเวอร์เมื่อมีการตอบกลับ นอกจากนี้โซพมักถูกใช้ร่วมกับเอกสารดับเบิ้ลยูเอสดีแอล (WSDL document) ซึ่งเอกสารนี้จะช่วยอธิบายว่า เมธอดหรือฟังก์ชันนั้นต้องการพารามิเตอร์อะไรและประเภทไหน และค่าที่ส่งกลับอยู่ในรูปแบบและประเภทใด ข้อดีของโซพคือมีความยืดหยุ่นสูงแต่จะมีความยุ่งยากซับซ้อนกว่าวิธีอื่น

2.1.3.3 **เรส** (REST)

เป็นรูปแบบหนึ่งของสถาปัตยกรรมซอฟต์แวร์ที่ใช้โพรโตคอลเอชทีทีพีในการติดต่อสื่อสารระหว่างไคลเอนต์และเซิร์ฟเวอร์ ข้อความที่ใช้ติดต่อสื่อสารกันอาจใช้เอ็กซ์เอ็มแอลเจสัน หรือเอชทีเอ็มแอล (HTML) หรืออาจเป็นอ็อบเจกต์อื่น ๆ ที่ระบบรองรับ สถาปัตยกรรมแบบเรสท์จะมองว่าทุกอย่างคือทรัพยากร (resource) ดังนั้นการใช้สถาปัตยกรรมแบบนี้จึงเน้นที่การเรียก เพิ่ม แก้ไข หรือลบทรัพยากรที่มีบนเซิร์ฟเวอร์ แทนที่จะเป็นการเรียกใช้ฟังก์ชันที่เซิร์ฟเวอร์ให้บริการเหมือนกับรูปแบบเอ็กซ์เอ็มแอลอาร์พีซีและโซพ ทรัพยากรแต่ละตัวของเว็บเซอร์วิสแบบเรสฟูล (RESTful web services) จะมีตัวบ่งชี้ประจำตัว (uniform resource identifier) หรือยูอาร์ไอ (URI) เพื่อที่ไคลเอนต์จะระบุได้ว่าต้องการเข้าถึงทรัพยากรตัวใด นอกจากนี้แต่ละทรัพยากรจะเข้าถึงได้ด้วยส่วนต่อประสานเดียวกัน (uniform interface) โดยใช้โพรโตคอลเอชทีทีพี และระบุชนิดของคำร้องขอด้วยเอชทีทีพีเมธอด (HTTP method) เช่น เก็ท (GET) โพสท์ (POST) พุท (PUT) ดีลีท (DELETE) เฮด (HEAD) ฯลฯ ซึ่งเมธอดเหล่านี้จะเป็นตัวบ่งบอกการกระทำที่เกิดขึ้นกับทรัพยากรนั้น ยกตัวอย่างเช่น การใช้เมธอดเก็ทโดยระบุยูอาร์ไอไปที่ http://example.com/resources/ จะเป็นการเรียกรายการทรัพยากรที่มีในเซิร์ฟเวอร์มาที่ไคลเอนต์ และถ้าเปลี่ยนเป็นเมธอดโพสท์โดยระบุยูอาร์ไอเดิม จะเป็นการเพิ่มทรัพยากรให้กับเซิร์ฟเวอร์ เป็นต้น

2.2 งานวิจัยที่เกี่ยวข้อง

เนื่องจากการวิจัยนี้คือการพัฒนาระบบแฟ้มข้อมูล งานวิจัยที่เกี่ยวข้องจึงเกี่ยวกับระบบแฟ้มข้อมูลต่าง ๆ ที่มีความคล้ายคลึงกับระบบในการวิจัยนี้นั่นเอง ซึ่งมีดังต่อไปนี้

2.2.1 Versatility and Unix Semantics in Namespace Unification [5]

งานวิจัยนี้กล่าวถึงระบบแฟ้มข้อมูลรวมยูเนียน (Union file system) ที่ถูกพัฒนาตามหลักการเดียวกับระบบแฟ้มข้อมูลรวม (unification file system) ระบบแฟ้มข้อมูลประเภทนี้จะ

ช่วยให้แฟ้มข้อมูลจากแต่ละที่เสมือนกับว่ามาอยู่ในที่ที่เดียวกัน โดยทั่วไปแล้ว ระบบแฟ้มข้อมูลรวมส่วนใหญ่จะมีหลักการพื้นฐาน 3 ประการ คือ

1. แบรนช์ (branch) และยูเนียน (union)

แบรนช์คือชื่อที่ใช้เรียกไดเรคทอรีหรือระบบแฟ้มข้อมูลที่ถูกนำมารวมกันที่ไดเรคทอรีหนึ่ง ซึ่งไดเรคทอรีที่แฟ้มข้อมูลอื่นมาอยู่รวมกันนั้นจะถูกเรียกว่ายูเนียน ในแต่ละแบรนช์ จะมีนโยบายการเข้าถึงเป็นของตัวเอง เช่น อ่านอย่างเดียว (read-only) หรืออ่านเขียน (read-write) เป็นต้น นอกจากนี้แบรนช์ยังมีการแยกลำดับที่ต่างกันด้วย ในกรณีที่แฟ้มข้อมูลมีชื่อเดียวกัน ผู้ใช้จะเห็นเฉพาะแฟ้มข้อมูลที่อยู่ในแบรนช์ที่มีลำดับสูงที่สุดเท่านั้น

2. ก็อปปีอัพ (copy up)

ในกรณีที่ต้องการแก้ไขแฟ้มข้อมูลที่อยู่ภายในแบรนช์ที่มีนโยบายอ่านอย่างเดียวนั้น ระบบแฟ้มข้อมูลรวมจะคัดลอกแฟ้มข้อมูลนั้นไปยังแบรนช์ที่มีนโยบายอ่านเขียน และมีลำดับที่สูงกว่าแล้วให้ผู้ใช้แก้ไขแฟ้มข้อมูลที่ถูกคัดลอกมาใหม่แทน

3. ไวท์เอาท์ (whiteout)

เมื่อต้องการลบแฟ้มข้อมูลในแบรนช์ที่มีนโยบายเป็นอ่านอย่างเดียว ระบบแฟ้มข้อมูลรวมจะสร้างแฟ้มข้อมูลประเภทไวท์เอาท์ขึ้นมา บนแบรนช์ที่อยู่ในลำดับที่สูงกว่า เพื่อใช้ในการซ่อนแฟ้มข้อมูลที่ต้องการลบนั้นไม่ให้ผู้ใช้เห็นต่อไปและแฟ้มข้อมูลที่ผู้ใช้ต้องการลบนั้นจะยังคงอยู่เพียงแต่ผู้ใช้จะมองไม่เห็น

2.2.2 Cumulus: File system Backup to the Cloud [6]

เป็นระบบสำรองข้อมูลที่พัฒนาบนโมเดลแบบธินคลาวด์ (thin cloud model) ฝั่งเซิร์ฟเวอร์รองรับเพียงส่วนต่อประสานอย่างง่าย ได้แก่ เก็ท (GET) พุท (PUT) ลิสต์ (LIST) ดีลีท (DELETE) การทำงานส่วนใหญ่ของระบบจะอยู่ทางฝั่งไคลเอนต์ ความสามารถหลักของระบบสำรองในงานวิจัยนี้ ได้แก่ การเก็บเฉพาะส่วนที่เปลี่ยนแปลงของแฟ้มข้อมูล (sub-file incremental) การเข้ารหัสข้อมูล (data encryption) การเรียกคืนข้อมูลชุดก่อนหน้า (roll back) การรวมบล็อกข้อมูลเล็ก ๆ เข้าด้วยกัน (data aggregation) เป็นกลุ่มข้อมูล เพื่อลดโอเวอร์เฮดที่เกิดจากการอัพโหลดแฟ้มข้อมูลขนาดเล็กเป็นจำนวนมาก นอกจากนี้ยังมีการกำจัดรูโหว่ภายในกลุ่มข้อมูลที่รวมจากบล็อกข้อมูลด้วย (segment cleaning) ข้อดีของระบบในงานวิจัยนี้คือผู้ใช้สามารถย้ายข้อมูลไปเก็บไว้ที่เซิร์ฟเวอร์อื่นได้สะดวก เพราะส่วนต่อประสานระหว่างไคลเอนต์และเซิร์ฟเวอร์ไม่ซับซ้อนและมีความยืดหยุ่นสูง

2.2.3 A distributed filesystem framework for transparent accessing heterogeneous storage services [7]

งานวิจัยนี้พัฒนาเฟรมเวิร์ค (framework) ที่ชื่อว่า YaFS ซึ่งใช้ในการพัฒนาระบบแฟ้มข้อมูลแบบกระจายที่รองรับเซิร์ฟเวอร์หลายตัวในคราวเดียวกันเป็นแหล่งเก็บข้อมูล แฟ้มข้อมูลที่อยู่ในระบบนี้จะถูกแบ่งออกเป็นบล็อกข้อมูล (block data) และจะถูกกระจายไปเก็บบนเซิร์ฟเวอร์ที่อยู่ภายในระบบ ทำให้การเข้าถึงแฟ้มข้อมูลใด ๆ ทำได้รวดเร็วเพราะเรียกบล็อกข้อมูลจากเซิร์ฟเวอร์หลาย ๆ ตัวได้พร้อมกัน และการเก็บข้อมูลแบบนี้ยังช่วยให้ใช้ประโยชน์จากพื้นที่ว่างได้เต็มที่ด้วย YaFS รองรับเซิร์ฟเวอร์ได้หลายประเภทเนื่องจากรองรับการติดตั้งปลั๊กอินใหม่ นอกจากนี้ยังรองรับการทำงานในขณะที่ตัดการเชื่อมต่อกับระบบเหมือนกับระบบแฟ้มข้อมูลโคดาด้วย

2.2.4 Personal cloud filesystem: A distributed unification filesystem for personal computer and portable device [8]

งานวิจัยนี้ได้พัฒนาระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคล (Personal cloud file system) ที่จะช่วยอำนวยความสะดวกในการเข้าถึงและจัดการแฟ้มข้อมูลระหว่างคอมพิวเตอร์ หรืออุปกรณ์เคลื่อนย้ายได้ โดยทำให้การเข้าถึงแฟ้มข้อมูลที่อยู่ในเครื่องปลายทางเสมือนกับว่าแฟ้มข้อมูลนั้นอยู่ในเครื่องของผู้ใช้เอง ซึ่งอยู่บนหลักการเดียวกับระบบแฟ้มข้อมูลแบบรวม องค์ประกอบภายในของระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลมีลักษณะเป็นโมดูล (Module) ซึ่งทำให้ง่ายต่อการปรับปรุงแก้ไของค์ประกอบในแต่ละส่วน

นอกจากนี้ระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลยังสนับสนุนการทำงานแบบออฟไลน์ (offline operation) โดยการแคชแฟ้มข้อมูลของเครื่องปลายทางเอาไว้ที่เครื่องผู้ใช้ ซึ่งเป็นหลักการเดียวกับระบบแฟ้มข้อมูลโคดา

บทที่ 3
หลักการและการออกแบบระบบ

บทนี้กล่าวถึงหลักการและแนวคิดที่ใช้ในการออกแบบระบบแฟ้มข้อมูลรวมบนเว็บเซอร์วิสแบบเรสฟูล โดยแบ่งออกเป็น 2 หัวข้อ คือ แนวคิดในการออกแบบ และการออกแบบระบบ

3.1 แนวคิดในการออกแบบ

แนวทางหนึ่งที่จะช่วยให้ผู้ใช้ทำงานกับแฟ้มข้อมูลข้ามเครื่องกันได้โดยสะดวกคือการฝากแฟ้มข้อมูลที่ต้องใช้งานเป็นประจำไว้บนพื้นที่เก็บข้อมูลแบบออนไลน์ และเข้าถึงแฟ้มข้อมูลเหล่านั้นผ่านทางระบบแฟ้มข้อมูล ซึ่งสำหรับการวิจัยนี้ได้พัฒนาระบบแฟ้มข้อมูลรวมอาร์เอฟเอส (RFS) โดยมีจุดประสงค์ตามแนวทางที่กล่าวไว้ข้างต้น และมีคุณสมบัติต่าง ๆ ดังต่อไปนี้

1. พัฒนาบนพื้นฐานของระบบแฟ้มข้อมูลรวม

ระบบแฟ้มข้อมูลรวมสามารถทำให้แฟ้มข้อมูลและไดเรคทอรีจากที่ต่าง ๆ เช่น ไดเรคทอรีบนเครื่องไคลเอนต์ ไดเรคทอรีจากเครื่องอื่น หรือแฟ้มข้อมูลที่ฝากไว้บนพื้นที่ออนไลน์ เป็นต้น เสมือนกับว่ามาอยู่ในที่ที่เดียวกัน ซึ่งเหมาะกับผู้ใช้ทั่วไปที่ชอบให้แฟ้มข้อมูลที่ใช้งานประจำมาอยู่ในที่เดียวกันเพื่อความสะดวกในการใช้งาน

2. ใช้เว็บเซิร์ฟเวอร์เป็นพื้นที่เก็บข้อมูลแบบออนไลน์

การใช้งานอินเทอร์เน็ตเป็นที่นิยมมากในปัจจุบัน จึงมีบริการต่าง ๆ เกิดขึ้นมากมาย รวมถึงการให้บริการเว็บเซิร์ฟเวอร์ ผู้ใช้สามารถหาเซิร์ฟเวอร์เหล่านี้มาใช้งานได้ไม่ยากนัก และยังมีเว็บเซิร์ฟเวอร์ที่ผู้ใช้สามารถใช้บริการโดยไม่เสียค่าใช้จ่ายด้วย จึงเหมาะมากสำหรับผู้ใช้ทั่วไป ดังนั้น การวิจัยนี้จึงออกแบบสำหรับการใช้เว็บเซิร์ฟเวอร์เป็นพื้นที่เก็บข้อมูลแบบออนไลน์ และเซิร์ฟเวอร์ที่ใช้ต้องรองรับภาษาพีเอชพี เนื่องจากเว็บเซิร์ฟเวอร์ที่รองรับภาษาพีเอชพีเป็นเซิร์ฟเวอร์ที่หาง่ายมากที่สุดประเภทหนึ่ง

3. ใช้โพรโตคอลเอชทีทีพีสำหรับการติดต่อกันระหว่างไคลเอนต์และเซิร์ฟเวอร์

โพรโตคอลเอชทีทีพีเป็นโพรโตคอลมาตรฐานที่ระบบปฏิบัติการส่วนใหญ่รองรับอยู่แล้ว และในระบบยังใช้หลักการของเรสฟูลเว็บเซอร์วิส (RESTful web service) เป็นพื้นฐานสำหรับการพัฒนาส่วนต่อประสานของเว็บเซิร์ฟเวอร์ โดยเรสฟูลเว็บเซอร์วิสจะสามารถใช้คุณสมบัติที่มีอยู่แล้วของโพรโตคอลเอชทีทีพีได้ เช่น การใช้ยูอาร์ไอ (URI) เพื่อระบุทรัพยากรในระบบ และการใช้เอชทีทีพีเมธอด (HTTP method) เพื่อกำหนดสิ่งที่จะกระทำกับทรัพยากร เป็นต้น และเรสฟูลเว็บ

เซอร์วิสในปัจจุบันถือว่าเป็นตัวเลือกที่มาแทนที่โซพเว็บเซอร์วิสที่ดีตัวหนึ่ง เนื่องจากโซพใช้ข้อความเอ็กซ์เอ็มแอลในการสื่อสาร ในขณะที่เรสฟูลเว็บเซอร์วิสอาจใช้ข้อความเอ็กซ์เอ็มแอล หรือข้อความเจสัน ซึ่งใช้พลังในการประมวลผลน้อยกว่าข้อความแบบเอ็กซ์เอ็มแอล สำหรับการส่งคำร้องขอที่คล้ายกัน เรสฟูลเว็บเซอร์วิสสามารถส่งด้วยจำนวนข้อมูลที่น้อยกว่า ส่งผลให้โอเวอร์เฮดต่ำกว่า เรสฟูลเว็บเซอร์วิสเข้าใจง่าย สามารถพัฒนาใหม่ตั้งแต่ต้นได้ง่าย ไม่เหมือนกับโซพเว็บเซอร์วิสซึ่งต้องพึ่งพาไลบรารีภายนอกมากกว่า

4. มีความสามารถในการขยายระบบ

แฟ้มข้อมูลของผู้ใช้มักจะมีจำนวนเพิ่มขึ้นเมื่อเวลาผ่านไป ระบบที่ออกแบบจึงควรรองรับการขยายระบบ หรือรองรับการขยายของพื้นที่เก็บข้อมูลโดยการใช้เว็บเซิร์ฟเวอร์ได้คราวละหลายตัว และเนื่องจากผู้ใช้อาจต้องย้ายเครื่องทำงานไปมา ซึ่งแต่ละเครื่องมีพื้นที่ว่างไม่เท่ากัน ดังนั้นการขยายระบบนี้ต้องไม่ถูกจำกัดโดยพื้นที่ว่างในฮาร์ดดิสก์ของแต่ละเครื่องด้วย

5. เก็บแฟ้มข้อมูลในรูปของบล็อกข้อมูล

การแบ่งแฟ้มข้อมูลออกเป็นหลายชิ้นหรือบล็อกข้อมูลจะช่วยให้ใช้ประโยชน์จากพื้นที่ที่เหลืออยู่บนเซิร์ฟเวอร์ได้ดีขึ้น และการดาวน์โหลดบล็อกข้อมูลหลายชิ้นพร้อมกันย่อมเร็วกว่าการดาวน์โหลดแฟ้มข้อมูลทั้งแฟ้มพอสมควร เนื่องจากใช้ประโยชน์จากขนาดแบนด์วิดท์ของเครือข่ายได้เต็มที่ ส่วนบล็อกข้อมูลจะใช้ขนาดเท่าใดนั้นจะขึ้นอยู่กับผลการทดสอบในหัวข้อที่ 5.2.1

6. แคชแฟ้มข้อมูลบนพื้นฐานของอัลกอริทึมแอลอาร์ยู

การแคชข้อมูลจะช่วยให้ระบบไม่ต้องดาวน์โหลดข้อมูลเดิมซ้ำเมื่อผู้ใช้เข้าถึงแฟ้มข้อมูลเดิมในเวลาต่อมา ทำให้ประหยัดเวลาและปริมาณการใช้งานเครือข่าย (Network traffic) การแคชจะพัฒนาบนพื้นฐานของอัลกอริทึมแอลอาร์ยู โดยเลือกเก็บแคชของแฟ้มข้อมูลที่ได้รับการใช้งานบ่อย และลบแคชของแฟ้มข้อมูลที่ไม่ค่อยได้ถูกใช้งานออกจากระบบ สาเหตุที่ใช้อัลกอริทึมแอลอาร์ยู เนื่องจากมีสมมติฐานที่ว่าในการทำงานอย่างใดอย่างหนึ่ง ผู้ใช้จะเข้าถึงแฟ้มข้อมูลชุดหนึ่งเป็นประจำ ซึ่งจะสอดคล้องกับสมมติฐานของอัลกอริทึมแอลอาร์ยูด้วย คือ ข้อมูลที่ถูกใช้งานบ่อยในช่วงเวลาใดเวลาหนึ่ง มักจะถูกใช้บ่อยในช่วงเวลาถัดจากนั้น

3.2 การออกแบบระบบ

ระบบในการวิจัยนี้ถูกออกแบบตามแนวคิดที่อธิบายไว้ในหัวข้อที่ 3.1 โดยแบ่งออกเป็น 3 ส่วนหลัก คือ สถาปัตยกรรมของระบบ ส่วนต่อประสานของระบบ และองค์ประกอบของอาร์เอฟเอสไคลเอนต์

3.2.1 **สถาปัตยกรรมของระบบ**

ระบบแฟ้มข้อมูลในการวิจัยนี้มีชื่อว่าอาร์เอฟเอส (RFS) ใช้สถาปัตยกรรมแบบไคลเอนต์-มัลติเปิลเซิร์ฟเวอร์ (client-multiple servers) ประกอบด้วย 3 ส่วนหลัก ได้แก่ อาร์เอฟเอสไคลเอนต์ (RFS client) เมตาเดตาโหนด และสตอร์เรจโหนด ซึ่งทั้งสองโหนดนี้เป็นเว็บเซิร์ฟเวอร์ที่ต้องรองรับภาษาพีเอชพี ภาพรวมของระบบแฟ้มข้อมูลอาร์เอฟเอสถูกแสดงไว้ดังภาพที่ 2

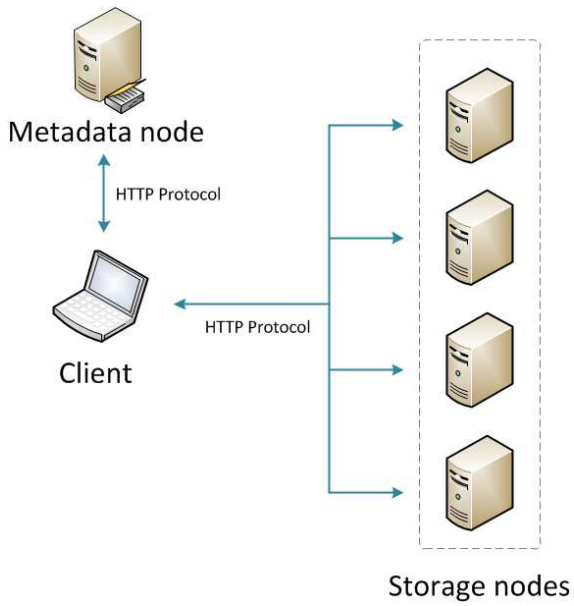
อาร์เอฟเอสไคลเอนต์ คือ แอพลิเคชันที่อยู่บนเครื่องทางฝั่งไคลเอนต์ ทำงานประสานกันกับเมตาเดตาโหนด และสตอร์เรจโหนด โดยรายละเอียดของอาร์เอฟเอสไคลเอนต์ถูกอธิบายในหัวข้อที่ 3.2.3

อาร์เอฟเอสเรียกใช้งานข้อมูลเมตาเดตาจากฐานข้อมูลที่อยู่บนเครื่องไคลเอนต์ เมื่อแฟ้มข้อมูลในระบบถูกแก้ไข หรือมีการสร้างแฟ้มข้อมูลใหม่ เมตาเดตาของแฟ้มข้อมูลเหล่านั้นก็จะถูกบันทึกไว้ในฐานข้อมูลบนเครื่องไคลเอนต์โดยไม่จำเป็นต้องติดต่อกับเมตาเดตาโหนด อย่างไรก็ตามข้อมูลที่อยู่ภายในฐานข้อมูลจะถูกสำรองไปที่เมตาเดตาโหนดในภายหลัง

อาร์เอฟเอสมีความสามารถในการขยายระบบ ดังนั้นจึงรองรับสตอร์เรจโหนดได้หลายตัวในคราวเดียวกัน ผู้ใช้ต้องติดตั้งพีเอชพีสคริปต์บนเซิร์ฟเวอร์ และเพิ่มชื่อหรือยูอาร์แอลซึ่งใช้บ่งบอกที่อยู่ของเซิร์ฟเวอร์นั้นลงในฐานข้อมูลเพื่อให้ใช้งานเป็นสตอร์เรจโหนดได้ พีเอชพีสคริปต์จะทำให้เซิร์ฟเวอร์มีส่วนต่อประสานของเว็บเซอร์วิสแบบเรสฟูล โดยที่เซิร์ฟเวอร์ต้องรองรับเมธอดเก็ท และเมธอดโพสท์ของโพรโตคอลเอชทีทีพีด้วย เนื่องจากอาร์เอฟเอสไคลเอนต์จะติดต่อกับโหนดทุกตัวในระบบผ่านทางโพรโตคอลเอชทีทีพี

ข้อมูลของแฟ้มข้อมูลในระบบจะถูกเก็บไว้บนสตอร์เรจโหนด ถ้าขณะนั้นระบบมีการใช้งานสตอร์เรจโหนดหลายตัว ข้อมูลก็จะถูกกระจายไปเก็บไว้ในแต่ละโหนดโดยเลือกเก็บบนโหนดที่มีพื้นที่เหลือมากที่สุดเป็นอันดับแรก ถ้าไม่สามารถติดต่อโหนดที่ต้องการได้ ระบบก็สามารถส่งข้อมูลนั้นไปเก็บบนโหนดอื่นแทน

นอกจากนี้อาร์เอฟเอสไคลเอนต์ยังสามารถแคชแฟ้มข้อมูลต่าง ๆ ในระบบได้ โดยประยุกต์ใช้อัลกอริทึมการแทนที่แอลอาร์ยู เมื่อผู้ใช้เข้าถึงแฟ้มข้อมูลที่ยังไม่ถูกแคช แคชที่อยู่ในระบบอยู่แล้วและไม่ถูกใช้งานนานที่สุดจะถูกลบทิ้ง ถ้าขนาดรวมของแคชทั้งหมดเกินกว่าที่กำหนด



ภาพที่ 2 ภาพรวมของระบบแฟ้มข้อมูลอาร์เอฟเอส

3.2.2 **ส่วนต่อประสานของระบบ**

เมตาเดตาโหนด และสตอร์เรจโหนดมีส่วนต่อประสานที่แตกต่างกัน แต่ละส่วนต่อประสานถูกระบุด้วยยูอาร์ไอ (URI) ผลลัพธ์จากส่วนต่อประสานอาจอยู่ในรูปของข้อความเจสัน (JSON) หรือเป็นแฟ้มข้อมูลขึ้นอยู่กับยูอาร์ไอ และเอชทีทีพีเมธอดที่ใช้ติดต่อ อาร์เอฟเอสไคลเอนต์จะใช้เมธอดโพสท์เวลาอัพโหลดหรือลบแฟ้มข้อมูลเท่านั้น โดยที่แฟ้มข้อมูลนั้นอาจเป็นได้ทั้งฐานข้อมูล หรือบล็อกข้อมูลภายในระบบก็ได้ โดยรายละเอียดของส่วนต่อประสานซึ่งแบ่งกลุ่มตามชนิดของโหนด มีดังต่อไปนี้

3.2.2.1 **ส่วนต่อประสานของเมตาเดตาโหนด**

ส่วนต่อประสานของเมตาเดตาโหนดจะถูกติดต่อผ่านทั้งเมธอดเก็ท และเมธอดโพสท์ สำหรับส่วนต่อประสานที่ใช้เมธอดเก็ทในการติดต่อจะถูกใช้เพื่อตรวจสอบข้อมูลในฐานข้อมูลล่าสุดผ่านทางเบราว์เซอร์เท่านั้น โดยส่วนต่อประสานของเมตาเดตาโหนดมีทั้งหมด 4 ส่วน ดังนี้

- GET {server_name}/file/

  ผู้ใช้สามารถตรวจสอบสถานะล่าสุดของเมตาเดตาในระบบผ่านทางส่วนต่อประสานนี้ ซึ่งคืนผลลัพธ์เป็นโครงสร้างข้อมูลแบบดิกชันนารี (dictionary) ของเมตาเดตาของแฟ้มข้อมูลในระบบในรูปของข้อความเจสัน ดังตัวอย่างด้านล่างนี้

```
{ /dat.txt: {
      fid: "2",
      path: "/dat.txt",
      size: "6",
      block: ["1"],
      uid: "1000",
      gid: "1000",
      parent: "1",
      atime: "1337760181",
      mtime: "1337760181",
      ctime: "1337760181",
      mode: "33204",
      link: "1",
      timestamp: "1337760181"
      }
}
```

- GET {server_name}/server/

  ส่วนต่อประสานนี้คืนผลลัพธ์เป็นโครงสร้างข้อมูลแบบรายการ (list) ของสตอร์เรจโหนดในรูปของข้อความเจสัน โดยแต่ละรายการมีข้อมูลชื่อ สถานะ และหมายเลขประจำตัวของเซิร์ฟเวอร์ (server identification number) ดังตัวอย่างต่อไปนี้

```
[{
      sid: "1",
      sname: "http://localhost/storage",
      status: "1",
      timestamp: "1337760143"
},
{
      sid: "2",
      sname: "http://192.168.1.33/storage",
      status: "0",
      timestamp: "1337760143"
}]
```

- GET {server_name}/block/

  ส่วนต่อประสานนี้คืนผลลัพธ์ของโครงสร้างข้อมูลแบบรายการของบล็อกข้อมูลภายในระบบในรูปของข้อความเจสัน แต่ละรายการจะระบุว่าแต่ละบล็อกข้อมูลอยู่บนโหนดตัวใด และมีหมายเลขประจำตัว (block identification number) เท่าใด ดังตัวอย่างด้านล่างนี้

```
[{
      bid: "1",
      sname: "http://localhost/storage"
},
{
      bid: "2",
      sname: "http://192.168.1.33/storage"
}]
```

- POST {server_name}/db/

  อาร์เอฟเอสไคลเอนต์จะอัพโหลดฐานข้อมูลล่าสุดไปที่เมตาเดตาโหนดโดยส่งคำร้องขอพร้อมกับระบุที่อยู่ฐานข้อมูลในเครื่องไคลเอนต์ผ่านส่วนต่อประสานนี้ ดังตัวอย่างด้านล่างนี้

```
{action    : upload,
 file      : path_to_database_file}
```

  ผลลัพธ์ของส่วนต่อประสานอื่นที่กล่าวไปแล้วจะขึ้นอยู่กับฐานข้อมูลที่ถูกอัพโหลดผ่านส่วนต่อประสานนี้นั่นเอง

สำหรับส่วนที่แสดงด้วย {server_name} ของส่วนต่อประสานของเมตาเดตาโหนดให้แทนด้วยชื่อเซิร์ฟเวอร์ของเมตาเดตาโหนด

3.2.2.2 **ส่วนต่อประสานของสตอร์เรจโหนด**

ส่วนต่อประสานของสตอร์เรจโหนดจะถูกติดต่อผ่านทั้งเมธอดเก็ท และเมธอดโพสท์เช่นเดียวกับเมตาเดตาโหนด โดยสตอร์เรจโหนดมีส่วนต่อประสานทั้งหมด 3 ส่วน ดังนี้

- GET {server_name}/file/{block_id}

  ส่วนต่อประสานนี้ใช้สำหรับดาวน์โหลดบล็อกข้อมูลจากสตอร์เรจโหนด โดยการเติมเลขประจำตัวของบล็อกข้อมูลต่อท้ายยูอาร์ไอ ยกตัวอย่างเช่น แฟ้มข้อมูล A ประกอบด้วยบล็อกข้อมูลหมายเลข 1 และหมายเลข 2 โดยทั้งสองบล็อกนี้อยู่บนสตอร์เรจโหนด http://192.168.1.33 อาร์เอฟเอสไคลเอนต์จะส่งคำร้องไปที่ยูอาร์ไอ http://192.168.1.33/file/1 และ http://192.168.1.33/file/2 เพื่อดาวน์โหลดบล็อกข้อมูลตามลำดับ

- GET {server_name}/status/

  ส่วนต่อประสานนี้คืนผลลัพธ์เป็นข้อมูลขนาดของพื้นที่ว่างและพื้นที่ทั้งหมดของสตอร์เรจโหนดตัวที่ต้องการ ดังตัวอย่างด้านล่างนี้

```
{"total":2000000000, "free":2000000000}
```

  ข้อมูลเหล่านี้จะถูกรวมรวมจากทุกสตอร์เรจโหนดเมื่ออาร์เอฟเอสไคลเอนต์ต้องการอัพโหลดบล็อกข้อมูล

- POST {server_name}/file/

  ส่วนต่อประสานนี้ทำงานได้ 2 อย่าง คือ การอัพโหลดบล็อกข้อมูล และการลบบล็อกข้อมูล การทำงานอย่างใดอย่างหนึ่งจะขึ้นอยู่กับข้อมูลที่แนบไปกับคำร้องขอ เช่น การอัพโหลดบล็อกข้อมูลต้องแนบที่อยู่ของบล็อกข้อมูลนั้นบนเครื่องของไคลเอนต์และข้อความ “upload” ไปกับคำร้องขอ ตามตัวอย่างด้านล่างนี้

```
{action    : upload,
 file      : path_to_block_file}
```

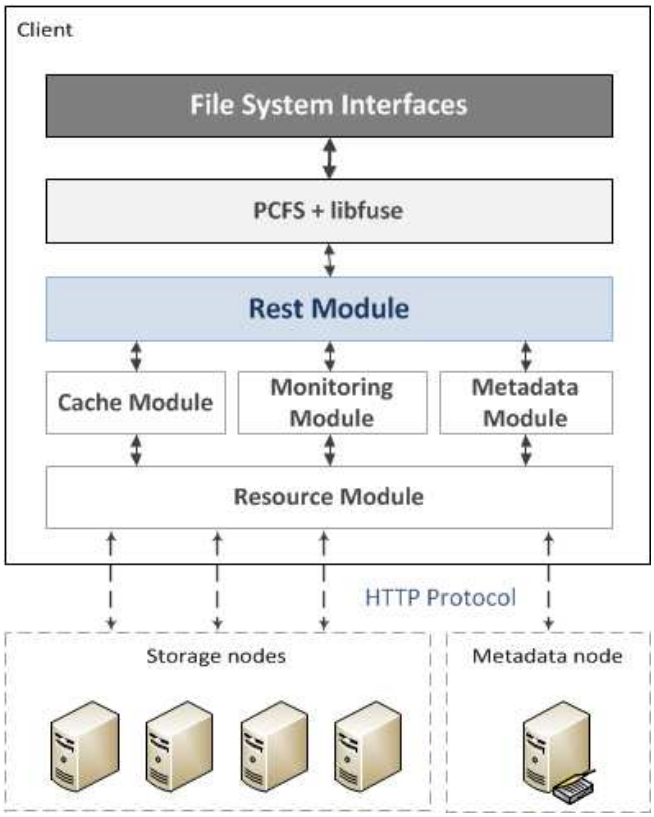
  ส่วนการลบบล็อกข้อมูลจะแนบหมายเลขประจำตัวของบล็อกข้อมูลและข้อความ “delete” ไปกับคำร้องขอ ดังตัวอย่างต่อไปนี้

```
{action    : delete,
 block     : block id number}
```

สำหรับส่วนที่แสดงด้วย {server_name} ของส่วนต่อประสานของสตอร์เรจโหนดให้แทนด้วยชื่อเซิร์ฟเวอร์ของสตอร์เรจโหนด

3.2.3 **องค์ประกอบของอาร์เอฟเอสไคลเอนต์**

อาร์เอฟเอสไคลเอนต์ประกอบด้วย โมดูลหลัก 1 โมดูล โมดูลย่อย 4 โมดูล และเครื่องมืออรรถประโยชน์ (utility tool) 1 ชุด ซึ่งองค์ประกอบเหล่านี้เป็นส่วนต่อเติมของระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคล ดังแสดงในภาพที่ 3



ภาพที่ 3 องค์ประกอบของอาร์เอฟเอสไคลเอนต์

โดยแต่ละองค์ประกอบมีรายละเอียดดังต่อไปนี้

1. เรสโมดูล (rest module)

เรสโมดูลเป็นโมดูลหลักของอาร์เอฟเอสไคลเอนต์ และถือว่าเป็นโมดูลเครือข่าย (network module) ของระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลด้วย โมดูลนี้ควบคุมการทำงานทั้งหมดของอาร์เอฟเอสไคลเอนต์ เมื่อมีการงานหรือการเปลี่ยนแปลงใด ๆ เกิดขึ้น เช่น การสร้างแฟ้มข้อมูลใหม่ การอ่านแฟ้มข้อมูล หรือการเขียนแฟ้มข้อมูล เป็นต้น อาร์เอฟเอสไคลเอนต์จะเรียกเมธอดที่เกี่ยวข้องภายในโมดูลให้ทำงานนั้น แล้วเมธอดเหล่านั้นจะเรียกโมดูลย่อยอื่นให้ทำงานอีกทอดหนึ่ง

2. รีซอร์สโมดูล (resource module)

โมดูลนี้มีหน้าที่หลัก 2 อย่าง คือ การดาวน์โหลดบล็อกข้อมูลจากสตอร์เรจโหนด และการแบ่งแฟ้มข้อมูลออกเป็นบล็อกข้อมูล เมื่อผู้ใช้เปิดแฟ้มข้อมูลที่ยังไม่มีในแคชของระบบ โมดูลนี้จะดาวน์โหลดบล็อกข้อมูลทุกส่วนของแฟ้มข้อมูลนั้น แล้วรวมเข้าด้วยกันเป็นแฟ้มข้อมูลดั้งเดิม จากนั้นจะถูกแคชไว้ในระบบ และเมื่อผู้ใช้สร้างแฟ้มข้อมูลใหม่หรือแก้ไขแฟ้มข้อมูลที่มีอยู่เดิม โมดูลนี้จะแบ่งแฟ้มข้อมูลเหล่านั้นออกเป็นบล็อกข้อมูลที่มีขนาดเท่า ๆ กัน แล้วเก็บบล็อกข้อมูลเหล่านี้ไว้ที่ไดเรคทอรีชั่วคราว เพื่อรอการอัพโหลดโดยเครื่องมืออรรถประโยชน์ต่อไป

3. เมตาเดตาโมดูล (metadata module)

โมดูลนี้ทำหน้าที่เป็นตัวกลางสำหรับการติดต่อกับระบบฐานข้อมูล ซึ่งอาร์เอฟเอสไคลเอนต์บันทึกเมตาเดตาของแฟ้มข้อมูลในระบบโดยใช้ระบบฐานข้อมูลเอสคิวไลท์ (SQLite) รวมทั้งบันทึกข้อมูลของบล็อกข้อมูล และเซิร์ฟเวอร์ที่มีในฐานข้อมูลด้วย

4. มอนิเตอร์ริงโมดูล (monitoring module)

โมดูลนี้ทำหน้าที่ตรวจสอบพื้นที่ว่างของสตอร์เรจโหนดเป็นระยะ ๆ เมื่อมีการอัพโหลดบล็อกข้อมูลไปที่สตอร์เรจโหนด อาร์เอฟเอสไคลเอนต์จะเรียกดูข้อมูลของพื้นที่ว่างจากมอนิเตอร์ริงโมดูล และเลือกโหนดที่มีพื้นที่มากที่สุดเป็นอันดับแรกเป็นจุดหมายของบล็อกข้อมูล โมดูลนี้ทำงานในลักษณะของงานเบื้องหลัง (background job) เพื่อหลีกเลี่ยงโอเวอร์เฮดที่อาจเกิดขึ้น

5. แคชโมดูล (cache module)

อาร์เอฟเอสไคลเอนต์สามารถทำงานได้ตามปกติแม้ใช้เพียงหน่วยความจำภายใน (Internal memory) สำหรับการอ่านแฟ้มข้อมูลต่าง ๆ ในระบบ แต่ประสิทธิภาพโดยรวมจะลดลง ถ้าแฟ้มข้อมูลมีขนาดใหญ่เกินไป เนื่องจากทำให้หน่วยความจำภายในไม่เพียงพอ ซึ่งแคชโมดูลจะช่วยป้องกันปัญหานี้โดยการแคชแฟ้มข้อมูลที่ถูกอ่าน และยังช่วยลดปริมาณการใช้งานเครือข่าย (network traffic) ที่เกิดจากการติดต่อกับสตอร์เรจโหนดด้วย

6. เครื่องมืออรรถประโยชน์ (utility tool)

เครื่องมืออรรถประโยชน์คือเครื่องมือภายนอกที่จะถูกใช้งานโดยอาร์เอฟเอสไคลเอนต์ หน้าที่หลักคือการอัพโหลดบล็อกข้อมูลจากไดเรคทอรีชั่วคราวไปที่สตอร์เรจโหนด และการลบบล็อกข้อมูลที่ไม่จำเป็นต่อการใช้งานออกจากระบบ เครื่องมือนี้ทำงานในลักษณะของงานเบื้องหลังเช่นเดียวกับมอนิเตอร์ริงโมดูล

บทที่ 4
การพัฒนาระบบ

ในบทนี้กล่าวถึงการพัฒนาระบบ และเครื่องมือที่ใช้ในการวิจัย ดังนี้

4.1 การพัฒนาระบบ

การพัฒนาอาร์เอฟเอสแบ่งออกเป็น 2 ส่วนคือ อาร์เอฟเอสไคลเอนต์ และส่วนต่อประสานของเซิร์ฟเวอร์ โดยอาร์เอฟเอสไคลเอนต์ทำงานบนเครื่องของผู้ใช้ และส่วนต่อประสานของเซิร์ฟเวอร์ก็คือส่วนต่อประสานของเมตาเดตาโหนด และสตอร์เรจโหนดที่อธิบายไว้ในหัวข้อ 3.2.2

อาร์เอฟเอสไคลเอนต์พัฒนาต่อจากระบบแฟ้มคลาวด์ส่วนบุคคลโดยใช้ภาษาไพธอน ซึ่งอาร์เอฟเอสไคลเอนต์อาจพัฒนาขึ้นมาใหม่โดยใช้ไลบรารีฟิวส์ (FUSE library) [9] แต่การพัฒนาต่อจากระบบแฟ้มคลาวด์ส่วนบุคคลทำให้การพัฒนาอาร์เอฟสเอสไคลเอนต์ใช้เวลาน้อยกว่ามาก และระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลยังมีคุณสมบัติหลายอย่างที่นำมาใช้ประโยชน์ได้ ได้แก่

- ความสามารถในการรวบรวมแฟ้มข้อมูลและไดเรคทอรีจากคอมพิวเตอร์ต่างเครื่องมาสู่ที่ที่เดียวกัน ซึ่งช่วยให้ผู้ใช้เข้าถึงแฟ้มข้อมูลของทั้งคอมพิวเตอร์บนเครือข่ายเดียวกันและระบบอาร์เอฟเอส โดยที่ผู้ใช้ไม่จำเป็นต้องรู้ว่าแฟ้มเหล่านี้อยู่บนเครื่องไหน และช่วยให้การจัดการแฟ้มข้อมูลข้ามเครื่องมีความสะดวกมากขึ้นด้วย
- ความสามารถในการเมาท์ (mount) ระบบแฟ้มข้อมูลจำลอง (virtual file system) บนเครื่องไคลเอนต์โดยอาศัยไลบรารีฟิวส์ที่อยู่ในระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคล ซึ่งช่วยให้ผู้ใช้เข้าถึงแฟ้มข้อมูลต่าง ๆ เสมือนกับว่าแฟ้มเหล่านั้นอยู่บนเครื่องของไคลเอนต์เอง
- องค์ประกอบภายในมีลักษณะเป็นโมดูล สามารถแก้ไขการทำงานในส่วนต่าง ๆ ได้ง่าย และเพิ่มการรองรับโพรโตคอลใหม่ ๆ ได้ด้วยการพัฒนาโมดูลเครือข่าย แล้วนำมาใช้ภายในระบบ ในปัจจุบันระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลรองรับโมดูลเครือข่ายเพียง 2 โมดูล คือ โลคัลโมดูล (local module) ซึ่งใช้กับแฟ้มข้อมูลภายในเครื่อง และเอสเอสเอชโมดูล (SSH module) สำหรับการเข้าถึงแฟ้มข้อมูลที่อยู่ต่างเครื่องกัน การวิจัย

นี้พัฒนาเรสโมดูลเพิ่มขึ้นมาเพื่อให้รองรับส่วนต่อประสานของเว็บเซอร์วิสแบบเรสฟูลที่เป็นส่วนต่อประสานของเมตาเดตาโหนด และสตอร์เรจโหนดนั่นเอง

นอกจากนี้อาร์เอฟเอสไคลเอนต์ใช้ระบบฐานข้อมูลเอสคิวไลท์สำหรับการบันทึกเมตาเดตา ข้อมูลของเซิร์ฟเวอร์ที่ใช้เป็นสตอร์เรจโหนด และข้อมูลของบล็อกข้อมูล โดยโครงสร้างของตารางฐานข้อมูลแสดงไว้ในภาคผนวก

สำหรับส่วนต่อประสานของเซิร์ฟเวอร์สามารถใช้ภาษาใด ๆ พัฒนาก็ได้ แต่สำหรับการวิจัยนี้จะพัฒนาด้วยภาษาพีเอชพี เพราะพีเอชพีเป็นภาษาที่มีความนิยมมากในปัจจุบัน ส่งผลให้จำนวนเซิร์ฟเวอร์ที่รองรับภาษาพีเอชพีมีมากตามไปด้วย และมากกว่าเซิร์ฟเวอร์ที่รองรับภาษาอื่น เช่น จาวา ไพธอน และรูบี (Ruby) เป็นต้น ซึ่งพีเอชพีเซิร์ฟเวอร์ที่นำมาใช้ในระบบอาจเป็นแบบที่เสียค่าใช้จ่ายหรือไม่เสียก็ได้

เนื้อหาส่วนที่เหลือในหัวข้อนี้จะอธิบายถึงรายละเอียดภายในระบบ ดังนี้

4.1.1 เมธอดของเรสโมดูล

เรสโมดูลถือว่าเป็นหนึ่งในโมดูลเครือข่ายของระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคล เมธอดที่อิมพลีเมนต์ในเรสโมดูลจึงต้องสอดคล้องกับแนวทางที่ระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลออกแบบไว้ โดยเมธอดส่วนใหญ่จะเป็นเมธอดที่เกี่ยวกับการจัดการแฟ้มข้อมูล ดังตารางที่ 1

ตารางที่ 1 ตารางแสดงเมธอดที่อิมพลีเมนต์ในเรสโมดูล

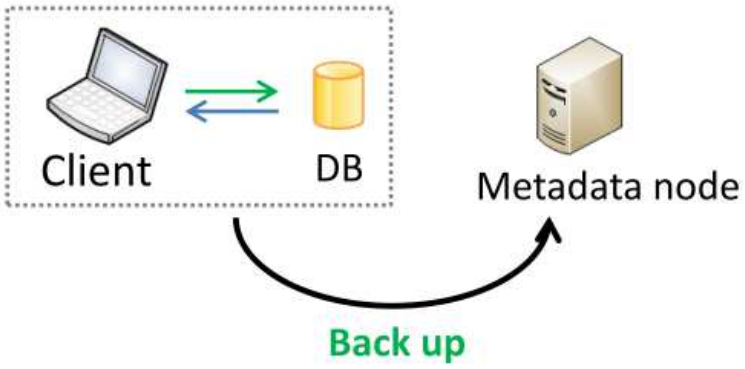
เมธอด	รายละเอียด
close_connection	จะถูกเรียกเมื่อผู้ใช้อันเมาท์ (unmount) ระบบแฟ้มข้อมูล
path_exists	ตรวจสอบการมีอยู่ของแฟ้มข้อมูลในระบบ
makedirs	สร้างไดเรคทอรีแบบหลายชั้น
create_whiteout	สร้างแฟ้มข้อมูลไวท์เอาท์ตามหลักการของระบบแฟ้มข้อมูลรวม
open	เปิดแฟ้มข้อมูลและคืนค่าไฟล์อ็อกเจกต์
lstat	รับข้อมูลเมตาเดตาของแฟ้มข้อมูล หรือไดเรคทอรี
listdir	ลิสต์รายการแฟ้มข้อมูลที่อยู่ในไดเรคทอรีที่ระบุ
mkdir	สร้างไดเรคทอรี
rmdir	ลบไดเรคทอรี
read	อ่านแฟ้มข้อมูล
mknod	สร้างแฟ้มข้อมูลใหม่
write	เขียนแฟ้มข้อมูล
rename	เปลี่ยนชื่อแฟ้มข้อมูลหรือไดเรคทอรี
truncate	ตัดปลายข้อมูลของแฟ้มข้อมูลให้เท่ากับขนาดที่ระบุ
unlink	ลบแฟ้มข้อมูลหรือไดเรคทอรี
chmod	เปลี่ยนโหมดของแฟ้มข้อมูล หรือไดเรคทอรี
chown	เปลี่ยนกลุ่มผู้ใช้ของแฟ้มข้อมูล หรือไดเรคทอรี
utime	แก้ไขเวลาที่มีการเข้าถึงแฟ้มข้อมูล หรือเวลาที่แฟ้มข้อมูลถูกแก้ไข
statfs	เรียกดูสถานะพื้นที่ของระบบแฟ้มข้อมูล
release	เมื่ออ่านหรือเขียนแฟ้มข้อมูลเสร็จ เมธอดนี้จะถูกเรียกทุกครั้ง

สำหรับการพัฒนาระบบแฟ้มข้อมูลอาร์เอฟเอส ยังได้เพิ่มเมธอดอีกตัวที่ไม่มีในระบบแฟ้มข้อมูลคลาวด์ส่วนบุคคลด้วย ซึ่งก็คือ เมธอด `release` เมธอดนี้จะถูกเรียกทุกครั้งเมื่ออ่านแฟ้มข้อมูล หรือเขียนแฟ้มข้อมูลเสร็จแล้ว ซึ่งข้อดีของเมธอดนี้คือจะช่วยให้รู้ว่าการเขียนแฟ้มข้อมูลเสร็จสิ้นที่ตรงไหน ทำให้ระบบสามารถบันทึกขนาดของแฟ้มข้อมูลลงในฐานข้อมูลได้ถูกต้อง และยังสั่งให้รีซอร์สโมดูลแบ่งแฟ้มข้อมูลชั่วคราวออกเป็นบล็อก และอัพโหลดบล็อกข้อมูลผ่านทางเมธอดนี้ได้ด้วย

ถ้าไม่พัฒนาต่อจากระบบแฟ้มคลาวด์ส่วนบุคคล อาจเริ่มพัฒนาจากไลบรารีฟิวส์ได้โดยใช้เมธอดที่คล้าย ๆ กัน

4.1.2 ฐานข้อมูล และการจัดการเมตาเดตาของแฟ้มข้อมูล

ในช่วงแรกของการพัฒนาระบบได้นำฐานข้อมูลไปไว้บนเมตาเดตาโหนด ดังนั้น อาร์เอฟเอสไคลเอนต์ก็จำเป็นต้องติดต่อกับเมตาเดตาโหนดทุกครั้งที่มีการเปลี่ยนแปลงเกิดขึ้นในระบบ และบันทึกการเปลี่ยนแปลงนั้นลงฐานข้อมูลที่อยู่ทางฝั่งเมตาเดตาโหนด ซึ่งข้อเสียคือระบบทำงานช้าลงเพราะต้องเสียเวลาเชื่อมต่อและส่งคำร้องขอไปที่เมตาเดตาโหนด และถ้าไม่สามารถเชื่อมต่อเครือข่ายอินเทอร์เน็ต ระบบก็จะทำงานไม่ได้ ดังนั้น จึงย้ายฐานข้อมูลมาไว้บนเครื่องไคลเอนต์ และให้เมตาเดตาโหนดทำหน้าที่สำรองแฟ้มข้อมูลของฐานข้อมูลแทน ดังแสดงในภาพที่ 4



ภาพที่ 4 การติดต่อกับฐานข้อมูลของอาร์เอฟเอสไคลเอนต์

เมตาเดตาโมดูลเป็นโมดูลที่ทำหน้าที่เป็นส่วนต่อประสานของฐานข้อมูลเอสคิวไลท์ ซึ่งหน้าที่หลักก็คือการบันทึก แก้ไขรายการของแฟ้มข้อมูล และเมตาเดตาของแฟ้มข้อมูลในระบบ ข้อมูลส่วนนี้จะอยู่ในตาราง file ในฐานข้อมูล โดยสามารถดูรายละเอียดได้ที่ภาคผนวก

สำหรับการสร้างแฟ้มข้อมูลใหม่ การสร้างแฟ้มไวท์เอาท์ การสร้างไดเรคทอรีใหม่ การเปลี่ยนชื่อ การลบแฟ้มข้อมูลและไดเรคทอรี ก็เป็นเพียงการเพิ่มระเบียนหรือแก้ไขระเบียนที่มีอยู่เดิมในฐานข้อมูล ซึ่งเป็นหน้าที่ของเมตาเดตาโหนดเช่นกัน

นอกจากข้อมูลเกี่ยวกับแฟ้มข้อมูลแล้ว ยังมีการบันทึกข้อมูลเกี่ยวกับบล็อกข้อมูล และสตอร์เรจโหนดที่ใช้ในระบบด้วยไว้ในฐานข้อมูลด้วย ซึ่งจะมีเมธอดที่ช่วยในการจัดการข้อมูลเหล่านี้ เช่น การเรียกรายการบล็อกข้อมูล การอัพเดทรายการบล็อกข้อมูล การเรียกดูรายการบล็อกข้อมูลที่ไม่ถูกอ้างอิงถึงในระบบแล้ว การเรียกดูรายการสตอร์เรจโหนดที่ใช้ในระบบ เป็นต้น

เมื่อผู้ใช้อันเมาท์ระบบแฟ้มข้อมูล เมธอด `close_connection` ในเรสโมดูลก็จะสำรองข้อมูลของฐานข้อมูลไปยังเมตาเดตาโหนดโดยอัตโนมัติ

4.1.3 **ความสามารถในการขยายระบบ** (Scalability)

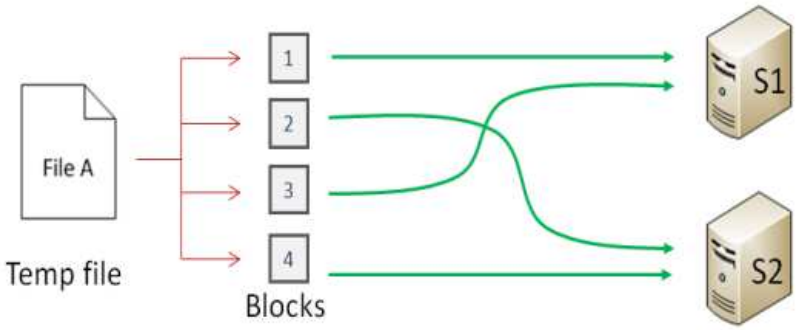
แฟ้มข้อมูลในระบบจะถูกเก็บไว้บนสตอร์เรจโหนด ผู้ใช้สามารถดูรายการแฟ้มข้อมูลในระบบได้โดยไม่จำเป็นต้องดาวน์โหลดแฟ้มข้อมูลเหล่านั้นมาเก็บไว้บนเครื่อง ยกเว้นกรณีการอ่านแฟ้มข้อมูล ผู้ใช้สามารถเพิ่มขนาดความจุที่ระบบรองรับโดยไม่ขึ้นกับขนาดของพื้นที่ว่างบนเครื่องไคลเอนต์ โดยนำเซิร์ฟเวอร์ที่ได้รับการติดตั้งพีเอชพีสคริปต์แล้วมาใช้งานในระบบ โดยการเพิ่มชื่อเซิร์ฟเวอร์ของสตอร์เรจโหนดลงในฐานข้อมูลโดยตรง หรืออาจเรียกเครื่องมือภายนอกก็ได้ โดยใช้คำสั่งดังนี้

```
$ python add_node.py {storage node'server name}
```

โดยแทนที่ {storage node'server name} ด้วยชื่อเซิร์ฟเวอร์ของสตอร์เรจโหนด

4.1.4 **การเขียนแฟ้มข้อมูล และการอัพโหลดบล็อกข้อมูล**

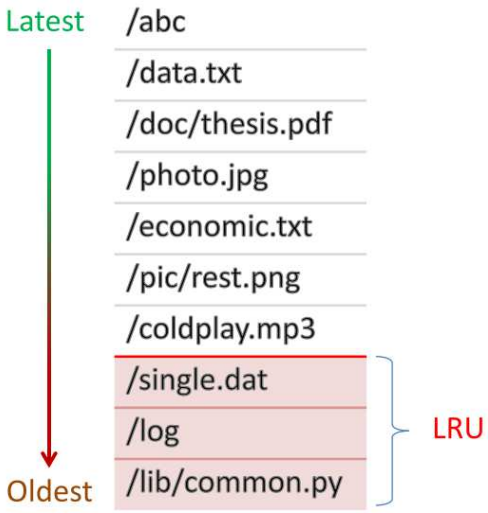
อาร์เอฟเอสไคลเอนต์สามารถเขียนข้อมูลไปที่สตอร์เรจโหนดได้โดยตรงผ่านทางเมธอดโพสท์ แต่จะทำให้ประสิทธิภาพในการเขียนแฟ้มข้อมูลลดลงมาก ดังนั้น อาร์เอฟเอสไคลเอนต์จึงเขียนข้อมูลเหล่านั้นใส่ในแฟ้มข้อมูลชั่วคราว (temporary file) จากนั้นแฟ้มข้อมูลชั่วคราวจะถูกแบ่งออกเป็นบล็อกข้อมูลที่มีขนาดเท่ากันโดยรีซอร์สโมดูล และบล็อกข้อมูลจะถูกอัพโหลดทันทีเมื่อการเขียนข้อมูลเสร็จ โดยระบบจะเลือกสตอร์เรจโหนดที่มีพื้นที่ว่างมากที่สุดเพื่ออัพโหลดบล็อกข้อมูลก่อนในกรณีที่มีพื้นที่เหลือไม่ต่างกัน แต่ถ้าต่างกันมากก็อาจประยุกต์ใช้วิธีราวด์โรบิน (round robin) แทนก็ได้ การทำงานคร่าว ๆ แสดงไว้ในภาพที่ 5 หลังจากนั้น แฟ้มข้อมูลชั่วคราวจะถูกแคชเก็บไว้ ซึ่งประสิทธิภาพในการเขียนแฟ้มข้อมูลของวิธีนี้จะดีกว่าวิธีแรก แต่ยังไม่ดีที่สุด เนื่องจากเวลาที่ใช้อัพโหลดบล็อกข้อมูลจะไปขัดจังหวะการทำงานของระบบ ส่งผลให้ทำงานช้าลงมากหรือน้อยขึ้นอยู่กับขนาดของแฟ้มข้อมูลที่เขียนและขนาดแบนด์วิดท์ของเครือข่าย (network bandwidth) ที่ใช้ได้ ดังนั้น อาร์เอฟเอสไคลเอนต์จึงใช้เครื่องมืออรรถประโยชน์สำหรับการอัพโหลดบล็อกข้อมูลในลักษณะของงานเบื้องหลัง ซึ่งทำให้ระบบทำงานราบรื่นมากขึ้นโดยที่ผู้ใช้สามารถทำงานอื่นต่อได้ทันทีหลังจากเขียนแฟ้มข้อมูลเสร็จสิ้น



ภาพที่ 5 การเขียนแฟ้มข้อมูล และการอัพโหลดบล็อกข้อมูล

4.1.5 การอ่านแฟ้มข้อมูล และแอลอาร์ยูแคช (LRU Cache)

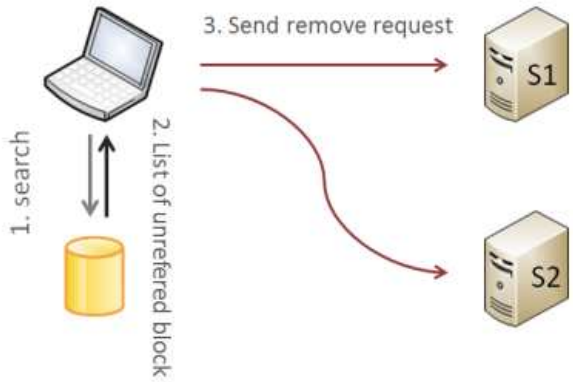
การอ่านแฟ้มข้อมูลโดยตรงจากสตอร์เรจโหนดทุกครั้งที่เข้าถึงแฟ้มข้อมูลนั้นจะเพิ่มปริมาณการใช้งานเครือข่ายโดยไม่จำเป็น ถ้าข้อมูลภายในไม่เปลี่ยนแปลง ก็ไม่ควรดาวน์โหลดบล็อกข้อมูลใหม่ทุกครั้งที่อ่านแฟ้มข้อมูล ดังนั้น อาร์เอฟเอสไคลเอนต์จึงมีความสามารถในการแคชข้อมูลที่อ่าน โดยพัฒนาบนพื้นฐานของอัลกอริทึมการแทนที่แอลอาร์ยู การแคชข้อมูลเป็นหน้าที่ของแคชโมดูล แคชโมดูลจะบันทึกทั้งขนาดและเวลาเข้าถึง (access time) ล่าสุดของแฟ้มข้อมูลที่ถูกแคชไว้ ดังตัวอย่างในภาพที่ 6 แฟ้มข้อมูลใดที่มีเวลาเข้าถึงเก่าที่สุดจะถือว่าเป็นแฟ้มข้อมูลที่ไม่ได้ถูกใช้งานนานแล้ว (least recently used file) และแคชของแฟ้มข้อมูลนั้นจะถูกลบออกจากระบบเมื่อแคชทั้งหมดมีขนาดรวมกันเกินกว่าที่กำหนดไว้



ภาพที่ 6 ตัวอย่างรายการแฟ้มข้อมูลที่ถูกแคช

4.1.6 การลบแฟ้มข้อมูล

เมื่อแฟ้มข้อมูลในระบบถูกลบออกโดยผู้ใช้ แคช และเมตาเดตาในฐานข้อมูลของแฟ้มข้อมูลนั้นจะถูกลบในทันที บล็อกข้อมูล และระเบียน (record) ที่เกี่ยวข้องในฐานข้อมูลก็จะถูกลบออกไปด้วย แต่ไม่จำเป็นต้องทำในทันที เพราะทำให้ระบบทำงานช้าลงคล้ายกับกรณีของการอัพโหลดบล็อกข้อมูล อาร์เอฟเอสไคลเอนต์จะเรียกเครื่องมืออรรถประโยชน์ในลักษณะของงานเบื้องหลังให้จัดการกับการลบบล็อกข้อมูล และระเบียนที่เกี่ยวข้อง เครื่องมือนี้จะค้นหาระเบียนของบล็อกข้อมูลที่ไม่ถูกอ้างอิงถึง หรือไม่จำเป็นต้องมีในฐานข้อมูล จากนั้นจะส่งคำร้องขอลบบล็อกข้อมูลเหล่านั้นไปที่สตอร์เรจโหนดที่เกี่ยวข้อง หลังจากบล็อกข้อมูลถูกลบเรียบร้อยแล้ว ระเบียนที่เกี่ยวข้องจะถูกลบต่อไป เป็นอันเสร็จสิ้นขั้นตอนในการลบแฟ้มข้อมูล โดยแสดงตัวอย่างการทำงานไว้ในภาพที่ 7



ภาพที่ 7 การลบแฟ้มข้อมูล

4.1.7 ส่วนต่อประสานแบบเรสฟูลเว็บเซอร์วิส

การพัฒนาส่วนต่อประสานนี้จะพัฒนาด้วยภาษาพีเอชพี โดยส่วนต่อประสานในหัวข้อ 3.2.2 จะถูกจับคู่กับฟังก์ชันแต่ละตัวในสคริปต์ภาษาพีเอชพี เช่น

```
{server_name}/file/          →    function file()
{server_name}/file/{bid}     →    function file($bid)
{server_name}/status/        →    function status()
```

การทำเช่นนี้จะทำให้พัฒนาได้สะดวกมากขึ้น เพราะฟังก์ชันแต่ละตัวของส่วนต่อประสานสามารถมารวมอยู่ในสคริปต์พีเอชพีแฟ้มข้อมูลเดียวได้ โดยอาจใช้ตัวแปรโกลบอล (global variable) ดังต่อไปนี้

```
$_SERVER['PATH_INFO'] หรือ $_SERVER['REQUEST_URI']
```

เพื่อรับค่ายูอาร์ไอที่ทางไคลเอนต์ส่งคำร้องขอมา แล้วจึงนำมาแบ่งส่วน และจับคู่กับฟังก์ชันต่าง ๆ ต่อไป

สำหรับส่วนต่อประสานที่พัฒนาขึ้นจะรองรับเพียงแค่เมธอดเก็ท และเมธอดโพสท์เท่านั้น เพราะเว็บเซิร์ฟเวอร์ส่วนใหญ่จะรองรับแค่เมธอด 2 ตัวนี้ โดยใช้ตัวแปรโกลบอล

```
$_SERVER['REQUEST_METHOD']
```

เพื่อช่วยในการพิจารณาว่าคำร้องขอที่ส่งมาเป็นแบบใด ซึ่งตัวแปรนี้จะคืนค่าเป็น `GET` หรือ `POST`

ในส่วนของการรับข้อมูลที่แนบมากับคำร้องขอ จะใช้ตัวแปรโกลบอล

`$_POST`	เมื่อสิ่งที่แนบมาเป็นข้อความ หรือ
`$_FILES`	เมื่อสิ่งที่ส่งมาเป็นบล็อกข้อมูล หรือแฟ้มข้อมูลของฐานข้อมูล

นอกจากนี้อาจต้องมีการกำหนดตัวเลขสถานะของเอชทีทีพี (HTTP status code) ไปกับคำตอบกลับด้วย เพื่อบอกไคลเอนต์ว่าคำร้องขอที่ส่งมานั้นทำงานได้ตามปกติ หรือมีความผิดพลาดเกิดขึ้น โดยระบุคำสั่งด้านล่างนี้ไปพร้อมกับการคืนผลลัพธ์

```
header("HTTP/1.1 $code");
```

โดยแทนที่ $code ด้วยตัวเลข 200 หรือ 201 เพื่อบอกว่าทำงานได้ตามปกติ หรือใช้ช่วงตัวเลขตั้งแต่ 400 ขึ้นไปเพื่อบอกว่ามีความผิดพลาดเกิดขึ้น ไม่สามารถทำงานได้ตามคำร้องขอ โดยตัวเลขสถานะต่าง ๆ มีมากมายซึ่งสามารถศึกษาต่อจากเอกสารของเวิลด์ไวด์เว็บคอนซอร์เทียม (The World Wide Web Consortium, W3C)

4.1.8 การเรียกใช้งานระบบแฟ้มข้อมูลอาร์เอฟเอส

การเมาท์ระบบแฟ้มข้อมูลอาร์เอฟเอสจะใช้คำสั่งต่อไปนี้

```
$ python pcfs.py {mount_point}–o branch={server_name} [–disable-lru]
```

โดยที่ mount_point คือไดเรคทอรีที่ระบบแฟ้มข้อมูลอาร์เอฟเอสจะไปปรากฏ
server_name คือชื่อเซิร์ฟเวอร์ของเมตาเดตาโหนด
--disable-lru คือบอกให้ระบบปิดการใช้งานแคช

เช่น

```
$ python pcfs.py /home/ping/mp –o branch=http://192.168.1.32
```

เนื่องจากพัฒนาต่อจากระบบแฟ้มคลาวด์ส่วนบุคคล ดังนั้นจึงสามารถเมาท์ไดเรคทอรีภายในเครื่องไคลเอนต์ และที่อยู่ต่างเครื่องได้ด้วย โดยคั่นด้วยเครื่องหมายชาร์ป (#) เช่น

```
$ python pcfs.py /home/ping/mp –o branch=http://192.168.1.32#local:/home/ping/test
```

ซึ่งจะเป็นการทำให้แฟ้มข้อมูลในระบบแฟ้มข้อมูลอาร์เอฟเอสมารวมอยู่ที่เดียวกับแฟ้มข้อมูลภายในไดเรคทอรี /home/ping/test ที่อยู่บนเครื่องไคลเอนต์

สำหรับการอันเมาท์จะใช้คำสั่งดังนี้

```
$ fusermount –u {mount_point}
```

4.1.9 **แหล่งดาวน์โหลดซอร์สโค๊ด**

ระบบแฟ้มข้อมูลอาร์เอฟเอสสามารถดาวน์โหลดได้จากเว็บไซต์ด้านล่างนี้

```
https://bitbucket.org/iamping/restfs
```

ภายในจะประกอบไปด้วยเรสโมดูล และโมดูลที่เกี่ยวข้องทั้งหมด รวมทั้งสคริปต์ภาษาพีเอชพีที่ใช้พัฒนาส่วนต่อประสาน

4.2 **เครื่องมือที่ใช้ในการวิจัย**

เครื่องมือที่ใช้ในการวิจัยนี้ ประกอบด้วย

1. อูบุนตู (Ubuntu) [10]

อูบุนตูคือลินุกซ์ดิสทริบิวชันที่ถูกพัฒนาต่อจากเดเบียน (Debian) และแจกจ่ายให้ผู้ใช้ทั่วไปใช้ได้โดยไม่เสียค่าใช้จ่าย อูบุนตูถูกออกแบบมาให้เหมาะกับการใช้งานทั่วไป โดยการวิจัยนี้

ใช้อูบุนตู รุ่น 12.04 เป็นระบบปฏิบัติการหลักในการพัฒนาอาร์เอฟเอสไคลเอนต์ และใช้ลินุกซ์เคอร์เนล (Linux kernel) เวอร์ชัน 3.2.0-24

2. ไอโอโซน (IOzone) [11]

ไอโอโซน เป็นเครื่องมือที่ใช้สำหรับทดสอบประสิทธิภาพของระบบแฟ้มข้อมูล (file system benchmark tool) โดยการวัดค่าต่าง ๆ ที่เกิดจากการดำเนินการทางแฟ้มข้อมูล (file operation) เช่น การอ่านแฟ้มข้อมูล การเขียนแฟ้มข้อมูล เป็นต้น สำหรับการวิจัยนี้ใช้ไอโอโซน รุ่น 3.373

3. ไทม์อิทโมดูล (Timeit module) [12]

ไทม์อิทโมดูลเป็นโมดูลที่ใช้สำหรับจับเวลาการทำงานของโปรแกรมที่พัฒนาด้วยภาษาไพธอน โดยมีเมธอดที่ช่วยอำนวยความสะดวกเช่น เมธอด timeit ซึ่งสามารถกำหนดจำนวนรอบที่จะให้โปรแกรมทำงาน และจับเวลาการทำงานของทุกรอบรวมกันได้ และเมธอด repeat เมธอดนี้สามารถกำหนดจำนวนรอบของการเรียกเมธอด timeit ได้ และคืนค่าเป็นรายการของเวลาการทำงานของเมธอด timeit ในแต่ละรอบ

4. ฟิวส์ (Fuse)

ฟิวส์คือไลบรารีที่ใช้สำหรับการพัฒนาระบบแฟ้มข้อมูลบนระดับผู้ใช้ (user level) ระบบแฟ้มที่พัฒนาด้วยฟิวส์สามารถทำงานได้โดยไม่ต้องอาศัยการเข้าถึงสิทธิ์ของรูท (root permission) ปัจจุบันฟิวส์รองรับภาษาต่าง ๆ มากมาย ซึ่งการวิจัยนี้เลือกใช้ฟิวส์ไพธอน (Fuse Python) [13] ในการพัฒนาอาร์เอฟเอสไคลเอนต์

5. ไพเคิร์ล (PycURL) [14]

ไพเคิร์ลคือไลบรารีในภาษาไพธอนที่พัฒนาเป็นส่วนต่อประสานของไลบรารีเคิร์ล (cURL) [15] เพื่อให้สามารถนำไลบรารีเคิร์ลมาใช้ในโปรแกรมภาษาไพธอนได้ โดยมีความสามารถในการติดต่อกับโพรโตคอลต่าง ๆ เช่น เอฟทีพี (FTP) เอสเอสเอช (SSH) เอชทีทีพี (HTTP) เป็นต้น

บทที่ 5
วิธีประเมินการวิจัย สรุป และอภิปรายผล

ในบทนี้กล่าวถึงวิธีประเมินการวิจัย สรุป และอภิปรายผล ดังนี้

5.1 วิธีประเมินการวิจัย

วิธีประเมินการวิจัยถูกแบ่งออกเป็น 3 ส่วน ดังนี้

5.1.1 การจัดเตรียมการประเมินการวิจัย

การประเมินการวิจัยใช้แล็ปท็อปคอมพิวเตอร์จำนวน 3 เครื่อง โดยมี 1 เครื่องทำหน้าที่เป็นทั้งไคลเอนต์ และเมตาเดตาโหนด ส่วนเครื่องที่เหลือทำหน้าที่เป็นสตอร์เรจโหนด ซึ่งมีรายละเอียดข้อกำหนดคุณลักษณะของระบบ (system specification) ดังตารางที่ 3 โดยทุกเครื่องเชื่อมต่อกันในลักษณะข่ายงานแบบดาว (star network) ผ่านทางอีเทอร์เน็ตสวิตช์ (Ethernet switch) ที่มีแบนด์วิดท์ 10/100 เมกะบิตต่อวินาที (Mbps)

ตารางที่ 2 ตารางสรุปรายละเอียดของแล็ปท็อปคอมพิวเตอร์ที่ใช้ในการประเมินการวิจัย

รุ่น	BenQ S41-M60	Lenovo G475	HP Probook 5330m
หน่วยประมวลผลกลาง	Intel® Core™2 Duo T8100	AMD E-350	Intel® Core™i3-2330
แรม	4GB DDRII	2GB DDRII	4GB DDRIII
ฮาร์ดดิสก์	250 GB SATA	500 GB SATA	320 GB SATA
ระบบปฏิบัติการ	Ubuntu 12.04 32-bit	Ubuntu 12.04 64-bit	Windows 7 32-bit
หน้าที่	ไคลเอนต์, เมตาเดตาโหนด	สตอร์เรจโหนด	สตอร์เรจโหนด

`

5.1.2 การทดสอบหาขนาดของแฟ้มข้อมูล และขนาดของบล็อกข้อมูลที่เหมาะสม

ระบบแฟ้มข้อมูลอาร์เอฟเอสรองรับขนาดของแฟ้มข้อมูล และขนาดของบล็อกข้อมูลที่หลากหลาย แต่ถ้าเลือกใช้ขนาดที่ไม่เหมาะสมก็จะทำให้ประสิทธิภาพโดยรวมของระบบลดลง ดังนั้นจึงต้องมีการทดสอบเพื่อหาขนาดของแฟ้มข้อมูล และบล็อกข้อมูล โดยแบ่งออกเป็น 2 การทดสอบย่อย คือ การทดสอบสำเนา (copy) แฟ้มข้อมูลจากเครื่องไคลเอนต์ไปที่สตอร์เรจโหนด

และการทดสอบดาวน์โหลด (download) แฟ้มข้อมูลจากสตอร์เรจโหนดมาที่เครื่องไคลเอนต์ โดยวัดปริมาณงาน (throughput) ที่ได้จากทั้ง 2 การทดสอบด้วยไทม์อิทโมดูล

แฟ้มข้อมูลที่ใช้ในการทดสอบมีขนาดตั้งแต่ 2 เมกะไบต์ จนถึง 256 เมกะไบต์ แต่ละขนาดของแฟ้มข้อมูลจะใช้ขนาดของบล็อกข้อมูลตั้งแต่ 256 กิโลไบต์ (kilobytes) จนถึง 2048 กิโลไบต์ การทดสอบจะถูกทดสอบทั้งหมด 30 รอบ จากนั้นจึงนำเวลาที่วัดได้มาหาค่าเฉลี่ย แล้วจึงนำมาคำนวณหาค่าปริมาณงานโดยมีหน่วยเป็นกิโลไบต์ต่อวินาที (kB/s)

5.1.3 การเปรียบเทียบประสิทธิภาพของระบบแฟ้มข้อมูลอาร์เอฟเอสกับระบบแฟ้มข้อมูลอื่น

การวิจัยนี้จะใช้เครื่องมือที่ชื่อว่า ไอโอโซน สำหรับการหาประสิทธิภาพของระบบแฟ้มข้อมูลอาร์เอฟเอสที่เปิดและปิดการใช้งานแคช โดยการวัดปริมาณงาน (throughput) ของการดำเนินการทางแฟ้มข้อมูล (file operation) ได้แก่ การเขียน (write) การเขียนซ้ำ (rewrite) การอ่าน (read) การอ่านซ้ำ (reread) การเขียนโดยสุ่ม (random write) และการอ่านโดยสุ่ม (random read) แต่ละการดำเนินการจะทดสอบกับแฟ้มข้อมูลจำนวน 1 แฟ้มทั้งหมด 30 รอบ โดยใช้ขนาดของแฟ้มข้อมูลที่ 8 เมกะไบต์ 16 เมกะไบต์ และ 256 เมกะไบต์ ขนาดของบล็อกข้อมูลที่ใช้จะขึ้นอยู่กับผลการทดสอบในข้อ 5.2.1 ส่วนขนาดของการโอนถ่ายข้อมูล (transfer size) อยู่ที่ 16 กิโลไบต์ และตั้งค่าขนาดแคชของหน่วยประมวลผล (processor cache) ไว้ที่ 1024 กิโลไบต์ โดยสามารถสั่งให้ไอโอโซนทำงานผ่านเทอร์มินัล (terminal) บนระบบปฏิบัติการอูบุนตูด้วยคำสั่งดังต่อไปนี้

```
$ iozone -a -i 0 -i 1 -i 2 -r 16k -f {mount_point/test_file} -s {file_size}
```

โดยที่

a	คือการสั่งให้ทดสอบทีละแฟ้มข้อมูล
i	คือการเลือกโหมดการดำเนินการทางแฟ้มข้อมูล
r	คือการกำหนดขนาดของการโอนถ่ายข้อมูล
f	คือการกำหนดตำแหน่งของแฟ้มข้อมูลที่จะมีการสร้างขึ้นใหม่เพื่อทดสอบ โดยจะเลือกไปที่เมาท์พอยต์ (mount point) ของระบบแฟ้มข้อมูลที่ต้องการทดสอบ
s	คือการกำหนดขนาดของแฟ้มข้อมูลที่ใช้ในการทดสอบ

ผลที่ได้จะมาคำนวณค่าเฉลี่ยแล้วนำไปเปรียบเทียบกับระบบแฟ้มข้อมูล ดังต่อไปนี้

- ระบบแฟ้มข้อมูลแบบโลคัล (local file system) ได้แก่ ระบบแฟ้มข้อมูลอีเอ็กซ์ทีโฟร์ (ext4) ซึ่งเป็นระบบแฟ้มข้อมูลขั้นพื้นฐานของระบบปฏิบัติการตระกูลลินุกซ์ในปัจจุบัน และระบบแฟ้มข้อมูลดัมมี (dummyfs) ซึ่งเป็นระบบแฟ้มข้อมูลแบบลูปแบค (loopback) ที่พัฒนาด้วยฟิวส์
- ระบบแฟ้มข้อมูลแบบรีโมต (remote file system) ได้แก่ ระบบแฟ้มข้อมูลเอสเอสเอช (SSHFS) ซึ่งเป็นระบบแฟ้มข้อมูลที่ใช้โปรโตคอลเอสเอฟทีพี (SFTP) ในการติดต่อกับอุปกรณ์ต่างเครื่อง และระบบแฟ้มข้อมูลตัวนี้พัฒนาด้วยฟิวส์

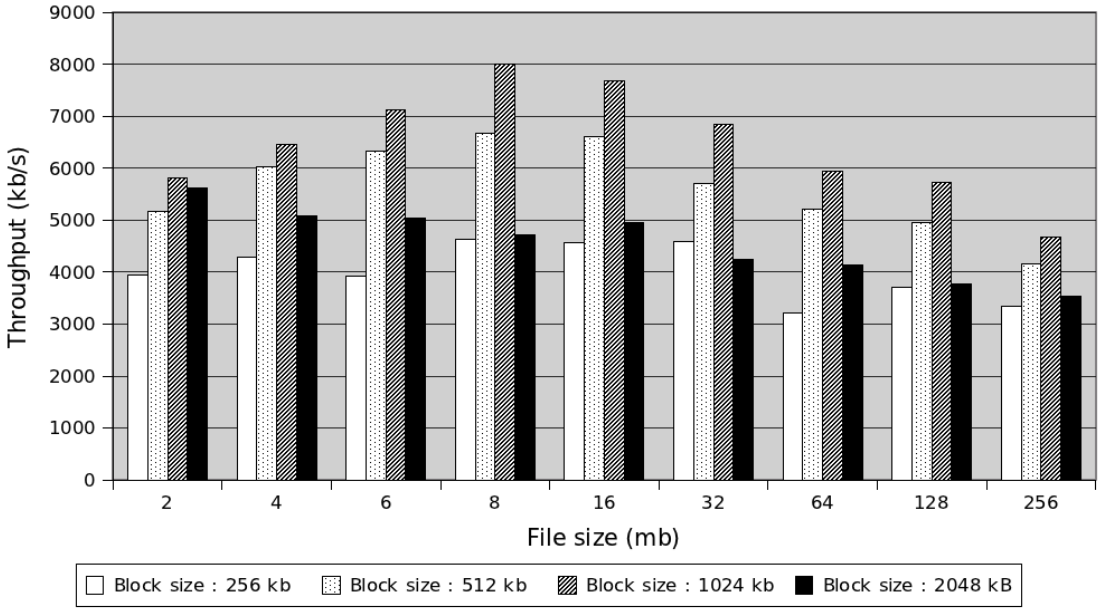
สำหรับเหตุผลที่เลือกทดสอบกับแฟ้มข้อมูลครั้งละ 1 แฟ้ม คือ ระบบแฟ้มข้อมูลอาร์เอฟเอสยังไม่รองรับการทำงานแบบมัลติเธรด (multithread) ในขณะที่ระบบแฟ้มข้อมูลอื่นสามารถทำงานแบบมัลติเธรดได้ (ยกเว้นระบบแฟ้มข้อมูลดัมมี) ดังนั้น เพื่อให้สามารถเปรียบเทียบกันได้จึงต้องกำหนดให้ทดสอบกับแฟ้มข้อมูลเพียงแฟ้มเดียวผ่านทางคำสั่งของไอโอโซน

5.2 **สรุปผลการวิจัย**

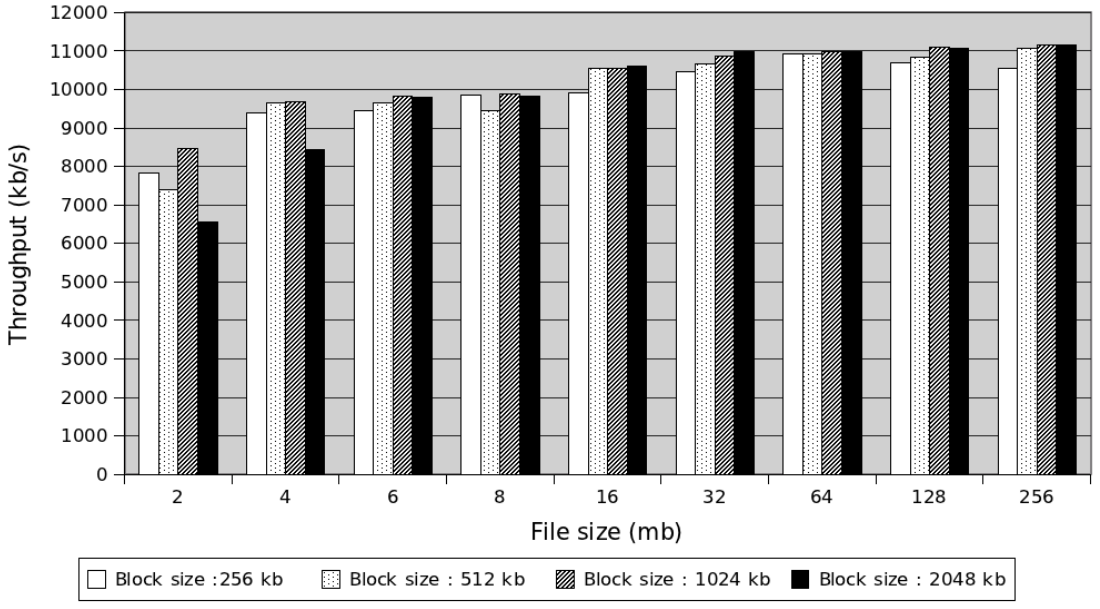
ผลการวิจัยแบ่งออกเป็น 2 หัวข้อตามวิธีประเมินการวิจัย ดังนี้

5.2.1 **ผลของการทดสอบหาขนาดของแฟ้มข้อมูล และขนาดของบล็อกข้อมูลที่เหมาะสม**

ข้อมูลปริมาณงานของการทดสอบสำเนาแฟ้มข้อมูลจากเครื่องไคลเอนต์ไปที่สตอร์เรจโหนดแสดงดังภาพที่ 8 และปริมาณงานของการทดสอบดาวน์โหลดแฟ้มข้อมูลจากสตอร์เรจโหนดมาที่เครื่องไคลเอนต์แสดงดังภาพที่ 9 (ข้อมูลที่เป็นตัวเลขแสดงไว้ในภาคผนวก) เมื่อพิจารณาจากทั้ง 2 รูปนี้จะพบว่าแฟ้มข้อมูลช่วง 8 เมกะไบต์จนถึง 16 เมกะไบต์ และบล็อกข้อมูลขนาด 1024 กิโลไบต์เป็นขนาดที่เหมาะสมที่สุด



ภาพที่ 8 ปริมาณงานของการทดสอบสำเนาแฟ้มข้อมูล

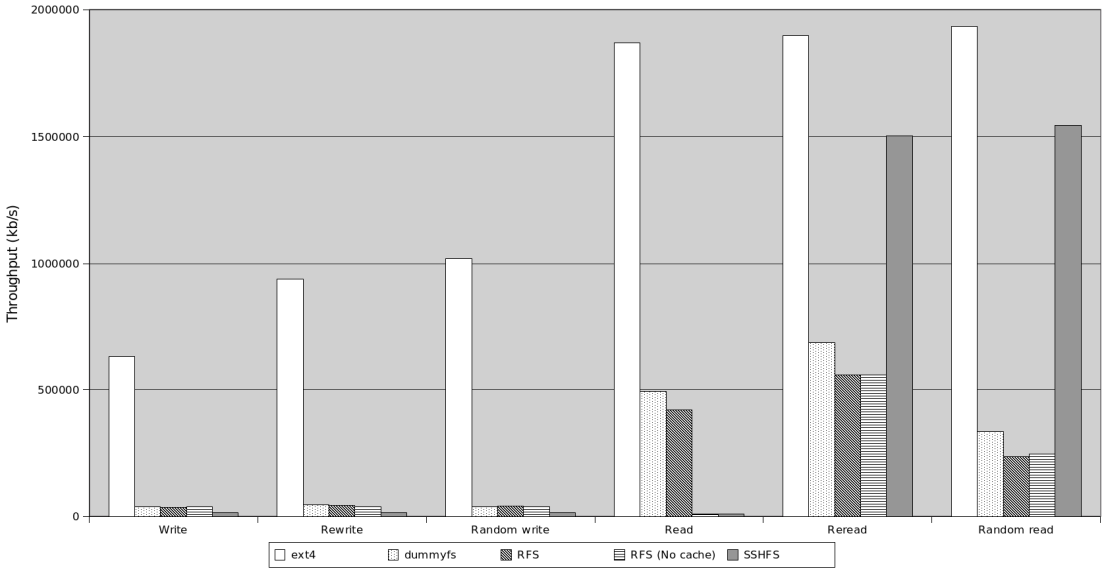


ภาพที่ 9 ปริมาณงานของการทดสอบดาวน์โหลดแฟ้มข้อมูล

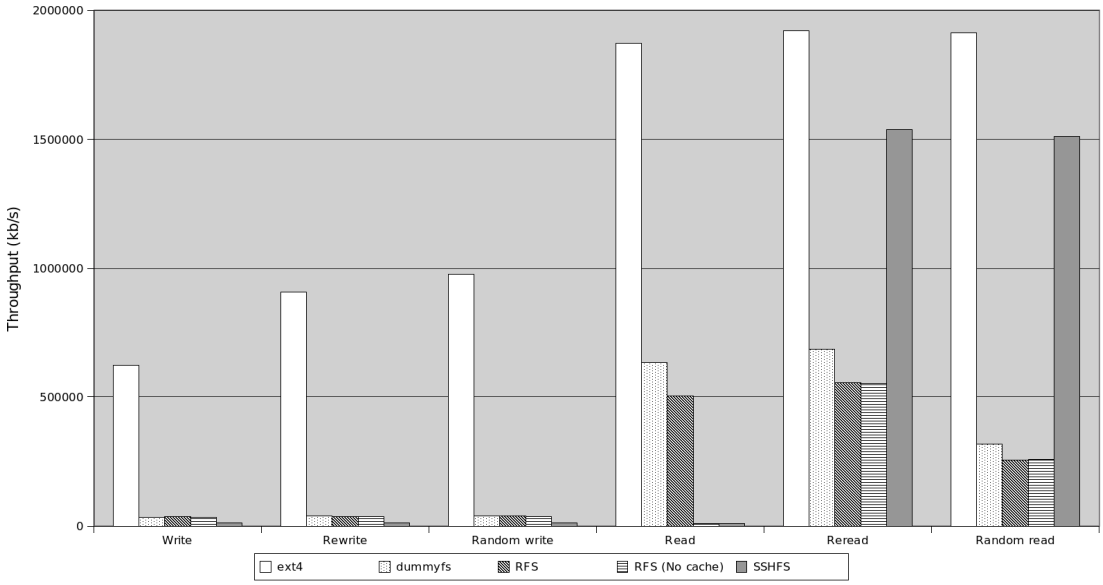
5.2.2 ผลของการทดสอบของการดำเนินการทางแฟ้มข้อมูล

ปริมาณงานของการดำเนินการทางแฟ้มข้อมูลของระบบแฟ้มข้อมูลที่นำมาเปรียบเทียบแสดงดังภาพที่ 10 ภาพที่ 11 และภาพที่ 12 การเปรียบเทียบนี้ใช้บล็อกข้อมูลที่มีขนาด 1 เมกะ

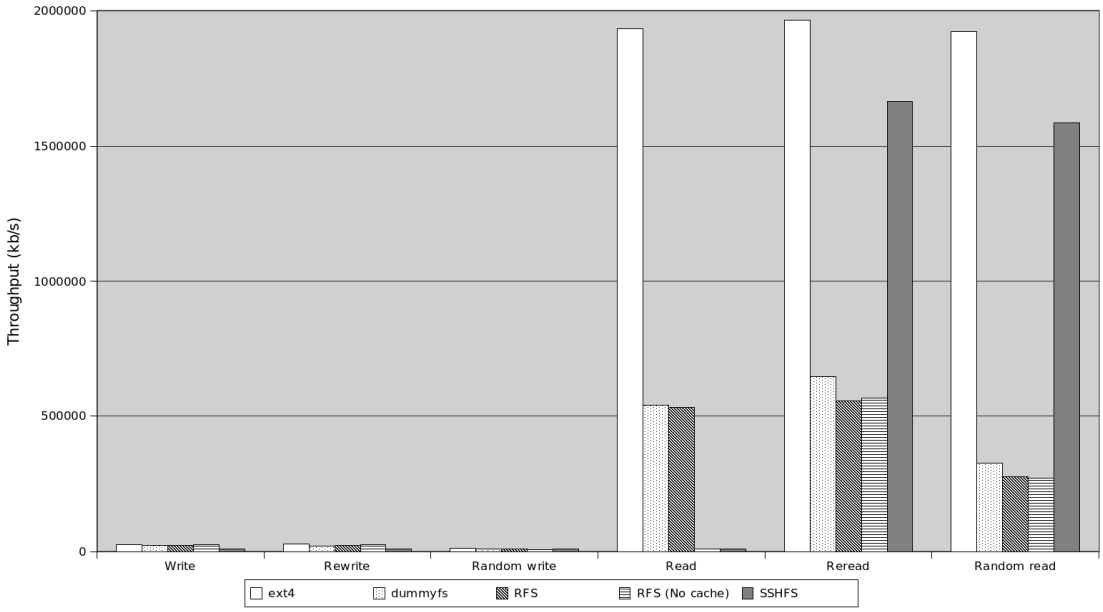
ไบต์ในการทดสอบโดยอ้างอิงจากผลในข้อที่ 5.2.1 โดยข้อมูลที่นำมาวาดกราฟถูกแสดงไว้ในภาคผนวก



ภาพที่ 10 ปริมาณงานของการดำเนินการทางแฟ้มข้อมูลที่แฟ้มข้อมูลขนาด 8 เมกะไบต์



ภาพที่ 11 ปริมาณงานของการดำเนินการทางแฟ้มข้อมูลที่แฟ้มข้อมูลขนาด 16 เมกะไบต์



ภาพที่ 12 ปริมาณงานของการดำเนินการทางแฟ้มข้อมูลที่แฟ้มข้อมูลขนาด 256 เมกะไบต์

5.3 อภิปรายผล

การอภิปรายผลแบ่งออกเป็น 2 หัวข้อ ตามวิธีประเมินการวิจัย ดังนี้

5.3.1 ขนาดของแฟ้มข้อมูล และบล็อกข้อมูลที่เหมาะสม

เมื่อพิจารณาผลของการทดสอบสำเนาแฟ้มข้อมูลจากเครื่องไคลเอนต์ไปที่สตอร์เรจโหนด จะพบว่าขนาดแฟ้มข้อมูลในช่วง 8 เมกะไบต์ และ 16 เมกะไบต์ และบล็อกข้อมูลที่ขนาด 1024 กิโลไบต์ทำให้ได้ปริมาณงานที่สูง แต่ผลของการทดสอบดาวน์โหลดแฟ้มข้อมูลจากสตอร์เรจโหนด บ่งบอกว่าขนาดที่ใหญ่ขึ้นของแฟ้มข้อมูลส่งผลให้ปริมาณงานสูงขึ้นด้วย ในขณะที่ขนาดของบล็อกข้อมูลที่ได้ปริมาณงานสูงสุดจะอยู่ในช่วง 1024 กิโลไบต์ และ 2048 กิโลไบต์

สำหรับขนาดของบล็อกข้อมูล เมื่อบล็อกข้อมูลมีขนาดเล็กลง จำนวนของบล็อกข้อมูลจะมีมากขึ้น ทำให้โอเวอร์เฮดที่อาจเกิดจากการทำงานภายในระบบ จำนวนการติดต่อกับสตอร์เรจโหนด และจากส่วนหัวของคำร้องขอ (HTTP header) เพิ่มขึ้น จึงส่งผลให้ปริมาณงานที่ได้ลดลง

เมื่อบล็อกข้อมูลมีขนาดใหญ่ขึ้น โอเวอร์เฮดที่เกิดขึ้นจะน้อยกว่า เพราะจำนวนของบล็อกข้อมูลน้อยกว่า อย่างไรก็ตาม ถ้าบล็อกมีขนาดใหญ่เกินไปและแฟ้มข้อมูลมีขนาดเล็ก อาจส่งผลให้ใช้แบนด์วิดท์ในการอัพโหลดบล็อกข้อมูลไปยังสตอร์เรจโหนดได้ไม่เต็มที่ ในขณะเดียวกันถ้า

แฟ้มข้อมูลมีขนาดใหญ่มาก จำนวนบล็อกข้อมูลก็จะมากขึ้นตามไปด้วย ส่งผลให้ระบบใช้ทรัพยากรของหน่วยความจำภายในเครื่องมากเช่นกัน ทำให้ระบบทำงานด้วยประสิทธิภาพที่ลดลง ดังนั้นปริมาณงานที่วัดได้จึงลดลงด้วย ซึ่งปัญหาในส่วนนี้อาจแก้ไขด้วยการทำโค๊ดออพติไมซิง (code optimizing) เพื่อปรับปรุงการทำงานภายในระบบแฟ้มข้อมูลอาร์เอฟเอสให้ใช้หน่วยความจำลดลง

สำหรับปริมาณงานที่วัดได้นั้นจะขึ้นอยู่กับแบนด์วิดท์ของอีเทอร์เน็ตสวิตช์ที่ใช้ในการทดสอบ ซึ่งถูกจำกัดอยู่ที่ความเร็ว 100 เมกะบิตต่อวินาที ปริมาณงานที่ได้จากทั้ง 2 การทดสอบจึงไม่เกินขีดจำกัดนี้ เมื่อสังเกตผลของการทดสอบดาวน์โหลดแฟ้มข้อมูล ปริมาณงานของการดาวน์โหลดแฟ้มข้อมูลขนาด 8 เมกะไบต์ และ 16 เมกะไบต์ต่ำกว่าโดยเฉลี่ยประมาณ 10% เมื่อเทียบกับปริมาณงานของการดาวน์โหลดแฟ้มข้อมูลที่ใหญ่กว่า ดังนั้นเมื่อพิจารณาผลของทั้ง 2 การทดสอบ และจากการทดลองใช้งานจริง สรุปได้ว่าแฟ้มข้อมูลที่มีขนาดอยู่ในช่วง 8 เมกะไบต์ และ 16 เมกะไบต์เหมาะสมสำหรับระบบแฟ้มข้อมูลอาร์เอฟเอสในเวอร์ชันปัจจุบัน

5.3.2 ประสิทธิภาพของระบบแฟ้มข้อมูลอาร์เอฟเอสเทียบกับระบบแฟ้มข้อมูลอื่น

เมื่อพิจารณาผลของการดำเนินการทางแฟ้มข้อมูลที่แฟ้มข้อมูลขนาด 8 เมกะไบต์ในภาพที่ 10 พบว่า การเขียน การเขียนซ้ำ และการเขียนโดยสุ่มของระบบแฟ้มข้อมูลอีเอ็กซ์ทีโฟร์จะมีปริมาณงานสูงที่สุด เนื่องจากรองรับการใช้งานบัฟเฟอร์แคช (buffer cache) จึงช่วยให้เขียนแฟ้มข้อมูลได้รวดเร็วยิ่งขึ้น

ส่วนปริมาณงานของระบบแฟ้มข้อมูลดัมมีจะน้อยกว่าเพราะไม่สามารถใช้งานบัฟเฟอร์แคชได้ สำหรับระบบแฟ้มข้อมูลเอสเอสเอชนั้น เนื่องจากเป็นระบบแฟ้มข้อมูลแบบรีโมต การเขียนแฟ้มข้อมูลจึงเป็นการเขียนไปยังอีกเครื่องหนึ่งที่เชื่อมต่อกับเครื่องไคลเอนต์โดยตรง ปริมาณงานที่วัดได้จึงถูกจำกัดด้วยแบนด์วิดท์ของอีเทอร์เน็ตสวิตช์

สำหรับระบบแฟ้มข้อมูลอาร์เอฟเอสทั้งขณะที่เปิดและปิดการใช้งานแคช จะมีการเขียนข้อมูลไปบนแฟ้มข้อมูลชั่วคราวบนเครื่องก่อน แล้วจึงอัพโหลดข้อมูลเหล่านั้นไปที่สตอร์เรจโหนดเมื่อการเขียนเสร็จสิ้น ปริมาณงานที่ได้จากการเขียนจึงควรเทียบเท่ากับปริมาณงานของระบบ

แฟ้มข้อมูลเอสเอสเอช แต่ผลที่ได้จริง ๆ กลับเทียบเท่ากับระบบแฟ้มข้อมูลดัมมี ที่เป็นเช่นนี้เพราะว่าไอโอโซนในเวอร์ชันนี้วัดปริมาณงานได้แค่ช่วงที่มีการเขียนข้อมูลเท่านั้น จึงไม่สามารถวัดปริมาณงานในช่วงที่มีการส่งข้อมูลไปที่สตอร์เรจโหนดได้

ในการทดสอบการอ่านแฟ้มข้อมูล พบว่า ระบบแฟ้มข้อมูลอีเอ็กซ์ทีโฟร์ยังคงมีปริมาณงานสูงที่สุดเช่นเดิม เนื่องจากบัฟเฟอร์แคช ส่วนระบบแฟ้มข้อมูลดัมมีจะอ่านได้ช้าว่าอีเอ็กซ์ทีโฟร์ แต่ก็ยังถือว่าอ่านได้รวดเร็วดีแม้จะไม่มีบัฟเฟอร์แคชก็ตาม

ระบบแฟ้มข้อมูลอาร์เอฟเอสแบบมีแคช จะอ่านแฟ้มข้อมูลได้เร็วเทียบเท่ากับระบบแฟ้มข้อมูลดัมมี เพราะเป็นการอ่านแฟ้มข้อมูลที่ถูกแคชไว้ด้วยแคชโมดูลของอาร์เอฟเอสไคลเอนต์ แต่เมื่อปิดการใช้งานแคช ระบบจะต้องเสียเวลาดาวน์โหลดข้อมูลจากสตอร์เรจโหนดก่อนถึงจะอ่านได้ ปริมาณงานที่ได้จึงถูกจำกัดด้วยแบนด์วิดท์ของอีเทอร์เน็ตสวิตช์ โดยปริมาณงานของการอ่านแฟ้มข้อมูลของระบบแฟ้มข้อมูลแบบเอสเอสเอชก็สามารถอธิบายได้ด้วยเหตุผลเดียวกัน

สำหรับปริมาณงานของการอ่านซ้ำ และการอ่านโดยสุ่มของระบบแฟ้มข้อมูลทุกตัวจะสูงกว่าเมื่อเทียบกับการอ่านครั้งแรก เพราะตัวระบบปฏิบัติการสามารถแคชข้อมูลที่เคยอ่านไว้ในหน่วยความจำได้ด้วย โดยเฉพาะระบบแฟ้มข้อมูลอีเอ็กซ์ทีโฟร์ และระบบแฟ้มข้อมูลเอสเอสเอชซึ่งจะมีปริมาณสูงเป็นพิเศษเนื่องจากรองรับการใช้งานบัฟเฟอร์แคชด้วย

สำหรับผลของการดำเนินการทางแฟ้มข้อมูลที่แฟ้มข้อมูลขนาด 16 เมกะไบต์ในภาพที่ 11 ก็สามารถอธิบายได้ด้วยเหตุผลเดียวกับการทดสอบที่แฟ้มข้อมูลขนาด 8 เมกะไบต์

เมื่อพิจารณาผลของการดำเนินการทางแฟ้มข้อมูลที่แฟ้มข้อมูลขนาด 256 เมกะไบต์ในภาพที่ 12 จะสังเกตเห็นว่าการเขียนแฟ้มข้อมูลของระบบแฟ้มข้อมูลทุกตัวจะเทียบเท่ากัน (ยกเว้นระบบแฟ้มข้อมูลเอสเอสเอชซึ่งปริมาณงานถูกจำกัดด้วยแบนด์วิดท์ของอีเทอร์เน็ตสวิตช์) ที่เป็นเช่นนี้อาจเป็นเพราะว่าแฟ้มข้อมูลมีขนาดใหญ่เกินกว่าที่บัฟเฟอร์แคชจะรองรับได้ ดังนั้นจึงเหมือนกับว่าระบบแฟ้มข้อมูลอีเอ็กซ์ทีโฟร์เขียนข้อมูลลงบนดิสก์โดยตรง

โดยรวมแล้วถึงแม้ระบบแฟ้มข้อมูลอาร์เอฟเอสจะทำงานได้ช้ากว่าระบบแฟ้มข้อมูลอีเอ็กซ์ทีโฟร์มาก จากการใช้งานจริงแล้วถือว่าทำงานได้รวดเร็วดีไม่ต่างจากระบบแฟ้มข้อมูลดัมมีซึ่งเป็นระบบแฟ้มข้อมูลแบบโลคัลมากนัก แต่จะอาจจะช้ากว่าบ้างเนื่องจากระบบแฟ้มข้อมูลอาร์เอฟเอสยังมีโอเวอร์เฮดอื่น ๆ อีก เช่น โอเวอร์เฮดที่เกิดจากการติดต่อกับฐานข้อมูลเอสคิวไลท์ เมื่อมีการ

เข้าถึงแฟ้มข้อมูล หรือมีการแก้ไขเกิดขึ้น อาร์เอฟเอสไคลเอนต์จะบางสิ่งต่อไปนี้กับฐานข้อมูล ได้แก่ การเพิ่มระเบียน การเปลี่ยนแปลงค่าของระเบียนที่มีอยู่ การลบระเบียน และการยืนยันการเปลี่ยนแปลงที่เกิดขึ้น นอกจากนี้ยังเกี่ยวกับโอเวอร์เฮดที่เกิดจากการเรียกอ็อบเจกต์ภายในโปรแกรมของระบบแฟ้มข้อมูลอาร์เอฟเอส ซึ่งการทำงานใด ๆ จะมีการเรียกผ่านเมธอดหลายชั้น และยังมีไลบรารีต่าง ๆ อีกหลายตัวที่ต้องเรียกใช้ด้วย จึงมีส่วนทำให้มีปริมาณงานน้อยกว่า

สำหรับแนวทางในการปรับปรุงประสิทธิภาพของระบบแฟ้มข้อมูลอาร์เอฟเอสในอนาคต คือ ปรับปรุงให้ระบบรองรับการใช้งานบัฟเฟอร์แคช ซึ่งจะช่วยให้อ่านข้อมูลและเขียนข้อมูลได้รวดเร็วยิ่งขึ้น อาจจะต้องปรับปรุงให้รองรับการใช้งานมัลติเธรดด้วย ซึ่งจะทำให้สามารถทดสอบเขียนแฟ้มข้อมูลได้ตั้งแต่สองแฟ้มขึ้นไปได้พร้อม ๆ กัน และอาจจะต้องเลือกเครื่องมือทดสอบตัวอื่น เพื่อทดสอบในด้านอื่น ๆ อีกในอนาคต เพราะไอโอโซนเพียงแค่ทดสอบการดำเนินการทางแฟ้มข้อมูลเท่านั้น อาจจะไม่ได้สะท้อนภาพการใช้งานระบบแฟ้มข้อมูลในความเป็นจริงมากนัก

บทที่ 6
บทสรุป

ในบทนี้จะกล่าวถึงสิ่งที่ได้จากการวิจัย แนวทางการวิจัยต่อ และบทสรุปของการวิจัยนี้ดังต่อไปนี้

6.1 **สิ่งที่ได้จากการวิจัย** (Contribution)

สิ่งที่ได้จากการวิจัย ได้แก่

1. แสดงให้เห็นว่าสามารถนำเว็บเซิร์ฟเวอร์มาใช้งานในด้านอื่นได้ เช่น ใช้เป็นแบคเอนด์ (backend) ของระบบแฟ้มข้อมูลอาร์เอฟเอสในการวิจัยนี้ เป็นต้น โดยที่เซิร์ฟเวอร์ที่เลือกมาอาจเป็นบริการที่ไม่เสียค่าใช้จ่ายก็ได้
2. สามารถนำแนวทางการออกแบบส่วนต่อประสานของเว็บเซอร์วิสแบบเรสฟูลในการวิจัยนี้ไปประยุกต์ใช้กับงานประเภทอื่นได้
3. ได้ระบบแฟ้มข้อมูลรวมชนิดใหม่ที่ใช้งานได้ดีสำหรับงานทั่วไปที่ขนาดของแฟ้มข้อมูลไม่ใหญ่มาก และยังช่วยอำนวยความสะดวกแก่ผู้ใช้ที่ต้องทำงานกับข้อมูลชุดเดิม แต่อยู่ต่างเครื่องได้

6.2 **แนวทางการวิจัยต่อ**

ระบบแฟ้มข้อมูลอาร์เอฟเอส ยังสามารถนำมาพัฒนาต่อได้ ดังต่อไนปี้

1. ความปลอดภัยของระบบมี 2 ประเภท คือ ความปลอดภัยของช่องทางการสื่อสาร และความปลอดภัยของข้อมูล สำหรับความปลอดภัยของช่องทางการสื่อสาร ส่วนต่อประสานแบบเรสฟูลเว็บเซอร์วิสสามารถดีพลอย (deploy) บนโพรโตคอลเอชทีทีพีเอส (HTTPS) ซึ่งเป็นการติดต่อสื่อสารบนช่องทางที่ปลอดภัย โดยไลบรารีไพเคิร์ล (PycURL) ที่ใช้ในอาร์เอฟเอสไคลเอนต์รองรับโพรโตคอลนี้อยู่แล้ว สำหรับความปลอดภัยของข้อมูลอาจต้องพัฒนาเพิ่มเติม โดยอาจประยุกต์ใช้อัลกอริทึมเออีเอส (AES) หรือโบลวฟิช (Blowsfish) สำหรับการเข้ารหัสข้อมูลในส่วนของรีซอร์สโมดูล

   นอกจากนี้อาจพัฒนาระบบพิสูจน์ตัวตนแบบโทเคนเบส (token-based authentication) ซึ่งจะทำให้ผู้ใช้ที่ได้รับอนุญาตเท่านั้นที่จะเข้าถึงแฟ้มข้อมูลได้ โดยอาจพัฒนาเมตาเดตาโหนดให้สามารถเก็บข้อมูลชื่อผู้ใช้และรหัสผ่านเอาไว้ รวมถึง

การกำหนดสิทธิ์ต่าง ๆ ในการเข้าใช้งาน เมื่อผู้ใช้เข้าใช้งานระบบแฟ้มข้อมูลอาร์เอฟเอสก็ต้องส่งคำร้องขอเพื่อล็อกอินที่เมตาเดตาโหนดก่อน เพื่อรับโทเคนที่ได้รับอนุญาต โดยโทเคนนี้จะอยู่ในลักษณะข้อความแบบสุ่ม และต้องแนบไปกับคำร้องขอทุกครั้งที่ต้องการเข้าถึงทรัพยากรในระบบ เช่น เมื่อไคลเอนต์มีการดาวน์โหลดบล็อกข้อมูล รีซอร์สโมดูลจะแนบโทเคนไปกับคำร้องขอที่ส่งไปที่สตอร์เรจโหนด สตอร์เรจโหนดจะนำโทเคนที่ได้รับไปตรวจสอบอีกทีกับเมตาเดตาโหนดว่า โทเคนนี้มีสิทธิ์ในการเข้าใจงานอยู่หรือไม่ เป็นต้น และอาจมีการกำหนดอายุของโทเคนหรือสิทธิ์ในการเข้าถึงแฟ้มข้อมูลภายในไดเรคทอรีในระบบเอาไว้ด้วย เพื่อป้องกันผู้อื่นที่บังเอิญได้รับโทเคนไปแล้วจะเข้าถึงทรัพยากรของเราได้ตลอดเวลา

นอกจากนี้อาจใช้การล็อกอินแบบซิงเกิลซายออน (Single sign-on) ซึ่งช่วยให้ผู้ใช้ล็อกอินเพียงครั้งเดียว แล้วสามารถเข้าถึงบริการต่าง ๆ ได้โดยไม่ต้องล็อกอินใหม่ โดยอาจแม็พทรัพยากรในโหนดทุกตัวเข้ากับบัญชีของกูเกิล (Google account) ข้อดีคือเราไม่จำเป็นต้องจัดเก็บข้อมูลชื่อผู้ใช้และรหัสผ่านเอาไว้เอง และกูเกิลยังมีส่วนต่อประสานเพื่อช่วยอำนวยความสะดวกให้ด้วย ทำให้ไม่ต้องพัฒนาระบบพิสูจน์ตัวตนใหม่ทั้งหมด

2. ระบบในขณะนี้เก็บข้อมูลของทุกแฟ้มข้อมูลเพียงชุดเดียวเท่านั้น ดังนั้น อาจไม่สามารถเข้าถึงแฟ้มข้อมูลที่ต้องการเมื่อไม่สามารถต่อต่อกับสตอร์เรจโหนดได้ ยกเว้นแฟ้มข้อมูลที่ถูกแคชไว้ในระบบ อาจแก้ปัญหาโดยพัฒนาการสำเนาข้อมูล (replication) อย่างง่ายโดยการสำเนาข้อมูลของโหนดหลักไปที่อีกโหนดที่กำหนดให้เป็นโหนดรอง (secondary node)

6.3 บทสรุป

ระบบแฟ้มข้อมูลอาร์เอฟเอสเหมาะกับการใช้งานทั่วไป และมีข้อดีหลายข้อด้วยกัน เช่น แฟ้มข้อมูลที่ถูกแก้ไขหรือสร้างผ่านระบบแฟ้มข้อมูลอาร์เอฟเอสจะถูกแบคอัพ (back up) ไปอยู่บนเซิร์ฟเวอร์หรือสตอร์เรจโหนดโดยอัตโนมัติผ่านทางส่วนต่อประสานแบบเรสฟูล ระบบแฟ้มข้อมูลอาร์เอฟเอสยังรองรับการขยายตัวของระบบโดยไม่ขึ้นกับพื้นที่ว่างบนเครื่องไคลเอนต์ที่ใช้งานอยู่ และผู้ใช้สามารถหาพีเอชพีเซิร์ฟเวอร์ได้โดยง่าย เพราะเป็นที่นิยมมาก การแคชข้อมูลด้วยอัลกอริทึมการแทนที่แอลอาร์ยูช่วยลดการติดต่อระหว่างไคลเอนต์และเซิร์ฟเวอร์ได้ ทำให้ลดปริมาณการใช้งานเครือข่ายลงด้วย นอกจากนี้ระบบแฟ้มข้อมูลอาร์เอฟเอสเหมาะกับผู้ใช้ตั้งแต่

ระดับมือใหม่ขึ้นไป เพราะติดตั้งไม่ยาก และใช้งานได้ง่ายผ่านทางส่วนต่อประสานของระบบแฟ้มข้อมูล (file system APIs) หรือผ่านทางโปรแกรมจัดการแฟ้มข้อมูล (file manager) ก็ได้

ประสิทธิภาพของระบบแฟ้มข้อมูลอาร์เอฟเอสอยู่ระดับที่ยอมรับได้ต่อการใช้งานโดยทั่วไป อาร์เอฟเอสไคลเอนต์ตัวต้นแบบรองรับขนาดแฟ้มข้อมูลที่หลากหลาย แต่จะไม่เหมาะกับแฟ้มข้อมูลที่มีขนาดใหญ่เกินไป จากการทดสอบพบว่าขนาดแฟ้มข้อมูลที่เหมาะสมควรอยู่ในช่วง 8 เมกะไบต์และ 16 เมกะไบต์ ที่บล็อกข้อมูลขนาด 1 เมกะไบต์ สำหรับการใช้งานโดยทั่วไป

รายการอ้างอิง

[1] Amazon. Amazon Simple Storage Service [Online]. 2011. Available from: http://aws.amazon.com/s3 [2012, April 3]

[2] Dropbox. Dropbox – Simplify your life [Online]. 2009. Available from: https://www.dropbox.com [2012, April 3]

[3] Satyanarayanan, M., et al. Coda: A highly available file system for a distributed workstation environment. IEEE Transactions on Computers 39 (1990): 447-459.

[4] Danga Interactive. MogileFS [Online]. 2010. Available from: https://github.com/mogilefs [2012, April 3]

[5] Wright, C. P., et al. Versatility and unix semantics in namespace unification. ACM Transactions on Storage 2 (2006): 74-105.

[6] Vrable, M., Savage, S., and Voelker, G. M. Cumulus: Filesystem backup to the cloud. ACM Transactions on Storage 5 (2009): 14:1-14:28.

[7] Lu, Y., Mao, H., and Shen, J. A Distributed filesystem framework for transparent accessing heterogeneous storage services. Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, pp. 1-8. Washington, DC, USA: IEEE Computer Society, 2009.

[8] Dhumbumroong, S., and Piromsopa, K. Personal cloud filesystem: A distributed unification filesystem for personal computer and portable device. Eighth International Joint Conference on Computer Science and Software Engineering, pp. 58-62. Nakhon Pathom, Thailand: IEEE Computer Society, 2011.

[9] Szeredi, M. Fuse: Filesystem in userspace [Online]. 2004. Available from: http://fuse.sourceforge.net [2012, April 3]

[10] Canonical Ltd. Ubuntu [Online]. 2011. Available from: http://www.ubuntu.com [2012, April 3]

[11] Norcott, W. IOzone Filesystem Benchmark [Online]. 2006. Available from: http://www.iozone.org [2012, April 3]

[12] Python. Timeit module [Online]. 2003. Available from: http://docs.python.org/library/timeit.html [2012, April 3]

[13] Henk, C., Delafond, S., James, S., and Epler, J. Fuse Python [Online]. 2007. Available from: http://sourceforge.net/projects/fuse/files/fuse-python [2012, April 3]

[14] Jacobsen, K., and Oberhumer, M. F.X.J. PycURL [Online]. 2008. Available from: http://pycurl.sourceforge.net [2012, April 3]

[15] Stenberg, D. cURL - libcurl [Online]. 1999. Available from: http://curl.haxx.se/libcurl [2012, April 3]

ภาคผนวก

โครงสร้างของตารางฐานข้อมูลที่ใช้ในระบบแฟ้มข้อมูลอาร์เอฟเอส

ตารางฐานข้อมูลมีทั้งหมด 3 ตาราง ได้แก่ ตาราง file ตาราง server และตาราง block โดยแต่ละตารางมีรายละเอียด ดังต่อไปนี้

1. ตาราง file

ตาราง file บันทึกข้อมูลเมตาเดตาของแฟ้มข้อมูล และไดเรคทอรี โดยมีโครงสร้างภายใน ดังตารางที่ ก-1

ตารางที่ ก-1 ตาราง file เก็บข้อมูลเมตาเดตา

ฟิลด์	ชนิดข้อมูล	ข้อกำหนดอื่น ๆ	เก็บข้อมูล
fid	integer	Primary key	เลขประจำตัวของแฟ้มข้อมูล
path	text	Not null	ฟูลพาธ (full path) ที่ใช้อ้างอิงภายในระบบ
size	integer	Not null	ขนาดของแฟ้มข้อมูล
block	text	Not null	- เลขประจำตัวบล็อกถ้าเป็นระเบียนของแฟ้มข้อมูล - เครื่องหมายยัติภังค์ (-) ถ้าเป็นระเบียนของไดเรคทอรี
uid	integer	Not null	เลขประจำตัวของผู้ใช้
gid	integer	Not null	เลขประจำตัวของกลุ่ม
parent	integer	Not null	เลขประจำตัวของไดเรคทอรีที่เป็นบัพแม่ (parent node)
atime	integer	Not null	เวลาเข้าถึง (access time)
mtime	integer	Not null	เวลาแก้ไข (modify time)
ctime	integer	Not null	เวลาเปลี่ยน (change time)
mode	integer	Not null	สิทธิ์ในการจัดการแฟ้มข้อมูล (file permission)
link	integer	Not null	จำนวนลิงค์ ถ้าเป็นแฟ้มข้อมูลจะมีค่าเท่ากับ 1 ถ้าเป็นไดเรคทอรีจะมีค่ามากกว่า 2 ขึ้นไป
timestamp	integer	Not null	เวลาที่สร้างระเบียน

2. ตาราง server

ตาราง server เก็บข้อมูลของเว็บเซิร์ฟเวอร์ที่ใช้เป็นสตอร์เรจโหนดภายในระบบ โดยมีโครงสร้างภายในดังตารางที่ ก-2

ตารางที่ ก-2 ตาราง server เก็บข้อมูลของสตอร์เรจโหนด

ฟิลด์	ชนิดข้อมูล	ข้อกำหนดอื่น ๆ	เก็บข้อมูล
sid	integer	Primary key	เลขประจำตัวของสตอร์เรจโหนด
sname	text	Not null	ชื่อหรือยูอาร์แอลของสตอร์เรจโหนด
status	integer	Not null	สถานะของเซิร์ฟเวอร์ 0 - ไม่เปิดการใช้งาน 1 - เปิดการใช้งาน
timestamp	integer	Not null	เวลาที่สร้างระเบียน

3. ตาราง block

ตาราง block เก็บข้อมูลของบล็อกข้อมูลของแฟ้มข้อมูลภายในระบบ โดยมีโครงสร้างภายในดังตารางที่ ก-3

ตารางที่ ก-3 ตาราง block เก็บข้อมูลของบล็อกข้อมูล

ฟิลด์	ชนิดข้อมูล	ข้อกำหนดอื่น ๆ	เก็บข้อมูล
bid	integer	Primary key	เลขประจำตัวของบล็อกข้อมูล
sid	integer	Not null	เลขประจำตัวของสตอร์เรจโหนดตัวที่เป็นที่อยู่ของบล็อกข้อมูลนั้น
timestamp	integer	Not null	เวลาที่สร้างระเบียน

ข้อมูลที่ใช้ในการวาดกราฟที่แสดงในบทที่ 5

1. ปริมาณงานของการทดสอบทำสำเนาแฟ้มข้อมูล และการดาวน์โหลดแฟ้มข้อมูล

ตารางที่ ก-4 ตารางแสดงปริมาณงานของการทดสอบทำสำเนาแฟ้มข้อมูล (หน่วย: kB/s)

Block size / File size	256 kB	512 kB	1024 kB	2048 kB
2 MB	3951.650	5177.553	5824.053	5631.219
4 MB	4294.265	6022.938	6461.653	5094.599
6 MB	3916.490	6335.684	7135.297	5048.990
8 MB	4636.637	6671.891	8009.761	4708.609
16 MB	4575.712	6618.691	7692.796	4944.995
32 MB	4583.387	5713.326	6854.661	4246.684
64 MB	3215.569	5216.574	5944.462	4146.137
128 MB	3710.533	4947.887	5725.643	3773.029
256 MB	3335.109	4156.145	4679.978	3528.094

ตารางที่ ก-5 ตารางแสดงปริมาณงานของการทดสอบดาวน์โหลดแฟ้มข้อมูล (หน่วย: kB/s)

Block size / File size	256 kB	512 kB	1024 kB	2048 kB
2 MB	7830.816	7397.811	8476.286	6561.944
4 MB	9397.697	9663.470	9668.687	8430.276
6 MB	9436.467	9647.862	9824.327	9788.818
8 MB	9852.454	9454.004	9885.520	9812.838
16 MB	9906.971	10541.517	10555.894	10611.421
32 MB	10475.399	10654.158	10865.895	10972.775
64 MB	10931.286	10934.395	10982.700	10990.626
128 MB	10681.674	10828.438	11110.581	11082.778
256 MB	10556.875	11077.924	11151.963	11157.194

2. ปริมาณงานของการทดสอบการดำเนินการทางแฟ้มข้อมูลที่วัดได้ด้วยไอโอโซน

ตารางที่ ก-6 ตารางปริมาณงานของการทดสอบที่แฟ้มข้อมูลขนาด 8 เมกะไบต์ (หน่วย: kB/s)

	ext4	dummyfs	RFS	RFS (No cache)	SSHFS
Write	630858.5	39913	36229.5	38713	14990.5
Rewrite	939179	46342.5	43801	39325	14951
Random write	1019281.5	40144	40921	37943	14942
Read	1868595	492884.5	420551	9707.085	11388.5
Reread	1898057	686729	559332.5	557422	1502816
Random read	1934821.5	335052.5	236707	246169	1543806

ตารางที่ ก-7 ตารางปริมาณงานของการทดสอบที่แฟ้มข้อมูลขนาด 16 เมกะไบต์(หน่วย: kB/s)

	ext4	dummyfs	RFS	RFS (No cache)	SSHFS
Write	623035	34614.5	36706	36133.5	12923
Rewrite	906239	40276	38441	36843	12914
Random write	977799	39452.5	39175	38784	12945.5
Read	1871586	633067	503904	10384.325	11395
Reread	1922343	683634.5	555709	553004	1538900
Random read	1913992	318107.5	255854	258776.5	1512498

ตารางที่ ก-8 ตารางปริมาณงานของการทดสอบที่แฟ้มข้อมูลขนาด 256 เมกะไบต์ (หน่วย: kB/s)

	ext4	dummyfs	RFS	RFS (No cache)	SSHFS
Write	25612	24348	24696	26268	10257
Rewrite	27906.5	21561.5	24701.5	25139.5	10445
Random write	12984	11301.5	11103	8599	10272
Read	1934734.5	539779	533227.5	10931.145	10599.5
Reread	1964683.5	644430	555066	567324.5	1665348
Random read	1923524	327559	276708	270881	1584982.5

ประวัติผู้เขียนวิทยานิพนธ์

นายวิชชา รตินิมิตธรรม เกิดเมื่อวันที่ 8 มิถุนายน พ.ศ. 2531 ที่จังหวัดกรุงเทพมหานคร สำเร็จการศึกษาหลักสูตรวิศวกรรมศาสตรบัณฑิต สาขาวิชาวิศวกรรมยานยนต์ จากภาควิชาวิศวกรรมเครื่องกล คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2552 และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2553