การรวมเครือข่ายเพียร์-ทู-เพียร์เข้ากับกริด

นายปกิต กาญจนะ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิศวกรรมศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมคอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2550

INTEGRATION PEER-TO-PEER NETWORK INTO THE GRID

Mr. Pakit Kanchana

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

หัวข้อวิทยานิพนธ์	การรวมเครือข่ายเพียร์-ทู-เพียร์เข้ากับกริด
โดย	นายปกิต กาญจนะ
สาขาวิชา	วิศวกรรมคอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.วีระ เหมืองสิน

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

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

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ศาสตราจารย์ ดร. ประภาส จงสถิตย์วัฒนา)

..... อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(อาจารย์ ดร. วีระ เหมืองสิน)

.....กรรมการภายนอกมหาวิทยาลัย
(ผู้ช่วยศาสตราจารย์ ดร. ภุชงค์ อุทโยภาศ)

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

ปกิต กาญจนะ : การรวมเครือข่ายเพียร์-ทู-เพียร์เข้ากับกริด. (INTEGRATION PEER-TO-PEER NETWORK INTO THE GRID) อาจารย์ที่ปรึกษา : อาจารย์ ดร.วีระเหมืองสิน, 94 หน้า.

การคำนวณแบบกริดมีการใช้งานมากขึ้นในปัจจุบันโดยเฉพาะในงานทางด้านวิทยาศาสตร์ ซึ่งต้องใช้ความสามารถในการคำนวณสูง ทรัพยากรในกริดมักจะประกอบด้วยเครื่องคอมพิวเตอร์สมรรถนะสูง หรือหน่วยเก็บข้อมูลขนาดใหญ่ ที่มักจะอยู่ในศูนย์คอมพิวเตอร์ ทำให้การขยายระบบจำเป็นต้องมีการดูแลเพิ่มเติม ในขณะที่ปัจจุบันคอมพิวเตอร์ส่วนบุคคลมีสมรรถนะสูงขึ้น และมีจำนวนมากในองค์กร ซึ่งอาจไม่ได้ใช้งานตลอดเวลา การนำทรัพยากรเหล่านี้มารวมกลุ่มกันเพื่อให้บริการแก่ระบบกริดจึงเป็นการเพิ่มเติมทรัพยากรให้แก่กริดได้โดยใช้ค่าใช้จ่ายและการดูแลระบบน้อย ซึ่งเทคโนโลยีที่สามารถจะรวบรวมทรัพยากรเหล่านั้นได้อย่างมีประสิทธิภาพก็คือเทคโนโลยีเพียร์-ทู-เพียร์

วิทยานิพนธ์ฉบับนี้เสนอการนำระบบเพียร์-ทู-เพียร์มาเชื่อมต่อกับกริดเพื่อเป็นทรัพยากรให้กับระบบกริด โดยใช้คอมพิวเตอร์ส่วนบุคคลที่มีอยู่ทั่วไปมาให้บริการร่วมกันเป็นกลุ่มเพียร์โดยใช้จักซ์ตา และออกแบบส่วนเชื่อมต่อกับกริดโดยใช้ความสามารถของมอดูลที่ขยายได้ของโกลบัส คือตัวจัดการงานของแกรม และดีเอสไอของกริดเอฟทีพี ทำให้ระบบมีลักษณะคล้ายกับคลัสเตอร์คอมพิวเตอร์ที่ให้บริการการจัดการงาน และการจัดการข้อมูล

ผู้วิจัยได้พัฒนาระบบและทดสอบประสิทธิภาพของระบบโดยรวม และได้เสนอแนะวิธีแก้ปัญหาจากการรวมทั้งสองระบบเข้าด้วยกัน ผลการทดสอบด้วยเครื่องคอมพิวเตอร์ส่วนบุคคลและเครือข่ายในภาควิชาวิศวกรรมคอมพิวเตอร์พบว่าสามารถทำงานได้ตามที่ออกแบบไว้ และให้ผลที่น่าพอใจ

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต ปกิต กาญจนะ

สาขาวิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่ออาจารย์ที่ปรึกษา วีระ เหมืองสิน

ปีการศึกษา 2550

497 04169 21 : MAJOR COMPUTER ENGINEERING

KEY WORD: GRID COMPUTING / PEER-TO-PEER / CLUSTER COMPUTING

PAKIT KANCHANA : INTREGRATION PEER-TO-PEER NETWORK INTO THE GRID. THESIS ADVISOR : VEERA MUANGSIN, Ph.D., 94 pp.

Grid Computing are gaining usability especially for resolving scientific problems which require high performance computation. Grid resources are usually high performance computers or large data storages locating in data centers so that installing more equipment needs more administration. However, the performance of personal computers has been increasing very fast. There are a large number of them in an institution which might not be used all the time. These resources would rather be organized in a group to provide some services in order to be additional Grid resources with less budget and administration. The technology that is suitable for this situation is peer-to-peer system.

This thesis proposes a model that integrates peer-to-peer system into the Grid system to be a Grid resource. Ubiquitous personal computers are organized in a group of services using JXTA and the interfaces between both systems are implemented using Globus extendable modules – GRAM Job Manager and GridFTP DSI. This makes the system much like a cluster computer which provides job management service and data management service.

The system is developed and the overall performance is evaluated. Some problems risen from integration both systems are resolved by the proposed solutions. The system test using personal computers and network devices in the Department of Computer Engineering shows that the system works properly.

Department Computer Engineering Student's signature ปกิท กาญจนา

Field of study Computer Engineering Advisor's signature วีระ เมืองสิน

Academic year 2550

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงด้วยดี ด้วยความช่วยเหลือของอาจารย์ ดร.วีระ เหมืองสิน ผู้เป็นอาจารย์ที่ปรึกษา ซึ่งได้ให้คำแนะนำตลอดทุกขั้นตอนของการทำวิจัย รวมทั้งรับฟังปัญหาสารพัดที่ข้าพเจ้าพบ และชี้แนะแนวทางและทัศนคติที่ดีซึ่งเป็นประโยชน์อย่างยิ่งต่อการทำงานของข้าพเจ้าเสมอมา

ข้าพเจ้าขอกราบขอบพระคุณอาจารย์ทุกท่านที่ประสิทธิประสาทวิชาตลอดช่วงชีวิตที่ผ่านมาของข้าพเจ้า ทำให้ข้าพเจ้ามีความรู้เป็นอาวุธติดตัวสะสมเรื่อยมาจนถึงทุกวันนี้

ข้าพเจ้าขอกราบขอบพระคุณนายเจริญ และนางยุพดี กาญจนะ บิดาและมารดา ผู้อยู่เบื้องหลังการใช้ชีวิตของข้าพเจ้า อบรมสั่งสอนให้ข้าพเจ้าใฝ่เรียนรู้ในสิ่งที่ดี เป็นแบบอย่างที่ดี เป็นผู้ให้กำลังใจที่ยิ่งใหญ่ รับฟังความคิดเห็น และอุปการะเลี้ยงดูข้าพเจ้าอย่างอบอุ่นเรื่อยมาจนถึงปัจจุบัน และนายพงศกร และนางสาวรสิกา กาญจนะ พี่ชายและน้องสาวที่คอยให้กำลังใจตลอดมา

สุดท้ายขอขอบคุณเพื่อนร่วมห้องปฏิบัติการ และเพื่อนนิสิตปริญญาโทและปริญญาเอกทุกคนที่ได้ฝ่าฟันทุกข์สุขมาด้วยกัน รับฟังเรื่องราวของกันและกัน และให้คำปรึกษาทั้งเรื่องงานวิจัยและการใช้ชีวิต ทำให้ข้าพเจ้ามีความสุขในการศึกษาที่ภาควิชาตลอด 2 ปีที่ผ่านมา

สารบัญ

หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญภาพ

ภาพประกอบ หน้า

ภาพประกอบ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมา

เทคโนโลยีกริด เป็นแนวคิดในการสร้างระบบการประมวลผลแบบกระจายขนาดใหญ่ข้ามองค์กรที่หลากหลาย ทรัพยากรนั้นอาจอยู่กระจัดกระจายกันตามสภาพทางภูมิศาสตร์ แต่ละองค์กรต่างก็มีนโยบายในการบริหารทรัพยากรของตนเองที่ต่างกัน ด้วยความร่วมมือระหว่างองค์กรที่มีการใช้ทรัพยากรร่วมกันจึงทำให้กริดมีพลังในการประมวลผล และพื้นที่เก็บข้อมูลมหาศาล ซึ่งสามารถใช้เป็นเครื่องมือในการทำงานสำคัญที่ต้องอาศัยการคำนวณสูง และบริหารข้อมูลจำนวนมากได้ ทรัพยากรบนกริดจึงมักเป็นอุปกรณ์ที่มีสมรรถนะสูง แต่โดยมากด้วยข้อจำกัดทางด้านงบประมาณ ทำให้อุปกรณ์ดังกล่าวมีจำนวนไม่มากนัก

การนำเอาทรัพยากรมาใช้งานร่วมกันเช่นนี้จะต้องมีระบบการบริหารที่ดี เพื่อให้ทุกองค์กรที่เข้าร่วมสามารถใช้งานได้อย่างสะดวก รวดเร็ว มีประสิทธิภาพ และปลอดภัย มีงานวิจัยจำนวนมากที่เสนอแนวคิดในการสร้างระบบกริดที่ดี และมาตรฐานของกริดได้ถูกพัฒนาขึ้น คือ OGSA (Open Grid Service Architecture) [1] และ OGSI (Open Grid Service Infrastructure) [2] ซึ่งปัจจุบัน OGSI ได้ถูกแทนที่ด้วย WSRF (Web Services Resource Framework) [3] และได้พัฒนาซอฟท์แวร์ตามมาตรฐานที่กำหนด ชื่อว่า โกลบัส ทูลคิท (Globus Toolkit) [4] ซึ่งในปัจจุบันได้กลายเป็นมาตรฐานมิดเดิลแวร์ของกริด และเป็นที่ยอมรับโดยทั่วไป

เทคโนโลยีเพียร์-ทู-เพียร์ (peer-to-peer) เป็นอีกแนวความคิดหนึ่งที่กำลังแพร่หลายมากในปัจจุบัน ทั้งนี้เนื่องจากระบบเพียร์-ทู-เพียร์มีลักษณะพิเศษที่เหมาะสมในการใช้งานสำหรับเครือข่ายที่ขยายตัวอย่างรวดเร็วในปัจจุบัน กล่าวคือ แต่ละเพียร์สามารถเป็นทั้งผู้ให้บริการและผู้รับบริการได้ในตัวเอง ลักษณะนี้เป็นการแก้ปัญหาในปัจจุบันของระบบรับ-ให้บริการ (client-server system) ที่ผู้ให้บริการมีเพียงผู้เดียวอยู่ที่ศูนย์กลางของระบบ ซึ่งอาจทำให้รองรับภาระหนักเกินไป นอกจากนี้ เพียร์สามารถเข้าร่วมเครือข่าย และออกจากเครือข่ายได้อย่างยืดหยุ่น และมีความสามารถในการจัดระเบียบกันเอง (self-organization) ลักษณะนี้ทำให้เครือข่ายเพียร์-ทู-เพียร์สามารถขยายตัวได้ดี ส่งผลให้ทรัพยากรรวมในระบบมีจำนวนมาก

ลักษณะของโปรแกรมประยุกต์ประเภทเพียร์-ทู-เพียร์ในปัจจุบันส่วนใหญ่คือระบบร่วมแฟ้ม (file sharing system) โดยที่แต่ละเพียร์มีแฟ้มข้อมูลของตัวเอง และสามารถแจกจ่ายแฟ้มไปให้เพียร์อื่นๆได้ ในขณะเดียวกันก็สามารถขอแฟ้มจากเพียร์อื่นได้เช่นกัน ระบบ

ร่วมแฟ้มที่เป็นที่นิยม ได้แก่ Napster [5], Gnutella [6], KaZaA [7] และ BitTorrent [8] เป็นต้น นอกจากระบบร่วมแฟ้มแล้ว เพียร์-ทู-เพียร์ยังมีศักยภาพในการพัฒนาโปรแกรมประยุกต์ประเภทอื่นๆอีก เช่น การคำนวณแบบกระจาย (distributed computing) ที่จะใช้พลังการคำนวณของเพียร์ต่างๆในเครือข่ายในการช่วยแก้ปัญหาขนาดใหญ่ที่ต้องใช้เวลา และทรัพยากรในการประมวลผลสูง โปรแกรมประยุกต์ประเภทนี้มักจะใช้เครื่องคอมพิวเตอร์ส่วนบุคคลที่เชื่อมต่อกันผ่านทางอินเทอร์เน็ต ซึ่งอาจเรียกว่า การคำนวณบนอินเทอร์เน็ต (internet computing) หรือ เดสก์ท็อปกริด (desktop grid) เช่น SETI@home [9], BOINC [10], Entropia[11], Javelin [12] CCOF [13] เป็นต้น การคำนวณแบบกระจายโดยใช้เครือข่ายเพียร์-ทู-เพียร์ก็เป็นส่วนหนึ่งในโปรแกรมประยุกต์ประเภทนี้เช่นกัน

กริดและเพียร์-ทู-เพียร์เป็นเทคโนโลยีที่มีวัตถุประสงค์ใกล้เคียงกัน นั่นคือการรวมเอาทรัพยากรที่มีอยู่อย่างกระจัดกระจายในเครือข่าย เข้ามารวมกัน และมีการจัดการทรัพยากรอย่างมีระบบ เพื่อให้สามารถนำทรัพยากรเหล่านั้นมาใช้ได้อย่างมีประสิทธิภาพ อย่างไรก็ตาม สถาปัตยกรรมและสภาพแวดล้อมในการใช้งานของระบบกริด และเพียร์-ทู-เพียร์นั้นต่างกันในหลายจุด กล่าวคือ

- ระบบกริด ริเริ่มจากการที่พยายามจะรวมทรัพยากรที่มีประสิทธิภาพสูงในองค์กรเข้ามารวมกันเพื่อเป็นทรัพยากรที่รองรับการคำนวณปัญหาขนาดใหญ่ ดังนั้นทรัพยากรบนกริดจึงประกอบคอมพิวเตอร์สมรรถนะสูง หน่วยเก็บข้อมูลขนาดใหญ่ เครือข่ายความเร็วสูง รวมถึงเครื่องมือเฉพาะทางต่างๆ

- ผู้ที่เข้าร่วมในกริดเป็นองค์กรหรือหน่วยงานที่เป็นเจ้าของทรัพยากร หรือผู้ใช้บริการ

- เครื่องคอมพิวเตอร์ในกริดมักจะมีหน้าที่ให้บริการตลอดเวลา (dedicate)

- การจัดสรรทรัพยากรบนกริดมักจะมีลักษณะแบบรับ-ให้บริการ (client-server) โดยการบริการอาจเป็นแบบรวมศูนย์ (centralized) แบบลำดับชั้น (hierarchy) หรือแบบกระจาย (decentralized) และมีกฎเกณฑ์ที่ชัดเจนในการจัดสรรทรัพยากรและสิทธิ์ในการเข้าใช้งาน

- โครงสร้างของทรัพยากรบนกริดไม่ได้มีการเปลี่ยนแปลงหรือเคลื่อนไหวบ่อยครั้ง ทำให้การเข้าถึงทรัพยากรมักรับรองคุณภาพในการบริการได้

- ปัจจุบันมีระบบกริดมาตรฐานที่ยอมรับโดยทั่วไป และมีมิดเดิลแวร์ที่ถูกพัฒนาขึ้นตามมาตรฐานจำนวนมาก ซึ่งสามารถประยุกต์ใช้ได้กับงานทั่วไป

- การขยายตัวของระบบมีข้อจำกัด เนื่องจากระบบส่วนใหญ่มักเป็นอุปกรณ์ระดับเซิร์ฟเวอร์, ใช้หมายเลขไอพีคงที่, มีขนาดใหญ่ เป็นต้น ทำให้มีข้อจำกัดในการขยายระบบที่ต้องดำเนินการโดยผู้ดูแลระบบที่เชี่ยวชาญ

ในขณะที่อีกมุมหนึ่งคือระบบเพียร์-ทู-เพียร์นั้นแตกต่างออกไป คือ

- ระบบเพียร์-ทู-เพียร์มีจุดเริ่มต้นมาจากการที่ผู้ใช้คอมพิวเตอร์ต้องการจะแบ่งปันทรัพยากรของตนให้กับผู้อื่น รวมถึงขอใช้ทรัพยากรของผู้อื่น ในระบบร่วมแฟ้ม ทรัพยากรที่กล่าวถึงก็คือแฟ้มข้อมูลนั่นเอง การใช้งานประเภทอื่นก็มีขึ้นในระบบเพียร์-ทู-เพียร์เช่นกัน เช่น การถ่ายโอนข้อมูลแบบทันที (real time data transfer) เช่น Skype [14] , หรือประเภท แย่งรอบทำงาน (cycle stealing)

- ผู้ที่เข้าร่วมในเครือข่ายเพียร์-ทู-เพียร์มักจะเป็นผู้ที่ใช้คอมพิวเตอร์ส่วนบุคคลที่เชื่อมต่อกับอินเทอร์เน็ต

- เพียร์ในเครือข่ายเพียร์-ทู-เพียร์ อาจจะเข้าและออกจากเครือข่ายเมื่อใดก็ได้ และไม่สามารถคาดการณ์ได้ล่วงหน้า

- จัดการทั้งหมดไม่มีศูนย์กลาง แต่จะมีการจัดการร่วมกันด้วยการส่งข้อความถึงกันเพื่อบอกสถานะของตนเอง และสถานะของสภาพแวดล้อมที่ตนรับรู้ กลไกการทำงานจึงมีลักษณะเป็นแบบกระจาย เพื่อลดการทำงานแบบรับ-ให้บริการ และยังมีความสามารถในการจัดระเบียบกันเอง

- ระบบเพียร์-ทู-เพียร์ขยายตัวได้ง่าย เนื่องจากอุปกรณ์ส่วนใหญ่เป็นเครื่องคอมพิวเตอร์ส่วนบุคคลที่หาได้ทั่วไป สามารถติดต่อกันได้ผ่านเครือข่ายอินเทอร์เน็ต

- ระบบเพียร์-ทู-เพียร์ไม่ได้มีมาตรฐานในการให้บริการที่ชัดเจน หรือเป็นไปในแนวทางเดียวกันเช่นในระบบกริด และมักเน้นไปที่การประยุกต์ใช้เพื่อจุดประสงค์อย่างใดอย่างหนึ่งเท่านั้น

จากลักษณะการทำงานของระบบกริด และระบบเพียร์-ทู-เพียร์ จะสังเกตว่าต่างก็มีข้อดีและข้อเสียต่างกัน และน่าจะนำประโยชน์ของทั้งสองระบบมารวมกันได้ เพื่อให้ได้เป็นระบบที่เพิ่มเติมความสามารถให้แก่กัน การนำระบบเพียร์-ทู-เพียร์เข้ามารวมกับระบบกริดนั้นสามารถเป็นไปได้หลายแนวทาง ไม่ว่าจะเป็นการนำเทคโนโลยีของเพียร์-ทู-เพียร์มาใช้จัดการทรัพยากรบนกริดเพื่อรองรับการขยายตัว หรือการนำเทคโนโลยีแบบกริดมาเพิ่มคุณภาพการให้บริการหรือความปลอดภัยในระบบเพียร์-ทู-เพียร์

ผู้วิจัยเล็งเห็นว่าประสิทธิภาพของเครื่องคอมพิวเตอร์ส่วนบุคคลในปัจจุบันเติบโตขึ้นอย่างรวดเร็ว และมีแนวโน้มที่จะเพิ่มประสิทธิภาพขึ้นเรื่อยๆ จึงเกิดแนวคิดที่จะสร้างกลุ่มของหน่วยประมวลผลที่ประกอบไปด้วยเครื่องคอมพิวเตอร์ส่วนบุคคล โดยใช้เทคโนโลยีของระบบเพียร์-ทู-เพียร์ในการเชื่อมต่อ และการจัดสรรทรัพยากร และนำไปเชื่อมต่อกับระบบกริด เพื่อใช้เป็นทรัพยากรของระบบกริด และให้บริการแก่ระบบกริด ซึ่งเมื่อมองโดยภาพรวม ระบบที่พัฒนาขึ้นมีจึงลักษณะคล้ายคลัสเตอร์คอมพิวเตอร์นั่นเอง และด้วยความสามารถในการขยายตัวของเครือข่ายเพียร์-ทู-เพียร์ จะทำให้รวบรวมทรัพยากรให้แก่กริดได้จำนวนมากอย่างรวดเร็ว

1.2 วัตถุประสงค์ของการวิจัย

งานวิจัยนี้มีวัตถุประสงค์เพื่อนำเทคโนโลยีกริดและเพียร์-ทู-เพียร์เข้ามาเชื่อมต่อกัน โดยการรวบรวมทรัพยากรคอมพิวเตอร์ส่วนบุคคลในรูปแบบเพียร์-ทู-เพียร์ และให้บริการแก่ระบบกริด เพื่อให้กริดมีทรัพยากรที่ขยายตัวได้ง่ายและรวดเร็ว

1.3 ขอบเขตของการวิจัย

1. นำระบบจ่ายงาน (job submission) และระบบแฟ้มข้อมูล (file system) ในเครือข่ายเพียร์-ทู-เพียร์มารวมกัน เพื่อสร้างระบบเพียร์-ทู-เพียร์คลัสเตอร์
2. พัฒนาส่วนเชื่อมต่อกริดและเพียร์-ทู-เพียร์อยู่บนมาตรฐานของ OGSA/OGSI โดยใช้ โกลบัส ทูลคิท รุ่น 2 เป็นหลัก และให้เรียกใช้งานได้จากสภาพแวดล้อมของกริดผ่านทางคำสั่ง globusrun และ globus-url-copy สำหรับการสั่งงาน และการถ่ายโอนแฟ้ม ตามลำดับ
3. เครื่องที่เข้าร่วมเครือข่ายเพียร์-ทู-เพียร์ใช้ระบบปฏิบัติการวินโดวส์และลินุกซ์
4. ทดสอบการทำงานของกริดและเพียร์-ทู-เพียร์โดยใช้เครื่องเซิร์ฟเวอร์ และเครื่องคอมพิวเตอร์ส่วนบุคคล ในภาควิชาวิศวกรรมคอมพิวเตอร์

1.4 ประโยชน์ที่คาดว่าจะได้รับ

งานวิจัยนี้เป็นการพัฒนาส่วนเชื่อมต่อระหว่างระบบกริดกับเพียร์-ทู-เพียร์เป็นหลัก เพื่อให้สามารถนำเอาทรัพยากรในเครือข่ายเพียร์-ทู-เพียร์ที่หาได้ง่ายในปัจจุบันมาใช้ให้เกิดประโยชน์ ผลที่จะได้รับคือ จะได้ทรัพยากรเพิ่มเติมให้แก่กริดจำนวนมาก ที่มีความยืดหยุ่นสูง และมีประสิทธิภาพ ซึ่งเป็นแนวทางที่จะพัฒนาให้สามารถนำไปใช้เป็นเครื่องมือสำหรับงานวิจัยอื่นๆ ได้ในอนาคต

1.5 วิธีดำเนินการวิจัย

1. ศึกษาทฤษฎีพื้นฐานเกี่ยวกับการคำนวณแบบกระจาย และงานวิจัยต่างๆ เกี่ยวกับการคำนวณแบบกระจาย
2. ศึกษางานวิจัย [15] และ [16] อย่างละเอียด
3. ออกแบบและพัฒนาวิธีการรวมระบบจ่ายงาน ระบบแฟ้มข้อมูล และส่วนเชื่อมต่อระหว่างกริดกับเพียร์-ทู-เพียร์
4. ทดสอบระบบ
5. วิเคราะห์ผลการทำงาน และสรุปผล
6. เขียนวิทยานิพนธ์

บทที่ 2

เอกสารและงานวิจัยที่เกี่ยวข้อง

2.1 แนวคิดและทฤษฎี

2.1.1 การคำนวณแบบกริด (Grid computing)

เทคโนโลยีกริด คือ การนำเอาทรัพยากรคอมพิวเตอร์ที่มีในแต่ละองค์กรมารวมกัน โดยมีจุดประสงค์ทั่วไปคือใช้สำหรับทำการวิจัยหรือเพื่อใช้งานระดับองค์กร ทรัพยากรในองค์กรที่กล่าวถึง อาจเป็นเครื่องซูเปอร์คอมพิวเตอร์ คลัสเตอร์คอมพิวเตอร์ หรือ เครื่องคอมพิวเตอร์ส่วนบุคคลก็ได้ ซึ่งการที่ทรัพยากรจำนวนมากถูกรวมกันนี้จะทำให้ได้เป็นเครื่องซูเปอร์คอมพิวเตอร์จำลองที่สามารถเข้าใช้ได้จากองค์กรต่างๆ ผู้ที่มีส่วนร่วมในงานสามารถแจกจ่ายพลังในการประมวลผลของเครื่องคอมพิวเตอร์ในองค์กรของตนและสามารถใช้พลังการประมวลผลของเครื่องในองค์กรอื่นได้ ลักษณะการทำงานของกริดคือผู้ใช้สามารถสั่งงานระยะไกลไปยังเครื่องคำนวณได้

รูปที่ 2-1 การคำนวณแบบกริด

การคำนวณแบบกริด มักใช้สำหรับงานที่มีการคำนวณสูงมาก ซึ่งไม่อาจทำได้สำเร็จในเวลาที่ต้องการด้วยเครื่องซูเปอร์คอมพิวเตอร์เพียงเครื่องเดียว ดังนั้นกริดจึงมีแนวคิดที่จะสร้างการคำนวณแบบกระจาย โดยที่แบ่งงานใหญ่เป็นงานย่อย แล้วกระจายไปประมวลผลที่เครื่องต่างๆ ซึ่งเป็นหลักการโดยทั่วไปของการคำนวณแบบกระจาย ปัญหาที่ตามมาก็คือ จะต้องมีการพัฒนาระบบการพิสูจน์ตัวตน (authentication) และสิทธิ์ (authorization) เพื่อขออนุญาตเครื่องคำนวณต่างๆ ก่อนที่จะส่งงานไปประมวลผล

กริดอาจประกอบไปด้วยเครื่องคอมพิวเตอร์ที่มีลักษณะต่างกัน (heterogeneous) เช่น แพลตฟอร์ม , สถาปัตยกรรมของฮาร์ดแวร์และซอฟท์แวร์ และภาษาคอมพิวเตอร์ เครื่องคอมพิวเตอร์เหล่านี้กระจายอยู่ในที่ต่างๆ ดังนั้นกริดจึงต้องมีมาตรฐานเปิด ที่จะทำให้มีการสื่อสารกันได้ หลักการนี้เชื่อมโยงไปสู่เรื่องทรัพยากรเสมือน (virtual resources) นั่นคือ ผู้ใช้จากภายนอกไม่จำเป็นต้องรู้ว่าทรัพยากรมีลักษณะเป็นแบบใด หรืออยู่ที่ใด เพียงแต่รู้ว่าทรัพยากรเหล่านั้นให้บริการอะไรบ้าง ในทำนองเดียวกัน การสื่อสารระหว่างองค์กรก็ไม่จำเป็นจะต้องรู้โครงสร้างขององค์กรที่ติดต่ออยู่ เพียงแต่มีมาตรฐานที่ใช้ร่วมกันในการสื่อสารก็เพียงพอแล้ว ลักษณะเช่นนี้จำเป็นต้องมีเครื่องมือที่จะทำให้ทรัพยากรต่างสื่อสารกันด้วยมาตรฐานเดียวกัน หรือเรียกว่า มิดเดิลแวร์ (middleware)

ในปัจจุบัน กริดถูกพัฒนาขึ้นให้มีมาตรฐานที่ยอมรับโดยทั่วไป ซึ่งผู้ที่ร่วมพัฒนามาตรฐานของกริดประกอบด้วยสถาบันสองแห่ง นั่นคือ Globus Alliance [17] และ Global Grid Forum (GGF) [18] ซึ่งสร้างกลุ่มของมาตรฐานเปิดสำหรับเทคโนโลยีกริดและโปรแกรมประยุกต์ โดย GGF ประกอบด้วยสถาบันการศึกษา หน่วยงานวิจัย และบริษัททั้งขนาดใหญ่และเล็ก แรงผลักดันหลักของ GGF ประกอบด้วย Open Grid Services Architecture (OGSA) และ Open Grid Services Infrastructure (OGSI) โดย Globus Alliance และ GGF ได้ร่วมกันพัฒนาโกลบัสทูลคิทที่เป็นมิดเดิลแวร์ที่สร้างขึ้นโดยอ้างอิงตามมาตรฐานของ OGSA/OGSI

2.1.2 **เพียร์-ทู-เพียร์** (Peer-to-peer: P2P)

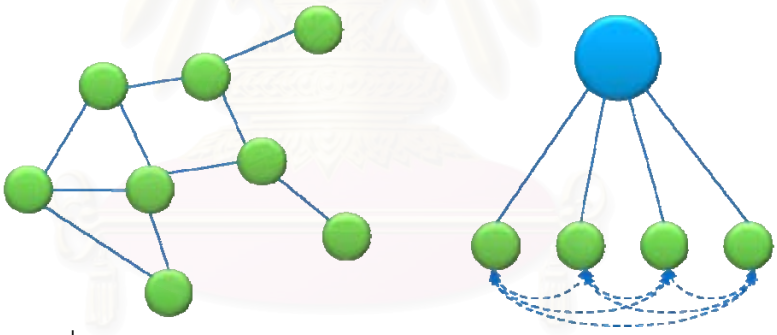
เครือข่ายเพียร์-ทู-เพียร์ ประกอบด้วยเครื่องคอมพิวเตอร์หลายๆเครื่องกระจายอยู่ในเครือข่าย โดยที่ไม่มีศูนย์กลาง หรือกลุ่มของศูนย์กลางที่เป็นเซิร์ฟเวอร์ของระบบ ทุกเครื่องจะมีสถานะเท่าเทียมกัน คือเป็นทั้งผู้ให้และผู้รับบริการได้ในเวลาเดียวกัน เครือข่ายเพียร์-ทู-เพียร์โดยปกติจะใช้สำหรับการติดต่อเฉพาะกิจ (ad hoc connection) เครือข่ายแบบนี้มีประโยชน์ในหลายแง่ ซึ่งส่วนใหญ่จะใช้สำหรับการแจกจ่ายแฟ้มข้อมูล เช่น แฟ้มข้อมูลเสียง วิดีโอ และข้อมูลดิจิตอลต่างๆ ซึ่งเหล่านี้เป็นรูปแบบทั่วไป นอกจากนี้ ข้อมูลแบบทันที (real time) เช่นระบบโทรศัพท์ก็สามารถใช้งานผ่านเทคโนโลยีเพียร์-ทู-เพียร์ได้เช่นกัน

ระบบเพียร์-ทู-เพียร์มักจะนำมาเปรียบเทียบกับระบบรับ-ให้บริการ (client-server system) สำหรับระบบเพียร์-ทู-เพียร์นั้น แต่ละเพียร์ที่มีสถานะเท่าเทียมกัน จะเป็นได้ทั้งเครื่องบริการ (server) และเครื่องลูกข่าย (client) ซึ่งโครงสร้างแบบนี้ต่างกับระบบรับ-ให้บริการที่การติดต่อมักจะอยู่ที่เครื่องบริการที่ศูนย์กลาง

ระบบเพียร์-ทู-เพียร์อาจแบ่งตามลักษณะทอพอโลยีได้เป็น

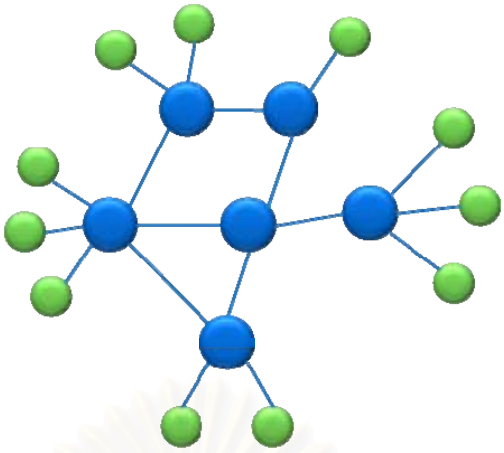
1. ระบบเพียร์-ทู-เพียร์แท้ (Pure P2P) คือ ระบบที่ทุกเครื่องมีความเท่าเทียมกันอย่างแท้จริง ไม่มีเครื่องศูนย์กลางที่จะจัดการทรัพยากรในเครือข่าย และไม่มีศูนย์กลางในการจัดเส้นทางการสื่อสารใดๆทั้งสิ้น ดังนั้นการเริ่มติดต่อระหว่างเพียร์อาจจะต้องใช้วิธีการกระจายข้อความไปยังเพียร์ต่างๆ เริ่มจากเพียร์ที่ตนเองรู้จัก และกระจายไปเรื่อยๆจนเจอข้อมูลที่ต้องการในเครือข่าย วิธีแบบนี้มีจุดประสงค์เพื่อจะหาข้อมูลให้ครอบคลุมทั้งเครือข่าย เพื่อให้ได้ผลลัพธ์อย่างแน่นอน แต่อาจต้องใช้เวลามาก และใช้แบนด์วิดท์จำนวนมากในการค้นหาเพียร์

2. ระบบเพียร์-ทู-เพียร์ลูกผสม (Hybrid P2P) คือ การผสมผสานกันระหว่างระบบรับให้บริการกับระบบเพียร์-ทู-เพียร์ กล่าวคือจะมีเพียร์หรือกลุ่มของเพียร์ที่ทำหน้าที่เป็นศูนย์กลางในการจัดเก็บดรรชนี (index) ของข้อมูลต่างๆในเครือข่ายเพียร์-ทู-เพียร์ อาจจะรวมถึงการเป็นศูนย์กลางของการค้นหาเส้นทางไปยังเพียร์ต่างๆอีกด้วย หลังจากนั้นเพียร์จะติดต่อสื่อสารกันโดยตรงโดยไม่ผ่านเพียร์ศูนย์กลางอีก ในการเริ่มต้นติดต่อระหว่างเพียร์ จะมีการค้นหาไปที่เครื่องศูนย์กลางเพื่อให้ได้ผลลัพธ์เป็นที่อยู่จริงของข้อมูล และอาจรวมถึงเส้นทางที่ไปสู่ข้อมูล หลังจากนั้นก็จะใช้ที่อยู่ที่ได้นี้เป็นการอ้างอิงไปถึงที่อยู่ของข้อมูลจริงในเครือข่าย



รูปที่ 2-2 ระบบเพียร์-ทู-เพียร์แท้ และระบบเพียร์-ทู-เพียร์ลูกผสม

3. ระบบเพียร์-ทู-เพียร์อีกแบบหนึ่งเป็นการผสมผสานข้อดีของเพียร์-ทู-เพียร์แท้และเพียร์-ทู-เพียร์ลูกผสม นั่นคือ ซูเปอร์เพียร์ (super peer) โดยในเครือข่ายจะมีซูเปอร์เพียร์ที่ทำหน้าที่คล้ายกับเพียร์ศูนย์กลางของเพียร์-ทู-เพียร์ลูกผสมในการให้ข้อมูลต่างๆแก่เพียร์ขอบ (edge peer) และซูเปอร์เพียร์แต่ละตัวก็จะติดต่อระหว่างกันในลักษณะเดียวกับแบบเพียร์-ทู-เพียร์แท้ ดังนั้นหน้าที่ในการค้นหาข้อมูลจะกระจายอยู่ในกลุ่มของซูเปอร์เพียร์ ระบบนี้ช่วยให้การขยายตัวของเครือข่ายดีขึ้น เนื่องจากหากมีเพียร์ใหม่เข้ามาในระบบมากขึ้น จะมีการคัดเลือกบางเพียร์ขึ้นมาเป็นซูเปอร์เพียร์เพิ่มเติม นอกจากนี้ระบบยังมีความคงทนสูง เนื่องจากหากซูเปอร์เพียร์ตัวใดเสียไป ก็จะมีเฉพาะเพียร์ในกลุ่มของตัวเองเท่านั้นที่จะเสีย แต่จะไม่รบกวนการทำงานทั้งระบบ



รูปที่ 2-3 ระบบซูเปอร์เพียร์

เป้าหมายสำคัญของเพียร์-ทู-เพียร์คือเครื่องลูกข่ายจะให้ทรัพยากร ประกอบด้วย แบนด์วิดท์ (bandwidth), พื้นที่หน่วยเก็บข้อมูล และพลังการคำนวณ ดังนั้นเมื่อมีจำนวนเพียร์เพิ่มขึ้นและความต้องการของระบบเพิ่มขึ้น ความสามารถโดยรวมของระบบก็เพิ่มขึ้นตามไปด้วย ซึ่งกรณีเช่นนี้จะไม่เกิดขึ้นในระบบรับ-ให้บริการ ที่เมื่อมีเครื่องลูกข่ายมากขึ้น ภาระในการให้บริการก็จะมากขึ้น และทำให้สภาพการทำงานของระบบช้าลง

ระบบเพียร์-ทู-เพียร์อาจแบ่งแยกตามลักษณะของโครงสร้าง ได้เป็น

1. เครือข่ายเพียร์-ทู-เพียร์แบบไม่มีโครงสร้าง(Unstructured peer-to-peer network) เพียร์ใหม่ที่เข้ามาในระบบจะคัดลอกเส้นทางจากเพียร์อื่น และจัดการเส้นทางการติดต่อด้วยตัวเองตลอดเวลา หากเพียร์ต้องการค้นหาข้อมูลใดๆ จะเริ่มต้นโดยการกระจายข้อความค้นหาไปในเครือข่าย เพื่อค้นหาให้ครอบคลุมที่สุดเท่าที่จะทำได้ ตัวอย่างของแบบไม่มีโครงสร้างได้แก่ Napster[5], Gnutella[6] และ KaZaA[7] เป็นต้น

2. เครือข่ายเพียร์-ทู-เพียร์แบบมีโครงสร้าง (Structured peer-to-peer network) มีการจัดการโครงสร้างโดยใช้ตารางแฮชแบบกระจาย (Distributed Hash Table : DHT) โดยที่แต่ละเพียร์จะมีหน้าที่รับผิดชอบข้อมูลบางส่วนของเครือข่าย ข้อมูลจะถูกเข้าฟังก์ชันแฮช และกระจายไปยังเพียร์ที่สมควรรับผิดชอบข้อมูลนั้นมากที่สุด โดยพิจารณาจากค่าแฮชของข้อมูลกับค่าแฮชของแต่ละเพียร์ ระบบนี้ทำให้ข้อมูลถูกกระจายอย่างสม่ำเสมอในเครือข่าย และเมื่อเพียร์ต้องการค้นหาข้อมูลที่ต้องการ ก็จะมีวิธีการหาเพียร์ที่รับผิดชอบข้อมูลนั้นได้โดยตรง การจัดการแบบนี้เป็นการแก้ปัญหาการกระจายข้อความค้นหาโดยไม่จำเป็นไปยังเพียร์ต่างๆดังเช่นในเครือข่ายเพียร์-ทู-เพียร์แบบไม่มีโครงสร้าง ตัวอย่างของโครงสร้าง ได้แก่ CAN[19], Chord[20], Pastry[21] และ Tapestry[22] เป็นต้น

นอกจากระบบเพียร์-ทู-เพียร์จะถูกนำมาใช้ในระบบร่วมแฟ้มที่เราเห็นได้ชัดแล้ว ยังมีโปรแกรมประยุกต์หลายประเภทที่ใช้ความสามารถของระบบเพียร์-ทู-เพียร์ เช่น การส่งสารทันที (instant messaging) ที่เป็นการส่งข้อความสื่อสารไปยังเพียร์ต่างๆได้, การคำนวณแบบกระจาย (distributed computing) ที่ใช้พลังการประมวลผลจากเพียร์ต่างๆในเครือข่าย เป็นต้น

2.1.3 **คลัสเตอร์คอมพิวเตอร์** (Cluster computer)

คลัสเตอร์คอมพิวเตอร์ ประกอบด้วยเครื่องคอมพิวเตอร์หลายเครื่อง ที่มักจะวางอยู่ใกล้กัน และเชื่อมต่อกันด้วยแลน (LAN) ที่มีความเร็วสูง เพื่อให้มีลักษณะเหมือนเครื่องคอมพิวเตอร์สมรรถนะสูงเครื่องเดียวที่มีราคาประหยัด คลัสเตอร์คอมพิวเตอร์อาจเป็นการเชื่อมต่อของเครื่องคอมพิวเตอร์ที่เหมือนกัน (homogeneous) หรือแตกต่างกัน (heterogeneous) ก็ได้ โดยส่วนใหญ่มักจะเป็นเครื่องที่เหมือนกัน เพื่อให้ลดปัญหาความแตกต่างของเทคโนโลยี คลัสเตอร์คอมพิวเตอร์มักใช้สำหรับเพิ่มประสิทธิภาพการทำงาน (performance) หรือเพิ่มสภาพพร้อมใช้งาน (availability) เมื่อเทียบกับคอมพิวเตอร์เครื่องเดียว

คลัสเตอร์คอมพิวเตอร์อาจจำแนกได้เป็น

1. คลัสเตอร์สภาพพร้อมใช้งานสูง (High-availability clusters) เป็นคลัสเตอร์ที่สร้างขึ้นเพื่อใช้สำหรับให้บริการบางอย่างที่ต้องการสภาพพร้อมใช้งานตลอดเวลา จึงต้องมีโหนดที่ทำหน้าที่ซ้ำซ้อน (redundant) ทั้งนี้เพื่อให้ป้องกันการเกิดปัญหาในกรณีที่โหนดใดโหนดหนึ่งทำงานผิดพลาด

2. คลัสเตอร์กระจายภาระ (Load-balancing clusters) เป็นคลัสเตอร์ที่สร้างขึ้นเพื่อเพิ่มประสิทธิภาพในการทำงาน โดยการกระจายภาระของบริการใดๆไปที่โหนดต่างๆของคลัสเตอร์อย่างสม่ำเสมอ เพื่อให้รองรับกับภาระจำนวนมากๆได้ คลัสเตอร์แบบนี้มักจะมีควบคู่กับคลัสเตอร์สภาพพร้อมใช้งานสูงอาจเรียกคลัสเตอร์ประเภทนี้ได้ว่าเซิร์ฟเวอร์ฟาร์ม (server farm)

3. คลัสเตอร์สมรรถนะสูง (High-performance clusters) เป็นคลัสเตอร์ที่สร้างขึ้นเพื่อเพิ่มสมรรถนะ โดยการแบ่งงานที่ต้องใช้การคำนวณสูงออกเป็นงานย่อย และกระจายไปตามโหนดต่างๆในคลัสเตอร์ โดยทั่วไปใช้สำหรับงานที่เกี่ยวกับการคำนวณทางวิทยาศาสตร์ คลัสเตอร์ประเภทนี้มักจะใช้รันโปรแกรมที่ออกแบบให้ทำงานแบบเชิงขนานได้ (parallel job) โปรแกรมหลายโปรแกรมใช้คลังคำสั่งเอ็มพีไอ (MPI: Message Passing Interface) ที่รองรับการคำนวณเชิงขนาน เนื่องจากคลัสเตอร์ประเภทนี้เกี่ยวข้องกับงานวิจัยนี้โดยตรง ดังนั้นในส่วนต่อๆไปที่กล่าวถึงคลัสเตอร์ จะหมายถึงคลัสเตอร์สมรรถนะสูงเท่านั้น

คลัสเตอร์เป็นเทคโนโลยีที่เกี่ยวข้อง และใกล้ชิดกับกริด กล่าวคือ ทั้งสองเทคโนโลยีต่างก็เป็นการรวมทรัพยากรเพื่อให้บริการสำหรับการคำนวณเป็นหลักเช่นเดียวกัน สิ่งที่ต่างกันระหว่างกริดกับคลัสเตอร์คือ กริดจะเป็นการรวมเครื่องข้ามองค์กร (รวมถึงเครื่องคลัสเตอร์คอมพิวเตอร์ด้วย) โดยที่เครื่องเหล่านั้นไม่ได้เชื่อถือกันได้เต็มที่ ซึ่งจำเป็นต้องมีระบบการพิสูจน์ตัวตน และสิทธิ์การใช้งานก่อนทุกครั้ง ดังนั้นกริดจึงเสมือนเป็นการรวบรวมเครื่องมือสำหรับการคำนวณมากกว่าจะเป็นหน่วยคำนวณเหมือนคลัสเตอร์ ส่วนคลัสเตอร์มักจะประกอบด้วยเครื่องที่อยู่ในแลน และเชื่อถือกันเต็มที่ การจัดการทางด้านความปลอดภัยของระบบจึงอ่อนกว่าระบบกริด ซึ่งทำให้คลัสเตอร์สามารถจัดการกับทรัพยากรในคลัสเตอร์ได้ง่ายกว่า นอกจากนี้ คลัสเตอร์เป็นเทคโนโลยีที่เกิดมาก่อนกริด ดังนั้นจึงมีเครื่องมือสำหรับบริหารจัดการทรัพยากรภายในคลัสเตอร์มากมาย ซึ่งค่อนข้างมีประสิทธิภาพ

2.1.4 Internet Computing/Desktop Grid

การคำนวณบนอินเทอร์เน็ต (internet computing) เป็นการนำคอมพิวเตอร์ที่อยู่ในเครือข่ายอินเทอร์เน็ตมาใช้ในการประมวลผลร่วมกัน จุดประสงค์ของระบบนี้คือใช้สำหรับเพิ่มปริมาณงาน (throughput) และจากการที่ส่วนมากเครื่องคอมพิวเตอร์เหล่านี้มักเป็นเครื่องคอมพิวเตอร์ส่วนบุคคล เราจึงอาจเรียกระบบนี้ได้ว่า “เดสก์ท็อปกริด” (desktop grid)

ในปัจจุบันมีงานวิจัยจำนวนมากที่มีลักษณะเป็นเดสก์ท็อปกริด และแต่ละงานก็จะมีวิธีการในการจัดการกับทรัพยากรจำนวนมากนั้นแตกต่างกันออกไป เราอาจแบ่งประเภทของระบบเดสก์ท็อปกริดได้ตามลักษณะต่างๆดังนี้ [23]

1. *การจัดองค์กร (Organization)* อาจแบ่งได้เป็นแบบรวมศูนย์ (centralized) และแบบกระจาย (distributed) ในแบบรวมศูนย์นั้นจะมีเครื่องแม่ข่าย (server) เป็นศูนย์กลาง โดยมีหน้าที่เพื่อจัดการลำดับงานและจัดสรรทรัพยากรที่เหมาะสมกับงาน การสั่งงานจะสั่งไปที่เครื่องแม่ข่าย ตัวอย่างของระบบนี้ เช่น BOINC[10], Entropia[11] ระบบแบบกระจายไม่มีเครื่องศูนย์กลาง แต่เครื่องผู้ให้ทรัพยากรทั้งจะแลกเปลี่ยนข้อมูลซึ่งกันและกันเพื่อจัดโครงสร้างกันเอง การสั่งงานจะสั่งไปที่เครื่องผู้ให้ทรัพยากรใดก็ได้ ตัวอย่างของระบบนี้ เช่น CCOF [24]

2. *แพลตฟอร์ม (Platform)* อาจแบ่งได้เป็นแบบเว็บ หรือจาวาแอปเพล็ต (Java Applet) และแบบมิดเดิลแวร์ ในแบบเว็บผู้สั่งงานเขียนโปรแกรมแบบขนานด้วยจาวาและประกาศบนเว็บ อาสาสมัครเพียงแค่เปิดเว็บด้วยโปรแกรมบราวเซอร์ แอปเพล็ตจะถูกดาวน์โหลดและรันบนเครื่องอาสาสมัคร ตัวอย่างเช่น Javelin [12] แบบมิดเดิลแวร์ อาสาสมัครจะต้องติดตั้งโปรแกรมมิดเดิลแวร์บนเครื่อง ตัวอย่างเช่น BOINC[10], Xtremweb[25], Entropia[11]

3. *การขยายขนาด (Scale)* อาจแบ่งได้เป็นระดับอินเทอร์เน็ต และแลน ขึ้นอยู่กับขนาดและสภาพแวดล้อมที่ติดตั้ง โปรแกรมที่รันบนอินเทอร์เน็ตจะต้องคำนึงถึงแบนด์วิดท์ที่น้อย, ไฟร์วอลล์ (firewall) และการเชื่อมต่อที่ไม่เสถียร เช่น BOINC ส่วนในแบบแลนเป็นการเชื่อมต่อด้วยเครื่องที่อยู่ใกล้กัน เช่นในห้องแล็บ หรือในองค์กร เป็นต้น ซึ่งจะมีการเชื่อมต่อที่เสถียรกว่าแบบอินเทอร์เน็ตมาก เช่น Condor [26]

4. *ผู้ให้ทรัพยากร (Resource Provider)* อาจแบ่งได้เป็นอาสาสมัคร (volunteer) และแบบองค์กร (enterprise) โดยที่แบบอาสาสมัครมักเชื่อมต่อแบบอินเทอร์เน็ต และแบบองค์กรมักจะเชื่อมต่อแบบแลน

ปัญหาที่ท้าทายของเดสก์ท็อปกริดได้แก่ ความไม่แน่นอน, สภาพแวดล้อมที่เคลื่อนไหวตลอดเวลา, ความน่าเชื่อถือ, ความผิดพลาด, ความแตกต่างของเทคโนโลยี, การขยายขนาด, และผู้เข้าร่วม เมื่อเปรียบเทียบระหว่างระบบเดสก์ท็อปกริดใดๆ ปัจจัยเหล่านี้มักจะต้องมีข้อดีและข้อเสียแลกเปลี่ยนกัน เพราะเป็นการยากที่จะทำให้สมบูรณ์ได้ทั้งหมด ดังนั้นระบบต่างๆ จึงมักออกแบบมาเพื่องานบางชนิดที่เหมาะสมเท่านั้น

2.2 เอกสารและงานวิจัยที่เกี่ยวข้อง

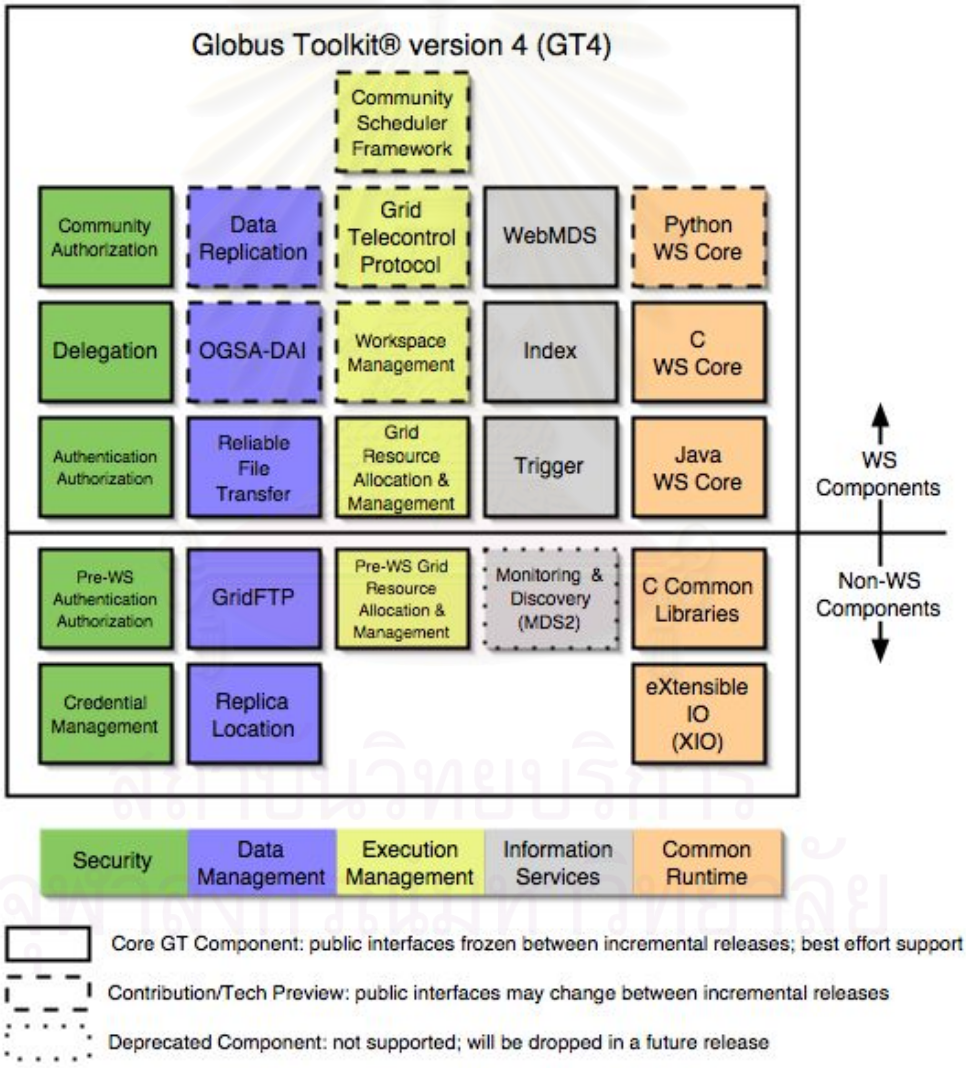
2.2.1 โกลบัสทูลคิท (Globus Toolkit)

โกลบัส ทูลคิท เป็นซอฟท์แวร์โอเพนซอร์ส ที่ใช้สำหรับเทคโนโลยีกริด ประกอบด้วยชุดเครื่องมือต่างๆ สำหรับการสร้างกริด ส่วนโปรแกรมย่อยเขียนด้วยภาษาซีและจาวา ส่วนที่เป็นภาษาซีใช้บนระบบปฏิบัติการยูนิกซ์ รวมถึงลินุกซ์ ส่วนที่เป็นจาวาจะเป็นช่องทางสำหรับทุกแพลตฟอร์ม โกลบัสทูลคิทพัฒนาขึ้นตามมาตรฐานของ GGF ชุดเครื่องมือประกอบด้วยซอฟท์แวร์และคลังคำสั่งสำหรับการเฝ้าสังเกตทรัพยากร (monitoring), การค้นหา (discovery), การจัดการ (management) และรวมถึงความปลอดภัย (security) และ การจัดการแฟ้มข้อมูล (file management)

โกลบัสทูลคิทประกอบด้วยบริการประเภทต่างๆดังนี้

1. การจัดการทรัพยากร (Resource management) เป็นการจัดสรรทรัพยากรที่เกี่ยวกับการสั่งงานเป็นหลัก ใช้โพรโทคอลที่มีชื่อว่า Grid Resource Allocation & Management Protocol (GRAM)

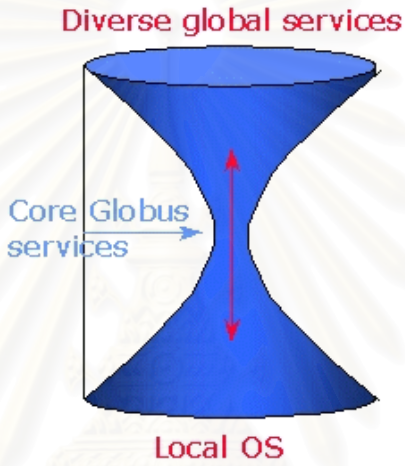
2. บริการสารสนเทศ (Information Services) ใช้สำหรับการค้นหาและตรวจสอบทรัพยากรต่างๆภายในเครือข่าย โดยใช้ Monitoring and Discovery Service (MDS)
3. บริการความปลอดภัย (Security Services) จัดการเกี่ยวกับความปลอดภัยของระบบ เช่นการพิสูจน์ตัวตน การตรวจสอบสิทธิ์ เป็นต้น โดยใช้ Grid Security Infrastructure (GSI)
4. การจัดการข้อมูล (Data Management) เป็นบริการสำหรับถ่ายโอนข้อมูล ใช้ Global Access to Secondary Storage (GASS) และ GridFTP



รูปที่ 2-4 ส่วนประกอบของโกลบัส ทูลคิท*

* http://www.globus.org/

ชุดเครื่องมือประกอบด้วยซอฟท์แวร์สำหรับความปลอดภัย, โครงสร้างพื้นฐานสารสนเทศ (information infrastructure), การจัดการทรัพยากร (resource management), การจัดการข้อมูล (data management), การสื่อสาร (communication), การทนต่อความผิดพร่อง (fault detection), และสามารถเคลื่อนย้ายได้ (portability) โปรแกรมแต่ละส่วนสามารถทำงานร่วมกันหรือแยกจากกันก็ได้ โกลบัสทูลคิทช่วยให้องค์กรที่มีระบบการบริหารทรัพยากรที่แตกกันสามารถใช้งานร่วมกันได้โดยผ่านทางบริการที่เป็นมาตรฐานกลางของโกลบัส และผู้ใช้สามารถเข้าถึงทรัพยากรในระยะไกลได้เสมือนกับใช้ทรัพยากรของตนเองด้วยระบบการทำงานของบริการหลัก (core service), ส่วนต่อประสาน (interface) และ โพรโทคอล (protocol) [17]



รูปที่ 2-5 สถาปัตยกรรมของโกลบัส ทูลคิท[†]

สถาปัตยกรรมของโกลบัสทูลคิทเป็นรูปนาฬิกาทรายดังรูปที่ 2-5 ตรงส่วนกลางของนาฬิกาทราย ซึ่งเป็นส่วนที่เล็กที่สุดนั้นคือบริการหลักของโกลบัส หรือเรียกว่า มิดเดิลแวร์ เป็นบริการพื้นฐานที่เป็นมาตรฐานสำหรับกริด ซึ่งคั่นระหว่างชั้นโปรแกรมประยุกต์ที่หลากหลายในระดับบน กับชั้นทรัพยากรที่หลากหลายในระดับล่าง

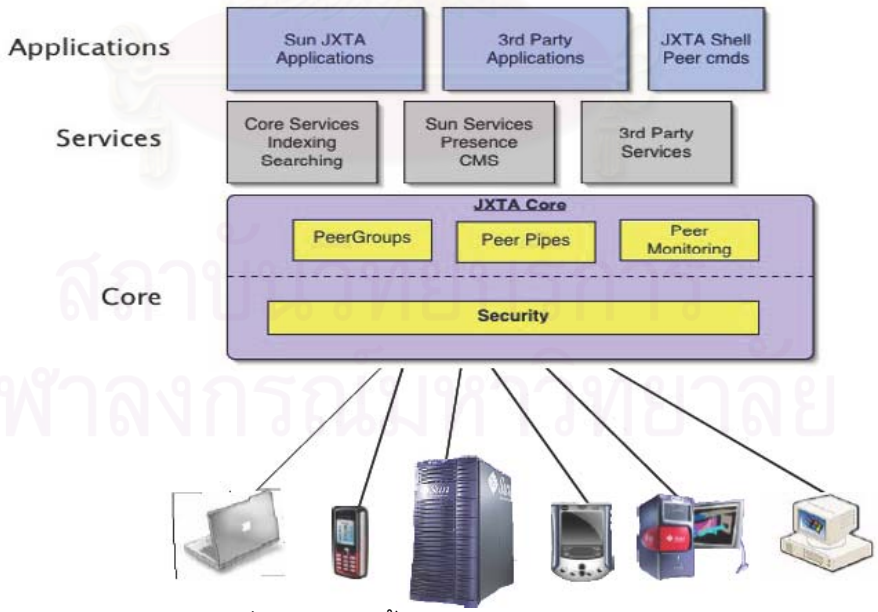
ปัจจุบัน โกลบัสทูลคิท พัฒนามาถึงรุ่น 4 ซึ่งได้เพิ่มเติมความสามารถของเว็บเซอร์วิสเข้ามาใช้ร่วมกับเทคโนโลยีกริดด้วย และยังมีการพัฒนาต่อไปเรื่อยๆ เพื่อทำให้เป็นมิดเดิลแวร์ที่สมบูรณ์ยิ่งขึ้น และเปิดให้ผู้สนใจร่วมกันออกแบบและพัฒนาเทคโนโลยีของกริดอีกด้วย

[†] http://dast.nlanr.net/

2.2.2 จักซ์ตา (JXTA)

จักซ์ตา [27] เป็นแพลตฟอร์มโอเพนซอร์ส สำหรับพัฒนาโปรแกรมประยุกต์แบบเพียร์-ทู-เพียร์ โดยมีโพรโทคอลพื้นฐานสำหรับการสื่อสารในเครือข่ายเพียร์-ทู-เพียร์ที่ใช้ภาษาเอ็กซ์เอ็มแอล (XML) เป็นหลัก จุดมุ่งหมายของจักซ์ตาคือการสร้างแพลตฟอร์มที่สนับสนุนโปรแกรมประยุกต์แบบกระจายในหลากหลายรูปแบบ และสามารถทำงานบนอุปกรณ์ดิจิตอลใดๆ โครงการจักซ์ตาเผยแพร่โดยบริษัทซันไมโครซิสเต็มส์ (Sun Microsystems) โดยมีเป้าหมายคือ

- Interoperability โดยการทำให้สามารถการระบุตำแหน่งของเพียร์ทำได้ง่าย สามารถทำกิจกรรมร่วมกันได้ง่าย และสามารถให้บริการซึ่งกันและกันข้ามเครือข่ายเพียร์-ทู-เพียร์ที่ต่างระบบและต่างกลุ่มได้
- Platform independence จักซ์ตาถูกออกแบบให้ไม่ขึ้นอยู่กับแพลตฟอร์ม เช่น ภาษาโปรแกรมใดๆ (เช่น ซี, จาวา, ฯลฯ), ระบบปฏิบัติการใดๆ (เช่น ระบบปฏิบัติการวินโดวส์, ระบบปฏิบัติการยูนิกซ์, ฯลฯ), และแพลตฟอร์มของเครือข่าย (เช่น ทีซีพี/ไอพี (TCP/IP), บลูทูท (Bluetooth), ฯลฯ)
- Ubiquity จักซ์ตาถูกออกแบบให้สามารถสร้างโปรแกรมได้บนอุปกรณ์ดิจิตอลทุกชนิด ได้แก่ เครื่องใช้ไฟฟ้า เครื่องคอมพิวเตอร์ และระบบเก็บข้อมูล



รูปที่ 2-6 ลำดับชั้นการทำงานของจักซ์ตา[‡]

[‡] JXTA v2.3.x: Java Programmer's Guide (Sun Microsystems, Inc., 2005), p.9

จักซ์ตาประกอบไปด้วย 3 ชั้นการทำงาน คือ

- ชั้นแกนกลาง (Core Layer) เป็นส่วนที่ใช้สำหรับควบคุมการทำงานของโพรโทคอลต่างๆ ซึ่งจะทำให้เพียร์สามารถค้นหาข้อมูล, เข้ากลุ่ม, สร้างกลุ่ม และแลกเปลี่ยนข้อความได้
- ชั้นบริการ (Service Layer) ถูกสร้างอยู่บนชั้นแกนกลาง ซึ่งอาจเป็นบริการที่ไม่จำเป็นจะต้องใช้สำหรับโปรแกรมประยุกต์บางโปรแกรม เช่น ระบบหน่วยเก็บข้อมูล, การแบ่งปันแฟ้ม, ระบบแฟ้มแบบกระจาย, การพิสูจน์ตัวตน และโครงสร้างกุญแจสาธารณะ เป็นต้น
- ชั้นโปรแกรมประยุกต์ (Application Layer) ในชั้นนี้เป็นส่วนที่ผู้พัฒนาจะเป็นคนออกแบบฟังก์ชันการทำงานของโปรแกรมประยุกต์แบบเพียร์-ทู-เพียร์

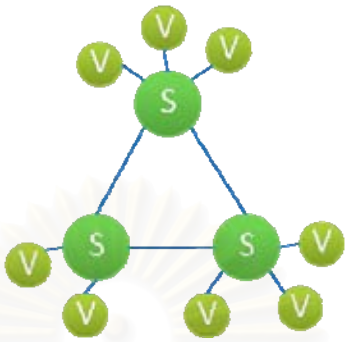
เครือข่ายจักซ์ตาประกอบด้วย เพียร์ (peer), กลุ่มเพียร์ (peer group) และไปป์ (pipe) โดยที่เพียร์จะเป็นสมาชิกในกลุ่มเพียร์ อย่างน้อยเพียร์ที่ไม่ได้อยู่ในกลุ่มเพียร์เฉพาะใดๆจะอยู่ในกลุ่มเน็ตเพียร์กรุ๊ป (NetPeerGroup) ซึ่งเป็นกลุ่มมาตรฐานของจักซ์ตา เพียร์จะติดต่อกับเพียร์อื่นๆผ่านทางไปป์ ซึ่งเป็นช่องทางการสื่อสารที่ใช้ในการส่งข้อความและข้อมูล ทรัพยากรในจักซ์ตาเช่น เพียร์, กลุ่มเพียร์ , ไปป์ และ บริการต่างๆจะถูกอธิบายด้วยการโฆษณา (advertisement) ซึ่งแต่ละเพียร์จะค้นหาทรัพยากรเหล่านี้ได้โดยค้นหาจากโฆษณา

2.2.3 **ฮาร์เวส** (Harvest)

โปรแกรมฮาร์เวส [15] เป็นระบบการคำนวณแบบกระจายบนเครือข่ายเพียร์-ทู-เพียร์ พัฒนาขึ้นบนจามจุรีเซอร์วิส [28] ซึ่งเป็นคลังคำสั่งระดับสูงที่ทำงานบนแพลตฟอร์มจักซ์ตาอีกชั้นหนึ่ง ฮาร์เวสใช้สถาปัตยกรรมแบบซูเปอร์เพียร์ สมาชิกในกลุ่มเพียร์ประกอบด้วยซูเปอร์เพียร์ (super peer) และโวลันเทียร์ (volunteer) โดยที่ซูเปอร์เพียร์จะทำหน้าที่ในการเก็บรวบรวมข้อมูลของทรัพยากรในระบบ ได้แก่ข้อมูลทางด้านกายภาพของโวลันเทียร์ต่างๆ และข้อมูลของงานที่โวลันเทียร์สั่งเข้ามา และยังมีหน้าที่แจกจ่ายงานไปยังเพียร์ที่เหมาะสม ส่วนโวลันเทียร์จะเป็นเพียร์ที่ทำการประมวลผลตามงานที่ได้รับมอบหมายจากซูเปอร์เพียร์ และโวลันเทียร์เองก็สามารถเป็นเพียร์ที่ส่งงานเข้าสู่ระบบเพื่อประมวลผลได้เช่นกัน ในกลุ่มเพียร์หนึ่งอาจประกอบด้วยหลายซูเปอร์เพียร์และ หลายโวลันเทียร์

ฮาร์เวสเป็นโปรแกรมประเภทแย่งรอบการทำงาน (cycle stealing) โดยจะส่งงานไปรันบนเครื่องอื่น ซึ่งว่างจากการทำงานโดยปกติ ในการตรวจสอบเครื่องที่ว่างจากการทำงานนั้น

ฮาร์เวสได้ใช้โปรแกรมถนอมหน้าจอ (screen saver) ที่พัฒนาขึ้นเอง โดยเมื่อผู้ใช้คอมพิวเตอร์ไม่ได้ใช้เครื่องเป็นเวลานานจะทำให้โปรแกรมถนอมหน้าจอเริ่มทำงาน โปรแกรมฮาร์เวสก็จะเริ่มทำงานด้วย และจะรับงานมารันโดยอัตโนมัติ



รูปที่ 2-7 การเชื่อมต่อด้วยโปรแกรมฮาร์เวส

ในงานวิจัยนี้ได้นำโปรแกรมฮาร์เวสมาปรับปรุง เพื่อให้สามารถเชื่อมต่อได้กับระบบกริด รวมทั้งได้นำมาใช้ร่วมกับระบบแฟ้มแบบกระจายที่พัฒนาขึ้นใหม่ ซึ่งรายละเอียดการเปลี่ยนแปลงและการพัฒนาจะกล่าวถึงในหัวข้อ 4.1

2.2.4 **จีไอเอสพี** (GISP)

ในการบริหารจัดการข้อมูลจำนวนมากในเครือข่ายเพียร์-ทู-เพียร์นั้น วิธีที่จะช่วยให้เครือข่ายมีประสิทธิภาพมากขึ้นก็คือการสร้างโครงสร้างให้กับระบบ วิธีการจัดการโครงสร้างของเพียร์-ทู-เพียร์ที่นิยมใช้กันมากในปัจจุบันก็คือตารางแฮชแบบกระจาย ตัวอย่างของระบบที่พัฒนาวิธีตารางแฮชแบบกระจายได้แก่ CAN[19], Chord[20], Pastry[21], Tapestry[22]

จีไอเอสพี [29] เป็นอีกโครงการหนึ่งที่พัฒนาตารางแฮชแบบกระจาย โดยเน้นไปที่การพัฒนาให้ใช้งานได้จริง โครงการนี้มีจุดประสงค์ดังนี้

- สร้างกลไกการแบ่งปันข้อมูลอย่างมีประสิทธิผลมากกว่าการแพร่สัญญาณ (broadcast) ในเครือข่ายเพียร์-ทู-เพียร์
- เพื่อใช้งานกับเครือข่ายเพียร์-ทู-เพียร์แบบแยกศูนย์อย่างเต็มที่ (fully decentralized peer-to-peer network) และพิจารณาถึงสมรรถนะที่แตกต่างกันของเพียร์ด้วย
- เน้นไปที่โพรโทคอลที่ง่าย แต่ไม่ได้เน้นที่ความสมบูรณ์ของข้อมูลสารสนเทศ

- ติดตั้งได้ง่ายบนเครือข่ายเพียร์-ทู-เพียร์ เช่น จักซ์ตา
- ใช้ได้จริงบนเครือข่ายจริงที่อาจถูกโจมตีได้

จีไอเอสพีใช้การฟังก์ชันแฮชกับข้อมูล และเพียร์ไอดี เพื่อให้ได้ค่าแฮชเป็นตัวเลขขนาด n บิต (ค่า n สามารถเปลี่ยนได้แล้วแต่การเขียนโปรแกรม) ฟังก์ชันแฮชสามารถใช้ขั้นตอนวิธี (algorithm) ใดก็ได้ แต่ต้องใช้เหมือนกันทั้งเครือข่าย และสิ่งสำคัญที่จีไอเอสพีจะใช้ในการจัดการข้อมูลก็คือ คำหลัก (keyword) โดยที่คำหลักที่ใช้ของแต่ละข้อมูลอาจจะไม่ต้องสื่อถึงความหมายของข้อมูลก็ได้ การเลือกใช้คำหลักใดๆ ขึ้นอยู่กับการเขียนโปรแกรมประยุกต์

เพียร์ในเครือข่ายจีไอเอสพีจะมีการแลกเปลี่ยนข้อมูลของเพียร์และข้อมูลที่เพียร์เก็บไว้อยู่ตลอดเวลา โดยการส่งต่อแบบเพียร์ต่อเพียร์ และหลีกเลี่ยงการแพร่สัญญาณโดยสิ้นเชิง ทำให้ปริมาณการใช้แบนด์วิดท์ในเครือข่ายต่ำ นอกจากนี้ข้อมูลย่อยที่จะแลกเปลี่ยนนั้นอาจถูกรวมเข้าเป็นข้อมูลชุดเดียวกัน แล้วส่งไปพร้อมกัน เพื่อประหยัดเวลาในการค้นหาเส้นทางส่งข้อมูล

เนื่องจากในเครือข่ายเพียร์-ทู-เพียร์มีความไม่แน่นอนสูง เพียร์อาจจะเข้าและออกจากเครือข่ายได้ทุกเมื่อโดยไม่คาดคิด จีไอเอสพีจึงแก้ปัญหาโดยการทำซ้ำของข้อมูล (replication) กล่าวคือ คำหลักหนึ่งนั้นจะถูกส่งไปยังเพียร์รับผิดชอบจำนวน n เพียร์ที่มีค่าระยะห่างในเชิงตัวเลข (numerical distance) น้อยที่สุดโดยพิจารณาจากค่าแฮช ทั้งนี้เพื่อสร้างส่วนซ้ำสำรอง (redundancy) ให้กับข้อมูล และหากมีเพียร์ที่รับผิดชอบเพียร์ใดออกจากเครือข่าย ก็จะมีกลไกการหาเพียร์ใหม่ขึ้นมารับผิดชอบแทน กรณีที่แย่ที่สุดที่จะทำให้ข้อมูลหายไปก็คือ เพียร์ทั้ง n เพียร์นั้นออกจากเครือข่ายพร้อมกัน

ความแตกต่างระหว่างเพียร์ (heterogeneity) มีผลต่อความรับผิดชอบของเพียร์ แต่ละเพียร์จะมีค่า peer strength และจะมีผลกับค่าระยะห่างด้วยสูตร $\text{distance}(X, Y)/2^{L-1}2^{M-1}$ เมื่อ X, Y เป็นค่าแฮชระหว่าง 2 เพียร์ และ L, M เป็นค่า peer strength ค่า peer strength นี้จะถูกกระจายในเครือข่ายด้วย

จีไอเอสพีสร้างเครือข่ายซ้อนทับ (overlay network) ขึ้น โดยไม่สนใจโครงสร้างเครือข่ายทางกายภาพ เพียร์ที่อยู่ติดกันในจีไอเอสพี อาจจะอยู่ห่างกันมากในเครือข่ายจริง ซึ่งในกรณีนี้อาจทำให้ระบบเสียเวลาในการแลกเปลี่ยนข้อมูลได้ ดังนั้นจึงอาจมีการคำนวณระยะห่างด้วยการเพิ่มปัจจัยเวลาแฝง (latency) ลงไปด้วย

จีไอเอสพีถูกพัฒนาครั้งแรกบนจักซ์ตา โดยใช้เพียร์ไอดี (PeerID) เป็นตัวกำหนดที่อยู่ โดยการส่งข้อความจะใช้บริการเอ็นด์พอยต์ (Endpoint Service) ส่งข้อความเอ็นด์พอยต์ (Endpoint Message) การค้นหาเส้นทางและค้นหาเพียร์ในเครือข่ายใช้โพรโทคอลต่างๆของจักซ์ตา

บทที่ 3

แนวคิดของงานวิจัย

ในบทนี้เป็นการวิเคราะห์ปัญหาของระบบต่างๆ ที่เกี่ยวข้องในปัจจุบัน เช่น กริด คลัสเตอร์ การคำนวณแบบอินเทอร์เน็ต การคำนวณแบบเพียร์-ทู-เพียร์ ซึ่งแต่ละระบบจะมีข้อดีและข้อเสียต่างกัน ดังนั้นจึงมีการเปรียบเทียบข้อดีและข้อเสียของแต่ละระบบ และส่วนต่อมาเป็นการเสนอแนวคิดเบื้องต้นของระบบที่จะพัฒนาขึ้นในงานวิจัยนี้ ที่จะเป็นการแก้ปัญหาบางอย่างและการรวบรวมข้อดีของระบบต่างๆไว้ด้วยกัน รวมถึงวิเคราะห์ความเป็นไปได้ของระบบ

3.1 ระบบการคำนวณแบบกระจาย

ระบบการคำนวณแบบกระจายในปัจจุบันมีความหลากหลาย ทั้งทางด้านสถาปัตยกรรม โครงสร้างพื้นฐาน โพรโทคอล แพลตฟอร์ม ซึ่งปัจจัยเหล่านี้ล้วนทำให้ระบบการคำนวณแบบกระจายมีความสามารถ ประสิทธิภาพ และระดับของการบริการแตกต่างกันไป ในที่นี้จะทำการเปรียบเทียบระบบกริด เพียร์-ทู-เพียร์ และคลัสเตอร์ เพื่อแสดงข้อดีและข้อเสียของแต่ละระบบ

เทคโนโลยีกริด เพียร์-ทู-เพียร์ และคลัสเตอร์ ต่างก็เป็นเทคโนโลยีที่เกี่ยวข้องกัน และแต่ละเทคโนโลยีต่างก็มีข้อได้เปรียบและเสียเปรียบซึ่งกันและกัน ทำให้ระบบบริการต่างๆ ที่มีอยู่บนเทคโนโลยีเหล่านี้แตกต่างกันไปด้วย ดังจะยกตัวอย่างของความแตกต่างระหว่างแต่ละเทคโนโลยี โดยพิจารณาจากฟังก์ชันสำหรับระบบการคำนวณแบบกระจายได้เป็น

1. การค้นหาและการจัดสรรทรัพยากร (Resource Discovery / Allocation)
2. การสั่งงาน และการจัดการกระทำการ (Job submission/Execution Management)
3. ระบบแฟ้ม และการถ่ายโอนข้อมูล (File system/Data transfer)
4. ความปลอดภัย (Security)

ตารางที่ 3-1 สรุปเทคโนโลยีที่ใช้ในการคำนวณแบบกระจายของกริด เพียร์-ทู-เพียร์ และคลัสเตอร์

ตารางที่ 3-1 เทคโนโลยีที่ใช้ในการคำนวณแบบกระจายของกริด เพียร์-ทู-เพียร์ และคลัสเตอร์

ฟังก์ชัน	กริด	เพียร์-ทู-เพียร์	คลัสเตอร์
การค้นหา และจัดสรรทรัพยากร (Resource Discovery/Allocation)	MDS : (GRIS, GIIS)	Unstructured: flooding + TTL + dynamic query Structured : DHT	centralized monitoring
ความปลอดภัย (Security)	GSI : PKI, X.509	BFT, TLS, trust-recommendation	SSH, secure master node
การสั่งงาน และการจัดการกระทำการ (Job Submission/ Execution)	GRAM, meta scheduler	replication	local scheduler
ระบบแฟ้ม และการถ่ายโอนข้อมูล (File System / Data Transfer)	GASS, GridFTP, RFT, RLS	DHT	NFS

3.1.1 **การค้นหาและการจัดสรรทรัพยากร** (Resource Discovery / Allocation)

การค้นหาทรัพยากรในระบบคำนวณแบบกระจาย เป็นการสำรวจทรัพยากรที่มีในระบบและเลือกทรัพยากรเพื่อให้ได้ตรงตามคุณสมบัติที่ต้องการและเพียงพอที่จะใช้สำหรับประมวลผล ระบบแบบกระจายจะต้องมีการประกาศข้อมูลของทรัพยากรของตนเอง ข้อมูลที่ประกาศประกอบด้วย

- ฮาร์ดแวร์และซอฟท์แวร์ เช่น ความเร็วซีพียู, ขนาดหน่วยความจำ, ระบบปฏิบัติการ ,ซอฟท์แวร์สำหรับประมวลผลเฉพาะด้าน เป็นต้น
- สถานะของทรัพยากร เช่น เวลาเดินเครื่องเปล่า(idle time), เวลาที่เปิดให้ใช้บริการ, หน่วยความจำที่ว่าง, หน่วยเก็บข้อมูลที่ว่าง, ภาระงาน (work load)
- สถานะของเครือข่าย เช่น แบนด์วิดท์ (bandwidth)

ปัญหาที่สำคัญของการค้นหาทรัพยากรในระบบแบบกระจายคือการค้นหาทรัพยากรที่เหมาะสมให้ได้อย่างรวดเร็วและครอบคลุม ในระบบกริดที่มีการจัดการแบบลำดับชั้น (Hierarchy) เครื่องประมวลผลจะทำหน้าที่ประกาศข้อมูลทรัพยากรของตนเองไปที่โหนดที่สูงกว่าซึ่งจะเป็นโหนดที่รวบรวมข้อมูลของทรัพยากรกลุ่มหนึ่ง แต่ละโหนดเหล่านี้อาจจะประกาศข้อมูลที่รวบรวมได้นี้ไปยังโหนดที่อยู่สูงกว่าอีกชั้นหนึ่ง เป็นลำดับ ดังนั้นที่โหนดระดับสูงสุดจะรับรู้ข้อมูล

ทรัพยากรของทุกเครื่อง ในโกลบัสทูลคิท มอดูลที่ทำหน้าที่นี้คือ MDS (Monitoring and Discovery System) ซึ่งประกอบด้วย GRIS(Grid Resource Information Service) GIIS (Grid Index Information Service) ทำหน้าที่ในการประกาศทรัพยากร และรวบรวมทรัพยากรตามลำดับ

ส่วนในระบบเพียร์-ทู-เพียร์ การค้นหาทรัพยากรของแต่ละระบบอาจจะทำงานต่างกัน ระบบเพียร์-ทู-เพียร์แบบไม่มีโครงสร้างนั้น การค้นหาทรัพยากรจะเป็นลักษณะการกระจายทุกทิศทาง (flooding) แต่ละเพียร์จะมีข้อมูลของทรัพยากรของตัวเองไว้ การค้นหาทรัพยากรทำได้โดยกระจายข้อความค้นหาออกไปทุกทิศทาง วิธีนี้จะทำให้ปริมาณการใช้งานเครือข่ายสูงมากโดยไม่จำเป็นและอาจเกิดข้อความย้อนกลับได้ จึงมีเทคนิคอื่นๆ เพิ่มเติมเช่น เพิ่มแท็ก Time-To-Live (TTL) แนบไปกับข้อความค้นหา โดยที่ TTL เป็นตัวเลขจำนวนน้อยๆ ซึ่งแสดงจำนวนครั้งของการส่งข้อความค้นไปในเครือข่าย การตั้งค่า TTL น้อยๆจะช่วยลดปริมาณข้อความในเครือข่ายได้ แต่ก็จะลดความครอบคลุมด้วย เทคนิคอีกแบบหนึ่งคือ การสอบถามแบบพลวัต(Dynamic querying) คือการจำกัดจำนวนการส่งข้อความ โดยไม่กระจายไปทุกทิศทาง แต่จะกระจายไปยังบางกลุ่มเท่านั้น นอกจากนี้ยังมีเทคนิคอื่นๆอีกมากมายในการค้นหาทรัพยากรบนเครือข่ายเพียร์-ทู-เพียร์แบบไม่มีโครงสร้าง ซึ่งแต่ละวิธีก็จะมีข้อแลกเปลี่ยนระหว่างความครอบคลุมกับปริมาณการใช้งานเครือข่าย

ในระบบเพียร์-ทู-เพียร์แบบมีโครงสร้าง ใช้หลักการของตารางแฮชแบบกระจาย ซึ่งจะใช้สำหรับการค้นหาทรัพยากรที่จับคู่แบบตรงพอดี ได้ภายในเวลา O(log N) เมื่อ N คือขนาดของคีย์สเปซ ซึ่งทำให้เพียร์-ทู-เพียร์แบบมีโครงสร้างนั้นสามารถขยายได้ (scalable) รองรับกับปริมาณเพียร์ที่เพิ่มขึ้นได้จำนวนมาก ลักษณะเช่นนี้เหมาะสำหรับระบบร่วมแฟ้มที่มีเพียร์จำนวนมากแบ่งปันแฟ้มของตน และมักจะใช้ชื่อแฟ้มเป็นคีย์ในการค้นหา แต่ในการค้นหาทรัพยากรในระบบกริดนั้นต่างกัน เนื่องจากทรัพยากรบนระบบกริดมีหลายประเภท เช่น ซีพียู หน่วยความจำ ระบบปฏิบัติการ ซอฟท์แวร์ เป็นต้น ซึ่งอาจจะต้องใช้หลายตารางประกอบกัน และทรัพยากรบางอย่างบนกริดเป็นแบบพลวัต คือมีการเปลี่ยนแปลงอยู่ตลอดเวลาตามสภาพการใช้งาน เช่น ปริมาณหน่วยความจำที่เหลือที่เวลาต่างๆอาจไม่เท่ากัน ดังนั้นตารางแฮชแบบกระจายแบบเดิมอาจไม่เหมาะสม นอกจากนี้ ทรัพยากรบางอย่างในกริด สามารถกำหนดเป็นช่วงพิสัยได้ เช่น ต้องการหน่วยความจำขนาด 20 – 50 เมกะไบต์ หรือ ความเร็วซีพียู 500 เมกะเฮิรตซ์ขึ้นไป เช่นนี้ก็ทำให้ใช้ตารางแฮชแบบเดิมไม่ได้ เนื่องจากตารางแฮชนั้นจะต้องจับคู่กับค่าที่แน่นอนเท่านั้น ปัญหาเหล่านี้เป็นอุปสรรคสำหรับระบบเพียร์-ทู-เพียร์ จึงมีงานวิจัยหลายงานที่พยายามแก้ปัญหาการค้นหาทรัพยากรบนเพียร์-ทู-เพียร์

ระบบเพียร์-ทู-เพียร์อีกรูปแบบหนึ่งคือโครงสร้างแบบซูเปอร์เพียร์ ที่มีซูเปอร์เพียร์ทำหน้าที่เหมือนเครื่องบริการ (Server) โดยจะรวบรวมข้อมูลของทรัพยากรที่เป็นโหนดปลายอื่นๆ ที่อยู่ใกล้เคียง ซูเปอร์เพียร์หลายตัวจะเชื่อมต่อกัน เมื่อมีการค้นหาทรัพยากรเกิดขึ้น จะถูกส่งคำร้องไปที่ซูเปอร์เพียร์ ซูเปอร์เพียร์จะเริ่มค้นหาจากข้อมูลที่ตนมีอยู่ หากไม่มีทรัพยากรที่เหมาะสมก็จะส่งคำร้องนั้นต่อไปยังซูเปอร์เพียร์อื่นๆ ซึ่งโดยรวมแล้วทำงานคล้ายกับระบบ กริดที่มีอยู่ในปัจจุบัน และเพิ่มความสามารถในการค้นหาโดยใช้การเก็บสถิติของความสำเร็จอีกด้วย

3.1.2 **การสั่งงาน และการจัดการกระทำการ** (Job submission/Execution Management)

กริดมีการจัดการเกี่ยวกับการสั่งงาน ซึ่งประกอบไปด้วยการส่งงาน, การจัดเตรียมทรัพยากร และการจัดการช่วงชีวิตของงาน บริการลักษณะนี้ในโกลบัสทูลคิทคือ แกรม (GRAM: Grid Resource Allocation and Management) โดยที่แกรมจะมีมาตรฐานของส่วนต่อประสาน (interface) สำหรับการสั่งงาน และการจัดการทรัพยากรในระยะไกล ผู้ใช้และผู้พัฒนาโปรแกรมประยุกต์จะใช้เมท็อดเดียวสำหรับเข้าถึงระบบการจัดการท้องถิ่น (local management system) ที่แตกต่างกันออกไป ผู้ใช้สามารถสั่งงานเชิงโต้ตอบ (interactive job), งานแบบกลุ่ม (batch job), การตรวจสอบสถานะของงาน (monitor) และยกเลิกงาน (cancel) ได้ นอกจากนี้ยังมีตัวจัดการงานในระดับกริด เรียกว่า ตัวจัดลำดับงานระดับบน (meta-scheduler) ที่จะเลือกกริดโหนดที่เหมาะสมให้กับงานต่างๆ

ในส่วนคลัสเตอร์นั้นเกี่ยวเนื่องกับกริด กล่าวคือ กริดมักจะมีโหนดประมวลผลเป็นเครื่องคอมพิวเตอร์สมรรถนะสูง รวมถึงคลัสเตอร์ ในคลัสเตอร์เองเมื่อรับงานเข้ามาแล้วก็จะมีการบริหารทรัพยากรภายในคลัสเตอร์อีกเช่นกัน คลัสเตอร์สามารถทำงานได้ทั้งในเชิงโต้ตอบ (interactive job) และงานแบบกลุ่ม (batch job) คลัสเตอร์สามารถใช้เอ็มพีไอ (MPI: message passing interface) เพื่อให้ทำงานแบบขนานได้ โดยที่ทุกเครื่องในคลัสเตอร์จะทำงานไปด้วยกันและส่งข้อความถึงกันเมื่อต้องมีการส่งข้อมูลที่จำเป็นให้กัน นอกจากนี้ คลัสเตอร์ยังมีตัวจัดลำดับงาน (scheduler) สำหรับเป็นตัวจ่ายงานไปยังเครื่องในคลัสเตอร์ที่ว่างงาน ซึ่งเป็นการกระจายภาระของการคำนวณให้เท่าเทียมกันทั้งคลัสเตอร์

ในการแจกจ่ายงานบนเพียร์-ทู-เพียร์นั้น หลายโปรแกรมประยุกต์มีวิธีที่แตกต่างกัน แต่มีลักษณะคล้ายกันคือ งานที่รันบนเพียร์-ทู-เพียร์มักจะมีข้อมูลเข้า (input) และข้อมูลออก (output) ขนาดเล็ก และใช้เวลาในการคำนวณไม่มาก เนื่องจากโอกาสที่เครื่องที่รันงานจะออกไปจากระบบนั้นมีมาก ลักษณะงานที่เหมาะสมสำหรับเพียร์-ทู-เพียร์จึงเป็นแบบมาสเตอร์/เวิร์กเกอร์ (master/ worker) ที่เป็นการแจกจ่ายงานย่อยไปทำที่เครื่องคำนวณ และเมื่อทำงานเสร็จก็ส่ง

ผลลัพธ์กลับมา ซึ่งหากงานใดที่ทำไม่สำเร็จ ก็สามารถส่งงานนั้นไปทำที่เพียร์อื่นได้ ส่วนการทำงานแบบขนานเช่น เอ็มพีไอ (MPI) นั้นสามารถทำได้ แต่อาจจะไม่เหมาะสมสำหรับสภาพแวดล้อมบนเพียร์-ทู-เพียร์นัก เนื่องจากความเป็นพลวัตของเพียร์-ทู-เพียร์ และความเร็วของเครือข่ายที่ไม่เอื้ออำนวย การเพิ่มโอกาสที่จะทำให้งานแบบขนานทำได้สำเร็จก็คือการทำซ้ำ (replication) โดยมีเพียร์ที่รับทำหน้าที่เดียวกันหลายๆเพียร์เพื่อป้องกันความผิดพลาด

3.1.3 **ระบบแฟ้ม และการถ่ายโอนข้อมูล** (File system/Data transfer)

แฟ้มข้อมูลเป็นส่วนที่สำคัญสำหรับการประมวลผลแบบกระจาย การจัดการระบบแฟ้มในระบบกระจายจะต้องเกี่ยวข้องกับการถ่ายโอนแฟ้ม, การทำซ้ำ, และการอัปเดทข้อมูล

ในโกลบัสทูลคิทมีบริการที่เกี่ยวข้องกับข้อมูล ได้แก่กริดเอฟทีพี (GridFTP), อาร์เอฟที (RFT: Reliable File Transfer) และอาร์แอลเอส (RLS: Replica Location Service) กริดเอฟทีพีเป็นโพรโทคอลสำหรับถ่ายโอนแฟ้มข้อมูลบนกริด โดยพัฒนาขึ้นบนเอฟทีพี (FTP) และเพิ่มระบบความปลอดภัยในการควบคุมช่องทางสื่อสาร สนับสนุนการถ่ายโอนแฟ้มโดยบุคคลที่สาม (Third-party transfer) และได้ปรับปรุงประสิทธิภาพการถ่ายโอนแฟ้ม เช่น ปรับขนาดบัฟเฟอร์ของทีซีพีและเพิ่มฟังก์ชั่นการถ่ายโอนแบบขนานได้ อาร์เอฟทีเป็นฟังก์ชั่นที่เพิ่มเติมความน่าเชื่อถือให้กับการถ่ายโอนแฟ้ม โดยการสร้างจุดตรวจสอบ (checkpoint) เพื่อเก็บสถานะของการถ่ายโอน และสามารถกู้สถานะการถ่ายโอนเมื่อมีความผิดพลาดของระบบใดๆ ส่วนอาร์แอลเอสเป็นฟังก์ชั่นที่เพิ่มการทำซ้ำของข้อมูล โดยมีกลไกสำหรับการลงทะเบียนและระบุตำแหน่งของข้อมูล

ในคลัสเตอร์นั้น การจัดการเกี่ยวกับแฟ้มข้อมูลระหว่างเครื่องจะมีระบบจัดการแฟ้มของคลัสเตอร์ ซึ่งแฟ้มอาจจะถูกแบ่งและกระจายไปอยู่ตามเครื่องต่างๆ โดยที่ผู้ใช้จะมองเห็นเป็นแฟ้มเดียวกัน นอกจากนี้ คลัสเตอร์มักจะเชื่อมต่อด้วยแลน (LAN) ดังนั้นจึงใช้ระบบแฟ้มเอ็นเอฟเอส (NFS: Network File System) ซึ่งช่วยให้การจัดการแฟ้มข้อมูลทำได้สะดวกขึ้น

ส่วนในระบบแฟ้มบนเพียร์-ทู-เพียร์นั้น ใช้หลักการของตารางแฮชแบบกระจายซึ่งใช้ได้ดีสำหรับการจัดการกับแฟ้มเนื่องจากแฟ้มเป็นทรัพยากรที่ไม่เปลี่ยนแปลงบ่อยเมื่อเทียบกับทรัพยากรอื่นๆเช่น ซีพียู หรือหน่วยความจำ อย่างไรก็ตามหากมีการเปลี่ยนแปลงแฟ้ม ก็จะต้องมีวิธีการจัดการกับรุ่น (version) ของแฟ้มด้วย ซึ่งมีงานวิจัยจำนวนมากได้เสนอวิธีการจัดการกับเวอร์ชันของแฟ้มที่มีการเปลี่ยนแปลง

เนื่องจากความไม่เสถียรของระบบเพียร์-ทู-เพียร์ เพียร์สามารถเข้าและออกจากเครือข่ายได้ตลอดเวลา ดังนั้นจึงต้องเพิ่มความน่าเชื่อถือของระบบแฟ้มบนเพียร์-ทู-เพียร์ วิธีหนึ่งคือการทำซ้ำ นั่นคือแฟ้มข้อมูลเดียวจะถูกเก็บไว้ที่หลายเพียร์ ซึ่งอาจจะกระจายด้วยกลไกของตารางแฮชแบบกระจาย นอกจากนี้ความไม่เสถียรของเพียร์-ทู-เพียร์ยังมีผลกับการถ่ายโอนแฟ้มอีกด้วย เพราะการถ่ายโอนแฟ้มจะผิดพลาดหากเพียร์ที่กำลังถ่ายโอนแฟ้มข้อมูลออกจากระบบไป ระบบเพียร์-ทู-เพียร์หลายระบบได้แก้ไขเรื่องนี้ด้วยการแบ่งการถ่ายโอนแฟ้มแบบขนานไปพร้อมกันจากหลายๆที่ หรืออาจใช้การแบ่งแฟ้มเป็นส่วนย่อย เพื่อเพิ่มโอกาสถ่ายโอนแฟ้มสำเร็จให้มากขึ้น

3.1.4 **ความปลอดภัย** (Security)

ในระบบกริดนั้นประกอบด้วยทรัพยากรจากองค์กรมากมาย และทรัพยากรเหล่านี้มักจะเป็นเครื่องมือที่มีสมรรถนะสูง รวมถึงใช้สำหรับงานที่ใหญ่และสำคัญ ดังนั้นระบบกริดจึงมักต้องการความปลอดภัยสูง ในโกลบัสทูลคิท มีบริการที่ทำหน้าที่ด้านความปลอดภัย ใช้ชื่อว่า Grid Security Infrastructure (GSI) โดยมีพื้นฐานทางด้านความปลอดภัยคือ การเข้ารหัสด้วยกุญแจสาธารณะ (Public key cryptography) และใบรับรอง X.509 (X.509 certificate) โกลบัสใช้ใบรับรอง (certificate) ในการพิสูจน์และแสดงตัวตนผู้ใช้และบริการ และถูกเข้ารหัสด้วยรูปแบบ X.509 ใบรับรอง และกุญแจส่วนบุคคล (private key) จะถูกเก็บอยู่ใน $HOME/.globus ของบัญชีผู้ใช้ และใบรับรองนี้จะต้องสร้างใหม่ทุกช่วงเวลาหนึ่ง ซึ่งผู้ใช้จะได้รับแจ้งทางอีเมล ในการแสดงตัวตนนั้น แต่ละฝ่ายจะต้องพิสูจน์กันและกัน โดยที่ทั้งคู่จะต้องเชื่อ CA ที่รับรองใบรับรองนั้น

ในการเข้าใช้งานเครื่องที่อยู่ระยะไกล เมื่อผ่านการแสดงตัวตนด้วยใบรับรองแล้ว ใบรับรองจะถูกนำไปจับคู่กับบัญชีผู้ใช้ เพื่อให้ได้สิทธิ์เสมือนเป็นผู้ใช้ที่มีบัญชีบนเครื่องนั้นโดยตรง โดยรายชื่อของบัญชีผู้ใช้งานกริดที่สามารถเข้าใช้งานได้จะอยู่ในกริดแมปไฟล์ (grid-mapfile) หากผู้ใช้ไม่มีรายชื่อในกริดแมปไฟล์จะไม่สามารถเข้าใช้ได้

อีกหลักการหนึ่งของกริดคือการลงชื่อเข้าใช้ครั้งเดียว (Single Sign-On) ผู้ใช้สามารถเข้าสู่ระบบกริดได้โดยใส่รหัสผ่านเพียงครั้งเดียว โดยใช้ใบรับรองแทน (Proxy credential) ซึ่งเป็นใบรับรองชั่วคราวที่จะทำให้เข้าใช้งานภายใน กริดได้โดยไม่ต้องใส่รหัสผ่านใหม่ภายในช่วงเวลาหนึ่ง นอกจากนี้ยังสนับสนุนการลงชื่อเข้าใช้ และการถ่ายโอนแฟ้มระยะไกล (remote login and file transfer) โดยใช้ GSI-OpenSSH ซึ่งทำหน้าที่เหมือน OpenSSH โดยเพิ่มฟังก์ชั่นของการลงชื่อเข้าใช้ครั้งเดียวเข้าไปด้วย

ระบบเพียร์-ทู-เพียร์ โดยพื้นฐานมาจากการแบ่งปันแฟ้มข้อมูลของตนเองให้กับผู้อื่น ซึ่งไม่จำเป็นต้องรู้จักคนที่แลกเปลี่ยนแฟ้ม ดังนั้นมาตรฐานของความปลอดภัยจึงไม่ใช่ประเด็นที่จำเป็นสำหรับโปรแกรมประยุกต์ส่วนใหญ่บนเพียร์-ทู-เพียร์ อย่างไรก็ตามระบบเพียร์-ทู-เพียร์ก็ได้พัฒนากลไกสำหรับจัดการเกี่ยวกับความปลอดภัยไว้บางส่วน ซึ่งอาจพบได้ในโปรแกรมประยุกต์บางโปรแกรม ระบบความปลอดภัยบนเพียร์-ทู-เพียร์เป็นสิ่งที่ทำได้ยากกว่าแบบระบบรับ-ให้บริการ เนื่องจากสภาพแวดล้อมของเพียร์-ทู-เพียร์ที่เป็นการจัดการแบบกระจาย นั่นคือแต่ละเพียร์มีหน้าที่รับผิดชอบทรัพยากรของตนเอง และอาจมีการตรวจสอบสภาพแวดล้อมรอบตัวเองบ้าง ในการติดต่อกับเพียร์อื่นที่ไม่รู้จักมาก่อนซึ่งไม่อาจรู้ได้ว่าเพียร์นั้นเป็นเพียร์ที่ดีหรือไม่ เพียร์ที่ไม่ดีอาจปล่อยข้อมูลที่เสียเข้ามาในเครือข่าย หรืออาจโจมตีเครือข่าย และทำให้เพียร์อื่นไม่สามารถทำงานได้ตามปกติ

ระบบเพียร์-ทู-เพียร์แบบมีโครงสร้าง แต่ละเพียร์จะมีไอดีของตัวเอง ซึ่งได้มาจากการสุ่ม ระบบแบบนี้จะมีการจัดการเส้นทางด้วยตัวเอง และมีการทำซ้ำของข้อมูล (replication) ดังนั้น การค้นหาแฟ้มในเครือข่ายก็จะเจอแฟ้มที่ต้องการในที่สุด ประกอบกับการเพิ่มวิธีการพิสูจน์บางอย่าง เช่น อัลกอริทึมที่ทนต่อความผิดพร่องแบบไบแซนไทน์ (Byzantine-fault tolerant algorithm: BFT) และการรับรองตนเอง (self certify) เป็นต้น ทำให้ช่วยป้องกันการโจมตีจากเพียร์ภายนอกได้ อย่างไรก็ตาม หากมีเพียร์ที่มุ่งร้ายจำนวนมากพอ ก็จะทำให้ระบบเดิมเสียหายได้เช่นกัน อีกแนวทางหนึ่งของการเพิ่มความปลอดภัยในระบบเพียร์-ทู-เพียร์คือการสร้างระบบความไว้ใจ (trust) ขึ้น โดยที่แต่ละเพียร์จะบันทึกระดับความไว้ใจต่อเพียร์อื่นๆ หากเพียร์ที่ติดต่อกันให้ผลของการติดต่อถูกต้องและเสร็จสมบูรณ์ก็จะเพิ่มระดับความไว้ใจซึ่งกันและกัน และในทางตรงกันข้าม หากผลการติดต่อผิดพลาดหรือไม่พึงประสงค์ ความไว้ใจจะลดลงไป จนถึงระดับหนึ่งก็จะไม่ติดต่อกับเพียร์นั้นอีก ต่อมาได้มีการเพิ่มเติมระบบการแนะนำ (recommendation) ขึ้น โดยหลักการคล้ายกับระบบความไว้ใจ แต่จะเพิ่มเติมโดยมีการกระจายคำแนะนำต่อเพียร์นั้นไปยังเพียร์อื่นๆด้วย ทำให้เพียร์อื่นๆรับรู้ถึงความไว้ใจได้ต่อเพียร์นั้นเพิ่มขึ้น

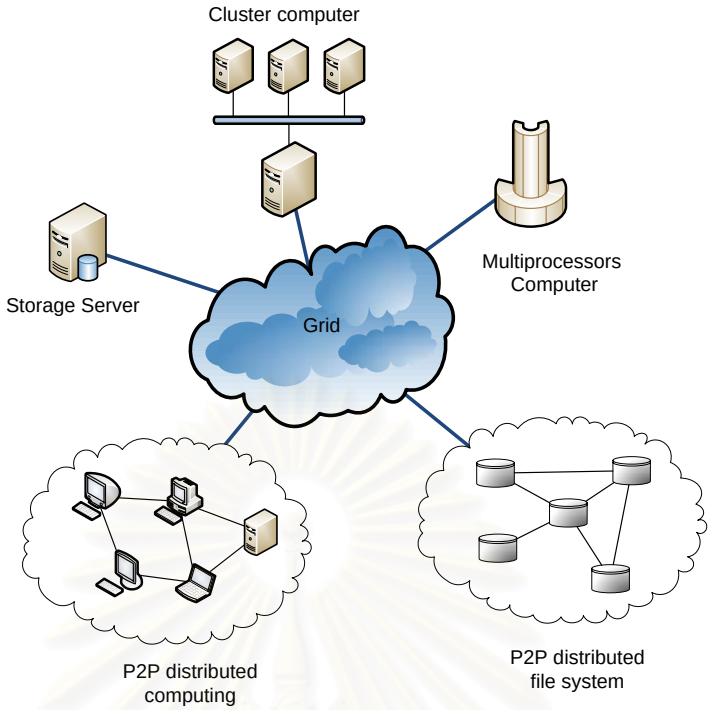
ในคลัสเตอร์ คอมพิวเตอร์ที่เชื่อมต่อกันในคลัสเตอร์อยู่ในสภาพแวดล้อมส่วนตัวและใช้เครือข่ายส่วนตัว ดังนั้นจึงไม่จำเป็นต้องมีความกังวลในเรื่องความปลอดภัยมากนัก การเข้าใช้งานเครื่องคลัสเตอร์นั้นจัดการโดยระบบปฏิบัติการ เครื่องคอมพิวเตอร์ที่ต่อกับภายนอกมีเพียงเครื่องมาสเตอร์เท่านั้น ดังนั้นการเพิ่มระบบความปลอดภัยของคลัสเตอร์สามารถทำได้โดยเพิ่มซอฟท์แวร์ความปลอดภัยที่เครื่องมาสเตอร์

เมื่อกล่าวโดยสรุป ระบบเพียร์-ทู-เพียร์จะเน้นในเรื่องการรวบรวมทรัพยากรที่หลากหลาย และมีจำนวนมากได้ดี เนื่องจากภาระจะกระจายอยู่ที่เครือข่ายทั้งหมด แต่จะมีข้อเสียในเรื่องการบริการที่ไม่มีมาตรฐานเดียวกัน ความไม่เสถียร โอกาสผิดพร่องสูงกว่า จึงต้องมีการพัฒนาวิธีป้องกันต่างๆมากขึ้นตามความเหมาะสมของแต่ละบริการ กริดนั้นเน้นที่โครงสร้างที่เข้มแข็งและปลอดภัย แต่อาจยังมีข้อด้อยในเรื่องการจัดการทรัพยากรจำนวนมาก ส่วนคลัสเตอร์นั้นเนื่องจากอยู่ในสภาพแวดล้อมปิด จึงสามารถจัดการได้ง่าย และไม่สร้างปัญหากับระบบโดยรวมมาก

การขยายขนาดของกริดวิธีหนึ่งคือการเพิ่มคลัสเตอร์เข้าไปเป็นทรัพยากรของกริด ซึ่งคลัสเตอร์ที่เพิ่มมักจะเป็นเครื่องเฉพาะงาน และองค์กรต้องจัดสรรค่าใช้จ่ายในการจัดซื้อและดูแล หากองค์กรมีการต้องการใช้คลัสเตอร์เพิ่มขึ้นก็จะต้องมีการจัดสถานที่เพื่อติดตั้งอุปกรณ์เพิ่มเติม ซึ่งจะต้องใช้ค่าใช้จ่าย เวลา และแรงงานอีกด้วย ดังนั้นหากเรามีระบบคลัสเตอร์ที่จัดการได้ง่าย ขยายตัวได้ดี และใช้ค่าใช้จ่ายน้อย ก็จะทำให้องค์กรสามารถจัดการได้สะดวกขึ้น ซึ่งเทคโนโลยีที่อาจจะช่วยได้ก็คือเพียร์-ทู-เพียร์

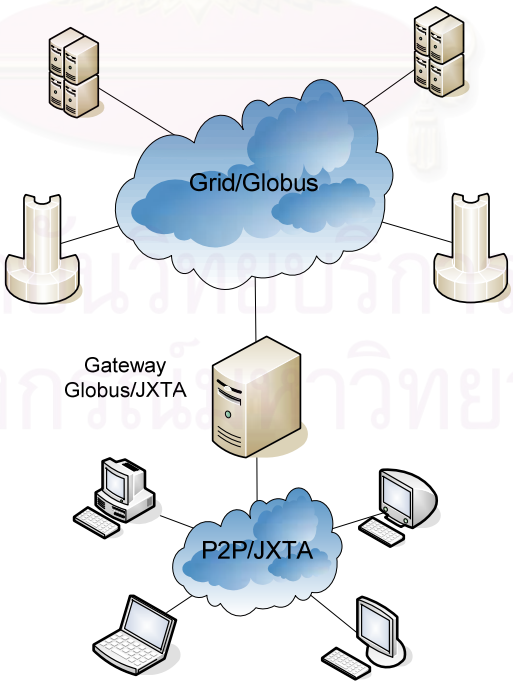
3.2 แนวคิดหลัก

เป้าหมายของงานวิจัยนี้คือการเชื่อมต่อระหว่างระบบกริดกับระบบเพียร์-ทู-เพียร์ โดยให้ผู้ใช้จากกริดสามารถใช้บริการของระบบเพียร์-ทู-เพียร์ได้ แต่การที่จะนำเครื่องคอมพิวเตอร์ส่วนบุคคลแต่ละเครื่องในระบบเพียร์-ทู-เพียร์ ไปเชื่อมต่อกับกริดโดยตรงนั้นไม่น่าจะเกิดประโยชน์ เนื่องจากสมรรถภาพของเครื่องอาจไม่เพียงพอสำหรับให้บริการ ผู้วิจัยจึงเสนอแนวคิดโดยให้เป็นกลุ่มเพียร์ (peer group) ที่ให้บริการร่วมกัน และพัฒนากลไกสำหรับเชื่อมต่อกับกริดเพื่อให้ผู้ใช้ในกริดสามารถใช้บริการของกลุ่มเพียร์ได้ ผ่านทางกลไกของโกลบัส ดังนั้นบริการต่างๆในกลุ่มเพียร์นั้นจะทำหน้าที่ให้บริการในลักษณะคล้ายกับคลัสเตอร์คอมพิวเตอร์



รูปที่ 3-1 ภาพรวมโครงสร้างของระบบ

เนื่องจากระบบความปลอดภัยที่เข้มงวดของกริด ทำให้เราไม่สามารถนำทรัพยากรเพียร์-ทู-เพียร์เข้าไปรวมกับกริดได้ทันที จึงต้องหาวิธีอื่นเพื่อเชื่อมต่อระหว่างสองระบบ ผู้วิจัยได้เสนอให้มีเครื่องหนึ่งที่เป็นส่วนหนึ่งของกริด และเป็นส่วนหนึ่งของเพียร์-ทู-เพียร์ในเวลาเดียวกัน เพื่อเป็นทางเชื่อมต่อของสองระบบ เครื่องนี้เรียกว่า “เกตเวย์” (Gateway)



รูปที่ 3-2 โครงสร้างเครือข่ายที่เชื่อมต่อระหว่างเพียร์-ทู-เพียร์และกริด

ผู้วิจัยเลือกใช้โกลบัสทูลคิทเป็นมิดเดิลแวร์ของระบบกริด เนื่องจากเป็นมาตรฐานของกริด และมีเครื่องมือที่พร้อมให้ใช้มากมาย และส่วนเพียร์-ทู-เพียร์นั้น ผู้วิจัยเลือกใช้จักซ์ตา เนื่องจากเป็นแพลตฟอร์มที่ใช้พัฒนาโปรแกรมประยุกต์ได้ง่าย และงานที่พัฒนาขึ้นมาก่อนหน้านี้ที่จะนำมารวมในงานวิจัยถูกพัฒนาขึ้นบนจักซ์ตาทั้งหมด จึงเป็นการง่ายที่จะใช้จักซ์ตาเป็นพื้นฐานของระบบ

3.3 โกลบัสทูลคิท และมอดูลที่ขยายได้

โกลบัสใช้สถาปัตยกรรมแบบนาฬิกาทราย ที่แกนกลางเป็นบริการหลักสำหรับเชื่อมต่อระหว่างโปรแกรมประยุกต์ที่หลากหลายในระดับบน และจัดการกับทรัพยากรที่หลากหลายในระดับล่าง สถาปัตยกรรมแบบนี้จึงช่วยให้ผู้พัฒนากริดสามารถพัฒนาโปรแกรมประยุกต์และบริการที่เหมาะสมกับเครือข่ายกริดของตนเองได้ โดยที่ยังสามารถเชื่อมต่อกับเครือข่ายกริดอื่นๆได้โดยไม่จำกัดความสามารถ

ในงานวิจัยนี้เกี่ยวข้องกับการนำเครือข่ายเพียร์-ทู-เพียร์เข้ามารวมเป็นทรัพยากรของระบบกริด แต่เนื่องจากสถาปัตยกรรมพื้นฐานของระบบเพียร์-ทู-เพียร์และกริดนั้นต่างกัน ดังนั้นจึงต้องสร้างส่วนต่อประสานระหว่างกริดกับเพียร์-ทู-เพียร์ เพื่อแลกเปลี่ยนข้อมูลและการทำงานกันระหว่างสองระบบ ซึ่งเกี่ยวข้องกับการพัฒนามอดูลที่ขยายได้ของกริด ในงานวิจัยนี้จะเน้นการสร้างบริการของกริด 2 บริการ คือ การจัดการงาน (job management) และการจัดการข้อมูล (data management)

3.3.1 การจัดการงาน

หน่วยย่อยของโกลบัสที่ทำหน้าที่หลักเกี่ยวกับการจัดการงานก็คือ แกรม (GRAM: Grid Resource Allocation Management Service) แกรมเป็นตัวติดต่อกับผู้ใช้เพื่อให้สามารถชี้ตำแหน่ง (locate) สั่งงาน (submit) เฝ้าสังเกต (monitor) และยกเลิก (cancel) งานบนสภาวะแวดล้อมแบบกริด นอกจากนี้ แกรมยังมีส่วนต่อประสาน (interface) ที่ใช้สำหรับติดต่อกับตัวจัดลำดับงานท้องถิ่น (local job scheduler) ต่างๆ ได้ เช่น SGE [30], PBS [31], LSF [32] เป็นต้น โดยจะต้องการสร้างตัวจัดการงาน (job manager) สำหรับตัวจัดลำดับงานนั้นๆ

สำหรับระบบคำนวณแบบเพียร์-ทู-เพียร์นั้น ฮาร์เวสถือได้ว่าเป็นโปรแกรมจัดการงานระดับท้องถิ่น และมีฟังก์ชันของตัวเองในการจัดลำดับงานด้วย ดังนั้นจึงสามารถสร้างส่วนเชื่อมต่อระหว่างกริดกับเพียร์-ทู-เพียร์ในส่วนการจัดการงานได้ผ่านทางส่วนต่อประสานของแกรม

การพัฒนาส่วนต่อประสานในการจัดการงานของแกรมทำได้โดยคำสั่งภาษาเพิร์ล (perl script) ในคำสั่งนี้จะเป็นการขยายจากตัวจัดการงานมาตรฐานของแกรม (GRAM Job Manager) และต้องเขียนเมท็อดหลัก 3 เมท็อด ดังนี้

1) submit() เป็นเมท็อดที่รองรับการสั่งงาน

2) poll() เป็นเมท็อดที่รองรับการสอบถามสถานะของงาน

3) cancel() เป็นเมท็อดที่รองรับการสั่งยกเลิกงาน

การสั่งงานผ่านทางกริดจะมีการสร้างคำอธิบายงานเป็นภาษาอาร์เอสแอล (RSL: Resource Specification Language) โดยในอาร์เอสแอลจะประกอบด้วยลักษณะประจำ (attribute) ต่างๆ ซึ่งมีลักษณะคล้ายกับโฆษณางาน (Job Advertisement) ของฮาร์เวส ดังนั้นจึงน่าจะสามารถแปลงอาร์เอสแอลเป็นโฆษณางานได้ เพื่อให้สามารถสั่งงานเป็นงานของฮาร์เวสต่อไปได้

กริดรองรับการสอบถามสถานะของงาน ในขณะเดียวกันฮาร์เวสก็มีการเก็บสถานะของงานแต่ละงานเอาไว้อยู่แล้ว ดังนั้นจึงน่าจะแปลงคำสั่งถามสถานะของแกรมให้มาเรียกถามสถานะงานของฮาร์เวสได้เช่นกัน

การยกเลิกงานนั้นไม่ได้รองรับในฮาร์เวส จึงอาจจะไม่สามารถส่งคำสั่งยกเลิกงานไปยังฮาร์เวสเพื่อให้งานนั้นหยุดทำทันทีได้ อย่างไรก็ถาม หากมีการสั่งยกเลิกงานจากกริด ก็สามารถที่จะละทิ้งผลลัพธ์ของงานที่ทำเสร็จแล้วในฮาร์เวสได้

จากเงื่อนไขทั้งหมดทำให้เราสามารถพัฒนาส่วนเชื่อมต่อระหว่างกริดกับเพียร์-ทู-เพียร์ในการจัดการงานได้

3.3.2 การจัดการข้อมูล

พื้นฐานของการจัดการข้อมูลในระดับกริดคือโพรโทคอลกริดเอฟทีพี (GridFTP) ที่เป็นส่วนขยายของเอฟทีพี (FTP) โดยมีระบบความปลอดภัยมากเพิ่มขึ้น, ถ่ายโอนข้อมูลขนาดใหญ่ได้ดีขึ้น และมีความน่าเชื่อถือมากขึ้น กริดเอฟทีพีจึงเป็นมาตรฐานของการถ่ายโอนแฟ้มในระดับกริด ไม่เฉพาะแค่ในโกลบัสทูลคิทเท่านั้น

กริดเอฟทีพีเซิร์ฟเวอร์ ซึ่งเป็นโปรแกรมในฝั่งเซิร์ฟเวอร์ของโพรโทคอลกริดเอฟทีพีนั้นเปิดโอกาสให้สามารถพัฒนาเพิ่มเติมได้ ให้เหมาะสมกับวิธีการจัดการข้อมูลท้องถิ่น ทั้งนี้

เนื่องจากทรัพยากรในการจัดการข้อมูลบนกริดมักจะมีความหลากหลายทางเทคโนโลยี เช่นเดียวกับจามจุรีเอ็กซ์พลอเรอร์ (Jamjuree Explorer) [33] ซึ่งเป็นระบบแฟ้มข้อมูลบนเพียร์-ทู-เพียร์ที่ผู้วิจัยได้พัฒนาขึ้นก็สามารถเป็นหน่วยเก็บข้อมูลสำหรับกริดได้เช่นกัน ดังนั้นจึงต้องมีการพัฒนาส่วนเชื่อมต่อการส่งข้อมูลระหว่างกริดกับเพียร์-ทู-เพียร์

กริดเอฟทีพีเซิร์ฟเวอร์แบ่งได้เป็น 3 มอดูล

1) กริดเอฟทีพีโพรโทคอลมอดูล (GridFTP Protocol Module) เป็นส่วนที่จัดการกับการส่งข้อความผ่านเครือข่ายและโพรโทคอล ส่วนนี้ไม่ควรมีการเปลี่ยนแปลงเพราะจะทำให้เข้ากันไม่ได้ และไม่สามารถสื่อสารกับเซิร์ฟเวอร์อื่นๆได้

2) ฟังก์ชันการแปลงข้อมูล (Data Transform Functionality) เป็นส่วนเสริมของกริดเอฟทีพีที่จะช่วยเพิ่มประสิทธิภาพในการถ่ายโอนข้อมูลได้ อย่างไรก็ตามส่วนนี้ไม่ได้เป็นส่วนหลักของกริดเอฟทีพีและมักไม่ได้ใช้งานบ่อยนัก

3) ส่วนต่อประสานหน่วยเก็บข้อมูล (Data Storage Interface) หรือดีเอสไอ (DSI) เป็นตัวจัดการกับการอ่านและเขียนข้อมูลในระบบเก็บข้อมูลท้องถิ่น ส่วนต่อประสานนี้ประกอบด้วยฟังก์ชันที่ต้องพัฒนา เช่น การส่ง (send), การรับ (receive), คำสั่งอื่นๆ (command) เช่น mkdir, list เป็นต้น มอดูลนี้จะต้องถูกพัฒนาเพิ่มเติมสำหรับหน่วยเก็บข้อมูลเฉพาะ

ในงานวิจัยนี้ได้นำจามจุรีเอ็กซ์พลอเรอร์มาใช้เป็นหน่วยเก็บข้อมูลหลัก ในกรณีนี้การพัฒนาดีเอสไอขึ้นมาจึงเป็นวิธีที่เหมาะสมในการเชื่อมต่อระหว่างกริดเอฟทีพีกับจามจุรี-เอ็กซ์พลอเรอร์ ผู้วิจัยจึงทำการศึกษาและพัฒนาหน่วยดีเอสไอสำหรับจามจุรีเอ็กซ์พลอเรอร์ขึ้นโดยศึกษาจากโปรแกรมโอเพนซอร์สต่างๆ เช่น เอสอาร์บีดีเอสไอ (SRB-DSI) [34] และ เอชพีเอส-เอสดีเอสไอ (HPSS-DSI) [35]

การพัฒนาดีเอสไอ ทำได้โดยการเขียนโปรแกรมภาษาซีที่เป็นส่วนขยายของกริดเอฟทีพีเซิร์ฟเวอร์ โดยจะต้องมีการเขียนฟังก์ชันต่างๆสำหรับจัดการกับหน่วยเก็บข้อมูลท้องถิ่น

3.4 ฮาร์เวส

สถาปัตยกรรมของฮาร์เวสนั้นเป็นแบบซูเปอร์เพียร์ ที่จะมีซูเปอร์เพียร์เป็นเสมือนศูนย์ย่อยของระบบในการจัดสรรทรัพยากร โดยหน้าที่หลักของซูเปอร์เพียร์ก็คือการจัดลำดับงานและอาจมีหน้าที่ในการรับแฟ้มผลลัพธ์แทนผู้สั่งงานในกรณีที่ผู้สั่งงานออกจากระบบไปก่อนงานวิจัย [15] ได้ศึกษาภาระของเครื่องซูเปอร์เพียร์ว่าจะรองรับโวลันเทียร์จำนวนมากได้ดีเพียงใด ซึ่งผลการทดลองด้วยโปรแกรมจำลอง (simulator) นั้นพบว่าซูเปอร์เพียร์สามารถรองรับโวลันเทียร์จำนวนมากได้โดยใช้แบนด์วิดธ์เพียงเล็กน้อยเท่านั้น ในกรณีที่มีการส่งข้อมูลไปยังซูเปอร์เพียร์น้อยๆ

อย่างไรก็ตาม งานวิจัย [15] ไม่ได้ศึกษาภาระของเครื่องอื่นๆที่เกี่ยวข้องในการสั่งงานเช่น เครื่องสั่งงานเอง หรือเครื่องที่ทำงาน เนื่องจากในบางกรณีอาจเกิดปัญหาคอขวดขึ้นได้ที่อื่นนอกจากที่ซูเปอร์เพียร์เอง เช่น ในกรณีที่ผู้สั่งงานได้สั่งงานจำนวนมากเข้าสู่ระบบ ข้อมูลเข้าและข้อมูลออกก็จะเพิ่มตามเป็นทวีคูณ ซึ่งสุดท้ายก็จะต้องใช้แบนด์วิดท์ของผู้สั่งงานจำนวนมากและทำให้เกิดปัญหาคอขวดได้ในที่สุด กรณีดังกล่าวนี้จะเป็นปัญหาที่สำคัญสำหรับงานวิจัยนี้ เพราะผู้สั่งงานในที่นี้ก็คือโหนดหนึ่งในกริดที่จะเป็นทางเชื่อมต่อระหว่างกริดกับเพียร์-ทู-เพียร์

ปัญหาหลักอย่างหนึ่งของระบบฮาร์เวสแบบเดิมนั้นคือเรื่องการถ่ายโอนแฟ้มข้อมูล หากแฟ้มข้อมูลเข้าและออกมีขนาดใหญ่มาก จะเป็นผลให้มีการใช้งานแบนด์วิดท์ของผู้สั่งงานสูง รวมทั้งอาจเพิ่มภาระให้ผู้สั่งงานมากขึ้นได้ ผู้วิจัยจึงได้ออกแบบวิธีการจัดการแฟ้มข้อมูลในฮาร์เวสใหม่ โดยใช้ระบบแฟ้มแบบกระจายบนเพียร์-ทู-เพียร์ที่พัฒนาขึ้น มีชื่อว่า จามจุรีเอ็กซ์พลอเรอร์ (Jamjuree Explorer) [33] ซึ่งเชื่อว่าจะช่วยลดปริมาณการใช้แบนด์วิดธ์ของเครื่องสั่งงานได้

3.5 ระบบแฟ้มแบบกระจายบนเครือข่ายเพียร์-ทู-เพียร์

ผู้วิจัยนำเสนอการนำระบบแฟ้มแบบกระจาย (distributed file system) มาใช้ร่วมกับระบบจัดการงาน เพื่อให้การจัดการแฟ้มมีประสิทธิภาพมากยิ่งขึ้น มีชื่อเรียกว่า จามจุรี-เอ็กซ์พลอเรอร์

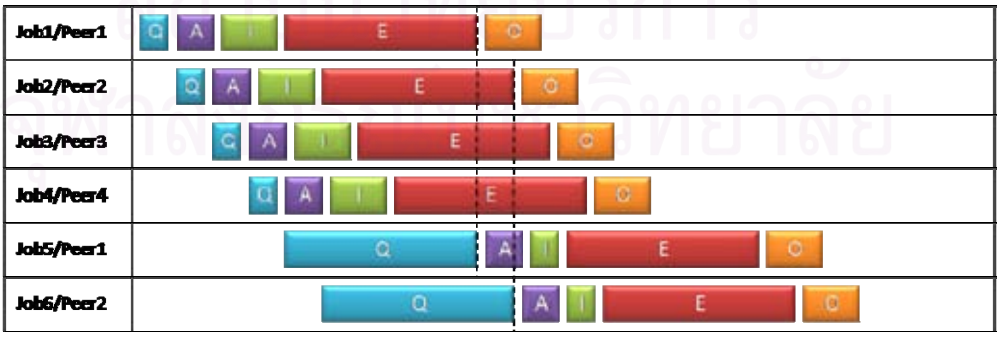
จามจุรีเอ็กซ์พลอเรอร์มีแนวคิดพื้นฐานมาจากระบบร่วมแฟ้ม (file sharing system) ซึ่งเป็นรูปแบบของโปรแกรมประยุกต์บนเครือข่ายเพียร์-ทู-เพียร์ที่เป็นที่นิยมอย่างกว้างขวางในปัจจุบัน เนื่องจากสามารถแบ่งปันแฟ้มข้อมูลได้อย่างรวดเร็ว และได้ใช้ทรัพยากรของทั้งระบบอย่างเต็มที่ คุณสมบัติที่จามจุรีเอ็กซ์พลอเรอร์ได้รับมาจากระบบร่วมแฟ้ม ได้แก่ การ

แบ่งปันแฟ้ม, วิธีการค้นหาแฟ้ม (โดยใช้ตารางแฮชแบบกระจาย) และการดาวน์โหลดแฟ้มแบบขนาน เป็นต้น อย่างไรก็ตามยังมีฟังก์ชันที่เพิ่มเติมจากระบบร่วมแฟ้มทั่วไปบางอย่างที่จะต้องมีเพิ่มเติม เพื่อให้มีลักษณะเป็นระบบแฟ้มมากขึ้น เช่น การควบคุมรุ่น (version control), โครงสร้างระบบแฟ้มแบบลำดับชั้น (file system hierarchy) เป็นต้น

จามจุรีเอ็กซ์พลอเรอร์จะช่วยให้ผู้ใช้มองเห็นหน่วยเก็บข้อมูลขนาดใหญ่ที่เป็นการรวมพื้นที่ว่างของหน่วยเก็บข้อมูลจากเพียร์ต่างๆทั้งเครือข่าย โดยจะมองเห็นโครงสร้างแฟ้มเดียวกันทั้งหมด ดังนั้นผู้ใช้จึงไม่ต้องสนใจว่าข้อมูลจริงจะอยู่ที่ไหน เพียงแค่ระบุชื่อแฟ้มและเส้นทาง (path) ที่ถูกต้องก็จะสามารถเข้าถึงแฟ้มนั้นได้ และหากนำแฟ้มใหม่ไว้ตามโครงสร้างแฟ้มแฟ้มนั้นก็จะถูกมองเห็นได้จากผู้ใช้อื่นๆได้เช่นกัน

จามจุรีเอ็กซ์พลอเรอร์จะช่วยเพิ่มความสามารถให้กับฮาร์เวสโดยจะทำให้ฮาร์เวสลดภาระของการจัดการแฟ้มข้อมูลลงไป ฮาร์เวสจะไม่ต้องสนใจกับการถ่ายโอนแฟ้มข้อมูลต่อไป เพียงแค่ระบุชื่อแฟ้มที่เกี่ยวข้องให้ถูกต้องเท่านั้น และส่วนของการเรียกคืนแฟ้มจะเป็นหน้าที่ของจามจุรีเอ็กซ์พลอเรอร์ทั้งหมด อีกทั้งยังช่วยลดเวลาในกระบวนการทำงานของฮาร์เวสได้อีกด้วย เนื่องจากแฟ้มจะถูกดาวน์โหลดได้เร็วขึ้นด้วยกลไกการดาวน์โหลดแบบขนาน และการแคช

ก) ไม่ใช้จามจุรีเอ็กซ์พลอเรอร์



ข) ใช้จามจุรีเอ็กซ์พลอเรอร์

Q = Queue wait A = Allocate I = Stage-in E = Execute O = Stage-out

รูปที่ 3-3 เปรียบเทียบขั้นตอนการทำงานของฮาร์เวส

รูปที่ 3-3 แสดงขั้นตอนต่างๆในกระบวนการรันงานของโปรแกรมฮาร์เวส เมื่อเปรียบเทียบก่อนและหลังใช้จามจุรีเอ็กซ์พลอเรอร์ จะเห็นว่าจามจุรีเอ็กซ์พลอเรอร์จะมีส่วนช่วยลดเวลาได้ใน 2 ขั้นตอน คือ ขั้นตอนการดาวน์โหลดแฟ้มเข้า (stage-in) แฟ้มจะถูกดาวน์โหลดจากเครือข่าย ซึ่งอาจจะไม่ใช่จากเจ้าของแฟ้มโดยตรง และอาจดาวน์โหลดแบบขนานเพื่อเพิ่มความเร็วได้ ในขณะที่ฮาร์เวสแบบเดิมนั้น แฟ้มจะถูกดาวน์โหลดจากเจ้าของแฟ้มเท่านั้น ทำให้เจ้าของแฟ้มต้องใช้แบนด์วิดท์จำนวนมากในกรณีส่งงานเดียวกันให้เพียร์อื่นๆหลายเพียร์ อีกขั้นตอนคือ หลังจากที่กระทำการ (execute) เสร็จสิ้น แฟ้มออก (output) จะถูกวางไว้ในไดเร็กทอรีแบ่งปัน ซึ่งแฟ้มจะเข้าสู่ระบบแฟ้มแบบกระจายโดยอัตโนมัติ และผู้สั่งงานจะมาสามารถดาวน์โหลดผลลัพธ์กลับไปได้ในเวลาต่อมา เป็นอันเสร็จสิ้นงานนั้น และเพียร์นั้นจะสามารถรับงานต่อไปได้ทันที ในขณะที่ฮาร์เวสแบบเดิมนั้น เพียร์ที่ทำงานจะต้องเสียเวลาในการส่งแฟ้มออก (stage-out) กลับไปยังเพียร์ที่สั่งงานก่อน แล้วจึงจะรับงานใหม่เข้ามาได้

จามจุรีเอ็กซ์พลอเรอร์นั้นไม่ได้ใช้เครือข่ายซ้อนทับ (overlay network) แบบซูเปอร์เพียร์ดังเช่นฮาร์เวส แต่เป็นแบบเสมอภาคกันทุกเพียร์ ดังนั้นการทำงานของฮาร์เวสกับจามจุรีเอ็กซ์พลอเรอร์จึงไม่เกี่ยวข้องกันและทำงานแยกจากกัน อย่างไรก็ตามพื้นฐานของการเชื่อมต่อระหว่างเพียร์ยังใช้จักซ์ตาซึ่งเป็นลักษณะแบบซูเปอร์เพียร์

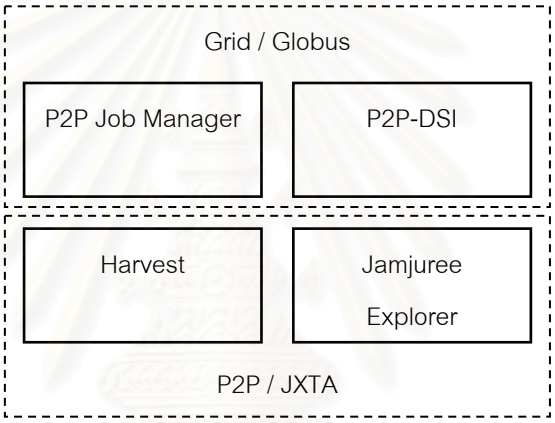
3.6 ระบบหลายเกตเวย์

โปรแกรมฮาร์เวสสนับสนุนการสั่งงานได้จากโวลันเทียร์ทุกเพียร์ ดังนั้นเมื่อนำมารวมกับระบบกริดโดยผ่านเกตเวย์นั้น ก็เป็นไปได้ที่จะมีเกตเวย์หลายเครื่อง ทั้งนี้เพื่อช่วยลดปัญหาคอขวดที่อาจเกิดจากการทำงานผ่านเกตเวย์เพียงเครื่องเดียว

บทที่ 4

การพัฒนาระบบต้นแบบ

ในบทนี้จะเป็นการอธิบายโครงสร้าง ส่วนประกอบ และการทำงานของระบบที่ผู้วิจัยได้พัฒนาขึ้นตามที่ได้ออกแบบไว้ ซึ่งมีชื่อว่าจามจุรีคลัสเตอร์ (Jamjuree Cluster) โดยสามารถแบ่งแยกออกได้เป็น 4 ส่วน แต่ละส่วนจะทำหน้าที่แตกต่างกันและใช้เทคนิคในการพัฒนาที่แตกต่างกัน ส่วนท้ายบทจะเป็นการรวมส่วนประกอบต่างๆเหล่านี้เข้าด้วยกัน โดยจะกล่าวรวมถึงปัญหาเมื่อนำไปติดตั้งบนสภาพแวดล้อมจริง ซึ่งเป็นผลมาจากความแตกต่างระหว่างเทคโนโลยีกริดกับเพียร์-ทู-เพียร์ และกล่าวถึงวิธีการแก้ไขปัญหาเหล่านั้น



รูปที่ 4-1 ส่วนประกอบของจามจุรีคลัสเตอร์

ระบบจามจุรีคลัสเตอร์ประกอบด้วยส่วนย่อย 4 ส่วน ดังรูปที่ 4-1

1. ฮาร์เวส (Harvest)
2. จามจุรีเอ็กซ์พลอเรอร์ (Jamjuree Explorer)
3. ตัวจัดการงานเพียร์-ทู-เพียร์ (JobManager-P2P)
4. เพียร์-ทู-เพียร์ดีเอสไอ (P2P-DSI)

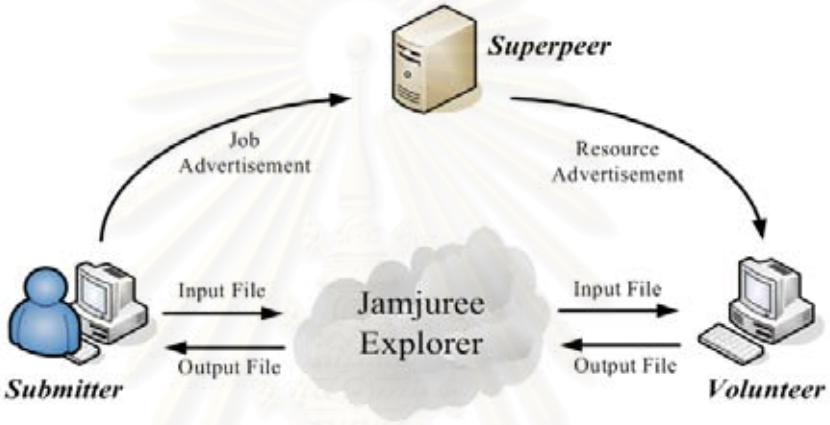
แต่ละส่วนมีรายละเอียดดังนี้

4.1 ฮาร์เวส (Harvest)

ฮาร์เวสเป็นส่วนหนึ่งของงานวิจัย [15] ซึ่งได้ถูกพัฒนาขึ้นมาเป็นรุ่นแรก ในงานวิจัยนี้ได้นำฮาร์เวสในรุ่นแรกมาแก้ไขปรับปรุงใหม่ เพื่อใช้กับระบบแฟ้มแบบเพียร์-ทู-เพียร์ และเพื่อเชื่อมต่อกับระบบกริด ซึ่งฮาร์เวสที่ปรับปรุงใหม่มีลักษณะดังนี้

4.1.1 **โพรโทคอลการสั่งงาน** (Job Submission Protocol)

โพรโทคอลสำหรับการสั่งงานบนฮาร์เวสนั้นถูกออกแบบไว้สำหรับความเป็นพลวัตของเครือข่ายเพียร์-ทู-เพียร์ กล่าวคือ แต่ละเพียร์นั้นสามารถจะเข้าสู่ระบบและออกจากระบบที่เวลาใดก็ได้ ซึ่งอาจเป็นผลเนื่องมาจากความไม่เสถียรของเครือข่าย ความผิดพลาดของเครื่องคอมพิวเตอร์แต่ละเครื่อง รวมไปถึงความต้องการของผู้ใช้เครื่องคอมพิวเตอร์เอง ดังนั้นฮาร์เวสจึงมีโพรโทคอลที่ป้องกันความผิดพลาดที่อาจเกิดขึ้นได้ในระดับหนึ่ง เพื่อรักษากระบวนการทำงานโดยรวมให้ดำเนินต่อไปได้ กระบวนการในการสั่งงาน แสดงได้ดังรูปที่ 4-2



รูปที่ 4-2 กระบวนการสั่งงานด้วยฮาร์เวส

การสั่งงานเริ่มจาก

1) ผู้ใช้เตรียมแฟ้มนำเข้า (input files) และคำบรรยายลักษณะงาน (job description) โดยในคำบรรยายลักษณะงานจะประกอบด้วยคำสั่งที่ใช้ทำงาน คุณลักษณะของโวลันเทียร์ที่จะทำงานได้ และชื่อแฟ้มข้อมูลที่เกี่ยวข้อง
2) ผู้ใช้วางแฟ้มข้อมูลไว้ที่โฮมไดเร็กทอรี (home directory) ของตนเอง แฟ้มข้อมูลจะถูกกระจายไปในเครือข่ายโดยอัตโนมัติ
3) ผู้ใช้สั่งงานไปยังซูเปอร์เพียร์ด้วยโปรแกรมสั่งงาน (Commander)
4) ซูเปอร์เพียร์รับงานที่ได้ และค้นหาโวลันเทียร์ที่เหมาะสมสำหรับงานนั้น
5) ซูเปอร์เพียร์ส่งข้อมูลของโวลันเทียร์นั้นไปยังเพียร์ที่สั่งงาน
6) เพียร์สั่งงานส่งคำบรรยายลักษณะงานไปยังโวลันเทียร์
7) โวลันเทียร์เริ่มรันงาน โดยเริ่มจากการค้นหาและดาวน์โหลดแฟ้มนำเข้า และเริ่มรันงานจนเสร็จ

8) โวลันเทียร์ส่งข้อความทำงานสำเร็จไปยังเพียร์ที่สั่งงาน และซูเปอร์เพียร์ เพื่อเตรียมตัวรับงานต่อไป

9) เพียร์สั่งงานค้นหาและดาวน์โหลดแฟ้มส่งออก (output files)

แฟ้มข้อมูลที่เกี่ยวข้องทั้งหมดจะถูกทำสำเนาและกระจายไว้ที่เพียร์ต่างๆ โดยอัตโนมัติด้วยการทำงานของจามจุรีเอ็กซ์พลอเรอร์ ซึ่งจะกล่าวถึงการทำงานไว้ที่หัวข้อ 4.2

4.1.2 การค้นหาและการเชื่อมต่อกับซูเปอร์เพียร์

ในเครือข่ายเพียร์-ทู-เพียร์ ความผิดพลาดในการเชื่อมต่อเป็นปัจจัยสำคัญที่อาจเกิดขึ้นได้ตลอดเวลา ผู้วิจัยจึงได้ออกแบบการเชื่อมต่อให้เสถียรและยืดหยุ่นมากขึ้นดังนี้

1) การค้นหาซูเปอร์เพียร์

การค้นหาซูเปอร์เพียร์ใช้กลไกการโฆษณา (advertisement) ของจักซ์ตา โดยจามจุรีเซอร์วิสซึ่งเป็นคลังคำสั่งหลัก จะทำการค้นหาข้อมูลซูเปอร์เพียร์ในเครือข่ายอยู่เป็นระยะๆ และจะเก็บไว้ในแคชของตัวเอง เมื่อเพียร์โวลันเทียร์เริ่มทำงานจะค้นหาข้อมูลซูเปอร์เพียร์จากแคชของตัวเอง แล้วจึงใช้ข้อมูลเหล่านั้นในการติดต่อกับซูเปอร์เพียร์ต่างๆ ซึ่งหากติดต่อกับซูเปอร์เพียร์ตัวใดได้สำเร็จ ก็จะรับงานจากซูเปอร์เพียร์นั้นเป็นหลัก แต่หากติดต่อไม่สำเร็จ ก็จะพยายามติดต่อกับซูเปอร์เพียร์ตัวต่อไปเรื่อยๆ และหากไม่สามารถติดต่อได้เลย ก็จะเว้นช่วงระยะเวลาหนึ่งเพื่อรอผลการค้นหาซูเปอร์เพียร์ต่อไป

2) การเชื่อมต่อกับซูเปอร์เพียร์

หลังจากที่เชื่อมต่อกับซูเปอร์เพียร์ได้แล้ว จะมีการตรวจสอบการเชื่อมต่อเป็นระยะๆ โดยการส่งข้อความไปหาซูเปอร์เพียร์ในลักษณะเดียวกับการติดต่อซูเปอร์เพียร์ในครั้งแรก หากไม่สามารถติดต่อได้ ก็จะเข้าสู่กระบวนการค้นหาซูเปอร์เพียร์เช่นเดิม อย่างไรก็ตาม ช่วงเวลาในการค้นหาของทั้งสองกระบวนการนี้แตกต่างกัน โดยในช่วงที่ค้นหาซูเปอร์เพียร์นั้น จะมีอัตราการลองใหม่ที่ถี่กว่าช่วงหลังจากเชื่อมต่อได้แล้ว

กระบวนการดังกล่าว นอกจากจะทำให้เพียร์ต่างๆรักษาสภาพการทำงานโดยรวมได้แล้ว ยังช่วยในการเฝ้าสังเกตข้อมูลของเพียร์โดยรวมของทั้งเครือข่ายด้วย เนื่องจากซูเปอร์เพียร์จะสามารถตรวจสอบการคงอยู่ของเพียร์ต่างๆได้จากข้อความที่ถูกส่งมา

4.1.3 **ไดเร็กทอรีทำงาน** (Working directory)

ไดเร็กทอรีทำงานถูกกำหนดขึ้นเพื่อเป็นที่รันงาน และเป็นที่ของการวางแฟ้มนำเข้าและแฟ้มส่งออกต่างๆในระหว่างการรันงาน การที่ต้องมีไดเร็กทอรีทำงานก็เพื่อให้แฟ้มที่อาจเกิดจากการรันงานไม่ไปรบกวนกับระบบแฟ้มข้อมูลอื่นๆของเครื่อง

4.1.4 **แบบวิธีกระทำการ** (Execution mode)

ฮาร์เวสรุ่นแรกถูกพัฒนาขึ้นและทดสอบโดยใช้เครื่องเป็นระบบปฏิบัติการวินโดวส์ ในงานวิจัยนี้ ผู้วิจัยต้องการรวมเข้าฮาร์เวสกับระบบกริดด้วย ซึ่งในระบบกริดนั้นมักจะใช้ระบบปฏิบัติการลินุกซ์เป็นหลัก จึงอนุมานได้ว่างานบางส่วนที่นำมารันบนฮาร์เวสนั้นจะเป็นงานบนระบบกริดที่ต้องการรันบนระบบปฏิบัติการลินุกซ์ ผู้วิจัยจึงได้ปรับวิธีการรันงานให้มีความหลากหลาย โดยพัฒนาระบบแบบวิธีกระทำการ (Execution mode) ขึ้น ซึ่งเป็นการกำหนดประเภทของงานที่จะทำ ตัวอย่างของแบบวิธีกระทำการได้แก่

- unix_cmd คือ การรันคำสั่งทั่วไปบนระบบปฏิบัติการยูนิกซ์ (หรือลินุกซ์) เช่น ls, hostname, mkdir เป็นต้น
- unix_sh คือ การรันคำสั่งประเภทเชลสคริปต์ (shell script) บนระบบปฏิบัติการยูนิกซ์ (หรือลินุกซ์)
- dos_cmd คือ การรันคำสั่งทั่วไปบนระบบปฏิบัติการดอส (หรือวินโดวส์)
- dos_batch คือ การรันคำสั่งประเภทแบทช์ (batch) บนระบบปฏิบัติการดอส (หรือวินโดวส์)

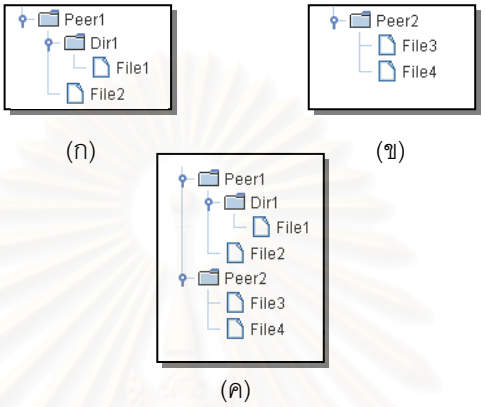
แบบวิธีกระทำการจะจับคู่กับจาวาคลาสใดๆ ที่ขยายจากคลาส Executor โดยคลาสนี้จะถูกโหลดด้วยจาวาคลาสโหลดเดอร์ในเวลารัน (runtime)

ผู้วิจัยได้ออกแบบให้สามารถเพิ่มเติมแบบวิธีกระทำการได้ในภายหลัง โดยมีส่วนต่อประสาน (interface) ของภาษาจาวา ให้ผู้พัฒนาสามารถพัฒนาเพิ่มเติมได้

ในการสั่งงานด้วยฮาร์เวส จำเป็นจะต้องระบุแบบวิธีกระทำการในคำบรรยายลักษณะงานด้วยทุกครั้ง เช่น execMode=unix_sh,unix_cmd เป็นต้น และโวลันเทียร์ที่เข้าร่วมในฮาร์เวสก็จะต้องระบุแบบวิธีกระทำการที่ตนสามารถทำได้เช่นกัน ซึ่งซูเปอร์เพียร์จะจับคู่งานกับโวลันเทียร์โดยพิจารณาจากแบบวิธีกระทำการด้วย

4.2 **จามจุรีเอ็กซ์พลอเรอร์** (Jamjuree Explorer)

จามจุรีเอ็กซ์พลอเรอร์ [33] เป็นระบบใช้แฟ้มร่วมกัน (file sharing system) ที่ให้เพียร์สามารถแบ่งปัน (share) แฟ้มของตนเองและให้เพียร์อื่นสามารถค้นหาและดาวน์โหลดแฟ้มได้ แฟ้มจะถูกจัดในลักษณะเป็นลำดับชั้น (hierarchy) และทุกเพียร์จะมองเห็นโครงสร้างแฟ้มข้อมูลเหมือนกัน และถูกพัฒนาความสามารถให้คล้ายคลึงกับระบบแฟ้มในปัจจุบัน



รูปที่ 4-3 โครงสร้างข้อมูลของจามจุรีเอ็กซ์พลอเรอร์

จามจุรีเอ็กซ์พลอเรอร์ถูกพัฒนาขึ้นโดยใช้หลักการเดียวกับงานวิจัย [16] กล่าวคือใช้หลักการของตารางแฮชแบบกระจาย (Distributed Hash Table: DHT) ในการสร้างดัชนีแบบกระจาย (Distributed Indexing) มีการทำสำเนาของข้อมูล มีการจัดข้อมูลในรูปลำดับชั้นแบบระบบแฟ้มทั่วไป อย่างไรก็ตาม ผู้วิจัยได้พัฒนาเพิ่มขึ้นจากงานวิจัยเดิม เพื่อให้เป็นระบบแฟ้มข้อมูลที่สมบูรณ์ขึ้น เช่น การค้นหาแฟ้มที่ยืดหยุ่นขึ้น, การดาวน์โหลดแฟ้มแบบขนานเพื่อเพิ่มความเร็วในการถ่ายโอนแฟ้ม, การกำหนดรุ่น (version), การจัดการสำเนาของแฟ้ม เป็นต้น

4.2.1 **ตารางแฮชแบบกระจาย และข้อความจีไอเอสพี** (DHT and GISP messages)

จามจุรีเอ็กซ์พลอเรอร์ใช้ตารางแฮชแบบกระจายในการจัดการโครงสร้างของแฟ้มข้อมูล ซึ่งมีข้อดีคือสามารถกระจายภาระของการทำดัชนีรายการแฟ้มข้อมูลทั้งระบบได้ดี และเนื่องจากโปรแกรมส่วนที่เป็นเพียร์-ทู-เพียร์ในงานวิจัยนี้พัฒนาบนจักซ์ตาทั้งหมด ผู้วิจัยจึงใช้บริการตารางแฮชแบบกระจายบนจักซ์ตา ซึ่งบริการหนึ่งที่สามารถใช้ได้ก็คือจีไอเอสพี (GISP) [29]

จีไอเอสพีมีโพรโทคอลของตัวเองในการจัดสรรหน้าที่ของเพียร์ต่างๆ บนจักซ์ตาให้มีการกระจายภาระอย่างเท่าเทียมกันตามหลักการที่กล่าวไปแล้วในหัวข้อ 2.2.4 และส่วนสำคัญก็คือ ข้อความจีไอเอสพี ซึ่งเป็นข้อความที่เก็บคู่ข้อมูล (key-value pair) ซึ่งในจามจุรีเอ็กซ์พลอเรอร์มีการเก็บข้อความคู่ข้อมูลแบ่งได้เป็น 2 ประเภท ดังนี้

1) ข้อความโครงสร้าง (Structure information)

ข้อความโครงสร้างใช้สำหรับเป็นข้อมูลในการจัดการโครงสร้างแฟ้มข้อมูลให้เป็นลำดับชั้น จะประกอบด้วยข้อมูลของชื่อแฟ้มหรือไดเร็กทอรี และใช้ชื่อไดเร็กทอรีแม่ (parent directory) เป็นคำหลัก ตัวอย่างของข้อความประเภทนี้ แสดงในตารางที่ 4-1

ตารางที่ 4-1 ข้อความโครงสร้างของจามจุรีเอ็กซ์พลอเรอร์

Key	Value	คำอธิบาย
/	tag=Dir DirName=dir1/ Path=/	ไดเร็กทอรีชื่อ “dir1” ซึ่งมีไดเร็กทอรีแม่ชื่อ “/”
/dir1/	tag=Dir DirName=subdir1/ Path=/dir1/	ไดเร็กทอรีชื่อ “subdir1” ซึ่งมีไดเร็กทอรีแม่ชื่อ “/dir1/”
/path/to/	tag=File FileName=file1.txt Path=/path/to/	แฟ้มชื่อ “file1.txt” ซึ่งมีไดเร็กทอรีแม่ชื่อ “/path/to/”

2) คำอธิบายข้อมูลแฟ้ม (File metadata)

คำอธิบายข้อมูลแฟ้มเป็นข้อความที่อธิบายลักษณะของแฟ้ม ประกอบด้วยข้อมูลชื่อแฟ้ม, เวอร์ชั่น, เจ้าของ และ คอนเทนต์แอดเวอทิสเมนต์ (Content Advertisement) ซึ่งเป็นส่วนที่ใช้ในการดาวน์โหลดแฟ้มด้วยบริการซีเอ็มเอส (CMS) [36] ของจักซ์ตา และใช้คำหลักเป็นเส้นทางสัมบูรณ์ของแฟ้ม ตัวอย่างของข้อความคำอธิบายแฟ้มข้อมูล แสดงในError! Not a valid bookmark self-reference.

ตารางที่ 4-2 ตัวอย่างคำอธิบายข้อมูลแฟ้ม

Key	Value	คำอธิบาย
/path/to/ file.txt	tag=FilePath FileName=file.txt Path=/path/to/ Owner=PeerID1 TimeStamp=1198051110046 ContentAdv=<?xml version="1.0" encoding="UTF-8"?> <!DOCTYPE jxta:ContentAdvertisement> <jxta:ContentAdvertisement> <name> file.txt </name> <cid> md5:d41d8cd98f00b…98</cid> <length> 512 </length> <address> jxta://uuid-596… </address> </jxta:ContentAdvertisement>	คำอธิบายข้อมูลแฟ้มชื่อ “file.txt”

4.2.2 **การทำซ้ำ** (Replication)

คู่ข้อมูลที่อธิบายในหัวข้อ 4.2.1 จะถูกกระจายไปในเครือข่ายอย่างสม่ำเสมอด้วยกลไกของจีไอเอสพี และแต่ละคู่ข้อมูลจะถูกกระจายไป 5 สำเนา (ค่าโดยปริยาย) เพื่อเพิ่มโอกาสในการค้นเจอข้อมูลเหล่านั้นในสภาพแวดล้อมของเพียร์-ทู-เพียร์ จามจุรีเอ็กซ์พลอเรอร์ใช้ประโยชน์จากกลไกนี้โดยการให้เพียร์ที่รับผิดชอบคู่ข้อมูลของคำอธิบายข้อมูลแฟ้ม (file metadata) ทำการดาวน์โหลดแฟ้มนั้นมาเก็บไว้ที่ตนเอง และในที่สุดจะทำให้แฟ้มมีสำเนาอยู่อย่างน้อย 5 สำเนา

ขั้นตอนในการสร้างสำเนาของข้อมูลเป็นดังนี้

1) เพียร์ P_1 วนค้นหาแฟ้มและไดเร็กทอรีในโฮมไดเร็กทอรีของตนและทำการแบ่งปัน (share) โดยการสร้างคู่ข้อมูลของแฟ้มและไดเร็กทอรีเหล่านั้น ซึ่งจะได้คู่ข้อมูลของโครงสร้าง และคำอธิบายข้อมูลแฟ้ม
2) เพียร์ P_1 แจกจ่ายคู่ข้อมูลเหล่านั้นไปยังเพียร์ต่างๆตามกลไกของตารางแฮชแบบกระจาย
3) เพียร์ P_n ที่ได้รับคู่ข้อมูลของคำอธิบายข้อมูลแฟ้มใช้คอนเทนต์แอดเวอทิสเมนต์ที่แนบไปด้วยนั้นในการดาวน์โหลดแฟ้มมาเก็บไว้ที่ตนเองในไดเร็กทอรีแคช (cache directory)
4) เพียร์ P_n ที่รับผิดชอบแฟ้มเหล่านั้นทำการแบ่งปันแฟ้มนั้นต่อ โดยการสร้างคู่ข้อมูลและแจกจ่ายไปยังเพียร์ต่างๆเช่นเดียวกับข้อ 1) และ 2)

ขั้นตอนเหล่านี้จะมีการทำงานอยู่เป็นระยะๆ เพื่อให้แฟ้มข้อมูลใหม่ที่สร้างสามารถกระจายในเครือข่ายได้ทั้งหมด

ในขั้นตอนที่ 4 คู่ข้อความที่เพียร์ P_n สร้างขึ้นมาใหม่จะมีคำหลัก และเนื้อความเหมือนกับคู่ข้อความจากเพียร์ P_1 ในขั้นตอนที่ 1 ยกเว้นคอนเทนต์แอดเวอทิสเมนต์ ซึ่งจะแตกต่างกันตามที่อยู่ของสำเนาแฟ้มข้อมูล อย่างไรก็ตาม คู่ข้อมูลจากขั้นตอนที่ 1 และ 4 ที่เกี่ยวข้องกันนั้นในที่สุดจะถูกกระจายไปยัง 5 เพียร์เท่านั้น เนื่องจากใช้คำหลักเดียวกัน

4.2.3 **ไดเร็กทอรีแบ่งปัน** (Shared directory)

ไดเร็กทอรีแบ่งปัน หมายถึงไดเร็กทอรีในเครื่องของเพียร์นั้นๆ ที่จะมีการแบ่งปันไปยังเพียร์ต่างๆ หากแฟ้มข้อมูลใดถูกวางไว้ในไดเร็กทอรีเหล่านี้ จะถูกแบ่งปันโดยอัตโนมัติ โดยไดเร็กทอรีแบ่งปันนี้สามารถกำหนดได้ในแฟ้มโครงแบบ (configuration file)

4.2.4 **การแคช** (Caching)

ตลอดระยะเวลาการทำงานของจามจุรีเอ็กซ์พลอเรอร์ แต่ละเพียร์จะมีการดาวน์โหลดแฟ้มข้อมูลมาเก็บเอาไว้ที่ตนเองเรื่อยๆ ซึ่งจะเก็บไว้ที่ไดเร็กทอรีแคช (cache directory) ในบางครั้งเพียร์อาจต้องการค้นหาแฟ้มใดๆในเครือข่าย ซึ่งอาจจะเคยดาวน์โหลดมาแล้วและยังอยู่ในแคช ดังนั้นสามารถนำแฟ้มนั้นในแคชมาใช้ได้เลย อย่างไรก็ตามจะต้องมีการตรวจสอบรุ่นของแฟ้มก่อนนำมาใช้ด้วย การตรวจสอบรุ่นของแฟ้มอธิบายไว้ในหัวข้อที่ 4.2.5

4.2.5 **การควบคุมรุ่น** (Version control)

ระบบแฟ้มในจามจุรีเอ็กซ์พลอเรอร์ใช้หลักการของตารางแฮชแบบกระจาย, การโฆษณา และการสร้างสำเนาเป็นหลัก หลักการเหล่านี้ล้วนแต่เป็นการกระจายข้อมูลไปอยู่ที่เพียร์ต่างๆ ซึ่งหากข้อมูลชุดใดชุดหนึ่งมีการเปลี่ยนแปลงไป ก็จะทำให้ขาดความต้องกัน (consistency) ในทันที วิธีที่จะทำให้ข้อมูลมีความต้องกันในท้ายที่สุดนั้นก็สามารถทำได้แต่อาจใช้เวลานาน และไม่คุ้มค่า จามจุรีเอ็กซ์พลอเรอร์มีวิธีในการควบคุมความต้องกันอย่างง่ายโดยการใช้ตราเวลา (timestamp) แนบไปกับข้อความคำอธิบายข้อมูลแฟ้ม (file metadata) แต่ละชุดด้วย ข้อมูลตราเวลานี้ได้มาจากเวลาดัดแปรล่าสุด (last modified time) ของแฟ้ม โดยที่เมื่อเจ้าของแฟ้มทำการเปลี่ยนแปลงแฟ้ม ก็จะมีการสร้างคู่ข้อมูลคำอธิบายข้อมูลแฟ้มใหม่ และกระจายไปยังเครือข่ายโดยใช้กลไกเดียวกับการแบ่งปันแฟ้มในหัวข้อ 4.2.1 และ 0

4.2.6 **การค้นหาแฟ้ม** (Searching)

การค้นหาแฟ้มข้อมูลทำได้โดยการค้นด้วยคำหลัก และใช้คำหลักนี้ในการค้นด้วยกลไกของจีไอเอสพี หากคำหลักเป็นชื่อไดเร็กทอรี จะได้ผลลัพธ์ออกมาเป็นรายการชื่อไดเร็กทอรีย่อย และแฟ้มข้อมูล ที่อยู่ในไดเร็กทอรีนั้น พร้อมด้วยรายละเอียดของแต่ละไดเร็กทอรีและแฟ้ม และหากคำหลักเป็นชื่อแฟ้มจะได้ผลลัพธ์เป็นรายละเอียดแฟ้มนั้น

ในการค้นหามีตัวแปรที่สำคัญคือ เวลารอ (timeout) เป็นการกำหนดเวลาที่ใช้รอผลการค้นหาที่จีไอเอสพีค้นในเครือข่าย หากหมดช่วงเวลานี้จะไม่รับผลอีกต่อไป ค่าโดยปริยายที่กำหนดคือ 2 วินาที

4.2.7 **การดาวน์โหลดแฟ้ม** (Download)

การดาวน์โหลดแฟ้มคือการถ่ายโอนแฟ้มจากเพียร์อื่น หรือจากแคชของตนเองมายังเส้นทาง (path) ที่กำหนด ในการดาวน์โหลดแฟ้มจะต้องกำหนดเส้นทางอ้างอิงของแฟ้มในเครือข่าย ซึ่งจะอยู่ในรูปโครงสร้างแฟ้มแบบลำดับชั้น เช่น /path/to/file.txt และจะใช้ค่านี้เป็นคำหลักไปค้นหาและดาวน์โหลดแฟ้ม ดังนั้นกระบวนการของการดาวน์โหลดแฟ้มจึงประกอบด้วย 2 ขั้นตอนใหญ่ๆคือ

4.2.7.1 **การค้นหา**

การค้นหาแฟ้มจะได้รายละเอียดของแฟ้ม เช่น ชื่อแฟ้ม, ชื่อเจ้าของแฟ้ม, รุ่นของแฟ้ม, และคอนเทนต์แอดเวอทิสเมนต์ เนื่องจากโฆษณาของแฟ้มหนึ่งๆอาจจะมีหลายรุ่นได้ในเวลาเดียวกัน ผู้ใช้อาจจะระบุรุ่นในการค้นหาด้วยเพื่อกรองเฉพาะรุ่นที่ต้องการเท่านั้น หากไม่ระบุรุ่น จะได้ผลลัพธ์เป็นแฟ้มรุ่นที่ใหม่ที่สุดที่หาเจอในเวลานั้น การค้นหาแฟ้มแสดงได้ดังรูปที่ 4-4

ในการค้นหาแฟ้มเพื่อดาวน์โหลดนั้น จะมีตัวแปรที่สำคัญ 3 ตัวแปรคือ

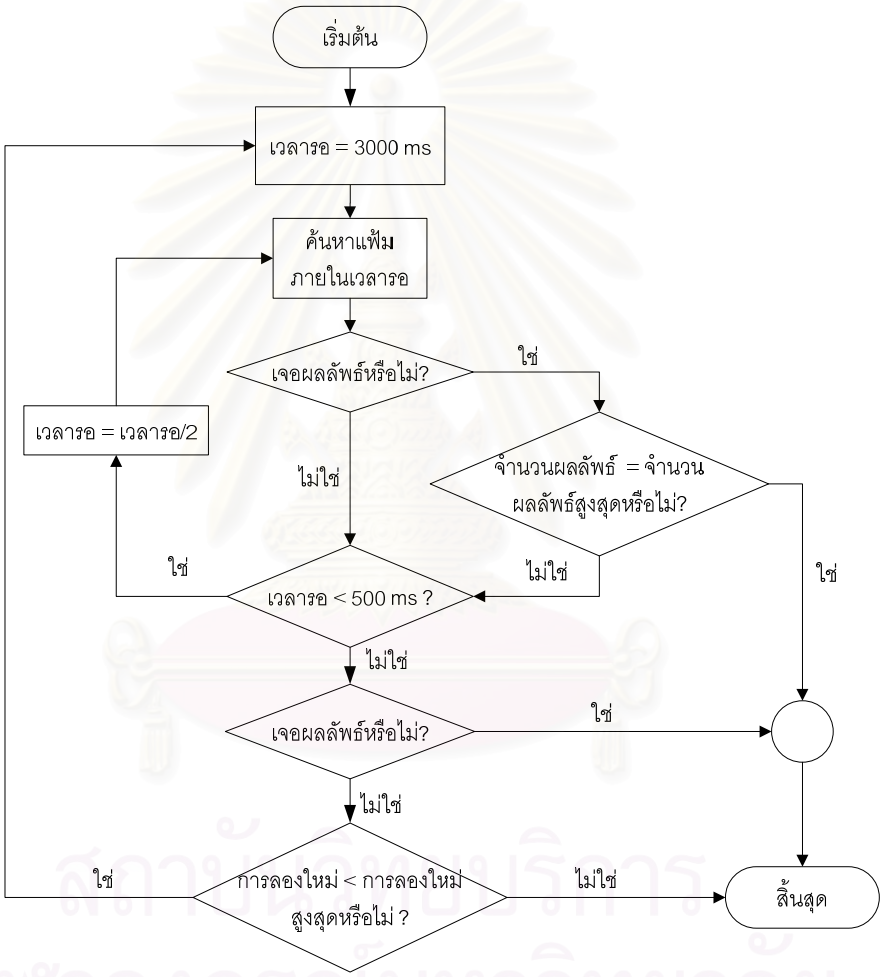
ก) เวลารอ (timeout) คือกำหนดเวลาสูงสุดที่จะรอผลลัพธ์จากการค้นหา หากเวลารอครั้งแรกหมดลง จะมีการเพิ่มเวลารอในแบบถอยหลังแบบชี้กำลัง (exponential backoff) โดยจะลดลงทีละครึ่งหนึ่ง จนถึงระยะเวลาหนึ่ง จะถือว่าไม่สามารถค้นเจอแฟ้มได้ เช่น หากกำหนดเวลารอไว้ที่ 3000 มิลลิวินาที จะมีเวลารอลดลงในแต่ละขั้น คือ 3000, 1500, 750, 375 มิลลิวินาที ตามลำดับ เงื่อนไขของการหมดเวลารอในแต่ละขั้นมี 2 กรณีคือ

- ค้นเจอแฟ้มที่ต้องการก่อนหมดเวลา
- ค้นไม่เจอเมื่อหมดเวลา

ทั้งสองกรณีนี้จะทำให้มีการกำหนดเวลารอใหม่ด้วยวิธีที่กล่าวไปแล้ว ด้วยวิธีนี้จะทำให้มีการกำหนดเวลารอที่ควบคุมได้ และหากพบข้อมูลแฟ้มเร็วขึ้นก็จะมีผลให้ใช้เวลารอน้อยลงด้วยเช่นกัน

ข) จำนวนผลลัพธ์สูงสุด (maximum result) คือจำนวนผลลัพธ์ที่เจอสูงสุด เนื่องจากในการค้นหาแต่ละครั้งอาจได้ผลลัพธ์การค้นหาจากแฟ้มหลายๆสำเนา ซึ่งผลลัพธ์เหล่านี้จะนำไปใช้ในการดาวน์โหลดแบบขนาน หากค้นหาเจอผลลัพธ์ถึงจำนวนผลลัพธ์สูงสุดแล้วก็จะหยุดกระบวนการค้นหา และเข้าสู่กระบวนการดาวน์โหลดทันที

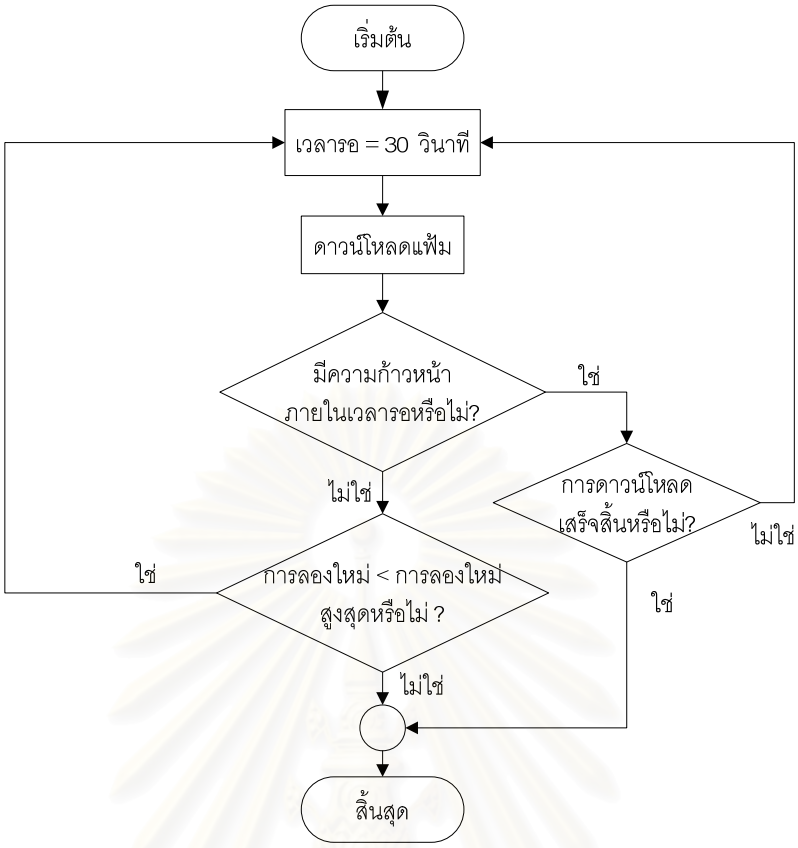
ค) การลองใหม่ (retry) เป็นการกำหนดจำนวนครั้งของการค้นหาใหม่ เนื่องจากในสภาวะที่เครือข่ายมีความเร็วต่ำ ข้อมูลอาจจะมาช้าเกินไป หรือหากกำหนดเวลารอน้อยเกินไป ข้อมูลอาจจะยังมาไม่ถึง ดังนั้นจึงได้กำหนดการลองใหม่ โดยการบวนการนี้จะเกิดขึ้นหากไม่เจอผลลัพธ์จากการค้นหาแฟ้มในรอบก่อนหน้านี้เท่านั้น และหากครบจำนวนการลองใหม่แล้วยังไม่เจอแฟ้มก็จะถือว่าไม่มีแฟ้มนั้นอยู่ในเครือข่าย



รูปที่ 4-4 ผังงานการค้นหาแฟ้มสำหรับการดาวน์โหลด

4.2.7.2 การดาวน์โหลด

หลังจากค้นหาแฟ้ม จะได้ข้อมูลต่างๆเกี่ยวกับแฟ้ม รวมถึงคอนเทนต์แอดเวอทิสเมนต์ ซึ่งคอนเทนต์แอดเวอทิสเมนต์นี้จะใช้ในการดาวน์โหลด หากในการค้นหาแฟ้มนั้นได้รับผลการค้นหามากกว่า 1 ผลลัพธ์จะทำให้ขั้นตอนการดาวน์โหลดแฟ้มทำแบบขนานได้ การดาวน์โหลดแสดงได้ดังรูปที่ 4-5



รูปที่ 4-5 ผังงานการดาวน์โหลดแฟ้ม

ในการดาวน์โหลดแฟ้มมีตัวแปรที่สำคัญ 2 ตัวแปรคือ

ก) เวลารอ (timeout) คือเวลาในการรอผลความเคลื่อนไหวจากการดาวน์โหลดแฟ้ม บริการซีเอ็มเอสจะแสดงความเคลื่อนไหวของการดาวน์โหลดแฟ้มออกมาเรื่อยๆ เป็นเปอร์เซ็นต์ที่ดาวน์โหลดเสร็จไปแล้ว หากความเคลื่อนไหวสิ้นสุดไปภายในเวลารอที่กำหนด จะถือว่าการดาวน์โหลดผิดพลาด (ค่าโดยปริยายคือ 30 วินาที) หรือหากซีเอ็มเอสฟ้องว่าการดาวน์โหลดแฟ้มผิดพลาดเองนั้นก็ให้ถือว่าเป็นการดาวน์โหลดผิดพลาดทันทีโดยไม่ต้องรอ

ข) การลองใหม่ (retry) คือการให้ซีเอ็มเอสดาวน์โหลดแฟ้มใหม่หลังจากที่ผิดพลาดจากการดาวน์โหลดครั้งก่อนหน้านี้ หากครบกำหนดการลองใหม่แล้ว จะถือว่าการดาวน์โหลดผิดพลาด และสิ้นสุดกระบวนการดาวน์โหลดแฟ้ม

4.3 **ตัวจัดการงานเพียร์-ทู-เพียร์** (P2P Job Manager)

ตัวจัดการงานเพียร์-ทู-เพียร์ เป็นส่วนหนึ่งของการเชื่อมต่อระหว่างกริดกับเพียร์-ทู-เพียร์ โดยมีหน้าที่ในการแปลงงานในระบบกริดให้เป็นงานที่รันได้บนเพียร์-ทู-เพียร์ ในที่นี้มิดเดิลแวร์บนระบบกริดคือโกลบัสทูลคิท และมิดเดิลแวร์บนเพียร์-ทู-เพียร์ก็คือฮาร์เวส ดังนั้น ตัวจัดการงานจะต้องสามารถแปลงงานที่อยู่ในรูปแบบของโกลบัสให้เป็นงานในรูปแบบของฮาร์เวสนั่นเอง

ในงานวิจัยนี้ ตัวจัดลำดับงานคือฮาร์เวส และผู้วิจัยได้พัฒนาตัวจัดการงานเพียร์-ทู-เพียร์ขึ้น ให้มีหน้าที่แปลงงานบนโกลบัสให้เป็นงานบนฮาร์เวส ตัวจัดลำดับงานนี้ได้พัฒนาขึ้นตามส่วนต่อประสานที่แกรมกำหนดไว้ โดยให้ชื่อว่า ตัวจัดลำดับงานเพียร์-ทู-เพียร์ (jobmanager-p2p) มีรายละเอียดดังนี้

4.3.1 **อาร์เอสแอล** (RSL)

อาร์เอสแอล (RSL : Resource Specification Language) เป็นภาษาที่ใช้สำหรับบรรยายลักษณะงานของโกลบัส โดยจะบรรยายในรูปคู่ลักษณะประจำ (<attribute,value> pair) เป็นลักษณะเดียวกันทั้งหมด และมีหลายๆคู่ลักษณะประจำประกอบกันเพื่อบรรยายลักษณะงาน รวมถึงทรัพยากรที่ต้องการใช้สำหรับงานนั้นๆ

```
(* this is a comment *)
& (executable = a.out (* <-- that is an unquoted literal *))
  (directory  = /home/nobody )
  (arguments  = arg1 "arg 2")
  (count = 1)
```

รูปที่ 4-6 ตัวอย่างแฟ้มอาร์เอสแอล

ในการสั่งงานบนกริด จะมีการสร้างอาร์เอสแอล แล้วส่งเป็นข้อความไปยังเครื่องปลายทาง และแกรมที่เครื่องปลายทางก็จะแจงออกมาเป็นลักษณะประจำแต่ละตัวที่กำหนด ถึงจุดนี้ ตัวจัดการงานก็สามารถนำลักษณะประจำเหล่านี้ไปใช้จัดการกับทรัพยากรต่อไป

สำหรับตัวจัดการงานเพียร์-ทู-เพียร์นั้น มีการจัดการกับอาร์เอสแอล คือ แปลงอาร์เอสแอลให้เป็นรูปโฆษณางานในรูปแบบฮาร์เวส ดังตารางที่ 4-3

ตารางที่ 4-3 ความสัมพันธ์ระหว่างโฆษณางานและอาร์เอสแอล

Job Advertisement	RSL
Command	executable
	arguments
Input	file_stage_in
Output	file_stage_out
	stdout
	stderr
jobName	N/A*
Owner	N/A*
execMode	exec_mode**

ลักษณะประจำอื่นๆในอาร์เอสแอลยังไม่สนับสนุนในฮาร์เวสรุ่นปัจจุบัน จึงไม่ได้แสดงไว้ เนื่องจากมีข้อแตกต่างระหว่างลักษณะประจำในอาร์เอสแอลกับโฆษณางานอีกมาก ซึ่งไม่อาจจับคู่กันได้โดยตรง และลักษณะประจำบางอย่างในอาร์เอสแอลไม่สามารถแปลงให้เข้ากับกลไกของฮาร์เวสในรุ่นปัจจุบันได้ อย่างไรก็ถาม หากในการสั่งงานมีการระบุลักษณะประจำอื่นๆเข้ามาในอาร์เอสแอล ก็ยังสามารถสั่งงานได้ แต่ลักษณะประจำใดที่ตัวจัดการงานนี้ไม่สนับสนุน ก็จะไม่มีผลต่อการรันโปรแกรม

4.3.2 การเขียนตัวจัดการงานเพียร์-ทู-เพียร์

ตัวจัดการงานเพียร์-ทู-เพียร์ถูกพัฒนาขึ้นตามส่วนเชื่อมต่อของแกรม โดยเขียนเป็นบทคำสั่งเพิร์ล (perl script) และมีฟังก์ชันที่ต้องพัฒนาดังนี้

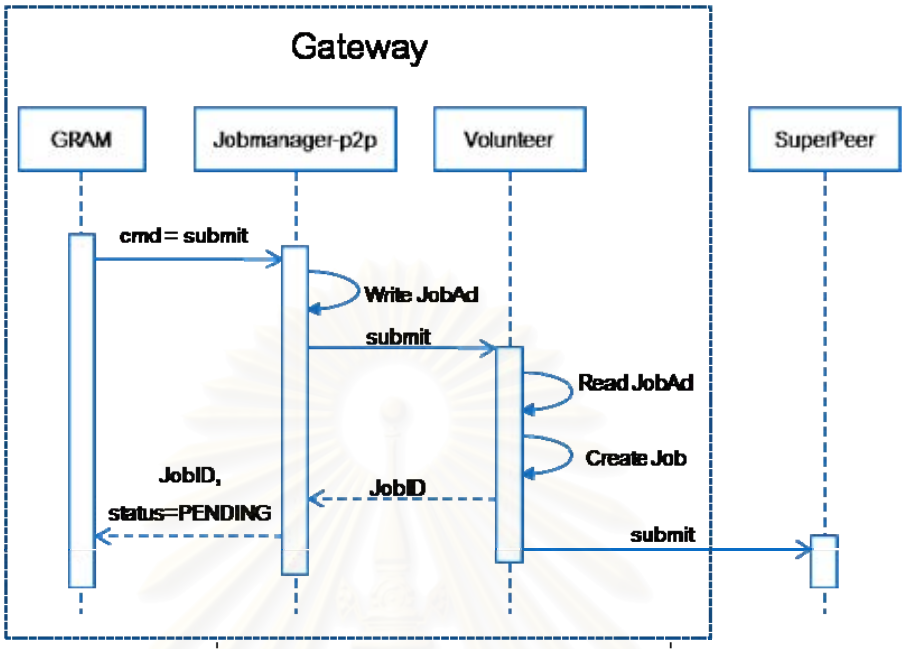
1) submit() เป็นฟังก์ชันที่จะถูกเรียกเมื่อมีการรับงานเข้ามา โดยในฟังก์ชันนี้จะเตรียมลักษณะประจำต่างๆที่ผ่านการแจงจากอาร์เอสแอลมาให้ ในตัวจัดการงานเพียร์-ทู-เพียร์จะเป็นการจัดรูปแบบของลักษณะประจำเหล่านี้ใหม่ให้อยู่ในรูปแบบโฆษณางานของฮาร์เวส แล้วจึงเขียนไว้ลงในแฟ้ม จากนั้นจะไปเรียกโปรแกรมสั่งงานของฮาร์เวสให้อ่านแฟ้มนั้นต่อไป (ในที่นี้ผู้สั่งงานคือโวลันเทียร์ที่เครื่องเกตเวย์) และจะได้รับเลขประจำงาน (Job ID) จากโปรแกรมฮาร์เวส

---

* jobName และ owner ไม่สามารถจับคู่กับลักษณะประจำใดๆในอาร์เอสแอลได้

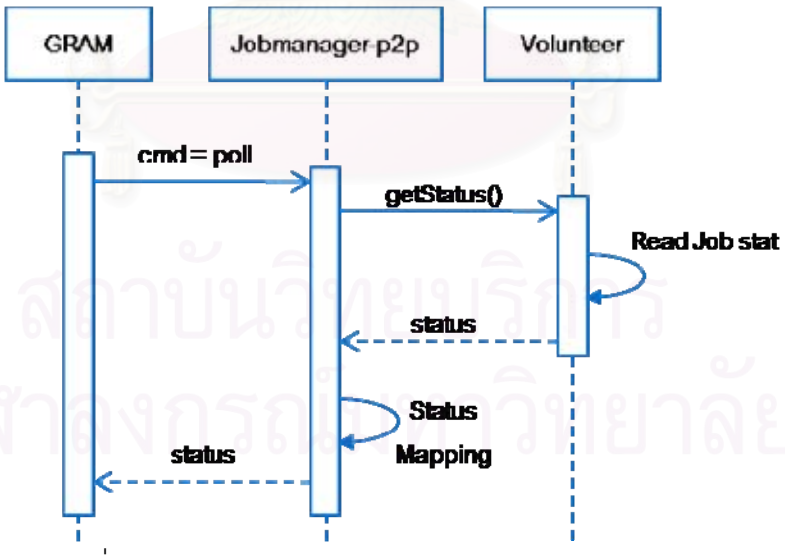
** exec_mode ไม่ใช่ลักษณะประจำมาตรฐานของอาร์เอสแอล ผู้วิจัยได้กำหนดขึ้นและอธิบายไว้ในหัวข้อ 4.3.3

แล้วจึงส่งค่าเลขประจำงาน และสถานะเป็น PENDING กลับคืนให้แก่แกรม เพื่อใช้ในกระบวนการของโกลบัสต่อไป



รูปที่ 4-7 แผนภาพลำดับแสดงกระบวนการสั่งงาน

2) poll() เป็นการสำรวจสถานะของงาน สำหรับฮาร์เวสสถานะของงานจะถูกเก็บเป็นแฟ้มไว้อยู่ที่เครื่องที่สั่งงาน ในที่นี้คือเครื่องเกตเวย์ หลังจากฟังก์ชันนี้ถูกเรียกก็จะไปถามสถานะของการสั่งงาน แล้วส่งสถานะกลับไปยังแกรม



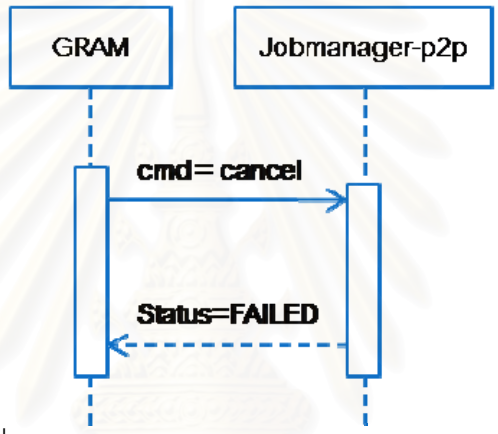
รูปที่ 4-8 แผนภาพลำดับแสดงกระบวนการสำรวจสถานะงาน

สถานะของงานในฮาร์เวสนั้นต่างกับสถานะของแกรม จึงต้องมีการแปลงสถานะดังตารางที่ 4-4

ตารางที่ 4-4 การแปลงสถานะงานจากฮาร์เวสไปสู่แกรม

Harvest	GRAM
Waiting	PENDING
Running	ACTIVE
Staging Out	ACTIVE
Completed	DONE

3) cancel() จะถูกเรียกเมื่อมีการสั่งยกเลิกงานจากฝั่งกริด ในฮาร์เวสนั้นไม่มีคำสั่งสำหรับการยกเลิกงาน ดังนั้นในฟังก์ชันนี้จึงไม่มีการกระทำใดๆ นอกจากการคืนค่าสถานะเป็น FAILED



รูปที่ 4-9 แผนผังลำดับแสดงกระบวนการยกเลิกงาน

4.3.3 แบบวิธีกระทำการ (Execution mode)

ดังที่กล่าวไปแล้วในหัวข้อ 4.1.4 งานที่รันบนฮาร์เวสนั้นเป็นได้หลายประเภทขึ้นอยู่กับระบบปฏิบัติการต่างๆที่โวลันเทียร์รันอยู่ ผู้วิจัยจึงออกแบบให้ผู้ใช้สามารถสั่งงานบนกริดได้หลากหลายขึ้นเช่นกัน โดยการเพิ่มลักษณะประจำอีกลักษณะหนึ่ง คือ แบบวิธีกระทำการ ในการสั่งงานผ่านกริดนั้นจะต้องระบุลักษณะประจำนี้ในอาร์เอสแอลด้วย เช่น (execmode = dos_batch)

อย่างไรก็ตาม ลักษณะประจำที่เพิ่มขึ้นมานี้ไม่ได้เป็นลักษณะประจำมาตรฐานของโกลบัส จึงต้องทำให้แกรมที่ฝั่งเกตเวย์สามารถแจงลักษณะประจำนี้ได้ด้วย ซึ่งจะต้องมีตัวตรวจสอบอาร์เอสแอลเพิ่มเติมขึ้นมาพิเศษ โดยสร้างแฟ้ม p2p.rvf และบรรยายลักษณะประจำนั้น เมื่อมีการเรียกตัวจัดการงานเพียร์-ทู-เพียร์ แกรมก็จะตรวจสอบลักษณะประจำของอาร์เอสแอลจากแฟ้มนี้เพิ่มเติมด้วย

4.3.4 **การถ่ายแฟ้มเข้า/ออก** (file_stage_in/file_stage_out)

ความสามารถอีกอย่างหนึ่งของแกรมคือสามารถถ่ายแฟ้มเข้าและออกได้ โดยใช้ลักษณะประจำ file_stage_in และ file_stage_out ตามลำดับ เพื่อเป็นการถ่ายโอนแฟ้มระหว่างโหนดในกริด ข้อมูลในลักษณะประจำทั้งสองนี้จะถูกแปลงเป็นลักษณะประจำในโฆษณางานของฮาร์เวสด้วย โดยจะแปลงเป็น input และ output ตามลำดับ

การถ่ายโอนแฟ้มแบบนี้เป็นการถ่ายโอนโดยใช้แกรมทั้งสิ้น ซึ่งจะต่างกับการถ่ายโอนด้วยโพรโทคอลกริดเอฟทีพีที่จะกล่าวต่อไปในหัวข้อ 4.4

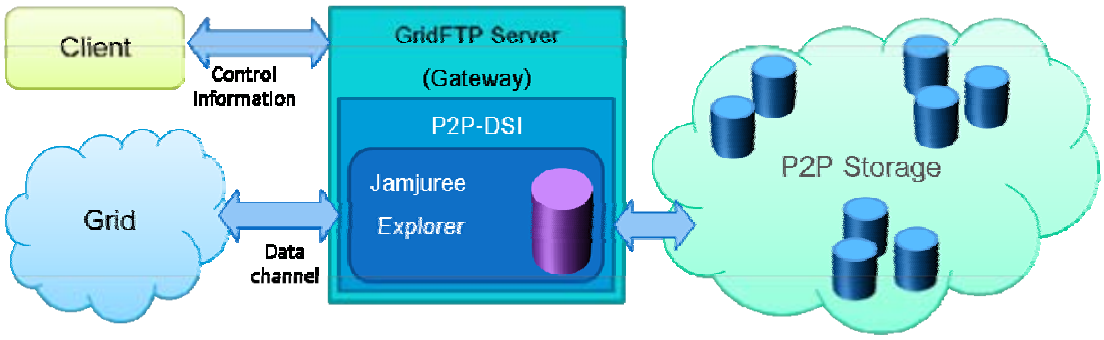
4.3.5 **ช่องทางข้อความออก, ช่องทางข้อความผิดพลาด** (Standard output, standard error)

ช่องทางข้อความออกและช่องทางข้อความผิดพลาด ถูกกำหนดเป็นลักษณะประจำอย่างหนึ่งในอาร์เอสแอล เขียนในรูป “stdout” และ “stderr” โดยเป็นการบอกว่าข้อความออกและข้อความผิดพลาดที่ได้จากการรันโปรแกรมนั้นจะเก็บไว้ที่ใด หรือเก็บไว้เป็นแฟ้มใด ซึ่งหากไม่กำหนดก็จะนำไปแสดงออกทางหน้าจอของผู้สั่งงานบนกริดเมื่องานเสร็จสิ้น

ในฮาร์เวส ข้อมูลช่องทางข้อความออกและช่องทางข้อความผิดพลาดจะถูกเก็บไว้เป็นแฟ้ม 2 แฟ้ม และจะถูกรวมเข้าเป็นลักษณะประจำ “output” ของฮาร์เวสด้วย เนื่องจากถือเป็นแฟ้มส่งออกอย่างหนึ่งด้วย และเมื่อแฟ้มเหล่านี้ถูกส่งไปยังเกตเวย์แล้ว ตัวจัดการงานเพียร์-ทู-เพียร์จะนำไปเก็บไว้ในแคชไดเร็กทอรี (cache directory) ของแกรม และปล่อยหน้าที่ในการจัดการให้แกรมต่อไป

4.4 **เพียร์-ทู-เพียร์ดีเอสไอ** (P2P-DSI)

ดีเอสไอ (DSI : Data Storage Interface) หรือส่วนต่อประสานหน่วยเก็บข้อมูลเป็นส่วนประกอบหนึ่งของกริดเอฟทีพีโพรโทคอล (GridFTP protocol) ที่ทำหน้าที่ในการจัดการหน่วยเก็บข้อมูลโดยเฉพาะ โดยจะไม่สนใจโพรโทคอลในการสื่อสารระหว่างโหนด สิ่งที่ดีเอสไอสนใจคือการอ่านและเขียนข้อมูลลงในหน่วยเก็บข้อมูลเท่านั้น ดีเอสไอที่มีส่วนต่อประสานสำหรับพัฒนาเองได้ และสามารถนำไปเชื่อมต่อกับส่วนควบคุมโพรโทคอลกริดเอฟทีพี เพื่อให้ทำงานเข้ากันกับกริดเอฟทีพีโหนดอื่นๆได้ ในงานวิจัยนี้ดีเอสไอเป็นอีกส่วนหนึ่งของการเชื่อมต่อระหว่างกริดกับเพียร์-ทู-เพียร์ ซึ่งผู้วิจัยได้พัฒนาดีเอสไอขึ้นเฉพาะสำหรับติดต่อกับจามจุรีเอ็กซ์พลอเรอร์ แสดงดังรูปที่ 4-10



รูปที่ 4-10 การเชื่อมต่อของเพียร์-ทู-เพียร์ดีเอสไอ

จากรูป client เป็นผู้สั่งคำสั่งด้วยโพรโทคอลกริดเอฟทีพี ซึ่งการจะมีการถ่ายโอนข้อมูลจาก/สู่ตัว client เอง หรือ client ยังสามารถสั่งงานแบบบุคคลที่สาม (3rd party) ได้ โดยที่ client จะเป็นเพียงผู้เริ่มสั่งคำสั่ง แต่การถ่ายโอนแฟ้มจริงจะเกิดขึ้นระหว่างกริดโหนดตัวอื่น

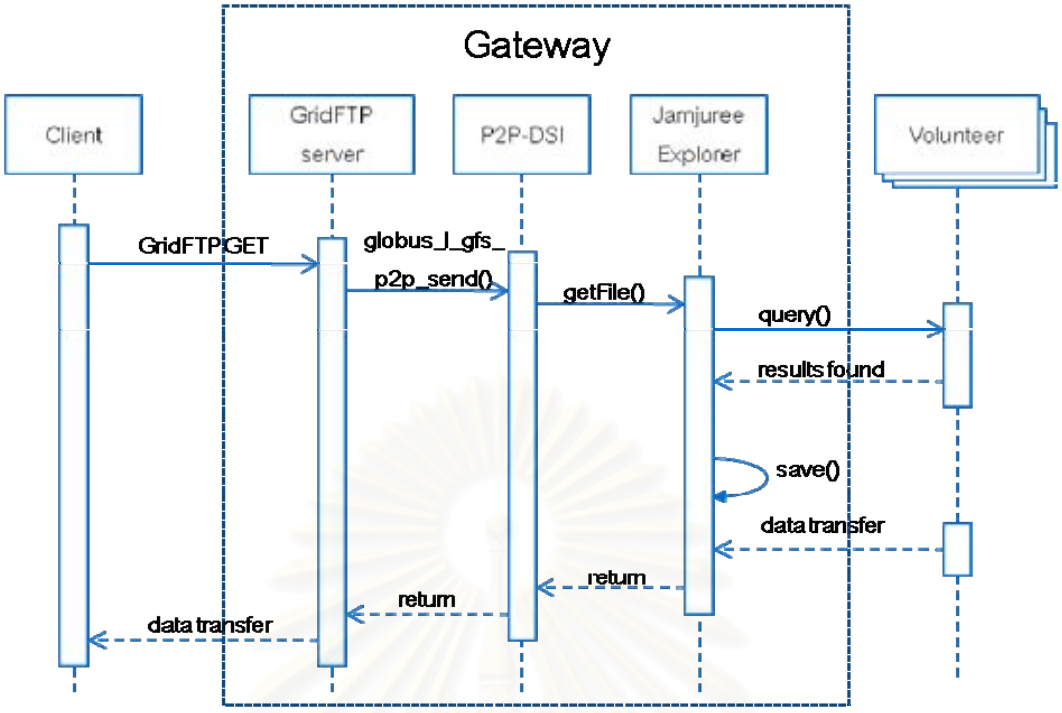
การพัฒนาเพียร์-ทู-เพียร์ดีเอสไอใช้ภาษาซี และต้องพัฒนาตามส่วนต่อประสาน มีรายละเอียดดังนี้

4.4.1 **คำสั่ง** GET

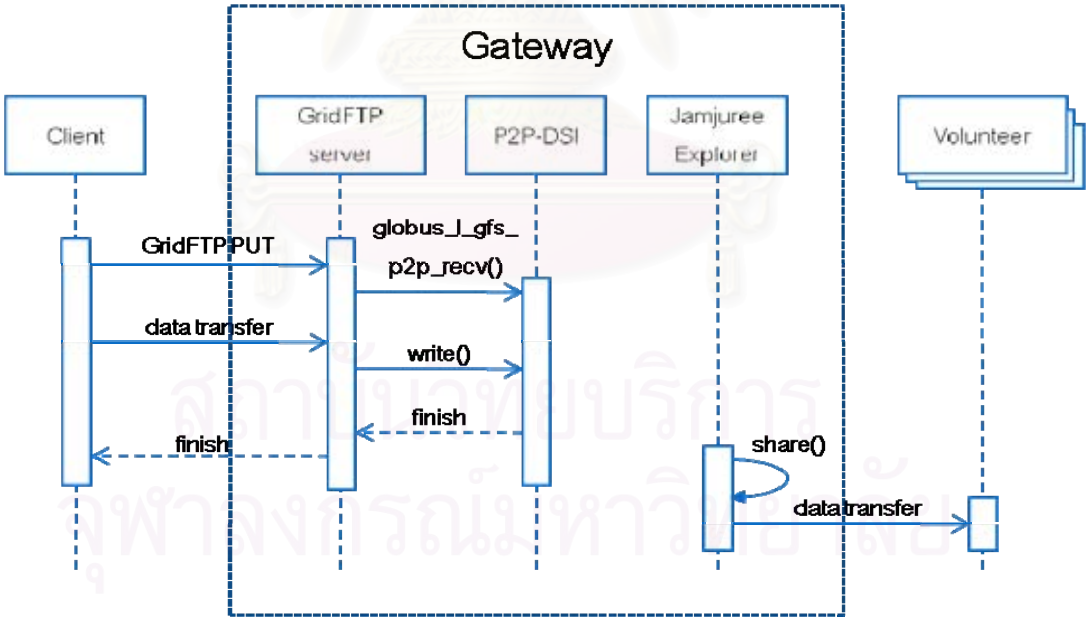
คำสั่ง GET จะถูกสั่งจากเครื่องลูกข่ายเพื่อต้องการเรียกแฟ้มจากเครื่องแม่ข่าย (เครื่องเกตเวย์) และให้ส่งไปยังปลายทางที่ใดที่หนึ่ง ในดีเอสไอ จะถูกเรียกผ่านฟังก์ชัน `globus_l_gfs_p2p_send()` ในฟังก์ชันนี้ จะเป็นการดาวน์โหลดแฟ้มในเพียร์-ทู-เพียร์มายังเครื่องเกตเวย์ก่อน โดยการสั่งให้จามจุรีเอ็กซ์พลอเรอร์ทำการดาวน์โหลดแฟ้ม เมื่อแฟ้มถูกดาวน์โหลดเสร็จสมบูรณ์แล้ว ดีเอสไอก็จะอ่านข้อมูลแฟ้มทีละบล็อกจากเครื่องเกตเวย์ และส่งเข้าสู่ส่วนต่อประสานของกริดเอฟทีพี เพื่อทำหน้าที่สื่อสารโดยตรงกับโหนดปลายทางอีกทีหนึ่ง

4.4.2 **คำสั่ง** PUT

คำสั่ง PUT ถูกสั่งจากเครื่องลูกข่ายเพื่อถ่ายโอนแฟ้มจากกริดเอฟทีพีโหนดใดๆ มายังเครื่องเกตเวย์ สำหรับเพียร์-ทู-เพียร์ดีเอสไอ จะเรียกผ่าน `globus_l_gfs_p2p_recv()` โดยจะทำการรับแฟ้มข้อมูลทีละบล็อกแล้วเขียนลงในหน่วยเก็บข้อมูลของเครื่องเกตเวย์ก่อนจนเสร็จ ก็จะเสร็จสิ้นกระบวนการในฝั่งกริด ส่วนในฝั่งเพียร์-ทู-เพียร์นั้น แฟ้มที่ถูกสร้างขึ้นใหม่นี้จะถูกกระจายโดยอัตโนมัติด้วยกลไกของจามจุรีเอ็กซ์พลอเรอร์



รูปที่ 4-11 แผนผังลำดับแสดงการส่งแฟ้มจากเพียร์-ทู-เพียร์ไปยังกริด (GET)



รูปที่ 4-12 แผนผังลำดับแสดงการรับแฟ้มจากกริดมายังเพียร์-ทู-เพียร์ (PUT)

4.5 การรวมส่วนประกอบ

ในหัวข้อนี้กล่าวถึงการรวมองค์ประกอบทั้ง 4 เข้าด้วยกัน และการทำงานโดยรวมของระบบ รวมถึงการแก้ปัญหาที่เกิดจากความแตกต่างระหว่างระบบกริดและเพียร์-ทู-เพียร์

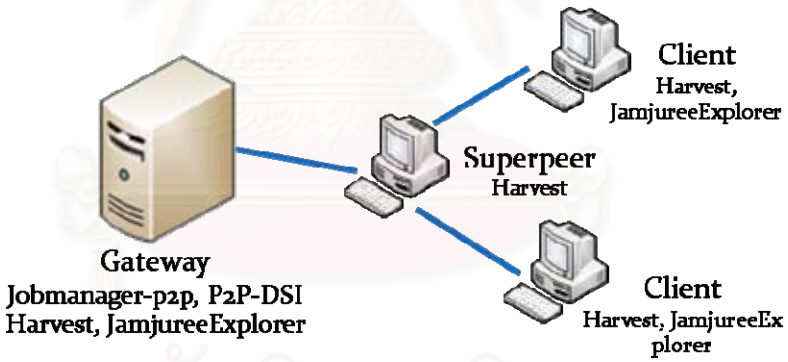
4.5.1 การติดตั้งระบบ

ส่วนย่อยแต่ละส่วนถูกติดตั้งที่เครื่องต่างๆ ต่างกันดังนี้

1) เครื่องเกตเวย์ จะมีการติดตั้งส่วนประกอบทั้ง 4 ส่วน เนื่องจากเครื่องเกตเวย์จะเป็นโหนดหนึ่งในกริด และเป็นเพียร์หนึ่งในเพียร์-ทู-เพียร์ด้วยเช่นกัน ตัวจัดการงานเพียร์-ทู-เพียร์ และเพียร์-ทู-เพียร์ดีเอส จะถูกเรจิสเตอร์เป็นบริการหนึ่งของโกลบัสที่เครื่องนี้ด้วย

2) เครื่องโวลันเทียร์ ติดตั้งโปรแกรมฮาร์เวสและจามจุรีเอ็กซ์พลอเรอร์

3) เครื่องซูเปอร์เพียร์ ติดตั้งเฉพาะโปรแกรมฮาร์เวสเท่านั้น ไม่ติดตั้งจามจุรี-เอ็กซ์พลอเรอร์ เพื่อลดภาระให้กับเครื่องซูเปอร์เพียร์



รูปที่ 4-13 การติดตั้งระบบ

4.5.2 การทำงานโดยรวม

ในการสั่งงานจากกริดมีขั้นตอนดังนี้

1) ผู้ใช้กริดสั่งงานไปยังเครื่องเกตเวย์ด้วยโปรแกรมใดๆ เช่น globus-job-run, globus-job-submit หรือ globusrun เป็นต้น (ดูตัวอย่างการสั่งงานบนกริดในภาคผนวก ก) โดยเสมือนว่างานถูกส่งมารันที่เครื่องเกตเวย์ งานที่สั่งจะอยู่ในรูปเอกสารอาร์เอสแอล

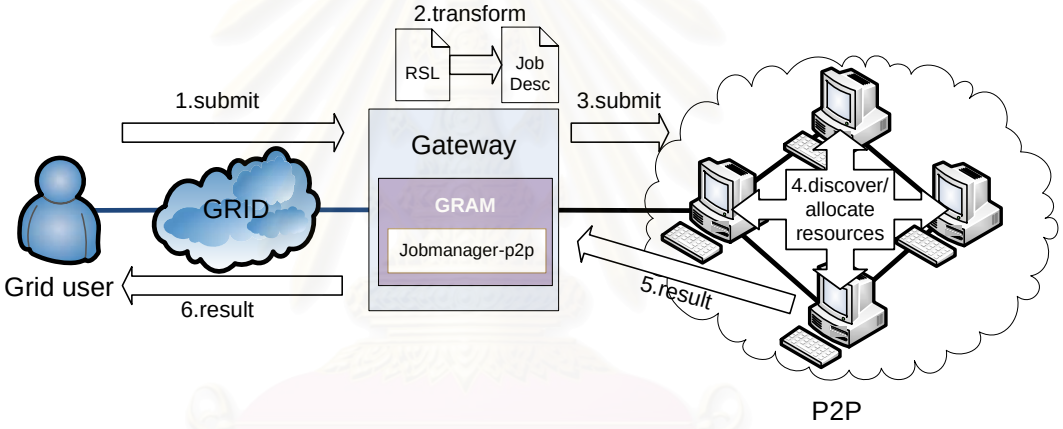
2) แกรมที่เครื่องเกตเวย์ทำการอ่านเอกสารอาร์เอสแอล และแจงออกมาเป็นลักษณะประจำต่างๆ (attributes) และเรียกฟังก์ชัน submit() ของตัวจัดการงานเพียร์-ทู-เพียร์ ในฟังก์ชันนี้จะมีการสร้างแฟ้มโฆษณางานชั่วคราว (Job Advertisement) แล้วสั่งให้โวลันเทียร์อ่านแฟ้มนี้เพื่อสั่งงาน

3) หลังจากที่โวลันเทียร์สร้างงานได้แล้ว จึงสั่งงานไปที่ซูเปอร์เพียร์

4) ด้วยกลไกของฮาร์เวส มีการค้นหาทรัพยากรที่เหมาะสม และส่งงานไปรันรวมถึงแฟ้มข้อมูลต่างๆ

5) เมื่อโวลันเทียร์ที่รับผิดชอบงานรันงานเสร็จแล้ว ก็จะส่งผลลัพธ์กลับมายังเพียร์ที่สั่งงาน(เครื่องเกตเวย์)

6) แฟ้มส่งออกที่เป็นผลลัพธ์ถูกส่งกลับไปยังผู้ใช้บนกริด



รูปที่ 4-14 กระบวนการสั่งงานของจามจุรีคลัสเตอร์

4.5.3 การจัดการบัญชีผู้ใช้

ในระดับกริด ผู้ใช้แต่ละคนจะมีบัญชีผู้ใช้เป็นของตัวเอง ซึ่งจะจับคู่กับบัญชีผู้ใช้ในระดับท้องถิ่นโดยผ่านทางกริดแมปไฟล์ (Grid mapfile) ดังนั้นบัญชีผู้ใช้กริดทุกคนเมื่อสั่งงานเข้าสู่ระบบก็จะผ่านทางเครื่องเกตเวย์ ทำให้เครื่องเกตเวย์รองรับผู้ใช้หลายคน ในขณะที่ระบบเพียร์-ทู-เพียร์นั้นเครื่องเกตเวย์ถือเป็นเพียร์ๆหนึ่ง หรือเทียบเท่ากับผู้ใช้ 1 คน ดังนั้นจึงเกิดข้อขัดแย้งขึ้นในการจัดการผู้ใช้ที่เครื่องเกตเวย์ ปัญหานี้เกิดขึ้นกับจามจุรีเอ็กซ์พลอเรอร์เนื่องจากเครื่องเกตเวย์จะมีโฮมไดเร็กทอรีของบัญชีผู้ใช้กริดแต่ละคนแยกกัน แต่ในเพียร์-ทู-เพียร์จะมีไดเร็กทอรีที่แบ่งปันเพียงไดเร็กทอรีเดียว ผู้วิจัยจึงเสนอให้ใช้ไดเร็กทอรีรากของจามจุรีเอ็กซ์พลอเรอร์เป็น /home และโฮมไดเร็กทอรีของแต่ละผู้ใช้ก็จะเป็นไดเร็กทอรีย่อยระดับแรก ซึ่งเทียบเท่ากับโฮมไดเร็กทอรีของแต่ละผู้ใช้ในระดับจามจุรีเอ็กซ์พลอเรอร์ เช่น ผู้ใช้มีบัญชีชื่อ user01 โฮมไดเร็กทอรีที่เครื่องเกตเวย์

ของผู้ใช้คนนี้จะเป็น /home/user01 และโฮมไดเร็กทอรีในจามจุรีเอ็กซ์พลอเรอร์จะเป็น /user01 ส่วนที่เพียร์อื่นๆนั้น จะใช้ชื่อเพียร์เป็นชื่อโฮมไดเร็กทอรี โดยโฮมไดเร็กทอรีจะจับคู่เข้ากับไดเร็กทอรีรากของจามจุรีเอ็กซ์พลอเรอร์ได้เลย เช่นหากเพียร์ชื่อ user02 และมีไดเร็กทอรีรากที่ /tmp/share ไดเร็กทอรีนี้จะจับคู่กับ /user02 ในจามจุรีเอ็กซ์พลอเรอร์ ส่วนแฟ้มและไดเร็กทอรีย่อยอื่นๆในโฮมไดเร็กทอรีก็จะถูกจัดตามโครงสร้างแฟ้มข้อมูลท้องถิ่น

ส่วนในการสั่งงานด้วยฮาร์เวสผ่านทาง คำอธิบายงานจะระบุชื่อเจ้าของงานเอาไว้ด้วย เช่น user01 ซึ่งจะอนุมานได้ถึงโฮมไดเร็กทอรีของผู้ใช้คนนี้ นั่นคือ /user01 เมื่อสั่งงานผ่านเกตเวย์ ตัวจัดการงานที่พัฒนาขึ้นจะสร้างคำอธิบายงานให้โดยอัตโนมัติ และระบุเจ้าของงานตามชื่อบัญชีผู้ใช้ในระดับกริด เช่น หาก user02 สั่งงานเข้ามา ระบบก็จะสร้างคำอธิบายงานโดยระบุชื่อเจ้าของงานเป็น user02 ซึ่งโฮมไดเร็กทอรีของผู้ใช้ก็คือ /user02 นั่นเอง

บทที่ 5

ผลการวิจัย

ในบทนี้จะเป็นการทดสอบซอฟท์แวร์ที่ได้พัฒนาขึ้นทั้งหมด ประกอบด้วยการทดสอบสมรรถนะของระบบสั่งงานและระบบแฟ้มข้อมูลบนเครือข่ายเพียร์-ทู-เพียร์ และทดสอบการสั่งงานผ่านทางกริด โดยการทดสอบทั้งหมดทำบนสภาพแวดล้อมจริงในภาควิชาวิศวกรรมคอมพิวเตอร์

ในการทดสอบระบบนั้น ผู้วิจัยได้ทดสอบปัจจัยต่างๆ ที่น่าจะมีผลต่อประสิทธิภาพโดยรวมของระบบ ได้แก่ จำนวนเพียร์ ขนาดของแฟ้มเข้า ขนาดของแฟ้มออก และความยาวของงาน ซึ่งจะมีผลทำให้ประสิทธิภาพต่างกันเช่น speedup, การใช้งานซีพียู, เวลาในแต่ละขั้นตอนของกระบวนการสั่งงาน เป็นต้น

5.1 เครื่องมือที่ใช้ในการทดลอง

เครื่องที่ใช้ทดสอบเป็นเครื่องในภาควิชาวิศวกรรมคอมพิวเตอร์ทั้งหมด ประกอบด้วยเครื่องที่ทำหน้าที่เป็นเครื่องคำนวณ (โวลันเทียร์) , เครื่องสั่งงาน (เกตเวย์), เครื่องจัดลำดับงาน (ซูเปอร์เพียร์) และอุปกรณ์เครือข่าย โดยมีรายละเอียดของฮาร์ดแวร์ และซอฟท์แวร์ดังนี้

1) เครื่องสั่งงาน ประกอบด้วย 2 เครื่อง ดังนี้

   ก. ซีพียู อินเทลเพนเทียม 4 ความเร็วสัญญาณนาฬิกา 3.20 กิกะเฮิรตซ์ (2 โพรเซสเซอร์), หน่วยความจำ 2 กิกะไบต์, ระบบปฏิบัติการวินโดวส์เซิร์ฟเวอร์ 2003, จาวารันไทม์เอ็นไวรอนเมนต์ รุ่น 1.6 ใช้ทดสอบการส่งงานบนระบบเพียร์-ทู-เพียร์

   ข. ซีพียู อินเทลเพนเทียม 4 ความเร็วสัญญาณนาฬิกา 2.80 กิกะเฮิรตซ์ (2 โพรเซสเซอร์), หน่วยความจำ 1 กิกะเฮิรตซ์, ระบบปฏิบัติการลินุกซ์, โกลบัสทูลคิท รุ่น 4.0.3, จาวารันไทม์เอ็นไวรอนเมนต์ รุ่น 1.5 ใช้เป็นเกตเวย์เพื่อทดสอบการสั่งงานและจัดการแฟ้มผ่านกริด

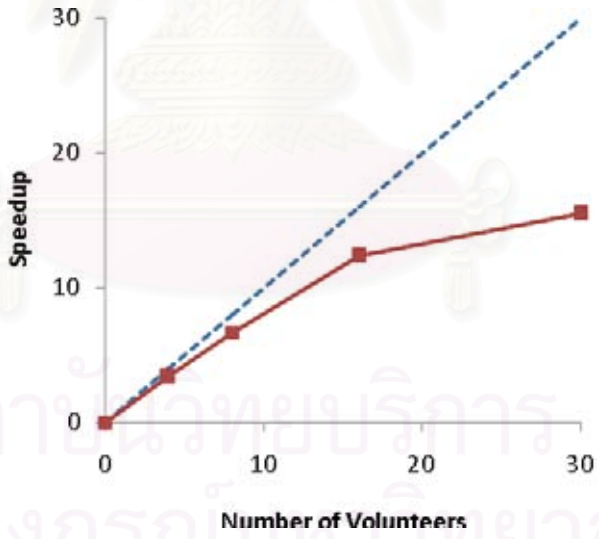
2) เครื่องซูเปอร์เพียร์ เป็นเครื่องเดียวกันกับเครื่องสั่งงาน ก.

3) เครื่องคำนวณ ซีพียู อินเทลเพนเทียม 4 ความเร็วสัญญาณนาฬิกา 1.8 กิกะเฮิรตซ์, หน่วยความจำหลัก 512 MB, ระบบปฏิบัติการวินโดวส์เอ็กซ์พี, จาวารันไทม์เอ็นไวรอนเมนต์ รุ่น 1.6

5.2 Speedup

ค่า speedup เป็นตัววัดประสิทธิภาพของระบบคำนวณเชิงขนาน ซึ่งเป็นค่าอัตราส่วนระหว่างเวลาที่ใช้สำหรับทำงานเชิงลำดับ (sequential) บนโพรเซสเซอร์เดียวกับการทำงานแบบขนาน (parallel) โดยใช้หลายโพรเซสเซอร์ ในที่นี้ผู้วิจัยทดสอบด้วยการวัดค่า speedup โดยใช้จำนวนโวลันเทียร์ต่างกัน

การทดลองนี้ทดสอบด้วยการรันโปรแกรมแมนเดลบร็อตเซ็ต (Mandelbrot Set) โดยแต่ละงานจะเป็นการสร้างรูปภาพจากกราฟแมนเดลบร็อตที่ตำแหน่งต่างๆกัน รวมทั้งสิ้นจำนวน 200 งาน ที่มีขนาดแฟ้มเข้าเหมือนกันคือ 6 กิโลไบต์ และแฟ้มออกมีขนาดต่างกันตั้งแต่ 2.7 - 205.7 กิโลไบต์ และแต่ละงานใช้เวลาในการคำนวณต่างกันตั้งแต่ 6265 - 158093 มิลลิวินาที



รูปที่ 5-1 ค่า speedup เมื่อใช้จำนวนโวลันเทียร์ต่างกัน

จากรูปที่ 5-1 ค่า speedup ของระบบ เมื่อจำนวนโวลันเทียร์ต่ำกว่า 10 เพียร์นั้นใกล้เคียงกับค่า speedup สูงสุด เมื่อเพิ่มจำนวนโวลันเทียร์ขึ้นค่า speedup จะเริ่มออกห่างจากค่า speedup สูงสุด โดยที่จำนวนโวลันเทียร์ 30 เพียร์นั้น ค่า speedup มีค่า 15.44 คิดเป็น 51.2 % ของค่า speedup สูงสุด ซึ่งกราฟเป็นไปตามกฎของอัมดาห์ล (Amdahl's law)[37]

ค่า speedup ของระบบเป็นผลมาจากส่วนของโปรแกรมเชิงลำดับ (sequential portion), ส่วนของโปรแกรมเชิงขนาน(parallel portion) และส่วนค่าใช้จ่ายอื่นๆ (overhead) ดังนี้

$$\psi(n, p) \leq \frac{\sigma(n) + \varphi(n)}{\sigma(n) + \varphi(n)/p + \kappa(n, p)} \tag{5.1}$$

โดยที่ $\sigma(n)$ แทนเวลาที่ใช้ในส่วนของโปรแกรมเชิงลำดับ, $\varphi(n)$ แทนเวลาที่ใช้ในส่วนของโปรแกรมเชิงขนาน, $\kappa(n, p)$ แทนเวลาที่ใช้ในส่วนของค่าใช้จ่ายอื่นๆ, p แทนจำนวนหน่วยประมวลผล และ $\psi(n, p)$ คือค่า speedup

ในกรณีของการทดลองนี้ โปรแกรมจะเป็นส่วนการคำนวณแบบขนานทั้งหมด เนื่องจากงานแต่ละงานไม่ได้มีส่วนเกี่ยวข้องกัน ส่วนของโปรแกรมเชิงลำดับจึงถูกละไว้ได้ ($\sigma(n) = 0$) ดังนั้น ส่วนที่จะมีผลทำให้ speedup ของระบบลดลงก็คือส่วนค่าใช้จ่ายอื่นๆ หรือ $\kappa(n, p)$ นั่นเอง

ผู้วิจัยจึงได้ใช้ตัววัดคาร์ป-แฟลต[38] (The Karp-Flatt Metric) ซึ่งสามารถบอกถึงส่วนค่าใช้จ่ายอื่นๆ ที่มีผลต่อการทำงานเชิงขนานได้เมื่อวัดจากการทดลองจริง ตัววัดคาร์ป-แฟลต สามารถคำนวณได้ดังสมการ 5.2

$$e = \frac{1/\psi - 1/p}{1 - 1/p} \tag{5.2}$$

โดยที่ e แทนส่วนอนุกรมที่ได้จากการทดลอง (experimentally determined serial fraction), ψ แทนค่า speedup ที่วัดได้ และ p แทนจำนวนหน่วยประมวลผล

จากการทดลองที่ในหัวข้อนี้ สามารถคำนวณค่า e ได้ตามตารางที่ 5-1

ตารางที่ 5-1 การคำนวณค่า e ที่ได้จากการทดลอง

p	4	8	16	30
ψ	3.38	6.64	12.33	15.45
e	0.0611	0.0293	0.01984	0.03247

จากการคำนวณ จะพบว่า ค่า e มีการเปลี่ยนแปลงในระยะแคบๆ เมื่อจำนวนโวลันเทียร์เพิ่มมากขึ้น และไม่ได้มีค่าเพิ่มขึ้นอย่างเห็นได้ชัด จึงทำให้สรุปได้ว่า ค่าใช้จ่ายอื่นๆที่ค่อนข้างคงที่นี้ทำให้เราสามารถเพิ่มจำนวนโวลันเทียร์มากขึ้นได้ โดยที่ไม่จำกัดโอกาสในการทำงานของโปรแกรมในเชิงขนาน

ในการคำนวณเชิงขนานนั้นมีปัจจัยหลายอย่างที่เป็นการเพิ่มค่าใช้จ่ายอื่นๆให้กับระบบ ได้แก่ จากการวิเคราะห์เบื้องต้น ผู้วิจัยเห็นว่าปัจจัยที่มีอาจจะมีผลได้แก่ โพรโทคอลการสั่งงาน โพรโทคอลการดาวน์โหลดแฟ้ม ขนาดแฟ้มเข้า ขนาดแฟ้มออก ระยะเวลาของการรันงาน และความสามารถของเครื่องสั่งงาน (เกตเวย์) และเครื่องจัดลำดับงาน (ซูเปอร์เพียร์) ดังนั้นในหัวข้อต่อๆไปจะเป็นการทดสอบเพื่อหาวิเคราะห์ผลกระทบจากปัจจัยเหล่านี้

5.3 โพรโทคอลการสั่งงาน

โพรโทคอลการสั่งงานของฮาร์เวสดังที่ได้กล่าวไปแล้วในหัวข้อ 4.1.1เป็นการทำงานในรูปแบบซูเปอร์เพียร์ คือ ซูเปอร์เพียร์หนึ่งจะเป็นผู้ดูแลโวลันเทียร์หลายตัว และจะคอยแจกจ่ายงานให้กับโวลันเทียร์เหล่านั้น นอกจากนี้ซูเปอร์เพียร์ยังสามารถติดต่อสื่อสารกันระหว่างซูเปอร์เพียร์อื่นได้ เพื่อกระจายภาระของซูเปอร์เพียร์ อย่างไรก็ตาม ข้อจำกัดอย่างหนึ่งของโปรแกรมฮาร์เวสรุ่นปัจจุบันคือ ยังไม่ได้ทำงานแบบซูเปอร์เพียร์อย่างเต็มที่ กล่าวคือ ซูเปอร์เพียร์ยังไม่มีโพรโทคอลในการสื่อสารกับซูเปอร์เพียร์อื่น ดังนั้นภาระของซูเปอร์เพียร์หนึ่งจึงไม่สามารถจะกระจายได้ ภาระของซูเปอร์เพียร์จึงได้แก่ การรับงานที่สั่งเข้ามา, การจัดลำดับงาน และการรับผลลัพธ์

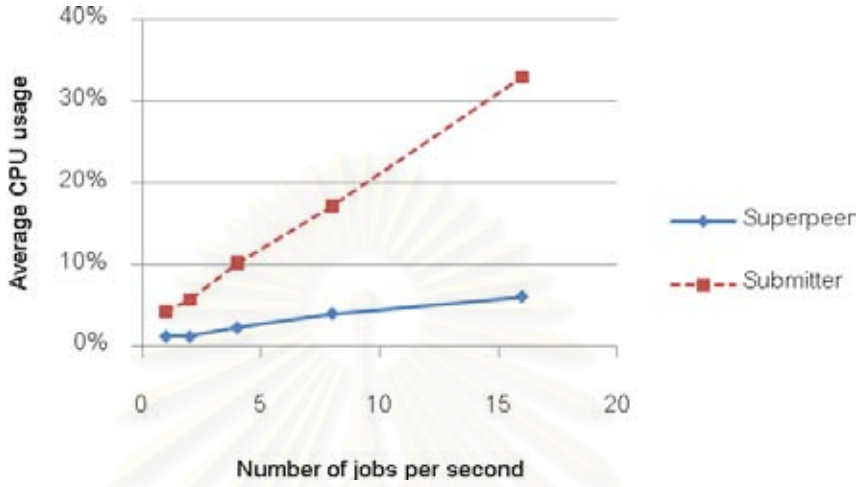
ส่วนสำคัญอีกส่วนหนึ่งก็คือโวลันเทียร์ที่สั่งงานนั้น (ในที่นี้คือเกตเวย์) จะต้องรับภาระส่วนหนึ่งในขั้นตอนต่างๆ ตลอดช่วงกระบวนการสั่งงาน ได้แก่ การสั่งงานไปยังซูเปอร์เพียร์ , การส่งงานไปยังโวลันเทียร์ที่จะรับงาน, และการรับผลลัพธ์กลับมา ซึ่งหากเพียร์นี้สั่งงานจำนวนมากก็จะทำให้เครื่องทำงานหนักได้เช่นกัน ในหัวข้อนี้จะศึกษาผลกระทบของกระบวนการเหล่านี้

การทดลองต่อไปนี้จะทดสอบภาระของเครื่องเกตเวย์ และซูเปอร์เพียร์ ในช่วงกระบวนการต่างๆ เนื่องจากเครื่องเกตเวย์จะเป็นเพียร์ที่รับงานในระบบจากกริด และมีหน้าที่สั่งงานด้วย จึงอาจจะมีภาระจำนวนมากในแต่ละขั้นตอนการสั่งงาน ส่วนซูเปอร์เพียร์นั้นจะต้องจัดการกับการจัดสรรเพียร์จำนวนมากเช่นกัน ดังนั้นการทดลองจะเป็นการวัดการใช้งานซีพียูของเครื่องเกตเวย์ และเครื่องซูเปอร์เพียร์ ที่ขั้นตอนต่างๆ โดยใช้ความถี่ของงานในแต่ละขั้นตอนต่างกัน แบ่งพิจารณาตามกระบวนการสั่งงานได้ดังนี้

5.3.1 การสั่งงาน (Job submission)

ในกระบวนการสั่งงาน ภาระของเกตเวย์จะอยู่ที่การอ่านแฟ้มโฆษณางาน (Job Advertisement) และส่งงานไปยังซูเปอร์เพียร์ ส่วนภาระของซูเปอร์เพียร์คือการรับงานจากโวลันเทียร์

การทดลองนี้ใช้การกำหนดอัตราของงานที่เกตเวย์ โดยจะสั่งงานด้วยความถี่สม่ำเสมอเป็นเวลา 1 นาที และวัดค่าการใช้งานซีพียูเฉลี่ยของทั้งซูเปอร์เพียร์และเกตเวย์ ใช้อัตราการสั่งงานที่ 1, 2, 4, 8 และ 16 งานต่อวินาที



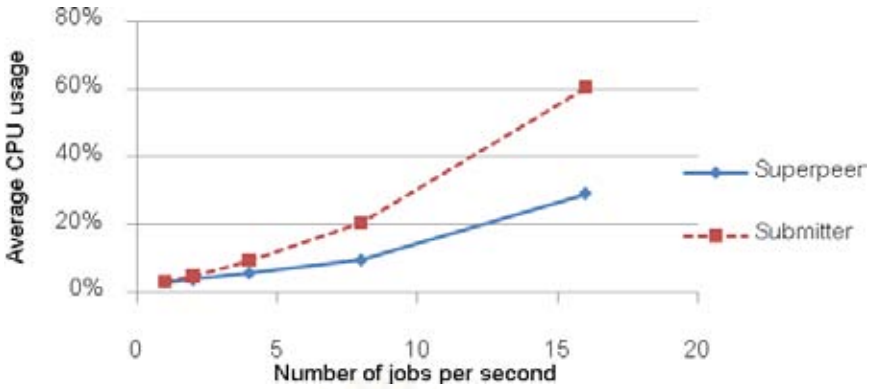
รูปที่ 5-2 การใช้งานซีพียูในขั้นตอนการสั่งงาน

จากรูปที่ 5-2 เมื่อมีการสั่งงานด้วยอัตราสูงขึ้นจะทำให้เครื่องเกตเวย์และซูเปอร์-เพียร์ทำงานหนักขึ้น โดยเฉพาะเครื่องเกตเวย์จะใช้ซีพียูสูงกว่าเครื่องซูเปอร์เพียร์มาก ค่าการใช้งานซีพียูจะเป็นอัตราส่วนกับอัตราการสั่งงานโดยประมาณ

5.3.2 **การจัดสรรงาน** (Job allocation)

ในการจัดสรรงานภาระของเกตเวย์จะอยู่ที่การรับข้อความจากซูเปอร์เพียร์ และคอยส่งงานไปรันยังโวลันเทียร์ ส่วนภาระของซูเปอร์เพียร์จะเป็นการเลือกเพียร์ที่เหมาะสมและส่งข้อความไปยังโวลันเทียร์สั่งงาน

การทดลองนี้ใช้การกำหนดอัตราการจ่ายงานที่ซูเปอร์เพียร์ โดยงานที่จ่ายนั้นได้มาจากขั้นตอนการสั่งงานในหัวข้อ 5.3.1 กำหนดอัตราการจ่ายงานที่ 1, 2, 4, 8 และ 16 งานต่อวินาที



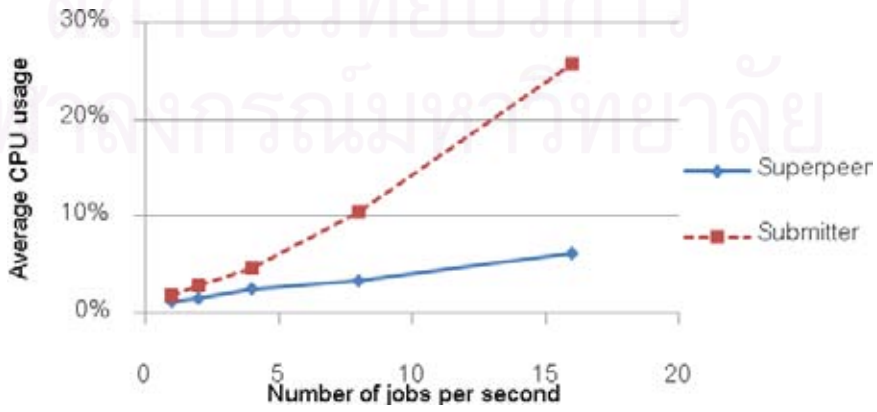
รูปที่ 5-3 การใช้งานซีพียูในขั้นตอนการจัดสรรงาน

การใช้งานซีพียูจะของซูเปอร์เพียร์และเกตเวย์จะเพิ่มขึ้นเป็นอัตราส่วนตามอัตราของงาน ตั้งแต่ 1 – 8 งานต่อวินาที และจะเพิ่มขึ้นมากที่อัตรา 16 งานต่อวินาที โดยจะเพิ่มเป็นประมาณ 3 เท่าจากที่อัตรา 8 งานต่อวินาที ค่าการใช้งานซีพียูในช่วงนี้จะสูงกว่าช่วงสั่งงาน

5.3.3 **การรับผลลัพธ์** (Getting result)

ในขั้นตอนการรับงาน ภาระของโวลันเทียร์ที่สั่งงานจะอยู่ที่การรับข้อความจากโวลันเทียร์ที่ทำงานและการดาวน์โหลดผลลัพธ์ ส่วนซูเปอร์เพียร์จะรับข้อความทำงานเสร็จจากโวลันเทียร์ที่ทำงาน

การทดลองนี้ใช้การกำหนดอัตราที่โวลันเทียร์ที่ทำงาน โดยโวลันเทียร์นี้จะส่งข้อความทำงานเสร็จไปยังเกตเวย์และซูเปอร์เพียร์ตามลำดับ จากนั้นเกตเวย์จะอ่านข้อความและดาวน์โหลดผลลัพธ์ ในที่นี้กำหนดให้ผลลัพธ์ของงานมีขนาด 1 กิโลไบต์ ส่วนซูเปอร์เพียร์จะนำโวลันเทียร์นั้นไปเข้าคิวเพื่อรับงานใหม่ กำหนดอัตราการส่งข้อความทำงานเสร็จของโวลันเทียร์เป็น 1, 2, 4, 8 และ 16 งานต่อวินาที



รูปที่ 5-4 การใช้งานซีพียูในขั้นตอนการรับผลลัพธ์

การใช้งานซีพียูจะเพิ่มขึ้นเป็นอัตราส่วนตามอัตรางานโดยประมาณเช่นเดียวกับ 2 ขั้นตอนแรก และในขั้นตอนนี้ทั้งซูเปอร์เพียร์และเกตเวย์จะมีการใช้งานซีพียูน้อยที่สุดเมื่อเทียบกับ 2 ขั้นตอนแรก

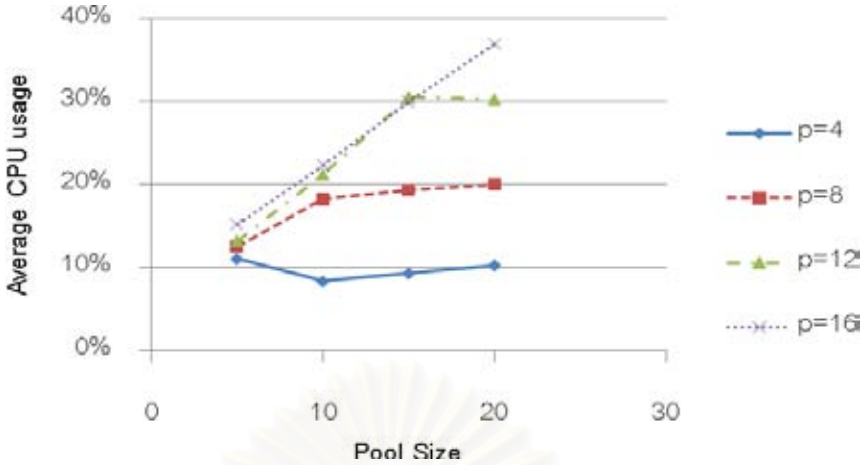
โดยสรุป เมื่อระบบมีการทำงานมากขึ้น ภาระของระบบจะตกอยู่ที่เกตเวย์มากกว่าที่ซูเปอร์เพียร์ โดยเฉพาะในกรณีที่มีการสั่งงานมากขึ้นถึงระดับหนึ่งเครื่องเกตเวย์จะมีการใช้งานซีพียูสูงมาก

อย่างไรก็ตาม ในกระบวนการทำงานจริงทั้ง 3 ขั้นตอนอาจจะเกิดขึ้นพร้อมๆกันได้ และอาจมีอัตราของงานที่ต่างกัน ดังนั้นค่าการใช้งานซีพียูอาจจะเพิ่มหรือลดกว่านี้ในบางเวลาได้

5.4 **ขนาดพูลข้อความ** (Message Pool Size)

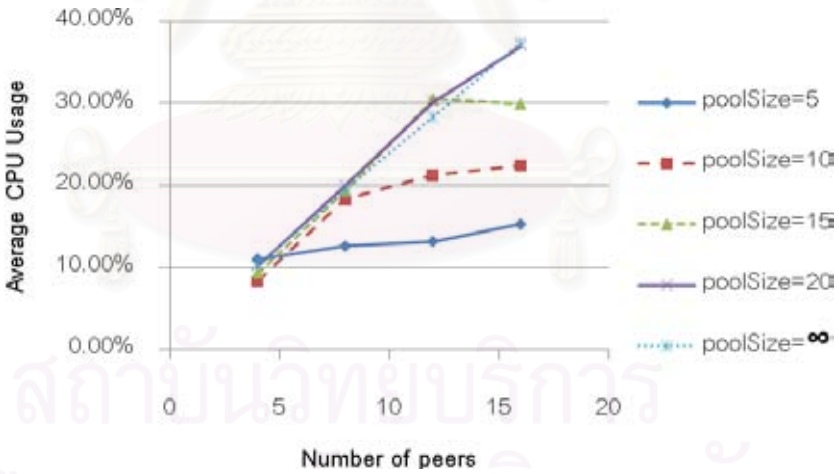
จากการผลทดลองในหัวข้อ 5.3 พบว่าภาระของซีพียูส่วนใหญ่จะอยู่ที่เกตเวย์ และโดยเฉพาะในขั้นตอนการจัดสรรงานนั้น พบว่าที่อัตราการจัดสรรงาน 16 งานต่อวินาทีมีการใช้งานซีพียูเพิ่มขึ้นจากอัตรา 8 งานต่อวินาทีถึงประมาณ 3 เท่า ซึ่งมากกว่าที่ควรจะเป็น ยิ่งไปกว่านั้นในการทำงานจริงๆนั้น ทั้งขั้นตอนการสั่งงาน การจัดสรรงาน และการเก็บผลลัพธ์อาจเกิดขึ้นพร้อมๆกัน ทำให้การใช้งานซีพียูอาจสูงขึ้นได้อีก ซึ่งการใช้งานซีพียูมากเกินไปอาจทำให้ระบบไม่เสถียร และไม่สามารถรองรับการทำงานได้ทั้งหมด เช่น ข้อความบางข้อความอาจถูกละทิ้งไป, การสอบถาม (query) ทำไม่สำเร็จ (ทั้งระดับจักซ์ตา และระดับโปรแกรมประยุกต์ชั้นบน) เป็นต้น ดังนั้นเพื่อให้ระบบดำเนินไปได้โดยถูกต้องจึงควรมีการกำหนดขอบเขตการใช้งานซีพียู

การกำหนดการใช้งานซีพียูโดยรวมของระบบนั้นทำได้ยาก ผู้วิจัยจึงใช้วิธีการจำกัดกิจกรรมของระบบแทน โดยใช้พูลข้อความ ซึ่งเป็นการจำกัดจำนวนข้อความที่จะประมวลผลได้พร้อมกันในเวลาหนึ่งๆ แล้ววัดการใช้งานซีพียูของระบบ



รูปที่ 5-5 การใช้งานซีพียูของเครื่องเกตเวย์เมื่อใช้พูลข้อความ พิจารณาแยกตามจำนวนเพียร์

รูปที่ 5-5เป็นกราฟแสดงการใช้งานซีพียูของเครื่องเกตเวย์เมื่อทดลองใช้พูลข้อความขนาดต่างๆ และทดลองโดยการสั่งงานไปยังโวลันเทียร์จำนวนต่างๆกัน ผลการทดลองพบว่า เมื่อเพิ่มจำนวนโวลันเทียร์มากขึ้น ที่เครื่องเกตเวย์ก็จะใช้งานซีพียูมากขึ้น สังเกตได้จากการที่กราฟแต่ละเส้นจะเริ่มคงที่ ณ จุดๆหนึ่ง และเส้นที่จำนวนโวลันเทียร์มากขึ้นจะมีจุดคงที่สูงขึ้น ซึ่งหากเราไม่จำกัดขนาดพูลก็จะทำให้กราฟสูงขึ้นไปเรื่อยๆตามขนาดของโวลันเทียร์

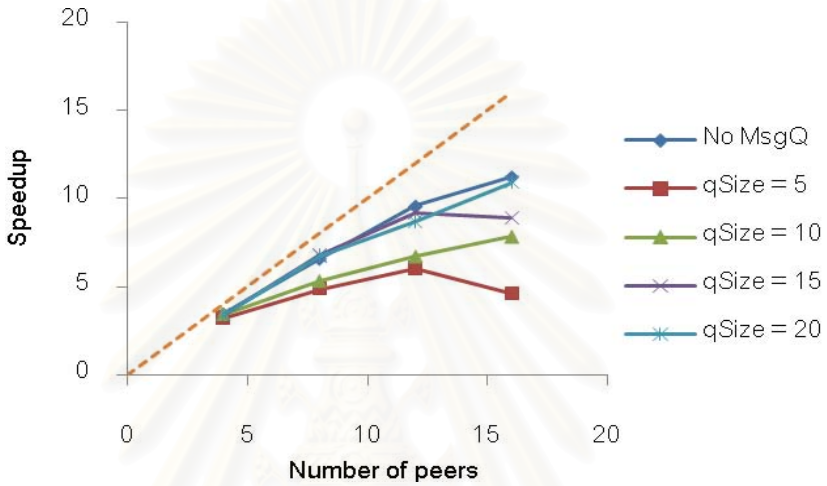


รูปที่ 5-6 การใช้งานซีพียูของเครื่องเกตเวย์เมื่อใช้พูลข้อความ พิจารณาแยกตามขนาดพูล

รูปที่ 5-6 เป็นผลการทดลองเดียวกันกับรูปที่ 5-5 แต่พิจารณาแยกตามขนาดของพูล โดยจากกราฟ จะเห็นว่าเมื่อกำหนดขนาดพูลคงที่ จะสามารถจำกัดการใช้งานซีพียูได้ที่ระดับหนึ่ง ซึ่งค่อนข้างคงที่แม้จำนวนโวลันเทียร์จะเพิ่มขึ้น และหากเพิ่มขนาดพูลมากขึ้น ก็จะจำกัดการใช้งานซีพียูให้คงที่ได้ที่จุดๆหนึ่งซึ่งสูงกว่าเดิม จึงแสดงว่าเราสามารถเพิ่มจำนวนโวลันเทียร์ได้มากขึ้นโดยยังควบคุมการใช้งานซีพียูได้เช่นเดิม

อย่างไรก็ถาม เมื่อมีการจำกัดการใช้งานของระบบก็จะมีผลให้ speedup ลดลงเช่นกัน ดังนั้นผู้วิจัยจึงได้วัด speedup ของระบบเมื่อใช้พูลข้อความขนาดต่างๆกันด้วย

รูปที่ 5-7 เป็นการวัดค่า speedup ของระบบเมื่อใช้ขนาดพูลข้อความต่างกัน พบว่าเมื่อใช้ขนาดพูลน้อยๆ ค่า speedup จะเพิ่มขึ้นเพียงเล็กน้อยหรือคงที่ หรือลดลง ถึงแม้จะเพิ่มจำนวนโวลันเทียร์มากขึ้น แต่เมื่อขนาดพูลเป็น 20 จะยังให้ค่า speedup ที่ใกล้เคียงกับค่า speedup เมื่อไม่ใช้พูลข้อความจนถึงที่จำนวนโวลันเทียร์เป็น 16 เพียร์



รูปที่ 5-7 ค่า speedup ของระบบเมื่อใช้พูลข้อความ

ในการใช้พูลข้อความ เมื่อเพิ่มจำนวนโวลันเทียร์จนถึงจุดๆหนึ่ง ค่า speedup จะไม่เพิ่มขึ้นอีกต่อไป และอาจจะตกลงได้ ซึ่งการที่ speedup ลดลงอาจเกิดได้จากปัจจัยหลายอย่าง เช่น ข้อความที่อยู่ในคิวเป็นข้อความผลลัพธ์พร้อมๆกัน ทำให้เกตเวย์ต้องใช้เวลานี้ในการดาวน์โหลดแฟ้มข้อมูลหลายๆแฟ้มพร้อมกัน ซึ่งอาจจะใช้แบนด์วิดท์จำนวนมาก ทั้งๆที่อาจมีข้อความการจัดสรรงานต่อไปที่ส่งมาจากซูเปอร์เพียร์ที่ยังรอประมวลผลอยู่ เมื่อข้อความนี้ถูกผลักออกไปก็ทำให้สูญเสียการทำงานแบบขนานในช่วงเวลาหนึ่งได้ ปัญหานี้อาจแก้ได้โดยการกำหนดลำดับความสำคัญ (priority) ของข้อความที่อยู่ในคิว

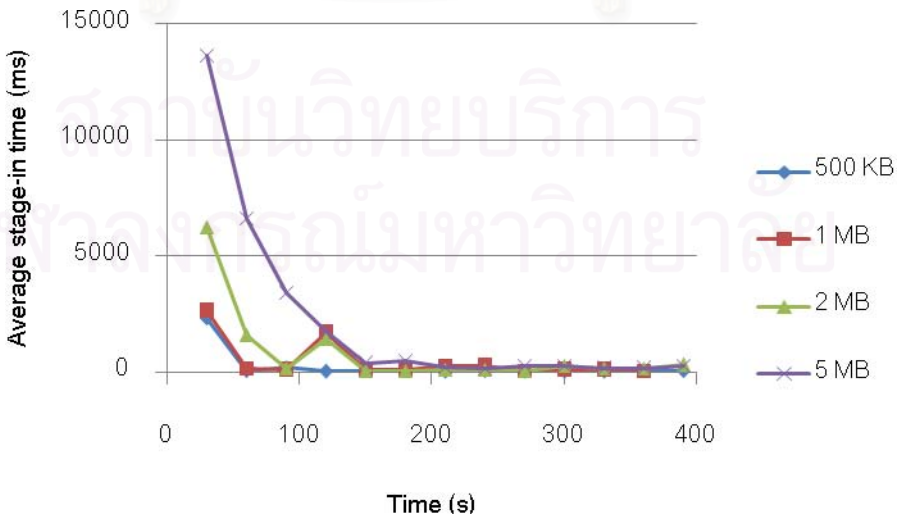
จากการทดลองนี้จึงสรุปได้ว่า ขนาดพูลสามารถควบคุมปริมาณการใช้งานซีพียูได้ และสามารถรองรับจำนวนโวลันเทียร์ได้มากขึ้นเรื่อย แต่ในขณะเดียวกัน หากจำกัดขนาดพูลไว้ก็จะทำให้ speedup ถูกจำกัดด้วยเช่นกัน ดังนั้นการเลือกขนาดพูลที่เหมาะสมจะต้องแลกเปลี่ยนระหว่างการขยายขนาดของระบบ (scalability) และ speedup

5.5 ขนาดแฟ้มเข้า

การดาวน์โหลดแฟ้มถือเป็นค่าใช้จ่าย (overhead) อย่างหนึ่งของระบบโดยเฉพาะเมื่อมีคอขวดอยู่ที่เครื่องเกตเวย์จะทำให้การส่งแฟ้มเข้าไปประมวลผลที่โวลันเทียร์ปลายทางหลายๆเพียร์นั้นต้องใช้แบนด์วิดท์จำนวนมาก ซึ่งอาจทำให้ความสามารถในการทำงานเชิงขนานช้าลงได้ งานวิจัยนี้ได้เสนอจามจุรีเอ็กซ์พลอเรอร์มาช่วยจัดการเรื่องแฟ้มข้อมูลซึ่งน่าจะช่วยให้การถ่ายโอนแฟ้มทำได้ดีขึ้น

หัวข้อนี้เป็นการทดสอบความสามารถของจามจุรีเอ็กซ์พลอเรอร์ในการดาวน์โหลดแฟ้มเข้าเพื่อคำนวณ โดยการสั่งงานที่ใช้แฟ้มเข้าขนาดต่างๆกัน 4 แฟ้ม ได้แก่ 500 กิโลไบต์, 1 เมกะไบต์, 2 เมกะไบต์ และ 5 เมกะไบต์ การสั่งงานจะเป็นแบบสุ่ม คือเลือกสั่งงานที่ใช้แฟ้มเหล่านี้คละกัน แล้ววัดเวลาในการดาวน์โหลดแฟ้มโดยเฉลี่ยของในช่วงเวลาต่างๆ จำนวนโวลันเทียร์ที่รับงานมีจำนวน 8 เพียร์

ในการดาวน์โหลดแฟ้มด้วยจามจุรีเอ็กซ์พลอเรอร์ จะประกอบด้วยขั้นตอนการค้นหาแฟ้มและดาวน์โหลดแฟ้มโดยใช้ระเบียบวิธีตามหัวข้อ 4.2.7 ในการค้นหาแฟ้ม จะเริ่มจากการค้นหาในไดเร็กทอรีราก และในไดเร็กทอรีแคช ซึ่งหากพบในไดเร็กทอรีรากก็จะนำแฟ้มมาใช้ได้ทันที ถ้าพบในไดเร็กทอรีแคช จะต้องตรวจสอบรุ่นของแฟ้มก่อน โดยการค้นหาด้วยจามจุรีเอ็กซ์พลอเรอร์จะได้ข้อมูลของแฟ้มซึ่งประกอบด้วยตราเวลา (timestamp) และค่าแฮชของแฟ้ม หากตราเวลาตรงกันก็เอามาใช้ได้เลย และหากไม่ตรงก็จะมีการคำนวณค่าแฮชของแฟ้มว่าเป็นแฟ้มเดียวกันหรือไม่ หากไม่ตรงกันจึงจะมีการดาวน์โหลดแฟ้มจากเครือข่ายจริง



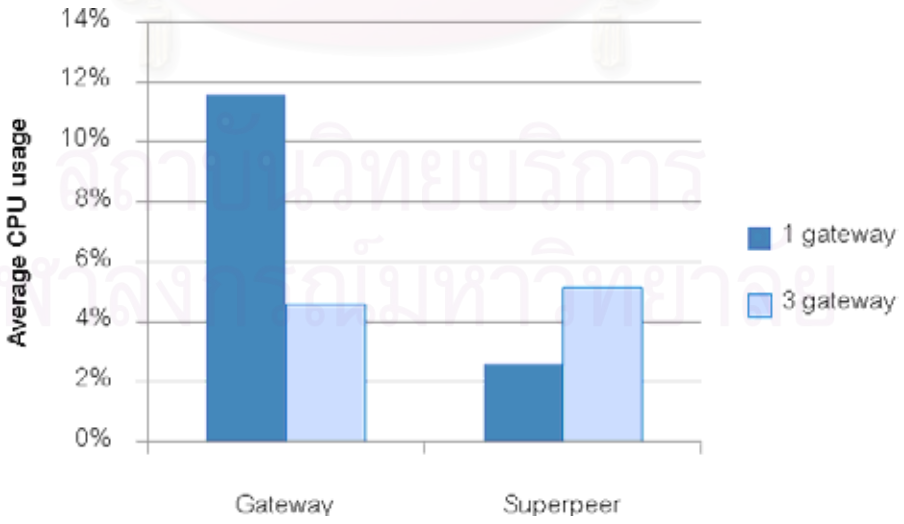
รูปที่ 5-8 เวลาในการดาวน์โหลดแฟ้มเข้า

จากรูปที่ 5-8 เมื่อโวลันเทียร์เริ่มรับงานในช่วงแรก เพียร์ส่วนใหญ่จะยังไม่มีแฟ้มเข้าในไดเร็กทอรีแคช จึงต้องมีการค้นหาและดาวน์โหลดจากเครือข่าย ซึ่งอาจเป็นการดาวน์โหลดแบบขนานจากหลายๆเพียร์ หลังจากที่โวลันเทียร์ใดๆทำงานเสร็จไปแล้วในครั้งแรก การรับงานครั้งต่อไปจะตรวจสอบพบได้ว่ามีแฟ้มเข้าที่เคยดาวน์โหลดไว้แล้ว โวลันเทียร์จึงมีหน้าที่เพียงตรวจสอบรุ่นของแฟ้ม จะเห็นได้ว่าใช้เวลาในการดาวน์โหลดลดลงมากจนคงที่ เวลาที่ใช้ในช่วงหลังจากนี้เป็นการตรวจสอบค่าแฮชของแฟ้มเท่านั้น ซึ่งการตรวจสอบแฮชของแฟ้มขนาดใหญ่ก็จะใช้เวลามากกว่าแฟ้มขนาดเล็กเพียงเล็กน้อย ดังนั้นการตรวจสอบแฟ้มของจามจุรีเอ็กซ์พลอเรอร์เช่นนี้จะช่วยลดเวลาของการทำงานโดยรวม และเพิ่ม speedup ให้กับระบบได้

5.6 การใช้หลายเกตเวย์

ระบบฮาร์เวสรองรับการสั่งงานจากหลายเพียร์อยู่แล้ว จึงสามารถติดตั้งเกตเวย์หลายที่ได้ ซึ่งการที่มีเกตเวย์จำนวนมากก็จะช่วยให้ลดภาระที่จะต้องสั่งงานผ่านทางเกตเวย์เดียว นอกจากนี้ด้วยความสามารถของจามจุรีเอ็กซ์พลอเรอร์ เราสามารถมองเห็นโครงสร้างแฟ้มทั้งระบบเป็นโครงสร้างเดียวกัน ในการสั่งงานที่หลายเกตเวย์จึงไม่ต้องนำแฟ้มเข้าไปวางไว้ที่แต่ละเกตเวย์ แต่วางไว้ที่เกตเวย์ตัวใดตัวหนึ่ง แล้วเกตเวย์ที่เหลือก็จะสามารถสั่งงานได้เสมือนกับได้วางแฟ้มนั้นไว้แล้ว

การทดลองในหัวข้อนี้จะเป็นการทดสอบการใช้งานซีพียูของเครื่องเกตเวย์และเครื่องซูเปอร์เพียร์ เมื่อใช้จำนวนเกตเวย์ในการสั่งงานต่างกัน



รูปที่ 5-9 การใช้งานซีพียูเมื่อใช้จำนวนเกตเวย์ในการสั่งงานต่างกัน

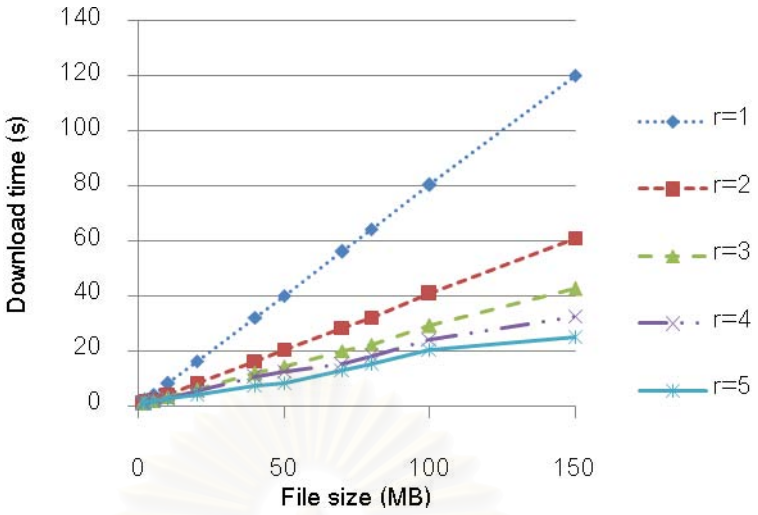
รูปที่ 5-9 แสดงการใช้งานซีพียูของเครื่องเกตเวย์และเครื่องซูเปอร์เพียร์เมื่อใช้จำนวนเกตเวย์ที่สั่งงานต่างกัน โดยวัดการใช้งานซีพียูเฉลี่ย ทั้ง 2 กรณีที่ทดสอบโดยใช้จำนวนโวลันเทียร์ทำงาน 8 เพียร์ และใช้จำนวนงานเท่ากัน ในกรณีที่ใช้เกตเวย์ 3 เครื่องนั้นใช้การแบ่งจำนวนงานออกเป็น 3 ส่วนให้เกตเวย์แต่ละเครื่อง ผลการทดลองพบว่า การใช้งานซีพียูเฉลี่ยของเกตเวย์เมื่อใช้ 3 เครื่องนั้น แต่ละเครื่องจะใช้งานน้อยลงอย่างเห็นได้ชัด ส่วนที่ซูเปอร์เพียร์เมื่อใช้หลายเกตเวย์จะทำงานหนักขึ้น อย่างไรก็ตามการใช้งานซีพียูของเครื่องซูเปอร์เพียร์นั้นน้อยอยู่แล้วจึงสามารถรองรับการเพิ่มเกตเวย์จำนวนมากได้

การใช้งานเกตเวย์หลายเครื่องนั้นย่อมหมายถึงการมีกริดโหนดหลายตัว ซึ่งในระบบกริดหมายถึงเครื่องเหล่านั้นจะต้องมีความเชื่อใจกัน (trust) ดังนั้นเกตเวย์เหล่านี้ควรจะอยู่ในองค์กรเสมือนเดียวกันด้วย ประเด็นนี้เป็นเรื่องของความปลอดภัยของข้อมูลในระดับกริดซึ่งอาจจะต้องมีการจัดการเพิ่มเติมนอกเหนือจากขอบเขตของงานวิจัยนี้ ในระบบเพียร์-ทู-เพียร์ไม่ได้คำนึงถึงส่วนนี้ อย่างไรก็ตามเราสามารถจำกัดขอบเขตการกระจายข้อมูลในเครือข่ายเพียร์-ทู-เพียร์ได้โดยใช้การเข้ากลุ่มเพียร์ที่ปลอดภัย (secure peer group) ซึ่งมีระเบียบวิธีที่สนับสนุนโดยจักซ์ตาอยู่แล้ว

5.7 การดาวน์โหลดแบบขนาน

จามจุรีเอ็กซ์พลอเรอร์สนับสนุนการดาวน์โหลดแฟ้มแบบขนาน โดยใช้ความสามารถของซีเอ็มเอส (CMS) การดาวน์โหลดแบบขนานจะใช้คอนเทนต์แอดเวอทิสเมนต์ (Content Advertisement) หลายๆตัว แต่ละตัวจะบอกที่อยู่ของแฟ้มสำเนาที่ต่างๆกัน ซึ่งจามจุรีเอ็กซ์พลอเรอร์นั้นจะมีการทำซ้ำของคอนเทนต์แอดเวอทิสเมนต์และแฟ้มสำเนาเอาไว้อยู่แล้ว ทำให้สามารถใช้ฟังก์ชันนี้ได้

การทดลองต่อไปนี้จะเป็นการทดสอบประสิทธิภาพของจามจุรีเอ็กซ์พลอเรอร์ในการดาวน์โหลดแบบขนาน โดยใช้จำนวนการทำซ้ำ (replication) ต่างๆกัน และขนาดแฟ้มต่างๆกัน และวัดความเร็วในการถ่ายโอนข้อมูล



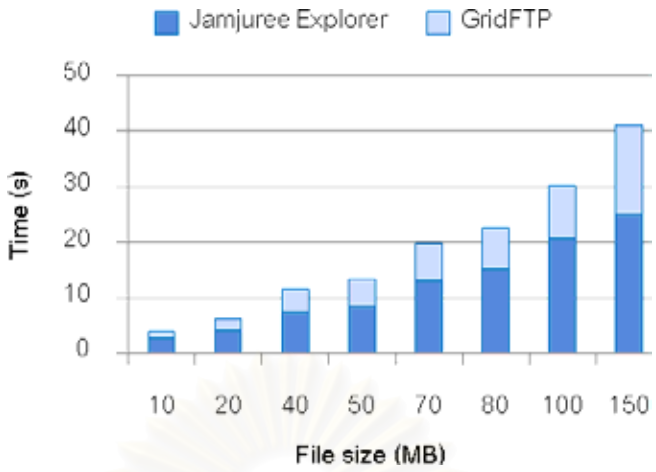
รูปที่ 5-10 การดาวโหลดแฟ้มโดยใช้จำนวนการทำซ้ำต่างๆ

จากรูปที่ 5-10 ใช้จามจุรีเอ็กซ์พลอเรอร์ในการดาวน์โหลดแฟ้มขนาดต่างๆกัน ตั้งแต่ 1 – 150 เมกะไบต์ และกำหนดให้ใช้การทำซ้ำต่างๆกันตั้งแต่ 1 – 5 สำเนา พบว่าเมื่อดาวน์โหลดโดยใช้จำนวนการทำซ้ำคงที่จะใช้เวลาเพิ่มขึ้นตามขนาดของแฟ้มเป็นเชิงเส้น และเมื่อใช้จำนวนการทำซ้ำมากขึ้นก็จะใช้เวลาในการถ่ายโอนข้อมูลน้อยลง เมื่อคำนวณเป็นอัตราการถ่ายโอนพบว่าที่จำนวนการทำซ้ำเป็น 1 จะมีอัตราการถ่ายโอนเป็น 1.25 เมกะไบต์ต่อวินาที และที่จำนวนการทำซ้ำเป็น 5 จะมีอัตราการถ่ายโอนเป็น 5.71 เมกะไบต์ต่อวินาที ซึ่งเร็วขึ้นประมาณ 4.57 เท่า

ในการทำงานจริงของจามจุรีเอ็กซ์พลอเรอร์นั้นไม่อาจคาดการณ์จำนวนการทำซ้ำได้ เนื่องจากขึ้นอยู่กับสำเนาที่จะพบ ณ เวลานั้น โดยมีตัวแปรที่สำคัญคือเวลารอ (timeout) หากหมดเวลารอแล้วก็จะใช้จำนวนสำเนาเท่าที่หาได้ ซึ่งอาจได้จำนวนมากหรือน้อยขึ้นอยู่กับเครือข่ายและเวลาที่แฟ้มนั้นอยู่ในเครือข่ายด้วย

5.8 การถ่ายโอนแฟ้มผ่านกริด

การทดลองนี้เป็นการถ่ายโอนแฟ้มจากเพียร์-ทู-เพียร์ไปยังกริด โดยการเรียกคำสั่ง globus-url-copy กำหนดต้นทางเป็นเครื่องเกตเวย์ และปลายทางเป็นอีกเครื่องหนึ่งในกริด ขั้นตอนการทำงานคือเมื่อเกตเวย์รับคำร้องขอแฟ้ม ดีเอสไอจะสั่งให้จามจุรีเอ็กซ์พลอเรอร์ดาวน์โหลดแฟ้มจากเครือข่ายมายังเครื่องเกตเวย์ก่อน โดยในที่นี้จะใช้จำนวนการทำซ้ำเป็น 5 หลังจากที่ดาวน์โหลดเสร็จแล้วดีเอสไอก็จะส่งแฟ้มนั้นผ่านโพรโทคอลกริดเอฟทีพีไปยังเครื่องปลายทางอีกทีหนึ่ง การทดลองนี้จะวัดเวลาของการถ่ายโอนแฟ้มในส่วนเพียร์-ทู-เพียร์ และส่วนกริด



รูปที่ 5-11 เวลาในการถ่ายโอนแฟ้มจากเพียร์-ทู-เพียร์ไปยังกริด

จากรูปที่ 5-11 พบว่าการถ่ายโอนแฟ้มด้วยจามจุรีเอ็กซ์พลอเรอร์จะใช้เวลาประมาณ 2/3 ของเวลาในการถ่ายโอนแฟ้มทั้งหมด ซึ่งอีก 1/3 ที่เหลือก็คือเวลาในการถ่ายโอนแฟ้มจากเครื่องเกตเวย์ไปยังเครื่องปลายทางด้วยโพรโทคอลกริดเอฟทีพี เมื่อคำนวณความเร็วของกริดเอฟทีพีพบว่าใกล้เคียงกับความเร็วสูงสุดของฟาสต์อีเทอร์เน็ต (100 Mb/s) การที่ถ่ายโอนจากเพียร์-ทู-เพียร์ได้ความเร็วน้อยกว่ากริดเอฟทีพีนั้นมีสาเหตุหลักมาจากโพรโทคอลซีเอ็มเอสที่จะมีการหยุดรอเป็นระยะเพื่อตรวจสอบความสำเร็จของการถ่ายโอนแต่ละกลุ่มข้อมูล (packet) ช่วงของการหยุดรอนี้ทำให้เสียเวลาไปบางส่วน แต่การดาวน์โหลดแบบขนานก็ช่วยให้ถ่ายโอนข้อมูลเร็วขึ้นได้ระดับหนึ่ง

บทที่ 6

สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอการเชื่อมต่อระหว่างกริดและเพียร์-ทู-เพียร์เพื่อรองรับระบบการคำนวณแบบกระจาย ผลที่ได้จากการพัฒนาระบบดังกล่าวคือช่วยให้สามารถเพิ่มทรัพยากรให้แก่กริดได้ง่ายโดยการใช้เทคโนโลยีเพียร์-ทู-เพียร์เข้ามาช่วยรวบรวมทรัพยากร ซึ่งทรัพยากรในที่นี้คือเครื่องคอมพิวเตอร์ส่วนบุคคลที่มีอยู่อย่างกระจายบนอินเทอร์เน็ต เมื่อนำทรัพยากรมารวมกันแล้วจะมีลักษณะเหมือนคลัสเตอร์คอมพิวเตอร์ขนาดใหญ่ที่มีเครื่องประมวลผลจำนวนมาก และมีส่วนเชื่อมต่อระหว่างระบบเพียร์-ทู-เพียร์กับกริด ที่ทำหน้าที่เหมือนมาสเตอร์โหนดของคลัสเตอร์ ระบบดังกล่าวนี้ชื่อว่าจามจุรีคลัสเตอร์

บริการที่มีให้สำหรับระบบที่พัฒนาขึ้นนี้ได้แก่ บริการการคำนวณแบบกระจาย และบริการหน่วยเก็บข้อมูล ทั้งสองบริการนี้ทำงานเกี่ยวข้องกันในระดับเพียร์-ทู-เพียร์ กล่าวคือการคำนวณแบบกระจายต้องใช้ระบบแฟ้มแบบกระจายในการจัดการแฟ้มข้อมูล นอกจากนี้บริการทั้งสองอย่างนี้อนุญาตให้ผู้ใช้สามารถเรียกใช้ได้จากระบบกริดโดยผ่านทางมาตรฐานและวิธีการของกริดที่มีอยู่แล้ว ดังนั้นผู้ใช้จากกริดจึงสามารถเรียกใช้บริการเหล่านี้ได้โดยไม่ต้องรู้ว่ามีการจัดการในระดับทรัพยากรอย่างไร

ผู้วิจัยได้พัฒนาระบบแฟ้มข้อมูลแบบกระจายบนเพียร์-ทู-เพียร์ขึ้น ซึ่งมีจุดประสงค์เพื่อให้ระบบรองรับกับการถ่ายโอนข้อมูลจำนวนมากได้ โดยเฉพาะรองรับกับระบบจ่ายงานของฮาร์เวส จามจุรีเอ็กซ์พลอเรอร์ใช้หลักการของระบบร่วมแฟ้ม และเพิ่มเติมเทคนิคต่างๆในแบบเพียร์-ทู-เพียร์ เช่น การทำซ้ำของข้อมูล การแคช การดาวน์โหลดแบบขนาน และการควบคุมรุ่นของแฟ้ม ซึ่งผลการทดลองพบว่าเทคนิคเหล่านี้ช่วยเพิ่มประสิทธิภาพโดยรวมให้กับระบบได้อย่างดี เมื่อนำมาใช้กับฮาร์เวสก็ทำให้ฮาร์เวสทำงานได้ดีขึ้น โดยช่วยลดเวลาในขั้นตอนในการถ่ายโอนแฟ้มลงไปได้มาก

การทดสอบระบบบนสภาพแวดล้อมจริงในบทที่ 5 พบว่า ส่วนที่จะมีผลต่อระบบเชื่อมต่อกริดกับเพียร์-ทู-เพียร์มากคือเครื่องเกตเวย์ เนื่องจากเป็นเครื่องที่เป็นทางผ่านของข้อมูลระหว่างกริดกับเพียร์-ทู-เพียร์ทั้งหมด อีกทั้งตลอดขั้นตอนการสั่งงานเครื่องเกตเวย์จะมีภาระค่อนข้างสูงเมื่อเทียบกับซูเปอร์เพียร์ ซึ่งอาจเป็นปัญหาคอขวดของระบบได้ ผู้วิจัยได้นำเสนอ

วิธีการลดภาระของเครื่องเกตเวย์ ได้แก่ การจำกัดพูลข้อความที่เกตเวย์จะประมวลผลได้ในเวลาหนึ่งๆ และการเพิ่มจำนวนเกตเวย์ อย่างไรก็ตาม ก็ต้องแลกกับประสิทธิภาพโดยรวมที่อาจลดลง

ระบบจามจุรีคลัสเตอร์ติดตั้งได้ง่ายโดยใช้เทคโนโลยีจาวาเว็บสตาร์ท (Java Web Start) ทำให้สามารถเพิ่มจำนวนทรัพยากรบนเพียร์-ทู-เพียร์ได้ง่ายและรวดเร็ว และด้วยเทคโนโลยีของจาวา ทำให้สามารถติดตั้งจามจุรีคลัสเตอร์ได้บนทุกแพลตฟอร์ม ซึ่งในการทดลองผู้วิจัยได้ติดตั้งบนระบบปฏิบัติการวินโดวส์และลินุกซ์ก็พบว่าระบบสามารถทำงานได้เป็นอย่างดี

6.2 การพัฒนาซอฟท์แวร์

ซอฟท์แวร์ที่พัฒนาขึ้นในงานวิจัยนี้ทั้งหมด ได้แก่

1. จามจุรีเอ็กซ์พลอเรอร์ (Jamjuree Explorer) พัฒนาขึ้นใหม่ทั้งหมด เพื่อแทนที่โปรแกรมเดิม มีรายละเอียดการพัฒนาดังนี้
   - กลไกการแบ่งปันแฟ้ม โดยใช้หลักการตารางแฮชแบบกระจาย
   - การจัดโครงสร้างแฟ้มแบบลำดับชั้น (hierarchical file structure)
   - กลไกการค้นหาแฟ้ม และดาวน์โหลดแฟ้ม
   - สร้างเอพีไอ (API) และสร้างทางเชื่อมต่อจากภายนอกโดยการเปิดซ็อกเก็ตรอรับคำสั่ง
   - ระบบการควบคุมรุ่น
   - การดาวน์โหลดแบบขนาน
2. ฮาร์เวส (Harvest) เป็นการแก้ไขโปรแกรมบางส่วน
   - ปรับปรุงกลไกการเชื่อมต่อระหว่างโวลันเทียร์กับซูเปอร์เพียร์ให้เสถียรขึ้น
   - เปลี่ยนวิธีการรับส่งแฟ้ม โดยให้มาใช้จามจุรีเอ็กซ์พลอเรอร์แทน
   - เพิ่มกลไกการเรียกใช้ตัวกระทำการ (executor)
   - เพิ่มกลไกพูลข้อความ (message pool)
   - สร้างส่วนต่อประสานกราฟิกกับผู้ใช้ (GUI) สำหรับโวลันเทียร์
   - สร้างทางเชื่อมต่อจากภายนอกโดยการเปิดซ็อกเก็ตรอรับคำสั่ง
   - สร้างแฟ้ม jnlp สำหรับติดตั้งโปรแกรมด้วยจาวาเว็บสตาร์ท (Java Web Start)

3. ตัวจัดการงานเพียร์-ทู-เพียร์ (Jobmanager-p2p)
    - เขียนเมท็อด submit, poll และ cancel ซึ่งขยายจากตัวจัดการงาน โดยประกอบด้วย การแปลงอาร์เอสแอลเป็นโฆษณางาน และการแปลงสถานะของงาน
    - สร้างแฟ้ม RVF (resource verification file) เพื่อเพิ่มเติมการแจงลักษณะประจำ “exec_mode”
    - ติดตั้งบนโกลบัส
4. เพียร์-ทู-เพียร์ดีเอสไอ (P2P-DSI)
    - เขียนฟังก์ชัน send และ recv ซึ่งขยายจากมอดูลกริดเอฟทีพีเซิร์ฟเวอร์
    - คอมไพล์และติดตั้งบนโกลบัส

6.3 ข้อเสนอแนะและแนวทางในงานวิจัยต่อไป

จากการพัฒนาซอฟท์แวร์ในงานวิจัย ยังมีข้อจำกัดของโปรแกรมที่น่าจะมีการพัฒนาให้ดีขึ้น รวมทั้งยังมีอีกหลายประเด็นที่อาจเป็นแนวทางในการวิจัยในอนาคต ดังนี้

1. ระบบฮาร์เวสที่นำมาใช้นี้ยังไม่ได้ใช้แบบจำลองแบบซูเปอร์เพียร์อย่างเต็มที่กล่าวคือ ระบบยังมีโพรโทคอลในการเชื่อมต่อระหว่างซูเปอร์เพียร์หลายตัว ทำให้การทดสอบทั้งหมดใช้งานซูเปอร์เพียร์เพียงตัวเดียวเท่านั้น ดังนั้นโพรโทคอลนี้จะเป็นงานที่ต้องพัฒนาและทดสอบต่อไปในอนาคต

2. เนื่องจากภาระของเครื่องเกตเวย์ที่ค่อนข้างสูง วิธีหนึ่งที่ผู้วิจัยนำเสนอคือการเพิ่มจำนวนเกตเวย์ให้กับระบบ ซึ่งวิธีนี้ช่วยลดภาระกับเกตเวย์ได้อย่างดี อย่างไรก็ตามผู้ใช้งานกริดจะมีภาระเพิ่มขึ้นในการเลือกเครื่องที่จะเข้าใช้งาน วิธีหนึ่งที่ช่วยเลือกเครื่องเกตเวย์ได้คือการใช้ตัวจัดลำดับงานระดับบน (Metascheduler)

3. สิ่งที่อาจเพิ่มเติมได้อีกส่วนหนึ่งคือการตรวจสอบสถานะของระบบ (system monitoring) เพื่อตรวจสอบจำนวนทรัพยากรที่อยู่ในระบบ, สภาพการใช้งานของระบบ, และอาจรวมถึงการเก็บสถิติต่างๆ บริการการตรวจสอบนี้อาจเชื่อมเข้ากับเอ็มดีเอส (MDS) ของโกลบัสทูลคิทได้เช่นกัน

4. การทดสอบระบบทั้งหมดในงานวิจัยนี้ทำบนแลน (LAN) ซึ่งใช้เครื่องคอมพิวเตอร์ในภาควิชาวิศวกรรมคอมพิวเตอร์ทั้งหมด ในขั้นตอนต่อไปอาจทดสอบในระดับแวน (WAN) หรืออินเทอร์เน็ต เพื่อศึกษาผลกระทบจากเครือข่ายทางกายภาพต่อการทำงานของระบบ

รายการอ้างอิง

[1] Foster, I., Kesselman, C., Nick, J.M., and Tuecke, S. (2002). The Phyology of the Grid : An Open Grid Services Architecture for Distributed Systems Integration.

[2] Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C., et al. (2003). Open Grid Services Infrastructure (OGSI)

[3] Czajkowski, K., Ferguson, D.F., Foster, I., Frey, J., Graham, S., Sedukhin, I., et al. (2004). The WS-Resource Framework.

[4] Foster, I. (2005). Globus Toolkit Version 4 : Software for Service-Oriented Systems. IFIP International Conference on Network and Parallel Computing: 2-13.

[5] Napster [Online]. Available from: http://www.napster.com/ [2008, April 1]

[6] Gnutella [Online]. Available from: http://www.napster.com/ [2008, April 1]

[7] Kazaa [Online]. Available from: http://www.kazaa.com/ [2008, April 1]

[8] BitTorrent [Online]. Available from: http://www.bittorrent.com/ [2008, April 1]

[9] Korpela, E., Werthimer, D., Anderson, D., Cobb, J., and Leboisky, M. (2001). SETI@home-massively distributed computing for SETI. Computing in Science & Engineering: 78-83.

[10] Anderson, D. (2004). BOINC: A System for Public Resource Computing and Storage. Intl Workshop on Grid Computing.

[11] Chien, A., Calder, B., Elbert, S., and Bhatia, K. (2003). Entropia: architecture and performance of an enterprise desktop grid system. Journal of Parallel and Distributed Computing 63: 597-610.

[12] Bernd O. Christiansen, P.C.M.F.I.M.O.N.K.E.S.D.W. (1997). Javelin: Internet-based parallel computing using Java. Proceedings of ACM Workshop on Java for Science and Engineering Computation: 1139-1160.

[13] Zhou, D., and Lo, V. (2004). Cluster Computing on the Fly: resource discovery in a cycle sharing peer-to-peer system. Cluster Computing and the Grid, 2004. CCGrid 2004. IEEE International Symposium.

[14] Skype [Online]. Available from: http://jxta-grid.jxta.org/ [2008, April 1]

[15] Tritrakan, K., and Muangsin, V. (2005). Using peer-to-peer communication to improve the performance of distributed computing on the Internet. Advanced

Information Networking and Applications, 2005. AINA 2005. 19th International Confer.

[16] Thanomtheeranant, K., and Maungsin, V. (2006). Integrating Peer-to-Peer File Sharing into the Grid. 10th Annual National Symposium on Computational Science & Engineering (ANSCSE10): 22-24.

[17] Globus Alliance [Online]. Available from: http://www.globus.org/ [2008, April 1]

[18] Global Grid Forum [Online]. Available from: http://www.ggf.org/ [2008, April 1]

[19] Sylvia, R., Paul, F., Mark, H., Richard, K., and Scott, S. (2001). A scalable content-addressable network. Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, San Diego, California, United States.

[20] Ion Stoicay , R.M., David Liben-Nowellz, David R. Kargerz, M. Frans Kaashoekz, Frank Dabekz,, and Balakrishnan, H. (2003). Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications.

[21] Antony, I.T.R., and Peter, D. (2001). Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg.

[22] Zhao, B.Y., Ling, H., Stribling, J., Rhea, S.C., Joseph, A.D., and Kubiatowicz, J.D. (2004). Tapestry: a resilient global-scale overlay for service deployment. Selected Areas in Communications, IEEE Journal on 22: 41-53.

[23] SungJin, C., HongSoo, K., EunJoung, B., MaengSoon, B., SungSuk, K., ChanYeol, P., et al. (2007). Characterizing and Classifying Desktop Grid. Proceedings of the Seventh IEEE International Symposium on Cluster Computing and the Grid.

[24] R., G., V., S., and A.K., S. (2006). CompuP2P: An Architecture for Internet Computing Using Peer-to-Peer Networks. Parallel and Distributed Systems: 1306 - 1320.

[25] Fedak, G., Germain, C., Neri, V., and Cappello, F. (2001). XtremWeb: a generic global computing system. Cluster Computing and the Grid, 2001. Proceedings. First IEEE/ACM International Symposium: 582-587.

[26] Litzkow, M.J., Livny, M., and Mutka, M.W. (1988). Condor-a hunter of idle workstations. 8th International Conference on Distributed Computing Systems: 104-111.

[27] JXTA(TM) Community Projects [Online]. Available from: https://jxta.dev.java.net/

[28] Kanchana, P., and Muangsin, V. (2007). JamjureeService : An API for peer-to-peer application development. Annual National Symposium on Computational Science & Engineering (ANSCSE11): 195-201.

[29] Kato, D. (2002). GISP: global information sharing protocol-a distributed index for peer-to-peer systems. Peer-to-Peer Computing, 2002. (P2P 2002). Proceedings. Second International Conference: 65 - 72.

[30] gridengine [Online]. Available from: http://gridengine.sunsource.net/ [2008, April 1]

[31] OpenPBS [Online]. Available from: http://www.pbsgridworks.com/ [2008, April 1]

[32] Platform LSF [Online]. Available from: http://www.platform.com/ [2008, April 1]

[33] Kanchana, P., and Muangsin, V. (2007). Jamjuree Explorer: A ScalableRead-Write Peer-to-Peer File System. Thai Grid Computing Conference2007 (TGCC2007), Bangkok, Thailand.

[34] GT 4.0 GridFTP: Storage Resource Broker (SRB) [Online]. Available from: http://www.globus.org/toolkit/docs/4.0/data/gridftp/GridFTP_SRB.html [2008, April 1]

[35] HPSS Module for GridFTP [Online]. Available from: http://www.hpss-collaboration.org/hpss/administrators/docs/HTML/rel6.2/GridFTPHPSS.jsp [2008, April 1]

[36] Content Management System [Online]. Available from: https://jxse-cms.dev.java.net/ [2008, April 1]

[37] Gene, M.A. (2000). Validity of the single processor approach to achieving large scale computing capabilities. Readings in computer architecture : 79-81. Morgan Kaufmann Publishers Inc.

[38] Alan, H.K., and Horace, P.F. (1990). Measuring parallel processor performance : 539-543. ACM.

[39] (2007). JXSE 2.5 Programmers Guide: Programmers Guide. Sun Microsystems, Inc.

[40] Java Service Wrapper [Online]. Available from: http://wrapper.tanukisoftware.org/ [2008, April 1]

ภาคผนวก

ภาคผนวก ก

การใช้งานจามจุรีคลัสเตอร์

จามจุรีคลัสเตอร์แบ่งเพียร์ออกเป็น 3 ประเภท ได้แก่ ซูเปอร์เพียร์, โวลันเทียร์ และเกตเวย์ รหัสต้นฉบับของเพียร์ทั้ง 3 ประเภทถูกคอมไพล์และบรรจุเป็นโปรแกรมสำเร็จในรูป jar

โวลันเทียร์และเกตเวย์แท้จริงเป็นเพียร์ประเภทเดียวกัน แต่จะทำหน้าที่ต่างกัน ดังนั้นจึงมีการตั้งค่าพารามิเตอร์และแฟ้มโครงแบบต่างกันเล็กน้อย

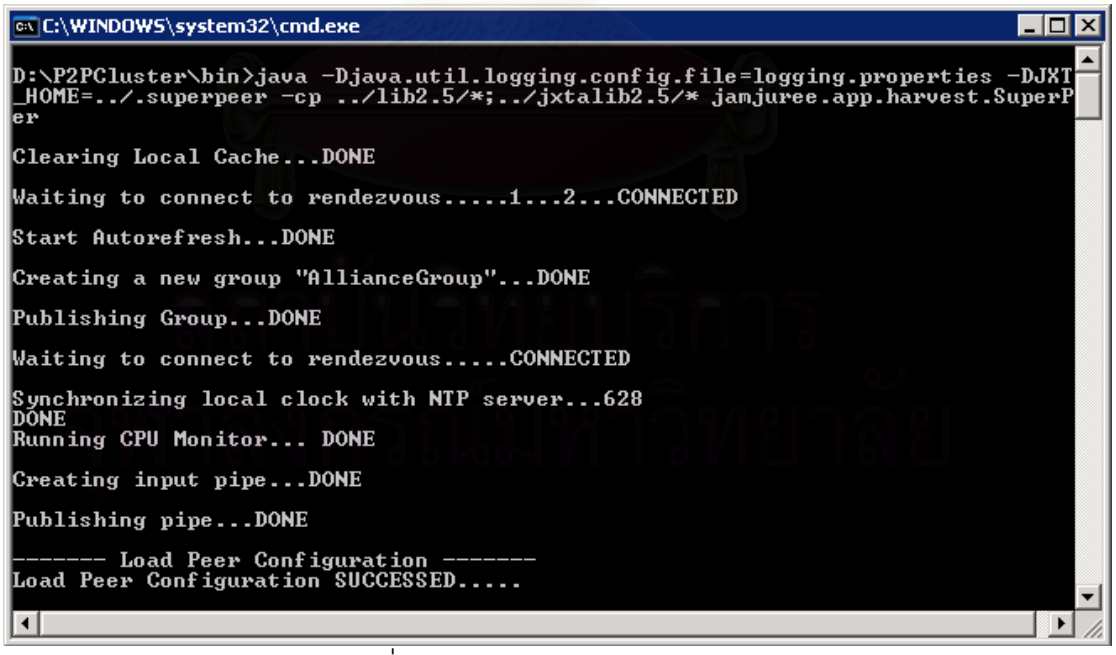
จามจุรีคลัสเตอร์ต้องการรันบนจาวารันไทม์เอ็นไวรอนเมนต์ รุ่น 1.5 ขึ้นไป

การรันโปรแกรม

การรันโปรแกรม ในที่นี้จะแยกตามประเภทของเพียร์

1. ซูเปอร์เพียร์

โปรแกรมซูเปอร์เพียร์สามารถรันด้วยส่วนต่อประสานรายคำสั่ง (command-line interface) โปรแกรมจะแสดงข้อความต่างๆทางจอเฝ้าคุม (console) ดังรูปที่ ก-1



รูปที่ ก-1 การรันโปรแกรมซูเปอร์เพียร์

ในการรันซูเปอร์เพียร์ครั้งแรก จะมีหน้าต่างตั้งค่าของจักซ์ตา ผู้ใช้สามารถศึกษาวิธีการตั้งค่าของจักซ์ตาได้จาก [39]

2. โวลันเทียร์

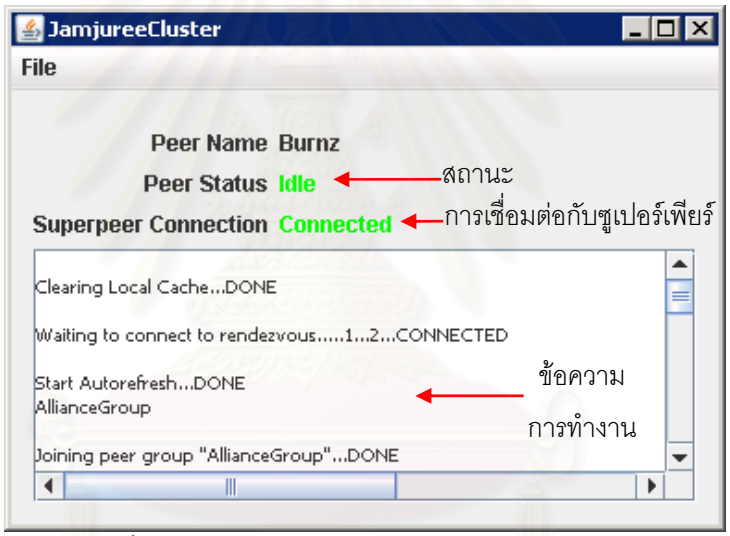
เนื่องจากโวลันเทียร์จะเป็นเพียร์ที่มีจำนวนมาก ซึ่งจะเป็นเพียร์ที่รันบนเครือข่ายอินเทอร์เน็ต ผู้วิจัยจึงใช้เทคโนโลยีจาวาเว็บสตาร์ท (Java Web Start) ซึ่งจะช่วยให้ผู้ใช้สามารถติดตั้งและรันโปรแกรมได้ง่าย ข้อดีของจาวาเว็บสตาร์ทคือ ผู้ใช้สามารถรันโปรแกรมผ่านทางอินเทอร์เน็ตได้ง่าย แฟ้มสำหรับติดตั้งเป็นแฟ้มขนาดเล็ก โดยมีชื่อว่า เจเอ็นแอลพี (JNLP: Java Network Launching Protocol) แฟ้มนี้จะเป็นคำอธิบายโปรแกรมเช่น ผู้พัฒนาโปรแกรม, เว็บไซต์ของโปรแกรม และส่วนทรัพยากรต่างๆที่รันโปรแกรม เช่น แฟ้มจาร์, คำสั่งที่ใช้รัน, พารามิเตอร์ของโปรแกรม, พารามิเตอร์ของเครื่องเสมือนจาวา (Java Virtual Machine) เป็นต้น เมื่อแฟ้มเจเอ็นแอลพีนี้ถูกรัน จาวารันไทม์จะทำการดาวน์โหลดโปรแกรมจากเว็บไซต์มายังเครื่องที่รัน เมื่อดาวน์โหลดครบก็จะเริ่มรันโปรแกรมบนเครื่อง ซึ่งจะเห็นว่ามีลักษณะคล้ายจาวาแอปเพล็ต ต่างกันที่จาวาเว็บสตาร์ทไม่ได้รันผ่านเว็บบราวเซอร์ และลดข้อจำกัดของการขออนุญาตเข้าใช้เครื่อง ข้อดีอีกอย่างหนึ่งคือ โปรแกรมจะทำการตรวจสอบรุ่นของแฟ้มเทียบกับแฟ้มที่เครื่องเซิร์ฟเวอร์ทุกครั้งที่รัน หากพบว่ามีโปรแกรมรุ่นใหม่กว่า ก็จะทำการดาวน์โหลดโปรแกรมมาใหม่ แต่หากไม่มีโปรแกรมรุ่นใหม่ก็จะเริ่มรันโปรแกรมได้เลยโดยไม่ต้องดาวน์โหลดอีก ซึ่งทำให้ผู้พัฒนามีความสะดวกในการเปลี่ยนแปลงโปรแกรมและกระจายสู่ผู้ใช้ และผู้ใช้ก็มีความสะดวกในการใช้งาน

รูปที่ ก-2 จาวาเว็บสตาร์ทดาวน์โหลดแฟ้มสำหรับรันโปรแกรมโวลันเทียร์

เมื่อรันโวลันเทียร์ด้วยแฟ้ม JamjureeCluster.jnlp โปรแกรมจะถูกดาวน์โหลดและเริ่มรัน โดยการรันครั้งแรก โปรแกรมจะให้ผู้ใช้ระบุชื่อเพียร์

รูปที่ ก-3 ผู้ใช้ระบุชื่อในการรันครั้งแรก

เมื่อกดปุ่ม “OK” โปรแกรมจะทำการตั้งค่าโครงแบบของจักซ์ตาให้โดยอัตโนมัติ ซึ่งจะเป็นค่าเริ่มต้นที่ผู้วิจัยใช้สำหรับระบบที่ติดตั้งไว้ในภาควิชาวิศวกรรมคอมพิวเตอร์ อย่างไรก็ตามผู้ใช้สามารถเปลี่ยนแปลงค่าเหล่านี้ด้วยตัวเองได้โดยกดปุ่ม “Advance Configuration” ซึ่งจะนำไปสู่การตั้งค่าโครงแบบของจักซ์ตา หลังจากที่ตั้งค่าแล้ว ข้อมูลการตั้งค่าจะถูกเก็บที่ไดเร็กทอรี “.volunteer” ซึ่งผู้ใช้ที่เชี่ยวชาญในการตั้งค่าจักซ์ตาก็สามารถเปลี่ยนแปลงแฟ้มต่างๆในไดเร็กทอรีนี้ได้ในภายหลังเช่นกัน



รูปที่ ก-4 หน้าต่างแสดงการทำงานของโวลันเทียร์

โปรแกรมโวลันเทียร์จะมีหน้าต่างขึ้นมา และจะแสดงข้อมูล ชื่อเพียร์, สถานะ (“Idle”, “Executing”), การเชื่อมต่อกับซูเปอร์เพียร์ (“Connected”, “Not Connected”), และข้อความแสดงการทำงานต่างๆของโปรแกรมจะแสดงในพื้นที่ข้อความ

3. เกตเวย์

โปรแกรมเกตเวย์จะถูกติดตั้งเป็นดีมอน (daemon) บนระบบปฏิบัติการลินุกซ์ โดยใช้เทคโนโลยีจาวาเซอร์วิสแรปเปอร์ (Java Service Wrapper) [40] ซึ่งจะทำให้โปรแกรมจาวาสามารถรันเป็นการประมวลผลส่วนหลัง (background process) ได้ และจะรันโดยใช้สิทธิ์เป็น root

โปรแกรมจะวางตามโครงสร้างแฟ้มดังนี้

- ($GRID_P2P_HOME)/bin/ เป็นที่อยู่ของแฟ้มต่างๆที่ใช้รันโปรแกรม
- ($GRID_P2P_HOME)/conf/ เป็นที่อยู่ของแฟ้มโครงแบบต่างๆ
- ($GRID_P2P_HOME)/jxtalib/ เป็นที่อยู่ของแฟ้มคลังคำสั่ง (library) ของจักซ์ตา
- ($GRID_P2P_HOME)/lib/ เป็นที่อยู่ของแฟ้มคลังคำสั่งของโปรแกรมฮาร์เวสและจาวาเซอร์วิสแรปเปอร์

โดยที่ ($GRID_P2P_HOME) เป็นไดเร็กทอรีที่อยู่ของโปรแกรมโวลันเทียร์ทั้งหมด แฟ้มที่สำคัญในการตั้งค่าโปรแกรมได้แก่

- ($GRID_P2P_HOME)/bin/run-Volunteer เป็นแฟ้มที่รันได้ (executable) เป็นแฟ้มหลักที่ใช้เริ่มรันโปรแกรมโวลันเทียร์ให้เป็นดีมอน แฟ้มนี้แก้ไขจาก sh.script.in ที่มีให้ในชุดโปรแกรมของจาวาเซอร์วิสแรปเปอร์ โดยเปลี่ยนค่า APP_NAME, APP_LONG_NAME, WRAPPER_CMD, WRAPPER_CONF ดังรูปที่ ก-5

```
# Application
APP_NAME="Volunteer"
APP_LONG_NAME="Jamjuree Cluster-Volunteer"

# Wrapper
WRAPPER_CMD="./wrapper"
WRAPPER_CONF="../conf/wrapper-Vol.conf"
```

รูปที่ ก-5 การตั้งค่าแฟ้ม run-Volunteer

- ($GRID_P2P_HOME)/conf/jc-Vol.conf เป็นแฟ้มสำหรับตั้งค่าโปรแกรมโวลันเทียร์ โดยมีรูปแบบดังรูปที่ ก-6

```
machineCfg=../conf/machine.cfg
jobStat=../conf/jobStat.jsf
jobSub=../conf/jobSub.txt
execMode=../conf/execmode.properties
VolInputPipeAdv=../conf/VolInputPipeAdv.ADV
jxCfg=../conf/JXConfig.xml
jxLog=../conf/jxlog.txt
log=../conf/log.txt
```

รูปที่ ก-6 การตั้งค่าแฟ้ม jc-Vol.conf

- ($GRID_P2P_HOME)/conf/wrapper-Vol.conf เป็นแฟ้มสำหรับตั้งค่าต่างๆของจาวาเซอร์วิสแรปเปอร์ ประกอบด้วยการตั้งค่า classpath, library path, parameter, log โดยตั้งค่าดังรูปที่ ก-7

```
# Java Main class.  This class must implement the WrapperListener interface
#  or guarantee that the WrapperManager class is initialized.  Helper
#  classes are provided to do this for you.  See the Integration section
#  of the documentation for details.
wrapper.java.mainclass=org.tanukisoftware.wrapper.WrapperSimpleApp

# Java Classpath (include wrapper.jar)  Add class path elements as needed
wrapper.java.classpath.1=../lib/wrapper.jar
wrapper.java.classpath.2=../jxtalib/swixml.jar
wrapper.java.classpath.3=../lib/JamjureeService.jar
wrapper.java.classpath.4=../jxtalib/org.mortbay.jetty.jar
wrapper.java.classpath.5=../jxtalib/log4j.jar
wrapper.java.classpath.6=../jxtalib/jxtaext.jar
wrapper.java.classpath.7=../jxtalib/jxtacms.jar
wrapper.java.classpath.8=../jxtalib/jxta.jar
wrapper.java.classpath.9=../jxtalib/jdom.jar
wrapper.java.classpath.10=../jxtalib/javax.servlet.jar
wrapper.java.classpath.11=../jxtalib/bcprov-jdk14.jar
wrapper.java.classpath.12=../lib/dom4j.jar
wrapper.java.classpath.13=../lib/p2pfile.jar
wrapper.java.classpath.14=../lib/n_gisp_r5.jar
wrapper.java.classpath.15=../lib/harvest.jar
wrapper.java.classpath.16=../lib/conf.jar

# Java Library Path (location of Wrapper.DLL or libwrapper.so)
wrapper.java.library.path.1=../lib

# Java Additional Parameters
wrapper.java.additional.1=-DJXTA_HOME=../Volunteer

...

# Application parameters.  Add parameters as needed starting from 1
wrapper.app.parameter.1=jamjuree.app.harvest.Volunteer
wrapper.app.parameter.2=-conf=../conf/jc-Vol.conf
wrapper.app.parameter.3=-gateway
wrapper.app.parameter.4=-nogui
wrapper.app.parameter.5=-name=GW-evasion
...

# Log file to use for wrapper output logging.
wrapper.logfile=../logs/vol.log
```

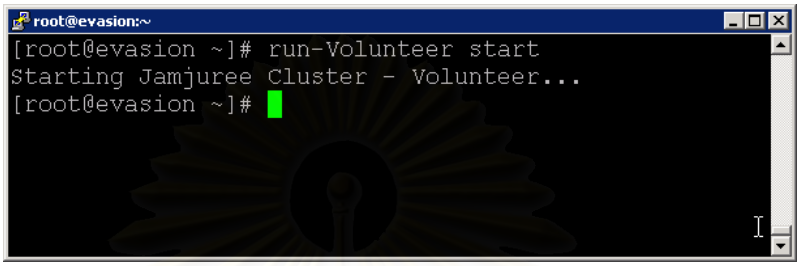
รูปที่ ก-7 การตั้งค่าแฟ้ม wrapper-Vol.conf

หลังจากตั้งค่าต่างๆของโปรแกรมเรียบร้อยแล้ว ควรจะตั้ง path ของโปรแกรมให้ชี้มายัง ($GRID_P2P_HOME) ด้วย โดยเขียนแฟ้ม /etc/profile.d/grid-p2p.sh ดังนี้

```
export GRID_P2P_HOME=/usr/grid-p2p
export PATH=$GRID_P2P_HOME/bin:$PATH
```

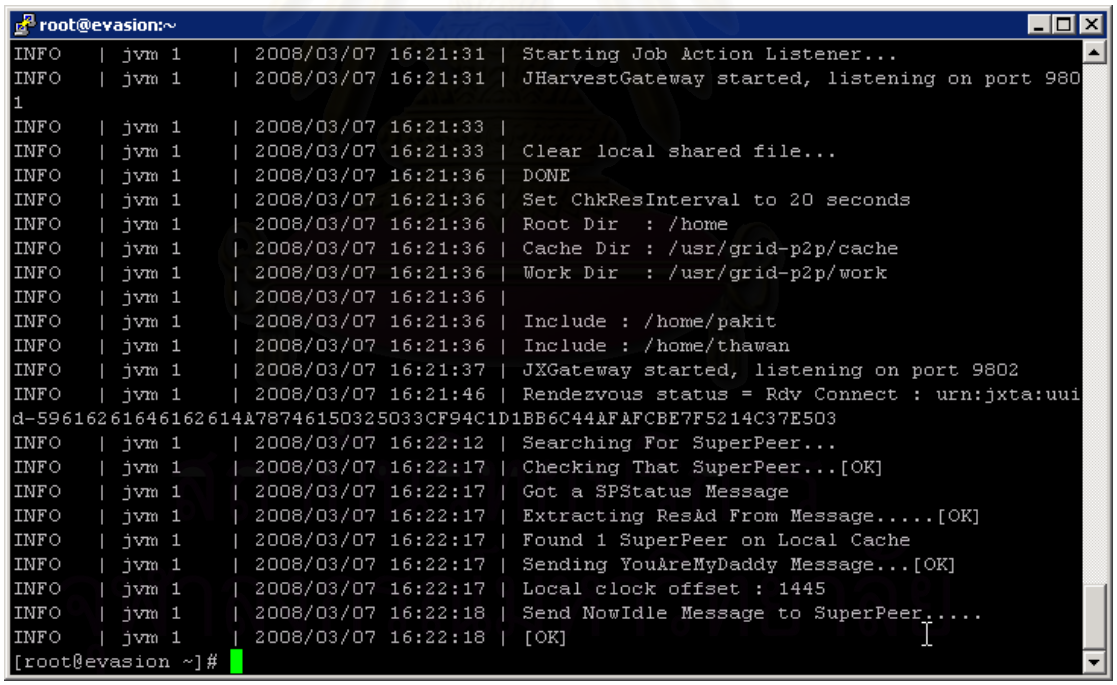
รูปที่ ก-8 การเขียนแฟ้ม /etc/profile.d/grid-p2p.sh

จากนั้นทดสอบรันโปรแกรมโดยเรียกคำสั่ง run-Volunteer start



รูปที่ ก-9 รันโปรแกรมโวลันเทียร์เป็นดีมอน

โปรแกรมจะทำงานเป็นพื้นหลัง ซึ่งเราสามารถตรวจสอบการทำงานได้โดยดูจากแฟ้ม log.txt



รูปที่ ก-10 ตรวจสอบการทำงานจากแฟ้ม log.txt

การตั้งค่า (Configuration)

แฟ้มโครงแบบ (configuration file) ของจามจุรีคลัสเตอร์ มีดังนี้

1) machine.cfg

เป็นแฟ้มที่บอกข้อมูลของฮาร์ดแวร์, ซอฟท์แวร์, เครือข่าย, และเวลาให้บริการต่างๆของเครื่อง machine.cfg จะใช้สำหรับโวลันเทียร์ เกตเวย์ และซูเปอร์เพียร์ โดยแฟ้มนี้จะถูกอ่านตอนเริ่มโปรแกรม ตัวอย่างของแฟ้มแสดงดัง รายละเอียดการตั้งค่าต่างๆ ศึกษาได้จากงานวิจัย [ ] โดยจะมีลักษณะประจำที่เพิ่มเติมขึ้นมาคือ execMode ซึ่งจะบอกถึงประเภทของงานที่รันได้

```
owner=Burnz
arch=INTEL
speed=1500
os=WINDOWS
memory=256
disk=1000
bandwidth=1024
program=jvm,dos
busyTime=8.30-16.00
execMode=dos_batch,dos_cmd
```

รูปที่ ก-11 แฟ้ม machine.cfg

2) JXConfig.xml

เป็นแฟ้มสำหรับตั้งค่าการทำงานของจามจุรีเอ็กซ์พลอเรอร์เช่น ไดเร็กทอรีราก, ไดเร็กทอรีแคช, ไดเร็กทอรีทำงาน, ชื่อเพียร์, ตัวเลือกการแบ่งปัน เป็นต้น ตัวอย่างของแฟ้ม JXConfig.xml แสดงดังรูปที่ ก-12 มีลักษณะประจำดังนี้

- <Root/> เป็นการตั้งค่าไดเร็กทอรีรากซึ่งเป็นที่แบ่งปันข้อมูลหลัก โดยระบุชื่อไดเร็กทอรีไว้ในแท็ก <RootDir/> สำหรับเกตเวย์ <RootDir/> จะต้องตั้งค่าเป็น /home
- <Cache/> เป็นการตั้งค่าไดเร็กทอรีแคชซึ่งเป็นที่เก็บแฟ้มข้อมูลชั่วคราวของเครือข่าย มีแท็ก <CacheDir/> สำหรับระบุชื่อไดเร็กทอรีแคช และ <MaxSpace/> สำหรับระบุพื้นที่มากสุดที่จะรองรับแฟ้มข้อมูลได้
- <Work/> เป็นไดเร็กทอรีทำงานของฮาร์เวส ใช้สำหรับเป็นที่พักข้อมูลและรันโปรแกรมของฮาร์เวส มีแท็ก <WorkDir/> สำหรับระบุชื่อไดเร็กทอรีทำงาน และ <MaxSpace/> สำหรับระบุพื้นที่ว่างมากสุดสำหรับรันงาน

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE jxConfig>
<jxConfig>
        <Root>
                <RootDir>/home</RootDir>
        </Root>
        <Cache>
                <CacheDir>cache</CacheDir>
                <MaxSpace>100</MaxSpace>
        </Cache>
        <WorkSpace>
                <WorkDir>work</WorkDir>
                <MaxSpace>200</MaxSpace>
        </WorkSpace>
        <Peers>
                <Peer>
                        <Name>user01</Name>
                        <Exclude>dir01</Exclude>
                </Peer>
                <Peer>
                        <Name>user02</Name>
                        <Include>dir02</Exclude>
                        <Exclude>dir02/dir03</Exclude>
                </Peer>
        </Peers>
        <ShareOptions>
                <ShareInterval>600</ShareInterval>
                <ItemExpireTime>600</ItemExpireTime>
                <SearchRetry>10</SearchRetry>
                <SearchTimeOut>3</SearchTimeOut>
                <DownloadRetry>3</DownloadRetry>
                <DownloadTimeOut>30</DownloadTimeOut>
                <Replication>true</Replication>
                <ChkResInterval>20</ChkResInterval>
        </ShareOptions>
</jxConfig>
```

รูปที่ ก-12 แฟ้มโครงแบบ JXConfig.xml

- <Peers/> แท็กนี้จะใช้เฉพาะที่เครื่องเกตเวย์เท่านั้น เป็นการกำหนดชื่อผู้ใช้ที่จะแบ่งปันแฟ้มในเครือข่ายได้ สามารถระบุผู้ใช้ได้หลายคนโดยแต่ละคนจะระบุในแท็ก <Peer/>ในแท็ก <Peer/> ประกอบด้วย <Name/> ซึ่งระบุชื่อผู้ใช้, <Include/> ระบุไดเร็กทอรีย่อยที่จะร่วมแบ่งปันในเครือข่าย และ <Exclude/> ระบุไดเร็กทอรีย่อยที่จะไม่ร่วมแบ่งปันในเครือข่าย
- <ShareOptions/> เป็นตัวเลือกต่างๆที่เกี่ยวข้องกับการจัดการแฟ้มข้อมูล ได้แก่ <ShareInterval/> เป็นช่วงเวลาในการแบ่งปันข้อมูลในไดเร็กทอรีรากแต่ละครั้ง (วินาที), <ItemExpireTime/> เป็นช่วงอายุของข้อมูลแฟ้มที่กระจายในเครือข่าย (วินาที), <SearchRetry/> เป็นจำนวนครั้งในการค้นหาข้อมูลแฟ้มใหม่หากผิดพลาด, <SearchTimeOut/> เป็นเวลารอในการค้นหาข้อมูลแฟ้ม (วินาที), <DownloadRetry/> เป็นจำนวนครั้งในการดาวน์โหลด

แฟ้มใหม่หากผิดพลาด, <DownloadTimeOut/> เป็นเวลารอในการดาวน์โหลดแฟ้มเมื่อไม่มีความก้าวหน้า (วินาที), <Replication/> เป็นการระบุว่าจะมีการรับฝากแฟ้มจากเครือข่ายหรือไม่, <ChkResInterval/> เป็นช่วงเวลาในการตรวจสอบความรับผิดชอบแคชของตัวเอง (วินาที)

3) execmode.properties

เป็นแฟ้มที่ใช้กำหนดแบบวิธีกระทำการต่างๆ โดยจะเป็นคู่ลักษณะประจำของแบบวิธีกระทำการกับคลาสที่ใช้รัน คลาสเหล่านี้จะถูกโหลดด้วยคลาสโหลดเดอร์ (Class loader) ในเวลารัน (runtime)

```
dos_batch=jamjuree.app.harvest.executor.DosBatchExecutor
dos_cmd=jamjuree.app.harvest.executor.DosCommandExecutor
dos_perl=jamjuree.app.harvest.executor.DosPerlExecutor
unix_sh=jamjuree.app.harvest.executor.CygwinShellScriptExecutor
unix_cmd=jamjuree.app.harvest.executor.CygwinCommandExecutor
default=jamjuree.app.harvest.executor.DosCommandExecutor
```

รูปที่ ก-13 แฟ้ม execmode.properties

execmode.properties ช่วยให้สามารถพัฒนาตัวกระทำการ (executor) แบบใหม่ได้ ซึ่งเป็นคลาสจาวาที่ขยายจากคลาส jamjuree.app.harvest.executor.Executor โดยอาจจะเป็นตัวกระทำการที่ออกแบบมาเฉพาะสำหรับงานนั้นๆ

การติดตั้งตัวจัดการงานเพียร์-ทู-เพียร์

แฟ้มที่ใช้ติดตั้งตัวจัดการงานเพียร์-ทู-เพียร์ประกอบด้วย 2 แฟ้ม คือ p2p.pm ซึ่งเป็นบทคำสั่งเพิร์ล (perl script) และ p2p.rvf ซึ่งเป็นแฟ้มที่กำหนดลักษณะประจำเพิ่มเติม การติดตั้งตัวจัดการงาน ทำได้โดยการนำแฟ้มทั้ง 2 แฟ้มไปวางยังที่ต่างๆดังนี้

```
$ cp p2p.pm $GLOBUS_LOCATION/lib/perl/Globus/GRAM/JobManager/p2p.pm
$ cp p2p.rvf $GLOBUS_LOCATION/share/globus_gram_job_manager/p2p.rvf
```

หลังจากนั้น ทำการเรจิสเตอร์แฟ้ม p2p.pm นี้เข้ากับตัวจัดการงาน โดยใช้ชื่อตัวจัดการงานนี้ว่า jobmanager-p2p

```
$ $GLOBUS_LOCATION/libexec/globus-job-manager-service -add -m p2p -s
jobmanager-p2p
```

ส่วน p2p.rvf นั้น ตัวจัดการงานเพียร์-ทู-เพียร์ jobmanager-p2p จะมาเรียกใช้เอง ไม่ต้องมีการเรจิสเตอร์ใดๆ

หลังจากที่เราเรจิสเตอร์ตัวจัดการงานเพียร์-ทู-เพียร์เข้ากับระบบแล้ว เราสามารถแก้ไขแฟ้ม p2p.pm ได้เรื่อย โดยที่ไม่ต้องเรจิสเตอร์กับระบบใหม่

การถอนเรจิสเตอร์ ใช้คำสั่ง

```
$ $GLOBUS_LOCATION/libexec/globus-job-manager-service -remove
-s jobmanager-p2p
```

ตัวจัดการงานที่เรจิสเตอร์จะปรากฏชื่ออยู่ใน $GLOBUS_LOCATION/etc/grid-services/ ผู้สั่งงานจากกริดจะสามารถเข้าถึงตัวจัดการงานได้ตามรายชื่อแฟ้มที่ไดเร็กทอรีนี้

```
$ ls -l $GLOBUS_LOCATION/etc/grid-services
total 16
lrwxrwxrwx 1 root root  15 May  4  2007 jobmanager -> jobmanager-fork
-rw-r--r-- 1 root root 197 Aug  7  2007 jobmanager-ccss
-rw-r--r-- 1 root root 197 May 18  2007 jobmanager-fork
-rw-r--r-- 1 root root 195 May 21  2007 jobmanager-p2p
-rw-r--r-- 1 root root 195 May  4  2007 jobmanager-sge
```

และหากต้องการให้ตัวจัดการงานที่เรจิสเตอร์นี้เป็นค่าปริยาย (default) ของระบบ ให้ทำการสร้างหรือเปลี่ยนลิงค์จาก jobmanager มายังตัวจัดการงานที่ต้องการโดยใช้คำสั่ง ln

การติดตั้งเพียร์-ทู-เพียร์ดีเอสไอ

ชุดโปรแกรมสำหรับติดตั้งดีเอสไอประกอบด้วยแฟ้ม Makefile.in เป็นแฟ้มต้นฉบับสำหรับสร้างแฟ้ม Makefile, globus_gridftp_server_dsi.c.in เป็นแฟ้มต้นฉบับของรหัสต้นฉบับ (source code), และแฟ้ม generate-stubs.sh ใช้สำหรับสร้าง Makefile

globus_gridftp_server_dsi.c.in ถูกแก้ไขเป็น globus_gridftp_server_p2p.c จากนั้นสั่งคำสั่ง generate-stubs.sh โดยระบุชื่อดีเอสไอและ flavor จะสร้าง Makefile และ makefile_header ออกมา

```
$ ./generate-stubs.sh p2p gcc32
```

จากนั้นเรียกคำสั่ง make และ make install เพื่อคอมไพล์และติดตั้ง

```
$ make
/usr/bin/gcc -fPIC -O -Wall -I/opt/globus/include/gcc32 \
        -shared -o libglobus_gridftp_server_p2p_gcc32.so \
        globus_gridftp_server_p2p.c \
        -L/opt/globus/lib -L/opt/globus/lib
$ make install
cp -f libglobus_gridftp_server_p2p_gcc32.so /opt/globus/lib
```

จากนั้นต้องเรจิสเตอร์เข้ากับระบบเพื่อให้ทำงานได้ โดยใช้คำสั่ง globus_gridftp_server

```
$ $GLOBUS_LOCATION/sbin/globus-gridftp-server -p <port> -dsi
srb -auth-level 4
```

โดยกำหนด <port> ของ server ตามต้องการ หากไม่กำหนดจะใช้พอร์ทมาตรฐานคือ 2811

การสั่งงานบนโกลบัส

การสั่งงานด้วยโกลบัสทำได้โดยใช้ส่วนต่อประสานรายคำสั่ง (command-line interface) โดยคำสั่งที่ใช้ในการสั่งงานได้แก่

1) globus-job-run ใช้สำหรับสั่งงานเชิงโต้ตอบ (interactive job) เมื่อสั่งงานด้วยคำสั่งนี้จะส่งงานไปยังเครื่องปลายทางและรอจนกว่าผลลัพธ์จะเสร็จสิ้นและแสดงผลลัพธ์จาก standard output ออกมา

2) globus-job-submit ใช้สำหรับสั่งงานแบบกลุ่ม (batch job) เมื่อสั่งงานด้วยคำสั่งนี้จะส่งงานไปยังเครื่องปลายทางและคืนค่า Job ID ออกมา ซึ่งจะใช้ในการตรวจสอบสถานะของงาน และรับผลลัพธ์กลับมา

3) globusrun เป็นคำสั่งพื้นฐานของ globus-job-run และ globus-job-submit จึงสามารถสั่งงานได้ทั้งแบบเชิงโต้ตอบและแบบกลุ่ม การสั่งงานด้วย globusrun จะต้องเขียนแฟ้มคำอธิบายงานเป็นอาร์เอสแอล

- **การสั่งงานด้วย** globus-job-run

การสั่งงานด้วยคำสั่ง globus-job-run มีรูปแบบดังนี้

globus-job-run <contact string> [-np N] <executable> [<arg>…]

- contact string คือเครื่องปลายทาง (พอร์ท) และตัวจัดการงาน เช่น host, host:2119, host/jobmanager-sge เป็นต้น
- np เป็นการระบุจำนวนงานที่จะรัน
- executable คือโปรแกรมที่จะรัน
- arg คืออาร์กิวเมนต์ของโปรแกรมที่จะรัน

ตัวอย่างการสั่งงานด้วย globus-job-run

```
$ globus-job-run evasion.cp.eng.chula.ac.th /bin/hostname
```

เป็นการรันคำสั่ง hostname ที่เครื่อง evasion.cp.eng.chula.ac.th

การใช้คำสั่ง globus-job-run สามารถศึกษารายละเอียดเพิ่มเติมได้โดยสั่ง globus-job-run -help

ในการสั่งงานไปยังฮาร์เวสนั้น ให้ระบุตัวจัดการงานเป็น jobmanager-p2p ที่ contact string ด้วย เช่น evasion.cp.eng.chula.ac.th/jobmanager-p2p นอกจากนี้ยังต้องระบุ execution mode เพิ่มเติม โดยเพิ่มคำสั่ง –x="(execmode=<*execmode*>)" ตัวอย่างการสั่งงาน เช่น

```
$ globus-job-run evasion.cp.eng.chula.ac.th/jobmanager-p2p -x
"(execmode= unix_cmd)" /bin/hostname
```

เราสามารถส่งแฟ้ม executable ที่จะรันไปพร้อมๆกับการสั่ง globus-job-run ได้ โดยใช้ –s <executable> เช่น

```
$ globus-job-run evasion.cp.eng.chula.ac.th/jobmanager-p2p -x
"(execmode=dos_batch)" –s mandelbrot.bat
```

แฟ้ม mandelbrot.bat จะถูกส่งไปยังเครื่อง evasion และรันด้วยฮาร์เวสได้

- **การสั่งงานด้วย** globus-job-submit

การสั่งงานด้วย globus-job-submit นั้นใช้รูปแบบคำสั่งเหมือนกับ globus-job-run ต่างกันที่เมื่อสั่ง globus-job-submit จะคืนค่า Job ID ออกมา ตัวอย่างเช่น

```
$ globus-job-submit evasion.cp.eng.chula.ac.th/jobmanager-p2p
-x "(execmode= dos_batch)" –s mandelbrot.bat
https://evasion.cp.eng.chula.ac.th:57387/10020/1204999301/
```

จะเห็นว่า https://evasion.cp.eng.chula.ac.th:57387/10020/1204999301/ คือค่า Job ID ซึ่งจะใช้ในการเก็บผลลัพธ์กลับในภายหลัง โดยใช้คำสั่ง globus-job-get-output

```
$ globus-job-get-output
https://evasion.cp.eng.chula.ac.th:57387/10020/1204999301/
Producing sample output ....
Number of pixel : 800000
Number of iteration : 2000
Computing : 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Computation time : 22922 ms.
Write to : mandelbrot.png
Image complete
```

ผลลัพธ์ในที่นี้คือ standard output และ standard error จากการรันโปรแกรม

- **การสั่งงานด้วย** globusrun

globusrun เป็นคำสั่งพื้นฐานของ globus-job-submit และ globus-job-run การสั่งงานด้วย globusrun จะใช้แฟ้มอาร์เอสแอล ตัวอย่างของอาร์เอสแอล mandelbrot.rsl

```
&(executable=./mandelbrot.bat)
 (arguments=1600 1200 2000 -2 1 -1 1 mandelbrot.png 1)
 (file_stage_in=($(GLOBUSRUN_GASS_URL)/home/pakit/jh-testapp/
      mandelbrot.bat /home/pakit/mandelbrot.bat)
      $(GLOBUSRUN_GASS_URL)/home/pakit/jh-testapp/
      mandelbrot.jar /home/pakit/mandelbrot.jar))
 (file_stage_out=(/home/pakit/ mandelbrot.png
      $(GLOBUSRUN_GASS_URL)/home/pakit/jh-testapp/))
 (execmode=dos_batch)
```

เอกสารอาร์เอสแอลจะอยู่ในรูปคู่ลักษณะประจำ โดยเราสามารถกำหนดแฟ้มเข้าและแฟ้มออกได้โดยใช้ลักษณะประจำ file_stage_in และ file_stage_out ตามลำดับ และกำหนดลักษณะประจำ execmode เพื่อบอกประเภทของ execution mode ให้กับฮาร์เวส

การสั่งงานด้วย globusrun มีรูปแบบดังนี้

```
globusrun [options] [RSL String]
```

ตัวเลือกที่ใช้ได้แก่

-f <rsl filename> หรือ -file <rsl filename> เป็นการระบุแฟ้มอาร์เอสแอล

-r <resource manager>หรือ –resource <resource manager> เป็นการกำหนดเครื่องและตัวจัดการงาน (เช่นเดียวกับ contact string ในคำสั่ง globus-job-run และ globus-job-submit)

-s หรือ -server เป็นการเปิด GASS server เพื่อให้ใช้แท็ก ($(GLOBUSRUN_GASS_URL) ได้ในแฟ้มอาร์เอสแอล ซึ่ง $(GLOBUSRUN_GASS_URL) จะเป็นการบอกที่อยู่ของเครื่องที่สั่งงาน มักใช้กับลักษณะประจำ file_stage_in และ file_stage_out โดยใช้ในการถ่ายโอนแฟ้มเข้าและแฟ้มออกจากเครื่องที่สั่งงาน GASS server ใช้โพรโทคอล HTTPS ในการถ่ายโอนข้อมูล

ตัวเลือกอื่นๆ สามารถศึกษาได้จากคำสั่ง globusrun –help

ตัวอย่างการสั่งงานด้วย globusrun

```
$ globusrun -r evasion.cp.eng.chula.ac.th/jobmanager-p2p -f
mandelbrot.rsl –s
Number of pixel : 1920000
Number of iteration : 2000
Computing : 10% 20% 30% 40% 50% 60% 70% 80% 90% 100%
Computation time : 29532 ms.
Write to : mandelbrot.png
Image complete
$
```

การถ่ายโอนแฟ้มด้วยกริดเอฟทีพี

การถ่ายโอนแฟ้มด้วยกริดเอฟทีพีบนโกลบัส ใช้คำสั่ง globus-url-copy โดยมีรูปแบบคำสั่งดังนี้

```
globus-url-copy [options] <sourceURL> <destURL>
```

โดยที่ sourceURL คือแฟ้มต้นทาง และ destURL คือแฟ้มปลายทาง ทั้ง sourceURL และ destURL สนับสนุนโพรโทคอล file:///, http://, https://, ftp://, gsiftp:// ตัวอย่างการใช้งานเช่น

```
$ globus-url-copy file:///home/user01/sum.sh gsiftp://evasion
.cp.eng.chula.ac.th/home/user01/sum.sh
```

ในการถ่ายโอนแฟ้มจากหรือสู่กริดเอฟทีพีเซิร์ฟเวอร์ที่ใช้เพียร์-ทู-เพียร์ดีเอสไอ ใช้คำสั่งเดียวกันทุกประการ เว้นแต่จะใช้พอร์ทที่เปิดบริการต่างออกไป ตัวอย่างของการกำหนดพอร์ท เป็น 5000

```
$ globus-url-copy file:///home/user01/sum.sh gsiftp://evasion
.cp.eng.chula.ac.th:5000/home/user01/sum.sh
```

ภาคผนวก ข

ผลงานที่ได้รับการตีพิมพ์

งานส่วนหนึ่งของวิทยานิพนธ์ที่ได้รับการตีพิมพ์เป็นบทความทางวิชาการจำนวน 3 ฉบับ ดังนี้

1. หัวข้อ “Jamjuree Explorer: A Scalable Read-Write Peer-to-Peer File System” โดย นายปกิต กาญจนะ และ อ.ดร.วีระ เหมืองสิน ในงานประชุมวิชาการ “Thai Grid Computing Conference (TGCC2007)” จัดที่โรงแรมรามาการ์เด้นส์ กรุงเทพมหานคร ในวันที่ 23-24 สิงหาคม 2550

2. หัวข้อ “Jamjuree Cluster: A Peer-to-Peer Cluster Computing System” โดย นายเกษม ตริตระการ, นายปกิต กาญจนะ และ อ.ดร. วีระ เหมืองสิน ในงานประชุมวิชาการ “The 1st International Conference on Network-Based Information Systems (NBiS2007)” จัดที่เมืองเรเกนสเบิร์ก ประเทศเยอรมนี ในวันที่ 3 – 7 กันยายน 2550

3. หัวข้อ “Integrating Peer-to-Peer Resources into the Grid” โดย นายปกิต กาญจนะ และ อ.ดร.วีระ เหมืองสิน ในงานประชุมวิชาการ “The 2nd International Conference on Advances Information Technology (IAIT2007)” จัดที่โรงแรมเอเชีย กรุงเทพมหานคร ในวันที่ 1-2 พฤศจิกายน 2550

ประวัติผู้เขียนวิทยานิพนธ์

นายปกิต กาญจนะ เกิดวันที่ 9 มีนาคม พ.ศ.2527 ในจังหวัดปัตตานี ปัจจุบันย้ายภูมิลำเนามาอยู่ที่จังหวัดสตูล สำเร็จการศึกษาในหลักสูตรวิศวกรรมศาสตรบัณฑิต จากภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย และเข้าศึกษาต่อในหลักสูตรวิศวกรรมศาสตรมหาบัณฑิต ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2549