ระบบไฟล์สำหรับคลัสเตอร์และกริดที่สนับสนุนการเข้าถึงไฟล์แบบท้องถิ่น

นายนิภัทร์ ลีละปัญญา

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2550
ลิขสิทธิ์ของจุฬาลงกรณ์มหาวิทยาลัย

GRID ENABLE CLUSTER FILE SYSTEM WITH LOCAL ACCESS SUPPORTS

Mr. Nipat Leelapanya

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

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 2007

Copyright of Chulalongkorn University

หัวข้อวิทยานิพนธ์	ระบบไฟล์สำหรับคลัสเตอร์และกริดที่สนับสนุนการเข้าถึงแบบท้องถิ่น
โดย	นาย นิภัทร์ ลีละปัญญา
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร. วีระ เหมืองสิน

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(รองศาสตราจารย์ ดร. ประภาส จงสถิตย์วัฒนา)

..... อาจารย์ที่ปรึกษา
(อาจารย์ ดร. วีระ เหมืองสิน)

..... กรรมการ
(อาจารย์ ดร. ณัฐวุฒิ หนูไพโรจน์)

..... กรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. ภุชงค์ อุทโยภาศ)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

นิภัทร์ ลีละปัญญา : ระบบไฟล์สำหรับคลัสเตอร์และกริดที่สนับสนุนการเข้าถึงไฟล์แบบท้องถิ่น. (GRID ENABLE CLUSTER FILE SYSTEM WITH LOCAL ACCESS SUPPORTS) อ. ที่ปรึกษา :อ.ดร. วีระ เหมืองสิน, 96 หน้า.

ในระบบกริดส่วนใหญ่ การประมวลผลจะเกิดขึ้นที่เครื่องคลัสเตอร์คอมพิวเตอร์ ส่วนไฟล์ที่ใช้ในการทำงานจะเก็บอยู่ที่เครื่องให้บริการไฟล์ การทำงานจะมีขั้นตอนเพิ่มเติมคือการเคลื่อนย้ายไฟล์อินพุตและเอาท์พุตระหว่างเครื่องให้บริการไฟล์และเครื่องคอมพิวเตอร์ที่ประมวลผล ซึ่งในสภาพแวดล้อมการทำงานของกริดและคลัสเตอร์คอมพิวเตอร์นั้น เครื่องให้บริการไฟล์และเครื่องที่ประมวลผลมักจะอยู่ต่างสถานที่กัน โดยในคลัสเตอร์คอมพิวเตอร์การเคลื่อนย้ายไฟล์จะทำผ่านข่ายงานบริเวณเฉพาะที่ (LAN) ส่วนในระบบกริดนั้นอาจจะต้องเคลื่อนย้ายไฟล์ผ่านระบบอินเตอร์เน็ต หากงานที่ทำเป็นประเภทใช้งานข้อมูลปริมาณมาก ขั้นตอนการเคลื่อนย้ายไฟล์นี้จะส่งผลให้เวลาที่ใช้ในการทำงานเพิ่มมากขึ้นตามแบนด์วิดท์ระหว่างเครื่องต้นทางและปลายทาง โดยเฉพาะอย่างยิ่งในกรณีของงานประเภทกระแสงาน จำนวนครั้งของการเคลื่อนย้ายไฟล์ก็จะมากขึ้นตามจำนวนขั้นตอนของงาน

วิทยานิพนธ์ฉบับนี้นำเสนอการออกแบบและการพัฒนาระบบที่ใช้จัดการไฟล์ภายในคลัสเตอร์คอมพิวเตอร์และองค์กรเสมือนเพื่อสนับสนุนการทำงานแบบส่งงานไปหาข้อมูล ไฟล์ที่ใช้ในงานจะถูกสร้างและเก็บบนคลัสเตอร์คอมพิวเตอร์แทนการส่งผ่านเครือข่ายไปเก็บที่เครื่องให้บริการไฟล์ ผู้ใช้และโปรแกรมจัดลำดับงานสามารถสอบถามระบบถึงตำแหน่งไฟล์เพื่อส่งงานถัดไปไปยังเครื่องที่มีไฟล์เก็บอยู่ได้ จากผลการทดลองสามารถสรุปได้ว่า สำหรับงานประเภทใช้ข้อมูลปริมาณมากที่มีลักษณะเป็นกระแสงาน การทำงานที่อ่านเขียนไฟล์แบบท้องถิ่นใช้เวลาน้อยกว่าการอ่านเขียนไฟล์ที่มีการเคลื่อนย้ายไฟล์ผ่านเครือข่ายที่นอกจากต้องจะเสียเวลาในการเคลื่อนย้ายไฟล์แล้ว ยังอาจเกิดปัญหาคอขวดที่เครื่องให้บริการไฟล์ ในกรณีที่มีหลายๆ โปรเซสอ่านเขียนไฟล์พร้อมๆ กันอีกด้วย

ภาควิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่อนิสิต.....นิภัทร์ ลีละปัญญา.....

สาขาวิชา.....วิศวกรรมคอมพิวเตอร์..... ลายมือชื่ออาจารย์ที่ปรึกษา.....

ปีการศึกษา.....2550....

4970393021 : MAJOR COMPUTER ENGINEERING

KEY WORD: GRID / CLUSTER COMPUTER / VO / FILE SYSTEM / LOCAL FILE ACCESS

NIPAT LEELAPANYA : GRID ENABLE CLUSTER FILE SYSTEM WITH LOCAL ACCESS SUPPORTS. THESIS ADVISOR : VEERA MUANGSIN, 96 pp.

On most grid systems, computation takes place on cluster computers, while data files are stored on file servers. Executing jobs requires an additional stage to transfer input and output files between file servers and computing nodes. In Grid and cluster computer environment, the file servers and the computing nodes are usually in different locations. Files in a cluster computer are transferred within LAN while files in Grid are usually transferred across the Internet. If the job is a kind of data intensive applications, the file transfer process results in much extra time depending on the bandwidth between the relating computers. Especially, in case of workflow application, the amount of workflow stages will affect the amount of transfer processes.

This thesis presents design and development of a file management system for handling files in cluster computers and Virtual Organizations (VOs). The system aims to support the moving-computation-to-data paradigm. Files relating to jobs are created and stored on computing nodes of cluster computers that execute the jobs. Users and job scheduler programs can query the system about the file location and submit the job to the computer that maintains the file. The experiments are set to evaluate the effect of file transfer processes in a cluster computer and in Grid environment. The results indicate that, for data intensive applications, computation with local file access spends less time than computation with file transfer processes. In additional, the later case may suffer from the bottleneck problem if there are many processes read and write files concurrently.

Department....Computer..Engineering... Student's Signature.....

Field of study..Computer..Engineering.. Advisor's Signature.....

Academic year....2007.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดีด้วยความช่วยเหลือของบุคคลหลายท่าน ท่านแรกคือ อาจารย์ ดร. วีระ เหมืองสิน อาจารย์ที่ปรึกษาที่ให้คำแนะนำที่ดีทั้งเรื่องแนวคิดในการทำวิจัย การเขียนเอกสารทางวิชา และคอยตรวจแก้งานมากมายของผู้วิจัย

ขอขอบคุณ ผศ.ดร. อาณัติ เรืองรัศมี และคุณ นพพร แซ่เหล่ม ภาควิชาวิศวกรรมโยธา คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย คุณนพรัตน์ นพกวด คณะวิศวกรรมศาสตร์ มหาวิทยาลัยเกษตรศาสตร์ ที่ให้ความช่วยเหลือเรื่องการทดลองที่ใช้ในการทำวิจัย

ขอขอบคุณเพื่อนๆ และพี่ๆ ในภาควิชาวิศวกรรมคอมพิวเตอร์ที่ให้คำแนะนำและความช่วยเหลือเมื่อมีปัญหา รวมทั้งเป็นเพื่อนเล่น เพื่อนคุย ตลอดระยะเวลาการทำวิจัย

ขอขอบคุณพี่สาว สำหรับขนมและอาหารอร่อยๆ ที่ซื้อมาให้ รวมถึงคอยขับรถรับส่งกลับบ้าน

สุดท้ายขอกราบขอบพระคุณบิดา มารดา ในทุกๆ เรื่อง ตั้งแต่การเลี้ยงดูด้วยความเมตตา คำแนะนำเมื่อมีปัญหา คำเตือนสำหรับเรื่องที่ไม่ถูกต้องเหมาะสม กำลังใจเมื่อเกิดความท้อแท้ และการสนับสนุนที่ดีตลอดมา

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญ

หน้า

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

หน้า

หน้า

สารบัญตาราง

หน้า

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

สารบัญรูปภาพ

ภาพประกอบ หน้า

ภาพประกอบ หน้า

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

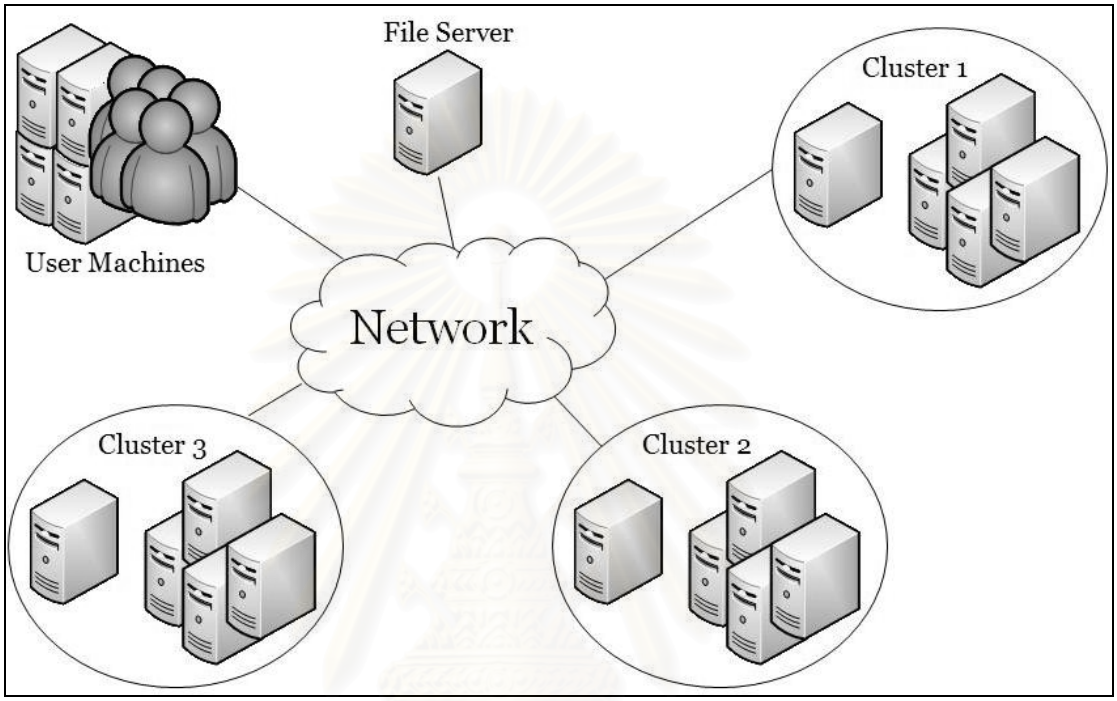
ในปัจจุบันงานวิจัยโดยเฉพาะงานทางด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ และการแพทย์มีแนวโน้มว่าจะมีขนาดและความซับซ้อนมากขึ้น ลักษณะของงานที่อยู่ในขอบเขตของงานวิจัยประเภทนี้ได้แก่ การสร้างแบบจำลอง (Modeling), การจำลองเหตุการณ์ (Simulation), การวิเคราะห์ (Analysis), และการทำจินตทัศน์ (Visualization) เป็นต้น งานวิจัยประเภทนี้ส่วนใหญ่จะมีการทำงานเป็นแบบกระแสงาน (Workflow) ซึ่งก็คือ เป็นงานที่ประกอบด้วยงานย่อยๆ หลายงาน แต่ละงานจะต้องทำเรียงลำดับก่อนหลัง และเมื่อโปรแกรมประยุกต์ในงานแรกสร้างผลลัพธ์ออกมา ผลลัพธ์นั้นก็จะถูกนำไปใช้ต่อโดยโปรแกรมประยุกต์ในงานหลัง โปรแกรมประยุกต์ที่ใช้ในแต่ละงานของกระแสงานโดยทั่วไปจะเป็นคนละโปรแกรมกัน โดยเราสามารถแบ่งประเภทของโปรแกรมเหล่านี้ตามลักษณะการใช้งานทรัพยากรได้เป็น 2 ประเภทคือ

1) *โปรแกรมประยุกต์ที่ใช้พลังในการประมวลผลสูง* (Computing-Intensive Application) งานบางประเภทอาจจะต้องใช้เวลาในการประมวลผลนานเป็นสัปดาห์ หรือบางงานอาจจะต้องใช้เวลาเป็นเดือนก็มี
2) *โปรแกรมประยุกต์ที่ใช้ข้อมูลปริมาณมาก* (Data-Intensive Application) ข้อมูลที่ใช้ในงานประเภทนี้จะมีขนาดใหญ่มาก ตั้งแต่หลายร้อยเมกะไบต์ ไปจนถึงหลายกิกะไบต์ หรือบางงานเช่นงานวิจัยด้านฟิสิกส์ อาจจะต้องใช้ข้อมูลถึงหลักเทระไบต์

เนื่องจากลักษณะที่กล่าวมานี้ ทำให้การทำงานวิจัยขนาดใหญ่งานหนึ่งมีความจำเป็นต้องใช้ทรัพยากร (Resource) ปริมาณมาก ดังนั้นในโครงการวิจัยเหล่านี้จึงมีการร่วมมือกันของกลุ่มนักวิจัยจากหลายๆ องค์กร ซึ่งในองค์กรเหล่านี้จะมีการใช้งานทรัพยากรร่วมกัน ระบบ “*กริด*” [1, 2] จึงถูกนำมาใช้เพื่อช่วยให้การแบ่งปันทรัพยากรสามารถทำได้อย่างมีประสิทธิภาพ

ระบบจัดการทรัพยากรภายในกริดประเภทหนึ่งคือ “*องค์กรเสมือน*” (VO: Virtual Organization) [3] องค์กรเสมือนจัดตั้งขึ้นเพื่อให้นักวิจัยที่มีความสนใจในเรื่องเดียวกันแต่อยู่ต่างสถานที่กันสามารถรวมกลุ่มเพื่อแบ่งปันทรัพยากรที่ใช้ในการวิจัยได้ ในองค์กรเสมือนหนึ่ง เจ้าของทรัพยากรสามารถกำหนดนโยบายและกฎเกี่ยวกับการเข้าใช้งานทรัพยากรขององค์กรตนเองได้ ซึ่งโครงสร้างของระบบกริดจะช่วยให้การใช้งานทรัพยากรสามารถทำได้อย่างมีประสิทธิภาพและมีความปลอดภัย และเมื่องานวิจัยสิ้นสุดลง แต่ละองค์กรก็สามารถยกเลิกการแบ่งปันทรัพยากรได้

รูปที่ 1.1 แสดงตัวอย่างขององค์กรเสมือนหนึ่งองค์กรซึ่งประกอบด้วยเครื่องคลัสเตอร์คอมพิวเตอร์สามกลุ่ม เครื่องให้บริการไฟล์หนึ่งเครื่องและผู้ใช้ในระบบ ซึ่งตามหลักการขององค์กรเสมือนแล้ว ทั้งเครื่องคอมพิวเตอร์และผู้ใช้ภายในหนึ่งองค์กรเสมือนสามารถอยู่ภายในองค์กรเดียวกันหรือต่างองค์กรก็ได้



รูปที่ 1.1: แสดงตัวอย่างส่วนประกอบขององค์กรเสมือน 1 องค์กร

"*คลัสเตอร์คอมพิวเตอร์*" (Cluster Computer) เป็น "*เครื่องคอมพิวเตอร์สมรรถนะสูง*" (HPC: High Performance Computer) ประเภทหนึ่งที่นิยมใช้ในองค์กรเสมือนเพื่อการทำงานที่ใช้ทรัพยากรปริมาณมาก ภายในหนึ่งคลัสเตอร์จะประกอบด้วยเครื่องคอมพิวเตอร์จำนวนมากเชื่อมต่อกันเป็นหนึ่งหน่วย โดยจะมีเครื่องหนึ่งทำหน้าที่เป็น "*เครื่องควบคุม*" (Master Node) ส่วนที่เหลือเป็น "*เครื่องคำนวณ*" (Computing Node) คอมพิวเตอร์ทั้งหมดจะเชื่อมต่อกันด้วยเครือข่ายส่วนตัว (Private Network) โดยมีเครื่องควบคุมทำหน้าที่เป็นเกตเวย์สำหรับติดต่อกับเครือข่ายภายนอก เนื่องจากหนึ่งคลัสเตอร์ประกอบด้วยเครื่องคอมพิวเตอร์จำนวนมาก ดังนั้นทรัพยากรหลักที่คลัสเตอร์คอมพิวเตอร์มีให้ใช้ก็คือส่วนประมวลผล (ซีพียู) และที่เก็บข้อมูลปริมาณมาก

สิ่งหนึ่งที่มีความคล้ายคลึงกันระหว่างองค์กรเสมือนและคลัสเตอร์คอมพิวเตอร์คือ การใช้งานจะมุ่งเน้นไปที่ทรัพยากรหลักสองชนิด คือส่วนประมวลผลและส่วนเก็บข้อมูล ดังนั้นจึงจำเป็นต้องมีระบบที่ใช้ควบคุมและจัดการการใช้งานทรัพยากรทั้งสองชนิดนี้ทั้งในระดับองค์กรเสมือนและในระดับคลัสเตอร์คอมพิวเตอร์

1) *ส่วนจัดการไฟล์* ทำหน้าที่จัดการไฟล์และควบคุมการใช้งานที่เก็บข้อมูล โดยปกติส่วนจัดการไฟล์มักจะมีคำสั่งเหมือนกับระบบไฟล์ (File System) ทั่วไป เช่น คำสั่งแสดงรายการไฟล์, คำสั่งสำเนาไฟล์, คำสั่งเคลื่อนย้ายไฟล์, และคำสั่งทำซ้ำไฟล์ เป็นต้น

   - *ระดับองค์กรเสมือน* ส่วนจัดการไฟล์จะมีลักษณะเป็นกริดข้อมูล (Data Grid) [4, 5] เช่น SRB [6] และ Gfarm [7] ไฟล์จะถูกเก็บอยู่บนทรัพยากรขององค์กรเสมือนที่ติดตั้งโปรแกรมกริดข้อมูล การจัดการไฟล์สามารถทำจากเครื่องใดก็ได้ที่สามารถติดต่อไปถึงระบบกริดข้อมูลได้
   - *ระดับคลัสเตอร์คอมพิวเตอร์* มีการใช้งานระบบไฟล์คลัสเตอร์ (CFS: Cluster File System) ในการจัดการไฟล์ ซึ่งในมุมมองของระบบไฟล์คลัสเตอร์จะมองทั้ง คลัสเตอร์คอมพิวเตอร์เป็นหนึ่งระบบ ไฟล์จะถูกเก็บไว้บนเครื่องที่เป็นทรัพยากรของคลัสเตอร์คอมพิวเตอร์ ซึ่งการจัดการไฟล์สามารถทำจากเครื่องใดในคลัสเตอร์ก็ได้

2) *ส่วนจัดการงาน* ทำหน้าที่ควบคุมการใช้งานทรัพยากรที่เป็นส่วนประมวลผล

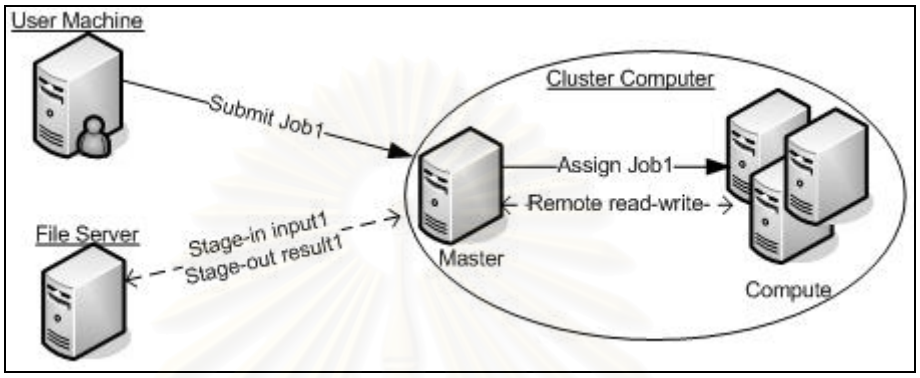
   - *ระดับองค์กรเสมือน* ในบางระบบ ส่วนจัดการงานจะมีข้อมูลสถานะของเครื่องคอมพิวเตอร์ที่เป็นทรัพยากร เมื่อผู้ใช้สั่งงานเข้ามา ส่วนจัดการงานก็สามารถส่งงานไปประมวลผลยังเครื่องที่เหมาะสมได้ ส่วนในบางระบบผู้ใช้จะต้องระบุเองว่าต้องการให้งานถูกส่งไปประมวลผลที่เครื่องใด
   - *ระดับเครื่องคลัสเตอร์คอมพิวเตอร์* จะใช้โปรแกรมจัดลำดับงาน (Job Scheduler) ทำหน้าที่ส่งต่องานจากเครื่องควบคุมไปประมวลผลที่เครื่องคำนวณ ตัวอย่างโปรแกรมประเภทโปรแกรมจัดลำดับงานที่มีใช้ในปัจจุบันได้แก่ SGE, PBS, Condor, ฯลฯ

นโยบายการจัดการงานภายในองค์กรเสมือนและภายในคลัสเตอร์คอมพิวเตอร์จะมีอยู่ 2 วิธี ได้แก่

   - *แบบส่งข้อมูลไปหางาน* (Moving Data to Computation) วิธีนี้ไฟล์ที่ใช้ในการประมวลผลจะเก็บอยู่ที่ "*เครื่องให้บริการไฟล์*" (File Server) และส่วนจัดการงานจะส่งงานไปยังเครื่องคอมพิวเตอร์เครื่องหนึ่ง จากนั้นจึงส่งไฟล์ที่เกี่ยวข้องตามไป วิธีนี้มีข้อดีคืองานจะถูกส่งไปประมวลผลทันทีเมื่อมีเครื่องว่างงาน แต่มีข้อเสียคือหากไฟล์มีขนาดใหญ่จะทำให้เสียเวลาเคลื่อนย้ายไฟล์จากเครื่องให้บริการไฟล์ไปยังเครื่องที่ทำการคำนวณ
   - *แบบส่งงานไปหาข้อมูล* (Moving Computation to Data) วิธีนี้เครื่องที่ทำการคำนวณจะทำหน้าที่เก็บไฟล์ด้วย ซึ่งส่วนจัดการงานจะส่งงานไปยังเครื่องที่มีไฟล์ที่เกี่ยวข้องเก็บอยู่ วิธีนี้มีข้อดีคือการประมวลผลสามารถ "*เข้าถึงไฟล์แบบท้องถิ่น*" (Local Access) ได้ ทำให้ไม่เสียเวลาเคลื่อนย้ายไฟล์ แต่หากเครื่องที่เก็บไฟล์อยู่ไม่ว่าง ก็จะเสียเวลารอให้เครื่องว่างหรืออาจต้องใช้วิธีการทำซ้ำไฟล์ไปเครื่องอื่นที่ว่าง

1.2 นิยามของปัญหา

โดยปกติ นโยบายที่ใช้ในการส่งงานของส่วนจัดการงานในองค์กรเสมือนจะเป็นแบบ “ส่งข้อมูลไปหางาน” ซึ่งจะเป็นไปตามที่แสดงในรูปที่ 1.2 นั่นคือ ไฟล์ของผู้ใช้จะเก็บอยู่ที่เครื่องให้บริการไฟล์ ส่วนการประมวลผลจะเกิดขึ้นที่เครื่องคำนวณภายในคลัสเตอร์คอมพิวเตอร์ ขั้นตอนการทำงานจะเป็นดังนี้

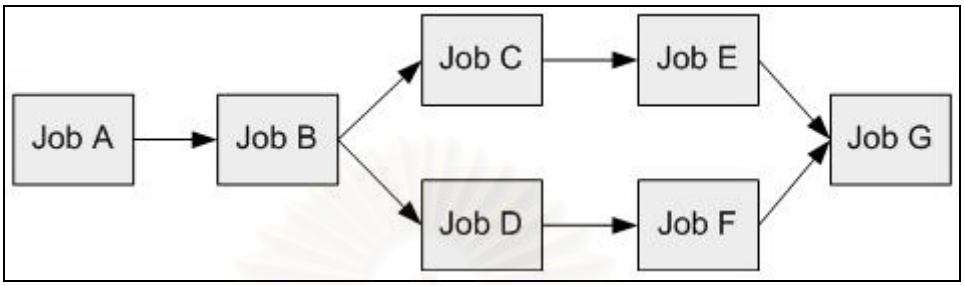


รูปที่ 1.2: แสดงการรับส่งไฟล์ภายในองค์กรเสมือนและคลัสเตอร์คอมพิวเตอร์

1) ผู้ใช้ล็อกอินเข้าไปใช้งานเครื่องที่เป็นทรัพยากรขององค์กรเสมือน
2) ผู้ใช้ส่งงานเข้าไปที่เครื่องคลัสเตอร์
3) ไฟล์ของผู้ใช้ที่เก็บอยู่บนเครื่องให้บริการไฟล์ถูกส่งไปยังเครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์ (Stage-in) ซึ่งเครื่องให้บริการไฟล์นี้อาจเป็นเครื่องเดียวกับเครื่องที่ผู้ใช้ใช้ทำงานก็ได้
4) การทำงานภายในคลัสเตอร์คอมพิวเตอร์จะเริ่มต้นจากโปรแกรมจัดลำดับงานส่งงานไปยังเครื่องคำนวณเครื่องหนึ่งเพื่อประมวลผล
5) เครื่องคำนวณจะอ่านเขียนไฟล์แบบทางไกล (Remote Access) โดยใช้ Network File System (NFS) [8]
6) เมื่อประมวลผลเสร็จ ไฟล์ผลลัพธ์ของงานจะถูกส่งไปเก็บที่เครื่องให้บริการไฟล์ หรือเครื่องที่ผู้ใช้ใช้ทำงาน (Stage-out)

จากขั้นตอนการทำงานดังกล่าว จะเห็นว่าเกิดการเคลื่อนย้ายไฟล์สองครั้ง คือระหว่างเครื่องให้บริการไฟล์และเครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์ในองค์กรเสมือน และการอ่านเขียนไฟล์แบบทางไกลอีกสองครั้งระหว่างเครื่องควบคุมและเครื่องคำนวณในคลัสเตอร์คอมพิวเตอร์ เวลาที่เสียไปนี้จะขึ้นอยู่กับขนาดของไฟล์และแบนด์วิดท์ของเครือข่าย ในกรณีของงานด้านวิทยาศาสตร์ วิศวกรรมศาสตร์ และการแพทย์นั้นมักจะใช้ไฟล์ขนาดใหญ่ในการทำงาน ทำให้ต้องเสียเวลาในการเคลื่อนย้ายไฟล์มาก และเนื่องจากงานด้านนี้ส่วนใหญ่มีลักษณะเป็นกระแสงาน นั้นคือประกอบด้วยงานย่อยหลายๆ งาน ทำให้มีขั้นตอนการเคลื่อนย้ายไฟล์และการอ่านเขียนไฟล์แบบทางไกลมากขึ้น เช่นในรูปที่ 1.3 แสดงตัวอย่างกระแสงานหนึ่งซึ่งประกอบด้วย 7 งานย่อย หากการทำงานเป็นไปตามที่แสดงในรูปที่ 1.2 คือมีการเคลื่อนย้ายไฟล์ระหว่างเครื่องเก็บไฟล์และเครื่องคำนวณในทุกๆ งานย่อยก็จะเกิดการเคลื่อนย้าย

ไฟล์และการอ่านเขียนไฟล์แบบทางไกลรวมกัน 28 ครั้ง (ในกรณีนี้งานแต่ละงานสามารถส่งไปที่คลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือนเครื่องใดก็ได้ และภายในเครื่องคลัสเตอร์คอมพิวเตอร์นั้นงานก็สามารถถูกส่งไปประมวลผลที่เครื่องคำนวณเครื่องใดก็ได้เช่นกัน)

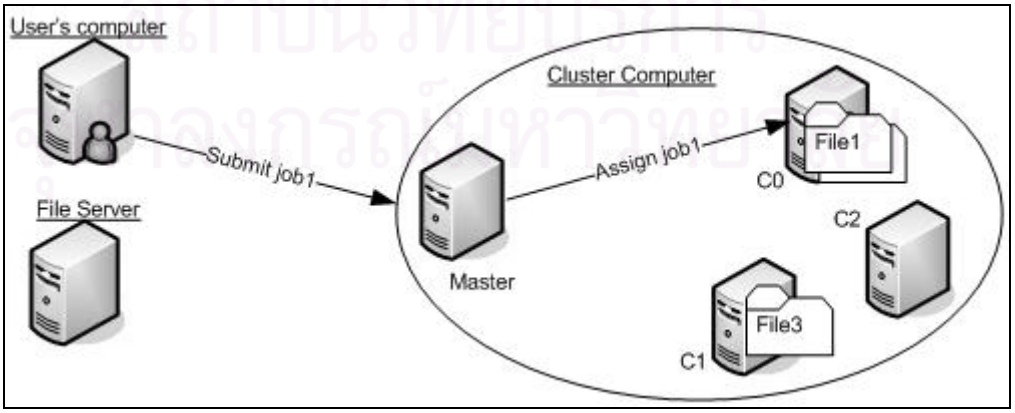


รูปที่ 1.3: แสดงตัวอย่างความเกี่ยวข้องกันของงานในงานประเภทกระแสงาน

นอกจากนี้การทำงานในคลัสเตอร์คอมพิวเตอร์โดยให้โปรเซสหลายๆ โปรเซสบนเครื่องคำนวณหลายๆ เครื่องเข้าถึงไฟล์ที่อยู่บนเครื่องควบคุมแบบทางไกลพร้อมๆ กัน อาจทำให้เกิดปัญหาคอขวดที่เครื่องควบคุมได้ กล่าวคือแต่ละโปรเซสบนเครื่องคำนวณจะต้องแบ่งปันแบนด์วิดท์ที่ได้รับจากเครื่องควบคุม ทำให้โปรเซสเหล่านั้นได้รับแบนด์วิดท์ไม่เต็มที่ ซึ่งหากแบนด์วิดท์ที่ได้รับน้อยเกินไปก็จะส่งผลให้การทำงานในส่วนการอ่านเขียนไฟล์ใช้เวลานานขึ้นมาก

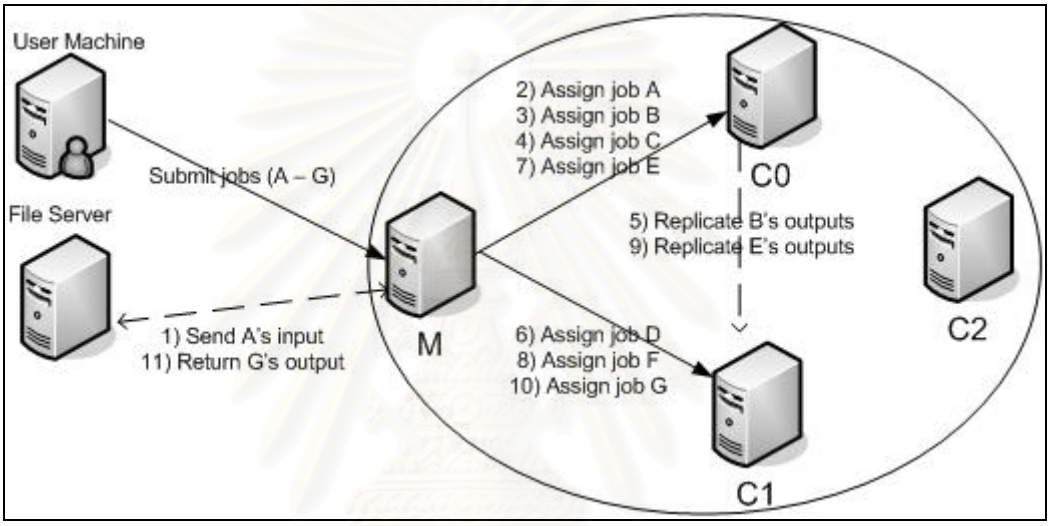
1.3 แนวทางการแก้ปัญหา

เพื่อแก้ไขปัญหาดังกล่าว โครงงานนี้ได้ออกแบบระบบเพื่อสนับสนุนนโยบายการส่งงานแบบ “ส่งงานไปหาข้อมูล” วิธีนี้จะช่วยให้สามารถลดปริมาณการเคลื่อนย้ายไฟล์ระหว่างเครื่องและการอ่านเขียนไฟล์แบบทางไกลลงได้ ไฟล์ที่ใช้ในการคำนวณจะถูกเก็บไว้บนเครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์แทนที่จะเก็บอยู่บนเครื่องให้บริการไฟล์ เพื่อให้เครื่องคำนวณสามารถประมวลผลโดยเข้าถึงไฟล์แบบท้องถิ่นได้



รูปที่ 1.4: แสดงการรับส่งไฟล์ภายในองค์กรเสมือนและคลัสเตอร์คอมพิวเตอร์

ดังที่แสดงในรูปที่ 1.4 สมมุติว่างานชื่อ “Job1” ต้องการใช้งานไฟล์ชื่อ “File1” ในการทำงาน เมื่อผู้ใช้ส่งงานมาที่คลัสเตอร์คอมพิวเตอร์ โปรแกรมจัดลำดับงานก็จะส่งงานไปประมวลผลยังเครื่องคำนวณที่มีไฟล์เก็บอยู่ดังแสดงในรูปนี้คือเครื่อง “C0” และหากงานอีกงานหนึ่งต้องการใช้ไฟล์ชื่อ “File3” ในการทำงาน งานนั้นก็จะถูกส่งมาประมวลผลที่เครื่อง “C1” วิธีนี้จะช่วยให้ลดเวลาที่ใช้ในการรับส่งไฟล์ผ่านเครือข่ายระหว่างเครื่องให้บริการไฟล์กับเครื่องคลัสเตอร์คอมพิวเตอร์และการอ่านเขียนไฟล์แบบทางไกลภายในคลัสเตอร์คอมพิวเตอร์ลงได้ นอกจากนี้การอ่านเขียนไฟล์แบบท้องถิ่นก็ยังลดการเกิดปัญหาคอขวดที่เกิดจากการอ่านเขียนไฟล์แบบทางไกลของหลายๆ โปรเซสพร้อมๆ กันได้



รูปที่ 1.5: แสดงแนวคิดขั้นตอนการทำงานกระแสงานในรูปที่ 1.3

หากใช้วิธีการนี้ในการทำงานกระแสงานในรูปที่ 1.3 จะมีขั้นตอนการทำงานตามรูปที่ 1.5 ดังนี้

1) ส่งไฟล์อินพุตของงาน A มาที่เครื่องควบคุมของเครื่องคลัสเตอร์คอมพิวเตอร์ที่ประมวลผล (M)
2) ส่งงาน A มาประมวลผลที่ C0 และเก็บไฟล์เอาท์พุตไว้บน C0
3) ส่งงาน B มาประมวลผลที่ C0 และเขียนไฟล์เอาท์พุตไว้บน C0
4) ส่งงาน C มาประมวลผลที่ C0 และเขียนไฟล์เอาท์พุตไว้บน C0
5) ทำซ้ำ (Replicate) ไฟล์เอาท์พุตของงาน B ไปไว้ที่เครื่อง C1
6) ส่งงาน D ไปประมวลผลที่เครื่อง C1 และเขียนไฟล์เอาท์พุตไว้บน C1
7) ส่งงาน E ไปประมวลผลที่เครื่อง C0 และเขียนไฟล์เอาท์พุตไว้บน C0
8) ส่งงาน F ไปประมวลผลที่เครื่อง C1 และเขียนไฟล์เอาท์พุตไว้บน C1
9) ทำซ้ำไฟล์เอาท์พุตของงาน E ไปที่เครื่อง C1
10) ส่งงาน G ไปประมวลผลที่เครื่อง C1 และเขียนไฟล์เอาท์พุตกลับไปบนเครื่องควบคุมของคลัสเตอร์
11) ส่งไฟล์เอาท์พุตของงาน G จากเครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์กลับไปให้ผู้ใช้

จากขั้นตอนดังกล่าวจะเกิดการเคลื่อนย้ายไฟล์ระหว่างเครื่องให้บริการไฟล์กับเครื่องคลัสเตอร์คอมพิวเตอร์เพียง 2 ครั้งคือไฟล์อินพุตของ A และไฟล์เอาท์พุตของ G และเกิดการอ่านเขียนไฟล์แบบทางไกลภายในคลัสเตอร์คอมพิวเตอร์ 2 ครั้งคือ C0 อ่านไฟล์อินพุตของ A และ C1 เขียนไฟล์เอาท์พุตของ G และมีการเคลื่อนย้ายไฟล์ภายในคลัสเตอร์คอมพิวเตอร์อีกสองครั้งคือการทำซ้ำไฟล์เอาท์พุตของ B และ E หรือหากไม่ต้องการทำซ้ำไฟล์ก็สามารถให้การทำงานในขั้นตอนที่ 6 และ 10 อ่านไฟล์แบบทางไกลแทนก็ได้

เพื่อให้สามารถทำงานตามขั้นตอนดังกล่าวได้นั้น ในระดับคลัสเตอร์คอมพิวเตอร์สามารถทำได้โดยใช้หลักการของระบบไฟล์คลัสเตอร์ เพื่อกระจายไฟล์ไปเก็บบนเครื่องคำนวณ และเพิ่มความสามารถของตัวจัดลำดับงานให้สามารถส่งงานไปยังเครื่องที่มีไฟล์เก็บอยู่ได้ ส่วนในระดับองค์กรเสมือนสามารถทำได้โดยใช้หลักการเรื่องมิดเดิลแวร์ [9] เพื่อเป็นตัวกลางในการทำงานระหว่างผู้ใช้ในองค์กรเสมือนกับระบบไฟล์บนเครื่องคลัสเตอร์คอมพิวเตอร์

1.4 วัตถุประสงค์

งานวิจัยนี้มีวัตถุประสงค์เพื่อการออกแบบและพัฒนาระบบจัดการไฟล์ โดยใช้วิธีการจัดเก็บไฟล์ไว้บนเครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์แทนการเก็บไว้บนเครื่องให้บริการไฟล์ ระบบจัดการไฟล์นี้จะต้องสามารถใช้งานได้ทั้งในระดับองค์กรเสมือนและในระดับคลัสเตอร์คอมพิวเตอร์ นอกจากนี้ระบบจัดการไฟล์จะต้องสามารถสนับสนุนการทำงานของตัวจัดลำดับงานเพื่อให้ตัวจัดลำดับงานสามารถใช้นโยบาย “ส่งงานไปหาข้อมูล” ได้

1.5 ขั้นตอนและวิธีการดำเนินงาน

1. ศึกษาทฤษฎีและงานวิจัยที่เกี่ยวข้อง
2. ออกแบบและพัฒนาระบบไฟล์สำหรับใช้งานในระดับคลัสเตอร์คอมพิวเตอร์
3. ออกแบบและพัฒนาระบบไฟล์สำหรับใช้งานในระดับองค์กรเสมือน
4. พัฒนาระบบการสั่งงานเพื่อใช้งานระบบไฟล์ในการสนับสนุนนโยบายการทำงานของโปรแกรมจัดลำดับงานแบบส่งงานไปหาข้อมูล
5. ทดสอบการทำงานและประสิทธิภาพของระบบ
6. สรุปผลการวิจัย
7. จัดทำวิทยานิพนธ์

1.6 ขอบเขตของงานวิจัย

1. ปัญหาที่สนใจในงานวิจัยนี้คือเรื่องการจัดการไฟล์ในองค์กรเสมือนและในคลัสเตอร์คอมพิวเตอร์เพื่อสนับสนุนนโยบายการส่งงานไปหาข้อมูล

2. ลักษณะของโปรแกรมประยุกต์ที่สนใจในงานวิจัยนี้คือ งานประเภทกระแสงานที่ไฟล์ระหว่างกลาง (Intermediate File) ที่ต้องใช้ในงานย่อยทั้งหมดอยู่บนเครื่องเดียวกัน
3. ใช้โปรแกรมจำลองการเกิดคลื่นสึนามิในการทดสอบระบบ
4. พัฒนาโปรแกรมต้นแบบเพื่อใช้ในการทดสอบการทำงานทั้งในระดับองค์กรเสมือนและระดับคลัสเตอร์คอมพิวเตอร์
5. ใช้ระบบคอมพิวเตอร์ภายในภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เป็นสภาพแวดล้อมหลักในการทำงานและการทดสอบ
6. ระบบจัดการองค์กรเสมือนที่ใช้ในงานวิจัยนี้คือระบบการให้บริการกริด [30] (GHOSTS: Grid Hosting System)
7. คลัสเตอร์คอมพิวเตอร์ที่ใช้ในงานวิจัยนี้ใช้ระบบปฏิบัติการ NPACI Rocks Cluster4.2 [10]

1.7 ประโยชน์ที่คาดว่าจะได้รับ

สามารถนำระบบไปใช้ในการทำงานประเภทกระแสงานที่ใช้ข้อมูลปริมาณมาก เพื่อลดเวลาที่ใช้ในการรับส่งข้อมูลผ่านเครือข่าย และช่วยเพิ่มความสะดวกในการจัดการไฟล์ในระบบไฟล์คลัสเตอร์สำหรับผู้ใช้ในองค์กรเสมือน

1.8 โครงสร้างของวิทยานิพนธ์

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บทคือ

บทที่ 1 ซึ่งเป็นบทนำซึ่งจะกล่าวถึงรูปแบบและลักษณะการใช้งานระบบกริดและคลัสเตอร์คอมพิวเตอร์ นิยามปัญหา แนวทางการแก้ปัญหา วัตถุประสงค์ ขั้นตอนการดำเนินงาน ขอบเขตของงานวิจัย และประโยชน์ที่คาดว่าจะได้รับ

บทที่ 2 เป็นการสรุปถึงทฤษฎีและงานวิจัยต่างๆ ที่เกี่ยวข้องกับงานวิจัยนี้ ได้แก่ ระบบไฟล์คลัสเตอร์ และระบบกริดข้อมูล

บทที่ 3 นำเสนอการออกแบบโครงสร้างและการพัฒนาระบบต้นแบบสำหรับการจัดการไฟล์ในคลัสเตอร์คอมพิวเตอร์

บทที่ 4 นำเสนอการออกแบบโครงสร้างและการพัฒนาระบบต้นแบบสำหรับการจัดการไฟล์ในระบบจัดการองค์กรเสมือน

บทที่ 5 นำเสนอการทดลองและวิเคราะห์ผลการทดลอง

บทที่ 6 เป็นสรุปผลการวิจัย ข้อจำกัดของระบบ ข้อเสนอแนะและแนวทางการวิจัยต่อ

1.9 ผลงานที่ได้รับการตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของงานวิทยานิพนธ์นี้ได้รับการตีพิมพ์เป็นบทความทางวิชาการจำนวน 2 ฉบับ ดังนี้

1. หัวข้อ “An Evaluation of Cluster File Systems with Workflow Applications: A Case Study on Tsunami Simulation Experiments” โดย นายนิภัทร์ ลีละปัญญา และอ.ดร. วีระ เหมืองสิน ในงานประชุมวิชาการ “The 11th Annual National Symposium on Computational Science and Engineering (ANSCSE11)” ซึ่งจัดขึ้น ณ จังหวัดภูเก็ต ในวันที่ 28-30 มีนาคม 2550
2. หัวข้อ “File management System to Support Local File Access” โดย นายนิภัทร์ ลีละปัญญา และอ.ดร. วีระ เหมืองสิน ในงานประชุมวิชาการ “The 2nd International Conference on Advances in Information Technology (IAIT2007)” ซึ่งจัดขึ้น ณ จังหวัดกรุงเทพมหานคร ในวันที่ 1-2 พฤษจิกายน 2550

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎี

งานวิจัยนี้เกี่ยวข้องกับระบบจัดการไฟล์ในองค์กรเสมือนและคลัสเตอร์คอมพิวเตอร์ ทฤษฎีที่เกี่ยวข้องกับงานวิจัยนี้จึงประกอบด้วย 2 ส่วนคือ (1) ส่วนระบบไฟล์คลัสเตอร์สำหรับจัดการไฟล์ภายในคลัสเตอร์คอมพิวเตอร์ (2) ระบบกริดข้อมูลสำหรับการจัดการไฟล์ภายในองค์กรเสมือน

2.1.1 ระบบไฟล์คลัสเตอร์

ไฟล์เป็นข้อมูลที่ระบบปฏิบัติการใช้อ้างอิงถึงข้อมูลจริงๆ ที่เก็บอยู่บนดิสก์ของเครื่องคอมพิวเตอร์ ในระบบปฏิบัติการแต่ละชนิดจะมีการจัดการไฟล์อย่างคร่าวๆ อยู่สองระดับ

1) ระดับล่างคือการจัดการในส่วนที่ติดต่อกับอุปกรณ์ด้วยไดรเวอร์อุปกรณ์ (Device Driver) การจัดการในระดับนี้จะอ้างอิงไปถึงตำแหน่งที่เก็บไฟล์จริงๆ บนดิสก์เช่น การย้ายข้อมูลจากบล็อกที่ 111 บนดิสก์มาเก็บยังหน่วยความจำหลักของเครื่องคอมพิวเตอร์ เป็นต้น
2) ระดับบนคือการติดต่อกับผู้ใช้และเรียกใช้งานการจัดการไฟล์ในระดับล่าง การจัดการไฟล์ในระดับบนนี้จะนำเสนอโครงสร้างของไฟล์และไดเรกทอรีในลักษณะโครงสร้างต้นไม้ (Tree Structure) ระบบจะจัดเตรียมเครื่องมือ (เช่นบรรทัดคำสั่ง และ API) เพื่อให้ผู้ใช้สามารถจัดการไฟล์ในระดับล่างได้สะดวกขึ้น เช่น คำสั่งแสดงรายการไฟล์ คำสั่งทำสำเนาไฟล์ คำสั่งเคลื่อนย้ายไฟล์ และคำสั่งลบไฟล์ เป็นต้น

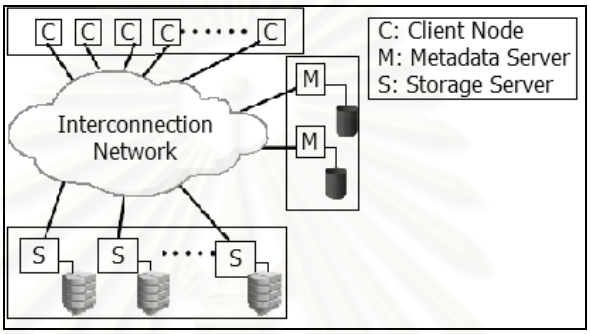
เนื่องจากคลัสเตอร์คอมพิวเตอร์ประกอบด้วยเครื่องคอมพิวเตอร์หลายๆ เครื่องติดต่อกันด้วยระบบเครือข่าย ดังนั้นจึงจำเป็นต้องมีระบบไฟล์ที่ใช้สำหรับจัดการไฟล์บนเครื่องอื่นๆ ด้วย ระบบไฟล์ที่ทำหน้าที่นี้คือ Network File System (NFS) [8] ซึ่งเป็นระบบการแบ่งปันไฟล์บนเครือข่ายโดยใช้รูปแบบ Client-Server ส่วนใหญ่แล้วระบบปฏิบัติการตระกูล UNIX จะติดตั้ง NFS มาให้พร้อมกับการติดตั้งตัวระบบปฏิบัติการ ไฟล์ใน NFS จะเก็บอยู่บนเครื่องที่ให้บริการ ในการใช้งาน ผู้ใช้ (บนเครื่องใช้บริการ) จะต้องเมาท์ (Mount) ไดเรกทอรีบนเครื่องให้บริการมาที่เครื่องของตัวเองและการเข้าถึงไฟล์จะทำผ่านเครือข่ายทั้งการอ่านและการเขียนไฟล์ ข้อดีของ NFS คือผู้ใช้ (บนเครื่องใช้บริการ) สามารถเข้าถึงไฟล์ที่อยู่บนเครื่องให้บริการได้ด้วยวิธีการปกติเช่นเดียวกับการเข้าถึงไฟล์บนเครื่องของตัวเอง

นอกจาก NFS แล้ว คลัสเตอร์คอมพิวเตอร์ยังมีการใช้งานระบบไฟล์อีกชนิดหนึ่ง คือระบบไฟล์คลัสเตอร์ซึ่งเป็นรูปแบบหนึ่งของระบบไฟล์แบบกระจาย (Distributed File System) ในมุมมองของระบบไฟล์คลัสเตอร์จะมองทั้งคลัสเตอร์คอมพิวเตอร์เป็นหนึ่งระบบ ทำให้ผู้ใช้สามารถจัดการไฟล์จาก

เครื่องใดก็ได้ โดยปกติระบบไฟล์คลัสเตอร์มักจะไม่ถูกติดตั้งมากับระบบปฏิบัติการ ดังนั้นผู้ใช้จำเป็นต้องติดตั้งเพิ่มเติมเองตามความต้องการใช้งาน

ในปัจจุบันมีระบบไฟล์คลัสเตอร์อยู่หลายชนิด ซึ่งระบบไฟล์คลัสเตอร์ส่วนใหญ่จะมีลักษณะและส่วนประกอบดังนี้

1) *ขอบเขตชื่อ* (Name Space) คือการกำหนดรูปแบบการเรียกชื่อของไฟล์ที่อยู่ในระบบ ขอบเขตชื่อที่นิยมใช้ได้แก่การกำหนดเป็นพาทในระบบปฏิบัติการยูนิกซ์ (UNIX) หรือการกำหนดเป็น URL



รูปที่ 2.1: สถาปัตยกรรมของ DCFS ซึ่งเป็น CFS ชนิดหนึ่ง [11]

2) *หน้าที่ของเครื่องให้บริการ* ในระบบไฟล์คลัสเตอร์จะมีหน้าที่หลักอยู่สองหน้าที่ หน้าที่แรกคือเครื่องให้บริการเมตะดาต้าทำหน้าที่เก็บข้อมูลของไฟล์ในระบบ อีกหน้าที่หนึ่งคือเครื่องเก็บไฟล์ โดยปกติเครื่องเก็บไฟล์จะมีอยู่หลายเครื่อง ส่วนเครื่องจัดการเมตะดาต้าอาจมีเพียงหนึ่งเครื่องหรือหลายเครื่องก็ได้ นอกจากสองหน้าที่นี้แล้ว ระบบไฟล์คลัสเตอร์บางชนิดอาจกำหนดหน้าที่ของเครื่องเพิ่มขึ้นมาเพื่อช่วยในการทำงานเช่น DCFS2 [12] จะมีเครื่องที่ควบคุมการใช้งานพื้นที่ดิสก์ทั้งหมดที่มีอยู่ในระบบ, GPFS [13] มีเครื่องที่ทำหน้าที่จัดการเรื่องการล็อก (Lock) ไฟล์ เป็นต้น ในระบบไฟล์คลัสเตอร์บางระบบเครื่องหนึ่งเครื่องอาจทำหลายหน้าที่พร้อมกัน แต่ในบางระบบเครื่องหนึ่งเครื่องสามารถทำได้เพียงหนึ่งหน้าที่เท่านั้น

3) *จำนวนเครื่องและวิธีการจัดการเมตะดาต้า* มีสองรูปแบบคือ

   - การจัดการแบบรวมศูนย์ (Centralize) คือใช้เพียงหนึ่งเครื่องทำหน้าที่จัดการเมตะดาต้าทั้งหมด วิธีนี้มีข้อดีคือง่ายต่อการจัดการ แต่ก็มีข้อเสียคือ ทำให้ระบบไม่มีความทนทาน (Robustness) หากเครื่องที่เก็บเมตะดาต้าเสีย จะทำให้ระบบไม่สามารถใช้งานได้ และอาจเกิดปัญหาคอขวด เมื่อมีการติดต่อจำนวนมากมาที่เครื่องจัดการเมตะดาต้าตัวอย่างระบบไฟล์คลัสเตอร์ที่จัดการเมตะดาต้ารูปแบบนี้ได้แก่ PVFS [14]

   - การจัดการแบบกระจาย (De-Centralize) คือใช้วิธีกระจายเมตะดาต้าไปเก็บยังหลายๆ เครื่อง วิธีนี้สามารถกำจัดข้อเสียของวิธีแรกได้ แต่ระบบก็มีความซับซ้อนมากขึ้น เช่นเมื่อมีการสร้างไฟล์ขึ้น ระบบจะต้องตัดสินใจว่าเมตะดาต้าของไฟล์นั้นจะเก็บไว้ที่เครื่องได้ และ

จะต้องมีวิธีการในการค้นหาข้อมูลเมตะดาต้าที่กระจายอยู่ในหลายๆ เครื่อง ตัวอย่างระบบไฟล์คลัสเตอร์ที่จัดการเมตะดาต้ารูปแบบนี้ได้แก่ PVFS2 [15] และ DCFS [11] เป็นต้น

4) *วิธีการเก็บเมตะดาต้า* ซึ่งจะมีผลโดยตรงต่อฟังก์ชันการทำงานเช่น การเพิ่ม, การลบ, และการค้นหาข้อมูล โดยทั่วๆ การเก็บเมตะดาต้าจะมีสองวิธีคือ

   - สร้างไฟล์และกำหนดโปรโตคอลสำหรับการเก็บและจัดการเมตะดาต้าขึ้นเอง วิธีนี้มีข้อดีคือผู้พัฒนาระบบสามารถกำหนดโครงสร้างของไฟล์เพื่อให้เหมาะสมกับใช้งานของระบบไฟล์คลัสเตอร์นั้นๆ ได้ แต่มีข้อเสียคือเสียเวลาในการพัฒนาระบบ

   - เก็บไว้ในฐานข้อมูล (Database) ผู้พัฒนาระบบจะกำหนดโครงสร้างของฐานข้อมูล แล้วเขียนโปรแกรมเพื่อติดต่อใช้งานฐานข้อมูลในการเก็บเมตะดาต้า วิธีนี้มีข้อดีคือสร้างง่ายเนื่องจากฐานข้อมูลมีฟังก์ชันการทำงานที่จำเป็นหลายๆ อย่างอยู่แล้ว นอกจากนี้ระบบฐานข้อมูลในปัจจุบันมี API สำหรับให้ผู้พัฒนาระบบสามารถเขียนโปรแกรมเพื่อเรียกใช้งานฐานข้อมูลได้สะดวก แต่วิธีนี้ก็มีข้อเสียคือการทำงานของระบบไฟล์คลัสเตอร์จะขึ้นอยู่กับฐานข้อมูลที่ใช้ การใช้งานระบบไฟล์คลัสเตอร์จะต้องติดตั้งระบบฐานข้อมูลลงไปด้วย

5) *ความสามารถในการเพิ่มเครื่องคอมพิวเตอร์ในระบบ* หากเครื่องเก็บไฟล์ที่มีอยู่ไม่เพียงพอต่อการใช้งาน ก็จำเป็นต้องเพิ่มเครื่องคอมพิวเตอร์เข้าไปในระบบ ประเด็นที่มีการคำนึงถึงในการเพิ่มเครื่องคอมพิวเตอร์เข้ามาในระบบได้แก่ความยากง่ายในการตั้งค่าระบบและจำนวนเครื่องมากที่สุดที่ระบบไฟล์คลัสเตอร์สามารถรองรับได้โดยที่ประสิทธิภาพการทำงานของระบบยังอยู่ในเกณฑ์ดี

6) *การจัดเก็บไฟล์* มีอยู่สองวิธีคือการเก็บไฟล์ทั้งไฟล์ไว้บนเครื่องเดียว และการแบ่งไฟล์เป็นส่วนๆ และกระจายแต่ละส่วนไปเก็บในเครื่องที่ทำหน้าที่เก็บไฟล์ วิธีหลังมีข้อดีคือทำให้สามารถเก็บไฟล์ที่มีขนาดใหญ่มาก (เกินขนาดของดิสก์หนึ่งอัน) ได้ แต่หากเครื่องที่เก็บไฟล์เครื่องหนึ่งเสียก็จะทำให้มีข้อมูลของไฟล์ไม่ครบ

7) *การเข้าถึงไฟล์* (File Access) คือวิธีการการอ่านเขียนไฟล์ การเข้าถึงไฟล์จะมีอยู่สองรูปแบบคือ

   - การเข้าถึงแบบตามลำดับ (Sequential Access) คือการเข้าถึงไฟล์โดยเริ่มจากตำแหน่งเริ่มต้นของไฟล์ไปจนกระทั่งตำแหน่งสุดท้ายของไฟล์

   - การเข้าถึงแบบขนาน (Parallel Access) คือการเข้าถึงไฟล์หลายๆ ส่วนพร้อมกัน การเข้าถึงแบบนี้จะต้องจัดเก็บไฟล์ไว้บนหลายๆ เครื่องโดยอาจจะใช้วิธีแบ่งไฟล์ หรือการทำซ้ำไฟล์ก็ได้ วิธีการนี้มีข้อดีคือเป็นการเพิ่มแบนด์วิดท์ในการอ่านเขียนไฟล์ ทำให้สามารถอ่านเขียนไฟล์ได้เร็วขึ้น และการใช้งานไฟล์ไม่จำเป็นต้องอ่านตั้งแต่จุดเริ่มต้นของไฟล์ แต่การเข้าถึงไฟล์แบบขนานก็มีความเหมาะสมกับโปรแกรมประยุกต์ประเภทที่สามารถทำงานแบบขนานได้ หรือประเภทที่ใช้งานข้อมูลเพียงบางส่วนของไฟล์เท่านั้น

8) *การทำซ้ำไฟล์* (Replication) เป็นวิธีที่ทำให้ระบบมีความทนทานและมีสภาพพร้อมใช้งาน (Availability) มากขึ้น แต่การทำซ้ำไฟล์ก็จะทำให้เกิดปัญหาเรื่องการจัดการความต้องกัน (Consistency) เมื่อมีการแก้ไขไฟล์บางฉบับ

9) *ฟังก์ชันการทำงานอื่นๆ* เพื่อเพิ่มความสามารถของระบบและความสะดวกในการใช้งาน เช่น การใช้แคชในการอ่านเขียนข้อมูล, การใช้แคชเก็บข้อมูลเมตะดาต้าที่มีการใช้งานบ่อย, การย้ายที่เก็บไฟล์ (File Migration) [16] เพื่อให้เกิดความสมดุลในการใช้งานเครื่องในคลัสเตอร์คอมพิวเตอร์, การเพิ่มเครื่องมือ (Tool) เพื่อช่วยให้การตั้งค่าระบบทำได้ง่าย เป็นต้น

2.1.2 กริดข้อมูล

กริดข้อมูล [4, 5] เป็นระบบกริดประเภทหนึ่งที่ใช้สำหรับเข้าถึงทรัพยากรข้อมูลที่กระจายตัวอยู่ในสถานที่ต่างๆ โดยปกติกริดข้อมูลจะใช้ติดต่อกับบริการหรือโปรแกรมประยุกต์ประเภทใช้ข้อมูลปริมาณมาก โดยปกติกริดข้อมูลจะมีความสามารถพื้นฐานได้แก่ การจัดการเรื่องความปลอดภัย, โปรโตคอลที่ใช้ในการรับส่งข้อมูลปริมาณมากอย่างมีประสิทธิภาพ, และการทำซ้ำข้อมูลเพื่อกระจายข้อมูลไปที่ต่างๆ กริดข้อมูลจะมีลักษณะที่สำคัญดังนี้

1) เกี่ยวข้องกับโปรแกรมประยุกต์ประเภทที่ใช้ข้อมูลปริมาณมาก ซึ่งโดยปกติจะสร้างข้อมูลออกมาในระดับพันล้านไบต์ (Gigabyte) หรือ ล้านล้านไบต์ (Terabyte)

2) มีการกระจายตัวของผู้ใช้และทรัพยากรข้อมูล เช่นนักวิทยาศาสตร์จากหลายๆ ทวีปร่วมมือกันวิจัยในเรื่องเดียวกันก็มีความจำเป็นต้องใช้ข้อมูลในการวิจัยร่วมกัน

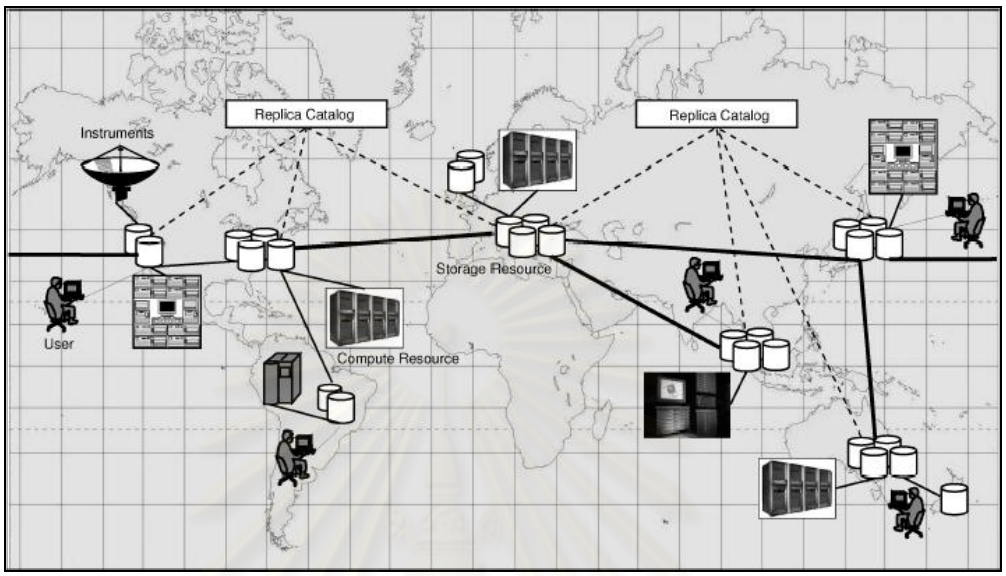
3) ส่วนใหญ่แล้วจะมีแหล่งข้อมูลเพียงที่เดียว เนื่องจากการสร้างข้อมูลปริมาณมหาศาลจะต้องใช้เครื่องมือที่ทันสมัยซึ่งมักมีอยู่แค่ที่เดียวในโครงการวิจัย เมื่อเครื่องมือนี้สร้างข้อมูลมาแล้ว ข้อมูลจึงถูกทำซ้ำและกระจายไปยังสถานที่ต่างๆ เพื่อให้นักวิจัยจากที่อื่นสามารถเข้าถึงข้อมูลได้ และเมื่อมีการแก้ไขข้อมูลต้นฉบับก็จะมีการส่งต่อไปยังข้อมูลที่ทำซ้ำเอาไว้

4) มีการใช้ขอบเขตชื่อในการอ้างอิงถึงไฟล์เพียงชื่อเดียว แต่สามารถอ้างอิงได้ทั้งไฟล์ต้นฉบับและไฟล์ที่ทำซ้ำไว้ ซึ่งระบบจะเป็นผู้เลือกว่าจะให้ผู้ใช้ได้ใช้ไฟล์ฉบับไหน ซึ่งอาจเลือกจากความห่างระหว่างสถานที่เก็บไฟล์และสถานที่ทำงาน

5) เนื่องจากทรัพยากรมีจำนวนจำกัดและกระจายตัวอยู่ ดังนั้นจึงต้องมีการจัดการทรัพยากรที่ดีเพื่อให้ผู้ใช้สามารถใช้งานทรัพยากรได้อย่างมีประสิทธิภาพ

6) เนื่องจากกริดเกิดจากการแบ่งปันทรัพยากรของหลายๆ องค์กร ดังนั้นเจ้าของทรัพยากรจะต้องสามารถกำหนดได้ว่าจะให้ใครสามารถเข้าใช้งานทรัพยากรของตัวเองได้บ้าง และผู้ใช้แต่ละคนจะสามารถใช้งานทรัพยากรได้ในระดับไหน

7) โดยปกติทรัพยากรที่มารวมกันในกริดไม่จำเป็นที่จะต้องเหมือนกัน ทั้งในส่วนอุปกรณ์และส่วนโปรแกรม ดังนั้นโปรแกรมที่จะนำมาใช้งานในกริดข้อมูลได้จะต้องสามารถใช้งานข้ามแพลตฟอร์มได้



รูปที่ 2.2: กริดข้อมูล [4]

รูปที่ 2.2 แสดงตัวอย่างของกริดข้อมูลที่ประกอบด้วยเครื่องเก็บข้อมูล, เครื่องคอมพิวเตอร์สมรรถนะสูง, และผู้ใช้ที่กระจายอยู่ในสถานที่ต่างๆ เส้นทึบแสดงเส้นทางหลักที่ใช้รับส่งข้อมูลระหว่างเครื่องเก็บข้อมูลเครื่องต่างๆ ส่วนเส้นที่บางกว่าจะมีปริมาณการรับส่งข้อมูลที่น้อยกว่า ซึ่งอาจจะเป็นระหว่างเครื่องเก็บข้อมูลกับเครื่องคอมพิวเตอร์สมรรถนะสูง หรือระหว่างเครื่องเก็บข้อมูลกับผู้ใช้

บริการที่สำคัญประการหนึ่งที่ใช้ในระบบกริดในปัจจุบันคือ บริการรับส่งไฟล์ระหว่างทรัพยากรต่างๆ ในกริดผ่านเครือข่าย โปรโตคอลหนึ่งที่ทำหน้าที่นี้คือ GridFTP [17] ซึ่งเป็นโปรโตคอลที่ขยายมาจาก FTP โดยเพิ่มความสามารถในการทำงานได้แก่

1) *ด้านความปลอดภัย* GridFTP สนับสนุนการทำงานของ Grid Security Infrastructure (GSI) และ Kerberos ในการพิสูจน์ตัวจริง

2) *การเคลื่อนย้ายไฟล์แบบบุคคลที่สาม* (Third-Party Transfer) ผู้ใช้สามารถรับส่งไฟล์ระหว่างสองเครื่องใดก็ได้ในกริด โดยที่ไม่จำเป็นต้องล็อกอินเข้าไปใช้งานที่เครื่องต้นทางหรือปลายทาง

3) *การเคลื่อนย้ายไฟล์แบบขนาน* (Parallel Data Transfer) เป็นการใช้ TCP หลายๆ สายเพื่อเพิ่มแบนด์วิดท์ในการเคลื่อนย้ายข้อมูล

4) *การเคลื่อนย้ายแบบแบ่งไฟล์* (Striped Data Transfer) เป็นการเคลื่อนย้ายไฟล์จากหลายๆ เครื่อง โดยทุกเครื่องที่เป็นแหล่งข้อมูลจะมีไฟล์ต้นฉบับอยู่อาจจะทั้งหมดหรือบางส่วนก็ได้ ส่วนฝั่งที่รับไฟล์

จะมีเพียงเครื่องเดียวหรือหลายเครื่องก็ได้เช่นกัน แต่ละเครื่องที่เป็นแหล่งข้อมูลจะส่งส่วนหนึ่งของไฟล์ไปให้เครื่องปลายทาง วิธีนี้เป็นการเพิ่มประสิทธิภาพเพิ่มเติมจากการเคลื่อนย้ายไฟล์แบบขนาน

5) *การเคลื่อนย้ายไฟล์บางส่วน* (Partial File Transfer) เนื่องจากโปรแกรมประยุกต์บางประเภทต้องการใช้ข้อมูลแค่ส่วนหนึ่งจากข้อมูลทั้งหมดที่มีขนาดใหญ่มาก การส่งเฉพาะข้อมูลที่มีความจำเป็นจะช่วยลดจำนวนข้อมูลที่ต้องเคลื่อนย้ายได้

6) *การเคลื่อนย้ายข้อมูลแบบเชื่อถือได้* โดยมีวิธีจัดการกับข้อผิดพลาดที่เกิดขึ้นในการเคลื่อนย้ายไฟล์และสามารถเริ่มต้นใหม่ได้ถ้าการรับส่งไฟล์ไม่สำเร็จ

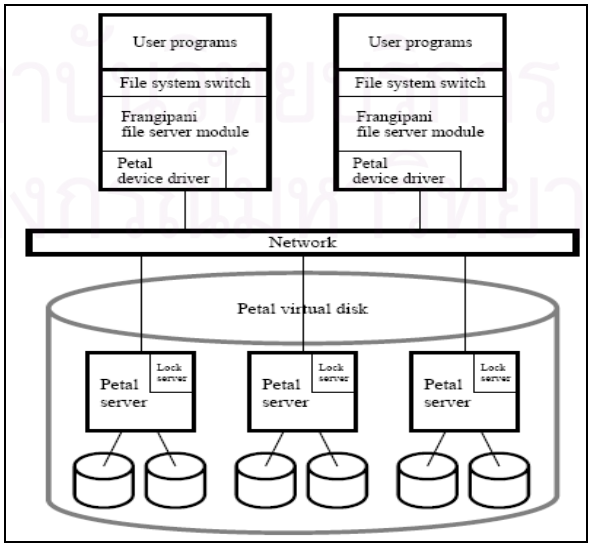
2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

2.2.1 ระบบไฟล์คลัสเตอร์

งานวิจัยที่เกี่ยวข้องในส่วนของระบบไฟล์คลัสเตอร์นั้นมีอยู่หลายงาน ซึ่งแต่ละงานก็ใช้วิธีการที่แตกต่างกันและมีความเหมาะสมกับสถานการณ์ที่แตกต่างกัน แต่โดยส่วนใหญ่แล้ว ระบบไฟล์คลัสเตอร์จะใช้วิธีการรวบรวมที่เก็บข้อมูลจากหลายๆ เครื่องมาสร้างเป็นที่เก็บข้อมูลเสมือน (Virtual Storage) ทำให้ผู้ใช้มองเห็นที่เก็บข้อมูลขนาดใหญ่ ในที่นี้ได้แบ่งระบบไฟล์คลัสเตอร์ออกเป็นสองประเภทตามรูปแบบการจัดการไฟล์นั่นคือ ระบบไฟล์คลัสเตอร์ที่มีการจัดการไฟล์แบบตามลำดับ และ ระบบไฟล์คลัสเตอร์ที่มีการจัดการไฟล์แบบขนาน

2.2.1.1 ระบบไฟล์คลัสเตอร์ที่จัดการไฟล์แบบตามลำดับ

ระบบไฟล์คลัสเตอร์แบบตามลำดับจะเก็บไฟล์ทั้งไฟล์ไว้บนเครื่องเดียว ตัวอย่างระบบไฟล์คลัสเตอร์ประเภทนี้ได้แก่ Frangipani [20], TH-CluFS [21], Lustre [22] เป็นต้น



รูปที่ 2.3: โครงสร้างของ Frangipani [20]

เดิมที Frangipani ถูกออกแบบมาเพื่อใช้สำหรับระบบคอมพิวเตอร์แบบกระจาย ซึ่งก็สามารถนำมาประยุกต์ใช้งานกับคลัสเตอร์คอมพิวเตอร์ได้เช่นกัน การทำงานของ Frangipani ใช้ส่วนโปรแกรมที่เรียกว่า Petal ในการจัดการที่เก็บข้อมูลเสมือน ส่วนระบบ Frangipani จะทำหน้าที่รับคำสั่งจากผู้ใช้แล้วส่งต่อไปยังส่วน Petal วิธีนี้ทำให้ Frangipani ไม่มีข้อมูลของตำแหน่งที่เก็บไฟล์จริงๆ ทำให้ไม่สามารถสนับสนุนการทำงานของโปรแกรมจัดลำดับงานแบบส่งงานไปหาข้อมูลได้ นอกจากนี้การทำงานของ Frangipani อยู่ภายในระบบปฏิบัติการ ทำให้การย้ายระบบจากแพลตฟอร์มหนึ่งไปอีกแพลตฟอร์มหนึ่งทำได้ยาก

TH-CluFS เป็นระบบไฟล์คลัสเตอร์สำหรับระบบปฏิบัติการลินุกซ์ ที่พัฒนาขึ้นมาจาก NFS2 [19] โดยการแก้ไขส่วนให้บริการเพื่อให้เหมาะกับการใช้งานบนคลัสเตอร์คอมพิวเตอร์ TH-CluFS ยังมีการเพิ่มเติมวิธีการต่างๆ เพื่อเพิ่มประสิทธิภาพการทำงานของระบบเช่น การใช้แคชเก็บข้อมูลเมตะดาต้าที่มีการใช้งานบ่อย, การจัดสมดุลเครื่องที่ทำหน้าที่เก็บไฟล์เพื่อกระจายไฟล์ไปเก็บยังเครื่องต่างๆ และเนื่องจาก TH-CluFS ไม่ได้ทำงานในระดับเคอร์เนล ทำให้สามารถใช้งานได้ในทุกสถาปัตยกรรม

Lustre เป็นระบบไฟล์คลัสเตอร์ที่รองรับการทำงานของคลัสเตอร์คอมพิวเตอร์ขนาดใหญ่ ในระบบของ Lustre จะใช้เครื่องเก็บข้อมูลเมตะดาต้าเครื่องหนึ่งทำหน้าที่เก็บข้อมูลคำสั่งในอดีตและสถานะของระบบในช่วงเวลาต่างๆ เพื่อใช้ในการกู้ระบบเมื่อเกิดข้อผิดพลาดขึ้น ส่วนการอ่านเขียนไฟล์บนเครื่องเก็บไฟล์นั้น Lustre ทำผ่านไดรเวอร์ของอุปกรณ์อีกทีหนึ่ง ทำให้ Lustre สามารถใช้ที่เก็บข้อมูลได้หลายชนิดไม่จำกัดเฉพาะดิสก์เท่านั้น แต่เนื่องจาก Lustre ทำงานในระดับเคอร์เนลทำให้การติดตั้งทำได้ยาก ต้องทำการแก้ไขเคอร์เนล (Patch Kernel) และเมื่อติดตั้งเสร็จแล้วจะต้องปิดเครื่องและเปิดใหม่เพื่อเข้าไปยังเคอร์เนลใหม่ของ Lustre

2.2.1.2 ระบบไฟล์คลัสเตอร์ที่จัดการไฟล์แบบขนาน

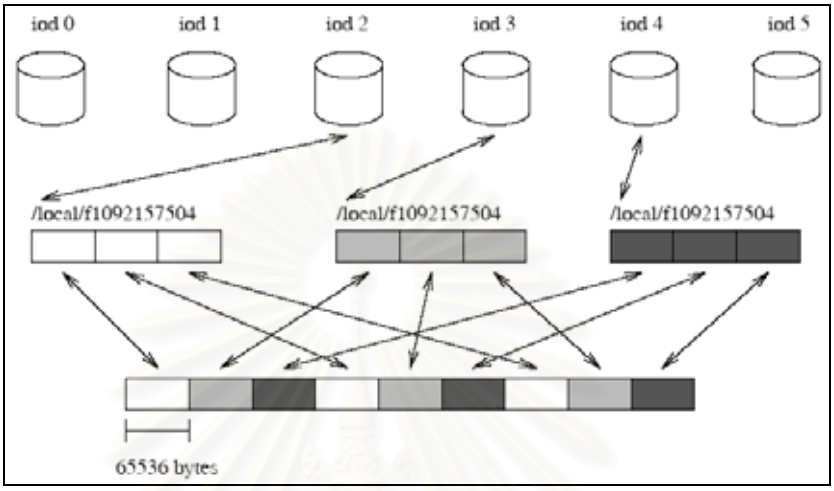
ระบบไฟล์คลัสเตอร์แบบขนานจะใช้การจัดการไฟล์โดยการแบ่งไฟล์ออกเป็นหลายๆ ส่วน และกระจายไปเก็บยังเครื่องต่างๆ ในคลัสเตอร์คอมพิวเตอร์ ระบบไฟล์ประเภทนี้จึงเหมาะสมกับโปรแกรมประยุกต์สองประเภทคือ

- มีการใช้งาน I/O ปริมาณมากและสามารถทำงานแบบขนานได้ เพราะตัวโปรแกรมจะสามารถเข้าถึงไฟล์หลายๆ ส่วนพร้อมกัน ซึ่งเป็นการเพิ่มแบนด์วิดท์ในการเข้าถึงไฟล์
- ใช้ข้อมูลเพียงบางส่วนของไฟล์ เพราะตัวโปรแกรมจะสามารถเข้าถึงส่วนที่จำเป็นในการทำงานโดยไม่ต้องอ่านตั้งแต่ต้นไฟล์

ตัวอย่างระบบไฟล์ประเภทนี้ได้แก่ PVFS [14], PVFS2 [15], และ DCFS [11]

PVFS และ PVFS2 เป็นระบบไฟล์คลัสเตอร์ที่สร้างขึ้นโดยมีจุดประสงค์การทำงานเหมือนกัน แต่มีวิธีการทำงานที่แตกต่างกันออกไป อย่างเช่น ใน PVFS2 เครื่องเก็บข้อมูลเมตะดาต้าสามารถมีกี่เครื่องก็

ได้ ในขณะที่ใน PVFS มีได้แค่เครื่องเดียว และใน PVFS2 ไม่มีการใช้แคชจึงไม่เหมาะกับโปรแกรมประยุกต์ประเภทที่เรียกดูข้อมูลสถานะของไฟล์บ่อยๆ เพราะว่า PVFS2 จะต้องอ่านข้อมูลใหม่ทุกครั้งที่มีการเรียกใช้



รูปที่ 2.4: ตัวอย่างการกระจายไฟล์ของระบบไฟล์คลัสเตอร์ที่จัดการไฟล์แบบขนาน [14]

Dawning Cluster File System (DCFS) เป็นระบบไฟล์คลัสเตอร์ที่สร้างขึ้นเพื่อใช้งานกับเครื่อง Dawning4000-L ใน DCFS ใช้เครื่องจัดการเมตะดาต้าหลายเครื่อง โดยมีเครื่องหนึ่งที่เป็นตัวควบคุมว่าเมตะดาต้าของไฟล์ใดจะเก็บไว้ที่เครื่องใด และ DCFS ยังมีวิธีการเพิ่มเติมเพื่อเพิ่มประสิทธิภาพในการอ่านเขียนไฟล์ เช่น การแคชข้อมูล และการใช้หลายเทรด (Multi-Thread) ในการควบคุมการอ่านเขียนไฟล์ นอกจากนี้ DCFS ยังมีระบบตรวจสอบสภาพของระบบ และเครื่องมือที่ช่วยให้การจัดการและการตั้งค่าระบบสามารถทำได้สะดวก

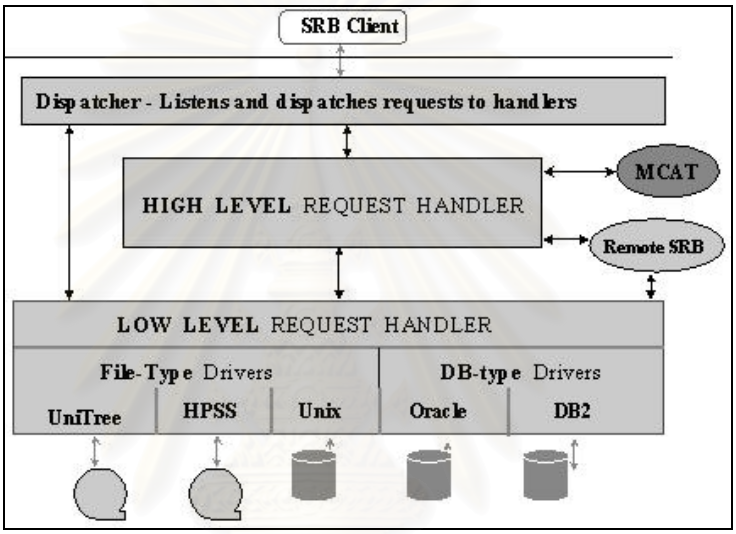
อย่างไรก็ตามระบบไฟล์คลัสเตอร์แบบขนานส่วนใหญ่ก็มีข้อจำกัดที่คล้ายๆ กันคือ หากเครื่องที่เก็บไฟล์เครื่องใดเครื่องหนึ่งเสียหาย ก็จะทำให้ไฟล์ทุกไฟล์ที่มีส่วนประกอบเก็บอยู่บนเครื่องนั้นมีข้อมูลไม่ครบและอาจไม่สามารถนำมาใช้งานได้ ดังนั้นเครื่องที่ทำหน้าที่เก็บไฟล์ในระบบไฟล์คลัสเตอร์แบบขนานจึงมักจะเป็นเครื่องที่ถูกกำหนดให้ทำน้าที่เก็บไฟล์อย่างเดียว (เช่นเดียวกับเครื่องให้บริการไฟล์) ซึ่งไม่ตรงกับความต้องการของโครงงานวิจัยนี้ที่ต้องการให้เครื่องที่เก็บไฟล์ทำหน้าที่ในการประมวลผลด้วย

2.2.2 กริดข้อมูล

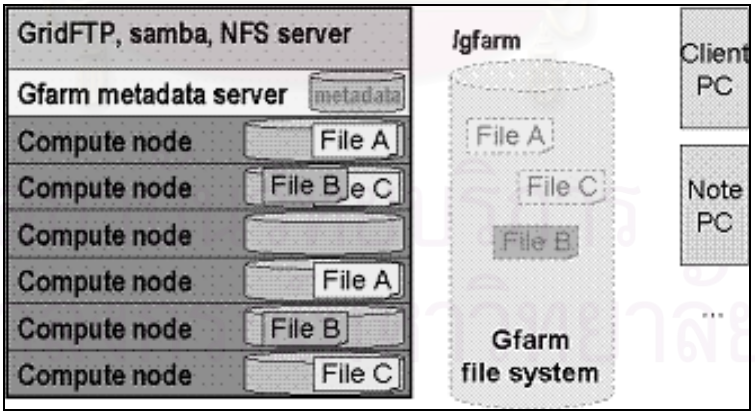
งานวิจัยส่วนนี้เป็นงานในลักษณะมิดเดิลแวร์ที่ทำหน้าที่จัดการไฟล์ภายในเครื่องที่เป็นทรัพยากรของกริดเพื่อสร้างเป็นกริดข้อมูล งานวิจัยในกลุ่มนี้ได้แก่

Storage Resource Broker (SRB) [6, 23] เป็นมิดเดิลแวร์สำหรับการจัดการไฟล์ในระบบกริดข้อมูล SRB ใช้ขอบเขตชื่อในการอ้างอิงถึงทรัพยากรในระบบ ทำให้ SRB สามารถใช้งานได้กับระบบที่มี

ความแตกต่างกันทั้งในเรื่องของแพลตฟอร์มและวิธีการจัดเก็บข้อมูล SRB รองรับการสร้างวงกริดข้อมูลหลายวงเพื่อกระจายการทำงาน โดยแต่ละวงจะจัดการเมตะดาต้าของตัวเอง และสามารถใช้งานเมตะดาต้าของวงอื่นได้ SRB ใช้ระบบ MCAT (Metadata Catalog) [24] ในการจัดการเมตะดาต้า การใช้งาน SRB จะต้องประกอบด้วยสองส่วนคือ (1) ไดรเวอร์ที่เตรียมไว้เพื่อการติดต่อไปยังอุปกรณ์เก็บข้อมูลแต่ละชนิด และ (2) ส่วนอินเตอร์เฟสสำหรับสั่งงาน ซึ่งได้แก่คำสั่งในลักษณะบรรทัดคำสั่ง (Command Line) ของระบบปฏิบัติการยูนิกซ์ (UNIX) และ API สำหรับให้ผู้พัฒนาระบบนำไปใช้เขียนโปรแกรมเพื่อใช้งาน SRB นอกจากนี้ SRB ยังมีเครื่องมือหลายชนิดเพื่อเพิ่มความสะดวกในการใช้งานของผู้ใช้



รูปที่ 2.5: แสดงการออกแบบ SRB [23]



รูปที่ 2.6: ระบบไฟล์ Gfarm [25]

Grid Datafarm (Gfarm) [7, 25] ใช้ในระบบที่ใช้ไฟล์ขนาดใหญ่มากจนไม่สามารถเก็บอยู่ในดิสก์อันเดียวได้ ไฟล์จะถูกแบ่งและกระจายไปเก็บยังเครื่องต่างๆ ในระบบกริด โดยในมุมมองของผู้ใช้ยังเห็นไฟล์หลายๆ ส่วนเป็นไฟล์เดียว เมื่อมีการเรียกใช้ไฟล์ใน Gfarm ด้วยคำสั่งของ Gfarm (เช่น gfrun) โปรแกรมจะส่งงานไปประมวลผลยังเครื่องที่มีข้อมูลเก็บอยู่ เพื่อให้เกิดการเคลื่อนย้ายข้อมูลน้อยที่สุด

Gfarm จึงเหมาะกับงานที่ใช้ข้อมูลเพียงบางส่วนในไฟล์ขนาดใหญ่ หรืองานที่สามารถทำแบบขนานได้ และไฟล์ที่ใช้ใน Gfarm จะเป็นประเภทที่ไม่มีการแก้ไขหลังจากสร้างขึ้นมาแล้ว ในส่วนการใช้งาน Gfarm มี API สำหรับให้ผู้พัฒนาระบบสามารถเขียนโปรแกรมเรียกใช้งานได้ สำหรับผู้ใช้ทั่วไปสามารถใช้งาน Gfarm ได้ผ่านทางบรรทัดคำสั่งหรือใช้โปรแกรม Samba และเรียกใช้งานผ่านทางระบบปฏิบัติการวินโดวส์ (Windows) โดยทำการตั้งค่าระบบและลงโปรแกรมที่เกี่ยวข้องเพิ่มเติม

LegionFS [26] เป็นมิดเดิลแวร์ที่ใช้ในข่ายงานบริเวณกว้าง (WAN: Wide Area Network) ระบบนี้จะมองส่วนประกอบต่างๆ ในระบบเป็นวัตถุ แต่ละวัตถุจะประกาศอินเตอร์เฟสของตนเองเพื่อให้ผู้ใช้สามารถเรียกใช้งานหรือให้วัตถุอื่นนำไปใช้เพื่อขยายเพิ่มเติมการทำงาน และจะมีวัตถุส่วนหนึ่งที่เป็นตัวควบคุมการทำงานของระบบ เช่นวัตถุซึ่งทำหน้าที่ตัดสินใจว่าไฟล์ใดควรจะเก็บไว้ที่เครื่องใดในกริด แต่ละวัตถุจะบังคับใช้นโยบายของตัวเองในการจัดการด้านความปลอดภัย นอกจากนี้ LegionFS ยังใช้แคชหลายระดับเพื่อเพิ่มประสิทธิภาพในการค้นหาไฟล์ในระบบ แต่ยังไม่สนับสนุนการทำซ้ำไฟล์อย่างเต็มที่

NeST [27] เป็นระบบไฟล์ที่ใช้ในกริด การทำงานจะแบ่งออกเป็น 4 ส่วนคือส่วนติดต่อเครือข่าย, ส่วนจ่ายงาน, ส่วนจัดการการเคลื่อนย้ายไฟล์, และส่วนจัดการที่เก็บข้อมูล การใช้งาน NeST สามารถทำได้โดยการสั่งคำสั่งผ่านทางเครือข่าย ซึ่งจุดเด่นประการหนึ่งของ NeST คือสามารถรองรับการทำงานได้หลายโปรโตคอล เช่น FTP, GridFTP, HTTP, และ NFS เป็นต้น นอกจากนี้ NeST ยังใช้วิธีการจองพื้นที่บนที่เก็บข้อมูลก่อนที่จะมีการสร้างไฟล์ เพื่อการันตีว่าเครื่องคอมพิวเตอร์มีที่เก็บข้อมูลเพียงพอที่จะสร้างไฟล์ได้

ในบรรดางานวิจัยเรื่องกริดข้อมูลที่กล่าวมา Gfarm มีความคล้ายคลึงกับโครงงานนี้มากที่สุด เนื่องจากเป็นระบบเดียวที่สนับสนุนในเรื่องนโยบายการส่งงานไปหาข้อมูล ในขณะที่งานวิจัยอื่นไม่ได้เน้นในเรื่องนี้ อย่างไรก็ตามทุกๆ งานวิจัยที่กล่าวมาเน้นการใช้งานบนเครื่องที่มีลักษณะเป็นเครื่องให้บริการไฟล์ ทุกๆ เครื่องที่ทำหน้าที่เก็บไฟล์ในงานวิจัยเหล่านี้จำเป็นต้องมี IP สาธารณะ (Public IP) เพื่อใช้ในการติดต่อกับเครื่องอื่นๆ ในระบบกริดผ่านทางอินเตอร์เน็ท ในขณะที่ในงานวิจัยนี้มุ่งเน้นที่จะเก็บไฟล์ไว้ที่เครื่องคำนวณบนคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรในระบบกริด ซึ่งเครื่องคำนวณเหล่านี้จะมีเพียง IP ส่วนตัว (Private IP) ไว้สำหรับติดต่อกับเครื่องอื่นๆ ที่อยู่ภายในคลัสเตอร์เดียวกัน หากจะใช้เครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ทำหน้าที่เก็บไฟล์ของระบบไฟล์เหล่านี้ เครื่องคำนวณจะต้องมี IP สาธารณะ ซึ่งโครงสร้างของระบบแบบนี้ไม่ใช่โครงสร้างของคลัสเตอร์คอมพิวเตอร์ที่ใช้ในงานวิจัยนี้

2.3 สรุป

ในบทนี้ได้กล่าวถึงทฤษฎีและงานวิจัยที่มีความเกี่ยวข้องกับงานวิจัยนี้ โดยได้แบ่งส่วนประกอบออกเป็นสองระดับ ระดับแรกเป็นระดับของการจัดการไฟล์ภายในคลัสเตอร์ ซึ่งแนวคิดของงานในระดับนี้จะเป็นลักษณะของระบบไฟล์คลัสเตอร์ซึ่งช่วยในการจัดการไฟล์ภายในคลัสเตอร์คอมพิวเตอร์ ในระดับที่สองจะเป็นการจัดการไฟล์ในระดับองค์กรเสมือน ซึ่งแนวคิดของงานในระดับนี้คือกริดข้อมูลซึ่งช่วยในการจัดการไฟล์ที่เก็บอยู่ในองค์กรที่กระจายตัวอยู่ตามสถานที่ต่างๆ

ดังที่ได้กล่าวไปแล้วว่า เราสามารถแบ่งปัญหาในเรื่องของการจัดการไฟล์ได้เป็นสองระดับคือในระดับของคลัสเตอร์คอมพิวเตอร์และในระดับองค์กรเสมือน ดังนั้นในงานวิจัยนี้จึงแบ่งการออกแบบและการพัฒนาระบบออกเป็นสองระดับดังกล่าว โดยทั้งสองระบบจะทำงานประสานกันเพื่อแก้ปัญหาที่เกิดขึ้นด้วยวิธีการสนับสนุนการทำงานแบบส่งงานไปหาข้อมูล ในสองบทถัดไปจะกล่าวถึงการออกแบบและการพัฒนาระบบที่ใช้จัดการไฟล์ในระดับคลัสเตอร์คอมพิวเตอร์และระดับองค์กรเสมือนตามลำดับ

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 3

การจัดการไฟล์ในระดับคลัสเตอร์คอมพิวเตอร์

ในบทนี้นำเสนอการออกแบบและการพัฒนาระบบไฟล์คลัสเตอร์ที่ใช้ในระบบปฏิบัติการลินุกซ์เพื่อสนับสนุนการทำงานแบบส่งงานไปหาข้อมูล ในงานวิจัยนี้ได้สร้างระบบต้นแบบเพื่อใช้ในการทดลอง โดยตั้งชื่อว่า “SPACE Cluster File System” (SCFS) โครงสร้างและหลักการทำงานของ SCFS จะคล้ายกับระบบไฟล์คลัสเตอร์ทั่วๆ ไป แต่การออกแบบจะเน้นในเรื่องการใช้เครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ในการจัดเก็บไฟล์เพื่อให้โปรแกรมประยุกต์สามารถเข้าถึงไฟล์แบบท้องถิ่นได้ โครงสร้างและหลักการทำงานของ SCFS มีดังนี้

3.1 ความต้องการของระบบ

1) การเก็บไฟล์ไว้บนเครื่องที่ทำการประมวลผล
2) การสนับสนุนการทำงานของตัวจัดลำดับงานให้สามารถส่งงานไปทำที่เครื่องที่มีไฟล์เก็บอยู่ได้
3) มีการเก็บข้อมูลตำแหน่งของไฟล์ที่อยู่ในระบบ เพื่อช่วยให้สามารถสอบถามตำแหน่งที่อยู่ของไฟล์ที่อยู่ในระบบได้
4) การใช้งานสามารถทำได้สองทางคือผ่านทางบรรทัดคำสั่งและ API ที่ระบบจัดไว้ให้

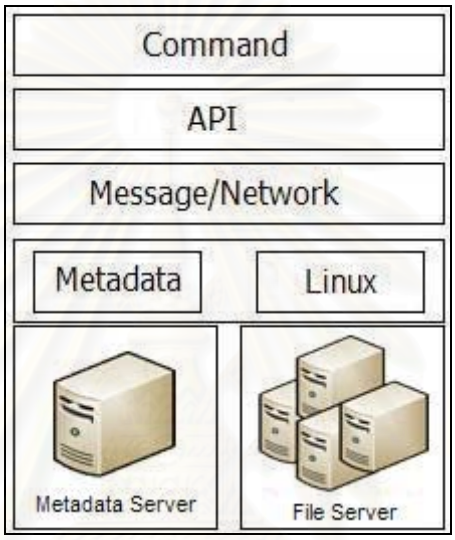
3.2 สถาปัตยกรรม

SCFS ประกอบด้วยเครื่องให้บริการสองประเภทคือเครื่องให้บริการเมตะดาต้าและเครื่องให้บริการไฟล์ โดยกำหนดให้เครื่องให้บริการเมตะดาต้ามีได้เพียงหนึ่งเครื่อง ส่วนเครื่องให้บริการไฟล์สามารถมีได้หลายเครื่อง เครื่องในคลัสเตอร์คอมพิวเตอร์สามารถทำหน้าที่ใดก็ได้ แต่ไม่สามารถทำสองหน้าที่พร้อมกันได้ และเพื่อให้ตรงตามจุดประสงค์ของงานวิจัยนี้ ระบบที่ใช้ทำการทดลองจึงกำหนดให้เครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ทำหน้าที่เป็นเครื่องให้บริการไฟล์และเครื่องควบคุมทำหน้าที่เป็นเครื่องให้บริการเมตะดาต้า

เนื่องจากข้อมูลของไฟล์ในระบบเก็บอยู่ในเครื่องให้บริการเมตะดาต้า และเครื่องควบคุมเป็นเครื่องเดียวในคลัสเตอร์คอมพิวเตอร์ที่สามารถติดต่อกับเครือข่ายภายนอกได้ ดังนั้นประโยชน์อีกประการหนึ่งของการใช้เครื่องควบคุมเป็นเครื่องให้บริการเมตะดาต้าคือ ระบบจัดการไฟล์ในระดับองค์กรเสมือน (ในบทที่ 4) สามารถติดต่อและส่งคำร้องขอข้อมูลของไฟล์ในระบบไฟล์คลัสเตอร์กับเครื่องควบคุมได้โดยตรง ถ้าหากใช้เครื่องคำนวณเป็นเครื่องให้บริการเมตะดาต้า คำร้องขอจะต้องถูกส่งต่อจากเครื่องควบคุมไปยังเครื่องคำนวณเครื่องนั้นอีกต่อหนึ่ง

SCFS ทำงานอยู่ระดับบนของเคอร์เนลของระบบปฏิบัติการทำให้การติดตั้งทำได้ง่าย การออกแบบโปรแกรมเป็นลักษณะโปรแกรมเชิงวัตถุ (Object Oriented Programming) โดยส่วนประกอบส่วนใหญ่ของโปรแกรม รวมถึงแกนหลักของระบบ สร้างขึ้นด้วยภาษาจาวา (Java) ยกเว้นโปรแกรมส่วนที่ทำหน้าที่เรียกใช้งานไลบรารีท้องถิ่น (Native Library) ของระบบปฏิบัติการซึ่งสร้างขึ้นด้วยภาษาซี (C) โปรแกรมส่วนภาษาจาวาและภาษาซีจะติดต่อกันผ่านทาง Java Native Interface (JNI)

สถาปัตยกรรมของ SCFS ได้แบ่งออกเป็นระดับทั้งหมดสี่ระดับตามรูปที่ 3.1 ได้แก่



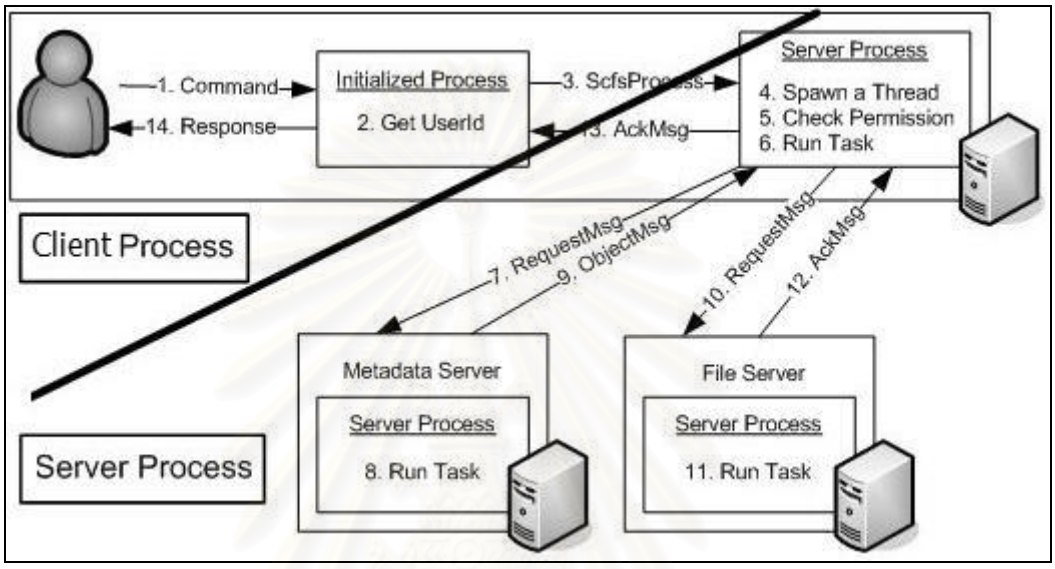
รูปที่ 3.1: สถาปัตยกรรมของ SCFS

1) Command เป็นระดับที่อยู่บนสุด ทำหน้าที่รับคำสั่งจากผู้ใช้ เพื่อให้ผู้ใช้สามารถจัดการไฟล์ในระบบได้ คำสั่งที่ให้บริการในระบบมีลักษณะเป็นบรรทัดคำสั่งในระบบปฏิบัติการลินุกซ์
2) API เป็นระดับที่ทำหน้าที่เป็นอินเตอร์เฟสสำหรับส่วนโปรแกรมอื่น (รวมทั้งส่วน Command ในระบบนี้ด้วย) เพื่อให้ส่วนโปรแกรมอื่นสามารถเรียกใช้ฟังก์ชันการทำงานของ SCFS ได้
3) Message/Network เป็นระดับที่ทำหน้าที่ในการรับส่งข้อมูลระหว่างเครื่องในระบบ ข้อมูลที่รับส่งภายในระบบได้แก่ คำสั่งการทำงาน, ข้อความแสดงสถานะของคำสั่ง, ไฟล์, ฯลฯ
4) ระดับล่างสุดประกอบด้วยสองส่วนคือ
   - Metadata ทำงานอยู่บนเครื่องให้บริการเมตะดาต้าโปรแกรมส่วนนี้ทำหน้าที่จัดการคำสั่งที่เกี่ยวข้องกับเมตะดาต้าของระบบ
   - Linux ทำหน้าที่ติดต่อกับระบบปฏิบัติการลินุกซ์เพื่อจัดการไฟล์ในระบบ

<u>หมายเหตุ</u>: หลังจากนี้หากใช้คำว่า “*วัตถุ*” หรือชื่อวัตถุภาษาอังกฤษที่เป็นตัวเอียงจะหมายถึงวัตถุในภาษาจาวา

3.3 โปรเซส

ในการออกแบบโปรเซส ได้ใช้รูปแบบ “Client-Server” โดยที่ทุกเครื่องในระบบทำหน้าที่เป็นทั้งผู้ให้บริการและใช้บริการของเครื่องอื่น โปรเซสใน SCFS จะมีสองลักษณะคือ (หมายเลขขั้นตอนการทำงานในรูปที่ 3.2 ใช้อ้างอิงในหัวข้อย่อยที่ 3.3.1 และ 3.3.2)



รูปที่ 3.2: แสดงตัวอย่างโปรเซสและชนิดของข้อความที่รับส่งในระบบ

3.3.1 โปรเซสเริ่มต้นการเรียกใช้บริการ (Client Process)

ผู้ใช้จะเป็นผู้เริ่มต้นการทำงานของ SCFS โดย (1) การเรียกใช้บริการผ่านทางบรรทัดคำสั่งหรือ API จากนั้น (2) โปรเซสเริ่มต้นการเรียกใช้บริการจะสร้าง *UserId* ของผู้ใช้ขึ้นมาเพื่อใช้ในการตรวจสอบสิทธิ์การทำงานคำสั่งนั้น และ (3) สร้างคำร้องขอสำหรับงานเพื่อส่งไปให้โปรเซสให้บริการ จากนั้นโปรเซสเริ่มต้นการเรียกใช้บริการจะรอจนกว่าโปรเซสให้บริการจะทำงานเสร็จ และ (14) เมื่องานเสร็จแล้วก็จะแจ้งผลการทำงานกลับไปให้ผู้ใช้ทราบ

3.3.2 โปรเซสให้บริการ (Server Process)

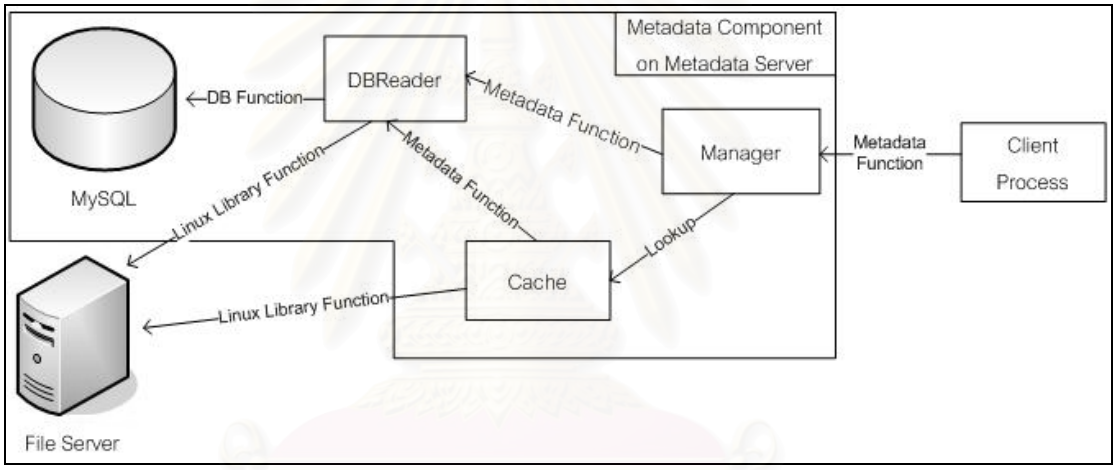
โปรเซสส่วนนี้ทำงานในลักษณะเดมอน (Daemon) ของระบบปฏิบัติการลินุกซ์ โดย SCFS ใช้โปรแกรม Java Service Wrapper (JSW) [31] ในการสร้างเดมอน โปรเซสนี้จะเปิดพอร์ท (Port) ประเภท TCP ขึ้นมาหนึ่งพอร์ทเพื่อรับคำร้องขอจากเครื่องอื่น ซึ่งผู้ร้องขออาจจะเป็น “โปรเซสให้บริการ” จากเครื่องใดก็ได้ในระบบ หรือ “โปรเซสเริ่มต้นการเรียกใช้บริการ” ที่อยู่บนเครื่องเดียวกัน

การทำงานของส่วนให้บริการจะเริ่มต้นเมื่อได้รับคำร้องขอ โดยเมื่อมีคำร้องขอเข้ามา (4) โปรเซสให้บริการจะสร้างเทรด (Thread) ขึ้นมาเพื่อจัดการกับคำร้องขอนั้น (5) หากคำร้องข้อมาจากโปรเซสเริ่มต้นการเรียกใช้บริการ จะมีการตรวจสอบสิทธิ์การใช้งานของผู้ใช้ก่อนที่ (6) จะมีการทำงาน

โดยในการทำงาน อาจจะต้องมีการสร้างและส่งคำร้องขอไปยังโปรเซสให้บริการบนเครื่องอื่นที่เกี่ยวข้องกับงานนั้น เช่น (7) ถ้าเป็นงานที่เกี่ยวข้องกับเมตะดาต้า ก็ส่งไปให้เครื่องให้บริการเมตะดาต้า (10) ถ้าเป็นงานที่เกี่ยวข้องกับไฟล์ ก็ส่งไปให้เครื่องให้บริการไฟล์ที่มีไฟล์นั้นเก็บอยู่ (8, 11) จากนั้นเครื่องให้บริการจึงทำงานตามที่ระบุไว้ในคำร้องขอ (9, 12, 13) เมื่อทำงานเสร็จเรียบร้อยแล้ว ก็จะส่งผลลัพธ์และสถานะของการทำงานกลับไปให้โปรเซสที่เป็นผู้เรียก

3.4 การจัดการเมตะดาต้า

ส่วนเมตะดาต้าจะเป็นส่วนที่เกี่ยวข้องกับข้อมูลไฟล์และไดเรกทอรีในระบบ ดังนั้นโปรแกรมในส่วนนี้จะทำงานในโปรเซสให้บริการบนเครื่องให้บริการเมตะดาต้า การออกแบบและพัฒนาระบบส่วนเมตะดาต้านี้ได้แบ่งออกเป็นสองส่วนคือ



รูปที่ 3.3: แสดงส่วนประกอบสำคัญของระบบส่วนเมตะดาต้า

3.4.1 ส่วนเก็บข้อมูล

ข้อมูลของไฟล์และไดเรกทอรีในระบบจะเก็บอยู่ในระบบสองส่วนคือในระบบปฏิบัติการลินุกซ์และในระบบฐานข้อมูล MySQL โดยในระบบฐานข้อมูลจะเก็บข้อมูลที่ใช้ในการบอกตำแหน่งของไฟล์และข้อมูลเพิ่มเติมที่ใช้ในการจัดการไดเรกทอรีภายในระบบ เมื่อมีการร้องขอข้อมูลเมตะดาต้า ในเบื้องต้นระบบจะร้องขอข้อมูลในระบบฐานข้อมูลก่อน การร้องขอนี้จะทำผ่านทาง*วัตถุ*ที่ชื่อว่า *DBReader* จากนั้นจึงตรวจสอบจากคำร้องขอที่มาจากผู้เรียกใช้งานว่าจำเป็นต้องใช้ข้อมูลจากเครื่องให้บริการไฟล์หรือไม่ หากจำเป็นจึงส่งคำร้องขอไปยังเครื่องให้บริการไฟล์ที่มีไฟล์เก็บอยู่เพื่อร้องขอข้อมูลเพิ่มเติม ซึ่งการร้องขอข้อมูลที่อยู่ในระบบปฏิบัติการจะทำผ่านฟังก์ชันในไลบรารีท้องถิ่นของระบบปฏิบัติการลินุกซ์ สุดท้ายจึงนำข้อมูลทั้งหมดที่ได้รับมาสร้างเป็น*วัตถุ*เพื่อส่งกลับไปให้ผู้เรียกใช้งาน

สาเหตุที่ต้องมีการตรวจสอบความจำเป็นในการร้องขอข้อมูลผ่านเครือข่ายไปยังเครื่องให้บริการไฟล์เนื่องจาก การร้องขอข้อมูลผ่านเครือข่ายจะใช้เวลามากกว่าการร้องขอข้อมูลจากระบบฐานข้อมูล

อย่างมาก หากข้อมูลดังกล่าวไม่มีความจำเป็นต้องใช้ในการทำงาน ก็จะเสียเวลาในการร้องขอโดยเปล่าประโยชน์ การตรวจสอบนี้จะช่วยลดปริมาณการร้องขอข้อมูลผ่านเครือข่ายและเพิ่มประสิทธิภาพการทำงานของระบบได้

3.4.1.1 การเชื่อมต่อกับฐานข้อมูล

ในส่วนเก็บข้อมูลนี้ได้ใช้งาน *DBReader* ทำหน้าที่เป็นตัวเชื่อมต่อกับระบบฐานข้อมูล เนื่องจาก *DBReader* เป็นส่วนเดียวในระบบที่รู้จักโครงสร้างของฐานข้อมูล (Database Schema) ที่ใช้เก็บข้อมูลเมตะดาต้า ดังนั้น *DBReader* จึงทำหน้าที่ทั้งสร้างการเชื่อมต่อกับระบบฐานข้อมูล และสร้างคำสั่ง SQL เพื่อเรียกใช้งานระบบฐานข้อมูล ผู้ดูแลระบบจะต้องสร้างไฟล์การตั้งค่าระบบเพื่อกำหนดข้อมูลที่จำเป็นต้องใช้ในการติดต่อกับฐานข้อมูลก่อนเปิดระบบขึ้นมา สำหรับการออกแบบ *DBReader* ได้ถูกออกแบบให้เป็นวัตถุที่ไม่มีการเก็บสถานะ (Stateless Object) ดังนั้น เมื่อสร้าง *DBReader* ขึ้นมาหนึ่งอันแล้ว ก็สามารถนำไปใช้งานได้หลายครั้งโดยหลายเทรดที่ไม่มีความเกี่ยวข้องกัน

3.4.1.2 *วัตถุ*เก็บข้อมูลเมตะดาต้า

*วัตถุ*ที่ทำหน้าที่เก็บข้อมูลเมตะดาต้ามีอยู่หลายชนิด ได้แก่

- *Attribute* ใช้เก็บคุณสมบัติของไฟล์ ข้อมูลที่เก็บได้แก่ สิทธิ์การใช้งาน, เจ้าของ, กลุ่ม, ขนาด, เวลาที่สร้างไฟล์, เวลาล่าสุดที่มีการแก้ไขไฟล์ เป็นต้น ข้อมูลที่ใช้ในการสร้าง *Attribute* บางส่วนมาจากการสอบถามระบบปฏิบัติการ บางส่วนเป็นข้อมูลที่เก็บอยู่ในระบบฐานข้อมูล
- *FileInstance* ใช้เก็บข้อมูลเครื่องคอมพิวเตอร์ที่ทำหน้าที่เก็บไฟล์แต่ละฉบับ ข้อมูลนี้เป็นข้อมูลที่เก็บอยู่ในระบบฐานข้อมูล
- *MetaData* ใช้เก็บชื่อและชนิดของไฟล์ นอกจากนี้ยังใช้เก็บ*วัตถุ*อื่น ซึ่งก็คือ *Attribute* และ *FileInstance* อย่างไรก็ตาม เฉพาะ *MetaData* ที่มีชนิดเป็นไฟล์เท่านั้นจึงจะมี *FileInstance* เก็บอยู่และจะมีจำนวน*วัตถุ*เท่ากับจำนวนฉบับของไฟล์ไฟล์นั้น

3.4.2 ส่วนจัดการเมตะดาต้า

โปรแกรมส่วนนี้ทำหน้าที่จัดการงานที่เกี่ยวข้องกับเมตะดาต้า โดยที่ตัวโปรแกรมจะประกอบด้วยโปรแกรมหลักๆ สองส่วนคือแคชและผู้จัดการ

3.4.2.1 แคช (Cache)

การทำงานของ SCFS จะต้องมีการร้องขอข้อมูล *MetaData* ในทุกๆ คำสั่ง ซึ่งในการสร้าง *MetaData* ในหลายๆ ครั้งจะต้องติดต่อกับส่วนเก็บข้อมูลทั้งในระบบฐานข้อมูลบนเครื่องให้บริการเมตะดาต้าและในระบบปฏิบัติการบนเครื่องให้บริการไฟล์ที่มีไฟล์เก็บอยู่ เพื่อลดจำนวนครั้งในการสอบถามข้อมูลสำหรับสร้างเมตะดาต้า งานวิจัยนี้จึงสร้างระบบแคชขึ้นมาเพื่อใช้สำหรับเก็บข้อมูล *MetaData* ที่

เพิ่งมีการใช้งาน หากมีความต้องการใช้ *MetaData* นี้ในคำสั่งต่อๆ ไป ก็สามารถนำข้อมูลจากแคชไปใช้งานได้ทันทีโดยไม่ต้องสร้างขึ้นมาใหม่

แคชจะจัดเก็บข้อมูลในลักษณะคู่อันดับคือการแมพ (Map) จากข้อมูลหนึ่งไปอีกข้อมูลหนึ่ง คู่อันดับที่เก็บอยู่ในแคชมีอยู่ 2 ประเภท คือ

1) แมพจากพาธของไฟล์ไปยัง ID ของไฟล์ คู่อันดับชนิดนี้ใช้เมื่อผู้ใช้ต้องการร้องขอเมตะดาต้า โดยส่งข้อมูลพาธของไฟล์มาเป็นอินพุต ID ที่ได้จะถูกนำไปใช้ในการค้นหาข้อมูลในคู่อันดับอีกอันหนึ่ง
2) แมพจาก ID ของไฟล์ไปยัง *MetaData* ของไฟล์ เมื่อมีการส่งค่า ID เข้ามา หากในแคชมี *MetaData* สำหรับ ID นี้เก็บอยู่ก็ส่ง *MetaData* นั้นคืนกลับไปให้ผู้เรียกใช้ได้ทันที แต่หากไม่มีก็จะทำการร้องขอจากส่วนเก็บข้อมูล เมื่อได้ข้อมูลมาแล้วก็จะเก็บลงในแคชและส่งกลับไปให้ผู้เรียกใช้

คู่อันดับแต่ละคู่ที่เก็บอยู่ในแคชจะมีข้อมูลแฝงเก็บอยู่คือ Time-to-Live (TTL) เพื่อใช้บอกว่าคู่อันดับใดจะถูกลบออกจากแคชเมื่อใด หากมีการเรียกใช้งานคู่อันดับใด TTL ของคู่อันดับนั้นจะได้รับการแก้ไขค่าให้มากขึ้น SCFS ได้สร้างเทรด (Thread) ขึ้นมาเทรดหนึ่งสำหรับตรวจสอบและลบคู่อันดับที่มี TTL น้อยกว่าเวลาปัจจุบัน นอกจากนี้ในกรณีที่แคชเต็ม SCFS จะใช้นโยบาย Least-Recently-Use (LRU) สำหรับการเลือกลบคู่อันดับเก่าเพื่อนำคู่อันดับใหม่มาเก็บแทน ซึ่งในการใช้งานนโยบาย LRU นี้สามารถตรวจสอบได้จากข้อมูล TTL ที่เก็บอยู่ในทุกๆ คู่อันดับ

3.4.2.2 **ผู้จัดการ**

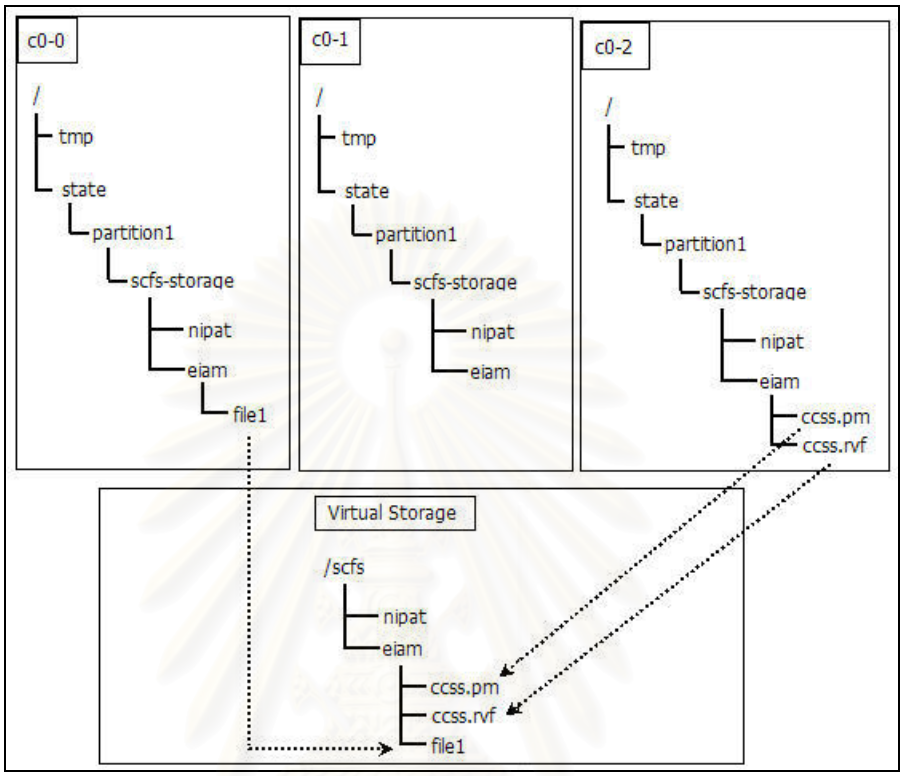
โปรแกรมส่วนนี้จะทำหน้าที่เป็นอินเตอร์เฟสสำหรับการจัดการเมตะดาต้า ทุกๆ ส่วนของระบบที่ต้องการทำงานที่เกี่ยวข้องกับเมตะดาต้าจะต้องติดต่อผ่านทางส่วนผู้จัดการ ฟังก์ชันที่ให้บริการโดยส่วนผู้จัดการได้แก่ การสร้าง, ลบ, แก้ไขข้อมูล, และการร้องขอข้อมูล *MetaData* เป็นต้น หน้าที่ของโปรแกรมส่วนนี้คือการตรวจสอบความถูกต้องของพารามิเตอร์ และการเตรียมการเบื้องต้นก่อนที่จะเรียกใช้ฟังก์ชันการจัดการเมตะดาต้าที่ให้บริการโดยส่วนประกอบอื่นๆ เช่นแคชและส่วนจัดเก็บข้อมูล

3.5 **การจัดการไฟล์**

SCFS รวบรวมที่เก็บข้อมูลจากเครื่องให้บริการไฟล์มาสร้างเป็นที่เก็บข้อมูลเสมือน โดยเครื่องแต่ละเครื่องจะสร้างไดเรกทอรีหนึ่งขึ้นมาสำหรับใช้เก็บไฟล์ในระบบ เช่น “/state/partition1/scfs-storage” เป็นต้น ข้อมูลพาธของไดเรกทอรีนี้จะถูกกำหนดลงในไฟล์การตั้งค่าของระบบ เพื่อให้ระบบสามารถรู้ได้ว่าจะเก็บไฟล์ไว้ที่ใดบนเครื่องให้บริการไฟล์แต่ละเครื่อง

SCFS ใช้ขอบเขตชื่อในการอ้างอิงถึงไฟล์ในระบบเพื่อให้ผู้ใช้มองเห็นไฟล์ทั้งหมดในระบบเป็นหนึ่งมุมมอง โดยขอบเขตชื่อที่ใช้ใน SCFS คือ “/scfs” ผู้ใช้สามารถจัดการไฟล์จากเครื่องใดก็ได้โดยใช้ขอบเขตชื่อนี้ เช่น ผู้ใช้สั่งคำสั่งโดยอ้างอิงถึงไฟล์ชื่อ “/scfs/eiam/file1” เมื่อคำสั่งนี้ถูกส่งไปถึงเครื่อง

ให้บริการไฟล์ที่มีไฟล์นี้เก็บอยู่ ชื่อไฟล์ก็จะถูกแปลไปเป็นตำแหน่งที่เก็บไฟล์จริงๆ บนเครื่องนั้นเช่น อาจจะเป็น “/state/partiton1/scfs-storage/eiam/file1” เป็นต้น



รูปที่ 3.4: แสดงโครงสร้างต้นไม้ของระบบไฟล์ใน SCFS

สำหรับการจัดเก็บไฟล์บนเครื่องให้บริการไฟล์นั้น SCFS ใช้วิธีการเก็บในลักษณะเป็นไฟล์ธรรมดาของระบบปฏิบัติการลินุกซ์ ผู้ใช้สามารถล็อกอินเข้าไปที่เครื่องให้บริการไฟล์แล้วอ่าน-เขียนไฟล์ด้วยวิธีปกติเช่นเดียวกับไฟล์ในลินุกซ์ โปรแกรมส่วนที่ติดต่อกับลินุกซ์ในการจัดการไฟล์นี้เขียนขึ้นด้วยภาษาซี โดยเรียกใช้ไลบรารีมาตรฐานของลินุกซ์เช่น unistd.h, sys/stat.h, stdio.h เป็นต้น

ในการใช้งาน SCFS ผู้ใช้จะต้องนำไฟล์เข้ามาในระบบก่อนด้วยคำสั่งลงทะเบียนไฟล์ (scfs-reg) เมื่อผู้ใช้สั่งคำสั่งนี้ระบบจะนำข้อมูลของไฟล์ไปเก็บลงในส่วนจัดการเมตะดาต้า จากนั้นจึงนำตัวไฟล์เข้าสู่ระบบ ซึ่งมีวิธีการอยู่ 2 วิธี คือ

1) การทำสำเนาไฟล์ต้นฉบับไปเก็บยังไดเรกทอรีของระบบ วิธีนี้ผู้ใช้สามารถกำหนดได้ว่าจะให้ไฟล์ไปเก็บบนเครื่องให้บริการไฟล์เครื่องใด
2) การสร้างลิงค์ (Link) จากไดเรกทอรีของระบบไปยังไฟล์ต้นฉบับ วิธีนี้มีข้อดีคือหากไฟล์มีขนาดใหญ่มาก การสร้างลิงค์จะช่วยลดเวลาที่ใช้ในการทำสำเนาได้มาก แต่วิธีนี้มีข้อจำกัดคือไฟล์ต้นฉบับจะต้องเก็บอยู่ภายในเครื่องให้บริการไฟล์เท่านั้น และลิงค์ก็จะอยู่บนเครื่องเดียวกับไฟล์ต้นฉบับ

หลังจากลงทะเบียนไฟล์เข้าสู่ระบบแล้ว ผู้ใช้สามารถจัดการกับไฟล์ภายใน SCFS ได้โดยใช้บรรทัดคำสั่งหรือ API ที่ระบบมีให้ตามที่แสดงในภาคผนวก ข.

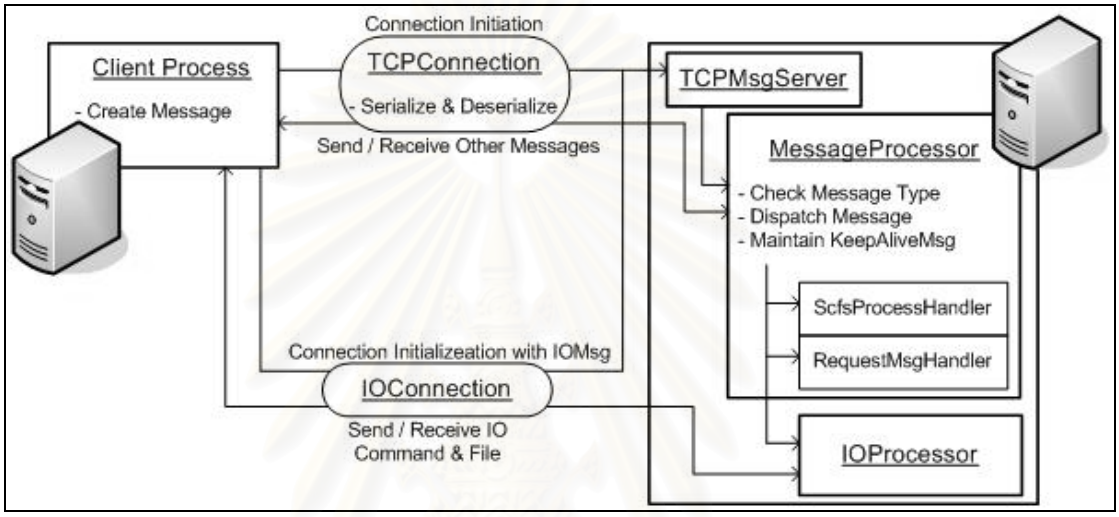
ตามที่กล่าวมาแล้วว่า SCFS สนับสนุนวิธีการส่งงานไปหาข้อมูล แต่หากเครื่องคำนวณที่มีไฟล์เก็บอยู่ไม่ว่าง ก็จะทำให้โปรแกรมที่จะใช้ไฟล์นั้นต้องรออยู่ในคิวของโปรแกรมจัดลำดับงาน เพื่อแก้ปัญหานี้ SCFS จึงได้สร้างฟังก์ชันการทำซ้ำไฟล์ (scfs-rep) ขึ้นมาเพื่อเพิ่มสภาพพร้อมใช้งานของไฟล์ ผู้ใช้สามารถสั่งทำซ้ำไฟล์ไปยังเครื่องคำนวณเครื่องอื่นที่ว่างงาน จากนั้นจึงค่อยส่งงานตามไป

3.6 ส่วนเครือข่ายและข้อความ

การติดต่อกันระหว่างโปรเซสในระบบใช้ TCP เป็นโปรโตคอลในการติดต่อ โปรเซสจะส่งข้อความหากันในการทำงาน โดยข้อความที่ใช้ในระบบมีอยู่หลายประเภทสำหรับจุดประสงค์การทำงานที่แตกต่างกัน ประเภทของข้อความที่มีการใช้งานใน SCFS ได้แก่

- *ScfsProcess* (ตามรูปที่ 3.2 คือขั้นตอนที่ 3) เป็นข้อความที่สร้างขึ้นเพื่อร้องขอการทำงานคำสั่งหนึ่งคำสั่ง เช่น การสร้างไดเรกทอรี การลบไฟล์ การแสดงรายการไฟล์ เป็นต้น ข้อความประเภทนี้สร้างขึ้นโดยโปรเซสเริ่มต้นการเรียกใช้บริการ (ซึ่งถูกสร้างขึ้นโดยผู้ใช้) ดังนั้นในการจัดการข้อความชนิดนี้จะต้องมีการตรวจสอบสิทธิ์การใช้งานของผู้ใช้ด้วย
- *RequestMsg* (ตามรูปที่ 3.2 คือขั้นตอนที่ 7 และ 10) เป็นข้อความที่สร้างขึ้นเพื่อใช้สำหรับร้องขอการทำงานฟังก์ชันย่อยหนึ่งฟังก์ชัน เช่น การสร้างข้อมูลเมตะดาต้าของไดเรกทอรี การลบข้อมูลเมตะดาต้าของไฟล์ การลบฉบับของไฟล์บนเครื่องให้บริการไฟล์ เป็นต้น ข้อความชนิดนี้สร้างขึ้นโดยโปรเซสให้บริการเท่านั้น ซึ่งก็หมายความว่าเมื่อถึงขั้นตอนนี้สิทธิ์การใช้งานของผู้ใช้ได้ผ่านการตรวจสอบแล้ว ดังนั้นโปรเซสที่ได้รับคำร้องขอจึงไม่จำเป็นต้องตวจสอบสิทธิ์การใช้งานอีกรอบ
- *AckMsg* ใช้แจ้งข้อมูลสถานะการทำงานของคำร้องขอต่างๆ
- *ErrorMsg* ใช้แจ้งข้อมูลความผิดพลาดเกิดขึ้นในการทำงาน
- *GeneralMsg* เป็นข้อความทั่วไปที่ใช้ส่งระหว่างเครื่อง อาจจะใช้เป็นการตอบกลับคำร้องขอหรือเป็นการส่งข้อมูลอื่นระหว่างเครื่อง
- *ObjectMsg* ใช้สำหรับส่ง*วัตถุ*ในระบบเช่น *UserId, MetaData*, และ *Attribute* เป็นต้น
- *IOMsg* เป็นข้อความที่ใช้เริ่มต้นการเชื่อมต่อประเภท IO
- *KeepAliveMsg* เป็นข้อความสำหรับรักษาสถานะของตัวเชื่อมต่อไม่ให้ถูกปิด เนื่องจากการเชื่อมต่อแบบ TCP จะมีการกำหนดอายุของการเชื่อมต่อ หากไม่มีการใช้งานเป็นเวลาหนึ่งการเชื่อมต่อนั้นก็จะถูกปิดลง

เนื่องจากงานใน SCFS มีอยู่หลายประเภท ดังนั้นคำร้องขอทั้งในส่วนของ *ScfsProcess* และ *RequestMsg* จึงประกอบด้วย*วัตถุ*ลูกหลายชนิด ซึ่ง*วัตถุ*ลูกแต่ละชนิดก็จะระบุการทำงานเฉพาะอย่างของตนเองเช่น*วัตถุ*หนึ่งอาจจะจัดการเรื่องการสร้างไฟล์ อีก*วัตถุ*อาจจัดการเรื่องการลบไฟล์ เป็นต้น การออกแบบการทำงานของ*วัตถุ*ทั้งสองชนิดนี้ได้ใช้ Strategy Design Pattern [32] ซึ่งทำให้การจัดการ*วัตถุ*ทั้งสองชนิดนี้สามารถทำได้โดยใช้ฟังก์ชันเพียงสองฟังก์ชันเท่านั้น คือฟังก์ชันหนึ่งสำหรับชนิด *ScfsProcess* และอีกฟังก์ชันหนึ่งสำหรับชนิด *RequestMsg*



รูปที่ 3.5: แสดงส่วนประกอบหลักและหน้าที่ของส่วนประกอบ ในส่วนเครือข่าย

*วัตถุ*ทั้งหมดใน SCFS จะผ่านกระบวนการ Serialization ของภาษาจาวาก่อนการรับส่งผ่านเครือข่าย *วัตถุ*ชนิดอื่นที่ไม่ใช้*วัตถุ*ประเภทข้อความเช่น *UserId* และ *MetaData* จะถูกเก็บเป็นสมาชิกของ*วัตถุ*ประเภทข้อความก่อนส่ง โดย SCFS ใช้*วัตถุ*ชื่อ *TCPConnection* ในการรับส่งข้อความชนิดต่างๆ ระหว่างโปรเซส ฟังก์ชันหลักๆ ที่มีให้บริการได้แก่ การส่งข้อความทั่วไป, การส่งคำร้องขอประเภท *ScfsProcess*, การส่งคำร้องขอประเภท *RequestMsg*, และการส่งไฟล์ เป็นต้น คำสั่งทุกๆ คำสั่งในระบบจะมีการเรียกใช้ *TCPConnection* ในการทำงานเพื่อติดต่อกับโปรเซสอื่น และเมื่อมีการเชื่อมต่อเกิดขึ้นแล้วก็สามารถใช้ตัวเชื่อมต่อนี้รับส่ง*วัตถุ*ประเภทข้อความกี่ข้อความก็ได้จนกว่าตัวเชื่อมต่อจะถูกปิดลง

เพื่อเพิ่มประสิทธิภาพของการทำงาน SCFS ได้มีการใช้งานตัวเชื่อมต่อแบบขนานสำหรับงานประเภทที่สามารถทำพร้อมกันหลายๆ เครื่องได้ เช่นคำสั่งลบไฟล์หลายๆ ไฟล์ที่อยู่บนเครื่องคนละเครื่องกัน ตัวเชื่อมต่อแบบขนานจะสร้าง *TCPConnection* ขึ้นมาหลายๆ อันและสร้างเทรดขึ้นมาตามจำนวน *TCPConnection* เพื่อควบคุมการทำงานของ *TCPConnection* แต่ละอัน วิธีนี้ทำให้สามารถสั่งงานไปยังเครื่องทุกเครื่องที่เกี่ยวข้องกับงานให้ประมวลผลพร้อมกันได้โดยไม่ต้องรอให้ทำเสร็จทีละเครื่อง

ดังที่กล่าวในหัวข้อ 3.3 ในการทำงาน โปรเซสหนึ่งจะต้องเรียกใช้บริการของอีกโปรเซสหนึ่ง โดยโปรเซสที่ทำหน้าที่เป็นผู้ใช้บริการจะสร้างตัวเชื่อมต่อ (ซึ่งก็คือ *TCPConnection*) ไปยังโปรเซสที่ทำหน้าที่

เป็นผู้ให้บริการเพื่อร้องขอการทำงาน ฝั่งโปรเซสผู้ให้บริการจะใช้*วัตถุ*ชื่อ *TCPMsgServer* ทำหน้าที่เปิดพอร์ทเพื่อรับการเชื่อมต่อจากผู้ใช้บริการ เมื่อ *TCPMsgServer* ได้รับคำร้องขอ (ทั้งชนิด *ScfsProcess* และ *RequestMsg*) ก็จะสร้างเทรดเพื่อจัดการกับคำร้องขอนั้น เทรดที่สร้างขึ้นเป็น*วัตถุ*ประเภท *MessageProcessor* ดังที่แสดงในรูปที่ 3.5 *TCPMsgServer* จะทำหน้าที่แค่รับการเชื่อมต่อเท่านั้น เมื่อถึงขั้นตอนการรับส่งข้อความ ผู้ใช้บริการจะติดต่อกับ *MessageProcessor* โดยตรง หน้าที่หลักของ *MessageProcessor* มี 3 อย่างคือ

1) ตรวจสอบว่าข้อความที่ได้รับมานั้นเป็นประเภทอะไร โดยข้อความหลักๆ ที่มีการส่งเข้ามาคือ *KeepAliveMsg*, ข้อความประเภท *ScfsProcess*, ข้อความประเภท *RequestMsg*, *IOMsg*, และข้อความสำหรับปิดการเชื่อมต่อ
2) ทำงานตามความเหมาะสมกับข้อความที่ได้รับ เช่น หากเป็น *KeepAliveMsg* ก็ไม่ต้องทำอะไร, หากเป็น *ScfsProcess* หรือ *RequestMsg* ก็เรียกฟังก์ชันการทำงานที่เหมาะสมกับข้อความประเภทนั้น, หากเป็น *IOMsg* ก็จัดการเตรียมการรอรับคำสั่งประเภท IO, และหากเป็นข้อความสำหรับปิดการเชื่อมต่อก็ดำเนินการปิดตัวเชื่อมต่อ
3) รับส่ง *KeepAliveMsg* เพื่อรักษาสถานภาพของตัวเชื่อมต่อ

นอกจาก *TCPConnection* แล้วยังมีตัวเชื่อมต่ออีกประเภทหนึ่งคือ *IOConnection* ซึ่งใช้สำหรับการทำงานคำสั่งประเภท IO โดย *IOConnection* จะใช้สำหรับการเข้าถึงไฟล์ทีละหนึ่งไฟล์ ฟังก์ชันที่ให้บริการได้แก่การเปิดไฟล์, การปิดไฟล์, การอ่านไฟล์, การเขียนไฟล์, และการตัดปลายไฟล์ โปรเซสที่ต้องการทำงานประเภท IO จะสร้าง *IOConnection* ขึ้นมาและเริ่มต้นการทำงานด้วยการส่งข้อความประเภท *IOMsg* ไปให้เครื่องให้บริการไฟล์ที่มีไฟล์ที่ต้องการใช้งานเก็บอยู่ ฝั่งเครื่องให้บริการจะรับการเชื่อมต่อด้วย *TCPMsgServer* และส่งต่อการทำงานไปที่ *MessageProcessor* เช่นเดียวกับกรณีของ *TCPConnection* เมื่อ *MessageProcessor* ตรวจสอบแล้วว่าข้อความที่ส่งเข้ามาเป็นประเภท *IOMsg* ก็จะสร้าง *IOProcessor* ขึ้นมาเพื่อจัดการกับการเชื่อมต่อนี้แทน หลังจากนั้นการรับส่งคำสั่งประเภท IO และไฟล์จะเป็นการติดต่อระหว่างโปรเซสผู้ใช้บริการและ *IOProcessor* โดยตรง

3.7 อินเตอร์เฟส

ส่วนอินเตอร์เฟสเป็นส่วนที่ใช้สำหรับสร้างโปรเซสเริ่มต้นการเรียกใช้บริการ (ในหัวข้อที่ 3.3.1) โดยโปรแกรมส่วนนี้จะอยู่ในระดับ Command และ API ของสถาปัตยกรรมในรูปที่ 3.1 โปรแกรมในระดับ Command สร้างขึ้นโดยมีวัตถุประสงค์ให้ผู้ใช้ทั่วไปสามารถจัดการไฟล์ใน SCFS ได้ ตัวโปรแกรมจะแบ่งออกเป็น 2 ส่วน คือ

1) Shell Script ซึ่งทำหน้าที่ตรวจสอบและตั้งค่าตัวแปรสภาพแวดล้อม (Environment Variable) ที่จำเป็นต้องใช้ในการทำงานเช่น “SCFS_HOME”, “LD_LIBRARY_PATH” เป็นต้น

2) โปรแกรมภาษาจาวา หน้าที่ของโปรแกรมส่วนนี้คือการตรวจสอบอาร์กิวเมนต์ที่ได้รับจากบรรทัดคำสั่ง ในส่วนนี้ได้ใช้ไลบรารี Apache Commons CLI [33] ช่วยในการทำงาน และเมื่อจัดการกับอาร์กิวเมนต์ที่ได้รับมาเรียบร้อยแล้วก็เรียกใช้งานโปรแกรมในระดับ API ต่ออีกทีหนึ่ง

การติดตั้ง การใช้งาน และ API ของ SCFS แสดงอยู่ในภาคผนวก ข.

3.8 ระบบล็อก (Log System)

SCFS ใช้ไลบรารี Apache Log4j [34] ช่วยในการจัดการเรื่องระบบล็อก โดยแบ่งการล็อกออกเป็นสองระดับคือระดับโปรเซสให้บริการและระดับโปรเซสเริ่มต้นการเรียกใช้บริการ และจะมีการกำหนดขนาดของไฟล์ที่ใช้บันทึกล็อกเอาไว้ หากมีข้อมูลเกินกว่าขนาดที่กำหนด ข้อมูลที่เก่าสุดก็จะถูกลบออกไป ผู้ดูแลระบบสามารถแก้ไขไฟล์ที่ใช้ในการตั้งค่าระบบได้ ส่วนผู้ใช้งานทั่วไปสามารถสร้างไฟล์สำหรับตั้งค่าระบบล็อกในระดับโปรเซสเริ่มต้นการเรียกใช้บริการได้โดยการเก็บไฟล์ไว้ที่ไดเรกทอรีบ้านของตนเอง (~/.scfs/log4j-client.properties) ซึ่งหากมีไฟล์นี้อยู่ SCFS จะใช้ไฟล์นี้แทนไฟล์ที่เป็นค่าปริยายของระบบ

3.9 โปรแกรมส่งงาน

โดยปกติ การทำงานของตัวจัดลำดับงาน (Job Scheduler) จะไม่สนับสนุนการทำงานแบบส่งงานไปหาข้อมูล ทั้งนี้เนื่องจาก

1) ไฟล์ที่ใช้ในการทำงานมักจะเก็บไว้ที่เครื่องให้บริการไฟล์และเมาท์ไปที่เครื่องคำนวณด้วย NFS โปรแกรมจัดลำดับงานสามารถส่งงานไปประมวลผลที่เครื่องใดก็ได้ และการอ่านเขียนไฟล์บนเครื่องคำนวณจะทำผ่านเครือข่าย (Remote File Access) ไปยังเครื่องให้บริการไฟล์
2) หากไฟล์ของผู้ใช้เก็บอยู่บนเครื่องที่ทำการคำนวณอยู่แล้ว โปรแกรมจัดลำดับงานก็ไม่มีข้อมูลว่า ไฟล์ที่ใช้ในการประมวลผลเก็บอยู่ที่เครื่องใด

ดังนั้นเพื่อสนับสนุนการทำงานแบบส่งงานไปหาข้อมูล โครงงานวิจัยนี้จึงได้สร้างโปรแกรมส่งงานขึ้นมา โปรแกรมส่งงานนี้พัฒนาขึ้นด้วยภาษา Perl โดยจะทำงานร่วมกับโปรแกรมจัดลำดับงานบนเครื่องคลัสเตอร์คอมพิวเตอร์ โดยโปรแกรมจัดลำดับงานที่ใช้ในงานวิจัยนี้คือ Sun Grid Engine (SGE) [35] ซึ่งเป็นโปรแกรมจัดลำดับงานที่ติดตั้งมาพร้อมกับระบบปฏิบัติการ NPACI Rocks Cluster [10]

ก่อนการใช้งานโปรแกรมส่งงานจะต้องทำการลงทะเบียนไฟล์ที่ต้องการจะใช้ในงานไว้ใน SCFS ก่อน ผู้ใช้สามารถเรียกใช้งานโปรแกรมส่งงานได้ผ่านทางบรรทัดคำสั่ง โดยมีรูปแบบการใช้งานดังนี้

```
Usage: scfs-submit -scfs <scfs-file> <SGE-option> -script <job-script> [<argument>]
ตัวอย่างการใช้งาน
# scfs-submit -scfs /scfs/nipat/case1/z1                    \
        -cwd -j y -N dx_case1                                \
        -script dx.sh /home/nipat/case1/net-files
```

โปรแกรมส่งงานจะทำงานตามขั้นตอนดังนี้

1) สอบถามชื่อเครื่องให้บริการไฟล์ที่ทำหน้าที่เก็บไฟล์ที่ต้องการใช้จาก SCFS ซึ่งข้อมูลที่ได้รับมาจาก SCFS จะเป็นรายชื่อของเครื่องที่มีไฟล์เก็บอยู่ (ทั้งไฟล์ต้นฉบับและไฟล์ฉบับที่ซ้ำไว้)
2) นำรายชื่อเครื่องที่ได้รับจาก SCFS ไปสอบถาม SGE ว่าบนเครื่องให้บริการไฟล์เหล่านี้มีคิว (Queue) ใดที่สามารถใช้งานได้บ้าง
3) นำข้อมูลคิวที่ได้จาก SGE, ข้อมูลตัวเลือกของ SGE, สคริปต์การทำงาน, และอาร์กิวเมนต์ของสคริปต์การทำงาน มาสร้างเป็นไฟล์อธิบายข้อกำหนดงาน (Job Specification File)
4) ส่งไฟล์อธิบายข้อกำหนดงานให้ SGE จัดการ

3.10 สรุป

ในบทนี้ได้กล่าวถึงการออกแบบและการพัฒนาระบบไฟล์คลัสเตอร์ ซึ่งมีส่วนประกอบหลักที่สำคัญได้แก่ ส่วนจัดการข้อมูลเมตะดาต้า, ส่วนจัดการไฟล์, ส่วนเครือข่ายและการจัดการข้อความที่รับส่งระหว่างเครื่อง, ส่วนอินเตอร์เฟสสำหรับเรียกใช้งานระบบ, ส่วนล็อก (Log), และส่วนโปรแกรมส่งงาน เนื่องจากระบบไฟล์คลัสเตอร์นี้มีวัตถุประสงค์เพื่อสนับสนุนการทำงานแบบเข้าถึงไฟล์แบบท้องถิ่น โดยวิธีการหลักๆ ที่ใช้เพื่อการทำงานตามวัตถุประสงค์นี้คือการเก็บไฟล์ไว้บนเครื่องคำนวณ, การเก็บและจัดการข้อมูลตำแหน่งของไฟล์ไว้ในเมตะดาต้า, และโปรแกรมส่งงานที่สามารถสอบถามข้อมูลตำแหน่งไฟล์และบอกกับโปรแกรมจัดลำดับงานให้ส่งงานไปยังเครื่องคำนวณที่มีไฟล์ที่ต้องการใช้งานเก็บอยู่ได้

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 4

การจัดการไฟล์ในระดับองค์กรเสมือน

ในบทนี้นำเสนอการออกแบบและการพัฒนาระบบมิดเดิลแวร์ที่ทำหน้าที่เป็นตัวกลางระหว่างผู้ใช้งานในองค์กรเสมือนกับระบบไฟล์คลัสเตอร์บนเครื่องคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือน วัตถุประสงค์ของมิดเดิลแวร์นี้คือเพื่อสนับสนุนการส่งงานไปประมวลผลที่เครื่องคลัสเตอร์คอมพิวเตอร์ที่มีข้อมูลเก็บอยู่ และการเพิ่มความสะดวกในการจัดการไฟล์ในระบบไฟล์คลัสเตอร์ของผู้ใช้งานในองค์กรเสมือน ในงานวิจัยนี้ได้สร้างระบบต้นแบบเพื่อใช้ในการทดลองโดยตั้งชื่อว่า “Cooperative Cluster Storage System” (CCSS) โดยมีโครงสร้างและหลักการทำงานดังนี้

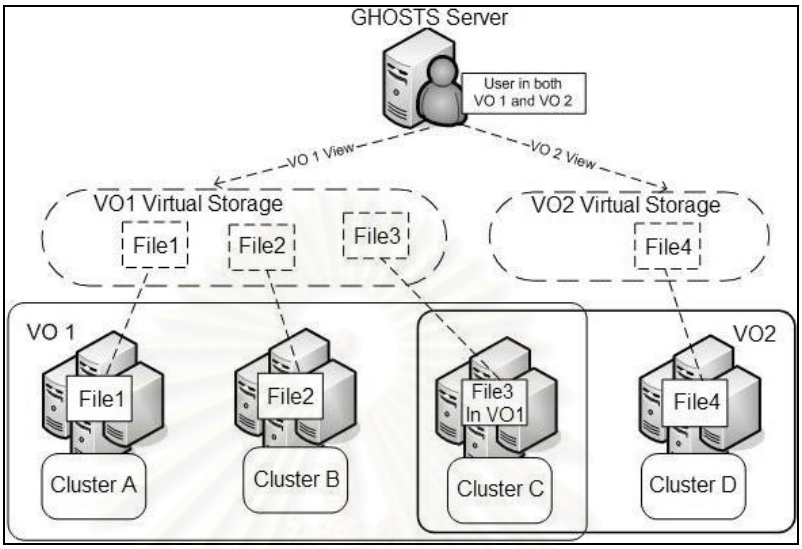
4.1 ความต้องการของระบบ

1) เก็บไฟล์ไว้ที่เครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์
2) สนับสนุนการทำงานของตัวจัดลำดับงาน โดยเมื่อส่งงานจากองค์กรเสมือนเข้าไปยังคลัสเตอร์คอมพิวเตอร์ ตัวจัดลำดับงานในคลัสเตอร์คอมพิวเตอร์สามารถส่งงานต่อไปยังเครื่องคำนวณที่มีไฟล์ที่ต้องการเก็บอยู่ได้
3) ผู้ใช้ที่อยู่ในองค์กรเสมือนสามารถสอบถามข้อมูลของไฟล์และจัดการไฟล์ที่อยู่ในคลัสเตอร์คอมพิวเตอร์ได้
4) การใช้งานสามารถทำได้ผ่านทางกริดพอร์ทอล

4.2 สถาปัตยกรรม

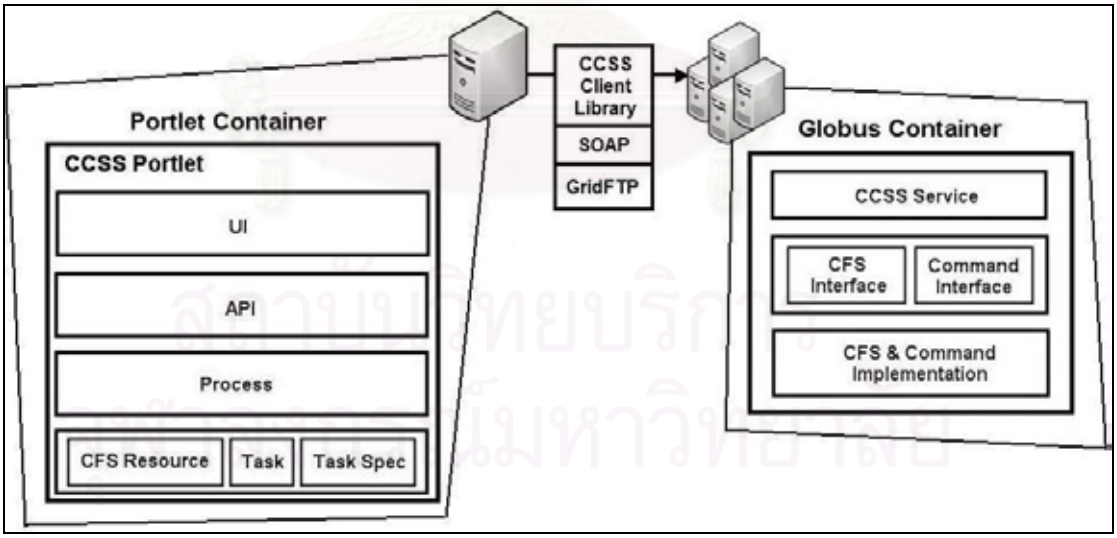
เนื่องจากการคำนวณภายในระบบกริดในปัจจุบันจะเกิดขึ้นที่เครื่องคลัสเตอร์คอมพิวเตอร์ ดังนั้นเพื่อให้การคำนวณสามารถเข้าถึงไฟล์แบบท้องถิ่นได้ ไฟล์จึงต้องเก็บอยู่ที่เครื่องคลัสเตอร์คอมพิวเตอร์ การออกแบบ CCSS ได้ใช้หลักการเดียวกับการออกแบบระบบไฟล์คลัสเตอร์ กล่าวคือ CCSS จะรวบรวมที่เก็บข้อมูลจากเครื่องคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือนหลายๆ คลัสเตอร์เพื่อสร้างเป็นที่เก็บข้อมูลเสมือนขนาดใหญ่ ผู้ใช้แต่ละคนจะมีที่เก็บข้อมูลเสมือนสำหรับระบบไฟล์คลัสเตอร์หนึ่งเท่ากับจำนวนองค์กรเสมือนที่ผู้ใช้คนนั้นเป็นสมาชิกอยู่ แต่จะสามารถใช้งานได้เฉพาะที่เก็บข้อมูลเสมือนขององค์กรเสมือนที่กำลังทำงานอยู่เท่านั้น เช่นในรูปที่ 4.1 ผู้ใช้เป็นสมาชิกขององค์กรเสมือนสององค์กรคือ VO1 และ VO2 ดังนั้นผู้ใช้จะมีที่เก็บข้อมูลเสมือนอยู่สองอันคือสำหรับ VO1 และ VO2 อย่างละอัน โดยที่เก็บข้อมูลที่อยู่ใน VO1 จะประกอบด้วยที่เก็บข้อมูลบนเครื่องคลัสเตอร์คอมพิวเตอร์ A, B, และ C ส่วนที่เก็บข้อมูลที่อยู่ใน VO2 จะประกอบด้วยที่เก็บข้อมูลบนเครื่องคลัสเตอร์คอมพิวเตอร์ C และ D ซึ่งในการใช้งานจริงๆ หากผู้ใช้กำลังใช้งาน VO2 ผู้ใช้ก็จะสามารถเข้าถึงไฟล์ที่

เก็บอยู่ในคลัสเตอร์คอมพิวเตอร์ C และ D ได้ แต่ในกรณีของคลัสเตอร์คอมพิวเตอร์ C นั้น ผู้ใช้จะไม่สามารถเข้าถึง File3 ได้ เนื่องจาก File3 ไม่ได้เก็บอยู่ในส่วนของ VO2



รูปที่ 4.1: แสดงที่เก็บข้อมูลเสมือนของผู้ใช้ที่เป็นสมาชิกในสององค์กรเสมือน

ส่วนอินเตอร์เฟสสำหรับให้ผู้ใช้เรียกใช้งานของ CCSS ติดตั้งอยู่ภายในโปรแกรมประเภทกริดพอร์ทอล (Grid Portal) [1] ทำให้การใช้งานสามารถทำได้ผ่านทางโปรแกรมประเภทเว็บบราวเซอร์แทนการล็อกอินเข้าไปจัดการที่คลัสเตอร์คอมพิวเตอร์ซึ่งช่วยให้การจัดการไฟล์สามารถทำได้สะดวกขึ้น



รูปที่ 4.2: แสดงสถาปัตยกรรมของ CCSS บนส่วนใช้บริการและส่วนให้บริการ

CCSS ใช้หลักการ 'Client-Server' ในการออกแบบ ระบบจะประกอบด้วยส่วนประกอบสองส่วนคือส่วนใช้บริการ (Client) ซึ่งติดตั้งอยู่บนเครื่องให้บริการองค์กรเสมือนและส่วนให้บริการ (Server) ซึ่งติดตั้งอยู่บนเครื่องคลัสเตอร์คอมพิวเตอร์ ระบบทั้งสองส่วนพัฒนาขึ้นด้วยภาษาจาวาโดยใช้ไลบรารี

ของ Globus [28, 36] ช่วยในการพัฒนา เครื่องคอมพิวเตอร์ในระบบติดต่อกันโดยใช้มาตรฐานของ Globus [37]

4.3 ระบบส่วนใช้บริการ

ระบบส่วนนี้ทำหน้าที่ติดต่อกับผู้ใช้ในองค์กรเสมือนเพื่อเรียกใช้บริการการจัดการไฟล์ของส่วนให้บริการบนเครื่องคลัสเตอร์คอมพิวเตอร์ (ในหัวข้อถัดไป) ส่วนใช้บริการจะติดตั้งอยู่บนเครื่องให้บริการการจัดการองค์กรเสมือน ซึ่งในงานวิจัยนี้ใช้ระบบให้บริการกริด (GHOSTS: Grid Hosting System) [30] ช่วยในการจัดการเรื่องบัญชีผู้ใช้และทรัพยากรขององค์กรเสมือน ทั้ง GHOSTS และระบบส่วนใช้บริการของ CCSS สร้างขึ้นอยู่บนฐานของโครงการ GridSphere และ GridPortlets [18] ซึ่งเป็นระบบที่ให้บริการเกี่ยวกับการใช้งานกริดพื้นฐาน เช่น การจัดการทรัพยากร, การจัดการงาน, และการจัดการไฟล์

ฟังก์ชันที่ GHOSTS ได้เพิ่มขึ้นมาจากฟังก์ชันของ GridPortlets คือฟังก์ชันที่เกี่ยวข้องกับการสร้าง การจัดการ และการใช้งานองค์กรเสมือน ส่วนฟังก์ชันที่ระบบส่วนใช้บริการของ CCSS เพิ่มขึ้นมาคือฟังก์ชันที่เกี่ยวข้องกับการจัดการไฟล์ในระบบไฟล์คลัสเตอร์บนเครื่องคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือน และในการทำงาน ส่วนใช้บริการของ CCSS จะติดต่อกับ GHOSTS ในเรื่องที่เกี่ยวข้องกับผู้ใช้ ทรัพยากร และสิทธิ์การใช้งานทรัพยากรของผู้ใช้ในองค์กรเสมือน

ส่วนใช้บริการสร้างขึ้นเป็นโปรแกรมประยุกต์ประเภทพอร์ทเลท โดยติดตั้งอยู่ภายในโปรแกรมที่บรรจุพอร์ทเลท (ซึ่งในงานวิจัยนี้คือ GridSphere) สถาปัตยกรรมของส่วนใช้บริการมีลักษณะเป็นระดับดังที่แสดงในรูปซ้ายของรูปที่ 4.2 โดยมีส่วนประกอบดังนี้

4.3.1 ส่วนทรัพยากร

โปรแกรมส่วนนี้เป็น*วัตถุ*ที่แทนทรัพยากรที่ใช้ในระบบ ระบบส่วนนี้ได้ออกแบบและพัฒนาตามรูปแบบการจัดการทรัพยากรของโครงการ GridPortlets โดย*วัตถุ*ชนิดหลักๆ ที่ใช้งานในระบบได้แก่

- *CFSResource* ใช้แทนหนึ่งระบบไฟล์คลัสเตอร์ที่ติดตั้งอยู่บนเครื่องคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือน
- *Task* เป็นงานที่มีให้บริการโดยพอร์ทเลทการจัดการไฟล์ของ CCSS ซึ่ง *Task* จะประกอบด้วย*วัตถุ*ลูกอีกหลายชนิดตามชนิดของงานที่มีให้บริการเช่น *MoveTask*, *CopyTask*, *DeleteTask*, *RenameTask* เป็นต้น ข้อมูลที่เก็บอยู่ใน *Task* ได้แก่ข้อมูลผู้ใช้, ข้อมูลเครื่องคลัสเตอร์คอมพิวเตอร์ที่เกี่ยวข้องกับงาน, และ*TaskSpec* เป็นต้น
- *TaskSpec* ใช้เก็บข้อมูลรายละเอียดของงานแต่ละงาน *TaskSpec* จะประกอบด้วย*วัตถุ*ลูกอีกหลายชนิดเช่นเดียวกับ *Task* ตัวอย่างข้อมูลที่เก็บได้แก่ ใน *RenameSpec* จะเก็บข้อมูลชื่อเดิมและชื่อใหม่ของไฟล์, ใน *DeleteSpec* จะเก็บข้อมูลพาทของไฟล์ที่จะลบ เป็นต้น

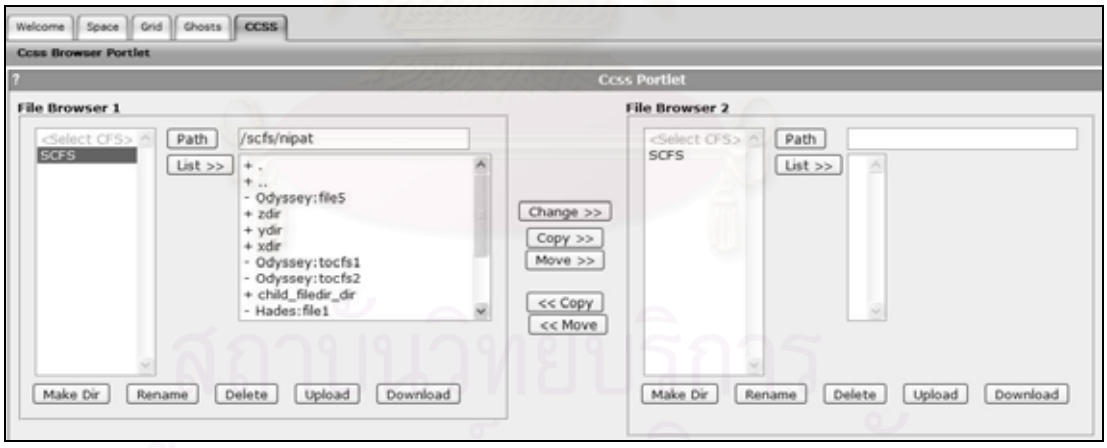
*วัตถุ*ส่วนใหญ่ที่ใช้ในระบบส่วนใช้บริการจะเกี่ยวข้องงาน ซึ่งในการออกแบบนี้ได้แบ่งงานออกเป็นสองประเภทตามระยะเวลาที่ใช้ในการทำงานคือ

1) งานประเภทที่ (น่าจะ) ใช้เวลาในการทำงานสั้น ซึ่งได้แก่ การแสดงรายการไฟล์, การสร้างไดเรกทอรี, การลบไฟล์, การเปลี่ยนชื่อไฟล์, การสอบถามข้อมูล, และการดาวน์โหลดไฟล์
2) งานประเภทที่ (น่าจะ) ใช้เวลาในการทำงานนาน ซึ่งได้แก่ การเคลื่อนย้ายไฟล์, การทำสำเนาไฟล์, และการอัพโหลดไฟล์

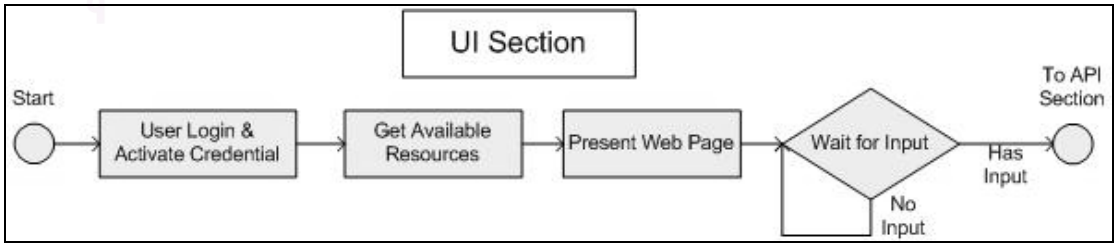
เนื่องจากส่วนทรัพยากรเป็น*วัตถุ*ที่ถูกใช้ในการทำงานจึงยกขึ้นมาอธิบายก่อน ในหัวข้อย่อยถัดไปจะเริ่มอธิบายตั้งแต่ส่วนที่ผู้ใช้เริ่มต้นทำงานไปกระทั่งถึงส่วนที่ทำหน้าที่ติดต่อกับระบบส่วนให้บริการบนเครื่องคลัสเตอร์คอมพิวเตอร์ตามลำดับ

4.3.2 อินเตอร์เฟสสำหรับผู้ใช้

หน้าที่ของส่วนนี้คือการติดต่อกับผู้ใช้เพื่อรับข้อมูลอินพุต, ตรวจสอบความถูกต้องเบื้องต้นของข้อมูลอินพุต และการแสดงผลข้อมูลเอาท์พุต ซึ่งการแสดงผลนี้ทำผ่านทางโปรแกรมประยุกต์ประเภทเว็บบราวเซอร์ ส่วนแสดงผลสร้างขึ้นโดยใช้เทคโนโลยี Java Server Page (JSP) และใช้โครงสร้างโปรแกรมประเภท Portlet และ UI ของ GridSphere ช่วยในการออกแบบและการพัฒนาโปรแกรม หน้าแสดงผลของ CCSS จะเป็นพอร์ทเลทอันหนึ่งที่ผนวกเข้ากับกริดพอร์ทอลของ GHOSTS



รูปที่ 4.3: ตัวอย่างหน้าแสดงผลของพอร์ทเลทของ CCSS



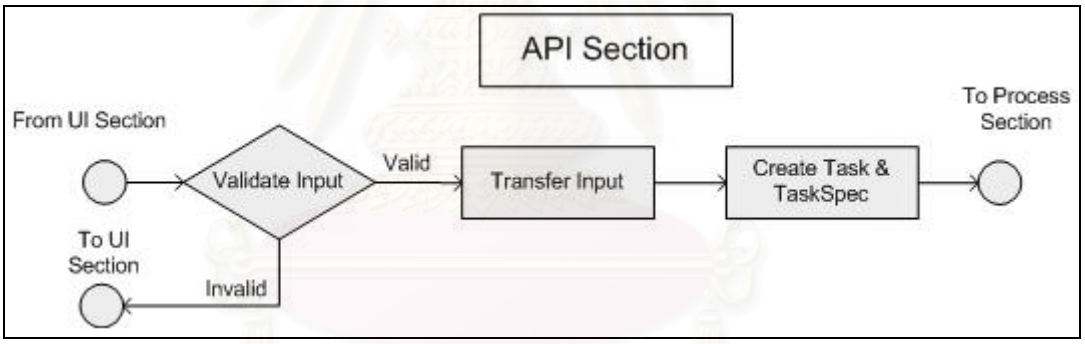
รูปที่ 4.4: แสดงขั้นตอนการทำงานของส่วนอินเตอร์เฟสผู้ใช้

ขั้นตอนการทำงานของส่วนอินเตอร์เฟสผู้ใช้เป็นไปตามที่แสดงในรูปที่ 4.4 คือ

1) ผู้ใช้ล็อกอินเข้ามาใช้งานกริดพอร์ทอลของ GHOSTS และเลือกว่าจะใช้ใบรับรองดิจิทัลใบใด ซึ่งใบรับรองดิจิทัลนี้จะเป็นตัวกำหนดองค์กรเสมือนที่ผู้ใช้กำลังจะเข้าไปใช้งาน
2) เมื่อผู้ใช้เลือกใช้งานพอร์ทเลทของ CCSS ส่วนอินเตอร์เฟสสำหรับผู้ใช้จะสอบถามจาก GHOSTS ว่าผู้ใช้มีสิทธิ์ใช้งานทรัพยากรประเภทคลัสเตอร์คอมพิวเตอร์ขององค์กรเสมือนเครื่องใดบ้าง
3) นำข้อมูลระบบไฟล์คลัสเตอร์บนทรัพยากรที่ได้รับจาก GHOSTS มาแสดงผลบนหน้าเว็บ และรอให้ผู้ใช้เรียกฟังก์ชันการทำงาน เมื่อได้รับคำสั่งจากผู้ใช้ ก็จะส่งต่อการทำงานไปยังส่วน API

4.3.3 API

เป็นส่วนอินเตอร์เฟสภาษาจาวาสำหรับให้โปรแกรมอื่นเรียกคำสั่งการทำงานของ CCSS ซึ่งคำสั่งที่มีให้บริการหลักๆ จะเป็นคำสั่งที่แสดงในหน้าแสดงผลหลักตามรูปที่ 4.3 นอกจากนั้นก็จะมีคำสั่งที่ใช้ในการสอบถามข้อมูลไฟล์และข้อมูลของระบบไฟล์คลัสเตอร์ สำหรับบางคำสั่งจะมีหลายฟังก์ชันสำหรับรับอินพุตหลายๆ รูปแบบเพื่อเพิ่มความสะดวกในการเรียกใช้งาน คำสั่งที่มีให้เรียกใช้งานใน API แสดงในภาคผนวก ค.



รูปที่ 4.5: แสดงขั้นตอนการทำงานของส่วน API

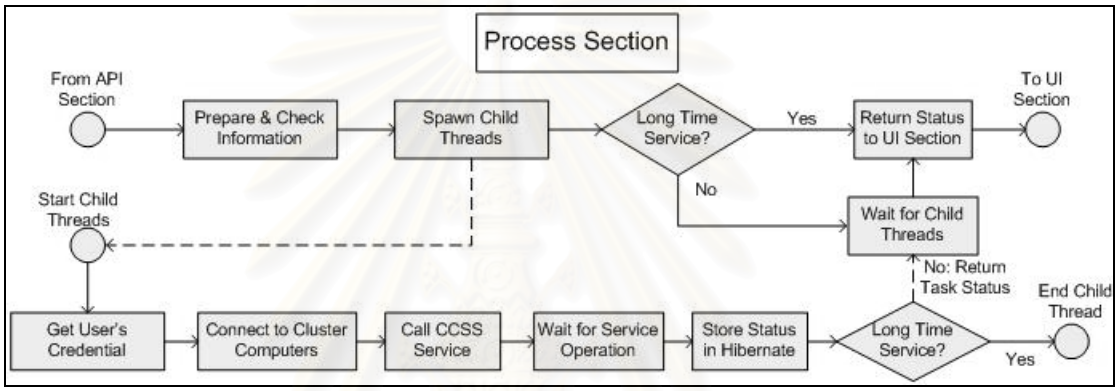
ขั้นตอนการทำงานของส่วน API เป็นไปตามที่แสดงในรูปที่ 4.5 คือการทำงานจะเริ่มต้นจาก

1) การตรวจสอบความถูกต้องของข้อมูลอินพุต หากมีข้อผิดพลาดก็จะแจ้งกลับไปที่ส่วนอินเตอร์เฟสสำหรับผู้ใช้เพื่อให้ผู้ใช้ทราบข้อผิดพลาดทันที
2) เนื่องจากฟังก์ชันใน API สามารถรับอินพุตได้หลายรูปแบบ ดังนั้นจึงจำเป็นต้องมีการแปลงข้อมูลอินพุตที่ได้รับให้อยู่ในรูปแบบเดียวเพื่อที่ระบบจะสามารถนำไปใช้งานต่อได้สะดวก
3) นำข้อมูลทั้งหมดมาสร้างเป็น *Task* และ *TaskSpec* จากนั้นจึงส่งต่อการทำงานไปให้ส่วนโปรเซส

4.3.4 โปรเซส

ส่วนโปรเซสทำหน้าที่ในการติดต่อกับส่วนให้บริการบนเครื่องคลัสเตอร์คอมพิวเตอร์ โดยในการออกแบบ ได้ออกแบบให้ระบบส่วนนี้ทำงานในลักษณะหลายเทรด (Multi Thread) คือมีเทรดแม่หนึ่ง

เทรดและเทรดลูกหลายเทรด เทรดแม่ซึ่งเป็นเทรดหลักจะทำหน้าที่ติดต่อกับส่วนประกอบอื่นๆ ภายในระบบ อีกหน้าที่หนึ่งของเทรดแม่คือการสร้างเทรดลูกเพื่อทำหน้าที่ติดต่อเรียกใช้บริการบนเครื่องคลัสเตอร์คอมพิวเตอร์ ซึ่งข้อมูลที่จะใช้ในการสร้างการเชื่อมต่อและเรียกใช้บริการที่คลัสเตอร์คอมพิวเตอร์จะมาจาก *Task* และ *TaskSpec* ที่ได้รับมาจากเทรดแม่ เทรดแม่จะสร้างเทรดลูกตามจำนวนเครื่องคลัสเตอร์คอมพิวเตอร์ที่ต้องการใช้งาน เช่นในคำสั่งแสดงรายการไฟล์ที่อยู่บนเครื่องคลัสเตอร์คอมพิวเตอร์จำนวน 3 เครื่อง เทรดแม่ก็จะสร้างเทรดลูกขึ้นมา 3 เทรด และเทรดลูกแต่ละเทรดก็จะรับผิดชอบการติดต่อกับคลัสเตอร์คอมพิวเตอร์ 1 เครื่อง เทรดลูกแต่ละเทรดจะทำงานพร้อมๆ กัน และการทำงานก็จะเป็นอิสระต่อกัน



รูปที่ 4.6: แสดงขั้นตอนการทำงานของส่วนโปรเซส

ขั้นตอนการทำงานของส่วนโปรเซสเป็นไปตามที่แสดงในรูปที่ 4.6 คือสำหรับการทำงานของเทรดแม่จะเริ่มต้นจาก

1) เตรียมและตรวจสอบข้อมูลที่จะใช้ในการสร้างการเชื่อมต่อและเรียกใช้บริการ
2) สร้างเทรดลูกตามจำนวนเครื่องคลัสเตอร์คอมพิวเตอร์ที่ต้องการติดต่อ
3) หากงานที่ทำเป็นประเภทใช้เวลานาน เทรดแม่ก็จะกลับไปทำงานต่อที่ส่วนอินเตอร์เฟสสำหรับผู้ใช้ทันที โดยแสดงผลให้ผู้ใช้ทราบว่าคำสั่งที่ผ่านมาได้ถูกส่งไปทำที่เครื่องคลัสเตอร์คอมพิวเตอร์แล้ว แต่หากเป็นงานที่ใช้เวลาสั้น เทรดแม่จะรอให้เทรดลูกทุกเทรดทำงานเสร็จก่อน แล้วค่อยกลับไปทำงานที่ส่วนอินเตอร์เฟสผู้ใช้ โดยนำผลลัพธ์การทำงานจากเทรดลูกไปแสดงให้ผู้ทราบ

สำหรับเทรดลูก การทำงานจะเริ่มต้นจาก

1) ร้องขอใบรับรองดิจิทัลของผู้ใช้จาก GHOSTS
2) สร้างตัวเชื่อมต่อไปยังเครื่องคลัสเตอร์คอมพิวเตอร์ที่เทรดนี้รับผิดชอบ หากการเชื่อมต่อสำเร็จก็แสดงว่าใบรับรองดิจิทัลของผู้ใช้ผ่านการตรวจสอบโดยส่วนให้บริการ
3) ร้องขอการทำงานของส่วนให้บริการโดยเรียกผ่านทางไลบรารีผู้ใช้บริการของ CCSS (จะกล่าวถึงในหัวข้อ 4.4.3) แล้วรอจนกว่าฝั่งโปรเซสผู้ให้บริการจะทำงานเสร็จ

4) เมื่อโปรเซสผู้ให้บริการทำงานเสร็จแล้ว เทรดลูกจะนำสถานะของงานไปเก็บไว้ใน Hibernate [38] ซึ่งเทรดแม่หรือเทรดอื่นๆ ที่มีสิทธิ์สามารถดึงข้อมูลสถานะของงานจาก Hibernate เพื่อนำไปใช้งานในภายหลังได้
5) หากเป็นงานที่ใช้เวลาสั้น เทรดลูกจะส่งผลลัพธ์การทำงานและสถานะของงานกลับไปให้เทรดแม่แล้วจึงจบการทำงานของตัวเอง แต่หากเป็นงานที่ใช้เวลานาน เทรดแม่ไม่ได้รออยู่ เทรดลูกก็จะจบการทำงานของตัวเองไป

4.4 ระบบส่วนให้บริการ

ระบบส่วนนี้สร้างขึ้นเป็นโปรแกรมประเภทกริดเซอร์วิส [38] ที่ให้บริการการจัดการไฟล์ภายในระบบไฟล์คลัสเตอร์ ส่วนให้บริการนี้จะติดตั้งอยู่ภายในโปรแกรมที่บรรจุของ Globus (Globus Container) บนเครื่องคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือนทุกเครื่อง สถาปัตยกรรมของส่วนให้บริการมีลักษณะเป็นระดับดังแสดงในรูปขวาของรูปที่ 4.2 โดยมีส่วนประกอบได้แก่ ส่วนบริการของ CCSS, ส่วนอินเตอร์เฟสสำหรับระบบไฟล์คลัสเตอร์และคำสั่ง, และส่วนขยายการทำงานของส่วนอินเตอร์เฟส นอกจากนี้ผู้วิจัยยังได้ออกแบบและพัฒนาส่วนไลบรารีผู้ใช้บริการขึ้นมาเพื่อเพิ่มความสะดวกในการเรียกใช้บริการ

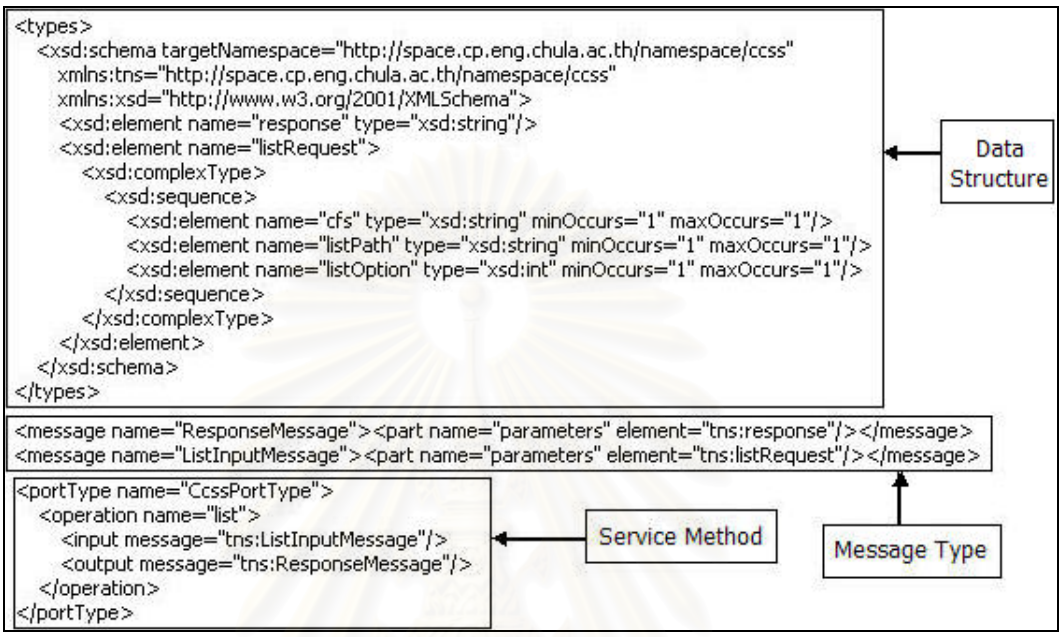
4.4.1 บริการของ CCSS

ระบบส่วนนี้เป็นบริการที่เปิดให้ใช้งาน เมื่อมีคำร้องขอเรื่องการจัดการไฟล์ของ CCSS เข้ามาที่โปรแกรมที่บรรจุของ Globus คำร้องขอจะถูกส่งต่อมาที่ส่วนนี้เพื่อประมวลผล การพัฒนาส่วนบริการแบ่งออกเป็นสองส่วน คือ

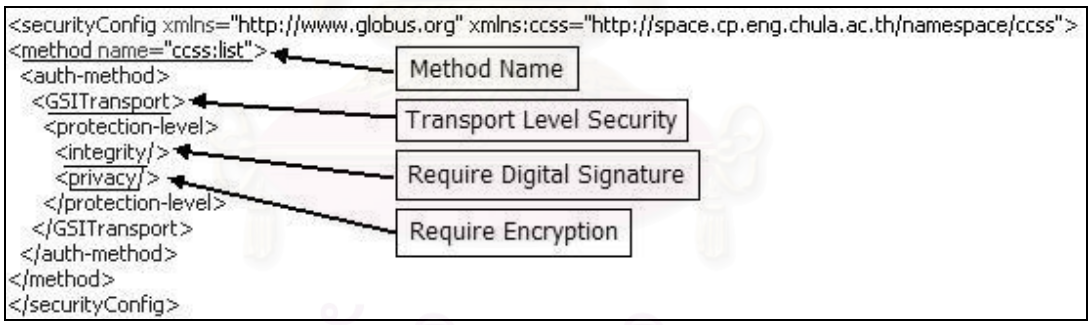
4.4.1.1 ไฟล์อธิบายบริการ

- ไฟล์อธิบายรายละเอียดของบริการ เป็นไฟล์ในรูปแบบ Web Services Description Language (WSDL) ในไฟล์นี้จะบอกว่าประเภทของข้อมูลที่ใช้ในส่วนบริการของ CCSS มีอะไรบ้าง, ข้อมูลแต่ละประเภทมีโครงสร้างอย่างไร, ในบริการมีฟังก์ชันอะไรที่เปิดให้ใช้งานบ้าง, แต่ละฟังก์ชันรับอินพุตเป็นข้อมูลประเภทอะไรและคืนค่าเอาท์พุตเป็นข้อมูลประเภทอะไร
- ไฟล์อธิบายนโยบายด้านความปลอดภัยของบริการ เป็นไฟล์รูปแบบ XML ซึ่งในบริการของ CCSS ได้ใช้ตามมาตรฐานของ Grid Security Infrastructure (GSI) [29] โดยกำหนดว่าผู้ใช้บริการจะต้องมีใบรับรองดิจิทัล และในการเรียกใช้งานแต่ละฟังก์ชันของบริการ ข้อมูลที่รับส่งระหว่างเครื่องจะถูกเข้ารหัสในระดับ Transport Layer และใช้วิธีการลายเซ็นดิจิทัล (Digital Signature) ในการยืนยันตัวตนของผู้ใช้บริการ

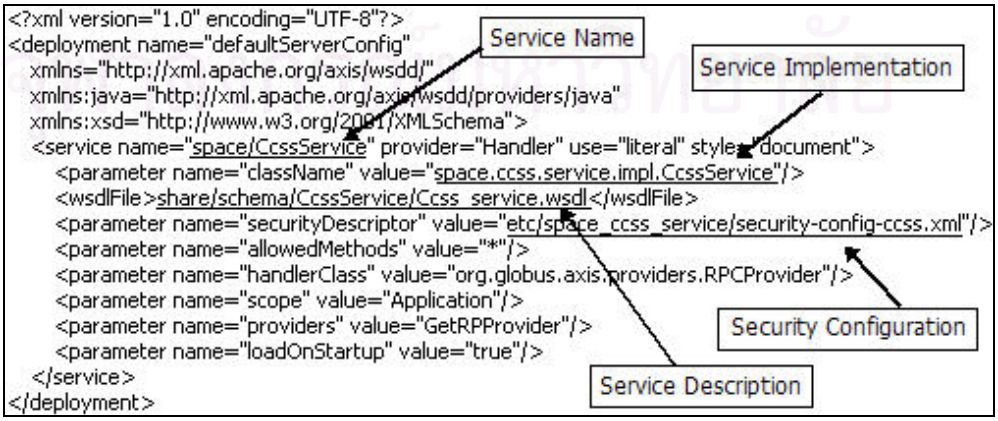
- ไฟล์ที่ใช้ในการติดตั้งบริการ เป็นไฟล์ในรูปแบบ Web Service Deployment Descriptor (WSDD) เป็นไฟล์ที่บอกโปรแกรมที่บรรจุของ Globus ว่าจะติดตั้งบริการนี้ไว้ที่ไหน ไฟล์ที่เกี่ยวข้องกับบริการนี้มีอะไรบ้างและอยู่ที่ไหน และจะจัดการกับบริการนี้อย่างไร



รูปที่ 4.7: แสดงตัวอย่างไฟล์อธิบายบริการ (WSDL)



รูปที่ 4.8: แสดงตัวอย่างไฟล์อธิบายนโยบายด้านความปลอดภัยของบริการ (XML)



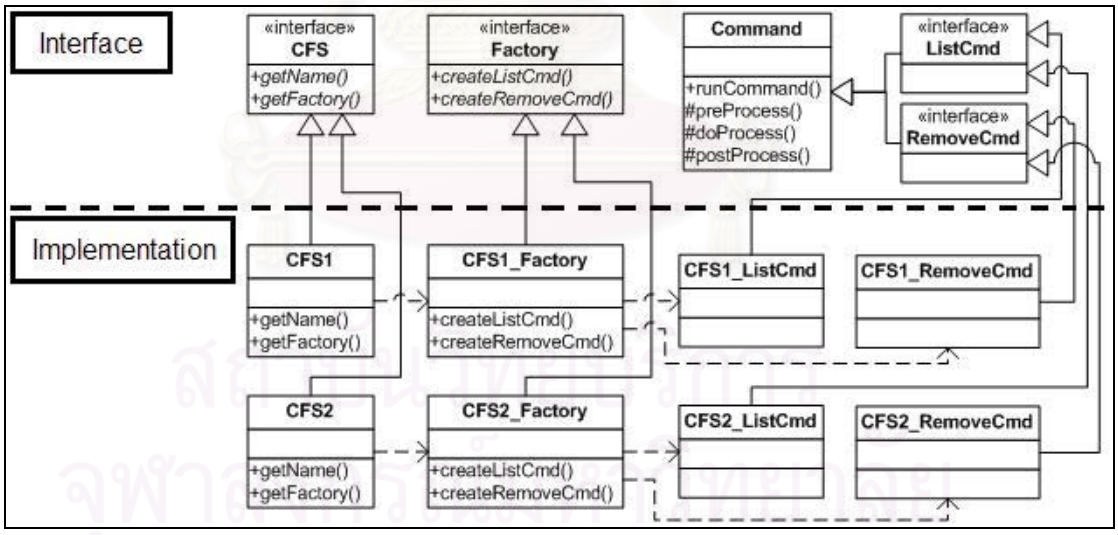
รูปที่ 4.9: แสดงตัวอย่างไฟล์อธิบายการติดตั้งบริการ (WSDD)

4.4.1.2 โปรแกรมระบุบริการ

1) โปรแกรมที่ได้จากการแปลไฟล์ WSDL ที่ใช้อธิบายรายละเอียดบริการของ CCSS ในการพัฒนาระบบ ได้ใช้โปรแกรมที่ได้จาก “The Globus Toolkit 4 Programmer’s Tutorial” [36] ในการแปลไฟล์ WSDL โปรแกรมกลุ่มนี้จะมีลักษณะเป็นสตับ (Stub) ที่ใช้เป็น*วัตถุ*สำหรับเรียกใช้โดยโปรแกรมที่เกี่ยวข้องกับบริการเช่น คำร้องขอ, โครงสร้างข้อมูลอินพุตของแต่ละฟังก์ชัน, โครงสร้างข้อมูลเอาท์พุตของแต่ละฟังก์ชัน, และบริการของ CCSS ที่เปิดให้ใช้
2) โปรแกรมส่วนระบุการทำงานของบริการ โปรแกรมส่วนนี้จะบอกว่าเมื่อมีคำร้องขอเข้ามา ระบบจะต้องทำงานอย่างไรเพื่อจัดการกับคำร้องขอนั้น

4.4.2 ส่วนอินเตอร์เฟสและส่วนขยายการทำงานของส่วนอินเตอร์เฟส

ส่วนอินเตอร์เฟสเป็นส่วนที่กำหนดว่าระบบไฟล์คลัสเตอร์ที่จะนำมาใช้ใน CCSS จะต้องมีฟังก์ชันอะไรบ้าง แต่ละฟังก์ชันทำงานอะไร รับอินพุตเป็นอะไร และคืนค่าเอาท์พุตเป็นอะไร และส่วนขยายการทำงานของอินเตอร์เฟสเป็นส่วนที่ระบุการทำงานในแต่ละฟังก์ชันของระบบไฟล์คลัสเตอร์ (ภาคผนวก ค. แสดงอินเตอร์เฟสหลักๆ ที่ระบบไฟล์คลัสเตอร์จะต้องระบุการทำงาน) ในโครงงานวิจัยนี้ได้สร้างส่วนขยายการทำงานของอินเตอร์เฟสสำหรับ SCFS ขึ้นมาเพื่อใช้ในการทดสอบระบบ โดยอินเตอร์เฟสที่สำคัญใน CCSS จะประกอบด้วย



รูปที่ 4.10: แสดงการออกแบบส่วนอินเตอร์เฟสและส่วนขยายการทำงานของอินเตอร์เฟส

4.4.2.1 อินเตอร์เฟสระบบไฟล์คลัสเตอร์

อินเตอร์เฟสหลักในส่วนนี้จะทำหน้าที่แทนระบบไฟล์คลัสเตอร์หนึ่งๆ ในอินเตอร์เฟสนี้จะมีฟังก์ชันที่จำเป็นที่ทุกระบบไฟล์คลัสเตอร์จะต้องมี เช่นการถามชื่อของระบบไฟล์คลัสเตอร์ และการถามขอบเขตชื่อที่ใช้ในระบบไฟล์ นอกจากนี้ยังมีอินเตอร์เฟสรองอื่นๆ ที่ทำหน้าที่เป็นคุณสมบัติเพิ่มเติมของ

ระบบไฟล์คลัสเตอร์ ระบบไฟล์คลัสเตอร์ที่จะใช้งาน CCSS จะต้องสร้าง*วัตถุ*ที่ขยายการทำงานของอินเตอร์เฟสหลักและลงทะเบียนระบบไฟล์เข้าสู่ส่วนให้บริการของ CCSS (หัวข้อ 4.4.2.4)

4.4.2.2 อินเตอร์เฟสสำหรับ*วัตถุ*คำสั่ง

อินเตอร์เฟสส่วนนี้จะเป็นคำสั่งที่เปิดให้ใช้งานใน CCSS ระบบส่วนนี้ใช้ Template Method Design Patten [32] ในการออกแบบ ส่วนที่เป็น Template Method จะบอกว่าในการทำงานแต่ละคำสั่งจะประกอบด้วยขั้นตอนอะไรบ้าง และแต่ละขั้นตอนจะเรียกฟังก์ชันใดในการทำงาน ซึ่งใน CCSS ขั้นตอนการทำงานจะประกอบด้วยสามขั้นตอนหลักคือ

1) การเตรียมข้อมูลอินพุตที่ได้รับจากส่วนบริการ ขั้นตอนนี้จะตรวจสอบว่า อินพุตที่ผู้ใช้บริการส่งเข้ามามีครบตามที่ระบุไว้หรือไม่ และนำค่าอินพุตไปเก็บในตัวแปรที่ระบบไฟล์คลัสเตอร์จะสามารถนำไปเรียกใช้ได้สะดวก
2) การทำงานคำสั่ง
3) การเตรียมข้อมูลเอาท์พุตส่งคืนให้ส่วนบริการ ซึ่งข้อมูลนี้จะอยู่ในรูปแบบ XML

การทำงานในส่วนที่หนึ่งและส่วนที่สามเป็นส่วนที่มีการระบุการทำงานอยู่แล้วในโปรแกรมของ CCSS ส่วนในขั้นตอนที่สองจะเป็นอินเตอร์เฟสเพื่อให้แต่ละระบบไฟล์คลัสเตอร์นำไประบุการทำงานของตัวเอง

4.4.2.3 อินเตอร์เฟสการสร้าง*วัตถุ*คำสั่ง

อินเตอร์เฟสส่วนนี้ทำหน้าที่สร้าง*วัตถุ*คำสั่ง โปรแกรมส่วนนี้ใช้ Abstract Factory Design Pattern [32] ในการออกแบบ แต่ละระบบไฟล์คลัสเตอร์จะต้องขยายการทำงานของ*วัตถุ*ชนิดนี้เพื่อระบุวิธีการสร้าง*วัตถุ*คำสั่งของตัวเอง หากระบบไฟล์ใดไม่ต้องการเปิดให้บริการคำสั่งใดก็ไม่ต้องระบุวิธีการสร้าง*วัตถุ*คำสั่งของคำสั่งนั้น

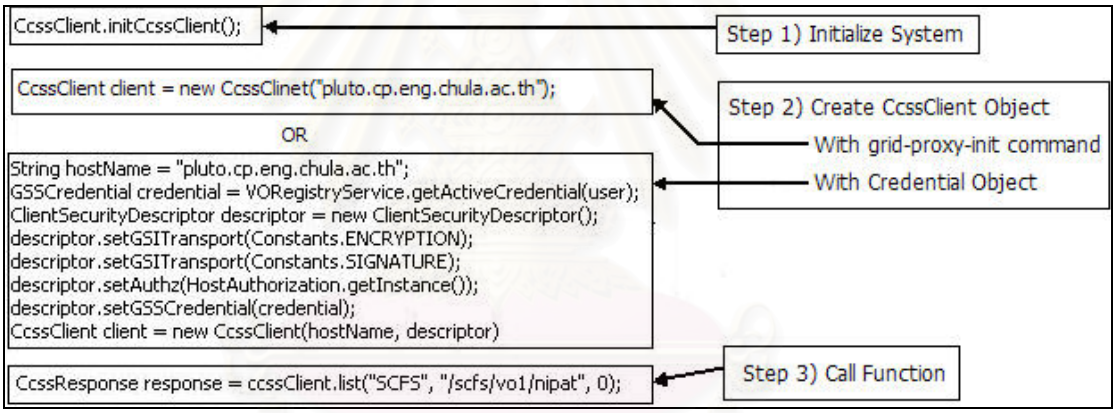
4.4.2.4 การลงทะเบียนระบบไฟล์คลัสเตอร์

การลงทะเบียนระบบไฟล์เป็นการทำให้ส่วนให้บริการของ CCSS รู้จักระบบไฟล์คลัสเตอร์นั้นและสามารถนำระบบไฟล์คลัสเตอร์นั้นมาใช้งานในส่วนให้บริการได้ การลงทะเบียนสามารถทำได้โดยการนำไฟล์ไลบรารีของระบบ (ไฟล์ .jar) ไปวางไว้ที่ไดเรกทอรีของส่วนให้บริการ CCSS แล้วเพิ่มข้อมูลของระบบไฟล์เข้าไปในไฟล์การตั้งค่าของ CCSS โดยข้อมูลที่จำเป็นต้องมีคือชื่อของระบบไฟล์ และชื่อของโปรแกรมที่ขยายการทำงานของอินเตอร์เฟสระบบไฟล์คลัสเตอร์ (ในหัวข้อ 4.4.2.1) นอกจากนี้ก็สามารถกำหนดค่าพารามิเตอร์เริ่มต้นที่จะใช้ในระบบไฟล์ของตัวเองได้ โดยการกำหนดว่าพารามิเตอร์นี้ชื่อว่าอะไร มีชนิดเป็นอะไร และมีค่าเป็นอะไร

4.4.3 ไลบรารีผู้ใช้บริการ

ตามที่กล่าวไว้ใน [36] ว่าผู้ใช้บริการของเว็บเซอร์วิสจะต้องเป็นโปรแกรมคอมพิวเตอร์ (ซึ่งถูกใช้โดยคนอีกทีหนึ่ง) ดังนั้น ในการเรียกใช้งานส่วนให้บริการของ CCSS นี้ ผู้ใช้จะต้องเขียนโปรแกรมขึ้นมาเพื่อติดต่อกับระบบ และเพื่อให้การเขียนโปรแกรมสามารถทำได้สะดวก โครงงานวิจัยนี้จึงได้สร้างไลบรารีผู้ใช้บริการขึ้นมา (API ที่มีให้ใช้งานของไลบรารีผู้ใช้บริการแสดงไว้ในภาคผนวก ค.) โดยรวบรวมขั้นตอนต่างๆ ที่จำเป็นเข้าไว้ในไลบรารีนี้ได้แก่

- การตั้งค่าระบบความปลอดภัยของการรับส่งข้อมูลให้ตรงตามที่ระบุไว้ในส่วนบริการ หากตั้งค่าส่วนนี้ผิด จะไม่สามารถสร้างการเชื่อมต่อและเรียกใช้งานส่วนให้บริการได้
- การเรียกใช้งานไลบรารีของ Globus และสตับของโปรแกรมส่วนบริการ ซึ่งเป็นการเขียนโปรแกรมในรูปแบบเฉพาะอย่าง นักพัฒนาระบบบางคนอาจไม่คุ้นเคย
- การแปลผลลัพธ์ที่ได้จากการทำงานที่อยู่ในรูปแบบ XML ให้กลายเป็น*วัตถุ*ที่จะสามารถนำไปใช้งานได้สะดวก



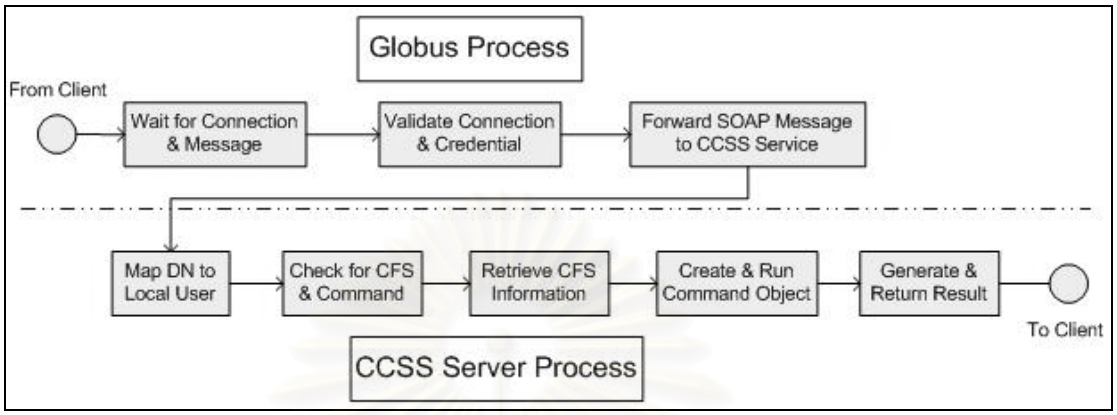
รูปที่ 4.11: แสดงตัวอย่างการเรียกใช้งานไลบรารีผู้ใช้บริการ

4.4.4 ขั้นตอนการทำงาน

ในการเรียกใช้งานส่วนให้บริการของ CCSS ผู้เรียกใช้งานจะต้องมีใบรับรองดิจิทัล และเปิดการใช้งานโปรแกรมประเภทกริดพรอกซี (Proxy) เรียบร้อยแล้ว ซึ่งในขั้นตอนนี้อาจจะมาจากการใช้งานส่วนใช้บริการบนเครื่องให้บริการ GHOSTS หรือใช้คำสั่ง ‘grid-proxy-init’ บนเครื่องคอมพิวเตอร์ในองค์กรเสมือนก็ได้ เมื่อมีการสร้างการเชื่อมต่อมาที่ส่วนให้บริการนี้ โปรแกรมที่บรรจุของ Globus จะใช้ใบรับรองดิจิทัลตรวจสอบว่าผู้ใช้มีสิทธิ์ใช้งานบริการบนเครื่องหรือไม่ หากการตรวจสอบนี้ผ่านก็จะสามารถใช้บริการได้

การใช้บริการจะเริ่มต้นจากโปรแกรมฝั่งผู้ใช้ส่งคำร้องขอเข้ามา ซึ่งการส่งคำร้องขอนี้จะใช้โปรโตคอล SOAP ซึ่งเป็นโปรโตคอลมาตรฐานของเว็บเซอร์วิสในการส่ง โดยข้อความ SOAP ที่ส่งมานี้

จะต้องเป็นไปตามนโยบายความปลอดภัยที่ตั้งไว้ในส่วนให้บริการคือข้อมูลที่ส่งจะต้องมีการเข้ารหัสในระดับ Transport Layer และมีการใช้ลายเซ็นดิจิทัล



รูปที่ 4.12: แสดงขั้นตอนการทำงานของระบบส่วนให้บริการ

เมื่อมีคำร้องขอบริการของ CCSS เข้ามาที่โปรแกรมที่บรรจุของ Globus คำร้องขอนั้นจะถูกส่งต่อมาที่ส่วนบริการของ CCSS ส่วนบริการจะตรวจสอบจากใบรับรองดิจิทัลของผู้ใช้เพื่อแมพเข้ากับรายชื่อผู้ใช้บนเครื่องคลัสเตอร์คอมพิวเตอร์ แล้วจึงตรวจสอบว่าคำร้องขอนั้นเรียกใช้งานระบบไฟล์คลัสเตอร์ใดและเป็นคำสั่งอะไร จากนั้นจึงดึงข้อมูลของระบบไฟล์คลัสเตอร์ขึ้นมาเพื่อใช้สร้าง*วัตถุ*คำสั่งของระบบไฟล์นั้น พารามิเตอร์ที่ผู้ใช้ส่งเข้ามาจะถูกส่งต่อให้กับ*วัตถุ*คำสั่งเพื่อนำไปใช้ในการทำงาน ซึ่งการทำงานของ*วัตถุ*คำสั่งจะเป็นไปตามขั้นตอนสามขั้นตอนที่ระบุไว้ใน Template Method (ในหัวข้อ 4.4.2.2) และเมื่อ*วัตถุ*คำสั่งทำงานเสร็จ ส่วนบริการก็จะสร้างข้อมูลผลลัพธ์ในรูปแบบ XML แล้วนำไปบรรจุไว้ในข้อความ SOAP เพื่อตอบกลับไปยังโปรแกรมผู้เรียกใช้บริการ

4.4.5 ข้อจำกัดของระบบส่วนให้บริการ

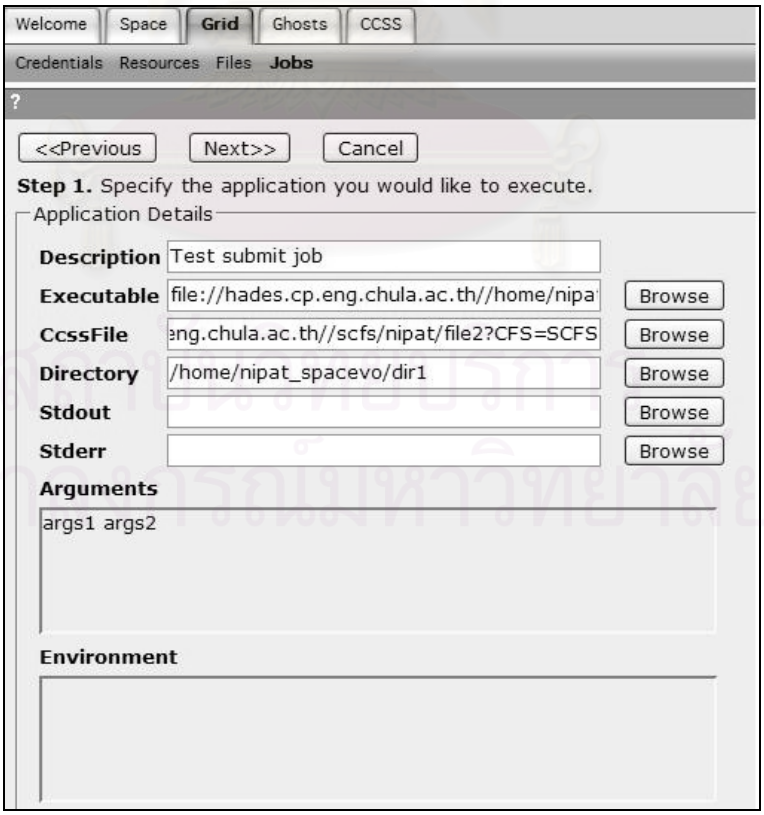
เพื่อลดความซับซ้อนในการออกแบบและการพัฒนาระบบในส่วนให้บริการ ดังนั้นจึงมีข้อจำกัดอยู่สองประการคือ (1) ระบบส่วนให้บริการของ CCSS จะรับผิดชอบเฉพาะการทำงานบนเครื่องคลัสเตอร์คอมพิวเตอร์ของตนเองเท่านั้น โดยจะไม่รับรู้ถึงการทำงานบนเครื่องอื่นเลย และ (2) คำร้องขอหนึ่งคำร้องจะสามารถเรียกใช้งานระบบไฟล์คลัสเตอร์ได้เพียงหนึ่งระบบไฟล์เท่านั้น

ด้วยข้อจำกัดทั้งสองประการจึงส่งผลให้ความซับซ้อนไปตกอยู่กับโปรแกรมส่วนใช้บริการในคำสั่งประเภทที่ต้องเคลื่อนย้ายไฟล์ระหว่างเครื่องคอมพิวเตอร์หรือระหว่างระบบไฟล์ กล่าวคือโปรแกรมส่วนผู้ใช้บริการจะต้องจัดการในขั้นตอนการเคลื่อนย้ายไฟล์เอง ซึ่งในส่วนใช้บริการของ CCSS ที่ติดตั้งใน GHOSTS ได้ใช้วิธีการส่งออกไฟล์จากระบบไฟล์คลัสเตอร์ไปยังที่เก็บข้อมูลชั่วคราวในระบบไฟล์ของลินุกซ์ก่อน หากจำเป็นต้องมีการเคลื่อนย้ายไฟล์ระหว่างเครื่องก็จะใช้โปรโตคอล GridFTP ในการเคลื่อนย้ายไฟล์ แล้วจึงลงทะเบียนไฟล์นั้นเข้าสู่ระบบไฟล์คลัสเตอร์ที่เครื่องปลายทาง

4.5 โปรแกรมผู้จัดการงาน

โปรแกรมผู้จัดการงานมีวัตถุประสงค์เพื่อให้ผู้ใช้งานในระบบกริดสามารถส่งงานไปประมวลผลที่เครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ที่มีไฟล์ที่ต้องการใช้เก็บอยู่ได้ ซึ่งโปรแกรมผู้จัดการงานนี้สร้างขึ้นด้วยภาษา Perl โดยสร้างเป็นมอดูลผู้จัดการงาน (Job Manager Module) และติดตั้งลงในระบบ Globus Resource Allocation Manager (GRAM) [29] โปรแกรมผู้จัดการงานของ CCSS จะขยายการทำงานของโปรแกรมผู้จัดการงานของ GRAM ในส่วนการติดต่อกับโปรแกรมจัดลำดับงานบนเครื่องคลัสเตอร์คอมพิวเตอร์ (ซึ่งในงานวิจัยนี้คือ SGE) หากงานที่ส่งเข้ามามีการกำหนดบริการเป็น jobmanager-ccss โปรแกรมผู้จัดการงานของ GRAM ก็จะส่งต่อการทำงานมาที่โปรแกรมผู้จัดการงานนี้

การทำงานของโปรแกรมผู้จัดการงานจะเริ่มต้นจากการตรวจสอบพารามิเตอร์ของงานที่เข้ามา ซึ่งโปรแกรมผู้จัดการงานได้เพิ่มคุณสมบัติชื่อ scfsfile ขึ้นมาเพื่อใช้ระบุชื่อไฟล์ที่เก็บอยู่ใน SCFS ที่ต้องการใช้ในงาน โปรแกรมจะสอบถาม SCFS ว่าไฟล์นี้เก็บอยู่บนเครื่องใดบ้าง แล้วจึงนำรายชื่อเครื่องไปถาม SGE ว่าเครื่องเหล่านี้มีคิว (Queue) ใดให้สามารถใช้งานได้บ้าง จากนั้นจึงนำข้อมูลที่ได้จากพารามิเตอร์ของงานและข้อมูลคิวมาสร้างเป็นไฟล์อธิบายข้อกำหนดงานตามรูปแบบของ SGE โดยในไฟล์นี้จะระบุว่างานจะต้องถูกส่งไปยังคิวใดของ SGE สุดท้ายจึงส่งไฟล์อธิบายข้อกำหนดงานนี้ให้ SGE ทำงานต่อ



รูปที่ 4.13: แสดงหน้าส่งงานของ CCSS

เนื่องจากโปรแกรมผู้จัดการงานจะต้องไปสอบถามข้อมูลไฟล์จาก SCFS ดังนั้นไฟล์ที่จะใช้ในการสอบถามจะต้องถูกลงทะเบียนเข้าไปเก็บไว้ใน SCFS ก่อนจึงจะสามารถเรียกใช้งานโปรแกรมผู้จัดการงานนี้ได้ การเรียกใช้งานโปรแกรมผู้จัดการงานนี้สามารถทำได้สองวิธี วิธีแรกคือการล็อกอินเข้าไปที่เครื่องคอมพิวเตอร์ที่สามารถใช้งานคลัสเตอร์คอมพิวเตอร์ที่เป็นทรัพยากรขององค์กรเสมือนได้ แล้วสั่งงานโดยใช้บรรทัดคำสั่งของ Globus ชื่อ ‘globus-job-submit’, ‘globus-job-run’, หรือ ‘globusrun’ โดยระบุบริการเป็น jobmanager-ccss ส่วนวิธีที่สองคือการส่งงานผ่านทางกริดพอร์ทอลของ GHOSTS โดยเลือกใช้บริการการส่งงานของ CCSS

ตัวอย่างคำสั่งที่ใช้สั่งงานผ่านทางบรรทัดคำสั่ง เช่น ‘globusrun -f test_tunami.rsl -r pluto.cp.eng.chula.ac.th/jobmanager-ccss’ โดยที่ ‘-f’ จะเป็นการกำหนดพาธของไฟล์อธิบายงาน (ไฟล์ RSL) ที่ต้องการจะทำ ส่วน ‘-r’ จะเป็นการกำหนดทรัพยากรของกริดที่ต้องการจะใช้ประมวลผล ‘jobmanager-ccss’ เป็นชื่อบริการของโปรแกรมผู้จัดการงานของ CCSS

```
&(rslSubstitution=(LOCAL_BASE_DIR      "/home/nipat_le/project/test/scenario"))
  (rslSubstitution=(BASE_DIR          "/home/nipat/project/test/scenario"))
  (rslSubstitution=( SCFS_FILE        "/scfs/nipat/file"))
  (rslSubstitution=(PROJ_NAME          "test_tunami_normal"))
  (rslSubstitution=(PROJ_TAR          "tunami.tar"))
  (rslSubstitution=(EXE_NAME          "test2.sh"))

  (directory=$(BASE_DIR))
  (scfsFile =$( SCFS_FILE))
  (executable=$(GLOBUSRUN_GASS_URL)$(LOCAL_BASE_DIR)/$(EXE_NAME))
  (arguments=$(PROJ_TAR)
       $(PROJ_NAME))
```

รูปที่ 4.14: แสดงตัวอย่างไฟล์ RSL ที่ใช้สั่งงานด้วยคำสั่ง ‘globusrun’

4.6 สรุป

ในบทนี้ได้กล่าวถึงการออกแบบและการพัฒนามิดเดิลแวร์ที่ใช้ในการจัดการไฟล์ภายในคลัสเตอร์คอมพิวเตอร์ขององค์กรเสมือน โดยมิดเดิลแวร์นี้ใช้มาตรฐานของ Globus ในการพัฒนา การออกแบบระบบใช้หลักการเดียวกับระบบไฟล์คลัสเตอร์คือการเก็บไฟล์ไว้ที่เครื่องประมวลผลแทนเครื่องให้บริการไฟล์ ตัวโปรแกรมได้แบ่งออกเป็นสองส่วน ส่วนแรกติดตั้งอยู่บนเครื่องให้บริการ GHOSTS ทำหน้าที่ติดต่อกับผู้ใช้งานเพื่อรับคำสั่งและแสดงผล ส่วนที่สองติดตั้งอยู่ภายในคลัสเตอร์คอมพิวเตอร์ ทำหน้าที่ติดต่อกับระบบไฟล์คลัสเตอร์เพื่อสอบถามข้อมูลและจัดการไฟล์ และผู้วิจัยยังได้พัฒนาโปรแกรมผู้จัดการงานที่ทำงานร่วมกับ GRAM, SCFS, และ SGE ขึ้นมา เพื่อช่วยให้ผู้ใช้สามารถส่งงานไปประมวลผลยังเครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ที่มีไฟล์ที่ต้องการเก็บอยู่ได้

บทที่ 5

การทดลองและวิเคราะห์ผลการทดลอง

5.1 การทดสอบวัดประสิทธิภาพของ SCFS

5.1.1 เครื่องมือที่ใช้ในการทดลอง

1) เครื่องคลัสเตอร์คอมพิวเตอร์ที่ใช้ในการทดลองเป็นคลัสเตอร์ที่มี 4 เครื่อง
   - เครื่องควบคุม 1 เครื่อง และเครื่องคำนวณ 3 เครื่อง
   - ความเร็วหน่วยประมวลผล 2.8 GHz
   - หน่วยความจำ 1 GB
2) ระบบปฏิบัติการที่ใช้คือ NPACI Rocks Cluster 4.2.1
3) โปรแกรมจัดลำดับงานที่ใช้คือ Sun Grid Engine (SGE)
4) โปรแกรมจาวารุ่น 1.5
5) ระบบฐานข้อมูล MySQL รุ่น 4.1.20
6) ระบบเครือข่ายเป็น LAN ความเร็ว 100 Mbit/s

5.1.2 การวัดประสิทธิภาพการทำงานของฟังก์ชันส่วนเมตะดาต้า

การทดลองนี้มีวัตถุประสงค์เพื่อวัดเวลาที่ใช้ในการทำงานของฟังก์หลักๆ ของส่วนเมตะดาต้า

ตารางที่ 5.1: เวลาที่ใช้ในการทำงานของฟังก์ชันการจัดการเมตะดาต้า

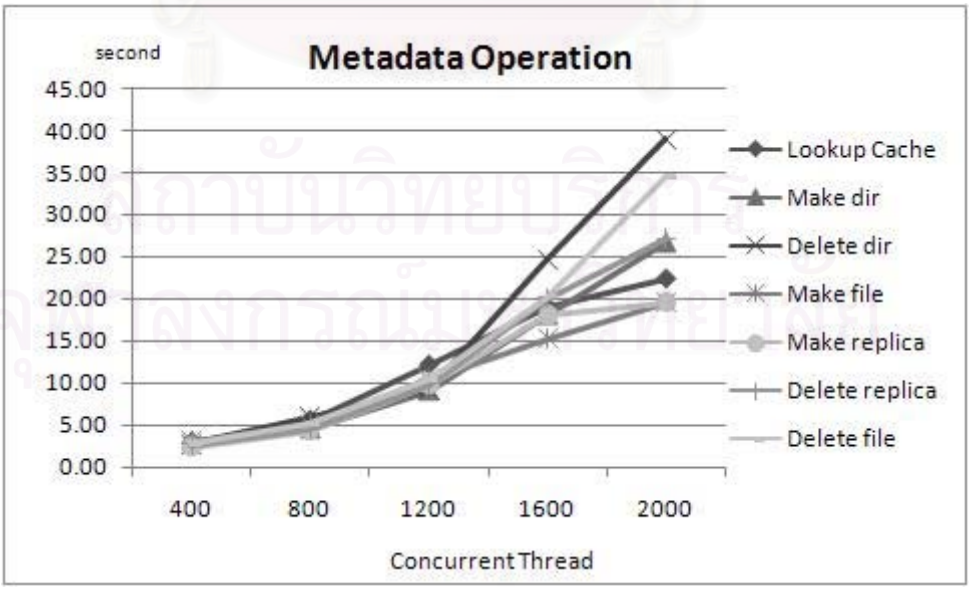
ฟังก์ชัน	เวลา (ms)
ค้นหา 1 ไฟล์และสร้าง *MetaData*	42.57
ค้นหา *MetaData* ของไฟล์ 1 ไฟล์ในแคช	6.12
เพิ่ม *MetaData* ของ 1 ไดเรกทอรี	6.27
ลบ *MetaData* ของ 1 ไดเรกทอรี	5.24
เพิ่ม *MetaData* ของ 1 ไฟล์	6.03
เพิ่ม *MetaData* เมื่อสั่งทำซ้ำไฟล์ 1 ฉบับ	3.95
ลบ *MetaData* ของฉบับของไฟล์ที่ทำซ้ำ 1 ฉบับ	3.78
ลบ *MetaData* ของ 1 ไฟล์	4.94

เนื่องจากการค้นหาข้อมูลเพื่อสร้าง *MetaData* ของไฟล์ขึ้นมานั้น ระบบจะต้องสอบถามข้อมูลจากระบบฐานข้อมูลและเครื่องให้บริการไฟล์ที่มีไฟล์เก็บอยู่ (ผ่านเครือข่าย) ดังนั้นเวลาที่ใช้ในการค้นหาและสร้าง *MetaData* ของไฟล์ในกรณีที่ *MetaData* ไม่ได้เก็บอยู่ในแคชจะใช้เวลานานกว่าฟังก์ชันอื่นๆ ค่อนข้างมาก และเนื่องจากทุกๆ ฟังก์ชันของ SCFS จะต้องมีการใช้ *MetaData* ในการทำงาน ดังนั้นการใช้แคชช่วยเก็บ *MetaData* จะมีส่วนช่วยลดเวลาที่ใช้ในการทำงานได้

5.1.3 การวัดประสิทธิภาพเมตะดาต้าในการรองรับหลายโปรเซสพร้อมกัน

การทดลองนี้มีวัตถุประสงค์เพื่อวัดประสิทธิภาพของโปรเซสส่วนเมตะดาต้าในกรณีที่มีหลายๆ โปรเซสเรียกใช้งานพร้อมๆ กัน โปรแกรมที่ใช้ในการทดลองเป็นโปรแกรมภาษาจาวาที่จะสร้างเทรดขึ้นมาจำนวนมากเพื่อเรียกใช้งานฟังก์ชันบนเครื่องให้บริการเมตะดาต้า ซึ่งในการทำการทดลอง แต่ละเทรดจะเริ่มต้นทำงานเกือบจะพร้อมๆ กัน และจะเริ่มต้นจับเวลาตั้งแต่เทรดแรกเริ่มทำงานจนกระทั่งเทรดสุดท้ายทำงานเสร็จ โดยฟังก์ชันที่ทำการทดสอบมีทั้งหมด 7 ฟังก์ชันคือ

1) การค้นหา *MetaData* ของไฟล์จากแคช (Lookup Cache)
2) การสร้าง *MetaData* ของไดเรกทอรี (Make dir)
3) การลบ *MetaData* ของไดเรกทอรี (Delete dir)
4) การสร้าง *MetaData* ของไฟล์ (Make file)
5) การสร้าง *MetaData* ของฉบับที่ทำซ้ำของไฟล์ (Make replica)
6) การลบ *MetaData* ของฉบับที่ทำซ้ำของไฟล์ (Delete replica)
7) การลบ *MetaData* ของไฟล์ (Delete file)



รูปที่ 5.1: แสดงเวลาที่ใช้การทำงานกรณีที่มีหลายๆ เทรดทำงานพร้อมกัน

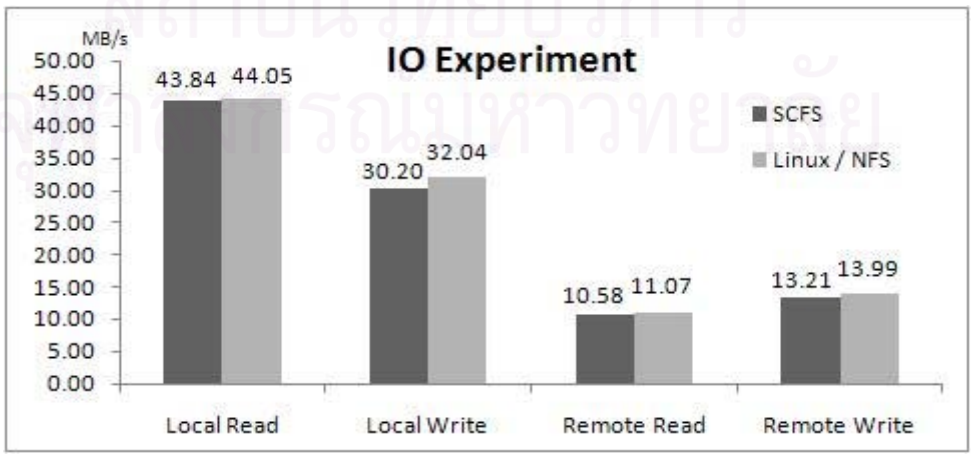
จากการเก็บข้อมูลที่โปรเซสให้บริการบนเครื่องให้บริการเมตะดาต้าพบว่า จำนวนเทรดที่กำลังทำงานพร้อมๆ กันมีน้อยกว่าจำนวนเทรดที่สั่งเข้ามาเล็กน้อย เช่นหากสั่งเข้ามา 1,200 เทรด ก็จะมีจำนวนเทรดที่ทำงานพร้อมกันประมาณ 950 – 1,200 เทรด เป็นต้น

จากผลการทดลองในรูปที่ 5.1 พบว่าหากจำนวนเทรดที่ทำงานพร้อมกันมีมากขึ้น จะทำให้เวลาที่ใช้ในการประมวลผลมีแนวโน้มที่จะเพิ่มขึ้นในรูปแบบของฟังก์ชันเลขชี้กำลัง (Exponential Function) สาเหตุที่แนวโน้มของกราฟเป็นเช่นนี้เนื่องจาก เมื่อมีจำนวนเทรดเพิ่มมากขึ้น หน่วยประมวลผลบนเครื่องให้บริการเมตะดาต้าจะต้องสลับเทรดที่ทำงานเพื่อให้ทุกๆ เทรดมีโอกาสทำงานเท่ากัน ซึ่งปัจจัยในเรื่องการสลับเทรดเข้าออกนี้เป็นปัจจัยที่ควบคุมไม่ได้และส่งผลให้เวลาที่ใช้ในการทำงานกรณีที่มีจำนวนเทรดมากๆ มีความแปรผันสูง

จากผลการทดลองที่ได้หากจำนวนเทรดยังไม่เกิน 800 เทรด ความแปรผันจะยังไม่สูงมาก (ไม่เกิน 1 วินาที) แต่หากมีจำนวนเทรดตั้งแต่ประมาณ 1,200 เทรดขึ้นไปแล้วความแปรผันของเวลาที่ใช้ในการทำงานจะเริ่มมีมากขึ้น (ประมาณ 4 – 10 วินาที) และหากมีจำนวนเทรดมากกว่า 2,000 เทรดก็จะเกิดปัญหาเรื่องหน่วยความจำไม่เพียงพอจนทำให้การทำงานบางครั้งก็เกิดข้อผิดพลาดขึ้นและไม่สามารถทำงานต่อได้

5.1.4 การวัดประสิทธิภาพของส่วน IO

การทดลองนี้มีวัตถุประสงค์เพื่อวัดประสิทธิภาพของการอ่านเขียนไฟล์ โดยใช้ API ของ SCFS คือ *IOConnection* โดย API ที่ใช้เปรียบเทียบคือ API ของจาวาซึ่งได้แก่ *java.io.FileInputStream* และ *java.io.FileOutputStream* การทดสอบการอ่านเขียนไฟล์แบบท้องถิ่นจะทำที่เครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ ส่วนการอ่านเขียนไฟล์แบบทางไกลจะทำจากเครื่องคำนวณไปที่เครื่องควบคุม ส่วนระบบไฟล์ที่ใช้ทดลองสำหรับ API ของจาวาคือระบบไฟล์ลินุกซ์ในการเข้าถึงไฟล์ท้องถิ่น และ NFS ในการเข้าถึงไฟล์แบบทางไกล ไฟล์ที่ใช้ในการทดลองมีขนาด 512 MB



รูปที่ 5.2: แสดงความเร็วที่ใช้ในการอ่านเขียนไฟล์ด้วย API ของ SCFS เทียบกับของจาวา

จากผลการทดลองในกราฟรูปที่ 5.2 ความเร็วในการเข้าถึงไฟล์ด้วย API ของ SCFS ในทุกๆ กรณีจะมีค่าน้อยกว่าความเร็วในการเข้าถึงไฟล์ด้วย API ของจาวา เนื่องจากส่วน API ของ SCFS มีขั้นตอนการทำงานเพิ่มขึ้นอีกระดับหนึ่งจากการอ่านเขียนไฟล์ปกติด้วย API ของจาวา อย่างไรก็ตามการทำงานเพิ่มเติมนี้ยังมีไม่มากนักจึงทำให้ความเร็วในการอ่านเขียนไฟล์ด้วย API ของ SCFS ช้ากว่า API ของจาวาเพียงเล็กน้อย

5.1.5 การวัดประสิทธิภาพของคำสั่งการทำงาน

การทดลองนี้มีวัตถุประสงค์เพื่อวัดความเร็วในการทำงานของคำสั่งหลักๆ ที่มีให้บริการ โดยเริ่มจับเวลาตั้งแต่เริ่มสั่งคำสั่งจนกระทั่งคำสั่งนั้นทำงานเสร็จและส่งผลลัพธ์ออกมา

ตารางที่ 5.2: ประสิทธิภาพของฟังก์ชันพื้นฐาน

ฟังก์ชัน	เวลา (ms)	ฟังก์ชัน	เวลา (ms)
สร้าง 1 ไดเรกทอรี	645.13	ทำซ้ำ 1 ไฟล์ (31 MB)	3464.21
สอบถามข้อมูล 1 ไฟล์	428.77	ลบ 1 ฉบับของไฟล์ที่ทำซ้ำ	654.59
สอบถามข้อมูล 1 ไดเรกทอรี	422.86	ย้าย 1 ไฟล์บนเครื่องเดียวกัน	736.04
เปลี่ยนคุณสมบัติ 1 ไฟล์	578.93	ทำสำเนา 1 ไฟล์ (31 MB)	3144.79
เปลี่ยนคุณสมบัติ 1 ไดเรกทอรี	582.27	แสดงรายการ 1 ไฟล์	423.56
เปลี่ยนคุณสมบัติ 3 ไดเรกทอรี	630.88	แสดงรายการ 10 ไฟล์	474.23
ลงทะเบียน 1 ไฟล์ (แบบลิงค์)	622.23	ลบ 1 ไฟล์	648.72
ส่งออก 1 ไฟล์ (31 MB)	989.36	ลบ 1 ไดเรกทอรี	685.70

5.2 การทดลองเปรียบเทียบการทำงานโดยการเข้าถึงไฟล์แบบท้องถิ่นและแบบทางไกล

การทดลองนี้มีวัตถุประสงค์เพื่อเปรียบเทียบเวลาที่ใช้ในการประมวลผลระหว่างการเข้าถึงไฟล์แบบท้องถิ่นและแบบทางไกล และทดสอบการนำ SCFS และโปรแกรมส่งงานของ SCFS มาประยุกต์ใช้ในการทำงาน โปรแกรมประยุกต์ที่ใช้ในการทดสอบคือการจำลองการเกิดคลื่นสึนามิ (ภาคผนวก ก.) ซึ่งเป็นการทำงานแบบกระแสงานอย่างหนึ่ง การทดลองนี้ประกอบด้วยงานย่อย 3 งานคือ

1) tunami เป็นการจำลองการทำงาน (Simulation) ซึ่งเป็นงานที่เน้นเรื่องการคำนวณและเขียนไฟล์เอาท์พุต ไฟล์อินพุตของโปรแกรมเป็นไฟล์ขนาดเล็ก 4 ไฟล์ ส่วนไฟล์เอาท์พุตเป็นไฟล์ขนาดเล็กจำนวน 400 ไฟล์ เมื่อรวมขนาดของเอาท์พุตแล้วมีขนาดประมาณ 6.9 GB

2) concat เป็นการต่อไฟล์ขนาดเล็กหลายๆ ไฟล์เข้าเป็นไฟล์ขนาดใหญ่ งานนี้เน้นการอ่านและเขียนไฟล์โดยเฉพาะ อินพุตเป็นไฟล์ผลลัพธ์ทั้งหมดจาก tunami ส่วนเอาท์พุตเป็นไฟล์ขนาดใหญ่ 11 ไฟล์ การทำงานในขั้นตอนนี้จะทำทันทีหลังจากที่งาน tunami ทำเสร็จ โดยคำสั่งทำงาน concat จะอยู่ในไฟล์สคริปต์เดียวกับคำสั่งทำงาน tunami
3) opendx เป็นการทำจินตทัศน์ (Visualization) งานนี้มีทั้งการคำนวณและการอ่านเขียนไฟล์ อินพุตคือไฟล์ผลลัพธ์ทั้งหมดจาก concat ส่วนเอาท์พุตเป็นไฟล์รูปภาพขนาดเล็กจำนวน 400 ไฟล์ ซึ่งมีขนาดรวมประมาณ 370 MB

ระบบไฟล์ที่ใช้ในการทดสอบการเข้าถึงไฟล์แบบท้องถิ่นคือ SCFS และ Gfarm (ใช้ในลักษณะที่เป็นระบบไฟล์คลัสเตอร์) สำหรับ SCFS สคริปต์งานย่อยแรกจะส่งด้วยคำสั่ง ‘qsub’ ของ SGE เนื่องจากงานนี้ไม่มีไฟล์อินพุตขนาดใหญ่และยังไม่มีไฟล์เก็บอยู่ใน SCFS จึงไม่สามาถใช้โปรแกรมส่งงานของ SCFS ได้ ส่วน Gfarm ส่งด้วยคำสั่ง ‘gfrun’ โดยผู้วิจัยได้กำหนดเองว่าจะให้กรณีทดลองใดไปประมวลผลที่เครื่องใด ในการทำงาน tunami จะสร้างไฟล์ผลลัพธ์ทั้งหมดขึ้นบนเครื่องที่ประมวลผล จากนั้น concat จะอ่านไฟล์เหล่านี้เพื่อสร้างไฟล์ผลลัพธ์ของตัวเองที่เครื่องประมวลผลเช่นกัน ไฟล์ผลลัพธ์จาก concat จะถูกลงทะเบียนเข้าสู่ SCFS และ Gfarm งานย่อยสุดท้ายคือ opendx จะถูกส่งมาประมวลผลที่เครื่องคำนวณที่มีไฟล์นี้เก็บอยู่ โดยในกรณีของ SCFS ได้ใช้โปรแกรมส่งงานของ SCFS โดยเลือกไฟล์หนึ่งจากไฟล์ผลลัพธ์ 11 ไฟล์ของ concat ระบุเป็นพารามิเตอร์ <scfs-file> ส่วน gfarm ก็ส่งด้วยคำสั่ง ‘gfrun’ แล้วกำหนดให้ไปทำงานบนเครื่องที่มีไฟล์เก็บอยู่

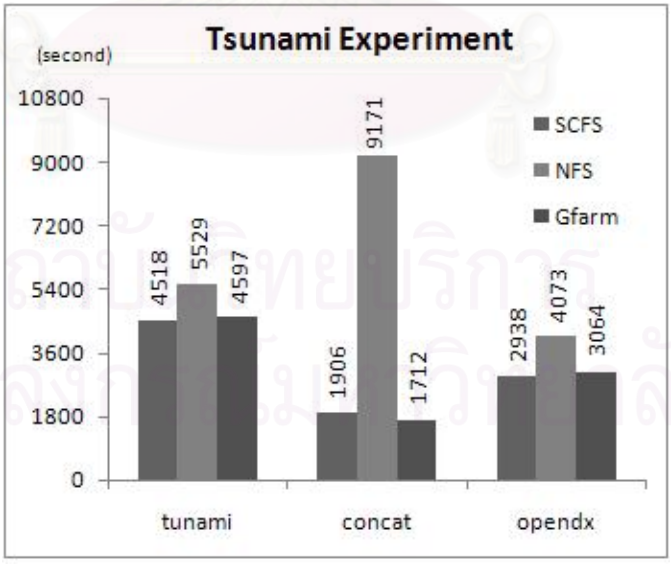
ระบบไฟล์ที่ใช้สำหรับการเข้าถึงไฟล์แบบทางไกลคือ NFS โดยเครื่องควบคุมจะทำหน้าที่เป็นเครื่องให้บริการไฟล์ ทุกๆ งานย่อยจะอ่านเขียนไฟล์แบบทางไกลในการทำงาน และใช้คำสั่ง ‘qsub’ ของ SGE ในการส่งงานทั้งหมด

ตารางที่ 5.3: เปรียบเทียบขั้นตอนการทำงานของการอ่านเขียนไฟล์แบบท้องถิ่นและแบบทางไกล

SCFS & Gfarm	NFS
1) ส่งงาน tunami + concat 6 งาน scfs - ใช้คำสั่ง qsub ของ SGE gfarm - ใช้คำสั่ง gfrun	1) ส่งงาน tunami + concat 6 งาน - ใช้คำสั่ง qsub ของ SGE
2) tunami 6 งาน แต่ละเครื่องประมวลผลทีละ 2 งาน โดย - อ่านอินพุตไฟล์แบบทางไกล - เขียนเอาท์พุตไฟล์แบบท้องถิ่น	2) tunami 6 งาน แต่ละเครื่องประมวลผลทีละ 2 งาน โดย - อ่านอินพุตไฟล์แบบทางไกล - เขียนเอาท์พุตไฟล์แบบทางไกล

SCFS & Gfarm	NFS
3) concat 6 งาน แต่ละเครื่องประมวลผลทีละ 2 งาน โดย - อ่านเขียนไฟล์แบบท้องถิ่น	3) concat 6 งาน แต่ละเครื่องประมวลผลทีละ 2 งาน โดย - อ่านเขียนไฟล์แบบทางไกล
4) ส่งงาน opendx 6 งาน scfs - ใช้คำสั่ง scfs-submit gfarm - ใช้คำสั่ง gfrun	4) ส่งงาน opendx 6 งาน - ใช้คำสั่งของ SGE
5) opendx แต่ละเครื่องประมวลผลทีละ 1 งาน โดย - อ่านไฟล์โปรแกรมแบบทางไกล - อ่านเขียนไฟล์ข้อมูลแบบท้องถิ่น	5) opendx แต่ละเครื่องประมวลผลทีละ 1 งาน โดย - อ่านไฟล์โปรแกรมแบบทางไกล - อ่านเขียนไฟล์ข้อมูลแบบทางไกล

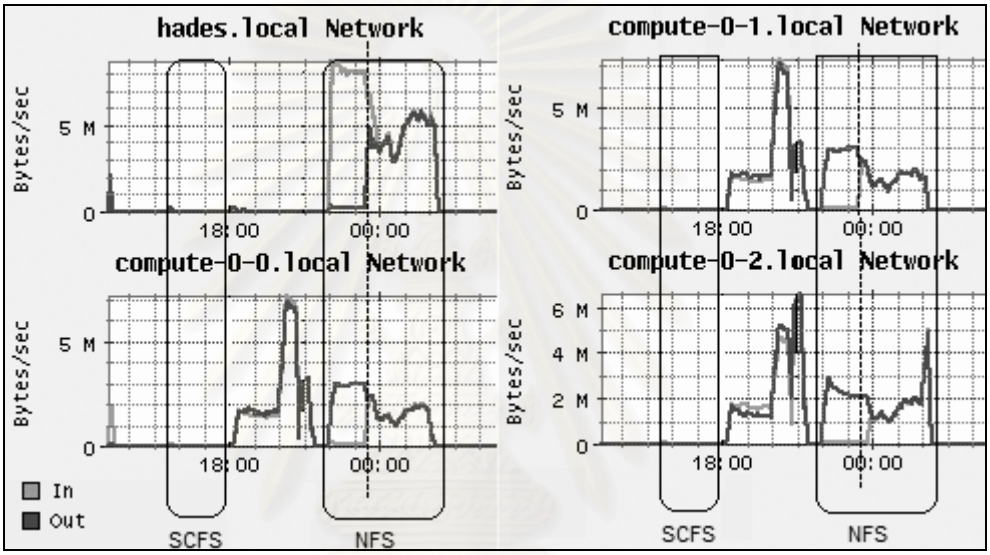
เนื่องจากไฟล์โปรแกรมของทั้งกรณีงาน tunami และ opendx มีขนาดเล็กมากเมื่อเทียบกับไฟล์ผลลัพธ์ของงาน ดังนั้นในการทดลองนี้จึงให้ NFS ในการเก็บไฟล์โปรแกรมทั้งหมด เครื่องคลัสเตอร์คอมพิวเตอร์ที่ใช้ในการทดลองมี 4 เครื่อง เป็นเครื่องควบคุม 1 เครื่อง เครื่องคำนวณ 3 เครื่อง เครื่องคำนวณแต่ละเครื่องสามารถประมวลผลได้ 2 งานพร้อมๆ กัน ยกเว้นงาน opendx ที่ต้องใช้หน่วยความจำปริมาณมาก ทำให้เครื่องคำนวณแต่ละเครื่องสามารถทำงาน opendx ได้แค่ทีละ 1 งานเท่านั้น ระบบปฏิบัติการที่ใช้คือ NPACI Rock Cluster 4.2



รูปที่ 5.3: แสดงเวลาที่ใช้ในการประมวลผลแต่ละขั้นตอนของการจำลองคลื่นสึนามิ

รูปที่ 5.3 แสดงให้เห็นว่าโปรแกรมที่ใช้การเข้าถึงไฟล์แบบท้องถิ่นทั้งในกรณีของ SCFS และ Gfarm ใช้เวลาน้อยกว่าการเข้าถึงไฟล์แบบทางไกลด้วย NFS ในทุกๆ ขั้นตอน คือกรณีขั้นตอน tunami

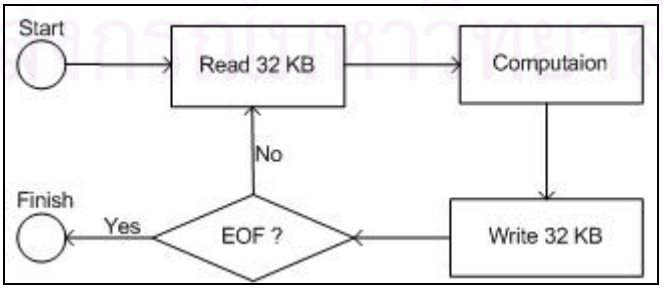
SCFS ใช้เวลาน้อยกว่า 18.29% ขั้นตอน concat ใช้เวลาน้อยกว่า 79.22% ส่วน opendx ใช้เวลาน้อยกว่า 27.87 % ส่วนที่สังเกตได้ชัดเจนคือในขั้นตอนการต่อไฟล์ (ในกรณีของ NFS คือตั้งแต่เวลาประมาณ 23.30 น. ตามรูปที่ 5.4) ซึ่งเป็นขั้นตอนที่เน้นในเรื่องการอ่านเขียนไฟล์ การประมวลผลแบบเข้าถึงไฟล์ท้องถิ่นให้ผลดีกว่ามาก (ใช้เวลาน้อยกว่ามาก) เนื่องจากในกรณีการเข้าถึงไฟล์แบบทางไกล ทุกๆ โปรเซสอ่านเขียนไฟล์ที่เครื่องให้บริการไฟล์ (ในที่นี้คือเครื่องควบคุม) พร้อมๆ กัน ทำให้เกิดปัญหาคอขวดที่เครื่องควบคุม (hades.local) ดังที่แสดงในรูปที่ 5.4 คือแบนด์วิดท์ที่ใช้สำหรับ NFS บนเครื่องควบคุมสูงกว่าบนเครื่องคำนวณทุกเครื่องประมาณสามเท่าตัว



รูปที่ 5.4: แสดงการใช้งานเครือข่ายของเครื่องภายในคลัสเตอร์คอมพิวเตอร์

5.3 การทดลองเรื่องปัญหาคอขวดบนเครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์

การทดลองนี้มีวัตถุประสงค์เพื่อทดสอบเรื่องปัจจัยที่มีผลต่อปัญหาคอขวดบนเครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์ โดยโปรแกรมที่ใช้ในการทดลองเป็นโปรแกรมที่เขียนขึ้นด้วยภาษาจาวาที่มีขั้นตอนการทำงานของโปรแกรมดังรูปที่ 5.5 คือ



รูปที่ 5.5: ขั้นตอนการทำงานของโปรแกรมที่ใช้ทดสอบปัญหาคอขวด

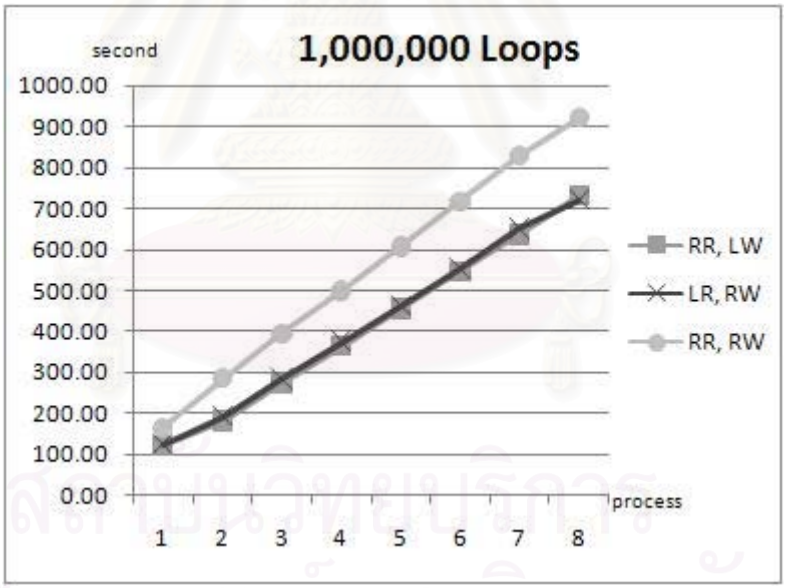
1) ส่วน IO จะประกอบด้วยการอ่านและเขียนไฟล์ครั้งละ 32 KB ตั้งแต่ต้นไฟล์จนจบไฟล์

2) ส่วนคำนวณ มีลักษณะเป็นรอบ (Loop) โดยที่ใน 1 รอบจะประกอบด้วยคำสั่งสามคำสั่งคือการเปรียบเทียบ (Comparing) การบวกเลข (Addition) และการเพิ่มค่า (Increment) โดย

ในการทดลองนี้ได้กำหนดให้อ่านเขียนไฟล์ขนาด 1 GB และได้กำหนดให้มีการคำนวณทั้งหมด 1,000,000 รอบ (3,000,000 คำสั่ง) ต่อการอ่านเขียนไฟล์ 32 KB ตัวแปรที่กำหนดในการทดลองนี้มีอยู่ 2 ชนิดคือ

1) วิธีการอ่านเขียนไฟล์
   - อ่านแบบทางไกลและเขียนแบบท้องถิ่น (RR, LW)
   - อ่านแบบท้องถิ่นและเขียนแบบทางไกล (LR, RW)
   - อ่านแบบทางไกลและเขียนแบบทางไกล (RR, RW)
2) จำนวนโปรเซส (จำนวนเครื่องคำนวณ) ที่อ่านเขียนไฟล์ที่เครื่องควบคุมพร้อมๆ กัน จะเริ่มตั้งแต่กรณี 1 โปรเซสไปจนถึงกรณี 8 โปรเซสพร้อมๆ กัน (1 เครื่องคำนวณต่อ 1 โปรเซส)

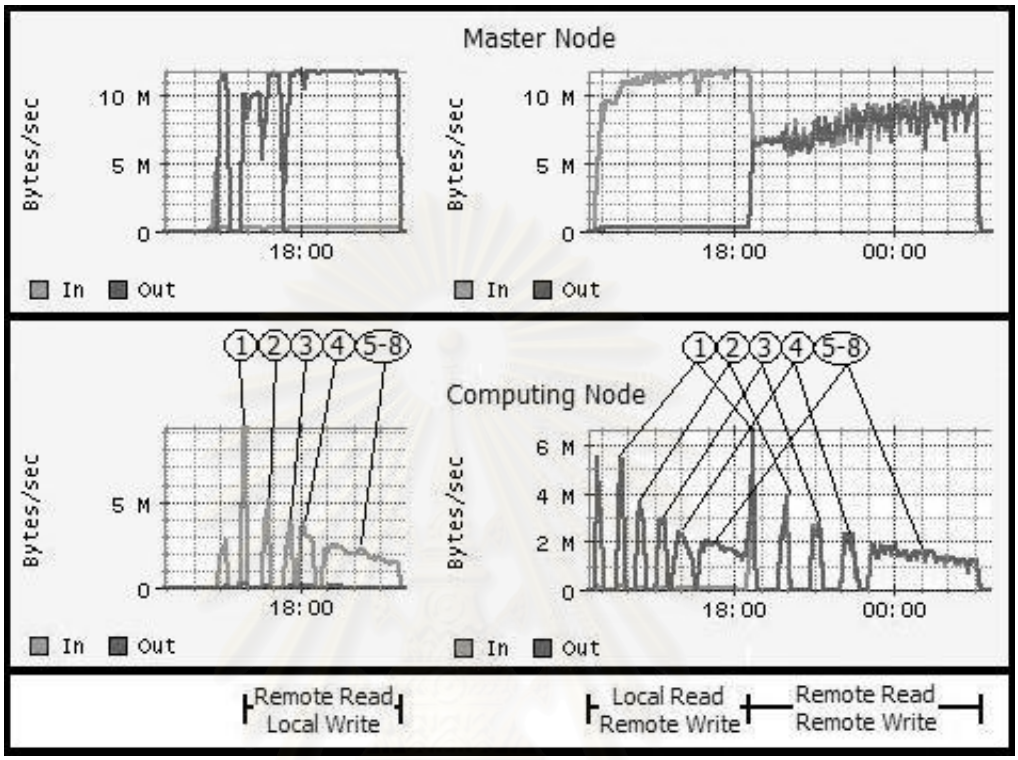
เครื่องคลัสเตอร์คอมพิวเตอร์ที่ใช้ในการทดลองเป็นเครื่องรุ่นเดียวกับการทดลองที่ผ่านๆ มาแต่เพิ่มจำนวนเครื่องคำนวณจาก 3 เครื่องเป็น 8 เครื่อง



รูปที่ 5.6: แสดงเวลาที่ใช้ในการทำงาน

ผลการทดลองในรูป 5.6 แสดงให้เห็นว่า เวลาที่ใช้ในการทำงานจะแปรผันตรงกับจำนวนโปรเซสที่ทำงานพร้อมๆ กัน โดยในกรณีที่มีการอ่านหรือเขียนไฟล์แบบทางไกลเพียงอย่างเดียวจะใช้เวลาในการทำงานเกือบจะเท่าๆ กัน และจะน้อยกว่ากรณีที่ทั้งการอ่านและเขียนไฟล์ต้องทำแบบทางไกล ซึ่งเมื่อดูจากแบนด์วิดท์ของเครือข่ายบนเครื่องควบคุมและเครื่องคำนวณที่แสดงในกราฟรูปที่ 5.7 จะเห็นว่า ในทุกๆ กรณีของการอ่านเขียนไฟล์เมื่อมีเพียงแค่ 1 โปรเซส ที่เครื่องคำนวณจะได้รับแบนด์วิดท์สูงสุด และเมื่อมีจำนวนโปรเซสเพิ่มขึ้น แบนด์วิดท์ที่เครื่องควบคุมจะถูกแบ่งปันให้กับทุกๆ โปรเซส ซึ่งจะเห็นได้จาก

รูปที่ 5.7 คือในกรณี 2 – 8 โปรเซส เครื่องคำนวณจะได้รับแบนด์วิดท์ลดน้อยลงเรื่อยๆ ตามจำนวนโปรเซส



รูปที่ 5.7: แสดงการใช้งานแบนด์วิดท์บนเครื่องควบคุมและเครื่องคำนวณเครื่องหนึ่ง

กรณีการทำงานเพียง 1 โปรเซสแสดงให้เห็นปริมาณแบนด์วิดท์ที่ 1 โปรเซสใช้ในการทำงานอย่างเต็มที่ ซึ่งสำหรับกรณีการอ่านหรือเขียนไฟล์แบบทางไกลเพียงอย่างเดียวจะใช้งานแบนด์วิดท์ประมาณ 10 MB/s ส่วนกรณีการอ่านและเขียนไฟล์แบบทางไกลจะใช้งานแบนด์วิดท์ทั้งอินพุตและเอาท์พุตประมาณ 7 MB/s และเมื่อเพิ่มจำนวนโปรเซสที่เข้าใช้งานไฟล์บนเครื่องควบคุมแล้ว ในกรณีการอ่านหรือเขียนไฟล์แบบทางไกลเพียงอย่างเดียว ปริมาณแบนด์วิดท์ที่เครื่องควบคุมจะถูกจำกัดอยู่ที่ 12 MB/s ส่วนในกรณีที่มีทั้งการอ่านและเขียนไฟล์แบบทางไกล ปริมาณแบนด์วิดท์ทั้งอินพุตและเอาท์พุตที่เครื่องควบคุมจะค่อยๆ สูงขึ้น ซึ่งในการทดลองนี้วัดปริมาณแบนด์วิดท์ได้สูงประมาณ 9 – 10 MB/s ในกรณี 5 – 8 โปรเซส

จากการทดลองนี้สามารถสรุปได้ว่าในกรณีที่โปรเซสบนเครื่องเครื่องคำนวณหลายๆ เครื่องต้องการอ่านหรือเขียนไฟล์หรือทั้งอ่านและเขียนไฟล์บนเครื่องควบคุมพร้อมๆ กัน จะเกิดการแบ่งปันการใช้งานแบนด์วิดท์ที่เครื่องควบคุม ซึ่งการแบ่งปันแบนด์วิดท์นี้จะขึ้นอยู่กับจำนวนโปรเซสและวิธีการที่แต่ละโปรเซสเข้าใช้งานไฟล์บนเครื่องควบคุมในขณะนั้น และหากแบนด์วิดท์ที่โปรเซสบนเครื่องคำนวณแต่ละเครื่องได้รับไม่เพียงพอกับความต้องการก็จะส่งผลให้โปรเซสนั้นต้องเสียเวลารอการทำงานของส่วน IO มากขึ้นดังเช่นในกรณีของขั้นตอน concat ในการทดลองที่ 5.2

5.4 การทดลองเรื่องเวลาเพิ่มเติมของการเคลื่อนย้ายไฟล์ผ่านเครือข่าย

การทดลองในส่วนนี้มีวัตถุประสงค์เพื่อแสดงให้เห็นเวลาเพิ่มเติมที่ใช้ในการรับส่งไฟล์ผ่านเครือข่าย โปรแกรมประยุกต์ที่ใช้ในการทดลองนี้คือโปรแกรมจำลองการเกิดคลื่นสึนามิเช่นเดียวกับการทดลองในหัวข้อ 5.2 เครื่องคลัสเตอร์คอมพิวเตอร์ที่ใช้ในการทดสอบคือ เครื่องที่เป็นทรัพยากรของศูนย์ไทยกริดแห่งชาติจำนวน 2 เครื่องคือ araya.thaigrid.or.th ที่มหาวิทยาลัยเกษตรศาสตร์ และ pluto.cp.eng.chula.ac.th ที่จุฬาลงกรณ์มหาวิทยาลัย ในการทดลองไฟล์ที่เป็นโปรแกรมและข้อมูลเบื้องต้นในการทำงานจะเก็บอยู่ที่เครื่อง araya ผู้วิจัยล็อกอินเข้าไปที่เครื่อง araya และสั่งงานมาประมวลผลที่เครื่อง pluto จากการวัดแบนด์วิดท์ด้วยโปรแกรม iperf [39] ระหว่างเครื่องทั้งสองในช่วงเวลาที่ใกล้เคียงกับเวลาช่วงที่ทำการทดลองได้ค่าเฉลี่ยคือ

- จาก pluto ไป araya มีแบนด์วิดท์ 86.11 Mbit/s
- จาก araya ไป pluto มีแบนด์วิดท์ 77.50 Mbit/s

ในการทดลองนี้ได้รวมงาน tunami และ concat เข้าเป็นหนึ่งงานย่อย และการส่งงานจะส่งแค่ทีละงานเพื่อตัดปัจจัยเรื่องปัญหาคอขวดออกจากการทดลอง เนื่องจากไฟล์โปรแกรมมีขนาดเล็กมากเมื่อเทียบกับไฟล์ข้อมูล ดังนั้นจึงเก็บอยู่ที่เครื่อง araya และจะถูกเคลื่อนย้ายไป pluto ทุกครั้งที่มีการคำนวณ และผลลัพธ์สุดท้ายที่ได้จาก opendx จะต้องถูกส่งกลับมาที่เครื่อง araya (สมมุติว่าผู้ใช้ต้องการดูรูปที่เครื่อง araya) โดยการทดลองนี้ได้แบ่งออกเป็น 2 กรณีคือ

กรณีแรกเป็นการจำลองวิธีการทำงานแบบเข้าถึงไฟล์แบบท้องถิ่น งาน tunami+concat จะถูกส่งเข้ามาโดยใช้บริการเป็น jobmanager-sge เนื่องจากตอนนี้ยังไม่มีไฟล์เก็บอยู่ใน SCFS ในการทำงานไฟล์ผลลัพธ์จะถูกเก็บไว้ที่เครื่องคำนวณของ pluto แล้วลงทะเบียนเข้าไปเก็บใน SCFS จากนั้นงาน opendx จึงถูกส่งเข้ามาโดยใช้บริการ jobmanager-ccss เพื่อส่งงานไปยังเครื่องคำนวณที่มีไฟล์ผลลัพธ์จากขั้นตอนแรกเก็บอยู่ ผลลัพธ์ของ opendx จะถูกเขียนผ่าน NFS ไปไว้ที่เครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์เพื่อเตรียมส่งกลับไปที่ araya

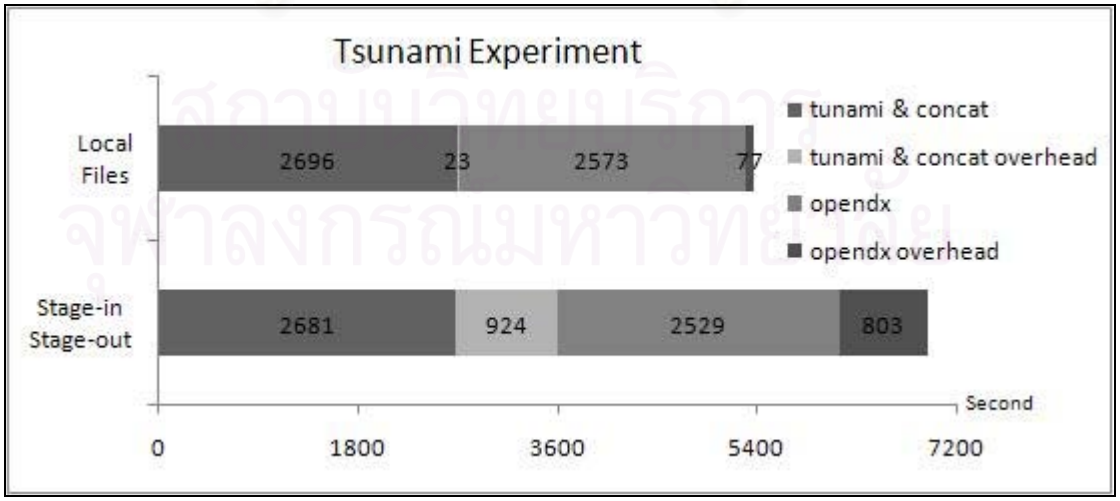
กรณีที่สองเป็นการจำลองวิธีการทำงานส่วนใหญ่ที่มีการใช้งานในระบบกริด คือจะมีการรับส่งไฟล์ทั้งหมดระหว่างเครื่องที่สั่งงานและคลัสเตอร์คอมพิวเตอร์ที่ประมวลผลก่อนและหลังการคำนวณ งานย่อยที่สั่งเข้ามาจะใช้บริการ jobmanager-sge และการอ่านเขียนไฟล์บน pluto จะทำผ่าน NFS คือเครื่องคำนวณอ่านเขียนไฟล์ที่อยู่บนเครื่องควบคุมแบบทางไกล

ตารางที่ 5.4: เปรียบเทียบขั้นตอนการทดลองโปรแกรมจำลองการเกิดคลื่นสึนามิในระดับกริด

tunami+concat	Local File	Remote File
โปรแกรมส่งงาน	jobmanager-sge	
stage-in	ไฟล์โปรแกรม	

	ไฟล์ข้อมูลอินพุต	
การประมวลผล	อ่านไฟล์โปรแกรมและไฟล์ข้อมูลอินพุตผ่านทาง NFS	
	เขียนไฟล์เอาท์พุตแบบท้องถิ่น และลงทะเบียนเข้า SCFS	เขียนไฟล์เอาท์พุตผ่านทาง NFS ไปที่เครื่องควบคุม
stage-out	-	ไฟล์ข้อมูลเอาท์พุต 11 ไฟล์ ขนาดประมาณ 6.9 GB
opendx	**Local File**	**Remote File**
โปรแกรมส่งงาน	jobmanager-ccss	jobmanager-sge
stage-in	ไฟล์โปรแกรม	ไฟล์โปรแกรม
		ไฟล์ข้อมูลอินพุต 11 ไฟล์ ขนาดรวมประมาณ 6.9 GB
การประมวลผล	อ่านไฟล์โปรแกรมแบบทางไกลผ่านทาง NFS	
	อ่านไฟล์อินพุตแบบท้องถิ่น	อ่านไฟล์อินพุตแบบทางไกล
	เขียนไฟล์เอาท์พุตแบบทางไกลผ่านทาง NFS ไปที่เครื่องควบคุม	
stage-out	ไฟล์รูปภาพ 400 ไฟล์ ขนาดประมาณ 370 MB	

เนื่องจากเวลาบนเครื่อง araya และ pluto อาจจะไม่ตรงกัน ดังนั้นจึงมีการจับเวลาบนทั้งสองเครื่อง ที่เครื่อง araya จะจับเวลาทั้งหมดที่ใช้ในการทำงาน (t_{total}) ส่วนที่เครื่อง pluto จะจับเวลาที่โปรแกรมใช้ในการประมวลผล ($t_{execution}$) ดังนั้น เวลาที่เพิ่มขึ้นมา ($t_{overhead}$) คือ เวลาที่ใช้ในการเตรียมการของ globus ทั้งบนเครื่อง pluto และ araya และเวลาที่ใช้ในการรับส่งข้อมูลระหว่างเครื่องทั้งสอง (ขั้นตอน stage-in และ stage-out) ซึ่งมีค่าเท่ากับ "$t_{overhead} = t_{total} - t_{execution}$"



รูปที่ 5.8: เวลาที่ใช้ในการประมวลผลแต่ละขั้นตอนของการจำลองคลื่นสึนามิ

จากผลการทดลองในรูปที่ 5.8 กรณีที่สองจะมีเวลาที่ต้องใช้เพิ่มเติมมากถึง 24.88% ของเวลาที่ใช้ในการทำงานทั้งหมด ส่วนในกรณีแรกใช้เวลาเพิ่มเติมเพียง 1.84% ของเวลาทั้งหมด ซึ่งเวลาที่มากกว่าของการทดลองที่สองนี้เกิดขึ้นจากการรับส่งไฟล์ระหว่างกลางขนาดใหญ่ระหว่างเครื่องทั้งสอง ซึ่งหากไฟล์มีขนาดใหญ่กว่านี้ หรือแบนด์วิดท์ของเครือข่ายน้อยกว่านี้ เวลาที่ใช้รับส่งไฟล์ก็จะมากขึ้น

5.6 **สรุป**

ในบทนี้ได้นำเสนอการทดลองและการวิเคราะห์ผลการทดลอง โดยเริ่มตั้งแต่การทดสอบการวัดประสิทธิภาพฟังก์ชันการทำงานของ SCFS ซึ่งได้แก่ส่วนเมตะดาต้า, ส่วน IO, และคำสั่งที่ใช้ในการทำงาน จากนั้นจึงทดลองนำระบบมาใช้ในโปรแกรมประยุกต์เรื่องการจำลองการเกิดคลื่นสึนามิเพื่อทดสอบเวลาที่ใช้ในการทำงานแบบเข้าถึงไฟล์ท้องถิ่นเทียบกับการเข้าถึงไฟล์ทางไกล แล้วจึงทำการทดลองเรื่องการเกิดปัญหาคอขวดที่เครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์ และสุดท้ายคือการทดลองเรื่องเวลาเพิ่มเติมที่ใช้ในการเคลื่อนย้ายไฟล์ผ่านเครือข่ายในระบบกริดด้วยโปรแกรมจำลองการเกิดคลื่นสึนามิ

ผลการทดลองด้วยโปรแกรมจำลองการเกิดคลื่นสึนามิทั้งในระดับคลัสเตอร์คอมพิวเตอร์และในระดับกริดแสดงให้เห็นว่าการทำงานโดยการอ่านเขียนไฟล์ท้องถิ่นให้ผลดีกว่าการทำงานโดยการอ่านเขียนไฟล์แบบทางไกล และการทำงานแบบมีการเคลื่อนย้ายไฟล์ผ่านเครือข่าย เพราะว่าการทำงานแบบอ่านเขียนไฟล์ท้องถิ่นสามารถแก้ได้ทั้งปัญหาคอขวดที่เกิดขึ้นที่เครื่องควบคุมของคลัสเตอร์คอมพิวเตอร์และปัญหาเรื่องการเคลื่อนย้ายไฟล์ผ่านเครือข่ายระหว่างเครื่องทรัพยากรในกริด

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

บทที่ 6

สรุปผลการวิจัย

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอระบบจัดการไฟล์ในระดับคลัสเตอร์คอมพิวเตอร์และองค์กรเสมือนเพื่อสนับสนุนการประมวลผลแบบเข้าถึงไฟล์แบบท้องถิ่น โดยวิธีการที่ใช้คือการสนับสนุนการทำงานของโปรแกรมจัดลำดับงานให้ทำงานแบบส่งงานไปหาข้อมูล สาเหตุเบื้องต้นของการทำวิจัยในเรื่องนี้คือปัญหาเรื่องเวลาที่ใช้ในการรับส่งไฟล์ระหว่างเครื่องที่เก็บไฟล์และเครื่องที่ประมวลผลในคลัสเตอร์คอมพิวเตอร์ โดยเฉพาะในกรณีที่เครือข่ายระหว่างเครื่องต้นทางและเครื่องปลายทางมีความเร็วต่ำ และในกรณีการทำงานประเภทกระแสงานที่ใช้ไฟล์ขนาดใหญ่

ระบบที่งานวิจัยนี้นำเสนอประกอบด้วยโปรแกรมสองระดับคือระบบไฟล์คลัสเตอร์สำหรับจัดการไฟล์ในระดับคลัสเตอร์คอมพิวเตอร์และมิดเดิลแวร์สำหรับจัดการไฟล์ในระดับองค์กรเสมือน วิธีการหลักที่นำมาใช้ในงานวิจัยนี้คือการจัดเก็บไฟล์ไว้ที่เครื่องคำนวณในคลัสเตอร์คอมพิวเตอร์ที่ทำการประมวลผลแทนการส่งไฟล์กลับไปเก็บที่เครื่องที่สั่งงานหรือเครื่องให้บริการไฟล์ การทำเมตะดาต้าเพื่อเก็บข้อมูลที่เกี่ยวกับตำแหน่งของไฟล์ และการสร้างคำสั่งเพื่อให้สามารถสอบถามข้อมูลนี้ได้ ระบบไฟล์คลัสเตอร์จะรับผิดชอบในสามหน้าที่นี้ ส่วนมิดเดิลแวร์จะใช้ข้อมูลและบริการที่ได้จากระบบไฟล์คลัสเตอร์ในการดูแลภาพรวมและจัดการไฟล์ในระดับองค์กรเสมือน ผู้ใช้ (หรือโปรแกรมจัดการงานในองค์กรเสมือน) สามารถสอบถามข้อมูลของไฟล์ที่ต้องการและส่งงานไปยังเครื่องคลัสเตอร์คอมพิวเตอร์ที่มีไฟล์นั้นเก็บอยู่ได้ผ่านทางกริดพอร์ทอลของระบบ นอกจากนี้ในงานวิจัยนี้ยังได้พัฒนาโปรแกรมส่งงานอย่างง่ายเพื่อสนับสนุนการทำงานของโปรแกรมจัดลำดับงานในคลัสเตอร์คอมพิวเตอร์ให้สามารถส่งงานไปยังเครื่องคำนวณที่มีไฟล์เก็บอยู่ได้

ในงานวิจัยนี้ได้ทำการทดลองวัดประสิทธิภาพของระบบไฟล์คลัสเตอร์ที่พัฒนาขึ้นมา และทดลองประมวลผลโปรแกรมประยุกต์ประเภทกระแสงานที่ใช้ไฟล์ขนาดใหญ่ โดยเปรียบเทียบระหว่างการเข้าถึงไฟล์แบบทางไกลและการเข้าถึงไฟล์แบบท้องถิ่นทั้งในระดับเครื่องคลัสเตอร์คอมพิวเตอร์และระบบกริด การทดลองในระดับของคลัสเตอร์คอมพิวเตอร์ได้แสดงให้เห็นว่าการทำงานประเภทใช้ข้อมูลปริมาณมากโดยอ่านเขียนไฟล์แบบท้องถิ่นใช้เวลาในการประมวลผลน้อยกว่าการทำงานโดยอ่านเขียนไฟล์แบบทางไกล เนื่องจากการอ่านเขียนไฟล์บนเครื่องควบคุมแบบทางไกลจากเครื่องคำนวณหลายๆ เครื่องพร้อมกัน จะส่งผลให้เกิดการแบ่งปันแบนด์วิดท์ของเครื่องควบคุมระหว่างเครื่องคำนวณ ทำให้การอ่านเขียนไฟล์ของทุกๆ โปรเซสบนเครื่องคำนวณช้าลงตามจำนวนโปรเซสที่เข้าอ่านเขียนไฟล์บนเครื่องควบคุม ส่วนการทดลองในระดับกริดนั้น แสดงให้เห็นถึงเวลาเพิ่มเติมที่ต้องใช้ในการเคลื่อนย้ายไฟล์ผ่าน

เครือข่ายและการเริ่มต้นระบบของ Globus ซึ่งมีค่าประมาณ 24.88% ของเวลาที่ใช้ในการประมวลผล ซึ่งค่อนข้างมากเมื่อเปรียบเทียบกับกรณีที่ไฟล์ระหว่างกลางอยู่ที่เครื่องที่ทำการประมวลผลอยู่แล้ว (1.84%)

6.2 ข้อจำกัดของระบบจัดการไฟล์

6.2.1 โปรแกรมส่งงานและโปรแกรมผู้จัดการงาน

1) โปรแกรมทั้งสองสามารถรับไฟล์อินพุตที่จะใช้สอบถาม SCFS ได้แค่หนึ่งไฟล์เท่านั้น เนื่องจากตัวโปรแกรมไม่มีระบบที่จะจัดการในกรณีไฟล์ที่เกี่ยวข้องกับงานมีอยู่หลายไฟล์ และแต่ละไฟล์อยู่ต่างเครื่องกัน
2) หากเครื่องที่มีไฟล์เก็บอยู่ไม่ว่าง ตัวโปรแกรมก็ไม่สามารถตัดสินใจแทนผู้ใช้หรือให้คำแนะนำกับผู้ใช้ได้ว่าควรจะรอให้เครื่องที่มีไฟล์เก็บอยู่ว่างหรือควรจะสั่งทำซ้ำไฟล์นั้นทันที และหากต้องการทำซ้ำไฟล์ โปรแกรมก็ไม่สามารถบอกได้ว่าควรจะทำซ้ำไฟล์ไปที่เครื่องให้บริการไฟล์เครื่องใด

6.2.2 การเคลื่อนย้ายไฟล์ข้ามคลัสเตอร์คอมพิวเตอร์

เนื่องจากระบบจัดการไฟล์นี้ใช้วิธีการเก็บไฟล์ไว้ในระบบไฟล์คลัสเตอร์บนเครื่องคำนวณของคลัสเตอร์คอมพิวเตอร์ ดังนั้นหากต้องการเคลื่อนย้ายไฟล์จากเครื่องคำนวณของคลัสเตอร์ต้นทางไปเก็บยังเครื่องคำนวณของคลัสเตอร์ปลายทาง ไฟล์จะต้องถูกเคลื่อนย้ายผ่านทางเครื่องควบคุมของทั้งคลัสเตอร์ต้นทางและปลายทางก่อน ซึ่งวิธีที่ใช้ในงานวิจัยนี้ใช้การสร้างไฟล์ชั่วคราวที่เครื่องควบคุมของทั้งคลัสเตอร์ต้นทางและคลัสเตอร์ปลายทาง

6.2.3 ข้อจำกัดของการทำซ้ำไฟล์

1) ระบบการจัดการไฟล์นี้สนับสนุนการทำซ้ำไฟล์เฉพาะในระดับของคลัสเตอร์คอมพิวเตอร์เท่านั้น ยังไม่สนับสนุนการทำซ้ำไฟล์ในระดับองค์กรเสมือน ทั้งนี้เนื่องระบบส่วนให้บริการของ CCSS จะรับผิดชอบเฉพาะการทำงานบนเครื่องคลัสเตอร์คอมพิวเตอร์ของตนเองเท่านั้น ไม่สามารถรับรู้ข้อมูลที่เก็บอยู่บนคลัสเตอร์คอมพิวเตอร์เครื่องอื่นได้
2) ไม่มีฟังก์ชันตรวจสอบความต้องกันของไฟล์ต้นฉบับกับฉบับของไฟล์ที่ถูกทำซ้ำ ดังนั้นไฟล์ที่จะทำซ้ำควรเป็นไฟล์ประเภทอ่านอย่างเดียว (Read-Only File) หากมีการแก้ไขไฟล์ที่มีการทำซ้ำไว้แล้ว ผู้ใช้จะต้องจัดการความต้องกันของไฟล์แต่ละฉบับด้วยตัวเอง เช่น อาจจะสอบถามระบบว่าฉบับอื่นๆ ของไฟล์เก็บอยู่ที่เครื่องให้บริการไฟล์เครื่องใดบ้าง แล้วตามไปแก้ไขทุกฉบับให้เหมือนกัน หรืออาจจะสั่งลบฉบับอื่นๆ ของไฟล์ให้หมด แล้วเก็บเฉพาะฉบับที่เพิ่งแก้ไขล่าสุด (scfs-retain) เป็นต้น

6.3 ข้อเสนอแนะและแนวทางการวิจัยต่อ

1) ส่วนโปรแกรมส่งงานและโปรแกรมผู้จัดการงาน สามารถทำวิจัยต่อได้เพิ่มความสามารถให้ตัวโปรแกรมสามารถรับอินพุตไฟล์ได้หลายไฟล์ และเพิ่มอัลกอลิทึมเพื่อตัดสินใจได้ว่าหากไฟล์ที่ต้องใช้ในการทำงานอยู่ต่างเครื่องกันจะส่งงานไปทำที่เครื่องใด และจะใช้วิธีใดในการเข้าถึงไฟล์ที่อยู่บนเครื่องอื่นๆ นอกจากนี้ยังสามารถทำวิจัยเพื่อให้ระบบสามารถรอบรับการทำงานของโปรแกรมแบบขนานได้ โดยใช้วิธีการส่งแต่ละโปรเซสของโปรแกรมแบบขนานไปยังเครื่องคำนวณที่มีฉบับของไฟล์ที่ถูกทำซ้ำเก็บอยู่

2) ในส่วนของการเคลื่อนย้ายไฟล์ข้ามคลัสเตอร์คอมพิวเตอร์ สามารถทำวิจัยต่อได้ในเรื่องการทำเป็นท่อข้อมูล (Pipe) แทนการสร้างไฟล์ชั่วคราว หรือการหาอัลกอลิทึมในการจัดการเรื่องความผิดพลาดที่อาจจะเกิดจากการรับส่งข้อมูลในขั้นตอนต่างๆ

3) ในเรื่องการทำซ้ำไฟล์ สามารถทำวิจัยต่อได้โดยการออกแบบและพัฒนาอัลกอริทึมเพื่อตรวจสอบความต้องการของไฟล์ที่มีการทำซ้ำภายในระบบไฟล์คลัสเตอร์ และเพิ่มฟังก์ชันเพื่อให้สามารถทำซ้ำไฟล์ในระดับองค์กรเสมือนได้

4) ในเรื่องของการพ่วง SCFS เข้ากับระบบปฏิบัติการลินุกซ์ สามารถทำได้ด้วยการใช้ FUSE [42] และ FUSE-J [43] ซึ่งในงานวิจัยนี้ได้พัฒนาระบบส่วนนี้ขึ้นมาแล้วแต่การทำงานยังไม่ถูกต้องทั้งหมด โดยเฉพาะในกรณีที่มีการเขียนและอ่านไฟล์ต่อกันทันที ซึ่งปัญหาที่เกิดขึ้นคือข้อมูลที่อ่านมาได้หลังจากการเขียนไม่มีการปรับให้เป็นปัจจุบัน

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

รายการอ้างอิง

[1] M. Li and M. Baker, The Grid Core Technologies, John Wiley & Sons Ltd.

[2] I. Foster, C. Kesselman, J. Nick and S. Tuecke, The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, 2002, pp.

[3] I. Foster, C. Kesselman and S. Tuecke, The Anatomy of the Grid: Enabling Scalable Virtual Organizations, International J. Supercomputer Applications, 2001.

[4] S. Venugopal, R. Buyya and K. Ramamohanarao, A Taxonomy of Data Grids for Distributed Data Sharing, Management and Processing, ACM Computing Surveys (CSUR), 2006.

[5] A. Chervenak, I. Foster, C. Kesselman, C. Salisbury and S. Tuecke, The Data Grid: Towards an architecture for the distributed management and analysis of large scientific datasets, Journal of Network and Computer Applications, 1999.

[6] R. W. Moore, M. Wan and A. Rajasekar, Storage resource broker; generic software infrastructure for managing globally distributed data, Local to Global Data Interoperability - Challenges and Technologies, 2005, pp. 65-69.

[7] O. Tatebe, Y. Morita, S. Matsuoka, N. Soda and S. Sekiguchi, Grid Datafarm Architecture for Petascale Data Intensive Computing, Proceedings of the 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID), 2002, pp. 102-102.

[8] Network File System Version 4 (nfsv4) [Online], http://www.ietf.org/html.charters/nfsv4-charter.html.

[9] D. C. Schmidt and F. Buschmann, Patterns, frameworks, and middleware: their synergistic relationships, Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, 2003, pp. 694-704.

[10] NPACI Rocks Clusters [Computer program], http://www.rocksclusters.org.

[11] H. Jin, X. Jin, W. Sining, L. Yi and M. Dan, DCFS: file service in commodity cluster dawning4000, Parallel and Distributed Computing, Applications and Technologies, 2003, pp. 80-84.

[12] X. Jin, T. Rongfeng, F. Zhihua, M. Jie, L. Hui, M. Dan, S. Ninghui and L. Guojie, A storage space management policy for a cluster file system, Proceedings of the Eighth

International Conference on High-Performance Computing in Asia-Pacific Region (HPCASIA'05), 2005, pp. 6 pp.

[13] F. Schmuck and R. Haskin, GPFS: A Shared-Disk File System for Large Computing Clusters, Proceedings of the 1st USENIX Conference on File and Storage Technologies, Monterey, CA, 2002.

[14] P. H. Carns, W. B. L. III, R. B. Ross and R. Thakur, PVFS: A Parallel File System For Linux Clusters, Proceedings of the 4th Annual Linux Showcase and Conference, Atlanta, GA, 2000, pp. 317-327.

[15] b. Latham, N. Miller, R. Ross and P. Carns, A Next-Generation Parallel File System for Linux Clusters. LinuxWorld Magazine (2004): 56 – 59.

[16] S. Jiwu, W. Bing, M. Weimin and D. Yiyan, Policy of file migration at server in cluster file system, Proceedings of the 2004 IEEE International Symposium on Cluster Computing and the Grid (CCGRID), 2004, pp. 691-698.

[17] G. v. Laszewski, B. Alunkal, J. Gawor, R. Madhuri, P. Plaszczak and X.-H. Sun, A File Transfer Component for Grids, The 2003 International Conference on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA, 2003.

[18] J. Novotny, M. Russell and O. Wehrens, GridSphere: an advanced portal framework, Euromicro Conference, 2004, pp. 412-419.

[19] NFS: Network File System Protocol Specification (NFS v.2) [Online], http://tools.ietf.org /html/rfc1094

[20] C. A. Thekkath, T. Mann and E. K. Lee, Frangipani: a scalable distributed file system, Proceedings of the sixteenth ACM symposium on Operating systems principles, Saint Malo, France, 1997, pp. 224-237.

[21] Q. Huang, W. Zheng and M. Shen, TH-CluFS: an open platform cluster file system, Fifth International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP'02), 2002, pp. 48 - 51.

[22] I. Cluster File Systems, Lustre [Computer program], http://www.lustre.org, 2006.

[23] San Diego Supercomputer Center (SDSC), Storage Resource Broker (SRB) [Computer program], http://www.sdsc.edu/srb/index.php/Main_Page (SRB WEB)

[24] A. Rajasekar, MCAT - A Meta Information Catalog (Version 1.1) [Computer program], http://www.npaci.edu/DICE/SRB/mcat.html, 1998.

[25] O. Tatebe, Grid Data Farm (Gfarm) [Computer program], http://datafarm.apgrid.org

[26] B. S. White, M. Walker, M. Humphrey and A. S. Grimshaw, LegionFS: A Secure and Scalable File System Supporting Cross-Domain High-Performance Applications, Supercomputing, ACM/IEEE 2001 Conference, 2001, pp. 19-29.

[27] J. Bent, V. Venkataramani, N. LeRoy, A. Roy, J. Stanley, A. C. Arpaci-Dusseau, R. H. Arpaci-Dusseau and M. Livny, Flexibility, manageability, and performance in a Grid storage appliance, High Performance Distributed Computing (HPDC), 2002, pp. 3-12.

[28] G. v. Laszewski and et al., Java CoG Kit [Computer software], http://wiki.cogkit.org /index.php/Java_CoG_Kit. (WEB)

[29] I. Foster and C. Kesselman, The Globus project: a status report, Heterogeneous Computing Workshop (HCW 98) Proceedings. 1998 Seventh, 1998, pp. 4-18.

[30] T. Kooburat and V. Muangsin, The Experience in Deploying MultipleVirtual Organizations across Grid Community, The 11th Annual NationalSymposium on Computational Science and Engineering, Phuket, 2007.

[31] L. Mortenson, R. Shaw and J. Sorlin, Java Service Wrapper [Computer program], http://wrapper.tanukisoftware.org.

[32] M. Grand, Patterns in Java: a catalog of reusable design patterns illustrated with UML, Robert Ipsen, 1998.

[33] J. Strachan, B. McWhirter, J. Keyes and R. Oxspring, Apache Commons CLI [Computer software], http://commons.apache.org, 2007.

[34] Apahce Log4j [Computer software], http://logging.apache.org, 2006.

[35] Sun Grid Engine (SGE) [Computer program], http://gridengine.sunsource.net/

[36] B. Sotomayor, The Globus Toolkit 4 Programmer's Tutorial [Online], http://gdp.globus.org/gt4-tutorial, 2005.

[37] Globus [Computer software], http://www.globus.org/

[38] C. Bauer and G. King, Hibernate in Action, Manning Publications Co., 2005.

[39] iperf [Computer program], http://www.noc.ucf.edu/Tools/Iperf/default.htm

[40] Disaster Control Research Center of Tohoku University, TUNAMI (Tohoku University's Numerical Analysis Model for Investigation of near field tsunamis) [Computer program].

[41] G. Abram, P. Kirchner, D. Thompson and M. Tignor, OpenDX [Computer program], http://www.opendx.org.

[42] C. Henk, M. Szeredi, D. Pavlinusic, R. Dawe and S. Delafond, Filesystem in Userspace (FUSE) [Computer Software], http://fuse.sourceforge.net/, 2007.

[43] P. Levart, FUSE-J [Computer Software], http://sourceforge.net/projects/fuse-j

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก

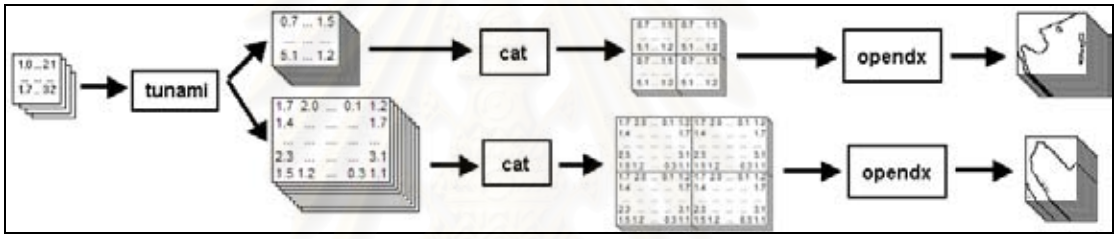
สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ก.

โปรแกรมจำลองการเกิดคลื่นสึนามิ

โปรแกรมจำลองการเกิดคลื่นสึนามิเป็นการทดลองที่ใช้โมเดลทางคณิตศาสตร์ในการจำลองลักษณะและคุณสมบัติของคลื่นสึนามิที่เกิดขึ้นในช่วงเวลาต่างๆ หลังจากเกิดแผ่นดินไหวในทะเล กรณีที่นำมาใช้ในการทดลองของงานวิจัยนี้เป็นกรณีที่เกิดแผ่นดินไหวบริเวณฝั่งตะวันตกของประเทศฟิลิปปินส์ และศึกษาผลของคลื่นที่เคลื่อนที่เข้ามาทางอ่าวไทย

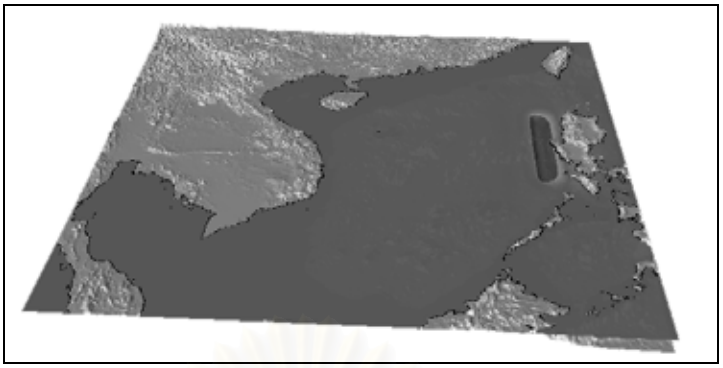
การจำลองการเกิดคลื่นสึนามิที่ใช้ในการทดลองของงานวิจัยนี้มีลักษณะเป็นกระแสงานที่ประกอบด้วย 3 ขั้นตอนได้แก่ การจำลองเหตุการณ์ด้วยโมเดลทางคณิตศาสตร์, การต่อไฟล์ระหว่างกลาง, และการทำจินตทัศน์



รูปที่ ก.1: แสดงขั้นตอนที่ใช้ในการจำลองการเกิดคลื่นสึนามิ

ขั้นตอนแรกเป็นการจำลองเหตุการณ์ด้วยโมเดลทางคณิตศาสตร์โดยใช้โปรแกรมชื่อ 'TUNAMI' [40] โปรแกรมจะรับอินพุตเป็นไฟล์ที่เก็บข้อมูลเมทริกซ์ของเลขจำนวนจริง 4 ไฟล์ซึ่งได้แก่ลักษณะของพื้นที่ที่ทำการจำลองและลักษณะของน้ำที่เปลี่ยนรูปไปเมื่อเกิดแผ่นดินไหวอย่างละ 2 ไฟล์ ไฟล์ชุดหนึ่งสำหรับมาตราส่วนอย่างหยาบ อีกชุดหนึ่งสำหรับมาตราส่วนอย่างละเอียด ขนาดของเมทริกซ์จะเท่ากับขนาดของพื้นที่ที่ทำการจำลองในแต่ละมาตราส่วน ในการประมวลผลโปรแกรมจะจำลองเหตุการณ์ของคลื่นในเวลาต่างๆ ตั้งแต่เริ่มเกิดคลื่น และสร้างไฟล์เอาท์พุตที่แสดงลักษณะของคลื่นในแต่ละเวลาขึ้นมา ดังนั้นเมื่อโปรแกรมทำงานเสร็จผลลัพธ์ที่ได้จะเป็นไฟล์เมทริกซ์ของเลขจำนวนจริงขนาดเล็กหลายๆ ไฟล์ (ขึ้นกับเวลาที่ทำการจำลองเหตุการณ์) แต่ละไฟล์เก็บข้อมูลของคลื่นในเวลาหนึ่ง

ขั้นตอนที่สองเป็นการนำไฟล์เอาท์พุตขนาดเล็กทั้ง 400 ไฟล์มาต่อกันให้ได้เป็นไฟล์ขนาดใหญ่ที่เก็บข้อมูลของคลื่นในหลายๆ เวลาจำนวน 11 ไฟล์ โดยใช้คำสั่ง 'cat' ของระบบปฏิบัติการลินุกซ์ ขั้นตอนนี้จะเน้นในเรื่องการอ่านเขียนไฟล์โดยเฉพาะ ส่วนขั้นตอนสุดท้ายเป็นการทำจินตทัศน์ โดยนำไฟล์เอาท์พุตทั้ง 11 ไฟล์จากขั้นตอน concat ที่เก็บข้อมูลเมทริกซ์ที่แสดงรูปร่างของคลื่นมาสร้างเป็นรูปภาพด้วยโปรแกรมชื่อ 'OpenDX' [41] ตัวอย่างของผลลัพธ์ที่ได้แสดงให้เห็นในรูปที่ ก.2



รูปที่ ก.2: ตัวอย่างผลลัพธ์ของโปรแกรมการจำลองการเกิดคลื่นสึนามิ

สาเหตุที่ต้องมีการต่อไฟล์เนื่องจากโปรแกรมที่ใช้ทำจินตทัศน์ที่สร้างขึ้นด้วย OpenDX นั้น สามารถรับไฟล์อินพุตได้เพียงทีละหนึ่งไฟล์ต่อการทำจิตทัศน์หนึ่งครั้ง ดังนั้นหากใช้ไฟล์อินพุตหลายไฟล์ ก็จะเสียเวลาในการเริ่มต้นโปรแกรมใหม่หลายครั้ง จากการทดลองเบื้องต้นเวลาที่ใช้สำหรับการทำจิตทัศน์ในกรณีที่ไม่มีการต่อไฟล์สูงกว่ากรณีที่มีการต่อไฟล์ถึงเกือบสองเท่า อย่างไรก็ตามเนื่องจากโปรแกรม OpenDX ใช้หน่วยความจำปริมาณมากในการทำงาน ดังนั้นจึงต้องมีการกำหนดขนาดของไฟล์ผลลัพธ์ที่ได้จากการต่อไฟล์ ซึ่งในการทดลองในงานวิจัยนี้ได้กำหนดให้แต่ละไฟล์มีขนาดประมาณ 600 MB เท่านั้น (เครื่องที่ใช้ทำการทดลองมีหน่วยความจำ 1 GB)

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ข.

การติดตั้งและการใช้งาน SCFS

ข.1 ขั้นตอนการติดตั้ง

1) ขยายไฟล์ ‘scfs.tar.gz’

```
~ # tar –xzf scfs.tar.gz
```

2) ตั้งค่าตัวแปรสภาพแวดล้อมชื่อ “SCFS_HOME” ไปที่พาทของ SCFS เช่น

```
~ # export SCFS_HOME=/opt/scfs
```

3) สำหรับเครื่องที่จะติดตั้งเป็นเครื่องให้บริการเมตะดาต้า ให้แก้ไขไฟล์การตั้งค่าสำหรับการติดต่อกับระบบฐานข้อมูล MySQLโดยกำหนดค่า username และ password ของไฟล์ให้ตรงกัน ไฟล์ที่ต้องทำการแก้ไขค่า ได้แก่

- scfs/config/meta-db-conf.xml
- scfs/scfs-table.sql
- scfs/scfs-user.sql

4) ใช้สคริปต์ (Script) สำหรับติดตั้งโปรแกรม

```
~ # cd scfs; ./install_scfs.sh <meta|file> <i386|x86_64>
```

พารามิเตอร์ตัวแรกเป็นการเลือกว่าจะติดตั้งเป็นเครื่องให้บริการเมตะดาต้าหรือเครื่องให้บริการไฟล์ พารามิเตอร์ตัวที่สองเป็นการเลือกสถาปัตยกรรมของเครื่องว่าเป็นแบบ i386 หรือ x86_64

5) ใช้โปรแกรมช่วยสร้างไฟล์การตั้งค่าของระบบ

```
~ # java -cp $SCFS_HOME/lib/scfs.jar space.scfs.tool.GenConfig
Metadata Server Name: master
Data server amount: 4
Insert name of the Data server. Range number can be represent as [begin:end]
Data server Name: compute-0-[0:3]
Message port (59113):
Storage dir: /state/partition1/scfs-storage
This is the input value
Metadata Server: Master_Node
Data Server 0: compute-0-0
```

```
Data Server 1: compute-0-1
Data Server 2: compute-0-2
Data Server 3: compute-0-3
Message Port: 59113
Storage: /state/partition1/scfs-storage
Is it ok? (y:n): y
```

6) แก้ไขข้อมูลที่ใช้สร้างล็อกในไฟล์ '$SCFS_HOME/config/log4j.properties'

ข.2 การเปิด - ปิดระบบ

1) การเปิดระบบ

```
~ # $SCFS_HOME/sbin/scfs console
```

2) การปิดระบบ

```
~ # $SCFS_HOME/sbin/scfs stop
```

3) การสอบถามสถานะของระบบ

```
~ # $SCFS_HOME/sbin/scfs status
```

ข.3 บรรทัดคำสั่ง

1) **scfs-chgrp** : เปลี่ยนกลุ่มของไฟล์

สิทธิการใช้งาน : Root ของระบบ

การใช้งาน : scfs-chgrp [-r] <group> <scfs-path>

-r : เปลี่ยนกลุ่มของไฟล์ในไดเรกทอรีแบบเรียกซ้ำ

ตัวอย่างการใช้งาน

- การเปลี่ยนกลุ่มของไดเรกทอรีชื่อ '/scfs/nipat' และไฟล์ภายในให้เป็นกลุ่ม 'users'

```
~ # scfs-chgrp –r users /scfs/nipat
```

2) **scfs-chhost** : เปลี่ยนเครื่องที่ทำหน้าที่เก็บไฟล์

สิทธิการใช้งาน : ต้องมีสิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-chhost [-r] <host> <scfs-path>

-r : เปลี่ยนเครื่องที่ทำหน้าที่เก็บไฟล์ในไดเรกทอรีแบบเรียกซ้ำ

ตัวอย่างการใช้งาน

- เปลี่ยนเครื่องที่ทำหน้าที่เก็บไฟล์ที่อยู่ในไดเรกทอรี'/scfs/nipat/dir1' ไปเป็นเครื่องที่ชื่อว่า 'compute-0-1'

```
~ # scfs-chhost –r compute-0-1 /scfs/nipat/dir1
```

- เปลี่ยนเครื่องที่ทำหน้าที่เก็บไฟล์ที่มีชื่อตามรูปแบบ ‘/scfs/nipat/dir2/file*’ ไปเป็นเครื่องที่ชื่อว่า ‘compute-0-3’

```
~ # scfs-chhost –r compute-0-3 /scfs/nipat/dir2/file*
```

3) **scfs-chmod** : เปลี่ยนโหมด (Mode) ของไฟล์

สิทธิการใช้งาน : เจ้าของไฟล์

การใช้งาน : scfs-chmod [-r] <mode> <scfs-path>

-r : เปลี่ยน Mode ของไฟล์ในไดเรกทอรีแบบเรียกซ้ำ

ตัวอย่างการใช้งาน

- เปลี่ยน Mode ของไฟล์ที่มีชื่อตรงรูปแบบ ‘/scfs/nipat/dir3/file*.bin’ เป็น 755

```
~ # scfs-chmod 755 /scfs/nipat/dir3/file*.bin
```

4) **scfs-chown** : เปลี่ยนเจ้าของไฟล์

สิทธิการใช้งาน : Root ของระบบ

การใช้งาน : scfs-chown [-r] < <owner[:group]> > <scfs-path>

-r : เปลี่ยนเจ้าของไฟล์ในไดเรกทอรีแบบเรียกซ้ำ

ตัวอย่างการใช้งาน

- เปลี่ยนเจ้าของไฟล์ในไดเรกทอรี‘/scfs/nipat’ เป็น ‘nipat’

```
~ # scfs-chown –r nipat /scfs/nipat
```

5) **scfs-cp** : ทำสำเนาไฟล์

สิทธิการใช้งาน : สิทธิ์ในการอ่านไฟล์ต้นฉบับ และสิทธิ์ในการเขียนไดเรกทอรีแม่ของไดเรกทอรีปลายทาง

การใช้งาน : scfs-cp [-rh <host>] <scfs-src-path> <scfs-dest-path>

-r : ทำสำเนาไฟล์ในไดเรกทอรีต้นฉบับแบบเรียกซ้ำ

-h <Host> : กำหนดเครื่องปลายทาง หากไม่มีการกำหนดเครื่องปลายทาง ระบบจะเลือกเครื่องที่เหมาะสมให้

ตัวอย่างการใช้งาน

- ทำสำเนาไฟล์ในไดเรกทอรีชื่อ ‘/scfs/nipat/dir1’ ไปเก็บในไดเรกทอรีชื่อ ‘/scfs/nipat/dir2’

```
~ # scfs-cp –r /scfs/nipat/dir1 /scfs/nipat/dir2
```

6) **scfs-export** : ส่งออกไฟล์จาก SCFS ไป Linux.

สิทธิการใช้งาน : สิทธิ์ในการอ่านไฟล์ต้นฉบับ และสิทธิ์ในการเขียนไดเรกทอรีแม่ของไดเรกทอรีปลายทาง

การใช้งาน : scfs-export [-rh <host>] <scfs-src-path> <linux-dest-path>

-r : ส่งออกไฟล์ในไดเรกทอรีต้นฉบับแบบเรียกซ้ำ

-h <Host> : เครื่องปลายทาง หากไม่กำหนดระบบจะส่งออกไปที่เครื่องท้องถิ่น

ตัวอย่างการใช้งาน

- ส่งออกไฟล์ในไดเรกทอรีชื่อ '/scfs/nipat/dir1' ไปเก็บในไดเรกทอรีชื่อ '/tmp' บนเครื่องท้องถิ่น

```
~ # scfs-export –r /scfs/nipat/dir1 /tmp
```

7) scfs-query : แสดงข้อมูลของไฟล์

สิทธิการใช้งาน : สิทธิในการอ่านไดเรกทอรีแม่ของไฟล์

การใช้งาน : scfs-query <-aplLsmughi <index>> <scfs-path>

-a : แสดง Attribute ทั้งหมดของไฟล์

-p : แสดงพาทของไฟล์

-l : ถ้าไฟล์เป็นลิงค์ ให้แสดงพาธของไฟล์จริงๆ

-L : ถ้าไฟล์เป็นลิงค์ ให้แสดงไดเรกทอรีแม่ของไฟล์จริงๆ

-s : แสดงขนาดของไฟล์

-m : แสดงโหมดของไฟล์

-u : แสดงเจ้าของไฟล์

-g : แสดงกลุ่มของไฟล์

-h : แสดงชื่อของเครื่องที่เก็บไฟล์

-i <index> : กำหนดหมายเลขของ Instance ของไฟล์ ที่ใช้ในการค้นหา เริ่มจากหมายเลข 1 (หมายเลข 0 หมายถึงทุกๆ Instance)

หมายเหตุ : สามารถแสดงข้อมูลของไฟล์แค่ 1 ไฟล์ในแต่ละครั้งของคำสั่ง

ตัวอย่างการใช้งาน

- แสดงเส้นทางของไฟล์ชื่อ 'fileA'

```
~ # scfs-query –p /scfs/nipat/dir1/fileA
```

- แสดงชื่อเครื่องที่เก็บ Instance ของไฟล์ชื่อ 'fileA'

```
~ # scfs-query –h /scfs/nipat/dir1/fileA
```

8) scfs-ls : แสดงรายการไฟล์

สิทธิการใช้งาน : สิทธิในการอ่านไดเรกทอรีแม่ของไฟล์ที่ถูกแสดงในรายการ

การใช้งาน : scfs-ls [-hald] <scfs-path>

-a : แสดงทุกไฟล์ รวมไฟล์ที่ถูกซ่อนด้วย

-h : แสดงชื่อเครื่องที่ทำหน้าที่เก็บไฟล์ด้วย

-d : แสดงข้อมูลของไดเรกทอรีนั้นแทนที่จะแสดงข้อมูลมูลของลูก

-l : แสดง Attribute ของไฟล์

ตัวอย่างการใช้งาน

- แสดงรายการไฟล์ในไดเรกทอรีชื่อ '/scfs/nipat'

```
~ # scfs-ls /scfs/nipat
```

- แสดงรายการไฟล์ชื่อ '/scfs/nipat/dir1'

```
~ # scfs-ls –d /scfs/nipat/dir1
```

- แสดงรายการไฟล์และ Attribute ของไฟล์ในไดเรกทอรีชื่อ '/scfs/nipat/dir1'

```
~ # scfs-ls –l /scfs/nipat/dir1
```

9) scfs-mkdir : สร้างไดเรกทอรี

สิทธิการใช้งาน : สิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-mkdir <scfs-path>

ตัวอย่างการใช้งาน

- สร้างไดเรกทอรีชื่อ '/scfs/nipat/dir2'

```
~ # scfs-mkdir /scfs/nipat/dir2
```

10) scfs-mv : เคลื่อนย้ายไฟล์ภายในเครื่อง

สิทธิการใช้งาน : สิทธิ์ในการเขียนไดเรกทอรีแม่ของไฟล์ต้นฉบับและไดเรกทอรีปลายทาง

การใช้งาน : scfs-mv <scfs-src-path> <scfs-dest-path>

ตัวอย่างการใช้งาน

- ย้ายไฟล์จาก /scfs/nipat/dir1 ไป /scfs/nipat/dir2

```
~ # scfs-mv /scfs/nipat/dir1/* /scfs/nipat/dir2
```

11) scfs-reg : ลงทะเบียนไฟล์จากระบบปฏิบัติการลินุกซ์เข้าสู่ SCFS

สิทธิการใช้งาน : สิทธิ์ในการอ่านไฟล์ต้นฉบับ และสิทธิ์ในการเขียนไดเรกทอรีแม่ของไดเรกทอรีปลายทาง

การใช้งาน : scfs-reg [-rch <host>] <linux-file> <scfs-file>

-r : ลงทะเบียนไฟล์ในไดเรกทอรีแบบเรียกซ้ำ

-c : ทำสำเนาไฟล์ต้นฉบับเข้าไปเก็บในระบบแทนการสร้างตัวเชื่อมโยง

-h <host> : กำหนดเครื่องที่จะทำหน้าที่เก็บไฟล์ หากไม่กำหนดจะเก็บลงในเครื่องท้องถิ่น

ตัวอย่างการใช้งาน

- ลงทะเบียนไดเรกทอรีชื่อ '/home/nipat/linuxdir1' เข้าสู่ไดเรกทอรีชื่อ '/scfs/nipat' บนเครื่อง 'compute-0-3' ด้วยวิธีการทำสำเนาไฟล์

```
Master: ~ # scfs-reg –rch compute-0-3 /home/nipat/linuxdir1 /scfs/nipat
```

- ลงทะเบียนไฟล์ ‘/tmp/file1’ บนเครื่อง ‘compute-0-0’ เข้าสู่ระบบในไดเรกทอรีชื่อ ‘/scfs/nipat/dir1’ ด้วยวิธีการสร้างตัวเชื่อมโยง

```
Master: ~ # ssh compute-0-0
compute-0-0: ~ # scfs-reg /tmp/file1 /scfs/nipat
```

12) **scfs-rep** : ทำซ้ำไฟล์

สิทธิการใช้งาน : สิทธิ์ในการอ่านไฟล์ต้นฉบับ และสิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-rep <host>+ <scfs-path>

<host>+ : เครื่องที่ทำหน้าที่เก็บ Replica ของไฟล์ สามารถระบุได้หลายเครื่อง แต่ละเครื่องแยกด้วยช่องว่าง

หมายเหตุ : ในการเรียกแต่ละครั้งสามารถสร้างได้ทีละไฟล์แต่กี่ฉบับก็ได้ (ไม่เกินจำนวนเครื่องให้บริการไฟล์ที่มีอยู่ในระบบ)

ตัวอย่างการใช้งาน

- ทำซ้ำไฟล์ ‘/scfs/nipat/fileB’ ไปที่เครื่อง ‘compute-0-3’ และ ‘compute-0-2’

```
~ # scfs-rep compute-0-3 compute-0-2 /scfs/nipat/fileB
```

13) **scfs-rm** : ลบไฟล์

สิทธิการใช้งาน : สิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-rm [-r] <scfs-path>

-r : ลบไฟล์ในไดเรกทอรีแบบเรียกซ้ำ

ตัวอย่างการใช้งาน

- ลบไฟล์ที่มีพาทตรงตามแบบ ‘/scfs/nipat/dir1/file*.dat’

```
~ # scfs-rm /scfs/nipat/dir1/file*.dat
```

- ลบไฟล์ในไดเรกทอรีที่มีพาทตรงตามแบบ ‘/scfs/nipat/dir*

```
~ # scfs-rm –r /scfs/nipat/dir*
```

14) **scfs-rmdir** : ลบไดเรกทอรีที่ว่างเปล่า

สิทธิการใช้งาน : สิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-rmdir <scfs-path>

ตัวอย่างการใช้งาน

- ลบไดเรกทอรีที่ตรงตามแบบ ‘/scfs/nipat/dir1*’

```
~ # scfs-rmdir /scfs/nipat/dir1*
```

15) **scfs-rmrep** : ลบไฟล์ฉบับที่มีการทำซ้ำ

สิทธิการใช้งาน : สิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-rmrep [host]+ <scfs-path>

- [host]+ : เครื่องที่เก็บฉบับของไฟล์ที่ต้องการลบ สามารถระบุได้หลายเครื่อง แต่ละเครื่องแยกกันด้วยช่องว่าง

หมายเหตุ : ในการเรียกใช้คำสั่งแต่ละครั้งสามารถลบไฟล์ได้แค่ทีละหนึ่งไฟล์แต่หลายฉบับ (อย่างน้อยต้องเหลือไฟล์ 1 ฉบับ)

ตัวอย่างการใช้งาน

- ลบไฟล์ '/scfs/nipat/fileA' ฉบับที่อยู่บนเครื่อง 'compute-0-0' และ 'compute-0-1'

```
~ # scfs-rmrep compute-0-0 compute-0-1 /scfs/nipat/fileA
```

16) **scfs-retain** : เก็บเฉพาะ 1 ฉบับของไฟล์ แล้วลบฉบับอื่นให้หมด

สิทธิการใช้งาน : สิทธิ์ในการเขียนไดเรกทอรีแม่

การใช้งาน : scfs-retain [-h <Host>] <scfs-path>

-h <host> : กำหนดเครื่องที่จะทำหน้าที่เก็บไฟล์

หมายเหตุ : สามารถระบุไฟล์ได้เพียงหนึ่งไฟล์ในการเรียกใช้คำสั่งแต่ละครั้ง

ตัวอย่างการใช้งาน

- ลบทุกๆ ฉบับของไฟล์ '/scfs/nipat/file1' ยกเว้นฉบับที่เก็บอยู่บนเครื่อง 'compute-0-2'

```
~ # scfs-retain –h compute-0-2 /scfs/nipat/file1
```

ข.4 API ของ SCFS

API สำหรับการจัดการไฟล์ของ SCFS สร้างไว้เป็นคลาสชื่อ "ScfsProcessAPI" ฟังก์ชันที่เปิดให้ใช้งานมีลักษณะเป็นแบบ static คือสามารถเรียกใช้งานได้โดยไม่ต้องมีการสร้าง*วัตถุ*ของคลาสนี้ ฟังก์ชันที่มีให้เรียกใช้งานได้มีดังนี้

```
public class space.scfs.api.ScfsProcessAPI

- public static int list(List result, String path, boolean isAll, boolean isLong, boolean
        isGetHost, boolean isDir, boolean isMakeLineResult);
- public static int query(List result, String path, String delimeter, boolean getHost, boolean
        getPhyPath, boolean getLink, boolean getLinkParent, boolean getSize,
        boolean getMode, boolean getUser, boolean getGroup);
- public static int changeMode(String path, String mode, boolean recursive);
- public static int changeOwn(String path, String user, String group, boolean recursive);
```

- public static int changeHost(String path, String srcHost, String dstHost, boolean recursive);
- public static int changeGroup(String path, String group, boolean recursive);
- public static int copy(String srcPath, String dstPath, String dstHost, boolean recursive);
- public static int export(String srcPath, String dstPath, String dstHost, boolean recursive);
- public static int lookup(List result, String path, boolean isGetAttribute);
- public static int makeEmptyFile(String path, String dstHost, int mode);
- public static int makeDir(String path, int mode);
- public static int move(String srcPath, String dstPath);
- public static int register(String[] srcPaths, String dstPath, String dstHost, boolean isCopyMethod, boolean recursive);
- public static int remove(String path, boolean recursive);
- public static int removeDir(String path);
- public static int replicate(String path, String[] hosts);
- public static int removeReplica(String path, String[] hosts);
- public static int retainReplica(String path, String host);

API สำหรับการเข้าถึงไฟล์ภายใน SCFS สร้างไฟล์เป็นคลาสชื่อ “IOConnection” การเรียกใช้งานจะต้องสร้าง*วัตถุ*ของ *IOConnection* ขึ้นมาก่อน โดย *IOConnection* หนึ่งอันจะใช้เข้าถึงไฟล์ได้เพียงหนึ่งไฟล์เท่านั้น ฟังก์ชันที่มีให้ใช้งานมีดังนี้

- public IOConnection(ScfsFileHandler fileHandler) throws SCFSException
- public synchronized void open() throws SCFSException
- public synchronized boolean read(ByteBuffer buffer, long offset) throws SCFSException
- public synchronized int read(byte[] buffer, long offset) throws SCFSException
- public synchronized int read(byte[] buffer) throws SCFSException
- public synchronized int read(byte[] buffer, int b_offset, int b_length) throws SCFSException
- public synchronized boolean write(ByteBuffer buffer, long offset) throws SCFSException
- public synchronized boolean write(byte[] data, long offset) throws SCFSException
- public synchronized boolean write(byte[] data) throws SCFSException
- public synchronized boolean write(byte[] data, int d_offset, int d_length) throws SCFSException

- public synchronized void seekToEoF() throws SCFSException
- public synchronized void seekToStart() throws SCFSException
- public synchronized void seek(long offset) throws SCFSException
- public synchronized void truncate(long size) throws SCFSException
- public synchronized void oneSideRelease()
- public synchronized void release()
- public synchronized void close()

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ภาคผนวก ค.

อินเตอร์เฟสของ CCSS

ค.1 API ของระบบส่วนใช้บริการบนเครื่องให้บริการ GHOSTS

API ของระบบส่วนใช้บริการบนเครื่องให้บริการ GHOSTS เป็น*วัตถุ*ของคลาสชื่อ “CcssApi” โดยมีฟังก์ชันให้เรียกใช้ดังนี้

public class space.ccss.ghosts.services.resources.CcssApi

- public CcssApi(CcssBrowserService broserService, User user);
- public CcssApi(CcssBrowserService ccssBrowserService, User user, ileBrowserService fileBrowserService) throws FileTaskException;
- public String getHome(String cfs) throws FileTaskException;
- public String makeHome(String cfs, String homeName) throws FileTaskException;
- public String getShare(String cfs) throws FileTaskException;
- public String makeShare(String cfs, String shareName, String voGroup) throws FileTaskException;
- public String getRoot(String cfs);
- public CcssLocation getRootLocation(String cfs) throws FileTaskException;
- public String query(String cfs, String question, String hostName) throws FileTaskException;
- public String[] query(String cfs, String question, List<String> hostList) throws FileTaskException;
- public CcssLocation info(String cfs, String path, String host);
- public List<CcssLocation> info(String cfs, String path, List<String> hostList);
- public boolean exist(String cfs, String path, String host);
- public boolean exists(String cfs, String path, List<String> hostList);
- public List list(String cfs, String listPath, List hostList, int listOption) throws FileTaskException;
- public List list(String cfs, FileLocation pathLocation, List hostList, int listOption) throws FileTaskException;

- public List list(CcssLocation pathLocation, List<String> hostList, int listOption) throws FileTaskException;
- public CcssMakeDir[] makeDir(String cfs, FileLocation pathLocation, List<String> hostList) throws FileTaskException;
- public CcssMakeDir[] makeDir(String cfs, String dirPath, List<String> hostList) throws FileTaskException;
- public CcssMakeDir[] makeDir(CcssLocation parentLocation, String newDirName, List<String> hostList) throws FileTaskException;
- public CcssRename[] rename(String cfs, String oldPath, String newName, List<String> hostList) throws FileTaskException;
- public CcssRename[] rename(CcssLocation oldLocation, String newName, List<String> hostList) throws FileTaskException;
- public CcssDelete[] delete(String cfs, FileLocation location, List<String> hostList) throws FileTaskException;
- public CcssDelete[] delete(Map<String, List<CcssLocation>> hostLocationMap) throws FileTaskException;
- public CcssCopy copy(CcssLocation src, CcssLocation dstParent, String dstNode) throws FileTaskException;
- public CcssCopy copy(FileSet src, CcssLocation dstParent, String dstNode) throws FileTaskException;
- public CcssMove move(CcssLocation src, String srcNode, CcssLocation dstParent, String dstNode) throws FileTaskException;
- public CcssMove move(FileSet src, String srcNode, CcssLocation dstParent, String dstNode) throws FileTaskException;
- public FileTransfer transfer(List<CcssLocation> srcLocList, String srcNode, CcssLocation dstParentLoc, String dstNode, String transferType) throws FileTaskException;
- public CcssUpload upload(String uploadedFilePath, CcssLocation dstLocation) throws FileTaskException;
- public CcssDownload download(CcssLocation src, String localFilePath) throws FileTaskException;
- public List getTaskStatusMsgs();

ค.2 อินเตอร์เฟสสำหรับการเชื่อมต่อระบบไฟล์คลัสเตอร์

ในส่วนนี้เป็นการแสดงอินเตอร์เฟสสำหรับให้ระบบไฟล์คลัสเตอร์นำไประบุการทำงานของตัวเอง โดยอินเตอร์เฟสและฟังก์ชันที่ระบบไฟล์คลัสเตอร์จะต้องนำไประบุการทำงานได้แก่

ค.2.1 อินเตอร์เฟสระบบไฟล์คลัสเตอร์

```
public interface space.ccss.cluster.cfs.ClusterFS
    - public String getName();
    - public String getDescription();
    - public boolean isParallel();
    - public String getNameSpace();
    - public CfsCmdFactory getCfsCmdFactory() throws CcssException;
    - public void initCfs(String name, String desc, boolean isParallel, HashMap params)
      throws CcssException;
public abstract class space.ccss.cluster.cfs.AbstractCFS implements ClusterFS
    - abstract protected void initCFS(HashMap params) throws CcssException;
public interface space.ccss.cluster.cfs.prop.HaveShareDir
    - public boolean makeVOShareDir(String voGroup, String dirPath) throws CmdException;
public interface space.ccss.cluster.cfs.prop.HaveUserHome
    - public boolean makeUserHome(User user, String dirPath) throws CmdException;
```

ค.2.2 อินเตอร์เฟสสำหรับวัตถุคำสั่ง

```
public abstract class space.ccss.cluster.cmd.CfsCmd
    - abstract protected boolean checkParams();
    - protected boolean preProcess() throws CcssException;
    - abstract protected boolean cmdProcess() throws CcssException;
    - protected boolean postProcess() throws CcssException;
    - abstract public String getName();
    - abstract public String getDesc();
```

นอกจาก *CfsCmd* ซึ่งเป็นอินเตอร์เฟสหลักแล้ว ยังมีอินเตอร์เฟสรองซึ่งขยายการทำงานมาจาก *CfsCmd* อีกทีหนึ่ง ผู้พัฒนาระบบจะต้องระบุการทำงานของอินเตอร์เฟสเหล่านี้ ซึ่งได้แก่

```
public abstract class space.ccss.cluster.cmd.CmdCopy extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdExport extends CfsCmd
```

```
public abstract class space.ccss.cluster.cmd.CmdList extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdMakeDir extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdMove extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdRegister extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdRemove extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdRemoveDir extends CfsCmd
public abstract class space.ccss.cluster.cmd.CmdRename extends CfsCmd
```

ค.2.3 อินเตอร์เฟสการสร้างวัตถุคำสั่ง

```
public class space.ccss.cluster.cmd.def.DefaultCmdFactory implements CfsCmdFactory
  - public CmdList createCmdList(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdRemove createCmdRemove(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdMakeDir createCmdMakeDir(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdRemoveDir createCmdRemoveDir(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdMove createCmdMove(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdCopy createCmdCopy(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdRegister createCmdRegister(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdExport createCmdExport(User user, Map params, ClusterFS cfs)
        throws CcssException
  - public CmdRename createCmdRename(User user, Map params, ClusterFS cfs)
        throws CcssException
```

ค.3 API สำหรับการใช้งานไลบรารีผู้ใช้บริการ

API สำหรับไลบรารีผู้ใช้บริการของ CCSS เป็น*วัตถุ*ของคลาสชื่อ “CcssClient” ซึ่งมีฟังก์ชันให้เรียกใช้ดังนี้

public class space.ccss.client.CcssClient

- public CcssClient(String hostName) throws CcssException
- public CcssClient(String hostName, ClientSecurityDescriptor descriptor) throws CcssException
- public static void initCcssClient() throws CcssException
- public void setPort(int port)
- public int getPort()
- public void setProtocol(String protocol)
- public String getProtocol()
- public CcssResponse query(String cfs, String question) throws CcssException
- public CcssResponse list(String cfs, String listPath, int listOption) throws CcssException
- public CcssResponse copy(String cfs, String srcPath, String dstNode, String dstPath) throws CcssException
- public CcssResponse export(String cfs, String srcPath, String dstNode, String dstPath) throws CcssException
- public CcssResponse register(String cfs, String srcPath, String dstNode, String dstPath) throws CcssException
- public CcssResponse mkdir(String cfs, String dirPath) throws CcssException
- public CcssResponse remove(String cfs, String srcPath) throws CcssException
- public CcssResponse rename(String cfs, String oldPath, String newName) throws CcssException
- public CcssResponse move(String cfs, String srcNode, String srcPath, String dstNode, String dstPath) throws CcssException
- public CcssResponse mkhome(String cfs, String dirPath) throws CcssException
- public CcssResponse mkshare(String cfs, String dirPath, String voGroup) throws CcssException

ข้อมูลผลลัพธ์และสถานะการทำงานจะถูกเก็บอยู่ใน*วัตถุ*ของคลาสที่มีชื่อว่า “CcssResponse” ซึ่งมีฟังก์ชันให้เรียกใช้ดังนี้

public class space.ccss.client.CcssResponse

- public String getCfs();
- public String getCmd();

- public String getHost();
- public String getListPath();
- public String getMsg();
- public List getResults();
- public int getStatus();

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย

ประวัติผู้เขียนวิทยานิพนธ์

นายนิภัทร์ ลีละปัญญา เกิดวันที่ 21 กันยายน พ.ศ. 2526 ในจังหวัดกรุงเทพมหานคร สำเร็จการศึกษาปริญญาวิศวกรรมศาสตรบัณฑิต (เกียรตินิยมอันดับหนึ่ง) สาขาวิชาวิศวกรรมคอมพิวเตอร์ จากคณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2548 และเข้าศึกษาในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต สาขาวิชาวิศวกรรมคอมพิวเตอร์ ณ ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย เมื่อปีการศึกษา 2549 งานวิจัยที่อยู่ในความสนใจคืองานที่เกี่ยวข้องกับระบบกริด

สถาบันวิทยบริการ
จุฬาลงกรณ์มหาวิทยาลัย